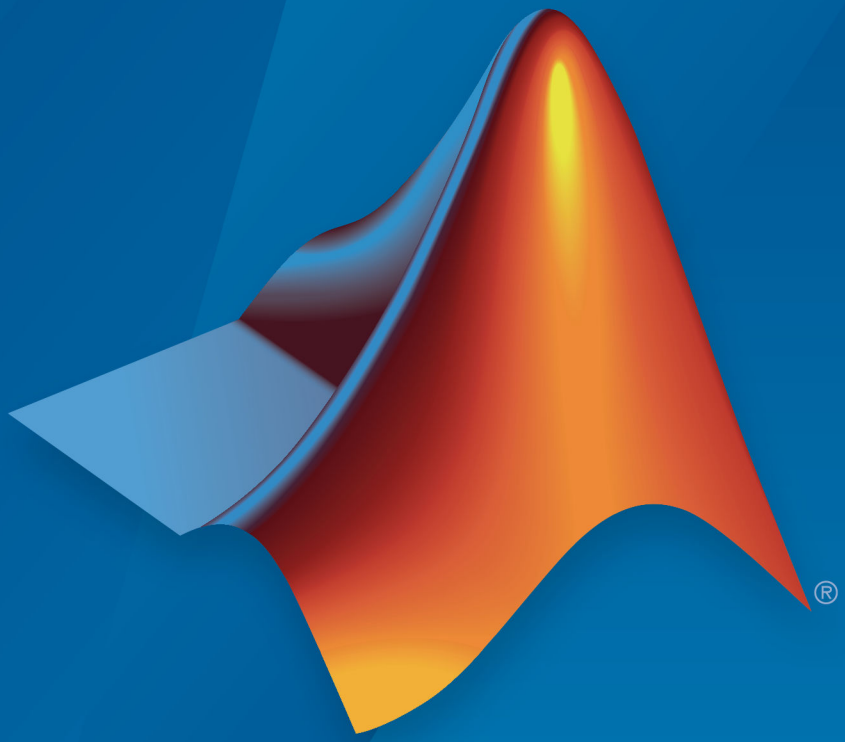
Antenna Toolbox™

Reference



MATLAB®

R2018b

MathWorks®

How to Contact MathWorks

Latest news: www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

Technical support: www.mathworks.com/support/contact_us

Phone: 508-647-7000

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

Antenna Toolbox™ Reference

Revision History

March 2015	Online only	New for Version 1.0 (R2015a)
September 2015	Online only	Revised for Version 1.1 (R2015b)
March 2016	Online only	Revised for Version 2.0 (R2016a)
September 2016	Online only	Revised for Version 2.1 (R2016b)
March 2017	Online only	Revised for Version 2.2 (R2017a)
September 2017	Online only	Revised for Version 3.0 (R2017b)
March 2018	Online only	Revised for Version 3.1 (R2018a)
September 2018	Online only	Revised for Version 3.2 (R2018b)

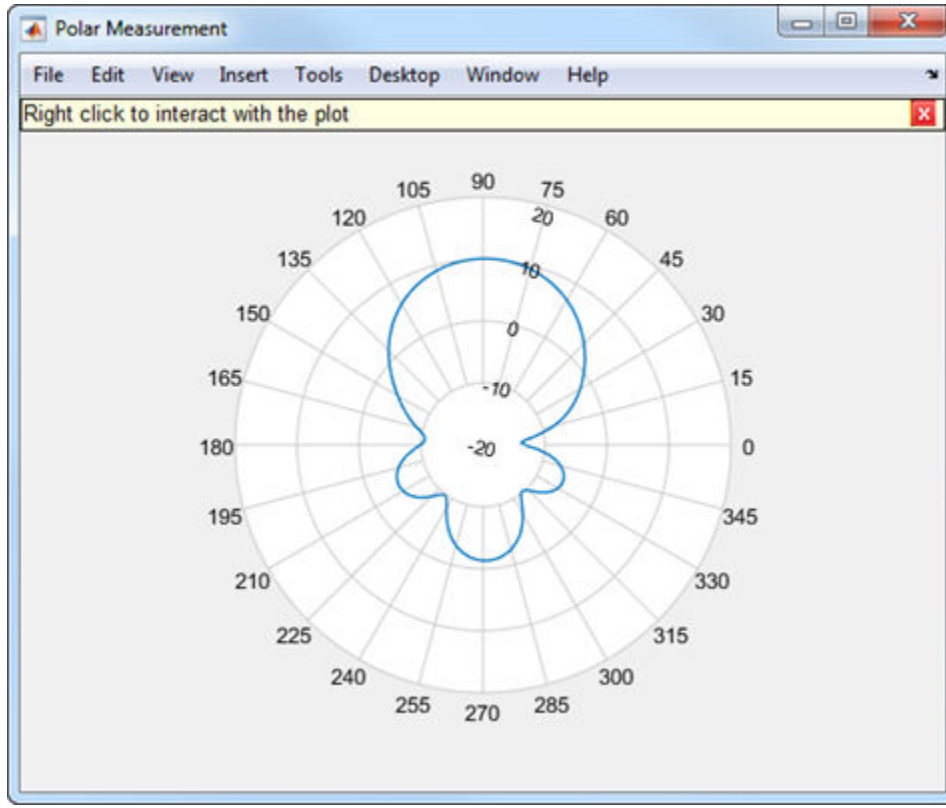
Contents

Antenna Classes — Alphabetical List

polarpattern class

Interactive plot of radiation patterns in polar format

Description



`polarpattern` class plots antenna or array radiation patterns in interactive polar format. You can also plot other types of polar data. Use these plots when interactive data visualization or measurement is required. Right-click the **Polar Measurement** window to change the properties, zoom in, or add more data to the plot.

Construction

`polarpattern` plots antenna or array radiation patterns and other types of data in polar format. `polarpattern` plots field value data of radiation patterns for visualization and measurement. Right-click the polar plot to interact.

`polarpattern(data)` creates a polar plot with magnitude values in the vector `d`. In this polar plot, angles are uniformly spaced on the unit circle, starting at `0` degrees.

`polarpattern(angle,magnitude)` creates a polar plot from a set of angle vectors and corresponding magnitudes. You can also create polar plots from multiple sets for angle vectors and corresponding sets of magnitude using the syntax: `polarpattern(angle1, magnitude1, angle2, magnitude2...)`.

`p = polarpattern( ___ )` returns an object handle that you can use to customize the plot or add measurements. You can specify any of the arguments from the previous syntaxes.

`p = polarpattern('gco')` returns an object handle from polar pattern in the current figure.

`polarpattern( ___ ,Name,Value)` creates a polar plot, with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding property value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values. To list all the property `Name,Value` pairs, use `details(p)`. To list all the property `Name,Value` pairs, use `details(p)`. You can use the properties to extract any data from the radiation pattern from the polar plot. For example, `p = polarpattern(data,'Peaks',3)` identifies and displays the three highest peaks in the pattern data.

For a list of properties, see PolarPattern.

`polarpattern(ax, ___ )` creates a polar plot using axes handle, `ax` instead of the current axes handle.

Input Arguments

data — Antenna or array data

real length-*M* vector | real *M*-by-*N* matrix | real *N*-*D* array | complex vector or matrix

Antenna or array data, specified as one of the following:

- A real length-*M* vector, where *M* contains the magnitude values with angles assumed to be $\frac{(0:M-1)}{M}\times 360^{\circ}$ degrees.
- A real *M*-by-*N* matrix, where *M* contains the magnitude values and *N* contains the independent data sets. Each column in the matrix has angles taken from the vector $\frac{(0:M-1)}{M}\times 360^{\circ}$ degrees.
- A real *N*-*D* array, where *N* is the number of dimensions. Arrays with dimensions 2 and greater are independent data sets.
- A complex vector or matrix, where `data` contains Cartesian coordinates (*x*, *y*) of each point. *x* contains the real (`data`) and *y* contains the imaginary (`data`).

When data is in a logarithmic form, such as dB, magnitude values can be negative. In this case,`polarpattern` plots the smallest magnitude values at the origin of the polar plot and largest magnitude values at the maximum radius.

`angle` — Set of angles
vector in degrees

Set of angles, specified as a vector in degrees.

`magnitude` — Set of magnitude values
vector | matrix

Set of magnitude values, specified as a vector or a matrix. For a matrix of magnitude values, each column is an independent set of magnitude values and corresponds to the same set of angles.

Methods

`add`	Add data to existing polar plot
`addCursor`	Add cursor to polar plot angle
`animate`	Replace existing data with new data for animation

`createLabels`	Create legend labels
`findLobes`	Main, back and side lobe data
`replace`	Replace existing data with new data in polar plot
`showPeaksTable`	Show or hide peak marker table
`showSpan`	Show or hide angle span between two markers

Examples

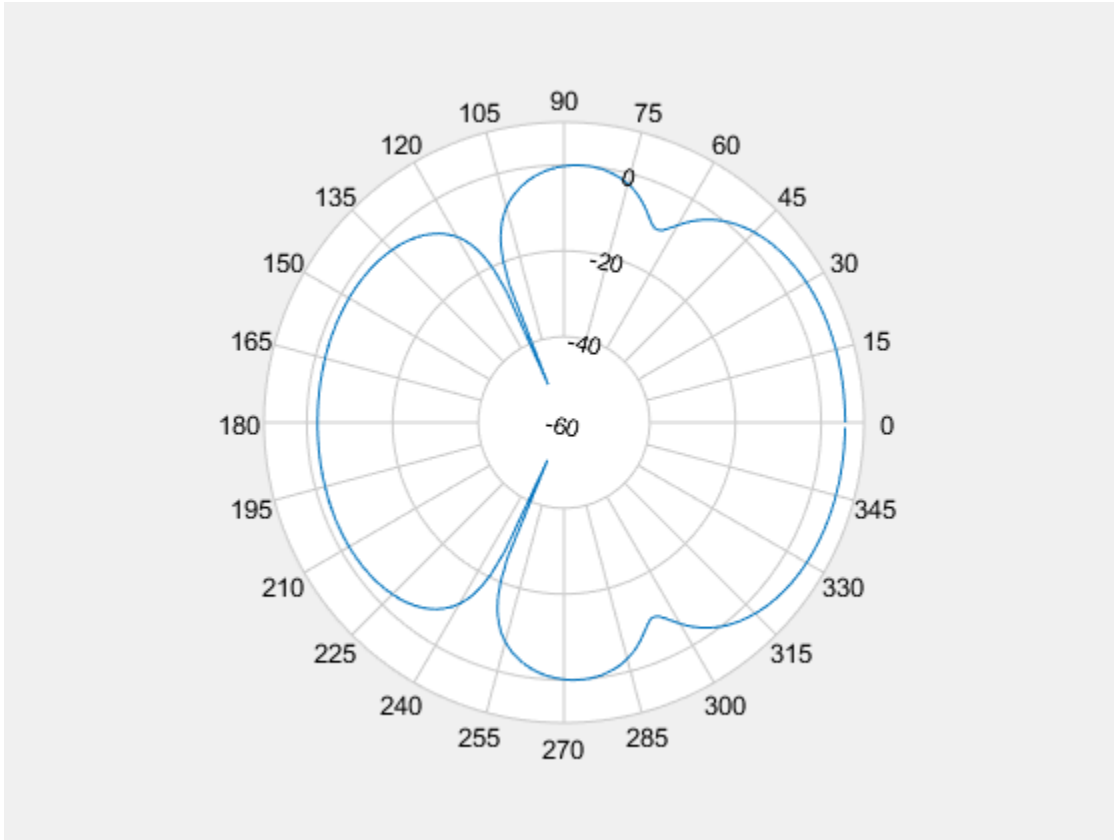
Polar Pattern for Vivaldi Antenna

Create a default Vivaldi antenna and calculate the directivity at 1.5 GHz.

```
v = vivaldi;
V = pattern(v,1.5e9,0,0:1:360);
```

Plot the polar pattern of the calculated directivity.

```
P = polarpattern(V);
```



Polar Pattern of Cavity Antenna

Create a default cavity antenna. Calculate the directivity of the antenna and write the data to `cavity.pln` using the `msiwrite` function.

```
c = cavity;
msiwrite(c,2.8e9,'cavity','Name','Cavity Antenna Specifications');
```

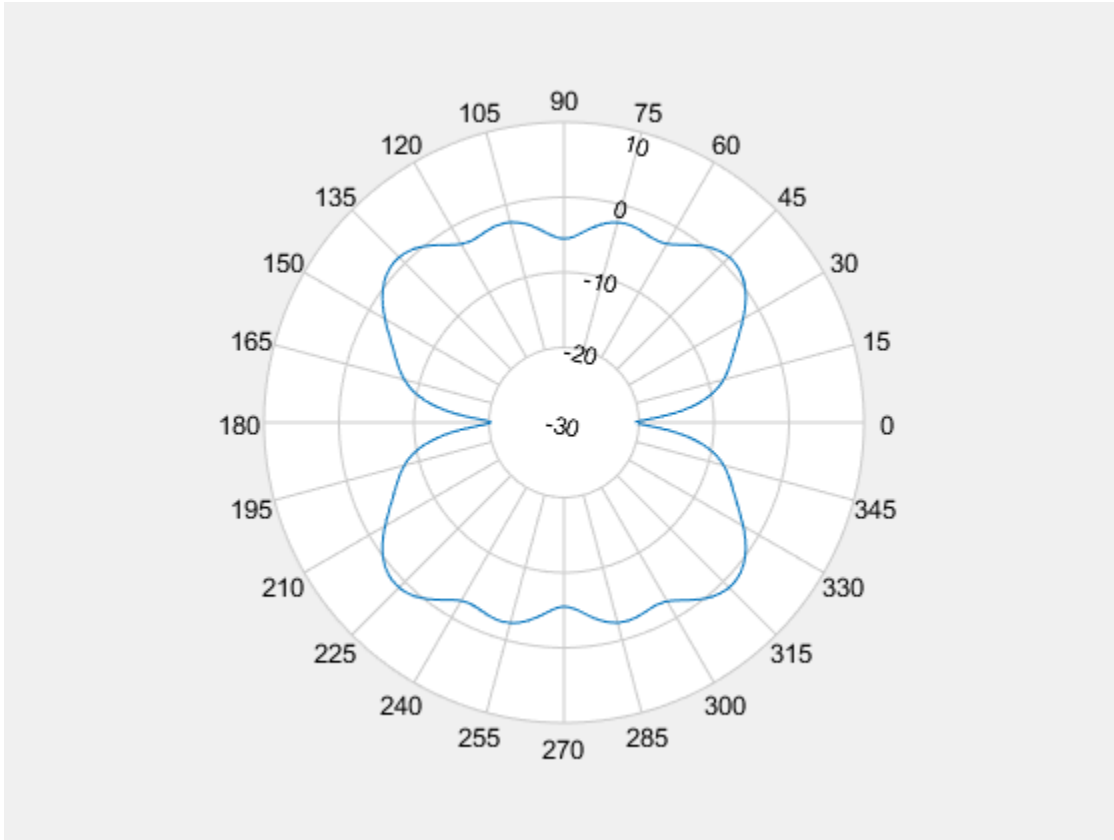
Read the cavity specification file into `Horizontal`, `Vertical`, and `Optional` structures using the `msiread` function.

```
[Horizontal,Vertical,Optional] = msiread('cavity.pln')
```

```
Horizontal = struct with fields:
    PhysicalQuantity: 'Gain'
           Magnitude: [360x1 double]
               Units: 'dBi'
             Azimuth: [360x1 double]
           Elevation: 0
           Frequency: 2.8000e+09
               Slice: 'Elevation'


Vertical = struct with fields:
    PhysicalQuantity: 'Gain'
           Magnitude: [360x1 double]
               Units: 'dBi'
             Azimuth: 0
           Elevation: [360x1 double]
           Frequency: 2.8000e+09
               Slice: 'Azimuth'


Optional = struct with fields:
         name: 'Cavity Antenna Specifications'
    frequency: 2.8000e+09
         gain: [1x1 struct]
```

Plot the polar pattern of the cavity at azimuth angles.

```
P = polarpattern(Horizontal.Azimuth,Horizontal.Magnitude);
```



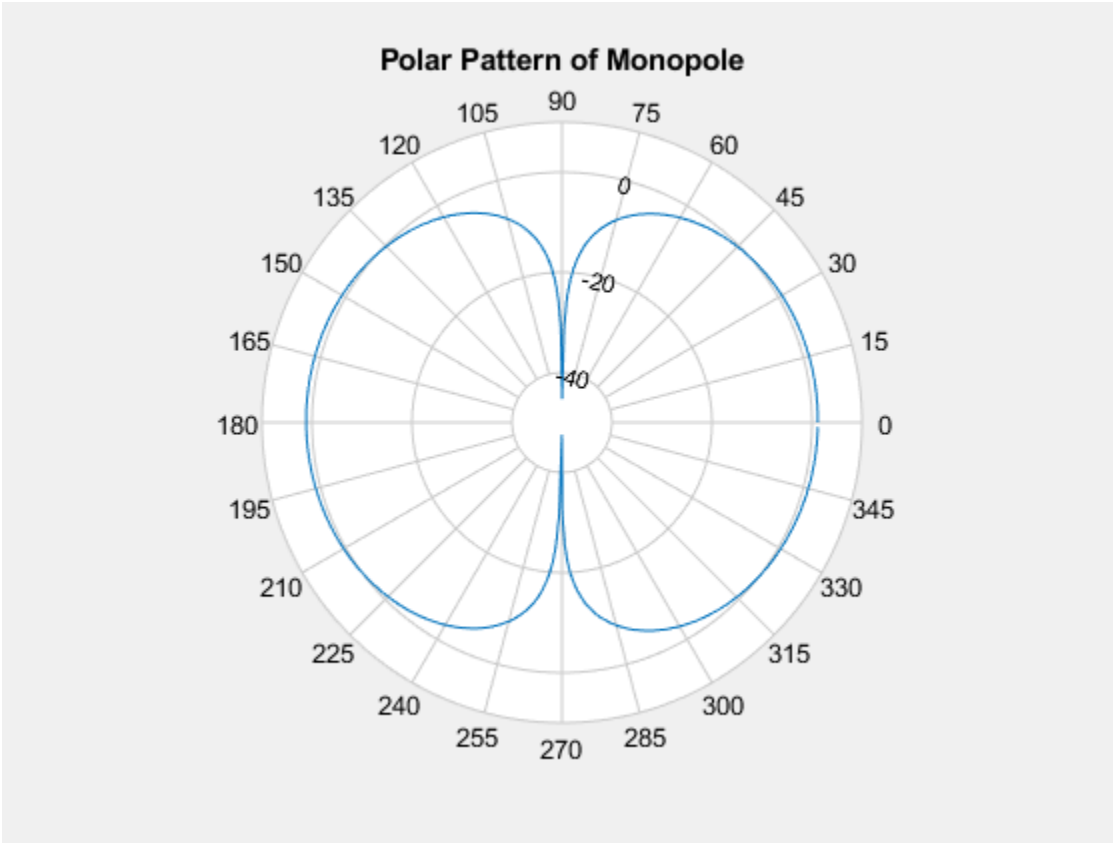
Add Title to Polar Plot

Create a default monopole antenna and calculate the directivity at 75 MHz.

```
m = monopole;
M = pattern(m,75e6,0,0:1:360);
```

Plot the polar pattern of the antenna.

```
P = polarpattern(M,'TitleTop','Polar Pattern of Monopole');
```



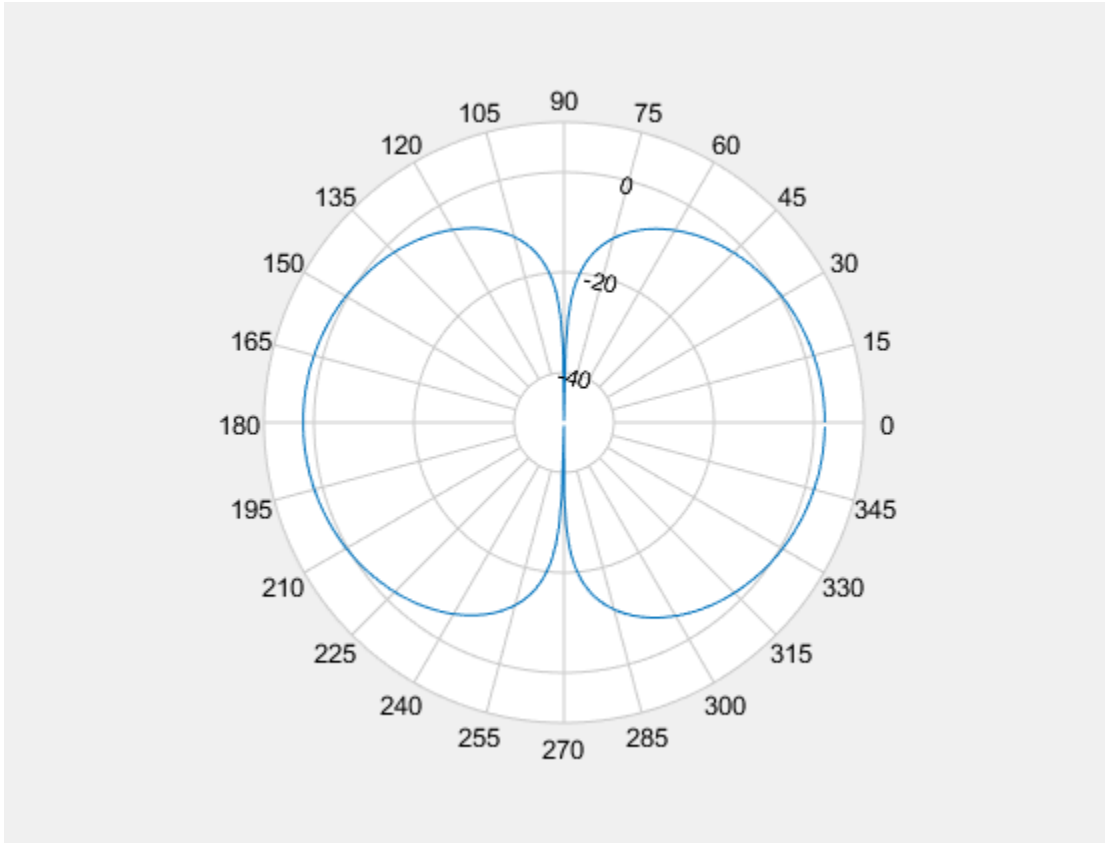
Polar Pattern Properties

Create a default dipole antenna and calculate the directivity at 75 MHz.

```
d = dipole;
D = pattern(d,75e6,0,0:1:360);
```

Plot the polar pattern of the antenna and display the properties of the plot.

```
P = polarpattern(D);
```




```
details(P)
```

```
  internal.polari handle with properties:

                    Interactive: 1
                   LegendLabels: ''
                AntennaMetrics: 0
                      CleanData: 1
                      AngleData: [361x1 double]
                  MagnitudeData: [361x1 double]
                  IntensityData: []
                   AngleMarkers: [0x1 struct]
                  CursorMarkers: [0x1 struct]
                    PeakMarkers: [0x1 struct]
```

```
                ActiveDataset: 1
              AngleLimVisible: 0
                LegendVisible: 0
                         Span: 0
                     TitleTop: ''
                  TitleBottom: ''
                        Peaks: []
                     FontSize: 10
                MagnitudeLim: [-50 10]
           MagnitudeAxisAngle: 75
                MagnitudeTick: [-40 -20 0]
     MagnitudeTickLabelColor: 'k'
                    AngleLim: [0 360]
                AngleTickLabel: {1x24 cell}
          AngleTickLabelColor: 'k'
   TitleTopFontSizeMultiplier: 1.1000
TitleBottomFontSizeMultiplier: 0.9000
           TitleTopFontWeight: 'bold'
        TitleBottomFontWeight: 'normal'
      TitleTopTextInterpreter: 'none'
   TitleBottomTextInterpreter: 'none'
               TitleTopOffset: 0.1500
            TitleBottomOffset: 0.1500
                     ToolTips: 1
          MagnitudeLimBounds: [-Inf Inf]
  MagnitudeFontSizeMultiplier: 0.9000
      AngleFontSizeMultiplier: 1
                  AngleAtTop: 90
              AngleDirection: 'ccw'
             AngleResolution: 15
       AngleTickLabelRotation: 0
          AngleTickLabelFormat: '360'
      AngleTickLabelColorMode: 'contrast'
                PeaksOptions: {}
        AngleTickLabelVisible: 1
                        Style: 'line'
                    DataUnits: 'dB'
                 DisplayUnits: 'dB'
               NormalizeData: 0
            ConnectEndpoints: 0
          DisconnectAngleGaps: 0
                    EdgeColor: 'k'
                    LineStyle: '-'
                    LineWidth: 1
```

```
                    FontName: 'Helvetica'
                FontSizeMode: 'auto'
          GridForegroundColor: [0.8000 0.8000 0.8000]
          GridBackgroundColor: 'w'
             DrawGridToOrigin: 0
                 GridOverData: 0
           GridAutoRefinement: 0
                    GridWidth: 0.5000
                GridVisible: 1
                    ClipData: 1
              TemporaryCursor: 1
            MagnitudeLimMode: 'auto'
       MagnitudeAxisAngleMode: 'auto'
            MagnitudeTickMode: 'auto'
  MagnitudeTickLabelColorMode: 'contrast'
    MagnitudeTickLabelVisible: 1
               MagnitudeUnits: ''
               IntensityUnits: ''
                       Marker: 'none'
                   MarkerSize: 6
                       Parent: [1x1 Figure]
                     NextPlot: 'replace'
                   ColorOrder: [7x3 double]
              ColorOrderIndex: 1
                 SectorsColor: [16x3 double]
                 SectorsAlpha: 0.5000
                         View: 'full'
                ZeroAngleLine: 0
```

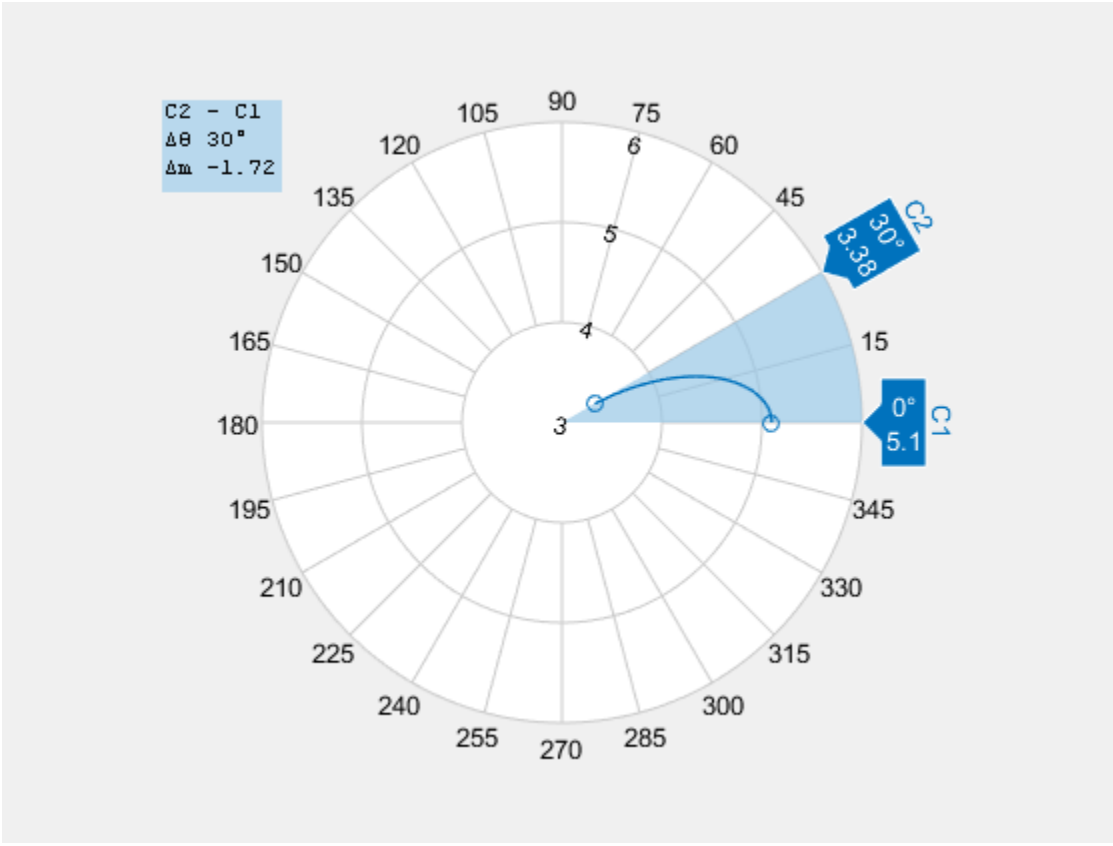
Remove -Inf and NaN Values in Antenna PolarPattern

Use `Clean Data` in `Antenna Metrics` to remove -inf and NaN values in a monopole antenna polar pattern. It is recommended to use `Clean Data` for partial data with -inf and NaN values.

```
m = monopole;
m.GroundPlaneLength = inf;
```

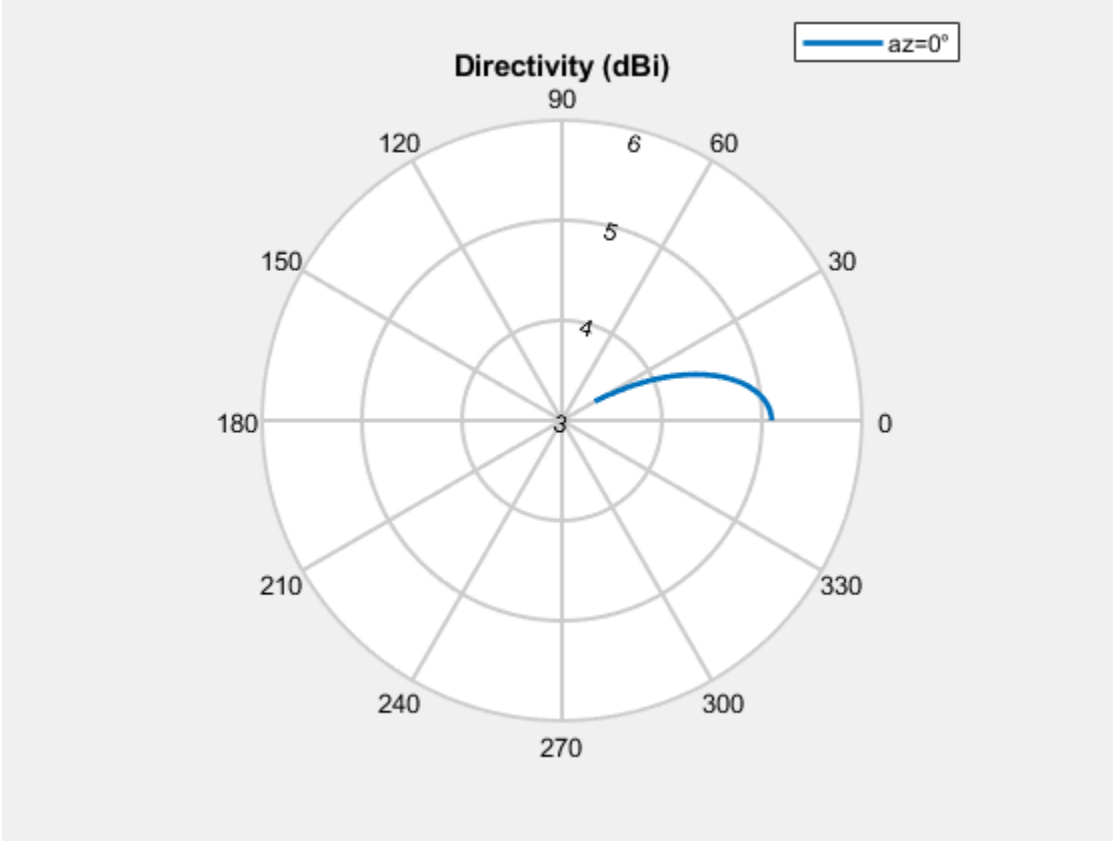
Plot the beamwidth of the antenna at 70 MHz.

```
figure;
beamwidth(m,70e6,0,-50:30)
```



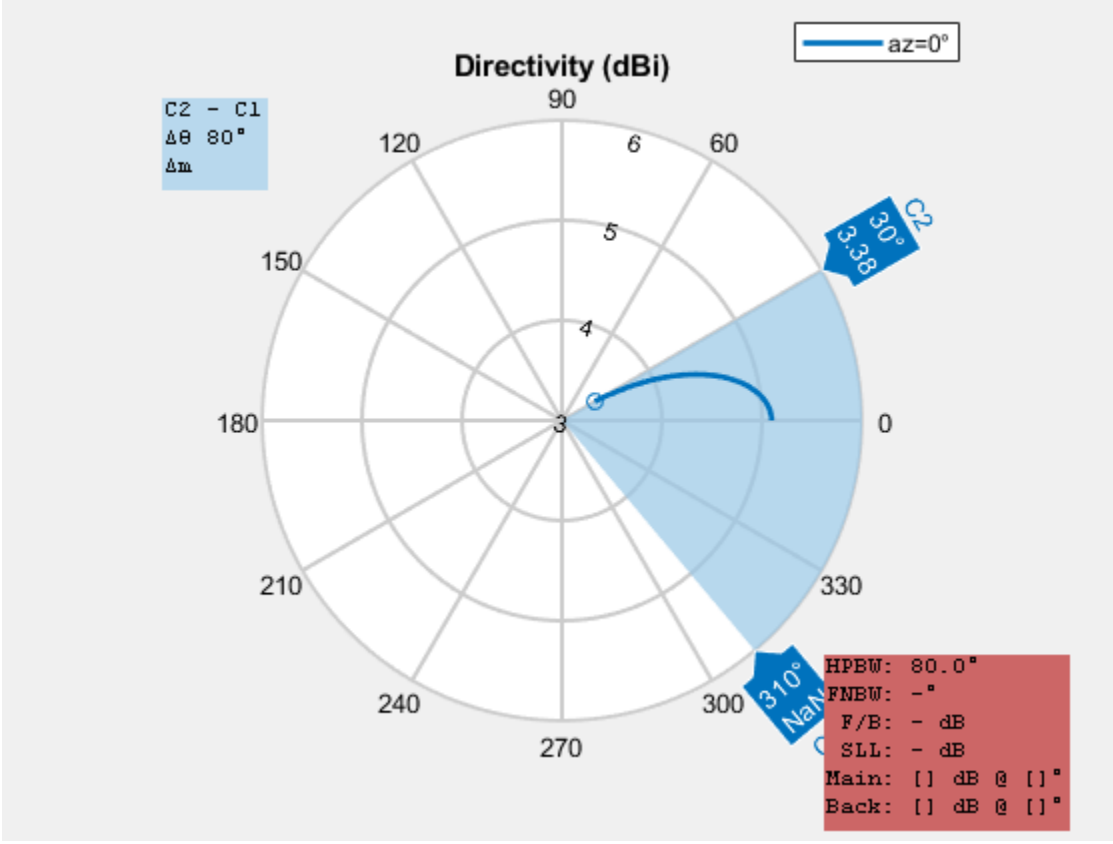
Plot the radiation pattern of the antenna at 70 MHz.

```
figure;
pattern(m,70e6,0,-50:30);
```



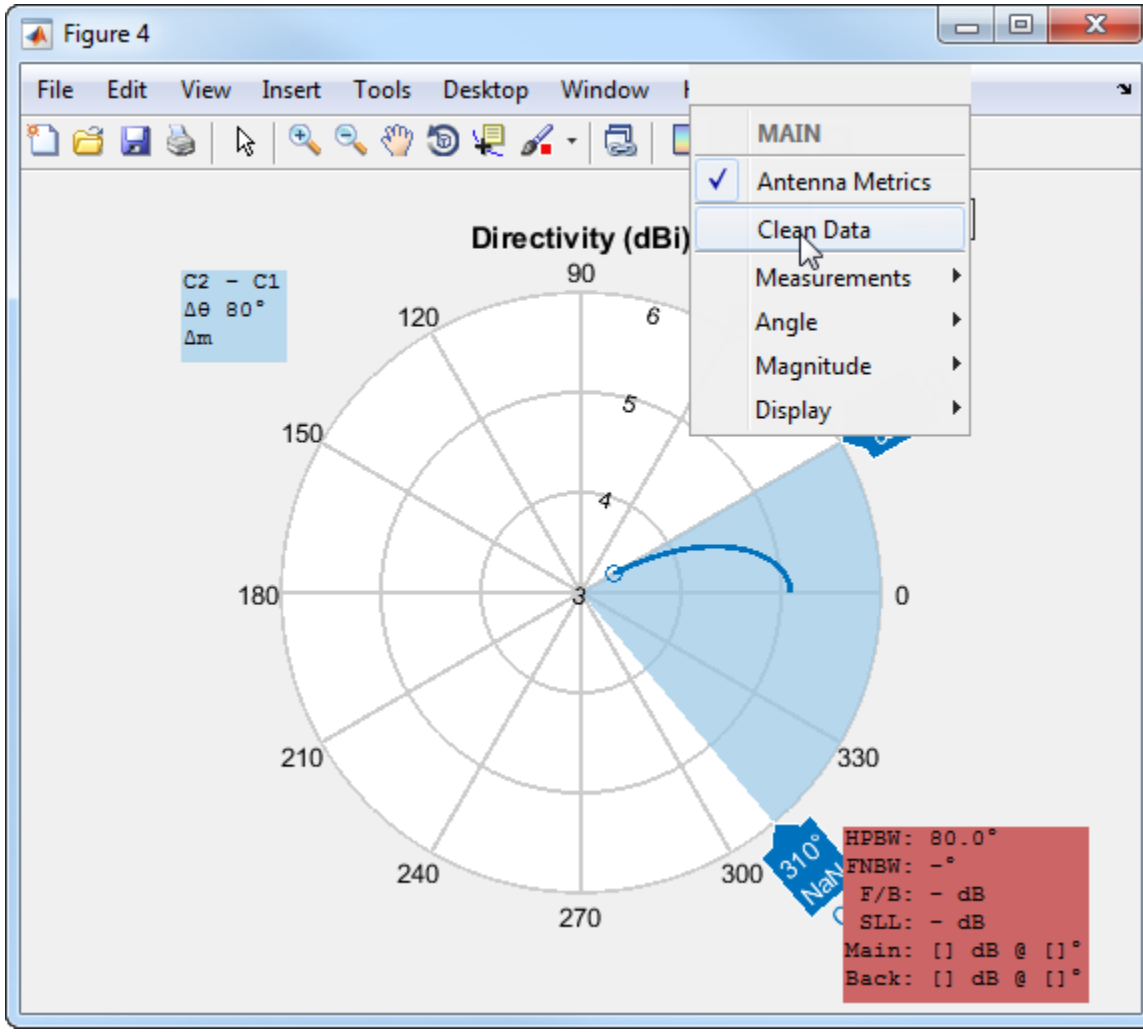
Use `polarpattern` to view antenna metrics of the radiation pattern.

```
P = polarpattern('gco');
P.AntennaMetrics = 1;
```

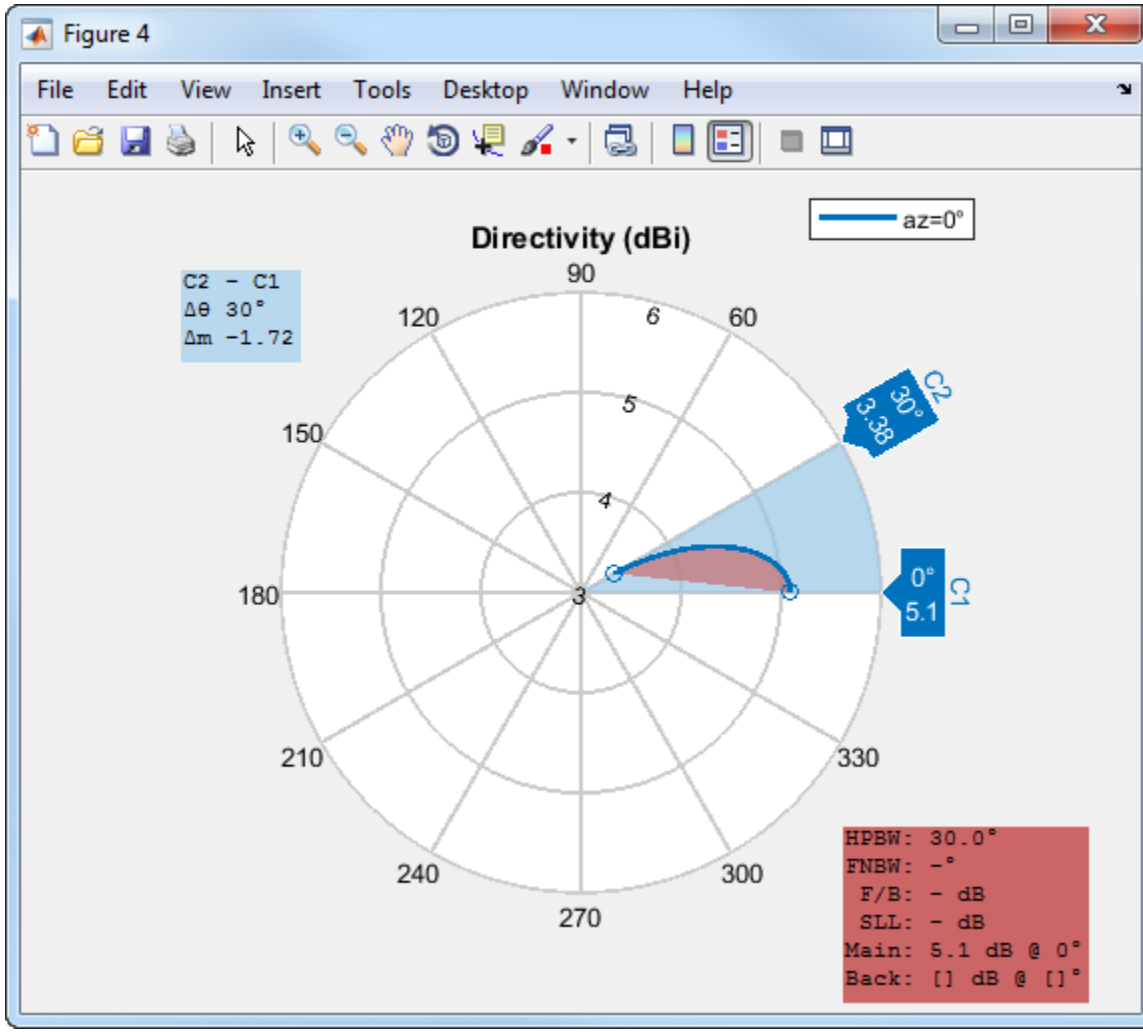


Compare the beamwidth plot and the polarpattern plot. You will see that Antenna Metrics does not represent the beamwidth correctly.

Use `Clean Data` to clean the -inf and NaN values.



After using `Clean Data`, you see that the polarpattern beamwidth calculation matches the beamwidth plot calculation.



See Also

Topics

“Interact with Polar Plot”

Introduced in R2016a

Antenna Objects — Alphabetical List

biquad

Create biquad antenna

Description

The `biquad` antenna is center fed and symmetric about its origin. The default length is chosen for an operating frequency of 2.8 GHz.

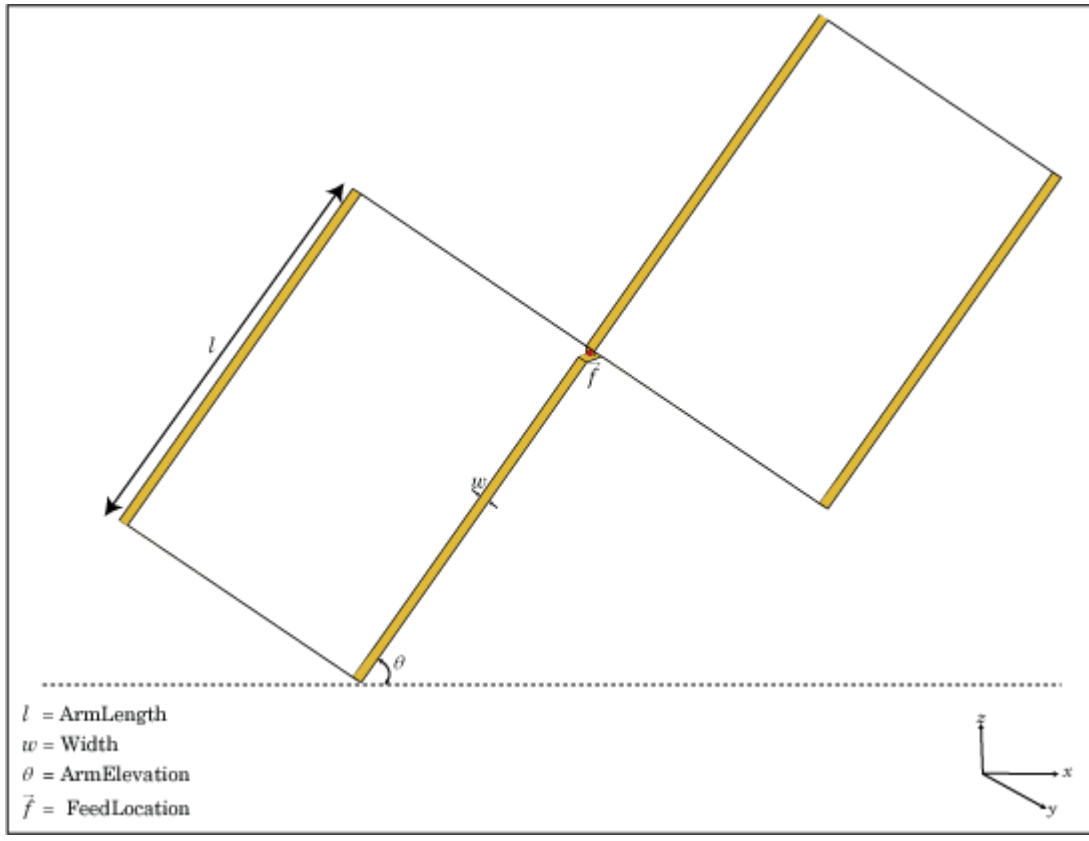
The width of the strip is related to the diameter an equivalent cylinder:

$$w = 2d = 4r$$

, where:

- d is the diameter of equivalent cylindrical dipole.
- r is the radius of equivalent cylindrical dipole.

For a given cylinder radius, use the `cylinder2strip` utility function to calculate the equivalent width. The default strip dipole is center-fed. The feed point coincides with the origin. The origin is located on the Y-Z plane.



Creation

Syntax

```
bq = biquad
bq = biquad(Name,Value)
```

Description

`bq = biquad` creates a biquad antenna.

`bq = biquad(Name,Value)` creates a biquad antenna with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`ArmLength` — Length of two arms

0.0305 (default) | scalar

Length of two arms, specified as a scalar in meters. The default length is chosen for an operating frequency of 2.8 GHz.

Example: `'ArmLength',0.0206`

Data Types: `double`

`Width` — Biquad arm width

1.0000e-03 (default) | scalar

Biquad arm width, specified as a scalar in meters.

Example: `'Width',0.006`

Data Types: `double`

`ArmElevation` — Angle formed by biquad arms to X-Y plane

45 (default) | scalar

Angle formed by biquad arms to the X-Y plane, specified a scalar in meters.

Example: `'ArmElevation', 50`

Data Types: `double`

`Load` — Lumped elements

[1x1 LumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load'`, lumpedelement. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `bq.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface

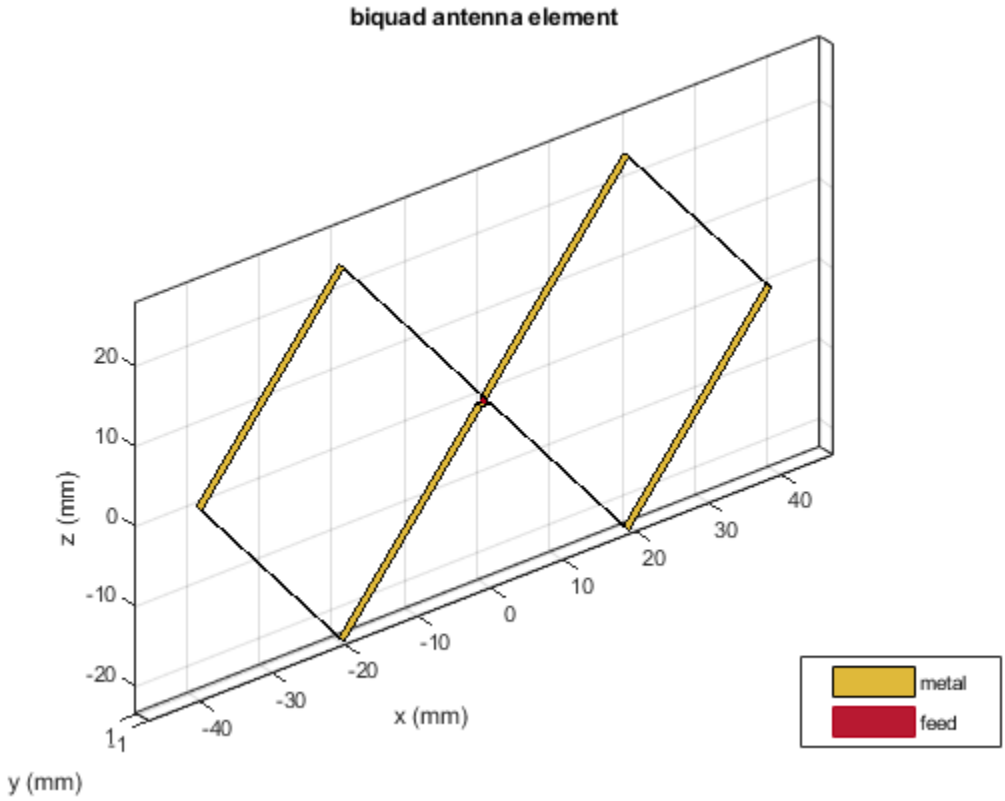
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

Create and View Biquad Antenna

Create a biquad antenna with arm angles at 50 degrees and view it.

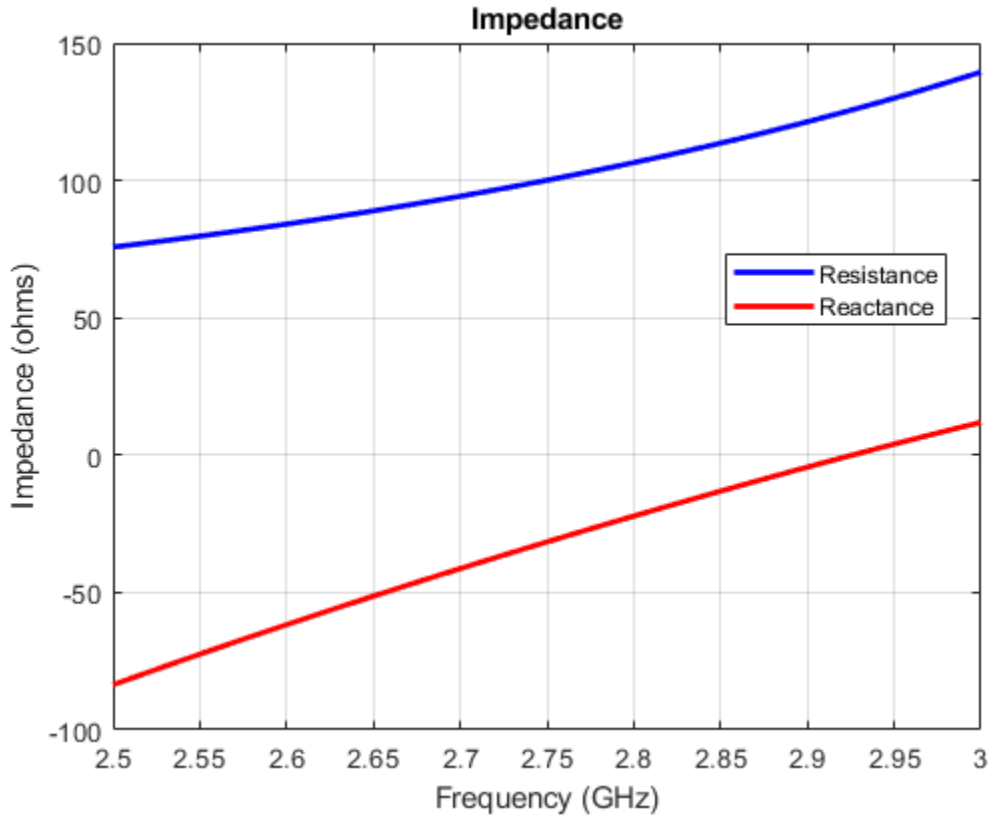
```
bq = biquad('ArmElevation',50);
show(bq)
```



Impedance of Biquad Antenna

Calculate the impedance of a biquad antenna over a frequency span 2.5GHz-3GHz.

```
bq = biquad('ArmElevation',50);
impedance(bq,linspace(2.5e9,3e9,51));
```



See Also

`dipole` | `dipoleFolded` | `loopCircular`

Topics

"Rotate Antenna and Arrays"

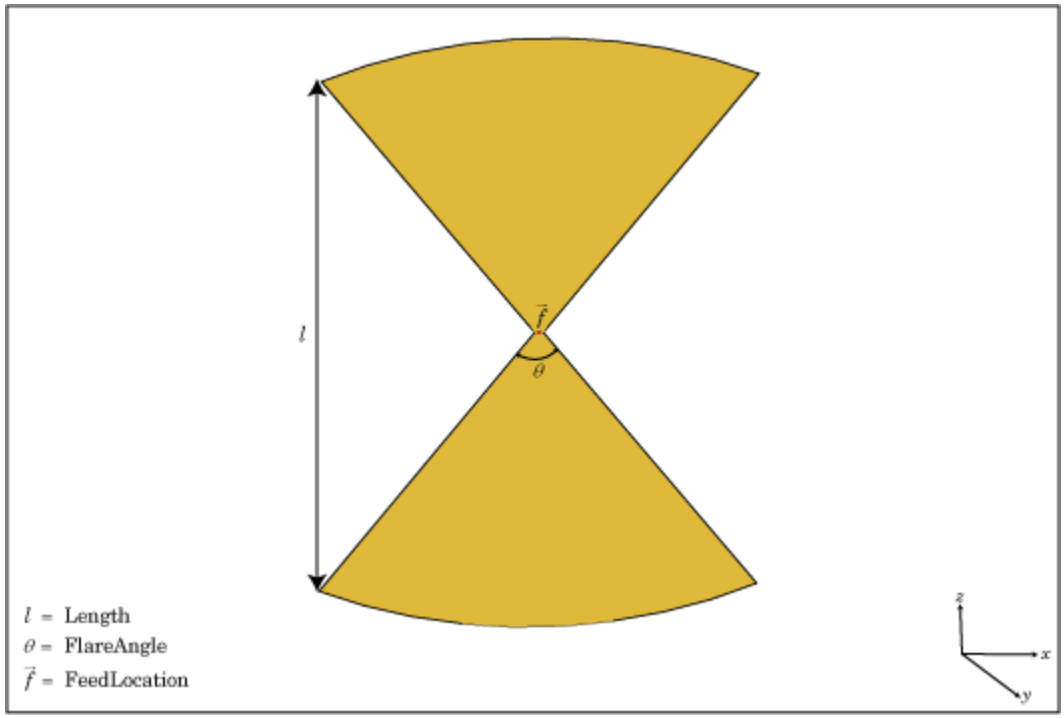
Introduced in R2015b

bowtieRounded

Create rounded bowtie dipole antenna

Description

The `bowtieRounded` object is a planar bowtie antenna, with rounded edges, on the Y–Z plane. The default rounded bowtie is center fed. The feed point coincides with the origin. The origin is located on the Y-Z plane.



Creation

Syntax

```
br = bowtieRounded
br = bowtieRounded(Name,Value)
```

Description

`br = bowtieRounded` creates a half-wavelength planar bowtie antenna with rounded edges.

`br = bowtieRounded(Name,Value)` creates a planar bowtie antenna with rounded edges, with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Length` — Rounded bowtie length

0.2000 (default) | scalar

Rounded bowtie length, specified a scalar in meters. By default, the length is chosen for the operating frequency of 490 MHz.

Example: `'Length',3`

Data Types: `double`

`FlareAngle` — Rounded bowtie flare angle

90 (default) | scalar

Rounded bowtie flare angle, specified a scalar in degrees.

Note Flare angle should be less than 175 degrees and greater than 5 degrees.

Example: `'FlareAngle',80`

Data Types: `double`

Load — Lumped elements

[1x1 LumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load'`, lumpedelement. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `br.Load = lumpedElement('Impedance',75)`

Tilt — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

TiltAxis — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

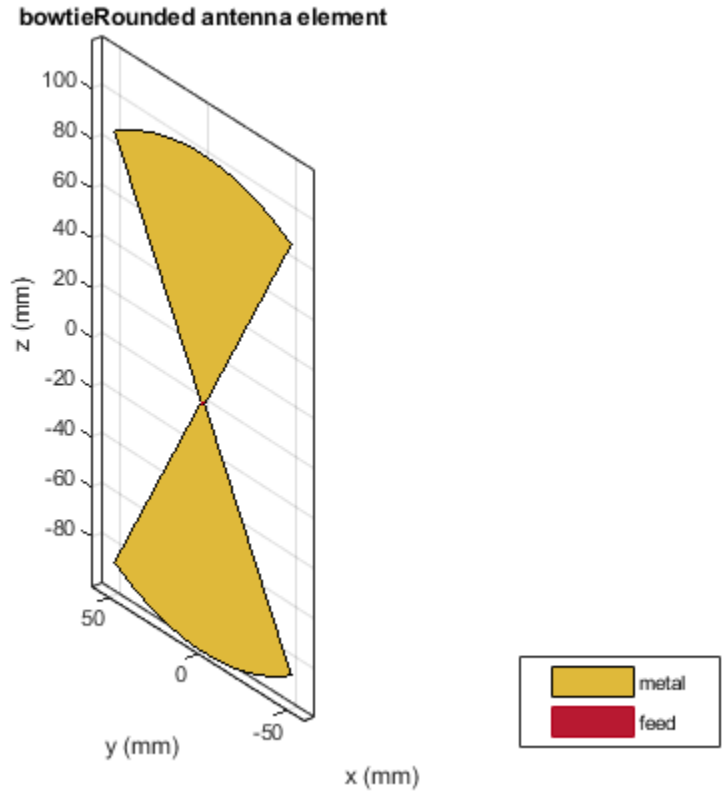
show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

Create and View Center-Fed Rounded Bowtie Antenna

Create and view a center-fed rounded bowtie that has a flare angle of 60 degrees.

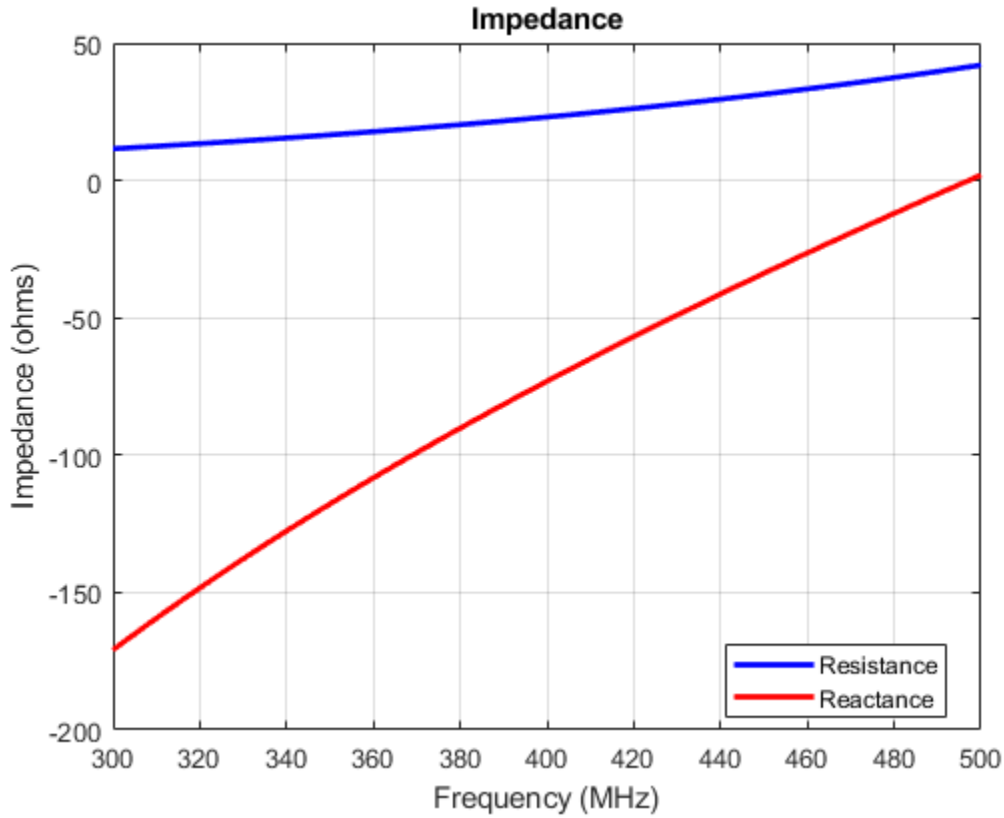
```
b = bowtieRounded('FlareAngle',60);
show(b);
```



Impedance of Rounded Bowtie Antenna

Calculate and plot the impedance of a rounded bowtie over a frequency range of 300MHz-500MHz.

```
b = bowtieRounded('FlareAngle',60);
impedance(b,linspace(300e6,500e6,51))
```



References

[1] Balanis, C.A.*Antenna Theory: Analysis and Design*.3rd Ed. New York: Wiley, 2005.

[2] Brown, G.H., and O.M. Woodward Jr. “Experimentally Determined Radiation Characteristics of Conical and Triangular Antennas”. *RCA Review*. Vol.13, No.4, Dec.1952, pp. 425–452

See Also

`bowtieTriangular` | `dipole` | `dipoleFolded`

Topics

“Rotate Antenna and Arrays”

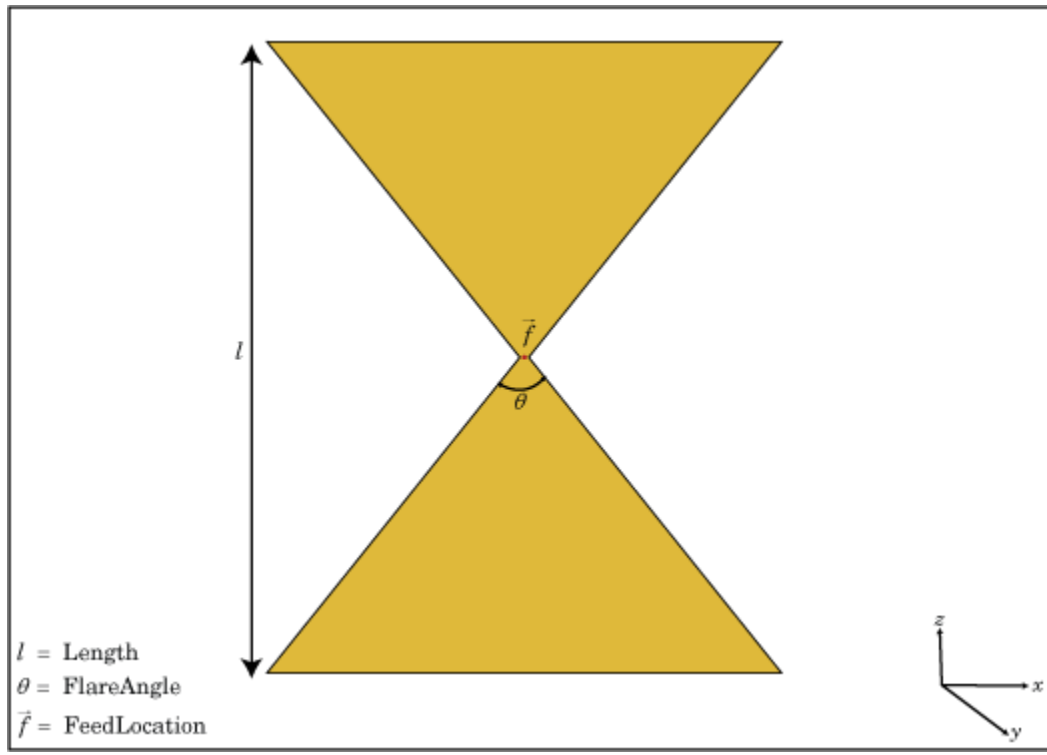
Introduced in R2015a

bowtieTriangular

Create planar bowtie dipole antenna

Description

The `bowtieTriangular` object is a planar bowtie antenna on the Y-Z plane. The default planar bowtie dipole is center-fed. The feed point coincides with the origin. The origin is located on the Y-Z plane.



Creation

Syntax

```
bt = bowtieTriangular
bt = bowtieTriangular(Name,Value)
```

Description

`bt = bowtieTriangular` creates a half-wavelength planar bowtie antenna.

`bt = bowtieTriangular(Name,Value)` creates a planar bowtie antenna with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Length` — Planar bowtie length

0.2000 (default) | scalar

Planar bowtie length, specified as a scalar in meters. By default, the length is chosen for the operating frequency of 410 MHz.

Example: `'Length',3`

Data Types: `double`

`FlareAngle` — Planar bowtie flare angle

90 (default) | scalar

Planar bowtie flare angle near the feed, specified as a scalar in meters.

Note Flare angle should be less than 175 degrees and greater than 5 degrees.

Example: `'FlareAngle',80`

Data Types: `double`

Load — Lumped elements
[1x1 LumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load'`, lumpedelement. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `bt.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

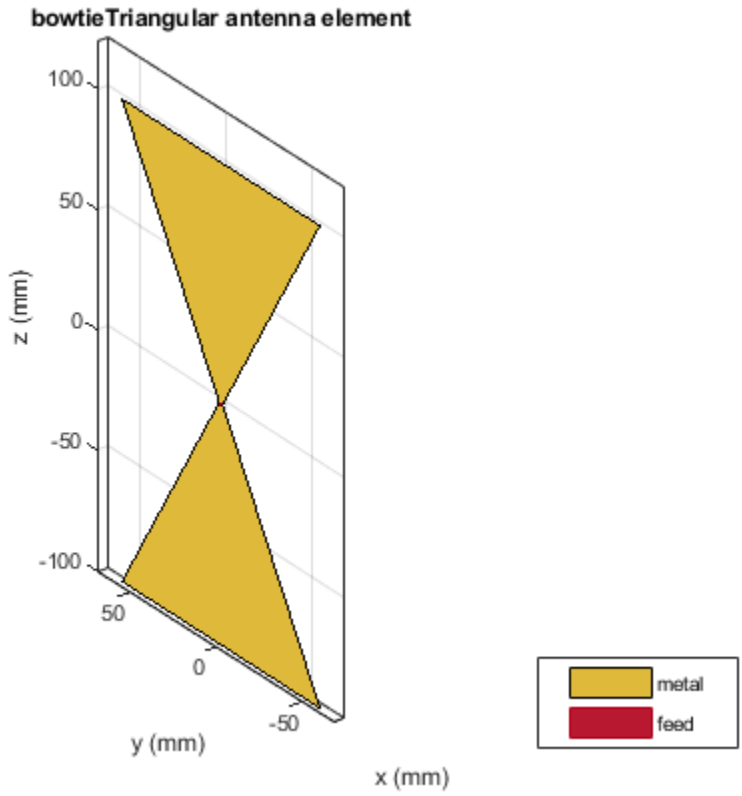
Create and View Center-Fed Planar Bowtie Antenna

Create and view a center-fed planar bowtie antenna that has a 60 degrees flare angle.

```
b = bowtieTriangular('FlareAngle',60)
```

```
b =
  bowtieTriangular with properties:

        Length: 0.2000
    FlareAngle: 60
          Tilt: 0
      TiltAxis: [1 0 0]
          Load: [1x1 lumpedElement]
```

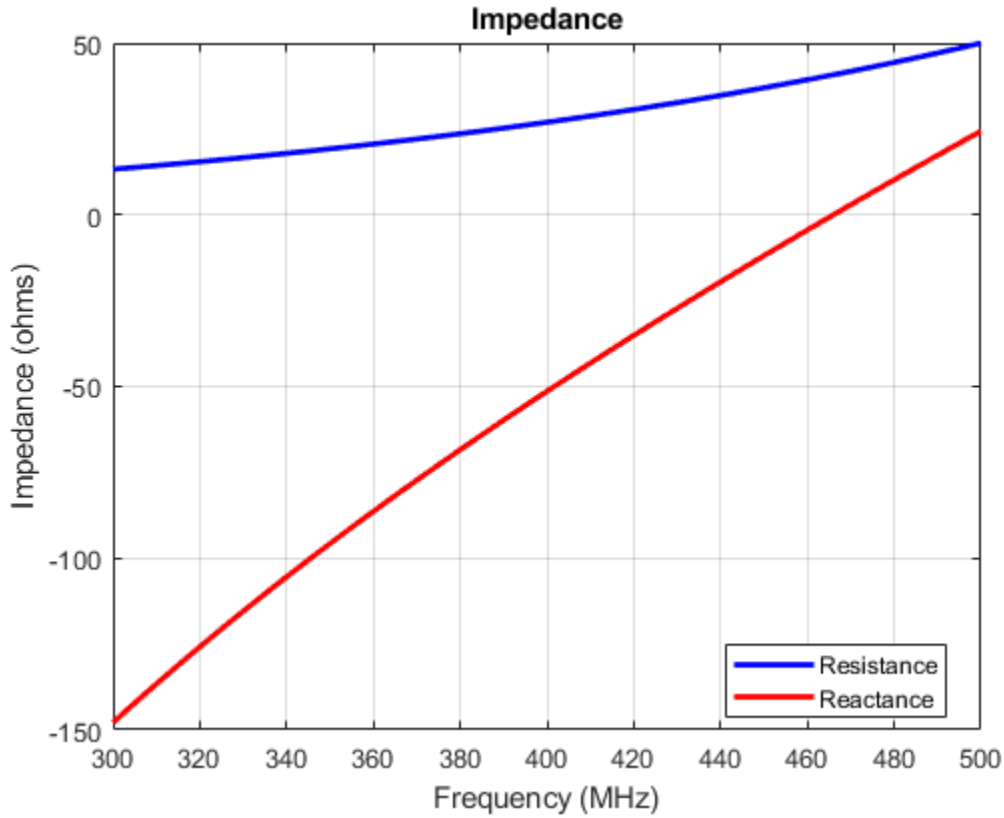
```
show(b)
```



Impedance of Planar Bowtie Antenna

Calculate and plot the impedance of a planar bowtie antenna over a frequency range of 300MHz-500MHz.

```
b = bowtieTriangular('FlareAngle',60);
impedance(b,linspace(300e6,500e6,51))
```



References

[1] Balanis, C.A.*Antenna Theory: Analysis and Design*.3rd Ed. New York: Wiley, 2005.

[2] Brown, G.H., and O.M. Woodward Jr. “Experimentally Determined Radiation Characteristics of Conical and Triangular Antennas”. *RCA Review*. Vol.13, No.4, Dec.1952, pp. 425–452

See Also

`bowtieRounded` | `dipole` | `dipoleVee`

Topics

“Rotate Antenna and Arrays”

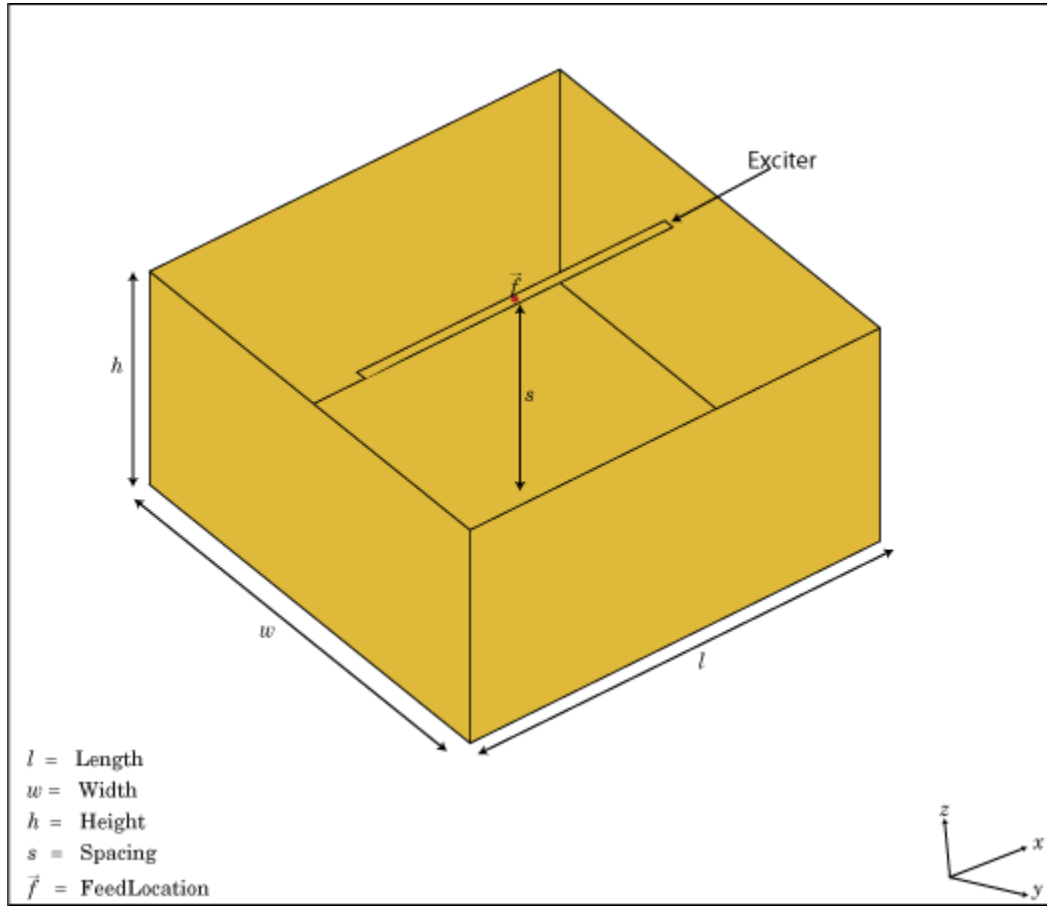
Introduced in R2015a

cavity

Create cavity-backed antenna

Description

The `cavity` object is a cavity-backed antenna located on the X-Y-Z plane. The default cavity antenna has a dipole as an exciter. The feed point is on the exciter.



Creation

Syntax

```
c = cavity
c = cavity(Name,Value)
```

Description

`c = cavity` creates a cavity backed antenna located on the X-Y-Z plane. By default, the dimensions are chosen for an operating frequency of 1 GHz.

`c = cavity(Name,Value)` creates a cavity-backed antenna, with additional properties specified by one or more name–value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Exciter` — Antenna type used as exciter
`dipole` (default) | object

Antenna type used as an exciter, specified as an object. Except reflector and cavity antenna elements, you can use all the single elements in the Antenna Toolbox as an exciter.

Example: `'Exciter',dipole`

Data Types: `char` | `string`

`Substrate` — Type of dielectric material
'Air' (default) | object

Type of dielectric material used as a substrate, specified as an object. For more information see, `dielectric`. For more information on dielectric substrate meshing, see “Meshing”.

Note The substrate dimensions must be equal to the groundplane dimensions.

Example: `d = dielectric('FR4'); 'Substrate',d`

Example: `d = dielectric('FR4'); cavity.Substrate = d`

`Length` — Length of rectangular cavity along x-axis

0.2000 (default) | scalar

Length of the rectangular cavity along the x-axis, specified as a scalar in meters.

Example: `'Length',30e-2`

Data Types: `double`

`Width` — Width of rectangular cavity along y-axis

0.2000 (default) | scalar

Width of the rectangular cavity along the y-axis, specified as a scalar in meters.

Example: `'Width',25e-2`

Data Types: `double`

`Height` — Height of rectangular cavity along z-axis

0.0750 (default) | scalar

Height of the rectangular cavity along the z-axis, specified as a scalar in meters.

Example: `'Height',7.5e-2`

Data Types: `double`

`Spacing` — Distance between exciter and base of cavity

0.0750 (default) | scalar

Distance between the exciter and the base of the cavity, specified as a scalar in meters.

Example: `'Spacing',7.5e-2`

Data Types: `double`

`Load` — Lumped elements

[1x1 LumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load'`, lumpedelement. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `c.Load = lumpedElement('Impedance',75)`

`EnableProbeFeed` — Create probe feed from backing structure to exciter
`0` (default) | `1`

Create probe feed from backing structure to exciter, specified as a `0` or `1`. By default, probe feed is not enabled.

Example: `'EnableProbeFeed',1`

Data Types: `double`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

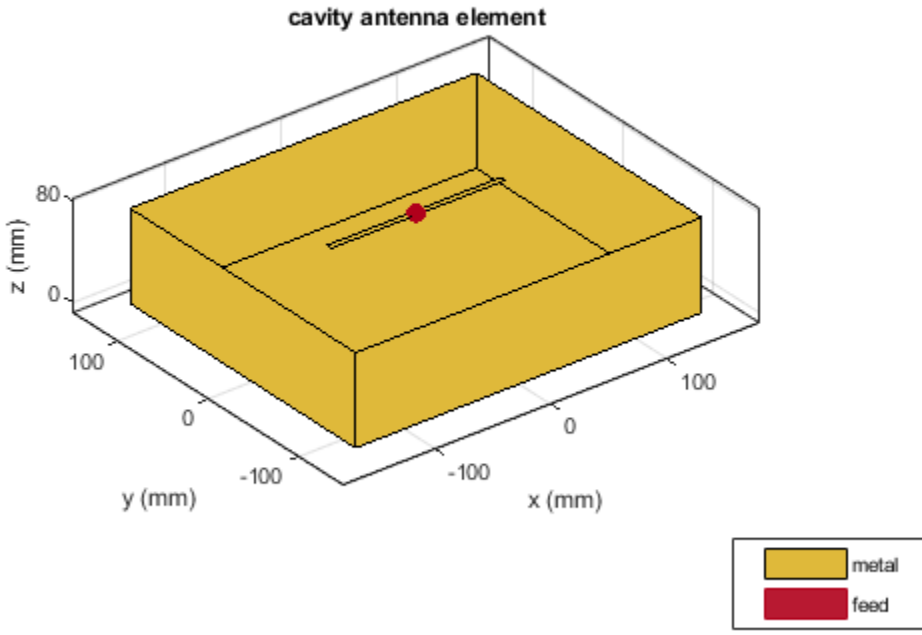
show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

Create and View Cavity-Backed Antenna.

Create and view a cavity-backed dipole antenna with 30cm length, 25cm width, 7.5cm height and spaced 7.5cm from the bowtie for operation at 1GHz.

```
c = cavity('Length',30e-2, 'Width',25e-2,'Height',7.5e-2,'Spacing',7.5e-2);
show(c)
```



Radiation Pattern of Cavity-Backed Antenna

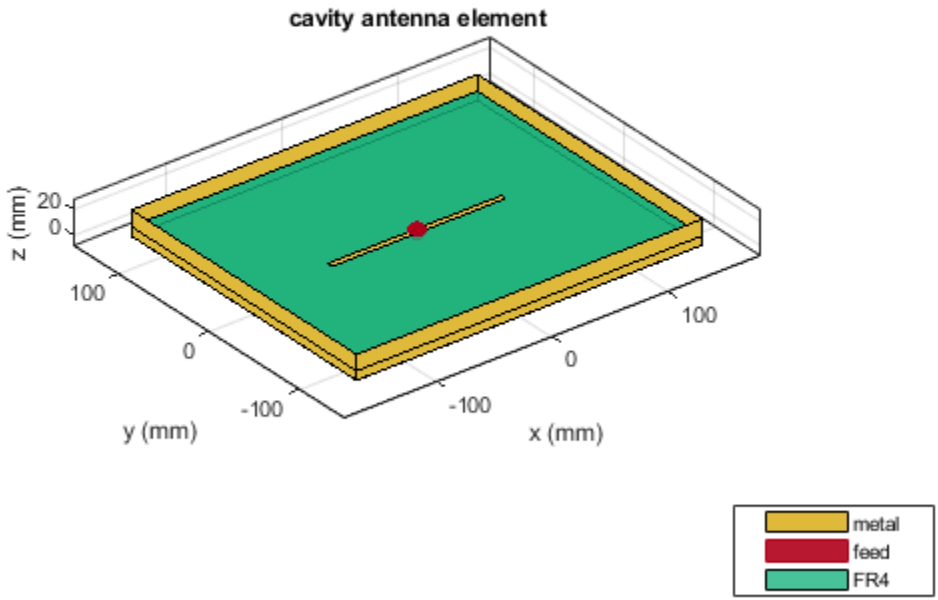
Create a cavity-backed antenna using a dielectric substrate **'FR4'**.

```
d = dielectric('FR4');
c = cavity('Length',30e-2,'Width',25e-2,'Height',20.5e-3,'Spacing',7.5e-3,...
    'Substrate',d)
```

```
c =
  cavity with properties:

            Exciter: [1x1 dipole]
```

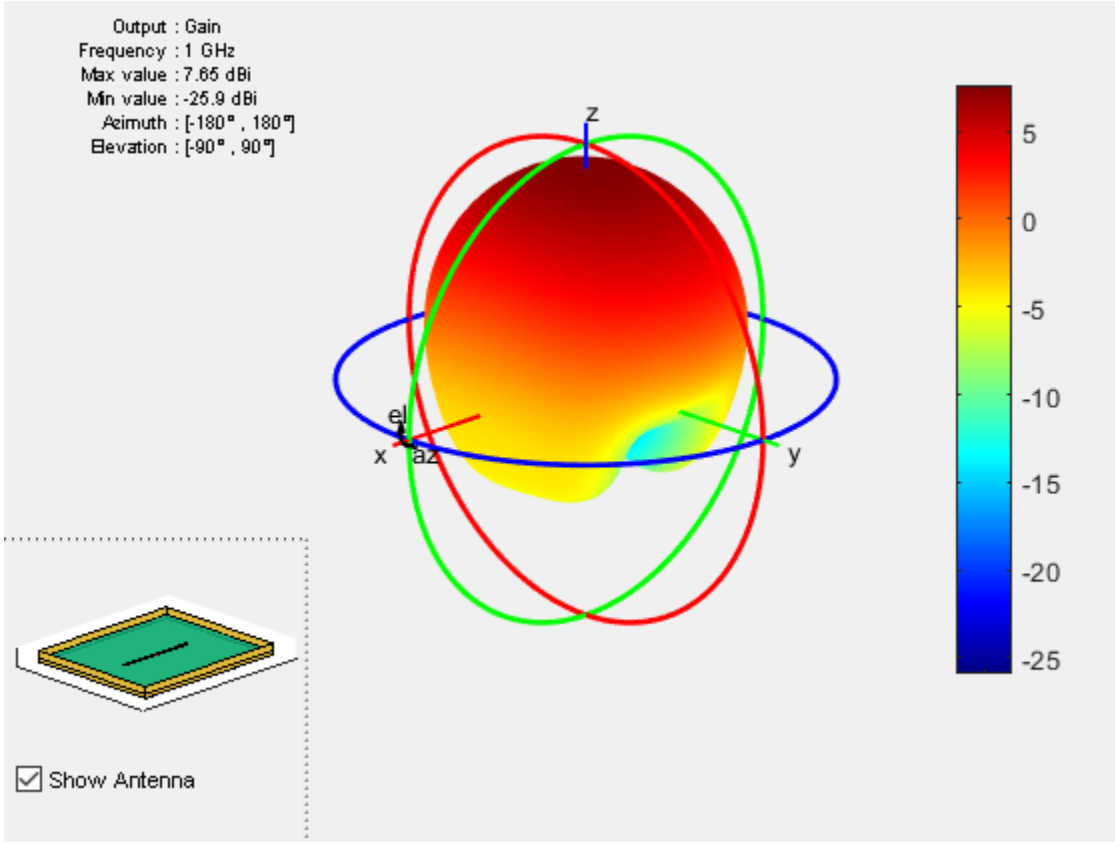
```
        Substrate: [1x1 dielectric]
           Length: 0.3000
            Width: 0.2500
           Height: 0.0205
          Spacing: 0.0075
    EnableProbeFeed: 0
             Tilt: 0
         TiltAxis: [1 0 0]
             Load: [1x1 lumpedElement]
```

```
show(c)
```



Plot the radiation pattern of the antenna at a frequency of 1 GHz.

```
figure
pattern(c,1e9)
```



References

[1] Balanis, C.A.*Antenna Theory: Analysis and Design*.3rd Ed. New York: Wiley, 2005.

See Also

`reflector` | `spiralArchimedean` | `spiralEquiangular`

Topics

“Rotate Antenna and Arrays”

Introduced in R2015a

dipole

Create strip dipole antenna

Description

The `dipole` object is a strip dipole antenna on the Y-Z plane.

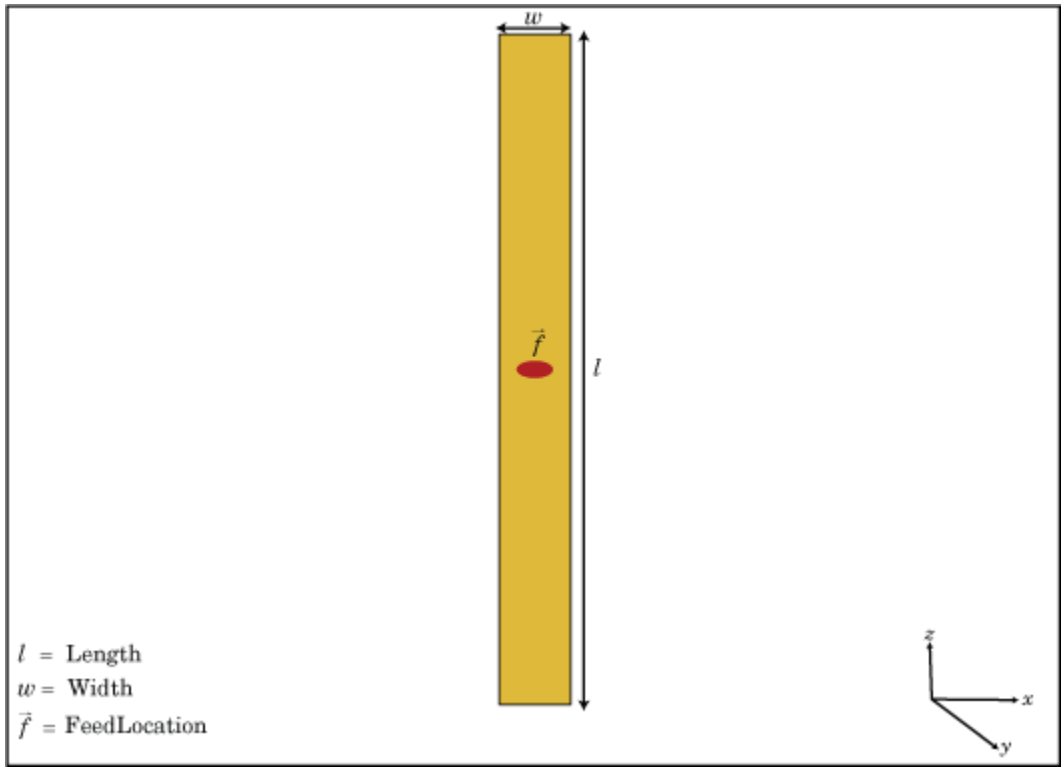
The width of the dipole is related to the diameter of an equivalent cylindrical dipole by the equation

$$w = 2d = 4r$$

where:

- d is the diameter of equivalent cylindrical dipole.
- r is the radius of equivalent cylindrical dipole.

For a given cylinder radius, use the `cylinder2strip` utility function to calculate the equivalent width. The default strip dipole is center-fed. The feed point coincides with the origin. The origin is located on the Y-Z plane.



Creation

Syntax

```
d = dipole
d = dipole(Name,Value)
```

Description

`d = dipole` creates a half-wavelength strip dipole antenna on the Y-Z plane.

`d = dipole(Name,Value)` creates a dipole antenna, with additional properties specified by one or more name-value pair arguments. `Name` is the property name and

`Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties you do not specify retains their default values.

Properties

`Length` — Dipole length
2 (default) | scalar

Dipole length, specified as a scalar in meters. By default, the length is chosen for an operating frequency of 75 MHz.

Example: `'Length',3`

Data Types: `double`

`Width` — Dipole width
0.1000 (default) | scalar

Dipole width, specified as a scalar in meters.

Note Dipole width should be less than `'Length'`/5 and greater than `'Length'`/1001. [2]

Example: `'Width',0.05`

Data Types: `double`

`FeedOffset` — Signed distance from center of dipole
0 (default) | scalar

Signed distance from center of dipole, specified as a scalar in meters. The feed location is on Y-Z plane.

Example: `'FeedOffset',3`

Data Types: `double`

Load — Lumped elements
[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `d.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna

beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

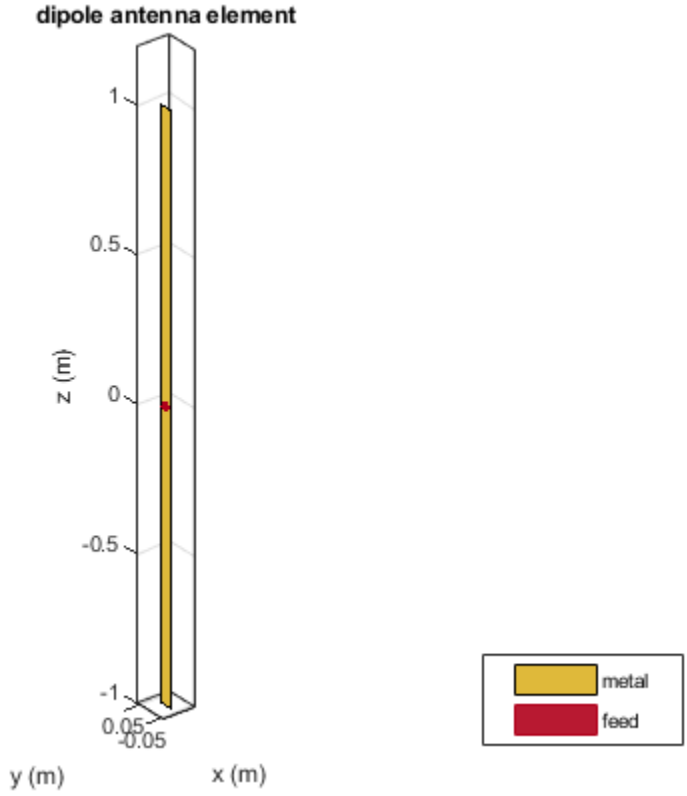
Create and View Dipole Antenna

Create and view a dipole with 2m length and 0.5m width.

```
d = dipole('Width',0.05)
```

```
d =
  dipole with properties:

        Length: 2
         Width: 0.0500
    FeedOffset: 0
          Tilt: 0
      TiltAxis: [1 0 0]
          Load: [1x1 lumpedElement]
```

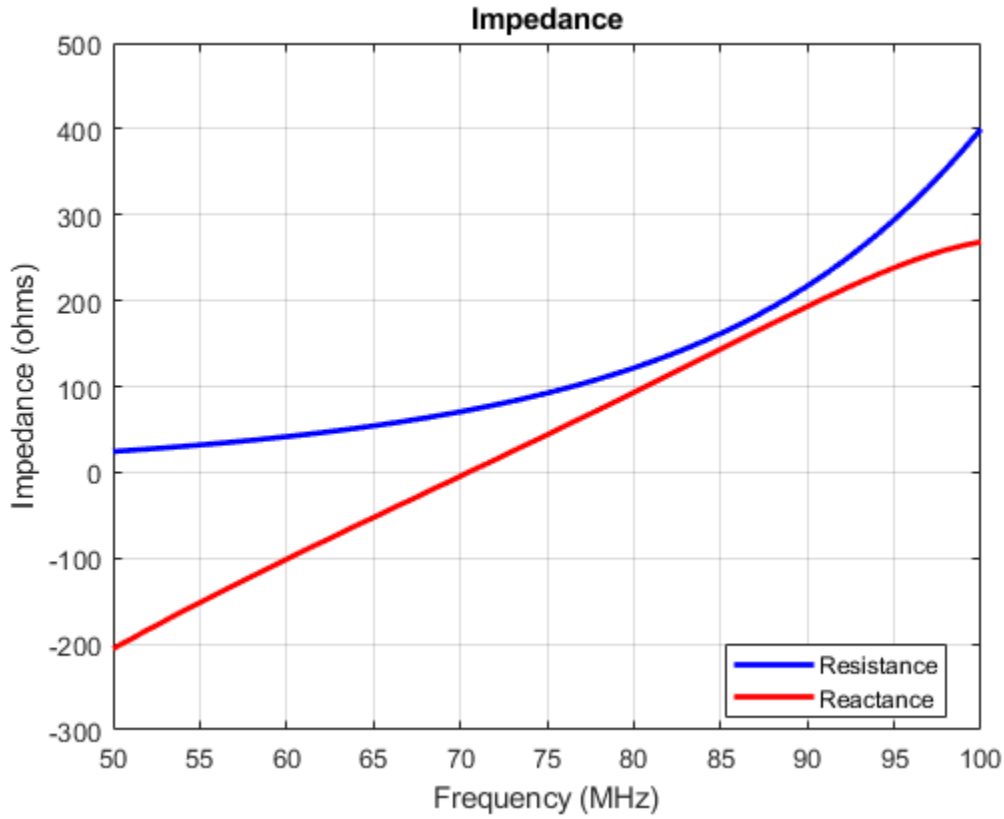
```
show(d)
```



Impedance of Dipole Antenna

Calculate the impedance of a dipole over a frequency range of 50MHz - 100MHz.

```
d = dipole('Width',0.05);
impedance(d,linspace(50e6,100e6,51))
```



Infinite Reflector Backed Dielectric Substrate Antenna

Design a dipole antenna backed by a dielectric substrate and an infinite reflector.

Create a dipole antenna of length, 0.15 m, and width, 0.015 m.

```
d = dipole('Length',0.15,'Width',0.015, 'Tilt',90,'TiltAxis',[0 1 0]);
```

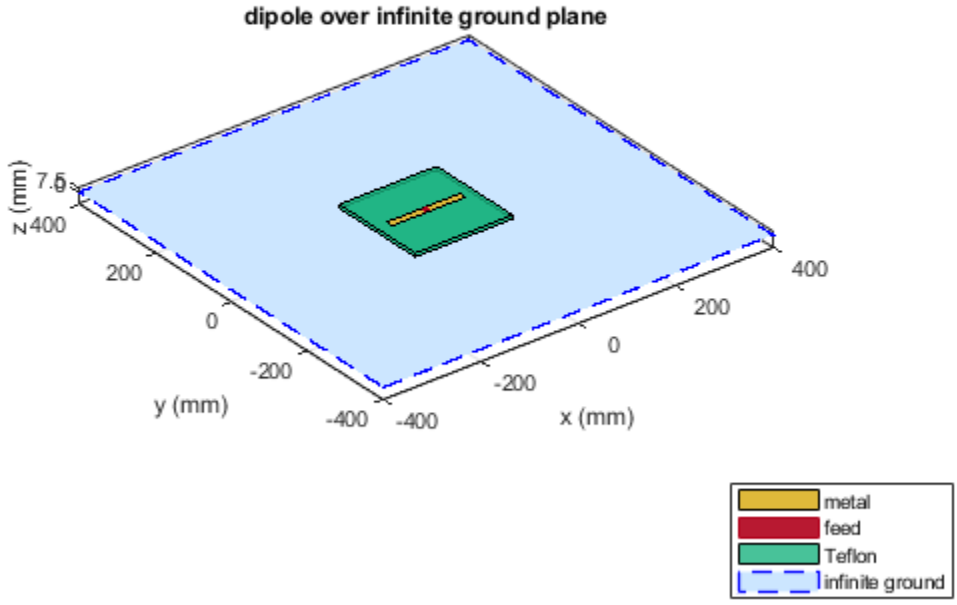
Create a reflector using the dipole antenna as an exciter and the dielectric, `teflon` as the substrate.

```
t = dielectric('Teflon')
rf = reflector('Exciter',d,'Spacing',7.5e-3,'Substrate',t);
```

```
t =

  dielectric with properties:

          Name: 'Teflon'
       EpsilonR: 2.1000
    LossTangent: 2.0000e-04
      Thickness: 0.0060

For more materials see <a href="matlab:openDielectricCatalog">catalog</a>
```

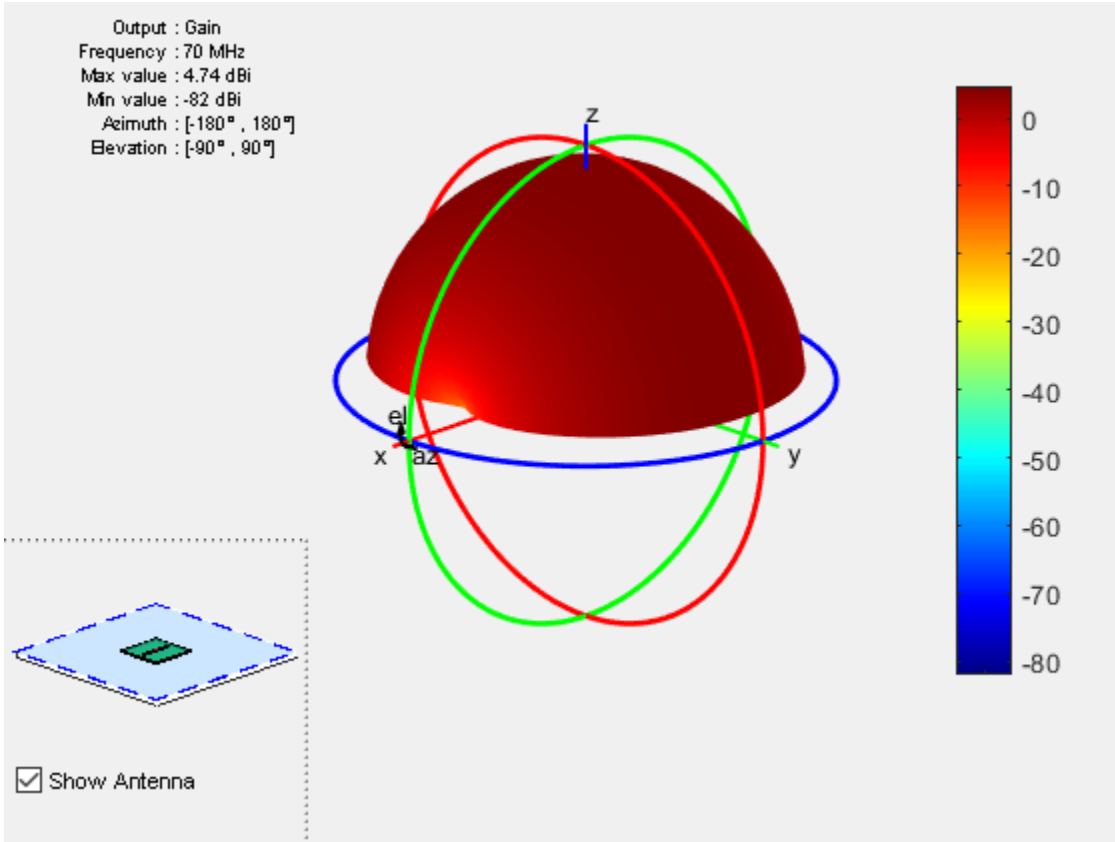
Set the groundplane length of the reflector to `inf`. View the structure.

```
rf.GroundPlaneLength = inf;
show(rf)
```



Calculate the radiation pattern of the antenna at 70 MHz.

```
pattern(rf,70e6)
```



References

[1] Balanis, C.A. *Antenna Theory: Analysis and Design*. 3rd Ed. New York: Wiley, 2005.

[2] Volakis, John. *Antenna Engineering Handbook*, 4th Ed. New York: Mcgraw-Hill, 2007.

See Also

`cylinder2strip` | `loopCircular` | `monopole` | `slot`

Topics

“Rotate Antenna and Arrays”

Introduced in R2015a

dipoleFolded

Create folded dipole antenna

Description

The `dipolefolded` object is a folded dipole antenna on the X-Y plane.

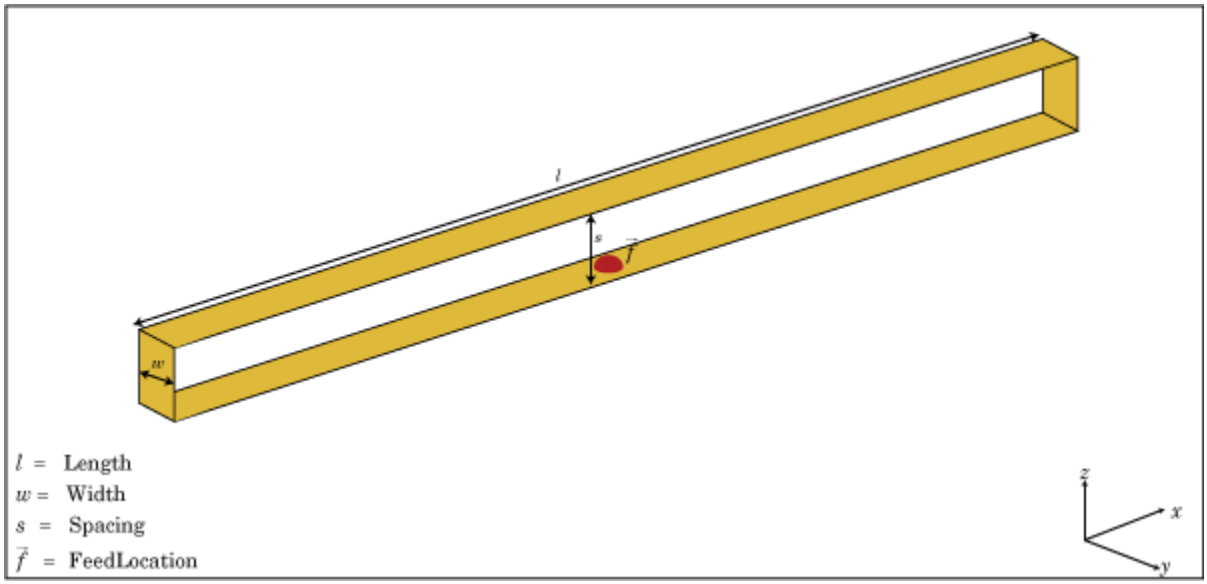
The width of the dipole is related to the diameter of an equivalent cylindrical dipole by the equation

$$w = 2d = 4r$$

, where

- d is the diameter of the equivalent cylindrical pole
- r is the radius of the equivalent cylindrical pole.

For a given cylinder radius, use the `cylinder2strip` utility function to calculate the equivalent width. The default folded dipole is center-fed. The feed point of the dipole coincides with the origin. The origin is located on the X-Y plane. When compared to the planar `dipole`, the folded dipole structure increases the input impedance of the antenna.



Creation

Syntax

```
dF = dipoleFolded
dF = dipoleFolded(Name,Value)
```

Description

`dF = dipoleFolded` creates a half-wavelength folded dipole antenna.

`dF = dipoleFolded(Name,Value)` creates a half-wavelength folded dipole antenna with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Length` — Folded dipole length

2 (default) | scalar

Folded dipole length, specified as a scalar in meters. By default, the length is chosen for an operating frequency of 70.5 MHz.

Example: `'Length',3`

Data Types: `double`

`Width` — Folded dipole width

0.0040 (default) | scalar

Folded dipole width, specified as a scalar in meters.

Note Folded dipole width should be less than `'Length'`/20 and greater than `'Length'`/1001. [2]

Example: `'Width',0.05`

Data Types: `double`

`Spacing` — Shorting stub lengths at dipole ends

0.0245 (default) | scalar

Shorting stub lengths at dipole ends, specified as a scalar in meters. The value must be less than `Length`/50.

Example: `'Spacing',3`

Data Types: `double`

`Load` — Lumped elements

[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `dF.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface

current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

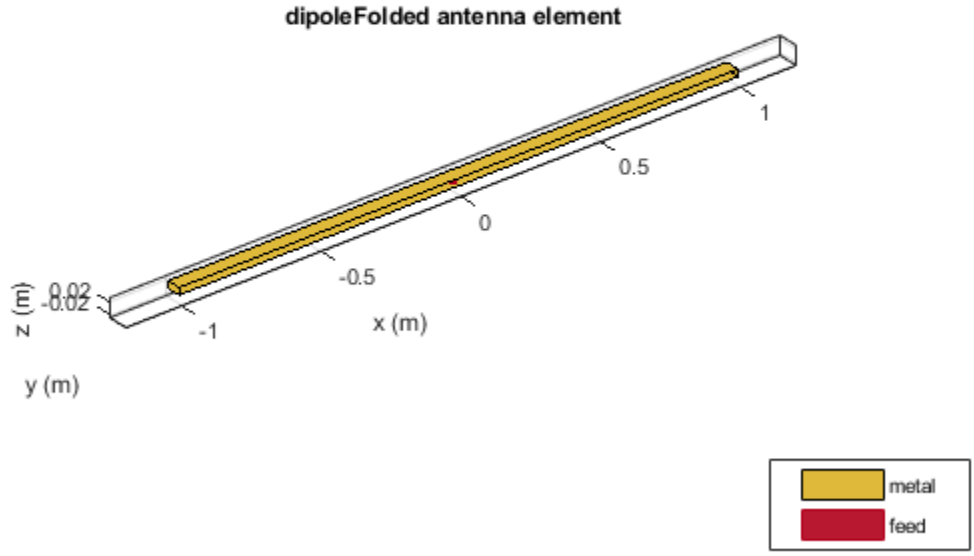
Create and View Folded Dipole Antenna

Create and view a folded dipole with 2m length and 0.05m width.

```
df = dipoleFolded('Length',2,'Width',0.05)
```

```
df =
  dipoleFolded with properties:

      Length: 2
       Width: 0.0500
     Spacing: 0.0245
        Tilt: 0
    TiltAxis: [1 0 0]
        Load: [1x1 lumpedElement]
```

```
show(df)
```



Radiation Pattern of Folded Dipole Antenna

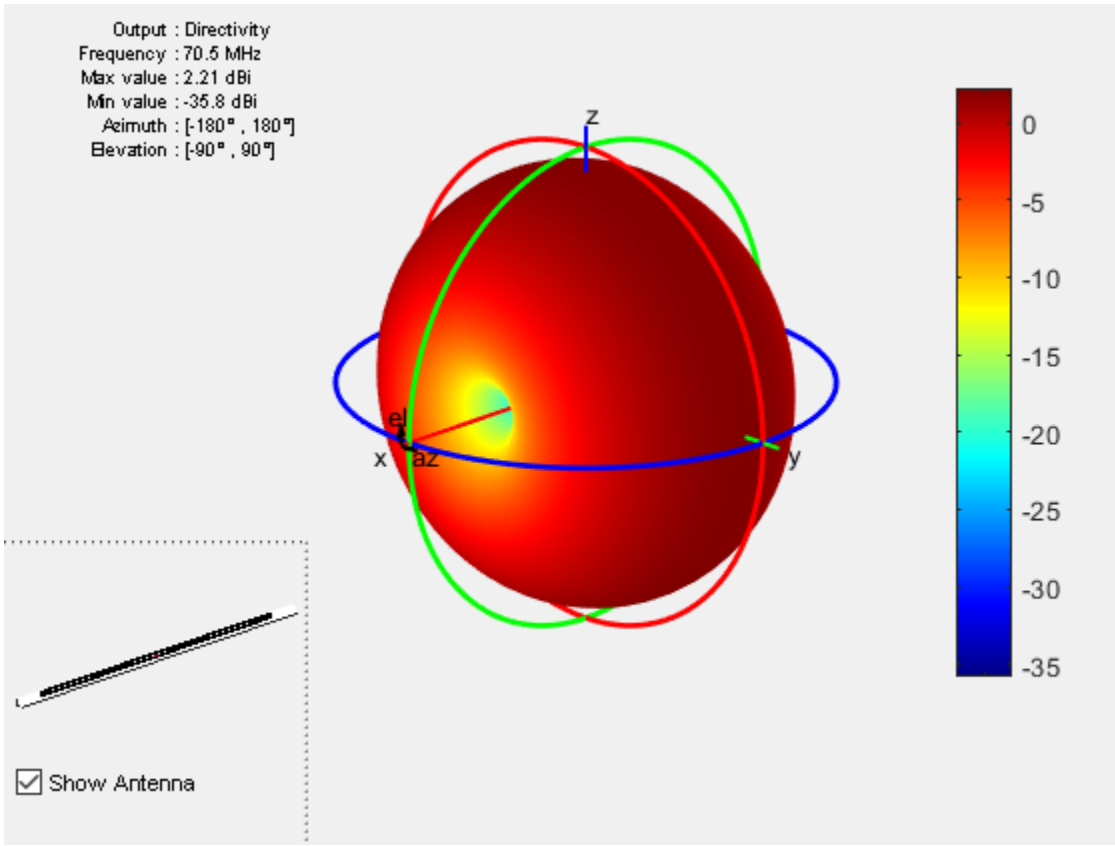
Plot the radiation pattern of a folded dipole at 70.5 MHz.

```
df = dipoleFolded
```

```
df =
  dipoleFolded with properties:

      Length: 2
       Width: 0.0180
     Spacing: 0.0245
```

```
        Tilt: 0
    TiltAxis: [1 0 0]
        Load: [1x1 lumpedElement]
```

```
pattern(df, 70.5e6);
```



References

[1] Balanis, C.A. *Antenna Theory: Analysis and Design*. 3rd Ed. New York: Wiley, 2005.

[2] Volakis, John. *Antenna Engineering Handbook*, 4th Ed. New York: Mcgraw-Hill, 2007.

See Also

`bowtieTriangular` | `cylinder2strip` | `dipole` | `monopole`

Topics

“Rotate Antenna and Arrays”

Introduced in R2015a

dipoleVee

Create V-dipole antenna

Description

The `dipoleVee` object is a planar V-dipole antenna in the X-Y plane.

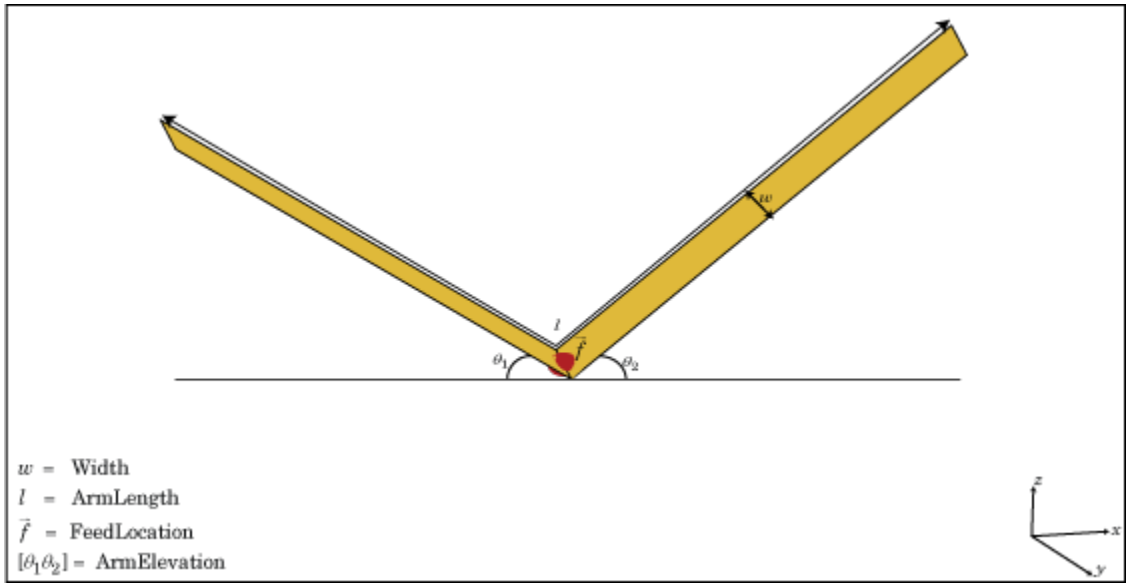
The width of the dipole is related to the circular cross-section by the equation

$$w = 2d = 4r$$

, where:

- d is the diameter of equivalent cylindrical pole
- r is the radius of equivalent cylindrical pole

For a given cylinder radius, use the `cylinder2strip` utility function to calculate the equivalent width. The V-dipole antenna is bent around the feed point. The default V-dipole is center-fed and is in the X-Y plane. The feed point of the V-dipole antenna coincides with the origin.



Creation

Syntax

```
dv = dipoleVee
dv = dipoleVee(Name,Value)
```

Description

`dv = dipoleVee` creates a half-wavelength V-dipole antenna.

`dv = dipoleVee(Name,Value)` creates a half-wavelength V-dipole antenna, with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`ArmLength` — Length of two arms
`[1 1]` (default) | two-element vector

Length of two arms, specified as a two-element vector in meters. By default, the arm lengths are chosen for an operating frequency of 75 MHz.

Example: `'ArmLength',[1,3]`

Data Types: `double`

`Width` — V-dipole arm width
0.1000 (default) | scalar

V-dipole arm width, specified as a scalar in meters.

Note Dipole width should be less than `Total Arm Length`/5 and greater than `Total Arm Length`/1001. [2]

Example: `'Width',0.05`

Data Types: `double`

`ArmElevation` — Angle made by two arms about X-Y plane
`[45 45]` (default) | two-element vector

Angle made by two arms about X-Y plane, specified as a two-element vector in degrees.

Example: `'ArmElevation',[55 35]`

Data Types: `double`

`Load` — Lumped elements
[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `dv.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface

design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

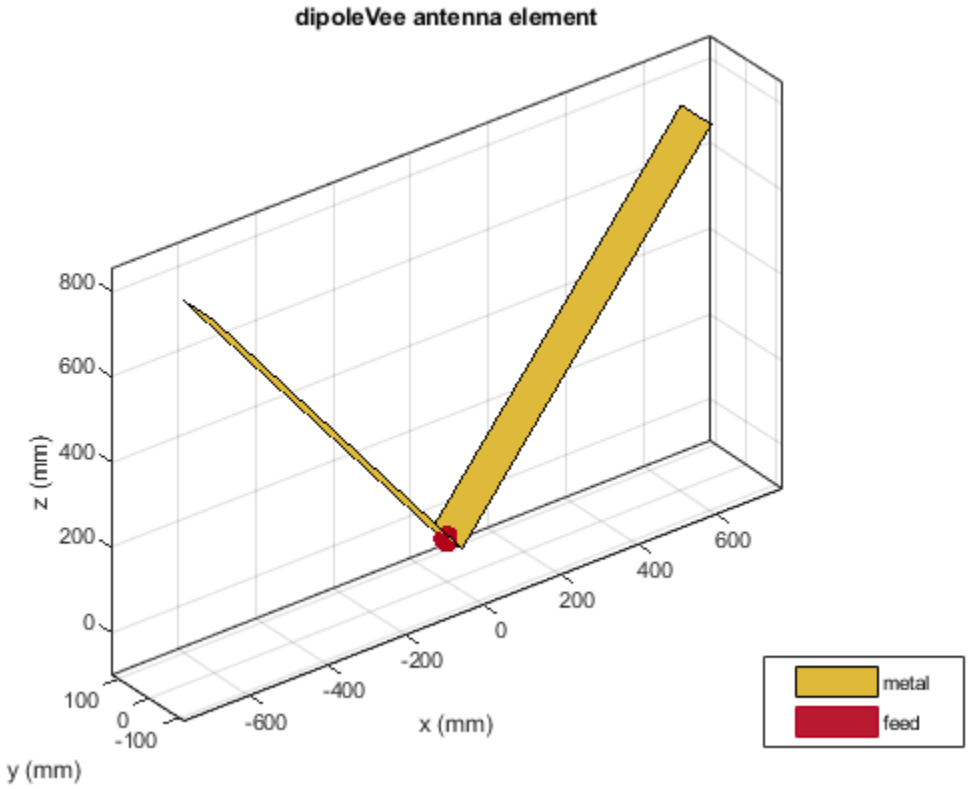
Create V-Dipole Antenna

Create and view a center-fed V-dipole that has 50 degree arm angles .

```
dv = dipoleVee('ArmElevation',[50 50])
```

```
dv =
  dipoleVee with properties:

       ArmLength: [1 1]
    ArmElevation: [50 50]
           Width: 0.1000
            Tilt: 0
        TiltAxis: [1 0 0]
            Load: [1x1 lumpedElement]
```

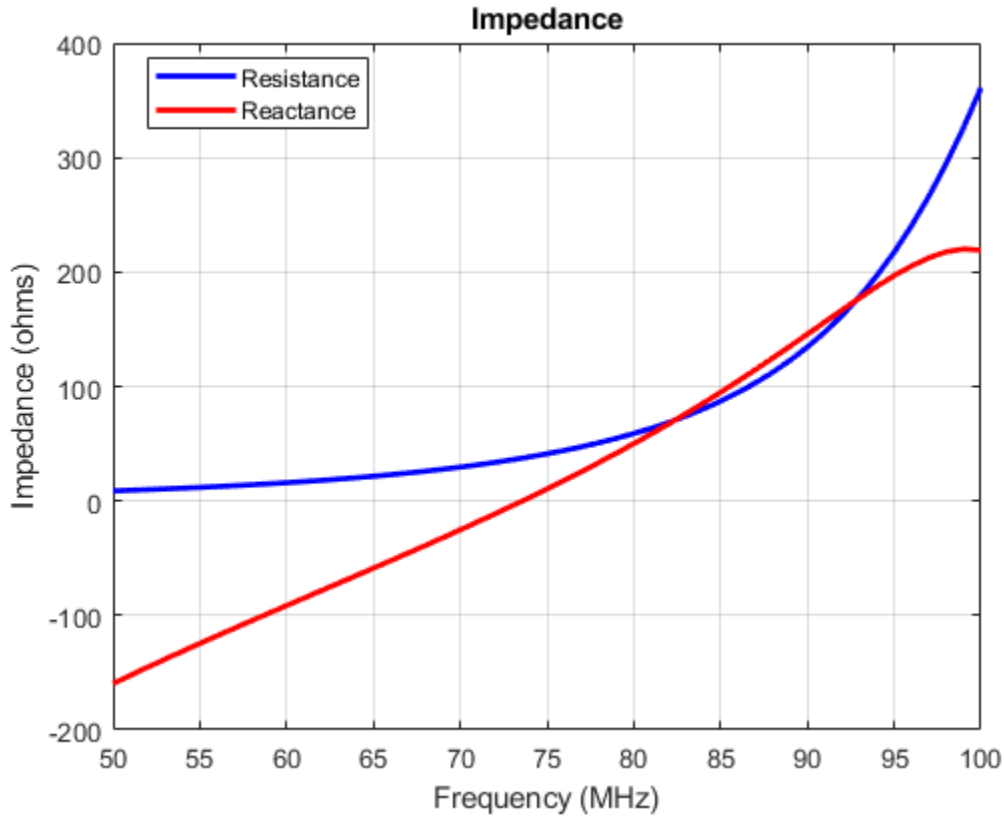
```
show(dv)
```



Impedance of V-Dipole Antenna

Calculate the impedance of a V-dipole antenna over the frequency range of 50MHz - 100MHz.

```
dv = dipoleVee('ArmElevation',[50 50]);
impedance(dv,linspace(50e6,100e6,51))
```



References

[1] Balanis, C.A. *Antenna Theory: Analysis and Design*. 3rd Ed. New York: Wiley, 2005.

[2] Volakis, John. *Antenna Engineering Handbook*. 4th Ed. New York: McGraw-Hill, 2007.

See Also

`cylinder2strip` | `dipole` | `dipoleFolded` | `loopCircular`

Topics

“Rotate Antenna and Arrays”

Introduced in R2015a

dipoleMeander

Create meander dipole antenna

Description

The `dipoleMeander` class creates a meander dipole antenna with four dipoles. The antenna is center fed and it is symmetric about its origin. The first resonance of meander dipole antenna is at 200 MHz.

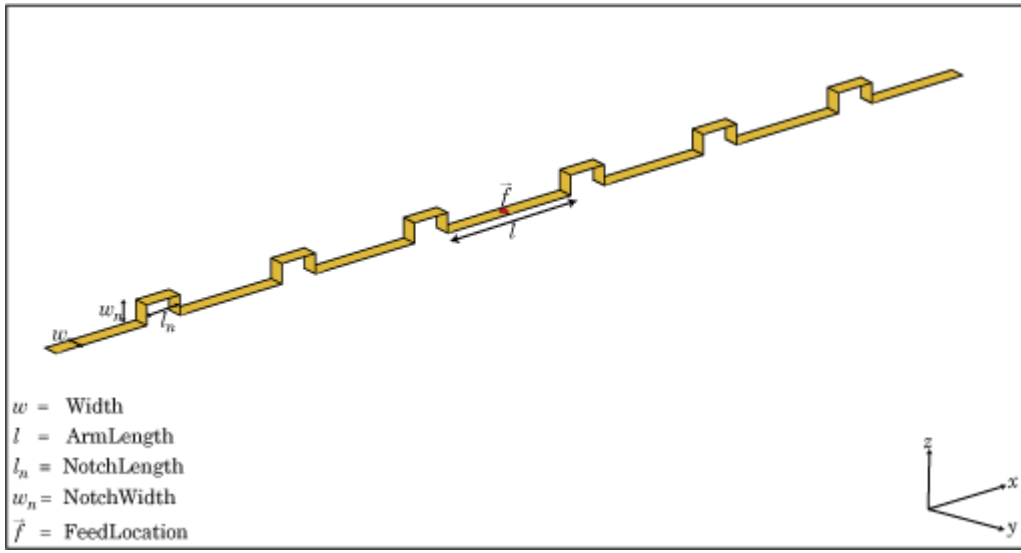
The width of the dipole is related to the diameter of an equivalent cylindrical dipole by the equation

$$w = 2d = 4r$$

, where:

- d is the diameter of equivalent cylindrical dipole.
- r is the radius of equivalent cylindrical dipole.

For a given cylinder radius, use the `cylinder2strip` utility function to calculate the equivalent width. The default strip dipole is center-fed. The feed point coincides with the origin. The origin is located on the X-Y plane.



Creation

Syntax

```
dm = dipoleMeander
dm = dipoleMeander(Name,Value)
```

Description

`dm = dipoleMeander` creates a meander dipole antenna with four dipoles.

`dm = dipoleMeander(Name,Value)` creates a meander dipole antenna with four dipoles, with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Width` — Dipole width
0.0040 (default) | scalar

Dipole width, specified as a scalar in meters.

Example: `'Width',0.05`

Data Types: `double`

`ArmLength` — Length of individual dipole arms
[0.0880 0.0710 0.0730 0.0650] (default) | vector

Length of individual dipole arms, specified as a vector in meters. The total number of dipole arms generated is :

$$2*N-1$$

where N is the number of specified arm lengths.

Example: `'ArmLength',[0.6000 0.5000 1 0.4000]`

Data Types: `double`

`NotchLength` — Notch length along length of antenna
0.0238 (default) | scalar

Notch length along the length of the antenna, specified as a scalar in meters.

For example, in a dipole meander antenna with seven stacked arms there are six notches.

Example: `'NotchLength',1`

Data Types: `double`

`NotchWidth` — Notch width perpendicular to length of antenna
0.0238 (default) | scalar

Notch width perpendicular to the length of the antenna, specified as a scalar in meters.

Example: `'NotchWidth',1`

Data Types: `double`

Load — Lumped elements

[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `dm.Load = lumpedElement('Impedance',75)`

Tilt — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

TiltAxis — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

Create and View Meander Dipole Antenna

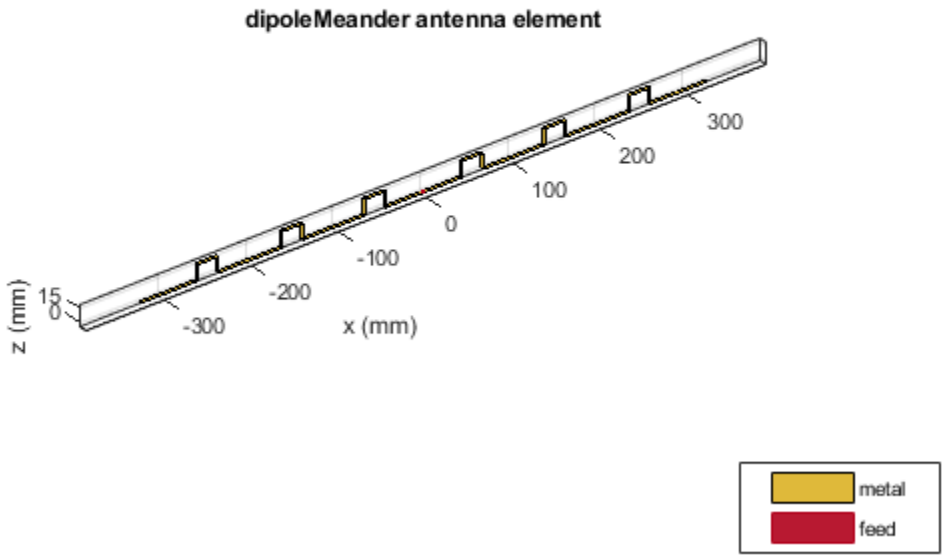
Create and view the default meander dipole antenna.

```
dm = dipoleMeander
```

```
dm =
  dipoleMeander with properties:

          Width: 0.0040
      ArmLength: [0.0880 0.0710 0.0730 0.0650]
    NotchLength: 0.0238
     NotchWidth: 0.0170
           Tilt: 0
       TiltAxis: [1 0 0]
```

```
            Load: [1x1 lumpedElement]
```

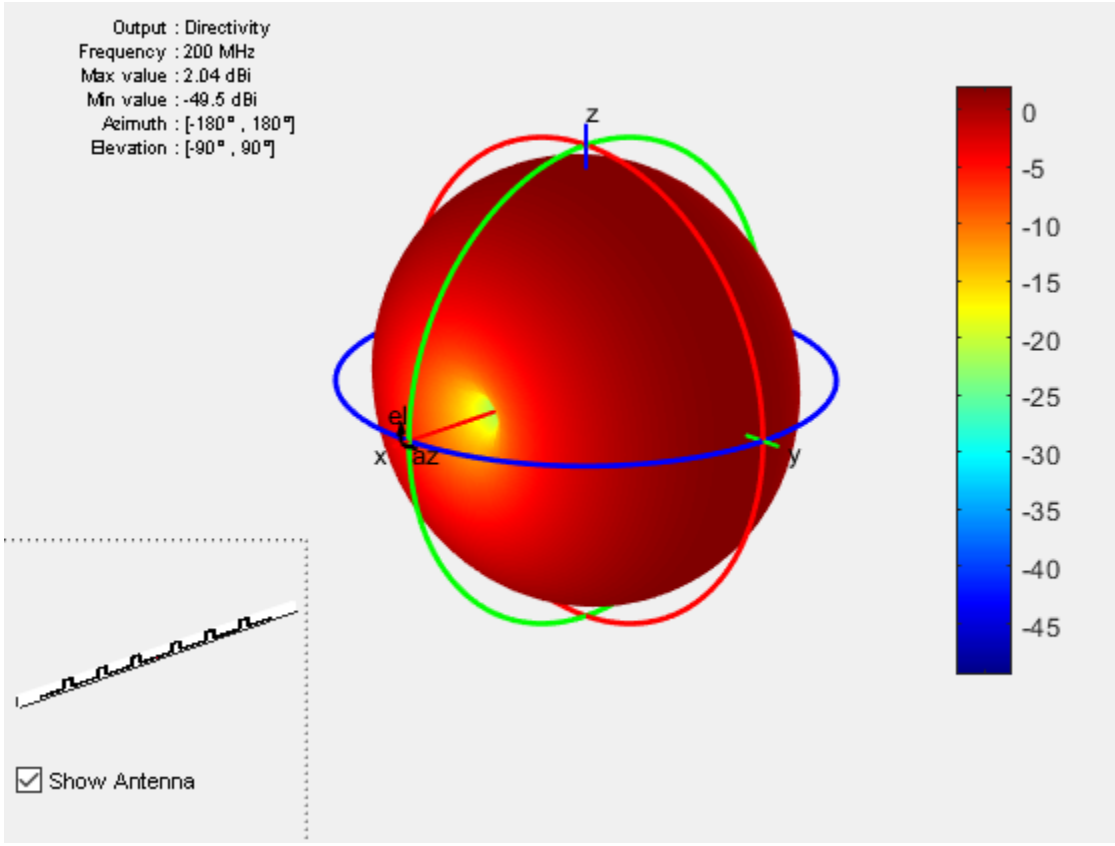
```
show(dm)
```



Plot Radiation Pattern Of Meander Dipole Antenna

Plot the radiation pattern of meander dipole antenna at a 200MHz frequency.

```
dm = dipoleMeander;
pattern(dm,200e6)
```



References

[1] Balanis, C.A. *Antenna Theory: Analysis and Design*. 3rd Ed. New York: Wiley, 2005.

See Also

`dipole` | `dipoleFolded` | `loopCircular`

Topics

“Rotate Antenna and Arrays”

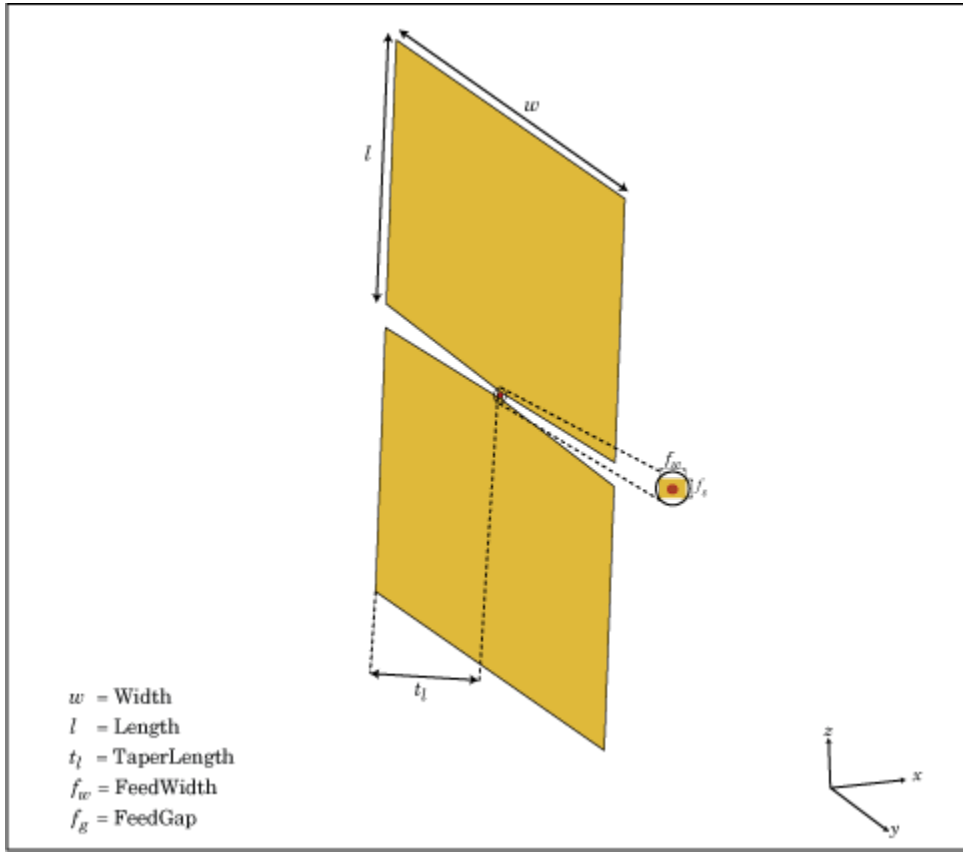
Introduced in R2015a

dipoleBlade

Create blade dipole antenna

Description

The `dipoleBlade` object is a wideband blade dipole antenna oriented along the X-Y plane.



The width of the dipole is related to the circular cross-section by the equation,

$$w = 2d = 4r$$

, where:

- d is the diameter of equivalent cylindrical pole
- r is the radius of equivalent cylindrical pole

For a given cylinder radius, use the `cylinder2strip` utility function to calculate the equivalent width.

Creation

Syntax

```
db = dipoleBlade
db = dipoleBlade(Name,Value)
```

Description

`db = dipoleBlade` creates a wideband blade dipole antenna on the X-Y plane.

`db = dipoleBlade(Name,Value)` creates a wideband blade dipole antenna, with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Length` — Blade dipole length
`0.1170` (default) | scalar

Blade dipole length, specified as a scalar in meters.

Example: `'Length',0.5`

Data Types: `double`

`Width` — Blade dipole width
`0.1400` (default) | scalar

Blade dipole width, specified as a scalar in meters.

Example: `'Width',0.2`

Data Types: `double`

`TaperLength` — Taper length
`0.1120` (default) | scalar

Taper length, specified as a scalar in meters.

Example: `'TaperLength',0.500`

Data Types: `double`

`FeedWidth` — Blade dipole feed width
`0.0030` (default) | scalar

Blade dipole feed width, specified as a scalar in meters.

Example: `'FeedWidth',0.006`

Data Types: `double`

`FeedGap` — Blade dipole feed length or distance between the two wings of the dipole
`0.0030` (default) | scalar

Blade dipole feed length or distance between the two wings of the dipole, specified as a scalar in meters.

Example: `'FeedGap',0.006`

Data Types: `double`

`Load` — Lumped elements
[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `db.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface

current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

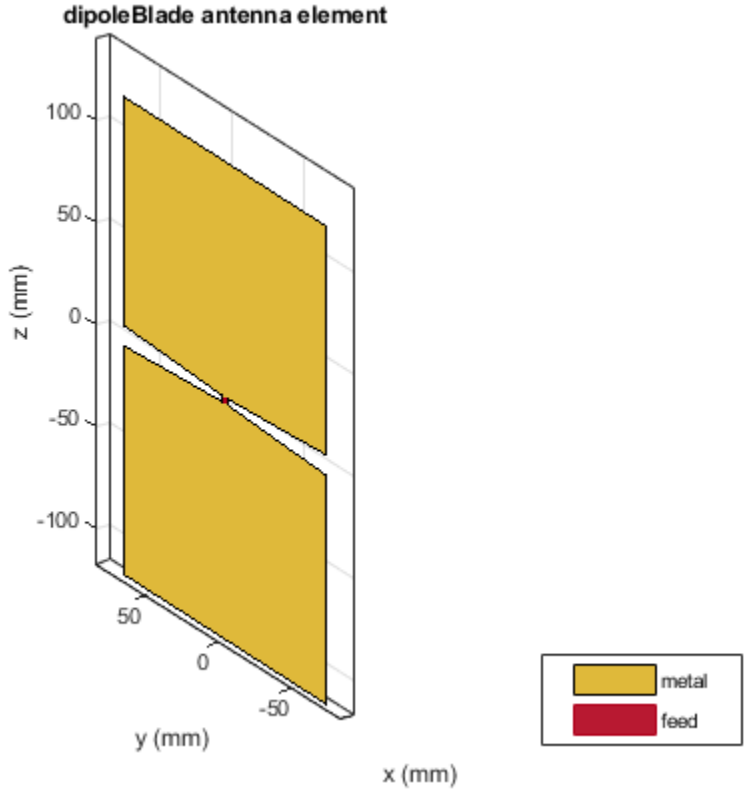
Default Blade Dipole and Radiation Pattern

Create and view a default blade dipole.

```
db = dipoleBlade
```

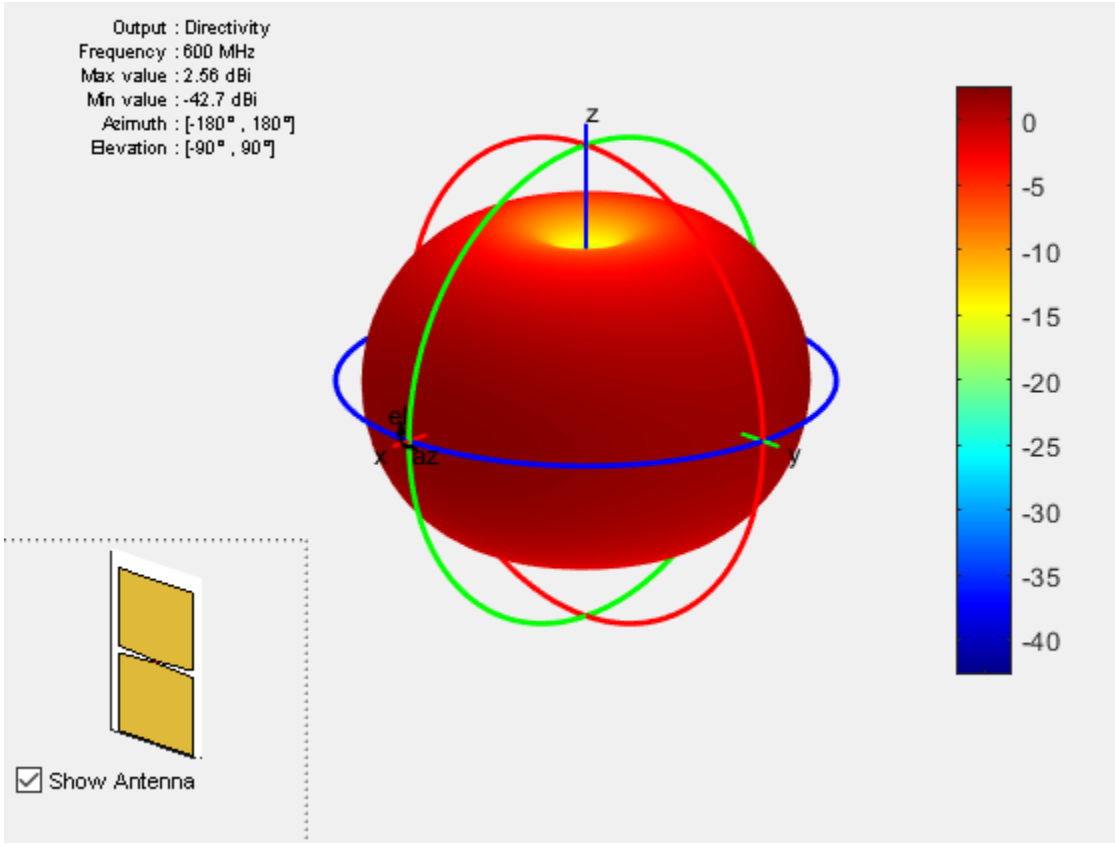
```
db =
  dipoleBlade with properties:

         Length: 0.1170
    TaperLength: 0.1120
          Width: 0.1400
      FeedWidth: 0.0030
        FeedGap: 0.0030
           Tilt: 0
       TiltAxis: [1 0 0]
           Load: [1x1 lumpedElement]
```

```
show(db);
```



Plot the radiation pattern of the blade dipole at 600 MHz.

```
pattern(db,600e6)
```



References

[1] Balanis, C.A. *Antenna Theory: Analysis and Design*. 3rd Ed. New York: Wiley, 2005.

[2] Volakis, John. *Antenna Engineering Handbook*. 4th Ed. New York: McGraw-Hill, 2007.

See Also

`cylinder2strip` | `dipole` | `loopCircular` | `slot`

Topics

“Rotate Antenna and Arrays”

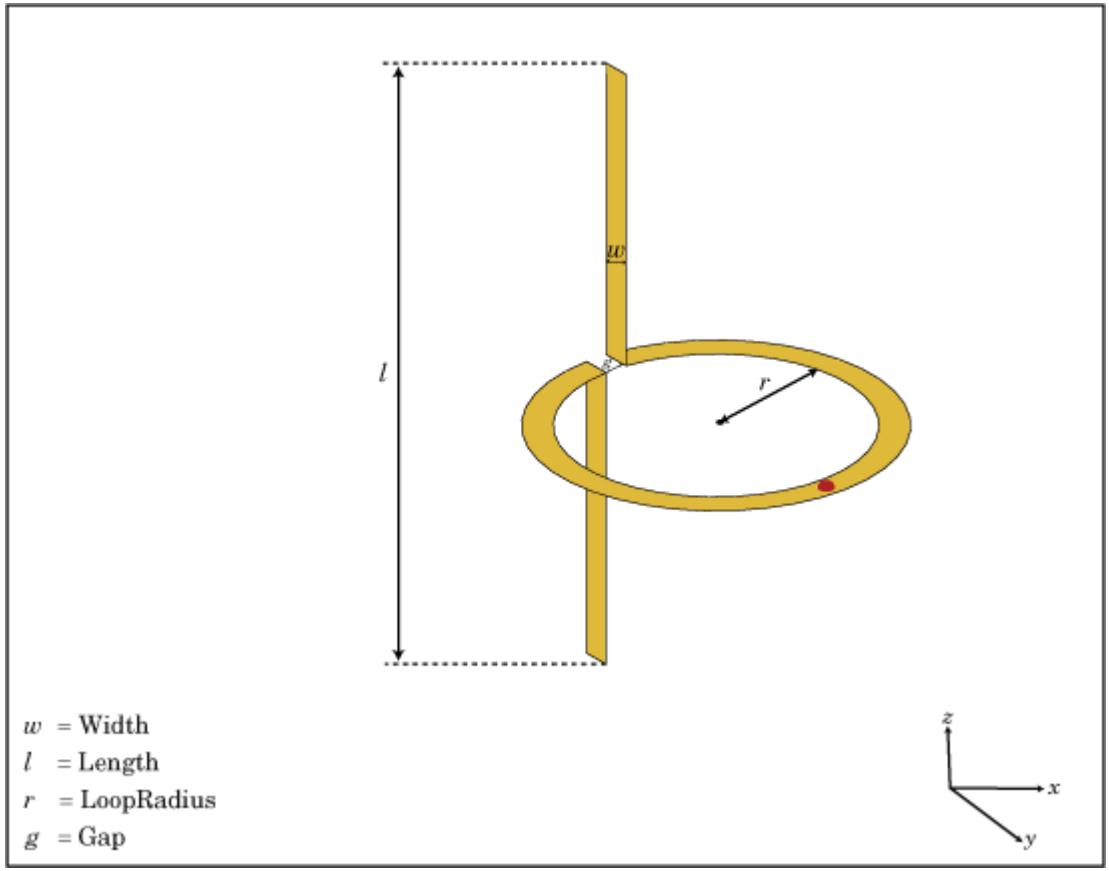
Introduced in R2017a

dipoleCycloid

Create cycloid dipole antenna

Description

The `dipoleCycloid` object is a half-wavelength cycloid dipole antenna. For the default cycloid dipole, the feed point is on the loop section. The default length is for an operating frequency of 48 MHz.



The width of the dipole is related to the circular cross-section by the equation

$$w = 2d = 4r$$

, where:

- d is the diameter of equivalent cylindrical pole
- r is the radius of equivalent cylindrical pole

For a given cylinder radius, use the `cylinder2strip` utility function to calculate the equivalent width.

Creation

Syntax

```
dc = dipoleCycloid
dc = dipoleCycloid(Name,Value)
```

Description

`dc = dipoleCycloid` creates a half-wavelength cycloid dipole antenna oriented along Z-axis.

`dc = dipoleCycloid(Name,Value)` creates a half-wavelength cycloid dipole antenna, with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Length` — Dipole length along z-axis

`1.2200` (default) | scalar

Dipole length along z-axis, specified as a scalar in meters. By default, the length is for an operating frequency of 48 MHz.

Example: `'Length',0.9`

Data Types: `double`

`Width` — Dipole width
`0.0508` (default) | scalar

Dipole width, specified as a scalar in meters.

Example: `'Width',0.09`

Data Types: `double`

`LoopRadius` — Circular loop radius in X-Y plane
`0.3100` (default) | scalar

Circular loop radius in X-Y plane, specified as a scalar in meters.

Example: `'LoopRadius',0.500`

Data Types: `double`

`Gap` — Gap of loop in X-Y plane
`0.0400` (default) | scalar

Gap of loop in X-Y plane, specified as a scalar in meters.

Example: `'Gap',0.006`

Data Types: `double`

`Load` — Lumped elements
[1x1 LumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as the comma-separated pair consisting of `'Load'` and a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `dc.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array

mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

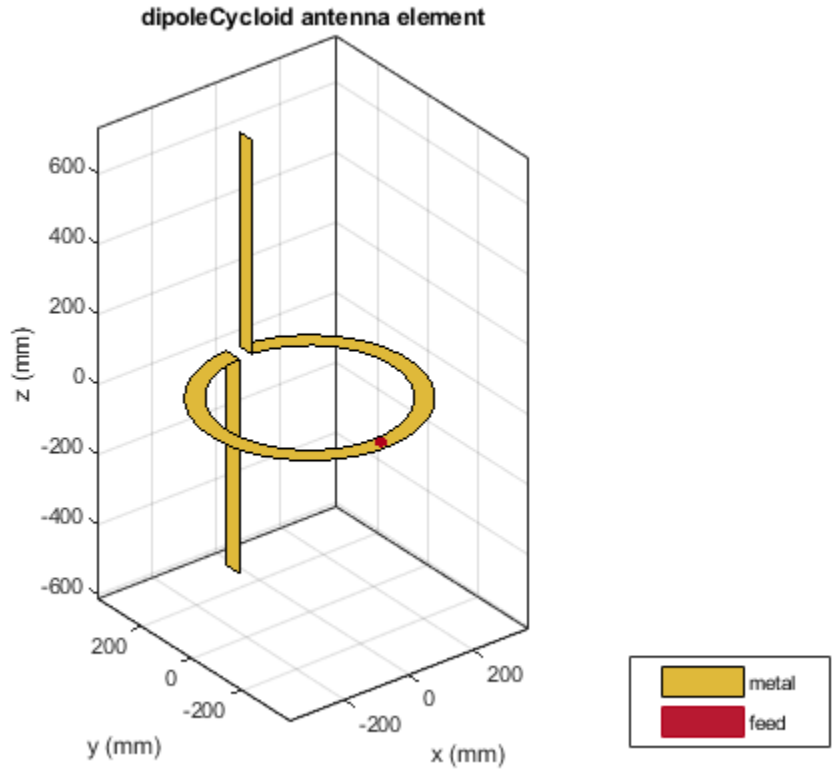
Default Cycloid Dipole

Create a default cycloid dipole antenna using the dipoleCycloid object and view it.

```
dc = dipoleCycloid
```

```
dc =
  dipoleCycloid with properties:

        Length: 1.2200
         Width: 0.0508
    LoopRadius: 0.3100
           Gap: 0.0400
          Tilt: 0
      TiltAxis: [1 0 0]
          Load: [1x1 lumpedElement]
```

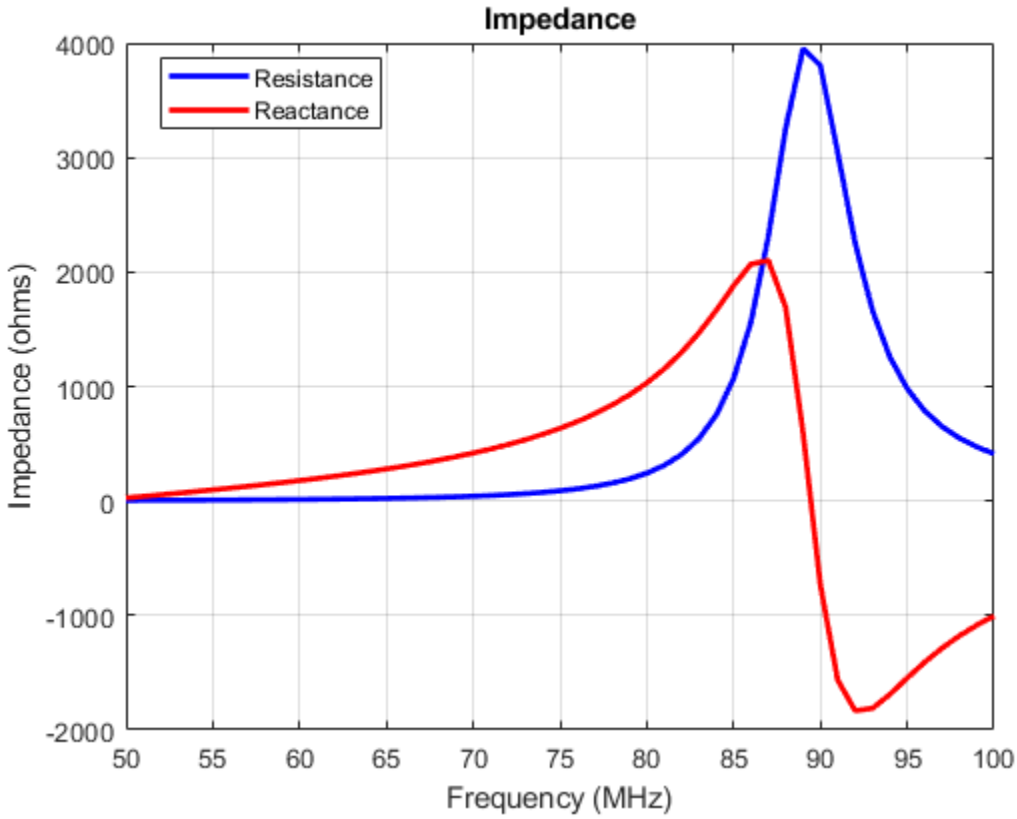
```
show(dc)
```



Impedance of Cycloid Dipole

Calculate the impedance of a cycloid dipole of width, 0.05 m, over a frequency span of 50 MHz - 100 MHz.

```
d = dipoleCycloid('Width',0.05);
impedance(d,linspace(50e6,100e6,51))
```



Radiation Pattern of Cycloid Dipole

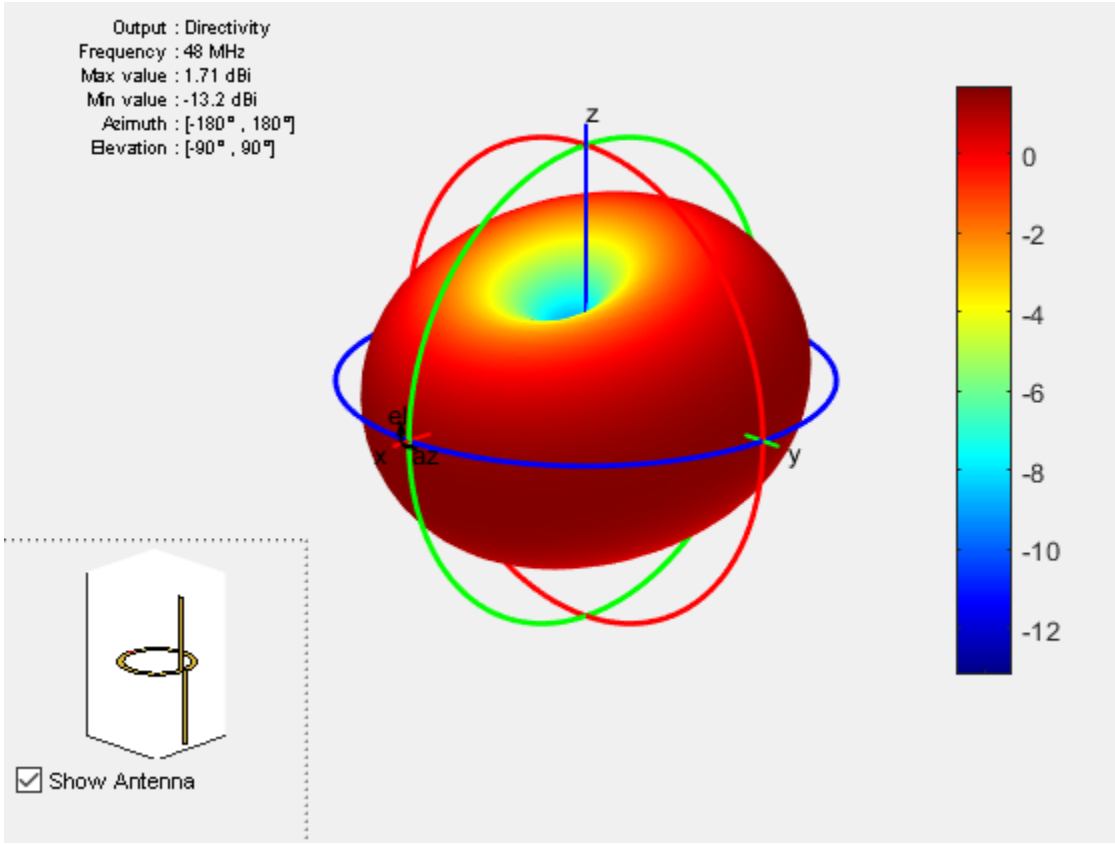
Plot the radiation pattern of a cycloid dipole of width,0.05 m, at a frequency of 48 MHz.

```
d = dipoleCycloid('Width',0.05)
```

```
d =
  dipoleCycloid with properties:

        Length: 1.2200
         Width: 0.0500
    LoopRadius: 0.3100
```

```
          Gap: 0.0400
         Tilt: 0
    TiltAxis: [1 0 0]
         Load: [1x1 lumpedElement]
```

```
pattern(d,48e6)
```



References

[1] Balanis, C.A. *Antenna Theory: Analysis and Design*. 3rd Ed. New York: Wiley, 2005.

[2] Volakis, John. *Antenna Engineering Handbook*. 4th Ed. New York: McGraw-Hill, 2007.

See Also

`cylinder2strip` | `dipole` | `loopCircular` | `slot`

Topics

“Rotate Antenna and Arrays”

Introduced in R2017a

dipoleHelix

Create helical dipole antenna

Description

The `dipoleHelix` object is a helical dipole antenna. The antenna is typically center-fed. You can move the feed along the antenna length using the feed offset property. Helical dipoles are used in satellite communications and wireless power transfers.

The width of the strip is related to the diameter of an equivalent cylinder by this equation

$$w = 2d = 4r$$

where:

- w is the width of the strip.
- d is the diameter of an equivalent cylinder.
- r is the radius of an equivalent cylinder.

For a given cylinder radius, use the `cylinder2strip` utility function to calculate the equivalent width. The default helical dipole antenna is center-fed. The circular ground plane is on the X-Y plane. Commonly, helical dipole antennas are used in axial mode. In this mode, the helical dipole circumference is comparable to the operating wavelength, and has maximum directivity along its axis. In normal mode, the helical dipole radius is small compared to the operating wavelength. In this mode, the helical dipole radiates broadside, that is, in the plane perpendicular to its axis. The basic equation for the helical dipole antenna is:

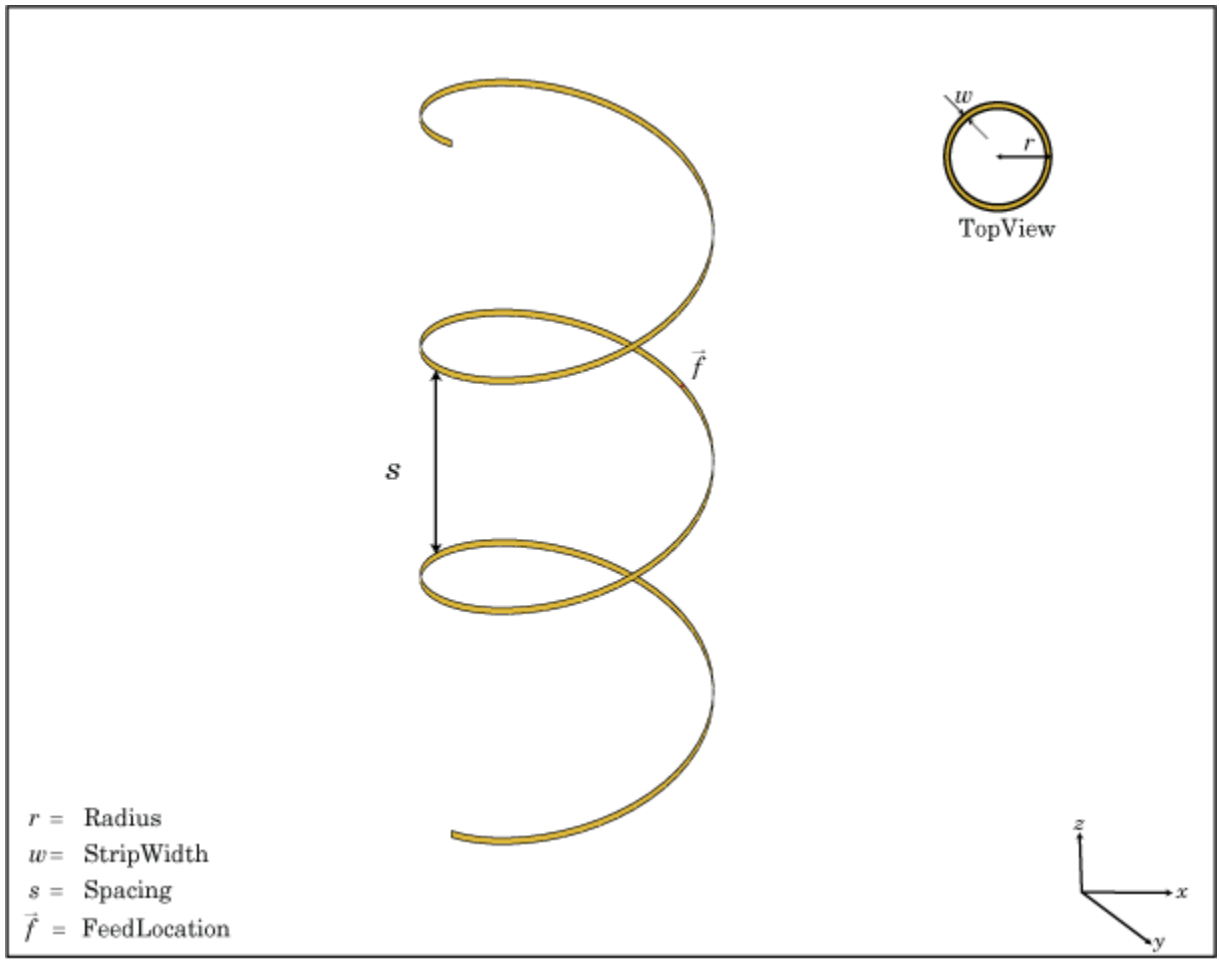
$$x = r\cos(\theta)$$
$$y = r\sin(\theta)$$
$$z = S\theta$$

where:

- r is the radius of the helical dipole.

- θ is the winding angle.
- S is the spacing between turns.

For a given pitch angle in degrees, use the `helixpitch2spacing` utility function to calculate the spacing between the turns in meters.



Creation

Syntax

```
dh = dipoleHelix
dh = dipoleHelix(Name,Value)
```

Description

`dh = dipoleHelix` creates a helical dipole antenna. The default antenna operates around 2 GHz.

`dh = dipoleHelix(Name,Value)` creates a helical dipole antenna, with additional properties specified by one or more name–value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Radius` — Turn radius
`0.0220` (default) | scalar

Turn radius, specified as a scalar in meters.

Example: `'Radius',2`

Data Types: `double`

`Width` — Strip width
`1.0000e-03` (default) | scalar

Strip width, specified as a scalar in meters.

Note Strip width should be less than `'Radius'`/5 and greater than `'Radius'`/250. [4]

Example: `'Width',5`

Data Types: `double`

`Turns` — Number of turns of helical dipole
`3` (default) | scalar

Number of turns of the helical dipole, specified a scalar.

Example: `'Turns',2`

Data Types: `double`

`Spacing` — Spacing between turns
`0.0350` (default) | scalar

Spacing between turns, specified as a scalar in meters.

Example: `'Spacing',1.5`

Data Types: `double`

`WindingDirection` — Direction of helical dipole turns (windings)
`'CCW'` (default) | `'CW'`

Direction of helical dipole turns (windings), specified as `'CW'` or `'CCW'`.

Example: `'WindingDirection','CW'`

Data Types: `char` | `string`

`Load` — Lumped elements
[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `dh.Load = lumpedElement('Impedance',75)`

`FeedOffset` — Signed distance of feedpoint from origin
`0` (default) | two-element vector

Signed distance from center along length and width of ground plane, specified as a two-element vector in meters. Use this property to adjust the location of the feedpoint relative to the ground plane and patch.

Example: `'FeedOffset',[0.01 0.01]`

Data Types: `double`

`Tilt` — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna

charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

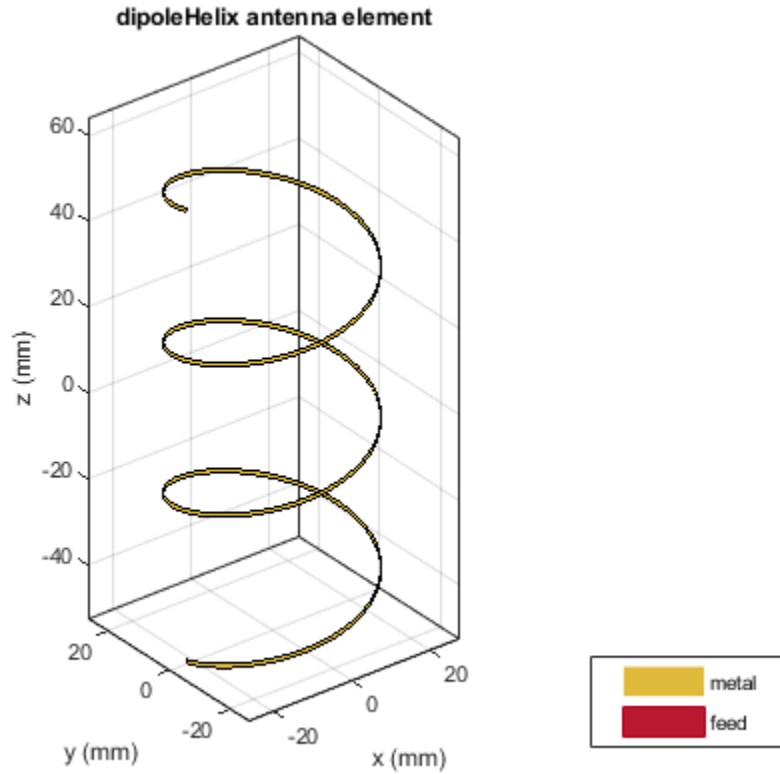
Helical Dipole Antenna

Create a default helical dipole antenna and view it.

```
dh = dipoleHelix
```

```
dh =
  dipoleHelix with properties:

              Radius: 0.0220
               Width: 1.0000e-03
               Turns: 3
             Spacing: 0.0350
    WindingDirection: 'CCW'
          FeedOffset: 0
                Tilt: 0
            TiltAxis: [1 0 0]
                Load: [1x1 lumpedElement]
```

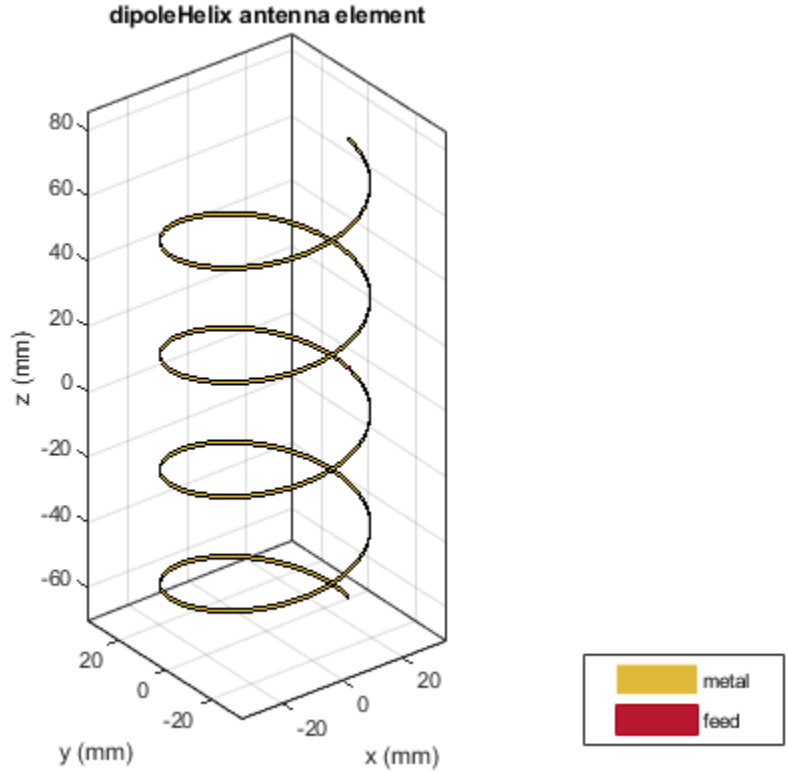
```
show(dh)
```



Radiation Pattern of Helical Dipole

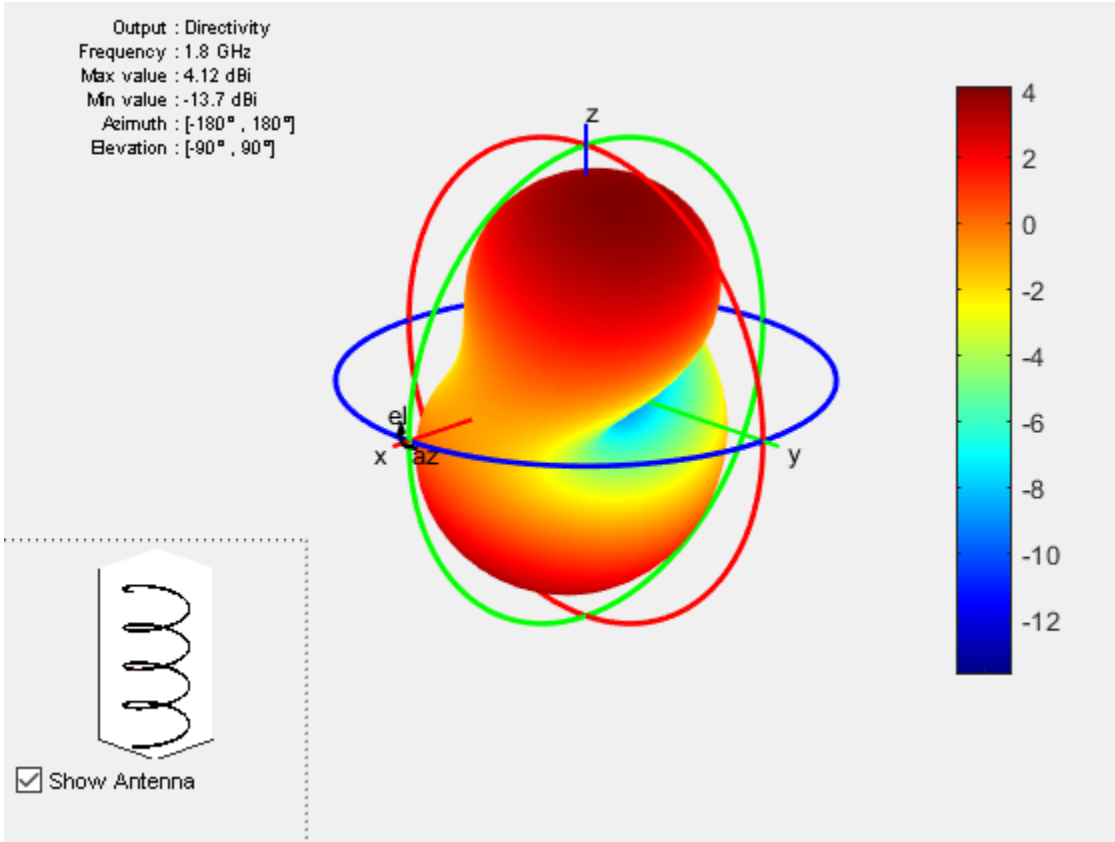
Create a four-turn helical dipole antenna with a turn radius of 28 mm and a strip width of 1.2 mm.

```
dh = dipoleHelix('Radius',28e-3,'Width',1.2e-3,'Turns',4);
show(dh)
```



Plot the radiation pattern of the helical dipole at 1.8 GHz.

```
pattern(dh, 1.8e9);
```



References

[1] Balanis, C. A. *Antenna Theory. Analysis and Design*. 3rd Ed. Hoboken, NJ: John Wiley & Sons, 2005.

[2] Volakis, John. *Antenna Engineering Handbook*. 4th Ed. New York: McGraw-Hill, 2007.

See Also

`cylinder2strip` | `helix` | `helixpitch2spacing` | `monopole` | `pifa` | `spiralArchimedean`

Topics

“Rotate Antenna and Arrays”

Introduced in R2016b

helix

Create helix antenna on ground plane

Description

The `helix` object is a helix antenna on a circular ground plane. The helix antenna is a common choice in satellite communication.

The width of the strip is related to the diameter of an equivalent cylinder by the equation

$$w = 2d = 4r$$

where:

- w is the width of the strip.
- d is the diameter of an equivalent cylinder.
- r is the radius of an equivalent cylinder.

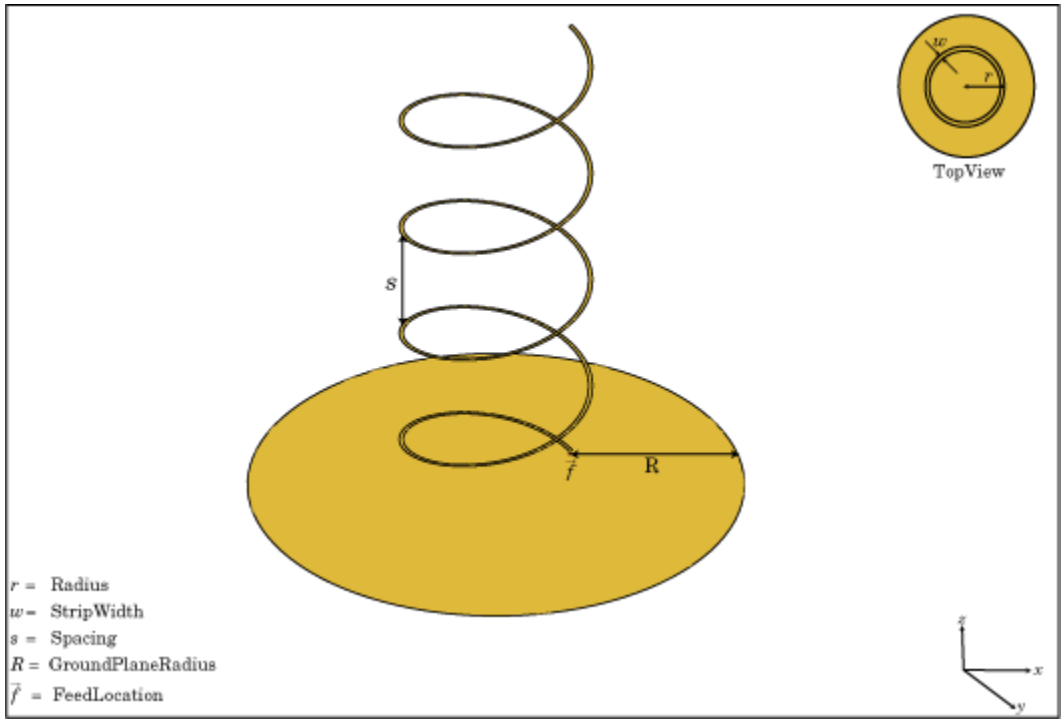
For a given cylinder radius, use the `cylinder2strip` utility function to calculate the equivalent width. The default helix antenna is end-fed. The circular ground plane is on the X-Y plane. Commonly, helix antennas are used in axial mode. In this mode, the helix circumference is comparable to the operating wavelength and the helix has maximum directivity along its axis. In normal mode, helix radius is small compared to the operating wavelength. In this mode, the helix radiates broadside, that is, in the plane perpendicular to its axis. The basic equation for the helix is

$$x = r\cos(\theta)$$
$$y = r\sin(\theta)$$
$$z = S\theta$$

where

- r is the radius of the helix.
- θ is the winding angle.
- S is the spacing between turns.

For a given pitch angle in degrees, use the `helixpitch2spacing` utility function to calculate the spacing between the turns in meters.



Creation

Syntax

```
hx = helix
hx = helix(Name,Value)
```

Description

`hx = helix` creates a helix antenna operating in axial mode. The default antenna operates around 2 GHz.

`hx = helix(Name,Value)` creates a helix antenna, with additional properties specified by one or more name–value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Radius` — Turn radius
0.0220 (default) | scalar

Turn radius, specified as a scalar in meters.

Example: `'Radius',2`

Data Types: `double`

`Width` — Strip width
1.0000e-03 (default) | scalar

Strip width, specified as a scalar in meters.

Note Strip width should be less than `'Radius'`/5 and greater than `'Radius'`/250. [4]

Example: `'Width',5`

Data Types: `double`

`Turns` — Number of turns of helix
3 (default) | scalar

Number of turns of the helix, specified as a scalar.

Example: `'Turns',2`

Data Types: `double`

`Spacing` — Spacing between turns
0.0350 (default) | scalar

Spacing between turns, specified as a scalar in meters.

Example: `'Spacing',1.5`

Data Types: `double`

`WindingDirection` — Direction of helix turns (windings)
`'CW'` | `'CCW'`

Direction of helix turns (windings), specified as `'CW'` or `'CCW'`.

Example: `'WindingDirection',CW`

Data Types: `char` | `string`

`GroundPlaneRadius` — Ground plane radius
0.0750 (default) | scalar in meters

Ground plane radius, specified as a scalar in meters. By default, the ground plane is on the X-Y plane and is symmetrical about the origin.

Example: `'GroundPlaneRadius',2.05`

Data Types: `double`

`FeedStubHeight` — Feeding stub height from ground
1.0000e-03 (default) | scalar

Feeding stub height from ground, specified as a scalar in meters. B

Example: `'FeedStubHeight',2.000e-03`

Note The default value is chosen to allow backward compatibility.

Data Types: `double`

`Load` — Lumped elements
[1x1 LumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. You can add a load anywhere on the surface of the antenna. By default, it is at the origin. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `hx.Load = lumpedElement('Impedance',75)`

Data Types: `double`

`Tilt` — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector in degrees.

Example: `'Tilt',90`

Example: `hx.Tilt = [90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna

`[1 0 0]` (default) | three-element vector of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X, Y, and Z axis.
- Two points in space as three-element vectors of Cartesian coordinates. In this case, the antenna rotates along the line joining the two points space.
- A string input describing simple rotations around the principal axis, X, Y, or Z.

For more information see, “Rotate Antenna and Arrays”

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `hx.TiltAxis = 'Z'`

Data Types: `double` | `char`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface

design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

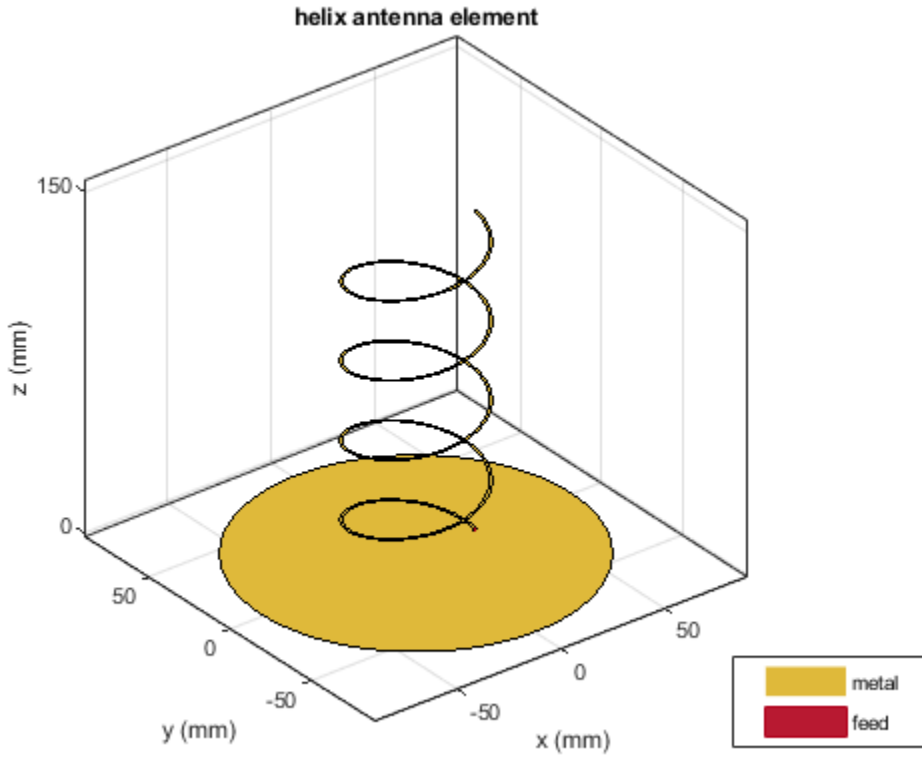
Create and View Helix Antenna

Create and view a helix antenna that has 28 mm turn radius, 1.2 mm strip width, and 4 turns.

```
hx = helix('Radius',28e-3,'Width',1.2e-3,'Turns',4)
```

```
hx =
  helix with properties:

               Radius: 0.0280
                Width: 0.0012
                Turns: 4
              Spacing: 0.0350
     WindingDirection: 'CCW'
       FeedStubHeight: 1.0000e-03
    GroundPlaneRadius: 0.0750
                 Tilt: 0
             TiltAxis: [1 0 0]
                 Load: [1x1 lumpedElement]
```

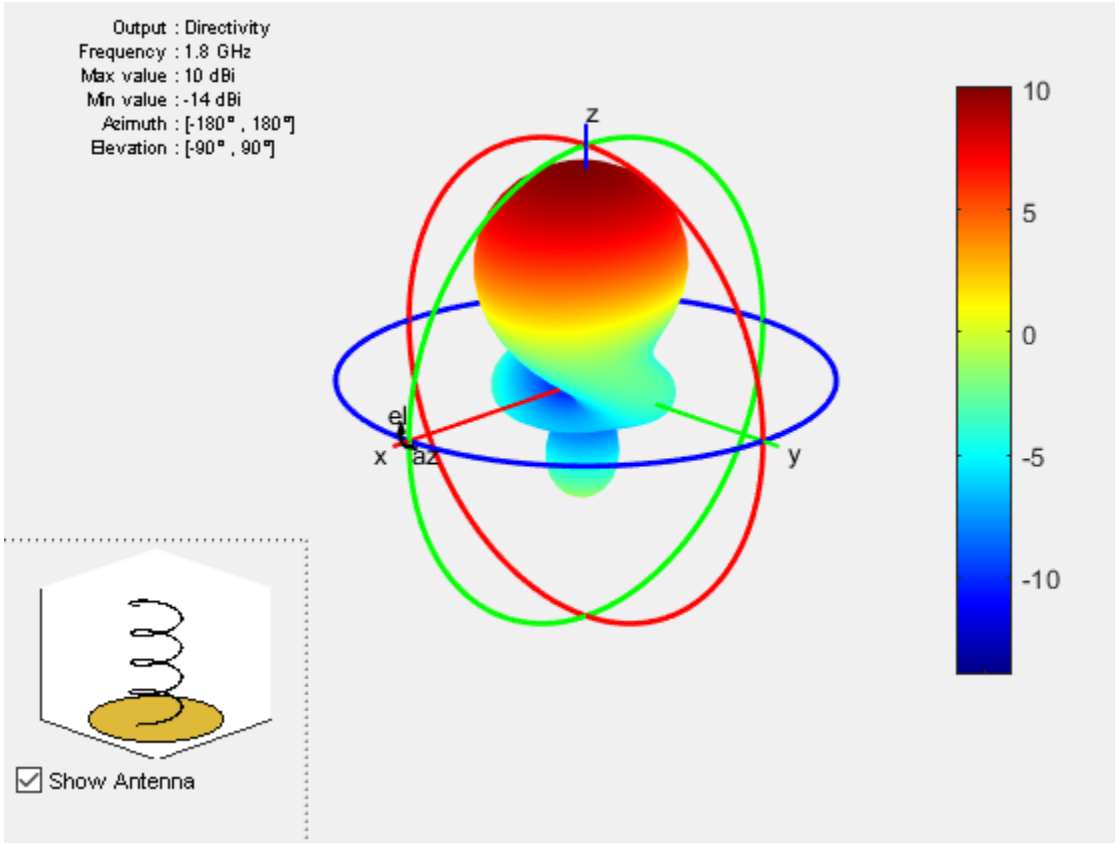
```
show(hx)
```



Radiation Pattern of Helix Antenna

Plot the radiation pattern of a helix antenna at a frequency of 1 GHz.

```
hx = helix('Radius',28e-3,'Width',1.2e-3,'Turns',4);
pattern(hx,1.8e9);
```



Calculate Spacing of Helix Antenna with Varying Radius

Calculate spacing of a helix that has a pitch of 12 degrees and a radius that varies from 20 mm to 22 mm in steps of 0.5 mm.

```
s = helixpitch2spacing(12,20e-3:0.5e-3:22e-3)
```

```
s = 1×5

    0.0267    0.0274    0.0280    0.0287    0.0294
```

References

[1] Balanis, C.A. *Antenna Theory. Analysis and Design*, 3rd Ed. New York: Wiley, 2005.

[2] Volakis, John. *Antenna Engineering Handbook*, 4th Ed. New York: Mcgraw-Hill, 2007.

[3] Zhang, Yan, Q. Ding, J. Chen, S. Lu, Z. Zhu and L. L. Cheng. "A Parametric Study of Helix Antenna for S-Band Satellite Communications." *9th International Symposium on Antenna Propagation and EM Theory (ISAPE)*. 2010, pp. 193–196.

[4] Djordjevic, A.R., Zajic, A.G., Ilic, M. M., Stuber, G.L. "Optimization of Helical antennas (Antenna Designer's Notebook)" *IEEE Antennas and Propagation Magazine*. December, 2006, pp. 107, pp.115.

See Also

`cylinder2strip` | `helixpitch2spacing` | `monopole` | `pifa` | `spiralArchimedean`

Topics

"Rotate Antenna and Arrays"

Introduced in R2015a

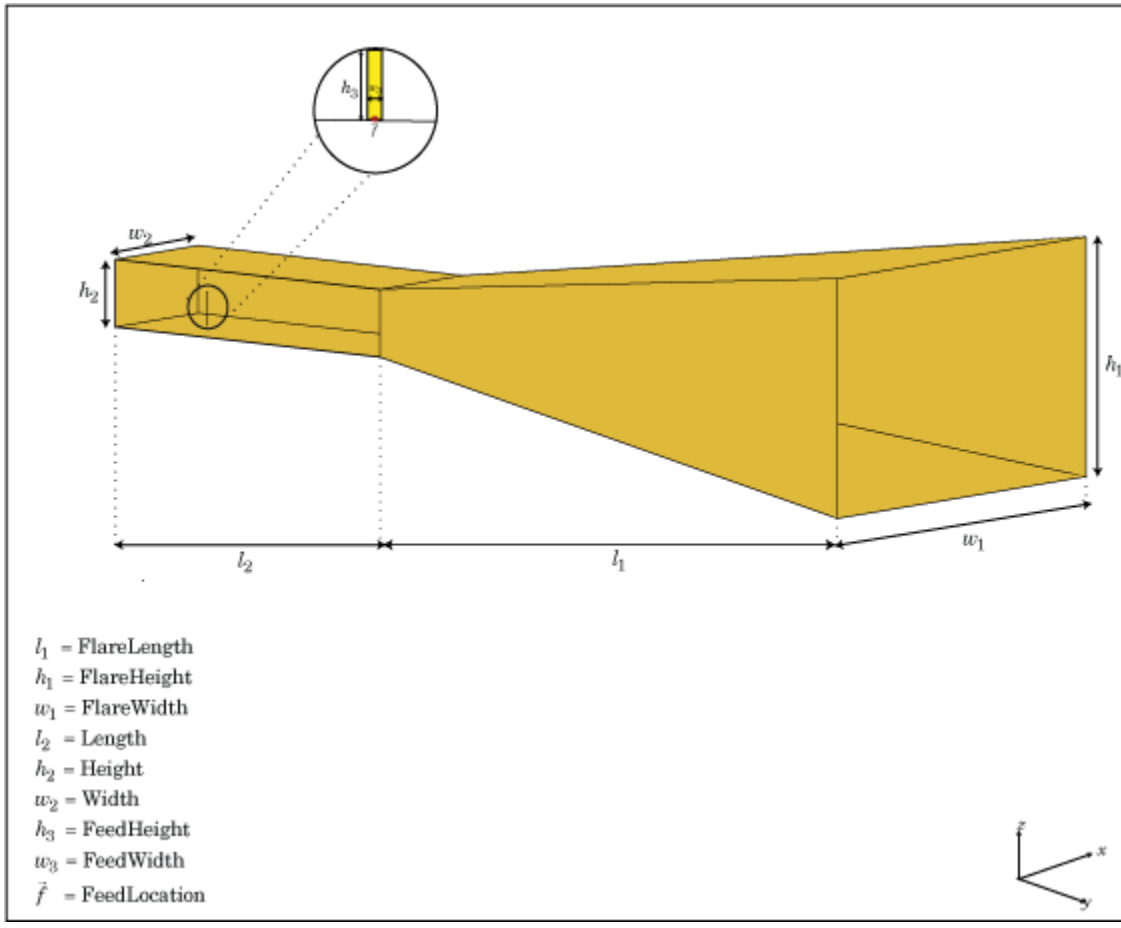
horn

Create horn antenna

Description

The `horn` object is a pyramidal horn antenna with a standard-gain, 15 dBi. The default horn antenna operates in the X-Ku band, which ranges from 10 GHz to 15 GHz. By default, the horn antenna feed is a WR-75 rectangular waveguide with an operating frequency at 7.87 GHz.

For a given flare angles of the horn and dimensions of the waveguide, use the `hornangle2size` utility function to calculate the equivalent flare width and flare height of the horn.



Creation

Syntax

```
hr = horn
hr = horn(Name,Value)
```

Description

`hr = horn` creates a standard-gain pyramidal horn antenna.

`hr = horn(Name,Value)` creates a horn antenna with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`. Properties not specified retain their default values.

Properties

`FlareLength` — Flare length of horn
`0.1020` (default) | scalar

Flare length of horn, specified as a scalar in meters.

Example: `'FlareLength',0.35`

Data Types: `double`

`FlareWidth` — Flare width of horn
`0.0571` (default) | scalar

Flare width of horn, specified as a scalar in meters.

Example: `'FlareWidth',0.2`

Data Types: `double`

`FlareHeight` — Flare height of horn
`0.0338` (default) | scalar

Flare height of horn, specified as a scalar in meters.

Example: `'FlareHeight',0.15`

Data Types: `double`

`Length` — Rectangular waveguide length
`0.0500` (default) | scalar

Rectangular waveguide length, specified as a scalar in meters.

Example: `'Length',0.09`

Data Types: `double`

`Width` — Rectangular waveguide width
`0.0190` (default) | scalar

Rectangular waveguide width, specified as a scalar in meters.

Example: `'Width',0.05`

Data Types: `double`

`Height` — Rectangular waveguide height
`0.0095` (default) | scalar

Rectangular waveguide height, specified as a scalar in meters.

Example: `'Height',0.0200`

Data Types: `double`

`FeedHeight` — Height of feed
`0.0048` (default) | scalar

Height of feed, specified as a scalar in meters.

Example: `'FeedHeight',0.0050`

Data Types: `double`

`FeedWidth` — Width of feed
`1.0000e-04` (default) | scalar

Width of feed, specified as a scalar in meters.

Example: `'FeedWidth',5e-05`

Data Types: `double`

`FeedOffset` — Signed offset of feedpoint from center of ground plane
`[–0.0155 0]` (default) | two-element vector

Signed offset from center of ground plane, specified as a two-element vector in meters.

Example: `'FeedOffset',[–0.0070 0.01]`

Data Types: `double`

Load — Lumped elements

[1x1 LumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `hr.Load = lumpedElement('Impedance',75)`

Tilt — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

TiltAxis — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

Default Horn Antenna

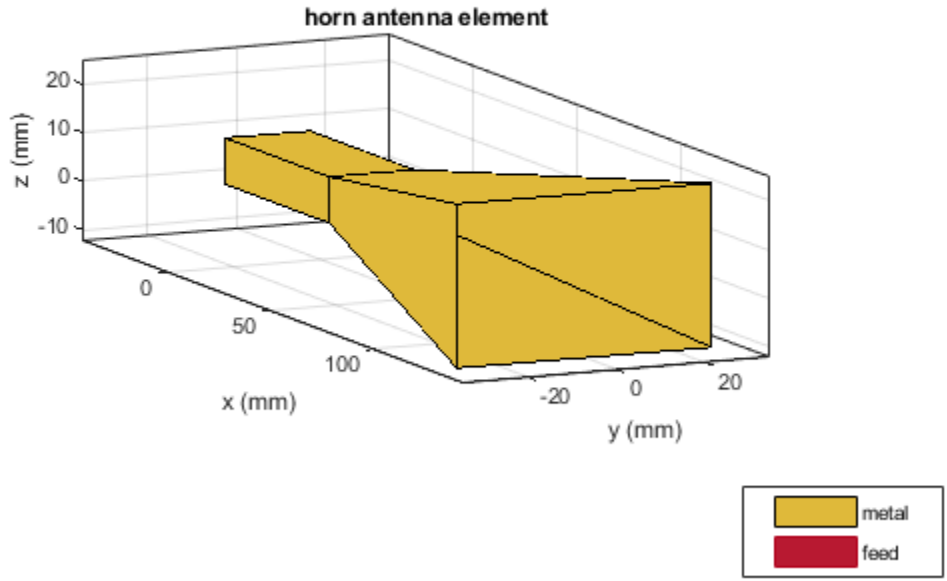
Create and view a default horn antenna.

```
h = horn
```

```
h =
  horn with properties:

    FlareLength: 0.1020
     FlareWidth: 0.0571
    FlareHeight: 0.0338
         Length: 0.0500
          Width: 0.0190
         Height: 0.0095
```

```
 FeedWidth: 1.0000e-04
FeedHeight: 0.0048
FeedOffset: [-0.0155 0]
      Tilt: 0
  TiltAxis: [1 0 0]
      Load: [1x1 lumpedElement]
```

```
show(h)
```



References

[1] Balanis, Constantine A.*Antenna Theory. Analysis and Design*. 3rd Ed. New York: John Wiley and Sons, 2005.

See Also

`waveguide`

Topics

“Rotate Antenna and Arrays”

Introduced in R2016a

invertedF

Create inverted-F antenna over rectangular ground plane

Description

The `invertedF` object is an inverted-F antenna mounted over a rectangular ground plane.

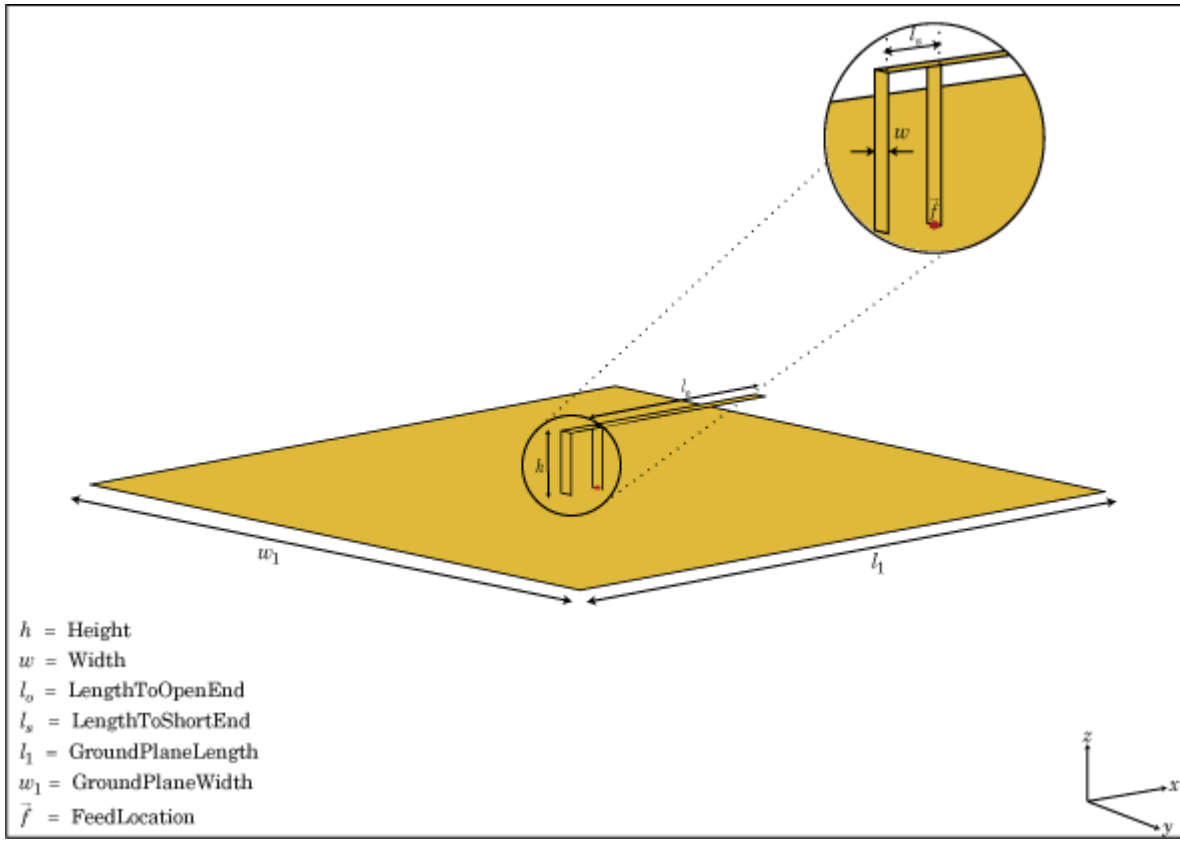
The width of the metal strip is related to the diameter of an equivalent cylinder by the equation

$$w = 2d = 4r$$

where:

- d is the diameter of equivalent cylinder
- r is the radius of equivalent cylinder

For a given cylinder radius, use the utility function `cylinder2strip` to calculate the equivalent width. The default inverted-F antenna is center-fed. The feed point coincides with the origin. The origin is located on the X-Y plane.



Creation

Syntax

```
f = invertedF
f = invertedF(Name,Value)
```

Description

`f = invertedF` creates an inverted-F antenna mounted over a rectangular ground plane. By default, the dimensions are chosen for an operating frequency of 1.7 GHz.

`f = invertedF(Name,Value)` creates an inverted-F antenna, with additional properties specified by one, or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, ..., `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Height` — Vertical element height along z-axis

0.0140 (default) | scalar

Vertical element height along z-axis, specified a scalar in meters.

Example: `'Height',3`

Data Types: `double`

`Width` — Strip width

0.0020 (default) | scalar

Strip width, specified as a scalar in meters.

Note Strip width should be less than `'Height'`/4 and greater than `'Height'`/1001. [2]

Example: `'Width',0.05`

Data Types: `double`

`LengthToOpenEnd` — Stub length from feed to open end

0.0310 (default) | scalar

Stub length from feed to open end, specified as a scalar in meters.

Example: `'LengthToOpenEnd',0.05`

`LengthToShortEnd` — Stub length from feed to shorting end
0.0060 (default) | scalar

Stub length from feed to shorting end, specified as a scalar in meters.

Example: `'LengthToShortEnd',0.0050`

`GroundPlaneLength` — Ground plane length along x-axis
0.1000 (default) | scalar

Ground plane length along x-axis, specified as a scalar in meters. Setting `'GroundPlaneLength'` to Inf, uses the infinite ground plane technique for antenna analysis.

Example: `'GroundPlaneLength',4`

Data Types: `double`

`GroundPlaneWidth` — Ground plane width along y-axis
0.1000 (default) | scalar

Ground plane width along y-axis, specified as a scalar in meters. Setting `'GroundPlaneWidth'` to Inf, uses the infinite ground plane technique for antenna analysis.

Example: `'GroundPlaneWidth',2.5`

Data Types: `double`

`FeedOffset` — Signed distance from center along length and width of ground plane
`[0 0]` (default) | two-element vector

Signed distance from center along length and width of ground plane, specified as a two-element vector.

Example: `'FeedOffset',[2 1]`

Data Types: `double`

`Load` — Lumped elements
[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `f.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna

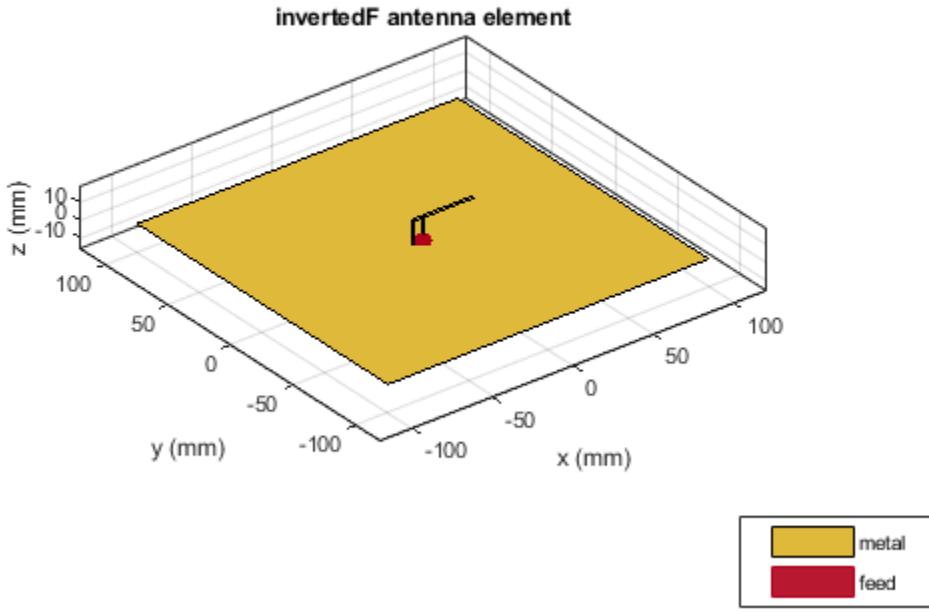
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

Create and View Inverted-F Antenna

Create and view an inverted-F antenna with 14mm height over a ground plane of dimensions 200mmx200mm.

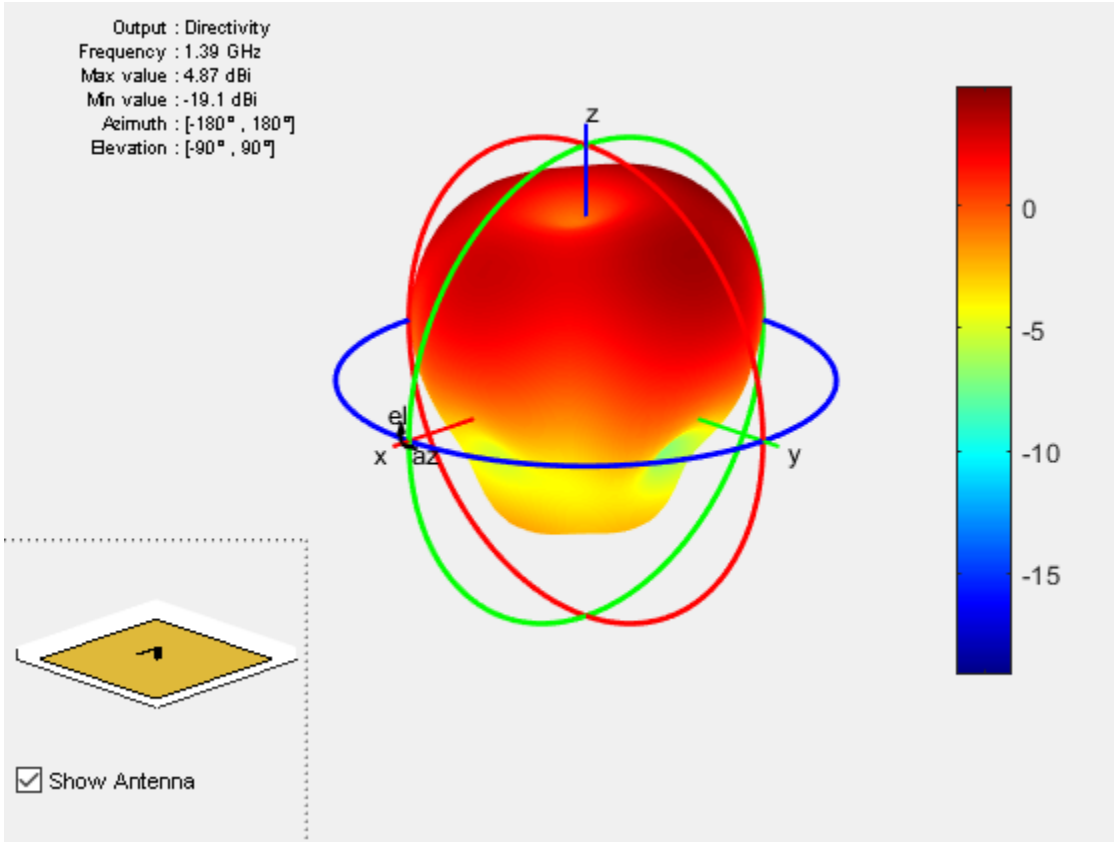
```
f = invertedF('Height',14e-3, 'GroundPlaneLength',200e-3,          ...
                  'GroundPlaneWidth',200e-3);
show(f)
```



Plot Radiation Pattern of Inverted-F

This example shows you how to plot the radiation pattern of an inverted-F antenna for a frequency of 1.3GHz.

```
f = invertedF('Height',14e-3, 'GroundPlaneLength', 200e-3,          ...
                  'GroundPlaneWidth', 200e-3);
pattern(f,1.39e9)
```



References

[1] Balanis, C.A. *Antenna Theory. Analysis and Design*, 3rd Ed. New York: Wiley, 2005.

[2] Volakis, John. *Antenna Engineering Handbook*, 4th Ed. New York: Mcgraw-Hill, 2007.

See Also

`cylinder2strip` | `invertedL` | `patchMicrostrip` | `pifa`

Topics

“Rotate Antenna and Arrays”

Introduced in R2015a

invertedL

Create inverted-L antenna over rectangular ground plane

Description

The `invertedL` object is an inverted-L antenna mounted over a rectangular ground plane.

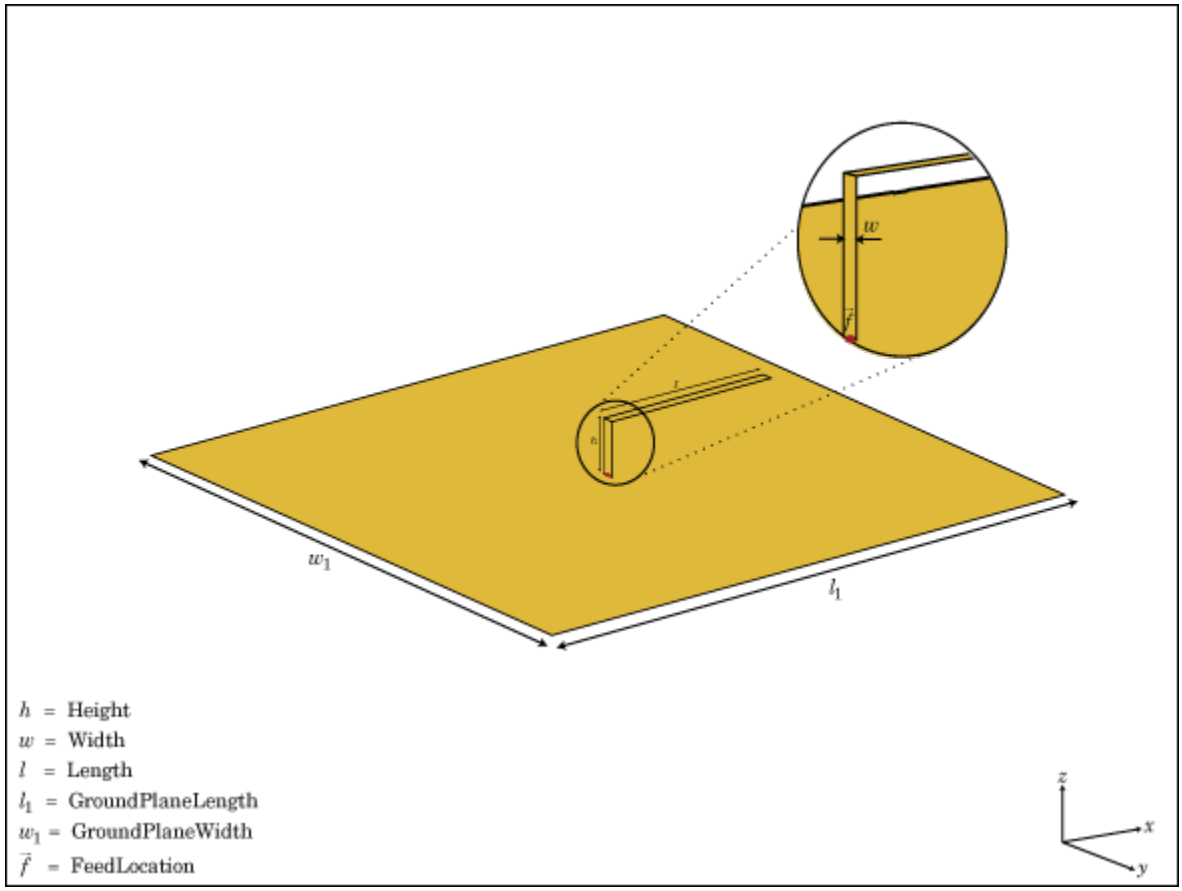
The width of the metal strip is related to the diameter of an equivalent cylinder by the equation

$$w = 2d = 4r$$

where:

- d = diameter of equivalent cylinder
- a = radius of equivalent cylinder

For a given cylinder radius, use the `cylinder2strip` utility function to calculate the equivalent width. The default inverted-L antenna is center-fed. The feed point coincides with the origin. The origin is located on the X-Y plane.



Creation

Syntax

```
l = invertedL
l = invertedL(Name,Value)
```

Description

`l = invertedL` creates an inverted-L antenna mounted over a rectangular ground plane. By default, the dimensions are chosen for an operating frequency of 1.7 GHz.

`l = invertedL(Name,Value)` creates an inverted-L antenna, with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Height` — Height of inverted element along z-axis

0.0140 (default) | scalar

Height of inverted element along z-axis, specified a scalar in meters.

Example: `'Height',3`

Data Types: `double`

`Width` — Strip width

0.0020 (default) | scalar

Strip width, specified as a scalar in meters.

Note Strip width should be less than `'Height'`/4 and greater than `'Height'`/1001. [2]

Example: `'Width',0.05`

Data Types: `double`

`Length` — Stub length along x-axis

0.0310 (default) | scalar

Stub length along x-axis, specified as a scalar in meters.

Example: `'Length',0.01`

`GroundPlaneLength` — Ground plane length along x-axis

0.1000 (default) | scalar

Ground plane length along x-axis, specified a scalar in meters. Setting `'GroundPlaneLength'` to `Inf`, uses the infinite ground plane technique for antenna analysis.

Example: `'GroundPlaneLength',4`

Data Types: `double`

`GroundPlaneWidth` — Ground plane width along y-axis

0.1000 (default) | scalar

Ground plane width along y-axis, specified as a scalar in meters. Setting `'GroundPlaneWidth'` to `Inf`, uses the infinite ground plane technique for antenna analysis.

Example: `'GroundPlaneWidth',2.5`

Data Types: `double`

`FeedOffset` — Signed distance from center along length and width of ground plane

`[0 0]` (default) | two-element vector

Signed distance from center along length and width of ground plane, specified as a two-element vector.

Example: `'FeedOffset',[2 1]`

Data Types: `double`

`Load` — Lumped elements

[1x1 LumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `l.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays

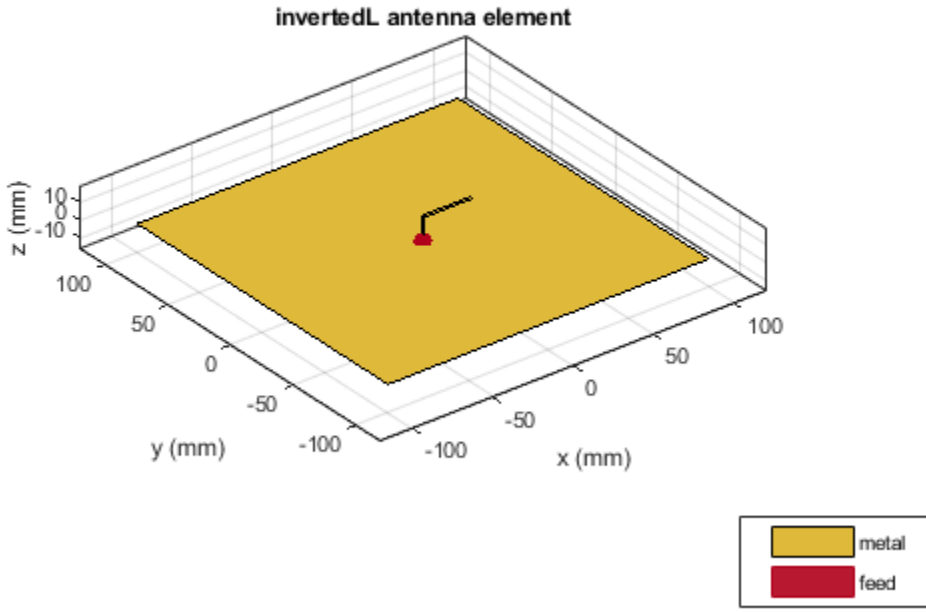
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

Create and View Inverted-L Antenna

Create and view an inverted-L antenna that has 30mm length over a ground plane of dimensions 200mmx200mm.

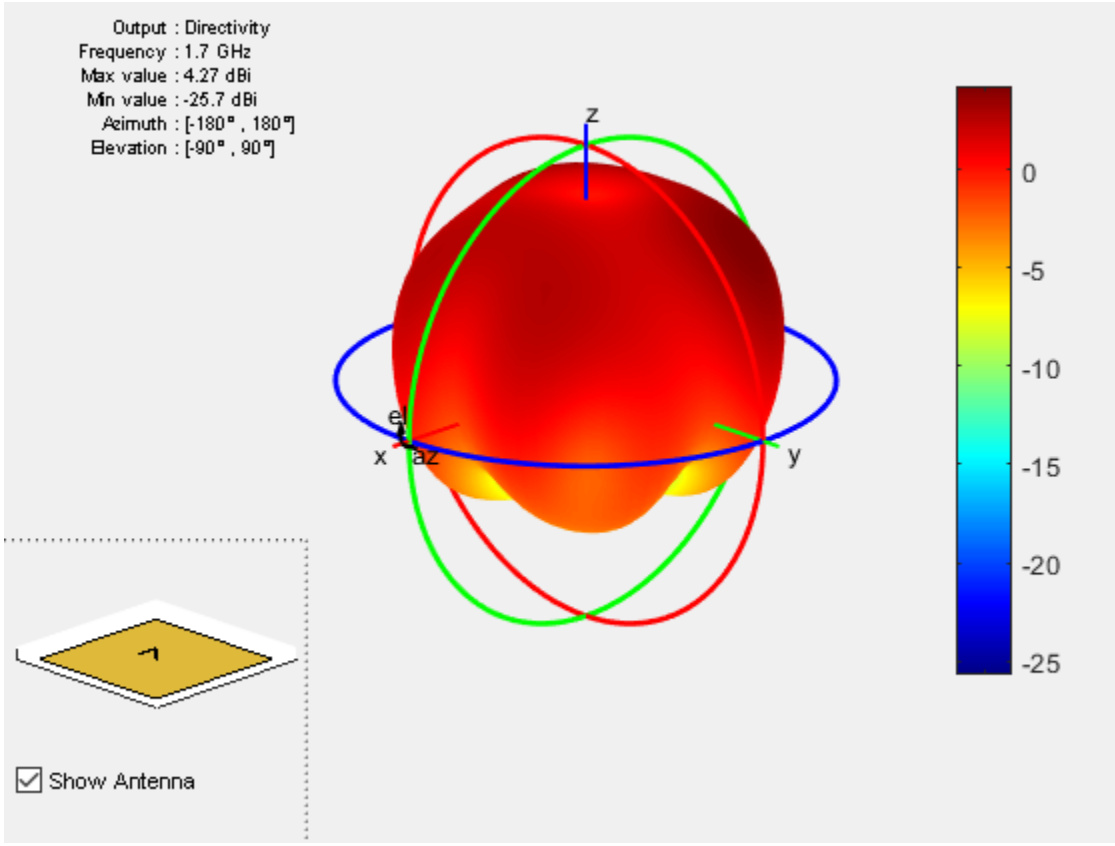
```
il = invertedL('Length',30e-3, 'GroundPlaneLength',200e-3,...
                  'GroundPlaneWidth',200e-3);
show(il)
```



Radiation Pattern of Inverted-L Antenna

Plot the radiation pattern of an inverted-L at a frequency of 1.7GHz.

```
iL = invertedL('Length',30e-3, 'GroundPlaneLength',200e-3,...
                    'GroundPlaneWidth',200e-3);
pattern(iL,1.7e9)
```



References

[1] Balanis, C.A. *Antenna Theory. Analysis and Design*, 3rd Ed. New York: Wiley, 2005.

[2] Volakis, John. *Antenna Engineering Handbook*, 4th Ed. New York: Mcgraw-Hill, 2007.

See Also

`cylinder2strip` | `invertedF` | `patchMicrostrip` | `pifa`

Topics

“Rotate Antenna and Arrays”

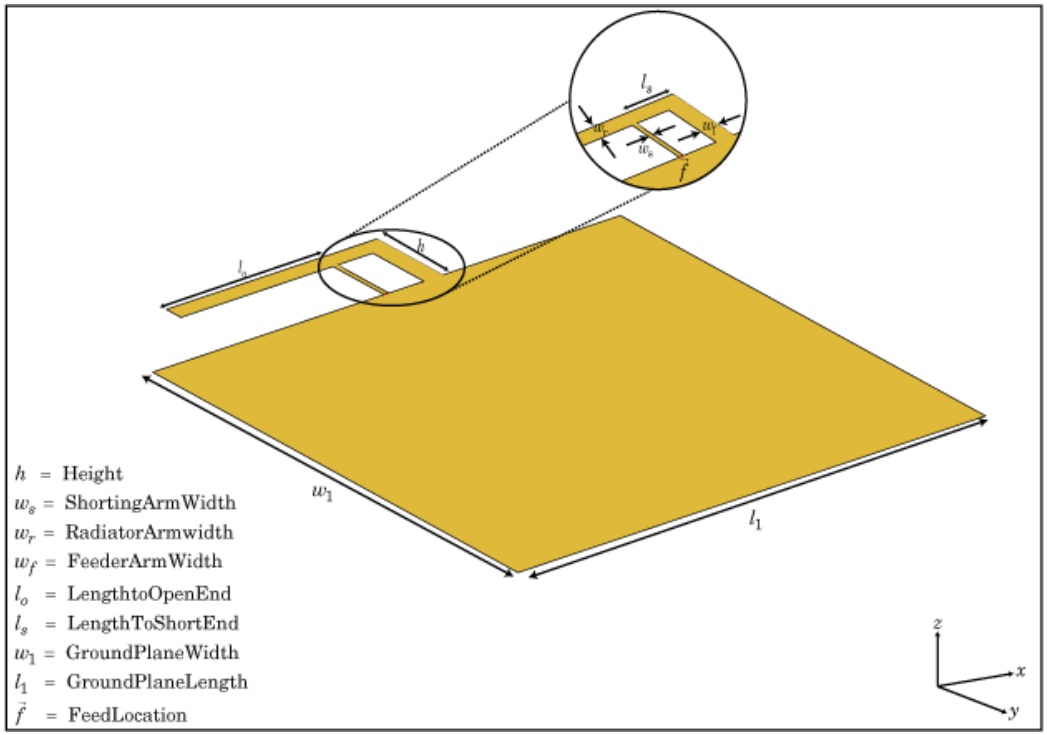
Introduced in R2015a

invertedFcoplanar

Create inverted-F antenna in same plane as rectangular ground plane

Description

The `invertedFcoplanar` object is a coplanar inverted-F antenna with a rectangular ground plane. By default, the dimensions are chosen for an operating frequency of 1.7 GHz. Coplanar inverted-F antennas are used in RFID tags and Internet of Things (IoT) applications. This antenna is an altered version of the inverted-F antenna, providing a low-profile antenna with more design parameters and a wider bandwidth.



Creation

Syntax

```
fco = invertedFcoplanar
fco = invertedF(Name,Value)
```

Description

`fco = invertedFcoplanar` creates a coplanar inverted-F antenna with the rectangular ground plane. By default, the antenna dimensions are for an operating frequency of 1.7 GHz.

`fco = invertedF(Name,Value)` creates a coplanar inverted-F antenna, with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`RadiatorArmWidth` — Width of radiating arm
`0.0040` (default) | scalar

Width of radiating arm, specified as the comma-separated pair consisting of `'RadiatorArmWidth'`and a scalar in meters.

Example: `'RadiatorArmWidth',0.05`

Data Types: `double`

`FeederArmWidth` — Width of feeding arm
`1.0000e-03` (default) | scalar

Width of feeding arm, specified as a scalar in meters.

Example: `'FeederArmWidth',0.05`

Data Types: `double`

`ShortingArmWidth` — Width of shorting arm
`0.0040` (default) | scalar

Width of shorting arm, specified as a scalar in meters.

Example: `'ShortingArmWidth',1`

Data Types: `double`

`Height` — Height of antenna
`0.0100` (default) | scalar

Height of antenna from ground plane, specified as a scalar in meters.

Example: `'Height',0.0800`

Data Types: `double`

`LengthToOpenEnd` — Length of stub from feed to open end
`0.0350` (default) | scalar

Length of the stub from feed to the open-end, specified as a scalar in meters.

Example: `'LengthToOpenEnd',0.050`

Data Types: `double`

`LengthToShortEnd` — Length of stub from feed to shorting end
`0.0100` (default) | scalar

Length of the stub from feed to the shorting end, specified as a scalar in meters.

Example: `'LengthToShortEnd',0.035`

Data Types: `double`

`GroundPlaneLength` — Length of ground plane
`0.0800` (default) | scalar

Length of the ground plane, specified as a scalar in meters.

Example: `'GroundPlaneLength',0.035`

Data Types: `double`

`GroundPlaneWidth` — Width of ground plane
`0.0700` (default) | scalar

Width of the ground plane, specified as a scalar in meters.

Example: `'GroundPlaneWidth',0.035`

Data Types: `double`

`FeedOffset` — Signed distance from center of ground plane
`0` (default) | scalar

Signed distance from center of groundplane, specified as a scalar in meters.

Example: `'FeedOffset',0.06`

Data Types: `double`

`Load` — Lumped elements
[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `fco.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.

- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Analysis Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
show	Display antenna or array structure; Display shape as filled patch
vswr	Voltage standing wave ratio of antenna

Examples

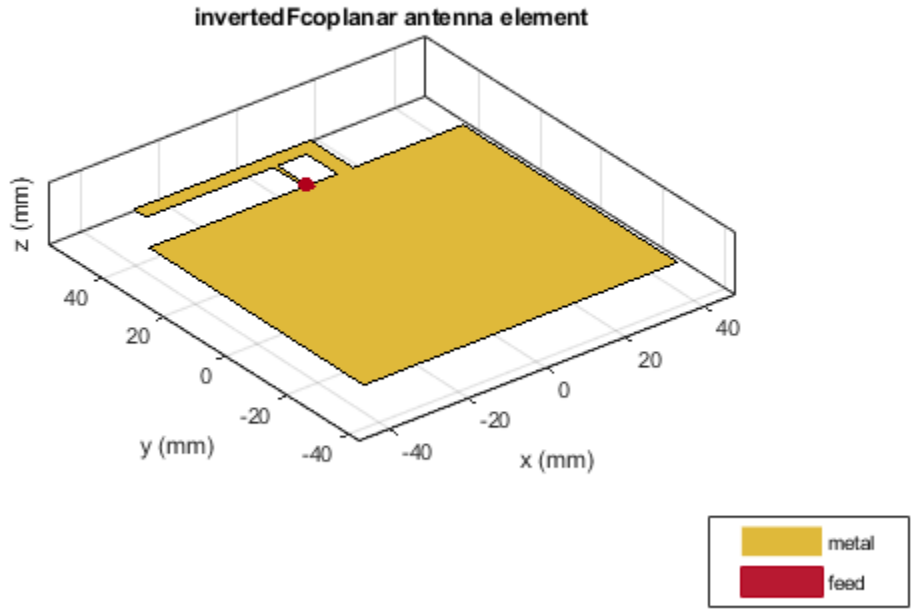
Coplanar Inverted-F Antenna

Create a default coplanar inverted-F antenna and view it.

```
fco = invertedFcoplanar
```

```
fco =
  invertedFcoplanar with properties:

     RadiatorArmWidth: 0.0040
       FeederArmWidth: 1.0000e-03
     ShortingArmWidth: 0.0040
      LengthToOpenEnd: 0.0350
     LengthToShortEnd: 0.0100
               Height: 0.0100
    GroundPlaneLength: 0.0800
     GroundPlaneWidth: 0.0700
           FeedOffset: 0
                 Tilt: 0
             TiltAxis: [1 0 0]
                 Load: [1x1 lumpedElement]
```

```
show(fco)
```



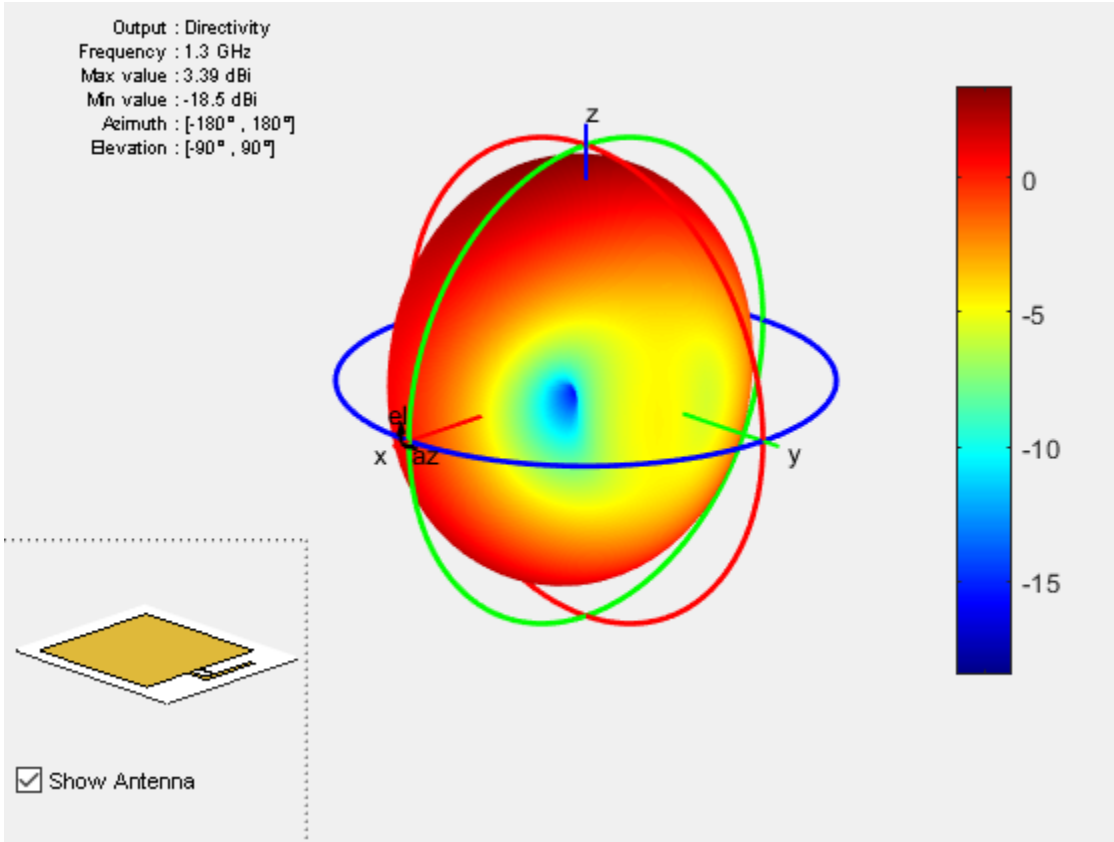
Radiation Pattern of Coplanar Inverted-F Antenna

Create a coplanar inverted-F antenna of height 0.014 m, ground plane length 0.1 m, and ground plane width 0.1 m.

```
fco = invertedFcoplanar('Height',14e-3,'GroundPlaneLength', 100e-3,  ...
                 'GroundPlaneWidth', 100e-3);
```

Plot the radiation pattern of the above antenna at 1.30 GHz.

```
pattern(fco,1.30e9)
```



References

[1] Balanis, C. A. *Antenna Theory. Analysis and Design*. 3rd Ed. Hoboken, NJ: John Wiley & Sons, 2005.

[2] Stutzman, W. L. and Gary A. Thiele. *Antenna Theory and Design*. 3rd Ed. River Street, NJ: John Wiley & Sons, 2013.

See Also

`invertedF` | `invertedL` | `invertedLcoplanar`

Topics

"Rotate Antenna and Arrays"

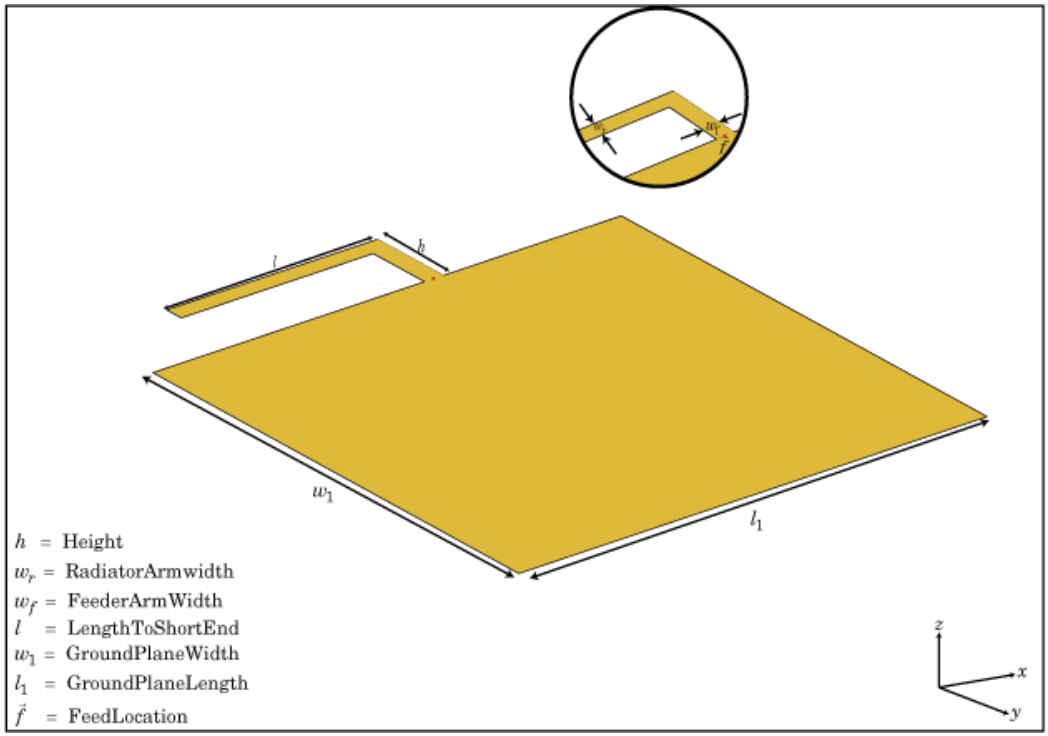
Introduced in R2016b

invertedLcoplanar

Create inverted-L antenna in same plane as rectangular ground plane

Description

The `invertedLcoplanar` object is a coplanar inverted-L antenna with the rectangular ground plane. By default, the dimensions are chosen for an operating frequency of 1.6 GHz. This antenna is used in applications that require low-profile narrow-bandwidth antennas, such as the transmitter for a garage door opener and Internet of Things (IoT) applications.



Creation

Syntax

```
lco = invertedLcoplanar
lco = invertedLcoplanar(Name,Value)
```

Description

`lco = invertedLcoplanar` creates a coplanar inverted-L antenna with the rectangular ground plane. By default, the antenna dimensions are for an operating frequency of 1.6 GHz.

`lco = invertedLcoplanar(Name,Value)` creates a coplanar inverted-L antenna, with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`RadiatorArmWidth` — Width of radiating arm
`0.0020` (default) | scalar

Width of radiating arm, specified as a scalar in meters.

Example: `'RadiatorArmWidth',0.05`

Data Types: `double`

`FeederArmWidth` — Width of feeding arm
`0.0020` (default) | scalar

Width of feeding arm, specified as scalar in meters.

Example: `'FeederArmWidth',0.05`

Data Types: `double`

`Height` — Height of antenna
`0.0100` (default) | scalar

Height of antenna from ground plane, specified as a scalar in meters.

Example: `'Height',0.0800`

Data Types: `double`

`Length` — Length of stub from feed to open end
`0.0350` (default) | scalar

Length of the stub from the feed to the open-end, specified as a scalar in meters.

Example: `'Length',0.0800`

Data Types: `double`

`GroundPlaneLength` — Length of ground plane
`0.0800` (default) | scalar in meters

Length of the ground plane, specified as a scalar in meters.

Example: `'GroundPlaneLength',0.035`

Data Types: `double`

`GroundPlaneWidth` — Width of ground plane
`0.0700` (default) | scalar

Width of the ground plane, specified as a scalar in meters.

Example: `'GroundPlaneWidth',0.035`

Data Types: `double`

`FeedOffset` — Signed distance from center of ground plane
`0` (default) | scalar

Signed distance from center of groundplane, specified a scalar in meters.

Example: `'FeedOffset',0.06`

Data Types: `double`

`Load` — Lumped elements
[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `lco.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
show	Display antenna or array structure; Display shape as filled patch
vswr	Voltage standing wave ratio of antenna

Examples

Coplanar Inverted-L Antenna

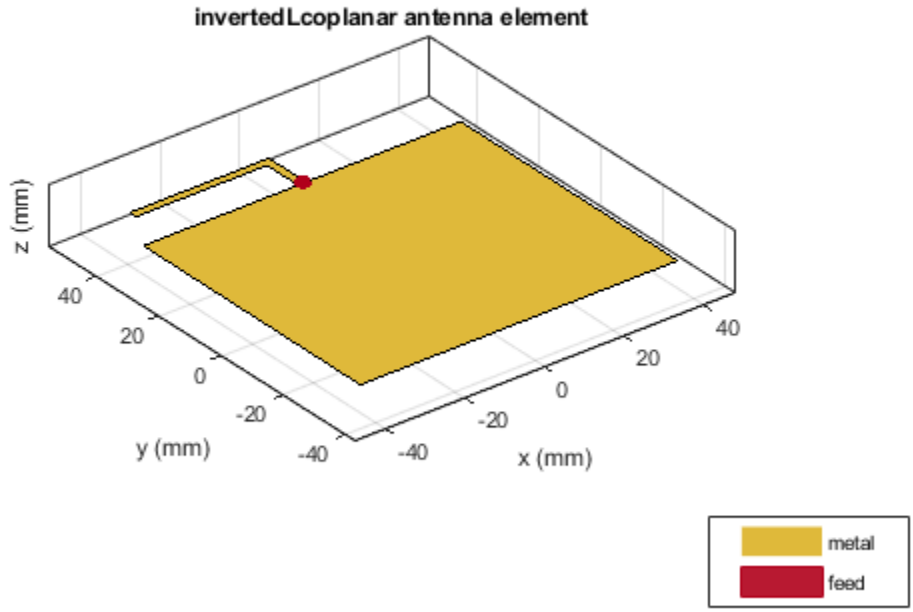
Create a default coplanar inverted-L antenna and view it.

```
lco =  invertedLcoplanar
```

```
lco =
  invertedLcoplanar with properties:

     RadiatorArmWidth: 0.0020
       FeederArmWidth: 0.0020
               Length: 0.0350
               Height: 0.0100
    GroundPlaneLength: 0.0800
```

```
    GroundPlaneWidth: 0.0700
          FeedOffset: 0
                Tilt: 0
            TiltAxis: [1 0 0]
                Load: [1x1 lumpedElement]
```

```
show(lco)
```



Impedance of Coplanar Inverted-L Antenna

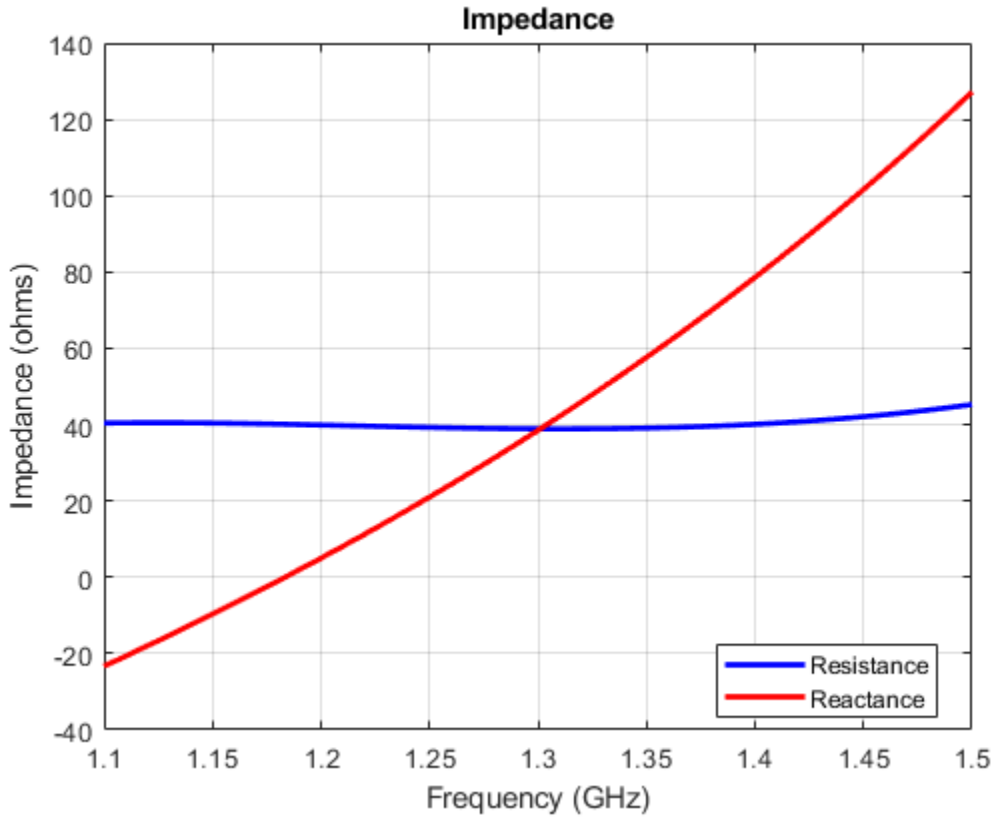
Create a coplanar inverted-L antenna of length 0.050 m, height 0.014m, ground plane length 0.1 m, and ground plane width 0.1 m.

```
lco = invertedLcoplanar('Length',50e-3, 'Height',14e-3,...
    'GroundPlaneLength',100e-3,'GroundPlaneWidth',100e-3)
```

```
lco =
  invertedLcoplanar with properties:

     RadiatorArmWidth: 0.0020
       FeederArmWidth: 0.0020
               Length: 0.0500
               Height: 0.0140
    GroundPlaneLength: 0.1000
     GroundPlaneWidth: 0.1000
           FeedOffset: 0
                 Tilt: 0
            TiltAxis: [1 0 0]
                 Load: [1x1 lumpedElement]
```

Plot the impedance over 1.1 GHz to 1.5 GHz in steps of 10 MHz.

```
impedance(lco,1.1e9:10e6:1.5e9);
```



References

[1] Balanis, C. A. *Antenna Theory. Analysis and Design*. 3rd Ed. Hoboken, NJ: John Wiley & Sons, 2005.

[2] Stutzman, W. L. and Gary A. Thiele. *Antenna Theory and Design*. 3rd Ed. River Street, NJ: John Wiley & Sons, 2013.

See Also

`invertedF` | `invertedFcoplanar` | `invertedL`

Topics

“Rotate Antenna and Arrays”

Introduced in R2016b

loopCircular

Create circular loop antenna

Description

The `loopCircular` object is a planar circular loop antenna on the X-Y plane.

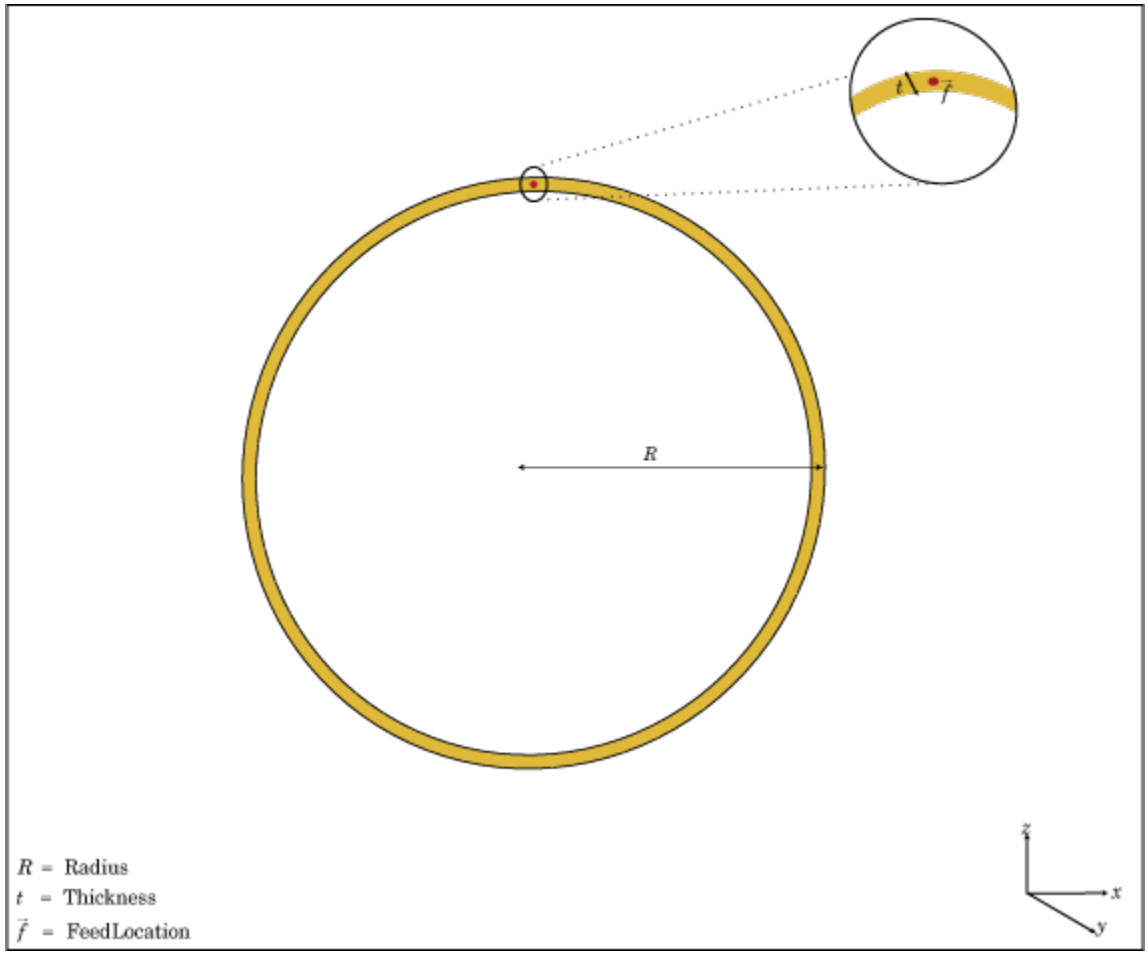
The thickness of the loop is related to the diameter of an equivalent cylinder loop by the equation

$$t = 2d = 4r$$

, where:

- d is the diameter of equivalent cylindrical loop
- r is the radius of equivalent cylindrical loop

For a given cylinder radius, use the `cylinder2strip` utility function to calculate the equivalent width. The default circular loop antenna is fed at the positive X-axis. The point of the X-axis is at the midpoint of the inner and outer radii.



Creation

Syntax

```
lc = loopCircular
lc = loopCircular(Name,Value)
```

Description

`lc = loopCircular` creates a one wavelength circular loop antenna in the X-Y plane. By default, the circumference is chosen for the operating frequency 75 MHz.

`lc = loopCircular(Name,Value)` creates a one wavelength circular loop antenna, with additional properties specified by one, or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Radius` — Outer radius of loop

0.6366 (default) | scalar

Outer radius of loop, specified as a scalar in meters.

Example: `'Radius',3`

Data Types: `double`

`Thickness` — Thickness of loop

0.0200 (default) | scalar

Thickness of loop, specified as a scalar in meters.

Example: `'Thickness',2`

Data Types: `double`

`Load` — Lumped elements

[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `lc.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface

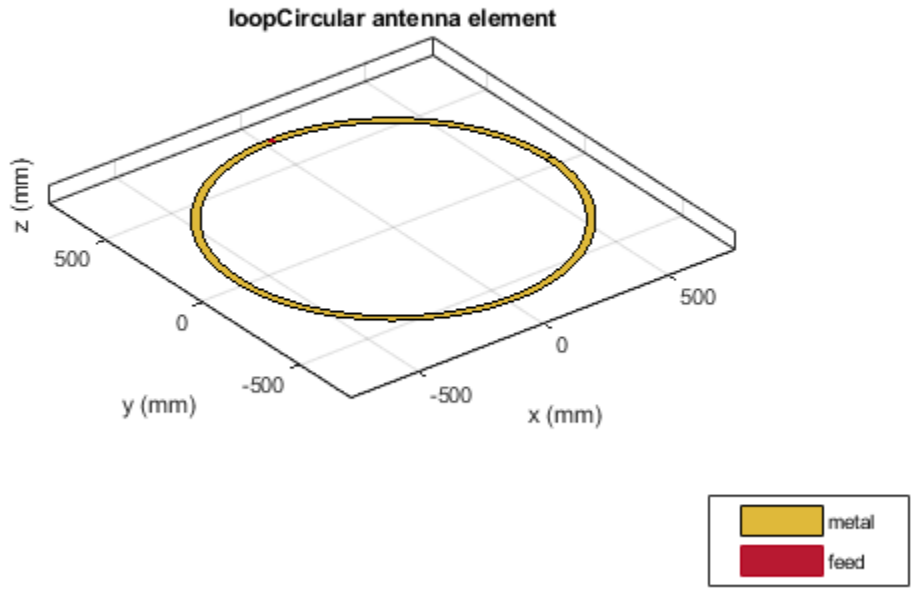
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

Create and View Circular Loop Antenna

Create and view a circular loop with 0.65 m radius and 0.01 m thickness.

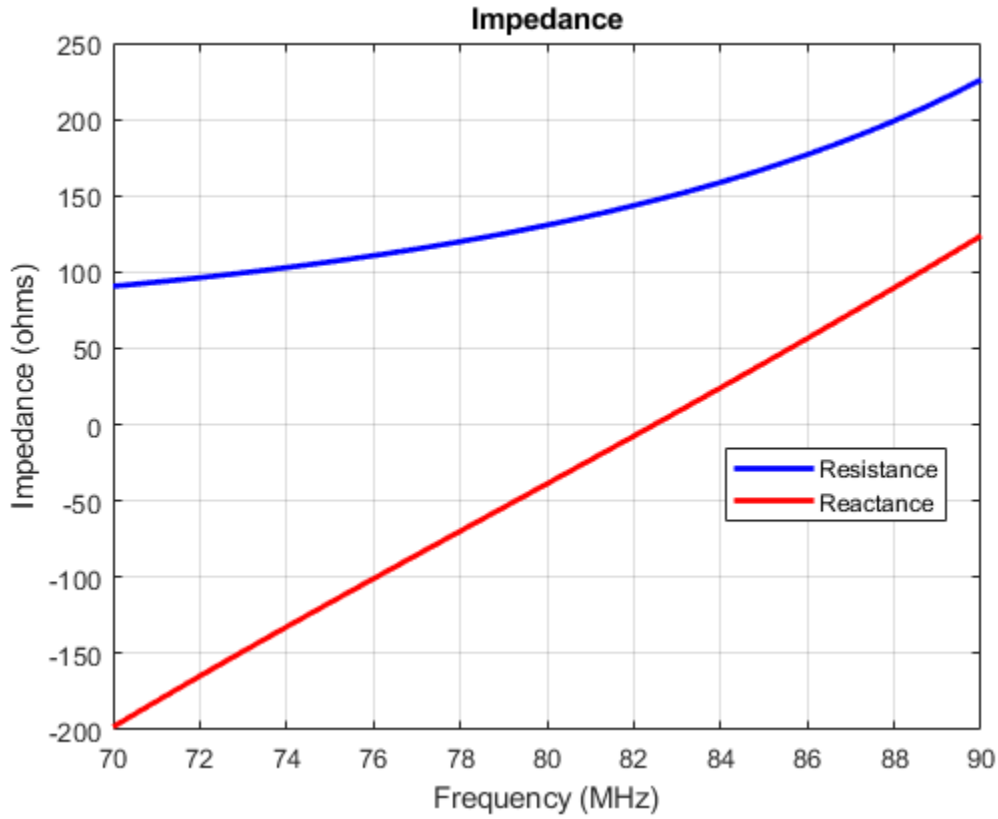
```
c = loopCircular('Radius',0.64,'Thickness',0.03);
show(c)
```



Impedance of Circular Loop Antenna

Calculate the impedance of a circular loop antenna over a frequency range of 70MHz-90MHz.

```
c = loopCircular('Radius',0.64,'Thickness',0.03);
impedance(c,linspace(70e6,90e6,31))
```



References

[1] Balanis, C.A. *Antenna Theory. Analysis and Design*, 3rd Ed. New York: Wiley, 2005.

See Also

`dipole` | `loopRectangular` | `slot`

Topics

“Rotate Antenna and Arrays”

Introduced in R2015a

loopRectangular

Create rectangular loop antenna

Description

The `loopRectangular` object is a rectangular loop antenna on the X-Y plane.

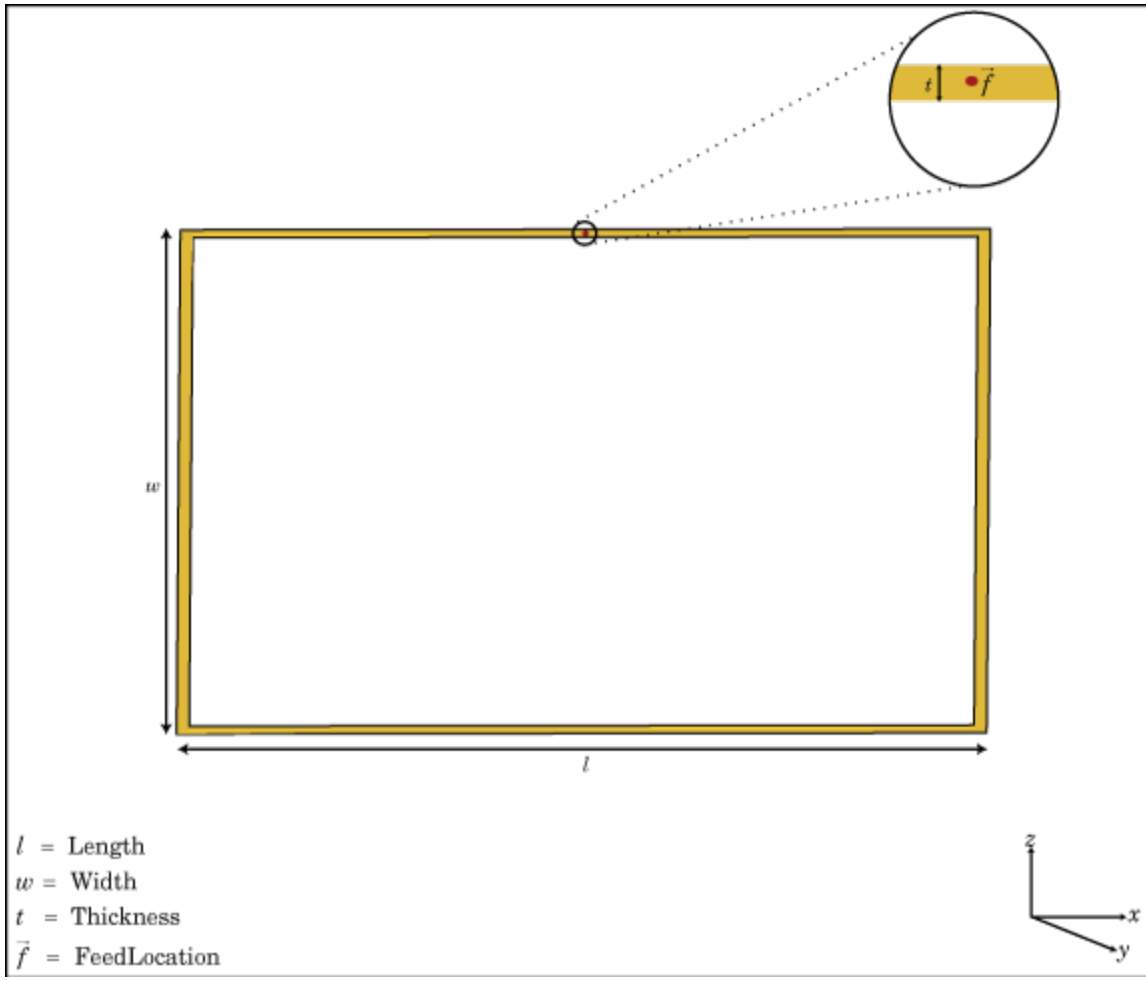
The thickness of the loop is related to the diameter of an equivalent cylinder loop by the equation

$$t = 2d = 4r$$

, where:

- d is the diameter of equivalent cylindrical loop
- r is the radius of equivalent cylindrical loop

For a given cylinder radius, use the `cylinder2strip` utility function to calculate the equivalent width. The default circular loop antenna is fed at the positive Y-axis. The point of the Y-axis is the midpoint of the inner and outer perimeter of the loop.



Creation

Syntax

```
lr = loopRectangular
lr = loopRectangular(Name,Value)
```

Description

`lr = loopRectangular` creates a rectangular loop antenna in the X-Y plane. By default, the dimensions are chosen for the operating frequency 53 MHz.

`lr = loopRectangular(Name,Value)` creates a rectangular loop antenna, with additional properties specified by one, or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retains their default values.

Properties

`Length` — Loop length along x-axis

2 (default) | scalar

Loop length along x-axis, specified as a scalar in meters.

Example: `'Length',3`

Data Types: `double`

`Width` — Loop width along y-axis

1 (default) | scalar

Loop width along y-axis, specified as a scalar in meters.

Example: `'Width',2`

Data Types: `double`

`Thickness` — Loop thickness

0.0100 (default) | scalar

Loop thickness, specified as a scalar in meters.

Example: `'Thickness',2`

Data Types: `double`

`Load` — Lumped elements

[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `lr.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

Create and View Rectangular Loop Antenna

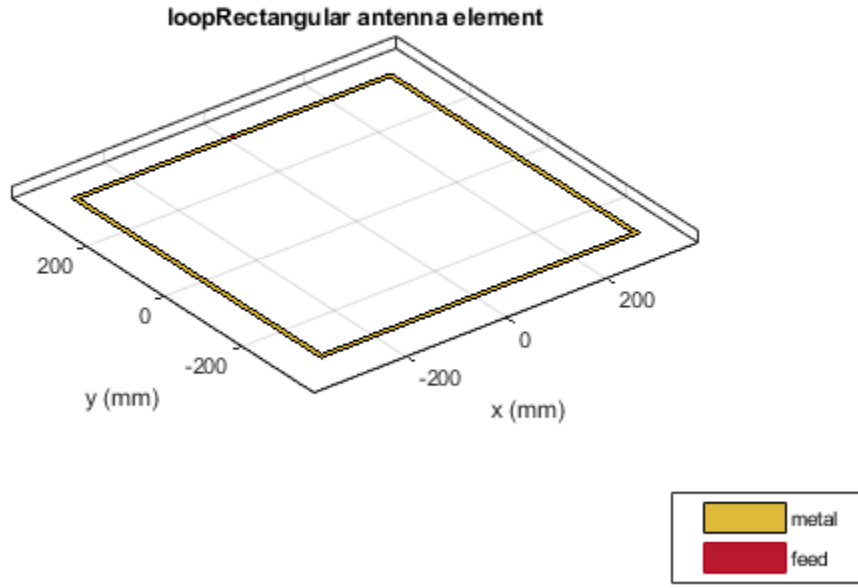
Create and view a rectangular loop antenna with 0.64m length, 0.64m width.

```
r = loopRectangular('Length',0.64,'Width',0.64)
```

```
r =
  loopRectangular with properties:

       Length: 0.6400
        Width: 0.6400
    Thickness: 0.0100
         Tilt: 0
     TiltAxis: [1 0 0]
```

```
          Load: [1x1 lumpedElement]
```

```
show(r)
```



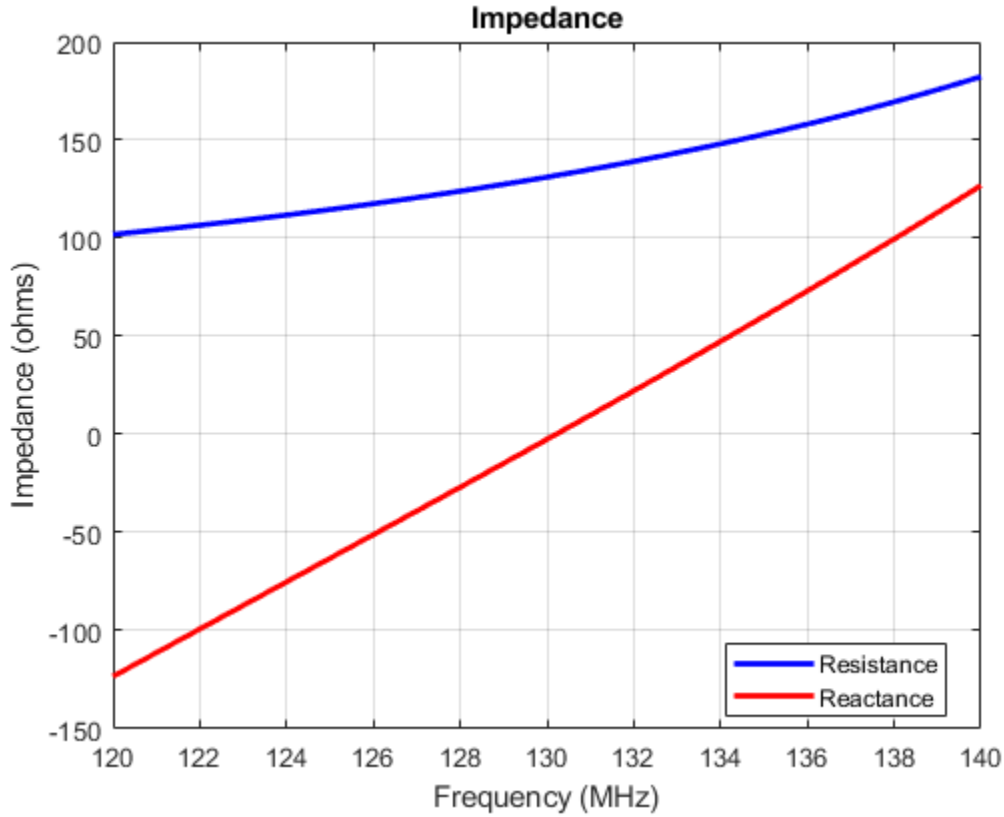
Impedance of Rectangular Loop Antenna

Calculate the impedance of a rectangular loop antenna over a frequency range of 120MHz-140MHz.

```
r = loopRectangular('Length',0.64,'Width',0.64)
```

```
r =
  loopRectangular with properties:

       Length: 0.6400
        Width: 0.6400
    Thickness: 0.0100
         Tilt: 0
     TiltAxis: [1 0 0]
         Load: [1x1 lumpedElement]
```

```
impedance(r,linspace(120e6,140e6,31))
```



References

[1] Balanis, C.A. *Antenna Theory. Analysis and Design*, 3rd Ed. New York: Wiley, 2005.

See Also

`dipole` | `loopCircular` | `monopole`

Topics

“Rotate Antenna and Arrays”

Introduced in R2015a

monopole

Create monopole antenna over rectangular ground plane

Description

The `monopole` object is a monopole antenna mounted over a rectangular ground plane.

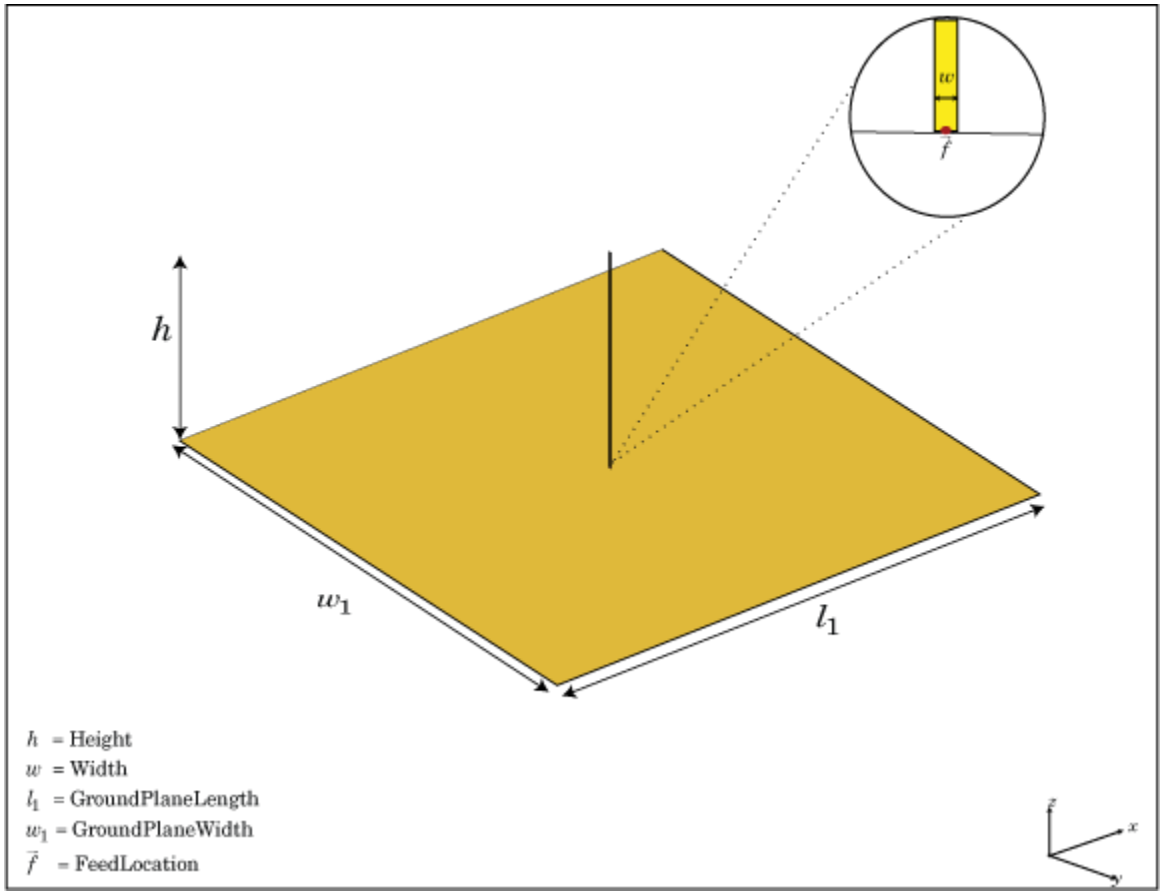
The width of the monopole is related to the diameter of an equivalent cylindrical monopole by the equation

$$w = 2d = 4r$$

, where:

- d is the diameter of equivalent cylindrical monopole
- r is the radius of equivalent cylindrical monopole.

For a given cylinder radius, use the `cylinder2strip` utility function to calculate the equivalent width. The default monopole is center-fed. The feed point coincides with the origin. The origin is located on the X-Y plane.



Creation

Syntax

```
mpl = monopole
mpl = monopole(Name,Value)
```

Description

`mpl = monopole` creates a quarter-wavelength monopole antenna.

`mpl = monopole(Name,Value)` creates a quarter-wavelength monopole antenna with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Height` — Height of vertical element along Z-axis

1 (default) | scalar

Height of vertical element along z-axis, specified as a scalar in meters. By default, the height is chosen for an operating frequency of 75 MHz.

Example: `'Height',3`

Data Types: `double`

`Width` — Monopole width

0.1000 (default) | scalar

Monopole width, specified as a scalar in meters.

Note Monopole width should be less than `'Height'`/4 and greater than `'Height'`/1001. [2]

Example: `'Width',0.05`

Data Types: `double`

`GroundPlaneLength` — Ground plane length along X-axis

2 (default) | scalar

Ground plane length along x-axis, specified as a scalar in meters. Setting `'GroundPlaneLength'` to `Inf`, uses the infinite ground plane technique for antenna analysis.

Example: `'GroundPlaneLength',4`

Data Types: `double`

`GroundPlaneWidth` — Ground plane width along Y-axis
2 (default) | scalar

Ground plane width along y-axis, specified as a scalar in meters. Setting `'GroundPlaneWidth'` to `Inf`, uses the infinite ground plane technique for antenna analysis.

Example: `'GroundPlaneWidth',2.5`

Data Types: `double`

`FeedOffset` — Signed distance from center along length and width of ground plane
`[0 0]` (default) | two-element vector

Signed distance from center along length and width of ground plane, specified as a two-element vector.

Example: `'FeedOffset',[2 1]`

Data Types: `double`

`Load` — Lumped elements
[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `mpl.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array

patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

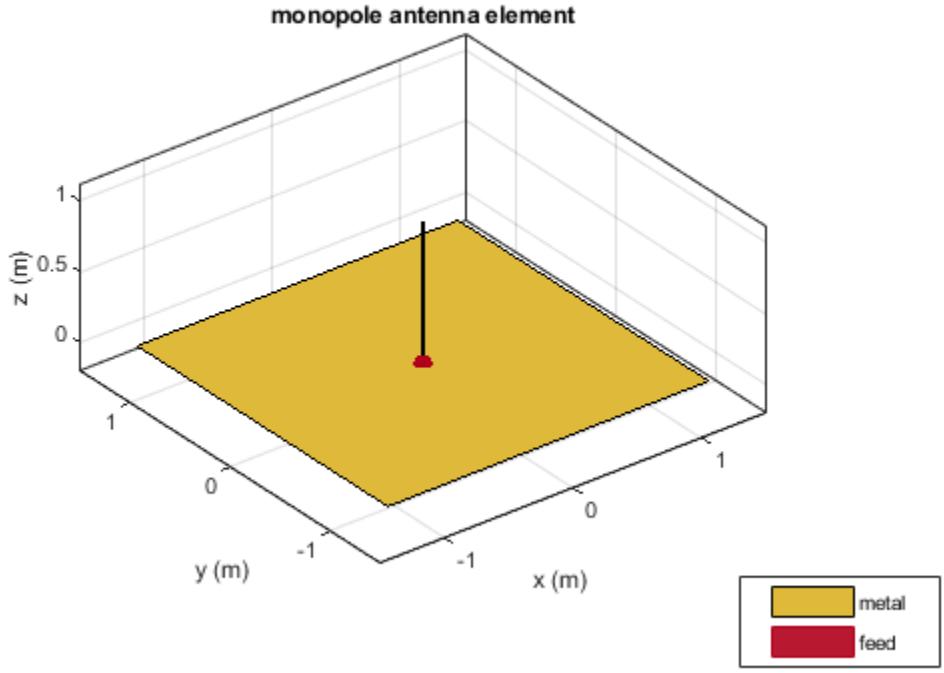
Create and View Monopole Antenna

Create and view a monopole of 1 m length, 0.01 m width and ground plane of dimensions 2.5mx2.5m.

```
m = monopole('GroundPlaneLength',2.5,'GroundPlaneWidth',2.5)
```

```
m =
  monopole with properties:

              Height: 1
               Width: 0.0100
    GroundPlaneLength: 2.5000
     GroundPlaneWidth: 2.5000
           FeedOffset: [0 0]
                 Tilt: 0
            TiltAxis: [1 0 0]
                Load: [1x1 lumpedElement]
```

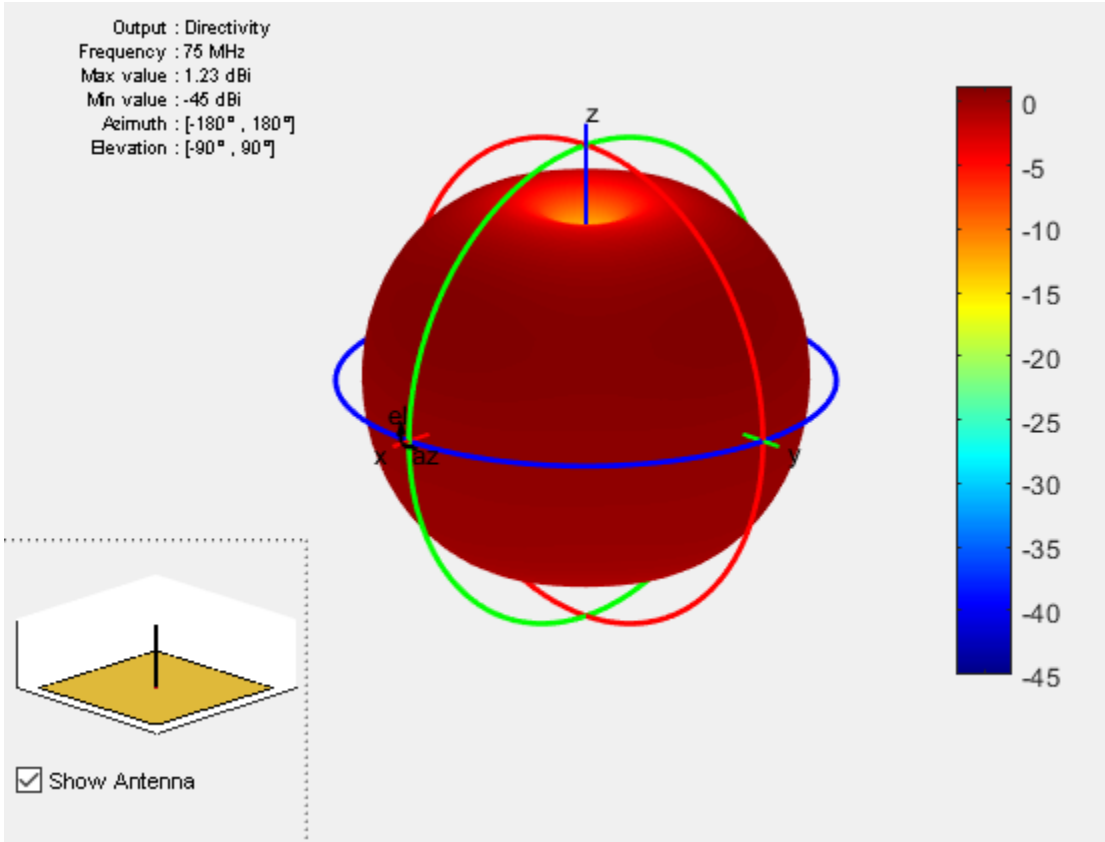
```
show(m)
```



Radiation Pattern of Monopole Antenna

Radiation pattern of a monopole at a frequency of 75MHz.

```
m = monopole('GroundPlaneLength',2.5, 'GroundPlaneWidth',2.5);
pattern (m,75e6)
```



References

[1] Balanis, C.A. *Antenna Theory. Analysis and Design*, 3rd Ed. New York: Wiley, 2005.

[2] Volakis, John. *Antenna Engineering Handbook*, 4th Ed. New York: Mcgraw-Hill, 2007.

See Also

`dipole` | `monopoleTopHat` | `patchMicrostrip`

Topics

“Rotate Antenna and Arrays”

Introduced in R2015a

monopoleTopHat

Create capacitively loaded monopole antenna over rectangular ground plane

Description

The `monopoleTopHat` object is a top-hat monopole antenna mounted over a rectangular ground plane. The monopole always connects with the center of top hat. The top hat builds up additional capacitance to ground within the structure. This capacitance reduces the resonant frequency of the antenna without increasing the size of the element.

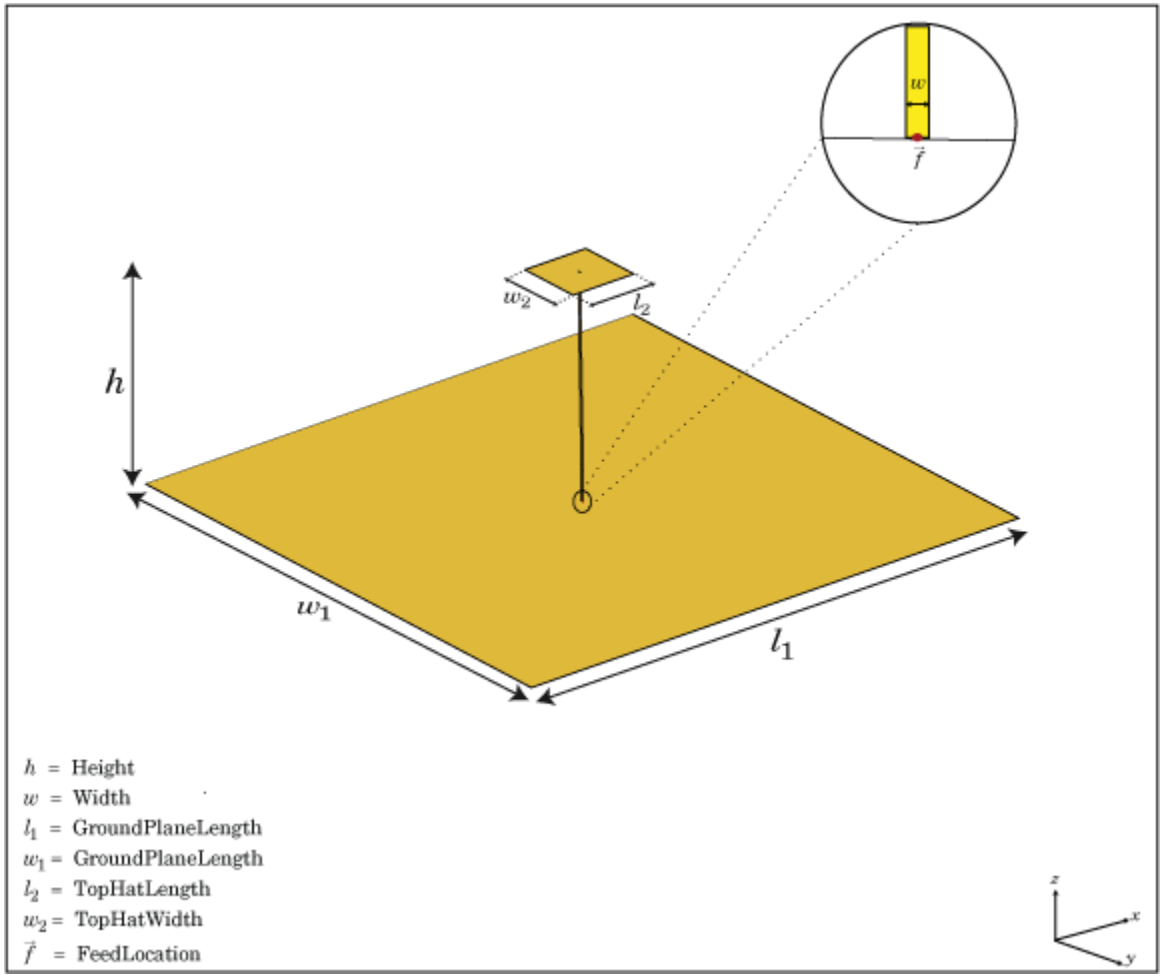
The width of the monopole is related to the diameter of an equivalent cylindrical monopole by the expression

$$w = 2d = 4r$$

,where:

- d is the diameter of equivalent cylindrical monopole
- r is the radius of equivalent cylindrical monopole.

For a given cylinder radius, use the `cylinder2strip` utility function to calculate the equivalent width. The default top-hat monopole is center-fed. The feed point coincides with the origin. The origin is located on the X-Y plane.



Creation

Syntax

```
mth = monopoleTopHat
mth = monopoleTopHat(Name,Value)
```

Description

`mth = monopoleTopHat` creates a capacitively loaded monopole antenna over a rectangular ground plane.

`mth = monopoleTopHat(Name,Value)` creates a capacitively loaded monopole antenna with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`,`...`, `NameN`, `ValueN`. Properties not specified retains their default values.

Properties

`Height` — Monopole height

1 (default) | scalar

Monopole height, specified as a scalar in meters. By default, the height is chosen for an operating frequency of 75 MHz.

Example: `'Height',3`

Data Types: `double`

`Width` — Monopole width

0.1000 (default) | scalar

Monopole width, specified as a scalar in meters.

Note Monopole width should be less than `'Height'`/4 and greater than `'Height'`/1001. [2]

Example: `'Width',0.05`

Data Types: `double`

`GroundPlaneLength` — Ground plane length along X-axis

2 (default) | scalar

Ground plane length along x-axis, specified as a scalar in meters. Setting `'GroundPlaneLength'` to `Inf`, uses the infinite ground plane technique for antenna analysis.

Example: `'GroundPlaneLength',4`

Data Types: `double`

`GroundPlaneWidth` — Ground plane width along Y-axis
2 (default) | scalar

Ground plane width along y-axis, specified as a scalar in meters. Setting `'GroundPlaneWidth'` to `Inf`, uses the infinite ground plane technique for antenna analysis.

Example: `'GroundPlaneWidth',2.5`

Data Types: `double`

`TopHatLength` — Top hat length along X-axis
0.2500 (default) | scalar

Top hat length along x-axis, specified as a scalar in meters.

Example: `'TopHatLength',4`

Data Types: `double`

`TopHatWidth` — Top hat width along Y-axis
0.2500 (default) | scalar

Top hat width along y-axis, specified as a scalar in meters.

Example: `'TopHatWidth',4`

Data Types: `double`

`FeedOffset` — Signed distance from center along length and width of ground plane
`[0 0]` (default) | two-element vector

Signed distance from center along length and width of ground plane, specified as a two-element vector.

Example: `'FeedOffset',[2 1]`

Data Types: `double`

Load — Lumped elements

[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `mth.Load = lumpedElement('Impedance',75)`

Tilt — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

TiltAxis — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

Create and View Top Hat Monopole.

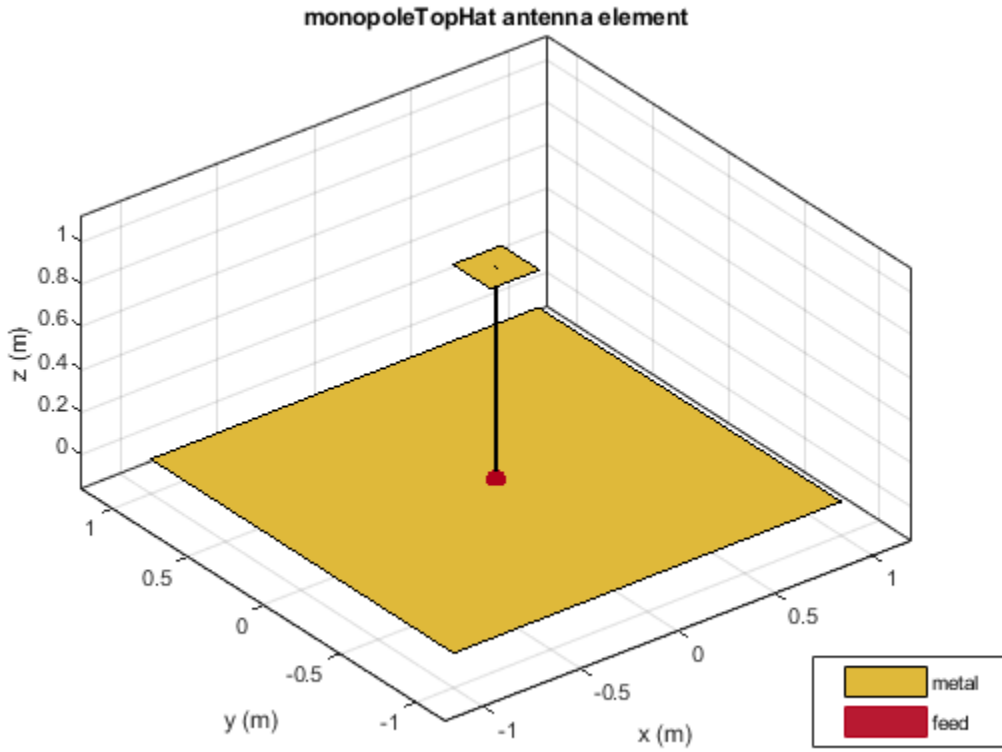
Create and view a top hat monopole with 1 m length, 0.01 m width, groundplane dimensions 2mx2m and top hat dimensions 0.25mx0.25m.

```
th = monopoleTopHat
```

```
th =
  monopoleTopHat with properties:

              Height: 1
               Width: 0.0100
    GroundPlaneLength: 2
     GroundPlaneWidth: 2
         TopHatLength: 0.2500
```

```
TopHatWidth: 0.2500
 FeedOffset: [0 0]
       Tilt: 0
   TiltAxis: [1 0 0]
       Load: [1x1 lumpedElement]
```

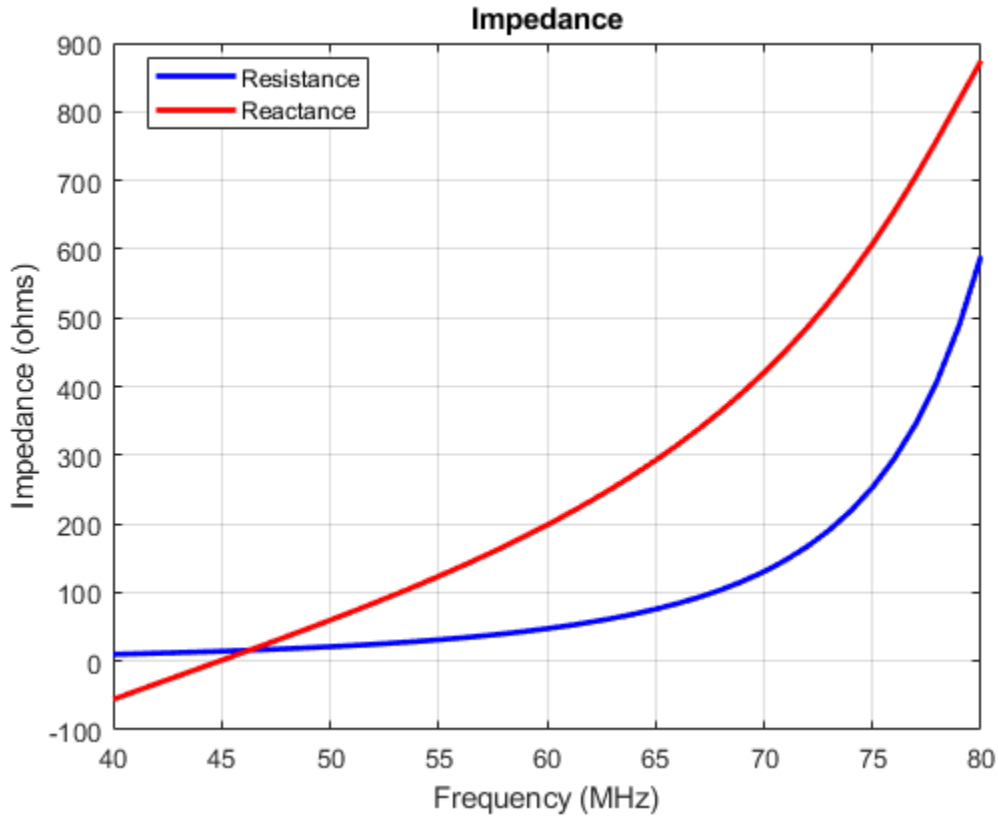
```
show(th)
```



Calculate Impedance of Top Hat Monopole Antenna

Calculate and plot the impedance of a top hat monopole over a frequency range of 40MHz-80MHz.

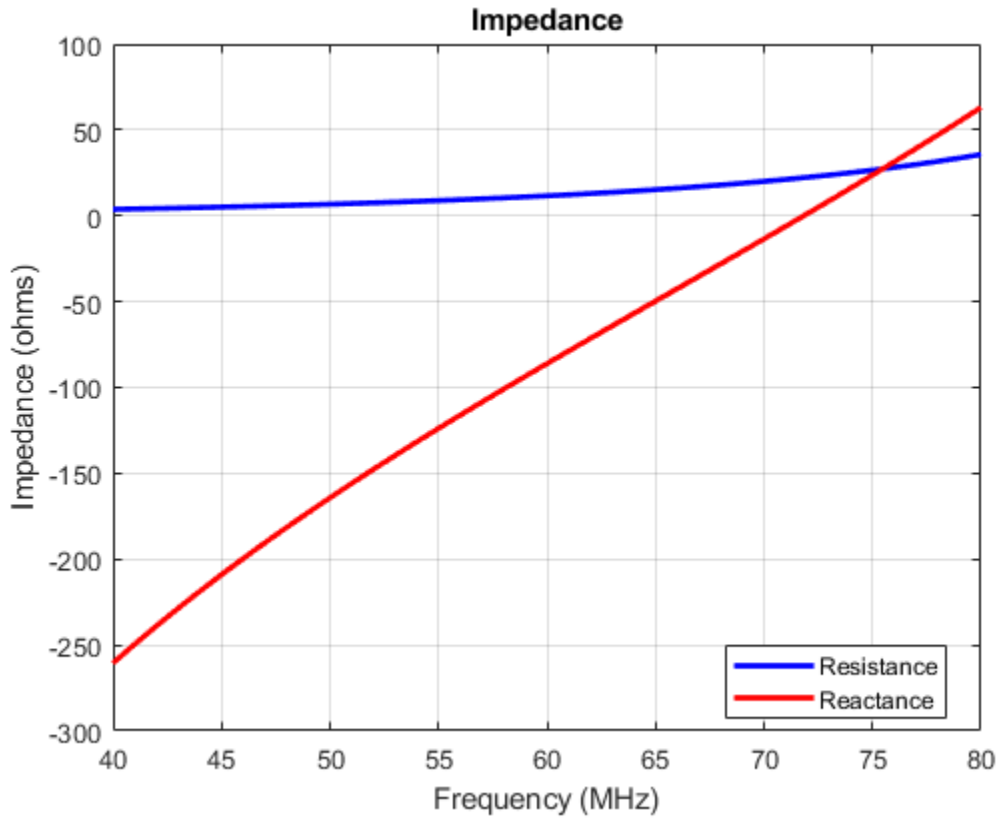
```
th = monopoleTopHat;
impedance(th,linspace(40e6,80e6,41));
```



Compare Impedance of Top Hat Monopole Antenna and Monopole Antenna

Impedance comparison between a monopole of similar dimensions and the top hat monopole in example 2.

```
m = monopole;
figure
impedance(m,linspace(40e6,80e6,41));
```



References

[1] Balanis, C.A. *Antenna Theory. Analysis and Design*, 3rd Ed. New York: Wiley, 2005.

[2] Volakis, John. *Antenna Engineering Handbook*, 4th Ed. New York: Mcgraw-Hill, 2007.

See Also

`dipole` | `monopole` | `patchMicrostrip`

Topics

“Rotate Antenna and Arrays”

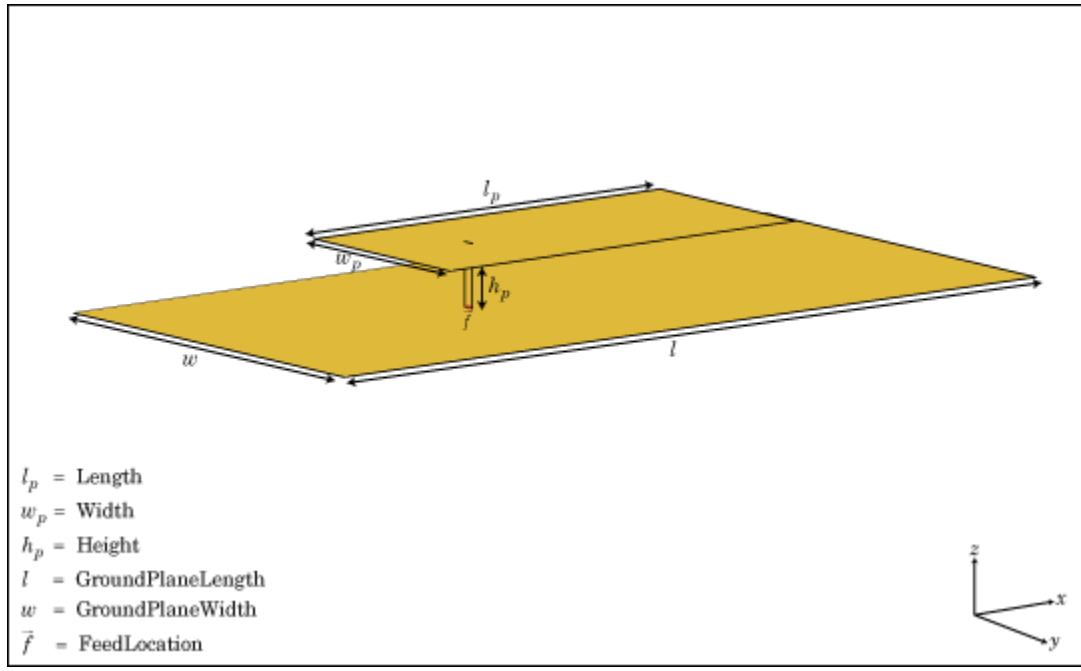
Introduced in R2015a

patchMicrostrip

Create microstrip patch antenna

Description

The `patchMicrostrip` object is a microstrip patch antenna. The default patch is centered at the origin. The feed point is along the length of the antenna.



Creation

Syntax

```
pm = patchMicrostrip
pm = patchMicrostrip(Name,Value)
```

Description

`pm = patchMicrostrip` creates a microstrip patch antenna.

`pm = patchMicrostrip(Name,Value)` creates a microstrip patch antenna, with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Length` — Patch length along X-axis

0.0750 (default) | scalar

Patch length, specified as a scalar in meters. By default, the length is measured along the x-axis.

Example: `'Length',50e-3`

Data Types: `double`

`Width` — Patch width along the Y-axis

0.0375 (default) | scalar

Patch width, specified as a scalar in meters. By default, the width is measured along the y-axis.

Example: `'Width',60e-3`

Data Types: `double`

`Height` — Height of substrate
0.0060 (default) | scalar

Height of substrate, specified as a scalar in meters.

Example: `'Height',37e-3`

Data Types: `double`

`Substrate` — Type of dielectric material
'Air' (default) | object

Type of dielectric material used as a substrate, specified as an object. For more information see, `dielectric`. For more information on dielectric substrate meshing, see “Meshing”.

Note The substrate dimensions must be equal to the groundplane dimensions.

Example: `d = dielectric('FR4'); 'Substrate',d`

Example: `d = dielectric('FR4'); pm.Substrate = d`

`GroundPlaneLength` — Ground plane length along x-axis
0.1500 (default) | scalar

Ground plane length, specified as a scalar in meters. By default, ground plane length is measured along x-axis. Setting `'GroundPlaneLength'` to `Inf`, uses the infinite ground plane technique for antenna analysis.

Example: `'GroundPlaneLength',120e-3`

Data Types: `double`

`GroundPlaneWidth` — Ground plane width along y-axis
0.0750 (default) | scalar

Ground plane width, specified as a scalar in meters. By default, ground plane width is measured along y-axis. Setting `'GroundPlaneWidth'` to `Inf`, uses the infinite ground plane technique for antenna analysis.

Example: `'GroundPlaneWidth',120e-3`

Data Types: `double`

`PatchCenterOffset` — Signed distance from center along length and width of ground plane

`[0 0]` (default) | two-element vector

Signed distance from center along length and width of ground plane, specified as a two-element vector in meters. Use this property to adjust the location of the patch relative to the ground plane.

Example: `'PatchCenterOffset',[0.01 0.01]`

Data Types: `double`

`FeedOffset` — Signed distance from center along length and width of ground plane

[–0.0187 0] (default) | two-element vector

Signed distance from center along length and width of ground plane, specified as a two-element vector. Use this property to adjust the location of the feedpoint relative to ground plane and patch.

Example: `'FeedOffset',[0.01 0.01]`

Data Types: `double`

`Load` — Lumped elements

[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `pm.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array

patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

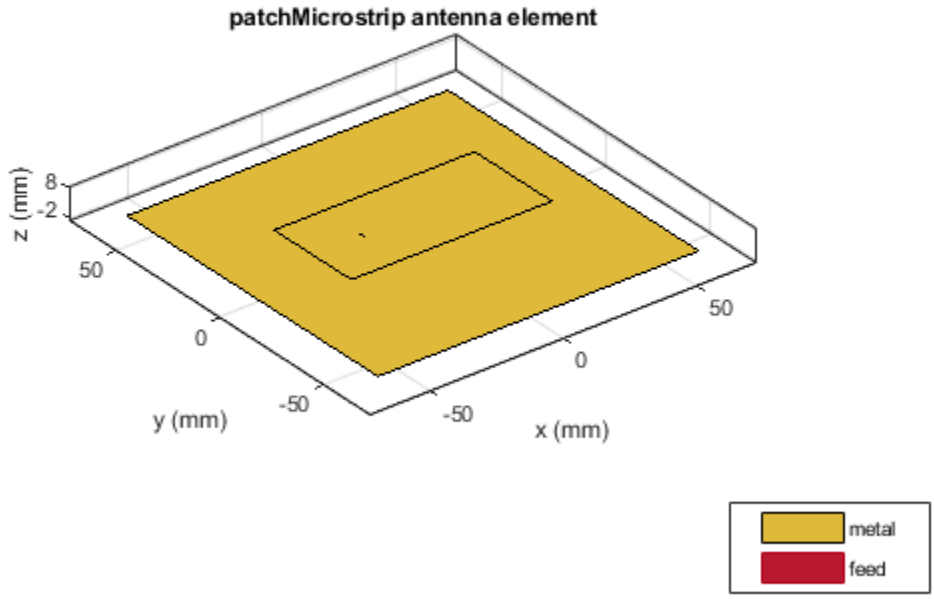
Create and View Microstrip Patch Antenna

Create and view a microstrip patch that has 75 mm length and 37.5 mm width over a 120 mm x 120 mm ground plane.

```
pm = patchMicrostrip('Length',75e-3, 'Width',37e-3,                  ...
        'GroundPlaneLength',120e-3, 'GroundPlaneWidth',120e-3)
```

```
pm =
  patchMicrostrip with properties:

              Length: 0.0750
               Width: 0.0370
              Height: 0.0060
           Substrate: [1x1 dielectric]
    GroundPlaneLength: 0.1200
     GroundPlaneWidth: 0.1200
    PatchCenterOffset: [0 0]
          FeedOffset: [-0.0187 0]
                Tilt: 0
            TiltAxis: [1 0 0]
                Load: [1x1 lumpedElement]
```

```
show (pm)
```



Radiation Pattern of Microstrip Patch Antenna

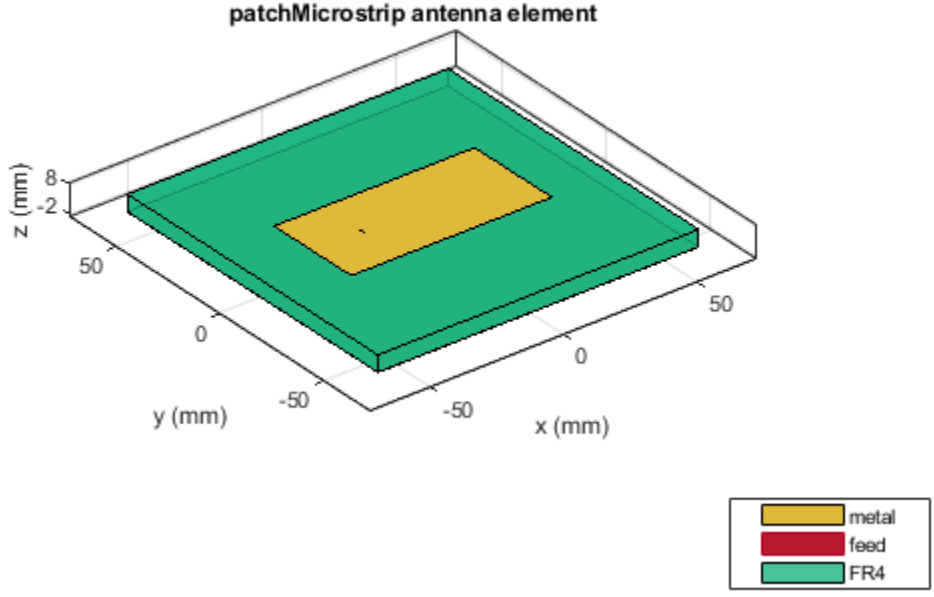
Create a microstrip patch antenna using **'FR4'** as the dielectric substrate.

```
d = dielectric('FR4');
pm = patchMicrostrip('Length',75e-3,'Width',37e-3,     ...
        'GroundPlaneLength',120e-3,'GroundPlaneWidth',120e-3, ...
        'Substrate',d)
```

```
pm =
  patchMicrostrip with properties:
```

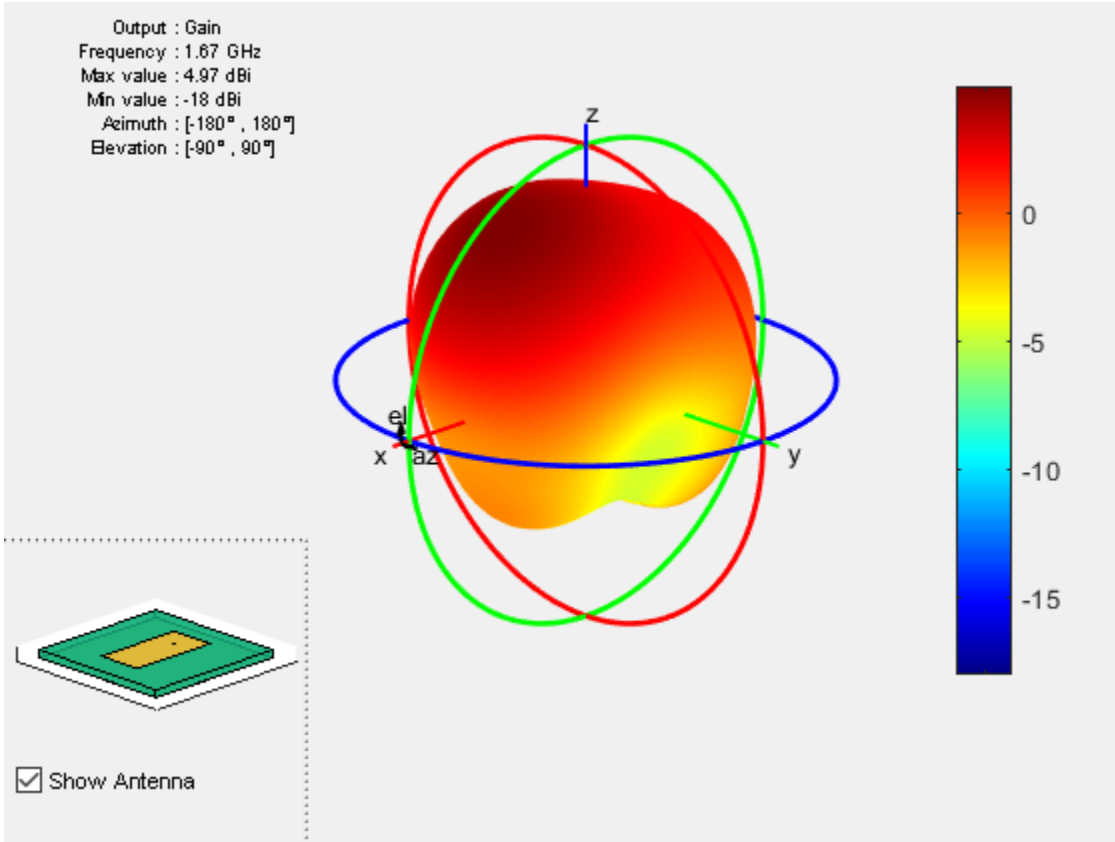
```
            Length: 0.0750
             Width: 0.0370
            Height: 0.0060
         Substrate: [1x1 dielectric]
GroundPlaneLength: 0.1200
 GroundPlaneWidth: 0.1200
PatchCenterOffset: [0 0]
       FeedOffset: [-0.0187 0]
             Tilt: 0
          TiltAxis: [1 0 0]
             Load: [1x1 lumpedElement]
```

```
show(pm)
```



Plot the radiation pattern of the antenna at a frequency of 1.67 GHz.

```
figure
pattern(pm,1.67e9)
```

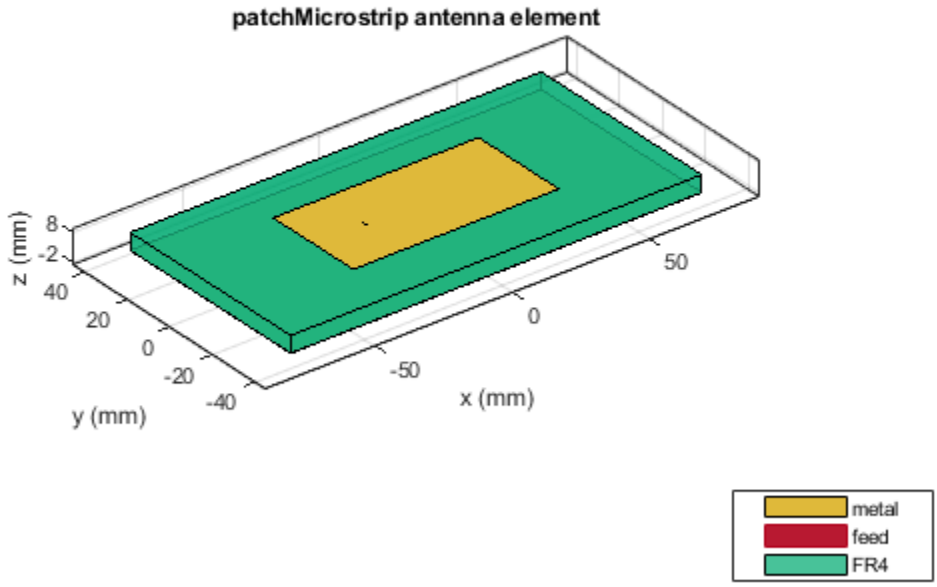


Impedance of Microstrip Patch Antenna

Create a microstrip patch antenna using **'FR4'** as the dielectric substrate.

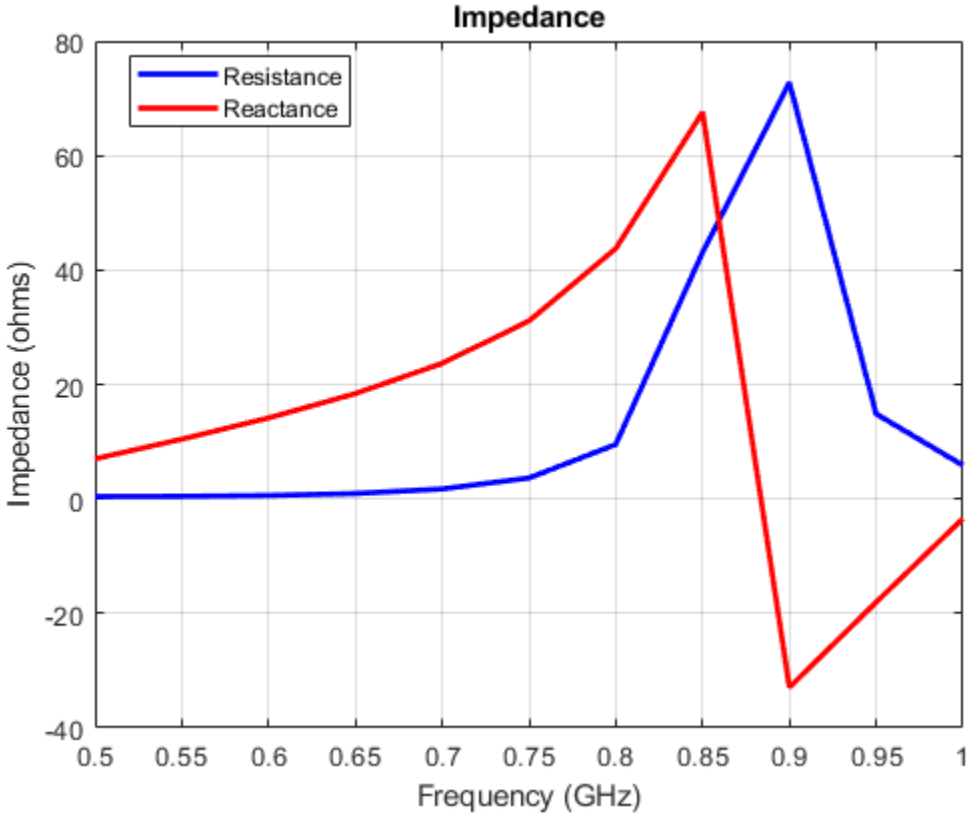
```
d = dielectric('FR4');
pm = patchMicrostrip('Substrate',d)
show(pm)
```

```
pm =

  patchMicrostrip with properties:

              Length: 0.0750
               Width: 0.0375
              Height: 0.0060
           Substrate: [1x1 dielectric]
    GroundPlaneLength: 0.1500
     GroundPlaneWidth: 0.0750
    PatchCenterOffset: [0 0]
          FeedOffset: [-0.0187 0]
                Tilt: 0
            TiltAxis: [1 0 0]
                Load: [1x1 lumpedElement]
```



Calculate and plot the impedance of the antenna over a frequency range of 1.5-2 GHz.

```
impedance(pm,linspace(0.5e9,1e9,11));
```



References

[1] Balanis, C.A. *Antenna Theory. Analysis and Design*, 3rd Ed. New York: Wiley, 2005.

See Also

`pifa` | `vivaldi` | `yagiUda`

Topics

“Rotate Antenna and Arrays”

Introduced in R2015a

planeWaveExcitation

Create plane wave excitation environment for antenna or array

Description

The `planeWaveExcitation` object creates an environment where a plane wave excites an antenna or array. Plane wave excitation is a scattering solution that solves the receiving antenna problem. By default, the antenna element is a dipole. The dipole is excited using a plane wave that travels along the positive x-axis having a z-polarization.

Creation

Syntax

```
h = planeWaveExcitation
h = planeWaveExcitation(Name,Value)
```

Description

`h = planeWaveExcitation` creates an environment where a plane wave excites the antenna or array. By default, the plane wave excites a dipole antenna.

`h = planeWaveExcitation(Name,Value)` returns a `planeWaveExcitation` environment, with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Element` — Antenna or array element
`dipole` (default) | object handle

Antenna or array element, specified as an object handle.

Example: `'Element',linearArray`

`Direction` — Incidence of plane wave
`[1 0 0]` (default) | three-element real vector

Incidence of plane wave, specified as a three-element real vector.

Example: `'Direction',[0 0 1]`

Data Types: `double`

`Polarization` — Polarization of incident electric field
`[0 0 1]` (default) | three-element complex vector

Polarization of incident electric field in *x*, *y*, and *z* components, specified as a three-element complex vector.

Example: `'Polarization',[0 1 0]`

Data Types: `double`

Object Functions

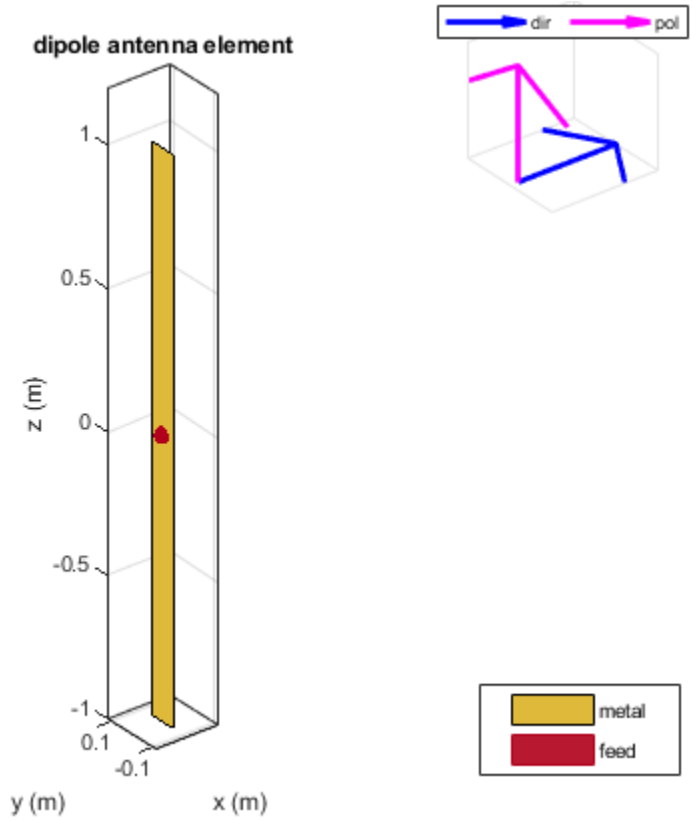
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
show	Display antenna or array structure; Display shape as filled patch

Examples

Default Plane Wave Excitation

Excite a dipole antenna using a plane wave and view it.

```
h = planeWaveExcitation;
show(h)
```



The blue arrow shows the direction of propagation of the plane wave. By default, the direction is along the x-axis. The pink arrow shows polarization of the plane wave. By default, the polarization is perpendicular to the direction of propagation i.e. along the z-axis.

Feed Current of Antenna Excited By Plane Wave.

Excite a dipole antenna using plane wave. Calculate the feed current at 70 MHz.

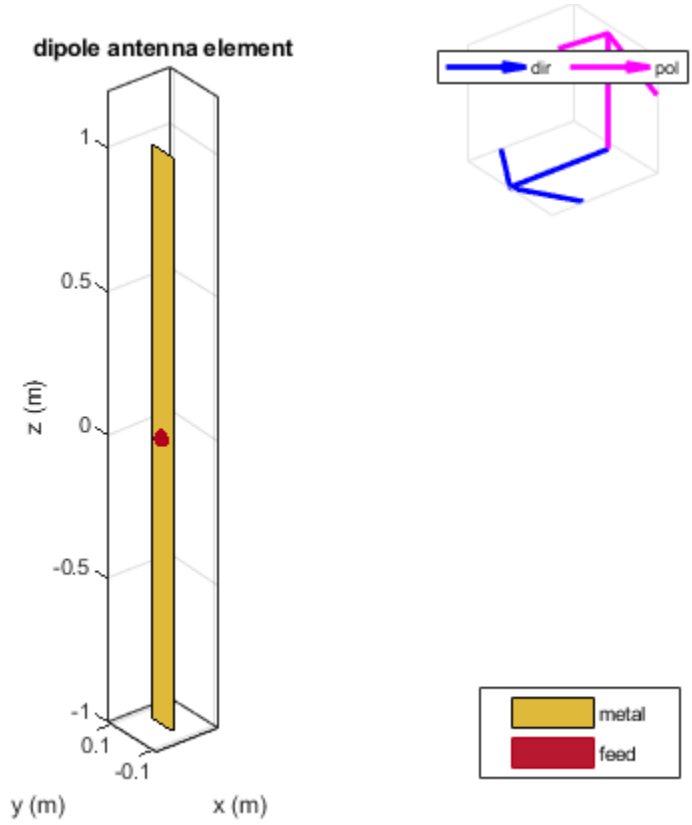
```
h = planeWaveExcitation
cur = feedCurrent(h, 70e6)
```

```
h =

  planeWaveExcitation with properties:

         Element: [1×1 dipole]
       Direction: [1 0 0]
    Polarization: [0 0 1]


cur =

   0.0179 - 0.0040i
```

Current Distribution On Antenna

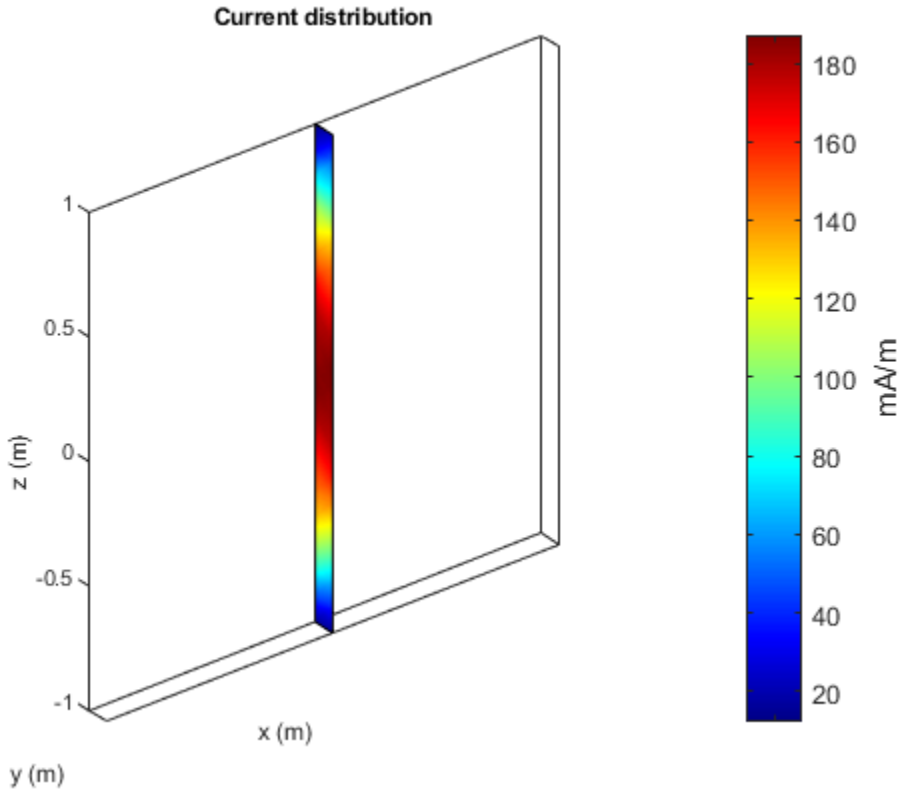
Excite a dipole antenna using a plane wave. The polarization of the wave is along the z-axis and the direction of propagation is along the negative x-axis. View the antenna.

```
p = planeWaveExcitation('Element', dipole, 'Direction', [-1 0 0], 'Polarization', [0 0
show(p);
```



Plot the current distribution on the dipole antenna at 70 MHz.

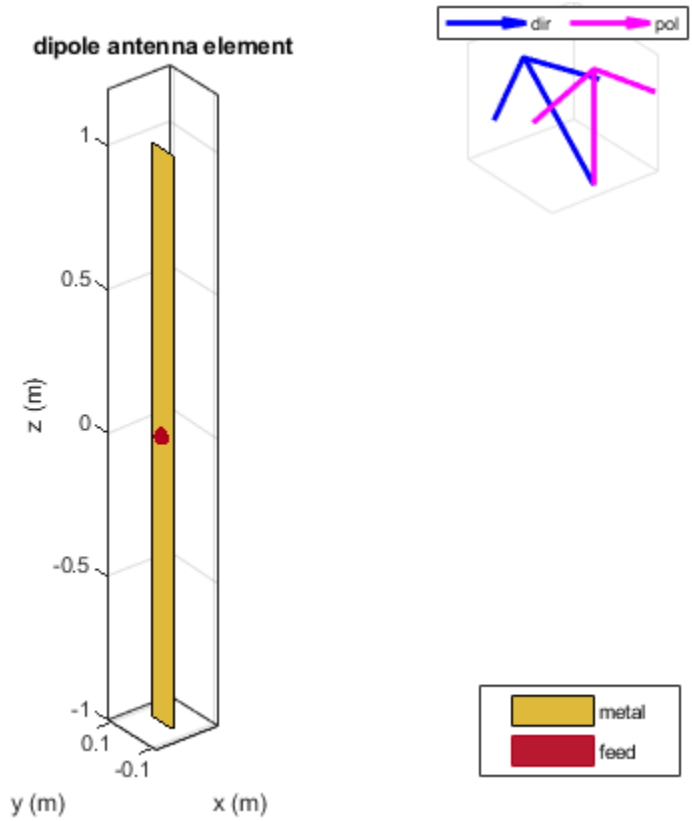
```
current(p, 70e6);
```



Antenna Excited By Plane Wave In Arbitrary Direction

Consider a dipole excited by a plane wave.

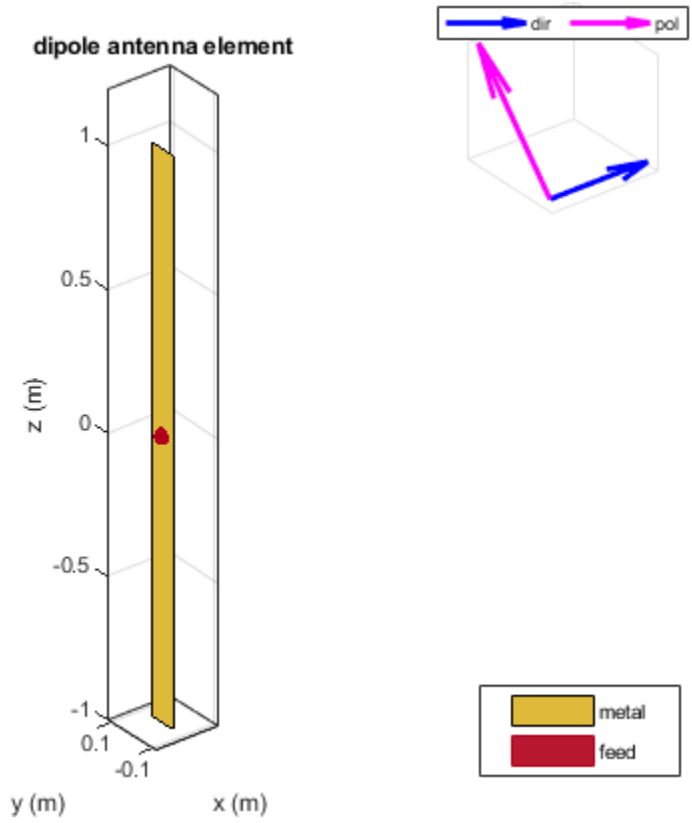
```
p = planeWaveExcitation;
p.Direction = [0 1 1];
show(p)
```



If you use the above option, any analysis of this antenna will error out as the polarization and direction vector are not orthogonal to each other.

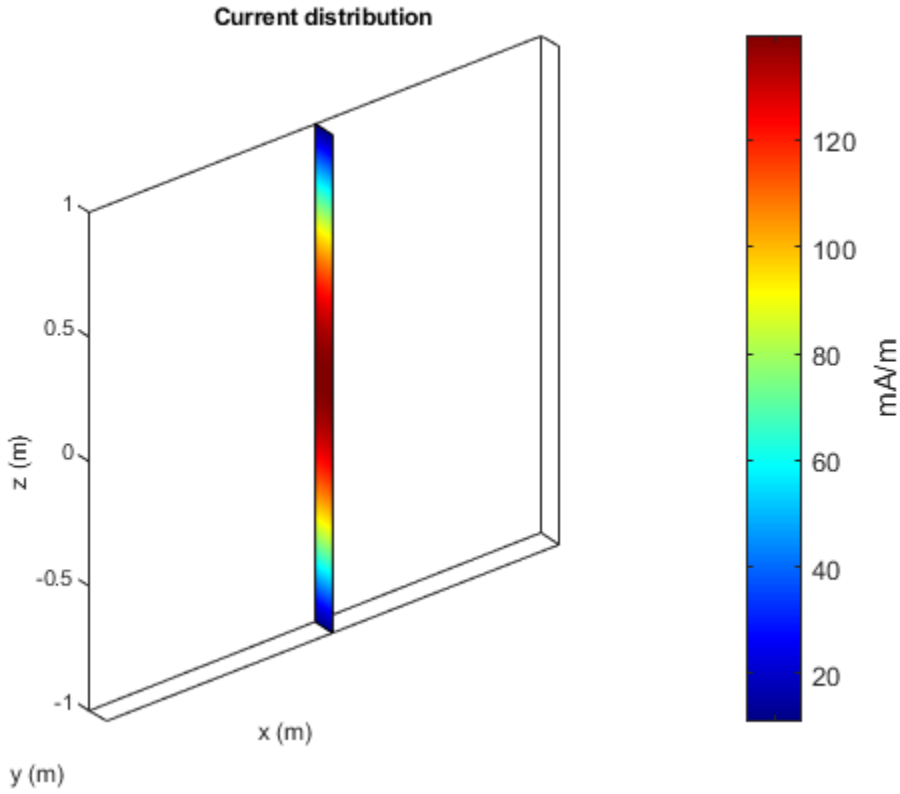
Use the cross-product function to find the appropriate polarization direction of such a wave.

```
p = planeWaveExcitation;
p.Polarization = cross(p.Direction, [0 1 1]);
show(p);
```



Calculate the current distribution of the antenna.

```
current(p,75e6);
```



References

[1] Balanis, C. A. *Antenna Theory. Analysis and Design*. 3rd Ed. Hoboken, NJ: John Wiley & Sons, 2005.

See Also

`dipole` | `linearArray`

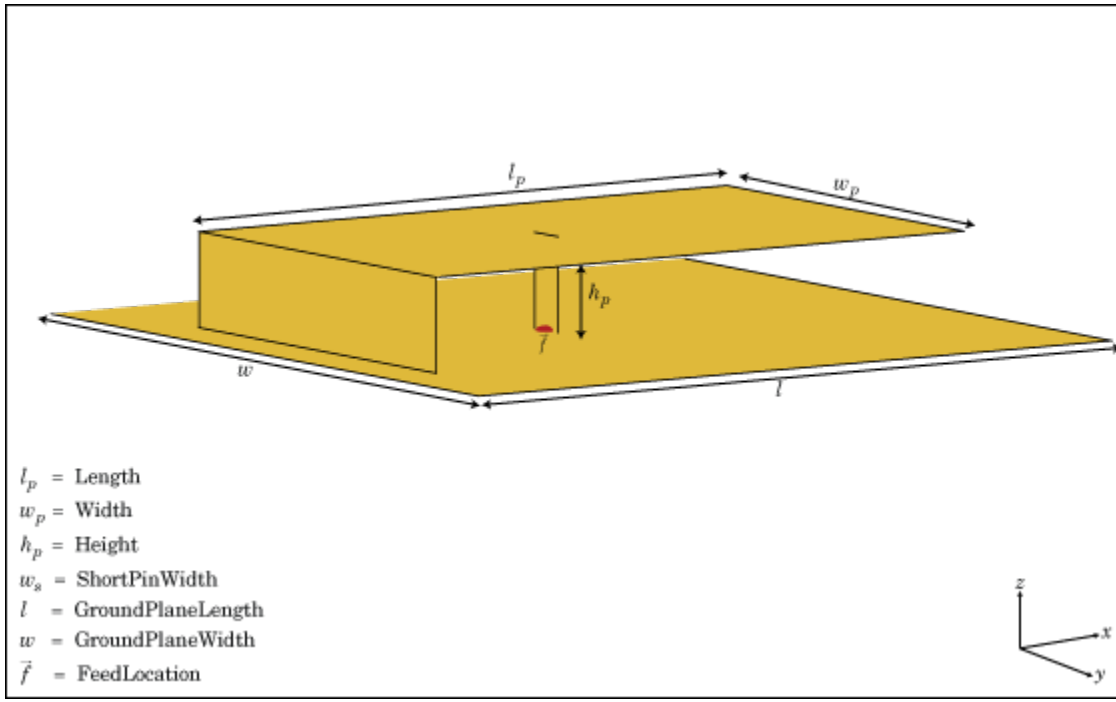
Introduced in R2017a

pifa

Create planar inverted-F antenna

Description

The `pifa` object is a planar inverted-F antenna. The default PIFA antenna is centered at the origin. The feed point is along the length of the antenna.



Creation

Syntax

```
pf = pifa
pf = pifa(Name,Value)
```

Description

`pf = pifa` class to create a planar inverted-F antenna.

`pf = pifa(Name,Value)` class to create a planar inverted-F antenna, with additional properties specified by one, or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Length` — PIFA antenna length

0.0300 (default) | scalar

PIFA antenna length, specified as a scalar in meters. By default, the length is measured along the x-axis.

Example: `'Length',75e-3`

Data Types: `double`

`Width` — PIFA antenna width

0.0200 (default) | scalar

PIFA antenna width, specified as a scalar in meters. By default, the width is measured along the y-axis.

Example: `'Width',35e-3`

Data Types: `double`

`Height` — Height of substrate

0.0100 (default) | scalar

Height of the substrate, specified as a scalar in meters.

Example: `'Height',37e-3`

Data Types: `double`

`Substrate` — Type of dielectric material

'Air' (default) | object

Type of dielectric material used as a substrate, specified as an object. For more information see, `dielectric`. For more information on dielectric substrate meshing, see “Meshing”.

Note The substrate dimensions must be equal to the groundplane dimensions.

Example: `d = dielectric('FR4'); 'Substrate',d`

Example: `d = dielectric('FR4'); pf.Substrate = d`

`GroundPlaneLength` — Ground plane length

0.0360 (default) | scalar

Ground plane length, specified as a scalar in meters. By default, ground plane length is measured along the x-axis. Setting `'GroundPlaneLength'` to `Inf`, uses the infinite ground plane technique for antenna analysis.

Example: `'GroundPlaneLength',3`

Data Types: `double`

`GroundPlaneWidth` — Ground plane width

0.0360 (default) | scalar

Ground plane width, specified as a scalar in meters. By default, ground plane width is measured along the y-axis. Setting `'GroundPlaneWidth'` to `Inf`, uses the infinite ground plane technique for antenna analysis.

Example: `'GroundPlaneWidth',2.5`

Data Types: `double`

`PatchCenterOffset` — Signed distance from center along length and width of ground plane

`[0 0]` (default) | two-element vector

Signed distance from the center along length and width of the ground plane, specified as a two-element vector in meters. Use this property to adjust the location of the patch relative to the ground plane.

Example: `'PatchCenterOffset',[0.01 0.01]`

Data Types: `double`

`ShortPinWidth` — Shorting pin width of patch

0.0200 (default) | scalar

Shorting pin width of patch, specified as a scalar in meters. By default, the shorting pin width is measured along the y-axis.

Example: `'ShortPinWidth',3`

Data Types: `double`

`FeedOffset` — Signed distance of feedpoint from origin

[–0.0020 0] (default) | two-element vector

Signed distance from center along length and width of ground plane, specified as a two-element vector. Use this property to adjust the location of the feedpoint relative to ground plane and patch.

Example: `'FeedOffset',[0.01 0.01]`

Data Types: `double`

`Load` — Lumped elements

[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `pf.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays

impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

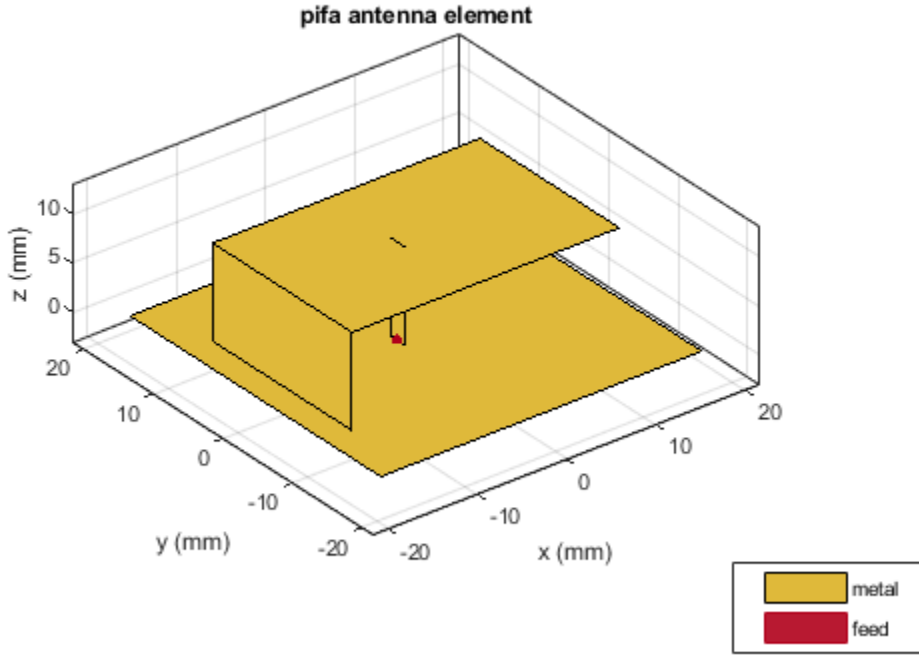
Create and View Planar Inverted-F Antenna(PIFA) Antenna

Create and view a PIFA antenna with 30mm length, 20mm width over a 35mm x 35mm ground plane, and feedpoint at (-2mm,0,0).

```
pf = pifa
```

```
pf =
  pifa with properties:

              Length: 0.0300
               Width: 0.0200
              Height: 0.0100
           Substrate: [1x1 dielectric]
    GroundPlaneLength: 0.0360
     GroundPlaneWidth: 0.0360
    PatchCenterOffset: [0 0]
        ShortPinWidth: 0.0200
           FeedOffset: [-0.0020 0]
                 Tilt: 0
            TiltAxis: [1 0 0]
                 Load: [1x1 lumpedElement]
```

```
show(pf)
```



Radiation Pattern of PIFA Antenna

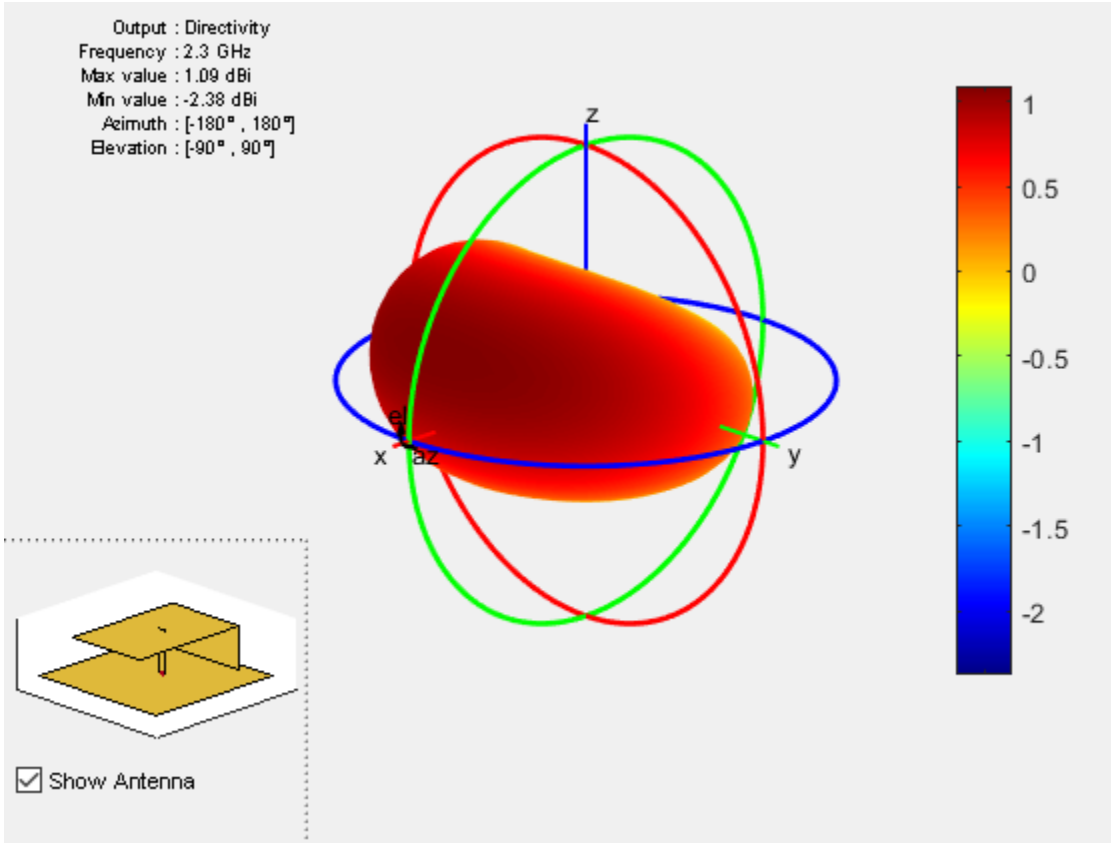
Plot the radiation pattern of a PIFA antenna at a frequency of 2.3 GHz.

```
pf = pifa('Length',30e-3, 'Width',20e-3, 'GroundPlaneLength',35e-3,...
        'GroundPlaneWidth',35e-3)
```

```
pf =
  pifa with properties:

              Length: 0.0300
               Width: 0.0200
```

```
              Height: 0.0100
           Substrate: [1x1 dielectric]
    GroundPlaneLength: 0.0350
     GroundPlaneWidth: 0.0350
    PatchCenterOffset: [0 0]
        ShortPinWidth: 0.0200
           FeedOffset: [-0.0020 0]
                 Tilt: 0
            TiltAxis: [1 0 0]
                 Load: [1x1 lumpedElement]
```

```
pattern(pf,2.3e9);
```

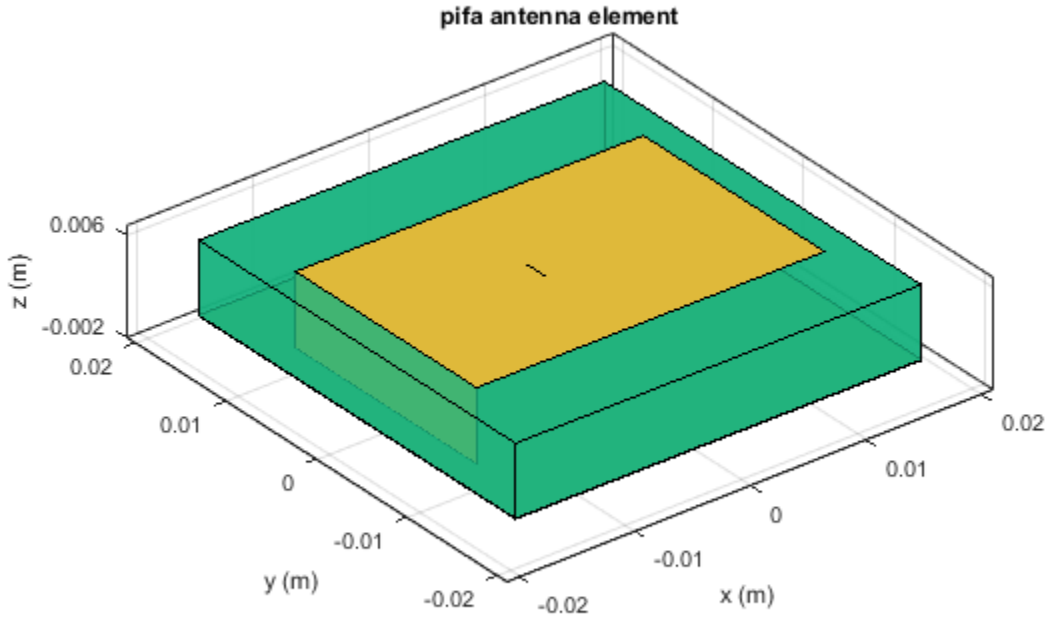


Impedance of PIFA Antenna

Create a PIFA antenna using a dielectric substrate **'RO4725JXR'**.

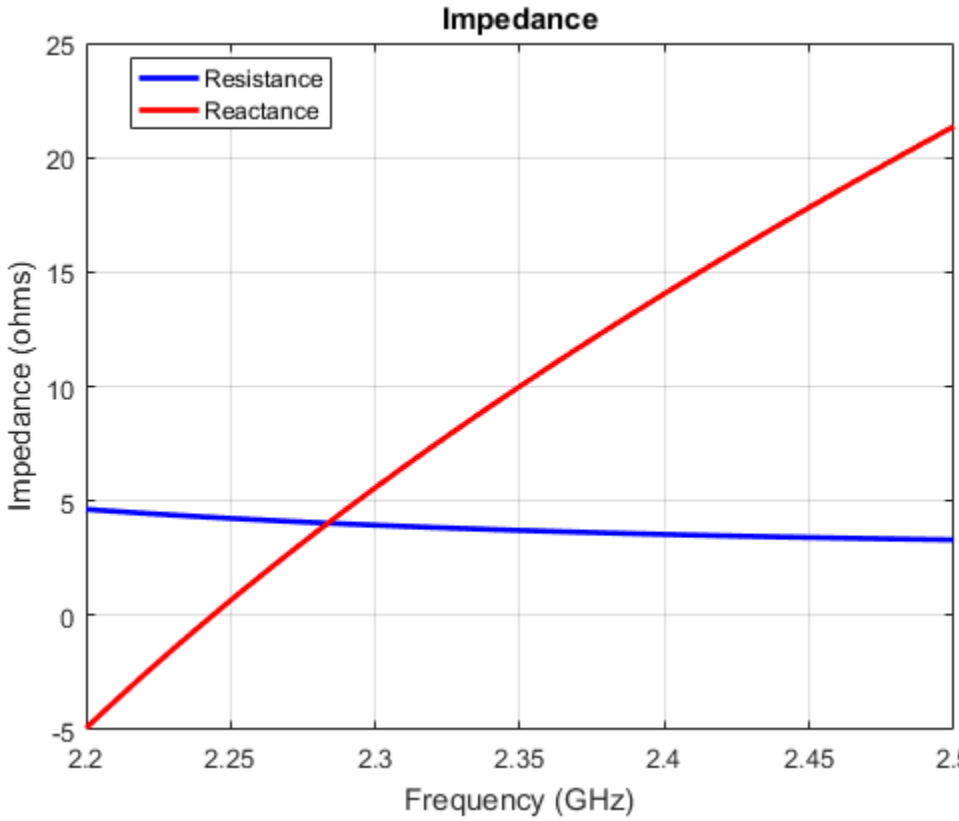
```
d = dielectric('RO4725JXR');
pf = pifa('Length',30e-3, 'Width',20e-3,'Height',0.0060, 'GroundPlaneLength',35e-3, ...
          'GroundPlaneWidth', 35e-3,'Substrate',d)
show(pf)
```

```
pf =

  pifa with properties:

              Length: 0.0300
               Width: 0.0200
              Height: 0.0060
           Substrate: [1x1 dielectric]
    GroundPlaneLength: 0.0350
     GroundPlaneWidth: 0.0350
    PatchCenterOffset: [0 0]
        ShortPinWidth: 0.0200
           FeedOffset: [-0.0020 0]
                 Tilt: 0
            TiltAxis: [1 0 0]
```



Calculate the impedance of the antenna over a frequency range of 2-2.6 GHz.

```
impedance(pf,linspace(2.2e9,2.5e9,31));
```



References

[1] Balanis, C.A. *Antenna Theory. Analysis and Design*, 3rd Ed. New York: Wiley, 2005.

See Also

`invertedF` | `invertedL` | `patchMicrostrip`

Topics

“Rotate Antenna and Arrays”

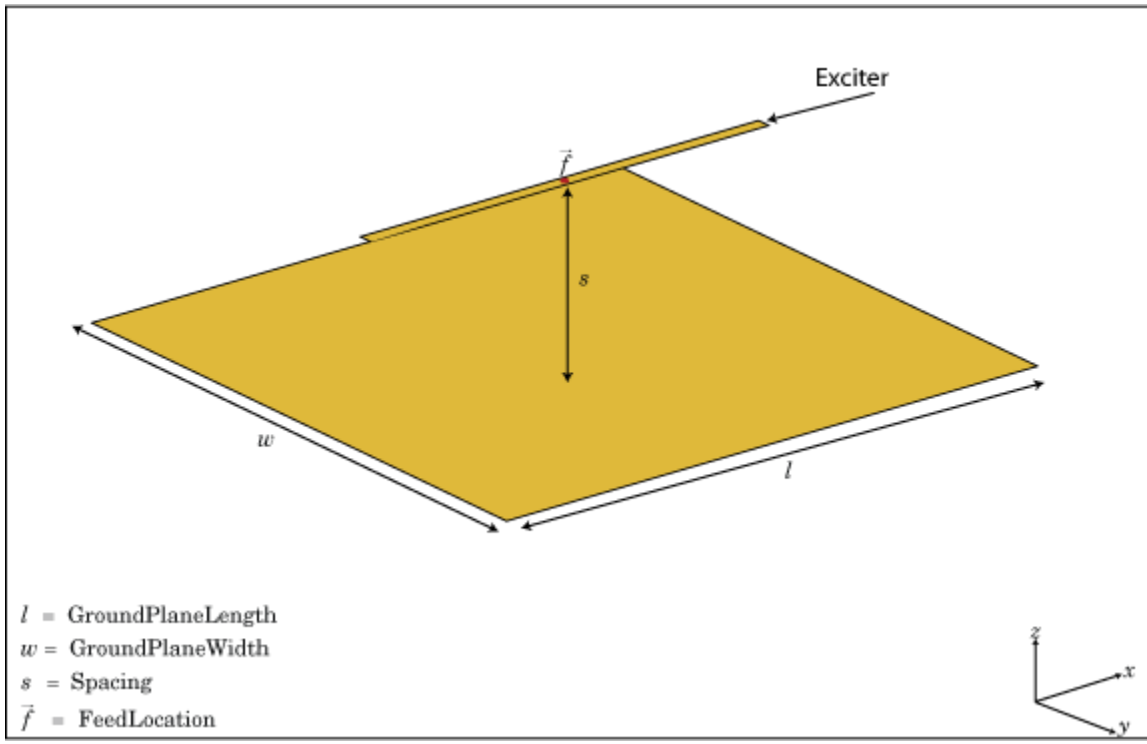
Introduced in R2015a

reflector

Create reflector-backed antenna

Description

The `reflector` object is a reflector-backed antenna on the X-Y-Z plane. The default reflector antenna uses a dipole as an exciter. The feed point is on the exciter.



Creation

Syntax

```
rf = reflector
rf = reflector(Name,Value)
```

Description

`rf = reflector` creates a reflector backed antenna located in the X-Y-Z plane. By default, dimensions are chosen for an operating frequency of 1 GHz.

`rf = reflector(Name,Value)` creates a reflector backed antenna, with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Exciter` — Antenna type used as exciter
`dipole` (default) | object

Antenna type used as an exciter, specified as an object. Except reflector and cavity antenna elements, you can use all the single elements in the Antenna Toolbox as an exciter.

Example: `'Exciter',dipole`

`Substrate` — Type of dielectric material
'Air' (default) | object

Type of dielectric material used as a substrate, specified as an object. For more information see, `dielectric`. For more information on dielectric substrate meshing, see “Meshing”.

Note The substrate dimensions must be equal to the groundplane dimensions.

Example: `d = dielectric('FR4'); 'Substrate',d`

Example: `d = dielectric('FR4'); rf.Substrate = d`

`GroundPlaneLength` — Reflector length along X-axis
0.2000 (default) | scalar

Reflector length along the x-axis, specified a scalar in meters. By default, ground plane length is measured along the x-axis. Setting `'GroundPlaneLength'` to`Inf`, uses the infinite ground plane technique for antenna analysis. You can also set the `'GroundPlaneLength'` to zero.

Example: `'GroundPlaneLength',3`

Data Types: `double`

`GroundPlaneWidth` — Reflector width along Y-axis
0.2000 (default) | scalar

Reflector width along the y-axis, specified as a scalar in meters. By default, ground plane width is measured along the y-axis. Setting `'GroundPlaneWidth'` to`Inf`, uses the infinite ground plane technique for antenna analysis. You can also set the `'GroundPlaneWidth'` to zero.

Example: `'GroundPlaneWidth',2.5`

Data Types: `double`

`Spacing` — Distance between reflector and exciter
0.0750 (default) | scalar

Distance between the reflector and the exciter, specified as a scalar in meters. By default, the exciter is placed along the x-axis.

Example: `'Spacing',7.5e-2`

Data Types: `double`

`Load` — Lumped elements
[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `rf.Load = lumpedElement('Impedance',75)`

`EnableProbeFeed` — Create probe feed from backing structure to exciter
`0` (default) | `1`

Create probe feed from backing structure to exciter, specified as `0` or `1`. By default, probe feed is not enabled.

Example: `'EnableProbeFeed',1`

Data Types: `double`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

Create and View Reflector-Backed Dipole Antenna

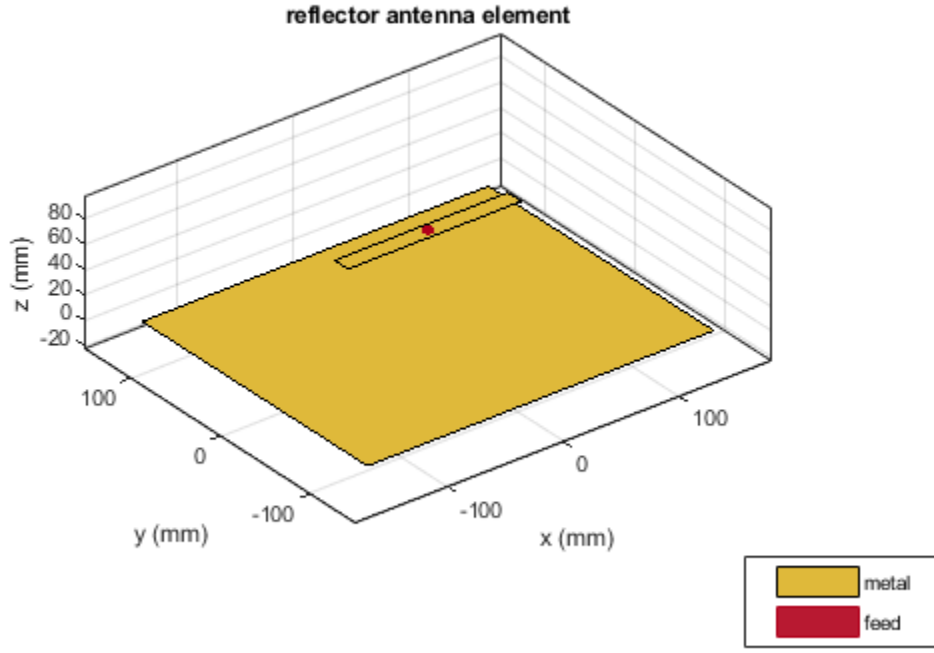
Create a reflector backed dipole that has 30cm length, 25cm width and spaced 7.5cm from the dipole for operation at 1 GHz.

```
d = dipole('Length',0.15,'Width',0.015, 'Tilt',90,'TiltAxis',[0 1 0]);
rf = reflector('GroundPlaneLength',30e-2, 'GroundPlaneWidth',25e-2,...
               'Spacing',7.5e-2);
rf.Exciter = d
```

```
rf =
  reflector with properties:

              Exciter: [1x1 dipole]
            Substrate: [1x1 dielectric]
```

```
    GroundPlaneLength: 0.3000
     GroundPlaneWidth: 0.2500
              Spacing: 0.0750
       EnableProbeFeed: 0
                  Tilt: 0
             TiltAxis: [1 0 0]
                 Load: [1x1 lumpedElement]
```

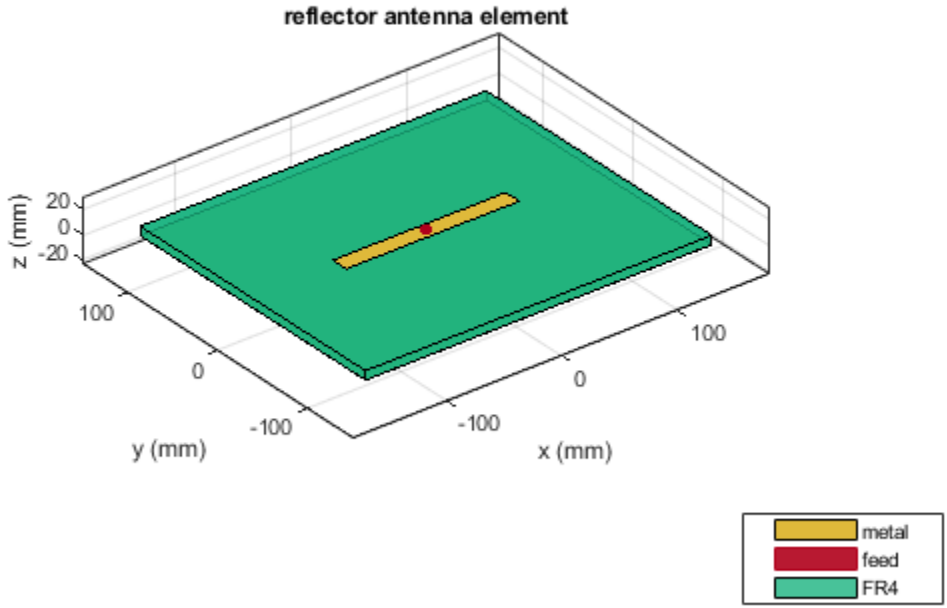
```
show(rf)
```



Radiation Pattern of Reflector Backed Antenna

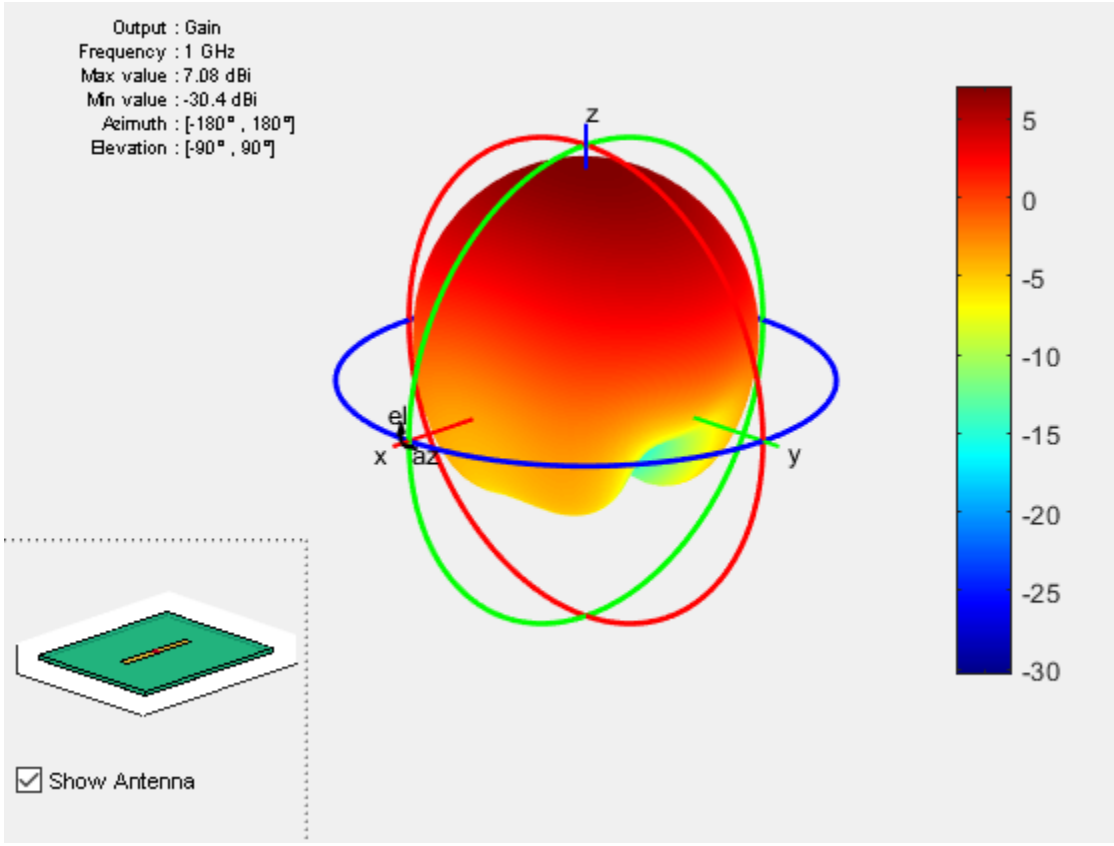
Create a reflector backed dipole antenna using a dielectric substrate **'FR4'**.

```
d = dielectric('FR4');
di = dipole('Length',0.15,'Width',0.015, 'Tilt',90,'TiltAxis','Y');
rf = reflector('GroundPlaneLength',30e-2, 'GroundPlaneWidth',25e-2, ...
               'Spacing',7.5e-3,'Substrate',d);
rf.Exciter = di;
show(rf)
```



Plot the radiation pattern of the antenna at a frequency of 1 GHz.

```
figure
pattern(rf,1e9)
```



Create Reflector-Backed Antenna Over Infinite Ground Plane

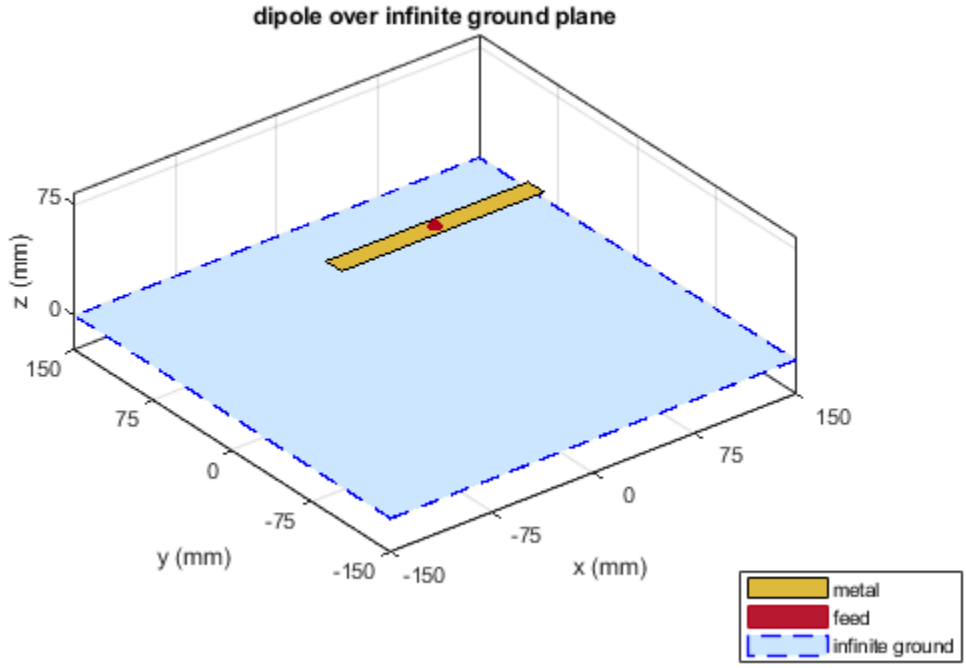
Create a reflector backed dipole that has 30cm length, 25cm width and spaced 7.5cm from the dipole for operation at 1 GHz.

```
d = dipole('Length',0.15,'Width',0.015, 'Tilt',90,'TiltAxis',[0 1 0]);
rf = reflector('GroundPlaneLength',inf, 'GroundPlaneWidth',25e-2,...
               'Spacing',7.5e-2);
rf.Exciter = d
```

```
rf =
  reflector with properties:
```

```
              Exciter: [1x1 dipole]
            Substrate: [1x1 dielectric]
    GroundPlaneLength: Inf
     GroundPlaneWidth: 0.2500
              Spacing: 0.0750
      EnableProbeFeed: 0
                 Tilt: 0
             TiltAxis: [1 0 0]
                 Load: [1x1 lumpedElement]
```

```
show(rf)
```



Antenna On Dielectric Substrate - Compare Gain Values

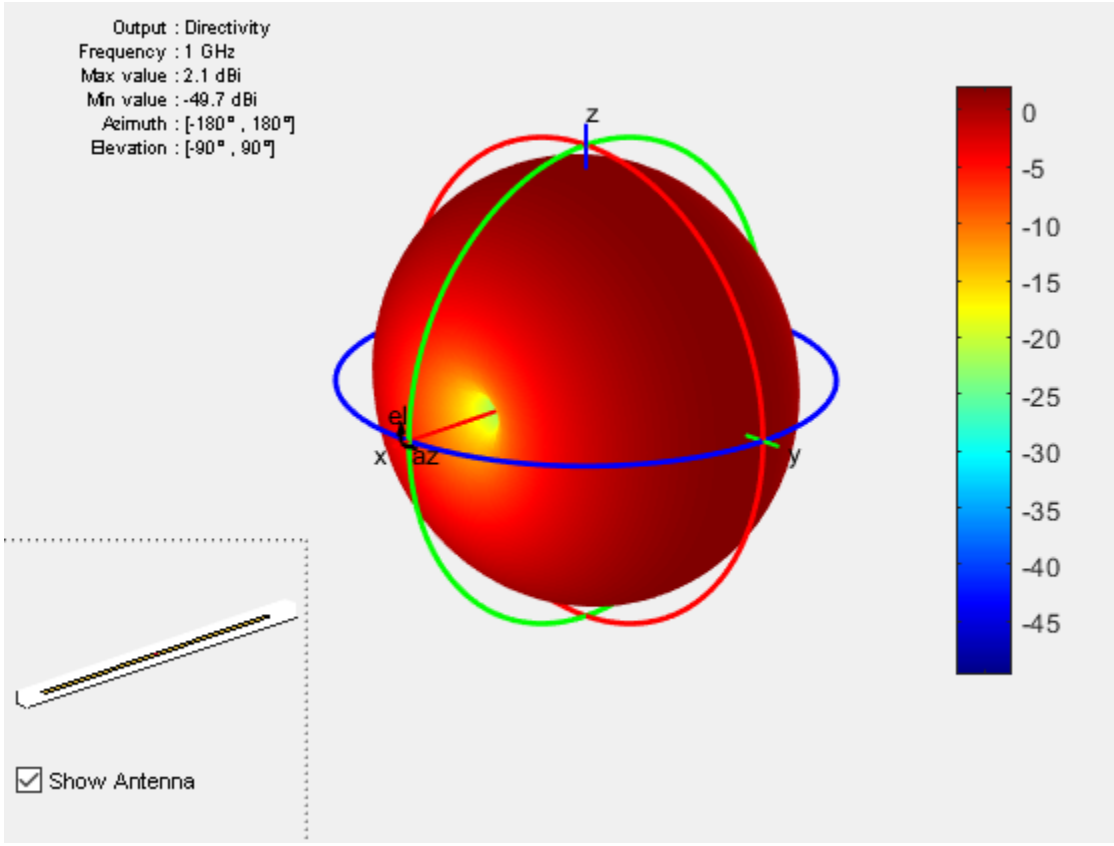
Compare the gain values of a dipole antenna in free space and dipole antenna on a substrate.

Design a dipole antenna at 1 GHz.

```
d = design(dipole,1e9);
l_by_w = d.Length/d.Width;
d.Tilt = 90;
d.TiltAxis = [0 1 0];
```

Plot the radiation pattern of the dipole in free space at 1GHz.

```
figure
pattern(d,1e9);
```



Use FR4 as the dielectric substrate.

```
t = dielectric('FR4')
eps_r = t.EpsilonR;
lambda_0 = physconst('lightspeed')/1e9;
lambda_d = lambda_0/sqrt(eps_r);
```

```
t =

  dielectric with properties:

          Name: 'FR4'
      EpsilonR: 4.8000
```

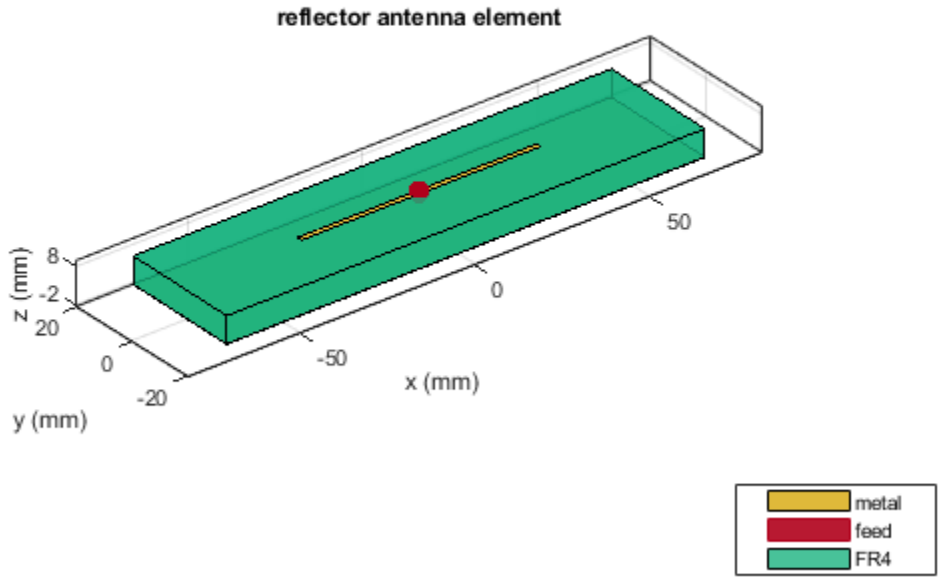
```
    LossTangent: 0.0260
      Thickness: 0.0060

For more materials see <a href="matlab:openDielectricCatalog">catalog</a>
```

Adjust the length of the dipole based on the wavelength.

```
d.Length = lambda_d/2;
d.Width = d.Length/l_by_w;
```

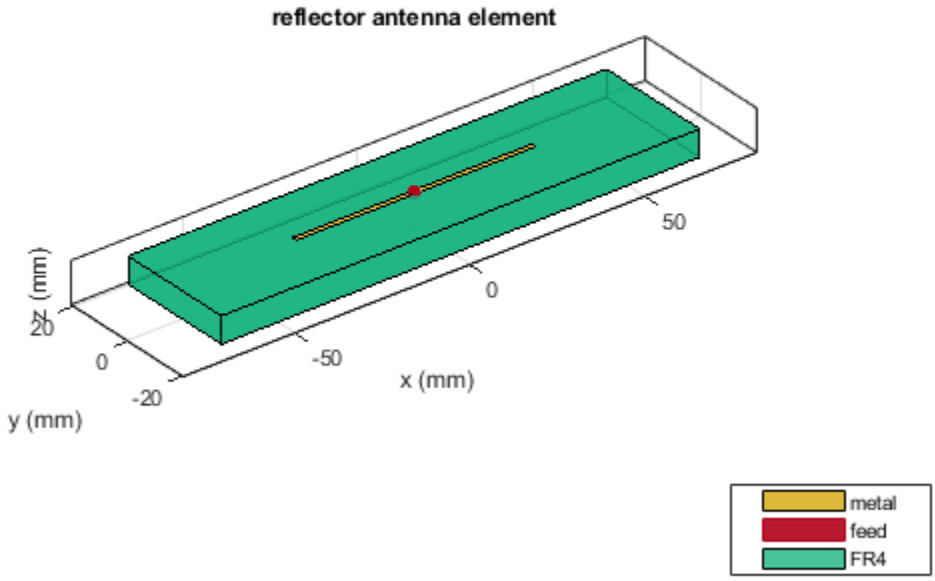
Design a reflector at 1 GHz with the dipole as the excitor and FR4 as the substrate.

```
rf = design(reflector,1e9);
rf = reflector('Exciter',d,'Spacing',7.5e-3,'Substrate',t);
rf.GroundPlaneLength = lambda_d;
rf.GroundPlaneWidth = lambda_d/4;
figure
show(rf)
```



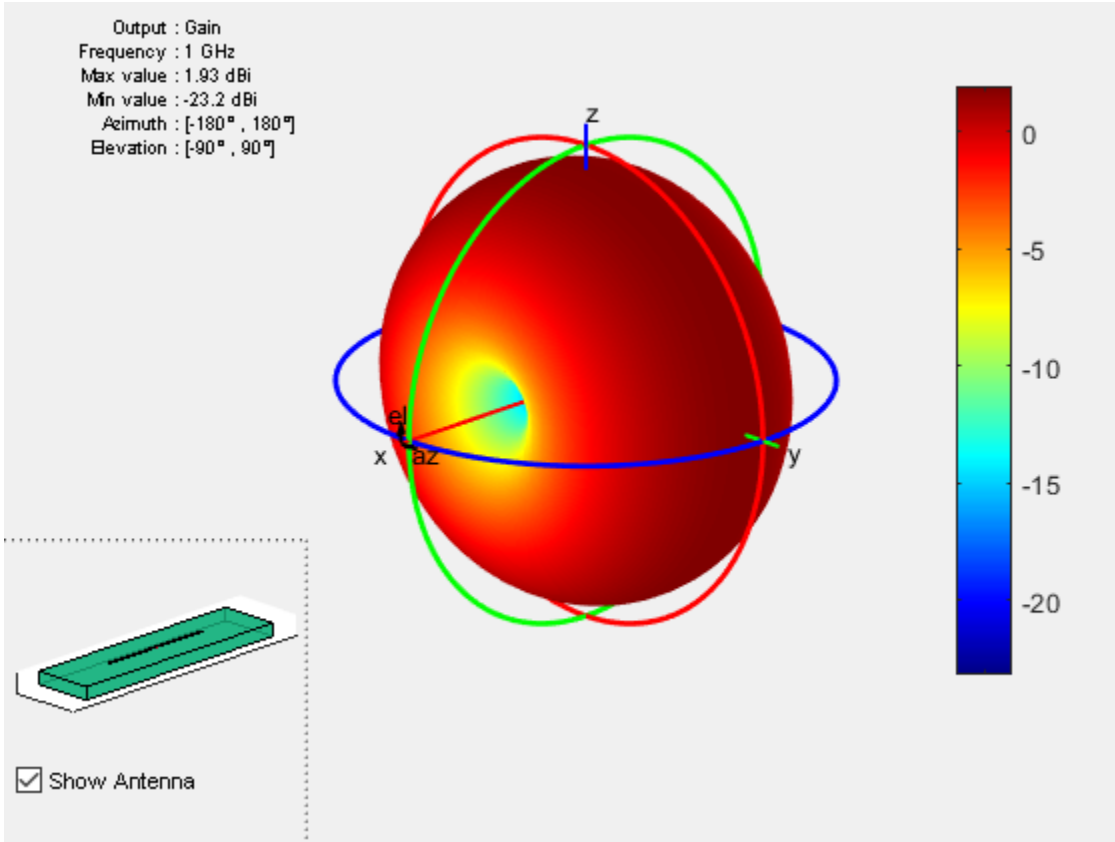
Remove the groundplane for plotting the gain of the dipole on the substrate.

```
rf.GroundPlaneLength = 0;
show(rf)
```



Plot the radiation pattern of the dipole on the substrate at 1 GHz.

```
figure
pattern(rf,1e9);
```



Compare the gain values.

- Gain of the dipole in free space = 2.11 dBi
- Gain of the dipole on substrate = 1.93 dBi

References

[1] Balanis, C.A. *Antenna Theory. Analysis and Design*, 3rd Ed. New York: Wiley, 2005.

See Also

`cavity` | `spiralArchimedean` | `spiralEquiangular`

Topics

“Rotate Antenna and Arrays”

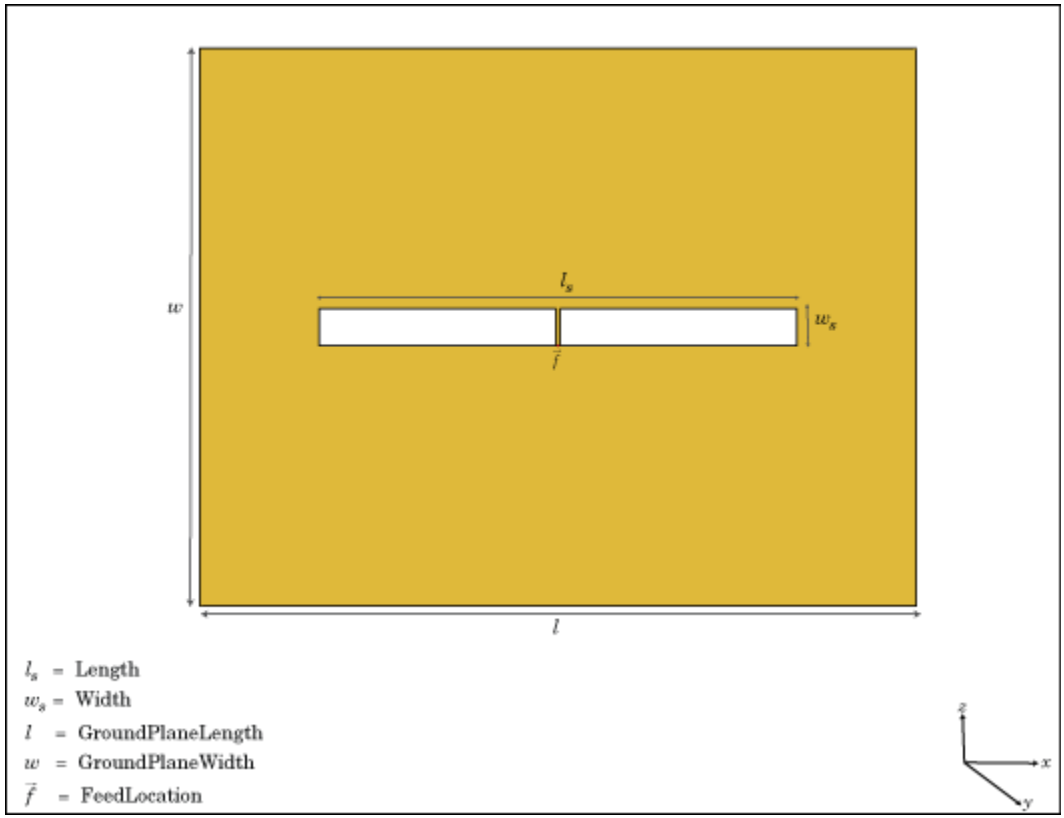
Introduced in R2015a

slot

Create rectangular slot antenna on ground plane

Description

The `slot` object is a rectangular slot antenna on a ground plane. The default slot has its first resonance at 130 MHz.



Creation

Syntax

```
s = slot
s = slot(Name,Value)
```

Description

`s = slot` creates a rectangular slot antenna on a ground plane.

`s = slot(Name,Value)` creates a rectangular slot antenna, with additional properties specified by one, or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain default values.

Properties

`Length` — Slot length
1 (default) | scalar

Slot length, specified as a scalar in meters.

Example: `'Length',2`

Data Types: `double`

`Width` — Slot width
0.1000 (default) | scalar

Slot width, specified a scalar in meters.

Example: `'Width',0.02`

Data Types: `double`

`SlotCenter` — Slot antenna center
`[0 0 0]` (default) | three-element vector in Cartesian coordinates

Slot antenna center, specified as a three-element vector in Cartesian coordinates.

Example: `'SlotCenter',[8 0 0]`

Data Types: `double`

`GroundPlaneLength` — Ground plane length

1.5000 (default) | scalar

Ground plane length, specified as a scalar in meters. By default, the length is measured along the x-axis.

Example: `'GroundPlaneLength',3`

Data Types: `double`

`GroundPlaneWidth` — Ground plane width

1.5000 (default) | scalar

Ground plane width, specified as a scalar in meters. By default, the width is measured along the y-axis.

Example: `'GroundPlaneWidth',4`

Data Types: `double`

`FeedOffset` — Distance from center along x-axis

0 (default) | scalar

Distance from center along x-axis, specified as a scalar in meters. Offset from slot center is measured along the length.

Example: `'FeedOffset',3`

Data Types: `double`

Load — Lumped elements

[1x1 LumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `s.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface

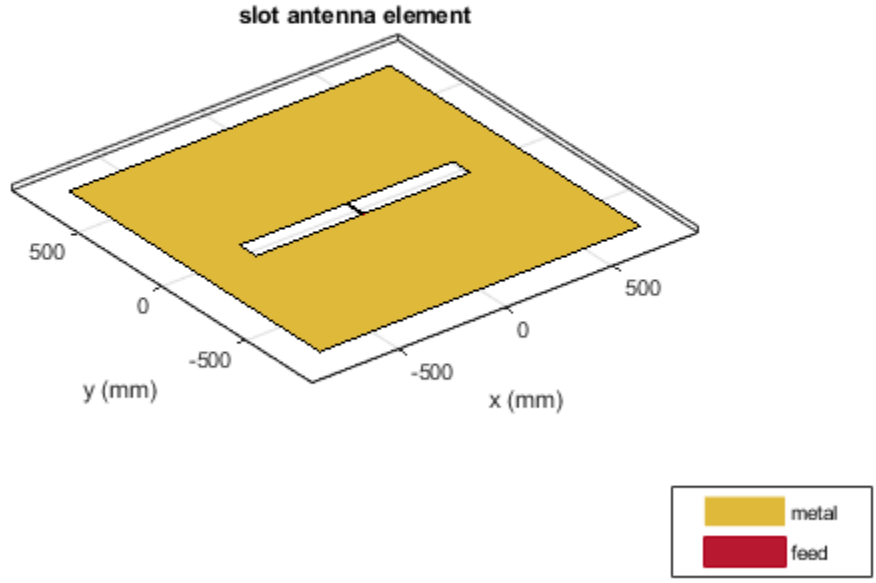
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

Create and View Slot Antenna

Create and view a slot antenna that has 1m length and 100mm width.

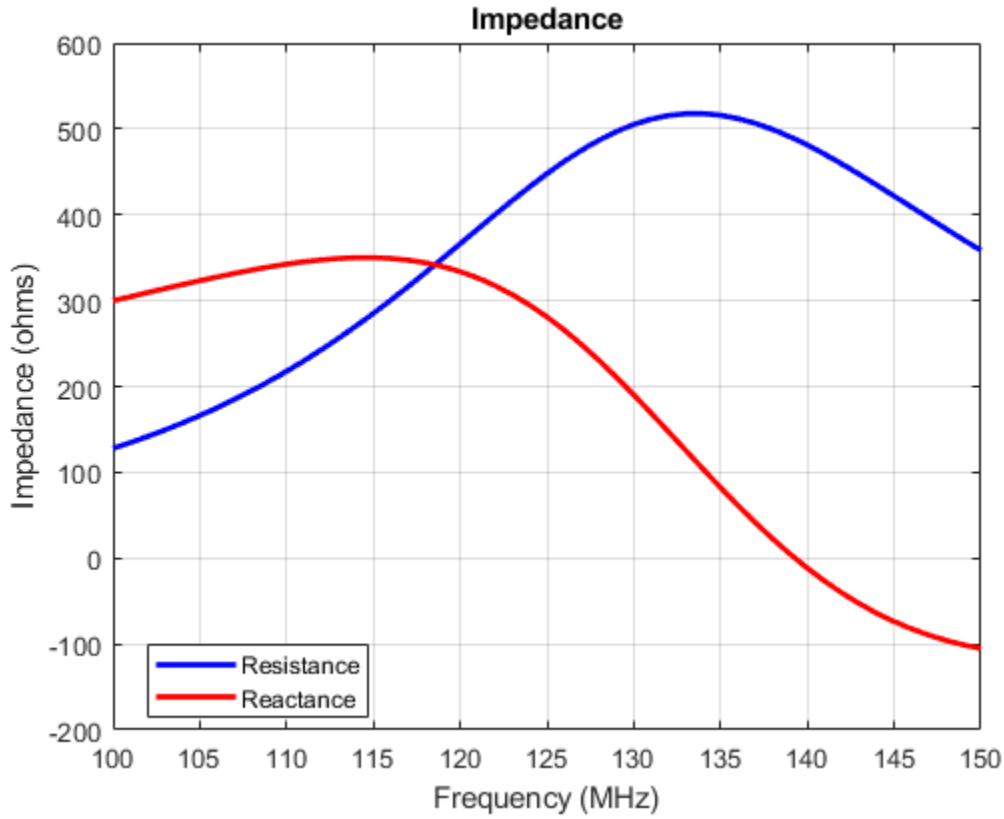
```
s = slot('Length',1,'Width',0.1);
show(s)
```



Impedance of Slot Antenna

Calculate and plot the impedance of a slot antenna over a frequency range of 100-150 MHz.

```
s = slot('Length',1,'Width',0.1);
impedance(s,linspace(100e6,150e6,51));
```



References

[1] Balanis, C.A. *Antenna Theory. Analysis and Design*, 3rd Ed. New York: Wiley, 2005.

See Also

`pifa` | `vivaldi` | `yagiUda`

Topics

“Rotate Antenna and Arrays”

Introduced in R2015a

spiralArchimedean

Create Archimedean spiral antenna

Description

The `spiralArchimedean` object creates a planar Archimedean spiral antenna on the X-Y plane. The default Archimedean spiral is always center fed and has two arms. The field characteristics of this antenna are frequency independent. A realizable spiral has finite limits on the feeding region and the outermost point of any arm of the spiral. The spiral antenna exhibits a broadband behavior. The outer radius imposes the low frequency limit and the inner radius imposes the high frequency limit. The arm radius grows linearly as a function of the winding angle.

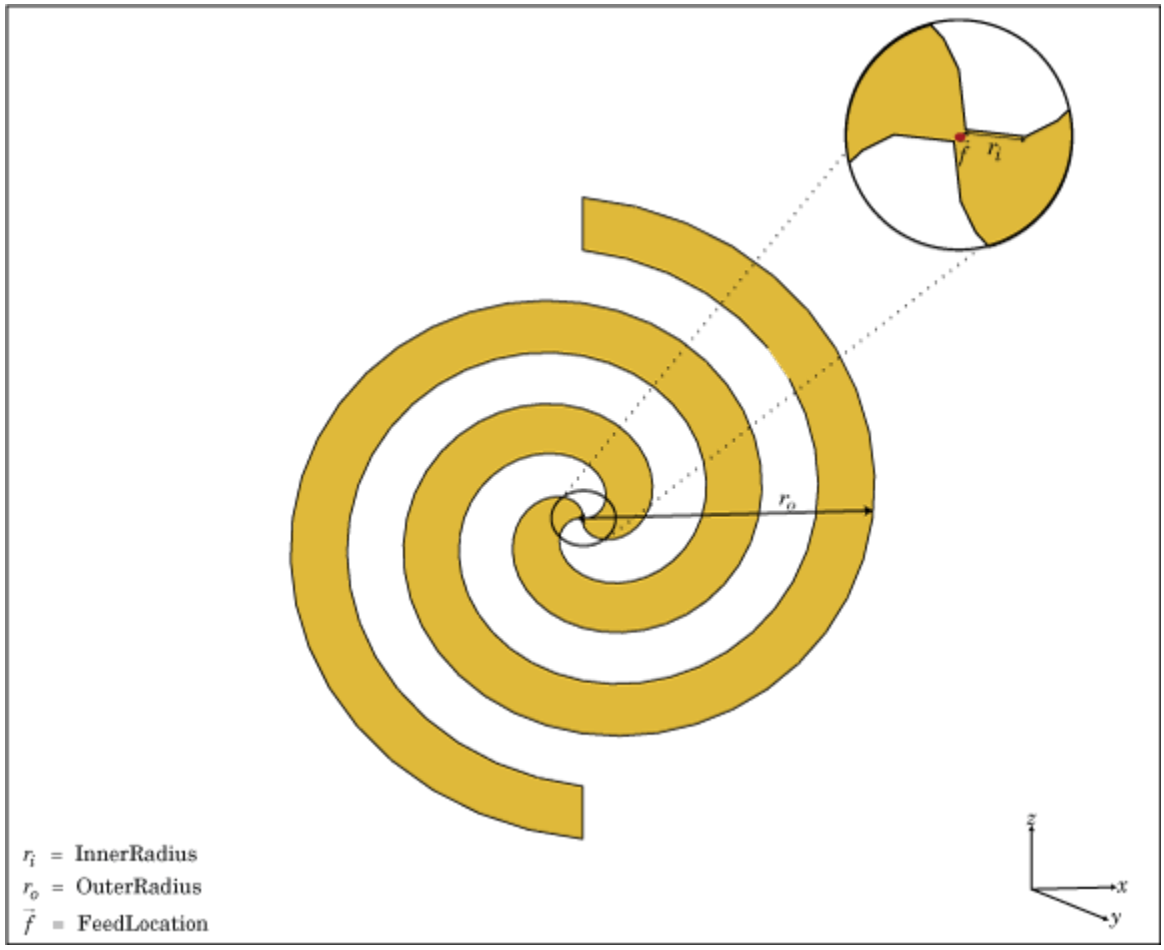
The equation of the Archimedean spiral is:

$$r = r_0 + a\phi$$

where:

- r_0 is the inner radius
- a is the growth rate
- ϕ is the winding angle of the spiral

Archimedean spiral antenna is a self-complimentary structure, where the spacing between the arms and the width of the arms are equal. The default antenna is center fed. The feed point coincides with the origin. The origin is in the X-Y plane.



Creation

Syntax

```
ant = spiralArchimedean
ant = spiralArchimedean(Name,Value)
```

Description

`ant = spiralArchimedean` creates a planar Archimedean spiral on the X-Y plane. By default, the antenna operates over a broadband frequency range of 3–5 GHz.

`ant = spiralArchimedean(Name,Value)` sets properties using one or more name-value pairs. For example, `ant = spiralArchimedean('Truns',6.25)` creates a Archimedean spiral of 6.25 turns.

Output Arguments

`ant` — MATLAB object
scalar `spiralArchimedean` object (default)

MATLAB object, returned as scalar `spiralArchimedean` object.

Properties

`NumArms` — Number of arms
2 (default) | scalar integer

Number of arms, specified as a scalar integer. You can also create a single arm Archimedean spiral by specifying `NumArms` is equal to one.

Example: `'NumArms',1`

Example: `ant.NumArms = 1`

Data Types: `double`

`Turns` — Number of turns of spiral antenna
1.5000 (default) | scalar

Number of turns of the spiral antenna, specified as a scalar.

Example: `'Turns',2`

Example: `ant.Turns = 2`

Data Types: `double`

`InnerRadius` — Inner radius of spiral antenna
5.0000e-04 (default) | scalar

inner radius of the spiral antenna, specified as a scalar in meters.

Example: `'InnerRadius',1e-3`

Example: `ant.InnerRadius = 1e-3`

Data Types: `double`

`OuterRadius` — Outer radius of spiral antenna
0.0398 (default) | scalar

Outer radius of the spiral antenna, specified as a scalar in meters.

Example: `'OuterRadius',1e-3`

Example: `ant.OuterRadius = 1e-3`

Data Types: `double`

`WindingDirection` — Direction of spiral turns (windings)
`'CW'` | `'CCW'`

Direction of the spiral turns (windings), specified as `'CW'` or `'CCW'`.

Example: `'WindingDirection','CW'`

Example: `ant.WindingDirection = CW`

Data Types: `char` | `string`

`Load` — Lumped elements
[1x1 LumpedElement] (default) | lumped element object handle

Lumped elements added to the spiral antenna feed, specified as a lumped element object handle. You can add a load anywhere on the surface of the antenna. By default, it is at the origin. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `ant.Load = lumpedElement('Impedance',75)`

Data Types: `double`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of the antenna, specified as a scalar or vector in degrees.

Example: `'Tilt',90`

Example: `ant.Tilt = [90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array

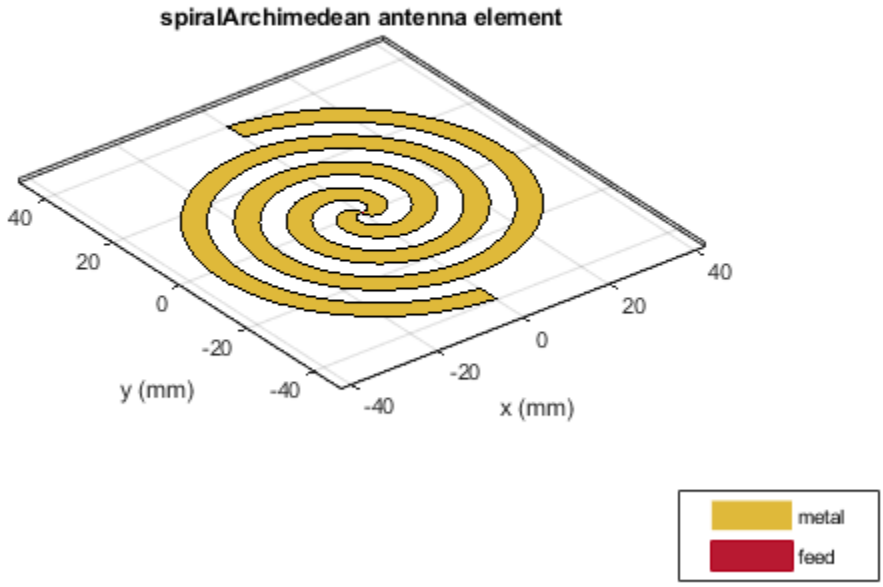
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

Create and View Archimedean Spiral Antenna

Create and view a 2-turn Archimedean spiral antenna with a 1 mm starting radius and 40 mm outer radius.

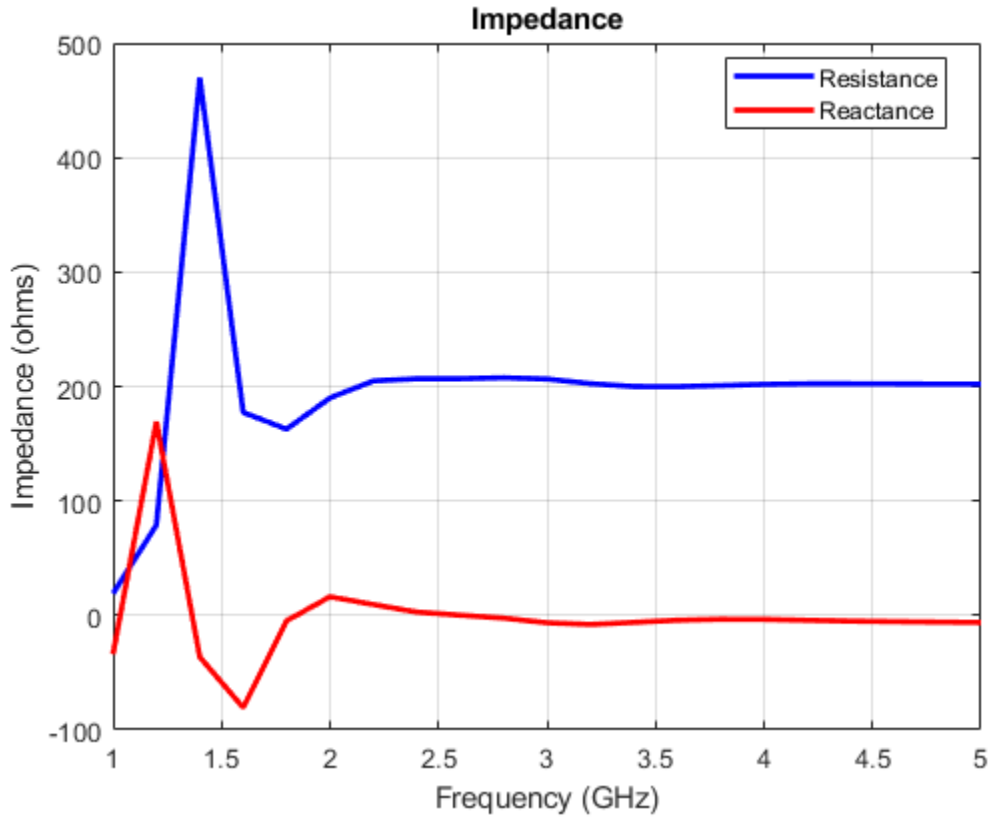
```
sa = spiralArchimedean('Turns',2, 'InnerRadius',1e-3, 'OuterRadius',40e-3);
show(sa)
```



Impedance of Archimedean Spiral Antenna

Calculate the impedance of an Archimedean spiral antenna over a frequency range of 1-5 GHz.

```
sa = spiralArchimedean('Turns',2, 'InnerRadius',1e-3, 'OuterRadius',40e-3);
impedance(sa, linspace(1e9,5e9,21));
```



Single-Arm Archimedean Spiral

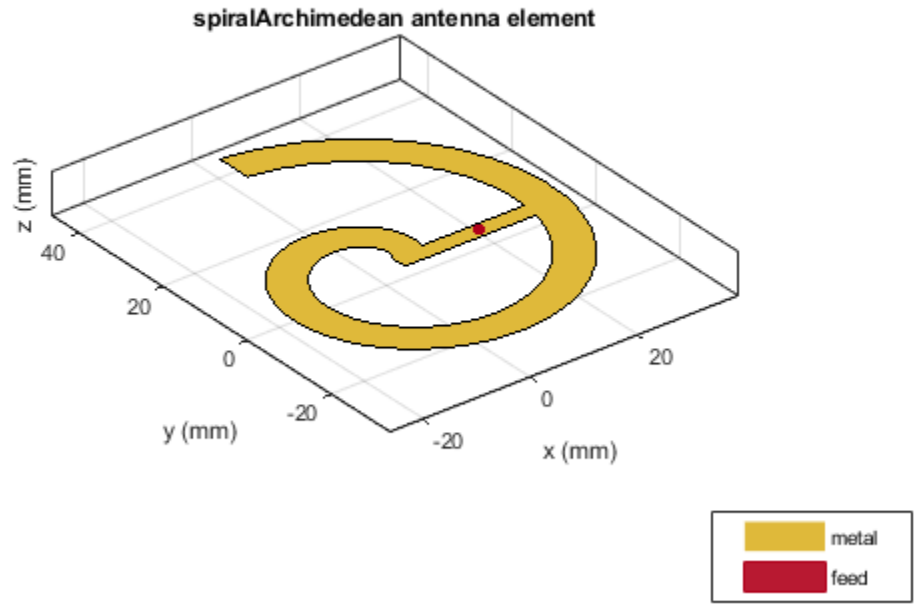
Create and view a single-arm Archimedean spiral.

```
ant = spiralArchimedean;
ant.NumArms = 1
```

```
ant =
  spiralArchimedean with properties:

            NumArms: 1
              Turns: 1.5000
```

```
          InnerRadius: 5.0000e-04
          OuterRadius: 0.0398
     WindingDirection: 'CCW'
                 Tilt: 0
            TiltAxis: [1 0 0]
                 Load: [1x1 lumpedElement]
```

```
show(ant)
```



References

[1] Balanis, C.A. *Antenna Theory. Analysis and Design*, 3rd Ed. New York: Wiley, 2005.

[2] Nakano, H., Oyanagi, H. and Yamauchi, J. “A Wideband Circularly Polarized Conical Beam From a Two-Arm Spiral Antenna Excited in Phase”. *IEEE Transactions on Antennas and Propagation*. Vol. 59, No. 10, Oct 2011, pp. 3518-3525.

[3] Volakis, John. *Antenna Engineering Handbook*, 4th Ed. McGraw-Hill

See Also

`helix` | `spiralEquiangular` | `yagiUda`

Topics

“Rotate Antenna and Arrays”

Introduced in R2015a

spiralEquiangular

Create equiangular spiral antenna

Description

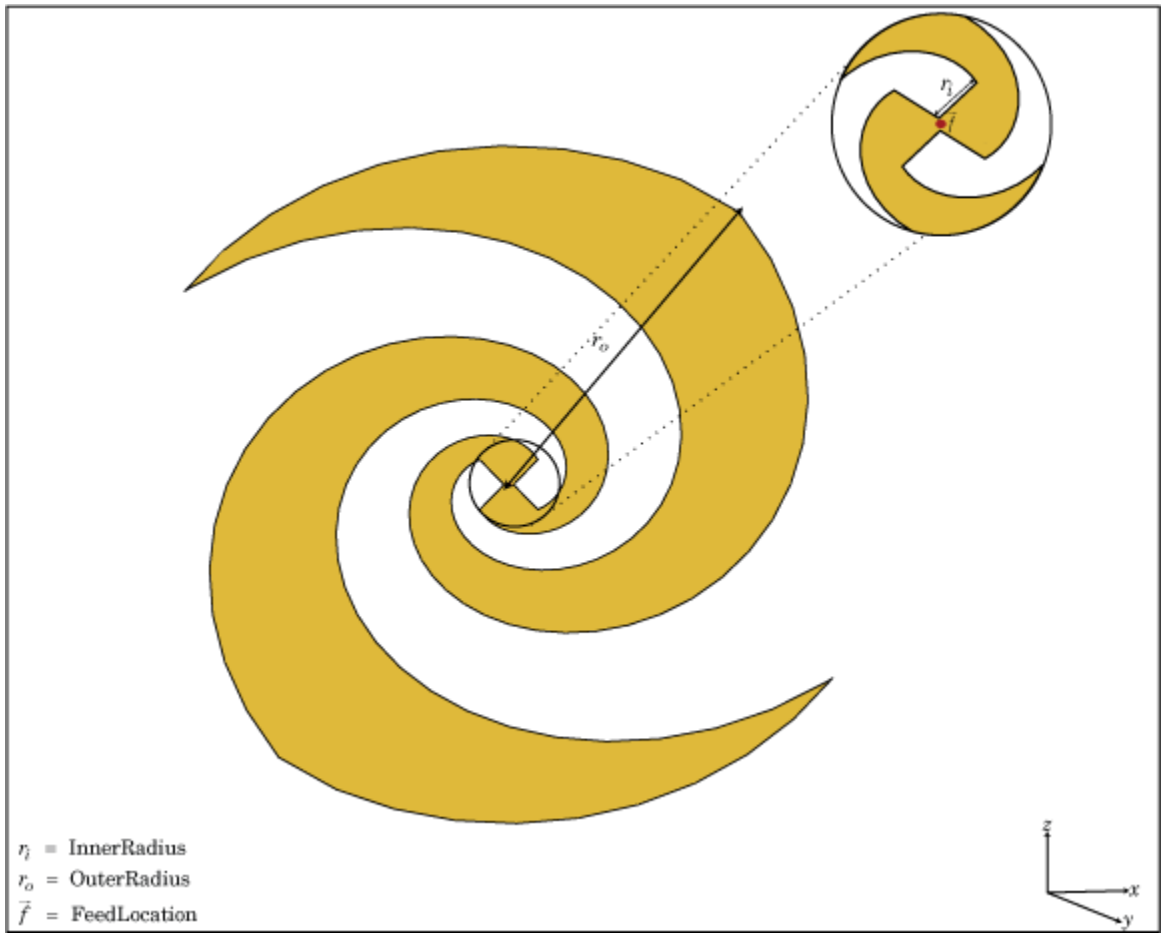
The `spiralEquiangular` object is a planar equiangular spiral antenna on the X-Y plane. The equiangular spiral is always center fed and has two arms. The field characteristics of the antenna are frequency independent. A realizable spiral has finite limits on the feeding region and the outermost point of any arm of the spiral. This antenna exhibits a broadband behavior. The outer radius imposes the low frequency limit and the inner radius imposes the high frequency limit. The arm radius grows linearly as a function of the winding angle. As a result, outer arms of the spiral are shaped to minimize reflections.

The equation of the equiangular spiral is:

$$r = r_0 e^{a\phi}$$

, where:

- r_0 is the starting radius
- a is the growth rate
- ϕ is the winding angle of the spiral



Creation

Syntax

```
se = spiralEquiangular
se = spiralEquiangular(Name,Value)
```

Description

`se = spiralEquiangular` creates a planar equiangular spiral in the X-Y plane. By default, the antenna operates over a broadband frequency 4–10 GHz.

`se = spiralEquiangular(Name,Value)` creates an equiangular spiral antenna, with additional properties specified by one, or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`GrowthRate` — Equiangular spiral growth rate
0.3500 (default) | scalar

Equiangular spiral growth rate, specified as a scalar.

Example: `'GrowthRate',1.2`

Data Types: `double`

`InnerRadius` — Inner radius of spiral
0.0020 (default) | scalar

Inner radius of spiral, specified as a scalar in meters.

Example: `'InnerRadius',1e-3`

Data Types: `double`

`OuterRadius` — Outer radius of spiral
0.0189 (default) | scalar

Outer radius of spiral, specified as a scalar in meters.

Example: `'OuterRadius',1e-3`

Data Types: `double`

`WindingDirection` — Direction of spiral turns (windings)
`'CW'` | `'CCW'`

Direction of spiral turns (windings), specified as `'CW'` or `'CCW'`.

Example: `'WindingDirection','CW'`

Data Types: `char`

Load — Lumped elements

[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `se.Load = lumpedElement('Impedance',75)`

Tilt — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

TiltAxis — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

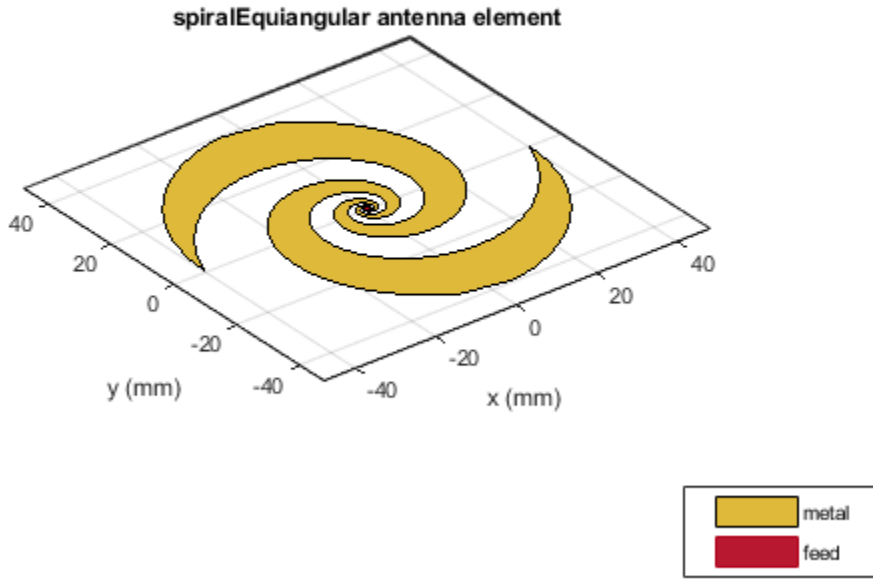
show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

Create and View Equiangular Spiral Antenna

Create and view an equiangular spiral antenna with 0.35 growth rate, 0.65 mm inner radius and 40 mm outer radius.

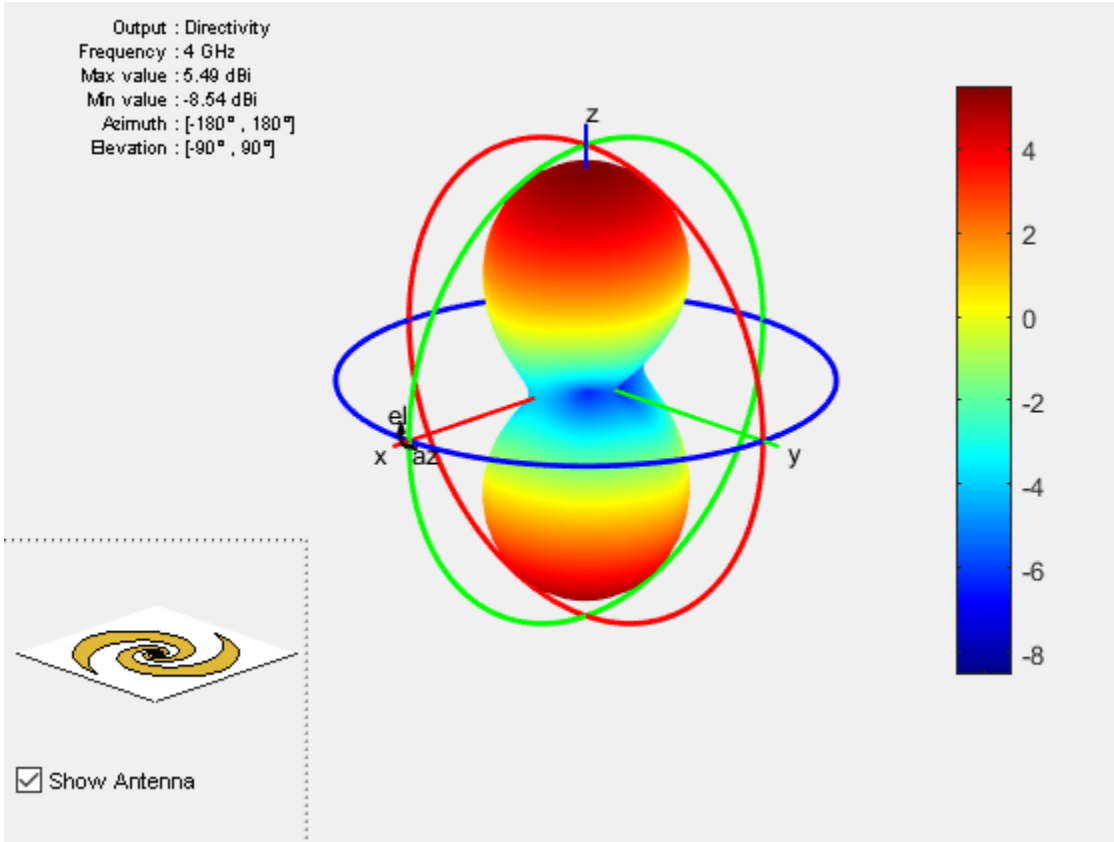
```
se = spiralEquiangular('GrowthRate',0.35, 'InnerRadius',0.65e-3,    ...
                       'OuterRadius',40e-3);
show(se)
```



Radiation Pattern of Equiangular Spiral Antenna

Plot the radiation pattern of equiangular spiral at a frequency of 4 GHz.

```
se = spiralEquiangular('GrowthRate',0.35, 'InnerRadius',0.65e-3, ...
                       'OuterRadius',40e-3);
pattern(se,4e9);
```



References

[1] Dyson, J. The equiangular spiral antenna.” *IRE Transactions on Antennas and Propagation*. Vol.7, Number 2, pp. 181, 187, April 1959.

[2] Nakano, H., K.Kikkawa, N.Kondo, Y.Iitsuka, J.Yamauchi. “Low-Profile Equiangular Spiral Antenna Backed by an EBG Reflector.” *IRE Transactions on Antennas and Propagation*. Vol. 57, No. 25, May 2009, pp. 1309–1318.

[3] McFadden, M., and Scott, W.R. “Analysis of the Equiangular Spiral Antenna on a Dielectric Substrate.” *IEEE Transactions on Antennas and Propagation*. Vol. 55, No. 11, Nov. 2007, pp. 3163–3171.

[4] Violates, John *Antenna Engineering Handbook*, 4th Ed., McGraw-Hill.

See Also

`cavity` | `spiralArchimedean` | `vivaldi`

Topics

“Rotate Antenna and Arrays”

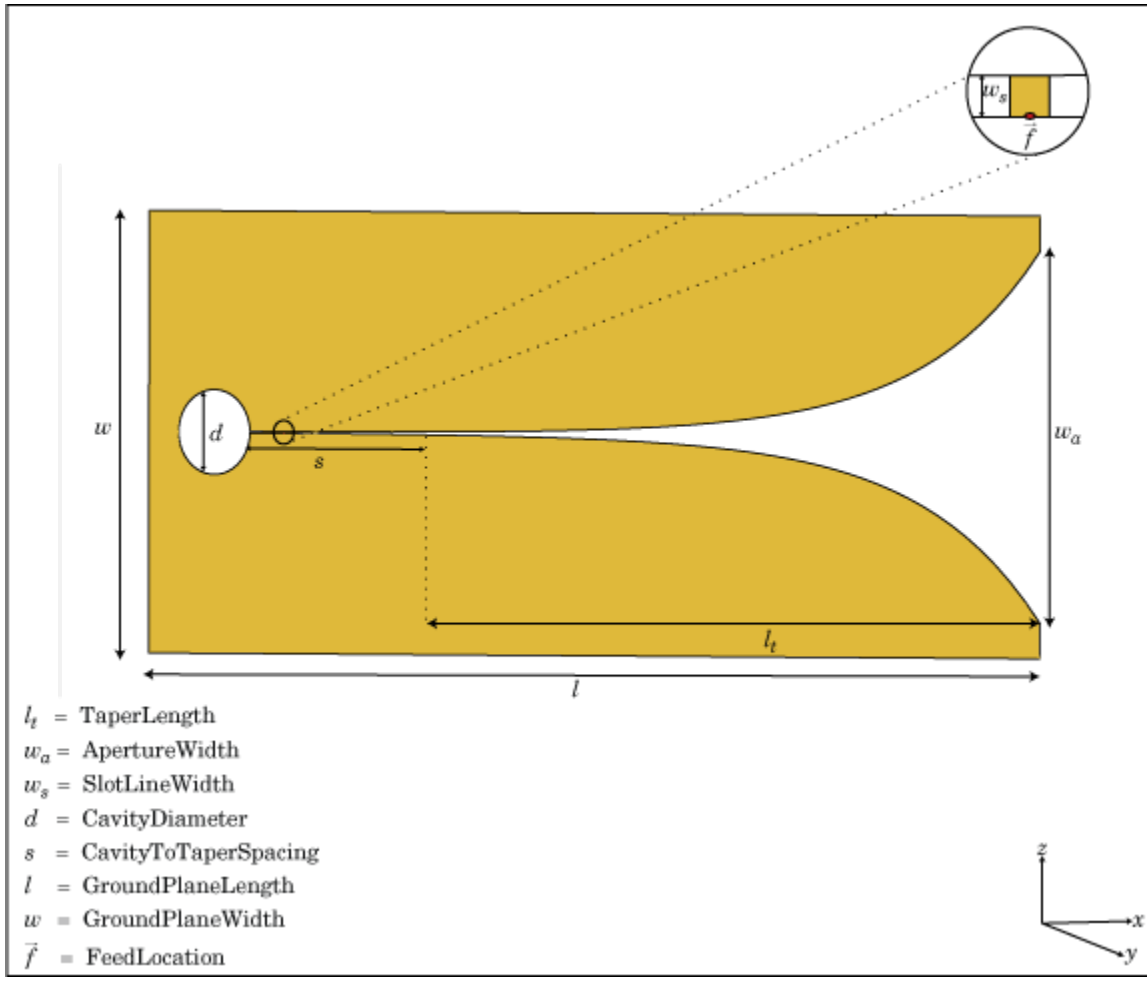
Introduced in R2015a

vivaldi

Create Vivaldi notch antenna on ground plane

Description

The `vivaldi` object is a Vivaldi notch antenna on a ground plane.



Creation

Syntax

```
vi = vivaldi
vi = vivaldi(Name,Value)
```

Description

`vi = vivaldi` creates a Vivaldi notch antenna on a ground plane. By default, the antenna operates at a frequency range of 1–2 GHz and is located in the X-Y plane.

`vi = vivaldi(Name,Value)` creates Vivaldi notch antenna, with additional properties specified by one, or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties you do not specify retains their default values.

Properties

`TaperLength` — Taper length
0.2430 (default) | scalar

Taper length of vivaldi, specified a scalar in meters.

Example: `'TaperLength',2e-3`

`ApertureWidth` — Aperture width
0.1050 (default) | scalar

Aperture width, specified as a scalar in meters.

Example: `'ApertureWidth',3e-3`

`OpeningRate` — Taper opening rate
25 (default) | scalar

Taper opening rate, specified a scalar. This property determines the rate at which the notch transitions from the feedpoint to the aperture. When `OpeningRate` is `0`, the notch has a linear profile and for other values it has an exponential profile.

Example: `'OpeningRate',0.3`

Data Types: `double`

`SlotLineWidth` — Slot line width
5.0000e-04 (default) | scalar

Slot line width, specified as a scalar in meters.

Example: `'SlotLineWidth',3`

Data Types: `double`

`CavityDiameter` — Cavity termination diameter
0.0240 (default) | scalar

Cavity termination diameter, specified a scalar in meters.

Example: `'CavityDiameter',2`

Data Types: `double`

`CavityToTaperSpacing` — Cavity to taper distance of transition
0.0230 (default) | scalar

Cavity to taper distance of transition, specified as a scalar in meters. By default, this property is measured along the x-axis.

Example: `'CavityToTaperSpacing',3`

Data Types: `double`

`GroundPlaneLength` — Ground plane length
0.3000 (default) | scalar

Ground plane length, specified as a scalar in meters. By default, ground plane length is measured along the x-axis.

Example: `'GroundPlaneLength',2`

Data Types: `double`

`GroundPlaneWidth` — Ground plane width
0.1250 (default) | scalar

Ground plane width, specified a scalar in meters. By default, ground plane width is measured along the y-axis.

Example: `'GroundPlaneWidth',4`

Data Types: `double`

`FeedOffset` — Distance from feed along x-axis
0 (default) | scalar

Distance from feed along x-axis, specified a scalar in meters.

Example: `'FeedOffset',3`

Data Types: `double`

Load — Lumped elements
[1x1 `lumpedElement`] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. You can add a load anywhere on the surface of the antenna. By default, the load is at the origin. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `vi.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector in degrees.

Example: `'Tilt',90`

Example: `vi.Tilt = [90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

Create and View Vivaldi Antenna

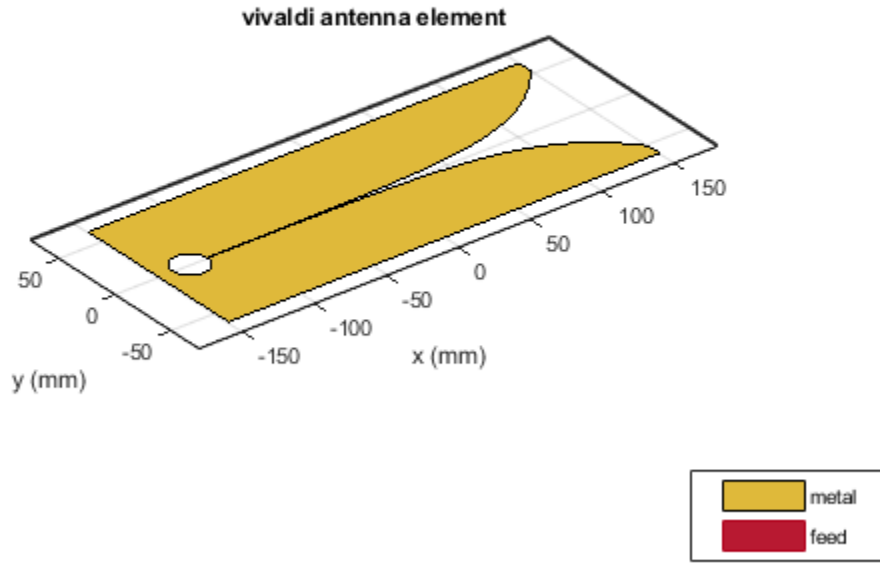
Create and view the default Vivaldi antenna.

```
vi = vivaldi
```

```
vi =
  vivaldi with properties:

             TaperLength: 0.2430
```

```
          ApertureWidth: 0.1050
            OpeningRate: 25
          SlotLineWidth: 5.0000e-04
         CavityDiameter: 0.0240
CavityToTaperSpacing: 0.0230
   GroundPlaneLength: 0.3000
    GroundPlaneWidth: 0.1250
          FeedOffset: -0.1045
                Tilt: 0
            TiltAxis: [1 0 0]
                Load: [1x1 lumpedElement]
```

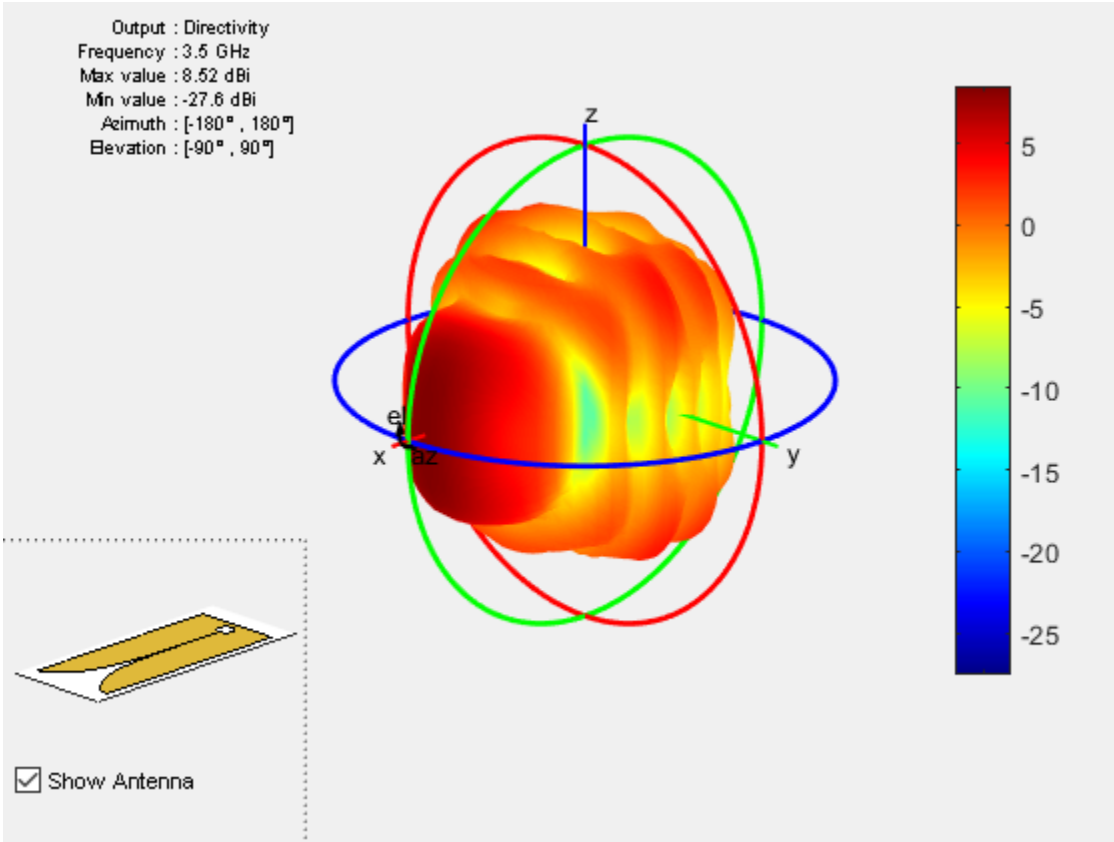
```
show(vi);
```



Radiation Pattern of Vivaldi Antenna

Plot the radiation pattern of a vivaldi antenna for a frequency of 3.5 GHz.

```
vi = vivaldi;
pattern(vi,3.5e9);
```



References

[1] Balanis, C.A. *Antenna Theory. Analysis and Design*, 3rd Ed. New York: Wiley, 2005.

See Also

`slot` | `spiralArchimedean` | `yagiUda`

Topics

“Rotate Antenna and Arrays”

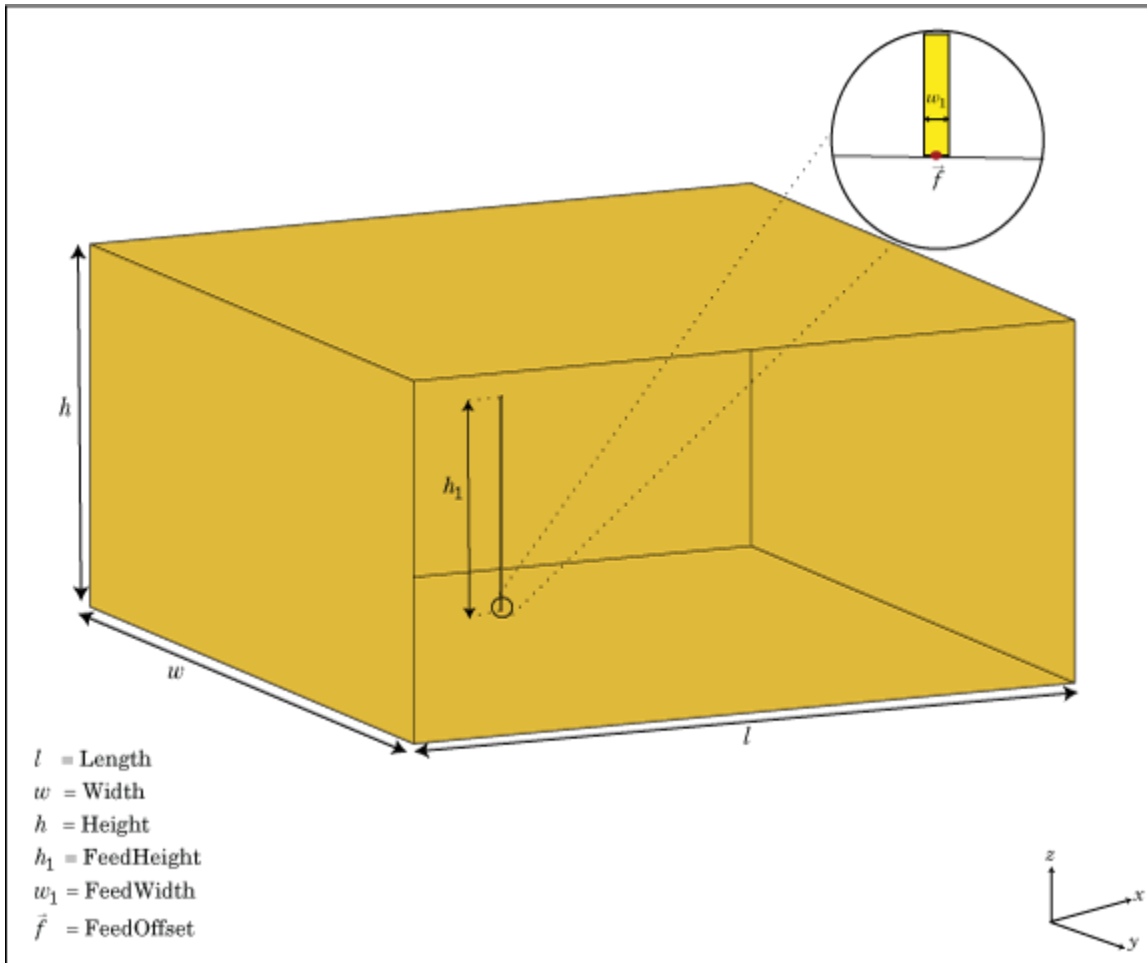
Introduced in R2015a

waveguide

Create rectangular waveguide

Description

The `waveguide` object is an open-ended rectangular waveguide. The default rectangular waveguide is the WR-90 and functions in the X-band. The X-band has a cutoff frequency of 6.5 GHz and ranges from 8.2 GHz to 12.5 GHz.



Creation

Syntax

```
wg = waveguide
wg = waveguide(Name,Value)
```

Description

`wg = waveguide` creates an open-ended rectangular waveguide.

`wg = waveguide(Name,Value)` creates a rectangular waveguide with additional properties specified by one, or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`. Properties not specified retain their default values.

Properties

`FeedHeight` — Height of feed

`0.0060` (default) | scalar

Height of feed, specified as a scalar in meters. By default, the feed height is chosen for an operating frequency of 12.5 GHz.

Example: `'FeedHeight',0.0050`

Data Types: `double`

`FeedWidth` — Width of feed

`6.0000e-05` (default) | scalar

Width of feed, specified as a scalar in meters.

Example: `'FeedWidth',5e-05`

Data Types: `double`

`Length` — Rectangular waveguide length

`0.0240` (default) | scalar in meters

Rectangular waveguide length, specified as a scalar in meters. By default, the waveguide length is 1λ, where:

$$\lambda = c / f$$

- `c` = speed of light, `299792458` m/s
- `f` = operating frequency of the waveguide

Example: `'Length',0.09`

Data Types: `double`

`Width` — Rectangular waveguide width
`0.0229` (default) | scalar in meters

Rectangular waveguide width, specified as a scalar in meters.

Example: `'Width',0.05`

Data Types: `double`

`Height` — Rectangular waveguide height
`0.0102` (default) | scalar

Rectangular waveguide height, specified as a scalar in meters.

Example: `'Height',0.0200`

Data Types: `double`

`FeedOffset` — Signed distance of feedpoint from center of ground plane
[–0.0060 0] (default) | two-element vector

Signed distance of feedpoint from center of ground plane, specified as a two-element vector in meters. By default, the feed is at an offset of λ/4 from the shortened end on the X-Y plane.

Example: `'FeedOffset',[–0.0070 0.01]`

Data Types: `double`

`Load` — Lumped elements
[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `wg.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays

impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

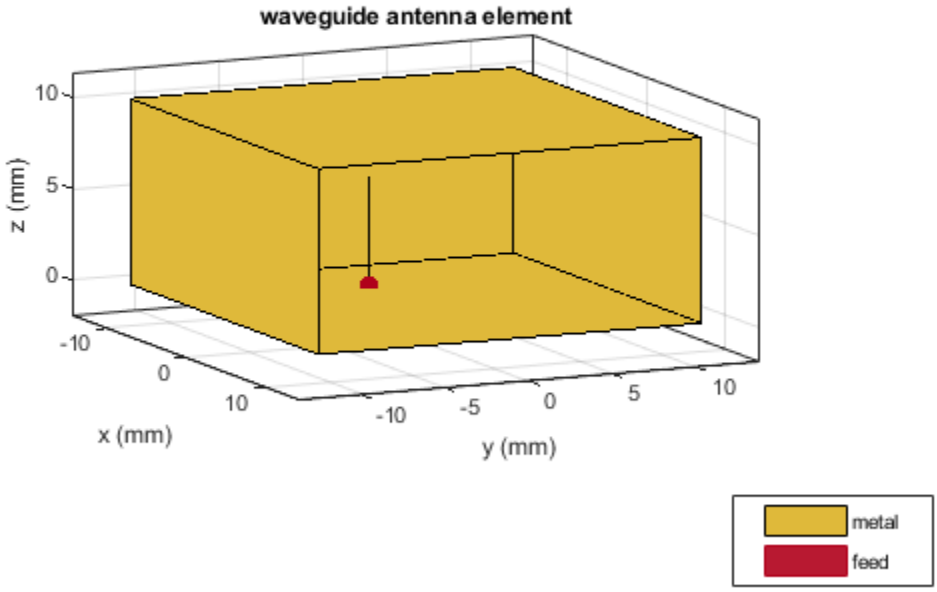
Default Rectangular Waveguide

Create a rectangular waveguide using default dimensions. Display the waveguide.

```
wg = waveguide
```

```
wg =
  waveguide with properties:

        Length: 0.0240
         Width: 0.0229
        Height: 0.0102
     FeedWidth: 6.0000e-05
    FeedHeight: 0.0060
    FeedOffset: [-0.0060 0]
          Tilt: 0
      TiltAxis: [1 0 0]
          Load: [1x1 lumpedElement]
```

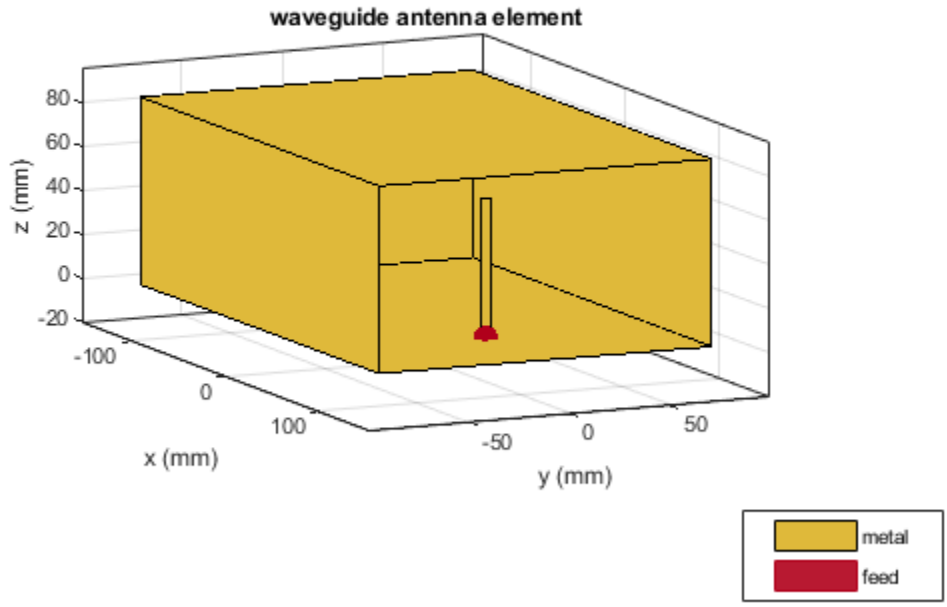
```
show(wg)
```



Radiation Pattern of WR-650 Rectangular Waveguide

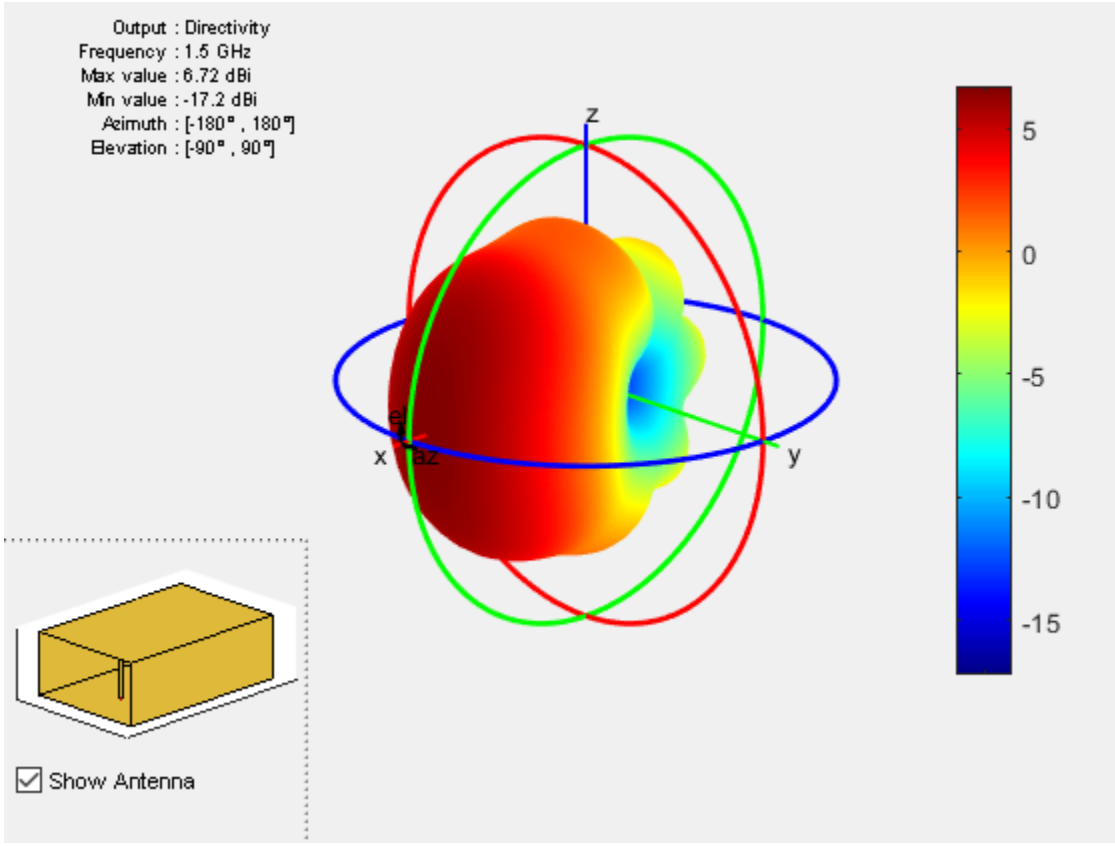
Create a WR-650 rectangular waveguide and display it.

```
wg = waveguide('Length',0.254,'Width',0.1651,'Height',0.0855,...
    'FeedHeight',0.0635,'FeedWidth',0.00508,'FeedOffset',[0.0635 0]);
show(wg)
```



Plot the radiation pattern of this waveguide at 1.5 GHz.

```
figure
pattern(wg,1.5e9)
```



References

[1] Balanis, Constantine A.*Antenna Theory. Analysis and Design*. 3rd Ed. New York: John Wiley and Sons, 2005.

See Also

`horn`

Topics

“Rotate Antenna and Arrays”

Introduced in R2016a

yagiUda

Create Yagi-Uda array antenna

Description

The `yagiUda` class creates a classic Yagi-Uda array comprised of an exciter, reflector, and *N*- directors along the z-axis. The reflector and directors create a traveling wave structure that results in a directional radiation pattern.

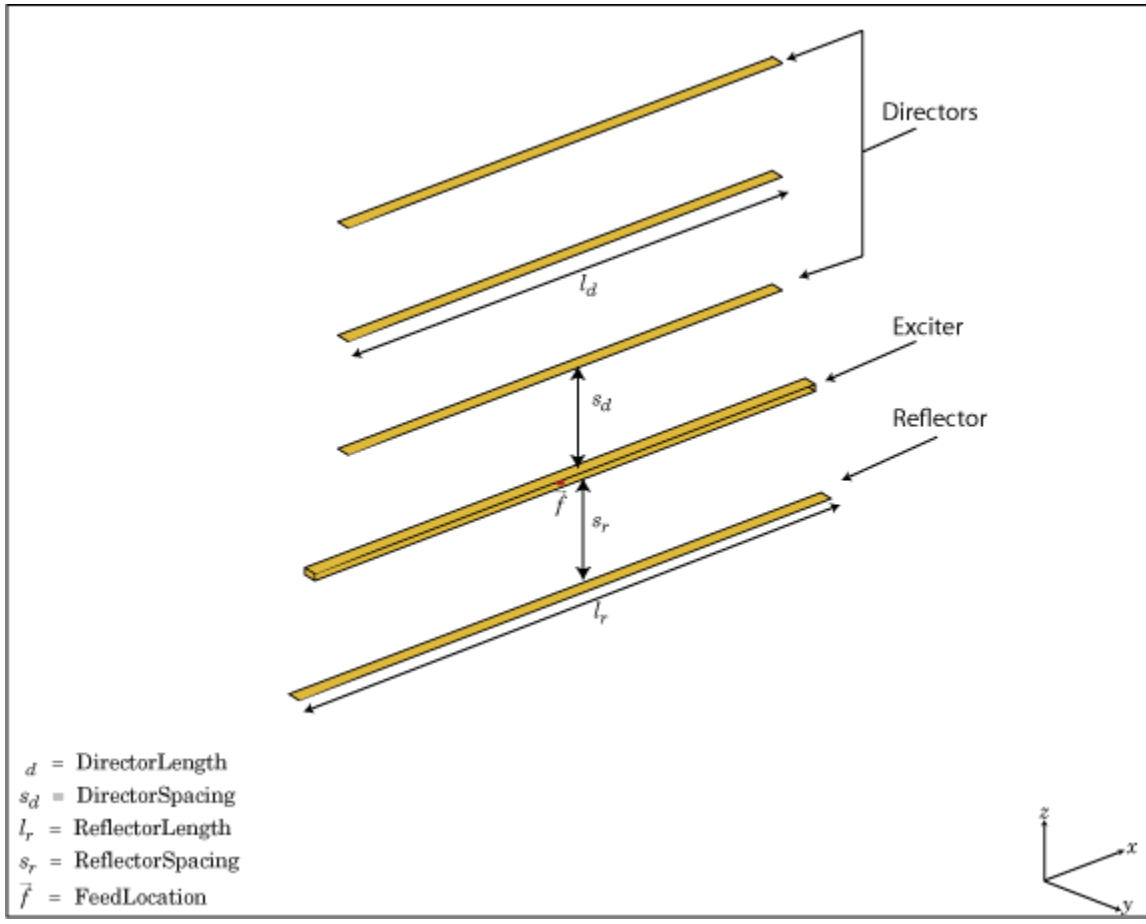
The exciter, reflector, and directors have equal widths and are related to the diameter of an equivalent cylindrical structure by the equation

$$w = 2d = 4r$$

where:

- d is the diameter of equivalent cylinder
- r is the radius of equivalent cylinder

For a given cylinder radius, use the `cylinder2strip` utility function to calculate the equivalent width. A typical Yagi-Uda antenna array uses folded dipole as an exciter, due to its high impedance. The Yagi-Uda is center-fed and the feed point coincides with the origin. In place of a folded dipole, you can also use a planar dipole as an exciter.



Creation

Syntax

```
yu = yagiUda
yu = yagiUda(Name,Value)
```

Description

`yu = yagiUda` creates a half-wavelength Yagi-Uda array antenna along the Z-axis. The default Yagi-Uda uses folded dipole as three directors, one reflector, and a folded dipole as an exciter. By default, the dimensions are chosen for an operating frequency of 300 MHz.

`yu = yagiUda(Name,Value)` creates a half-wavelength Yagi-Uda array antenna, with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain default values.

Properties

`Exciter` — Antenna type used as exciter

dipoleFolded (default) | object

Antenna Type used as exciter, specified as the comma-separated pair consisting of `'Exciter'` and an object.

Example: `'Exciter',dipole`

`NumDirectors` — Total number of director elements

3 (default) | scalar

Total number of director elements, specified as a scalar.

Note Number of director elements should be less than or equal to 20.

Example: `'NumDirectors',13`

Data Types: `double`

`DirectorLength` — Director length

0.4080 (default) | scalar | vector

Director length, specified as a scalar or vector in meters.

Example: `'DirectorLength',[0.4 0.5]`

Data Types: `double`

`DirectorSpacing` — Spacing between directors
0.3400 (default) | scalar | vector

Spacing between directors, specified as a scalar or vector in meters.

Example: `'DirectorSpacing',[0.4 0.5]`

Data Types: `double`

`ReflectorLength` — Reflector length
0.5000 (default) | scalar

Reflector length, specified as a scalar in meters.

Example: `'ReflectorLength',0.3`

Data Types: `double`

`ReflectorSpacing` — Spacing between exciter and reflector
0.2500 (default) | scalar

Spacing between exciter and reflector, specified as a scalar in meters.

Example: `'ReflectorSpacing', 0.4`

Data Types: `double`

`Load` — Lumped elements
[1x1 LumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `yu.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure

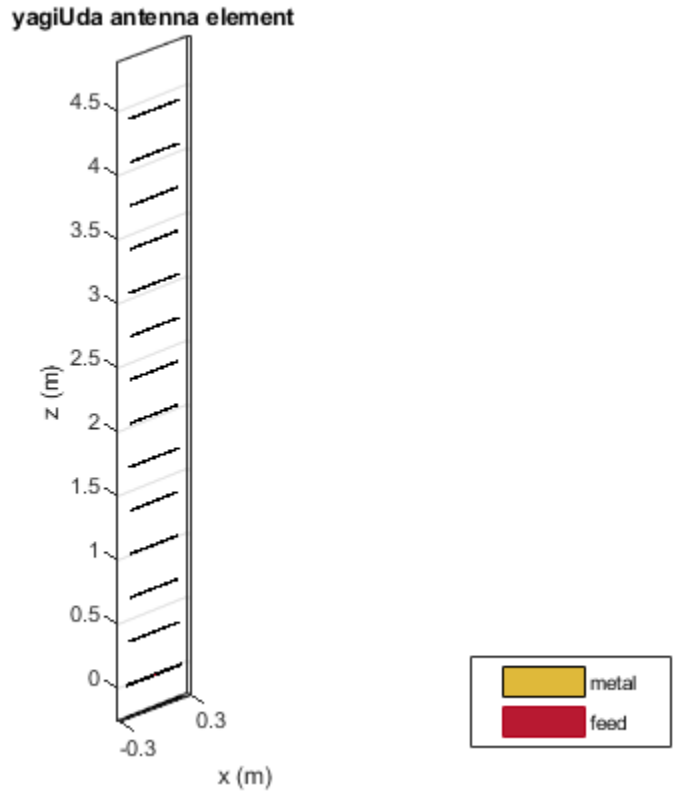
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

Create and View Yagi-Uda Array Antenna

Create and view a Yagi-Uda array antenna with 13 directors.

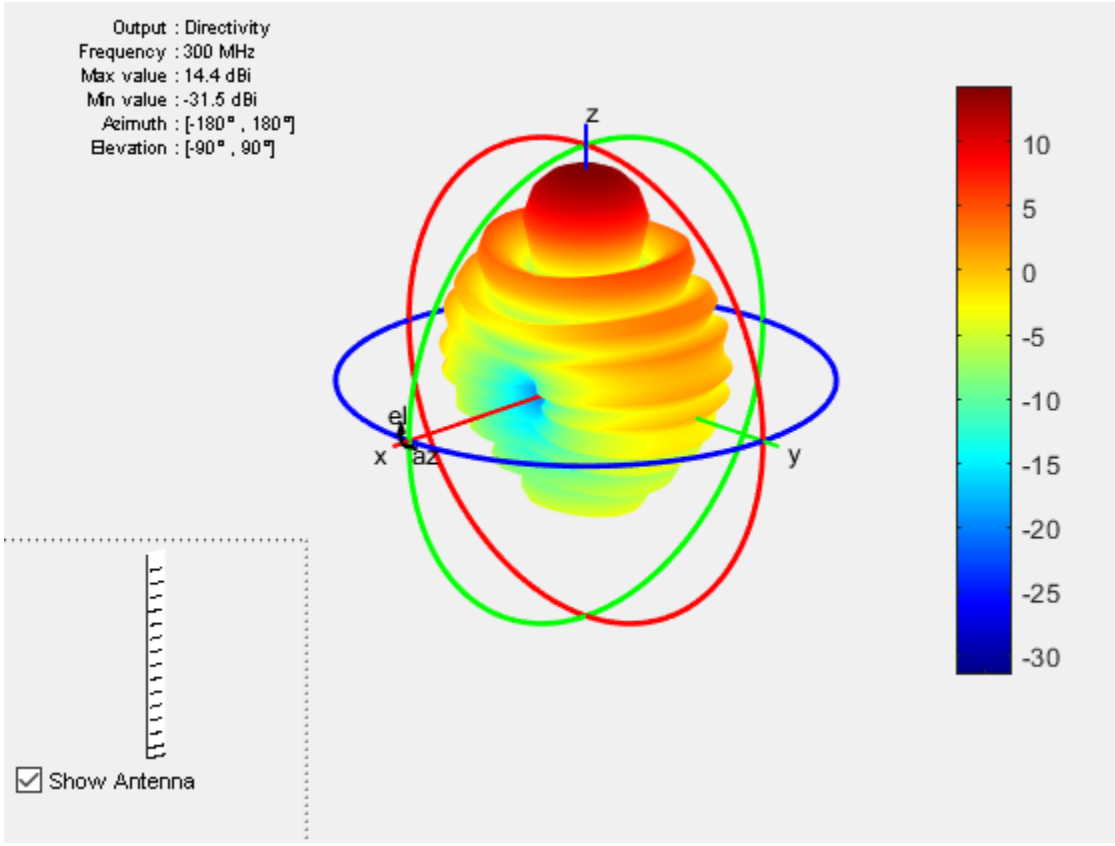
```
y = yagiUda('NumDirectors',13);
show(y)
```



Radiation Pattern of Yagi-Uda Array Antenna

Plot the radiation pattern of a Yagi-Uda array antenna at a frequency of 30 0MHz.

```
y = yagiUda('NumDirectors',13);
pattern(y,300e6)
```



Calculate Cylinder to Strip Approximation

Calculate the width of the strip approximation to a cylinder of radius 20 mm.

```
w = cylinder2strip(20e-3)
```

```
w = 0.0800
```

References

[1] Balanis, C.A. *Antenna Theory. Analysis and Design*, 3rd Ed. New York: Wiley, 2005.

See Also

`cylinder2strip` | `dipole` | `dipoleFolded` | `slot`

Topics

“Rotate Antenna and Arrays”

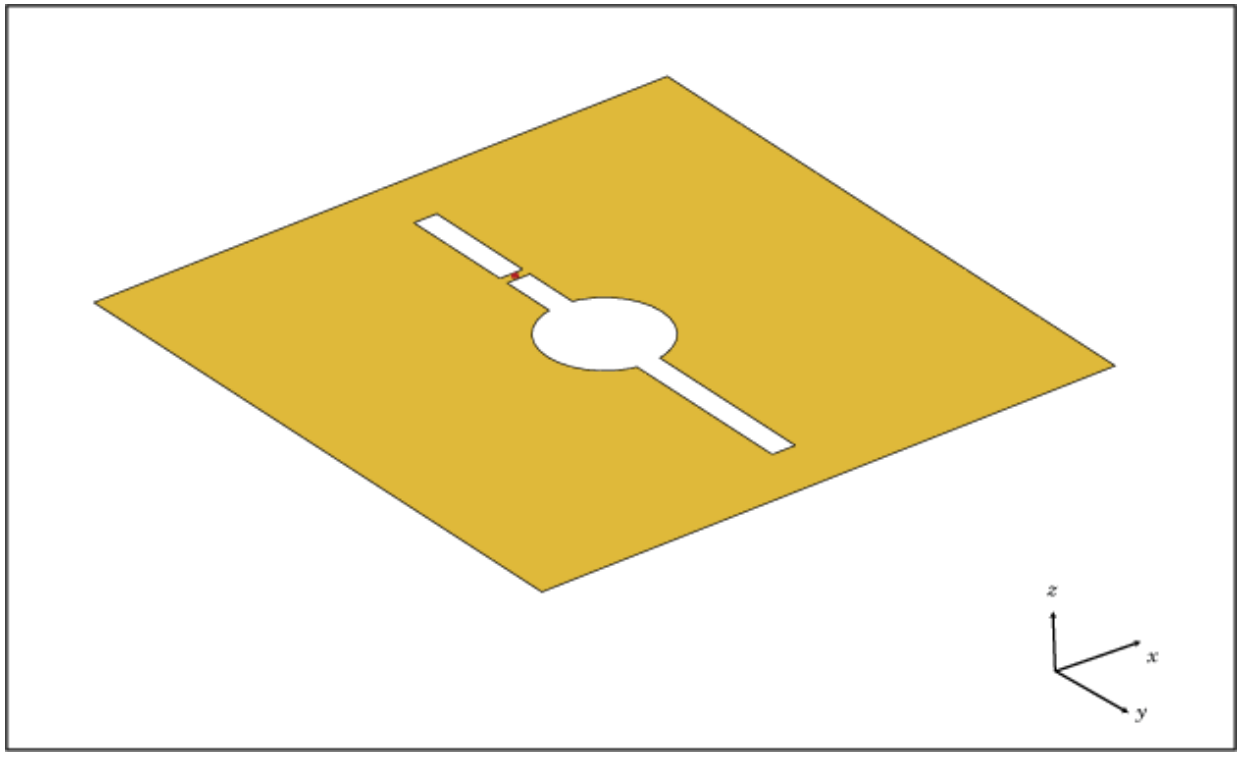
Introduced in R2015a

customAntennaGeometry

Create antenna represented by 2-D custom geometry

Description

The `customAntennaGeometry` object is an antenna represented by a 2-D custom geometry on the X-Y plane. Using `customAntennaGeometry`, you can import a planar mesh, define the feed for this mesh to create an antenna, analyze the antenna, and use it in finite or infinite arrays. The image shown is a custom slot antenna.



Creation

Syntax

```
ca = customAntennaGeometry
ca = customAntennaGeometry(Name,Value)
```

Description

`ca = customAntennaGeometry` creates a 2-D antenna represented by a custom geometry, based on the specified boundary.

`ca = customAntennaGeometry(Name,Value)` creates a 2-D planar antenna geometry, with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Boundary` — Boundary information in Cartesian coordinates
cell array

Boundary information in Cartesian coordinates, specified as a cell array in meters.

Data Types: `double`

`Operation` — Boolean operation performed on boundary list
`'P1'` (default) | character vector

Boolean operation performed on the boundary list, specified as a character vector.

Example: `'Operation','P1-P2'`

Data Types: `double`

`FeedLocation` — Antenna feed location in Cartesian coordinates
`[0 0 0]` (default) | three-element vector

Antenna feed location in Cartesian coordinates, specified as a three-element vector. The three-element vector is the X, Y, and Z coordinates respectively.

Example: `'FeedLocation', [0 0.2 0]`

Data Types: `double`

`FeedWidth` — Width of feed section
`0.0100` (default) | scalar

Width of feed section, specified as a scalar in meters.

Example: `'FeedWidth',0.05`

Data Types: `double`

`Load` — Lumped elements
[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load'`, lumpedelement. `lumpedelement` is the object handle for the load created using `lumpedElement`.

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.

- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
show	Display antenna or array structure; Display shape as filled patch
vswr	Voltage standing wave ratio of antenna

Examples

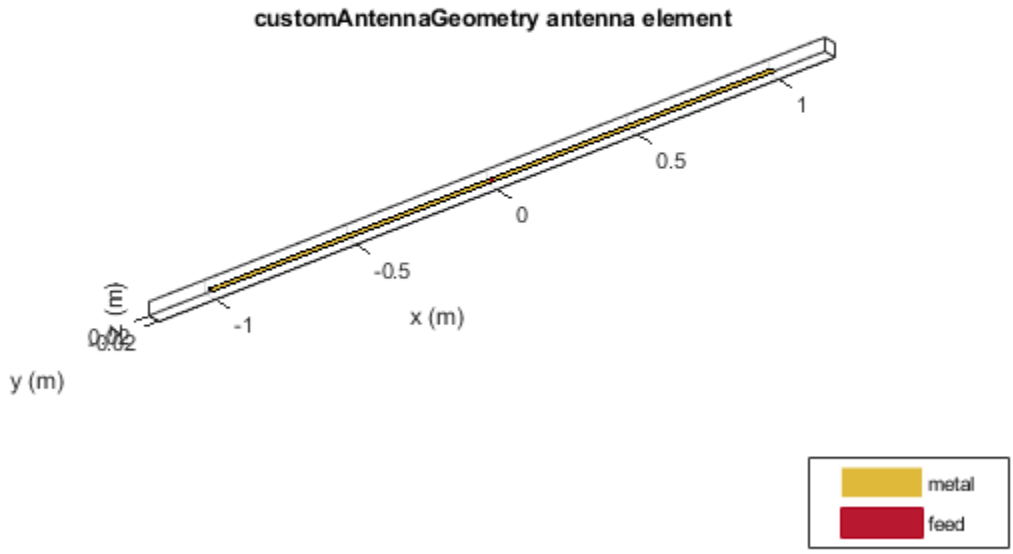
Custom Dipole Antenna

Create a custom dipole antenna and view it.

```
ca = customAntennaGeometry
```

```
ca =
  customAntennaGeometry with properties:

        Boundary: {[4x3 double]}
       Operation: 'P1'
    FeedLocation: [0 0 0]
       FeedWidth: 0.0200
            Tilt: 0
         TiltAxis: [1 0 0]
            Load: [1x1 lumpedElement]
```

```
show(ca)
```



Custom Slot Antenna

Create a custom slot antenna using three rectangles and a circle.

Make three rectangles of 0.5 m x 0.5 m, 0.02 m x 0.4 m and 0.03 m x 0.008 m.

```
pr = em.internal.makerectangle(0.5,0.5);
pr1 = em.internal.makerectangle(0.02,0.4);
pr2 = em.internal.makerectangle(0.03,0.008);
```

Make a circle of radius 0.05 m.

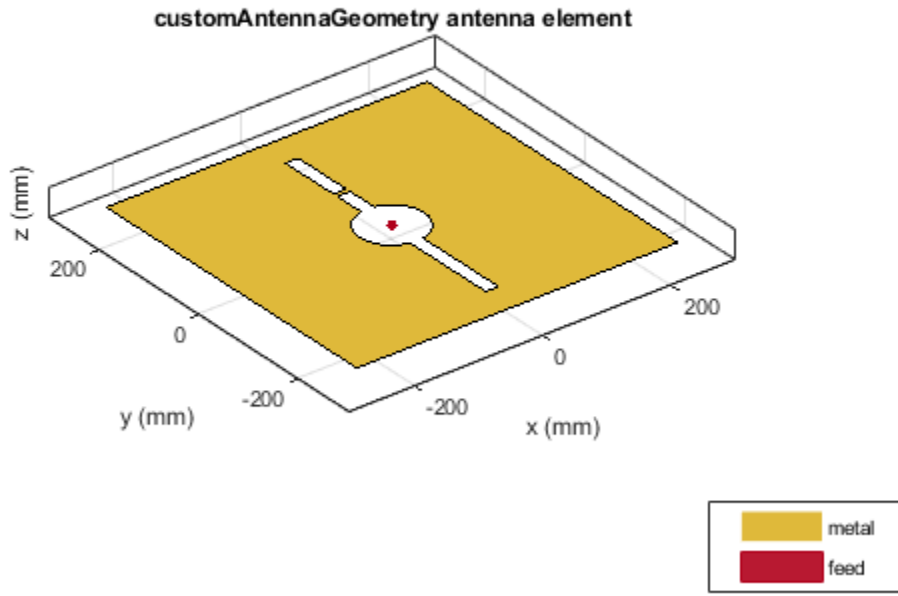
```
ph = em.internal.makecircle(0.05);
```

Translate the third rectangle to the X-Y plane using the coordinates [0 0.1 0].

```
pf = em.internal.translateshape(pr2,[0 0.1 0]);
```

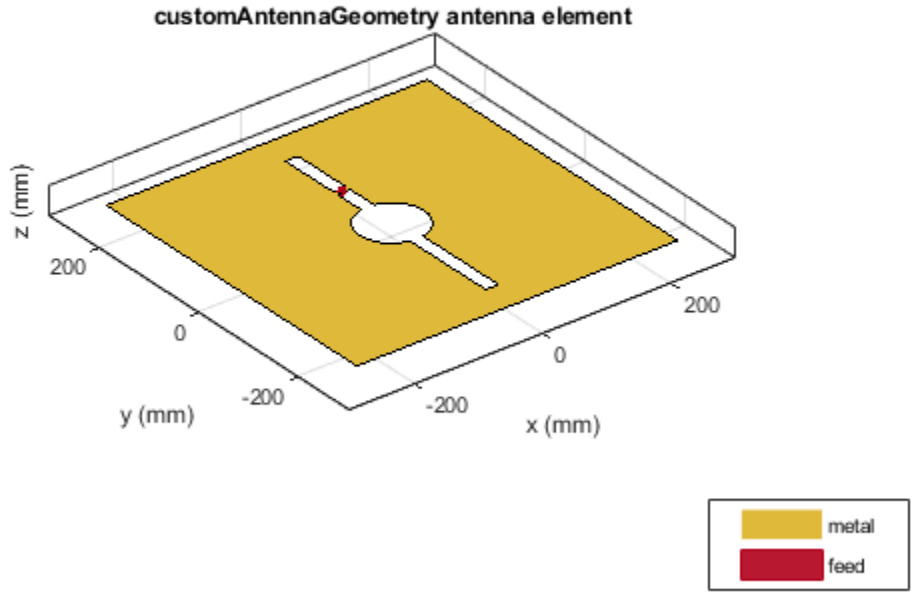
Create a custom slot antenna element using the specified boundary shapes. Transpose pr, ph, pr1, and pf to make sure the boundary inputs are column vector arrays.

```
c = customAntennaGeometry('Boundary',{pr',ph',pr1',pf'},...
    'Operation','P1-P2-P3+P4');
figure;
show(c);
```



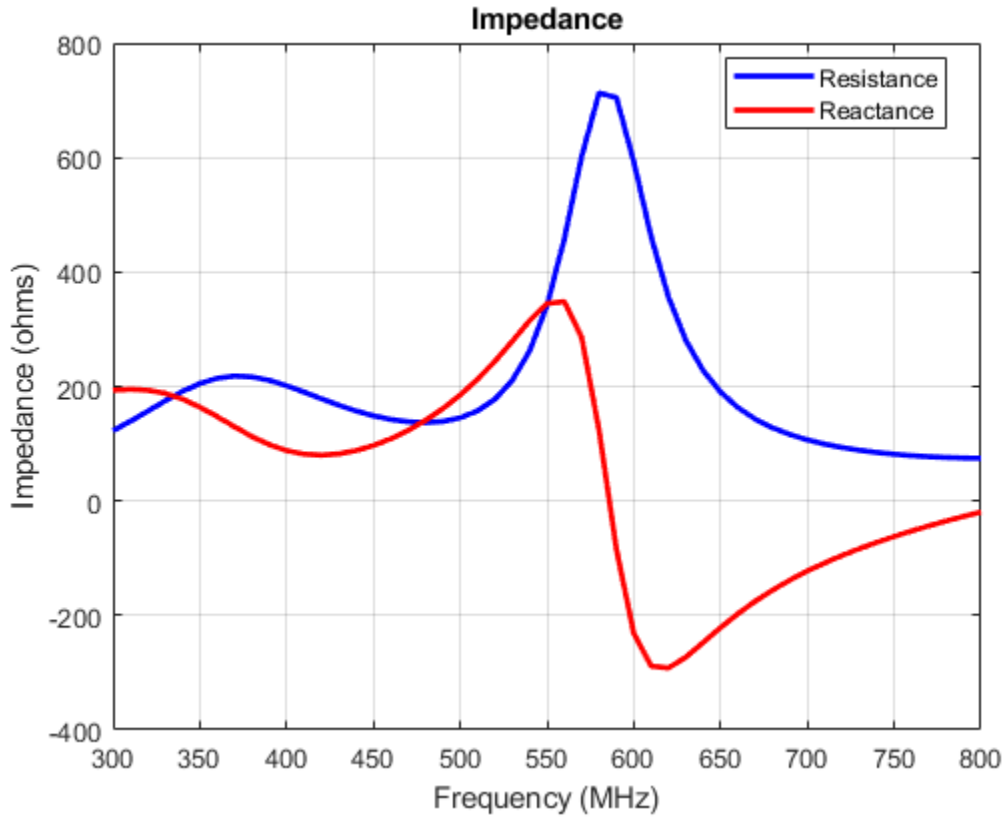
Move the feed location to new coordinates.

```
c.FeedLocation = [0,0.1,0];
figure;
show(c);
```



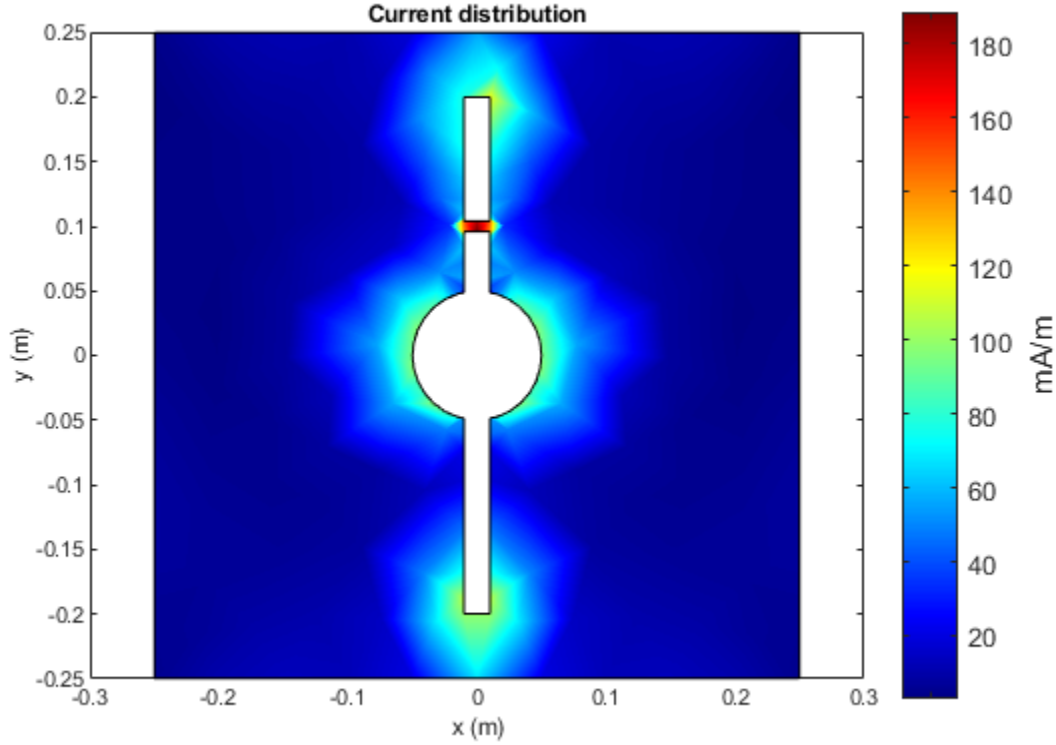
Analyze the impedance of the antenna from 300 MHz to 800 MHz.

```
figure;
impedance(c, linspace(300e6,800e6,51));
```



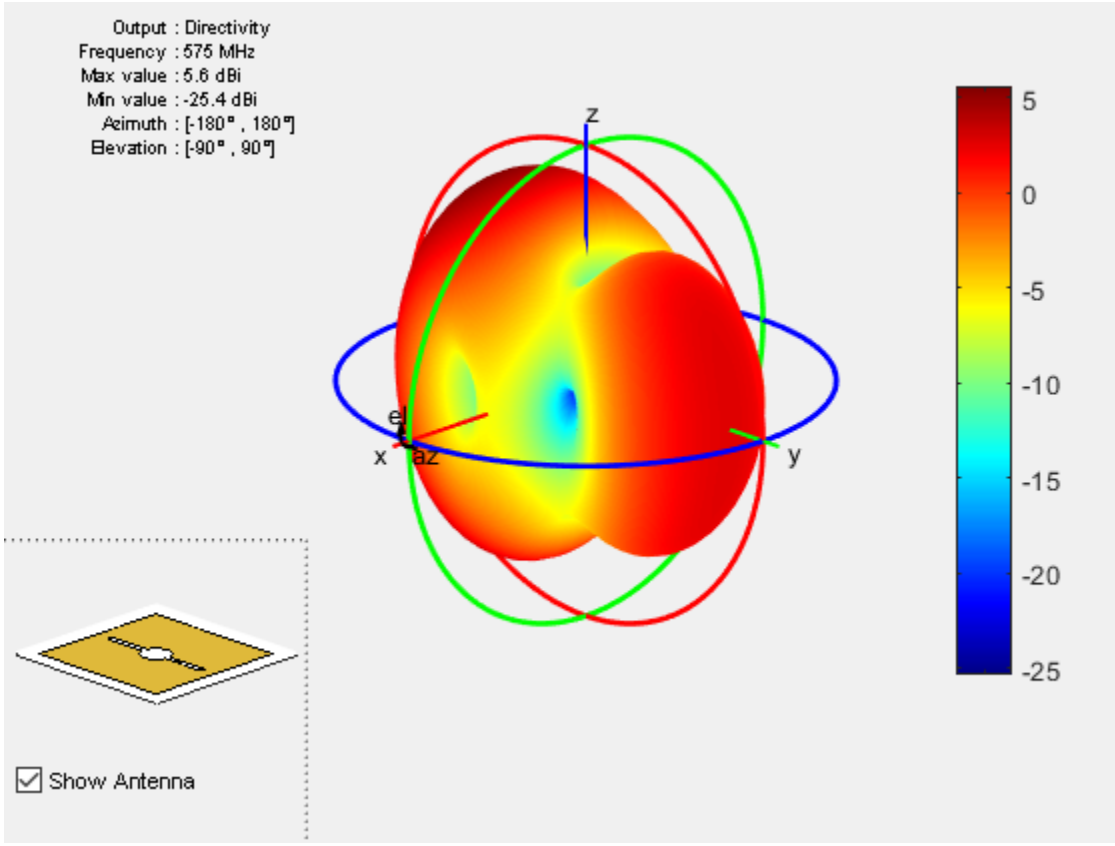
Analyze the current distribution of the antenna at 575 MHz.

```
figure;
current(c,575e6)
```



Plot the radiation pattern of the antenna at 575 MHz.

```
figure;
pattern(c,575e6)
```



References

[1] Balanis, C. A. *Antenna Theory. Analysis and Design*. 3rd Ed. Hoboken, NJ: John Wiley & Sons, 2005.

See Also

Topics

“Rotate Antenna and Arrays”

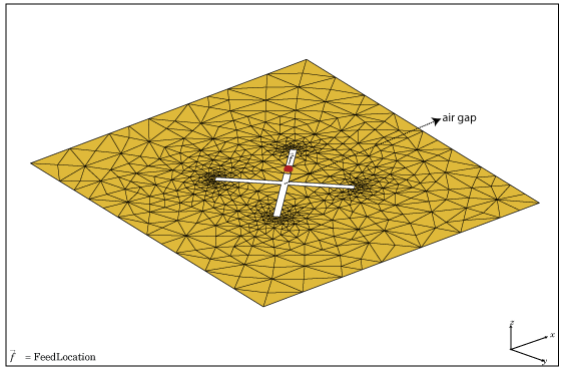
Introduced in R2016b

customAntennaMesh

Create 2-D custom mesh antenna on X-Y plane

Description

The `customAntennaMesh` object creates an antenna represented by a 2-D custom mesh on the X-Y plane. You can provide an arbitrary antenna mesh to the Antenna Toolbox and analyze this mesh as a custom antenna for port and field characteristics.



Creation

Description

`customantenna = customAntennaMesh(points,triangles)` creates a 2-D antenna represented by a custom mesh, based on the specified points and triangles.

Input Arguments

`points` — Points in custom mesh

`2-by-N` or `3-by-N` integer matrix of Cartesian coordinates in meters

Points in a custom mesh, specified as a `2-by-N` or `3-by-N` integer matrix of Cartesian coordinates in meters. *N* is the number of points. In case you specify a `3`×*N* integer

matrix, the Z-coordinate must be zero or a constant value. This value sets the `'Points'` property in the custom antenna mesh.

Example: `[0 1 0 1;0 1 1 0]`

Data Types: `double`

`triangles` — Triangles in mesh

`4-by-M` integer matrix

Triangles in the mesh, specified as a `4-by-M` integer matrix. *M* is the number of triangles. The first three rows are indices to the points matrix and represent the vertices of each triangle. The fourth row is a domain number useful for identifying separate parts of an antenna. This value sets the `'Triangles'` property in the custom antenna mesh.

Data Types: `double`

Properties

`Points'` — Points in custom mesh

`2-by-N` or `3-by-N` integer matrix of Cartesian coordinates

Points in a custom mesh, specified as a `2-by-N` or `3-by-N` integer matrix of Cartesian coordinates in meters. *N* is the number of points.

Example: `[0.1 0.2 0]`

Data Types: `double`

`Triangles` — Triangles in mesh

`4-by-M` integer matrix

Triangles in the mesh, specified as a `4-by-M` integer matrix. *M* is the number of triangles.

Data Types: `double`

`Tilt` — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
createFeed	Create feed location for custom antenna
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array

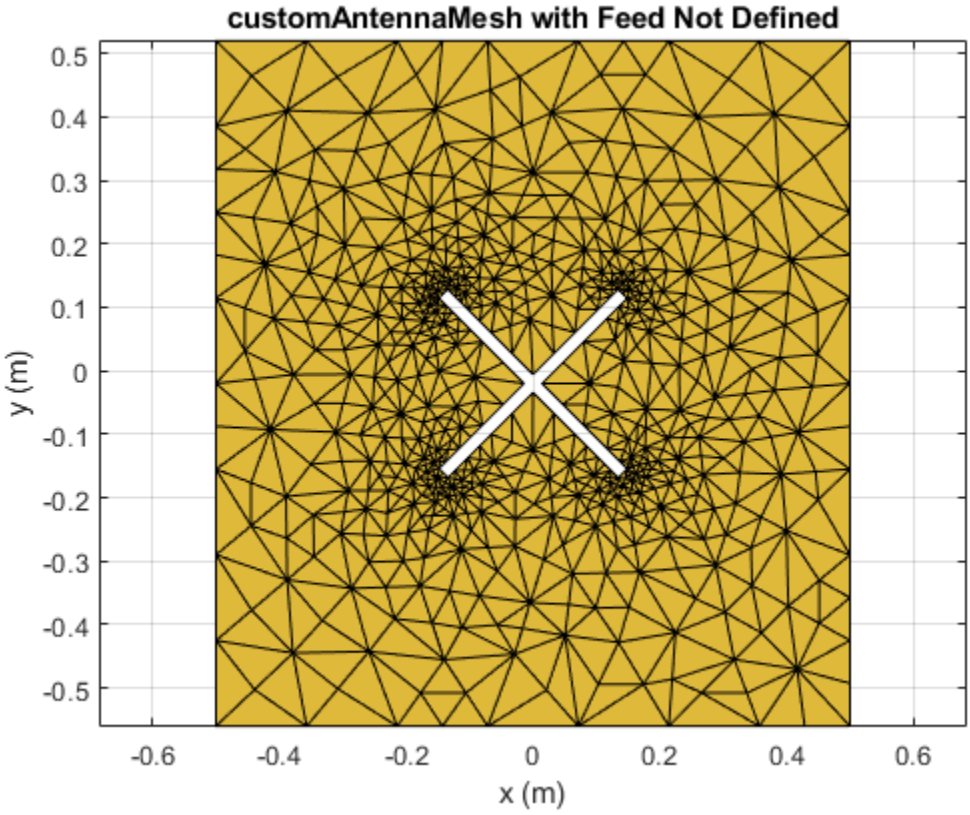
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

Custom Mesh Antenna

Load a custom planar mesh. Create the antenna and antenna feed. View the custom planar mesh antenna and calculate the impedance at 100 MHz.

```
load planarmesh.mat;
c = customAntennaMesh(p,t);
show(c)
```




```
createFeed(c,[0.07,0.01],[0.05,0.05]);
Z = impedance(c,100e6)
```

```
Z = 0.5091 + 57.2103i
```

References

[1] Balanis, C.A. *Antenna Theory: Analysis and Design*. 3rd Ed. New York: Wiley, 2005.

See Also

`cavity` | `reflector`

Topics

“Rotate Antenna and Arrays”

Introduced in R2015b

pcbStack

Single-feed or multi-feed PCB antenna

Description

The `pcbStack` object is a single-feed or multi-feed printed circuit board (PCB) antenna. Using the PCB stack, you can create antennas using single-layer or multilayer metal or metal-dielectric substrates. You can also use the PCB stack to create antennas with an arbitrary number of feeds and vias. You can also use Antenna Toolbox catalog antennas to create a PCB antenna.

Creation

Syntax

```
pcbant = pcbStack
pcbant = pcbStack(Name,Value)
pcbant = pcbStack(ant)
```

Description

`pcbant = pcbStack` creates an air-filled single-feed PCB with two metal layers.

`pcbant = pcbStack(Name,Value)` creates a PCB antenna, with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

`pcbant = pcbStack(ant)` converts any 2-D or 2.5D antenna from the antenna catalog into a PCB antenna for further modeling and analysis.

Properties

`Name` — Name of PCB antenna
`'MyPCB'` (default) | character vector

Name of PCB antenna, specified a character vector.

Example: `'Name','PCBPatch'`

Data Types: `char`

`Revision` — Revision details of PCB antenna design
`'1.0'` (default) | character vector

Revision details of PCB antenna design, specified as a character vector.

Example: `'Revision','2.0'`

Data Types: `char` | `string`

`BoardShape` — Shape of PC board
`antenna.Rectangle` (default) | object

Shape of PC board, specified as an object. Rectangle is the only supported board shape.

Example: `'BoardShape',antenna.Rectangle`

`BoardThickness` — Thickness of PC board
`0.0100` (default) | positive scalar

Thickness of PC board, specified as a positive scalar.

Example: `'BoardThickness',0.02000`

Data Types: `double`

`Layers` — Metal and dielectric layers
`{[1×1 antenna.Rectangle] [1×1 antenna.Rectangle]}` (default) | cell array of metal layer shapes and dielectric

Metal and dielectric layers, specified a cell array of metal layer shapes and dielectric. You can specify one metal shape or one dielectric per layer starting with the top layer and proceeding down.

Data Types: `double`

`FeedLocations` — Feed locations for antenna in Cartesian coordinates
`[-0.0187 0 1 2]` (default) | *N* -by-3 or *N*-by-4 array

Feed locations for PCB antenna in Cartesian coordinates, specified as *N* -by-3 or *N*-by-4 array. The arrays translate to the following:

- *N* -by-3 – [*x*, *y*, *Layer*]
- *N*-by-4 – [*x*, *y*, *SigLayer*, *GndLayer*]

Example: `'FeedLocations',[-0.0187 0 1 2]`

Data Types: `double`

`FeedDiameter` — Center pin diameter of feed connector
`1.0000e-03` (default) | positive scalar in meters

Center pin diameter of feed connector, specified as a positive scalar in meters.

Example: `'FeedDiameter',2.000e-04`

Data Types: `double`

`ViaLocations` — Electrical short locations for antenna in Cartesian coordinates
`[0 0 0]` (default) | real vector of size*M*-by-4 array

Electrical short locations for antenna in Cartesian coordinates, specified as a real vector of size *M*-by-4 array. The arrays translate to the following:

- *M*-by-4 – [*x*, *y*, *SigLayer*, *GndLayer*]

Example: `'ViaLocations',[0 -0.025 1 2]`

Data Types: `double`

`ViaDiameter` — Electrical shorting pin diameter between metal layers
positive scalar in meters

Electrical shorting pin diameter between metal layers, specified a positive scalar in meters.

Example: `'ViaDiameter',1.0e-3`

Data Types: `double`

`FeedVoltage` — Magnitude voltage applied at the feeds
`1` (default) | positive scalar in volts

Magnitude voltage applied at the feeds, specified as a positive scalar in volts.

Example: `'FeedVoltage',2`

Data Types: `double`

`FeedViaModel` — Model for approximating feed and via
`'strip'` (default) | `'square'` | `'hexagon'` | `'octagon'`

Model for approximating feed and via, specified as one of the following:

- `'strip'` – A rectangular strip approximation to the feed or via cylinder. This approximation is the simplest and results in a small mesh.
- `'square'` – A 4-sided polyhedron approximation to the feed or via cylinder.
- `'hexagon'` – A 6-sided polyhedron approximation to the feed or via cylinder.
- `'octagon'` – A 8-sided polyhedron approximation to the feed or via cylinder.

Example: `'FeedViaModel','octagon'`

Data Types: `double`

`FeedPhase` — Excitation phase at each feed
`0` (default) | real vector in degrees

Excitation phase at each feed, specified as a real vector in degrees.

Example: `'FeedPhase',2`

Data Types: `double`

`Load` — Lumped elements
[1x1 LumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `pcbant.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array

mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
show	Display antenna or array structure; Display shape as filled patch
vswr	Voltage standing wave ratio of antenna

Examples

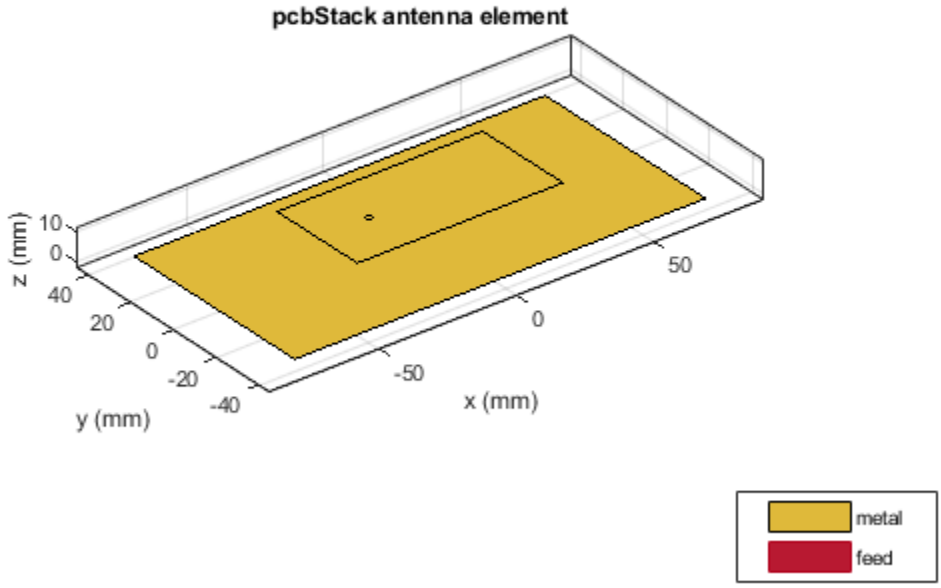
Default PCB Antenna

Create and view a default PCB antenna.

```
p = pcbStack
```

```
p =
  pcbStack with properties:

             Name: 'MyPCB'
         Revision: 'v1.0'
       BoardShape: [1x1 antenna.Rectangle]
    BoardThickness: 0.0100
           Layers: {[1x1 antenna.Rectangle]  [1x1 antenna.Rectangle]}
     FeedLocations: [-0.0187 0 1 2]
      FeedDiameter: 1.0000e-03
      ViaLocations: []
       ViaDiameter: []
      FeedViaModel: 'strip'
       FeedVoltage: 1
         FeedPhase: 0
              Tilt: 0
          TiltAxis: [1 0 0]
              Load: [1x1 lumpedElement]
```

```
show(p);
```



End Loaded Planar Dipole

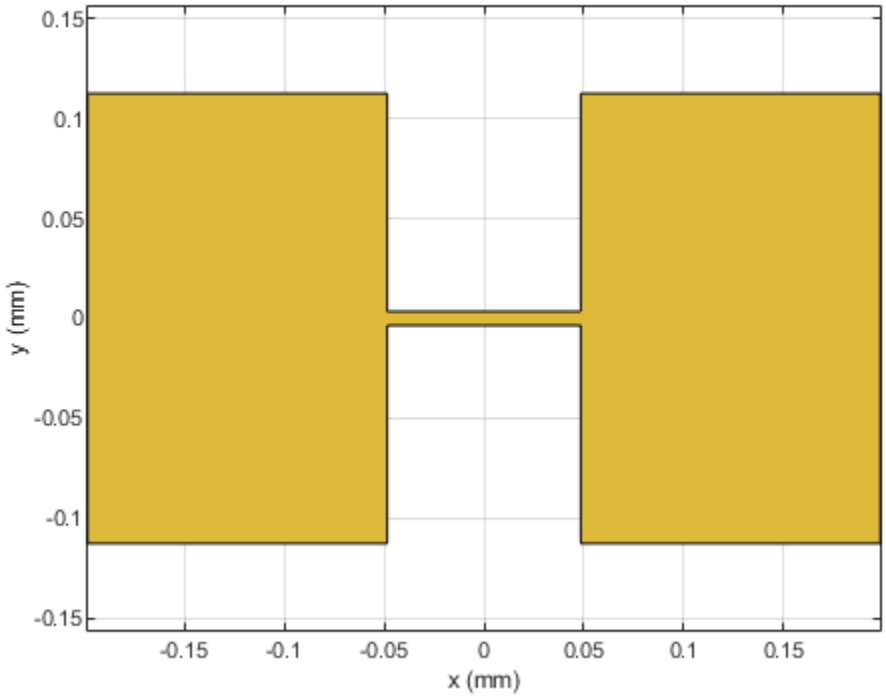
Setup parameters.

```
vp  = physconst('lightspeed');
f   = 850e6;
lambda = vp./f;
```

Build a planar dipole with capacitive loading at the ends.

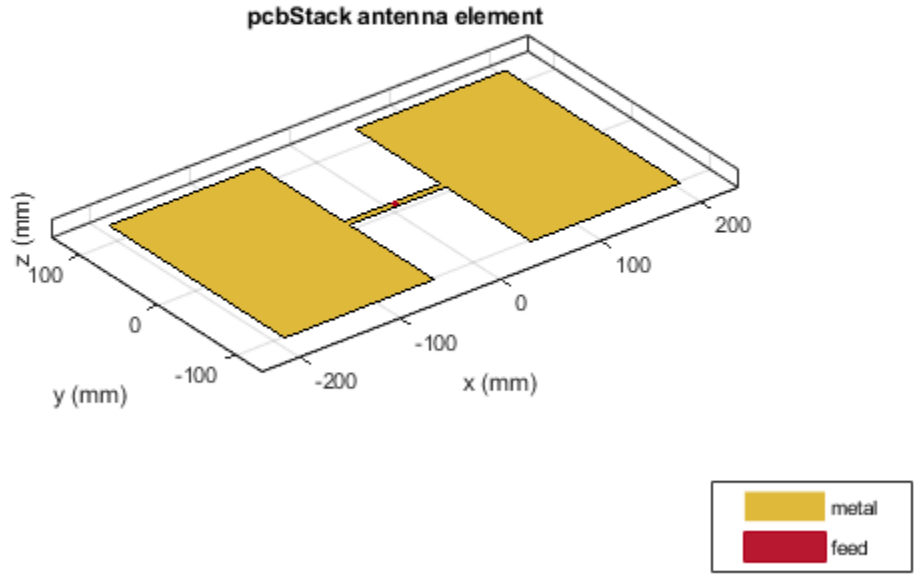
```
L = 0.15;
W = 1.5*L;
```

```
stripL = L;
gapx = .015;
gapy = .01;
r1 = antenna.Rectangle('Center',[0,0],'Length',L,'Width',W,'Center',[lambda*0.35,0]);
r2 = antenna.Rectangle('Center',[0,0],'Length',L,'Width',W,'Center',[-lambda*0.35,0]);
r3 = antenna.Rectangle('Length',0.5*lambda,'Width',0.02*lambda,'NumPoints',2);
s = r1 + r2 + r3;
figure
show(s)
```



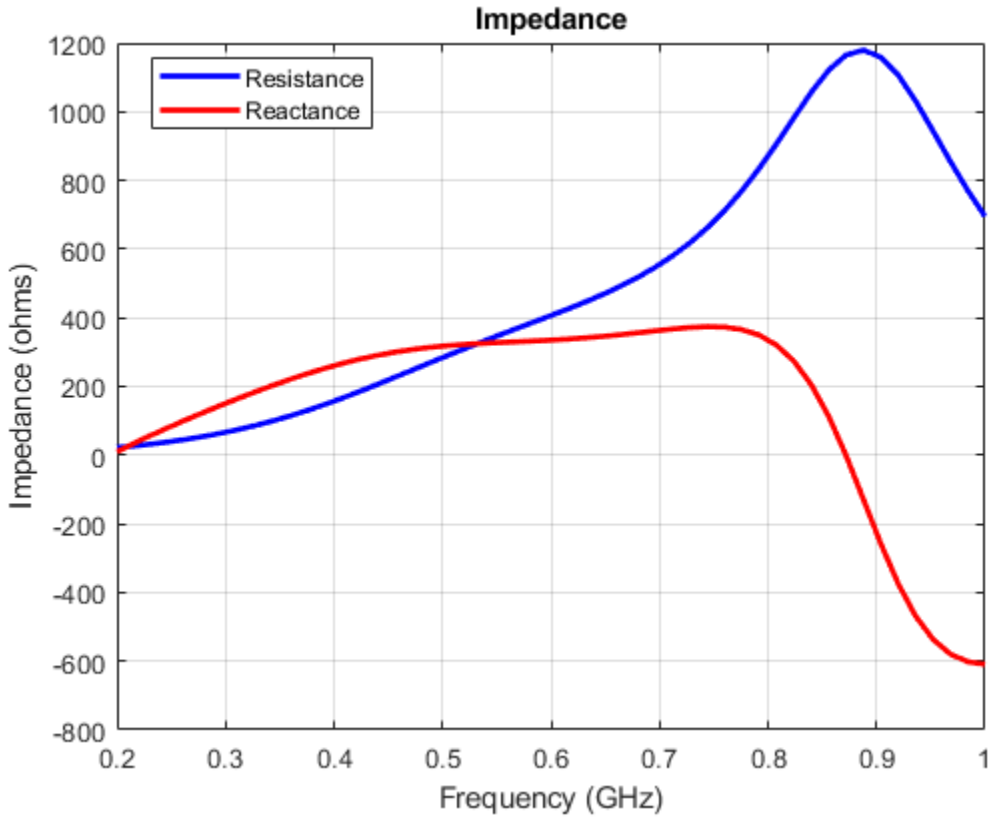
Assign the radiator shape to pcbStack and make the changes to the board shape and feed diameter properties.

```
boardShape = antenna.Rectangle('Length',0.6,'Width',0.3);
p = pcbStack;
p.BoardShape = boardShape;
p.Layers = {s};
p.FeedDiameter = .02*lambda/2;
p.FeedLocations = [0 0 1];
figure
show(p)
```



Analyze the impedance of the antenna. Effect of the end-loading should result in the series resonance to be pushed lower in the band.

```
figure
impedance(p,linspace(200e6,1e9,51))
```

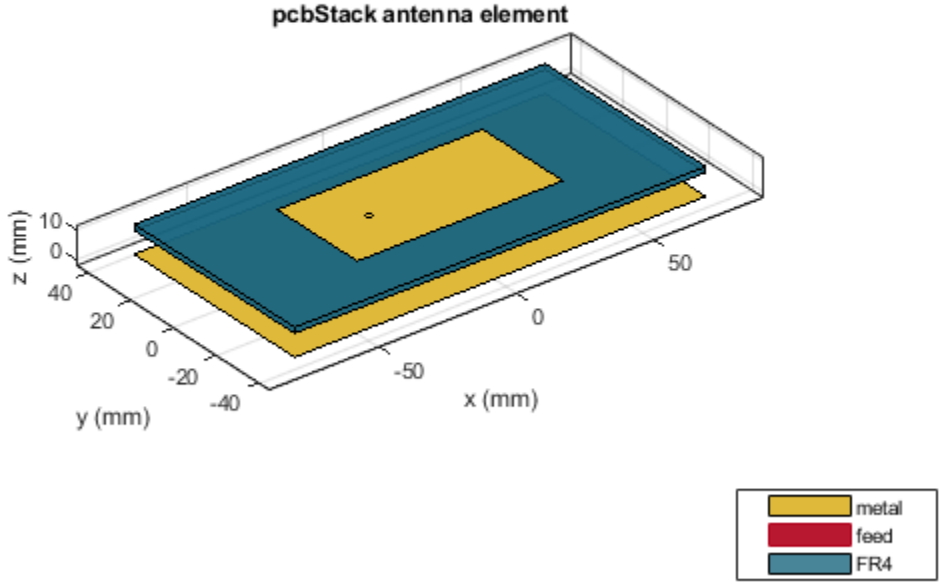


PCB Stack of Dielectric Antenna

Create a pcb stack antenna with 2 mm dielectric thickness at the radiator and air below it. Display the structure.

```
p = pcbStack;
d1 = dielectric('FR4');
d1.Thickness = 2e-3;
d2 = dielectric('Air');
d2.Thickness = 8e-3;
p.Layers = {p.Layers{1},d1,d2,p.Layers{2}};
```

```
p.FeedLocations(3:4) = [1 4];
show(p)
```



Directivity Pattern of PCB Stack Antenna

Create a PCB stack antenna from reflector backed bowtie.

```
b = design(bowtieRounded,1e9);
b.Tilt = 90
```

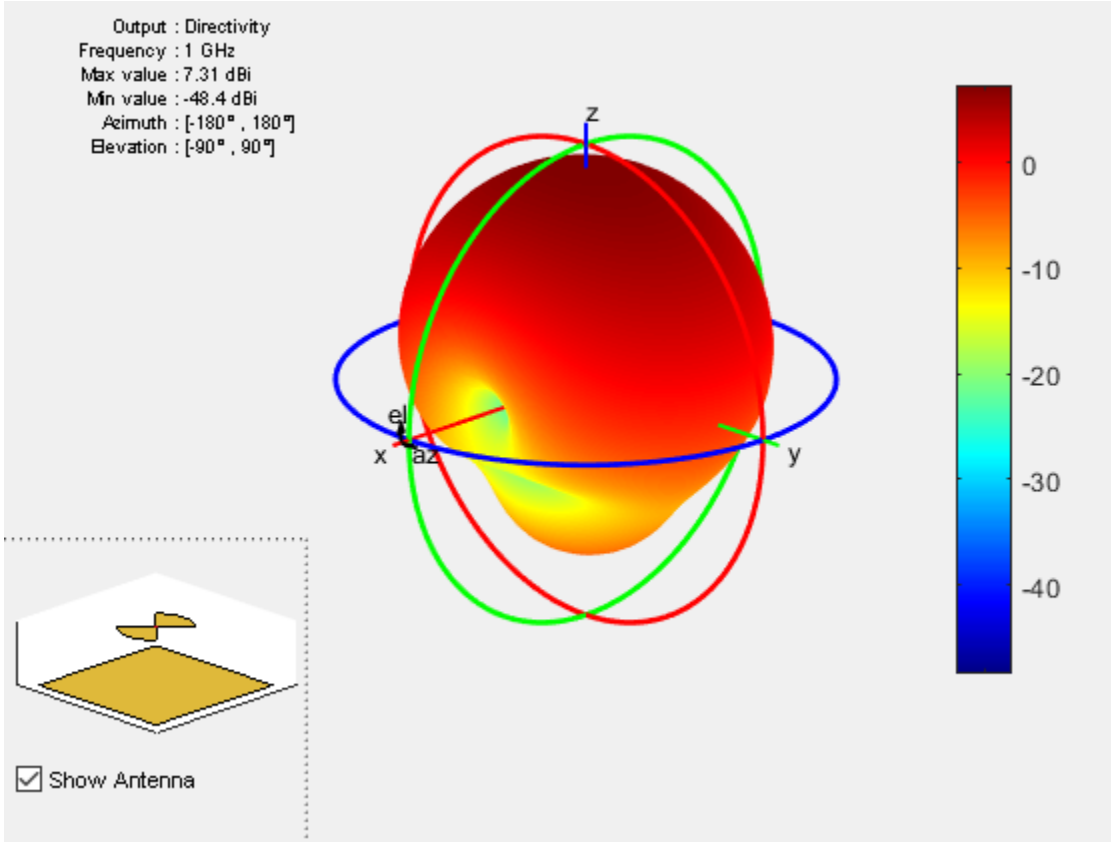
```
b =
  bowtieRounded with properties:
```

```
        Length: 0.0959
    FlareAngle: 90
          Tilt: 90
      TiltAxis: [1 0 0]
          Load: [1x1 lumpedElement]
```

```
b.TiltAxis = [0 1 0];
r = reflector('Exciter',b);
p = pcbStack(r);
```

Plot the directivity pattern of the antenna at 1 GHz.

```
pattern(p,1e9);
```



PCB Antenna From Antenna Library Elements

Create a coplanar inverted F antenna.

```
fco = invertedFcoplanar('Height',14e-3,'GroundPlaneLength', 100e-3,  ...
                  'GroundPlaneWidth', 100e-3);
```

Use this antenna to create a `pcbStack` object.

```
p = pcbStack(fco);
```

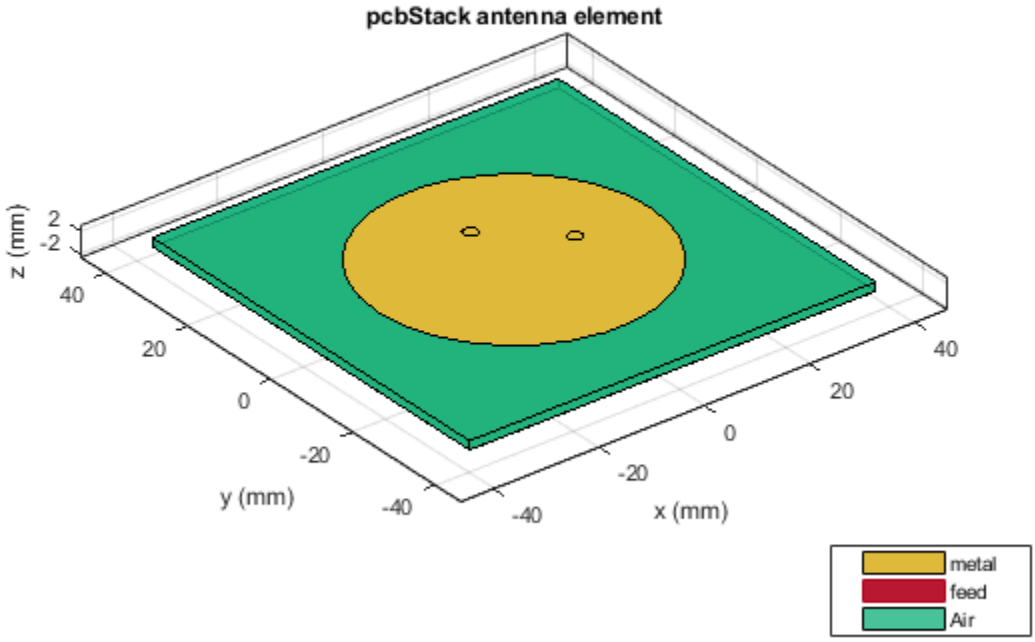
Stack Conversion

Create a circular microstrip patch.

```
p = patchMicrostripCircular;
d = dielectric;
d.EpsilonR = 4.4;
p.Radius = .0256;
p.Height = 1.6e-3;
p.Substrate = d;
p.GroundPlaneLength = 3*.0256;
p.GroundPlaneWidth = 3*.0256;
p.FeedOffset = [.0116 0];
```

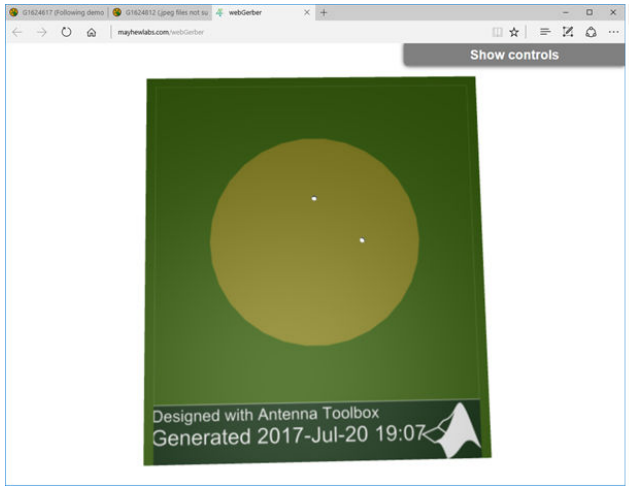
Create a PCB circular microstrip patch using `pcbStack`.

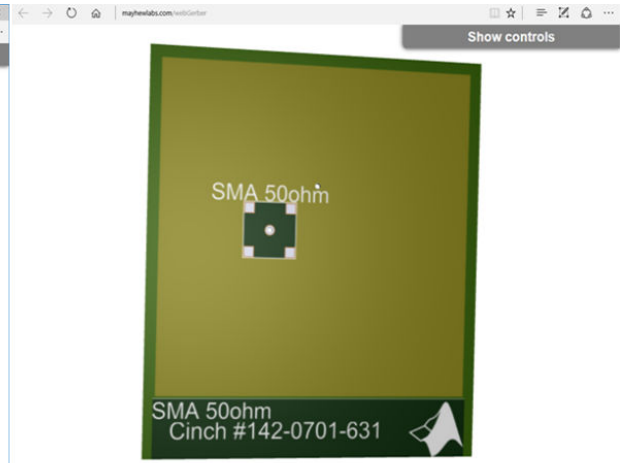
```
pb = pcbStack(p);
pb.FeedDiameter = 1.27e-3;
pb.ViaLocations = [0 pb.FeedLocations(1)/1.1 1 3];
pb.ViaDiameter = pb.FeedDiameter;
figure
show(pb)
```




```
C = SMA_Jack_Cinch;
O = PCBServices.MayhewWriter;
O.DefaultViaDiam = pb.ViaDiameter;
O.Filename = 'Microstrip circular patch-9a';
Am = PCBWriter(pb,O,C);
gerberWrite(Am)
```

Images using Mayhew Labs 3-D Viewer.





References

[1] Balanis, C. A. *Antenna Theory. Analysis and Design*. 3rd Ed. Hoboken, NJ: John Wiley & Sons, 2005.

[2] Stutzman, W. L. and Gary A. Thiele. *Antenna Theory and Design*. 3rd Ed. River Street, NJ: John Wiley & Sons, 2013.

See Also

`antenna.Circle` | `antenna.Polygon` | `antenna.Rectangle` | `customAntennaMesh` | `customArrayMesh`

Topics

“Design Variations On Microstrip Patch Antenna Using PCB Stack”
“Rotate Antenna and Arrays”

Introduced in R2017a

txsite

Create radio frequency transmitter site

Description

Use `txsite` object to create a radio frequency transmitter site.

Creation

Syntax

```
tx = txsite
tx = txsite(Name,Value)
```

Description

`tx = txsite` creates a radio frequency transmitter site.

`tx = txsite(Name,Value)` sets properties using one or more name-value pairs. For example, `tx = txsite('Name','TX Site')` creates a transmitter site with name `TX Site`. Enclose each property name in quotes.

Properties

Name — Site name
character vector | string | row or column vector

Site name, specified as a character vector or string, or row or column vector of *N* elements.

Example: `'Name','Site 2'`

Example: `TX.Name = 'Fenway Park'`

Example: If you want to assign multiple values then - `names = ["Fenway Park","Faneuil Hall","Bunker Hill Monument"]; TX = txsite('Name',names)`

Data Types: `char` | `string`

`Latitude` — Site latitude coordinates
`42.3001` (default) | numeric scalar | row or column vector

Site latitude coordinates, specified as a numeric scalar in the range of `-90` to `90`, or as a row or column vector of *N* elements. Coordinates are defined using Earth ellipsoid model WGS-84. Latitude is the north/south angle.

Example: `'Latitude',45.098`

Example: `TX.Latitude = 45.098`

Example: If you want to assign multiple values then - `latitude = [42.3467,42.3598,42.3763]; TX = txsite('Latitude',latitude)`

`Longitude` — Site longitude coordinates
`-71.3504` (default) | numeric scalar | row or column vector

Site longitude coordinates, specified as a numeric scalar or as a row or column vector of *N* elements. Coordinates are defined using Earth ellipsoid model WGS-84. Longitude is the east/west angle.

Example: `'Longitude',-68.890`

Example: `TX.Longitude = -71.0972`

Example: If you want to assign multiple values then - `longitude = [-71.0972,-71.0545,-71.0611]; TX = txsite('Longitude',longitude)`

`Antenna` — Antenna element or array
`dipole` (default) | object | row vector

Antenna element or array specified as an object. By default, the antenna is a dipole designed for 1.9 GHz.

Example: `'Antenna',monopole`

Example: `TX.Antenna = monopole`

`AntennaAngle` — Antenna x-axis angle
`0` (default) | numeric scalar | 2-by-1 vector | 2-by-*N* matrix

Antenna x-axis angle, specified as a numeric scalar or a 2-by-1 vector or a 2-by-*N* matrix in degrees.

The azimuth angle measured counterclockwise from the east to the antenna x-axis.

The elevation angle measures from the horizontal plane to antenna x-axis from `-90` to `90` degrees.

Example: `'AntennaAngle',25`

Example: `TX.AntennaAngle = [25,-80]`

`AntennaHeight` — Antenna height

`10` (default) | nonnegative numeric scalar

Antenna height from the ground elevation at the site to the center of the antenna element, specified as a nonnegative numeric scalar in meters. Maximum value for this property is 6,371,000 m.

Example: `'AntennaHeight',25`

Example: `TX.AntennaHeight = 15`

`SystemLoss` — System loss

`0` (default) | nonnegative numeric scalar | row vector

System loss, specified as a non-negative numeric scalar in dB.

System loss includes transmission line loss and any other miscellaneous system losses.

Example: `'SystemLoss',10`

Example: `txsite.SystemLoss = 10`

Data Types:

`TransmitterFrequency` — Transmitter operating frequency

`1.9000e+09` (default) | numeric scalar | row vector

Transmitter operating frequency, specified as a numeric scalar in Hz. The range is from `1e3` to `200e9`.

Example: `'TransmitterFrequency',30e9`

Example: `txsite.TransmitterFrequency = 30e9`

Data Types: `double`

`TransmitterPower` — Signal power at transmitter output
`10` (default) | positive numeric scalar

Signal power at transmitter output, specified as a positive numeric scalar in watts. The transmitter out is connected to the antenna.

Example: `'TransmitterPower',30`

Example: `txsite.TransmitterPower = 30`

Data Types: `double`

Object Functions

show	Show site location on map
hide	Hide site location on map
distance	Distance between sites
angle	Angle between sites
elevation	Ground elevation of site
location	Location coordinates at a given distance and angle from site
los	Plot or compute the line-of-sight (LOS) visibility between sites on a map
coverage	Display coverage map
sinr	Display signal-to-interference-plus-noise ratio (SINR) map
pattern	Plot antenna radiation pattern on map

Examples

Transmitter Site

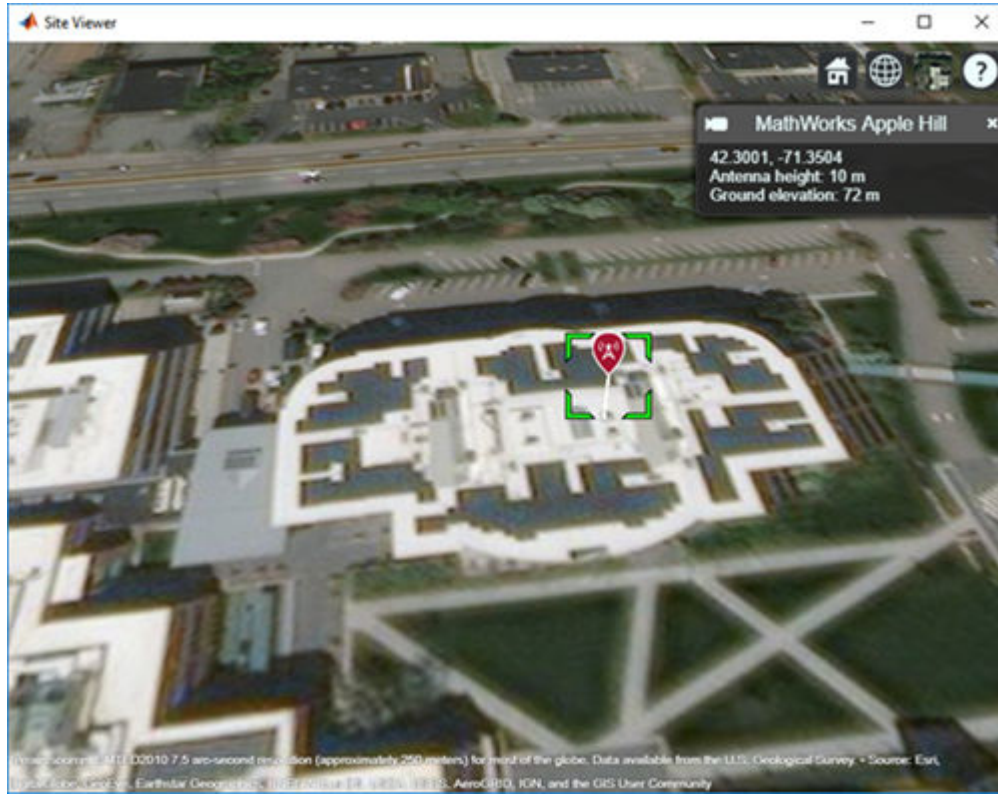
Create and view a transmitter site at a latitude of 42.3001 and a longitude of -71.3504.

```
tx = txsite('Name','MathWorks Apple Hill','Latitude',42.3001,...
    'Longitude',-71.3504)
```

```
tx =
  txsite with properties:

                    Name: 'MathWorks Apple Hill'
                Latitude: 42.3001
               Longitude: -71.3504
                 Antenna: [1×1 dipole]
```

```
          AntennaAngle: 0
         AntennaHeight: 10
            SystemLoss: 0
TransmitterFrequency: 1.9000e+09
     TransmitterPower: 10
```

```
show(tx);
```



Transmitter Array

Specify the names, latitudes, and longitudes of three transmitter locations.

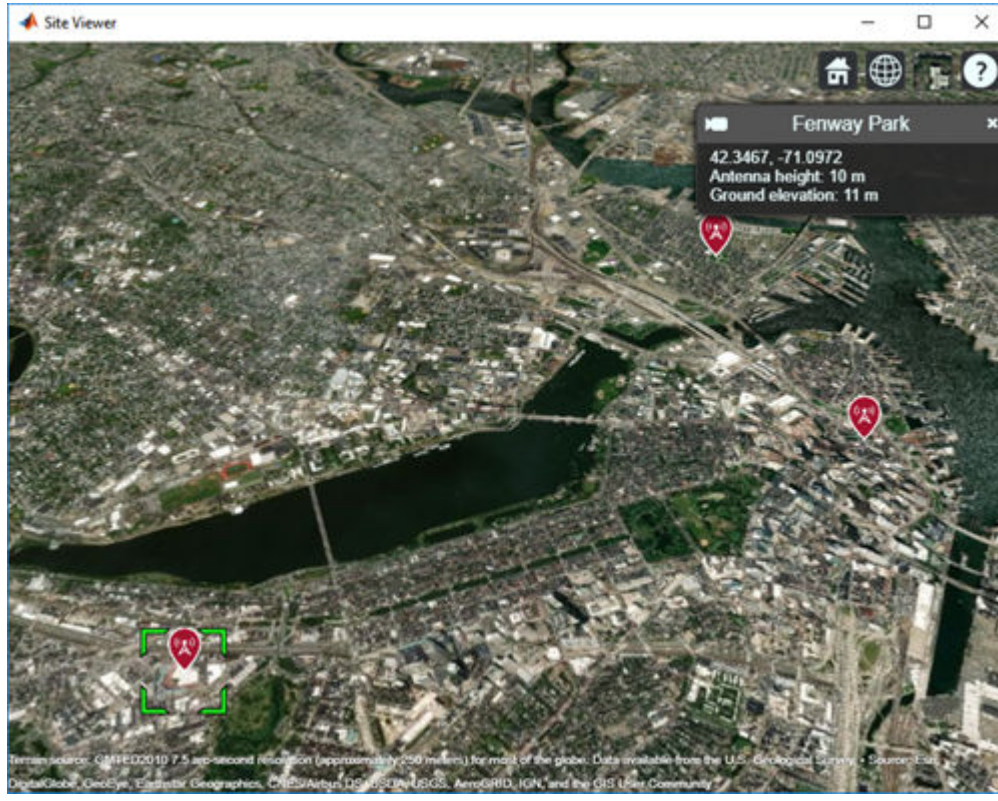
```
names = ["Fenway Park","Faneuil Hall","Bunker Hill Monument"];
lats = [42.3467,42.3598,42.3763];
lons = [-71.0972,-71.0545,-71.0611];
```

Define the frequency of the transmitters.

```
fq = 2.5e9;
```

Create and view the transmitter array.

```
txs = txsite('Name', names,...
'Latitude',lats,...
'Longitude',lons, ...
'TransmitterFrequency',fq);
show(txs)
```



See Also

`rxsite`

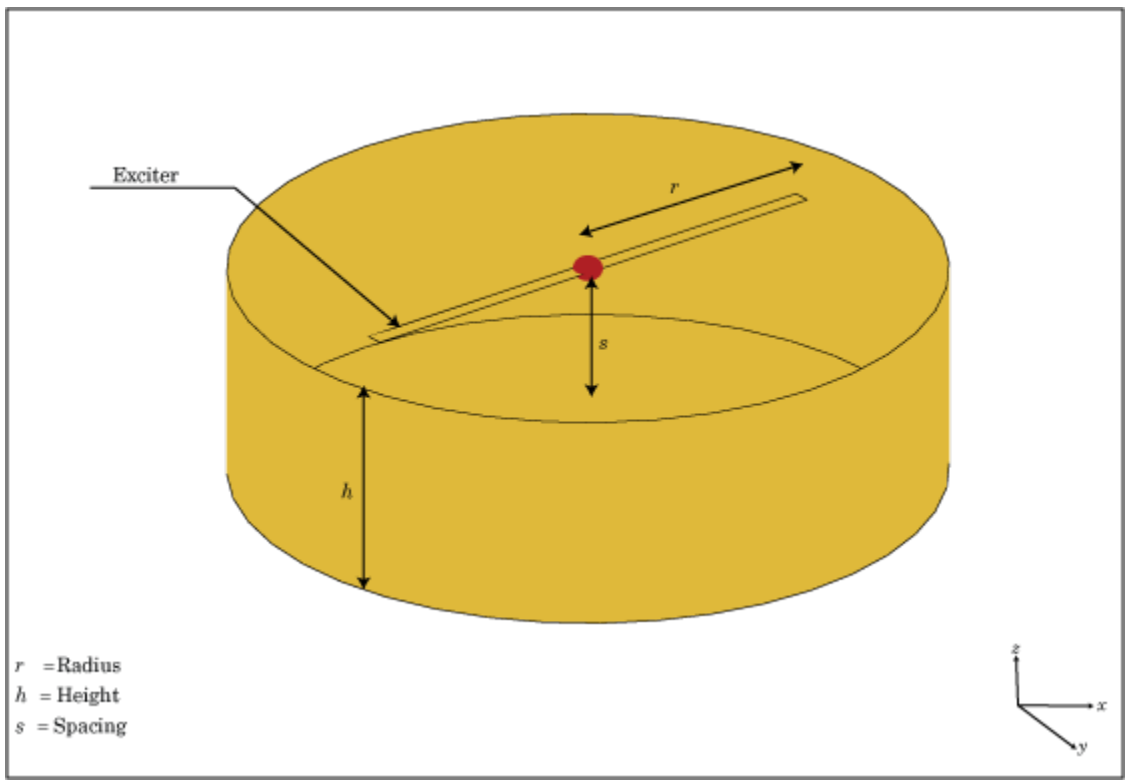
Introduced in R2017b

cavityCircular

Create circular cavity-backed antenna

Description

Use the `circularCavity` object to create a circular cavity-backed antenna. By default, the exciter used is a dipole. The dimensions are chosen for an operating frequency of 1 GHz.



Creation

Syntax

```
circularcavity = cavityCircular
circularcavity = cavityCircular(Name,Value)
```

Description

`circularcavity = cavityCircular` creates a circular cavity-backed antenna.

`circularcavity = cavityCircular(Name,Value)` sets properties using one or more name-value pairs. For example, `circularcavity = cavityCircular('Radius',0.2)` creates a circular cavity of radius 0.2 m. Enclose each property name in quotes.

Properties

`Exciter` — Antenna type used as exciter
`dipole` (default) | object

Antenna type used as an exciter, specified as an object. Except for reflector and cavity antenna elements, you can use any of the single elements in the Antenna Toolbox as an exciter.

Example: `'Exciter',monopole`

Example: `circularcavity.Exciter = monopole`

Data Types: `char` | `string`

`Radius` — Cavity radius
`0.1000` (default) | scalar

Radius of cavity, specified as a scalar in meters.

Example: `'Radius',0.2`

Example: `circularcavity.Radius = 0.2`

Data Types: `double`

`Height` — Cavity height along z-axis
`0.0750` (default) | scalar

Cavity height along z-axis, specified as a scalar in meters.

Example: `'Height',0.001`

Example: `circularcavity.Height = 0.001`

Data Types: `double`

`Spacing` — Distance between exciter and base of cavity
`0.0750` (default) | scalar

Distance between the exciter and the base of the cavity, specified a scalar in meters.

Example: `'Spacing',7.5e-2`

Example: `circularcavity.Spacing = 7.5e-2`

Data Types: `double`

`Substrate` — Type of dielectric material
'Air' (default) | object

Type of dielectric material used as a substrate, specified as a object. For more information see, `dielectric`. For more information on dielectric substrate meshing, see “Meshing”.

Note The substrate dimensions must be equal to the groundplane dimensions.

Example: `d = dielectric('FR4'); 'Substrate',d`

Example: `d = dielectric('FR4'); circularcavity.Substrate = d`

`EnableProbeFeed` — Create probe feed from backing structure to exciter
`0` (default) | `1`

Create probe feed from backing structure to exciter, specified as `0` or `1` or a positive scalar. By default, probe feed is not enabled.

Example: `'EnableProbeFeed',1`

Example: `circularcavity.EnableProbeFeed = 1`

Data Types: `double` | `logical`

Load — Lumped elements
[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `circularcavity.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

Circular Cavity-Backed Antenna

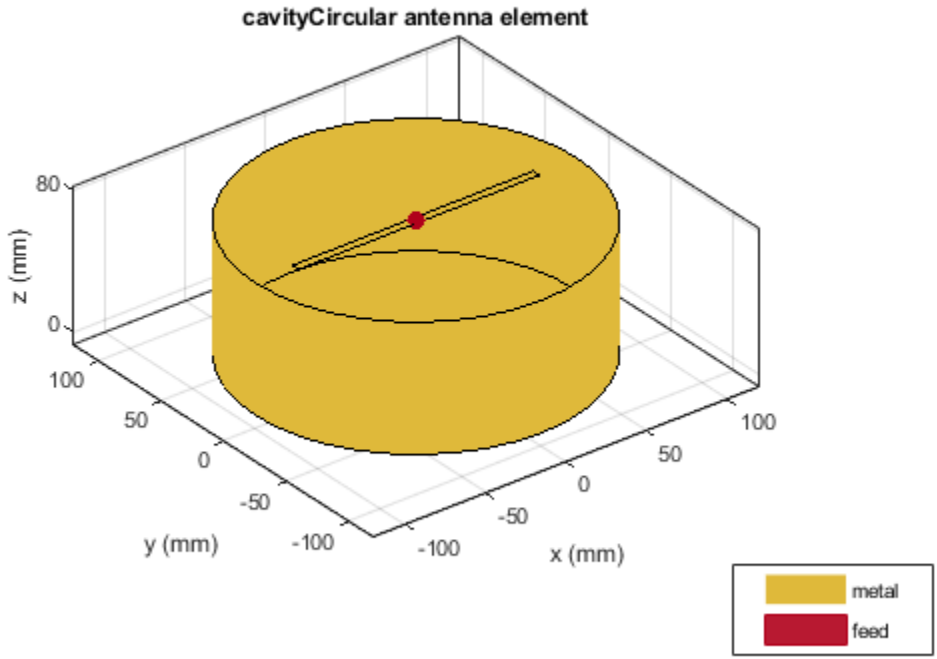
Create and view a default circular cavity-backed antenna.

```
a = cavityCircular
```

```
a =
  cavityCircular with properties:

           Exciter: [1x1 dipole]
         Substrate: [1x1 dielectric]
            Radius: 0.1000
            Height: 0.0750
           Spacing: 0.0750
    EnableProbeFeed: 0
              Tilt: 0
```

```
    TiltAxis: [1 0 0]
        Load: [1x1 lumpedElement]
```

```
show(a)
```



Circular Cavity-Backed Equiangular Spiral

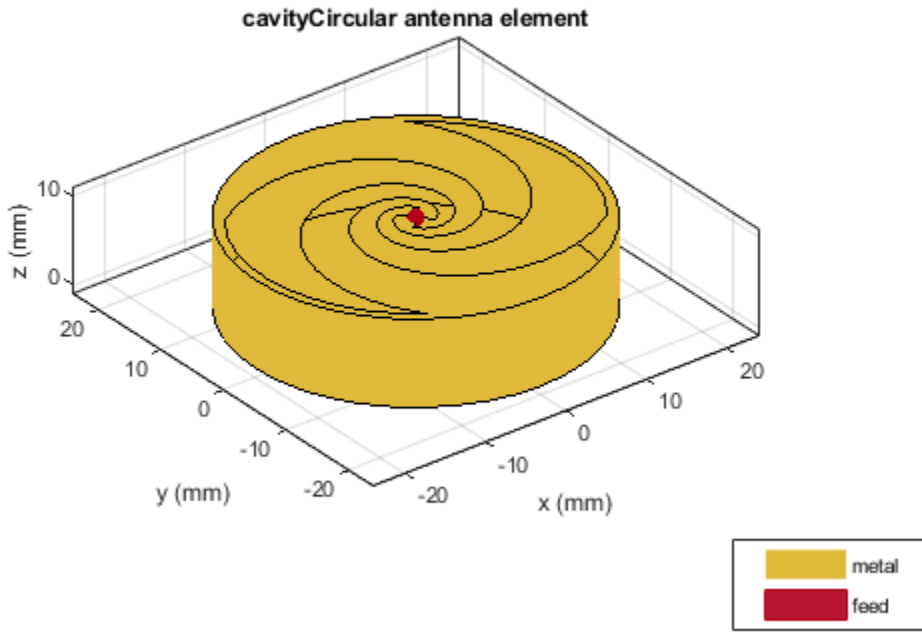
Create and view an equiangular spiral backed by a circular cavity. The cavity dimensions are:

Radius = 0.02 m

Height = 0.01 m

Spacing = 0.01 m

```
ant = cavityCircular('Exciter',spiralEquiangular,'Radius',0.02,   ...
        'Height',0.01,'Spacing', 0.01);
show(ant)
```



See Also

`cavity` | `reflector` | `reflectorCircular`

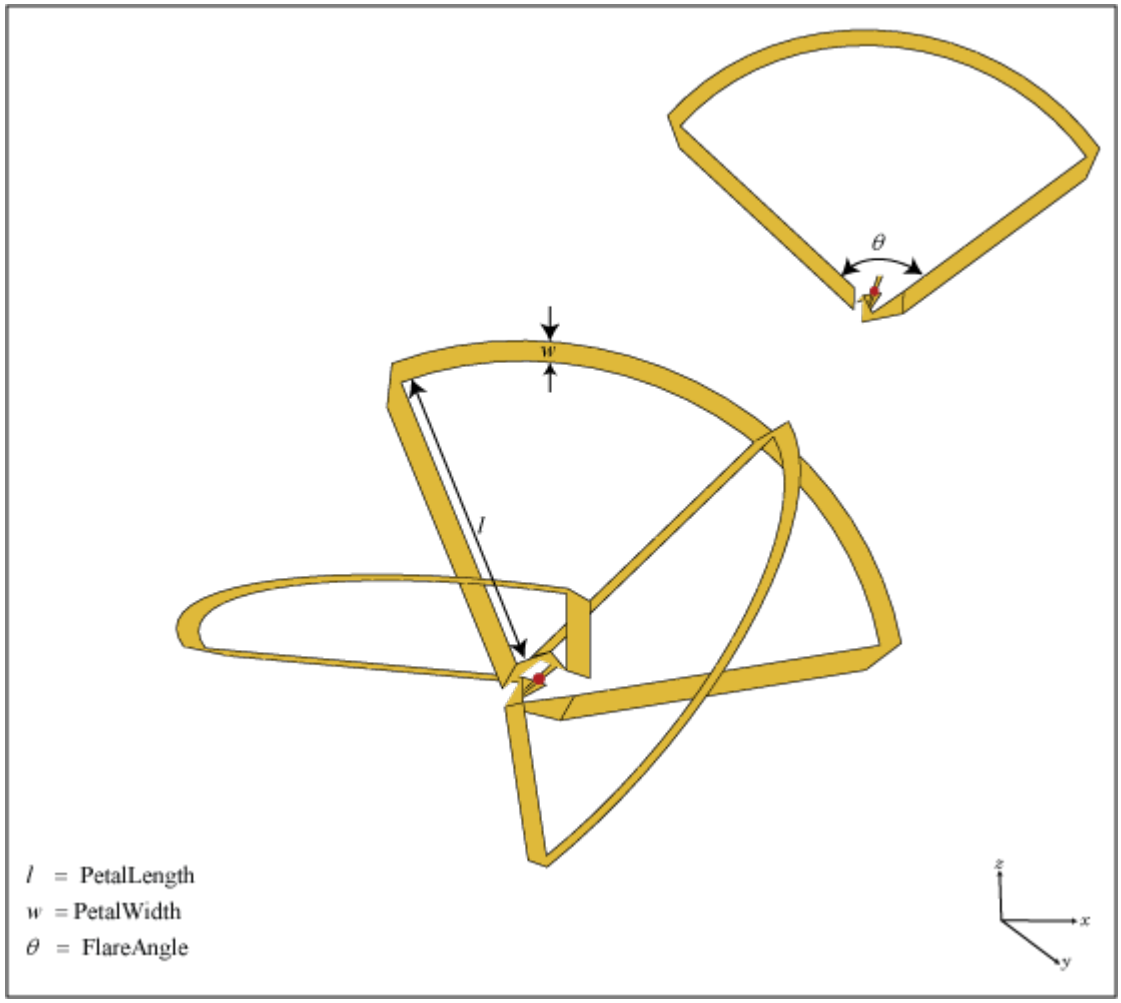
Introduced in R2017b

cloverleaf

Create three-petal cloverleaf antenna

Description

Use the `cloverleaf` object to create a three-petal cloverleaf antenna. The default cloverleaf has 3 petals and operates at around 5.8 GHz. It has a wideband circular polarization and an omnidirectional antenna.



Creation

Syntax

```
cl = cloverleaf
```

```
cl = cloverleaf(Name,Value)
```

Description

`cl = cloverleaf` creates a three-petal cloverleaf antenna.

`cl = cloverleaf(Name,Value)` sets properties using one or more name-value pairs. For example, `cl = cloverleaf('NumPetals',4)` creates a five petal cloverleaf antenna. Enclose each property name in quotes.

Properties

`NumPetals` — Number of petals
3 (default) | scalar

Number of petals, specified as a scalar.

Example: `'NumPetals',4`

Example: `cl.NumPetals = 4`

Data Types: `double`

`PetalLength` — Total length of leaf
`0.0515` (default) | scalar

Total length of leaf, specified as a scalar in meters.

Example: `'PetalLength',0.0025`

Example: `cl.PetalLength = 0.0025`

Data Types: `double`

`PetalWidth` — Leaf strip width
`8.0000e-04` (default) | scalar

Leaf strip width, specified as a scalar in meters.

Example: `'PetalWidth',0.001`

Example: `cl.PetalWidth = 0.001`

Data Types: `double`

`FlareAngle` — Leaf flare angle

`105` (default) | scalar

Leaf flare angle, specified as a scalar in degrees.

Example: `'FlareAngle',100`

Example: `cl.FlareAngle = 100`

Data Types: `double`

`Load` — Lumped elements

[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. You can add a load anywhere on the surface of the antenna. By default, it is at the origin. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `cl.Load = lumpedElement('Impedance',75)`

Data Types: `double`

`Tilt` — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector in degrees.

Example: `'Tilt',90`

Example: `cl.Tilt = [90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.

- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

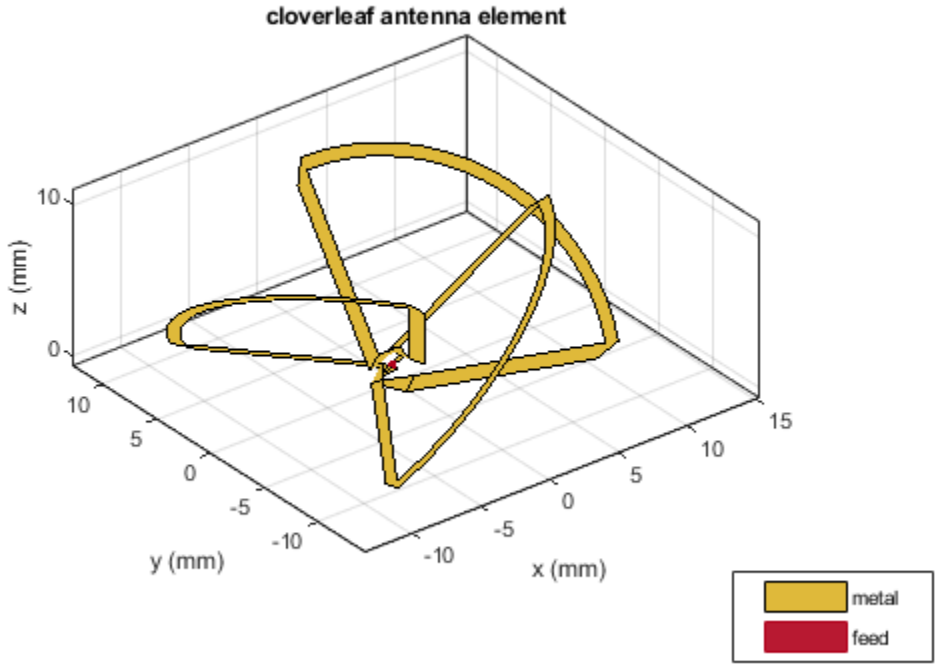
Clover Leaf Antenna

Create and view a default cloverleaf antenna.

```
cl = cloverleaf
```

```
cl =
  cloverleaf with properties:

      NumPetals: 3
    PetalLength: 0.0515
     PetalWidth: 8.0000e-04
     FlareAngle: 105
           Tilt: 0
       TiltAxis: [1 0 0]
           Load: [1x1 lumpedElement]
```

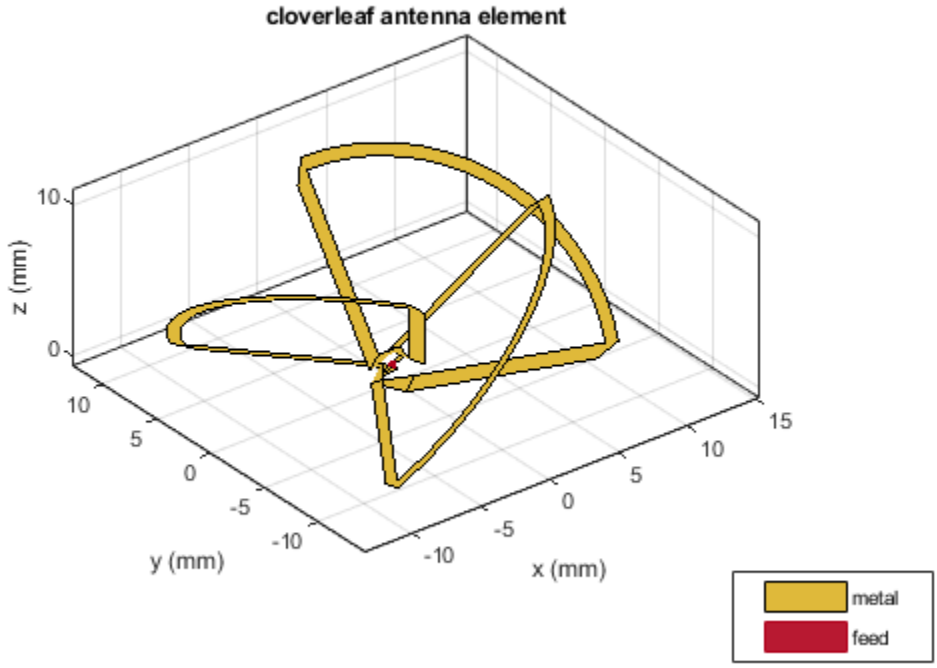
```
show(cl)
```



Axial Ratio of Cloverleaf Antenna

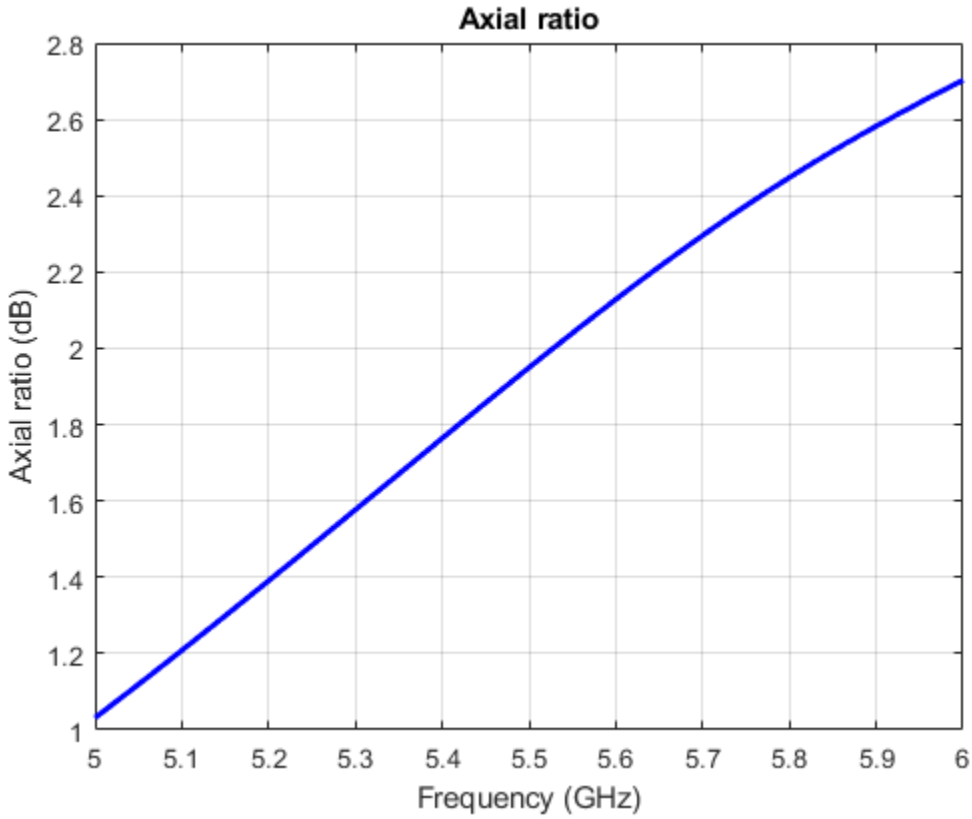
Create a cloverleaf antenna.

```
cl = cloverleaf;
show(cl);
```



Plot the axial ratio of the antenna from 5 GHz to 6 GHz.

```
freq = linspace(5e9,6e9,101);
axialRatio(cl,freq,0,0);
```



You can see from the axial ratio plot that the antenna supports circular polarization over the entire frequency range.

See Also

`dipole` | `spiralArchimedean`

Introduced in R2017b

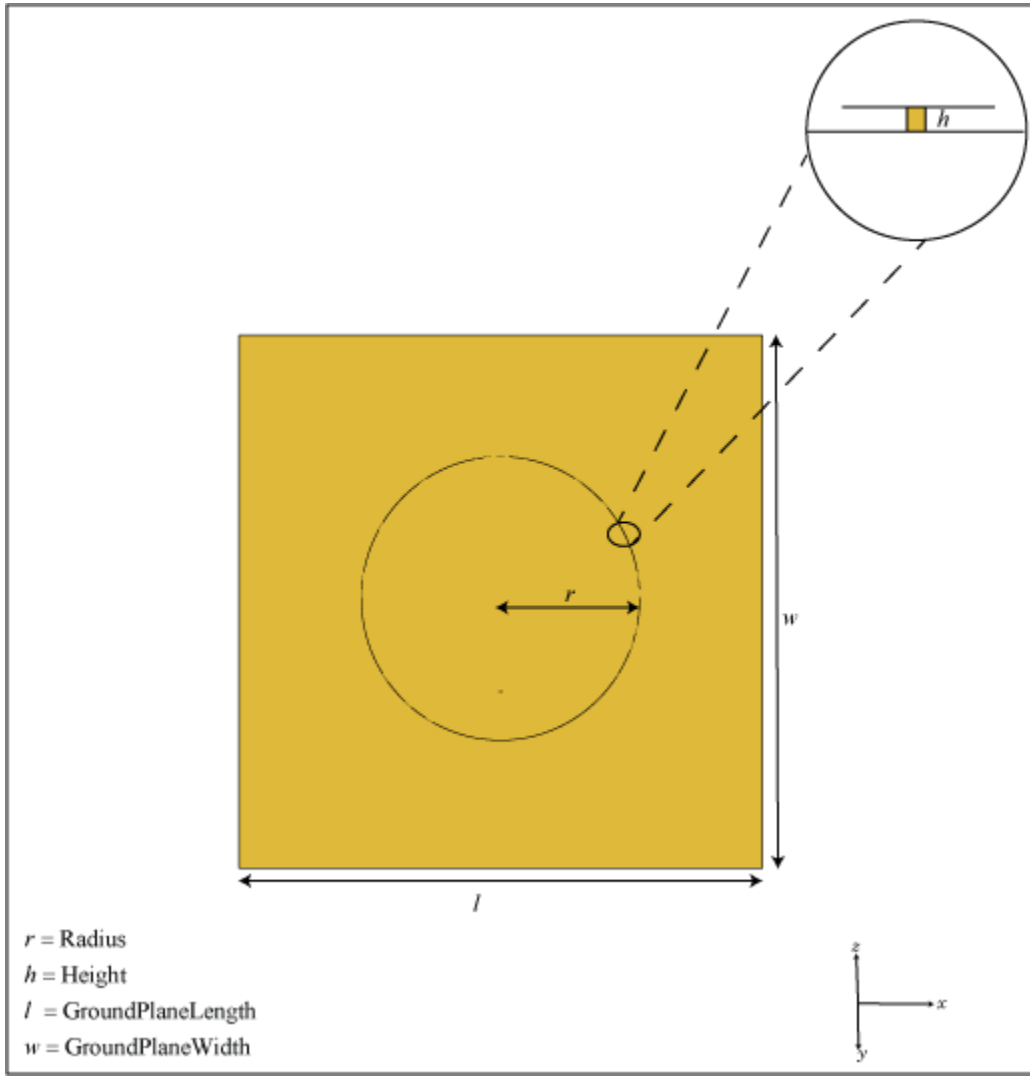
patchMicrostripCircular

Create probe-fed circular microstrip patch antenna

Description

Use the `patchMicrostripCircular` object to create a probe-fed circular microstrip patch antenna. By default, the patch is centered at the origin with feed point along the radius and the groundplane on the X-Y plane at z = 0.

Circular microstrip antennas are used as low-profile antennas in airborne and spacecraft applications. These antennas also find use in portable wireless applications because they are lightweight, low cost, and easily manufacturable.


h
r
w
l
r = Radius
h = Height
l = GroundPlaneLength
w = GroundPlaneWidth
z
x
y

Creation

Syntax

```
circularpatch = patchMicrostripCircular
circularpatch = patchMicrostripCircular(Name,Value)
```

Description

`circularpatch = patchMicrostripCircular` creates a probe-fed circular microstrip patch antenna.

`circularpatch = patchMicrostripCircular(Name,Value)` sets properties using one or more name-value pairs. For example, `circularpatch = patchMicrostripCircular('Radius',0.2)` creates a circular patch of radius 0.2 m. Enclose each property name in quotes.

Properties

`Radius` — Patch radius
`0.0798` (default) | scalar

Patch radius, specified as a scalar in meters. The default radius is for an operating frequency of 1 GHz.

Example: `'Radius',0.2`

Example: `circularpatch.Radius = 0.2`

Data Types: `double`

`Height` — Height of patch
`0.0060` (default) | scalar

Height of patch above the ground plane along the Z-axis, specified as a scalar in meters.

Example: `'Height',0.001`

Example: `circularpatch.Height = 0.001`

Data Types: `double`

`GroundPlaneLength` — Ground plane length
0.3000 (default) | scalar

Ground plane length along the X-axis, specified as a scalar in meters. Setting `'GroundPlaneLength'` to `Inf`, uses the infinite ground plane technique for antenna analysis.

Example: `'GroundPlaneLength',120e-3`

Example: `circularpatch.GroundPlaneLength = 120e-3`

Data Types: `double`

`GroundPlaneWidth` — Ground plane width
0.3000 (default) | scalar

Ground plane width along the Y-axis, specified as a scalar in meters. Setting `'GroundPlaneWidth'` to `Inf`, uses the infinite ground plane technique for antenna analysis.

Example: `'GroundPlaneWidth',120e-3`

Example: `circularpatch.GroundPlaneWidth = 120e-3`

Data Types: `double`

`Substrate` — Type of dielectric material
'Air' (default) | dielectric material object handle | dielectric material from dielectric catalog

Type of dielectric material used as a substrate, specified as a dielectric material object handle. For more information see, `dielectric`. For more information on dielectric substrate meshing, see “Meshing”.

Note The substrate dimensions must be lesser than the groundplane dimensions.

Example: `d = dielectric('FR4'); 'Substrate',d`

Example: `d = dielectric('FR4'); circularpatch.Substrate = d`

`PatchCenterOffset` — Signed distance from center along length and width of ground plane

`[0 0]` (default) | two-element real vector

Signed distance from center along length and width of ground plane, specified as a two-element real vector with each element unit in meters. Use this property to adjust the location of the patch relative to the ground plane.

Example: `'PatchCenterOffset',[0.01 0.01]`

Example: `circularpatch.PatchCenterOffset = [0.01 0.01]`

Data Types: `double`

`FeedOffset` — Signed distance from center along length and width of ground plane

[–0.0525 0] (default) | two-element real vector

Signed distance from center along length and width of ground plane, specified as a two-element real vector with each element unit in meters. Use this property to adjust the location of the feedpoint relative to the ground plane and patch.

Example: `'FeedOffset',[0.01 0.01]`

Example: `circularpatch.FeedOffset = [0.01 0.01]`

Data Types: `double`

`Load` — Lumped elements

[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `circularpatch.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `circularPatch.Tilt = [90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array

patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

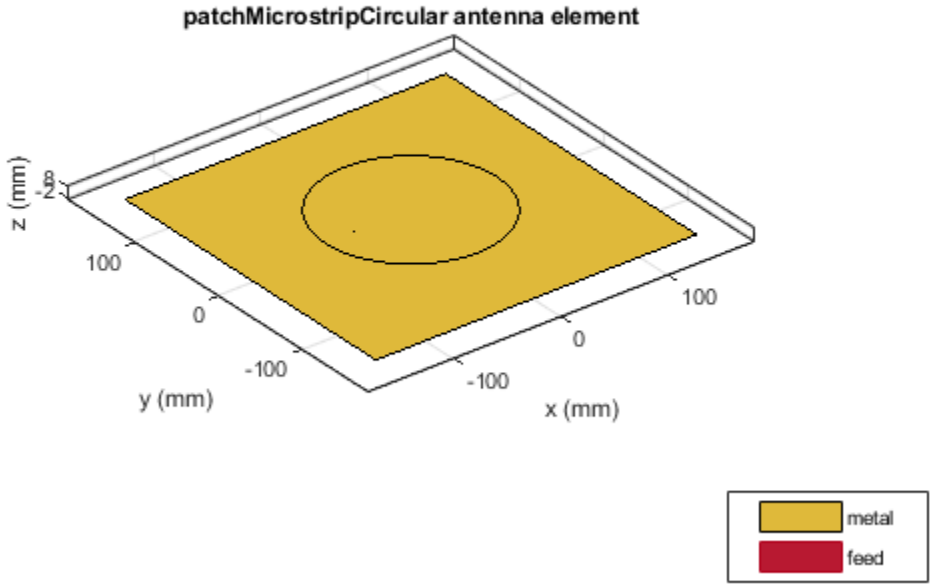
Circular Microstrip Patch

Create and view a default circular microstrip patch.

```
cp = patchMicrostripCircular
```

```
cp =
  patchMicrostripCircular with properties:

              Radius: 0.0798
              Height: 0.0060
           Substrate: [1x1 dielectric]
    GroundPlaneLength: 0.3000
     GroundPlaneWidth: 0.3000
    PatchCenterOffset: [0 0]
          FeedOffset: [-0.0525 0]
                Tilt: 0
            TiltAxis: [1 0 0]
                Load: [1x1 lumpedElement]
```

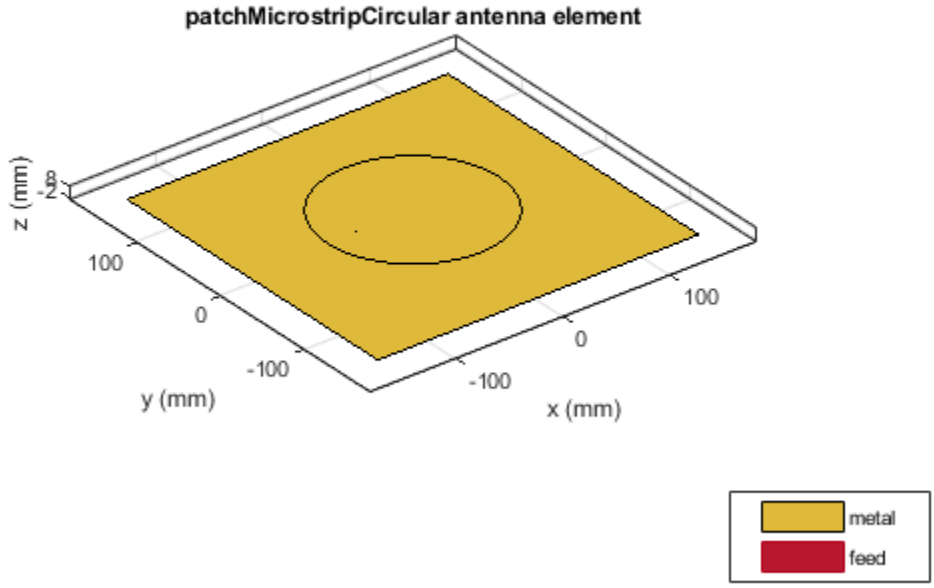
```
show(cp)
```



Radiation Pattern and Impedance of Circular Microstrip Patch

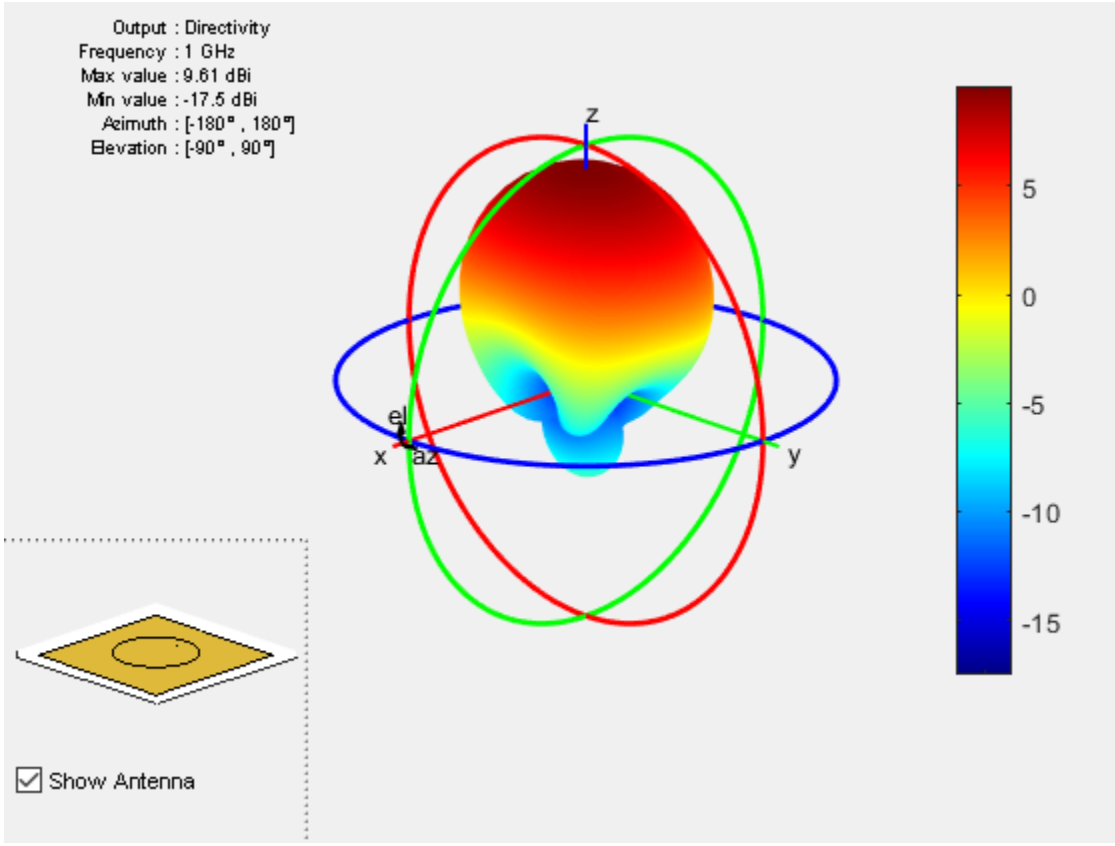
Create a circular patch antenna with radius 79.8 mm over a 300 mm by 300 mm ground plane, and offset the feed by (-52.5mm,0). Display the antenna.

```
cp = patchMicrostripCircular('Radius',0.0798,'Height',6e-3,...
       'GroundPlaneLength',0.3,'GroundPlaneWidth',0.3,...
       'FeedOffset',[-0.0525 0]);

show(cp)
```



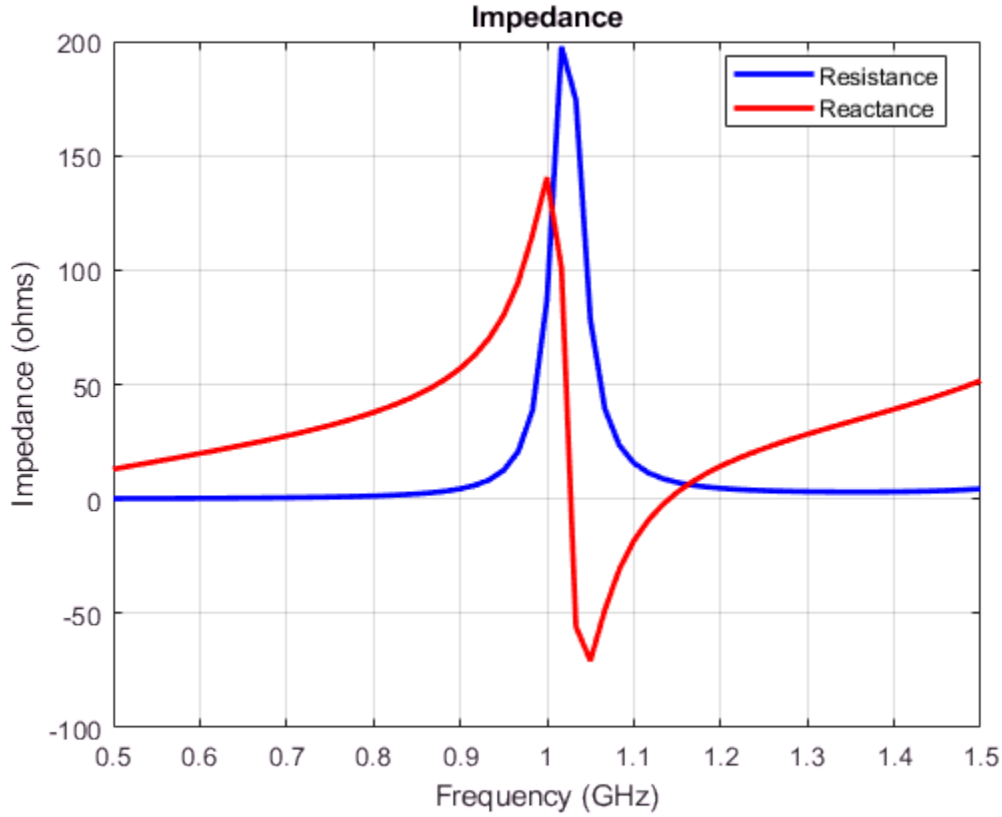
Plot the pattern of the patch antenna at 1 GHz.

```
pattern(cp,1e9);
```



Calculate the impedance of the antenna over a frequency span of 0.5GHz to 1.5GHz.

```
f = linspace(0.5e9,1.5e9,61);
impedance(cp,f);
```



See Also

`patchMicrostrip` | `patchMicrostripInsetfed`

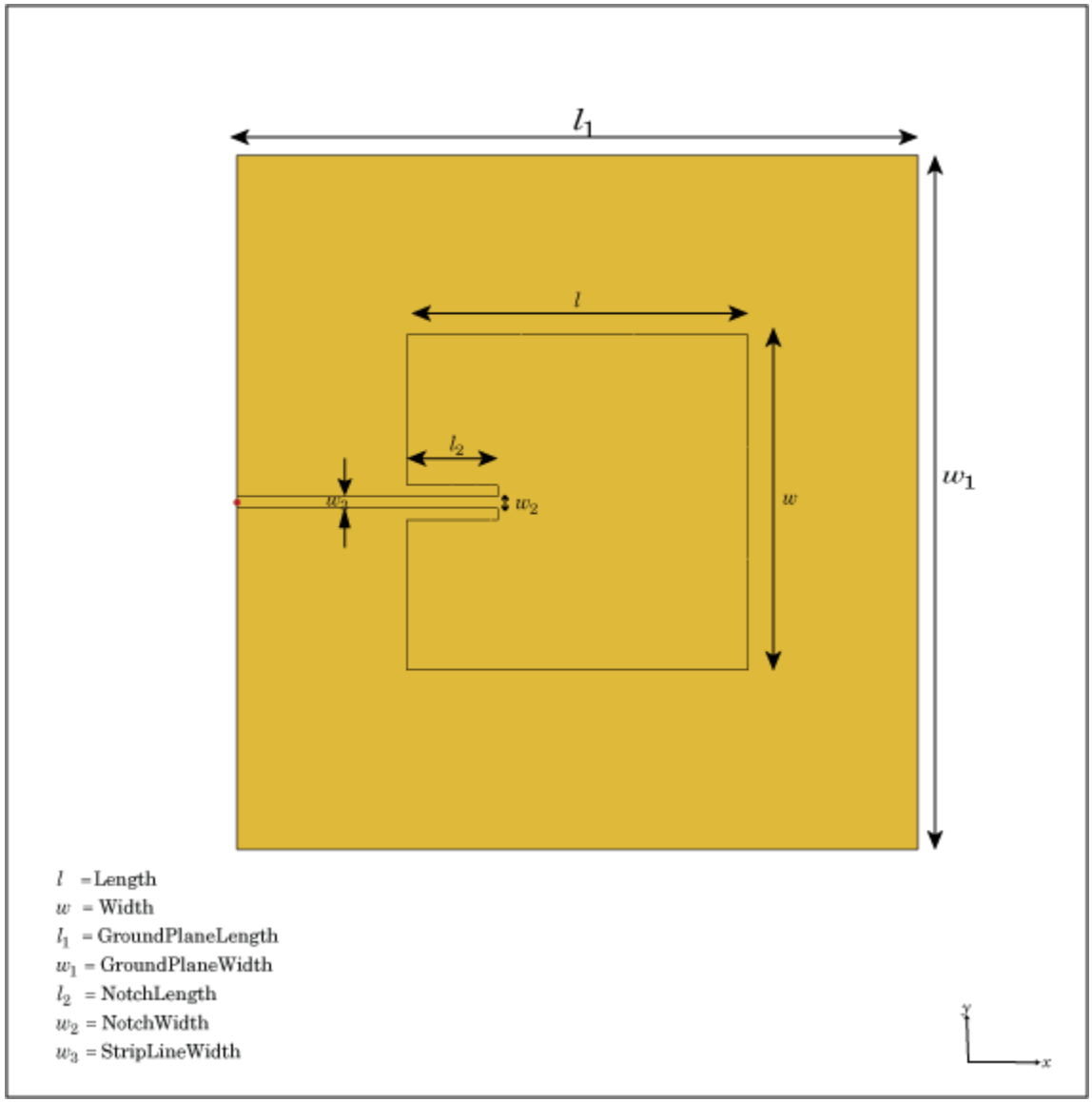
Introduced in R2017b

patchMicrostripInsetfed

Create inset-fed microstrip patch antenna

Description

Use the `patchMicrostripInsetfed` object to create an inset-fed microstrip patch antenna. The default patch is centered at the origin.


l_1
l
l_2
w_3
w_2
w
w_1
l = Length
w = Width
l_1 = GroundPlaneLength
w_1 = GroundPlaneWidth
l_2 = NotchLength
w_2 = NotchWidth
w_3 = StripLineWidth
y
x

Creation

Syntax

```
insetpatch = patchMicrostripInsetfed
insetpatch = patchMicrostripInsetfed(Name,Value)
```

Description

`insetpatch = patchMicrostripInsetfed` creates an inset-fed microstrip patch antenna centered at the origin.

`insetpatch = patchMicrostripInsetfed(Name,Value)` sets properties using one or more name-value pair. For example, `insetpatch = patchMicrostripInsetfed('Length',0.2)` creates an inset-fed patch of length 0.2 m. Enclose each property name in quotes.

Properties

`Length` — Patch length along X-axis

`0.0300` (default) | scalar

Patch length along X-axis, specified as a scalar in meters. The default length is for an operating frequency of 4.5 GHz.

Example: `'Length',0.2`

Example: `insetpatch.Length = 0.2`

Data Types: `double`

`Width` — Patch width along Y-axis

`0.0290` (default) | scalar

Patch width along Y-axis, specified as a scalar in meters.

Example: `'Width',0.1`

Example: `insetpatch.Width = 0.1`

Data Types: `double`

`Height` — Patch height along Z-axis
`0.0013` (default) | scalar

Patch height along Z-axis, specified as a scalar in meters.

Example: `'Height',0.001`

Example: `insetpatch.Height = 0.001`

Data Types: `double`

`GroundPlaneLength` — Ground plane length along X-axis
0.0600 (default) | scalar

Ground plane length along X-axis, specified as a scalar in meters. Setting `'GroundPlaneLength'` to `Inf`, uses the infinite ground plane technique for antenna analysis.

Example: `'GroundPlaneLength',120e-3`

Example: `insetpatch.GroundPlaneLength = 120e-3`

Data Types: `double`

`GroundPlaneWidth` — Ground plane width along Y-axis
0.0600 (default) | scalar

Ground plane width along Y-axis, specified as a scalar in meters. Setting `'GroundPlaneWidth'` to `Inf`, uses the infinite ground plane technique for antenna analysis.

Example: `'GroundPlaneWidth',120e-3`

Example: `insetpatch.GroundPlaneWidth = 120e-3`

Data Types: `double`

`Substrate` — Type of dielectric material
'Air' (default) | dielectric material object handle

Type of dielectric material used as a substrate, specified as a dielectric material object handle. For more information see, `dielectric`. For more information on dielectric substrate meshing, see “Meshing”.

Note The substrate dimensions must be equal to the groundplane dimensions.

Example: `d = dielectric('FR4'); 'Substrate',d`

Example: `d = dielectric('FR4'); insetpatch.Substrate = d`

`PatchCenterOffset` — Signed distance of patch from origin

`[0 0]` (default) | two-element real vector

Signed distance of patch from origin, specified as a two-element real vector with each element unit in meters. Use this property to adjust the location of the patch relative to the ground plane.

Example: `'PatchCenterOffset',[0.01 0.01]`

Example: `insetpatch.PatchCenterOffset = [0.01 0.01]`

Data Types: `double`

`FeedOffset` — Signed distance of feed from origin

[–0.0300 0] (default) | two-element real vector

Signed distance of feed from origin, specified as a two-element real vector with each element unit in meters. Use this property to adjust the location of the feedpoint relative to the ground plane and patch.

Example: `'FeedOffset',[0.01 0.01]`

Example: `insetpatch.FeedOffset = [0.01 0.01]`

Data Types: `double`

`StripLineWidth` — Strip line width along Y-axis

`1.0000e-03` (default) | scalar

Strip line width along Y-axis, specified as a scalar in meters.

Example: `'StripLineWidth',0.1`

Example: `insetpatch.StripLineWidth = 0.1`

Data Types: `double`

`NotchLength` — Notch length along X-axis

`0.0080` (default) | scalar

Notch length along X-axis, specified as a scalar in meters.

Example: `'NotchLength',0.2`

Example: `insetpatch.NotchLength = 0.2`

Data Types: `double`

`NotchWidth` — Notch width along Y-axis

`0.0030` (default) | scalar

Notch width along Y-axis, specified as a scalar in meters.

Example: `'NotchWidth',0.1`

Example: `insetpatch.NotchWidth = 0.1`

Data Types: `double`

`Load` — Lumped elements

[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `insetpatch.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `insetpatch.Tilt = [90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

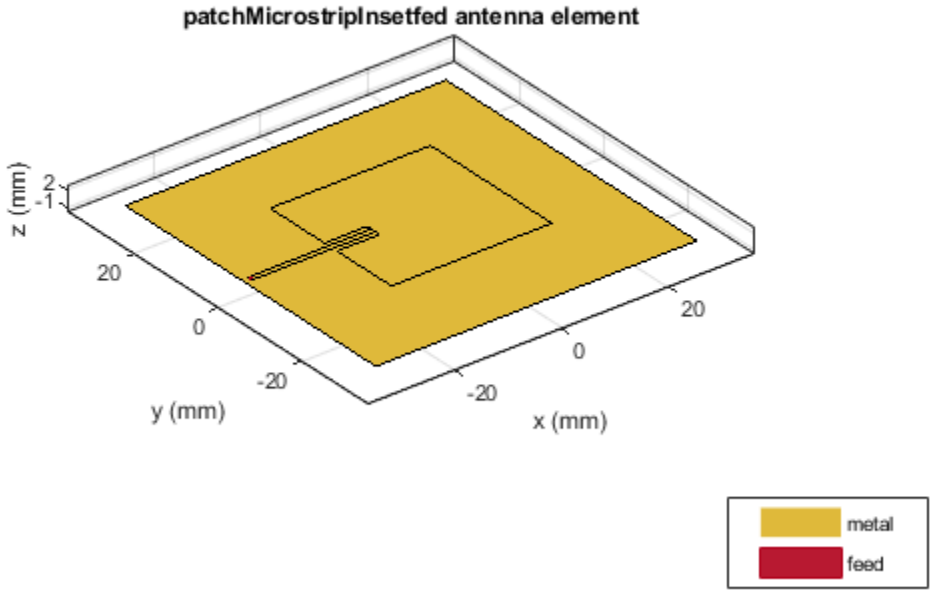
Inset-Fed Microstrip Patch

Create and view a default inset-fed microstrip patch.

```
insetpatch = patchMicrostripInsetfed
```

```
insetpatch =
  patchMicrostripInsetfed with properties:

               Length: 0.0300
                Width: 0.0290
               Height: 0.0013
            Substrate: [1x1 dielectric]
    PatchCenterOffset: [0 0]
           FeedOffset: [-0.0300 0]
       StripLineWidth: 1.0000e-03
          NotchLength: 0.0080
           NotchWidth: 0.0030
    GroundPlaneLength: 0.0600
     GroundPlaneWidth: 0.0600
                 Tilt: 0
             TiltAxis: [1 0 0]
                 Load: [1x1 lumpedElement]
```

```
show(insetpatch)
```



See Also

`patchMicrostrip` | `patchMicrostripCircular`

Topics

"Analysis of an Inset-Feed Patch Antenna on a Dielectric Substrate"

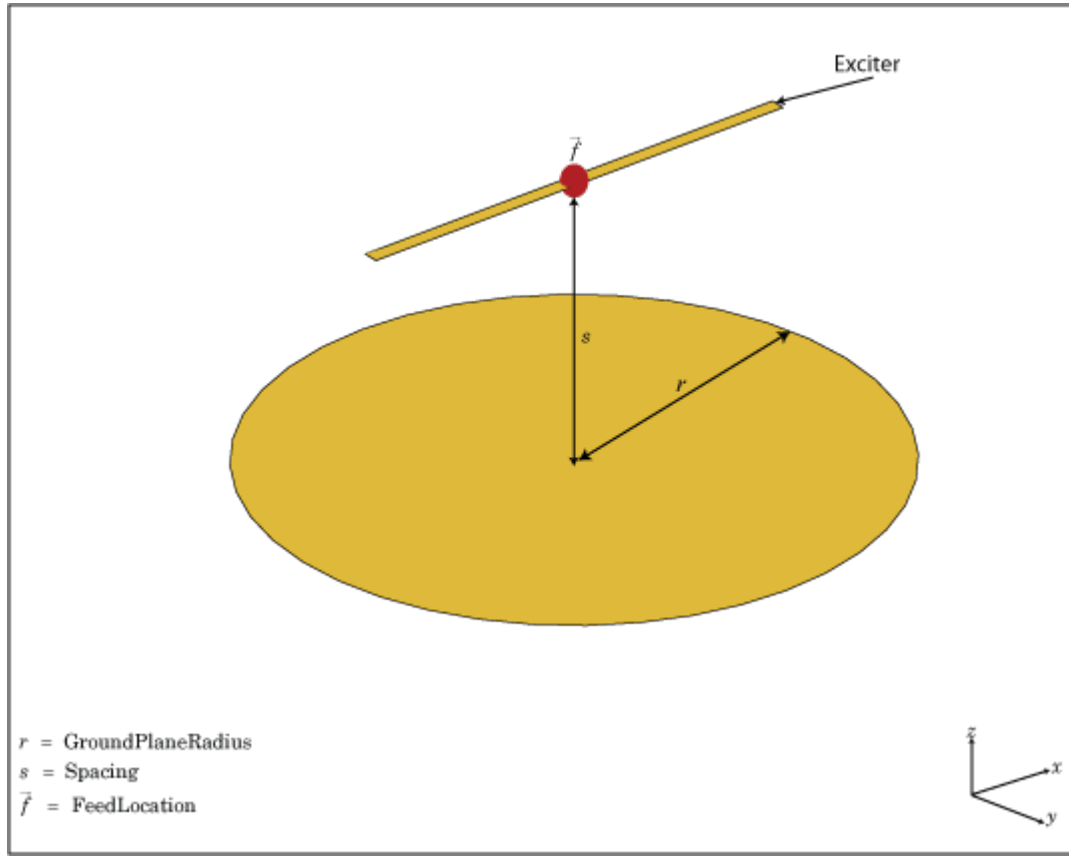
Introduced in R2017b

reflectorCircular

Create circular reflector-backed antenna

Description

Use the `reflectorCircular` object to create a circular reflector-backed antenna. By default the exciter is a dipole. The dimensions are chosen for an operating frequency of 1 GHz.



Creation

Syntax

```
rc = reflectorCircular
rc = reflectorCircular(Name,Value)
```

Description

`rc = reflectorCircular` creates a circular reflector backed antenna.

`rc = reflectorCircular(Name,Value)` sets properties using one or more name-value pair. For example, `rc = reflectorCircular('Radius',0.2)` creates a circular reflector of radius 0.2 m. Enclose each property name in quotes.

Properties

`Exciter` — Antenna type used as exciter
`dipole` (default) | object

Antenna type used as an exciter, specified as an object. Except for reflector and cavity antenna elements, you can use all the single elements in the Antenna Toolbox as an exciter.

Example: `'Exciter',spiralEquiangular`

Example: `rc.Exciter = spiralEquiangular`

`GroundPlaneRadius` — Reflector radius
`0.1000` (default) | scalar

Radius of reflector, specified as a scalar in meters.

Example: `'Radius',0.2`

Example: `rc.Radius = 0.2`

Data Types: `double`

`Spacing` — Distance between exciter and reflector bottom

`0.0750` (default) | scalar

Distance between the exciter and the reflector, specified as a scalar in meters.

Example: `'Spacing',7.5e-2`

Example: `rc.Spacing = 7.5e-2`

Data Types: `double`

`Substrate` — Type of dielectric material

'Air' (default) | object

Type of dielectric material used as a substrate, specified as an object. For more information see, `dielectric`. For more information on dielectric substrate meshing, see “Meshing”.

Note The substrate dimensions must be equal to the groundplane dimensions.

Example: `d = dielectric('FR4'); 'Substrate',d`

Example: `d = dielectric('FR4'); rc.Substrate = d`

`EnableProbeFeed` — Create probe feed from backing structure to exciter

`0` (default) | `1` | scalar

Create probe feed from backing structure to exciter, specified as `0` or `1` or a scalar. By default, probe feed is not enabled.

Example: `'EnableProbeFeed',1`

Example: `rc.EnableProbeFeed = 1`

Data Types: `double` | `logical`

`Load` — Lumped elements

[1x1 `lumpedElement`] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. You can add a load anywhere on the surface of the antenna. By default, the load is at the origin. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `rc.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector in degrees.

Example: `'Tilt',90`

Example: `rc.Tilt = [90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna

charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

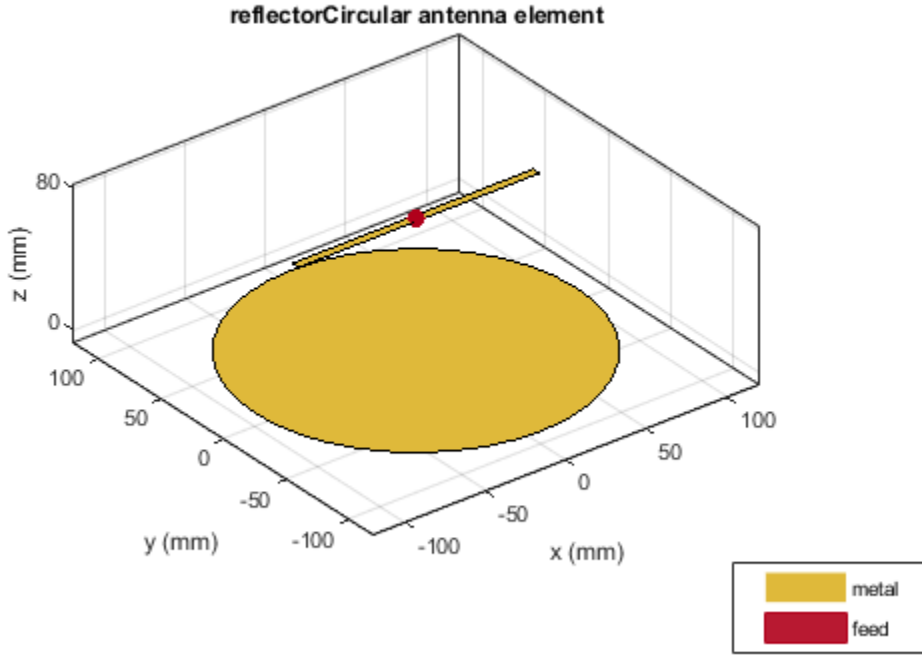
Circular Reflector Backed Antenna

Create and view a default circular reflector backed antenna.

```
rc = reflectorCircular
```

```
rc =
  reflectorCircular with properties:

             Exciter: [1x1 dipole]
           Substrate: [1x1 dielectric]
    GroundPlaneRadius: 0.1000
              Spacing: 0.0750
      EnableProbeFeed: 0
                 Tilt: 0
             TiltAxis: [1 0 0]
                 Load: [1x1 lumpedElement]
```

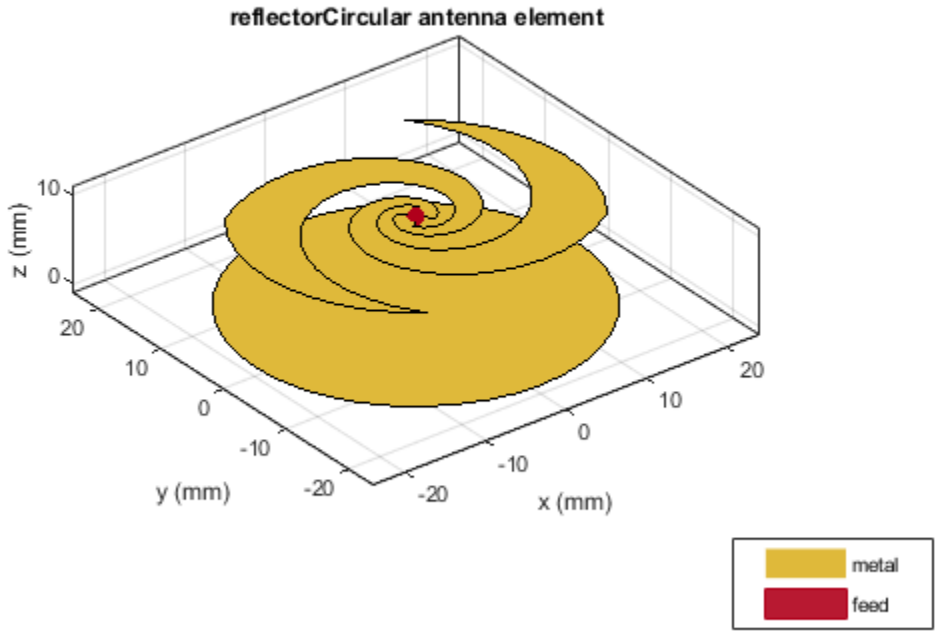
```
show(rc);
```



Radiation Pattern of Circular Reflector Backed Antenna

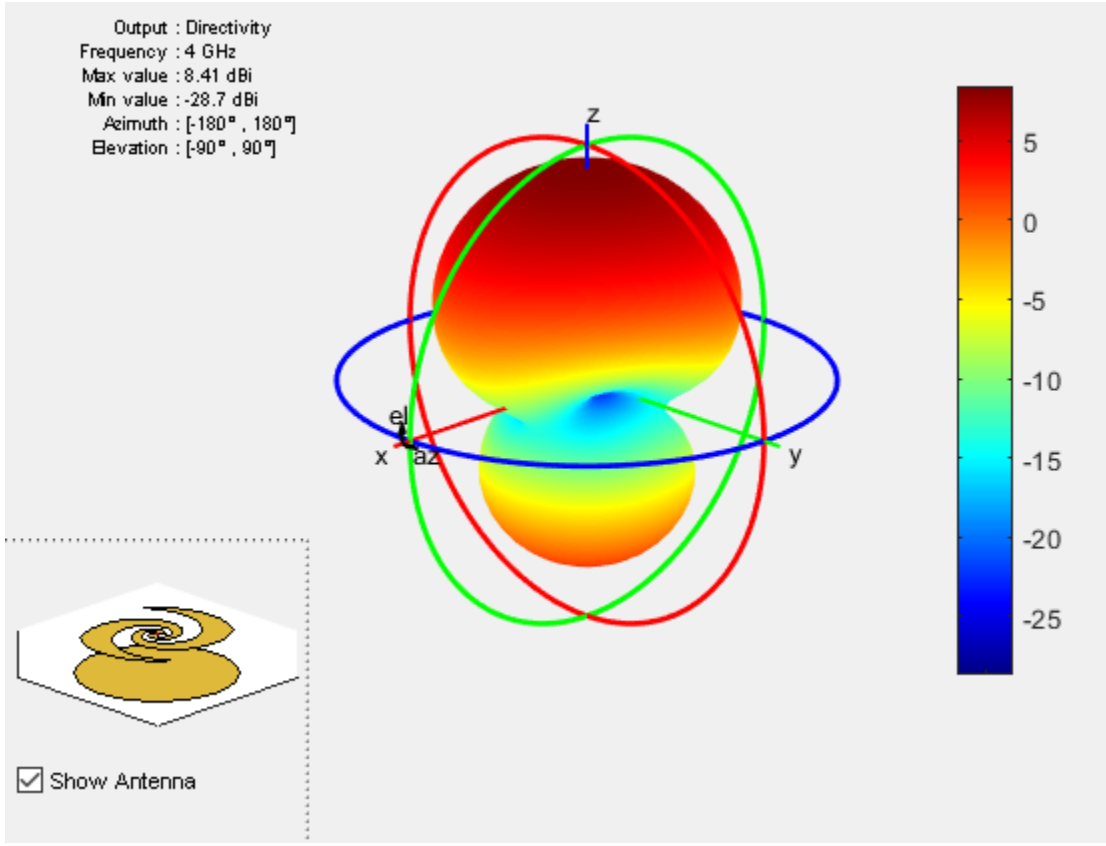
Create an equiangular spiral backed by a circular reflector.

```
ant = reflectorCircular('Exciter',spiralEquiangular,'GroundPlaneRadius',  ...
          0.02,'Spacing', 0.01);
show(ant)
```



Plot the radiation pattern of the antenna at 4 GHz.

```
pattern(ant,4e9)
```



See Also

`cavity` | `cavityCircular` | `reflector`

Introduced in R2017b

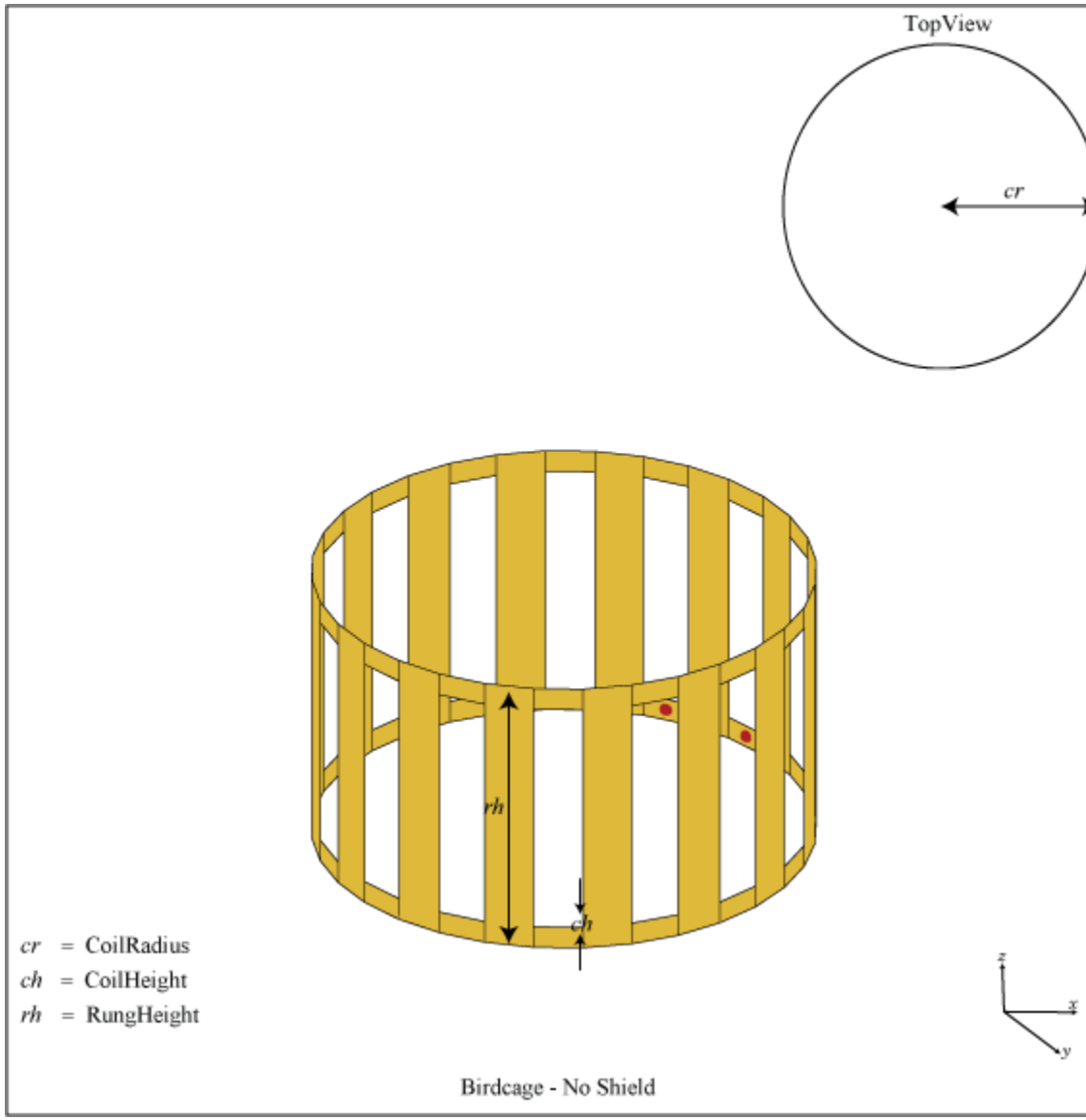
birdcage

Creates birdcage (MRI coil)

Description

The `birdcage` object creates to create a birdcage MRI coil. This antenna is most commonly used in clinical MRI. The antenna structure consists of two circular coils connected by conductive elements called `rungs`. The number of rungs depends on the size of the coil and is generally an even number.

The coil is operated at 64 MHz or 128 MHz. The birdcage can be loaded/excited to model a highpass or lowpass coil.


TopView
cr
rh
ch
cr = CoilRadius
ch = CoilHeight
rh = RungHeight
z
x
y
Birdcage - No Shield

Creation

Syntax

```
bc = birdcage
bc = birdcage(Name,Value)
```

Description

`bc = birdcage` creates a birdcage antenna to model an MRI coil.

`bc = birdcage(Name,Value)` sets properties using one or more name-value pairs. For example, `bc = birdcage('NumRungs',8)` creates a birdcage with eight rungs. Enclose each property name in quotes.

Properties

`NumRungs` — Number of rungs
`16` (default) | scalar integer

Number of rungs, specified as a scalar.

Example: `'NumRungs',20`

Example: `bc.NumRungs = 20`

Data Types: `int8`

`CoilRadius` — Coil radius
`0.4000` (default) | scalar

Coil radius, specified as a scalar in meters.

Example: `'CoilRadius',0.2`

Example: `bc.CoilRadius = 0.2`

Data Types:

`CoilHeight` — Coil height
`0.0400` (default) | scalar

Coil height, specified as a scalar in meters.

Example: `'CoilHeight',0.089`

Example: `bc.CoilHeight = 0.089`

Data Types: `double`

`RungHeight` — Height of rungs
`0.4600` (default) | scalar

Height of rungs, specified as a scalar in meters. Distance is measured from the middle of the upper coil to the middle of the lower coil.

Example: `'RungHeight',0.56`

Example: `bc.RungHeight = 0.56`

Data Types: `double`

`ShieldRadius` — Shield radius
`0` (default) | scalar

Shield radius, specified as a scalar in meters. A value of zero indicates that the shield is absent.

Example: `'ShieldRadius',0.2`

Example: `bc.ShieldRadius = 0.2`

Data Types: `double`

`ShieldHeight` — Shield height
`0` (default) | scalar

Shield height, specified as a scalar in meters. A value of zero indicates that the shield is absent.

Example: `'ShieldHeight',0.089`

Example: `bc.ShieldHeight = 0.089`

Data Types: `double`

`Phantom` — Dielectric mesh to load birdcage
structure

Dielectric mesh to load birdcage, specified as a structure having the following fields:

`Points` — Points in custom dielectric mesh
N-by-3 matrix

Points in custom dielectric mesh, specified as an *N*-by-3 matrix in meters. *N* is the number of points.

You can use the phantom property to insert a dielectric mesh in the shape of a human head into the bird cage antenna. This dielectric cylinder has a permeability of 80. You can upload this mesh in the form of a mat file.

Data Types: `double`

`Tetrahedra` — Tetrahedra in custom dielectric mesh
M-by-3 integer matrix

Tetrahedra in custom dielectric mesh, specified as an *M*-by-3 integer matrix. *M* is the number of tetrahedra.

Data Types: `double`

`EpsilonR` — Relative permittivity of dielectric material
scalar

Relative permittivity of dielectric material, specified as a scalar.

Data Types: `double`

`LossTangent` — Loss in dielectric material
scalar

Loss in dielectric material, specified as a scalar.

Data Types: `double`

Data Types: `struct`

`FeedLocations` — Location of feeds in Cartesian coordinates
`0` (default) | *N*-by-3 matrix

Location of feeds in Cartesian coordinates, specified as an *N*-by-3 matrix. You can also use the `getLowPassLocs` and `getHighPassLocs` functions to determine the feed locations in low-pass or high-pass mode.

Example: `'FeedLocations'= [0.3981 0.0392 -0.2300;0.3528 0.1886 -0.2300]`

Example: `b.FeedLocations = getLowPassLocs(b)`

Data Types: `double`

`FeedVoltage` — Magnitude of voltage
`1` (default) | scalar | 1-by-*N* vector

Magnitude of voltage applied to each feed, specified as a scalar or 1-by-*N* vector with each element unit in volts.

Example: `'FeedVoltage',2`

Example: `bc.FeedVoltage = 2`

Data Types: `double`

`FeedPhase` — Phase shift to the voltage
`0` (default) | scalar | 1-by-*M* vector

Phase shift to the excitation voltage at each feed, specified as a scalar or 1-by-*M* vector with each element unit in degrees.

Example: `'FeedPhase',45`

Example: `bc.FeedPhase = 45`

Data Types: `double`

`Load` — Lumped elements
[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. You can add a load anywhere on the surface of the antenna. By default, it is at the origin. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `bc.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector in degrees.

Example: `'Tilt',90`

Example: `bc.Tilt = [90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

getLowPassLocs	Feeding location to operate birdcage as lowpass coil
getHighPassLocs	Feeding location to operate birdcage as highpass coil
show	Display antenna or array structure; Display shape as filled patch
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure

meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

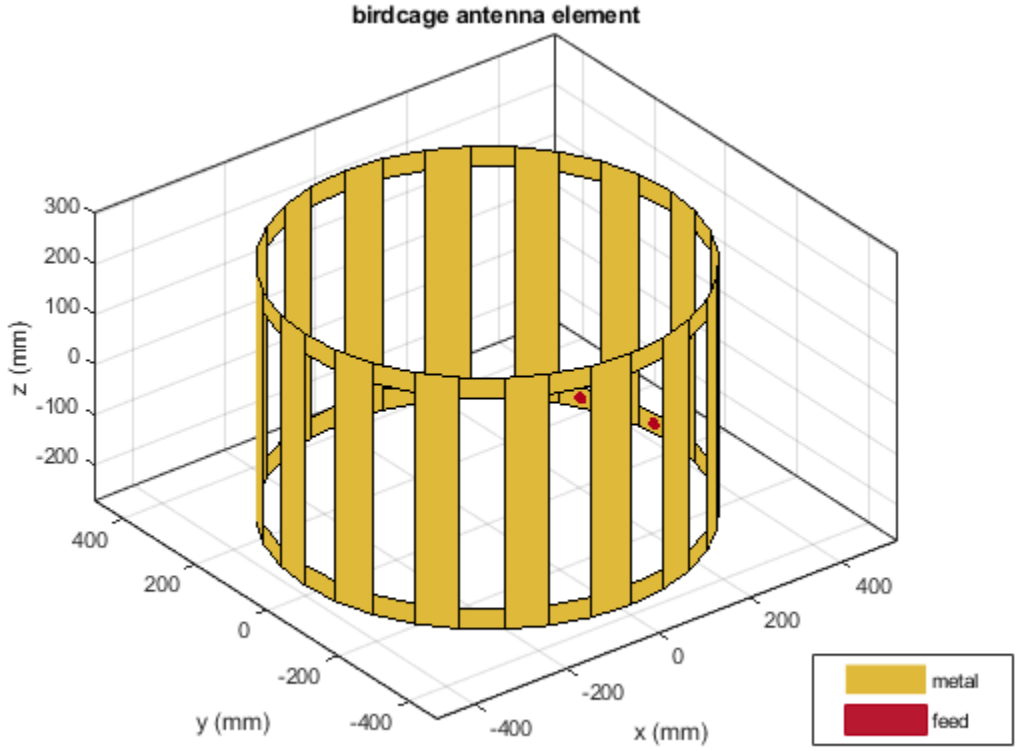
Birdcage Antenna

Create and view a default birdcage antenna.

```
bc = birdcage
```

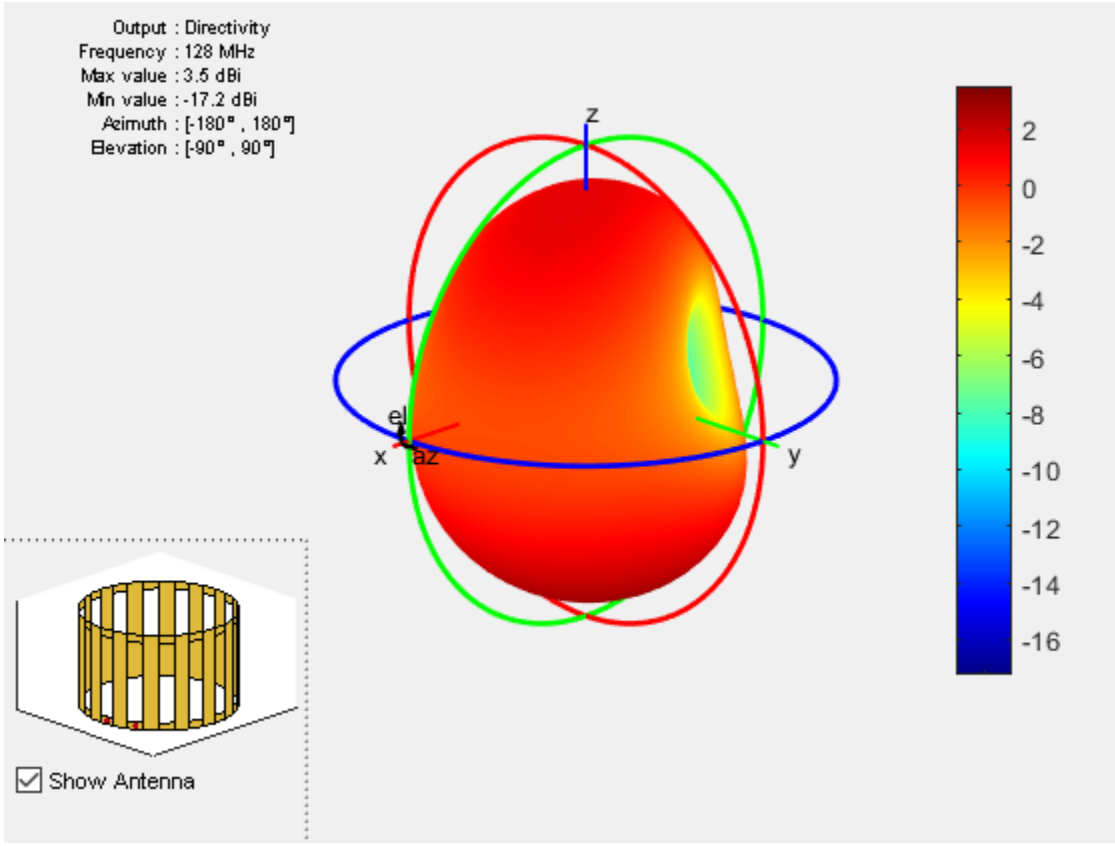
```
bc =
  birdcage with properties:

         NumRungs: 16
       CoilRadius: 0.4000
       CoilHeight: 0.0400
       RungHeight: 0.4600
     ShieldRadius: 0
     ShieldHeight: 0
          Phantom: []
    FeedLocations: [2x3 double]
      FeedVoltage: 1
        FeedPhase: 0
             Tilt: 0
         TiltAxis: [1 0 0]
             Load: [1x1 lumpedElement]
```

```
show(bc);
```



Plot the radiation pattern at 128 MHz.

```
pattern(bc,128e6)
```



Human Head Model Inside BirdCage

Antenna Toolbox™ provides two `.mat` files to load a phantom human head model into a birdcage antenna. The humanheadcoarse.mat contains a coarse dielectric mesh of the human head model and the humanheadfine.mat provides the user with a finer dielectric mesh. Load the coarse human head model.

Load human head model file. Extract the values of `Points` and `Tetrahedra`. Add a relative permittivity (EpsilonR) of 10 and a dielectric loss (LossTangent) of 0.002. Scale

the dielectric mesh to fit in the birdcage antenna. In this case, the mesh points are multiplied by 0.003.

```
load humanheadcoarse.mat
humanhead = struct('Points',0.003*P,'Tetrahedra',T,'EpsilonR',10,...
                   'LossTangent',0.002)
```

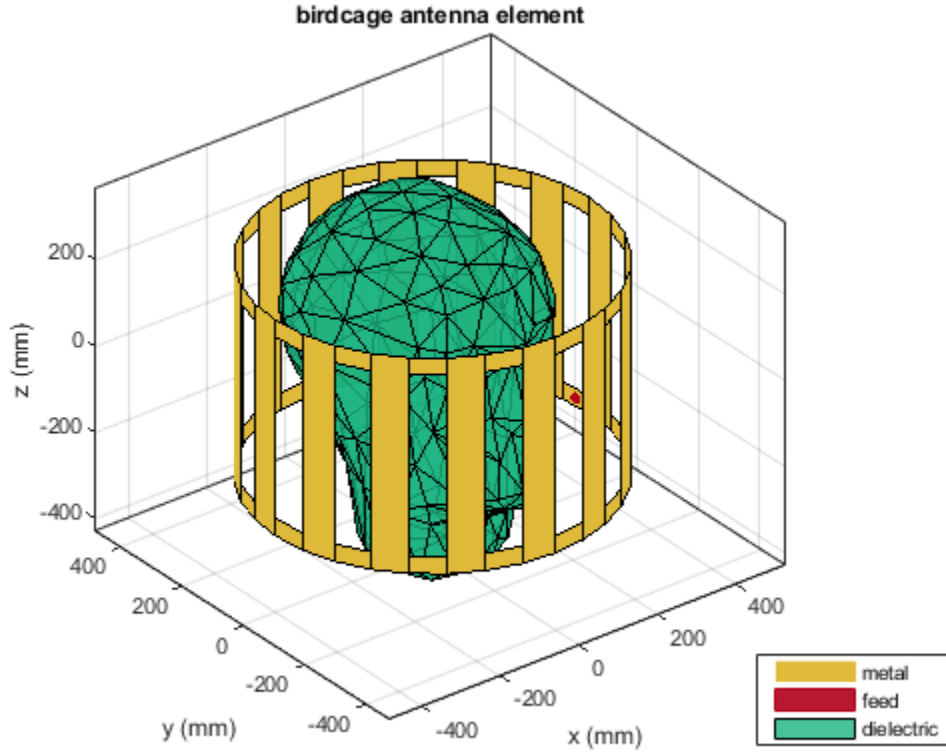
```
humanhead = struct with fields:
         Points: [584x3 double]
     Tetrahedra: [2818x4 double]
       EpsilonR: 10
    LossTangent: 0.0020
```

Add and view the human head mesh inside the birdcage.

```
b = birdcage('Phantom',humanhead)
```

```
b =
  birdcage with properties:

         NumRungs: 16
       CoilRadius: 0.4000
       CoilHeight: 0.0400
       RungHeight: 0.4600
     ShieldRadius: 0
     ShieldHeight: 0
          Phantom: [1x1 struct]
    FeedLocations: [2x3 double]
      FeedVoltage: 1
        FeedPhase: 0
             Tilt: 0
         TiltAxis: [1 0 0]
             Load: [1x1 lumpedElement]
```

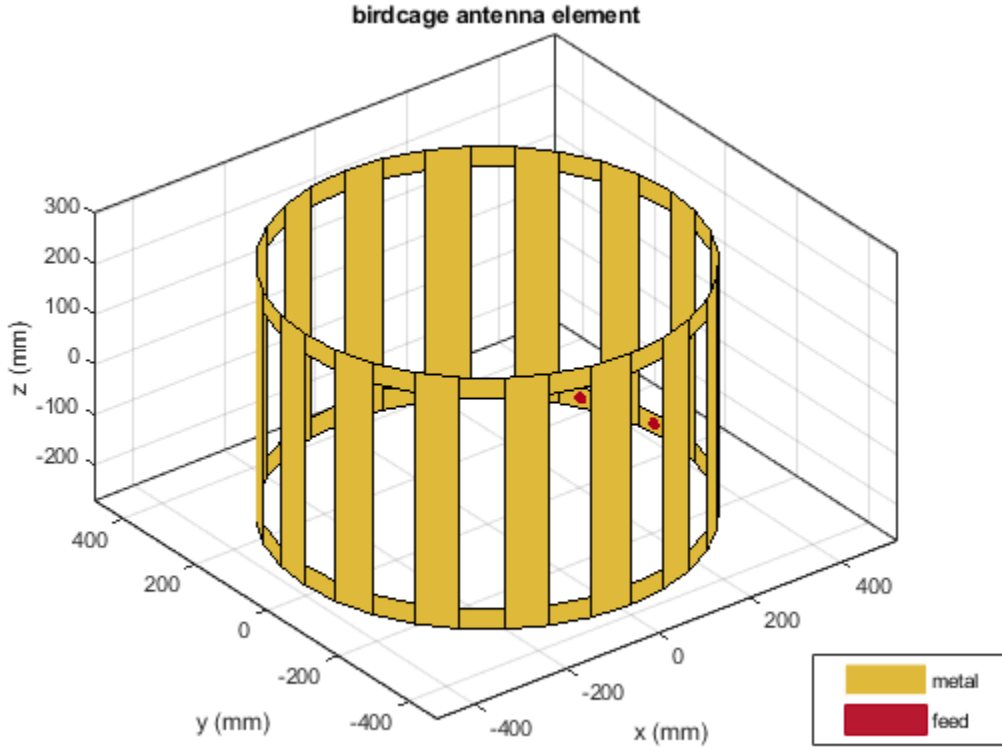
```
show(b)
```



Birdcage In High-Pass Operation

Create a birdcage antenna.

```
b = birdcage;
show(b);
```

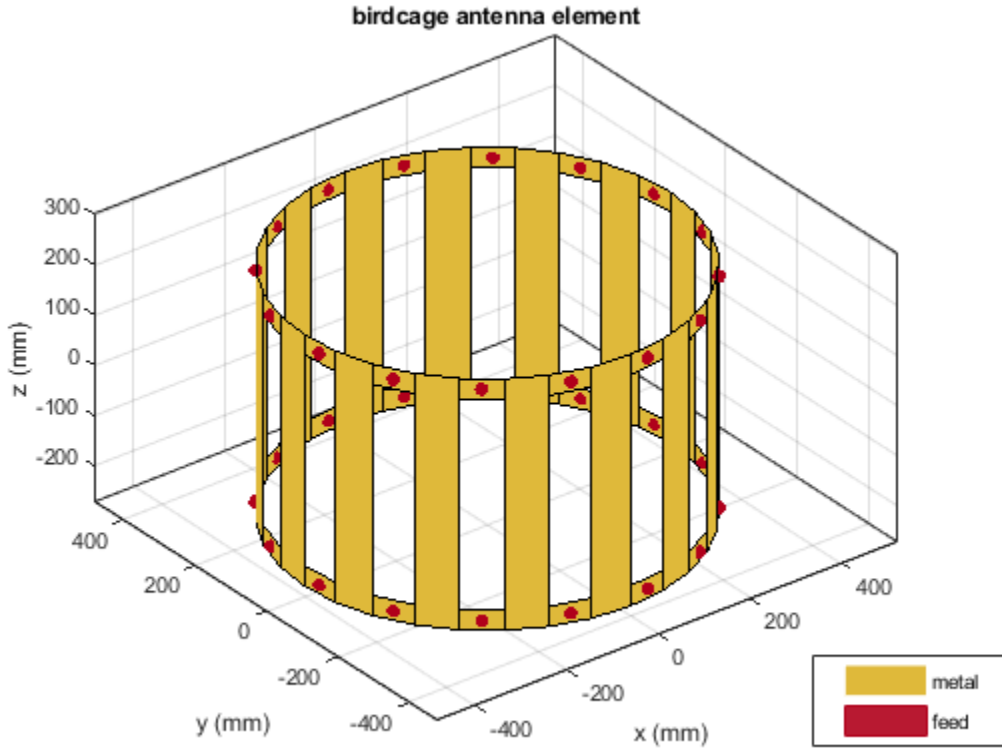


Use the birdcage as a high-pass coil.

```
b.FeedLocations = getHighPassLocs(b)
show(b);
```

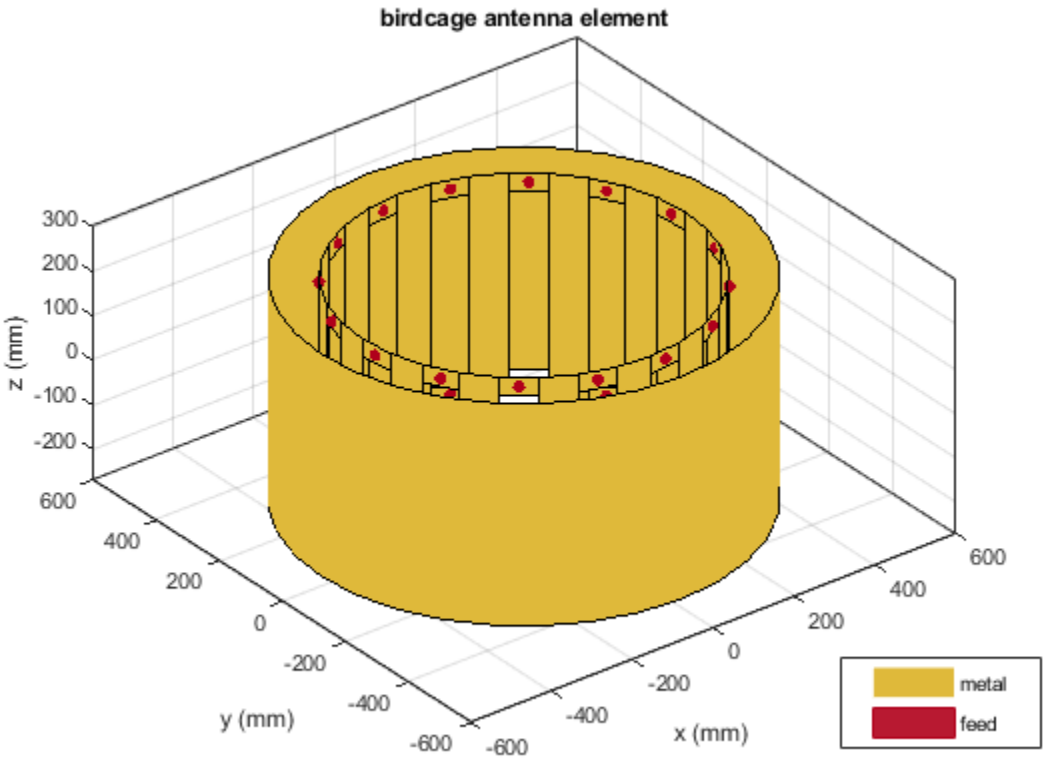
```
b =

  birdcage with properties:

        NumRungs: 16
      CoilRadius: 0.4000
      CoilHeight: 0.0400
      RungHeight: 0.4600
```

```
     ShieldRadius: 0
     ShieldHeight: 0
          Phantom: []
    FeedLocations: [32x3 double]
      FeedVoltage: 1
        FeedPhase: 0
             Tilt: 0
         TiltAxis: [1 0 0]
             Load: [1x1 lumpedElement]
```



Shield the antenna to ensure that radiation does not leak out.

```
b.ShieldRadius = 0.5;
b.ShieldHeight = 0.5;
show(b) ;
```



See Also

`dipole` | `loopCircular`

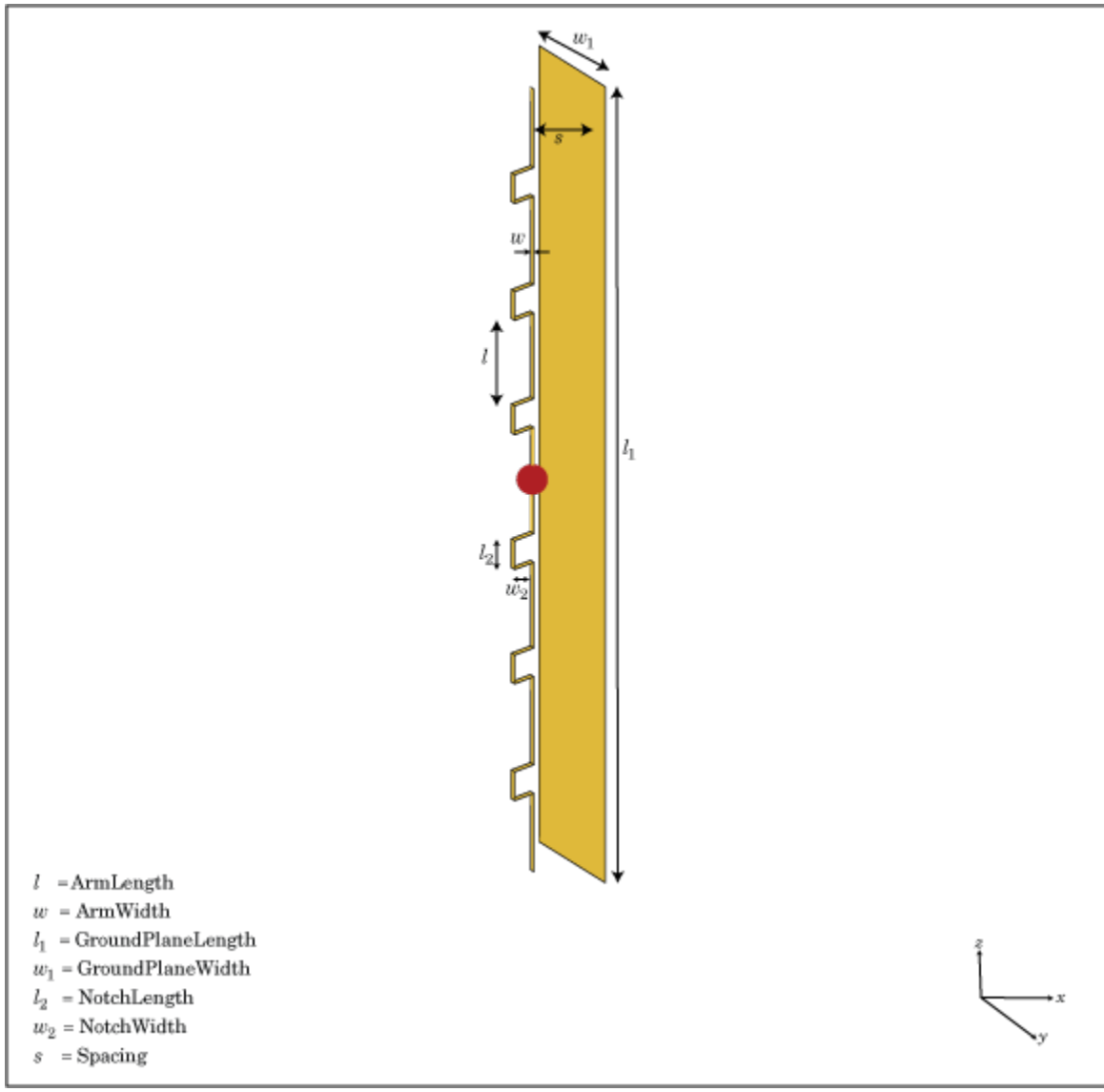
Introduced in R2017b

sectorInvertedAmos

Create inverted Amos sector antenna

Description

Use the `sectorInvertedAmos` object to create an inverted Amos sector antenna consisting of four dipole-like arms. The antenna is fed at the origin of the dipole. The dipole arms are symmetric about the origin. The operating frequency of the antenna is at 2.45 GHz wireless.


w_1
s
w
l
l_1
l_2
w_2
l = ArmLength
w = ArmWidth
l_1 = GroundPlaneLength
w_1 = GroundPlaneWidth
l_2 = NotchLength
w_2 = NotchWidth
s = Spacing
z
x
y

Creation

Syntax

```
amossector = sectorInvertedAmos
amossector = sectorInvertedAmos(Name,Value)
```

Description

`amossector = sectorInvertedAmos` creates an inverted Amos sector antenna with four dipole-like arms.

`amossector = sectorInvertedAmos(Name,Value)` sets properties using one or more name-value pair. For example, `amossector = sectorInvertedAmos('ArmWidth',0.2)` creates an inverted Amos sector with a dipole width of 0.2 m. Enclose each property name in quotes.

Properties

`ArmLength` — Individual dipole arm length
`[0.0880 0.0710 0.0730 0.0650]` (default) | vector

Length of individual dipole arms, specified as a vector with each element unit in meters.

Example: `'ArmLength',[0.0980 0.0810 0.0830 0.0750]`

Example: `amossector.ArmLength = [0.0980 0.0810 0.0830 0.0750]`

Data Types: `double`

`ArmWidth` — Dipole arm width
`0.0040` (default) | scalar

Width of dipole arms, specified as a scalar in meters.

Example: `'ArmWidth',0.0025`

Example: `amossector.ArmWidth = 0.0025`

Data Types: `double`

`NotchLength` — Notch length
`0.0238` (default) | scalar

Notch length, specified as a scalar in meters. For an inverted Amos sector antenna with seven stacked arms, six notches are generated. Notch length is measured along the length of the antennas.

Example: `'NotchLength',0.001`

Example: `amossector.NotchLength = 0.001`

Data Types: `double`

`NotchWidth` — Notch width
`0.0170` (default) | scalar

Notch width, specified as a scalar in meters. For an inverted Amos sector antenna with seven stacked arms, six notches are generated. Notch width is measured perpendicular to the length of the antenna.

Example: `'NotchWidth',0.00190`

Example: `amossector.NotchWidth = 0.00190`

Data Types: `double`

`GroundPlaneLength` — Ground plane length
0.6600 (default) | scalar

Ground plane length, specified as a scalar in meters. By default, ground plane length is measured along x-axis.

Example: `'GroundPlaneLength',0.7500`

Example: `amossector.GroundPlaneLength = 0.7500`

Data Types: `double`

`GroundPlaneWidth` — Ground plane width
0.0750 (default) | scalar

Ground plane width, specified as a scalar in meters. By default, ground plane width is measured along y-axis.

Example: `'GroundPlaneWidth',0.0500`

Example: `amossector.GroundPlaneWidth = 0.0500`

Data Types: `double`

`Spacing` — Distance between ground plane and antenna element
0.0355 (default) | scalar

Distance between ground plane and antenna element, specified as a scalar in meters.

Example: `'Spacing',0.0355`

Example: `amossector.Spacing = 0.0355`

Data Types: `double`

`Load` — Lumped elements
[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. You can add a load anywhere on the surface of the antenna. By default, it is at the origin. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `amossector.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector in degrees.

Example: `'Tilt',90`

Example: `amossector.Tilt= [90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.

- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

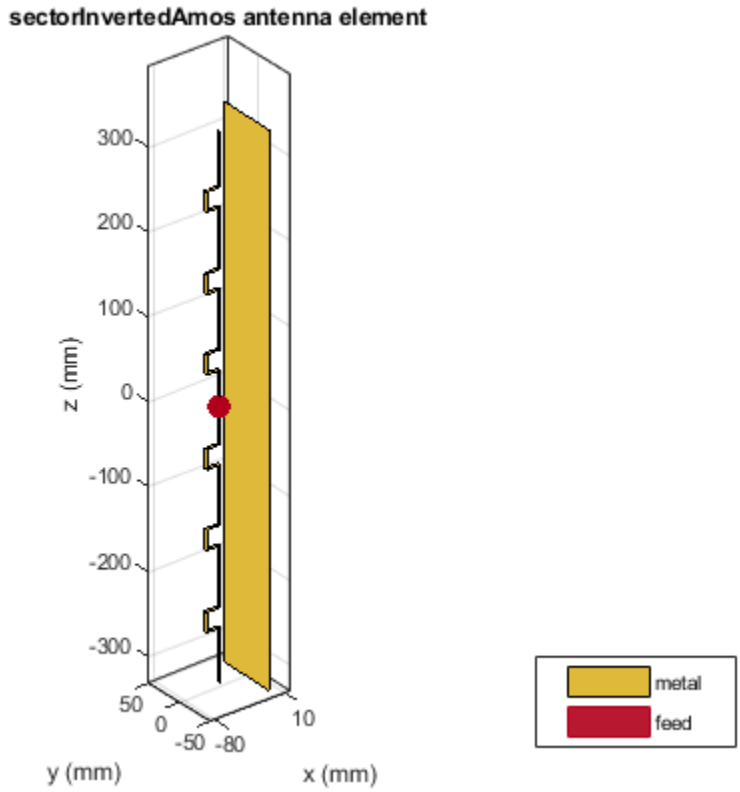
show	Display antenna or array structure; Display shape as filled patch
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

Inverted Amos Sector

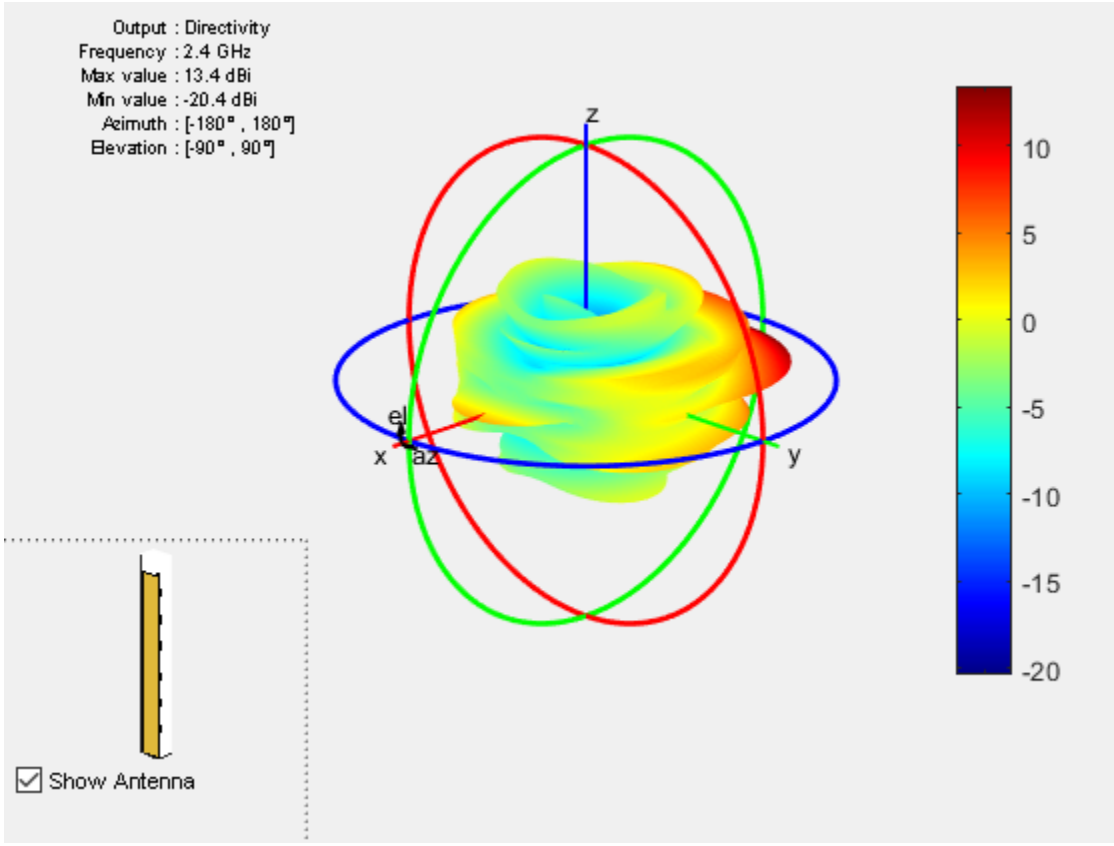
Create and view an inverted Amos sector antenna.

```
sectoria = sectorInvertedAmos

sectoria =
  sectorInvertedAmos with properties:

            ArmLength: [0.0880 0.0710 0.0730 0.0650]
             ArmWidth: 0.0040
           NotchLength: 0.0238
            NotchWidth: 0.0170
    GroundPlaneLength: 0.6600
     GroundPlaneWidth: 0.0750
              Spacing: 0.0355
                 Tilt: 0
             TiltAxis: [1 0 0]
                 Load: [1x1 lumpedElement]


show(sectoria)
```



Plot Radiation Pattern at 2.4 GHz

```
pattern(sectoria,2.4e9)
```



See Also

`dipoleMeander` | `reflector`

Introduced in R2017b

rxsite

Create radio frequency receiver site

Description

Use the `rxsite` object to create a radio frequency receiver site.

Creation

Syntax

```
rx = rxsite
rx = rxsite(Name,Value)
```

Description

`rx = rxsite` creates a radio frequency receiver site.

`rx = rxsite(Name,Value)` sets properties using one or more name-value pairs. For example, `rx = rxsite('Name','RX Site')` creates a receiver site with name `RX Site`. Enclose each property name in quotes.

Properties

`Name` — Site name
character vector | string | row or column vector

Site name, specified as a character vector or as a row or column vector or as a string.

Example: `'Name','Site 3'`

Example: `RX.Name = 'Site 3'`

Example: If you want to assign multiple values then - `names = ["Fenway Park","Faneuil Hall","Bunker Hill Monument"]; RX = rxsite('Name',names)`

Data Types: `char` | `string`

`Latitude` — Site latitude coordinates

`42.3021` (default) | numeric scalar | row or column vector

Site latitude coordinates, specified as a numeric scalar or a row or column vector in the range of range `-90` to `90`. Coordinates are defined using Earth ellipsoid model WGS-84. Latitude is the north/south angle.

Example: `'Latitude',45.098`

Example: `RX = 45.098`

Example: If you want to assign multiple values then - `latitude = [42.3467,42.3598,42.3763]; RX = rxsite('Latitude',latitude)`

`Longitude` — Site longitude coordinates

`-71.3764` (default) | numeric scalar | row or column vector

Site longitude coordinates, specified as a numeric scalar or a row or column vector. Coordinates are defined using Earth ellipsoid model WGS-84. Longitude is the east/west angle.

Example: `'Longitude',-68.890`

Example: `RX.Longitude = -68.890`

Example: If you want to assign multiple values then - `longitude = [-71.0972,-71.0545,-71.0611]; RX = rxsite('Longitude',longitude)`

`Antenna` — Antenna element or array

`dipole` (default) | object | row vector

Antenna element or array object, specified as an object or a row vector of objects. By default, the antenna is a dipole designed for 1.9 GHz.

Example: `'Antenna',monopole`

Example: `RX.Antenna = monopole`

Data Types: `double`

`AntennaAngle` — Antenna x-axis angle

`0` (default) | numeric scalar | 2-by-1 vector | 2-by-*N* matrix

Antenna x-axis angle, specified as a numeric scalar, a 2-by-1 vector, or a 2-by-*N* matrix in degrees.

The numeric scalar is the azimuth angle measured counterclockwise from the east to the antenna x-axis.

In the 2-by-1 vector, the first element is the azimuth angle and the second element elevation angle. The elevation angle measures from the horizontal plane to antenna x-axis from `-90` to `90` degrees.

Example: `'AntennaAngle',25`

Example: `RX.AntennaAngle = [25,-80]`

Data Types: `double`

`AntennaHeight` — Antenna height

`1` (default) | nonnegative numeric scalar | row vector

Antenna height from the ground elevation at the site to the center of the antenna element, specified as a nonnegative numeric scalar in meters. Maximum value for this property is 6,371,000 m.

Example: `'AntennaHeight',25`

Example: `RX.AntennaHeight = 15`

Data Types:

`SystemLoss` — System loss

`0` (default) | nonnegative numeric scalar | row vector

System loss, specified as a non-negative numeric scalar or a row vector in dB.

System loss includes transmission line loss and any other miscellaneous system losses.

Example: `'SystemLoss',10`

Example: `RX.SystemLoss = 10`

Data Types:

`ReceiverSensitivity` — Minimum received power to detect signal

`-100` (default) | numeric scalar | row vector

Minimum received power to detect the signal, specified as a numeric scalar or a row vector in dBm.

Example: `'ReceiverSensitivity',-80`

Example: `RX.ReceiverSensitivity = -80`

Data Types: `double`

Object Functions

show	Show site location on map
hide	Hide site location on map
distance	Distance between sites
angle	Angle between sites
elevation	Ground elevation of site
location	Location coordinates at a given distance and angle from site
sigstrength	Signal strength due to transmitter
los	Plot or compute the line-of-sight (LOS) visibility between sites on a map
link	Display communication link on map
pattern	Plot antenna radiation pattern on map

Examples

Default Receiver Site

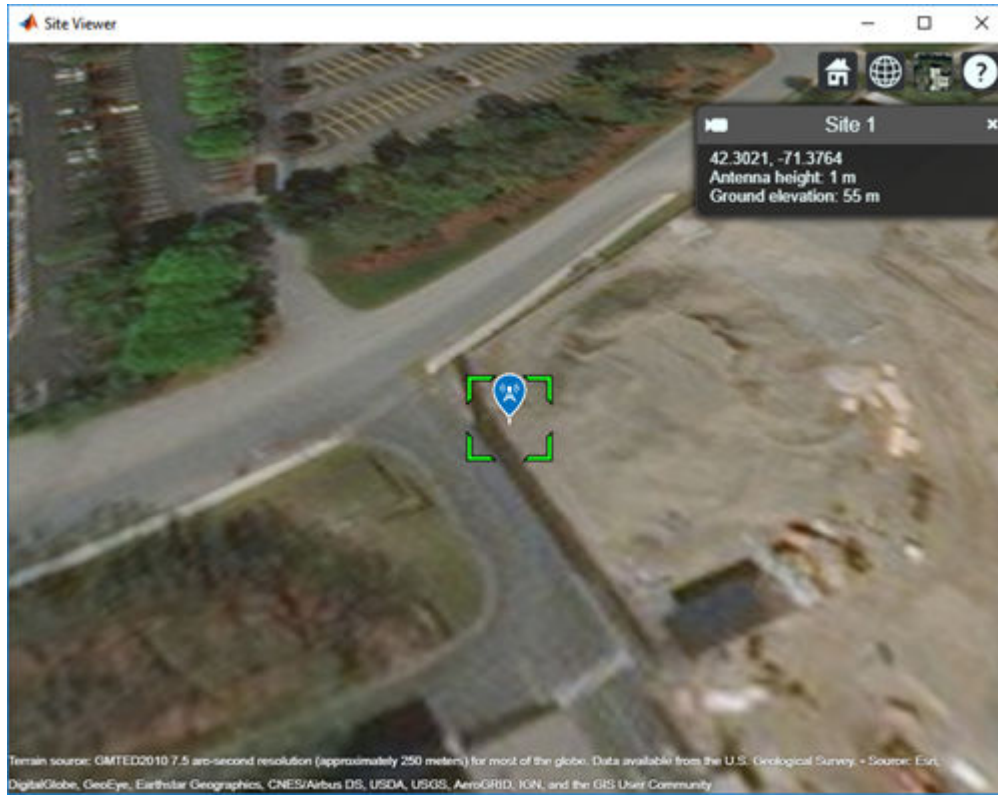
Create and show the default receiver site.

```
rx = rxsite
```

```
rx =
  rxsite with properties:

                  Name: 'Site 2'
              Latitude: 42.3021
             Longitude: -71.3764
               Antenna: [1×1 dipole]
          AntennaAngle: 0
         AntennaHeight: 1
            SystemLoss: 0
```

```
    ReceiverSensitivity: -100
```

```
show(rx)
```



Receiver Array Site

Create and show a 1-by-3 receiver site array.

Define names and locations of the sites around Boston.

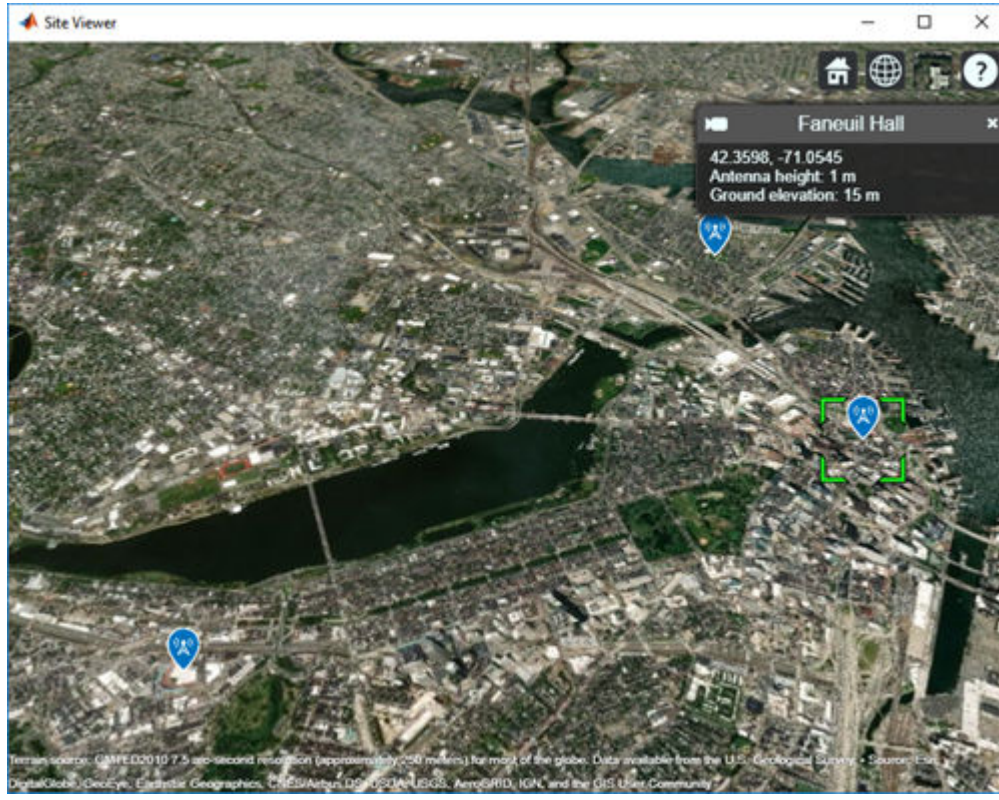
```
names = ["Fenway Park","Faneuil Hall","Bunker Hill Monument"];
lats = [42.3467,42.3598,42.3763];
lons = [-71.0972,-71.0545,-71.0611];
```

Define the sensitivity of the receivers.

```
sens = -90;
```

Create and show receiver site array.

```
rxs = rxsite('Name', names,...
       'Latitude',lats,...
       'Longitude',lons, ...
       'ReceiverSensitivity',sens);
show(rxs)
```



See Also

`txsite`

Introduced in R2017b

antenna.Circle

Create circle centered at origin on X-Y plane

Description

Use the `antenna.Circle` object to create a circle centered at the origin and on the X-Y plane. You can use `antenna.Polygon` to create single-layer or multi-layered antennas using `pcbStack`.

Creation

Syntax

```
circle = antenna.Circle
circle = antenna.Circle(Name,Value)
```

Description

`circle = antenna.Circle` creates a circle centered at the origin and on the X-Y plane.

`circle = antenna.Circle(Name,Value)` sets properties using one or more name-value pair. For example, `circle = antenna.Circle('Radius',0.2)` creates a circle of radius 0.2 m. Enclose each property name in quotes.

Properties

Name — Name of circle
`'mycircle'` (default) | character vector

Name of circle, specified a character vector.

Example: `'Name','Circle1'`

Example: `circle.Name= 'Circle1'`

Data Types: `char` | `string`

`Center` — Cartesian coordinates of center of circle

`[ 0 0 ]` (default) | 2-element vector

Cartesian coordinates of center of circle, specified a 2-element vector with each element measured in meters.

Example: `'Center',[0.006 0.006]`

Example: `circle.Center= [0.006 0.006]`

Data Types: `double`

`Radius` — Circle radius

1 (default) | scalar

Circle radius, specified a scalar in meters.

Example: `'Radius',2`

Example: `circle.Radius= 2`

Data Types: `double`

`NumPoints` — Number of discretization points on circumference

20 (default) | scalar

Number of discretization points on circumference, specified a scalar.

Example: `'NumPoints',16`

Example: `circle.NumPoints= 2`

Data Types: `double`

Object Functions

add	Boolean unite operation on two shapes
subtract	Boolean subtraction operation on two shapes
area	Calculate area of shape in sq.m
intersect	Boolean intersection operation on two shapes
rotate	Rotate shape about axis and angle

rotateX	Rotate shape about X-axis and angle
rotateY	Rotate shape about Y-axis and angle
rotateZ	Rotate shape about Z-axis and angle
translate	Move shape to new location
show	Display antenna or array structure; Display shape as filled patch
mesh	Mesh properties of metal or dielectric antenna or array structure

Examples

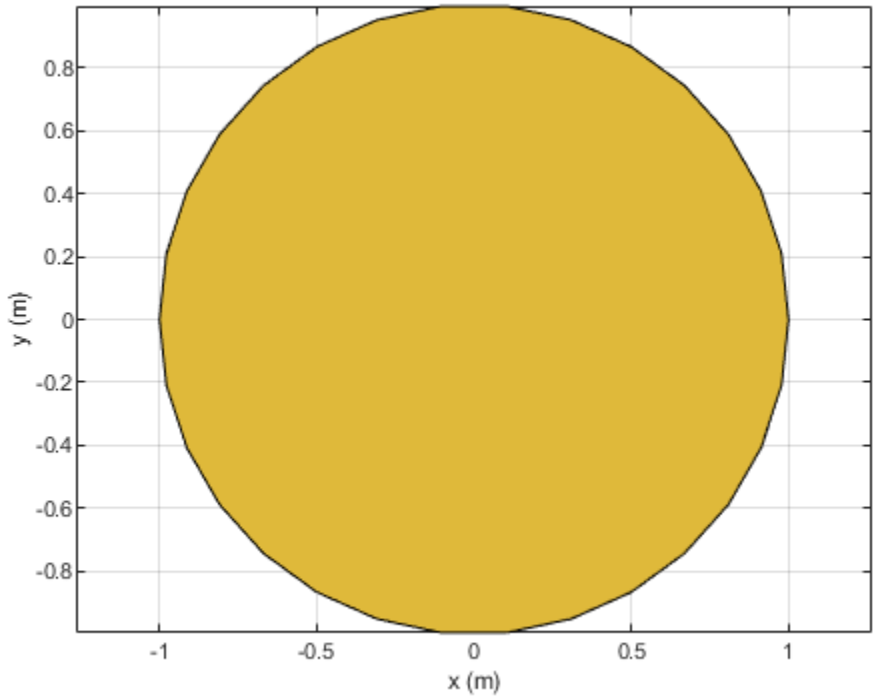
Create Circle with Default Properties

Create and view circle using `antenna.Circle` and view it.

```
c1 = antenna.Circle
```

```
c1 =
  Circle with properties:

         Name: 'mycircle'
       Center: [0 0]
       Radius: 1
    NumPoints: 30
```

```
show(c1)
```



Create Circle with Specified Properties

Create a circle with a radius of 4 m.

```
c2 = antenna.Circle('Radius',4)
```

```
c2 =
  Circle with properties:

        Name: 'mycircle'
      Center: [0 0]
      Radius: 4
```

```
    NumPoints: 30
```

Add Two Shapes

Create circle with a radius of 1 m. The center of the circle is at [1 0].

```
circle1 = antenna.Circle('Center',[1 0],'Radius',1);
```

Create a rectangle with a length of 2 m and a width of 4 m centered at the origin.

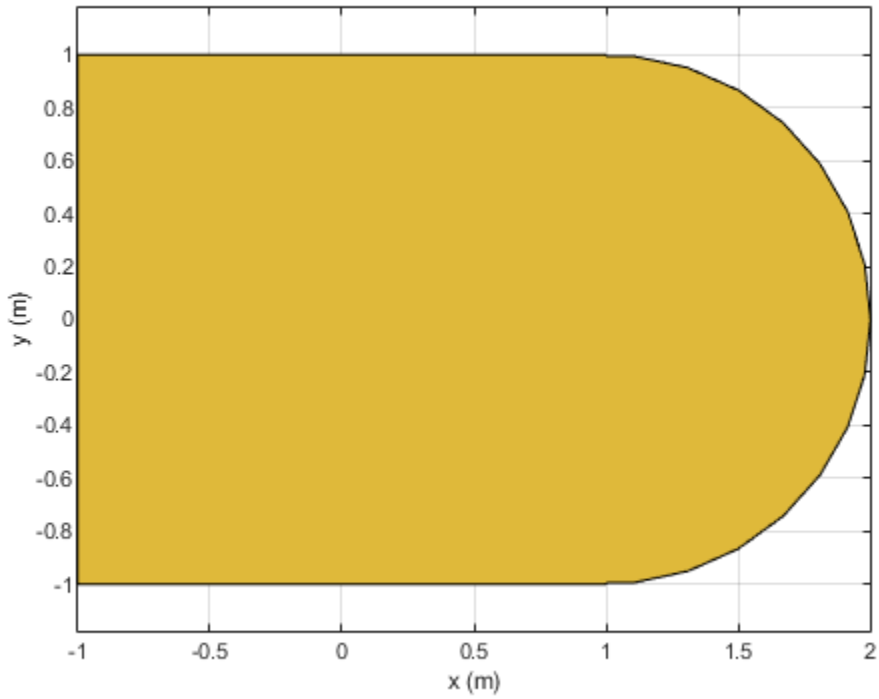
```
rect1 = antenna.Rectangle('Length',2,'Width',2);
```

Add the two shapes together using the + function.

```
polygon1 = circle1+rect1
```

```
polygon1 =
  Polygon with properties:

        Name: 'mypolygon'
    Vertices: [21x3 double]
```

```
show(polygon1)
```



See Also

`antenna.Polygon` | `antenna.Rectangle`

Introduced in R2017a

antenna.Polygon

Create polygon on X-Y plane

Description

Use the `antenna.Polygon` object to create a polygonal shape centered at the origin and on the X-Y plane. You can use `antenna.Polygon` to create single-layer or multilayered antennas using `pcbStack`.

Creation

Syntax

```
polygon = antenna.Polygon
polygon = antenna.Polygon(Name,Value)
```

Description

`polygon = antenna.Polygon` creates a polygonal shape centered at the origin and on the X-Y plane.

`polygon = antenna.Polygon(Name,Value)` sets properties using one or more name-value pair. For example, `polygon = antenna.Polygon('Name','mypolygon')` creates a polygon of name `'mypolygon'`. Enclose each property name in quotes.

Properties

`Name` — Name of polygon
`'mypolygon'` (default) | character vector

Name of polygon, specified a character vector.

Example: `'Name','Polygon1'`

Example: `polygon.Name = 'Polygon1'`

Data Types: `char` | `string`

`Vertices` — Cartesian coordinates of polygon vertices
[ ] (default) | *N*-by-3 vector

Cartesian coordinates of polygon vertices, specified as a *N*-by-3 vector with each element measured in meters.

Example: `'Vertices',[-1 0 0;-0.5 0.2 0;0 0 0]`

Example: `polygon.Vertices = [-1 0 0;-0.5 0.2 0;0 0 0]`

Data Types: `double`

Object Functions

add	Boolean unite operation on two shapes
area	Calculate area of shape in sq.m
subtract	Boolean subtraction operation on two shapes
intersect	Boolean intersection operation on two shapes
rotate	Rotate shape about axis and angle
rotateX	Rotate shape about X-axis and angle
rotateY	Rotate shape about Y-axis and angle
rotateZ	Rotate shape about Z-axis and angle
translate	Move shape to new location
show	Display antenna or array structure; Display shape as filled patch
mesh	Mesh properties of metal or dielectric antenna or array structure

Examples

Create and Transform Polygon

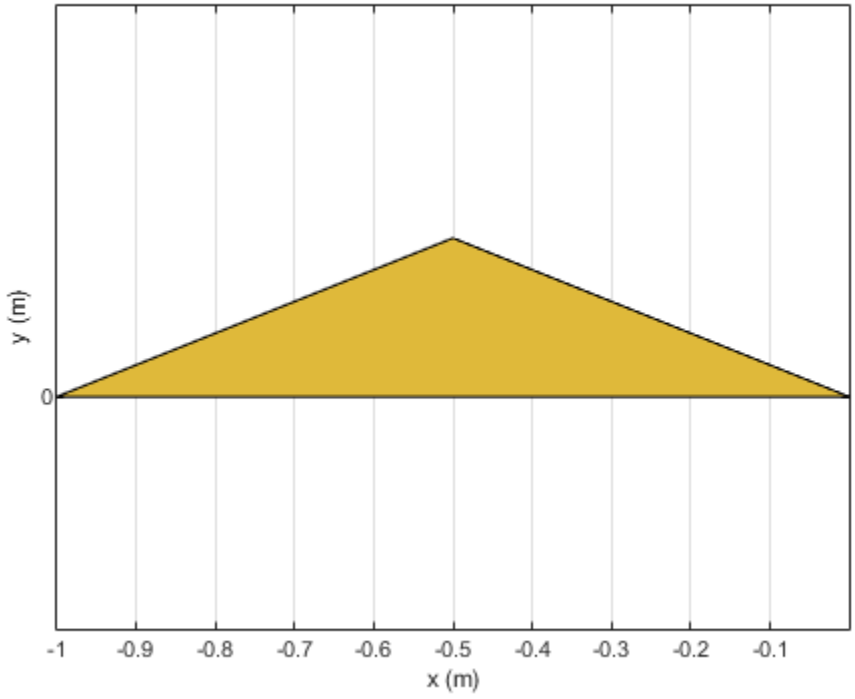
Create a polygon using `antenna.Polygon` with vertices at `[-1 0 0;-0.5 0.2 0;0 0 0]` and view it.

```
p = antenna.Polygon('Vertices', [-1 0 0;-0.5 0.2 0;0 0 0])
```

```
p =
  Polygon with properties:
```

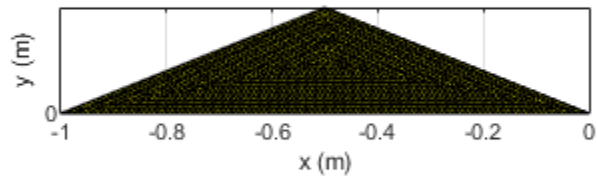
```
        Name: 'mypolygon'
    Vertices: [3x3 double]
```

```
show(p)
axis equal
```



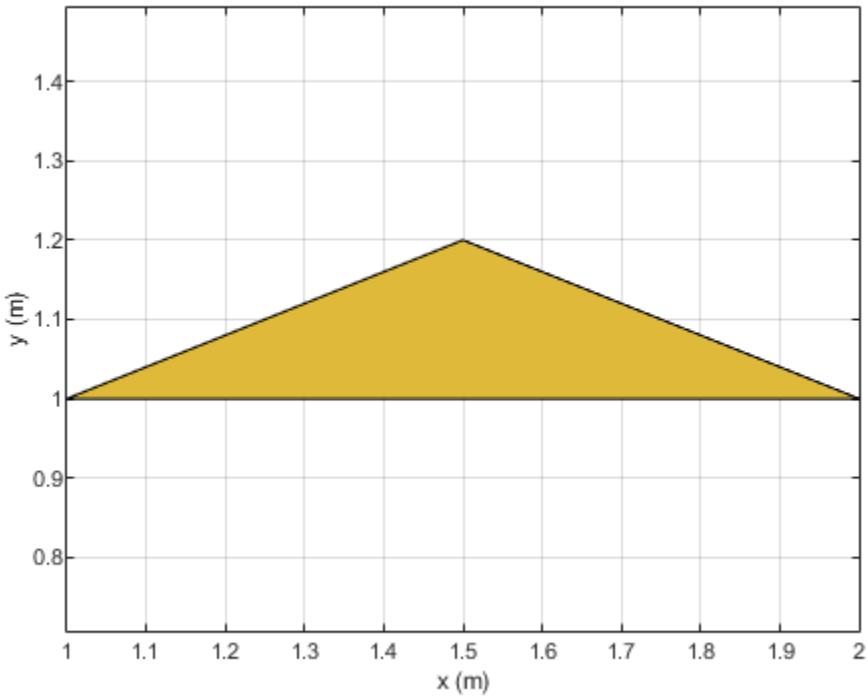
Mesh the polygon and view it.

```
mesh(p,0.2)
```



Move the polygon to a new location on the X-Y plane.

```
translate(p,[2,1,0])
axis equal
```



See Also

`antenna.Circle` | `antenna.Rectangle`

Introduced in R2017a

antenna.Rectangle

Create rectangle centered at origin on X-Y plane

Description

Use the `antenna.Rectangle` object to create a rectangle centered at the origin and on the X-Y plane. You can use `antenna.Polygon` to create single-layer or multi-layered antennas using `pcbStack`.

Creation

Syntax

```
rect = antenna.Rectangle
rect = antenna.Rectangle(Name,Value)
```

Description

`rect = antenna.Rectangle` creates a rectangle centered at the origin and on the X-Y plane.

`rect = antenna.Rectangle(Name,Value)` sets properties using one or more name-value pair. For example, `rectangle = antenna.Rectangle('Length',0.2)` creates a rectangle of length 0.2 m. Enclose each property name in quotes.

Properties

`Name` — Name of rectangle
`'myrectangle'` (default) | character vector

Name of rectangle, specified a character vector.

Example: `'Name','Rect1'`

Example: `rectangle.Name = 'Rect1'`

Data Types: `char` | `string`

`Center` — Cartesian coordinates of center of rectangle
`[ 0 0 ]` (default) | 2-element vector

Cartesian coordinates of center of rectangle, specified a 2-element vector with each element measured in meters.

Example: `'Center',[0.006 0.006]`

Example: `rectangle.Center = [0.006 0.006]`

Data Types: `double`

`Length` — Rectangle length
1 (default) | scalar

Rectangle length, specified a scalar in meters.

Example: `'Length',2`

Example: `rectangle.Length = 2`

Data Types: `double`

`Width` — Rectangle width
2 (default) | scalar

Rectangle width, specified a scalar in meters.

Example: `'Width',4`

Example: `rectangle.Width = 4`

Data Types: `double`

`NumPoints` — Number of discretization points per side
2 (default) | scalar

Number of discretization points per side, specified a scalar.

Example: `'NumPoints',16`

Example: `rectangle.NumPoints = 16`

Data Types: `double`

Object Functions

add	Boolean unite operation on two shapes
area	Calculate area of shape in sq.m
subtract	Boolean subtraction operation on two shapes
intersect	Boolean intersection operation on two shapes
rotate	Rotate shape about axis and angle
rotateX	Rotate shape about X-axis and angle
rotateY	Rotate shape about Y-axis and angle
rotateZ	Rotate shape about Z-axis and angle
translate	Move shape to new location
show	Display antenna or array structure; Display shape as filled patch
mesh	Mesh properties of metal or dielectric antenna or array structure

Examples

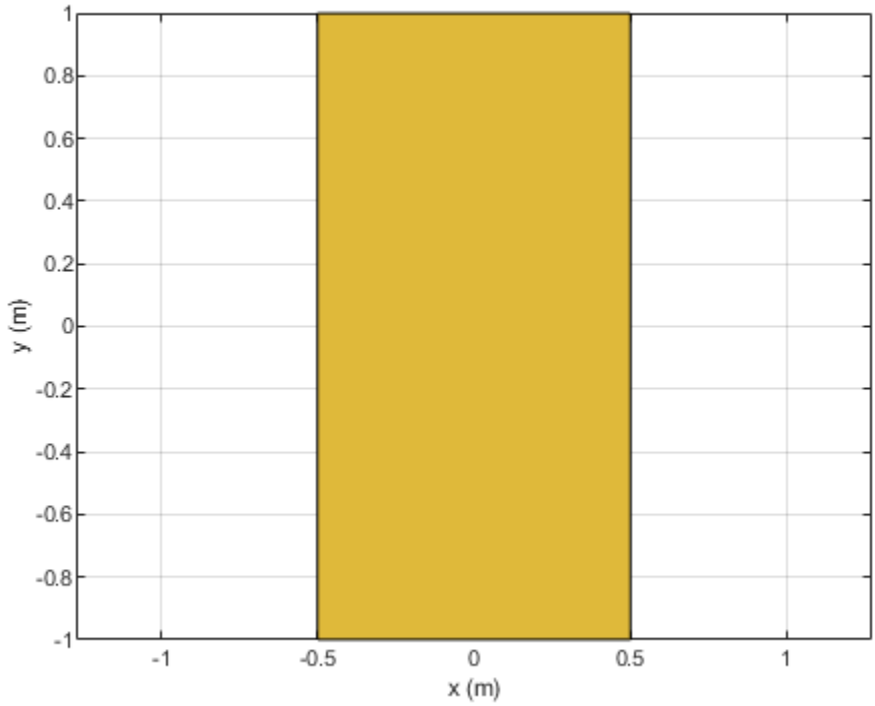
Create Rectangle with Default Properties

Create a rectangle shape using antenna.Rectangle and view it.

```
r1 = antenna.Rectangle
```

```
r1 =
  Rectangle with properties:

         Name: 'myrectangle'
       Center: [0 0]
       Length: 1
        Width: 2
    NumPoints: 2
```

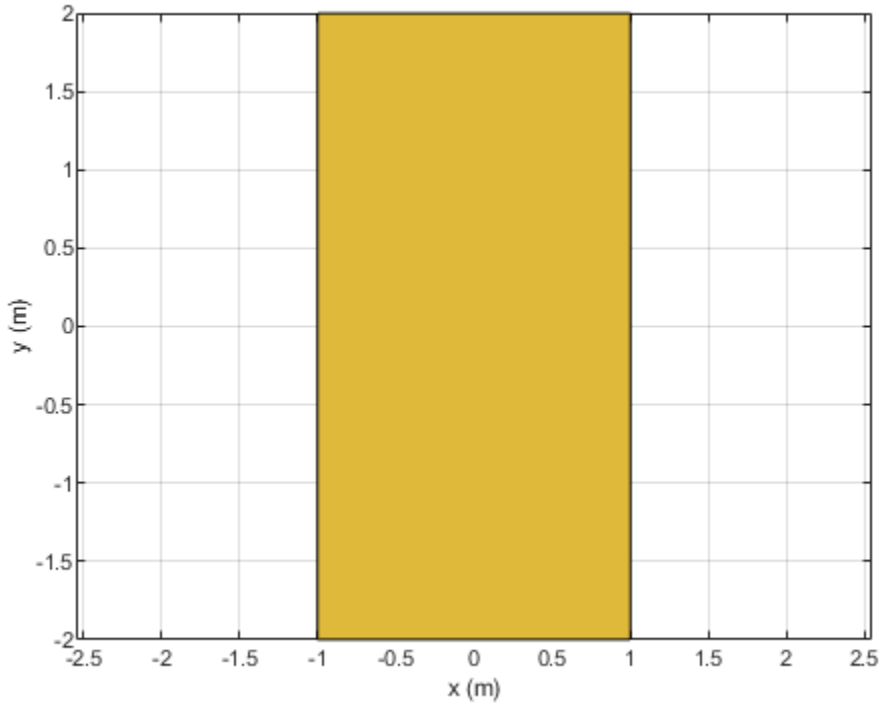
```
show(r1)
```



Create and Rotate Rectangle Using Specified Properties

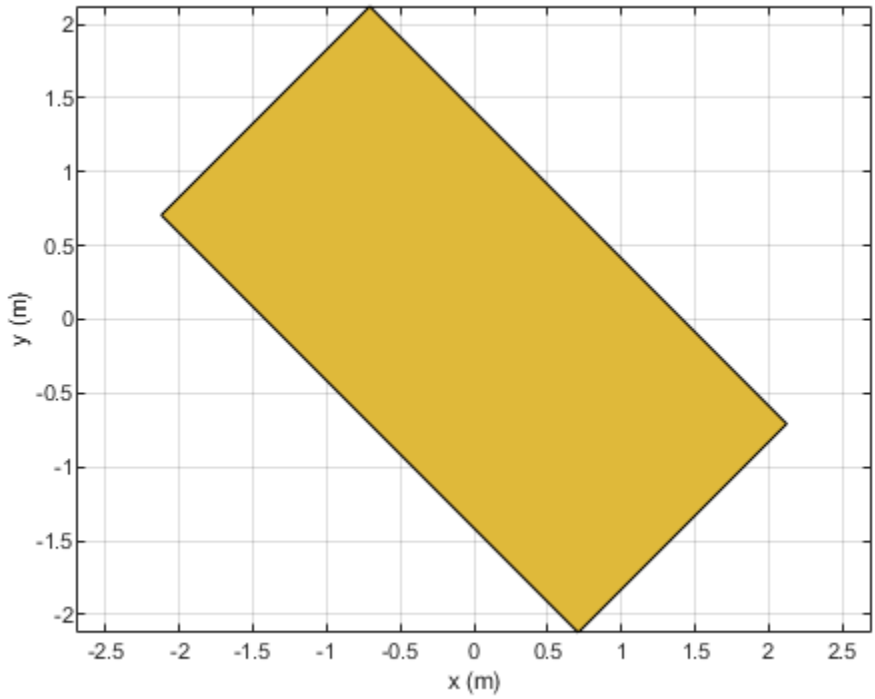
Create and view a rectangle with a length of 2 m and a width of 4 m.

```
r2 = antenna.Rectangle('Length',2,'Width',4);
show(r2)
axis equal
```



Rotate the rectangle.

```
rotateZ(r2,45);
show(r2)
```



Create Notched Rectangle

Create a rectangle with a length of 0.15 m, and a width of 0.15 m.

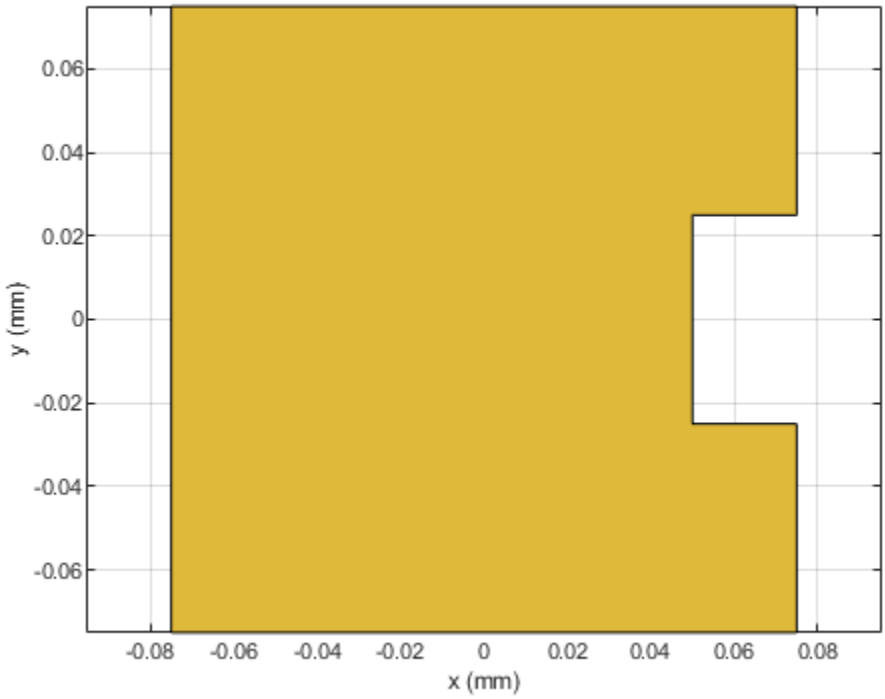
```
r  = antenna.Rectangle('Length',0.15,'Width',0.15);
```

Create a second rectangle with a length of 0.05 m, and a width of 0.05 m. Set the center of the second rectangle at half the length of the first rectangle r.

```
n = antenna.Rectangle('Center',[0.075,0],'Length',0.05,'Width',0.05);
```

Create and view a notched rectangle by subtracting n from r.

```
rn  = r-n;
show(rn)
```



Calculate the area of the notched rectangle.

```
area(rn)
```

```
ans = 0.0212
```

See Also

`antenna.Circle` | `antenna.Polygon`

Introduced in R2017a

PCBWriter

Create PCB board definitions from 2-D antenna designs

Description

Use the `PCBWriter` object to create a printed circuit board (PCB) design files based on multilayer 2-D antenna design. A set of manufacturing files known as Gerber files describes a PCB antennas. A Gerber file uses an ASCII vector format for 2-D binary images.

Creation

Syntax

```
b = PCBWriter(pcbstackobject)
b = PCBWriter(pcbstackobject,rfconnector)
```

Description

`b = PCBWriter(pcbstackobject)` creates a `PCBWriter` object that generates Gerber-format PCB design files based on a 2-D antenna design geometry using PCB stack.

`b = PCBWriter(pcbstackobject,rfconnector)` creates a customized PCB file using specified `rfconnector` type.

`b = PCBWriter(pcbstackobject,writer)` creates a customized PCB file using a specified PCB service, `writer`.

`b = PCBWriter(pcbstackobject,rfconnector,writer)` creates customised PCB file using specified PCB service and PCB connector type.

Input Arguments

`pcbstackobject` — Single feed PCB antenna

`pcbStack` object

Single feed PCB antenna, specified as a `pcbStack` object. For more information, see `pcbStack`.

Example: `p1 = pcbStack` creates a PCB stack object,`p1 a = PCBWriter(p1)`, uses `p1` to create a `PCBWriter` object `a`.

`writer` — PCB service to view PCB design

object

PCB service to view PCB design, specified as `PCBServices` object.

Example: `s =PCBServices.MayhewWriter; a = PCBWriter(p1,s)` uses Mayhew Labs PCB service to view the PCB design. For more information on manufacturing services, see `PCBServices`

`rfconnector` — RF connector type

object

RF connector type for PCB antenna feedpoint, specified as `PCBConnectors` object. For information about connectors , see `PCBConnectors`.

Example: `c = PCBConnectors.SMA_Cinch;a = PCBWriter(p1,c)` uses SMA_Cinch RF connector at feedpoint.

Output Arguments

`b` — PCB Board definition of 2.5D antenna design

object

PCB Board definition of 2.5D antenna design, returned as an object.

Properties

`UseDefaultConnector` — Use default connector

`1` (default) | `0`

Use default connector, specified as `0` or `1`.

Example: `a.UseDefaultConnector = 1`, where `a` is a `PCBWriter` object.

Data Types: `logical`

`ComponentBoundaryLineWidth` — Line widths drawn around components on silk screens
8 (default) | positive scalar

Line widths drawn around components on silk screens, specified as a positive scalar in mils.

Example: `a.ComponentBoundaryLineWidth = 10`, where `a` is a `PCBWriter` object.

Data Types: `double`

`ComponentNameFontSize` — Font size to label components on silk screen
positive scalar

Font size to label components on silk screen, specified as a positive scalar in points.

Example: `a.ComponentNameFontSize = 12`, where `a` is a `PCBWriter` object.

Data Types: `double`

`DesignInfoFontSize` — Font size for design information added outside board profile
positive scalar

Design information text font size added outside board profile, specified as a positive scalar.

Example: `a.DesignInfoFontSize = 12`, where `a` is a `PCBWriter` object.

Data Types: `double`

`Font` — Font used for component name and design info
`'Arial'` (default) | character vector

Font used for component name and design info, specified as a character vector.

Example: `a.Font = 'TimesNewRoman'`, where `a` is a `PCBWriter` object.

Data Types: `char` | `string`

`PCBMargin` — Copper free margin around board
`0.5e-3` (default) | positive scalar

Copper free margin around board, specified as a positive scalar in meters.

Example: `a.PCBMargin = 0.7e-3`, where `a` is a `PCBWriter` object.

Data Types: `double`

`SolderMask` — Add solder mask to top and bottom of PCB
`'both'` (default) | `'top'` | `'bottom'`

Add solder mask to top and bottom of PCB, specified as `'both'`, `'top'`, `'bottom'`.

Example: `a.SolderMask = 'top'`, where `a` is a `PCBWriter` object.

Data Types: `char` | `string`

`SolderPaste` — Generate solder paste files
`1` (default) | `0`

Generate solder paste files as a part of PCB stack, specified as `1` or `0`.

Example: `a.SolderPaste = 0`, where `a` is a `PCBWriter` object.

Data Types: `logical`

Object Functions

gerberWrite Generate Gerber files

Examples

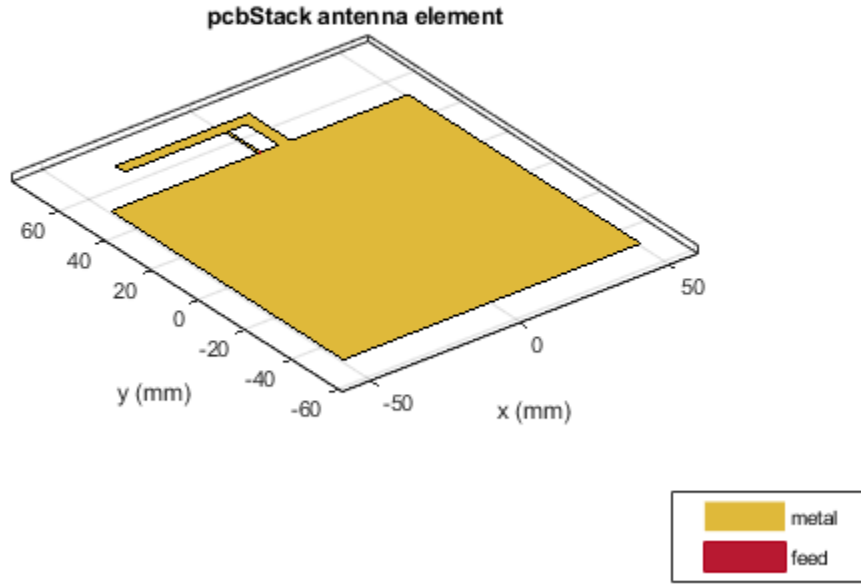
Generate Gerber Format Files From PCB Stack Object

Create a coplanar inverted F antenna

```
fco = invertedFcoplanar('Height',14e-3,'GroundPlaneLength', 100e-3, ...
      'GroundPlaneWidth', 100e-3);
```

Create a `pcbStack` object.

```
p = pcbStack(fco);
show (p);
```



Generate a Gerber format design file using PCB Writer.

```
PW = PCBWriter(p)
```

```
PW =
  PCBWriter with properties:

                      Design: [1x1 struct]
                      Writer: [1x1 Gerber.Writer]
                   Connector: []
          UseDefaultConnector: 1
    ComponentBoundaryLineWidth: 8
```

```
         ComponentNameFontSize: []
            DesignInfoFontSize: []
                          Font: 'Arial'
                     PCBMargin: 5.0000e-04
                    Soldermask: 'both'
                   Solderpaste: 1

   See info for details
```

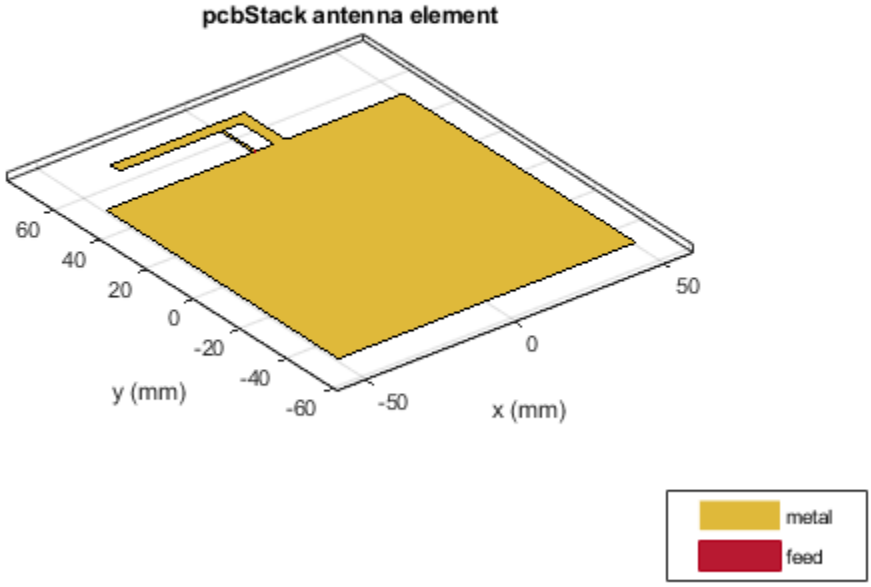
Antenna PCB Design Using SMA Cinch Connector

Create a coplanar inverted F antenna.

```
fco = invertedFcoplanar('Height',14e-3,'GroundPlaneLength', 100e-3, ...
      'GroundPlaneWidth', 100e-3);
```

Create a `pcbStack` object.

```
p = pcbStack(fco);
show(p)
```



Create an SMA_Cinch connector using the `PCBConnectors` object.

```
c = PCBConnectors.SMA_Cinch
```

```
c =
  SMA_Cinch with properties:

                  Type: 'SMA'
                   Mfg: 'Cinch'
                  Part: '142-0711-202'
            Annotation: 'SMA'
             Impedance: 50
             Datasheet: 'https://belfuse.com/resources/Johnson/drawings/dr-142-0711-2
              Purchase: 'https://www.digikey.com/product-detail/en/cinch-connectivity
```

```
              TotalSize: [0.0071 0.0071]
          GroundPadSize: [0.0024 0.0024]
      SignalPadDiameter: 0.0017
        PinHoleDiameter: 0.0013
          IsolationRing: 0.0041
   VerticalGroundStrips: 1

  Cinch 142-0711-202 (Example Purchase)
```

Create an antenna PCB design file using the connector.

```
PW = PCBWriter(p,c)
```

```
PW =
  PCBWriter with properties:

                      Design: [1x1 struct]
                      Writer: [1x1 Gerber.Writer]
                   Connector: [1x1 PCBConnectors.SMA_Cinch]
          UseDefaultConnector: 0
    ComponentBoundaryLineWidth: 8
         ComponentNameFontSize: []
            DesignInfoFontSize: []
                        Font: 'Arial'
                   PCBMargin: 5.0000e-04
                  Soldermask: 'both'
                 Solderpaste: 1

  See info for details
```

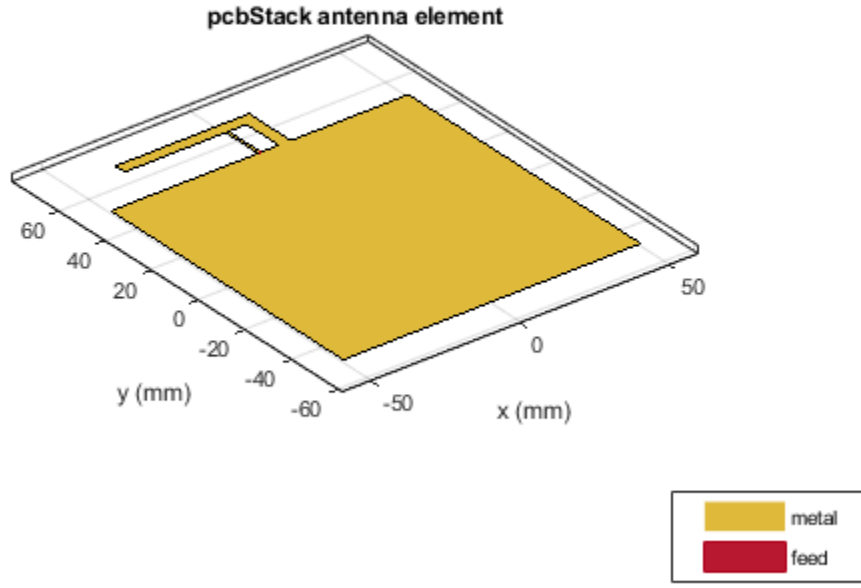
Antenna Design Files Using Advanced Circuits Writer Service

Create a coplanar inverted F antenna.

```
fco = invertedFcoplanar('Height',14e-3,'GroundPlaneLength', 100e-3, ...
      'GroundPlaneWidth', 100e-3);
```

Create a `pcbStack` object.

```
p = pcbStack(fco);
show(p)
```



Use an Advanced Circuits Writer as a PCB manufacturing service.

```
s = PCBServices.AdvancedCircuitsWriter
```

```
s =
  AdvancedCircuitsWriter with properties:

              BoardProfileFile: 'legend'
         BoardProfileLineWidth: 1
               CoordPrecision: [2 6]
                   CoordUnits: 'in'
           CreateArchiveFile: 1
              DefaultViaDiam: 3.0000e-04
          DrawArcsUsingLines: 0
```

```
                ExtensionLevel: 1
                      Filename: 'untitled'
                         Files: {}
        IncludeRootFolderInZip: 0
                   PostWriteFcn: @(obj)sendTo(obj)
    SameExtensionForGerberFiles: 0
                    UseExcellon: 1
```

Create an antenna PCB design file using the above service.

```
PW = PCBWriter(p,s)
```

```
PW =
  PCBWriter with properties:

                       Design: [1x1 struct]
                       Writer: [1x1 PCBServices.AdvancedCircuitsWriter]
                    Connector: []
           UseDefaultConnector: 1
    ComponentBoundaryLineWidth: 8
         ComponentNameFontSize: []
            DesignInfoFontSize: []
                         Font: 'Arial'
                    PCBMargin: 5.0000e-04
                   Soldermask: 'both'
                  Solderpaste: 1

   See info for details
```

Show Antenna PCB Design Using Mayhew Manufacturing Service

Create a coplanar inverted F antenna.

```
fco = invertedFcoplanar('Height',14e-3,'GroundPlaneLength', 100e-3,  ...
                  'GroundPlaneWidth', 100e-3);
```

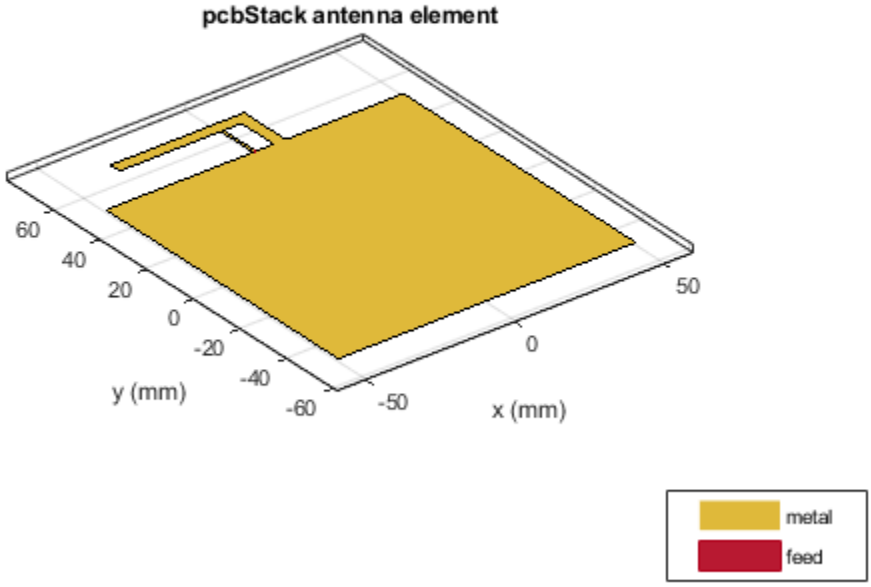
Use this antenna in creating a `pcbStack` object.

```
p = pcbStack(fco)
```

```
p =
  pcbStack with properties:
```

```
              Name: 'Coplanar Inverted-F'
          Revision: 'v1.0'
        BoardShape: [1×1 antenna.Rectangle]
    BoardThickness: 0.0013
            Layers: {[1×1 antenna.Polygon]}
     FeedLocations: [0 0.0500 1]
      FeedDiameter: 5.0000e-04
      ViaLocations: []
       ViaDiameter: []
      FeedViaModel: 'strip'
       FeedVoltage: 1
         FeedPhase: 0
              Tilt: 0
          TiltAxis: [1 0 0]
              Load: [1×1 lumpedElement]
```

```
figure
show(p)
```



Use an SMA_Cinch as an RF connector and Mayhew Writer as a 3-D viewer.

```
c = PCBConnectors.SMA_Cinch
```

```
c =
  SMA_Cinch with properties:

                    Type: 'SMA'
                     Mfg: 'Cinch'
                    Part: '142-0711-202'
              Annotation: 'SMA'
               Impedance: 50
               Datasheet: 'https://belfuse.com/resources/Johnson/drawings/dr-142-0711-2
                Purchase: 'https://www.digikey.com/product-detail/en/cinch-connectivity
```

```
              TotalSize: [0.0071 0.0071]
          GroundPadSize: [0.0024 0.0024]
      SignalPadDiameter: 0.0017
        PinHoleDiameter: 0.0013
          IsolationRing: 0.0041
    VerticalGroundStrips: 1

   Cinch 142-0711-202 (Example Purchase)
```

```
s = PCBServices.MayhewWriter
```

```
s =
  MayhewWriter with properties:

              BoardProfileFile: 'legend'
         BoardProfileLineWidth: 1
                CoordPrecision: [2 6]
                    CoordUnits: 'in'
            CreateArchiveFile: 0
               DefaultViaDiam: 3.0000e-04
           DrawArcsUsingLines: 1
               ExtensionLevel: 1
                     Filename: 'untitled'
                        Files: {}
       IncludeRootFolderInZip: 0
                  PostWriteFcn: @(obj)sendTo(obj)
    SameExtensionForGerberFiles: 0
                   UseExcellon: 1
```

Create an antenna design file using `PCBWriter` .

```
PW = PCBWriter(p,s,c)
```

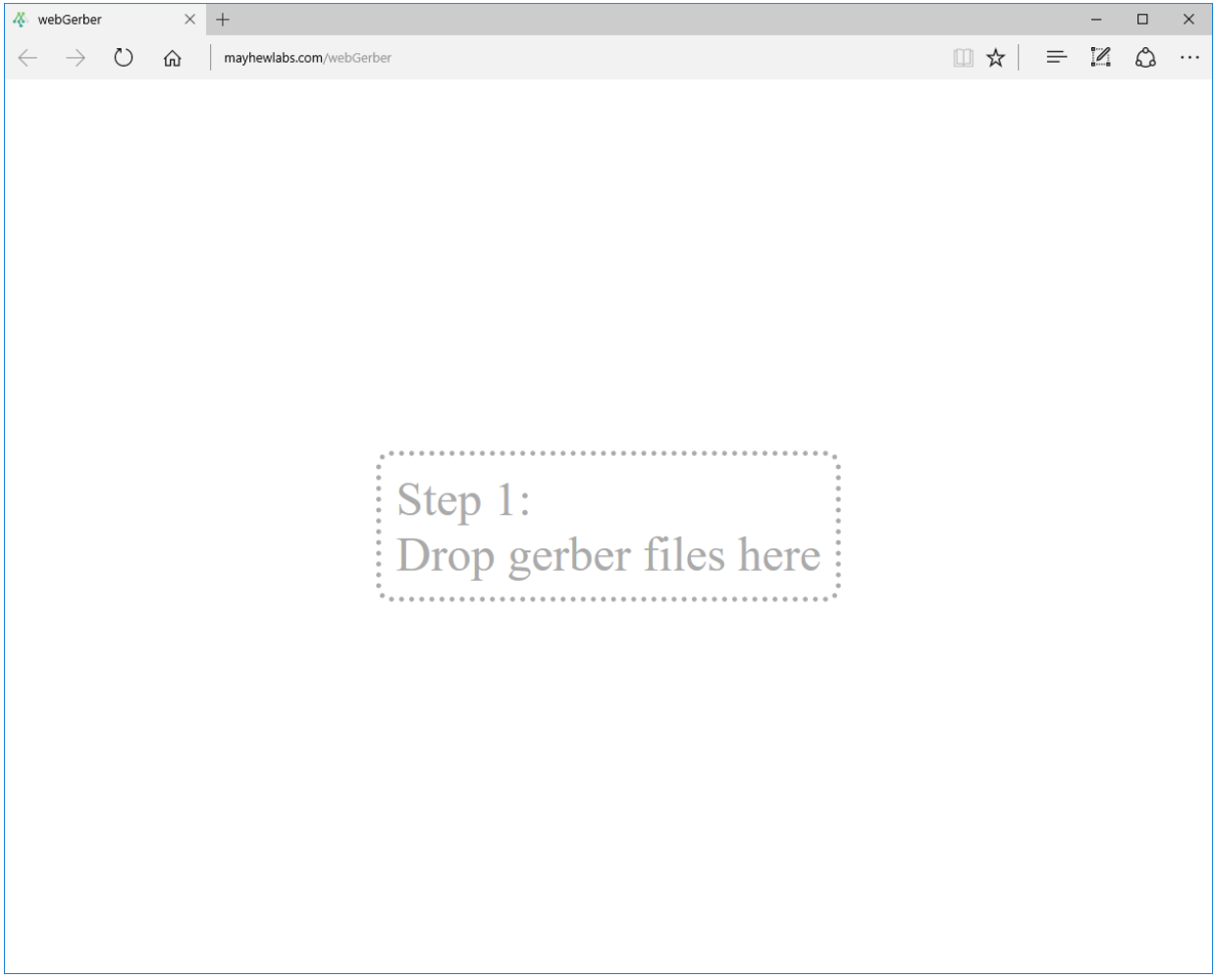
```
PW =
  PCBWriter with properties:

                      Design: [1×1 struct]
                      Writer: [1×1 PCBServices.MayhewWriter]
                   Connector: [1×1 PCBConnectors.SMA_Cinch]
          UseDefaultConnector: 0
    ComponentBoundaryLineWidth: 8
         ComponentNameFontSize: []
           DesignInfoFontSize: []
```

```
                    Font: 'Arial'
                 PCBMargin: 5.0000e-04
                Soldermask: 'both'
               Solderpaste: 1

   See info for details
```

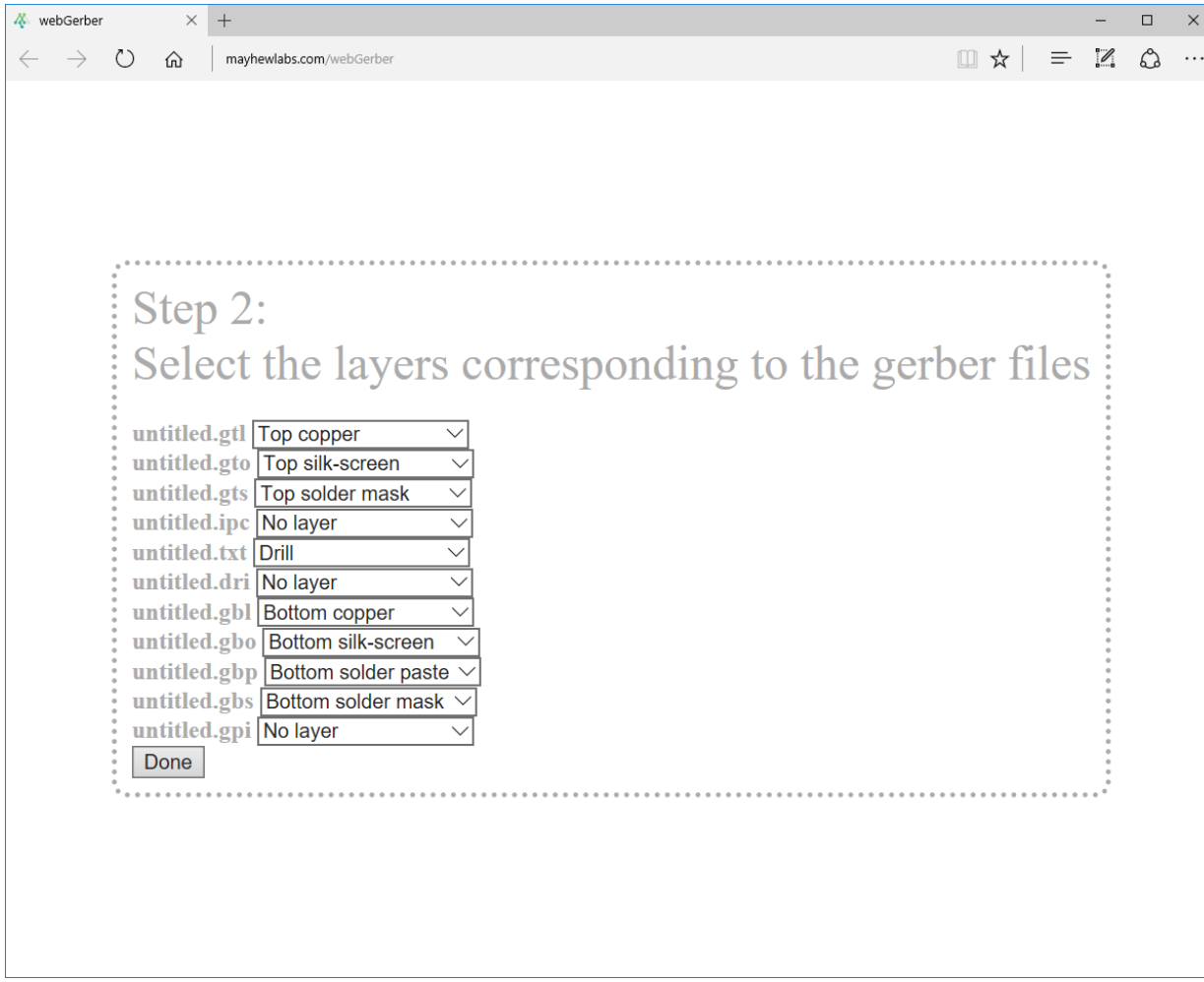
Use the gerberWrite method to create gerber files from the antenna design files. The files generated are then send to the Mayhew writer manufacturing service.

```
gerberWrite(PW)
```

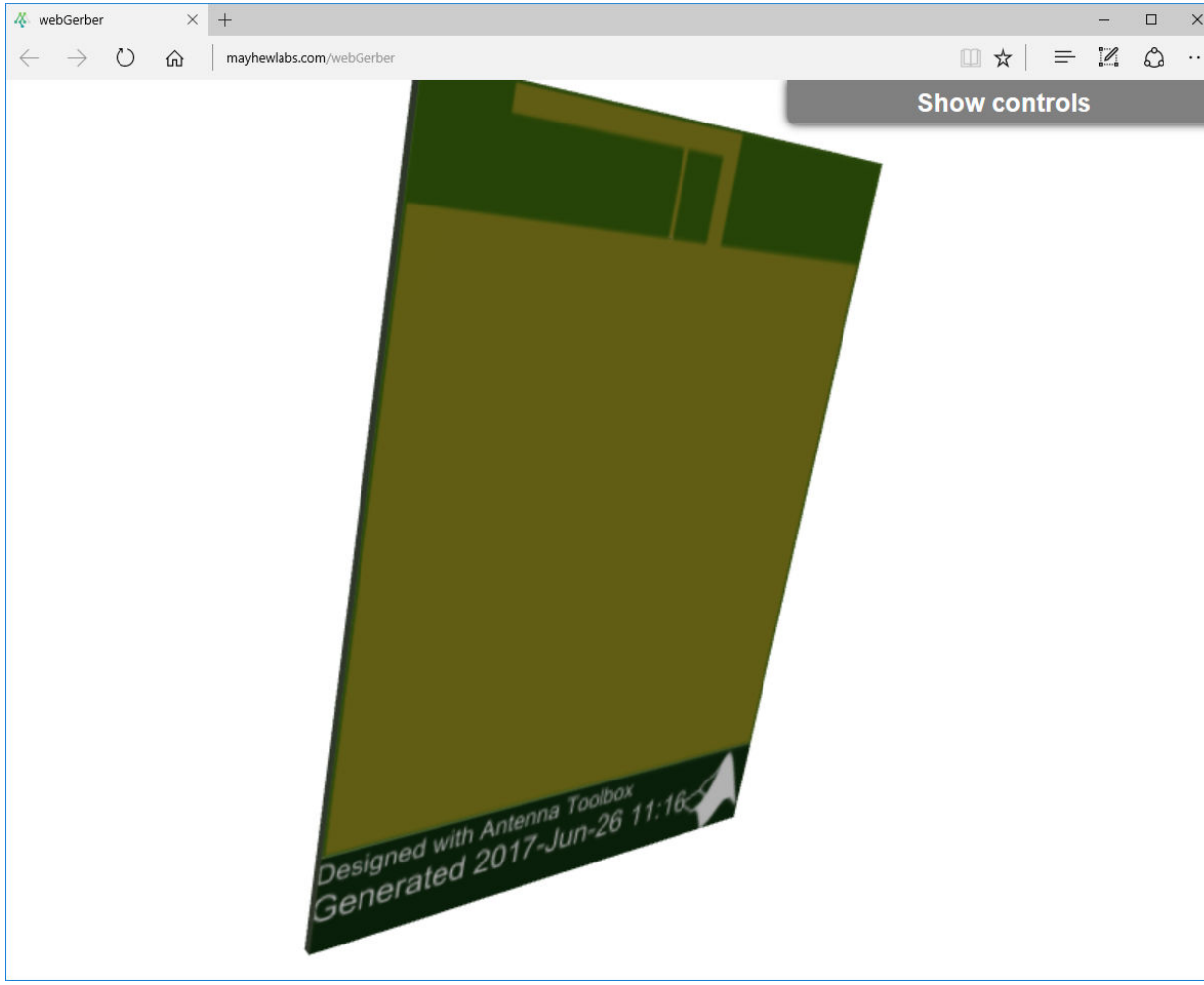
By default, the folder containing the gerber files is called "untitled" and is located in your MATLAB folder. Running this example automatically opens up the Mayhew Labs PCB manufacturing service in your internet browser.



Drag and drop all your files from the "untitled" folder.



Click **Done** to view your Antenna PCB.



See Also

`PCBConnectors` | `PCBServices`

Introduced in R2017b

PCBServices

Customize PCB file generation for PCB manufacturing service

Description

Use the `PCBServices` object to customize printed circuit board (PCB) file generation for a PCB manufacturing service.

Creation

Syntax

```
w = PCBServices.servicetype
```

Description

`w = PCBServices.servicetype` creates a Gerber file based on the type of service specified in `servicetype`.

Input Arguments

`servicetype` — Type of service from PCB services package
character vector

Type of service from PCB services package, specified as one of the following:

- AdvancedCircuitsWriter – Configure Gerber file generation for Advanced Circuits manufacturing.
- CircuitPeopleWriter – -Configure Gerber file generation for CircuitPeople online viewer.
- DirtyPCBsWriter – Configure Gerber file generation for Dirty PCBs manufacturing.
- EuroCircuitsWriter – Configure Gerber file generation for EuroCircuits online viewer.

- GerberLookWriter – Configure Gerber file generation for GerbLook online viewer.
- GerberViewerWriter – Configure Gerber file generation for GerberViewer online viewer.
- MayhewWriter – Configure Gerber file generation for Mayhew Labs online 3-D viewer.
- OSHParkWriter – Configure Gerber file generation for OSH Park PCB manufacturing.
- PCBWayWriter – Configure Gerber file generation for PCBWay PCB manufacturing.
- ParagonWriter – Configure Gerber file generation for Paragon Robotics online viewer.
- SeeedWriter – Configure Gerber file generation for Seeed Fusion PCB manufacturing.
- SunstoneWriter – Configure Gerber file generation for Sunstone PCB manufacturing.
- ZofzWriter – Configure Gerber file generation for Zofz 3-D viewer.

Example: `w = PCBServices.SunstoneWriter` creates Gerber files configured to use Sunstone PCB manufacturing service.

Output Arguments

`w` — PCB manufacturing service
object

PCB manufacturing service, returned as an object.

Properties

`BoardProfileFile` — File type for board profile
`'legend'` | `'profile'`

File type for board profile, specified as `'legend'` or `'profile'`.

Example: `w = PCBServices.SunstoneWriter; w.BoardProfileFile = 'profile'`.

Data Types: `char` | `string`

`BoardProfileLineWidth` — Width of line
`1` | positive scalar

Width of line, specified as a positive scalar in mils.

PCB manufacturers vary on board profile. The most common line width is zero of a fraction width in the chosen unit, for example, 0.1 mil.

Example: `w = PCBServices.SunstoneWriter; w.BoardProfileLineWidth = 0.1`

Data Types: `double`

`CoordPrecision` — Precision of X and Y coordinates written to file

`[2 6]` | 1-by-2 vector

Precision of X and Y coordinates written to file, specified as a 1-by2 vector [*I F*], where,

- *I* – Number of digits in the integer part, 0<=*I*<=6.
- *F* – Number of digits in the fractional part, 4<=*F*<=6.

Example: `w = PCBServices.SunstoneWriter; w.CoordPrecision = [1 3]`

Data Types: `double`

`CoordUnits` — Units of X and Y coordinate

`'in'` | `'mm'`

Units of X and Y coordinates, specified as inches or millimeters.

Example: `w = PCBServices.SunstoneWriter; w.CoordUnits = 'mm'`

Data Types: `char` | `string`

`CreateArchiveFile` — Creates single archive file with all Gerber files

`1` (default) | `0`

Creates single archive file with all Gerber files, specified as `1` or `0`.

Example: `w = PCBServices.SunstoneWriter; w.CreateArchiveFile = 0`

Data Types: `logical`

`DefaultViaDiameter` — Via drill diameter

`3.0000e-04` | positive scalar

Via drill diameter, specified as a positive scalar in meters. PCB manufacturers also call it minimum drilling hole diameter.

Example: `w = PCBServices.SunstoneWriter; w.DefaultViaDiameter = 0.1`

Data Types: `double`

`DrawArcsUsingLines` — Force arcs to be drawn using lines

`0` | `1`

Force arcs to be drawn using lines, specified as `1` or `0`.

Example: `w = PCBServices.SunstoneWriter; w.DrawArcsUsingLines = 0`

Data Types: `logical`

`ExtensionLevel` — Feature content for Gerber file format

`1` (default) | `2`

Feature content for Gerber file format, specified as:

- `1` - Extension 1 is the most compatible setting for downstream PCB manufacturing tools.
- `2` - Extension 2 adds file attributes `%TF.<attr>*%"` to the header and footer of Gerber files.

Example: `w = PCBServices.SunstoneWriter; w.ExtensionLevel = 2`

Data Types: `double`

`Filename` — Name of all files containing Gerber design

`'untitled'` (default) | character vector

Name of all files containing Gerber design, specified as a character vector.

Example: `w = PCBServices.SunstoneWriter; w.Filename = 'antenna_design'.`

Data Types: `char` | `string`

`Files` — Define stack of PCB files

character vector

Define stack of PCB files, specified as a character vector. This definition includes:

- Multiples files describing one PCB.
- A "file" as a memory object containing buffers that describe or hold the file content before the file is written.
- Cell vector of `Gerber.FileFunction` objects, one per file.

Data Types: `cell` | `char` | `string`

`IncludeRootFolderInZip` — Include top-level folder in zip archive
`1` | `0`

Include top-level folder in zip archive, specified as `1` or `0`.

Example: `w = PCBServices.SunstoneWriter; w.IncludeRootFolderInZip = 0`

Data Types: `logical`

`PostWriteFcn` — Function to invoke after a successful write operation
function handle (default)

Function to invoke after a successful write operation, specified as a function handle. In this case, it is the `sendTo` function. This property makes sure that the location of the Gerber files and the website of the manufacturing service is open after a successful write function.

Example: `w = PCBServices.SunstoneWriter; w.PostWriteFcn = @(obj)sendTo(obj)`

Data Types: `function_handle`

`SameExtensionForGerberFiles` — Use `.gbr` to be file extension for all Gerber files
`0` | `1`

Use `.gbr` to be file extension for all Gerber files, specified as `0` or `1`.

Example: `w = PCBServices.SunstoneWriter; w.SameExtensionForGerberFiles = 1`

Data Types: `logical`

`UseExcellon` — Generate Excellon drill files
`1` | `0`

Generate Excellon drill files, specified as `0` or `1`.

Example: `w = PCBServices.SunstoneWriter; w.UseExcellon = 1`, generates Gerber format drill files with `'x2'` extension.

Data Types: `logical`

Examples

PCB Using Mayhew Labs 3-D Viewer

Create a coplanar inverted F antenna.

```
fco = invertedFcoplanar('Height',14e-3,'GroundPlaneLength', 100e-3,  ...
                  'GroundPlaneWidth', 100e-3);
```

Use this antenna in creating a PCB stack object.

```
p = pcbStack(fco);
```

Use a Mayhew Writer with a `profile` board for viewing the PCB in 3D.

```
s = PCBServices.MayhewWriter;
s.BoardProfileFile = 'profile'
```

```
s =
  MayhewWriter with properties:

               BoardProfileFile: 'profile'
          BoardProfileLineWidth: 1
                 CoordPrecision: [2 6]
                     CoordUnits: 'in'
                CreateArchiveFile: 0
                   DefaultViaDiam: 3.0000e-04
              DrawArcsUsingLines: 1
                  ExtensionLevel: 1
                        Filename: 'untitled'
                           Files: {}
         IncludeRootFolderInZip: 0
                    PostWriteFcn: @(obj)sendTo(obj)
    SameExtensionForGerberFiles: 0
                    UseExcellon: 1
```

Create an antenna design file using `PCBWriter`.

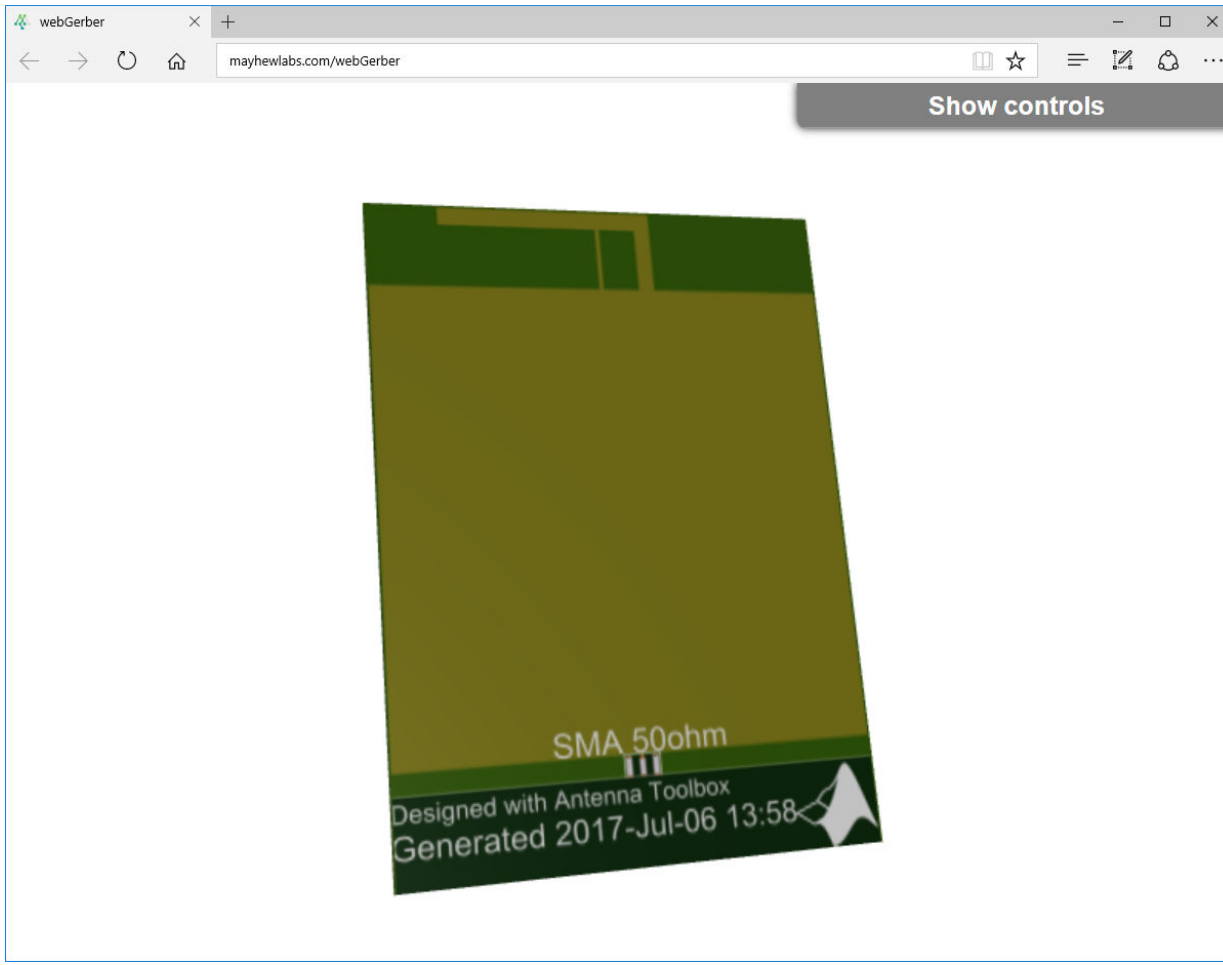
```
PW = PCBWriter(p,s);
```

Use the `gerberWrite` method to create Gerber files from the antenna design files.

```
gerberWrite(PW)
```

The location of the folder and the Mayhew labs website opens automatically.

To view the board, drag and drop the files. Click **Done**.



See Also

`PCBConnectors` | `PCBWriter` | `gerberWrite`

Introduced in R2017b

PCBConnectors

RF connector at antenna feedpoint

Description

Use `PCBConnectors` object to specify RF connectors used for antenna printed circuit board (PCB) feed points. The result is generally a set of modifications to the PCB design files. The changes to the PCB include new copper landing pads and traces, and changes to solder mask, silk screen, and solder paste files.

Creation

Syntax

```
c = PCBConnectors.connectortype
```

Description

`c = PCBConnectors.connectortype` creates Gerber files based on the type of connector to use at antenna feedpoint specified in `connectortype`.

Input Arguments

`connectortype` — Type of connector from PCB connector package
character vector

Type of connector from PCB connector package, specified as one of the following:

- Coax Connectors - Coax RG11, RG174, RG58, and RG59 connectors directly soldered to PCB pads.
- IPX Connectors - LightHorse IPX SMT jack or plug surface mount RF connector.
- MMCX Connectors - MMCX Cinch or Samtec surface mount RF connectors.

- SMA Connectors - Generic 5-pad SMA surface mount RF connectors, with 4 corner rectangular pads, 1 round center pin. Cinch and Multicomp SMA RF connectors.
- SMAEdge Connectors- Generic SMA edge-launch surface mount RF connector. Cinch and Samtec SMA edge-launch RF connectors.
- SMB Connectors - Johnson/Emerson and Pasternack SMB surface mount RF connectors.
- SMC Connectors - Pasternack SMC and SMC edge-launch surface mount RF connectors.
- Coaxial Cable Connectors - Semi-rigid `0.020 inch`, `0.034 inch`, `0.047 inch`, and `0.118 inch` coaxial cable soldered to PCB pads.

Example: `c = PCBConnectors.Semi_020` creates Gerber files configured to use semi-rigid `0.020 inch` coaxial cables.

Output Arguments

`c` — PCB connector
object

PCB connector, returned as an object.

Properties

Common Properties for All Connectors

Type — Type of connector
character vector

Type of connector, specified as a character vector.

Example: `'Coax_RG11'`

Data Types: `char` | `string`

`Mfg` — Name of component manufacturer
character vector

Name of component manufacturer, specified as a character vector.

Example: `'Belden'`

Data Types: `char` | `string`

`Part` — Manufacturer part number
character vector | string

Manufacturer part number, specified as a character vector or string.

Example: `'RG11'`

Data Types: `char` | `string`

`Annotation` — Text added to PCB to identify component
character vector

Text added to PCB to identify component, specified as a character vector.

Example: `'RG59U'`

Data Types: `char` | `string`

`Impedance` — Connector impedance
`50` | positive scalar

Connector impedance, specified as a positive scalar in ohms.

Example: `c = PCBConnectors.MMCX_Cinchc.Impedance = 70`

Data Types: `double`

`Datasheet` — URL for component specifications
character vector

URL for component specifications, specified as a character vector. Data sheets are typically PDF files.

Data Types: `char` | `string`

`Purchase` — URL for purchasing connector
character vector

URL for purchasing connector, specified as a character vector.

Data Types: `char` | `string`

Common Properties for All Coax Connectors

`SignalPinDiameter` — Circular pad diameter

positive scalar

Circular pad diameter connecting the signal wire of the coax to the feedpoint, specified as a positive scalar in meters. The pin diameter is greater than the diameter of the signal wire.

Example: `c = PCBConnectors.Coax_RG59c.SignalPinDiameter = 1.0000e-03`

Data Types: `double`

`DielectricDiameter` — Dielectric diameter

positive scalar

Dielectric diameter (white material around signal wire), specified as a positive scalar in meters. Dielectric diameter specifies the size of the non-conductive isolation ring on the PCB between the signal wire and the ground plane.

Example: `c = PCBConnectors.Coax_RG59c.DielectricDiameter = 0.0073`

Data Types: `double`

`ShieldDiameter` — Ground ring diameter

positive scalar

Ground ring diameters used to solder coax shield, specified as a positive scalar in meters.

Example: `c = PCBConnectors.Coax_RG59c.ShieldDiameter = 0.0085`

Data Types: `double`

`AddThermals` — Thermal relief

`1` | `0`

Thermal relief around coaxial shield connection, specified as `0` or `1`. Thermal relief reduces the heat needed to solder the coax shield to the ground.

Example: `c = PCBConnectors.Coax_RG59c.AddThermals = 0`

Data Types: `logical`

`ThermalsDiameter` — Arc-shaped gaps outer diameter

positive scalar

Arc-shaped gaps outer diameter in the ground plane, specified as a positive scalar in meters.

Example: `c = PCBConnectors.Coax_RG59c.ThermalsDiameter = 0.0100`

Data Types: `double`

`ThermalsBridgeWidth` — Width of four conductive bridges
positive scalar

Width of four conductive bridges created across thermal gap, specified as a positive scalar in meters. The bridges are established during electrical grounding.

Example: `c = PCBConnectors.Coax_RG59c.ThermalBridgeWidth = 0.0015`

Data Types: `double`

Common Properties for All 5-Pad Symmetric Surface Mount Connectors

`TotalSize` — Total length of each side of rectangular connector footprint
2-element vector

Total length of each side of rectangular connector footprint, specified as a 2-element vector with each element unit in meters.

Example: `c = PCBConnectors.SMA_Multicompc.TotalSize = [0.0063 0.0063]`

Data Types: `double`

`GroundPadSize` — Length of each side of ground pad
2-element vector

Length of each side of ground pad, specified as a 2-element vector with each element unit in meters. The pads are located in each of the four corners of the connector footprint.

Example: `c = PCBConnectors.SMA_Multicompc.GroundPadSize = [0.0016 0.0016]`

Data Types: `double`

`SignalPadDiameter` — Circular pad diameter
positive scalar

Circular pad diameter connecting the signal pin of the coax connector, specified as a positive scalar in meters. The pad is at the center of the connector footprint.

Example: `c = PCBConnectors.SMA_Multicompc.SignalPadDiameter = 0.0012`

Data Types: `double`

`PinHoleDiameter` — Via pin diameter
positive scalar

Via pin diameter, specified as a positive scalar in meters.

Example: `c = PCBConnectors.SMA_Multicompc.ViaPinDiameter = 0.0012`

Data Types: `double`

`IsolationRing` — Diameter of isolation ring that removes semicircle of copper from inner corner of ground pads
scalar

Diameter of isolation ring that removes semicircle of copper from inner corner of ground pads, specified as a scalar in meters.

Example: `c = PCBConnectors.SMA_Multicompc.IsoltationRing =0.0012`

Data Types:

`VerticalGroundStrips` — Vertical ground strips between upper and lower ground pads
scalar

Vertical ground strips between upper and lower ground pads, specified as a scalar.

Example: `c = PCBConnectors.SMA_Multicompc.VerticalGroundStrips = 1`

Data Types: `double`

Common Properties for All Edge-Launch Surface Mount Connectors

`GroundPadSize` — Ground pad size
2-element vector

Ground pad size, specified as a 2-element vector with each element unit in meters.

Example: `c = PCBConnectors.SMAEdgec.GroundPadSize = [0.0014 0.0042]`

Data Types: `double`

`GroundSeparation` — Space between ground pads
positive scalar

Space between ground pads on the ground side of the board, specified as a positive scalar in meters.

Example: `c = PCBConnectors.SMAEdgec.GroundSeparation = 0.0043`

Data Types: `double`

`GroundPadIsolation` — Width of copper removed around top layer ground pads
positive scalar

Width of copper removed around top layer ground pads, specified as a positive scalar in meters. This property isolates the ground pads from any signal traces or structures.

Example: `c = PCBConnectors.SMAEdgec.GroundPadIsolation = 2.5000e-04`

Data Types: `double`

`SignalPadSize` — Signal pad size
2-element vector

Signal pad size, specified as a 2-element vector with each element unit in meters.

Example: `c = PCBConnectors.SMAEdgec.SignalPadSize = [0.0013 0.0036]`

Data Types: `double`

`SignalGap` — Gap between PCB edge and start of signal pad copper
positive scalar

Gap between PCB edge and start of signal pad copper, specified as a positive scalar in meters.

Example: `c = PCBConnectors.SMAEdgec.SignalGap = 1.0000e-04`

Data Types: `double`

`SignalLineWidth` — Width of signal trace
positive scalar

Width of signal trace extending from the signal pad to the feedpoint location, specified as a positive scalar in meters.

Example: `c = PCBConnectors.SMAEdgec.SignalLineWidth = 8.0000e-04`

Data Types: `double`

`EdgeLocation` — PCB side that receives edge connector

`'north'` | `'south'` | `'east'` | `'west'`

PCB side that receives edge connector, specified as `'north'`, `'south'`, `'east'`, `'west'`.

Example: `c = PCBConnectors.SMAEdgec.EdgeLocation = 'south'`

Data Types: `char`

`EdgeBoardProfile` — Extend PCB to add connector beyond design area

`0` | `1`

Extend PCB to add connector beyond design area, specified as `0` or `1`

Example: `c = PCBConnectors.SMAEdgec.EdgeBoardProfile = 1`

Data Types: `logical`

`FillGroundSide` — Fill connector region on ground side of board with copper

`0` | `1`

Fill connector region on ground side of the board with copper, specified as `0` or `1`

Example: `c = PCBConnectors.SMAEdgec.FillGroundSide = 1`

Data Types: `logical`

Common Properties for All Staggered Surface Mount Connectors

`GroundPadSize` — Ground pad size

2-element vector

Ground pad size, specified as a 2-element vector with each element unit in meters.

Example: `c = PCBConnectors.IPX_Plug_Lighthorsec.GroundPadSize = [0.0010 0.0022]`

Data Types: `double`

`GroundPadXSeparation` — Distance between pair of ground pads along X-axis

positive scalar

Distance between pair of ground pads along X-axis, specified as a positive scalar in meters.

Example: `c = PCBConnectors.IPX_Plug_Lighthorsec.GroundPadXSeparation = 0.0019`

Data Types: `double`

`GroundPadYOffset` — Y-offset from signal pad to signal pad center line
positive scalar

Y-offset from signal pad to signal pad center line, specified as a positive scalar in meters.

Example: `c = PCBConnectors.IPX_Plug_Lighthorsec.GroundPadYOffset = 0.0015`

Data Types: `double`

`SignalPadSize` — Signal pad size
2-element vector

Signal pad size, specified as a 2-element vector with each element unit in meters.

Example: `c = PCBConnectors.IPX_Plug_Lighthorsec.SignalPadSize = [1.0000e-03 1.0000e-03]`

Data Types: `double`

`SignalMinYSeparation` — Minimum separation from ground at bottom or top for signal pad
positive scalar

Minimum separation from ground at bottom or top for signal pad, specified as a positive scalar in meters.

Example: `c = PCBConnectors.IPX_Plug_Lighthorsec.SignalMinYSeparation = 1.0000e-03`

Data Types: `double`

Examples

PCB Using Coax_RG11 Connector

Create a coplanar inverted F antenna.

```
fco = invertedFcoplanar('Height',14e-3,'GroundPlaneLength', 100e-3,  ...
                  'GroundPlaneWidth', 100e-3);
```

Use this antenna to create a `pcbStack` object.

```
p = pcbStack(fco);
```

Use a Coax_RG11 RF connector with a pin diameter of 2 mm.

```
c = PCBConnectors.Coax_RG11;
c.PinDiameter = 2.000e-03
s = PCBServices.MayhewWriter;
```

```
c =

  Coax_RG11 with properties:

                  Type: 'Coax'
                   Mfg: 'Belden'
                  Part: 'RG11'
            Annotation: 'RG11'
             Impedance: 75
             Datasheet: 'http://www.belden.com/techdatas/english/8233.pdf'
              Purchase: ''
           PinDiameter: 0.0020
    DielectricDiameter: 0.0072
        ShieldDiameter: 0.0085
      ThermalsDiameter: 0.0100
   ThermalsBridgeWidth: 0.0015
           AddThermals: 1

   <a href="matlab:web('http://www.belden.com/techdatas/english/8233.pdf','-browser');'
```

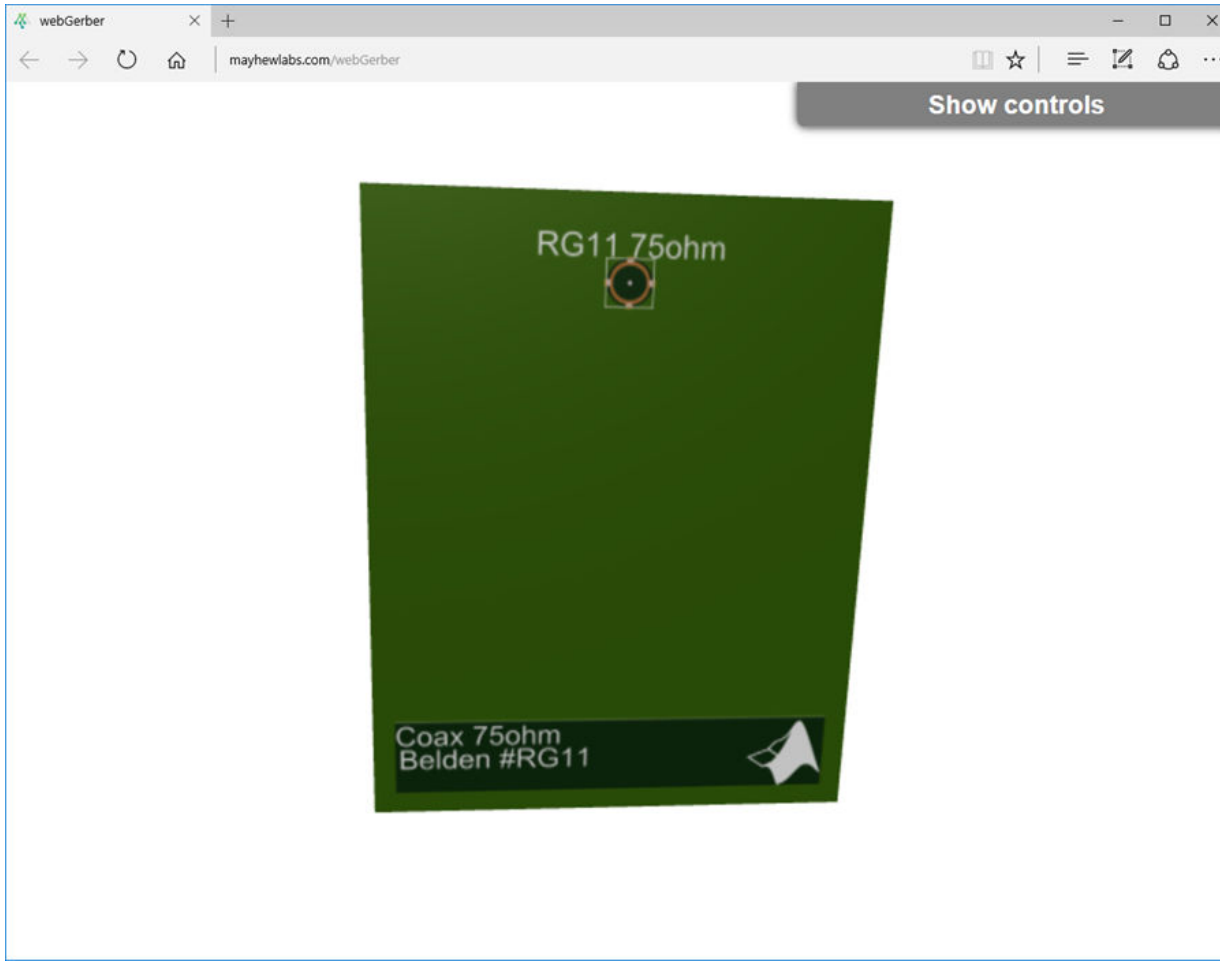
Create an antenna design file using `PCBWriter` .

```
PW = PCBWriter(p,s,c);
```

Use the `gerberWrite` method to create Gerber files from the antenna design files.

```
gerberWrite(PW)
```

To view the board, drag and drop the files. Click **Done**.



Authoring Custom RF Connector

```
classdef SMA_Jack_Cinch < PCBConnectors.BaseSMT5PadSymmetric
    % Cinch SMA surface mount RF connector.

    properties (Constant) % Abstract
        Type       = 'SMA'
        Mfg        = 'Cinch'
        Part       = '142-0701-631'
        Annotation = 'SMA'
```

```
        Impedance  = 50
        Datasheet  = 'http://www.farnell.com/datasheets/1720451.pdf?_ga=2.164811836.207
        Purchase   = 'http://www.newark.com/johnson/142-0701-631/rf-coaxial-sma-jack-st
    end

    methods
        function RFC = SMA_Jack_Cinch
            RFC.TotalSize           = [0.5 0.5]*25.4e-3;
            RFC.GroundPadSize       = [0.102 0.102]*25.4e-3;
            RFC.SignalPadDiameter   = 0.1*25.4e-3;
            RFC.PinHoleDiameter     = 1.27e-3;
            RFC.IsolationRing       = 0.22*25.4e-3;
            RFC.VerticalGroundStrips = false;
        end
    end
end
```

See Also

PCBServices | PCBWriter | gerberWrite

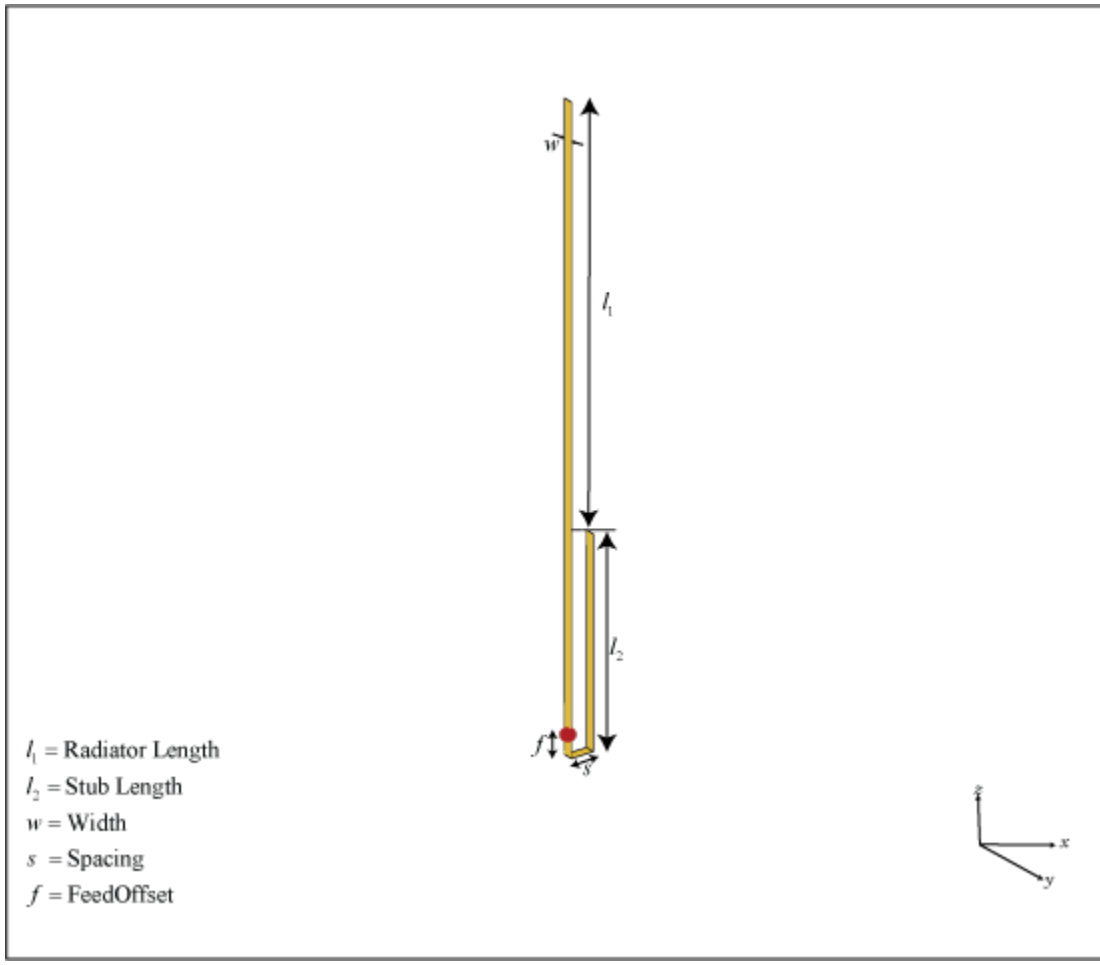
Introduced in R2017b

dipoleJ

Create J-dipole antenna

Description

Use the `dipoleJ` object to create a J-dipole on the Y-Z plane. The antenna contains a half-wavelength radiator and a quarter-wavelength stub. By default, the antenna dimensions are for an operating frequency of 144 MHz.



Creation

Syntax

```
jdipole = dipoleJ
jdipole = dipoleJ(Name,Value)
```

Description

`jdipole = dipoleJ` creates a J-dipole antenna for an operating frequency of 144 MHz.

`jdipole = dipoleJ(Name,Value)` creates a J-dipole antenna with additional properties specified by one or more name-value pair arguments. For example, `jdipole = dipoleJ('Width',0.2)` creates a J-dipole with a strip width of 0.2 m. Enclose each property name in quotes.

Properties

`RadiatorLength` — Radiator length

`0.9970` (default) | scalar

Radiator length, specified as a scalar in meters.

Example: `'RadiatorLength',0.9`

Example: `jdipole.RadiatorLength = 0.9`

Data Types: `double`

`StubLength` — Parallel line stub length

`0.4997` (default) | scalar

Parallel line stub length, specified as a scalar in meters.

Example: `'StubLength',0.3`

Example: `jdipole.StubLength = 0.3`

Data Types: `double`

`Width` — Strip width

`0.0200` (default) | scalar

Strip width, specified as a scalar in meters.

Example: `'StripWidth',0.0500`

Example: `jdipole.StripWidth = 0.0500`

Data Types: `double`

`Spacing` — Space between the stub and the radiator

`0.0460` (default) | scalar

Space between the parallel line stub and the radiator, specified as a scalar in meters.

Example: `'Spacing',0.0500`

Example: `jdipole.Spacing = 0.0500`

Data Types: `double`

`FeedOffset` — Signed distance to feed from base of stub on large arm

0.0490 (default) | scalar

Signed distance to the feed from the base of stub on the large arm, specified as a scalar in meters.

Example: `'FeedOffset',0.0345`

Example: `jdipole.FeedOffset = 0.0345`

Data Types: `double`

`Load` — Lumped elements

[1x1 `lumpedElement`] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. You can add a load anywhere on the surface of the antenna. By default, the load is at the origin. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`, where, `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `jdipole.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector in degrees.

Example: `'Tilt',90`

Example: `jdipole.Tilt = [90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array

returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

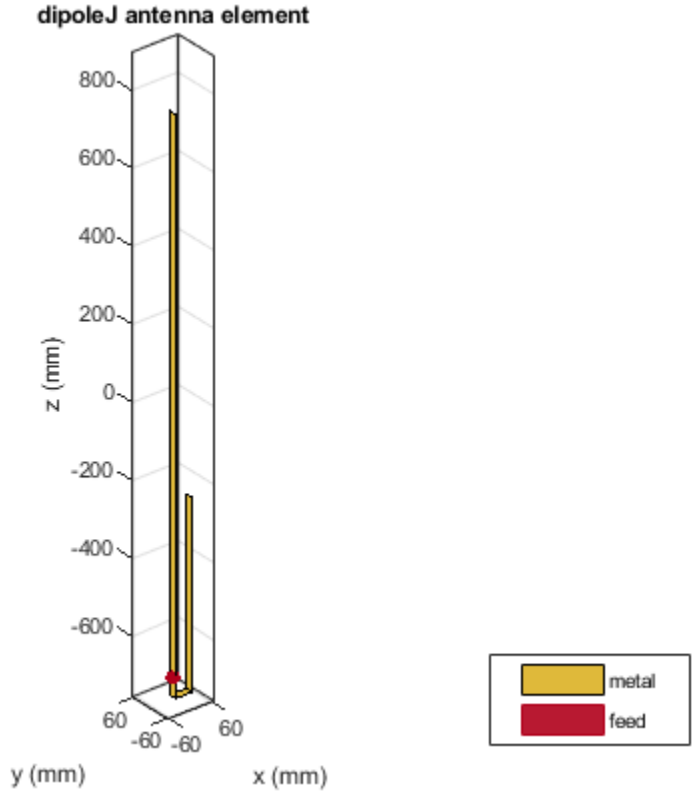
Default J-Dipole Antenna

Create and view a default J-dipole antenna.

```
d = dipoleJ
```

```
d =
  dipoleJ with properties:

    RadiatorLength: 0.9970
        StubLength: 0.4997
           Spacing: 0.0460
             Width: 0.0200
        FeedOffset: -0.6994
              Tilt: 0
          TiltAxis: [1 0 0]
              Load: [1x1 lumpedElement]
```

```
show(d)
```



Impedance of J-Dipole Antenna

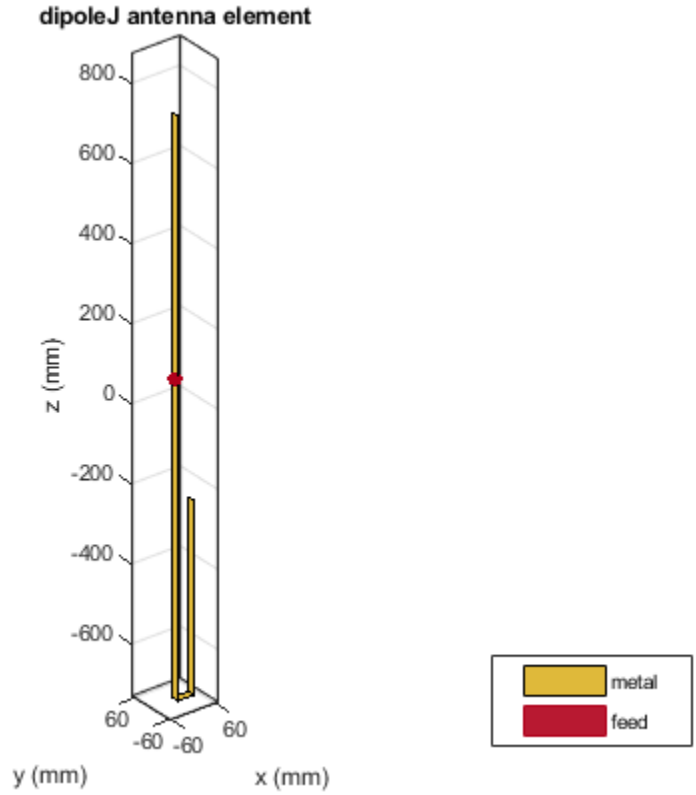
Create and view a J-dipole antenna with the following specifications:

Radiator length = 0.978 m

Stub length = 0.485 m

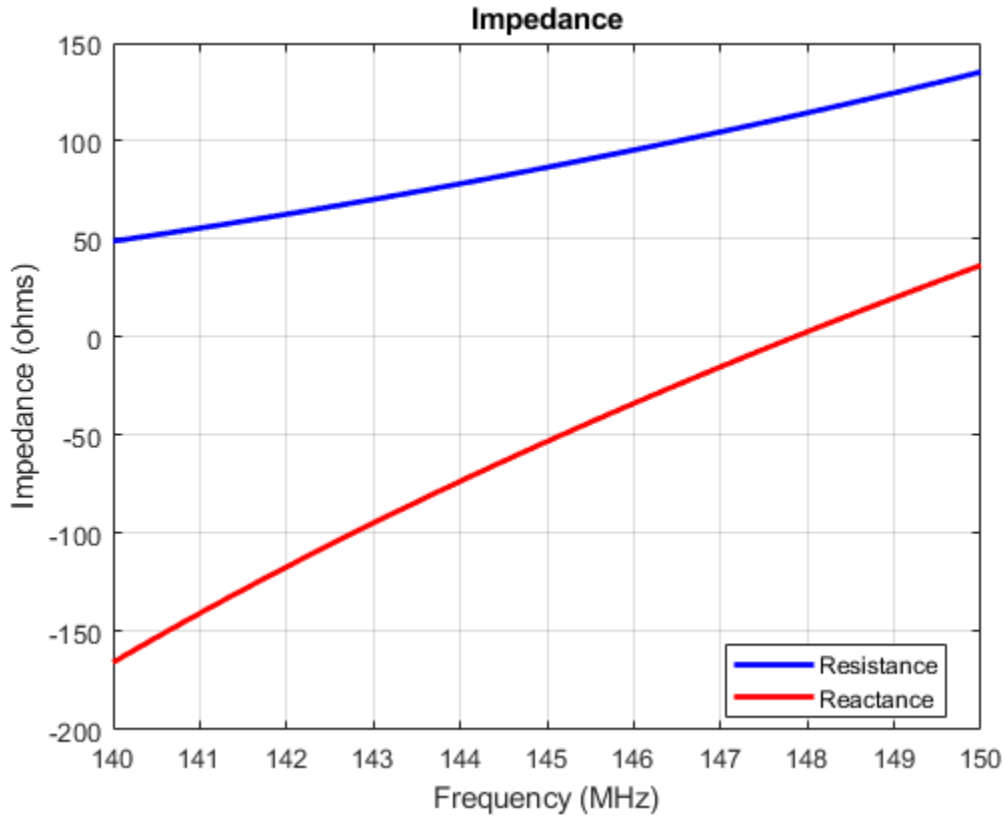
FeedOffset = 0.049 m

```
dj = dipoleJ('RadiatorLength',0.978,'StubLength',0.485, ...
      'FeedOffset',0.070);
show(dj)
```



Calculate the impedance of the antenna over a frequency span 140MHz - 150MHz.

```
impedance(dj,linspace(140e6,150e6,51));
```



See Also

Topics

“Rotate Antenna and Arrays”

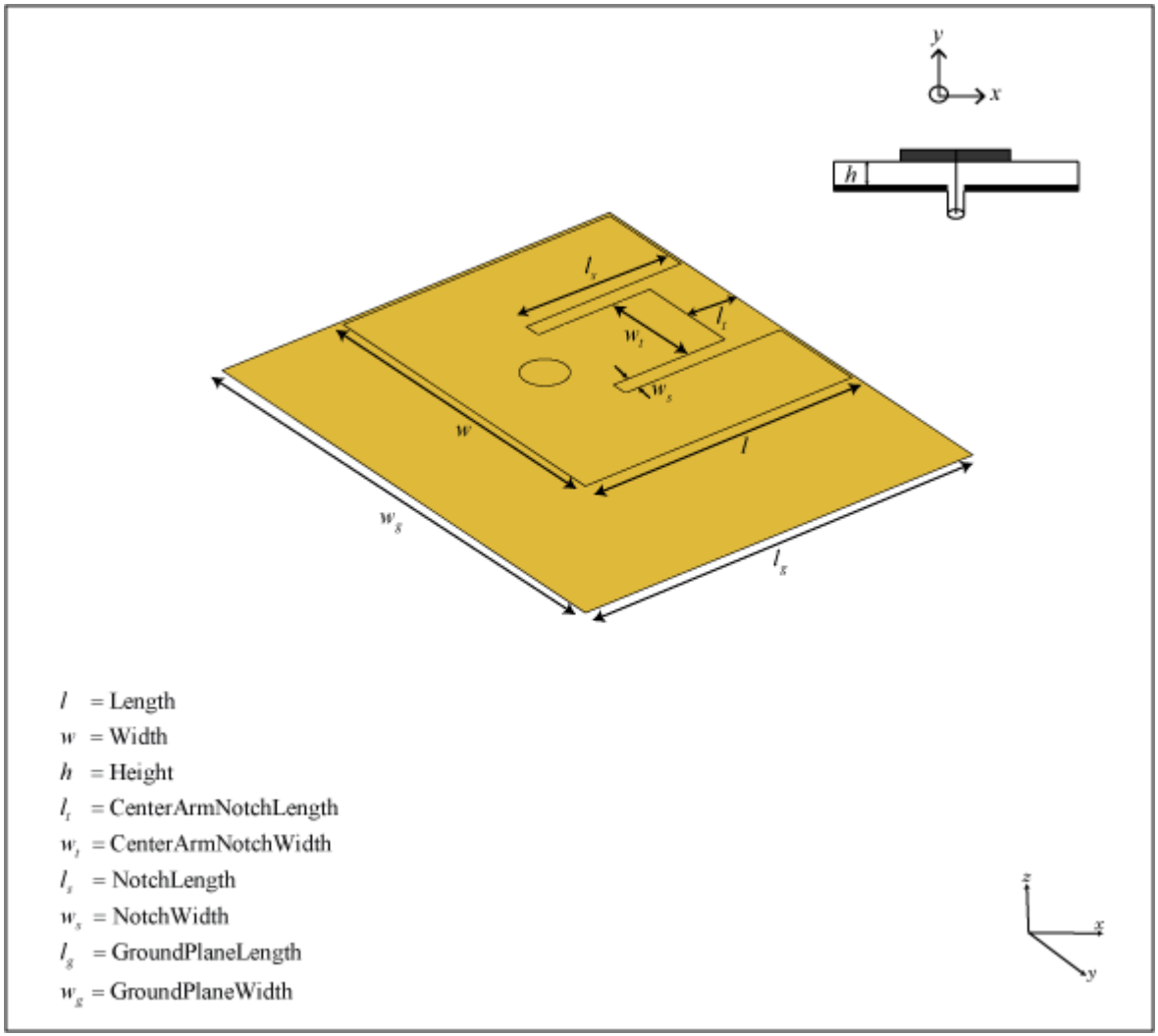
Introduced in R2018a

patchMicrostripEnotch

Create probe-fed E-shaped microstrip patch antenna

Description

Use the `patchMicrostripEnotch` object to create a probe-fed E-shaped microstrip patch antenna. The default patch is centered at the origin with the feedpoint along the length. By default, the dimensions are chosen for an operating frequency of 6.6 GHz for air or 5.5 GHz for Teflon.



l = Length
w = Width
h = Height
l_t = CenterArmNotchLength
w_t = CenterArmNotchWidth
l_s = NotchLength
w_s = NotchWidth
l_g = GroundPlaneLength
w_g = GroundPlaneWidth

Creation

Syntax

```
epatch = patchMicrostripEnotch
epatch = patchMicrostripEnotch(Name,Value)
```

Description

`epatch = patchMicrostripEnotch` creates an E-shaped microstrip patch antenna.

`epatch = patchMicrostripEnotch(Name,Value)` sets properties using one or more name-value pairs. For example, `epatch = patchMicrostripEnotch('Width',0.2)` creates a microstrip E-patch with a patch width of 0.2 m. Enclose each property name in quotes.

Properties

`Length` — Patch length along X-axis

`0.0172` (default) | scalar

Patch length along X-axis, specified as a scalar in meters.

Example: `'Length',0.9`

Example: `epatch.Length = 0.9`

Data Types: `double`

`Width` — Patch width along Y-axis

`0.0200` (default) | scalar

Patch width along Y-axis, specified as a scalar in meters.

Example: `'Width',0.0500`

Example: `epatch.Width = 0.0500`

Data Types: `double`

`Height` — Patch height above ground plane along Z-axis
`0.0032` (default) | scalar

Patch height above ground plane along Z-axis, specified as a scalar in meters.

Example: `'Height',0.00500`

Example: `epatch.Height = 0.00500`

Data Types: `double`

`CenterArmNotchLength` — Notch length on center E-arm along X-axis
`0.0028` (default) | scalar

Notch length on center E-arm along X-axis, specified as a scalar in meters.

Example: `'CenterArmNotchLength',0.100`

Example: `epatch.CenterArmNotchLength = 0.100`

Data Types: `double`

`CenterArmNotchWidth` — Notch width on center E-arm along Y-axis
`0.0062` (default) | scalar

Notch width on center E-arm along Y-axis, specified as a scalar in meters.

Example: `'CenterArmNotchWidth',0.0600`

Example: `epatch.CenterArmNotchWidth = 0.0600`

Data Types: `double`

`NotchLength` — Notch length along X-axis
`0.0100` (default) | scalar

Notch length along X-axis, specified as a scalar in meters.

Example: `'NotchLength',0.0200`

Example: `epatch.NotchLength = 0.0200`

Data Types: `double`

`NotchWidth` — Notch width along Y-axis
`1.00003-03` (default) | scalar

Notch width along Y-axis, specified as a scalar in meters.

Example: `'NotchWidth',0.00600`

Example: `epatch.NotchWidth = 0.00600`

Data Types: `double`

`GroundPlaneLength` — Ground plane length along X-axis
`0.0250` (default) | scalar

Ground plane length along X-axis, specified as a scalar in meters.

Example: `'GroundPlaneLength',120e-3`

Example: `epatch.GroundPlaneLength = 120e-3`

Data Types: `double`

`GroundPlaneWidth` — Ground plane width along Y-axis
`0.0300` (default) | scalar

Ground plane width along Y-axis, specified as a scalar in meters.

Example: `'GroundPlaneWidth',120e-3`

Example: `epatch.GroundPlaneWidth = 120e-3`

Data Types: `double`

`PatchCenterOffset` — Signed distance of patch from origin
`[0 0]` (default) | two-element real-valued vector

Signed distance of patch from origin, specified as a two-element real-valued vector. Units are in meters. Use this property to adjust the location of the patch relative to the ground plane. Distances are measured along the length and width of the ground plane.

Example: `'PatchCenterOffset',[0.01 0.01]`

Example: `epatch.PatchCenterOffset = [0.01 0.01]`

Data Types: `double`

`FeedOffset` — Signed distance of feed from origin
`[–0.0034 0]` (default) | two-element real-valued vector

Signed distance of feed from origin, specified as a two-element real-valued vector. Units are in meters. Use this property to adjust the location of the feedpoint relative to the ground plane and patch. Distances are measured along the length and width of the ground plane.

Example: `'FeedOffset',[0.01 0.01]`

Example: `epatch.FeedOffset = [0.01 0.01]`

Data Types: `double`

`FeedDiameter` — Feed diameter
`0.0013` (default) | scalar

Feed diameter, specified as a scalar in meters.

Example: `'FeedDiameter',0.0600`

Example: `epatch.FeedDiameter = 0.0600`

Data Types: `double`

`Substrate` — Type of dielectric material
`'Air'` (default) | dielectric object

Type of dielectric material used as a substrate, specified as a dielectric object. You place the patch over this dielectric substrate. For more information, see `dielectric`. For more information on dielectric substrate meshing, see “Meshing”.

Note The substrate dimensions must be equal to the groundplane dimensions.

Example: `d = dielectric('FR4'); 'Substrate',d`

Example: `d = dielectric('FR4'); epatch.Substrate = d`

`Load` — Lumped elements
[1x1 `lumpedElement`] (default) | lumped element object

Lumped elements added to the antenna feed, specified as a lumped element object. You can add a load anywhere on the surface of the antenna. By default, the load is at the origin. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`, where `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `epatch.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
`0` (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector in degrees.

Example: `'Tilt',90`

Example: `epatch.Tilt = [90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array

mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

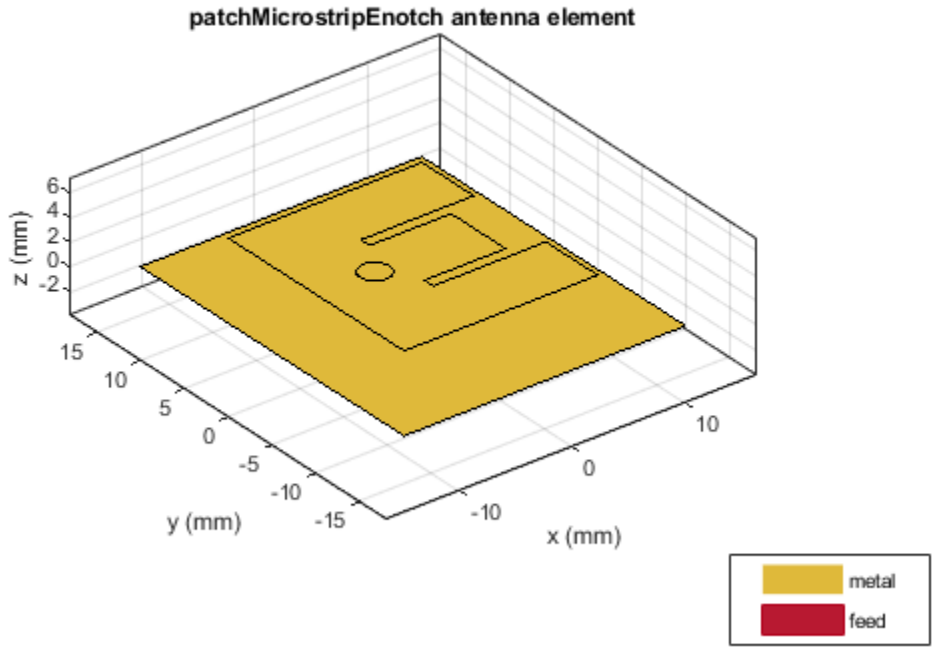
Default E-Shaped Patch Antenna

Create and view a default E-shaped patch antenna.

```
epatch = patchMicrostripEnotch
```

```
epatch =
  patchMicrostripEnotch with properties:

                 Length: 0.0172
                  Width: 0.0200
             NotchLength: 0.0100
              NotchWidth: 1.0000e-03
    CenterArmNotchLength: 0.0028
     CenterArmNotchWidth: 0.0062
                  Height: 0.0032
               Substrate: [1x1 dielectric]
       GroundPlaneLength: 0.0250
        GroundPlaneWidth: 0.0300
       PatchCenterOffset: [0 0]
              FeedOffset: [-0.0034 0]
            FeedDiameter: 0.0013
                    Tilt: 0
                TiltAxis: [1 0 0]
                    Load: [1x1 lumpedElement]
```

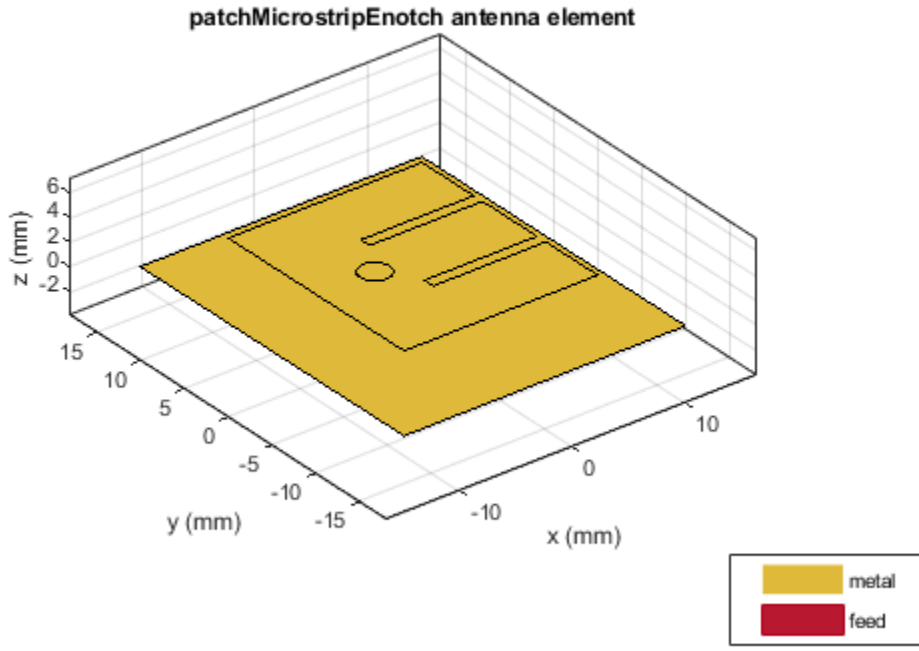
```
show(epatch)
```



E-Shaped Patch with No Slot Along Center E-Arm

Create and view an E-shaped patch with no slot on the center E-arm.

```
epatch = patchMicrostripEnotch('CenterArmNotchLength',0);
show(epatch);
```



See Also

`patchMicrostrip` | `patchMicrostripCircular` | `patchMicrostripTriangular`

Topics

“Rotate Antenna and Arrays”

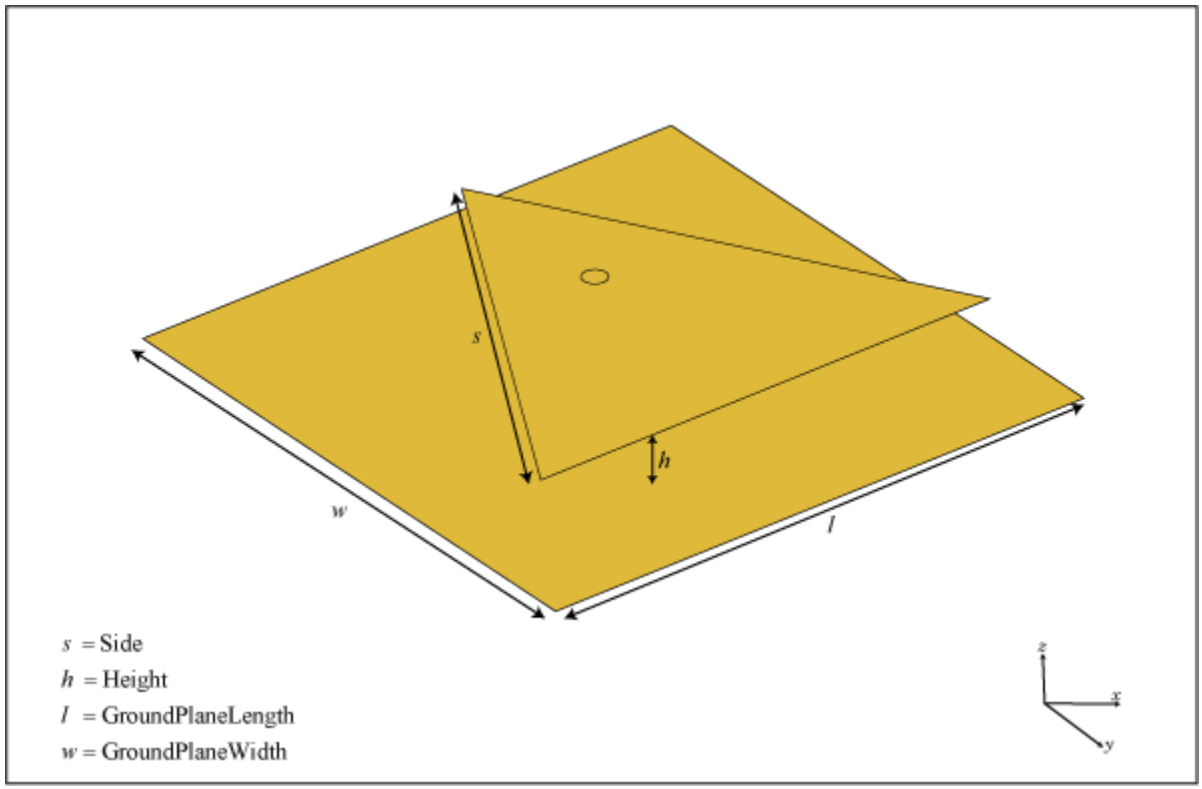
Introduced in R2018a

patchMicrostripTriangular

Create triangular microstrip patch antenna

Description

Use the `patchMicrostripTriangular` object to create a triangular microstrip patch antenna. The default patch is centered at the origin. By default, the dimensions are chosen for an operating frequency of 15 GHz. If you use a Teflon substrate, the default operating frequency is at 12.5 GHz.



Creation

Syntax

```
trianglepatch = patchMicrostripTriangular
trianglepatch = patchMicrostripTriangular(Name,Value)
```

Description

`trianglepatch = patchMicrostripTriangular` creates a triangular microstrip patch antenna.

`trianglepatch = patchMicrostripTriangular(Name,Value)` sets properties using one or more name-value pairs. For example, `trianglepatch = patchMicrostripTriangular('Side',0.2)` creates a triangular microstrip patch with a side length of 0.2 m. Enclose each property name in quotes.

Properties

`Side` — Side lengths of triangular patch

`0.0102` (default) | scalar | two or three-element vector

Side lengths of triangular patch, specified as a scalar in meters or a two or three-element vector with each element unit in meters.

- Equilateral triangle - `Side` property value is a scalar. All three sides of the triangle are equal.
- Isosceles triangle - `Side` property value is a two-element vector. The first value specifies the base of the triangle along the x-axis. The second value specifies the other two sides of the triangle.
- Scalene triangle - `Side` property value is a three-element vector. The first value specifies the base of the triangle along the x-axis. The remaining two values specify the other two sides of the triangle.

Example: `'Side',0.2`

Example: `trianglepatch.Side = [0.2,0.3,0.4]` where the first value is the base of the scalene triangle along the x-axis.

Data Types: `double`

`Height` — Patch height above ground along Z-axis
`0.0016` (default) | scalar

Patch height above ground along Z-axis, specified as a scalar in meters.

Example: `'Height',0.2`

Example: `trianglepatch.Height = 0.002`

Data Types: `double`

`GroundPlaneLength` — Ground plane length along X-axis
`0.0120` (default) | scalar

Ground plane length along X-axis, specified as a scalar in meters.

Example: `'GroundPlaneLength',120e-3`

Example: `trianglepatch.GroundPlaneLength = 120e-3`

Data Types: `double`

`GroundPlaneWidth` — Ground plane width along Y-axis
`0.0120` (default) | scalar

Ground plane width along Y-axis, specified as a scalar in meters.

Example: `'GroundPlaneWidth',120e-3`

Example: `trianglepatch.GroundPlaneWidth = 120e-3`

Data Types: `double`

`PatchCenterOffset` — Signed distance of patch from origin
`[0 0]` (default) | two-element real vector

Signed distance of patch from origin, specified as a two-element real vector with each element unit in meters. Use this property to adjust the location of the patch relative to the ground plane. Distances are measured along the length and width of the ground plane.

Example: `'PatchCenterOffset',[0.01 0.01]`

Example: `trianglepatch.PatchCenterOffset = [0.01 0.01]`

Data Types: `double`

`FeedOffset` — Signed distance of feed from origin

`[0 5.4173e-04]` (default) | two-element real vector

Signed distance of feed from origin, specified as a two-element real vector with each element unit in meters. Use this property to adjust the location of the feedpoint relative to the ground plane and patch. Distances are measured along the length and width of the ground plane.

Example: `'FeedOffset',[0.01 0.01]`

Example: `trianglepatch.FeedOffset = [0.01 0.01]`

Data Types: `double`

`FeedDiameter` — Feed diameter

`2.5000e-04` (default) | scalar

Feed diameter, specified as a scalar in meters.

Example: `'FeedDiameter',0.0600`

Example: `trianglepatch.FeedDiameter = 0.0600`

Data Types: `double`

`Substrate` — Type of dielectric material

`'Air'` (default) | dielectric object

Type of dielectric material used as a substrate, specified as a dielectric object. You place the patch over this dielectric substrate. For more information, see `dielectric`. For more information on dielectric substrate meshing, see “Meshing”.

Note The substrate dimensions must be equal to the groundplane dimensions.

Example: `d = dielectric('FR4'); 'Substrate',d`

Example: `d = dielectric('FR4'); trianglepatch.Substrate = d`

`Load` — Lumped elements

[1x1 `lumpedElement`] (default) | lumped element object

Lumped elements added to the antenna feed, specified as a lumped element object. You can add a load anywhere on the surface of the antenna. By default, the load is at the origin. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`, where `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `trianglepatch.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector in degrees.

Example: `'Tilt',90`

Example: `trianglepatch.Tilt = [90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna

charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

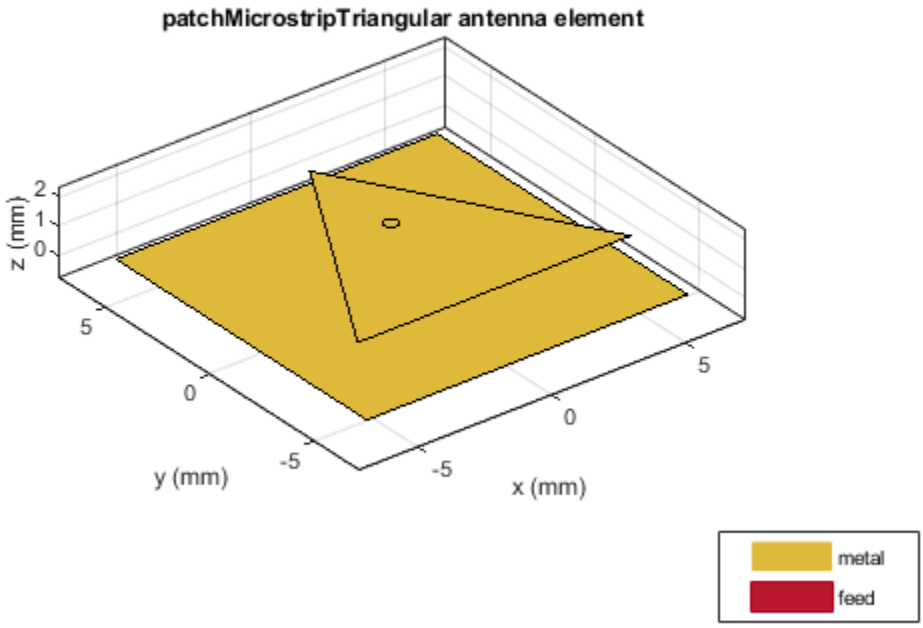
Default Triangular Microstrip Patch and Radiation Pattern

Create and view a default triangular microstrip patch.

```
p = patchMicrostripTriangular
```

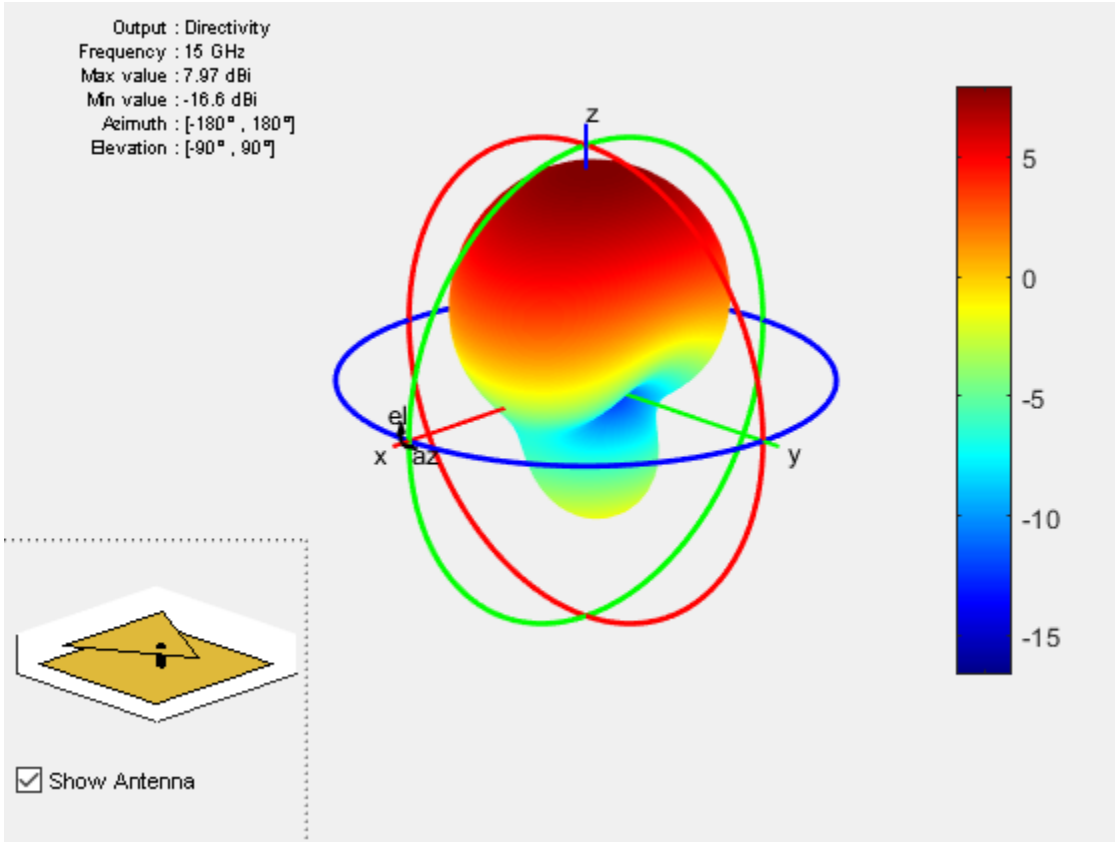
```
p =
  patchMicrostripTriangular with properties:

                Side: 0.0102
              Height: 0.0016
           Substrate: [1x1 dielectric]
    GroundPlaneLength: 0.0120
     GroundPlaneWidth: 0.0120
    PatchCenterOffset: [0 0]
          FeedOffset: [0 5.4173e-04]
       FeedDiameter: 2.5000e-04
               Tilt: 0
           TiltAxis: [1 0 0]
               Load: [1x1 lumpedElement]
```

```
show(p)
```



Plot the radiation pattern at 15 GHz.

```
pattern(p,15e9)
```



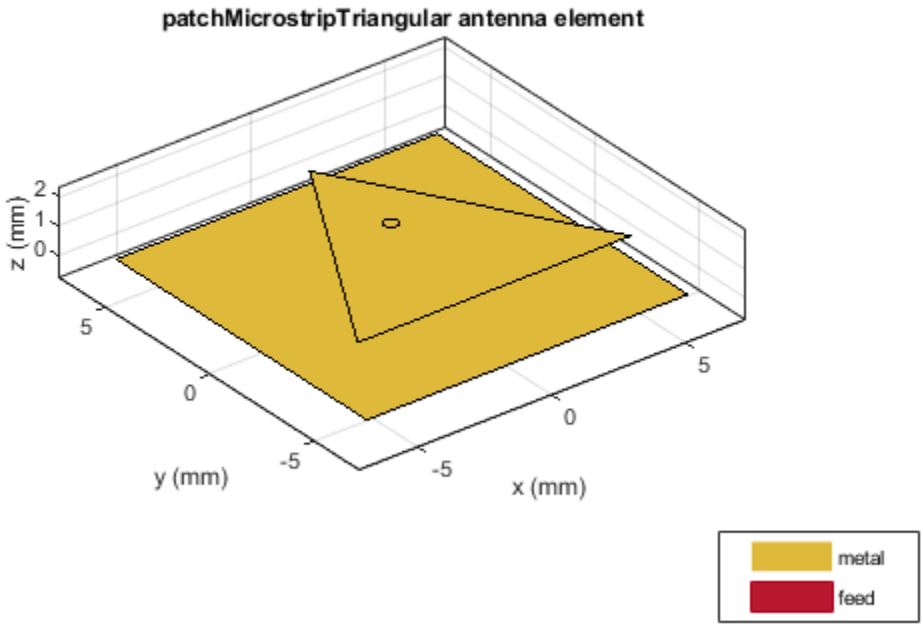
Different Types of Triangular Patch Antennas

Create different types of triangles to use in the patch.

Equilateral Triangle

Create an equilateral triangle patch of side 10.2mm.

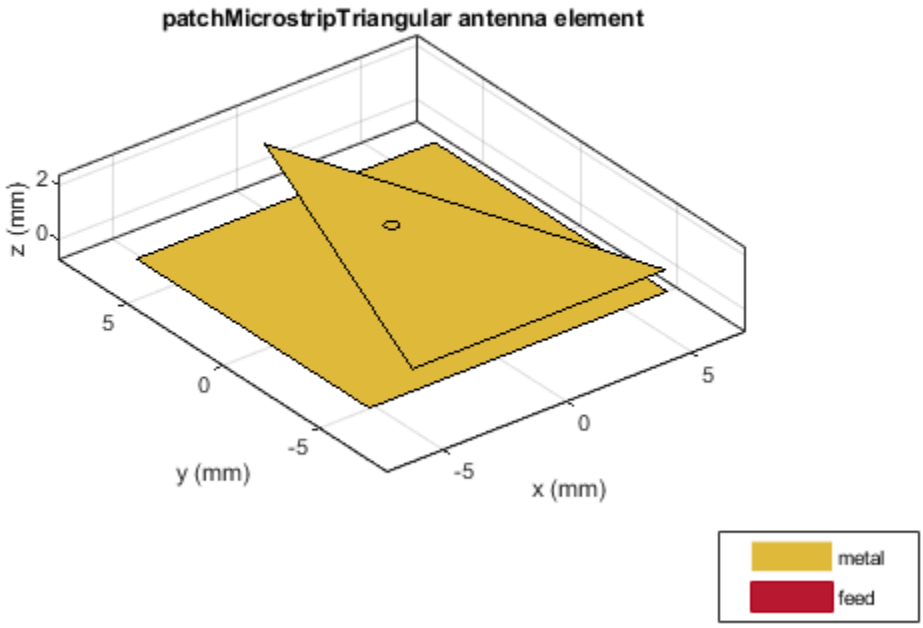
```
ant = patchMicrostripTriangular('Side',10.2e-3);
show(ant);
```



Isosceles Triangle

Create an isosceles triangular patch antenna with sides using the following dimensions: 10.2 mm and 15 mm.

```
ant =  patchMicrostripTriangular('Side',[10.2e-3,15e-3]);
show(ant);
```

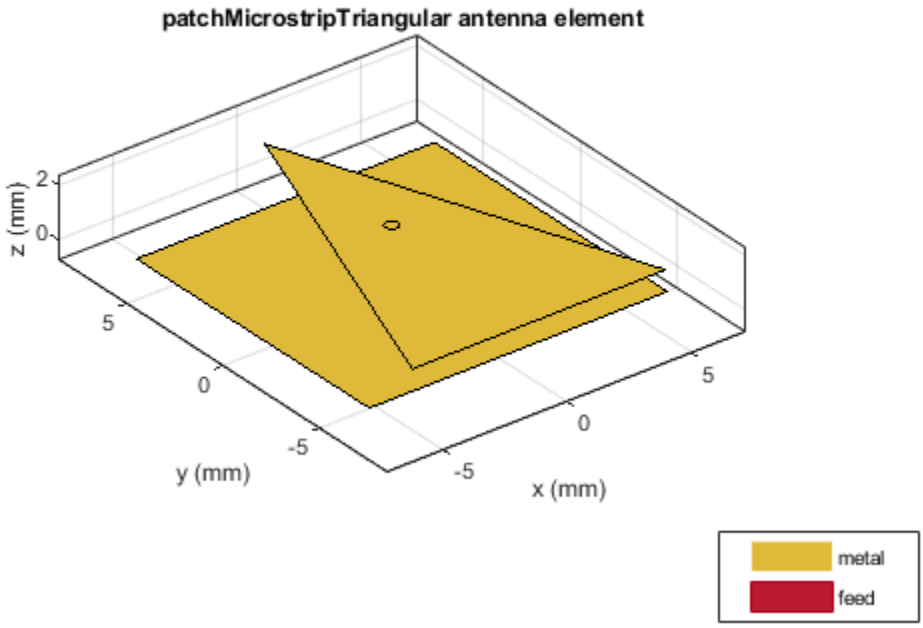


In the above figure, you will see that the first value of the side is chosen as the base of the triangle.

Scalene Triangle

Create a scalene triangular patch antenna with side using the following dimensions: 21 mm, 13 mm, and 20 mm.

```
patchMicrostripTriangular('Side',[21e-3,13e-3,20e-3]);
show(ant);
```

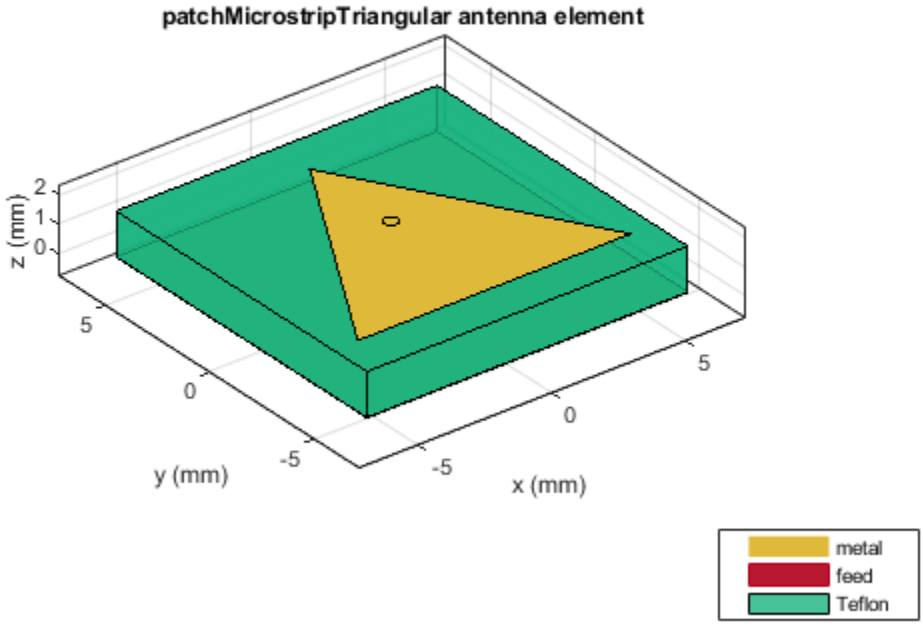


In the above figure, you will see that the first value of the side is chosen as the base of the triangle.

Triangle Patch Using Teflon Substrate and Radiation Pattern

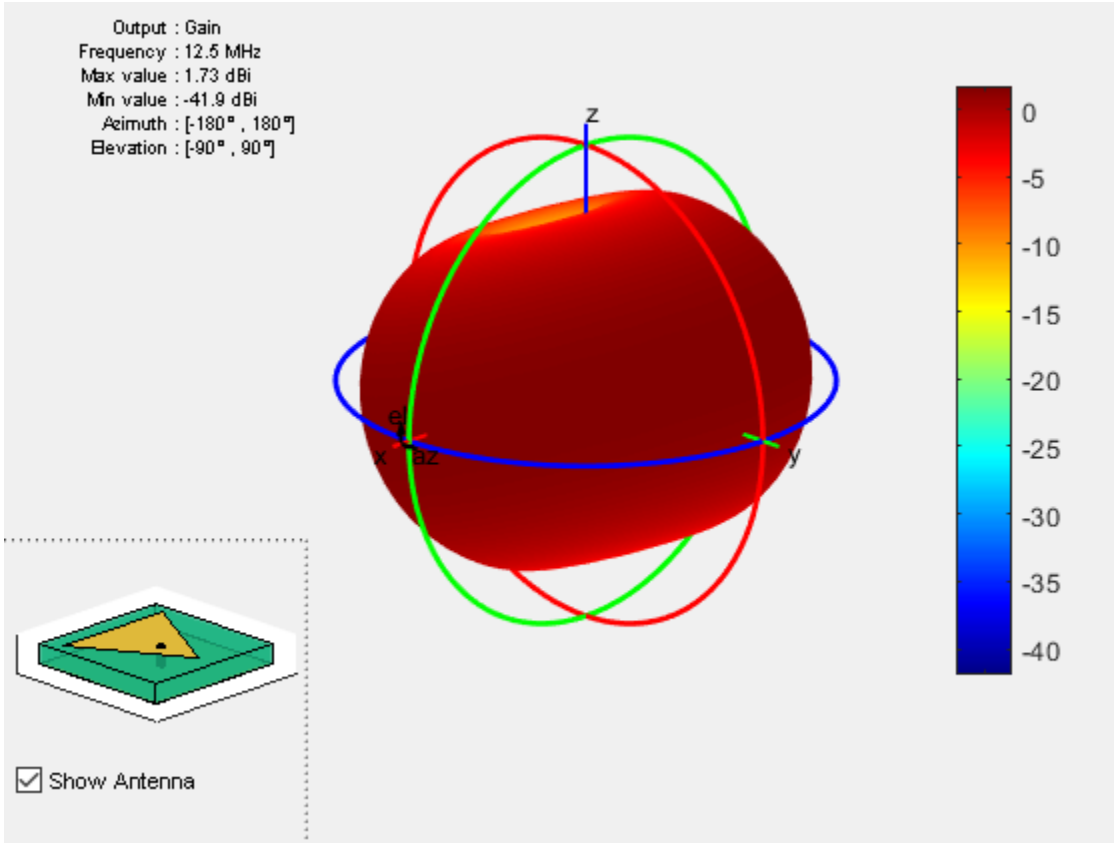
Create and view a triangular microstrip patch using Teflon substrate.

```
d = dielectric('Teflon');
p = patchMicrostripTriangular('Substrate',d);
show(p)
```



Plot the radiation pattern of the antenna at 12.5 GHz.

```
pattern(p,12.5e6)
```



See Also

`patchMicrostrip` | `patchMicrostripCircular` | `patchMicrostripEnotch`

Topics

“Rotate Antenna and Arrays”

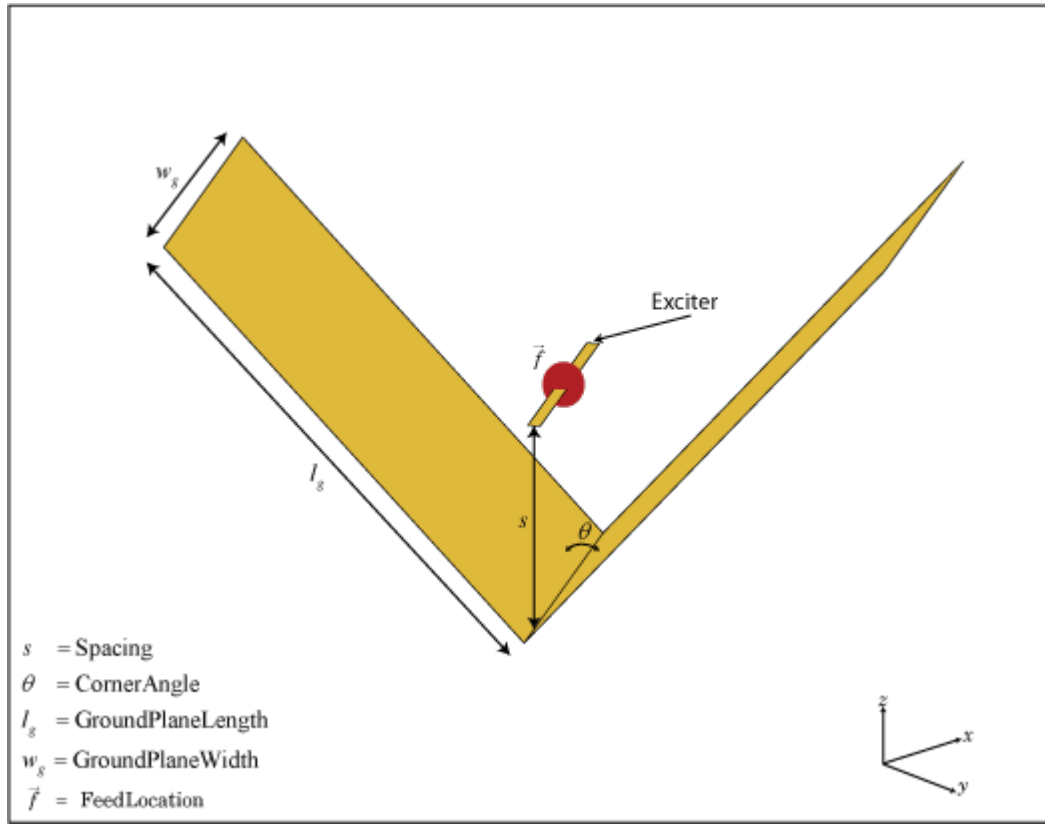
Introduced in R2018a

reflectorCorner

Create corner reflector-backed antenna

Description

Use the `reflectorCorner` object to create a corner reflector-backed antenna. By default, the exciter antenna is a dipole. The feedpoint of the dipole is at the origin. The default dimensions are for an operating frequency of 1 GHz.



Creation

Syntax

```
cornerreflector = reflectorCorner
cornerreflector = reflectorCorner(Name,Value)
```

Description

`cornerreflector = reflectorCorner` creates a corner reflector backed dipole antenna for an operating frequency of 1 GHz using default values.

`cornerreflector = reflectorCorner(Name,Value)` sets properties using one or more name-value pairs. For example, `cornerreflector = reflectorCorner('CornerAngle',45)` creates a corner reflector-backed antenna with a corner angle of 45 degrees. Enclose each property name in quotes.

Properties

`Exciter` — Antenna type used as exciter
dipole (default) | antenna object

Antenna type used as an exciter, specified as an antenna object. Except for reflector and cavity antenna elements, you can use any of the single elements in the Antenna Toolbox as an exciter.

Example: `'Exciter',spiralEquiangular`

Example: `cornerreflector.Exciter = spiralEquiangular`

`Spacing` — Distance between exciter and reflector
`0.0750` (default) | scalar

Distance between exciter and reflector, specified as a scalar in meters.

Example: `'Spacing',0.0624`

Example: `cornerreflector.Spacing = 0.0624`

Data Types: `double`

`CornerAngle` — Angle made by corner reflector

`90` (default) | scalar

Angle made by corner reflector, specified as a scalar in degrees.

Example: `'CornerAngle',60`

Example: `cornerreflector.CornerAngle = 60`

Data Types: `double`

`GroundPlaneLength` — Reflector length along X-axis

`0.2000` (default) | scalar

Reflector length along the X-axis, specified as a scalar in meters. By default, ground plane length is measured along the X-axis. You can also set the `'GroundPlaneLength'` to zero.

Example: `'GroundPlaneLength',0.4000`

Example: `cornerreflector.GroundPlaneLength = 0.4000`

Data Types: `double`

`GroundPlaneWidth` — Reflector width along Y-axis

`0.4000` (default) | scalar

Reflector width along the Y-axis, specified as a scalar in meters. By default, ground plane width is measured along the Y-axis. You can also set the `'GroundPlaneWidth'` to zero.

Example: `'GroundPlaneWidth',0.6000`

Example: `cornerreflector.GroundPlaneWidth = 0.6000`

Data Types: `double`

`Load` — Lumped elements

[1x1 `lumpedElement`] (default) | lumped element object

Loads added to the antenna feed, specified as a lumped element object. You can add a load anywhere on the surface of the antenna. By default, the load is at the origin. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`, where, `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `cornerreflector.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
`0` (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector in degrees.

Example: `'Tilt',90`

Example: `cornerreflector.Tilt = [90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency

EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

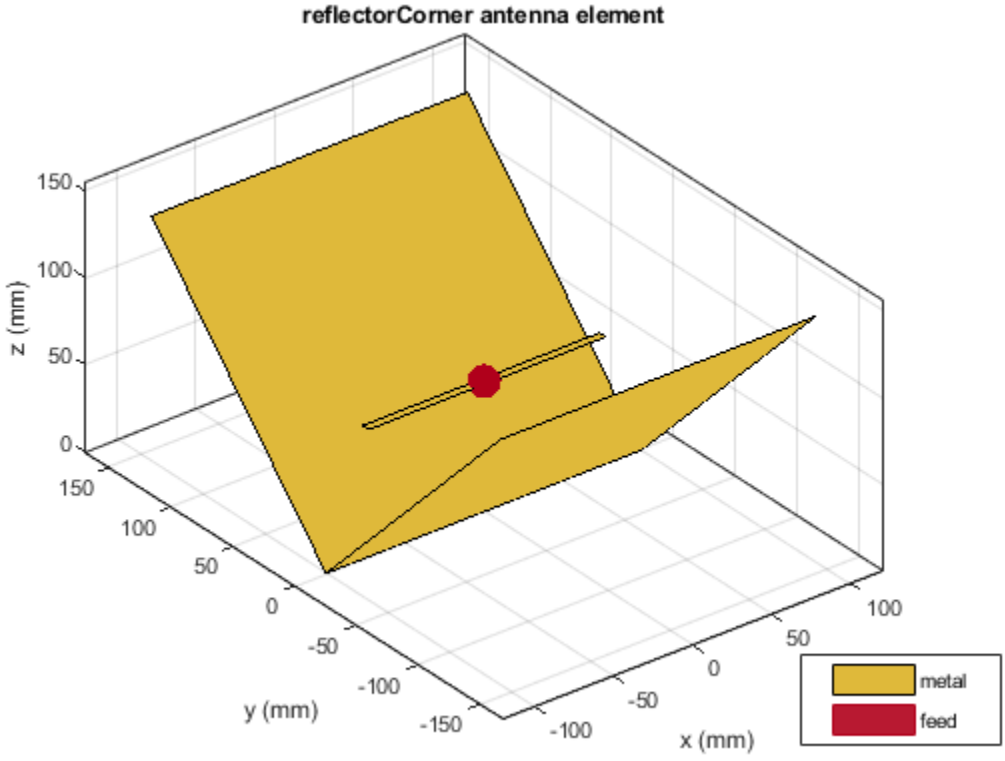
Default Corner Reflector-Backed Antenna and Radiation Pattern

Create and view a corner reflector-backed dipole.

```
cornerreflector = reflectorCorner
```

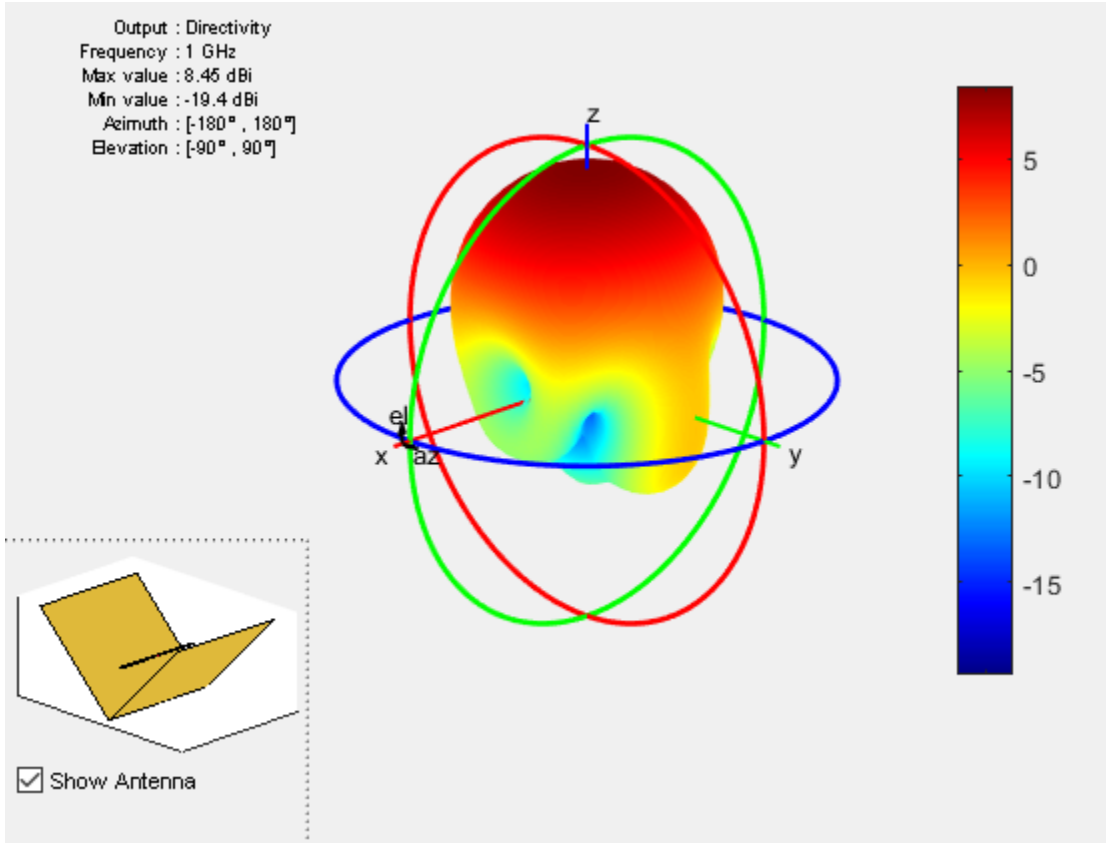
```
cornerreflector =
  reflectorCorner with properties:

              Exciter: [1x1 dipole]
    GroundPlaneLength: 0.2000
     GroundPlaneWidth: 0.4000
          CornerAngle: 90
              Spacing: 0.0750
                 Tilt: 0
            TiltAxis: [1 0 0]
                 Load: [1x1 lumpedElement]
```

```
show(cornerreflector)
```



Plot the radiation pattern at 1 GHz.

```
pattern(cornerreflector,1e9)
```



See Also

`reflector` | `reflectorCircular`

Topics

“Rotate Antenna and Arrays”

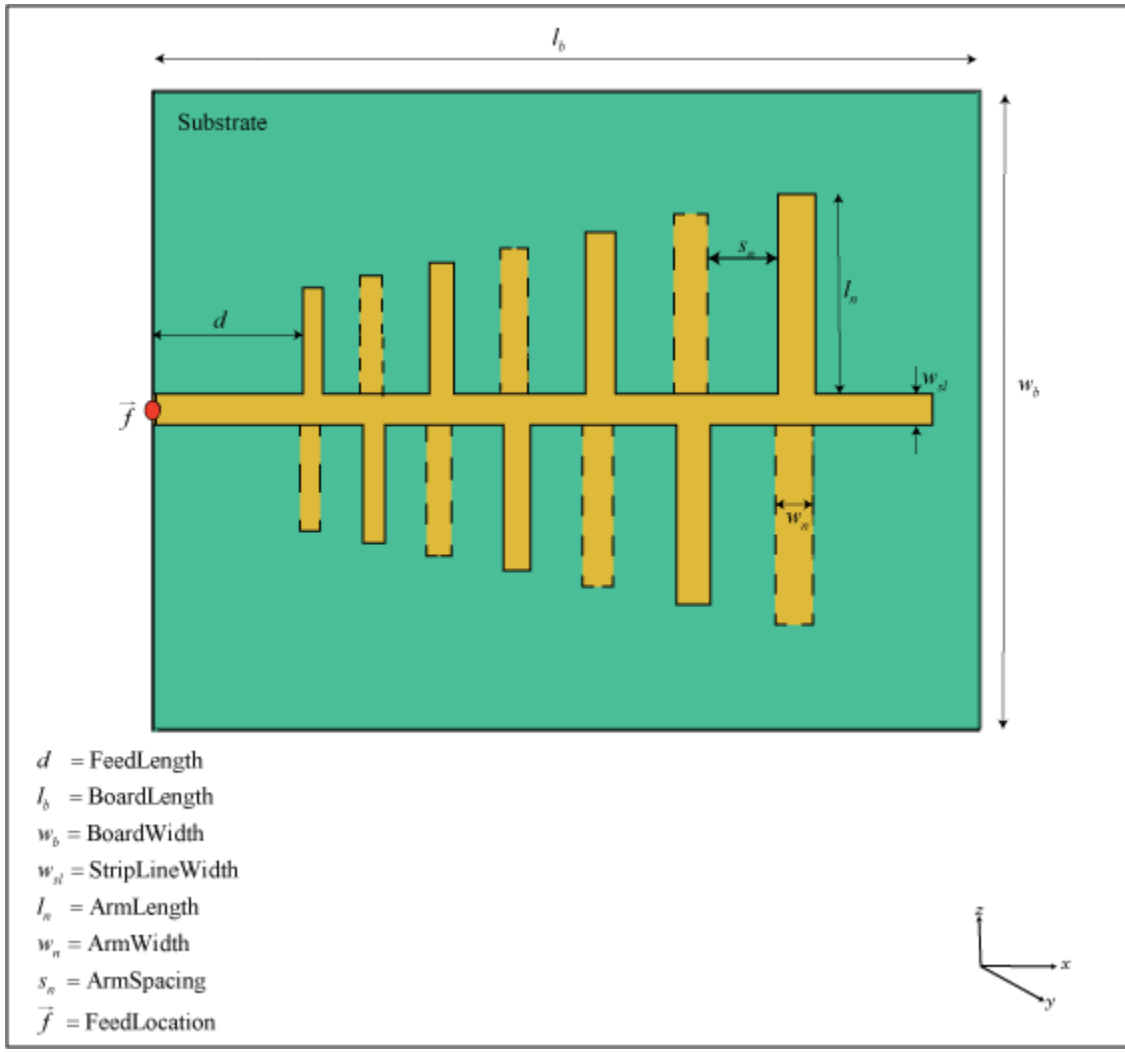
Introduced in R2018a

lpda

Create printed log-periodic dipole array antenna

Description

Use the `lpda` object to create a printed log-periodic dipole array antenna. The default antenna is centered at the origin and uses an FR4 substrate.


l_b
Substrate
s_n
l_n
d
w_{sl}
w_b
$\vec{f}$
w_n
d = FeedLength
l_b = BoardLength
w_b = BoardWidth
w_{sl} = StripLineWidth
l_n = ArmLength
w_n = ArmWidth
s_n = ArmSpacing
$\vec{f}$ = FeedLocation
z
x
y

Creation

Syntax

```
lpdipole = lpda
lpdipole = lpda(Name,Value)
```

Description

`lpdipole = lpda` creates a printed log-periodic dipole array antenna using default property values.

`lpdipole = lpda(Name,Value)` sets properties using one or more name-value pairs. For example, `lpdipole = lpdipole('BoardLength',0.2)` creates a printed log-periodic dipole array with a board length of 0.2 m. Enclose each property name in quotes.

Properties

`BoardLength` — PCB length along X-axis

`0.0366` (default) | scalar

Printed circuit board (PCB) length along X-axis, specified as a scalar in meters.

Example: `'BoardLength',0.2`

Example: `lpdipole.BoardLength = 0.2`

Data Types: `double`

`BoardWidth` — PCB width along Y-axis

`0.0244` (default) | scalar

PCB width along Y-axis, specified as a scalar in meters.

Example: `'BoardWidth',0.06`

Example: `lpdipole.BoardWidth = 0.06`

Data Types: `double`

`Height` — PCB height along Z-axis
`0.0016` (default) | scalar

PCB height along Z-axis, specified as a scalar in meters.

Example: `'Height',0.0018`

Example: `lpdipole.Height = 0.0018`

Data Types: `double`

`StripLineWidth` — Parallel strip line width
`0.0012` (default) | scalar

Parallel strip line width, specified as a scalar in meters.

Example: `'StripLineWidth',0.0014`

Example: `lpdipole.StripLineWidth = 0.0014`

Data Types: `double`

`FeedLength` — Distance from edge feed point to smallest dipole
`0.0065` (default) | scalar

Distance from edge feed point to smallest dipole, specified as a scalar in meters.

Example: `'FeedLength',0.0055`

Example: `lpdipole.FeedLength = 0.0055`

Data Types: `double`

`ArmLength` — Lengths of individual dipole arms
`[0.0040 0.0045 0.0050 0.0056 0.0062 0.0069 0.0076 0.0085]` (default) | vector

Lengths of individual dipole arms, specified as a vector with each element unit in meters.

Example: `'ArmLength',[0.0050 0.0055 0.0060 0.0066 0.0072 0.0079 0.0086 0.0095]`

Example: `lpdipole.ArmLength = [0.0050 0.0055 0.0060 0.0066 0.0072 0.0079 0.0086 0.0095]`

Data Types: `double`

`ArmWidth` — Widths of individual dipole arms

`[8.8000e-04 9.8000e-04 0.0011 0.0012 0.0013 0.0015 0.0017 0.0019]` (default) | vector

Widths of individual dipole arms, specified as a vector with each element unit in meters.

Example: `'ArmWidth',[9.8000e-04 10.8000e-04 0.0021 0.0022 0.0023 0.0025 0.0027 0.0029]`

Example: `lpdipole.ArmWidth = [9.8000e-04 10.8000e-04 0.0021 0.0022 0.0023 0.0025 0.0027 0.0029]`

Data Types: `double`

`ArmSpacing` — Spacing between individual dipole arms

`[0.0027 0.0030 0.0033 0.0037 0.0041 0.0046 0.0051]` (default) | vector

Spacing between individual dipole arms, specified as a vector with each element unit in meters.

Example: `'ArmSpacing',[0.0037 0.0040 0.0043 0.0047 0.0051 0.0056 0.0061]`

Example: `lpdipole.ArmSpacing = [0.0037 0.0040 0.0043 0.0047 0.0051 0.0056 0.0061]`

Data Types: `double`

`Substrate` — Type of dielectric material

`'FR4'` (default) | dielectric object

Type of dielectric material used as a substrate, specified as an dielectric object. For more information, see `dielectric`. For more information on dielectric substrate meshing, see “Meshing”.

Note The substrate dimensions must be equal to the groundplane dimensions.

Example: `d = dielectric('Teflon'); 'Substrate',d`

Example: `d = dielectric('Teflon'); lpdipole.Substrate = d`

Load — Lumped elements

[1x1 `lumpedElement`] (default) | lumped element object

Lumped elements added to the antenna feed, specified as a lumped element object. You can add a load anywhere on the surface of the antenna. By default, the load is at the origin. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`, where `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `lpda.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna

0 (default) | scalar | vector

Tilt angle of antenna, specified as a scalar or vector in degrees.

Example: `'Tilt',90`

Example: `lpdipole.Tilt = [90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

Default Printed Log-Periodic Antenna

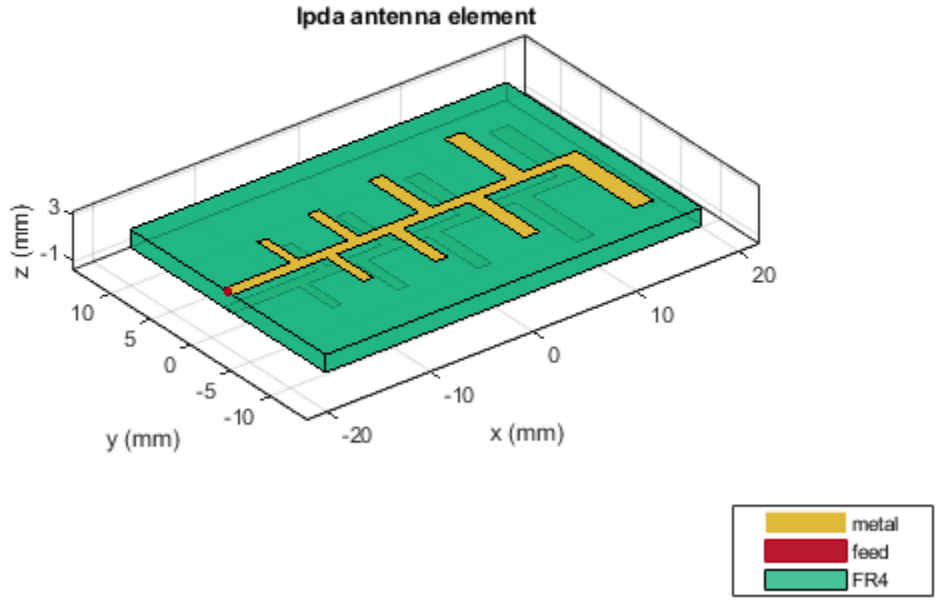
Create and view a printed log-periodic dipole array antenna.

```
lpdipole = lpda
```

```
lpdipole =
  lpda with properties:

       BoardLength: 0.0366
        BoardWidth: 0.0244
            Height: 0.0016
    StripLineWidth: 0.0012
        FeedLength: 0.0065
         ArmLength: [0.0040 0.0045 0.0050 0.0056 0.0062 0.0069 0.0076 0.0085]
          ArmWidth: [1x8 double]
```

```
        ArmSpacing: [0.0027 0.0030 0.0033 0.0037 0.0041 0.0046 0.0051]
         Substrate: [1x1 dielectric]
              Tilt: 0
           TiltAxis: [1 0 0]
              Load: [1x1 lumpedElement]
```

```
show(lpdipole)
```



See Also

`pcbStack` | `yagiUda`

Topics

“Rotate Antenna and Arrays”

Introduced in R2018a

helixMultifilar

Creates bifilar or quadrafilar helix antenna on circular ground plane

Description

The `helixMultifilar` object creates a bifilar or quadrafilar helix antenna on a circular ground plane. You can create both short-circuited and open-ended helix multifilar antennas. Bifilar and quadrafilar helix antennas are used in aerospace and defense applications.

The width of the strip is related to the diameter of an equivalent cylinder by the equation

$$w = 2d = 4r$$

where:

- w is the width of the strip.
- d is the diameter of an equivalent cylinder.
- r is the radius of an equivalent cylinder.

For a given cylinder radius, use the `cylinder2strip` utility function to calculate the equivalent width. The default helix antenna is end-fed. The circular ground plane is on the X-Y plane. Helix antennas are commonly used in axial mode. In this mode, the helix circumference is comparable to the operating wavelength, and the helix has maximum directivity along its axis. In normal mode, helix radius is small compared to the operating wavelength. In this mode, the helix radiates broadside, that is, in the plane perpendicular to its axis. The basic equations for the helix are

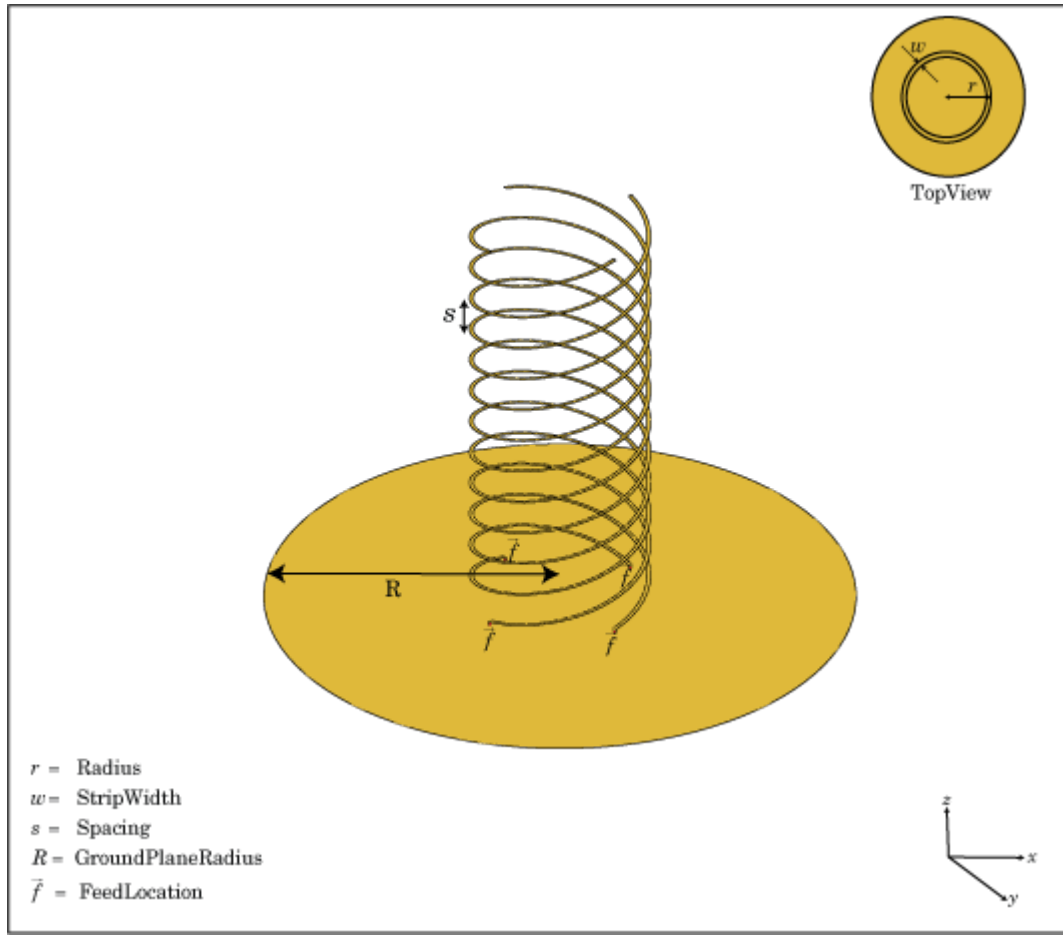
$$x = r\cos(\theta)$$
$$y = r\sin(\theta)$$
$$z = S\theta$$

where:

- r is the radius of the helical dipole.

- θ is the winding angle.
- S is the spacing between turns.

For a given pitch angle in degrees, use the `helixpitch2spacing` utility function to calculate the spacing between the turns in meters.



Creation

Syntax

```
ant = helixMultifilar
ant = helixMultifilar(Name,Value)
```

Description

`ant = helixMultifilar` creates a bifilar or quadrafilar helix antenna operating in the axial mode. The default multifilar helical antenna is end-fed and has a circular ground plane on the X-Y plane. The default operating frequency is around 2 GHz.

`ant = helixMultifilar(Name,Value)` sets properties using one or more name-value pairs. For example, `ant = helixMultifilar('Radius',28e-03)` creates a multifilar helix with turns of radius 28e-03 m.

Output Arguments

`ant` — Multifilar helix antenna
`helixMultifilar` object

Multifilar helix antenna, returned as a `helixMultifilar` object.

Properties

`NumArms` — Number of helical elements
4 (default) | 2

Number of helical elements, specified as 4 or 2. Specify two elements to create a bifilar helix antenna, and four elements to create a quadrafilar helix antenna.

Example: `'NumArms',2`

Example: `ant.NumArms = 2`

Data Types: `double`

`Radius` — Radius of turns
`0.0220` (default) | positive scalar integer

Radius of the turns, specified as a positive scalar integer in meters.

Example: `'Radius',28e-03`

Example: `ant.Radius = 28e-03`

Data Types: `double`

`Width` — Width of strip
`1000e-03` (default) | positive scalar integer

Width of the strip, specified as a positive scalar integer in meters.

Example: `'Width',0.2`

Example: `ant.Width = 0.2`

Data Types: `double`

`Turns` — Number of turns
`3` (default) | scalar integer

Number of turns, specified as a scalar integer.

Example: `'Turns',4`

Example: `ant.Turns = 4`

Data Types: `double`

`Spacing` — Spacing between turns
`0.0350` (default) | positive scalar integer

Spacing between the turns, specified as a positive scalar integer in meters.

Example: `'Spacing',7.5e-2`

Example: `ant.Spacing = 7.5e-2`

Data Types: `double`

`ShortEnds` — Status of helix ends
`0` (default) | `1`

Status of helix ends, specified as `0` or `1`. By default, the `helixMultifilar` is an open circuit. Setting the property to `1` makes the helix antenna short circuit.

Example: `'ShortEnds',1`

Example: `ant.ShortEnds = 1`

Data Types: `double`

`WindingDirection` — Direction of helix turns (windings)
`'CW'` | `'CCW'`

Direction of the helix turns (windings), specified as `'CW'` for clockwise or `'CCW'` for counter-clockwise.

Example: `'WindingDirection','CW'`

Example: `ant.WindingDirection = 'CW'`

Data Types: `char` | `string`

`FeedStubHeight` — Height of feeding stub from ground plane
`1.000e-03` (default) | positive scalar integer

Height of the feeding stub from the ground plane, specified as a positive scalar integer in meters.

Example: `'FeedStubHeight',7.5e-2`

Example: `ant.FeedStubHeight = 7.5e-2`

Data Types: `double`

`GroundPlaneRadius` — Ground plane radius
`0.0750` (default) | positive scalar integer

Ground plane radius, specified as a positive scalar integer in meters. By default, the ground plane is on the X-Y plane and is symmetrical about the origin.

Setting this value to `Inf` uses the infinite ground plane technique for antenna analysis.

Example: `'GroundPlaneRadius',2.05`

Example: `ant.GroundPlaneRadius = 7.5e-2`

Data Types: `double`

`FeedVoltage` — Excitation voltage applied to individual antenna feeds
1 (default) | scalar integer | vector integers

Excitation voltage applied to individual antenna feeds, specified as a scalar integer or vector integers. A scalar value applies the same voltage to all feeds.

Example: `'FeedVoltage',[1 2]`

Example: `ant.FeedVoltage = [1 2]`

Data Types: `double`

`FeedPhase` — Excitation voltage phase applied to individual antenna feeds
0 (default) | scalar integer | vector integers

Excitation voltage phase applied to individual antenna feeds, specified as a scalar integer or vector integers. A scalar value applies the same voltage phase to all feeds.

Example: `'FeedPhase',[0 45]`

Example: `ant.FeedPhase = [0 45]`

Data Types: `double`

`Load` — Lumped elements
[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. You can add a load anywhere on the surface of the antenna. By default, the load is at the origin. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `ant.Load = lumpedElement('Impedance',75)`

Data Types: `double`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of the antenna, specified as a scalar or vector in degrees.

Example: `'Tilt',90`

Example: `ant.Tilt = [90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna

`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array

returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

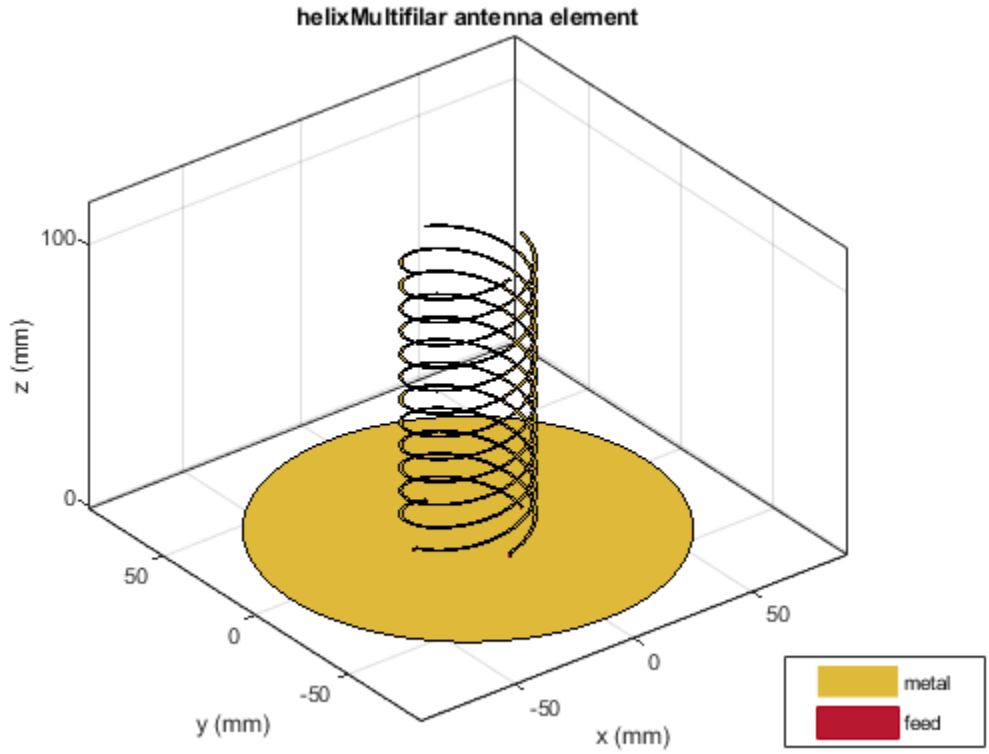
Quadrafilar Helix

Create and view a Quadrafilar helix antenna.

```
ant = helixMultifilar
```

```
ant =
  helixMultifilar with properties:

             NumArms: 4
              Radius: 0.0220
               Width: 1.0000e-03
               Turns: 3
             Spacing: 0.0350
           ShortEnds: 0
    WindingDirection: 'CCW'
      FeedStubHeight: 1.0000e-03
   GroundPlaneRadius: 0.0750
         FeedVoltage: 1
           FeedPhase: 0
                Tilt: 0
            TiltAxis: [1 0 0]
                Load: [1×1 lumpedElement]
```

```
show(ant)
```



Bifilar Helix

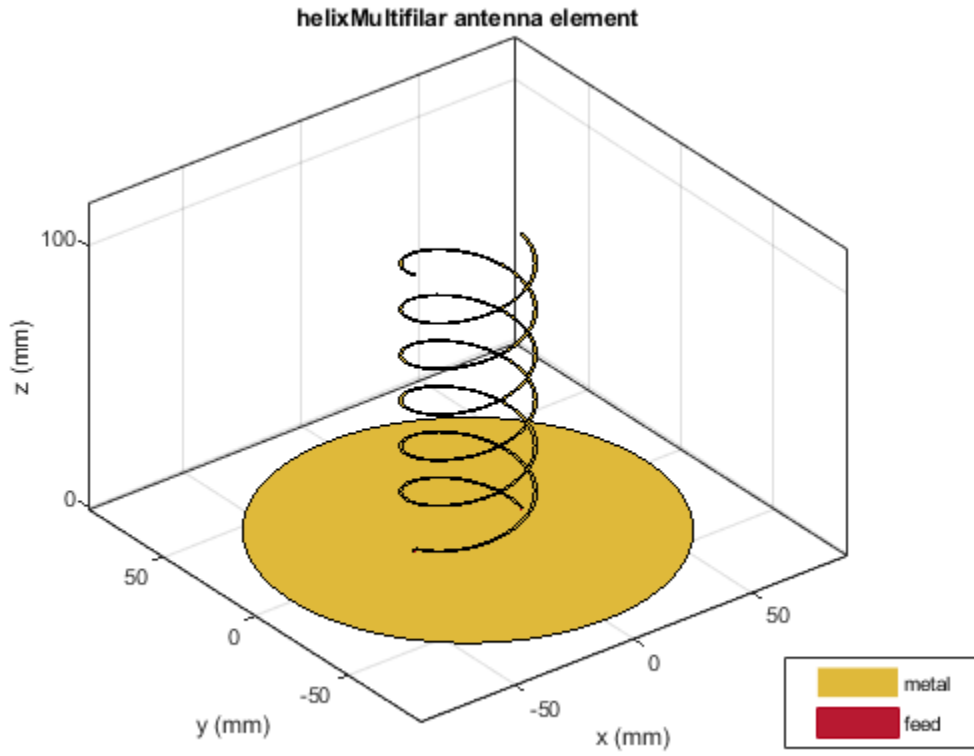
Create and view a bifilar helix antenna.

```
ant=helixMultifilar('NumArms',2)
```

```
ant =
  helixMultifilar with properties:

             NumArms: 2
              Radius: 0.0220
               Width: 1.0000e-03
```

```
              Turns: 3
            Spacing: 0.0350
          ShortEnds: 0
   WindingDirection: 'CCW'
     FeedStubHeight: 1.0000e-03
  GroundPlaneRadius: 0.0750
        FeedVoltage: 1
          FeedPhase: 0
               Tilt: 0
           TiltAxis: [1 0 0]
               Load: [1x1 lumpedElement]
```

```
show(ant)
```



See Also

`cylinder2strip` | `dipoleHelix` | `dipolehelixMultifilar` | `helix` | `helixpitch2spacing`

Introduced in R2018b

dipoleHelixMultifilar

Create balanced bifilar or quadrafilar dipole helix antenna without circular ground plane

Description

The `dipoleHelixMultifilar` object creates a balanced bifilar or quadrafilar helix antenna without a circular ground plane. You can create both short-circuited and open-ended dipole helix multifilar antennas. Bifilar and quadrafilar helix antennas are used in aerospace and defense applications.

The width of the strip is related to the diameter of an equivalent cylinder by the equation

$$w = 2d = 4r$$

where:

- w is the width of the strip.
- d is the diameter of an equivalent cylinder.
- r is the radius of an equivalent cylinder.

For a given cylinder radius, use the `cylinder2strip` utility function to calculate the equivalent width. The default helix antenna is end-fed. The circular ground plane is on the X-Y plane. Helix antennas are used commonly in axial mode. In this mode, the helix circumference is comparable to the operating wavelength, and the helix has maximum directivity along its axis. In normal mode, the helix radius is small compared to the operating wavelength. In this mode, the helix radiates broadside, that is, in the plane perpendicular to its axis. The basic equations for the helix are

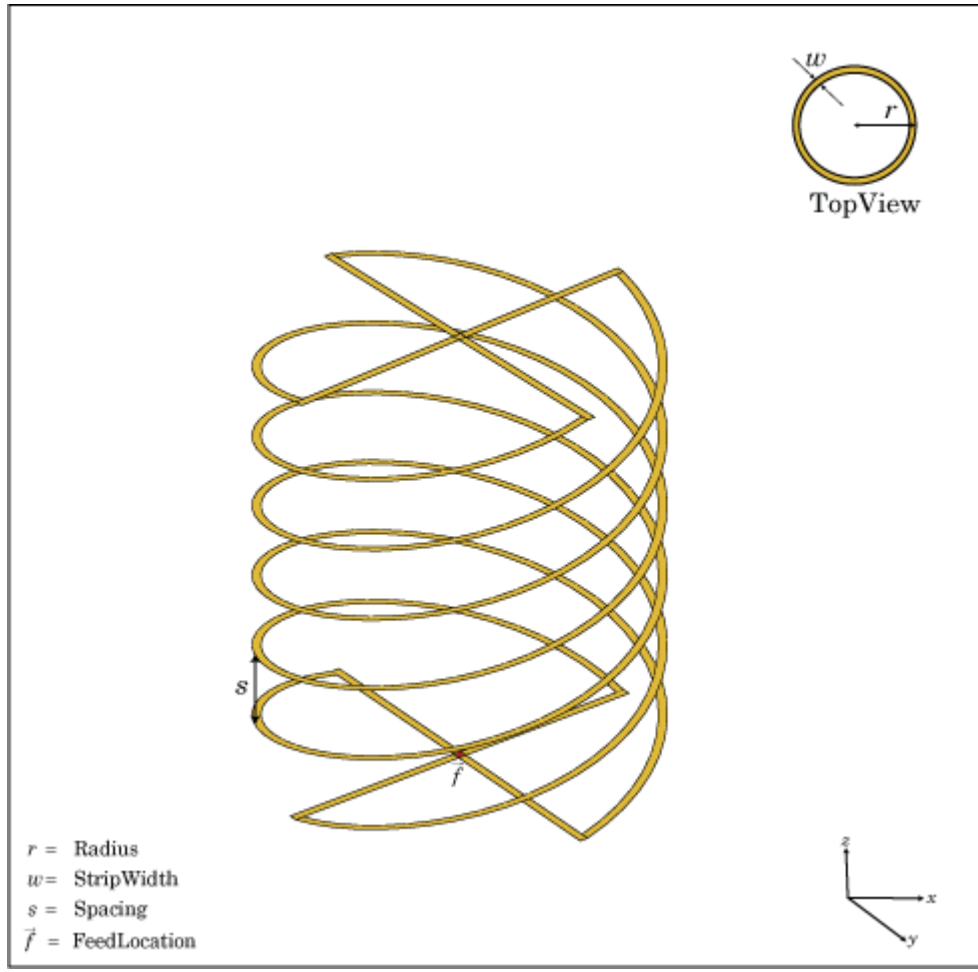
$$x = r\cos(\theta)$$
$$y = r\sin(\theta)$$
$$z = S\theta$$

where:

- r is the radius of the helical dipole.

- θ is the winding angle.
- S is the spacing between turns.

For a given pitch angle in degrees, use the `helixpitch2spacing` utility function to calculate the spacing between the turns in meters.



Creation

Syntax

```
ant = dipoleHelixMultifilar
ant = dipoleHelixMultifilar(Name,Value)
```

Description

`ant = dipoleHelixMultifilar` creates a bifilar or quadrafilar helix antenna without a circular ground plane. The default multifilar helical antenna is end-fed. The default helix operates around 2 GHz.

`ant = dipoleHelixMultifilar(Name,Value)` sets properties using one or more name-value pairs. For example, `ant = dipoleHelixMultifilar('Radius',28e-03)` creates a multifilar helix with turns of radius $28e^{-03}$ m. Enclose each property name in quotes.

Output Arguments

`ant` — Dipole multifilar helix antenna
`dipoleHelixMultifilar` object

Dipole multifilar helix antenna, returned as a `dipoleHelixMultifilar` object.

Properties

`NumArms` — Number of helical elements
4 (default) | 2

Number of helical elements, specified as a 4 or 2. Two elements create a bifilar dipole helix antenna, and four elements create a quadrafilar dipole helix antenna.

Example: `'NumArms',2`

Example: `ant.NumArms = 2`

Data Types: `double`

`Radius` — Radius of turns
`0.0220` (default) | positive real scalar

Radius of the turns, specified as a positive real scalar meter.

Example: `'Radius',28e-03`

Example: `ant.Radius = 28e-03`

Data Types: `double`

`Width` — Width of strip
`1.000e-03` (default) | positive real scalar

Width of the strip, specified as a positive real scalar in meters.

Example: `'Width',0.2`

Example: `ant.Width = 0.2`

Data Types: `double`

`Turns` — Number of turns
`3` (default) | scalar integer

Number of turns, specified as a scalar integer.

Example: `'Turns',4`

Example: `ant.Turns = 4`

Data Types: `double`

`Spacing` — Spacing between turns
`0.0350` (default) | positive real scalar

Spacing between the turns, specified as a positive real scalar in meters.

Example: `'Spacing',7.5e-2`

Example: `ant.Spacing = 7.5e-2`

Data Types: `double`

`ShortEnds` — Status of ends of helix
`1` (default) | `0`

Status of ends of the helix, specified as `0` or `1`. By default, the `dipoleHelixMultifilar` is short circuited. Setting the property to `0` makes the helix antenna an open circuit.

Example: `'ShortEnds',0`

Example: `ant.ShortEnds = 0`

Data Types: `double`

`WindingDirection` — Direction of helix turns (windings)
`'CCW'` (default) | `'CW'`

Direction of helix turns (windings), specified as `CW` or `CCW`.

Example: `'WindingDirection','CW'`

Example: `ant.WindingDirection = 'CW'`

Data Types: `char` | `string`

`Load` — Lumped elements
[1x1 LumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. You can add a load anywhere on the surface of the antenna. By default, the load is at the origin. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `ant.Load = lumpedElement('Impedance',75)`

Data Types: `double`

`Tilt` — Tilt angle of antenna
0 (default) | scalar | vector

Tilt angle of the antenna, specified as a scalar or vector in degrees.

Example: `'Tilt',90`

Example: `ant.Tilt = [90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

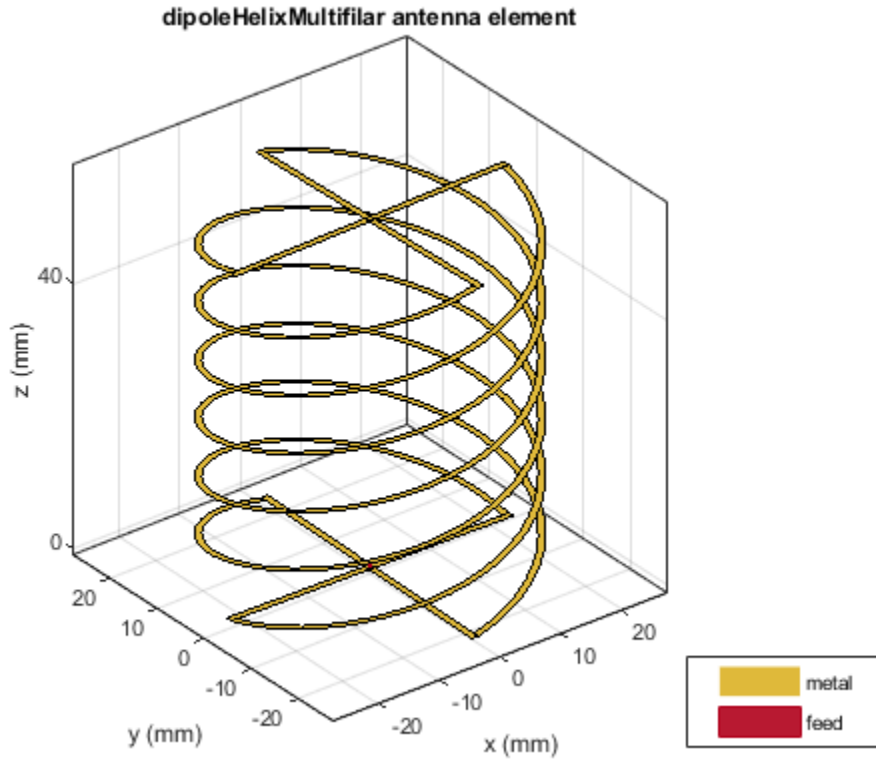
Default Multifilar Helical Dipole Antenna

Create and view a default multifilar helical dipole antenna.

```
ant = dipoleHelixMultifilar
```

```
ant =
  dipoleHelixMultifilar with properties:

             NumArms: 4
              Radius: 0.0220
               Width: 1.0000e-03
               Turns: 3
             Spacing: 0.0350
           ShortEnds: 1
    WindingDirection: 'CCW'
                Tilt: 0
            TiltAxis: [1 0 0]
                Load: [1×1 lumpedElement]
```

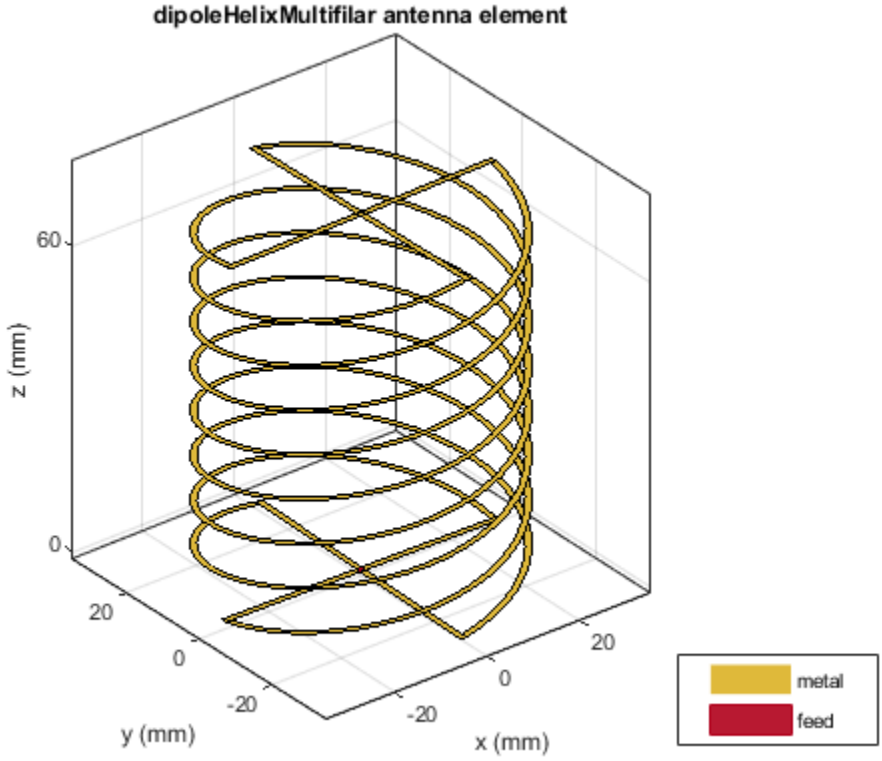
```
show(ant)
```



Quadrafilar Helical Dipole Antenna and Radiation Pattern

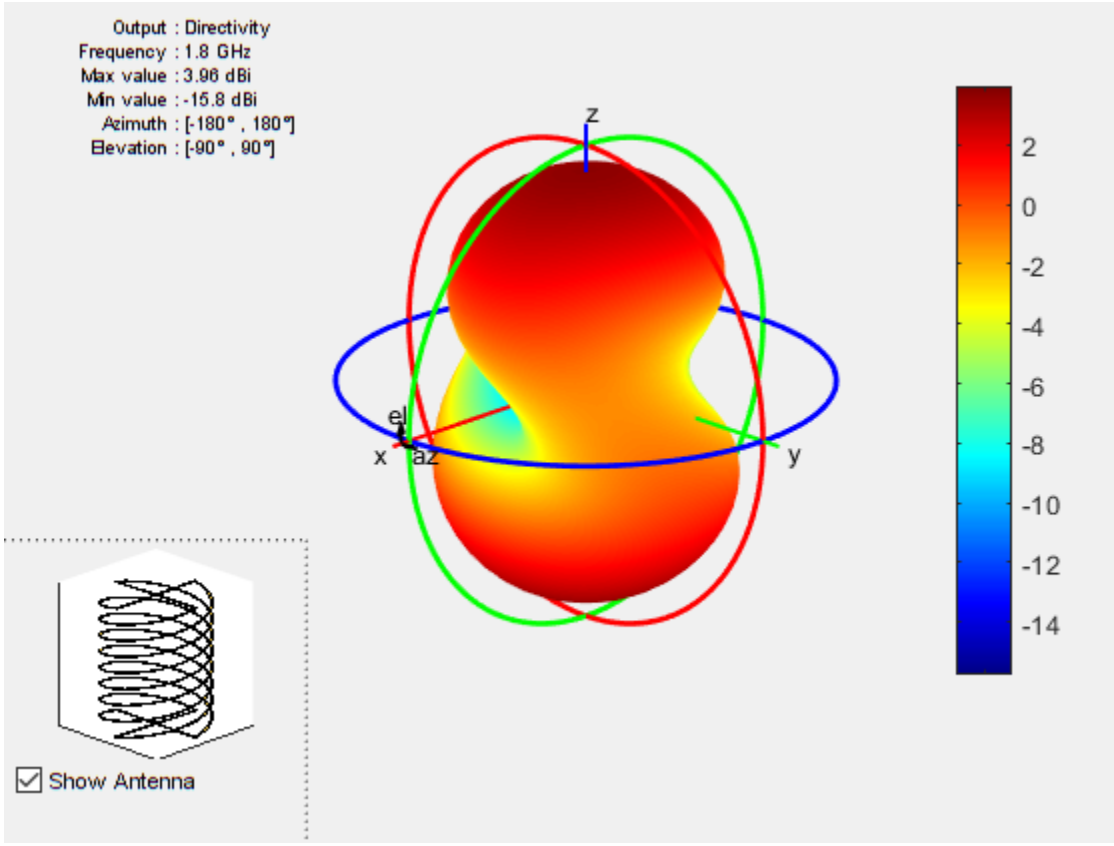
Create and view a quadrafilar helical dipole antenna with turn radius of 28 mm and strip width of 1.2 mm.

```
ant = dipoleHelixMultifilar('Radius',28e-3,'Width',1.2e-3,'Turns',4);
show(ant)
```



Plot the radiation pattern of the helical dipole at 1.8 GHz.

```
pattern(ant,1.8e9);
```



See Also

`dipoleHelix` | `helix` | `helixMultifilar`

Introduced in R2018b

fractalGasket

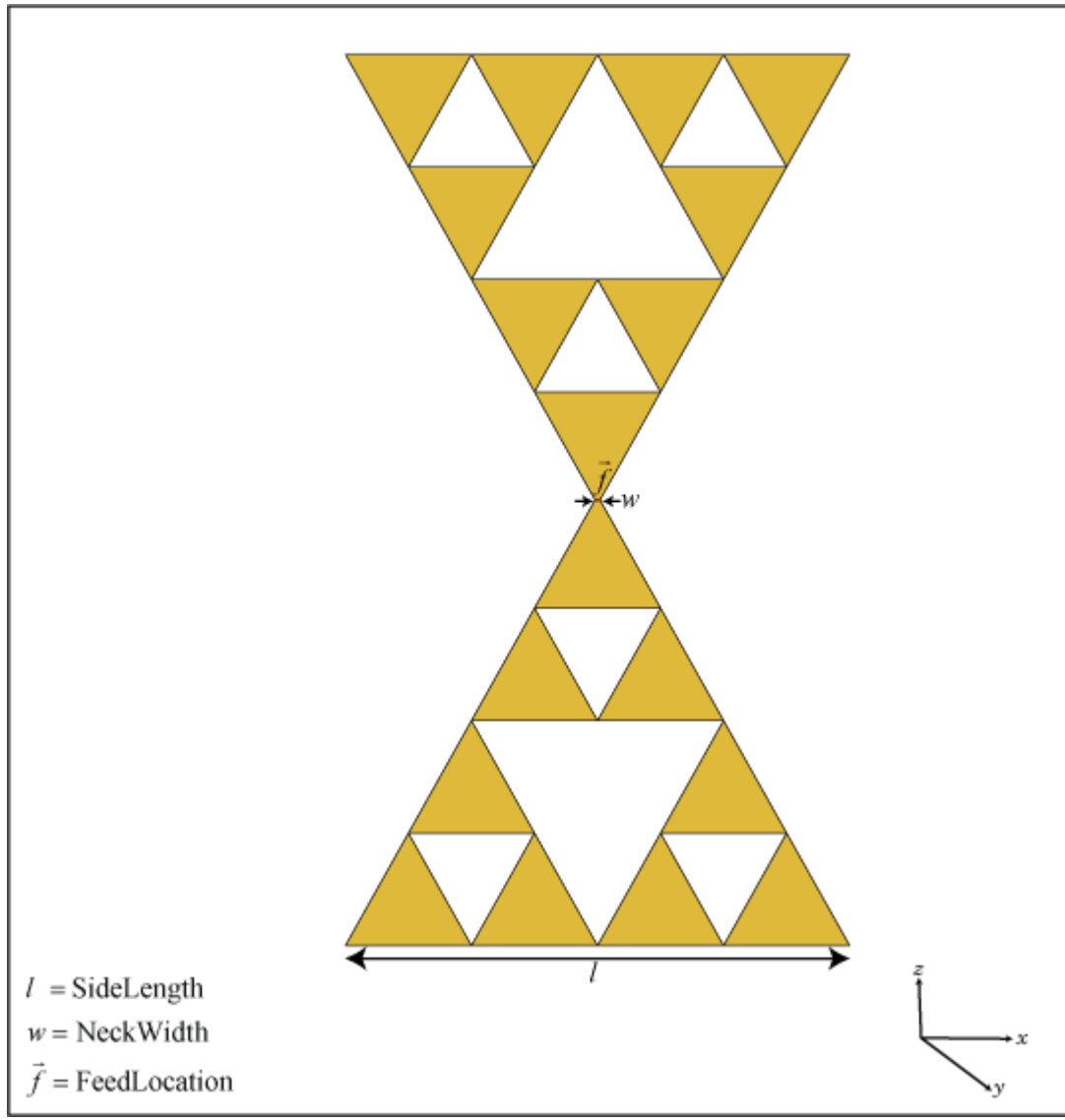
Create Sierpinski's Gasket fractal antenna on X-Y plane

Description

The `fractalGasket` object creates an equilateral triangle-shaped Sierpinski's Gasket fractal antenna. These fractals are used in building communications systems, wireless networks, universal tactic communications systems, mobile devices, telematics, and radio frequency identification (RFID) antennas.

A fractal antenna uses a self-similar design to maximize the length or increase the perimeter of a material that transmits or receives electromagnetic radiation within a given volume or area. The main advantage of fractal antennas is that they are compact, which is important requirement for small and complex circuits. Fractal antennas also have more input impedance or resistance due to increased length or perimeter.

All fractal antennas are printed structures that are etched on a dielectric substrate.


w
l
z
x
y
l = SideLength
w = NeckWidth
f = FeedLocation

Creation

Syntax

```
ant = fractalGasket
ant = fractalGasket(Name,Value)
```

Description

`ant = fractalGasket` creates an equilateral triangle-shaped Sierpinski's gasket fractal antenna. The default planar fractal antenna is in the shape of a bowtie which is center-fed. The antenna resonates at a frequency of 1.3 GHz.

`ant = fractalGasket(Name,Value)` sets properties using one or more name-value pairs. For example, `ant = fractalGasket('Numiterations',4)` creates a Sierpinski's Gasket with four iterations.

Output Arguments

`ant` — Sierpinski's Gasket fractal antenna

`fractalGasket` object

Sierpinski's Gasket fractal antenna, returned as a `fractalGasket` object.

Properties

`NumIterations` — Number of iterations of fractal antenna

2 (default) | scalar integer

Number of iterations of the fractal antenna, specified as a scalar integer.

Example: `'NumIterations',2`

Example: `ant.NumIterations = 2`

Data Types: `double`

`Side` — Lengths for three sides of triangle

`0.2000` (default) | scalar | two-element vector | three-element vector

Lengths for three sides of the triangle, specified as a scalar in meters or a two- or three-element vector in meters.

- Scalar – The triangle is equilateral.
- Two-element vector – The first value specifies the base of the triangle along the X-axis. The second value specifies the other two sides of the triangle. The triangle is isosceles.
- Three-element vector – The first value specifies the base of the triangle along the X-axis. The remaining two values specify the other two sides of the triangle. The triangle is scalene.

Example: `'Side',[0.5000,1.000]`

Example: `ant.Side = [0.5000,1.000]`

Data Types: `double`

`NeckWidth` — Width at neck of fractal antenna
`0.0020` (default) | positive scalar integer

Width at the neck of the fractal antenna where the feed is located, specified as a positive scalar integer in meters.

Example: `'NeckWidth',0.0050`

Example: `ant.NeckWidth = 0.0050`

Data Types: `double`

`Load` — Lumped elements
[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. You can add a load anywhere on the surface of the antenna. By default, the load is at the origin. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `ant.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
`0` (default) | scalar | vector

Tilt angle of the antenna, specified as a scalar or vector in degrees.

Example: `'Tilt',90`

Example: `ant.Tilt = [90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vector of Cartesian coordinates | two three-element vectors of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure

meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

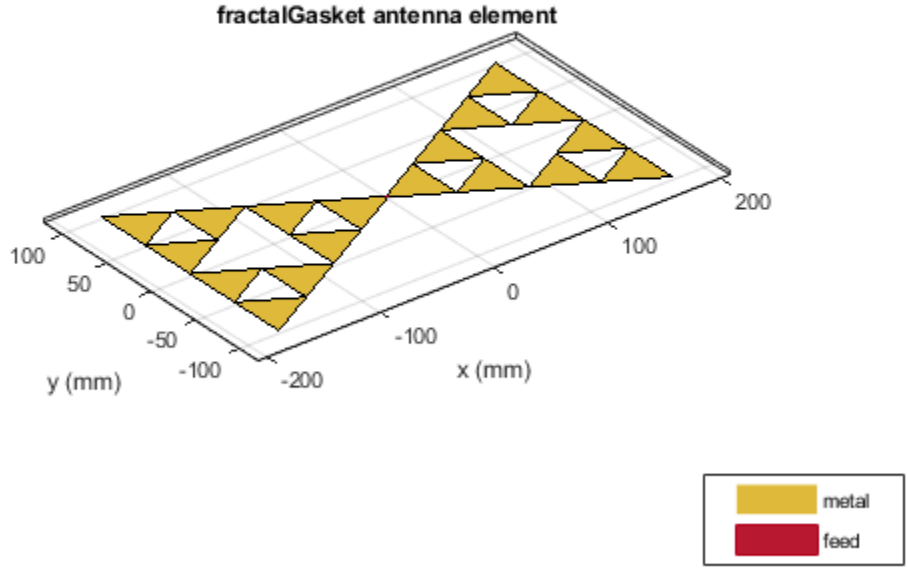
Default Sierpinski's Gasket

Create and view a default fractal Sierpinski's Gasket.

```
ant = fractalGasket
```

```
ant =
  fractalGasket with properties:

    NumIterations: 2
             Side: 0.2000
        NeckWidth: 0.0020
             Tilt: 0
         TiltAxis: [1 0 0]
             Load: [1x1 lumpedElement]
```

```
show(ant)
```



See Also

`fractalKoch`

Topics

“Rotate Antenna and Arrays”

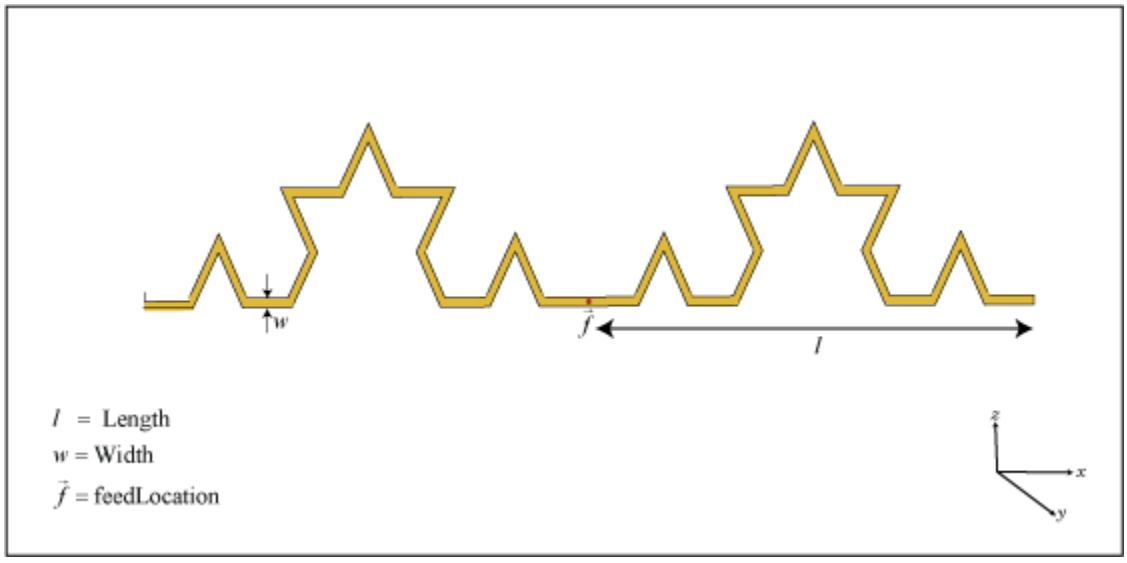
Introduced in R2018b

fractalKoch

Create Koch curve fractal dipole or loop antenna on X-Y plane

Description

The `fractalKoch` object creates a Koch curve fractal dipole or loop antenna on an X-Y plane. These fractals are used in multiband and wideband applications like Global System for Mobile Communications (GSM), Universal Mobile Telecommunication Service (UMTS), and Bluetooth.



A fractal antenna uses a self-similar design to maximize the length or increase the perimeter of a material that transmits or receives electromagnetic radiation within a given volume or area. The main advantage of fractal antennas is that they are compact, which is an important requirement for small and complex circuits. Fractal antennas also have more input impedance or resistance due to increased length or perimeter, respectively.

All fractal antennas are printed structures that are etched on a dielectric substrate.

Creation

Syntax

```
ant = fractalKoch
ant = fractalKoch(Name,Value)
```

Description

`ant = fractalKoch` creates a Koch curve fractal antenna on an X-Y plane. The default is a dipole with Koch curve length chosen for an operating frequency of 0.86 GHz.

`ant = fractalKoch(Name,Value)` sets properties using one or more name-value pairs. For example, `ant = fractalKoch('Numiterations',4)` creates a Koch curve fractal antenna with four iterations. Enclose each property name in quotes.

Output Arguments

`ant` — Koch curve fractal antenna

`fractalKoch` object (default)

Koch curve fractal antenna, returned as a `fractalKoch` object.

Properties

`NumIterations` — Number of iterations of fractal antenna

2 (default) | scalar integer

Number of iterations of the fractal antenna, specified as a scalar integer.

Example: `'NumIterations',2`

Example: `ant.NumIterations = 2`

Data Types: `double`

`Length` — Length of Koch curve along X-axis

`0.0600` (default) | positive scalar integer

Length of the Koch curve along the x-axis, specified as a positive scalar integer in meters.

Example: `'Length',0.5000`

Example: `ant.Length = 0.5000`

Data Types: `double`

`Width` — Width of Koch curve along Y-axis
`1.0000e-03` (default) | positive scalar integer

Width of the Koch curve along y-axis, specified as a positive scalar integer in meters.

Example: `'Width',0.0050`

Example: `ant.Width = 0.0050`

Data Types: `double`

`Type` — Type of Koch configuration
`'dipole'` (default) | `'loop'`

Type of Koch configuration, specified as `'dipole'` or `'loop'`.

Example: `'Type','loop'`

Example: `ant.Type = 'loop'`

Data Types: `char` | `string`

`Load` — Lumped elements
[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. You can add a load anywhere on the surface of the antenna. By default, the load is at the origin. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `ant.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
`0` (default) | scalar | vector

Tilt angle of the antenna, specified as a scalar or vector in degrees.

Example: `'Tilt',90`

Example: `ant.Tilt = [90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure

pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

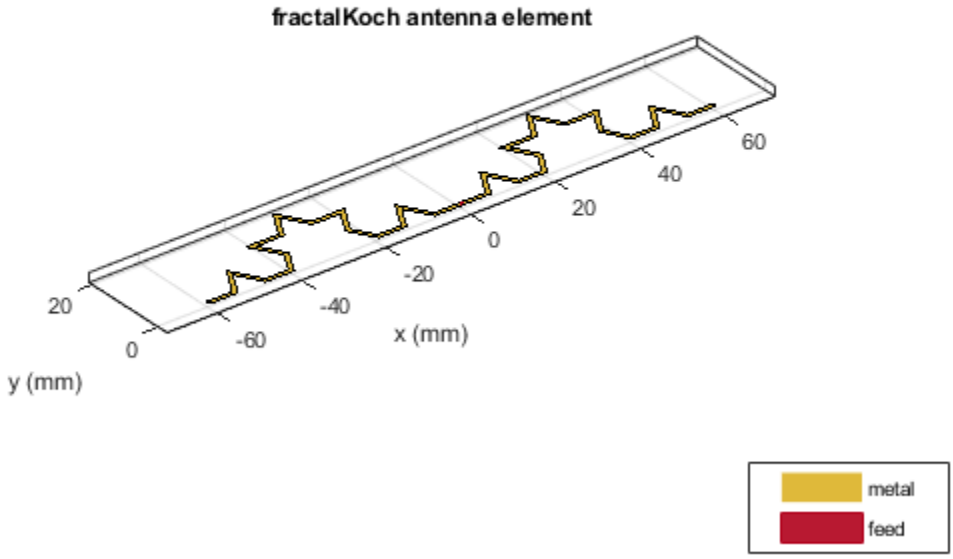
Default Koch Curve Fractal Antenna

Create and view a default Koch curve fractal antenna.

```
ant = fractalKoch
```

```
ant =
  fractalKoch with properties:

    NumIterations: 2
           Length: 0.0600
            Width: 1.0000e-03
             Type: 'dipole'
             Tilt: 0
         TiltAxis: [1 0 0]
             Load: [1x1 lumpedElement]
```

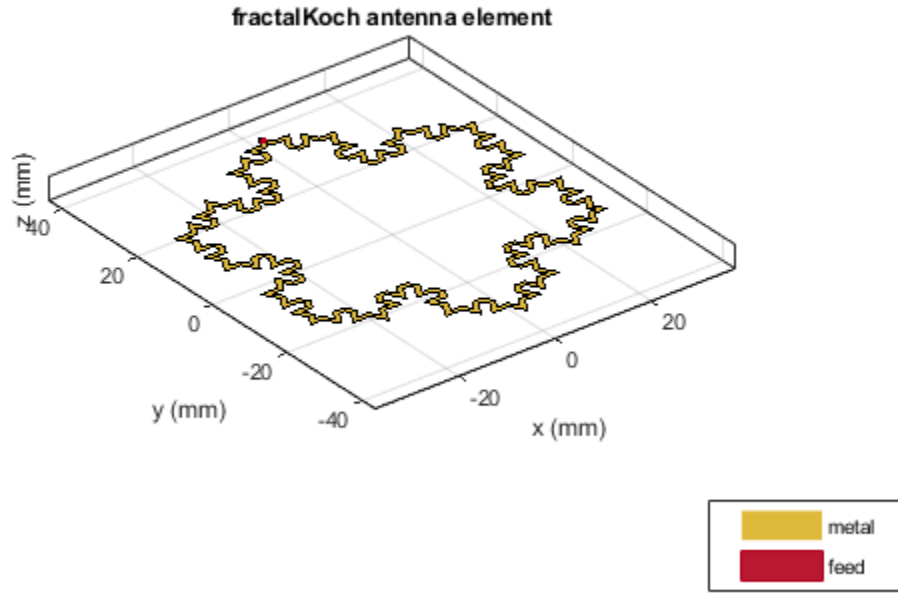
```
show(ant)
```



Koch Loop Fractal Antenna

Create and view a Koch loop fractal antenna with three iterations.

```
ant = fractalKoch('NumIteration',3,'Type','loop');
show(ant)
```



See Also

`fractalGasket`

Topics

"Rotate Antenna and Arrays"

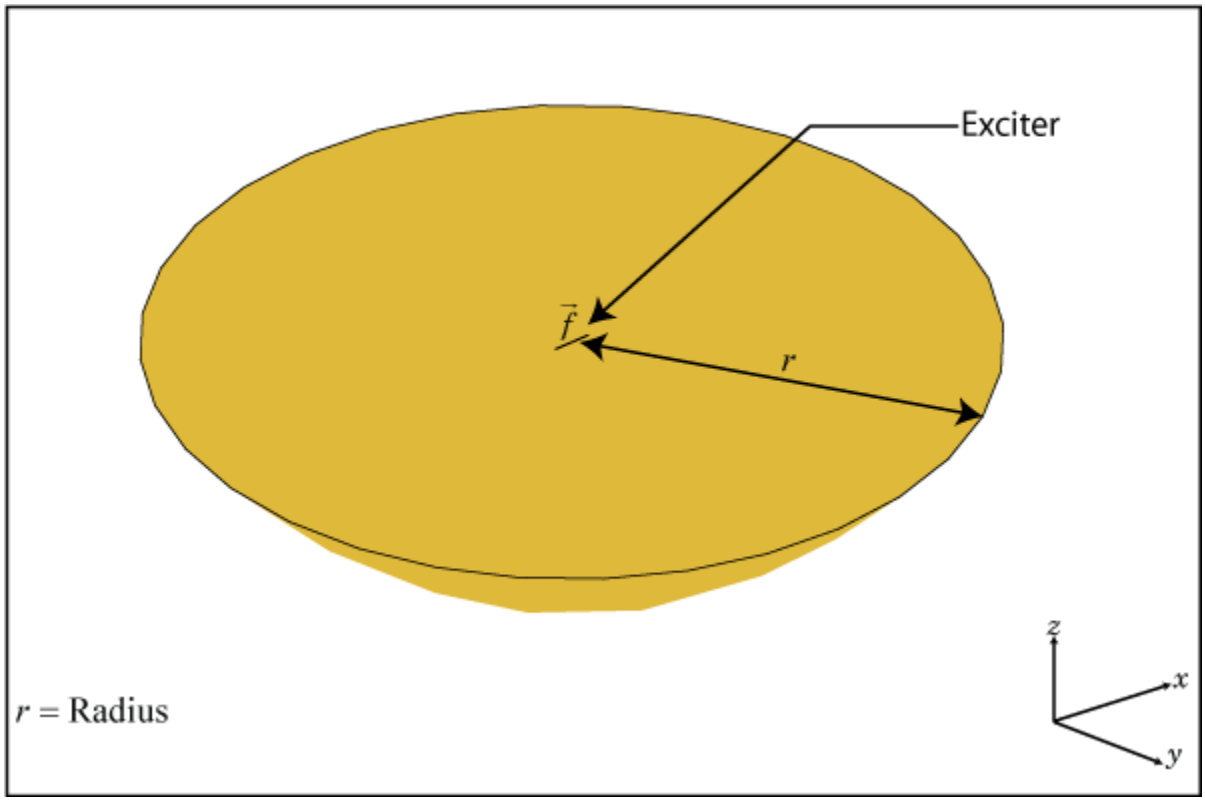
Introduced in R2018b

reflectorParabolic

Create parabolic reflector antenna

Description

The `reflectorParabolic` object creates a parabolic reflector antenna. Parabolic reflector antennas are electrically large structures and are at least 10 wavelengths in diameter. These reflectors are used in TV antennas and satellite communications, for example.



Creation

Syntax

```
ant = reflectorParabolic
ant = reflectorParabolic(Name,Value)
```

Description

`ant = reflectorParabolic` creates a dipole-fed parabolic reflector antenna. The default antenna exciter operates at 10 GHz. The reflector is 10λ in diameter, where λ corresponds to the value of wavelength.

`ant = reflectorParabolic(Name,Value)` sets properties using one or more name-value pairs. For example, `ant = reflectorParabolic('FocalLength',0.5)` creates a parabolic reflector antenna of focal length 0.5 meters.

Output Arguments

`ant` — Parabolic reflector antenna

`reflectorParabolic` object (default)

Parabolic reflector antenna, returned as a `reflectorParabolic` object.

Properties

`Exciter` — Antenna type used as exciter

`dipole` (default) | any single-element antenna object

Antenna type used as an exciter, specified as any single-element antenna object. Except reflector and cavity antenna elements, you can use any of the single elements in the Antenna Toolbox as an exciter.

Example: `'Exciter',horn`

Example: `ant.Exciter = horn`

`Radius` — Radius of parabolic reflector
`0.1500` (default) | positive scalar integer

Radius of the parabolic reflector, specified as a positive scalar integer in meters.

Example: `'Radius',0.22`

Example: `ant.Radius = 0.22`

Data Types: `double`

`FocalLength` — Focal length of parabolic dish
`0.0750` (default) | positive scalar integer

Focal length of the parabolic dish, specified as a positive scalar integer in meters.

Example: `'FocalLength',0.0850`

Example: `ant.FocalLength = 0.0850`

Data Types: `double`

`FeedOffset` — Signed distance from focus
`[0 0 0]` (default) | three-element vector

Signed distance from the focus of the parabolic dish, specified as a three-element vector in meters. By default, the antenna exciter is at the focus of the parabola. Using the `FeedOffset` property, you can place the exciter anywhere on the parabola.

Example: `'FeedOffset',[0.0850 0 0]`

Example: `ant.FeedOffset = [0.0850 0 0]`

Data Types: `double`

`Load` — Lumped elements
[1x1 lumpedElement] (default) | lumped element object handle

Lumped elements added to the antenna feed, specified as a lumped element object handle. For more information, see `lumpedElement`.

Example: `'Load',lumpedelement`. `lumpedelement` is the object handle for the load created using `lumpedElement`.

Example: `ant.Load = lumpedElement('Impedance',75)`

`Tilt` — Tilt angle of antenna
`0` (default) | scalar | vector

Tilt angle of the antenna, specified as a scalar or vector in degrees.

Example: `'Tilt',90`

Example: `ant.Tilt = [90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of antenna
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
axialRatio	Axial ratio of antenna
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array

mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
vswr	Voltage standing wave ratio of antenna

Examples

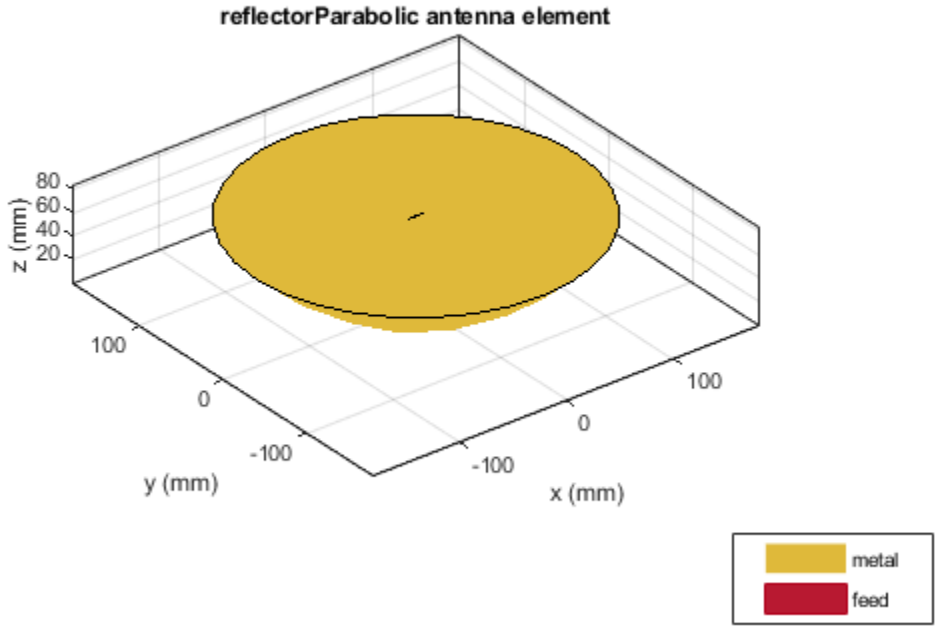
Default Parabolic Reflector and Radiation Pattern

Create and view a default parabolic reflector antenna.

```
ant = reflectorParabolic
```

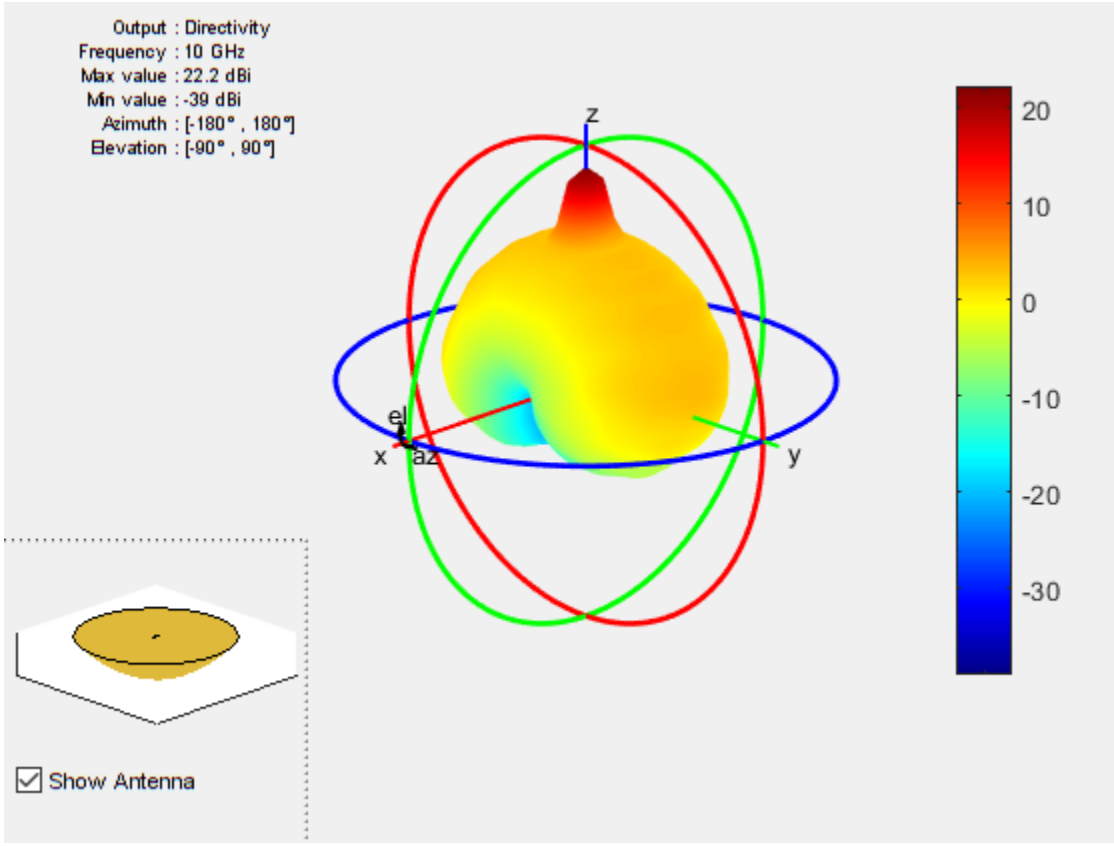
```
ant =
  reflectorParabolic with properties:

        Exciter: [1×1 dipole]
         Radius: 0.1500
    FocalLength: 0.0750
     FeedOffset: [0 0 0]
           Tilt: 0
      TiltAxis: [1 0 0]
           Load: [1×1 lumpedElement]
```

```
show(ant)
```



Plot the radiation pattern of the parabolic reflector at 10 GHz.

```
pattern(ant,10e9)
```



See Also

Topics

“Rotate Antenna and Arrays”

Introduced in R2018b

Antenna Apps — Alphabetical List

Antenna Designer

Design, visualize, and analyze antennas

Description

The **Antenna Designer** app lets you design, visualize, and analyze antennas in the Antenna Toolbox library interactively.

Using this app, you can:

- Select antennas based on general properties or antenna performance.
- Visualize antennas based on frequency and frequency range.
- Analyze antennas based on radiation pattern, polarization, and bandwidth.
- Export selected and designed antennas as a variable to the MATLAB® workspace, as either live script or a function.

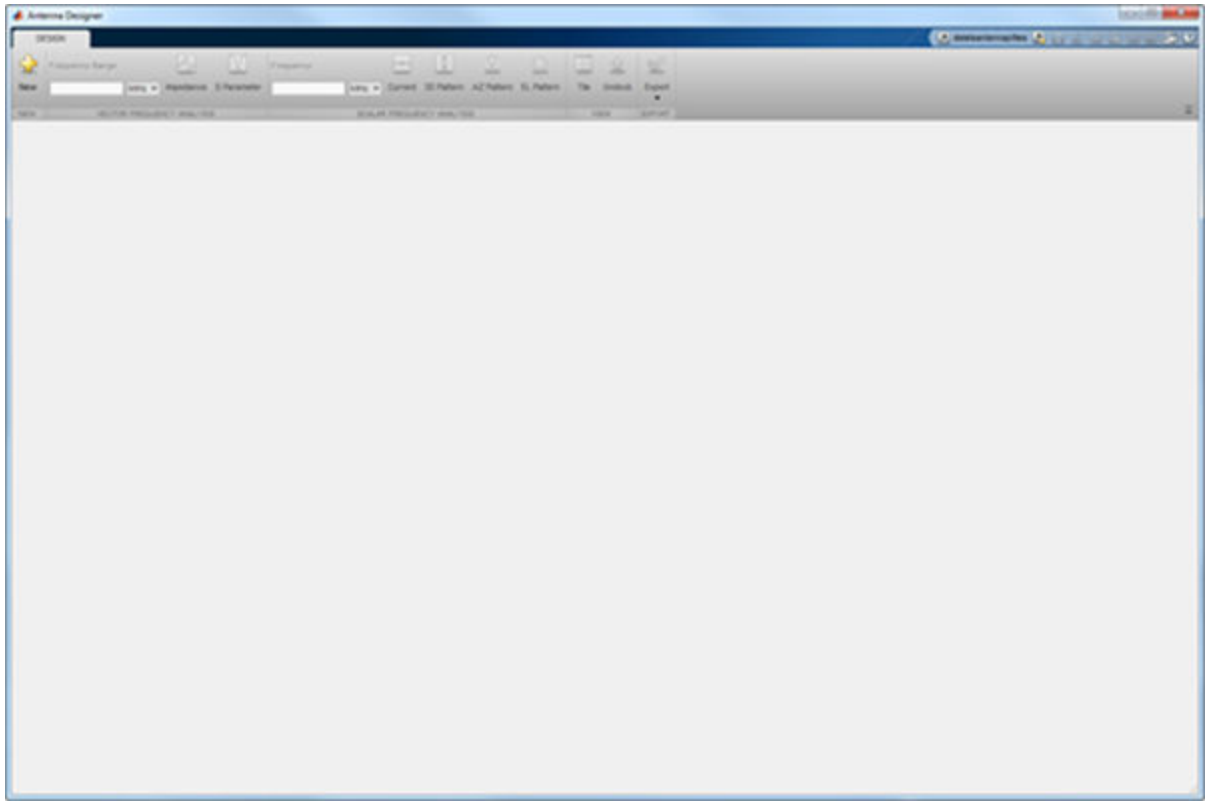
Open the Antenna Designer App

- MATLAB Toolstrip: In the **Apps** tab, under **Signal Processing and Communications**, click the app icon.
- MATLAB command prompt: Enter `antennaDesigner`.

Examples

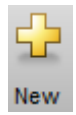
Antenna Designer Canvas

The **Antenna Designer** opens a blank canvas.



1 Select and Visualize Antenna

- Click

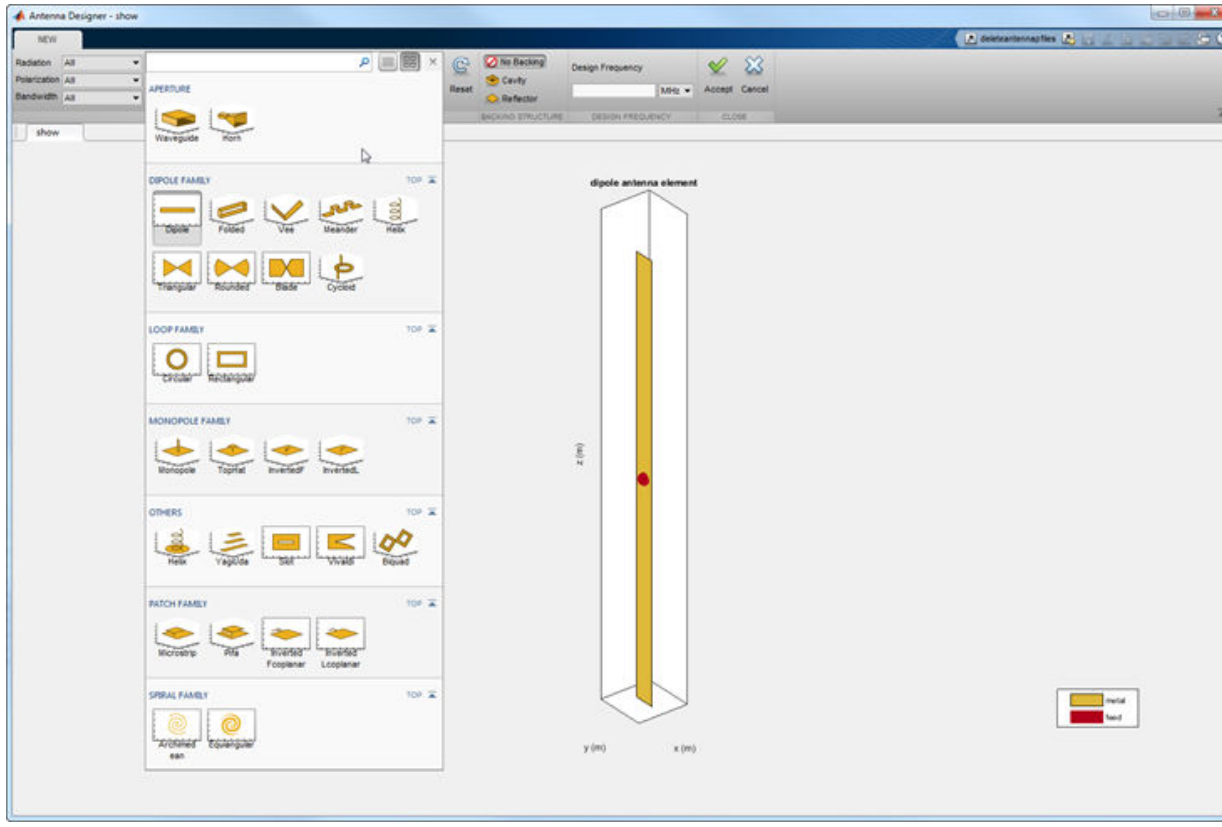
  in the canvas toolstrip to choose the antenna you want to analyze.
- The default antenna is a dipole antenna.

You can filter the antennas based on `Radiation` pattern, `Polarization`, and `Bandwidth`.

- Using the toolstrip you can also add **Cavity** backing, or **Reflector** backing to the antennas.
- You can also specify the **Design Frequency** of the antenna. Setting this value scales the antenna to resonate at the specified frequency. You can also tune the antenna using **Antenna Properties** tab during analysis.
- Use **Reset**, to go back to default settings.
- Use **Accept**, to analyze the antenna characteristics.
- Use **Cancel**, to start over.

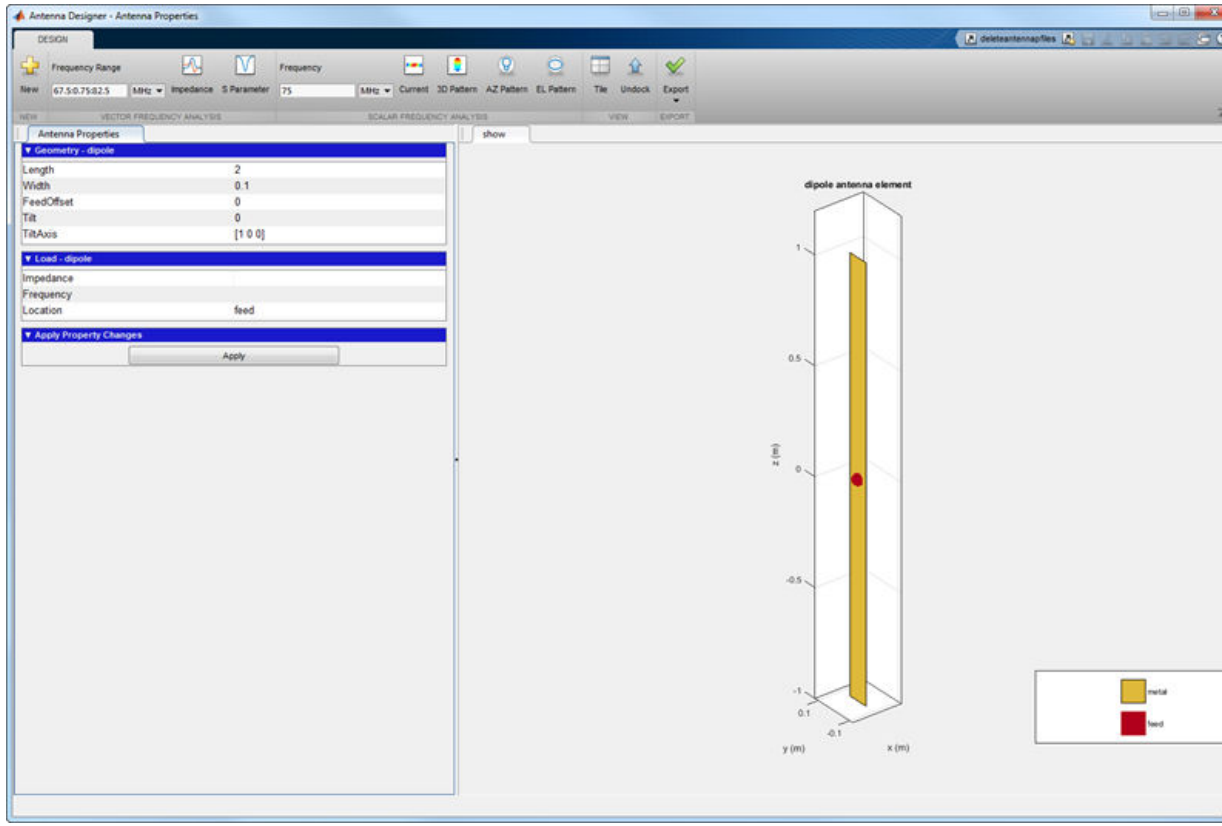
2 Antenna Gallery

- You can choose your antennas from the **Antenna Gallery**.



- The **Antenna Gallery** drop-down menu consists of: `APERTURE`, `DIPOLE FAMILY`, `LOOP FAMILY`, `MONOPOLE FAMILY`, `OTHERS`, `PATCH FAMILY`, and `SPIRAL FAMILY`.
- When you filter antennas based on `Radiation` pattern, `Polarization`, or `Bandwidth`, the antenna gallery greys out the antennas that do not belong to the chosen filter.

3 Analyze Antenna



You can plot the **Impedance** and **S Parameter** of the antenna based on the specified **Frequency Range** in Hz.

- You can visualize the **Current** distribution on the antenna based on the specified **Frequency** in Hz.
- You can visualize the **3D Pattern**, **AZ Pattern**, **EL Pattern** of the antenna based on the specified frequency. Here AZ stands for azimuth and EL stands for elevation.
- Use **Export** to view your antenna in MATLAB workspace or MATLAB script.

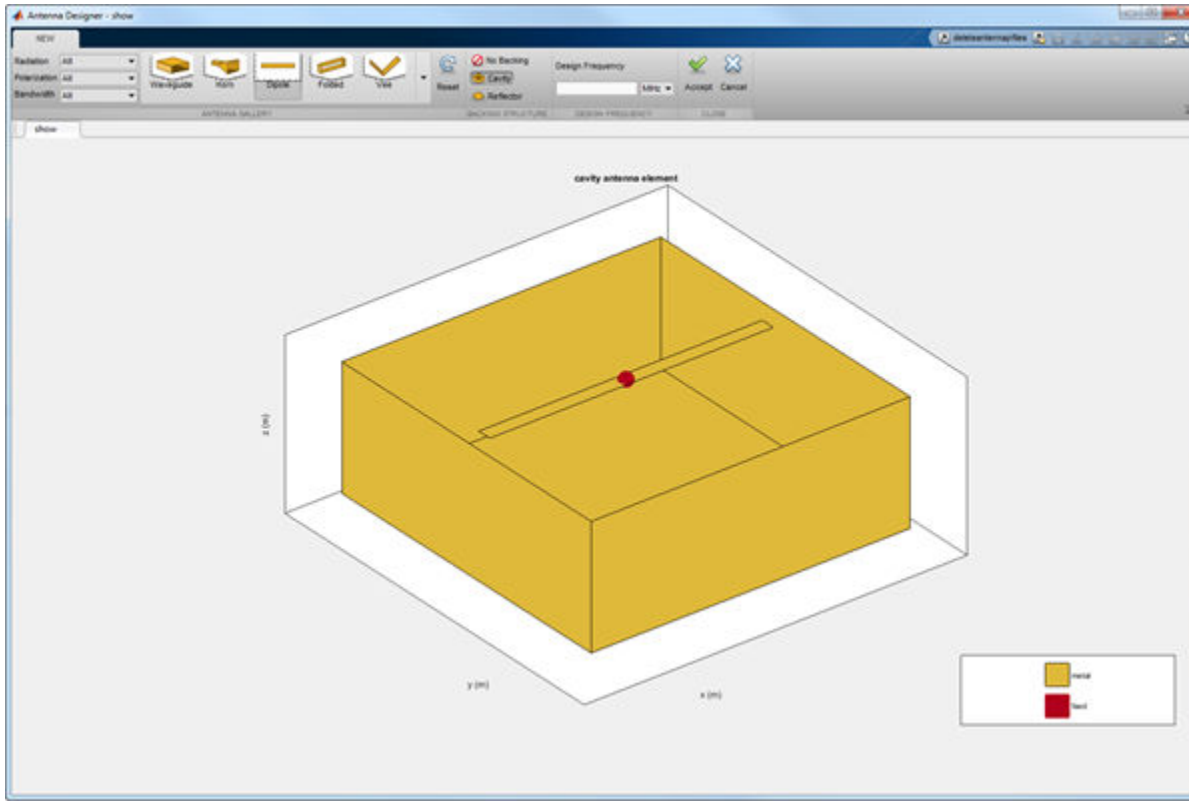
- Manually change the antenna properties using the **Antenna Properties** tab. In this tab, you can change the geometrical properties of the antenna, add a dielectric substrate to the antenna, and change the value and location of the load.

Plot Radiation Pattern of Cavity-Backed Dipole

Use the **Antenna Designer** app to plot the radiation pattern of a cavity-backed dipole antenna.

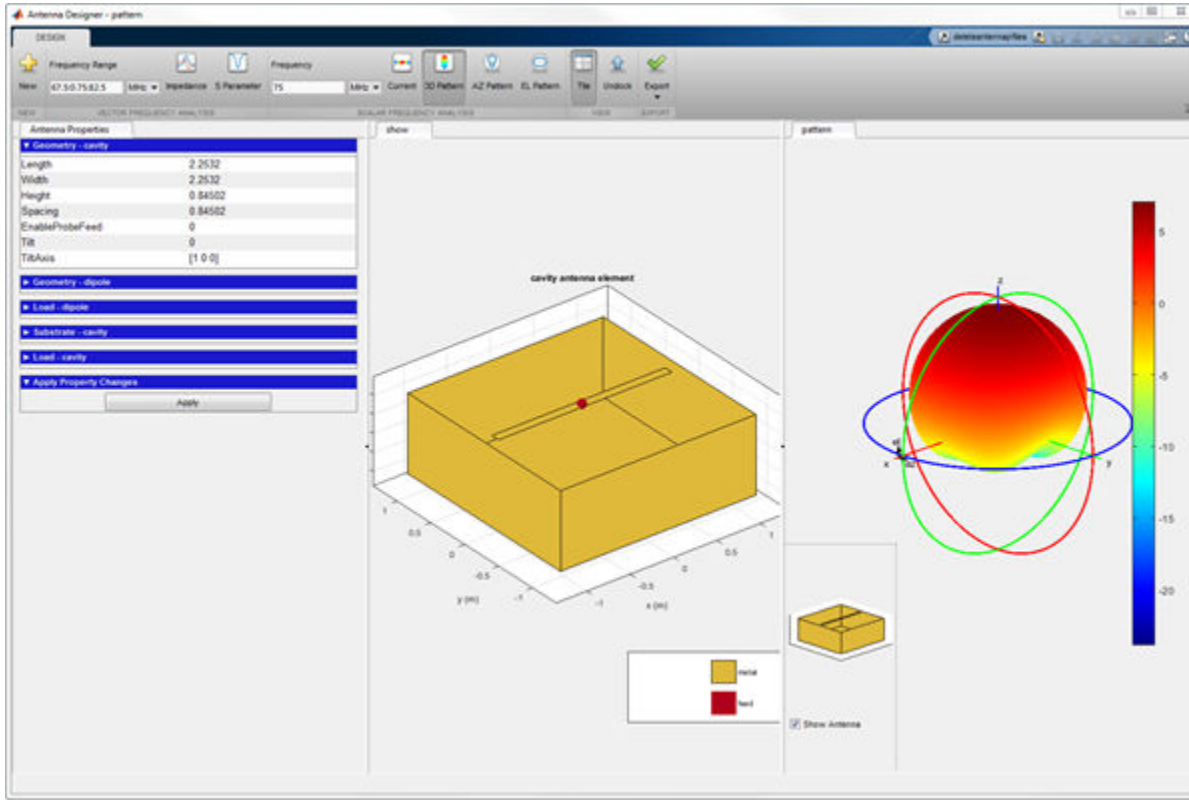
Open the app and click **New** to show the default dipole antenna.

In the **Backing Structure section,** click **Cavity** to create a cavity-backed dipole antenna.



Click **Accept.**

In **SCALAR FREQUENCY ANALYSIS**, click **3D Pattern** to calculate the radiation pattern of the cavity-backed dipole. The default frequency used is 75 MHz. Click **Tile** to view both the antenna and the radiation pattern.

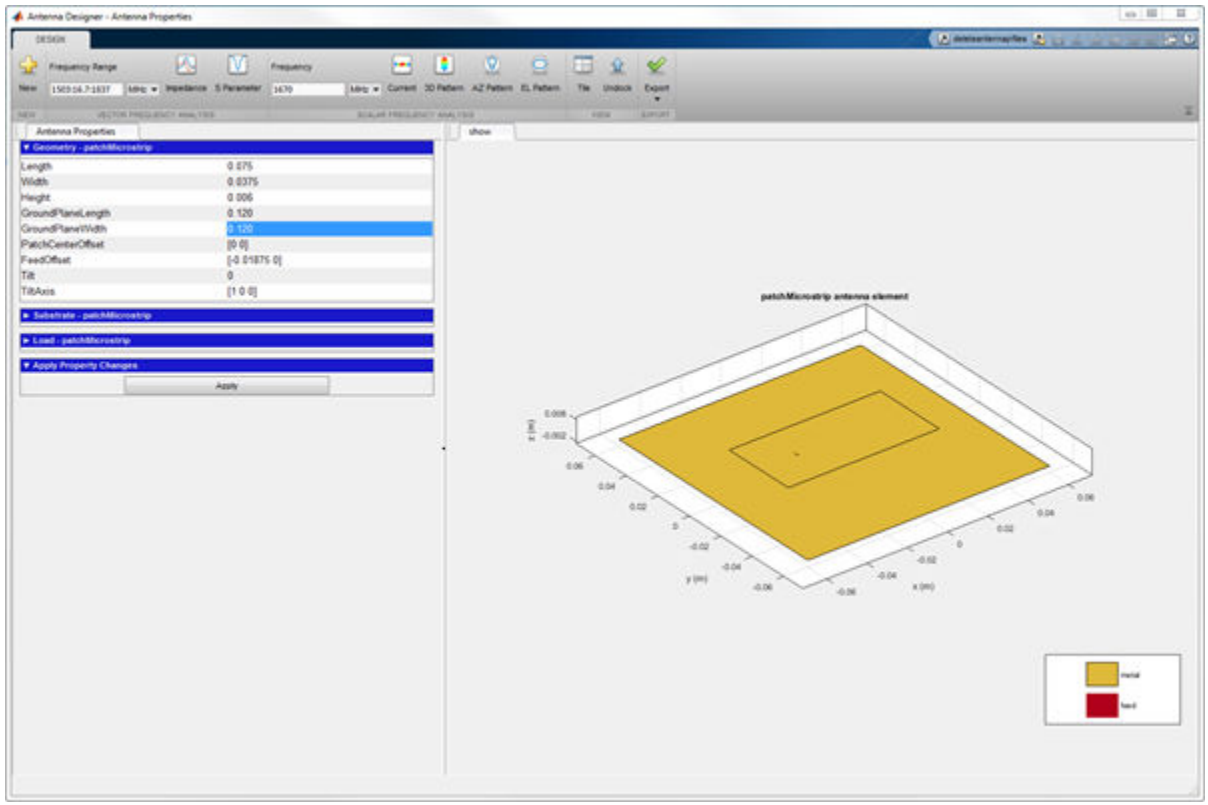


Analyze Patch Microstrip Antenna Having Dielectric Substrate

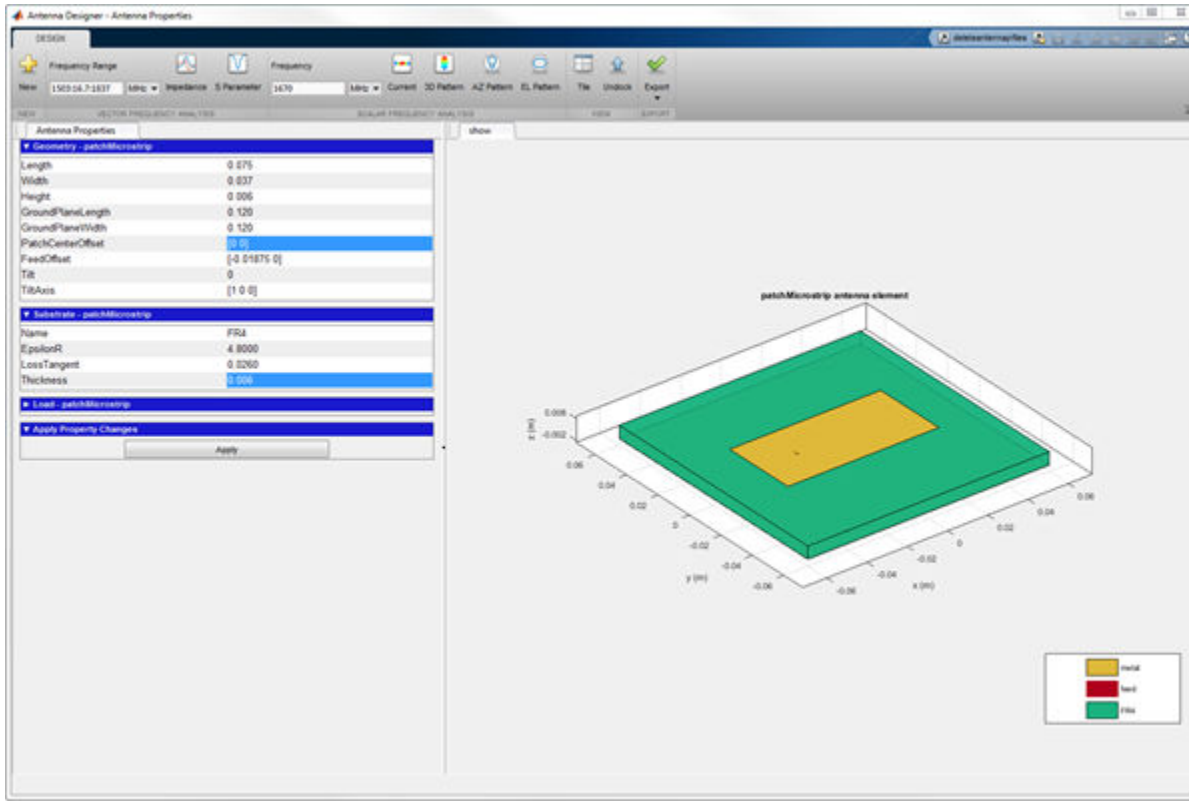
Use the **Antenna Designer** app to plot the radiation pattern of a patch microstrip antenna with a dielectric substrate.

Open the app and click **New**. In the **ANTENNA GALLERY** section, under **PATCH FAMILY**, click `Microstrip` Click **Accept**.

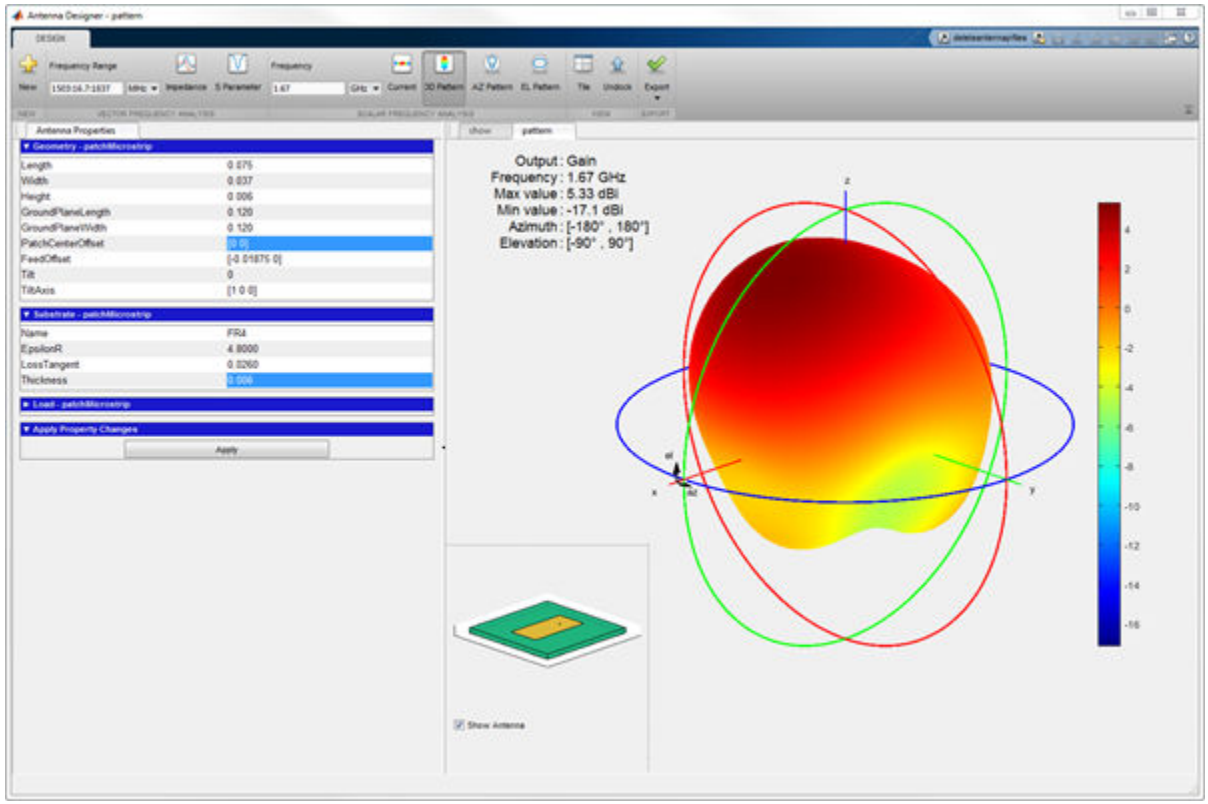
On the **Antenna Properties** tab, change the groundplane length and groundplane width to 0.120 m. Click **Apply** to see the changes.



Add an FR4 dielectric as a substrate to the patch microstrip antenna. To add the dielectric, open the **Substrate** section and set **Name** to FR4, **EpsilonR** to 4.8000, and **Loss Tangent** to 0.0260. Click **Apply** to see the antenna.



Click **3D Pattern** to plot the radiation pattern of the antenna at the default frequency of 1.67 GHz.



- “Antenna Design and Analysis Using Antenna Designer App”

Programmatic Use

`antennaDesigner` opens the **Antenna Designer** app, enabling you to design and analyze antennas present in the Antenna Toolbox library.

See Also

Topics

“Antenna Design and Analysis Using Antenna Designer App”

Introduced in R2017a

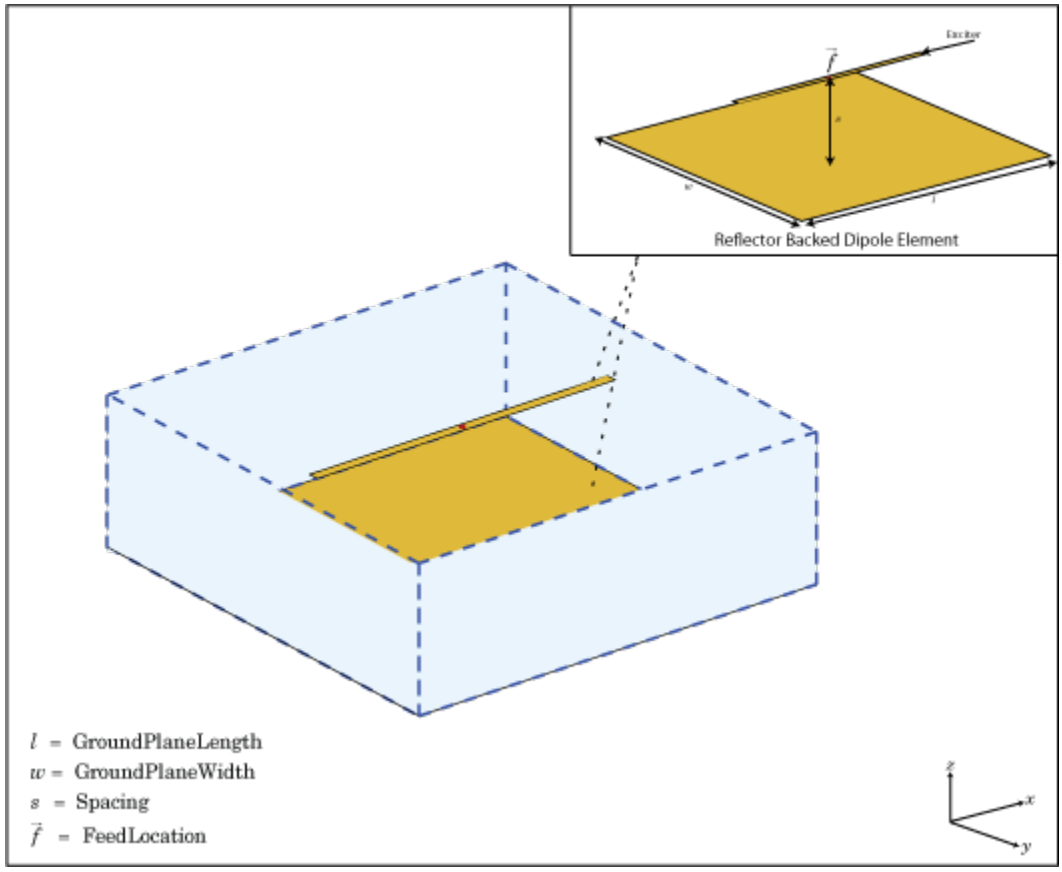
Array Objects— Alphabetical List

infiniteArray

Create 2-D custom mesh antenna on X-Y plane

Description

The `infiniteArray` object is an infinite antenna array in the X-Y plane. Infinite array models a single antenna element called the *unit cell*. Ground plane of the antennas specifies the boundaries of the unit cell. Antennas without a ground plane require a reflector. By default, the infinite array has reflected-backed dipoles as antenna elements. The default dimensions are chosen for an operating frequency of 1 GHz.



Creation

Description

`infa = infiniteArray` creates an infinite antenna array in the X-Y plane.

`infa = infiniteArray(Name,Value)` creates an infinite antenna array with additional properties specified by one, or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value

pair arguments in any order as `Name1, Value1, ..., NameN, ValueN`. Properties not specified retain default values.

Properties

`Element` — Type of individual antenna elements in unit cell

reflector-backed dipole (default) | object

Type of individual antenna elements in unit cell, specified as an object. Antenna without a groundplane is backed using a reflector. The ground plane size specifies the unit cell boundaries.

Example: `'Element',reflector`

`ScanAzimuth` — Scan direction in azimuth plane

0 (default) | scalar

Scan direction in azimuth plane, specified as a scalar in degrees.

Example: `'ScanAzimuth',25`

Data Types: `double`

`ScanElevation` — Scan direction in elevation plane

0 (default) | scalar

Scan direction in elevation plane, specified as a scalar in degrees.

Example: `'ScanElevation',80`

Data Types: `double`

Object Functions

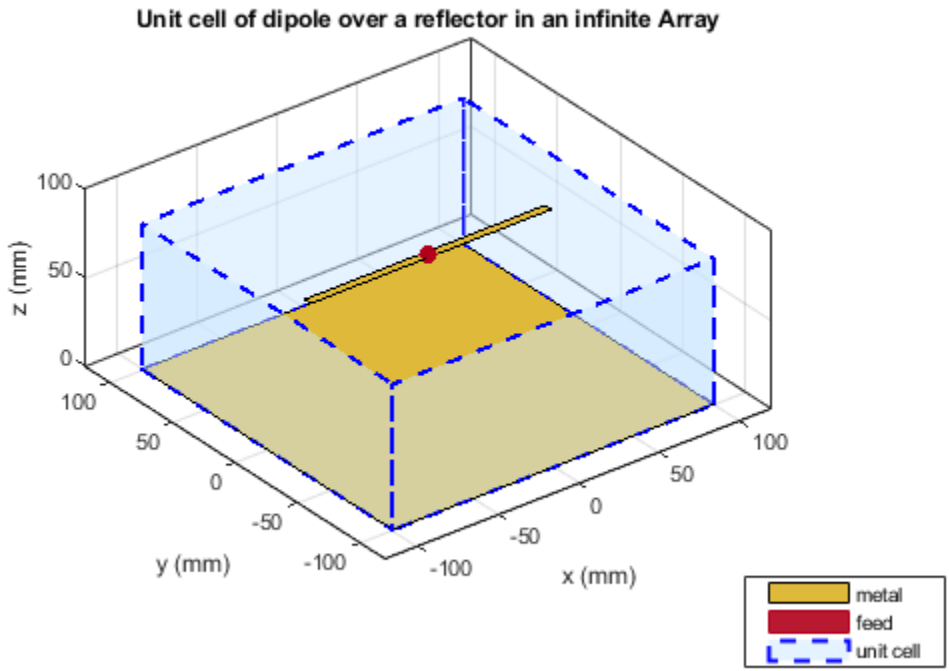
numSummationTerms	Change number of summation terms for calculating periodic Green's function

Examples

Infinite Array of Reflector-Backed Dipoles

Create an infinite array with reflector-backed dipoles as unit cells. Scan the array at boresight. Visualize the unit cell.

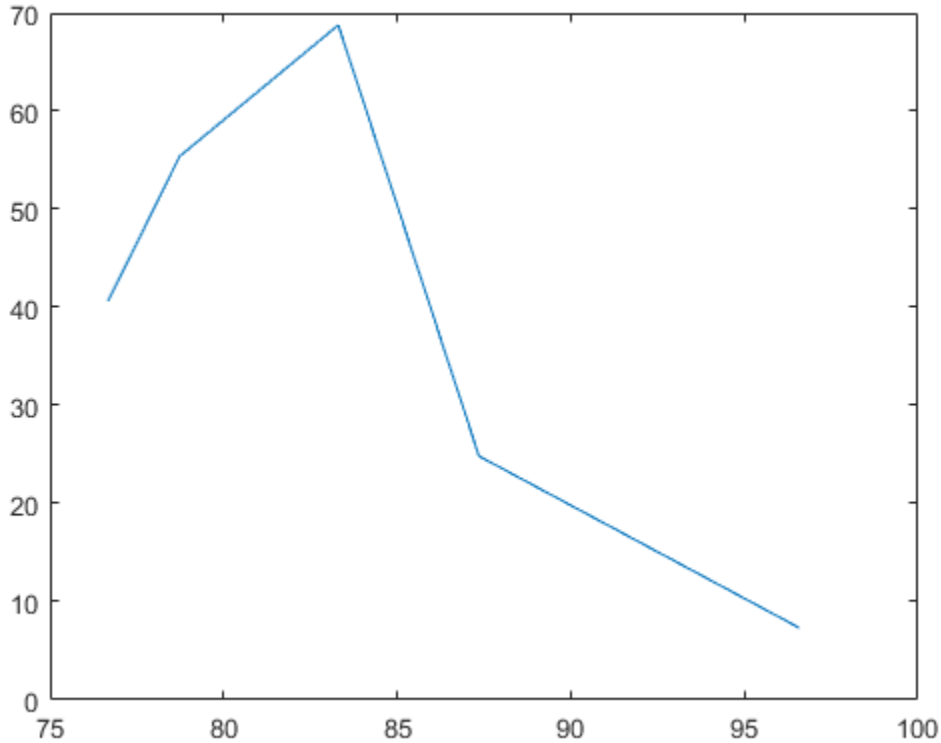
```
infa = infiniteArray('Element',reflector,'ScanAzimuth',0, ...
    'ScanElevation',90);
show(infa)
```



Scan Impedance of Infinite Array

Calculate the scan impedance of an infinite array at 1GHz. To calculate the impedance, scan the infinite array from boresight to horizon in the elevation plane.

```
infa = infiniteArray;
theta0deg = linspace(0,90,5);
zscan = nan(1,numel(theta0deg));
    for j = 1:numel(theta0deg)
      infa.ScanElevation = theta0deg(j);
      zscan(1,j) = impedance(infa,1e9);
    end
 plot(zscan)
```



References

[1] Balanis, C.A. *Antenna Theory: Analysis and Design*. 3rd Ed. New York: Wiley, 2005.

See Also

`circularArray` | `conformalArray` | `linearArray` | `rectangularArray`

Topics

“Rotate Antenna and Arrays”

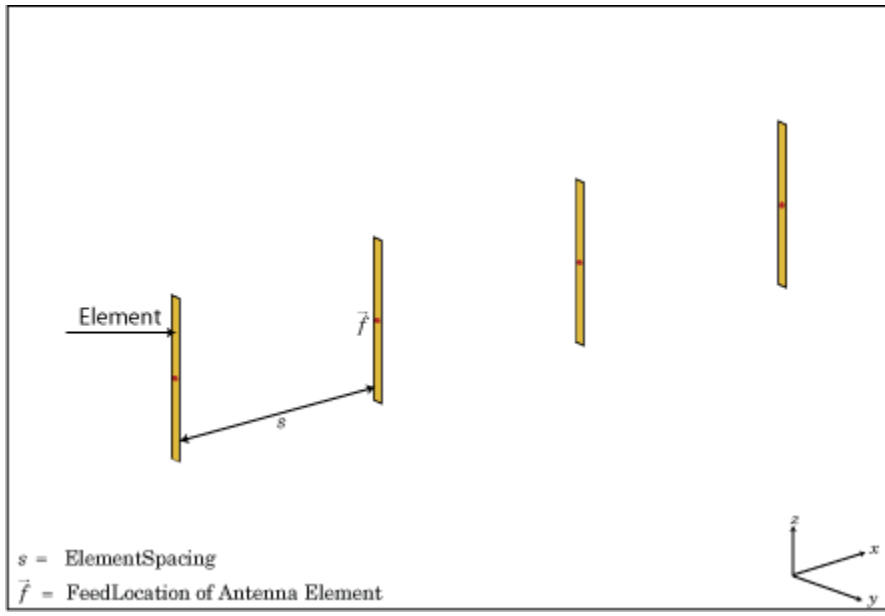
Introduced in R2015b

linearArray

Create linear antenna array

Description

The `linearArray` class creates a linear antenna array in the X-Y plane. By default, the linear array is a two-element dipole array. The dipoles are center fed. Each dipole resonates at 70 MHz when isolated.



Creation

Syntax

```
la = linearArray
```

```
la = linearArray(Name,Value)
```

Description

`la = linearArray` creates a linear antenna array in the X-Y plane.

`la = linearArray(Name,Value)` class to create a linear antenna array, with additional properties specified by one, or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`,`...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Element` — Individual antenna elements used in array
dipole (default) | antenna object

Individual antenna elements used in array, specified as an antenna object.

Example: `'Element',monopole`

`NumElements` — Number of antenna elements in array
2 (default) | scalar

Number of antenna elements in array, specified as a scalar.

Example: `'NumElements',4`

`'ElementSpacing'` — Spacing between antenna elements
2 (default) | scalar | vector

Spacing between antenna elements, specified as a scalar or vector in meters. By default, the dipole elements are spaced `2 m` apart.

Example: `'ElementSpacing',3`

Data Types: `double`

`AmplitudeTaper` — Excitation amplitude of antenna elements
1 (default) | scalar | vector

Excitation amplitude of antenna elements, specified as a scalar or vector. Set the property value to `0` to model dead elements. This value corresponds to the excitation voltages for the elements in the array.

Example: `'AmplitudeTaper',3`

Data Types: `double`

`Phaseshift` — Phase shift for antenna elements
0 (default) | scalar | vector

Phase shift for antenna elements, specified as a scalar or vector in degrees. This value corresponds to the excitation voltages for the elements in the array.

Example: `'PhaseShift',[3 3 0 0]`

Data Types: `double`

`Tilt` — Tilt angle of array
0 (default) | scalar | vector

Tilt angle of array, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of array
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

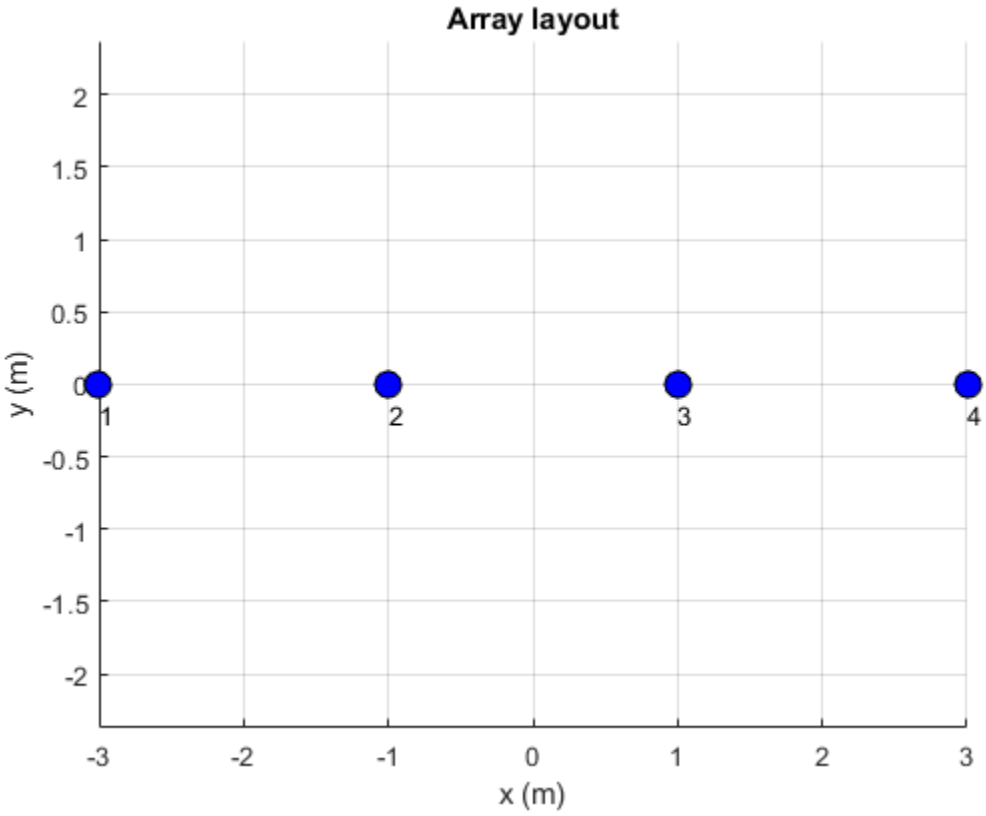
show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
correlation	Correlation coefficient between two antennas in array
current	Current distribution on metal or dielectric antenna or array surface
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object

Examples

Create and Plot Layout of Linear Array

Create a linear array of four dipoles and plot the layout of the array.

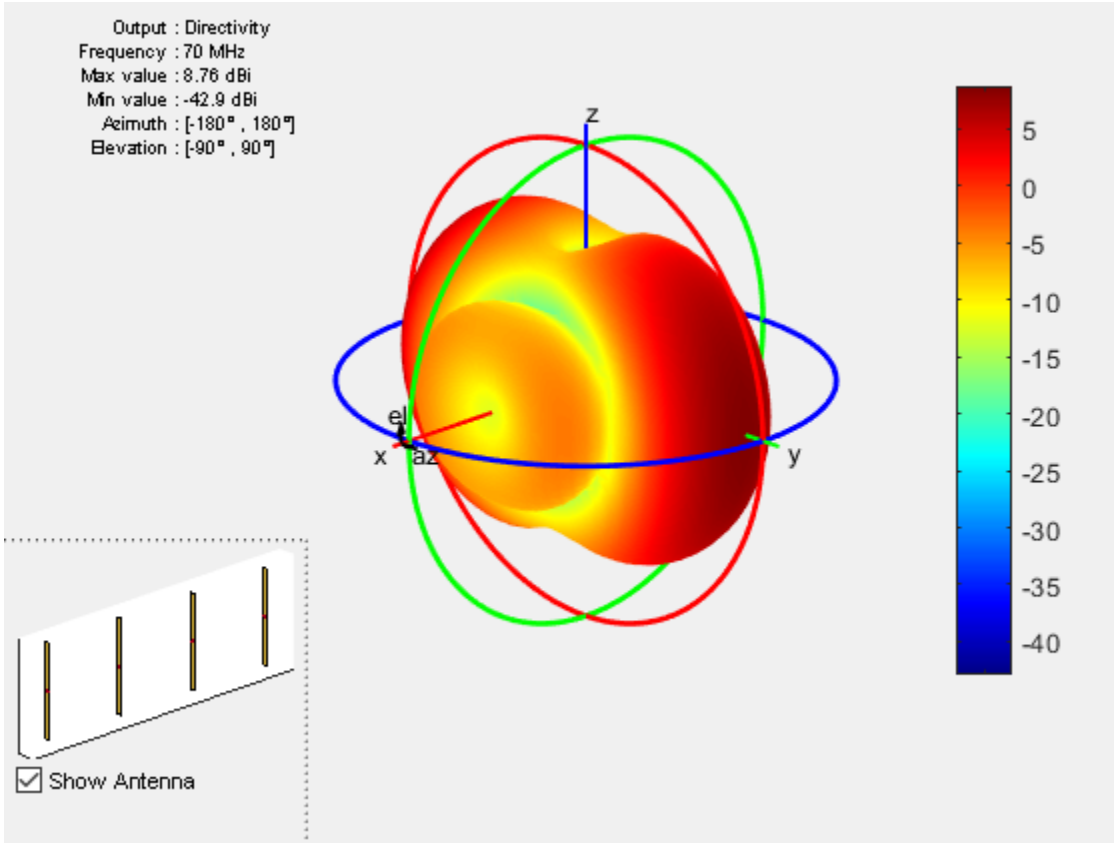
```
la = linearArray;
la.NumElements = 4;
layout(la);
```



Radiation Pattern of Linear Array

Plot the radiation pattern of a four element linear array of dipoles at a frequency 70MHz.

```
la = linearArray('NumElements',4);
pattern(la,70e6);
```



Linear Array Using Groundplane Antennas

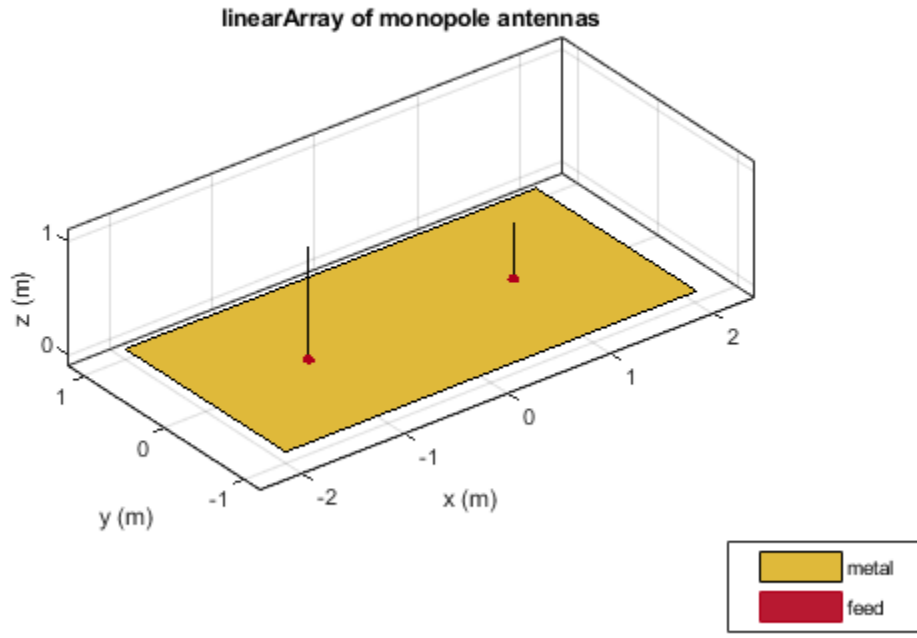
Create a linear array of two monopoles.

```
m1 = monopole;
m2 = monopole('Height',0.5);
mla = linearArray
```

```
mla =
  linearArray with properties:

           Element: [1x1 dipole]
```

```
       NumElements: 2
    ElementSpacing: 2
    AmplitudeTaper: 1
        PhaseShift: 0
              Tilt: 0
          TiltAxis: [1 0 0]
```

```
mla.Element = [m1,m2];
show(mla);
```



References

[1] Balanis, C.A. *Antenna Theory. Analysis and Design*, 3rd Ed. New York: Wiley, 2005.

See Also

`circularArray` | `conformalArray` | `infiniteArray` | `rectangularArray`

Topics

“Rotate Antenna and Arrays”

Introduced in R2015a

conformalArray

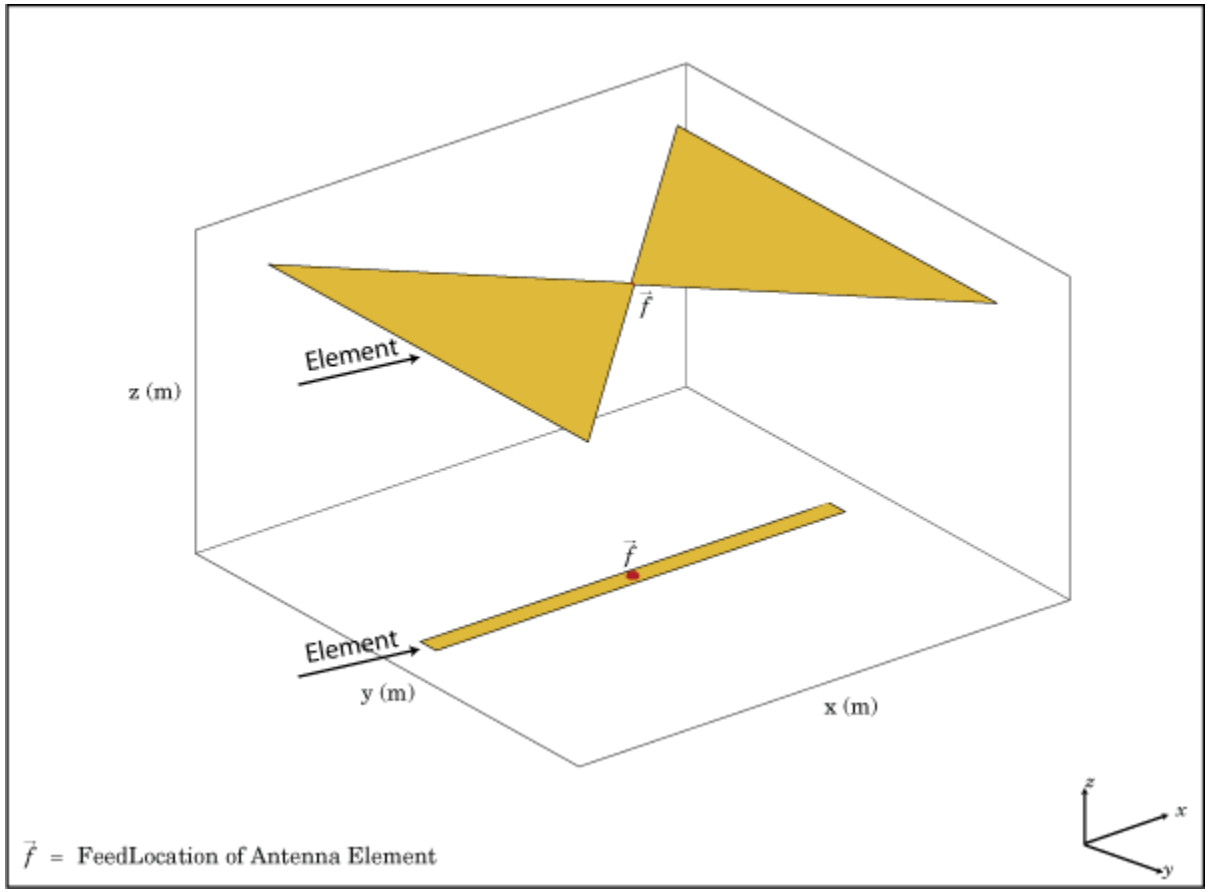
Create conformal antenna array

Description

The `conformalArray` class creates an antenna array using any element from the antenna library. You can also specify an array of any arbitrary geometry, such as a circular array, a nonplanar array, or an array with nonuniform geometry.

Conformal arrays are used in:

- Direction-finding systems that uses circular arrays or stacked circular arrays
- Aircraft systems due to surface irregularities or mechanical stress



Creation

Syntax

```
ca = conformalArray
ca = conformalArray(Name,Value)
```

Description

`ca = conformalArray` creates a conformal antenna array using the default antenna element, shape, and antenna positions.

`ca = conformalArray(Name,Value)` creates a conformal antenna array with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain default values.

Properties

`ElementPosition` — Position of feed or origin

`[0 0 0; 0 0 0.1500]` (default) | *M*-by-3 real matrix

Position of the feed or origin for each antenna element, specified as *M*-by-3 real matrix. *M* is the number of element positions. By default, *M* is 2. To specify additional antenna elements, add additional element positions in the conformal array.

Example: `'ElementPosition',[0.1 0.1 0.1; -0.1 -0.1 -0.1;0.2 0.0.2]`

Data Types: `double`

`Element` — Individual antenna elements in array

scalar | array of objects | cell array of objects

Individual antenna elements in the array, specified as one of the following values:

- A scalar
- An array of objects
- Cell array of objects

By default, conformal array has two antenna elements, the dipole and the bowtie. To specify additional antenna elements, add additional element positions in the conformal array. You can add both balanced and unbalanced antennas to the same conformal array.

Example: `m = monopole; h = conformalArray('Element', [m,m])`. Creates a conformal array consisting of two monopoles antenna elements.

Example: `d = dipole`; `mt = monopoleTopHat`; `h = conformalArray('Element',{d,mt})`. Creates a conformal array consisting of a dipole antenna and a monopole tophat antenna.

Data Types: `cell`

`Reference` — Position reference for antenna element
`'feed'` (default) | `'origin'`

Position reference for the antenna element, specified as either `'origin'` or `'feed'`. For more information, see “Position Reference” on page 4-35

Example: `'Reference','origin'`

Data Types: `char` | `string`

`AmplitudeTaper` — Excitation amplitude of antenna elements
1 (default) | scalar | nonnegative vector

Excitation amplitude of the antenna elements, specified as a scalar or a nonnegative vector. To model dead elements, set the property value to `0`.

Example: `'AmplitudeTaper',3`

Example: `'AmplitudeTaper',[3 0]`. Creates a two-element conformal array, where `3` and `0` are the excitations amplitudes of two elements.

Data Types: `double`

`PhaseShift` — Phase shift for antenna elements
0 (default) | scalar | real vector

Phase shift for antenna elements, specified as a scalar or a real vector in degrees.

Example: `'PhaseShift',[-45 -45 45 45]`

Data Types: `double`

`Tilt` — Tilt angle of array
0 (default) | scalar | vector

Tilt angle of array, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of array
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
correlation	Correlation coefficient between two antennas in array
current	Current distribution on metal or dielectric antenna or array surface
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array

returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object

Examples

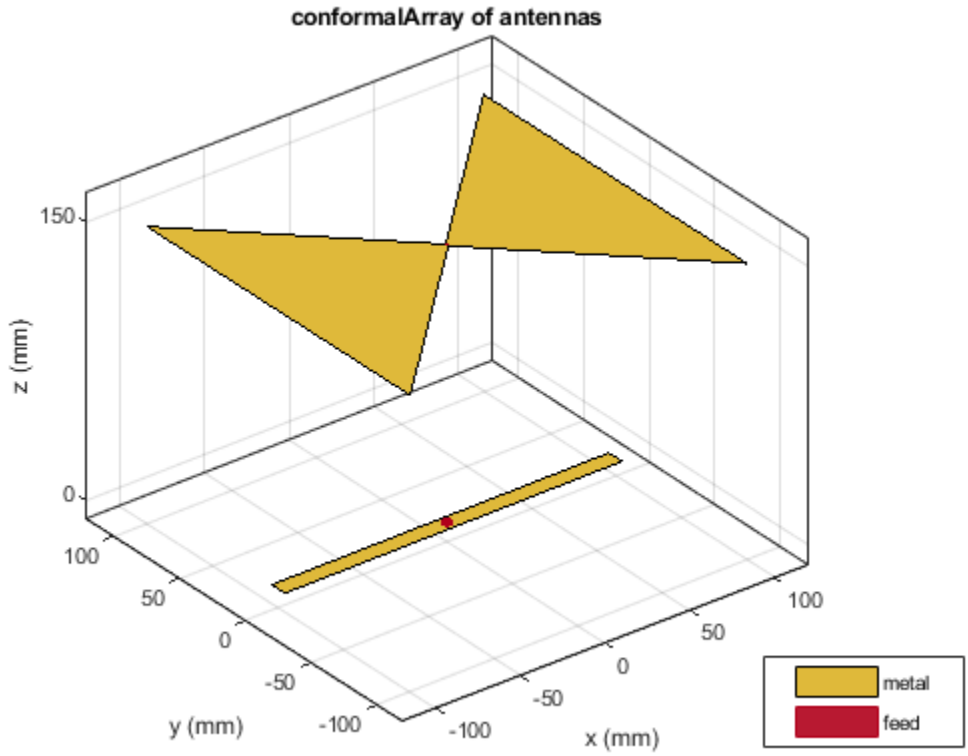
Default Conformal Array

Create a default conformal array.

```
c = conformalArray
```

```
c =
  conformalArray with properties:

            Element: {[1x1 dipole]  [1x1 bowtieTriangular]}
    ElementPosition: [2x3 double]
          Reference: 'feed'
     AmplitudeTaper: 1
         PhaseShift: 0
               Tilt: 0
           TiltAxis: [1 0 0]
```

```
show(c)
```



Circular Array of Dipoles

Define the radius and the number of elements for the array.

```
r = 2;
N = 12;
```

Create an array of 12 dipoles.

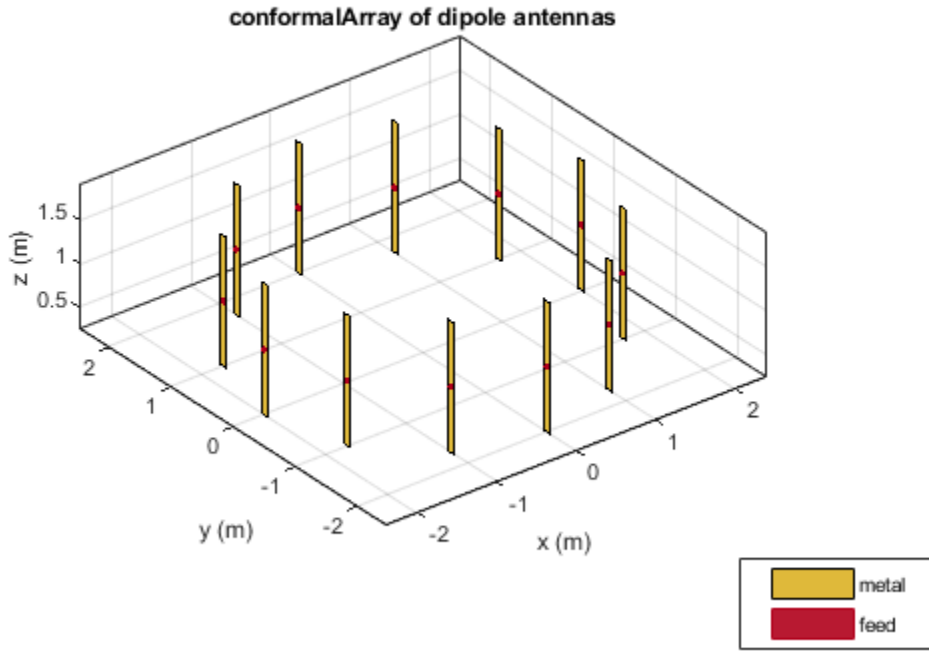
```
elem = repmat(dipole('Length',1.5),1,N);
```

Define the x,y,z values for the element positions in the array.

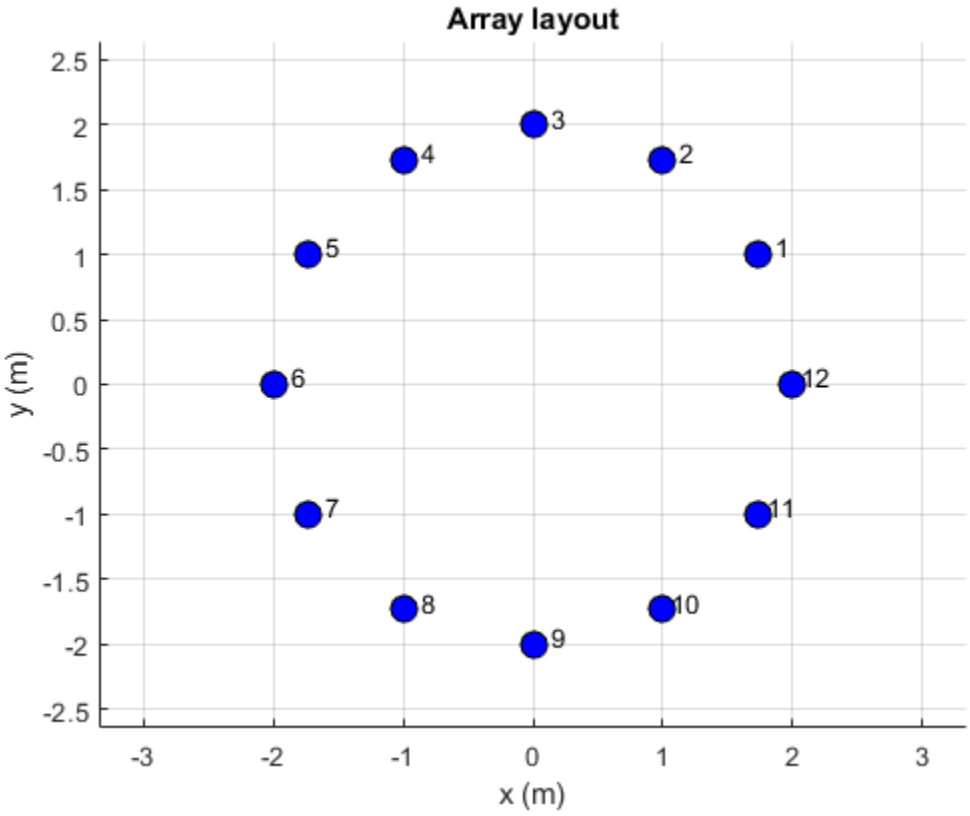
```
del_th = 360/N;
th = del_th:del_th:360;
x = r.*cosd(th);
y = r.*sind(th);
z = ones(1,N);
pos = [x;y;z];
```

Create a circular array using the defined dipoles and then visualize it. Display the layout of the array.

```
c = conformalArray('Element',elem,'ElementPosition',pos');
show(c)
```

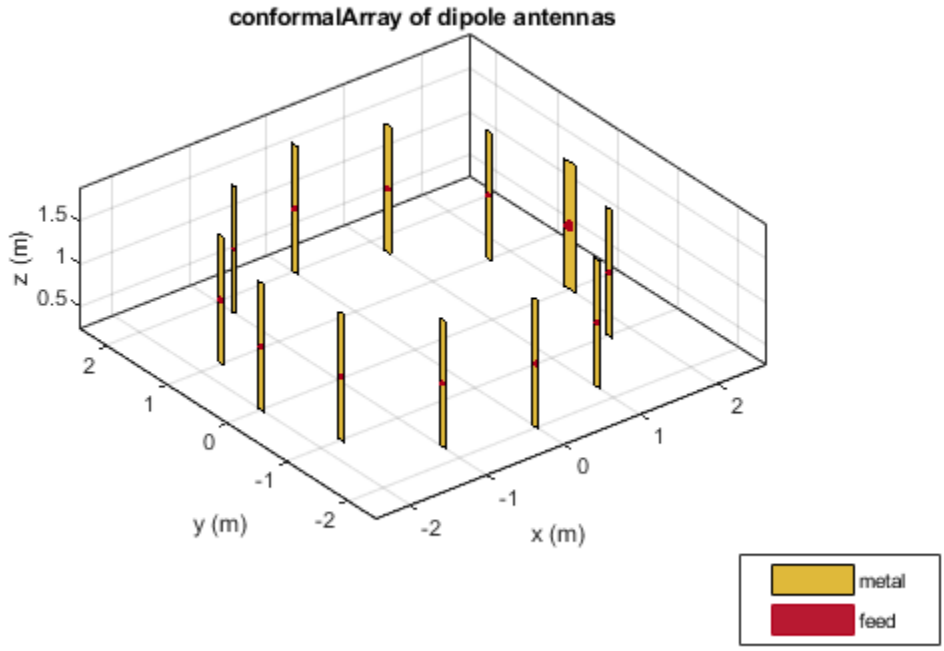


```
figure
layout(c)
```



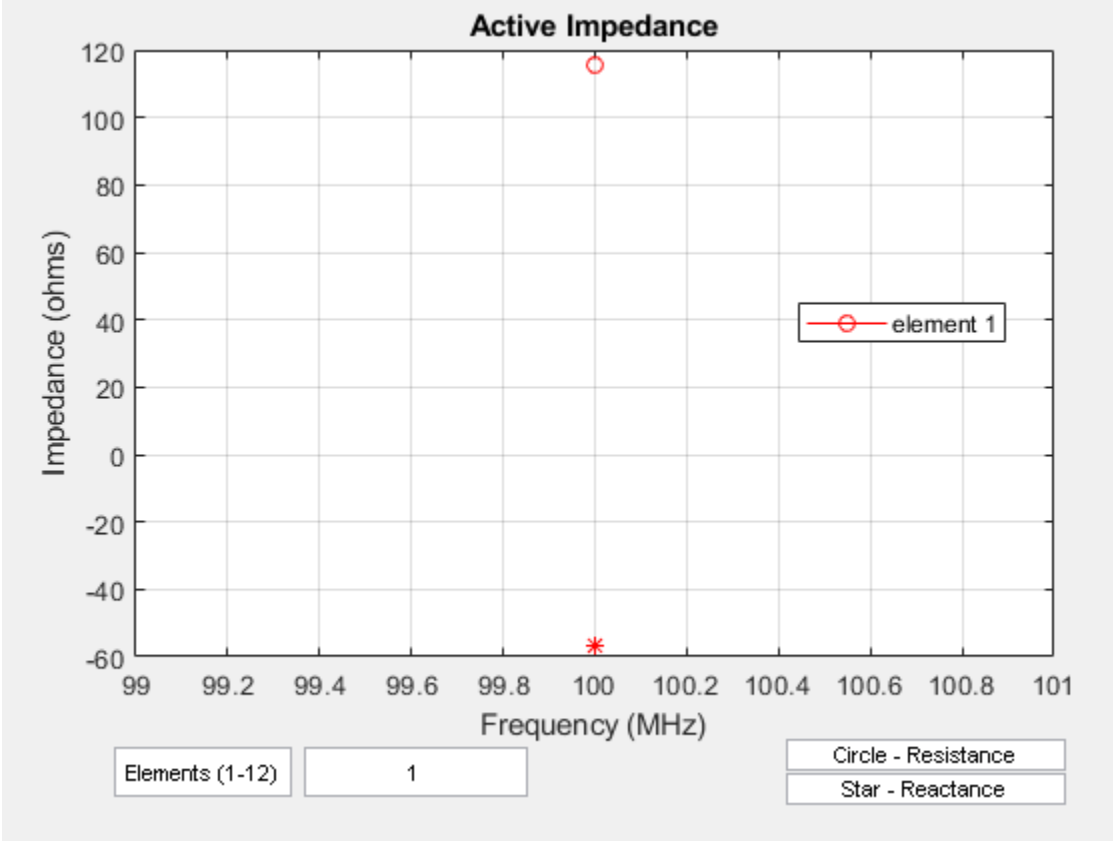
Change the width of the fourth and the twelfth element of the circular array. Visualize the new arrangement.

```
c.Element(4).Width = 0.05;
c.Element(12).Width = 0.2;
figure
show(c)
```

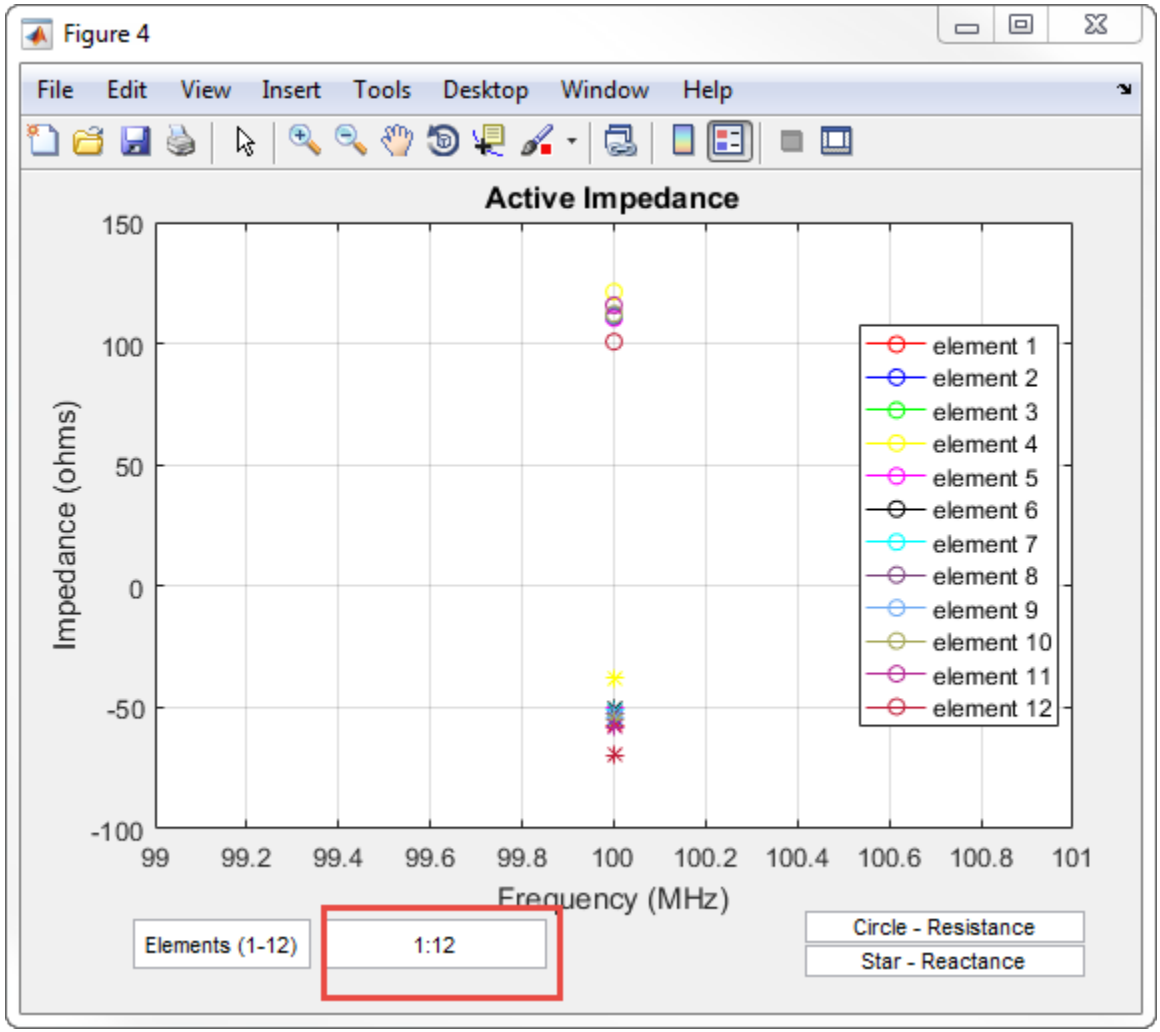


Calculate and plot the impedance of the circular array at 100 MHz. The plot shows the impedance of the first element in the array.

```
figure
impedance(c,100e6)
```



To view the impedance of all the elements in the array change the value from **1** to **1:12** as shown in the figure.



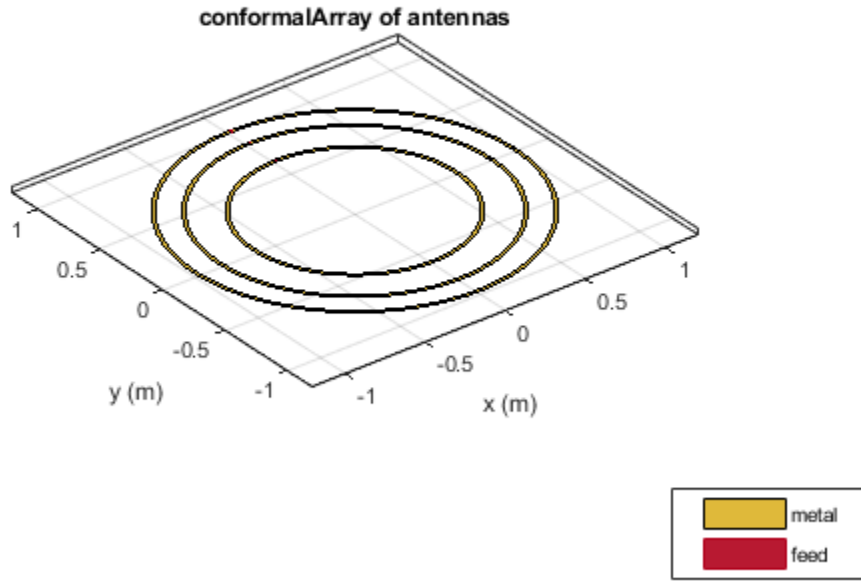
Radiation Pattern of Concentric Array of Circular Loop Antennas

Define three circular loop antennas of radii 0.6366 m(default), 0.85 m, and 1 m, respectively.

```
l1 = loopCircular;
l2 = loopCircular('Radius',0.85);
l3 = loopCircular('Radius',1);
```

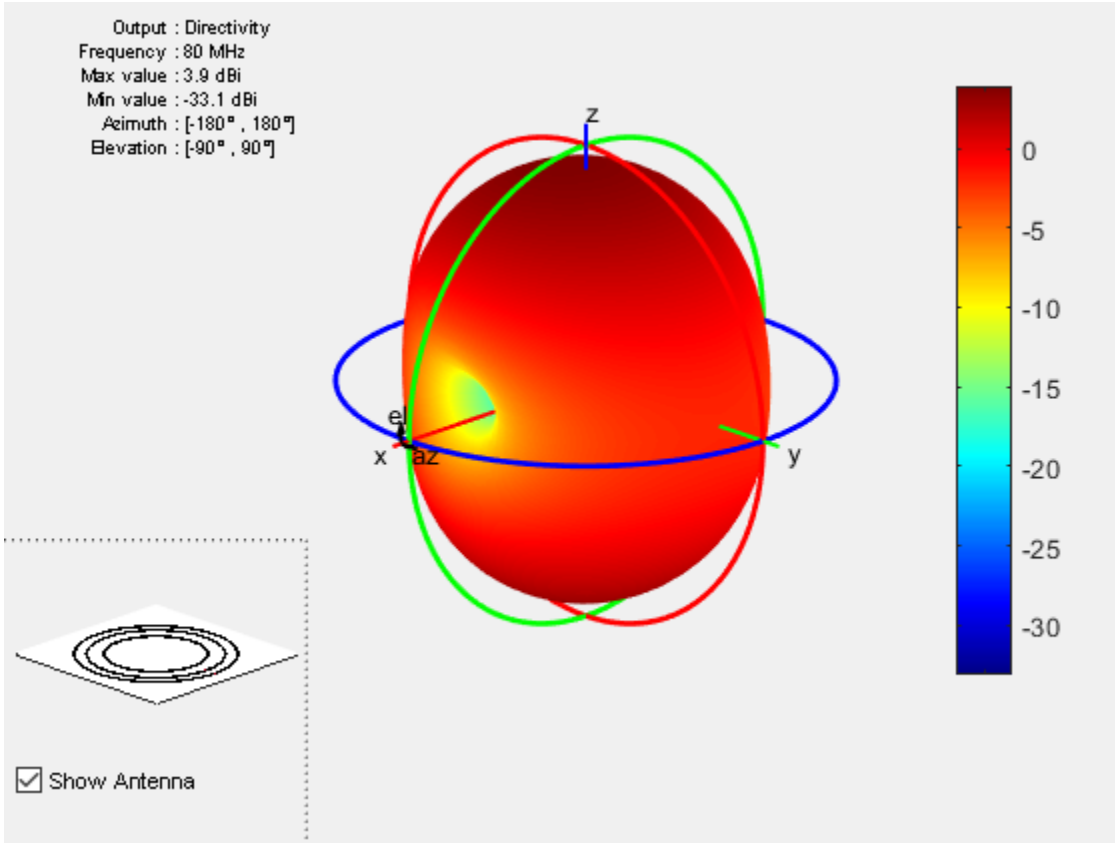
Create a concentric array that uses the origin of circular loop antennas as its position reference.

```
c = conformalArray('Element',{l1,l2,l3},'ElementPosition',[0 0 0;0 0 0;...
    0 0 0],'Reference','origin');
show(c)
```



Visualize the radiation pattern of the array at 80 MHz.

```
pattern(c,80e6)
```



Conformal Array Using Infinite Ground Plane Antenna

Create a dipole antenna to use in the reflector and the conformal array.

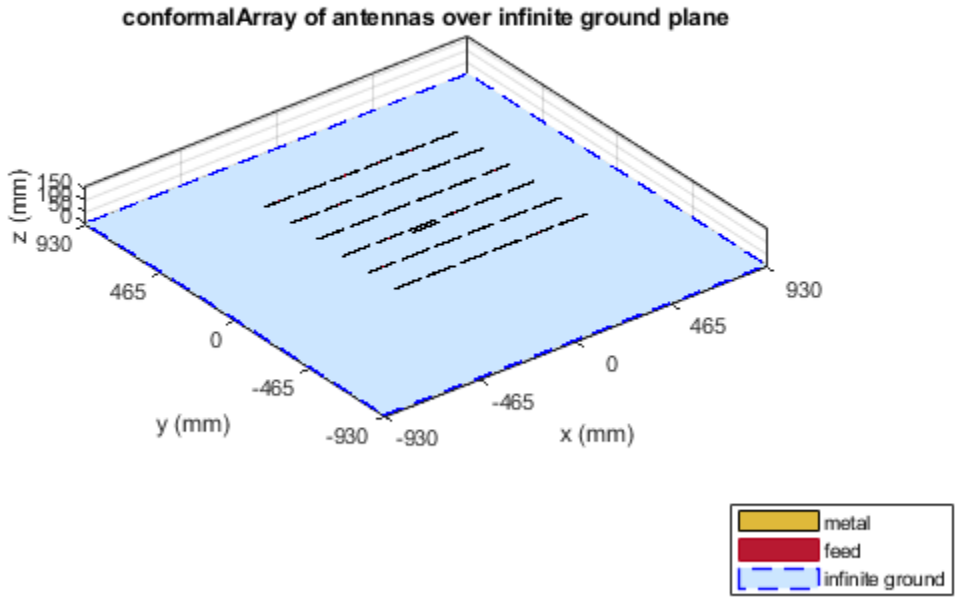
```
d = dipole('Length',0.13,'Width',5e-3,'Tilt',90,'TiltAxis','Y');
```

Create an infinite groundplane reflector antenna using the dipole as exciter.

```
rf = reflector('Exciter',d,'Spacing',0.15/2,'GroundPlaneLength',inf);
```

Create a conformal array using 36 dipole antennas and one infinite groundplane reflector antenna. View the array.

```
x = linspace(-0.4,0.4,6);
y = linspace(-0.4,0.4,6);
[X,Y] = meshgrid(x,y);
pos = [X(:) Y(:) 0.15*ones(numel(X),1)];
for i = 1:36
    element{i} = d;
end
element{37} = rf;
lwa = conformalArray('Element',element,'ElementPosition',[pos;0 0 0.15/2]);
show(lwa)
```

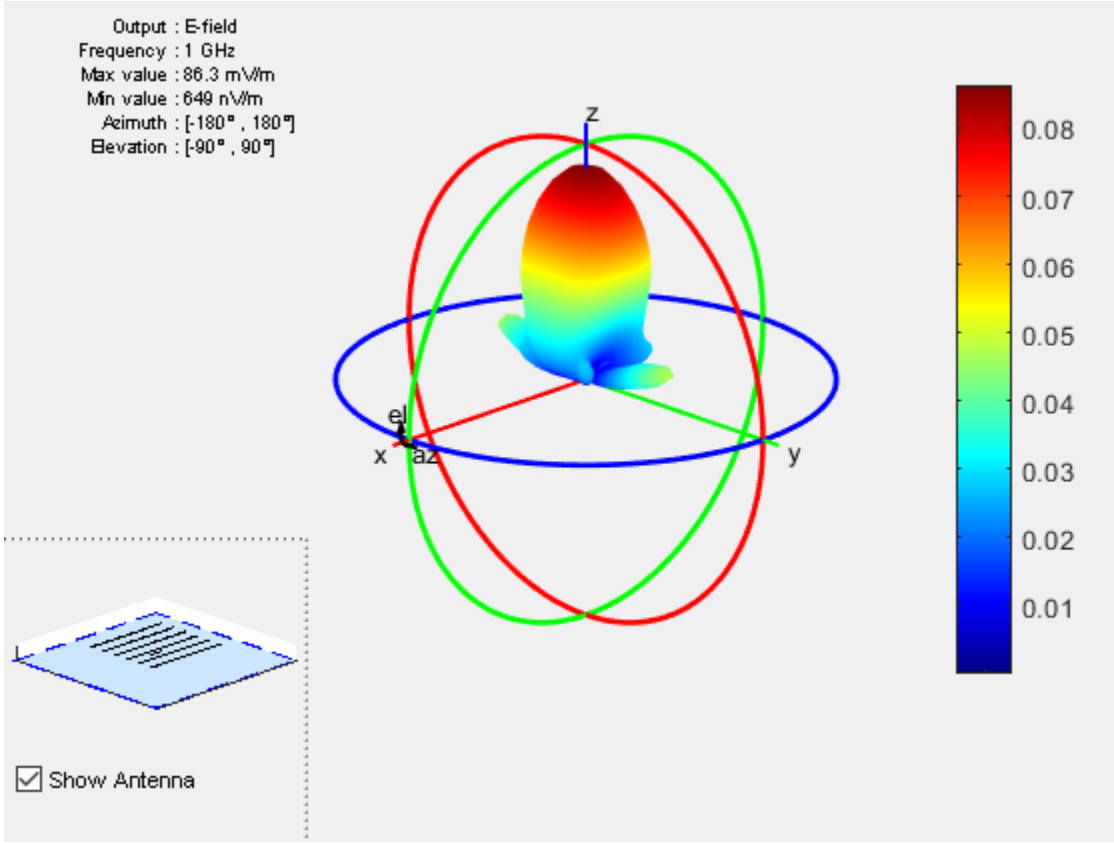


Drive only the reflector antenna with an amplitude of 1.

```
V = zeros(1,37);
V(end) = 1;
lwa.AmplitudeTaper = V;
```

Compute the radiation pattern of the conformal array.

```
figure
pattern(lwa,1e9,'Type','efield')
```



Conformal Array Using Dielectric Antennas

Create two patch microstrip antennas using dielectric substrate FR4. Tilt the second patch microstrip antenna by 180 degrees.

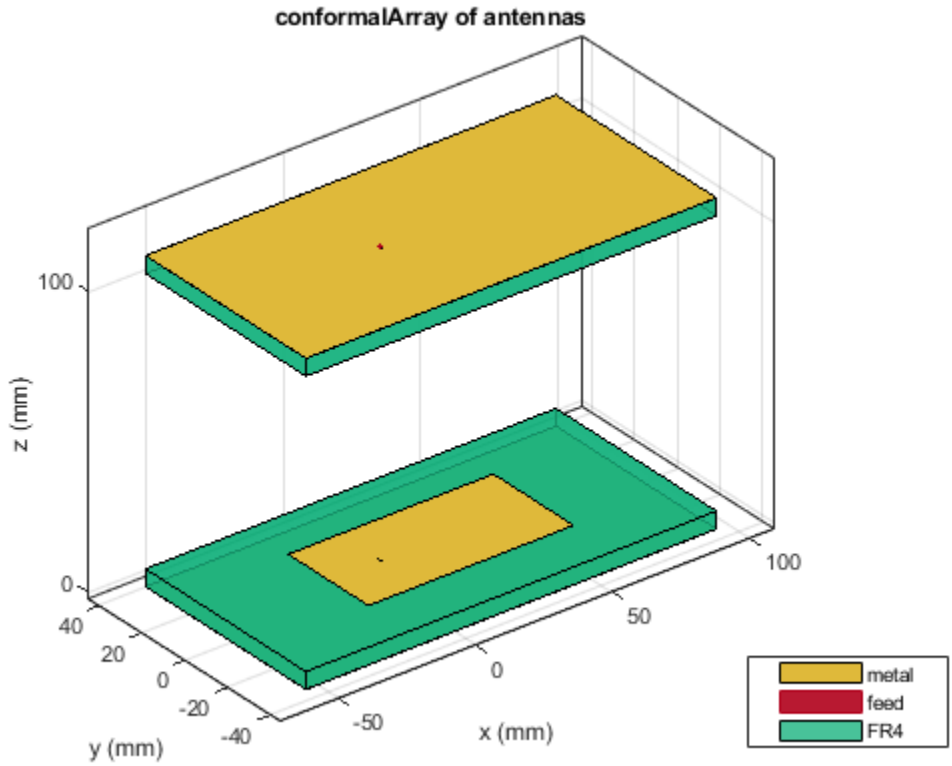
```
d = dielectric('FR4');
p1 = patchMicrostrip('Substrate',d);
p2 = patchMicrostrip('Substrate',d,'Tilt',180);
```

Create and view a conformal array using the two patch microstrip antennas placed 11 cm apart.

```
c = conformalArray('ElementPosition',[0 0 0;0 0 0.1100],'Element',{p1,p2})
```

```
c =
  conformalArray with properties:

            Element: {[1x1 patchMicrostrip]  [1x1 patchMicrostrip]}
    ElementPosition: [2x3 double]
          Reference: 'feed'
     AmplitudeTaper: 1
         PhaseShift: 0
               Tilt: 0
           TiltAxis: [1 0 0]
```

```
show(c)
```



Conformal Array Using Balanced and Unbalanced Antennas

Create a conformal array using dipole and monopole antennas.

```
c = conformalArray('Element', {dipole, monopole})
```

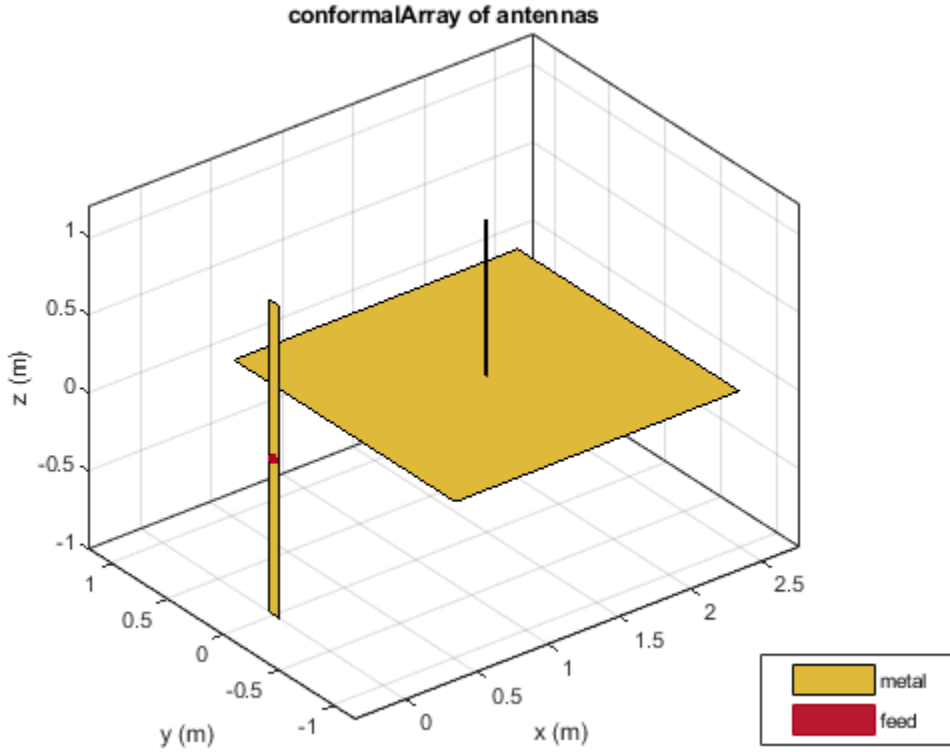
```
c =
  conformalArray with properties:

            Element: {[1x1 dipole]  [1x1 monopole]}
    ElementPosition: [2x3 double]
          Reference: 'feed'
```

```
    AmplitudeTaper: 1
        PhaseShift: 0
              Tilt: 0
          TiltAxis: [1 0 0]
```

```
c.ElementPosition = [0 0 0; 1.5 0 0];
```

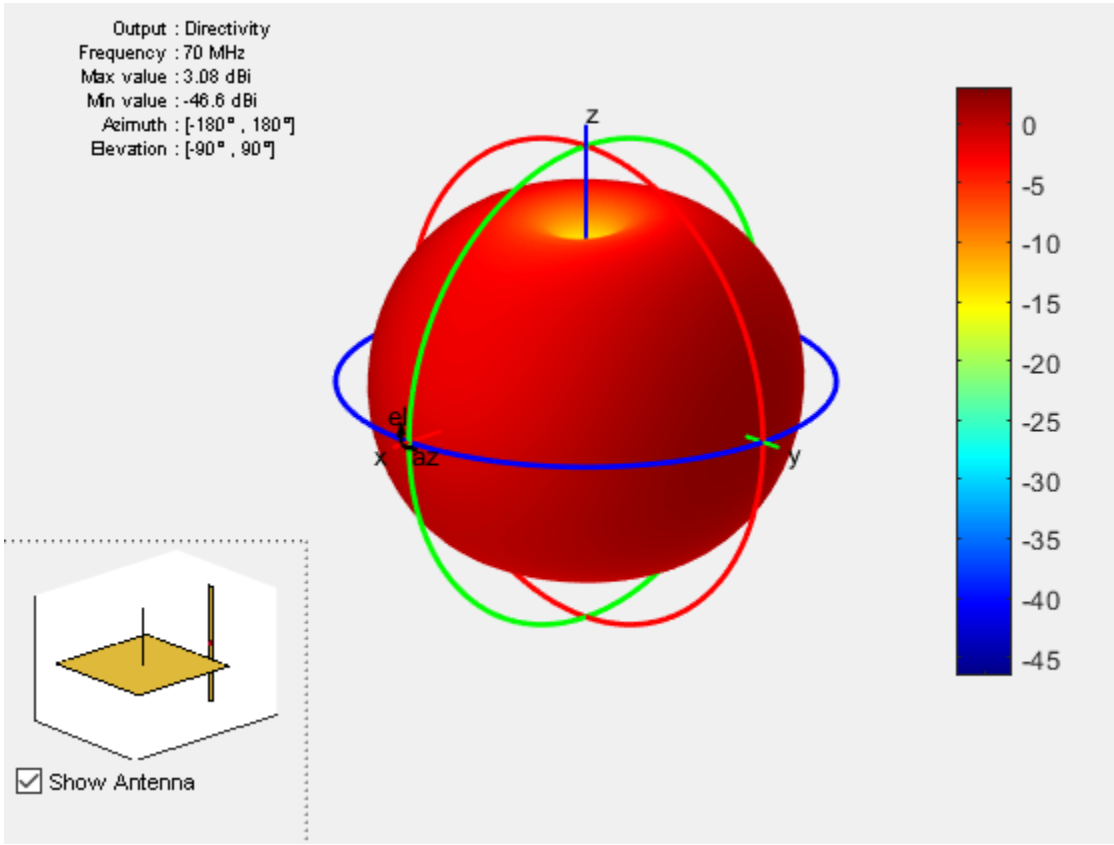
Visualize the array.

```
figure;
show(c);
```



Plot the radiation pattern of the array at 70 MHz.

```
pattern(c, 70e6)
```

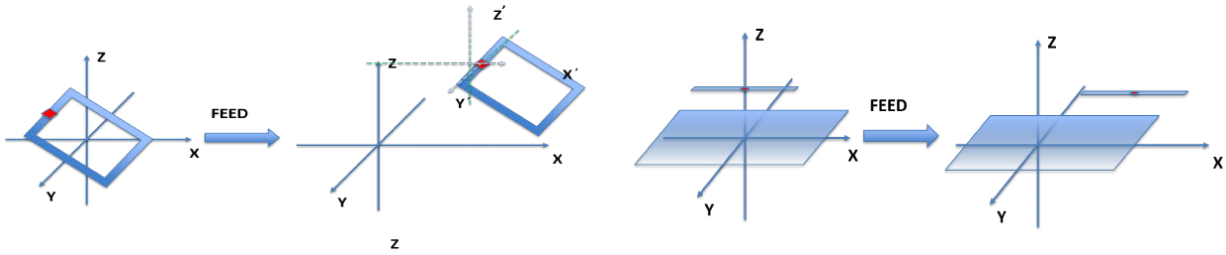


Definitions

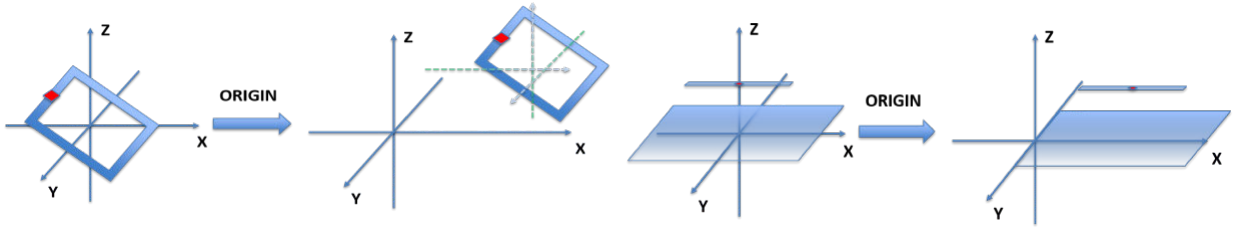
Position Reference

`'Reference'` property of `conformalArray` class defines the position reference of an antenna element in 3–D space. You can position the antenna by specifying the `Reference` property as `feed` or `origin`.

Choosing `feed` as the position reference moves the antenna element with so that the new feed location is at the specified coordinates. The loop rectangle antenna and reflector-backed antenna show the new position with respect to feed:



Choosing `origin` as the position reference moves the antenna element so that new antenna origin is at the specified coordinates. The loop rectangle antenna and reflector-backed antenna show the new position with respect to origin:



References

[1] Balanis, Constantine A. *Antenna Theory: Analysis and Design*. 3rd Ed. New York: John Wiley and Sons, 2005.

See Also

`circularArray` | `infiniteArray` | `linearArray` | `rectangularArray`

Topics

“Rotate Antenna and Arrays”

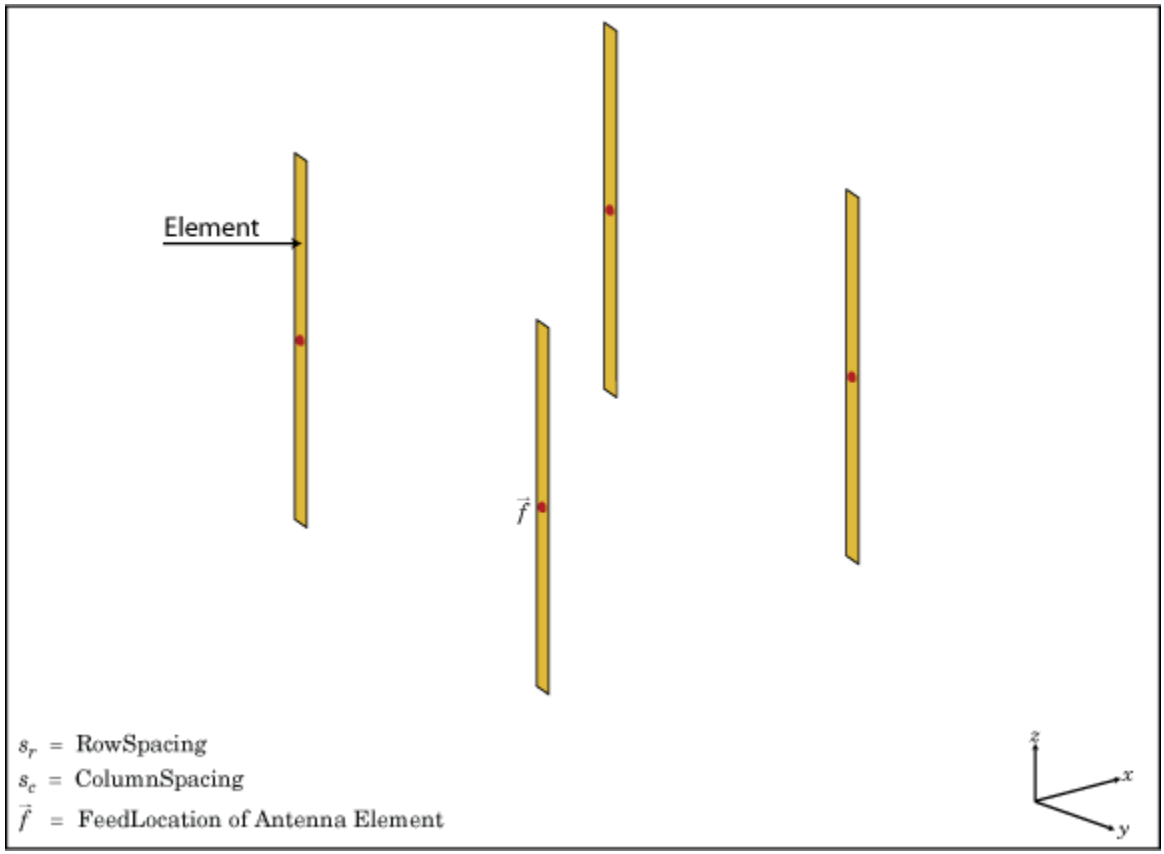
Introduced in R2016a

rectangularArray

Create rectangular antenna array

Description

The `rectangularArray` class creates a rectangular antenna array in the X-Y plane. By default, the rectangular array is a four-element dipole array in a 2 x 2 rectangular lattice. The dipoles are center-fed. Each dipole resonates at 70 MHz when isolated.



Creation

Syntax

```
ra = rectangularArray
ra = rectangularArray(Name,Value)
```

Description

`ra = rectangularArray` creates a rectangular antenna array in the X-Y plane.

`ra = rectangularArray(Name,Value)` creates a rectangular antenna array, with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain default values.

Properties

`Element` — Individual antenna elements used in array
dipole (default) | antenna object

Individual antenna elements used in array, specified as an antenna object.

Example: `'Element',monopole`

`Size` — Number of antenna elements in row and column of array
`[2 2]` (default) | two-element vector

Number of antenna elements in row and column of array, specified as a two-element vector.

Example: `'Size',[4 4]`

`RowSpacing` — Row spacing between two antenna elements
2 (default) | scalar | vector

Row spacing between two antenna elements, specified as a scalar or vector in meters. By default, the antenna elements are spaced `2m` apart.

Example: `'RowSpacing',[5 6]`

Data Types: `double`

`ColumnSpacing` — Column spacing between two antenna elements
2 (default) | scalar | vector

Column spacing between two antenna elements, specified as a scalar or vector in meters. By default, the antenna elements are spaced `2m` apart.

Example: `'ColumnSpacing',[3 4]`

Data Types: `double`

`Lattice` — Antenna elements spatial arrangement
`'Rectangular'` (default) | `"Triangular"`

Antenna elements spatial arrangement, specified as a text input.

Example: `'Lattice',"Triangular"`

Data Types: `char` | `string`

`AmplitudeTaper` — Excitation amplitude of antenna elements
1 (default) | scalar | vector

Excitation amplitude of antenna elements, specified as a scalar or vector. Set the property value to `0` to model dead elements.

Example: `'AmplitudeTaper',3`

Data Types: `double`

`Phaseshift` — Phase shift for antenna elements
0 (default) | scalar | vector

Phase shift for antenna elements, specified as a scalar or vector in degrees.

Example: `'PhaseShift',[3 3 0 0]`

Data Types: `double`

`Tilt` — Tilt angle of array
0 (default) | scalar | vector

Tilt angle of array, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of array
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
correlation	Correlation coefficient between two antennas in array
current	Current distribution on metal or dielectric antenna or array surface
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure

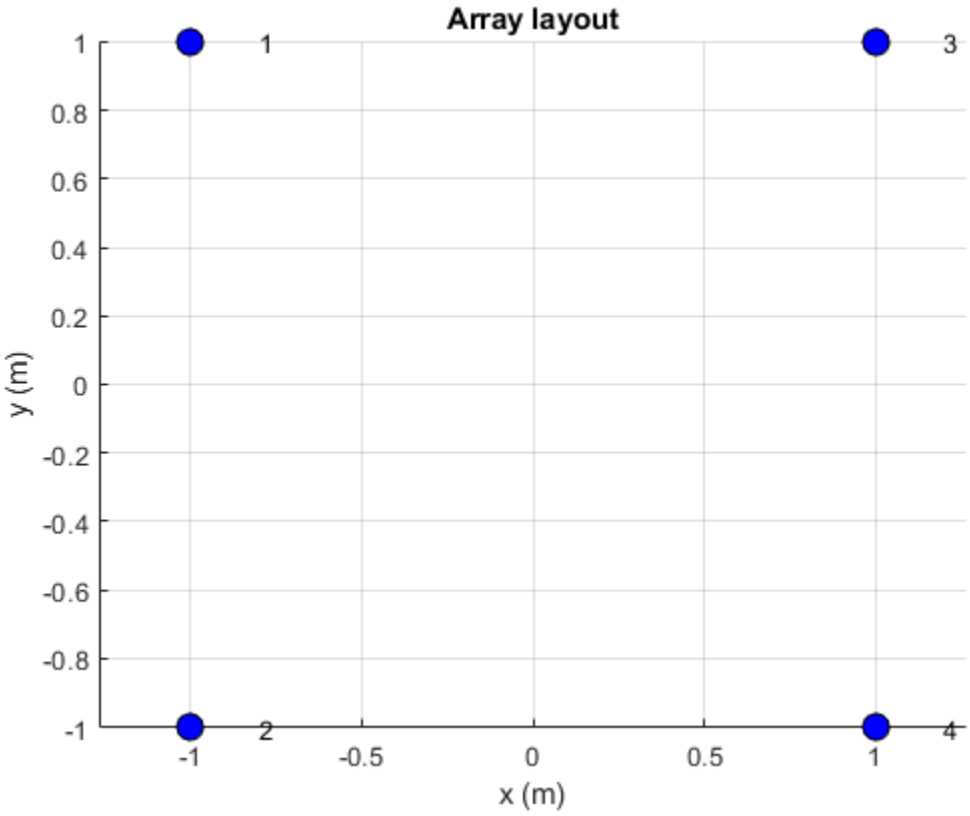
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object

Examples

Create and Plot Layout of Rectangular Array

Create and plot the layout of a rectangular array of four dipoles.

```
ra = rectangularArray;
ra.Size = [2 2];
layout(ra);
```



Calculate Scan Impedance of Rectangular Array

Calculate the scan impedance of a 2x2 rectangular array of dipoles at 70 MHz.

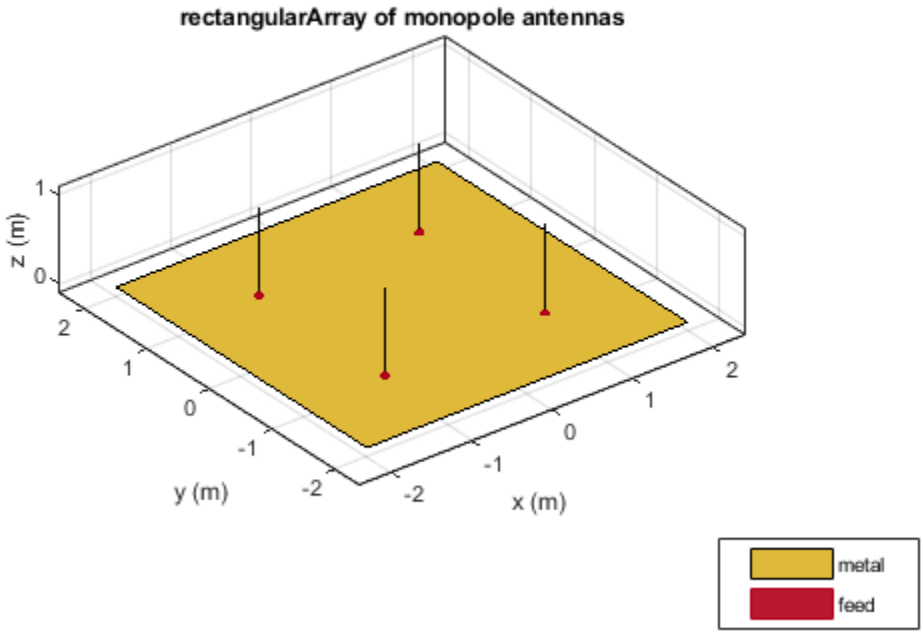
```
h = rectangularArray('Size',[2 2]);
Z = impedance(h,70e6)
```

```
Z = 1×4 complex

  26.2446 -57.3842i  26.2477 -57.3833i  26.2446 -57.3842i  26.2477 -57.3833i
```

Rectangular Array Using Groundplane Antennas

Create a rectangular array of monopoles.

```
m1 = monopole;
mra = rectangularArray('Element',m1);
show(mra);
```



References

[1] Balanis, C.A. *Antenna Theory. Analysis and Design*, 3rd Ed. New York: Wiley, 2005.

See Also

`circularArray` | `conformalArray` | `infiniteArray` | `linearArray`

Topics

“Rotate Antenna and Arrays”

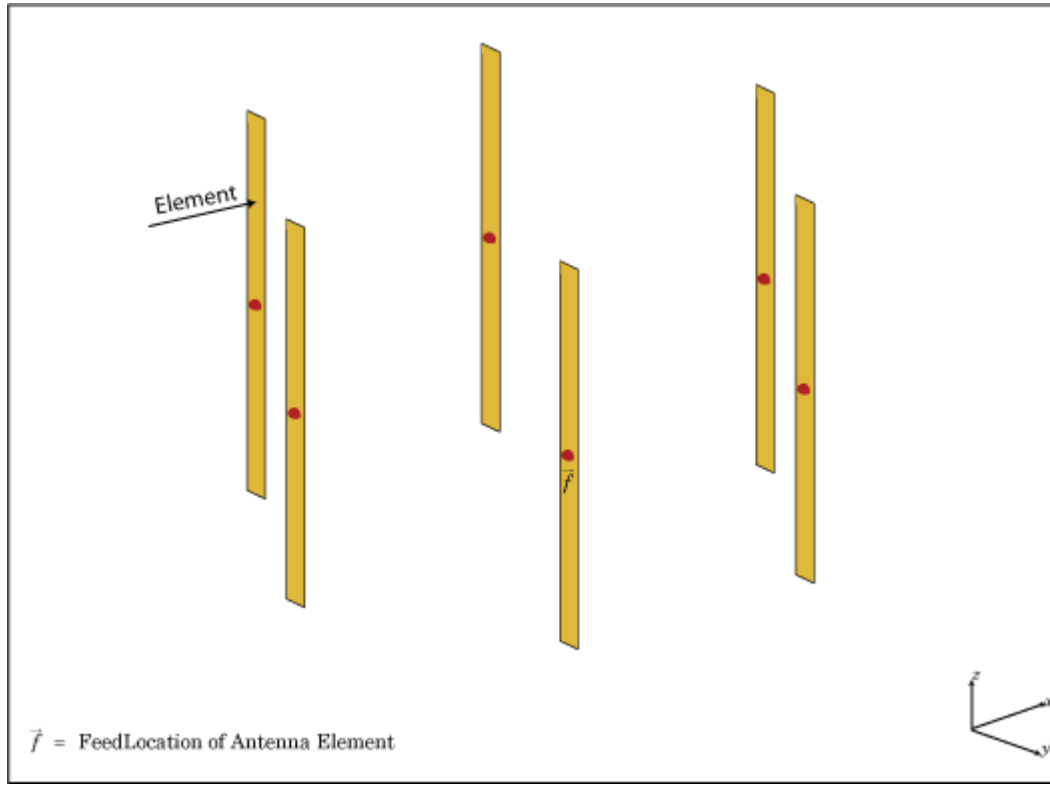
Introduced in R2015a

circularArray

Create circular antenna array

Description

The `circularArray` object is a circular antenna array. Circular array finds application in direction of arrival (DoA) systems. You can sue circular arrays to perform 2-D scanning, while lowering element counts. These arrays also have the ability for 360-degree scanning. The individual elements in the circular array are part of the same array environment. This property reduces the impact of edge effects and other coupling variation.



Creation

Syntax

```
ca = circularArray
ca = circularArray(Name,Value)
```

Description

`ca = circularArray` creates a circular antenna array in the X-Y plane.

`ca = circularArray(Name,Value)` class to create a circular antenna array, with additional properties specified by one, or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`,`...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Element` — Individual antenna type
`dipole` (default) | vector of objects

Individual antenna type, specified as a vector of objects. This property supports scalar expansion.

Example: `'Element',[monopole,monopole]`

Data Types: `char` | `string`

`NumElements` — Number of elements in array
6 (default) | positive scalar integer

Number of elements in the array,specified as a positive scalar integer. The elements in the array are arranged along the X-axis.

Example: `'NumElements',4`

Data Types: `double`

`Radius` — Radius of array
`1` (default) | positive scalar integer

Radius of array, specified as a positive scalar integer in meters.

Example: `'Radius',0.4`

Data Types: `double`

`AngleOffset` — Starting angle offset for first element in array
`0` (default) | real scalar

Starting angle offset for first element in array, specified as a real scalar in degrees.

Example: `'AngleOffset',8`

Data Types: `double`

`AmplitudeTaper` — Excitation amplitude for antenna elements in array
`1` (default) | real positive vector of size `'Element'`

Excitation amplitude for antenna elements in the array, specified as a real positive vector of size `'Element'`.

Example: `'AmplitudeTaper',[0 1]`

Data Types: `double`

`PhaseShift` — Phase shift for each element in array
`0` (default) | real vector of size `'Element'` in degrees

Phase shift for each element in the array, specified as a real vector of size `'Element'` in degrees.

Example: `'PhaseShift',[0 2]`

Data Types: `double`

`Tilt` — Tilt angle of array
0 (default) | scalar | vector

Tilt angle of array, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of array
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Analysis Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
layout	Display array layout
show	Display antenna or array structure; Display shape as filled patch
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
correlation	Correlation coefficient between two antennas in array
current	Current distribution on metal or dielectric antenna or array surface
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array

patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object

Examples

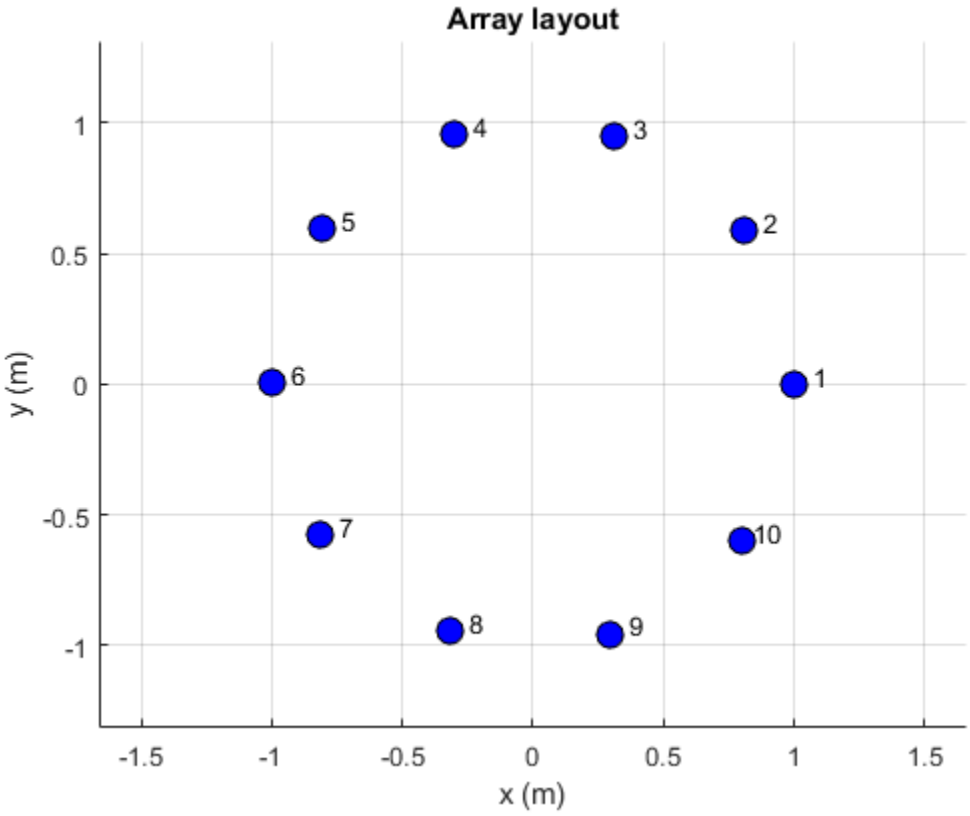
Plot Elevation Pattern of Circular Antenna Array

Create a circular antenna array using 10 antenna elements. View the layout of the antenna elements in the array.

```
ca = circularArray('NumElements',10)
```

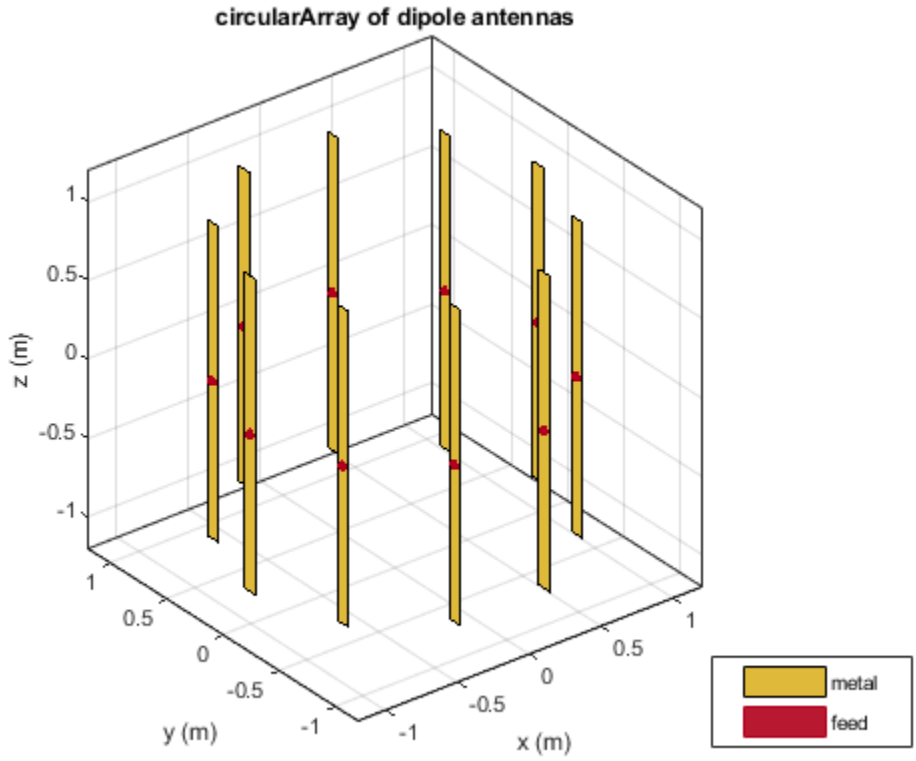
```
ca =
  circularArray with properties:

           Element: [1x1 dipole]
       NumElements: 10
            Radius: 1
       AngleOffset: 0
    AmplitudeTaper: 1
        PhaseShift: 0
              Tilt: 0
         TiltAxis: [1 0 0]
```

```
figure;
layout(ca)
```



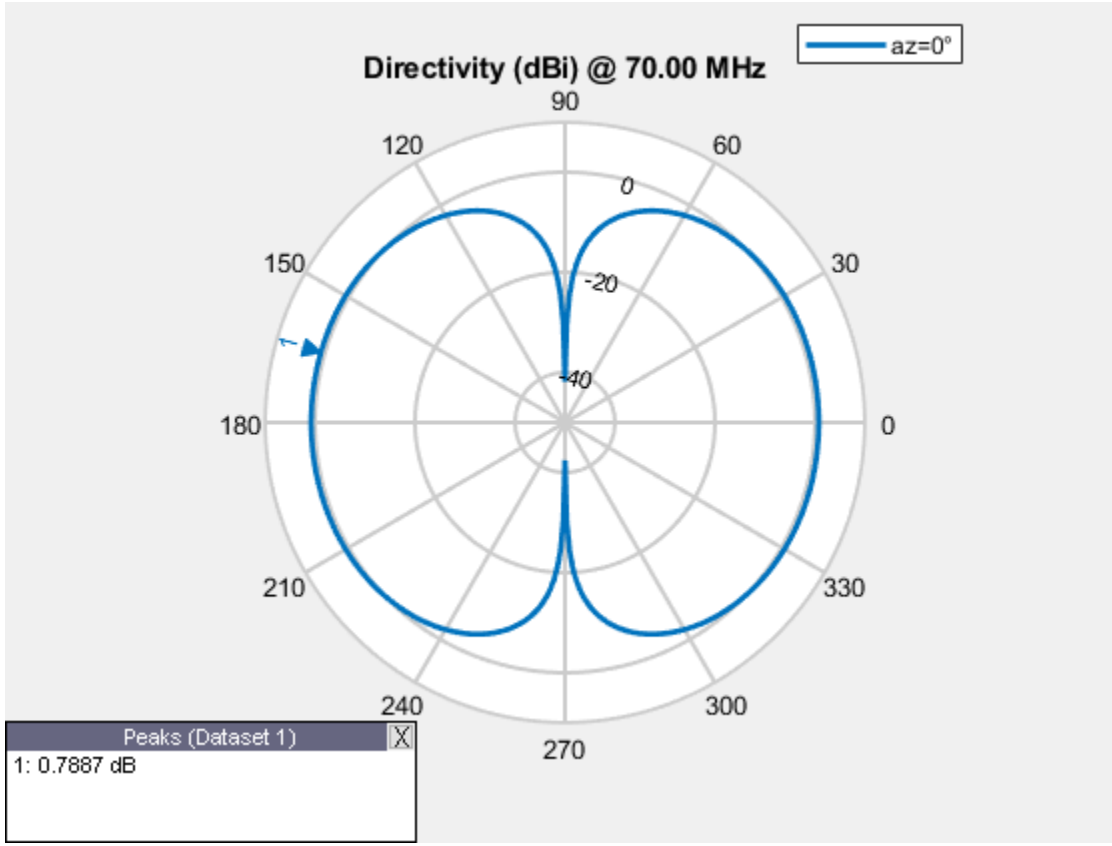
Display the array.

```
figure;
show(ca)
```



Plot the elevation pattern of the circular array at a frequency of 70 MHz.

```
figure;
patternElevation(ca,70e6)
```



See Also

`conformalArray` | `infiniteArray` | `linearArray` | `rectangularArray`

Topics

“Rotate Antenna and Arrays”

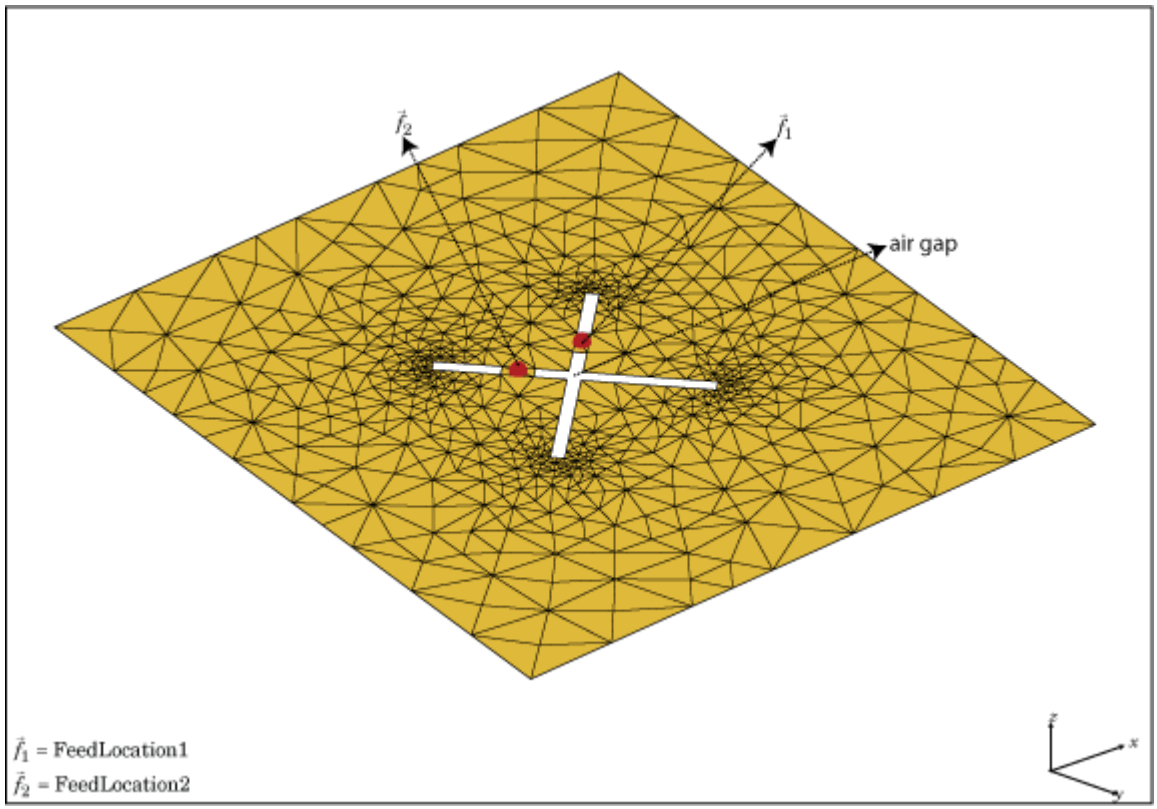
Introduced in R2016b

customArrayMesh

Create 2-D custom mesh antenna on X-Y plane

Description

The `customArrayMesh` object creates an array represented by a 2-D custom mesh on the X-Y plane. You can provide an arbitrary array mesh to the Antenna Toolbox and analyze this mesh as a custom array for port and field characteristics.



Creation

Description

`customarray = customArrayMesh(points,triangles,numfeeds)` creates a 2-D array represented by a custom mesh, based on the specified points and triangles.

Input Arguments

`points` — Points in custom mesh

`2-by-N` or `3-by-N` matrix of Cartesian coordinates in meters

Points in custom mesh, specified as a `2-by-N` or `3-by-N` matrix of Cartesian coordinates in meters. *N* is the number of points. In case you specify a `3-by-N` integer matrix, the Z-coordinate must be zero or a constant value. This value sets the `'Points'` property in the custom array mesh.

Example: `load planarmesh.mat; c = customArrayMesh(p,t,4)`. Creates a custom array mesh from the points, `p`, extracted from the `planarmesh.mat` file.

Data Types: `double`

`triangles` — Triangles in mesh

`4-by-M` matrix

Triangles in the mesh, specified as a `4-by-M` matrix. *M* is the number of triangles. The first three rows are indices to the points matrix and represent the vertices of each triangle. The fourth row is a domain number useful for identifying separate parts of an array. This value sets the `'Triangles'` property in the custom array mesh.

Example: `load planarmesh.mat; c = customArrayMesh(p,t,4)`. Creates a custom array mesh from the triangles, `t`, extracted from the `planarmesh.mat` file.

Data Types: `double`

`numfeeds` — Number of feeding points in array

2 (default) | scalar

Number of feeding points in array, specified as a scalar. By default, the number of feed points are 2.

Example: `load planarmesh.mat; c = customArrayMesh(p,t,4)`. Creates a custom array mesh requiring 4 feed points.

Data Types: `double`

Properties

`Points` — Points in custom mesh
`2-by-N` or `3-by-N` matrix of Cartesian coordinates

Points in a custom mesh, specified as a `2-by-N` or `3-by-N` matrix of Cartesian coordinates in meters. *N* is the number of points.

Data Types: `double`

`Triangles` — Triangles in mesh
`4-by-M` matrix

Triangles in the mesh, specified as a `4-by-M` matrix. *M* is the number of triangles.

Data Types: `double`

`'NumFeeds'` — Number of feeding points
scalar

Number of feeding points in the array, specified as a scalar.

Data Types: `double`

`FeedLocation` — Feed location of array
Cartesian coordinates

Feed locations of array, specified as Cartesian coordinates in meters. Feed location is a read-only property. To create a feed for the 2–D custom mesh, use the `createFeed` method.

Data Types: `double`

`AmplitudeTaper` — Excitation amplitude of antenna elements
1 (default) | scalar | non-negative vector

Excitation amplitude of antenna elements, specified as a scalar or a non-negative vector. Set the property value to `0` to model dead elements.

Example: `'AmplitudeTaper',3`

Data Types: `double`

`PhaseShift` — Phase shift for antenna elements
0 (default) | scalar | real vector

Phase shift for antenna elements, specified as a scalar or a real vector in degrees.

Example: `'PhaseShift',[3 3 0 0]`. Creates a custom array mesh of four antennas with phase shifts specified.

Data Types: `double`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
createFeed	Create feed locations for custom array
beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
correlation	Correlation coefficient between two antennas in array
current	Current distribution on metal or dielectric antenna or array surface
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object

Examples

Custom Array Mesh Impedance.

Load a custom mesh and create an array.

```
load planarmesh.mat;
c = customArrayMesh(p,t,2);
```

Create feeds for the custom array mesh.

```
createFeed(c,[0.07,0.01],[0.05,0.05], [-0.07,0.01],[-0.05,0.05])
```

Calculate the impedance of the array.

```
Z = impedance(c,1e9)
```

```
Z = 1×2 complex

  64.3919 - 7.8288i  58.9595 -11.3554i
```

References

[1] Balanis, C.A. *Antenna Theory: Analysis and Design*. 3rd Ed. New York: Wiley, 2005.

See Also

`conformalArray` | `linearArray` | `rectangularArray`

Topics

“Rotate Antenna and Arrays”

Introduced in R2015b

customArrayGeometry

Create array represented by 2-D custom geometry

Description

The `customArrayGeometry` object is an array represented by a 2-D custom geometry on the X-Y plane. You can use the `customArrayGeometry` to import a 2-D custom geometry, define feeds to create an array element, and analyze the custom array.

Creation

Syntax

```
cg = customArrayGeometry
ca = customArrayGeometry(Name,Value)
```

Description

`cg = customArrayGeometry` creates a custom array represented by 2-D geometry on the X-Y plane, based on the specified boundary.

`ca = customArrayGeometry(Name,Value)` creates a 2-D array geometry, with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`Boundary` — Boundary information in Cartesian coordinates
cell array

Boundary information in Cartesian coordinates, specified as a cell array in meters.

Data Types: `double`

`Operation` — Boolean operation performed on boundary list
`'P1'` (default) | character vector

Boolean operation performed on the boundary list, specified as a character vector. operation set is; [+, -, *].

Example: `'Operation','P1-P2'`

Data Types: `double`

`FeedLocation` — Array element feed location in Cartesian coordinates
`[0 0 0]` (default) | three-element vector

Array element feed location in Cartesian coordinates, specified as a three-element vector. The three elements represent the X, Y, and Z coordinates respectively.

Example: `'FeedLocation', [0 0.2 0]`

Data Types: `double`

`FeedWidth` — Width of feed for array elements
`0.0100` (default) | scalar

Width of feed for array elements, specified as a scalar in meters.

Example: `'FeedWidth',0.05`

Data Types: `double`

`AmplitudeTaper` — Excitation amplitude for array elements
`1` (default) | non-negative scalar | vector of non-negative scalars

Excitation amplitude for array elements, specified as a non-negative scalar or vector of non-negative scalars. Set property value to `0` to model dead elements.

Example: `'AmplitudeTaper',3`

Data Types: `double`

`PhaseShift` — Phase shift for array elements
`0` (default) | real scalar | real vector

Phase shift for array elements, specified as a real scalar in degrees or a real vector in degrees.

Example: `'PhaseShift',[3 3 0 0]` specified the phase shift for custom array containing four elements.

Data Types: `double`

`Tilt` — Tilt angle of array
0 (default) | scalar | vector

Tilt angle of array, specified as a scalar or vector with each element unit in degrees.

Example: `'Tilt',90`

Example: `'Tilt',[90 90 0]`

Data Types: `double`

`TiltAxis` — Tilt axis of array
`[1 0 0]` (default) | three-element vectors of Cartesian coordinates | two three-element vector of Cartesian coordinates | `'X'` | `'Y'` | `'Z'`

Tilt axis of the antenna, specified as:

- A three-element vector of Cartesian coordinates in meters. In this case, each vector starts at the origin and lies along the specified points on the X-, Y-, and Z axes.
- Two points in space, each specified as a three-element vector of Cartesian coordinates. In this case, the antenna rotates around the line joining the two points in space.
- A string input describing simple rotations around one of the principal axes, 'X', 'Y', or 'Z'.

For more information, see “Rotate Antenna and Arrays”.

Example: `'TiltAxis',[0 1 0]`

Example: `'TiltAxis',[0 0 0;0 1 0]`

Example: `ant.TiltAxis = 'Z'`

Data Types: `double` | `char` | `string`

Object Functions

show	Display antenna or array structure; Display shape as filled patch
info	Display information about antenna or array
axialRatio	Axial ratio of antenna

beamwidth	Beamwidth of antenna
charge	Charge distribution on metal or dielectric antenna or array surface
current	Current distribution on metal or dielectric antenna or array surface
design	Design prototype antenna or arrays for resonance at specified frequency
EHfields	Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays
impedance	Input impedance of antenna; scan impedance of array
mesh	Mesh properties of metal or dielectric antenna or array structure
meshconfig	Change mesh mode of antenna structure
pattern	Radiation pattern of antenna or array; Embedded pattern of antenna element in array
patternAzimuth	Azimuth pattern of antenna or array
patternElevation	Elevation pattern of antenna or array
returnLoss	Return loss of antenna; scan return loss of array
sparameters	S-parameter object
show	Display antenna or array structure; Display shape as filled patch
vswr	Voltage standing wave ratio of antenna

Examples

Create Custom Slot Antenna Array

Create a custom array using `customArrayGeometry`. Visualize it and plot the impedance. Also, visualize the current distribution on the array.

Create a ground plane with a length of 0.6 m and a width of 0.5 m.

```
Lp  = 0.6;
Wp  = 0.5;
[~,p1]   = em.internal.makeplate(Lp,Wp,2,'linear');
```

Create slots on the ground plane with a length 0.05 m and a width of 0.4 m.

```
Ls  = 0.05;
Ws  = 0.4;
offset = 0.12;
[~,p2]   = em.internal.makeplate(Ls,Ws,2,'linear');
p3 = em.internal.translateshape(p2, [offset, 0, 0]);
p2 = em.internal.translateshape(p2, [-offset, 0, 0]);
```

Create a feed in between the slots on the ground plane.

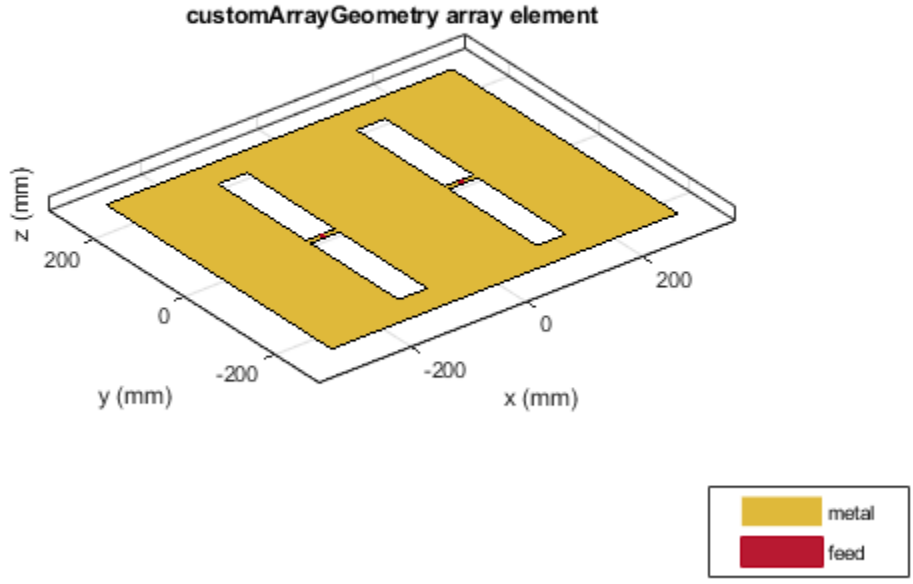
```
Wf  = 0.01;
[~,p4]   = em.internal.makeplate(Ls,Wf,2,'linear');
p5 = em.internal.translateshape(p4, [offset, 0, 0]);
p4 = em.internal.translateshape(p4, [-offset, 0, 0]);
```

Create an array using the slotted ground plane.

```
carray = customArrayGeometry;
carray.Boundary = {p1', p2', p3', p4', p5'};
carray.Operation= 'P1-P2-P3+P4+P5';
carray.NumFeeds = 2;
carray.FeedWidth= [0.01 0.01];
carray.FeedLocation = [-offset,0,0 ; offset,0,0];
```

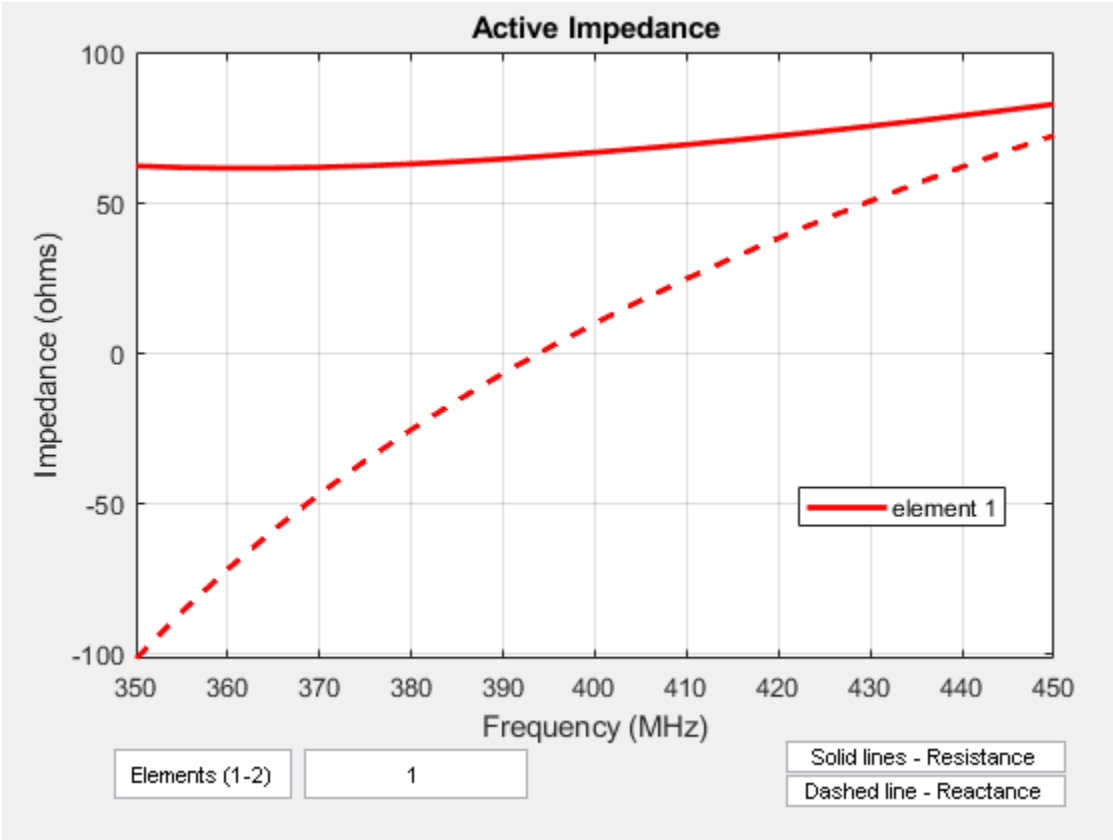
Visualize the array.

```
figure; show(carray);
```



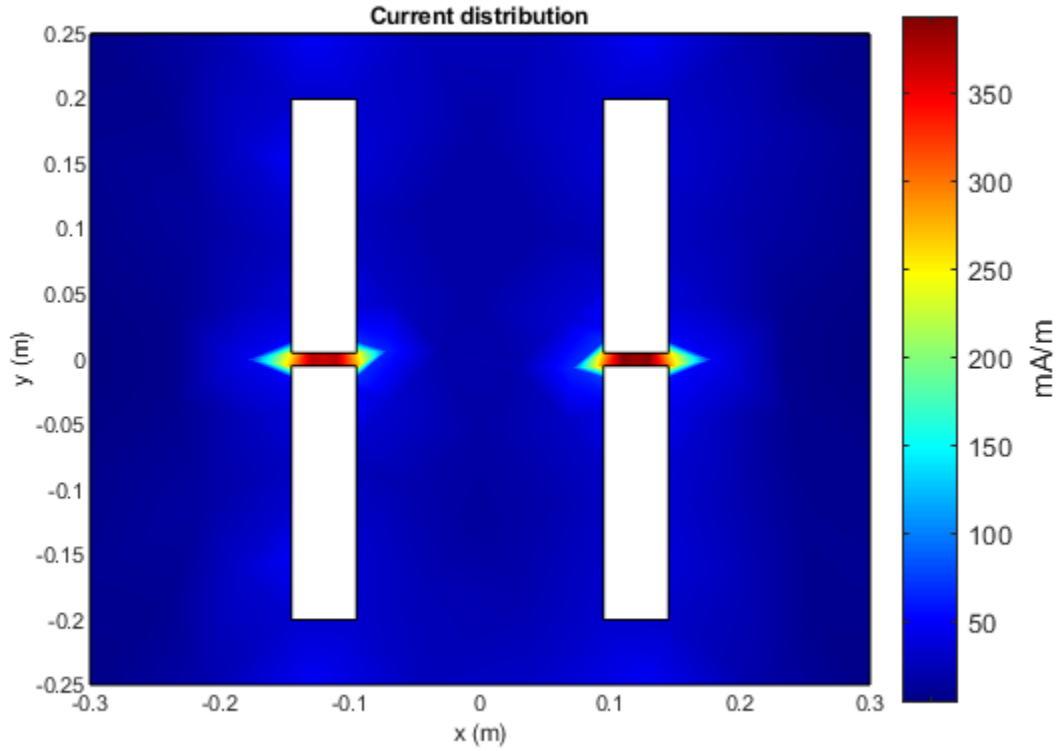
Calculate the impedance of the array using the frequency range of 350 MHz to 450 MHz.

```
figure; impedance(carray, 350e6:5e6:450e6);
```



Visualize the current distribution of the array at 410 MHz.

```
figure; current(carray, 410e6);
```



References

[1] Balanis, C. A. *Antenna Theory. Analysis and Design*. 3rd Ed. Hoboken, NJ: John Wiley & Sons, 2005.

See Also

Topics

“Rotate Antenna and Arrays”

Introduced in R2017a

Methods — Alphabetical List

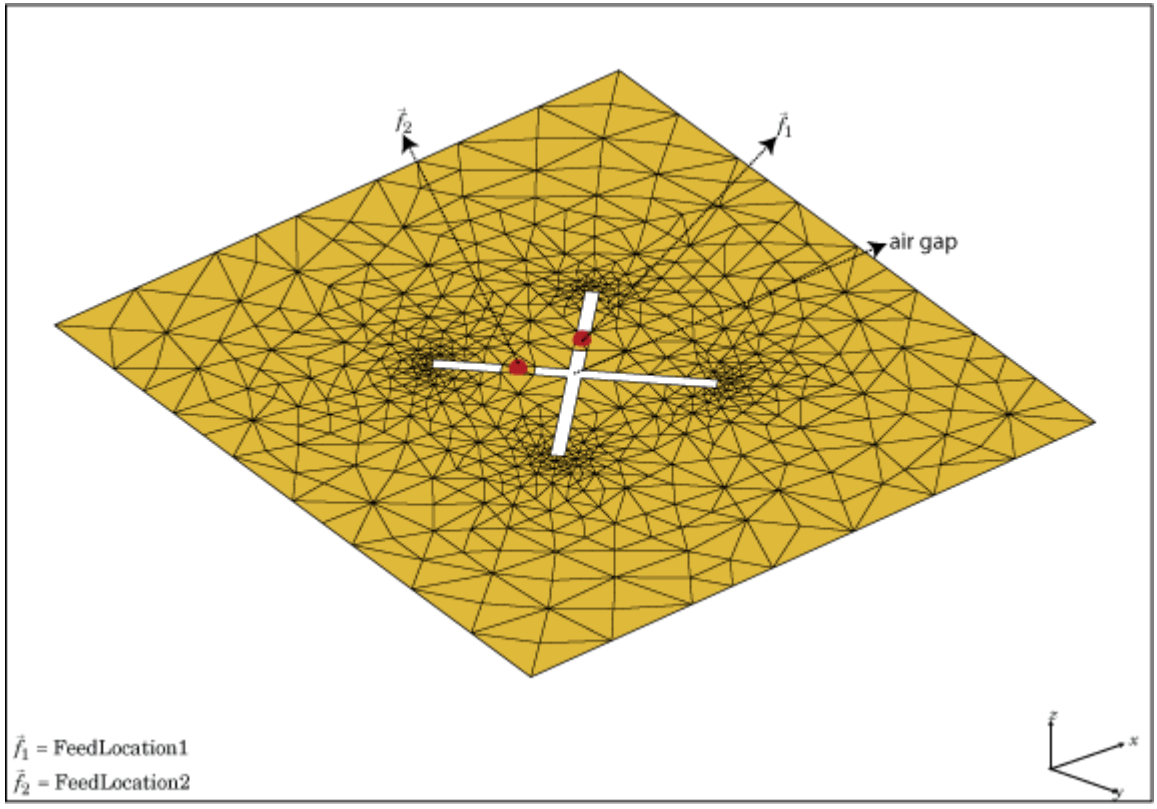
createFeed

Create feed locations for custom array

Syntax

```
createFeed(array)
createFeed(array,point1a,point1b,point2a,point2b,.....)
```

Description



`createFeed(array)` plots a custom array mesh in a figure window. From the figure window, you can specify feed locations by clicking on the mesh and create a custom array. To specify a region for the feed point, select two pairs of points, inside triangles on either side of the air gap.

`createFeed(array,point1a,point1b,point2a,point2b,.....)` creates the feed across the triangle edges identified by pairs of points (`point1a` and `point1b`, `point2a`, and `point2b`). After creating the feed, feed location is highlighted when you plot the resulting array mesh.

Input Arguments

`array` — Custom array mesh
scalar handle

Custom mesh array, specified as a scalar handle.

`point1a,point1b` — Point pairs to identify feed region
Cartesian coordinates in meters

Point pairs to identify feed region, specified as Cartesian coordinates in meters. Specify the points in the format $[x_1, y_1]$, $[x_2, y_2]$.

Example: `createFeed(c,[0.07,0.01],[0.05,0.05],[-0.07,0.01],[-0.05,0.05])`. Creates two pairs of feedpoints for a custom array mesh at the x-y coordinates specified.

Examples

Two-Feed Custom Array Mesh Using GUI

Create a custom array with two feeds.

Load a 2-D custom mesh. Create a custom array using the points and triangles.

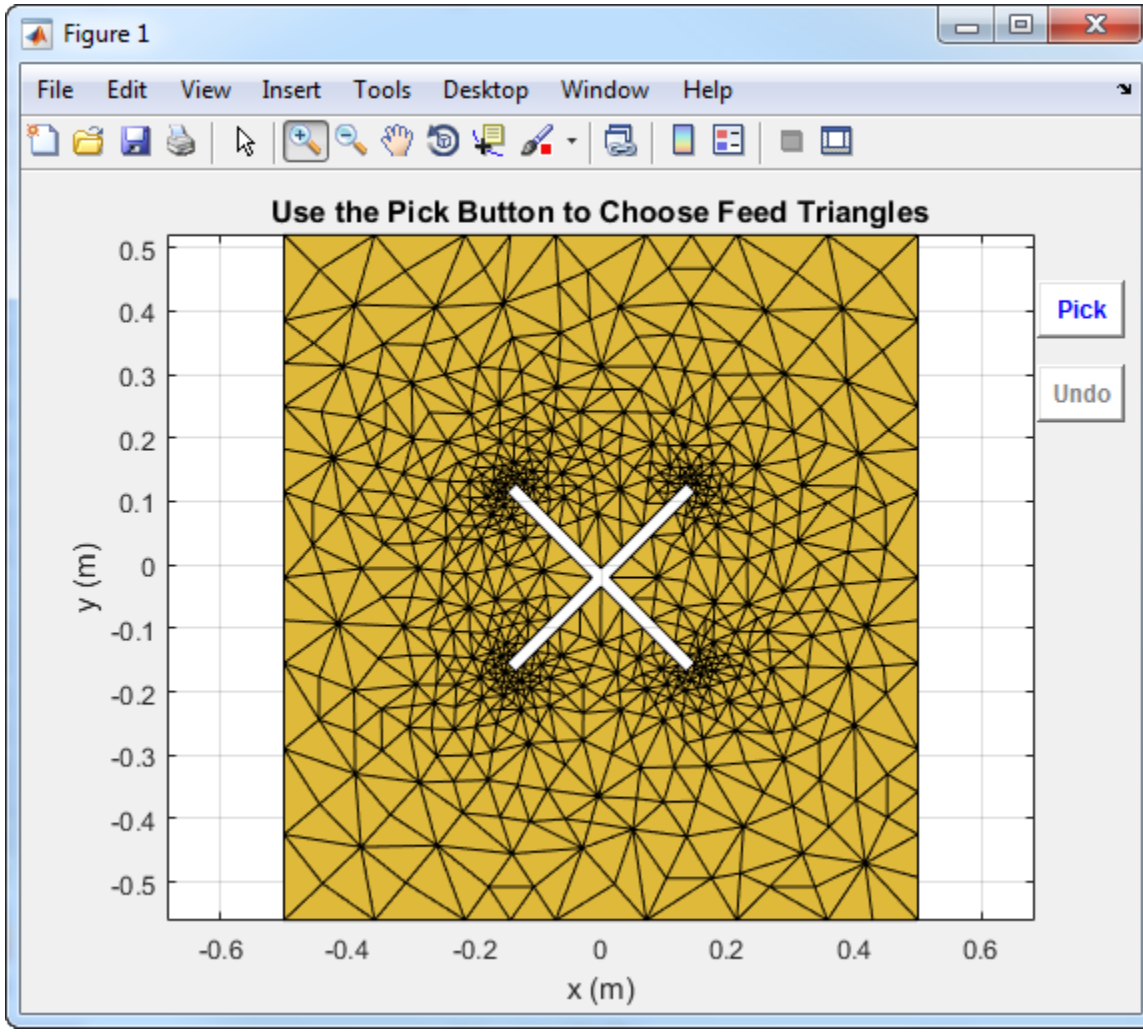
```
load planarmesh.mat;
c = customArrayMesh(p,t,2);
```

```
c =
  customArrayMesh with properties:
```

```
          Points: [3x658 double]
       Triangles: [4x1219 double]
        NumFeeds: 2
    FeedLocation: []
AmplitudeTaper: 1
     PhaseShift: 0
            Tilt: 0
        TiltAxis: [1 0 0]
```

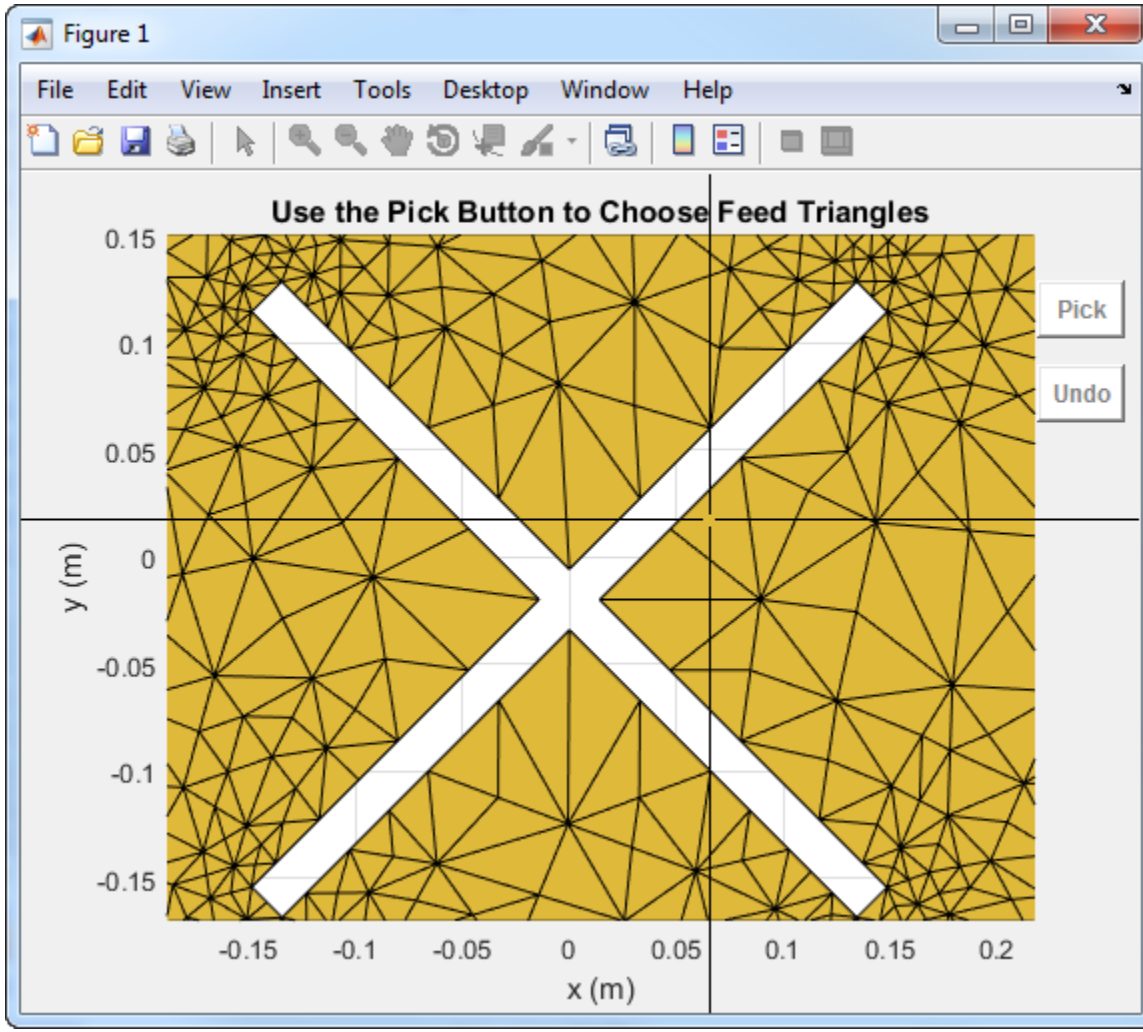
Use the `createFeed` function to view the array mesh structure. In this array mesh view, you see **Pick** and **Undo** buttons. The **Pick** button is highlighted.

```
createFeed(c)
```

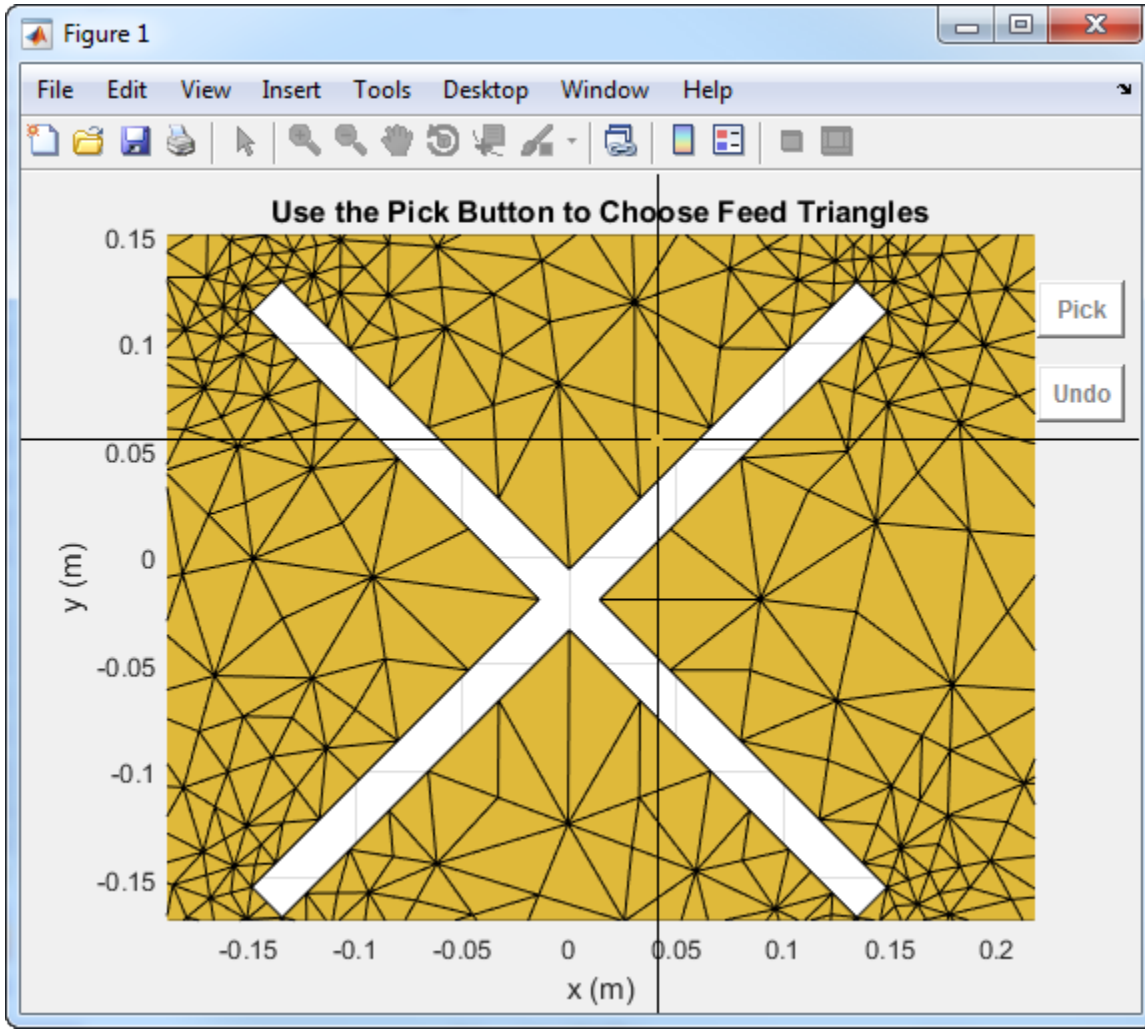


Click **Pick** to display the cross hairs. For an array with two feeds, select two pairs (four points) in the mesh. To specify a feed-region for the, zoom in and select two points each, one inside each triangle on either side of the air gap. Select the points using the cross hairs.

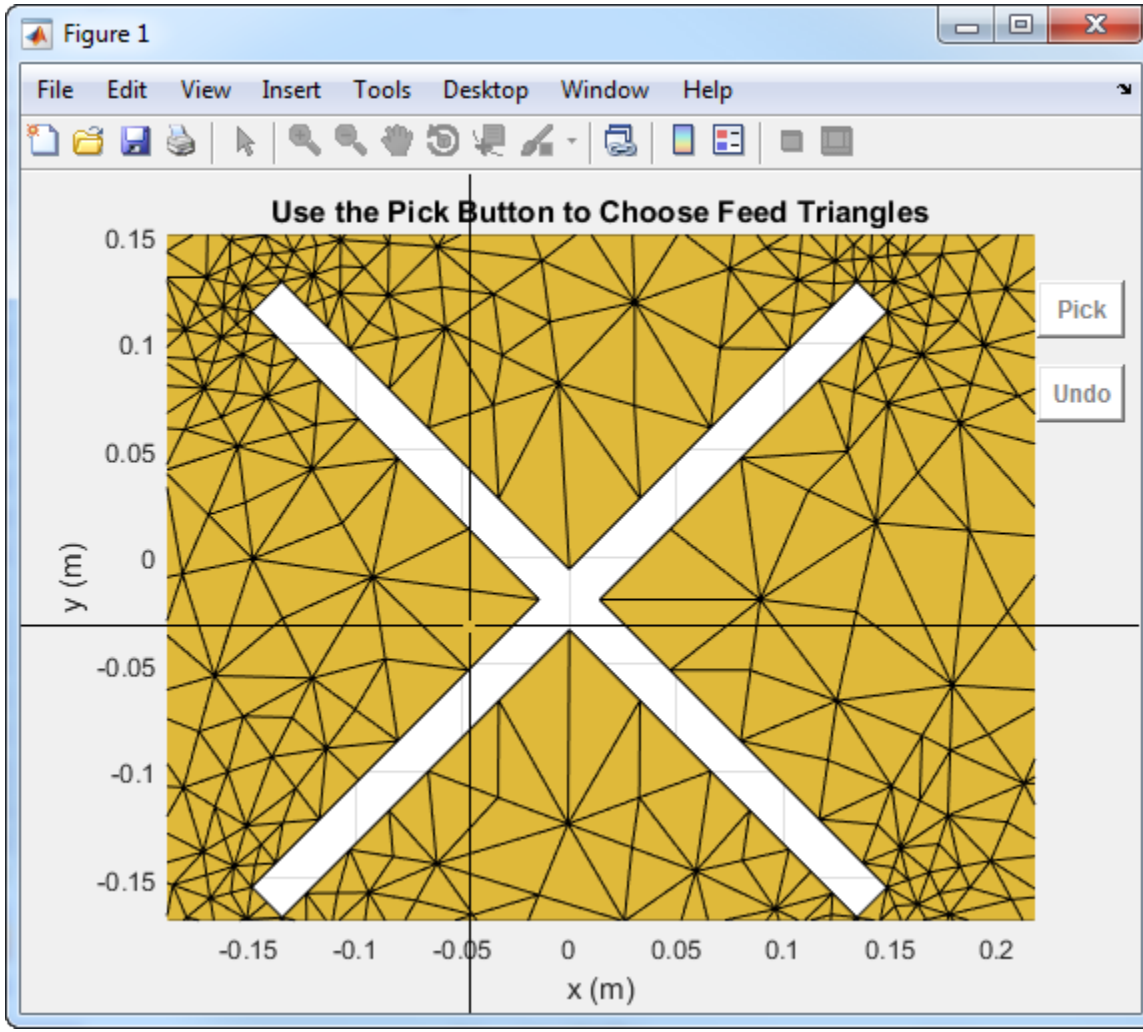
- Select the first triangle for feedpoint 1.



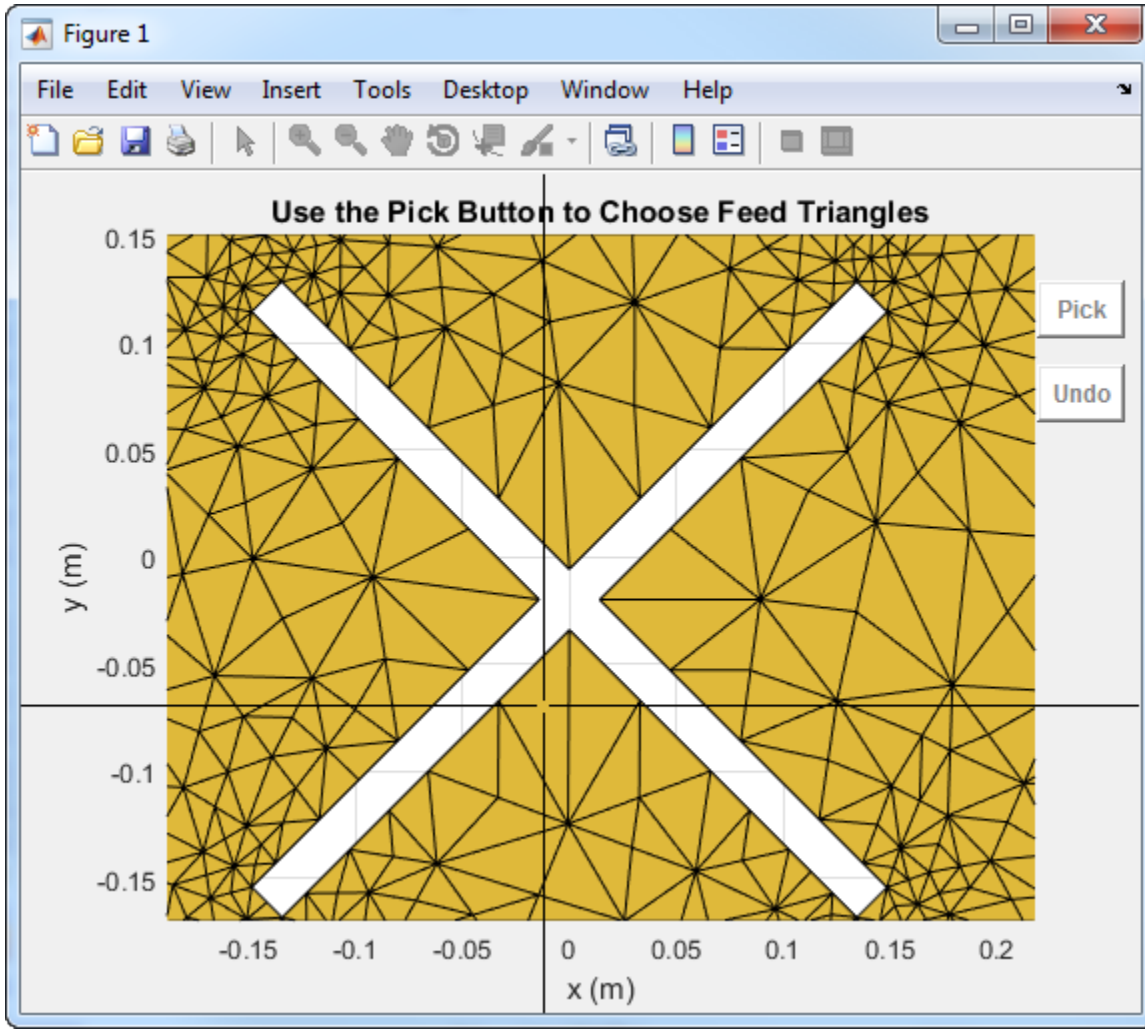
- Select the second triangle on the other side of the air gap for feedpoint 1.



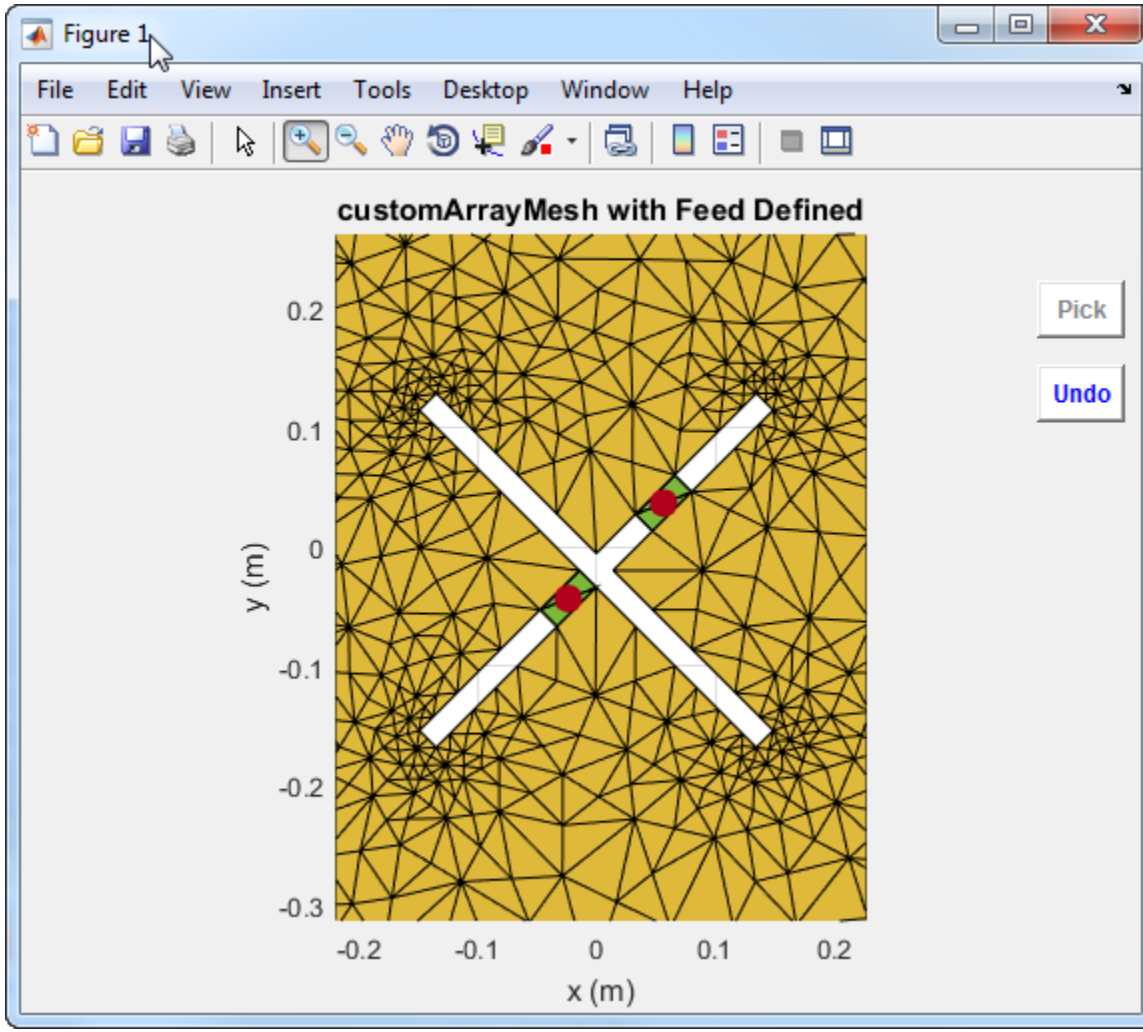
- Select first triangle for feedpoint 2.



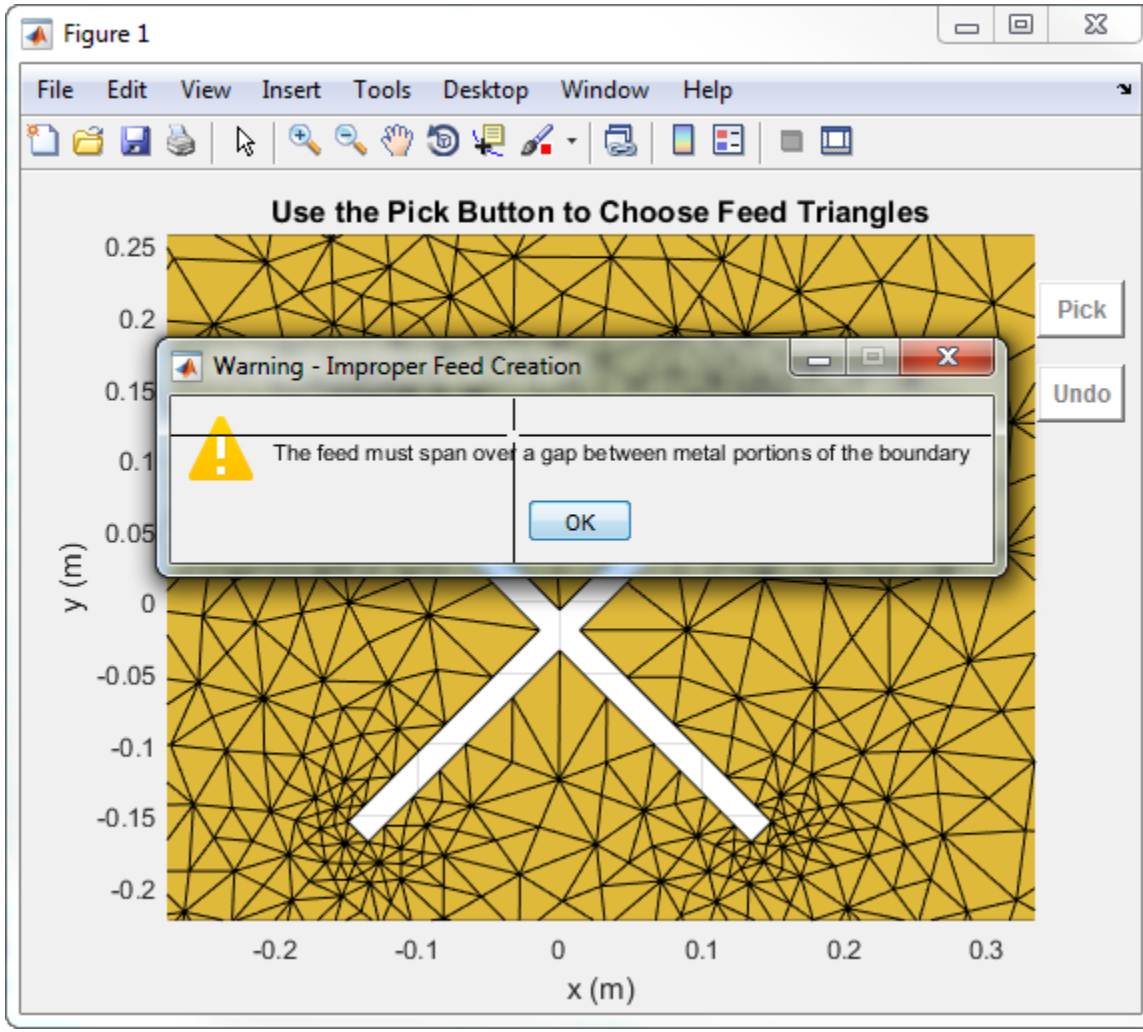
- Select the second triangle on the other side of the air gap for feedpoint 2.



Selecting the fourth triangle creates and displays the array feeds.



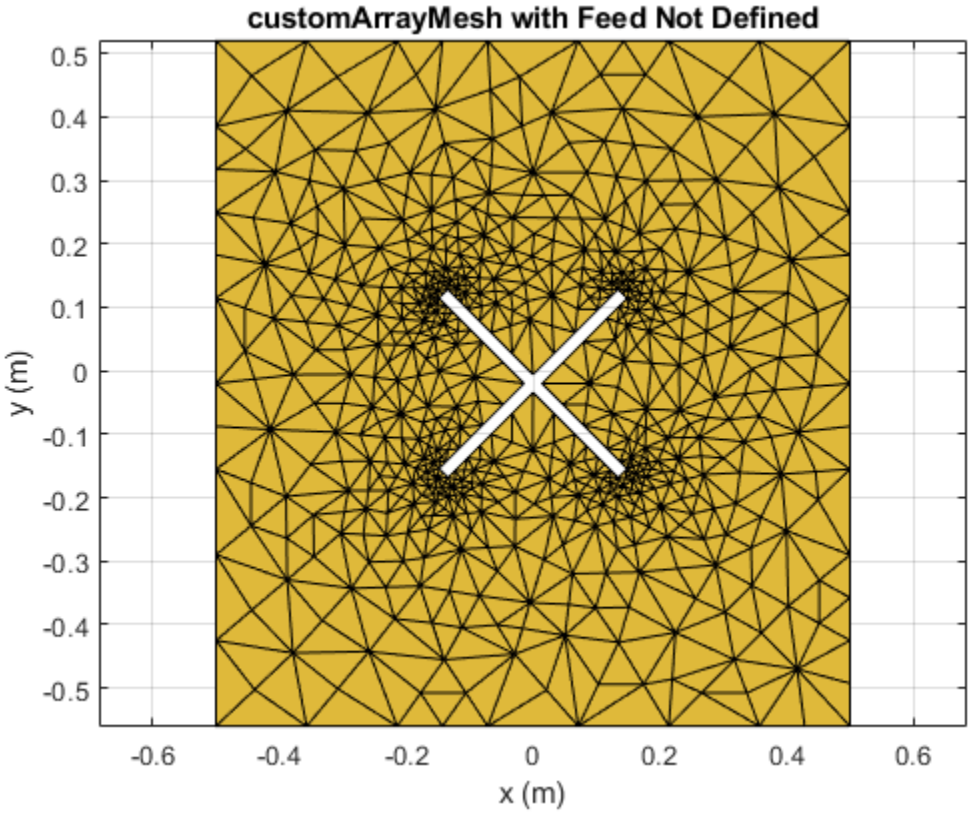
You must select the two triangles on either side of the air gap. Otherwise, the function displays an error message.



Create Feed for Custom Array Mesh

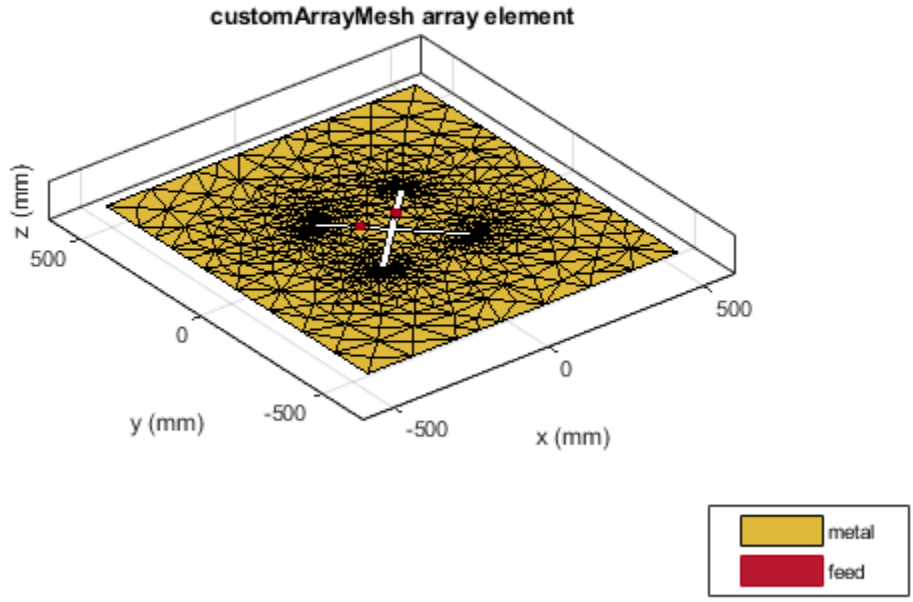
Load a custom mesh and create an array.

```
load planarmesh.mat;
c = customArrayMesh(p,t,2);
show(c)
```



Create feeds for the custom array mesh.

```
createFeed(c,[0.07,0.01],[0.05,0.05], [-0.07,0.01],[-0.05,0.05]);
show(c)
```



See Also

`returnLoss` | `sparameters`

Introduced in R2016a

impedance

Input impedance of antenna; scan impedance of array

Syntax

```
impedance(antenna,frequency)
z = impedance(antenna,frequency)

impedance(array,frequency,elementnumber)
z = impedance(array,frequency,elementnumber)
```

Description

`impedance(antenna,frequency)` calculates the input impedance of an antenna object and plots the resistance and reactance over a specified frequency.

`z = impedance(antenna,frequency)` returns the impedance of the antenna object, over a specified frequency.

`impedance(array,frequency,elementnumber)` calculates and plots the scan impedance of a specified antenna element in an array.

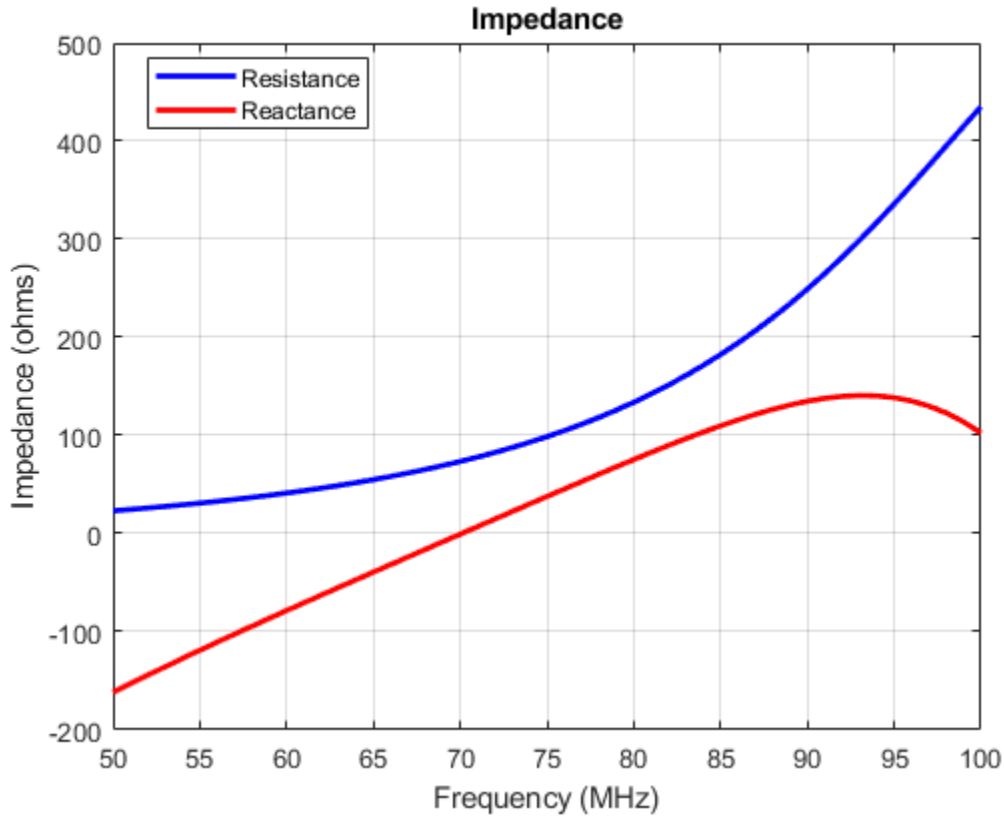
`z = impedance(array,frequency,elementnumber)` returns the scan impedance of a specified antenna element in an array.

Examples

Calculate and Plot Impedance of Antenna

Calculate and plot the impedance of a planar dipole antenna over a frequency range of 50MHz - 100MHz.

```
h = dipole;
impedance (h,50e6:1e6:100e6);
```



Calculate Scan Impedance of Array

Calculate scan impedance of default linear array over a frequency range of 50MHz to 100MHz.

```
h = linearArray;
z = impedance(h,50e6:1e6:100e6)
```

```
z = 51×2 complex
10^2 ×
```

```
0.2887 - 1.7446i   0.2887 - 1.7446i
0.2999 - 1.6634i   0.2999 - 1.6634i
0.3114 - 1.5838i   0.3114 - 1.5838i
0.3231 - 1.5059i   0.3231 - 1.5059i
0.3351 - 1.4293i   0.3351 - 1.4293i
0.3474 - 1.3541i   0.3474 - 1.3541i
0.3599 - 1.2800i   0.3599 - 1.2800i
0.3728 - 1.2069i   0.3728 - 1.2069i
0.3860 - 1.1346i   0.3860 - 1.1346i
0.3995 - 1.0631i   0.3995 - 1.0631i
   ⋮
```

Input Arguments

`antenna` — Antenna or array object

scalar handle

Antenna object, specified as a scalar handle.

`array` — Array object

scalar handle

Array object, specified as a scalar handle.

`frequency` — Frequency range used to calculate impedance

vector in Hz

Frequency range to calculate impedance, specified as a vector in Hz.

Example: 50e6:1e6:100e6

Data Types: `double`

`elementnumber` — Antenna element number in array

scalar

Antenna element number in array, specified as a scalar.

Example: 1

Data Types: `double`

Output Arguments

z — Input impedance of antenna or scan impedance of array

complex number in ohms

Input impedance of antenna or scan impedance of array, returned as a complex number in ohms. The real part of the complex number indicates the resistance. The imaginary part of the complex number indicates the reactance.

Note Antenna Toolbox caches the impedance values while running for the first time so that the subsequent runs are faster.

See Also

`returnLoss`

Introduced in R2015a

sparameters

S-parameter object

Syntax

```
sobj = sparameters(netparamobj)
sobj = sparameters(netparamobj, Z0)

sobj = sparameters(antenna,freq,Z0)
sobj = sparameters(array,freq,Z0)
```

Description

`sobj = sparameters(netparamobj)` converts the network parameter object, `netparamobj`, to S-parameter object with the default reference impedance.

`sobj = sparameters(netparamobj, Z0)` converts the network parameter object, `netparamobj`, to S-parameter object with a given reference impedance, `Z0`.

`sobj = sparameters(antenna,freq,Z0)` calculates the complex s-parameters for an `antenna` object over specified frequency values and for a given reference impedance, `Z0`.

`sobj = sparameters(array,freq,Z0)` calculates the complex s-parameters for an `array` object over specified frequency values and for a given reference impedance, `Z0`.

Examples

Calculate S-Parameter Matrix For Antenna

Calculate the complex s-parameters for a default dipole at 70MHz frequency.

```
 h = dipole
```

```
h =
  dipole with properties:
```

```
        Length: 2
         Width: 0.1000
    FeedOffset: 0
          Tilt: 0
      TiltAxis: [1 0 0]
          Load: [1x1 lumpedElement]
```

```
 sparameters (h, 70e6)
```

```
ans =
  sparameters: S-parameters object

       NumPorts: 1
    Frequencies: 70000000
     Parameters: 0.1866 - 0.0092i
      Impedance: 50

  rfparam(obj,i,j) returns S-parameter Sij
```

Calculate S-parameter Matrix For Array

Calculate the complex s-parameters for a default rectangular array at 70MHz frequency.

```
h = rectangularArray;
sparameters(h,70e6)
```

```
ans =
  sparameters: S-parameters object

       NumPorts: 4
    Frequencies: 70000000
     Parameters: [4x4 double]
      Impedance: 50

  rfparam(obj,i,j) returns S-parameter Sij
```

Input Arguments

`filterobj` — RF filter
object handle

RF filter, specified as an `rffilter` object.

`antenna` — antenna object
scalar handle

Antenna object, specified as a scalar handle.

`array` — array object
scalar handle

Array object, specified as a scalar handle.

`freq` — S-parameter frequencies
vector of positive real numbers

S-parameter frequencies, specified as a vector of positive real numbers, sorted from smallest to largest.

`Z0` — Reference impedance
`50` (default) | positive real scalar

Reference impedance in ohms, specified as a positive real scalar. You cannot specify `Z0` if you are importing data from a file. The argument Z0 is optional and is stored in the `Impedance` property.

Output Arguments

`sobj` — S-parameter data
S-parameter object

S-parameter data, returned as an object. `disp(sobj)` returns the properties of the object:

- `NumPorts` — Number of ports, specified as an integer. The function calculates this value automatically when you create the object.

- `Frequencies` — S-parameter frequencies, specified as a *K*-by-1 vector of positive real numbers sorted from smallest to largest. The function sets this property from the `filename` or `freq` input arguments.
- `Parameters` — S-parameter data, specified as an *N*-by-*N*-by-*K* array of complex numbers. The function sets this property from the `filename` or `data` input arguments.
- `Impedance` — Reference impedance in ohms, specified as a positive real scalar. The function sets this property from the `filename` or `Z0` input arguments. If no reference impedance is provided, the function uses a default value of `50`.

See Also

`circuit` | `correlation` | `impedance` | `rfparam` | `rfplot` | `smithplot` | `yparameters` | `zparameters`

Introduced in R2012a

rfparam

Extract vector of network parameters

Syntax

```
n_ij = rfparam(hnet,i,j)
```

Description

`n_ij = rfparam(hnet,i,j)` extracts the network parameter vector (*i*,*j*) from the network parameter object, `hnet`.

Examples

Create Data Vector From S-Parameter Object

Read in the file default.s2p into an sparameters object and get the S21 value.

```
S = sparameters('default.s2p');
s21 = rfparam(S,2,1)
```

```
s21 = 191×1 complex

  -0.6857 + 1.7827i
  -0.6560 + 1.7980i
  -0.6262 + 1.8131i
  -0.5963 + 1.8278i
  -0.5664 + 1.8422i
  -0.5363 + 1.8563i
  -0.5062 + 1.8700i
  -0.4760 + 1.8835i
  -0.4457 + 1.8966i
  -0.4152 + 1.9094i
      ⋮
```

Input Arguments

`hnet` — Network parameters
network parameter object

Network parameters, specified as an RF Toolbox™ network parameter object.

`i` — Row index
positive integer

Row index of data to extract, specified as a positive integer.

`j` — Column index
positive integer

Column index of data to extract, specified as a positive integer.

Output Arguments

`n_ij` — Network parameters (*i*, *j*)
vector

Network parameters (*i*, *j*), returned as a vector. The `i` and `j` input arguments determine which parameters the function returns.

Example: `S_21 = rfparam(hs,2,1)`

See Also

`rfinterp1` | `rfplot` | `rfplot` | `sparameters` | `sparameters`

Introduced before R2006a

rfplot

Plot S-parameter data

Syntax

```
rfplot(s_obj)
rfplot(s_obj,i,j)
rfplot( ___ ,lineSpec)
rfplot( ___ ,plotflag)
hline = rfplot( ___ )
```

Description

`rfplot(s_obj)` plots the magnitude in dB versus frequency of all S-parameters (S_{11}, S_{12} ... S_{NN}) on the current axis. `s_obj` must be an s-parameter object.

`rfplot(s_obj,i,j)` plots the magnitude of $S_{i,j}$, in decibels, versus frequency on the current axis.

`rfplot( ___ ,lineSpec)` plots S-parameters using optional line types, symbols, and colors specified by `linespec`.

`rfplot( ___ ,plotflag)` allows to specify the type of plot by using the `plotflag`.

`hline = rfplot( ___ )` plots the S-parameters and returns the column vector of handles to the line objects, `hline`.

Examples

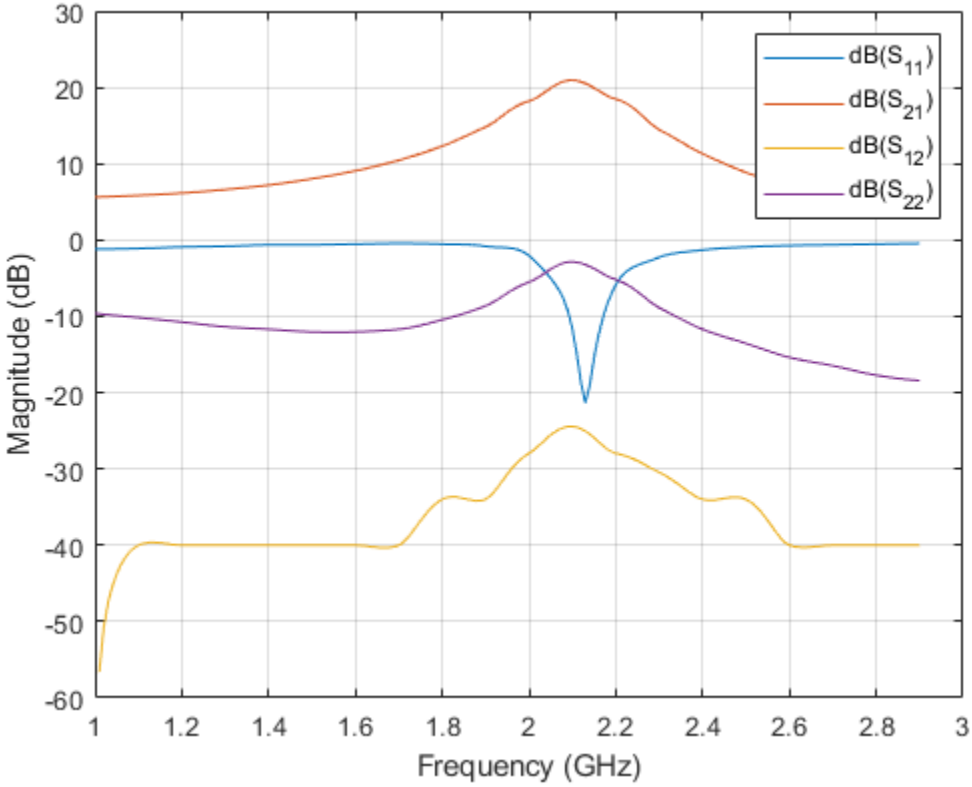
Plot S-Parameter Data Using `rfplot`

Use `sparameters` to create a set S-parameters.

```
hs = sparameters('default.s2p');
```

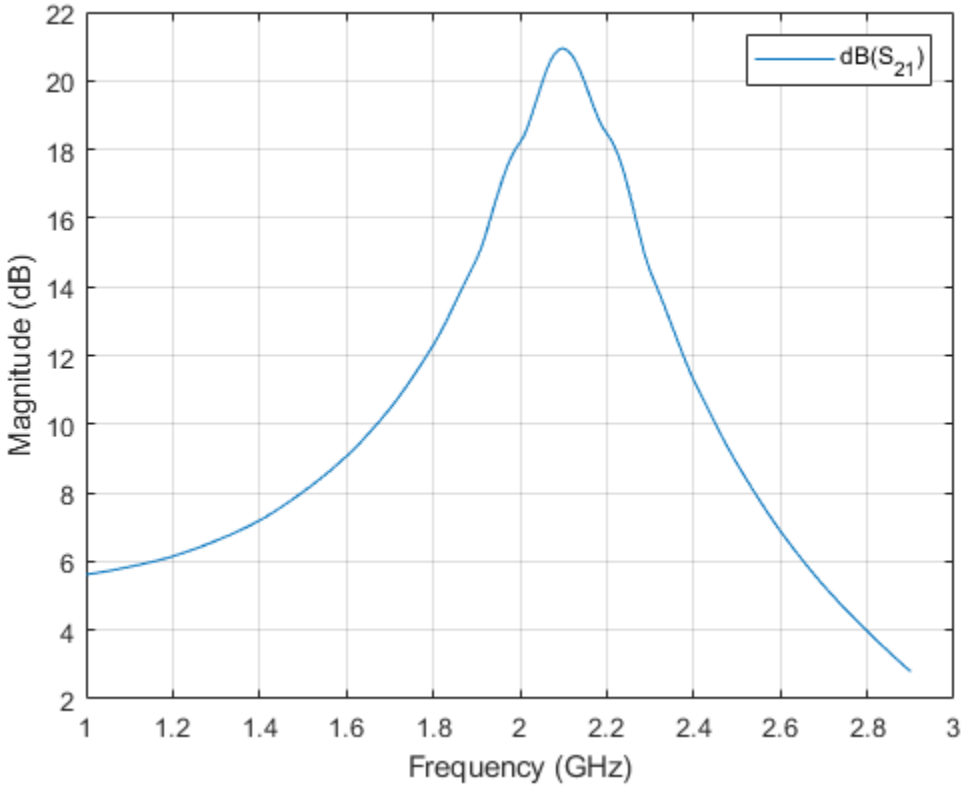
Plot all S-parameters.

```
rfplot(hs)
```



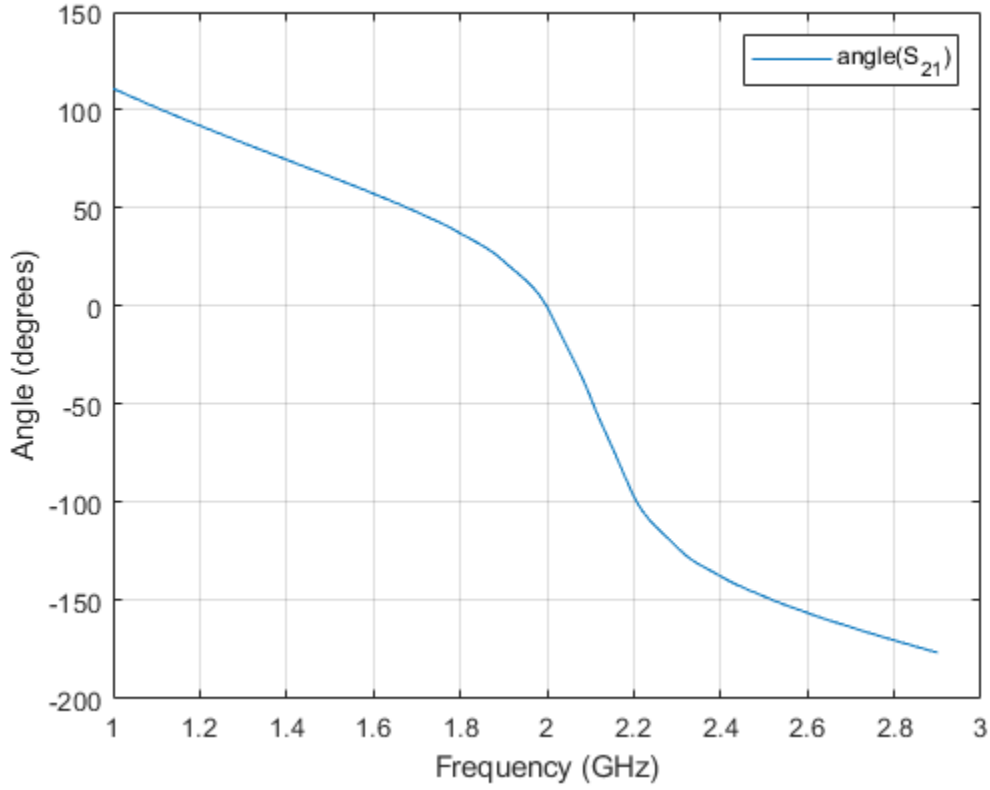
Plot S21.

```
rfplot(hs,2,1)
```



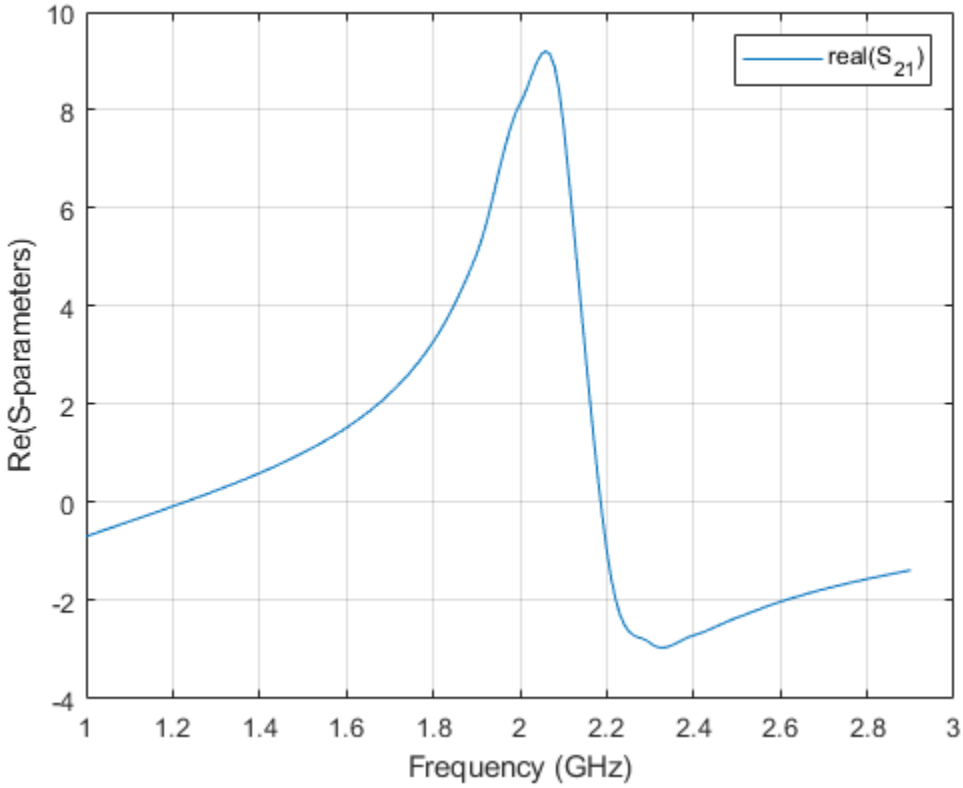
Plot the angle of S21 in degrees.

```
rfplot(hs,2,1,'angle')
```



Plot the real part of S21.

```
rfplot(hs,2,1,'real')
```



Input Arguments

`s_obj` — S-parameters

network parameter object

S-parameters, specified as an RF Toolbox network parameter object. To create this type of object, use the `sparameters` function.

`i` — Row index

positive integer

Row index of data to plot, specified as a positive integer.

`j` — Column index
positive integer

Column index of data to plot, specified as a positive integer.

`lineSpec` — Line specification
character array

Line specification, specified as a text input, that modifies the line types, symbols, and colors of the plot. The function takes text inputs in the same format as `plot` command. For more information on line specification values, see `linespec`.

Example: `'-or'`

`plotflag` — Plot types
`'db'` (default)

Plot types, specified as the following values: `'db'`, `'real'`, `'imag'`, `'abs'`, `'angle'`.

Example: `'angle'`

Output Arguments

`hline` — Line
line handle

Line containing the S-parameter plot, returned as a line handle.

See Also

`sparameters`

Introduced before R2006a

show

Display antenna or array structure; Display shape as filled patch

Syntax

```
show(object)
```

```
show(shape)
```

Description

`show(object)` displays the structure of an antenna or array object.

`show(shape)` displays shape as filled region using patches.

Examples

Display Antenna Structure

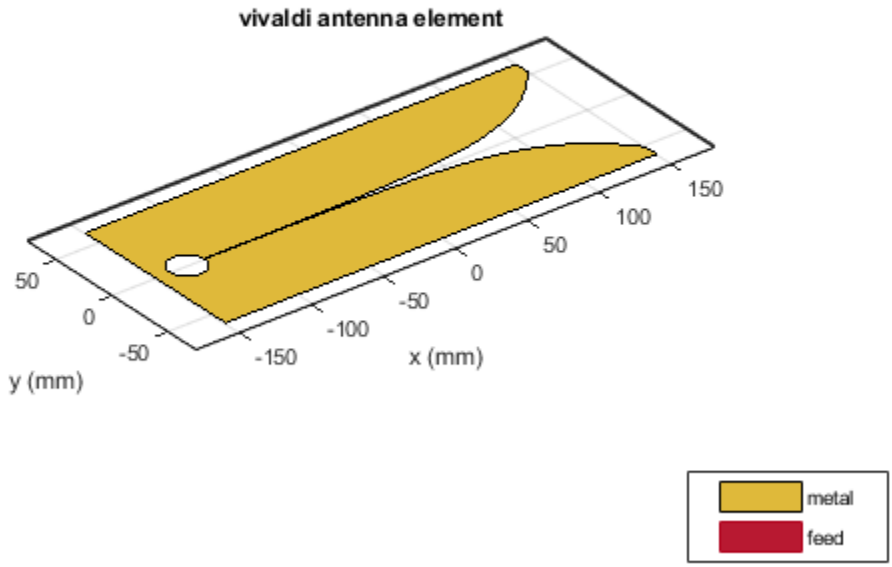
This example shows how to create a vivaldi antenna and display the antenna structure.

```
h = vivaldi
```

```
h =
  vivaldi with properties:

            TaperLength: 0.2430
          ApertureWidth: 0.1050
            OpeningRate: 25
          SlotLineWidth: 5.0000e-04
         CavityDiameter: 0.0240
    CavityToTaperSpacing: 0.0230
       GroundPlaneLength: 0.3000
        GroundPlaneWidth: 0.1250
              FeedOffset: -0.1045
```

```
    Tilt: 0
TiltAxis: [1 0 0]
    Load: [1x1 lumpedElement]
```

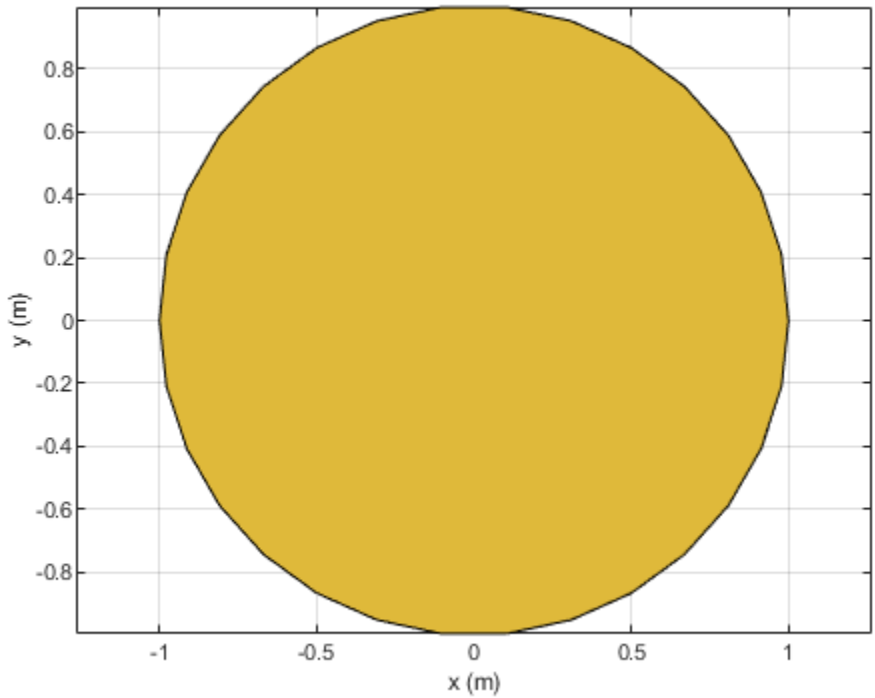
```
show(h)
```



Show Circle Shape

Create a circular shape and visualize the filled regions.

```
c  = antenna.Circle;
show(c);
```



Input Arguments

`object` — Antenna or array object
scalar handle

Antenna or array object, specified as a scalar handle.

`shape` — Shape created using custom elements and shape objects
object handle

Shape created using custom elements and shape objects of Antenna Toolbox, specified as an object handle. You can create the shapes using `antenna.Circle`, `antenna.Polygon`, or `antenna.Rectangle`.

Example: `c = antenna.Rectangle; show(c)`

See Also

`layout` | `mesh` | `plot`

Introduced in R2015a

returnLoss

Return loss of antenna; scan return loss of array

Syntax

```
returnLoss(antenna,frequency,z0)
rl = returnLoss(antenna ,frequency, z0)

returnLoss(array,frequency,elementnumber)
rl = returnLoss(array,frequency,elementnumber)
```

Description

`returnLoss(antenna,frequency,z0)` calculates and plots the return loss of an antenna, over a specified frequency and a given reference impedance, `z0`.

`rl = returnLoss(antenna ,frequency, z0)` returns the return loss of an antenna.

`returnLoss(array,frequency,elementnumber)` calculates and plots the scan return loss of a specified antenna element in an array.

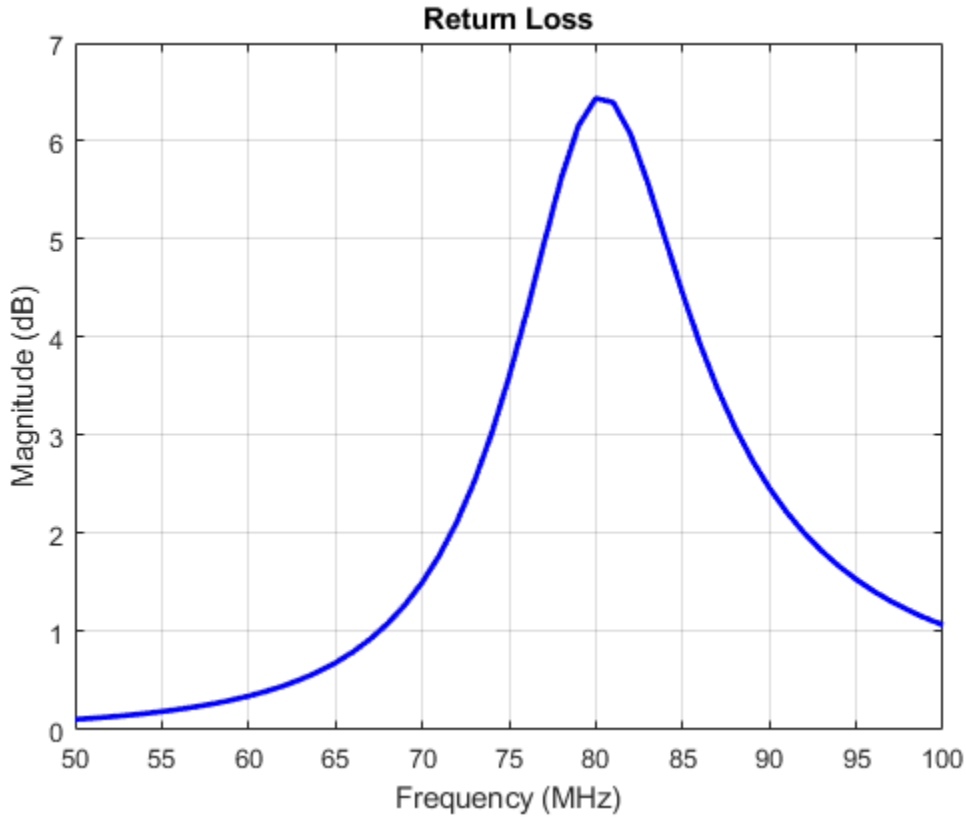
`rl = returnLoss(array,frequency,elementnumber)` returns the scan return loss of a specified antenna element in an array.

Examples

Calculate and Plot Return Loss of Antenna

This example shows how to calculate and plot the return loss of a circular loop antenna over a frequency range of 50MHz-100MHz.

```
h = loopCircular;
returnLoss (h, 50e6:1e6:100e6);
```



Input Arguments

`antenna` — Antenna object
scalar handle

Antenna object, specified as a scalar handle.

`array` — array object
scalar handle

Array object, specified as a scalar handle.

`frequency` — Frequency range used to calculate return loss
vector in Hz

Frequency range used to calculate return loss, specified as a vector in Hz.

Example: 50e6:1e6:100e6

Data Types: `double`

`z0` — Reference impedance
50 (default) | scalar in ohms

Reference impedance, specified as a scalar in ohms.

Example: 40

Data Types: `double`

`elementnumber` — Antenna element number in array
scalar

Antenna element number in array, specified as a scalar.

Example: 1

Data Types: `double`

Output Arguments

`rl` — Return loss of antenna object or scan return loss of array object
vector in dB

Return loss of antenna object or scan return loss of array object, returned as a vector in dB. The return loss is calculated using the formula

$$RL = 20\log 10\left|\frac{(Z - Z_0)}{(Z + Z_0)}\right|$$

where,

- Z = input impedance of antenna or scan impedance of array
- Z_0 = reference impedance

See Also

`EHfields` | `impedance` | `sparameters`

Introduced in R2015a

pattern

Radiation pattern of antenna or array; Embedded pattern of antenna element in array

Syntax

```
pattern(object,frequency)
pattern(object,frequency,azimuth,elevation)
pattern( ___ ,Name,Value)

[fieldval,azimuth,elevation] = pattern(object,frequency)
[pat,azimuth,elevation] = pattern(object,frequency,azimuth,
elevation)
[pat,azimuth,elevation] = pattern( ___ ,Name,Value)
```

Description

`pattern(object,frequency)` plots the 3-D radiation pattern of the antenna or array object over a specified frequency. By default, in Antenna Toolbox, the far-field radius is set to `100λ`.

`pattern(object,frequency,azimuth,elevation)` plots the radiation pattern of the antenna or array object using the specified `azimuth` and `elevation` angles.

`pattern( ___ ,Name,Value)` uses additional options specified by one or more Name, Value pair arguments. You can use any of the input arguments from previous syntaxes.

Use the `'ElementNumber'` and `'Termination'` property to calculate the embedded pattern of the antenna element in an array connected to a voltage source. The voltage source model consists of an ideal voltage source of 1 volt in series with a source impedance. The embedded pattern includes the effect of mutual coupling due to the other antenna elements in the array.

`[fieldval,azimuth,elevation] = pattern(object,frequency)` returns the field value of an antenna or array object over a specified frequency. `azimuth` and `elevation` are the angles at which the `pattern` function calculates the directivity.

`[pat,azimuth,elevation] = pattern(object,frequency,azimuth, elevation)` returns the pattern value, `pat`, value of an antenna or array object at specified frequency. `azimuth` and `elevation` are the angles at which the pattern function calculates the directivity.

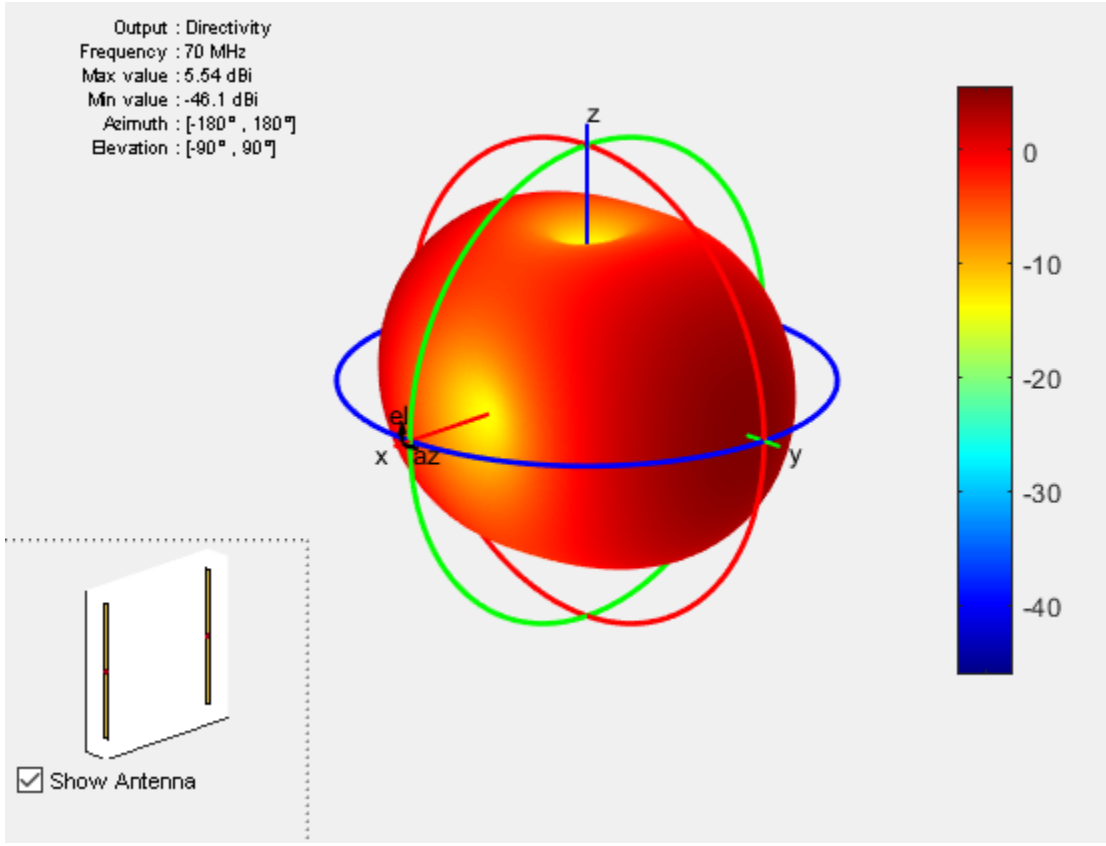
`[pat,azimuth,elevation] = pattern( ___ ,Name,Value)` uses additional options specified by one or more `Name,Value` pair arguments.

Examples

Calculate Radiation Pattern of Array

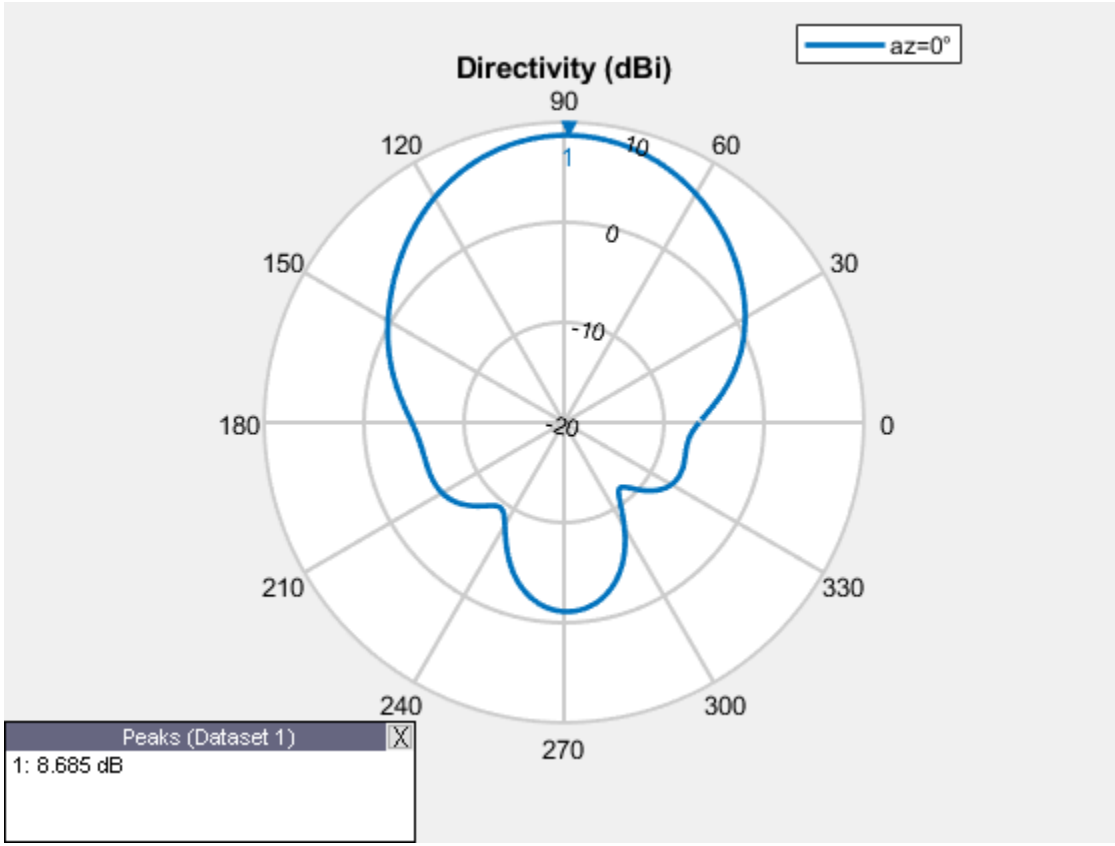
Calculate radiation pattern of default linear array for a frequency of 70 MHZ.

```
l = linearArray;
pattern(l,70e6)
```



Radiation Pattern of Helix in X-Z Plane

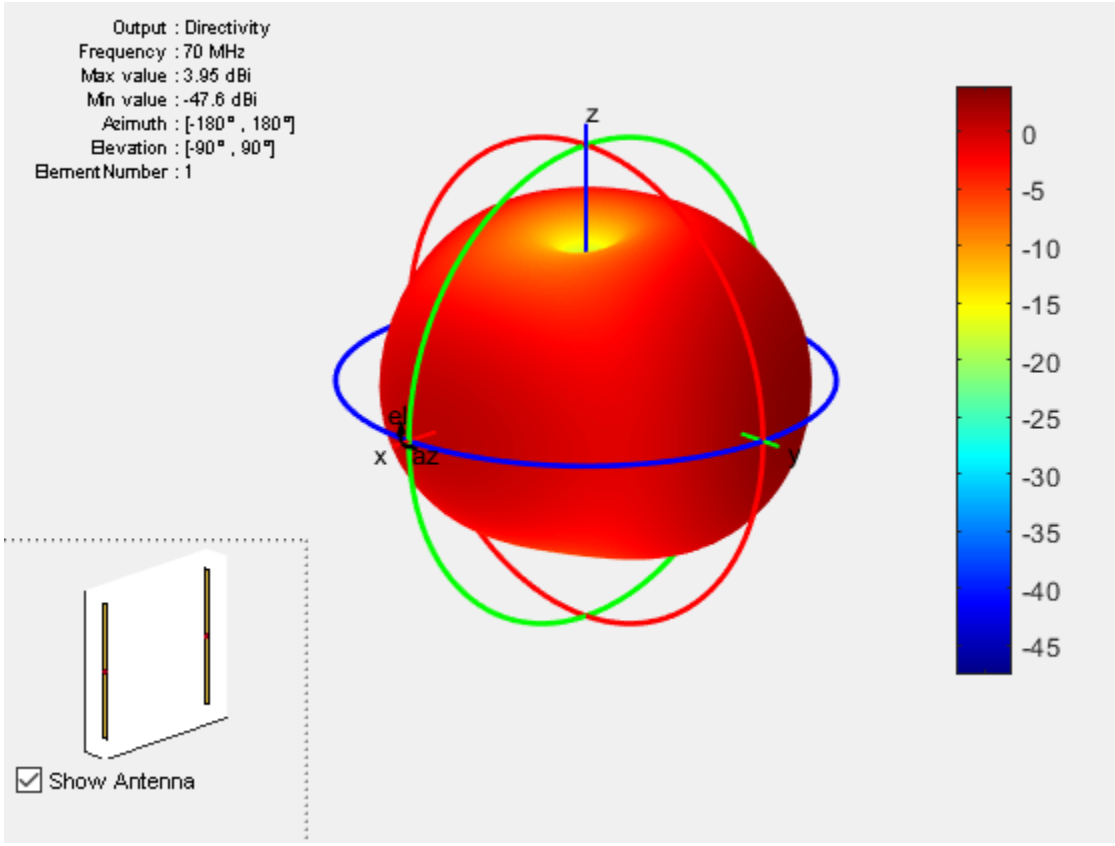
```
h = helix;
pattern (h, 2e9, 0, 1:1:360);
```



Embedded Element Pattern of Linear Array

Calculate the embedded element pattern of a linear array. Excite the first antenna element in the array. Terminate all the other antenna elements using a 50-ohm resistance.

```
l = linearArray;
pattern(l, 70e6,'ElementNumber', 1,'Termination', 50);
```



Directivity Value of Helix Antenna.

Calculate the directivity of a helix antenna.

```
h = helix;
D = pattern(h, 2e9, 0, 1:1:360);
```

Showing the first five directivity values.

```
Dnew = D(1:5)
```

```
Dnew = 5×1

   -6.2750
   -6.0599
   -5.8321
   -5.5934
   -5.3455
```

Input Arguments

`object` — Antenna or array element

object

Antenna or array element, specified as an objects.

`frequency` — Frequency to calculate or plot antenna or array radiation pattern

scalar

Frequency to calculate or plot antenna or array radiation pattern, specified as a scalar in Hz.

Example: 70e6

Data Types: `double`

`azimuth` — Azimuth angles and spacing between angles

`–180:5:180` (default) | vector

Azimuth angles and spacing between the angles to visualize radiation pattern, specified vector in degrees. If the coordinate system is set to uv, then the U values are specified in this parameter. The values of U are between `-1` to `1`.

Example: 90

Data Types: `double`

`elevation` — Elevation angles and spacing between angles

`–90:5:90` (default) | vector

Elevation angles and spacing between the angles to visualize radiation pattern, specified vector in degrees. If the coordinate system is set to uv, then the V values are specified in this parameter. The values of V are between `-1` to `1`.

Example: 0:1:360

Data Types: double

Name-Value Pair Arguments

Specify optional comma-separated pairs of Name,Value pair arguments. Name is the argument name and Value is the corresponding value. Name must appear inside single quotes (''). You can specify several name and value pair arguments in any order as Name1, Value1, ..., NameN, ValueN.

Example: 'CoordinateSystem', 'uv'

CoordinateSystem — Coordinate system to visualize radiation pattern

'polar' (default) | 'rectangular' | 'uv'

Coordinate system to visualize radiation pattern, specified as the comma-separated pair consisting of 'CoordinateSystem' and one of these values: 'polar', 'rectangular', 'uv'.

Example: 'CoordinateSystem', 'polar'

Data Types: char

Type — Quantity to plot

'directivity' | 'gain' | 'efield' | 'power' | 'powerdb'

Quantity to plot, specified as a comma-separated pair consisting of 'Type' and one of these values:

- directivity – Directivity in dBi (lossless antenna or array)
- gain – Gain in dBi (lossy antenna or array)
- efield – Electric field in volt/meter
- power – Power in watts
- powerdb – Power in dB

The default value is 'directivity' for a lossless antenna and 'gain' for a lossy antenna. You cannot plot the 'directivity' of a lossy antenna.

Example: 'Type', 'efield'

Data Types: char

`Normalize` — Normalize field pattern
`true` (default) | `false` | boolean

Normalize field pattern, specified as the comma-separated pair consisting of `'Normalize'` and either `true` or `false`. You cannot use this property if the `'Type'` is set to `'directivity'`.

Example: `'Normalize', false`

Data Types: `logical`

`PlotStyle` — 2-D pattern display style when frequency is vector
`'overlay'` (default) | `'waterfall'`

2-D pattern display style when frequency is vector, specified as the comma-separated pair consisting of `'PlotStyle'` and one of these values:

- `'overlay'` – Overlay frequency data in a 2-D line plot
- `'waterfall'` – Plot frequency data in a waterfall plot

You can use this property when using `pattern` function with no output arguments.

Example: `'PlotStyle', 'waterfall'`

Data Types: `char`

`Polarization` — Field polarization
`'H'` | `'V'` | `'RHCP'` | `'LHCP'`

Field polarization, specified as the comma-separated pair consisting of `'Polarization'` and one of these values:

- `'H'` – Horizontal polarization
- `'V'` – Vertical polarization
- `'RHCP'` – Right-hand circular polarization
- `'LHCP'` – Left-hand circular polarization

By default, you can visualize a combined polarization.

Example: `'Polarization', 'RHCP'`

Data Types: `char`

`ElementNumber` — Antenna element in array
scalar

Antenna element in array, specified as the comma-separated pair consisting of `'ElementNumber'` and scalar This antenna element is connected to the voltage source.

Example: `'ElementNumber',1`

Data Types: `double`

Note Use this property to calculate the embedded pattern of an array.

`Termination` — Impedance value for array element termination
50 (default) | scalar

Impedance value for array element termination, specified as the comma-separated pair consisting of `'Termination'` and scalar. The impedance value terminates other antenna elements of an array while calculating the embedded pattern of the antenna connected to the voltage source.

Example: `'Termination',40`

Data Types: `double`

Note Use this property to calculate the embedded pattern of an array.

Output Arguments

`pat` — Radiation pattern of antenna or array or embedded pattern of array
matrix

Radiation pattern of antenna or array or embedded pattern of array, returned as a matrix of number f elevation values by number of azimuth values. The pattern is one of the following:

- `directivity` – Directivity in dBi (lossless antenna or array)
- `gain` – Gain in dBi (lossy antenna or array)
- `efield` – Electric field in volt/meter

- `power` – Power in watts
- `powerdb` – Power in dB

Matrix size is number of elevation values multiplied by number of azimuth values.

`azimuth` — Azimuth angles to calculate radiation pattern
vector in degrees

Azimuth angles to calculate radiation pattern, returned as a vector in degrees.

`elevation` — Elevation angles to calculate radiation pattern
vector in degrees

Elevation angles to calculate radiation pattern, returned as a vector in degrees.

References

[1] Makarov, Sergey N. *Antenna and Em Modeling in MATLAB*. Chapter3, Sec 3.4 3.8. Wiley Inter-Science.

[2] Balanis, C.A. *Antenna Theory, Analysis and Design*, Chapter 2, sec 2.3-2.6, Wiley.

See Also

`EHfields` | `current`

Introduced in R2015a

patternAzimuth

Azimuth pattern of antenna or array

Syntax

```
patternAzimuth(object,frequency,elevation)
patternAzimuth(object,frequency,elevation,Name,Value)

directivity = patternAzimuth(object,frequency,elevation)
directivity = patternAzimuth(object,frequency,elevation,'Azimuth')
```

Description

`patternAzimuth(object,frequency,elevation)` plots the 2-D radiation pattern of the antenna or array object over a specified frequency. Elevation values defaults to zero if not specified.

`patternAzimuth(object,frequency,elevation,Name,Value)` uses additional options specified by one or more `Name,Value` pair arguments.

`directivity = patternAzimuth(object,frequency,elevation)` returns the directivity of the antenna or array object over a specified frequency. Elevation values defaults to zero if not specified.

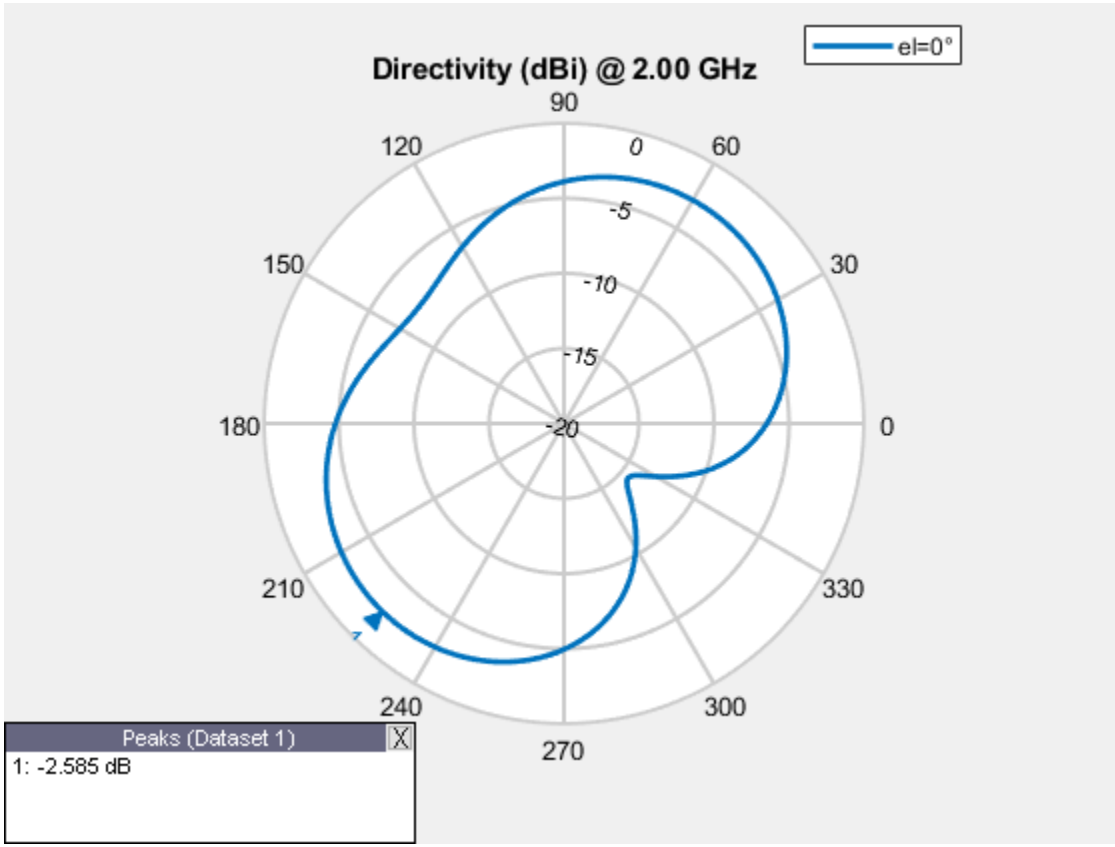
`directivity = patternAzimuth(object,frequency,elevation,'Azimuth')` uses additional options specified by one or more `Name,Value` pair arguments.

Examples

Azimuth Radiation Pattern of Helix Antenna

Calculate and plot the azimuth radiation pattern of the helix antenna at 2 GHz.

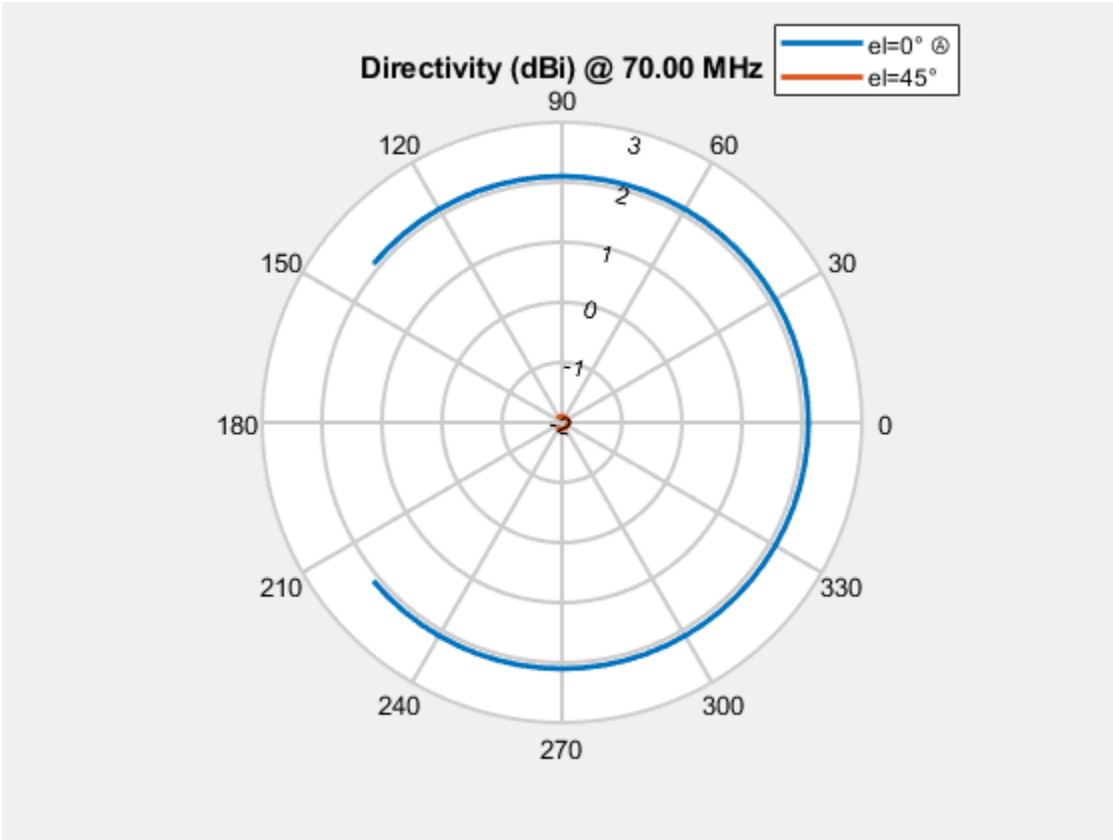
```
h = helix;
patternAzimuth(h,2e9);
```



Azimuth Radiation Pattern of Dipole Antenna

Calculate and plot the azimuth radiation pattern of the dipole antenna at 70 MHz at elevation values of 0 and 45.

```
d = dipole;
patternAzimuth(d,70e6,[0 45],'Azimuth',-140:5:140);
```



Input Arguments

`object` — antenna or array object
scalar handle

Antenna or array object, specified as a scalar handle.

`frequency` — Frequency used to calculate charge distribution
scalar in Hz

Frequency used to calculate charge distribution, specified as a scalar in Hz.

Example: 70e6

Data Types: `double`

`elevation` — Elevation angle values
vector in degrees

Elevation angle values, specified as a vector in degrees.

Example: `[0 45]`

Data Types: `double`

`'Azimuth'` — Azimuth angles of antenna
–180:1:180 (default) | vector in degrees

Azimuth angles of antenna, specified as the comma-separated pair consisting of `'Azimuth'` and a vector in degrees.

Example: `'Azimuth',2:2:340`

Data Types: `double`

Output Arguments

`directivity` — Antenna or array directivity
matrix in `dBi`

Antenna or array directivity, returned as a matrix in `dBi`. The matrix size is the product of number of elevation values and number of azimuth values.

See Also

`pattern` | `patternElevation` | `polarpattern`

Introduced in R2015a

patternMultiply

Radiation pattern of array using pattern multiplication

Syntax

```
patternMultiply(array,frequency)
patternMultiply(array,frequency,azimuth)
patternMultiply(array,frequency,azimuth, elevation)
patternMultiply( ___ ,Name,Value)

[fieldval,azimuth,elevation] = patternMultiply(array,frequency)
[fieldval,azimuth,elevation] = patternMultiply(array,frequency,
azimuth)
[fieldval,azimuth,elevation] = patternMultiply(array,frequency,
azimuth,elevation)
[fieldval,azimuth,elevation] = patternMultiply( ___ ,Name,Value)
```

Description

`patternMultiply(array,frequency)` plots the 3-D radiation pattern of the array object over a specified frequency. `patternMultiply` calculates the full array pattern without taking the effect of mutual coupling between the different array elements.

`patternMultiply(array,frequency,azimuth)` plots the radiation pattern of the array object for the given azimuth angles. Elevation angles retain default values.

`patternMultiply(array,frequency,azimuth, elevation)` plots the radiation pattern of the array object for the given azimuth and elevation angles.

`patternMultiply( ___ ,Name,Value)` uses additional options specified by one or more `Name,Value` pair arguments. Specify name-value pair arguments after all other input arguments.

`[fieldval,azimuth,elevation] = patternMultiply(array,frequency)` returns the field value such as the directivity of the lossless array in dBi or gain of the

lossy array in dBi at the specified frequency. The size of the field value matrix is number of elevation values x number of azimuth values.

`[fieldval,azimuth,elevation] = patternMultiply(array,frequency, azimuth)` returns the field value at the specified azimuth angles. Elevation angles retain default values.

`[fieldval,azimuth,elevation] = patternMultiply(array,frequency, azimuth,elevation)` returns the field value at the specified azimuth angles, and elevation angles.

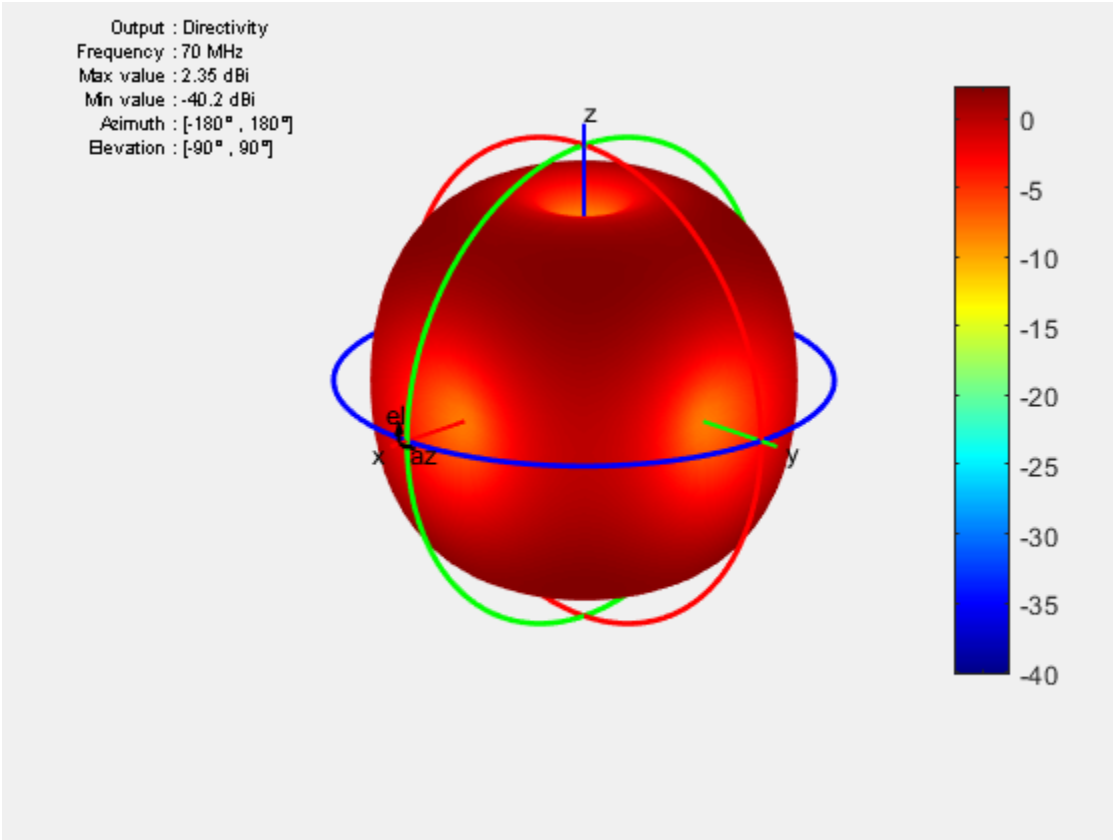
`[fieldval,azimuth,elevation] = patternMultiply( ___ ,Name,Value)` returns the field value using additional options specified by one or more `Name,Value` pair arguments. Specify name-value pair arguments after all other input arguments.

Examples

Radiation Pattern of Rectangular Array

Plot the radiation pattern of a default rectangular array at 70 MHz. Pattern multiplication does not take into consideration the effect of mutual coupling in array elements.

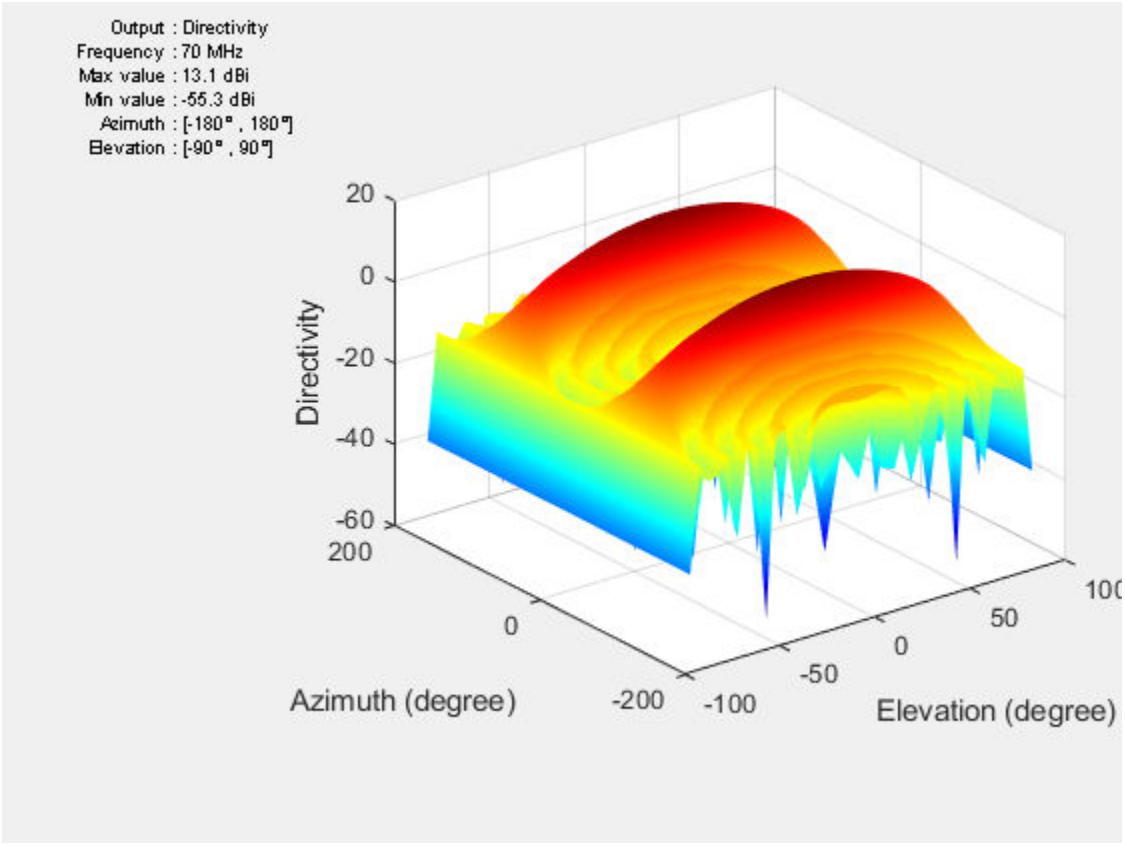
```
h = rectangularArray;
patternMultiply(h,70e6);
```



Radiation Pattern of Linear Array in Rectangular Coordinates

Plot the radiation pattern of a 10-element linear array at 70 MHz. Visualize the pattern using the rectangular coordinate system.

```
l = linearArray('NumElements',10);
patternMultiply(l,70e6,'CoordinateSystem','rectangular');
```



Input Arguments

`array` — Input antenna array

object handle

Array object, specified as an object handle.

Example: `r = rectangularArray; patternMultiply(r,70e6)`. Plot the pattern of a rectangular array.

`frequency` — Frequency used to calculate array pattern

scalar in Hz

Frequency used to calculate array pattern, specified as a scalar in Hz.

Example: `70e6`

Data Types: `double`

`azimuth` — Azimuth angle of antenna
`-180:5:180` (default) | vector in degrees

Azimuth angle of the antenna, specified as a vector in degrees.

Example: `-90:5:90`

Data Types: `double`

`elevation` — Elevation angle of antenna
`-90:5:90` (default) | vector in degrees

Elevation angle of the antenna, specified as a vector in degrees.

Example: `0:1:360`

Data Types: `double`

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` pair arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside single quotes (`''`). You can specify several name and value pair arguments in any order as `Name1, Value1, ..., NameN, ValueN`.

Example: `'CoordinateSystem', rectangular`

`CoordinateSystem` — Coordinate system of radiation pattern
`'polar'` (default) | `'rectangular'` | `'uv'`

Coordinate system of radiation pattern, specified as the comma-separated pair consisting of `'CoordinateSystem'` and one of these values: `'polar'`, `'rectangular'`, `'uv'`.

Example: `'CoordinateSystem', 'polar'`

Data Types: `char`

Type — Value to plot
`'directivity'` (default) | `'gain'` | `'efield'` | `'power'` | `'powerdb'`

Value to plot, specified as a comma-separated pair consisting of `'Type'` and one of these values:

- `'directivity'` – Radiation intensity in a given direction of antenna in dB
- `'gain'` – Radiation intensity in a given direction of antenna, when the antenna has a lossy substrate in dB
- `'efield'` – Electric field of antenna in volt/meter
- `'power'` – Antenna power in watts
- `'powerdb'` – Antenna power in dB

Example: `'Type'`, `'efield'`

Data Types: `char`

`Normalize` — Normalize field pattern
`true` (default) | `false` | boolean

Normalize field pattern, specified as the comma-separated pair consisting of `'Normalize'` and either `true` or `false`. For directivity patterns, this property is not applicable.

Example: `'Normalize'`, `false`

Data Types: `double`

`Polarization` — Field polarization
`'H'` | `'V'` | `'RHCP'` | `'LHCP'`

Field polarization, specified as the comma-separated pair consisting of `'Polarization'` and one of these values:

- `'H'` – Horizontal polarization
- `'V'` – Vertical polarization
- `'RHCP'` – Right-hand circular polarization
- `'LHCP'` – Left-hand circular polarization

By default, you can visualize a combined polarization.

Example: `'Polarization'`, `'RHCP'`

Data Types: `char`

Output Arguments

`fieldval` — Array directivity or gain
matrix in `dBi`

Array directivity or gain, returned as a matrix in `dBi`. The matrix size is the product of the number of elevation values and azimuth values.

`azimuth` — Azimuth angles
vector in degrees

Azimuth angle used to calculate field values, returned as a vector in degrees.

`elevation` — Elevation angles
vector in degrees

Elevation angles used to calculate field values, returned as a vector in degrees.

See Also

`pattern` | `patternElevation`

Introduced in R2017a

patternElevation

Elevation pattern of antenna or array

Syntax

```
patternElevation(object,frequency,azimuth)
patternElevation(object,frequency,azimuth,Name,Value)

directivity = patternElevation(object,frequency,azimuth)
directivity = patternElevation(object,frequency,azimuth,'Elevation')
```

Description

`patternElevation(object,frequency,azimuth)` plots the 2-D radiation pattern of the antenna or array object over a specified frequency. Azimuth values defaults to zero if not specified.

`patternElevation(object,frequency,azimuth,Name,Value)` uses additional options specified by one or more `Name, Value` pair arguments.

`directivity = patternElevation(object,frequency,azimuth)` returns the directivity of the antenna or array object at specified frequency. Azimuth values defaults to zero if not specified.

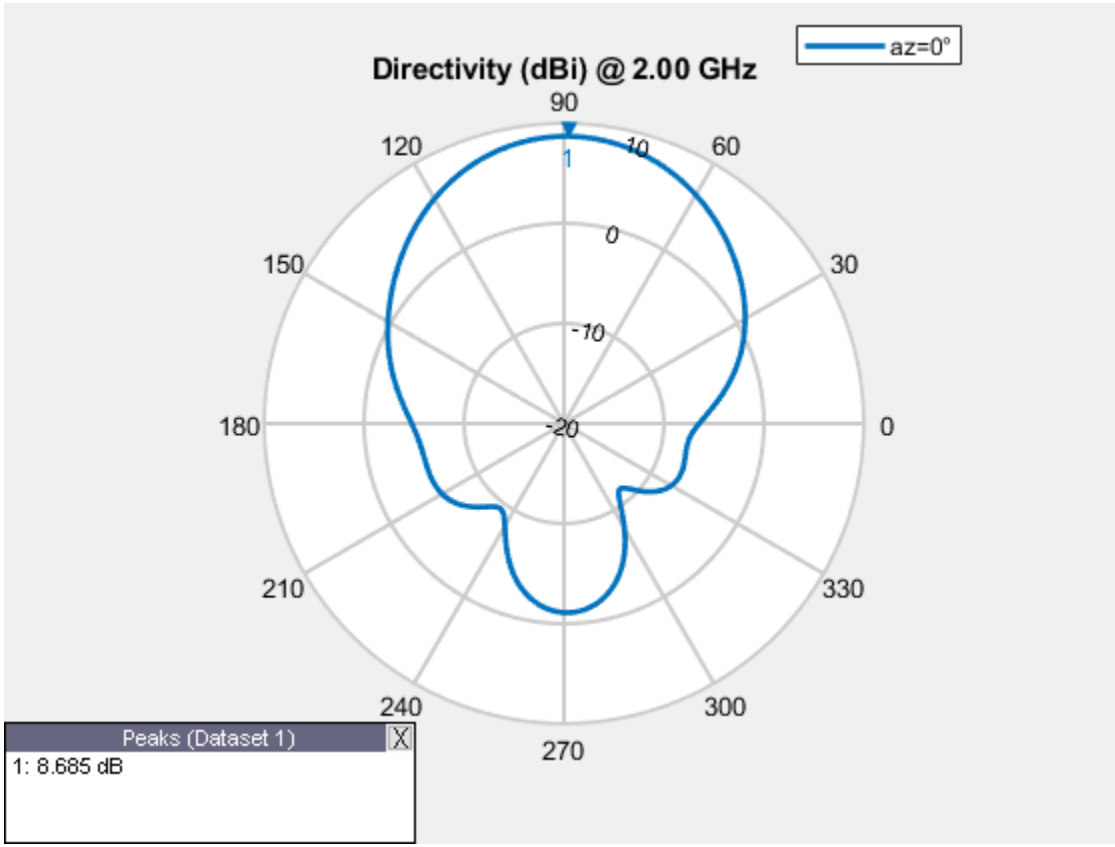
`directivity = patternElevation(object,frequency,azimuth,'Elevation')` uses additional options specified by one or more `Name, Value` pair arguments.

Examples

Elevation Radiation Pattern of Helix

Calculate and plot the elevation pattern of the helix antenna at 2 GHz.

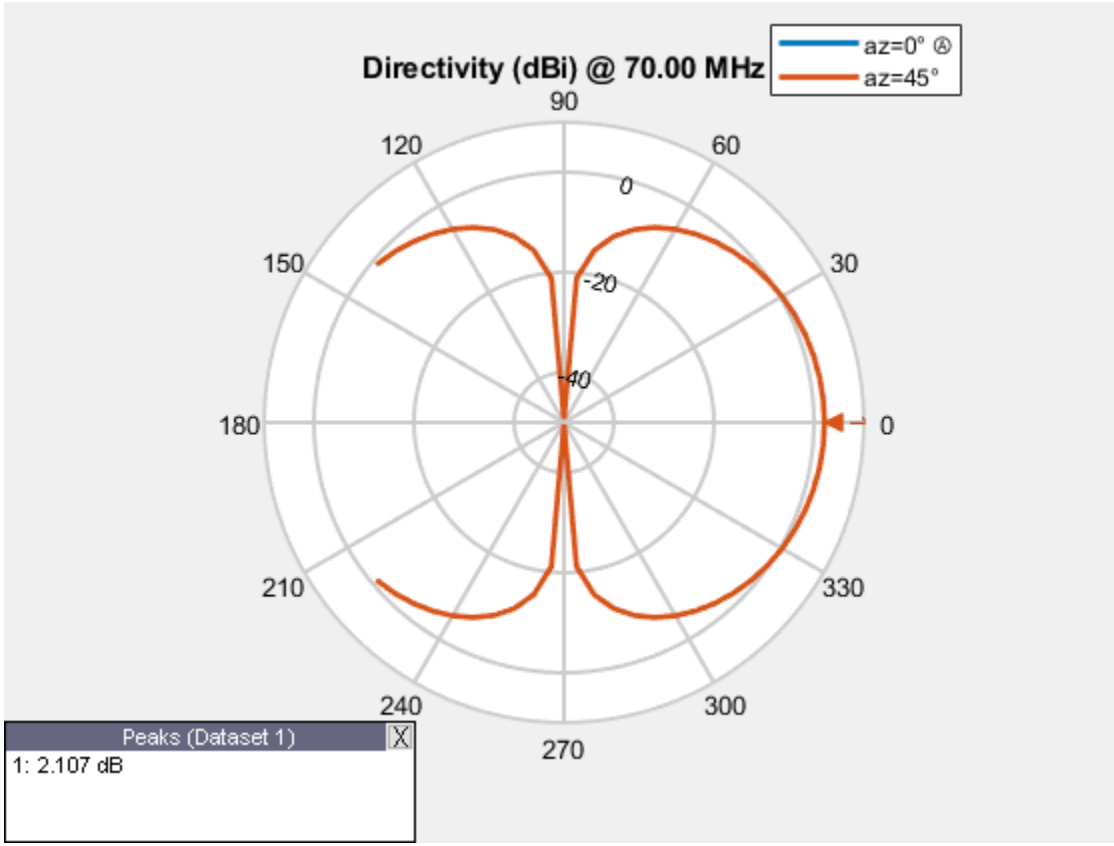
```
h = helix;
patternElevation (h, 2e9);
```



Elevation Radiation Pattern of Dipole Antenna

Calculate and plot the elevation radiation pattern of the dipole antenna at 70 MHz at elevation values of 0 and 45.

```
d = dipole;
patternElevation(d,70e6,[0 45],'Elevation',-140:5:140);
```



Input Arguments

`object` — Antenna or array object
scalar handle

Antenna or array object, specified as a scalar handle.

`frequency` — Frequency used to calculate charge distribution
scalar in Hz

Frequency used to calculate charge distribution, specified as a scalar in Hz.

Example: 70e6

Data Types: `double`

`azimuth` — Azimuth angle values
vector in degrees

Azimuth angle values, specified as a vector in degrees.

Example: `[0 45]`

Data Types: `double`

`'Elevation'` — Elevation angles of antenna
–90:1:90 (default) | vector in degrees

Elevation angles of antenna, specified the comma-separated pair consisting of `'Elevation'` and a vector in degrees.

Example: `'Elevation'`, 0:1:360

Data Types: `double`

Output Arguments

`directivity` — Antenna or array directivity
matrix in `dBi`

Antenna or array directivity, returned as a matrix in `dBi`. The matrix size is the product of number of elevation values and number of azimuth values.

See Also

`pattern` | `patternAzimuth` | `polarpattern`

Introduced in R2015a

current

Current distribution on metal or dielectric antenna or array surface

Syntax

```
current(object,frequency)

i = current(object,frequency)

current(object,frequency,'dielectric')
i = current(object,frequency,'dielectric')
```

Description

`current(object,frequency)` calculates and plots the absolute value of the current on the surface of an antenna or array object, at a specified frequency.

`i = current(object,frequency)` returns the *x*, *y*, *z* components of the current on the surface of an antenna or array object, at a specified frequency.

`current(object,frequency,'dielectric')` calculates and plots the absolute value of current at a specified frequency value on the dielectric face of the antenna or array.

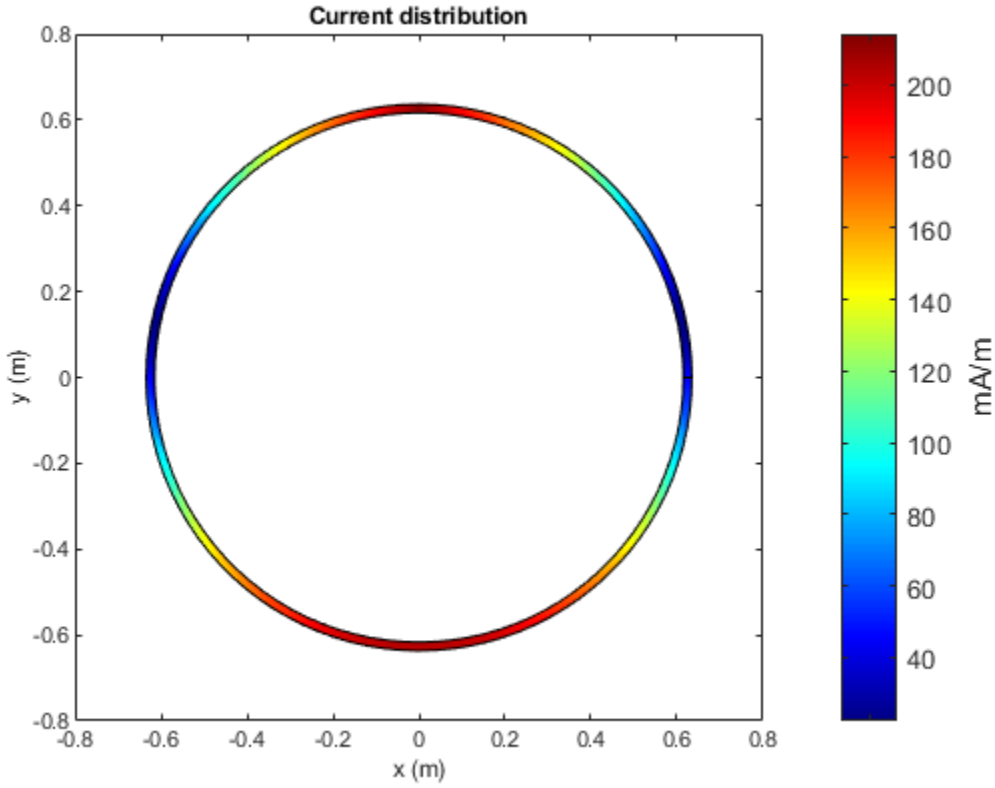
`i = current(object,frequency,'dielectric')` returns the *x*, *y*, *z* components of the current on the dielectric surface of an antenna or array object, at a specified frequency.

Examples

Calculate and Plot Current Distribution on Antenna Surface

Calculate and plot the current distribution for a circular loop antenna at 70MHz frequency.

```
h = loopCircular;
current(h,70e6);
```



Calculate Current Distribution of Array

Calculate the current distribution of a default rectangular array at 70MHz frequency.

```
h = rectangularArray;
i = current(h,70e6)
```

```
i = 3×160 complex
```

```
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 -
   0.0007 + 0.0017i  -0.0007 - 0.0018i   0.0005 + 0.0015i  -0.0005 - 0.0015i  -0.0000 -
   0.0605 + 0.1151i   0.0576 + 0.1077i   0.0618 + 0.1185i   0.0638 + 0.1248i   0.0659 -
```

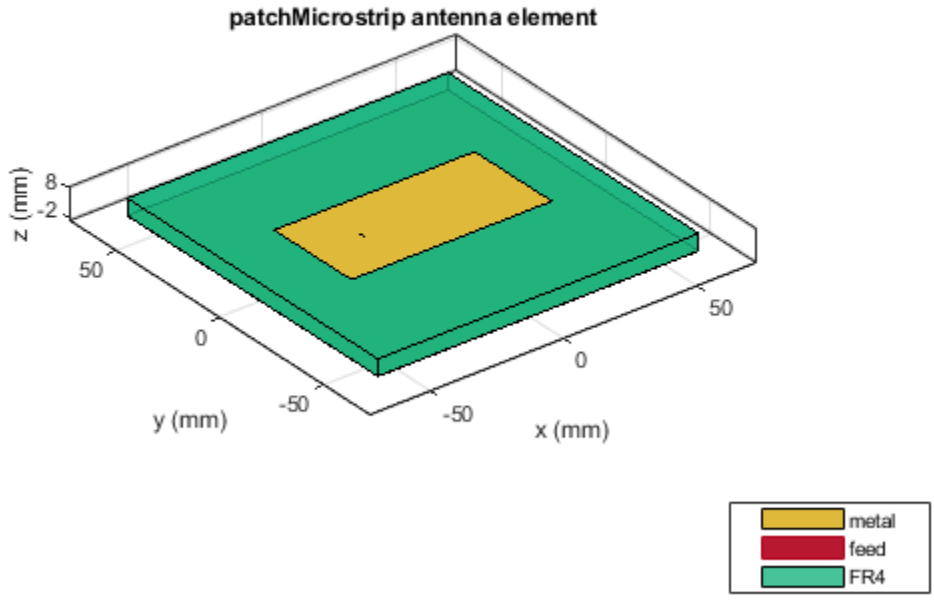
Current Distribution On Microstrip Patch Antenna

Create a microstrip patch antenna using **'FR4'** as the dielectric substrate.

```
d = dielectric('FR4');
pm = patchMicrostrip('Length',75e-3, 'Width',37e-3,                  ...
        'GroundPlaneLength',120e-3, 'GroundPlaneWidth',120e-3, ...
        'Substrate',d)
```

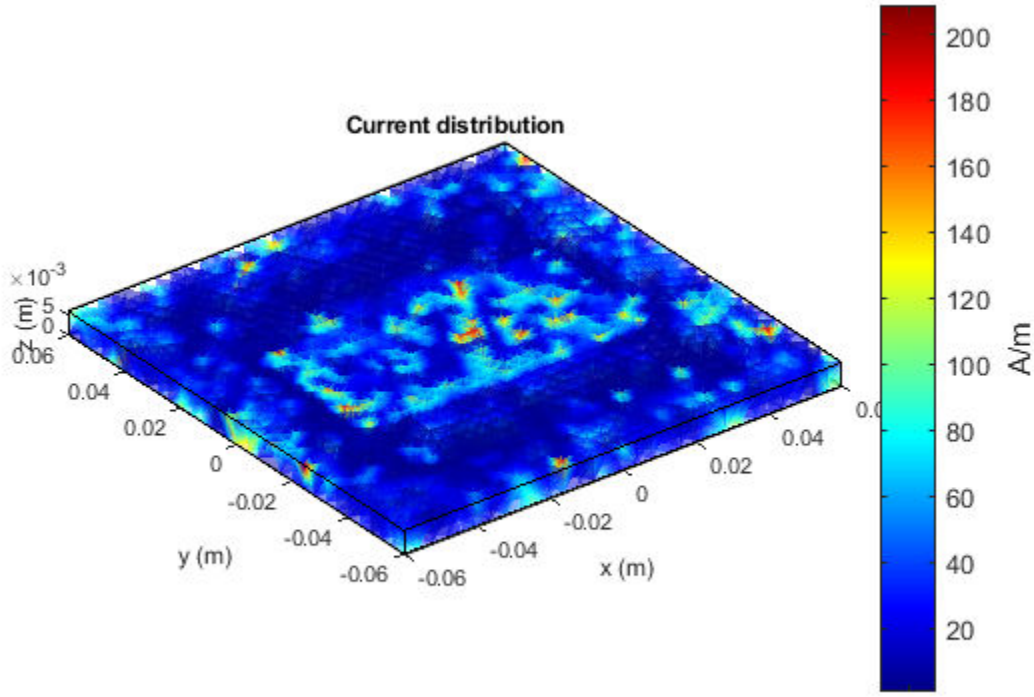
```
pm =
  patchMicrostrip with properties:

              Length: 0.0750
               Width: 0.0370
              Height: 0.0060
           Substrate: [1x1 dielectric]
    GroundPlaneLength: 0.1200
     GroundPlaneWidth: 0.1200
    PatchCenterOffset: [0 0]
          FeedOffset: [-0.0187 0]
                Tilt: 0
            TiltAxis: [1 0 0]
                Load: [1x1 lumpedElement]
```

```
show(pm)
```



Plot the current distribution on the antenna at a frequency of 1.67 GHz.

```
figure
current(pm,1.67e9,'dielectric')
```



Input Arguments

`object` — Antenna or array object

scalar handle

Antenna or array object, specified as a scalar handle.

`frequency` — Frequency used to calculate current distribution

scalar in Hz

Frequency to calculate current distribution, specified as a scalar in Hz.

Example: 70e6

Data Types: `double`

Output Arguments

`i` — *x*, *y*, *z* components of current distribution

3-by-*n* complex matrix in A/m

x, *y*, *z* components of current distribution, returned as a 3-by-*n* complex matrix in A/m. The value of the current is calculated on every triangle mesh or every dielectric tetrahedron face on the surface of an antenna or array.

See Also

`axialRatio` | `charge`

Introduced in R2015a

charge

Charge distribution on metal or dielectric antenna or array surface

Syntax

```
charge(object,frequency)

c = charge(object,frequency)

charge(object,frequency,'dielectric')
c = charge(object,frequency,'dielectric')
```

Description

`charge(object,frequency)` calculates and plots the absolute value of the charge on the surface of an antenna or array object surface at a specified frequency.

`c = charge(object,frequency)` returns a vector of charges in C/m on the surface of an antenna or array object, at a specified frequency.

`charge(object,frequency,'dielectric')` calculates and plots the absolute value of charge at a specified frequency value on the dielectric face of the antenna or array.

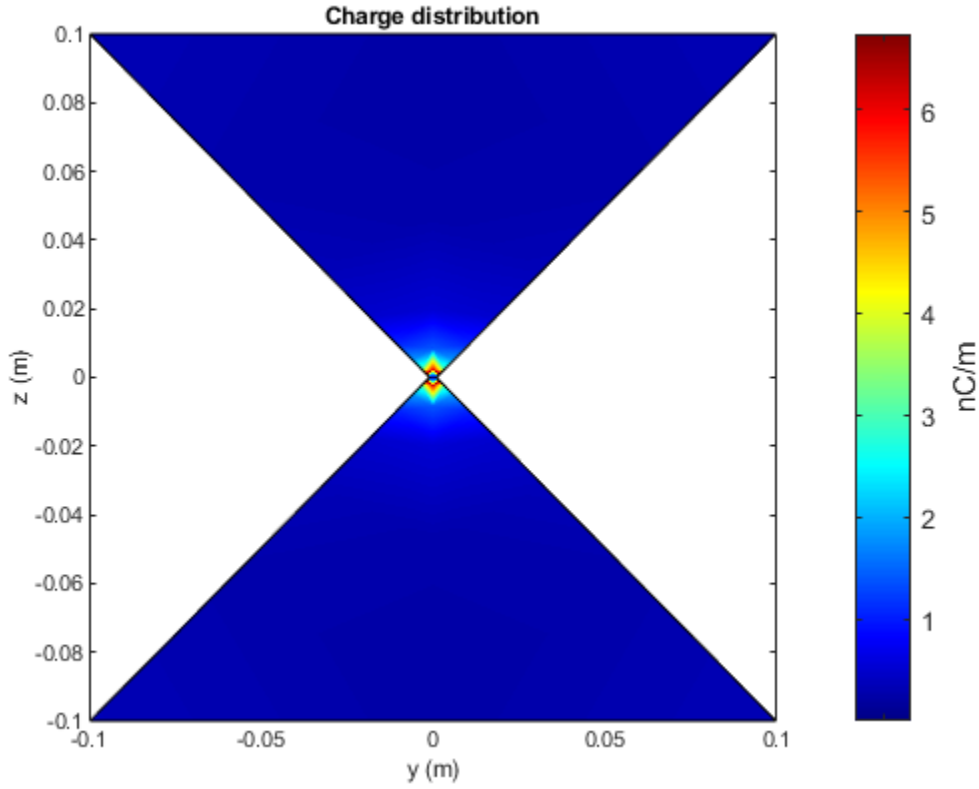
`c = charge(object,frequency,'dielectric')` returns the *x*, *y*, *z* components of the charge on the dielectric surface of an antenna or array object, at a specified frequency.

Examples

Calculate and Plot Charge Distribution on Antenna Surface

Calculate and plot the charge distribution on a bowtieTriangular antenna at 70MHz frequency.

```
h = bowtieTriangular;
charge (h, 70e6);
```



Calculate Charge Distribution of Array

Calculate charge distribution of linear array at 70 MHz frequency.

```
h = linearArray;
h.NumElements = 4;
C = charge(h,70e6);
```

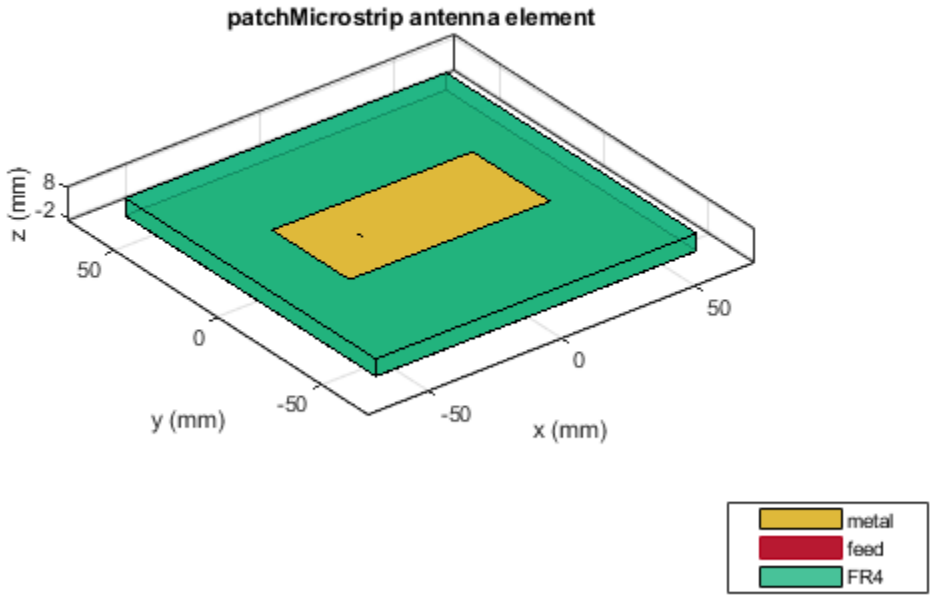
Charge Distribution On Microstrip Patch Antenna

Create a microstrip patch antenna using **'FR4'** as the dielectric substrate.

```
d = dielectric('FR4');
pm = patchMicrostrip('Length',75e-3, 'Width',37e-3,                  ...
        'GroundPlaneLength',120e-3, 'GroundPlaneWidth',120e-3, ...
        'Substrate',d)
```

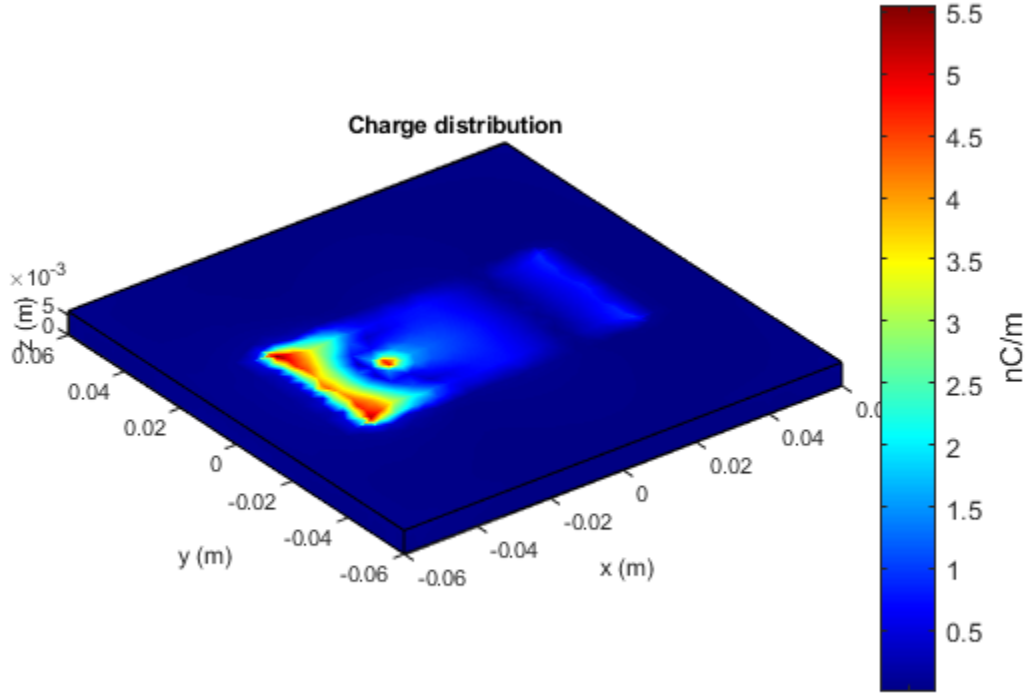
```
pm =
  patchMicrostrip with properties:

              Length: 0.0750
               Width: 0.0370
              Height: 0.0060
           Substrate: [1x1 dielectric]
    GroundPlaneLength: 0.1200
     GroundPlaneWidth: 0.1200
    PatchCenterOffset: [0 0]
          FeedOffset: [-0.0187 0]
                Tilt: 0
            TiltAxis: [1 0 0]
                Load: [1x1 lumpedElement]
```

```
show(pm)
```



Plot the charge distribution on the antenna at a frequency of 1.67 GHz.

```
figure
charge(pm,1.67e9,'dielectric')
```



Input Arguments

`object` — Antenna or array object

scalar handle

Antenna or array object, specified as a scalar handle.

`frequency` — Frequency used to calculate charge distribution

scalar in Hz

Frequency used to calculate charge distribution, specified as a scalar in Hz.

Example: 70e6

Data Types: `double`

Output Arguments

c — Complex charges

1x*n* vector in C/m

Complex charges, returned as a 1x*n* vector in C/m. This value is calculated on every triangle mesh or every dielectric tetrahedron face on the surface of an antenna or array. of antenna or array.

See Also

`EHfields` | `current`

Introduced in R2015a

design

Design prototype antenna or arrays for resonance at specified frequency

Syntax

```
hant = design(antenna,frequency)

harray = design(array,frequency)
harray = design(array,frequency,element)
```

Description

`hant = design(antenna,frequency)` designs any antenna object from the antenna library to resonate at the specified frequency.

`harray = design(array,frequency)` designs an array of dipoles for operation at a specified `frequency`. The elements are separated by half-wavelength.

`harray = design(array,frequency,element)` designs an array of elements for operation at a specified `frequency`. The elements are separated by half-wavelength, if possible. If you cannot achieve half-wavelength spacing, the element size is used to calculate inter-element separation.

Examples

Prototype Antenna Design

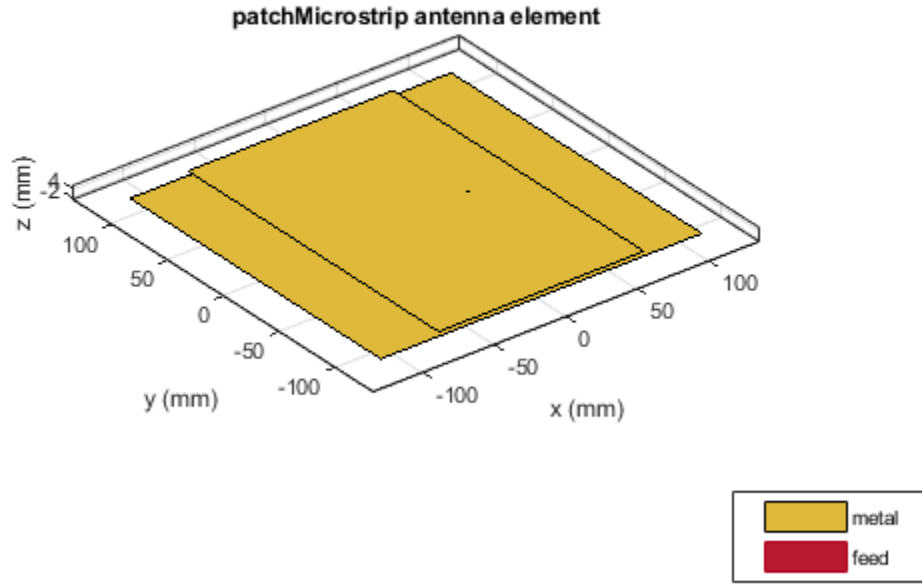
Design a prototype microstrip patch antenna that resonates at a frequency of 1 GHz.

```
p = design(patchMicrostrip,1e9)
```

```
p =
  patchMicrostrip with properties:
```

```
            Length: 0.1439
             Width: 0.2248
            Height: 0.0030
         Substrate: [1x1 dielectric]
GroundPlaneLength: 0.2248
 GroundPlaneWidth: 0.2248
PatchCenterOffset: [0 0]
       FeedOffset: [0.0360 0]
             Tilt: 0
          TiltAxis: [1 0 0]
             Load: [1x1 lumpedElement]
```

```
show(p)
```



Calculate the impedance of the above antenna at the same frequency.

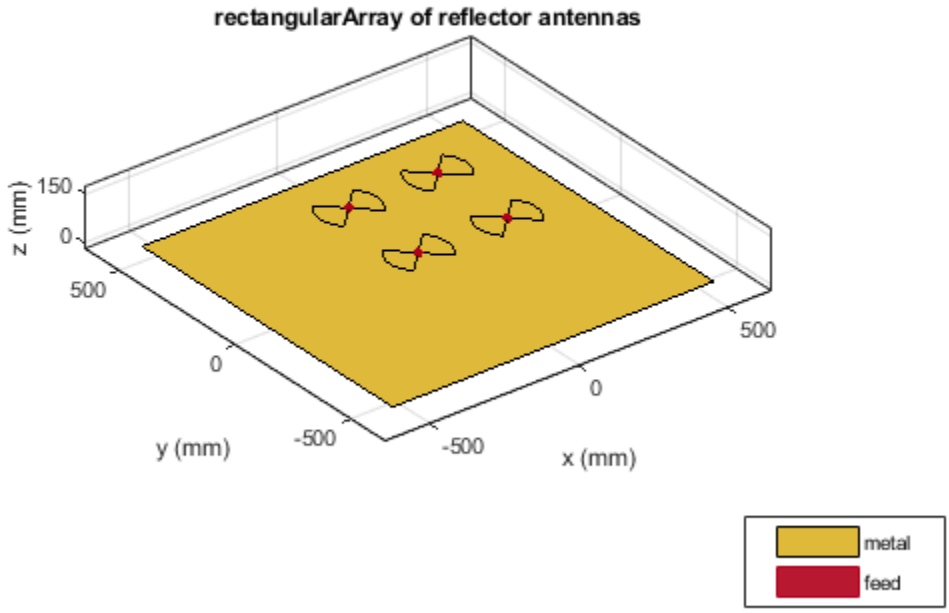
```
Z = impedance(p,1e9)
```

```
Z = 49.5447 + 0.7888i
```

Rectangular Array of Reflector Backed Rounded Bowtie Antennas

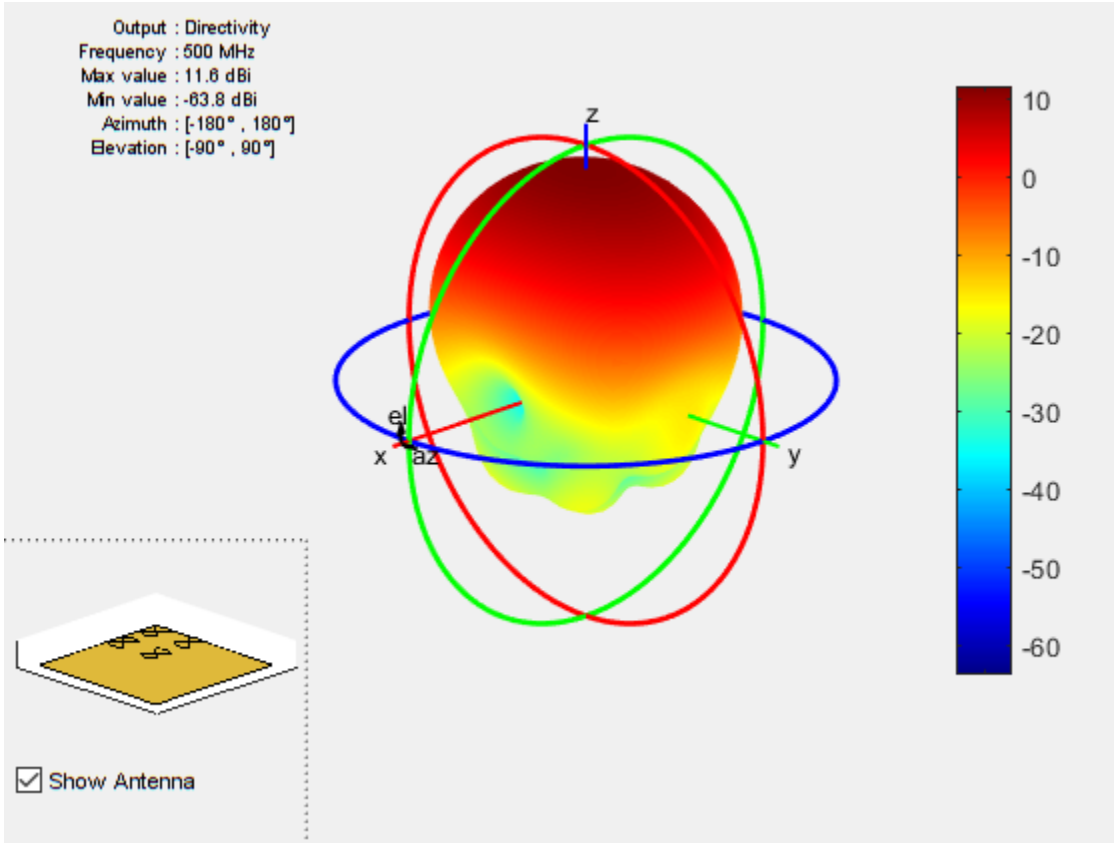
Design a rectangular array of reflector backed rounded bowtie antennas to operate at 500 MHz.

```
b = bowtieRounded('Tilt',90,'TiltAxis',[0 1 0]);
r = reflector('Exciter',b);
ra = design(rectangularArray,500e6,r);
show(ra)
```



Plot the radiation pattern of the rectangular array at 500 MHz.

```
pattern(ra,500e6)
```



Input Arguments

`antenna` — Antenna object

scalar handle

Antenna object from antenna library, specified as a scalar handle.

Example: `dipole`

`array` — Array object

scalar handle

Array object from antenna library, specified as a scalar handle.

Example: `dipole`

`frequency` — Resonant frequency of antenna
real positive scalar

Resonant frequency of the antenna, specified as a real positive scalar.

Example: `55e6`

Data Types: `double`

`element` — Antenna object in array
scalar handle

Antenna object from the antenna library used in the array, specified as a scalar handle.

Example: `r = reflector;ra = design(rectangularArray,500e6,r);` Designs a rectangular array of reflectors operating at a frequency of 500 MHz.

Output Arguments

`hant` — Antenna object operating at specified reference frequency
scalar handle

Antenna object operating at the specified reference frequency, returned as a scalar handle.

`harray` — Array object operating at specified reference frequency
scalar handle

Array object operating at the specified reference frequency, returned as a scalar handle.

See Also

`show`

Introduced in R2016b

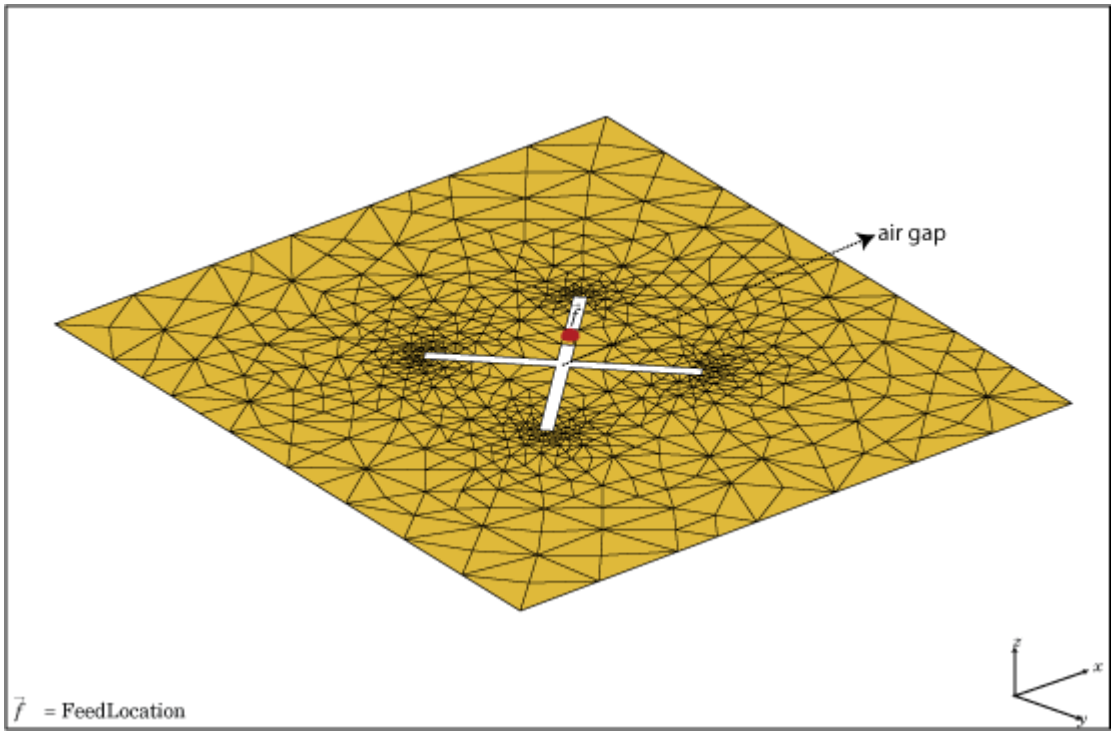
createFeed

Create feed location for custom antenna

Syntax

```
createFeed(antenna)
createFeed(antenna,point1,point2)
```

Description



`createFeed(antenna)` plots a custom antenna mesh in a figure window. From the figure window, you can specify a feed location for the mesh and create a custom antenna.

To specify a region for the feed point, select two points, inside triangles on either side of the air gap or inside triangles that share a common edge.

`createFeed(antenna,point1,point2)` creates the feed across the triangle edges identified by `point1` and `point2`. After the feed is created, when you plot the resulting antenna mesh the feed location is highlighted.

Input Arguments

`antenna` — Custom antenna mesh
scalar handle

Custom mesh antenna, specified as a scalar handle.

`point1,point2` — Points to identify feed region
Cartesian coordinates in meters

Points to identify feed region, specified as Cartesian coordinates in meters. Specify the points in the format $[x_1, y_1]$, $[x_2, y_2]$.

Example: `createFeed(c,[0.07,0.01],[0.05,0.05]);`

Examples

Create Feed for Custom Mesh Antenna Using Air Gap between Triangles

Load a 2-D custom mesh. Create a custom antenna using the points and triangles.

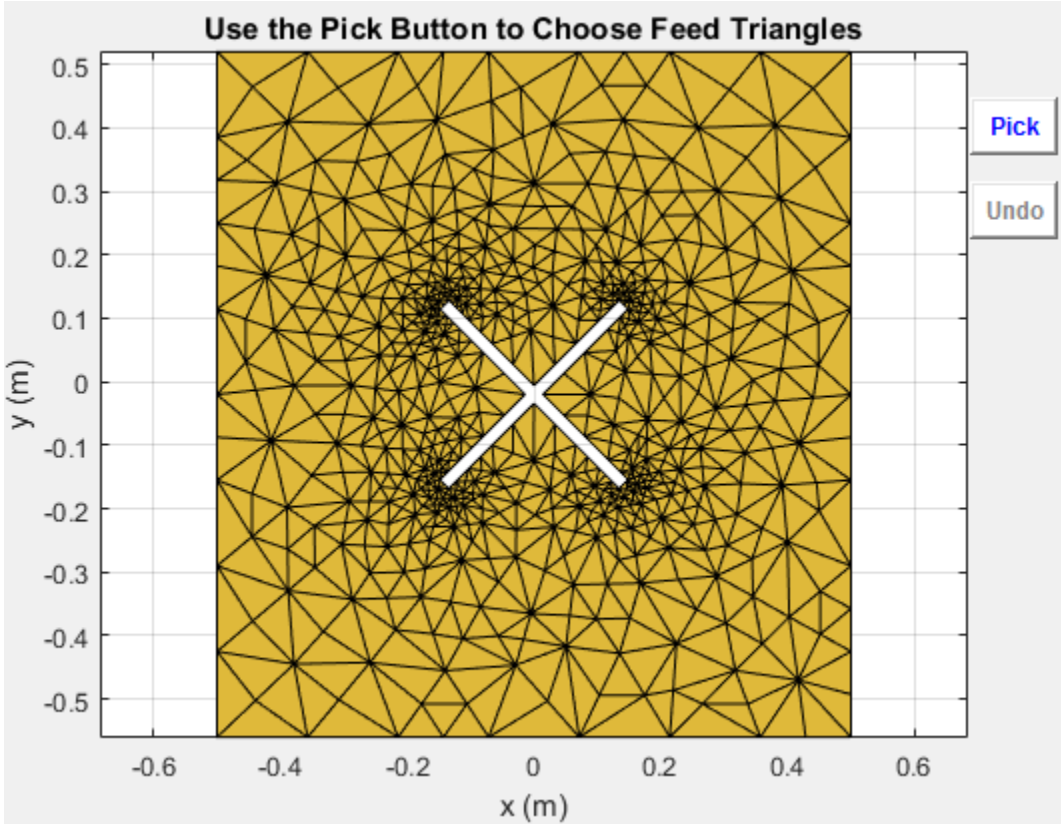
```
load planarmesh.mat
c = customAntennaMesh(p,t)
```

```
c =

  customAntennaMesh with properties:

          Points: [3x658 double]
       Triangles: [4x1219 double]
    FeedLocation: []
```

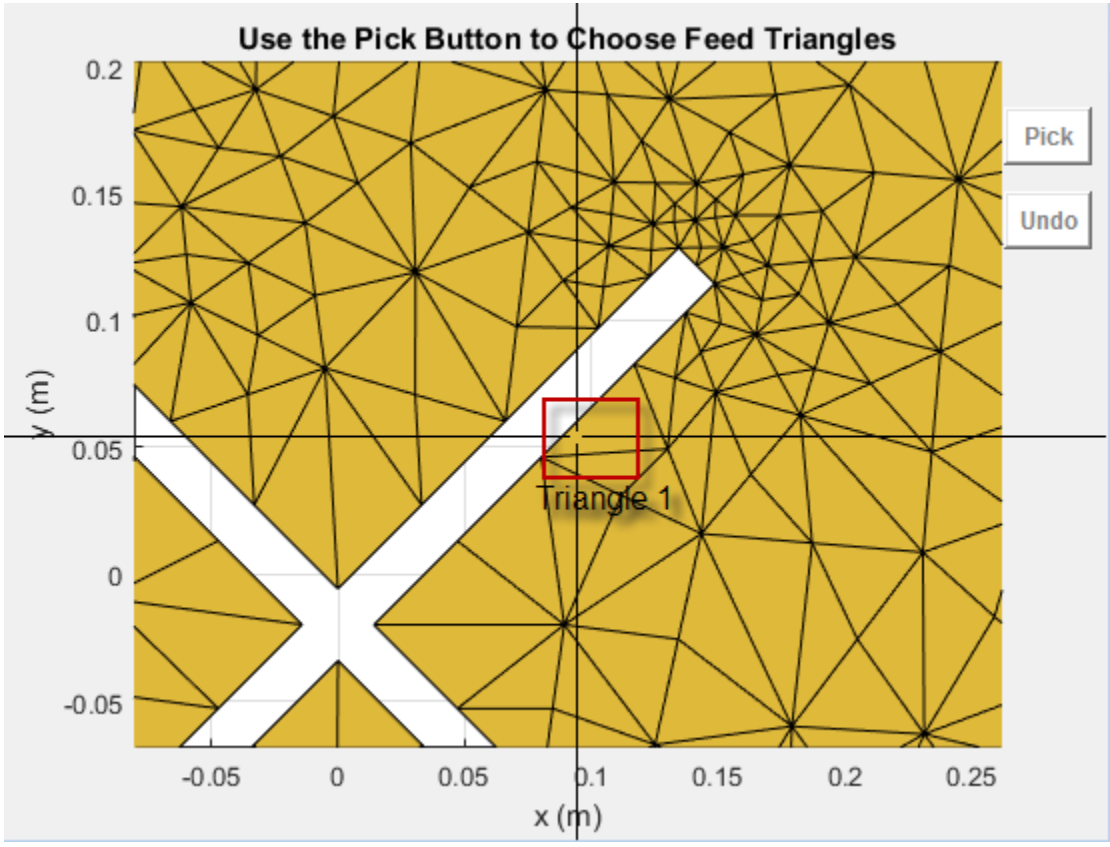
```
    Tilt: 0
TiltAxis: [1 0 0]
```

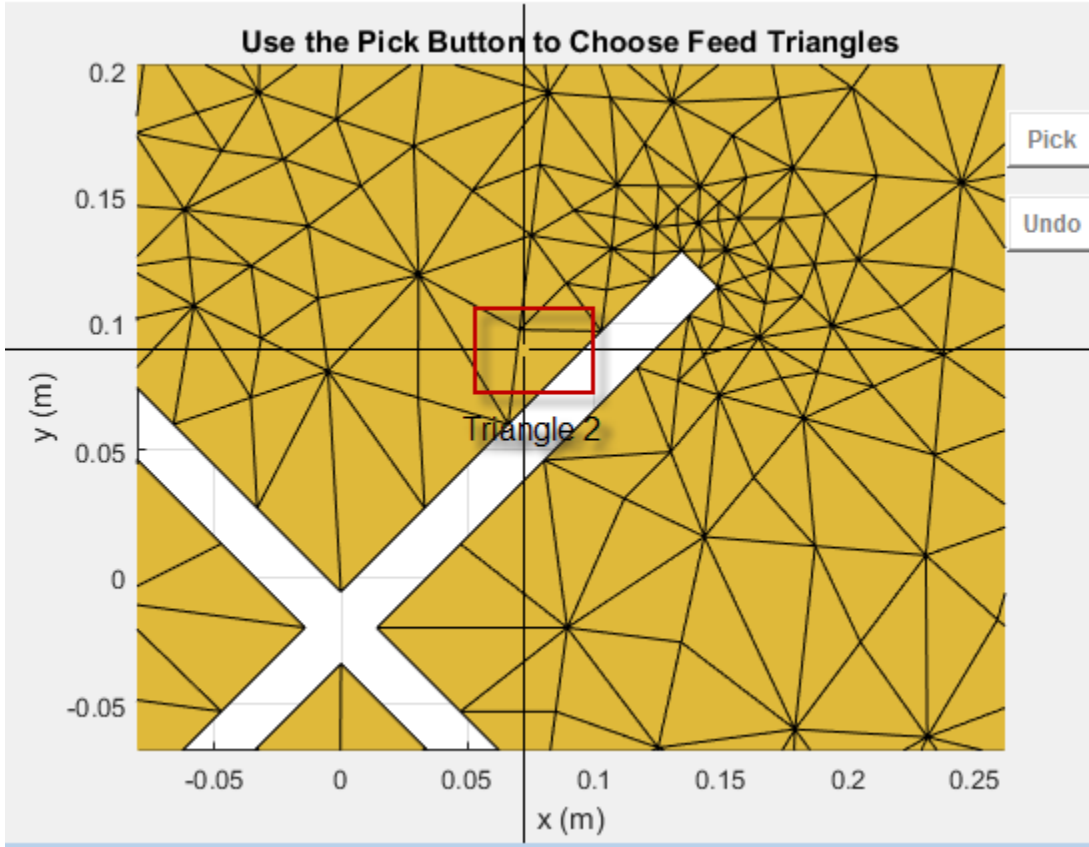
Use the `createFeed` function to view the antenna mesh structure. In this antenna mesh view, you see **Pick** and **Undo** buttons. The **Pick** button is highlighted.

```
createFeed(c)
```



Click **Pick** to display the cross-hairs. To specify a region for the feed point, zoom in and select two points, one inside each triangle on either side of the air gap. Select the points using the cross-hairs.


Use the Pick Button to Choose Feed Triangles
Pick
Undo
Triangle 1
y (m)
x (m)
0.2
0.15
0.1
0.05
0
-0.05
-0.05
0
0.05
0.1
0.15
0.2
0.25



Selecting the second triangle creates and displays the antenna feed.

Create Feed for Custom Mesh Antenna Using Triangles Sharing Edge

Load a 2-D custom mesh. Create a custom antenna using the points and triangles.

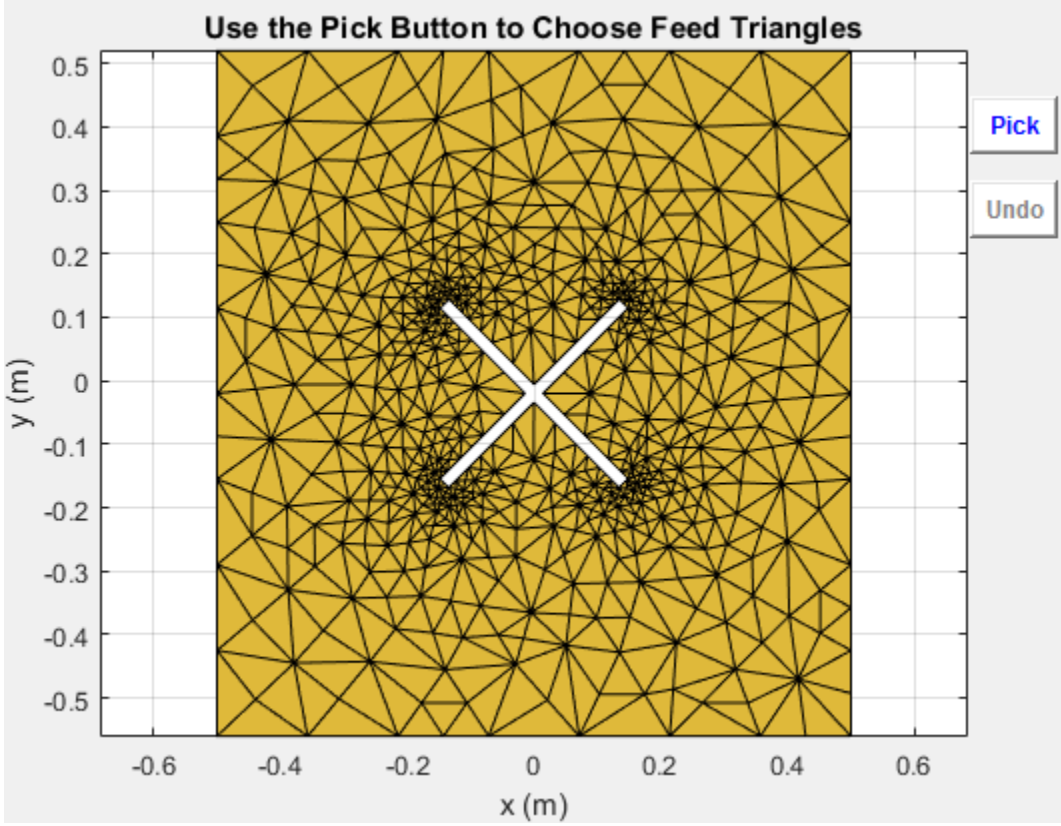
```
load planarmesh.mat
c = customAntennaMesh(p,t)
```

```
c =

  customAntennaMesh with properties:

         Points: [3x658 double]
      Triangles: [4x1219 double]
```

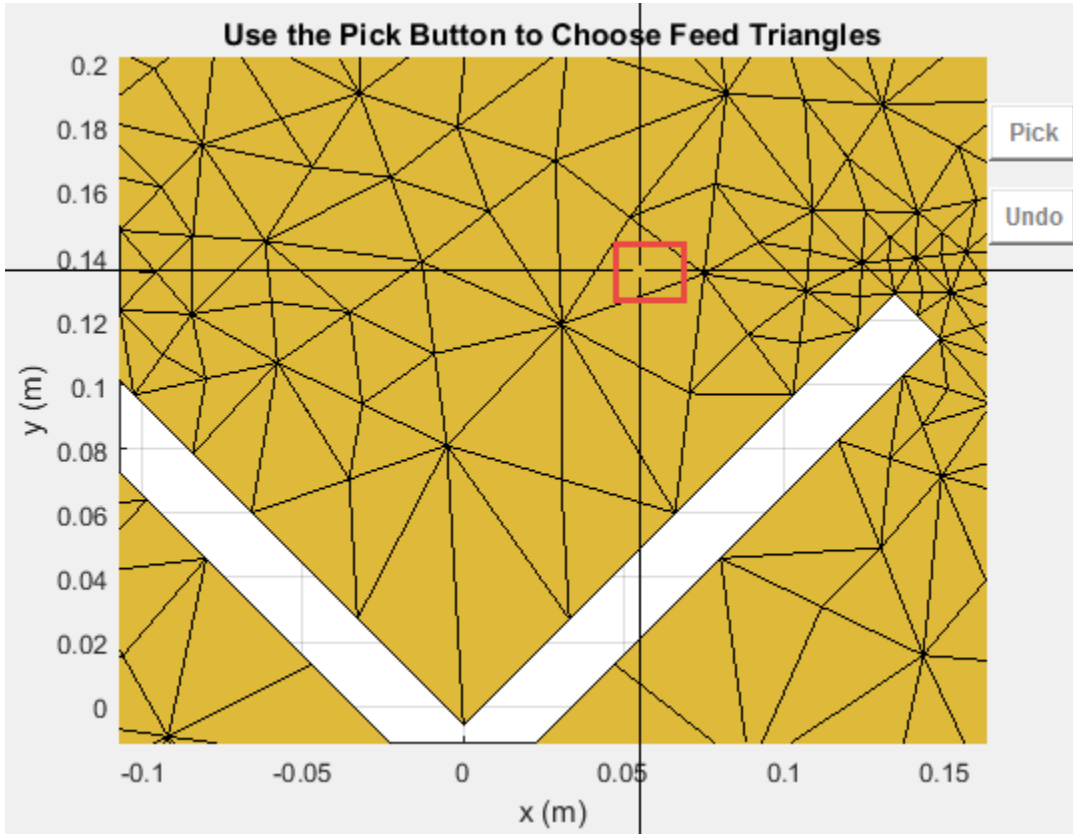
```
    FeedLocation: []
            Tilt: 0
        TiltAxis: [1 0 0]
```

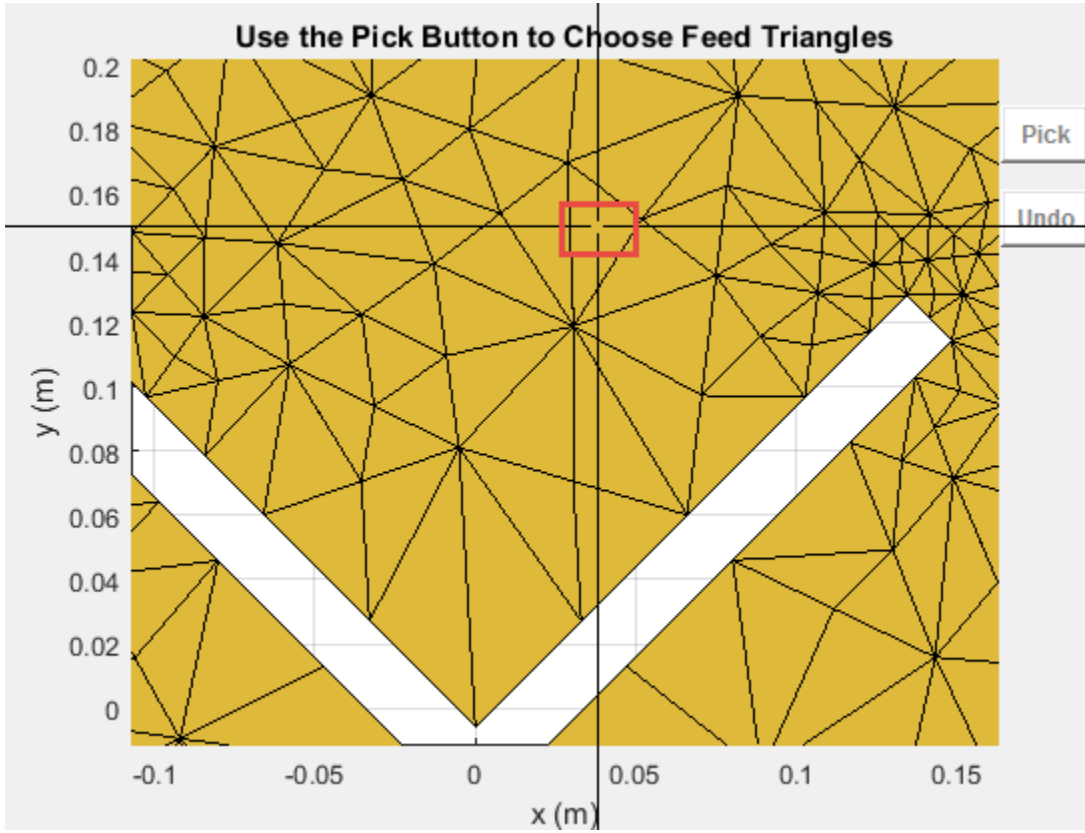
Use the `createFeed` function to view the antenna mesh structure. In this antenna mesh view, you see **Pick** and **Undo** buttons. The **Pick** button is highlighted.

```
createFeed(c)
```

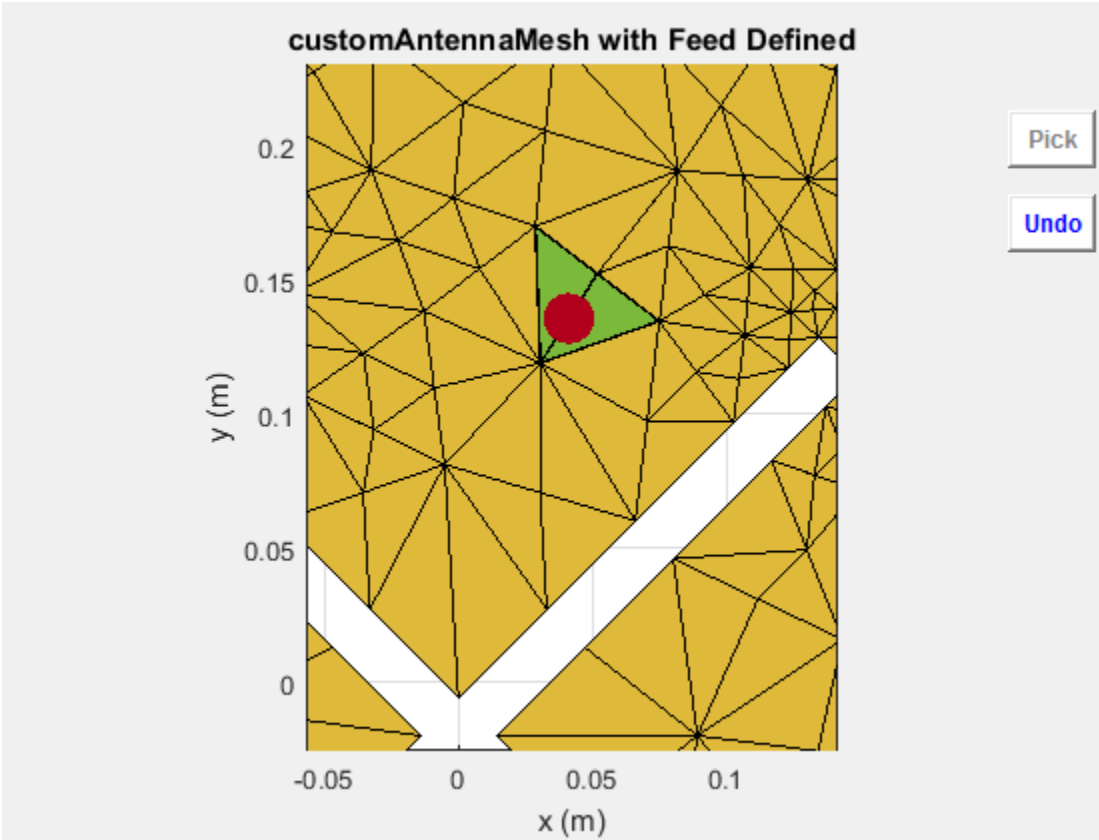


Click **Pick** to display the cross-hairs. To specify a region for the feed point, zoom in and select two points, one inside each triangle sharing an edge. Select the points using the cross-hairs.


Use the Pick Button to Choose Feed Triangles
Pick
Undo
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
y (m)
-0.1
-0.05
0
0.05
0.1
0.15
x (m)



Selecting the second triangle creates and displays the antenna feed.



Create Feed for Custom Antenna Mesh

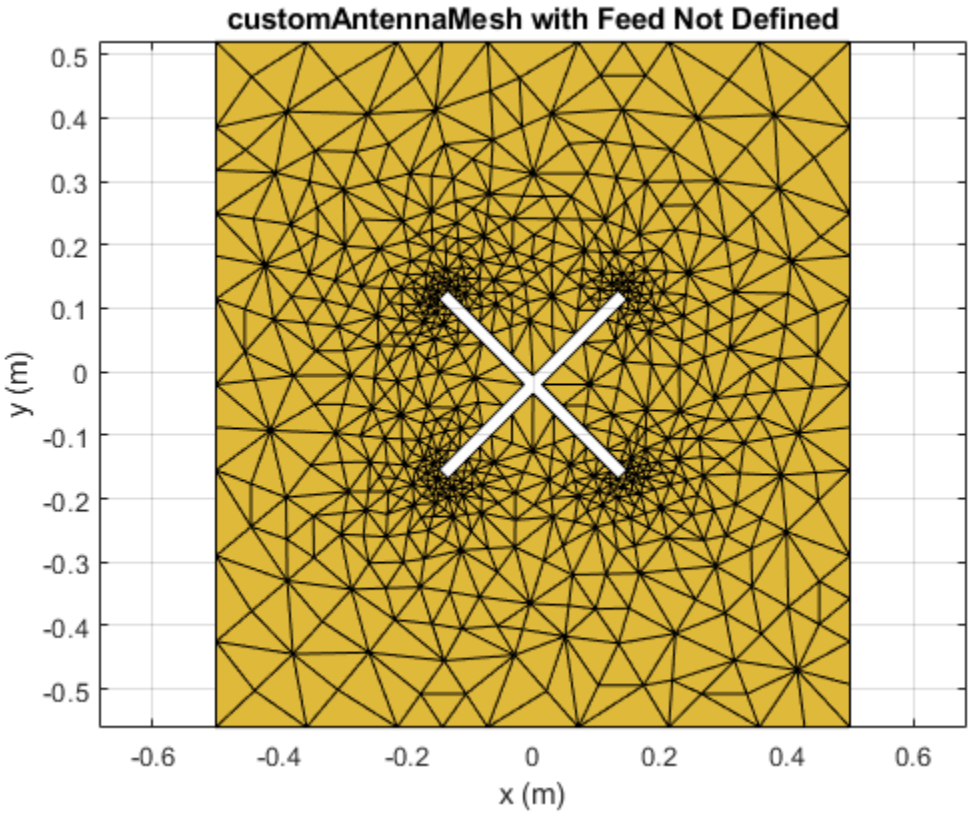
Load a 2-D custom mesh using the planarmesh.mat. Create a custom antenna using the points and triangles.

```
load planarmesh.mat
c = customAntennaMesh(p,t)
```

```
c =
  customAntennaMesh with properties:

          Points: [3x658 double]
       Triangles: [4x1219 double]
```

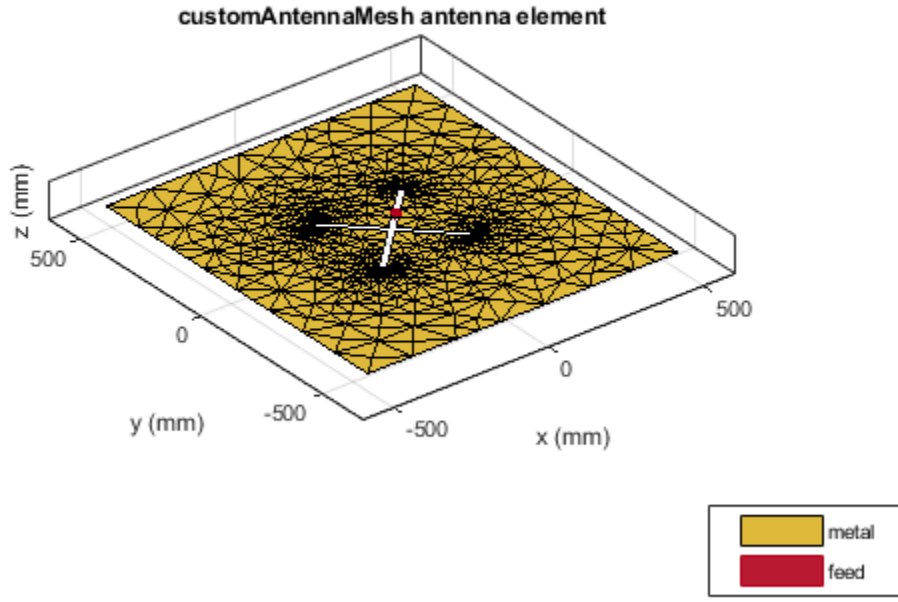
```
    FeedLocation: []
            Tilt: 0
        TiltAxis: [1 0 0]
            Load: [1x1 lumpedElement]
```

```
show (c)
```



Create the feed for the custom antenna across the points (0.07,0.01) and (0.05,0.05) meters respectively.

```
createFeed(c,[0.07,0.01],[0.05,0.05])
show(c)
```



See Also

`returnLoss` | `sparameters`

Introduced in R2015b

EHfields

Electric and magnetic fields of antennas; Embedded electric and magnetic fields of antenna element in arrays

Syntax

```
[e,h] = EHfields(object,frequency)
EHfields(object,frequency)

[e,h] = EHfields(object,frequency,points)
EHfields(object, frequency, points)

EHfields( ___ ,Name,Value)
```

Description

`[e,h] = EHfields(object,frequency)` calculates the *x*, *y*, and *z* components of electric field and magnetic field of an antenna or array object at a specified frequency.

`EHfields(object,frequency)` plots the electric and magnetic field vectors at specified frequency values and at specified points in space.

`[e,h] = EHfields(object,frequency,points)` calculates the *x*, *y*, and *z* components of electric field and magnetic field of an antenna or array object. These fields are calculated at specified points in space and at a specified frequency.

`EHfields(object, frequency, points)` plots the electric and magnetic field vectors at specified frequency values and at specified points in space.

`EHfields( ___ ,Name,Value)` plots the electric and magnetic field vectors with additional options specified by one or more `Name Value` pair arguments using any of the preceding syntaxes.

Use the `'ElementNumber'` and `'Termination'` property to calculate the embedded electric and magnetic fields of the antenna element in an array connected to a voltage source. The voltage source model consists of an ideal voltage source of 1 volt in series

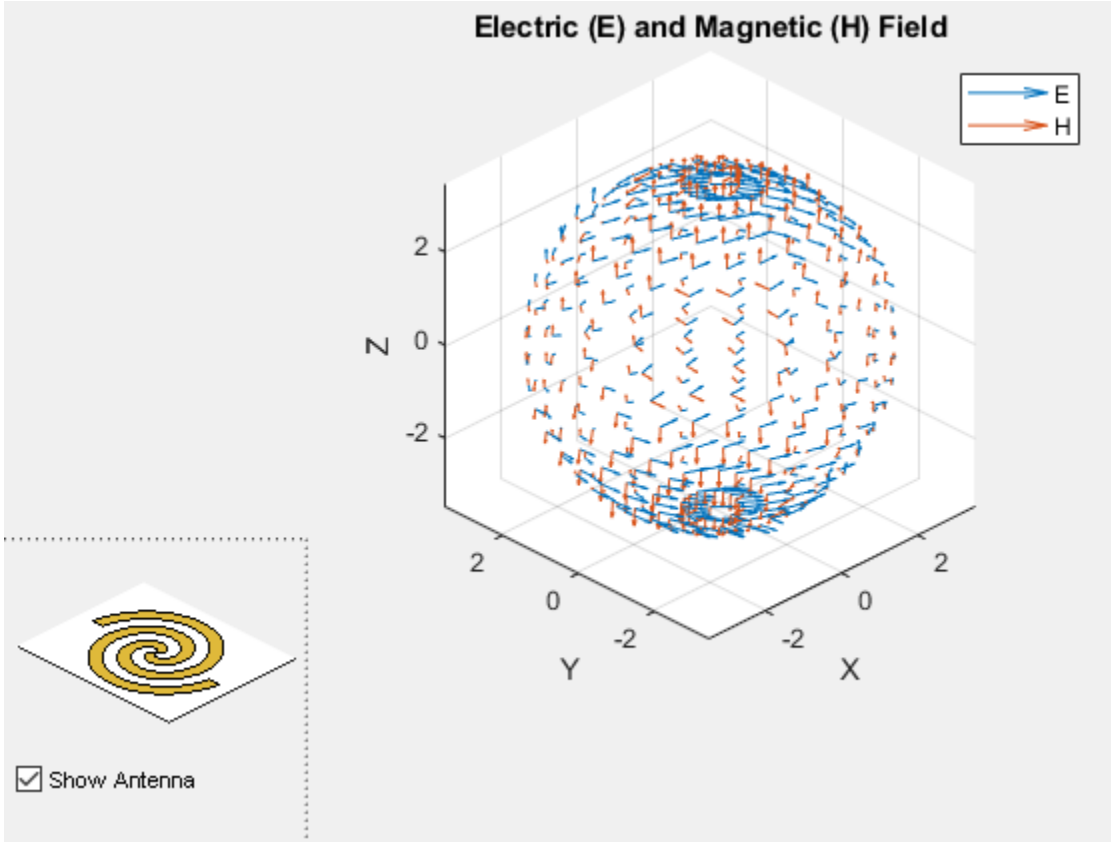
with a source impedance. The embedded pattern includes the effect of mutual coupling due to the other antenna elements in the array.

Examples

Plot E and H Fields of Antenna

Plot electric and magnetic fields of a default Archimedean spiral antenna.

```
h = spiralArchimedean;
EHfields(h,4e9)
```



Calculate EH Fields of Antenna

Calculate electric and magnetic fields at a point 1m along the z-axis from an Archimedean spiral antenna.

```
h = spiralArchimedean;
[e,h] = EHfields(h,4e9,[0;0;1])
```

```
e = 3×1 complex

   0.4137 + 0.2557i
   0.3040 - 0.4084i
   0.0000 + 0.0000i


h = 3×1 complex

  -0.0008 + 0.0011i
   0.0011 + 0.0007i
  -0.0000 - 0.0000i
```

Plot Electric and Magnetic Field Vector of Antenna

Create an Archimedean spiral antenna. Plot electric and magnetic field vector at the z = 1cm plane from the antenna.

```
h = spiralArchimedean;
```

Define points on a rectangular grid in the X-Y plane.

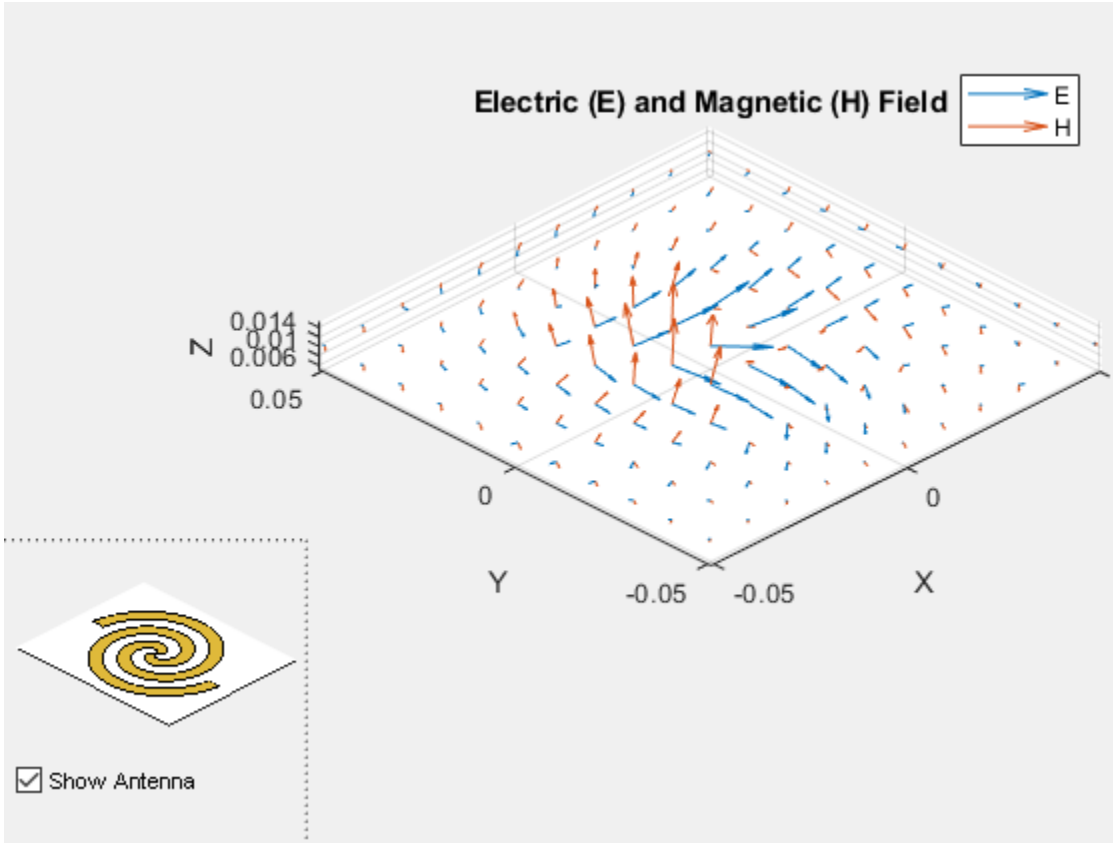
```
[X,Y] = meshgrid(-.05:.01:.05,-.05:.01:.05);
```

Add a z-offset of 0.01.

```
p = [X(:)';Y(:)';.01*ones(1,prod(size(X)))];
```

Plot electric and magnetic field vector at the z = 1cm plane. from the antenna

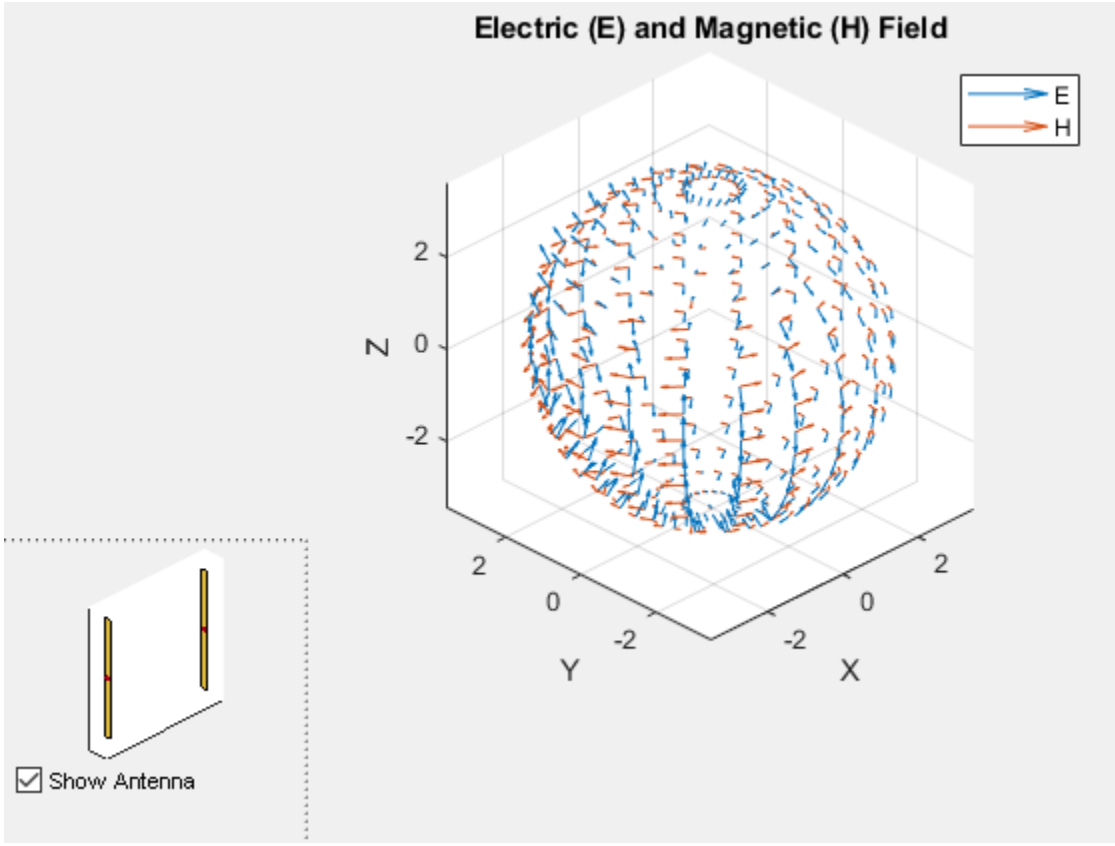
```
EHfields (h,4e9,p)
```



Embedded Vector Fields of Linear Array

Plot the embedded vector fields of a linear array when the first element is excited and all the other antenna elements are terminated using 50-ohm resistance.

```
l = linearArray;
EHfields(l, 70e6, 'ElementNumber', 1, 'Termination', 50);
```



Input Arguments

`object` — Antenna or array object

scalar handle

Antenna or array object, specified as a scalar handle.

Example: `h = spiralArchimedean`

Data Types: `function_handle`

`frequency` — Frequency used to calculate electric and magnetic fields
scalar in Hz

Frequency used to calculate electric and magnetic fields, specified as a scalar in Hz.

Example: 70e6

Data Types: `double`

`points` — Cartesian coordinates of points in space
3-by-*p* complex matrix

Cartesian coordinates of points in space, specified as a 3-by-*p* complex matrix. *p* is the number of points at which to calculate the `E-H` field.

Example: `[0;0;1]`

Data Types: `double`

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` the corresponding value. `Name` must appear inside single quotes (`''`). You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `'ScaleFields',[2 0.5]` specifies scalar values of the electric and magnetic fields

`ScaleFields` — Value by which to scale electric and magnetic fields
two-element vector

Value by which to scale the electric and magnetic fields, specified as the comma-separated pair consisting of `'ScaleFields'` and a two-element vector. The first element scales the E field and the second element scales the H-field. A value of `2` doubles the relative length of either field. A value of `0.5` to halves the length of either field. A value of `0` plots either field without automatic scaling.

Example: `'ScaleFields',[2 0.5]`

Data Types: `double`

`ViewField` — Field to display
`'E'` | `'H'`

Field to display, specified as the comma-separated pair consisting of `'ViewField'` and a text input. `'E'` displays the electric field and `'H'` displays the magnetic field.

Example: `'ViewField', 'E'`

Data Types: `char`

`ElementNumber` — Antenna element in array
scalar

Antenna element in array, specified as the comma-separated pair consisting of `'ElementNumber'` and scalar This antenna element is connected to the voltage source.

Note Use this property to calculate the embedded pattern of an array.

Example: `'ElementNumber',1`

Data Types: `double`

`Termination` — Impedance value for array element termination
50 (default) | scalar

Impedance value for array element termination, specified as the comma-separated pair consisting of `'Termination'` and scalar. The impedance value terminates other antenna elements of an array while calculating the embedded pattern of the antenna connected to the voltage source.

Note Use this property to calculate the embedded pattern of an array.

Example: `'Termination',40`

Data Types: `double`

Output Arguments

e — *x*, *y*, *z* components of electrical field
3-by-*p* complex matrix in V/m

x, *y*, *z* components of electrical field, returned as 3-by-*p* complex matrix in V/m. The dimension *p* is the Cartesian coordinates of points in space.

h — *x*, *y*, *z* components of magnetic field
3-by-*p* complex matrix in H/m

x, *y*, *z* components of magnetic field, returned as a 3-by-*p* complex matrix in H/m. The dimension *p* is the Cartesian coordinates of points in space.

See Also

`axialRatio` | `beamwidth`

Introduced in R2015a

axialRatio

Axial ratio of antenna

Syntax

```
axialRatio(antenna,frequency,azimuth,elevation)
ar= axialRatio(antenna,frequency,azimuth,elevation)
```

Description

`axialRatio(antenna,frequency,azimuth,elevation)` plots axial ratio of an antenna over a specified frequency, and in the direction specified by `azimuth` and `elevation`.Any one among frequency, azimuth, or elevation values must be scalar. If only one of the values are scalar, the plot is 3-D. If two values are scalar, the plot is 2-D.

`ar= axialRatio(antenna,frequency,azimuth,elevation)` returns the axial ratio of an antenna, over the specified frequency, and in the direction specified by, `azimuth` and `elevation`.

Examples

Calculate Axial Ratio of Antenna

Calculate the axial ratio of an equiangular spiral antenna at azimuth=0 and elevation=0.

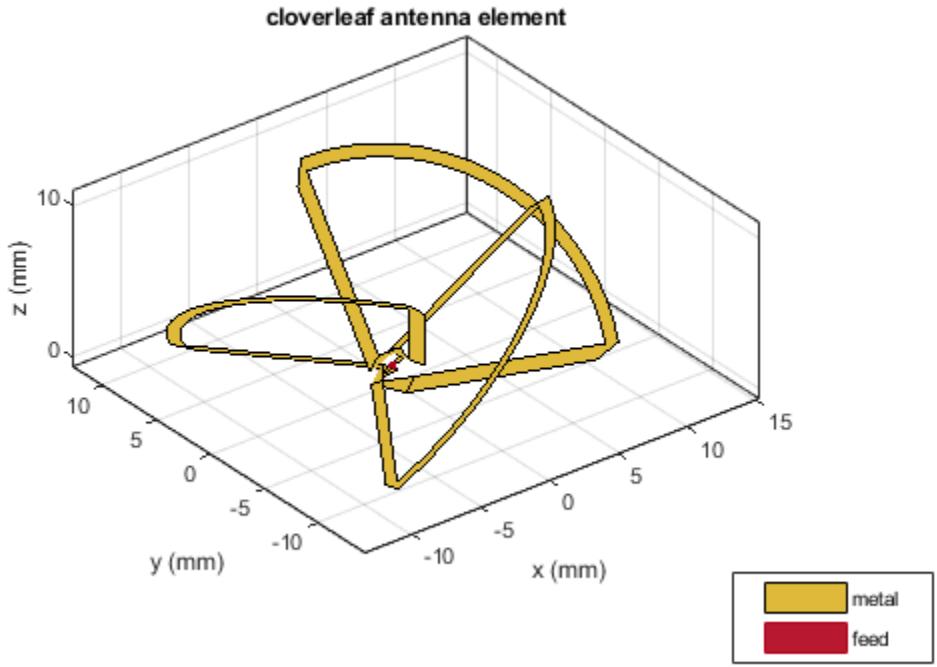
```
s  = spiralEquiangular;
ar = axialRatio(s,3e9,0,0)
```

```
ar = Inf
```

Axial Ratio of Cloverleaf Antenna

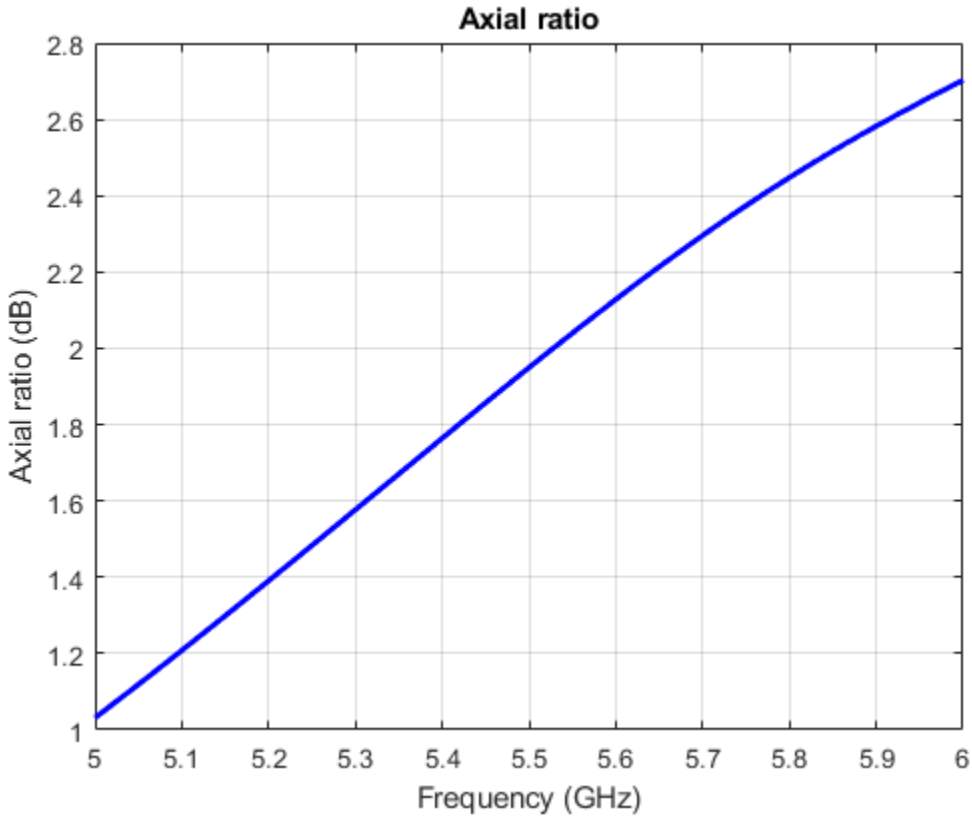
Create a cloverleaf antenna.

```
cl = cloverleaf;
show(cl);
```



Plot the axial ratio of the antenna from 5 GHz to 6 GHz.

```
freq = linspace(5e9,6e9,101);
axialRatio(cl,freq,0,0);
```



You can see from the axial ratio plot that the antenna supports circular polarization over the entire frequency range.

Input Arguments

`antenna` — Antenna element
object

Antenna object, specified as an object.

`frequency` — Frequency used to calculate axial ratio
scalar | vector

Frequency used to calculate axial ratio, specified as a scalar or vector with each element in Hz.

Example: 70e6

Data Types: `double`

`azimuth` — Azimuth angle of antenna
scalar | vector

Azimuth angle of antenna, specified as a scalar or vector with each element in degrees.

`elevation` — Elevation angle of antenna
scalar | vector

Elevation angle of antenna, specified as a scalar or vector with each element in degrees.

Output Arguments

`ar` — Axial ratio of antenna
scalar in dB

Axial ratio of antenna, returned as a scalar in dB.

See Also

`beamwidth` | `pattern`

Introduced in R2015a

beamwidth

Beamwidth of antenna

Syntax

```
beamwidth(antenna,frequency,azimuth,elevation)
bw = beamwidth(antenna,frequency,azimuth,elevation,dBdown)

[bw,angles] = beamwidth(____)
```

Description

`beamwidth(antenna,frequency,azimuth,elevation)` plots the beamwidth of the input antenna at a specified frequency. The beamwidth is the angular separation at which the magnitude of the directivity pattern decreases by a certain value from the peak of the main beam. The directivity decreases in the direction specified by azimuth and elevation angles of the antenna.

`bw = beamwidth(antenna,frequency,azimuth,elevation,dBdown)` returns the beamwidth of the antenna at a specified `dBdown` value from the peak of the radiation pattern main beam.

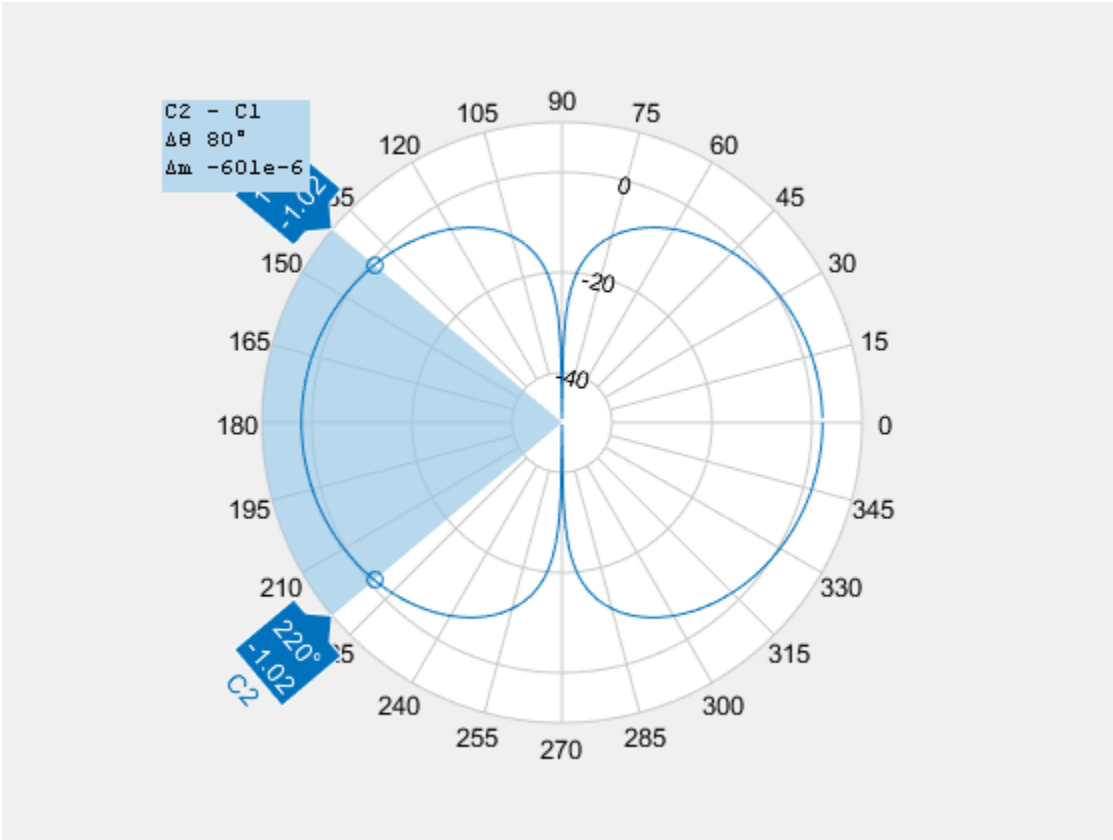
`[bw,angles] = beamwidth(____)` returns the beamwidth and angles (points in a plane) using any input arguments from previous syntaxes.

Examples

Plot Beamwidth of Dipole Antenna

Plot the beamwidth for a dipole antenna at azimuth=0 and elevation=1:1:360 (x-z plane)

```
d  = dipole;
beamwidth(d,70e6,0,1:1:360);
```



Calculate Beamwidth and Angles of Antenna

Calculate the beamwidth of a helix antenna and the angles of the beamwidth. The antenna has an azimuth angle of 1:1:360 degrees, an elevation angle of 0 degrees on the X-Y plane, and a dB down value of 5 dB.

```
hx = helix;
[bw,angles] = beamwidth(hx,2e9,1:1:360,0,5)
```

```
bw = 145
```

```
angles = 1×2

   143   288
```

Input Arguments

`antenna` — Antenna object
scalar handle

Antenna object, specified as a scalar handle.

`frequency` — Frequency used to calculate beamwidth
scalar in Hz

Frequency to calculate beamwidth, specified as a scalar in Hz.

Example: 70e6

Data Types: `double`

`azimuth` — Azimuth angle of antenna
scalar in degrees | vector in degrees

Azimuth angle of the antenna, specified as a scalar or vector in degrees. If the elevation angle is specified as a vector, then the azimuth angle must be a scalar.

Example: 3

Data Types: `double`

`elevation` — Elevation angle of antenna
scalar in degrees | vector in degrees

Elevation angle of the antenna, specified as a scalar or vector in degrees. If the azimuth angle is specified as a vector, then the elevation angle must be a scalar.

Example: 1:1:360

Data Types: `double`

`dBdown` — Power point from peak of main beam of antenna
3 (default) | scalar in dB

Power point from peak of main beam of antenna, specified as a scalar in dB.

Example: 5

Data Types: `double`

Output Arguments

`bw` — Beamwidth of antenna
scalar in degrees

Beamwidth of antenna, returned as a scalar in degrees.

`angles` — Points on plane
vector in degrees

Points on plane used to measure beamwidth, returned as a vector in degrees.

See Also

`axialRatio` | `pattern`

Introduced in R2015a

mesh

Mesh properties of metal or dielectric antenna or array structure

Syntax

```
mesh(object)
mesh(shape)
mesh(object,Name,Value)
meshdata = mesh( ___ ,Name,Value)
```

Description

`mesh(object)` plots the mesh used to analyze antenna or array element.

`mesh(shape)` plots the mesh for the shapes.

`mesh(object,Name,Value)` changes and plots the mesh structure of an antenna or array element, using additional options specified by the name-value pairs. You can also determine the number of unknowns from the number of basis functions in the output.

`meshdata = mesh( ___ ,Name,Value)` returns a mesh structure that specifies the properties used to analyze the antenna or array.

Examples

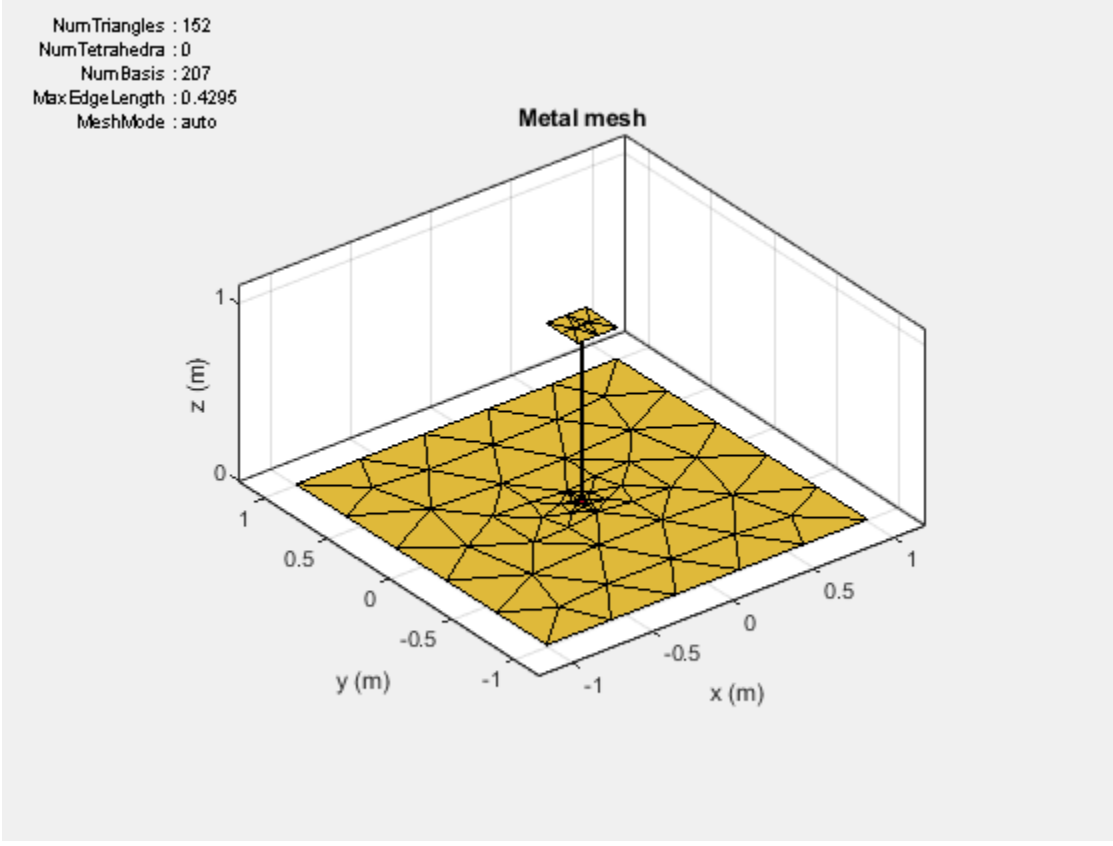
View Mesh Structure of Antenna

Create and view the mesh structure of a top hat monopole antenna with Maximum edge length of 0.1 m.

```
h = monopoleTopHat;
i = impedance(h,75e6)
```

```
i = 2.5322e+02 + 6.0784e+02i
```

```
mesh(h)
```



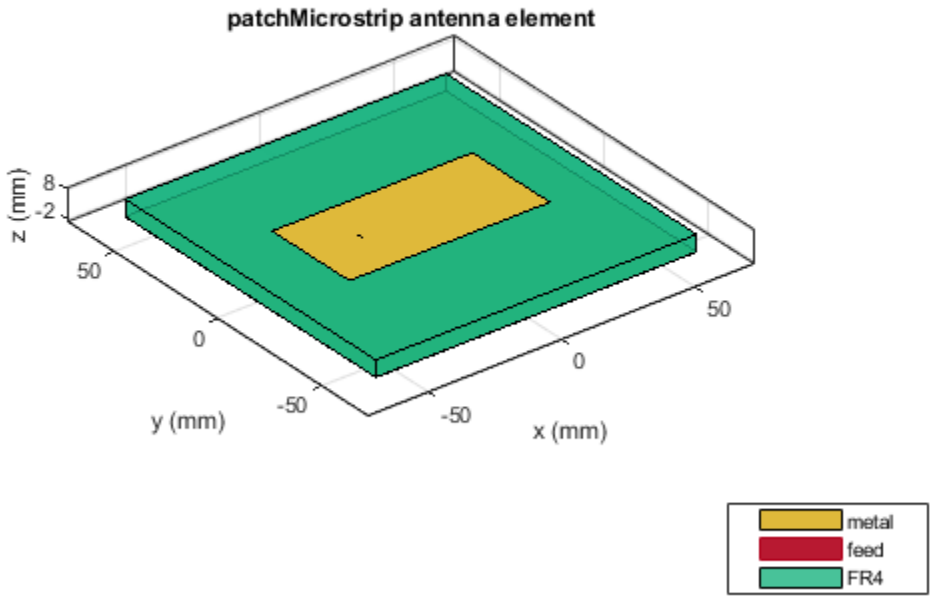

```
m = mesh(h)

m = struct with fields:
     NumTriangles: 152
    NumTetrahedra: 0
         NumBasis: 207
    MaxEdgeLength: 0.4295
         MeshMode: 'auto'
```

Mesh Microstrip Patch Metal-Dielectric Antenna

Radiation Pattern of Microstrip Patch Antenna

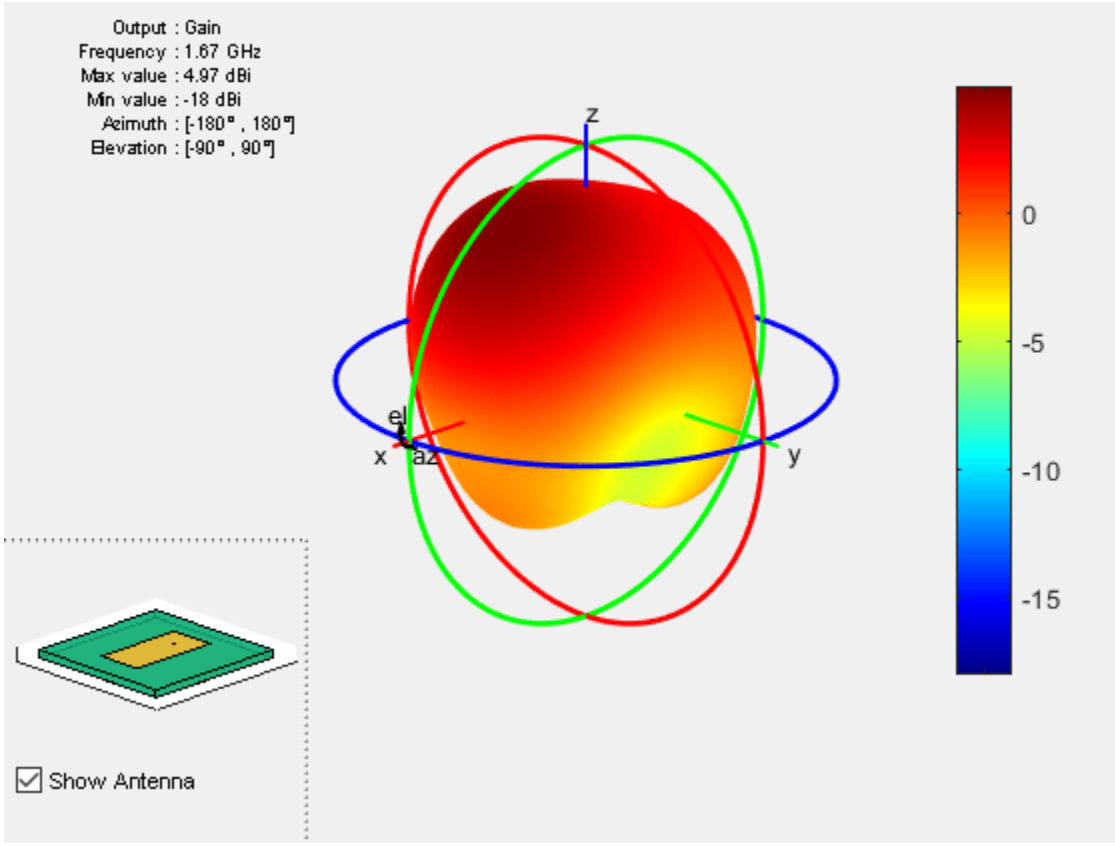
Create a microstrip patch antenna using **'FR4'** as the dielectric substrate.

```
d = dielectric('FR4');
pm = patchMicrostrip('Length',75e-3, 'Width',37e-3,                  ...
        'GroundPlaneLength',120e-3, 'GroundPlaneWidth',120e-3, ...
        'Substrate',d);
show(pm)
```



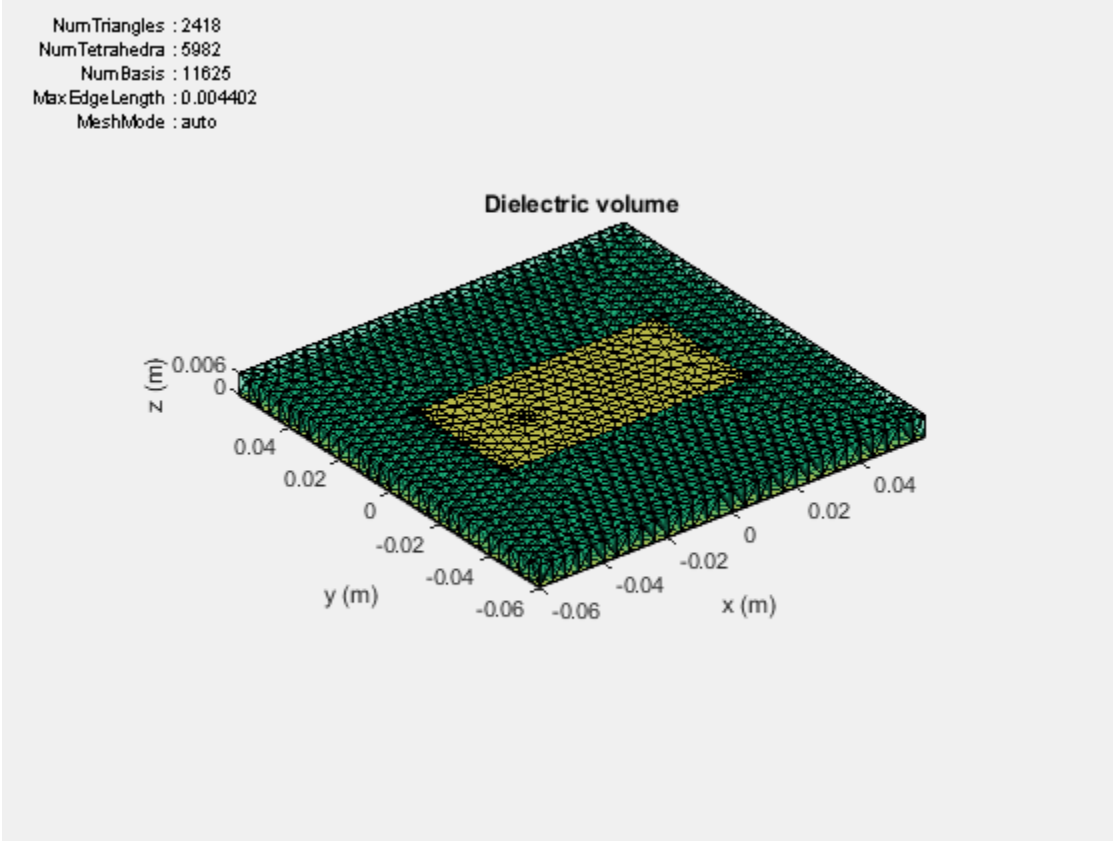
Plot the radiation pattern of the antenna at a frequency of 1.67 GHz.

```
figure
pattern(pm,1.67e9)
```



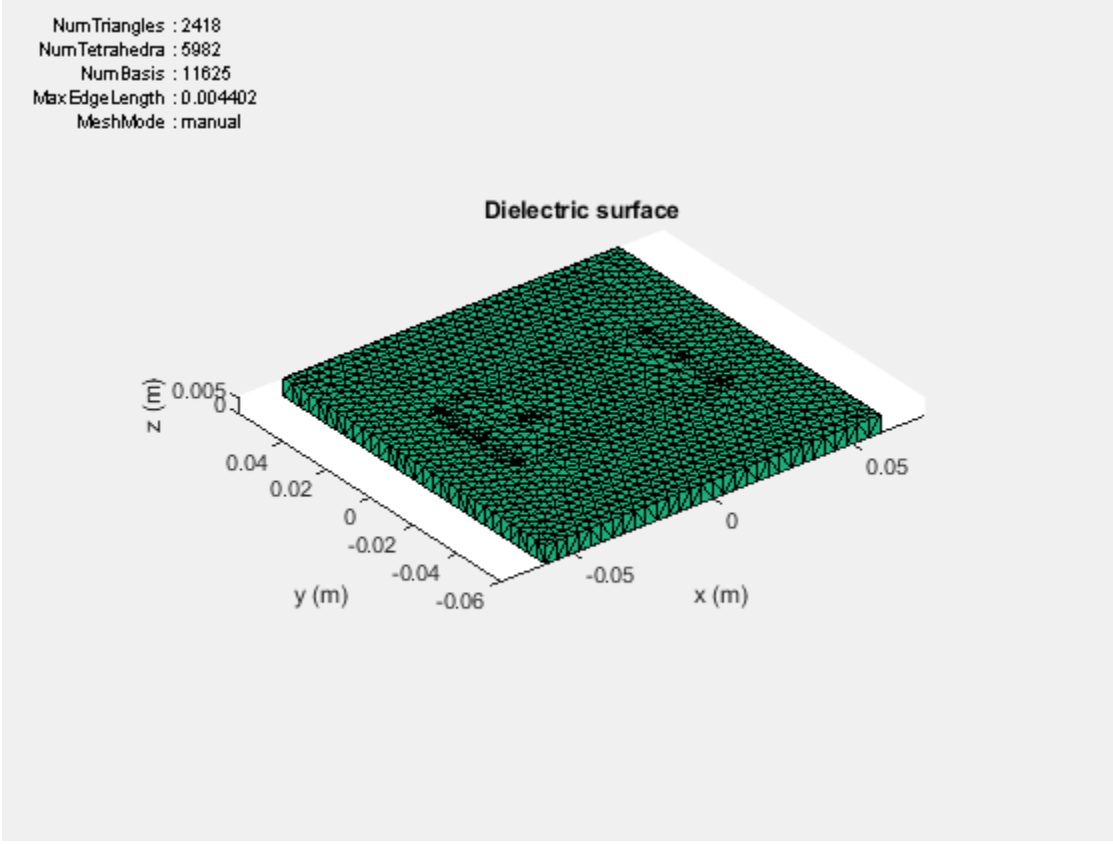
Mesh the whole antenna.

```
figure
mesh(pm)
```



Mesh only the dielectric surface of the antenna.

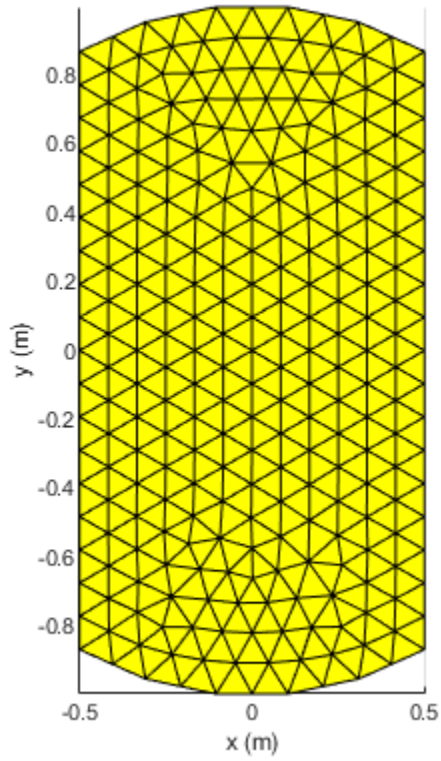
```
figure
mesh(pm,'View','dielectric surface')
```



Mesh Arbitrary Shape

Create a rectangular and circular shape, intersect them and mesh at a wavelength of 2 m.

```
r  = antenna.Rectangle;
c  = antenna.Circle;
p = r&c;
mesh(p,2);
```



Input Arguments

`object` — Antenna or array element
object

Antenna or array element, specified as an object.

`shape` — Shape created using custom elements and shape objects
object handle

Shape created using custom elements and shape objects of Antenna Toolbox, specified as an object handle. You can create the shapes using `antenna.Circle`, `antenna.Polygon`, or `antenna.Rectangle`.

Example: `c = antenna.Rectangle; mesh(c)`

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` pair arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside single quotes (`''`). You can specify several name and value pair arguments in any order as `Name1, Value1, ..., NameN, ValueN`.

Example: 'MaxEdgeLength', 0.1

`MaxEdgeLength` — Maximum edge length of triangles in mesh

scalar

Maximum edge length of triangles in mesh, specified as a comma-separated pair consisting of `'MaxEdgeLength'` and a scalar. All triangles in the mesh have sides less than or equal to the `'MaxEdgeLength'` value.

Data Types: `double`

`MinEdgeLength` — Minimum edge length of triangles in mesh

scalar

Minimum edge length of triangles in mesh, specified as a comma-separated pair consisting of `'MinEdgeLength'` and a scalar. All triangles in the mesh have sides less than or equal to the `'MinEdgeLength'`.

Note You can use this property only with the `pcbStack` object.

Data Types: `double`

`GrowthRate` — Mesh growth rate

`0.7` (default) | scalar

Mesh growth rate, specified as a comma-separated pair consisting of `'GrowthRate'` and a scalar. The default value of `0.7` states that the growth rate of the mesh is 70 percent. Growth rate values lie between `0` and `1`.

Note You can use this property only with the `pcbStack` object.

Data Types: `double`

`View` — Customize mesh view of antenna or array element

`'all'` (default) | `'metal'` | `'dielectric surface'` | `'dielectric volume'`

Customize mesh view of antenna or array element, specified as a comma-separated pair consisting of `'View'` and `'all'`, `'metal'`,`'dielectric surface'`, or `'dielectric volume'`.

You choose `'dielectric surface'` to view the boundary triangle mesh of the dielectric. You choose `'dielectric volume'` to view the tetrahedral volume mesh of the dielectric.

Data Types: `char`

See Also

`meshconfig` | `plot` | `show`

Introduced in R2015a

layout

Display array layout

Syntax

```
layout(array)
```

Description

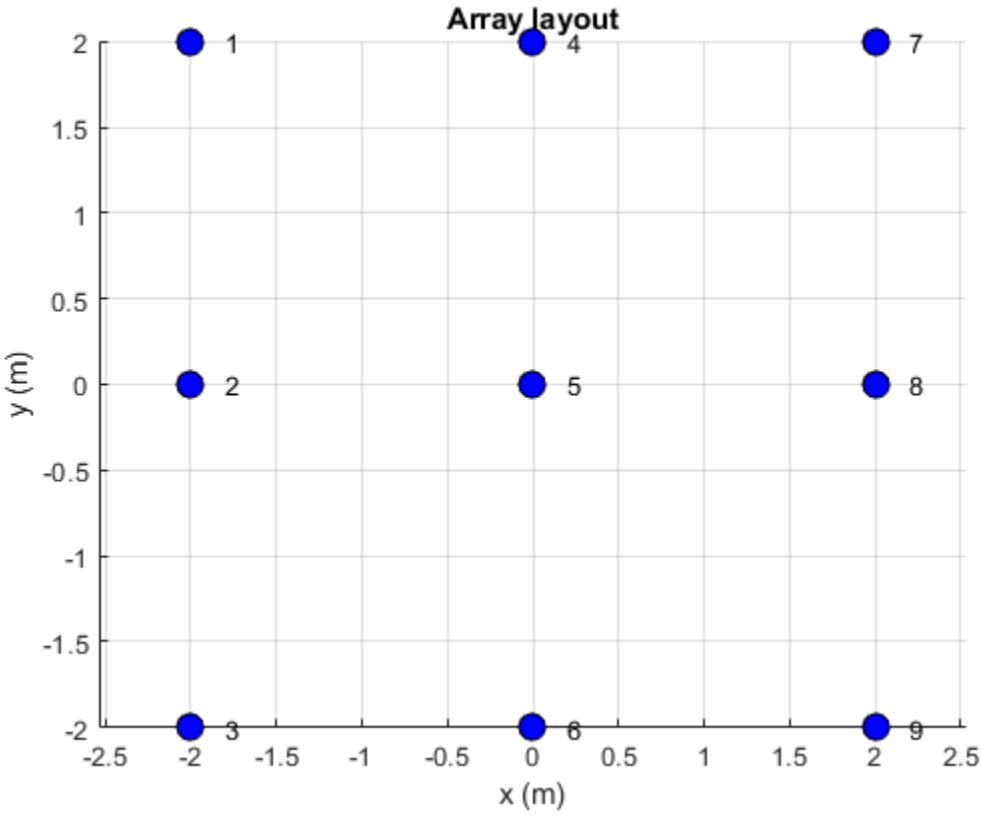
`layout(array)` displays the layout of the array object. The circles in the layout corresponds to antenna feed points within the array.

Examples

Display Array Layout on X-Y Plane

Create and view a 3x3 rectangular array layout on the X-Y plane.

```
h = rectangularArray('Size',[3 3]);
layout(h)
```



Input Arguments

`array` — Array object
scalar handle

Array object, specified as a scalar handle.

See Also

`show`

Introduced in R2015a

lumpedElement

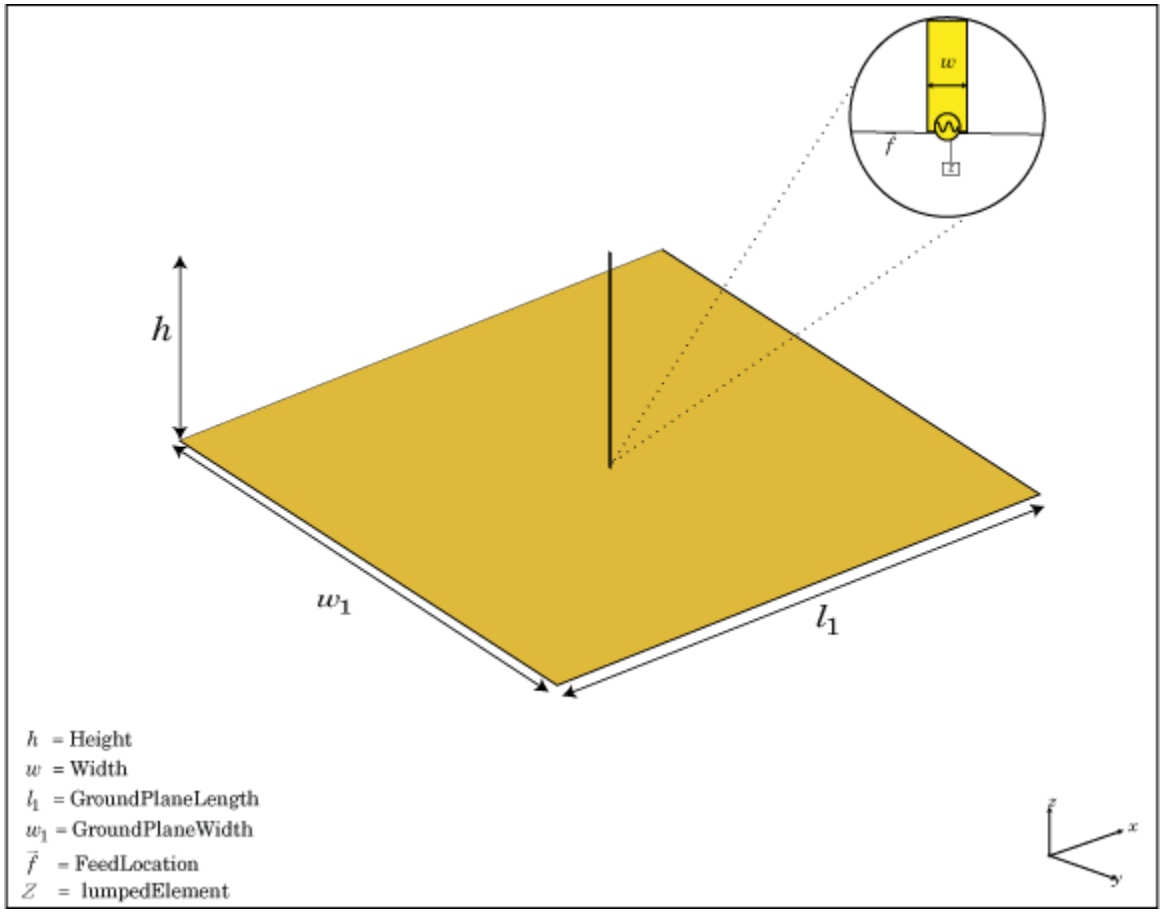
Lumped element circuit to load antenna

Syntax

```
le = lumpedElement
le = lumpedElement(Name,Value)
```

Description

`le = lumpedElement` creates a lumped element circuit. The default value is an empty `lumpedElement` object.



When you load an antenna using a lumped resistor, capacitor, or inductor, the electrical properties of the antennas changes. These lumped elements are typically added to the antenna feed. You can use lumped elements to increase the bandwidth of the antenna without increasing the size of the antenna.

`le = lumpedElement(Name,Value)` returns the lumped element circuit based on the properties specified by one or more `Name,Value` pair arguments.

Examples

Antenna Using Frequency Independent Load

Create a resistor with 50 Ohms of impedance. Any pure resistive load has a nonvariable impedance when the frequency changes.

```
le = lumpedElement('Impedance',50);
```

Create a dipole antenna. Calculate the impedance of the antenna without loading the antenna.

```
d = dipole;
i1 = impedance(d,70e6)
```

```
i1 = 72.9288 - 1.3947i
```

Load the antenna using a frequency-independent resistor. Calculate the impedance of the antenna.

```
d.Load = le;
i1e1 = impedance(d,70e6)
```

```
i1e1 = 1.2293e+02 - 1.3947e+00i
```

Change the frequency to 85 MHz and calculate the impedance of the antenna.

```
ile2 = impedance(d,85e6)
```

```
ile2 = 1.8009e+02 + 1.1005e+02i
```

Antenna with Two Loads at Arbitrary Locations

Create a dipole antenna using one load at the antenna feed and one load at a location above the antenna feed.

Create a dipole antenna.

```
d = dipole;
```

Create two lumped elements to load the dipole antenna.

One lumped element of impedance, `50 Ohms`, loads the antenna at the feed.

```
l1 = lumpedElement('Impedance', complex(50, -20), 'Location', 'feed');
```

The second lumped element of complex impedance, `50+ j*20 Ohms`, loads the antenna at the top. Locate the load half distance from the feed.

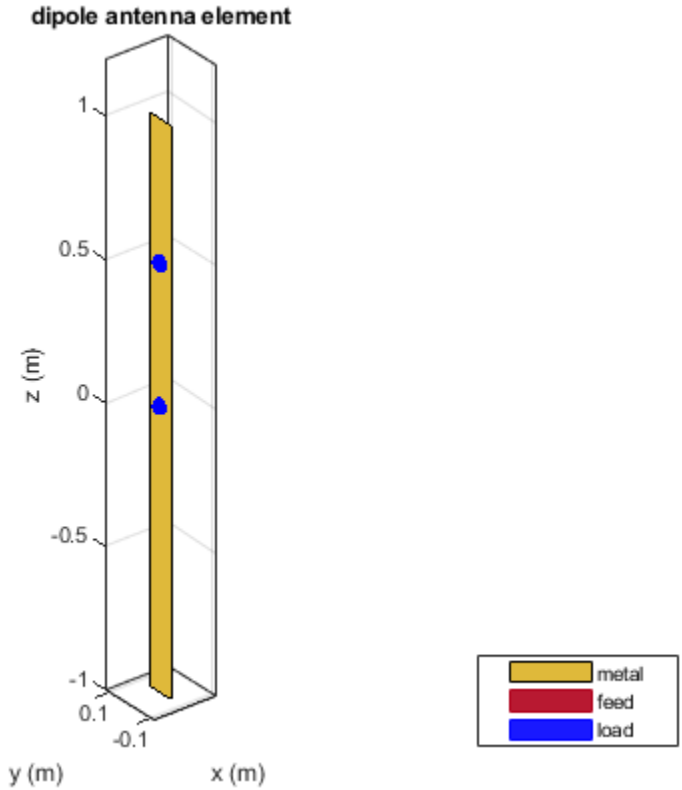
```
l2 = lumpedElement('Impedance', complex(50, -20), 'Location', [0 0 0.5]);
```

Add the two loads to the dipole antenna.

```
 d.Load = [l1, l2];
```

View the dipole antenna.

```
show(d);
```



Input Arguments

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `'Frequency',2e9`

`Impedance` — Complex impedance of circuit
real or complex vector of Z-parameters in ohms

Complex impedance of circuit, specified as the comma-separated pair consisting of `'Impedance'` and a real or complex vector of z-parameters in ohms.

Example: `'Impedance',complex(75,30)` specifies a complex impedance of `75+i30`.

Data Types: `double`

`Frequency` — Frequency of operation
real vector in Hz

Frequency of operation, specified as the comma-separated pair consisting of `'Frequency'` and a real vector in Hz.

Example: `'Frequency',[10e6,20e6,30e6]`

Data Types: `double`

`Location` — Location of load
`[0 0 0]` (default) | Cartesian coordinates

Location of load, specified as the comma-separated pair consisting of `'Location'` and Cartesian coordinates.

Example: `'Location',[0 0 0.5]`

Data Types: `double`

Output Arguments

`le` — Lumped element
`lumpedElement` object

Lumped element, returned as a `lumpedElement` object. The real part of the complex number indicates the resistance. The imaginary part of the complex number indicates the reactance.

See Also

`dielectric`

Introduced in R2016b

vswr

Voltage standing wave ratio of antenna

Syntax

```
vswr(antenna,frequency,z0)
vswrant = vswr(antenna,frequency,z0)
```

Description

`vswr(antenna,frequency,z0)` calculates and plots the voltage standing wave ratio of an antenna, over specified frequency range, and given reference impedance, `z0`.

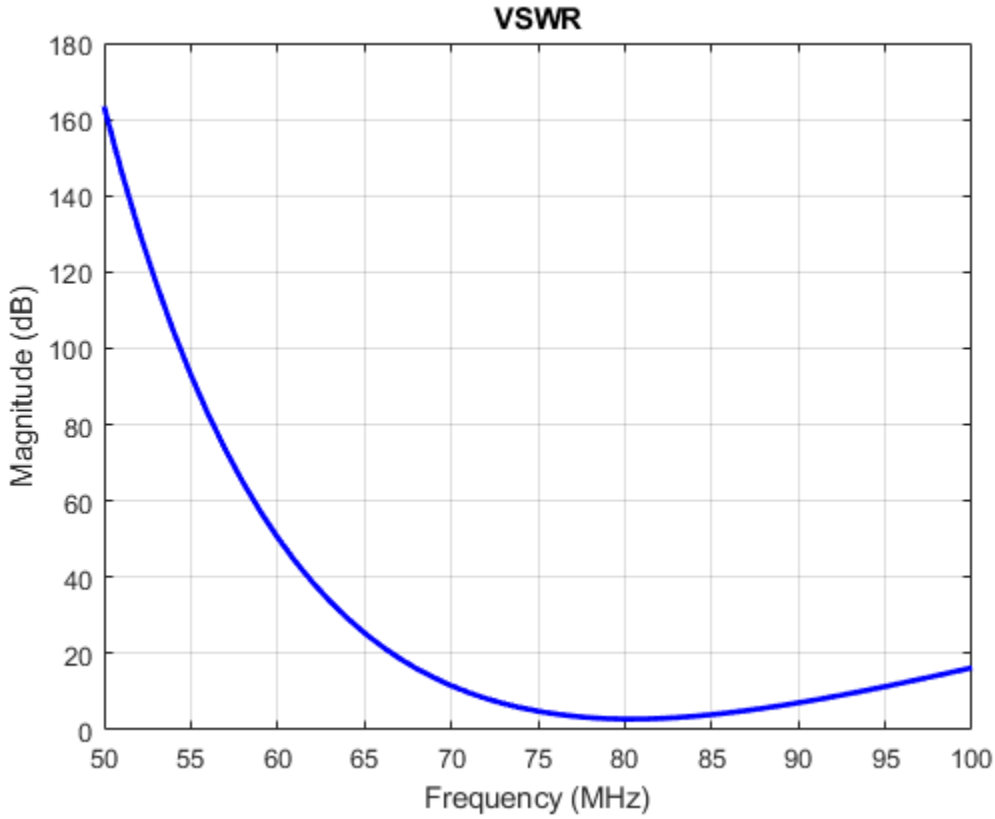
`vswrant = vswr(antenna,frequency,z0)` returns the vswr of the antenna.

Examples

Plot VSWR of Antenna

Plot vswr (voltage standing wave ratio) of a circular loop antenna.

```
h = loopCircular;
vswr(h,50e6:1e6:100e6,50)
```



Calculate VSWR of Antenna

Calculate vswr (voltage standing wave ratio) of a helix antenna.

```
h = helix;
hvswr = vswr(h,2e9:1e9:4e9,50)
```

```
hvswr = 1×3

    3.5730    6.7041    3.3598
```

Input Arguments

`antenna` — Antenna object
scalar handle

Antenna object, specified as a scalar handle.

`frequency` — Frequency range used to calculate VSWR
vector in Hz

Frequency range used to calculate VSWR, specified as a vector in Hz. The minimum value of frequency must be 1 kHz.

Example: 50e6:1e6:100e6

Data Types: `double`

`z0` — Reference impedance
50 (default) | scalar in dB

Reference impedance, specified as a scalar in dB.

Output Arguments

`vswrant` — Voltage standing wave ratio
vector in dB

Voltage standing wave ratio, returned as a vector in dB.

See Also

`impedance`

Introduced in R2015a

correlation

Correlation coefficient between two antennas in array

Syntax

```
correlation(array,frequency,elem1,elem2,z0)
rho = correlation(array,frequency,elem1,elem2,z0)
```

Description

`correlation(array,frequency,elem1,elem2,z0)` calculates and plots the correlation coefficient between two antenna elements, `elem1` and `elem2` of an array. The correlation values are calculated for a specified frequency and impedance and for a specified impedance `z0`.

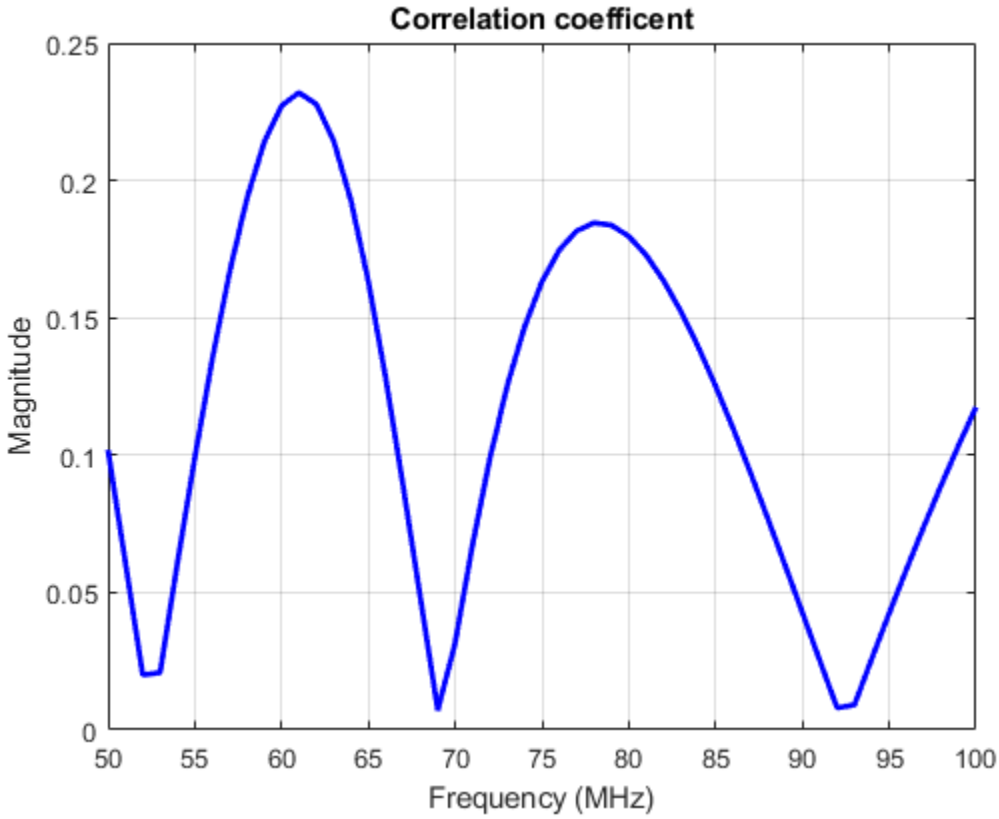
`rho = correlation(array,frequency,elem1,elem2,z0)` returns the correlation coefficient between two antenna elements, `elem1` and `elem2` of an array.

Examples

Plot Correlation of Array

Plot the correlation between 1 and 2 antenna elements in a default linear array over a frequency range of 50MHz to 100MHz.

```
h = linearArray;
correlation (h,50e6:1e6:100e6,1,2);
```



Calculate Correlation Coefficient of Array

Calculate correlation coefficient of default rectangular array at a frequency range of 50MHz to 100MHz.

```
h = rectangularArray;
rho = correlation (h, 50e6:1e6:100e6, 1, 2)
```

```
rho = 51×1

    0.1419
```

```
    0.1124
    0.0828
    0.0526
    0.0218
    0.0099
    0.0426
    0.0759
    0.1091
    0.1406
      ⋮
```

Input Arguments

`array` — Array object

scalar handle

Array object, specified as a scalar handle.

`frequency` — Frequency range used to calculate correlation

vector in Hz

Frequency range used to calculate correlation, specified as a vector in Hz.

Example: 50e6:1e6:100e6

Data Types: `double`

`elem1,elem2` — Antenna elements in an array

scalar handle

Antenna elements in an array, specified as a scalar handle.

`z0` — Reference impedance

50 (default) | scalar in ohms

Reference impedance, specified as a scalar in ohms.

Example: 70

Data Types: `double`

Output Arguments

`rho` — Correlation coefficient between two antenna elements of an array
vector

Correlation coefficient between two antenna elements of an array, returned as a vector.

See Also

`impedance` | `returnLoss` | `sparameters`

Introduced in R2015a

cylinder2strip

Cylinder equivalent width approximation

Syntax

```
w = cylinder2strip(r)
```

Description

`w = cylinder2strip(r)` calculates the equivalent width of a strip approximation for a cylinder cross section.

Examples

Calculate Cylinder to Strip Approximation

Calculate the width of the strip approximation to a cylinder of radius 20 mm.

```
w = cylinder2strip(20e-3)
```

```
w = 0.0800
```

Input Arguments

`r` — Cylindrical cross-section radius

scalar in meters | vector in meters

Cylindrical cross-section radius, specified as a scalar or vector in meters.

Example: 20e-3

Output Arguments

`w` — Equivalent width of strip
scalar | vector

Equivalent width of strip, returned as a scalar or vector.

See Also

`helixpitch2spacing`

Introduced in R2015a

helixpitch2spacing

Spacing between turns of helix

Syntax

```
s = helixpitch2spacing(a,r)
```

Description

`s = helixpitch2spacing(a,r)` calculates the spacing between the turns of a helix antenna given the pitch angle, `a`, and the radius of the helix, `r`.

Examples

Calculate Spacing Between Helix Turns

Calculate spacing for helix with pitch varying from 12 degrees to 14 degrees in steps of 0.5 and 20 mm radius.

```
s = helixpitch2spacing(12:0.5:14,20e-3)
```

```
s = 1×5

    0.0267    0.0279    0.0290    0.0302    0.0313
```

Calculate Spacing for Helix with Varying Pitch

Calculate spacing for helix with pitch varying from 12 degrees to 14 degrees in steps of 0.5 and radius 20 mm.

```
s = helixpitch2spacing(12:0.5:14,20e-3)
```

```
s = 1×5

    0.0267    0.0279    0.0290    0.0302    0.0313
```

Calculate Spacing of Helix Antenna with Varying Radius

Calculate spacing of a helix that has a pitch of 12 degrees and a radius that varies from 20 mm to 22 mm in steps of 0.5 mm.

```
s = helixpitch2spacing(12,20e-3:0.5e-3:22e-3)
```

```
s = 1×5

    0.0267    0.0274    0.0280    0.0287    0.0294
```

Calculate Spacing of Helix with Varying Pitch and Radius

Calculate spacing for helix with pitch varying from 12 degrees to 14 degrees in steps of 0.5 and radius varying from 20mm to 22mm in steps of 0.5.

```
s = helixpitch2spacing(12:0.5:14,20e-3:0.5e-3:22e-3)
```

```
s = 1×5

    0.0267    0.0286    0.0305    0.0324    0.0345
```

Input Arguments

a — Pitch angle of helix

scalar in meters | vector in meters

Pitch angle of helix, specified as a scalar or vector in meters.

Example: 12:0.5:14

`r` — Radius of helix
scalar in meters | vector in meters

Radius of helix, specified as a scalar or vector in meters.

Example: 20e-3

Note If the pitch angle and radius are both vectors, then their lengths must be equal.

Output Arguments

`s` — Spacing between helix turns
scalar in meters | vector in meters

Spacing between helix turns, returned as a scalar or vector in meters.

See Also

`cylinder2strip`

Introduced in R2015a

meshconfig

Change mesh mode of antenna structure

Syntax

```
meshconfig(antenna,mode)
```

Description

`meshconfig(antenna,mode)` changes the meshing mode of the antenna according to the text input mode.

Examples

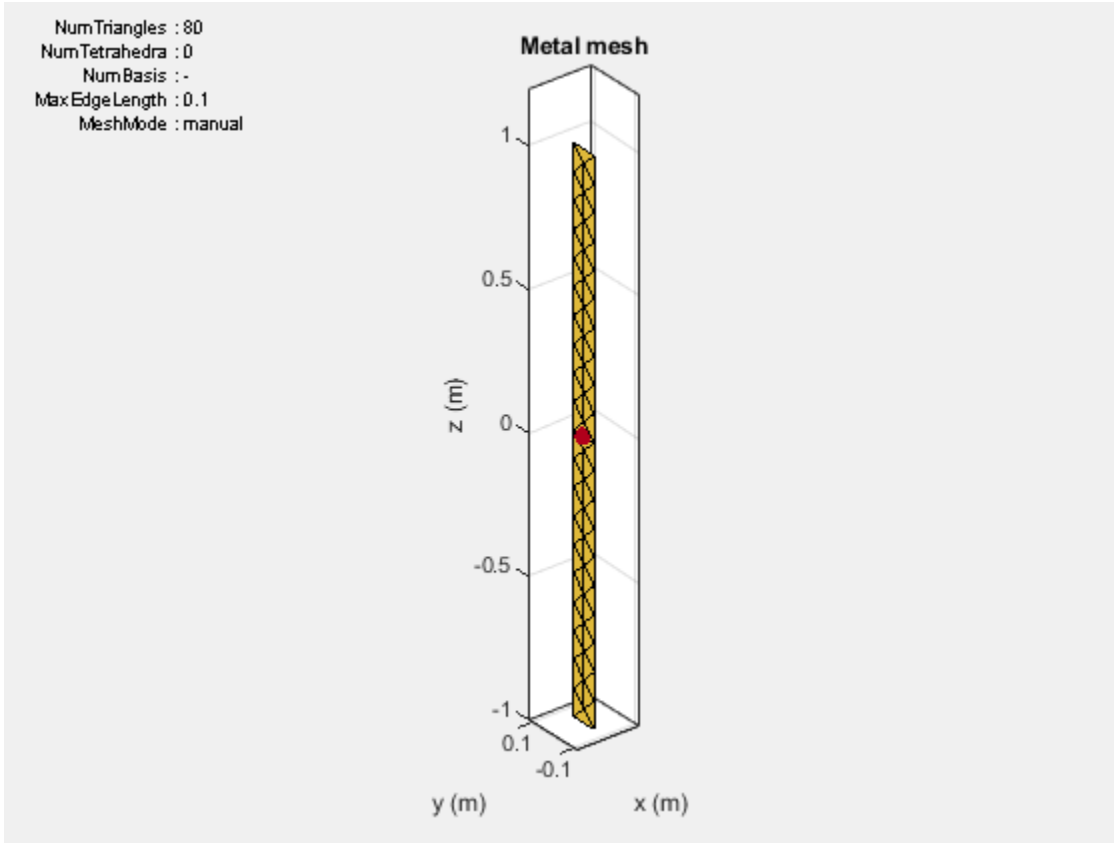
Change Mesh Configuration of Antenna

Change the mesh configuration of a dipole antenna from auto (default) to manual mode.

```
h = dipole;
meshconfig(h,'manual')
```

```
ans = struct with fields:
     NumTriangles: 0
    NumTetrahedra: 0
         NumBasis: []
    MaxEdgeLength: []
         MeshMode: 'manual'
```

```
mesh(h,'MaxEdgeLength',0.1)
```



Input Arguments

`antenna` — Antenna object
scalar handle

Antenna object, specified as a scalar handle.

`mode` — Meshing mode
`'auto'` (default) | `'manual'`

Meshing mode, specified as `'auto'` or `'manual'`.

Data Types: `char`

See Also

`mesh` | `show`

Introduced in R2015a

numSummationTerms

Change number of summation terms for calculating periodic Green's function

Syntax

```
numSummationTerms(array,num)
```

Description

`numSummationTerms(array,num)` changes the number of summation terms used to calculate periodic Green's function of the infinite array. This method calculates

$2*num+1$ of the periodic Green's function. The summation is carried out from –`num` to +`num`. A higher number of terms results in better accuracy but increases the overall computation time.

Input Arguments

`array` — Infinite array
scalar handle

Infinite array, specified as a scalar handle.

`num` — Number to calculate summation terms
10 (default) | scalar

Number to calculate summation terms, specified as a scalar. The summation is carried out from –`num` to +`num`.

Example: `50`

Examples

Change Number of Summation Terms in Infinite Array

Create an infinite array with the scan elevation at 45 degrees. Calculate the scan impedance. By default, the number of summation terms used is 21.

```
h = infiniteArray('ScanElevation',45);
s = impedance(h,1e9)
```

```
s = 83.3054 + 68.7805i
```

Change the number of summation terms to 51. Calculate the scan impedance again.

```
numSummationTerms(h,25)
s = impedance(h,1e9)
```

```
s = 83.4476 + 68.8165i
```

Change the number of terms to 101. Increasing the number of summation terms results in a more accurate scan impedance. However, the time required to calculate the scan impedance increases.

```
numSummationTerms(h,50)
s = impedance(h,1e9)
```

```
s = 83.4919 + 68.8217i
```

See Also

`beamwidth` | `pattern`

Topics

“Infinite Arrays”

Introduced in R2015b

feedCurrent

Calculate current at feed for antenna or array

Syntax

```
feedCurrent(obj,frequency)
```

Description

`feedCurrent(obj,frequency)` calculates the current at the feed for an antenna or array object at a specified frequency. The feed current when multiplied by the antenna impedance gives the voltage across the antenna.

Examples

Feed Current of Monopole Antenna Excited By Plane Wave.

Excite a monopole antenna using plane wave. Calculate the feed current at 75 MHz.

```
h = planeWaveExcitation('Element',monopole, 'Direction',[1 0 0])
cur = feedCurrent(h,75e6)
```

```
h =

  planeWaveExcitation with properties:

         Element: [1×1 monopole]
       Direction: [1 0 0]
    Polarization: [0 0 1]


cur =
```

```
   0.0132 - 0.0133i
```

Feed Current of Rounded-Bowtie Antenna

Calculate the feed current of a rounded-bowtie designed for operation at 2.4 GHz.

```
b  = design(bowtieRounded,2.4e9);
If = feedCurrent(b,2.4e9)
```

```
If = 0.0294 - 0.0012i
```

Input Arguments

`obj` — Antenna or array object

object handle

Antenna or array object, specified as an object handle.

`frequency` — Frequency to calculate feed current

scalar integer in Hz

Frequency to calculate feed current, specified as a scalar integer in Hz.

See Also

`current`

Introduced in R2017a

fieldsCustom

Plot electric or magnetic fields of antenna

Syntax

```
fieldsCustom(fields,points)
fieldsCustom(fields,points,scalefield)
qobj = fieldsCustom( ___ )

fieldsCustom(axeshandle, ___ )
```

Description

`fieldsCustom(fields,points)` plots electric or magnetic field vectors, `fields`, at specified points in space, `points`, in the current axes.

`fieldsCustom(fields,points,scalefield)` scales the field arrows by a scalar value, `scalefield`.

`qobj = fieldsCustom( ___ )` returns the quiver object, using either of the previous syntaxes.

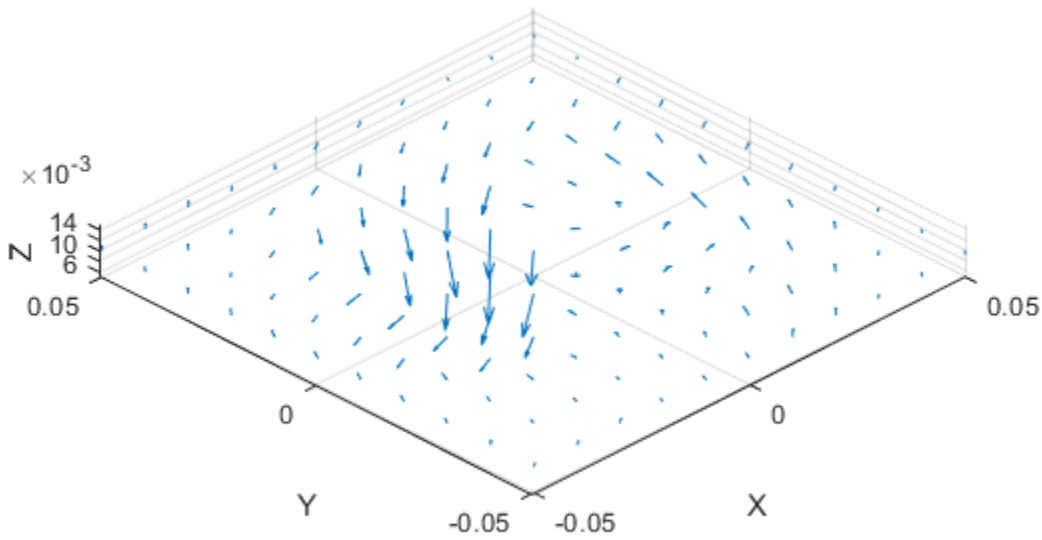
`fieldsCustom(axeshandle, ___ )` plots into the axes specified by `axeshandle` instead of the current axes.

Examples

Visualize Magnetic Field of Antenna Using `fieldsCustom`

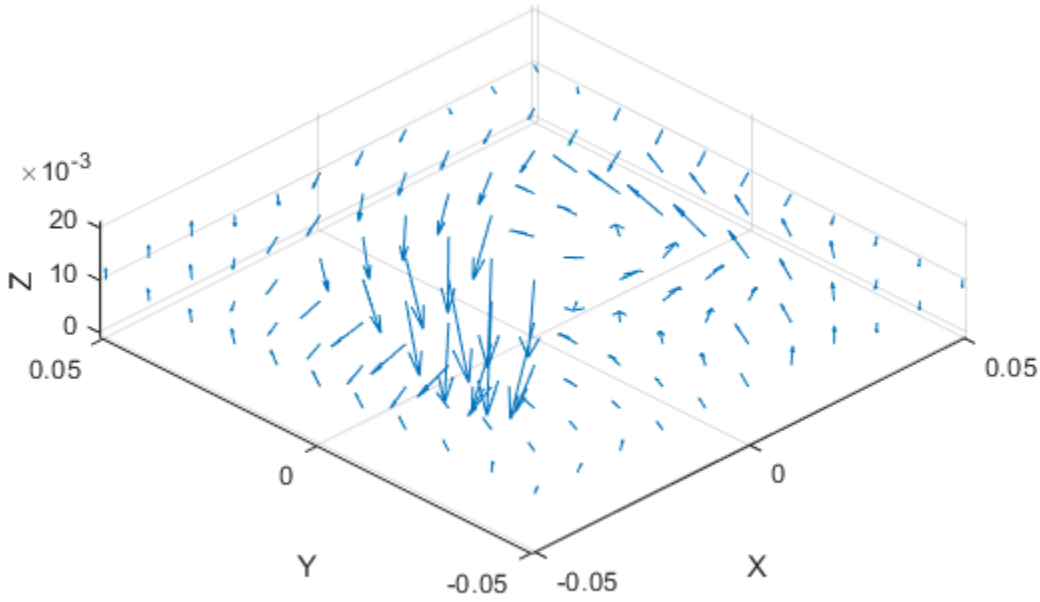
Load and visualize the magnetic field data available in the file '`fielddata.mat`'.

```
load fielddata
fieldsCustom(H,p)
```



Scale the magnetic field arrows by a factor of 2.

```
figure
fieldsCustom(H,p,2)
```



Input Arguments

`fields` — Electric or magnetic field vectors

3-by-*p* complex matrix

Electric or magnetic field vectors, specified as a 3-by-*p* complex matrix. *p* is the number of points in space.

Data Types: `double`

`points` — x, y, z coordinates in space

3-by-*p* real matrix

`x`, `y`, `z` coordinates in space, specified as a 3-by-*p* real matrix. *p* is the number of points in space.

Data Types: `double`

`axeshandle` — Axes object
object handle

Axes object, specified as an object handle.

Data Types: `char`

`scalefield` — Value by which to scale field arrows
0.9 (default) | scalar

Value by which to scale the field arrows, specified as a scalar. A value of `2` doubles the relative length of the field arrows. A value of `0.5` halves the length of the field arrows. A value of `0` plots the field arrows without automatic scaling.

Example: 2

Data Types: `double`

Output Arguments

`qobj` — Electric or magnetic field plot
quiver object handle

Electric or magnetic field plot, returned as quiver object handle.

See Also

`EHfields` | `pattern` | `patternCustom`

Introduced in R2016a

patternCustom

Plot radiation pattern

Syntax

```
patternCustom(magE,theta,phi)
patternCustom(magE,theta,phi,Name,Value)
hplot = patternCustom( ___ )
```

Description

`patternCustom(magE,theta,phi)` plots the 3-D radiation pattern of an antenna magnitude, `magE` over the specified `phi` and `theta` angle vectors.

`patternCustom(magE,theta,phi,Name,Value)` uses additional options specified by one or more `Name,Value` pair arguments.

`hplot = patternCustom( ___ )` returns handles of the lines or surface in the figure window. This syntax accepts any combination of arguments from the previous syntaxes

Examples

Visualize 3-D Electric Field Pattern of Dipole

Calculate the magnitude, azimuth, and elevation angles of a dipole's electric field at 75 MHz.

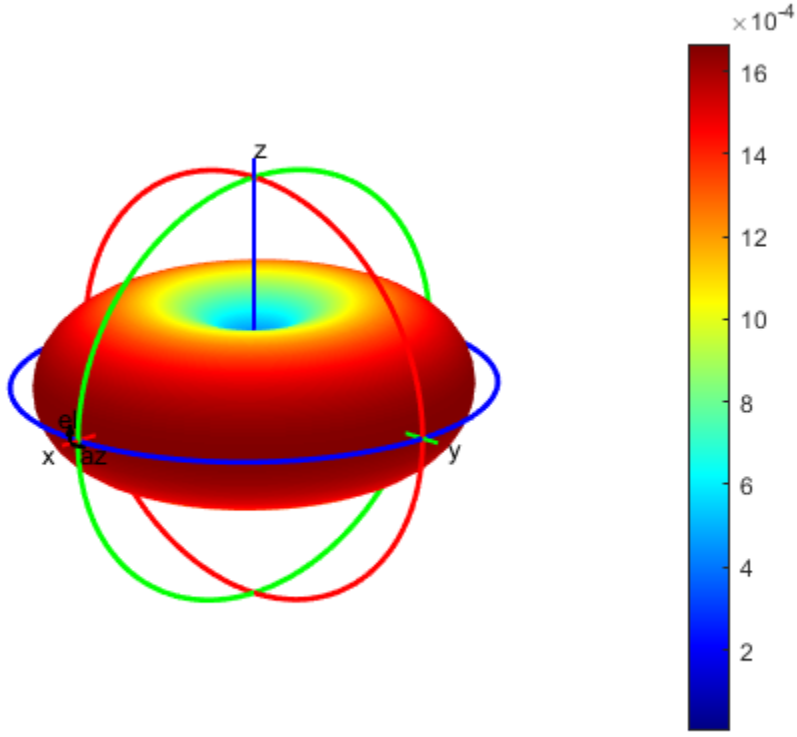
```
d = dipole;
[efield,az,el] = pattern(d, 75e6,'Type','efield');
```

Extract the theta and phi angles of the electric field magnitude of the antenna.

```
phi = az';
theta = (90-el);
MagE = efield';
```

Plot the 3-D electric field pattern.

```
patternCustom(MagE,theta,phi);
```



Visualize 2-D Radiation Patterns of Helix Directivity

Calculate the magnitude, azimuth, and elevation angles of a helix's directivity at 2 GHz.

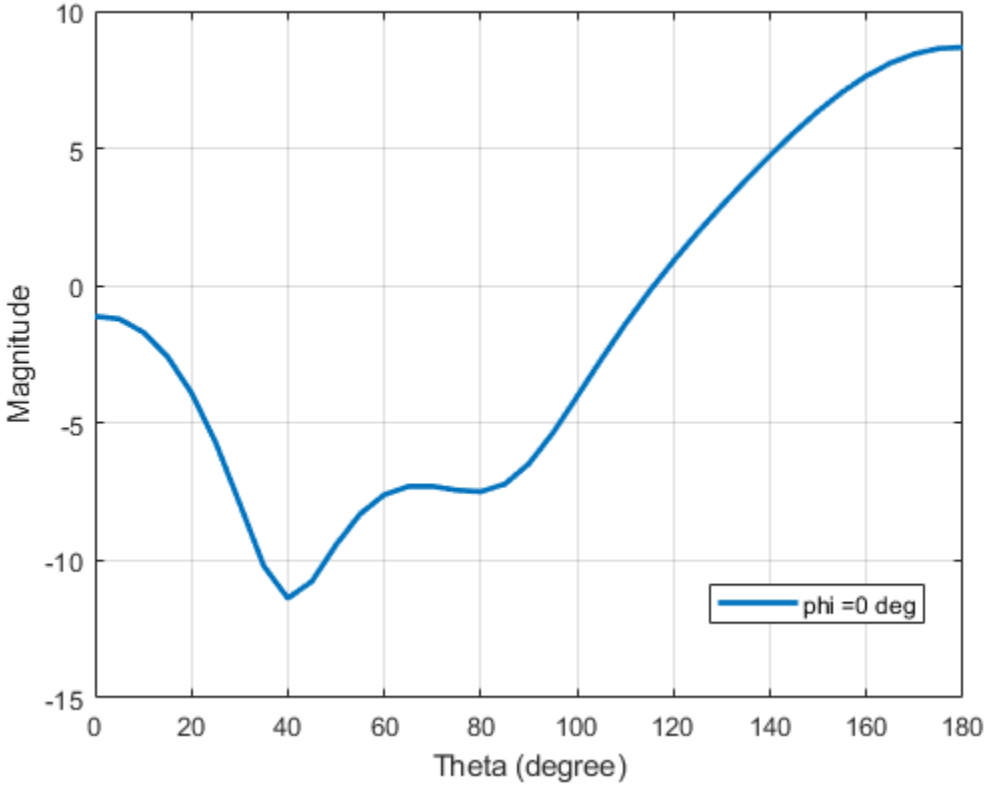
```
h = helix;
[D,az,el] = pattern(h,2e9);
```

Extract theta and phi angles of the directivity magnitude.

```
phi = az';
theta = (90-el);
MagE = D';
```

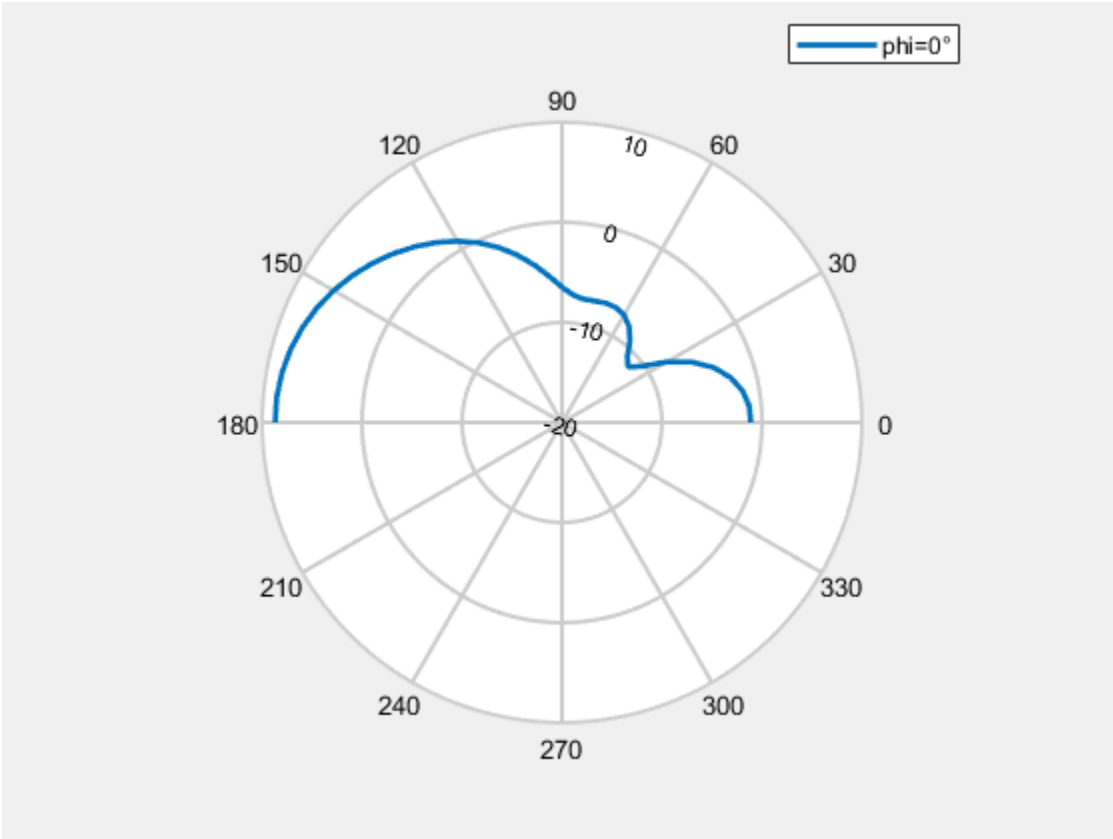
Plot 2-D phi slice of the antenna in rectangular coordinates.

```
figure;
patternCustom(MagE,theta,phi,'CoordinateSystem','rectangular',...
    'Slice','phi','SliceValue',0);
```



Plot 2-D phi slice of the antenna in polar coordinates.

```
figure;
patternCustom(MagE, theta, phi,'CoordinateSystem','polar',...
    'Slice','phi','SliceValue',0);
```



Visualize Radiation Patterns from Antenna Data File

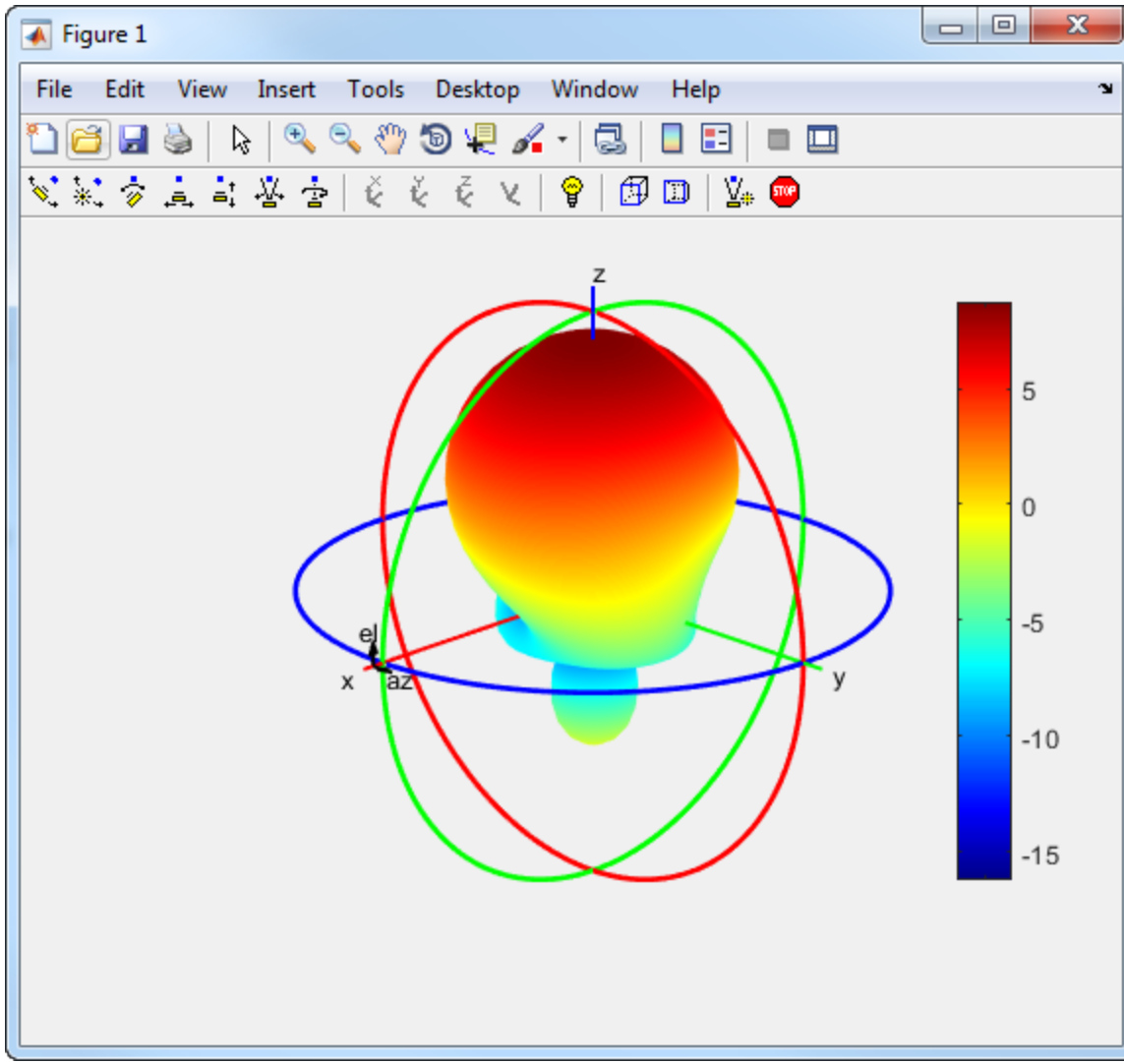
Consider a helix antenna data file in `.csv` format. This file contains the magnitude of the antenna directivity in `phi` and `theta` angles. Read the file .

Read the `.csv` data file.

```
helixdata = csvread('antennadata_test.csv',1,0);
```

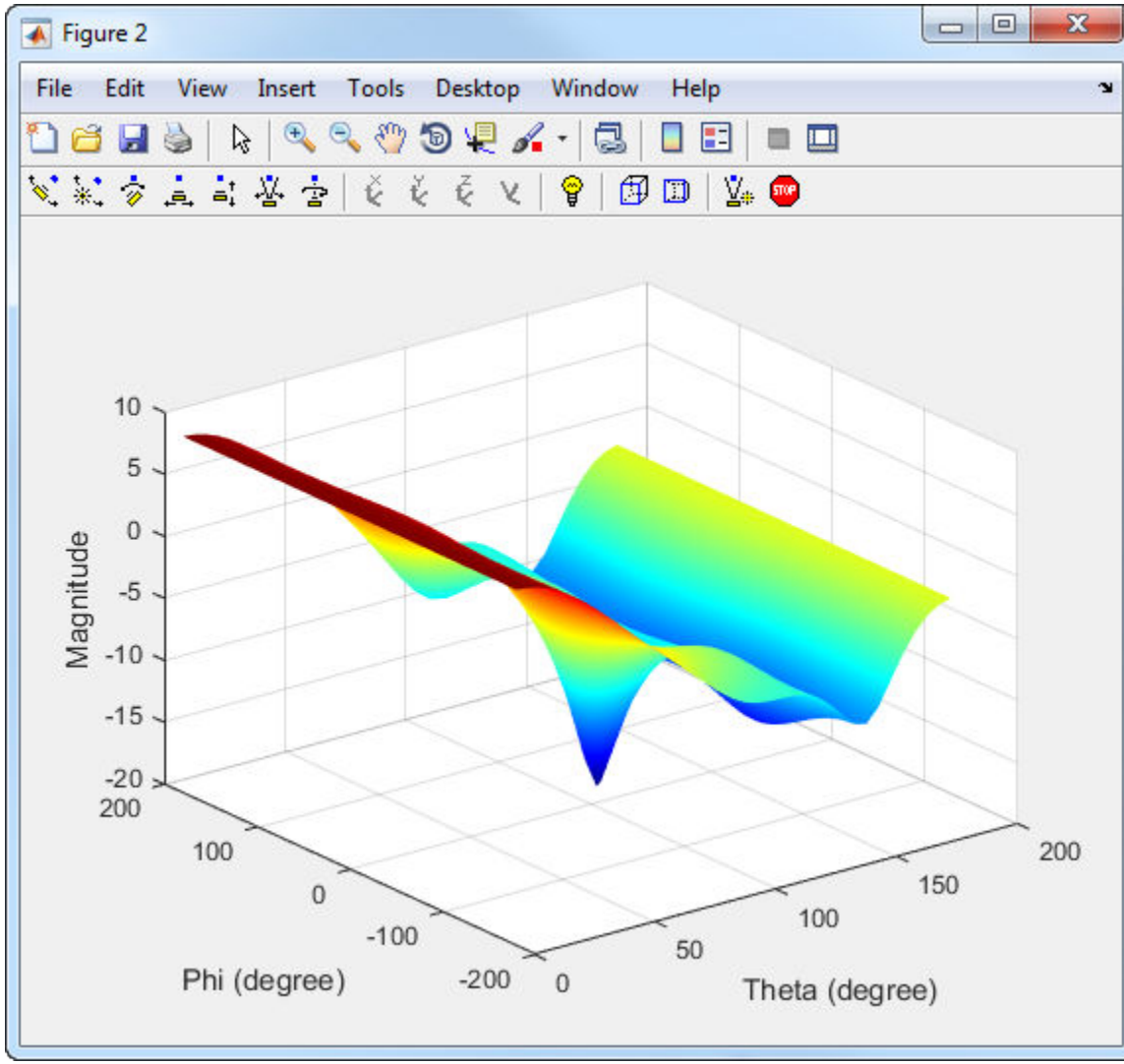
Use `patternCustom` to extract the magnitude of directivity, and the `phi`, and `theta` angle values. Plot the 3-D polar radiation pattern.

```
patternCustom(helixdata(:,3),helixdata(:,2),helixdata(:,1));
```



Use the same data to plot the 3-D rectangular radiation pattern.

```
figure
patternCustom(helixdata(:,3),helixdata(:,2),helixdata(:,1),...
 'CoordinateSystem','rectangular');
```



Input Arguments

magE — Magnitude of plotted quantity
real vector | matrix

Magnitude of plotted quantity, specified as one of the following:

- A *N*-by-1 real vector . *N* is the same size as the `phi` and `theta` angle vectors.
- A *M*-by-*R* matrix. The matrix should be the same size as `phixtheta`.

where `theta` and `phi` angles are in the spherical coordinate system specified as a vector.

Data quantities plotted include directivity, E-fields, H-fields, or power of an antenna or array object.

Data Types: `double`

`theta` — Theta angles in spherical coordinates
vector in degrees

Theta angles in spherical coordinates, specified as a vector in degrees.

Data Types: `double`

`phi` — Phi angles in spherical coordinates
vector in degrees

Phi angles in spherical coordinates, specified as a vector in degrees.

Data Types: `double`

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `'CoordinateSystem','rectangular'`

`CoordinateSystem` — Coordinate system of radiation pattern
`'polar'` (default) | `'rectangular'`

Coordinate system of radiation pattern, specified as the comma-separated pair consisting of `'CoordinateSystem'` and one of these values: `'polar'`, `'rectangular'`.

Example: `'CoordinateSystem','polar'`

Data Types: `char`

`Slice` — Plane to visualize 2-D data
`'theta'` | `'phi'`

Plane to visualize 2-D data, specified as a comma-separated pair consisting of `'Slice'` and `'theta'` or `'phi'`.

Example: `'Slice','phi'`

Data Types: `char`

`SliceValue` — Angle values for slice
scalar | vector

Angle values for slice, specified as a comma-separated pair consisting of `'SliceValue'` and a scalar or a vector.

Output Arguments

`hplot` — Lines or surfaces in figure window
object handle

Lines or surfaces in figure window, returned as object handle.

See Also

`EHfields` | `fieldsCustom` | `pattern` | `polarpattern`

Introduced in R2016a

msiread

Read MSI planet antenna file

Syntax

```
msiread(fname)
[horizontal] = msiread(fname)
[horizontal,vertical] = msiread(fname)
[horizontal,vertical,optional] = msiread(fname)
```

Description

`msiread(fname)` reads an MSI planet antenna file in `.pln`, or `.msi` formats.

`[horizontal] = msiread(fname)` reads the file and returns a structure containing horizontal gain data.

`[horizontal,vertical] = msiread(fname)` reads the file and returns structures containing horizontal and vertical gain data.

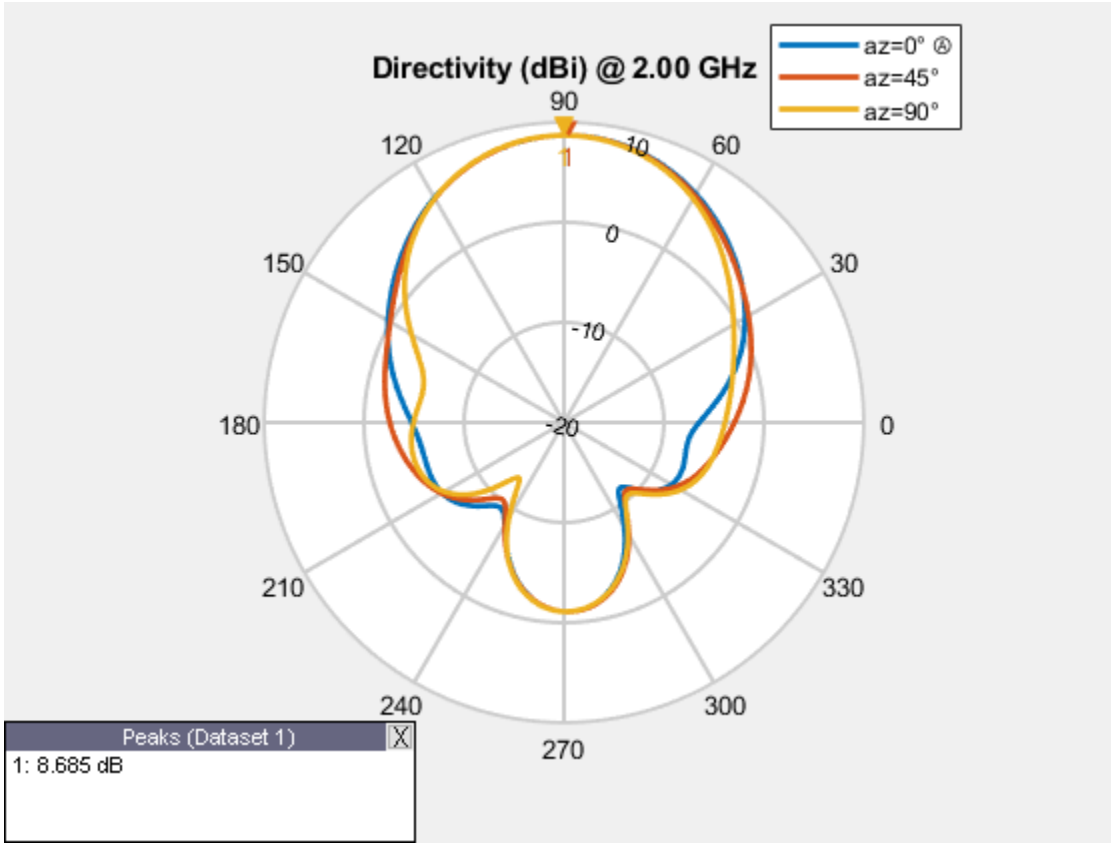
`[horizontal,vertical,optional] = msiread(fname)` reads the file and returns structures containing horizontal gain data, vertical gain data, and all additional data in the file.

Examples

Write and Read MSI Antenna Data File

Create a helix antenna and plot the elevation pattern at 2 GHz.

```
h = helix;
patternElevation(h,2e9,[0 45 90],'Elevation',0:1:360);
```



Write the elevation pattern of the helix antenna in an MSI Planet Antenna file.

```
msiwrite(h,2e9,'helix','Name','Helix Antenna Specifications')
```

The msiwrite function saves a file named `helix.pln` to the default MATLAB™ folder.

```
NAME Helix Antenna Specifications
FREQUENCY 2000.0
GAIN 8.74 dBi
HORIZONTAL 360
0.00 13.56
1.00 13.48
2.00 13.39
3.00 13.30
```

```
4.00 13.22
5.00 13.13
```

Read the MSI antenna data file created.

```
msiread helix.pln
```

```
ans = struct with fields:
    PhysicalQuantity: 'Gain'
           Magnitude: [360x1 double]
               Units: 'dBi'
             Azimuth: [360x1 double]
           Elevation: 0
           Frequency: 2.0000e+09
               Slice: 'Elevation'
```

Read Horizontal, Vertical and Optional Data from Antenna File

Read horizontal, vertical and optional data from the antenna data file **Test_file_demo.pln**.

```
[Horizontal,Vertical,Optional] = msiread('Test_file_demo.pln')
```

```
Horizontal = struct with fields:
    PhysicalQuantity: 'Gain'
           Magnitude: [360x1 double]
               Units: 'dBd'
             Azimuth: [360x1 double]
           Elevation: 0
           Frequency: 659000000
               Slice: 'Elevation'


Vertical = struct with fields:
    PhysicalQuantity: 'Gain'
           Magnitude: [360x1 double]
               Units: 'dBd'
             Azimuth: 0
           Elevation: [360x1 double]
           Frequency: 659000000
               Slice: 'Azimuth'
```

```
Optional = struct with fields:
             name: 'Sample.pln'
             make: 'Sample 4DR-16-2HW'
        frequency: 659000000
          h_width: 180
          v_width: 7.3000
    front_to_back: 34
             gain: [1x1 struct]
             tilt: 'MECHANICAL'
     polarization: 'POL_H'
          comment: 'Ch-45 0 deg dt'
     scaling_mode: 'AUTOMATIC'
```

Input Arguments

`fname` — Name of MSI file

character vector

Name of MSI file, specified as a character vector. The files must be a `.pln` or `.msi` format.

Output Arguments

`horizontal` — Horizontal gain data

structure

Horizontal gain data, returned as a structure containing the following fields:

- `PhysicalQuantity` — Quantity specified in the MSI file, returned as one of the values: `'E-field'`, `'H-field'`, `'directivity'`, `'power'`, `'powerdB'`, or `'Gain'`.
- `Magnitude` — Magnitude values of the quantity specified in the MSI file, returned as a real vector of size *N*–by–`1` where *N* is same size as `theta` and `phi` angles.
- `Units` — Units of the quantity specified in the MSI file, returned as one of the values: `'dBi'`, `'dB'`, `'V/m'`, `'watts'`, or `'dBd'`.
- `Azimuth` — Azimuth angles specified in the MSI file, returned as a scalar or a vector in degrees.

- `Elevation` — Elevation angles specified in the MSI file, returned as a scalar or a vector in degrees.
- `Frequency` — Frequency specified in the MSI file, returned as a scalar or a vector in Hertz.
- `Slice` — Type of data set variation, returned as text. The variations are `'Azimuth'` or `'Elevation'`.

`vertical` — Vertical gain data
structure

Vertical gain data, returned as a structure containing the following fields:

- `PhysicalQuantity` — Quantity specified in the MSI file, returned as one of the values: `'E-field'`, `'H-field'`, `'directivity'`, `'power'`, `'powerdB'`, or `'Gain'`.
- `Magnitude` — Magnitude values of the quantity specified in the MSI file, returned as a real vector of size *N*–by–`1` where *N* is same size as `theta` and `phi` angles.
- `Units` — Units of the quantity specified in the MSI file, returned as one of the values: `'dBi'`, `'dB'`, `'V/m'`, `'watts'`, or `'dBd'`.
- `Azimuth` — Azimuth angles specified in the MSI file, returned as a scalar or a vector in degrees.
- `Elevation` — Elevation angles specified in the MSI file, returned as a scalar or a vector in degrees.
- `Frequency` — Frequency specified in the MSI file, returned as a scalar or a vector in Hertz.
- `Slice` — Type of data set variation, returned as text. The variations are `Azimuth` or `Elevation`.

`optional` — Additional data
structure

Additional data, returned as a structure containing (but not limited to): `Name`, `Make`, `Frequency`, `H_width`, `V_width`, `Front_to_back`, `Gain`, `Tilt`, `Polarization`, `Comment`.

See Also

`msiwrite`

Topics

“Read, Visualize and Write MSI Planet Antenna Files”

Introduced in R2016a

msiwrite

Write data in MSI planet antenna file format

Syntax

```
msiwrite(fname,dataslice1,dataslice2)
msiwrite(fname,dataslice1,dataslice2,optional)

msiwrite(objname,frequency,fname)
msiwrite(objname,frequency,fname,Name,Value)
```

Description

`msiwrite(fname,dataslice1,dataslice2)` writes the data from structures `dataSlice1` and `dataSlice2` to an MSI planet antenna file called `fname`.

`msiwrite(fname,dataslice1,dataslice2,optional)` writes the data from structures `dataSlice1`, `dataSlice2`, and `optional` to an MSI planet antenna file called `fname`.

`msiwrite(objname,frequency,fname)` writes calculated data of an antenna or array object at a specified frequency to an MSI planet antenna file called `fname`.

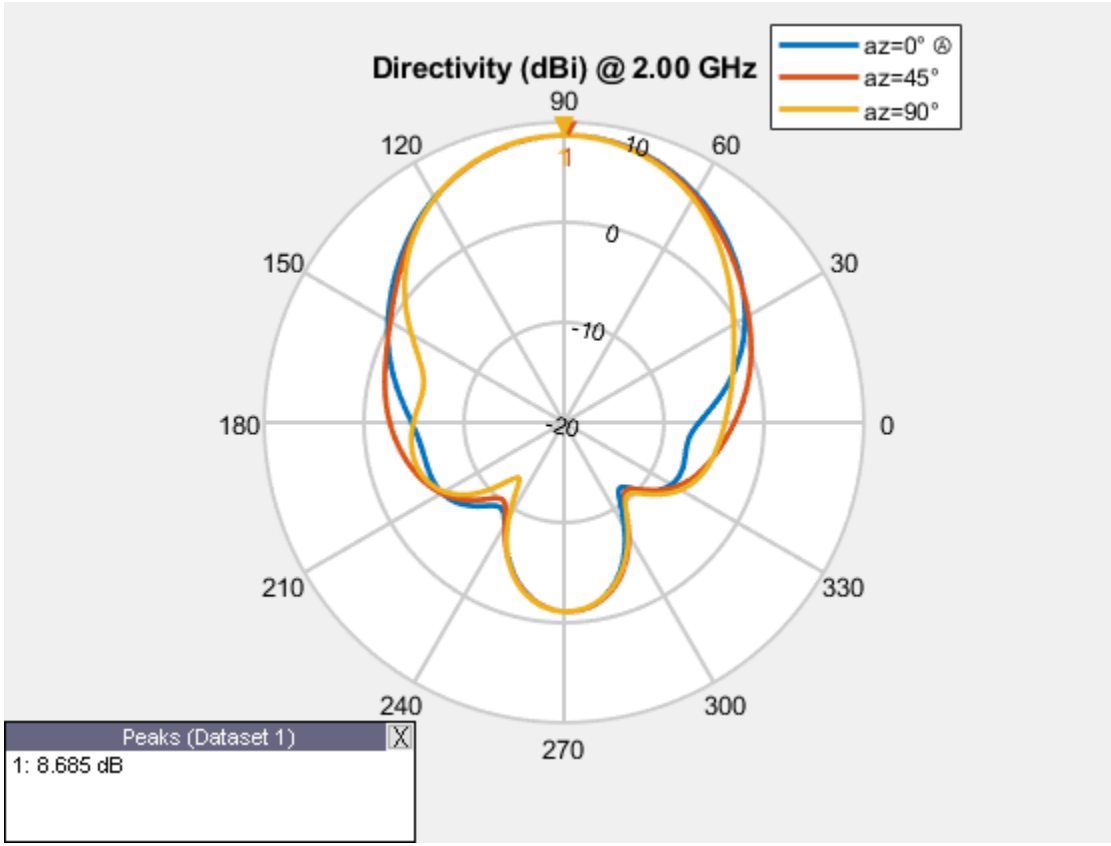
`msiwrite(objname,frequency,fname,Name,Value)` uses additional options specified by one or more `Name,Value` pair arguments.

Examples

Write and Read MSI Antenna Data File

Create a helix antenna and plot the elevation pattern at 2 GHz.

```
h = helix;
patternElevation(h,2e9,[0 45 90],'Elevation',0:1:360);
```



Write the elevation pattern of the helix antenna in an MSI Planet Antenna file.

```
msiwrite(h,2e9,'helix','Name','Helix Antenna Specifications')
```

The msiwrite function saves a file named `helix.pln` to the default MATLAB™ folder.

```
NAME Helix Antenna Specifications
FREQUENCY 2000.0
GAIN 8.74 dBi
HORIZONTAL 360
0.00 13.56
1.00 13.48
2.00 13.39
3.00 13.30
```

```
4.00 13.22
5.00 13.13
```

Read the MSI antenna data file created.

```
msiread helix.pln
```

```
ans = struct with fields:
    PhysicalQuantity: 'Gain'
           Magnitude: [360x1 double]
               Units: 'dBi'
             Azimuth: [360x1 double]
           Elevation: 0
           Frequency: 2.0000e+09
               Slice: 'Elevation'
```

Input Arguments

`fname` — Name of MSI file

`.pln` (default) | character vector

Name of MSI file, specified as a character vector. By default, `msiwrite` writes the MSI planet antenna file that has a `.pln` format.

`dataslice1` — Horizontal or vertical gain data

structure

Horizontal or vertical gain data, specified as a structure containing the following fields:

- `PhysicalQuantity` — Measured quantity in the MSI file: `E-field`, `H-field`, `directivity`, `power`, `powerdB`, or, `gain`.
- `Magnitude` — Magnitude values of the measured quantity.
- `Units` — Units of the measured quantity.
- `Azimuth` — Azimuth angles.
- `Elevation` — Elevation angles.
- `Frequency` — Frequency of operation.
- `Slice` — Type of data set variation: `Azimuth`, or `Elevation`.

`dataslice2` — Horizontal or vertical gain data
structure

Horizontal or vertical gain data, specified as a structure containing the following fields:

- `PhysicalQuantity` — Measured quantity in the MSI file: `E-field`, `H-field`, `directivity`, `power`, `powerdB`, or, `gain`.
- `Magnitude` — Magnitude values of the measure quantity.
- `Units` — Units of the measured quantity.
- `Azimuth` — Azimuth angles.
- `Elevation` — Elevation angles.
- `Frequency` — Frequency of operation.
- `Slice` — Type of data set variation: `Azimuth`, or `Elevation`.

`optional` — Additional data
structure

Additional data, specified as a structure containing the following fields: `Name`, `Make`, `Frequency`, `H_width`, `V_width`, `Front_to_back`, `Gain`, `Tilt`, `Polarization`, `Comment`.

`objname` — Antenna or array object
antenna or array handle

Antenna or array object, specified as an antenna or array handle.

`frequency` — Frequency of operation of antenna or array object
positive numeric scalar

Frequency of operation of antenna or array object, specified as a positive numeric scalar.

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `'Comment','horn antenna'`

`Name` — Title of file
character vector

Title of file in the first line, specified as the comma-separated pair consisting of `'Name'` and a character vector.

Example: `'Name','Designed Helix Antenna in MATLAB'`

Data Types: `char`

`Comment` — Comments about antenna or array data file
character array

Comments about an antenna or array data file, specified as the comma-separated pair consisting of `'Comment'` and a character array.

Example: `'Comment','This antenna is for space simulations.'`

Data Types: `char`

See Also
`msiread`

Topics
“Read, Visualize and Write MSI Planet Antenna Files”

Introduced in R2016a

dielectric

Dielectric material for use as substrate

Syntax

```
d = dielectric(material)
d = dielectric(Name,Value)
```

Description

`d = dielectric(material)` returns dielectric materials for use as a substrate in antenna elements.

`d = dielectric(Name,Value)` returns dielectric materials, based on the properties specified by one or more `Name,Value` pair arguments.

Examples

PIFA Antenna with Dielectric Substrate

Use a Teflon dielectric material as a substrate for a PIFA antenna. View the antenna.

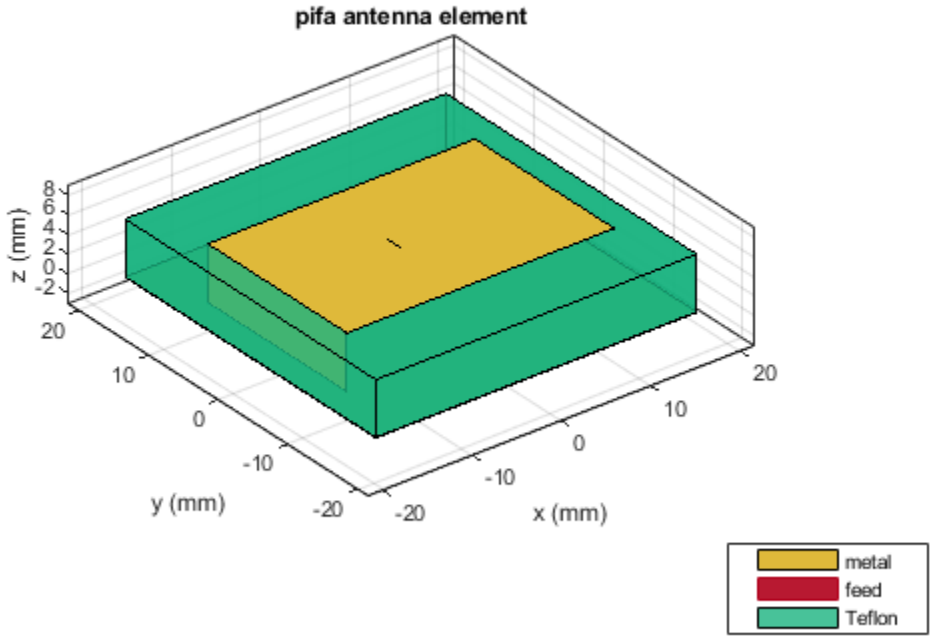
```
d = dielectric('Teflon')
```

```
d =
  dielectric with properties:

           Name: 'Teflon'
       EpsilonR: 2.1000
    LossTangent: 2.0000e-04
      Thickness: 0.0060

For more materials see catalog
```

```
p = pifa('Height',0.0060,'Substrate',d)
```

```
p =
  pifa with properties:

              Length: 0.0300
               Width: 0.0200
              Height: 0.0060
           Substrate: [1x1 dielectric]
    GroundPlaneLength: 0.0360
     GroundPlaneWidth: 0.0360
    PatchCenterOffset: [0 0]
        ShortPinWidth: 0.0200
           FeedOffset: [-0.0020 0]
                 Tilt: 0
            TiltAxis: [1 0 0]
                Load: [1x1 lumpedElement]
```

```
show(p)
```



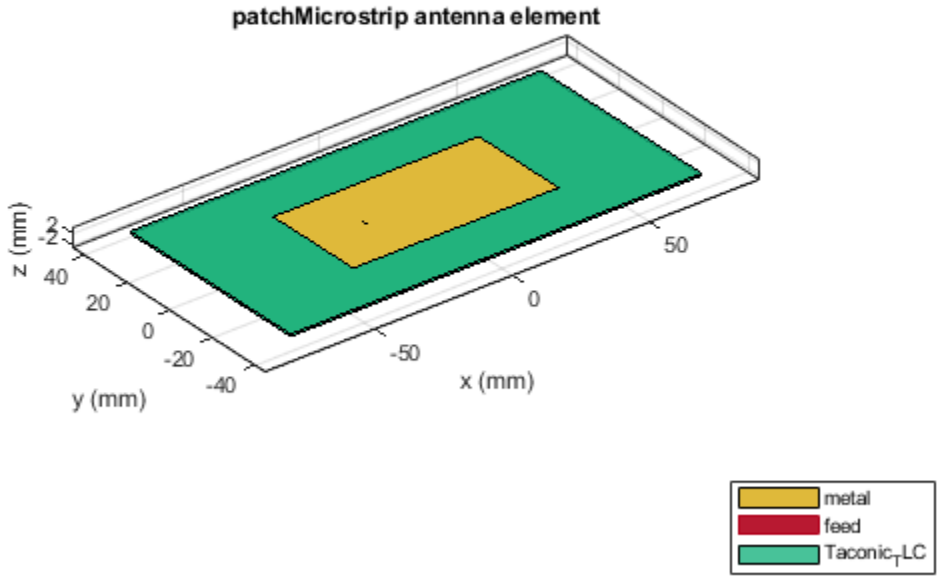
Custom Dielectric Properties

Create a patch microstrip antenna using a substrate with a relative permittivity of 2.70, a loss tangent of 0.002 and a thickness of 0.0008 m. View the antenna.

```
t = dielectric('Name','Taconic_TLC','EpsilonR',2.70,'LossTangent',0.002,...
    'Thickness',0.0008);
p = patchMicrostrip('Height',0.0008,'Substrate',t)

p =
  patchMicrostrip with properties:
```

```
               Length: 0.0750
                Width: 0.0375
               Height: 8.0000e-04
            Substrate: [1x1 dielectric]
    GroundPlaneLength: 0.1500
     GroundPlaneWidth: 0.0750
    PatchCenterOffset: [0 0]
           FeedOffset: [-0.0187 0]
                 Tilt: 0
              TiltAxis: [1 0 0]
                 Load: [1x1 lumpedElement]
```

```
show(p)
```



Patch Antenna with Air Gap between Groundplane and Dielectric

Create a microstrip patch antenna.

```
p = patchMicrostrip;
```

```
For properties of air and teflon dielectrics use Dielectric Catalog.
```

```
openDielectricCatalog
```

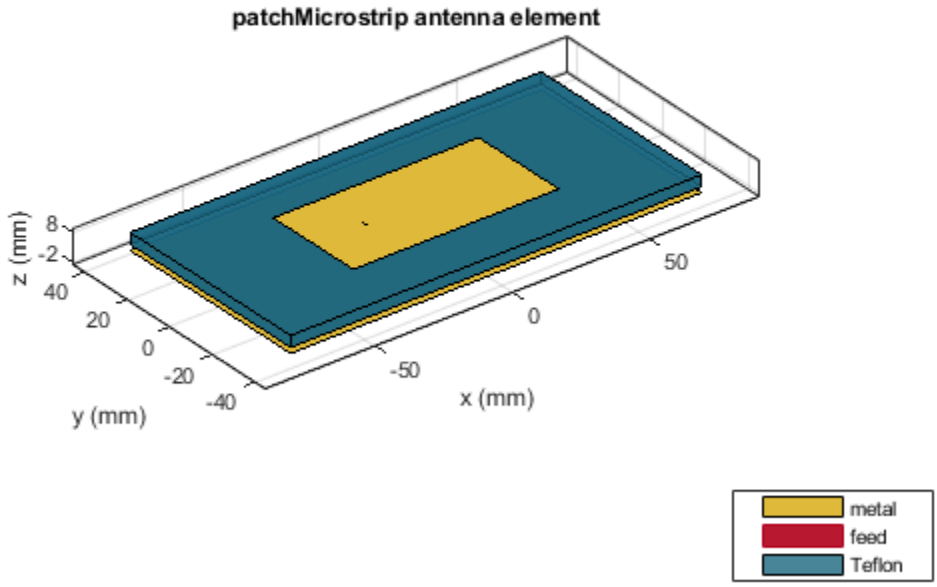
	Name	Relative_Permittivity	Loss_Tangent	Frequency	Comments
1	Air	1	0	1.0000e+009	
2	FR4	4.8000	0.0260	100.0000e+0...	
3	Teflon	2.1000	2.0000e-04	100.0000e+0...	
4	Foam	1.0300	1.5000e-04	50.0000e+006	
5	Polystyrene	2.5500	1.0000e-04	100.0000e+0...	
6	Plexiglas	2.5900	0.0068	10.0000e+009	
7	Fused quartz	3.7800	1.0000e-04	10.0000e+009	
8	E glass	6.2200	0.0023	100.0000e+0...	
9	RO4725JXR	2.5500	0.0022	2.5000e+009	
10	RO4730JXR	3	0.0023	2.5000e+009	
11	TMM3				

Use Teflon as a dielectric substrate. There is an air gap between the patch groundplane and the dielectric.

```
sub = dielectric('Name',{'Air','Teflon'},'EpsilonR',[1 2.1],...
    'Thickness',[.002 .004],'LossTangent',[0 2e-04]);
```

Add the substrate to the patch antenna.

```
p.Substrate = sub;
figure
show(p)
```



Three Layer Dielectric Substrate between Patch and Ground Plane

Create a microstrip patch antenna.

```
p = patchMicrostrip;
```

For dielectric properties, use the Dielectric Catalog.

```
openDielectricCatalog
```

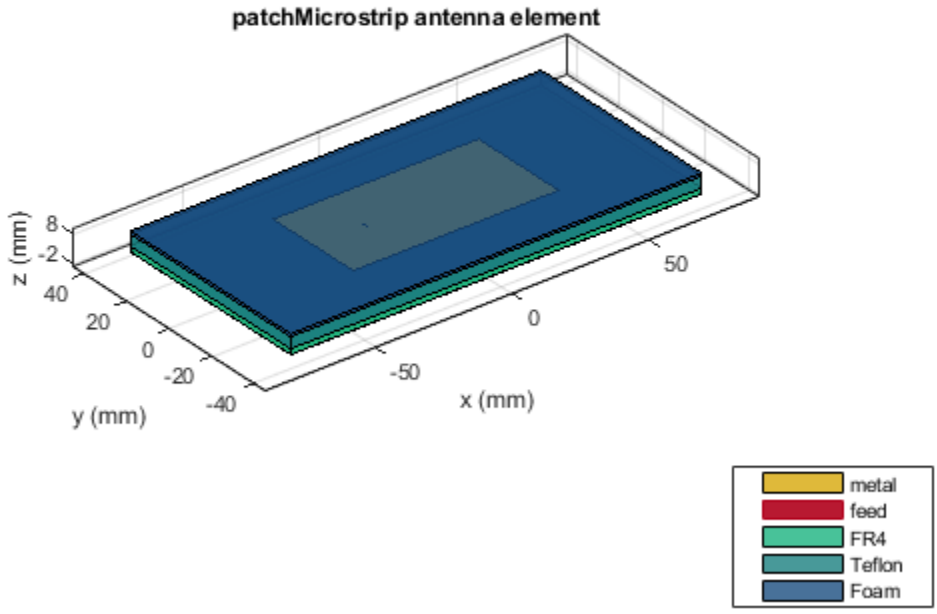
	Name	Relative_Permittivity	Loss_Tangent	Frequency	Comments
1	Air	1	0	1.0000e+009	
2	FR4	4.8000	0.0260	100.0000e+0...	
3	Teflon	2.1000	2.0000e-04	100.0000e+0...	
4	Foam	1.0300	1.5000e-04	50.0000e+006	
5	Polystyrene	2.5500	1.0000e-04	100.0000e+0...	
6	Plexiglas	2.5900	0.0068	10.0000e+009	
7	Fused quartz	3.7800	1.0000e-04	10.0000e+009	
8	E glass	6.2200	0.0023	100.0000e+0...	
9	RO4725JXR	2.5500	0.0022	2.5000e+009	
10	RO4730JXR	3	0.0023	2.5000e+009	
11	TMM3				

Use FR4, Teflon and Foam as the three layers of the substrate.

```
sub = dielectric('Name',{'FR4','Teflon','Foam'},'EpsilonR',...
    [4.80 2.10 1.03],'Thickness',[0.002 0.004 0.001],...
    'LossTangent',[0.0260 2e-04 1.5e-04]);
```

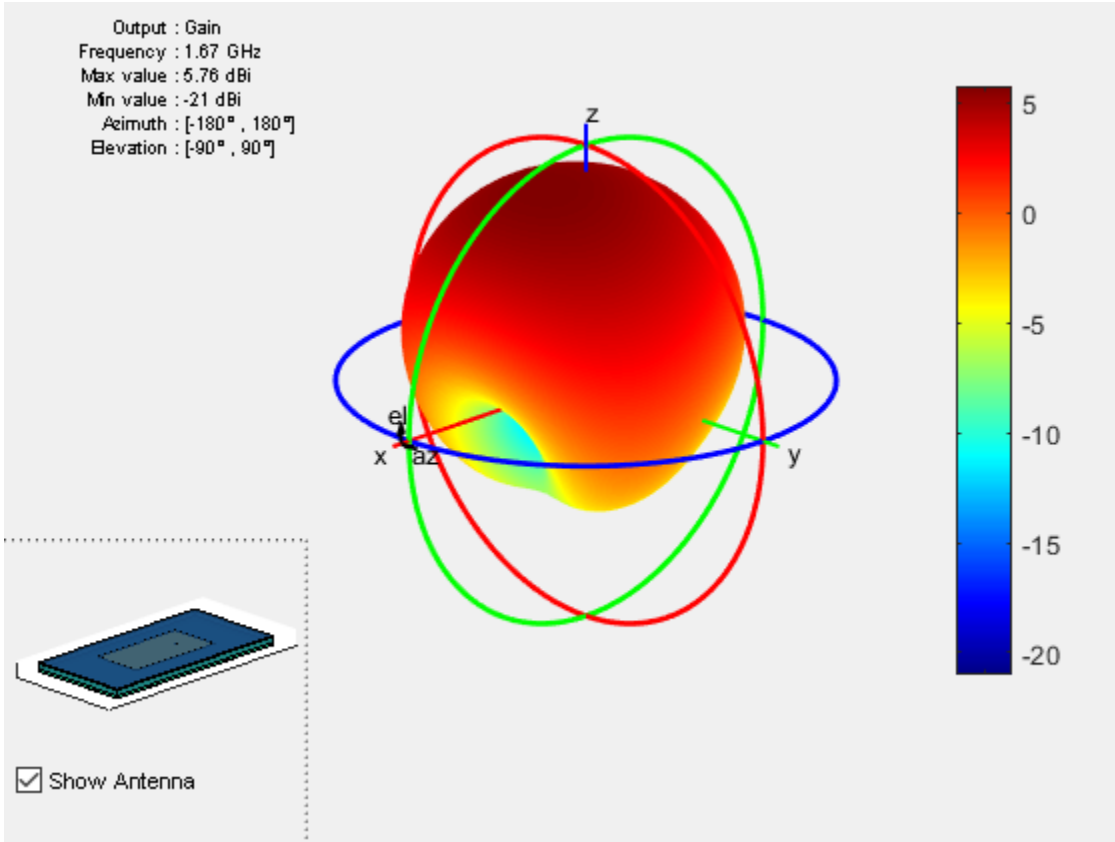
Add the three layer substrate to the patch antenna.

```
p.Substrate = sub;
figure
show(p)
```



Plot the radiation pattern of the antenna.

```
figure
pattern(p,1.67e9)
```



Infinite Reflector Backed Dielectric Substrate Antenna

Design a dipole antenna backed by a dielectric substrate and an infinite reflector.

Create a dipole antenna of length, 0.15 m, and width, 0.015 m.

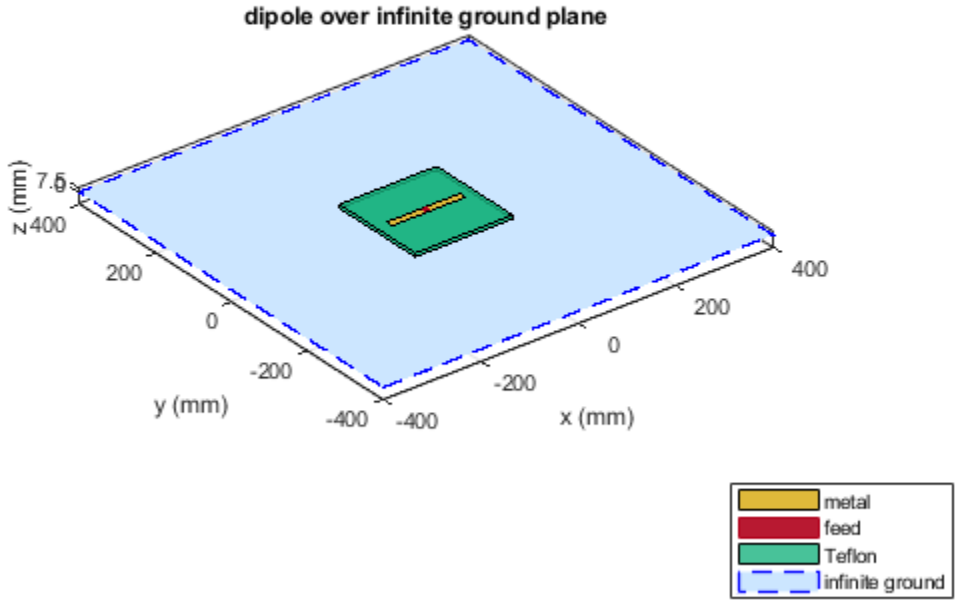
```
d = dipole('Length',0.15,'Width',0.015, 'Tilt',90,'TiltAxis',[0 1 0]);
```

Create a reflector using the dipole antenna as an exciter and the dielectric, `teflon` as the substrate.

```
t = dielectric('Teflon')
rf = reflector('Exciter',d,'Spacing',7.5e-3,'Substrate',t);


t =

  dielectric with properties:

          Name: 'Teflon'
       EpsilonR: 2.1000
    LossTangent: 2.0000e-04
      Thickness: 0.0060

For more materials see <a href="matlab:openDielectricCatalog">catalog</a>
```

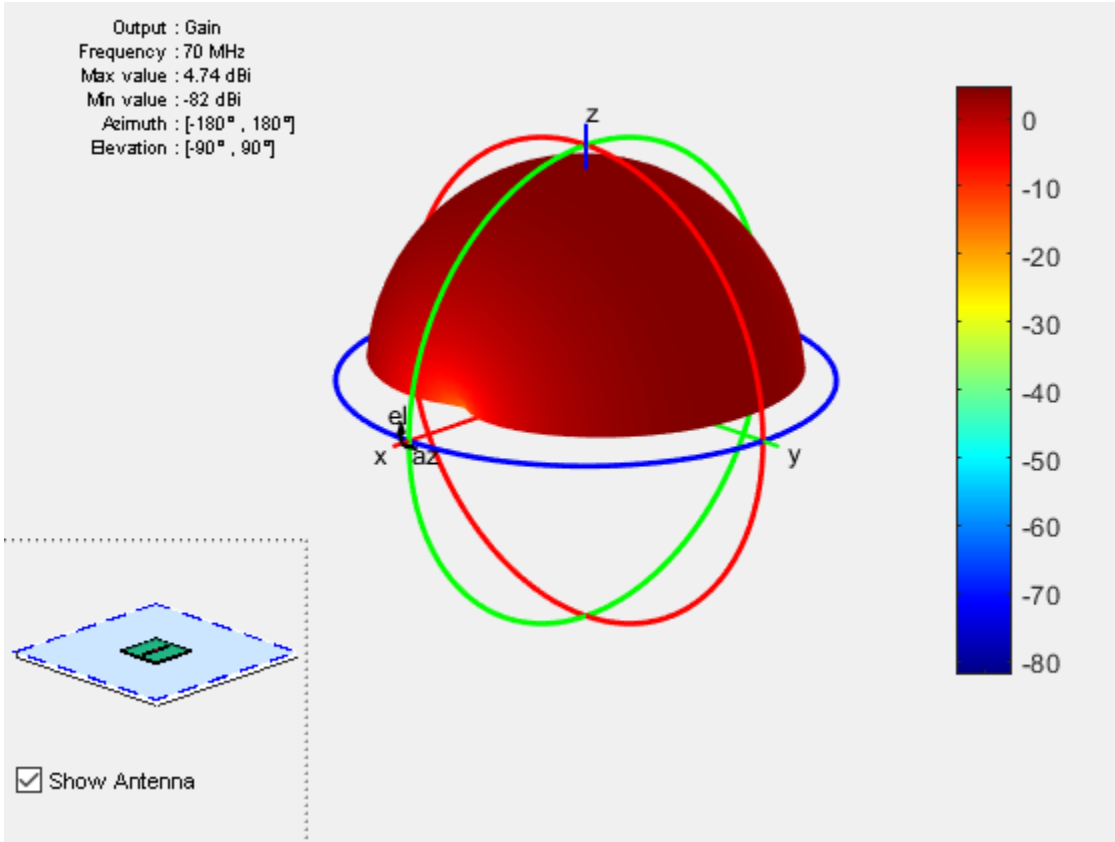
Set the groundplane length of the reflector to `inf`. View the structure.

```
rf.GroundPlaneLength = inf;
show(rf)
```



Calculate the radiation pattern of the antenna at 70 MHz.

```
pattern(rf,70e6)
```



Antenna On Dielectric Substrate - Compare Gain Values

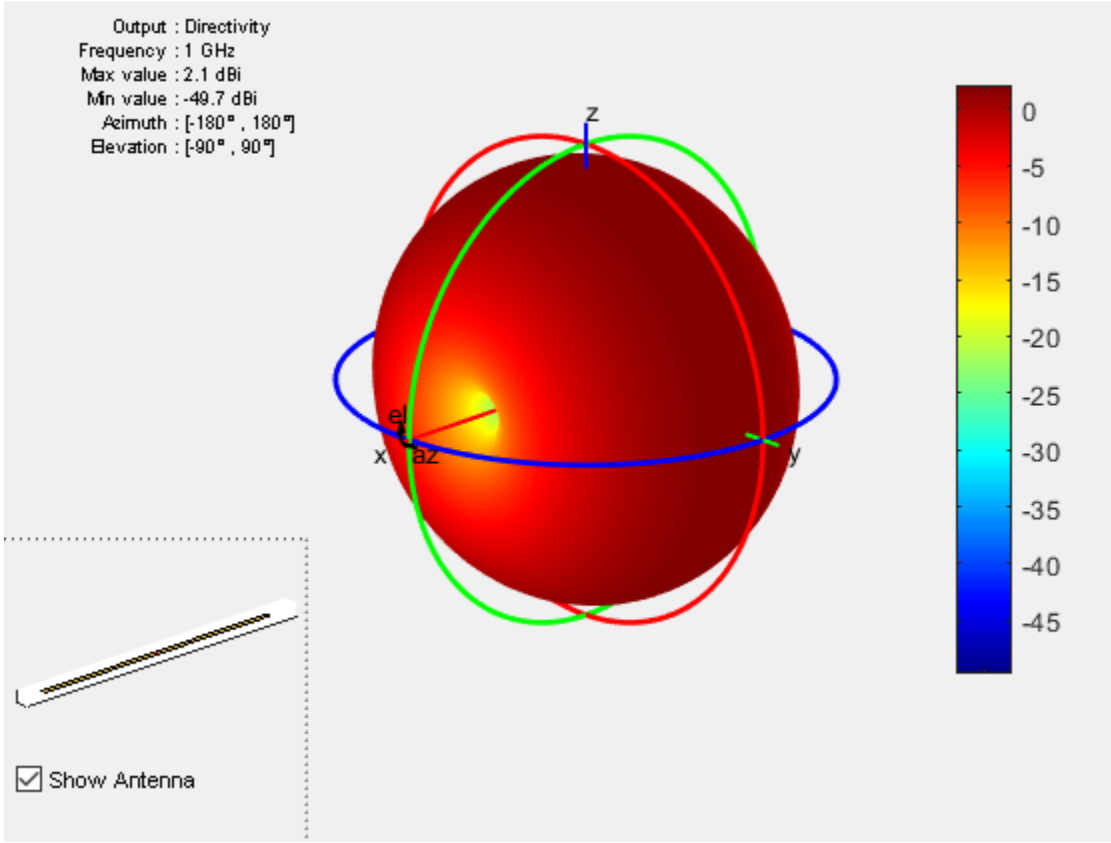
Compare the gain values of a dipole antenna in free space and dipole antenna on a substrate.

Design a dipole antenna at 1 GHz.

```
d = design(dipole,1e9);
l_by_w = d.Length/d.Width;
d.Tilt = 90;
d.TiltAxis = [0 1 0];
```

Plot the radiation pattern of the dipole in free space at 1GHz.

```
figure
pattern(d,1e9);
```



Use FR4 as the dielectric substrate.

```
t = dielectric('FR4')
eps_r = t.EpsilonR;
lambda_0 = physconst('lightspeed')/1e9;
lambda_d = lambda_0/sqrt(eps_r);
```

```
t =

  dielectric with properties:
```

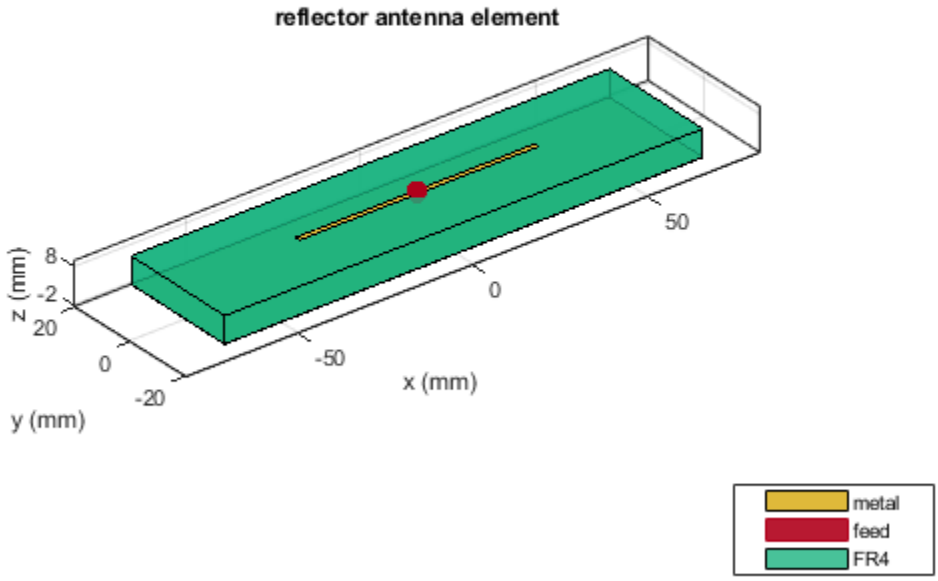
```
           Name: 'FR4'
       EpsilonR: 4.8000
    LossTangent: 0.0260
      Thickness: 0.0060

For more materials see <a href="matlab:openDielectricCatalog">catalog</a>
```

Adjust the length of the dipole based on the wavelength.

```
d.Length = lambda_d/2;
d.Width = d.Length/l_by_w;
```

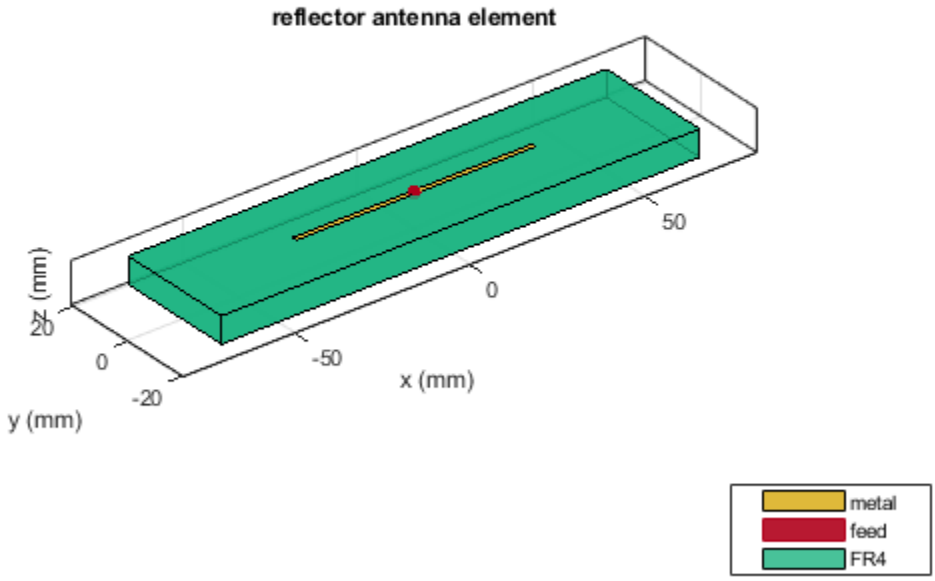
Design a reflector at 1 GHz with the dipole as the excitor and FR4 as the substrate.

```
rf = design(reflector,1e9);
rf = reflector('Exciter',d,'Spacing',7.5e-3,'Substrate',t);
rf.GroundPlaneLength = lambda_d;
rf.GroundPlaneWidth = lambda_d/4;
figure
show(rf)
```



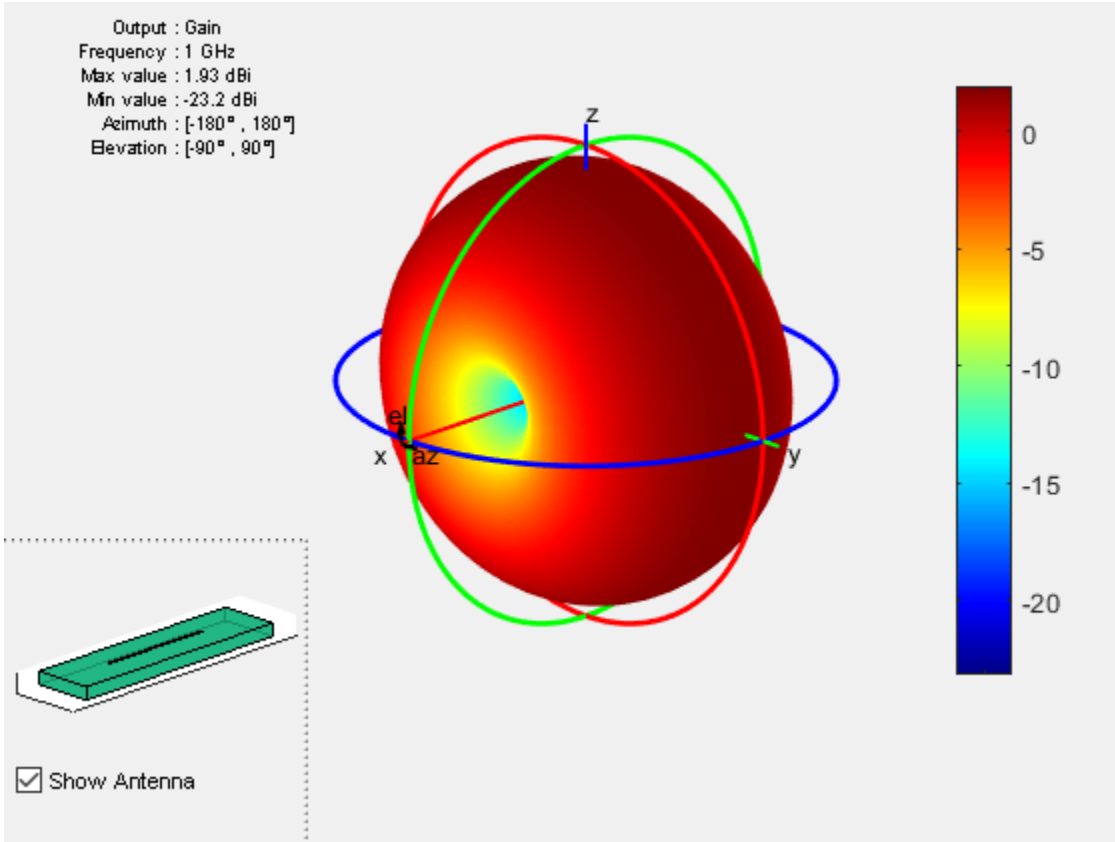
Remove the groundplane for plotting the gain of the dipole on the substrate.

```
rf.GroundPlaneLength = 0;
show(rf)
```



Plot the radiation pattern of the dipole on the substrate at 1 GHz.

```
figure
pattern(rf,1e9);
```



Compare the gain values.

- Gain of the dipole in free space = 2.11 dBi
- Gain of the dipole on substrate = 1.93 dBi

Input Arguments

`material` — Material from dielectric catalog

`'Air'` (default)

Material from the dielectric catalog, specified as one of the values from the `DielectricCatalog`.

Example: `'FR4'`

Data Types: `char`

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `'Name','Air'`

`Name` — Name of dielectric material

character vector

Name of the dielectric material you want to specify in the output, specified as the comma-separated pair consisting of `'Name'` and a character vector.

Example: `'Name','Taconic_TLC'`

Data Types: `char`

`EpsilonR` — Relative permittivity of dielectric material

`1` | vector

Relative permittivity of the dielectric material, specified as the comma-separated pair consisting of `'EpsilonR'` and vector.

Example: `'EpsilonR',4.8000`

Data Types: `double`

`LossTangent` — Loss in dielectric material

`0` (default) | vector

Loss in the dielectric material, specified as the comma-separated pair consisting of `'LossTangent'` and vector.

Example: `'LossTangent',0.0260`

Data Types: `double`

`Thickness` — Thickness of dielectric material

`0.0060` (default) | vector in meters

Thickness of the dielectric material along default z-axis, specified as the comma-separated pair consisting of `'Thickness'` and vector in meters. This property applies only when you call the function with no output arguments.

Example: `'Thickness', 0.05`

Data Types: `double`

Output Arguments

d — Dielectric material
object handle

Dielectric material, returned as an object handle. You can use the dielectric material object handle to add dielectric material to an antenna.

See Also

`DielectricCatalog`

Topics

“Antenna Toolbox Limitations”

Introduced in R2016a

DielectricCatalog

Catalog of dielectric materials

Syntax

```
dc = DielectricCatalog
```

Description

`dc = DielectricCatalog` creates an object handle for the dielectric catalog.

- To open the dielectric catalog, use `open(dc)`
- To know the properties of a dielectric material from the dielectric catalog, use `s = find(dc,name)`.

Examples

Use Dielectric Catalog Element in Cavity

Open the dielectric catalog.

```
dc = DielectricCatalog;
open(dc)
```

	Name	Relative_Permittivity	Loss_Tangent	Frequency	Comments
1	Air	1	0	1.0000e+009	
2	FR4	4.8000	0.0260	100.0000e+0...	
3	Teflon	2.1000	2.0000e-04	100.0000e+0...	
4	Foam	1.0300	1.5000e-04	50.0000e+006	
5	Polystyrene	2.5500	1.0000e-04	100.0000e+0...	
6	Plexiglas	2.5900	0.0068	10.0000e+009	
7	Fused quartz	3.7800	1.0000e-04	10.0000e+009	
8	E glass	6.2200	0.0023	100.0000e+0...	
9	RO4725JXR	2.5500	0.0022	2.5000e+009	
10	RO4730JXR	3	0.0023	2.5000e+009	
11	TMM3	3.4500	0.0020	10.0000e+009	

List the properties of the dielectric material `Foam`.

```
s = find(dc,'Foam')
```

```
s = struct with fields:
                      Name: 'Foam'
    Relative_Permittivity: 1.0300
             Loss_Tangent: 1.5000e-04
                Frequency: 50000000
                 Comments: ''
```

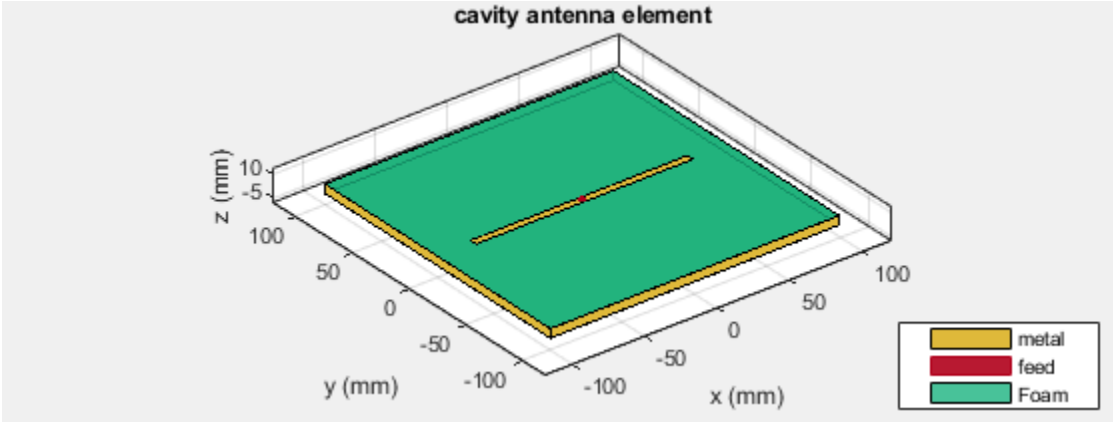
Use the material Foam as a dielectric in a cavity antenna of height and spacing, 0.0060 m.

```
d = dielectric('Foam');
c = cavity('Height',0.0060,'Spacing',0.0060,'Substrate',d)
```

```
c =
  cavity with properties:

            Exciter: [1x1 dipole]
          Substrate: [1x1 dielectric]
             Length: 0.2000
              Width: 0.2000
             Height: 0.0060
            Spacing: 0.0060
    EnableProbeFeed: 0
               Tilt: 0
           TiltAxis: [1 0 0]
```

```
                Load: [1x1 lumpedElement]
```

```
show (c)
```



Input Arguments

name — Name of dielectric material

`'Air'` (default) | character vector

Name of a dielectric material from the dielectric catalog, specified as a character vector.

Example: `'FR4'`

Data Types: `char`

dc — Dielectric catalog

object handle

Dielectric catalog, specified as an object handle.

Data Types: `char`

Output Arguments

dc — Dielectric catalog
object handle

Dielectric catalog, returned as an object handle.

s — Parameters of dielectric material
structure

Parameters of a dielectric material from the dielectric catalog, returned as a structure.

See Also

`dielectric`

Introduced in R2016a

hornangle2size

Equivalent flare width and flare height from flare angles

Syntax

```
[flarewidth,flareheight]= hornangle2size(width,height,flarelength,
angleE,angleH)
```

Description

`[flarewidth,flareheight]= hornangle2size(width,height,flarelength, angleE,angleH)` calculates the equivalent `flarewidth` and `flareheight` for a rectangular horn antenna from its flare angles, `angleE`, and `angleH`.

Examples

Calculate Flare Width and Flare Height of Horn Antenna

Calculate the flare width and the flare height of a horn antenna with

- Width of the waveguide = 0.0229 m
- Height of the waveguide = 0.0102 m
- Flare length of the horn = 0.2729 m
- Flare angle in the E-plane = 12.2442 degrees
- Flare angle in the H-plane = 14.4712 degrees

```
width = 0.0229;
height = 0.0102;
flarelength = 0.2729;
angleE = 12.2442;
angleH = 14.4712;
[flarewidth,flareheight] = hornangle2size(width,height,flarelength,...
                           angleE,angleH)
```

```
flarewidth = 0.1638

flareheight = 0.1286
```

Input Arguments

`width` — Rectangular waveguide width

scalar in meters

Rectangular waveguide width, specified as the comma-separated pair consisting of `'Width'` and a scalar in meters.

Data Types: `double`

`height` — Rectangular waveguide height

scalar in meters

Rectangular waveguide height, specified as the comma-separated pair consisting of `'Height'` and a scalar in meters.

Data Types: `double`

`flarelength` — Flare length of horn

scalar in meters

Flare length of horn, specified as the comma-separated pair consisting of `'FlareLength'` and a scalar in meters.

Data Types: `double`

`angleE` — Flare angle in E-plane

scalar in degrees

Flare angle in E-plane of the horn, specified as a scalar in degrees.

Data Types: `double`

`angleH` — Flare angle in H-plane

scalar in meters

Flare angle in H-plane of the horn, specified as a scalar in degrees.

Data Types: `double`

Output Arguments

`flarewidth` — Flare width of horn
scalar in meters

Flare width of horn, returned as a scalar in meters.

Data Types: `double`

`flareheight` — Flare height of horn
scalar in meters

Flare height of horn, returned as a scalar in meters.

Data Types: `double`

See Also

`horn`

Introduced in R2016a

add

Class: polarpattern

Add data to polar plot

Syntax

```
add(p,d)
add(p,angle,magnitude)
```

Description

`add(p,d)` adds new antenna data to the polar plot, `p` based on the real amplitude values, `data`.

`add(p,angle,magnitude)` adds data sets of `angle` vectors and corresponding `magnitude` matrices to polar plot `p`.

Input Arguments

p — Polar plot

scalar handle

Polar plot, specified as a scalar handle.

data — Antenna or array data

real length-*M* vector | real *M*-by-*N* matrix | real *N*-*D* array | complex vector or matrix

Antenna or array data, specified as one of the following:

- A real length-*M* vector, where *M* contains the magnitude values with angles assumed to be $\frac{(0:M-1)}{M}\times 360^{\circ}$ degrees.

- A real *M*-by-*N* matrix, where *M* contains the magnitude values and *N* contains the independent data sets. Each column in the matrix has angles taken from the vector $\frac{(0:M-1)}{M}\times 360^\circ$ degrees. The set of each angle can vary for each column.
- A real *N*-*D* array, where *N* is the number of dimensions. Arrays with dimensions 2 and greater are independent data sets.
- A complex vector or matrix, where `data` contains Cartesian coordinates (*(x,y)* of each point. *x* contains the real part of `data` and *y* contains the imaginary part of `data`.

When data is in a logarithmic form such as dB, magnitude values can be negative. In this case,`polarpattern` plots the lowest magnitude values at the origin of the polar plot and highest magnitude values at the maximum radius.

`angle` — Set of angles
vector in degrees

Set of angles, specified as a vector in degrees.

`magnitude` — Set of magnitude values
vector | matrix

Set of magnitude values, specified as a vector or a matrix. For a matrix of magnitude values, each column is an independent set of magnitude values and corresponds to the same set of angles.

Examples

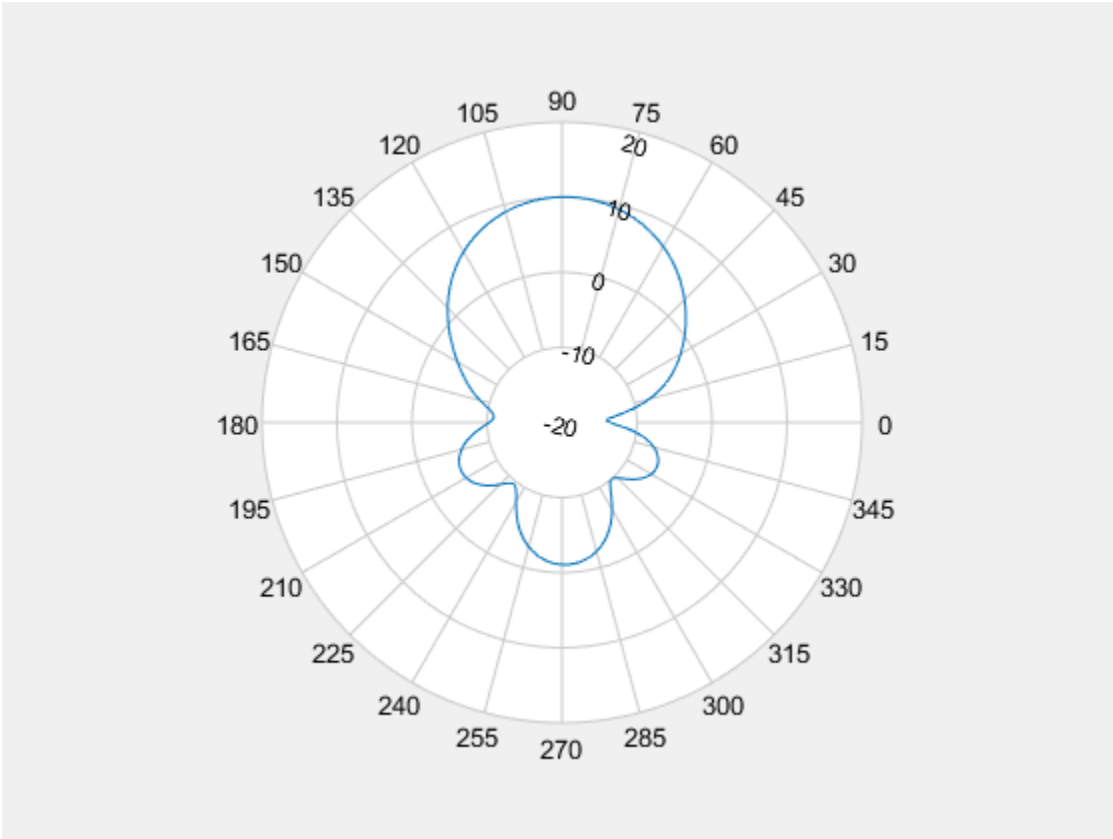
Add Data To Polar Plot

Create a helix antenna that has 28 mm radius, a 1.2 mm width, and 4 turns. Calculate the directivity of the antenna at 1.8 GHz.

```
hx = helix('Radius',28e-3,'Width',1.2e-3,'Turns',4);
H = pattern(hx, 1.8e9,0,0:1:360);
```

Plot the polar pattern.

```
P = polarpattern(H);
```

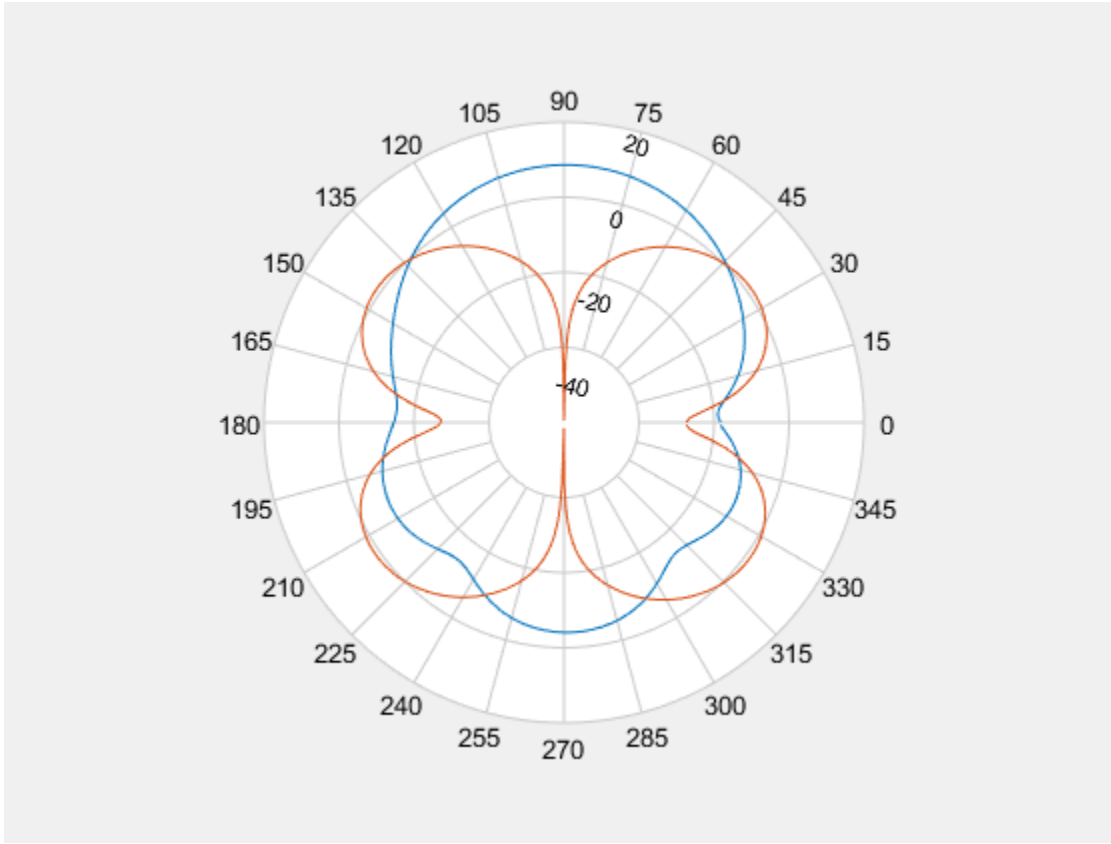


Create a dipole antenna and calculate the directivity at 270 MHz.

```
d = dipole;
D = pattern(d,270e6,0,0:1:360);
```

Add the directivity of the dipole to the existing polar plot of helix antenna.

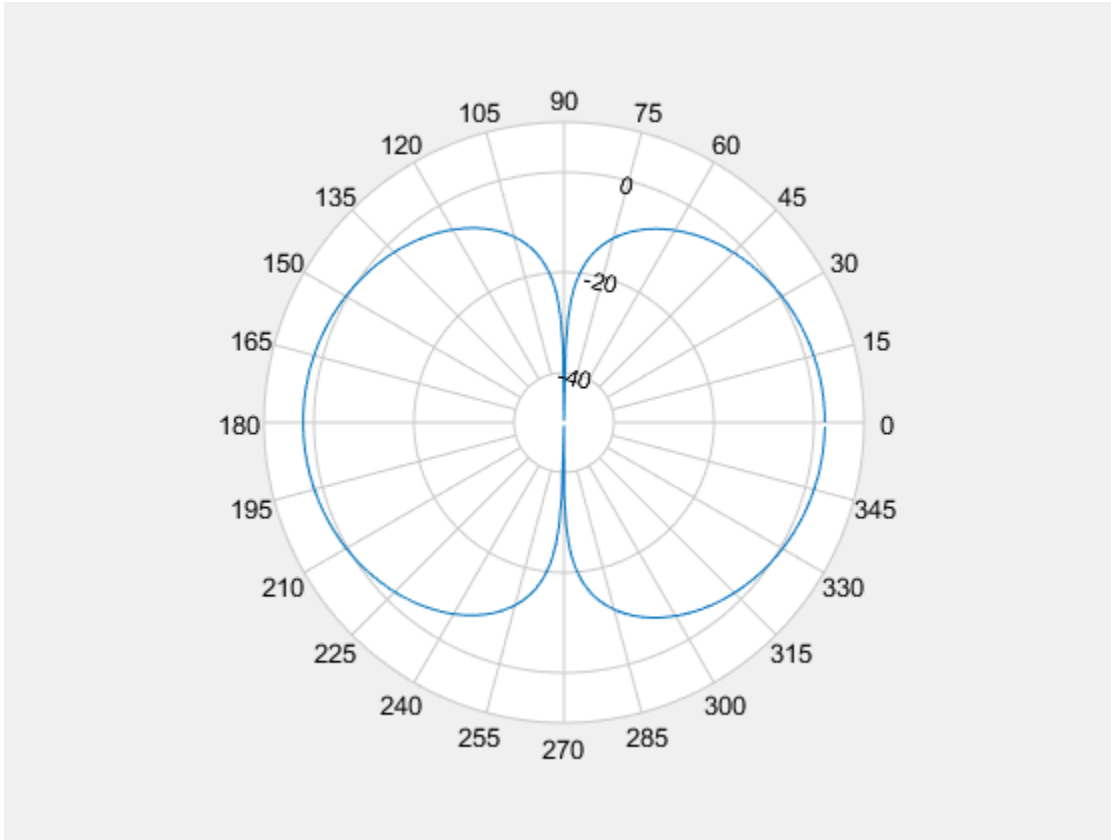
```
add(P,D);
```



Add Angle and Magnitude Data to Polar Pattern

Create a dipole and plot the polar pattern of its directivity at 75 MHz.

```
d = dipole;
D = pattern(d,75e6,0,0:1:360);
P = polarpattern(D);
```



Create a cavity antenna. Calculate the directivity of the antenna at 1 GHz. Write the directivity of the antenna to `cavity.pln` using the `msiwrite` function.

```
c = cavity;
msiwrite(c,1e9,'cavity','Name','Cavity Antenna Specifications');
```

Read the data from `cavity.pln` to `Horizontal`, `Vertical` and `Optional` structures using the `msiread` function.

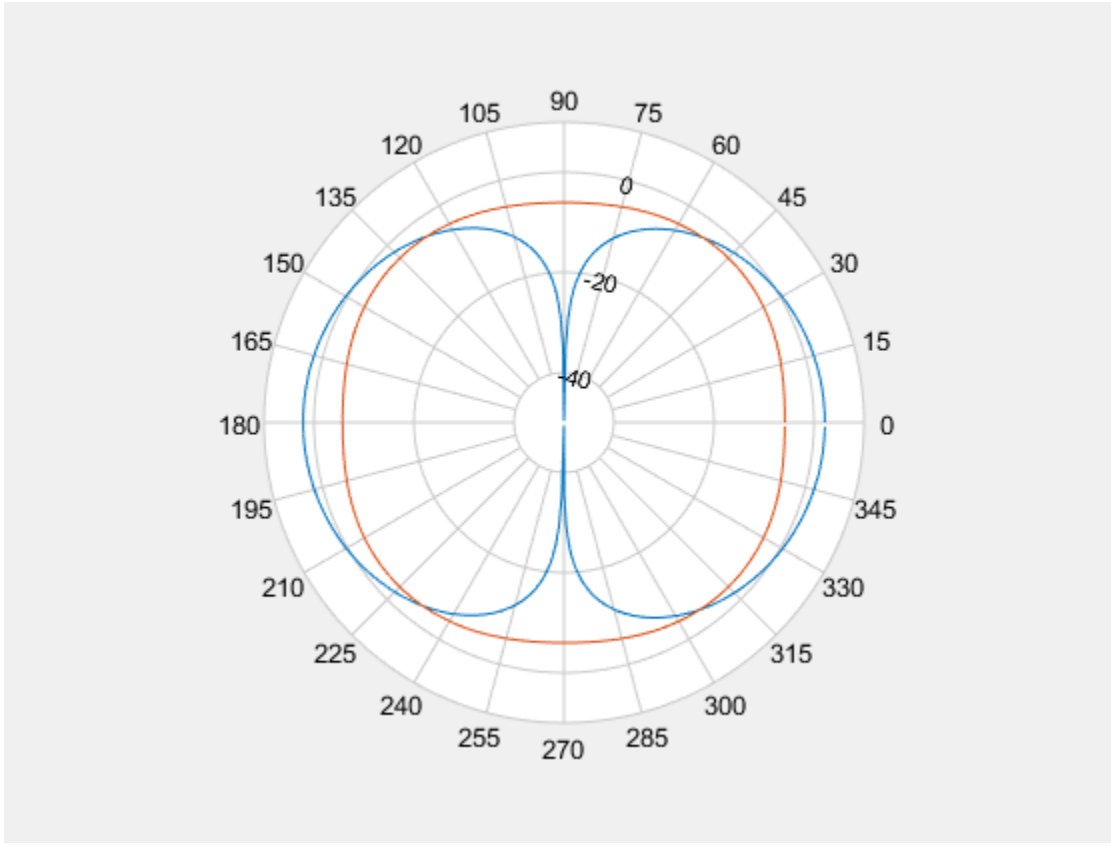
```
[Horizontal,Vertical,Optional] = msiread('cavity.pln')
```

```
Horizontal = struct with fields:
    PhysicalQuantity: 'Gain'
           Magnitude: [360x1 double]
```

```
               Units: 'dBi'
             Azimuth: [360x1 double]
           Elevation: 0
           Frequency: 1.0000e+09
               Slice: 'Elevation'


Vertical = struct with fields:
    PhysicalQuantity: 'Gain'
           Magnitude: [360x1 double]
               Units: 'dBi'
             Azimuth: 0
           Elevation: [360x1 double]
           Frequency: 1.0000e+09
               Slice: 'Azimuth'


Optional = struct with fields:
         name: 'Cavity Antenna Specifications'
    frequency: 1.0000e+09
         gain: [1x1 struct]
```

Add horizontal directivity data of the cavity antenna to the existing polar pattern of the dipole

```
add(P,Horizontal.Azimuth,Horizontal.Magnitude);
```



See Also

addCursor | animate | createLabels | findLobes | replace | showPeaksTable | showSpan

Introduced in R2016a

addCursor

Class: polarpattern

Add cursor to polar plot angle

Syntax

```
addCursor(p,angle)
addCursor(p,angle,index)
id = addCursor( ___ )
```

Description

`addCursor(p,angle)` adds a cursor to the active polar plot, `p`, at the data point closest to the specified `angle`. Angle units are in degrees.

The first cursor added is called `'C1'`, the second `'C2'`, and so on.

`addCursor(p,angle,index)` adds a cursor at a specified data set `index`. `index` can be a vector of indices.

`id = addCursor( ___ )` returns a cell array with one ID for each cursor created. You can specify any of the arguments from the previous syntaxes.

Input Arguments

`p` — Polar plot

scalar handle

Polar plot, specified as a scalar handle.

`angle` — Angle values

scalar in degrees | vector in degrees

Angle values at which the cursor is added, specified as a scalar or a vector in degrees.

`index` — Data set index
scalar | vector

Data set index, specified as a scalar or a vector.

Examples

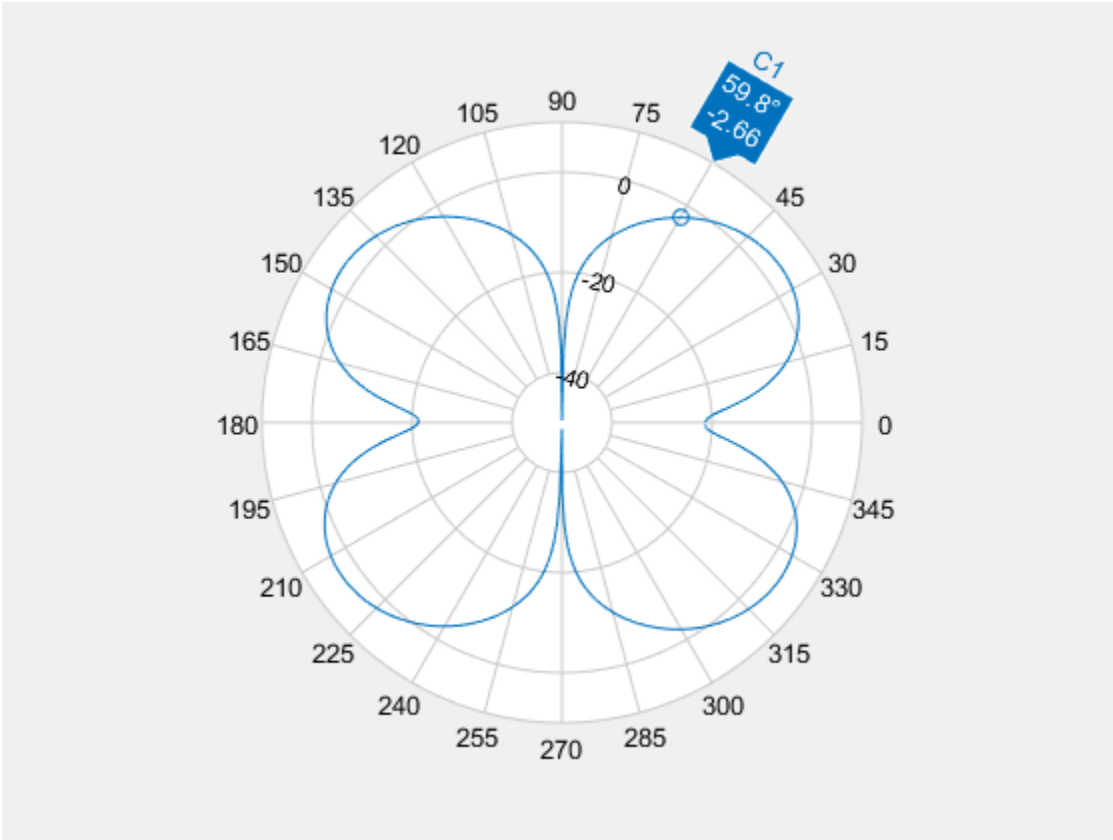
Add Cursor to Plot

Create a dipole antenna and calculate its directivity at 270 MHz.

```
d = dipole;
D = pattern(d,270e6,0,0:1:360);
```

Add a cursor to the polar plot at approximately 60 degrees. To place the cursor at 60 degrees, move it there by placing the pointer on the cursor and dragging.

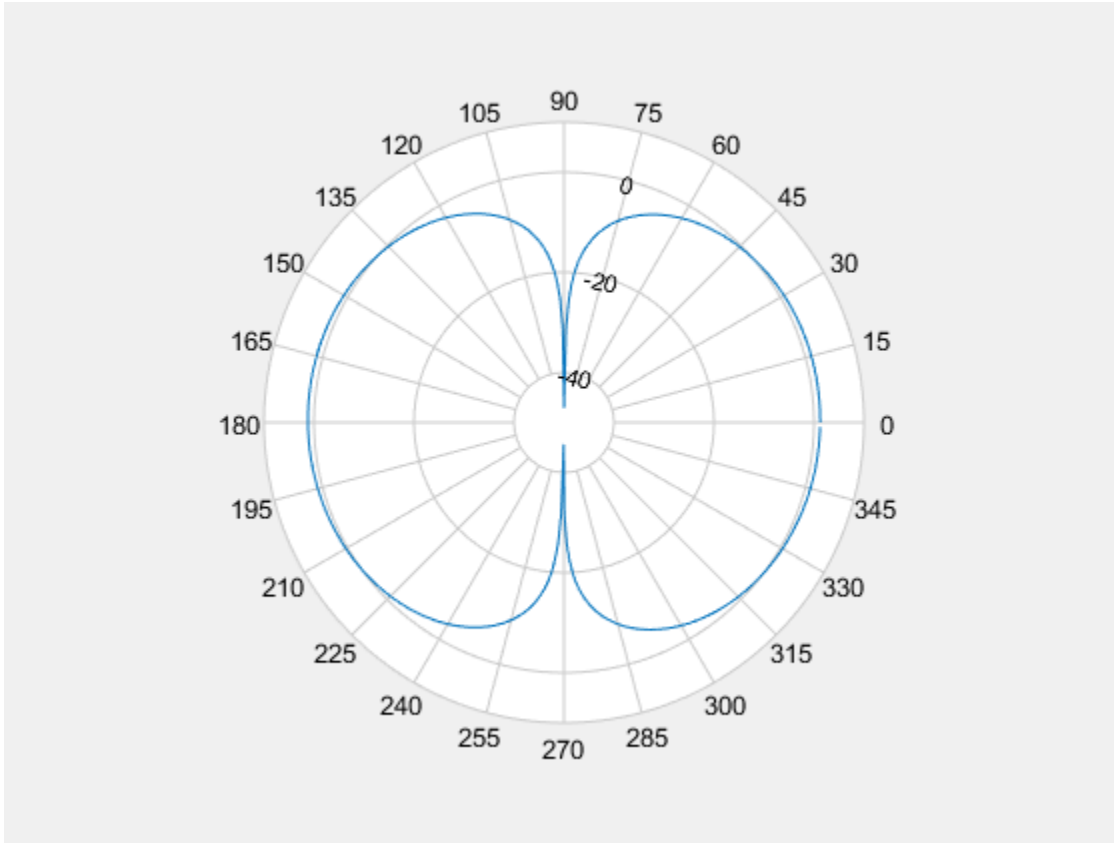
```
p = polarpattern(D);
addCursor(p,60);
```



Add Cursors to Two Data Sets

Create a top-hat monopole and plot its directivity at 75 MHz.

```
m = monopoleTopHat;
M = pattern(m,75e6,0,0:1:360);
P = polarpattern(M);
```

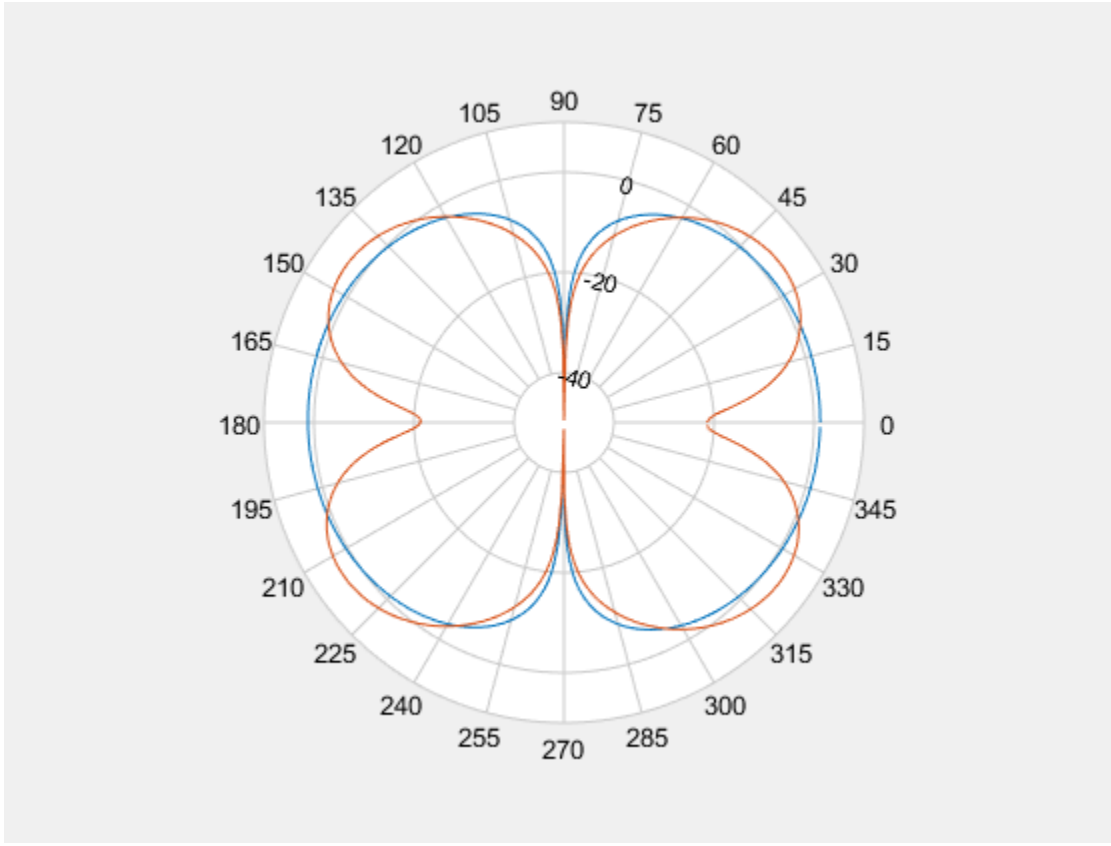


Create a dipole antenna and calculate its directivity at 270 MHz.

```
d = dipole;
D = pattern(d,270e6,0,0:1:360);
```

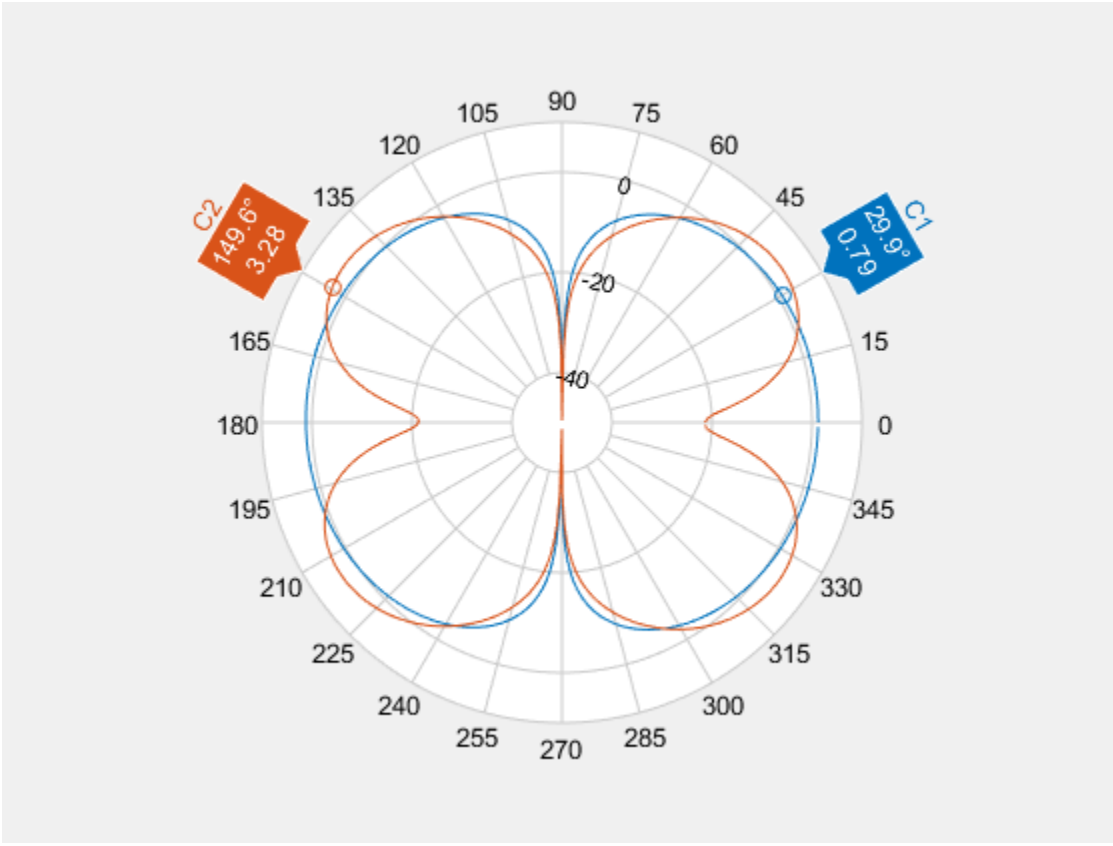
Add the directivity pattern of the dipole to the polar plot of the top-hat monopole.

```
add(P,D);
```



Add a cursor at approximately 30 degrees to the top-hat monopole polar pattern (data set 1) and at approximately 150 degrees to the dipole polar pattern (data set 2).

```
addCursor(P,[30 150],[1 2]);
```



See Also

add | animate | createLabels | findLobes | replace | showPeaksTable | showSpan

Introduced in R2016a

animate

Class: polarpattern

Replace existing data with new data for animation

Syntax

```
animate(p,data)
animate(p,angle,magnitude)
```

Description

`animate(p,data)` removes all the current data from polar plot, `p` and adds new data, based on real amplitude values, `data`.

`animate(p,angle,magnitude)` removes all the current data polar plot, `p` and adds new data sets of angle vectors and corresponding magnitude matrices.

Input Arguments

p — Polar plot

scalar handle

Polar plot, specified as a scalar handle.

data — Antenna or array data

real length-*M* vector | real *M*-by-*N* matrix | real *N*-*D* array | complex vector or matrix

Antenna or array data, specified as one of the following:

- A real length-*M* vector, where *M* contains the magnitude values with angles assumed to be $\frac{(0:M-1)}{M}\times 360^{\circ}$ degrees.

- A real *M*-by-*N* matrix, where *M* contains the magnitude values and *N* contains the independent data sets. Each column in the matrix has angles taken from the vector $\frac{(0:M-1)}{M}\times 360^\circ$ degrees. The set of each angle can vary for each column.
- A real *N*-*D* array, where *N* is the number of dimensions. Arrays with dimensions 2 and greater are independent data sets.
- A complex vector or matrix, where `data` contains Cartesian coordinates (*(x,y)* of each point. *x* contains the real part of `data` and *y* contains the imaginary part of `data`.

When data is in a logarithmic form such as dB, magnitude values can be negative. In this case,`polarpattern` plots the lowest magnitude values at the origin of the polar plot and highest magnitude values at the maximum radius.

`angle` — Set of angles
vector in degrees

Set of angles, specified as a vector in degrees.

`magnitude` — Set of magnitude values
vector | matrix

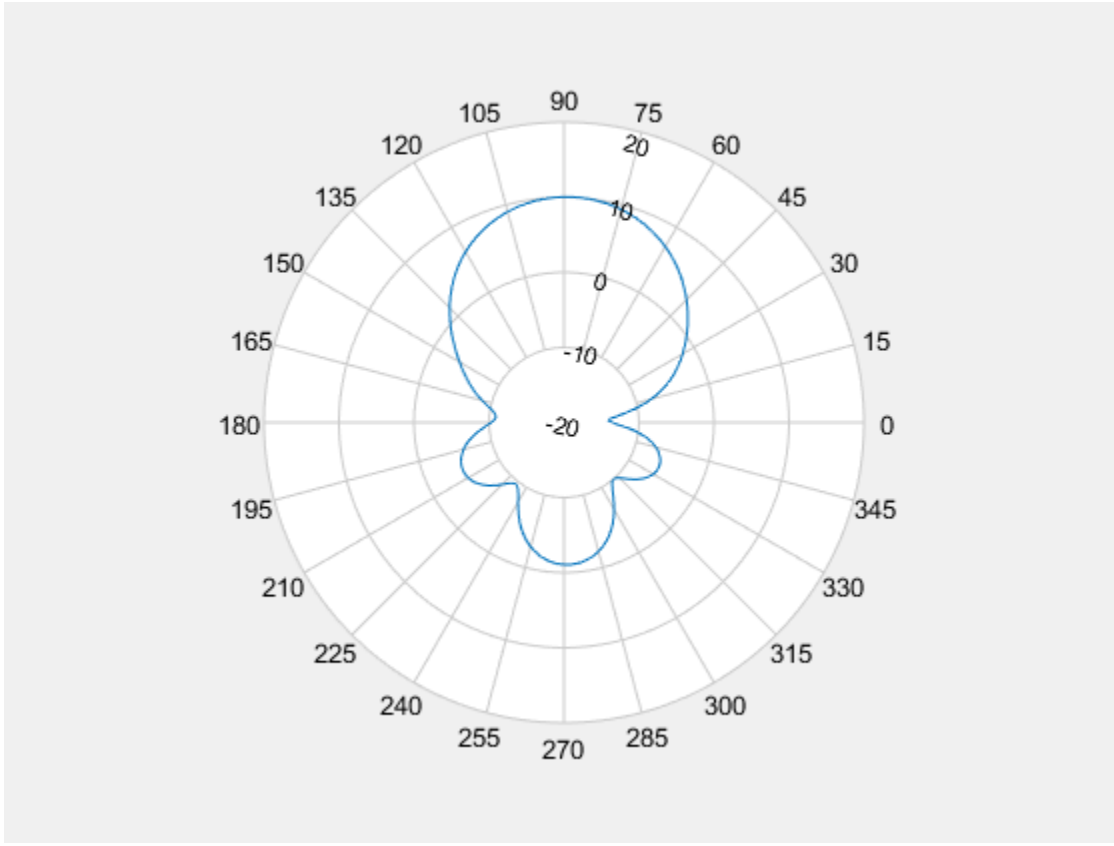
Set of magnitude values, specified as a vector or a matrix. For a matrix of magnitude values, each column is an independent set of magnitude values and corresponds to the same set of angles.

Examples

Replace Existing Polar Plot Data For Animation

Create a helix antenna that has a 28 mm radius, a 1.2 mm width, and 4 turns. Plot the directivity of the antenna at 1.8 GHz.

```
hx = helix('Radius',28e-3,'Width',1.2e-3,'Turns',4);
H = pattern(hx, 1.8e9,0,0:1:360);
P = polarpattern(H);
```

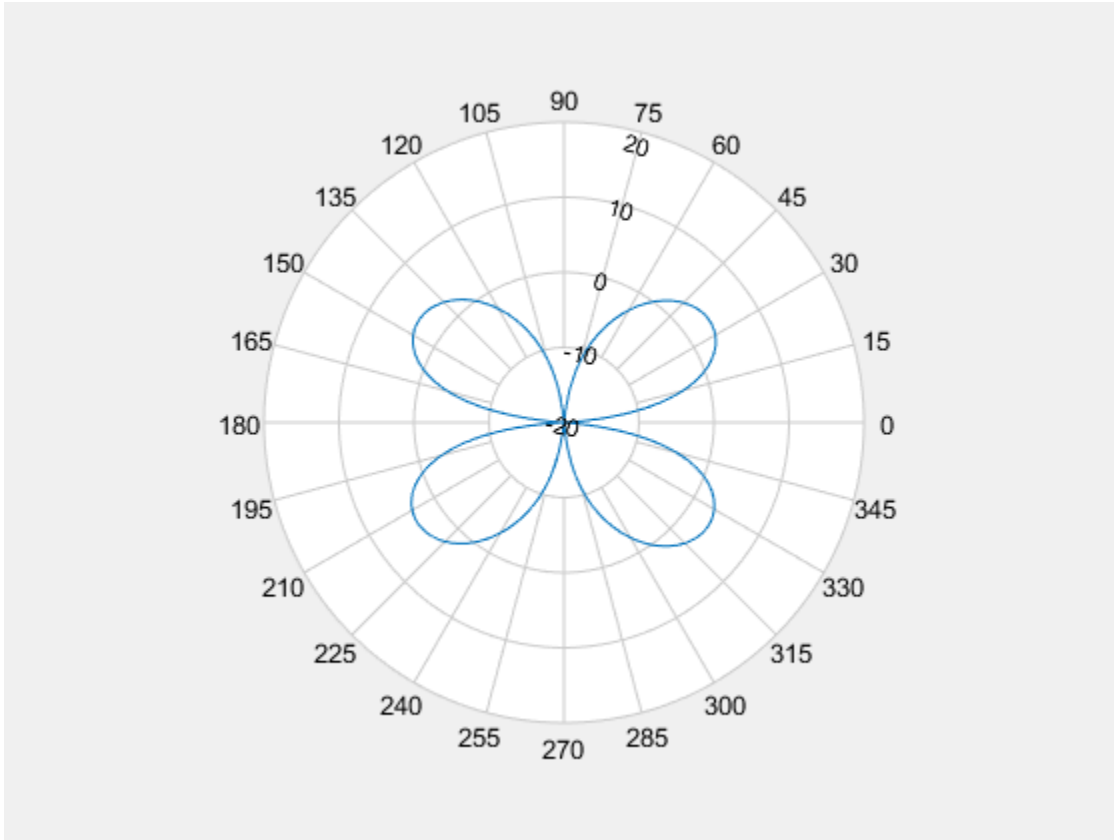


Create a dipole antenna and calculate its directivity at 270 MHz.

```
d = dipole;
D = pattern(d,270e6,0,0:1:360);
```

Replace the existing polar plot of the helix antenna with the directivity of the dipole using the `animate` method.

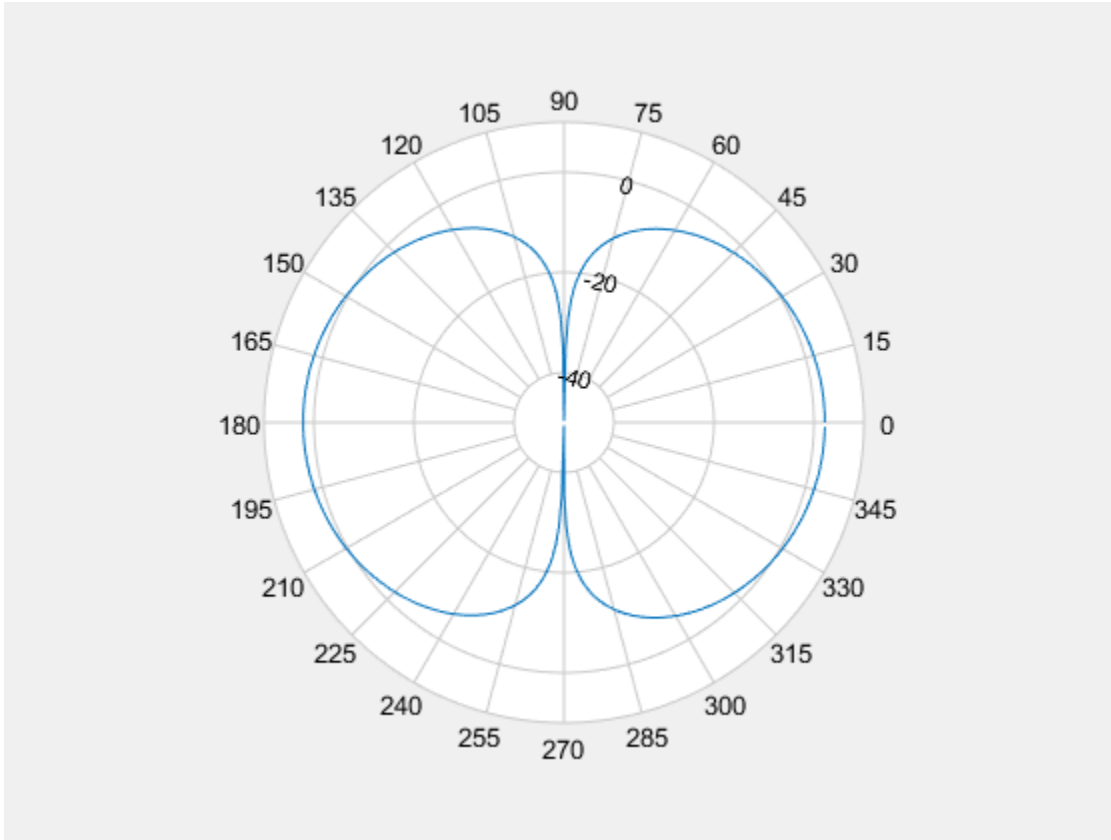
```
animate(P,D);
```



Animate Using Cavity Data

Create a default dipole antenna and plot the polar pattern of its directivity at 1 GHz.

```
d = dipole;
D = pattern(d,75e6,0,0:1:360);
P = polarpattern(D);
```



Create a default cavity antenna. Calculate the directivity of the antenna and write the data to `cavity.pln` using the `msiwrite` function.

```
c = cavity;
msiwrite(c,2.8e9,'cavity','Name','Cavity Antenna Specifications');
```

Read the cavity specifications file into `Horizontal`, `Vertical` and `Optional` structures using the `msiread` function.

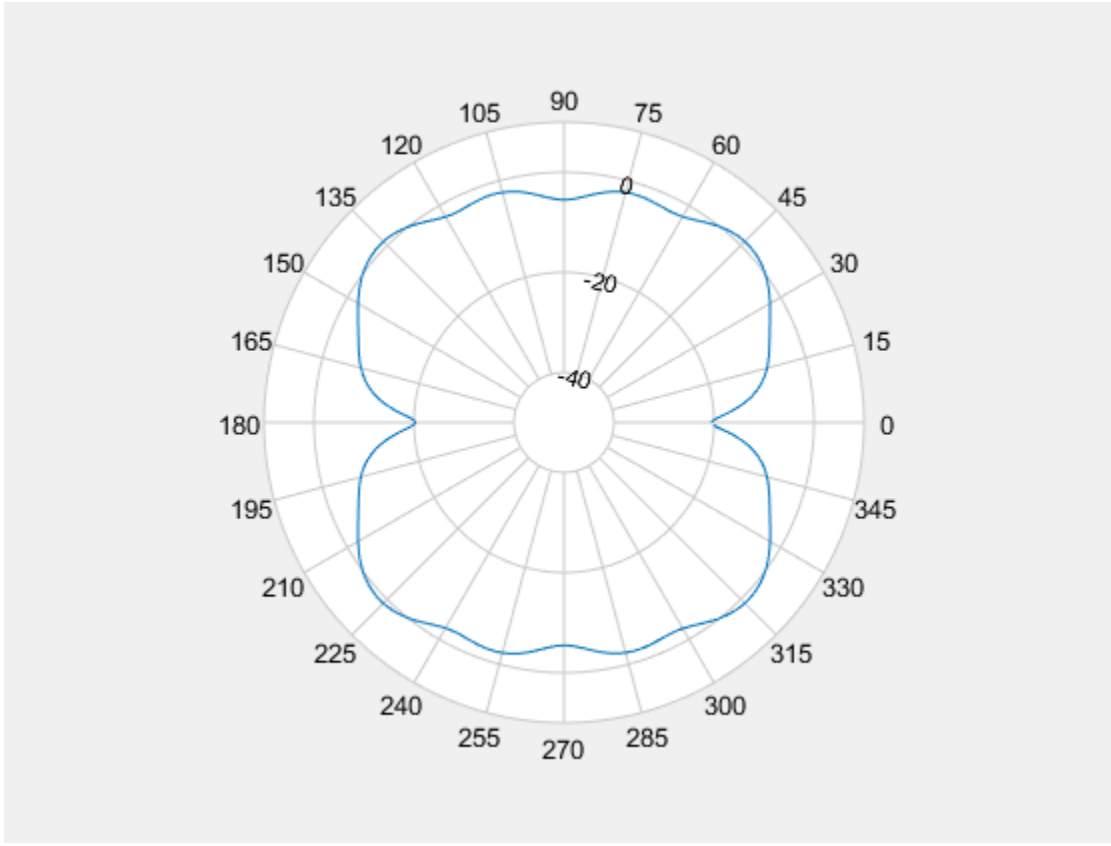
```
[Horizontal,Vertical,optional]= msiread('cavity.pln')
```

```
Horizontal = struct with fields:
    PhysicalQuantity: 'Gain'
           Magnitude: [360x1 double]
```

```
               Units: 'dBi'
             Azimuth: [360x1 double]
           Elevation: 0
           Frequency: 2.8000e+09
               Slice: 'Elevation'


Vertical = struct with fields:
    PhysicalQuantity: 'Gain'
           Magnitude: [360x1 double]
               Units: 'dBi'
             Azimuth: 0
           Elevation: [360x1 double]
           Frequency: 2.8000e+09
               Slice: 'Azimuth'


optional = struct with fields:
         name: 'Cavity Antenna Specifications'
    frequency: 2.8000e+09
         gain: [1x1 struct]
```

Replace data from the dipole antenna with data from cavity antenna.

```
animate(P,Horizontal.Azimuth,Horizontal.Magnitude);
```



See Also

add | addCursor | createLabels | findLobes | replace | showPeaksTable | showSpan

Introduced in R2016a

createLabels

Class: polarpattern

Create legend labels for polar plot

Syntax

```
createLabels(p,format,array)
```

Description

`createLabels(p,format,array)` adds the specified `format` label to each `array` of the polar plot `p`. The labels are stored as a cell array in the `LegendLabels` property of `p`.

Input Arguments

`p` — Polar plot

scalar handle

Polar plot, specified as a scalar handle.

`format` — Format for legend label

cell array

Format for legend label added to the polar plot, specified as a cell array. For more information on legend label format see, `legend`.

Data Types: `char`

`array` — Values to apply to `format`

array

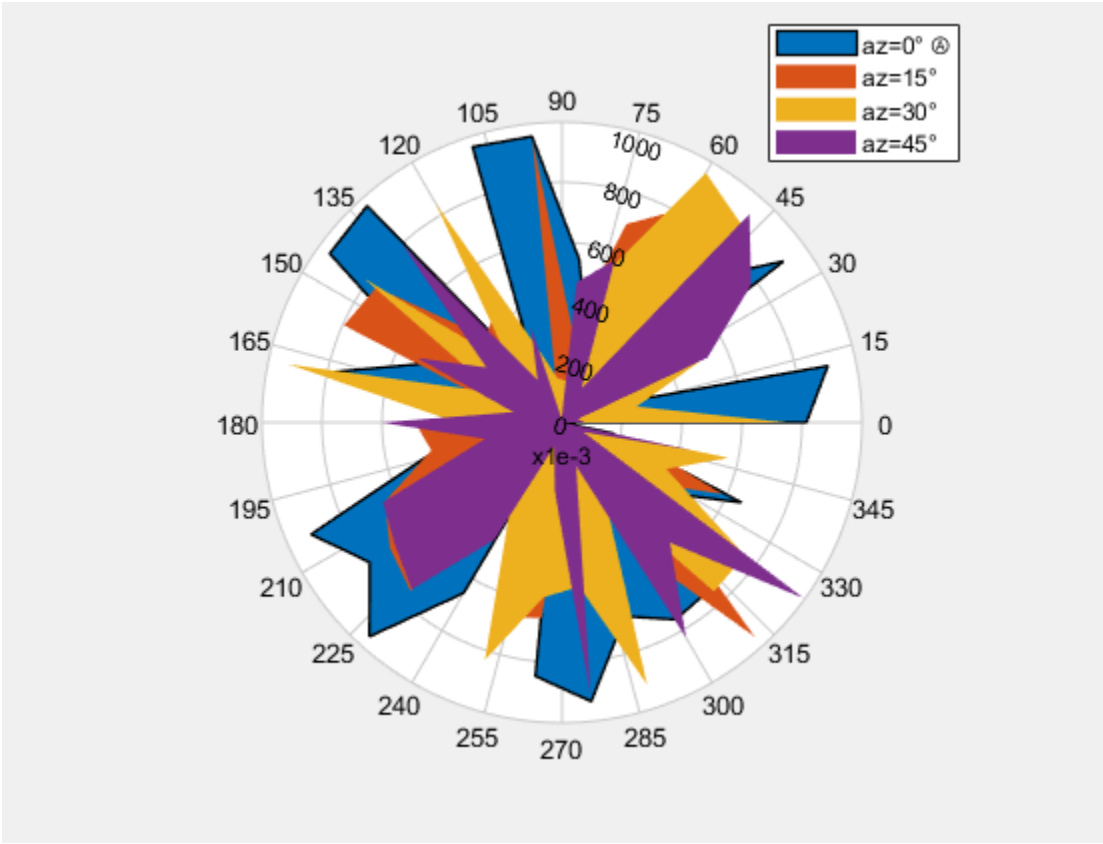
Values to apply to `format` , specified as an array. The values can be an array of angles or array of magnitude.

Examples

Add Legend Label to Polar Plot

Create a polar plot of unique values. Generate a legend label for this plot.

```
p = polarpattern(rand(30,4),'Style','filled');
createLabels(p,'az=%d#deg',0:15:45)
```



See Also

add | addCursor | animate | findLobes | replace | showPeaksTable | showSpan

Introduced in R2016a

findLobes

Class: polarpattern

Main, back, and side lobe data

Syntax

```
L = findLobes(p)
L = findLobes(p,index)
```

Description

`L = findLobes(p)` returns a structure, `L`, defining the main, back, and side lobes of the antenna or array radiation pattern in the specified polar plot, `p`.

`L = findLobes(p,index)` returns the radiation pattern lobes from the data set specified in `index`.

Input Arguments

`p` — Polar plot
scalar handle

Polar plot, specified as a scalar handle.

`index` — Index of data set
scalar

Index of data set, specified as a scalar.

Examples

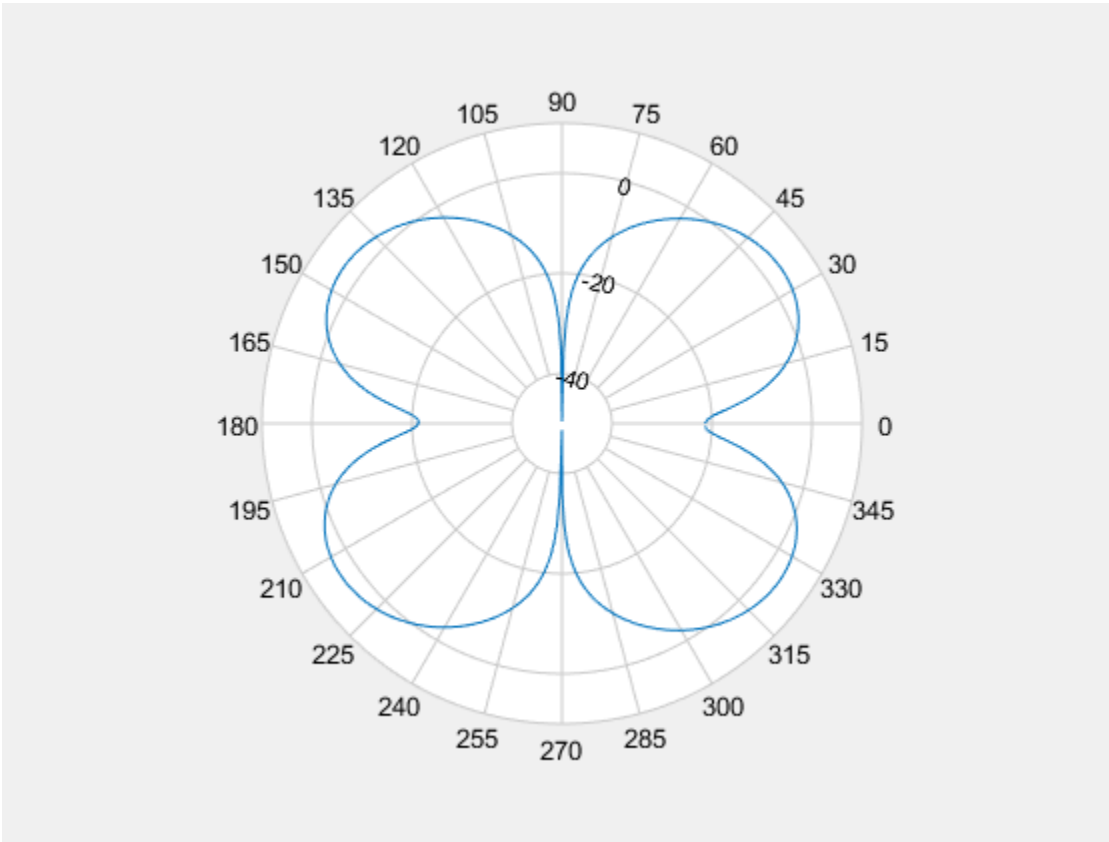
Find Main, Back, and Side Lobes

Create a dipole antenna and calculate its directivity at 270 MHz.

```
d = dipole;
D = pattern(d,270e6,0,0:1:360);
```

Create a polar plot of the dipole directivity. Find the main, back, and side lobes of the dipole antenna.

```
p = polarpattern(D);
```




```
L = findLobes(p)
```

```
L = struct with fields:
     mainLobe: [1x1 struct]
     backLobe: [1x1 struct]
    sideLobes: [1x1 struct]
           FB: 0.0139
          SLL: 0
         HPBW: 30.9141
         FNBW: 89.7507
        FBIdx: [146 326.5000]
       SLLIdx: [146 36]
      HPBWIdx: [129 160]
      HPBWAng: [127.6454 158.5596]
      FNBWIdx: [91 181]
```

Inspect main, back, and side lobe data.

```
MainLobe = L.mainLobe
```

```
MainLobe = struct with fields:
        index: 146
    magnitude: 3.6685
        angle: 144.5983
       extent: [91 181]
```

```
BackLobe = L.backLobe
```

```
BackLobe = struct with fields:
    magnitude: 3.6546
        angle: -35.4017
       extent: [271 361]
        index: 326.5000
```

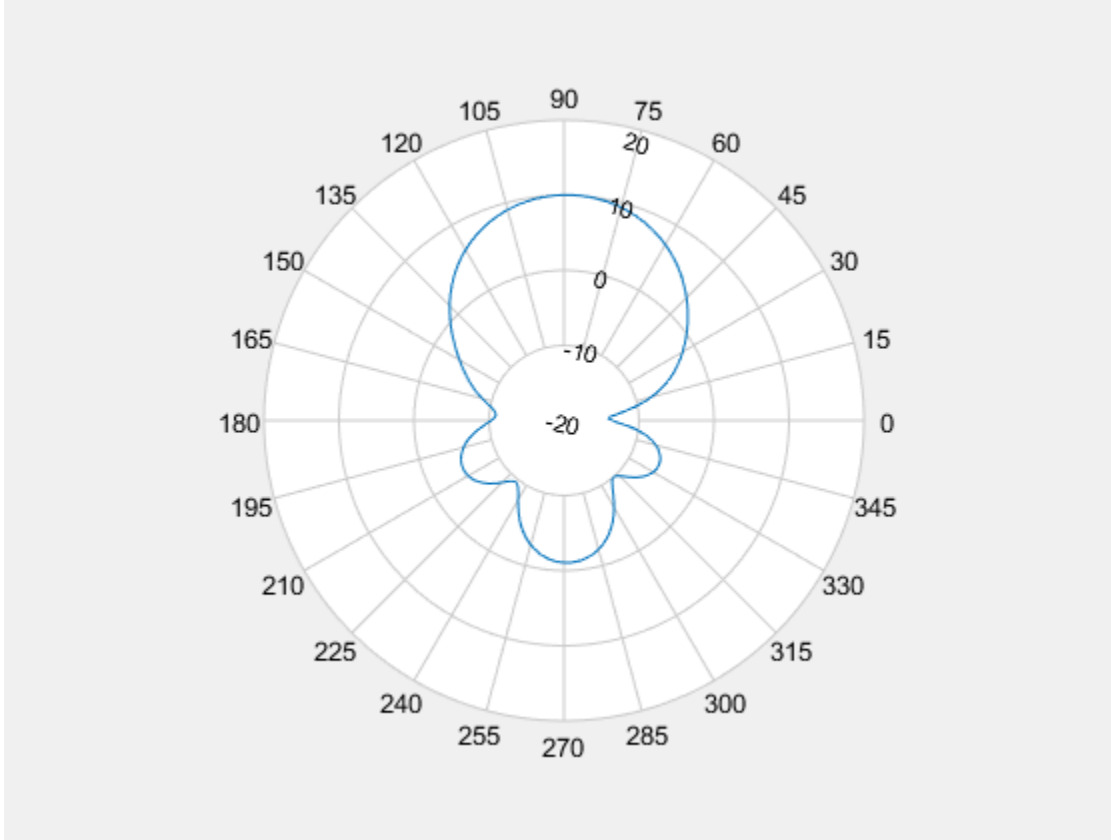
```
SideLobe = L.sideLobes
```

```
SideLobe = struct with fields:
        index: 36
    magnitude: 3.6685
        angle: 34.9030
       extent: [2x2 double]
```

Find Lobes in Two Data Sets

Create a helix antenna that has a 28 mm radius, a 1.2 mm width, and 4 turns. Calculate and plot the directivity of the antenna at 1.8 GHz.

```
hx = helix('Radius',28e-3,'Width',1.2e-3,'Turns',4);
H = pattern(hx, 1.8e9,0,0:1:360);
P = polarpattern(H);
```

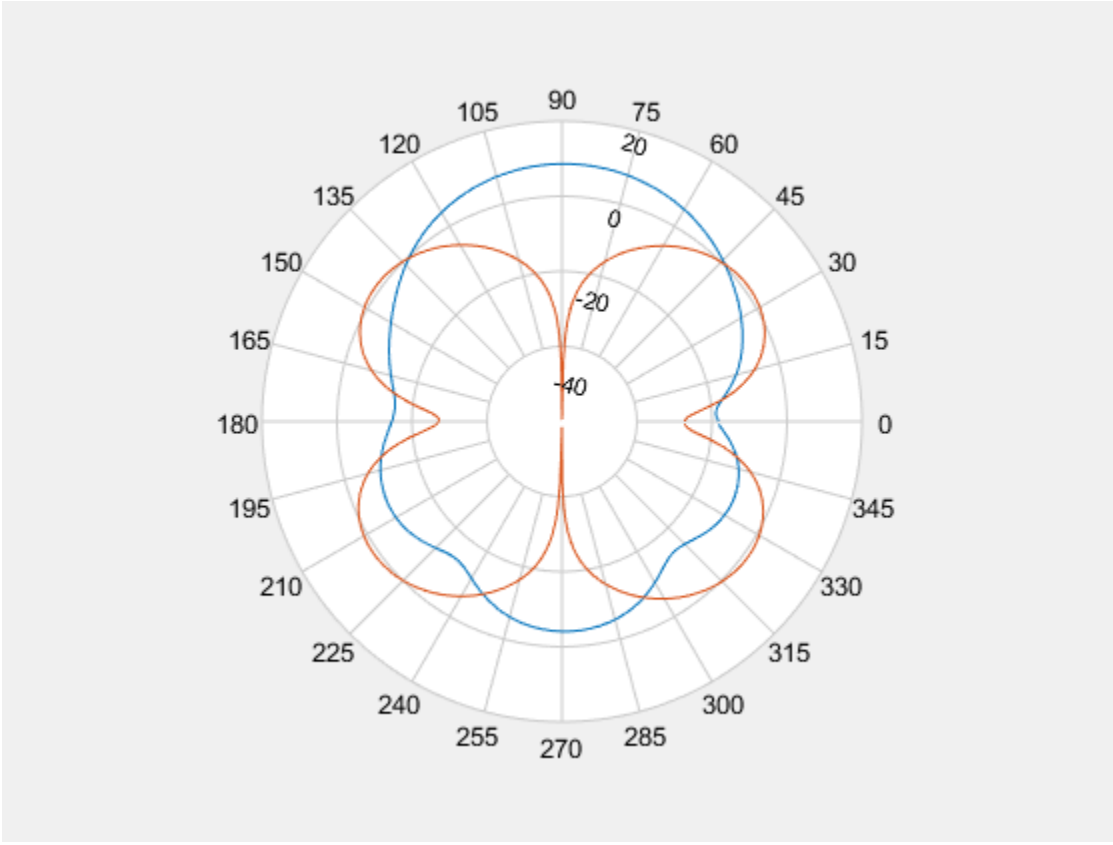


Create a dipole antenna and calculate the directivity at 270 MHz.

```
d = dipole;
D = pattern(d,270e6,0,0:1:360);
```

Add the directivity of the dipole to the existing polar plot.

```
add(P,D);
```



Find the main, back, and side lobes of helix antenna.

```
L = findLobes(P,1)
```

```
L = struct with fields:
     mainLobe: [1x1 struct]
     backLobe: [1x1 struct]
    sideLobes: [1x1 struct]
           FB: 11.1523
          SLL: 11.0997
         HPBW: 56.8421
         FNBW: 172.5208
        FBIdx: [90 270.5000]
```

```
 SLLIdx: [90 273]
HPBWIdx: [61 118]
HPBWAng: [59.8338 116.6759]
FNBWIdx: [4 177]
```

See Also

add | addCursor | animate | createLabels | replace | showPeaksTable | showSpan

Introduced in R2016a

replace

Class: polarpattern

Replace polar plot data with new data

Syntax

```
replace(p,data)
replace(p,angle,magnitude)
```

Description

`replace(p,data)` removes all data from polar plot, `p` and adds new data based on real amplitude values, `data`.

`replace(p,angle,magnitude)` removes all the current data and adds new data sets of angle vectors and corresponding magnitude matrices to the polar plot, `p`.

Input Arguments

p — Polar plot

scalar handle

Polar plot, specified as a scalar handle.

data — Antenna or array data

real length-*M* vector | real *M*-by-*N* matrix | real *N*-*D* array | complex vector or matrix

Antenna or array data, specified as one of the following:

- A real length-*M* vector, where *M* contains the magnitude values with angles assumed to be $\frac{(0:M-1)}{M}\times 360^{\circ}$ degrees.

- A real *M*-by-*N* matrix, where *M* contains the magnitude values and *N* contains the independent data sets. Each column in the matrix has angles taken from the vector $\frac{(0:M-1)}{M}\times 360^\circ$ degrees. The set of each angle can vary for each column.
- A real *N*-*D* array, where *N* is the number of dimensions. Arrays with dimensions 2 and greater are independent data sets.
- A complex vector or matrix, where `data` contains Cartesian coordinates (*(x,y)* of each point. *x* contains the real part of `data` and *y* contains the imaginary part of `data`.

When data is in a logarithmic form such as dB, magnitude values can be negative. In this case,`polarpattern` plots the lowest magnitude values at the origin of the polar plot and highest magnitude values at the maximum radius.

`angle` — Set of angles
vector in degrees

Set of angles, specified as a vector in degrees.

`magnitude` — Set of magnitude values
vector | matrix

Set of magnitude values, specified as a vector or a matrix. For a matrix of magnitude values, each column is an independent set of magnitude values and corresponds to the same set of angles.

Examples

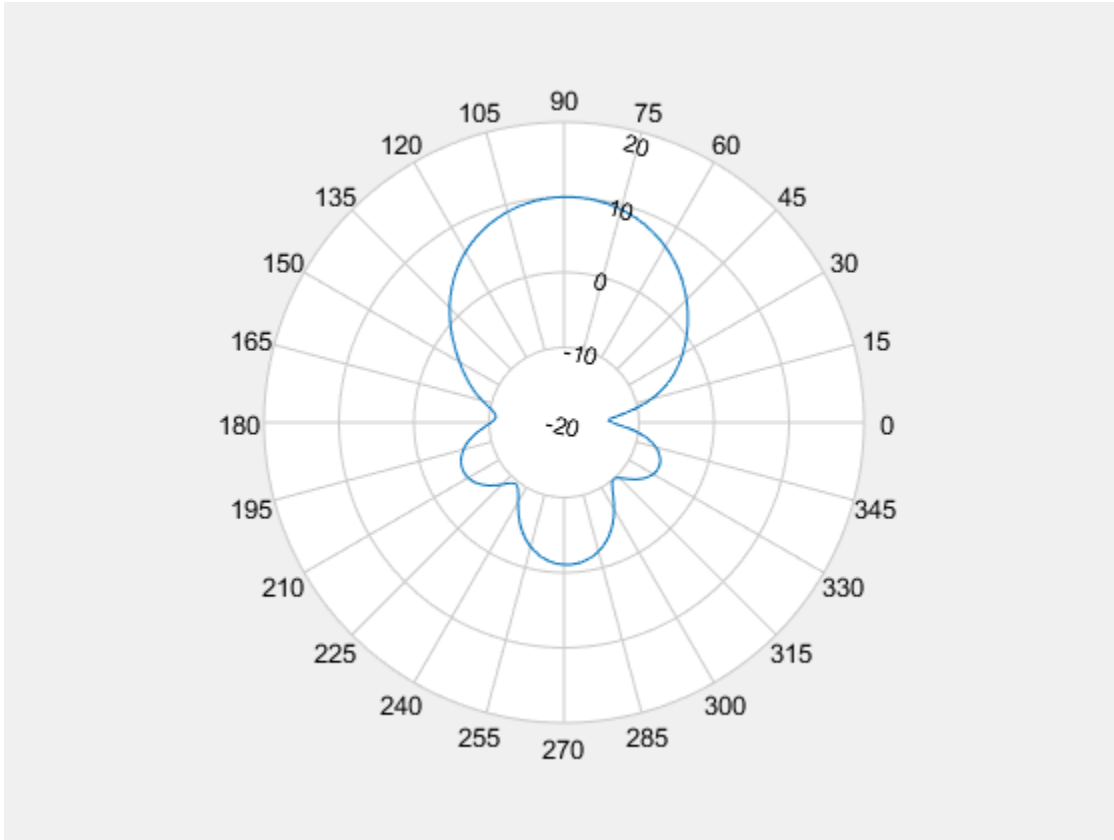
Replace Polar Plot Data with New Data

Create a helix antenna that has a 28 mm radius, a 1.2 mm width, and 4 turns. Calculate the directivity of the antenna at 1.8 GHz.

```
hx = helix('Radius',28e-3,'Width',1.2e-3,'Turns',4);
H = pattern(hx, 1.8e9,0,0:1:360);
```

Plot the polar pattern.

```
P = polarpattern(H);
```

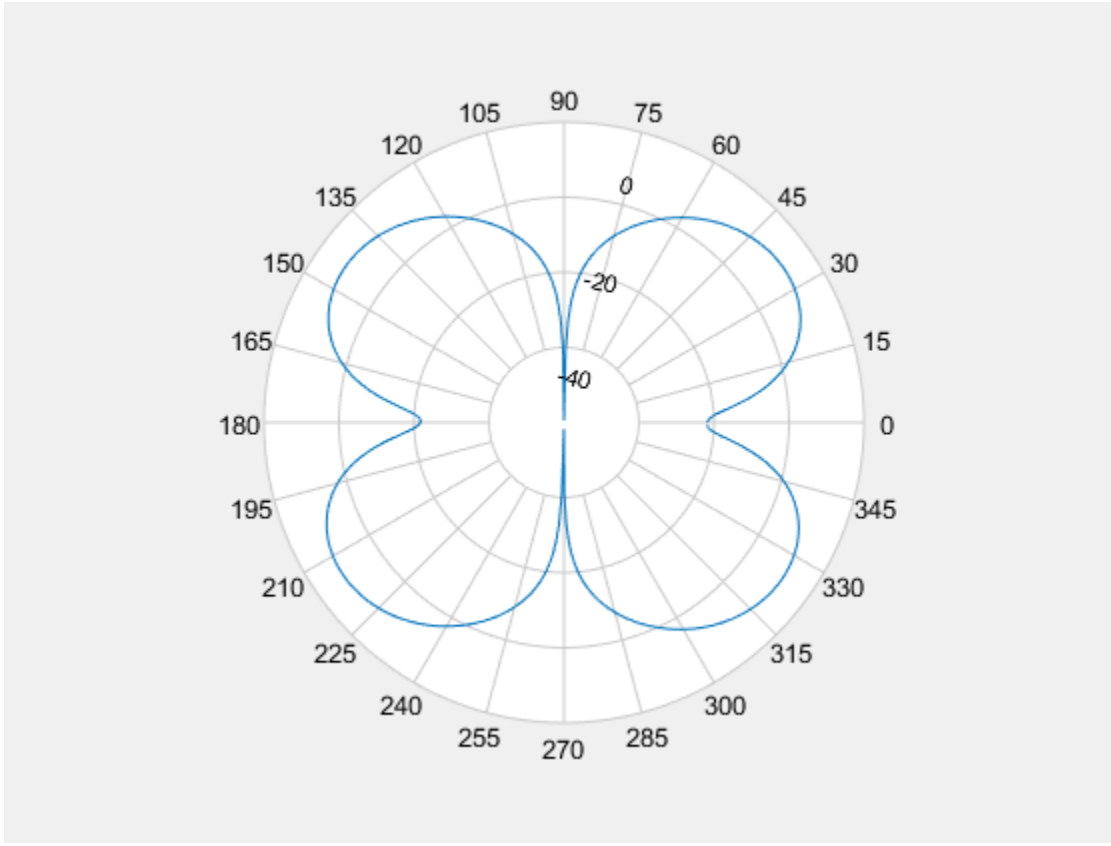


Create a dipole antenna and calculate its directivity at 270 MHz.

```
d = dipole;
D = pattern(d,270e6,0,0:1:360);
```

Replace the existing polar plot of the helix antenna with the directivity of the dipole.

```
replace(P,D);
```



See Also

`add` | `addCursor` | `animate` | `createLabels` | `findLobes` | `showPeaksTable` | `showSpan`

Introduced in R2016a

showPeaksTable

Class: polarpattern

Show or hide peak marker table

Syntax

```
showPeaksTable(p,vis)
```

Description

`showPeaksTable(p,vis)` shows or hides a table of the peak values. By default, the peak values table is visible.

Input Arguments

`p` — Polar plot
scalar handle

Polar plot, specified as a scalar handle.

`vis` — Show or hide peaks table
`0` | `1`

Show or hide peaks table, specified as `0` or `1`.

Examples

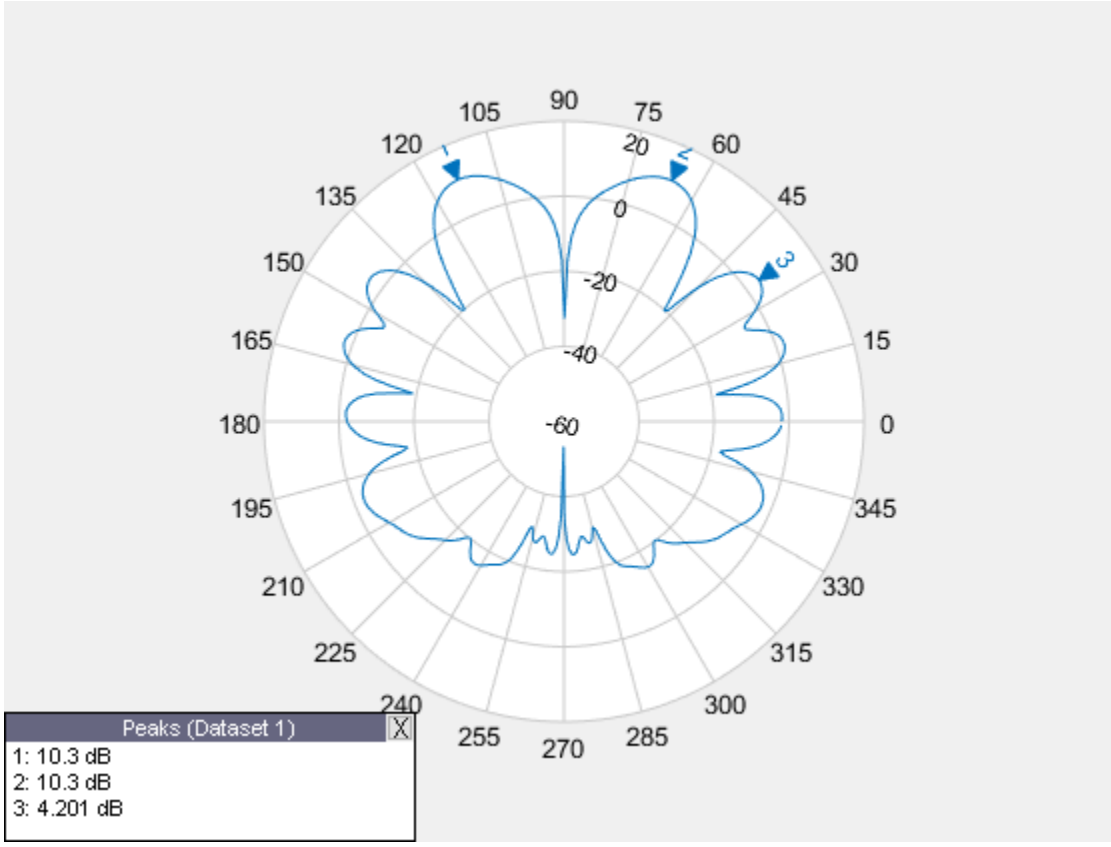
Peaks of Antenna in Polar Pattern

Create a monopole antenna and calculate the directivity at 1 GHz.

```
m = monopole;
M = pattern(m,1e9,0,0:1:360);
```

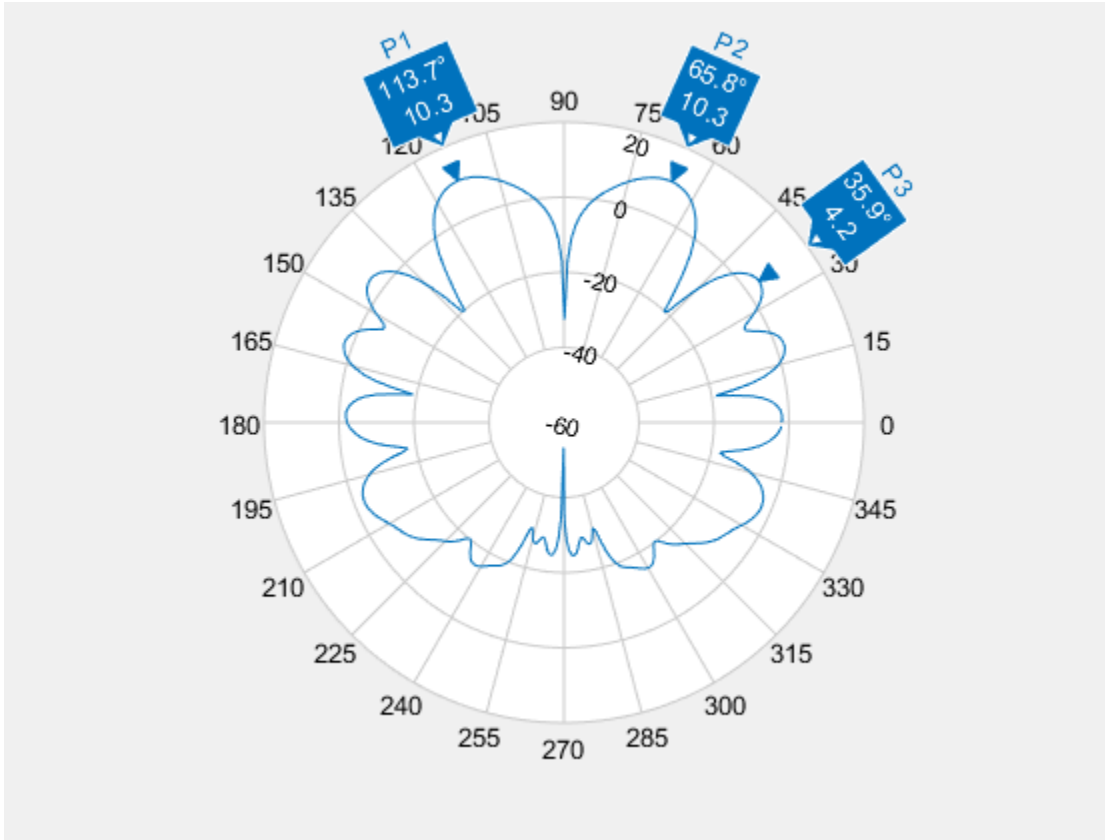
Plot the polar pattern and show three peaks of the antenna. When creating a `polarpattern` plot, if you specify the Peaks property, the peaks table is displayed by default.

```
P = polarpattern(M,'Peaks',3);
```



Hide the table. When the peaks table is hidden, the peak markers display the peak values.

```
showPeaksTable(P,0);
```



See Also

add | addCursor | animate | createLabels | findLobes | replace | showSpan

Introduced in R2016a

showSpan

Class: polarpattern

Show or hide angle span between two markers

Syntax

```
showSpan(p,id1,id2)
showSpan(p,id1,id2,true)
showSpan(p,vis)
showSpan(p)
d = showSpan( ___ )
```

Description

`showSpan(p,id1,id2)` displays the angle span between two angle markers, `id1` and `id2`. The angle span is calculated counterclockwise.

`showSpan(p,id1,id2,true)` automatically reorders the angle markers such that the initial angle span is less than or equal to 180° counterclockwise.

`showSpan(p,vis)` sets angle span visibility by setting `vis` to `true` or `false`.

`showSpan(p)` toggles the angle span display on and off.

`d = showSpan( ___ )` returns angle span details in a structure, `d` using any of the previous syntaxes.

Input Arguments

p — Polar plot

scalar handle

Polar plot, specified as a scalar handle.

`id1,id2` — Cursor or peak marker identifiers
character vector

Cursor or peak marker identifiers, specified as character vector. Adding cursors to the polar plot creates cursor marker identifiers. Adding peaks to the polar plot creates peak marker identifiers.

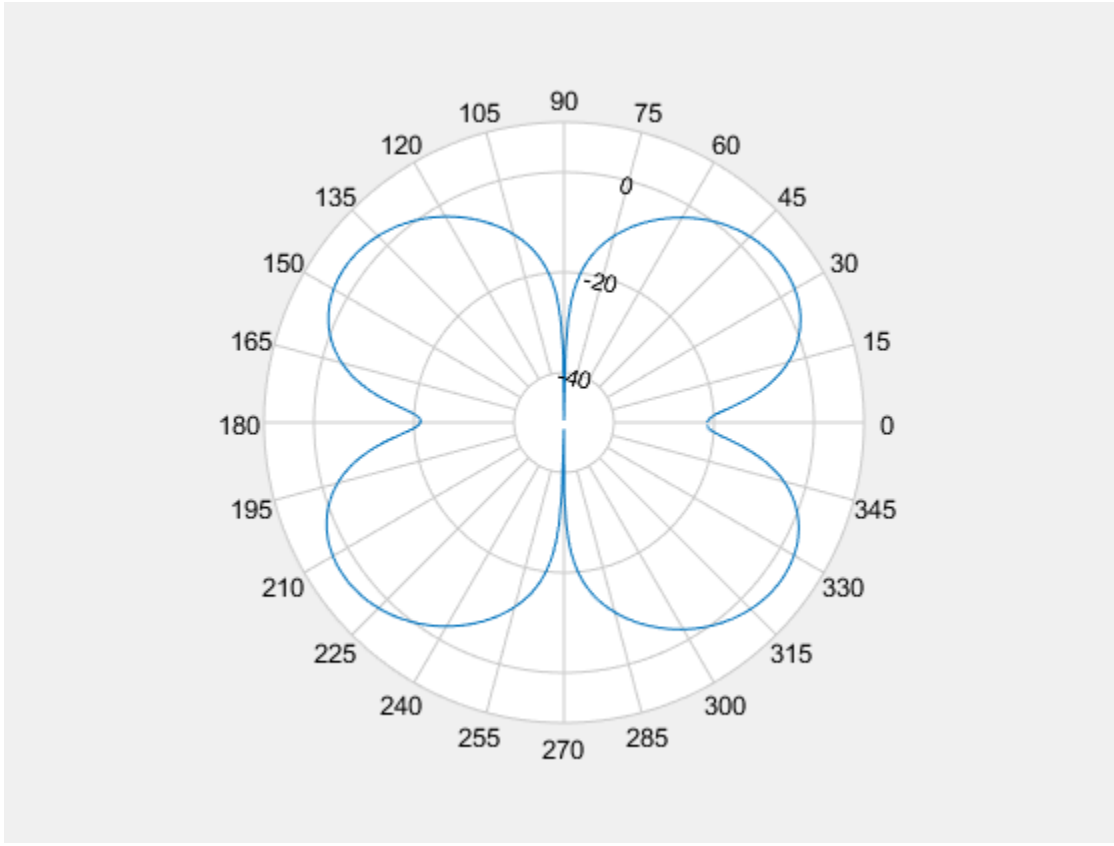
Example: `showspan(p,'C1','C2')`. Displays the angle span between cursors, `C1` and `C2` in polar plot, `p`.

Examples

Show Angle Span

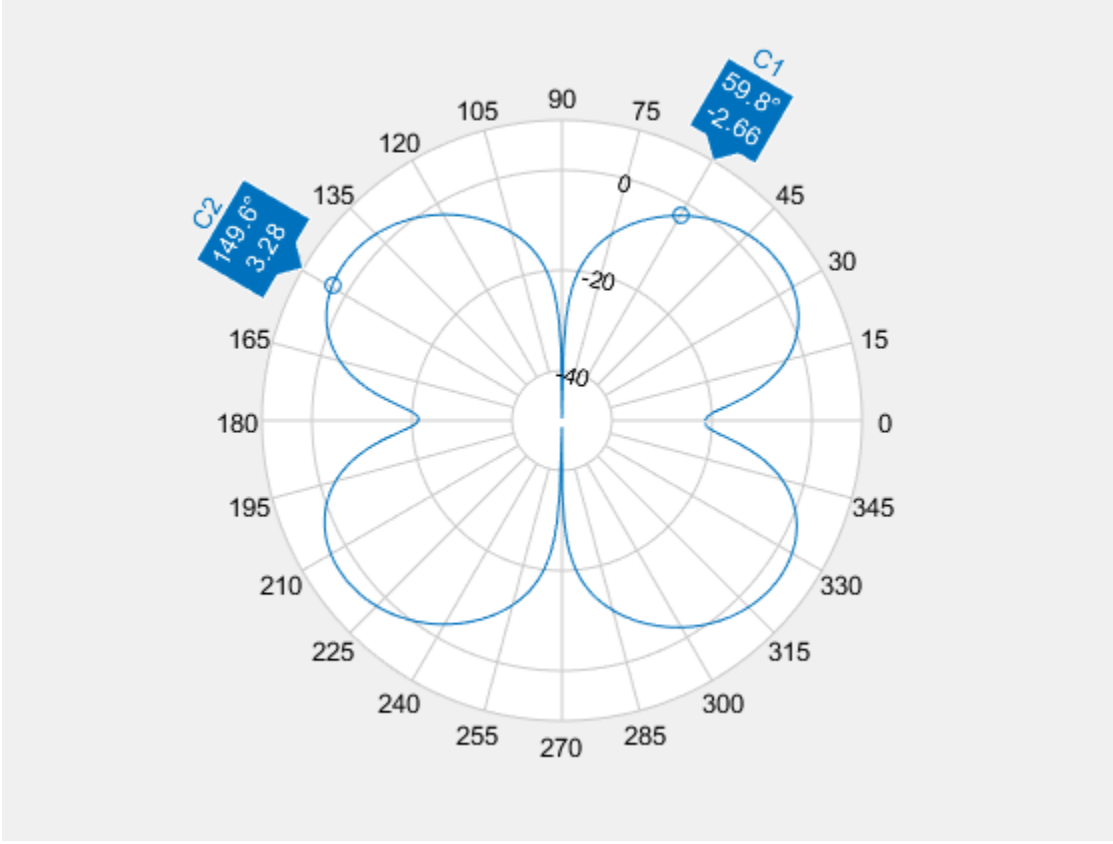
Create a dipole antenna and plot the directivity at 270 MHz.

```
d = dipole;
D = pattern(d,270e6,0,0:1:360);
p = polarpattern(D);
```



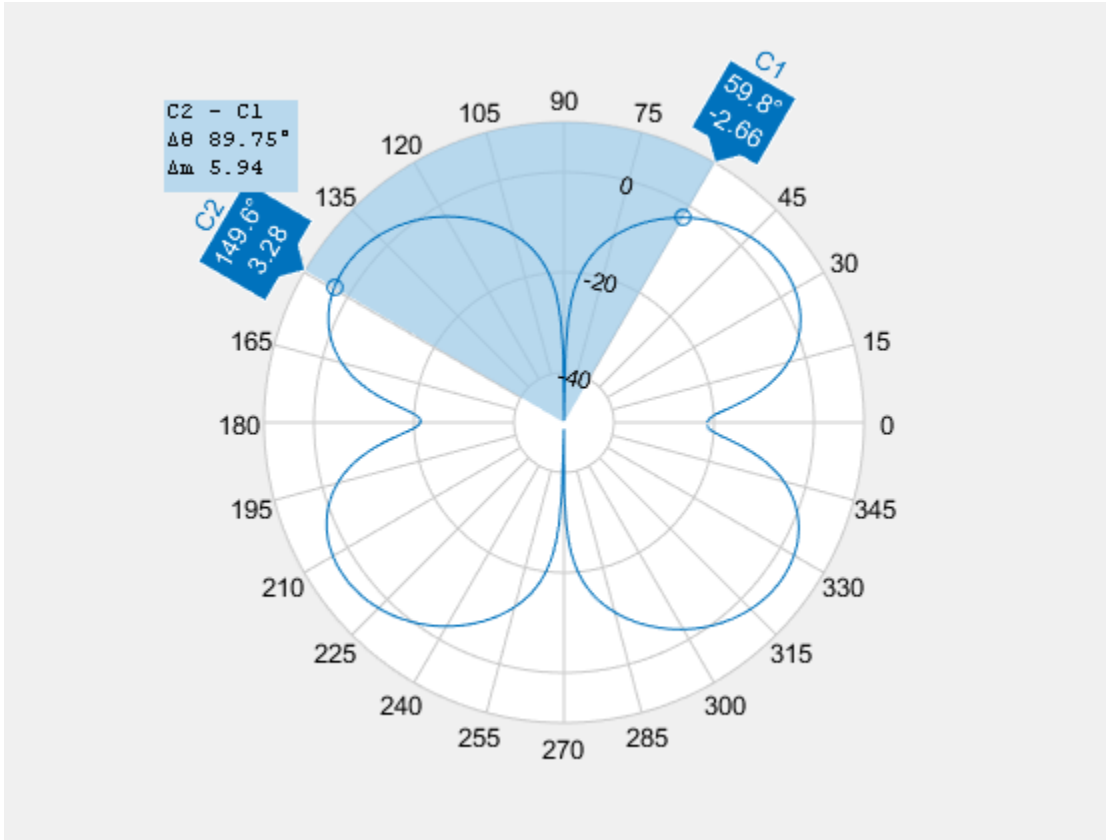
Add cursors to the polar plot at approximately 60 and 150 degrees.

```
addCursor(p,[60 150]);
```



Show the angle span between the two angles.

```
showSpan(p,'C1','C2');
```



See Also

add | addCursor | animate | createLabels | findLobes | replace | showPeaksTable

Introduced in R2016a

arrayFactor

Array factor in dB

Syntax

```
arrayFactor(object,frequency)
arrayFactor(object,frequency,azimuth,elevation)
arrayFactor( ___ ,Name,Value)

[af] = arrayFactor(object,frequency)
[af,azimuth,elevation] = arrayFactor( ___ )
[af,azimuth,elevation] = arrayFactor( ___ ,Name,Value)
```

Description

`arrayFactor(object,frequency)` plots the 3-D array factor over the specified frequency value in dB.

`arrayFactor(object,frequency,azimuth,elevation)` plots the array factor over the specified frequency, azimuth, and elevation values.

`arrayFactor( ___ ,Name,Value)` plots the array factor using additional options specified by one or more `Name,Value` pair arguments. Specify name-value pair arguments after all other input arguments.

`[af] = arrayFactor(object,frequency)` returns the 3-D array factor over the specified frequency value.

`[af,azimuth,elevation] = arrayFactor( ___ )` returns the array factor at the specified frequency, azimuth, and elevation values.

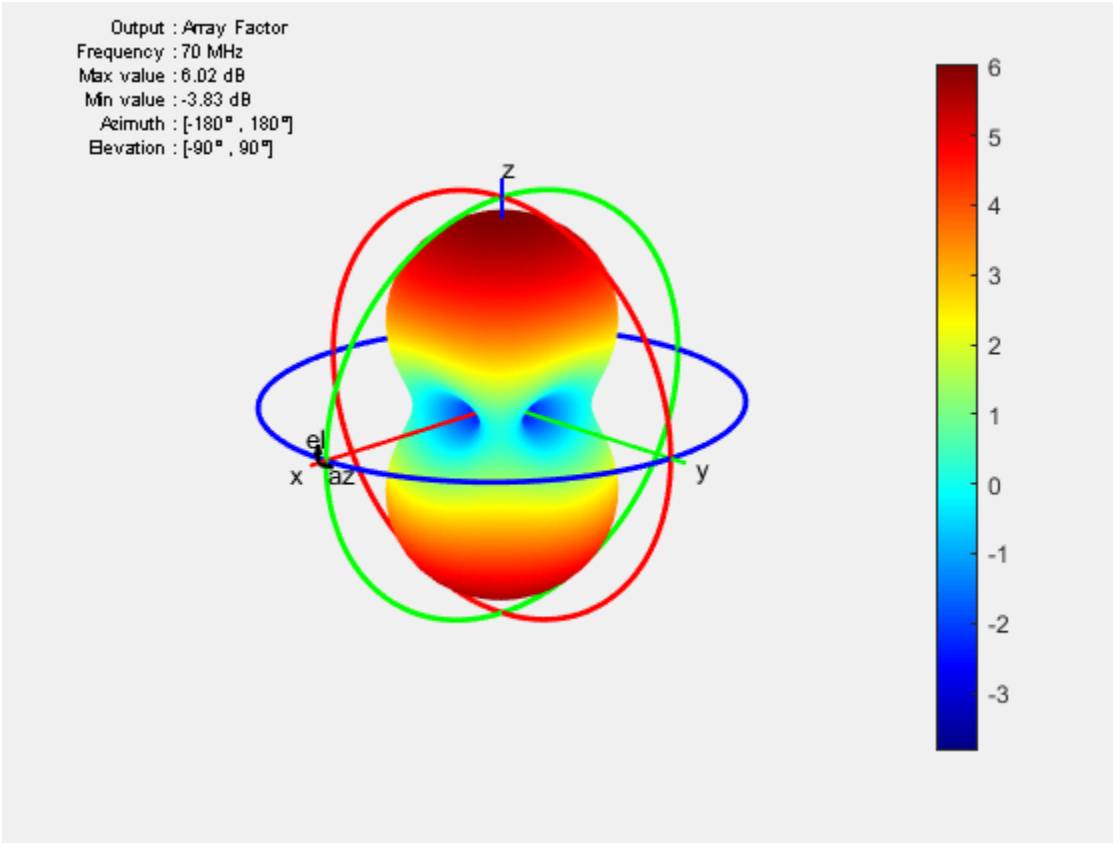
`[af,azimuth,elevation] = arrayFactor( ___ ,Name,Value)` returns the array factor using additional options specified by one or more `Name,Value` pair arguments. Specify name-value pair arguments after all other input arguments.

Examples

Plot Array Factor

Plot the array factor of a default rectangular array at a frequency of 70 MHz.

```
ra = rectangularArray;
arrayFactor(ra,70e6);
```



Input Arguments

`object` — Input antenna array
object handle

Input antenna array object, specified as an object handle.

Example: `r = rectangularArray; arrayFactor (r,70e6)`. Calculates the array factor of a rectangular array.

`frequency` — Frequency value used to calculate array factor
scalar in Hz

Frequency value used to calculate array factor, specified as a scalar in Hz.

Example: `70e6`

Data Types: `double`

`azimuth` — Azimuth angle of antenna
`–180:5:180` (default) | vector in degrees

Azimuth angle of the antenna, specified as a vector in degrees.

Example: `–90:5:90`

Data Types: `double`

`elevation` — Elevation angle of antenna
`–90:5:90` (default) | vector in degrees

Elevation angle of the antenna, specified as a vector in degrees.

Example: `0:1:360`

Data Types: `double`

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` pair arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside single quotes (`''`). You can specify several name and value pair arguments in any order as `Name1, Value1, ..., NameN, ValueN`.

Example: `'CoordinateSystem'`, `rectangular`

`CoordinateSystem` — Coordinate system of array factor
`'polar'` (default) | `'rectangular'` | `'uv'`

Coordinate system of array factor, specified as the comma-separated pair consisting of `'CoordinateSystem'` and one of these values: `'polar'`, `'rectangular'`, `'uv'`.

Example: `'CoordinateSystem','polar'`

Data Types: `char`

Output Arguments

`af` — Array factor
matrix in `dB`

Array factor, returned as a matrix in `dB`. The matrix size is the product of number of elevation values and number of azimuth values.

`azimuth` — Azimuth values
vector in degrees

Azimuth values used to calculate the array factor, returned as a vector in degrees.

`elevation` — Elevation values
vector in degrees

Elevation values used to calculate the array factor, returned as a vector in degrees.

See Also
`feedCurrent` | `pattern` | `patternMultiply`

Introduced in R2017a

angle

Angle between sites

Syntax

```
[az,el] = angle(site1,site2)
[az,el] = angle(site1,site2,path)
```

Description

`[az,el] = angle(site1,site2)` returns the azimuth and elevation angles between site 1 and site 2.

`[az,el] = angle(site1,site2,path)` returns the angles using a specified path type, either Euclidean or geodesic.

Examples

Angle Between Sites

Create transmitter and receiver sites.

```
tx = txsite('Name','MathWorks','Latitude',42.3001,'Longitude',-71.3504);
rx = rxsite('Name','Fenway Park','Latitude',42.3467,'Longitude',-71.0972);
```

Get the azimuth and elevation angles between the sites.

```
[az,el] = angle(tx,rx)
```

```
az = 14.0142
```

```
el = -0.1204
```

Get the azimuth angle between sites in degrees clockwise from north.

```
azFromEast = angle(tx,rx); % Unit: degrees counter-clockwise from east
azFromNorth = -azFromEast + 90 % Convert angle to clockwise from north
```

```
azFromNorth = 75.9858
```

Angle Between Sites When Path is Geodesic

Create transmitter and receiver sites.

```
tx = txsite('Name','MathWorks','Latitude',42.3001,'Longitude',-71.3504);
rx = rxsite('Name','Fenway Park','Latitude',42.3467,'Longitude',-71.0972);
```

Get the azimuth and elevation angles between the sites.

```
[az,el] = angle(tx,rx,'geodesic')
```

```
az = 14.0142
```

```
el = 0
```

Input Arguments

`site1,site2` — Transmitter or receiver site

`txsite` or `rxsite` object

Transmitter or receiver site, specified as a `txsite` or `rxsite` object. You can use array inputs to specify multiple sites.

`path` — Measurement path type

`'euclidean'` or `'geodesic'`

Measurement path type, specified as one of the following:

- `'euclidean'`: Uses the shortest path through space connecting the antenna center positions of the site 1 and site 2.
- `'geodesic'`: Uses the shortest path on the surface of the earth connecting the latitude and longitude locations of site 1 and site 2. This path uses Earth ellipsoid model WGS-84.

Data Types: `char`

Output Arguments

az — Azimuth angle between site 1 and site 2

M-by-*N* arrays

Azimuth angle between site 1 and site 2, returned as *M*-by-*N* arrays in degrees. *M* is the number of sites in sites 2 and *N* is the number of sites in sites 1. The values range from `-180` to `180`.

el — Elevation angle between site 1 and site 2

M-by-*N* arrays

Elevation angle between site 1 and site 2, returned as *M*-by-*N* arrays in degrees. *M* is the number of sites in sites 2 and *N* is the number of sites in sites 1 The values range from `-90` to `90`.

When the path type specified is `'geodesic'`, elevation angle is always zero.

See Also

`distance`

Introduced in R2017b

coverage

Display coverage map

Syntax

```
coverage(tx)
coverage(tx,propmodel)
coverage(tx,rx)
coverage(tx,rx,propmodel)
coverage( ___ ,Name,Value, ___ )
```

Description

`coverage(tx)` displays the coverage map for the transmitter site. Each colored contour of the map defines an area where the corresponding signal strength is transmitted to the mobile receiver.

`coverage(tx,propmodel)` displays the coverage map based on the specified propagation model.

`coverage(tx,rx)` displays the coverage map based on the receiver site properties.

`coverage(tx,rx,propmodel)` displays the coverage map based on the receiver site properties and specified propagation model.

`coverage( ___ ,Name,Value, ___ )` displays the coverage map using additional options specified by the `Name,Value` pairs.

Examples

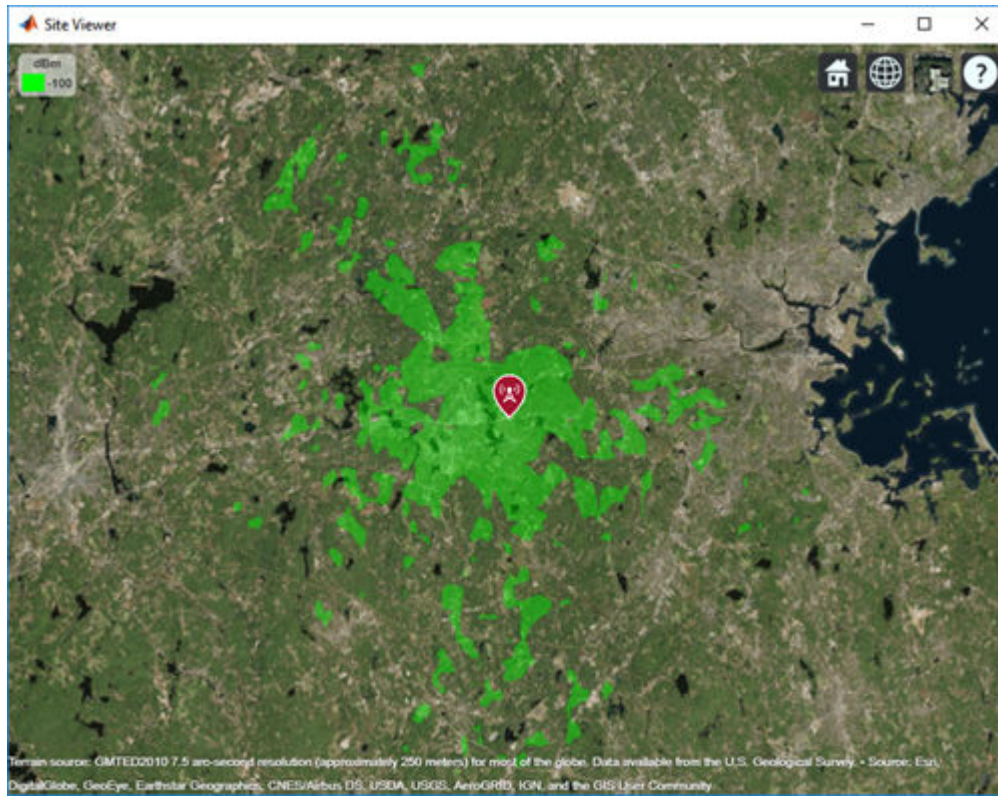
Coverage Map of Transmitter

Create a transmitter site at MathWorks headquarters.

```
tx = txsite('Name','MathWorks', ...
        'Latitude', 42.3001, ...
        'Longitude', -71.3503);
```

Show the coverage map.

```
coverage(tx)
```



Coverage Map Using Transmitter and Receiver.

Create a transmitter site at MathWorks headquarters.

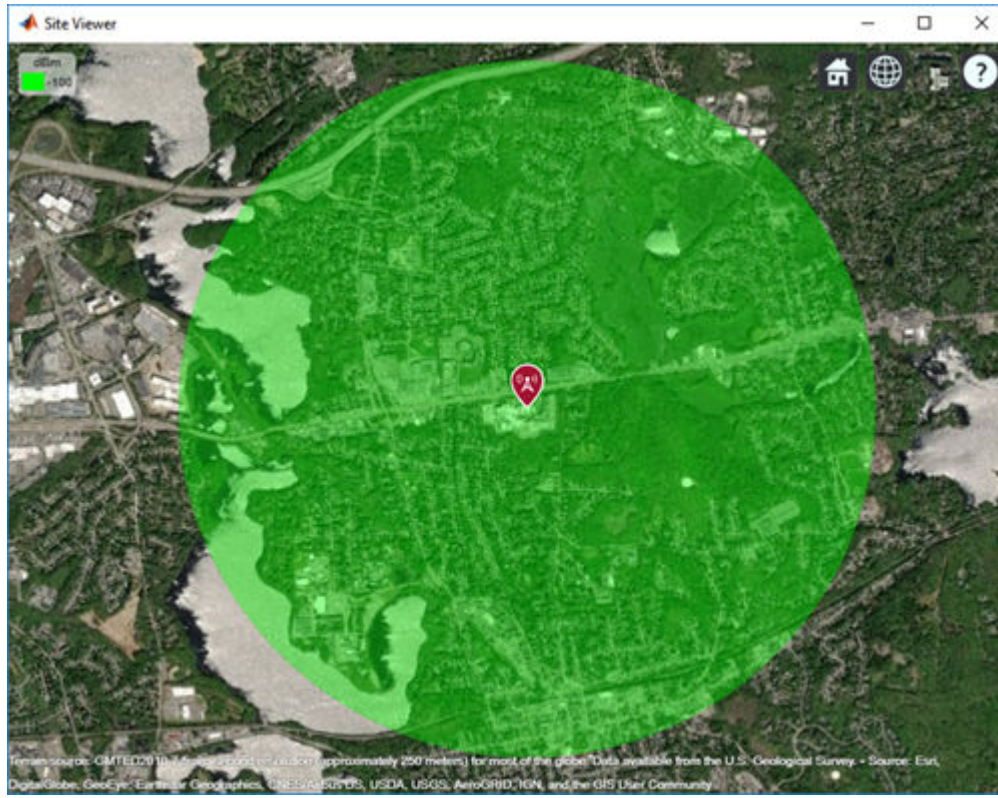
```
tx = txsite('Name','MathWorks', ...
        'Latitude', 42.3001, ...
        'Longitude', -71.3503);
```

Create a receiver site at Fenway Park with an antenna height of 1.2 m and system loss of 10 dB.

```
rx = rxsite('Name','Fenway Park', ...
       'Latitude',42.3467, ...
       'Longitude',-71.0972,'AntennaHeight',1.2,'SystemLoss',10);
```

Calculate the coverage area of the transmitter using a close-in propagation model.

```
coverage(tx,rx,'PropagationModel','closein')
```



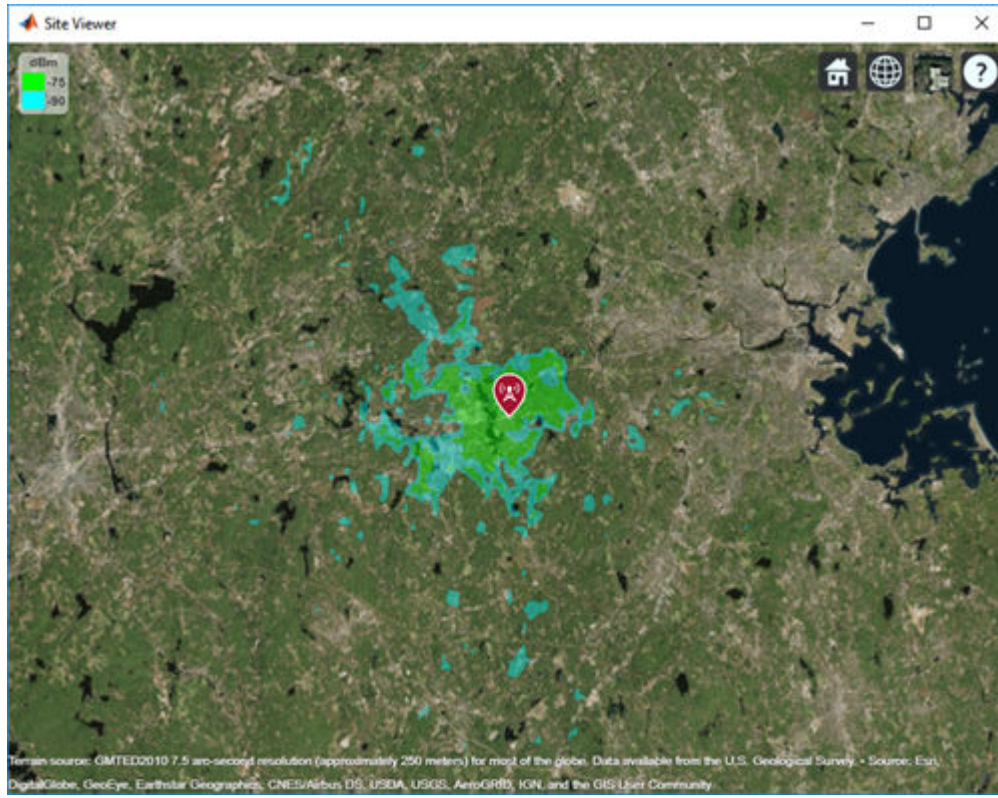
Coverage Map for Strong and Weak Signals

Define strong and weak signal strengths with corresponding colors.

```
strongSignal = -75;
strongSignalColor = "green";
weakSignal = -90;
weakSignalColor = "cyan";
```

Create a transmitter site and display the coverage map.

```
tx = txsite('Name','MathWorks','Latitude', 42.3001,'Longitude', -71.3503);
coverage(tx,'SignalStrengths',[strongSignal,weakSignal], ...
       'Colors', [strongSignalColor,weakSignalColor])
```



Coverage Map of Directional Antenna in Rain

Define a Yagi-Uda antenna designed for a transmitter frequency of 4.5 GHz. Tilt the antenna to direct radiation in the XY-plane (i.e., geographic azimuth).

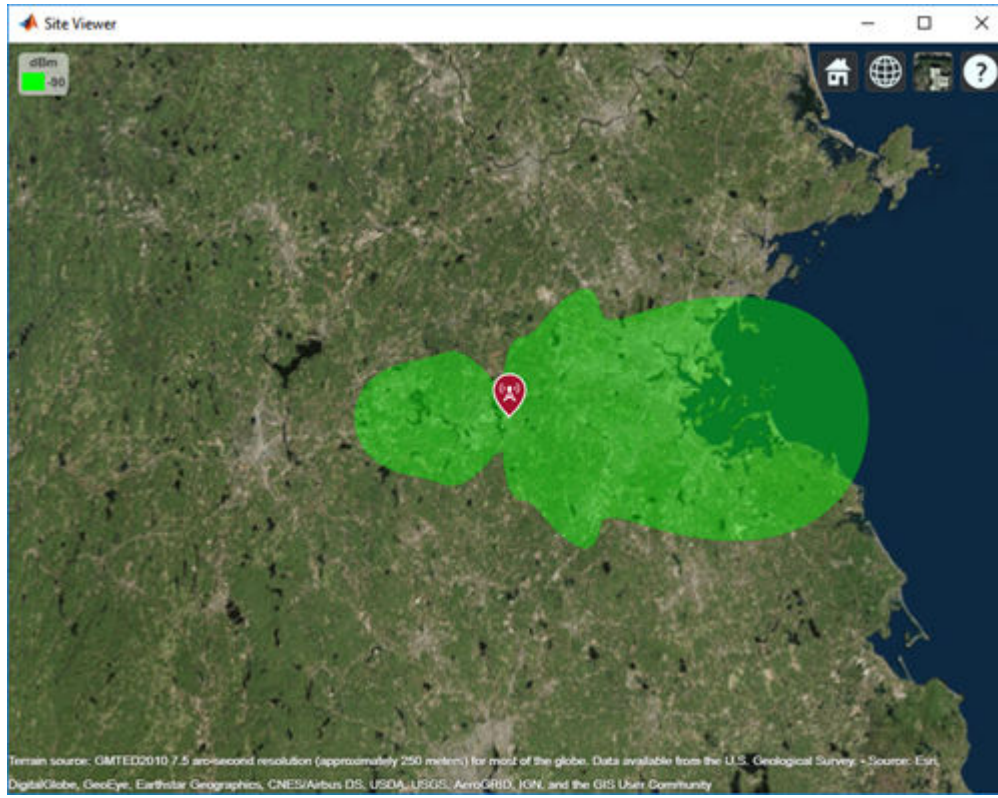
```
fq = 4.5e9;
y = design(yagiUda,fq);
y.Tilt = 90;
y.TiltAxis = 'y';
```

Create a transmitter site with this directional antenna.

```
tx = txsite('Name','MathWorks',...
        'Latitude', 42.3001, ...
        'Longitude', -71.3503, ...
        'Antenna', y, ...
        'AntennaHeight', 60, ...
        'TransmitterFrequency', fq, ...
        'TransmitterPower', 10);
```

Display the coverage map using the rain propagation model. The map pattern points east, which corresponds to default antenna angle value of 0 degrees.

```
coverage(tx,'rain','SignalStrengths',-90)
```



Combined Coverage Map of Multiple Transmitters

Define the names and the locations of sites around Boston.

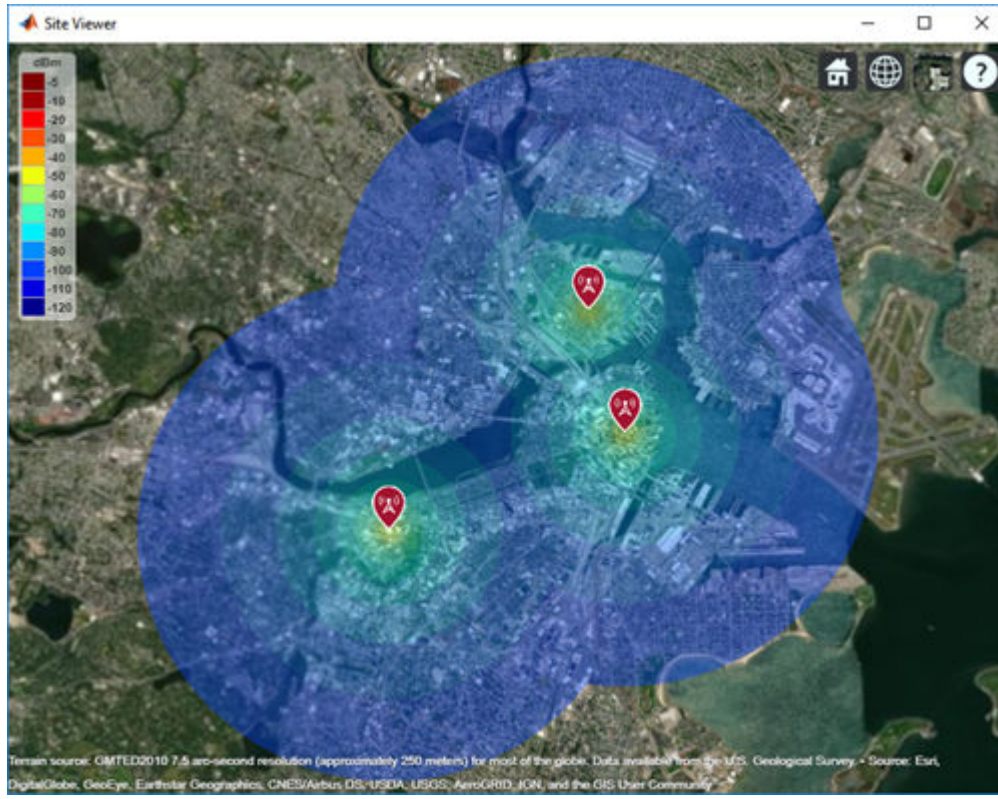
```
names = ["Fenway Park","Faneuil Hall","Bunker Hill Monument"];
lats = [42.3467,42.3598,42.3763];
lons = [-71.0972,-71.0545,-71.0611];
```

Create the transmitter site array.

```
txs = txsite('Name', names,...
       'Latitude',lats,...
       'Longitude',lons, ...
       'TransmitterFrequency',2.5e9);
```

Display the combined coverage map for multiple signal strengths, using close-in propagation model.

```
coverage(txs,'close-in','SignalStrengths',-100:5:-60)
```



Input Arguments

`tx` — Transmitter site

`txsite` object | array of `txsite` objects

Transmitter site, specified as a `txsite` object. You can use array inputs to specify multiple sites.

`rx` — Receiver site
`rxsite` object

Receiver site, specified as an `rxsite` object. You can also use the name-value pairs `'ReceiverGain'` and `'ReceiverAntennaHeight'` to specify the receiver values.

`propmodel` — Propagation model
character vector | string

Propagation model, specified as a character vector or string. You can also use the name-value pair `'PropagationModel'` to specify this parameter. You can also use the `propagationModel` function to define this input.

Data Types: `char` | `string`

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `'Type','power'`

`Type` — Type of signal strength to compute
`'power'` (default) | `'efield'`

Type of signal strength to compute, specified as `'power'` or `'efield'`.

Power is expressed in power units (dBm) of the signal at the receiver input. E-field is expressed in electric field strength units (dBµV/m) of signal wave incident on the antenna.

Data Types: `char`

`SignalStrengths` — Signal strengths to display on coverage map
numeric vector

Signal strengths to display on coverage map, specified as a numeric vector.

Each strength uses different colored filled contour on the map. The default value is `-100` dBm if the `'Type'` name-value pair is `'power'` and `40` dBµV/m if `'Type'` is `'efield'`.

Note Signal strength is computed using a propagation model that disregards the curvature of the Earth, terrain, or other obstacles.

Data Types: `char`

`PropagationModel` — Propagation model to use for path loss calculations
`'freespace'` (default) | `'close-in'` | `'rain'` | `'gas'` | `'fog'` | `'longley-rice'` | propagation model object

Propagation model to use for the path loss calculations, specified as `'freespace'`, `'close-in'`, `'rain'`, `'gas'`, `'fog'`, `'longley-rice'`, or as an object created using the `propagationModel` function.

Data Types: `char`

`MaxRange` — Maximum range of coverage map from each transmitter site
numeric scalar

Maximum range of the coverage map from each transmitter site, specified as a numeric scalar in meters. If using a terrain propagation model, the default value is 30000 m or 30 km. If using non-terrain propagation model, the default value is computed to include all `SignalStrengths` in the coverage map region.

Note When using terrain, the `MaxRange` limit is 300000 m.

Data Types: `double`

`Resolution` — Resolution of coverage map
`'auto'` (default) | numeric scalar

Resolution of coverage map, specified as a numeric scalar in meters.

The resolution of `'auto'` computes the maximum value scaled to `'MaxRange'`. Decreasing the resolution increases the quality of the coverage map and the time required to create it.

Data Types: `char` | `double`

`ReceiverGain` — Mobile receiver gain
`2.1` (default) | numeric scalar

Mobile receiver gain, specified as a numeric scalar in dB. The receiver gain value includes the mobile receiver antenna gain and system loss.

The receiver gain computes received signal strength when the `'Type'` is `'power'`.

If receiver site argument `rx` is passed to coverage, the default value is the maximum gain of the receiver antenna with the system loss subtracted. Otherwise the default value is 2.1.

Data Types: `char` | `double`

`ReceiverAntennaHeight` — Mobile receiver antenna height above ground elevation
1 (default) | numeric scalar

Mobile receiver antenna height above ground elevation, specified as a numeric scalar in meters.

If receiver site argument `rx` is passed to coverage, the default value is the `AntennaHeight` of the receiver. Otherwise the default value is 1.

Data Types: `double`

`Colors` — Colors of filled contours on coverage map
M-by-3 array of RGB triplets | array of strings | cell array of character vectors

Filled contours color of coverage map, specified as an *M*-by-3 array of RGB triplets, an array of strings, or a cell array of character vectors.

Colors are assigned element-wise to `'SignalStrengths'` values for coloring the corresponding filled contours.

`'Colors'` cannot be used with `'ColorLimits'` or `'ColorMap'`.

For more information, see `ColorSpec (Color Specification)`.

Data Types: `char` | `string` | `double`

`ColorLimits` — Color limits for colormap
two-element vector

Color limits for colormap, specified as a two-element vector of type `[min max]`.

The color limits indicate the signal level values that map to the first and last colors on the colormap.

The default value is `[-120 -5]` if the `'Type'` name-value pair is `'power'` and `[20 135]` if `'Type'` is `'efields'`.

`'ColorLimits'` cannot be used with `'Color'`.

Data Types: `double`

`ColorMap` — Colormap filled contours for coverage map
`'jet'` (default) | predefined color map | *M*-by-3 array of RGB triplets

Colormap filled contours on coverage map, specified as a predefined colormap or *M*-by-3 array of RGB triplets, where *M* defines individual colors.

`'ColorMap'` cannot be used with `'Colors'`.

Data Types: `char` | `double`

`Transparency` — Transparency of coverage map
`0.4` (default) | numeric scalar

Transparency of coverage map, specified as a numeric scalar in the range `0` to `1`. `0` is transparent and `1` is opaque.

Data Types: `double`

See Also

`link` | `propagationModel` | `sigstrength` | `sinr`

Topics

`ColorSpec (Color Specification)`

Introduced in R2017b

distance

Distance between sites

Syntax

```
d = distance(site1,site2)
d = distance(site1,site2,path)
```

Description

`d = distance(site1,site2)` returns the distance in meters between site1 and site2.

`d = distance(site1,site2,path)` returns the distance using a specified path type, either Euclidean or geodesic.

Examples

Distance Between Transmitter and Receiver Site

Create transmitter and receiver sites.

```
tx = txsite('Name','MathWorks','Latitude',42.3001,'Longitude',-71.3504);
rx = rxsite('Name','Fenway Park','Latitude',42.3467,'Longitude',-71.0972);
```

Get the Euclidean distance in km between the sites.

```
dme = distance(tx,rx)
```

```
dme = 2.1504e+04
```

```
dkm = dme / 1000
```

```
dkm = 21.5038
```

Get the geodesic distance between the two sites.

```
dmg = distance(tx,rx,'geodesic')
```

```
dmg = 2.1504e+04
```

Input Arguments

site1,site2 — Transmitter or receiver site

`txsite` or `rxsite` object

Transmitter or receiver site, specified as a `txsite` or `rxsite`. You can use array inputs to specify multiple sites.

path — Measurement path type

`'euclidean'` | `'geodesic'`

Measurement path type, specified as one of the following:

- `'euclidean'`: Uses the shortest path through space that connects the antenna center positions of the site 1 and site 2.
- `'geodesic'`: Uses the shortest path on the surface of the earth that connects the latitude and longitude locations of site 1 and site 2. This path uses Earth ellipsoid model WGS-84.

Data Types: `char`

Output Arguments

d — Distance between sites

M-by-*N* numeric array

Distance between sites, returned as *M*-by-*N* arrays in degrees. *M* is the number of sites in site 2 and *N* is the number of sites in site 1.

See Also

`angle`

Introduced in R2017b

hide

Hide site location on map

Syntax

```
hide(site)
```

Description

`hide(site)` hides the site location of the antenna site on a map.

Examples

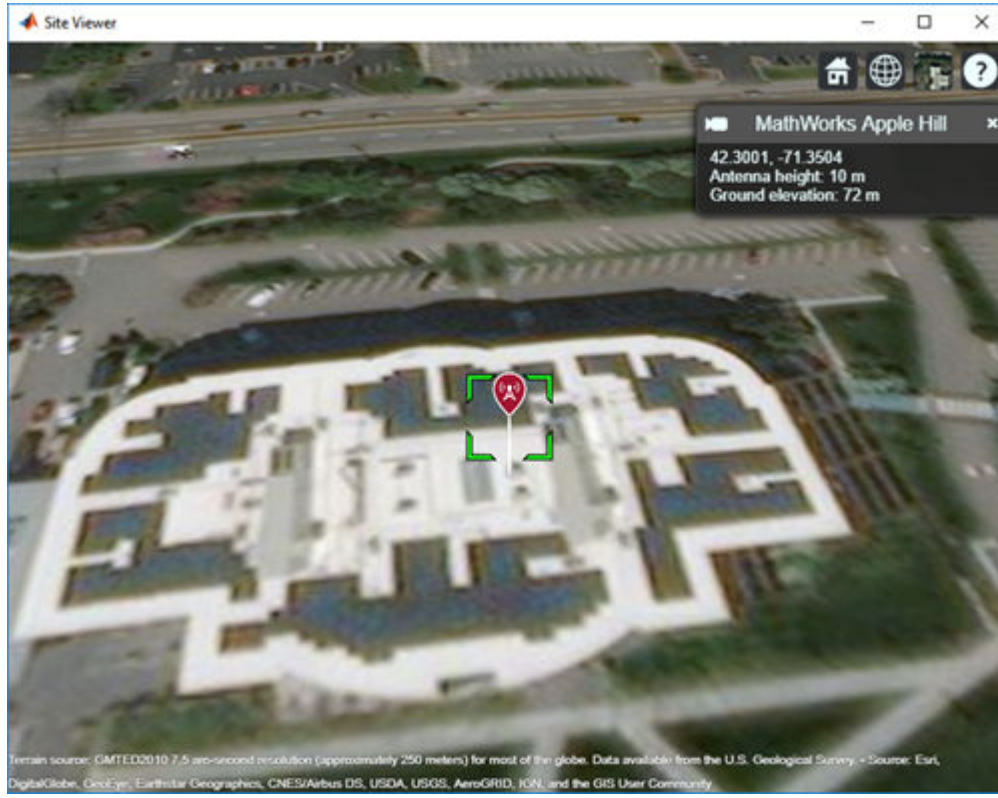
Show and Hide Transmitter Site

Create a transmitter site.

```
tx = txsite('Name','MathWorks Apple Hill',...
       'Latitude',42.3001, ...
       'Longitude',-71.3504);
```

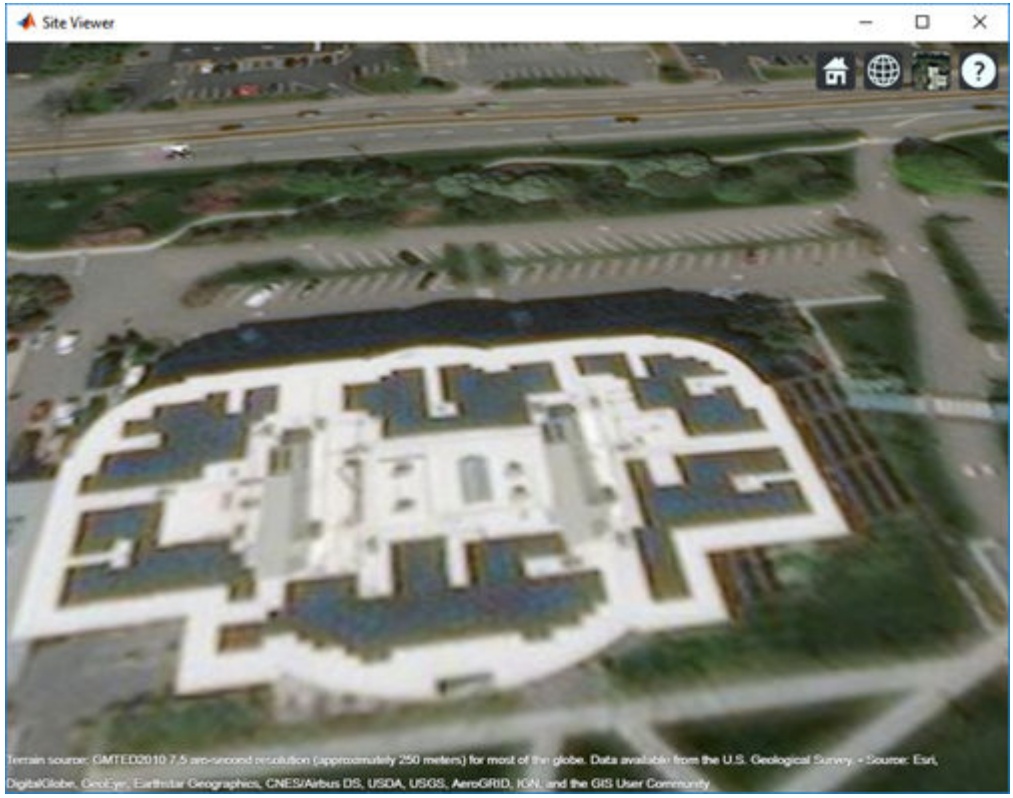
Show the transmitter site.

```
show(tx)
```



Hide the transmitter site.

```
hide(tx)
```



Input Arguments

`site` — Transmitter or receiver site
`txsite` or `rxsite` object | array of `txsite` or `rxsite` objects

Transmitter or receiver site, specified as a `txsite` or `rxsite` object or an array of `txsite` or `rxsite` objects.

See Also
`show`

Introduced in R2017b

link

Display communication link on map

Syntax

```
link(rx,tx)
link(rx,tx,propmodel)
link(___,Name,Value)
status = link(___)
```

Description

`link(rx,tx)` plots a one-way point-to-point communication link between a receiver site and transmitter site. The plot is color coded to identify the link success status.

`link(rx,tx,propmodel)` plots the communication link based on the specified propagation model.

`link(___,Name,Value)` plots a communication link using additional options specified by `Name,Value` pairs.

`status = link(___)` returns the success status of the communication link as `true` or `false`.

Examples

Communication Link Between Transmitter and Receiver

Create a transmitter site.

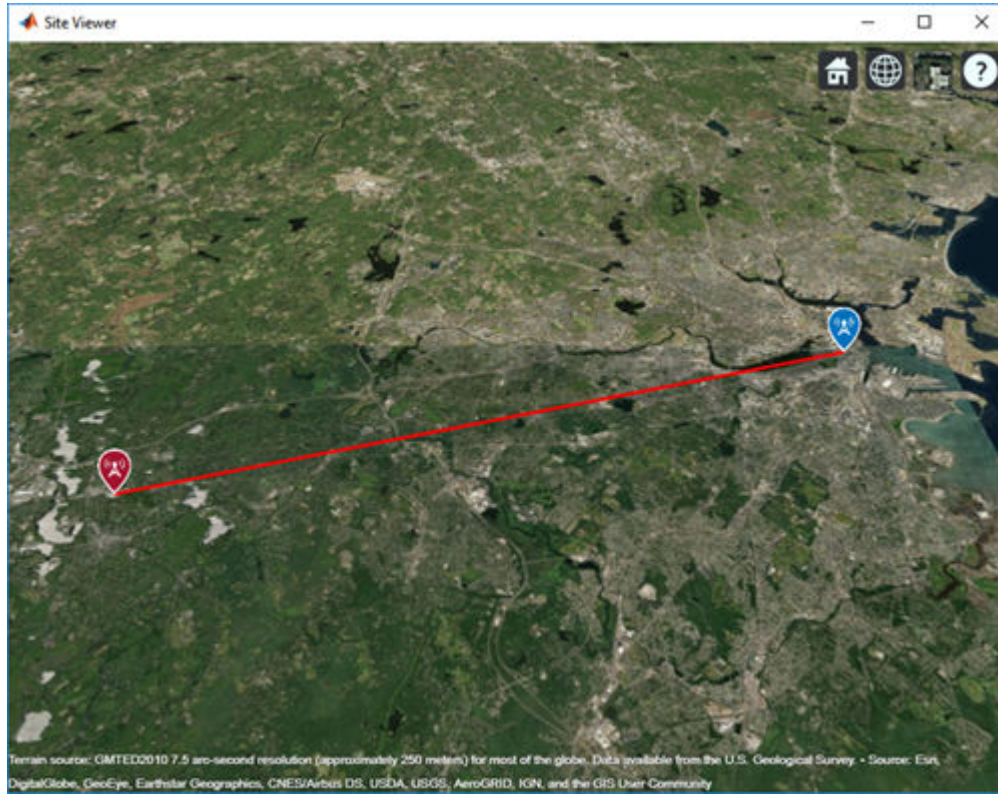
```
tx = txsite('Name','MathWorks', ...
        'Latitude', 42.3001, ...
        'Longitude', -71.3503);
```

Create a receiver site with sensitivity defined in dBm.

```
rx = rxsite('Name','Boston', ...
       'Latitude', 42.3601, ...
       'Longitude', -71.0589, ...
       'ReceiverSensitivity', -90);
```

Plot the communication link between the transmitter and the receiver.

```
link(rx,tx)
```



Input Arguments

`rx` — Receiver site
`rxsite` object | array of `rxsite` objects

Receiver site, specified as a `rxsite` object. You can use array inputs to specify multiple sites.

`tx` — Transmitter site
`txsite` object | array of `txsite` objects

Transmitter site, specified as a `txsite` object. You can use array inputs to specify multiple sites.

`propmodel` — Propagation model
character vector | string

Propagation model, specified as a character vector or string. You can also use the name-value pair `'PropagationModel'` to specify this parameter.

Data Types: `char` | `string`

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `'Type','power'`

`PropagationModel` — Propagation model to use for path loss calculations
`'freespace'` (default) | `'close-in'` | `'rain'` | `'gas'` | `'fog'` | propagation model object

Propagation model to use for path loss calculations, specified as `'freespace'`, `'close-in'`, `'rain'`, `'gas'`, `'fog'`, or as an object created using the `propagationModel` function.

Data Types: `char`

`SuccessColor` — Color of successful links
`'green'` (default) | RGB triplet | character vector

Color of successful links, specified as an RGB triplet or character vector. For more information, see `ColorSpec (Color Specification)`.

Data Types: `char` | `double`

`FailColor` — Color of unsuccessful links
`'red'` (default) | RGB triplet | character vector

Color of unsuccessful links, specified as RGB triplet or character vector. For more information, see `ColorSpec (Color Specification)`.

Data Types: `char` | `double`

Output Arguments

`status` — Success status of communication link
M-by-*N* array

Success status of communication links, returned as an *M*-by-*N* arrays. *M* is the number of transmitter sites and *N* is the number of receiver sites.

See Also

`coverage` | `los` | `propagationModel` | `sigstrength` | `sinr`

Topics

`ColorSpec (Color Specification)`

Introduced in R2017b

propagationModel

Create RF propagation model

Syntax

```
pm = propagationModel(modelname)
pm = propagationModel( ___ ,Name,Value)
```

Description

`pm = propagationModel(modelname)` creates an RF propagation model for the specified model.

`pm = propagationModel( ___ ,Name,Value)` sets properties using one or more name-value pairs. For example, `pm = propagationModel('rain','RainRate',96)` creates a rain propagation model with a rain rate of 96 mm/h. Enclose each property name in quotes.

Examples

Signal Strength of Receiver in Heavy Rain

Specify transmitter and receiver sites.

```
tx = txsite('Name','MathWorks Apple Hill',...
       'Latitude',42.3001, ...
       'Longitude',-71.3504, ...
       'TransmitterFrequency', 2.5e9);

rx = rxsite('Name','Fenway Park',...
       'Latitude',42.3467, ...
       'Longitude',-71.0972);
```

Create the propagation model for a heavy rainfall rate.

```
pm = propagationModel('rain','RainRate',50)
```

```
pm =
  Rain with properties:

    RainRate: 50
        Tilt: 0
```

Calculate the signal strength at the receiver using the rain propagation model.

```
ss = sigstrength(rx,tx,pm)
```

```
ss = -82.3103
```

Longley-Rice Propagation Model

Create a transmitter site.

```
tx = txsite
```

```
tx =
  txsite with properties:

                    Name: 'Site 3'
                Latitude: 42.3001
               Longitude: -71.3504
                 Antenna: [1×1 dipole]
            AntennaAngle: 0
           AntennaHeight: 10
              SystemLoss: 0
    TransmitterFrequency: 1.9000e+09
        TransmitterPower: 10
```

Create a Longley-Rice propagation model using the `propagationModel` function.

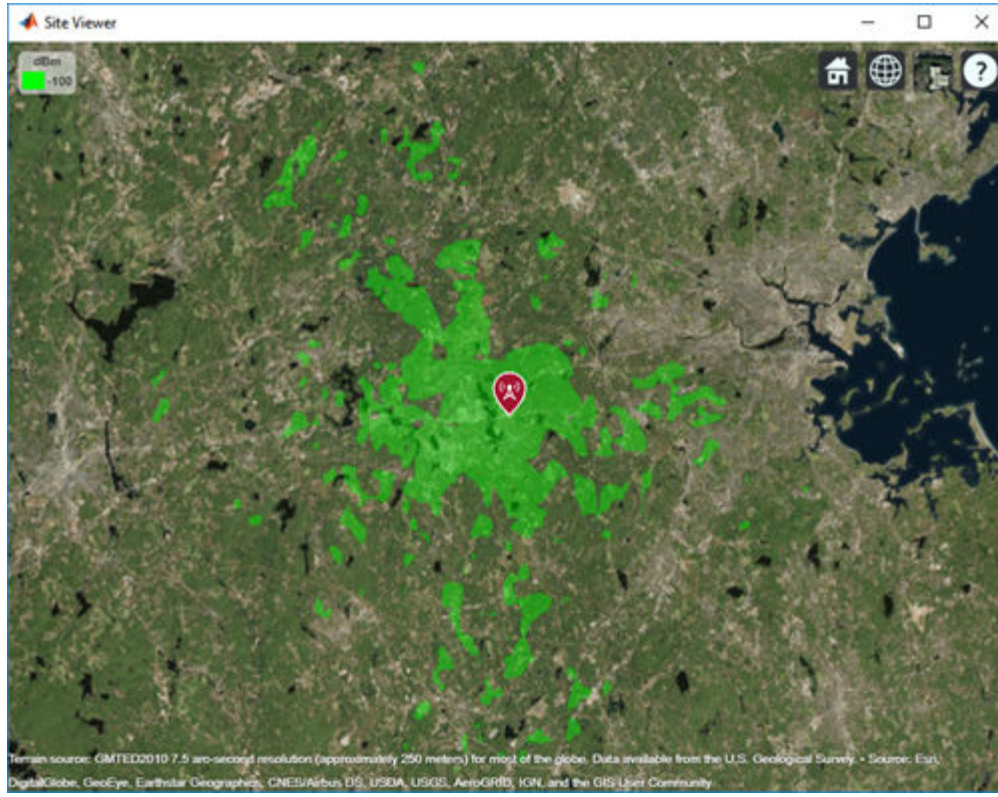
```
pm = propagationModel('longley-rice','TimeVariabilityTolerance',0.7)
```

```
pm =
  LongleyRice with properties:

              AntennaPolarization: 'horizontal'
```

```
                GroundConductivity: 0.0050
                GroundPermittivity: 15
            AtmosphericRefractivity: 301
                        ClimateZone: 'continental-temperate'
           TimeVariabilityTolerance: 0.7000
      SituationVariabilityTolerance: 0.5000
```

Find the coverage of the transmitter site using the defined propagation model.

```
coverage(tx,'PropagationModel',pm)
```



Input Arguments

`modelname` — Type of propagation model
`'freespace'|'rain'|'gas'|'fog'|'close-in'|'longley-rice'`

Type of propagation model:

- `'freespace'` – Free space propagation model
- `'rain'` – Rain propagation model. For more information, see [3].
- `'gas'` – Gas propagation model
- `'fog'` – Fog propagation model. For more information, see [2].
- `'close-in'` – Close-in propagation model. This model is valid for distances greater than or equal to the `ReferenceDistance` property. If the distance is less than `ReferenceDistance` used, path loss is 0. For more information, see [1].
- `'longley-rice'` – Longley-Rice propagation model. This model is also known as Irregular Terrain Model (ITM). For more information and list of limitations, see [4].

Note The Longley-Rice propagation model supported by Antenna Toolbox is point-to-point.

You can use the following functions on RF propagation models:

- `range` – Calculate the range of the radio wave under different propagation scenarios. `range` function does not support Longley-Rice propagation model.
- `pathloss` – Calculate the path loss of radio wave propagation between the transmitter and receiver sites under different propagation scenarios.

Data Types: `char`

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `'RainRate',50`

Rain

`RainRate` — Rain rate

`95` (default) | positive scalar

Rain rate, specified as a positive scalar in millimeters per hour (mm/h).

Dependencies

To specify `'RainRate'`, you must specify `'rain'` propagation model.

Data Types: `double`

`Tilt` — Polarization tilt angle of the signal

`0` (default) | scalar

Polarization tilt angle of the signal, specified as scalar in degrees.

Dependencies

To specify `'Tilt'`, you must specify `'rain'` propagation model.

Data Types: `double`

Gas

`Temperature` — Air temperature

`15` (default) | scalar

Air temperature, specified as a scalar in Celsius (C).

Dependencies

To specify `'Temperature'`, you must specify `'gas'` propagation model.

Data Types: `double`

`AirPressure` — Dry air pressure

`101300` (default) | scalar

Dry air pressure, specified as a scalar in pascals (Pa).

Dependencies

To specify `'AirPressure'`, you must specify `'gas'` propagation model.

Data Types: `double`

`WaterDensity` — Water vapor density

`7.5` (default) | scalar

Water vapor density, specified as a scalar in grams per cubic meter (g/m^3).

Dependencies

To specify `'WaterDensity'`, you must specify `'gas'` propagation model.

Data Types: `double`

Fog

`Temperature` — Air temperature

`15` (default) | scalar

Air temperature, specified as a scalar in Celsius (C).

Dependencies

To specify `'Temperature'`, you must specify `'fog'` propagation model.

Data Types: `double`

`WaterDensity` — Liquid water density

`0.5` (default) | scalar

Liquid water density, specified as a scalar in grams per cubic meter (g/m^3).

Dependencies

To specify `'WaterDensity'`, you must specify `'fog'` propagation model.

Data Types: `double`

Close-In

`ReferenceDistance` — Free-space reference distance

`1` (default) | scalar

Free-space reference distance, specified as a scalar in meters.

Dependencies

To specify `'ReferenceDistance'`, you must specify the `'close-in'` propagation model.

Data Types: `double`

`PathLossExponent` — Path loss exponent

`2.9` (default) | scalar

Path loss exponent, specified as a scalar.

Dependencies

To specify `'PathLossExponent'`, you must specify `'close-in'` propagation model.

Data Types: `double`

`Sigma` — Standard deviation

`5.7` (default) | scalar

Standard deviation of the zero-mean Gaussian random variable, specified as a scalar in decibels (dB).

Dependencies

To specify `'Sigma'`, you must specify `'close-in'` propagation model.

Data Types: `double`

`NumDataPoints` — Number of data points

`1869` (default) | integer

Number of data points of zero-mean Gaussian random variable, specified as an integer.

Dependencies

To specify `'NumDataPoints'`, you must specify `'close-in'` propagation model.

Data Types: `double`

Longley-Rice

`AntennaPolarization` — Polarization of transmitter and receiver antennas

`'horizontal'` (default) | `'vertical'`

Polarization of transmitter and receiver antennas, specified as `'horizontal'` or `'vertical'`. Both antennas are assumed to have the same polarization. This value is used to calculate path loss due to ground reflection.

Dependencies

To specify `'AntennaPolarization'`, you must specify `'longley-rice'` propagation model.

Data Types: `char` | `string`

`GroundConductivity` — Conductivity of ground

`0.005` (default) | scalar

Conductivity of the ground, specified as a scalar in siemens per meter (S/m). This value is used to calculate path loss due to ground reflection. The default value corresponds to average ground.

Dependencies

To specify `'GroundConductivity'`, you must specify `'longley-rice'` propagation model.

Data Types: `double`

`GroundPermittivity` — Relative permittivity of ground

`15` (default) | scalar

Relative permittivity of the ground, specified as a scalar. Relative permittivity is expressed as a ratio of absolute material permittivity to the permittivity of vacuum. This value is used to calculate the path loss due to ground reflection. The default value corresponds to average ground.

Dependencies

To specify `'GroundPermittivity'`, you must specify `'longley-rice'` propagation model.

Data Types: `double`

`AtmosphericRefractivity` — Atmospheric refractivity near ground

`301` (default) | scalar

Atmospheric refractivity near the ground, specified as a scalar in N-units. This value is used to calculate the path loss due to refraction through the atmosphere and tropospheric scatter. The default value corresponds to average atmospheric conditions.

Dependencies

To specify `'AtmosphericRefractivity'`, you must specify `'longley-rice'` propagation model.

Data Types: `double`

`ClimateZone` — Radio climate zone

`'continental-temperate'` (default) | `'equatorial'` | `'continental-subtropical'` | `'maritime-subtropical'` | `'desert'` | `maritime-over-land'` | `'maritime-over-sea'`

Radio climate zone. This value is used to calculate the variability due to changing atmospheric conditions. The default value corresponds to average atmospheric conditions in a particular climate zone.

Dependencies

To specify `'ClimateZone'`, you must specify `'longley-rice'` propagation model.

Data Types: `char` | `string`

`TimeVariabilityTolerance` — Time variability tolerance level

`0.5` (default) | scalar

Time variability tolerance level of the path loss, specified as a scalar between [0.001, 0.999]. Time variability occurs due to changing atmospheric conditions. This value gives the required system reliability or the fraction of time during which the actual path loss is expected to be less than or equal to model prediction. For more information, see [5].

Dependencies

To specify `'TimeVariabilityTolerance'`, you must specify `'longley-rice'` propagation model.

Data Types: `double`

`SituationVariabilityTolerance` — Situation variability tolerance level

`0.5` (default) | scalar

Situation variability tolerance level of the path loss, specified as a scalar in between [0.001, 0.999]. Situation variability occurs due to uncontrolled or hidden random variables. This value gives the required system confidence or the fraction of similar situations for which the actual path loss is expected to be less than or equal to the model prediction. For more information, see [5].

Dependencies

To specify `'SituationVariabilityTolerance'`, you must specify `'longley-rice'` propagation model.

Data Types: `double`

References

[1] Sun, S.,Rapport, T.S., Thomas, T., Ghosh, A., Nguyen, H., Kovacs, I., Rodriguez, I., Koymen, O.,and Prartyka, A. "Investigation of prediction accuracy, sensitivity, and parameter stability of large-scale propagation path loss models for 5G wireless communications." *IEEE Transactions on Vehicular Technology*, Vol.65, No 5, pp 2843-2860, May 2016.

[2] ITU-R P.840-6. "Attenuation due to cloud and fog." *Radiocommunication Sector of ITU*

[3] ITU-R P.838-3. "Specific attenuation model for rain for use in prediction methods." *Radiocommunication Sector of ITU*

[4] Huffor, George A., Anita G. Longley, and Wlliam A.Kissick. "A Guide to the Use of the ITS Irregular Terrain Model in the Area Prediction Mode." *NTIA Report 82-100*. Pg-7.

[5] *SoftWright Homepage* https://www.softwright.com/faq/support/longley_rice_variability.html

[6] Seybold, John. *Introduction to RF Propagation*. Wiley, 2005

[7] ITU-R P.676-11. "Attenuation by atmospheric gases." *Radiocommunication Sector of ITU*

See Also

`coverage` | `link` | `pathloss` | `range` | `sigstrength` | `sinr`

Introduced in R2017b

sigstrength

Signal strength due to transmitter

Syntax

```
ss = sigstrength(rx,tx)
ss = sigstrength(rx,tx,propmodel)
ss = sigstrength( ___ ,Name,Value)
```

Description

`ss = sigstrength(rx,tx)` returns the signal strength at the receiver site due to the transmitter site.

`ss = sigstrength(rx,tx,propmodel)` returns the signal strength at the receiver site using the specified propagation model. Specifying propagation model is same as specifying the `'PropagationModel'` name-value pair.

`ss = sigstrength( ___ ,Name,Value)` returns the signal strength using additional options specified by `Name,Value` pairs and either of the previous syntaxes.

Examples

Received Power and Link Margin at Receiver

Create a transmitter site.

```
tx = txsite('Name','MathWorks', ...
        'Latitude', 42.3001, ...
        'Longitude', -71.3503);
```

Create a receiver site with sensitivity defined (in dBm).

```
rx = rxsite('Name','Boston', ...
        'Latitude', 42.3601, ...
```

```
        'Longitude', -71.0589, ...
        'ReceiverSensitivity', -90);
```

Calculate the received power and link margin. Link margin is the difference between the receiver's sensitivity and the received power.

```
ss = sigstrength(rx,tx)
```

```
ss = -124.0581
```

```
margin = abs(rx.ReceiverSensitivity - ss)
```

```
margin = 34.0581
```

Input Arguments

`rx` — Receiver site

`rxsite` object | array of `rxsite` objects

Receiver site, specified as a `rxsite` object. You can use array inputs to specify multiple sites.

`tx` — Transmitter site

`txsite` object | array of `txsite` objects

Transmitter site, specified as a `txsite` object. You can use array inputs to specify multiple sites.

`propmodel` — Propagation model

character vector | string

Propagation model, specified as a character vector or string. You can also use the name-value pair `'PropagationModel'` to specify this parameter. You can also use the `propagationModel` function to define this input.

Data Types: `char` | `string`

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes.

You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `'Type','power'`

Type — Type of signal strength to compute
`'power'` (default) | `'efield'`

Type of signal strength to compute, specified as `'power'` or `'efield'`.

Power is expressed in power units (dBm).of signal at receiver input. E-Field is expressed in electric field strength units (dBuV/m) of signal wave incident on antenna.

Data Types: `char` | `string`

`PropagationModel` — Propagation model to use for path loss calculations
`'freespace'` (default) | `'close-in'` | `'rain'` | `'gas'` | `'fog'` | propagation model object

Propagation model to use for path loss calculations, specified as `'freespace'`, `'close-in'`, `'rain'`, `'gas'`, `'fog'`, or as an object created using the `propagationModel` function.

Data Types: `char` | `string`

Output Arguments

`ss` — Signal strength
M-by-*N* array

Signal strength, returned as *M*-by-*N* array in dBm. *M* is the number of TX sites and *N* is the number of RX sites.

See Also

`link` | `propagationModel` | `sinr`

Introduced in R2017b

add

Boolean unite operation on two shapes

Syntax

```
c = add(shape1,shape2)
```

Description

`c = add(shape1,shape2)` unites `shape1` and `shape2` using the add operation. You can also use the + to add the two shapes together.

Examples

Add Two Circles

Create and view a default circle.

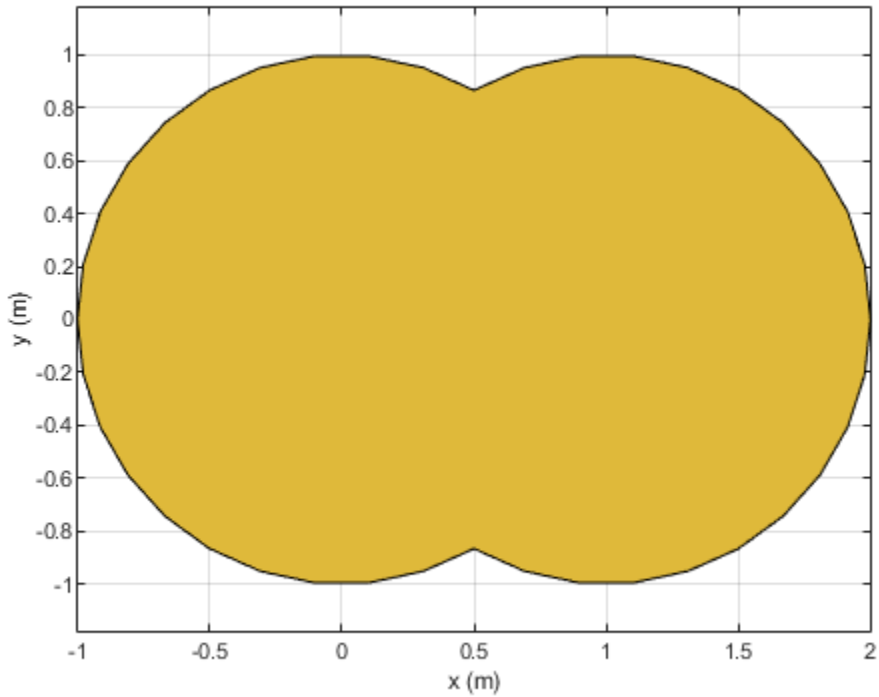
```
circle1 = antenna.Circle;
```

Create a circle with a radius of 1 m. The center of the circle is at [1 0].

```
circle2 = antenna.Circle('Center',[1 0],'Radius',1);
```

Add the two circles.

```
add(circle1,circle2)
```



Add Two Shapes

Create circle with a radius of 1 m. The center of the circle is at [1 0].

```
circle1 = antenna.Circle('Center',[1 0],'Radius',1);
```

Create a rectangle with a length of 2 m and a width of 4 m centered at the origin.

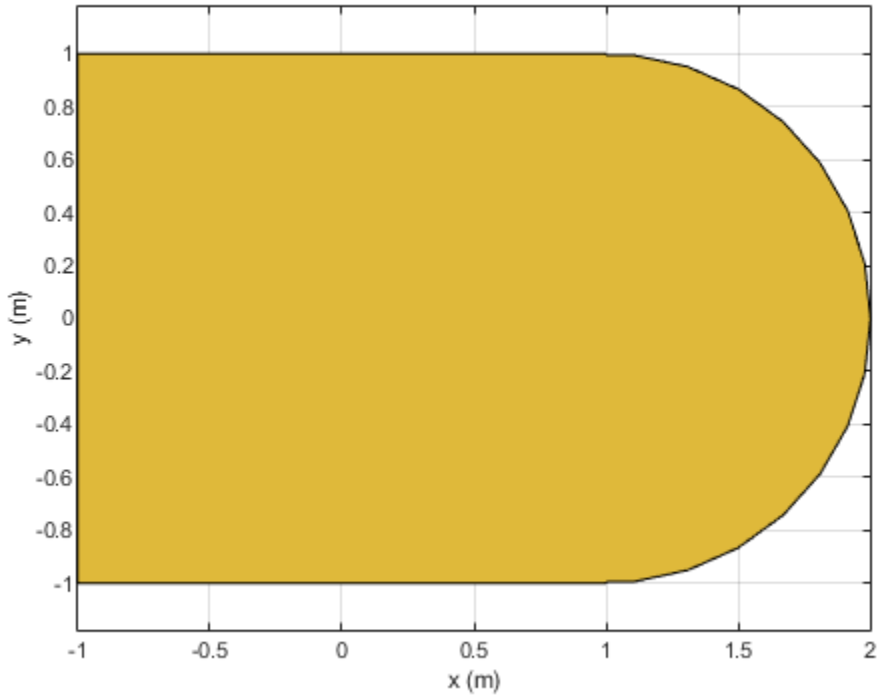
```
rect1 = antenna.Rectangle('Length',2,'Width',2);
```

Add the two shapes together using the + function.

```
polygon1 = circle1+rect1
```

```
polygon1 =
  Polygon with properties:

        Name: 'mypolygon'
    Vertices: [21x3 double]
```

```
show(polygon1)
```



Input Arguments

shape1,shape2 — Shapes created using custom elements and shape objects
object handle

Shapes created using custom elements and shape objects of Antenna Toolbox, specified as an object handle.

Example: `c = add(rectangle1, rectangle2)` where rectangle1 and rectangle2 are shapes created using `antenna.Rectangle` object.

See Also

`area` | `intersect` | `mesh` | `plot` | `rotate` | `rotateX` | `rotateY` | `rotateZ` | `scale` | `show` | `subtract` | `translate`

Introduced in R2017a

area

Calculate area of shape in sq.m

Syntax

```
a = area(shape)
```

Description

`a = area(shape)` calculate area of the shape in units sq.m.

Examples

Create Notched Rectangle

Create a rectangle with a length of 0.15 m, and a width of 0.15 m.

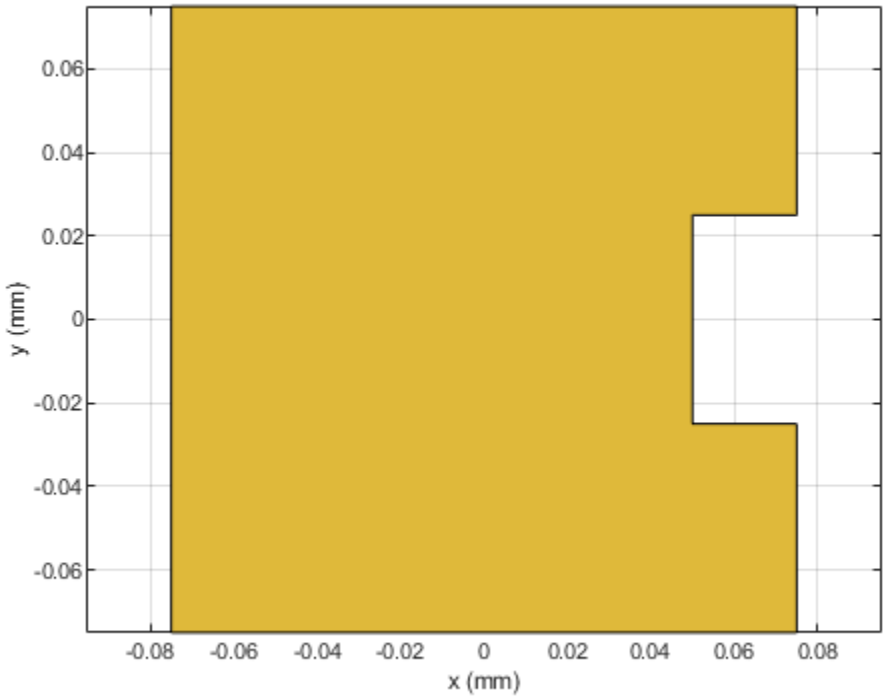
```
r  = antenna.Rectangle('Length',0.15,'Width',0.15);
```

Create a second rectangle with a length of 0.05 m, and a width of 0.05 m. Set the center of the second rectangle at half the length of the first rectangle r.

```
n = antenna.Rectangle('Center',[0.075,0],'Length',0.05,'Width',0.05);
```

Create and view a notched rectangle by subtracting n from r.

```
rn  = r-n;
show(rn)
```



Calculate the area of the notched rectangle.

```
area(rn)
```

```
ans = 0.0212
```

Input Arguments

shape — Shape created using custom elements and shape objects
object handle

Shape created using custom elements and shape objects of Antenna Toolbox, specified as an object handle.

Example: `c = area(rectangle)` where rectangle is created using `antenna.Rectangle` object.

See Also

`add` | `intersect` | `mesh` | `plot` | `rotate` | `rotateX` | `rotateY` | `rotateZ` | `scale` | `show` | `subtract` | `translate`

Introduced in R2017a

intersect

Boolean intersection operation on two shapes

Syntax

```
c = intersect(shape1,shape2)
```

Description

`c = intersect(shape1,shape2)` intersect `shape1` and `shape2` using the intersect operation. You can also use the & to intersect the two shapes.

Examples

Intersect Rectangle and Circle

Create a default rectangle.

```
r = antenna.Rectangle;
```

Create a default circle.

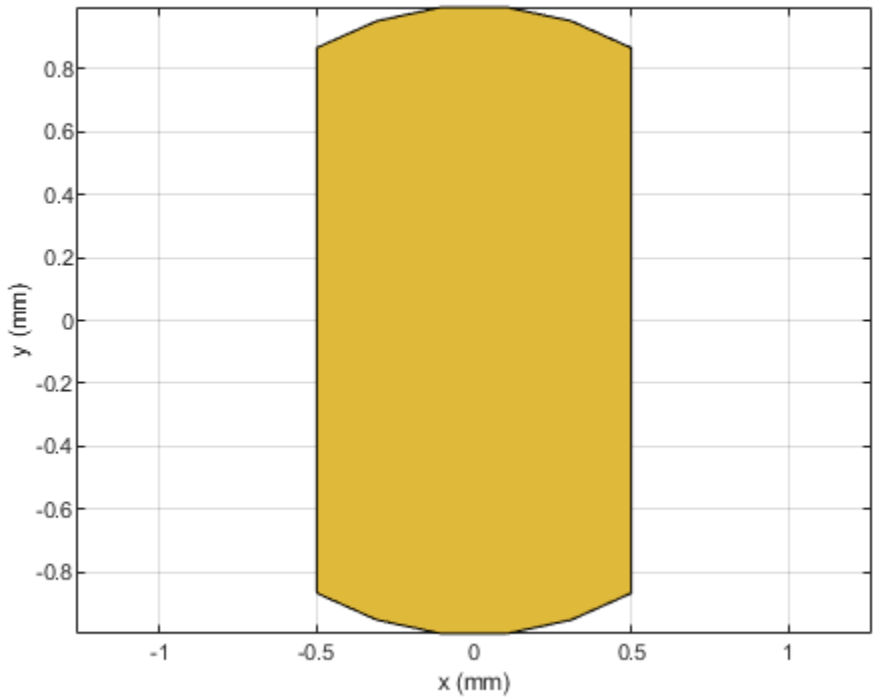
```
c = antenna.Circle;
```

Use `intersect` to combine the shared surfaces of the rectangle and the circle.

```
rc = intersect(r,c)
```

```
rc =
  Polygon with properties:

        Name: 'mypolygon'
    Vertices: [16x3 double]
```

```
show(rc)
axis equal
```



Input Arguments

shape1, shape2 — Shapes created using custom elements and shape objects
object handle

Shapes created using custom elements and shape objects of Antenna Toolbox, specified as an object handle.

Example: `c = intersect(rectangle1, rectangle2)` where rectangle1 and rectangle2 are shapes created using `antenna.Rectangle` object.

See Also

`add` | `area` | `mesh` | `plot` | `rotate` | `rotateX` | `rotateY` | `rotateZ` | `show` | `subtract` | `translate`

Introduced in R2017a

rotate

Rotate shape about axis and angle

Syntax

```
rotate(shape,angle,axis1,axis2)
c = rotate(shape,angle,axis1,axis2)
```

Description

`rotate(shape,angle,axis1,axis2)` rotate shape about an axes object and angle.

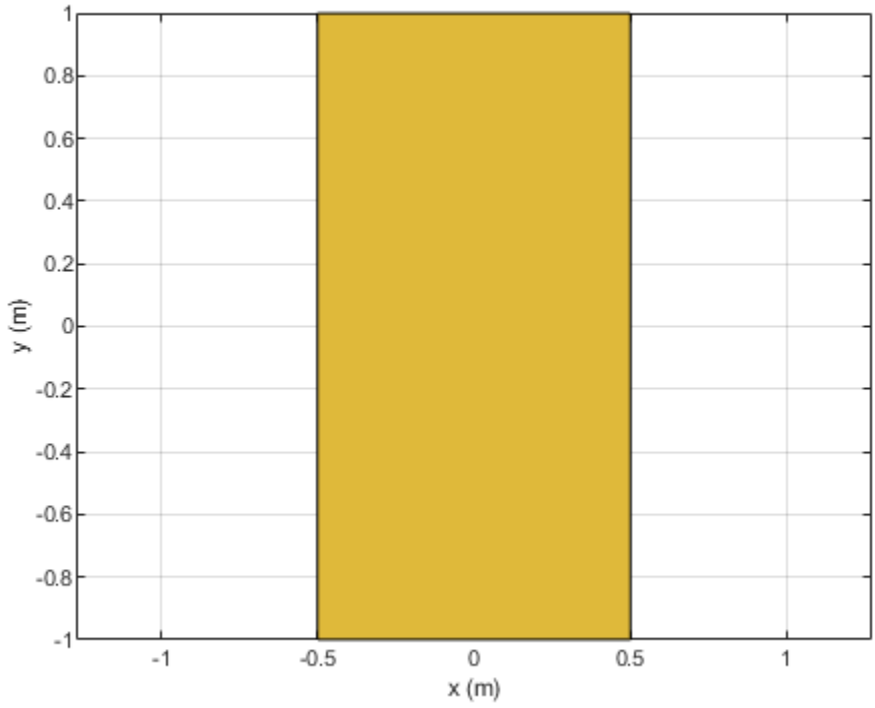
`c = rotate(shape,angle,axis1,axis2)` rotate shape about an axes object and angle.

Examples

Rotate Rectangle

Create a rectangle shape.

```
r = antenna.Rectangle;
show(r)
axis equal
```

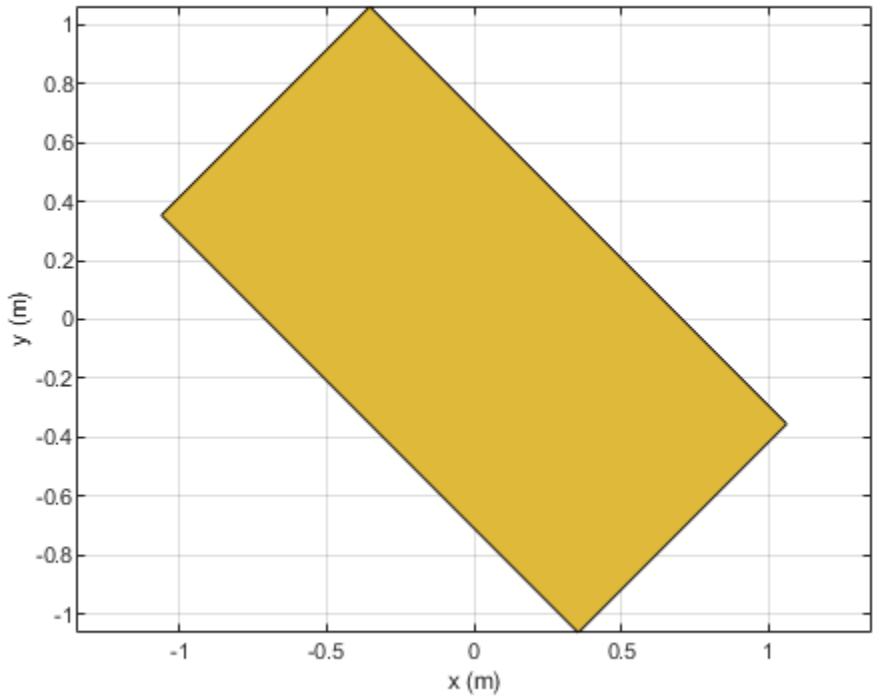


Rotate the rectangle at 45 degrees about the Z-axis.

```
r1 = rotate(r,45,[0 0 0],[0 0 1])
```

```
r1 =
  Rectangle with properties:

         Name: 'myrectangle'
       Center: [0 0]
       Length: 1
        Width: 2
    NumPoints: 2
```

```
show(r1)
```



Input Arguments

shape — Shape created using custom elements and shape objects

object handle

Shape created using custom elements and shape objects of Antenna Toolbox, specified as an object handle.

Example: `area(rectangle)` where rectangle is created using `antenna.Rectangle` object.

`axis1,axis2` — Axis of rotation
two three-element vector of Cartesian coordinates in meters

Axis of rotation,specified as two unique three-element vectors of Cartesian coordinates in meters

Example: `rotate(rectangle,45,[0 0 0], [0 0 1])` where rectangle is created using `antenna.Rectangle` object.

Data Types: `double`

`angle` — Angle of rotation
scalar

Angle of rotation, specified as a scalar in degrees

Example: `rotate(rectangle,45,[0 0 1], [0 0 0])` rotates the rectangle around X-axis by 45 degrees.

Data Types: `double`

See Also

`add` | `area` | `intersect` | `mesh` | `plot` | `rotateX` | `rotateY` | `rotateZ` | `scale` | `show` | `subtract` | `translate`

Introduced in R2017a

subtract

Boolean subtraction operation on two shapes

Syntax

```
c = subtract(shape1,shape2)
```

Description

`c = subtract(shape1,shape2)` subtracts `shape1` and `shape2` using the subtract operation. You can also use the `-` to subtract the two shapes.

Examples

Create Notched Rectangle

Create a rectangle with a length of 0.15 m, and a width of 0.15 m.

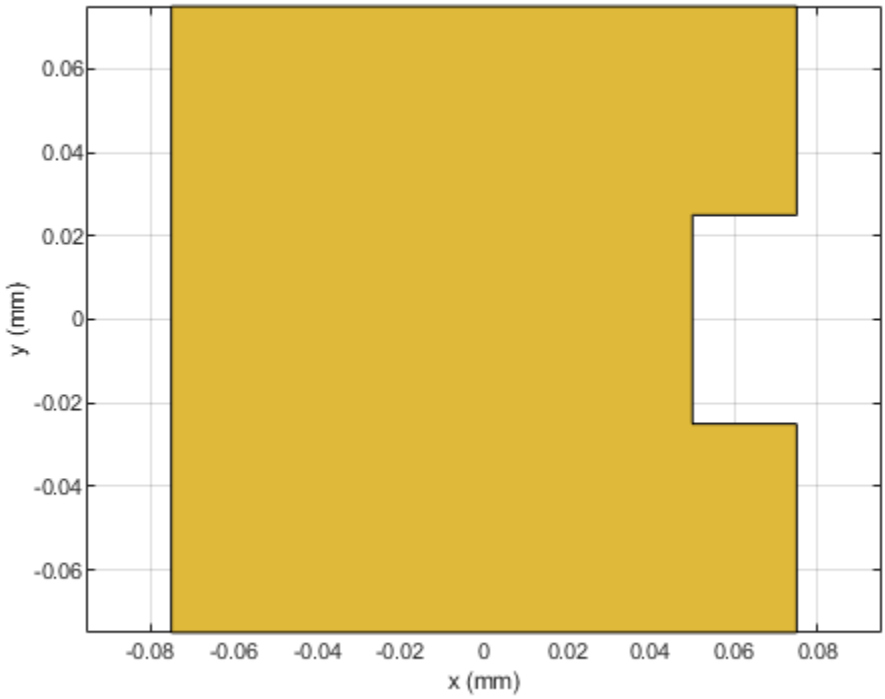
```
r  = antenna.Rectangle('Length',0.15,'Width',0.15);
```

Create a second rectangle with a length of 0.05 m, and a width of 0.05 m. Set the center of the second rectangle at half the length of the first rectangle r.

```
n = antenna.Rectangle('Center',[0.075,0],'Length',0.05,'Width',0.05);
```

Create and view a notched rectangle by subtracting n from r.

```
rn  = r-n;
show(rn)
```



Calculate the area of the notched rectangle.

```
area(rn)
```

```
ans = 0.0212
```

Input Arguments

shape1, shape2 — Shapes created using custom elements and shape objects
object handle

Shapes created using custom elements and shape objects of Antenna Toolbox, specified as an object handle.

Example: `c = subtract(rectangle1, rectangle2)` where rectangle1 and rectangle2 are shapes created using `antenna.Rectangle` object.

See Also

`add` | `area` | `intersect` | `mesh` | `plot` | `rotate` | `rotateX` | `rotateY` | `rotateZ` | `scale` | `show` | `translate`

Introduced in R2017a

gerberWrite

Generate Gerber files

Syntax

```
gerberWrite(designobject)
gerberWrite(designobject,rfconnector)
gerberWrite(designobject,writer)
gerberWrite(designobject,writer,rfconnector)
[a,g] = gerberWrite(designobject,writer,rfconnector)
```

Description

`gerberWrite(designobject)` creates a Gerber file from PCB specification files, such as `PCBWriter` object or `pcbStack` object.

`gerberWrite(designobject,rfconnector)` creates Gerber file using specified RF connector.

`gerberWrite(designobject,writer)` creates a Gerber file using specified PCB writer services.

`gerberWrite(designobject,writer,rfconnector)` creates a Gerber file using specified PCB writer and connector services.

`[a,g] = gerberWrite(designobject,writer,rfconnector)` creates a Gerber file using specified PCB writer and connector services.

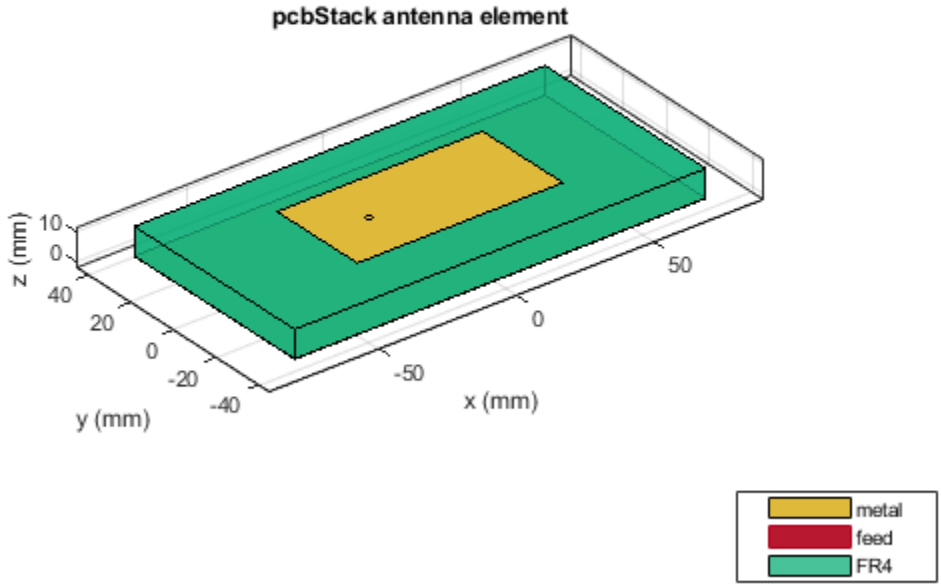
Note You can only use output arguments if the `designobject` is a `pcbStack` object.

Examples

Antenna Gerber Files from PCB Stack

Create a patch antenna with FR4 as dielectric material using |pcbStack| object.

```
p = pcbStack;
d = dielectric('FR4');
p.Layers = {p.Layers{1},d,p.Layers{2}};
p.FeedLocations(3:4) = [1 3];
show(p)
```



Use a Cinch SMA for feeding the antenna. Use the Mayhew Labs PCB viewer as the 3-D viewer. Change the file name of the Mayhew Writer services to `antenna_design_file`.

```
C = PCBConnectors.SMA_Cinch;
W = PCBServices.MayhewWriter;
W.Filename = 'antenna_design_file';
```

Generate the Gerber-format files.

```
[A,g] = gerberWrite(p,W,C)
```

```
A =
  PCBWriter with properties:

                      Design: [1x1 struct]
                      Writer: [1x1 PCBServices.MayhewWriter]
                   Connector: [1x1 PCBConnectors.SMA_Cinch]
           UseDefaultConnector: 0
    ComponentBoundaryLineWidth: 8
         ComponentNameFontSize: []
            DesignInfoFontSize: []
                        Font: 'Arial'
                   PCBMargin: 5.0000e-04
                  Soldermask: 'both'
                 Solderpaste: 1

   See info for details

g =
'C:\TEMP\Bdoc18b_943130_7372\ib632619\29\tp177aec5b\antenna-ex96485213\antenna_design_
```

Show Antenna PCB Design Using Mayhew Manufacturing Service

Create a coplanar inverted F antenna.

```
fco = invertedFcoplanar('Height',14e-3,'GroundPlaneLength', 100e-3,  ...
                  'GroundPlaneWidth', 100e-3);
```

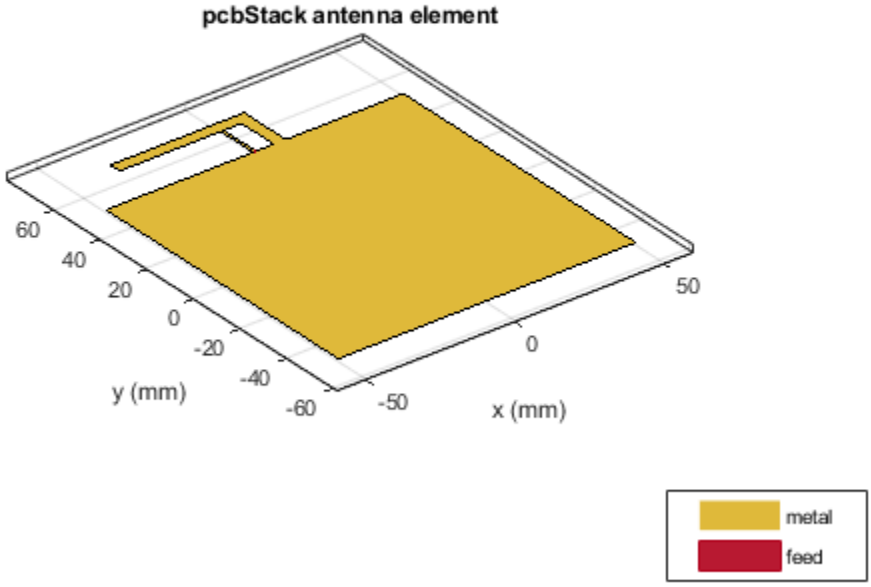
Use this antenna in creating a `pcbStack` object.

```
p = pcbStack(fco)
```

```
p =
  pcbStack with properties:

              Name: 'Coplanar Inverted-F'
```

```
          Revision: 'v1.0'
         BoardShape: [1×1 antenna.Rectangle]
    BoardThickness: 0.0013
            Layers: {[1×1 antenna.Polygon]}
     FeedLocations: [0 0.0500 1]
      FeedDiameter: 5.0000e-04
      ViaLocations: []
       ViaDiameter: []
      FeedViaModel: 'strip'
       FeedVoltage: 1
         FeedPhase: 0
              Tilt: 0
           TiltAxis: [1 0 0]
              Load: [1×1 lumpedElement]
```

```
figure
show(p)
```



Use an SMA_Cinch as an RF connector and Mayhew Writer as a 3-D viewer.

```
c = PCBConnectors.SMA_Cinch
```

```
c =
  SMA_Cinch with properties:

                  Type: 'SMA'
                   Mfg: 'Cinch'
                  Part: '142-0711-202'
            Annotation: 'SMA'
             Impedance: 50
             Datasheet: 'https://belfuse.com/resources/Johnson/drawings/dr-142-0711-2
              Purchase: 'https://www.digikey.com/product-detail/en/cinch-connectivity
```

```
               TotalSize: [0.0071 0.0071]
           GroundPadSize: [0.0024 0.0024]
       SignalPadDiameter: 0.0017
         PinHoleDiameter: 0.0013
           IsolationRing: 0.0041
    VerticalGroundStrips: 1

   Cinch 142-0711-202 (Example Purchase)
```

```
s = PCBServices.MayhewWriter
```

```
s =
  MayhewWriter with properties:

              BoardProfileFile: 'legend'
         BoardProfileLineWidth: 1
                CoordPrecision: [2 6]
                    CoordUnits: 'in'
               CreateArchiveFile: 0
                  DefaultViaDiam: 3.0000e-04
              DrawArcsUsingLines: 1
                  ExtensionLevel: 1
                        Filename: 'untitled'
                           Files: {}
        IncludeRootFolderInZip: 0
                   PostWriteFcn: @(obj)sendTo(obj)
    SameExtensionForGerberFiles: 0
                   UseExcellon: 1
```

Create an antenna design file using `PCBWriter` .

```
PW = PCBWriter(p,s,c)
```

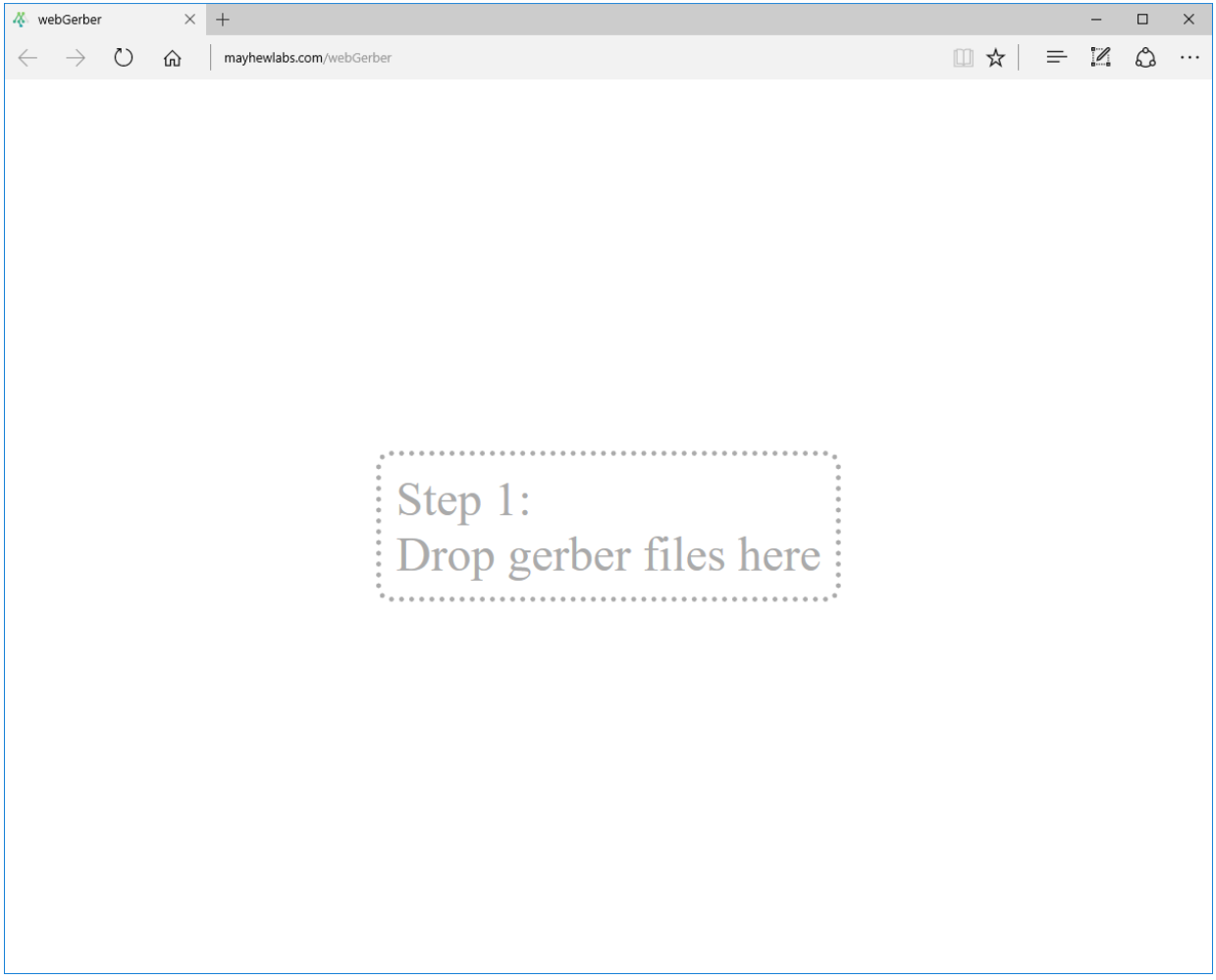
```
PW =
  PCBWriter with properties:

                       Design: [1×1 struct]
                       Writer: [1×1 PCBServices.MayhewWriter]
                    Connector: [1×1 PCBConnectors.SMA_Cinch]
          UseDefaultConnector: 0
    ComponentBoundaryLineWidth: 8
         ComponentNameFontSize: []
            DesignInfoFontSize: []
```

```
                    Font: 'Arial'
                 PCBMargin: 5.0000e-04
                Soldermask: 'both'
               Solderpaste: 1

   See info for details
```

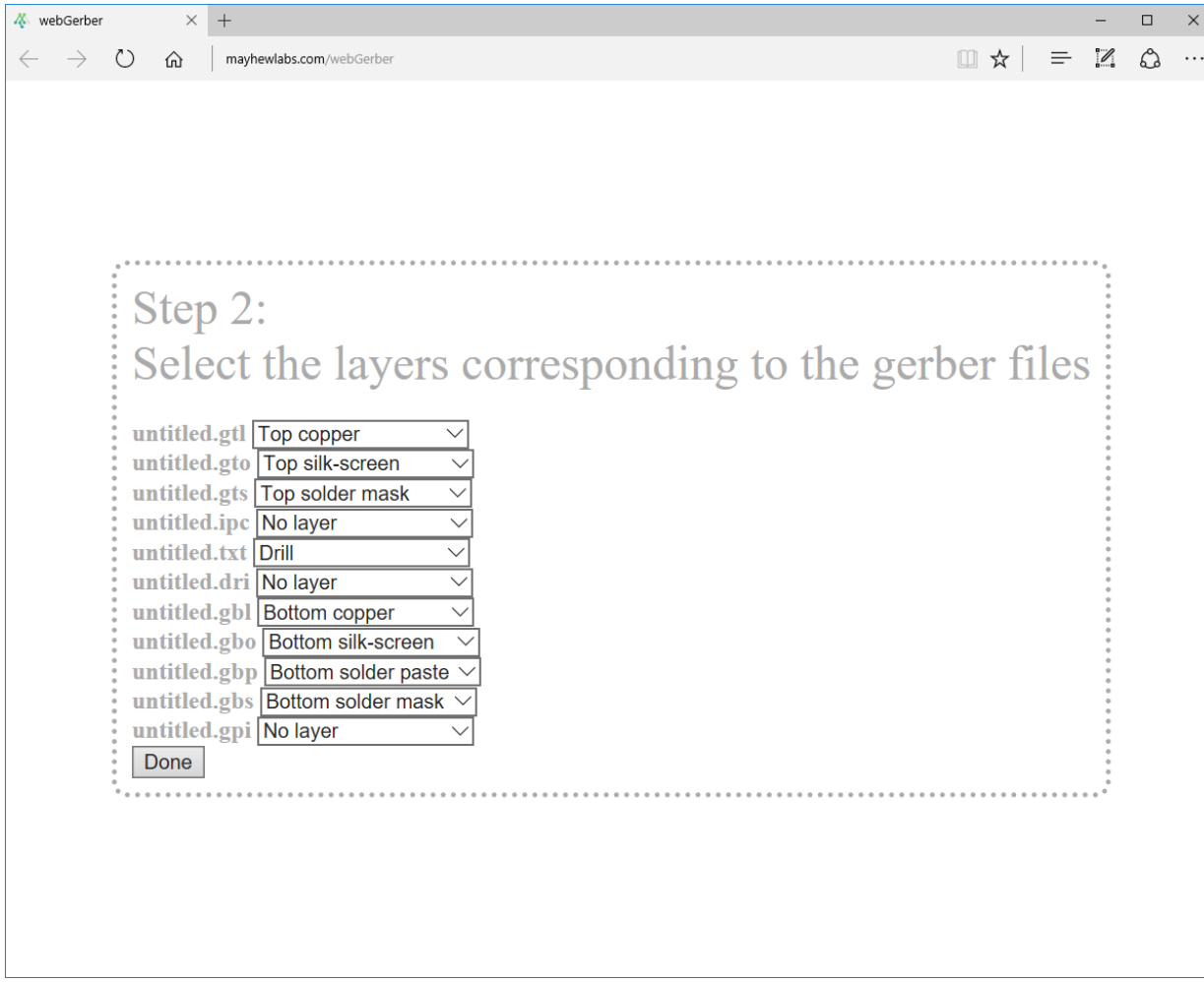
Use the gerberWrite method to create gerber files from the antenna design files. The files generated are then send to the Mayhew writer manufacturing service.

```
gerberWrite(PW)
```

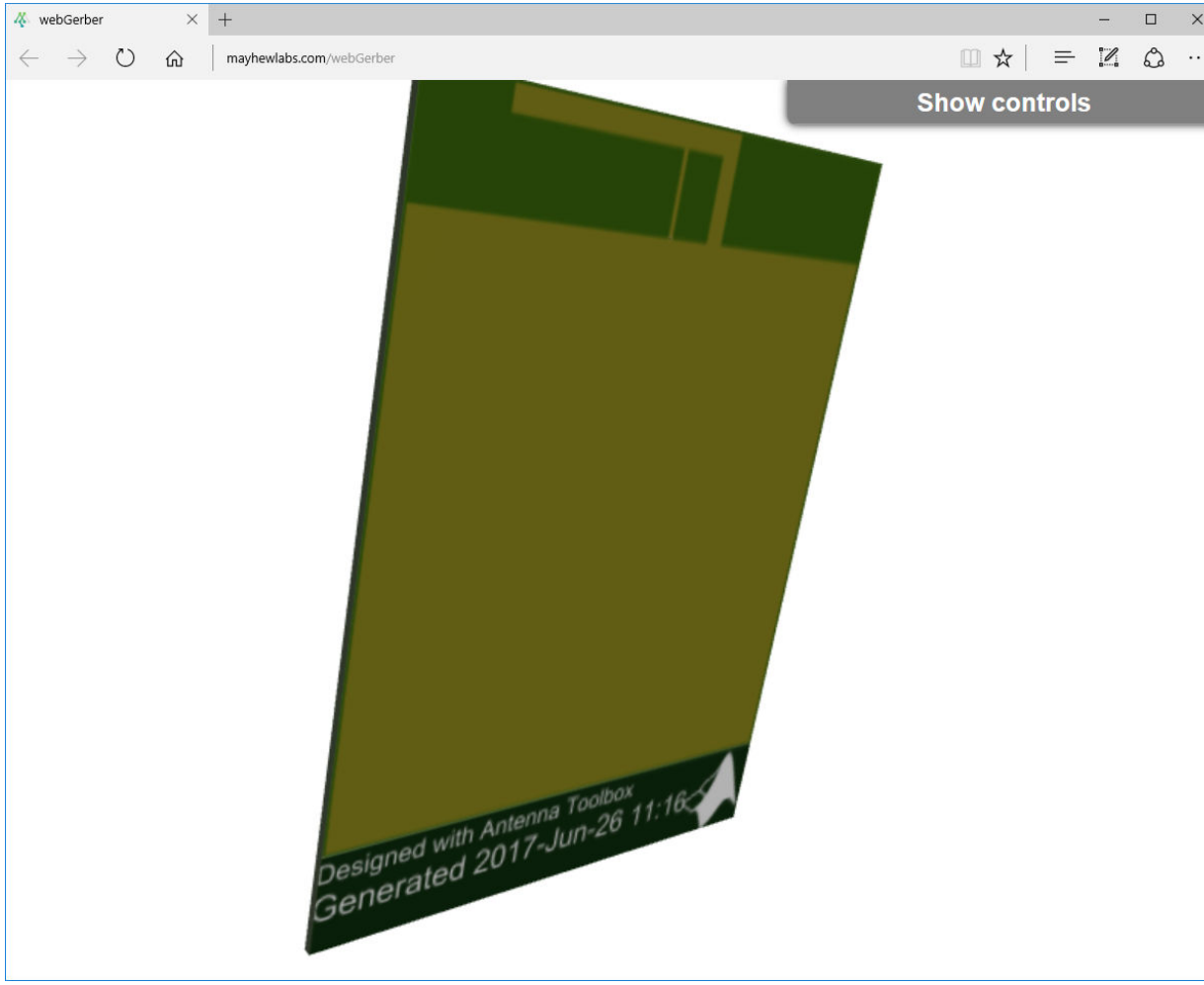
By default, the folder containing the gerber files is called "untitled" and is located in your MATLAB folder. Running this example automatically opens up the Mayhew Labs PCB manufacturing service in your internet browser.



Drag and drop all your files from the "untitled" folder.



Click **Done** to view your Antenna PCB.



Input Arguments

`designobject` — Antenna design geometry file

`pcbStack` object | `PCBWriter` object

Antenna design geometry file, specified as a `pcbStack` object or `PCBWriter` object.

Example: `p1 = pcbStack` creates a PCB stack object.`p1 gerberWrite(p1)` creates a Gerber file using `p1`.

Example: `p1 = pcbStack` creates a PCB stack object.`p1 a = PCBWriter(p1)`, creates a `PCBWriter` object, `a`. `gerberWrite(a)`, creates a Gerber file using `a`.

`rfconnector` — RF connector type
`PCBConnector` object

RF connector type, specified as a `PCBConnector` object.

Example: `c = PCBConnectors.SMA_Cinch;gerberWrite(p1,c)` uses SMA_Cinch RF connector at the feedpoint.

`writer` — PCB service
`PCBServices` object

PCB service, specified as a `PCBServices` object.

Example: `s =PCBServices.MayhewWriter;gerberWrite(p1,s)` uses Mayhew Labs PCB service to create and view the PCB design.

Output Arguments

Note You can only use output arguments if the `designobject` is a `pcbStack` object.

`a` — PCBWriter object
object handle

`PCBWriter` object that generated the Gerber files, returned as an object handle.

`g` — Path to generated Gerber files folder
character vector

Path to generated Gerber files folder, returned as character vector.

See Also
`PCBConnectors` | `PCBServices`

Introduced in R2017b

show

Show site location on map

Syntax

```
show(site)
show(site,Name,Value)
```

Description

`show(site)` displays the location of transmitter or receiver site on a map using a marker.

`show(site,Name,Value)` uses `icon` displays the site map using additional options specified by the `Name,Value` pairs.

Examples

Default Receiver Site

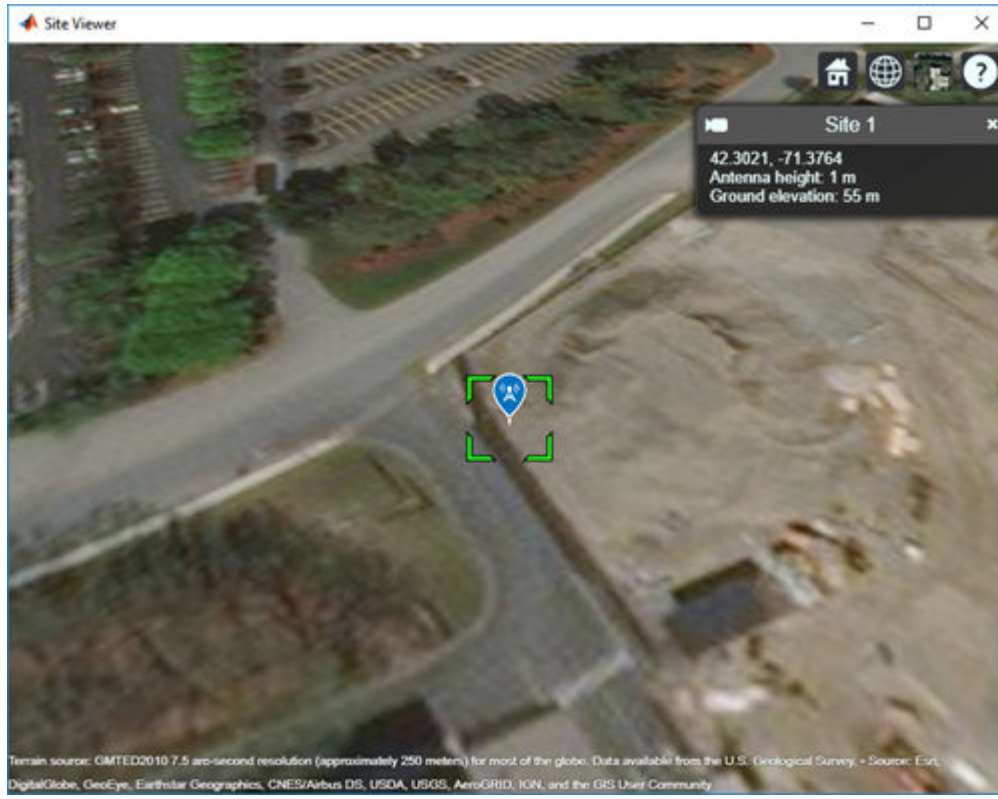
Create and show the default receiver site.

```
rx = rxsite
```

```
rx =
  rxsite with properties:

                  Name: 'Site 2'
              Latitude: 42.3021
             Longitude: -71.3764
               Antenna: [1×1 dipole]
          AntennaAngle: 0
         AntennaHeight: 1
            SystemLoss: 0
```

```
    ReceiverSensitivity: -100
```

```
show(rx)
```



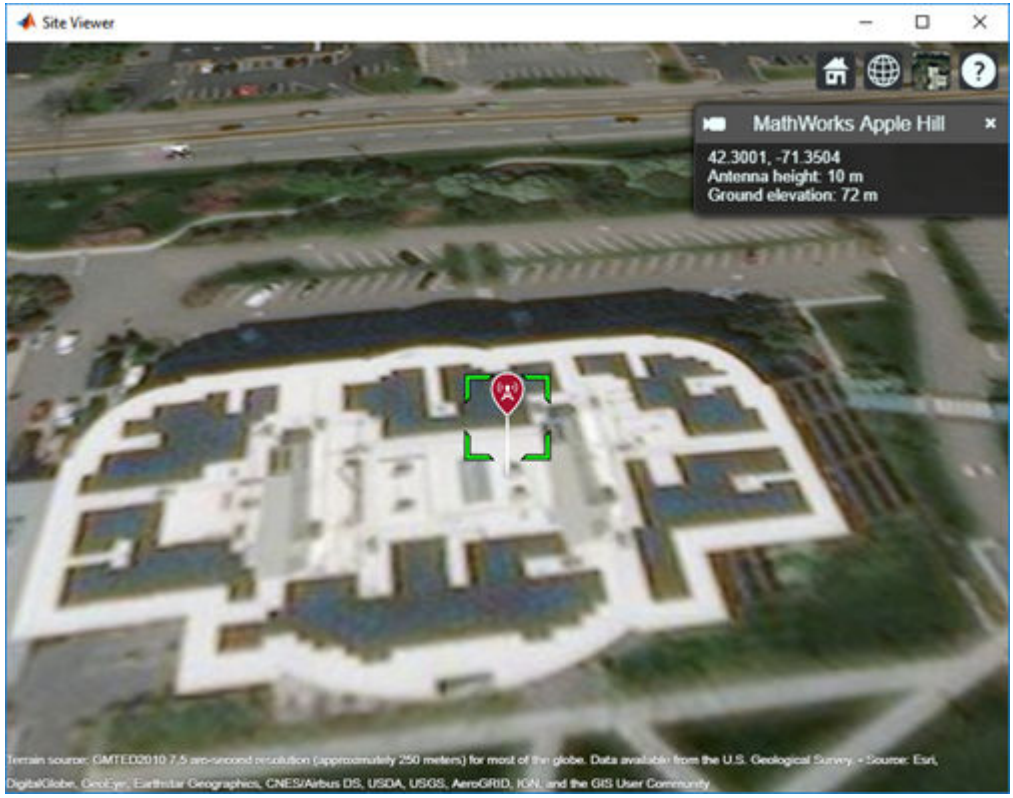
Show and Hide Transmitter Site

Create a transmitter site.

```
tx = txsite('Name','MathWorks Apple Hill',...
       'Latitude',42.3001, ...
       'Longitude',-71.3504);
```

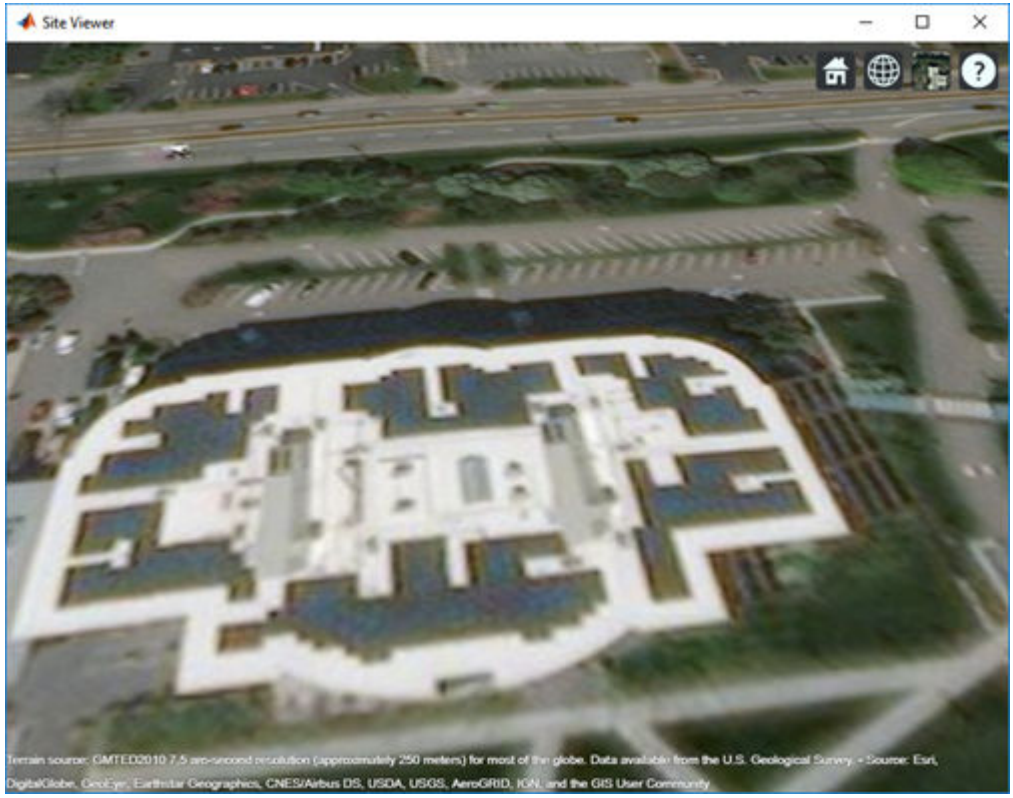
Show the transmitter site.

```
show(tx)
```



Hide the transmitter site.

```
hide(tx)
```



Input Arguments

`site` — Transmitter or receiver site

`txsite` or `rxsite` object | array of `txsite` or `rxsite` objects

Transmitter or receiver site, specified as a `txsite` or `rxsite` object or an array of `txsite` or `rxsite` objects.

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes.

You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `'ClusterMarkers',true`

`Icon` — Image file
character vector

Image file, specified as a character vector.

Data Types: `char`

`ClusterMarkers` — Combine nearby markers into groups or clusters
true | false

Combine nearby markers into groups or clusters, specified as true or false.

Data Types: `char`

See Also

`hide`

Introduced in R2017b

range

Range of radio wave propagation

Syntax

```
r = range(propmodel,tx,pl)
```

Description

`r = range(propmodel,tx,pl)` returns the range of radio wave propagation from the transmitter site.

Examples

Range of Transmitter In Heavy Rain

Specify transmitter and receiver sites.

```
tx = txsite('Name','MathWorks Apple Hill',...
       'Latitude',42.3001, ...
       'Longitude',-71.3504, ...
       'TransmitterFrequency', 2.5e9);

rx = rxsite('Name','Fenway Park',...
       'Latitude',42.3467, ...
       'Longitude',-71.0972);
```

Create the propagation model for heavy rainfall rate.

```
pm = propagationModel('rain','RainRate',50)
```

```
pm =
  Rain with properties:

    RainRate: 50
```

```
        Tilt: 0
```

Calculate the range of transmitter using the rain propagation model and a path loss of 127 dB.

```
r = range(pm,tx,127)
```

```
r = 2.1123e+04
```

Input Arguments

`propmodel` — Propagation model

character vector or string

Propagation model, specified as a character vector or string.

Data Types: `char`

`tx` — Transmitter site

`txsite` object

Transmitter site, specified as a `txsite` object. You can use array inputs to specify multiple sites.

Data Types: `char`

`pl` — Path loss

scalar

Path loss, specified as a scalar in decibels.

Data Types: `double`

Output Arguments

`r` — range

scalar | *M*-by-1 arrays

Range, returned as a scalar or *M*-by-1 array with each element in meters. *M* is the number of TX sites.

Range is the maximum distance for which the path loss does not exceed the value of specified `pl`.

See Also

`pathloss` | `propagationModel`

Introduced in R2017b

pathloss

Path loss of radio wave propagation

Syntax

```
pl = pathloss(propmodel,rx,tx)
```

Description

`pl = pathloss(propmodel,rx,tx)` returns the path loss of radio wave propagation at the receiver site from the transmitter site.

Examples

Path Loss of Receiver In Heavy Rain

Specify the transmitter and the receiver sites.

```
tx = txsite('Name','MathWorks Apple Hill',...
       'Latitude',42.3001, ...
       'Longitude',-71.3504, ...
       'TransmitterFrequency', 2.5e9);

rx = rxsite('Name','Fenway Park',...
       'Latitude',42.3467, ...
       'Longitude',-71.0972);
```

Create the propagation model for heavy rainfall rate.

```
pm = propagationModel('rain','RainRate',50)
```

```
pm =
  Rain with properties:

    RainRate: 50
```

```
        Tilt: 0
```

Calculate the pathloss at the receiver using the rain propagation model.

```
pl = pathloss(pm,rx,tx)
```

```
pl = 127.1559
```

Input Arguments

`propmodel` — Propagation model

character vector or string

Propagation model, specified as a character vector or string.

Data Types: `char`

`rx` — Receiver site

`rxsite` object

Receiver site, specified as a `rxsite` object. You can use array inputs to specify multiple sites.

Data Types: `char`

`tx` — Transmitter site

`txsite` object

Transmitter site, specified as a `txsite` object. You can use array inputs to specify multiple sites.

Data Types: `char`

Output Arguments

`pl` — Path loss

scalar | *M*-by-*N* arrays

Path loss, returned as a scalar or *M*-by-*N* arrays with each element in decibels. *M* is the number of TX sites and *N* is the number of RX sites.

Path loss is computed along the shortest path shortest path through space connecting the transmitter and receiver antenna centers.

See Also

`propagationModel` | `range`

Introduced in R2017b

openFolder

Open file browser to generated Gerber file folder

Syntax

```
openFolder(pcbWriterobject)
```

Description

`openFolder(pcbWriterobject)` opens the parent folder to the PCB writer Gerber design files. You use this function once the Gerber files are generated from the PCB Writer object using the `gerberWrite` function.

Examples

Location of Gerber Files

Create a coplanar inverted F antenna.

```
fco = invertedFcoplanar('Height',14e-3,'GroundPlaneLength', 100e-3,  ...
                  'GroundPlaneWidth', 100e-3);
```

Use this antenna in creating a pcb stack object.

```
p = pcbStack(fco);
```

Use a SMA_Cinch as an RF connector and Mayhew Writer as a manufacturing service.

```
c = PCBConnectors.SMA_Cinch;
s = PCBServices.MayhewWriter;
```

Create an antenna design file using PCBWriter.

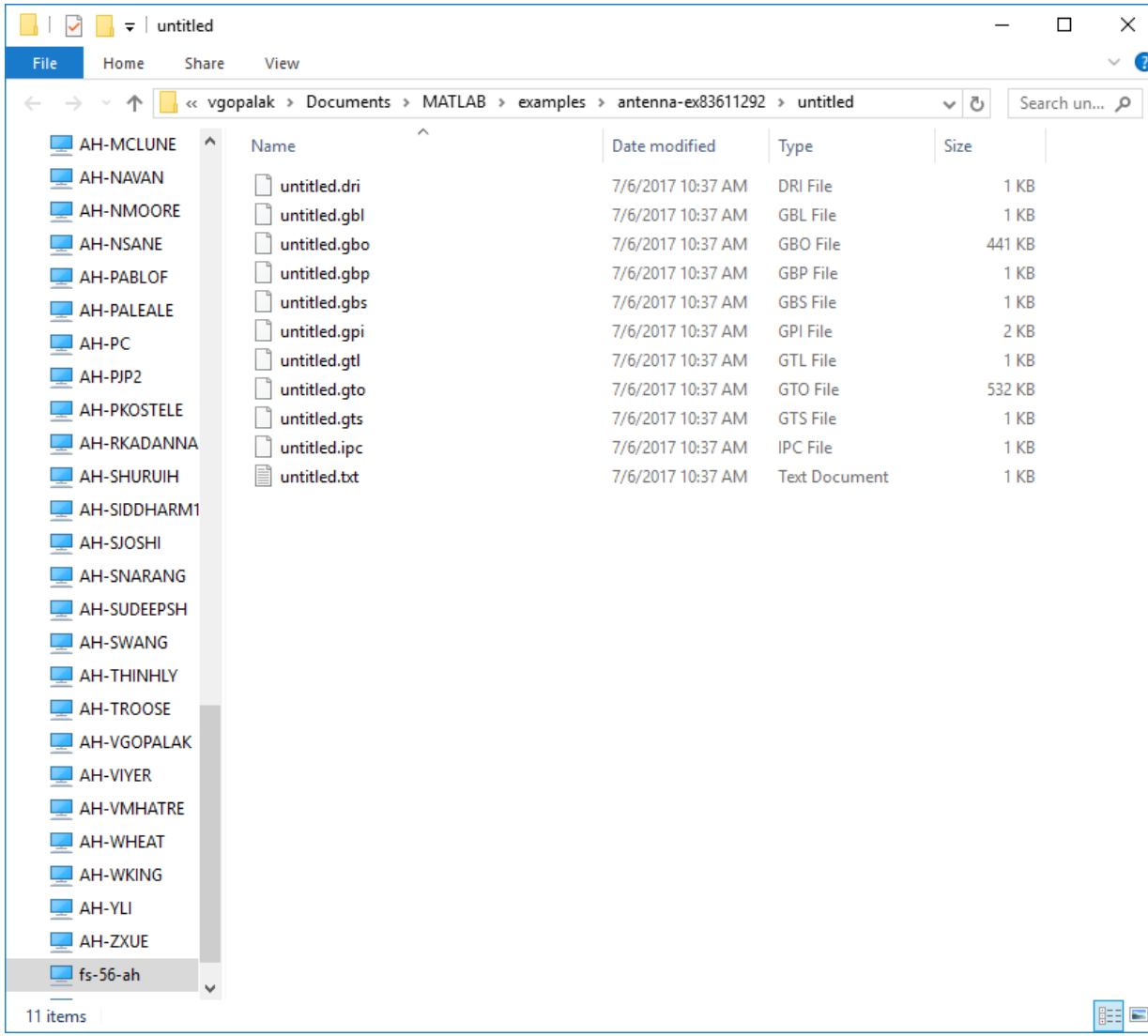
```
PW = PCBWriter(p,s,c);
```

Use the gerberWrite method to create Gerber files from the antenna design files.

```
gerberWrite(PW)
```

Open the folder that contains the Gerber files.

```
openFolder(PW)
```



Input Arguments

`pcbWriterobject` — Antenna design files

`PCBWriter` object

Antenna design files specified as a `PCBWriter` object.

Example: `p1 = pcbStack` creates a PCB stack object.`p1 a = PCBWriter(p1)`.

See Also

`gerberWrite` | `info` | `sendTo`

Introduced in R2017b

info

Display information about antenna or array

Syntax

```
info(antenna)
info(array)
```

Description

`info(antenna)` displays information about antenna element. as a structure:

- `isSolved` – Logical specifying if an antenna is solved.
- `isMeshed` – Logical specifying if an antenna is meshed.
- `MeshingMode` – String specifying the meshing mode.
- `HasSubstrate` – Logical specifying if an antenna uses a substrate.
- `HasLoad` – Logical specifying if an antenna has a load
- `PortFrequency` – Scalar or vector of frequencies used for port analysis.
- `FieldFrequency` – Scalar or vector of frequencies used for field analysis.
- `MemoryEstimate` – Approximate memory requirement for solving the antenna.

`info(array)` displays information about array element as a structure:.

- `isSolved` – Logical specifying if an array is solved.
- `isMeshed` – Logical specifying if an array is meshed.
- `MeshingMode` – String specifying the meshing mode.
- `HasSubstrate` – Logical specifying if an array uses a substrate.
- `HasLoad` – Logical specifying if an array has a load
- `PortFrequency` – Scalar or vector of frequencies used for port analysis.
- `FieldFrequency` – Scalar or vector of frequencies used for field analysis.

- `MemoryEstimate` – Approximate memory requirement for solving the array.

Examples

Antenna Information

Create a dipole antenna and calculate the impedance at 70 MHz.

```
d = dipole;
Z = impedance(d,70e6)
```

```
Z = 72.9288 - 1.3947i
```

Display all the information about the dipole antenna.

```
info(d)
```

```
ans = struct with fields:
          IsSolved: "true"
          IsMeshed: "true"
       MeshingMode: "auto"
      HasSubstrate: "false"
           HasLoad: "false"
     PortFrequency: 70000000
    FieldFrequency: []
    MemoryEstimate: "740 MB"
```

Input Arguments

`antenna` — Antenna element

antenna object

Antenna element, specified as an antenna object.

Example: `d = dipole;`

`array` — Array element

array object

Array element, specified as an array object.

Example: `d = dipole;`

See Also

`show`

Introduced in R2017b

sendTo

Export generated Gerber Files to service provider

Syntax

```
sendTo(pcbWriterobject)
```

Description

`sendTo(pcbWriterobject)` opens the manufacturing service browser page on your default web browser and opens the folder containing the Gerber files.

For example, if the manufacturing service is MayhewWriter, then `sendTo` action opens the Mayhew Labs online PCB viewer in your default web browser. This function also opens the folder containing the Gerber files. This simplifies use of the service, enabling you to drag and drop the files to the website and view the design.

Examples

Open Manufacturing Service Website

Create a coplanar inverted F antenna.

```
fco = invertedFcoplanar('Height',14e-3,'GroundPlaneLength', 100e-3,  ...
                  'GroundPlaneWidth', 100e-3);
```

Use this antenna in creating a pcb stack object.

```
p = pcbStack(fco);
```

Use a SMA_Cinch as an RF connector and Mayhew Writer as a manufacturing service.

```
c = PCBConnectors.SMA_Cinch;
s = PCBServices.MayhewWriter;
```

Create an antenna design file using PCBWriter.

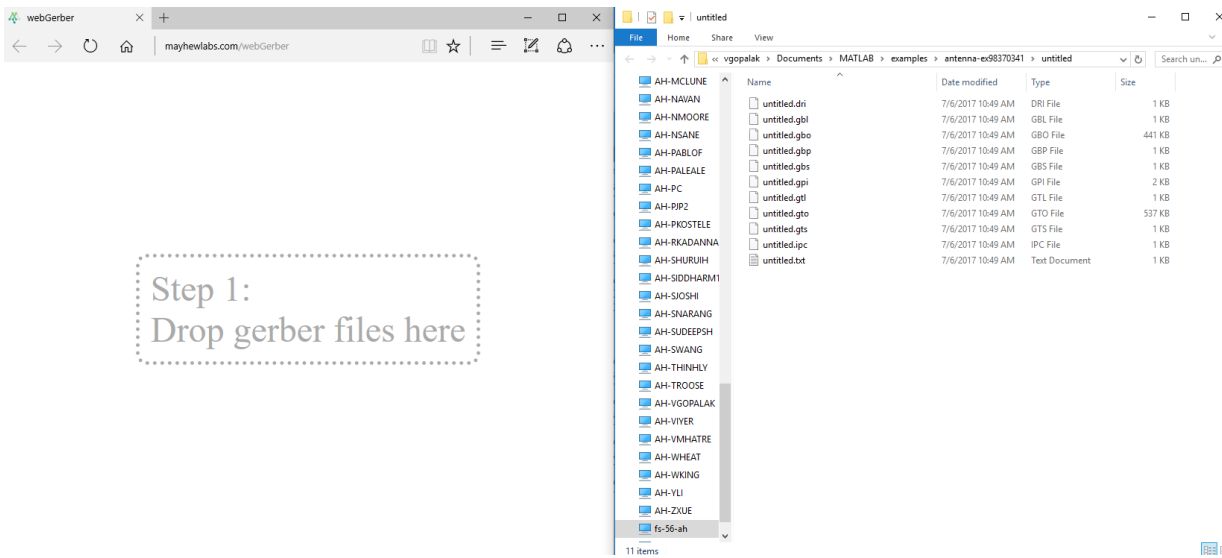
```
PW = PCBWriter(p,s,c);
```

Use the gerberWrite method to create Gerber files from the antenna design files.

```
gerberWrite(PW)
```

Open the manufacturing service website to send the Gerber files.

```
sendTo(PW)
```



Input Arguments

`pcbWriter` object — Antenna design files

`PCBWriter` object

Antenna design files, specified as a `PCBWriter` object.

Example: `p1 = pcbStack` creates a PCB stack object.`p1 a = PCBWriter(p1)`.

See Also

`gerberWrite` | `info` | `sendTo`

Introduced in R2017b

getLowPassLocs

Feeding location to operate birdcage as lowpass coil

Syntax

```
getLowPassLocs(birdcageantenna)
```

Description

`getLowPassLocs(birdcageantenna)` returns all the feed locations on the birdcage to operate the antenna as a lowpass coil. The feeding locations are located in the center of the rungs. Use this function to find the `FeedLocations` property value for `birdcage`.

Examples

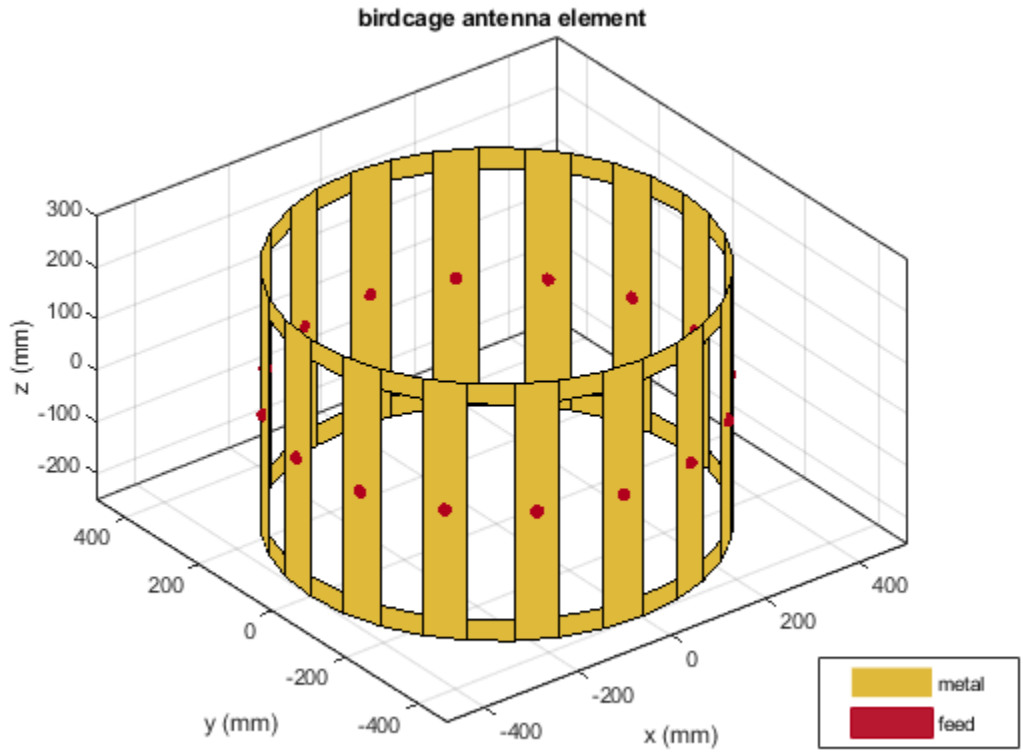
Birdcage as Lowpass Coil

```
b = birdcage;
b.FeedLocations = getLowPassLocs(b)
```

```
b =
  birdcage with properties:

         NumRungs: 16
       CoilRadius: 0.4000
       CoilHeight: 0.0400
       RungHeight: 0.4600
     ShieldRadius: 0
     ShieldHeight: 0
          Phantom: []
    FeedLocations: [16x3 double]
      FeedVoltage: 1
        FeedPhase: 0
             Tilt: 0
         TiltAxis: [1 0 0]
```

```
                Load: [1x1 lumpedElement]
```

```
show(b)
```



Input Arguments

`birdcageantenna` — Birdcage antenna

object

Birdcage antenna, specified as an object.

Example: `b = birdcage b.FeedLocations = getLowPassLocs(b)`

See Also

Introduced in R2017b

getHighPassLocs

Feeding location to operate birdcage as highpass coil

Syntax

```
getHighPassLocs(birdcageantenna)
```

Description

`getHighPassLocs(birdcageantenna)` returns all the feed locations on the birdcage to operate the antenna as a highpass coil. The feeding locations are along the circumference on the upper and lower coils of the birdcage. Use this function to find the `FeedLocations` property value for `birdcage`.

Examples

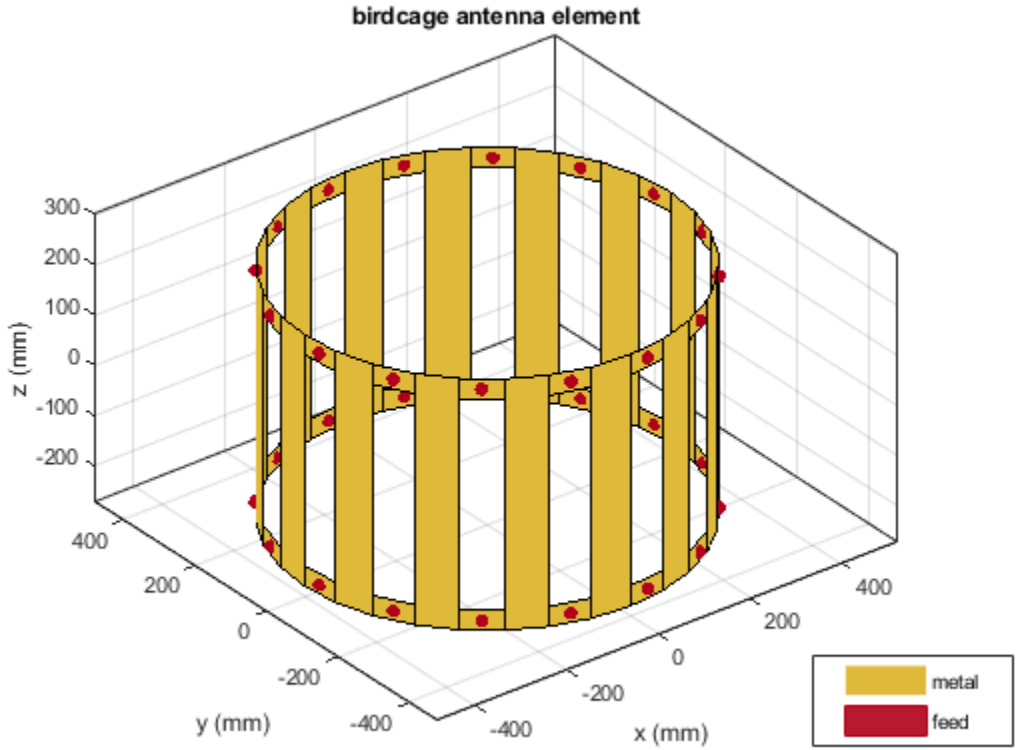
Birdcage as Highpass Coil

```
b = birdcage;
b.FeedLocations = getHighPassLocs(b)
```

```
b =
  birdcage with properties:

         NumRungs: 16
       CoilRadius: 0.4000
       CoilHeight: 0.0400
       RungHeight: 0.4600
     ShieldRadius: 0
     ShieldHeight: 0
          Phantom: []
    FeedLocations: [32x3 double]
      FeedVoltage: 1
        FeedPhase: 0
             Tilt: 0
```

```
    TiltAxis: [1 0 0]
        Load: [1x1 lumpedElement]
```

```
show(b)
```



Input Arguments

`birdcageantenna` — Birdcage antenna

object

Birdcage antenna, specified as an object.

Example: `b = birdcage b.FeedLocations = getHighPassLocs(b)`

See Also

Introduced in R2017b

rotateX

Rotate shape about X-axis and angle

Syntax

```
rotateX(shape,angle)
c =c rotateX(shape,angle)
```

Description

`rotateX(shape,angle)` rotate shape about an axes object and angle.

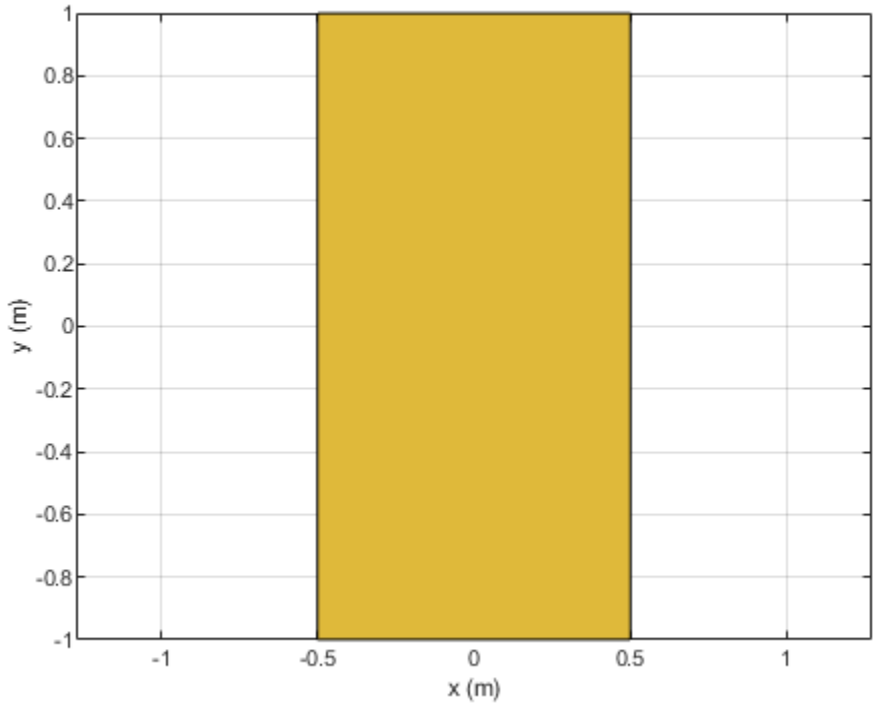
`c =c rotateX(shape,angle)` rotate shape about an axes object and angle.

Examples

Rotate Rectangle About X-Axis

Create a rectangle shape.

```
r = antenna.Rectangle;
show(r)
axis equal
```

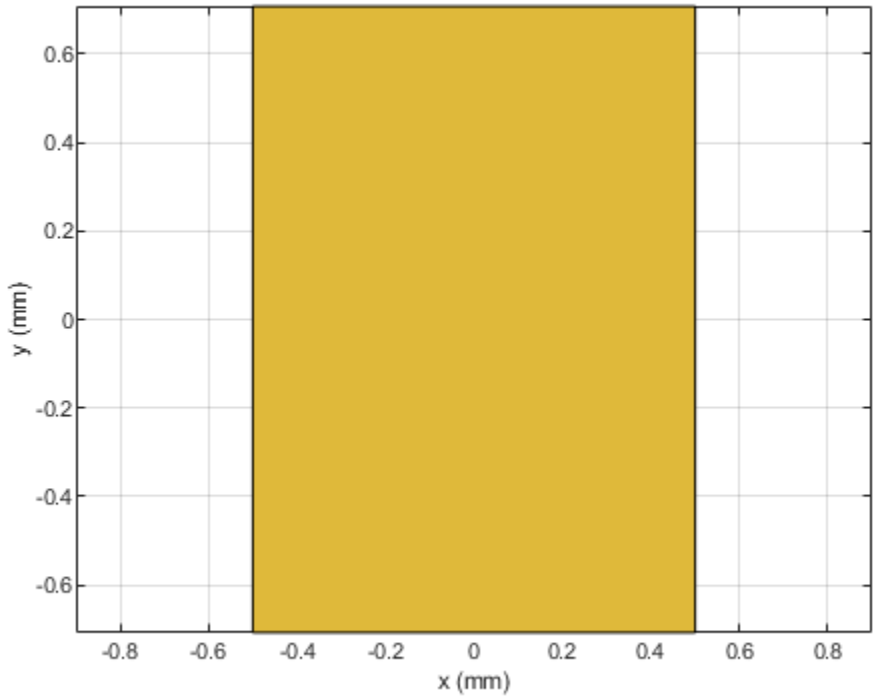


Rotate the rectangle at 45 degrees about the x-axis.

```
r1 = rotateX(r,45)
show(r1)
axis equal
```

```
r1 =

  Rectangle with properties:

        Name: 'myrectangle'
      Center: [0 0]
      Length: 1
```

```
    Width: 2
NumPoints: 2
```



Input Arguments

shape — Shape created using custom elements and shape objects

object handle

Shape created using custom elements and shape objects of Antenna Toolbox, specified as an object handle.

Example: `area(rectangle)` where rectangle is created using `antenna.Rectangle` object.

`angle` — Angle of rotation
scalar

Angle of rotation, specified as a scalar in degrees

Example: `rotateX(rectangle,45)` rotates the rectangle around X-axis by 45 degrees.

Data Types: `double`

See Also

`add` | `area` | `intersect` | `mesh` | `plot` | `rotate` | `rotateY` | `rotateZ` | `scale` | `show` | `subtract` | `translate`

Introduced in R2017a

rotateY

Rotate shape about Y-axis and angle

Syntax

```
rotateY(shape,angle)
c = rotateY(shape,angle)
```

Description

`rotateY(shape,angle)` rotate shape about the Y-axis and angle.

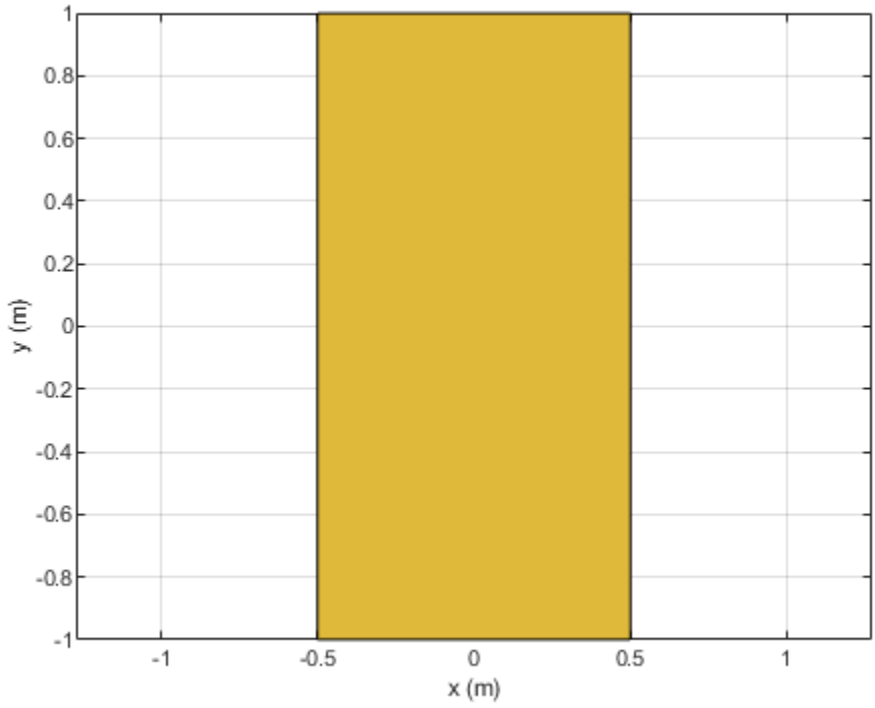
`c = rotateY(shape,angle)` rotate shape about the Y-axis and angle.

Examples

Rotate Rectangle About Y-Axis

Create a rectangle shape.

```
r = antenna.Rectangle;
show(r)
axis equal
```

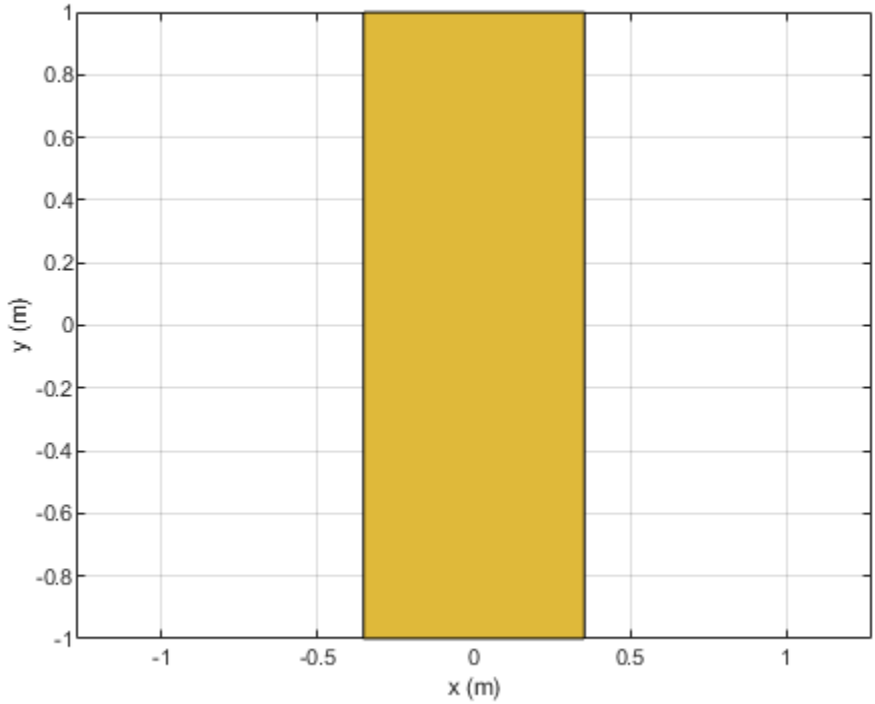


Rotate the rectangle at 45 degrees about the Y-axis.

```
r1 = rotateY(r,45)
show(r1)
axis equal
```

```
r1 =

  Rectangle with properties:

        Name: 'myrectangle'
      Center: [0 0]
      Length: 1
```

```
    Width: 2
NumPoints: 2
```



Input Arguments

shape — Shape created using custom elements and shape objects

object handle

Shape created using custom elements and shape objects of Antenna Toolbox, specified as an object handle.

Example: `rotateY(rectangle)` where rectangle is created using `antenna.Rectangle` object.

`angle` — Angle of rotation
scalar

Angle of rotation, specified as a scalar in degrees

Example: `rotateY(rectangle,45)` rotates the rectangle around Y-axis by 45 degrees.

Data Types: `double`

See Also

`add` | `area` | `intersect` | `mesh` | `plot` | `rotate` | `rotateX` | `rotateZ` | `scale` | `show` | `subtract` | `translate`

Introduced in R2017a

rotateZ

Rotate shape about Z-axis and angle

Syntax

```
rotateZ(shape,angle)
c = rotateZ(shape,angle)
```

Description

`rotateZ(shape,angle)` rotate shape about the Z-axis and angle.

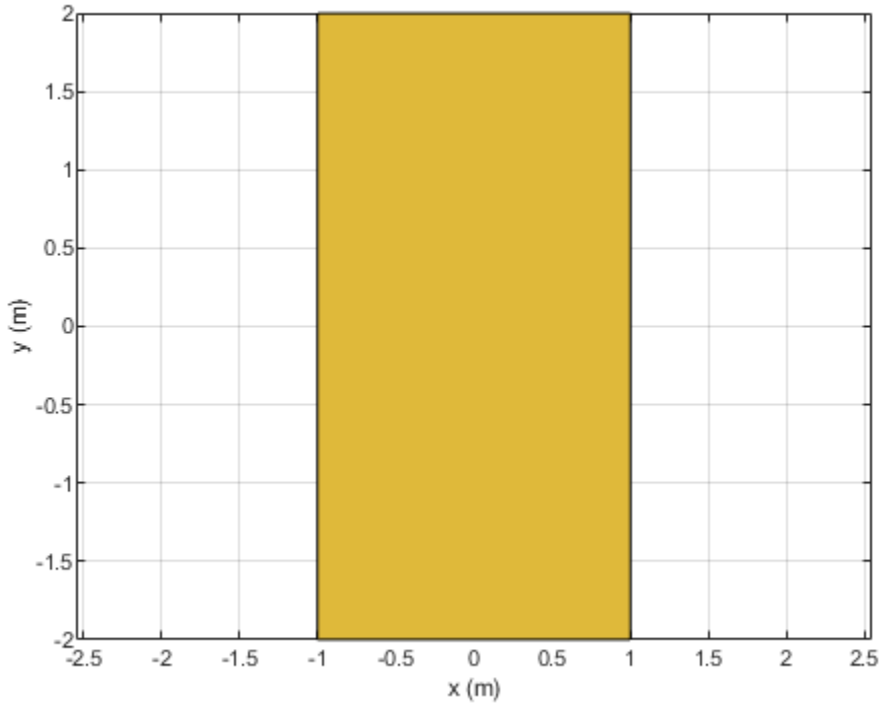
`c = rotateZ(shape,angle)` rotate shape about the Z-axis and angle.

Examples

Create and Rotate Rectangle Using Specified Properties

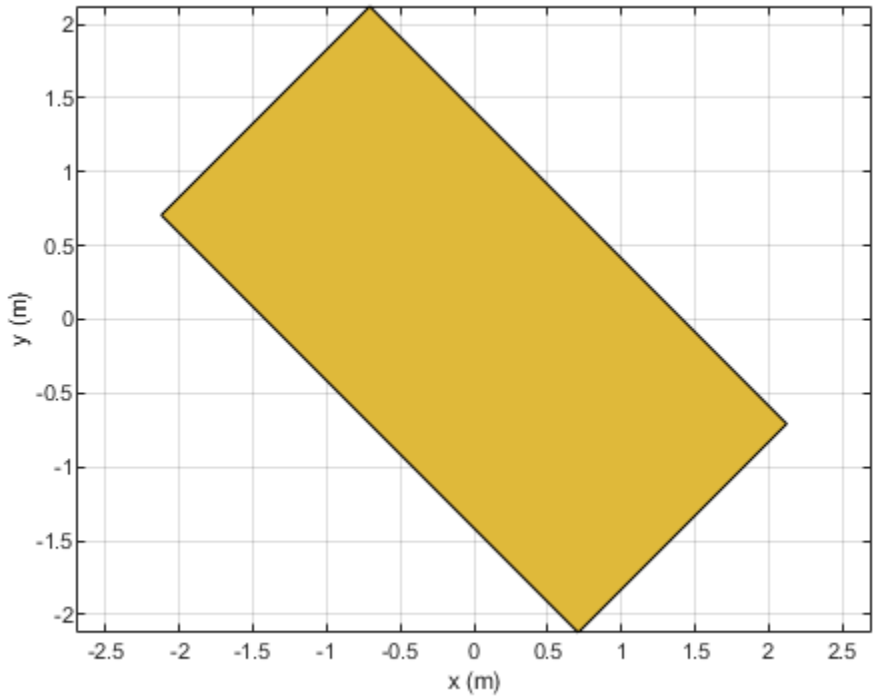
Create and view a rectangle with a length of 2 m and a width of 4 m.

```
r2 = antenna.Rectangle('Length',2,'Width',4);
show(r2)
axis equal
```



Rotate the rectangle.

```
rotateZ(r2,45);
show(r2)
```



Input Arguments

shape — Shape created using custom elements and shape objects

object handle

Shape created using custom elements and shape objects of Antenna Toolbox, specified as an object handle.

Example: `rotateZ(rectangle)` where rectangle is created using `antenna.Rectangle` object.

`angle` — Angle of rotation
scalar

Angle of rotation, specified as a scalar in degrees

Example: `rotateZ(rectangle,45)` rotates the rectangle around Z-axis by 45 degrees.

Data Types: `double`

See Also

`add` | `area` | `intersect` | `mesh` | `plot` | `rotate` | `rotateX` | `rotateY` | `scale` | `show` | `subtract` | `translate`

Introduced in R2017a

translate

Move shape to new location

Syntax

```
c = translate(shape,locationpoints)
```

Description

`c = translate(shape,locationpoints)` moves the shape to a new specified location using a translation vector.

Examples

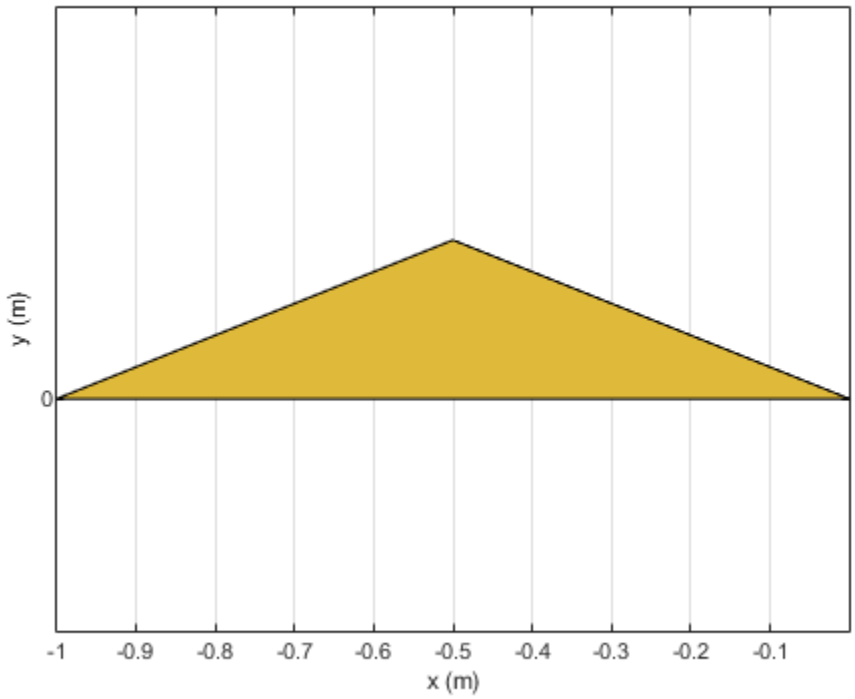
Create and Transform Polygon

Create a polygon using `antenna.Polygon` with vertices at `[-1 0 0;-0.5 0.2 0;0 0 0]` and view it.

```
p = antenna.Polygon('Vertices', [-1 0 0;-0.5 0.2 0;0 0 0])
```

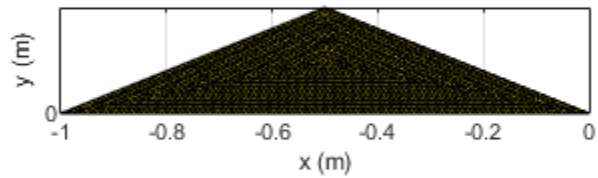
```
p =
  Polygon with properties:

        Name: 'mypolygon'
    Vertices: [3x3 double]
```

```
show(p)
axis equal
```



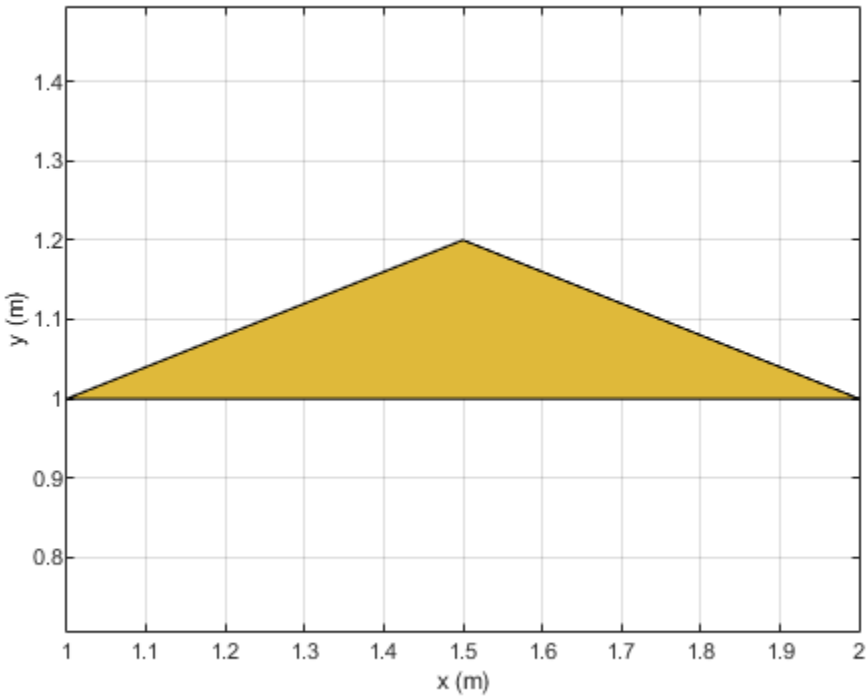
Mesh the polygon and view it.

```
mesh(p,0.2)
```



Move the polygon to a new location on the X-Y plane.

```
translate(p,[2,1,0])
axis equal
```



Input Arguments

shape — Shape created using custom elements and shape objects

object handle

Shape created using custom elements and shape objects of Antenna Toolbox, specified as an object handle.

Example: `c = translate(rectangle1,[2 1 0])` where rectangle1 is created using `antenna.Rectangle` object.

`locationpoints` — Translation vector
vector

Translation vector, specified as a vector.

Data Types: `double`

See Also

`add` | `area` | `intersect` | `mesh` | `plot` | `rotate` | `rotateX` | `rotateY` | `rotateZ` | `scale` | `show` | `subtract`

Introduced in R2017a

plot

Plot boundary of shape

Syntax

```
p = plot(shape,varargin)
```

Description

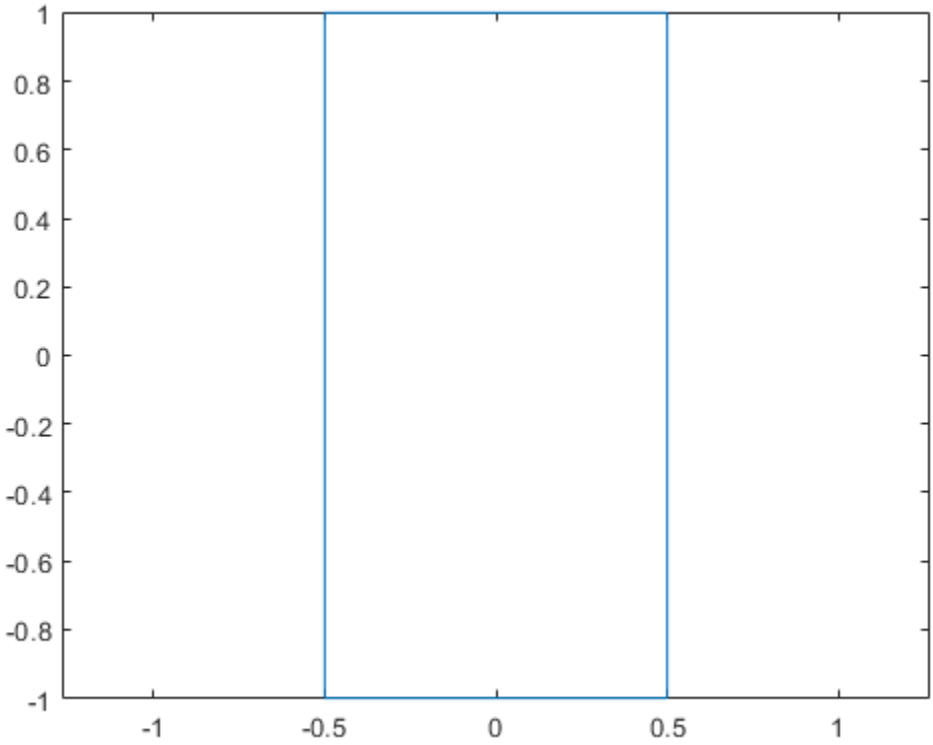
`p = plot(shape,varargin)` plots the boundary of the shape and returns the line handle.

Examples

Plot Rectangle Shape

Create a rectangular shape and plot it.

```
r = antenna.Rectangle;
p = plot(r);
```



Input Arguments

shape — Shape created using custom elements and shape objects
object handle

Shape created using custom elements and shape objects of Antenna Toolbox, specified as an object handle.

Example: `plot(rectangle)` where rectangle is created using `antenna.Rectangle` object.

See Also

`mesh` | `show`

Introduced in R2017a

scale

Change the size of the shape by a fixed amount

Syntax

```
c = scale(shape,scaling)
```

Description

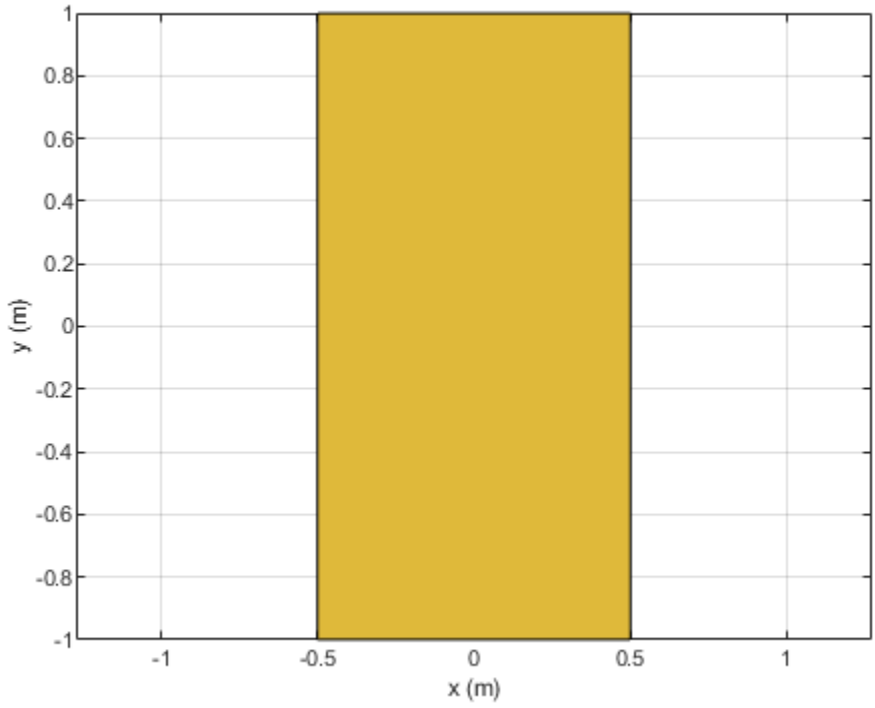
`c = scale(shape,scaling)` scales the shape by a constant factor

Examples

Scale Rectangle Shape

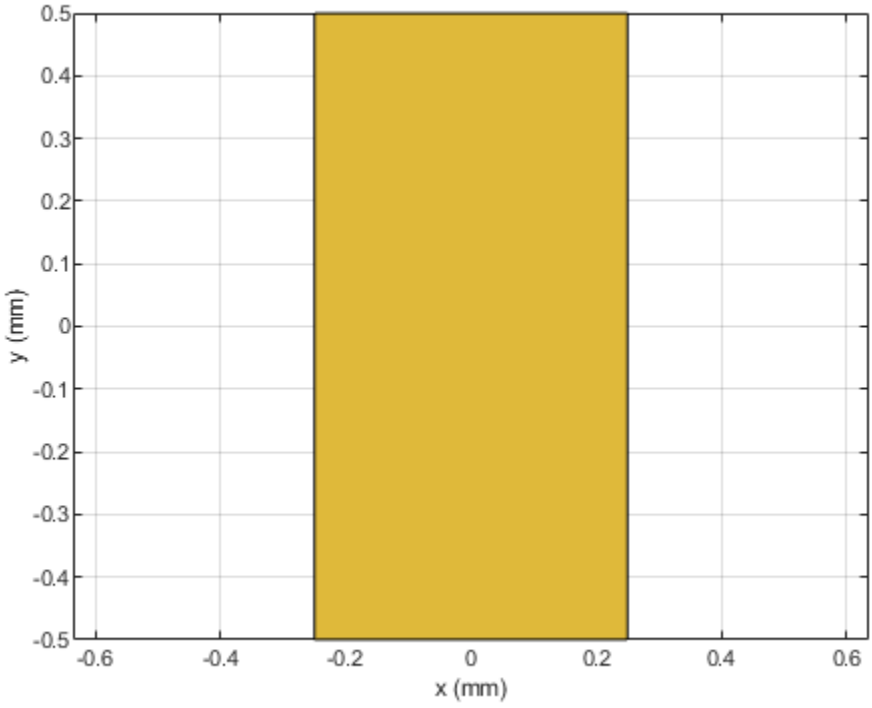
Create a rectangular shape.

```
r  = antenna.Rectangle;
show(r)
axis equal
```



Shrink the rectangle by 50%.

```
scale(r,0.5);
```



Input Arguments

shape — Shape created using custom elements and shape objects

object handle

Shape created using custom elements and shape objects of Antenna Toolbox, specified as an object handle.

Example: `c = scale(rectangle1,0.5)` where rectangle1 is created using `antenna.Rectangle` object.

`scaling` — Constant factor to change shape size
scalar

Constant factor to change shape size, specified as a scalar.

Data Types: `double`

See Also

`add` | `area` | `intersect` | `mesh` | `plot` | `rotate` | `rotateX` | `rotateY` | `rotateZ` | `show` | `subtract`

Introduced in R2017a

plus

Shape1 + Shape2

Syntax

```
c = plus(shape1,shape2)
```

Description

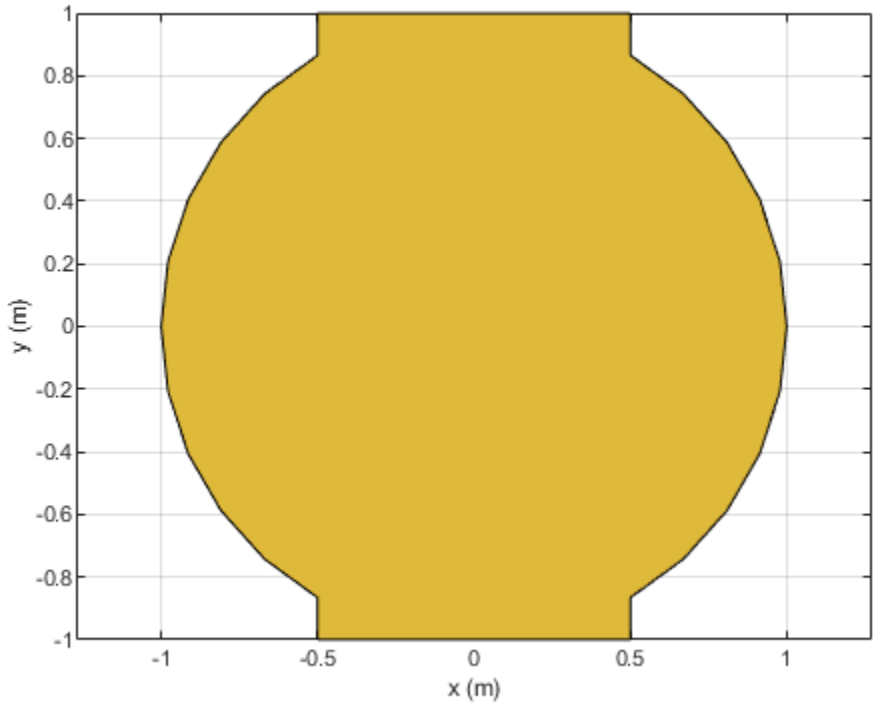
`c = plus(shape1,shape2)` calls the syntax `shape1 + shape2` to unite two shapes.

Examples

Unite Rectangle and Circle

Create a rectangular and circular shape and unite them.

```
r  = antenna.Rectangle;
c  = antenna.Circle;
r+c;
```



Input Arguments

shape1, shape2 — Shapes created using custom elements and shape objects
object handle

Shapes created using custom elements and shape objects of Antenna Toolbox, specified as an object handle.

Example: `rectangle1+rectangle2` where rectangle1 and rectangle2 are shapes created using `antenna.Rectangle` object.

See Also

`add` | `area` | `intersect` | `mesh` | `plot` | `rotate` | `rotateX` | `rotateY` | `rotateZ` | `scale` | `show` | `subtract` | `translate`

Introduced in R2017a

minus

Shape1 - Shape2

Syntax

```
c = minus(shape1,shape2)
```

Description

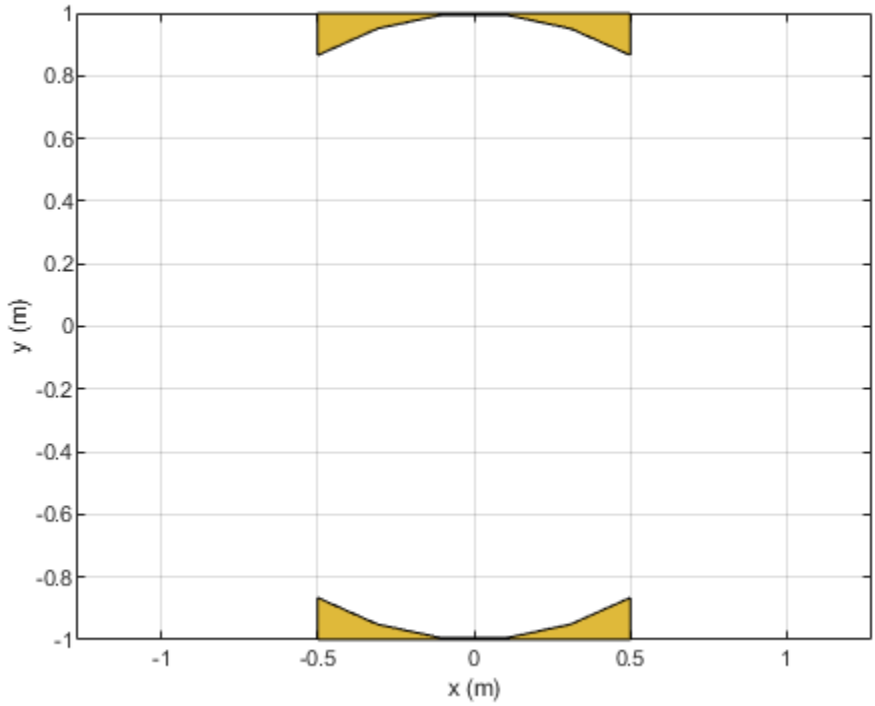
`c = minus(shape1,shape2)` calls the syntax `shape1 - shape2` to subtract two shapes.

Examples

Subtract Rectangle and Circle

Create a rectangular and circular shape and subtract them.

```
r  = antenna.Rectangle;
c  = antenna.Circle;
r-c;
```



Input Arguments

shape1, shape2 — Shapes created using custom elements and shape objects
object handle

Shapes created using custom elements and shape objects of Antenna Toolbox, specified as an object handle.

Example: `rectangle1-rectangle2` where rectangle1 and rectangle2 are shapes created using `antenna.Rectangle` object.

See Also

add | area | intersect | mesh | plot | rotate | rotateX | rotateY | rotateZ | scale | show | subtract | translate

Introduced in R2017a

and

Shape1 & Shape2

Syntax

```
c = and(shape1,shape2)
```

Description

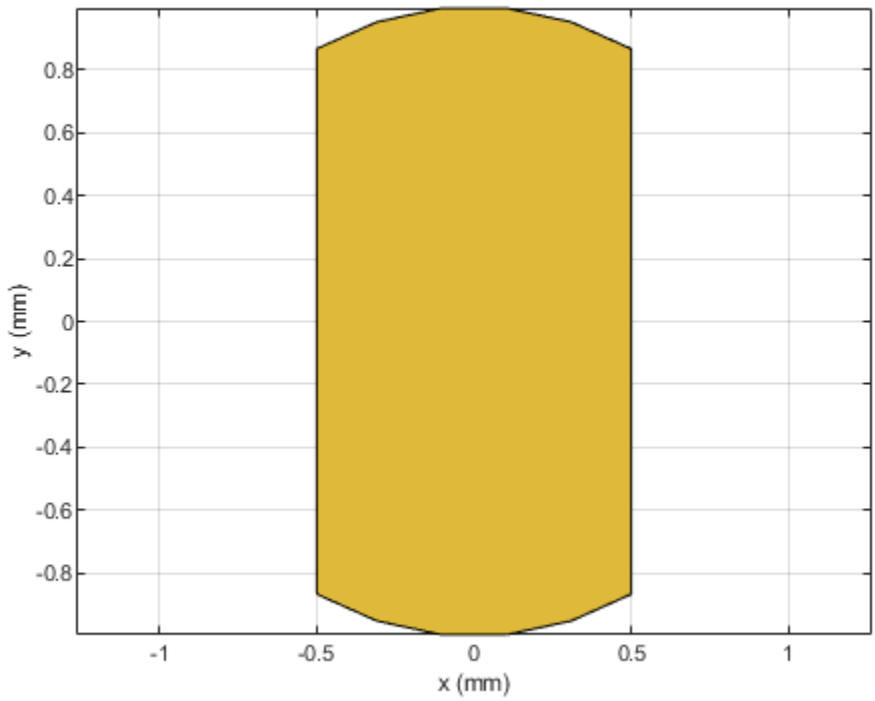
`c = and(shape1,shape2)` calls the syntax `shape1 & shape2` to intersect two shapes.

Examples

Intersect Rectangle and Circle

Create a rectangular and circular shape and intersect them.

```
r  = antenna.Rectangle;
c  = antenna.Circle;
r&c;
```



Input Arguments

shape1, shape2 — Shapes created using custom elements and shape objects
object handle

Shapes created using custom elements and shape objects of Antenna Toolbox, specified as an object handle.

Example: `rectangle1&rectangle2` where rectangle1 and rectangle2 are shapes created using `antenna.Rectangle` object.

See Also

add | area | intersect | mesh | plot | rotate | rotateX | rotateY | rotateZ | scale | show | subtract | translate

Introduced in R2017a

add

Add additional data to existing Smith chart

Syntax

```
add(plot,data)
add(plot,frequency,data)
```

Description

`add(plot,data)` adds data to an existing Smith chart.

`add(plot,frequency,data)` adds data to an existing Smith chart based on multiple data sets containing frequencies corresponding to columns of data matrix.

Examples

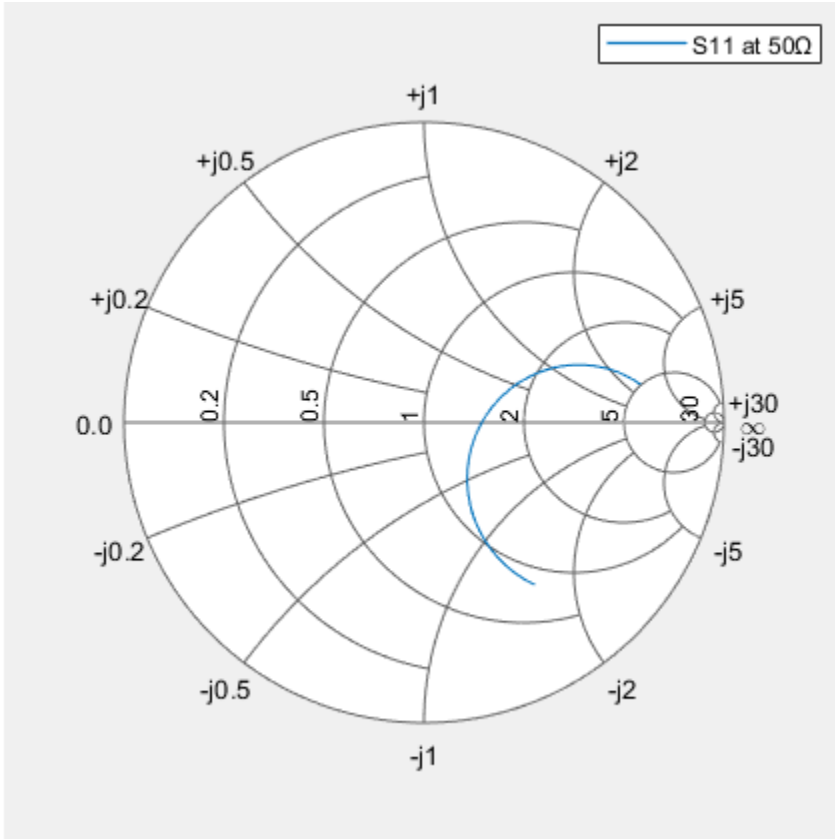
Add S-Parameter Data to an Existing Smith Plot

Plot the reflection coefficients of a dipole antenna.

Create a strip dipole antenna on the Y-Z plane. Calculate the complex s-parameters of the dipole antenna from 60 MHz to 90 MHz, with an interval of 150 kHz.

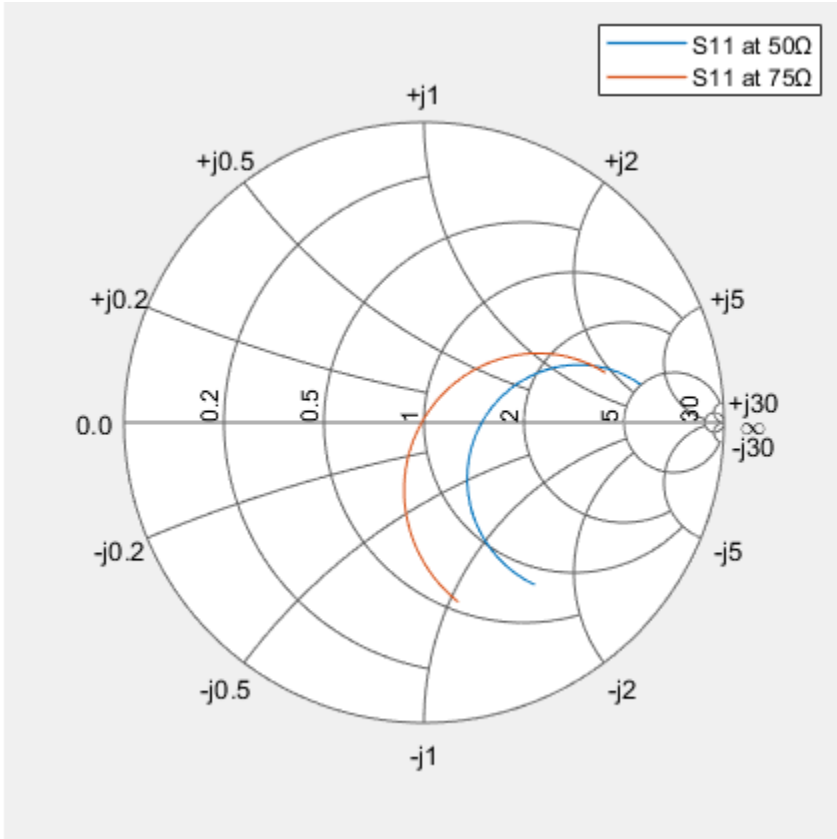
Plot S11 on a Smith plot for a reference impedance of 50 ohm.

```
d = dipole;
freq = linspace(60e6, 90e6, 200);
s_50 = sparameters(d, freq,50);
hg = smithplot(s_50,[1,1]);
hg.LegendLabels = {"S11 at 50#ohm"};
```



Find S11 for a new impedance of 75 ohm. Add new S11 to the existing Smith plot.

```
s_75 = sparameters(d, freq, 75);
gamma = rfparam(s_75,1,1);
add(hg, gamma);
hg.LegendLabels = {"S11 at 50#ohm","S11 at 75#ohm"};
```



Input Arguments

`plot` — Smith chart
function handle

Smith chart handle, specified as a function handle. If the handle of the Smith chart is not retained during creation, it is obtained by using the command `p = smithplot('gco')`.

Data Types: `double`

`data` — Input data
complex vector | complex matrix

Input data, specified as a complex vector or complex matrix.

For a matrix *D*, the columns of *D* are independent data sets. For *N*-by-*D* arrays, dimensions 2 and greater are independent data sets.

Data Types: `double`
Complex Number Support: Yes

`frequency` — Frequency data
real vector

Frequency data, specified as a real vector.

Data Types: `double`

See Also

`replace` | `smithplot`

Introduced in R2017b

replace

Remove current data and add new data to Smith chart

Syntax

```
replace(plot,data)
replace(plot,frequency,data)
```

Description

`replace(plot,data)` removes all current data from a Smith chart, `plot`, and adds new data to the Smith chart.

`replace(plot,frequency,data)` removes all current data and adds new data to the Smith chart based on multiple data sets containing frequencies corresponding to columns of the data matrix.

Examples

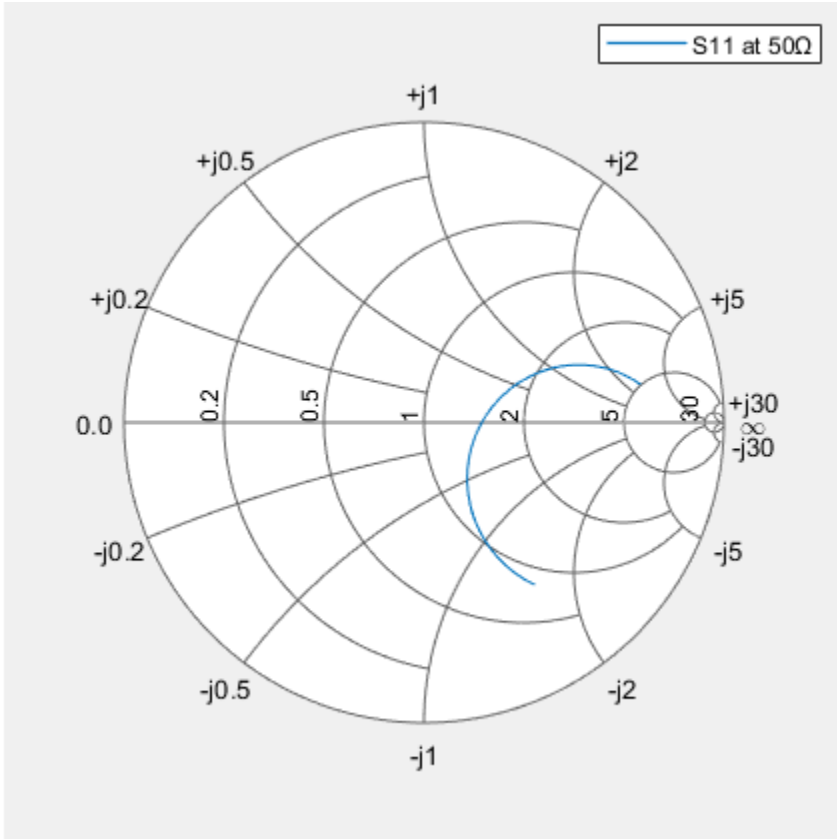
Replace S-Parameter Data on Existing Smith Chart

Plot the reflection coefficients of a dipole antenna.

Create a strip dipole antenna on the Y-Z plane. Calculate the complex S-parameters of the dipole antenna from 60 MHz to 90 MHz, with an interval of 150 kHz.

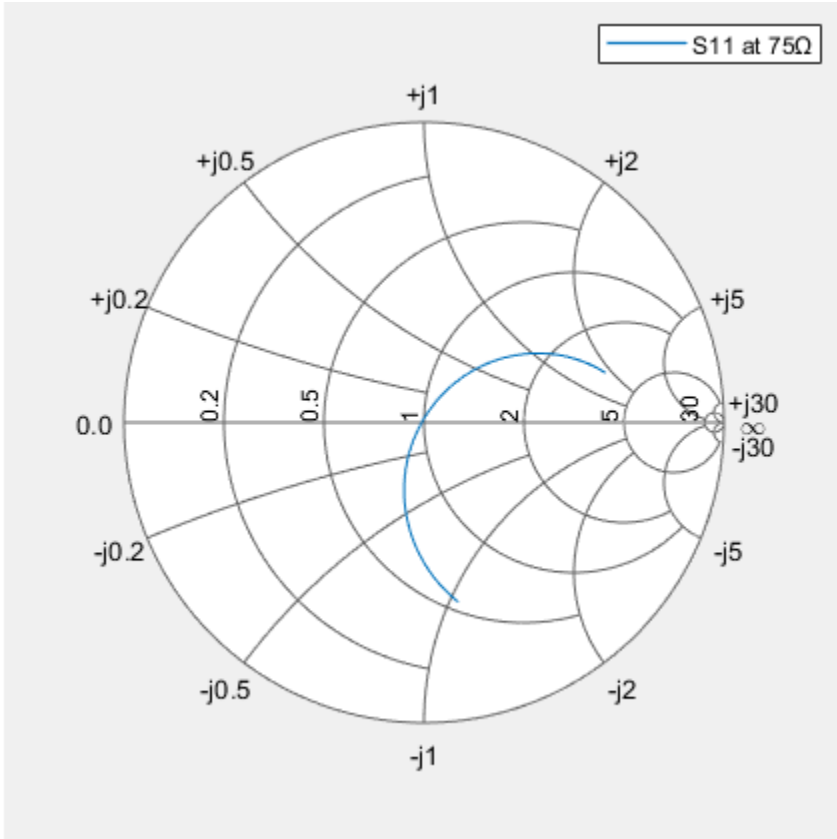
Plot S11 on a Smith chart for a reference impedance of 50 ohm.

```
d = dipole;
freq = linspace(60e6,90e6,200);
s_50 = sparameters(d,freq,50);
hg = smithplot(s_50,[1,1]);
hg.LegendLabels = 'S11 at 50#ohm';
```



Find S11 for a new impedance of 75 ohm. Replace the old S11 by the new S11 on the existing Smith chart.

```
s_75 = sparameters(d,freq,75);
gamma = rfparam(s_75,1,1);
replace(hg,gamma);
hg.LegendLabels = 'S11 at 75#ohm';
```



Input Arguments

`plot` — Smith plot

plot handle

Smith chart handle, specified as a plot handle. If the handle of the Smith chart is not retained during creation, use `p = smithplot('gco')`.

`data` — Input data

complex vector | complex matrix

Input data, specified as a complex vector or complex matrix.

For a matrix *D*, the columns of *D* are independent datasets. For *N*-by-*D* arrays, dimensions 2 and greater are independent datasets.

Data Types: `double`
Complex Number Support: Yes

`frequency` — Frequency data
real vector

Frequency data, specified as a real vector.

Data Types: `double`

See Also

`add` | `smithplot`

Introduced in R2017b

smithplot

Plot measurement data on Smith chart

Syntax

```
smithplot(data)
smithplot(frequency,data)
smithplot(ax, ___ )
smithplot(hnet)
smithplot(hnet,i,j)
smithplot(hnet,[i1,j1;i2,j2;....,in,jn])
s = smithplot( ___ )
s = smithplot('gco')
smithplot( ___ ,Name,Value)
```

Description

`smithplot(data)` creates a Smith chart based on input data values.

Note The Smith chart is commonly used to display the relationship between a reflection coefficient, typically given as S11 or S22, and a normalized impedance.

`smithplot(frequency,data)` creates a Smith chart based on frequency and data values.

`smithplot(ax, ___ )` creates a Smith chart with a user defined axes handle, `ax`, instead of the current axes handle. Axes handles are not supported for network parameter objects. This parameter can be used with either of the two previous syntaxes.

`smithplot(hnet)` plots all the network parameter objects in `hnet`.

`smithplot(hnet,i,j)` plots the (*i*, *j*)th parameter of `hnet`. `hnet` is a network parameter object.

`smithplot(hnet,[i1,j1;i2,j2;....,in,jn])` plots multiple parameters ($i_1, j_1, i_2, j_2, \ldots, i_n, j_n$) of `hnet`. `hnet` is a network parameter object.

`s = smithplot( ___ )` returns a Smith chart object handle so you can customize the plot and add measurements.

`s = smithplot('gco')` returns a Smith chart object handle of the current plot. This syntax is useful when the function handle, `p` was not returned or retained.

`smithplot( ___ ,Name,Value)` creates a Smith chart with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding property value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Examples

Plot the Reflection Coefficient of a Dipole Antenna

Smith Plot of the Reflection Coefficient of a Dipole Antenna

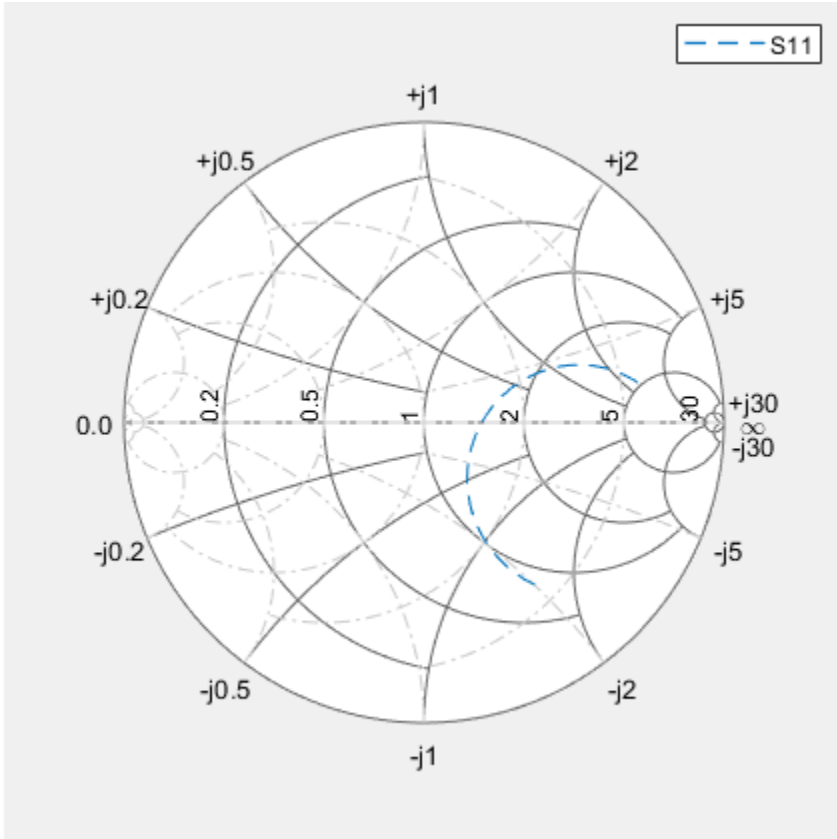
Create a strip dipole antenna on the Y-Z plane. Calculate the complex s-parameters of the dipole antenna from 60 MHz to 90 MHz, with an interval of 150 kHz.

Plot the S11 on a Smith plot.

```
d = dipole;
freq = linspace(60e6, 90e6, 200);
s = sparameters(d, freq);
hg = smithplot(s,1,1, 'GridType','ZY')
```

```
hg =
  smithplot with properties:

         Data: [200x1 double]
    Frequency: [200x1 double]

   Show all properties, methods
```

```
hg.LineStyle = '--';
```

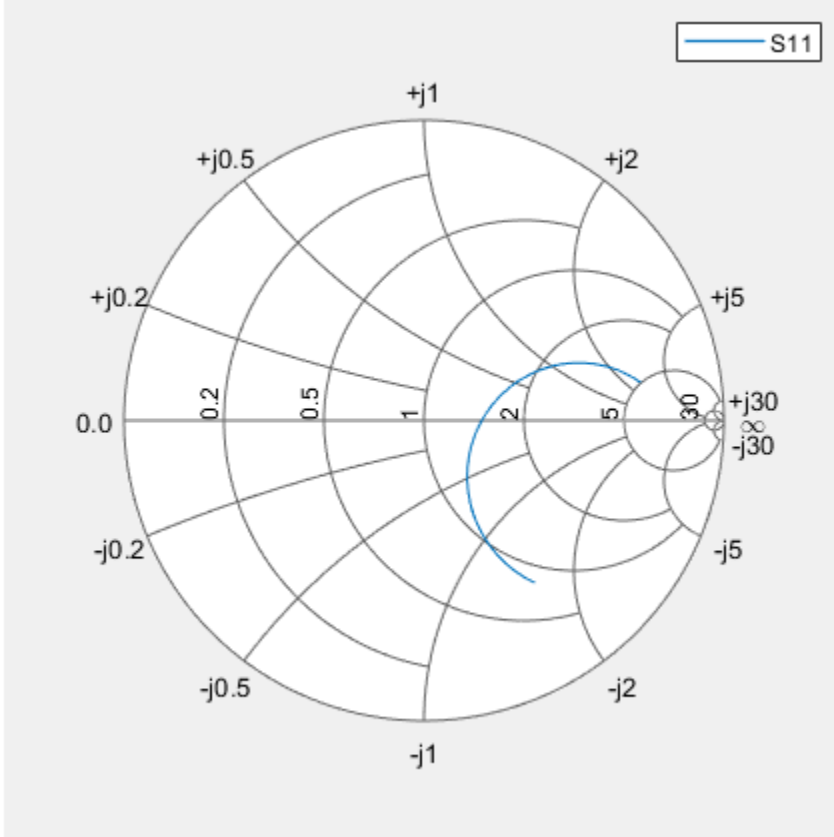


Smith Plot Interactive Menu

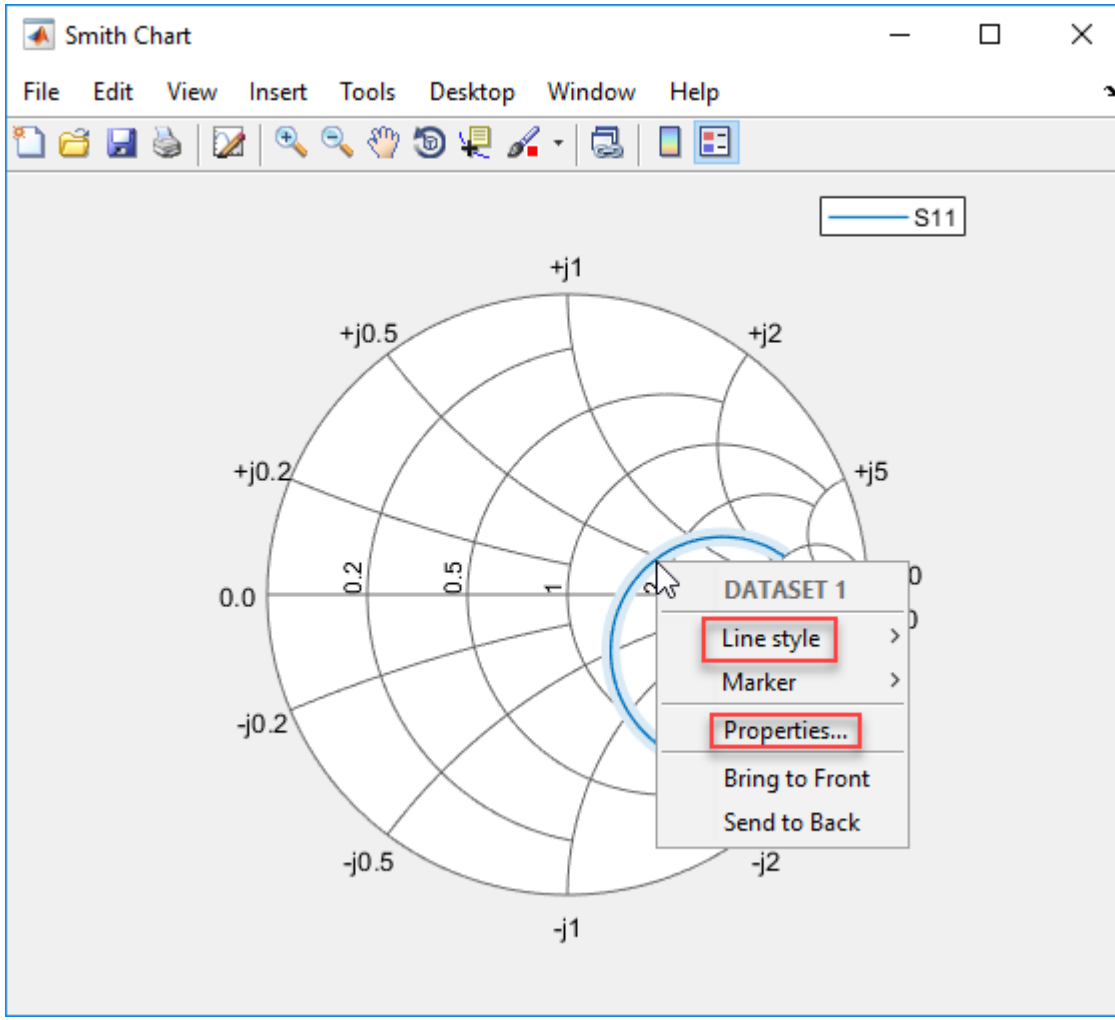
Use the Smith plot interactive menu for changing line and marker styles.

Plot the Smith plot of s-parameters of dipole d.

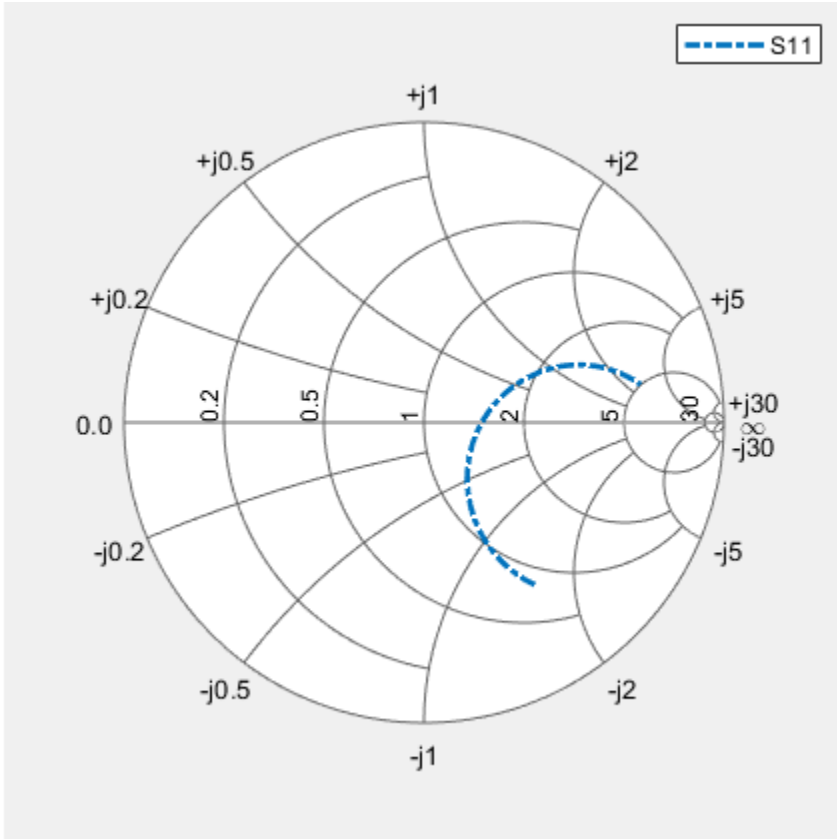
```
smithplot(s)
```



Right click on the S11 line to reveal interactive menu, `DATASET 1.` Use Line style and Properties to change the line style and width of S11 line on the Smith plot.



You can see the changes you made on the Smith plot.



Input Arguments

data — Input data

complex vector | complex matrix

Input data, specified as a complex vector or complex matrix.

For a matrix *D*, the columns of *D* are independent data sets. For *N*-by-*D* arrays, dimensions 2 and greater are independent data sets.

Data Types: `double`
Complex Number Support: Yes

`frequency` — Frequency data
real vector

Frequency data, specified as a real vector.

Data Types: `double`

`hnet` — Input objects
Antenna Toolbox network parameter object

Input objects, specified as a network parameter object.

Data Types: `double`

Output Arguments

`s` — Smith chart object handle
object

Smith chart object handle. You can use the handle to customize the plot and add measurements using MATLAB commands.

Tips

- To list all the property `Name,Value` pairs in `smithplot`, use `details(s)`. You can use the properties to extract any data from the Smith chart. For example, `s = smithplot(data,'GridType','Z')` displays the impedance data grid from the Smith chart.
- For a list of properties of `smithplot`, see SmithPlot Properties.
- You can use the `smithplot` interactive menu to change the line and marker styles.

See Also
`add` | `replace`

Introduced in R2017b

location

Location coordinates at a given distance and angle from site

Syntax

```
sitelocation = location(site)
[lat,lon] = location(site)
[___] = location(site,distance,azimuth)
```

Description

`sitelocation = location(site)` returns the site location of the antenna.

`[lat,lon] = location(site)` returns the latitude and longitude of the antenna site.

`[___] = location(site,distance,azimuth)` returns the new location achieved by moving the antenna site by the distance specified in the direction of the azimuth angle. The location is calculated by moving along the surface of the earth, using Earth ellipsoid model WGS-84.

Examples

Location of Antenna Site

Create a site 1 km north of a given site.

Create the first transmitter site.

```
tx = txsite('Name','MathWorks',...
       'Latitude',42.3001, ...
       'Longitude',-71.3504);
```

Calculate the location 1 km north of the first site.

```
[lat,lon] = location(tx,1000,90)
```

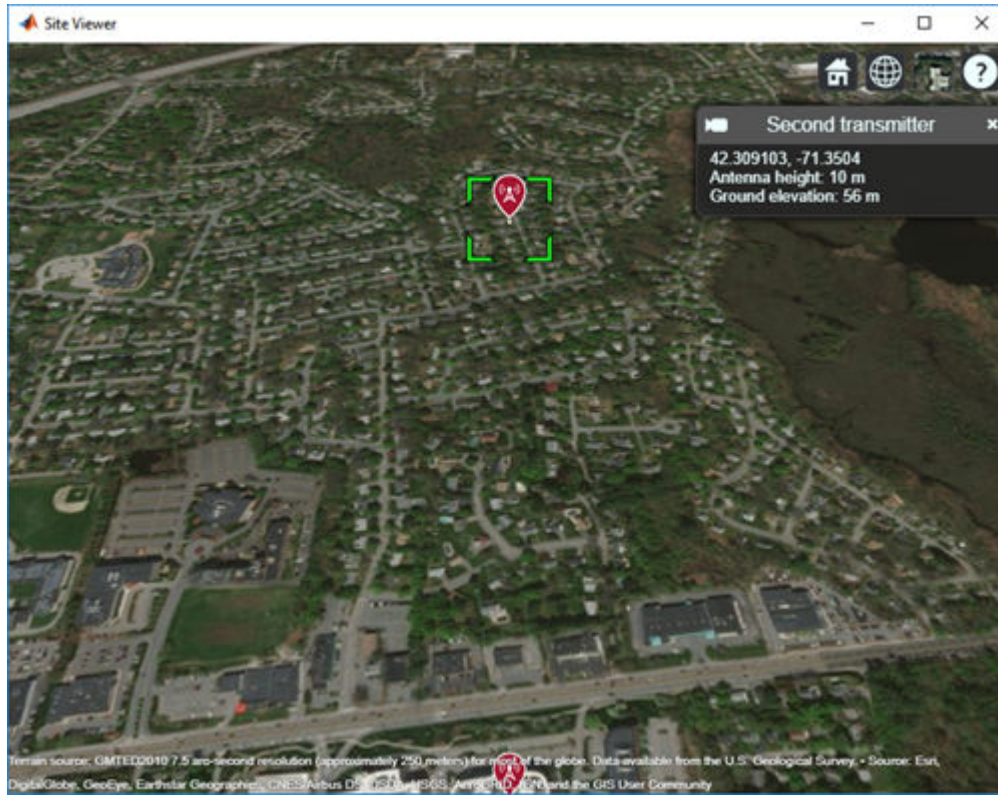
```
lat = 42.3091

lon = -71.3504
```

Create a second transmitter site at the location specified by `lat and` lon.

```
tx2 = txsite('Name','Second transmitter', ...
       'Latitude',lat, ...
       'Longitude',lon);
```

Show the two transmitter sites.

```
show([tx,tx2])
```



Input Arguments

`site` — Antenna site
scalar | array

Antenna site, specified as a scalar or an array. It is either a txsite or an rxsite object. For more information, see `txsite`, and `rxsite`

`distance` — Distance to move antenna site
scalar

Distance to move antenna site, specified as a scalar in meters.

`azimuth` — Azimuth angle
scalar

Azimuth angle, specified as a scalar in degrees. Azimuth angle is measured counterclockwise from due east.

Output Arguments

`sitelocation` — Location of antenna site
M-by-2 matrix

Location of antenna site, returned as an *M*-by-2 matrix with each element unit in degrees. *M* is the number of sites in sites. The location value includes the latitude and longitude of the antenna site.

`lat` — Latitude of one or more antenna sites
M-by-1 vector

Latitude of one or more antenna sites, returned as an *M*-by-1 vector with each element unit in degrees. *M* is the number of sites in `site`.

`lon` — Longitude of one or more antenna sites
M-by-1 matrix

Longitude of one or more antenna sites, returned as an *M*-by-1 matrix with each element unit in degrees. *M* is the number of sites in `site`.

See Also

`angle` | `distance` | `rxsite` | `txsite`

Introduced in R2018a

sinr

Display signal-to-interference-plus-noise ratio (SINR) map

Syntax

```
sinr(txs)
sinr(txs,propmodel)
sinr( ___ ,Name,Value)
```

Description

`sinr(txs)` displays the signal-to-interference-plus-noise ratio (SINR) for transmitter sites, `txs`. Each colored contour of the map defines the areas where the corresponding SINR is available to a mobile receiver. For each location, the signal source is the transmitter site in `txs` with the greatest signal strength. The remaining transmitter sites in `txs` act as interference. If `txs` is scalar, or there are no sources of interference, the resultant map displays signal-to-noise ratio (SNR).

`sinr(txs,propmodel)` displays the SINR map with the propagation model set to the value in `propmodel`.

`sinr( ___ ,Name,Value)` sets properties using one or more name-value pairs, in addition to the input arguments in previous syntaxes. For example, `sinr(txs,'MaxRange',8000)` sets the range from the site location at 8000 meters to include in the SINR map region.

Examples

SINR Map for Multiple Transmitters

Define names and location of sites in Boston.

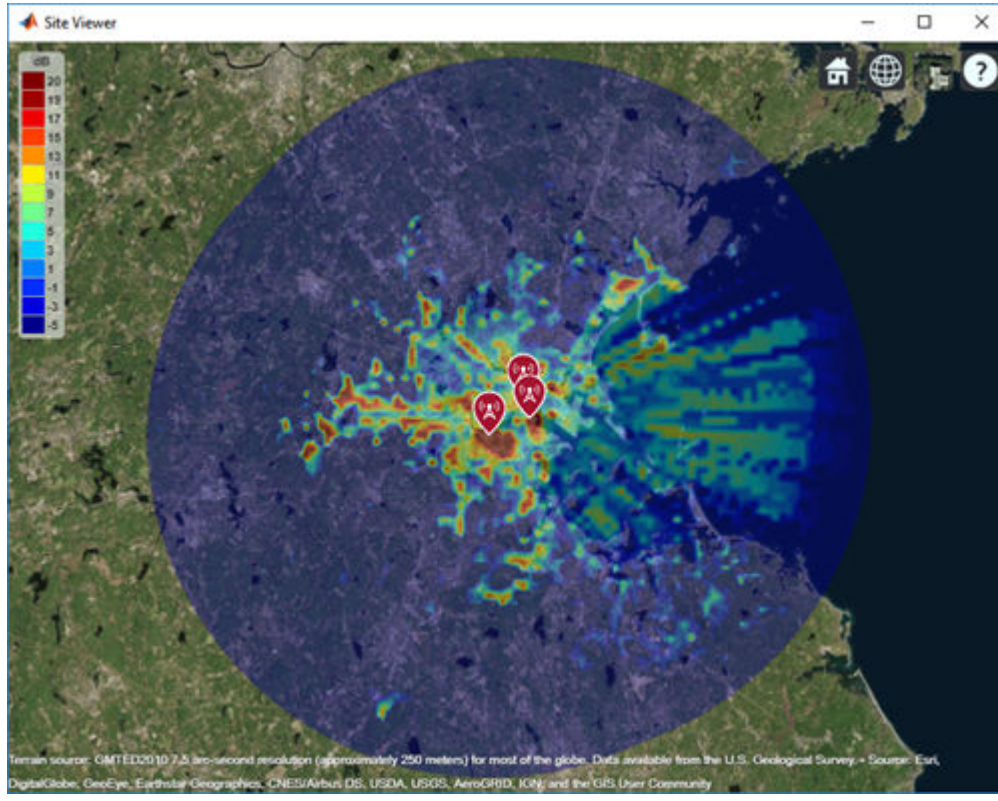
```
names = ["Fenway Park","Faneuil Hall","Bunker Hill Monument"];
lats = [42.3467,42.3598,42.3763];
lons = [-71.0972,-71.0545,-71.0611];
```

Create a transmitter site array.

```
txs = txsite('Name', names,...
       'Latitude',lats,...
       'Longitude',lons, ...
       'TransmitterFrequency',2.5e9);
```

Display the SINR map, where signal source for each location is selected as the transmitter site with the strongest signal.

```
sinr(txs)
```



Input Arguments

`txs` — Transmitter sites
`txsite` object | array of `txsite` objects

Transmitter site, specified as a `txsite` object. Use array inputs to specify multiple sites.

`propmodel` — Propagation model
character vector | string

Propagation model, specified as a character vector or string. You can use the `propagationModel` function to define this input.

You can also use the name-value pair `'PropagationModel'` to specify this parameter.

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `'MaxRange',8000`

`Values` — Values of SINR for display
`[-5:20]` (default) | numeric vector

Values of SINR for display, specified as numeric vector. Each value is displayed as a different colored, filled on the contour map. The contour colors are derived using `Colormap` and `ColorLimits`.

`SignalSource` — Signal source of interest
`'strongest'` (default) | transmitter site object

Signal source of interest, specified as `'strongest'` or as a transmitter site object. When the signal source of interest is `'strongest'`, the transmitter with the greatest signal strength is chosen as the signal source of interest for that location.

`PropagationModel` — Propagation model to use for path loss calculations
`'freespace'` (default) | `'close-in'` | `'rain'` | `'gas'` | `'fog'` | `'longley-rice'` | propagation model object

Propagation model to use for path loss calculations, specified as `'freespace'`, `'close-in'`, `'rain'`, `'gas'`, `'fog'`, `'longley-rice'`, or as an object created using the `propagationModel` function.

`MaxRange` — Maximum range of SINR map from each transmitter site
`30000` (default) | numeric scalar

Maximum range of the SINR map from each transmitter site, specified as a numeric scalar in meters.

Note When using terrain, the `MaxRange` limit is 300000 m.

Data Types: `double`

`Resolution` — Resolution of sample location grid used to compute SINR values
`'auto'` (default) | numeric scalar

Resolution of sample location grid used to compute SINR values, specified as `'auto'` or a numeric scalar in meters. If resolution is `'auto'`, `sinr` computes a value scaled to `MaxRange`. Decreasing the resolution increases the quality of the SINR map and the time required to create it.

`ReceiverNoisePower` — Total noise power at mobile receiver
`-107` (default) | scalar

Total noise power at mobile receiver, specified as a scalar in dBm. The default value assumes that the receiver bandwidth is 1 MHz and receiver noise figure is 7 dB.

$$N = -174 + 10 * \log(B) + F$$

where,

- N = Receiver noise in dBm.
- B = Receiver bandwidth in Hz
- F = Noise figure in dB

`ReceiverGain` — Mobile receiver gain
`2.1` (default) | scalar

Mobile receiver gain, specified as a scalar in dB. The receiver gain values include the antenna gain and the system loss.

`ReceiverAntennaHeight` — Mobile receiver antenna height
`1` (default) | scalar

Mobile receiver antenna height, specified as a scalar in meters.

`Colormap` — Colormap for coloring filled contours
`'jet'` (default) | *M*-by-3 array of RGB triplets

Colormap for coloring filled contours, specified as an *M*-by-3 array of RGB triplets, where *M* is the number of individual colors.

`ColorLimits` — Color limits for color maps
`[-5 20]` (default) | two-element vector

Color limits for color maps, specified as a two-element vector of the form [min max]. The color limits indicate the SINR values that map to the first and last colors in the colormap.

`ShowLegend` — Show signal strength color legend on map
`'true'` (default) | `'false'`

Show signal strength color legend on map, specified as `'true'` or `'false'`.

`Transparency` — Transparency of SINR map
`0.4` (default) | numeric scalar

Transparency of SINR map, specified as a numeric scalar in the range 0–1.

See Also

`coverage` | `propagationModel`

Introduced in R2018a

los

Plot or compute the line-of-sight (LOS) visibility between sites on a map

Syntax

```
los(site1,site2)
los(site1,site2,Name,Value)
vis = los(site1,site2,Name,Value)
```

Description

`los(site1,site2)` plots the LOS from site 1 to site 2. The plot is color coded to identify the visibility of the points along the LOS.

`los(site1,site2,Name,Value)` sets properties using one or more name-value pairs. For example, `los(site1,site2,'ObstructedColor','red')` plots the LOS using red to show blocked visibility.

`vis = los(site1,site2,Name,Value)` returns the status of the LOS visibility.

Examples

LOS from a Transmitter Site to a Receiver Site

Plot the LOS from the MathWorks Apple Hill campus to the MathWorks Lakeside campus.

Create a transmitter site with an antenna of height 30 m.

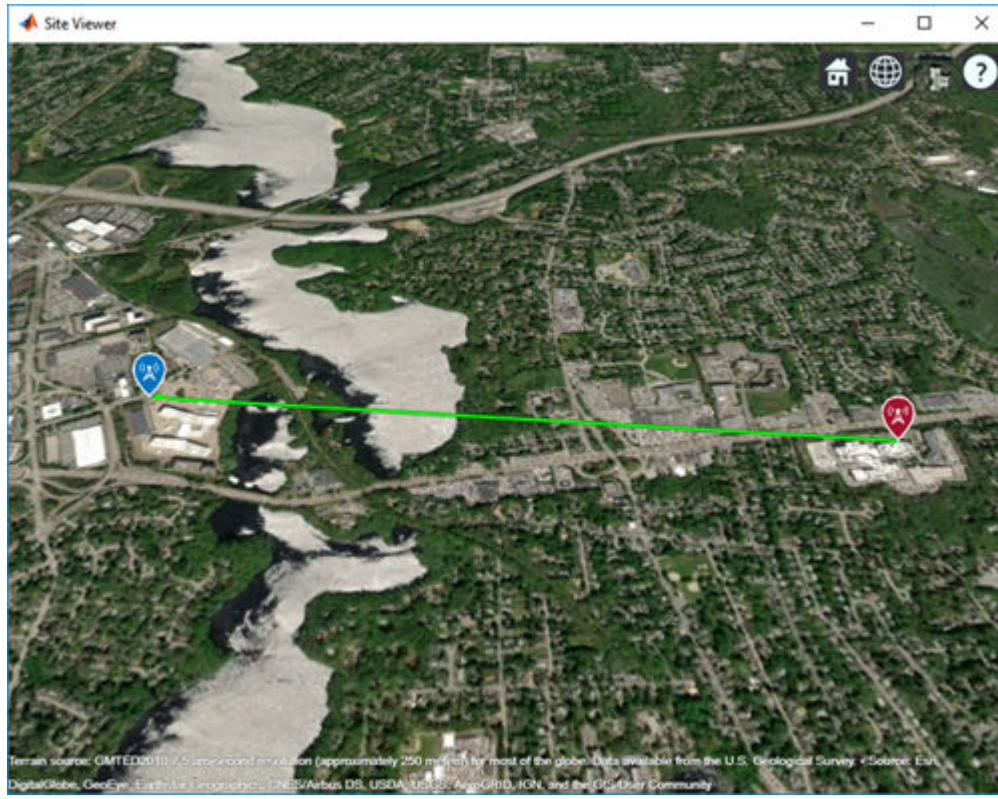
```
tx = txsite('Name','MathWorks Apple Hill',...
          'Latitude',42.3001,'Longitude',-71.3504,'AntennaHeight',30);
```

Create a receiver site with an antenna at ground level.

```
rx = rxsite('Name','MathWorks Lakeside', ...
           'Latitude',42.3021,'Longitude',-71.3764);
```

Plot the LOS between the two sites.

```
los(tx,rx);
```



LOS from a Transmitter Site to Two Receiver Sites

Create a transmitter site with an antenna of height 30 m.

```
tx = txsite('Name','MathWorks Apple Hill',...
          'Latitude',42.3001,'Longitude',-71.3504,'AntennaHeight',30);
```

Create two receiver sites with antennas at ground level.

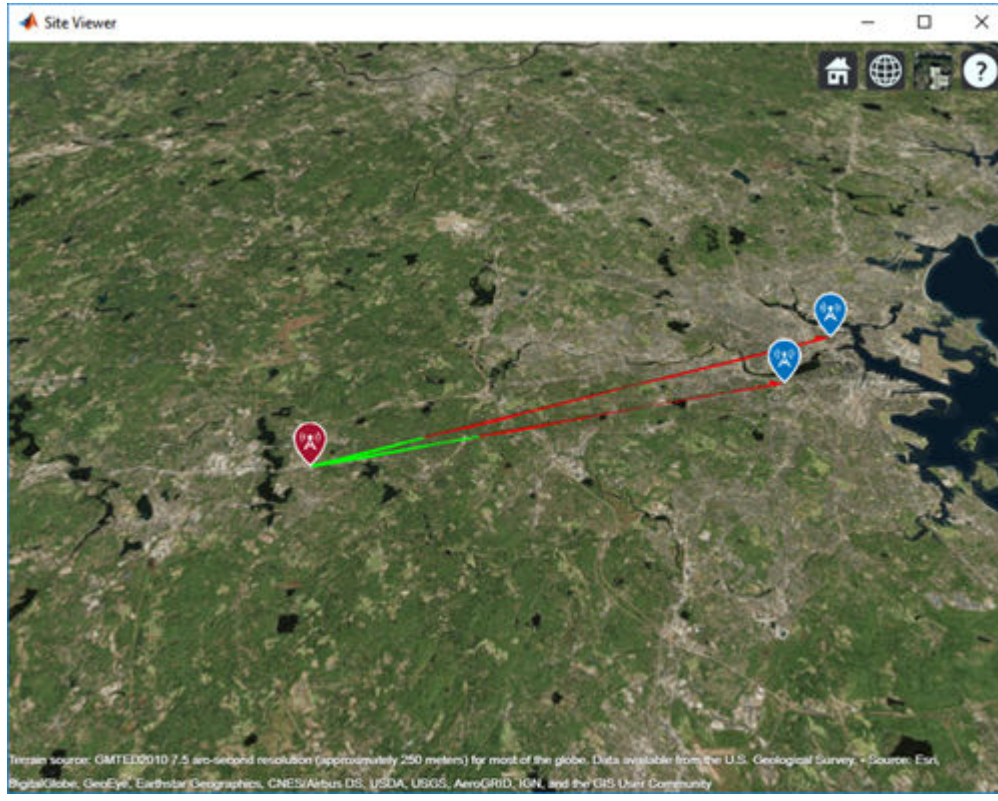
```
names = ["Fenway Park","Bunker Hill Monument"];
lats = [42.3467,42.3763];
lons = [-71.0972,-71.0611];
```

Create the receiver site array.

```
rxs = rxsite('Name', names,...
      'Latitude',lats,...
      'Longitude',lons);
```

Plot the lines of sight to the receiver sites. The red portion of the LOS represents obstructed visibility.

```
los(tx,rxs);
```



Input Arguments

`site1` — Source antenna site
`txsite` object | `rxsite` object

Source antenna site, specified as a `txsite` object or a `rxsite` object. Site 1 must be a single site object.

`site2` — Target antenna site
`txsite` object | `rxsite` object | vector of `txsite` or `rxsite` objects

Target antenna site, specified as a `txsite` object or a `rxsite` object. Site 2 can be a single site object or a vector of multiple site objects.

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `'ObstructedColor','blue'`

`VisibleColor` — Plot color for successful visibility
`'green'` (default) | RGB triplet | character vector | color name string

Plot color for successful visibility, specified as an RGB triplet, a character vector, or a color name specified as a string. For more information, see `ColorSpec (Color Specification)`.

`ObstructedColor` — Plot color for blocked visibility
`'red'` (default) | RGB triplet | character vector | color name string

Plot color for blocked visibility, specified as an RGB triplet, a character vector, or a color name specified as a string. For more information, see `ColorSpec (Color Specification)`.

`Resolution` — Sampling distance between two sites
`'auto'` (default) | numeric scalar

Resolution of sample locations used to compute line-of-sight visibility, specified as `'auto'` or a numeric scalar expressed in meters. `Resolution` defines the distance between

samples on the geodesic path between sites, using Earth ellipsoid model WGS-84. If `Resolution` is `'auto'`, the function computes a value based on the distance between the sites.

Output Arguments

`vis` — Status of LOS visibility

`'true'` | `'false'` | *n*-by 1 logical array

Status of LOS visibility, returned as `'true'` or `'false'`. If there are multiple target sites, the function returns a logical array of *n*-by-1.

See Also

`angle` | `distance` | `link`

Topics

`ColorSpec (Color Specification)`

Introduced in R2018a

phaseShift

Calculate phase shift values to scan or steer beam for arrays

Syntax

```
ps = phaseShift(array,frequency,angle)
```

Description

`ps = phaseShift(array,frequency,angle)` calculates the phase shift values of an array operating at a specified frequency to scan the beam at the given angle. The velocity of light is assumed to be that in free space.

Examples

Scan Main Beam of 3-by-3 Rectangular Array of Reflector-Backed Dipoles

Create a 3-by-3 rectangular array of reflector-backed dipoles at an operating frequency of 1.8 GHz, and scan the main beam at 30 degrees along the azimuth and 45 degrees along the elevation.

```
a = design(rectangularArray('Size',[3 3]),1.8e9,reflector);
ps = phaseShift(a,1.8e9,[30;45])
```

```
ps = 9×1

  350.5337
   54.1733
  117.8129
  240.3066
  303.9462
    7.5858
  130.0796
  193.7192
```

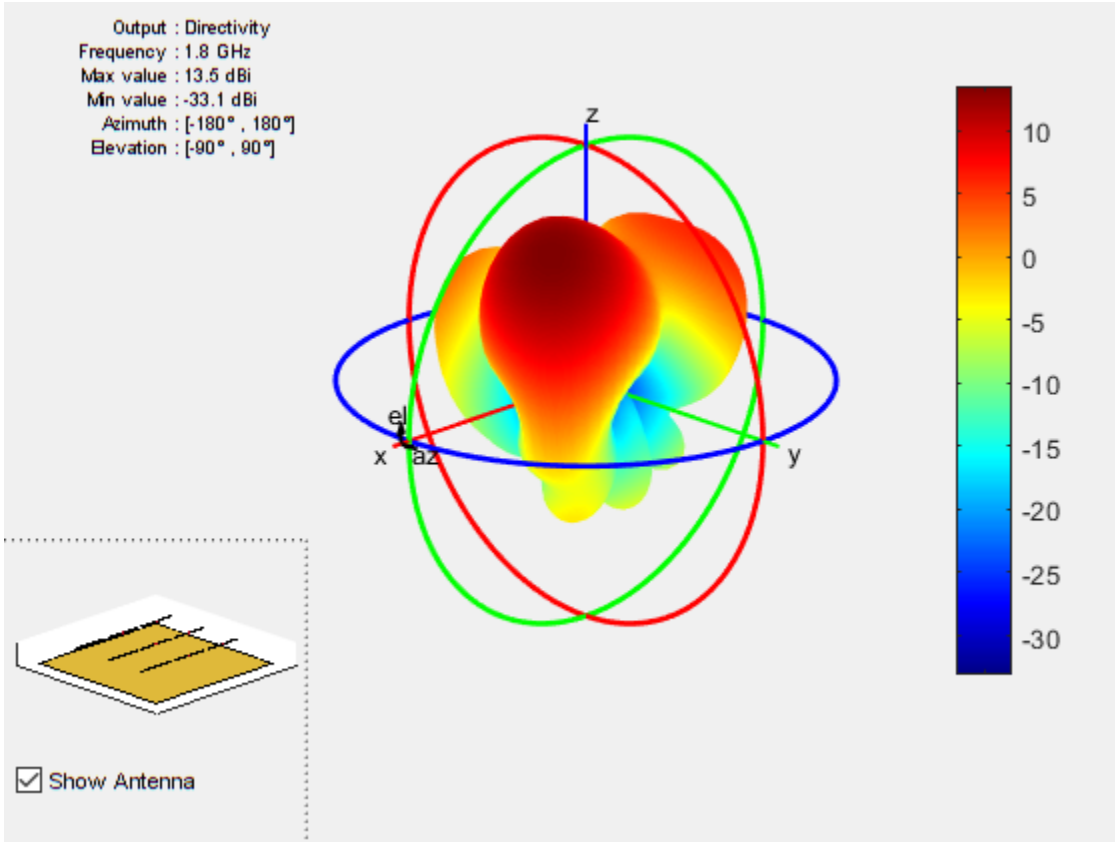
```
  257.3588
```

```
a.PhaseShift = ps
```

```
a =
  rectangularArray with properties:

          Element: [1×1 reflector]
             Size: [3 3]
        RowSpacing: 0.0833
     ColumnSpacing: 0.0833
           Lattice: 'Rectangular'
    AmplitudeTaper: 1
        PhaseShift: [9×1 double]
              Tilt: 0
         TiltAxis: [1 0 0]
```

Calculate the radiation pattern of the array.

```
pattern(a,1.8e9)
```



Input Arguments

`array` — Antenna array

array object

Antenna array from the Antenna Toolbox array library, specified as an array object.

Example: `r = rectangularArray; phaseShift (r,70e6,[60;40])`. Calculates the phase shift of the rectangular array.

`frequency` — Frequency value to calculate phase shift

scalar

Frequency value used to calculate the phase shift, specified as a scalar in Hz.

Example: `70e6`

Data Types: `double`

`angle` — Azimuth and elevation angle pair
2-element vector

Azimuth and elevation angle pair to scan the array toward, specified as a 2-element vector in degrees.

Example: `[35;40]`

Data Types: `double`

Output Arguments

`ps` — Phase shift values
1-by-*N* vector

Phase shift values, returned as a 1-by-*N* vector in degrees. Phase shift value calculation does not consider mutual coupling.

See Also

`feedCurrent` | `pattern` | `patternMultiply`

Introduced in R2018b

elevation

Ground elevation of site

Syntax

```
z = elevation(site)
```

Description

`z = elevation(site)` returns the ground elevation of an antenna site in meters. Elevation is measured as the height of the ground at the site location relative to mean sea level.

Note Enable terrain data to use this function.

Examples

Elevation at Mount Washington

Compute the elevation at Mount Washington.

```
mtwash = txsite('Name','Mt Washington','Latitude',44.2706, ...
       'Longitude',-71.3033);
z = elevation(mtwash)
```

```
z = 1.8581e+03
```

Input Arguments

`site` — Transmitter or receiver site

`txsite` or `rxsite` object | array of `txsite` or `rxsite` objects

Transmitter or receiver site, specified as a `txsite` or `rxsite` object or an array of `txsite` or `rxsite` objects.

Output Arguments

z — Ground elevation of antenna site

M-by-1 matrix

Ground elevation of antenna site, returned as an *M*-by-1 matrix with each element unit in meters. *M* is the number of sites in `site`.

See Also

`angle` | `distance` | `rxsite` | `txsite`

Introduced in R2018b

pattern

Plot antenna radiation pattern on map

Syntax

```
pattern(tx)
pattern(rx,frequency)
pattern( ___ ,Name,Value)
```

Description

`pattern(tx)` plots the 3-D antenna radiation pattern for the transmitter site, `txsite`. Signal gain value (dBi) in a particular direction determines the color of the pattern.

`pattern(rx,frequency)` plots the 3-D radiation pattern for the receiver site, `rxsite` for the specified `frequency`.

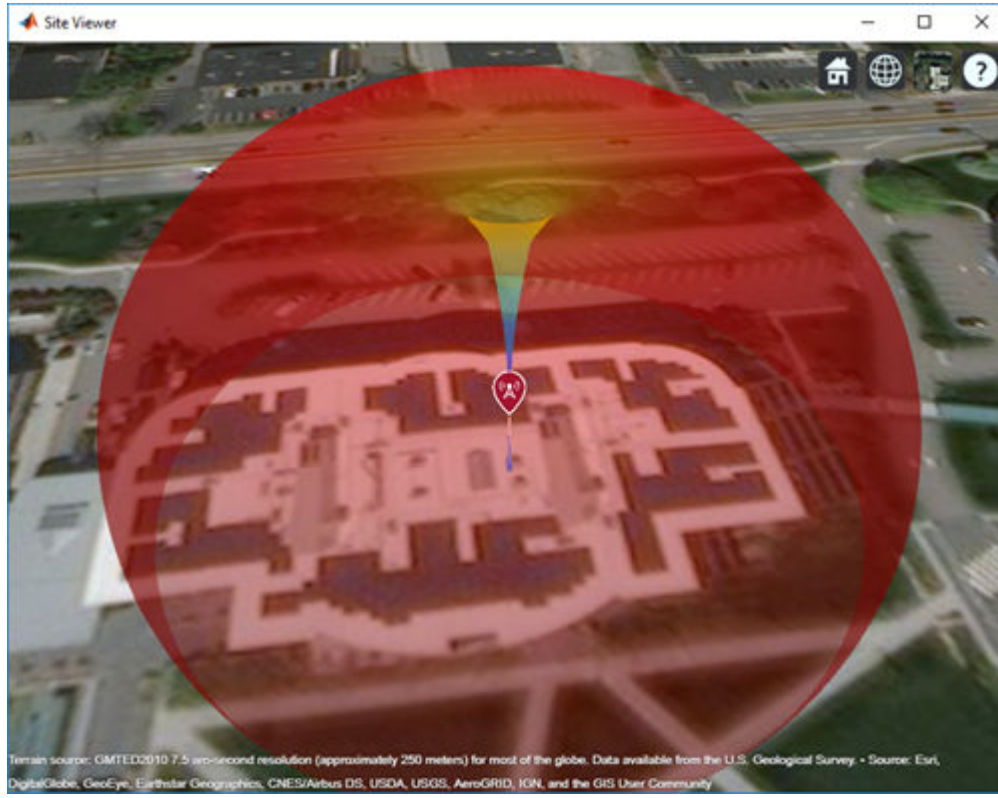
`pattern( ___ ,Name,Value)` plots the 3-D radiation pattern with additional options specified by name-value pair arguments.

Examples

Single Transmitter Site Pattern

Define and visualize the radiation pattern of a single transmitter site.

```
tx = txsite;
pattern(tx)
```



Single Receiver Site Pattern

Design a receiver site using a dipole antenna at a height of 30 meters.

```
d = dipole;
rx= rxsite('Name','Mathworks Lakeside','Latitude',42.30321,'Longitude',-71.3764,...
    'Antenna',d,'AntennaHeight',30)
```

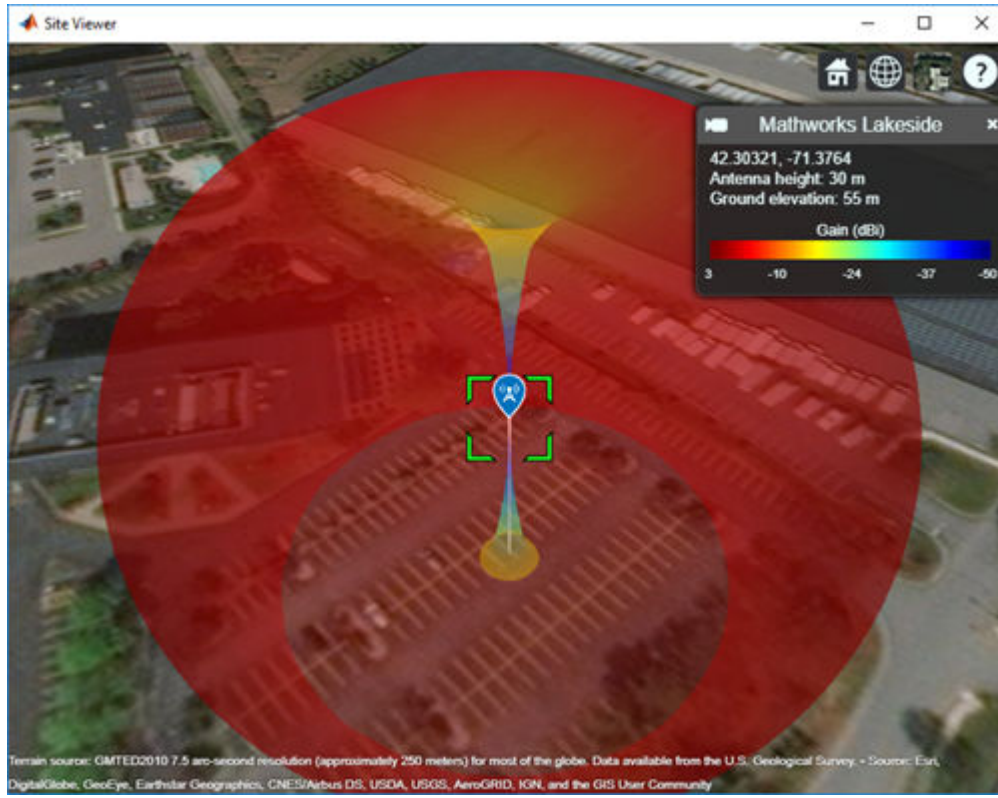
```
rx =
  rxsite with properties:

                  Name: 'Mathworks Lakeside'
              Latitude: 42.3032
```

```
              Longitude: -71.3764
                Antenna: [1×1 dipole]
           AntennaAngle: 0
          AntennaHeight: 30
             SystemLoss: 0
    ReceiverSensitivity: -100
```

```
show(rx)
```

Visualize the pattern of the receiver site at 75 MHz.

```
pattern(rx,75e6)
```



Pattern for Directional Transmitter and Receiver

Create directional antenna.

```
yagiAntenna = design(yagiUda,4.5e9);
yagiAntenna.Tilt = 90;
yagiAntenna.TiltAxis = 'y';
```

Create transmitter and receiver sites at a frequency of 4.5 GHz. Use the Yagi antenna as the transmitter antenna. Design a dipole at 4.5 GHz and use this as the receiver antenna.

```
fq = 4.5e9;
tx = txsite('Name','MathWorks','Latitude',42.3001,'Longitude', -71.3503, ...
```

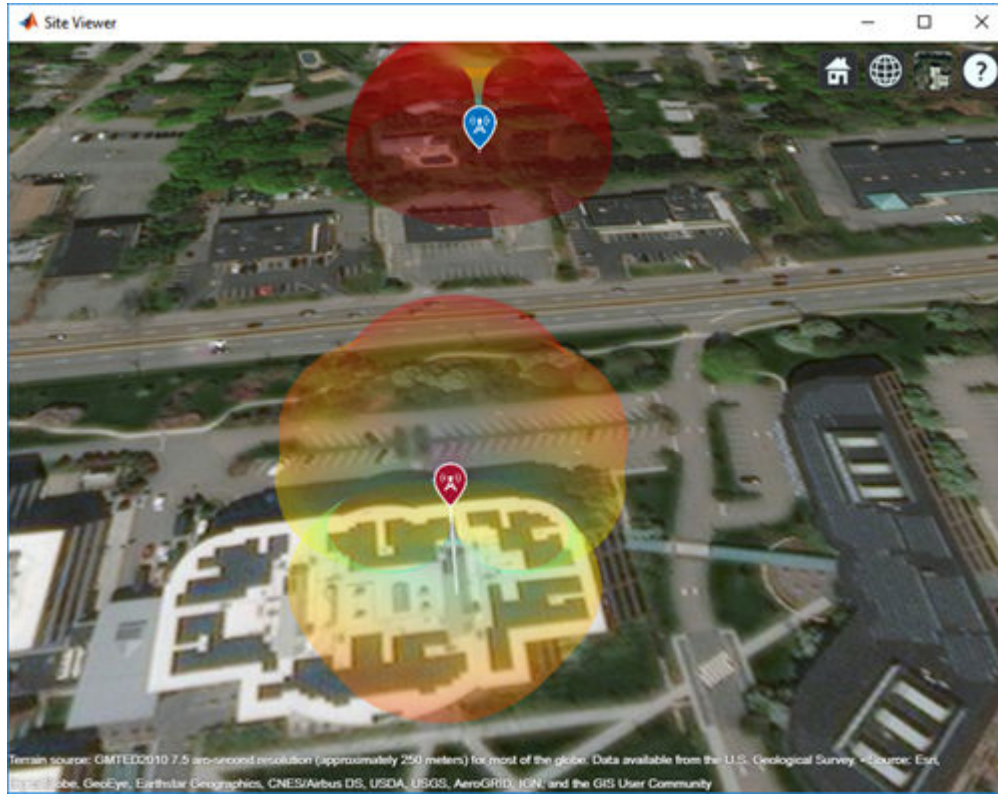
```
      'Antenna', yagiAntenna,'AntennaAngle', 90,'AntennaHeight', 30, ...
      'TransmitterFrequency', fq,'TransmitterPower', 10);
rx = rxsite('Antenna',design(dipole, fq));
```

Position the receiver 200 meters from the transmitter.

```
[lat,lon] = location(tx,200,90);
rx.Latitude = lat;
rx.Longitude = lon;
```

Display both transmitter and receiver patterns.

```
pattern(tx,'Transparency',0.2);
pattern(rx, fq);
```



Input Arguments

`tx` — Transmitter site
`txsite` object

Transmitter site, specified as a `txsite` object.

`rx` — Receiver site
`rxsite` object

Receiver site, specified as a `rxsite` object.

`frequency` — Frequency to calculate radiation pattern
positive scalar

Frequency to calculate radiation pattern, specified as a positive scalar.

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `'Size',2`

`Size` — Size of pattern plot
`50` (default) | numerical scalar

Size of the pattern plot, specified as a numerical scalar in meters. This parameter represents the distance between the antenna position and the point on the plot with the highest gain.

Data Types: `double`

`Transparency` — Transparency of pattern plot
`0.4` (default) | real number in the range of [0,1]

Transparency of the pattern plot, specified as a real number in the range of [0,1], where `0` is completely transparent and `1` is completely opaque.

Data Types: `double`

`Colormap` — Colormap for coloring of pattern plot
`'jet(256)'` (default) | predefined colormap name | *M*-by-3 array of RGB triplets

Colormap for coloring of the pattern plot, specified as a predefined colormap name or an *M*-by-3 array of RGB (red, blue, green) triplets that define *M* individual colors.

Data Types: `double`

See Also

`coverage`

Introduced in R2018b

Properties — Alphabetical List

PolarPattern Properties

Control appearance and behavior of polar plot

Description

Polar pattern properties control the appearance and behavior of the polar pattern object. By changing property values, you can modify certain aspects of the polar plot. To change the default properties use:

```
p = polarpattern(____,Name,Value)
```

To view all the properties of the polar pattern object use:

```
details(p)
```

You can also interact with the polar plot to change the properties. For more information, see “Interact with Polar Plot”.

Properties

Antenna Metrics

`'AntennaMetrics'` — Show antenna metric
`0` (default) | `1`

Show antenna metrics, specified as a comma-separated pair consisting of `'AntennaMetrics'` and `0` or `1`. Antenna metric displays main, back, and side lobes of antenna/array pattern passed as input.

Data Types: `logical`

`'Peaks'` — Maximum number of peaks to compute for each data set
positive integer | vector of integers

Maximum number of peaks to compute for each data set, specified as a comma-separated pair consisting of `'Peaks'` and a positive scalar or vector of integers.

Data Types: `double`

Angle Properties

`'AngleAtTop'` — Angle at top of polar plot
`90` (default) | scalar in degrees

Angle at the top of the polar plot, specified as a comma-separated pair consisting of `'AngleAtTop'` and a scalar in degrees.

Data Types: `double`

`'AngleLim'` — Visible polar angle span
`[0 360]` (default) | 1-by-2 vector of real values

Visible polar angle span, specified as a comma-separated pair consisting of `'AngleLim'` and a 1-by-2 vector of real values.

Data Types: `double`

`'AngleLimVisible'` — Show interactive angle limit cursors
`0` (default) | `1`

Show interactive angle limit cursors, specified as a comma-separated pair consisting of `'AngleLimVisible'` and `0` or `1`.

Data Types: `logical`

`'AngleDirection'` — Direction of increasing angle
`'ccw'` (default) | `'cw'`

Direction of increasing angle, specified as a comma-separated pair consisting of `'AngleDirection'` and `'ccw'` (counterclockwise) or `'cw'` (clockwise).

Data Types: `char`

`'AngleResolution'` — Number of degrees between radial lines
`15` (default) | scalar in degrees

Number of degrees between radial lines depicting angles in the polar plot, specified as a comma-separated pair consisting of `'AngleResolution'` and a scalar in degrees.

Data Types: `double`

`'AngleTickLabelRotation'` — Rotate angle tick labels
`0` (default) | `1`

Rotate angle tick labels, specified as a comma-separated pair consisting of `'AngleTickLabelRotation'` and `0` or `1`.

Data Types: `logical`

`'AngleTickLabelVisible'` — Show angle tick labels
`1` (default) | `0`

Show angle tick labels, specified as a comma-separated pair consisting of `'AngleTickLabelVisible'` and `0` or `1`.

Data Types: `logical`

`'AngleTickLabelFormat'` — Format for angle tick labels
`360` (default) | `180`

Format for angle tick labels, specified as a comma-separated pair consisting of `'AngleTickLabelFormat'` and `360` degrees or `180` degrees.

Data Types: `double`

`'AngleFontSizeMultiplier'` — Scale factor of angle tick font
`1` (default) | numeric value greater than zero

Scale factor of angle tick font, specified as a comma-separated pair consisting of `'AngleFontSizeMultiplier'` and a numeric value greater than zero.

Data Types: `double`

`'Span'` — Show angle span measurement
`0` (default) | `1`

Show angle span measurement, specified as a comma-separated pair consisting of `'Span'` and `0` or `1`.

Data Types: `logical`

`'ZeroAngleLine'` — Highlight radial line at zero degrees
`0` (default) | `1`

Highlight radial line at zero degrees, specified as a comma-separated pair consisting of `'ZeroAngleLine'` and `0` or `1`.

Data Types: `logical`

`'DisconnectAngleGaps'` — Show gaps in line plots with nonuniform angle spacing

`1` (default) | `0`

Show gaps in line plots with nonuniform angle spacing, specified as a comma-separated pair consisting of `'DisconnectAngleGaps'` and `0` or `1`.

Data Types: `logical`

Magnitude Properties

`'MagnitudeAxisAngle'` — Angle of magnitude tick label radial line

`75` (default) | real scalar in degrees

Angle of magnitude tick label radial line, specified as a comma-separated pair consisting of `'MagnitudeAxisAngle'` and real scalar in degrees.

Data Types: `double`

`'MagnitudeTick'` — Magnitude ticks

`[0 0.2 0.4 0.6 0.8]` (default) | 1-by-N vector

Magnitude ticks, specified as a comma-separated pair consisting of `'MagnitudeTick'` and a 1-by-N vector, where N is the number of magnitude ticks.

Data Types: `double`

`'MagnitudeTickLabelVisible'` — Show magnitude tick labels

`1` (default) | `0`

Show magnitude tick labels, specified as a comma-separated pair consisting of `'MagnitudeTickLabelVisible'` and `0` or `1`.

Data Types: `logical`

`'MagnitudeLim'` — Minimum and maximum magnitude limits

`[0 1]` (default) | two-element vector of real values

Minimum and maximum magnitude limits, specified as a comma-separated pair consisting of `'MagnitudeLim'` and a two-element vector of real values.

Data Types: `double`

`'MagnitudeLimMode'` — Determine magnitude dynamic range

`'auto'` (default) | `'manual'`

Determine magnitude dynamic range, specified as a comma-separated pair consisting of `'MagnitudeLimMode'` and `'auto'` or `'manual'`.

Data Types: `char`

`'MagnitudeAxisAngleMode'` — Determine angle for magnitude tick labels
`'auto'` (default) | `'manual'`

Determine angle for magnitude tick labels, specified as a comma-separated pair consisting of `'MagnitudeAxisAngleMode'` and `'auto'` or `'manual'`.

Data Types: `char`

`'MagnitudeTickMode'` — Determine magnitude tick locations
`'auto'` (default) | `'manual'`

Determine magnitude tick locations, specified as a comma-separated pair consisting of `'MagnitudeTickMode'` and `'auto'` or `'manual'`.

Data Types: `char`

`'MagnitudeUnits'` — Magnitude units
`'dB'` | `'dBLoss'`

Magnitude units, specified as a comma-separated pair consisting of `'MagnitudeUnits'` and `'db'` or `'dBLoss'`.

Data Types: `char`

`'MagnitudeFontSizeMultiplier'` — Scale factor of magnitude tick font
`0.9000` (default) | numeric value greater than zero

Scale factor of magnitude tick font, specified as a comma-separated pair consisting of `'MagnitudeFontSizeMultiplier'` and a numeric value greater than zero.

Data Types: `double`

Miscellaneous Properties

`'NormalizeData'` — Normalize each data trace to maximum value
`0` (default) | `1`

Normalize each data trace to maximum value, specified as a comma-separated pair consisting of `'NormalizeData'` and `0` or `1`.

Data Types: `logical`

`'ConnectEndpoints'` — Connect first and last angles
`0` (default) | `1`

Connect first and last angles, specified as a comma-separated pair consisting of `'ConnectEndpoints'` and `0` or `1`.

Data Types: `logical`

`'Style'` — Style of polar plot display
`'line'` (default) | `'filled'`

Style of polar plot display, specified as a comma-separated pair consisting of `'Style'` and `'line'` or `'filled'`.

Data Types: `char`

`'TemporaryCursor'` — Create temporary cursor
`0` (default) | `1`

Create a temporary cursor, specified as a comma-separated pair consisting of `'TemporaryCursor'` and `0` or `1`.

Data Types: `logical`

`'ToolTips'` — Show tool tips
`1` (default) | `0`

Show tool tips when you hover over a polar plot element, specified as a comma-separated pair consisting of `'ToolTips'` and `0` or `1`.

Data Types: `logical`

`'ClipData'` — Clip data to outer circle
`0` (default) | `1`

Clip data to outer circle, specified as a comma-separated pair consisting of `'ClipData'` and `0` or `1`.

Data Types: `logical`

`'NextPlot'` — Directive on how to add next plot
`'replace'` (default) | `'new'` | `'add'`

Directive on how to add next plot, specified as a comma-separated pair consisting of `'NextPlot'` and one of the values in the table:

Property Value	Effect
`'new'`	Creates a figure and uses it as the current figure.
`'add'`	Adds new graphics objects without clearing or resetting the current figure.
`'replace'`	Removes all axes objects and resets figure properties to their defaults before adding new graphics objects.

Legend and Title Properties

`'LegendLabels'` — Data tables for legend annotation
character vector | cell array of character vectors

Data tables for legend annotation, specified as a comma-separated pair consisting of `'LegendLabels'` and a character vector or cell array of character vectors. Ⓐ denotes the active line for interactive operation.

Data Types: `char`

`'LegendVisible'` — Show legend label
`0` (default) | `1`

Show legend label, specified as a comma-separated pair consisting of `'LegendVisible'` and `0` or `1`.

Data Types: `logical`

`'TitleTop'` — Title to display above the polar plot
character vector

Title to display above the polar plot, specified as a comma-separated pair consisting of `'TitleTop'` and a character vector.

Data Types: `char`

`'TitleBottom'` — Title to display below the polar plot
character vector

Title to display below the polar plot, specified as a comma-separated pair consisting of `'TitleBottom'` and a character vector.

Data Types: `char`

`'TitleTopOffset'` — Offset between top title and angle ticks
`0.1500` (default) | scalar

Offset between top title and angle ticks, specified as a comma-separated pair consisting of `'TitleTopOffset'` and a scalar. The value must be in the range `[-0.5,0.5]`.

Data Types: `double`

`'TitleBottomOffset'` — Offset between bottom title and angle ticks
`0.1500` (default) | scalar

Offset between bottom title and angle ticks, specified as a comma-separated pair consisting of `'TitleBottomOffset'` and a scalar. The value must be in the range `[-0.5,0.5]`.

Data Types: `double`

`'TitleTopFontSizeMultiplier'` — Scale factor of top title font
`1.1000` (default) | numeric value greater than zero

Scale factor of top title font, specified as a comma-separated pair consisting of `'TitleTopFontSizeMultiplier'` and a numeric value greater than zero.

Data Types: `double`

`'TitleBottomFontSizeMultiplier'` — Scale factor of bottom title font
`0.9000` (default) | numeric value greater than zero

Scale factor of bottom title font, specified as a comma-separated pair consisting of `'TitleBottomFontSizeMultiplier'` and a numeric value greater than zero.

Data Types: `double`

`'TitleTopFontWeight'` — Thickness of top title font
`'bold'` (default) | `'normal'`

Thickness of top title font, specified as a comma-separated pair consisting of `'TitleTopFontWeight'` and `'bold'` or `'normal`.

Data Types: `char`

'TitleBottomFontWeight' — Thickness of bottom title font

'normal' (default) | 'bold'

Thickness of bottom title font, specified as a comma-separated pair consisting of 'TitleBottomFontWeight' and 'bold' or 'normal'.

Data Types: char

'TitleTopTextInterpreter' — Interpretation of top title characters

'none' (default) | 'tex' | 'latex'

Interpretation of top title characters, specified as a comma-separated pair consisting of 'TitleTopTextInterpreter' and:

- 'tex' — Interpret using a subset of TeX markup
- 'latex' — Interpret using LaTeX markup
- 'none' — Display literal characters

TeX Markup

By default, MATLAB supports a subset of TeX markup. Use TeX markup to add superscripts and subscripts, modify the text type and color, and include special characters in the text.

This table lists the supported modifiers when the TickLabelInterpreter property is set to 'tex', which is the default value. Modifiers remain in effect until the end of the text, except for superscripts and subscripts which only modify the next character or the text within the curly braces {}.

Modifier	Description	Example
^{ }	Superscript	'text^{superscript}'
_{ }	Subscript	'text_{subscript}'
\bf	Bold font	'\bf text'
\it	Italic font	'\it text'
\sl	Oblique font (rarely available)	'\sl text'
\rm	Normal font	'\rm text'

Modifier	Description	Example
`\fontname{specifier}`	Set `specifier` as the name of a font family to change the font style. You can use this with other modifiers.	`'\fontname{Courier} text'`
`\fontsize{specifier}`	Set `specifier` as a scalar numeric value to change the font size.	`'\fontsize{15} text'`
`\color{specifier}`	Set `specifer` as one of these colors: `red`, `green`, `yellow`, `magenta`, `blue`, `black`, `white`, `gray`, `darkGreen`, `orange`, or `lightBlue`.	`'\color{magenta} text'`
`\color[rgb] {specifier}`	Set `specifier` as a three-element RGB triplet to change the font color.	`'\color[rgb] {0,0.5,0.5} text'`

LaTeX Markup

To use LaTeX markup, set the `TickLabelInterpreter` property to `'latex'`. The displayed text uses the default LaTeX font style. The `FontName`, `FontWeight`, and `FontAngle` properties do not have an effect. To change the font style, use LaTeX markup within the text.

The maximum size of the text that you can use with the LaTeX interpreter is 1200 characters. For multiline text, the maximum size reduces by about 10 characters per line.

Data Types: `char`

`'TitleBottomTextInterpreter'` — Interpretation of bottom title characters

`'none'` (default) | `'tex'` | `'latex'`

Interpretation of bottom title characters, specified as a comma-separated pair consisting of `'TitleBottomTextInterpreter'` and:

- `'tex'` — Interpret using a subset of TeX markup
- `'latex'` — Interpret using LaTeX markup
- `'none'` — Display literal characters

TeX Markup

By default, MATLAB supports a subset of TeX markup. Use TeX markup to add superscripts and subscripts, modify the text type and color, and include special characters in the text.

This table lists the supported modifiers when the `TickLabelInterpreter` property is set to `'tex'`, which is the default value. Modifiers remain in effect until the end of the text, except for superscripts and subscripts which only modify the next character or the text within the curly braces `{}`.

Modifier	Description	Example
`^{ }`	Superscript	`'text^{superscript}'`
`_{ }`	Subscript	`'text_{subscript}'`
`\bf`	Bold font	`'\bf text'`
`\it`	Italic font	`'\it text'`
`\sl`	Oblique font (rarely available)	`'\sl text'`
`\rm`	Normal font	`'\rm text'`
`\fontname{specifier}`	Set `specifier` as the name of a font family to change the font style. You can use this with other modifiers.	`'\fontname{Courier} text'`
`\fontsize{specifier}`	Set `specifier` as a scalar numeric value to change the font size.	`'\fontsize{15} text'`
`\color{specifier}`	Set `specifer` as one of these colors: `red`, `green`, `yellow`, `magenta`, `blue`, `black`, `white`, `gray`, `darkGreen`, `orange`, or `lightBlue`.	`'\color{magenta} text'`
`\color[rgb] {specifier}`	Set `specifier` as a three-element RGB triplet to change the font color.	`'\color[rgb] {0,0.5,0.5} text'`

LaTeX Markup

To use LaTeX markup, set the `TickLabelInterpreter` property to `'latex'`. The displayed text uses the default LaTeX font style. The `FontName`, `FontWeight`, and `FontAngle` properties do not have an effect. To change the font style, use LaTeX markup within the text.

The maximum size of the text that you can use with the LaTeX interpreter is 1200 characters. For multiline text, the maximum size reduces by about 10 characters per line.

Data Types: `char`

Grid Properties

`'GridOverData'` — Draw grid over data plots
`0` (default) | `1`

Draw grid over data plots, specified as a comma-separated pair consisting of `'GridOverData'` and `0` or `1`.

Data Types: `logical`

`'DrawGridToOrigin'` — Draw radial lines within innermost circle
`0` (default) | `1`

Draw radial lines within innermost circle of the polar plot, specified as a comma-separated pair consisting of `'DrawGridToOrigin'` and `0` or `1`.

Data Types: `logical`

`'GridAutoRefinement'` — Increase angle resolution
`0` (default) | `1`

Increase angle resolution in the polar plot, specified as a comma-separated pair consisting of `'GridAutoRefinement'` and `0` or `1`. This property increases angle resolution by doubling the number of radial lines outside each magnitude.

Data Types: `logical`

`'GridWidth'` — Width of grid lines
`0.5000` (default) | positive scalar

Width of grid lines, specified as a comma-separated pair consisting of `'GridWidth'` and a positive scalar.

Data Types: `double`

`'GridVisible'` — Show grid lines

`1` (default) | `0`

Show grid lines, including magnitude circles and angle radii, specified as a comma-separated pair consisting of `'GridVisible'` and `0` or `1`.

Data Types: `logical`

`'GridForeGroundColor'` — Color of foreground grid lines

`[0.8000 0.8000 0.8000]` (default) | `'none'` | character vector of color names

Color of foreground grid lines, specified as a comma-separated pair consisting of `'GridForeGroundColor'` and an RGB triplet, character vector of color names, or `'none'`.

An RGB triplet is a three-element row vector whose elements specify the intensities of the red, green, and blue components of the color. The intensities must be in the range `[0,1]`; for example, `[0.4 0.6 0.7]`. Alternatively, you can specify some common colors by name. This table lists the long and short color name options and the equivalent RGB triplet values.

Option	Description	Equivalent RGB Triplet
`'red'` or `'r'`	Red	`[1 0 0]`
`'green'` or `'g'`	Green	`[0 1 0]`
`'blue'` or `'b'`	Blue	`[0 0 1]`
`'yellow'` or `'y'`	Yellow	`[1 1 0]`
`'magenta'` or `'m'`	Magenta	`[1 0 1]`
`'cyan'` or `'c'`	Cyan	`[0 1 1]`
`'white'` or `'w'`	White	`[1 1 1]`
`'black'` or `'k'`	Black	`[0 0 0]`

Data Types: `double` | `char`

`'GridBackGroundColor'` — Color of background grid lines

`'w'` (default) | character vector of color names | `'none'`

Color of background grid lines, specified as a comma-separated pair consisting of `'GridBackGroundColor'` and an RGB triplet, character vector of color names, or `'none'`.

An RGB triplet is a three-element row vector whose elements specify the intensities of the red, green, and blue components of the color. The intensities must be in the range `[0,1]`; for example, `[0.4 0.6 0.7]`. Alternatively, you can specify some common colors by name. This table lists the long and short color name options and the equivalent RGB triplet values.

Option	Description	Equivalent RGB Triplet
`'red'` or `'r'`	Red	`[1 0 0]`
`'green'` or `'g'`	Green	`[0 1 0]`
`'blue'` or `'b'`	Blue	`[0 0 1]`
`'yellow'` or `'y'`	Yellow	`[1 1 0]`
`'magenta'` or `'m'`	Magenta	`[1 0 1]`
`'cyan'` or `'c'`	Cyan	`[0 1 1]`
`'white'` or `'w'`	White	`[1 1 1]`
`'black'` or `'k'`	Black	`[0 0 0]`

Data Types: `double` | `char`

Marker, Color, Line, and Font Properties

`'Marker'` — Marker symbol

`'none'` (default) | character vector of symbols

Marker symbol, specified as a comma-separated pair consisting of `'Marker'` and either `'none'` or one of the symbols in this table. By default, a line does not have markers. Add markers at selected points along the line by specifying a marker.

Value	Description
`'o'`	Circle
`'+'`	Plus sign
`'*'`	Asterisk
`'.'`	Point

Value	Description
`'x'`	Cross
`'square'` or `'s'`	Square
`'diamond'` or `'d'`	Diamond
`'^'`	Upward-pointing triangle
`'v'`	Downward-pointing triangle
`'>'`	Right-pointing triangle
`'<'`	Left-pointing triangle
`'pentagram'` or `'p'`	Five-pointed star (pentagram)
`'hexagram'` or `'h'`	Six-pointed star (hexagram)
`'none'`	No markers

`'MarkerSize'` — Marker size

6 (default) | positive value

Marker size, specified as a comma-separated pair consisting of `'MarkerSize'` and a positive value in point units.

Data Types: `double`

`'ColorOrder'` — Colors to use for multiline plots

seven predefined colors (default) | three-column matrix of RGB triplets

Colors to use for multiline plots, specified as a comma-separated pair consisting of `'ColorOrder'` and a three-column matrix of RGB triplets. Each row of the matrix defines one color in the color order.

Data Types: `double`

`'ColorOrderIndex'` — Next color to use in color order

1 (default) | positive integer

Next color to use in color order, specified as a comma-separated pair consisting of `'ColorOrderIndex'` and a positive integer. New plots added to the axes use colors based on the current value of the color order index.

Data Types: `double`

`'EdgeColor'` — Color of data lines

`'k'` (default) | RGB triplet vector

Color of data lines, specified as a comma-separated pair consisting of `'EdgeColor'` and a character vector of color names or RGB triplet vector.

An RGB triplet is a three-element row vector whose elements specify the intensities of the red, green, and blue components of the color. The intensities must be in the range `[0,1]`; for example, `[0.4 0.6 0.7]`. Alternatively, you can specify some common colors by name. This table lists the long and short color name options and the equivalent RGB triplet values.

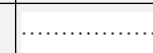
Option	Description	Equivalent RGB Triplet
`'red'` or `'r'`	Red	`[1 0 0]`
`'green'` or `'g'`	Green	`[0 1 0]`
`'blue'` or `'b'`	Blue	`[0 0 1]`
`'yellow'` or `'y'`	Yellow	`[1 1 0]`
`'magenta'` or `'m'`	Magenta	`[1 0 1]`
`'cyan'` or `'c'`	Cyan	`[0 1 1]`
`'white'` or `'w'`	White	`[1 1 1]`
`'black'` or `'k'`	Black	`[0 0 0]`

Data Types: `double` | `char`

`'LineStyle'` — Line style of the plot

`'-'` (default) | `'--'` | `':'` | `'-.'` | `'none'`

Line style of the plot, specified as a comma-separated pair consisting of `'LineStyle'` and one of the symbols in the table:

Symbol	Line Style	Resulting Line
`'-'`	Solid line	
`'--'`	Dashed line	
`':'`	Dotted line	

Symbol	Line Style	Resulting Line
`'-.'`	Dash-dotted line	
`'none'`	No line	No line

`'LineWidth'` — Line width of plot
`1` (default) | positive scalar | positive vector

Line width of the plot, specified as a comma-separated pair consisting of `'LineWidth'` and a positive scalar or vector.

`'FontSize'` — Font size of text in plot
`10` (default) | positive scalar

Font size of text in the plot, specified as a comma-separated pair consisting of `'FontSize'` and a positive scalar.

`'FontSizeAutoMode'` — Set font size
`'auto'` (default) | `'manual'`

Set font size, specified as a comma-separated pair consisting of `'FontSizeAutoMode'` and `'auto'` or `'manual'`.

Data Types: `char`

See Also

“Interact with Polar Plot”