Antenna Toolbox™

Reference

MATLAB®

R2016a

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)

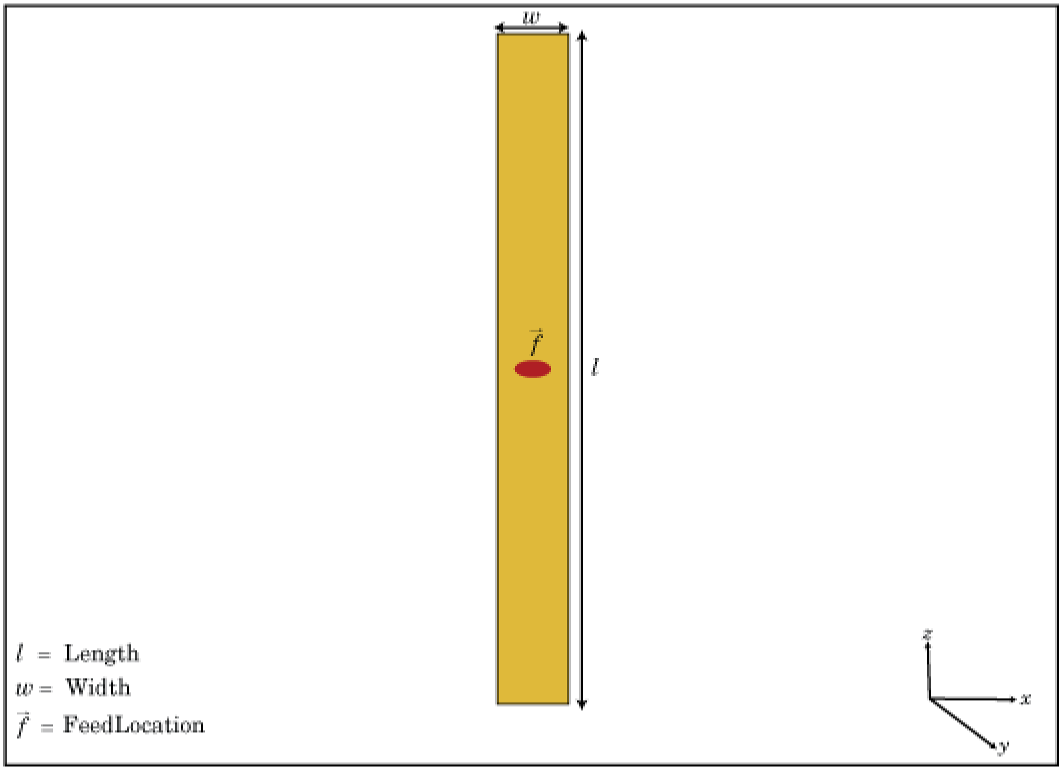
Contents

Antenna Classes — Alphabetical List

dipole class

Create strip dipole antenna

Description



The `dipole` class creates 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.

Construction

`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 in meters

Dipole length, specified as the comma-separated pair consisting of `'Length'` and 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 in meters

Dipole width, specified as the comma-separated pair consisting of `'Width'` and 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 in meters

Signed distance from center of dipole, specified as the comma-separated pair consisting of `'FeedOffset'` and a scalar in meters. The feed location is on Y-Z plane.

Example: `'FeedOffset'`,3

Data Types: `double`

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

Tilt angle of antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or vector in degrees.

Example: `'Tilt',90`

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

Data Types: `double`

`'TiltAxis'` — Tilt axis of antenna
[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,“Rotate Antenna and Arrays”

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

Example: `'TiltAxis'`,`'Z'`

Data Types: `double`

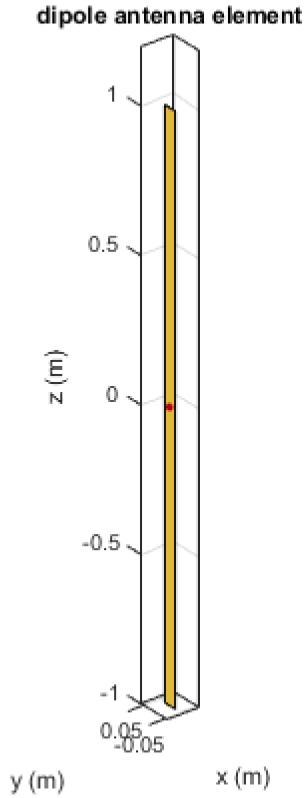
Examples

Create and View Dipole Antenna

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

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

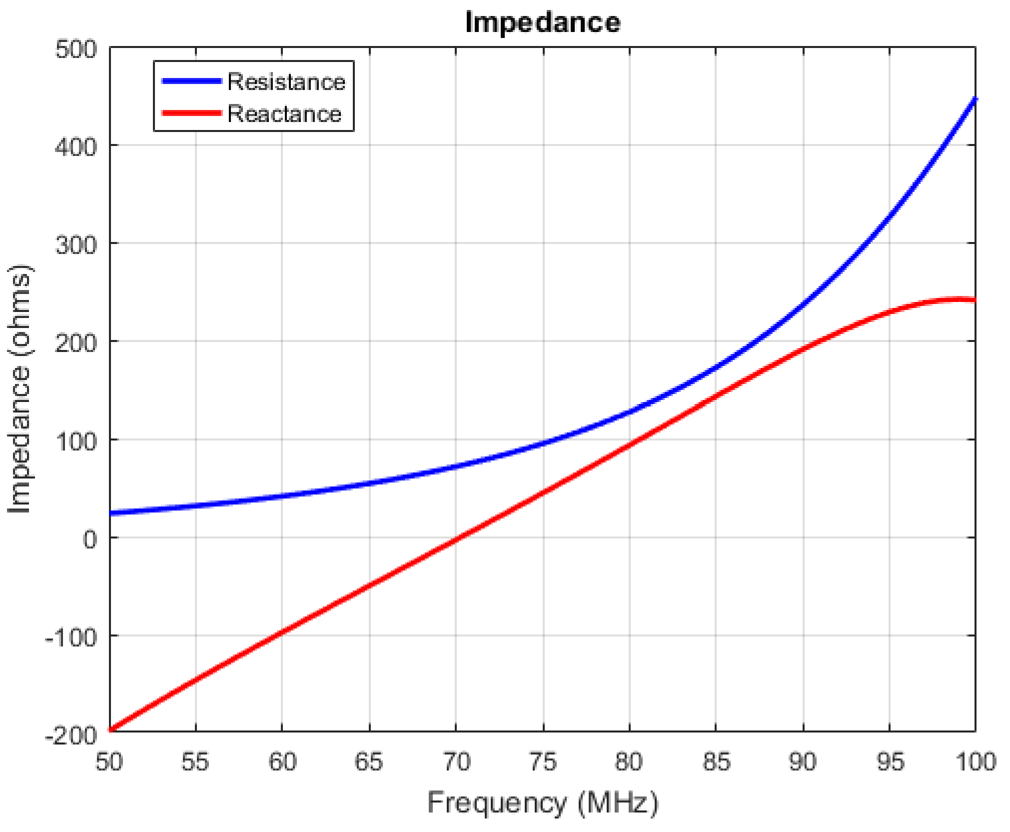
```
d =

  dipole with properties:

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



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))
```



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

monopole | loopCircular | slot | `cylinder2strip`

More About

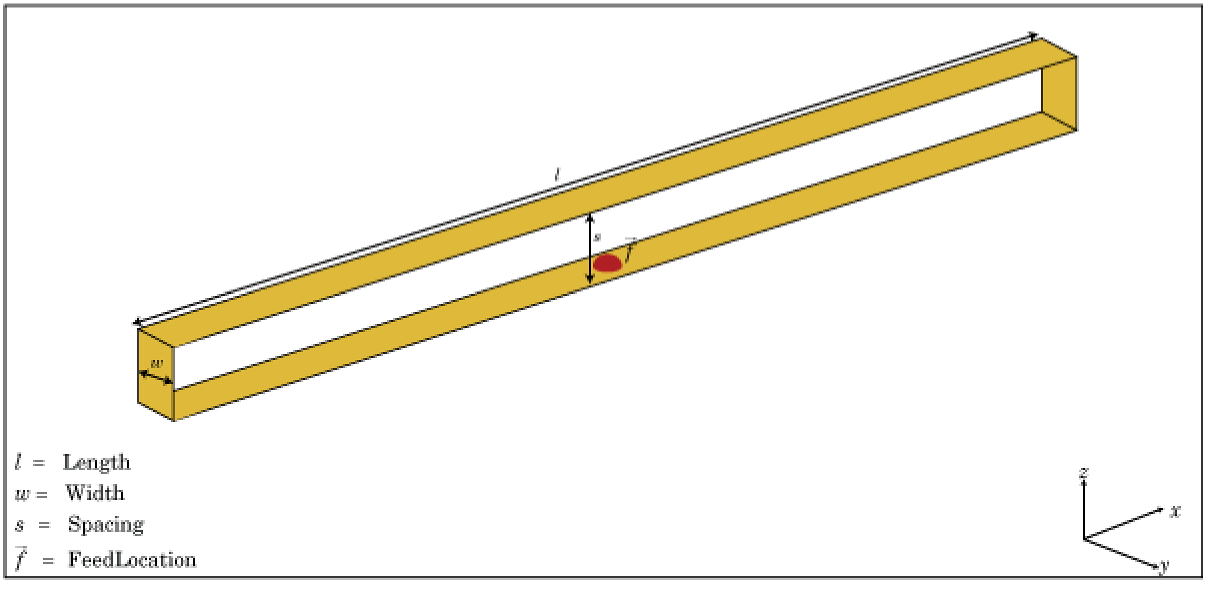
- “Rotate Antenna and Arrays”

Introduced in R2015a

dipoleFolded class

Create folded dipole antenna

Description



The `dipolefolded` class creates 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.

Construction

`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 in meters

Folded dipole length, specified as the comma-separated pair consisting of `'Length'` and 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 in meters

Folded dipole width, specified as the comma-separated pair consisting of `'Width'` and 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 the comma-separated pair consisting of `'Spacing'` and a scalar in meters. The value must be less than `Length`/50.

Example: `'Spacing'`,3

Data Types: `double`

`'Tilt'` — Tilt angle of antenna

0 (default) | scalar in degrees | vector in degrees

Tilt angle of antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or vector in degrees.

Example: `'Tilt',90`

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

Data Types: `double`

`'TiltAxis'` — Tilt axis of antenna

[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,"Rotate Antenna and Arrays"

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

Example: `'TiltAxis'`,`'Z'`

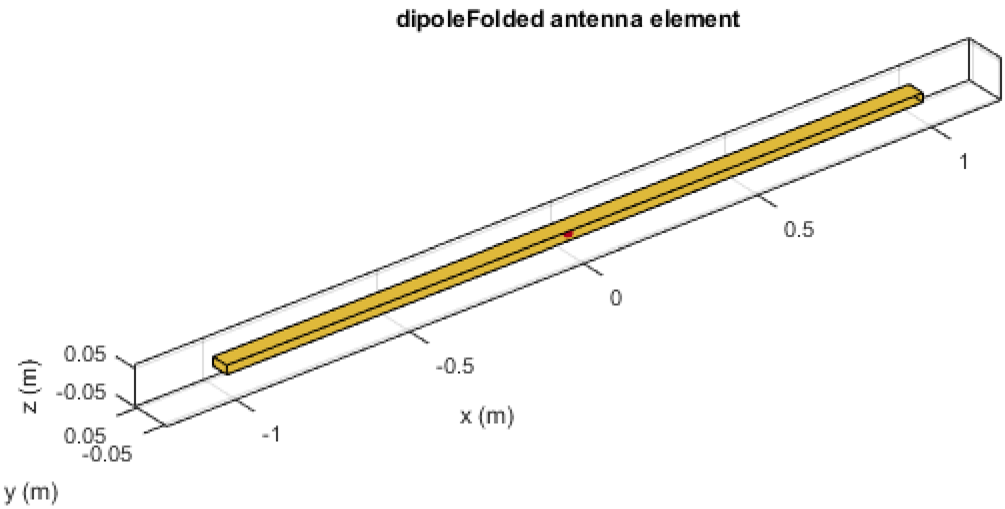
Data Types: `double`

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)
show(df)


df = 

  dipoleFolded with properties:

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



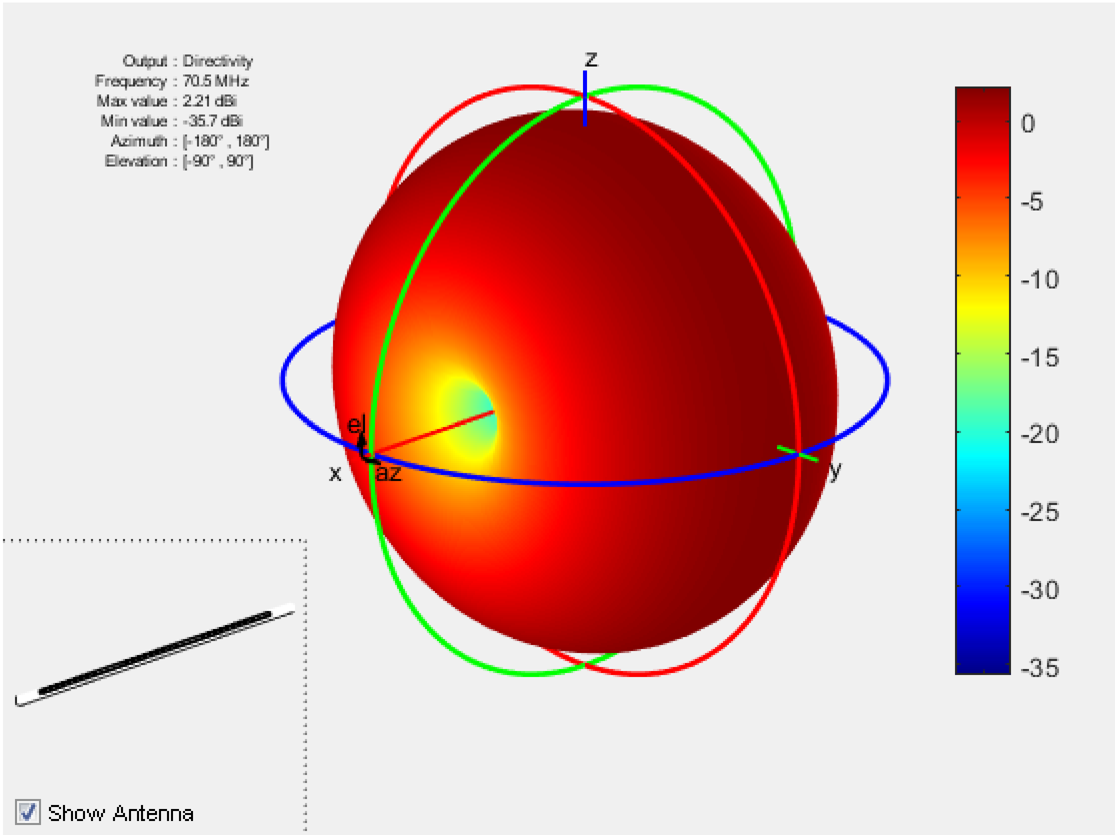
Raditaion Pattern of Folded Dipole Antenna

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

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

```
df =

  dipoleFolded with properties:

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

```
        Tilt: 0
    TiltAxis: [1 0 0]
```



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 | dipole | monopole | `cylinder2strip`

More About

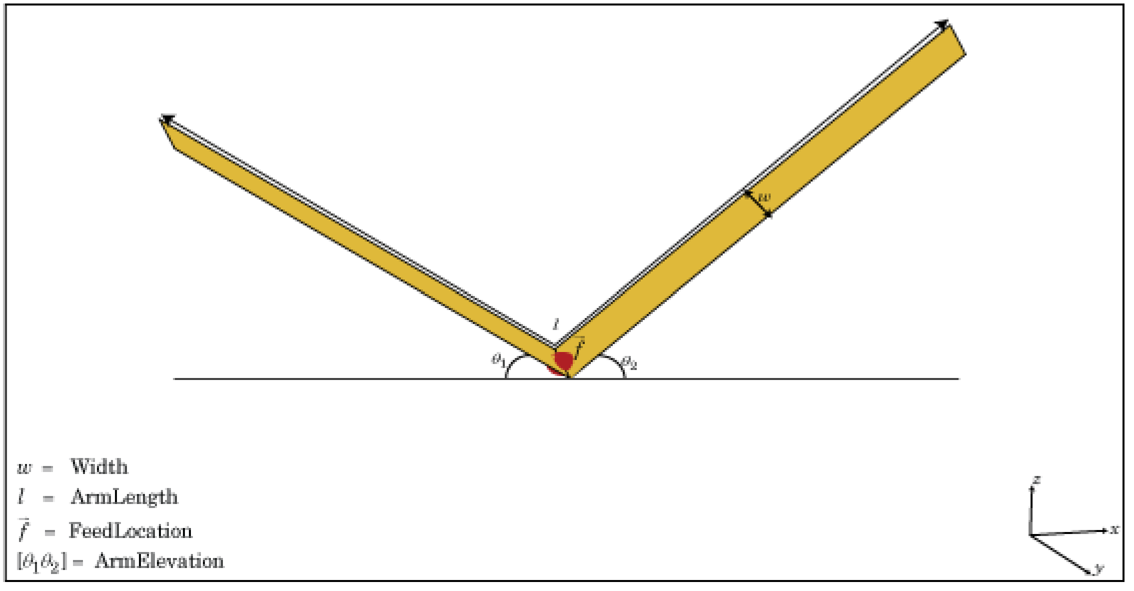
- “Rotate Antenna and Arrays”

Introduced in R2015a

dipoleVee class

Create V-dipole antenna

Description



The `dipoleVee` class creates 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.

Construction

`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 in meters

Length of two arms, specified as the comma-separated pair consisting of `'ArmLength'` and 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 in meters

V-dipole arm width, specified as the comma-separated pair consisting of `'Width'` and 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 in degrees

Angle made by two arms about X-Y plane, specified as the comma-separated pair consisting of `'ArmElevation'` and a two-element vector in degrees.

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

Data Types: `double`

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

Tilt angle of antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or vector in degrees.

Example: `'Tilt',90`

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

Data Types: `double`

`'TiltAxis'` — Tilt axis of antenna
[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,“Rotate Antenna and Arrays”

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

Example: `'TiltAxis'`,`'Z'`

Data Types: `double`

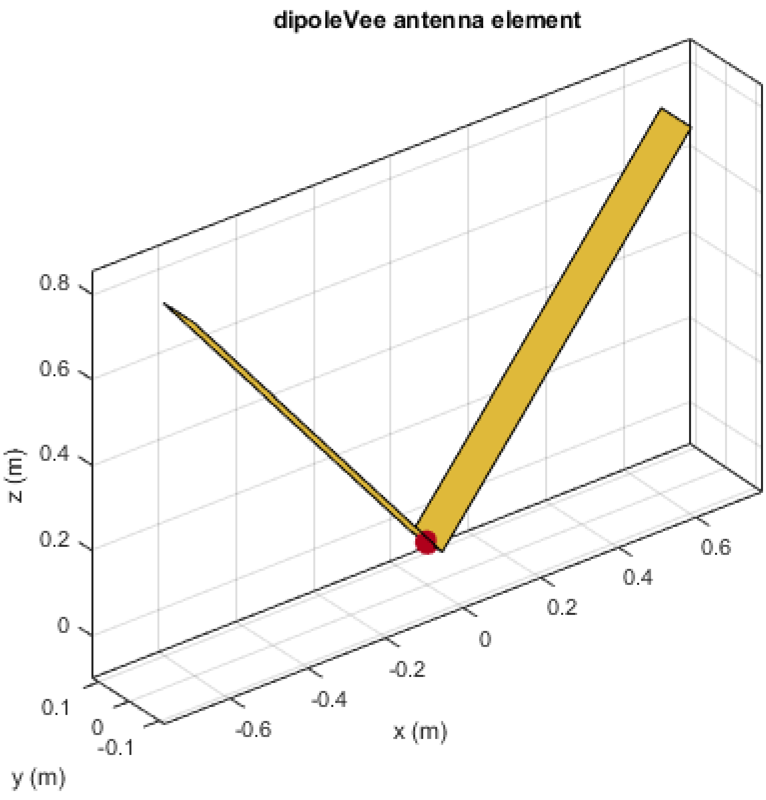
Examples

Create V-Dipole Antenna

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

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

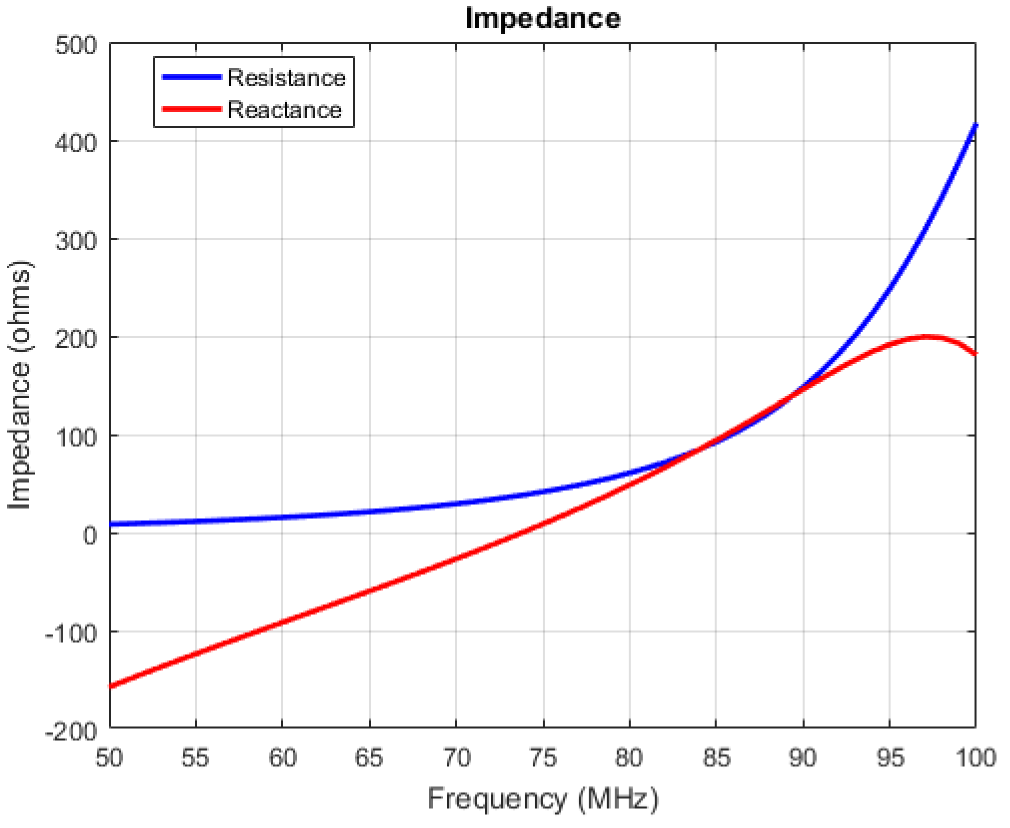
```
dv =

  dipoleVee with properties:

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



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

dipole | dipoleFolded | loopCircular | `cylinder2strip`

More About

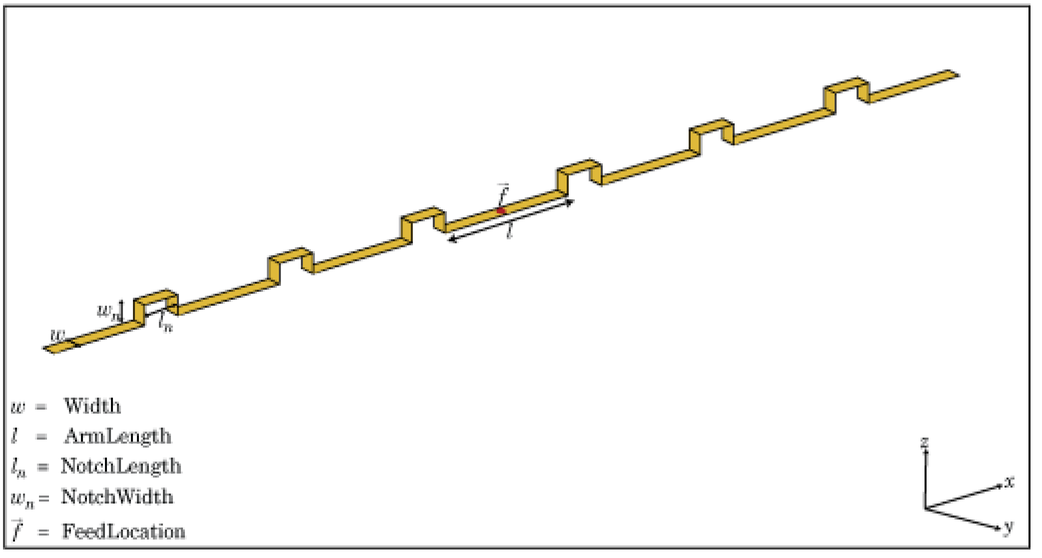
- "Rotate Antenna and Arrays"

Introduced in R2015a

dipoleMeander class

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 Y-Z plane.

Construction

`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 in meters

Dipole width, specified as the comma-separated pair consisting of `'Width'`and 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 in meters

Length of individual dipole arms, specified as the comma-separated pair consisting of `'ArmLength'` and 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 in meters

Notch length along the length of the antenna, specified as the comma-separated pair consisting of `'NotchLength'` and 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 in meters

Notch width perpendicular to the length of the antenna, specified as the comma-separated pair consisting of `'NotchWidth'` and a scalar in meters.

Example: `'NotchWidth'`,1

Data Types: `double`

`'Tilt'` — Tilt angle of antenna

0 (default) | scalar in degrees | vector in degrees

Tilt angle of antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or vector in degrees.

Example: `'Tilt',90`

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

Data Types: `double`

`'TiltAxis'` — Tilt axis of antenna

[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,“Rotate Antenna and Arrays”

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

Example: 'TiltAxis','Z'

Data Types: double

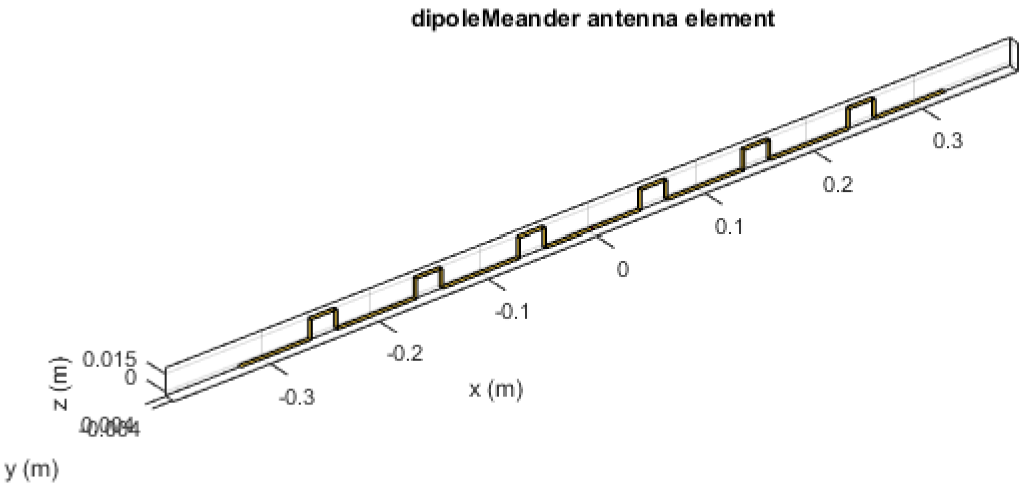
Examples

Create and View Meander Dipole Antenna

Create and view the default meander dipole antenna.

```
dm = dipoleMeander
show(dm)
```

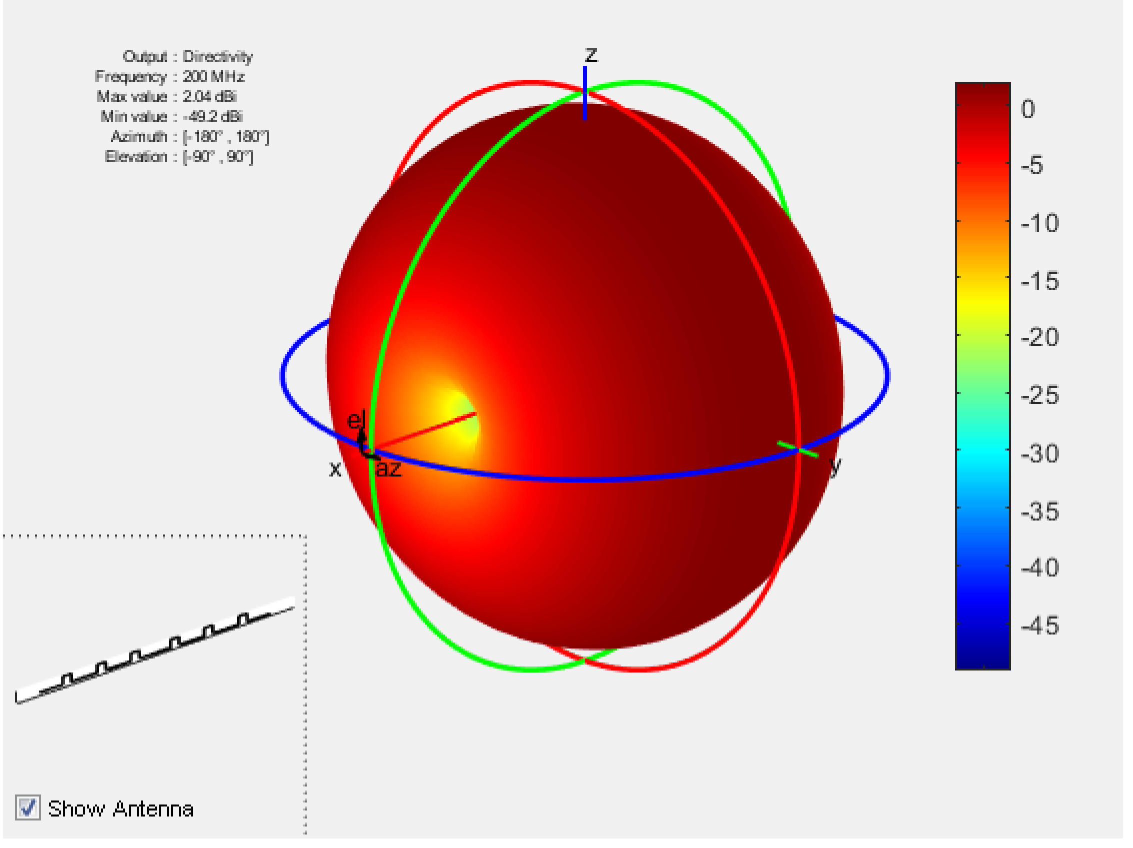
```
dm =

  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]
```



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

More About

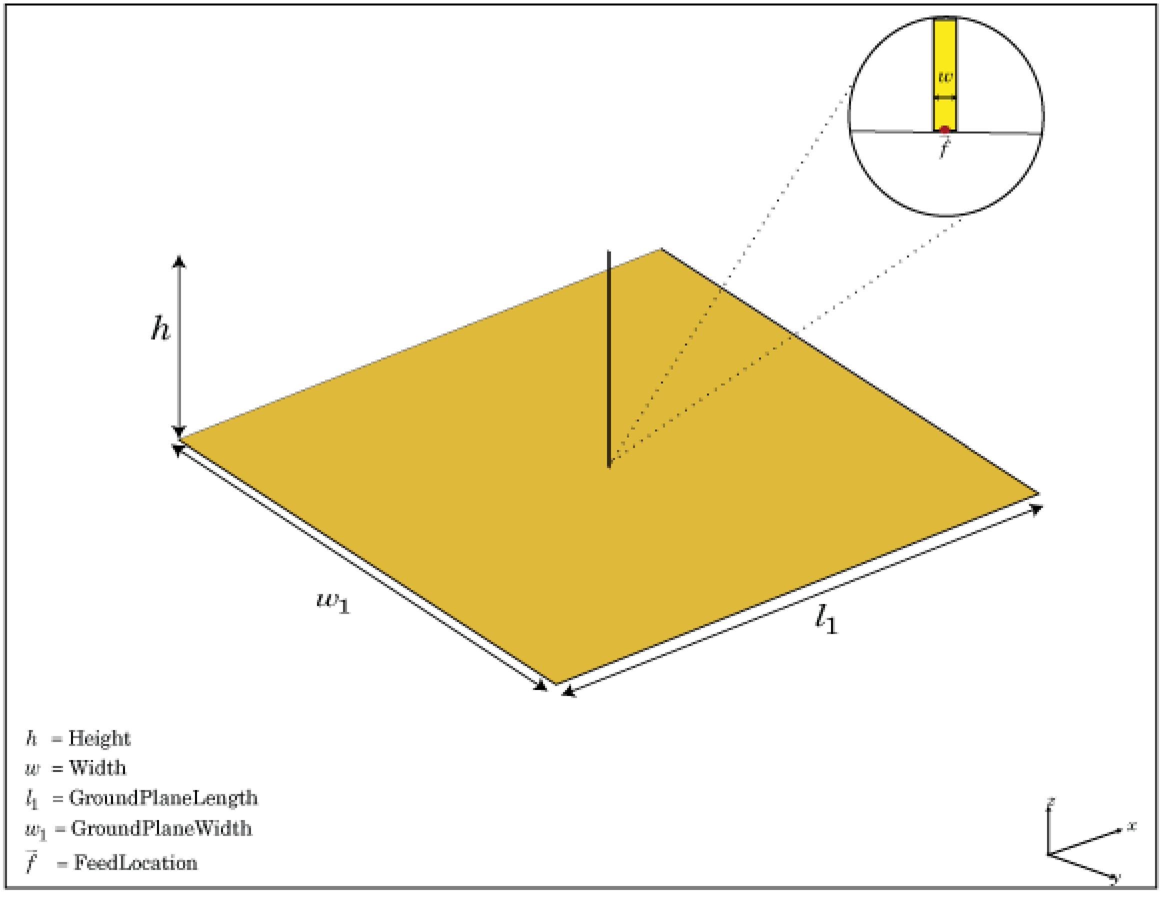
- "Rotate Antenna and Arrays"

Introduced in R2015a

monopole class

Create monopole antenna over rectangular ground plane

Description



The `monopole` class creates 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.

Construction

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

`h = 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 in meters

Height of vertical element along z-axis, specified as the comma-separated pair consisting of `'Height'` and 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 in meters

Monopole width, specified as the comma-separated pair consisting of `'Width'` and 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 in meters

Ground plane length along x-axis, specified as the comma-separated pair consisting of 'GroundPlaneLength' and 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 in meters

Ground plane width along y-axis, specified as the comma-separated pair consisting of 'GroundPlaneWidth' and 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 the comma-separated pair of 'FeedOffset' and a two-element vector.

Example: 'FeedOffset',[2 1]

Data Types: double

'Tilt' — Tilt angle of antenna

0 (default) | scalar in degrees | vector in degrees

Tilt angle of antenna, specified as the comma-separated pair consisting of 'Tilt' and a scalar or vector in degrees.

Example: `'Tilt',90`

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

Data Types: `double`

`'TiltAxis'` — Tilt axis of antenna
[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,“Rotate Antenna and Arrays”

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

Example: `'TiltAxis','Z'`

Data Types: `double`

Examples

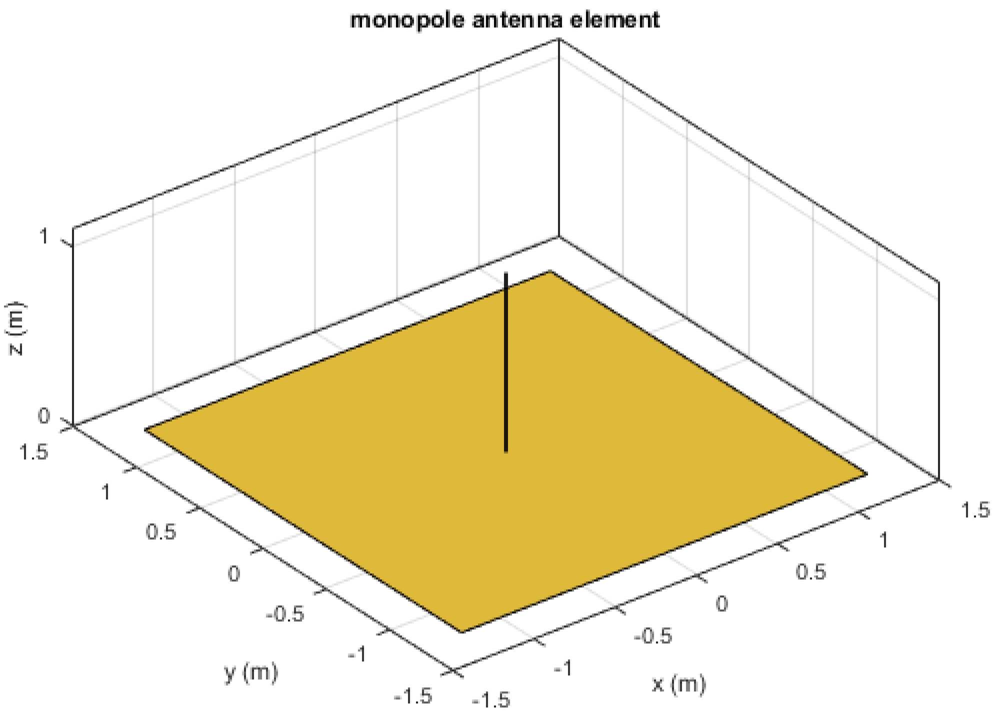
Create and View Monopole Antenna

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

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

```
m =

  monopole with properties:

             Height: 1
```

```
              Width: 0.0100
   GroundPlaneLength: 2.5000
    GroundPlaneWidth: 2.5000
          FeedOffset: [0 0]
                Tilt: 0
            TiltAxis: [1 0 0]
```

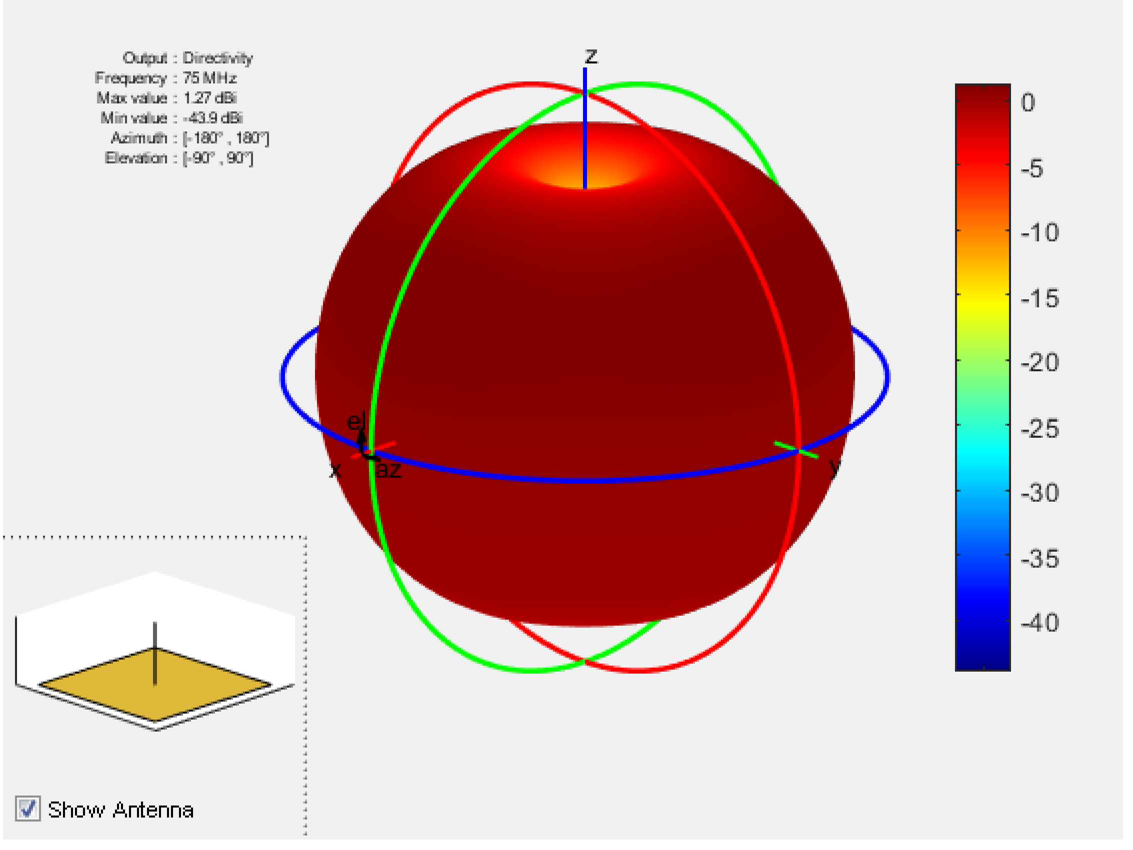


Radiation Pattern of Monopole Antenna

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 | patchMicrostrip | monopoletophat

More About

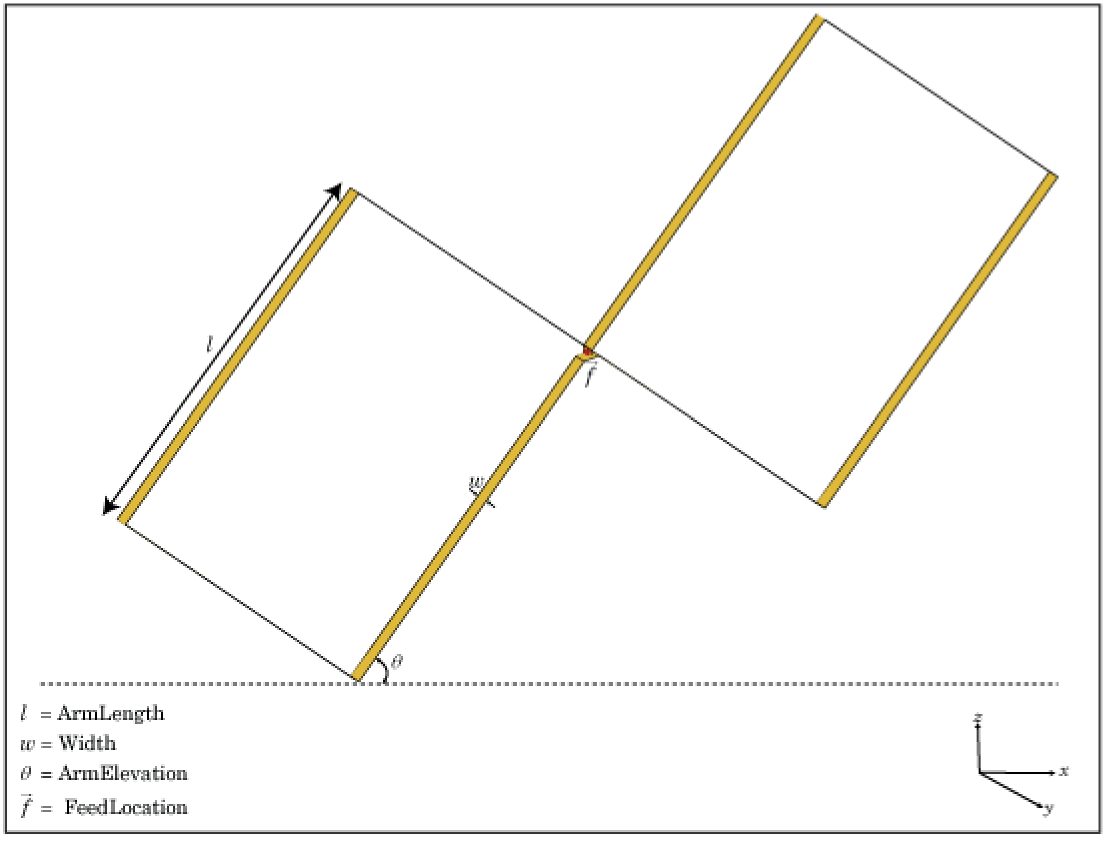
- “Rotate Antenna and Arrays”

Introduced in R2015a

biquad class

Create biquad antenna

Description



The `biquad` class creates a biquad antenna. The 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:

$$w = 2d = 4r$$

- *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.

Syntax

Construction

`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 in meters

Length of two arms, specified as the comma-separated pair consisting of `'ArmLength'` and 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 in meters

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

Example: `'Width',0.006`

Data Types: `double`

`'ArmElevation'` — Angle formed by biquad arms to X-Y plane
45 (default) | scalar in degrees

Angle formed by biquad arms to the X-Y plane, specified as the comma-separated pair consisting of `'ArmElevation'` and a scalar in meters.

Example: `'ArmElevation', 50`

Data Types: `double`

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

Tilt angle of antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or vector in degrees.

Example: `'Tilt',90`

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

Data Types: `double`

`'TiltAxis'` — Tilt axis of antenna
[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,“Rotate Antenna and Arrays”

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

Example: `'TiltAxis','Z'`

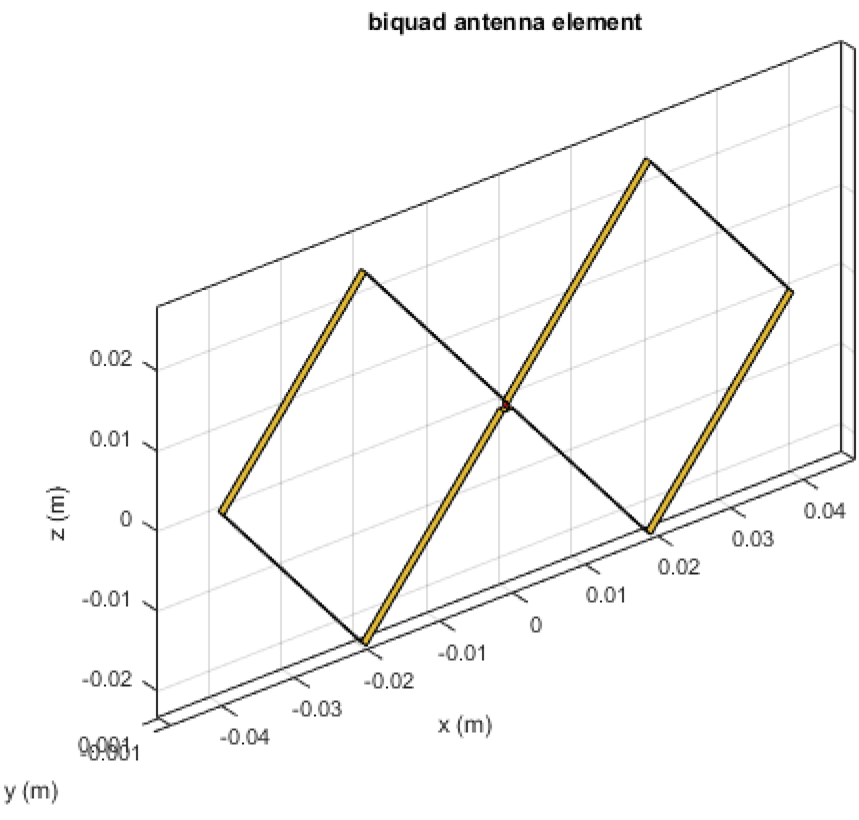
Data Types: `double`

Examples

Create and View Biquad Antenna

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

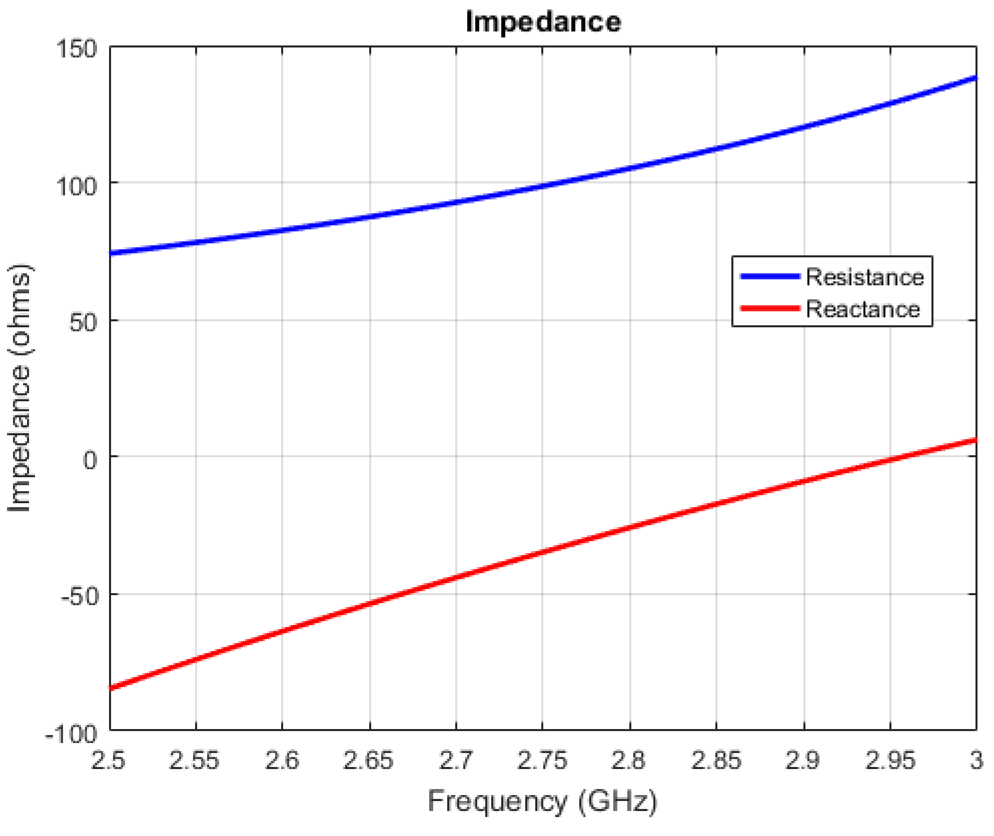
```
bq = biquad('ArmElevation',50);
show(bq)
```



Impedance of Biquad Antenna

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

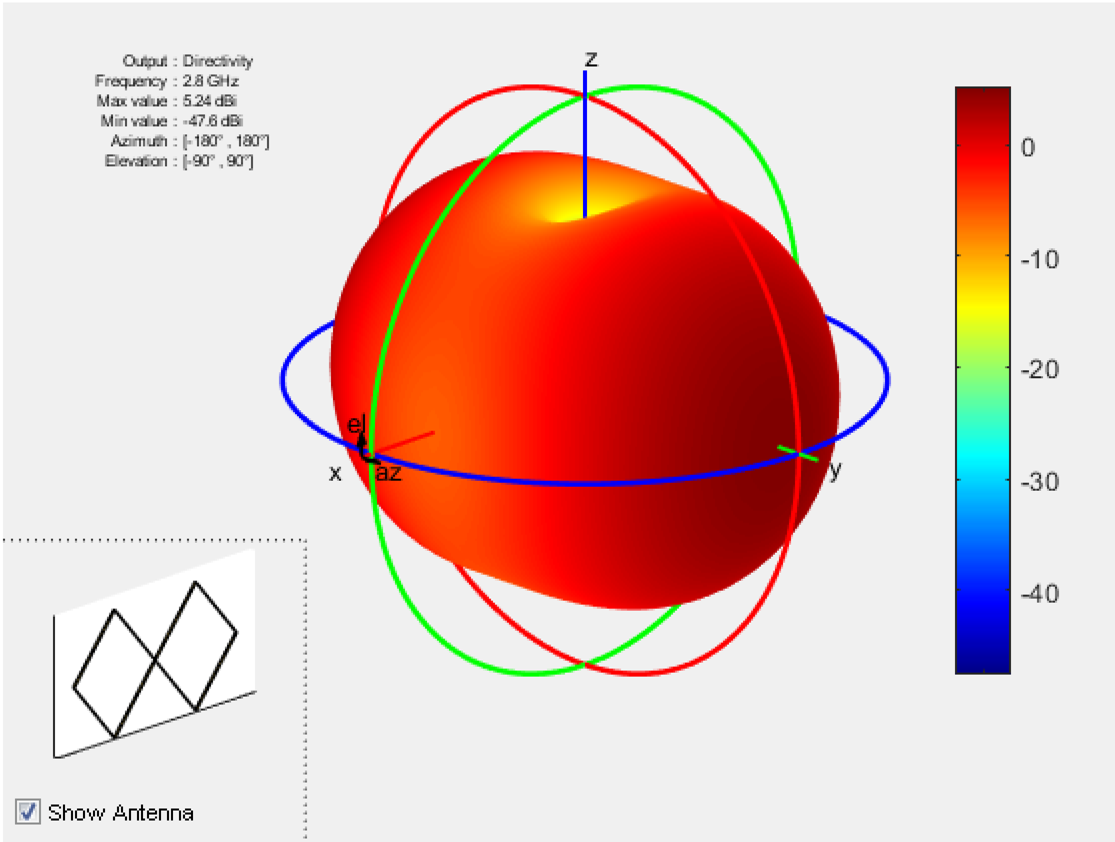
```
bq = biquad('ArmElevation',50);
impedance(bq,linspace(2.5e9,3e9,51));
```



Radiation Pattern of Biquad Antenna

Calculate the radiation pattern of a biquad antenna at a frequency of 2.8 GHz.

```
bq = biquad('ArmElevation',50);
pattern(bq, 2.8e9)
```



See Also

dipole | dipoleFolded | loopCircular

More About

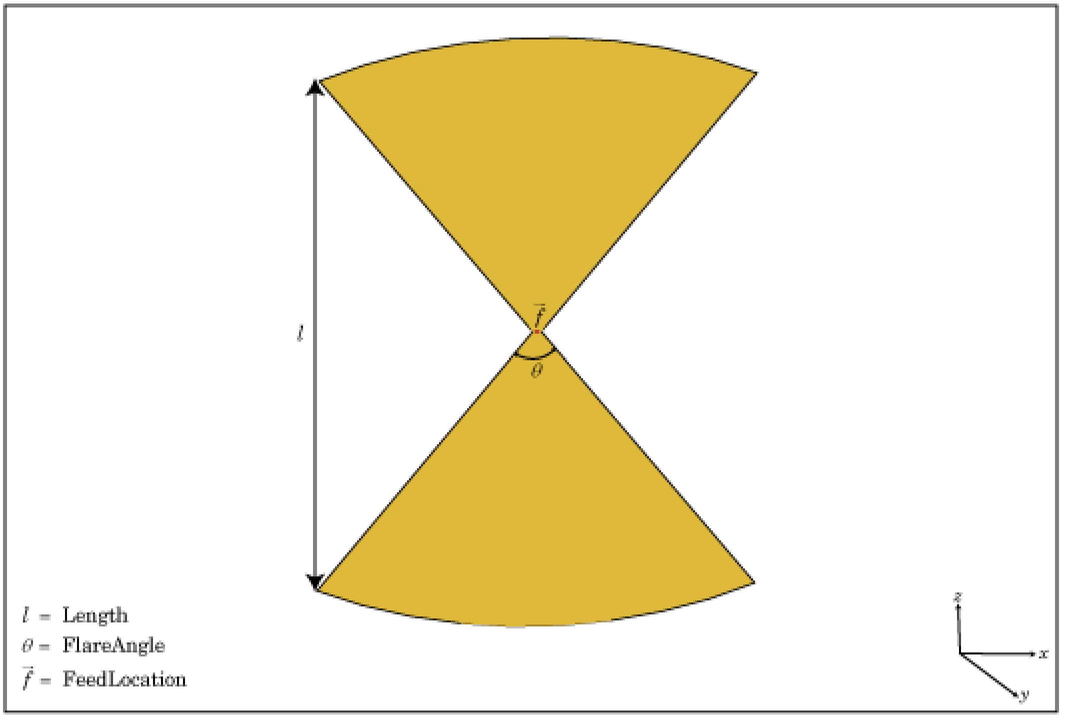
- “Rotate Antenna and Arrays”

Introduced in R2015b

bowtieRounded class

Create rounded bowtie dipole antenna

Description



The `bowtieRounded` class creates 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.

Construction

`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 in meters

Rounded bowtie length, specified as the comma-separated pair consisting of `'Length'` and 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 in degrees

Rounded bowtie flare angle, specified as the comma-separated pair consisting of `'FlareAngle'` and a scalar in degrees.

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

Example: `'FlareAngle'`,80

Data Types: `double`

`'Tilt'` — Tilt angle of antenna

0 (default) | scalar in degrees | vector in degrees

Tilt angle of antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or vector in degrees.

Example: `'Tilt',90`

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

Data Types: `double`

`'TiltAxis'` — Tilt axis of antenna
[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,“Rotate Antenna and Arrays”

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

Example: `'TiltAxis','Z'`

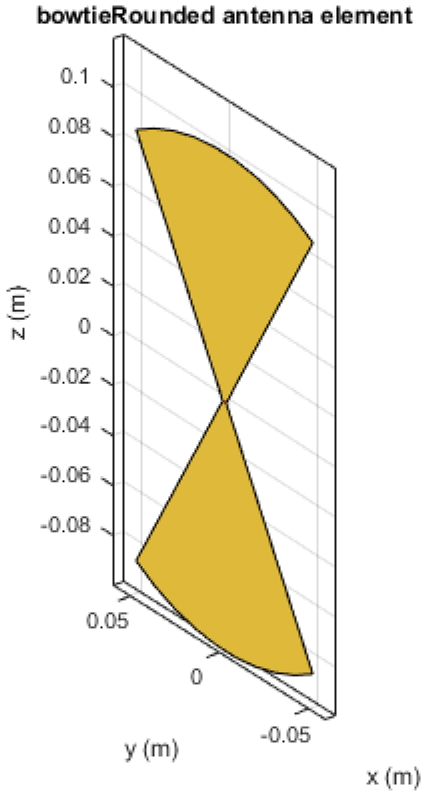
Data Types: `double`

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.

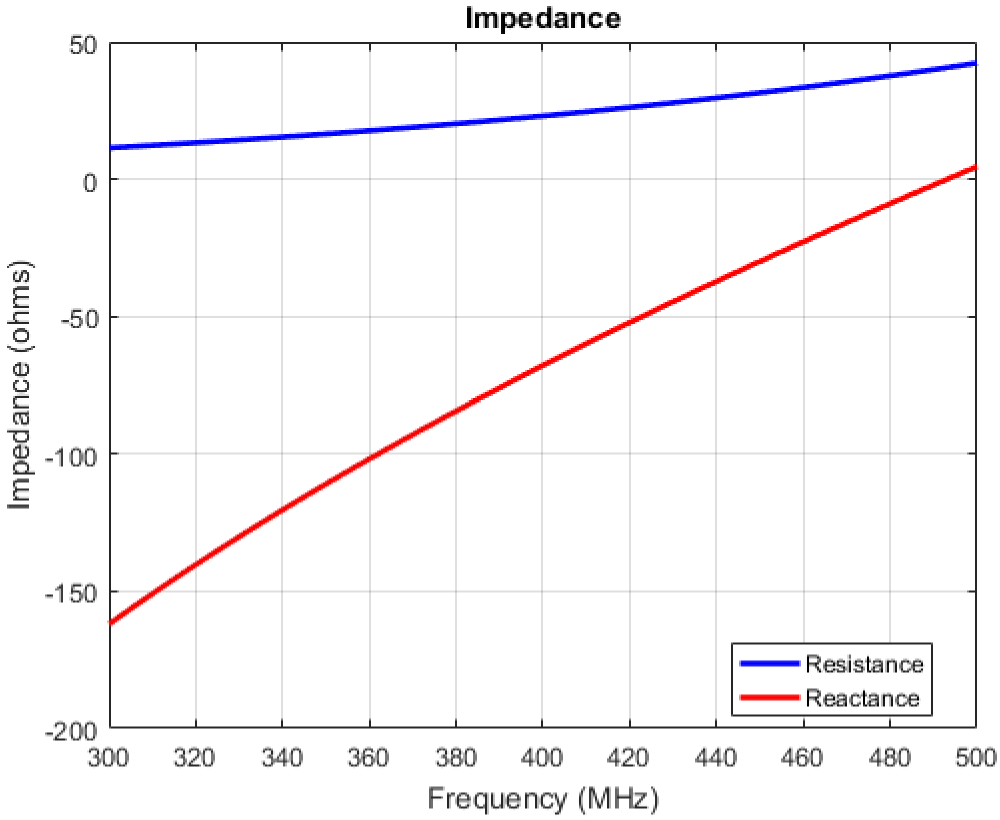
```
b = bowtieRounded('FlareAngle',60);
show(b);
```



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

dipole | dipoleFolded | bowtieTriangular

More About

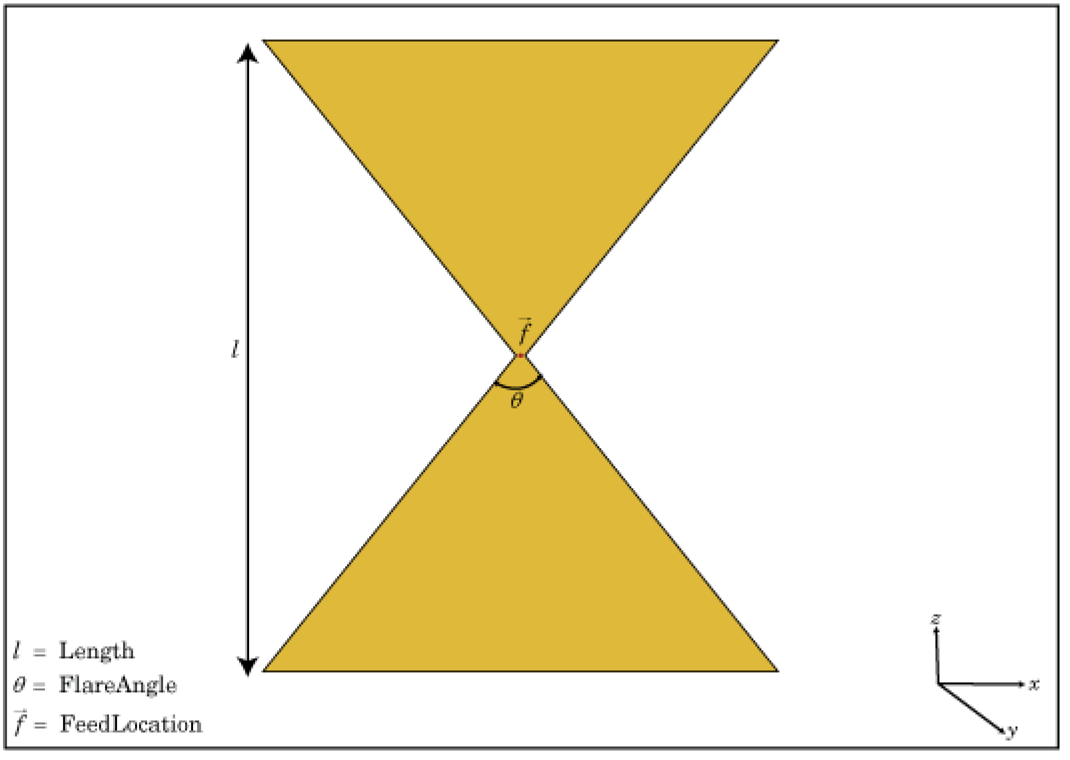
- "Rotate Antenna and Arrays"

Introduced in R2015a

bowtieTriangular class

Create planar bowtie dipole antenna

Description



The `bowtieTriangular` class creates 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.

Construction

`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 in meters

Planar bowtie length, specified as the comma-separated pair consisting of `'Length'` and 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 in degrees

Planar bowtie flare angle near the feed, specified as the comma-separated pair consisting of `'FlareAngle'` and a scalar in meters.

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

Example: `'FlareAngle'`,80

Data Types: `double`

`'Tilt'` — Tilt angle of antenna

0 (default) | scalar in degrees | vector in degrees

Tilt angle of antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or vector in degrees.

Example: `'Tilt',90`

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

Data Types: `double`

`'TiltAxis'` — Tilt axis of antenna

[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,"Rotate Antenna and Arrays"

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

Example: `'TiltAxis'`,`'Z'`

Data Types: `double`

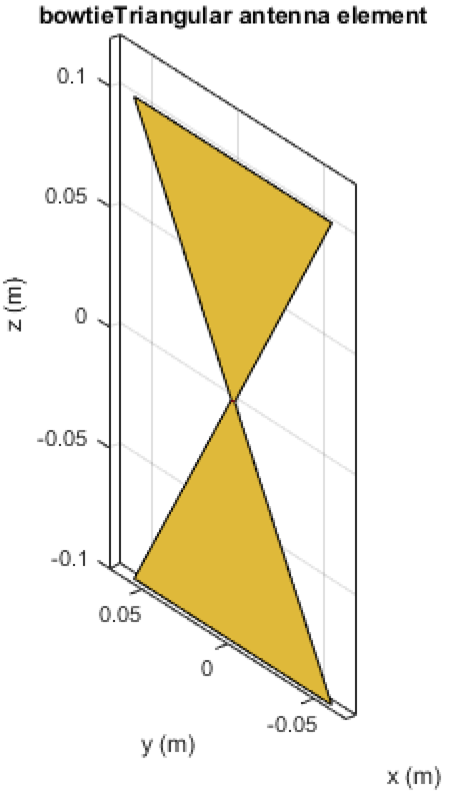
Examples

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)
show(b)
```

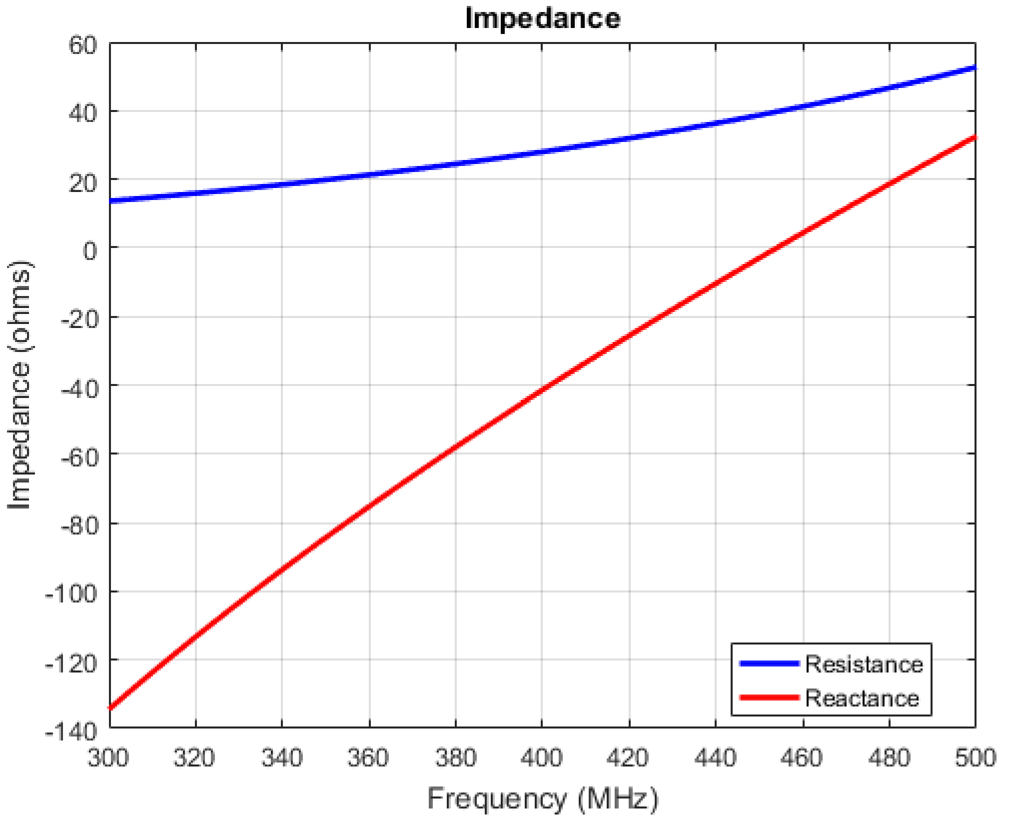
```
b =

  bowtieTriangular with properties:

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



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

dipole | dipoleVee | bowtieRounded

More About

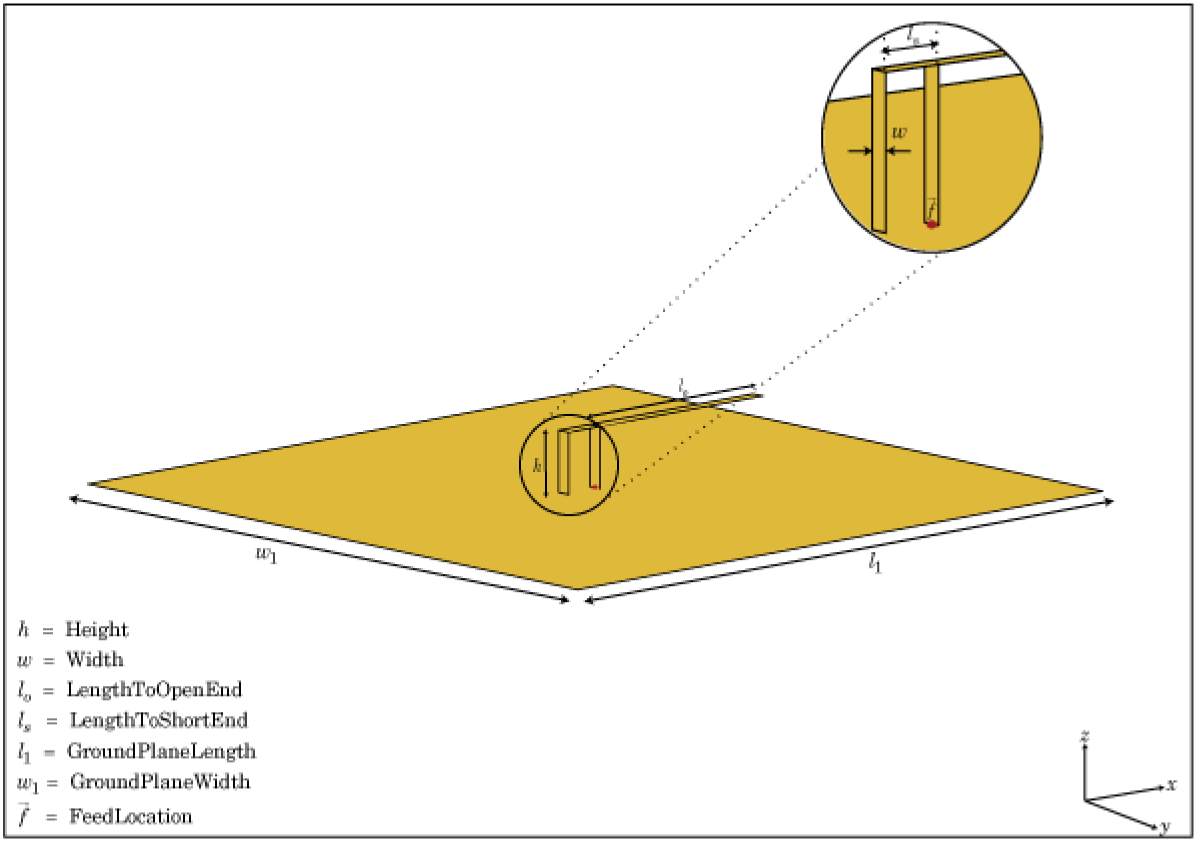
- “Rotate Antenna and Arrays”

Introduced in R2015a

invertedF class

Create inverted-F antenna over rectangular ground plane

Description



The `invertedF` class creates 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.

Construction

`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 in meters

Vertical element height along z-axis, specified as the comma-separated pair consisting of `'Height'` and a scalar in meters.

Example: `'Height'`, 3

Data Types: `double`

`'Width'` — Strip width
0.0020 (default) | scalar in meters

Strip width, specified as the comma-separated pair consisting of `'Width'` and 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 in meters

Stub length from feed to open end, specified as the comma-separated pair consisting of 'LengthToOpenEnd' and a scalar in meters.

Example: 'LengthToOpenEnd',0.05

'LengthToShortEnd' — Stub length from feed to shorting end

0.0060 (default) | scalar in meters

Stub length from feed to shorting end, specified as the comma-separated pair consisting of 'LengthToShortEnd' and a scalar in meters.

Example: 'LengthToShortEnd',0.0050

'GroundPlaneLength' — Ground plane length along x-axis

0.1000 (default) | scalar in meters

Ground plane length along x-axis, specified as the comma-separated pair consisting of 'GroundPlaneLength' and a scalar in meters. Setting 'GroundPlaneLength' to Inf, will use 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 in meters

Ground plane width along y-axis, specified as the comma-separated pair consisting of 'GroundPlaneWidth' and a scalar in meters. Setting 'GroundPlaneWidth' to Inf, will use 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 the comma-separated pair of `‘FeedOffset’` and a two-element vector.

Example: `‘FeedOffset’`,[2 1]

Data Types: `double`

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

Tilt angle of antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or vector in degrees.

Example: `'Tilt',90`

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

Data Types: `double`

`'TiltAxis'` — Tilt axis of antenna
[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,“Rotate Antenna and Arrays”

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

Example: `'TiltAxis'`,`'Z'`

Data Types: `double`

Definitions

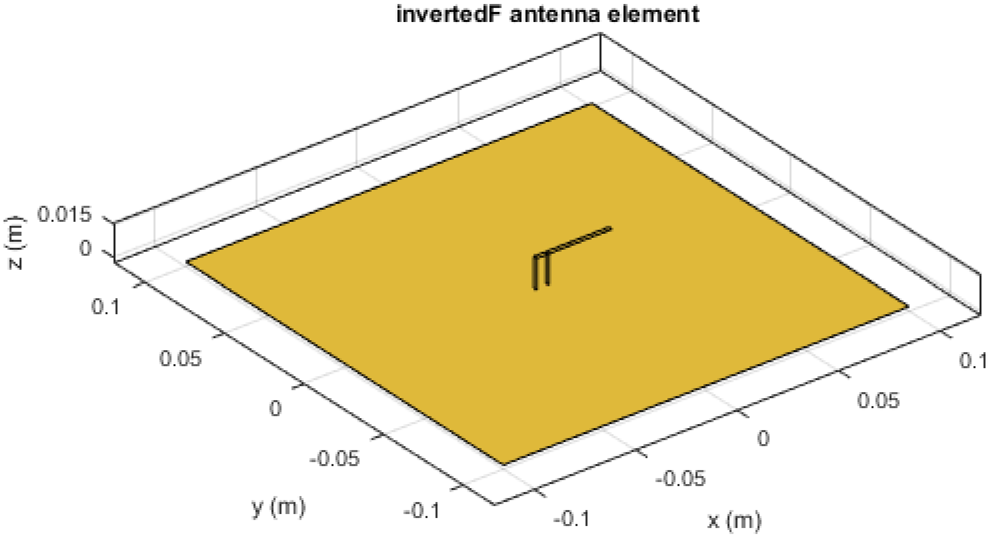
To rotate antenna elements in Antenna Toolbox™, use the *Tilt* and *TiltAxis* property. For more information please refer, “Rotate Antenna and Arrays”.

Examples

Create and View Inverted-F Antenna

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

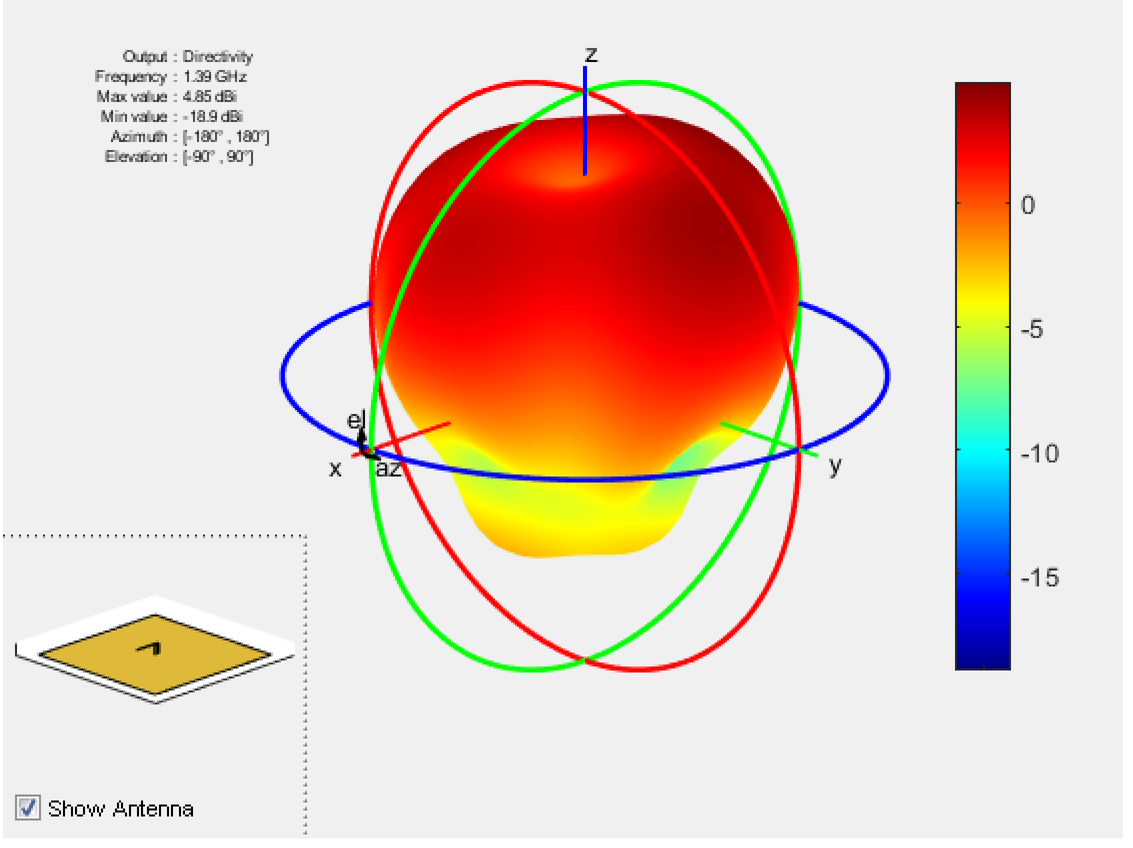
```
f = invertedF('Height',14e-3, 'GroundPlaneLength',200e-3,          ...
                  'GroundPlaneWidth',200e-3);
show(f)
```



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

pifa | patchMicrostrip | `cylinder2strip` | invertedF

More About

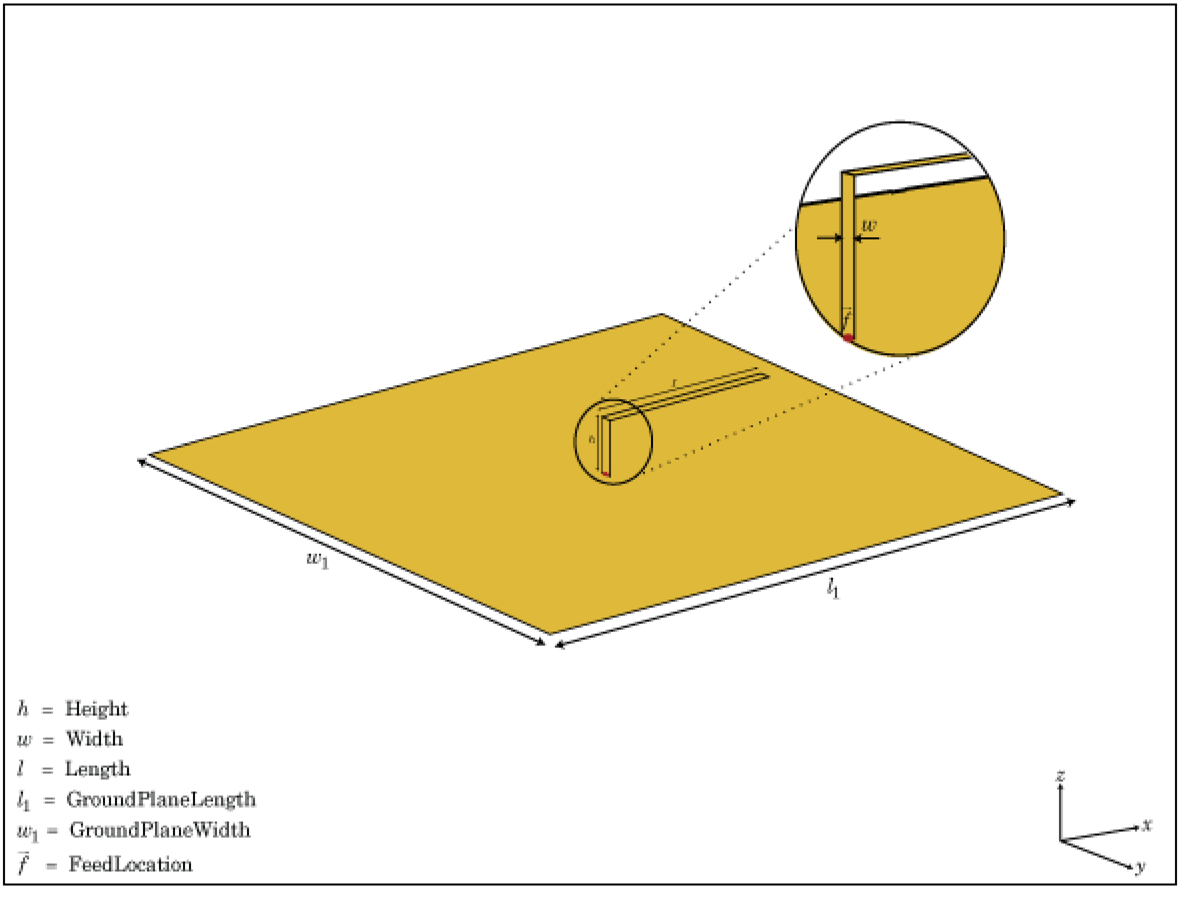
- “Rotate Antenna and Arrays”

Introduced in R2015a

invertedL class

Create inverted-L antenna over rectangular ground plane

Description



The `invertedF` class creates 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.

Construction

`h = 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.

`h = 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 in meters

Height of inverted element along z-axis, specified as the comma-separated pair consisting of `'Height'` and a scalar in meters.

Example: `'Height'`, 3

Data Types: `double`

`'Width'` — Strip width

0.0020 (default) | scalar in meters

Strip width, specified as the comma-separated pair consisting of `'Width'` and 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 in meters

Stub length along x-axis, specified as the comma-separated pair consisting of 'Length' and a scalar in meters.

Example: 'Length',0.01

'GroundPlaneLength' — Ground plane length along x-axis

0.1000 (default) | scalar in meters

Ground plane length along x-axis, specified as the comma-separated pair consisting of 'GroundPlaneLength' and 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 in meters

Ground plane width along y-axis, specified as the comma-separated pair consisting of 'GroundPlaneWidth' and 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 the comma-separated pair of ‘FeedOffset’ and a two-element vector.

Example: ‘FeedOffset’,[2 1]

Data Types: double

'Tilt' — Tilt angle of antenna
0 (default) | scalar in degrees | vector in degrees

Tilt angle of antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or vector in degrees.

Example: `'Tilt',90`

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

Data Types: `double`

'TiltAxis' — Tilt axis of antenna
[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,"Rotate Antenna and Arrays"

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

Example: `'TiltAxis','Z'`

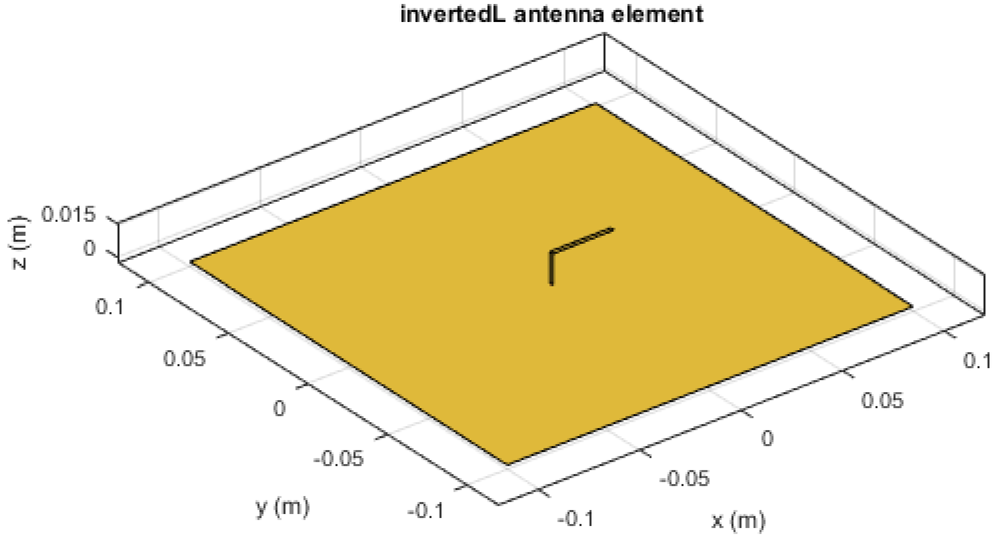
Data Types: `double`

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.

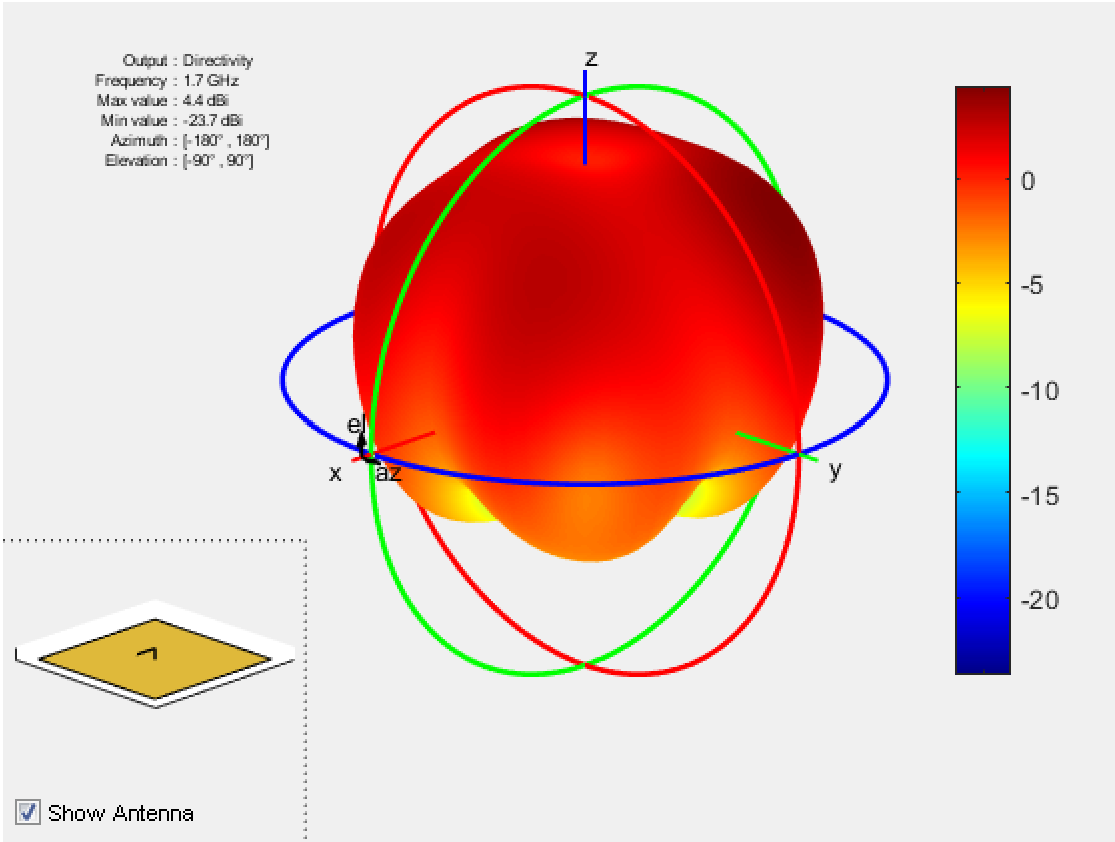
```
il = invertedL('Length',30e-3, 'GroundPlaneLength',200e-3,...
                    'GroundPlaneWidth',200e-3);
show(il)
```



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

vivaldi | invertedF | `cylinder2strip` | monopole

More About

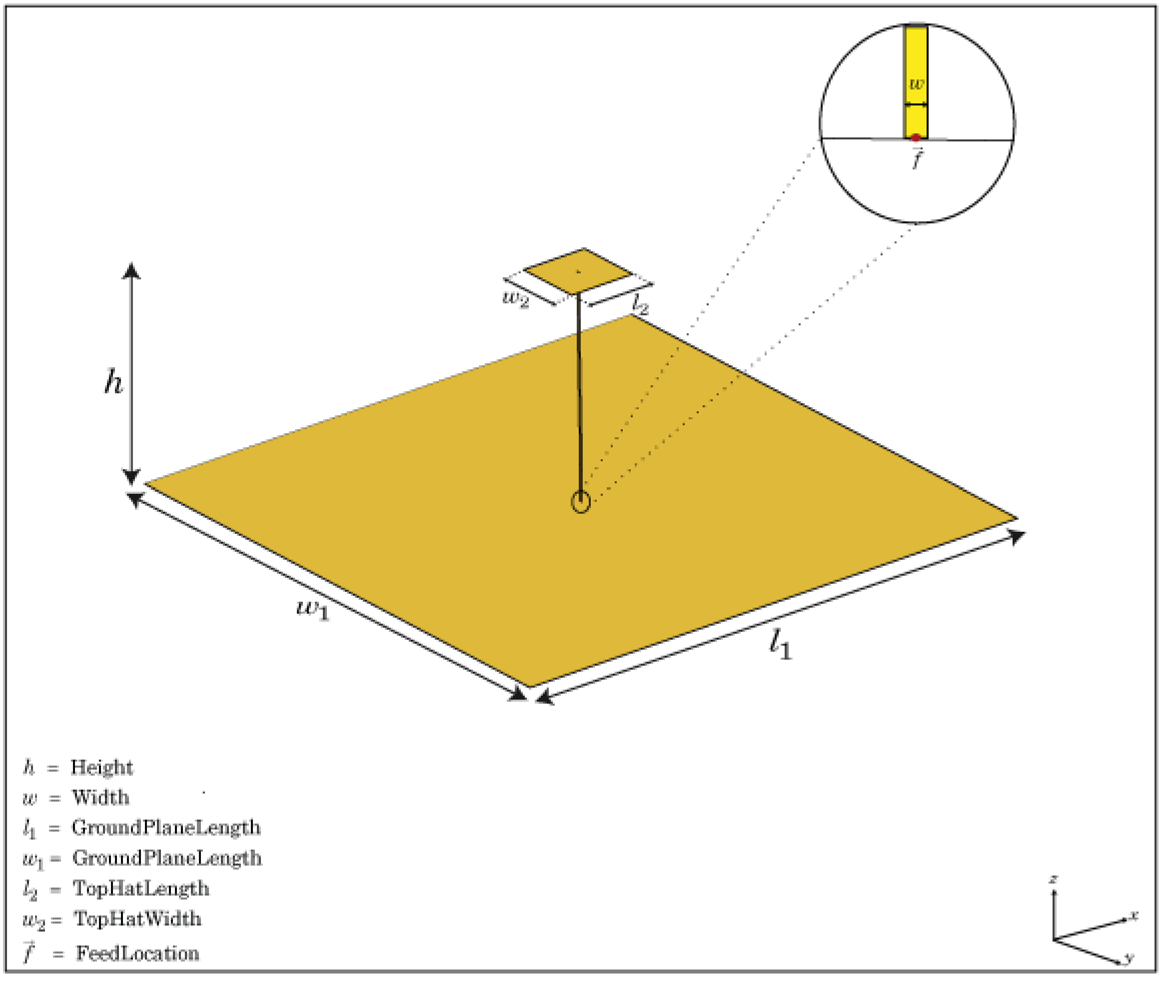
- "Rotate Antenna and Arrays"

Introduced in R2015a

monopoleTopHat class

Create capacitively loaded monopole antenna over rectangular ground plane

Description



The `monopoleTopHat` class creates 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.

Construction

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

`h = 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 in meters

Monopole height, specified as the comma-separated pair consisting of 'Height' and 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 in meters

Monopole width, specified as the comma-separated pair consisting of 'Width' and 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 in meters

Ground plane length along x-axis, specified as the comma-separated pair consisting of 'GroundPlaneLength' and 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 in meters

Ground plane width along y-axis, specified as the comma-separated pair consisting of 'GroundPlaneWidth' and 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 in meters

Top hat length along x-axis, specified as the comma-separated pair consisting of `'TopHatLength'` and a scalar in meters.

Example: `'TopHatLength'`,4

Data Types: `double`

`'TopHatWidth'` — Top hat width along y-axis
0.2500 (default) | scalar in meters

Top hat width along y-axis, specified as the comma-separated pair consisting of `'TopHatWidth'` and 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 the comma-separated pair of `‘FeedOffset’` and a two-element vector.

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

Data Types: `double`

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

Tilt angle of antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or vector in degrees.

Example: `'Tilt',90`

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

Data Types: `double`

`'TiltAxis'` — Tilt axis of antenna
[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,“Rotate Antenna and Arrays”

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

Example: `'TiltAxis','Z'`

Data Types: `double`

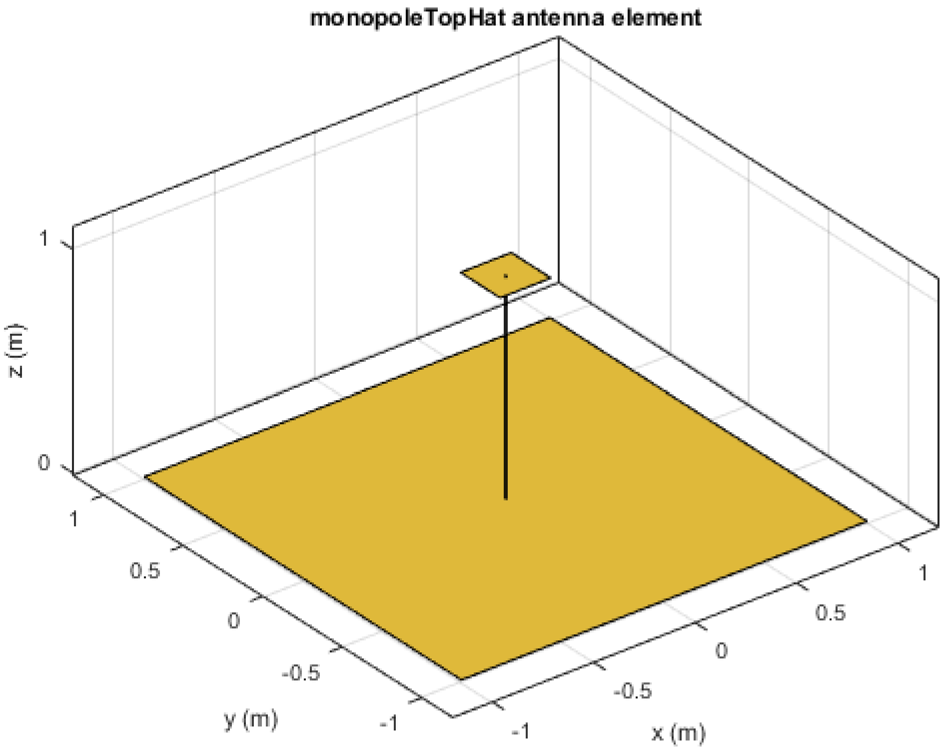
Examples

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
show(th)
```

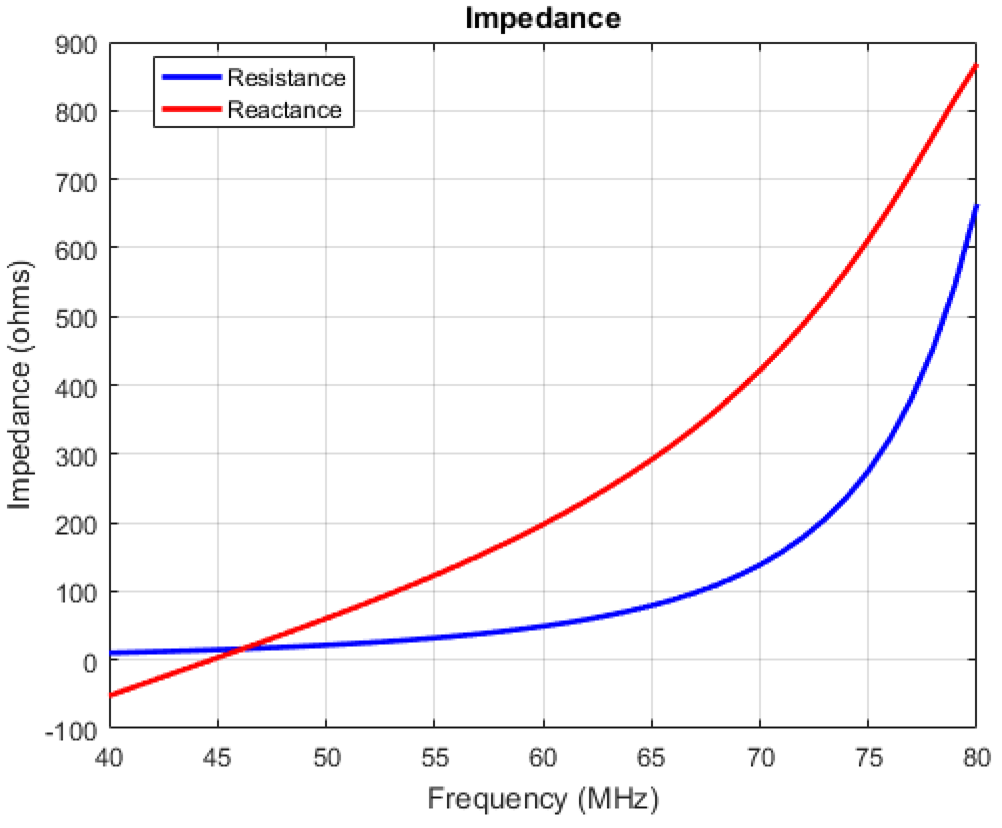
```
th = 

  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]
```



Calculate Impedance of Top Hat Monopole Antenna

Calculate and plot the impedance of a top hat monopole over a frequency range of 40MHz-80MHz.

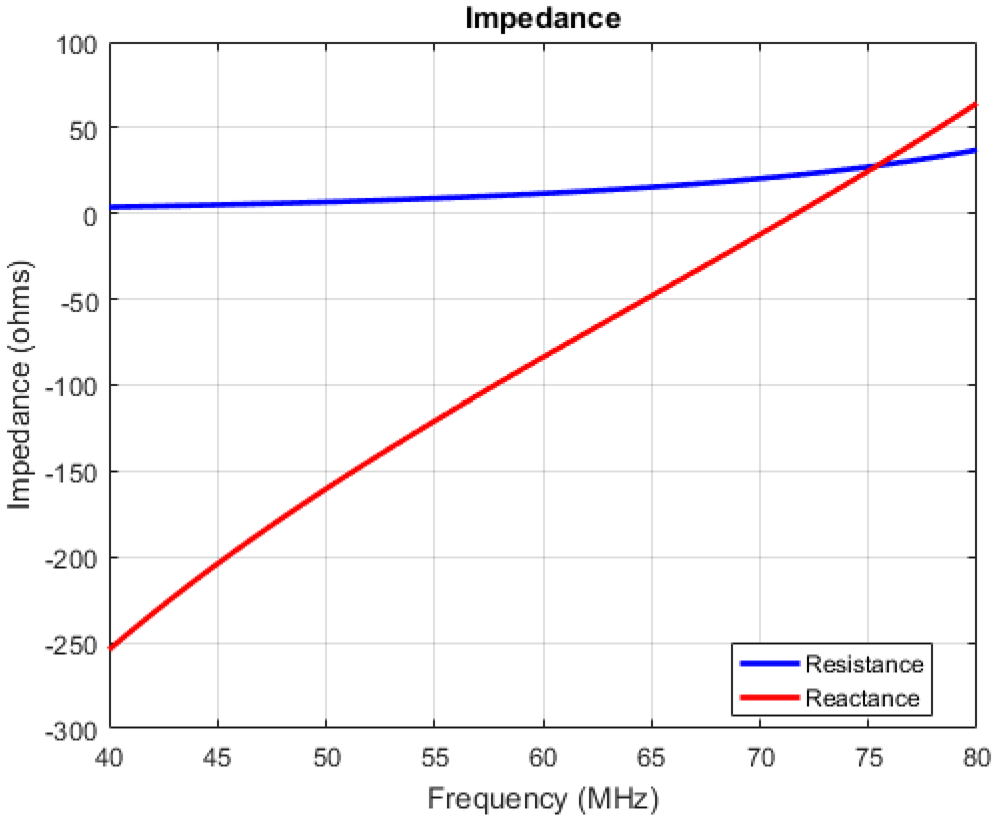
```
th = monopoleTopHat;
impedance(th,linspace(40e6,80e6,41));
```



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 | loopCircular | monopoletophat

More About

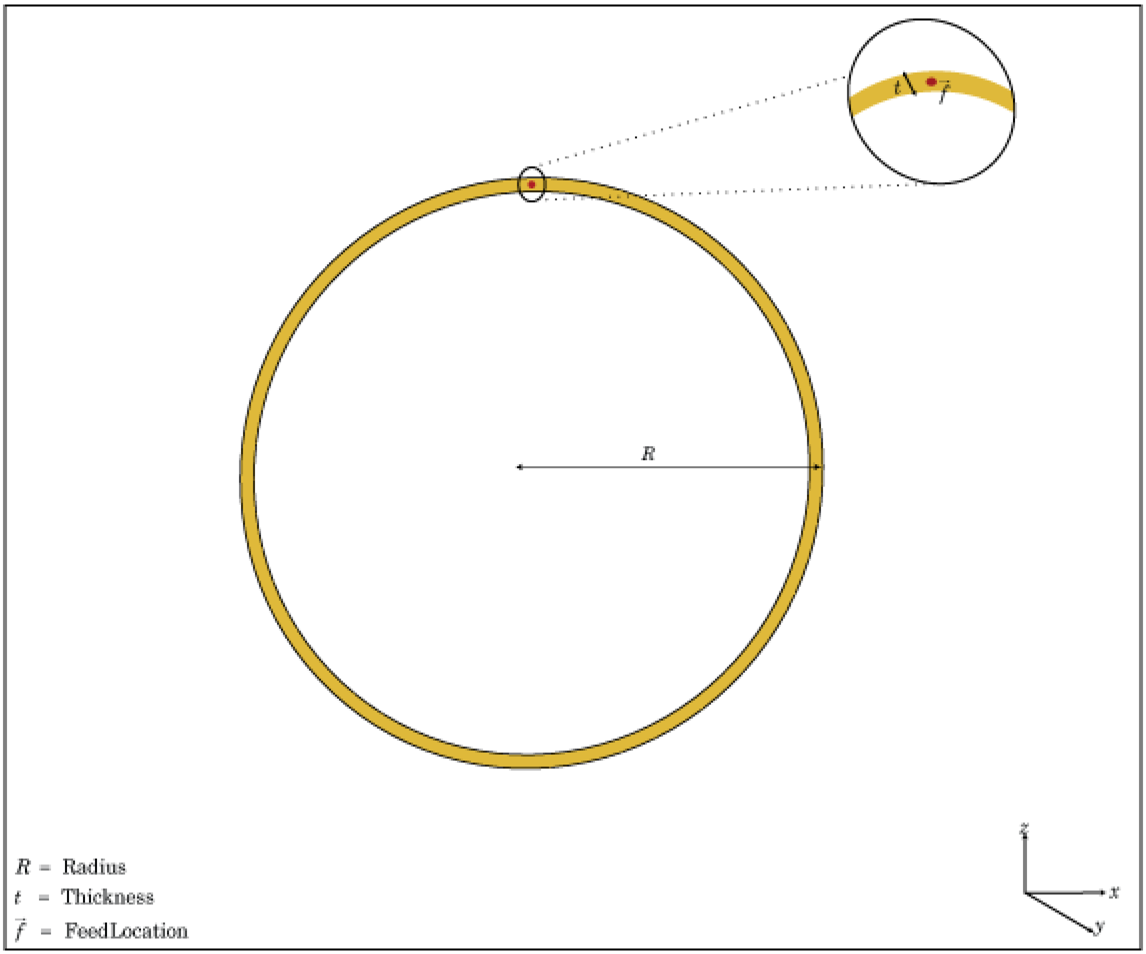
- "Rotate Antenna and Arrays"

Introduced in R2015a

loopCircular class

Create circular loop antenna

Description



The `loopCircular` class creates 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.

Construction

`h = 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.

`h = 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 in meters

Outer radius of loop, specified as the comma-separated pair consisting of `'Radius'` and a scalar in meters.

Example: `'Radius'`,3

Data Types: `double`

`'Thickness'` — Thickness of loop

0.0200 (default) | scalar in meters

Thickness of loop, specified as the comma-separated pair consisting of `'Thickness'` and a scalar in meters.

Example: `'Thickness'`,2

Data Types: `double`

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

Tilt angle of antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or vector in degrees.

Example: `'Tilt',90`

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

Data Types: `double`

`'TiltAxis'` — Tilt axis of antenna
[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,“Rotate Antenna and Arrays”

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

Example: `'TiltAxis','Z'`

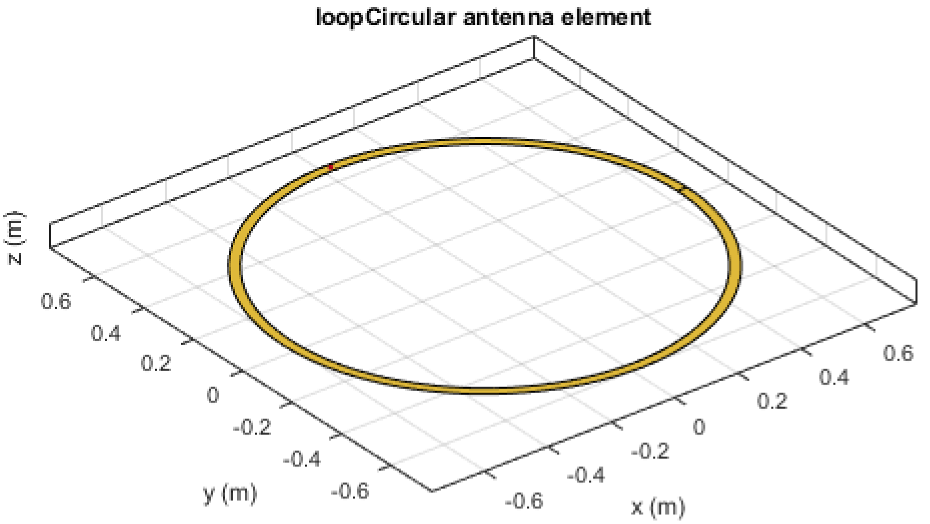
Data Types: `double`

Examples

Create and View Circular Loop Antenna

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

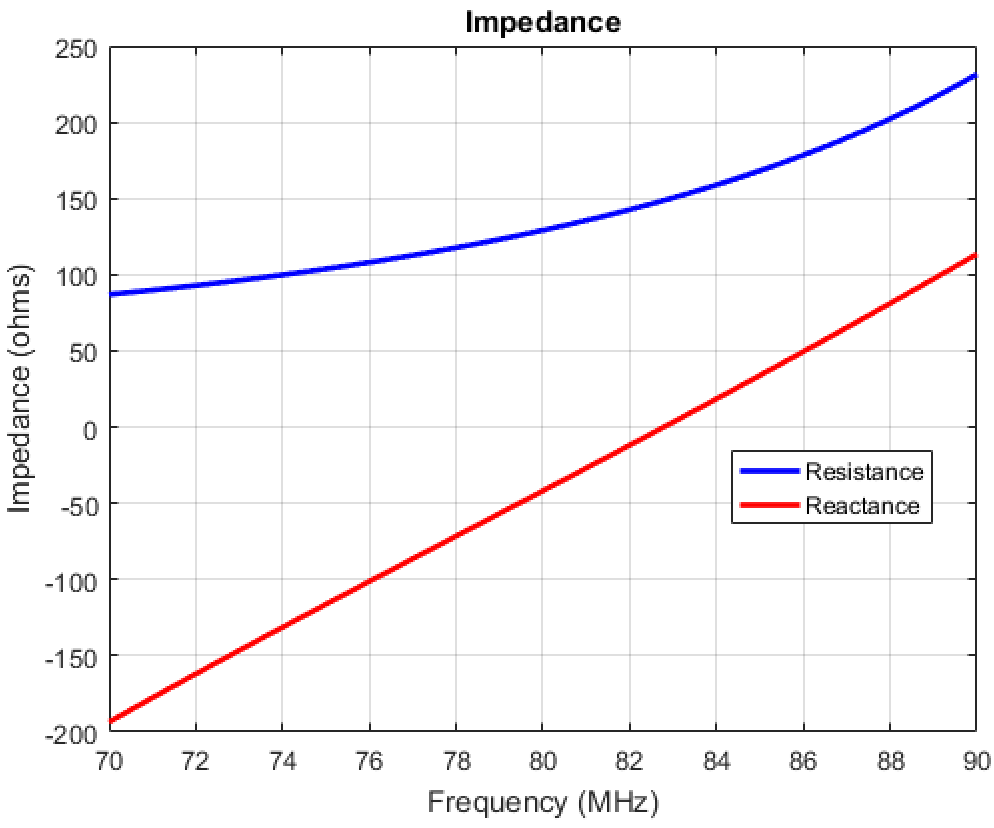
```
c = loopCircular('Radius',0.64,'Thickness',0.03);
show(c)
```



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

loopRectangular | dipole | slot

More About

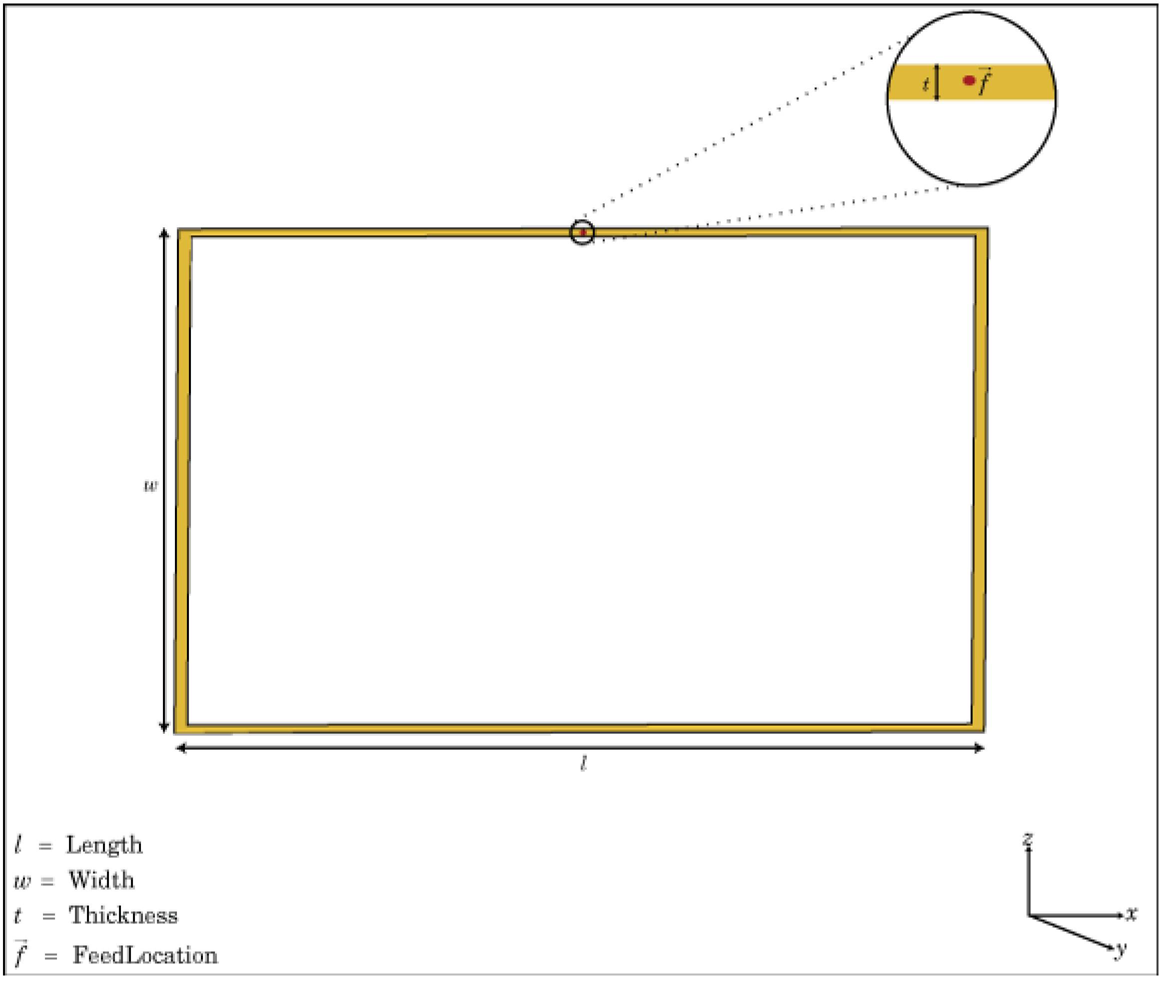
- “Rotate Antenna and Arrays”

Introduced in R2015a

loopRectangular class

Create rectangular loop antenna

Description



The `loopRectangular` class creates 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.

Construction

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

`h = 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 in meters

Loop length along x-axis, specified as the comma-separated pair consisting of `'Length'` and a scalar in meters.

Example: `'Length'`,3

Data Types: `double`

'Width' — Loop width along y-axis

1 (default) | scalar in meters

Loop width along y-axis, specified as the comma-separated pair consisting of 'Width'and a scalar in meters.

Example: 'Width' ,2

Data Types: double

'Thickness' — Loop thickness

0.0100 (default) | scalar in meters

Loop thickness, specified as the comma-separated pair consisting of 'Thickness' and a scalar in meters.

Example: 'Thickness', 2

Data Types: double

'Tilt' — Tilt angle of antenna

0 (default) | scalar in degrees | vector in degrees

Tilt angle of antenna, specified as the comma-separated pair consisting of 'Tilt' and a scalar or vector in degrees.

Example: 'Tilt',90

Example: 'Tilt',[90 90 0]

Data Types: double

'TiltAxis' — Tilt axis of antenna

[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of 'TiltAxis' and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,“Rotate Antenna and Arrays”

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

Example: `'TiltAxis','Z'`

Data Types: `double`

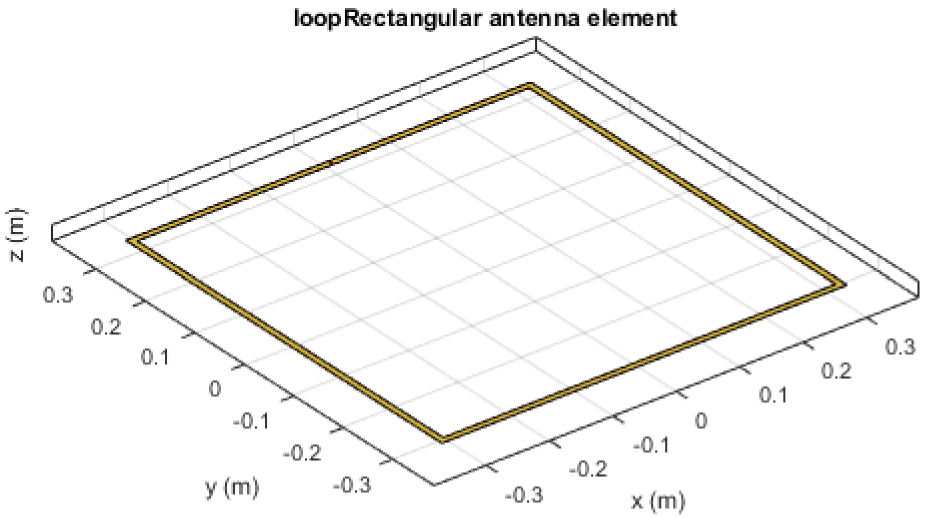
Examples

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)
show(r)
```

```
r = 

  loopRectangular with properties:

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



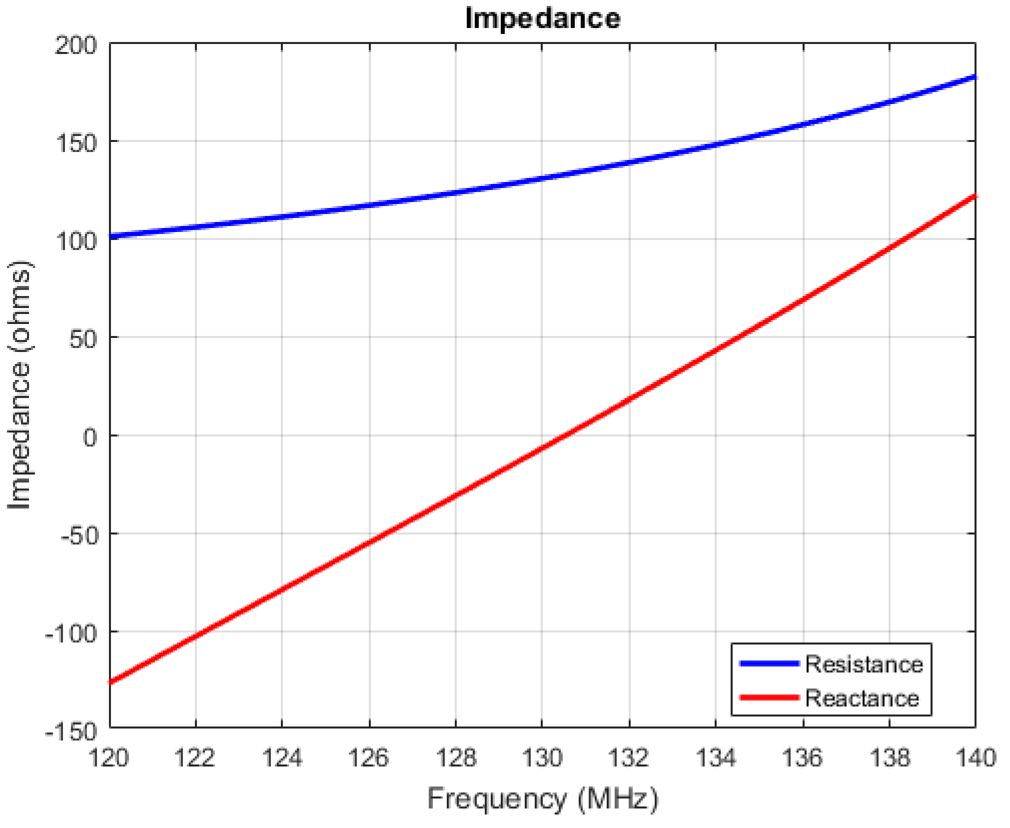
Impedance of Rectangular Loop Antenna

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

```
r = loopRectangular('Length',0.64,'Width',0.64)
impedance(r,linspace(120e6,140e6,31))
```

```
r = 

  loopRectangular with properties:

      Length: 0.6400
```

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



References

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

See Also

loopCircular | dipole | monopole | `cylinder2strip`

More About

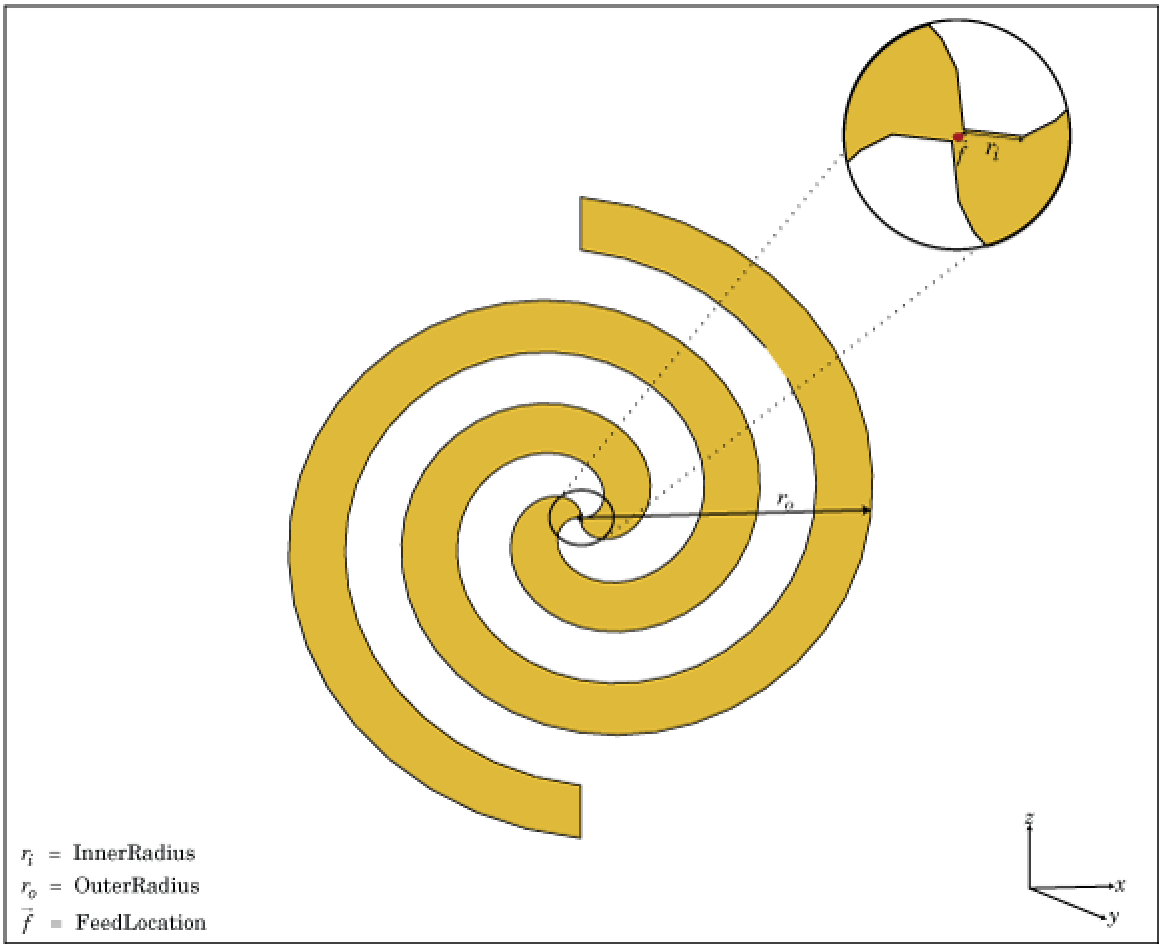
- “Rotate Antenna and Arrays”

Introduced in R2015a

spiralArchimedean class

Create Archimedean spiral antenna

Description



The `spiralArchimedean` class creates a planar Archimedean spiral antenna on the X-Y plane. The 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 radius is measured from the center. 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 located in the X-Y plane.

Construction

`sa = 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.

`sa = spiralArchimedean(Name,Value)` creates a planar Archimedean spiral, 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

`'Turns'` — Number of turns of spiral

1.5000 (default) | scalar

Number of turns of spiral, specified as the comma-separated pair consisting of `'Turns'` and a scalar.

Example: `'Turns'`,2

Data Types: `double`

'InnerRadius' — Inner radius of spiral
5.0000e-04 (default) | scalar in meters

Spiral inner radius, specified as the comma-separated pair consisting of `'InnerRadius'` and a scalar in meters.

Example: `'InnerRadius'`,1e-3

Data Types: `double`

'OuterRadius' — Outer radius of spiral
0.0398 (default) | scalar in meters

Outer radius of spiral, specified as a comma-separated pair consisting of `'OuterRadius'` and a scalar in meters.

Example: `'OuterRadius'`,1e-3

Data Types: `double`

'WindingDirection' — Direction of spiral turns (wingdings)
CW | CCW

Direction of spiral turns (wingdings), specified as the comma-separated pair consisting of `'WindingDirection'` and `CW` or `CCW`.

Example: `'WindingDirection'`,`CW`

Data Types: `string`

'Tilt' — Tilt angle of antenna
0 (default) | scalar in degrees | vector in degrees

Tilt angle of antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or vector in degrees.

Example: `'Tilt',90`

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

Data Types: `double`

'TiltAxis' — Tilt axis of antenna
[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,"Rotate Antenna and Arrays"

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

Example: `'TiltAxis','Z'`

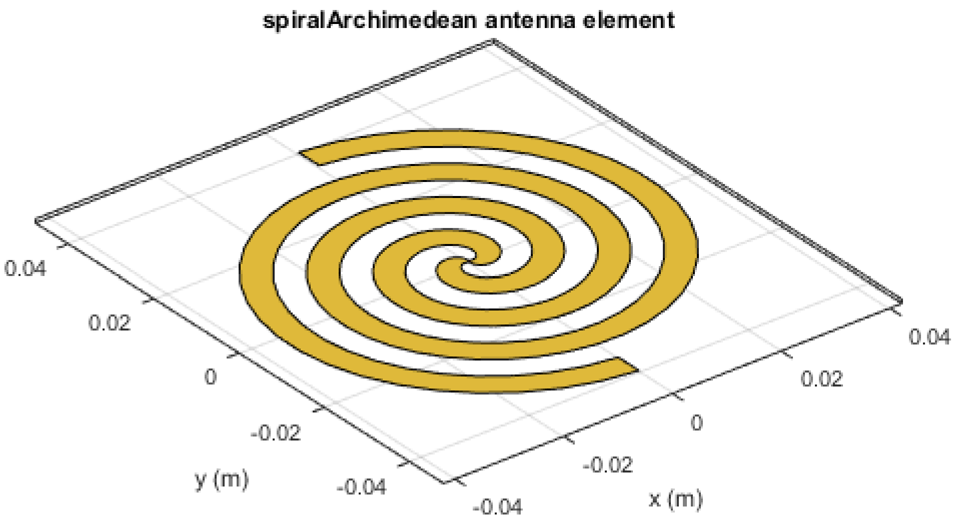
Data Types: `double`

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.

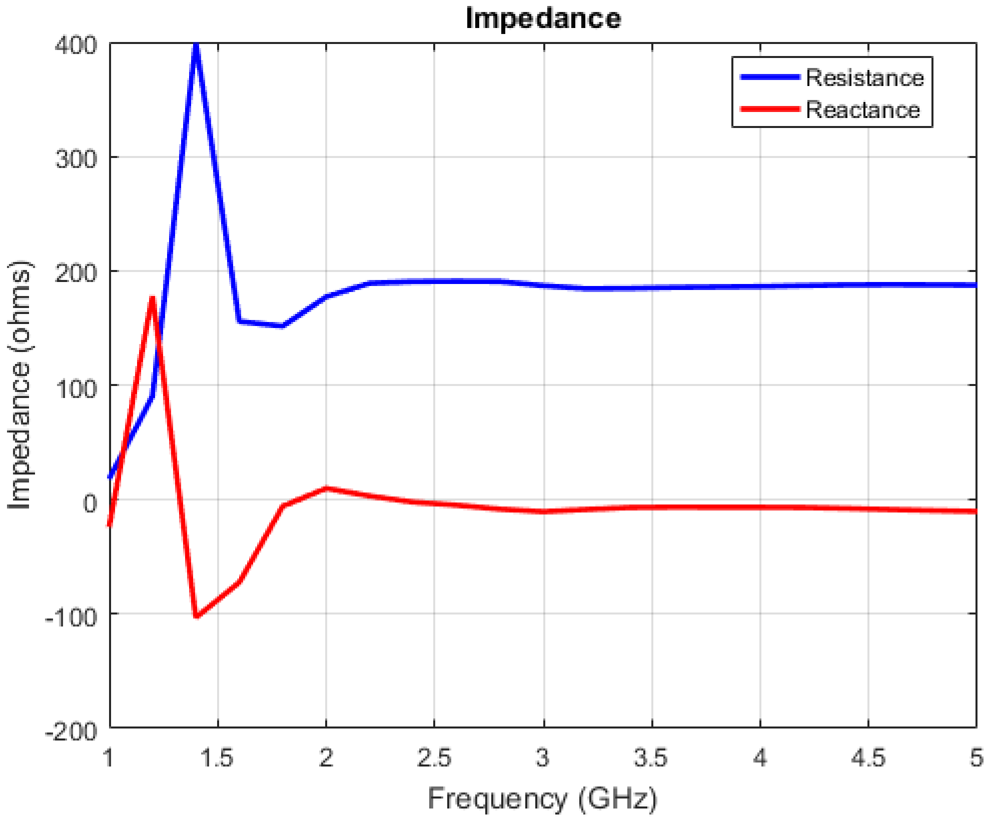
```
sa = spiralArchimedean('Turns',2, 'InnerRadius',1e-3, 'OuterRadius',40e-3);
show(sa)
```



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));
```



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

spiralEquiangular | helix | yagiUda

More About

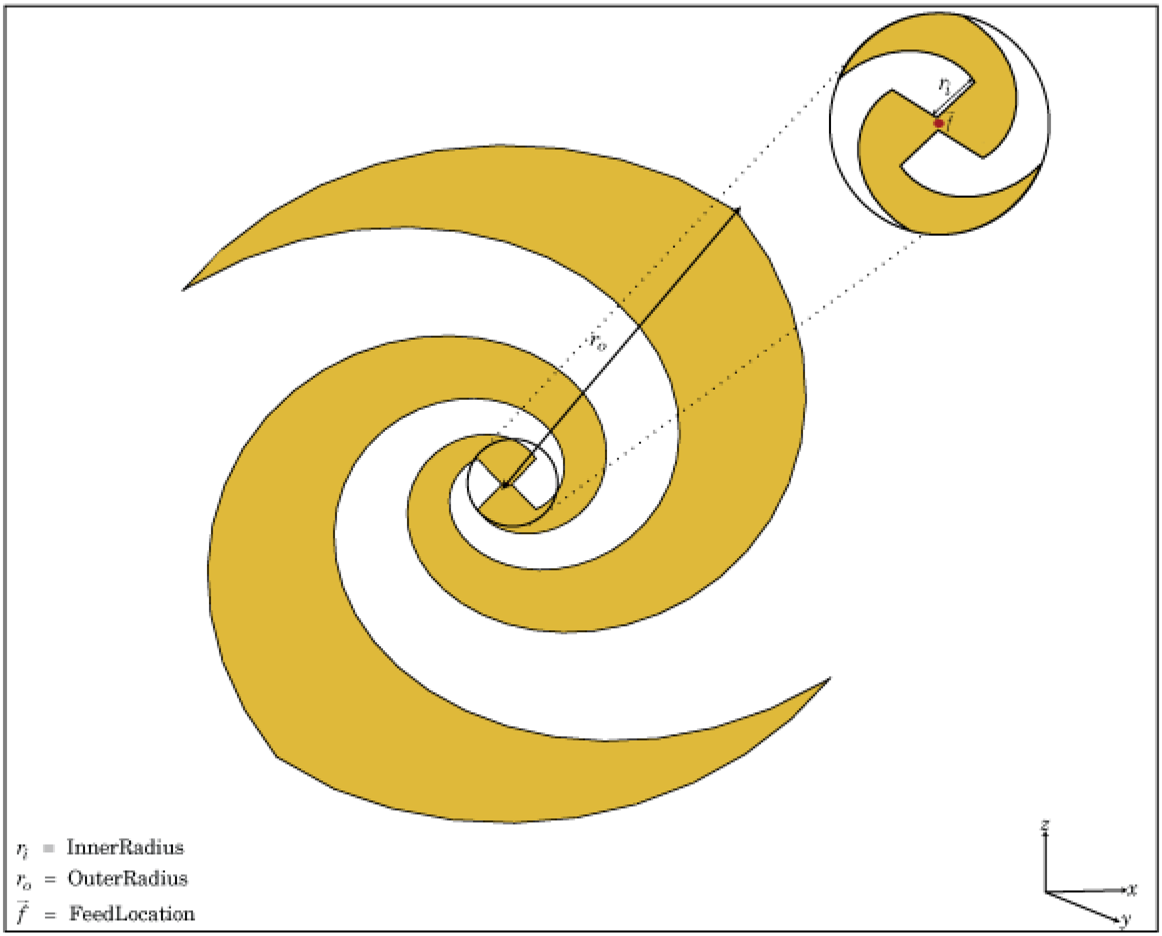
- “Rotate Antenna and Arrays”

Introduced in R2015a

spiralEquiangular class

Create equiangular spiral antenna

Description



The `spiralEquiangular` class creates 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

Construction

`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 the comma-separated pair consisting of `'GrowthRate'` and a scalar.

Example: `'GrowthRate'`,1.2

Data Types: `double`

`'InnerRadius'` — Inner radius of spiral

0.0020 (default) | scalar in meters

Inner radius of spiral , specified as the comma-separated pair consisting of `'InnerRadius'` and a scalar in meters.

Example: `'InnerRadius'`,1e-3

Data Types: `double`

`'OuterRadius'` — Outer radius of spiral

0.0189 (default) | scalar in meters

Outer radius of spiral, specified as the comma-separated pair consisting of `'OuterRadius'` and a scalar in meters.

Example: `'OuterRadius'`,1e-3

Data Types: `double`

`'WindingDirection'` — Direction of spiral turns (wingdings)

CW | CCW

Direction of spiral turns (wingdings), specified as the comma-separated pair consisting of `'WindingDirection'` and `CW` or `CCW`.

Example: `'WindingDirection'`,`CW`

Data Types: `string`

`'Tilt'` — Tilt angle of antenna

0 (default) | scalar in degrees | vector in degrees

Tilt angle of antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or vector in degrees.

Example: `'Tilt',90`

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

Data Types: `double`

`'TiltAxis'` — Tilt axis of antenna

[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of 'TiltAxis' and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,"Rotate Antenna and Arrays"

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

Example: 'TiltAxis','Z'

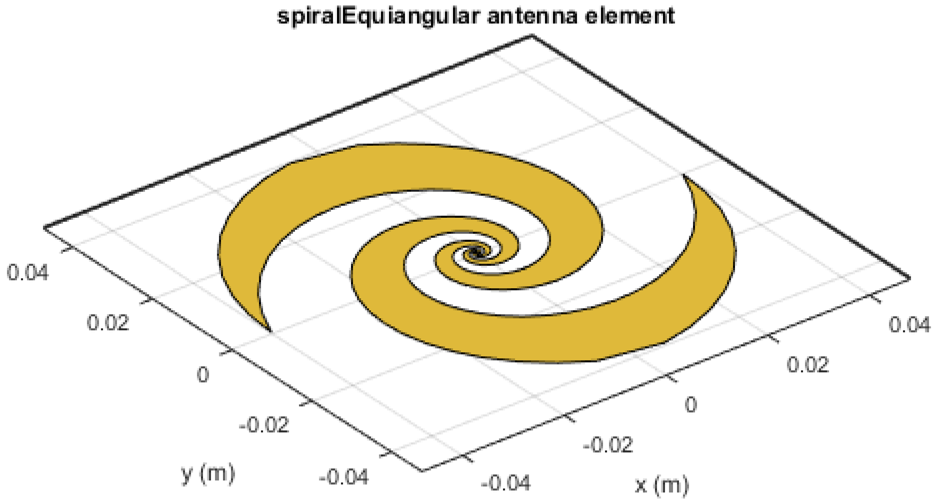
Data Types: double

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.

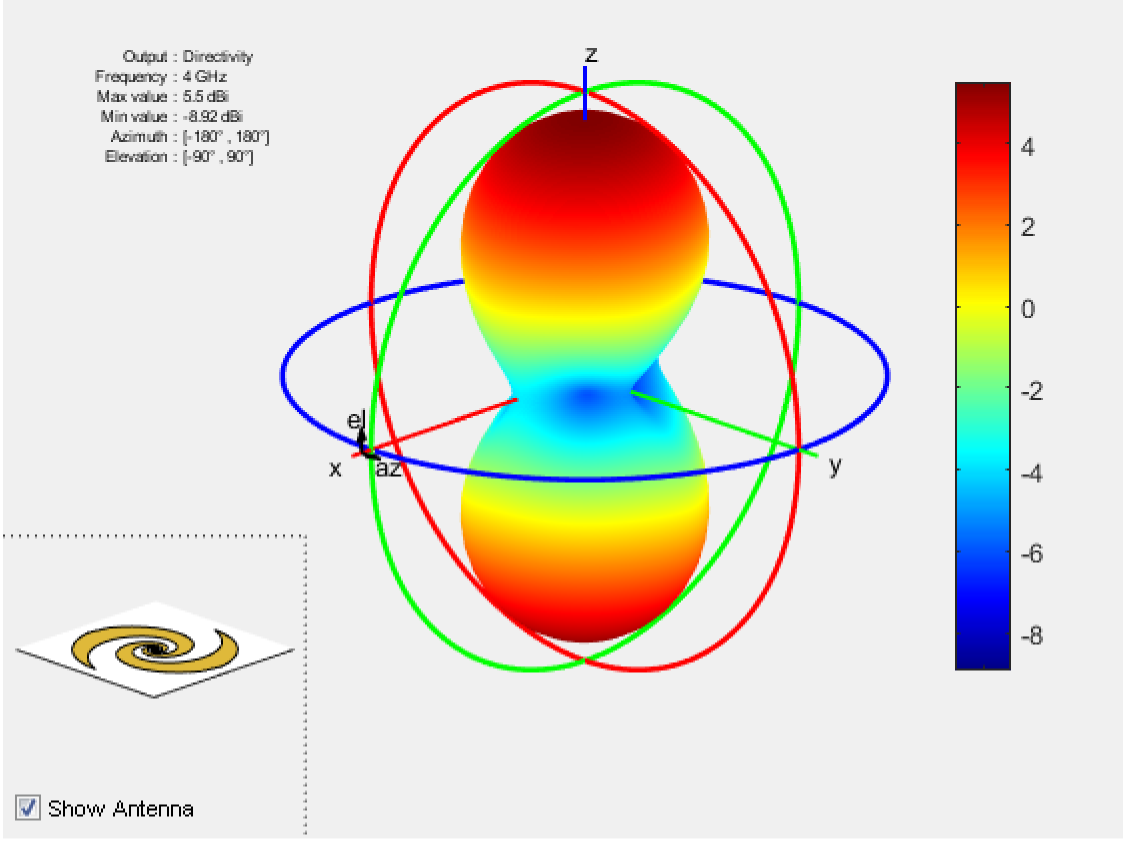
```
se = spiralEquiangular('GrowthRate',0.35, 'InnerRadius',0.65e-3,    ...
                          'OuterRadius',40e-3);
show(se)
```



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

vivaldi | cavity | spiralArchimedean

More About

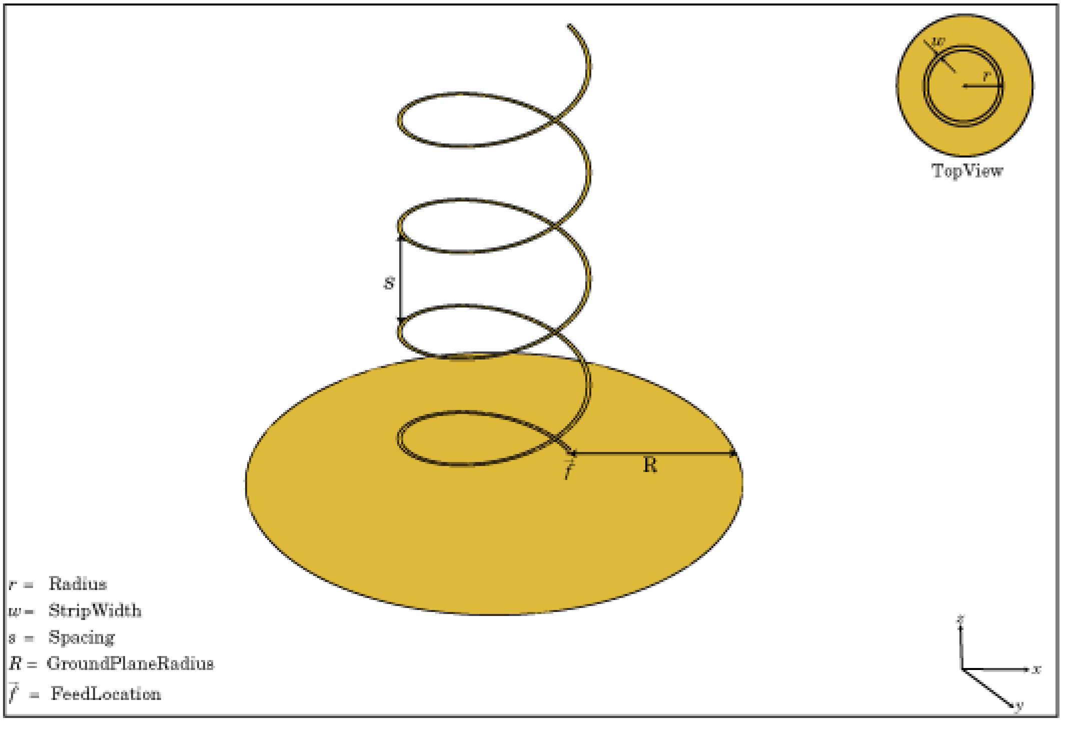
- "Rotate Antenna and Arrays"

Introduced in R2015a

helix class

Create helix antenna on ground plane

Description



The `helix` class creates 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.

Construction

`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 in meters

Turn radius, specified as the comma-separated pair consisting of `'Radius'` and a scalar in meters.

Example: `'Radius'`,2

Data Types: `double`

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

Strip width, specified as the comma-separated pair consisting of `'Width'` and 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 the comma-separated pair consisting of `'Turns'` and a scalar.

Example: `'Turns'`,2

Data Types: `double`

`'Spacing'` — Spacing between turns
0.0350 (default) | scalar in meters

Spacing between turns, specified as the comma-separated pair consisting of `'Spacing'` and a scalar in meters.

Example: `'Spacing'`,1.5

Data Types: `double`

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

Direction of helix turns (wingdings), specified as the comma-separated pair consisting of `'WindingDirection'` and `CW` or `CCW`.

Example: 'WindingDirection',CW

Data Types: string

'GroundPlaneRadius' — Ground plane radius

0.0750 (default) | scalar in meters

Ground plane radius, specified as the comma-separated pair consisting of 'GroundPlaneRadius' and 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

'Tilt' — Tilt angle of antenna

0 (default) | scalar in degrees | vector in degrees

Tilt angle of antenna, specified as the comma-separated pair consisting of 'Tilt' and a scalar or vector in degrees.

Example: 'Tilt',90

Example: 'Tilt',[90 90 0]

Data Types: double

'TiltAxis' — Tilt axis of antenna

[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of 'TiltAxis' and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,“Rotate Antenna and Arrays”

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

Example: 'TiltAxis','Z'

Data Types: double

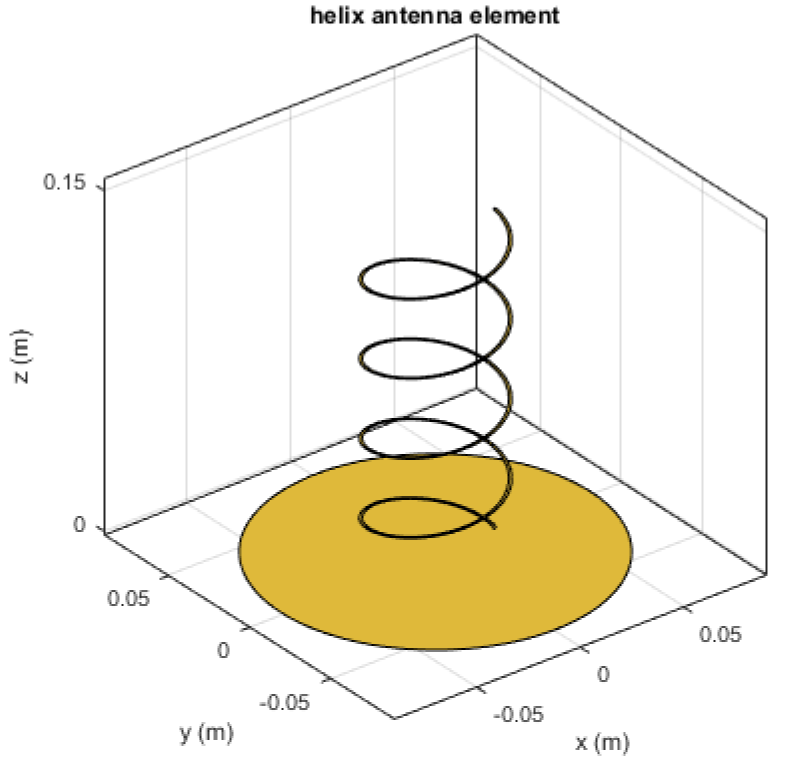
Examples

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)
show(hx)
```

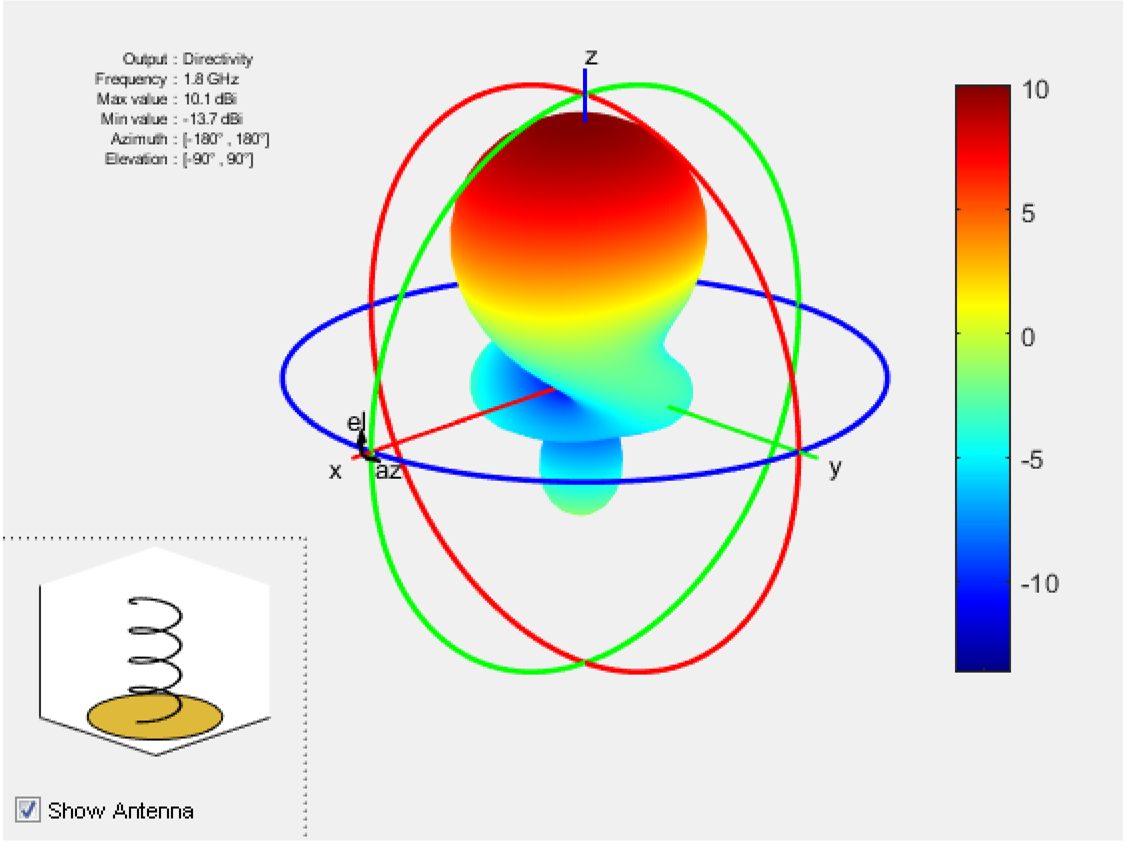
```
hx =

  helix with properties:

               Radius: 0.0280
                Width: 0.0012
                Turns: 4
              Spacing: 0.0350
     WindingDirection: 'CCW'
     GroundPlaneRadius: 0.0750
                 Tilt: 0
             TiltAxis: [1 0 0]
```



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 =

    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 Thoery (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

More About

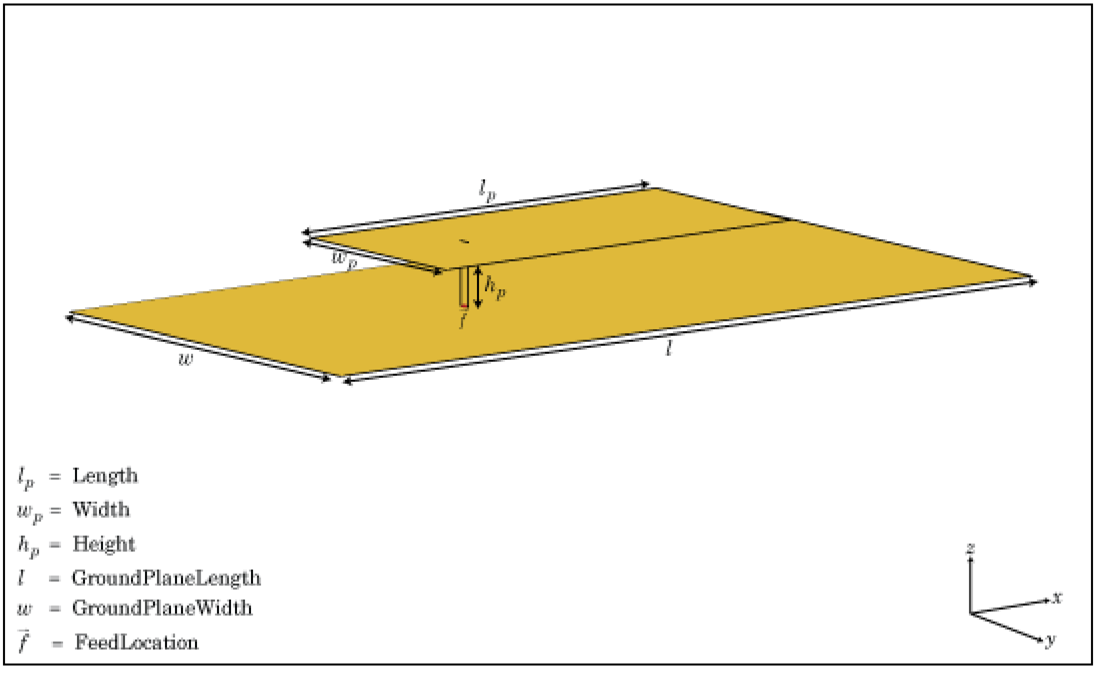
- “Rotate Antenna and Arrays”

Introduced in R2015a

patchMicrostrip class

Create microstrip patch antenna

Description



The `patchMicrostrip` class creates a microstrip patch antenna. The default patch is centered at the origin. The feed point is along the length of the antenna.

Construction

`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 in meters

Patch length, specified as the comma-separated pair consisting of `'Length'`and 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 in meters

Patch width, specified as the comma-separated pair consisting of `'Width'` and 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 in meters

Height of substrate, specified as the comma-separated pair consisting of `'Height'` and a scalar in meters.

Example: `'Height'`,37e-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 the comma-separated pair consisting of `'Substrate'` and dielectric material object handle or dielectric material from dielectric catalog. For more information refer, `dielectric`. For more information on dielectric substrate meshing, refer “Meshing”.

Note: The substrate dimensions must be lesser than the groundplane dimensions.

Example: `'Substrate','FR4'`

`'GroundPlaneLength'` — Ground plane length along x-axis

0.1500 (default) | scalar in meters

Ground plane length, specified as the comma-separated pair consisting of `'GroundPlaneLength'` and 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 in meters

Ground plane width, specified as the comma-separated pair consisting of `'GroundPlaneWidth'` and 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 in meters

Signed distance from center along length and width of ground plane, specified as the comma-separated pair consisting of `'PatchCenterOffset'` and 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 in meters

Signed distance from center along length and width of ground plane, specified as the comma-separated pair of 'FeedOffset' and 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

'Tilt' — Tilt angle of antenna
0 (default) | scalar in degrees | vector in degrees

Tilt angle of antenna, specified as the comma-separated pair consisting of 'Tilt' and a scalar or vector in degrees.

Example: 'Tilt',90

Example: 'Tilt',[90 90 0]

Data Types: double

'TiltAxis' — Tilt axis of antenna
[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of 'TiltAxis' and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,"Rotate Antenna and Arrays"

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

Example: 'TiltAxis','Z'

Data Types: double

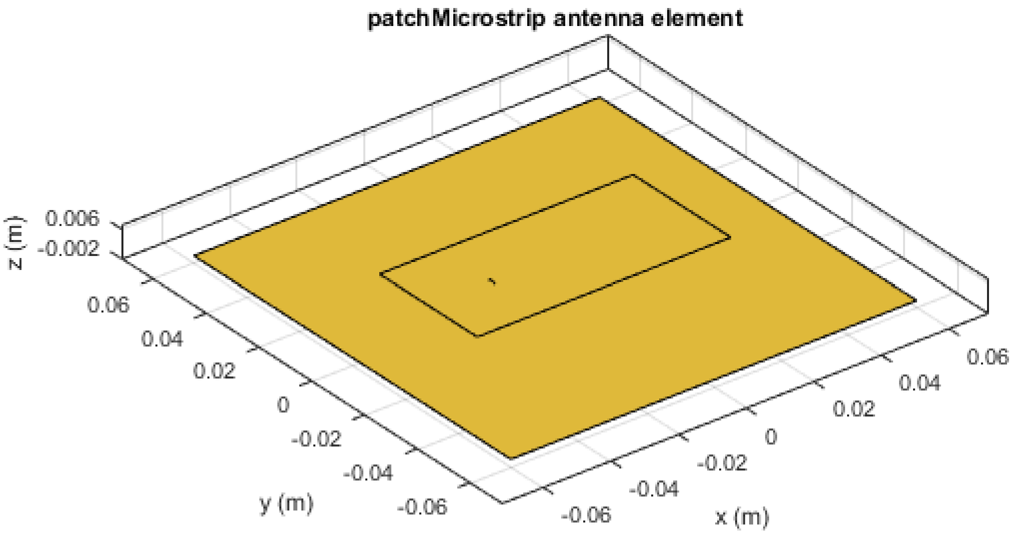
Examples

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)

show (pm)
```

```
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]
```



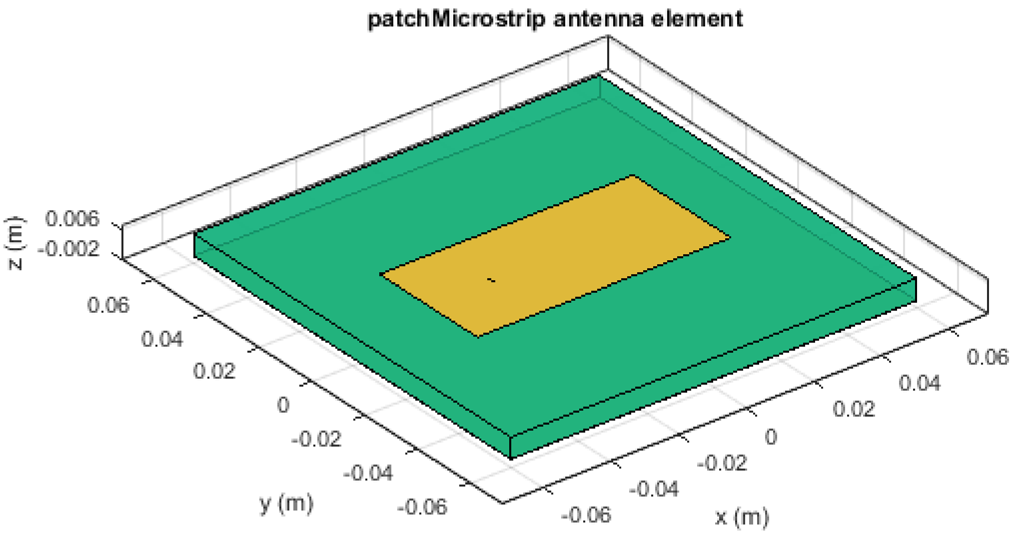
Radiation Pattern of Microstrip Patch Antenna

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)
```

```
pm =

  patchMicrostrip with properties:
```

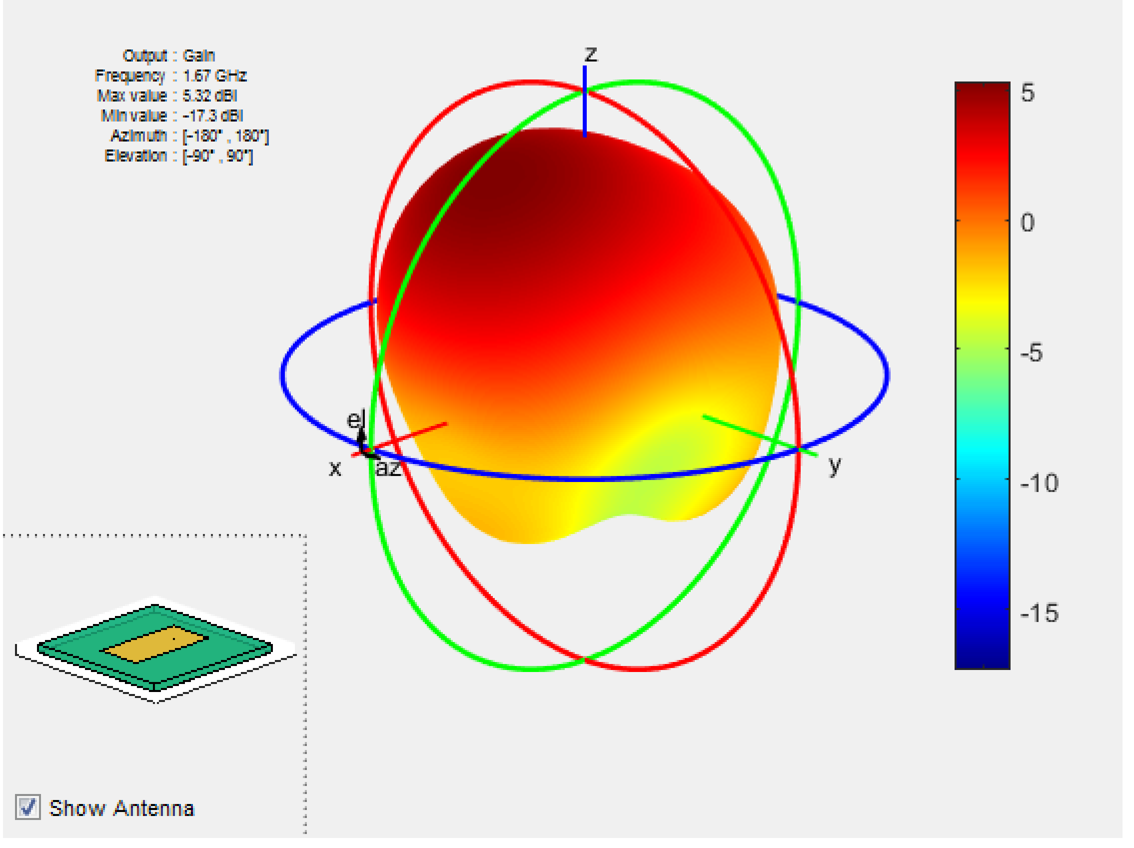
```
              Length: 0.0750
               Width: 0.0370
              Height: 0.0060
           Substrate: [1x1 dielectric]
   GroundPlaneLength: 0.1200
    GroundPlaneWidth: 0.1200
   PatchCenterOffset: [0 0]
          FeedOffset: [-0.0187 0]
                Tilt: 0
            TiltAxis: [1 0 0]
```



Plot the radiation pattern of the antenna at a frequency of 1.67 GHz.

```
figure
```

```
pattern(pm,1.67e9)
```



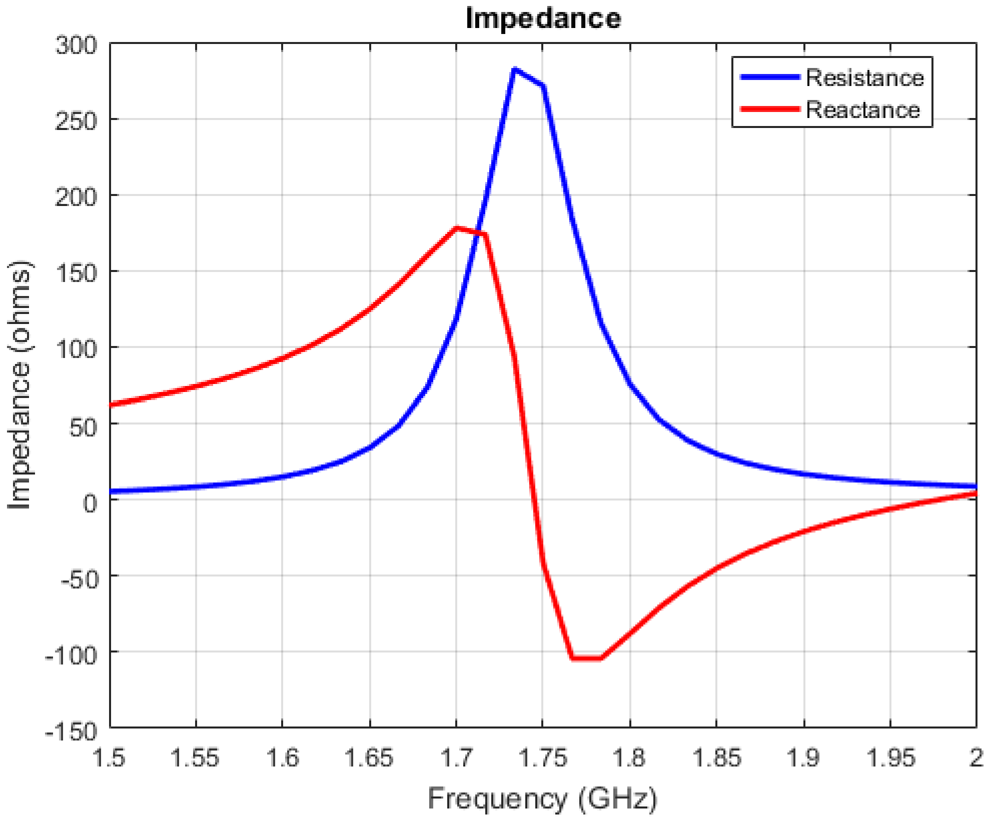
Impedance of Microstrip Patch Antenna

Calculate and plot the impedance of a microstrip patch antenna over a frequency range of 1.5-2 GHz.

```
pm = patchMicrostrip
impedance(pm,linspace(1.5e9,2e9,31));
```

```
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]
```



References

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

See Also

vivaldi | yagiUda | pifa

More About

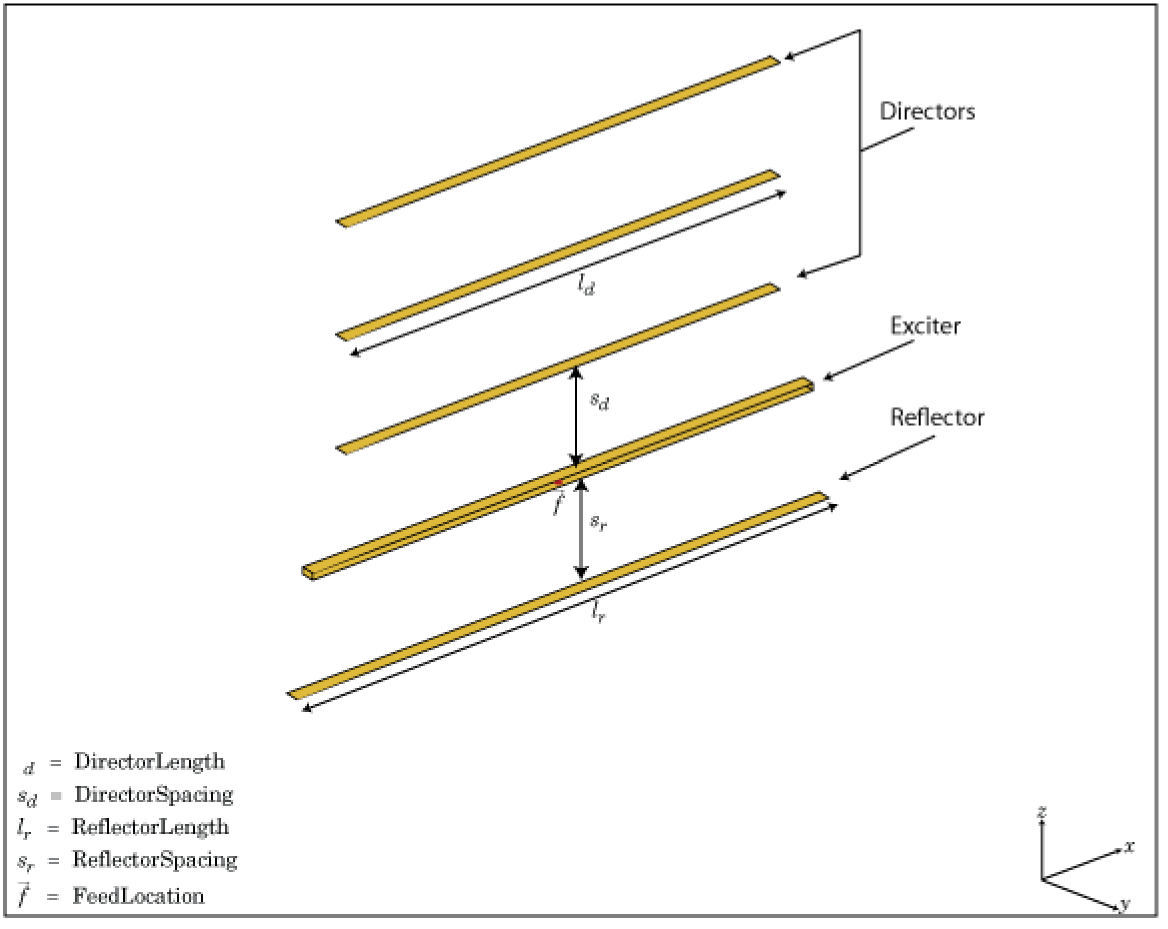
- "Rotate Antenna and Arrays"

Introduced in R2015a

yagiUda class

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.

Construction

`h = 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.

`h = 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 antenna element handle or antenna element.

Example: `'Exciter'`,dipole

`'NumDirectors'` — Total number of director elements
3 (default) | scalar

Total number of director elements, specified as the comma-separated pair consisting of `'NumDirectors'` and 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 in meters | vector in meters

Director length, specified as the comma-separated pair consisting of `'DirectorLength'` and a scalar or vector in meters.

Example: `'DirectorLength'`,[0.4 0.5]

Data Types: `double`

`'DirectorSpacing'` — Spacing between directors
0.3400 (default) | scalar in meters | vector in meters

Spacing between directors, specified as the comma-separated pair consisting of `'DirectorSpacing'` and a scalar or vector in meters.

Example: `'DirectorSpacing'`,[0.4 0.5]

Data Types: `double`

`'ReflectorLength'` — Reflector length
0.5000 (default) | scalar in meters

Reflector length, specified as the comma-separated pair consisting of `'ReflectorLength'` and a scalar in meters.

Example: `'ReflectorLength'`,0.3

Data Types: `double`

`'ReflectorSpacing'` — Spacing between exciter and reflector
0.2500 (default) | scalar in meters

Spacing between exciter and reflector, specified as the comma-separated pair consisting of `'ReflectorSpacing'` and a scalar in meters.

Example: 'ReflectorSpacing', 0.4

Data Types: `double`

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

Tilt angle of antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or vector in degrees.

Example: `'Tilt',90`

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

Data Types: `double`

`'TiltAxis'` — Tilt axis of antenna
[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,"Rotate Antenna and Arrays"

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

Example: `'TiltAxis','Z'`

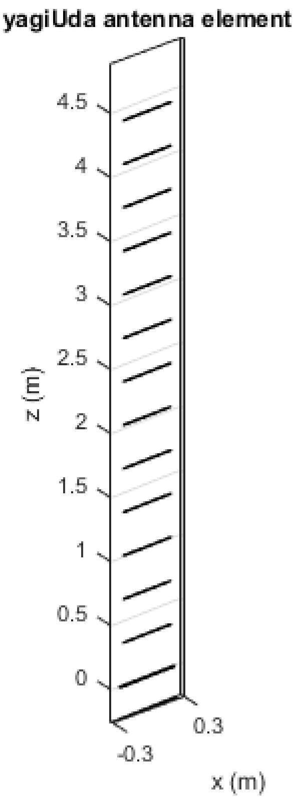
Data Types: `double`

Examples

Create and View Yagi-Uda Array Antenna

Create and view a Yagi-Uda array antenna with 13 directors.

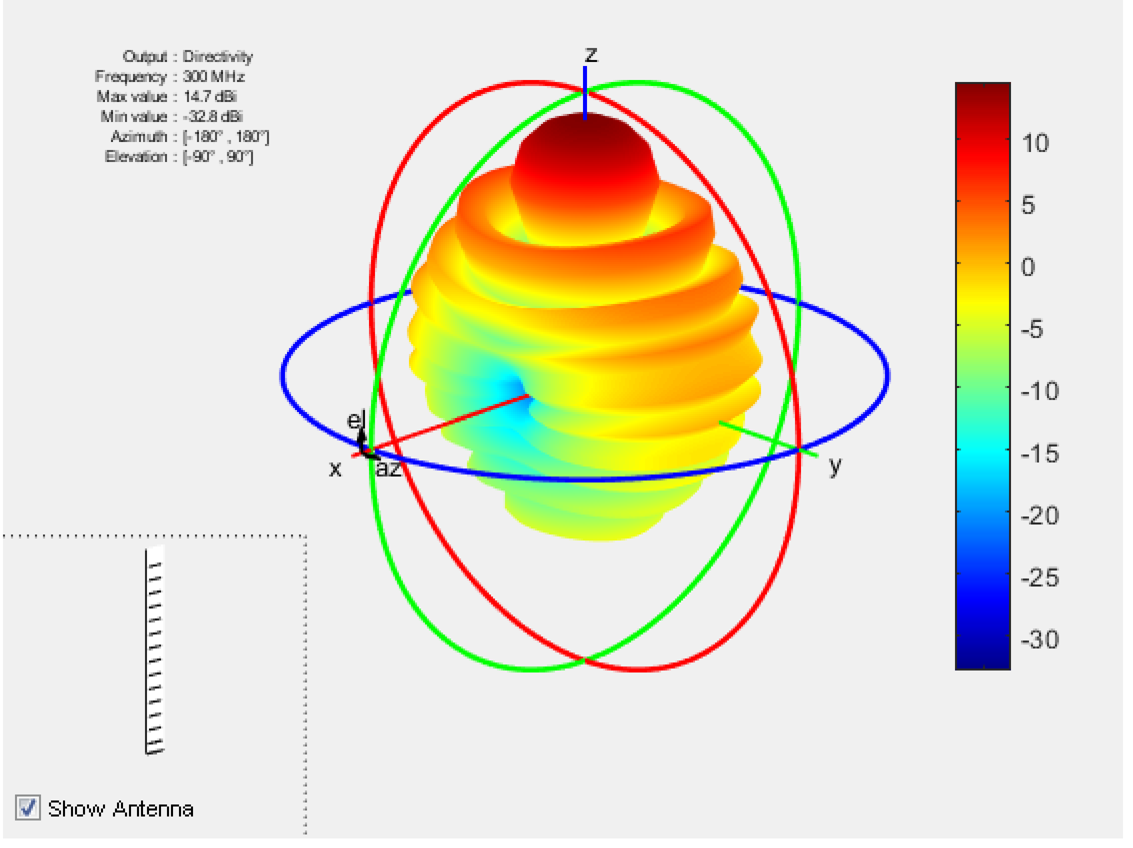
```
y = yagiUda('NumDirectors',13);
show(y)
```



Radiation Pattern of Yagi-Uda Array Antenna

Plot 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

dipoleFolded | slot | `cylinder2strip` | dipole

More About

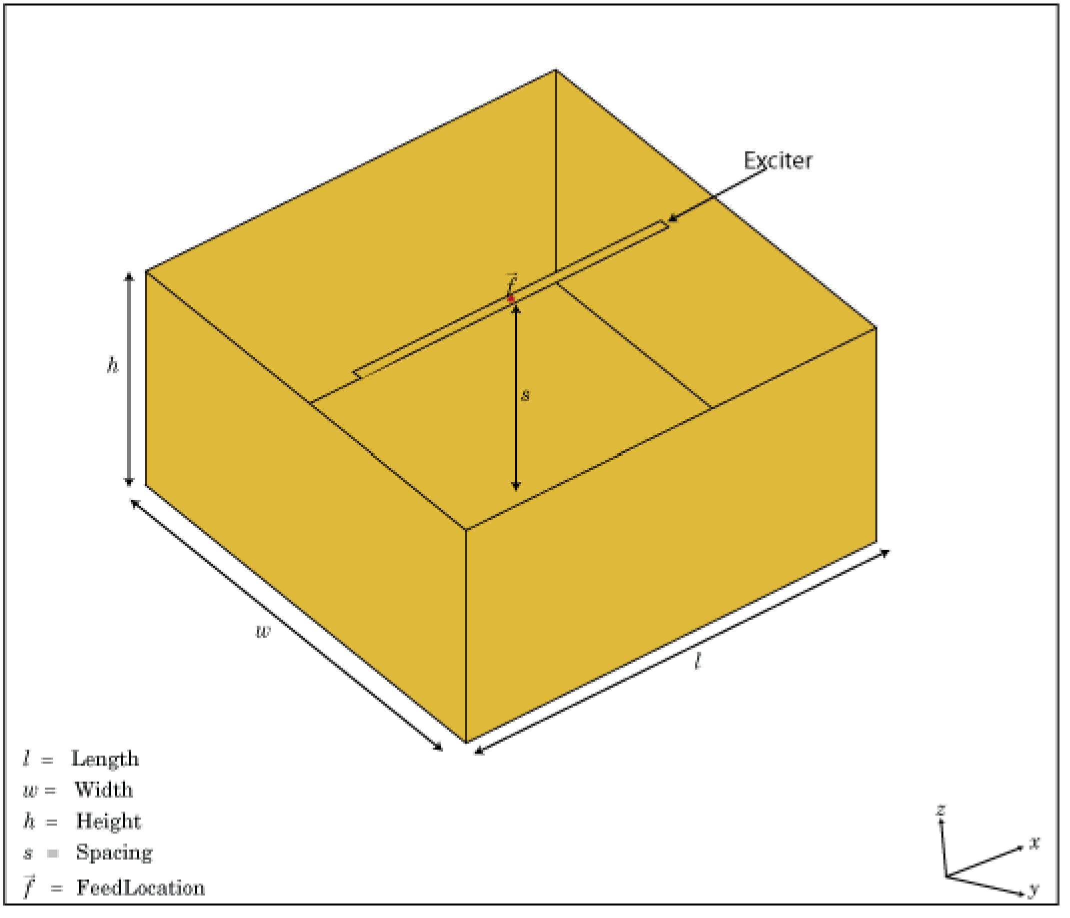
- “Rotate Antenna and Arrays”

Introduced in R2015a

cavity class

Create cavity-backed antenna

Description



The `cavity` class creates 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.

Construction

`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) | antenna element handle or antenna element

Antenna type used as an exciter, specified as the comma-separated pair consisting of `'Exciter'` and an antenna element handle or antenna element. 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) | dielectric material object handle | dielectric material from dielectric catalog

Type of dielectric material used as a substrate, specified as the comma-separated pair consisting of `'Substrate'` and dielectric material object handle or dielectric material from dielectric catalog. For more information refer, `dielectric`. For more information on dielectric substrate meshing, refer "Meshing".

Example: `'Substrate','FR4'`

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

0.2000 (default) | scalar in meters

Length of the rectangular cavity along the x-axis, specified as the comma-separated pair consisting of `'Length'` and a scalar in meters.

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

Data Types: `double`

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

0.2000 (default) | scalar in meters

Width of the rectangular cavity along the y-axis, specified as the comma-separated pair consisting of `'Width'` and a scalar in meters.

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

Data Types: `double`

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

0.0750 (default) | scalar in meters

Height of the rectangular cavity along the z-axis, specified as the comma-separated pair consisting of `'Height'` and a scalar in meters.

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

Data Types: `double`

`Spacing` — Distance between exciter and base of cavity

0.0750 (default) | scalar in meters

Distance between the exciter and the base of the cavity, specified as the comma-separated pair consisting of `'Spacing'` and a scalar in meters.

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

Data Types: `double`

`Tilt` — Tilt angle of antenna

0 (default) | scalar in degrees | vector in degrees

Tilt angle of the antenna, specified as a comma-separated pair consisting of `'Tilt'` and a scalar or a in degrees. To specify multiple `Tilt` angles to an antenna, there must be multiple `TiltAxis` three-element vectors.

Example: `'Tilt',90`

Example: `'Tilt',[90 90]` . For this example of `Tilt` angle, the `TiltAxis` is two points in space as three-element vectors.

Data Types: `double`

'TiltAxis' — Tilt axis of antenna
[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of 'TiltAxis' and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,“Rotate Antenna and Arrays”

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

Example: 'TiltAxis','Z'

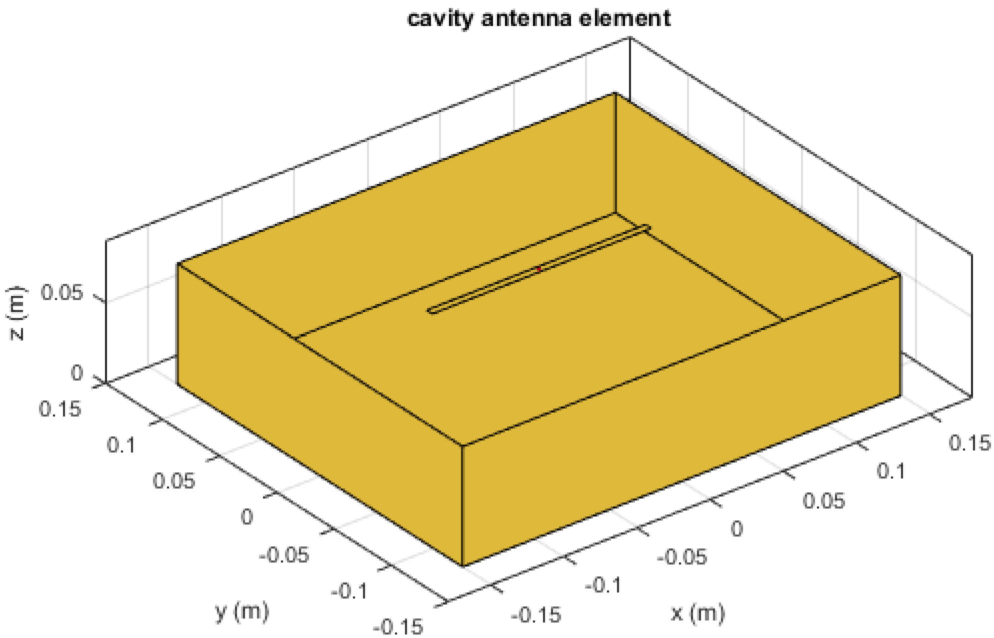
Data Types: double

Examples

Create and View Cavity-Backed Antenna.

Create and view a cavity-backed dipole antenna with 30cm length, 25cm width, 7.5cm heigth 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)
```



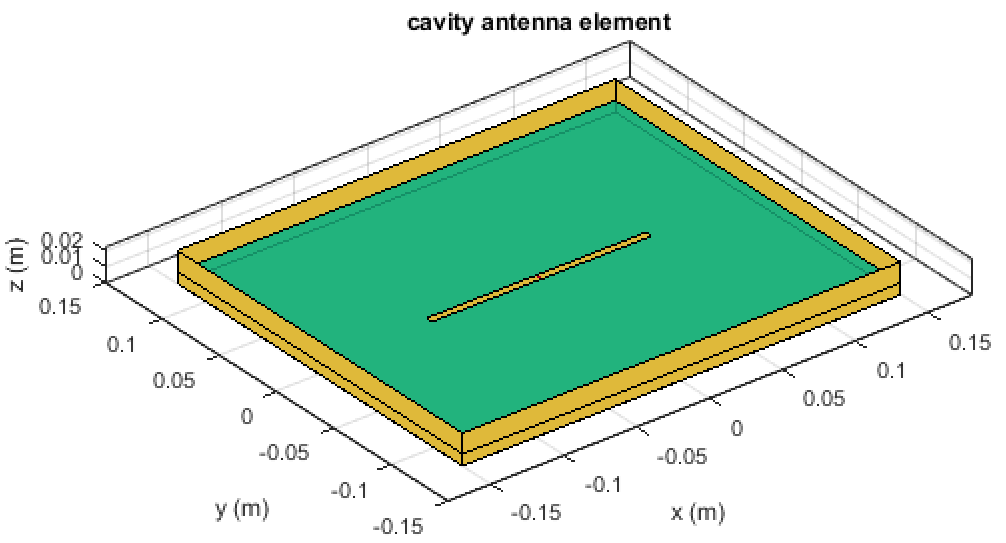
Radiation Pattern of Cavity-Backed Antenna

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)
show(c)
```

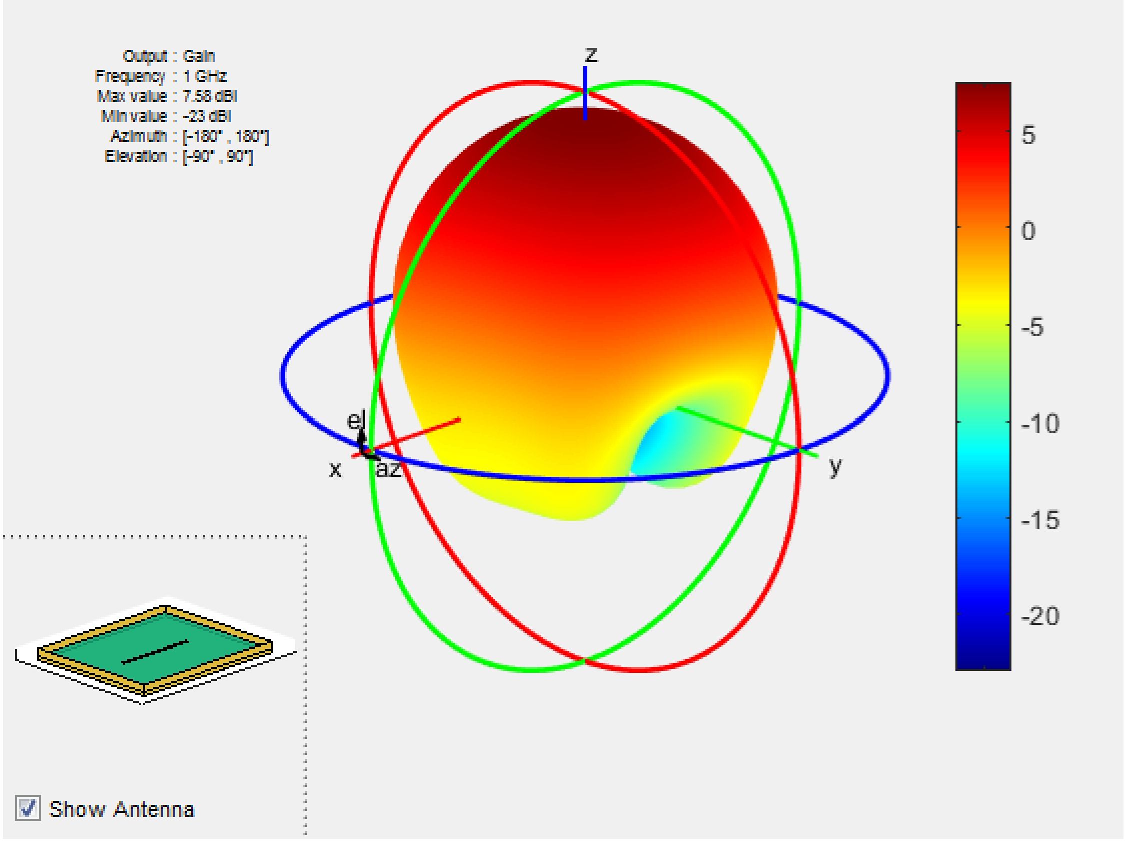
```
c = 

  cavity with properties:
```

```
     Exciter: [1x1 dipole]
   Substrate: [1x1 dielectric]
      Length: 0.3000
       Width: 0.2500
      Height: 0.0205
     Spacing: 0.0075
        Tilt: 0
    TiltAxis: [1 0 0]
```



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

spiralArchimedean | reflector | spiralEquiangular

More About

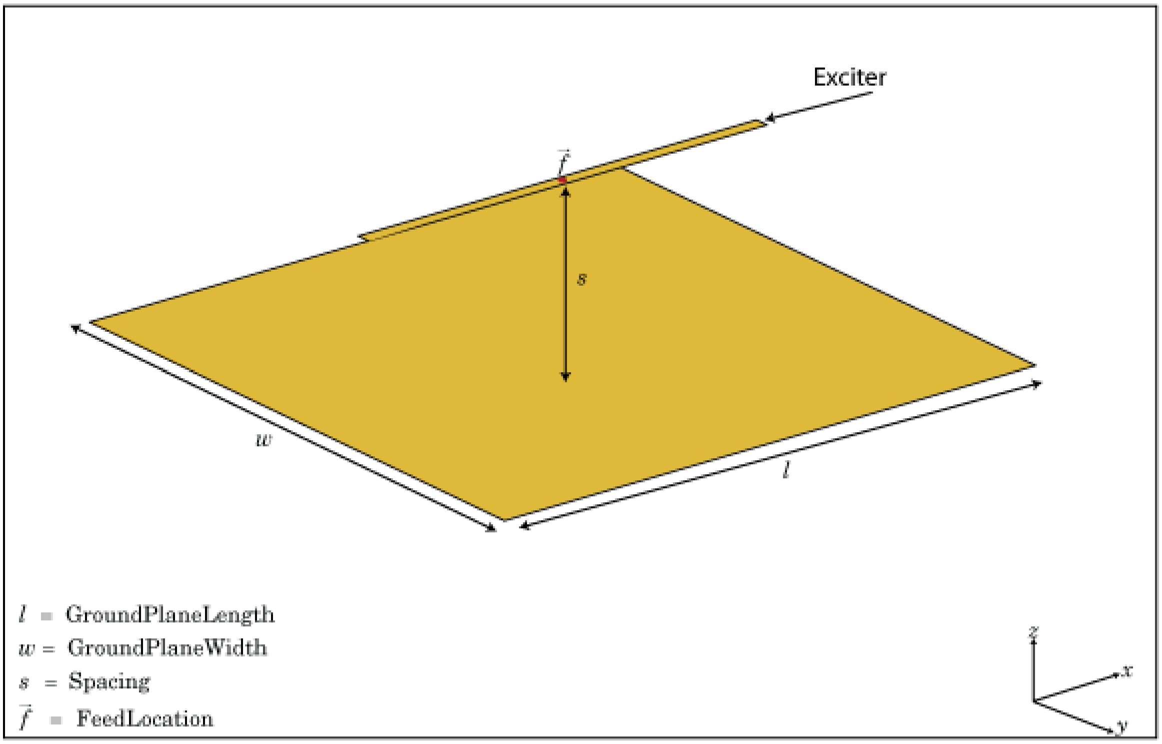
- "Rotate Antenna and Arrays"

Introduced in R2015a

reflector class

Create reflector-backed antenna

Description



The `reflector` class creates a reflector-backed antenna located on the X-Y-Z plane. The default reflector antenna uses a dipole as an exciter. The feed point is on the exciter.

Construction

`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) | antenna element handle or antenna element

Antenna type used as an exciter, specified as the comma-separated pair consisting of `'Exciter'` and an antenna element handle or antenna element. 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) | dielectric material object handle | dielectric material from dielectric catalog

Type of dielectric material used as a substrate, specified as the comma-separated pair consisting of `'Substrate'` and dielectric material object handle or dielectric material from dielectric catalog. For more information refer, `dielectric`. For more information on dielectric substrate meshing, refer "Meshing".

Note: The substrate dimensions must be lesser than the groundplane dimensions.

Example: `'Substrate','FR4'`

`'GroundPlaneLength'` — Reflector length along x-axis

0.2000 (default) | scalar in meters

Reflector length along the x-axis, specified as the comma-separated pair consisting of `'GroundPlaneLength'` and 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'` — Reflector width along y-axis
0.2000 (default) | scalar in meters

Reflector width along the y-axis, specified as the comma-separated pair consisting of `'GroundPlaneWidth'` and 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`

`'Spacing'` — Distance between reflector and exciter
0.0750 (default) | scalar in meters

Distance between the reflector and the exciter, specified as the comma-separated pair consisting of `'Spacing’` and a scalar in meters. By default, the exciter is placed along the x-axis.

Example: `'Spacing’`,7.5e-2

Data Types: `double`

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

Tilt angle of the antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or a vector in degrees.

Example: `'Tilt'`,90

Data Types: `double`

`'TiltAxis'` — Tilt axis of antenna
[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the array, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,“Rotate Antenna and Arrays”

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

Example: `'TiltAxis','Z'`

Data Types: `double`

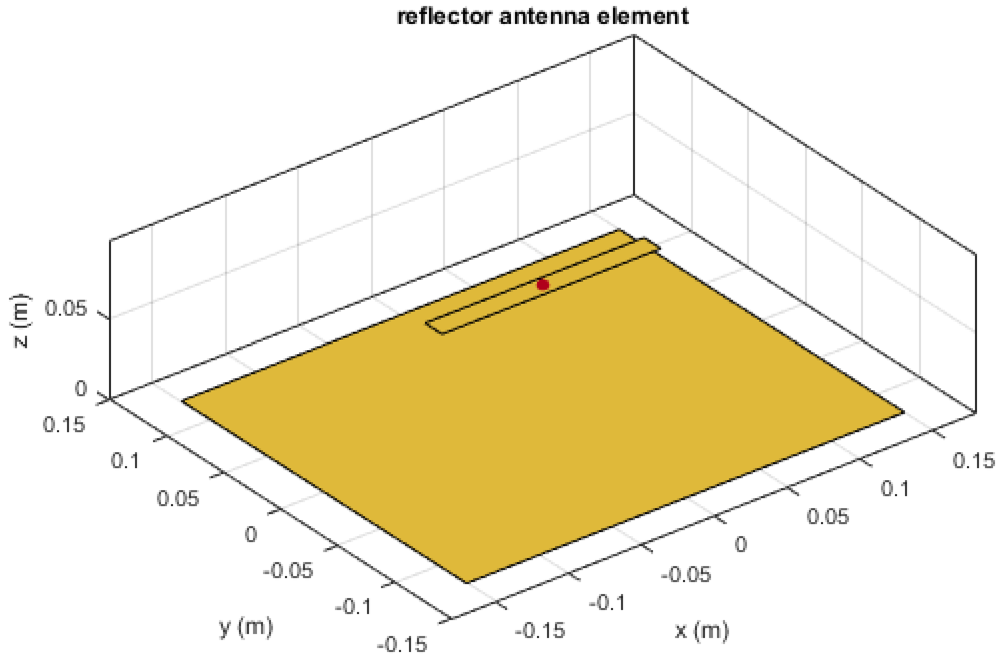
Examples

Create and View Reflector-Backed Dipole Antennna

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
show(rf)
```

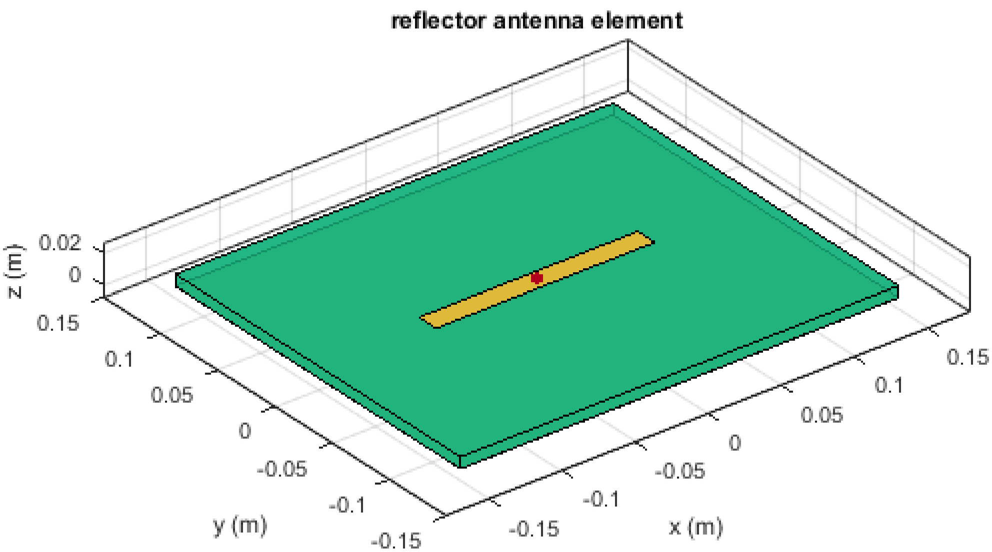
```
rf = 

  reflector with properties:

              Exciter: [1x1 dipole]
            Substrate: [1x1 dielectric]
    GroundPlaneLength: 0.3000
     GroundPlaneWidth: 0.2500
              Spacing: 0.0750
                 Tilt: 0
             TiltAxis: [1 0 0]
```



Radiation Pattern of Reflector Backed Antenna

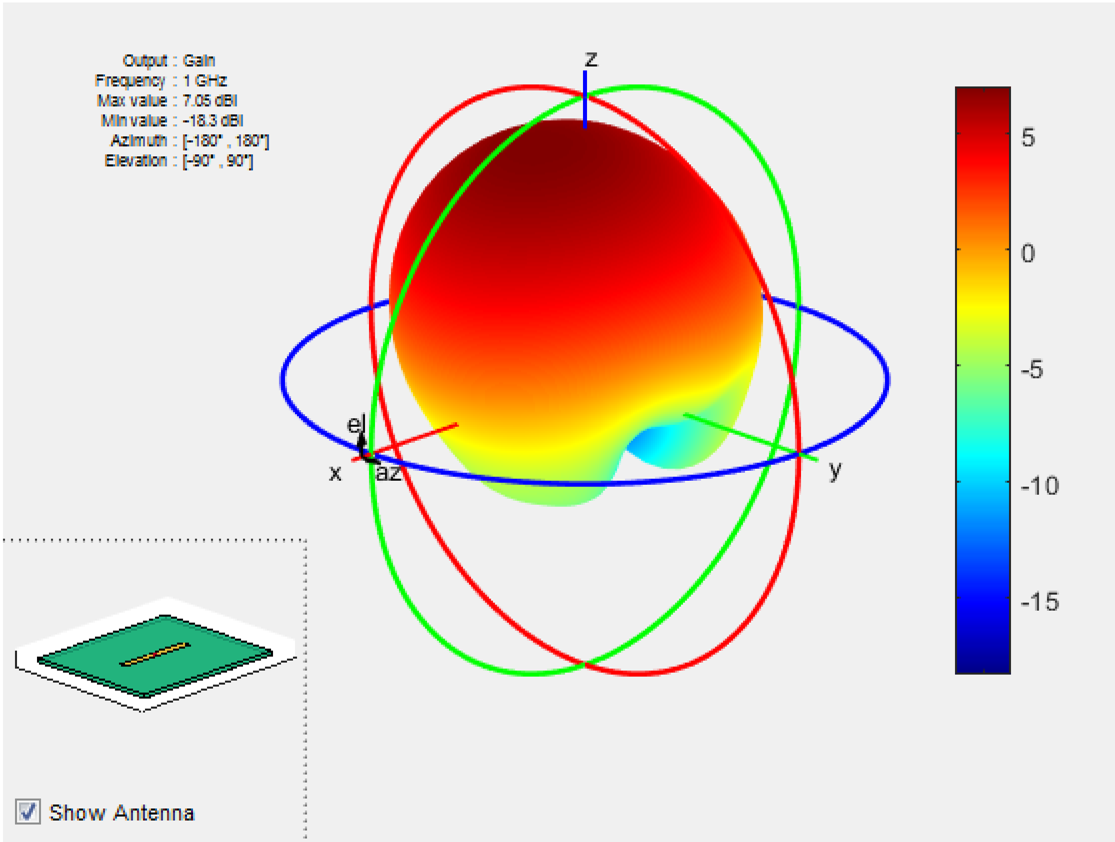
Create a reflector backed dipole antenna using a dielectric substrate **'FR4'**.

```
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 antena at a frequency of 1 GHz.

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



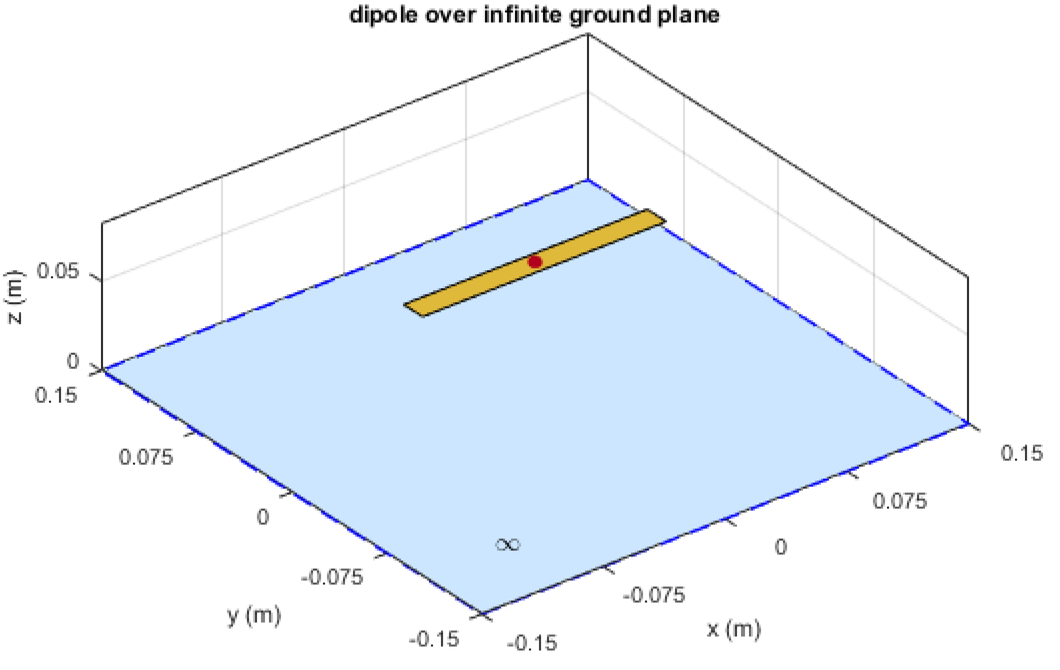
Create Reflector-Backed Antennna Over Infinite Ground Plane

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
show(rf)
```

```
rf =
```

```
reflector with properties:

            Exciter: [1x1 dipole]
          Substrate: [1x1 dielectric]
  GroundPlaneLength: Inf
   GroundPlaneWidth: 0.2500
            Spacing: 0.0750
               Tilt: 0
           TiltAxis: [1 0 0]
```



References

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

See Also

spiralArchimedean | cavity | spiralEquiangular

More About

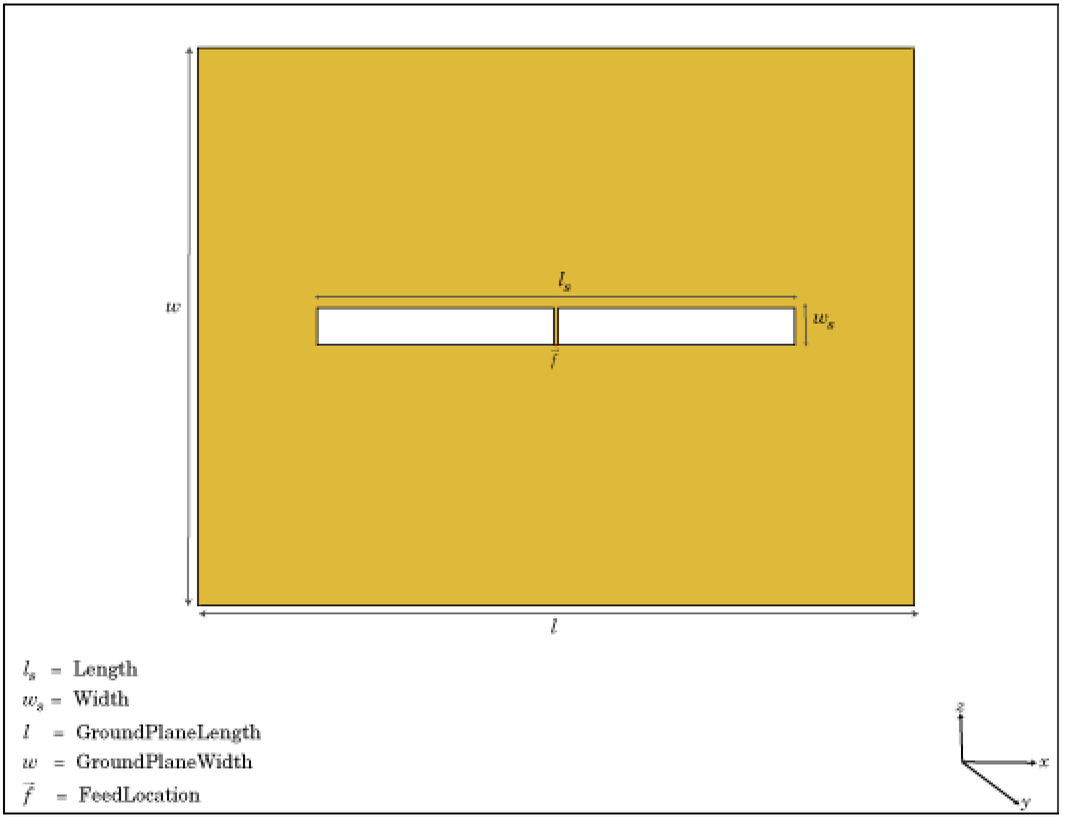
- "Rotate Antenna and Arrays"

Introduced in R2015a

slot class

Create rectangular slot antenna on ground plane

Description



The `slot` class creates a rectangular slot antenna on a ground plane. The default slot has its first resonance at 130 MHz.

Construction

`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 in meters

Slot length, specified as the comma-separated pair consisting of `'Length'` and a scalar in meters.

Example: `'Length'`,2

Data Types: `double`

`'Width'` — Slot width

0.1000 (default) | scalar in meters

Slot width, specified as the comma-separated pair consisting of `'Width'` and a scalar in meters.

Example: `'Width'`,0.2

Data Types: `double`

`'SlotCenter'` — Slot antenna center

[0 0 0] (default) | three-element vector in Cartesian coordinates

Slot antenna center, specified as the comma-separated pair consisting of `'SlotCenter'` and a three-element vector in Cartesian coordinates.

Example: `'SlotCenter'`,[8 0 0]

Data Types: `double`

`'GroundPlaneLength'` — Ground plane length
1.5000 (default) | scalar in meters

Ground plane length, specified as the comma-separated pair consisting of `'GroundPlaneLength'` and 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 in meters

Ground plane width, specified as the comma-separated pair consisting of `'GroundPlaneWidth'` and 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 in meters

Distance from center along x-axis, specified as the comma-separated pair consisting of `'FeedOffset'` and a scalar in meters. Offset from slot center is measured along the length.

Example: `'FeedOffset'`,3

Data Types: `double`

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

Tilt angle of antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or vector in degrees.

Example: `'Tilt',90`

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

Data Types: `double`

`'TiltAxis'` — Tilt axis of antenna
[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,“Rotate Antenna and Arrays”

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

Example: `'TiltAxis'`,`'Z'`

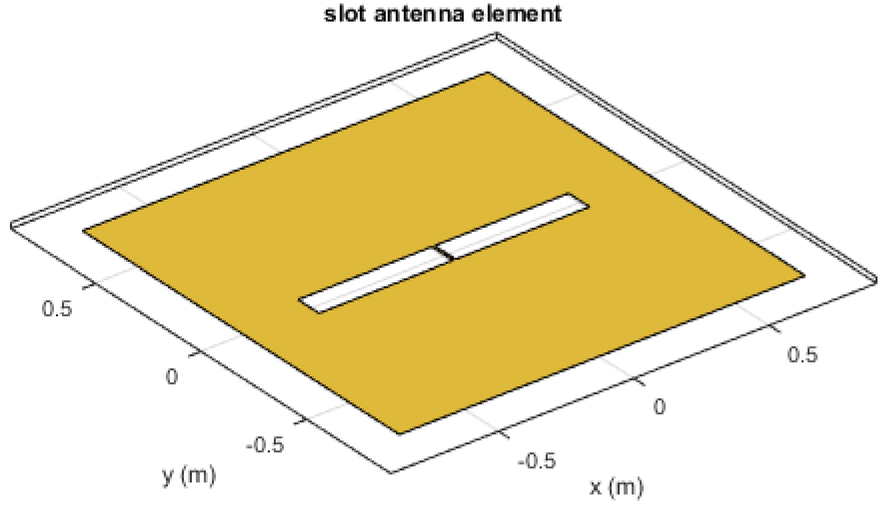
Data Types: `double`

Examples

Create and View Slot Antenna

Create and view a slot antenna that has 1m length and 100mm width.

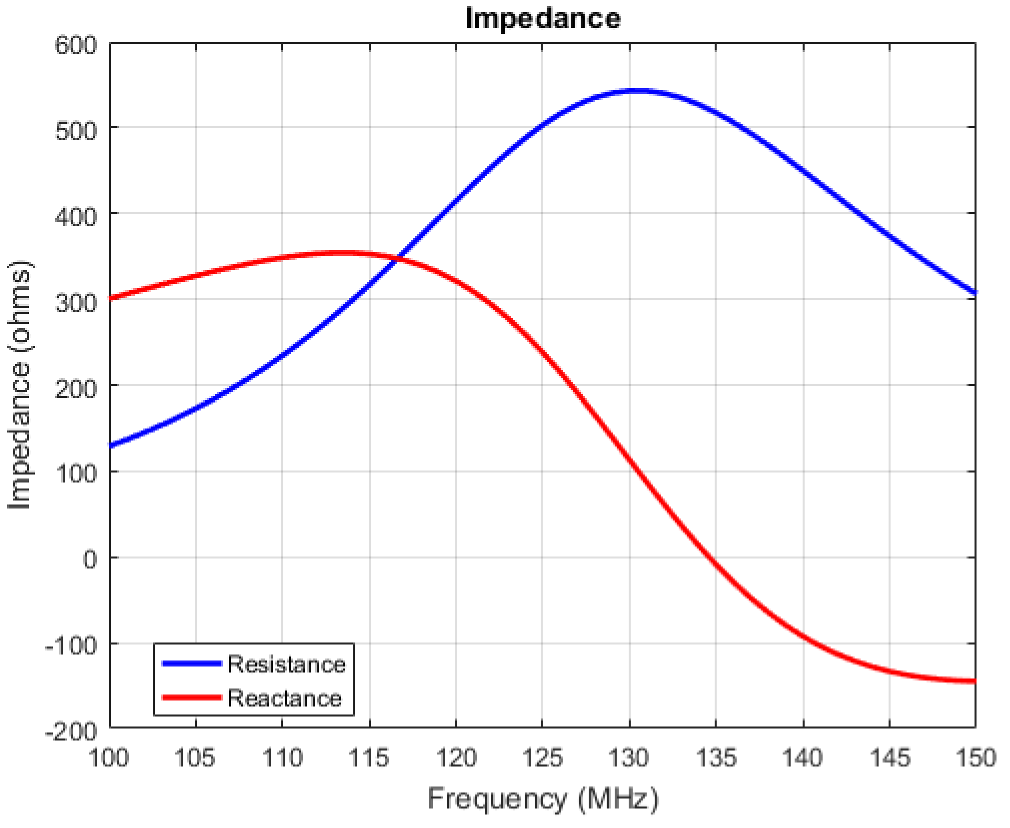
```
s = slot('Length',1,'Width',0.1);
show(s)
```



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

vivaldi | yagiUda | pifa

More About

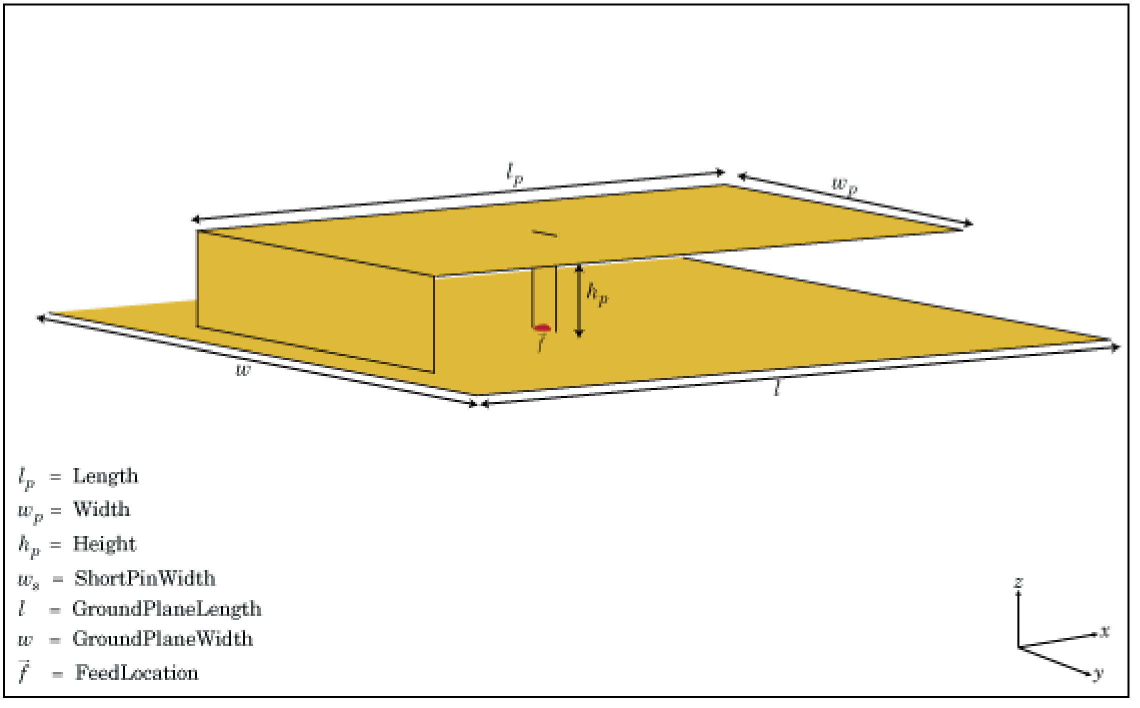
- "Rotate Antenna and Arrays"

Introduced in R2015a

pifa class

Create planar inverted-F antenna

Description



The `pifa` class creates a planar inverted-F antenna. The default PIFA antenna is centered at the origin. The feed point is along the length of the antenna.

Construction

`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 in meters

PIFA antenna length, specified as the comma-separated pair consisting of `'Length'` and 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 in meters

PIFA antenna width, specified as the comma-separated pair consisting of `'Width'` and 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 in meters

Height of the substrate, specified as the comma-separated pair consisting of `'Height'` and a scalar in meters.

Example: `'Height'`,37e-3

Data Types: `double`

`Substrate` — Type of dielectric material

'Air' (default) | dielectric material object handle | dielectric material from dielectric catalog

Type of the dielectric material used as a substrate, specified as the comma-separated pair consisting of `'Substrate'` and dielectric material object handle or dielectric material from dielectric catalog. For more information refer, `dielectric`. For more information on dielectric substrate meshing, refer “Meshing”.

Example: `'Substrate','FR4'`

`'GroundPlaneLength'` — Ground plane length

0.0360 (default) | scalar in meters

Ground plane length, specified as the comma-separated pair consisting of `'GroundPlaneLength'` and 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 in meters

Ground plane width, specified as the comma-separated pair consisting of `'GroundPlaneWidth'` and 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 in meters

Signed distance from the center along length and width of the ground plane, specified as the comma-separated pair consisting of `'PatchCenterOffset'` and 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 in meters

Shorting pin width of patch, specified as the comma-separated pair consisting of `'ShortPinWidth'` and 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 in meters

Signed distance from center along length and width of ground plane, specified as the comma-separated pair of `‘FeedOffset’` and 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`

`'Tilt'` — Tilt angle of antenna

0 (default) | scalar in degrees | vector in degrees

Tilt angle of antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or a vector in degrees.

Example: `'Tilt',90`

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

Data Types: `double`

`'TiltAxis'` — Tilt axis of antenna

[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,“Rotate Antenna and Arrays”

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

Example: `'TiltAxis'`,`'Z'`

Data Types: `double`

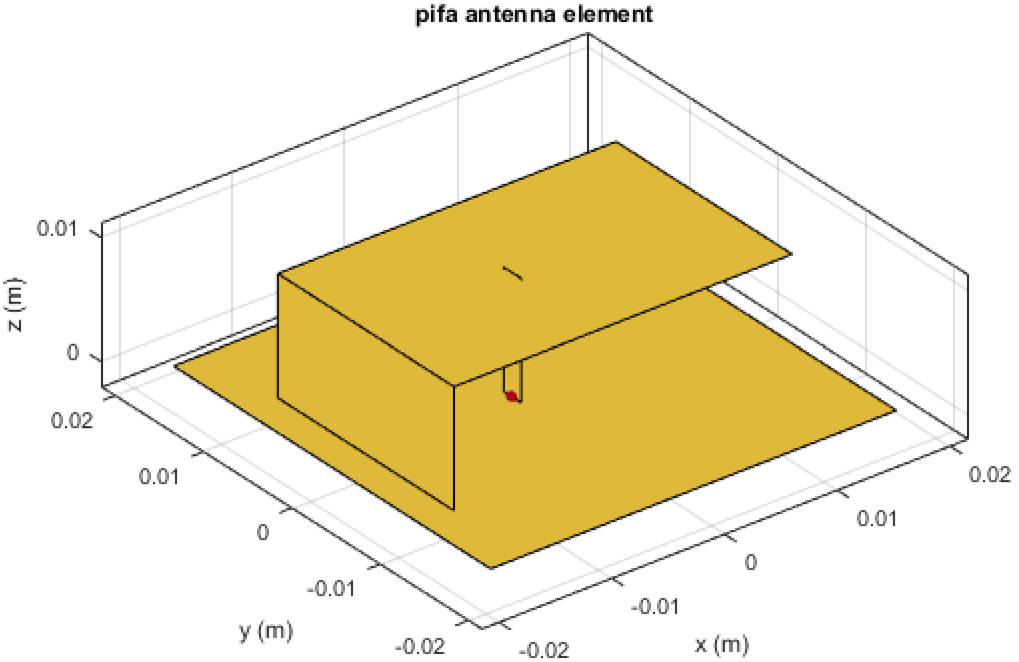
Examples

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
show(pf)
```

```
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]
```



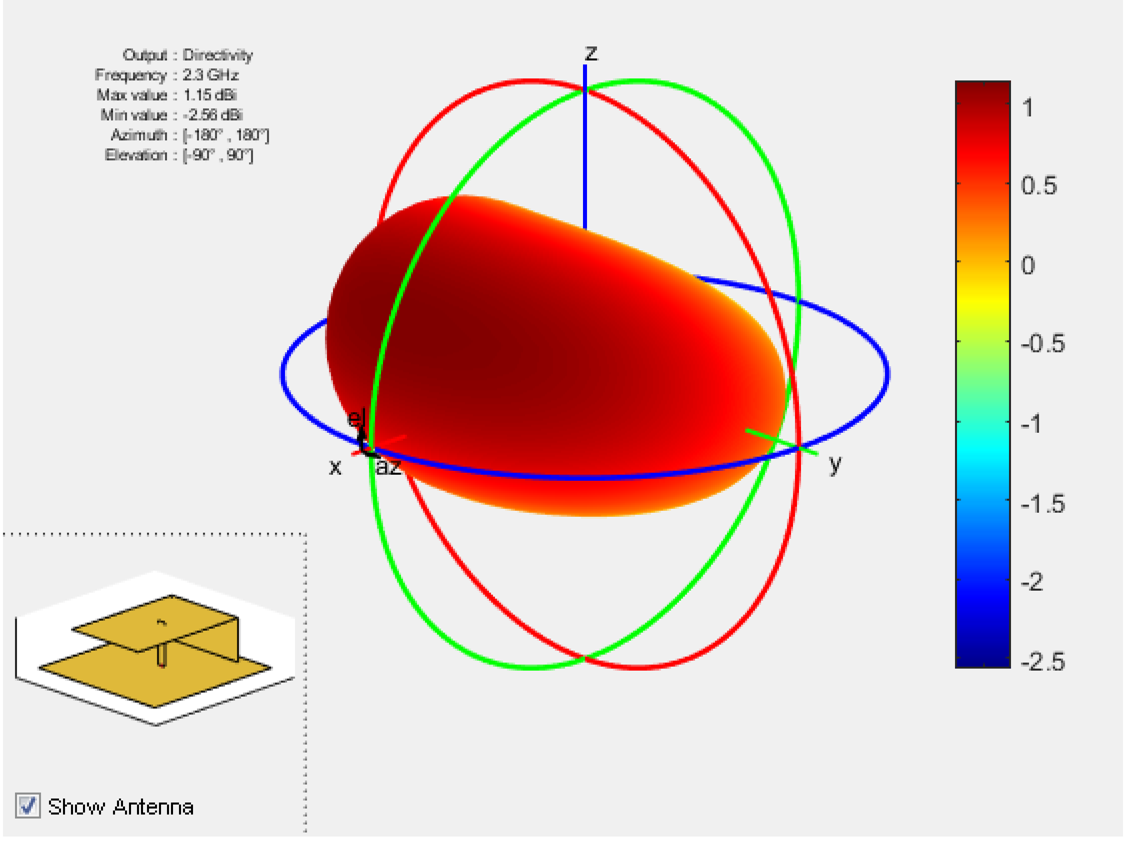
Radiation Pattern of PIFA Antenna

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)
pattern(pf,2.3e9);
```

```
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]
```

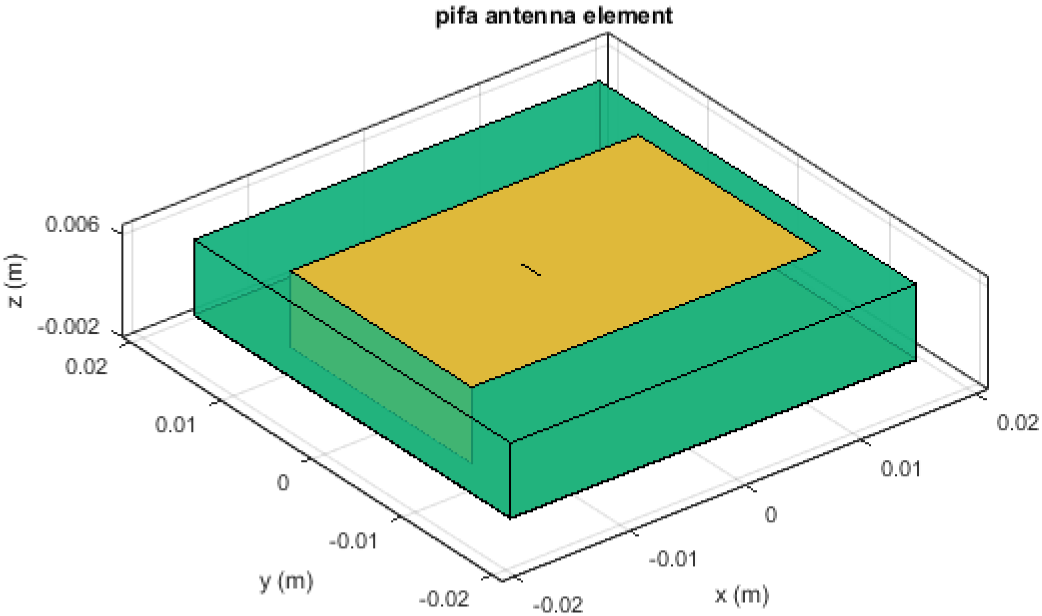


Impedance of PIFA Antenna

Create a PIFA antenna using a dielectric substrate **'RO4725JXR'**.

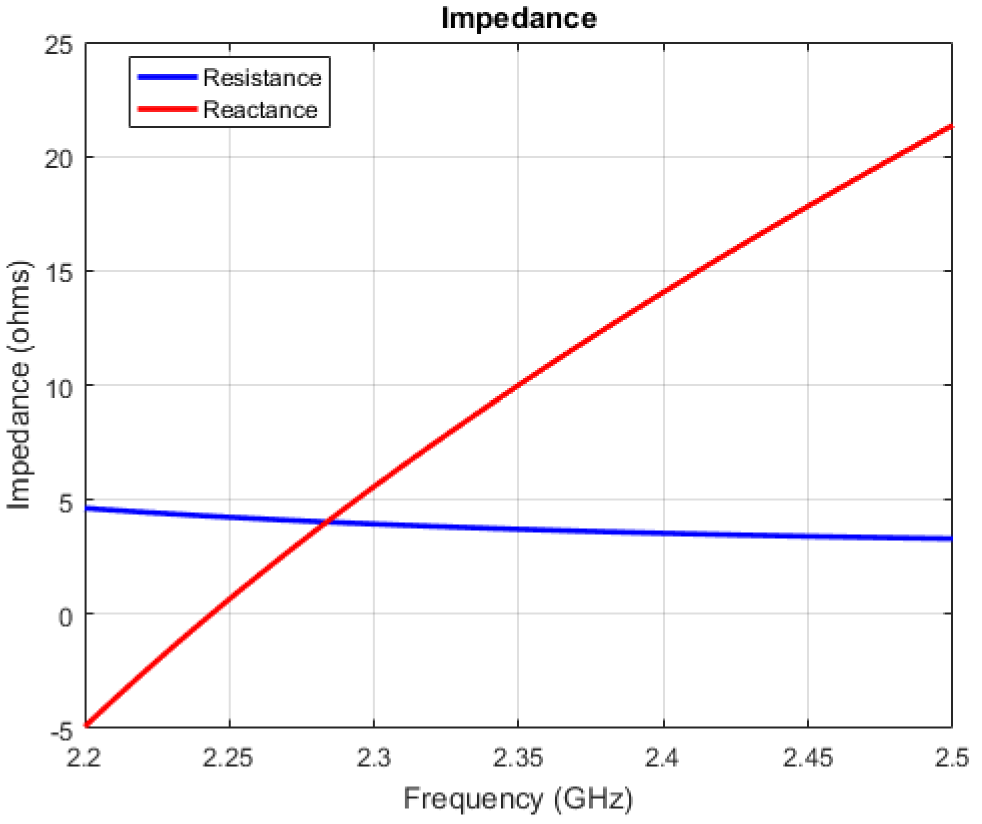
```
d = dielectric('RO4725JXR');
pf = pifa('Length',30e-3, 'Width',20e-3,'Height',0.0060, 'GroundPlaneLength',35e-3, ...
         'GroundPlaneWidth', 35e-3,'Substrate',d)
show(pf)
```

```
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

More About

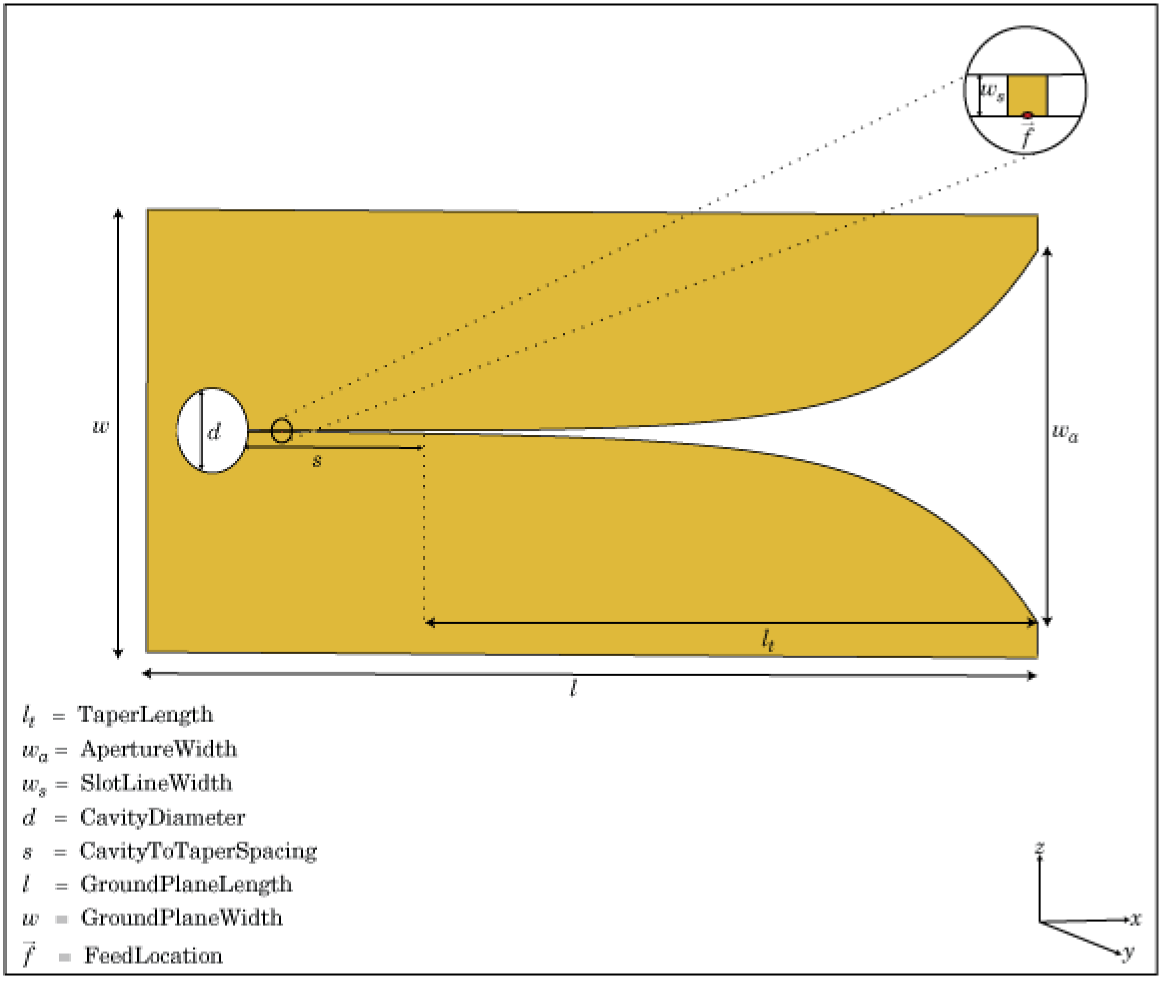
- "Rotate Antenna and Arrays"

Introduced in R2015a

vivaldi class

Create Vivaldi notch antenna on ground plane

Description



The `vivaldi` class creates a Vivaldi notch antenna on a ground plane.

Construction

`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 default values.

Properties

`'TaperLength'` — Taper length

0.2430 (default) | scalar in meters

Taper length of vivaldi, specified as the comma-separated pair consisting of `'TaperLength'` and a scalar in meters.

Example: `'TaperLength'`,2e-3

`'ApertureWidth'` — Aperture width

0.1050 (default) | scalar in meters

Aperture width, specified as the comma-separated pair consisting of `'ApertureWidth'` and a scalar in meters.

Example: `'ApertureWidth'`,3e-3

`'OpeningRate'` — Taper opening rate

0.2500 (default) | scalar

Taper opening rate, specified as the comma-separated pair consisting of `'OpeningRate'`and a scalar.

Example: `'OpeningRate'`,0.3

Data Types: `double`

`'SlotLineWidth'` — Slot line width

5.0000e-04 (default) | scalar in meters

Slot line width, specified as the comma-separated pair consisting of `'SlotLineWidth'` and a scalar in meters.

Example: `'SlotLineWidth'`,3

Data Types: `double`

`'CavityDiameter'` — Cavity termination diameter

0.0240 (default) | scalar in meters

Cavity termination diameter, specified as the comma-separated pair consisting of `'CavityDiameter'` and a scalar in meters.

Example: `'CavityDiameter'`,2

Data Types: `double`

`'CavityToTaperSpacing'` — Cavity to taper distance of transition

0.0230 (default) | scalar in meters

Cavity to taper distance of transition, specified as the comma-separated pair consisting of `'CavityToTaperSpacing'` and a scalar in meters. By default, this property is measured along x-axis.

Example: `'CavityToTaperSpacing'`,3

Data Types: `double`

`'GroundPlaneLength'` — Ground plane length

0.3000 (default) | scalar in meters

Ground plane length, specified as the comma-separated pair consisting of `'GroundPlaneLength'` and a scalar in meters. By default, ground plane length is measured along the x-axis.

Example: `'GroundPlaneLength'`,3

Data Types: `double`

`'GroundPlaneWidth'` — Ground plane width

0.1250 (default) | scalar in meters

Ground plane width, specified as the comma-separated pair consisting of `'GroundPlaneWidth'` and 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 in meters

Distance from feed along x-axis, specified as the comma-separated pair consisting of `'FeedOffset'` and a scalar in meters.

Example: `'FeedOffset'`,3

Data Types: `double`

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

Tilt angle of antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or vector in degrees.

Example: `'Tilt',90`

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

Data Types: `double`

`'TiltAxis'` — Tilt axis of antenna
[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,“Rotate Antenna and Arrays”

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

Example: `'TiltAxis'`,`'Z'`

Data Types: `double`

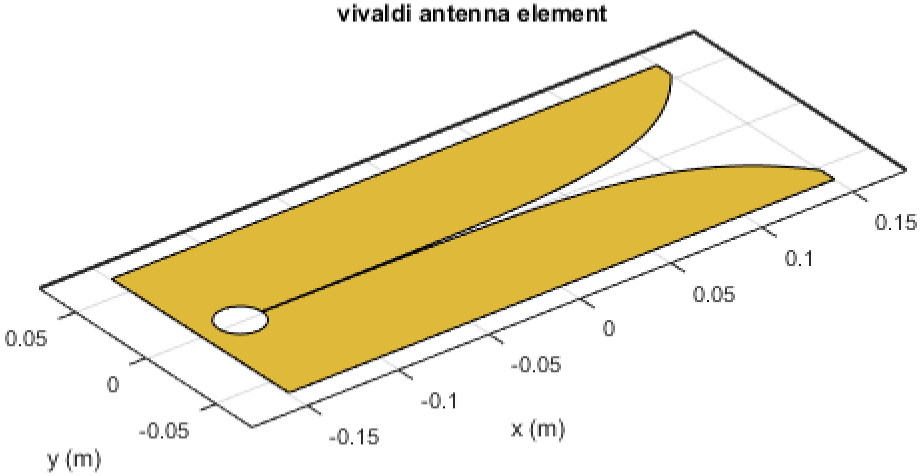
Examples

Create and View Vivaldi Antenna

Create and view the default Vivaldi antenna.

```
vi = vivaldi
show(vi);
```

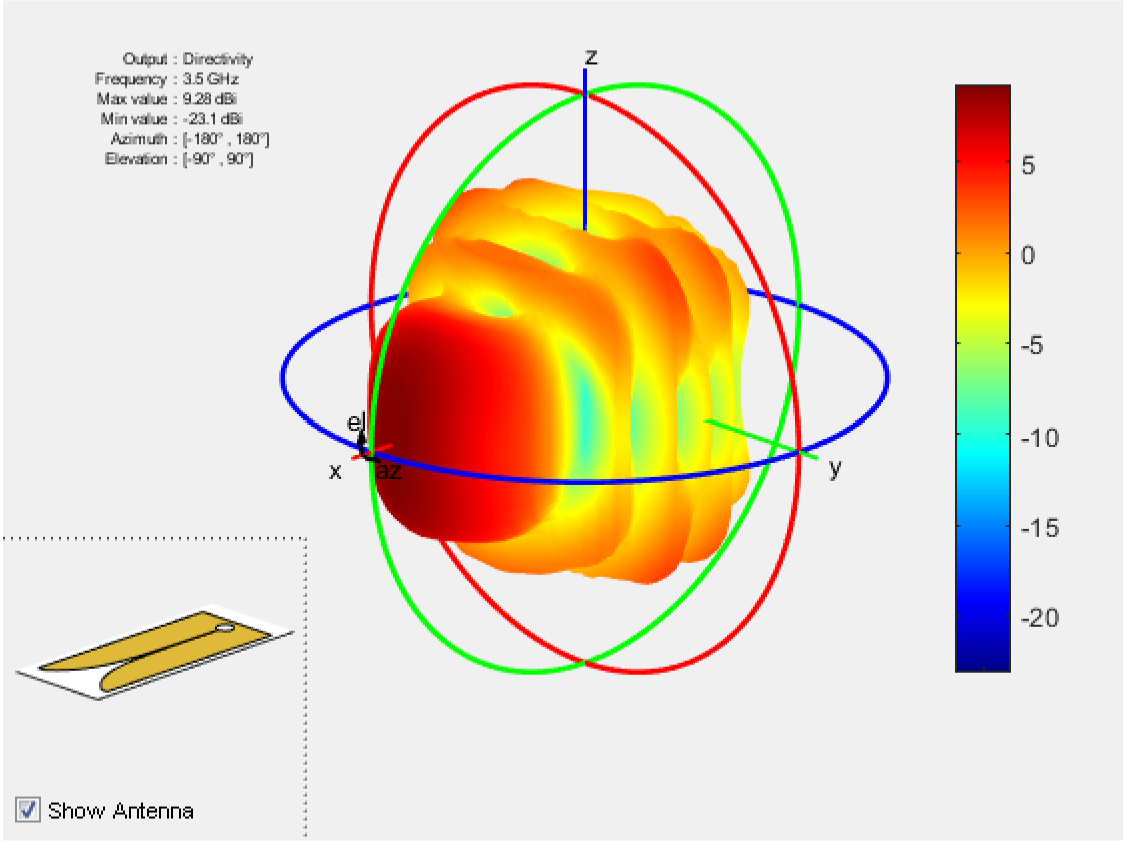
```
vi = 

  vivaldi with properties:

            TaperLength: 0.2430
          ApertureWidth: 0.1050
            OpeningRate: 0.2500
          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]
```



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

spiralArchimedean | slot | yagiUda

More About

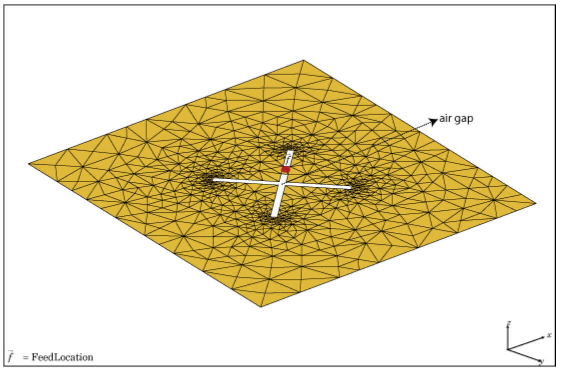
- “Rotate Antenna and Arrays”

Introduced in R2015a

customAntennaMesh class

Create 2-D custom mesh antenna on X-Y plane

Description



The `customAntennaMesh` class 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.

Construction

`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 `3x`*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 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.

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 in degrees

Tilt angle of the antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar in degrees.

Example: `'Tilt'`,0

Data Types: `double`

'Tilt' — Tilt angle of antenna
0 (default) | scalar in degrees | vector in degrees

Tilt angle of antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or vector in degrees.

Example: `'Tilt',90`

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

Data Types: `double`

'TiltAxis' — Tilt axis of antenna
[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,"Rotate Antenna and Arrays"

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

Example: `'TiltAxis','Z'`

Data Types: `double`

Methods

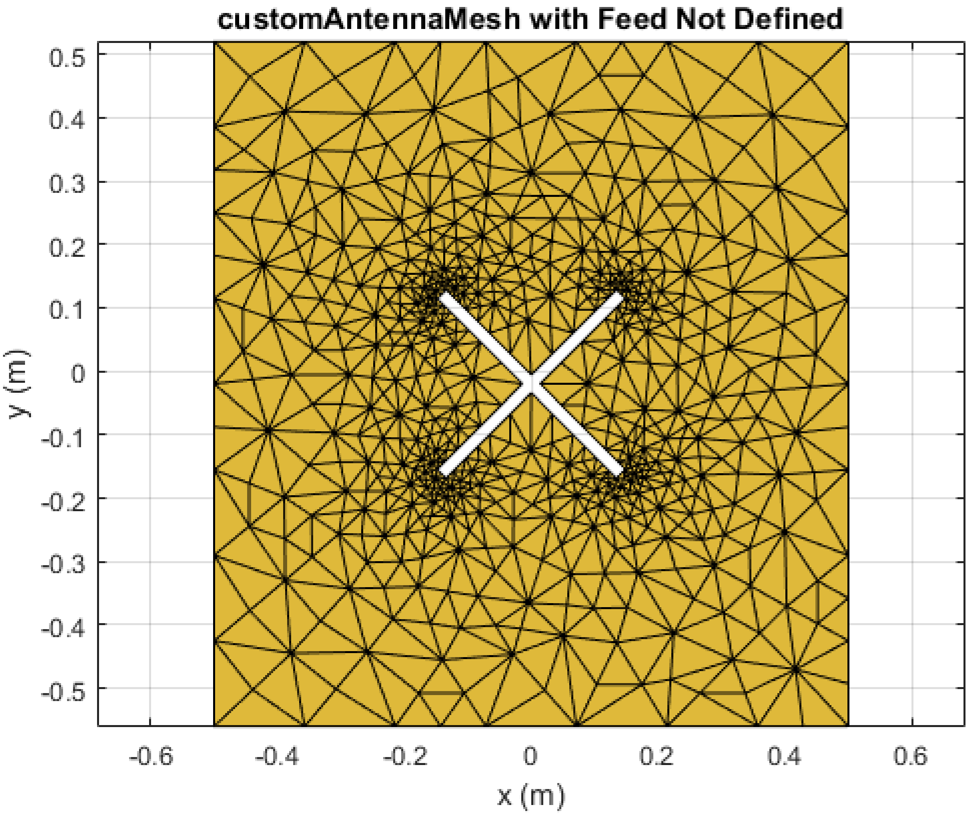
`createFeed`	Create feed location for custom antenna

Examples

Custom Planar 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 =

   0.5377 +55.2703i
```

See Also

reflector | cavity

More About

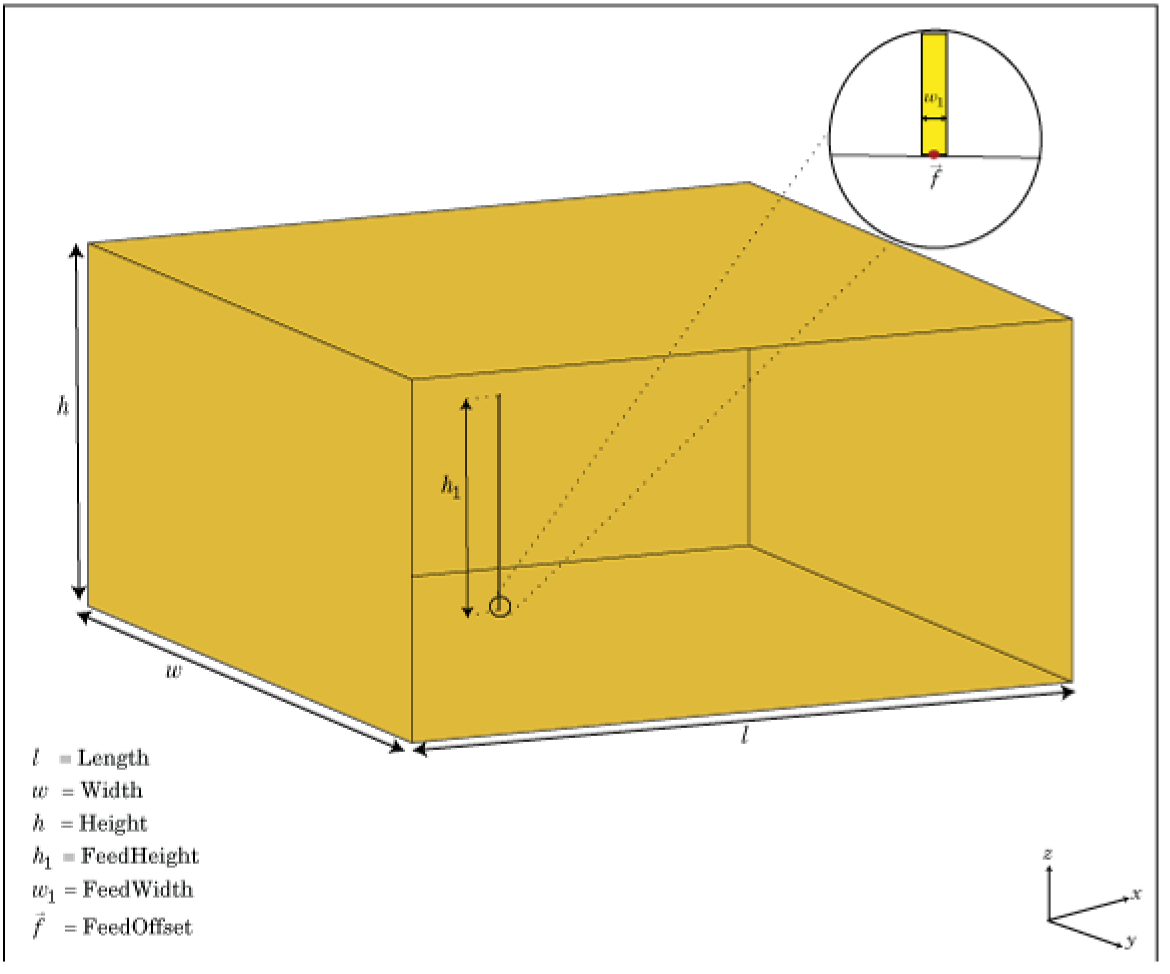
- "Rotate Antenna and Arrays"

Introduced in R2015b

waveguide class

Create rectangular waveguide

Description



The waveguide class creates 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.

Construction

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 in meters

Height of feed, specified as the comma-separated pair consisting of 'FeedHeight' and 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 in meters

Width of feed, specified as the comma-separated pair consisting of 'FeedWidth' and 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 the comma-separated pair consisting of 'Length' and 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 the comma-separated pair consisting of `'Width'` and a scalar in meters.

Example: `'Width',0.05`

Data Types: `double`

`'Height'` — Rectangular waveguide height
`0.0102` (default) | scalar in meters

Rectangular waveguide height, specified as the comma-separated pair consisting of `'Height'` and 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 in meters

Signed distance of feedpoint from center of ground plane, specified as the comma-separated pair of `'FeedOffset'` and 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`

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

Tilt angle of antenna, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or vector in degrees.

Example: 'Tilt',90

Example: 'Tilt',[90 90 0]

Data Types: double

'TiltAxis' — Tilt axis of antenna
[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of 'TiltAxis' and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,“Rotate Antenna and Arrays”

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

Example: 'TiltAxis','Z'

Data Types: double

Examples

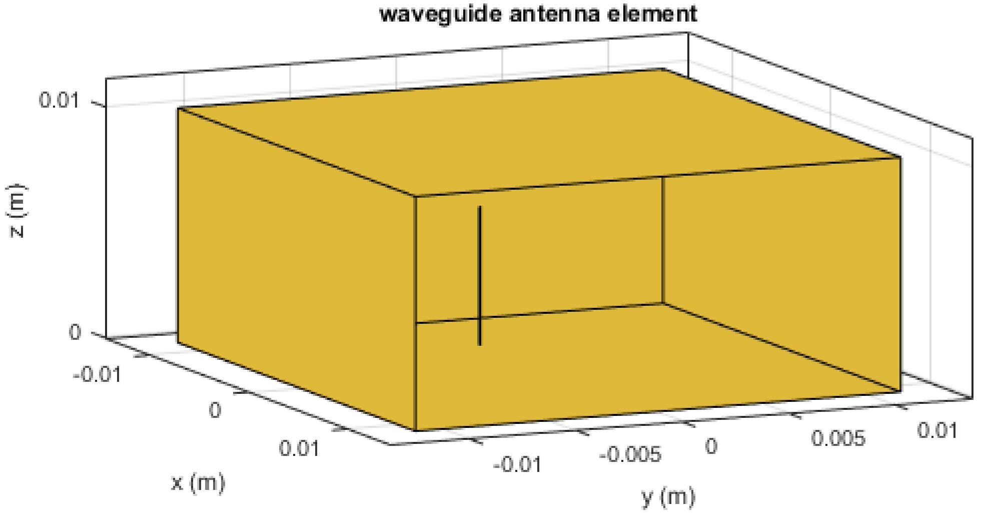
Default Rectangular Waveguide

Create a rectangular waveguide using default dimensions. Display the waveguide.

```
wg = waveguide
show(wg)
```

```
wg =

  waveguide with properties:

        Length: 0.0240
         Width: 0.0229
        Height: 0.0102
```

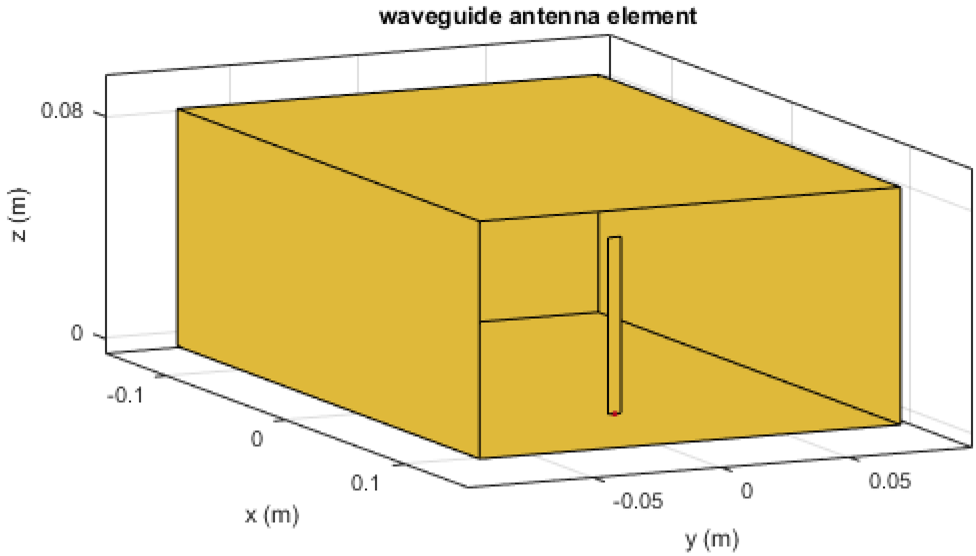
```
 FeedWidth: 6.0000e-05
FeedHeight: 0.0060
FeedOffset: [-0.0060 0]
      Tilt: 0
  TiltAxis: [1 0 0]
```



Radiation Pattern of WR-650 Rectangular Waveguide

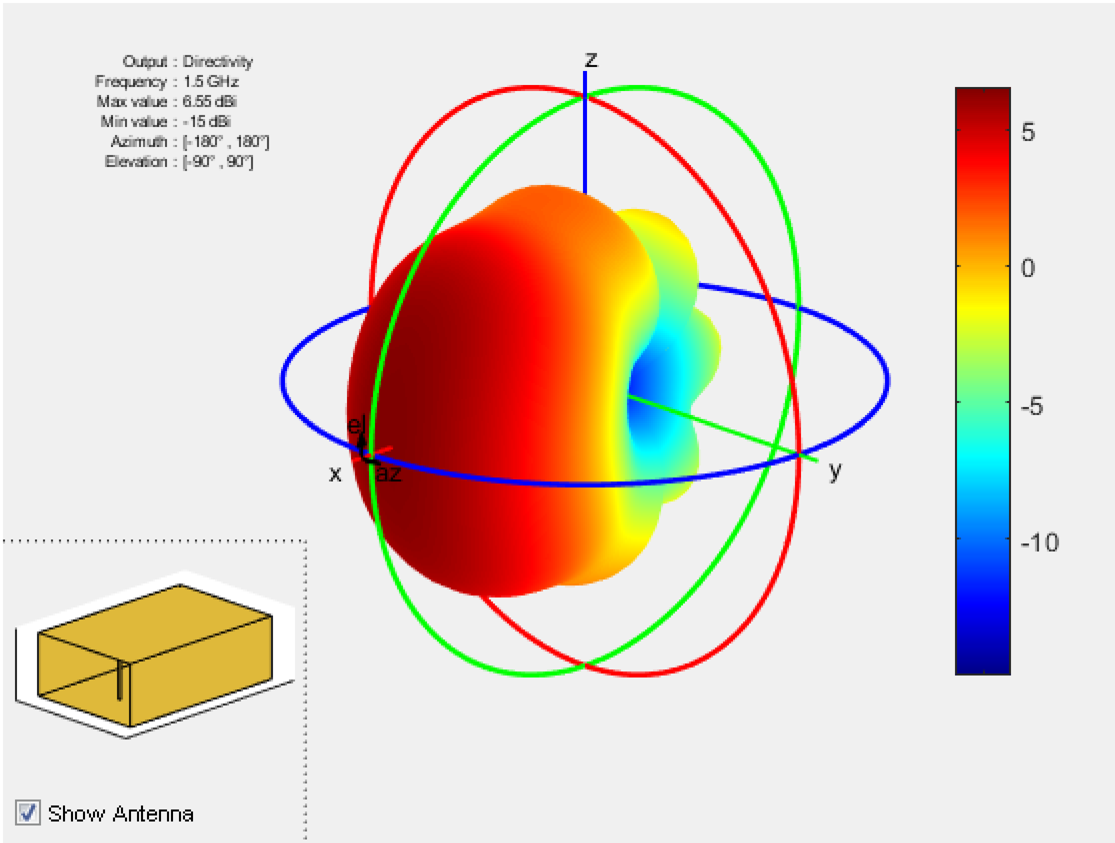
Create a WR-650 rectangular waveguide and display it.

```
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

More About

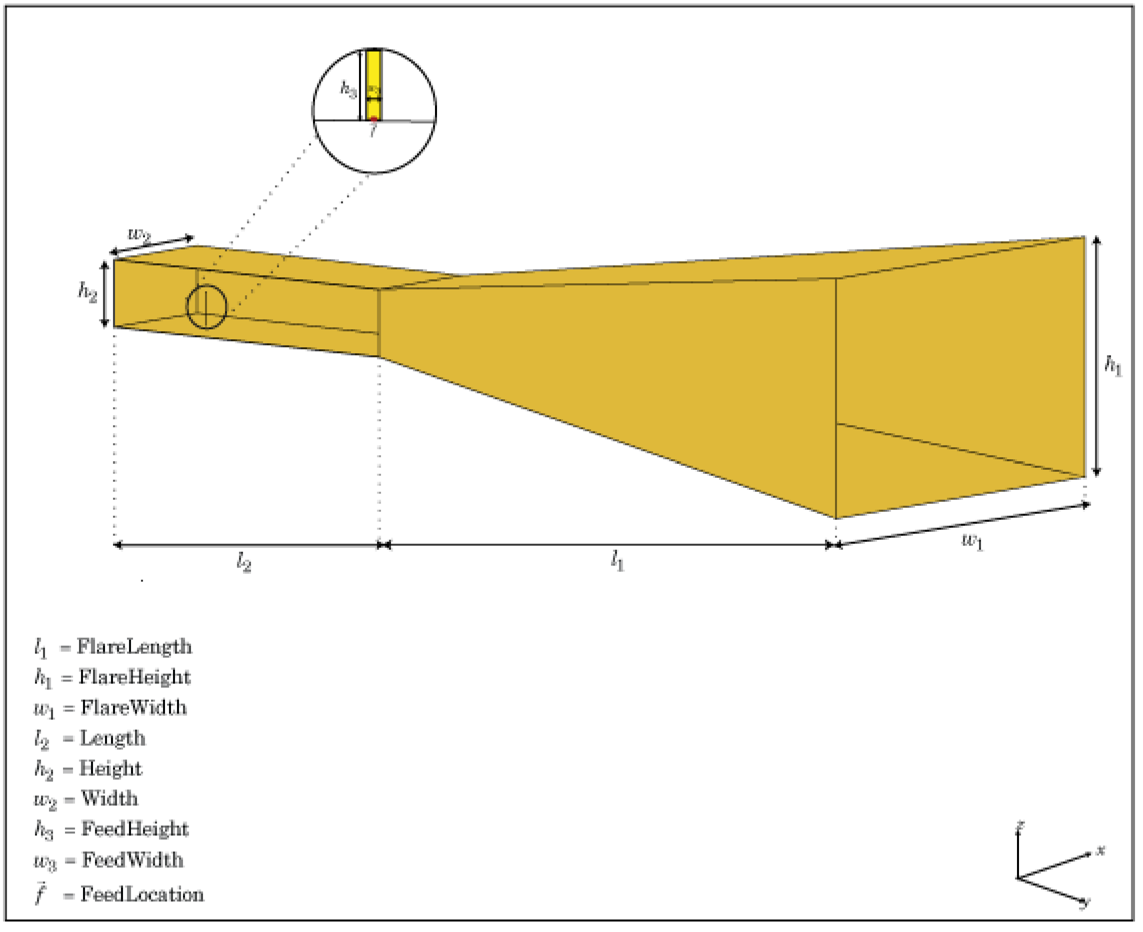
- "Rotate Antenna and Arrays"

Introduced in R2016a

horn class

Create horn antenna

Description



The `horn` class creates 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.

Construction

`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 in meters

Flare length of horn, specified as the comma-separated pair consisting of `'FlareLength'` and a scalar in meters.

Example: `'FlareLength',0.35`

Data Types: `double`

`'FlareWidth'` — Flare width of horn

`0.0571` (default) | scalar in meters

Flare width of horn, specified as the comma-separated pair consisting of `'FlareWidth'` and a scalar in meters.

Example: `'FlareWidth',0.2`

Data Types: `double`

`'FlareHeight'` — Flare height of horn

`0.0338` (default) | scalar in meters

Flare height of horn, specified as the comma-separated pair consisting of `'FlareHeight'` and a scalar in meters.

Example: `'FlareHeight',0.15`

Data Types: `double`

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

Rectangular waveguide length, specified as the comma-separated pair consisting of `'Length'` and a scalar in meters.

Example: `'Length',0.09`

Data Types: `double`

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

Rectangular waveguide width, specified as the comma-separated pair consisting of `'Width'` and a scalar in meters.

Example: `'Width',0.05`

Data Types: `double`

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

Rectangular waveguide height, specified as the comma-separated pair consisting of `'Height'` and a scalar in meters.

Example: `'Height',0.0200`

Data Types: `double`

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

Height of feed, specified as the comma-separated pair consisting of `'FeedHeight'` and a scalar in meters.

Example: `'FeedHeight',0.0050`

Data Types: `double`

'FeedWidth' — Width of feed

1.0000e-04 (default) | scalar in meters

Width of feed, specified as the comma-separated pair consisting of 'FeedWidth' and 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 in meters

Signed offset from center of ground plane, specified as the comma-separated pair of 'FeedOffset' and a two-element vector in meters.

Example: 'FeedOffset',[–0.0070 0.01]

Data Types: double

'Tilt' — Tilt angle of antenna

0 (default) | scalar in degrees | vector in degrees

Tilt angle of antenna, specified as the comma-separated pair consisting of 'Tilt' and a scalar or vector in degrees.

Example: 'Tilt',90

Example: 'Tilt',[90 90 0]

Data Types: double

'TiltAxis' — Tilt axis of antenna

[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of 'TiltAxis' and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,“Rotate Antenna and Arrays”

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

Example: `'TiltAxis','Z'`

Data Types: `double`

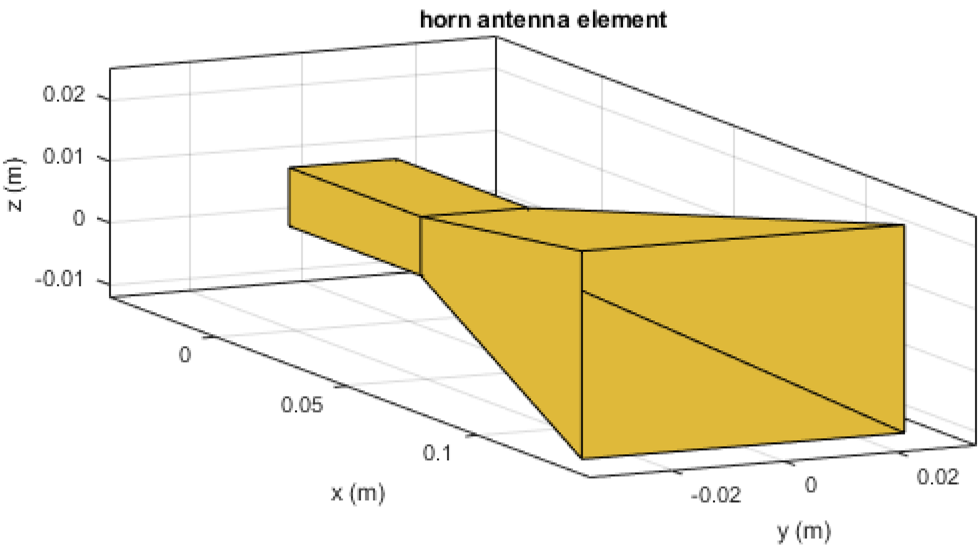
Examples

Default Horn Antenna

Create and view a default horn antenna.

```
h = horn
show(h)
```

```
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]
```



References

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

See Also

waveguide

More About

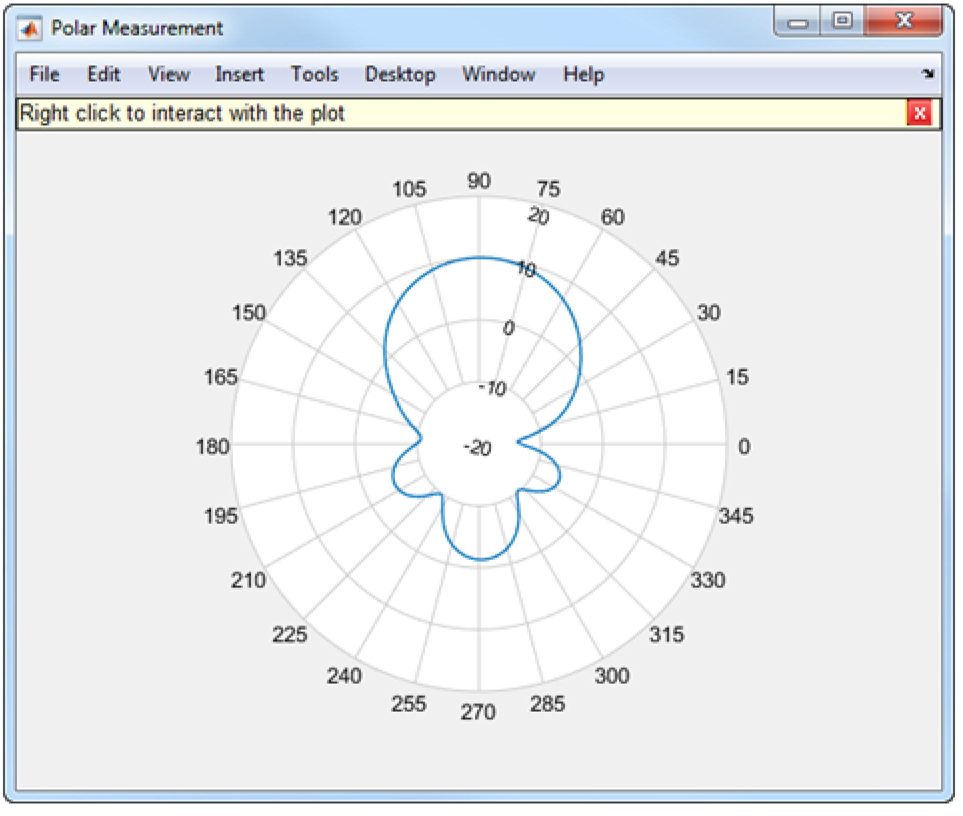
- "Rotate Antenna and Arrays"

Introduced in R2016a

polarpattern class

Interactive plot of radiation patterns in polar format

Description



`polarpattern` class plots antenna radiation pattern 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 radiation patterns and other types of data in polar format. `polarpattern` plots field value data of antennas 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 antenna pattern from the polar plot. For example, `p = polarpattern(data,'Peaks',3)` identifies and displays the three highest peaks in the antenna data. For a list of properties, see PolarPattern Properties.

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

Input Arguments

`data` — Antenna data

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

Antenna data, specified as one of the following:

- A real length-M vector, where M contains the magnitude values with angles assumed to be $\frac{(0:M-1)}{M \times 360}$ 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 $\frac{(0:M-1)}{M \times 360}$ degrees. The set of each angle can vary for each column.
- `N`-by-`D` arrays, 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 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.

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

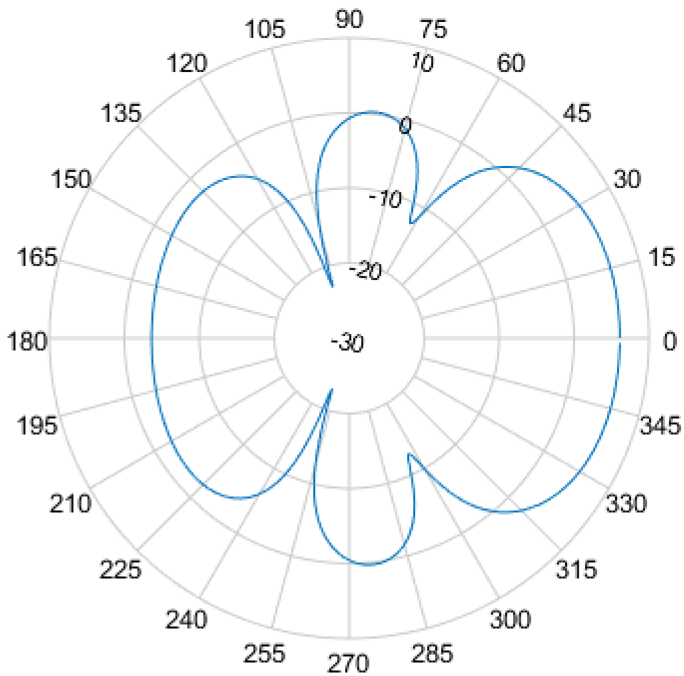
Polar Pattern for Vivaldi Antenna

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');
```

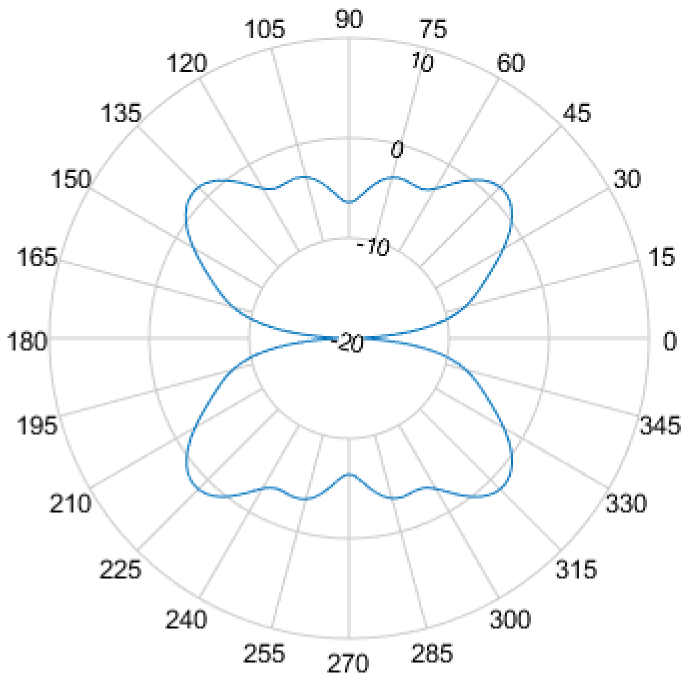
Read the cavity specification file into `Horizontal`, `Vertical`, and `Optional` structures using the msiread function.

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

```
Horizontal =

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


Vertical =

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


Optional =

         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);
```



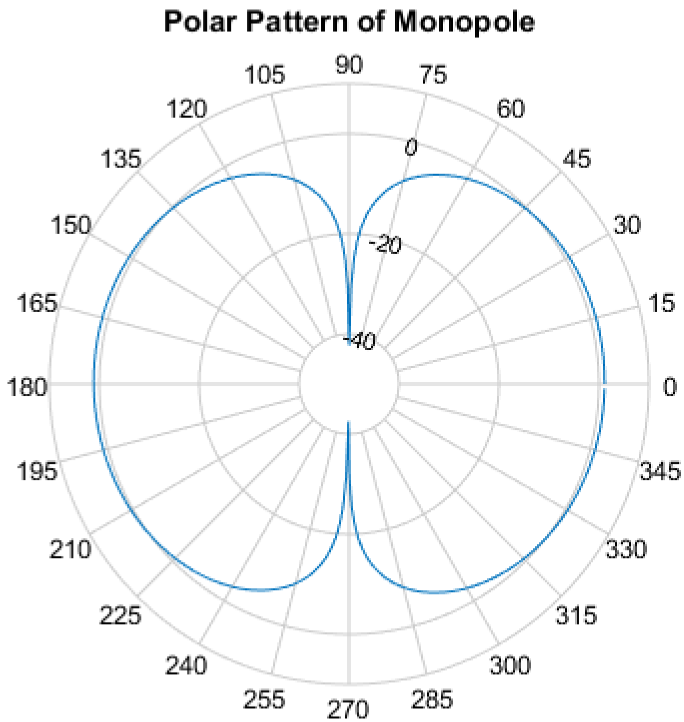
Add Title to Polar Plot

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);
```

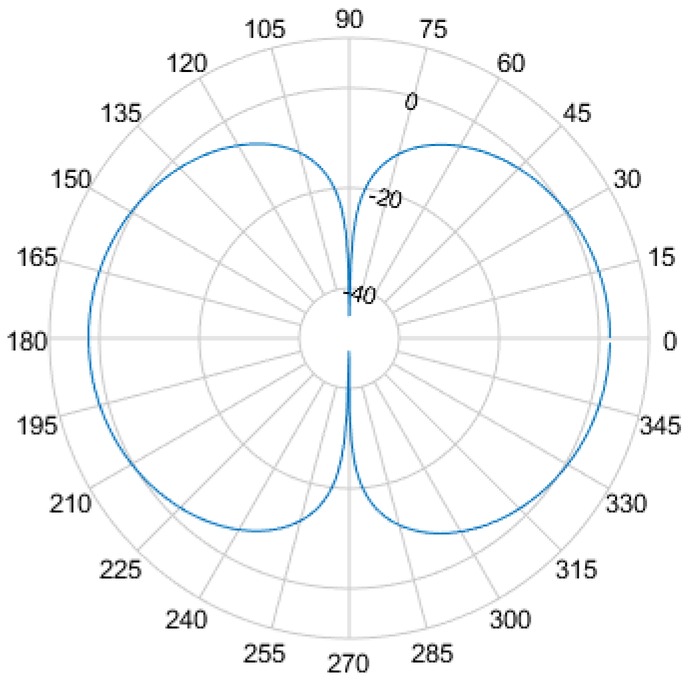
Plot the polar pattern of the antenna and display the properties of the plot.

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

```
  internal.polari handle with properties:
```

```
                Interactive: 1
               LegendLabels: ''
             AntennaMetrics: 0
                  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: 'linear'
              DisplayUnits: 'linear'
             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
```



- “Interact with Polar Plot”

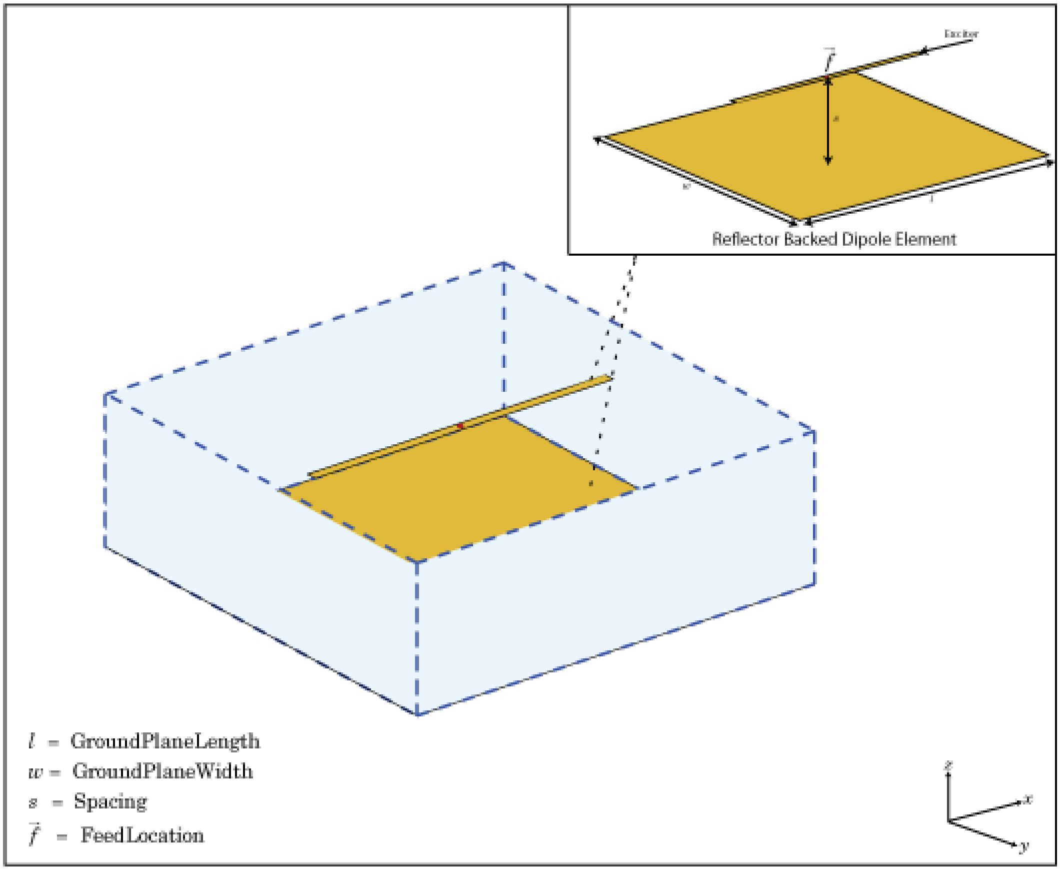
Introduced in R2016a

Array Classes — Alphabetical List

infiniteArray class

Create infinite antenna array

Description



The `infiniteArray` class creates 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.

Construction

`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) | antenna object

Type of individual antenna elements in unit cell, specified as the comma-separated pair consisting of `'Element'` and an antenna object. Antennas 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 in degrees

Scan direction in azimuth plane, specified as the comma-separated pair consisting of `'ScanAzimuth'` and a scalar in degrees.

Example: `'ScanAzimuth',25`

Data Types: `double`

`'ScanElevation'` — Scan direction in elevation plane

0 (default) | scalar in degrees

Scan direction in elevation plane, specified as the comma-separated pair consisting of `'ScanElevation'` and a scalar in degrees.

Example: `'ScanElevation',80`

Data Types: `double`

Methods

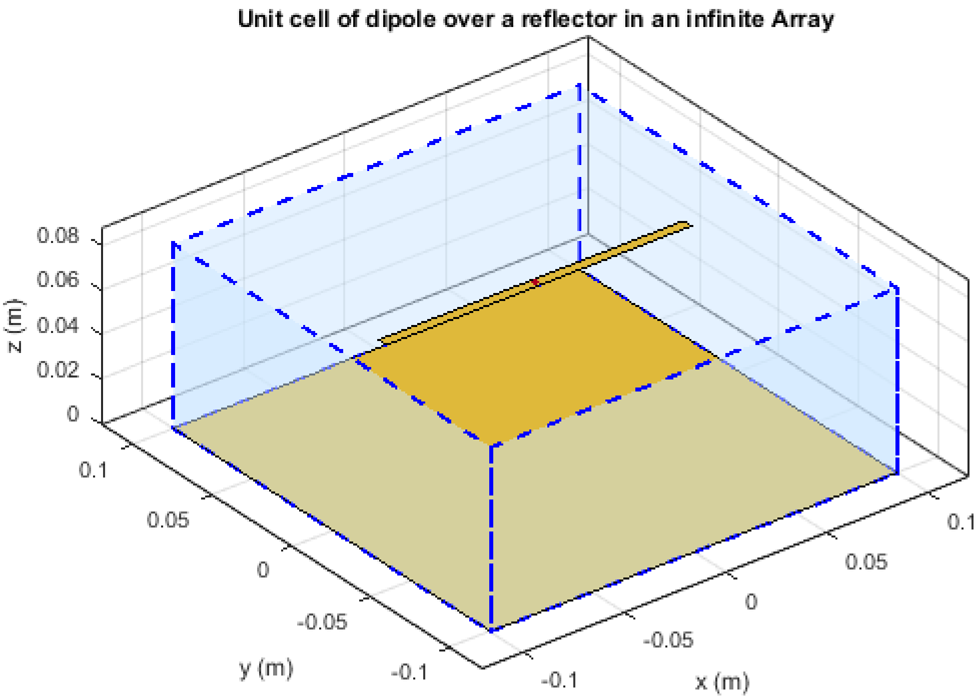
`numSummationTerms`	Modify the number of summation terms for calculating periodic Green's function

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.

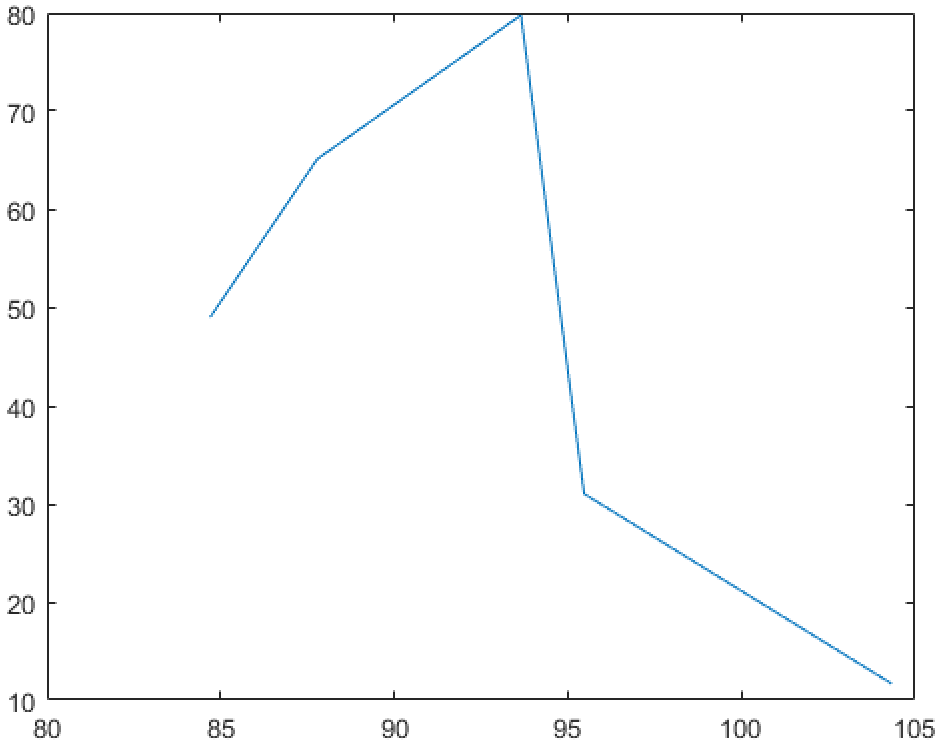
```
infa = infiniteArray('Element',reflector,'ScanAzimuth',0, ...
    'ScanElevation',90);
show(infa)
```



Scan Impedance of Infinite Array

Calculate the scan impedance of an infinite array at 1GHz. To calculate the impedance, scan the inifinite 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

linearArray | rectangularArray

More About

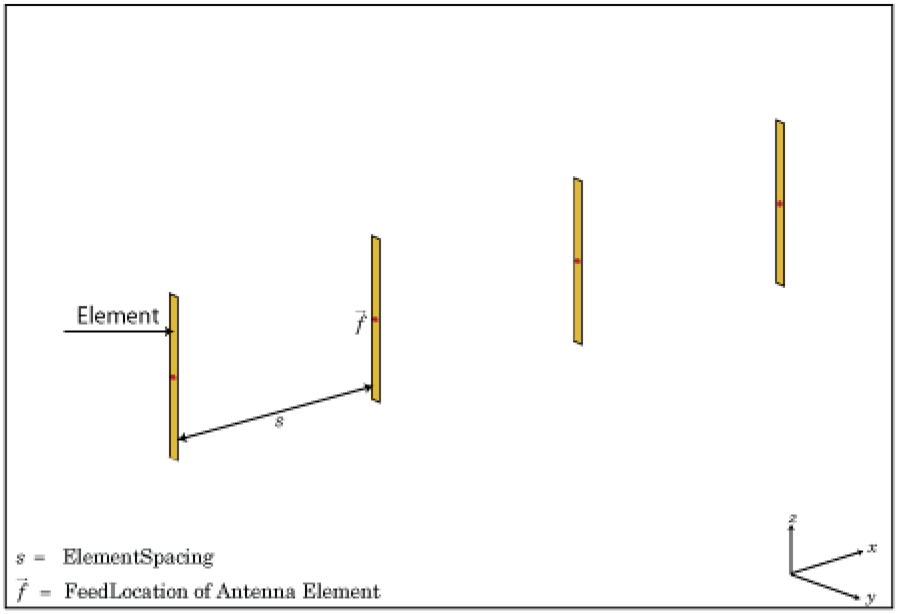
- "Infinite Arrays"
- "Antenna Toolbox Limitations"

Introduced in R2015b

linearArray class

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.

Construction

`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 the comma-separated pair consisting of `'Element'` and an antenna object.

Example: `'Element'`,monopole

`'NumElements'` — Number of antenna elements in array

2 (default) | scalar

Number of antenna elements in array, specified as the comma-separated pair consisting of `'NumElements'` and a scalar.

Example: `'NumElements'`,4

`'ElementSpacing'` — Spacing between antenna elements

2 (default) | scalar in meters | vector in meters

Spacing between antenna elements, specified as the comma-separated pair consisting of `'ElementSpacing'` and 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 the comma-separated pair consisting of `'AmplitudeTaper'` and 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 in degrees | vector in degrees

Phase shift for antenna elements, specified as the comma-separated pair consisting of `'PhaseShift'` and a scalar or vector in degrees.

Example: `'PhaseShift'`,[3 3 0 0]

Data Types: `double`

'Tilt' — Tilt angle of array
0 (default) | scalar in degrees | vector in degrees

Tilt angle of an array, specified as the comma-separated pair consisting of `'Tilt'` and a scalar or vector in degrees.

Example: `'Tilt',90`

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

Data Types: `double`

'TiltAxis' — Tilt axis of antenna
[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,"Rotate Antenna and Arrays"

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

Example: `'TiltAxis','Z'`

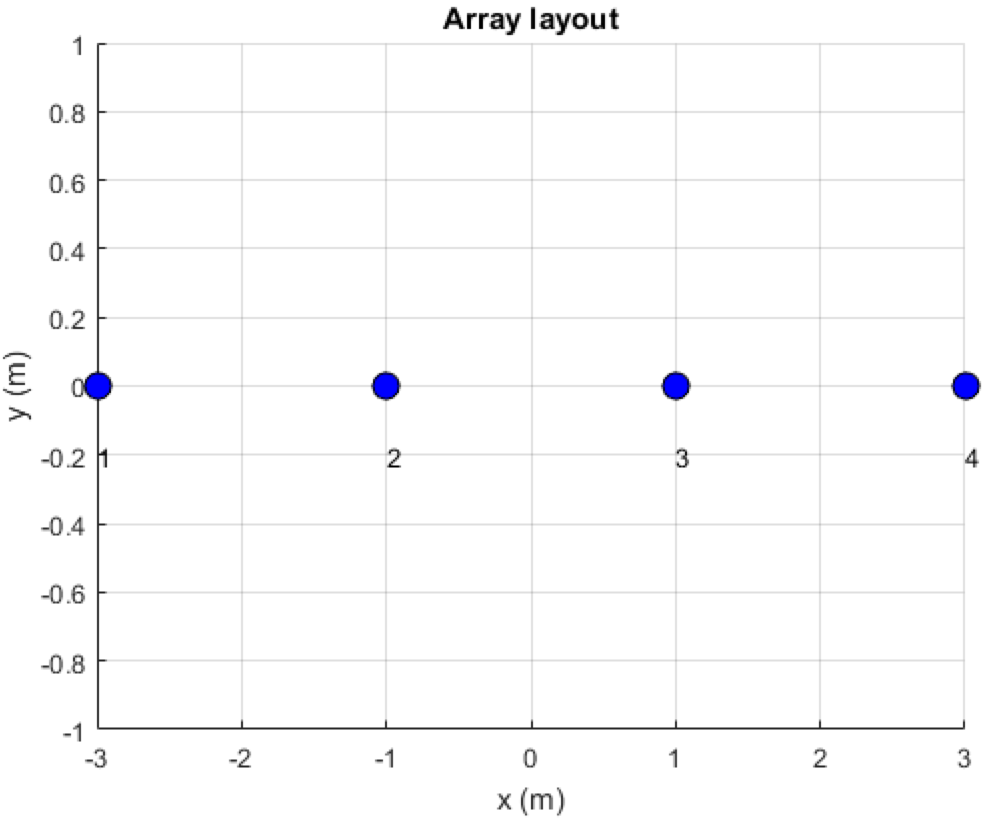
Data Types: `double`

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);
```

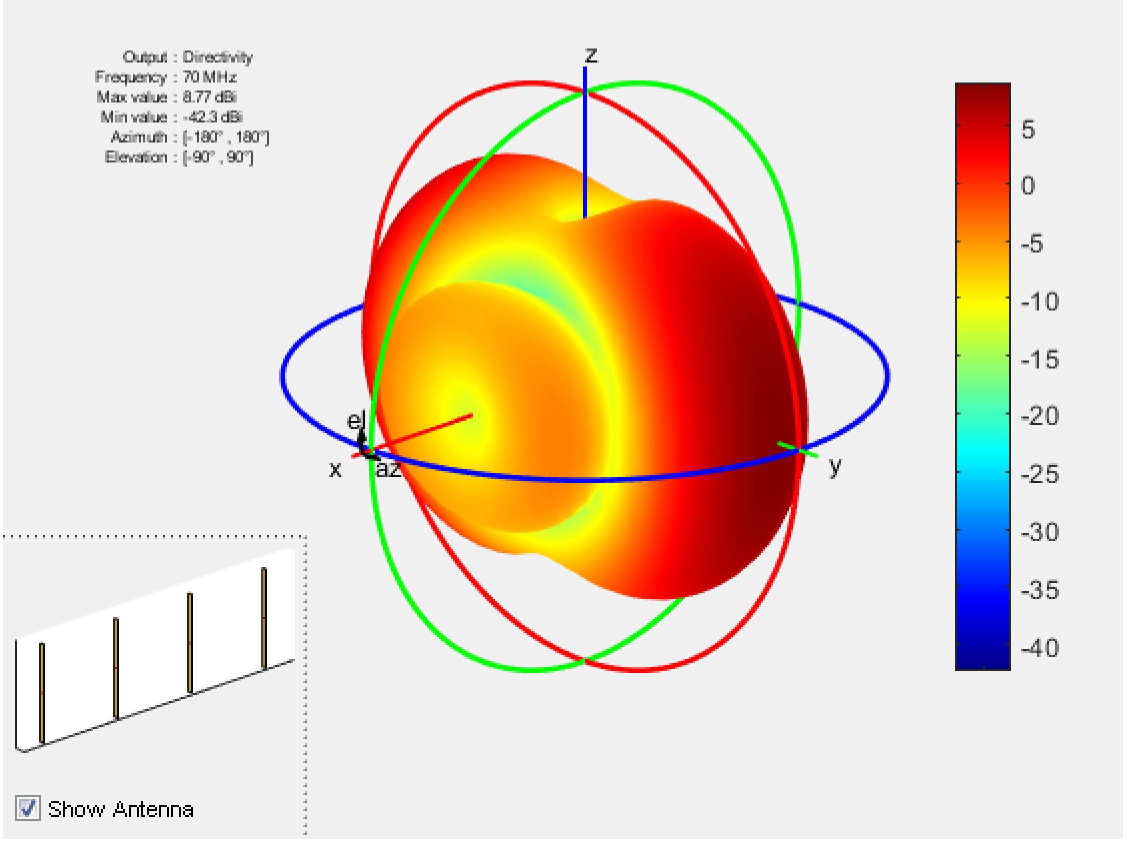


Radiation Pattern of Linear Array

Plot the radiation pattern of a four element linear array of dipoles at a frequency 70MHz.

```
la = linearArray('NumElements',4);
```

```
pattern(la,70e6);
```



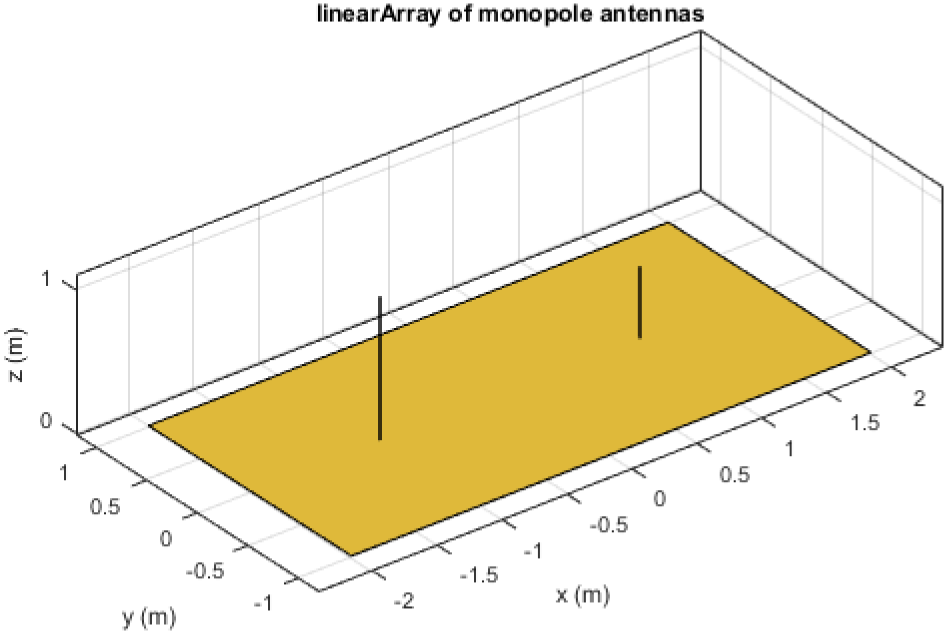
Linear Array Using Groundplane Antennas

Create a linear array of two monopoles.

```
m1 = monopole;
m2 = monopole('Height',0.5);
mla = linearArray
mla.Element = [m1,m2];
show(mla);
```

```
mla =
```

```
linearArray with properties:

         Element: [1x1 dipole]
     NumElements: 2
  ElementSpacing: 2
  AmplitudeTaper: 1
      PhaseShift: 0
            Tilt: 0
        TiltAxis: [1 0 0]
```



References

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

See Also

rectangularArray | infiniteArray

More About

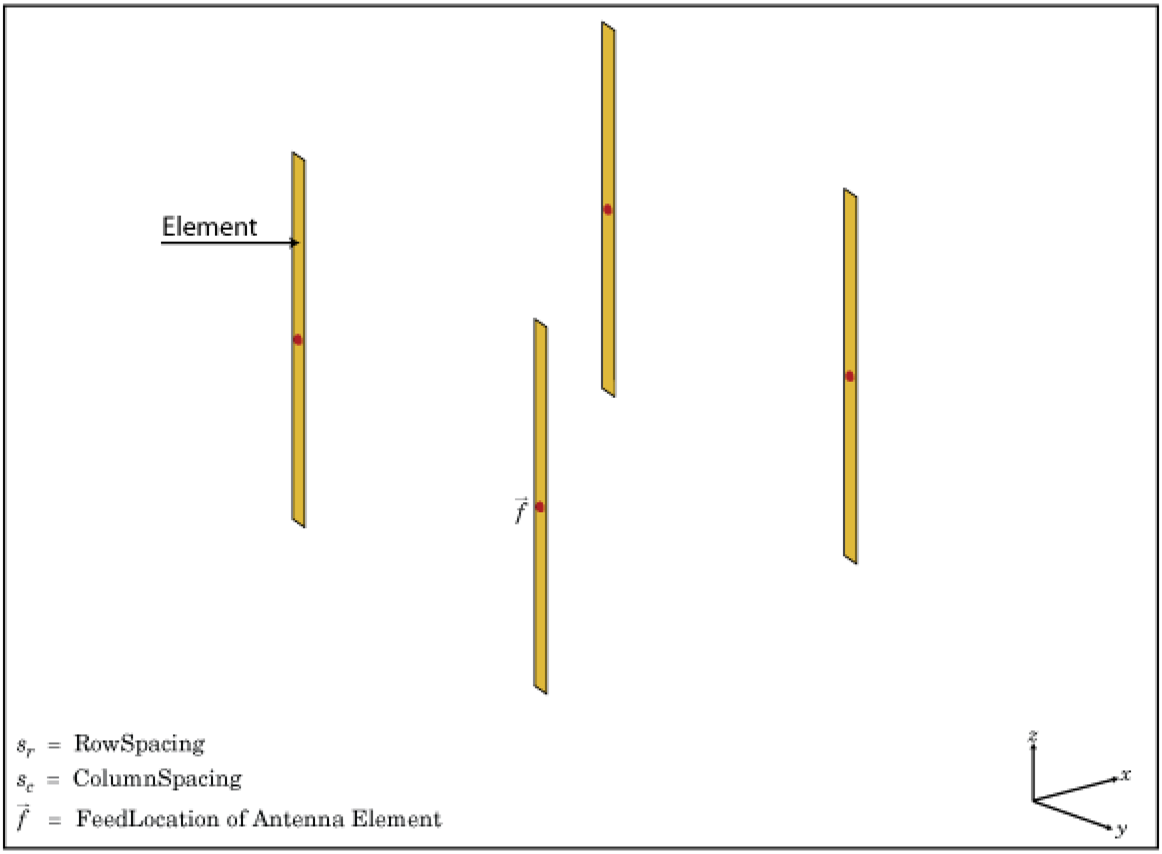
- "Rotate Antenna and Arrays"
- "Antenna Toolbox Limitations"

Introduced in R2015a

rectangularArray class

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.

Construction

`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 the comma-separated pair consisting of `'Element'` and 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 the comma-separated pair consisting of `'Size'` and a two-element vector.

Example: `'Size'`,[4 4]

`'RowSpacing'` — Row spacing between two antenna elements

2 (default) | scalar in meters | vector in meters

Row spacing between two antenna elements, specified as the comma-separated pair consisting of `'RowSpacing'` and 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 in meters | vector in meters

Column spacing between two antenna elements, specified as the comma-separated pair consisting of 'ColumnSpacing' and 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' | string

Antenna elements spatial arrangement, specified as the comma-separated pair consisting of 'Lattice' and a string.

Example: 'Lattice',Triangular

Data Types: double

'AmplitudeTaper' — Excitation amplitude of antenna elements
1 (default) | scalar | vector

Excitation amplitude of antenna elements , specified as a the comma-separated pair consisting of 'AmplitudeTaper' and 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 in degrees | vector in degrees

Phase shift for antenna elements, specified as the comma-separated pair consisting of 'PhaseShift' and a scalar or vector in degrees.

Example: 'PhaseShift',[3 3 0 0]

Data Types: double

'Tilt' — Tilt angle of array
0 (default) | scalar in degrees | vector in degrees

Tilt angle of the array, specified as the comma-separated pair consisting of 'Tilt' and a scalar or vector in degrees.

Example: 'Tilt',90

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

Data Types: `double`

`'TiltAxis'` — Tilt axis of antenna
[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the antenna, specified as the comma-separated pair consisting of `'TiltAxis'` and:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,“Rotate Antenna and Arrays”

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

Example: `'TiltAxis','Z'`

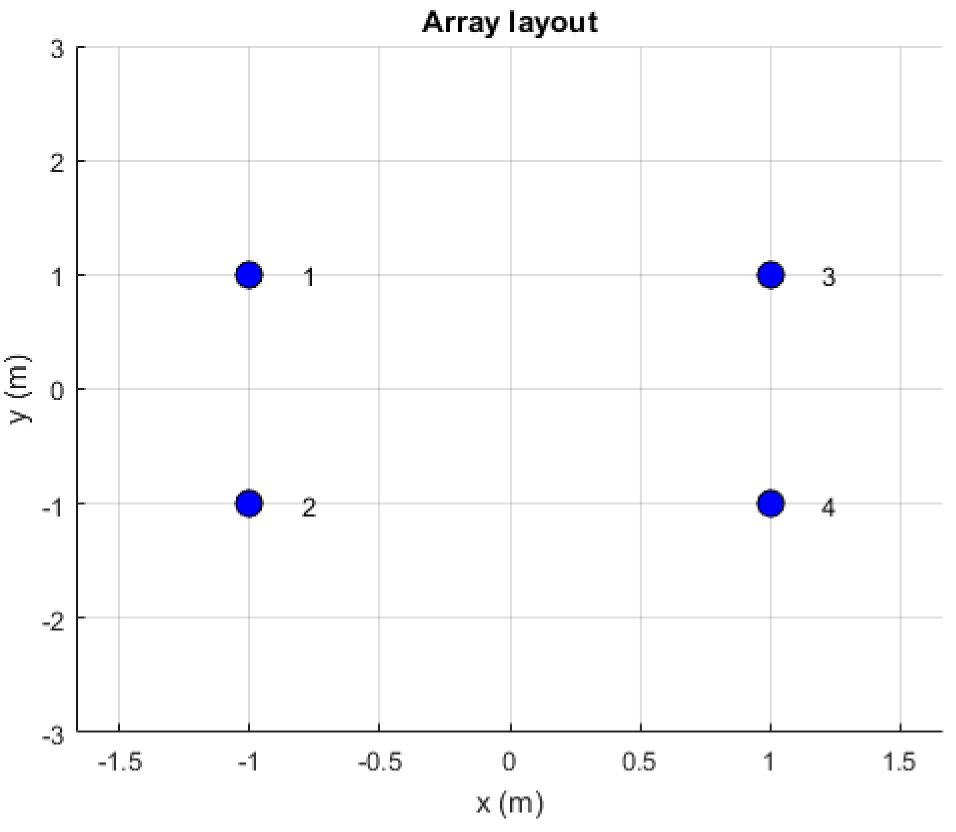
Data Types: `double`

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.

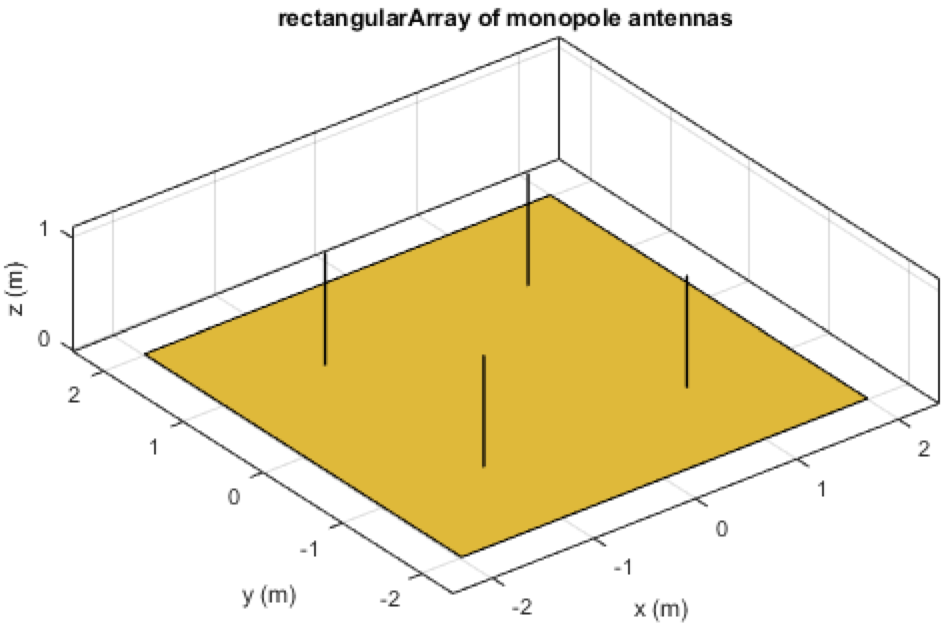
```
h = rectangularArray('Size',[2 2]);
Z = impedance(h,70e6)
```

```
Z =

  25.9763 -54.1987i  25.9763 -54.1995i  25.9763 -54.1987i  25.9763 -54.1995i
```

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

linearArray | infiniteArray

More About

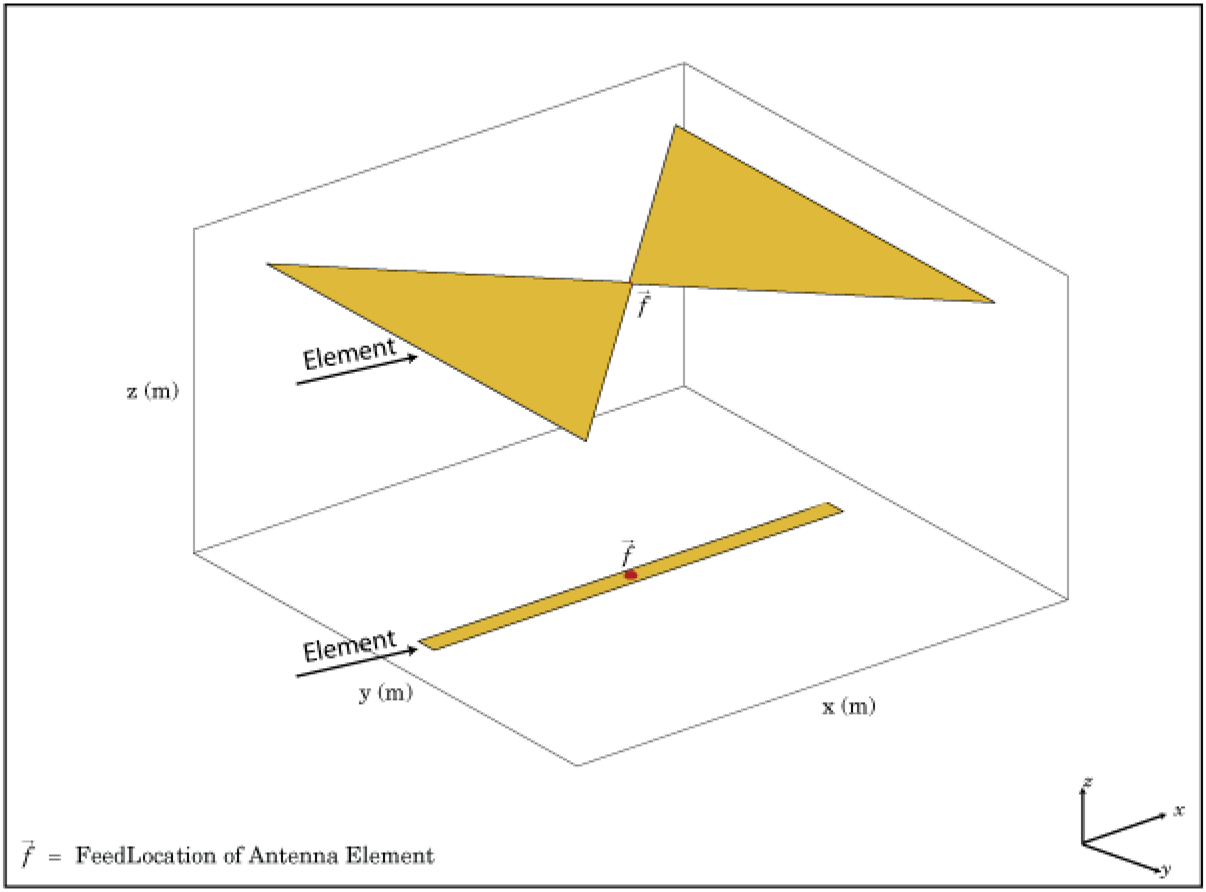
- "Rotate Antenna and Arrays"
- "Antenna Toolbox Limitations"

Introduced in R2015a

conformalArray class

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

Construction

`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 the comma-separated pair consisting of `'ElementPosition'` and *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 handles | cell array of antenna object handles

Individual antenna elements in the array, specified as the comma-separated pair consisting of `'Element'` and one of the following values:

- A scalar
- An array of handles
- Cell array of antenna object handles

By default, conformal array have two antenna elements, the dipole and the bowtie. To specify additional antenna elements, add additional element positions in the conformal array.

Note: You cannot design a conformal array that combines antennas balanced and unbalanced antennas.

Example: `m = monopole; h = conformalArray('Element', [m,m])`. Creates a conformal array consisting of two monopoles antenna elements.

Example: `m = monopole; mt = monopoleTopHat; h = conformalArray('Element', {m,mt})`. Creates a conformal array consisting of a monopole 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 the comma-separated pair consisting of `'Reference'` and either `'origin'` or `'feed'`. For more information see “Position Reference” on page 2-26

Example: `'Reference','origin'`

Data Types: `char`

`'AmplitudeTaper'` — Excitation amplitude of antenna elements
1 (default) | scalar | nonnegative vector

Excitation amplitude of the antenna elements, specified as the comma-separated pair consisting of `'AmplitudeTaper'` and 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 in degrees

Phase shift for antenna elements, specified as the comma-separated pair consisting of `'PhaseShift'` and 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 in degrees | vector in degrees

Tilt angle of the array, specified as a comma-separated pair consisting of `'Tilt'` and a scalar or a vector in degrees.

Example: `'Tilt', 90`

Data Types: `double`

`'TiltAxis'` — Tilt axis of array

[1 0 0] (default) | three-element vector of Cartesian coordinates in meters

Tilt axis of the array, specified as the comma-separated pair consisting of `'TiltAxis'` and one of the following:

- A three-element vector of Cartesian coordinates in meters. In this case, the first element in the three-element vector is the origin and the third element is the 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 for simple rotations around the principle planes, X, Y, or Z.

For more information see,“Rotate Antenna and Arrays”

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

Example: `'TiltAxis','Z'`

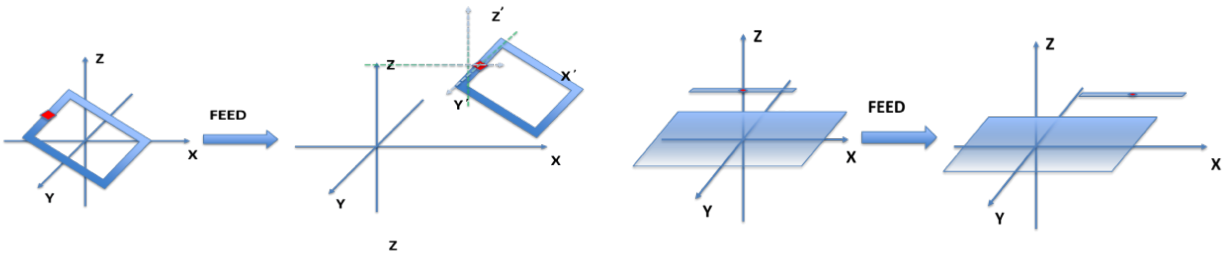
Data Types: `double`

Definitions

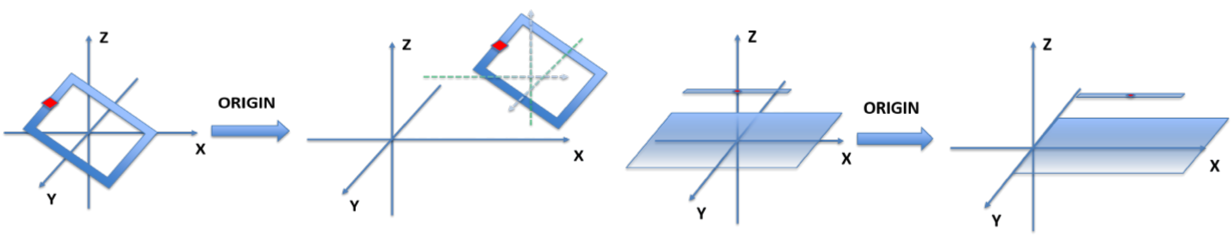
Position Reference

'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:



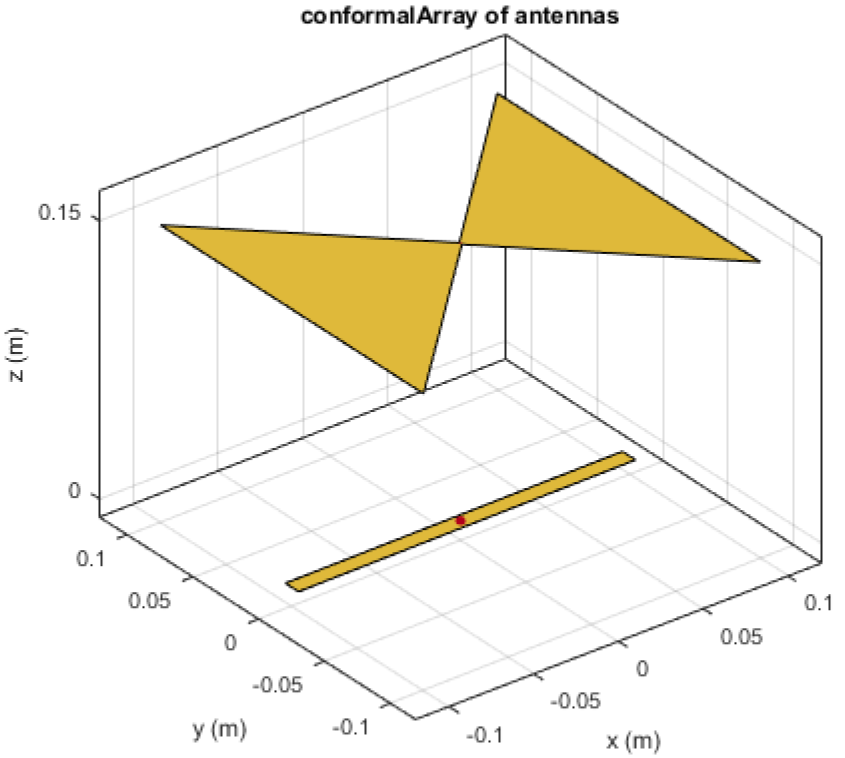
Examples

Default Conformal Array

Create a default conformal array.

```
c = conformalArray
show(c)
```

```
c =

  conformalArray with properties:

            Element: {[1x1 dipole]  [1x1 bowtieTriangular]}
    ElementPosition: [2x3 double]
          Reference: 'feed'
     AmplitudeTaper: 1
         PhaseShift: 0
               Tilt: 0
           TiltAxis: [1 0 0]
```



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);
```

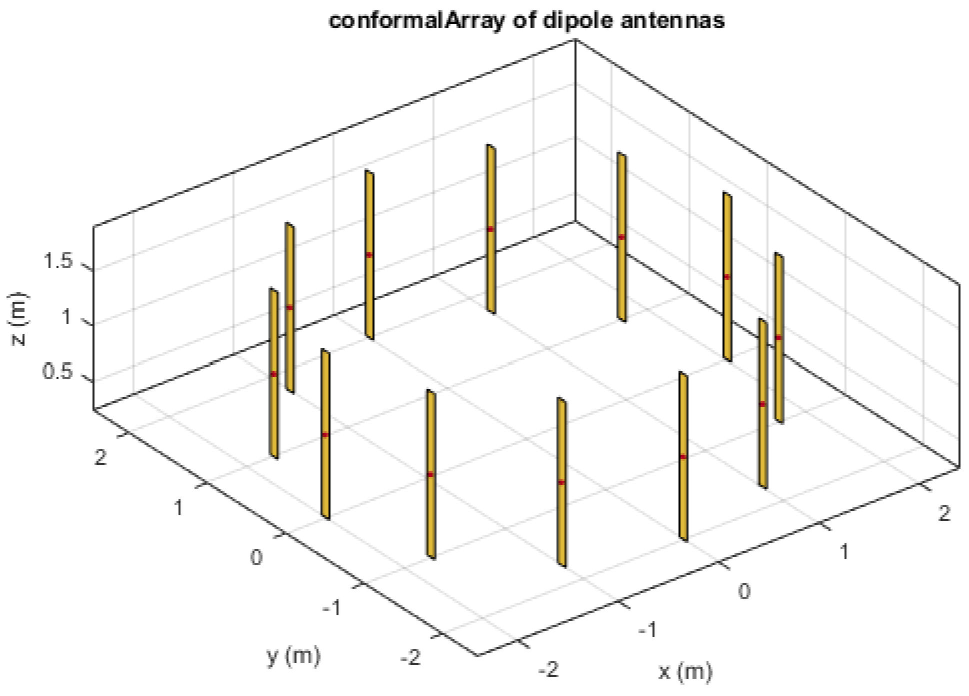
Define the x,y,z values for the element positions in the array.

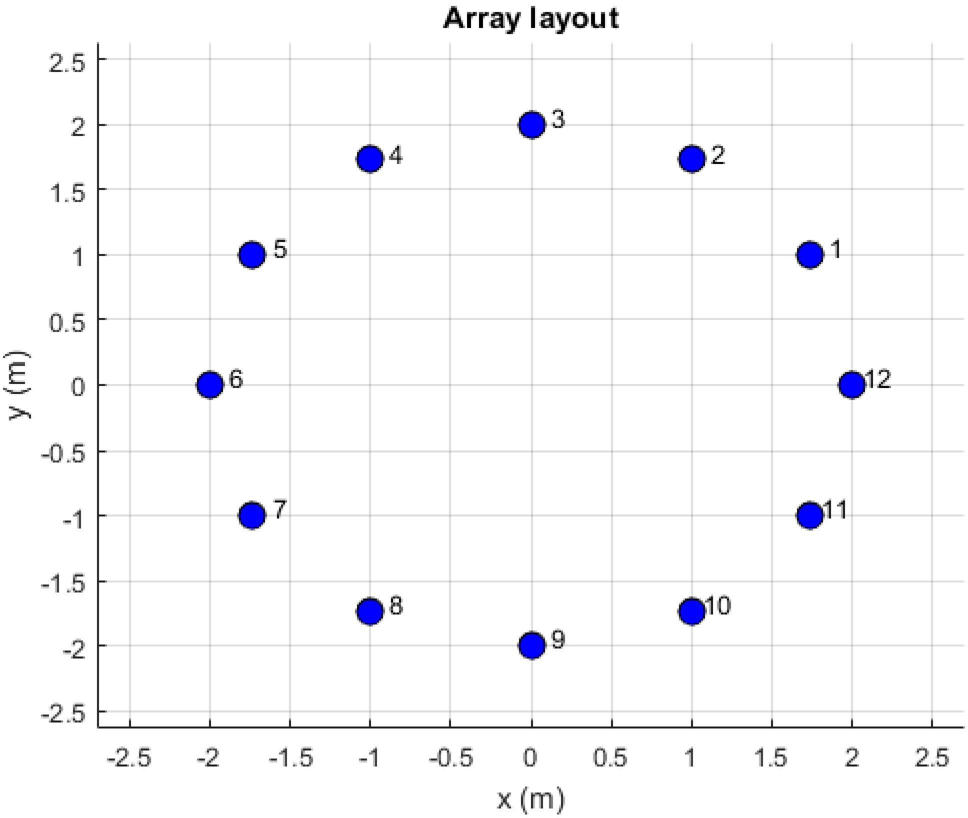
```
del_th = 360/N;
```

```
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.

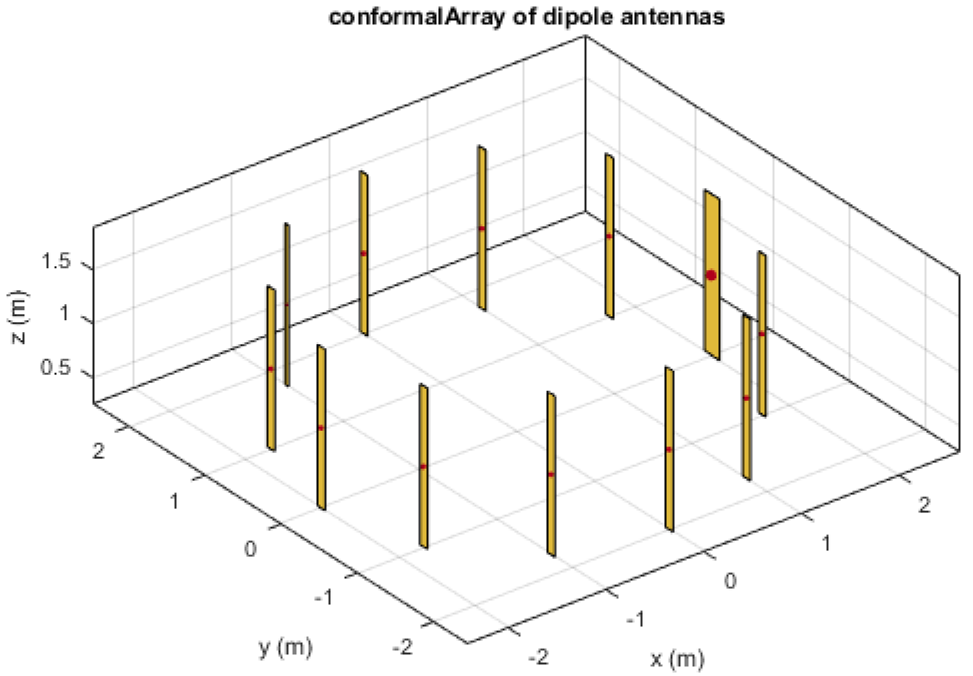
```
c = conformalArray('Element',elem,'ElementPosition',pos');
show(c)
figure
layout(c)
```





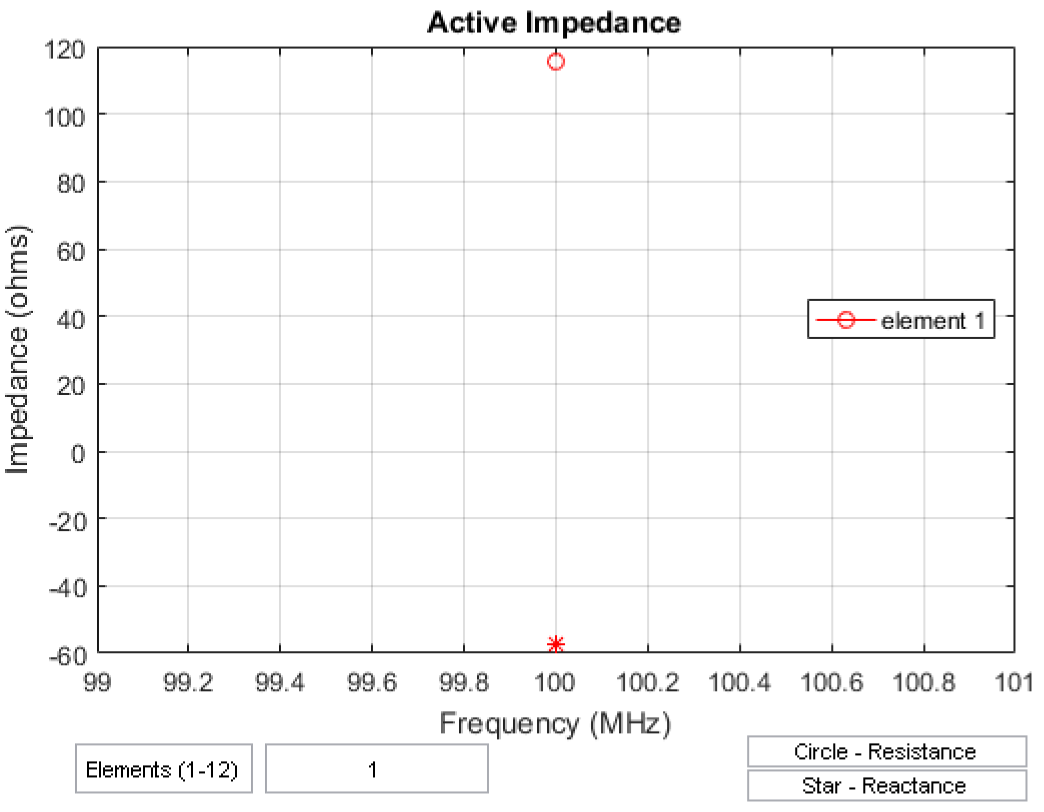
Change the width of the fourth and the twelfth element of the circular array. Visualize the new arrangement.

```
c.Element(4).Width = 0.05;
c.Element(12).Width = 0.2;
figure
show(c)
```

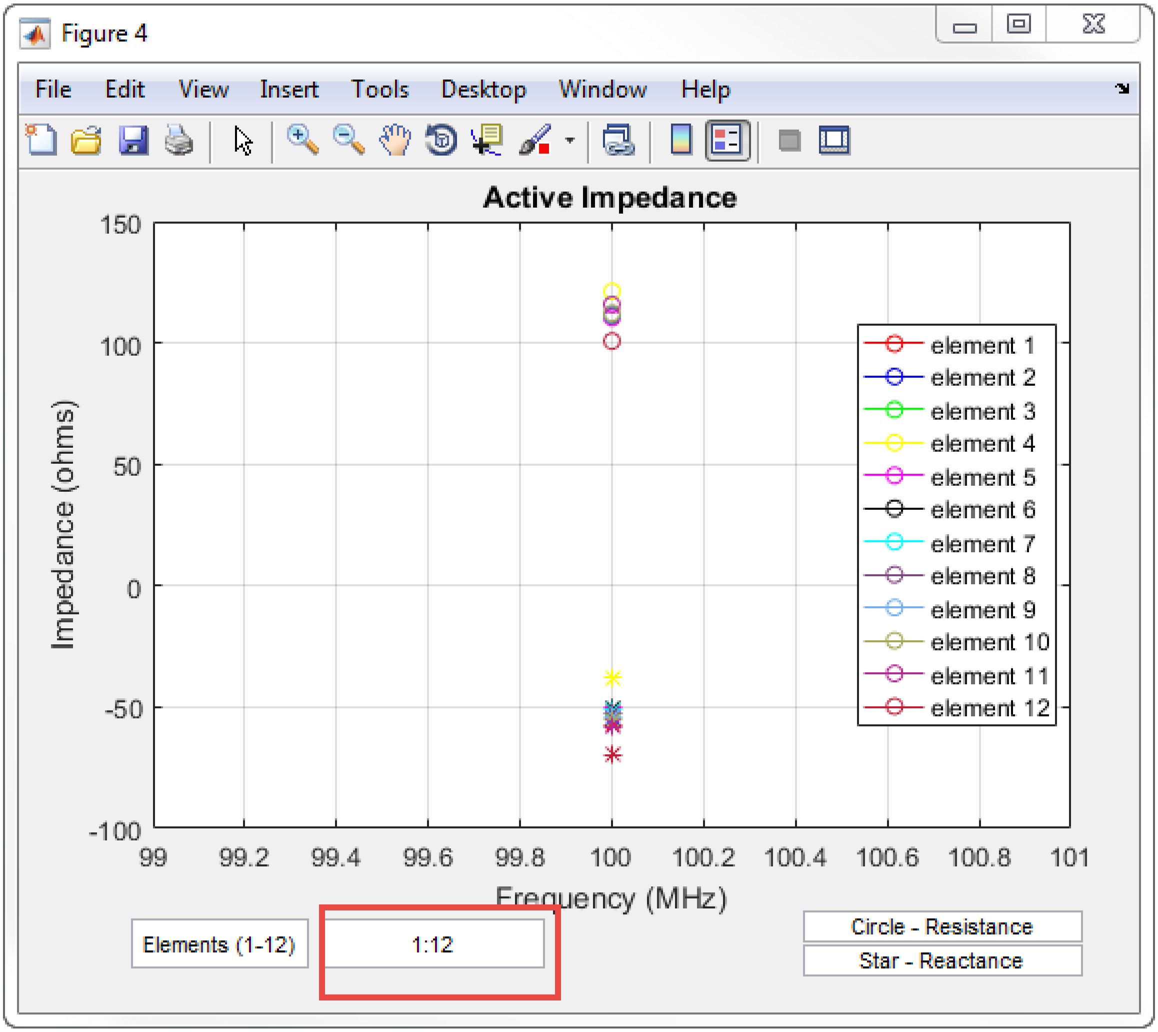


Calculate and plot the impedance of the circular array at 100 MHz.The plot shows the impedance of the first element in the array.

```
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.



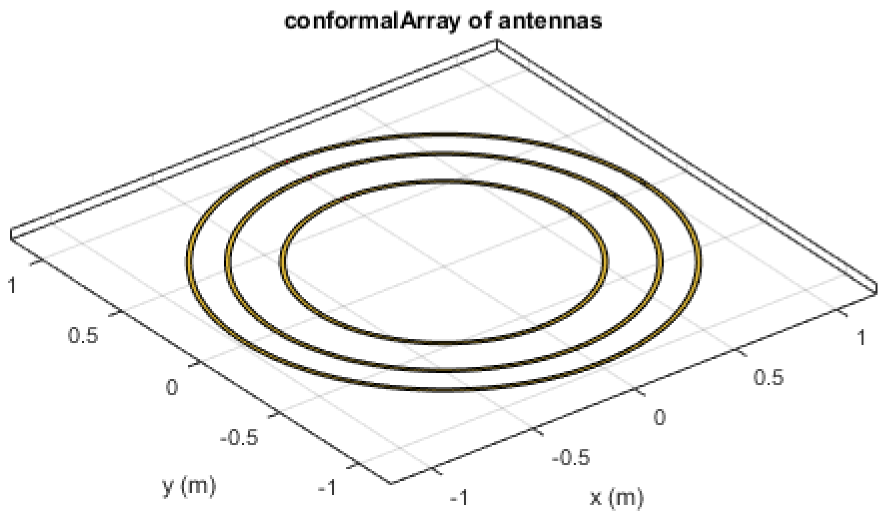
Radiation Pattern of Concentric Array of Circular Loop Antennas

Define three circular loop antennas of radii 0.6366 m(default), 0.85 m, and 1 m, repectively.

```
l1 = loopCircular;
l2 = loopCircular('Radius',0.85);
l3 = loopCircular('Radius',1);
```

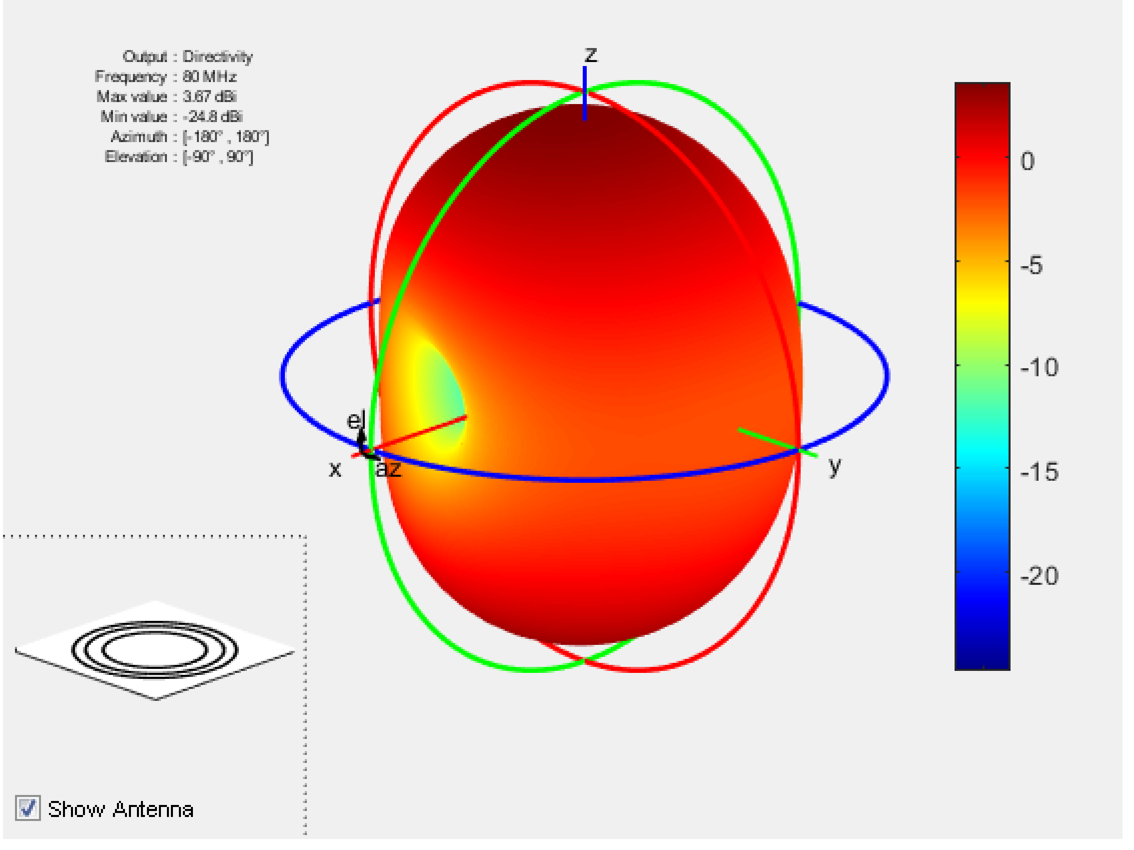
Create a concentric array that uses the origin of circular loop antennas as its position reference.

```
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');
```

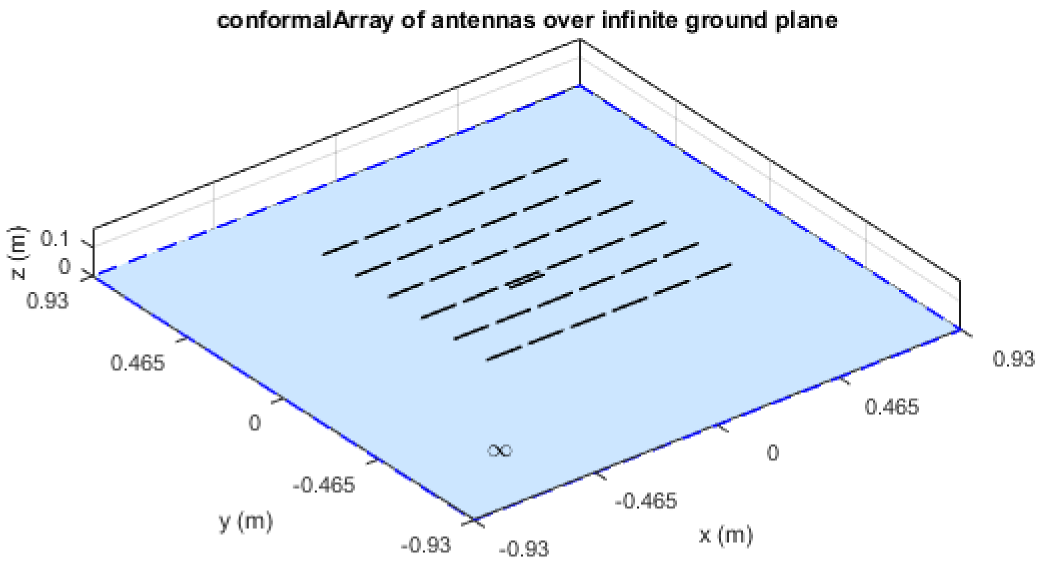
Create an infinite groundplane reflector antenna using the dipole as exciter.

```
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)
```



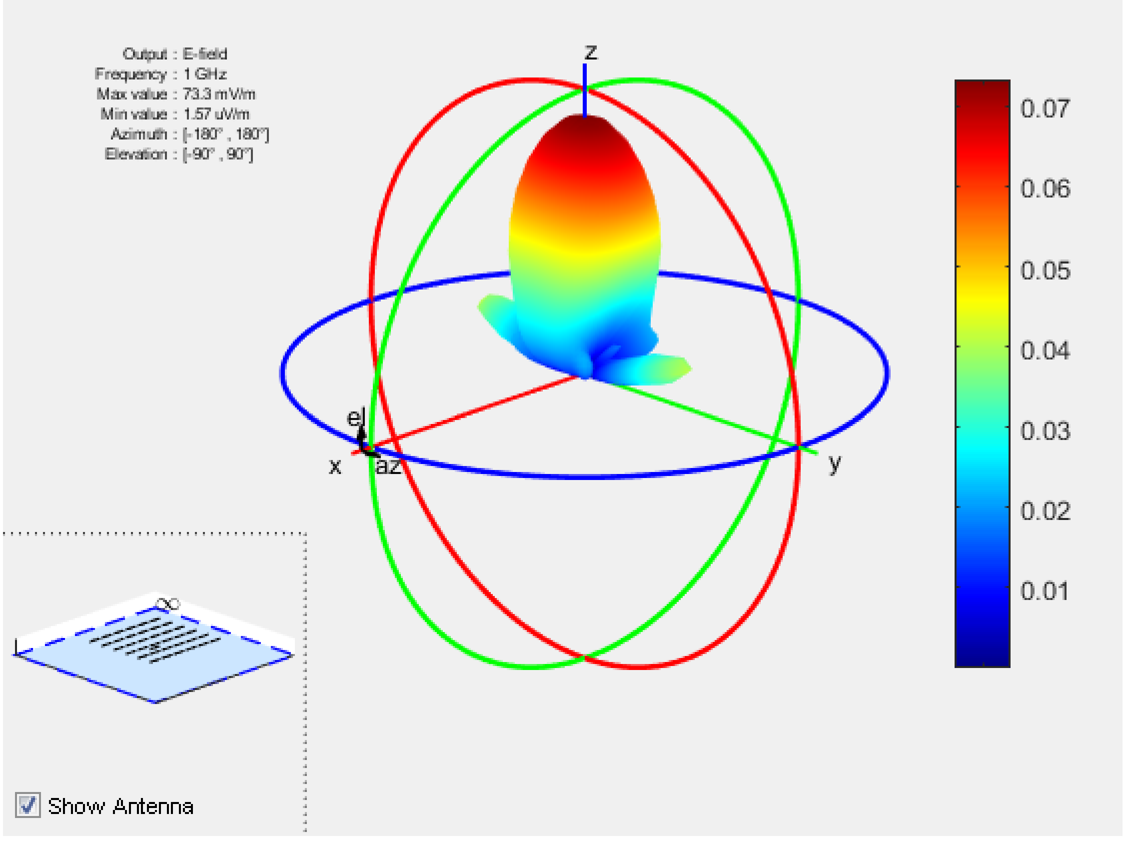
Drive only the reflector antenna with an amplitude of 1.

```
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.

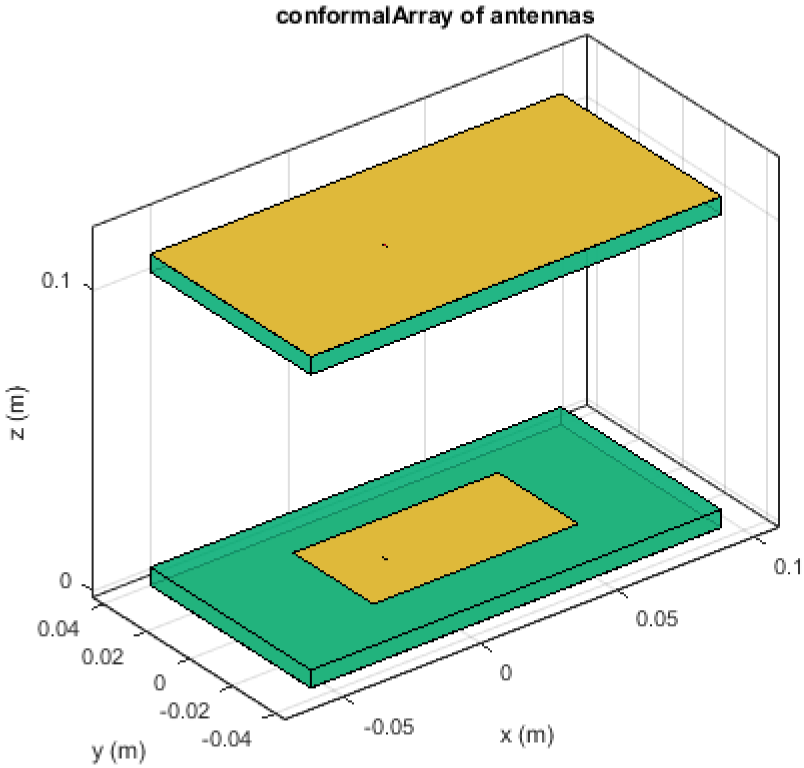
```
d = dielectric('FR4');
```

```
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})
show(c)
```

```
c =

  conformalArray with properties:

            Element: {[1x1 patchMicrostrip]  [1x1 patchMicrostrip]}
    ElementPosition: [2x3 double]
          Reference: 'feed'
     AmplitudeTaper: 1
         PhaseShift: 0
               Tilt: 0
           TiltAxis: [1 0 0]
```



References

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

See Also

linearArray | rectangularArray

More About

- "Rotate Antenna and Arrays"

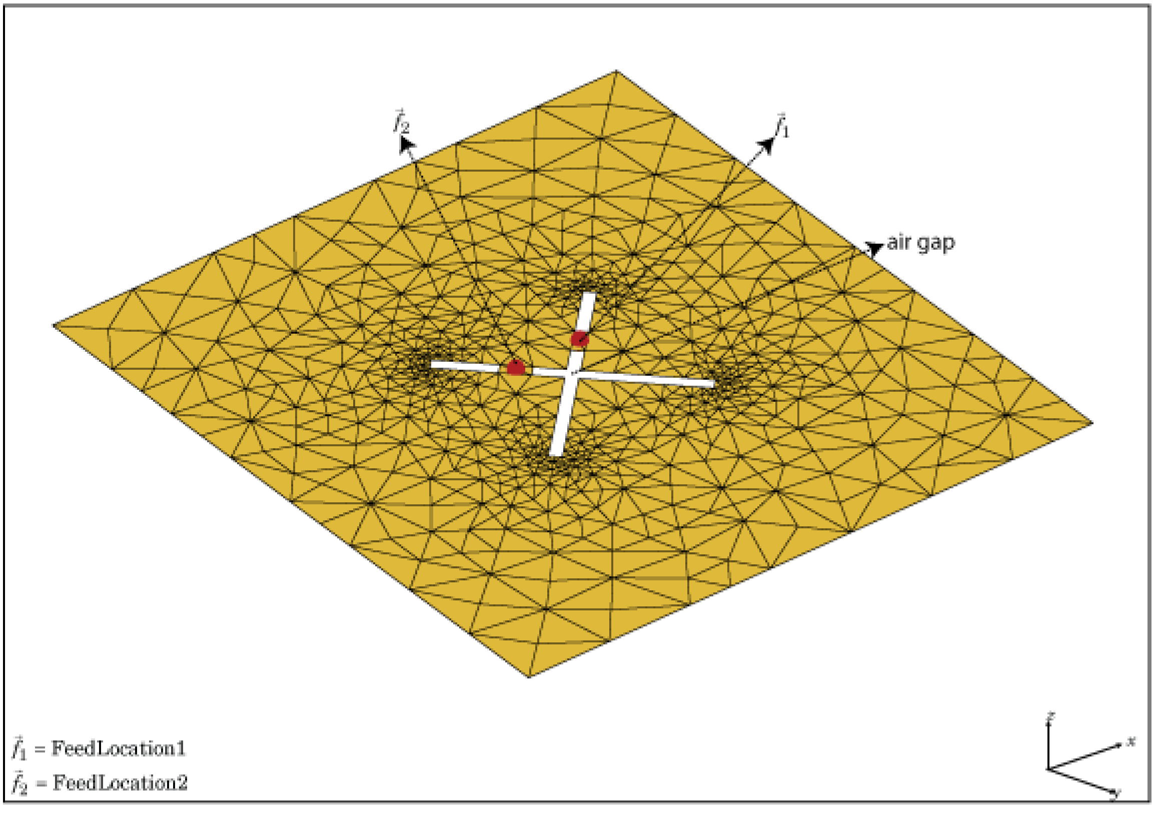
- “Antenna Toolbox Limitations”

Introduced in R2016a

customArrayMesh class

Create 2-D custom mesh array on X-Y plane

Description



The `customArrayMesh` class 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.

Construction

`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 in meters

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 in meters

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 the comma-separated pair consisting of `'AmplitudeTaper'` and 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 in degrees

Phase shift for antenna elements, specified as the comma-separated pair consisting of `'PhaseShift'` and 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`

Methods

`createFeed`	Create feed locations for custom array

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 =

  35.6701 -35.4284i  35.9773 -24.7044i
```

See Also

linearArray | rectangularArray | conformalArray

Introduced in R2016a

Methods — Alphabetical List

addCursor
animate
createLabels
findLobes
replace
showPeaksTable
showSpan

createFeed

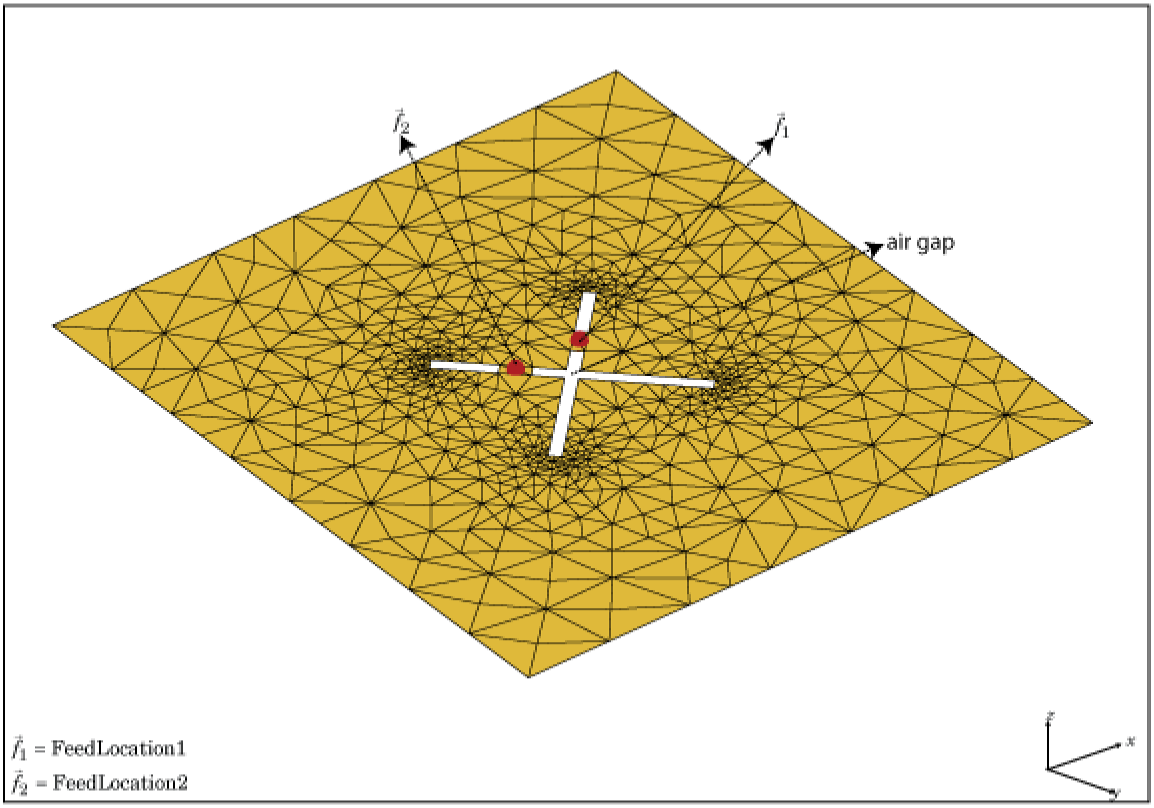
Class: customArrayMesh

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.

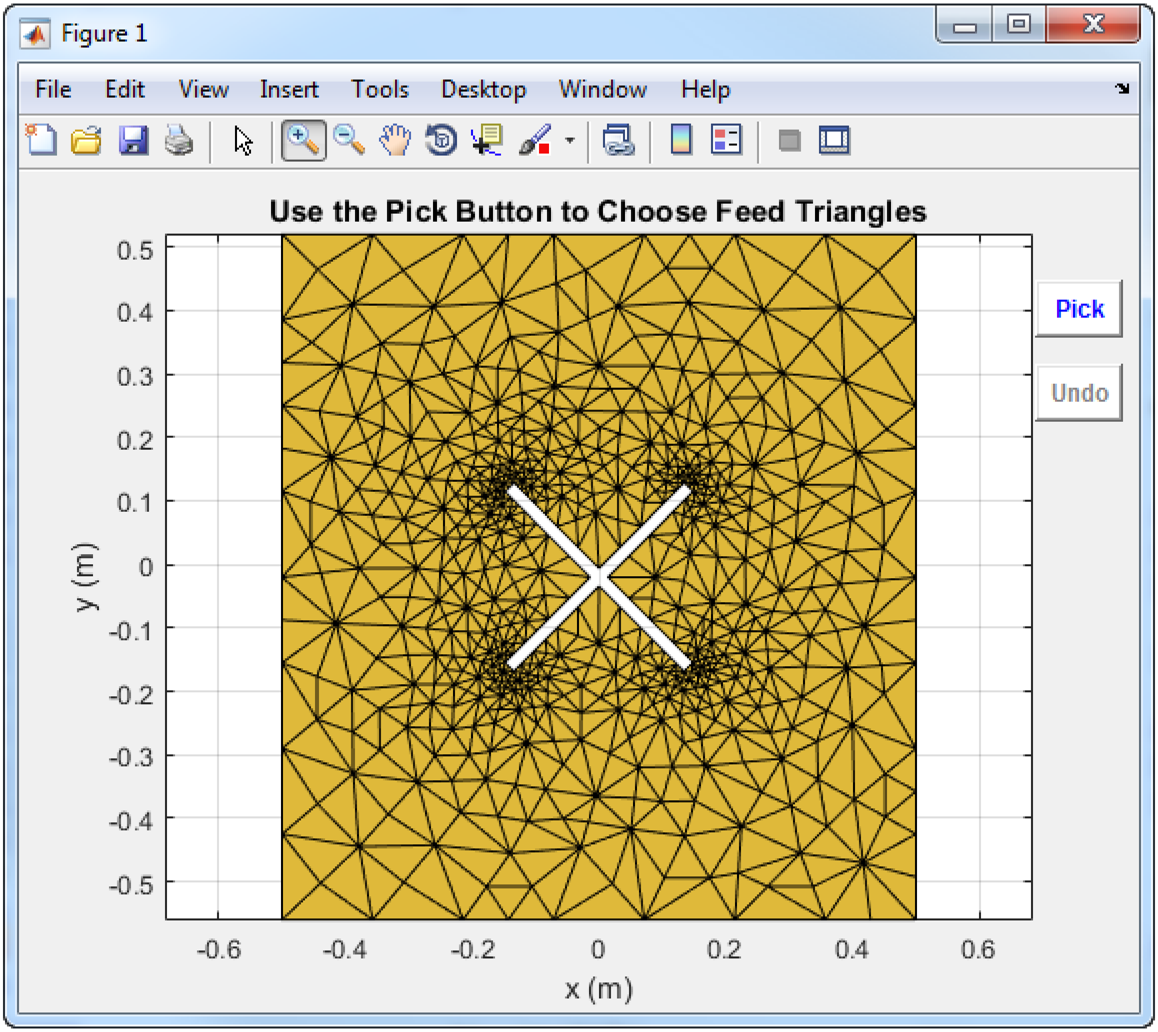
Load a 2-D custom mesh. Create a custom array using the points and triangles.

```
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]
```

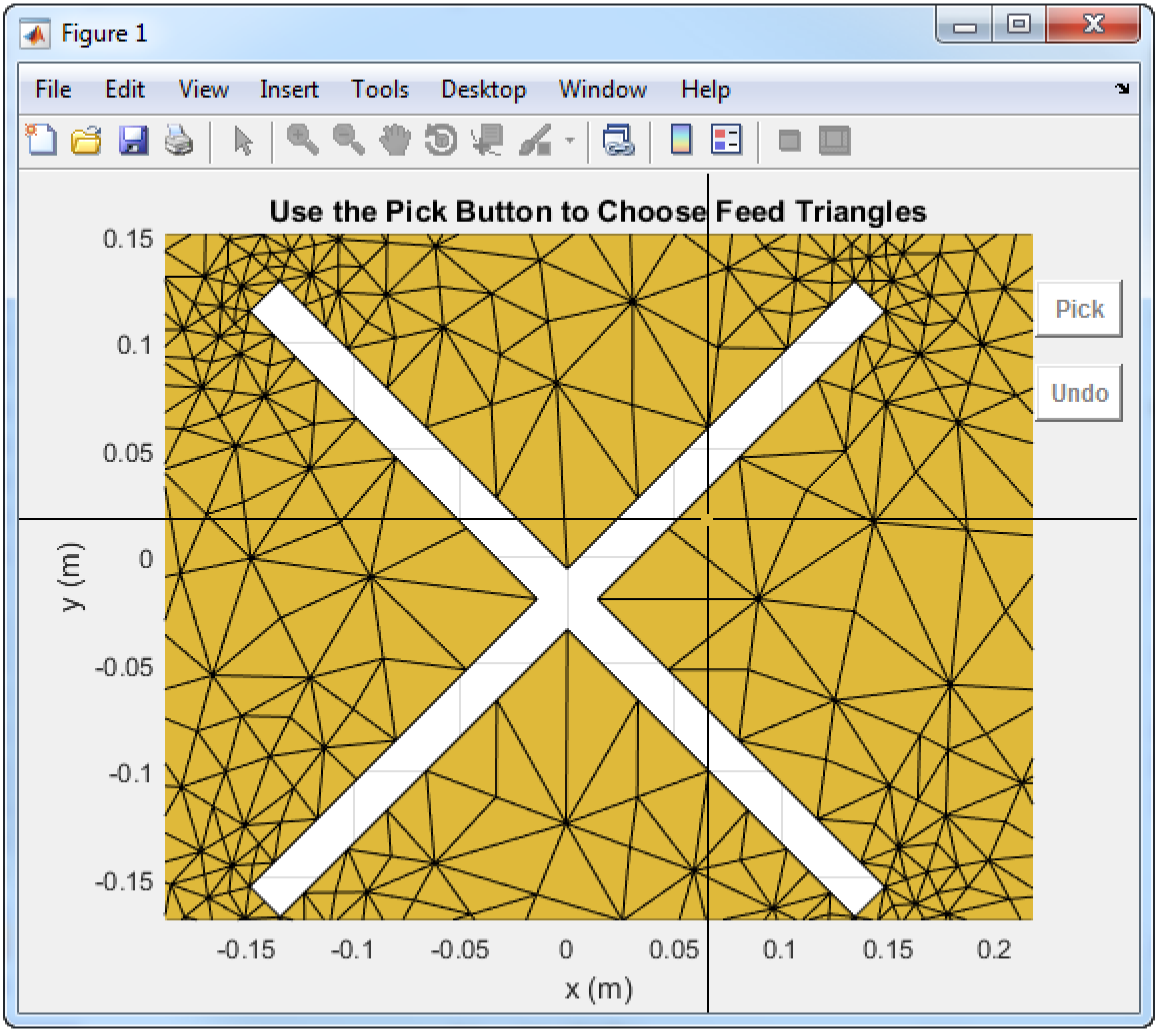
Use the `createFeed` function to view the array mesh structure. In this array mesh view, you see **Pick** and **Undo** buttons. The **Pick** button is highlighted.

```
createFeed(c)
```

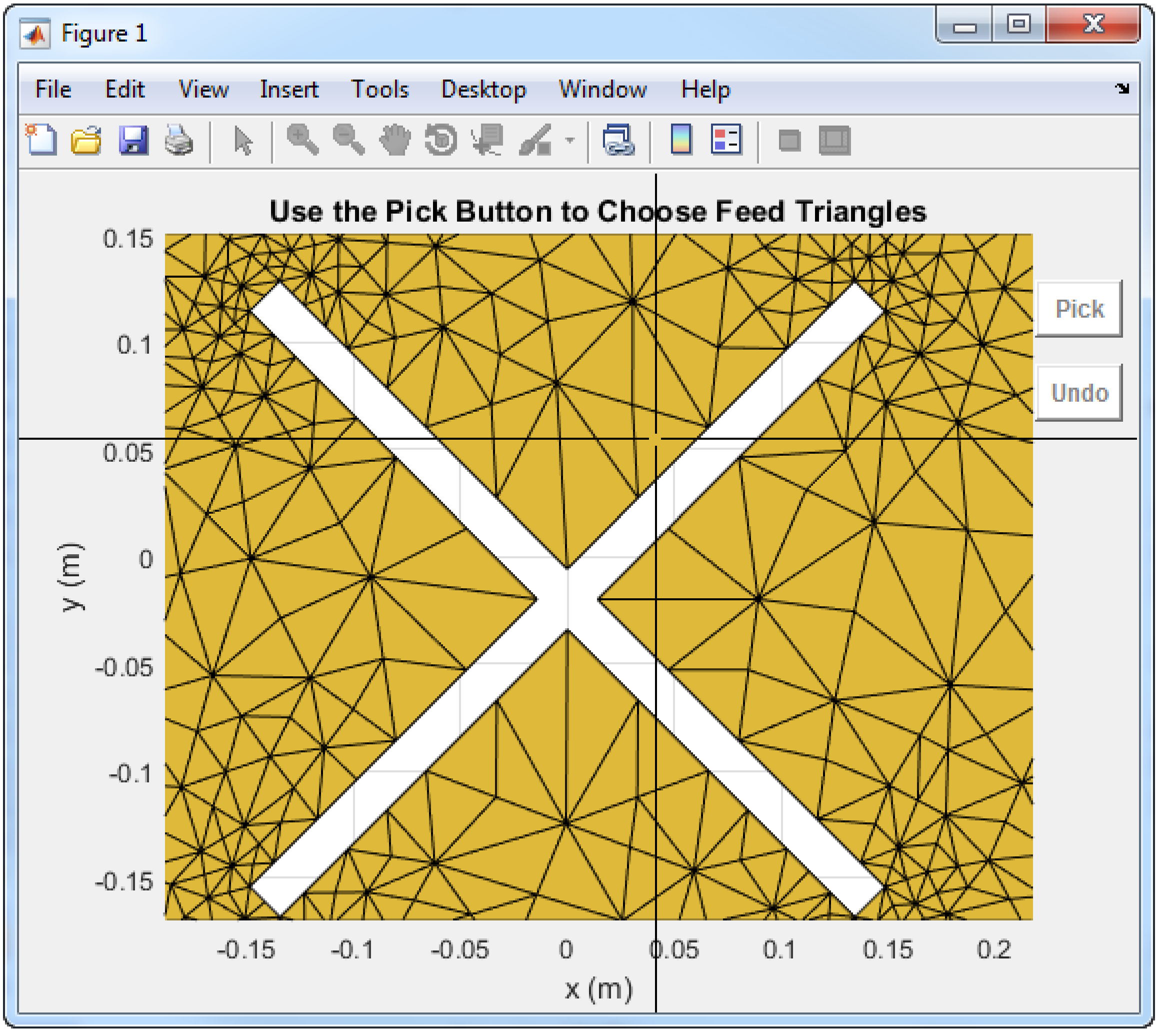


Click **Pick** to display the cross hairs. For an array with two feeds, select two pairs (four points) in the mesh. To specify a feed-region for the, zoom in and select two points each, one inside each triangle on either side of the air gap. Select the points using the cross hairs.

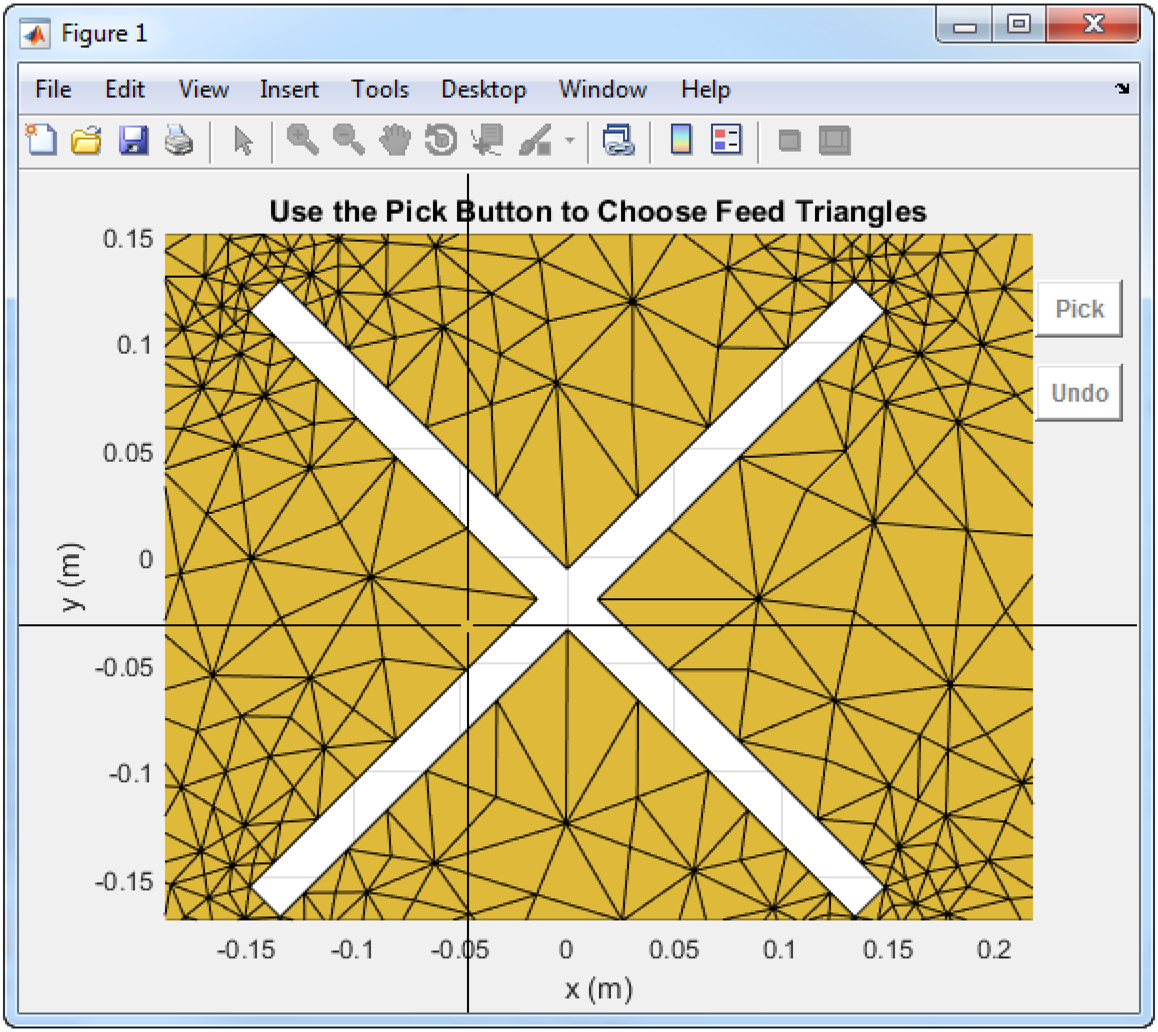
- Select the first triangle for feedpoint 1.



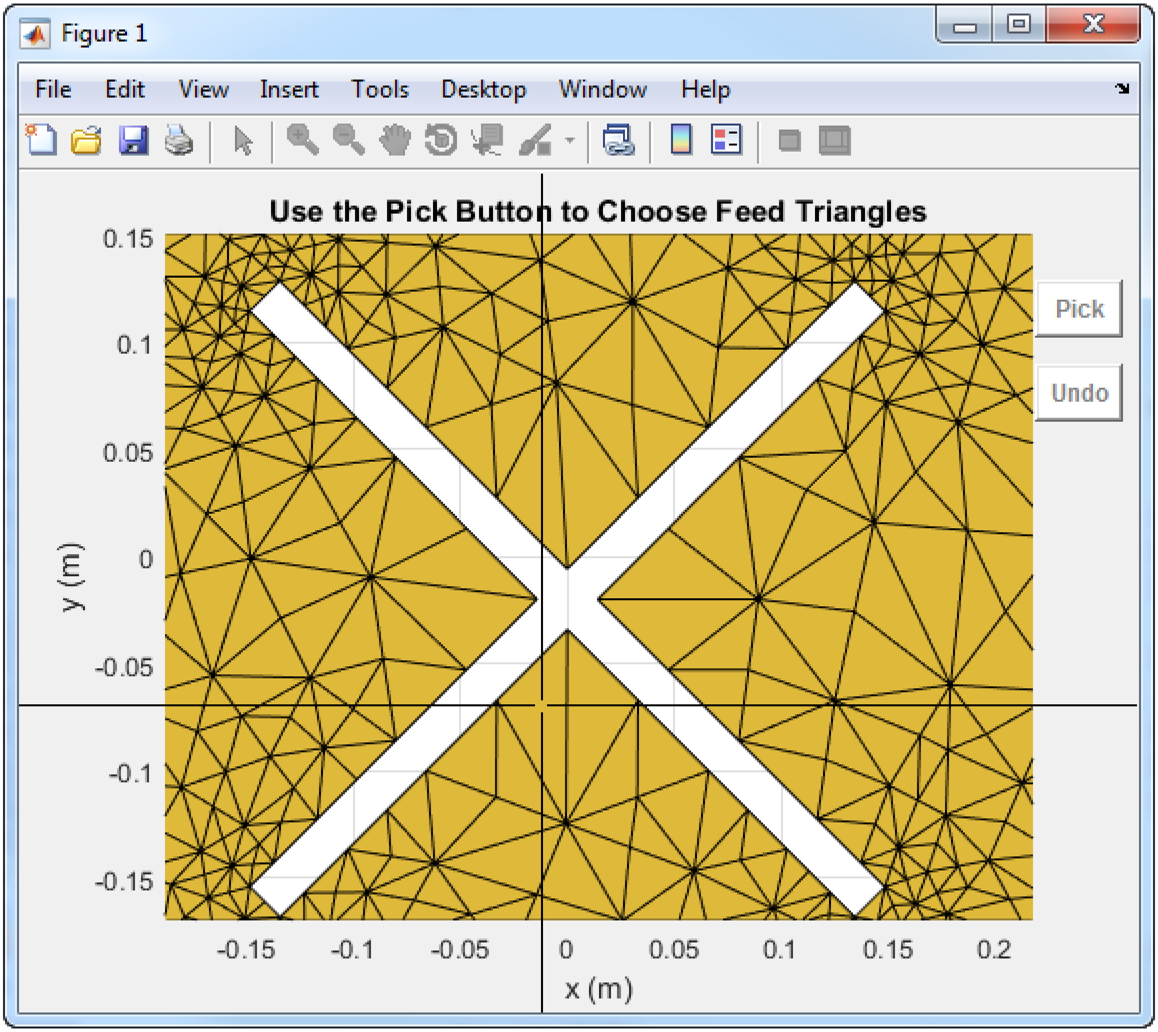
- Select the second triangle on the other side of the air gap for feedpoint 1.



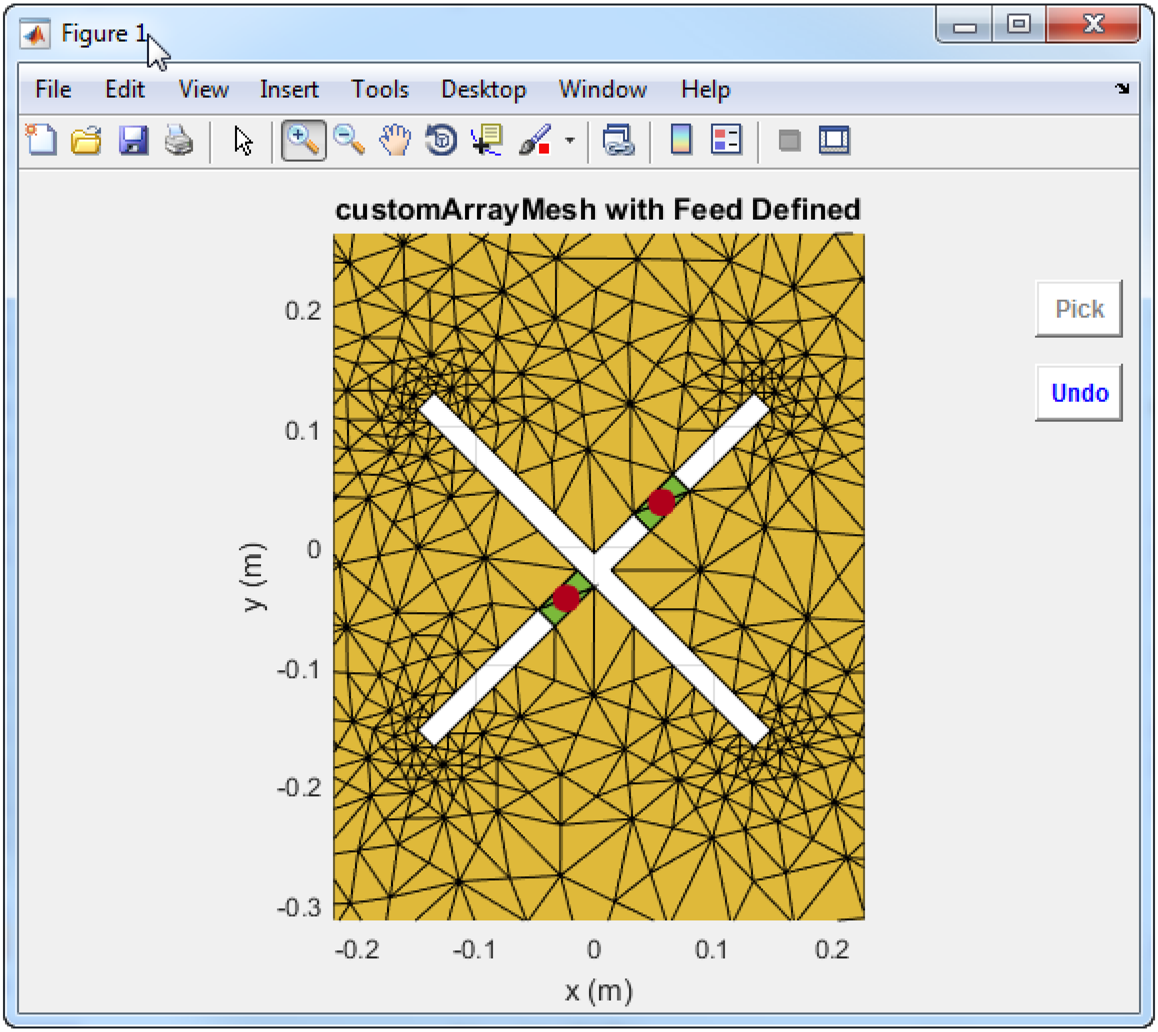
- Select first triangle for feedpoint 2.



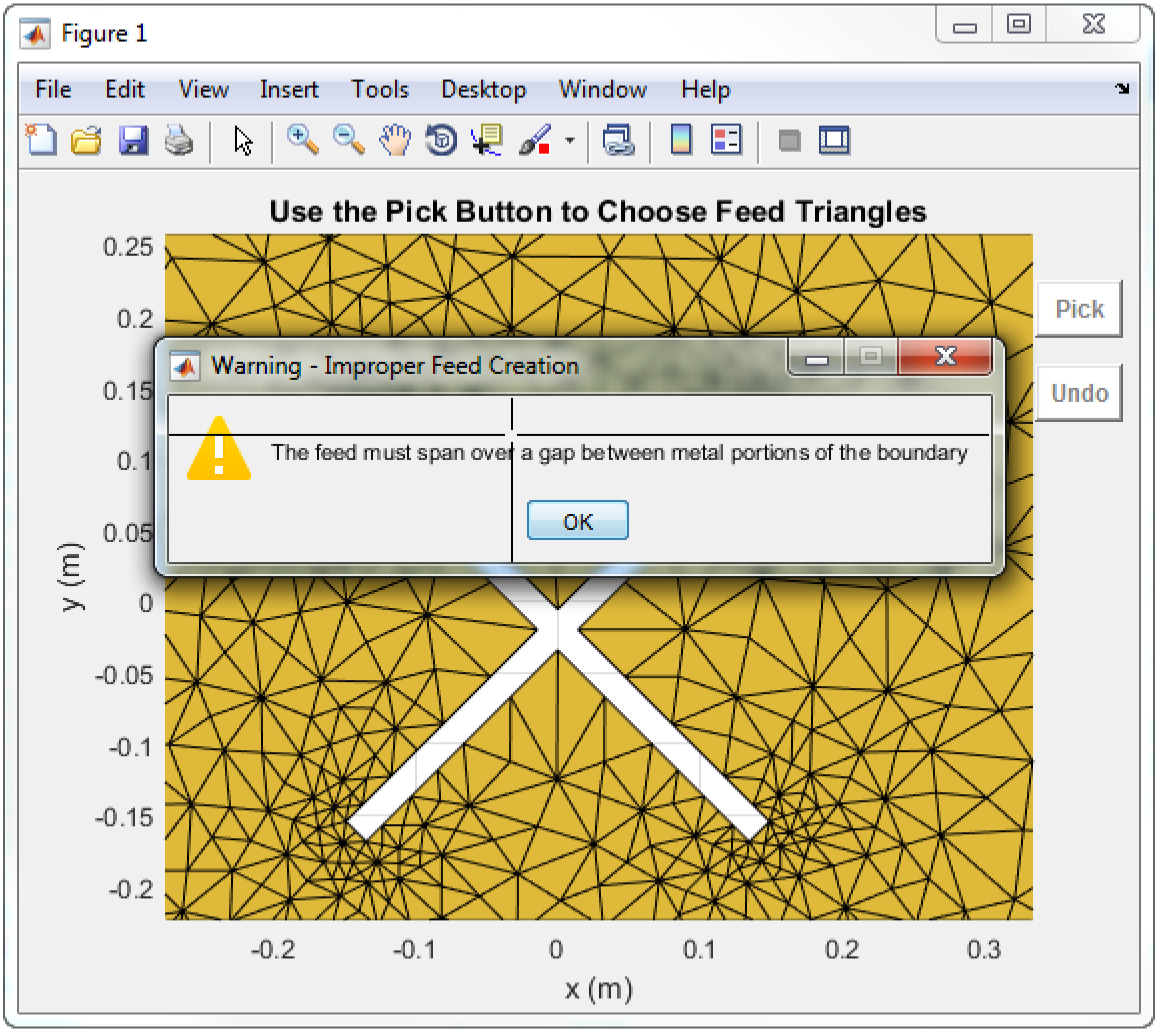
- Select the second triangle on the other side of the air gap for feedpoint 2.



Selecting the fourth triangle creates and displays the array feeds.



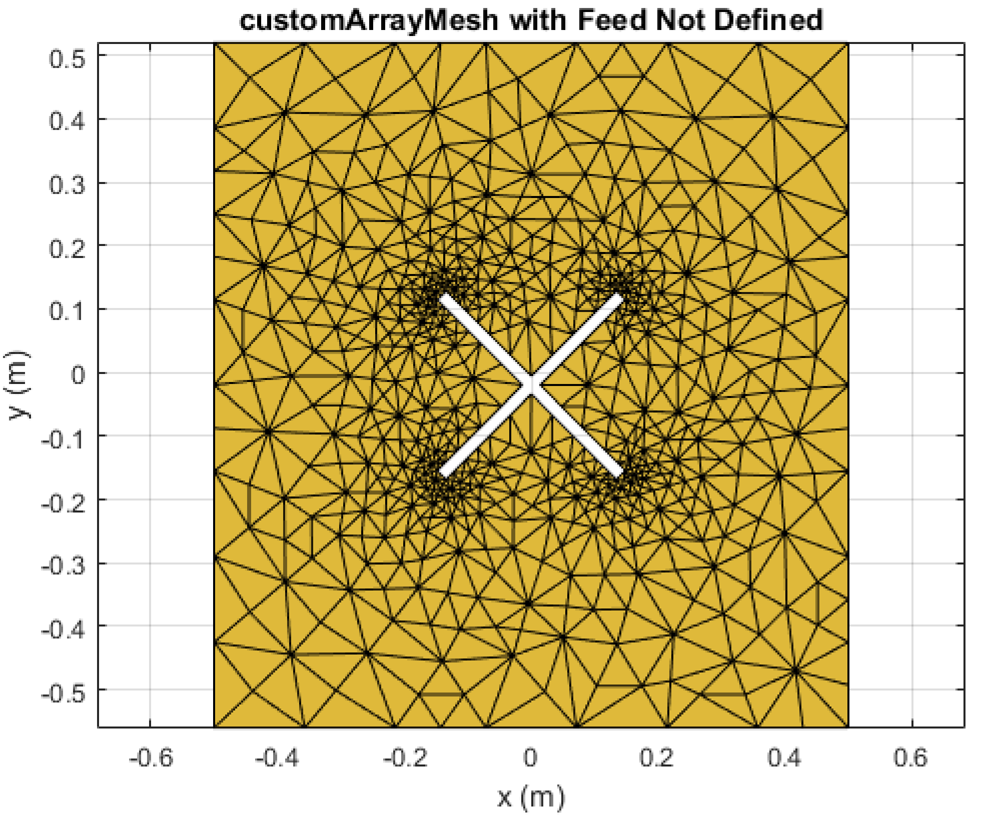
You must select the two triangles on either side of the air gap. Otherwise, the function displays an error message.



Create Feed for Custom Array Mesh

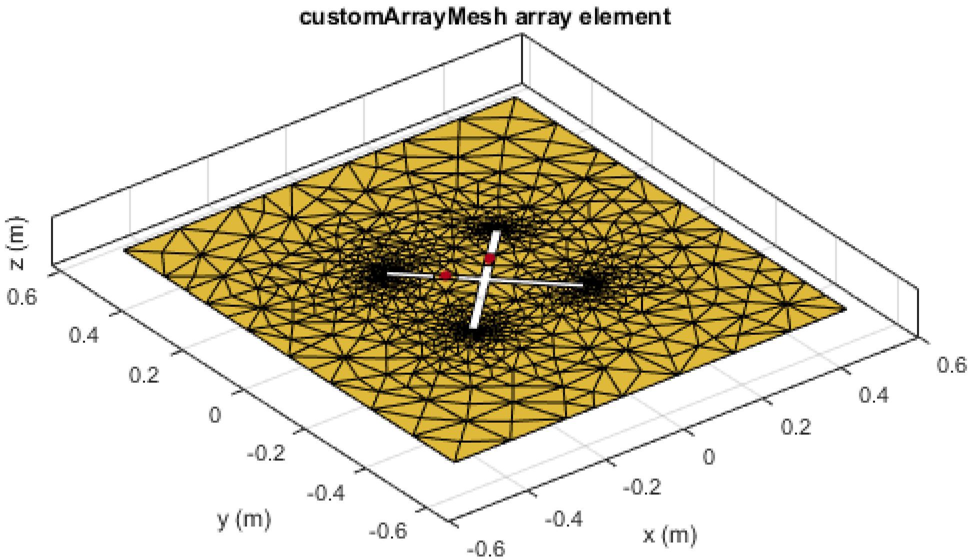
Load a custom mesh and create an array.

```
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.

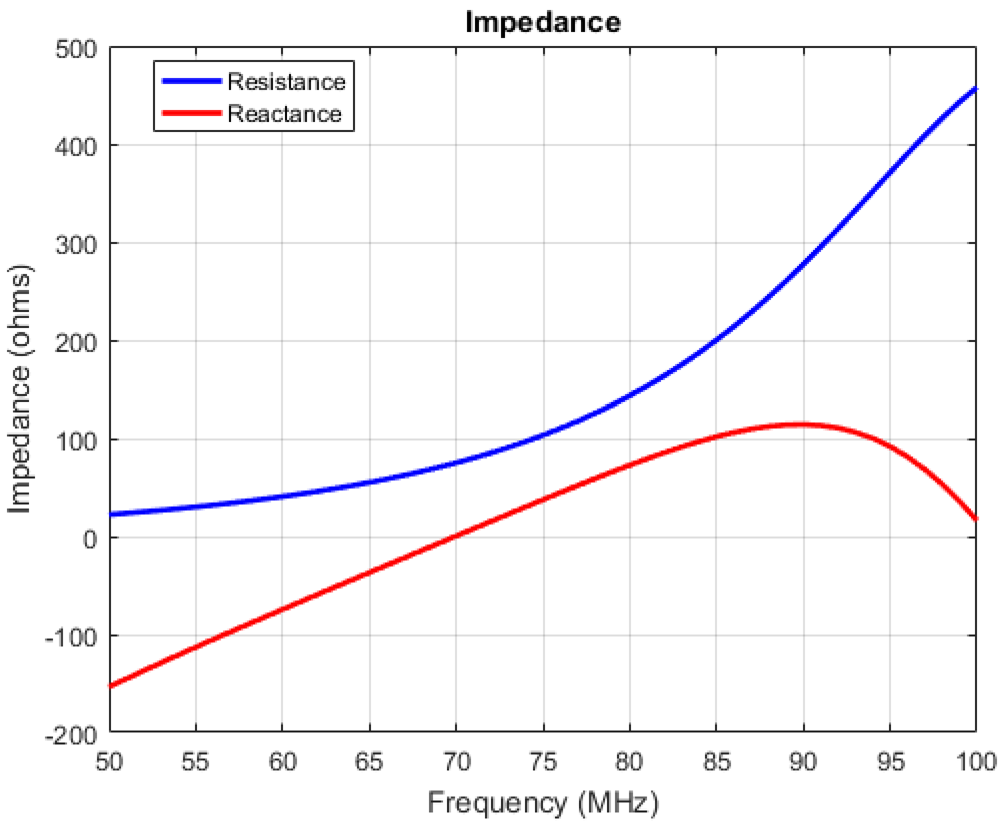
`z = impedance(array,frequency,elementnumber)` returns the scan impedance of a specified antenna element in an array.

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 =

   1.0e+02 *

   0.2751 - 1.6565i   0.2751 - 1.6565i
```

```
0.2864 - 1.5802i   0.2864 - 1.5802i
0.2979 - 1.5055i   0.2979 - 1.5055i
0.3097 - 1.4322i   0.3097 - 1.4322i
0.3218 - 1.3601i   0.3218 - 1.3601i
0.3343 - 1.2893i   0.3343 - 1.2893i
0.3471 - 1.2194i   0.3471 - 1.2194i
0.3603 - 1.1504i   0.3603 - 1.1504i
0.3739 - 1.0821i   0.3739 - 1.0821i
0.3879 - 1.0145i   0.3879 - 1.0145i
0.4024 - 0.9474i   0.4024 - 0.9474i
0.4175 - 0.8806i   0.4175 - 0.8806i
0.4331 - 0.8141i   0.4331 - 0.8141i
0.4493 - 0.7477i   0.4493 - 0.7477i
0.4663 - 0.6813i   0.4663 - 0.6813i
0.4840 - 0.6148i   0.4840 - 0.6148i
0.5025 - 0.5480i   0.5025 - 0.5480i
0.5219 - 0.4808i   0.5219 - 0.4808i
0.5424 - 0.4131i   0.5424 - 0.4131i
0.5640 - 0.3447i   0.5640 - 0.3447i
0.5869 - 0.2755i   0.5869 - 0.2755i
0.6111 - 0.2054i   0.6111 - 0.2054i
0.6370 - 0.1341i   0.6370 - 0.1341i
0.6645 - 0.0616i   0.6645 - 0.0616i
0.6941 + 0.0124i   0.6941 + 0.0124i
0.7258 + 0.0879i   0.7258 + 0.0879i
0.7599 + 0.1653i   0.7599 + 0.1653i
0.7969 + 0.2446i   0.7969 + 0.2446i
0.8369 + 0.3260i   0.8369 + 0.3260i
0.8805 + 0.4098i   0.8805 + 0.4098i
0.9281 + 0.4961i   0.9281 + 0.4961i
0.9801 + 0.5851i   0.9801 + 0.5851i
1.0374 + 0.6770i   1.0374 + 0.6770i
1.1004 + 0.7720i   1.1004 + 0.7720i
1.1701 + 0.8701i   1.1701 + 0.8701i
1.2475 + 0.9715i   1.2475 + 0.9715i
1.3336 + 1.0763i   1.3336 + 1.0763i
1.4298 + 1.1843i   1.4298 + 1.1843i
1.5375 + 1.2955i   1.5375 + 1.2955i
1.6585 + 1.4096i   1.6585 + 1.4096i
1.7948 + 1.5258i   1.7948 + 1.5258i
1.9488 + 1.6435i   1.9488 + 1.6435i
2.1232 + 1.7612i   2.1232 + 1.7612i
2.3208 + 1.8769i   2.3208 + 1.8769i
2.5451 + 1.9881i   2.5451 + 1.9881i
```

```
2.7996 + 2.0906i   2.7996 + 2.0906i
3.0878 + 2.1794i   3.0878 + 2.1794i
3.4130 + 2.2473i   3.4130 + 2.2473i
3.7776 + 2.2849i   3.7776 + 2.2849i
4.1824 + 2.2807i   4.1824 + 2.2807i
4.6248 + 2.2203i   4.6248 + 2.2203i
```

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.

See Also

`returnLoss`

Introduced in R2015a

sparameters

S-parameter object

Syntax

```
obj = sparameters(antenna,freq,Z0 )
obj = sparameters(array,freq,Z0 )
```

Description

`obj = sparameters(antenna,freq,Z0 )` calculates the complex s-parameters for an `antenna` object over specified frequency values and for a given reference impedance, `Z0`

`obj = 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;
sparameters (h, 70e6)
```

```
ans =

  sparameters: S-parameters object

       NumPorts: 1
    Frequencies: 70000000
     Parameters: 0.2000 + 0.0042i
      Impedance: 50

  rfparam(obj,i,j) returns S-parameter Sij
```

Calculate S-parameter Matirx 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
```

- "Bisect S-Parameters of Cascaded Probes"

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.

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. The function uses this input argument to set the value of the `Frequencies` property of `hs`.

Z0 — Reference impedance
50 (default) | positive real scalar

Reference impedance in ohms, specified as a positive real scalar. The function uses this input argument to set the value of the `Impedance` property of `hs`. You cannot specify `Z0` if you are importing data from a file. The argument Z0 is optional and will be stored in the `Impedance` property.

When making a deep copy of an S-parameter object, this input argument is not supported. To change the reference impedance of an S-parameters object, use `newref`.

Output Arguments

obj — S-parameter data
scalar handle

S-parameter data, returned as a scalar handle. `disp(hs)` 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

`correlation` | `impedance` | `rfparam` | `rfplot`

rfparam

Extract vector of network parameters

Syntax

```
n_ij = rfparam(hnet,i,j)
abcd_vector = rfparam(habcd,abcdflag)
```

Description

`n_ij = rfparam(hnet,i,j)` extracts the network parameter vector (*i*,*j*) from the network parameter object, `hnet`.

`abcd_vector = rfparam(habcd,abcdflag)` extracts the *A*, *B*, *C*, or *D* vector from ABCD-parameter object, `habcd`.

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 =

  -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
  -0.3847 + 1.9219i
```

```
  -0.3542 + 1.9339i
  -0.3236 + 1.9455i
  -0.2930 + 1.9566i
  -0.2623 + 1.9674i
  -0.2316 + 1.9779i
  -0.2008 + 1.9882i
  -0.1698 + 1.9983i
  -0.1387 + 2.0084i
  -0.1073 + 2.0185i
  -0.0758 + 2.0286i
  -0.0441 + 2.0387i
  -0.0124 + 2.0488i
   0.0194 + 2.0588i
   0.0513 + 2.0687i
   0.0834 + 2.0785i
   0.1158 + 2.0882i
   0.1484 + 2.0977i
   0.1813 + 2.1072i
   0.2145 + 2.1164i
   0.2482 + 2.1256i
   0.2821 + 2.1344i
   0.3161 + 2.1430i
   0.3504 + 2.1513i
   0.3849 + 2.1595i
   0.4197 + 2.1676i
   0.4550 + 2.1757i
   0.4908 + 2.1839i
   0.5272 + 2.1922i
   0.5642 + 2.2007i
   0.6020 + 2.2095i
   0.6403 + 2.2186i
   0.6792 + 2.2281i
   0.7186 + 2.2377i
   0.7587 + 2.2476i
   0.7994 + 2.2575i
   0.8410 + 2.2675i
   0.8833 + 2.2774i
   0.9266 + 2.2871i
   0.9708 + 2.2967i
   1.0161 + 2.3061i
   1.0623 + 2.3152i
   1.1091 + 2.3243i
   1.1567 + 2.3333i
   1.2053 + 2.3423i
```

```
1.2551 + 2.3512i
1.3062 + 2.3600i
1.3588 + 2.3687i
1.4131 + 2.3774i
1.4691 + 2.3860i
1.5272 + 2.3944i
1.5870 + 2.4032i
1.6484 + 2.4123i
1.7115 + 2.4218i
1.7768 + 2.4313i
1.8443 + 2.4407i
1.9143 + 2.4497i
1.9871 + 2.4582i
2.0629 + 2.4659i
2.1419 + 2.4726i
2.2243 + 2.4782i
2.3101 + 2.4840i
2.3991 + 2.4911i
2.4918 + 2.4987i
2.5887 + 2.5060i
2.6900 + 2.5120i
2.7962 + 2.5161i
2.9077 + 2.5174i
3.0248 + 2.5150i
3.1481 + 2.5082i
3.2778 + 2.4961i
3.4155 + 2.4848i
3.5624 + 2.4786i
3.7185 + 2.4736i
3.8836 + 2.4662i
4.0576 + 2.4524i
4.2405 + 2.4287i
4.4322 + 2.3911i
4.6326 + 2.3359i
4.8415 + 2.2595i
5.0590 + 2.1579i
5.3116 + 2.0531i
5.6159 + 1.9604i
5.9571 + 1.8657i
6.3204 + 1.7550i
6.6908 + 1.6143i
7.0535 + 1.4295i
7.3937 + 1.1868i
7.6964 + 0.8720i
```

```
   7.9468 + 0.4711i
   8.1299 - 0.0298i
   8.3110 - 0.6357i
   8.5403 - 1.3306i
   8.7814 - 2.0977i
   8.9975 - 2.9196i
   9.1519 - 3.7795i
   9.2080 - 4.6601i
   9.1291 - 5.5445i
   8.8786 - 6.4155i
   8.4198 - 7.2560i
   7.7160 - 8.0490i
   6.8506 - 8.6946i
   5.9420 - 9.1242i
   5.0061 - 9.3672i
   4.0588 - 9.4532i
   3.1158 - 9.4116i
   2.1931 - 9.2719i
   1.3066 - 9.0637i
   0.4720 - 8.8165i
  -0.2947 - 8.5596i
  -0.9777 - 8.3228i
  -1.5383 - 8.0622i
  -1.9620 - 7.7264i
  -2.2692 - 7.3328i
  -2.4800 - 6.8992i
  -2.6148 - 6.4430i
  -2.6939 - 5.9818i
  -2.7376 - 5.5332i
  -2.7663 - 5.1147i
  -2.8001 - 4.7441i
  -2.8594 - 4.4387i
  -2.9211 - 4.1801i
  -2.9519 - 3.9375i
  -2.9569 - 3.7102i
  -2.9413 - 3.4973i
  -2.9102 - 3.2982i
  -2.8689 - 3.1120i
  -2.8225 - 2.9379i
  -2.7761 - 2.7753i
  -2.7349 - 2.6234i
  -2.7041 - 2.4813i
  -2.6776 - 2.3487i
  -2.6464 - 2.2251i
```

```
-2.6116 - 2.1099i
-2.5741 - 2.0022i
-2.5348 - 1.9015i
-2.4946 - 1.8069i
-2.4544 - 1.7178i
-2.4154 - 1.6335i
-2.3782 - 1.5531i
-2.3440 - 1.4761i
-2.3111 - 1.4026i
-2.2778 - 1.3333i
-2.2442 - 1.2679i
-2.2106 - 1.2060i
-2.1771 - 1.1474i
-2.1442 - 1.0918i
-2.1119 - 1.0388i
-2.0805 - 0.9882i
-2.0504 - 0.9396i
-2.0216 - 0.8929i
-1.9938 - 0.8481i
-1.9662 - 0.8054i
-1.9391 - 0.7647i
-1.9124 - 0.7258i
-1.8862 - 0.6887i
-1.8605 - 0.6532i
-1.8353 - 0.6190i
-1.8108 - 0.5861i
-1.7870 - 0.5543i
-1.7640 - 0.5235i
-1.7415 - 0.4938i
-1.7195 - 0.4652i
-1.6978 - 0.4378i
-1.6766 - 0.4114i
-1.6558 - 0.3860i
-1.6353 - 0.3615i
-1.6152 - 0.3377i
-1.5954 - 0.3147i
-1.5759 - 0.2924i
-1.5567 - 0.2706i
-1.5377 - 0.2493i
-1.5189 - 0.2286i
-1.5003 - 0.2086i
-1.4819 - 0.1892i
-1.4638 - 0.1704i
-1.4459 - 0.1523i
```

```
-1.4283 - 0.1349i
-1.4110 - 0.1182i
-1.3940 - 0.1022i
-1.3773 - 0.0869i
```

Input Arguments

`abcdflag` — ABCD-parameter index

`'A'` | `'B'` | `'C'` | `'D'`

Flag that determines which ABCD parameters the function extracts, specified as `'A'`, `'B'`, `'C'`, or `'D'`.

`habcd` — 2-port ABCD parameters

ABCD parameter object

2-port ABCD parameters, specified as an RF Toolbox™ ABCD parameter object. When you specify `abcdflag`, you must also specify an ABCD parameter object.

`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)`

`abcd_vector` — *A*, *B*, *C*, or *D*- parameters
vector

A, *B*, *C*, or *D*- parameters, returned as a vector. The `abcdflag` input argument determines which parameters the function returns. The function supports only 2-port ABCD parameters; thus, the output is always a vector.

Example: `a_vector = rfparam(habcd,'A');`

See Also

`rfinterp1` | `rfplot` | `rfplot` | `sparameters` | `sparameters`

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

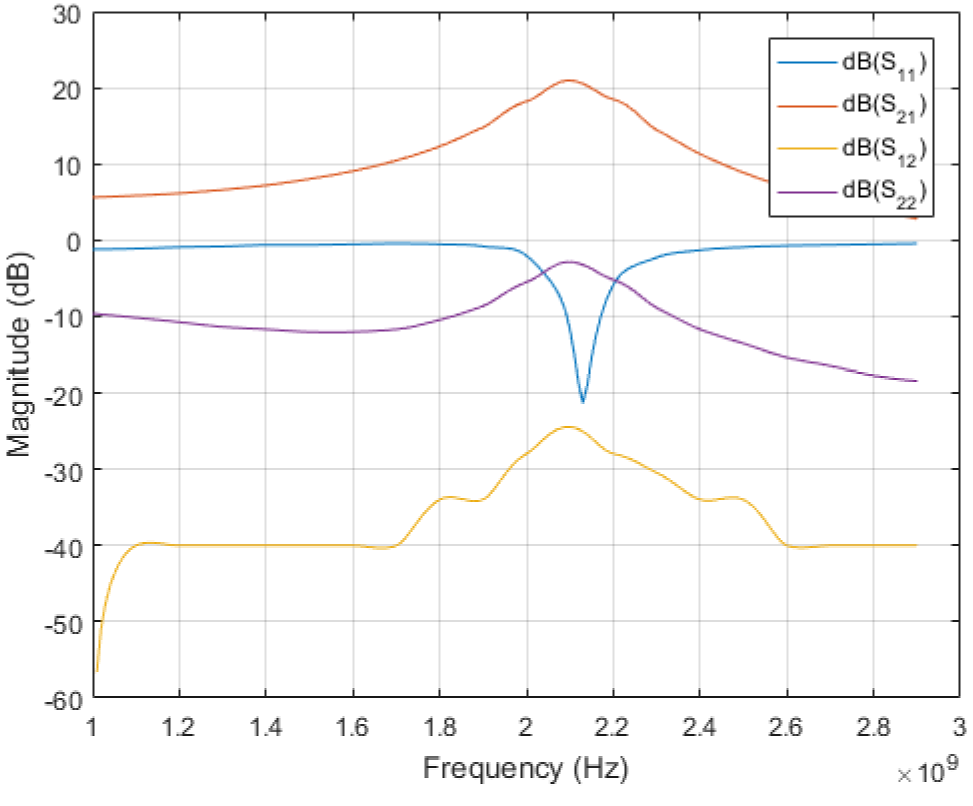
Plot S-Parameter Data Using rfplot

Create S-parameter

```
hs = sparameters('default.s2p');
```

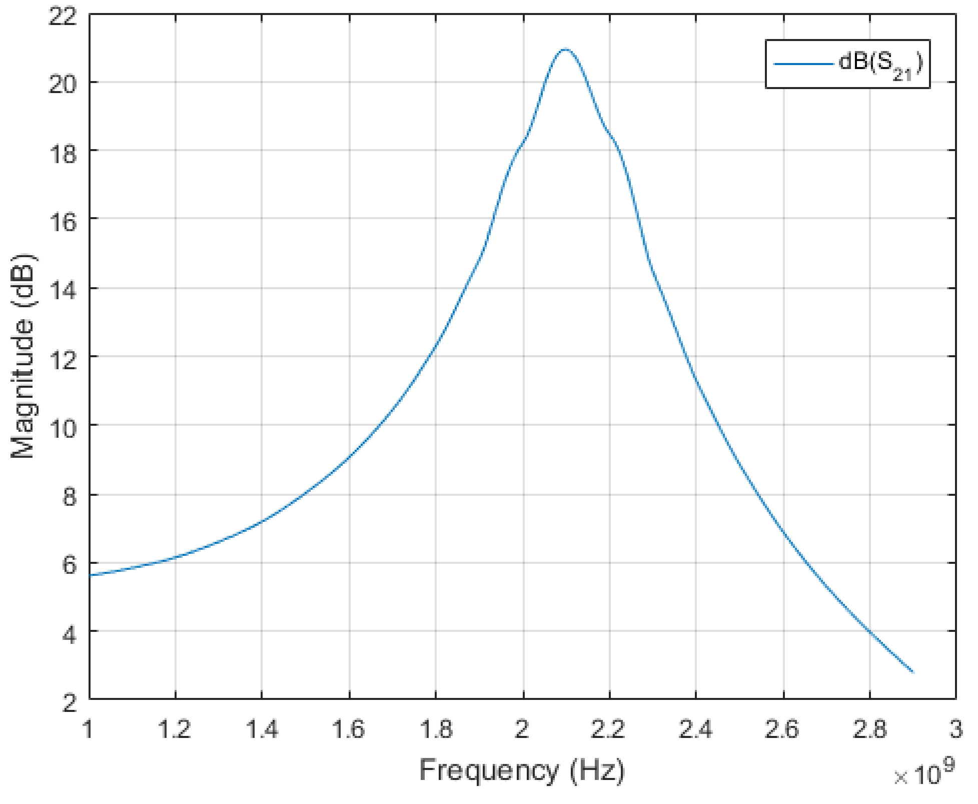
Plot all S-paramteres

```
figure;
rfplot(hs)
```



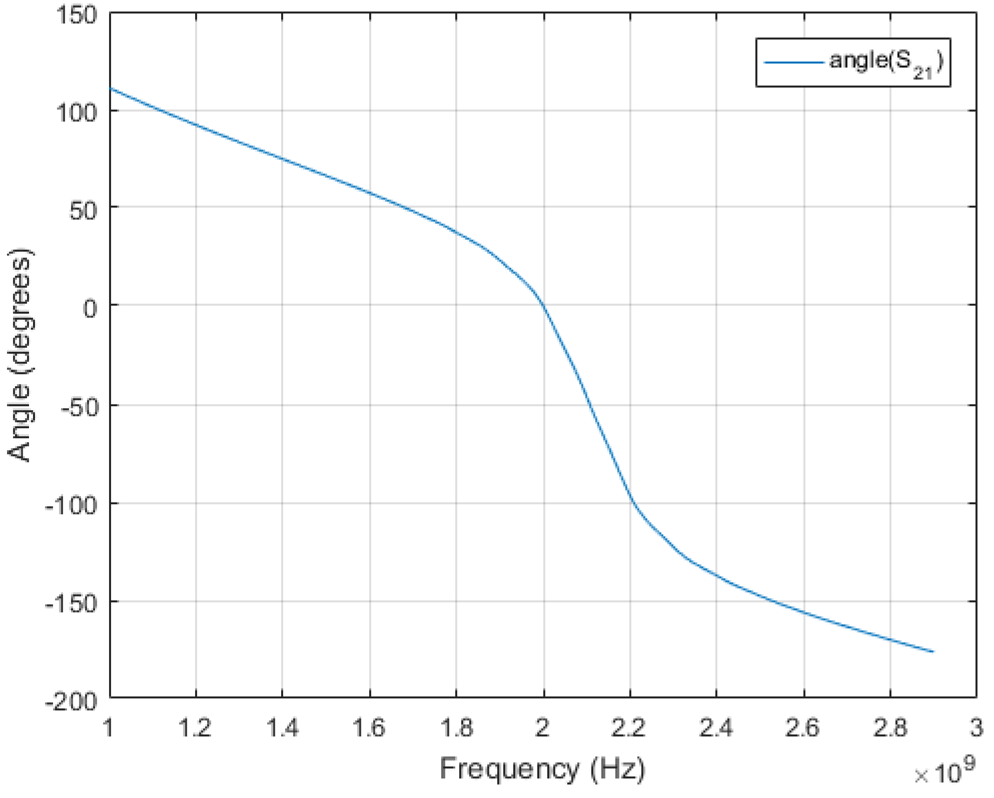
Plot S21

```
figure;
rfplot(hs,2,1)
```



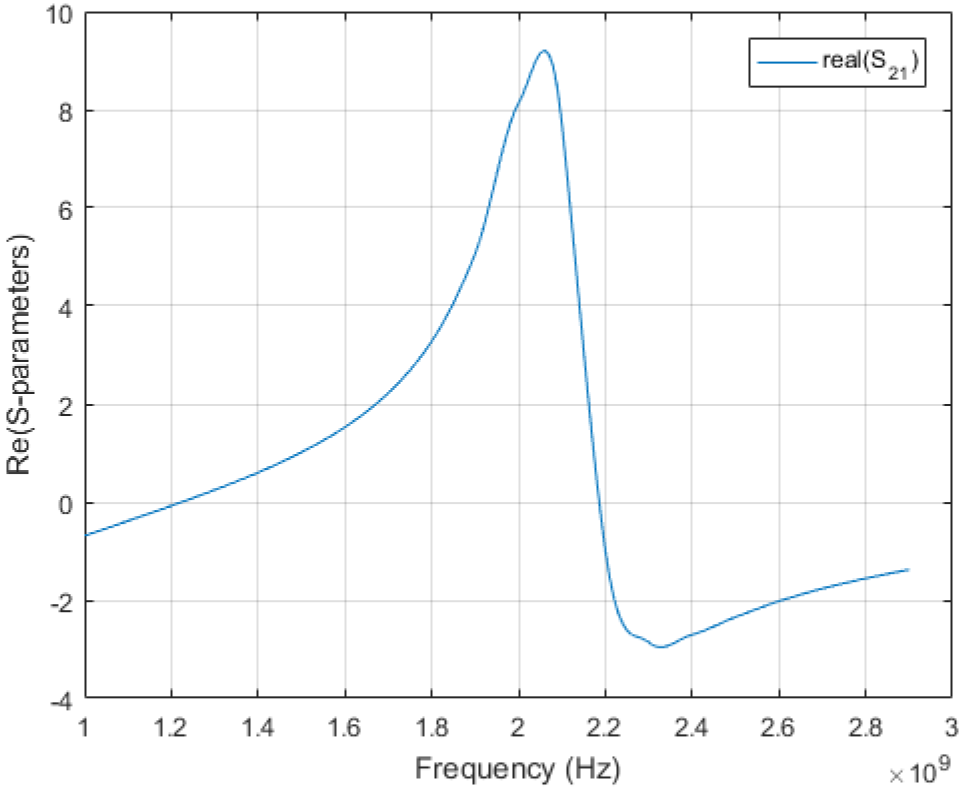
Plot the angle of S21 in degrees

```
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 string

Line specification, specified as a character string, that modifies the line types, symbols, and colors of the plot. The function takes string specifiers in the same format as `plot` command. For more information on line specification strings, see `linespec`.

Example: `'-or'`

plotflag — Plot types
`'db'` (default) | character string

Plot types, specified as a character string. The valid plot flags are `'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`

show

Display antenna or array structure

Syntax

```
show(object)
```

Description

`show(object)` displays the structure of an antenna or array object.

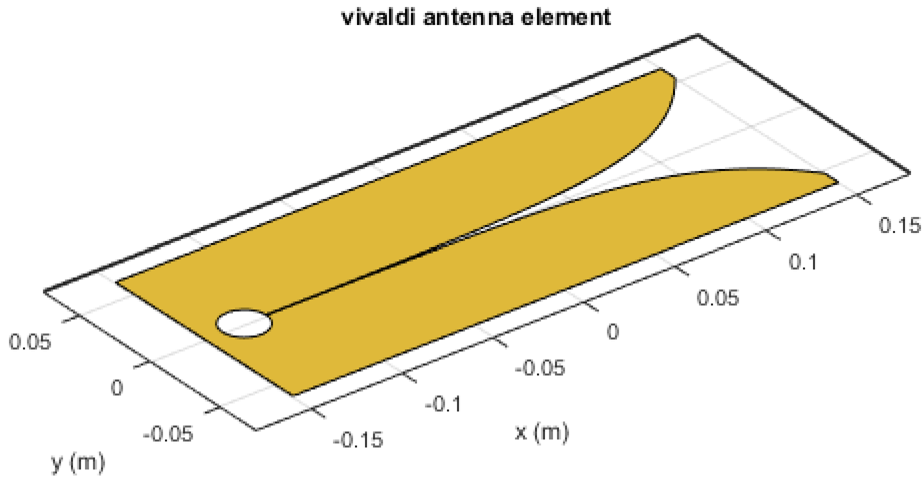
Examples

Display Antenna Structure

This example shows how to create a vivaldi antenna and display the antenna structure.

```
h = vivaldi
show(h)
```

```
h = 

  vivaldi with properties:

              TaperLength: 0.2430
            ApertureWidth: 0.1050
              OpeningRate: 0.2500
            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]
```



Input Arguments

`object` — Antenna or array object
scalar handle

Antenna or array object, specified as a scalar handle.

See Also

`layout` | `mesh`

Introduced in R2015a

returnLoss

Return loss of antenna; scan return loss of array

Syntax

```
returnLoss(antenna,frequency,z0)
rl = returnLoss(antenna ,frequency, z0)

retunrLoss(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.

`retunrLoss(array,frequency,elementnumber)` calculates and plots the scan return loss of a specified antenna element in an array.

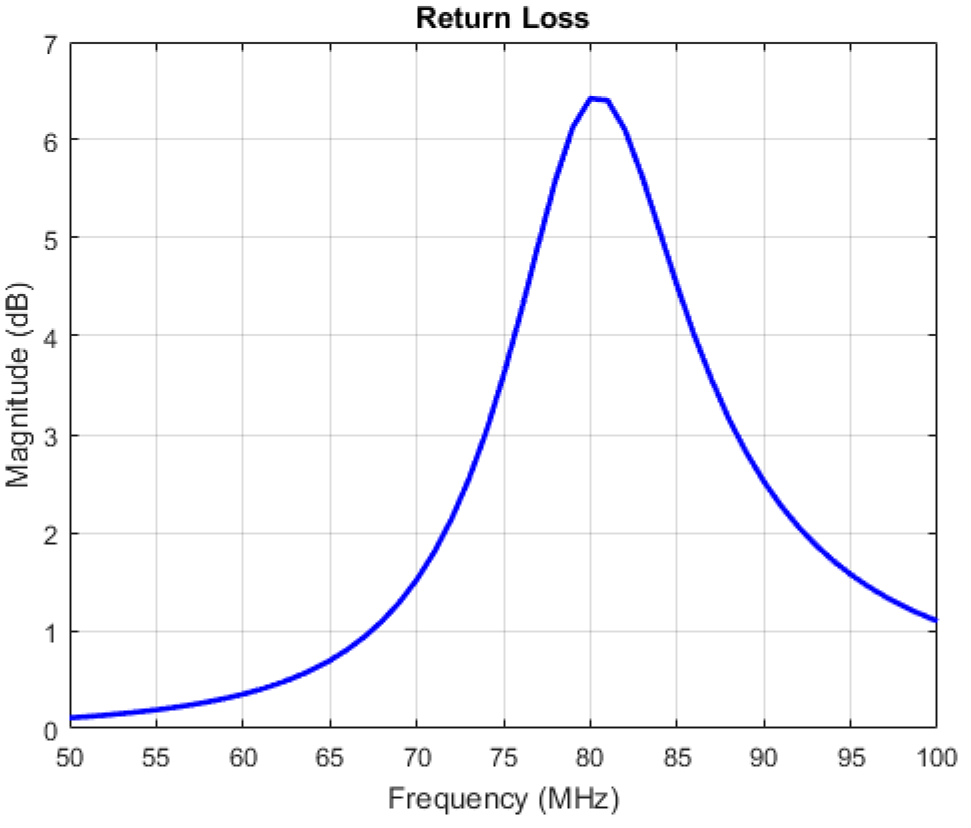
`rl = returnLoss(array,frequency,elementnumber)` returns the scan return loss of a specified antenna element in an array.

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

Syntax

```
pattern(object,frequency)
pattern(object,frequency,azimuth,elevation)
pattern(___,Name,Value)

[fieldval,azimuth,elevation] = pattern(object,frequency)
[fieldval,azimuth,elevation] = pattern(object,frequency,azimuth,
elevation)
[fieldval,azimuth,elevation] = pattern(___,Name,Value)
```

Description

`pattern(object,frequency)` plots the 3-D radiation pattern of an antenna or array object over a specified frequency.

`pattern(object,frequency,azimuth,elevation)` plots the radiation pattern of an 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.

`[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.

`[fieldval,azimuth,elevation] = pattern(object,frequency,azimuth,elevation)` returns the fields value of an antenna or array object at specified frequency. `azimuth` and `elevation` are the angles at which the pattern function calculates the directivity.

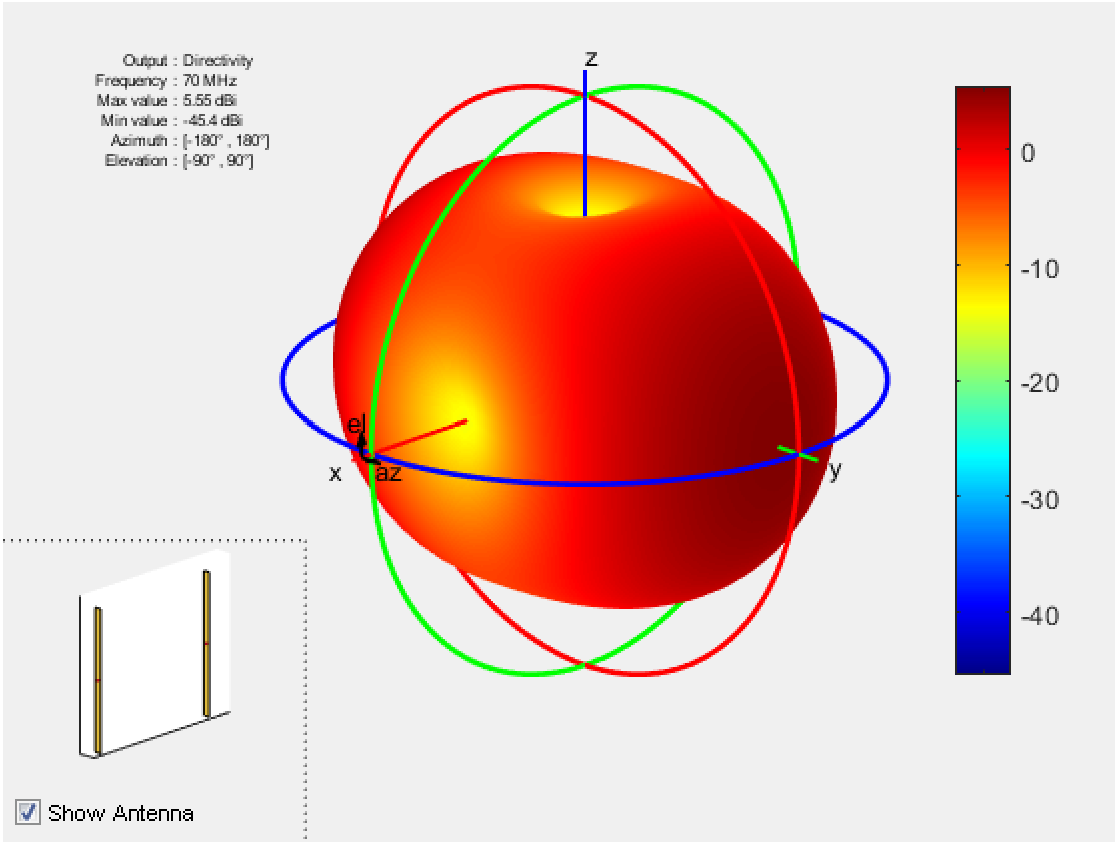
`[fieldval,azimuth,elevation] = pattern(___,Name,Value)`uses additional options specified by one or more `Name,Value` pair arguments.

Examples

Calculate Radiation Pattern of Array

Calculate radiation pattern of default linear array for a frequency of 70 MHZ.

```
l = linearArray;
pattern(l,70e6)
```



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 to calculate charge distribution, specified as a scalar in Hz.

Example: 70e6

Data Types: double

azimuth — Azimuth angle of antenna
–180:5:180 (default) | scalar in degrees | vector in degrees

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

Example: 90

Data Types: double

elevation — Elevation angle of antenna
–90:5:90 (default) | scalar in degrees | vector in degrees

Elevation angle of the antenna, specified as a scalar or 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', 'uv'

'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 strings: 'polar', 'rectangular', 'uv'.

Example: 'CoordinateSystem', 'polar'

Data Types: char

'Type' — Value to plot
'directivity' (default) | 'gain' | 'efield' | 'power' | 'powerdb' | string

Value to plot, specified as a comma-separated pair consisting of 'Type' and one of these strings:

- `'directivity'` – Radiation intensity in a given direction of antenna
- `'gain'` – Radiation intensity in a given direction of antenna, when the antenna has a lossy substrate
- `'efield'` – Electric field of antenna
- `'power'` – Antenna power in watts
- `'powerdb'` – Antenna power in dB

Example: `'Type'`, `'efield'`

Data Types: `char`

`'Normalize'` — Normalize filed 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`

`'PlotStyle'` — 2-D pattern display style

`'overlay'` (default) | `'waterfall'`

2-D pattern display style, specified as the comma-separated pair consisting of `'PlotStyle'` and one of these strings:

- `'overlay'` – Overlay frequency data in a 2-D line plot
- `'waterfall'` – Plot frequency data in a waterfall plot

This property applies only when you call the function with no output arguments.

Example: `'PlotStyle'`, `'waterfall'`

Data Types: `char`

`'Polarization'` — Field polarization

`'H'` | `'V'` | `'RHCP'` | `'LHCP'` | string

Field polarization, specified as the comma-separated pair consisting of `'Polarization'` and one of these strings:

- `'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.

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 required antenna.

Example: `'Termination'`,40

Data Types: `double`

Output Arguments

`fieldval` — Field value of antenna
matrix

Field value of the antenna, returned as a matrix of one of the following values:

- `directivity` – Radiation intensity in a given direction of antenna

- `gain` – Radiation intensity in a given direction of antenna, when the antenna has a lossy substrate
- `efield` – Electric field of antenna
- `power` – Antenna power in watts
- `powerdb` – Antenna power in dB

Matrix size is number of elevation values multiplied by number of azimuth values.

`azimuth` — Azimuth angles over which directivity is calculated

vector in degrees

Azimuth angles over which directivity is calculated, returned as a vector in degrees.

`elevation` — Elevation angles over which directivity is calculated

vector in degrees

Elevation angles over which directivity is calculated, returned as a vector in degrees.

See Also

`current` | `EHfields`

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,Name,Value)
```

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.

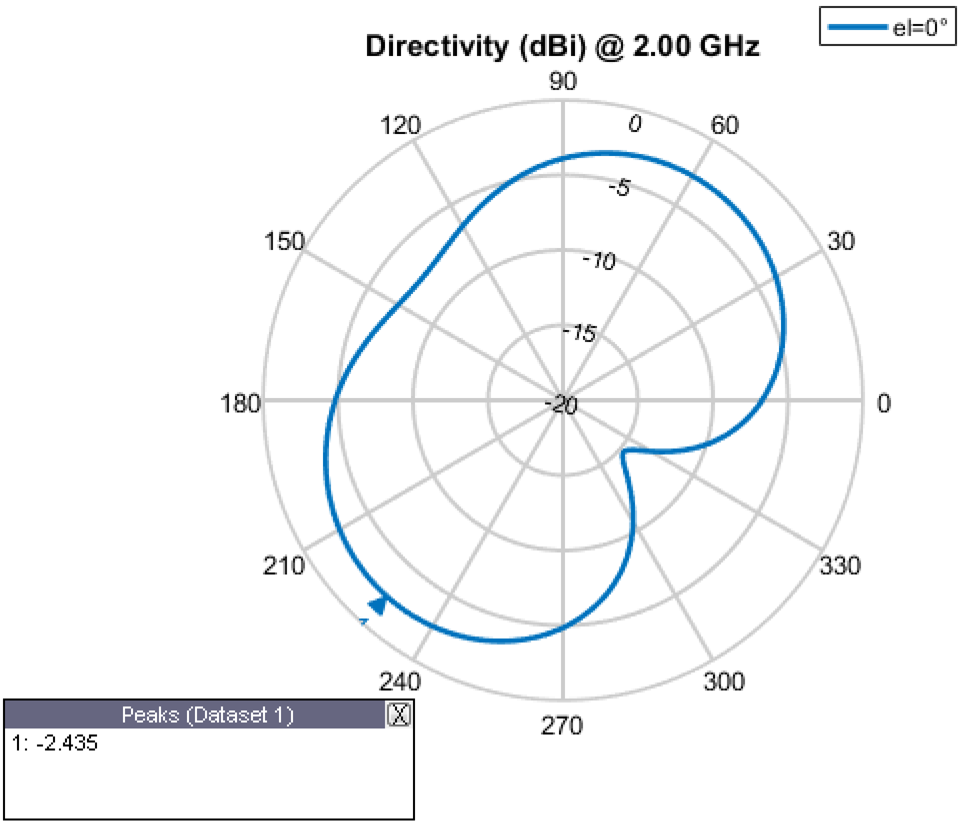
`directivity = patternAzimuth(object,frequency,elevation,Name,Value)` uses additional options specified by one or more `Name,Value` pair arguments.

Examples

Azimuth Radiation Pattern of Helix Antenna

Calculate and plot the azimuth radiation pattern of the helix antenna at 2 GHz.

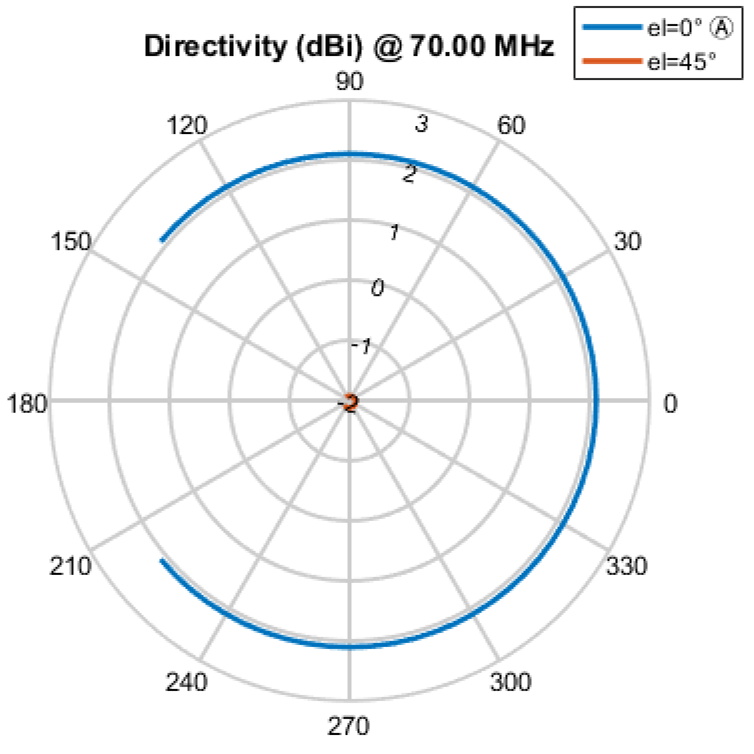
```
h = helix;
patternAzimuth(h,2e9);
```



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`

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: `'Azimuth'`,2:2:340

`'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 id the product of number of elevation values and number of azimuth values.

See Also

`pattern` | `patternElevation`

Introduced in R2015a

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,Name,Value)
```

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.

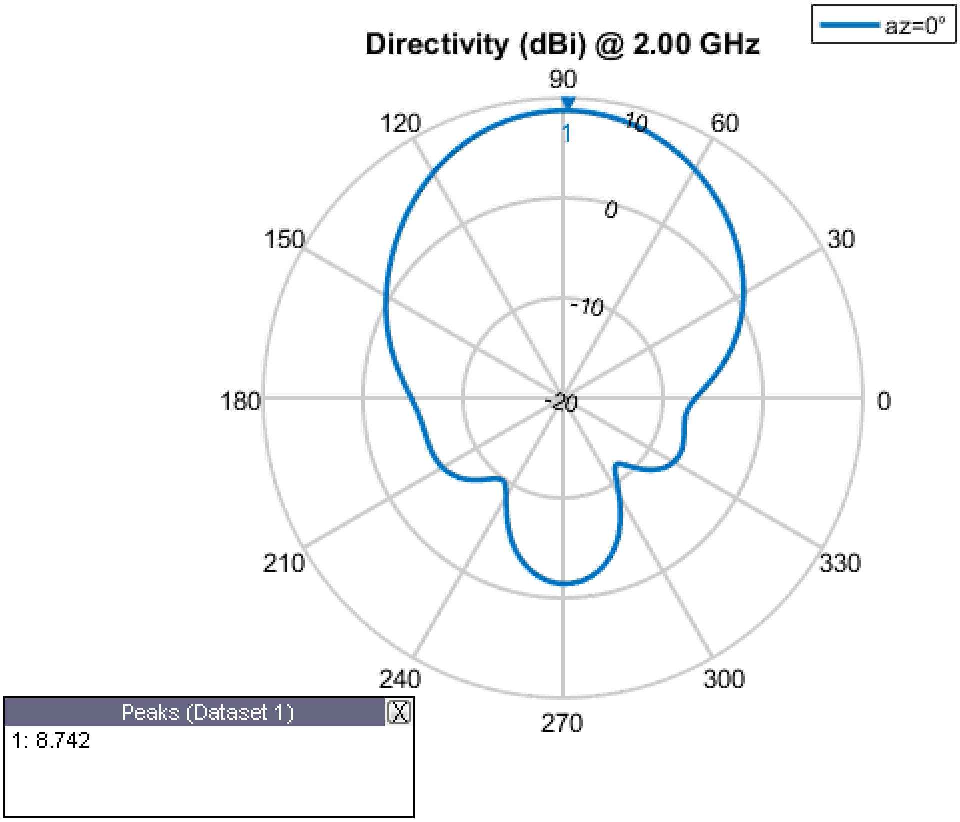
`directivity = patternElevation(object,frequency,azimuth,Name,Value)` uses additional options specified by one or more `Name, Value` pair arguments.

Examples

Elevation Radiation Pattern of Helix

Calculate and plot the elevation pattern of the helix antenna at 2 GHz.

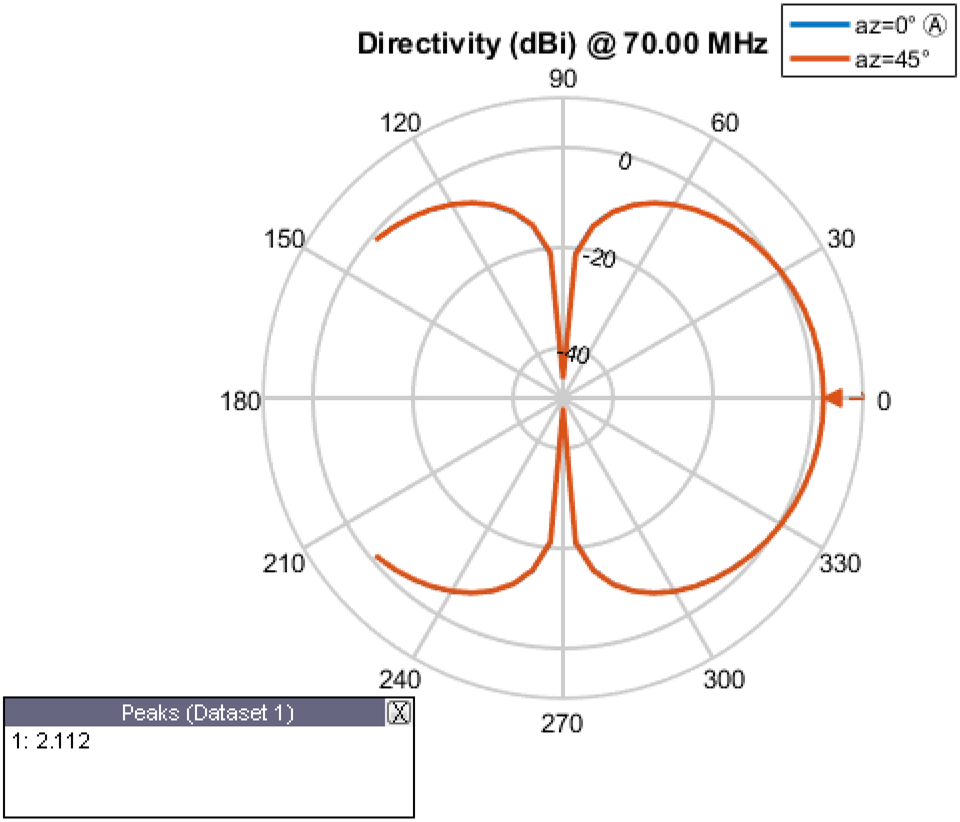
```
h = helix;
patternElevation (h, 2e9);
```



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

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: 'Elevation', 0:1:360

'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 id the product of number of elevation values and number of azimuth values.

See Also

pattern | patternAzimuth

Introduced in R2015a

current

Current distribution on antenna or array surface

Syntax

`current(object,frequency)`

`i = current(object,frequency)`

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.

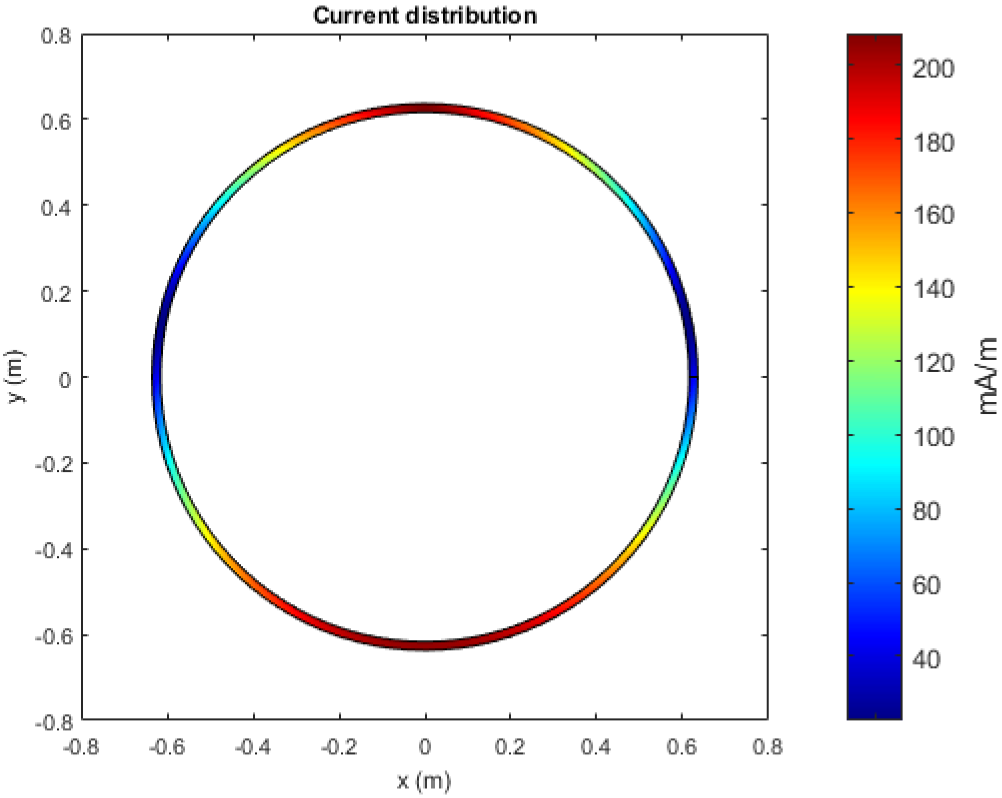
`i = current(object,frequency)` returns the x, y, z components of the current on the surface of an antenna or array object, at a specified frequency.

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 defualt rectangular array at 70MHz frequency.

```
h = rectangularArray;
i = current(h,70e6)
```

```
i =

  Columns 1 through 4

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0039 + 0.0064i  -0.0017 - 0.0026i   0.0019 + 0.0033i  -0.0017 - 0.0028i
   0.0041 + 0.0067i   0.0160 + 0.0258i   0.0198 + 0.0320i   0.0274 + 0.0448i
```

```
  Columns 5 through 8

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0017 + 0.0030i  -0.0015 - 0.0024i   0.0015 + 0.0029i  -0.0013 - 0.0022i
   0.0310 + 0.0509i   0.0377 + 0.0625i   0.0409 + 0.0681i   0.0468 + 0.0787i

  Columns 9 through 12

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0014 + 0.0027i  -0.0010 - 0.0019i   0.0012 + 0.0025i  -0.0008 - 0.0016i
   0.0496 + 0.0838i   0.0546 + 0.0934i   0.0570 + 0.0980i   0.0611 + 0.1066i

  Columns 13 through 16

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0010 + 0.0022i  -0.0005 - 0.0013i   0.0007 + 0.0020i  -0.0003 - 0.0009i
   0.0629 + 0.1106i   0.0661 + 0.1180i   0.0674 + 0.1215i   0.0696 + 0.1277i

  Columns 17 through 20

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0004 + 0.0018i  -0.0001 - 0.0010i   0.0001 + 0.0013i  -0.0001 - 0.0066i
   0.0703 + 0.1306i   0.0716 + 0.1364i   0.0718 + 0.1381i   0.0719 + 0.1465i

  Columns 21 through 24

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0001 - 0.0067i   0.0001 + 0.0013i  -0.0002 - 0.0011i   0.0003 + 0.0015i
   0.0719 + 0.1465i   0.0718 + 0.1381i   0.0715 + 0.1363i   0.0705 + 0.1308i

  Columns 25 through 28

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0005 - 0.0013i   0.0006 + 0.0016i  -0.0007 - 0.0017i   0.0008 + 0.0019i
   0.0696 + 0.1278i   0.0675 + 0.1215i   0.0662 + 0.1181i   0.0630 + 0.1107i

  Columns 29 through 32

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0009 - 0.0020i   0.0011 + 0.0021i  -0.0011 - 0.0022i   0.0013 + 0.0024i
   0.0611 + 0.1066i   0.0570 + 0.0980i   0.0547 + 0.0935i   0.0496 + 0.0838i

  Columns 33 through 36
```

```
  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
 -0.0013 - 0.0025i   0.0015 + 0.0026i  -0.0015 - 0.0027i   0.0017 + 0.0027i
  0.0469 + 0.0787i   0.0409 + 0.0680i   0.0378 + 0.0624i   0.0311 + 0.0509i

 Columns 37 through 40

  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
 -0.0018 - 0.0031i   0.0018 + 0.0030i  -0.0017 - 0.0029i   0.0040 + 0.0063i
  0.0274 + 0.0447i   0.0198 + 0.0320i   0.0161 + 0.0259i   0.0042 + 0.0066i

 Columns 41 through 44

  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  0.0040 + 0.0064i  -0.0016 - 0.0027i   0.0020 + 0.0032i  -0.0016 - 0.0028i
  0.0042 + 0.0067i   0.0160 + 0.0258i   0.0198 + 0.0320i   0.0275 + 0.0448i

 Columns 45 through 48

  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  0.0018 + 0.0030i  -0.0014 - 0.0025i   0.0017 + 0.0029i  -0.0011 - 0.0022i
  0.0311 + 0.0509i   0.0378 + 0.0624i   0.0409 + 0.0681i   0.0468 + 0.0787i

 Columns 49 through 52

  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  0.0015 + 0.0027i  -0.0009 - 0.0019i   0.0013 + 0.0025i  -0.0007 - 0.0016i
  0.0496 + 0.0838i   0.0547 + 0.0934i   0.0570 + 0.0980i   0.0611 + 0.1066i

 Columns 53 through 56

  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  0.0010 + 0.0022i  -0.0005 - 0.0013i   0.0008 + 0.0020i  -0.0002 - 0.0009i
  0.0629 + 0.1106i   0.0661 + 0.1180i   0.0674 + 0.1214i   0.0696 + 0.1277i

 Columns 57 through 60

  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  0.0005 + 0.0018i  -0.0001 - 0.0010i   0.0001 + 0.0013i  -0.0000 - 0.0066i
  0.0703 + 0.1306i   0.0716 + 0.1364i   0.0718 + 0.1381i   0.0719 + 0.1465i

 Columns 61 through 64

  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
```

```
  -0.0001 - 0.0067i   0.0001 + 0.0013i  -0.0002 - 0.0011i   0.0003 + 0.0015i
   0.0719 + 0.1465i   0.0717 + 0.1381i   0.0715 + 0.1363i   0.0705 + 0.1308i

  Columns 65 through 68

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0005 - 0.0013i   0.0005 + 0.0016i  -0.0007 - 0.0016i   0.0008 + 0.0019i
   0.0696 + 0.1278i   0.0675 + 0.1215i   0.0662 + 0.1181i   0.0630 + 0.1107i

  Columns 69 through 72

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0010 - 0.0020i   0.0010 + 0.0021i  -0.0012 - 0.0022i   0.0012 + 0.0024i
   0.0611 + 0.1066i   0.0570 + 0.0980i   0.0547 + 0.0935i   0.0496 + 0.0838i

  Columns 73 through 76

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0014 - 0.0025i   0.0014 + 0.0026i  -0.0016 - 0.0027i   0.0015 + 0.0028i
   0.0468 + 0.0787i   0.0409 + 0.0681i   0.0377 + 0.0625i   0.0310 + 0.0509i

  Columns 77 through 80

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0019 - 0.0031i   0.0017 + 0.0030i  -0.0018 - 0.0029i   0.0039 + 0.0063i
   0.0274 + 0.0447i   0.0198 + 0.0320i   0.0161 + 0.0259i   0.0041 + 0.0066i

  Columns 81 through 84

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0039 + 0.0064i  -0.0017 - 0.0026i   0.0019 + 0.0033i  -0.0017 - 0.0028i
   0.0041 + 0.0067i   0.0160 + 0.0258i   0.0198 + 0.0320i   0.0274 + 0.0448i

  Columns 85 through 88

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0017 + 0.0030i  -0.0015 - 0.0024i   0.0015 + 0.0029i  -0.0013 - 0.0022i
   0.0310 + 0.0509i   0.0377 + 0.0625i   0.0409 + 0.0681i   0.0468 + 0.0787i

  Columns 89 through 92

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0014 + 0.0027i  -0.0010 - 0.0019i   0.0012 + 0.0025i  -0.0008 - 0.0016i
   0.0496 + 0.0838i   0.0546 + 0.0934i   0.0570 + 0.0980i   0.0611 + 0.1066i
```

```
  Columns 93 through 96

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0010 + 0.0022i  -0.0005 - 0.0013i   0.0007 + 0.0020i  -0.0003 - 0.0009i
   0.0629 + 0.1106i   0.0661 + 0.1180i   0.0674 + 0.1215i   0.0696 + 0.1277i

  Columns 97 through 100

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0004 + 0.0018i  -0.0001 - 0.0010i   0.0001 + 0.0013i  -0.0001 - 0.0066i
   0.0703 + 0.1306i   0.0716 + 0.1364i   0.0718 + 0.1381i   0.0719 + 0.1465i

  Columns 101 through 104

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0001 - 0.0067i   0.0001 + 0.0013i  -0.0002 - 0.0011i   0.0003 + 0.0015i
   0.0719 + 0.1465i   0.0718 + 0.1381i   0.0715 + 0.1363i   0.0705 + 0.1308i

  Columns 105 through 108

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0005 - 0.0013i   0.0006 + 0.0016i  -0.0007 - 0.0017i   0.0008 + 0.0019i
   0.0696 + 0.1278i   0.0675 + 0.1215i   0.0662 + 0.1181i   0.0630 + 0.1107i

  Columns 109 through 112

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0009 - 0.0020i   0.0011 + 0.0021i  -0.0011 - 0.0022i   0.0013 + 0.0024i
   0.0611 + 0.1066i   0.0570 + 0.0980i   0.0547 + 0.0935i   0.0496 + 0.0838i

  Columns 113 through 116

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0013 - 0.0025i   0.0015 + 0.0026i  -0.0015 - 0.0027i   0.0017 + 0.0027i
   0.0469 + 0.0787i   0.0409 + 0.0680i   0.0378 + 0.0624i   0.0311 + 0.0509i

  Columns 117 through 120

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0018 - 0.0031i   0.0018 + 0.0030i  -0.0017 - 0.0029i   0.0040 + 0.0063i
   0.0274 + 0.0447i   0.0198 + 0.0320i   0.0161 + 0.0259i   0.0042 + 0.0066i

  Columns 121 through 124
```

```
  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  0.0040 + 0.0064i  -0.0016 - 0.0027i   0.0020 + 0.0032i  -0.0016 - 0.0028i
  0.0042 + 0.0067i   0.0160 + 0.0258i   0.0198 + 0.0320i   0.0275 + 0.0448i

 Columns 125 through 128

  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  0.0018 + 0.0030i  -0.0014 - 0.0025i   0.0017 + 0.0029i  -0.0011 - 0.0022i
  0.0311 + 0.0509i   0.0378 + 0.0624i   0.0409 + 0.0681i   0.0468 + 0.0787i

 Columns 129 through 132

  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  0.0015 + 0.0027i  -0.0009 - 0.0019i   0.0013 + 0.0025i  -0.0007 - 0.0016i
  0.0496 + 0.0838i   0.0547 + 0.0934i   0.0570 + 0.0980i   0.0611 + 0.1066i

 Columns 133 through 136

  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  0.0010 + 0.0022i  -0.0005 - 0.0013i   0.0008 + 0.0020i  -0.0002 - 0.0009i
  0.0629 + 0.1106i   0.0661 + 0.1180i   0.0674 + 0.1214i   0.0696 + 0.1277i

 Columns 137 through 140

  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  0.0005 + 0.0018i  -0.0001 - 0.0010i   0.0001 + 0.0013i  -0.0000 - 0.0066i
  0.0703 + 0.1306i   0.0716 + 0.1364i   0.0718 + 0.1381i   0.0719 + 0.1465i

 Columns 141 through 144

  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
 -0.0001 - 0.0067i   0.0001 + 0.0013i  -0.0002 - 0.0011i   0.0003 + 0.0015i
  0.0719 + 0.1465i   0.0717 + 0.1381i   0.0715 + 0.1363i   0.0705 + 0.1308i

 Columns 145 through 148

  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
 -0.0005 - 0.0013i   0.0005 + 0.0016i  -0.0007 - 0.0016i   0.0008 + 0.0019i
  0.0696 + 0.1278i   0.0675 + 0.1215i   0.0662 + 0.1181i   0.0630 + 0.1107i

 Columns 149 through 152

  0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
```

```
  -0.0010 - 0.0020i   0.0010 + 0.0021i  -0.0012 - 0.0022i   0.0012 + 0.0024i
   0.0611 + 0.1066i   0.0570 + 0.0980i   0.0547 + 0.0935i   0.0496 + 0.0838i

  Columns 153 through 156

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0014 - 0.0025i   0.0014 + 0.0026i  -0.0016 - 0.0027i   0.0015 + 0.0028i
   0.0468 + 0.0787i   0.0409 + 0.0681i   0.0377 + 0.0625i   0.0310 + 0.0509i

  Columns 157 through 160

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0019 - 0.0031i   0.0017 + 0.0030i  -0.0018 - 0.0029i   0.0039 + 0.0063i
   0.0274 + 0.0447i   0.0198 + 0.0320i   0.0161 + 0.0259i   0.0041 + 0.0066i
```

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 on the surface of an antenna or array.

See Also
`axialRatio` | `charge`

Introduced in R2015a

charge

Charge distribution on antenna or array surface

Syntax

`charge(object,frequency)`

`c = charge(object,frequency)`

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.

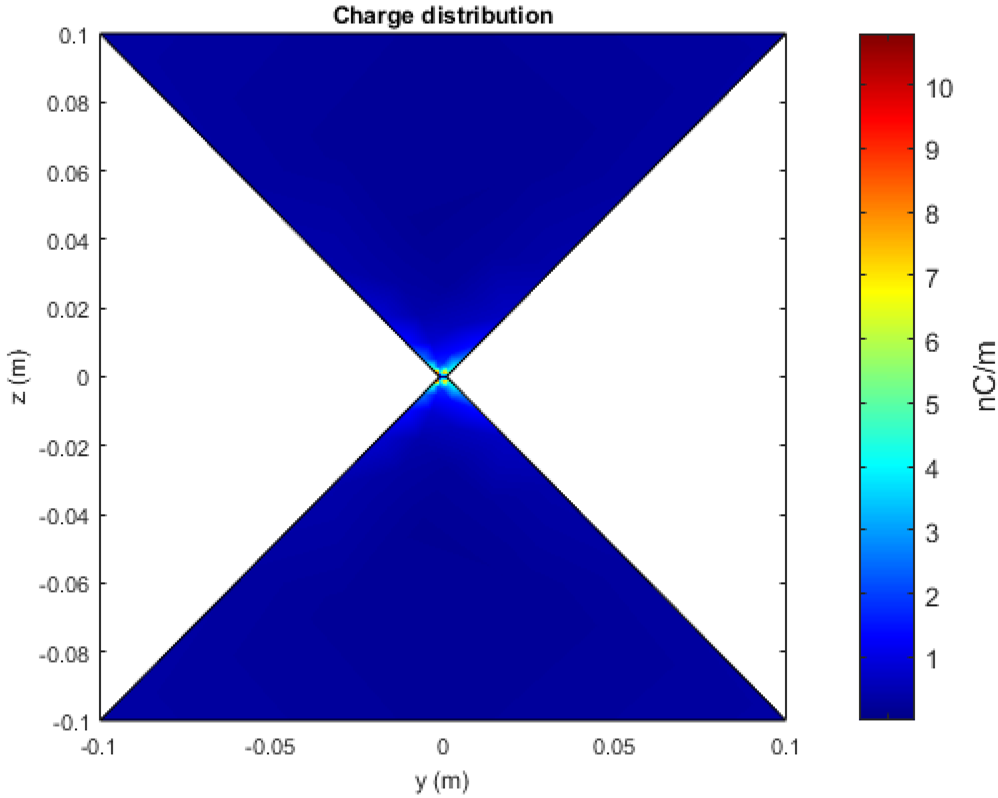
`c = charge(object,frequency)` returns a vector of charges in C/m on the surface of an antenna or array object, at a specified frequency.

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)
```

```
C =

   1.0e-08 *

  Columns 1 through 4
```

```
-0.0159 + 0.1008i  -0.0070 + 0.0414i  -0.0098 + 0.0492i  -0.0083 + 0.0424i

Columns 5 through 8

-0.0103 + 0.0440i  -0.0083 + 0.0359i  -0.0110 + 0.0402i  -0.0085 + 0.0301i

Columns 9 through 12

-0.0117 + 0.0357i  -0.0086 + 0.0245i  -0.0123 + 0.0307i  -0.0086 + 0.0186i

Columns 13 through 16

-0.0130 + 0.0251i  -0.0085 + 0.0126i  -0.0144 + 0.0191i  -0.0082 + 0.0065i

Columns 17 through 20

-0.0167 + 0.0114i  -0.0109 + 0.0034i  -0.0171 + 0.0016i  -0.0898 + 0.0013i

Columns 21 through 24

 0.0905 - 0.0015i   0.0160 - 0.0016i   0.0115 - 0.0048i   0.0156 - 0.0078i

Columns 25 through 28

 0.0100 - 0.0119i   0.0126 - 0.0137i   0.0102 - 0.0177i   0.0115 - 0.0201i

Columns 29 through 32

 0.0101 - 0.0240i   0.0109 - 0.0255i   0.0098 - 0.0293i   0.0105 - 0.0311i

Columns 33 through 36

 0.0095 - 0.0349i   0.0101 - 0.0357i   0.0092 - 0.0400i   0.0095 - 0.0401i

Columns 37 through 40

 0.0090 - 0.0467i   0.0091 - 0.0446i   0.0076 - 0.0448i   0.0155 - 0.0990i

Columns 41 through 44

-0.0492 + 0.1082i  -0.0207 + 0.0445i  -0.0260 + 0.0527i  -0.0222 + 0.0455i

Columns 45 through 48
```

```
-0.0248 + 0.0472i  -0.0201 + 0.0385i  -0.0243 + 0.0431i  -0.0184 + 0.0323i

Columns 49 through 52

-0.0234 + 0.0383i  -0.0166 + 0.0262i  -0.0224 + 0.0328i  -0.0147 + 0.0199i

Columns 53 through 56

-0.0213 + 0.0269i  -0.0126 + 0.0134i  -0.0206 + 0.0204i  -0.0103 + 0.0069i

Columns 57 through 60

-0.0204 + 0.0122i  -0.0120 + 0.0037i  -0.0176 + 0.0017i  -0.0902 + 0.0014i

Columns 61 through 64

 0.0909 - 0.0016i   0.0166 - 0.0017i   0.0131 - 0.0051i   0.0181 - 0.0083i

Columns 65 through 68

 0.0139 - 0.0127i   0.0171 - 0.0146i   0.0160 - 0.0189i   0.0181 - 0.0215i

Columns 69 through 72

 0.0179 - 0.0257i   0.0193 - 0.0273i   0.0194 - 0.0313i   0.0207 - 0.0333i

Columns 73 through 76

 0.0210 - 0.0374i   0.0218 - 0.0382i   0.0224 - 0.0429i   0.0227 - 0.0430i

Columns 77 through 80

 0.0244 - 0.0501i   0.0238 - 0.0479i   0.0224 - 0.0480i   0.0483 - 0.1062i

Columns 81 through 84

-0.0492 + 0.1082i  -0.0207 + 0.0445i  -0.0260 + 0.0527i  -0.0222 + 0.0455i

Columns 85 through 88

-0.0248 + 0.0472i  -0.0201 + 0.0385i  -0.0243 + 0.0431i  -0.0184 + 0.0323i

Columns 89 through 92
```

```
-0.0234 + 0.0383i  -0.0166 + 0.0262i  -0.0224 + 0.0328i  -0.0147 + 0.0199i

Columns 93 through 96

-0.0213 + 0.0269i  -0.0126 + 0.0134i  -0.0206 + 0.0204i  -0.0103 + 0.0069i

Columns 97 through 100

-0.0204 + 0.0122i  -0.0120 + 0.0037i  -0.0176 + 0.0017i  -0.0902 + 0.0014i

Columns 101 through 104

 0.0909 - 0.0016i   0.0166 - 0.0017i   0.0131 - 0.0051i   0.0181 - 0.0083i

Columns 105 through 108

 0.0139 - 0.0127i   0.0171 - 0.0146i   0.0160 - 0.0189i   0.0181 - 0.0215i

Columns 109 through 112

 0.0179 - 0.0257i   0.0193 - 0.0273i   0.0194 - 0.0313i   0.0207 - 0.0333i

Columns 113 through 116

 0.0210 - 0.0374i   0.0218 - 0.0382i   0.0224 - 0.0429i   0.0227 - 0.0430i

Columns 117 through 120

 0.0244 - 0.0501i   0.0238 - 0.0479i   0.0224 - 0.0480i   0.0483 - 0.1062i

Columns 121 through 124

-0.0159 + 0.1008i  -0.0070 + 0.0414i  -0.0098 + 0.0492i  -0.0083 + 0.0424i

Columns 125 through 128

-0.0103 + 0.0440i  -0.0083 + 0.0359i  -0.0110 + 0.0402i  -0.0085 + 0.0301i

Columns 129 through 132

-0.0117 + 0.0357i  -0.0086 + 0.0245i  -0.0123 + 0.0307i  -0.0086 + 0.0186i

Columns 133 through 136
```

```
-0.0130 + 0.0251i  -0.0085 + 0.0126i  -0.0144 + 0.0191i  -0.0082 + 0.0065i

Columns 137 through 140

-0.0167 + 0.0114i  -0.0109 + 0.0034i  -0.0171 + 0.0016i  -0.0898 + 0.0013i

Columns 141 through 144

 0.0905 - 0.0015i   0.0160 - 0.0016i   0.0115 - 0.0048i   0.0156 - 0.0078i

Columns 145 through 148

 0.0100 - 0.0119i   0.0126 - 0.0137i   0.0102 - 0.0177i   0.0115 - 0.0201i

Columns 149 through 152

 0.0101 - 0.0240i   0.0109 - 0.0255i   0.0098 - 0.0293i   0.0105 - 0.0311i

Columns 153 through 156

 0.0095 - 0.0349i   0.0101 - 0.0357i   0.0092 - 0.0400i   0.0095 - 0.0401i

Columns 157 through 160

 0.0090 - 0.0467i   0.0091 - 0.0446i   0.0076 - 0.0448i   0.0155 - 0.0990i
```

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 on the surface of antenna or array.

See Also

`current` | `EHfields`

Introduced in R2015a

createFeed

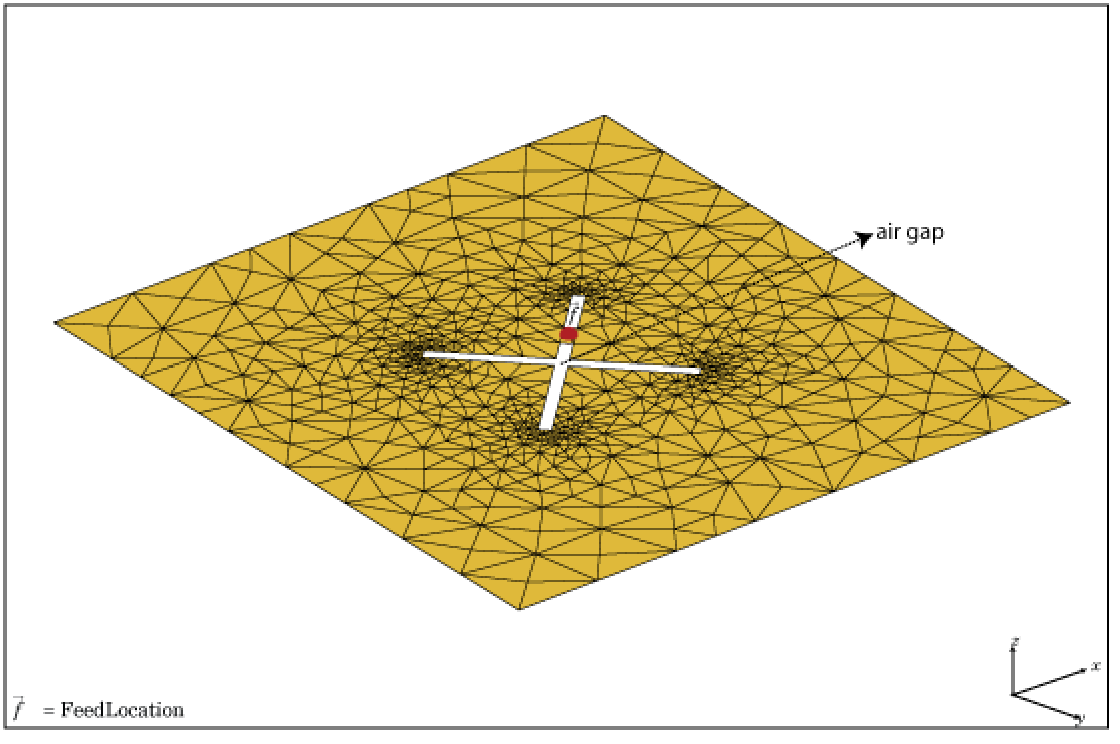
Class: customAntennaMesh

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.

`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 GUI

Load a 2-D custom mesh. Create a custom antenna using the points and triangles.

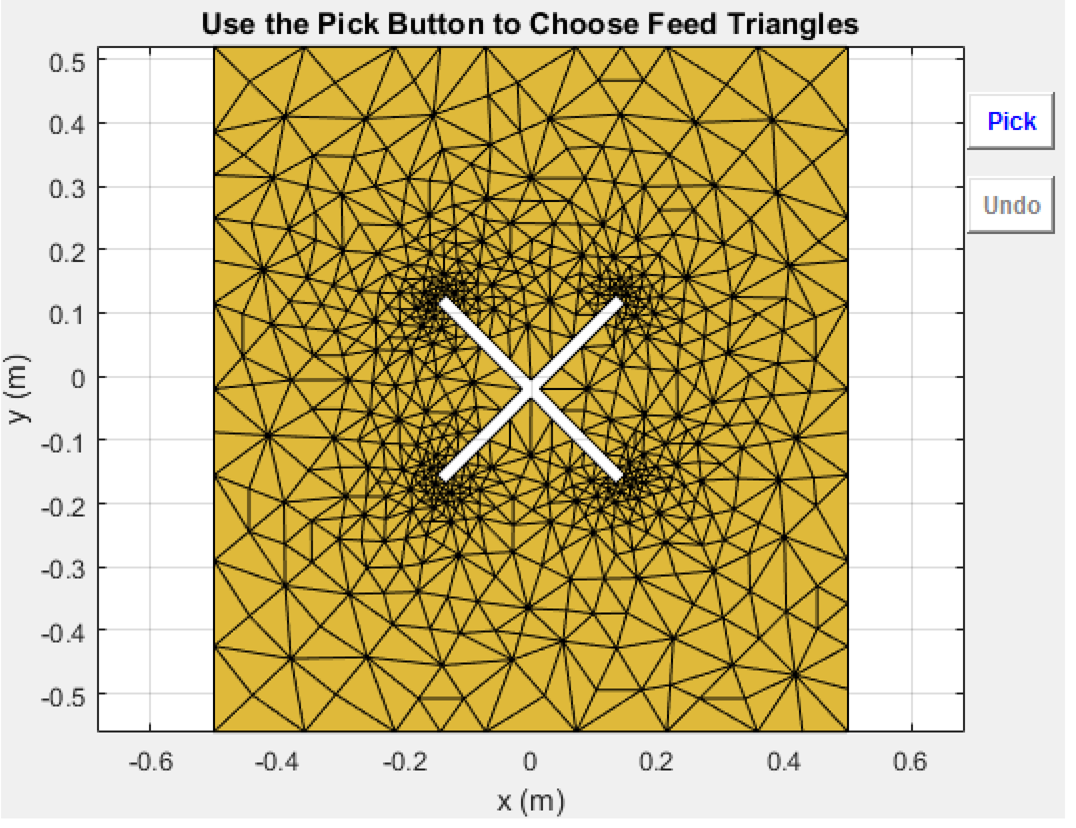
```
load planarmesh.mat
c = customAntennaMesh(p,t)
```

```
c = 

  customAntennaMesh with properties:

          Points: [3x658 double]
```

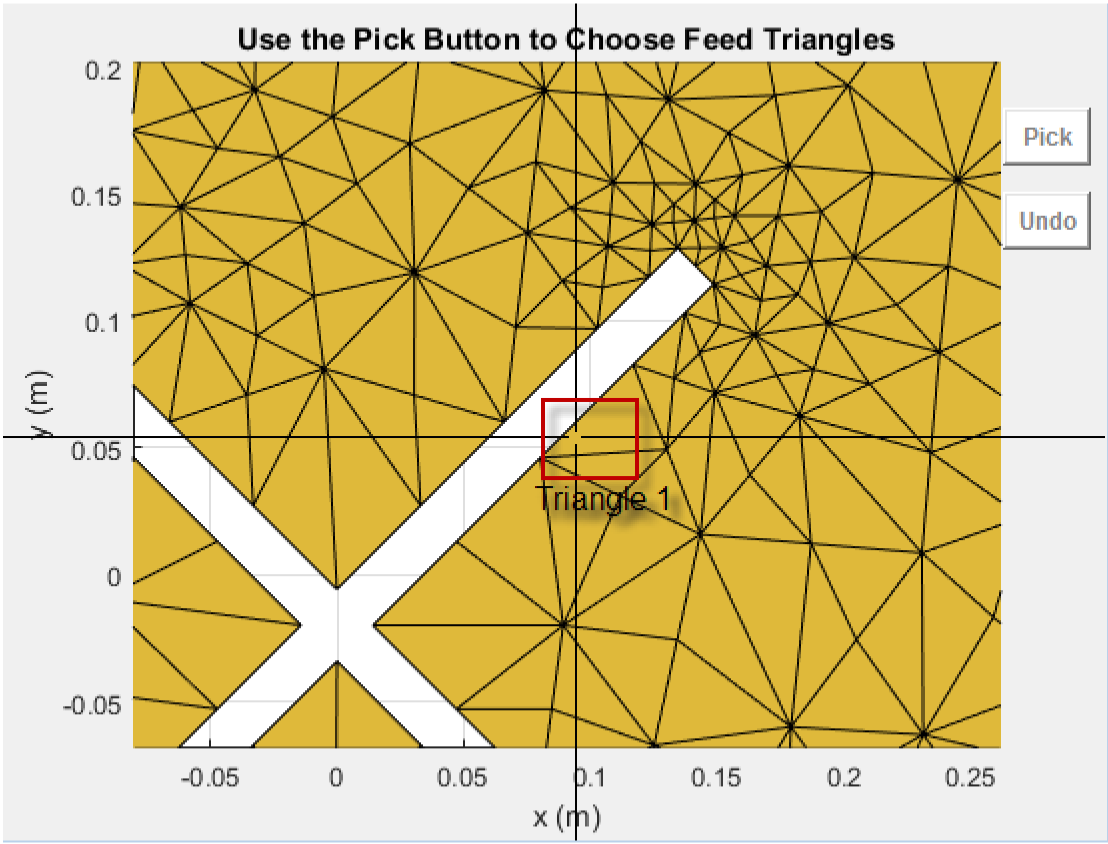
```
     Triangles: [4x1219 double]
  FeedLocation: []
          Tilt: 0
     TiltAxis: [1 0 0]
```

Use the `createFeed` function to view the antenna mesh structure. In this antenna mesh view, you will see **Pick** and **Undo** buttons. The **Pick** button is highlighted.

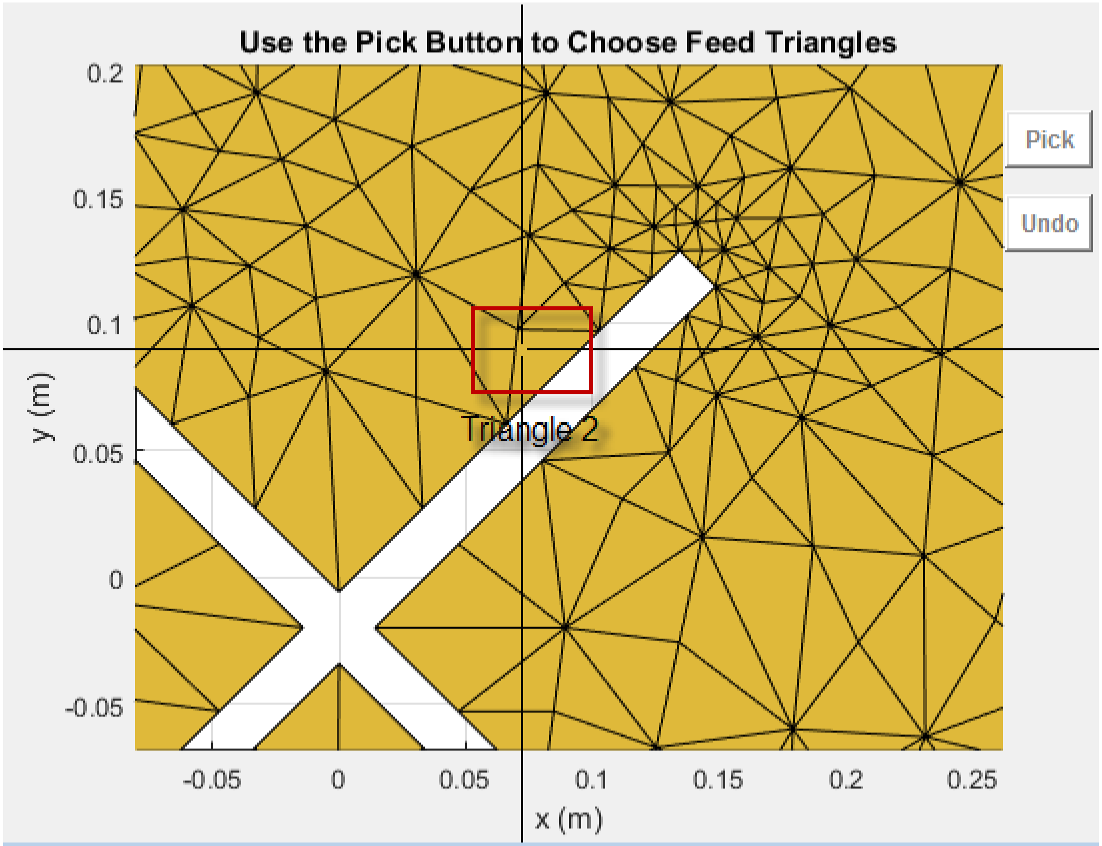
```
createFeed(c)
```



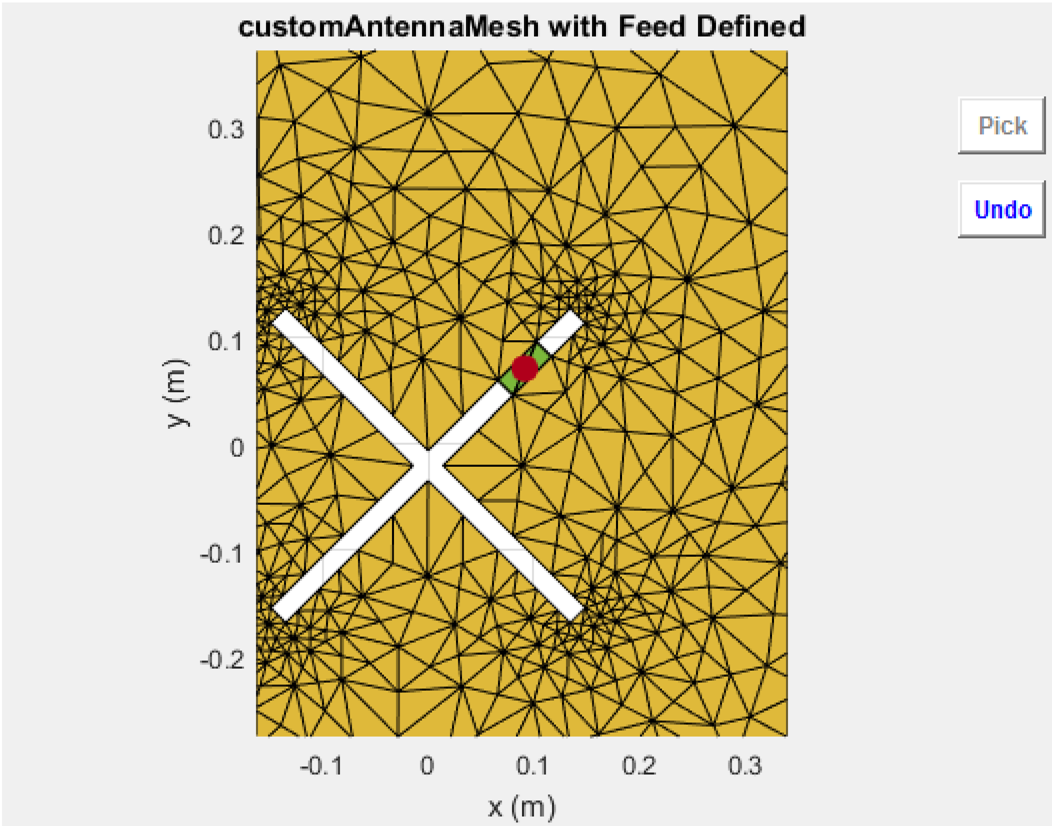
Click **Pick** to display the crosshairs. To specify a region for the feed point, zoom in and select two points, one inside each triangle on either side of the air gap. Select the points using the crosshairs.



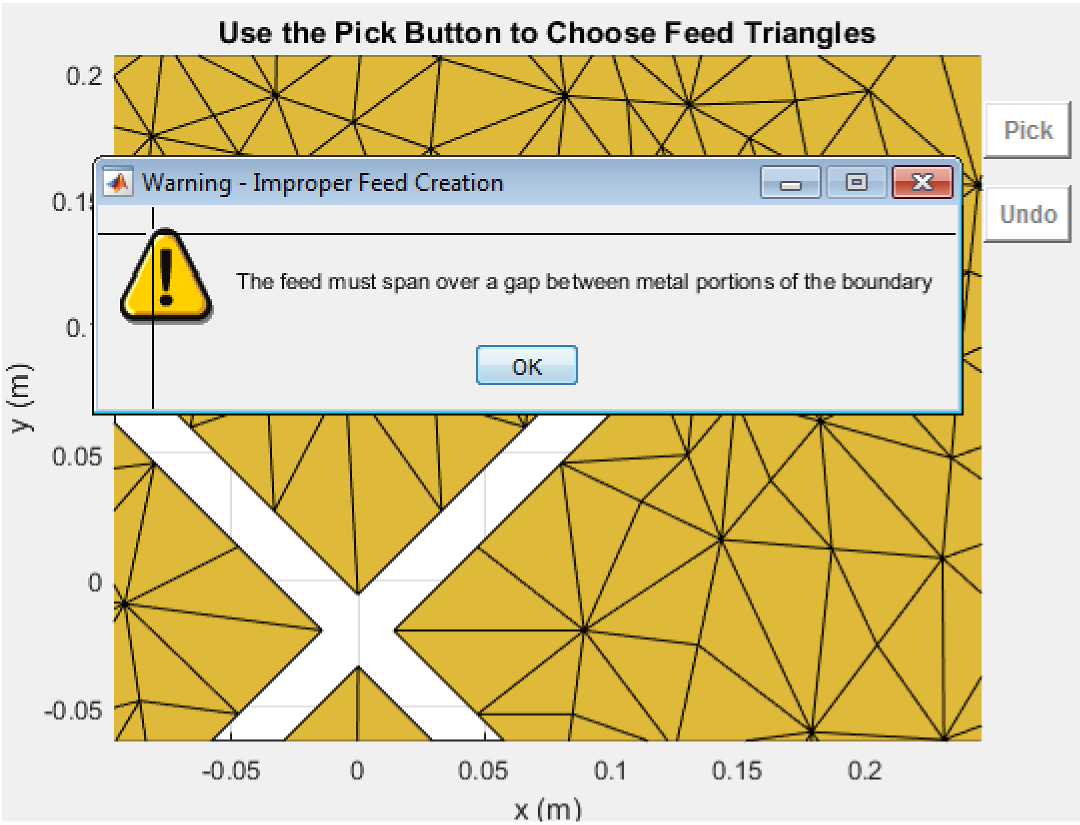
Use the Pick Button to Choose Feed Triangles
0.2
0.15
0.1
0.05
0
-0.05
y (m)
-0.05
0
0.05
0.1
0.15
0.2
0.25
x (m)
Triangle 1
Pick
Undo



Selecting the second triangle creates and displays the antenna feed.



You must select the two triangles on either side of the air gap. Otherwise, the function displays an error message.



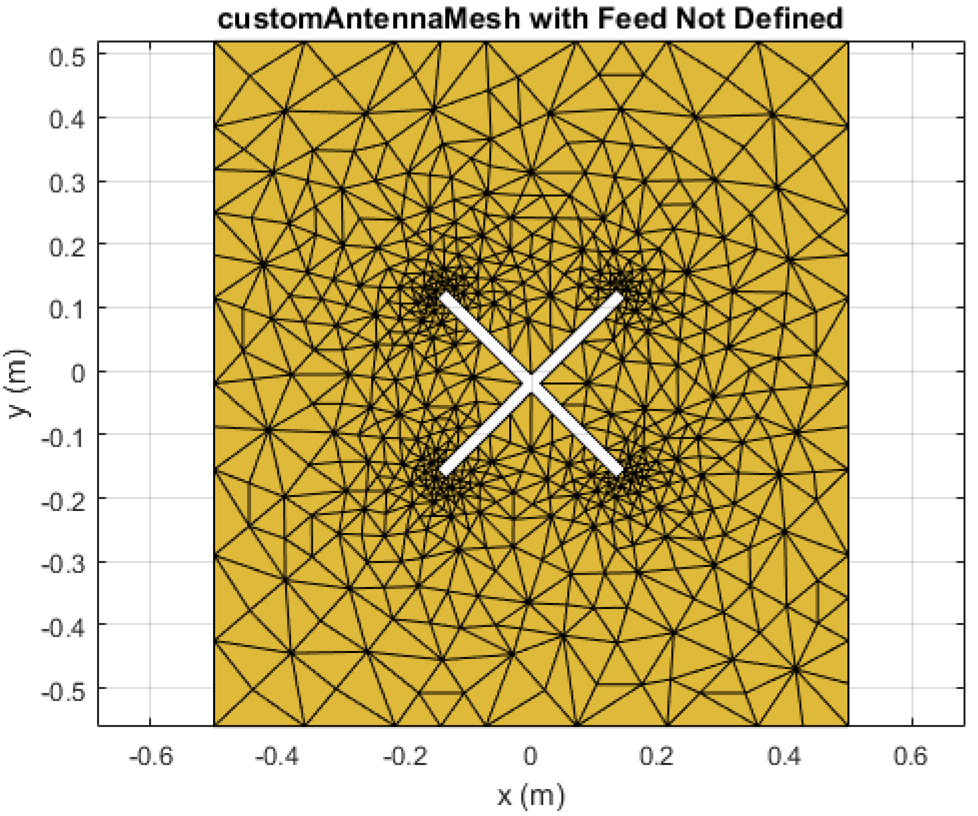
Create Feed for Custom Antenna Mesh

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)
show (c)
```

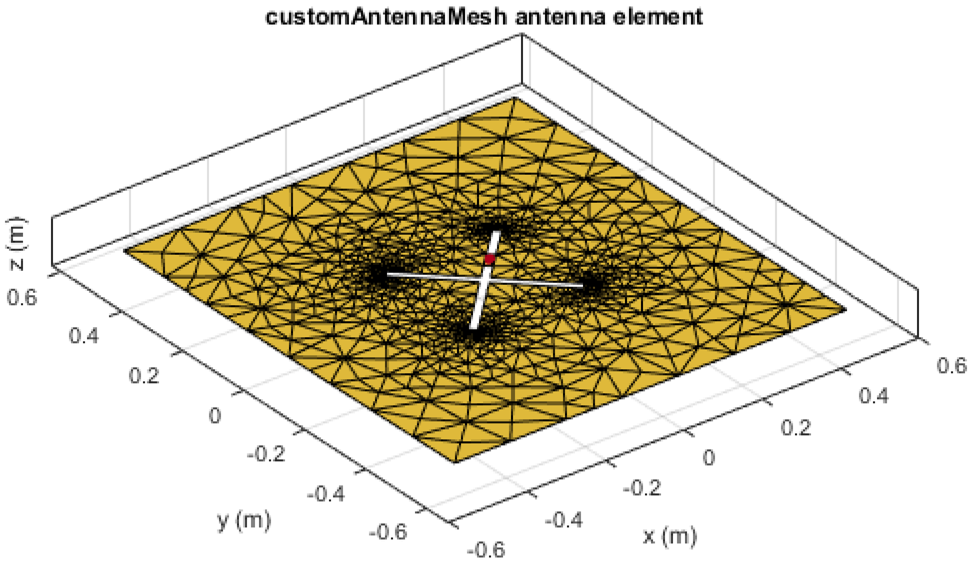
```
c =

  customAntennaMesh with properties:
```

```
        Points: [3x658 double]
     Triangles: [4x1219 double]
  FeedLocation: []
          Tilt: 0
      TiltAxis: [1 0 0]
```



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

Syntax

```
[e,h] = EHfields(object,frequency,points)
```

```
EHfields(object, frequency, points)
EHfields(object,frequency,points,Name,Value)
```

Description

`[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(object,frequency,points,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.

Examples

Calculate E and H 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 =

  -0.4283 - 0.2675i
  -0.3047 + 0.4377i
   0.0000 - 0.0000i


h =

   0.0008 - 0.0012i
  -0.0011 - 0.0007i
  -0.0000 - 0.0000i
```

Plot Electric and Magentic 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.

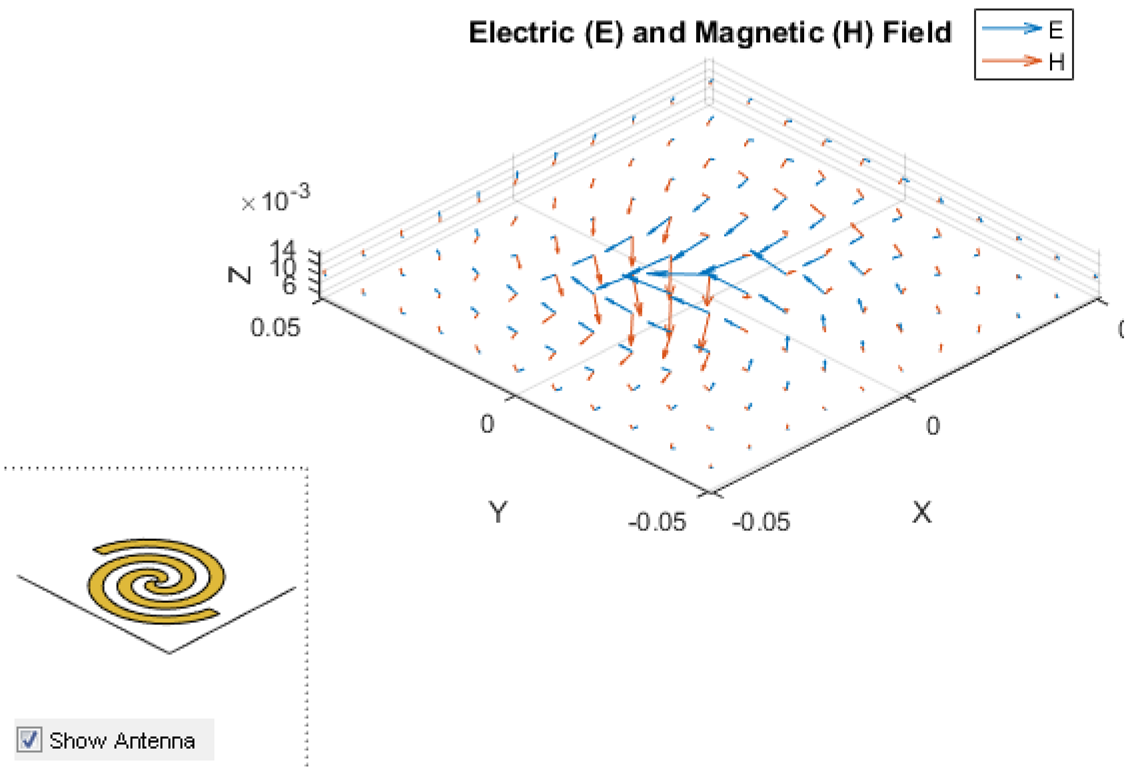
```
[X,Y] = meshgrid(-.05:.01:.05,-.05:.01:.05);
```

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)
```



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
string | `E` | `H`

Field to display, specified as the comma-separated pair consisting of `'ViewField'` and a string. `'E'` displays the electric field and `'H'` displays the magnetic field.

Example: `'ViewField'`, `'E'`

Data Types: `char`

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

```
ar= axialRatio(antenna,frequency,azimuth,elevation)
```

Description

`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
```

Input Arguments

`antenna` — Antenna object
scalar handle

Antenna object, specified as a scalar handle.

`frequency` — Frequency used to calculate axial ratio

scalar in Hz

Frequency used to calculate axial ratio, specified as a scalar in Hz.

Example: 70e6

Data Types: `double`

`azimuth` — Azimuth angle of antenna

scalar in degrees

Azimuth angle of antenna, specified as a scalar in degrees.

`elevation` — Elevation angle of antenna

scalar in degrees

Elevation angle of antenna, specified as a scalar 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

```
[bw] = beamwidth(antenna,frequency,azimuth,elevation)
[bw] = beamwidth(antenna,frequency,azimuth,elevation,dBdown)

[bw,angles] = beamwidth(____)
```

Description

`[bw] = beamwidth(antenna,frequency,azimuth,elevation)` returns 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's main beam.

`[bw,angles] = beamwidth(____)` returns the beamwidth and angles (points in a plane) using any input arguments from previous syntaxes.

Examples

Calculate Beamwidth for Antenna

Calculate the beamwidth for a helix at frequency=2GHz, azimuth=0, elevation=1:1:360 (x-z plane).

```
h  = helix;
[BW] = beamwidth(h,2e9,0,1:1:360,5)
```

```
BW =

    90
```

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 dBdown value of 5 dB.

```
hx = helix;
[bw,angles] = beamwidth(hx,2e9,1:1:360,0,5)
```

```
bw =

   141


angles =

   147   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 antenna or array structure

Syntax

```
mesh(object,Name,Value)
```

Description

`mesh(object,Name,Value)` changes and plots the mesh structure of an antenna or array object, using additional options specified by the name-value pair. You can also determine the number of unknowns from the number of basis functions in the output.

Examples

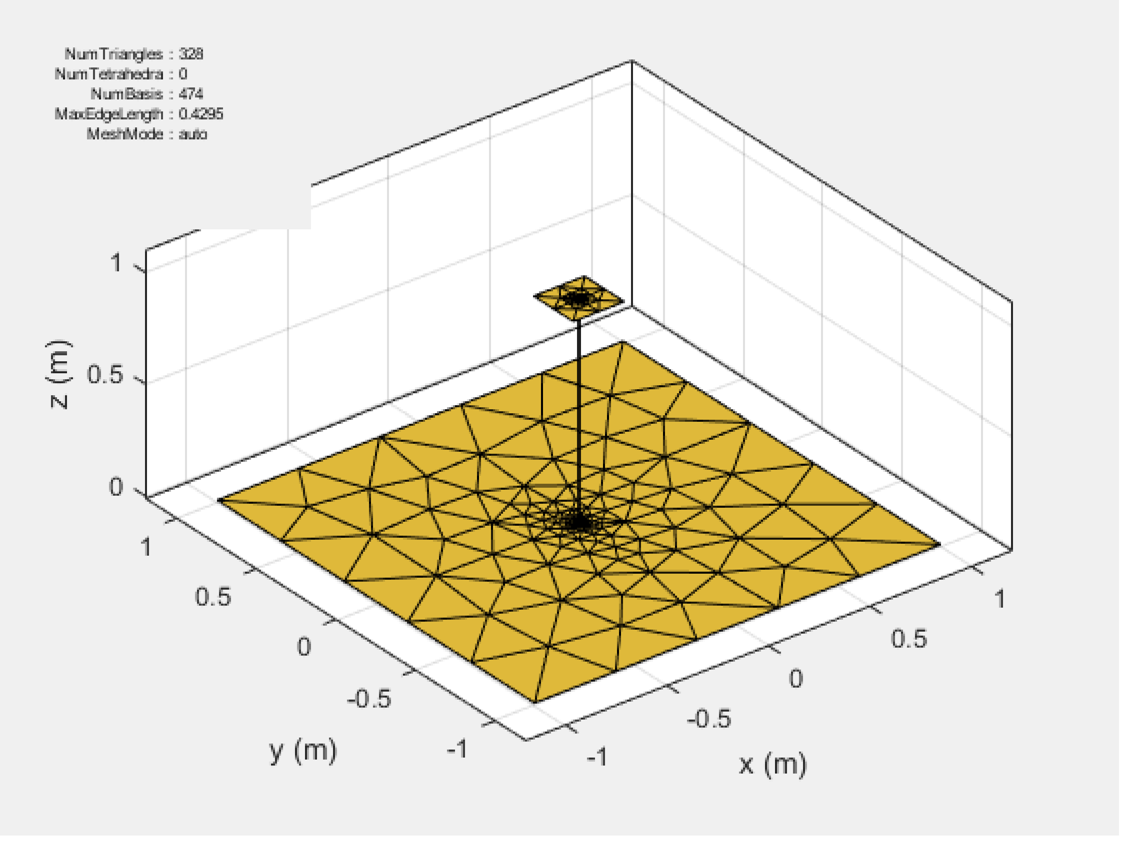
View Mesh Structure of Antenna

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)
mesh(h)
m = mesh(h)
```

```
i =

   2.7245e+02 + 6.0930e+02i


m =

     NumTriangles: 328
    NumTetrahedra: 0
```

```
     NumBasis: 474
MaxEdgeLength: 0.4295
     MeshMode: 'auto'
```



Input Arguments

object — Antenna or array object
scalar handle

Antenna or array object, specified as a scalar handle.

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'`.

See Also

`show`

Introduced in R2015a

layout

Display array layout

Syntax

`layout(array)`

Description

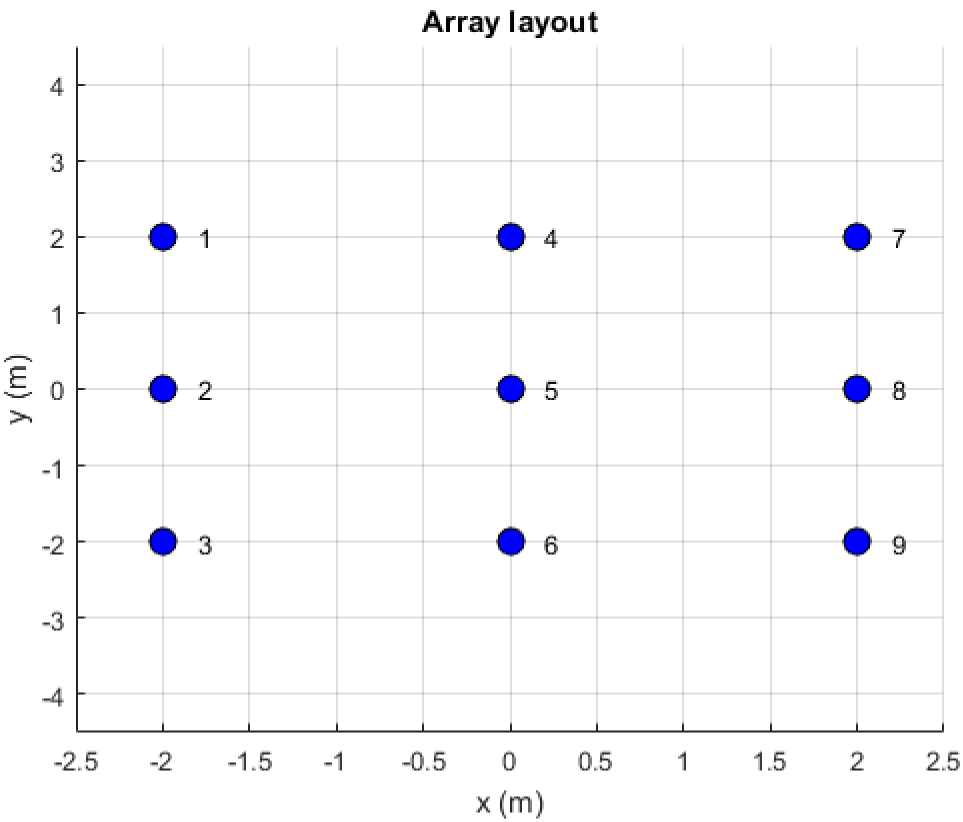
`layout(array)` displays the layout of the array object. The circles in the layout corresponds to antenna feed points within the array.

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

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`.

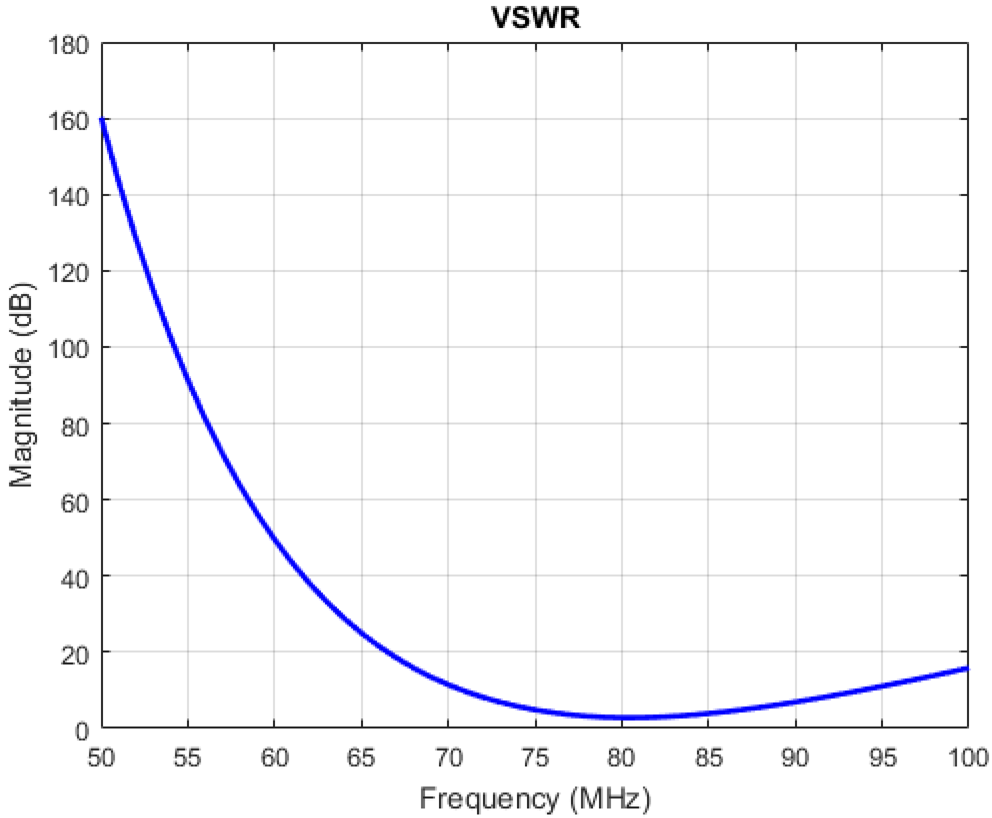
`vswrant = vswr(antenna,frequency,z0)` returns the vswr of the antenna.

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 =

    3.6021    6.6244    3.2850
```

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.

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`.

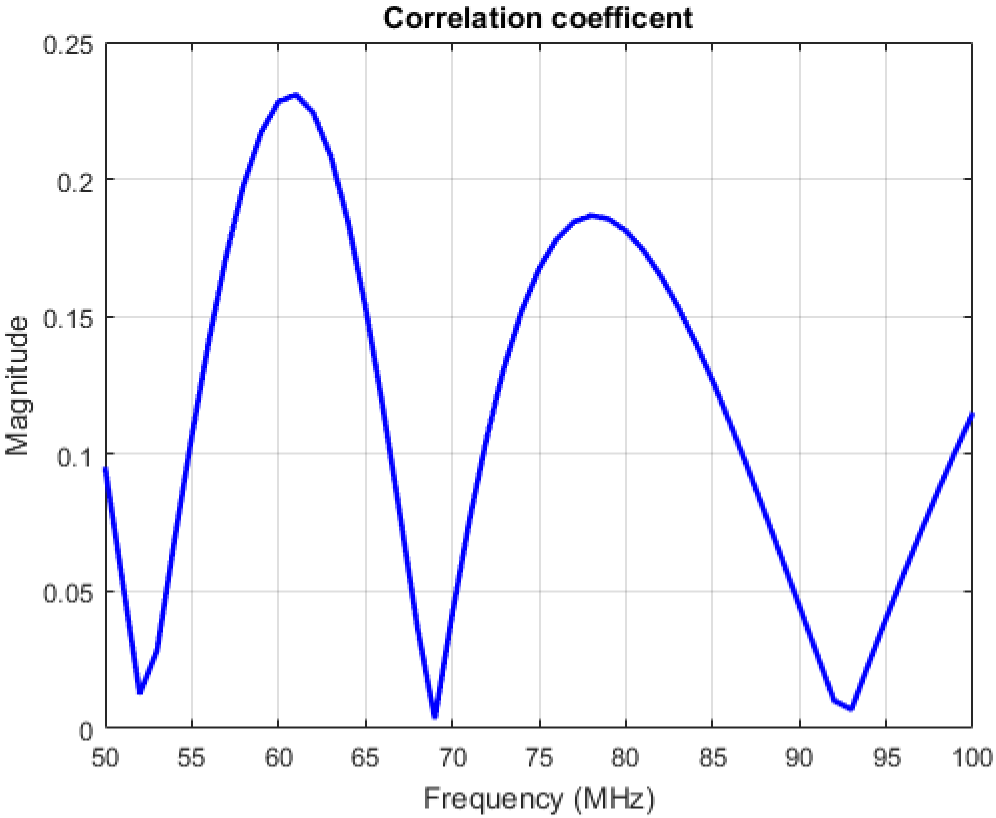
`rho = correlation(array,frequency,elem1,elem2,z0)` returns the correlation coefficient between two antenna elements, `elem1` and `elem2` of an array.

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 Correleation Coeffecient of Array

Calculate correlation coeffecient of default rectangular array at a frequency range of 50MHz to 100MHz.

```
h = rectangularArray;
rho = correlation (h, 50e6:1e6:100e6, 1, 2)
```

```
rho =

    0.1377
    0.1081
    0.0782
```

```
0.0477
0.0165
0.0156
0.0486
0.0822
0.1153
0.1463
0.1725
0.1912
0.1999
0.1977
0.1850
0.1635
0.1355
0.1030
0.0675
0.0301
0.0084
0.0474
0.0862
0.1235
0.1578
0.1868
0.2081
0.2195
0.2193
0.2076
0.1859
0.1568
0.1236
0.0892
0.0559
0.0252
0.0022
0.0261
0.0466
0.0641
0.0789
0.0914
0.1020
0.1110
0.1186
0.1252
0.1309
```

```
0.1359
0.1403
0.1442
0.1478
```

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 =

    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 =
```

```
    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 =

    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 =

    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 string input mode.

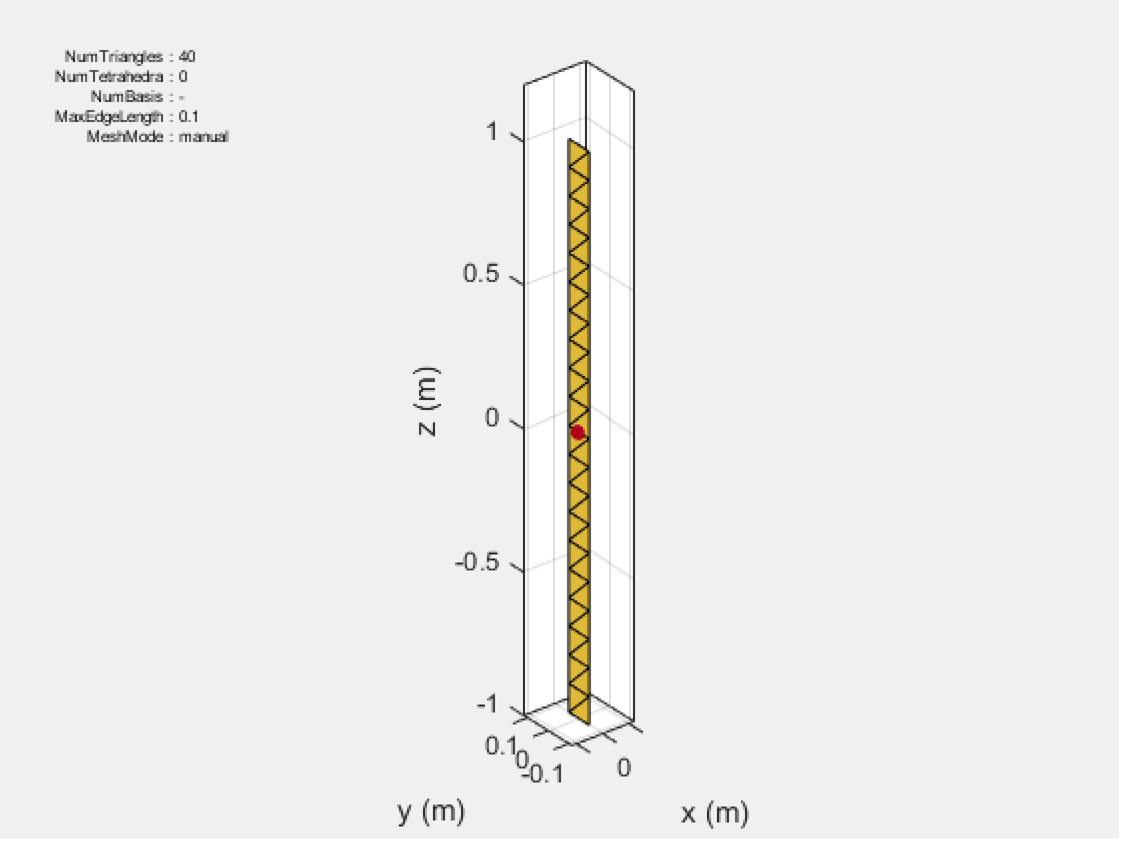
Examples

Change Mesh Configuration of Antenna

Change the mesh configuration of a dipole antenna from auto (default) to manual mode.

```
h = dipole;
meshconfig(h,'manual')
mesh(h,'MaxEdgeLength',0.1)
```

```
ans = 

     NumTriangles: 0
    NumTetrahedra: 0
         NumBasis: []
    MaxEdgeLength: []
         MeshMode: 'manual'
```



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'`.

See Also

`mesh` | `show`

Introduced in R2015a

numSummationTerms

Class: infiniteArray

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 =

  93.6494 +79.7794i
```

Change the number of summation terms to 51. Calculate the scan impedance again.

```
numSummationTerms(h,25)
s = impedance(h,1e9)
```

```
s =

  93.8121 +79.8081i
```

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 =

  93.8622 +79.8103i
```

See Also

beamwidth | pattern

More About

- "Infinite Arrays"

Introduced in R2015b

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.

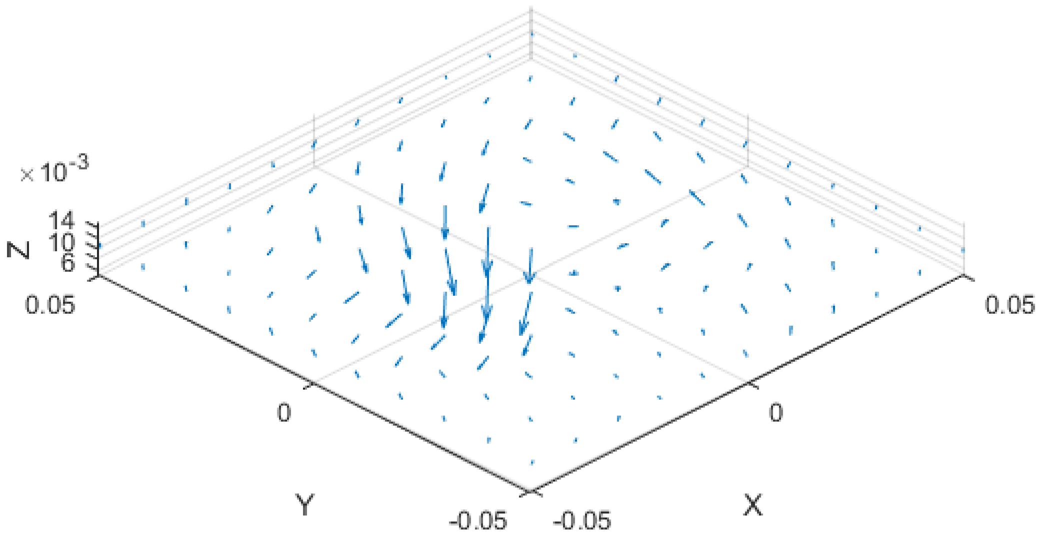
`fieldsCustom(axeshandle, ___ )` plots into the axes specified by `axeshandle` instead of the current axes.

Examples

Visualize Magnetic Field of Antenna

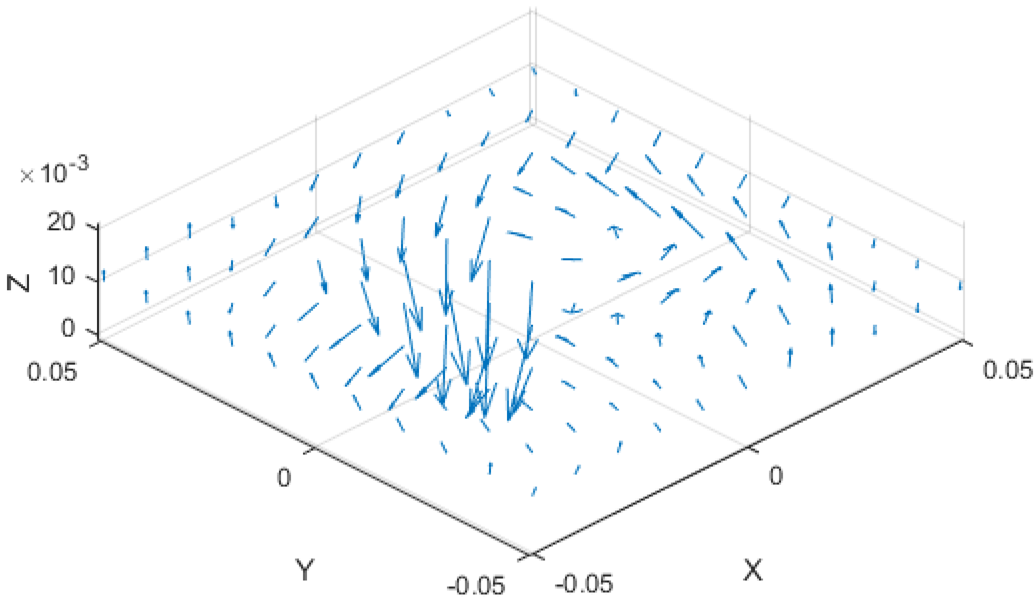
Load and visualize the magnetic field data available in the file `fielddata.mat`.

```
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: `string`

`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.

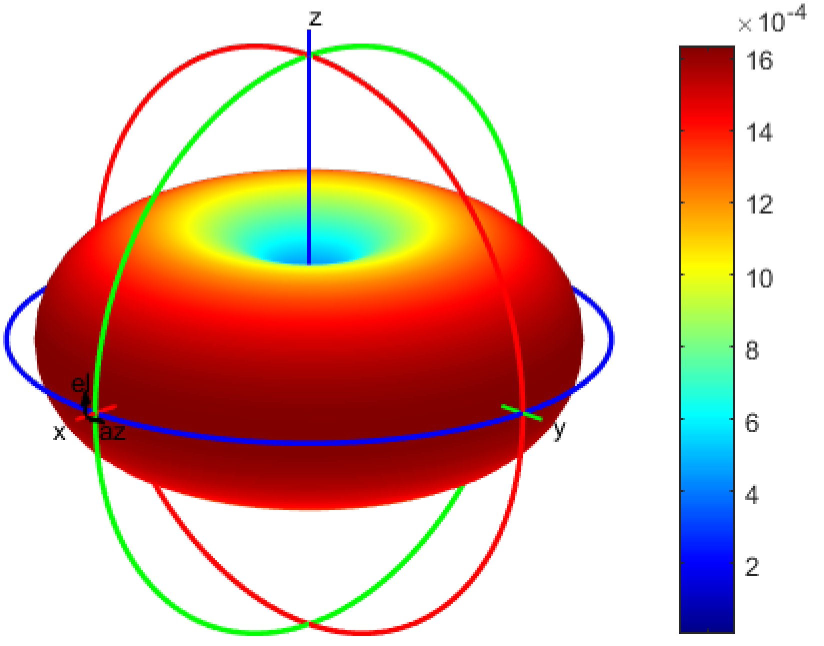
```
d = dipole;
[efield,az,el] = pattern(d, 75e6,'Type','efield');
```

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);
```

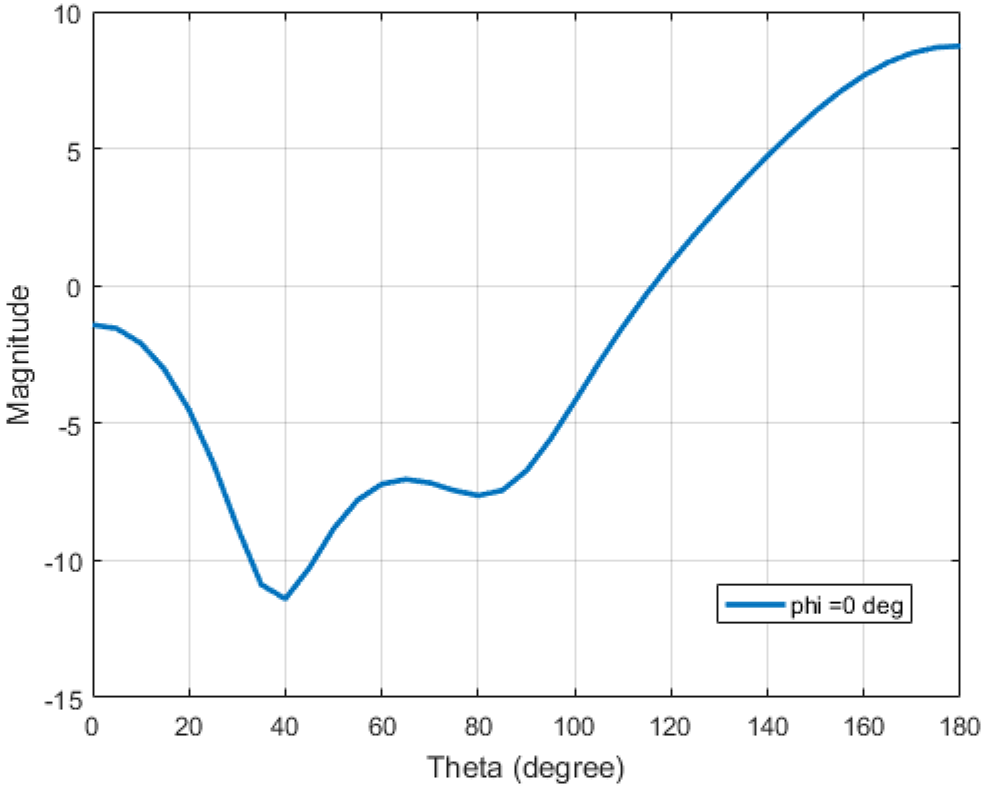
Extract theta and phi angles of the directivity magnitude.

```
phi = az';
```

```
theta = (90-el);
MagE = D';
```

Plot 2-D phi slice of the antenna in rectangular coordinates.

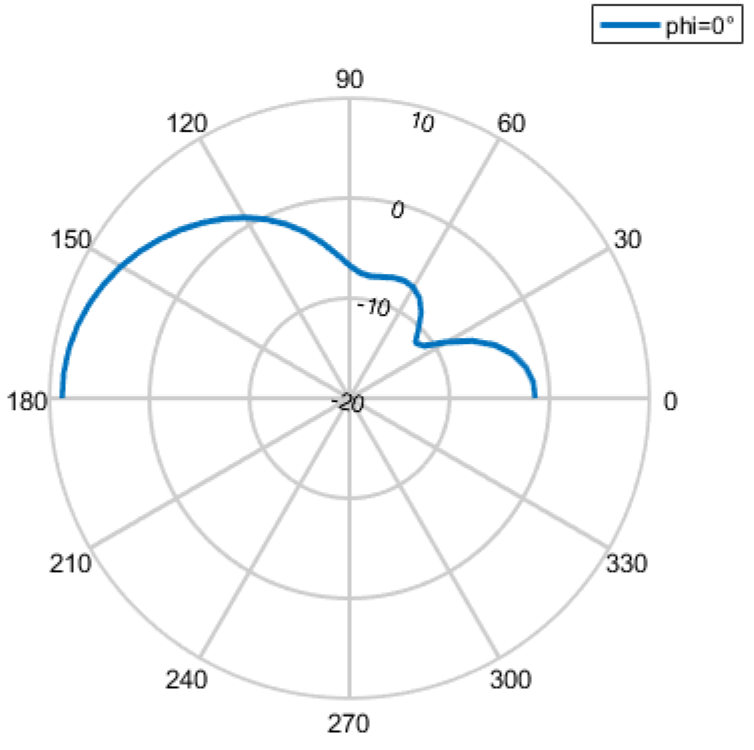
```
figure;
patternCustom(MagE,theta,phi,'CoordinateSystem','rectangular',...
    'Slice','phi','SliceValue',0);
```



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

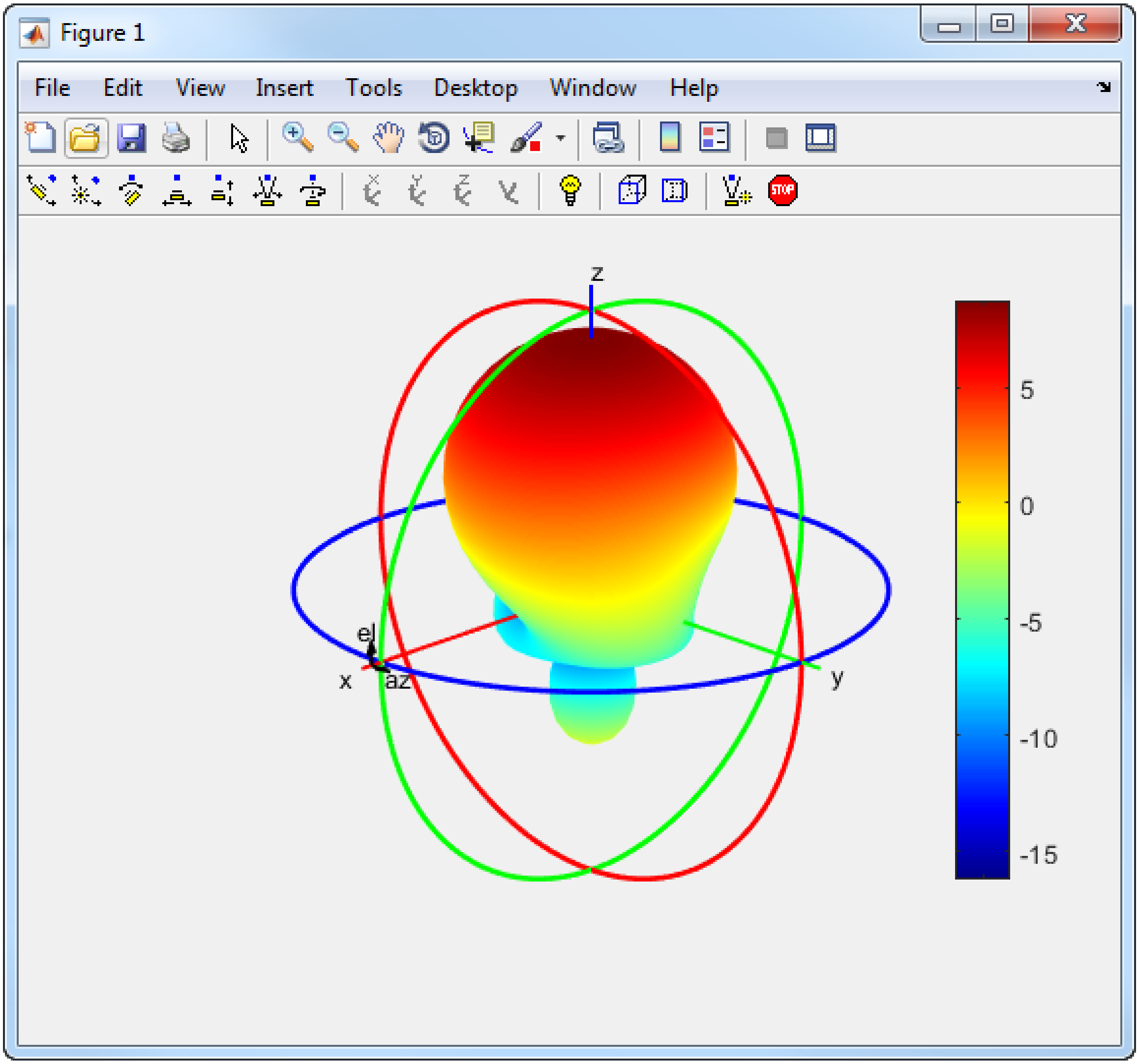
Consider a helix antenna data file in `.csv` format. This file contains the magnitude of the antenna directivity in `phi` and `theta` angles. Read the file .

Read the `.csv` data file.

```
helixdata = csvread('antennadata_test.csv',1,0);
```

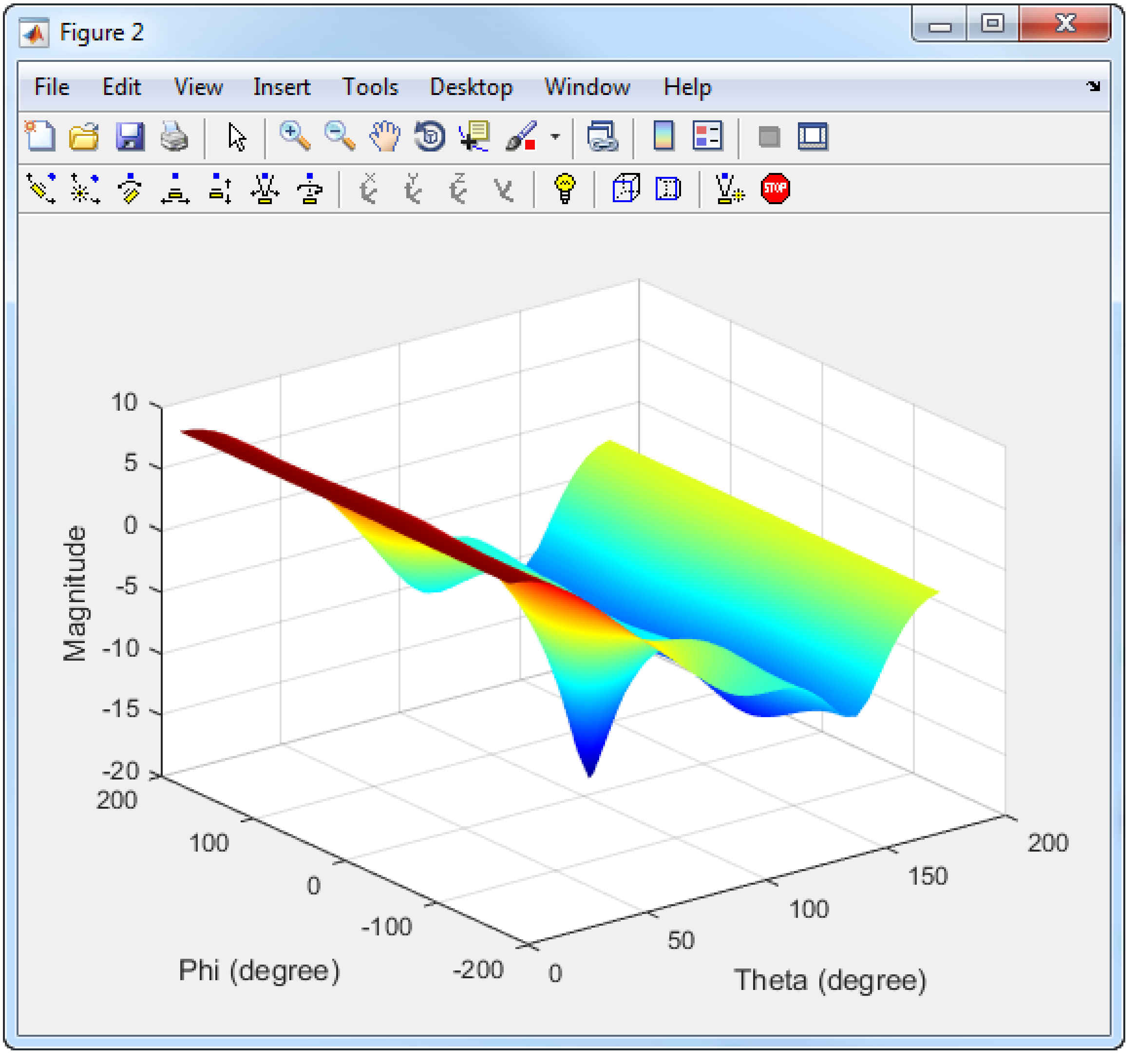
Use `patternCustom` to extract the magnitude of directivity, and the `phi`, and `theta` angle values. Plot the 3-D polar radiation pattern.

```
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','rectangu
```



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 `theta` and `phi` angle vectors.
- A M-by-R matrix. M is the same size as the `theta` angle vector and R is the same size as the `phi` angle 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 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'`

Coordinate system of radiation pattern, specified as the comma-separated pair consisting of `'CoordinateSystem'` and one of these strings: `'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

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.

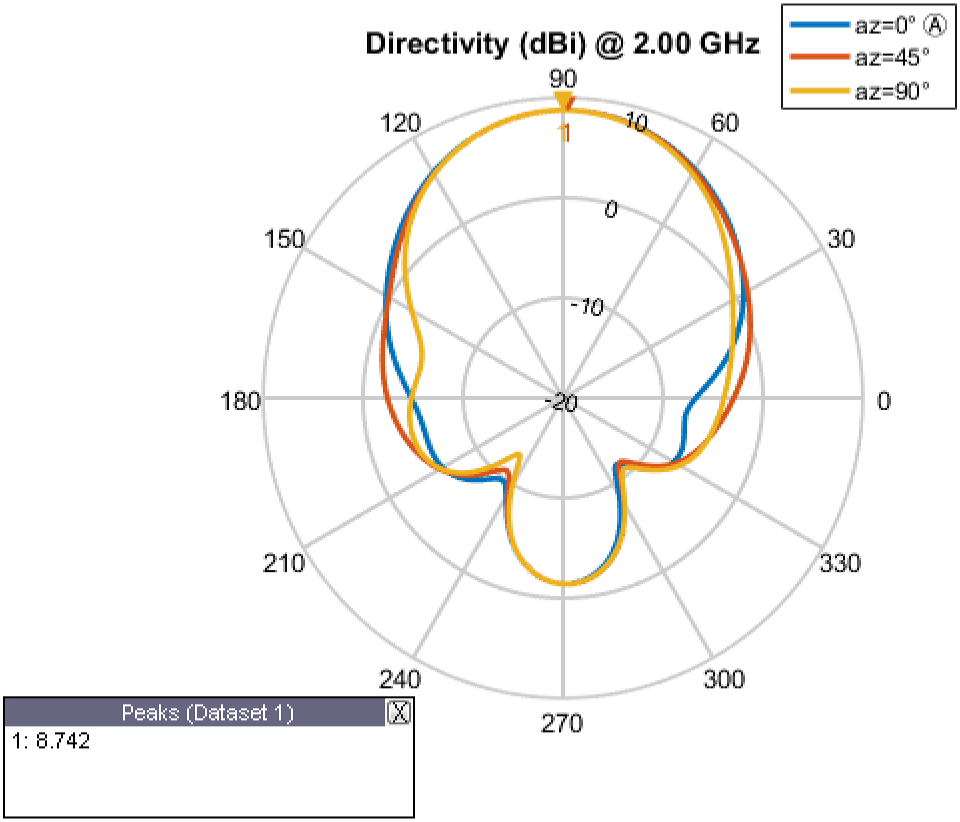
`[horizontal,vertical,optional] = msiread(fname)` reads the file and returns structures containing horizontal gain data, vertical gain data, and all additional data in the file.

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.plnto 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 =

    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 =

    PhysicalQuantity: 'Gain'
           Magnitude: [360x1 double]
               Units: 'dBd'
             Azimuth: [360x1 double]
           Elevation: 0
           Frequency: 659000000
               Slice: 'Elevation'


Vertical =

    PhysicalQuantity: 'Gain'
           Magnitude: [360x1 double]
               Units: 'dBd'
             Azimuth: 0
           Elevation: [360x1 double]
```

```
          Frequency: 659000000
              Slice: 'Azimuth'


Optional =

             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'
```

- "Read, Visualize and Write MSI Planet Antenna Files"

Input Arguments

`fname` — Name of MSI file

string

Name of MSI file, specified as a string. 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 these strings: `'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 these strings: `'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 a string. 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 these strings: `'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 these strings: `'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 a string. 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`

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`.

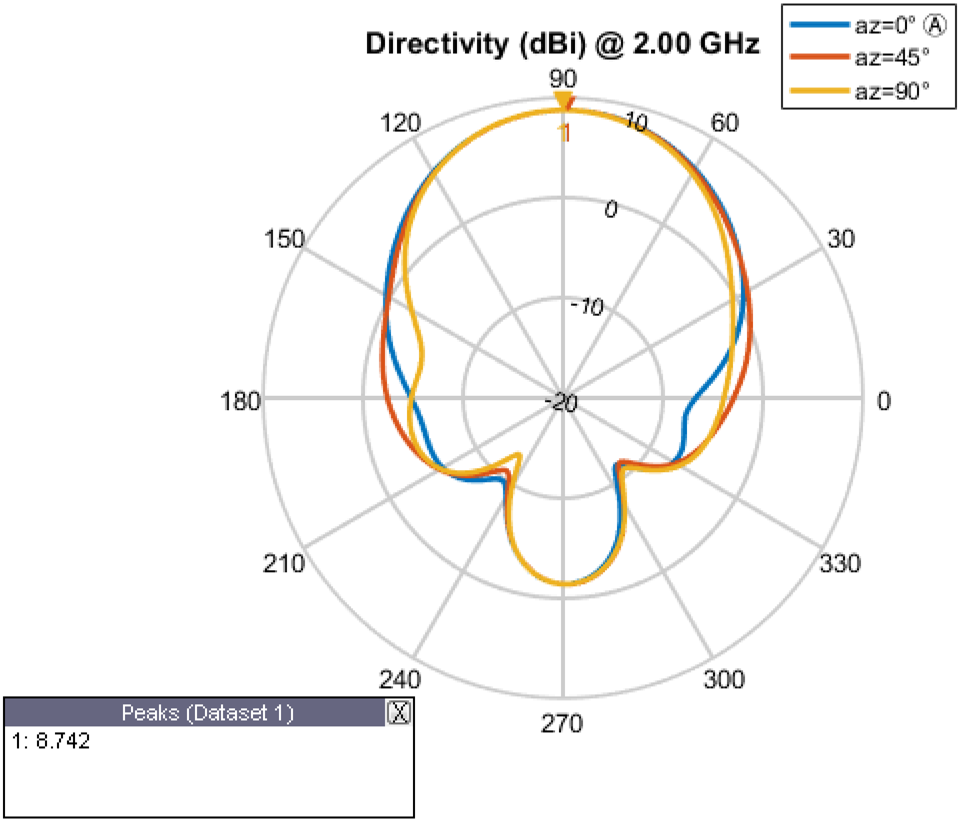
`msiwrite(objname,frequency,fname,Name,Value)` uses additional options specified by one or more `Name,Value` pair arguments.

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.plnto 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 =

    PhysicalQuantity: 'Gain'
           Magnitude: [360x1 double]
               Units: 'dBi'
             Azimuth: [360x1 double]
           Elevation: 0
           Frequency: 2.0000e+09
               Slice: 'Elevation'
```

- “Read, Visualize and Write MSI Planet Antenna Files”

Input Arguments

fname — Name of MSI file

.pln (default) | string

Name of MSI file, specified as a string. 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 single 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
string

Title of file in the first line, specified as the comma-separated pair consisting of `'Name'` and a string.

Example: `'Name','Designed Helix Antenna in MATLAB'`

Data Types: `char`

`'Comment'` — Comments about antenna or array data file
string

Comments about an antenna or array data file, specified as the comma-separated pair consisting of `'Comment'` and a string.

Example: `'Comment','This antenna is for space simulations.'`

Data Types: `char`

See Also
`msiread`

Introduced in R2016a

dielectric

Dielectric material for use as substrate

Syntax

```
d = dielectric(material)
d = dielectric(Name,Value)
```

Description

`d = dielectric(material)` returns a dielectric material for use as a substrate in antenna elements.

`d = dielectric(Name,Value)` returns a dielectric material, based on the properties specified by one or more `Name,Value` pair arguments.

Examples

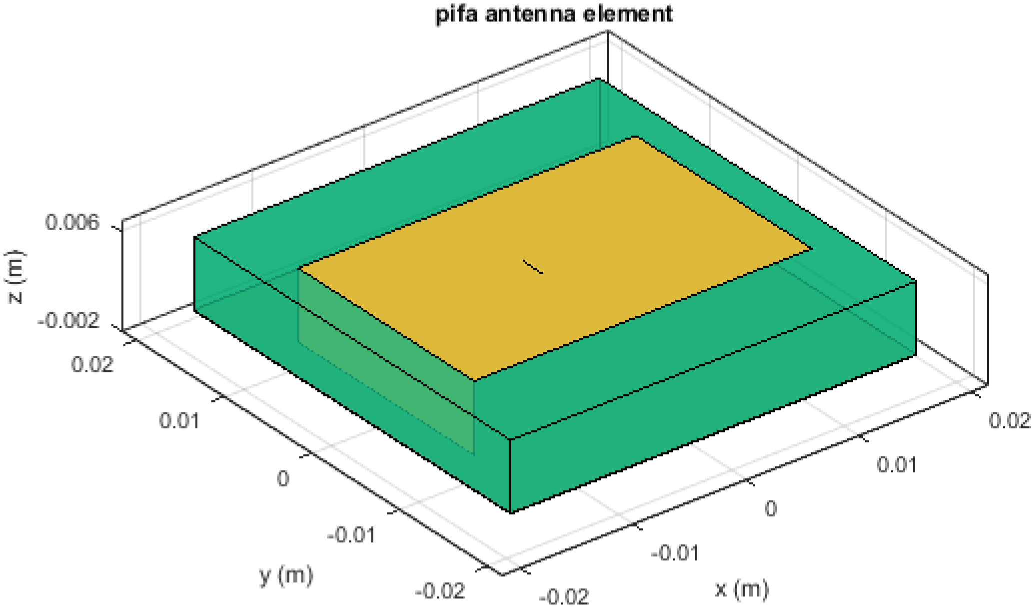
PIFA Antenna with Dielectric Substrate

Use a Teflon dielectric material as a substrate for a PIFA antenna. View the antenna.

```
d = dielectric('Teflon')
p = pifa('Height',0.0060,'Substrate',d)
show(p)
```

```
d =

  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>

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]
```



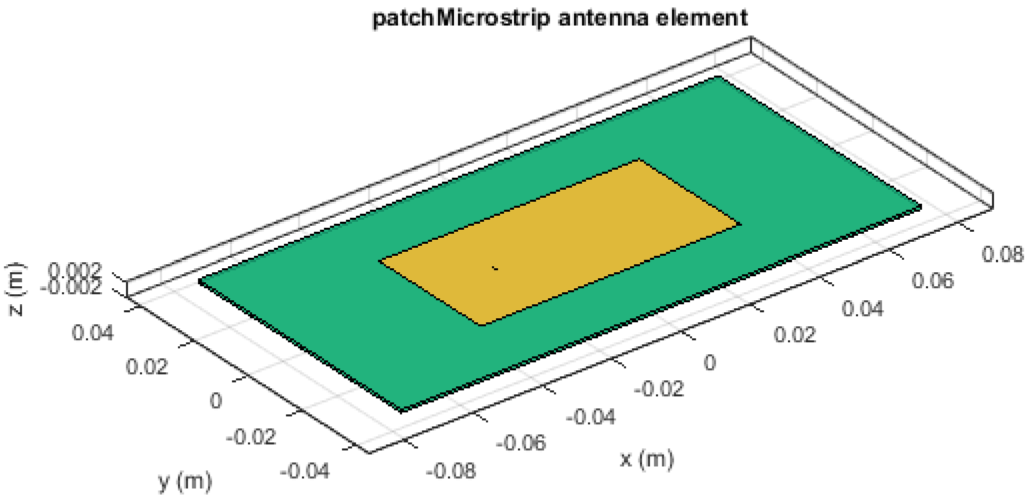
Custom Dielectric Properties

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 0.008 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)
show(p)
```

```
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]
```



Input Arguments

`material` — Material from dielectric catalog
`'Air'` (default) | string

Material from the dielectric catalog, specified as a string. For more information on the different dielectric elements in Antenna Toolbox refer, `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 single 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

string

Name of the dielectric material you want to specify in the output, specified as the comma-separated pair consisting of `'Name'` and a string.

Example: `'Name'`,`'Taconic_TLC'`

Data Types: `char`

`'EpsilonR'` — Relative permittivity of dielectric material

`1` | scalar

Relative permittivity of the dielectric material, specified as the comma-separated pair consisting of `'EpsilonR'` and a scalar.

Example: `'EpsilonR'`,`4.8000`

Data Types: `double`

`'LossTangent'` — Loss in dielectric material

`0` (default) | scalar

Loss in the dielectric material, specified as the comma-separated pair consisting of `'LossTangent'` and a scalar.

Example: `'LossTangent'`,`0.0260`

Data Types: `double`

`'Thickness'` — Thickness of dielectric material

`0.0060` (default) | scalar in meters

Thickness of the dielectric material along default z-axis, specified as the comma-separated pair consisting of `'Thickness'` and a scalar 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.

More About

- "Antenna Toolbox Limitations"

See Also

`DielectricCatalog`

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	[illegible]	[illegible]	[illegible]	

List the properties of the dielectric material `Foam`.

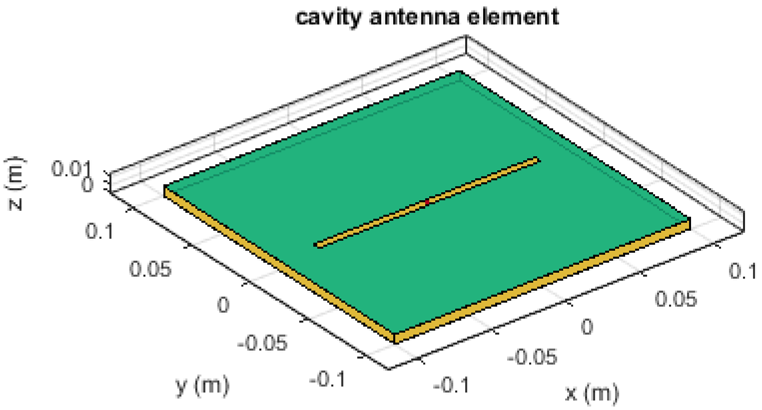
```
s = find(dc,'Foam')
```

```
s = 

                   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)
show (c)
```

```
c = 

  cavity with properties:

      Exciter: [1x1 dipole]
    Substrate: [1x1 dielectric]
       Length: 0.2000
        Width: 0.2000
       Height: 0.0060
      Spacing: 0.0060
         Tilt: 0
     TiltAxis: [1 0 0]
```



Input Arguments

`name` — Name of dielectric material

`'Air'` (default) | string

Name of a dielectric material from the dielectric catalog, specified as a string.

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 height from flare angles

Syntax

```
[flarewidth,flareheight]= horn2angle(width,height,flarelength,
angleE,angleH)
```

Description

`[flarewidth,flareheight]= horn2angle(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`

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, `d`.

`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 data

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

Antenna data, specified as one of the following:

- A real length-M vector, where M contains the magnitude values with angles assumed to be $\frac{(0:M-1)}{M \times 360}$ 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 $\frac{(0:M-1)}{M \times 360}$ degrees. The set of each angle can vary for each column.
- `N`-by-`D` arrays, 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 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

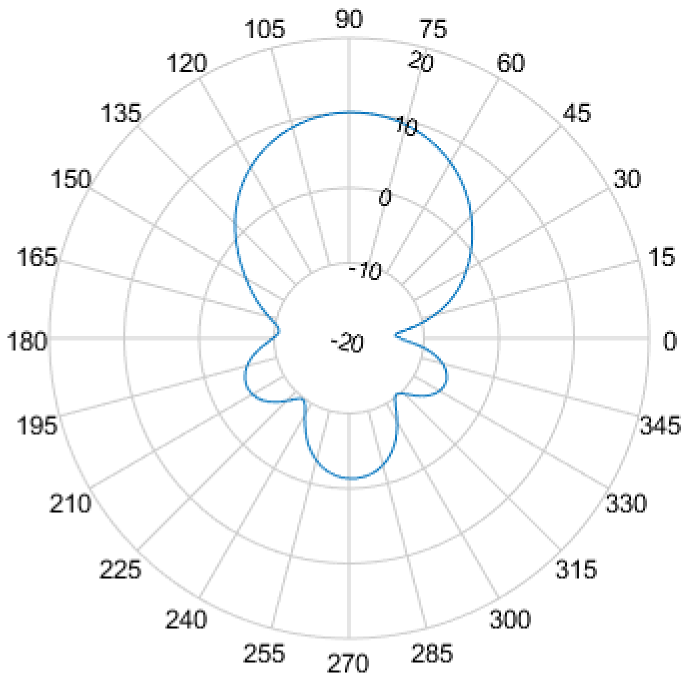
Add Data To Polar Plot

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);
```

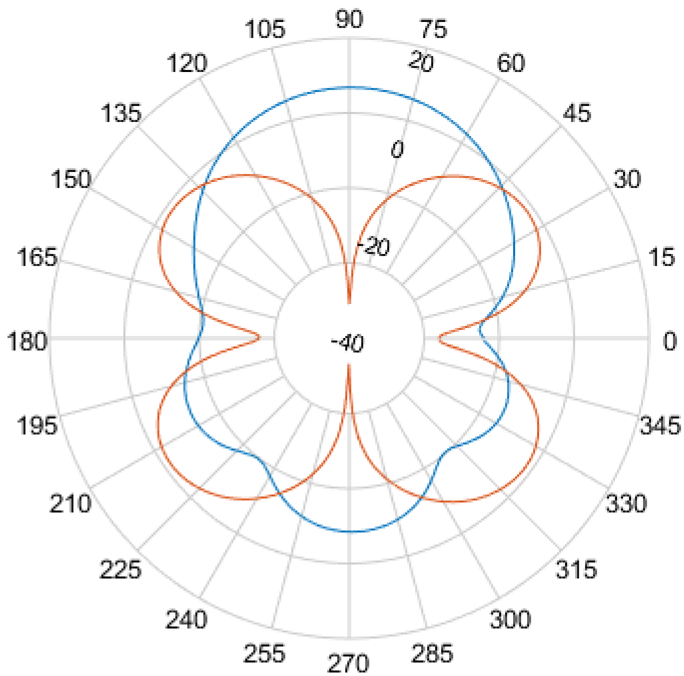


Create a dipole antenna and calculate the directivity at 270 MHz.

```
d = dipole;
D = pattern(d,270e6,0,0:1:360);
```

Add the directivity of the dipole to the existing polar plot of helix antenna.

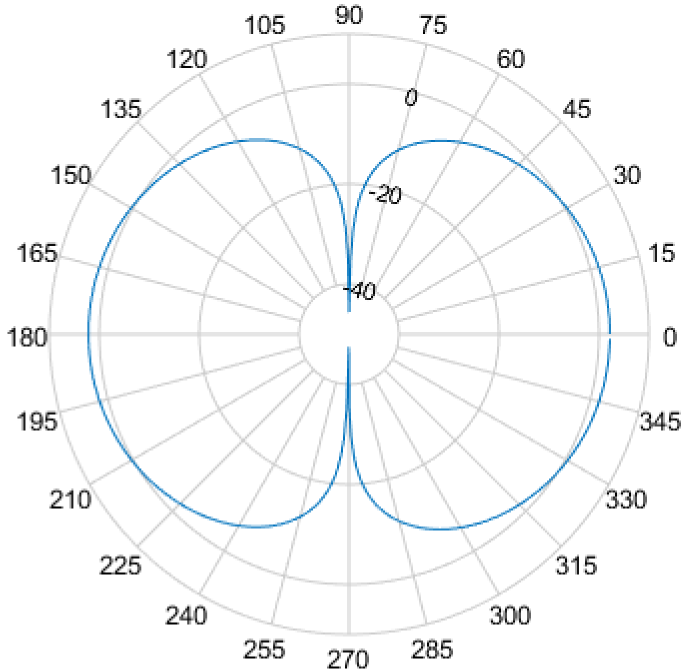
```
add(P,D);
```



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.

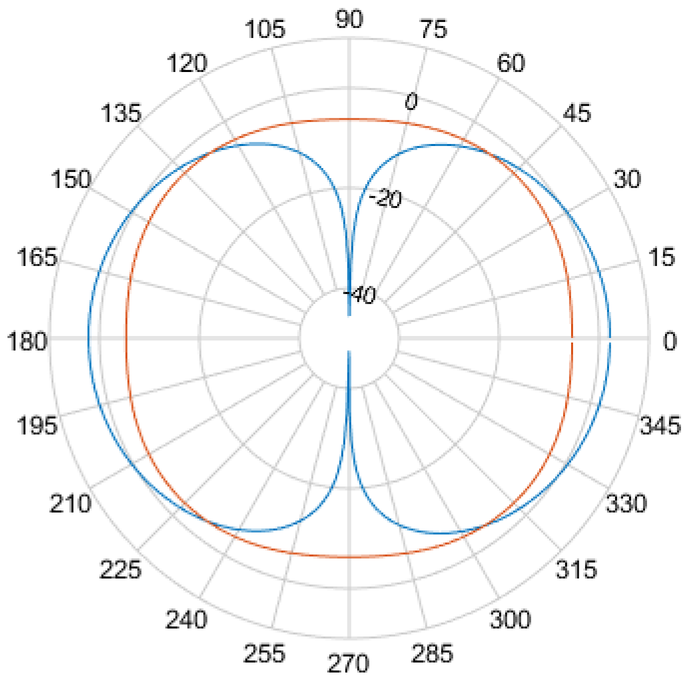
```
[Horizontal,Vertical,Optional] = msiread('cavity.pln')
```

```
Horizontal =
```

```
    PhysicalQuantity: 'Gain'
           Magnitude: [360x1 double]
               Units: 'dBi'
             Azimuth: [360x1 double]
           Elevation: 0
           Frequency: 1.0000e+09
               Slice: 'Elevation'


Vertical =

    PhysicalQuantity: 'Gain'
           Magnitude: [360x1 double]
               Units: 'dBi'
             Azimuth: 0
           Elevation: [360x1 double]
           Frequency: 1.0000e+09
               Slice: 'Azimuth'


Optional =

         name: 'Cavity Antenna Specifications'
    frequency: 1.0000e+09
         gain: [1x1 struct]
```

Add horizontal directivity data of the cavity antenna to the exisiting 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 string 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.

Example:

`index` — Data set index
scalar | vector

Data set index, specified as a scalar or a vector.

Examples

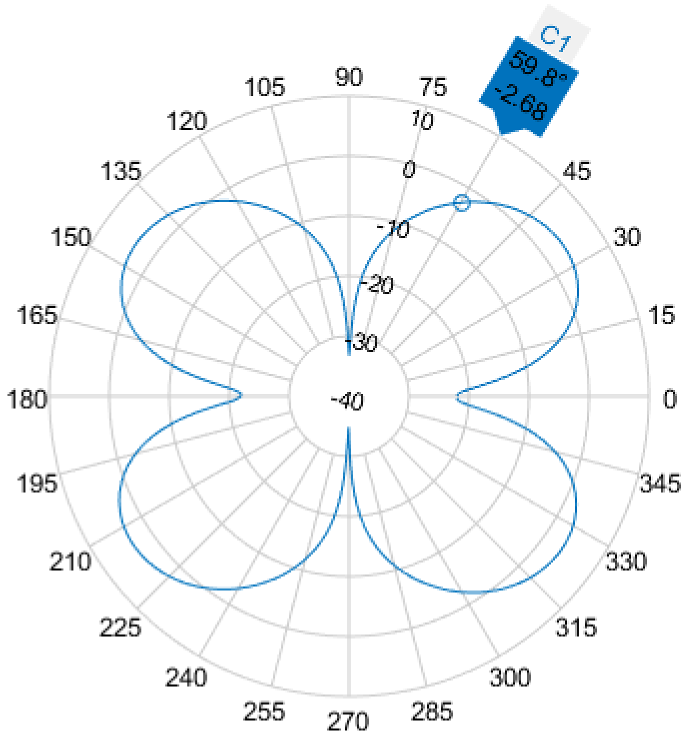
Add Cursor to Plot

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.

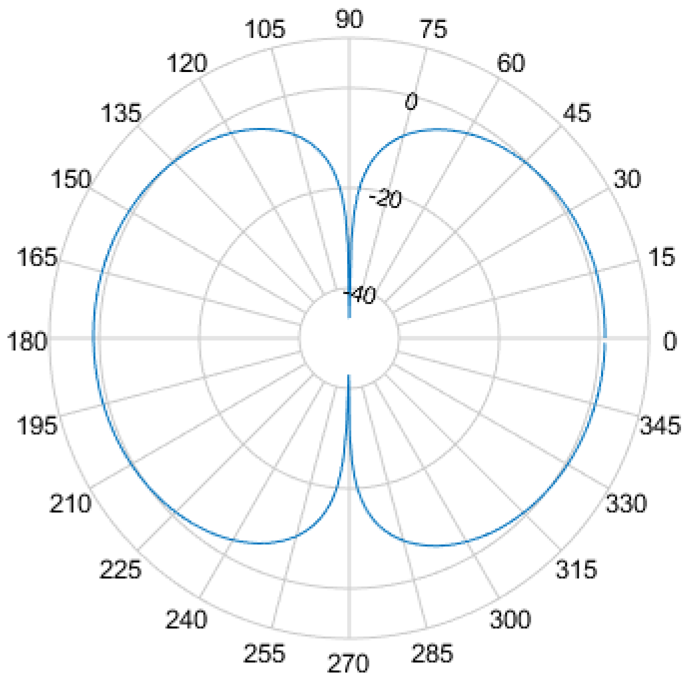
```
p = polarpattern(D);
addCursor(p,60);
```



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);
```

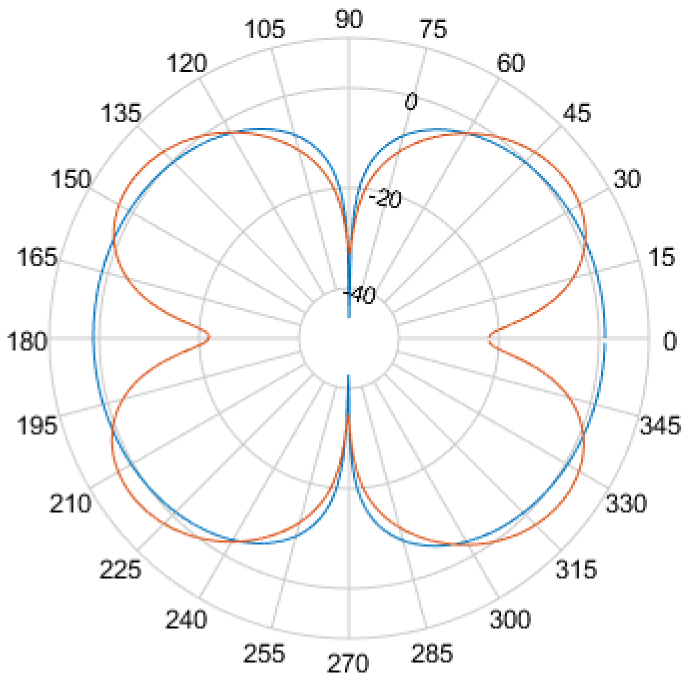


Create a dipole antenna and calculate its directivity at 270 MHz.

```
d = dipole;
D = pattern(d,270e6,0,0:1:360);
```

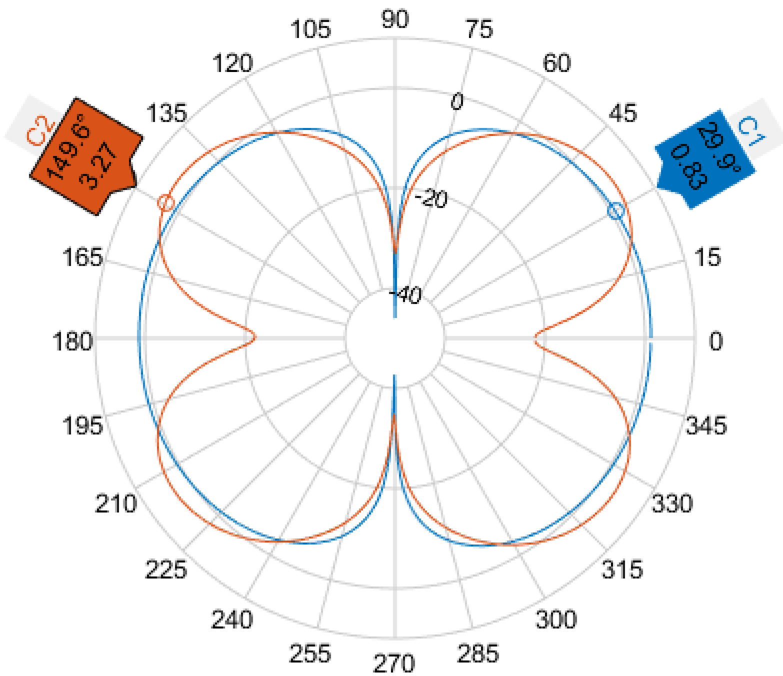
Add the directivity pattern of the dipole to the polar plot of the top-hat monopole.

```
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 data
real length-M vector | real M-by-N matrix | `N`-by-`D` arrays | complex vector or matrix

Antenna data, specified as one of the following:

- A real length-M vector, where M contains the magnitude values with angles assumed to be $\frac{(0:M-1)}{M \times 360}$ 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 $\frac{(0:M-1)}{M \times 360}$ degrees. The set of each angle can vary for each column.
- N-by-D arrays, 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 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

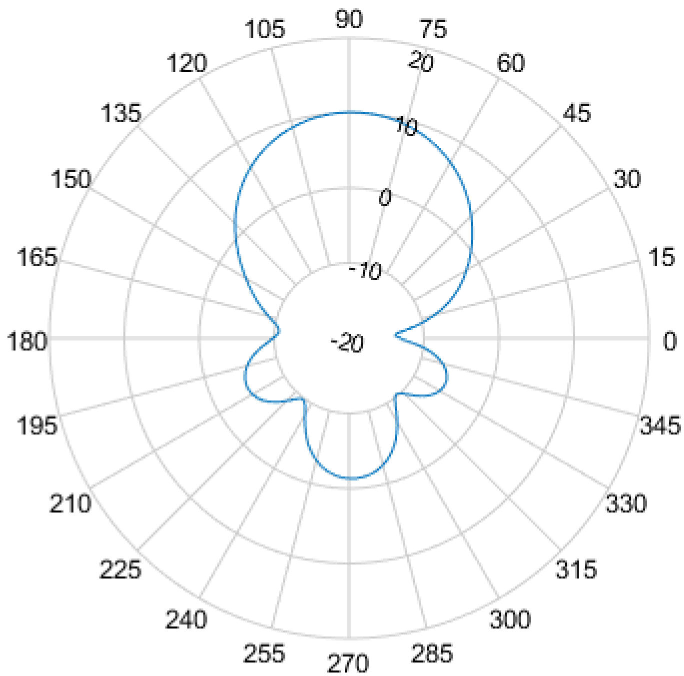
Set of magnitude values, specified as a vector or a matrix. For a matrix of magnitude values, each column is an independent set of magnitude values and corresponds to the same set of angles.

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);
```

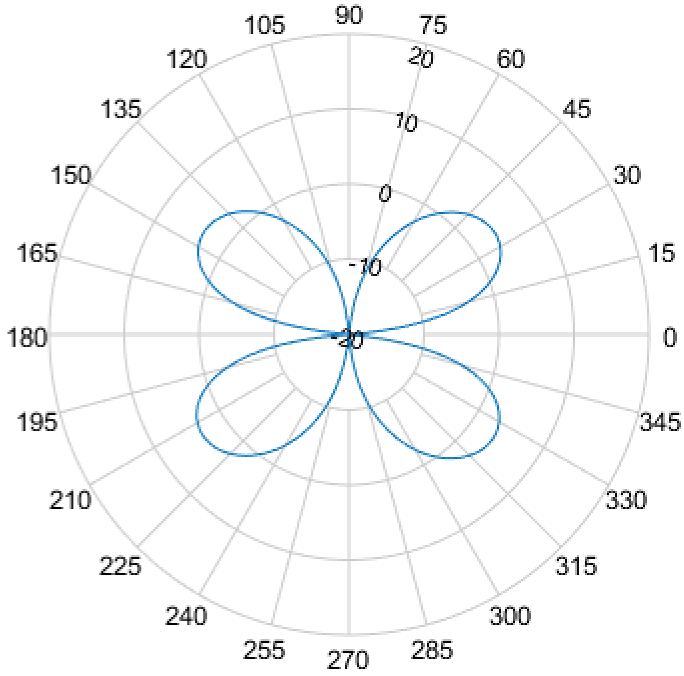


Create a dipole antenna and calculate its directivity at 270 MHz.

```
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.

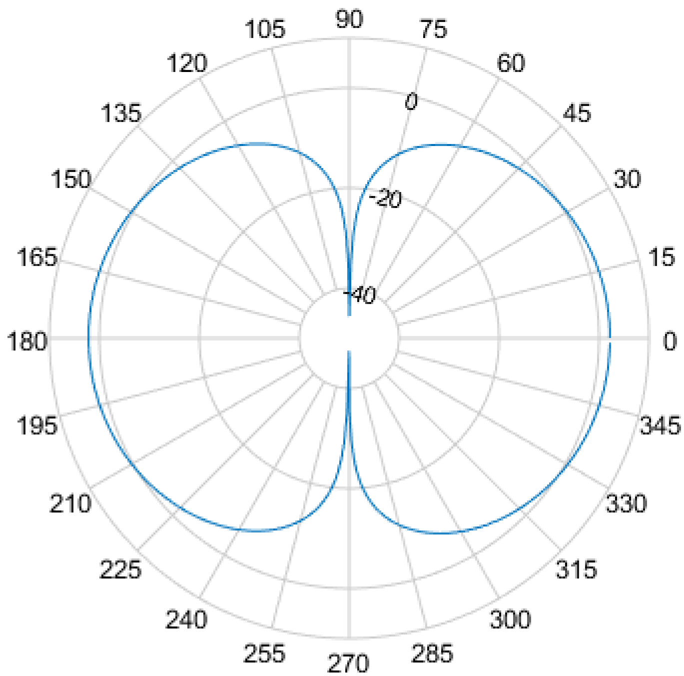
```
animate(P,D);
```



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.

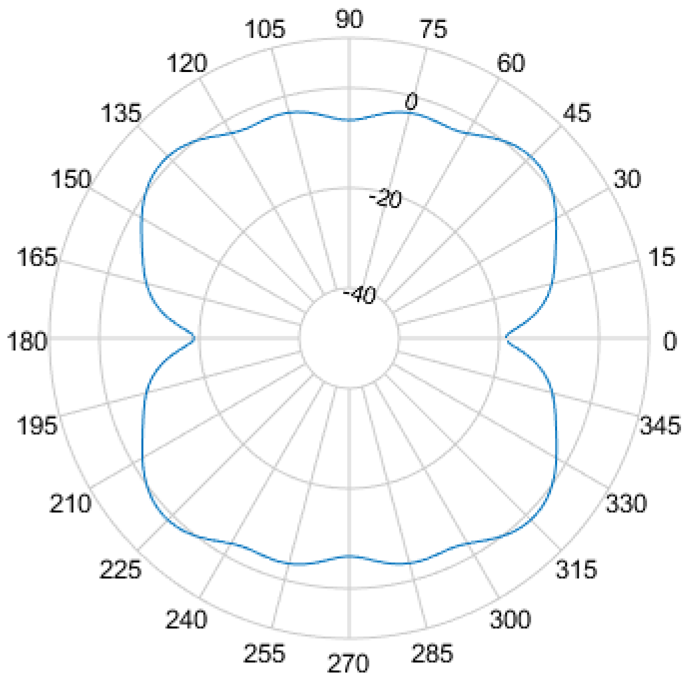
```
[Horizontal,Vertical,optional]= msiread('cavity.pln')
```

```
Horizontal =
```

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


Vertical =

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


optional =

         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` string label to each `array` of the polar plot `p`. The labels are stored as a cell array of strings 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

string

Format for legend label added to the polar plot, specified as a string. For more information on legend label format see, `legend`.

array — Values to apply format string

array

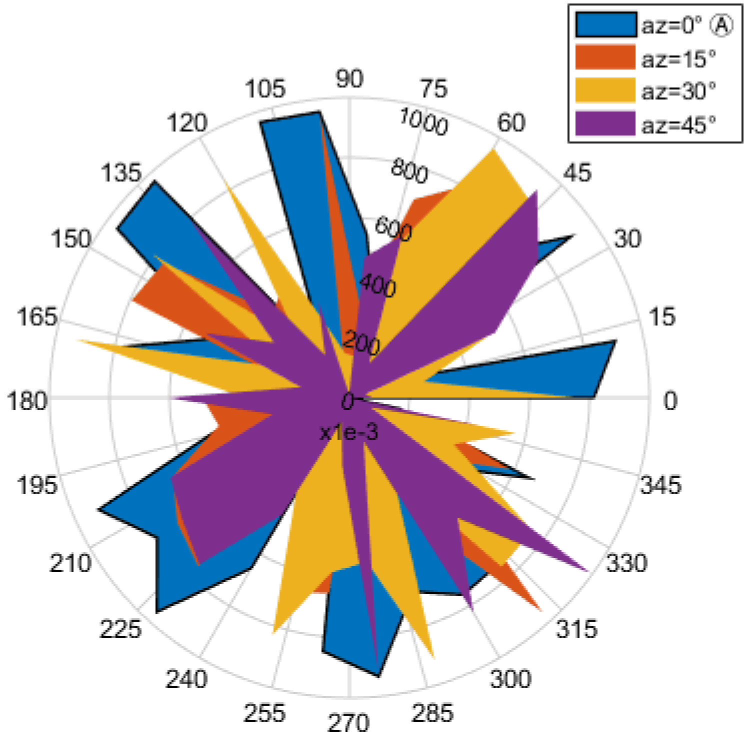
Values to apply the `format` string, specified as an array. The values can be an array of angles or array of magnitude.

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 | 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 in the specified polar plot, `p`.

`L = findLobes(p,index)` returns the antenna 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

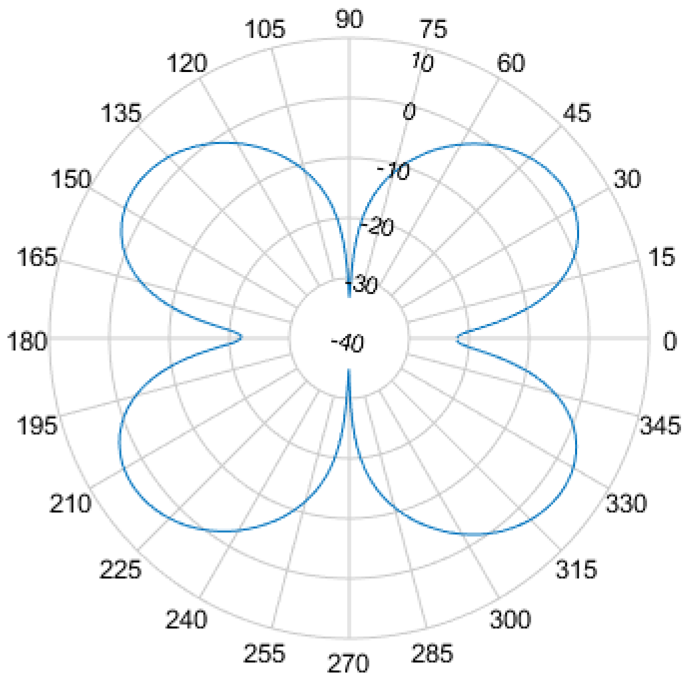
Find Main, Back, and Side Lobes

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 = 

     mainLobe: [1x1 struct]
     backLobe: [1x1 struct]
    sideLobes: [1x1 struct]
           FB: 0.0073
          SLL: 0
         HPBW: 30.1825
         FNBW: 90.7479
        FBIdx: [36 216.5000]
       SLLIdx: [36 146]
      HPBWIdx: [23.4776 52.7440]
      HPBWAng: [21.4181 51.6006]
      FNBWIdx: [361 91]
```



Inspect main, back, and side lobe data.

```
MainLobe = L.mainLobe
BackLobe = L.backLobe
SideLobe = L.sideLobes
```

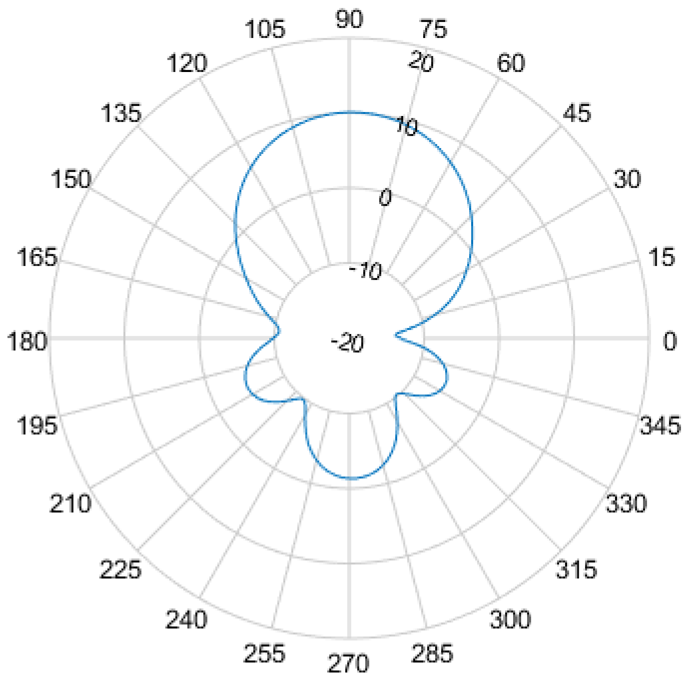
```
MainLobe = 

        index: 36
    magnitude: 3.6587
        angle: 34.9030
       extent: [361 91]
```

```
BackLobe = 

    magnitude: 3.6514
        angle: -145.0970
       extent: [181 271]
        index: 216.5000


SideLobe = 

        index: 146
    magnitude: 3.6587
        angle: 144.5983
       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);
```

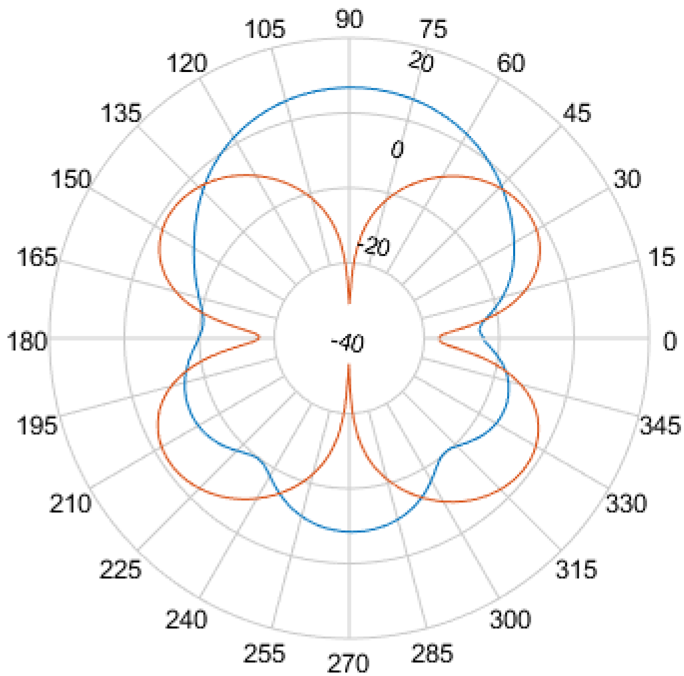


Create a dipole antenna and calculate the directivity at 270 MHz.

```
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 = 

     mainLobe: [1x1 struct]
     backLobe: [1x1 struct]
    sideLobes: [1x1 struct]
           FB: 11.4645
          SLL: 11.4110
         HPBW: 56.1444
         FNBW: 171.5235
```

```
  FBIdx: [90 270.5000]
 SLLIdx: [90 273]
HPBWIdx: [61.4890 117.7893]
HPBWAng: [60.3214 116.4658]
FNBWIdx: [4 176]
```

See Also

add | addCursor | 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 data

real length-M vector | real M-by-N matrix | `N`-by-`D` arrays | complex vector or matrix

Antenna data, specified as one of the following:

- A real length-M vector, where M contains the magnitude values with angles assumed to be $\frac{(0:M-1)}{M \times 360}$ 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 $\frac{(0:M-1)}{M \times 360}$ degrees. The set of each angle can vary for each column.
- `N`-by-`D` arrays, 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 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

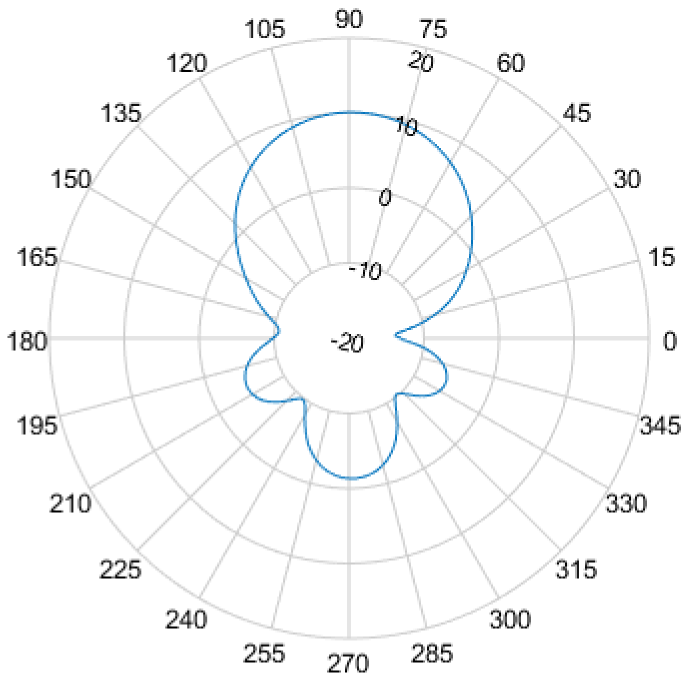
Replace Polar Plot Data with New Data

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);
```

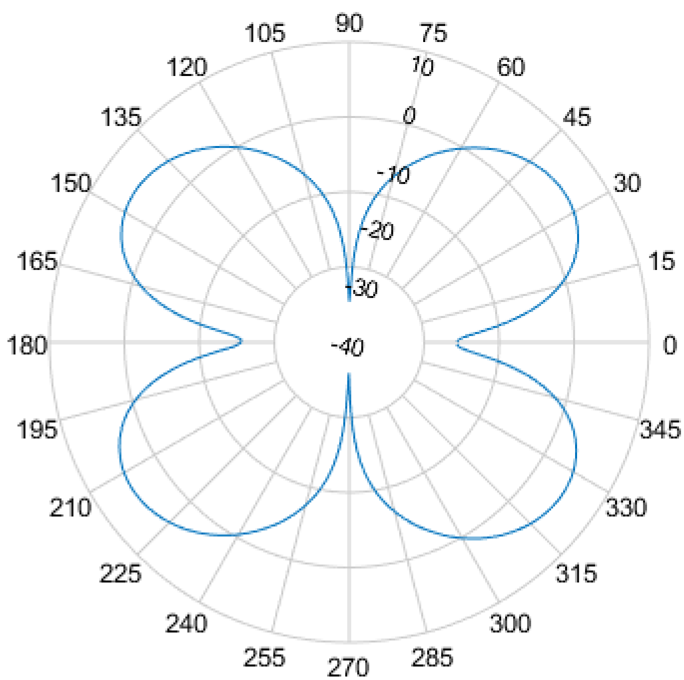


Create a dipole antenna and calculate its directivity at 270 MHz.

```
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

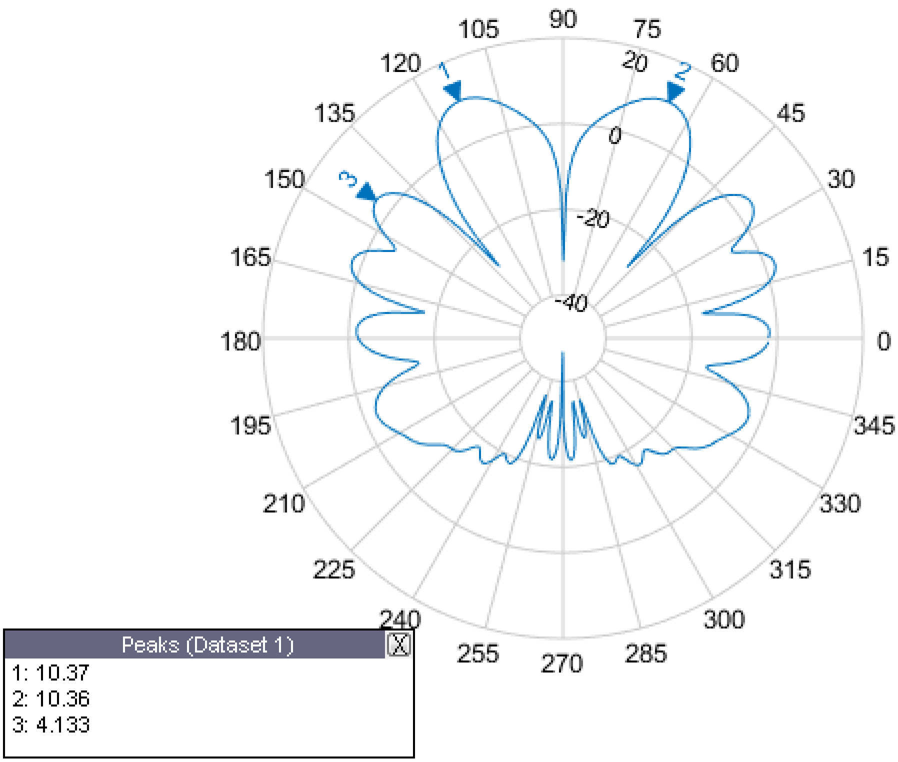
Peaks of Antenna in Polar Pattern

Create a monopole antenna and calculate the directivity at 1 GHz.

```
m = monopole;
M = pattern(m,1e9,0,0:1:360);
```

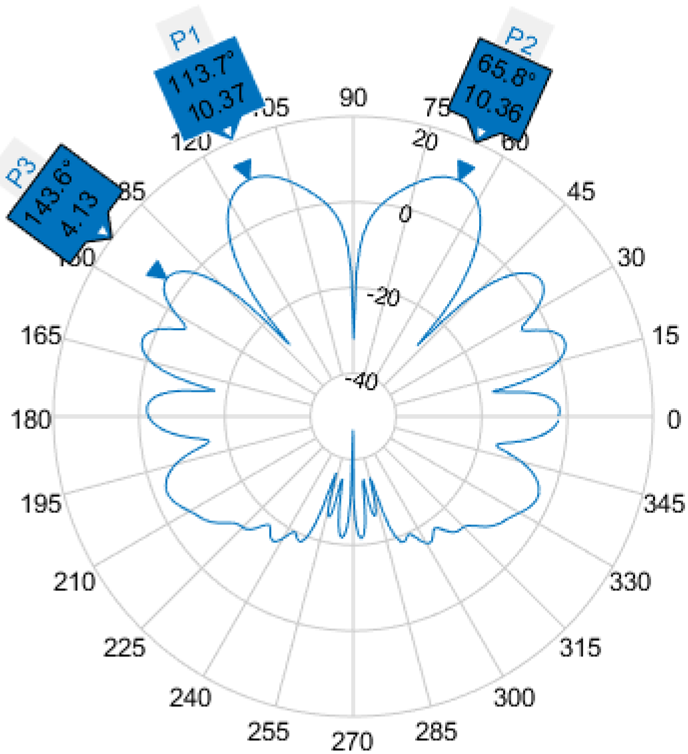
Plot the polar pattern and show three peaks of the antenna. When creating a `polarpattern` plot, if you specify the Peaks property, the peaks table is displayed by default.

```
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 degrees 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
strings

Cursor or peak marker identifiers, specified as a string. Adding cursors to the polar plot creates cursor marker identifiers. Adding peaks to the polar plot creates peak marker identifiers.

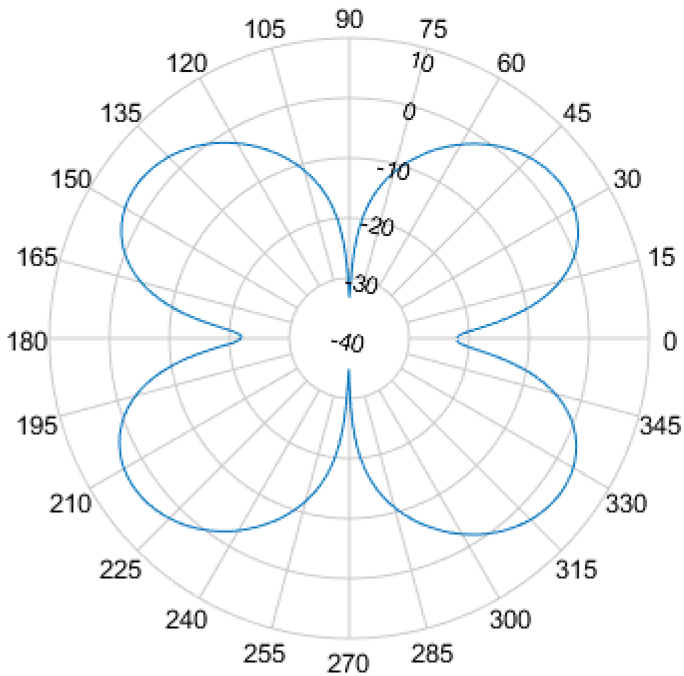
Example: `showspan(p,'C1','C2')`. Displays the angle span between cursors, `C1` and `C2` in polar plot, `p`.

Examples

Show Angle Span

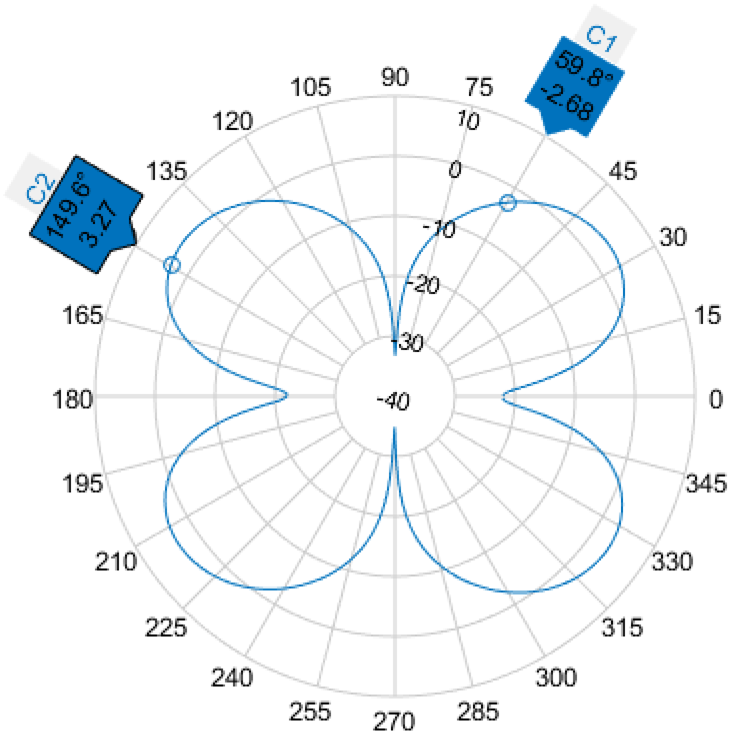
Create a dipole antenna and plot the directivity at 270 MHz.

```
d = dipole;
D = pattern(d,270e6,0,0:1:360);
p = polarpattern(D);
```



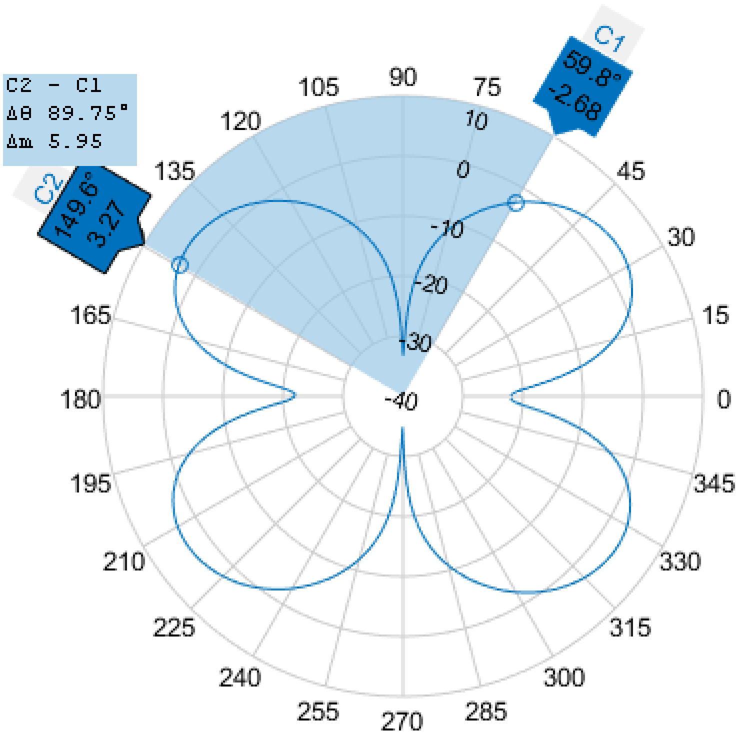
Add cursors to the polar plot at approximately 60 and 150 degrees.

```
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

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”.

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: `string`

`'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: `string`

'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: string

'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: string

'MagnitudeUnits' — Magnitude units
'dB' | 'dBLoss'

Magnitude units, specified as a comma-separated pair consisting of 'MagnitudeUnits' and 'db' or 'dBLoss'.

Data Types: string

'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: `string`

`'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
string | cell array of strings

Data tables for legend annotation, specified as a comma-separated pair consisting of 'LegendLabels' and a string or cell array of strings.

Data Types: string

'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 string to display above the polar plot
string

Title string to display above the polar plot, specified as a comma-separated pair consisting of 'TitleTop' and a string.

Data Types: string

'TitleBottom' — Title string to display below the polar plot
string

Title string to display below the polar plot, specified as a comma-separated pair consisting of 'TitleBottom' and a string.

Data Types: string

'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: string

'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: `string`

'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 strings using a subset of TeX markup
- 'latex' — Interpret strings 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 string.

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 string, except for superscripts and subscripts which only modify the next character or the text within the curly braces {}.

Modifier	Description	Example of String
^{ }	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'

Modifier	Description	Example of String
`\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 string.

The maximum size of the string that you can use with the LaTeX interpreter is 1200 characters. For multiline strings, the maximum size reduces by about 10 characters per line.

For more information about the LaTeX system, see The LaTeX Project website at http://www.latex-project.org/.

Data Types: `string`

`'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 strings using a subset of TeX markup
- `'latex'` — Interpret strings 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 string.

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 string, except for superscripts and subscripts which only modify the next character or the text within the curly braces `{}`.

Modifier	Description	Example of String
`^{ }`	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 string.

The maximum size of the string that you can use with the LaTeX interpreter is 1200 characters. For multiline strings, the maximum size reduces by about 10 characters per line.

For more information about the LaTeX system, see The LaTeX Project website at http://www.latex-project.org/.

Data Types: `string`

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) | color string | 'none'

Color of foreground grid lines, specified as a comma-separated pair consisting of 'GridForeGroundColor' and an RGB triplet, color string, 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]. This table lists the long and short color name options and the equivalent RGB triplet values.

Long Name	Short Name	RGB Triplet
'yellow'	'y'	[1 1 0]
'magenta'	'm'	[1 0 1]
'cyan'	'c'	[0 1 1]
'red'	'r'	[1 0 0]
'green'	'g'	[0 1 0]
'blue'	'b'	[0 0 1]
'white'	'w'	[1 1 1]
'black'	'k'	[0 0 0]

Data Types: double | string

'GridBackGroundColor' — Color of background grid lines

'w' (default) | color string | 'none'

Color of background grid lines, specified as a comma-separated pair consisting of 'GridBackGroundColor' and an RGB triplet, color string, 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]. This table lists the long and short color name options and the equivalent RGB triplet values.

Long Name	Short Name	RGB Triplet
'yellow'	'y'	[1 1 0]
'magenta'	'm'	[1 0 1]
'cyan'	'c'	[0 1 1]
'red'	'r'	[1 0 0]
'green'	'g'	[0 1 0]
'blue'	'b'	[0 0 1]
'white'	'w'	[1 1 1]
'black'	'k'	[0 0 0]

Data Types: double | string

Marker, Color, Line, and Font Properties

'Marker' — Marker symbol

'none' (default) | string

Marker symbol, specified as a comma-separated pair consisting of 'Marker' and either 'none' or one of the strings 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

Value	Description
`'.'`	Point
`'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 color string 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]. This table lists the long and short color name options and the equivalent RGB triplet values.

Long Name	Short Name	RGB Triplet
'yellow'	'y'	[1 1 0]
'magenta'	'm'	[1 0 1]
'cyan'	'c'	[0 1 1]
'red'	'r'	[1 0 0]
'green'	'g'	[0 1 0]
'blue'	'b'	[0 0 1]
'white'	'w'	[1 1 1]
'black'	'k'	[0 0 0]

Data Types: double | string

'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 strings in the table:

String	Line Style	Resulting Line
'-'	Solid line	————————
'--'	Dashed line	- — — — —
':'	Dotted line	....
'-.'	Dash-dotted line	—·—·—·—·-

String	Line Style	Resulting Line
`'none'`	No line	No line

`'LineWidth'` — Line width of plot
`1` (default) | positive scalar

Line width of the plot, specified as a comma-separated pair consisting of `'LineWidth'` and a positive scalar.

`'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: `string`

See Also
"Interact with Polar Plot"