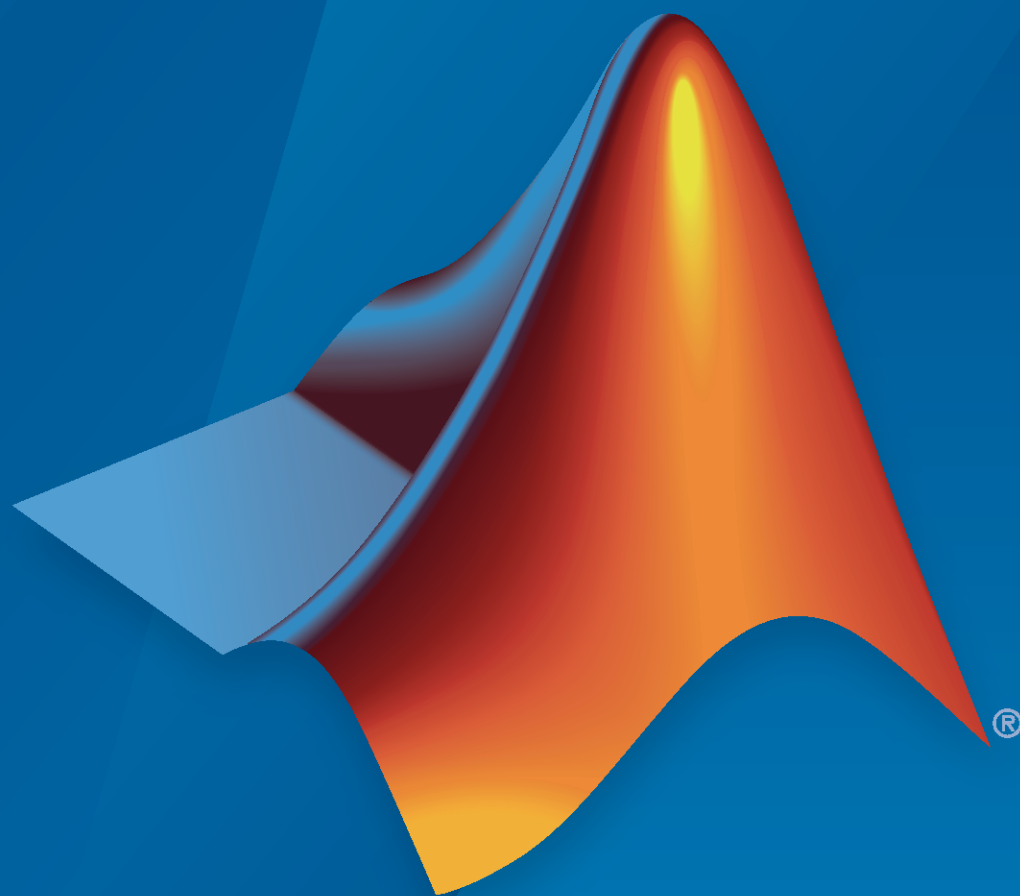


Antenna Toolbox™

User's Guide



MATLAB®

R2023a



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.
1 Apple Hill Drive
Natick, MA 01760-2098

Antenna Toolbox™ User's Guide

© COPYRIGHT 2015–2023 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

Revision History

March 2015	Online only	New for Version 1.0 (R2015a)
September 2015	Online only	Revised for Version 1.1 (R2015b)
March 2016	Online only	Revised for Version 2.0 (R2016a)
September 2016	Online only	Revised for Version 2.1 (R2016b)
March 2017	Online only	Revised for Version 2.2 (R2017a)
September 2017	Online only	Revised for Version 3.0 (R2017b)
March 2018	Online only	Revised for Version 3.1 (R2018a)
September 2018	Online only	Revised for Version 3.2 (R2018b)
March 2019	Online only	Revised for Version 4.0 (R2019a)
September 2019	Online only	Revised for Version 4.1 (R2019b)
March 2020	Online only	Revised for Version 4.2 (R2020a)
September 2020	Online only	Revised for Version 4.3 (R2020b)
March 2021	Online only	Revised for Version 5.0 (R2021a)
September 2021	Online only	Revised for Version 5.1 (R2021b)
March 2022	Online only	Revised for Version 5.2 (R2022a)
September 2022	Online only	Revised for Version 5.3 (R2022b)
March 2023	Online only	Revised for Version 5.4 (R2023a)

Port Analysis	1-2
Input Impedance	1-2
Resonance	1-3
Reflection Coefficient	1-4
Return Loss	1-5
Voltage Standing Wave Ratio	1-6
Bandwidth	1-7
Surface Analysis	1-8
Charge Distribution	1-8
Current Distribution	1-11
Meshing	1-15
Automatic Meshing Mode	1-15
Manual Meshing Mode	1-16
Strip Meshing	1-17
Surface Meshing	1-17
Meshing a Dielectric Substrate	1-18
Field Analysis	1-20
Radiation Pattern	1-20
Beamwidth	1-23
E-Plane and H-Plane	1-24
Polarization	1-26
Axial Ratio	1-30
Rotate Antennas and Arrays	1-31
Rotate Single Antenna Element	1-31
Rotate Single Antenna Element About Multiple Axes	1-33
Rotate Dipole-Backed Reflector	1-34
Rotate Antenna Array	1-37
Infinite Ground Plane	1-39
Image Theory	1-39
Formulate the Image Theory Technique	1-40
Model Infinite Ground Plane for Balanced Antennas	1-43
Model Infinite Ground Plane for Unbalanced Antennas	1-48
Electrical Length of Antenna	1-53
Elevation Patterns of Dipole Antennas	1-53
Elevation Patterns of Rectangular Waveguides at Different Frequencies	1-55

Effect of Materials and Reactance on Electrical Length	1-56
Mesh at Various Frequencies	1-59
Far-field Terminologies	1-61
Directivity	1-61
Gain	1-62
Realized Gain	1-63
Field Calculation in Antennas	1-65
Field Calculation in Metal Antennas	1-65
Field Calculation in Metal-Dielectric Antenna	1-66
Field Parameters	1-66

Array Concepts

2

Mutual Coupling	2-2
Active or Scan Impedance	2-2
Mutual Impedance	2-3
Coupling Matrix	2-4
Array Factor and Pattern Multiplication	2-5
Isolated Element Pattern	2-10
Embedded Element Pattern	2-11
Beamforming	2-13
Side Lobe Control	2-13
Beam Scanning	2-14
Grating Lobes	2-17
Correlation Coefficient	2-20
Far-Field Radiation Pattern	2-20
S-Parameter Characterization	2-20
Infinite Arrays	2-22
What Are Infinite Arrays?	2-22
Infinite Array Analysis	2-22
Create Infinite Array Using Antenna Toolbox	2-23
Choose a Unit Cell	2-24
Scan Infinite Arrays	2-25
Scan Impedance and Scan Element Pattern	2-26
Compare Scan Element Pattern of Finite and Infinite Arrays	2-26
Impact of Infinite Double Summation	2-32
Manipulate Array Elements	2-35

Method of Moments Solver for Metal Structures	3-2
MoM Formulation	3-2
Neighbor Region	3-5
Singularity Extraction	3-6
Finite Arrays	3-7
Infinite Array	3-7
Method of Moments Solver for Metal and Dielectric Structures	3-9
MoM Formulation	3-9
Neighbor Region	3-11
Singularity Extraction	3-12
Finite Arrays	3-13
Infinite Array	3-13
Hybrid MoM-PO Method for Metal Antennas with Large Scatterers . . .	3-15
Subdomain RWG Basis Functions and Extra Dimensions	3-15
MoM Region and PO Region	3-16
MoM Solution and PO Solution	3-16
Finding ZPO	3-17
Direct Solution Method	3-18
Physical Optics Solver	3-20
Subdomain RWG Basis Functions and Extra Dimensions	3-20
Finding IPO	3-21
Determination of Illuminated or Shadow Regions	3-22
Feed Model	3-23
Delta-Gap Source Feed Model	3-23
Double-Edge Delta Source Feed Models	3-24
Multi-Edge Delta Source Feed Models	3-26
Antenna Optimization Algorithm	3-29
Algorithm Outline	3-29
Wire Solver	3-31
Fast Multipole Method for Large Structures	3-33
Direct Solvers	3-33
Relation Between Memory Used and Problem Size	3-34
Fast Multipole Method (FMM)	3-35
Finite Conductivity and Thickness Effect in MoM Solver	3-38
MoM Formulation	3-39
Solvers	3-41

4

Troubleshooting Site Viewer	4-2
Internet Connection Failure	4-2
Graphics Environment	4-2
Access Basemaps and Terrain in Site Viewer	4-3
Use Installed Basemap	4-3
Download Basemaps	4-3
Add Custom Basemaps	4-3
Access Terrain	4-4
Access TIREM Software	4-5
Choose a Propagation Model	4-6
Atmospheric	4-6
Empirical	4-6
Terrain	4-7
Ray Tracing	4-7
Ray Tracing for Wireless Communications	4-12
Introduction	4-12
Ray Tracing Methods	4-13
Propagation Loss	4-15

Antenna Toolbox Examples

5

Port Analysis of Antenna	5-4
Current Visualization on Antenna Surface	5-10
Antenna Far-Field Visualization	5-16
Antenna Near-Field Visualization	5-27
Wave Impedance	5-32
Antenna Array Analysis	5-36
Modeling Infinite Ground Plane in Antennas and Arrays	5-48
Infinite Array Analysis	5-58
Parallelization of Antenna and Array Analyses	5-68
Analysis of Monopole Impedance	5-72
Analysis of Dipole Impedance	5-77

Monopole Measurement Comparison	5-80
Equiangular Spiral Antenna Design Investigation	5-85
Archimedean Spiral Design Investigation	5-88
Patch Antenna on Dielectric Substrate	5-95
Sector Antenna for 2.4 GHz Wi-Fi™	5-107
Design PIFA for WLAN Wi-Fi™ Applications	5-117
Helical Antenna Design	5-124
Reflector Backed Equiangular Spiral	5-129
Impedance Matching of Non-resonant (Small) Monopole	5-141
Direct Search Based Optimization of Six-Element Yagi-Uda Antenna .	5-147
Antenna Diversity Analysis for 800 MHz MIMO	5-159
Polarization Analysis for X-band Microstrip Patch Antenna	5-166
FMCW Patch Antenna Array	5-172
Modeling Mutual Coupling in Large Arrays Using Infinite Array Analysis	5-184
Modeling Mutual Coupling in Large Arrays Using Embedded Element Pattern	5-195
Modeling Resonant Coupled Wireless Power Transfer System	5-209
Crossed-Dipole (Turnstile) Antenna and Array	5-218
Visualize Antenna Field Strength Map on Earth	5-230
RFID Antenna Design	5-236
Wideband Blade Dipole Antenna and Array	5-243
Analysis of Inset-Feed Patch Antenna on Dielectric Substrate	5-249
Read, Visualize and Write MSI Planet Antenna Files	5-256
Custom Radiation Pattern and Fields	5-263
Double-Slot Cavity Patch on TMM10 Substrate	5-272
Effect of Mutual Coupling on MIMO Communication	5-281
Switched Beam Array with Butler Matrix	5-289

Antenna Model Generation and Full-Wave Analysis From A Photo	5-294
Visualize Antenna Coverage Map and Communication Links	5-309
Modeling and Analysis of Single Layer Multi-band U-Slot Patch Antenna	5-317
Antenna Array Beam Scanning Visualization on a Map	5-335
Modeling Planar Photonic Band Gap Structure	5-344
Plane Wave Excitation - Scattering Solution	5-353
Design, Analysis, and Prototyping of Microstrip-Fed Wide-Slot Antenna	5-356
Comparison of Antenna Array Transmit and Receive Manifold	5-364
Verification of Far-Field Array Pattern Using Superposition with Embedded Element Patterns	5-372
Metasurface Antenna Modeling	5-381
Modeling and Analysis of Probe-Fed Stacked Patch Antenna	5-395
Design Internally Matched Ultra-Wideband Vivaldi Antenna	5-410
Planning a 5G Fixed Wireless Access Link over Terrain	5-432
SINR Map for a 5G Urban Macro-Cell Test Environment	5-447
Surrogate Based Optimization Design of Six-Element Yagi-Uda Antenna	5-459
Model and Analyze Dual Polarized Patch Microstrip Antenna	5-476
Installed Antenna Analysis - Modelling Antennas with Platforms	5-486
3D Reconstruction of Radiation Pattern From 2D Orthogonal Slices . .	5-497
Loading Using Lumped Elements	5-510
Planning Radar Network Coverage over Terrain	5-521
Visualize Viewsheds and Coverage Maps Using Terrain	5-532
Subarrays in Large Finite Array For Hybrid Beamforming	5-562
Pattern Analysis of the Symmetric Parabolic Reflector	5-570
Radar Cross Section Benchmarking	5-579
Design and Analyze Cassegrain Antenna	5-585

Discone Antenna for TV Broadcasting System	5-595
Analysis of Biquad Yagi for Wi-Fi Applications	5-600
Analysis of Broad-wall Slotted Array Waveguide for High Frequency Applications	5-606
Urban Link and Coverage Analysis Using Ray Tracing	5-612
Maximizing Gain and Improving Impedance Bandwidth of E-Patch Antenna	5-625
Optimization of Antenna Array Elements Using Antenna Array Designer App	5-636
Import and Analyze Custom 3-D Antenna Geometry	5-647
Model, Analyze and Compare Horn Antennas	5-673
Multiband Nature and Miniaturization of Fractal Antennas	5-683
ISM Band Patch Microstrip Antennas and Mutually Coupled Patches	5-695
Design and Analyze Curved Reflectors	5-705
VHF/UHF Biconical Antenna for Testing Applications	5-714
Ultra-Wideband (UWB) Planar Monopole Antennas	5-721
Maximize Impedance Bandwidth of Triangular Patch Antenna	5-736
Create Antenna Model from Gerber Files	5-745
Design Log-Periodic Sawtooth Planar Antenna for UHF Ultra-Wideband Applications	5-750
Calculate Radiation Efficiency of Antenna	5-759
Design 77 GHz Patch Microstrip for Automotive Radar Receiver	5-774
Infinite Array of Microstrip Patch Antenna on Teflon Substrate	5-780
Modeling Wire Antenna and Arrays	5-786
Analysis of Electrically Large Structures Using Hybrid MoM and FMM	5-793
Analyze Cylindrical Reflector Antenna with Horn Array Feed	5-802
Design and Analysis Using PCB Antenna Designer	5-811
Define PCB Antenna Layers	5-813
Design H-Notch Patch Unit Element	5-818

Analyze H-Notch Unit Element	5-831
Design, Analyze, and Export 1-by-2 H-Notch Linear Array	5-836
Design, Analyze, and Optimize H-Notch Patch Using Design Variables	5-852
Design Series-Fed Patch Antenna Array for 5G Base Station	5-864
PCB Antenna for USB Dongle and BLE Applications	5-874
Miniaturize Patch Antennas Using Metamaterial-Inspired Technique	5-883
Design and Analysis of Compact Ultra-Wideband MIMO Antenna Array	5-893
Direction of Arrival Determination Using Full-Wave Electromagnetic Analysis	5-903
Field Analysis of Monopole Antenna	5-909
Design and Analysis of a Diamond-Shaped Antenna for Ultra-Wideband Applications	5-920
Board Thickness versus Dielectric Thickness in PCB	5-927
Analyze Metal Conductors in Helical Dipole Antenna	5-936
Design, Analyze, and Prototype 2x2 Patch Array Antenna	5-959
Modified Sierpinski Monopole Fractal Antenna for Dual-Band Application	5-983
Design and Analyze Parabolic-Reflector-Backed Wideband Eggcrate Array	5-990
Design and Analyze Tapered-Slot SIW Filtenna	5-1000
Antennas on A Glider as Platform	5-1014
Electromagnetic Analysis of Intelligent Reflecting Surface	5-1017
Design S-Band Monopulse Tracking RADAR Antenna	5-1025

Antenna Concepts

- “Port Analysis” on page 1-2
- “Surface Analysis” on page 1-8
- “Meshing” on page 1-15
- “Field Analysis” on page 1-20
- “Rotate Antennas and Arrays” on page 1-31
- “Infinite Ground Plane” on page 1-39
- “Model Infinite Ground Plane for Balanced Antennas” on page 1-43
- “Model Infinite Ground Plane for Unbalanced Antennas” on page 1-48
- “Electrical Length of Antenna” on page 1-53
- “Far-field Terminologies” on page 1-61
- “Field Calculation in Antennas” on page 1-65

Port Analysis

In this section...

“Input Impedance” on page 1-2

“Resonance” on page 1-3

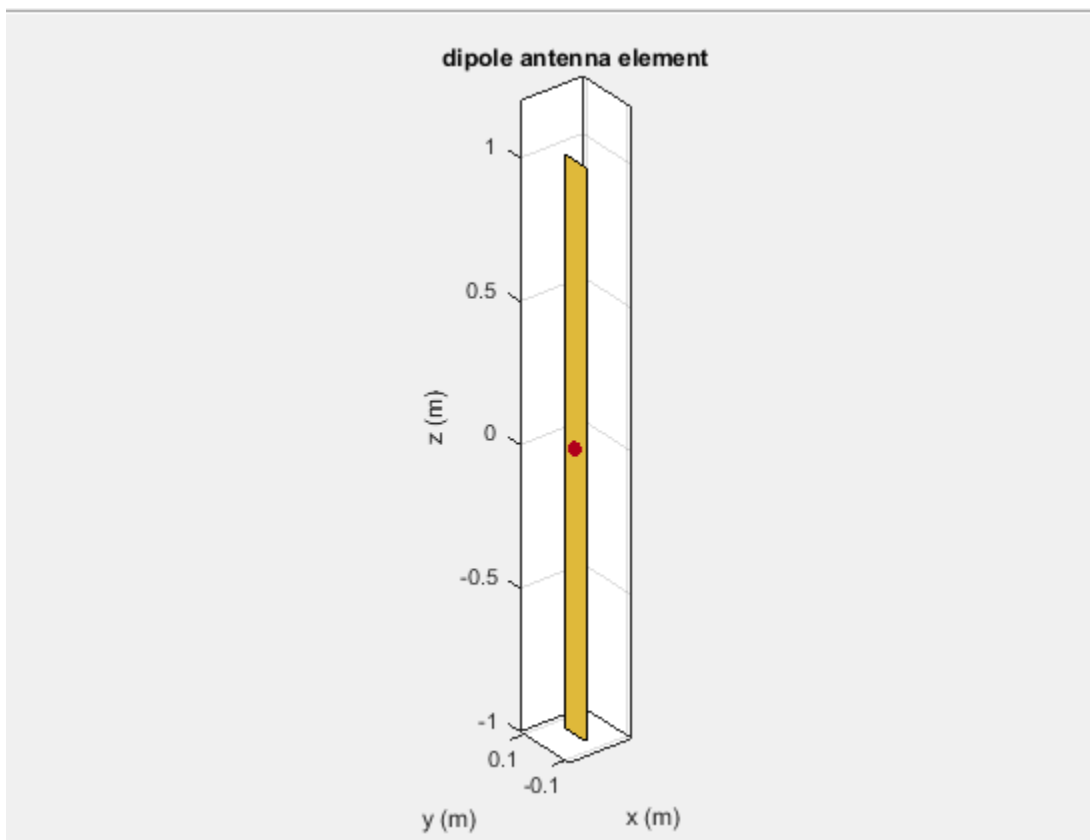
“Reflection Coefficient” on page 1-4

“Return Loss” on page 1-5

“Voltage Standing Wave Ratio” on page 1-6

“Bandwidth” on page 1-7

The *port* of an antenna is the physical location where the RF source is connected. From a network theory perspective, the antenna has a single port. In Antenna Toolbox, a red dot on the antenna figure represents the feed point. A half-wavelength dipole is shown with its feed point:



All antennas are excited by a voltage of 1V at the port. The various terminal port parameters are as follows:

Input Impedance

Input impedance is the ratio of voltage to current at the port. Antenna impedance is calculated as the ratio of the phasor voltage, which is 1V at a phase angle of θ deg, to the phasor current at the port. The impedance equation is:

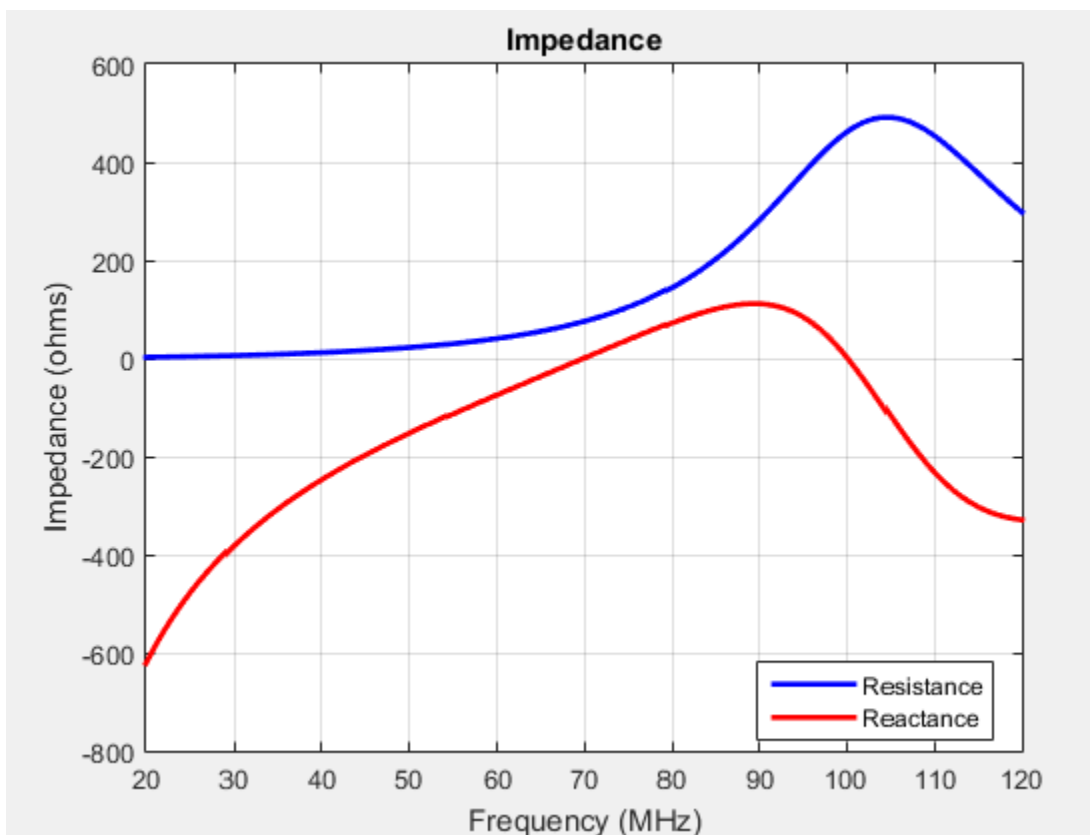
$$Z = V/I = R + jX$$

where:

- V is the antenna excitation voltage
- I is the current
- R is the antenna resistance in ohms
- X is the antenna reactance in ohms

Antenna input impedance is a frequency-dependent quantity. The plot shows the input impedance of a dipole antenna over the frequency band 20–120 MHz. The resistance and reactance traces vary with frequency. The variation can be qualitatively described in terms of resonances.

```
d = dipole;
impedance(d, 20e6:1e6:120e6)
```

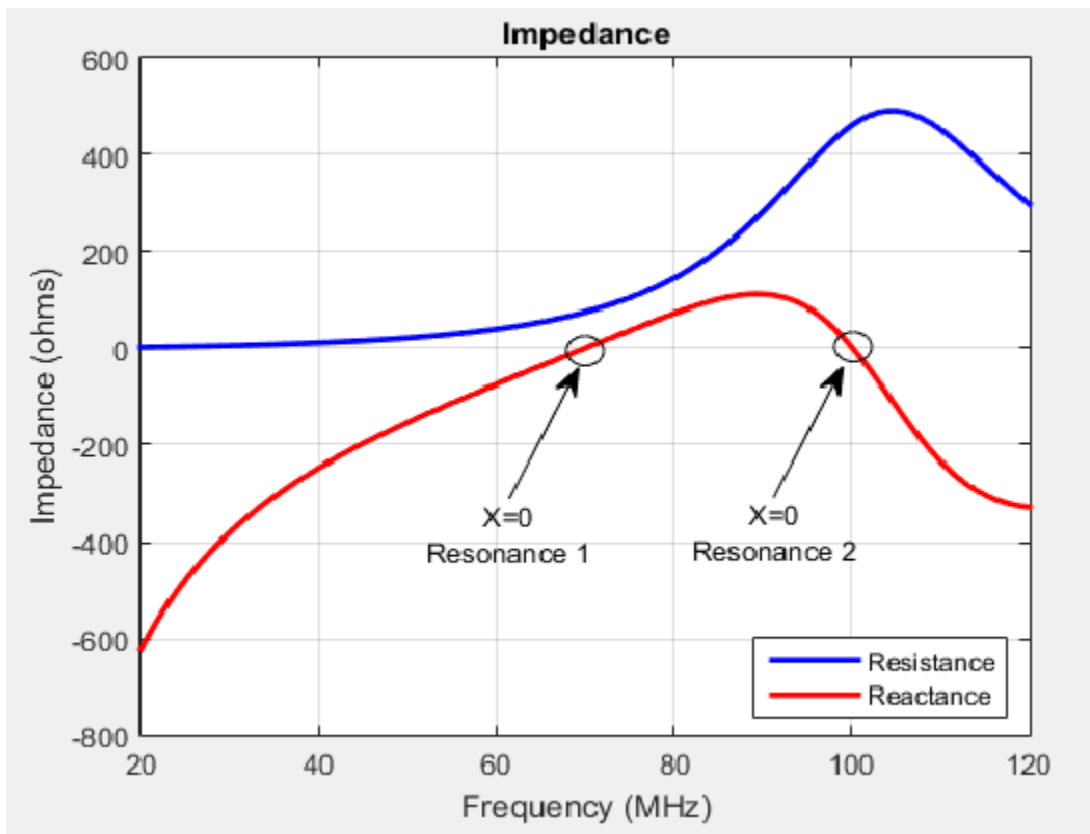


Use `impedance` to calculate the input impedance of any antennas in Antenna Toolbox.

Resonance

The *resonant frequency* of the antenna is the frequency at which the reactance of the antenna is equal to zero.

The plot shows two resonance points of a dipole antenna.

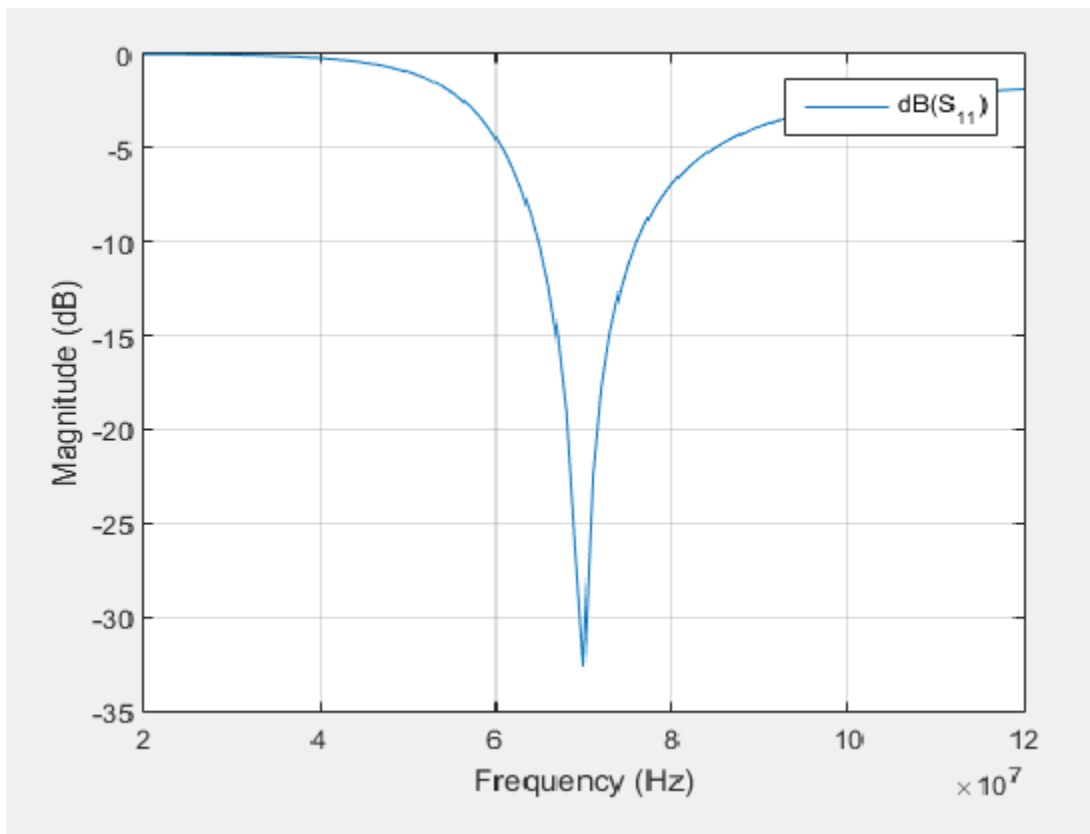


In the plot, the reactance values are negative, or capacitive, before the resonance. These values are positive or inductive after the resonance. This type of resonance is called series resonance. You can model this type of resonance using a series RLC circuit. If the impedance curve goes from positive reactance to negative reactance, it is called parallel resonance. You can model this type of resonance using a parallel RLC circuit.

Reflection Coefficient

The *reflection coefficient*, or S_{11} , of the antenna describes a relative fraction of the incident RF power that is reflected back due to the impedance mismatch. Impedance mismatch is the difference between the input impedance of the antenna and the characteristic impedance of the transmission line (or the generator impedance when the transmission line is not present). The characteristic impedance is the reference impedance.

```
S = sparameters(d,20e6:1e6:120e6,72)
rfplot(S)
```



The reflection coefficient also gives the operating bandwidth of the antenna. Antenna bandwidth is usually the frequency band over which the magnitude of the reflection coefficient is below -10 dB.

Use `sparameters` to calculate the value of S_{11} for any antenna in the Antenna Toolbox.

Return Loss

The *return loss* of an antenna is a measure of the effectiveness of power delivery from a transmission line or coaxial cable to a load such as an antenna. The return loss can also be defined as the difference in dB between the power sent toward the antenna and the power reflected back from it. The higher the power ratio, the better matching between load and line. Return loss equation is:

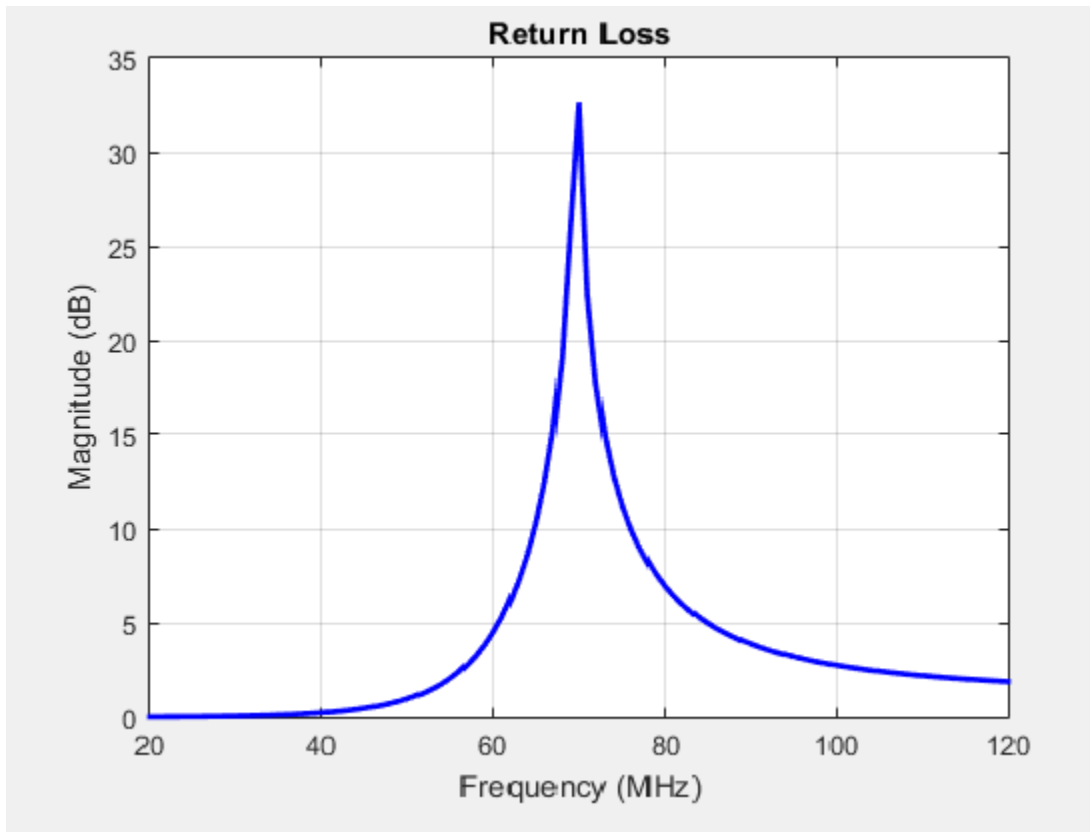
$$RL = -20\log_{10}|S_{11}|$$

where:

- RL is the return loss
- S_{11} is the reflection coefficient, or power reflected from the antenna.

For passive devices, the return loss is a positive nondissipative term representing the reduction in amplitude of the reflected wave in comparison to the incident wave. In active devices, a negative return loss is possible.

```
d = dipole;
returnLoss(d,20e6:1e6:120e6,72)
```



Return loss plots also give the operating bandwidth of the antenna. Antenna bandwidth is the frequency band over which the magnitude of return loss is greater than 10 dB. Use the `returnLoss` function to calculate the return loss of any antenna in the Antenna Toolbox library.

Voltage Standing Wave Ratio

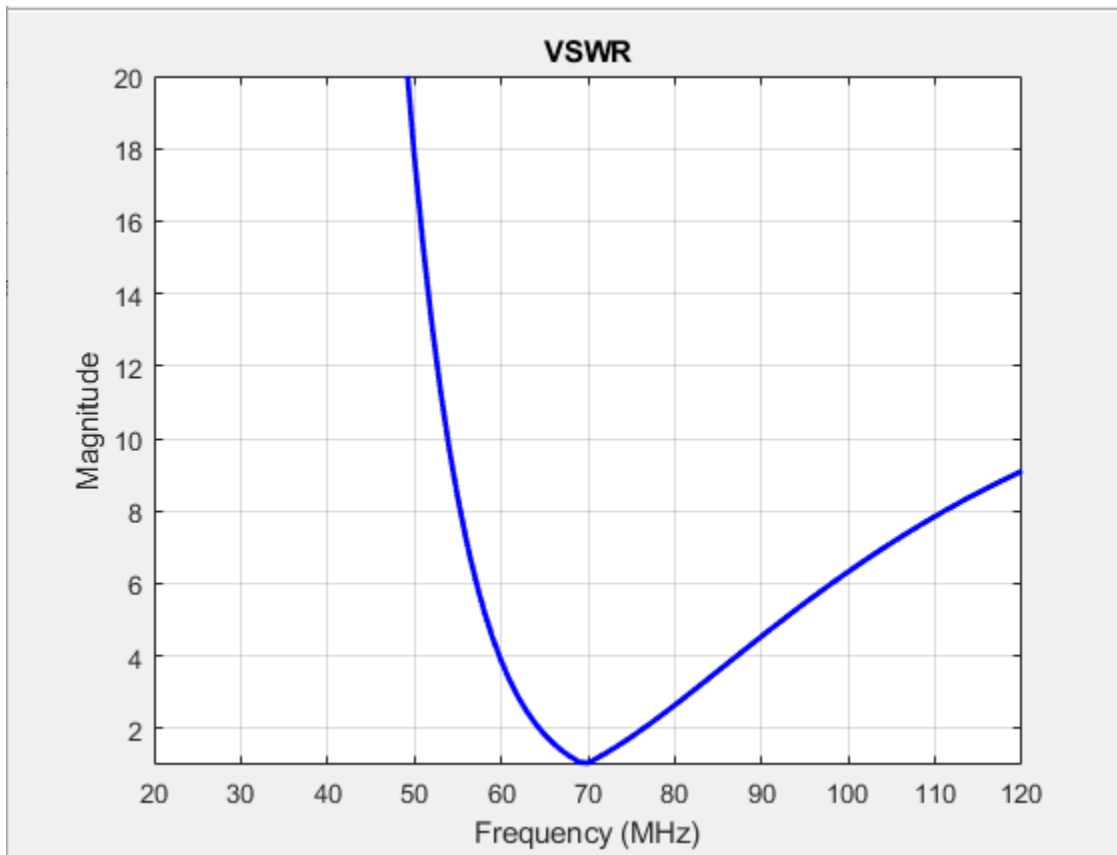
The *voltage standing wave ratio* (VSWR) of an antenna is another measure of impedance matching between transmission line and antenna. The standing wave is generated because of the impedance mismatch at the port. VSWR equation is:

$$VSWR = \frac{1 + |S_{11}|}{1 - |S_{11}|}$$

where:

- S_{11} is the reflection coefficient.

```
d = dipole;
vswr(d,20e6:1e6:120e6,72)
axis([20 120 1 20])
```

VSWR is scalar and contains no phase information. The value of VSWR lies between 1 and infinity. Antenna bandwidth is usually the frequency band over which the VSWR is less than approximately 2.

Use `vswr` to calculate the voltage standing wave ratio for any antenna in Antenna Toolbox.

Bandwidth

Bandwidth describes the range of frequencies over which the antenna can properly radiate or receive energy. It is a fundamental antenna parameter. Often, the desired bandwidth is one of the parameters used to determine which antenna to use. Antenna bandwidth is usually the frequency band over which the magnitude of the reflection coefficient is below -10 dB, or the magnitude of the return loss is greater than 10 dB, or the VSWR is less than approximately 2. All these criteria are equivalent. You can control the bandwidth using proper antenna design.

References

- [1] Balanis, C.A. *Antenna Theory: Analysis and Design*. 3rd Ed. New York: Wiley, 2005.
- [2] Stutzman, Warren L., and Thiele, Gary A. *Antenna Theory and Design*. 3rd Ed. New York: Wiley, 2013.
- [3] Bird, T.S. "Definition and Misuse of Return Loss." *IEEE Antennas and Propagation Magazine*. Vol. 51, Issue 2, April 2009, pp. 166-167.

Surface Analysis

In this section...

"Charge Distribution" on page 1-8

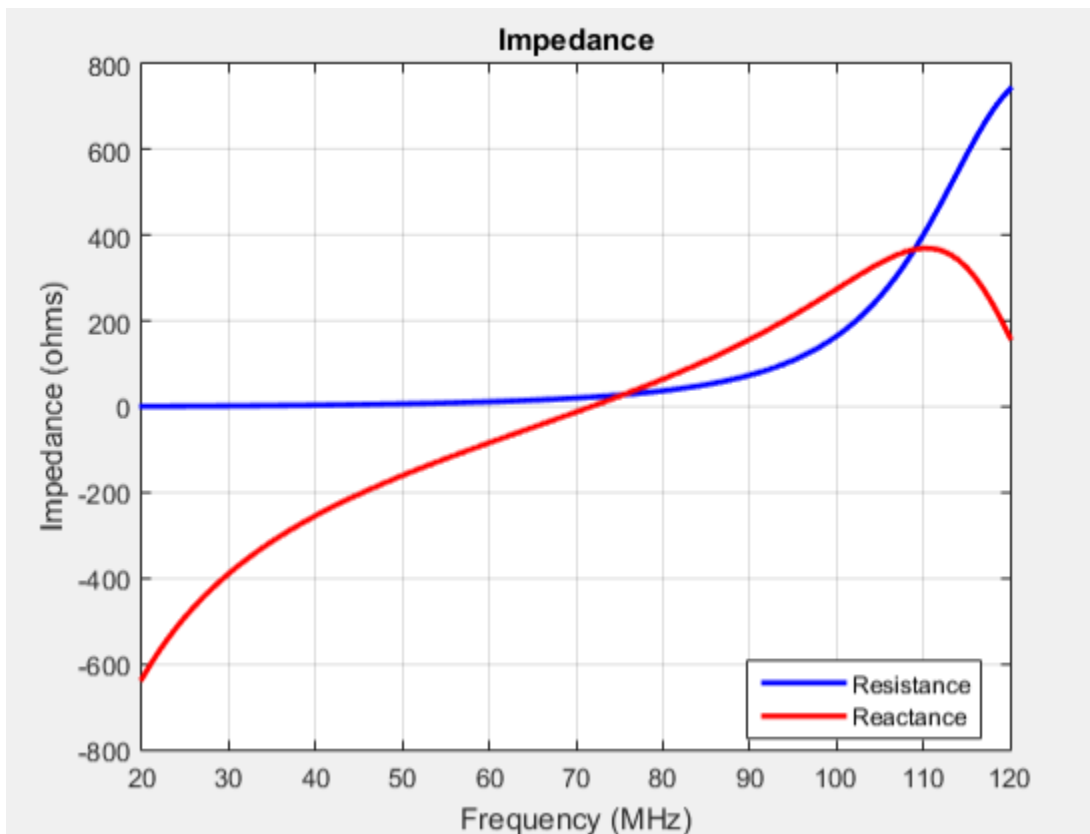
"Current Distribution" on page 1-11

Charge Distribution

The flow of charges on the antenna surface determines the surface currents of the antenna. For antennas to radiate, there must be acceleration or deceleration of charges. The deceleration of charges is caused due to buildup of charges at the end of the wire, which leads to impedance discontinuities. This mechanism creates electromagnetic radiation. The accumulation of charges varies according to time and structure of the antenna.

The accumulation of charges is exploited in many ways. If you calculate the impedance of this monopole antenna using the impedance function, you get the following plot:

```
m = monopole
impedance(m, 20e6:1e6:120e6)
```

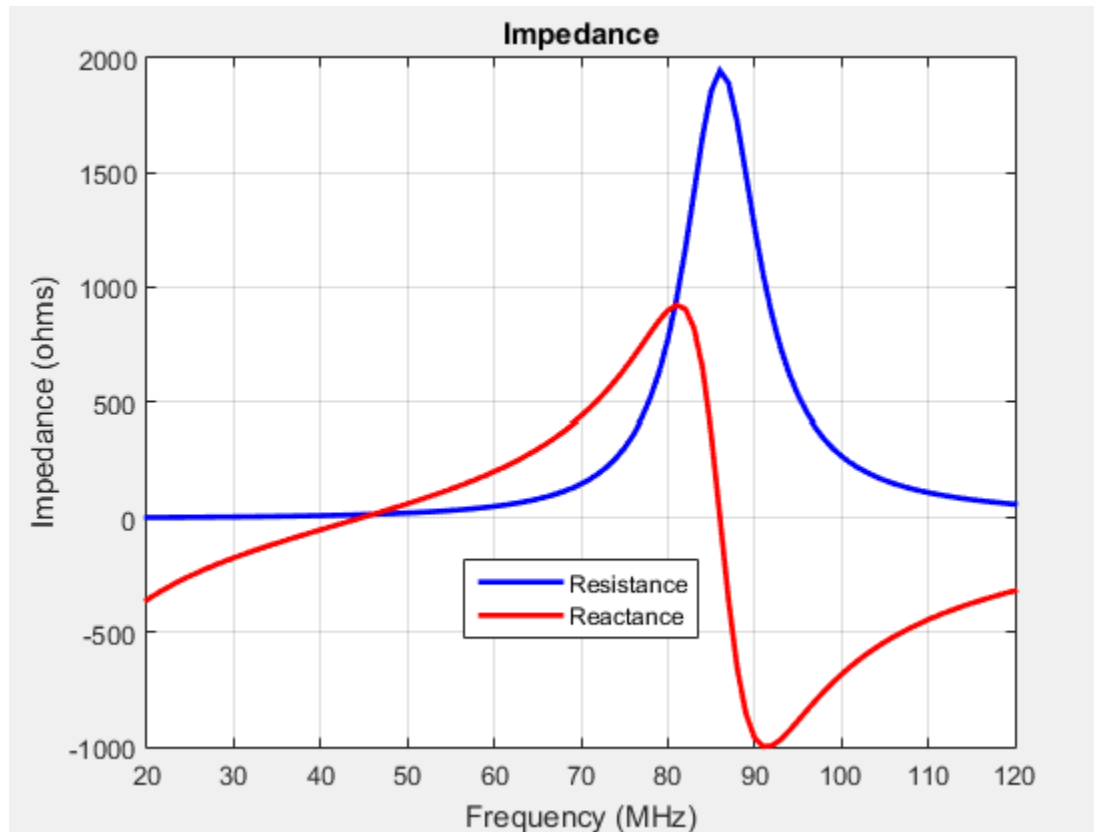


You can observe the first resonance is at approximately 71 MHz. To lower the resonance frequency, recalculate the height of the monopole to quarter wavelength. The frequency of operation is also lower. You must also have to increase the size of the corresponding ground plane. This increase in

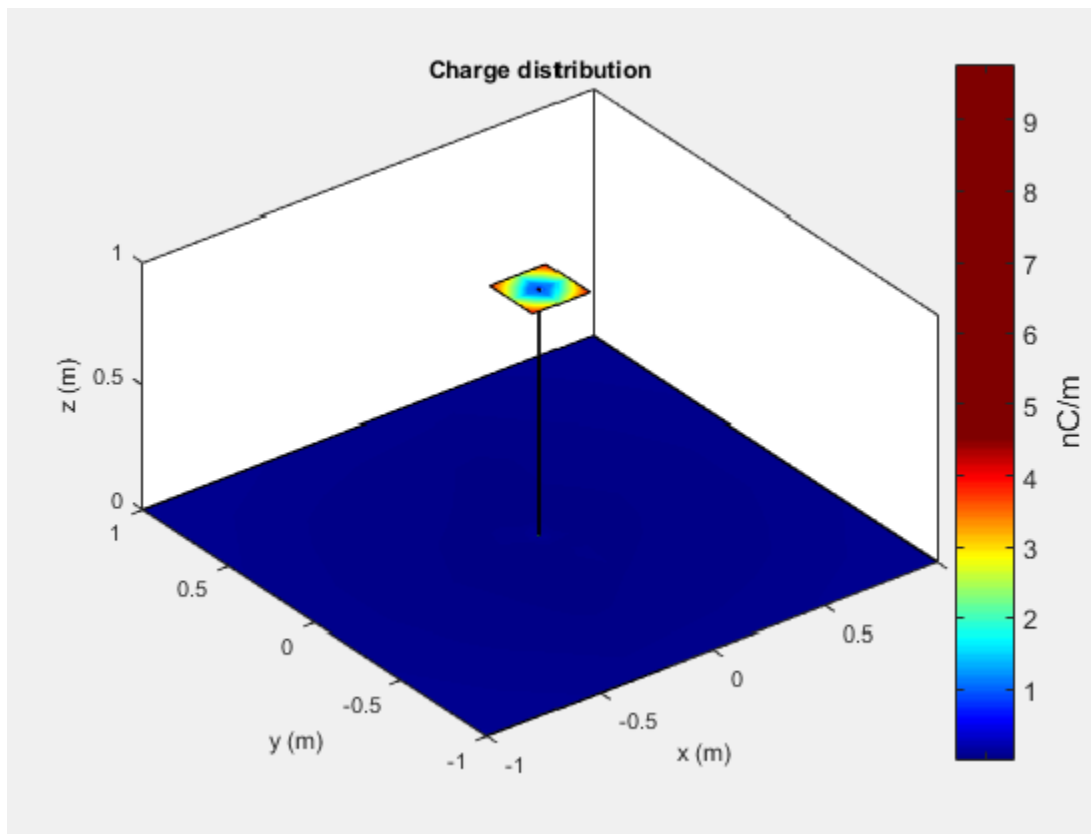
size means that to achieve similar performance at a lower frequency, you need a larger antenna. This approach is not possible due to physical space constraints.

Alternatively, you can exploit the fact that antennas have charge accumulation. If you provide appropriate structural modification to the antenna, charges accumulate. For a monopole antenna, you can enable charge accumulation by adding a top-hat to the monopole. Now, if you calculate the impedance of the antenna using the top-hat, the plot is:

```
mt = monopoleTopHat  
impedance(mt, 20e6:1e6:120e6)
```



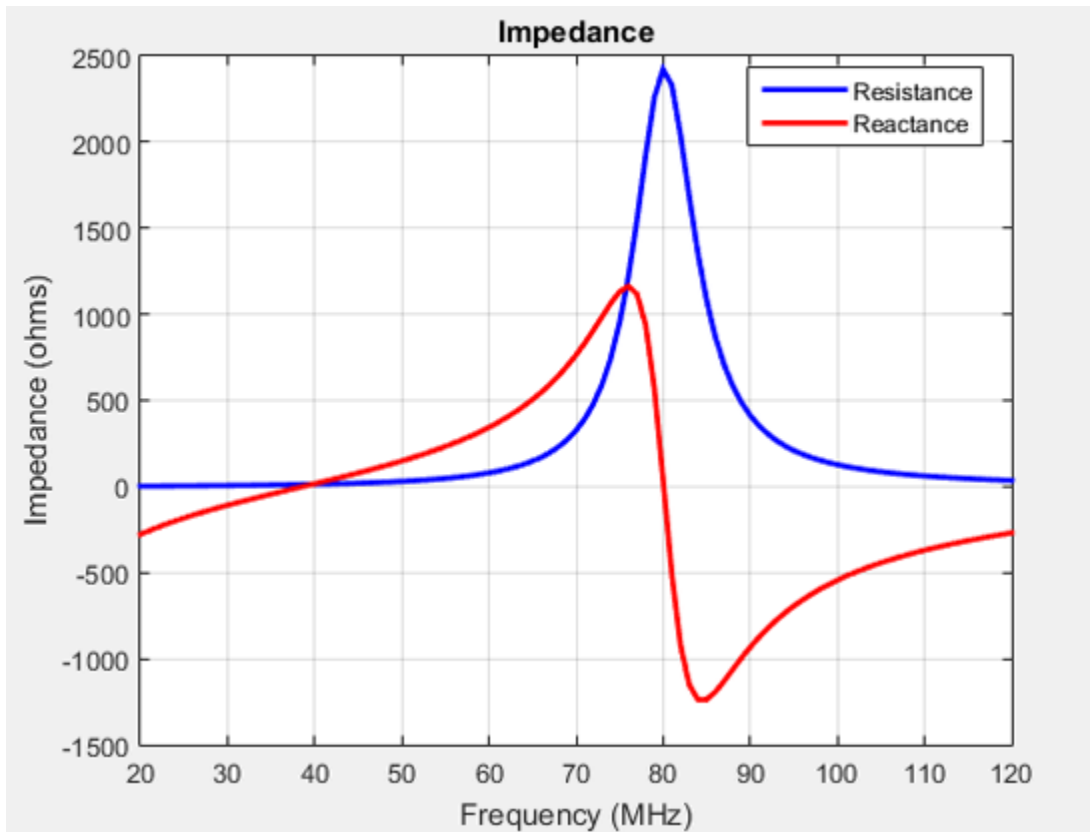
All the dimensions of this antenna are same as the monopole. The first resonance of the antenna is approximately 45 MHz. To view the accumulated charges on the top-hat monopole, use the charge function:



The increase in capacitance lowers the frequency of the antenna. By keeping the physical volume of the antenna same, the resonance point is shifted.

Increasing the top-hat dimensions provides more surface area for charges to accumulate. More charge accumulation increases the capacitance and pushes the resonant frequency lower. For example:

```
mt.TopHatLength = 0.35  
mt.TopHatWidth = 0.35  
impedance(mt, 20e6:1e6:120e6)
```

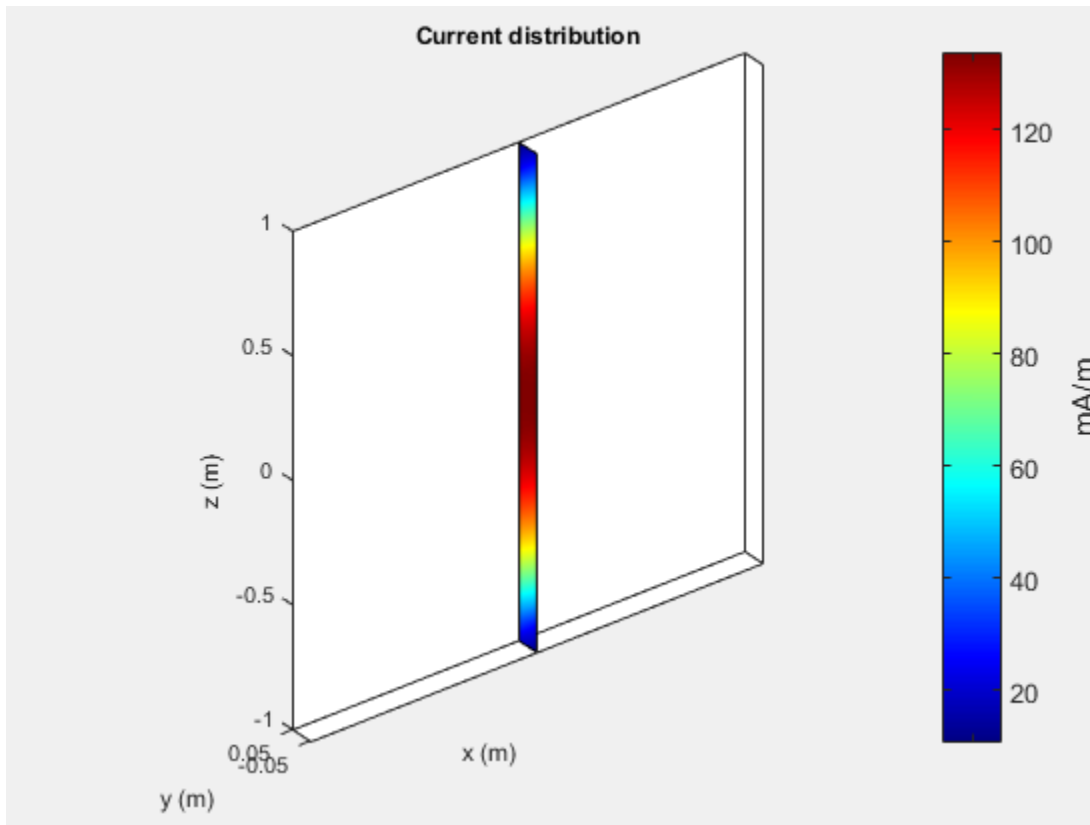


The resonance of the antenna further reduces to approximately 40 MHz.

Current Distribution

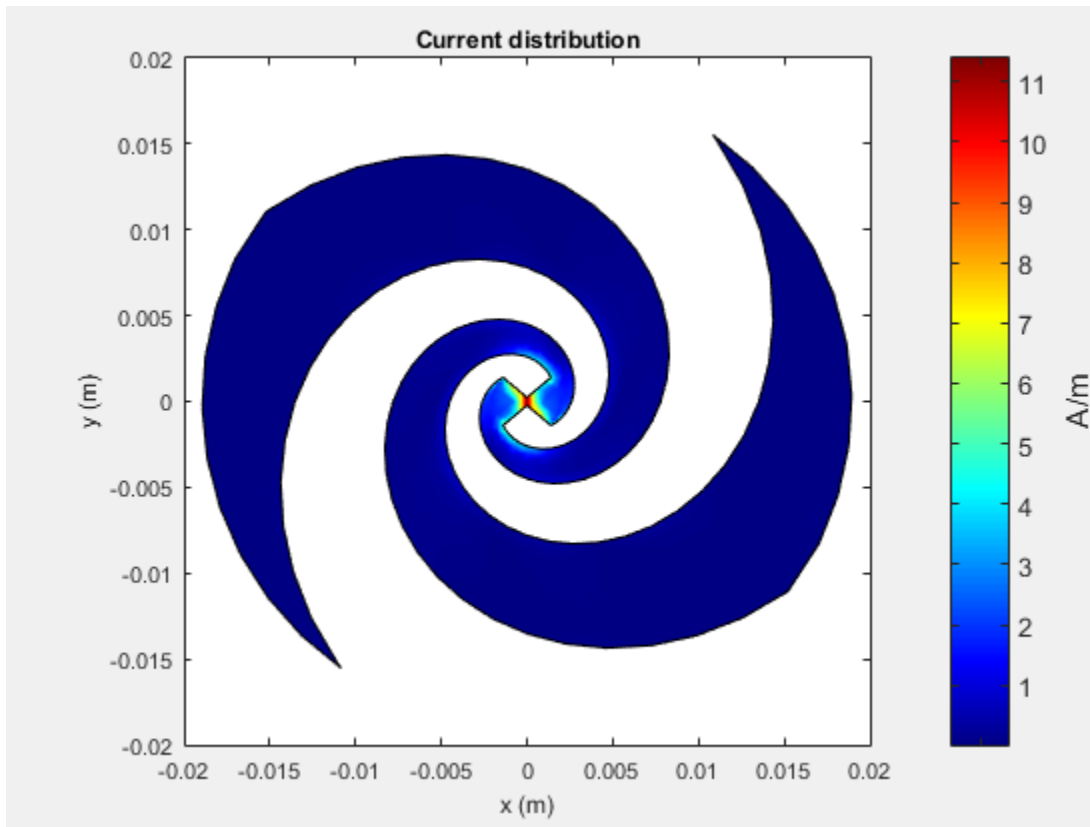
A typical antenna surface has current flowing on it. The behavior of the antenna surface current depends on the frequency of the input source, the geometry of the antenna, and the material properties of the antenna. The current is a vector and is spatially related to the structure of the antenna. In a dipole antenna, the maximum current distribution is in the middle of the antenna and the minimum is toward the end:

```
d = dipole;
current(d,70e6)
```



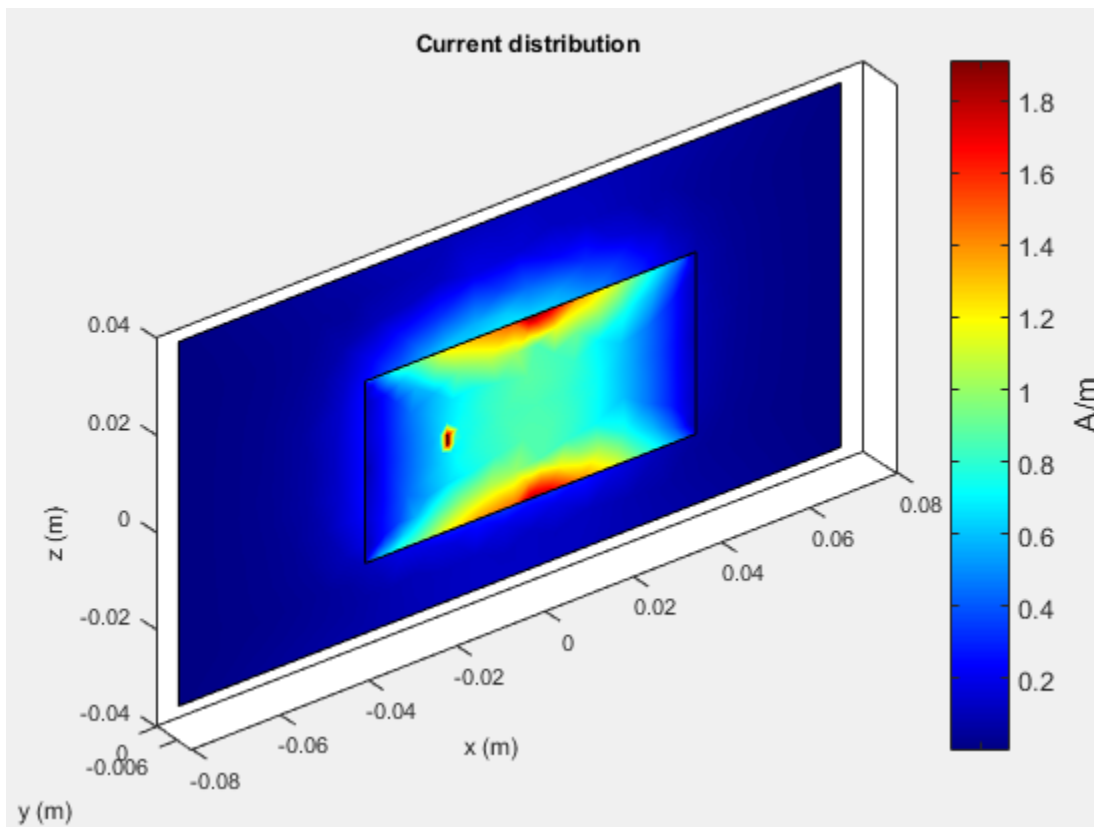
The same is true for a spiral antenna:

```
s = spiralEquiangular;  
current(s,4e9)
```



The patch also shows the current distribution of the classic $\lambda/2$ open-open resistor. The two ends of the patch represent an open circuit since the current is at a minimum.

```
pm = patchMicrostrip;  
current(pm, 1.75e9)
```



The spatial relationship between the current and structure of antenna is termed mode.

References

- [1] Balanis, C.A. *Antenna Theory. Analysis and Design*, 3rd Ed. New York: Wiley, 2005.
- [2] Makarov, S.N. *Antenna and EM Modeling with MATLAB*, New York: Wiley & Sons, 2002, p. 66.

Meshing

In this section...

“Automatic Meshing Mode” on page 1-15

“Manual Meshing Mode” on page 1-16

“Strip Meshing” on page 1-17

“Surface Meshing” on page 1-17

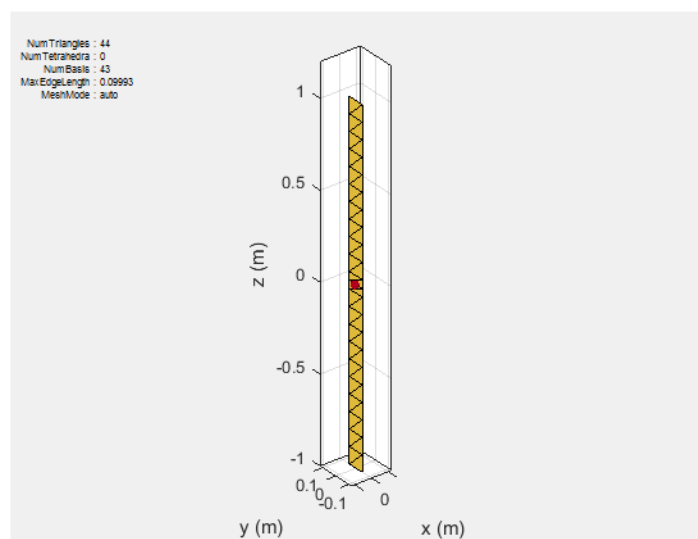
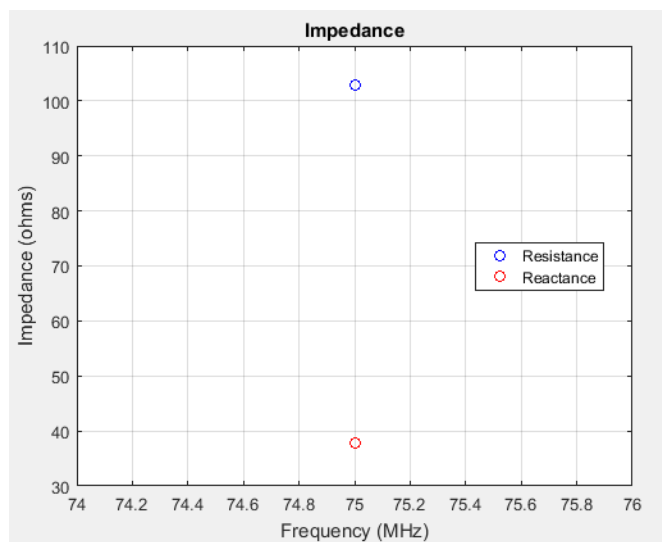
“Meshing a Dielectric Substrate” on page 1-18

Method of Moments (MoM) is a numerical method that transforms Maxwell’s continuous integral equations into an approximate discrete formulation that requires inversion of a large matrix. *Meshing* is the process of converting the continuous domain into the discrete domain for solving the equations. For discretizing surfaces, typically either triangles or rectangles are used. Antenna Toolbox uses triangular element for meshing as it conforms better for arbitrary shaped surfaces. The triangles are used to approximate the surface current using the Rao-Wilton-Glisson (RWG) basis functions. To get an accurate result, ensure that large number of triangles are present in the region where current variation is the highest. This region is typically either the corners in the antenna geometry or at the point where the antenna is excited.

Automatic Meshing Mode

In Antenna Toolbox, the antenna structures mesh automatically based on the analysis frequency chosen. For analysis functions that accept a scalar frequency, the antennas mesh at that single frequency to satisfy the minimum triangles required. Then the functions calculate the corresponding antenna parameter.

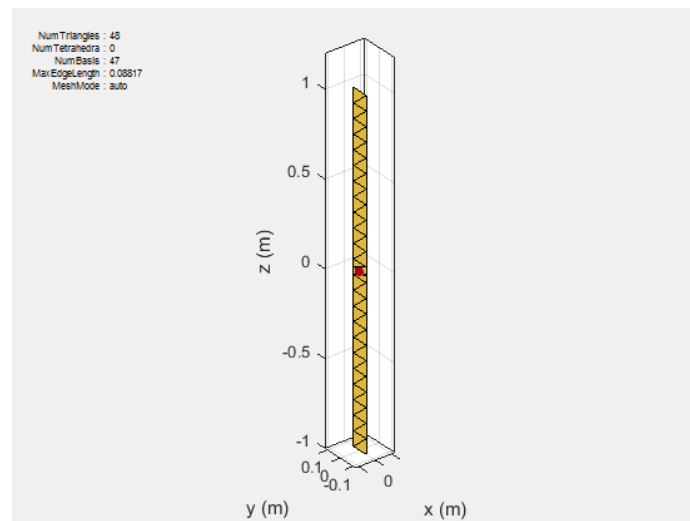
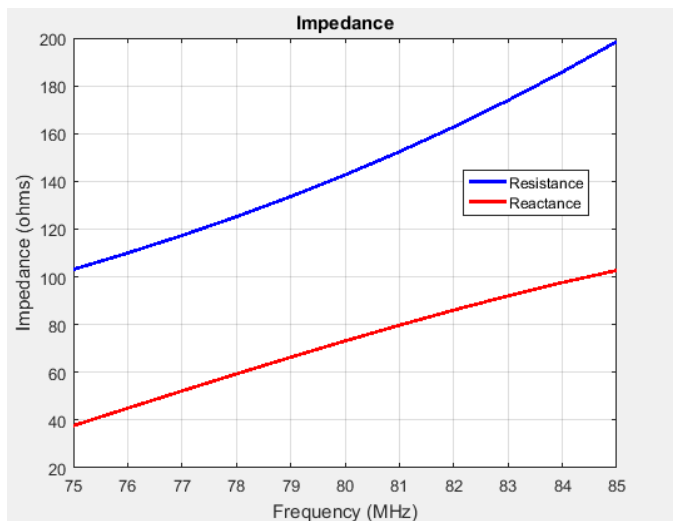
```
d = dipole;
impedance(d,75e6);
mesh(d)
```



In above example, the dipole is meshed at 75 MHz automatically before calculating the impedance at that value. Use the mesh command to view the meshed dipole. The number of triangles is 44.

For analysis functions that accept a frequency vector (`impedance`, `sparameters`, `returnLoss`, `vswr`), each antenna meshes once at the highest frequency. Then, the functions calculate the corresponding antenna parameters at all the frequencies in the range.

```
d = dipole;
impedance(d, 75e6:1e6:85e6);
mesh(d)
```



In above example, the dipole is meshed at the highest frequency, 85 MHz automatically before calculating the impedance at all the frequencies from 75 to 85 MHz. Meshing at the highest frequency, 85 MHz, ensures maximum number of triangles and a smoother plot of the dipole impedance. Use the `mesh` command to view the meshed dipole. The number of triangles is 48, which is more than single frequency meshing.

Manual Meshing Mode

You can choose to mesh the structure manually at the highest frequency of interest. Manual meshing is done by specifying the maximum edge length that is used for discretizing the structure. One option is to specify the value to be one-tenth of the wavelength at the highest frequency of interest. For example:

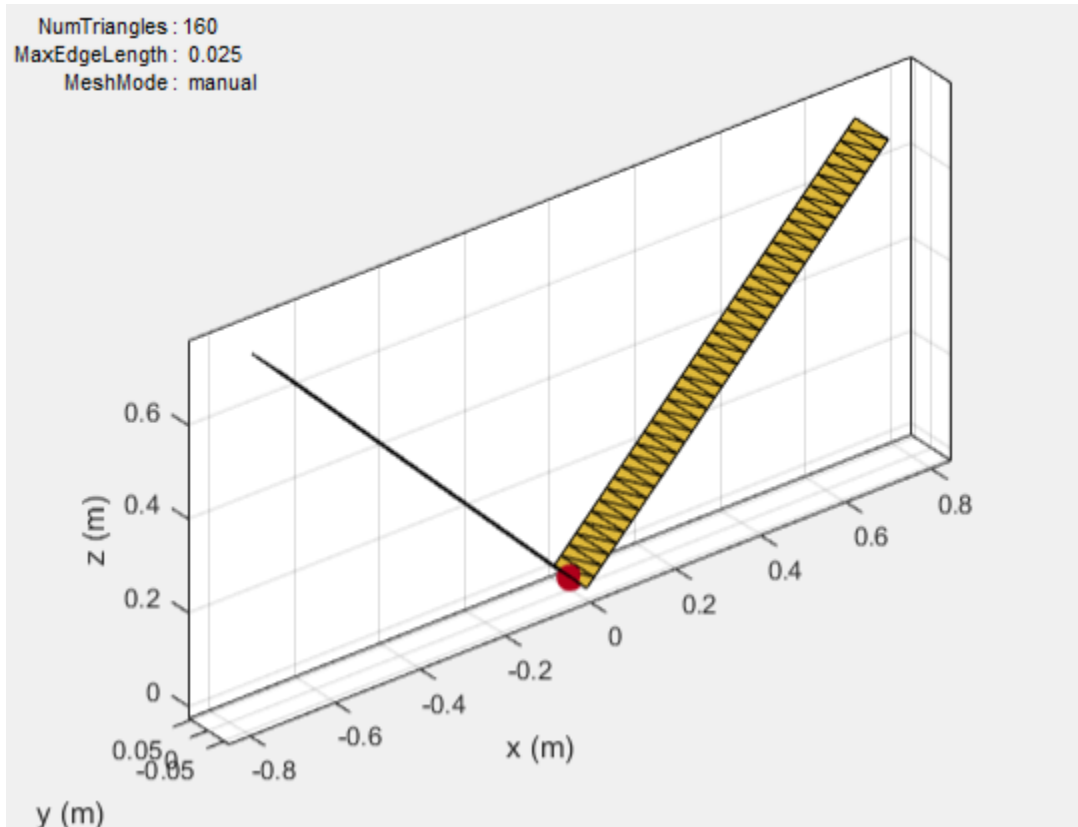
```
sp = spiralArchimedean;
freq = 0.8e9:100e6:2.5e9;
lambda = 3e8/freq(end);
mesh(sp, 'MaxEdgeLength', lambda/10);
```

Alternatively, you can run an analysis at the highest frequency of interest and get the maximum edge length. Specify this maximum edge length using the `mesh` function as shown. This mesh is used for all other calculations.

```
sp = spiralArchimedean;
freq = 0.8e9:100e6:2.5e9;
temp = axialRatio(sp, freq(end), 0, 90);
meshdata = mesh(sp);
mesh(sp, 'MaxEdgeLength', meshdata.MaxEdgeLength);
```

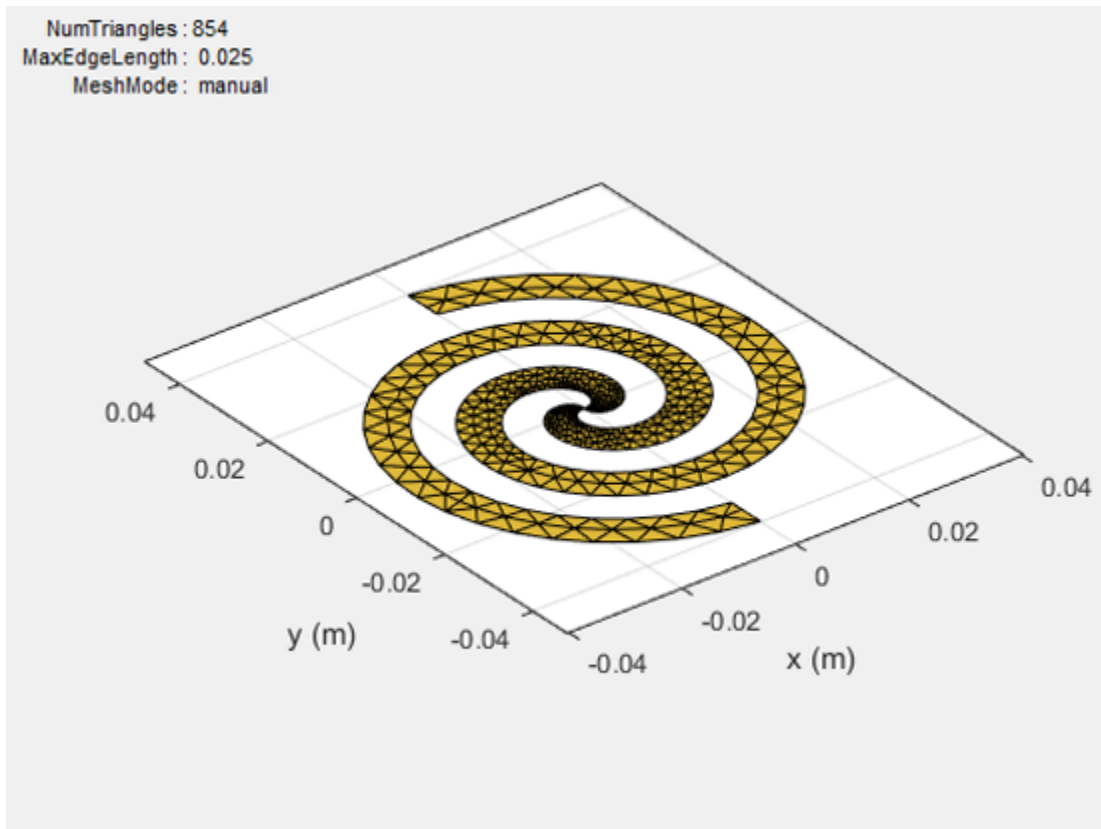
Strip Meshing

For strip meshing, include at least 10 triangles per wavelength in a strip. This rule applies for structures such as dipoles, monopoles, and loops. Antenna Toolbox antenna meets the requirement automatically, based on the analysis frequency specified. The structured mesh generated in such cases is shown:



Surface Meshing

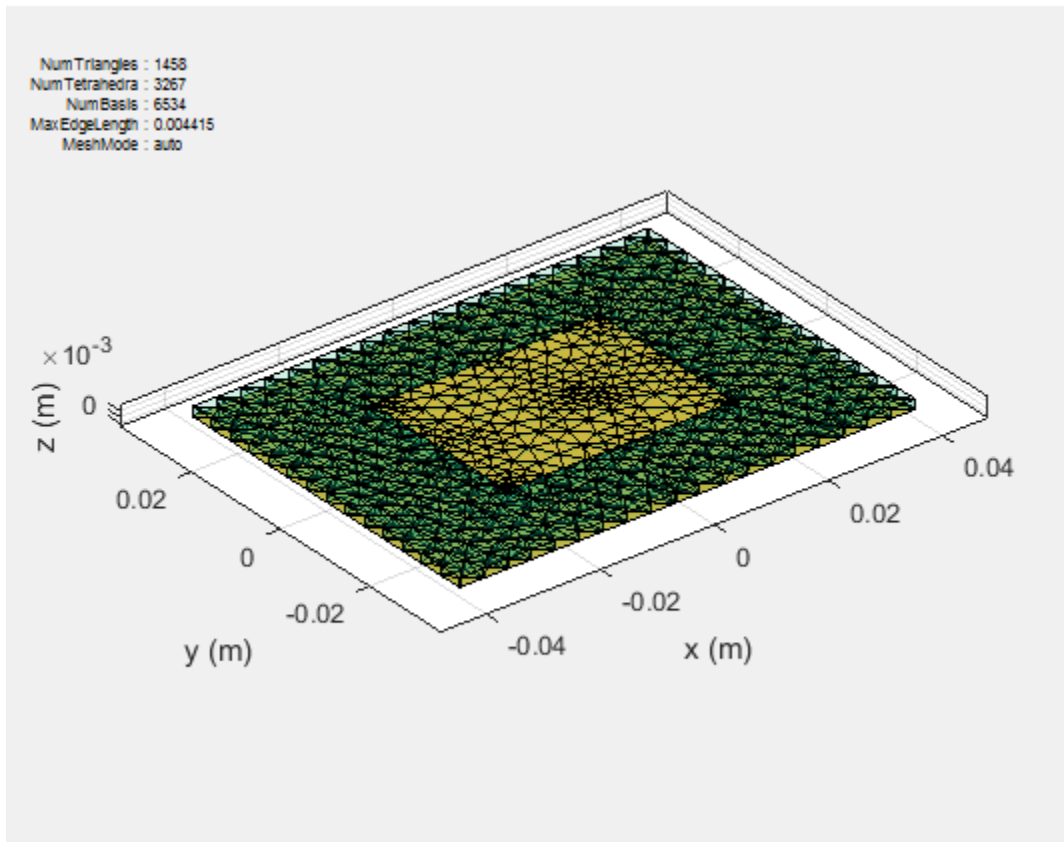
For surface meshing, it is recommended that there be at least 100 elements per wavelength in a particular area. This rule applies to structures such as spirals, patches, and ground planes in general. Antenna Toolbox antenna meets the requirement automatically, based on the analysis frequency specified. In these cases, a non-uniform mesh is generated as shown:



Larger number of triangles are added in regions with higher current density.

Meshing a Dielectric Substrate

For antennas using dielectrics and metals, Antenna Toolbox uses tetrahedrons to discretize the volume of the dielectric substrate.



Thickness of the dielectric substrate is measured with respect to the wavelength. A dielectric substrate with thickness less than or equal to 1/50th of the wavelength is a thin substrate. When you mesh an antenna using dielectric in auto mode, thin substrates yield more accurate solutions.

A substrate with a thickness of 1/10th of the wavelength is a thick dielectric substrate. The method of moments solver requires 10 elements per wavelength to yield an accurate solution. Manual meshing yields more accurate solutions for antennas using thick dielectric substrate, as it satisfies the 10 elements per wavelength criteria.

References

- [1] Makarov, S.N. *Antenna and EM Modeling with MATLAB*, New York: Wiley & Sons, 2002

Field Analysis

In this section...

“Radiation Pattern” on page 1-20

“Beamwidth” on page 1-23

“E-Plane and H-Plane” on page 1-24

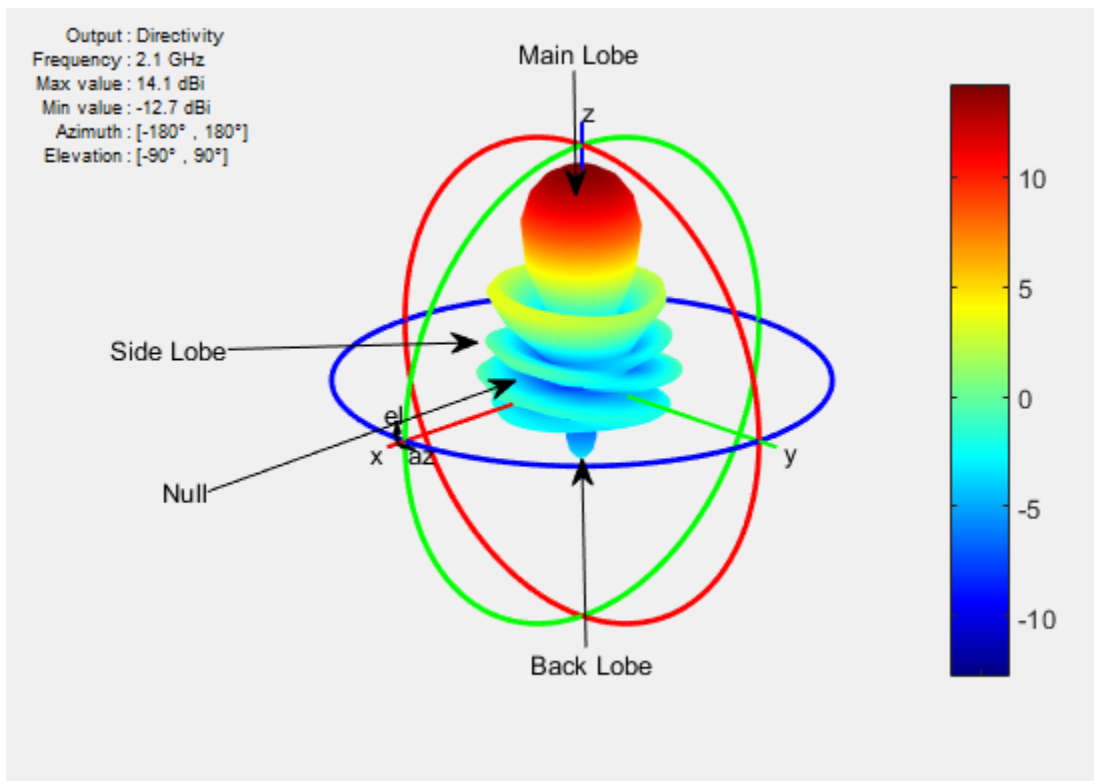
“Polarization” on page 1-26

“Axial Ratio” on page 1-30

Radiation Pattern

The *radiation pattern* of an antenna is the spatial distribution of power. The pattern displays the directivity or gain of the antenna. The *power pattern* of an antenna plots the transmitted or received power for a given radius. The *field pattern* of an antenna plots the variation in the electric or magnetic field for a given radius. The radiation pattern provides details such as the maximum and minimum value of the field quantity and the range of angles over which data is plotted.

```
h = helix;
h.Turns = 13;
h.Radius = 0.025;
pattern(h,2.1e9)
```

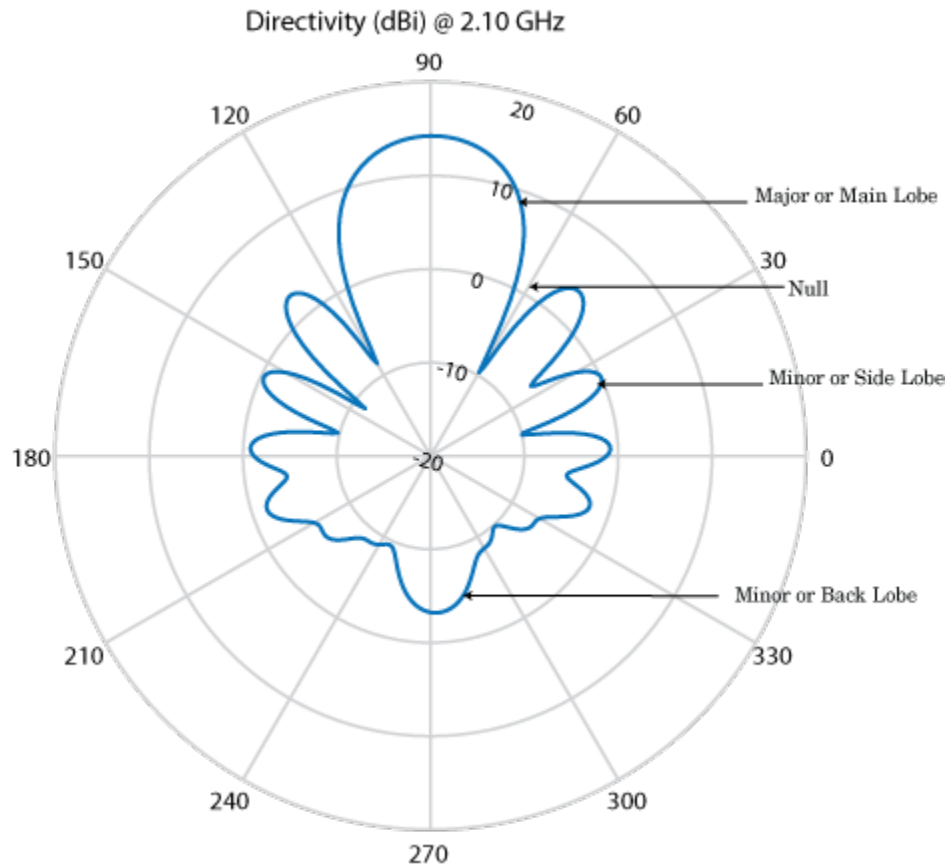


Use the `pattern` function to plot radiation pattern of any antenna in the Antenna Toolbox. By default, the function plots the directivity of the antenna. You can also plot the electric field and power pattern by using `Type` name-value pair argument of the `pattern` function.

Lobes

Each radiation pattern of an antenna contains *radiation lobes*. The lobes are divided into *major lobes* (also called main lobes) and *minor lobes*. *Side lobes* and *back lobes* are variations of minor lobes.

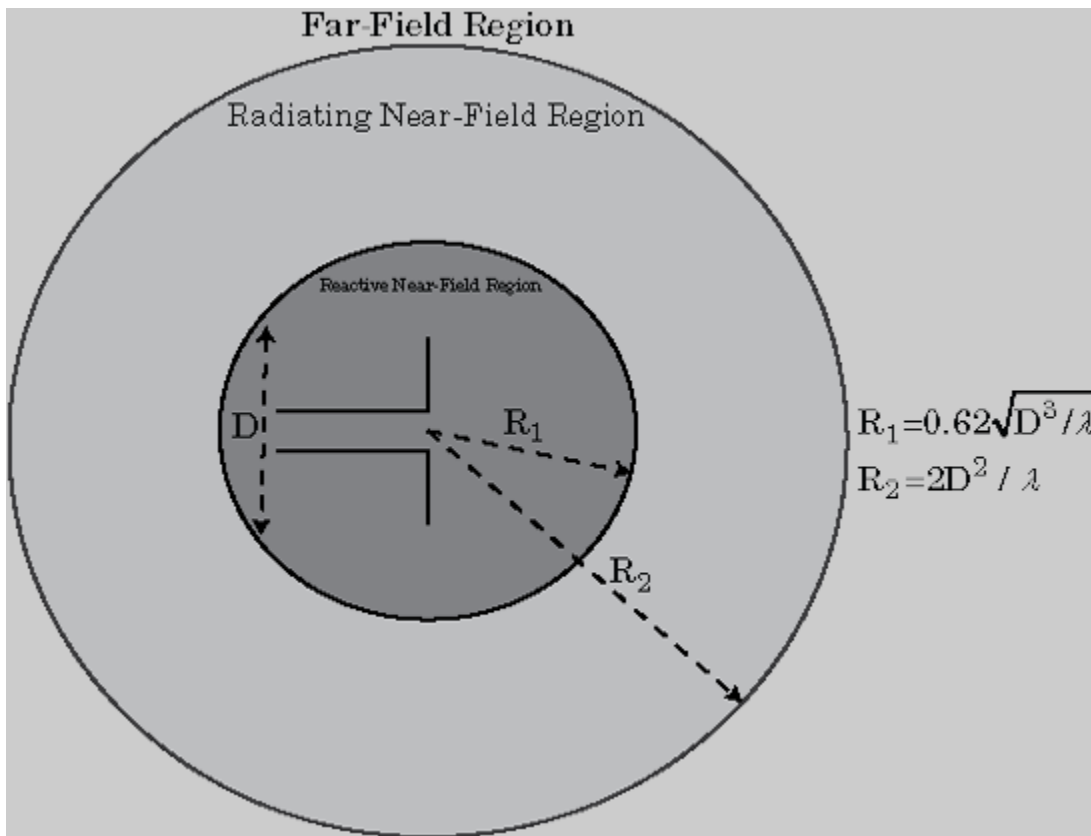
```
h = helix;
h.Turns = 13;
h.Radius = 0.025;
patternElevation(h,2.1e9)
```



- **Major or Main lobe:** Shows the direction of maximum radiation, or power, of the antenna.
- **Minor lobe:** Shows the radiation in undesired directions of antenna. The fewer the number of minor lobes, the greater the efficiency of the antenna. *Side lobes* are minor lobes that lie next to the major lobe. *Back lobes* are minor lobes that lie opposite to the major lobe of antenna.
- **Null:** Shows the direction of zero radiation intensity of the antenna. Nulls usually lie between the major and minor lobe or in between the minor lobes of the antennas.

Field Regions

For an antenna engineer and an electromagnetic compatibility (EMC) engineer, it is important to understand the regions around the antenna.



The region around an antenna is defined in many ways. The most used description is a 2- or 3-region model. The 2-region model uses the terms *near field* and the *far field* to identify specific dominant field mechanisms. The diagram is a representation of antenna fields and boundaries. The 3-field region splits the near field into a transition zone, where a weakly radiative mechanism is at work.

Near-Field Region: The *near-field* region is divided into two transition zones: a reactive zone and radiating zone.

- **Reactive Near-Field Region:** This region is closest to the antenna surface. The reactive field dominates this region. The reactive field is stored energy, or standing waves. The fields in this region change rapidly with distance from the antenna. The equation for outer boundary of this region is: $R < 0.62\sqrt{D^3/\lambda}$ where R is the distance from the antenna, λ is the wavelength, and D is the largest dimension of the antenna. This equation holds true for most antennas. In a very short dipole, the outer boundary of this region is $\lambda/2\pi$ from the antenna surface.
- **Radiating Near-Field Region:** This region is also called the *Fresnel region* and lies between the reactive near-field region and the far-field region. The existence of this region depends on the largest dimension of the antenna and the wavelength of operation. The radiating fields are dominant in this region. The equation for the inner boundary of the region is equation $R \geq 0.62\sqrt{D^3/\lambda}$ and the outer boundary is $R < 2D^2/\lambda$. This holds true for most antennas. The field distribution depends on the distance from the antenna.

Far-field Region: This region is also called *Fraunhofer region*. In this region, the field distribution does not depend on the distance from the antenna. The electric and magnetic fields in

this region are orthogonal to each other. This region contains propagating waves. The equation for the inner boundary of the far-field is $R = 2D^2/\lambda$ and the equation for the outer boundary is infinity.

Directivity and Gain

Directivity is the ability of an antenna to radiate power in a particular direction. It can be defined as ratio of maximum radiation intensity in the desired direction to the average radiation intensity in all other directions. The equation for directivity is:

$$D = \frac{4\pi U(\theta, \phi)}{P_{rad}}$$

where:

- D is the directivity of the antenna
- U is the radiation intensity of the antenna
- P_{rad} is the average radiated power of antenna in all other directions

Antenna directivity is dimensionless and is calculated in decibels compared to the isotropic radiator (dBi).

The *gain* of an antenna depends on the directivity and efficiency of the antenna. It can be defined as the ratio of maximum radiation intensity in the desired direction to the total power input of the antenna. The equation for gain of an antenna is:

$$G = \frac{4\pi U(\theta, \phi)}{P_{in}}$$

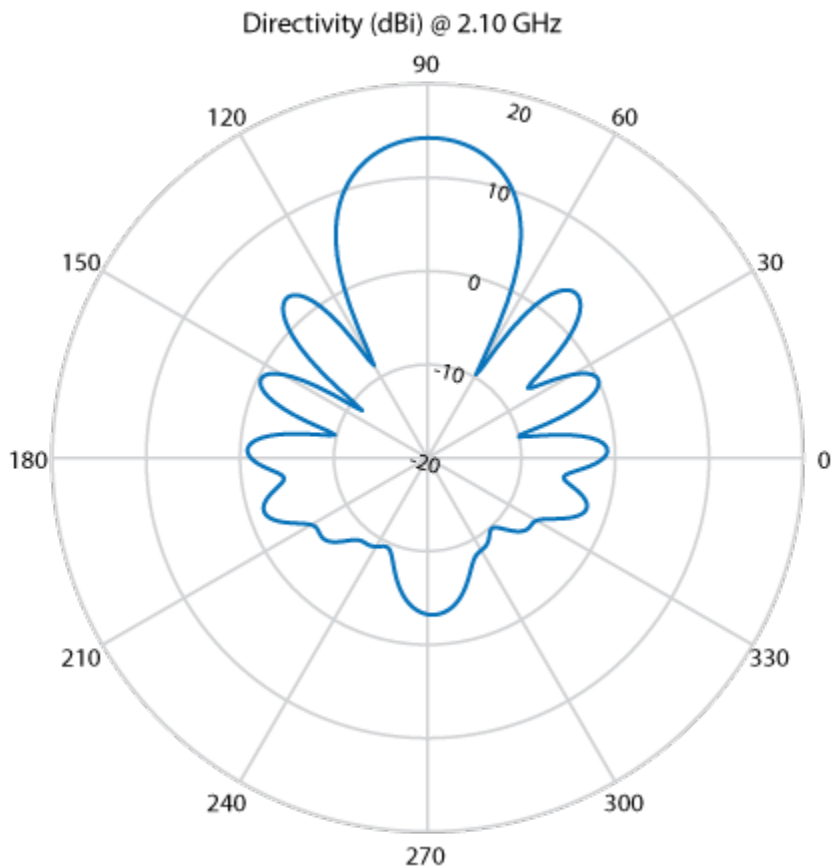
where:

- G is the gain of the antenna
- U is the radiation intensity of the antenna
- P_{in} is the total power input to the antenna

If the efficiency of the antenna in the desired direction is 100%, then the total power input to the antenna is equal to the total power radiated by the antenna, that is, $P_{in} = P_{rad}$. In this case, the antenna directivity is equal to the antenna gain.

Beamwidth

Antenna beamwidth is the angular measure of the antenna pattern coverage. As seen in the figure, the main beam is a region around maximum radiation. This beam is also called the major lobe, or main lobe of the antenna.



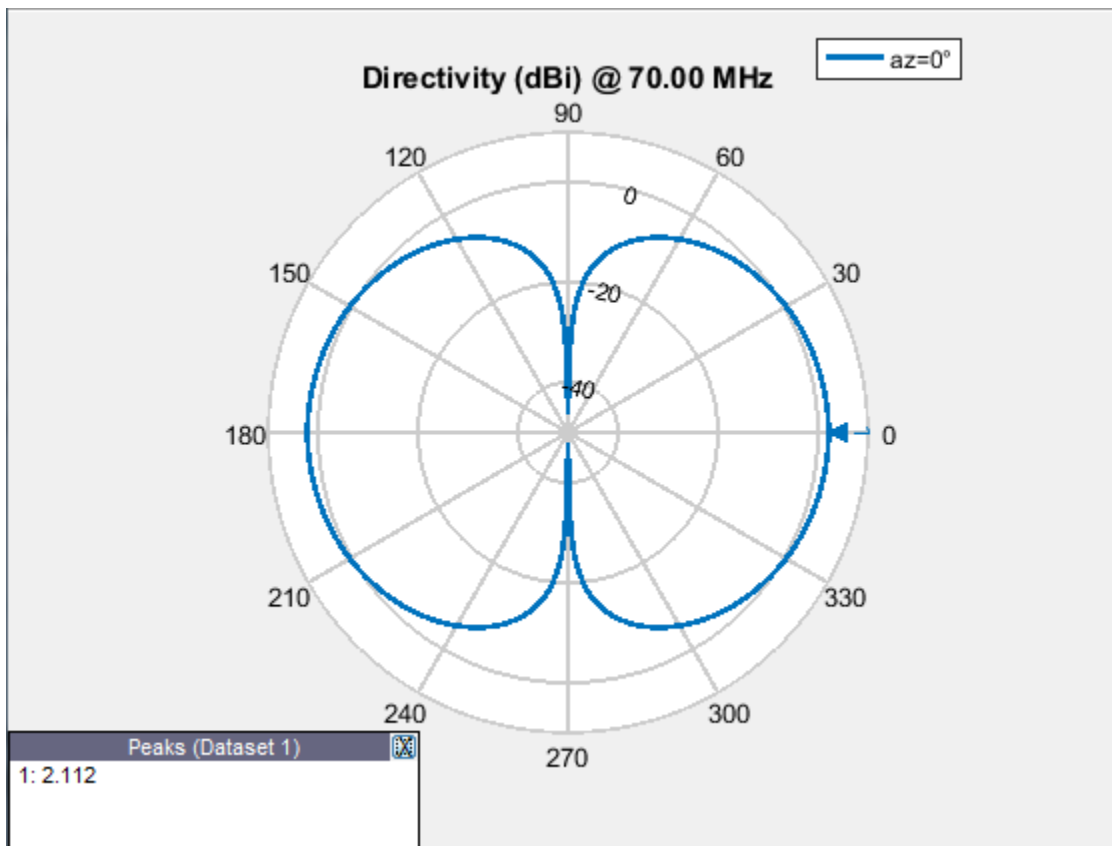
Half power beamwidth (HPBW) is the angular separation in which the magnitude of the radiation pattern decreases by 50% (or -3dB) from the peak of the main beam

Use the `beamwidth` function to calculate the beamwidth of any antenna in Antenna Toolbox.

E-Plane and H-Plane

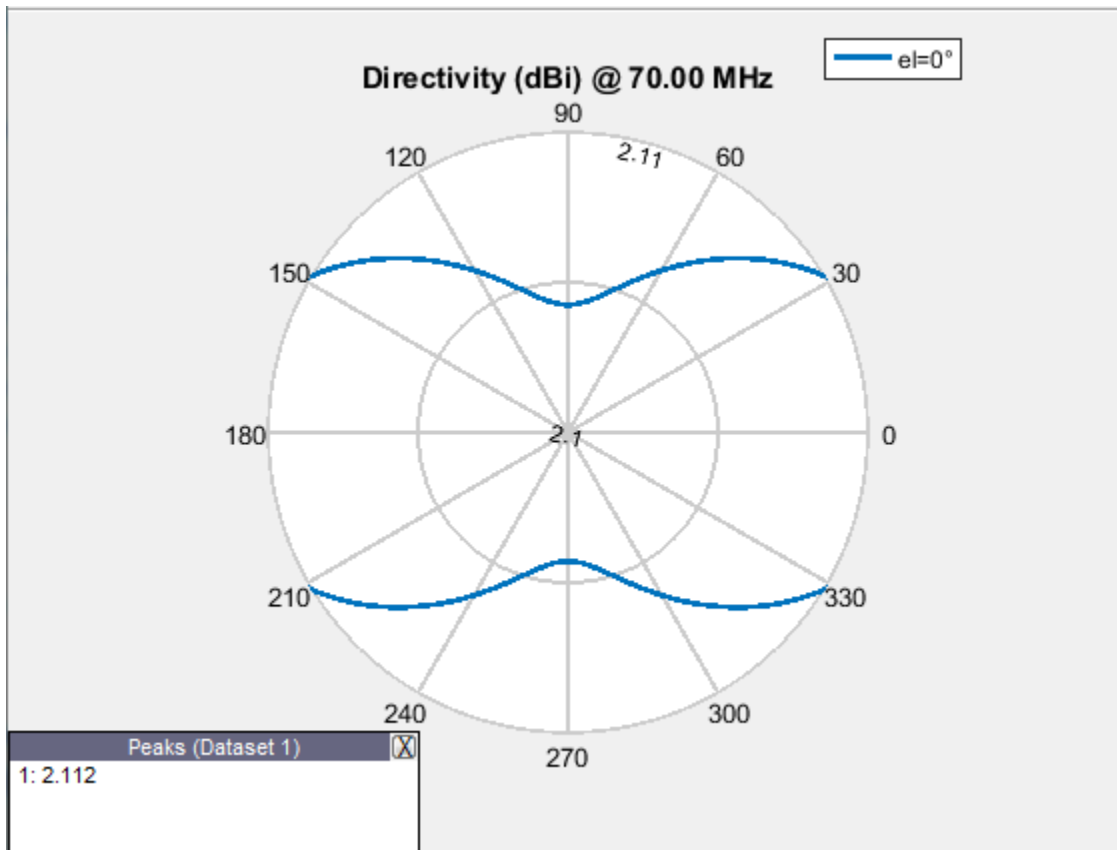
E-plane: Plane containing the electric field vector and the direction of maximum radiation. Consider a dipole antenna that is vertical along the z-axis. Use the `patternElevation` function to plot the elevation plane pattern. The elevation plane pattern shown captures the E-plane behavior of the dipole antenna.

```
d = dipole;
patternElevation(d,70e6)
```



H-plane: Plane containing the magnetic field vector and the direction of maximum radiation. Use the `patternAzimuth` function to plot the azimuth plane pattern of a dipole antenna. The azimuthal variation in pattern shown captures the H-plane behavior of the dipole antenna.

```
d = dipole;  
patternAzimuth(d,70e6)
```



Use `EHfields` to measure the electric and magnetic fields of the antenna. The function can be used to calculate both near and far fields.

Polarization

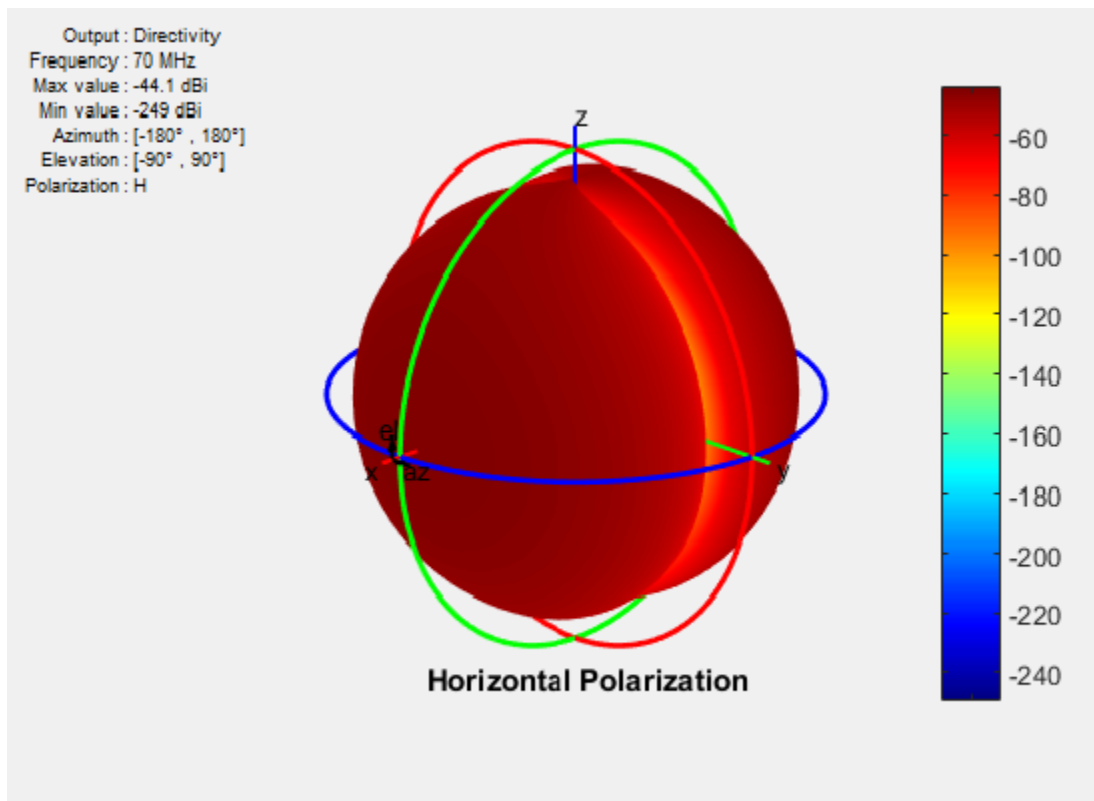
Polarization is the orientation of the electric field, or E-field, of an antenna. Polarization is classified as elliptical, linear, or circular.

Elliptical polarization: If the electric field remains constant along the length but traces an ellipse as it moves forward, the field is elliptically polarized. Linear and circular polarizations are special cases of elliptical polarization.

Linear polarization: If the electric field vector at a point in space is directed along a straight line, the field is linearly polarized. A linearly polarized antenna radiates only one plane and this plane contains the direction of propagation of the radio waves. There are two types of linear polarization:

- **Horizontal Polarization:** The electric field vector is parallel to the ground plane. To view the horizontal polarization pattern of an antenna, use the `pattern` function, with the 'Polarization' name-value pair argument set to 'H'. The plot shows the horizontal polarization pattern of a dipole antenna:

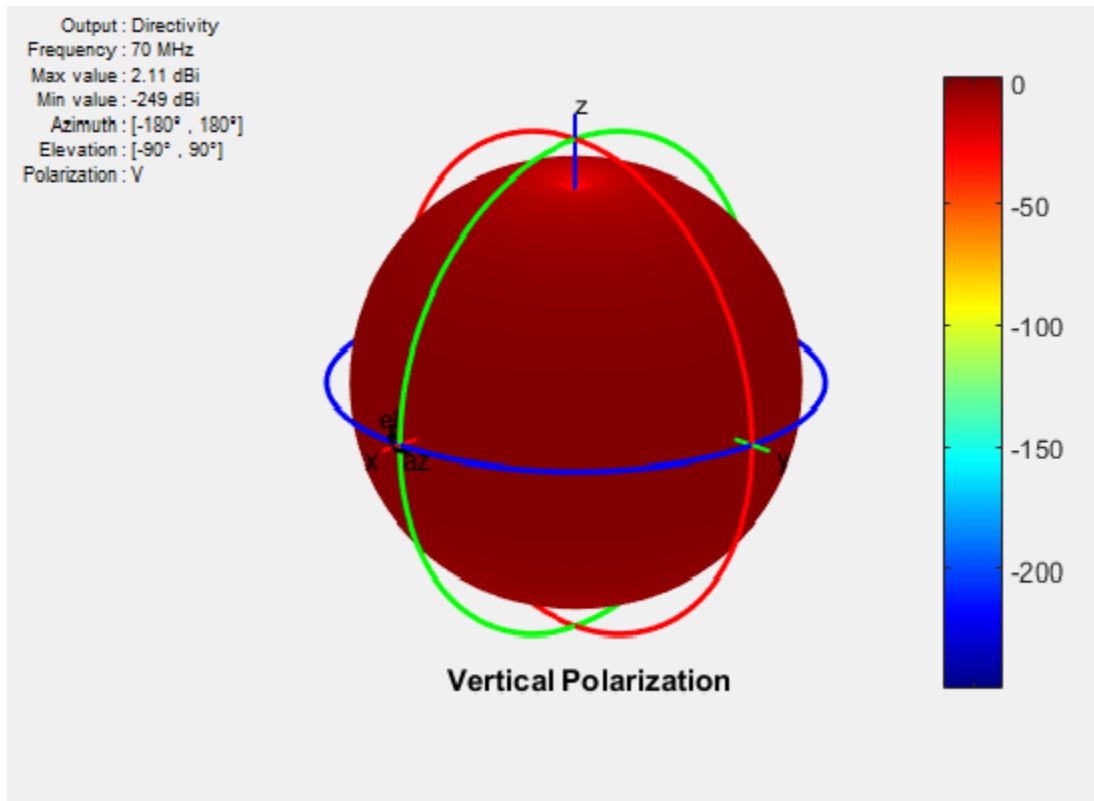
```
d = dipole;
pattern(d,70e6,'Polarization','H')
```



USA television networks use horizontally polarized antennas for broadcasting.

- **Vertical Polarization:** The electric field vector is perpendicular to the ground plane. To view the vertical polarization pattern of an antenna, use the `pattern` function, with the 'Polarization' name-value pair argument set to 'V'. Vertical polarization is used when a signal has to be radiated in all directions. The plot shows the vertical polarization pattern of a dipole antenna:

```
d = dipole;
pattern(d,70e6, 'Polarization', 'V')
```

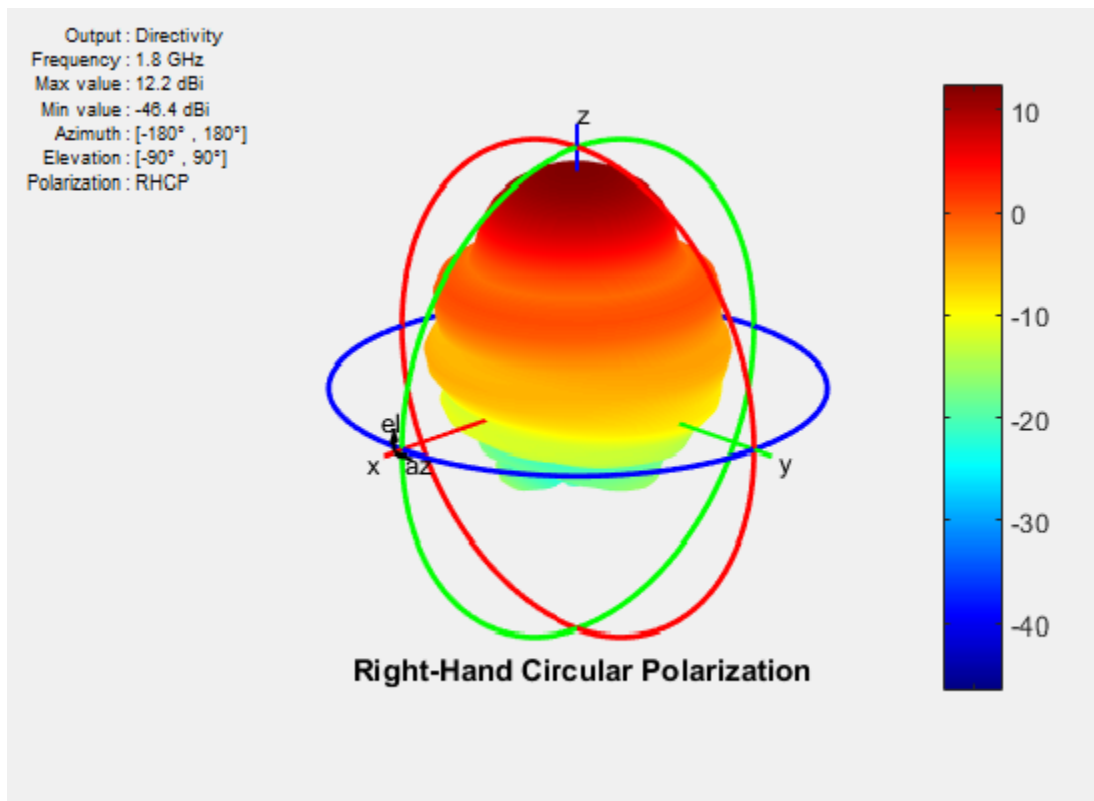


An AM radio broadcast antenna or an automobile whip antenna are some examples of vertically polarized antennas.

Circular Polarization: If the electric field remains constant along the straight line but traces circle as it moves forward, the field is circularly polarized. This wave radiates in both vertical and horizontal planes. Circular polarization is most often used in satellite communications. There are two types of circular polarization:

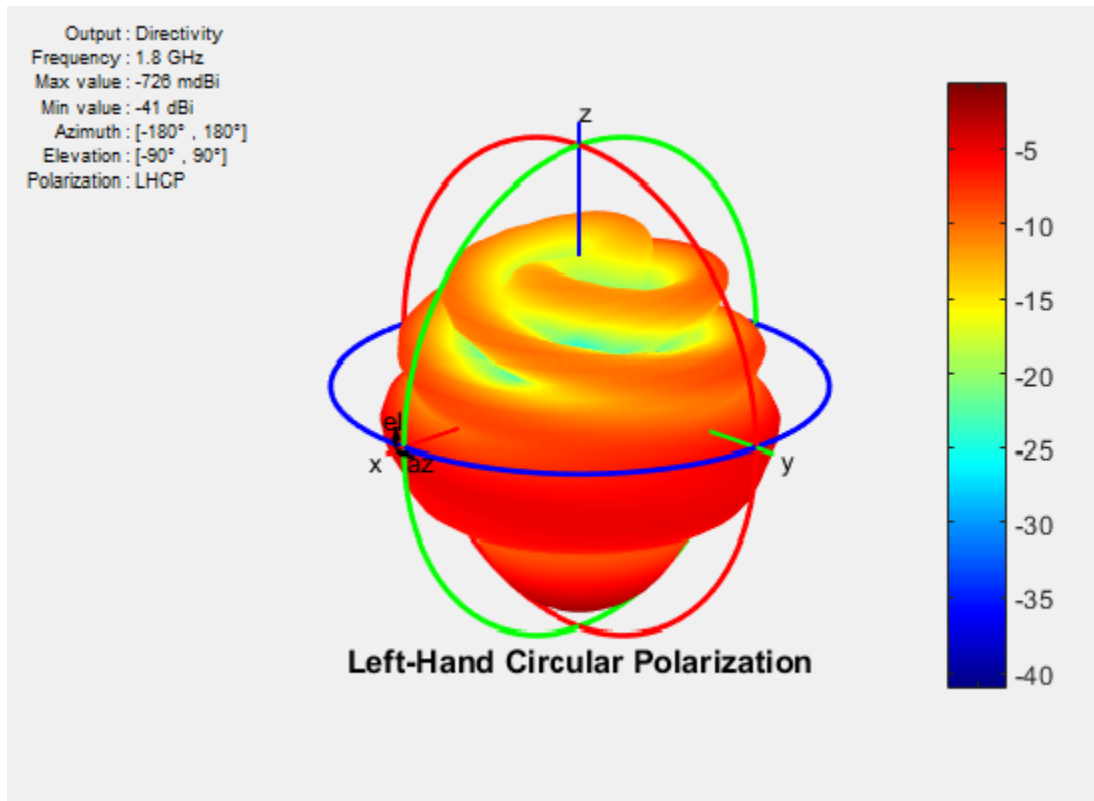
- **Right-Hand Circularly Polarized (RHCP):** The electric field vector is traced in the counterclockwise direction. To view the RHCP pattern of an antenna, use the `pattern` function, with the 'Polarization' name-value pair argument set to 'RHCP'. The plot shows RHCP pattern of helix antenna:

```
h = helix;  
h.Turns = 13;  
h.Radius = 0.025;  
pattern(h,1.8e9,'Polarization','RHCP')
```



- Left-Hand circularly polarized (LHCP): The electric field vector is traced in the clockwise direction. To view the LHCP pattern of an antenna, use the `pattern` function, with the 'Polarization' name-value pair argument set to 'LHCP'. The plot shows LHCP pattern of helix antenna:

```
h = helix;  
h.Turns = 13;  
h.Radius = 0.025;  
pattern(h,1.8e9,'Polarization','LHCP')
```



For efficient communications, the antennas at the transmitting and receiving end must have same polarization.

Axial Ratio

Axial ratio (AR) of an antenna in a given direction quantifies the ratio of orthogonal field components radiated in a circularly polarized wave. An axial ratio of infinity implies a linearly polarized wave. When the axial ratio is 1, the radiated wave has pure circular polarization. Values greater than 1 imply elliptically polarized waves.

Use `axialRatio` to calculate the axial ratio for any antenna in the Antenna Toolbox.

References

- [1] Balanis, C.A. *Antenna Theory. Analysis and Design*, 3rd Ed. New York: Wiley, 2005.
- [2] Stutzman, Warren.L and Thiele, Gary A. *Antenna Theory and Design*, 3rd Ed. New York: Wiley, 2013.
- [3] Capps, C. *Near Field or Far Field*, EDN, August 16, 2001, pp.95 - pp.102.

Rotate Antennas and Arrays

In this section...

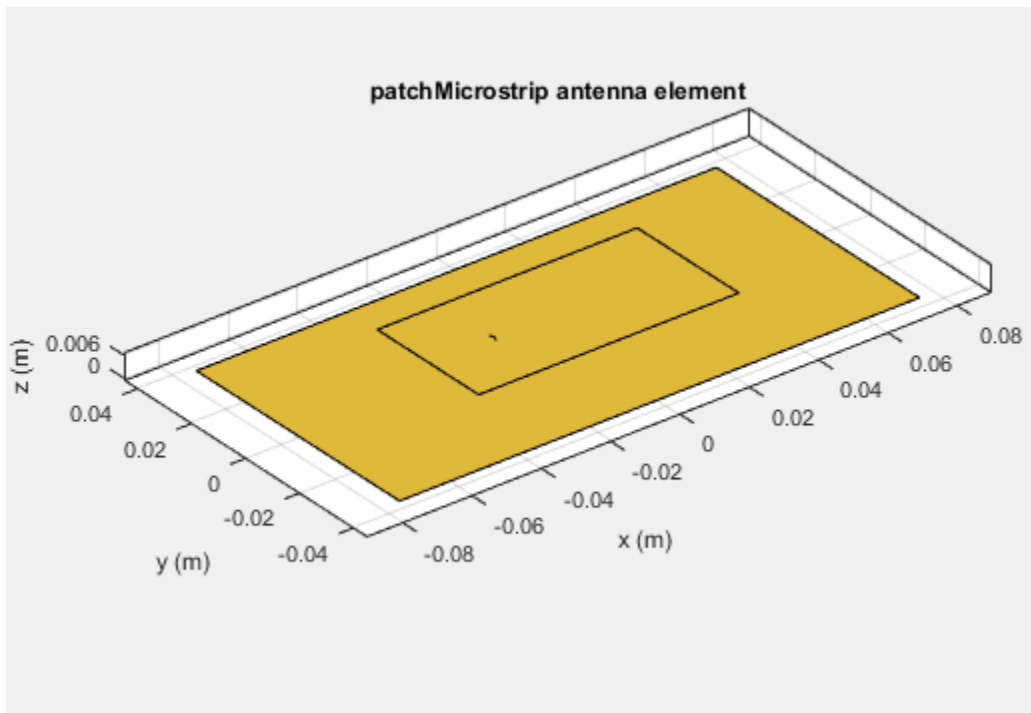
“Rotate Single Antenna Element” on page 1-31
 “Rotate Single Antenna Element About Multiple Axes” on page 1-33
 “Rotate Dipole-Backed Reflector” on page 1-34
 “Rotate Antenna Array” on page 1-37

To rotate antenna elements in Antenna Toolbox, use the `Tilt` and `TiltAxis` properties. The `Tilt` property specifies the angles of rotation of the antenna in degrees. The `TiltAxis` property specifies the one or more axes of rotation (X , Y , Z) of the antenna.

Rotate Single Antenna Element

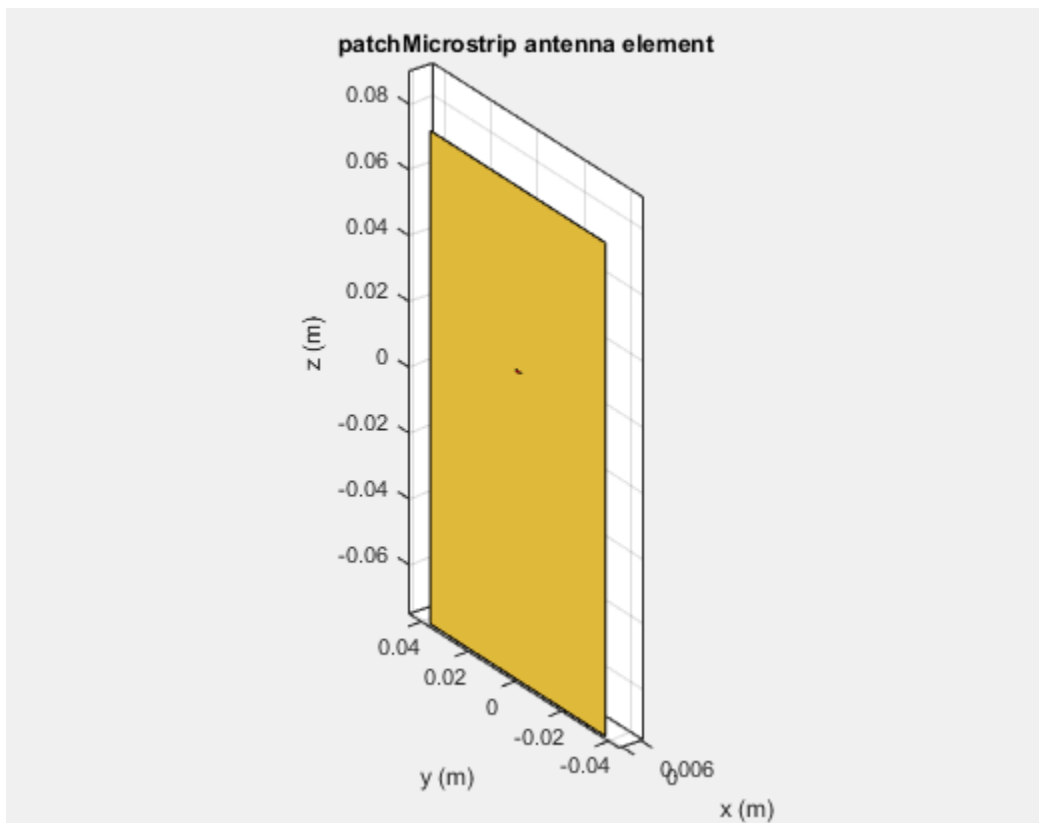
Create a patch antenna. By default, this antenna is on the X - Y plane.

```
patch = patchMicrostrip;
show(patch)
```



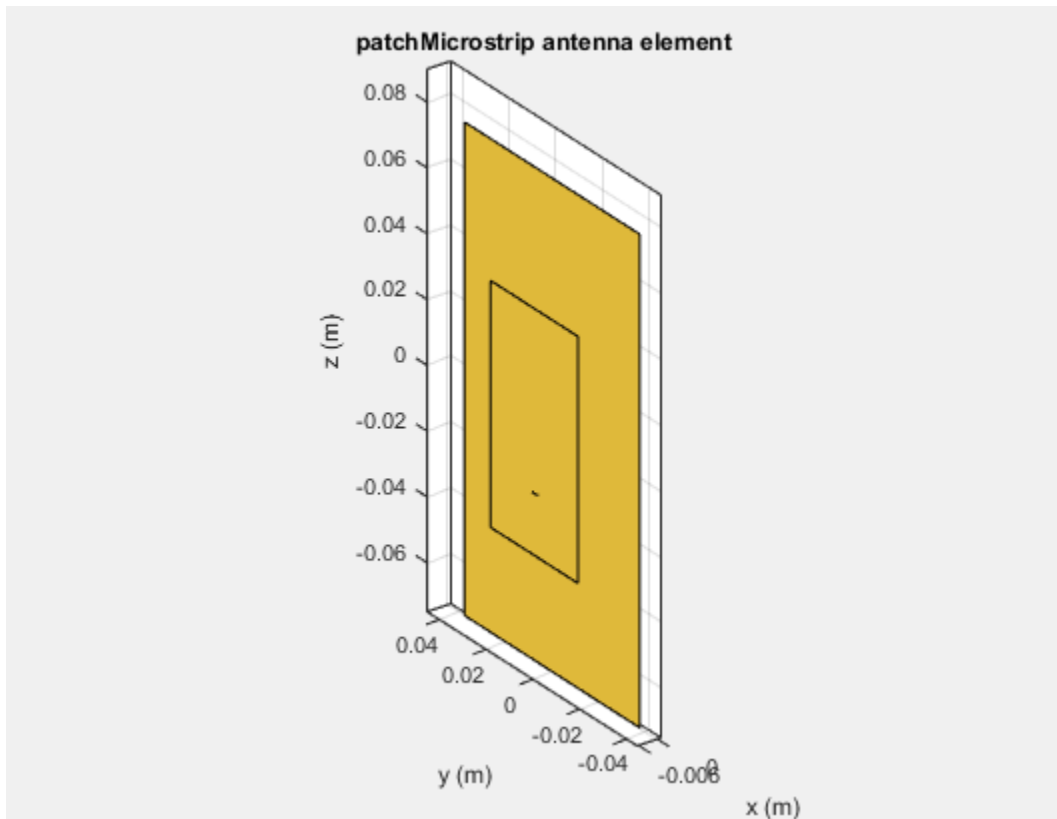
To make the antenna lie on the Y - Z plane, rotate it by 90 degrees about the Y -axis. The rotation follows the standard right-hand rule.

```
patch.TiltAxis = 'Y';
patch.Tilt = 90;
show(patch)
```



The patch lies behind the ground plane and is not visible. To make the patch visible change the `TiltAxis` property to `[0 -1 0]`. The `Tilt` property is still `90`, but the axis of rotation is now the Y-axis. The negative number in the `TiltAxis` vector determines the direction of rotation about the Y-axis as per the right-hand rule.

```
patch.TiltAxis = [0 -1 0];  
show(patch)
```



Rotate Single Antenna Element About Multiple Axes

Create a dipole antenna. Change the direction of rotation of the antenna with two rotations simultaneously. Rotate the antenna by 90 degrees about the axis specified by $[0 \ 1 \ 0]$ and rotate it by 90 degree about the axis specified by $[0 \ 1 \ 1]$.

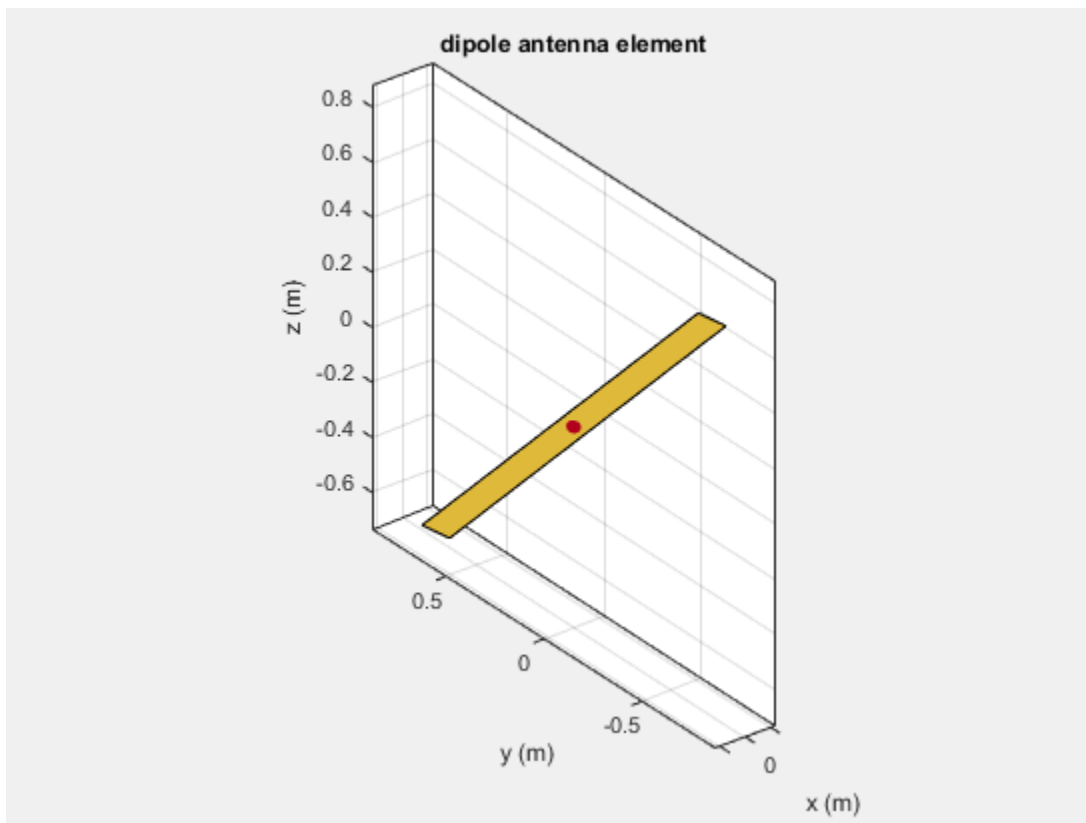
```
d = dipole('Tilt',[90 90],'TiltAxis', [0 1 0; 0 1 1])
```

dipole with properties:

```

    Length: 2
    Width: 0.1000
    FeedOffset: 0
    Tilt: [90 90]
    TiltAxis: [2x3 double]
```

```
show(d)
```

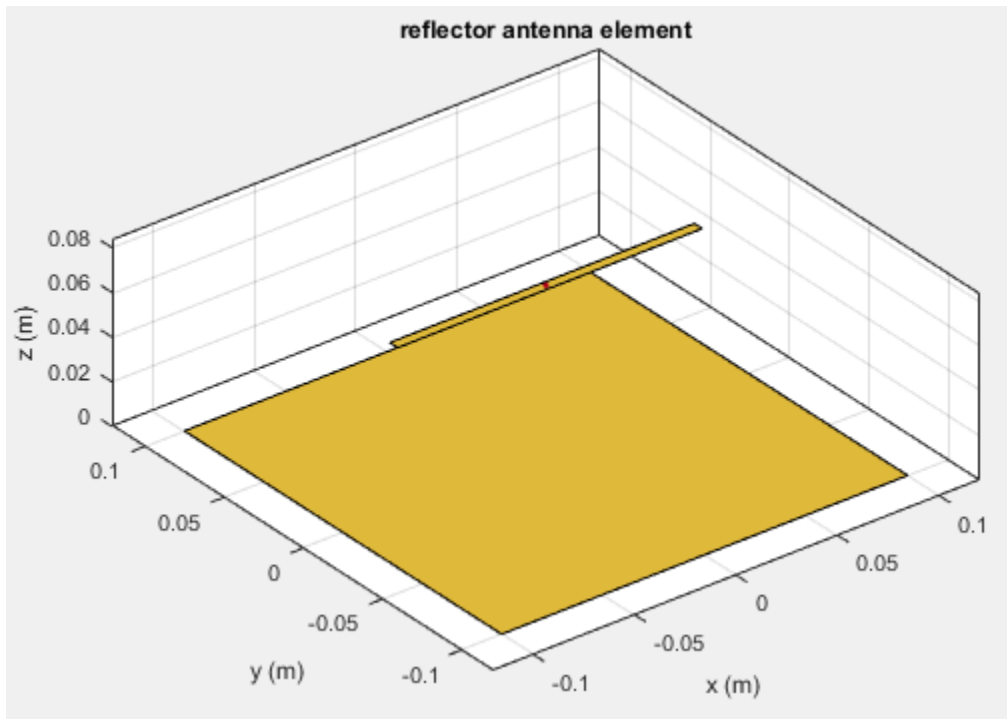


Rotate Dipole-Backed Reflector

The `Tilt` and `TiltAxis` properties are available for dipole and reflector elements. Use these properties if you want to model an antenna parallel or perpendicular to the ground plane.

Create a reflector element. By default, the dipole is parallel to the reflector element.

```
r = reflector;  
show(r)
```



The dipole is the exciter element of the reflector. View its properties.

```
r.Exciter;
```

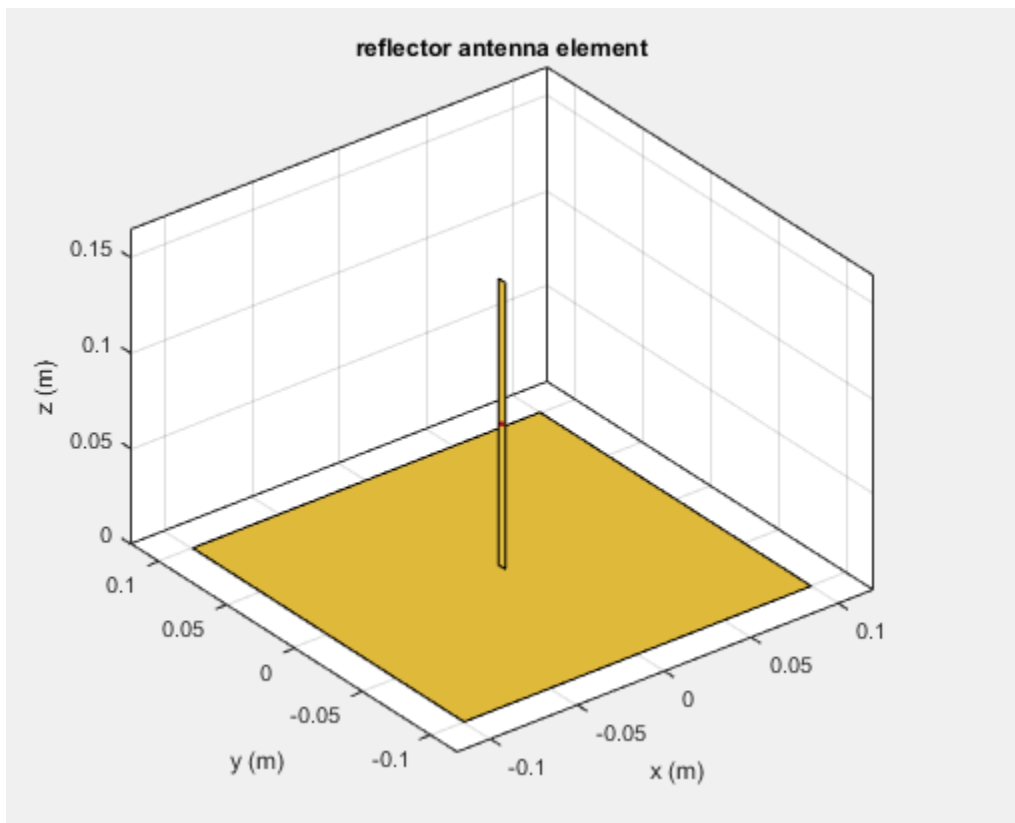
```
dipole with properties:
```

```

    Length: 0.1500
    Width: 0.0050
    FeedOffset: 0
    Tilt: 90
    TiltAxis: [0 1 0]
```

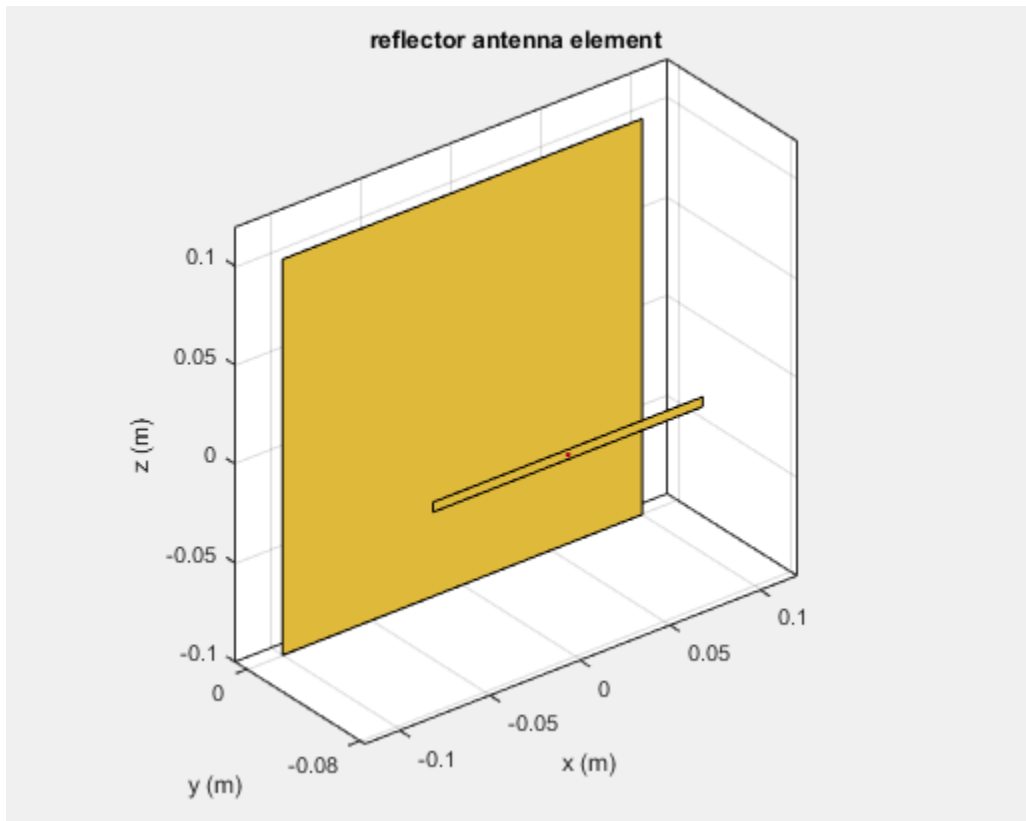
Based on the `Tilt` and `TiltAxis` properties the dipole element is rotated by 90 degrees about the y-axis and is parallel to the XY plane. To make the dipole perpendicular to the XY plane, change the `Tilt` property of the dipole to 0 degrees.

```
r.Exciter.Tilt = 0;
show(r)
```



Rotating the reflector element rotates the entire structure by the specified angle. To rotate the complete antenna, use the `Tilt` property of the reflector object. For example, rotate the reflector by 90 degrees about the X-axis. According to the right-hand rule, the reflector now lies in the X-Z plane with the dipole.

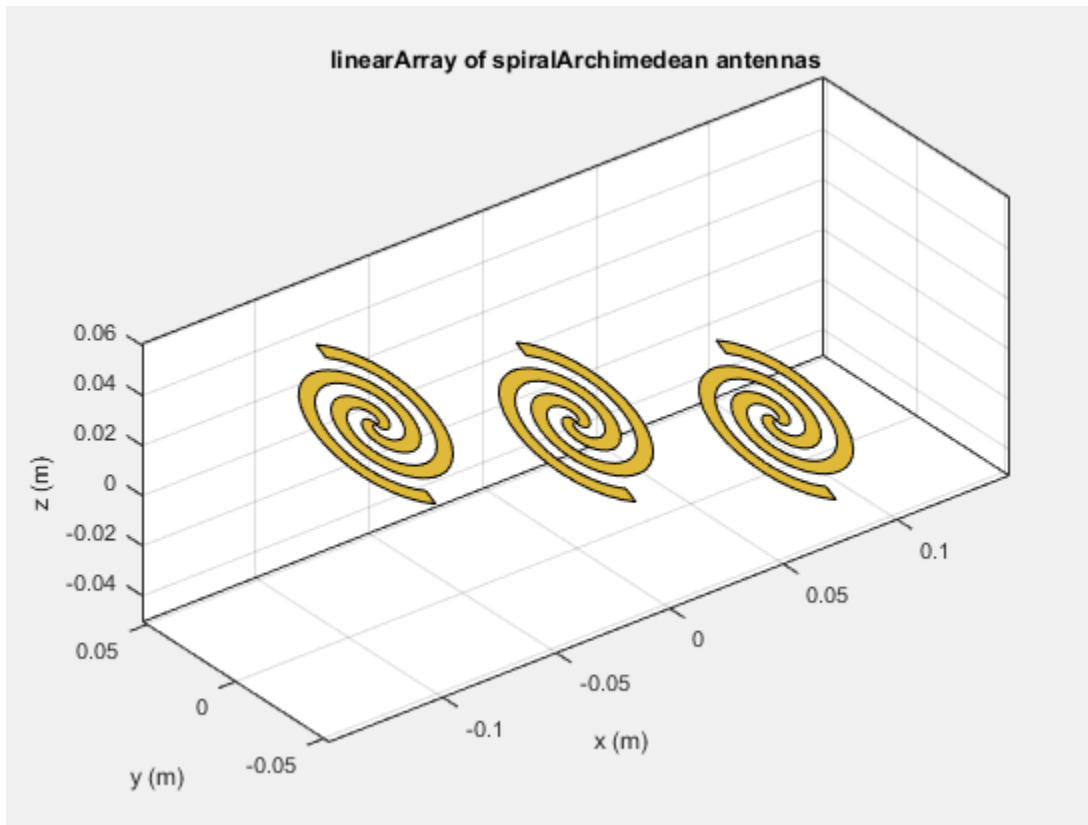
```
r = reflector;  
r.Tilt = 90;  
show(r)
```



Rotate Antenna Array

Create an array of Archimedean spirals with the whole array rotated at 30 degrees about the X-axis and Y-axis. First create one spiral antenna.

```
s = spiralArchimedean;
l = linearArray('Element',s,'ElementSpacing',0.1,...
'NumElements',3,'Tilt',30,'TiltAxis',[1 1 0]);
show(l)
```



See Also

More About

- "Antenna Classification"

Infinite Ground Plane

In this section...

"Image Theory" on page 1-39

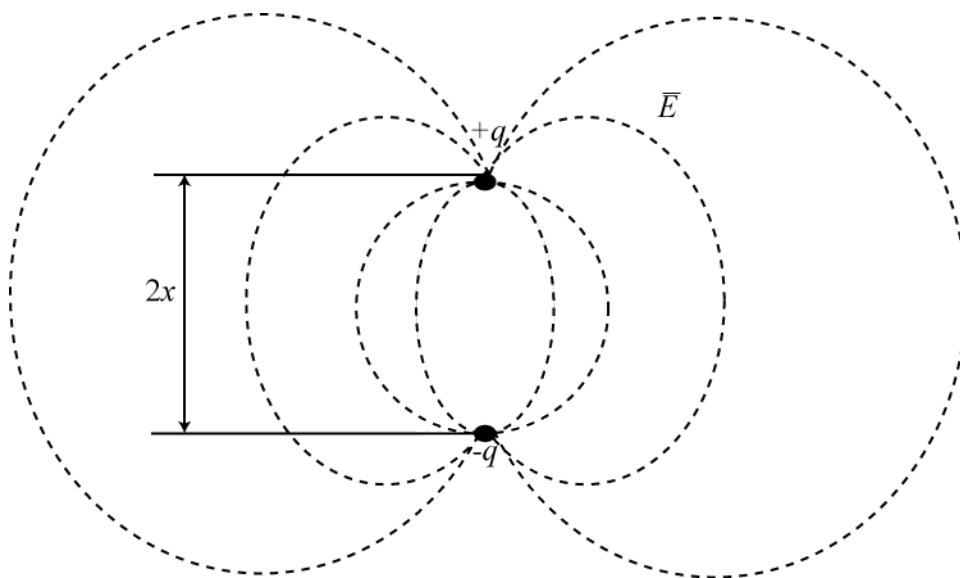
"Formulate the Image Theory Technique" on page 1-40

Antenna Toolbox library element uses the image theory technique to model an infinite ground plane. The main advantage of image theory technique is that you do not need to discretize the ground plane. Image theory reduces the overall size of the problem and you get a solution faster.

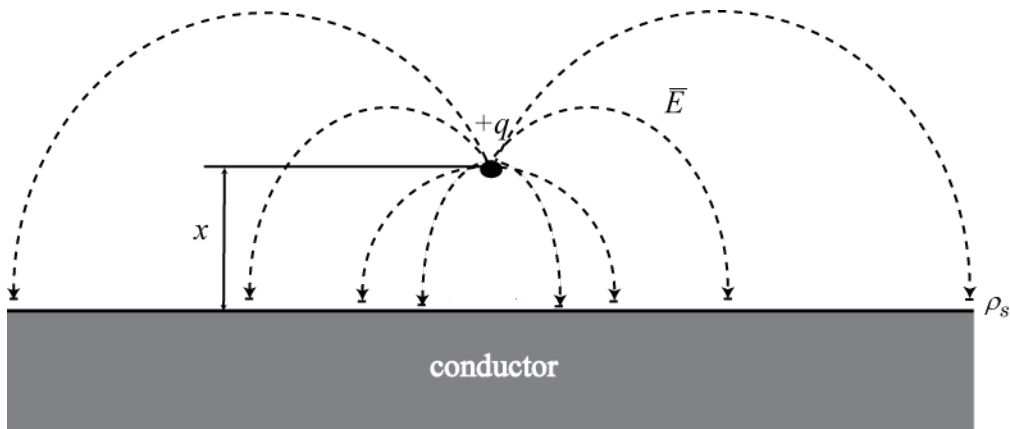
Image Theory

The image theory technique uses a catalog of electromagnetic problems that produce identical field distributions. You can identify these electromagnetic problems by noting that conducting surfaces are of constant potential. Placing these conducting surfaces along any equipotential lines in any field distributions does not alter the fields.

Consider a positive and negative charge placed a distance of $2x$ apart from each other. The equipotential surface for these two charges forms at a distance midway between them.



If you place a conducting object along this equipotential surface, then the field above the surface does not change.

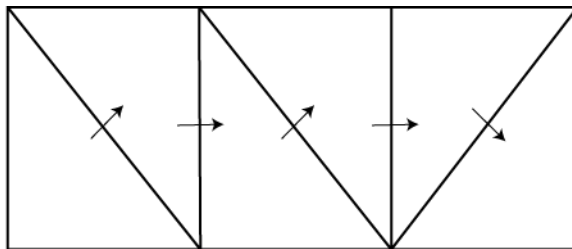


You can also apply this equivalence in reverse. Consider charges that are distance x apart from a conducting surface. You can replace the conducting surface with a set of image charges whose signs are opposite those of the original charge. Place these image charges at a distance x below the original conducting surface. This method eliminates the conducting plate, leaving only charges in unbound space. This equivalence is called the *method of images*.

Formulate the Image Theory Technique

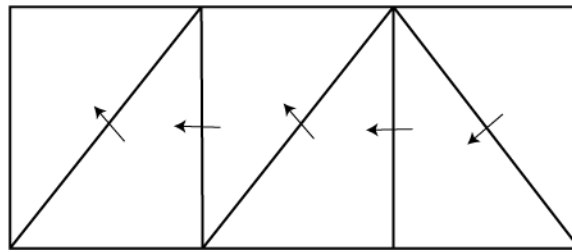
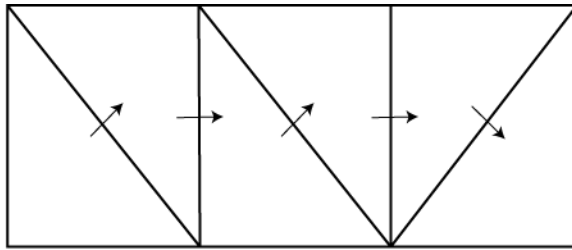
Consider a rectangular plate over an infinite ground plane. This rectangular plate is divided into a triangular mesh. The induced currents flow over the triangulated surface using the RWG basis functions.

Original Problem:



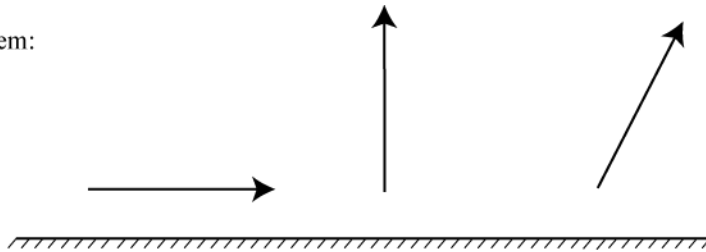
You formulate the problem by considering real and image basis functions. You can show the same problem using image theory or method of images. The interaction matrix is calculated using the interaction between real and image basis functions.

Equivalent Problem:

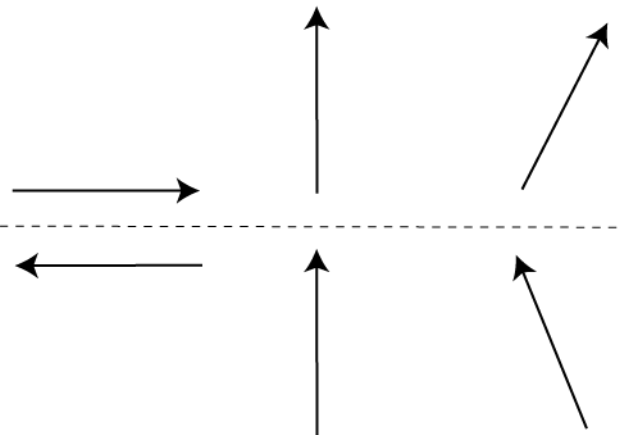


According to the method of images, the real and image electric currents are equal to each other in terms of magnitude. The parallel components are in opposite direction. The normal components are in the same direction.

Original Problem:



Equivalent Problem:



References

- [1] Balanis, C.A. *Antenna Theory. Analysis and Design*, 3rd Ed. New York: Wiley, 2005.
- [2] Yildirim, Cemal. "Scattering and Radiation problems of arbitrary shaped conducting bodies above ground plane" Ph.D. Thesis. Bilkent University, Ankara, Turkey, October, 2002

See Also

More About

- "Method of Moments Solver for Metal Structures" on page 3-2

Model Infinite Ground Plane for Balanced Antennas

This example shows how to model an infinite ground plane and calculate fundamental antenna parameters for balanced antennas.

Create Antenna On Infinite Ground Plane

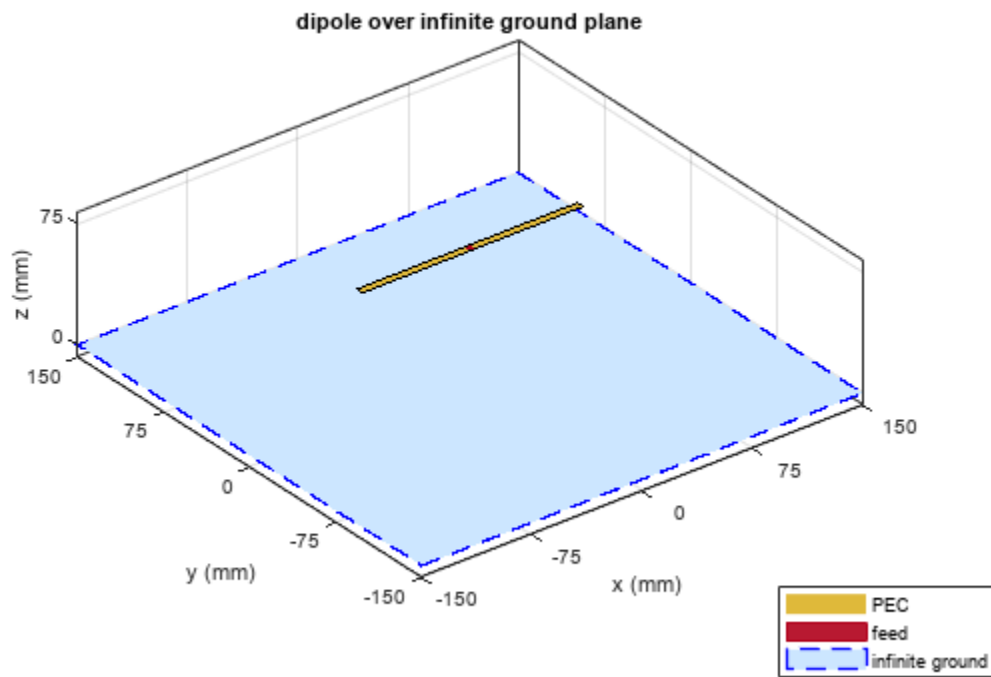
Create a reflector antenna on a ground plane of infinite length.

```
r = reflector ('GroundPlaneLength', inf)
r =
  reflector with properties:
      Exciter: [1x1 dipole]
      Substrate: [1x1 dielectric]
      GroundPlaneLength: Inf
      GroundPlaneWidth: 0.2000
      Spacing: 0.0750
      EnableProbeFeed: 0
      Conductor: [1x1 metal]
      Tilt: 0
      TiltAxis: [1 0 0]
      Load: [1x1 lumpedElement]
```

View Antenna Geometry

View the physical construction of infinite ground plane reflector antenna.

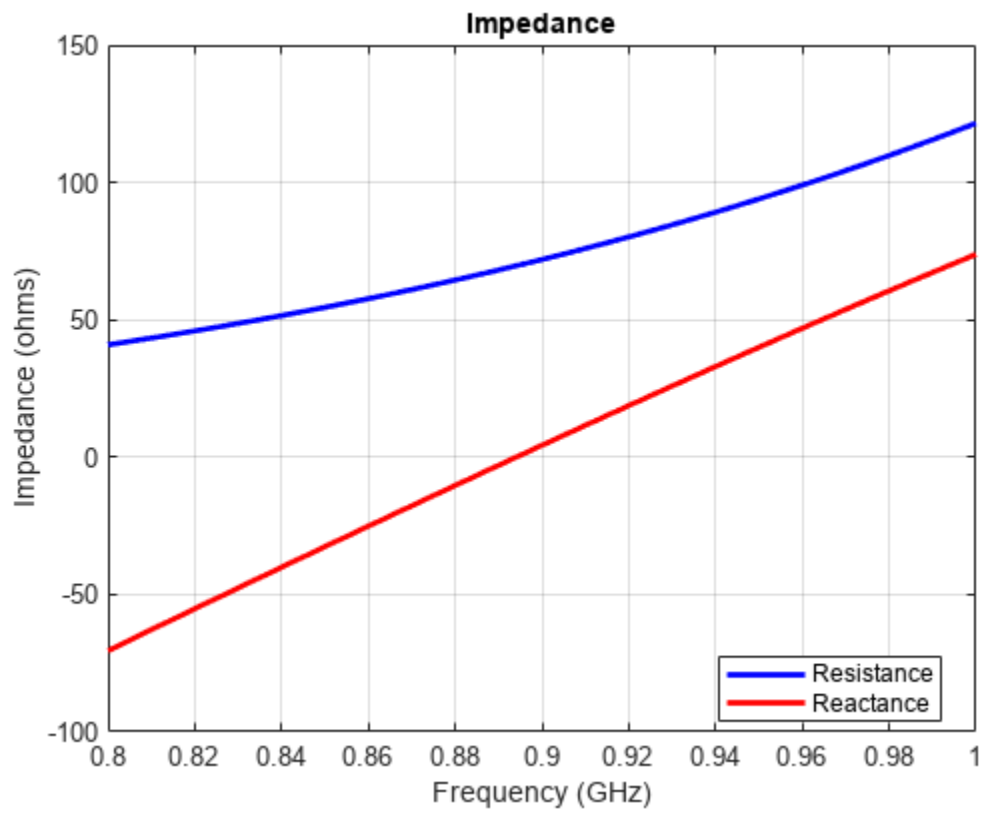
```
figure;show(r);
```



Calculate Impedance of Antenna

Calculate the impedance of reflector antenna over a frequency range of 800MHz to 1GHz.

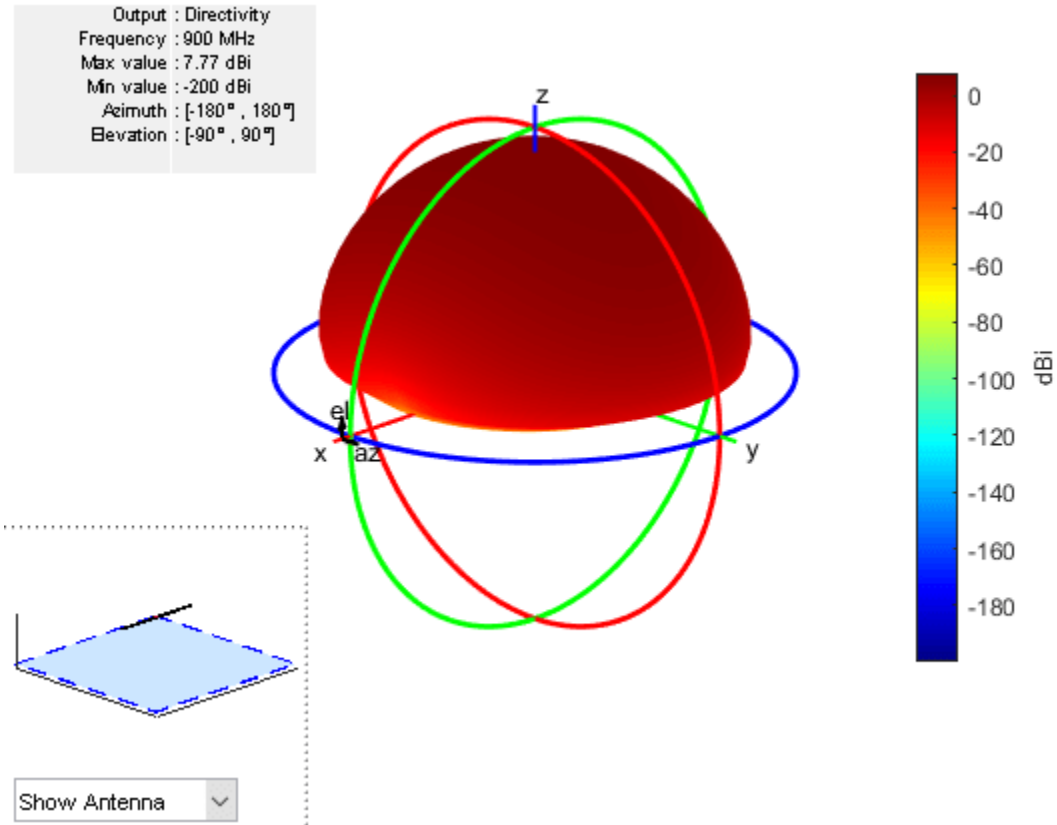
```
figure;impedance(r,800e6:5e6:1e9);
```



Plot Radiation Pattern of Antenna

Plot the radiation pattern of reflector antenna at a frequency of 900MHz.

```
figure;pattern(r,900e6);
```

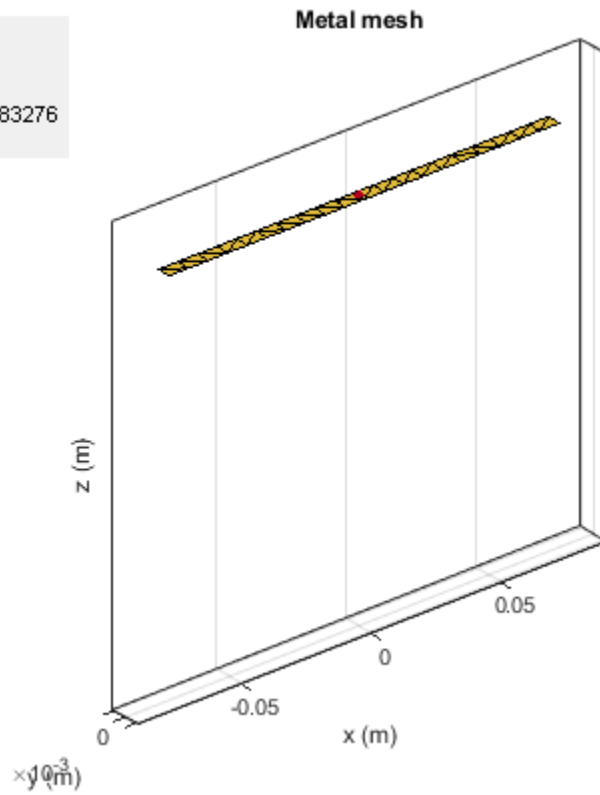


Visualize Antenna Mesh

Mesh and view the infinite ground plane reflector antenna.

```
figure; mesh(r);
```


NumTriangles: 40
NumTetrahedra: 0
NumBasis: 39
MaxEdgeLength: 0.0083276
MeshMode: auto



See Also

"Reflector Backed Equiangular Spiral" on page 5-129

Model Infinite Ground Plane for Unbalanced Antennas

This example shows how to model an infinite ground plane and calculate fundamental antenna parameters for unbalanced antennas.

Create Antenna on Infinite Ground Plane

Create a patch microstrip antenna on a ground plane of infinite length.

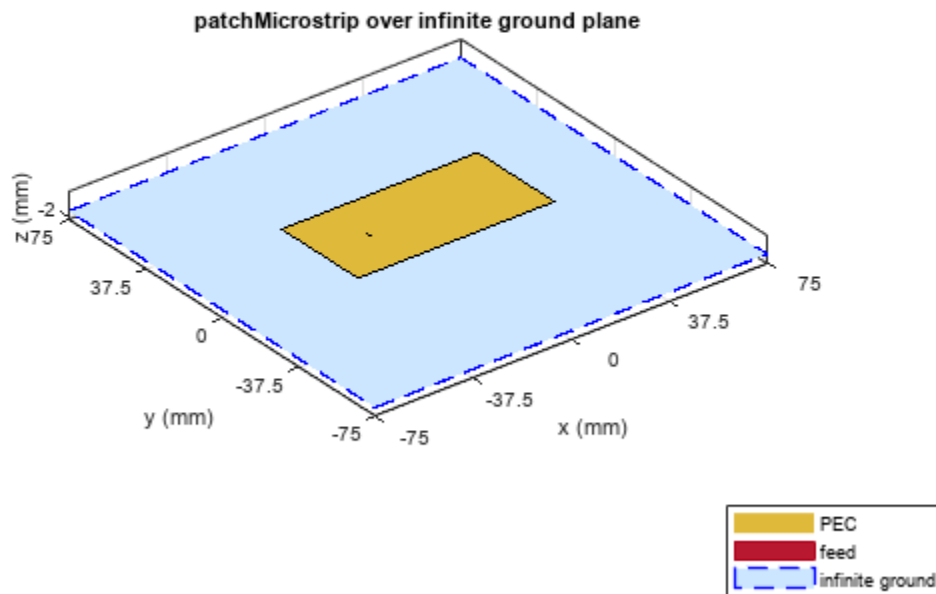
```
p = patchMicrostrip('GroundPlaneLength', inf)
```

```
p =  
patchMicrostrip with properties:  
  
    Length: 0.0750  
    Width: 0.0375  
    Height: 0.0060  
    Substrate: [1x1 dielectric]  
GroundPlaneLength: Inf  
GroundPlaneWidth: 0.0750  
PatchCenterOffset: [0 0]  
    FeedOffset: [-0.0187 0]  
    Conductor: [1x1 metal]  
    Tilt: 0  
    TiltAxis: [1 0 0]  
    Load: [1x1 lumpedElement]
```

View Antenna Geometry

View the physical construction of the patch microstrip antenna using infinite ground plane.

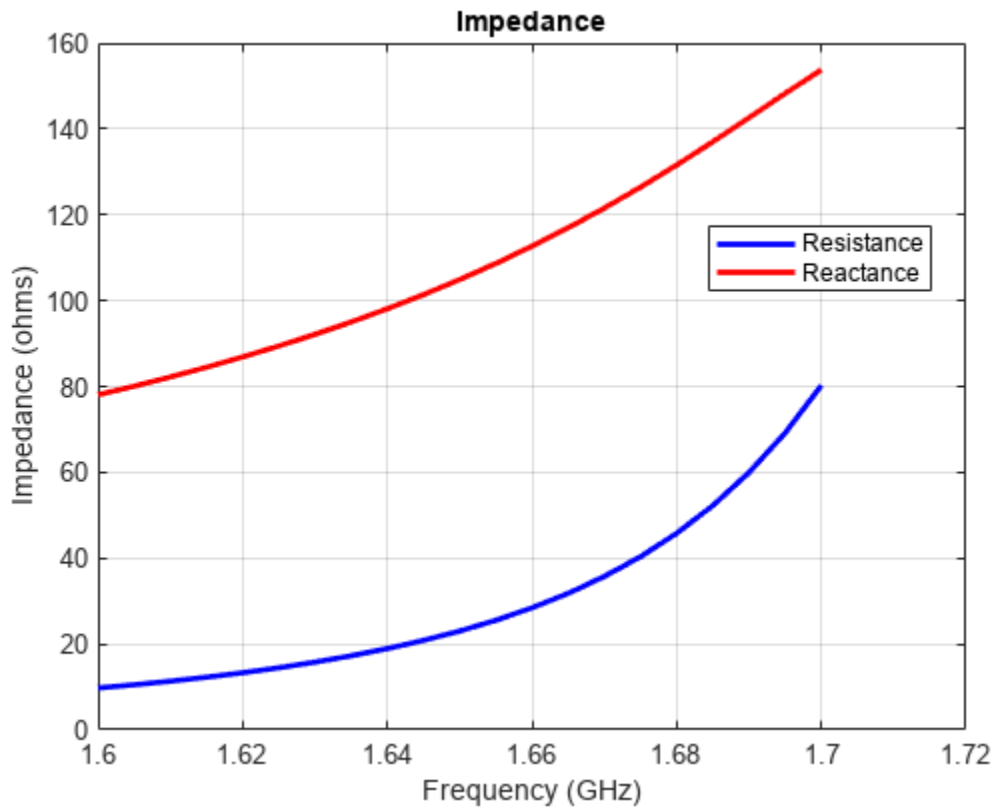
```
figure;  
show(p);
```



Calculate Impedance of Antenna

Calculate the impedance of the antenna over a frequency range of 1.60 GHz to 1.70 GHz.

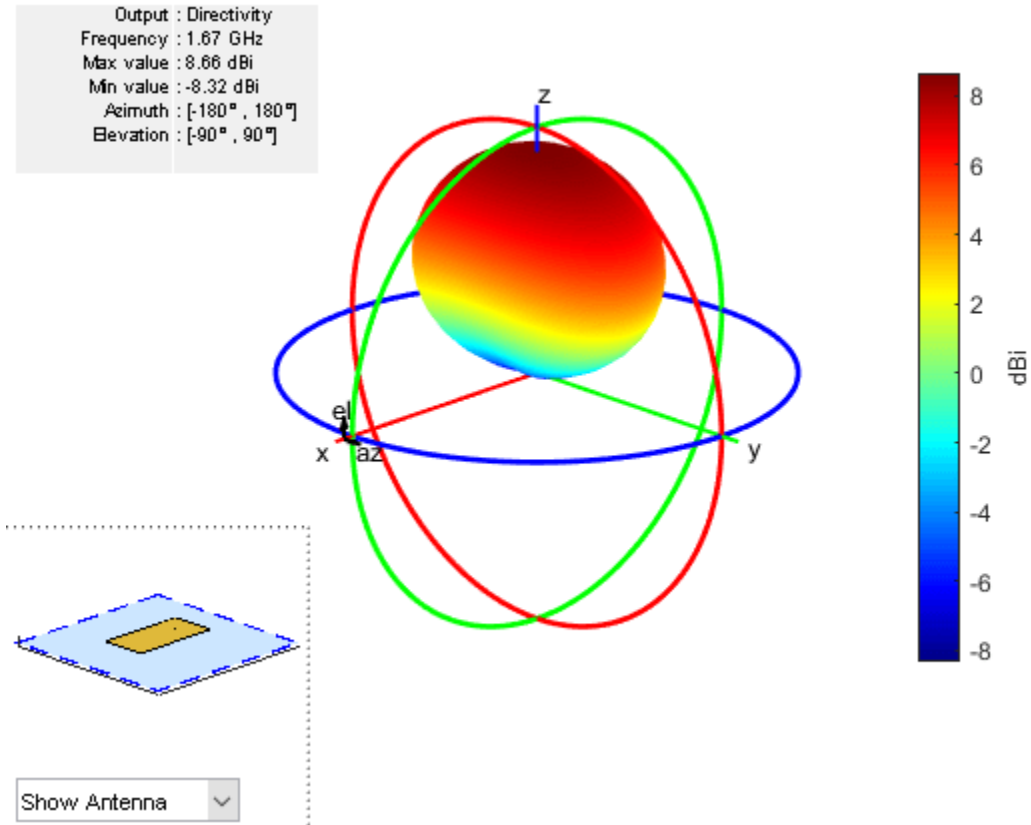
```
figure;  
impedance(p,1.60e9:5e6:1.70e9);
```



Plot Radiation Pattern of Antenna

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

```
figure;  
pattern(p,1.67e9);
```

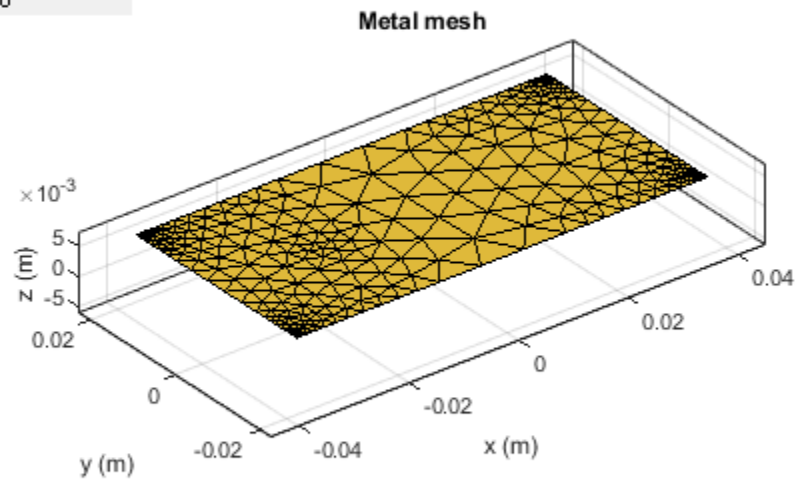


Visualize Antenna Mesh

Mesh and view the infinite ground plane patch microstrip antenna.

```
figure;  
mesh(p);
```

NumTriangles: 544
NumTetrahedra: 0
NumBasis: 763
MaxEdgeLength: 0.0096451
MeshMode: auto



See Also

“Design and Analyze Curved Reflectors” on page 5-705

Electrical Length of Antenna

The physical length of an antenna is the actual physical dimensions of the antenna. Consider a dipole antenna from the Antenna Toolbox with an operating frequency of 75 MHz and corresponding wavelength of 4 m. The dipole object creates a half-wavelength dipole antenna, hence its electrical length is $\lambda/2$ or 2 m, which is equal to the physical length of the antenna.

The electrical length of an antenna is βl

where: $\beta = \frac{2\pi}{\lambda} \sqrt{\mu\epsilon}$ is the propagation constant dependent on the wavelength λ , permittivity ϵ , and permeability μ .

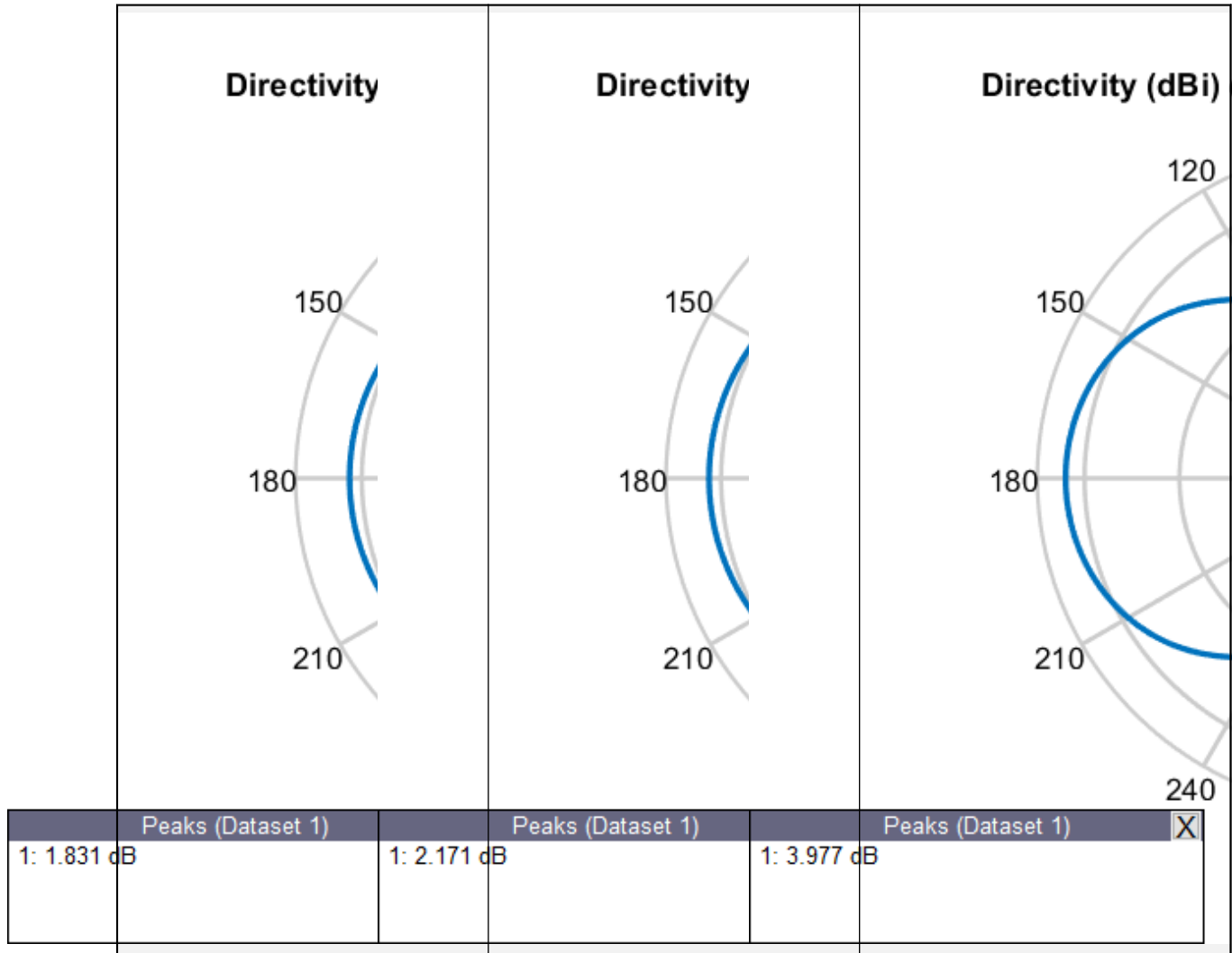
l is the physical length of the antenna.

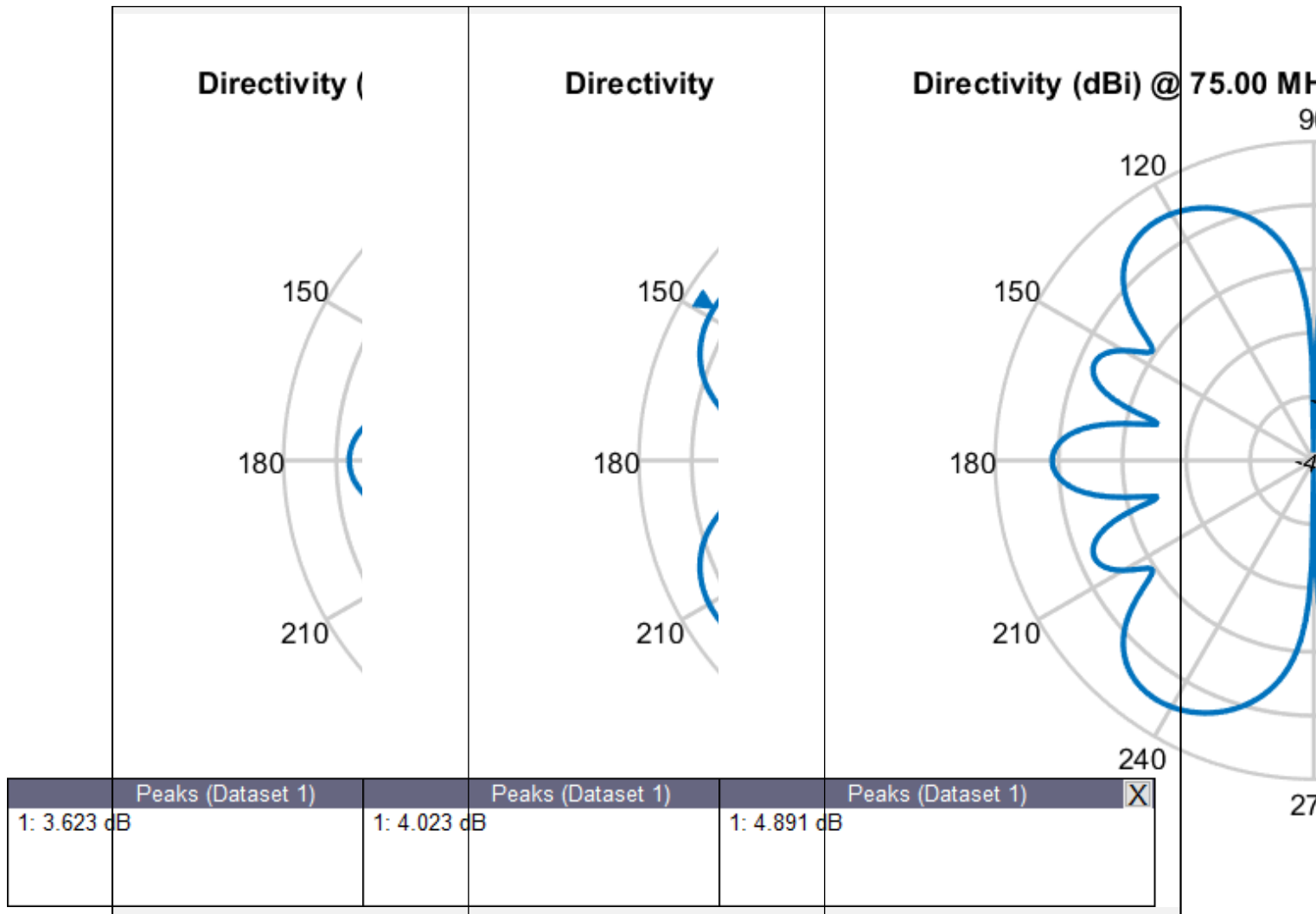
You can change the electrical length by changing the physical length and the propagation constant. The propagation constant depends on permittivity, permeability of antenna materials, and operating frequency.

Elevation Patterns of Dipole Antennas

Observe gain of the dipole antennas of various physical lengths at a frequency of 75 MHz in the table below.

Elevation Patterns of Dipole Antennas of Various Physical Lengths at Frequency of 75 MHz

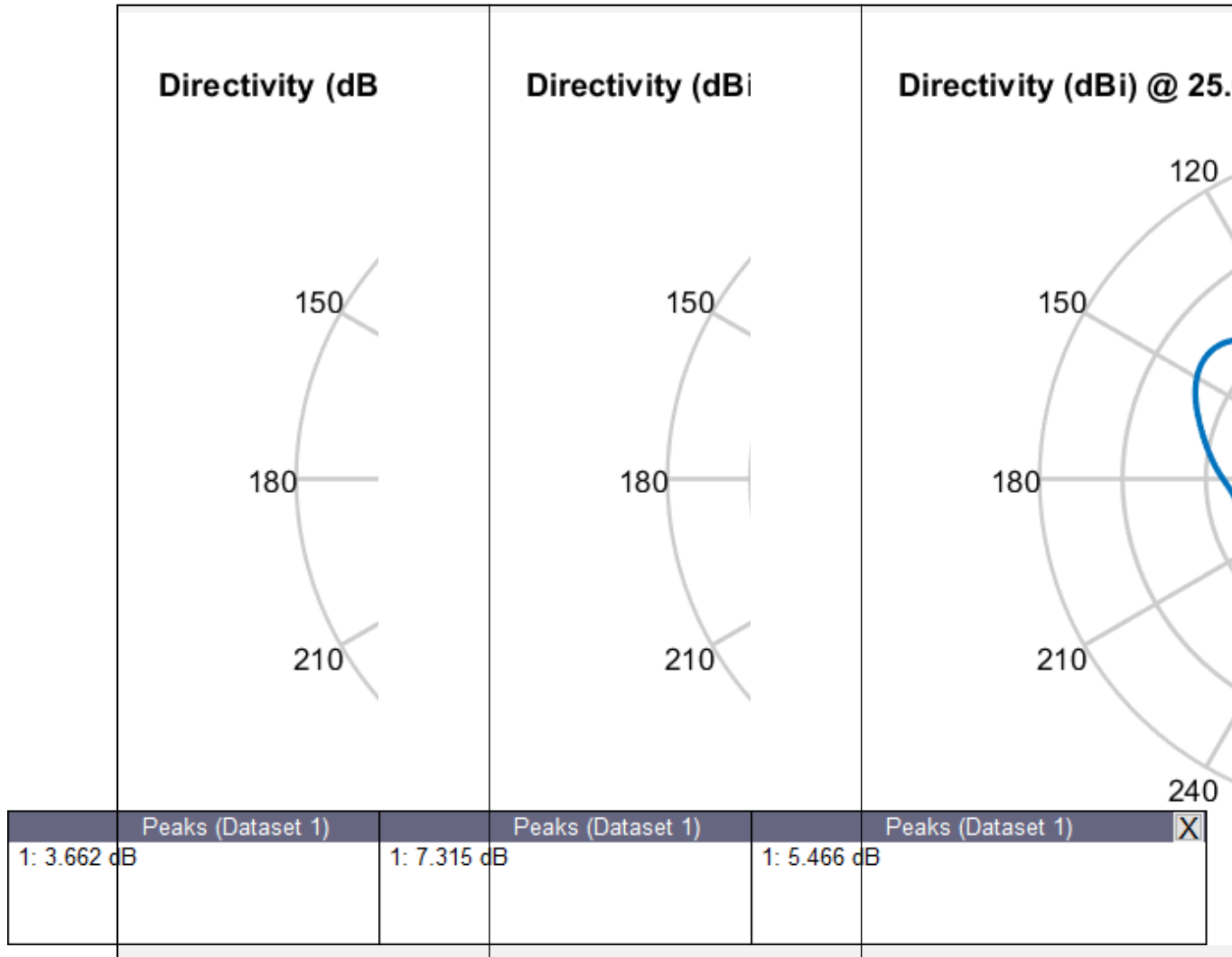




Elevation Patterns of Rectangular Waveguides at Different Frequencies

Design a rectangular WR-90 waveguide with default length of 1 Lambda at an operating frequency of 12.5 GHz. Examine its elevation pattern. Vary the operating frequency and electrical length of the antenna and examine how the elevation pattern changes.

Elevation Patterns of Waveguides of Various Electrical Lengths



Effect of Materials and Reactance on Electrical Length

Effect of Materials

Increasing the relative permittivity or dielectric constant of a material increases the propagation constant. Similarly, increasing the relative permeability of a material also increases the propagation constant. Design two microstrip patch antennas, one with air as the substrate and another with a FR4 substrate. Analyze their dimensions at the same operating frequency.

Compare Dimensions of Microstrip Patch Antennas with Different Substrates

Design a microstrip patch antenna with air as a substrate at a frequency of 1 GHz.

```
design(patchMicrostrip, 1e9)
```

```
ans =
  patchMicrostrip with properties:

        Length: 0.1439
        Width: 0.1874
        Height: 0.0030
        Substrate: [1x1 dielectric]
  GroundPlaneLength: 0.2998
  GroundPlaneWidth: 0.2998
  PatchCenterOffset: [0 0]
    FeedOffset: [0.0303 0]
      Conductor: [1x1 metal]
        Tilt: 0
    TiltAxis: [1 0 0]
      Load: [1x1 lumpedElement]
```

Design a microstrip patch antenna with FR4 as a substrate at a frequency of 1 GHz.

```
design(patchMicrostrip('Substrate',dielectric('FR4')),1e9)
```

```
ans =
  patchMicrostrip with properties:

        Length: 0.0664
        Width: 0.0855
        Height: 0.0014
        Substrate: [1x1 dielectric]
  GroundPlaneLength: 0.1368
  GroundPlaneWidth: 0.1368
  PatchCenterOffset: [0 0]
    FeedOffset: [0.0140 0]
      Conductor: [1x1 metal]
        Tilt: 0
    TiltAxis: [1 0 0]
      Load: [1x1 lumpedElement]
```

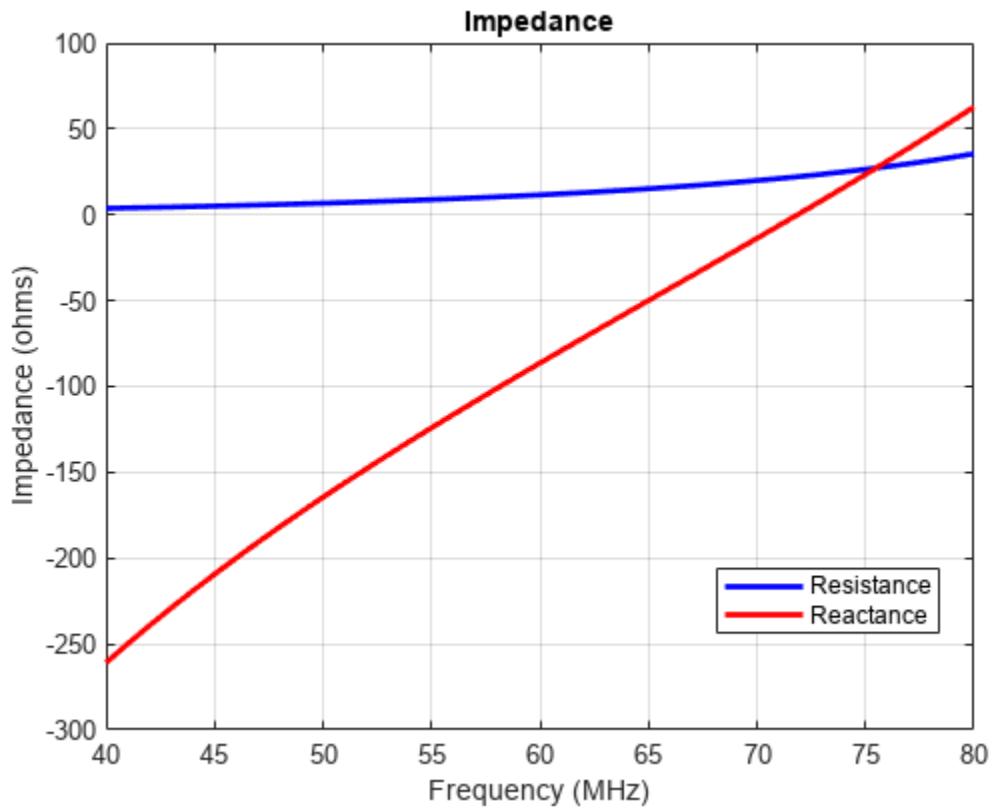
Effect of Antenna Loading

The effective electrical length of an antenna can be changed without changing its physical length by adding reactance, (inductance or capacitance) in series with the antenna. This is called lumped-impedance matching or loading. For example, consider a monopole antenna and a monopole antenna with a top hat of the same physical length. The monopole antenna with a top hat resonates at a lower frequency as compared to the monopole antenna without top hat. Hence, the corresponding wavelength for monopole with a top hat is higher and, its electrical length is larger.

Compare Impedance of Monopole and Top Hat Monopole Antennas

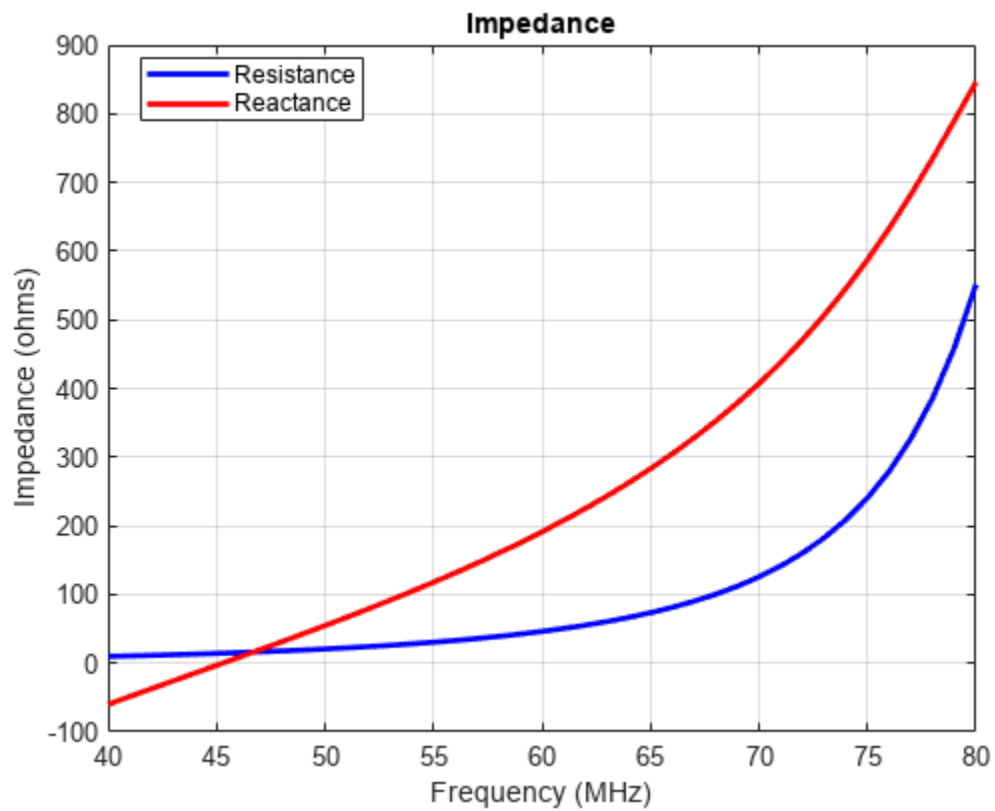
Calculate and plot the impedance of a monopole antenna over a frequency range of 40 MHz-80 MHz.

```
figure;
impedance(monopole,linspace(40e6,80e6,41))
```



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

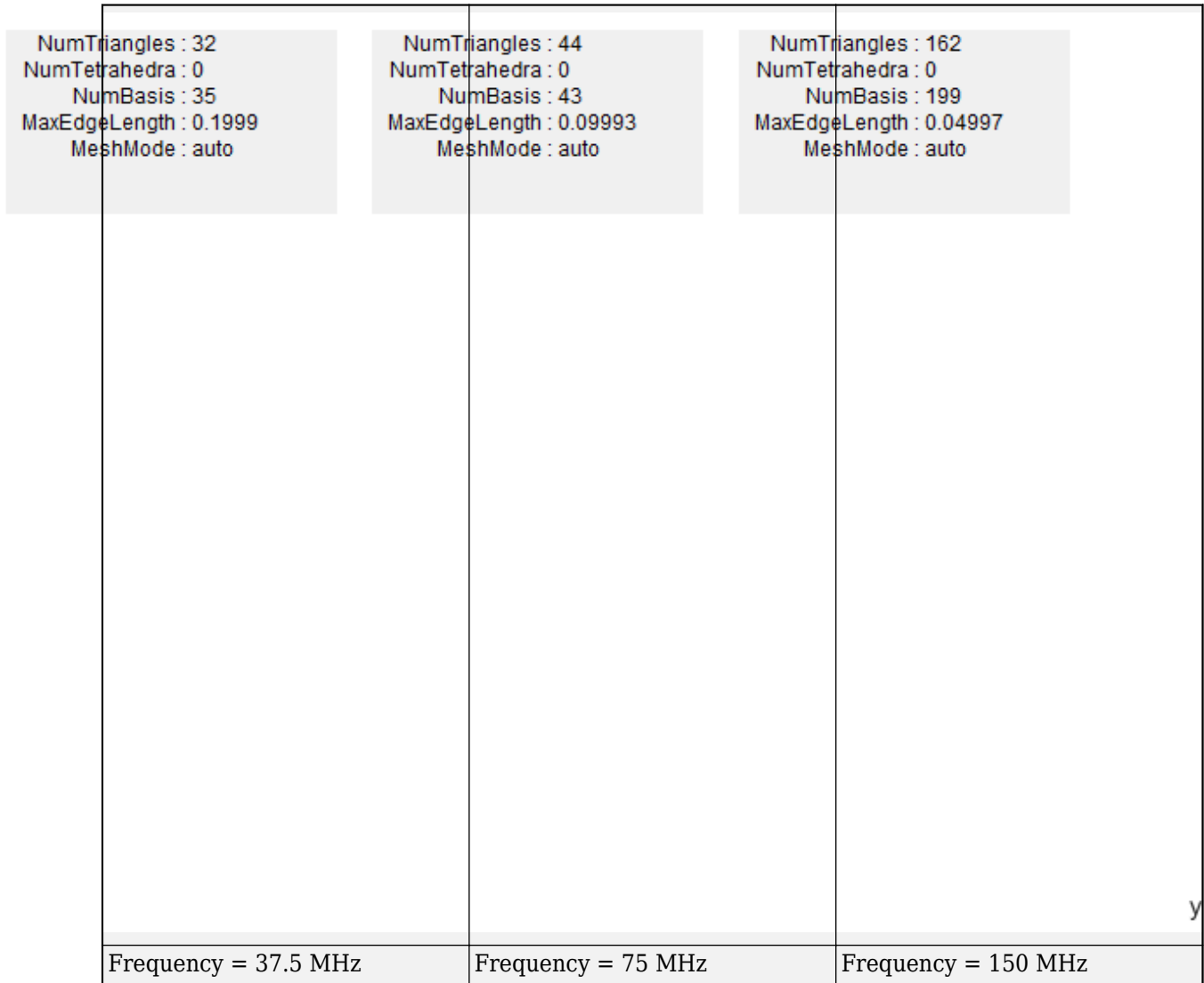
```
figure;  
impedance(monopoleTopHat, linspace(40e6, 80e6, 41))
```



Mesh at Various Frequencies

Observe meshing of the dipole antennas of same physical lengths at frequencies of 37.5, 75, and 150 MHz in the table below.

Meshing of Dipole Antenna at Various Frequencies

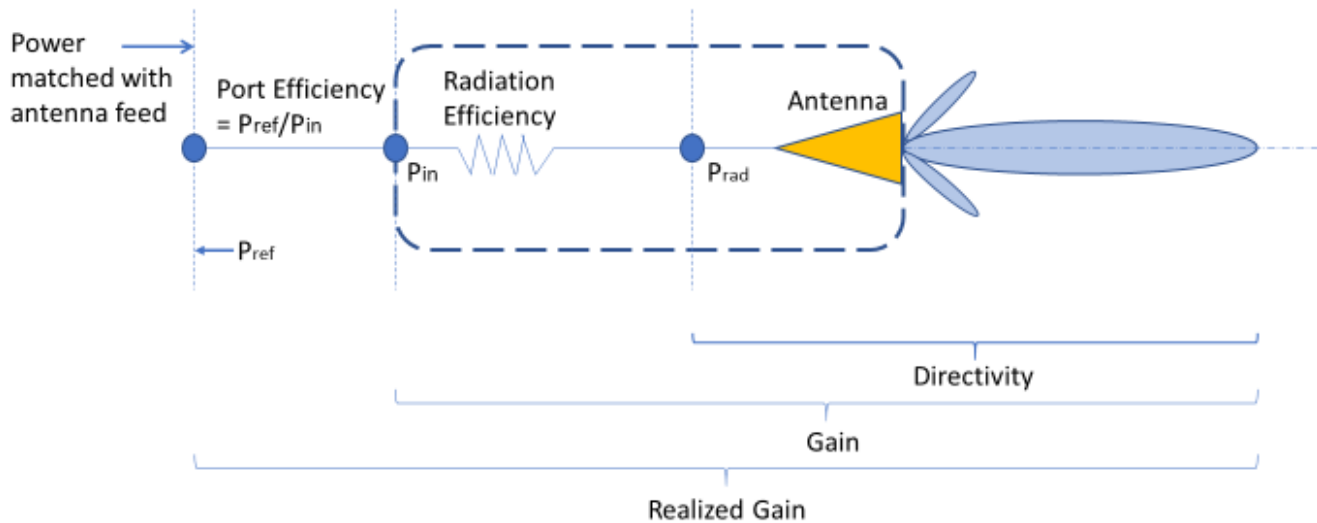


The antenna lengths stay the same, but as the operating frequency of the antenna changes, the number of triangles in the mesh change as do the solutions for the analysis functions.

References

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

Far-field Terminologies



Directivity

Directivity is the ability of an antenna or array to radiate power in a particular direction. It can be defined as ratio of maximum radiation intensity in the desired direction to the average radiation intensity over all the directions. The equation for directivity is:

$$D = \frac{4\pi U(\theta, \phi)}{P_{rad}}$$

where:

- D is the directivity of the antenna.
- U is the radiation intensity of the antenna as a function of direction.
- P_{rad} is the average radiated power of the antenna over all the directions.

Antenna directivity is dimensionless and is calculated in decibels compared to the isotropic radiator (dBi).

Directivity Value of Helix Antenna.

Calculate the directivity of a helix antenna.

```
h = helix;
D = pattern(h, 2e9, 0, 1:1:360);
```

Showing the first five directivity values.

```
Dnew = D(1:5)
```

```
Dnew = 5×1
```

-6.3885
-6.1747
-5.9472
-5.7081
-5.4590

Gain

The *gain* of an antenna or array depends on the directivity and efficiency of the antenna or array. It can be defined as the ratio of maximum radiation intensity in the desired direction to the total power input of the antenna. The equation for gain of an antenna is:

$$G = \frac{4\pi U(\theta, \phi)}{P_{in}}$$

where:

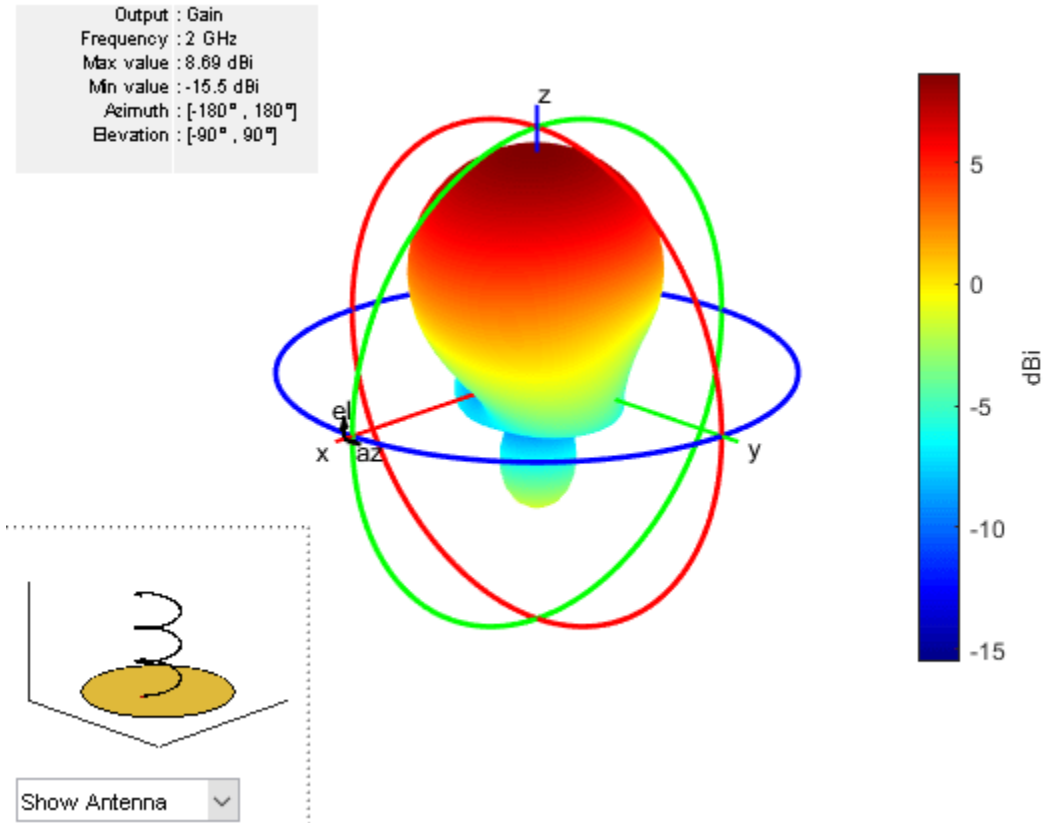
- G is the gain of the antenna.
- U is the radiation intensity of the antenna.
- P_{in} is the total power input to the antenna.

If the efficiency of the antenna in the desired direction is 100%, then the total power input to the antenna is equal to the total power radiated by the antenna, that is, $P_{in} = P_{rad}$. In this case, the antenna directivity is equal to the antenna gain.

Gain of Helix Antenna

Calculate the gain of the helix antenna.

```
h = helix;  
pattern(h, 2e9, 'Type', "gain")
```

Realized Gain

The realized gain or absolute gain of an antenna or array takes into account reflection (mismatch) losses. The port mismatch losses arise due to the difference in antenna input impedance and matching impedance. In figure above,

- P_{in} is the input power to the antenna.
- P_{rad} is the radiated power of the antenna.
- P_{ref} is the reflected power from antenna due to the port mismatch.

This figure shows relation between gain, realized gain, and directivity as follows:

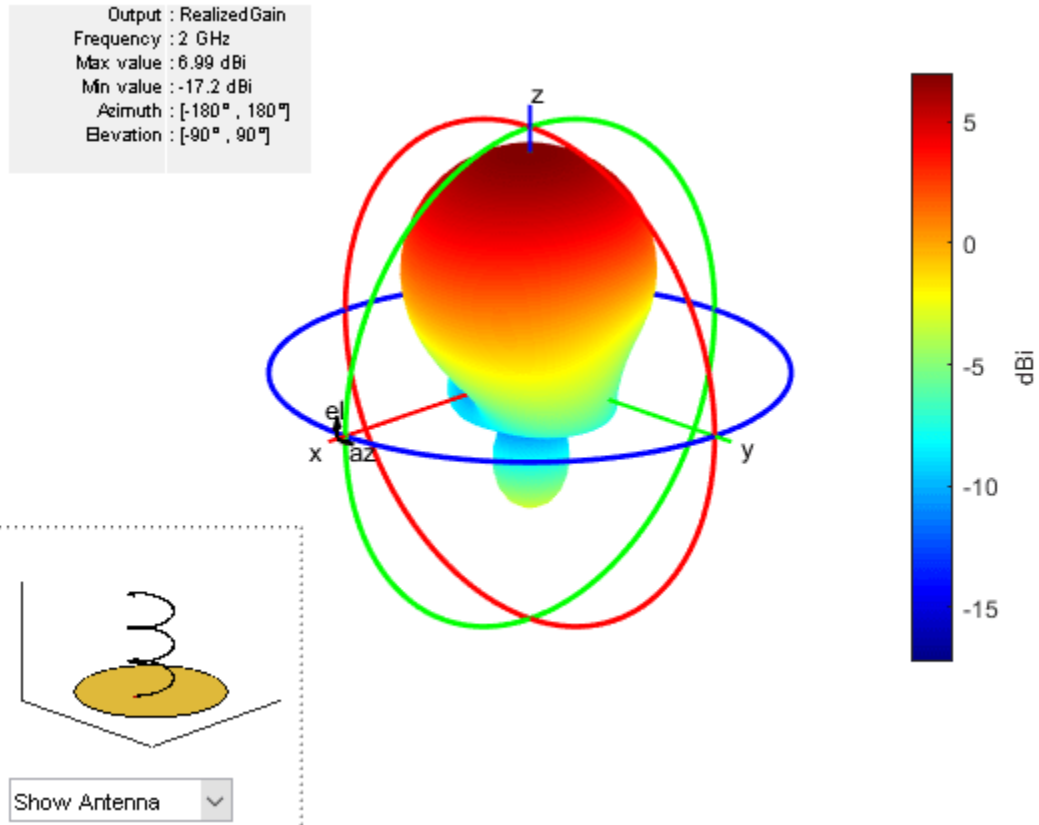
Realized Gain = (Port efficiency) * (Radiation efficiency) * (Gain) = (Port efficiency) * (Directivity)

Port efficiency = P_{ref} / P_{in}

Realized Gain of Helix Antenna

Calculate the realized gain of the helix antenna.

```
h = helix;
pattern(h,2e9, 'Type', "realizedgain");
```



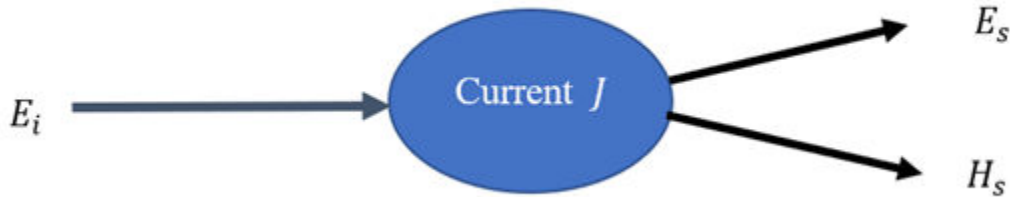
References

- [1] Makarov, Sergey N. *Antenna and EM Modeling in MATLAB*. Chapter 3, Sec 3.4-3.8. Wiley Inter-Science.
- [2] Balanis, C.A. *Antenna Theory, Analysis and Design*, Chapter 2, sec 2.3-2.6, Wiley.

Field Calculation in Antennas

Field Calculation in Metal Antennas

The electromagnetic analysis of an antenna or array starts from determining its equivalent current density. You can compute the current density using the field boundary conditions on the antenna's surface or volume.



Incident excitation field E_i induces equivalent current J to produce scattered fields E_s and H_s .

Use the method-of-moments (MoM) technique described in “Method of Moments Solver for Metal Structures” on page 3-2 to calculate the equivalent current density for metal only antennas. For the MoM technique, you can express the total current density as a weighted linear combination of N number basis functions (f_n^M) as

$$J = \sum_{n=1}^N I_n f_n^M$$

Where the weighting coefficient of the n^{th} basis function is I_n . After calculating the current the current density vector J , compute the scattered electric (E_s) and magnetic (H_s) fields

$$E_s = E_s^M = \frac{-j\omega\mu_0}{4\pi} \sum_{n=1}^{N_M} I_n \int_S f_n^M(r') G(r, r') ds' - \frac{1}{4\pi\omega\epsilon_0} \sum_{n=1}^{N_M} I_n \int_S \nabla f_n^M(r') \nabla_r G(r, r') ds$$

The curl of the magnetic vector potential gives the scattered magnetic field of the metal antenna

$$H_s = H_s^M = -\frac{1}{4\pi} \sum_{n=1}^{N_M} I_n \int_S f_n^M(r') \times \nabla_r G(r, r') ds$$

where the location of the source current density is r' and the observation location is r . The free-space Green's function is

$$G(r, r') = \frac{e^{-jk|r-r'|}}{|r-r'|}$$

where the free-space propagation constant is k . The observation location can be either in the near field region or in the far-field region with respect to the source current location. For a certain observation location $r = [r_x, r_y, r_z]$ each of the electric field E_s^M and magnetic field H_s^M is a 3-by-1 vector with components along three orthogonal co-ordinate axes. If the observation location is in the far-field region, the radial components of the electric and magnetic fields are negligible.

Field Calculation in Metal-Dielectric Antenna

For a metal-dielectric antenna, the total scattered electric and magnetic fields are a superposition of the fields (E_s^M, H_s^M) due to the metal surface current and the fields (E_s^D, H_s^D) due to volume polarization current of the dielectric. In "Method of Moments Solver for Metal and Dielectric Structures" on page 3-9, the fields due to the dielectric current are given by:

$$E_s^D = \frac{\omega^2 \mu_0}{4\pi} \sum_{n=1}^{N_T} D_n \sum_{p=1}^P K_p \int_{V_p} G(r, r') f_{np}(r') dr' - \frac{1}{4\pi \epsilon_0} \sum_{n=1}^{N_T} D_n \sum_{q=1}^Q \hat{K}_q \int_{\Omega_q} \nabla_r G(r, r') f$$

$$H_s^D = -\frac{j\omega}{4\pi} \sum_{n=1}^{N_T} D_n \sum_{p=1}^P K_p \int_{V_p} (f_{np}(r') \times \nabla_r G(r, r')) dr'$$

So the total scattered field from metal to dielectric is:

$$E_s = E_s^M + E_s^D$$

$$H_s = H_s^M + H_s^D$$

Field Parameters

The electric and magnetic fields at a near or far-field location is determined using the EHfields function.

Directivity

Radiation intensity of the antenna for a far-field sphere of radius R is

$$U(\theta, \phi) = \frac{R}{2\eta_0} |E_s(R, \theta, \phi)|^2$$

By default, $R = 100\lambda$ is considered in Antenna Toolbox for pattern calculation. The free-space wavelength at minimum frequency is λ .

For a far-field location, the default electric field is a 3-by-1 vector with Cartesian coordinates of:

$$E_s(R, \theta, \phi) = \begin{pmatrix} E_{xr} + j * E_{xi} \\ E_{yr} + j * E_{yi} \\ E_{zr} + j * E_{zi} \end{pmatrix} = \begin{pmatrix} E_x \\ E_y \\ E_z \end{pmatrix}$$

r and i subscripts denote the real and imaginary parts of the X-, Y-, and Z-components of the electric field.

The absolute magnitude of the $E_s(R, \theta, \phi)$ is:

$$E_s(R, \theta, \phi) = |E_s(R, \theta, \phi)|_2$$

Total radiated power is:

$$P_{rad} = \int_0^\pi \int_0^{2\pi} U(\theta, \phi) \sin\theta d\theta d\phi$$

The directivity of an antenna is:

$$D(\theta, \phi) = \frac{4\pi U(\theta, \phi)}{P_{rad}}$$

Gain

If the antenna has finite amount of conduction loss and dielectric loss, the total radiated power is:

$$P_{rad} = P_{in} - P_{loss}$$

where the resistive power is P_{in} and the power dissipation due to finite conductivity and non-zero loss tangent of the dielectric is P_{loss} . The radiation efficiency of the antenna is:

$$\eta = \frac{P_{rad}}{P_{in}}$$

The gain of the antenna is defined as:

$$G(\theta, \phi) = \eta D(\theta, \phi) = \eta \frac{4\pi U(\theta, \phi)}{P_{rad}}$$

Use efficiency to determine the radiation efficiency of an antenna or an array.

The equivalent isotropically radiated power or effective isotropically radiated power (EIRP) is an important parameter to characterize antenna's radiation performance. The IEEE definition of EIRP in a given direction (θ, ϕ) is:

$$EIRP(\theta, \phi) = G(\theta, \phi) * P_{in} = 4\pi U(\theta, \phi)$$

Realized Gain

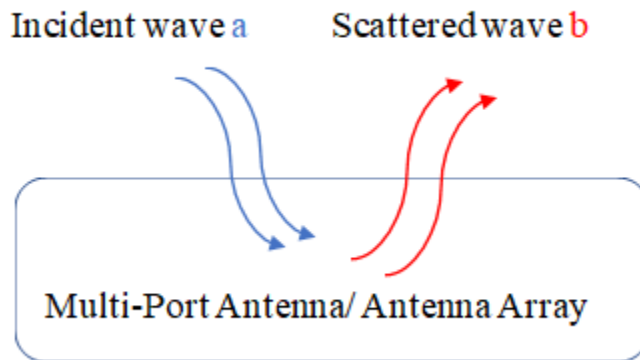
If a single feed antenna is connected to an excitation port impedance of Z_L and its corresponding scattering parameter is S_{11} , the realized gain of the antenna is:

$$G(\theta, \phi) = (1 - |S_{11}|^2)G(\theta, \phi) = (1 - |S_{11}|^2)\eta \frac{4\pi U(\theta, \phi)}{P_{rad}}$$

From the previous relations of different pattern functions, the directivity does not consider any loss. The material loss is considered in the gain. The realized gain considers both the material loss and port mismatch loss.

For single port antenna, the port mismatch is characterized by the port efficiency parameter:

$$PE = (1 - |S_{11}|^2)$$



For multi-port antenna and arrays, the port efficiency is:

$$PE = \frac{(a^H a - b^H b)}{(a^H a)} = 1 - \frac{(b^H b)}{(a^H a)}$$

In multiple input multiple output antenna (MIMO) array, total active reflection coefficient (TARC) parameter is used as an equivalent to S_{11} parameter of a single port antenna. The TARC is mathematically defined in 4.

$$TARC = \sqrt{\left| \frac{b^H b}{a^H a} \right|} = \sqrt{\frac{\sum_{m=1}^M |b_m|^2}{\sum_{n=1}^N |a_n|^2}}$$

Consider $[S]$ as the scattering matrix of a multi-port network, the port efficiency is:

$$PE = \frac{a^H ([I] - [S])^H [S] a}{a^H a} = 1 - TARC^2$$

where $[I]$ is an identity matrix with the same dimension of $[S]$. The amplitude taper and phase taper of the n^{th} excitation port is $|a_n|$ and $\angle a_n$.

The default electric field $E_s(R, \theta, \phi)$ is a 3-by-1 complex vector. The absolute magnitude of $E_s(R, \theta, \phi)$ is:

$$E_s^{abs}(R, \theta, \phi) = |E_s(R, \theta, \phi)|_2$$

The phase computation of pattern requires specification of polarization. Based on the polarization the electric fields are:

- Vertical polarization the electric field is $\tilde{E}_s(R, \theta, \phi) = E_\theta$
- Horizontal polarization the electric field is $\tilde{E}_s(R, \theta, \phi) = E_\phi$
- Right-hand circular polarization electric field is $\tilde{E}_s(R, \theta, \phi) = \frac{(E_\theta + jE_\phi)}{\sqrt{2}}$
- Left-hand circular polarization electric field is $\tilde{E}_s(R, \theta, \phi) = \frac{(E_\theta - jE_\phi)}{\sqrt{2}}$

E_θ and E_ϕ are the components of the electric field along the θ and ϕ axes in the conventional spherical co-ordinate system. Based on the 'Polarization', the phase of the electric field is:

$$E_S^{phase}(R,\theta,\varphi) = \angle \tilde{E}_S(R,\theta,\varphi).$$

In `pattern`, the options for Type name-value pair is defined as:

- 'directivity': Directivity = $10 \log_{10} |D(\theta,\varphi)|$ in dBi
- 'gain': Gain = $10 \log_{10} |G(\theta,\varphi)|$ in dBi
- 'realizedgain': Realized gain = $10 \log_{10} |G_R(\theta,\varphi)|$ in dBi
- 'efield': Electric field magnitude = $E_S^{abs}(R,\theta,\varphi)$ in volt per meter
- 'power': Power = $|E_S^{abs}(R,\theta,\varphi)|^2$ in volt per meter squared
- 'powerdB': Power = $20 \log_{10} |E_S^{abs}(R,\theta,\varphi)|$ in dB
- 'phase': Phase = $E_S^{phase}(R,\theta,\varphi)$ in degrees

For lossless antennas, the default value for Type name-value pair is 'directivity'. For lossy antennas, the default value for Type name-value pair is 'gain'. The radiated power (P_{rad}) is different from Type as 'power'.

For a detailed example on the pattern function, see "Field Analysis of Monopole Antenna" on page 5-909.

See Also

`pattern` | `efficiency` | `EHfields`

Array Concepts

- “Mutual Coupling” on page 2-2
- “Beamforming” on page 2-13
- “Grating Lobes” on page 2-17
- “Correlation Coefficient” on page 2-20
- “Infinite Arrays” on page 2-22
- “Manipulate Array Elements” on page 2-35

Mutual Coupling

In this section...
“Active or Scan Impedance” on page 2-2
“Mutual Impedance” on page 2-3
“Coupling Matrix” on page 2-4
“Array Factor and Pattern Multiplication” on page 2-5
“Isolated Element Pattern” on page 2-10
“Embedded Element Pattern” on page 2-11

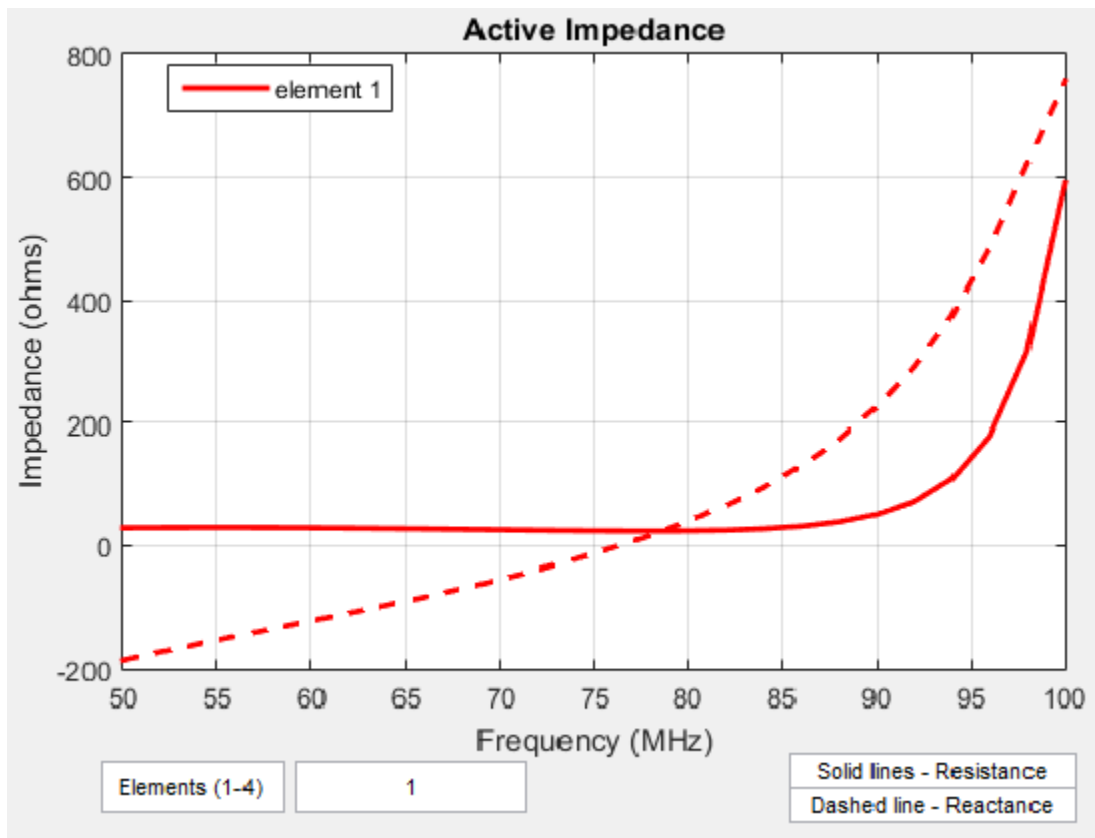
Mutual coupling is the electromagnetic interaction between the antenna elements in an array. The current developed in each antenna element of an array depends on their own excitation and also on the contributions from adjacent antenna elements. Mutual coupling is inversely proportional to the spacing between the different antenna elements in an array. Mutual coupling in an array causes:

- Changes in the radiation pattern of the array
- Changes in the input impedance of the individual antenna elements in an array

To characterize mutual coupling, you can use mutual impedance, S-parameters, a coupling matrix, or an embedded element pattern.

Active or Scan Impedance

Active impedance, or *scan impedance*, is the input impedance of each antenna element in an array, when all elements are excited.

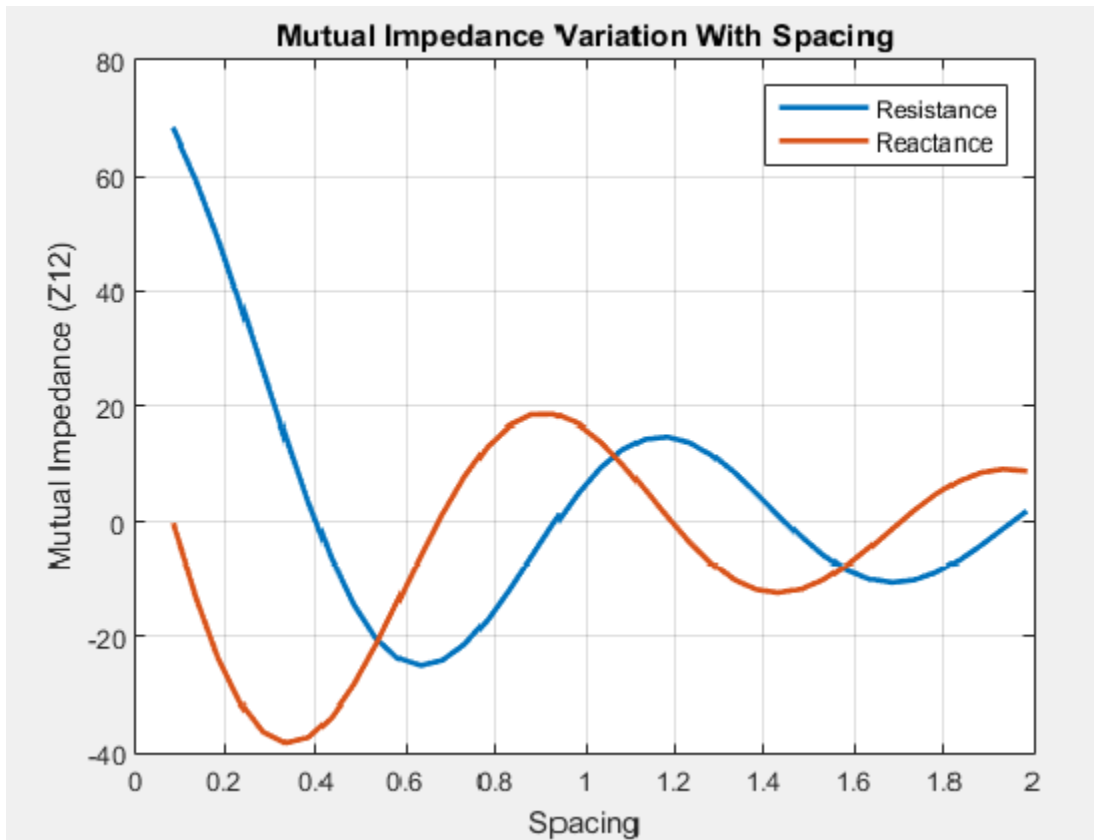


The active impedance of an array depends on:

- Array configuration
- Spacing between elements
- Phase shift applied at each element

Mutual Impedance

The effect of mutual coupling is observed or modeled by varying the space between the antenna elements in the array. Any change in the inter-element spacings changes the *mutual impedance* between the antenna elements. For example, the plot shows the mutual impedance of a two-element dipole array as a function of inter-element spacing.



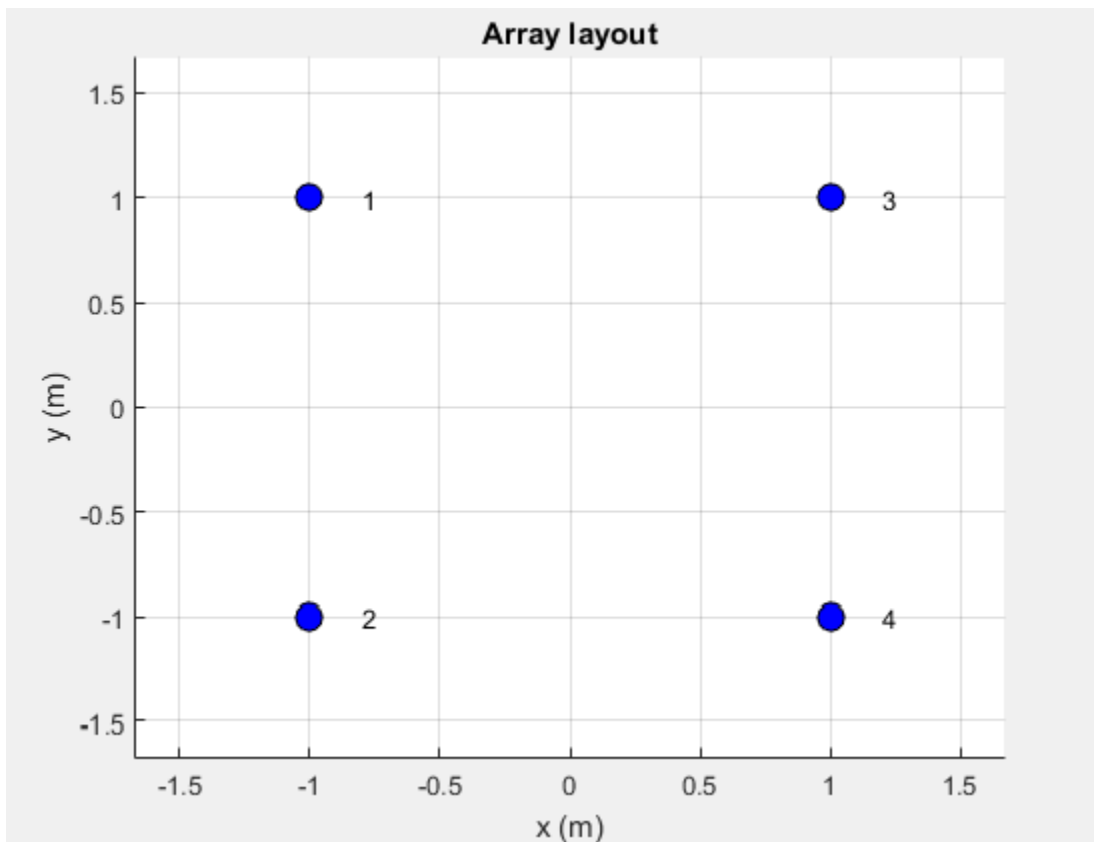
The plot shows that as the spacing between elements increases, the mutual impedance, and hence mutual coupling decreases.

Coupling Matrix

A *coupling matrix* is used to characterize the mutual coupling between the antenna elements at the port level. This matrix is calculated using S-parameters or Z-parameters and is used to decouple the array.

S-Parameter Matrix

To calculate the coupling matrix, you can use the S-parameter matrix. You calculate each column of the S-parameter matrix by feeding the antenna in that column by 1V. Consider an element array arranged in a 2x2 grid. Visualize the grid and the element numbers using the `layout`.



There are four ports in this array. The corresponding S-parameter matrix would be of size 4 x 4:

$$S = \begin{pmatrix} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} \\ S_{31} & S_{32} & S_{33} & S_{34} \\ S_{41} & S_{42} & S_{43} & S_{44} \end{pmatrix}$$

The termination resistance is internally set to a default of 50 ohms and can be omitted during the calculation. If the termination is different, specify the resistance value when using this function. The diagonal terms in the matrix shown, represent the self-interaction which is also commonly referred to as the reflection coefficient. The off-diagonal terms capture the mutual coupling between the ports of the antenna.

Use `sparameters` to calculate the S-parameter coupling matrix of an array in Antenna Toolbox.

Array Factor and Pattern Multiplication

The basis of the array theory is the *pattern multiplication* theorem. This theorem states that the combined pattern of N identical array elements is expressed as the element pattern times the array factor.

The array factor is calculated using the formula:

$$AF = \sum_{i=0}^N V(i) \cdot e^{j(k\sin\theta\cos\varphi \cdot x(i) + k\sin\theta\sin\varphi \cdot y(i) + k\cos\theta \cdot z(i))}$$

where:

- N is the number of elements in the array.
- V is the applied voltage (amplitude and phase) at each element in the array.
- k is the wave number.
- θ and φ are the elevation and azimuth angles.
- x , y , and z are the Cartesian coordinates of the feed locations for every antenna element of the array.

Once the array factor is calculated using the above equation, you can calculate the beam pattern of the array as the product of the array factor and the beam pattern of the individual antenna element of the array.

Array pattern = AF individual antenna element pattern*

The analysis assumes that the array elements are uncoupled. This means that the current in one element does not excite currents in the other elements or there is no mutual coupling between different elements of the array. This is the most serious limitation of the pattern multiplication theorem, restricting its use to arrays with large element spacing.

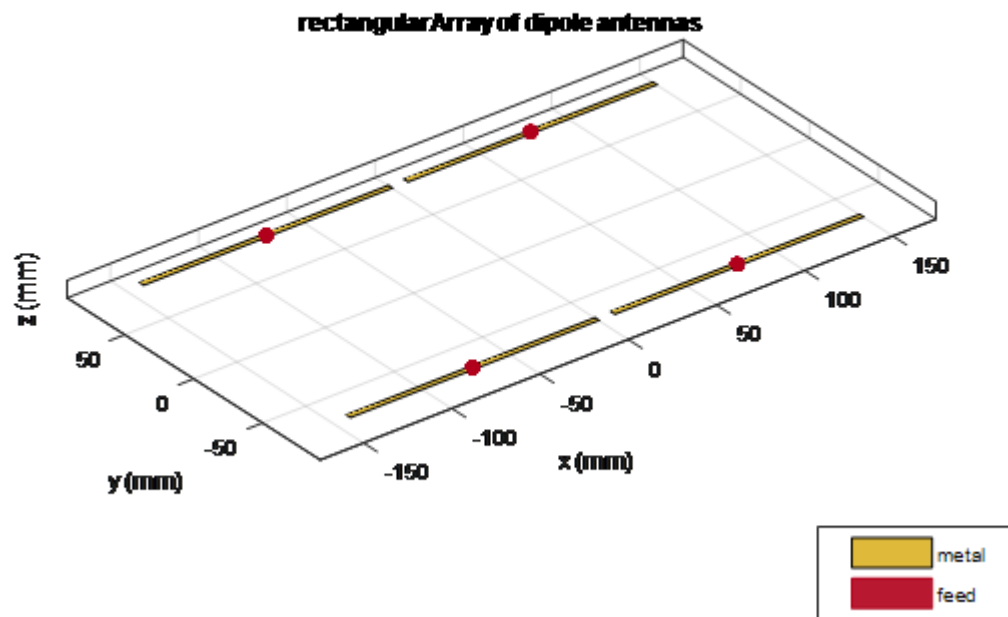
For example:

Calculate the array pattern of the rectangular array of dipoles in the x-y plane with a spacing of half-lambda

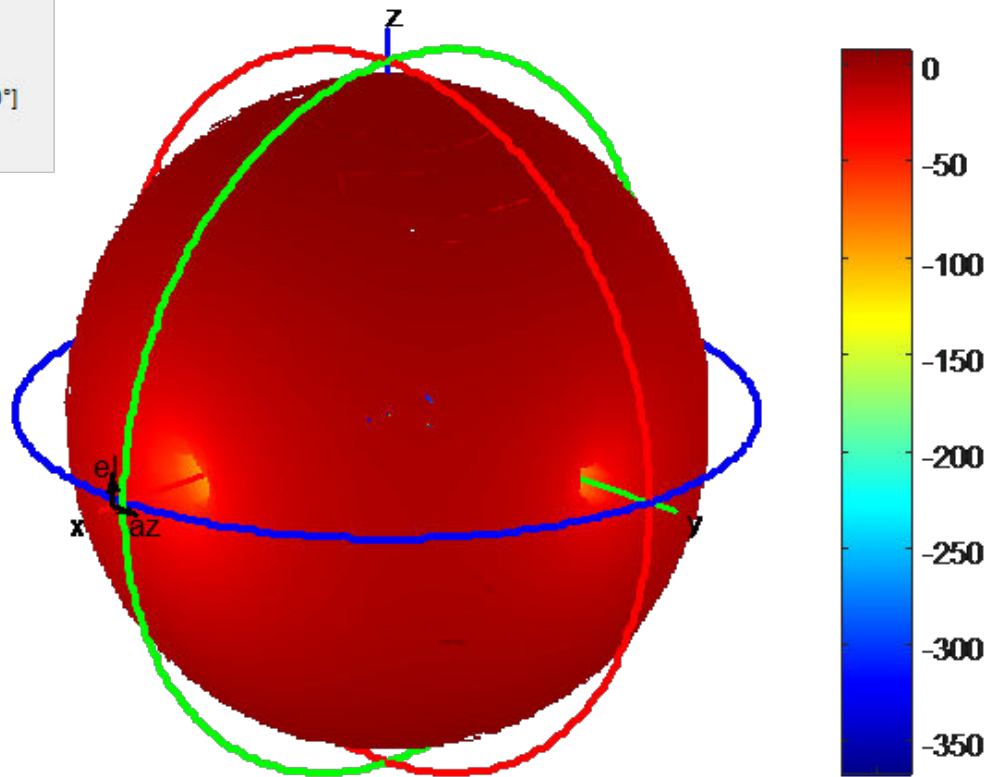
```
fc = 1e9;
lambda = physconst('lightspeed')/fc;
az = -180:0.1:180;
el = -90:0.1:90;

%% Element
d = design(dipole,1e9);
d.Tilt = 90;
d.TiltAxis = [0 1 0];

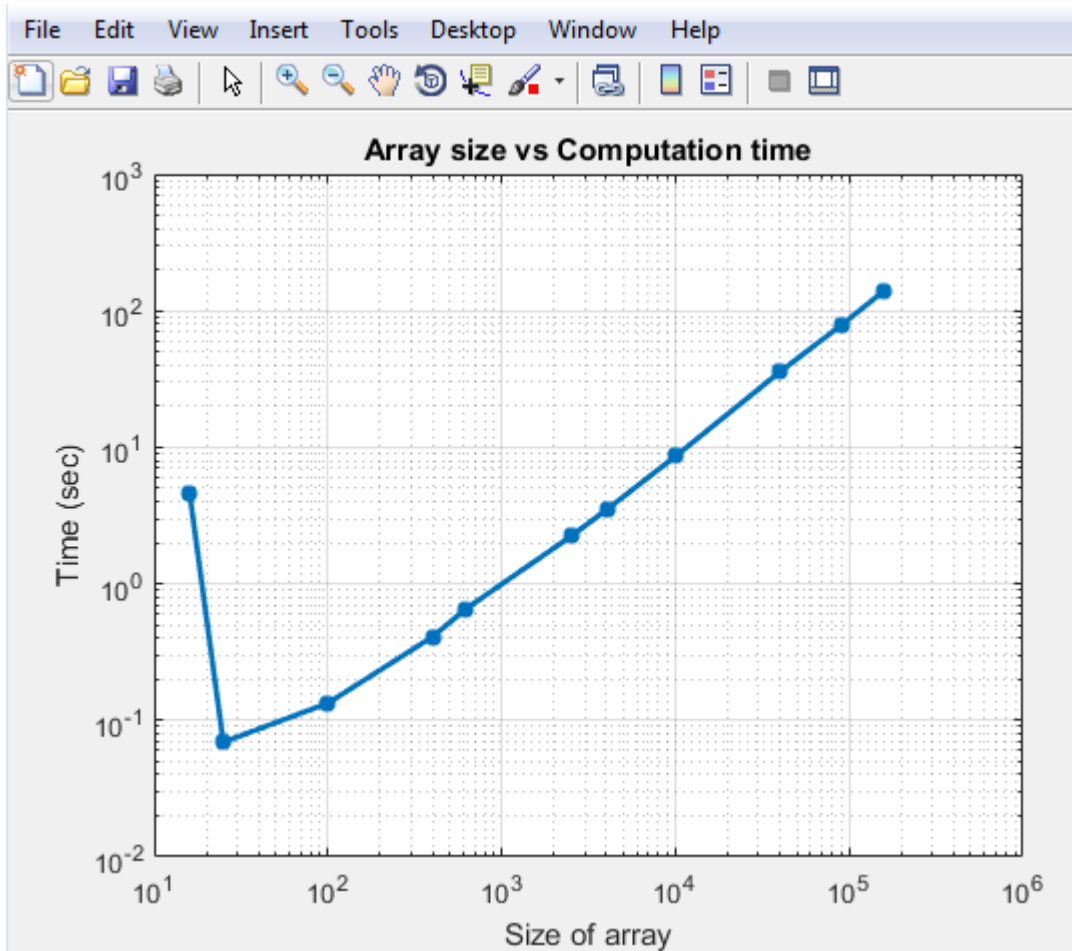
%% Array
r = rectangularArray;
r.Element = d;
r.RowSpacing = lambda/2;
r.ColumnSpacing = lambda/2;
figure; show(r) ;
figure; patternMultiply(r, fc, az, el);
```

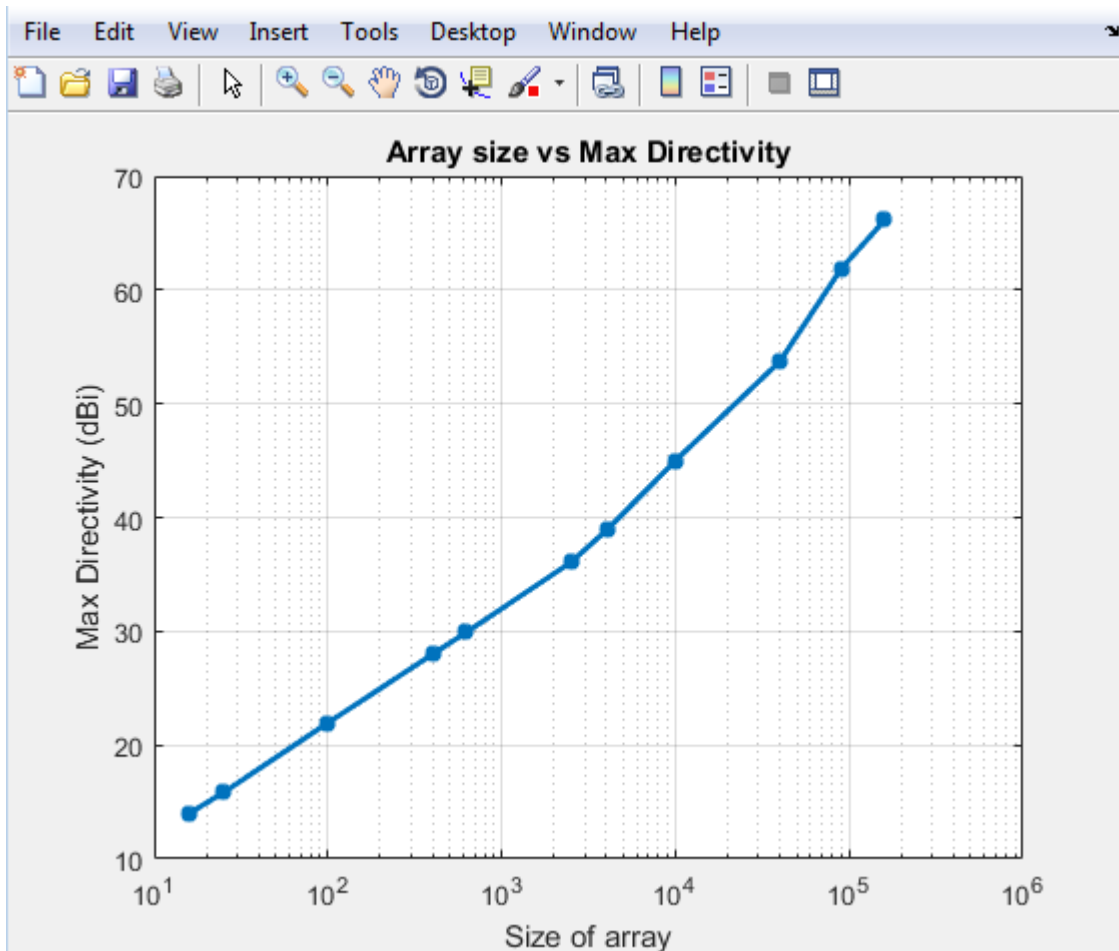


Output	: Directivity
Frequency	: 1 GHz
Max value	: 8.05 dBi
Min value	: -368 dBi
Azimuth	: [-180°, 180°]
Elevation	: [-90°, 90°]



Increase the size of the rectangular array to 200k elements. Below is the time taken for the computation and the calculated directivity.



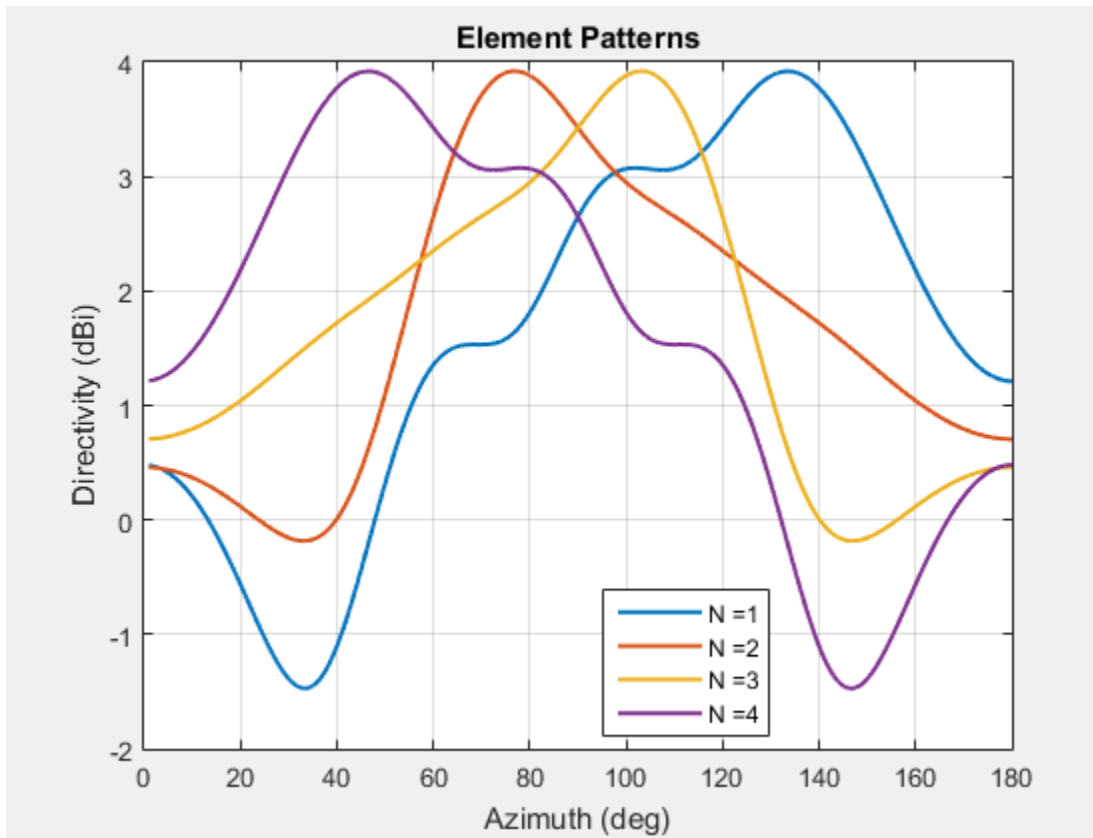


You see that the time taken to solve a 100k array is less than 100 seconds. The initial bump in time for the small array is because you perform EM analysis to compute the pattern of the individual dipole element in the array. Once this analysis is done the results are cached and the successive calls do not perform any EM analysis. As a result the increase in time is fairly linear. This is the biggest advantage of using pattern multiplication. It lets you solve large arrays quickly and with limited memory requirements.

Isolated Element Pattern

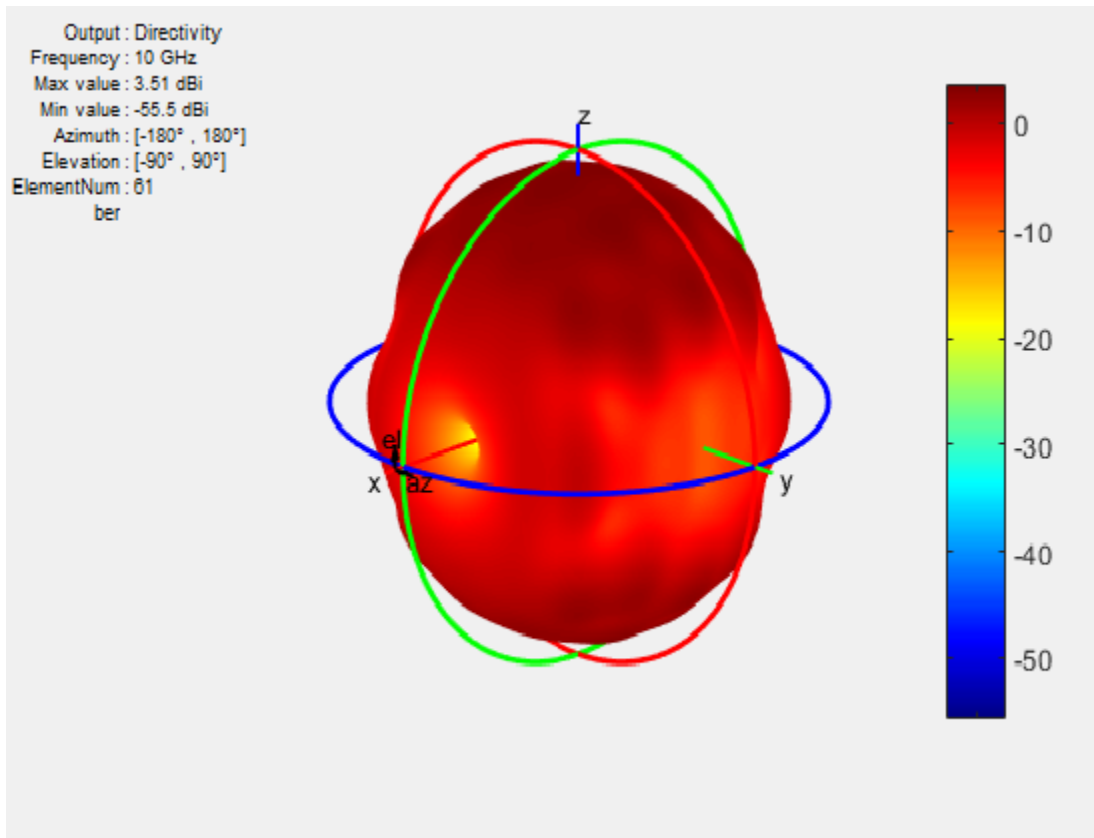
Patterns of individual elements in small arrays vary significantly. Therefore you cannot use pattern multiplication for complete array pattern because isolated element pattern assumes that all elements have the same pattern.

You can calculate the complete array pattern of small arrays by plotting the pattern all the elements separately in a small array. To obtain this pattern, each element is individually excited and the rest of the array elements terminated using reference impedance. The plot shows the radiation pattern of individual elements of 4-element array.



Embedded Element Pattern

The *embedded element pattern* is the pattern of a single element embedded in a finite array, calculated by driving a specific (typically the central) element in the array. The rest of the array elements are terminated using reference impedance. This method is useful for large array because the effect of mutual coupling on the individual element is captured. It is important to note that the edge effects can be ignored since the size of the array is assumed to be very large. It is common to use the center antenna element for this calculation. Due to the size of the array the radiation patterns of the elements in the array can be approximated with the embedded element pattern instead of the isolated element pattern. Finally, pattern multiplication is used to calculate the complete array pattern.



The figure shows the embedded element pattern of a center element in a 11x11 array. You can also calculate the embedded element pattern as a magnitude of electric field.

Isolated element pattern is not recommended for large arrays as this method does not account for the coupling effects of elements around it.

Scan Blindness

In large arrays, it is possible that the array directivity reduces drastically at certain scan angles. At these scan angles, referred to as the blind angles, the array does not radiate the power supplied at its input terminals [1]. The scan blindness can occur while using these common mechanisms:

- Surface wave excitation
- Grating lobe excitation

To detect scan blindness in large finite arrays, study the embedded element pattern. In infinite array analysis, this pattern is known as the array element pattern.

References

- [1] Stutzman, W.L. Thiele, G.A. *Antenna Theory and Design*, 3rd Edition. New York: Wiley, 2013, p. 307.

Beamforming

In this section...

“Side Lobe Control” on page 2-13

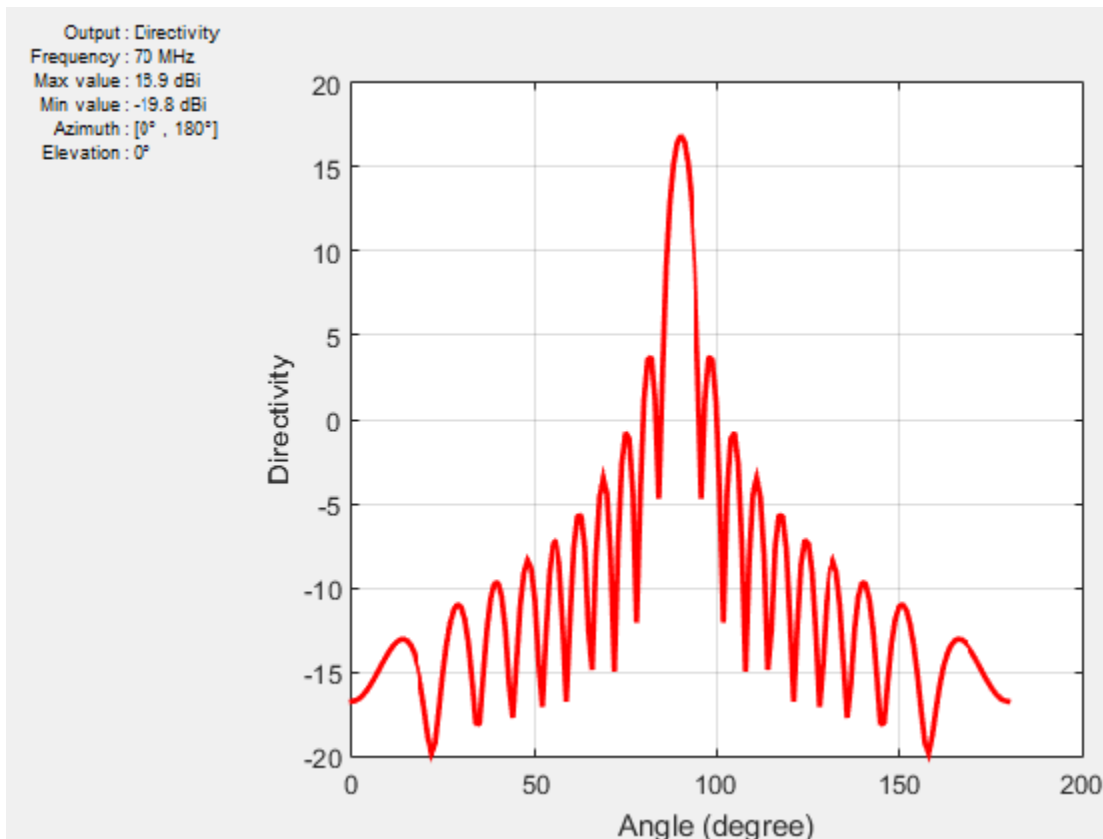
“Beam Scanning” on page 2-14

Beamforming is the process of generating a directional beam from an antenna array. Achieve beamforming by weighting individual elements by using side lobe control or beam scanning.

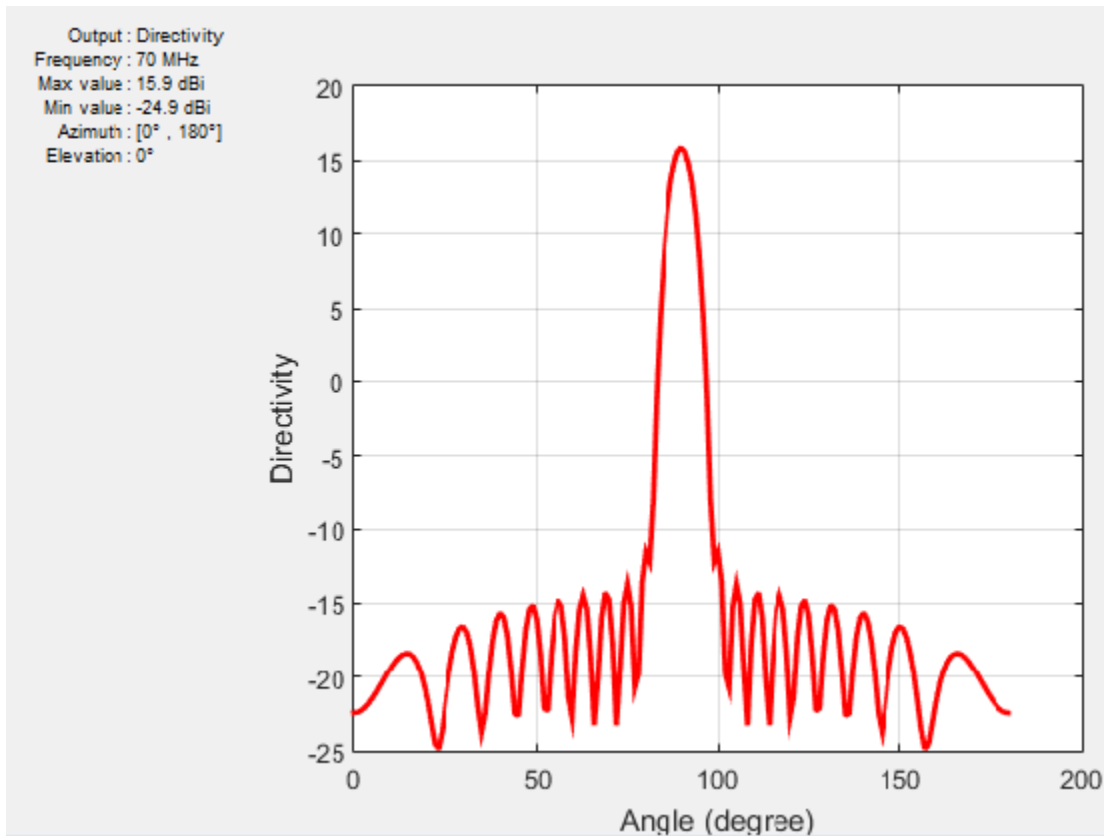
Side Lobe Control

Side lobes are undesired and lead to reception or transmission of energy in unwanted directions. *Side lobe control* in an array is achieved using amplitude taper or amplitude weighting. Amplitude tapering changes the excitation amplitude of each element in the array. Minor lobe levels are controlled using amplitude taper that runs from the center of the array to the end of the array. Smoother amplitude tapering gives larger small side lobe levels but half-power beamwidth. These arrays are non-uniformly excited arrays.

Consider a linear array of 21 elements. Without amplitude tapering, the array contains unwanted side lobes in the desired direction:



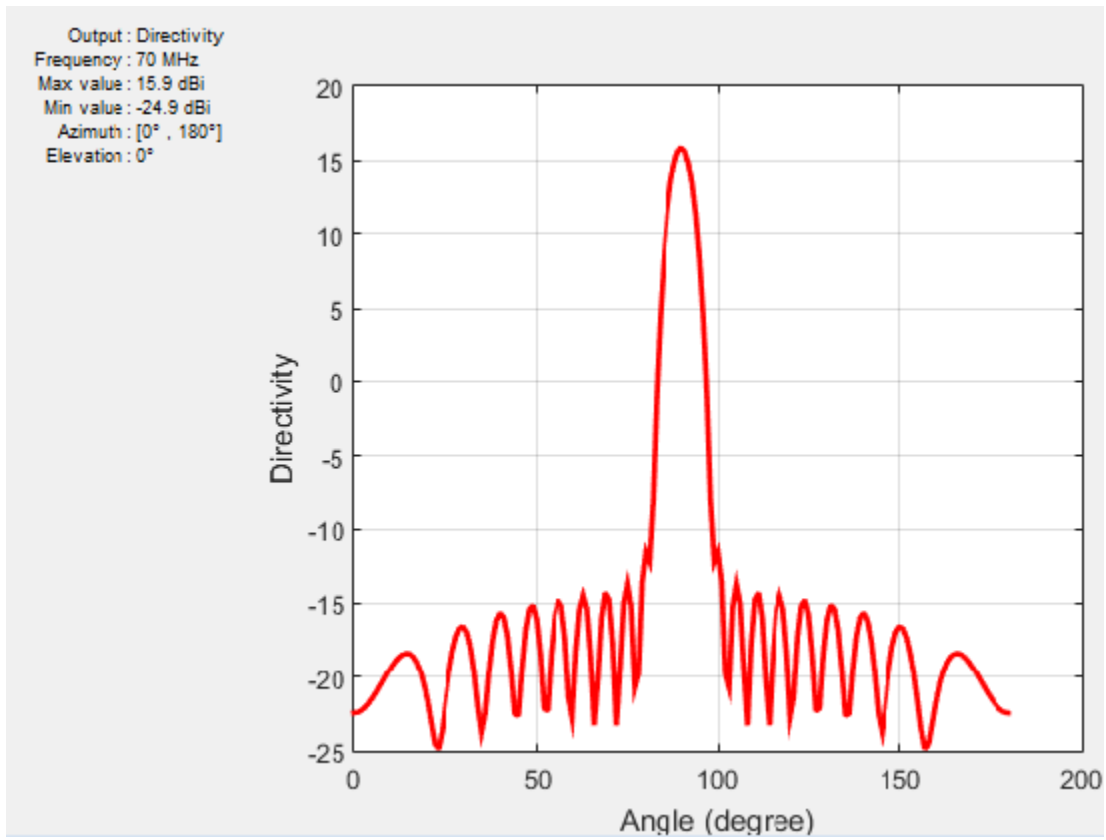
By using amplitude tapering on the linear array, you control the side lobes and achieve a better main beam in the desired direction:



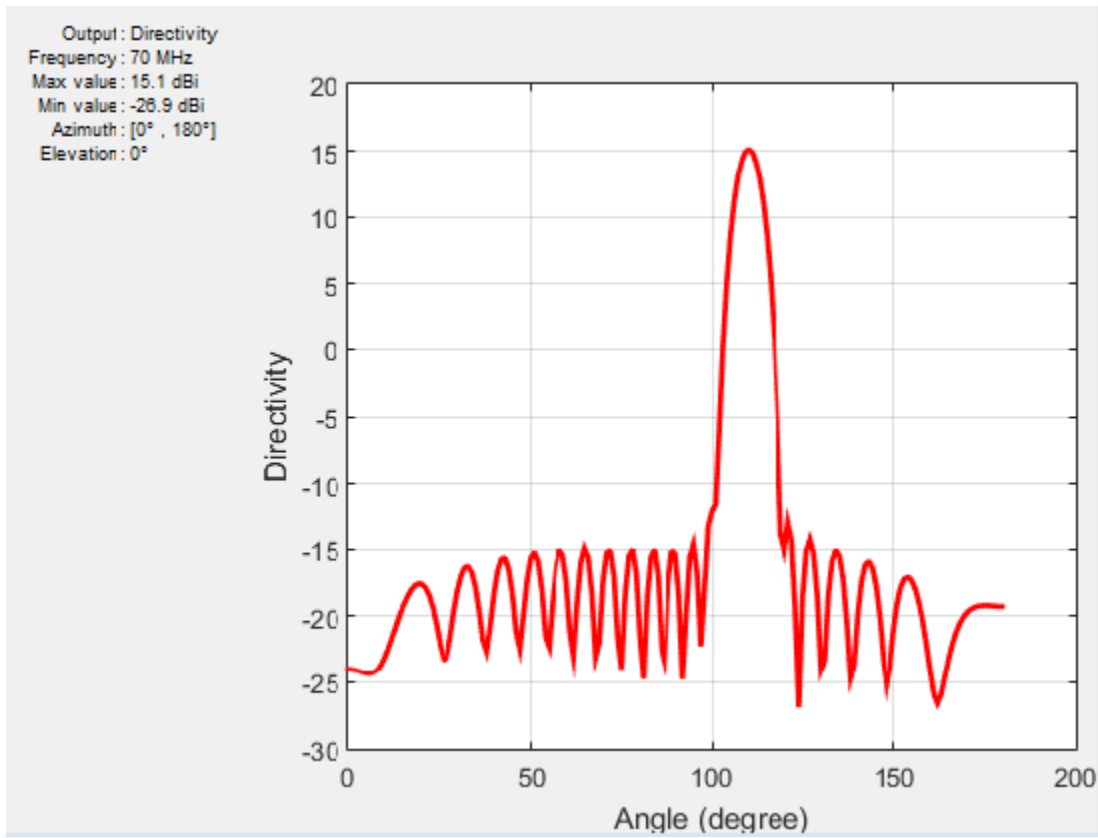
Beam Scanning

Beam scanning is the movement of a radiation pattern in space. You can achieve beam scanning by controlling the progressive phase difference between the elements to direct the beam in any desired direction. The phase shift changes the phase of the excitation currents of each element in an array. You can achieve this phase shift using phase shifter devices, a time delay, frequency scanning, beam switching, or digital beamforming.

Consider a linear array of 21 elements where peak directivity is shown without beam scanning:



To scan the beam at a specific angle, use phase shift on the linear array:



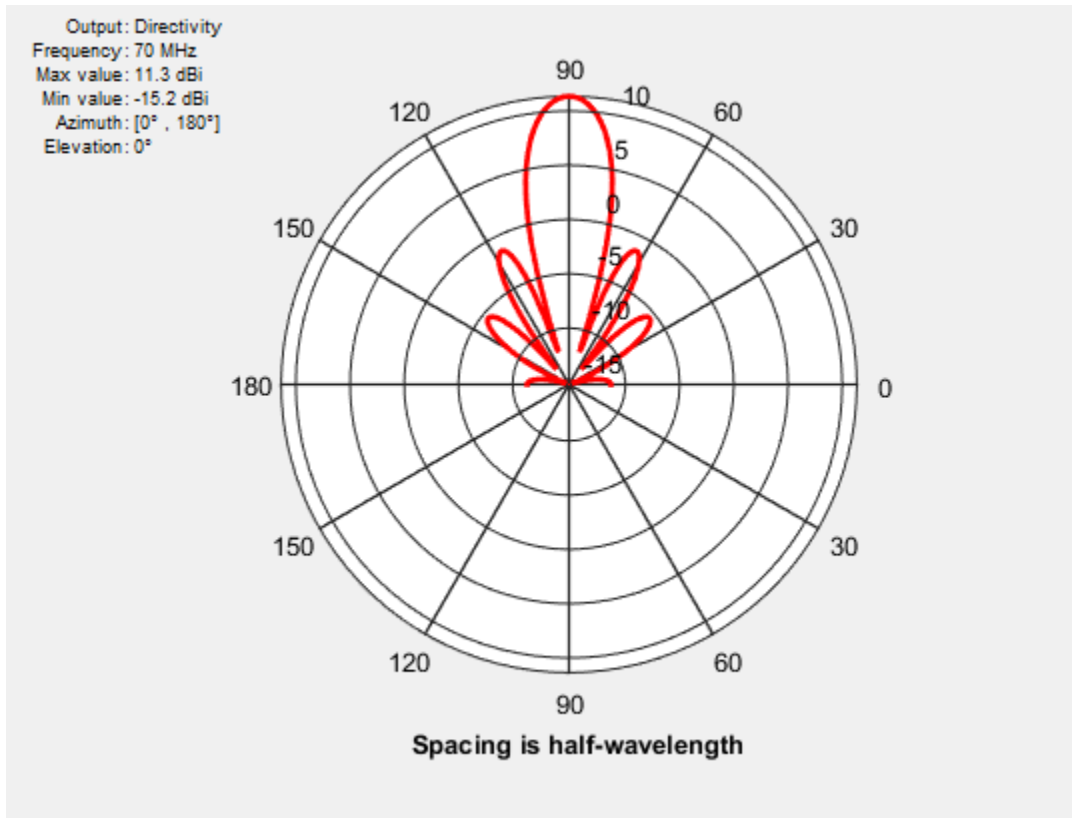
References

- [1] Stutzman, W.L. Thiele, G.A. *Antenna Theory and Design*, 3rd Edition. New York: Wiley, 2013, p. 307.

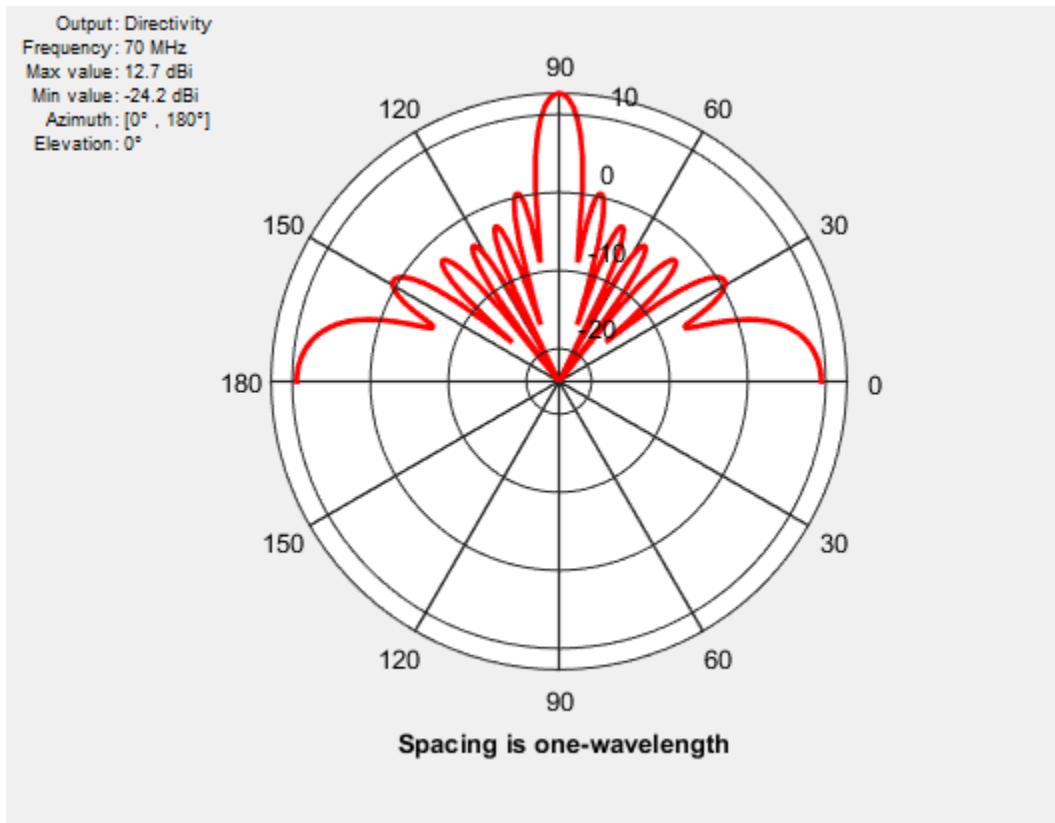
Grating Lobes

Grating lobes are the maxima of the main beam, as predicted by the pattern multiplication theorem. When the array spacing is less than or equal to $\lambda/2$, only the main lobe exists in the visible space, with no other grating lobes. Grating lobes appear when the array spacing is greater than $\lambda/2$. For large spacing, grating lobes can appear in the visible space even at a zero scan angle.

Consider a linear array of seven elements spaced one-half wavelength apart. This array contains no grating lobes.



When you increase the element spacing to one wavelength, the number of side lobes increases.



When you increase the element spacing to 1.5 wavelengths apart, the grating lobes appear in the visible region.



The grating lobes move in or out of the visible region depending on the spacing between the elements and the relative phase between them.

References

- [1] Stutzman, W.L. Thiele, G.A. *Antenna Theory and Design*, 3rd Edition. New York: Wiley, 2013, p. 307.

Correlation Coefficient

In this section...

“Far-Field Radiation Pattern” on page 2-20

“S-Parameter Characterization” on page 2-20

The *correlation coefficient* is the relationship between the incoming signals at the antenna ports in an array. Mutual coupling in array systems degrades the performance of the array. The correlation coefficient between antennas is used as a performance metric in multiple-input multiple-output (MIMO) systems to quantify the system performance and efficiency of the antennas. By using the correlation coefficient, a MIMO system designer is able to understand the level of coupling that exists between the antenna ports in the system. To minimize the mutual coupling would imply to reduce the correlation coefficient between the pairs of ports. Antenna designers use two approaches to the calculate correlation coefficient: the far-field Radiation pattern and S-parameters.

Far-Field Radiation Pattern

The correlation coefficient of a two antenna array system is:

$$\rho_e = \frac{\left| \int \int_{4\pi} [\vec{F}_1(\theta, \phi) \cdot \vec{F}_2(\theta, \phi)] d\Omega \right|^2}{\int \int_{4\pi} |\vec{F}_1(\theta, \phi)|^2 d\Omega \int \int_{4\pi} |\vec{F}_2(\theta, \phi)|^2 d\Omega}$$

where $\vec{F}_i(\theta, \phi)$ is the radiation pattern of the antenna system when port i is excited. Computing the correlation coefficient using this formula, requires the radiation pattern of the antenna. This approach is hard and time consuming.

S-Parameter Characterization

Antenna Toolbox™ uses the S-parameter characterization to calculate correlation between antenna elements in an array. This approach is simpler than the far-field approach because the S-parameter calculation does not use the radiation patterns of the antennas. Correlation coefficient is calculated using S-parameter by using:

$$\rho_e = \frac{|S_{11}^* S_{12} + S_{21}^* S_{22}|^2}{(1 - (|S_{11}|^2 + |S_{21}|^2))(1 - (|S_{22}|^2 + |S_{12}|^2))}$$

The advantages of this method are quick analysis and broadband correlation results. However, this approach assumes that the antennas are lossless and that incoming waves are uniformly distributed. To calculate and plot the correlation between antennas in an array, use the `correlation` function in Antenna Toolbox™.

References

- [1] Blanch, S. Romeu, J. and Corbella, I. *Exact Representation of antenna system diversity performance from input parameter description*

[2] Stutzman, W.L. Thiele, G.A. *Antenna Theory and Design*, 3rd Edition. New York: Wiley, 2013, p. 307.

Infinite Arrays

In this section...
“What Are Infinite Arrays?” on page 2-22
“Infinite Array Analysis” on page 2-22
“Create Infinite Array Using Antenna Toolbox” on page 2-23
“Choose a Unit Cell” on page 2-24
“Scan Infinite Arrays” on page 2-25
“Scan Impedance and Scan Element Pattern” on page 2-26
“Compare Scan Element Pattern of Finite and Infinite Arrays” on page 2-26
“Impact of Infinite Double Summation” on page 2-32

What Are Infinite Arrays?

Infinite arrays are rectangular arrays of infinite extent. In an infinite array, a single element called a *unit cell*, is repeated uniformly an infinite number of times along a plane.

Infinite Array Analysis

All arrays used in real-world scenarios are finite. But antenna arrays used in radio astronomy, air defense, or surveillance radar can have more than 1000 antenna elements. In such large arrays, the electromagnetic analysis of each element is tedious and time consuming.

Infinite array analysis ignores the effect of truncation (edge effect) at array edges. The method analyzes the behavior of the active antenna element as a function of frequency and scan. The goal of infinite array analysis is to extract the behavior of the active antenna element embedded in the array.

Assumptions

For infinite array analysis, array size must be greater than 10×10 . The technique makes other assumptions:

- Each element is identical.
- Each element is uniformly excited in amplitude.
- All elements are spaced uniformly in two dimensions.

Infinite Array Solver

To model an infinite array, the method of moments (MoM) formulation is changed to account for the infinite behavior by replacing Green's functions with periodic Green's functions. The periodic Green's function is an infinite double summation.

Green's Function	Periodic Green's Function
$g = \frac{e^{-jkR}}{R}$ $R = \vec{r} - \vec{r}' $	$g_{\text{periodic}} = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} e^{j\phi_{mn}} \frac{e^{-jkR_{mn}}}{R_{mn}}$ $R_{mn} = \sqrt{(x - x' - x_m)^2 + (y - y' - y_n)^2 + (z - z')^2}$ $\phi_{mn} = -k(x_m \sin\theta \cos\varphi + y_n \sin\theta \sin\varphi)$ $x_m = m \cdot d_x, y_n = n \cdot d_y$

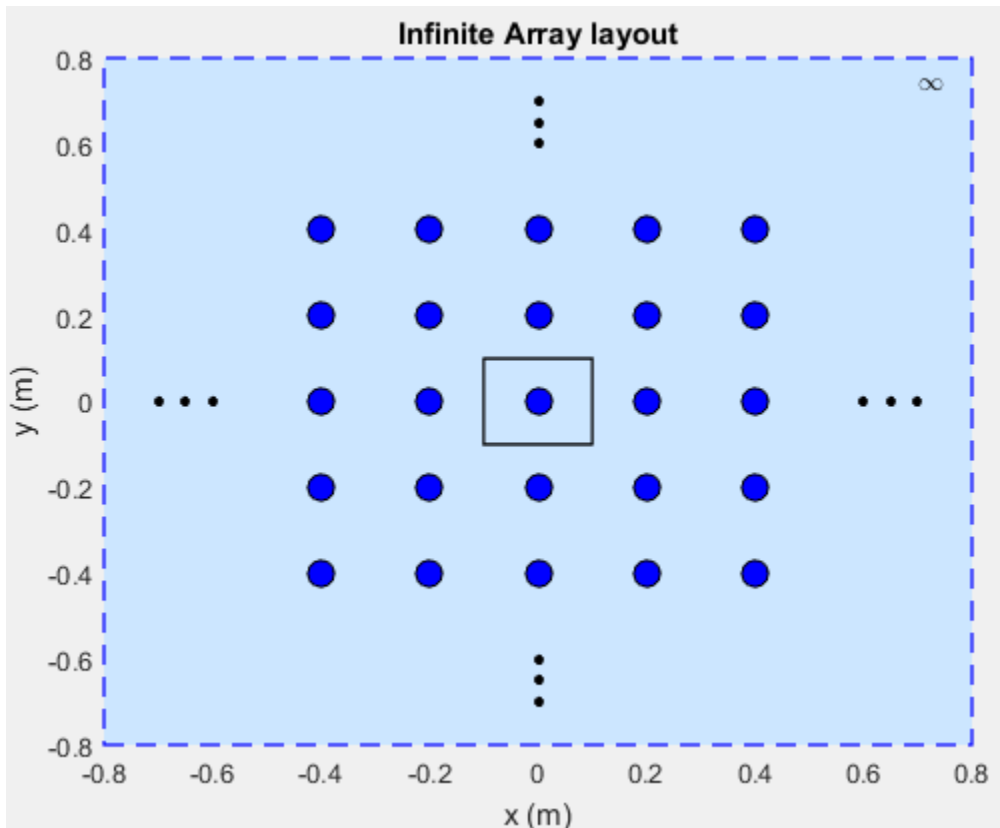
d_x and d_y are the ground plane dimensions that define the x and y dimensions of the unit cell. θ and Φ are the scan angles.

The periodic Green's function has an additional exponential term added to the infinite sum. The Φ_{mn} term accounts for the scanning of the infinite array. The periodic Green's function also accounts for the effect of mutual coupling.

Create Infinite Array Using Antenna Toolbox

To create an infinite array, use the `infiniteArray` object to repeat a single antenna element (unit cell), infinitely along the X-Y plane. The `layout` function displays a typical unit cell.

```
infarray = infiniteArray;
layout(infarray)
```



Choose a Unit Cell

You can use any antenna from the Antenna Toolbox as the unit cell. The unit cell requires a ground plane to specify the boundaries. You can use a reflector to back antennas that do not have a ground plane.

The default reflector properties are:

```
r = reflector
```

```
reflector with properties:
```

```

    Exciter: [1x1 dipole]
GroundPlaneLength: 0.2000
GroundPlaneWidth: 0.2000
    Spacing: 0.0750
    Tilt: 0
    TiltAxis: [1 0 0]
```

The default unit cell in an infinite array is a reflector that has a dipole as an exciter. The `Spacing` property gives the distance between the reflector and the exciter. The default infinite array properties are:

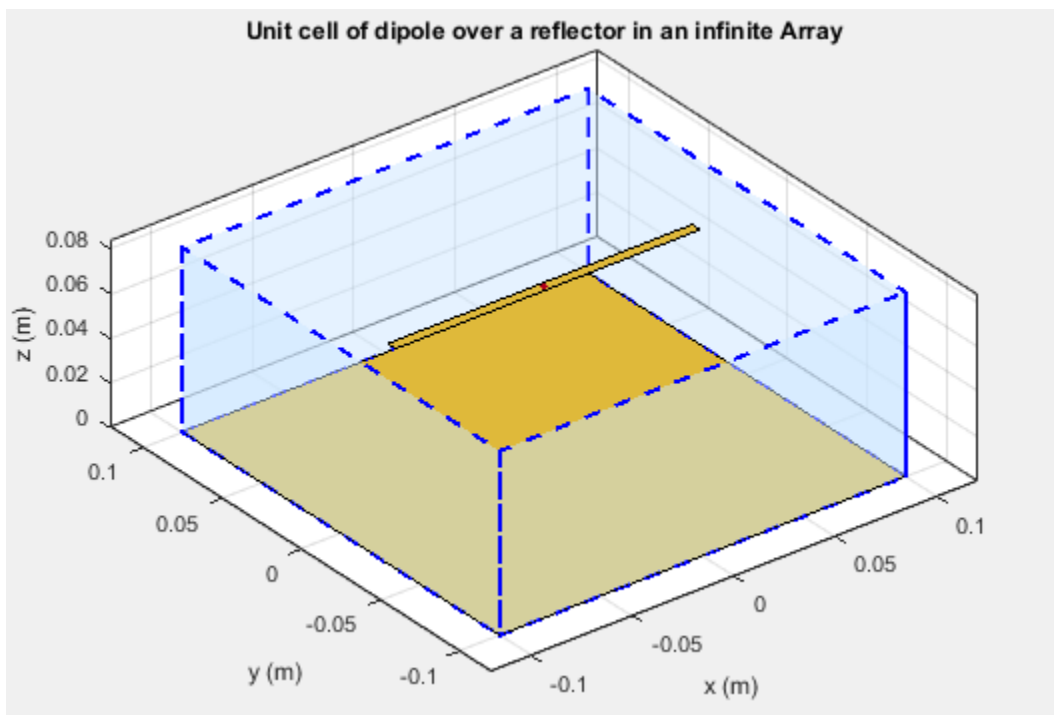
```
infarray = infiniteArray
```

```
infiniteArray with properties:
```

```

    Element: [1x1 reflector]
    ScanAzimuth: 0
    ScanElevation: 90
```

```
show (infarray)
```



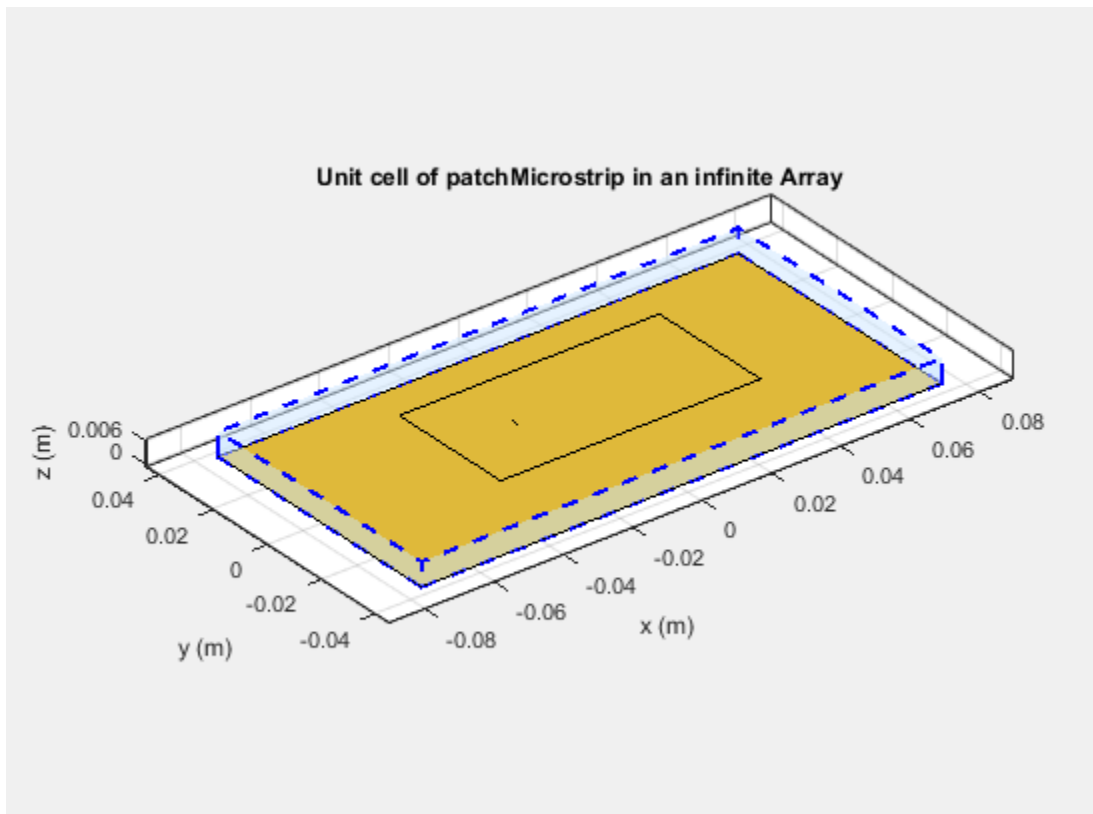
The dotted blue box bounds the unit cell. Ground plane length and ground plane width of the unit cell are the dimensions of the antenna element of the infinite array.

An antenna with a ground plane, such as a microstrip patch antenna, is specified directly as an `Element` of an infinite array.

```
infarray = infiniteArray('Element', patchMicrostrip)
```

```
infarray =
  infiniteArray with properties:
      Element: [1x1 patchMicrostrip]
      ScanAzimuth: 0
      ScanElevation: 90
```

```
show(infarray)
```



The Antenna Toolbox infinite array is located in the X-Y plane. Unit cells consisting of antennas with ground planes are also located in the X-Y plane. For antennas used as unit cells, such as the one in this example, you ignore the value of the `Tilt` property.

Scan Infinite Arrays

You scan a finite array by specifying the appropriate phase shift for each antenna element. In Antenna Toolbox, you specify the scan angle (in azimuth and elevation) and frequency for infinite array analysis. By default, an array always scans at boresight (azimuth = 0 degrees and elevation = 90 degrees).

```
infarray = infiniteArray
    infiniteArray with properties:
        Element: [1x1 reflector]
        ScanAzimuth: 0
        ScanElevation: 90
```

To change the scan angles, change the values of `ScanAzimuth` and `ScanElevation`.

Scan Impedance and Scan Element Pattern

To calculate the scan impedance for an infinite array, use the `impedance` function as a function of scan angle. Fixing the scan angle pair and sweeping the frequency variable reveals the frequency dependency in the scan impedance. Because `ScanAzimuth` and `ScanElevation` are scalar values, you must use a `for`-loop to calculate the complete scan impedance of the array. For more information on calculating the scan impedance and the scan element pattern see, “Infinite Array Analysis” on page 5-58.

Scan Element Pattern

To calculate the scan element pattern using scan impedance, use these expressions:

$$g_s(\theta) = \frac{4R_g R_{\text{iso}} g_{\text{iso}}(\theta)}{|Z_s(\theta) + Z_g|^2}$$

- R_g — Resistance of generator
- Z_g — Impedance of generator
- Z_s — Scan impedance
- $g_{\text{iso}}(\theta)$ — Pattern of isolated element
- R_{iso} — Resistance of isolated element

The scan element pattern can also be expressed in terms of the reflection coefficient, $\Gamma(\theta)$:

$$g_s(\theta) = \frac{4R_{\text{iso}} g_{\text{iso}}(\theta)}{R_s(\theta)} (1 - |\Gamma(\theta)|^2)$$

The Antenna Toolbox software calculates the scan element pattern of a finite array by driving just a single element. You terminate all the other elements using a suitable impedance. The resulting element pattern includes mutual coupling and is valid for all scan angles.

Compare Scan Element Pattern of Finite and Infinite Arrays

Case 1: Compare finite array and infinite array with unit cell of dimensions $0.5\lambda \times 0.5\lambda$

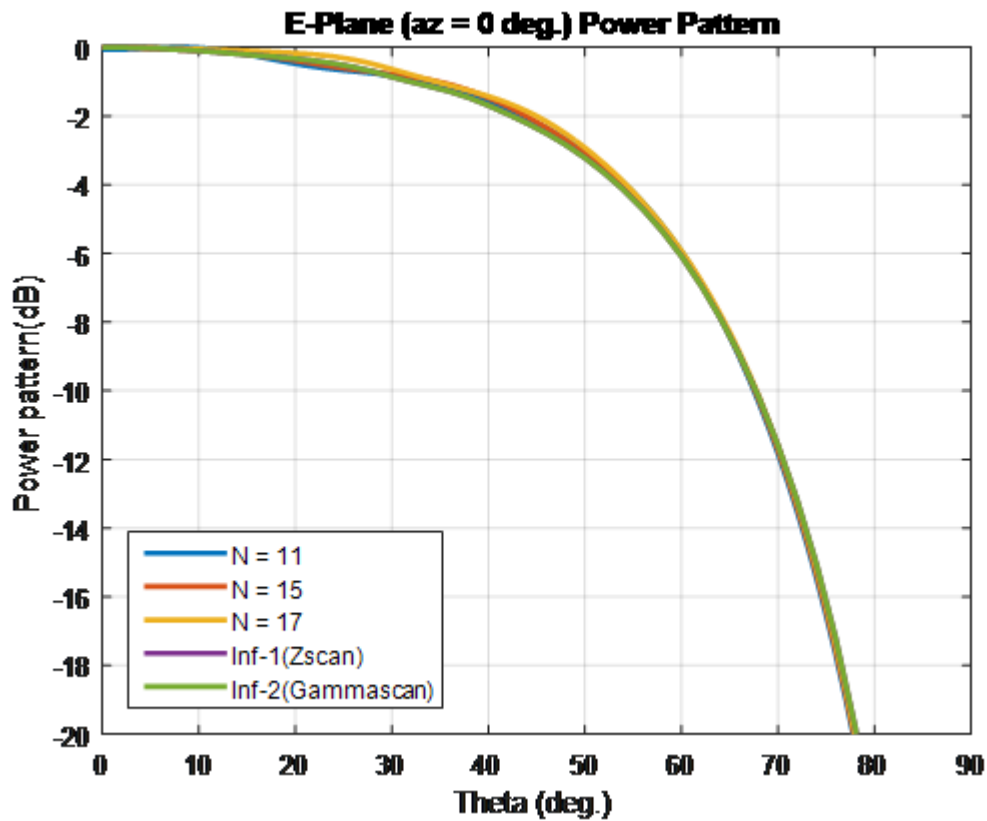
To calculate the scan element pattern of the finite arrays, first, create a reflector-backed dipole. Set the dipole dimensions to $\text{Length}(L) = 0.495\lambda$ and $\text{Width}(W) = \lambda/160$ and the ground plane dimensions to $0.5\lambda \times 0.5\lambda$. Place the dipole at a distance of $h = \lambda/4$ from the reflector. The ground plane dimensions set the boundaries of the unit cell. Create finite arrays of sizes 11×11 , 15×15 , and 17×17 using this unit cell.

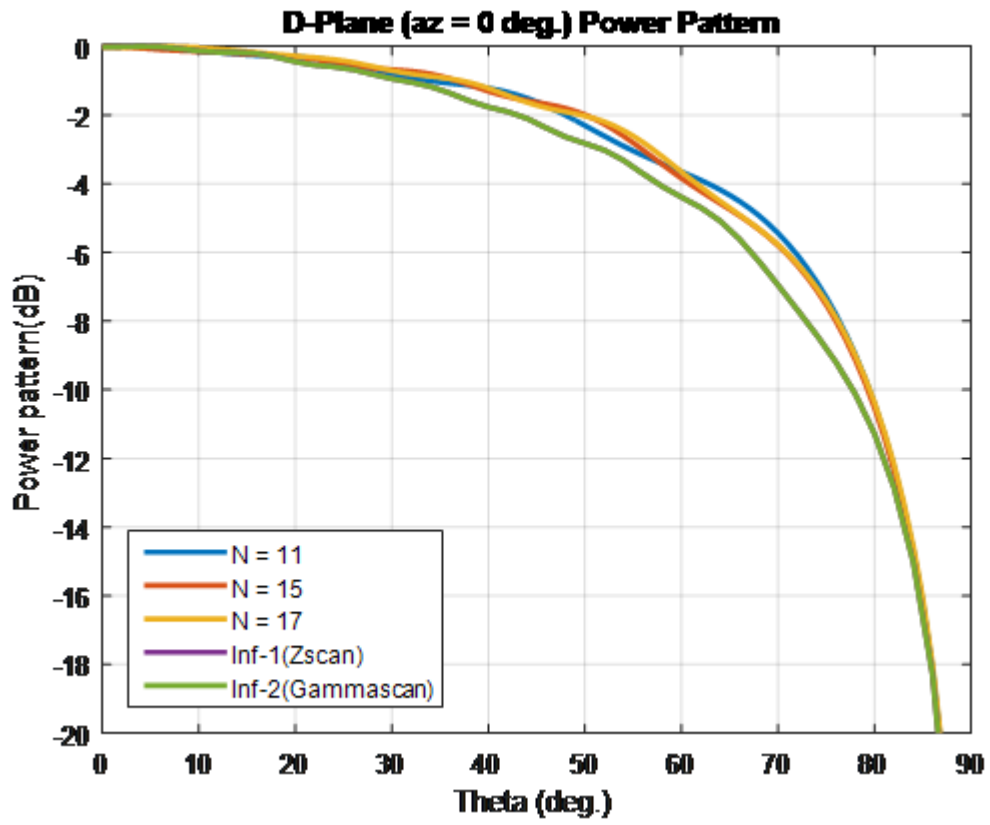
For finite arrays, calculate the scan element pattern by driving a single element in the array. Terminate all other finite array elements using the broadside resistance of the infinite array. For an

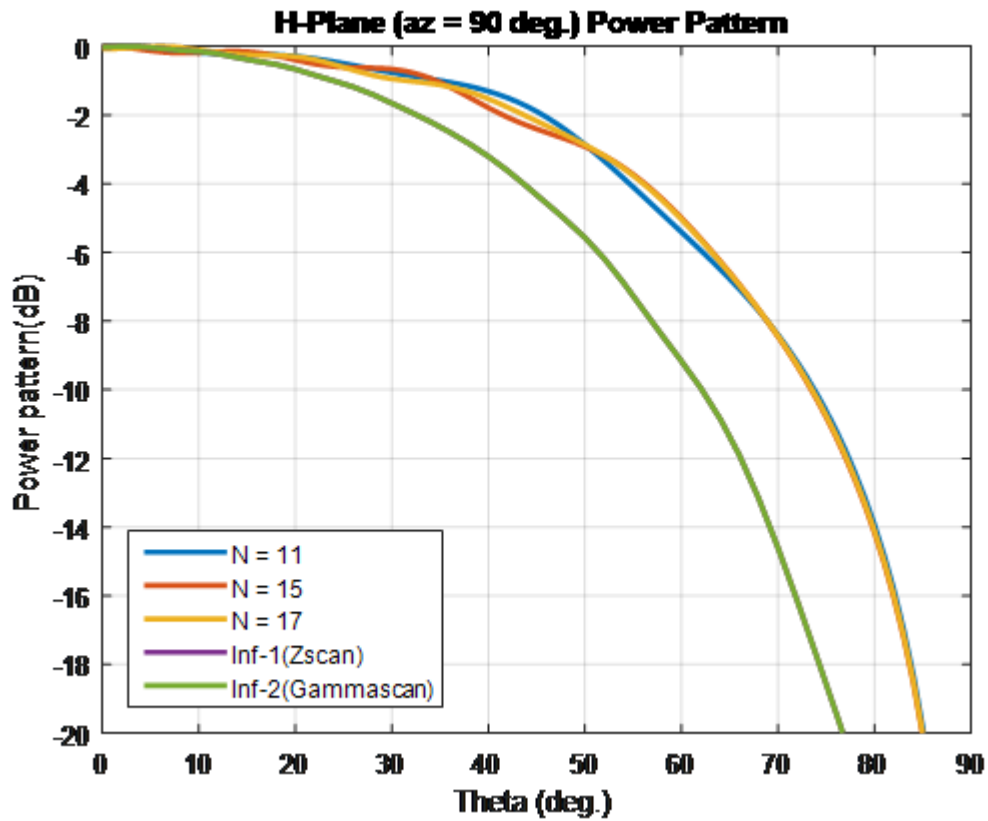
infinite array with the unit cell of dimensions $0.5\lambda \times 0.5\lambda$, the broadside resistance is 176Ω . Calculate the scan element pattern for E-, D-, and H-planes of all three finite arrays.

To calculate the scan element pattern of an infinite array, create an infinite array using the same unit cell and the `infiniteArray` class. Calculate the scan impedance for three scan planes: E, D, and H. Compute the pattern of the isolated element (dipole backed by reflector). Finally, use the equations from the previous section to generate the scan element pattern for the infinite array.

Perform all analysis at 10 GHz. To compare the patterns of finite and infinite array, overlay them on the same plot.

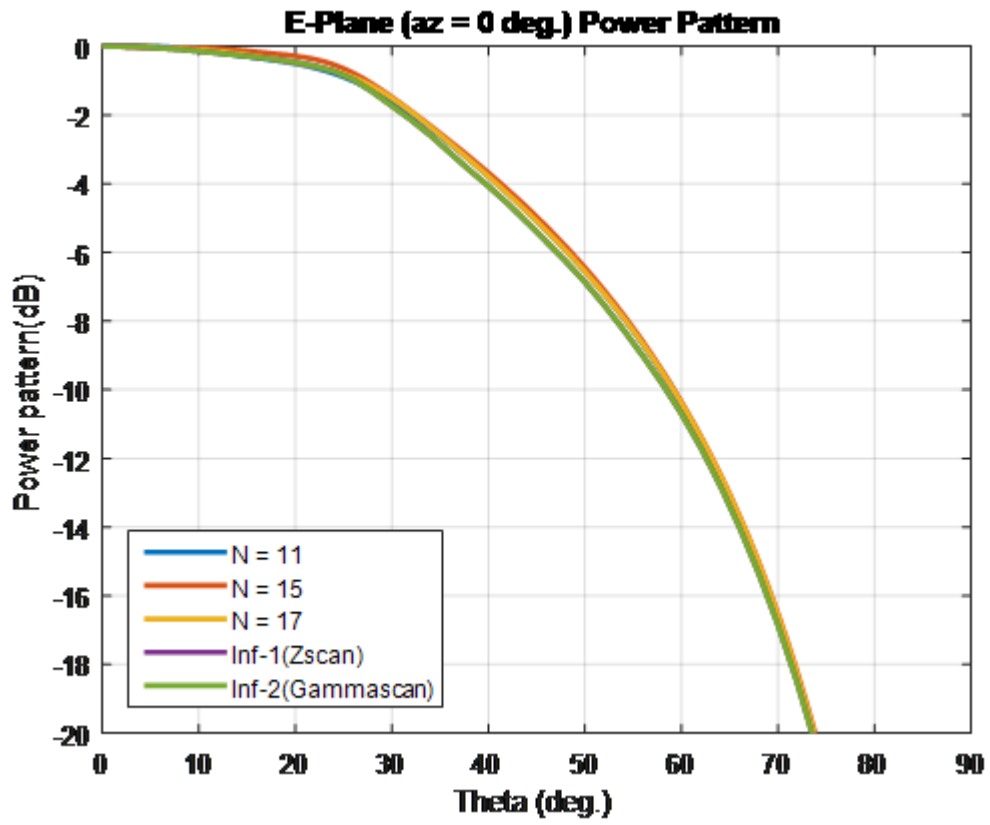


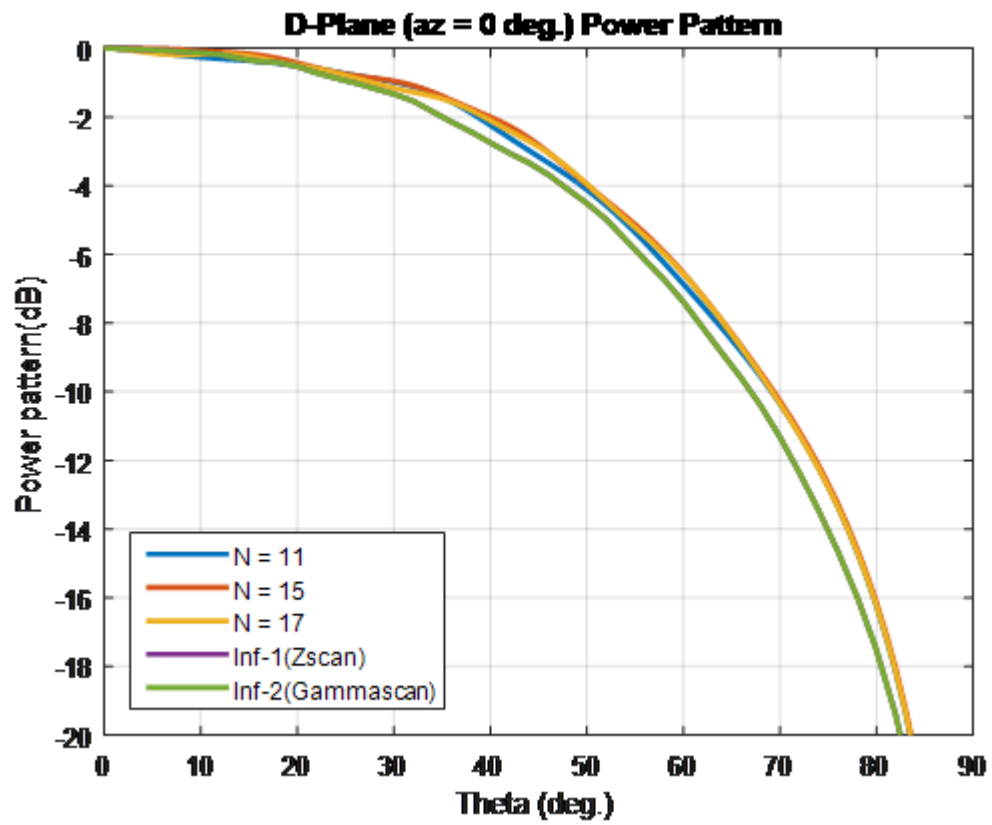


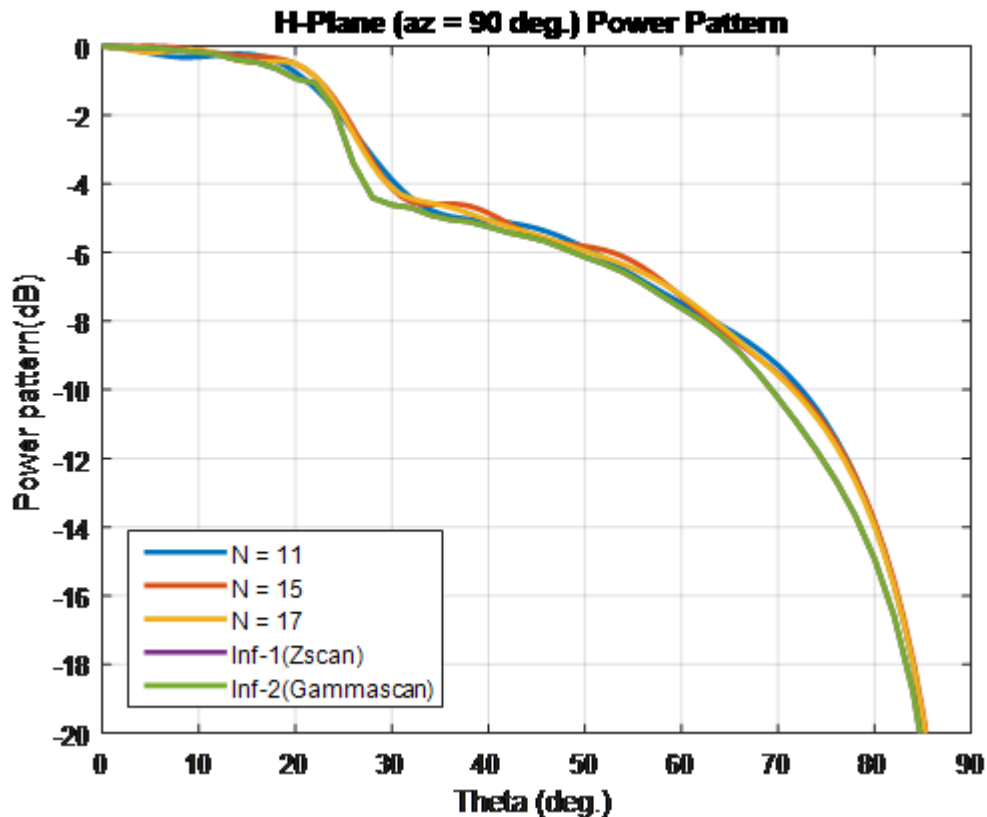


Case 2: Compare finite array and infinite array with unit cell of dimensions $0.7\lambda \times 0.7\lambda$

To compare the scan element pattern of these array types and infinite arrays, repeat the process in case 1. Using these unit cell dimensions creates grating lobes. Terminate the finite arrays using $86\text{-}\Omega$ resistance. For an infinite array with unit cell of dimensions $0.7\lambda \times 0.7\lambda$, the broadside resistance is $86\ \Omega$.







For finite arrays of size greater than 10 x 10, the scan element patterns in the E-, D-, and H-planes match the patterns of the infinite array scan element.

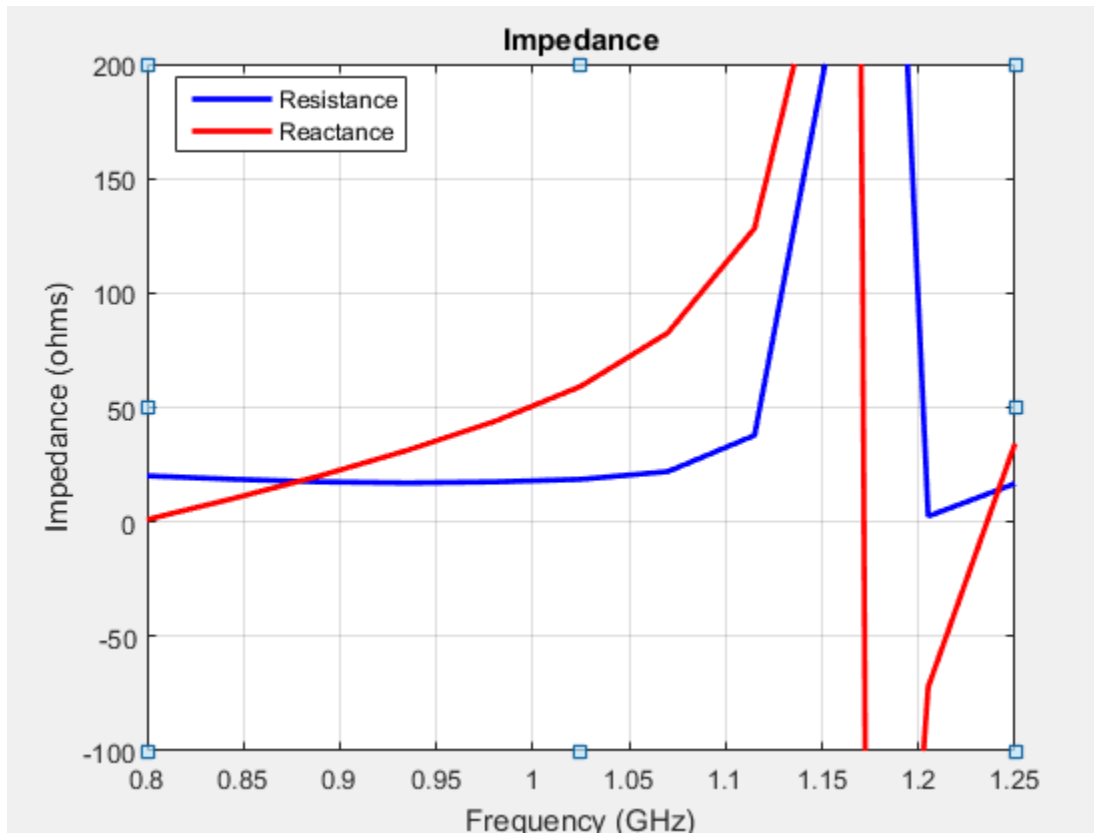
Impact of Infinite Double Summation

As show in the Green's equations, the periodic Green's function has an infinite double summation in (m, n) . When performing infinite array analysis, the number of terms in the double summation affects the accuracy of the final solution. Higher number of terms results in better accuracy but increases computation time.

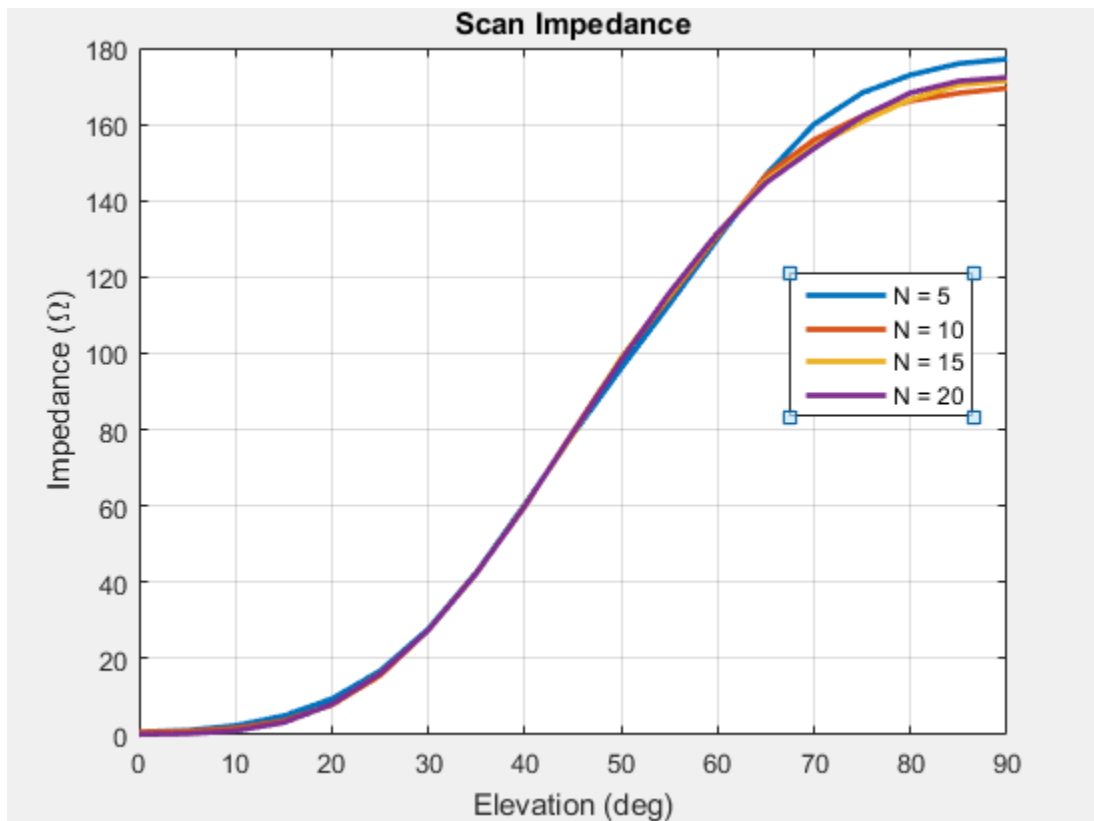
By default, Antenna Toolbox uses 10 terms for each summation term (m, n) to perform infinite array analysis. The total summation term length is $2*10+1$ (-10 to +10). To modify the number of terms, use the method `numSummationTerms`.

Higher number of terms are required if:

- You observe negative values for scan resistance for certain scan angles at certain frequencies.



- You must investigate for convergence when scan impedance shows slow variations.



References

- [1] Mailloux, R. J. *Phased Array Antenna Handbook*. Norwood, MA: Artech House. 2nd Edition. 2005.
- [2] Hansen, R. C. *Phased Array Antennas*. Hoboken, NJ: John Wiley & Sons Inc. 2nd Edition. 1998, pp. 221-313.
- [3] Allen, J. "Gain and impedance variation in scanned dipole arrays." *IRE Transactions on Antennas and Propagation*. Vol. 10, Number 5, September 1962, pp. 566-572.
- [4] Wasylkiwskyj, W., and W. Kahn. "Efficiency as a measure of size of a phased-array antenna." *IEEE Transactions on Antennas and Propagation*. Vol. 21, Number 6, November 1973, pp. 879-884.
- [5] Holter, H., and H. Steyskal. "On the size requirement for finite phased-array models." *IEEE Transactions on Antennas and Propagation*. Vol. 50, Number 6, June 2002, pp. 836-840.

Manipulate Array Elements

This example shows you how to control each individual element in a linear or rectangular array. You can use this technique to change the size and tilt of the antenna, or to model dead elements etc. with individual elements in an array.

Define Dipole Array

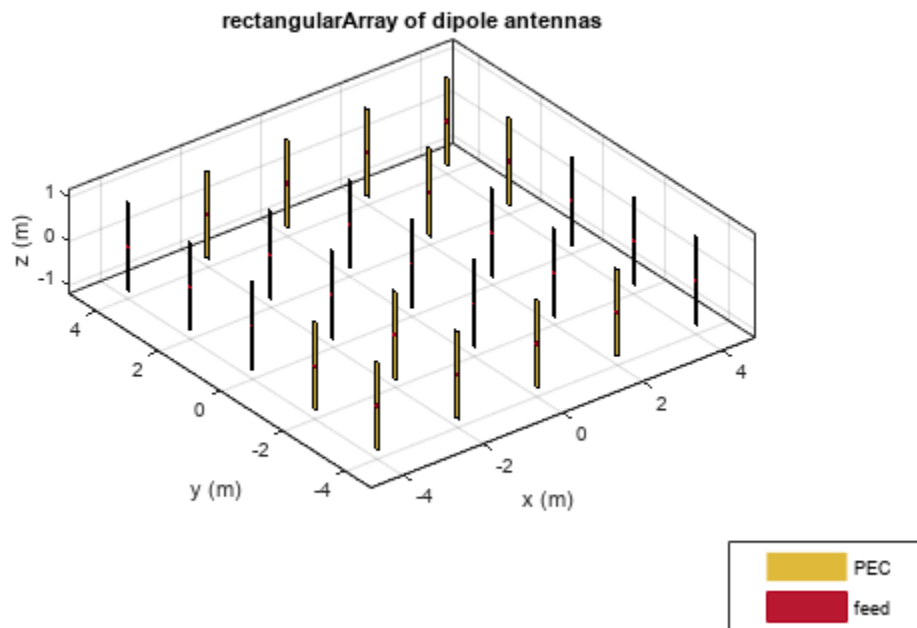
Create a dipole antenna using the `dipole` class. To create a 5x5 dipole array, replicate the dipole antenna using a 5x5 matrix. Create a rectangular array using the dipole array as a single element.

```
d = dipole;
N = 5;
df = repmat(d,N)

df =
    5x5 dipole array with properties:

        Length: {25x1 cell}
         Width: {25x1 cell}
    FeedOffset: {25x1 cell}
   Conductor: {25x1 cell}
         Tilt: {25x1 cell}
    TiltAxis: {25x1 cell}
         Load: {25x1 cell}

r = rectangularArray('Element',df);
show(r)
```



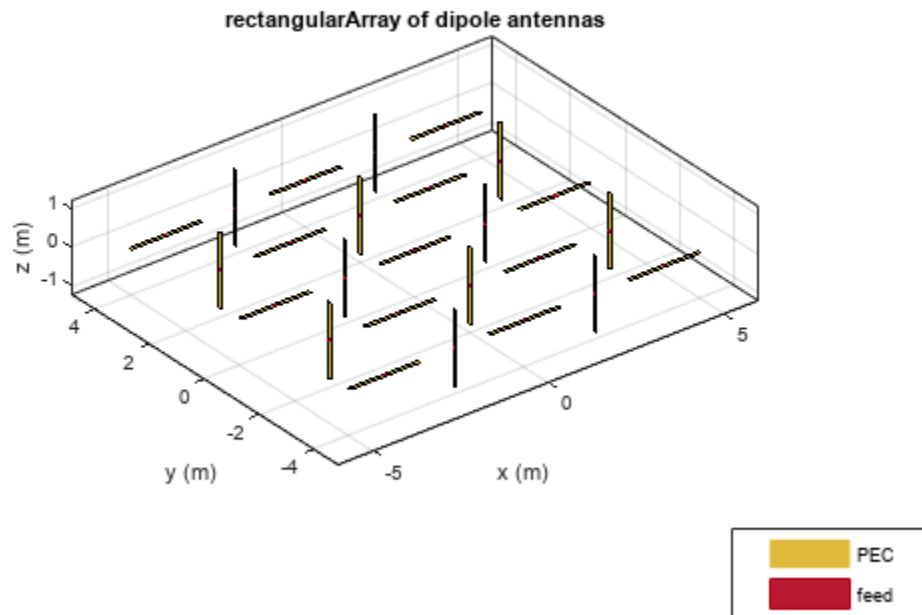
Tilt Alternate Elements

Create a vector of alternate elements in the rectangular array.

```
S = [1:2:N*N];
```

Tilt the alternate elements in the array by 90 degrees about the y-axis.

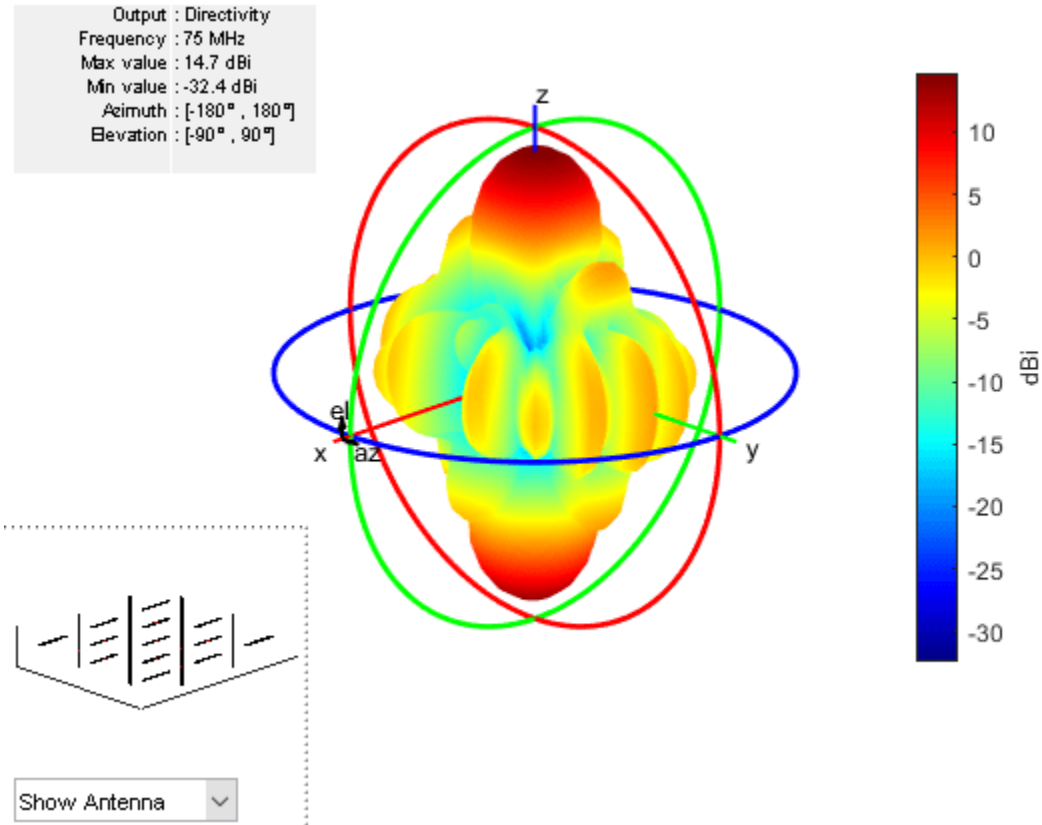
```
for i = 1:25
    if any(S==i)
        r.Element(i).Tilt = 90;
        r.Element(i).TiltAxis = [0 1 0];
    end
end
show(r)
```



Pattern of Rectangular Array

Plot the pattern of the array at 75 MHz.

```
pattern(r,75e6)
```



Model Dead Elements

Antennas with a zero excitation voltage feed are called dead elements. By default, each element in an array is excited by an amplitude of 1 V.

```
Vfeed = ones(1,N*N)
```

```
Vfeed = 1x25
```

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

To model dead elements, set the voltage to zero for all horizontal elements. To control voltage, use the `AmplitudeTaper` property. This property is the excitation amplitude of the antennas in an array.

```
Vfeed(S) = 0
```

```
Vfeed = 1x25
```

```
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
```

```
r.AmplitudeTaper = Vfeed
```

```
r =  

    rectangularArray with properties:
```

```
    Element: [5x5 dipole]
```

```

        Size: [5 5]
      RowSpacing: 2
    ColumnSpacing: 2
      Lattice: 'Rectangular'
    AmplitudeTaper: [0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0]
      PhaseShift: 0
        Tilt: 0
      TiltAxis: [1 0 0]

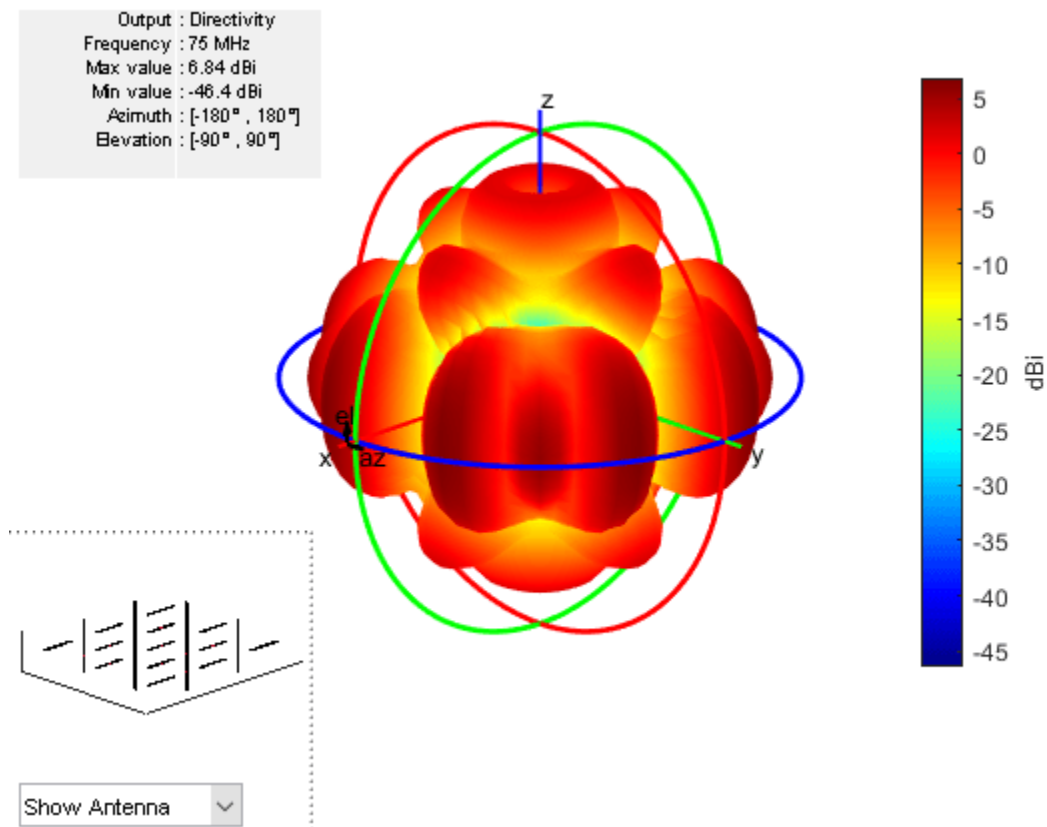
```

Plot the radiation pattern of the array.

```

figure
pattern(r,75e6)

```



Computational Techniques

- “Method of Moments Solver for Metal Structures” on page 3-2
- “Method of Moments Solver for Metal and Dielectric Structures” on page 3-9
- “Hybrid MoM-PO Method for Metal Antennas with Large Scatterers” on page 3-15
- “Physical Optics Solver” on page 3-20
- “Feed Model” on page 3-23
- “Antenna Optimization Algorithm” on page 3-29
- “Wire Solver” on page 3-31
- “Fast Multipole Method for Large Structures” on page 3-33
- “Finite Conductivity and Thickness Effect in MoM Solver” on page 3-38
- “Solvers” on page 3-41

Method of Moments Solver for Metal Structures

In this section...

“MoM Formulation” on page 3-2
 “Neighbor Region” on page 3-5
 “Singularity Extraction” on page 3-6
 “Finite Arrays” on page 3-7
 “Infinite Array” on page 3-7

Method of Moments computation technique for metal antennas.

The first step in the computational solution of electromagnetic problems is to discretize Maxwell's equations. The process results in this matrix-vector system:

$$V = ZI$$

- V - Applied voltage vector. This signal can be voltage or power applied to the antenna or an incident signal falling on the antenna.
- I - Current vector that represents current on the antenna surface.
- Z - Interaction matrix or impedance matrix that relates V to I .

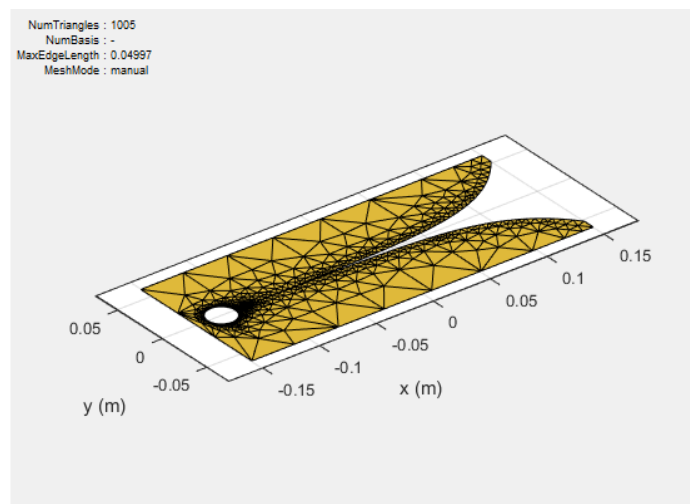
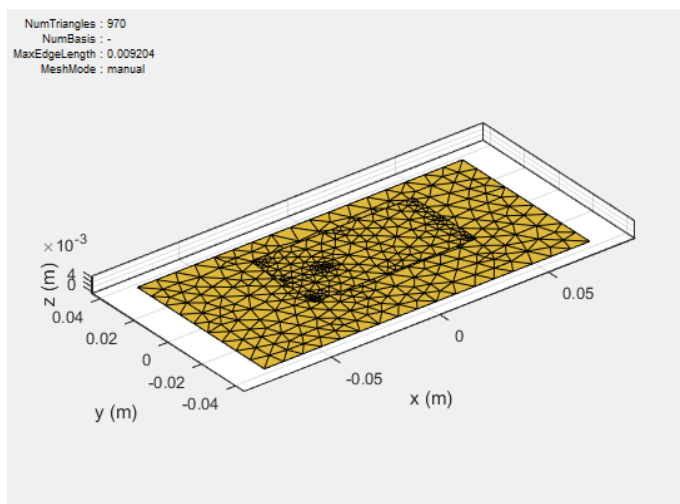
Antenna Toolbox uses method of moments (MoM) to calculate the interaction matrix and solve system equations.

MoM Formulation

The MoM formulation is split into three parts.

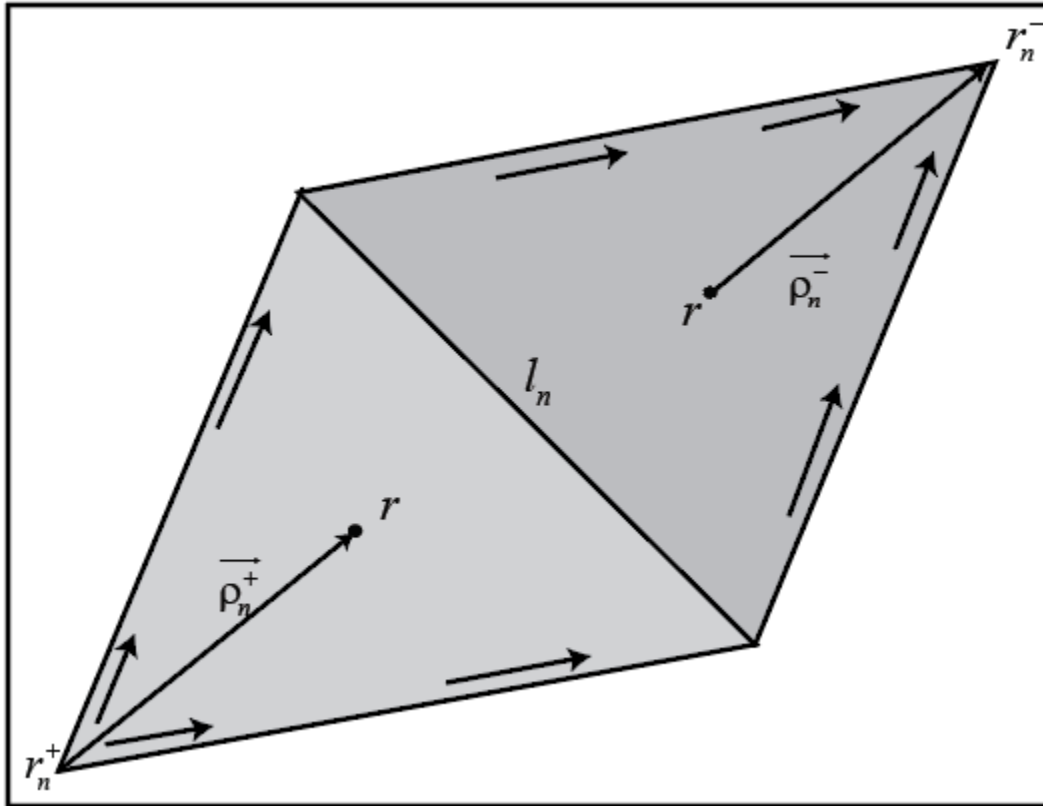
Discretization of Metals

Discretization enables the formulation from the continuous domain to the discrete domain. This step is called *meshing* in antenna literature. In the MoM formulation, the metal surface of the antenna is meshed into triangles.



Basis Functions

To calculate the surface currents on the antenna structure, you first define basis functions. Antenna Toolbox uses Rao-Wilton-Glisson (RWG) [2] basis functions. The arrows show the direction of current flow.



The basis function includes a pair of adjacent (not necessarily coplanar) triangles and resembles a small spatial dipole with linear current distribution. Each triangle is associated with a positive or negative charge.

For any two triangle patches, t_n^+ and t_n^- , having areas A_n^+ and A_n^- , and sharing common edge l_n , the basis function is

$$\vec{f}_n(\vec{r}) = \begin{cases} \frac{l_n}{2A_n^+} \vec{\rho}_n^{+S}, & \vec{r} \in t_n^+ \\ \frac{l_n}{2A_n^-} \vec{\rho}_n^{-S}, & \vec{r} \in t_n^- \end{cases}$$

- $\vec{\rho}_n^+ = \vec{r} - \vec{r}_n^+$ — Vector drawn from the free vertex of triangle t_n^+ to observation point \vec{r}
- $\vec{\rho}_n^- = \vec{r}_n^- - \vec{r}$ — Vector drawn from the observation point to the free vertex of the triangle t_n^-

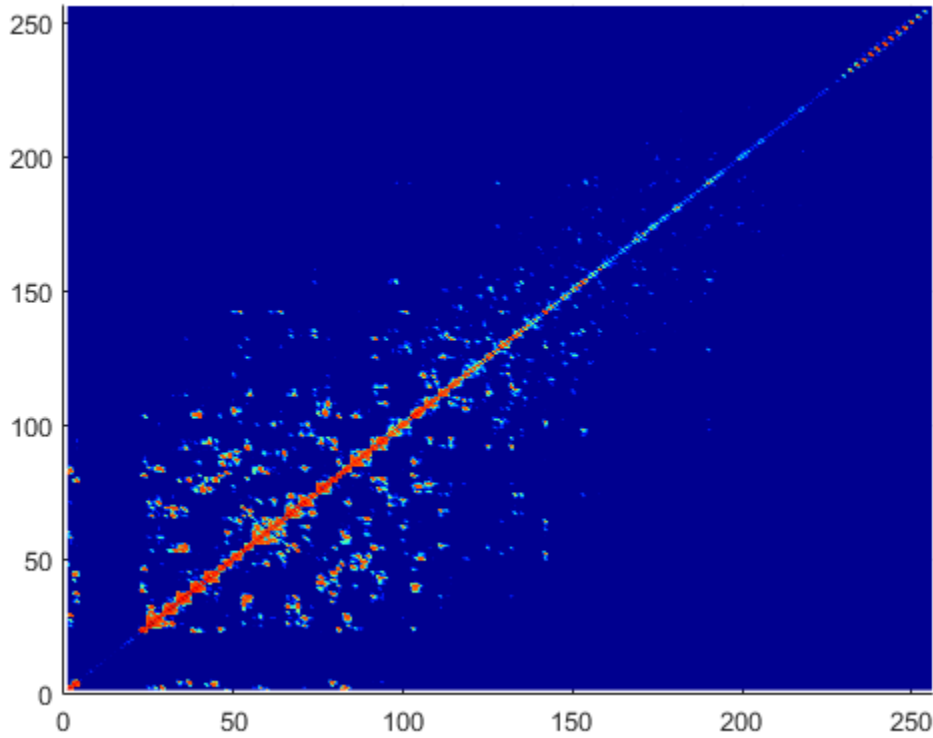
and

$$\nabla \cdot \vec{f}_n(\vec{r}) = \begin{cases} \frac{l_n}{A_n^+}, & \vec{r} \in t_n^+ \\ -\frac{l_n}{A_n^-}, & \vec{r} \in t_n^- \end{cases}$$

The basis function is zero outside the two adjacent triangles t_n^+ and t_n^- . The RWG vector basis function is linear and has no flux (no normal component) through its boundary.

Interaction Matrix

The interaction matrix is a complex dense symmetric matrix. It is a square N -by- N matrix, where N is the number of basis functions, that is, the number of interior edges in the structure. A typical interaction matrix for a structure with 256 basis functions is shown:



To fill out the interaction matrix, calculate the free-space Green's function between all basis functions on the antenna surface. The final interaction matrix equations are:

$$Z_{mn} = \left(\frac{j\omega\mu}{4\pi} \right) \int_S \int_S \vec{f}_m(\vec{r}) \cdot \vec{f}_m(\vec{r}') g d\vec{r}' d\vec{r} - \left(\frac{j}{4\pi\omega\epsilon} \right) \int_S \int_S (\nabla \cdot \vec{f}_m) (\nabla \cdot \vec{f}_m) g d\vec{r}' d\vec{r}$$

where

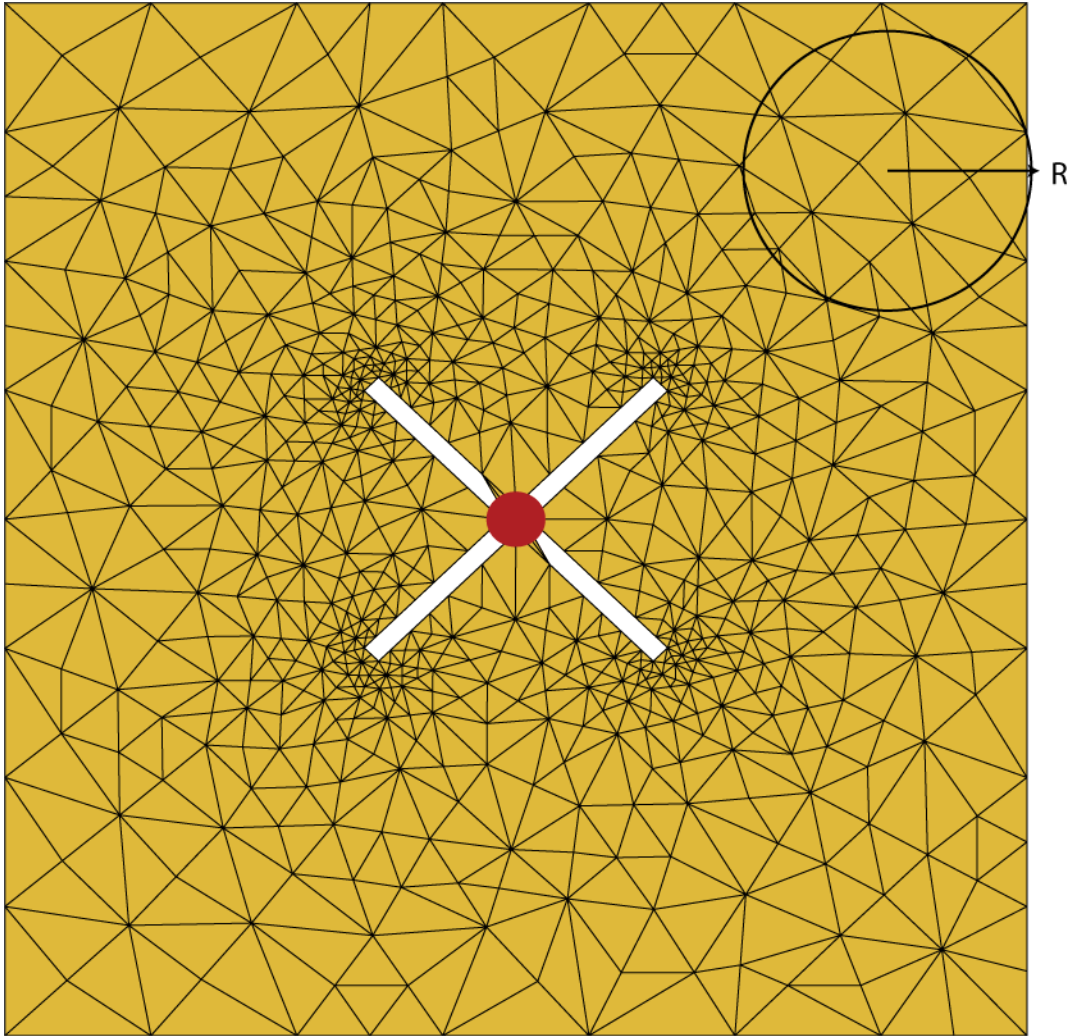
$$\bullet \quad g(\vec{r}, \vec{r}') = \frac{\exp(-jk|\vec{r} - \vec{r}'|)}{|\vec{r} - \vec{r}'|} \text{ — Free-space Green's function}$$

To calculate the interaction matrix, excite the antenna by a voltage of 1 V at the feeding edge. So the voltage vector has zero values everywhere except at the feeding edge. Solve the system of equations to calculate the unknown currents. Once you determine the unknown currents, you can calculate the field and surface properties of the antenna.

Neighbor Region

From the interaction matrix plot, you observe that the matrix is diagonally dominant. As you move further away from the diagonal, the magnitude of the terms decreases. This behavior is same as the Green's function behavior. The Green's function decreases as the distance between r and r' increases. Therefore, it is important to calculate the region on the diagonal and close to the diagonal accurately.

This region on and around the diagonal is called *neighbor region*. The neighbor region is defined within a sphere of radius R , where R is in terms of triangle size. The size of a triangle is the maximum distance from the center of the triangle to any of its vertices. By default, R is twice the size of the triangle. For better accuracy, a higher-order integration scheme is used to calculate the integrals.



Singularity Extraction

Along the diagonal, r and r' are equal and defines Green's function becomes singular. To remove the singularity, extraction is performed on these terms.

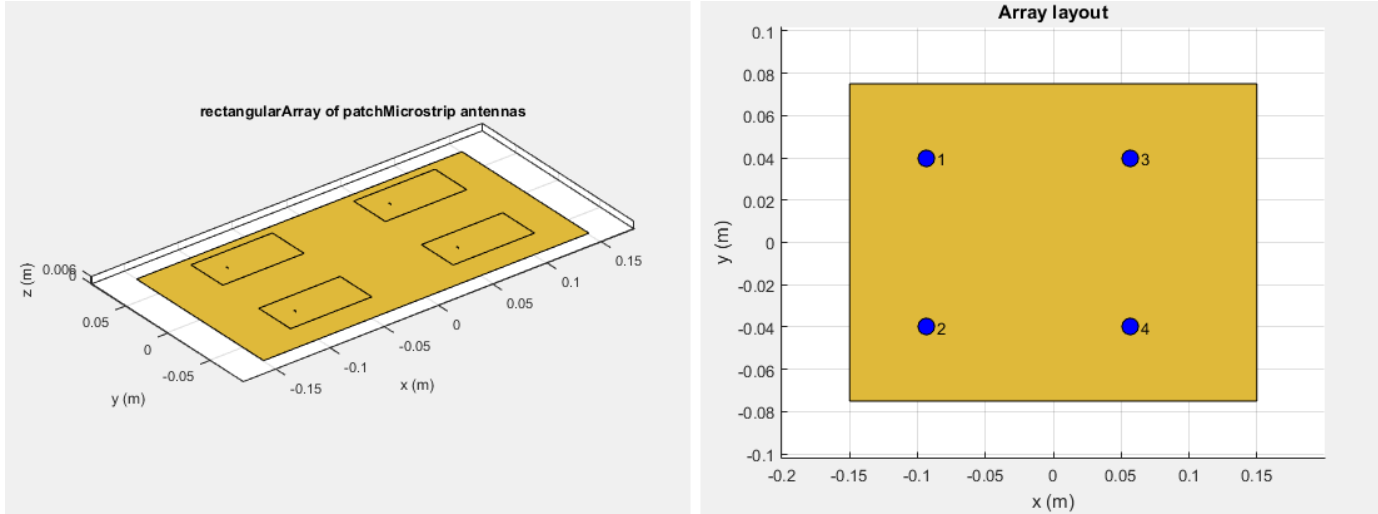
$$\int_{t_p} \int_{t_q} (\vec{\rho}_i \cdot \vec{\rho}'_j) g(\vec{r}, \vec{r}') ds' ds = \int_{t_p} \int_{t_q} \frac{(\vec{\rho}_i \cdot \vec{\rho}'_j)}{|\vec{r} - \vec{r}'|} ds' ds + \int_{t_p} \int_{t_q} \frac{(\exp(-jk|\vec{r} - \vec{r}'|) - 1)(\vec{\rho}_i \cdot \vec{\rho}'_j)}{|\vec{r} - \vec{r}'|} ds' ds$$

$$\int_{t_p} \int_{t_q} g(\vec{r}, \vec{r}') ds' ds = \int_{t_p} \int_{t_q} \frac{1}{|\vec{r} - \vec{r}'|} ds' ds + \int_{t_p} \int_{t_q} \frac{(\exp(-jk|\vec{r} - \vec{r}'|) - 1)}{|\vec{r} - \vec{r}'|} ds' ds$$

The two integrals on the right side of the equations, called potential or static integrals are found using analytical results [3].

Finite Arrays

The MoM formulation for finite arrays is the same as for a single antenna element. The main difference is the number of excitations (feeds). For finite arrays, the voltage vector is now a voltage matrix. The number of columns are equal to the number of elements in the array.



For example, the voltage vector matrix for a 2x2 array of rectangular patch antenna has four columns as each antenna can be excited separately.

Infinite Array

To model an infinite array, you change the MoM to account for the infinite behavior. To do so, you replace the free-space Green's functions with periodic Green's functions. The periodic Green's function is an infinite double summation.

Green's Function	Periodic Green's Function
$g = \frac{e^{-jkR}}{R}$ $R = \vec{r} - \vec{r}' $	$g_{\text{periodic}} = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} e^{j\phi_{mn}} \frac{e^{-jkR_{mn}}}{R_{mn}}$ $R_{mn} = \sqrt{(x - x' - x_m)^2 + (y - y' - y_n)^2 + (z - z')^2}$ $\phi_{mn} = -k(x_m \sin\theta \cos\varphi + y_n \sin\theta \sin\varphi)$ $x_m = m \cdot d_x, y_n = n \cdot d_y$

d_x and d_y are the ground plane dimensions that define the x and y dimensions of the unit cell. θ and φ are the scan angles.

Comparing the two Green's functions, you observe an additional exponential term that is added to the infinite sum. The ϕ_{mn} accounts for the scanning of the infinite array. The periodic Green's function also accounts for the effect of mutual coupling.

For more information see, "Infinite Arrays" on page 2-22.

References

- [1] Harrington, R. F. *Field Computation by Moment Methods*. New York: Macmillan, 1968.
- [2] Rao, S. M., D. R. Wilton, and A. W. Glisson. "Electromagnetic scattering by surfaces of arbitrary shape." *IEEE. Trans. Antennas and Propagation*, Vol. AP-30, No. 3, May 1982, pp. 409-418.
- [3] Wilton, D. R., S. M. Rao, A. W. Glisson, D. H. Schaubert, O. M. Al-Bundak, and C. M. Butler. "Potential Integrals for uniform and linear source distribution on polygonal and polyhedral domains." *IEEE. Trans. Antennas and Propagation*. Vol. AP-30, No. 3, May 1984, pp. 276-281.
- [4] Balanis, C.A. *Antenna Theory. Analysis and Design*. 3rd Ed. New York: John Wiley & Sons, 2005.

See Also

"Finite Conductivity and Thickness Effect in MoM Solver" on page 3-38 | "Infinite Arrays" on page 2-22

Method of Moments Solver for Metal and Dielectric Structures

In this section...

“MoM Formulation” on page 3-9
 “Neighbor Region” on page 3-11
 “Singularity Extraction” on page 3-12
 “Finite Arrays” on page 3-13
 “Infinite Array” on page 3-13

Method of Moments computation technique for metal and dielectric antennas.

Antennas using dielectric substrate consists of a metal part and a dielectric part. The first step in the computational solution of electromagnetic problems is to discretize Maxwell's equations. The process results in this matrix-vector system:

$$V = ZI$$

- V – Applied voltage vector. This signal can be voltage or power applied to the antenna or an incident signal falling on the antenna.
- I – Current vector that represents current on the antenna surface.
- Z – Interaction matrix or impedance matrix that relates V to I . For calculating the interaction matrix, the effect of metal and dielectric parts in an antenna are taken separately.

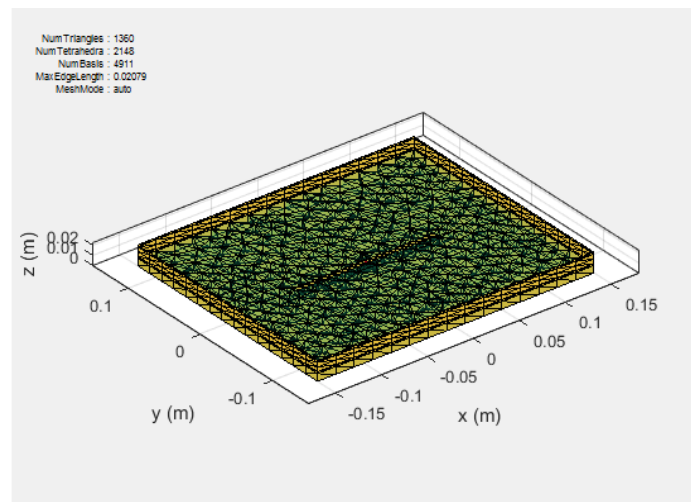
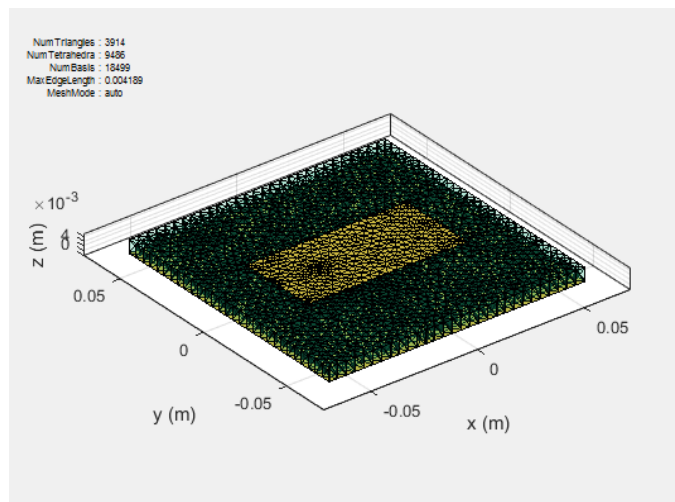
Antenna Toolbox uses method of moments (MoM) to calculate the interaction matrix and solve system equations.

MoM Formulation

The MoM formulation is split into three parts.

Discretization of Dielectrics

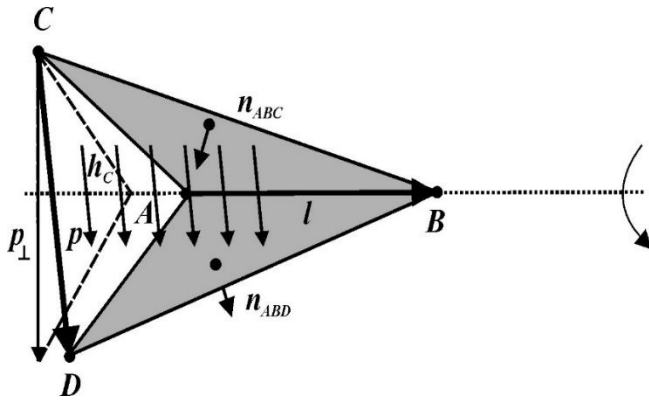
Discretization enables the formulation from the continuous domain to the discrete domain. This step is called *meshing* in antenna literature. In the MoM formulation, the metal surface of the antenna is meshed into triangles and the dielectric volume is meshed into tetrahedrons.



Basis Functions

Basis functions are used to represent unknown quantities. In the case of antennas using dielectrics, the unknown quantities are the surface current on the metal structure and flux density due to dielectric volume. Antenna Toolbox uses Rao-Wilton-Glisson (RWG) [2] basis functions. For basis functions for a metal structure in an antenna refer, “Method of Moments Solver for Metal Structures” on page 3-2.

For the dielectric volume of the antenna, Antenna Toolbox uses a zeroth order edge basis function to model the flux density.



The figure shows an edge-based basis function. The vector variation is perpendicular to the base edge AB (or \bar{l}). The vector of the edge CD (or \bar{p}) defines the basis function. Within a tetrahedron, the basis function is a constant field given by

$$\vec{f} = c\vec{p}$$

- c - normalization coefficient.
- p - vector of the edge defining the basis function.

Interaction Matrix

The interaction matrix is a complex dense symmetric matrix. For a metal-dielectric antenna, there are two sets of basis functions and four interactions. To fill out the interaction matrix, calculate the free-space Green's function between all the basis functions on the antenna surface. The final interaction matrix equations are:

- Z_{MM} - metal to metal interaction. For a pure metal structure, you only calculate this symmetric square matrix.

$$Z_{mn}^{MM} = \left(\frac{j\omega\mu}{4\pi}\right) \int_S \int_S \vec{f}_m^M(\vec{r}) \cdot \vec{f}_n^M(\vec{r}') g d\vec{r} d\vec{r}' - \left(\frac{j}{4\pi\omega\epsilon}\right) \int_S \int_S (\nabla \cdot \vec{f}_m^M) (\nabla \cdot \vec{f}_n^M) g d\vec{r} d\vec{r}'$$

- Z_{DD} - dielectric to dielectric interaction. For pure dielectric structures, you only calculate this symmetric square matrix.

$$\begin{aligned}
 \widehat{Z}_{mn}^{DD} &= \sum_{p=1}^P \sum_{p'=1}^{P'} \frac{K_p}{\widehat{\varepsilon}_p} \int_{V_D} \vec{f}_{mp}(\vec{r}) \cdot \vec{f}_{np'}(\vec{r}) d\vec{r} \\
 &\quad - \frac{\omega^2 \mu_0}{4\pi} \sum_{p=1}^P \sum_{p'=1}^{P'} K_p K_{p'} \int_{V_D} \int_{V_{D'}} g(\vec{r}, \vec{r}') \vec{f}_{mp}(\vec{r}) \cdot \vec{f}_{np'}(\vec{r}') d\vec{r} d\vec{r}' \\
 &\quad - \frac{1}{4\pi \varepsilon_0} \sum_{q=1}^Q \sum_{q'=1}^{Q'} \widehat{K}_q \widehat{K}_{q'} \int_{\Omega_q} \int_{\Omega_{q'}} g(\vec{r}, \vec{r}') f_{\perp mq}(\vec{r}) f_{\perp nq'}(\vec{r}') d_s d_{s'} \quad m, n = 1, \dots, N
 \end{aligned}$$

- Z_{MD} and Z_{DM} - These matrices calculate the interaction between metal and dielectric. This matrix is not a symmetrical square matrix.

$$\begin{aligned}
 Z_{mn}^{MD} &= -\frac{\omega^2 \mu_0}{4\pi} \sum_{p=1}^P \sum_{p'=1}^{P'} K_p \int_{V_D} \int_{V_{D'}} \vec{f}_n^M(\vec{r}') \cdot \vec{f}_{mp'}(\vec{r}) g(\vec{r}, \vec{r}') d\vec{r}' d_s \\
 &\quad - \frac{1}{4\pi \varepsilon_0} \sum_{p=1}^P \sum_{q=1}^Q \widehat{K}_q \int_{\Omega_q} \int_{\Omega_{q'}} (\nabla_s \cdot \vec{f}_n^M(\vec{r}')) f_{\perp mq}(\vec{r}) g(\vec{r}, \vec{r}') d_{\Omega} d_s \\
 &\quad m = 1, \dots, N_D; n = 1, \dots, N_M
 \end{aligned}$$

$$\begin{aligned}
 Z_{mn}^{DM} &= -\frac{j\omega \mu_0}{4\pi} \sum_{p=1}^P \sum_{p'=1}^{P'} K_{p'} \int_{V_D} \int_{S_{D'}} \vec{f}_{np'}(\vec{r}) \cdot \vec{f}_m^M(\vec{r}') g(\vec{r}, \vec{r}') d_s d\vec{r}' \\
 &\quad + \frac{1}{4\pi \varepsilon_0 \omega} \sum_{p=1}^P \sum_{q=1}^Q \widehat{K}_q \int_{\Omega_q} \int_{S_{D'}} f_{\perp nq}(\vec{r}) \cdot (\nabla_s \cdot \vec{f}_m^M(\vec{r}')) g(\vec{r}, \vec{r}') d_s d_{\Omega} \\
 &\quad m = 1, \dots, N_D; n = 1, \dots, N_M
 \end{aligned}$$

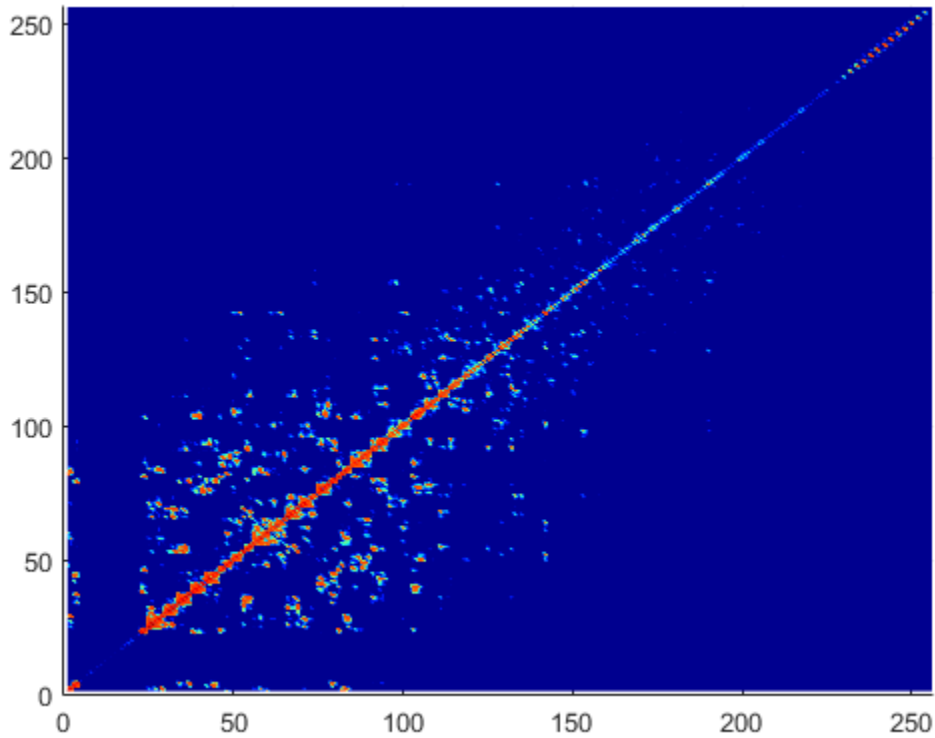
where

- $g(\vec{r}, \vec{r}') = \frac{\exp(-jkR)}{R}$, $R = |\vec{r} - \vec{r}'|$ is the free space Green's function.
- $K = \frac{\widehat{\varepsilon}^{\pm} - \varepsilon_0}{\widehat{\varepsilon}^{\pm}}$ is the complex dielectric constant within every tetrahedron.
- $\widehat{K}_q = K_+ - K_-$ is the differential contrast on every face of the tetrahedron.

For a composite metal structure, you must calculate all four matrices.

Neighbor Region

The figure shows a typical interaction matrix for a metal structure Z_{MM} with 256 basis functions.



From the interaction matrix plot, you observe that the matrix is diagonally dominant. The dielectric interaction matrix is also diagonally dominant. As you move further away from the diagonal, the magnitude of the terms decreases. This behavior is same as the Green's function behavior. The Green's function decreases as the distance between r and r' increases. Therefore, it is important to calculate the region on the diagonal and close to the diagonal accurately.

This region on and around the diagonal is called *neighbor region*. For a metal-dielectric antenna, the neighborhood region is based on the average size of the tetrahedron.

For neighboring region details for metal antennas refer, "Method of Moments Solver for Metal Structures" on page 3-2.

Singularity Extraction

Along the diagonal, r and r' are identical and the defined Green's function becomes singular. To remove the singularity, extraction is performed on these terms. The equations for the singularity extraction of the Z_{MM} matrix are:

$$\int_{t_p} \int_{t_q} (\vec{\rho}_i \cdot \vec{\rho}'_j) g(\vec{r}, \vec{r}') ds' ds = \int_{t_p} \int_{t_q} \frac{(\vec{\rho}_i \cdot \vec{\rho}'_j)}{|\vec{r} - \vec{r}'|} ds' ds + \int_{t_p} \int_{t_q} \frac{(\exp(-jk|\vec{r} - \vec{r}'|) - 1)(\vec{\rho}_i \cdot \vec{\rho}'_j)}{|\vec{r} - \vec{r}'|} ds' ds$$

$$\int_{t_p} \int_{t_q} g(\vec{r}, \vec{r}') ds' ds = \int_{t_p} \int_{t_q} \frac{1}{|\vec{r} - \vec{r}'|} ds' ds + \int_{t_p} \int_{t_q} \frac{(\exp(-jk|\vec{r} - \vec{r}'|) - 1)}{|\vec{r} - \vec{r}'|} ds' ds$$

The two integrals on the right side of the equations, called potential or static integrals are found using analytical results [3].

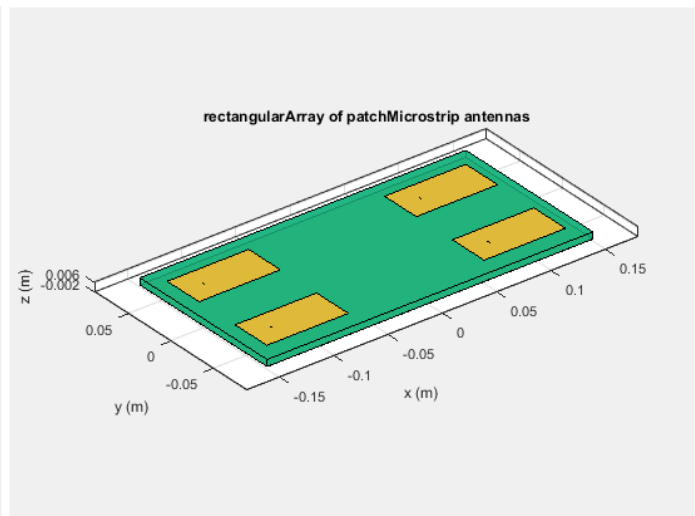
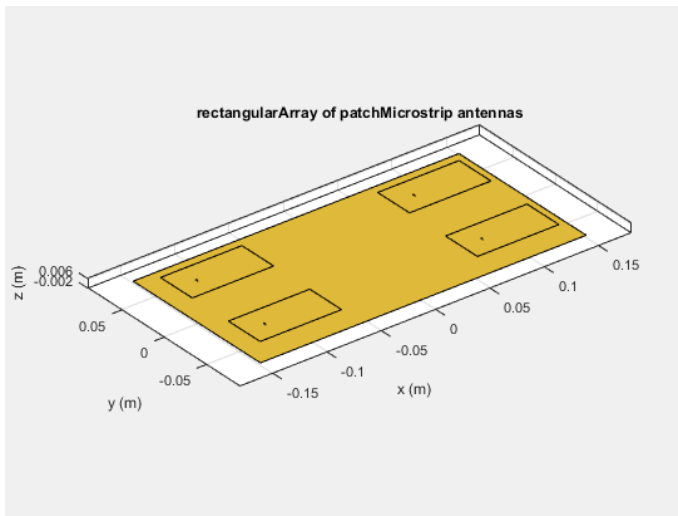
The equations for the singularity extraction of the Z_{DD} matrix are:

$$\int_{V_D} \int_{V_{D'}} g(\vec{r}, \vec{r}') d\vec{r} d\vec{r}' = \int_{V_D} \int_{V_q} \frac{1}{|\vec{r} - \vec{r}'|} d\vec{r} d\vec{r}' + \int_{V_D} \int_{V_q} \frac{(\exp(-jk|\vec{r} - \vec{r}'|) - 1)}{|\vec{r} - \vec{r}'|} d\vec{r} d\vec{r}'$$

$$\int_{\Omega_q} \int_{\Omega_{q'}} g(\vec{r}, \vec{r}') d\Omega d\Omega' = \int_{\Omega_D} \int_{\Omega_q} \frac{1}{|\vec{r} - \vec{r}'|} d\Omega d\Omega' + \int_{S_D} \int_{S_q} \frac{(\exp(-jk|\vec{r} - \vec{r}'|) - 1)}{|\vec{r} - \vec{r}'|} d\Omega d\Omega'$$

Finite Arrays

The MoM formulation for finite arrays is the same as for a single antenna element. The main difference is the number of excitations (feeds). For finite arrays, the voltage vector is now a voltage matrix. The number of columns are equal to the number of elements in the array.



For example, the voltage vector matrix for a 2x2 array of rectangular patch antenna (with and without dielectric substrate) has four columns as each antenna can be excited separately.

Infinite Array

To model an infinite array, you change the MoM to account for the infinite behavior. To do so, you replace the free-space Green's functions with periodic Green's functions. The periodic Green's function is an infinite double summation.

Green's Function	Periodic Green's Function
$g = \frac{e^{-jkR}}{R}$ $R = \vec{r} - \vec{r}' $	$g_{\text{periodic}} = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} e^{j\phi_{mn}} \frac{e^{-jkR_{mn}}}{R_{mn}}$ $R_{mn} = \sqrt{(x - x' - x_m)^2 + (y - y' - y_n)^2 + (z - z')^2}$ $\phi_{mn} = -k(x_m \sin\theta \cos\varphi + y_n \sin\theta \sin\varphi)$ $x_m = m \cdot d_x, y_n = n \cdot d_y$

d_x and d_y are the ground plane dimensions that define the x and y dimensions of the unit cell. θ and Φ are the scan angles.

Comparing the two Green's functions, you observe an additional exponential term that is added to the infinite sum. The ϕ_{mn} accounts for the scanning of the infinite array. The periodic Green's function also accounts for the effect of mutual coupling.

For more information see, "Infinite Arrays" on page 2-22.

References

- [1] Harrington, R. F. *Field Computation by Moment Methods*. New York: Macmillan, 1968.
- [2] Rao, S. M., D. R. Wilton, and A. W. Glisson. "Electromagnetic scattering by surfaces of arbitrary shape." *IEEE. Trans. Antennas and Propagation*, Vol. AP-30, No. 3, May 1982, pp. 409-418.
- [3] Wilton, D. R., S. M. Rao, A. W. Glisson, D. H. Schaubert, O. M. Al-Bundak. and C. M. Butler. "Potential Integrals for uniform and linear source distribution on polygonal and polyhedral domains." *IEEE. Trans. Antennas and Propagation*. Vol. AP-30, No. 3, May 1984, pp. 276-281.
- [4] Balanis, C.A. *Antenna Theory. Analysis and Design*. 3rd Ed. New York: John Wiley & Sons, 2005.

See Also

More About

- "Infinite Arrays" on page 2-22
- "Method of Moments Solver for Metal Structures" on page 3-2
- "Finite Conductivity and Thickness Effect in MoM Solver" on page 3-38

Hybrid MoM-PO Method for Metal Antennas with Large Scatterers

In this section...

“Subdomain RWG Basis Functions and Extra Dimensions” on page 3-15

“MoM Region and PO Region” on page 3-16

“MoM Solution and PO Solution” on page 3-16

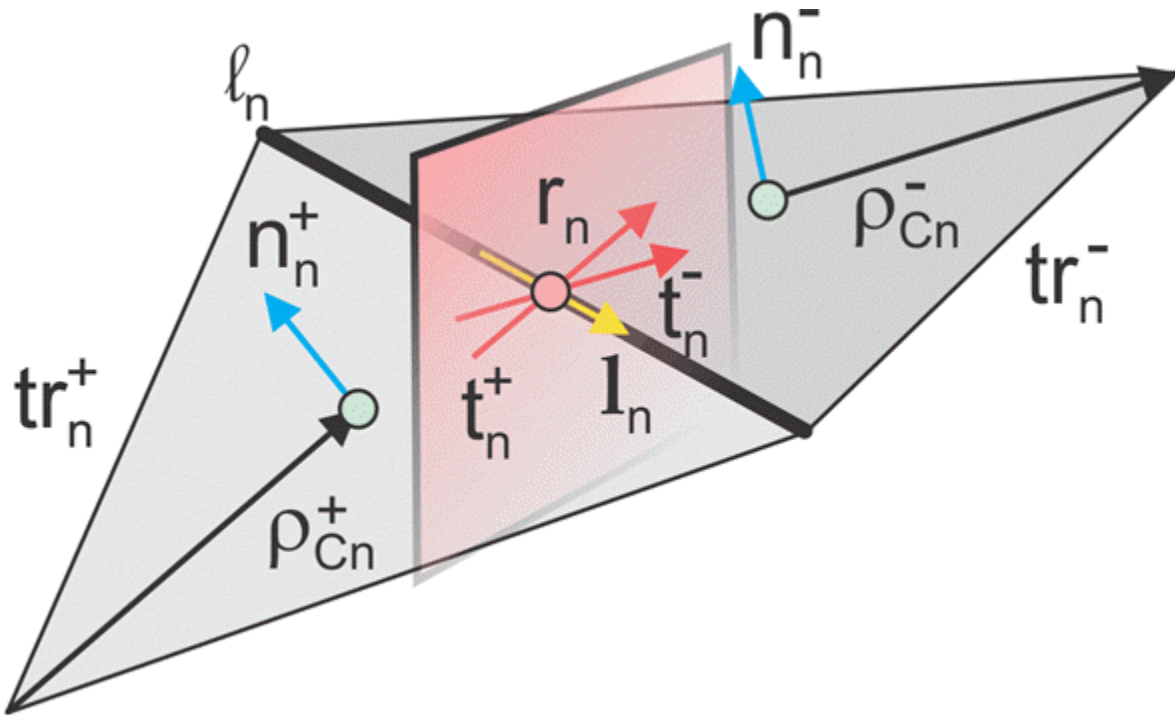
“Finding ZPO” on page 3-17

“Direct Solution Method” on page 3-18

Hybrid method of moments (MoM) and physical optics (PO) computational technique in Antenna Toolbox allows you to model antennas near large scatterers such as parabolic reflectors. The antenna element is modeled using MoM while the effect of electrically large structures is considered using PO.

Subdomain RWG Basis Functions and Extra Dimensions

The familiar Rao-Wilton-Glisson (RWG) basis functions on triangles are based on [2].



In the image, for two arbitrary triangular patches tr_n^+ and tr_n^- having areas A_n^+ and A_n^- and sharing a common edge l_n the basis functions has the form

$$\vec{f}_n(\vec{r}) = \begin{cases} \frac{l_n}{2A_n^+} \vec{\rho}_n^+ & \vec{r} \text{ in } tr_n^+ \\ \frac{l_n}{2A_n^-} \vec{\rho}_n^- & \vec{r} \text{ in } tr_n^- \end{cases} \quad (1)$$

where $\vec{\rho}_n^+ = \vec{r} - \vec{r}_n^-$ is the vector drawn from the free vertex of the triangle tr_n^+ to the observation point \vec{r} ; $\vec{\rho}_n^- = \vec{r}_n^- - \vec{r}$ is the vector drawn from the observation point to the free vertex of the triangle tr_n^- . The basis function is zero outside the two adjacent triangles. The RWG vector basis function is linear and has no flux (that is, has no normal component) through its boundary.

From [1], along with the standard definition, this method requires two unit normal vectors \vec{n}_n^\pm and two-unit vectors \vec{t}_n^\pm also shown in the figure. Vector \vec{t}_n^+ is the plane of triangle tr_n^+ ; both vectors are perpendicular to the edge l_n . They are defined at the center of edge l_n , which is denoted by \vec{r}_n .

Directions of \vec{t}_n^\pm

are also shown in the figure. This technique assumes that the normal vectors are properly (angle between adjacent \vec{n}_n^\pm must be less than 180 degrees) and uniquely defined. Specific vector orientation (e.g. outer or inner normal vectors) does not matter. We then form two cross product vectors \vec{l}_n^\pm ,

$$\vec{l}_n^\pm = \vec{t}_n^\pm \times \vec{n}_n^\pm \quad (2)$$

and establish that both such unit vectors directed along the edge are identical,

$$\vec{l}_n^\pm = \vec{l}_n^- = \vec{l}_n \quad (3)$$

Only vector \vec{l}_n is eventually needed.

MoM Region and PO Region

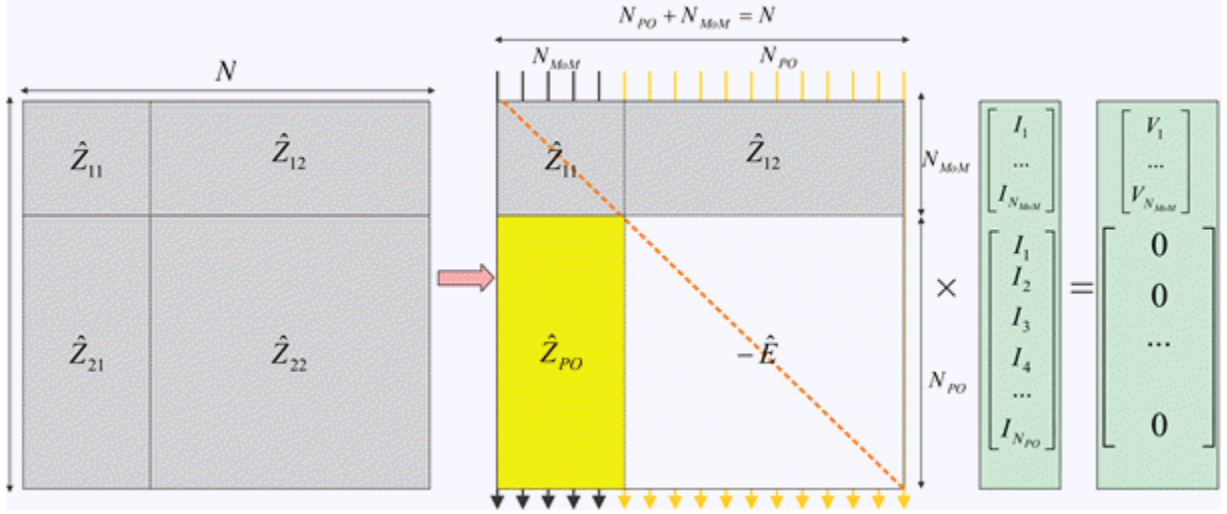
The surface current density, $\vec{J}(\vec{r})$, on the entire metal surface is expanded into N RWG basis functions. However, a part of such basis functions belongs to the MoM region (or "exact region") while another part will belong to the PO region (or "approximate region"). These basis functions (or regions) can overlap and be arbitrarily distributed in space (not necessarily be contiguous). The method assumes that N_{MoM} basis functions from the MoM region up front in the list and N_{PO} basis functions for the PO region afterward. Therefore, you have ($N_{PO} + N_{MoM} = N$)

$$\vec{J}(\vec{r}) = \sum_{n=1}^{N_{MoM}} I_n^{MoM} \vec{f}_n(\vec{r}), \quad \vec{J}(\vec{r}) = \sum_{n=1}^{N_{PO}} I_n^{PO} \vec{f}_{n+N_{MoM}}(\vec{r}) \quad (4)$$

MoM Solution and PO Solution

If there is no PO region, you can solve the entire problem using the MoM with single square MoM system matrix \widehat{Z} , which may be subdivided into 4 matrices as shown.

$$\widehat{Z} = \begin{pmatrix} \widehat{Z}_{11} & \widehat{Z}_{12} \\ \widehat{Z}_{21} & \widehat{Z}_{22} \end{pmatrix}, \quad \dim(\widehat{Z}_{11}) = N_{MoM} \times N_{MoM}, \quad \dim(\widehat{Z}_{12}) = N_{MoM} \times N_{PO} \quad (5)$$



The figure shows the matrix interpretation of the hybrid MoM-PO solution and its comparison with plain MoM solution. The method assumes the antenna feeds gives the vector, \vec{V} that describes the excitation, which belongs to the MoM region only.

The hybrid solution keeps submatrices \hat{Z}_{11} and \hat{Z}_{12} . In other words, the method solves the standard system of the linear equations for the MoM region where radiation from the PO region via \hat{Z}_{12} is considered.

The hybrid solution ignores the submatrices, \hat{Z}_{22} entirely. Here, the currents in the PO region do not interact with each other. They are found via the radiated magnetic field, $\vec{H}(\vec{r})$, from the MoM region, using PO approximation [1]. A new matrix describes this operation, \hat{Z}_{PO} , and negative identity matrix, E , which replaces \hat{Z}_{22} .

Finding ZPO

The suitable PO approximation has the form [1]

$$\vec{J}(\vec{r}) = 2\delta(\vec{r})[\vec{n}(\vec{r}) \times \vec{H}(\vec{r})] \quad (6)$$

where δ accounts for the shadowing effects. If the observation point lies in the shadow region, δ must be zero. Otherwise it equals ± 1 depending on the direction of incidence with respect to the orientation normal vector $\vec{n}(\vec{r})$. Using second Eq.(4) yields:

$$\sum_{n=1}^{N_{PO}} I_n^{PO} \vec{f}_{n+N_{MoM}}(\vec{r}) = 2\delta(\vec{r})[\vec{n}(\vec{r}) \times \vec{H}(\vec{r})] \quad (7)$$

Reference [1] outlines an elegant way to express unknowns I_n^{PO} explicitly, using an interesting variation of the collocation method. First, we consider a collocation point that tends to the edge center $\vec{r}_{n+N_{MoM}}$ of a certain basis function $\vec{f}_{n+N_{MoM}}(\vec{r})$ and is in its plus triangle. We then multiply Eq. (7) by vector $\vec{t}_{n+N_{MoM}}^+$. Since the normal component of the basis function under interest at the

edge is one and all other basis functions sharing the same triangle have no normal component at this edge, the result becomes

$$I_n^{PO} = 2\delta(\vec{r}_{n+N_{MoM}}) \vec{t}_{n+N_{MoM}}^+ \cdot \left[\vec{n}_{n+N_{MoM}}^+ \times \vec{H}(\vec{r}_{n+N_{MoM}}) \right] \quad (8a)$$

Repeat the same operation with the minus triangle and obtain

$$I_n^{PO} = 2\delta(\vec{r}_{n+N_{MoM}}) \vec{t}_{n+N_{MoM}}^- \cdot \left[\vec{n}_{n+N_{MoM}}^- \times \vec{H}(\vec{r}_{n+N_{MoM}}) \right] \quad (8b)$$

Add both Eqs. (8a) and (8b) together, divide the result by two, and transform the triple vector product to obtain

$$I_n^{PO} = 2\delta(\vec{r}_{n+N_{MoM}}) \vec{H}(\vec{r}_{n+N_{MoM}}) \cdot \left(\left[\vec{t}_{n+N_{MoM}}^+ \times \vec{n}_{n+N_{MoM}}^+ \right] + \left[\vec{t}_{n+N_{MoM}}^- \times \vec{n}_{n+N_{MoM}}^- \right] \right) / 2 \quad (9)$$

Therefore, according to Eqs. (2) and (3),

$$I_n^{PO} = 2\delta(\vec{r}_{n+N_{MoM}}) \vec{H}(\vec{r}_{n+N_{MoM}}) \cdot \vec{l}_{n+N_{MoM}} \quad (10)$$

To complete the derivation, the H-field radiated by the MoM region is always written in the form

$$\vec{H}(\vec{r}) = \sum_{n=1}^{N_{MoM}} \vec{C}_n(\vec{r}) I_n^{MoM} \quad (11)$$

where $\vec{C}_n(r)$ are given by individual basis function contributions. In the simplest case, every such contribution is the dipole radiation [3]. Substitution of Eq. (11) into Eq. (10) yields

$$I_n^{PO} = \sum_{n=1}^{N_{MoM}} \hat{Z}_{POmn} I_n^{MoM} \quad (12)$$

$$\hat{Z}_{POmn} = 2\delta(\vec{r}_{n+N_{MoM}}) \vec{C}(\vec{r}_{n+N_{MoM}}) \cdot \vec{l}_{m+N_{MoM}} \quad m = 1, \dots, N_{PO}, \quad n = 1, \dots, N_{MoM}$$

Direct Solution Method

According to the second figure, the coupled system of equations has the form

$$\hat{Z}_{11} \vec{I}^{\rightarrow MoM} + \hat{Z}_{12} \vec{I}^{\rightarrow PO} = \vec{V} \quad (13)$$

$$\vec{I}^{\rightarrow PO} = \hat{Z}_{PO} \vec{I}^{\rightarrow MoM}$$

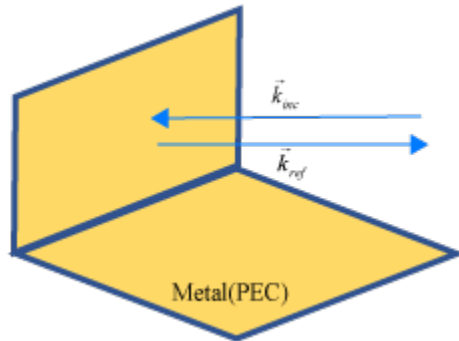
The direct solution method results in the substitution of the expression for the PO current into the first equation,

$$(\hat{Z}_{11} + \hat{Z}_{12} \hat{Z}_{PO}) \vec{I}^{\rightarrow MoM} = \vec{V} \quad (14)$$

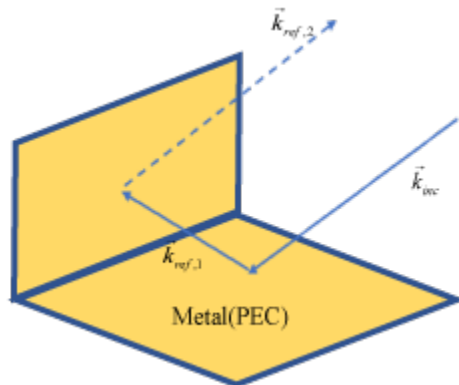
Note The classic physical optics (PO) formulation does not support multiple reflections from a physical structure illuminated by a plane wave. The PO current density is valid only in the illuminated

region of the structure. This formulation does not handle any reflections from the illuminated region that result in secondary illumination of a different region of the structure.

- Case 1: When the direction of the incident plane wave results in a reflection back in the direction of the incoming source.



- Case 2: When the angle of the incident plane wave causes a second reflection from a different part of the structure, this reflection contributes significantly to the scattered field and is not be considered by the PO solver.



References

- [1] U. Jakobus and F. M. Landstorfer, "Improved PO-MM Formulation for Scattering from Three-Dimensional Perfectly Conducting Bodies of Arbitrary Shape," *IEEE Trans. Antennas and Propagation*, vol. AP-43, no. 2, pp. 162-169, Feb. 1995.
- [2] S. M. Rao, D. R. Wilton, and A. W. Glisson, "Electromagnetic scattering by surfaces of arbitrary shape," *IEEE Trans. Antennas and Propagation*, vol. AP-30, no. 3, pp. 409-418, May 1982.
- [3] S. Makarov, *Antenna and EM Modeling in MATLAB*, Wiley, New York, 2002.

Physical Optics Solver

In this section...

“Subdomain RWG Basis Functions and Extra Dimensions” on page 3-20

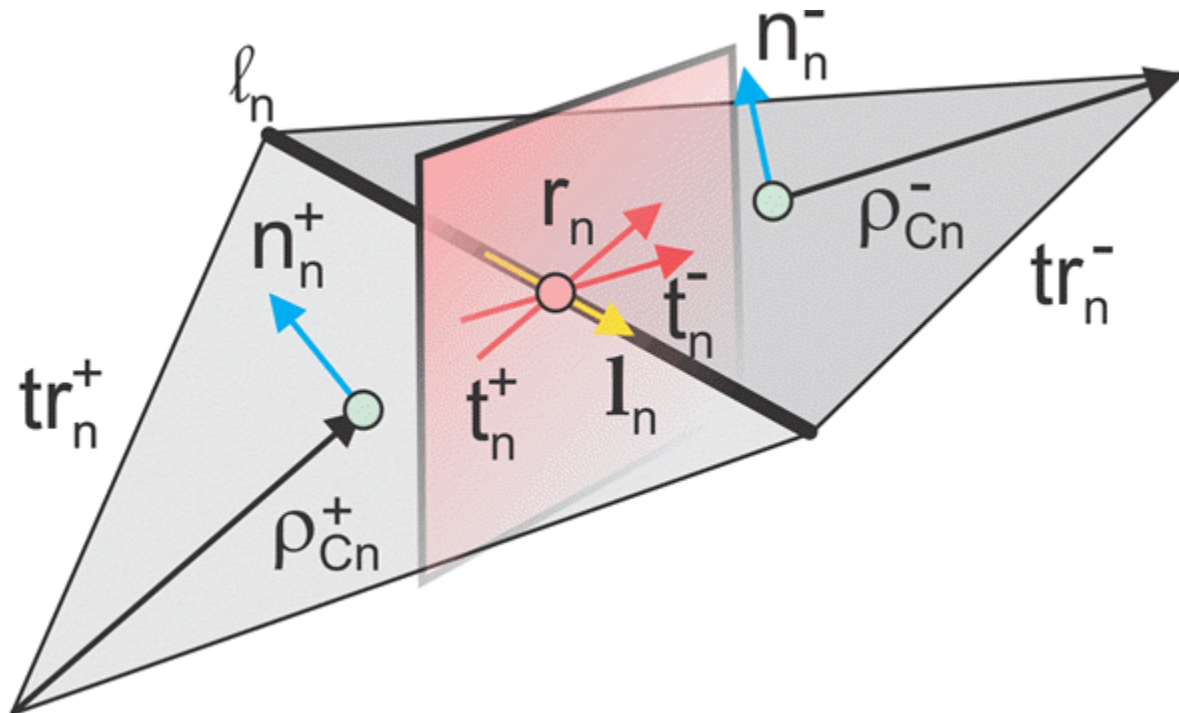
“Finding IPO” on page 3-21

“Determination of Illuminated or Shadow Regions” on page 3-22

Physical optics (PO) solver in Antenna Toolbox allows you to solve for the RCS of an object. In physical optics, the incident field is used to calculate the currents on the surface of the structure in response to the impinging plane wave. With the currents available, you can obtain the scattered field at desired points in the far-field.

Subdomain RWG Basis Functions and Extra Dimensions

The familiar Rao-Wilton-Glisson (RWG) basis functions on triangles are based on [2].



In the image, for two arbitrary triangular patches tr_n^+ and tr_n^- having areas A_n^+ and A_n^- and sharing a common edge l_n the basis functions has the form

$$\vec{f}_n(\vec{r}) = \begin{cases} \frac{l_n}{2A_n^+} \vec{\rho}_n^+ & \vec{r} \text{ in } tr_n^+ \\ \frac{l_n}{2A_n^-} \vec{\rho}_n^- & \vec{r} \text{ in } tr_n^- \end{cases} \quad (1)$$

where $\vec{\rho}_n^+ = \vec{r} - \vec{r}_n^-$ is the vector drawn from the free vertex of the triangle tr_n^+ to the observation point \vec{r} ; $\vec{\rho}_n^- = \vec{r}_n^- - \vec{r}$ is the vector drawn from the observation point to the free vertex of the triangle

tr_n^- . The basis function is zero outside the two adjacent triangles. The RWG vector basis function is linear and has no flux (that is, has no normal component) through its boundary.

From [1], along with the standard definition, this method requires two unit normal vectors \vec{n}_n^\pm and two-unit vectors \vec{t}_n^\pm also shown in the figure. Vector \vec{t}_n^+ is the plane of triangle tr_n^+ ; both vectors are perpendicular to the edge l_n . They are defined at the center of edge, which is l_n denoted by \vec{r}_n .

Directions of \vec{t}_n^\pm

are also shown in the figure. This technique assumes that the normal vectors are properly (angle between adjacent \vec{n}_n^\pm must be less than 180 degrees) and uniquely defined. Specific vector orientation (e.g. outer or inner normal vectors) does not matter. We then form two cross product vectors \vec{l}_n^\pm ,

$$\vec{l}_n^\pm = \vec{t}_n^\pm \times \vec{n}_n^\pm \quad (2)$$

and establish that both such unit vectors directed along the edge are identical,

$$\vec{l}_n^\pm = \vec{l}_n^- = \vec{l}_n^+ \quad (3)$$

Only vector \vec{l}_n^+ is eventually needed.

Finding IPO

The suitable PO approximation has the form:

$$\vec{J}(\vec{r}) = 2\delta(\vec{r})[\vec{n}(\vec{r}) \times \vec{H}(\vec{r})] \quad (5)$$

where δ accounts for the shadowing effects. If the observation point lies in the shadowed region, δ must be zero. Otherwise it equals to ± 1 depending on the direction of incidence with respect to the orientation of the normal vector $\vec{n}(\vec{r})$. Using Eq.(4) yields:

$$\sum_{n=1}^{NPO} I_n^{PO} \vec{f}_n(\vec{r}) = 2\delta(\vec{r})[\vec{n}(\vec{r}) \times \vec{H}(\vec{r})] \quad (6)$$

Ref [3] outlines an elegant way to express unknowns I_n^{PO} explicitly, using an interesting variation of the collocation method. First, consider a collocation point that tends to the edge center \vec{r}_n of a center basis function $\vec{f}_n(\vec{r})$ and is located in the plus triangle. Multiply Eq.(6) by vector \vec{t}_n^+ . Since the normal component of the basis function at the edge is one and all the other basis functions sharing the same triangle have no normal component at the edge, the result becomes

$$I_n^{PO} = 2\delta(\vec{r}_n) \vec{t}_n^+ \cdot [\vec{n}_n^+ \times \vec{H}(\vec{r}_n)] \quad (7a)$$

Repeat the same operation with minus triangle:

$$I_n^{PO} = 2\delta(\vec{r}_n) \vec{t}_n^- \cdot [\vec{n}_n^- \times \vec{H}(\vec{r}_n)] \quad (7b)$$

Add equations 7(a) and 7(b), divide the result by two, and transform the triple vector product to obtain:

$$I_n^{PO} = \frac{2\delta(\vec{r}_n)\vec{H}(\vec{r}_n) \cdot ([\vec{t}_n^+ \times \vec{n}_n^+] + [\vec{t}_n^- \times \vec{n}_n^-])}{2} \quad (8)$$

Therefore, according to equations (2) and (3),

$$I_n^{PO} = 2\delta(\vec{r}_n)\vec{H}(\vec{r}_n) \cdot \vec{l}_n \quad (9)$$

Determination of Illuminated or Shadow Regions

The calculation of $\delta(\vec{r})$ needs to account for the effect of shadowing. For simple convex structures the use of the normal to test against the direction of the radiation would indicate the illuminated or shadow region. If the normal of the triangle is pointing in the opposite direction of the radiation, then the face is illuminated. If the normal of the triangle is in the same direction, then the face is shadowed. But this simple test fails when the object is nonconvex as is the case in more complex structures. To handle this, perform a segment-triangle intersection test to rigorously determine the value of $\delta(\vec{r})$. The value of $\delta(\vec{r})$ is 0 for shadow faces or ± 1 depending on the direction of incidence with respect to the orientation of normal vector. To implement this relative to the RWG basis functions that are formed on the surface of the PO region, check for both arbitrary triangular patches tr_n^+ and tr_n^- to be in the illuminated region and only then consider the contribution made by the edge to the calculation of the PO current. If either triangle is in the shadow region, the delta value is evaluated to zero and therefore the edge does not contribute.

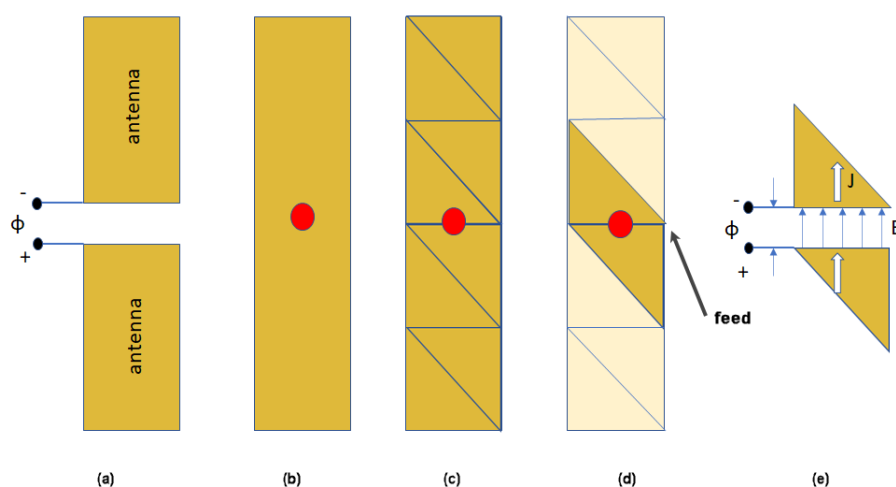
References

- [1] U. Jakobus and F. M. Landstorfer, "Improved PO-MM Formulation for Scattering from Three-Dimensional Perfectly Conducting Bodies of Arbitrary Shape," *IEEE Trans. Antennas and Propagation*, vol. AP-43, no. 2, pp. 162-169, Feb. 1995.
- [2] S. M. Rao, D. R. Wilton, and A. W. Glisson, "Electromagnetic scattering by surfaces of arbitrary shape," *IEEE Trans. Antennas and Propagation*, vol. AP-30, no. 3, pp. 409-418, May 1982.
- [3] S. Makarov, *Antenna and EM Modeling in MATLAB*, Wiley, New York, 2002.

Feed Model

An antenna feed usually uses a conventional transmission line through two electrically close terminals. The feed model and feed location are important variables in antenna analysis because the input impedance at the feed location affects the resulting radiation pattern directivity and gain of the antenna. An antenna can be fed using an open-wire transmission line or a coaxial feed through the ground plane. These feed systems directly impact the antenna impedance characteristics. Antennas in Antenna Toolbox use the delta-gap source feed model.

Delta-Gap Source Feed Model



Consider an antenna with a voltage source ϕ connected between the terminals of the antenna as shown in (a). This approximate strip dipole antenna model is designed in Antenna Toolbox as shown in (b) along with the meshed structure in (c). The two highlighted triangles in (d) correspond to edge elements for the Rao-Wilton-Glisson (RWG) basis function. This feed model is the simplest model suited for RWG edge elements and it is called the delta-function generator or feeding edge model.

Delta-gap source models the feed assuming that the electric field exists only in the infinitesimally small gap between the terminals (Δ). In the case of RWG edge elements, the gap field is described using a delta-function generator or feeding edge model. This model assumes that the excitation voltage at the feed terminals is of a constant value, V_i , and zero elsewhere. The incident electric field is defined as:

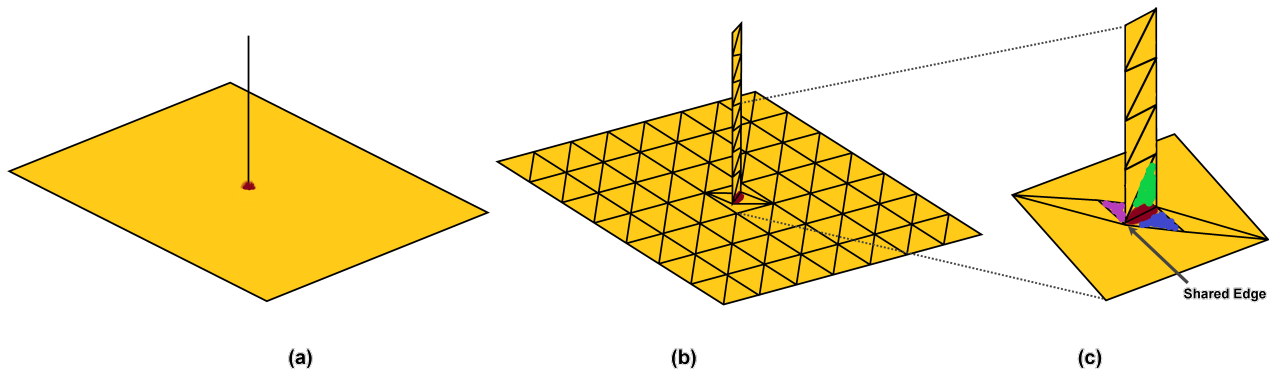
$$E = -\nabla\phi = \frac{V}{\Delta}n_y$$

where:

- ϕ — Electric potential
- V — Voltage across the gap
- Δ — Gap width

This electric field is also constant at the feed terminals and zero elsewhere.

Double-Edge Delta Source Feed Models

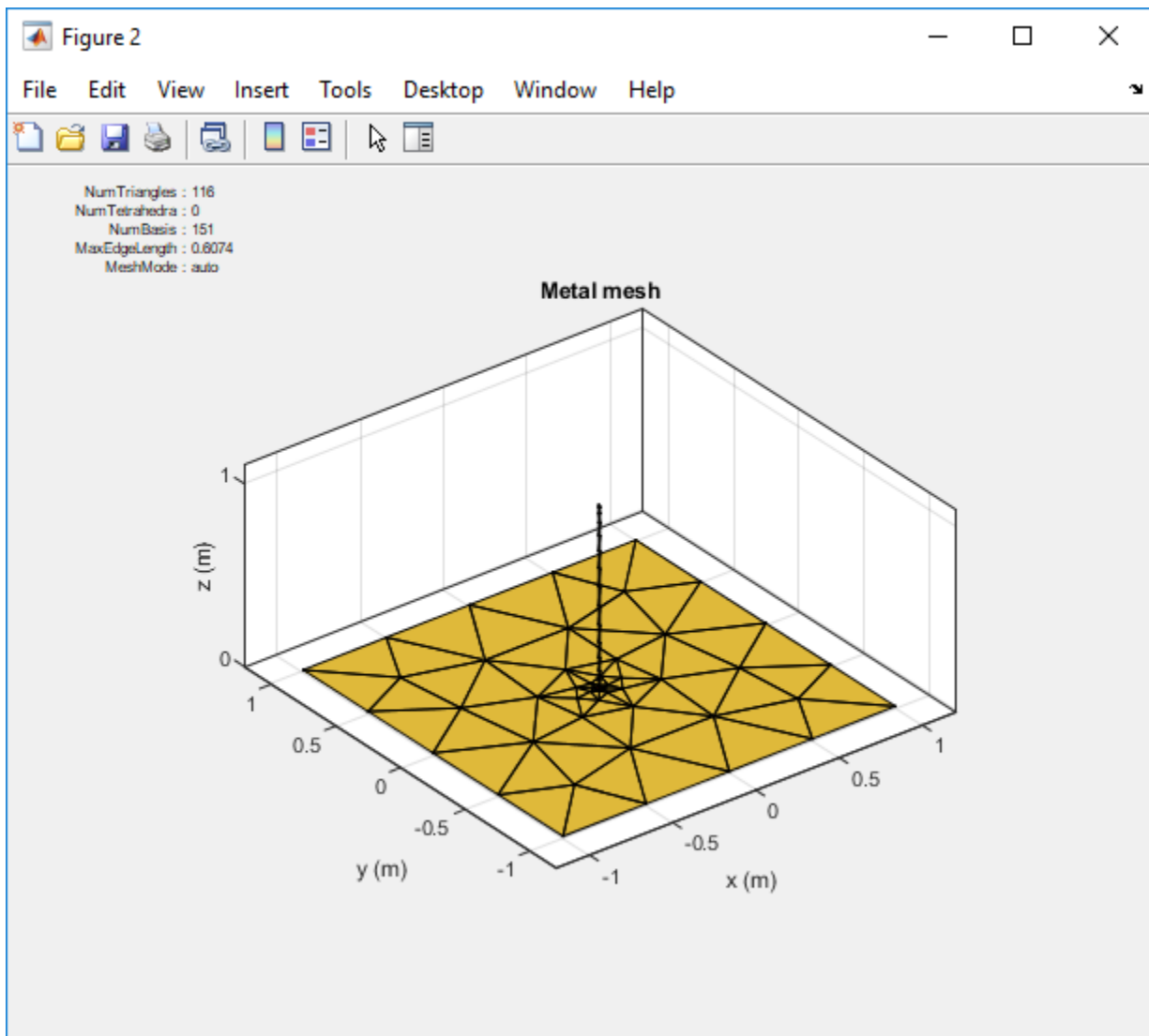


The monopole antenna design in Antenna Toolbox is shown in (a) along with the meshed structure in (b). The edge at the junction in (c) shows three triangles attached to it. The green and blue triangles in figure (c) form one RWG element, and the purple and green triangles form another RWG element. Thus, there are two RWG elements corresponding to single edge. This feed model is called the double-edge delta source feed model.

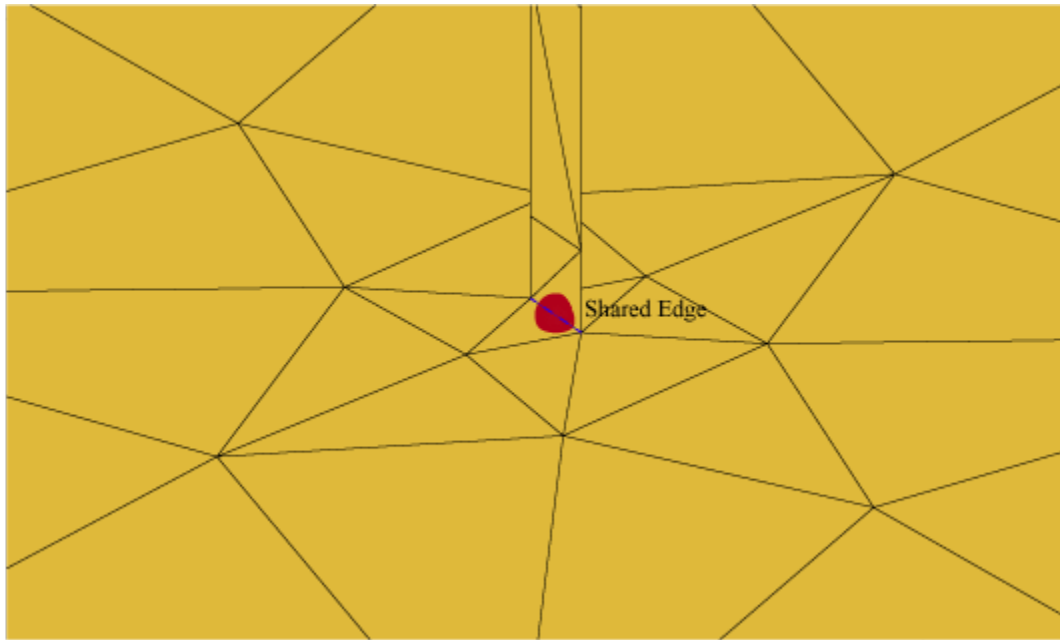
Monopole Antenna

Create a monopole antenna using the `monopole` object and mesh the antenna.

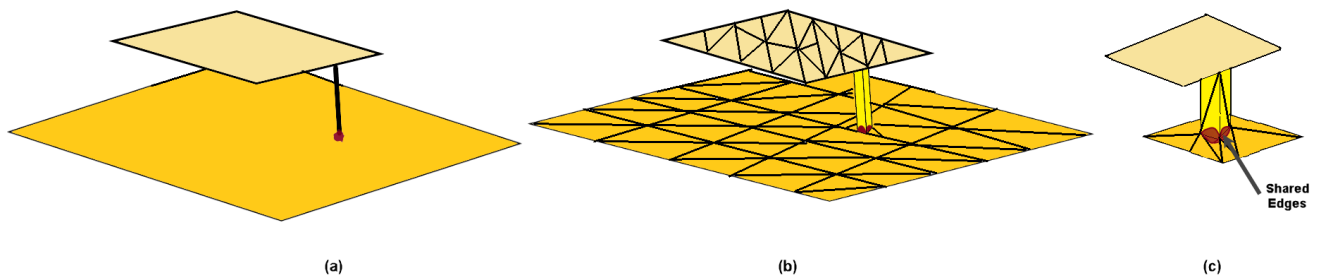
```
ant = monopole;
i = impedance(ant, 75e6)
mesh(ant)
```

Zoom in on the feed region of the antenna. You will see that the feed point shares one common edge between the vertical radiator and the ground plane. In the case of the monopole antenna, this is where the gap exists. Here the voltage is 1 and the phase applied is 0 degrees. Everywhere else on the monopole the voltage is zero.

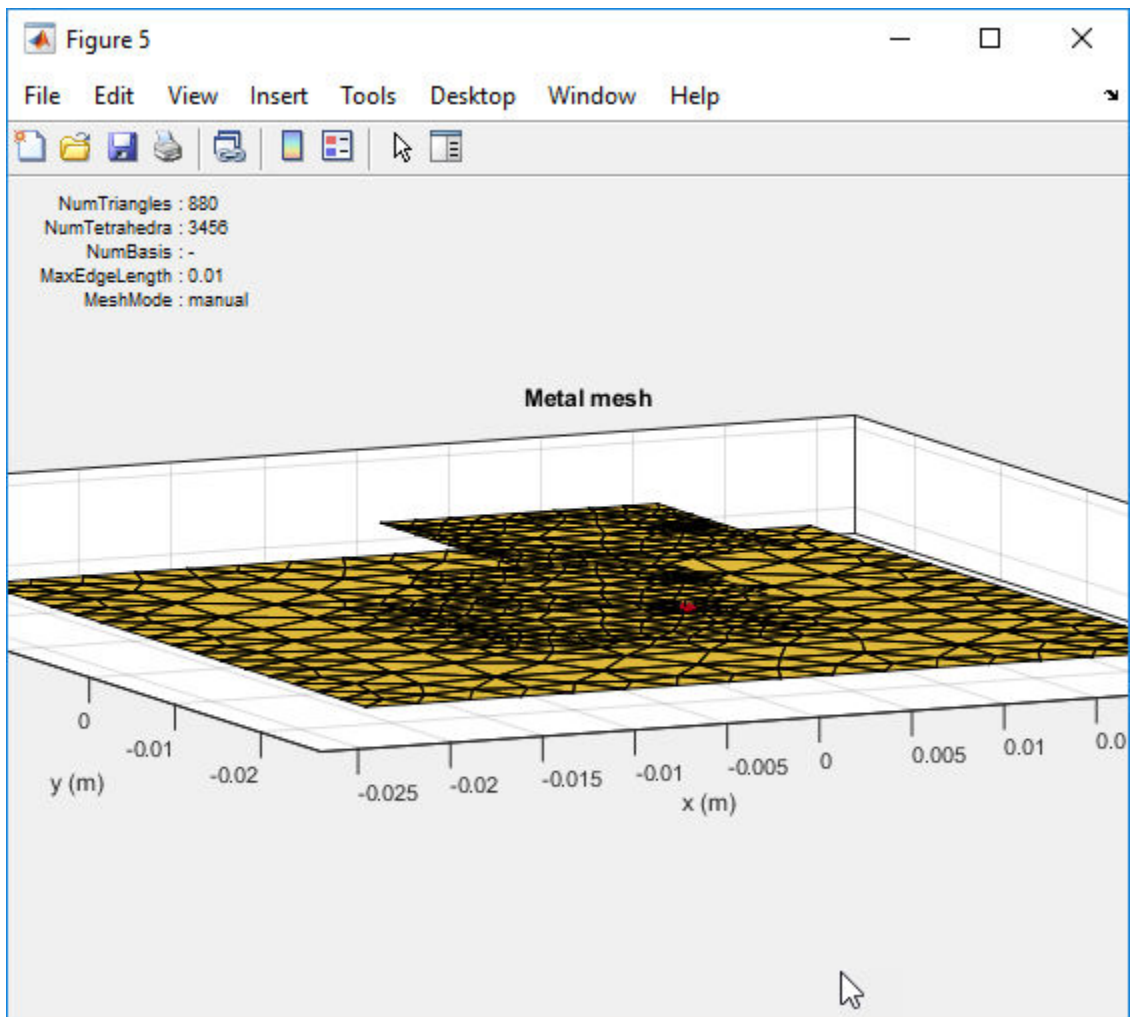


Multi-Edge Delta Source Feed Models



The microstrip patch antenna design in Antenna Toolbox is shown in (a) along with the meshed structure in (b). You can see that the strip connecting the two plates indicates two possible feeding edge positions. This is an example of multi-edge delta source feed model.

You can also improve on the single-edge feed by creating more accurate multi-edge feeds using the `pcbStack` object.

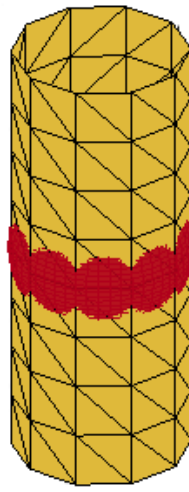


Consider this mesh for a stacked patch antenna with a solid-square feed model. Zoom in on the feed model. You can see that the solid-square feed has four edges connected to the ground plane. This feed-model increases the number of unknowns but represents the feed geometry more accurately.

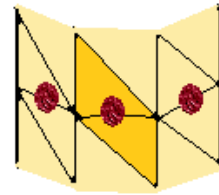
Note To create a stacked patch antenna, see “Modeling and Analysis of Probe-Fed Stacked Patch Antenna” on page 5-395.



(a)



(b)



(c)

The cylindrical dipole antenna design in Antenna Toolbox is shown in (a) along with the meshed structure in (b). You can see the RWG edge elements in (c) around the cylindrical surface forming a closed structure. This is an example of multi-edge delta source feed model.

Antenna Optimization Algorithm

Surrogate model assisted differential evolution for antenna synthesis (SADEA) is an artificial intelligence (AI) driven antenna design method. It is based on machine learning and evolutionary computation techniques, with the advantages of optimization quality, efficiency, generality and robustness. SADEA carries out global optimization and employs a surrogate model built by statistical learning techniques. The method to make surrogate modeling and optimization work harmoniously is critical in such surrogate model-assisted optimization methods. In SADEA, some ideas of the surrogate model-aware evolutionary search framework are borrowed, see [3] and [4].

SADEA uses differential evolution (DE) as the search engine and Gaussian process (GP) machine learning as the surrogate modeling method. For more information, please see [1].

Algorithm Outline

Initialization

Use the Latin Hypercube sampling (LHS) to generate α design samples from $[a, b]^d$, evaluate all the design samples using EM simulations and then use them to form the initial database. $[a, b]^d$ is the search range defined by the user. The value of α is determined self-adaptively.

Iteration steps

- Select the λ best candidate designs from the database to form a population P . Update the best candidate design obtained so far. The value of λ is determined self-adaptively.
- Apply the differential evolution current-to-best/1 mutation and binomial crossover operators on P to generate λ child solutions.
- For each child solution in P , select τ nearest design samples (based on Euclidean distance) as the training data points and construct a local Gaussian process surrogate model. The value of τ is determined self-adaptively.
- Prescreen the λ child solutions generated before by using the Gaussian process surrogate model with the lower confidence bound prescreening.
- Carry out an EM simulation to the prescreened best child solution, add this simulated candidate design and its function value to the database.

Stopping criteria

- The specification(s) is (are) met.
- The standard deviation of the population is smaller than a threshold and the current best objective function value does not improve for a certain number of iterations. (It is better to be controlled using the figure displaying the convergence trend).
- The computing budget (the number of EM simulations) is exhausted. Note that the number of EM simulations can be added anytime.

References

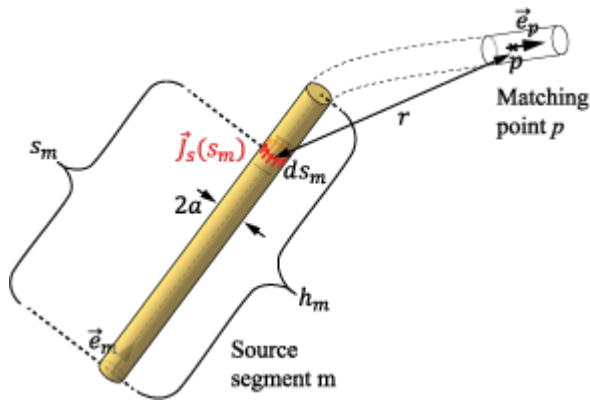
- [1] Liu, Bo, Hadi Aliakbarian, Zhongkun Ma, Guy A. E. Vandenbosch, Georges Gielen, and Peter Excell. "An Efficient Method for Antenna Design Optimization Based on Evolutionary Computation and Machine Learning Techniques." *IEEE Transactions on Antennas and Propagation* 62, no. 1 (January 2014): 7–18. <https://doi.org/10.1109/TAP.2013.2283605>.

- [2] Liu, Bo, Alexander Irvine, Mobayode O. Akinsolu, Omer Arabi, Vic Grout, and Nazar Ali. "GUI Design Exploration Software for Microwave Antennas." *Journal of Computational Design and Engineering* 4, no. 4 (October 2017): 274-81. <https://doi.org/10.1016/j.jcde.2017.04.001>.
- [3] Liu, Bo, Qingfu Zhang, and Georges G. E. Gielen. "A Gaussian Process Surrogate Model Assisted Evolutionary Algorithm for Medium Scale Expensive Optimization Problems." *IEEE Transactions on Evolutionary Computation* 18, no. 2 (April 2014): 180-92. <https://doi.org/10.1109/TEVC.2013.2248012>.
- [4] Liu, Bo, Qingfu Zhang, Georges G. E. Gielen, A.Karkar, A.Yakovlev, V.Grout. "SMAS: A Generalized and Efficient Framework for Computationally Expensive Electronic Design Optimization Problems." *Computational Intelligence in Electronic Design*, Springer, 2015.
- [5] Grout, Vic, Mobayode O. Akinsolu, Bo Liu, Pavlos I. Lazaridis, Keyur K. Mistry, and Zaharias D. Zaharis. "Software Solutions for Antenna Design Exploration: A Comparison of Packages, Tools, Techniques, and Algorithms for Various Design Challenges." *IEEE Antennas and Propagation Magazine* 61, no. 3 (June 2019): 48-59. <https://doi.org/10.1109/MAP.2019.2907887>.

See Also

"Optimization of Antenna Array Elements Using Antenna Array Designer App" on page 5-636 |
"Maximizing Gain and Improving Impedance Bandwidth of E-Patch Antenna" on page 5-625

Wire Solver



The following is an abbreviated description of the derivation of the equations solved for by the wire solver. For more detailed information, see [1].

For PEC (Perfect electrically conducting) wires, the boundary conditions require the tangential component of the total electric field on the wire surface to be zero:

$$(E + E_i)_{\text{tan}} = 0 \quad (1)$$

You can also express the equation in terms of the surface current density, \vec{J}_s , on the PEC wire and its spatial derivative. This is done by integrating the effect of the surface current density using a Green's function, $g(|\vec{r} - \vec{r}'|)$. The Green's function abstractly represents the effect of an infinitesimal surface-current density source located at \vec{r}' on the electric and magnetic fields at an observation point \vec{r} . The Green's function for free space propagation is:

$$g(|\vec{r} - \vec{r}'|) = \frac{\exp(-jk|\vec{r} - \vec{r}'|)}{4\pi|\vec{r} - \vec{r}'|} \quad (2)$$

where $k = \omega\sqrt{\epsilon_0\mu_0}$ is the wave number and ω is the angular frequency. In general, the following steps are needed to solve the electromagnetic (EM) problem and obtain the EM fields in the medium. To represent the EM problem accurately, both longitudinal and transverse components of integrated current are taken into consideration on the wire surface and require that Eq. (1) holds over the entire wire surface. Discretize Eq.(1) over a finite set of points or basis functions to obtain a set of equations. Discretize the surface current-density into a finite set of basis functions to obtain a set of unknowns. Solve the set of equations of unknowns to yield a discrete approximation to the surface currents-densities on the wire. These current densities enables calculation of the electromagnetic fields at any desired point. For thin wires the above steps are exhaustive and, simplify the Green's function to :

$$g(r_a) = \frac{\exp(-jkr_a)}{4\pi r_a} \quad (3)$$

also referred to as the thin-wire kernel approximation, where $r_a = \sqrt{r^2 + a^2}$ is an approximate average of the distance $|\vec{r} - \vec{r}'|$, r is the distance between the infinitesimal surface current-density source and

the wire axis, and a is the wire radius. In addition, the current density on the wire is replaced by the wire current $I = 2\pi a \left| \vec{J}_s \right|$. Using the above approximation, write Eq. (1) as:

$$\left(-j\omega\mu_0 \int_s \left[(Is)\vec{e}(s)g(r_a) + \frac{1}{k^2} \frac{dI(s)}{ds} \vec{\nabla} g(r_a) \right] ds + \vec{E}_i \right)_{\tan} = 0 \quad (4)$$

where s is the longitudinal location along the wire and $\vec{e}(s)$ is the unit vector representing the wire orientation at that location. Discretize the wire into a finite number of segments $s(m)$ where $m = 1 \dots N$, and then further discretize Eq.(4) into a finite set of equations imposed at points matching $p = 1 \dots N$ to obtain:

$$\sum_{m=1}^N \int_0^{h_m} \left[\vec{e}_p \vec{e}_m I_m(s_m) + \frac{\vec{e}_p}{k^2} \frac{dI_m(s_m)}{ds_m} \nabla \right] g(r_a) ds_m = \frac{\vec{e}_p \vec{E}_i}{j\omega\mu_0} \quad (5)$$

where $I_m(s_m)$ is the current distribution along the segment m approximated by the polynomial of the form:

$$I_m(s_m) = \sum_{i=0}^{n_m} I_{m,i} \left(\frac{s_m - \frac{h_m}{2}}{h_m} \right)^i, \quad 0 \leq s_m \leq h_m, \quad m = 1, \dots, N \quad (6)$$

with h_m being the length of the segment and n_m the chosen degree of the polynomial (which is different from segment to segment). Substituting Eq. (6) into (5), the set of equations can be written in a matrix format as:

$$[Z_{p,mi}][I_{mi}] = [V_{ex,p}] \quad (7)$$

where $[Z_{p,mi}]$ is the impedance matrix, $[I_{mi}]$ is the vector of unknown coefficients representing the current on the wire, and $[V_{ex,p}]$ is the excitation vector. More details about the calculation of the elements of $[Z_{p,mi}]$ are given in [1].

References

- [1] Popovic,B.C, M.B. Dragovic and A.R. Djordjevic. *Analysis and Synthesis of Wire Antennas* Research Studies Press, 1982

Fast Multipole Method for Large Structures

The fast multipole method (FMM) computational technique in Antenna Toolbox allows you to model and analyze antennas and arrays on large platforms like aircraft and automobiles.

Direct Solvers

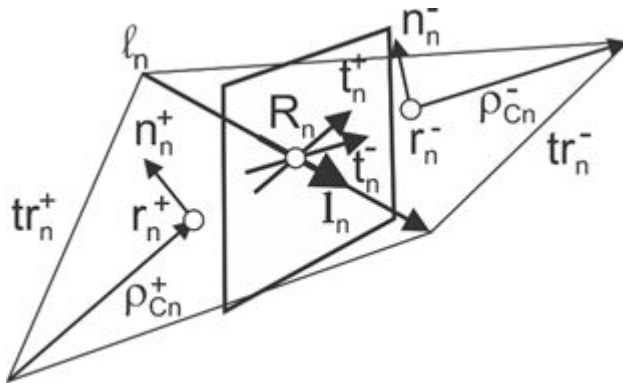
The first step in the computational solution of electromagnetic problems is to discretize Maxwell's equations. The process results in this matrix-vector system:

$$V = ZI$$

- V – Applied voltage vector. This signal can be voltage or power applied to the antenna or an incident signal falling on the antenna.
- I – Current vector that represents current on the antenna surface.
- Z – Interaction matrix or impedance matrix that relates V to I .

Antenna Toolbox uses “Method of Moments Solver for Metal and Dielectric Structures” on page 3-9 to calculate the interaction matrix and solve system equations.

To calculate the surface currents on the antenna structure, you first define Rao-Wilton-Glisson (RWG) basis functions. A RWG basis function is a pair of triangles that share an edge, and is shown in the figure.



For any two triangle patches, t_n^+ and t_n^- , having areas A_n^+ and A_n^- , and sharing common edge l_n , the basis function is

$$\vec{f}_n(\vec{r}) = \begin{cases} \frac{l_n}{2A_n^+} \vec{\rho}_n^{+S}, & \vec{r} \in t_n^+ \\ \frac{l_n}{2A_n^-} \vec{\rho}_n^{-S}, & \vec{r} \in t_n^- \end{cases}$$

- $\vec{\rho}_n^+ = \vec{r} - \vec{r}_n^+$ – Vector drawn from the free vertex of triangle t_n^+ to observation point \vec{r}
- $\vec{\rho}_n^- = \vec{r}_n^- - \vec{r}$ – Vector drawn from the observation point to the free vertex of the triangle t_n^-

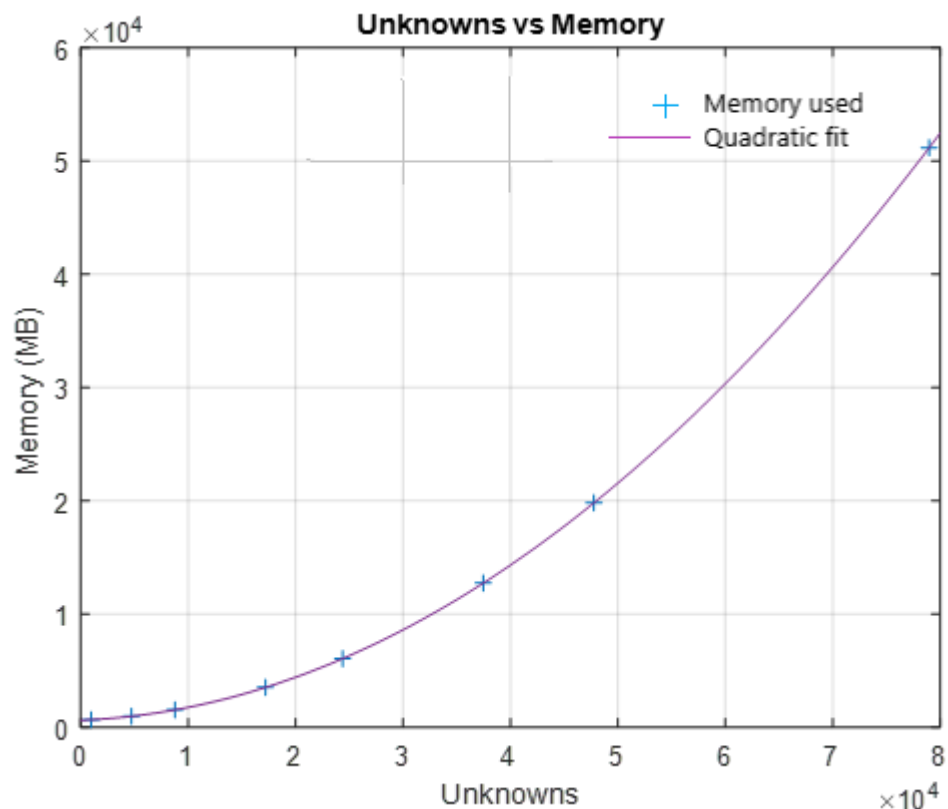
and

$$\nabla \cdot \vec{f}_n(\vec{r}) = \begin{cases} \frac{l_n}{A_n^+}, & \vec{r} \in t_n^+ \\ -\frac{l_n}{A_n^-}, & \vec{r} \in t_n^- \end{cases}$$

The basis function is zero outside the two adjacent triangles t_n^+ and t_n^- . The RWG vector basis function is linear and has no flux (no normal component) through its boundary.

Relation Between Memory Used and Problem Size

The interaction matrix Z is a complex dense symmetric matrix. It is a square N -by- N matrix, where N is the number of basis functions, that is, the number of interior edges in the structure. Consider the scenario of a large structure like an aircraft or a ship. Typical narrow-band antennas like the dipole or patch are half-wavelength in size, but ships or aircraft can often be at least 100 wavelengths or more in size. To solve for the electromagnetic effects of either radiation or scattering from this structure using a full-wave solver, the first step is to mesh the structure and then form the basis functions. Doing so generates more than 50,000 triangles. Since the memory requirement for the direct solver is of the order of $O(N^2)$, in basis function space, the growth is as shown in this plot.



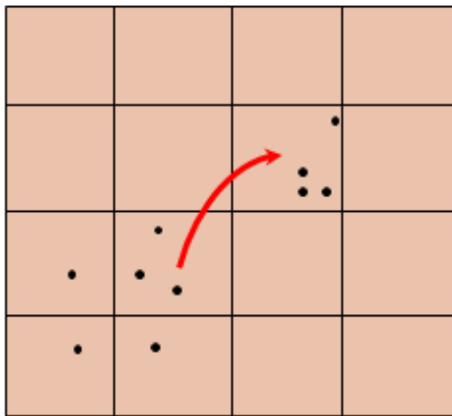
Under any of the following conditions the number of the unknowns become very large:

- High analysis frequency
- Structure refined with a finer mesh

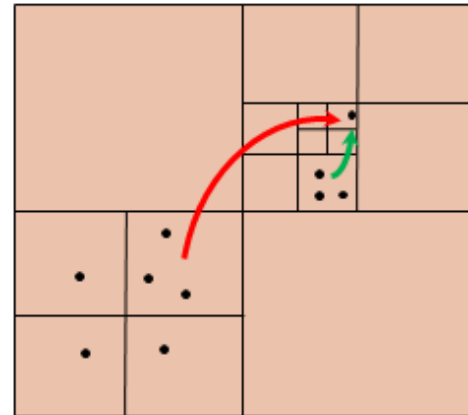
- Analysis of a physically large structure

Fast Multipole Method (FMM)

The acceleration achieved by the FMM algorithm is due to its ability to subdivide the problem into successively smaller spatial regions, thereby ensuring that a given pair of source and target clusters are distant enough for the interaction to be computed using multipole expansions. The following figure illustrates that.



Single-layer FMM



Multi-layer FMM

This approach fits well with the need to accelerate the computation of interactions between separated pairs of basis functions, that is, source and target dipole pairs. The problem of determining the electromagnetic potential at a given set of target points in a Helmholtz type of problem can be expressed as:

$$u(\mathbf{r}) = \sum_{n=1}^N c_n \frac{\exp(jk|\mathbf{r} - \mathbf{r}_n|)}{|\mathbf{r} - \mathbf{r}_n|} - \mathbf{v}_n \cdot \nabla \left(\frac{\exp(jk|\mathbf{r} - \mathbf{r}_n|)}{|\mathbf{r} - \mathbf{r}_n|} \right)$$

wherein, c_n and v_n represent the collection of charge and dipole strengths, respectively, k is the wavenumber, and $u(\mathbf{r})$ is the potential computed by FMM in 3-D space.

FMM speeds up the computation of the matrix-vector product by substantially accelerating the computation of point-to-point interactions mediated by the Green's function. The original current and charge distributions on the surface of the target are determined by introducing these coefficients back into the basis function expansion. Scattered or radiated field of the target including its radar cross-sections is then found by computing the radiation of the known surface currents and charges at required points in space. The iterative approach to determining a matrix inverse is a well-studied and established field of applied linear algebra. Among the variety of iterative solvers that exist, the generalized minimum residual (GMRES) method is a well-known technique. Antenna Toolbox uses this iterative solver.

Electric Field Integral Equation (EFIE)

The direct solver implemented in the Antenna Toolbox is based on EFIE. EFIE uses the electric field relationships on the surface of a metal and at any point in free space to set up the system of equations.

$$E_t^s = -E_t^i$$

$$E^s(r) = -j\omega A - \nabla\varphi$$

The index t in the first of the two equations is used to describe the tangential component of the electric field on a metal surface, index s describes the scattered field, and index i denotes the incident field. In the second equation the relationship of the scattered field is shown in terms of the electric scalar potential φ and magnetic vector potential A .

Applying the Galerkin approach, where the test using the basis functions leads to the following key equation:

$$j\omega \left\{ \frac{l_m}{2} p_m^+(r_m^+) \cdot A(r_m^+) + \frac{l_m}{2} p_m^-(r_m^-) \cdot A(r_m^-) \right\} - \{l_m \varphi(r_m^+) - l_m \varphi(r_m^-)\} = V_m$$

$$V_m = \frac{l_m}{2} p_m^+(r_m^+) \cdot E^i(r_m^+) + \frac{l_m}{2} p_m^-(r_m^-) \cdot E^i(r_m^-)$$

Magnetic Field Integral Equation (MFIE)

MFIE equation expresses the surface current density $J(\mathbf{r})$ developed on the body of a metallic object in response to a magnetic field excitation. An important observation here is that the second term of MFIE is the exact physical optics (PO) approximation. This equation captures the first order solution as the PO approximation, while the second term involving the integral captures the full-wave effects, thus providing a complete solution.

MFIE can be applied only to closed structures such as boxes, spheres, closed shells of aircraft, and so on. It cannot be applied, for example, to a strip dipole or monopole antenna.

$$J(\mathbf{r}) = 2\mathbf{n}(\mathbf{r}) \times \int_s J(\mathbf{r}') \times \nabla_{r'} \frac{\exp(-jk|\mathbf{r} - \mathbf{r}'|)}{4\pi|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}' + 2\mathbf{n}(\mathbf{r}) \times H^i(\mathbf{r})$$

Using the collocation approach leads to the equation for the MFIE implementation:

$$c_m - \left\{ I_m \cdot \sum_{n=1}^{N_{facets}} \begin{pmatrix} M_1 \cdot \nabla \\ M_2 \cdot \nabla \\ M_3 \cdot \nabla \end{pmatrix} \frac{\exp(-jk|R_m - r_n|)}{4\pi|R_m - r_n|} \right\} = I_m^{PO}$$

$$M_1 = \begin{pmatrix} 0 \\ -m_z \\ +m_y \end{pmatrix}, M_2 = \begin{pmatrix} +m_z \\ 0 \\ -m_x \end{pmatrix}, M_3 = \begin{pmatrix} -m_y \\ +m_x \\ 0 \end{pmatrix}, m = I_n r_n$$

Combined Field Integral Equation (CFIE)

CFIE uses the two equations shown for EFIE and MFIE. The term α is chosen to be 0.5 and $\eta = 376.3\Omega$ is the free space impedance.

$$\alpha LHSE_m + (1 - \alpha)\eta LSH_m = \alpha V_m + (1 - \alpha)\eta I_m^{PO}$$

The FMM solver is applied to compute the left side of this equation. $LHSE_m$ represents the left side of EFIE and LSH_m represents the left side of MFIE.

References

- [1] *Flatironinstitute/FMM3D*. Fortran. 2018. Reprint, Flatiron Institute, 2021. <https://github.com/flatironinstitute/FMM3D>.
- [2] Greengard, L, and V Rokhlin. "A Fast Algorithm for Particle Simulations." *Journal of Computational Physics* 73, no. 2 (December 1987): 325-48. [https://doi.org/10.1016/0021-9991\(87\)90140-9](https://doi.org/10.1016/0021-9991(87)90140-9).
- [3] Rius JM, Úbeda E, Parrón J. On The Testing of the Magnetic Field Integral Equation With RWG Basis Functions in Method of Moments. *IEEE Trans. Antennas and Propagation*, vol. AP-49, no. 11, pp. 1550-1553.
- [4] Rao SM, Wilton DR, Glisson AW. Electromagnetic Scattering by Surfaces of Arbitrary Shape. *IEEE Trans. on Antennas and Propagation*. 1982 May;30(3):409-418. doi: 001 8-926X/82/0500-0409.

Finite Conductivity and Thickness Effect in MoM Solver

Antenna Toolbox uses the method of moments (MoM) computation technique for finite metal dielectric antennas. The MoM technique provides a computational solution to the electromagnetic problems by discretizing Maxwell's equations. The process results in this matrix-vector system:

$$V = ZI$$

where,

- V — Applied voltage vector. This signal can be voltage or power applied to the antenna or an incident signal falling on the antenna.
- I — Current vector that represents current on the antenna surface.
- Z — Interaction matrix or impedance matrix that relates V to I . For more information on the interaction matrix, see “Method of Moments Solver for Metal and Dielectric Structures” on page 3-9.

In perfect electrically conducting (PEC) metals, the electrical conductivity is infinite, and the thickness is assumed to be infinitely thin. Therefore, the skin effect is not considered in PEC metal antenna. However, in most practical use metal antennas, the metal sheet has finite conductivity and finite thickness. This causes the formation of a finite skin depth effect that is commonly characterized by an equivalent surface impedance [1].

Antenna Toolbox uses the MoM [2] technique to solve finite metal antennas where the finite conductivity and thickness of the metal is considered through an equivalent surface. Thus, compared to PEC metal, there is no change in computational time and meshing in non-PEC metal antennas. The underlying electromagnetic theory of modelling finite conductor is explained below.

Consider a metal sheet with finite dimensions in free space as shown in Figure 1.

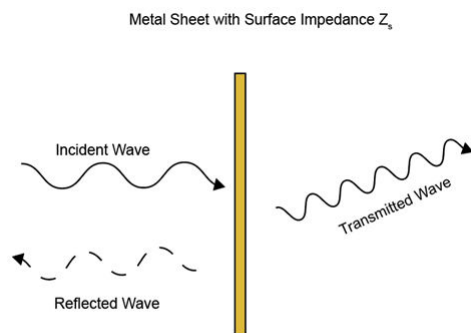


Figure 1. Finite metal sheet in free space

Using transmission line modelling, its equivalent circuit representation is modeled with equivalent surface impedance Z_s . This is shown in Figure 2.

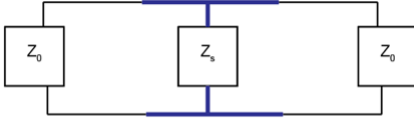


Figure 2. Equivalent transmission line representation

Finite conductivity poses a hindrance to the equivalent surface current flow on the antenna surface. Following the field continuity relation on the conductor surface, model the finite conductivity using an equivalent surface impedance. The equivalent surface impedance Z_s can be written using [1] as

$$Z_s = \frac{k_c}{\sigma_c} \left(\frac{\exp(k_c t) + \frac{\sigma_c Z_0 - k_c}{\sigma_c Z_0 + k_c} \exp(-k_c t)}{\exp(k_c t) - \frac{\sigma_c Z_0 - k_c}{\sigma_c Z_0 + k_c} \exp(-k_c t)} \right)$$

where,

t - thickness of metal

σ_c - conductivity of metal

$Z_0 = \sqrt{\frac{\mu}{\epsilon}}$, free space intrinsic impedance

$$k_c = \frac{1+j}{\delta_c}$$

$$\delta_c = \sqrt{\frac{2}{\omega \mu \sigma_c}} = \sqrt{\frac{1}{\pi f \mu \sigma_c}}, \text{ skin depth}$$

The parameter skin depth, physically signifies how much the surface current can penetrate the conducting surface. For PEC metals, the skin depth is zero as the conductivity is infinite. Thus, the entire current distribution lies over the surface layer in PEC metals. For non-PEC metals, if the skin depth value is very low, $\exp(k_c t)$ tends to be very high. In most practical use cases, the conductivity is very high and therefore, $Z_0 \gg k_c/\sigma_c$. In such case Z_s becomes:

$$Z_s = \frac{k_c}{\sigma_c} \left(\frac{1 + \exp(k_c t)}{1 - \exp(k_c t)} \right)$$

If the thickness is high, $\exp(-2k_c t)$ tends to be negligible and Z_s becomes:

$$Z_s = \frac{k_c}{\sigma_c} = (1 + j) \sqrt{\frac{\omega \mu}{2 \sigma_c}}$$

Thus, for metals with high conductivity and thickness values, the surface impedance becomes independent of thickness.

MoM Formulation

In finite metal-dielectric structures, the elements of the metal-to-metal interaction matrix are computed as given in [2] and [3].

$$Z_{mn}^{MM} = \left(\frac{j\omega\mu}{4\pi} \right) \int_S \int_S \vec{f}_m^M(\vec{r}) \cdot \vec{f}_n^M(\vec{r}') g d\vec{r}' d\vec{r} - \frac{j}{4\pi\omega\epsilon} \int_S \int_S \nabla \vec{f}_m^M(\vec{r}) \cdot \nabla \vec{f}_n^M(\vec{r}') g d\vec{r}' d\vec{r} + Z_s \int_S \vec{f}_m^M(\vec{r}) \cdot \vec{f}_n^M(\vec{r}') d\vec{r}$$

The first two terms in the equation consider the electric scalar potential and the magnetic vector potential of the electromagnetic fields. The third term considers the finite surface impedance effect due to the nonzero skin depth of the finite metal antennas. In PEC antennas, $Z_s = 0$; thus, no finite conduction loss occurs in PEC metals. The Rao-Wilton-Glisson (RWG) [4] basis functions in MoM equation are used. For more information, see “Method of Moments Solver for Metal and Dielectric Structures” on page 3-9.

References

- [1] A. Kerr. “Surface impedance of superconductors and normal conductors in EM simulators”. MMA Memo, vol. 21, no. 245, pp. 1-17, 1999.
- [2] Sarkar, T.K., and E. Arvas. “An Integral Equation Approach to the Analysis of Finite Microstrip Antennas: Volume/Surface Formulation.” *IEEE Transactions on Antennas and Propagation* 38, no. 3 (March 1990): 305-12. <https://doi.org/10.1109/8.52238>.
- [3] Rengarajan, Sembiam Rajagopal, and Richard E. Hodges. “Accurate Evaluation of the Conductor Loss in Rectangular Microstrip Patch Reflectarrays”. *Progress In Electromagnetics Research M* 75 (2018): 159-66. <https://doi.org/10.2528/PIERM18072606>.
- [4] Rao, S., D. Wilton, and A. Glisson. “Electromagnetic Scattering by Surfaces of Arbitrary Shape.” *IEEE Transactions on Antennas and Propagation* 30, no. 3 (May 1982): 409-18. <https://doi.org/10.1109/TAP.1982.1142818>.

See Also

“Method of Moments Solver for Metal Structures” on page 3-2 | “Method of Moments Solver for Metal and Dielectric Structures” on page 3-9

Solvers

Antenna Toolbox provides six solvers, each with a specific application.

Solver Types	Materials	Applications
Method of Moments (MoM) solver using free-space Green's function	Metals and dielectrics	Antennas and arrays
MoM solver using periodic Green's function	Metals and dielectrics	Infinite arrays
Physical optics (PO) solver	Metals	Radar cross sections
Hybrid MoM-PO solver	Metals	Electrically large antennas, installed antennas. For more information, see "Solver Recommendations for Installed Antenna Analysis".
Fast Multipole Method (FMM) solver	Metals	Electrically large antennas, installed antennas. For more information, see "Solver Recommendations for Installed Antenna Analysis".
MoM solver using wire basis	Metals	Wire antennas

For more details on each of these solvers, see:

- "Method of Moments Solver for Metal Structures" on page 3-2
- "Method of Moments Solver for Metal and Dielectric Structures" on page 3-9
- "Hybrid MoM-PO Method for Metal Antennas with Large Scatterers" on page 3-15
- "Physical Optics Solver" on page 3-20
- "Wire Solver" on page 3-31
- "Fast Multipole Method for Large Structures" on page 3-33
- "Finite Conductivity and Thickness Effect in MoM Solver" on page 3-38

RF Propagation

- “Troubleshooting Site Viewer” on page 4-2
- “Access Basemaps and Terrain in Site Viewer” on page 4-3
- “Access TIREM Software” on page 4-5
- “Choose a Propagation Model” on page 4-6
- “Ray Tracing for Wireless Communications” on page 4-12

Troubleshooting Site Viewer

In this section...
“Internet Connection Failure” on page 4-2
“Graphics Environment” on page 4-2

Internet Connection Failure

When you create a Site Viewer, a check is made to make sure that you have an internet connection to retrieve the default basemap and terrain data.

If Site Viewer cannot connect to the Internet the following warning messages are displayed:

- *Warning: Unable to access the Internet, showing Dark Water instead of Satellites. See Access Basemaps and Terrain in Site Viewer.*
- *Warning: Unable to access terrain data. See Access Basemaps and Terrain in Site Viewer.*

If Site Viewer cannot connect to the Internet, then terrain data is not used and the Dark Water basemap is selected.

Graphics Environment

When JavaScript® for WebGL™ support fails, Site Viewer issues an error message in the Command Line, notifying you to update the graphics hardware driver. For more information, see “Resolving Low-Level Graphics Issues”.

See Also

Functions

renderinfo

Objects

siteviewer

More About

- “Access Basemaps and Terrain in Site Viewer” on page 4-3
- “System Requirements for Graphics”
- “Resolving Low-Level Graphics Issues”

Access Basemaps and Terrain in Site Viewer

In this section...

"Use Installed Basemap" on page 4-3

"Download Basemaps" on page 4-3

"Add Custom Basemaps" on page 4-3

"Access Terrain" on page 4-4

Site Viewer displays data over basemaps and terrain. You can access different basemap and terrain choices in different ways.

MathWorks® offers a selection of basemaps, including two-tone maps created using Natural Earth, high-zoom-level maps hosted by Esri®, and a street map from OpenStreetMap®. For more information about basemap options, see the Basemap property of the siteviewer object.

Use Installed Basemap

The "darkwater" basemap is installed with MATLAB®. The other basemaps are not installed with MATLAB, but you can access them over an internet connection.

Download Basemaps

To work offline or to improve map responsiveness, you can download the basemaps created using Natural Earth onto your local system. The other basemaps are not available for download.

Download basemaps using the Add-On Explorer.

- 1 On the MATLAB **Home** tab, in the **Environment** section, click **Add-Ons > Get Add-Ons**.
- 2 In the Add-On Explorer, scroll to the **MathWorks Optional Features** section, and click **Show All** to find the basemap add-ons. You can also search for the basemap add-ons by name (listed in the following table) or click **Optional Features** in **Filter by Type**.
- 3 Select the basemap add-ons that you want to download.

Basemap Name	Basemap Data Package Name
"bluegreen"	MATLAB Basemap Data - bluegreen
"grayland"	MATLAB Basemap Data - grayland
"colorterrain"	MATLAB Basemap Data - colorterrain
"grayterrain"	MATLAB Basemap Data - grayterrain
"landcover"	MATLAB Basemap Data - landcover

Add Custom Basemaps

Add custom basemaps from a URL by using the addCustomBasemap function. MATLAB requires an active internet connection to add and use custom basemaps from a URL.

If you have Mapping Toolbox™, you can also create custom basemaps from MBTiles files. MATLAB does not require internet access to add custom basemaps from MBTiles files. See `addCustomBasemap` for more information.

Note If the basemap does not render correctly in Site Viewer (for example only the ocean is visible), check if the basemap server supports CORS (cross-origin resource sharing). Site Viewer does not support basemaps that do not support CORS.

Access Terrain

By default, Site Viewer uses terrain data hosted by MathWorks and derived from the GMTED2010 model by the USGS and NGA. You need an active internet connection to access this terrain data, and you cannot download it.

To work offline or to improve terrain responsiveness, add custom terrain from DTED files using the `addCustomTerrain` function. You do not need an active internet connection to add or use custom terrain.

Alternatively, you can set the `Terrain` property of the Site Viewer to "none".

See Also

Objects

`siteviewer`

More About

- "Troubleshooting Site Viewer" on page 4-2
- "Use Basemaps in Offline Environments" (Mapping Toolbox)
- "System Requirements for Graphics"
- "Resolving Low-Level Graphics Issues"

Access TIREM Software

The Terrain Integrated Rough Earth Model™ (TIREM™) is a propagation model for computing the path loss for irregular terrain and seawater scenarios. TIREM is developed, trademarked, and licensed by Huntington Ingalls Industries. To use TIREM, you need to acquire it from Huntington Ingalls Industries.

TIREM is designed to calculate the reference basic median propagation loss (path loss) based on the terrain profile along the great circle path between two antennas, for example, using digital terrain elevation data (DTED). You can use TIREM model to calculate the point-to-point path loss between sites over irregular terrain. The model combines physics with empirical data to provide path loss estimates. The TIREM propagation model can predict path loss at frequencies between 1 MHz and 1 THz.

Use `tiremSetup` to enable TIREM access from within MATLAB. The TIREM library folder contains the `tirem3` shared library. The full library name is platform-dependent:

Platform	Shared Library Name
Windows	<code>libtirem3.dll</code> or <code>tirem3.dll</code>
Linux	<code>libtirem3.so</code>
Mac	<code>libtirem3.dylib</code>

See Also

Functions

`tiremSetup`

Objects

TIREM

Related Examples

- “Planning Radar Network Coverage over Terrain” on page 5-521

Choose a Propagation Model

Propagation models allow you to predict the propagation and attenuation of radio signals as the signals travel through the environment. You can simulate different models by using the `propagationModel` function. Additionally, you can determine the range and path loss of radio signals in these simulated models by using the `range` and `pathloss` functions.

The following sections describe various propagation and ray tracing models. The tables in each section list the models that are supported by the `propagationModel` function and compare, for each model, the supported frequency ranges, model combinations, and limitations.

Atmospheric

Atmospheric propagation models predict path loss between sites as a function of distance. These models assume line-of-sight (LOS) conditions and disregard the curvature of the Earth, terrain, and other obstacles.

Model	Description	Frequency	Combinations	Limitations
freespace (FreeSpace)	Ideal propagation model with clear line of sight between transmitter and receiver	No enforced range	Can be combined with rain, fog, and gas	Assumes line of sight
rain (Rain)	Propagation of a radio wave signal and its path loss in rain. For more information, see [3].	1 GHz to 1000 GHz	Can be combined with any other propagation model	Assumes line of sight
gas (Gas)	Propagation of radio wave signal and its path loss due to oxygen and water vapor. For more information, see [5].	1GHz to 1000 GHz	Can be combined with any other propagation model	Assumes line of sight
fog (Fog)	Propagation of the radio wave signal and its path loss in cloud and fog. For more information, see [2].	10GHz to 1000 GHz	Can be combined with any other propagation model	Assumes line of sight

Empirical

Like atmospheric propagation models, empirical models predict path loss as a function of distance. Unlike atmospheric models, the close-in empirical model supports non-line-of-sight (NLOS) conditions.

Model	Description	Frequency	Combinations	Limitations
close-in (CloseIn)	Propagation of signals in urban macro cell scenarios. For more information, see [1].	No enforced range	Can be combined with rain, fog, and gas	—

Terrain

Terrain propagation models assume that propagation occurs between two points over a slice of terrain. Use these models to calculate the point-to-point path loss between sites over irregular terrain, including buildings.

Terrain models calculate path loss from free-space loss, terrain and obstacle diffraction, ground reflection, atmospheric refraction, and tropospheric scatter. They provide path loss estimates by combining physics with empirical data.

Model	Description	Frequency	Combinations	Limitations
longley-rice (LongleyRice)	Also known as Irregular Terrain Model (ITM). For more information, see [4].	20 MHz to 20 GHz	Can be combined with rain, fog, and gas	<ul style="list-style-type: none"> Designed for antenna heights from 0.5 to 3000 m Designed for distances from 1 to 2000 km
tirem (TIREM)	Terrain Integrated Rough Earth Model	1 MHz to 1000 GHz	Can be combined with rain, fog, and gas	<ul style="list-style-type: none"> Requires access to external TIREM library Antenna height maximum is 30000 m

Ray Tracing

Ray tracing models, represented by RayTracing objects, compute propagation paths using 3-D environment geometry [7][8]. They determine the path loss and phase shift of each ray using electromagnetic analysis, including tracing the horizontal and vertical polarizations of a signal through the propagation path. The path loss calculations include free-space loss, reflection losses, and diffraction losses. For each reflection and diffraction, the model calculates loss on the horizontal and vertical polarizations by using the Fresnel equation, the Uniform Theory of Diffraction (UTD), the relevant angles, and the real relative permittivity and conductivity of the interaction materials [5][6] at the specified frequency.

While the other supported models compute single propagation paths, ray tracing models compute multiple propagation paths.

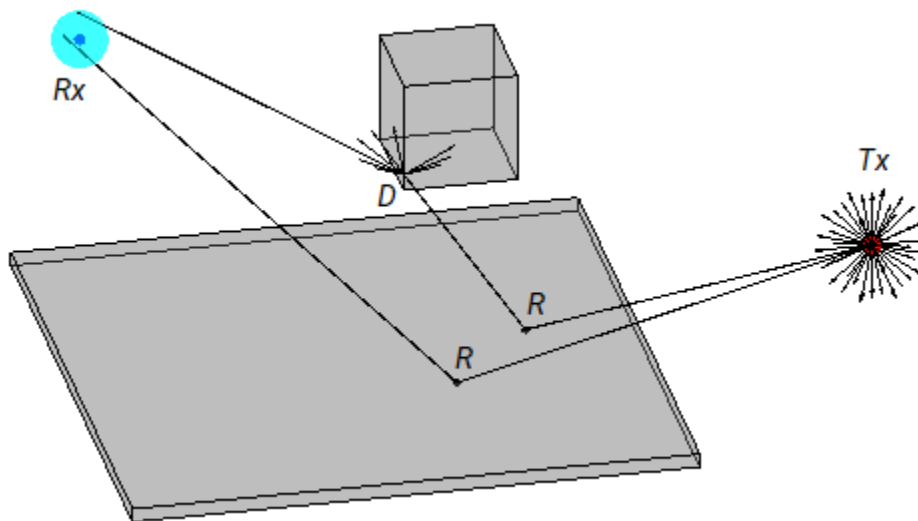
These models support both 3-D outdoor and indoor environments.

Ray Tracing Method	Description	Frequency	Combinations	Limitations
shooting and bouncing rays (SBR)	<ul style="list-style-type: none"> • Supports calculation of propagation paths for up to ten path reflections and two edge diffractions. • Calculates an approximate number of propagation paths with exact geometric accuracy. • Computational complexity increases linearly with the number of reflections and exponentially with the number of diffractions. The SBR method is generally faster than the image method. 	100 MHz to 100 GHz	Can be combined with rain, fog, and gas	Does not include effects from corner diffraction, refraction, or diffuse scattering

Ray Tracing Method	Description	Frequency	Combinations	Limitations
image	<ul style="list-style-type: none"> • Supports calculation of propagation paths for up to two path reflections. • Calculates an exact number of propagation paths with exact geometric accuracy. • Computational complexity increases exponentially with the number of reflections. 	100 MHz to 100 GHz	Can be combined with rain, fog, and gas	Does not include effects from diffraction, refraction, or diffuse scattering

SBR Method

This figure illustrates the SBR method for calculating propagation paths from a transmitter, T_x , to a receiver, R_x .



The SBR method launches many rays from a geodesic sphere centered at T_x . The geodesic sphere enables the model to launch rays that are approximately uniformly spaced.

Then, the method traces every ray from T_x and can model different types of interactions between the rays and surrounding objects, such as reflections, diffractions, refractions, and scattering. Note that the current implementation of the SBR method considers only reflections and edge diffractions.

- When a ray hits a flat surface, shown as R , the ray reflects based on the law of reflection.
- When a ray hits an edge, shown as D , the ray spawns many diffracted rays based on the law of diffraction [9][10]. Each diffracted ray has the same angle with the diffracting edge as the incident ray. The diffraction point then becomes a new launching point and the SBR method traces the diffracted rays in the same way as the rays launched from Tx . A continuum of diffracted rays forms a cone around the diffracting edge, which is commonly known as a Keller cone [10].

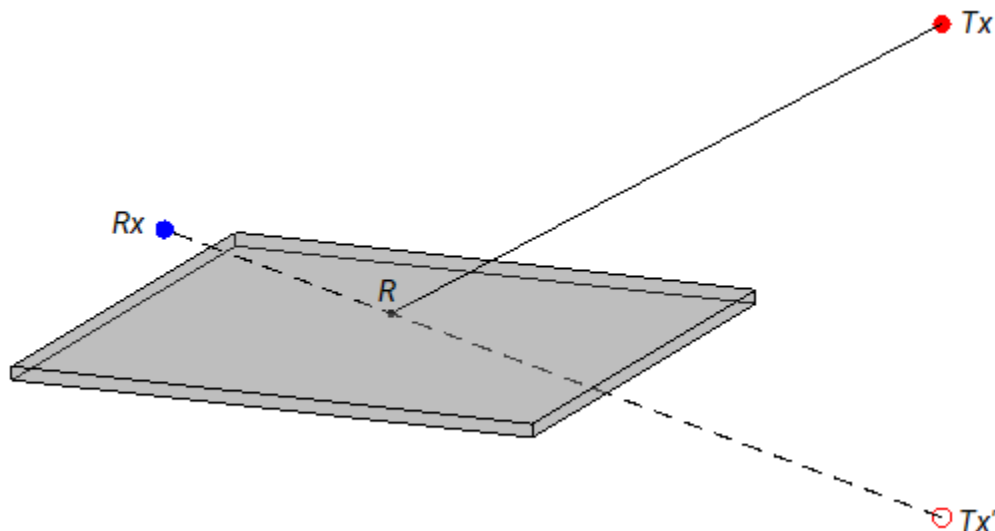
For each launched ray, the SBR method surrounds Rx with a sphere, called a reception sphere, with a radius that is proportional to the distance the ray travels and the average number of degrees between the launched rays. If the ray intersects the sphere, then the model considers the ray a valid path from Tx to Rx . The SBR method corrects the valid paths so that the paths have exact geometric accuracy.

When you increase the number of rays by decreasing the number of degrees between rays, the reception sphere becomes smaller. As a result, in some cases, launching more rays results in fewer or different paths. This situation is more likely to occur with custom 3-D scenarios created from STL files or triangulation objects than with scenarios that are automatically generated from OpenStreetMap buildings and terrain data.

The SBR method finds paths using double-precision floating-point computations.

Image Method

This figure illustrates the image method for calculating the propagation path of a single reflection ray for the same transmitter and receiver as the SBR method. The image method locates the image of Tx with respect to a planar reflection surface, Tx' . Then, the method connects Tx' and Rx with a line segment. If the line segment intersects the planar reflection surface, shown as R in the figure, then a valid path from Tx to Rx exists. The method determines paths with multiple reflections by recursively extending these steps. The image method finds paths using single-precision floating-point computations.



References

- [1] Sun, Shu, Theodore S. Rappaport, Timothy A. Thomas, Amitava Ghosh, Huan C. Nguyen, Istvan Z. Kovacs, Ignacio Rodriguez, Ozge Koymen, and Andrzej Partyka. "Investigation of Prediction

- Accuracy, Sensitivity, and Parameter Stability of Large-Scale Propagation Path Loss Models for 5G Wireless Communications.” *IEEE Transactions on Vehicular Technology* 65, no. 5 (May 2016): 2843–60. <https://doi.org/10.1109/TVT.2016.2543139>.
- [2] International Telecommunications Union Radiocommunication Sector. *Attenuation due to clouds and fog*. Recommendation P.840-6. ITU-R, approved September 30, 2013. <https://www.itu.int/rec/R-REC-P.840/en>.
- [3] International Telecommunications Union Radiocommunication Sector. *Specific attenuation model for rain for use in prediction methods*. Recommendation P.838-3. ITU-R, approved March 8, 2005. <https://www.itu.int/rec/R-REC-P.838/en>.
- [4] Hufford, George A., Anita G. Longley, and William A. Kissick. *A Guide to the Use of the ITS Irregular Terrain Model in the Area Prediction Mode*. NTIA Report 82-100. National Telecommunications and Information Administration, April 1, 1982.
- [5] International Telecommunications Union Radiocommunication Sector. *Effects of building materials and structures on radiowave propagation above about 100MHz*. Recommendation P.2040-1. ITU-R, approved July 29, 2015. <https://www.itu.int/rec/R-REC-P.2040/en>.
- [6] International Telecommunications Union Radiocommunication Sector. *Electrical characteristics of the surface of the Earth*. Recommendation P.527-5. ITU-R, approved August 14, 2019. <https://www.itu.int/rec/R-REC-P.527/en>.
- [7] Yun, Zhengqing, and Magdy F. Iskander. “Ray Tracing for Radio Propagation Modeling: Principles and Applications.” *IEEE Access* 3 (2015): 1089–1100. <https://doi.org/10.1109/ACCESS.2015.2453991>.
- [8] Schaubach, K.R., N.J. Davis, and T.S. Rappaport. “A Ray Tracing Method for Predicting Path Loss and Delay Spread in Microcellular Environments.” In *[1992 Proceedings] Vehicular Technology Society 42nd VTS Conference - Frontiers of Technology*, 932–35. Denver, CO, USA: IEEE, 1992. <https://doi.org/10.1109/VETEC.1992.245274>.
- [9] International Telecommunications Union Radiocommunication Sector. *Propagation by diffraction*. Recommendation P.526-15. ITU-R, approved October 21, 2019. <https://www.itu.int/rec/R-REC-P.526/en>.
- [10] Keller, Joseph B. “Geometrical Theory of Diffraction.” *Journal of the Optical Society of America* 52, no. 2 (February 1, 1962): 116. <https://doi.org/10.1364/JOSA.52.000116>.

See Also

Functions

propagationModel | raytrace

Related Examples

- “Ray Tracing for Wireless Communications” on page 4-12
- “Urban Link and Coverage Analysis Using Ray Tracing” on page 5-612
- “Visualize Antenna Coverage Map and Communication Links” on page 5-309
- “Planning a 5G Fixed Wireless Access Link over Terrain” on page 5-432

Ray Tracing for Wireless Communications

Introduction

Wireless communication systems use radio waves to transmit signals. Propagation modeling enables you to estimate the strength of signals based on system parameters such as frequency, antenna height, terrain properties, and building properties.

Theoretical and empirical models estimate path loss based on range, and are valid only for those environments that resemble the modeling environment. As a result, these models usually do not provide accurate temporal or spatial information. Unlike these models, ray tracing models are specific to the 3-D environment, and are therefore appropriate for scenarios such as urban environments.

For propagation modeling, a ray is an individual radio signal that [1]:

- Travels in a straight line through a homogeneous medium.
- Obeys the laws of reflection, refraction, and diffraction.
- Carries energy. Propagation models treat rays like tubes, where the energy density on the cross section becomes smaller as the ray interacts with the environment.

For a given 3-D environment, ray tracing models use numerical simulations to:

- Predict the paths of rays from transmitters to receivers. The models can find many rays from a transmitter to a receiver. The models derive the angle of departure, angle of arrival, and time of arrival from the paths.
- Estimate the path loss and phase change for each ray. Total path loss is the sum of interaction losses, free space loss, and, optionally, atmospheric loss.

A ray interacts with the environment in several ways [1].

Interaction	Description
Line of sight (LOS)	The ray travels directly from the transmitter to the receiver.
Reflection	The ray reflects off a surface according to the law of reflection.
Refraction (transmission)	The ray refracts as it moves into a new medium, according to the law of refraction.
Diffraction	The ray diffracts off a surface according to the law of diffraction. One ray can spawn many diffracted rays.
Diffuse scattering	The ray interacts with a rough surface such as the ocean or a building facade.

Use these functions to create ray tracing models, predict propagation paths, and calculate path losses and phase shifts.

- `propagationModel` — Create a ray tracing model as a `RayTracing` object. Specify options such as the ray tracing method, the maximum number of reflections and diffractions, and the interaction materials. You can use ray tracing models as inputs when conducting RF analysis, such

as when generating coverage maps by using the `coverage` function or when calculating total received power by using the `sigstrength` function.

- `raytrace` — Display propagation paths (rays) on a map or return propagation paths as `comm.Ray` objects. Each object represents the full path from the transmitter to the receiver, and contains information such as the path loss, phase shift, and types of surface interactions.
- `raypl` — Calculate the path loss and phase shift for a propagation path based on surface materials and antenna polarization types.

For an example that shows ray tracing in an urban environment, see “Urban Link and Coverage Analysis Using Ray Tracing” on page 5-612.

Ray Tracing Methods

The `propagationModel` and `raytrace` functions use a ray tracing model that finds LOS and non-line-of-sight (NLOS) paths.

- The model finds LOS paths by shooting a ray from the transmitter toward the receiver. If the ray does not interact with a surface before reaching the receiver, then an LOS path exists.
- The model finds NLOS paths by using either the shooting and bouncing rays (SBR) method [2] or the image method. You can specify the method by using the `propagationModel` function.

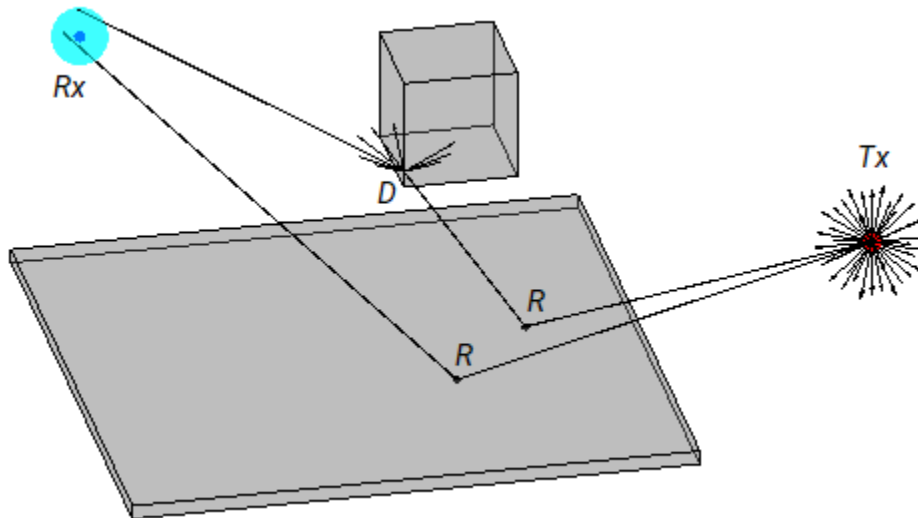
Choose a method based on the types of interactions you want to model, the computation speed, and the accuracy.

Method	Interaction Types	Computation Speed	Computation Accuracy
SBR (the default)	Includes effects from reflection and edge diffraction. Does not include effects from corner diffraction, refraction, or diffuse scattering. For each path, supports up to ten path reflections and two edge diffractions.	Computational complexity increases linearly with the number of reflections and exponentially with the number of diffractions. The SBR method is generally faster than the image method.	Calculates an approximate number of propagation paths with exact geometric accuracy.
Image	Includes effects from reflection. Does not include effects from diffraction, refraction, or diffuse scattering. For each path, supports up to two path reflections.	Computational complexity increases exponentially with the number of reflections.	Calculates an exact number of propagation paths with exact geometric accuracy.

When both the image and SBR methods find the same path, the points along the path are the same within a tolerance of machine precision for single-precision floating-point values.

SBR Method

This figure illustrates the SBR method for calculating propagation paths from a transmitter, T_x , to a receiver, R_x .



The SBR method launches many rays from a geodesic sphere centered at T_x . The geodesic sphere enables the model to launch rays that are approximately uniformly spaced.

Then, the method traces every ray from T_x and can model different types of interactions between the rays and surrounding objects, such as reflections, diffractions, refractions, and scattering. Note that the current implementation of the SBR method considers only reflections and edge diffractions.

- When a ray hits a flat surface, shown as R , the ray reflects based on the law of reflection.
- When a ray hits an edge, shown as D , the ray spawns many diffracted rays based on the law of diffraction [3][4]. Each diffracted ray has the same angle with the diffracting edge as the incident ray. The diffraction point then becomes a new launching point and the SBR method traces the diffracted rays in the same way as the rays launched from T_x . A continuum of diffracted rays forms a cone around the diffracting edge, which is commonly known as a Keller cone [4].

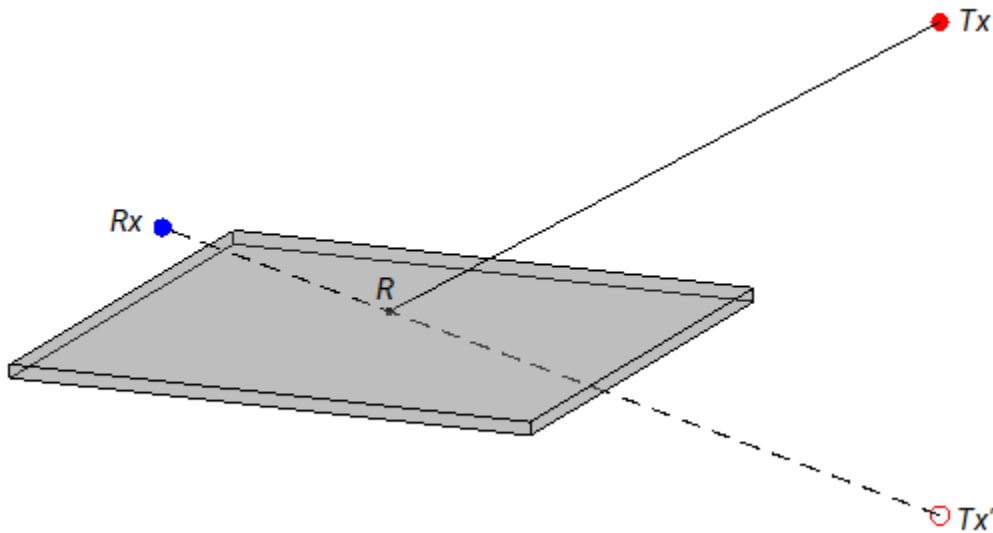
For each launched ray, the SBR method surrounds R_x with a sphere, called a reception sphere, with a radius that is proportional to the distance the ray travels and the average number of degrees between the launched rays. If the ray intersects the sphere, then the model considers the ray a valid path from T_x to R_x . The SBR method corrects the valid paths so that the paths have exact geometric accuracy.

When you increase the number of rays by decreasing the number of degrees between rays, the reception sphere becomes smaller. As a result, in some cases, launching more rays results in fewer or different paths. This situation is more likely to occur with custom 3-D scenarios created from STL files or triangulation objects than with scenarios that are automatically generated from OpenStreetMap buildings and terrain data.

The SBR method finds paths using double-precision floating-point computations.

Image Method

This figure illustrates the image method for calculating the propagation path of a single reflection ray for the same transmitter and receiver as the SBR method. The image method locates the image of T_x with respect to a planar reflection surface, T_x' . Then, the method connects T_x' and R_x with a line segment. If the line segment intersects the planar reflection surface, shown as R in the figure, then a valid path from T_x to R_x exists. The method determines paths with multiple reflections by recursively extending these steps. The image method finds paths using single-precision floating-point computations.



Propagation Loss

RayTracing objects calculate reflection and diffraction losses by tracking the horizontal and vertical polarizations of signals through the propagation path. Total power loss is the sum of free space loss, reflection loss, and diffraction loss.

Effect of Surface Materials

When a ray interacts with a surface, the surface material impacts the reflection losses.

The ray tracing model incorporates building and surface materials into the propagation loss calculations by using the complex relative permittivity of the surface, ϵ_r . The ITU-R P.2040-1 [5] and ITU-R P.527 [6] recommendations include methods, equations, and values used to calculate ϵ_r for a range of frequencies.

The equations for ϵ_r are:

$$\epsilon_r = \epsilon_r' + j\epsilon_r''$$

$$\epsilon_r'' = \frac{\sigma}{2\pi\epsilon_0 f}$$

where:

- ϵ_r' is the real relative permittivity.
- σ is the conductivity in S/m.
- ϵ_0 is the permittivity of free space (electric constant).
- f is the frequency in Hz.

For building materials, the ray tracing model calculates ϵ_r' and σ as:

$$\epsilon_r' = af^b$$

$$\sigma = cf^d,$$

where a , b , c , and d are constants determined by the surface material. For readability, this table shows the frequency range in GHz.

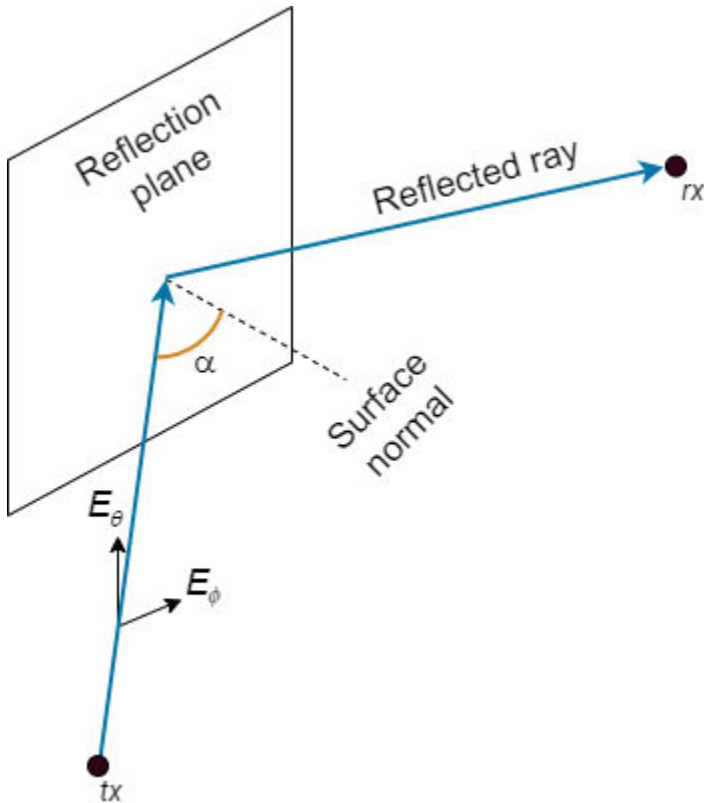
Material Class	Real Part of Relative Permittivity		Conductivity (S/m)		Frequency Range (GHz)
	a	b	c	d	
Vacuum (~ air)	1	0	0	0	[0.001, 100]
Concrete	5.31	0	0.0326	0.8095	[1, 100]
Brick	3.75	0	0.038	0	[1, 10]
Plasterboard	2.94	0	0.0116	0.7076	[1, 100]
Wood	1.99	0	0.0047	1.0718	[0.001, 100]
Glass	6.27	0	0.0043	1.1925	[0.1, 100]
Ceiling board	1.50	0	0.0005	1.1634	[1, 100]
Chipboard	2.58	0	0.0217	0.78	[1, 100]
Floorboard	3.66	0	0.0044	1.3515	[50, 100]
Metal	1	0	10^7	0	[1, 100]
Very dry ground	3	0	0.00015	2.52	[1, 10] only ^(a)
Medium dry ground	15	- 0.1	0.035	1.63	[1, 10] only ^(a)
Wet ground	30	- 0.4	0.15	1.30	[1, 10] only ^(a)

Note (a): For the three ground types (very dry, medium dry, and wet), the noted frequency limits cannot be exceeded.

For earth surfaces such as water, sea water, dry or wet ice, dry or wet soil, and vegetation, the ray tracing model calculates ϵ_r using the methods and equations presented in ITU-R P.527 [6].

Reflection Loss

This image shows a reflection path from a transmitter site tx to a receiver site rx .



The model determines polarization and reflection loss using these steps.

- 1 Track the propagation of the ray in 3-D space by calculating the propagation matrix P . The matrix is a repeating product, where i is the number of reflection points.

$$P = \prod_i P_i$$

For each reflection, calculate P_i by transforming the global coordinates of the incident electromagnetic field into the local coordinates of the reflection plane, multiplying the result by a reflection coefficient matrix, and transforming the coordinates back into the original global coordinate system [7]. The equations for P_i and P_0 are:

$$P_i = [s_{out} \ p_{out} \ k_{out}]_i \begin{bmatrix} R_V(\alpha) & 0 & 0 \\ 0 & R_H(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}_i [s_{in} \ p_{in} \ k_{in}]_i^{-1}$$

$$P_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where:

- s , p , and k form a basis for the plane of incidence (the plane created by the incident ray and the surface normal of the reflection plane). s and p are perpendicular and parallel, respectively, to the plane of incidence.

- k_{in} and k_{out} are the directions (in global coordinates) of the incident and exiting rays, respectively.
- s_{in} and s_{out} are the directions (in global coordinates) of the horizontal polarizations for the incident and exiting rays, respectively.
- p_{in} and p_{out} are the directions (in global coordinates) of the vertical polarizations for the incident and exiting rays, respectively.
- R_H and R_V are the Fresnel reflection coefficients for the horizontal and vertical polarizations, respectively. α is the incident angle of the ray and ϵ_r is the complex relative permittivity of the material.

$$R_H(\alpha) = \frac{\cos(\alpha) - \sqrt{(\epsilon_r - \sin^2(\alpha))/\epsilon_r^2}}{\cos(\alpha) + \sqrt{(\epsilon_r - \sin^2(\alpha))/\epsilon_r^2}}$$

$$R_V(\alpha) = \frac{\cos(\alpha) - \sqrt{\epsilon_r - \sin^2(\alpha)}}{\cos(\alpha) + \sqrt{\epsilon_r - \sin^2(\alpha)}}$$

- 2 Project the propagation matrix P into a 2-by-2 polarization matrix R . The model rotates the coordinate systems for the transmitter and receiver so that they are in global coordinates.

$$R = \begin{bmatrix} H_{in} \cdot H_{rx} & V_{in} \cdot H_{rx} \\ H_{in} \cdot V_{rx} & V_{in} \cdot V_{rx} \end{bmatrix}$$

$$H_{in} = P(V_{tx} \times k_{tx})$$

$$V_{in} = PV_{tx}$$

where:

- H_{rx} and V_{rx} are the directions (in global coordinates) of the horizontal (E_θ) and vertical (E_ϕ) polarizations, respectively, for the receiver.
 - H_{in} and V_{in} are the directions (in global coordinates) of the propagated horizontal and vertical polarizations, respectively.
 - V_{tx} is the direction (in global coordinates) of the nominal vertical polarization for the ray departing the transmitter.
 - k_{tx} is the direction (in global coordinates) of the ray departing the transmitter.
- 3 Specify the normalized horizontal and vertical polarizations of the electric field at the transmitter and receiver by using the 2-by-1 Jones polarization vectors J_{tx} and J_{rx} , respectively. If either the transmitter or receiver are unpolarized, then the model assumes $J_{tx} = J_{rx} = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.
 - 4 Calculate the polarization and reflection loss IL by combining R , J_{tx} , and J_{rx} .

$$IL = -20 \log_{10} |J_{rx}^{-1} R J_{tx}|$$

Diffraction Loss

The model calculates diffraction loss by using computations based on the Uniform Theory of Diffraction (UTD) [8].

For a first order signal diffraction, the equation for path loss, PL_D , is:

$$PL_D = JV_{rx}'H_{diff1}JV_{tx},$$

where:

- JV_{rx} and JV_{tx} are polarization vectors for the receiver and transmitter, respectively, specified as Jones vectors.
- H_{diff1} is the diffraction matrix.

The equation for the diffraction matrix contains three terms.

- The first term is a geometric coupling matrix that rotates the polarization vector from the basis of the ray coordinates to the basis of the edge-fixed incidence plane. The edge-fixed incidence plane contains the ray and the edge.
- The second term is a polarization matrix containing diffraction coefficients for the local horizontal and vertical polarizations, D_{\perp} and D_{\parallel} , and an amplitude scaling factor. For more information about the diffraction coefficients and amplitude scaling factor, see [3] and [8].
- The third term is a geometric coupling matrix that rotates the polarization vector from the basis of the edge-fixed incidence plane to the basis of the edge-fixed diffraction plane. The edge-fixed diffraction plane contains the diffracted ray and the edge.

References

- [1] Yun, Zhengqing, and Magdy F. Iskander. "Ray Tracing for Radio Propagation Modeling: Principles and Applications." *IEEE Access* 3 (2015): 1089–1100. <https://doi.org/10.1109/ACCESS.2015.2453991>.
- [2] Schaubach, K.R., N.J. Davis, and T.S. Rappaport. "A Ray Tracing Method for Predicting Path Loss and Delay Spread in Microcellular Environments." In *[1992 Proceedings] Vehicular Technology Society 42nd VTS Conference - Frontiers of Technology*, 932–35. Denver, CO, USA: IEEE, 1992. <https://doi.org/10.1109/VETEC.1992.245274>.
- [3] International Telecommunications Union Radiocommunication Sector. *Propagation by diffraction*. Recommendation P.526-15. ITU-R, approved October 21, 2019. <https://www.itu.int/rec/R-REC-P.526/en>.
- [4] Keller, Joseph B. "Geometrical Theory of Diffraction." *Journal of the Optical Society of America* 52, no. 2 (February 1, 1962): 116. <https://doi.org/10.1364/JOSA.52.000116>.
- [5] International Telecommunications Union Radiocommunication Sector. *Effects of building materials and structures on radiowave propagation above about 100MHz*. Recommendation P.2040-1. ITU-R, approved July 29, 2015. <https://www.itu.int/rec/R-REC-P.2040/en>.
- [6] International Telecommunications Union Radiocommunication Sector. *Electrical characteristics of the surface of the Earth*. Recommendation P.527-5. ITU-R, approved August 14, 2019. <https://www.itu.int/rec/R-REC-P.527/en>.
- [7] Chipman, Russell A., Garam Young, and Wai Sze Tiffany Lam. "Fresnel Equations." In *Polarized Light and Optical Systems*. Optical Sciences and Applications of Light. Boca Raton: Taylor & Francis, CRC Press, 2019.
- [8] McNamara, D. A., C. W. I. Pistorius, and J. A. G. Malherbe. *Introduction to the Uniform Geometrical Theory of Diffraction*. Boston: Artech House, 1990.

See Also

Functions

propagationModel | raytrace | raypl | buildingMaterialPermittivity |
earthSurfacePermittivity

Objects

RayTracing | comm.Ray

Related Examples

- “Urban Link and Coverage Analysis Using Ray Tracing” on page 5-612

Antenna Toolbox Examples

- “Port Analysis of Antenna” on page 5-4
- “Current Visualization on Antenna Surface” on page 5-10
- “Antenna Far-Field Visualization” on page 5-16
- “Antenna Near-Field Visualization” on page 5-27
- “Wave Impedance” on page 5-32
- “Antenna Array Analysis” on page 5-36
- “Modeling Infinite Ground Plane in Antennas and Arrays” on page 5-48
- “Infinite Array Analysis” on page 5-58
- “Parallelization of Antenna and Array Analyses” on page 5-68
- “Analysis of Monopole Impedance” on page 5-72
- “Analysis of Dipole Impedance” on page 5-77
- “Monopole Measurement Comparison” on page 5-80
- “Equiangular Spiral Antenna Design Investigation” on page 5-85
- “Archimedean Spiral Design Investigation” on page 5-88
- “Patch Antenna on Dielectric Substrate” on page 5-95
- “Sector Antenna for 2.4 GHz Wi-Fi™” on page 5-107
- “Design PIFA for WLAN Wi-Fi™ Applications” on page 5-117
- “Helical Antenna Design” on page 5-124
- “Reflector Backed Equiangular Spiral” on page 5-129
- “Impedance Matching of Non-resonant (Small) Monopole” on page 5-141
- “Direct Search Based Optimization of Six-Element Yagi-Uda Antenna” on page 5-147
- “Antenna Diversity Analysis for 800 MHz MIMO” on page 5-159
- “Polarization Analysis for X-band Microstrip Patch Antenna” on page 5-166
- “FMCW Patch Antenna Array” on page 5-172
- “Modeling Mutual Coupling in Large Arrays Using Infinite Array Analysis” on page 5-184
- “Modeling Mutual Coupling in Large Arrays Using Embedded Element Pattern” on page 5-195
- “Modeling Resonant Coupled Wireless Power Transfer System” on page 5-209
- “Crossed-Dipole (Turnstile) Antenna and Array” on page 5-218
- “Visualize Antenna Field Strength Map on Earth” on page 5-230
- “RFID Antenna Design” on page 5-236
- “Wideband Blade Dipole Antenna and Array” on page 5-243
- “Analysis of Inset-Feed Patch Antenna on Dielectric Substrate” on page 5-249
- “Read, Visualize and Write MSI Planet Antenna Files” on page 5-256
- “Custom Radiation Pattern and Fields” on page 5-263
- “Double-Slot Cavity Patch on TMM10 Substrate” on page 5-272

- “Effect of Mutual Coupling on MIMO Communication” on page 5-281
- “Switched Beam Array with Butler Matrix” on page 5-289
- “Antenna Model Generation and Full-Wave Analysis From A Photo” on page 5-294
- “Visualize Antenna Coverage Map and Communication Links” on page 5-309
- “Modeling and Analysis of Single Layer Multi-band U-Slot Patch Antenna” on page 5-317
- “Antenna Array Beam Scanning Visualization on a Map” on page 5-335
- “Modeling Planar Photonic Band Gap Structure” on page 5-344
- “Plane Wave Excitation - Scattering Solution” on page 5-353
- “Design, Analysis, and Prototyping of Microstrip-Fed Wide-Slot Antenna” on page 5-356
- “Comparison of Antenna Array Transmit and Receive Manifold” on page 5-364
- “Verification of Far-Field Array Pattern Using Superposition with Embedded Element Patterns” on page 5-372
- “Metasurface Antenna Modeling” on page 5-381
- “Modeling and Analysis of Probe-Fed Stacked Patch Antenna” on page 5-395
- “Design Internally Matched Ultra-Wideband Vivaldi Antenna” on page 5-410
- “Planning a 5G Fixed Wireless Access Link over Terrain” on page 5-432
- “SINR Map for a 5G Urban Macro-Cell Test Environment” on page 5-447
- “Surrogate Based Optimization Design of Six-Element Yagi-Uda Antenna” on page 5-459
- “Model and Analyze Dual Polarized Patch Microstrip Antenna” on page 5-476
- “Installed Antenna Analysis - Modelling Antennas with Platforms” on page 5-486
- “3D Reconstruction of Radiation Pattern From 2D Orthogonal Slices” on page 5-497
- “Loading Using Lumped Elements” on page 5-510
- “Planning Radar Network Coverage over Terrain” on page 5-521
- “Visualize Viewsheds and Coverage Maps Using Terrain” on page 5-532
- “Subarrays in Large Finite Array For Hybrid Beamforming” on page 5-562
- “Pattern Analysis of the Symmetric Parabolic Reflector” on page 5-570
- “Radar Cross Section Benchmarking” on page 5-579
- “Design and Analyze Cassegrain Antenna” on page 5-585
- “Discone Antenna for TV Broadcasting System” on page 5-595
- “Analysis of Biquad Yagi for Wi-Fi Applications” on page 5-600
- “Analysis of Broad-wall Slotted Array Waveguide for High Frequency Applications” on page 5-606
- “Urban Link and Coverage Analysis Using Ray Tracing” on page 5-612
- “Maximizing Gain and Improving Impedance Bandwidth of E-Patch Antenna” on page 5-625
- “Optimization of Antenna Array Elements Using Antenna Array Designer App” on page 5-636
- “Import and Analyze Custom 3-D Antenna Geometry” on page 5-647
- “Model, Analyze and Compare Horn Antennas” on page 5-673
- “Multiband Nature and Miniaturization of Fractal Antennas” on page 5-683
- “ISM Band Patch Microstrip Antennas and Mutually Coupled Patches” on page 5-695
- “Design and Analyze Curved Reflectors” on page 5-705

- “VHF/UHF Biconical Antenna for Testing Applications” on page 5-714
- “Ultra-Wideband (UWB) Planar Monopole Antennas” on page 5-721
- “Maximize Impedance Bandwidth of Triangular Patch Antenna” on page 5-736
- “Create Antenna Model from Gerber Files” on page 5-745
- “Design Log-Periodic Sawtooth Planar Antenna for UHF Ultra-Wideband Applications” on page 5-750
- “Calculate Radiation Efficiency of Antenna” on page 5-759
- “Design 77 GHz Patch Microstrip for Automotive Radar Receiver” on page 5-774
- “Infinite Array of Microstrip Patch Antenna on Teflon Substrate” on page 5-780
- “Modeling Wire Antenna and Arrays” on page 5-786
- “Analysis of Electrically Large Structures Using Hybrid MoM and FMM” on page 5-793
- “Analyze Cylindrical Reflector Antenna with Horn Array Feed” on page 5-802
- “Design and Analysis Using PCB Antenna Designer” on page 5-811
- “Define PCB Antenna Layers” on page 5-813
- “Design H-Notch Patch Unit Element” on page 5-818
- “Analyze H-Notch Unit Element” on page 5-831
- “Design, Analyze, and Export 1-by-2 H-Notch Linear Array” on page 5-836
- “Design, Analyze, and Optimize H-Notch Patch Using Design Variables” on page 5-852
- “Design Series-Fed Patch Antenna Array for 5G Base Station” on page 5-864
- “PCB Antenna for USB Dongle and BLE Applications” on page 5-874
- “Miniaturize Patch Antennas Using Metamaterial-Inspired Technique ” on page 5-883
- “Design and Analysis of Compact Ultra-Wideband MIMO Antenna Array” on page 5-893
- “Direction of Arrival Determination Using Full-Wave Electromagnetic Analysis” on page 5-903
- “Field Analysis of Monopole Antenna” on page 5-909
- “Design and Analysis of a Diamond-Shaped Antenna for Ultra-Wideband Applications” on page 5-920
- “Board Thickness versus Dielectric Thickness in PCB ” on page 5-927
- “Analyze Metal Conductors in Helical Dipole Antenna” on page 5-936
- “Design, Analyze, and Prototype 2x2 Patch Array Antenna” on page 5-959
- “Modified Sierpinski Monopole Fractal Antenna for Dual-Band Application” on page 5-983
- “Design and Analyze Parabolic-Reflector-Backed Wideband Eggcrate Array” on page 5-990
- “Design and Analyze Tapered-Slot SIW Filtenna ” on page 5-1000
- “Antennas on A Glider as Platform” on page 5-1014
- “Electromagnetic Analysis of Intelligent Reflecting Surface” on page 5-1017
- “Design S-Band Monopulse Tracking RADAR Antenna ” on page 5-1025

Port Analysis of Antenna

This example quantifies terminal antenna parameters, with regard to the antenna port. The antenna is a one-port network. The antenna port is a physical location on the antenna where an RF source is connected to it. The terminal port parameters supported in Antenna Toolbox™ are

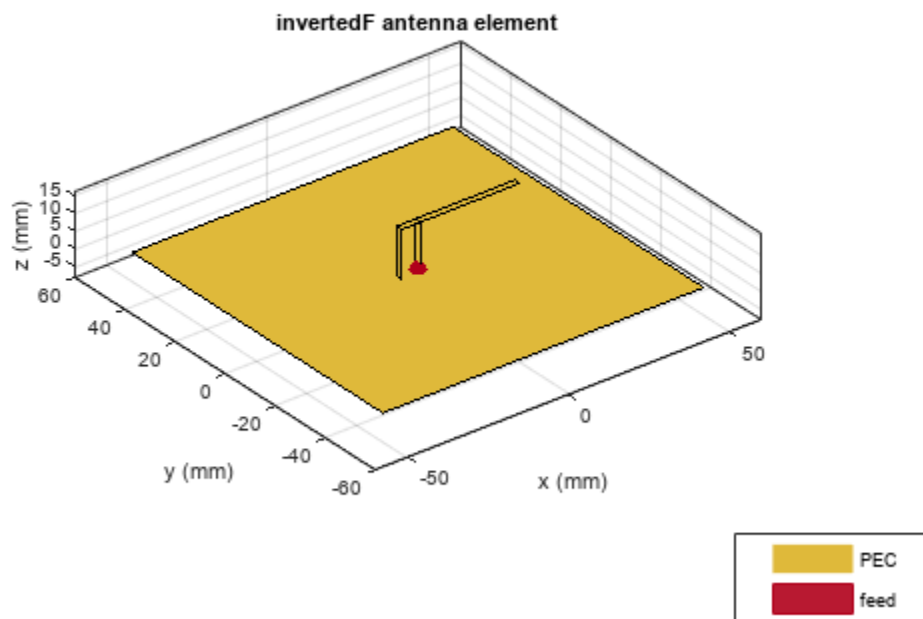
- Antenna impedance
- Return loss
- S-parameters
- VSWR

The example uses a Planar Inverted-F Antenna (PIFA), performs the corresponding computational analysis, and returns all terminal antenna parameters listed above.

Create Inverted-F Antenna

Create the default geometry for the PIFA antenna. The (small) red dot on the antenna structure is the feed point location where an input voltage generator is applied. It is the port of the antenna. In Antenna Toolbox™, all antennas are excited by a time-harmonic voltage signal with the amplitude of 1 V at the port. The port must connect two distinct conductors; it has an infinitesimally small width.

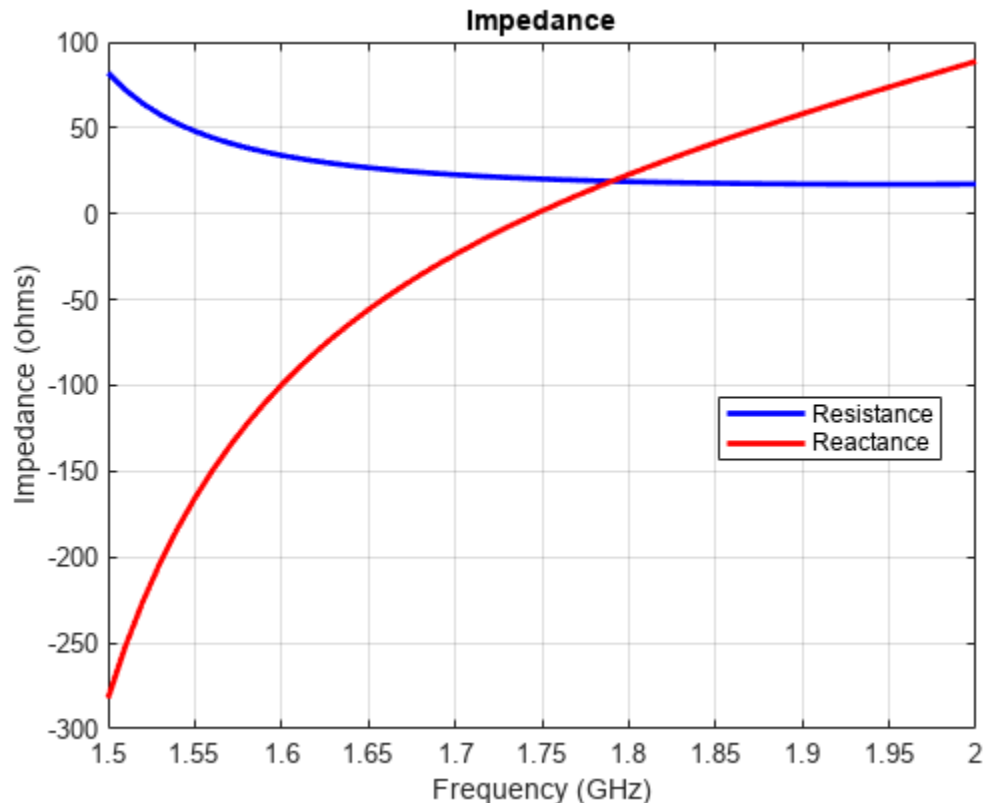
```
ant = invertedF;  
show(ant);
```



Impedance

To plot the antenna impedance, specify the frequency band over which the data needs to be plotted using `impedance` function. The antenna impedance is calculated as the ratio of the phasor voltage (which is simply 1) and the phasor current at the port.

```
freq = linspace(1.5e9, 2.0e9, 51);
figure;
impedance(ant, freq);
```

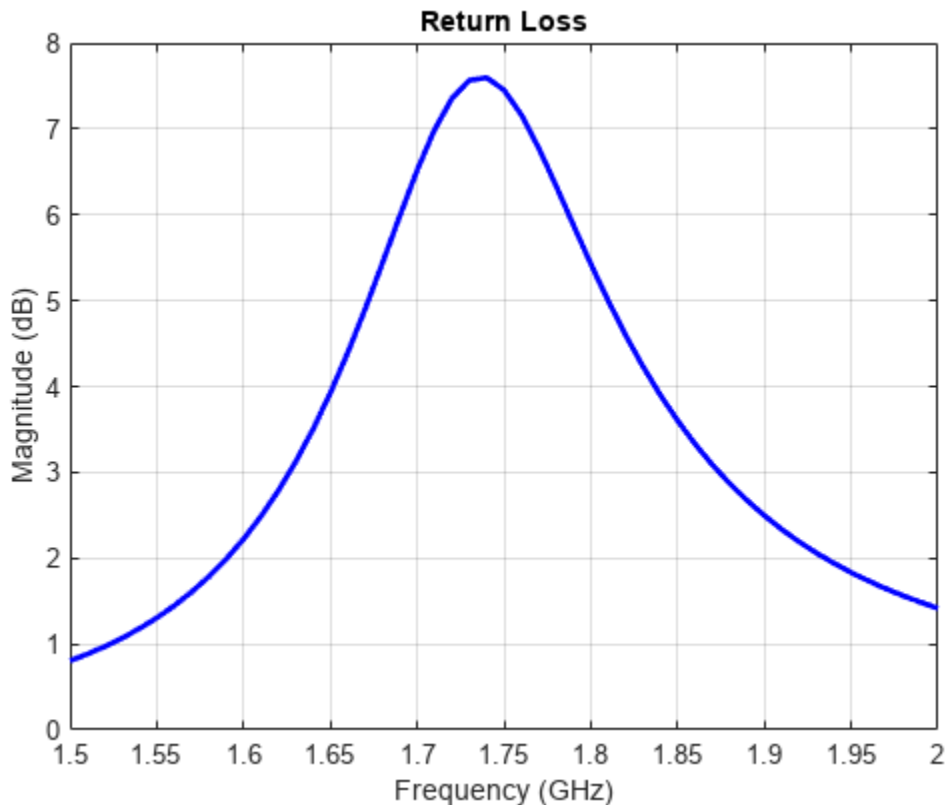


The plot displays the real part of the impedance, i.e. resistance as well as its imaginary part, i.e. reactance, over the entire frequency band. Antenna resonant frequency is defined as the frequency at which the reactance of the antenna is exactly zero. Looking at the impedance plot, we observe that the inverted-F antenna resonates at 1.74 GHz. The resistance value at that frequency is about 20Ω . The reactance values for the antenna are negative (capacitive) before resonance and become positive (inductive) after resonance, indicating that it is the series resonance of the antenna (modeled by a series RLC circuit). If the impedance curve goes from positive reactance to negative, it is the parallel resonance [1] (modelled by a parallel RLC circuit).

Return Loss

To plot the return loss of an antenna, specify the frequency band over which the data needs to be plotted. Return loss is the measure of the effectiveness of power delivery from the transmission line to an antenna. Quantitatively, the return loss is the ratio, in dB, of the power sent towards the antenna and the power reflected back. It is a positive quantity for passive devices. A negative return loss is possible with active devices [2].

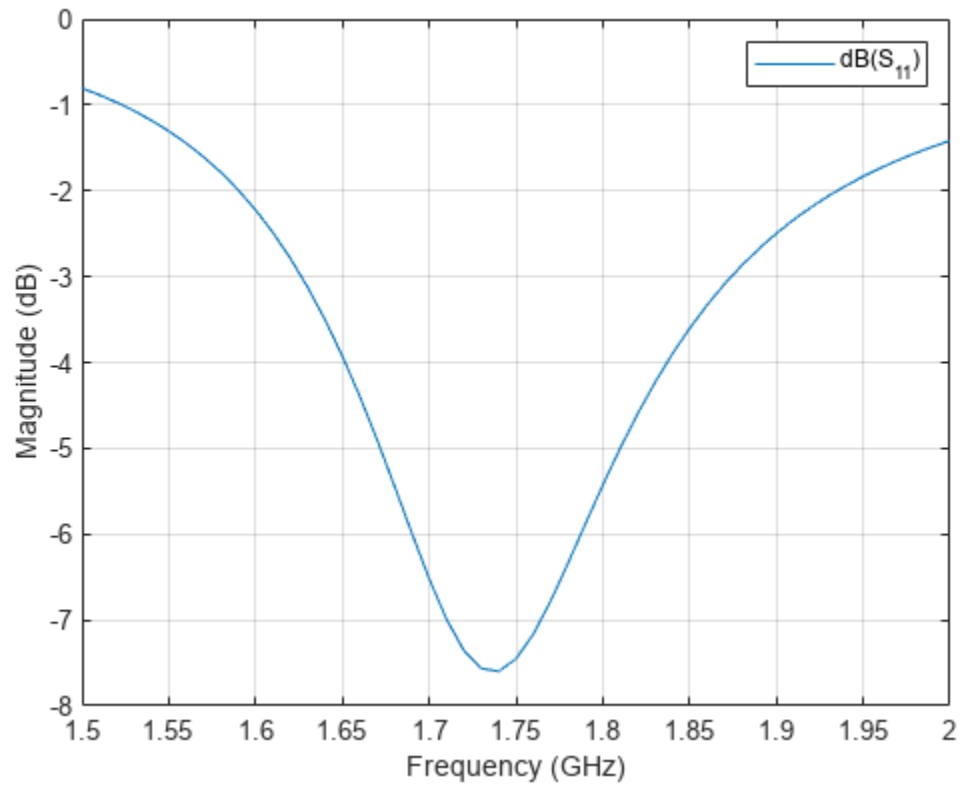
```
figure;  
returnLoss(ant, freq);
```



Reflection Coefficient

The return loss introduced above is rarely used for the antenna analysis. Instead, a reflection coefficient or S_{11} in dB is employed, which is also often mistakenly called the "return loss" [2]. In fact, the reflection coefficient in dB is the negative of the return loss as seen in the following figure. The reflection coefficient describes a relative fraction of the incident RF power that is reflected back due to the impedance mismatch. This mismatch is the difference between the input impedance of the antenna and the characteristic impedance of the transmission line (or the generator impedance when the transmission line is not present). The characteristic impedance is the reference impedance. The `sparameters` function used below accepts the reference impedance as its third argument. The same is valid for the `returnLoss` function. By default, we assume the reference impedance of 50Ω .

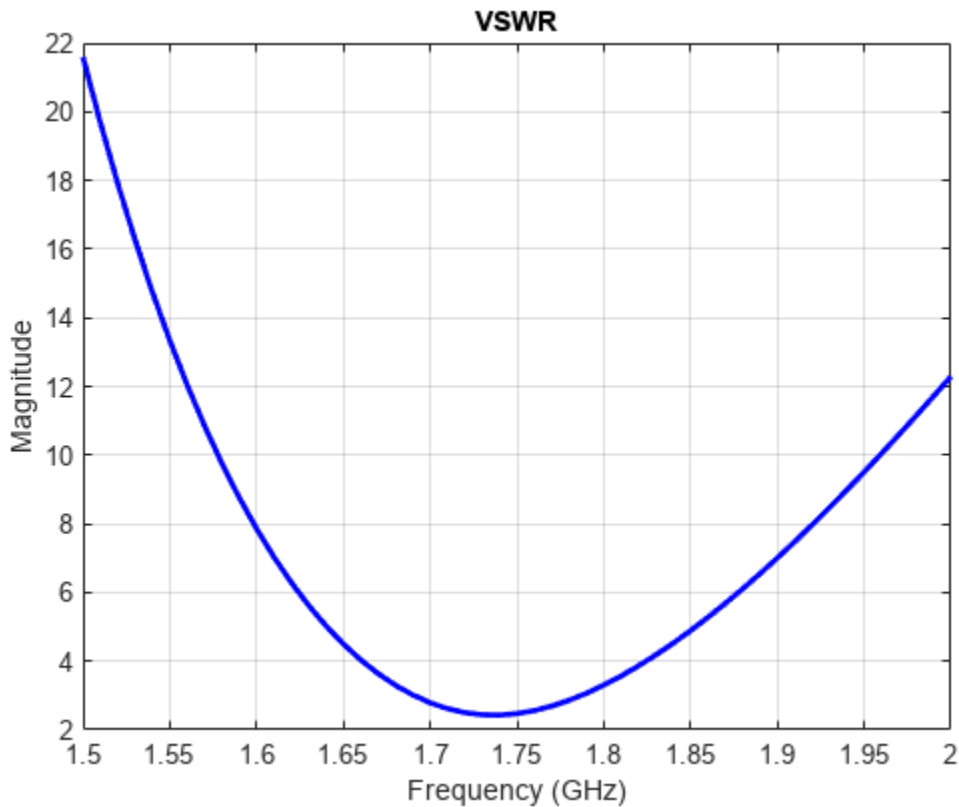
```
S = sparameters(ant, freq);  
figure;  
rfplot(S);
```



Voltage Standing Wave Ratio (VSWR)

The VSWR of the antenna can be plotted using the function `vswr` used below. A VSWR value of 1.5:1 means that a maximum standing wave amplitude is 1.5 times greater than the minimum standing wave amplitude. The standing waves are generated because of impedance mismatch at the port. The VSWR is expressed through the reflection coefficient as $(1 + |S_{11}|)/(1 - |S_{11}|)$.

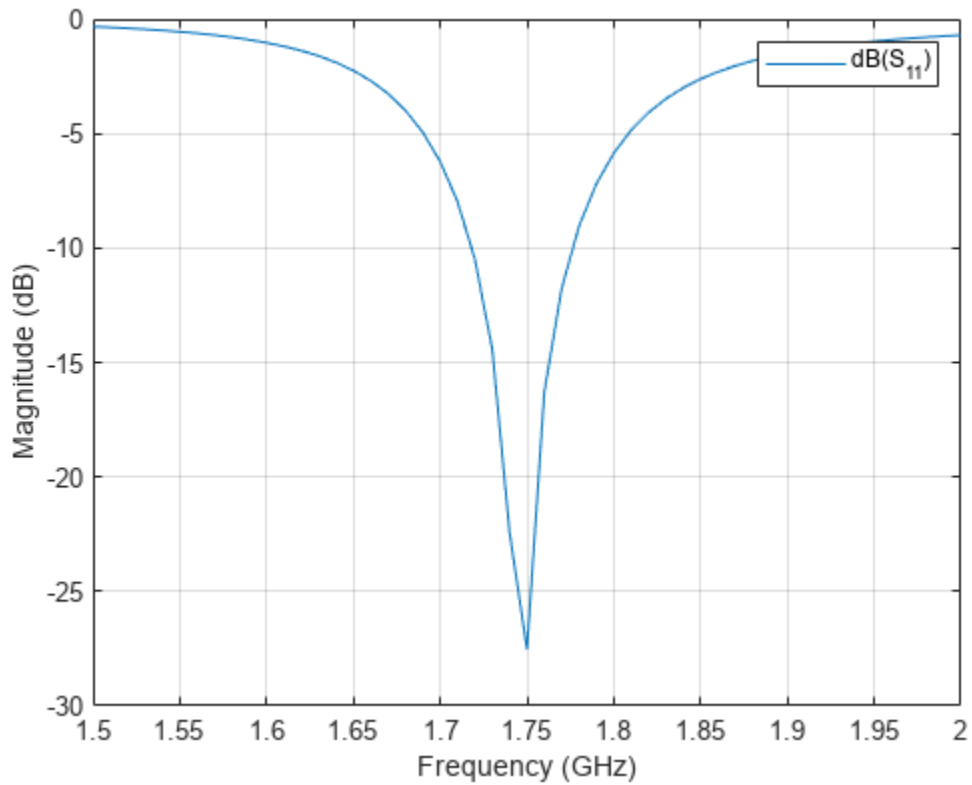
```
figure;  
vswr(ant, freq);
```



Antenna Bandwidth

Bandwidth is a fundamental antenna parameter. The antenna bandwidth is the band of frequencies over which the antenna can properly radiate or receive power. Often, the desired bandwidth is one of the critical parameters used to decide an antenna type. Antenna bandwidth is usually the frequency band over which the magnitude of the reflection coefficient is below -10 dB, or the magnitude of the return loss is greater than 10 dB, or the VSWR is less than approximately 2. All these criteria are equivalent. We observe from the previous figures that the PIFA has no operating bandwidth in the frequency band of interest. The bandwidth is controlled by the proper antenna design. Sometimes, the reference impedance may be changed too. In the impedance plot we observe that the resistance of the present antenna is close to 20Ω at the resonance. Choose the reference impedance of 20Ω instead of 50Ω and plot the reflection coefficient.

```
S = sparameters(ant, freq, 20);
figure;
rfplot(S);
```



Now, we observe the reflection coefficient of less than -10 dB over the frequency band from 1.71 to 1.77 GHz. This is the antenna bandwidth. The same conclusion holds when using the VSWR or return loss calculations.

References

- [1] C. A. Balanis, Antenna Theory. Analysis and Design, p. 514, Wiley, New York, 3rd Edition, 2005.
- [2] T. S. Bird, "Definition and Misuse of Return Loss," Antennas and Propagation Magazine, April 2009.

See Also

"Current Visualization on Antenna Surface" on page 5-10 | "Analysis of Monopole Impedance" on page 5-72

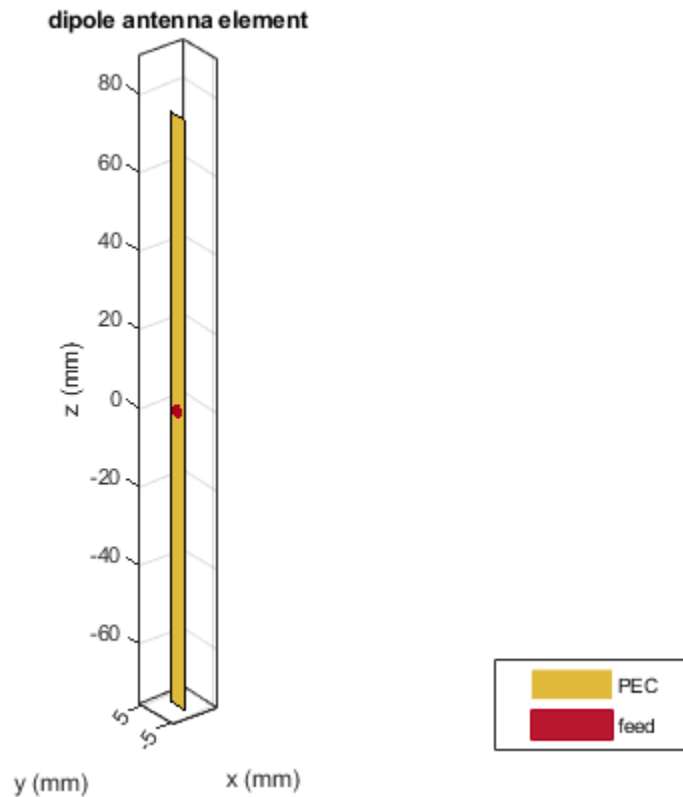
Current Visualization on Antenna Surface

This example shows how to visualize surface currents on the half-wavelength dipole and how to observe the individual current components. Finally, it shows how to interact with the colorbar to change its dynamic range to better visualize the surface currents.

Create Dipole Antenna

Design the dipole antenna to resonate around 1 GHz. The wavelength at this frequency is 30 cm. The dipole length is equal to half-wavelength, which corresponds to 15 cm. The width of the dipole strip is chosen to be 5 mm.

```
mydipole = dipole('Length',15e-2, 'Width', 5e-3);
show(mydipole);
```



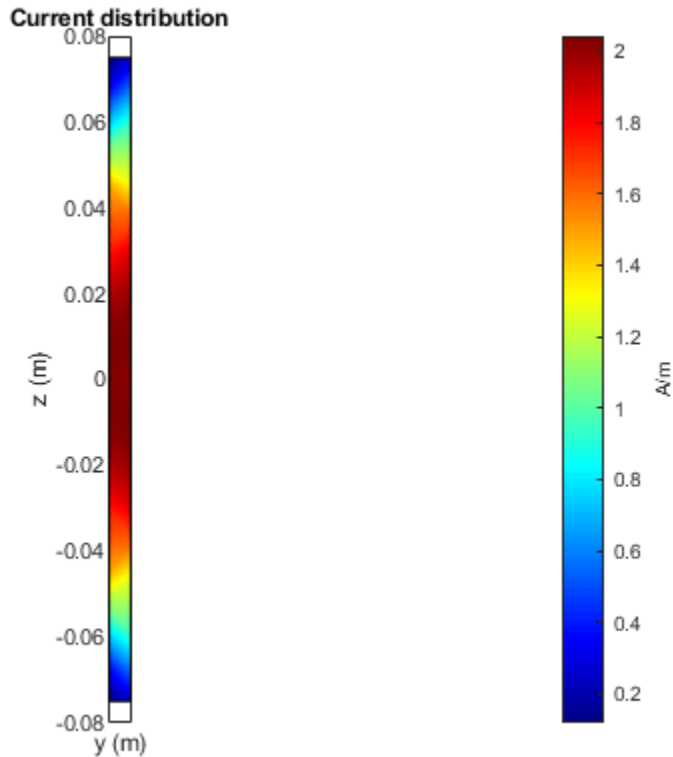
Calculate Current Distribution on Dipole Surface

Since the dipole length is 15 cm, choose the operation frequency as $f = c/(2 * l)$, where c is the speed of light. A current distribution in the form of the half period of a sine wave along the dipole length is observed with the maximum occurring at the antenna center. A periodic current distribution along the dipole is further observed with multiple maxima and minima at higher resonant frequencies ($2f, 3f, \dots$). This periodic current distribution is typical for dipoles and similar resonant wire antennas [1].


```

c = 2.99792458e8;
f = c/(2*mydipole.Length);
current(mydipole, f);
view(90,0);

```



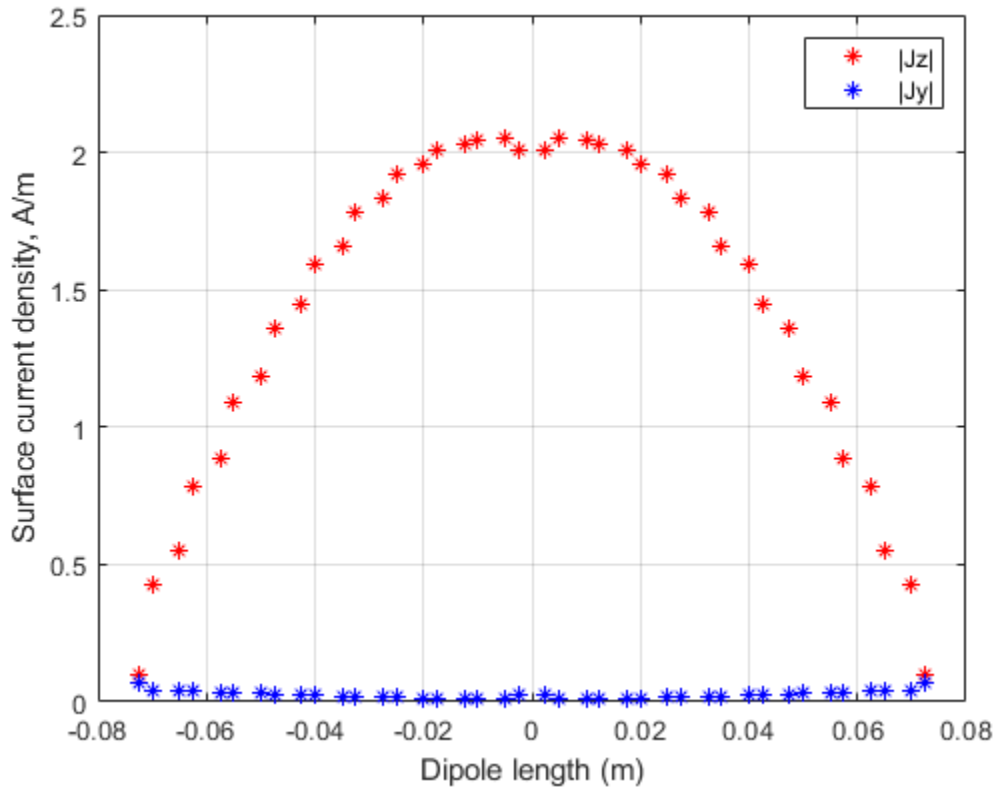
Calculate and Plot Individual Current Components

Specify the output arguments in the function `current` and access the individual current components. The vector points correspond to the triangle centers where the current values are evaluated. The plot below displays the longitudinal (z) component and the transverse (y) component of the surface current density magnitude.

```

[C, points] = current(mydipole, f);
Jy = abs(C(2,:));
Jz = abs(C(3,:));
figure;
plot(points(3,:), Jz, 'r*', points(3,:), Jy, 'b*');
grid on;
xlabel('Dipole length (m)')
ylabel('Surface current density, A/m');
legend('|Jz|', '|Jy|');

```

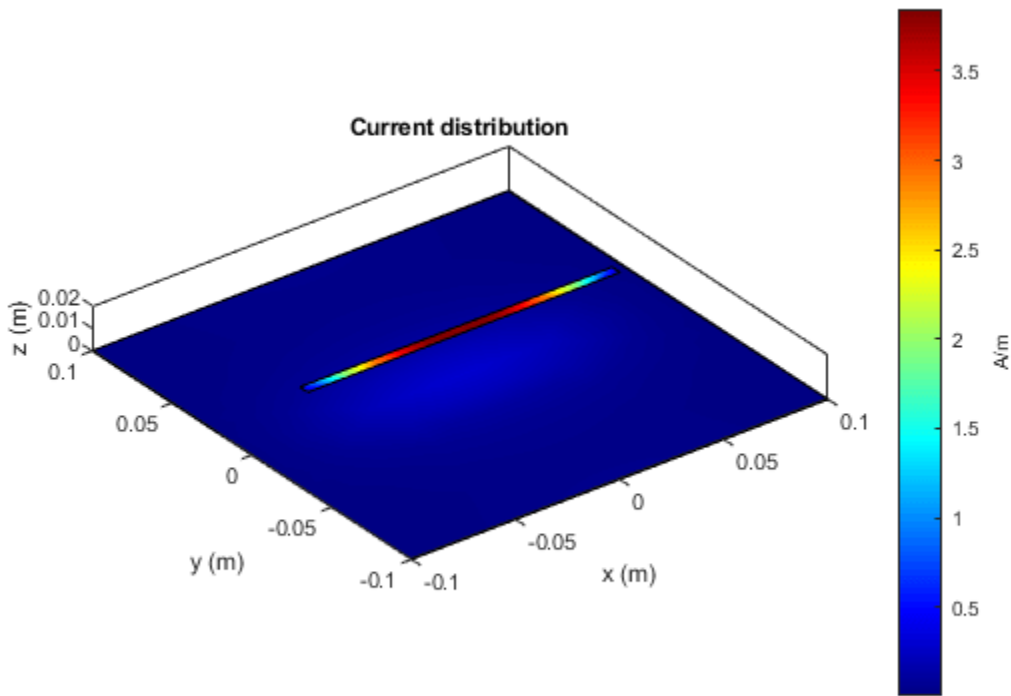


This current distribution is typical in the MoM analysis [1], [2]. Better (smoother) results will be obtained with finer triangular surface meshes and when the current is plotted exactly on the centerline of the strip.

Place the Dipole in Front of Reflector

Place the dipole in front of a finite planar reflector by choosing the dipole antenna as an exciter for the reflector antenna. Orient the dipole so that it is parallel to the reflector. This is done by modifying the tilt of the exciter (the dipole) so that it now lies along the x-axis. A tilt of 90 degrees is applied about the Y axis to achieve this. The spacing between the dipole and reflector is chosen to be 2 cm.

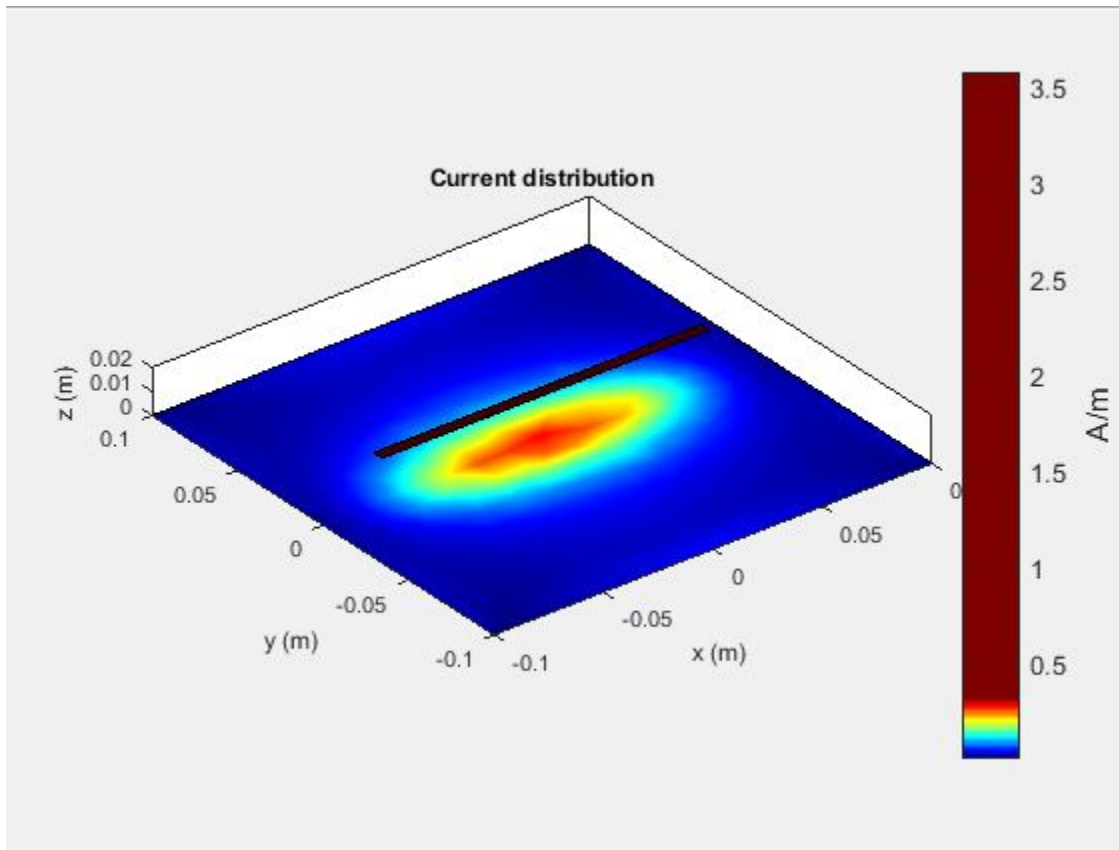
```
myreflector = reflector('Exciter', mydipole, 'Spacing', 0.02);
myreflector.Exciter.Tilt = 90;
myreflector.Exciter.TiltAxis = [0 1 0];
current(myreflector, f);
```



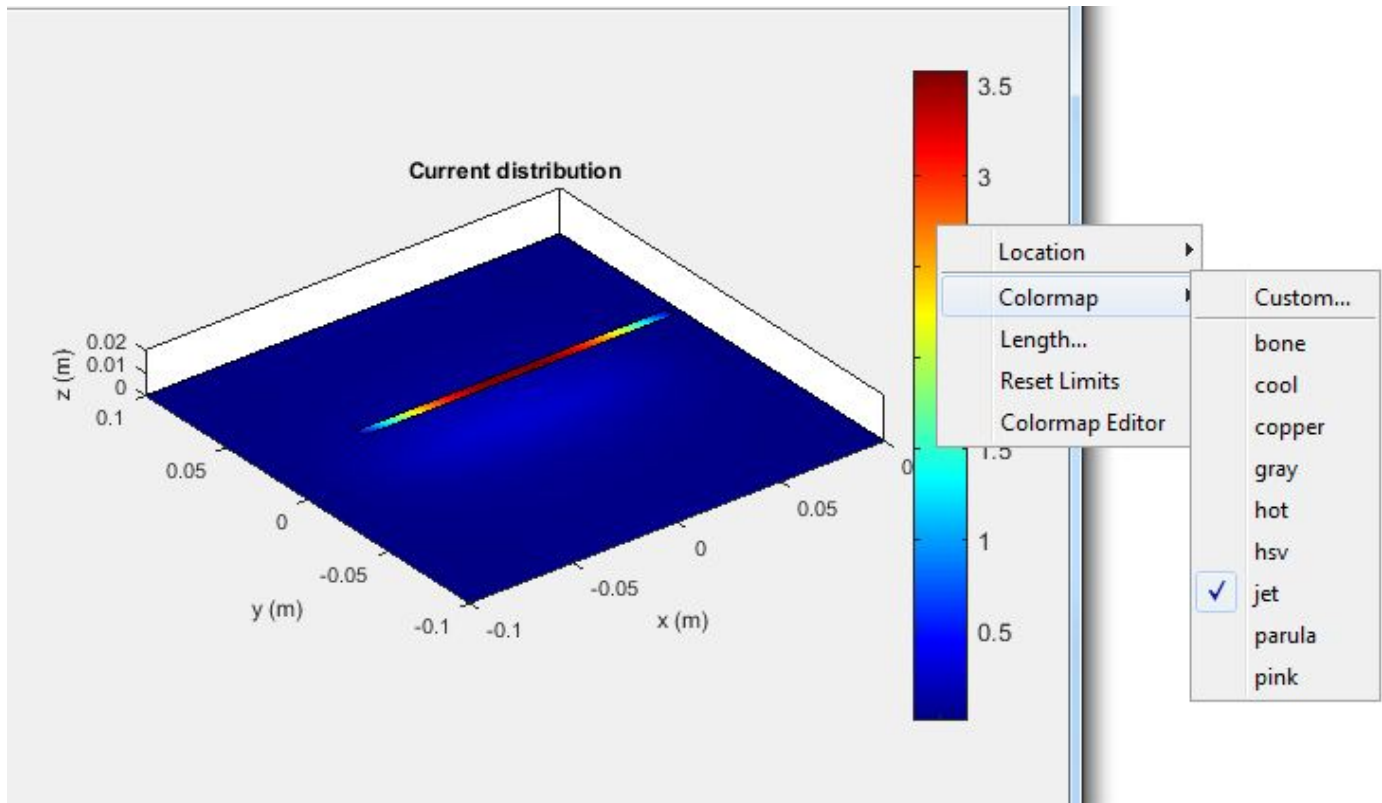
Currents are also induced on the reflector surface as shown in the figure above. To better visualize the currents on the reflector surface, hover the mouse on the colorbar and change the dynamic range (scale) of the current plot as explained below.

Interaction with Colorbar

Hover the mouse over the colorbar. The mouse arrow is now transformed into a flat arrow. Pull the mouse up or down to change the dynamic range of the current. This helps us to better visualize the current distribution on the reflector surface as shown below. You can also pull the magnitude of the currents next to the colorbar to change the dynamic range.



Right click on the colorbar and select the Reset limit tab to change the current limits to the original values. There is also an option to change the colormap and move the colorbar to a different location as shown below.



References

- [1] C. A. Balanis, 'Antenna Theory. Analysis and Design,' Wiley, New York, 3rd Edition, 2005.
- [2] S. N. Makarov, 'Antenna and EM Modeling with MATLAB,' p.66, Wiley & Sons, 2002.

See Also

"Verification of Far-Field Array Pattern Using Superposition with Embedded Element Patterns" on page 5-372 | "Comparison of Antenna Array Transmit and Receive Manifold" on page 5-364

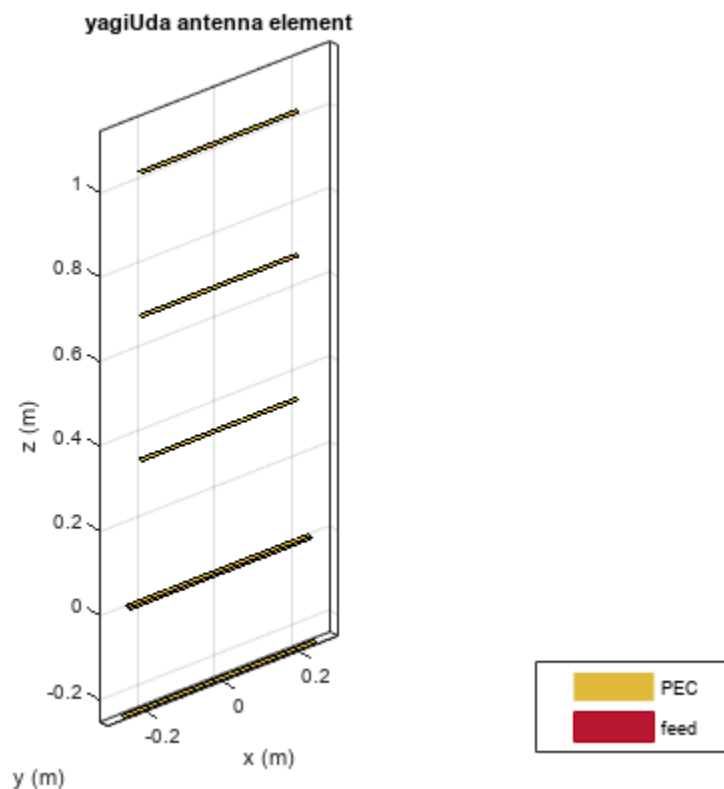
Antenna Far-Field Visualization

This example shows how to visualize the far fields for antennas and antenna arrays. Radiation patterns can be plotted in Antenna Toolbox™ using the `pattern` function. The example explains different options available in the `pattern` function.

Create Yagi-Uda Antenna

Create the Yagi-Uda antenna in the default configuration.

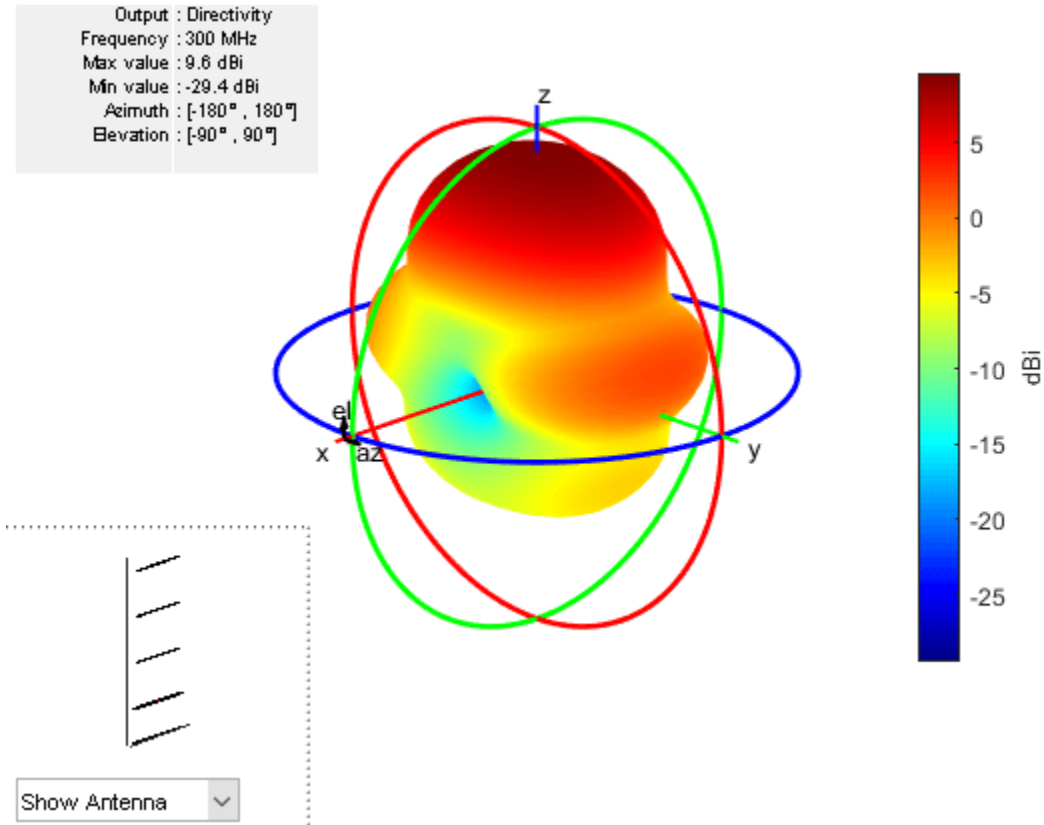
```
ant = yagiUda;  
show(ant);
```



Plot Radiation Pattern

To plot the radiation pattern of the antenna, specify the frequency at which the data needs to be plotted. If no angles are provided, the azimuth spacing of 5 degrees between -180 to 180 degrees and the elevation spacing of 5 degrees from -90 to 90 degrees, respectively, are assumed. By default, the 3D antenna directivity is plotted expressed in decibels.

```
freq = 300e6;  
pattern(ant, freq);
```



The list at top left corner of the figure window provides additional details such as minimum and maximum values of the quantity to be plotted as well as the range of angles where the data is plotted. In the lower left corner, there is the inset of the antenna. Rotating the radiation pattern rotates the antenna too. To remove the antenna image from the plot, unselect the Show Antenna button.

Animate the Field in the E-plane

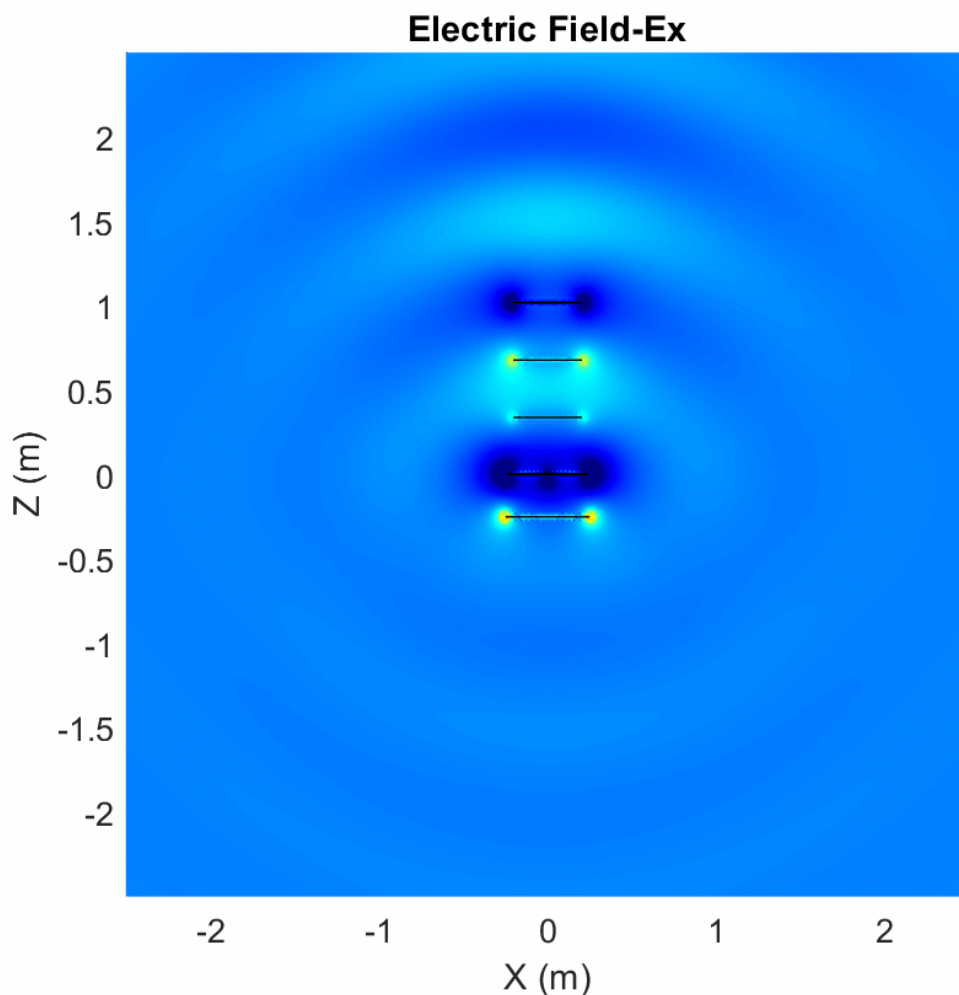
The analysis from the solver provides us with the steady state result. The field calculation provides the 3 components of the electric and magnetic field in phasor form in the cartesian coordinate system. Animate this field by using a propagator defined by the frequency, `freq`, to bring the result back to time-harmonic form and visualize on the E-plane and H-plane. In the case of the horn, E-plane is the XZ plane and H-plane is the XY plane. As noted in the far-field pattern plot shown in the previous section, the main lobe is along the positive x-axis. Use the `helperAnimateEHFields` function as shown below. Define the spatial extent, point density as well as the total number of time periods defined in terms of the frequency `freq`. Choose the `Ez` component with the plotting plane defined as XZ for the E-field. If the sampling time is not defined, the function automatically chooses a sampling time based on 10 times the center frequency. There is also an option available to save the animation output to a gif file with an appropriate name specified.

```
% Define Spatial Plane Extent
lambda = physconst('lightspeed')/freq;
SpatialInfo.XSpan = 5*lambda;
SpatialInfo.YSpan = 5*lambda;
SpatialInfo.ZSpan = 5*lambda;
SpatialInfo.NumPointsX = 400;
SpatialInfo.NumPointsY = 400;
```

```

SpatialInfo.NumPointsZ = 400;
% Define Time Extent
Tperiod = 1/freq;
T = 2*Tperiod;
TimeInfo.TotalTime = T;
TimeInfo.SamplingTime = [];
% Define E-Field Data details
DataInfo.Component = 'Ex';
DataInfo.PlotPlane = 'XZ';
DataInfo.DataLimits = [-2 6];
DataInfo.ScaleFactor = 1;
DataInfo.GifFileName = 'yagiEz.gif';
DataInfo.CloseFigure = true;
%% Animate
helperAnimateEHFields(ant,freq,SpatialInfo, TimeInfo, DataInfo)

```

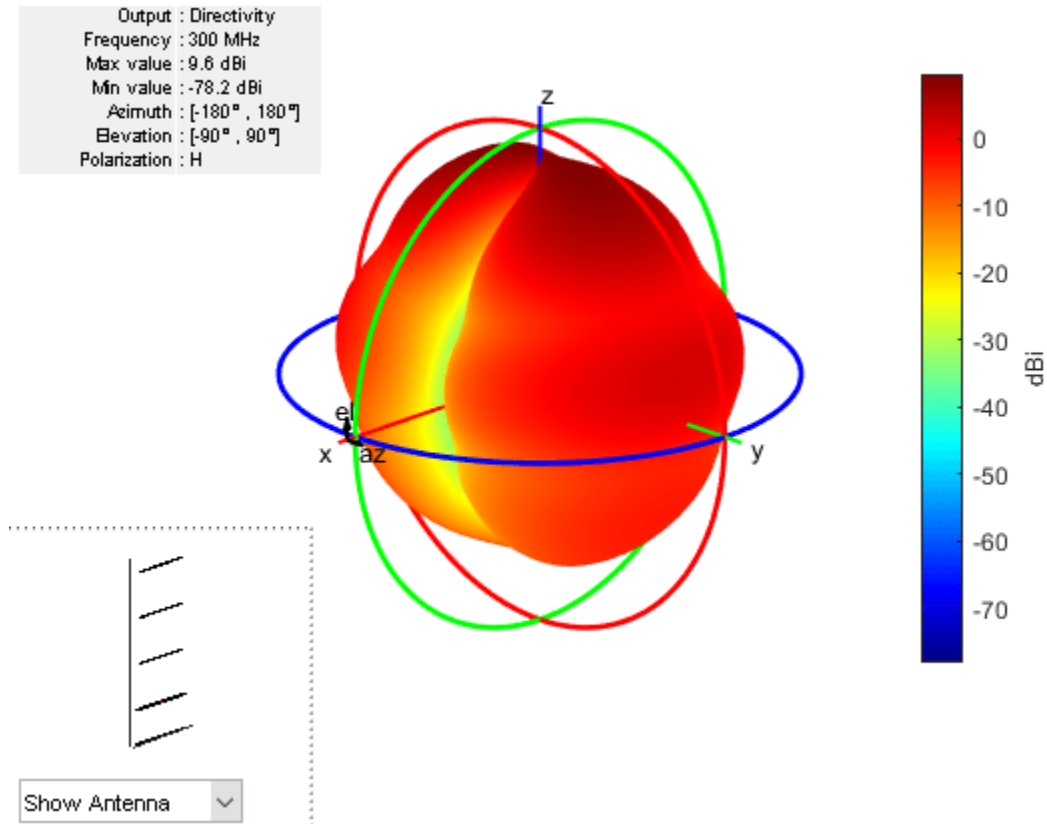


Plot Individual Polarization Components

By default, the function pattern plots the overall directivity. To plot the directivity of the individual field components separately, use the Polarization flag. The options available are the directivity of the azimuthal electric field (H), the directivity of the elevation electric field (V), Right-handed circular

polarization (RHCP) and Left Hand circular polarization (LHCP) components of the electric field. The plot below shows the directivity of the azimuthal component of the electric field.

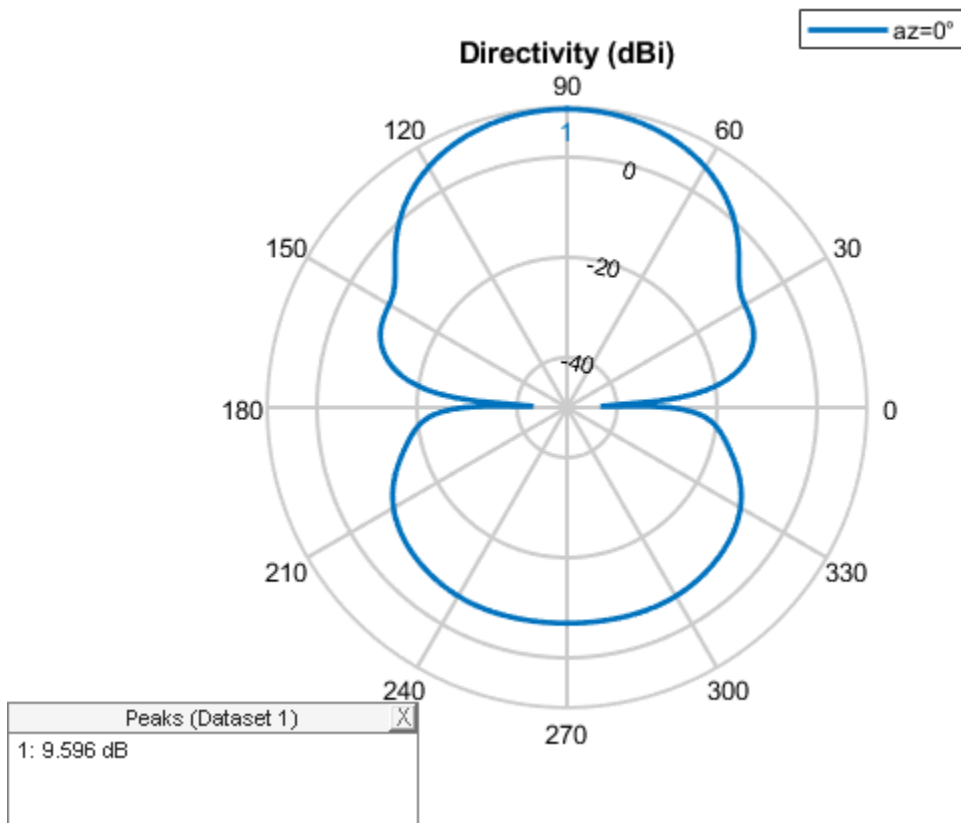
```
pattern(ant, freq, 'Polarization', 'H');
```



Plot 2D Patterns

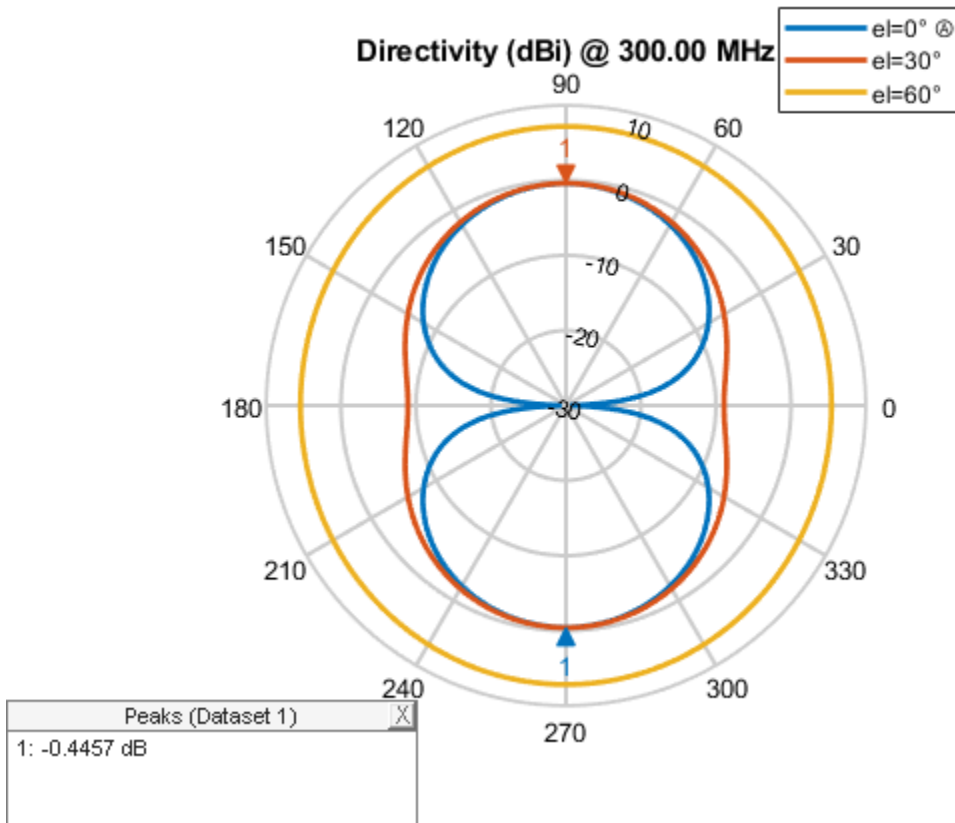
Individual slices of the directivity can be plotted using a polar or rectangular plot, to give a better understanding of how the field varies in different planes. The plot below shows the total directivity in the elevation plane (a 2D radiation pattern).

```
pattern(ant, freq, 0, 0:1:360);
```



Use the `patternAzimuth` multiple patterns and overlay them on the single plot. The plot below shows three radiation patterns at azimuthal angles of 0, 30 and 60 degrees.

```
patternAzimuth(ant, freq, [0 30 60]);
```

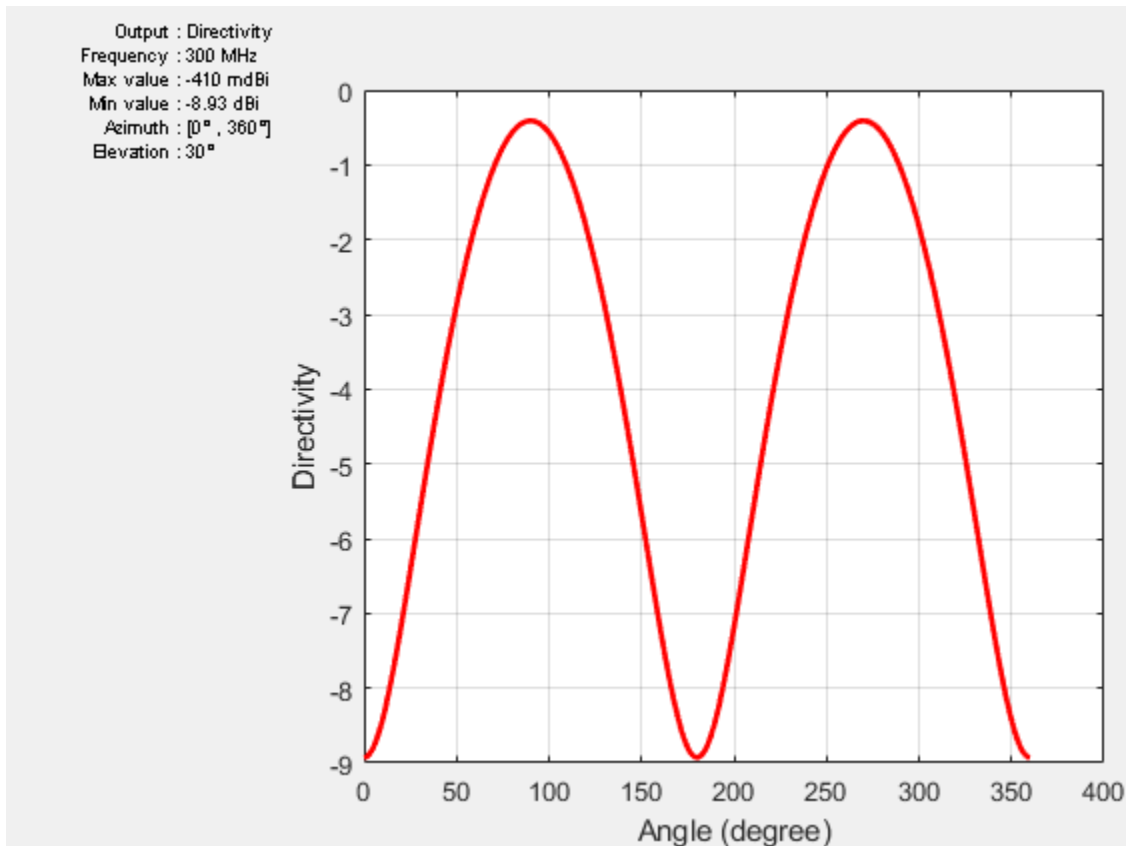


To plot the patterns at fixed elevation angles, use the function `patternElevation`.

Directivity on a Rectangular Plot

The `pattern` function also allows us to visualize the patterns using a rectangular plot. This can be done by modifying the `CoordinateSystem` flag as shown below. By default, the flag is set to `polar`. Change it to `rectangular` to visualize the data in the rectangular coordinate system.

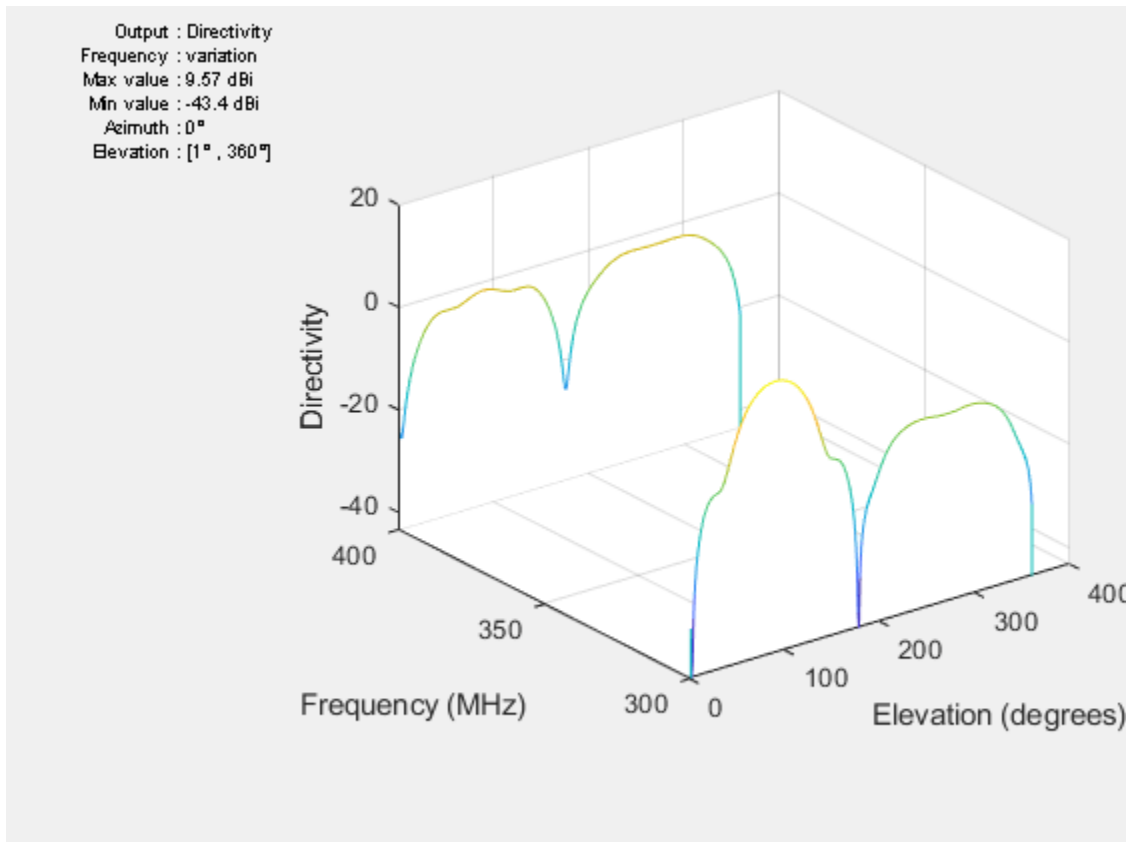
```
pattern(ant, freq, 0:1:360, 30, 'CoordinateSystem', 'rectangular');
```



Plot Results at Multiple Frequencies

To plot patterns at multiple frequencies on the same graph, use the function `pattern` with the `PlotStyle` flag allowing a choice between `Overlay` or `Waterfall` style of plotting. The feature is important for the analysis of large-bandwidth antennas. The `PlotStyle` flag needs to be set to `waterfall` to display a waterfall plot. The default value of this flag is `overlay`.

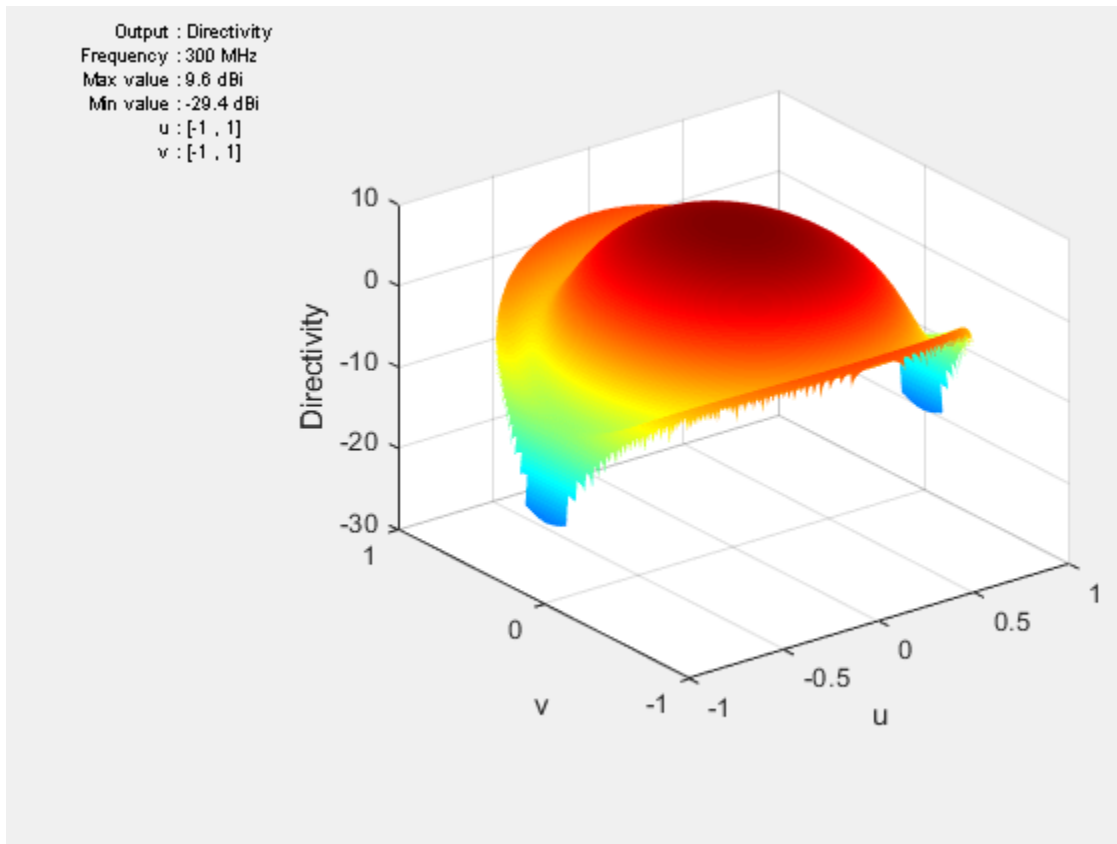
```
pattern(ant, [300e6 400e6], 0, 1:1:360, 'PlotStyle', 'waterfall', ...  
       'CoordinateSystem', 'rectangular');
```



U-V Plot

The third option for the `CoordinateSystem` flag is `uv`. This will plot the field data in the `u-v` coordinate system. The functionality is mostly useful for antenna arrays to visualize the array lobes.

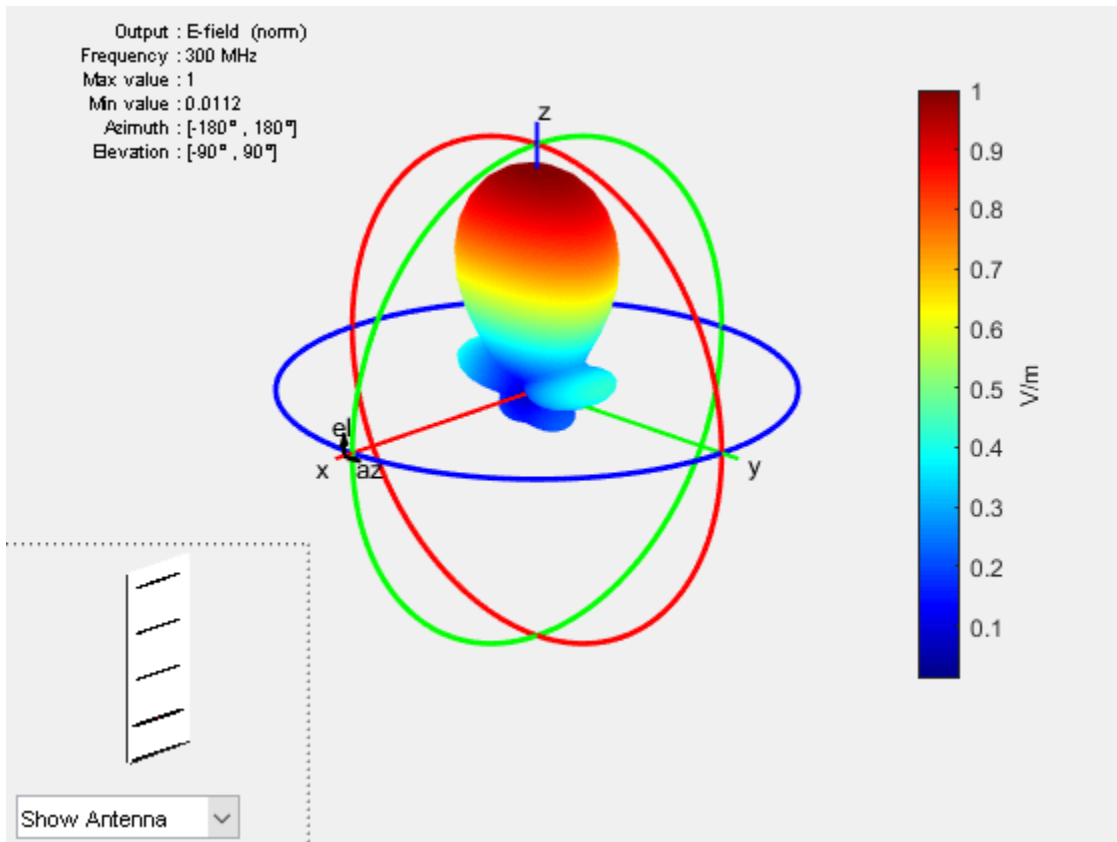
```
pattern(ant, freq, 'CoordinateSystem', 'uv');
```



Plot Electric Field and Power

The pattern function also allows us to plot the linear field patterns. Different field quantities can be plotted by modifying the Type flag. By default it is set to directivity. It can be set to efield to plot the magnitude of the electric field. The plot below shows the normalized magnitude of the electric field.

```
pattern(ant, freq, 'Type', 'efield', 'Normalize', true);
```



Beamwidth

The main beam is the region of the maximum radiation. In the present case, the main beam is along the z-axis. The half-power beamwidth (HPBW) is the angular separation in which the magnitude of the radiation pattern decreases by 50% (or -3dB) from the peak of the main beam. For example, we want to measure the width of the main beam in the elevation plane. The HPBW can be calculated as shown below

```
[bw, angles] = beamwidth(ant, freq, 0, 1:1:360)
```

```
bw = 52.0000
```

```
angles = 1x2
```

```
64 116
```

The beamwidth is calculated at a single frequency and in a specified plane by choosing the respective values of azimuth or elevation angles. We observe that the width of the main beam is 52 degrees; the beam is located between 64 degrees and 116 degrees in the elevation plane.

References

[1] C. A. Balanis, 'Antenna Theory. Analysis and Design,' p. 514, Wiley, New York, 3rd Edition, 2005.

See Also

"Antenna Near-Field Visualization" on page 5-27 | "Plane Wave Excitation - Scattering Solution" on page 5-353

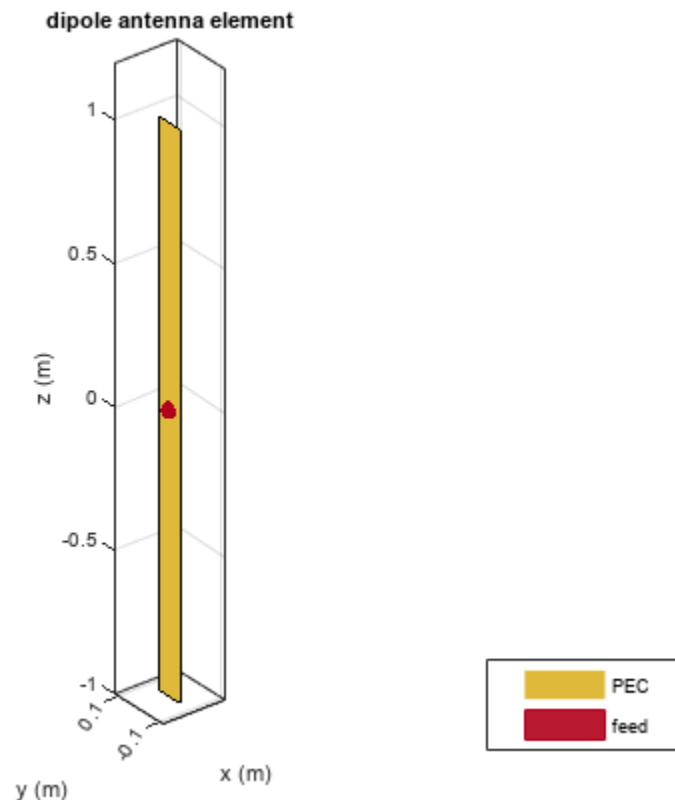
Antenna Near-Field Visualization

This example shows how to calculate and visualize the near-fields for antennas. Near fields can be plotted in Antenna Toolbox™ using the `EHfields` function. This function can also be used to calculate and plot the near-fields for antenna arrays. In the near field region, the electric and magnetic fields can exist independently of each other, and one type of field can dominate the other.

Create Dipole Antenna

Create the dipole antenna in the default configuration.

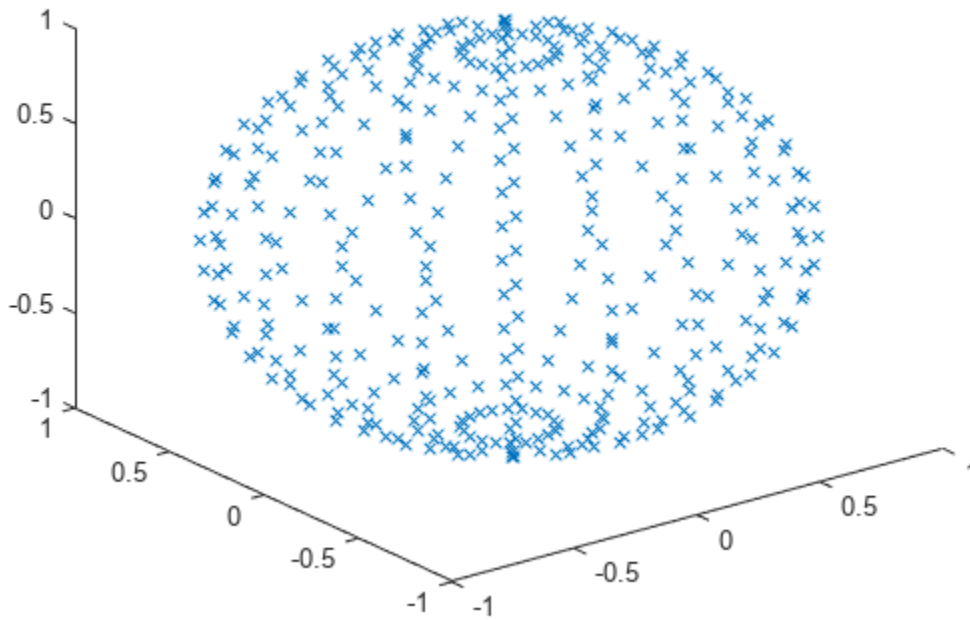
```
ant = dipole;
show(ant);
```



Define Points to Calculate Near-Field

The near fields as the name suggest are fields that are calculated in the close proximity of the antenna. In this example, we plot the fields over a sphere of diameter 2 meters and centered at the origin. The dipole is also of length two meters and centered at the origin. So the sphere completely encloses the dipole. The sphere command generates 21 x 21 matrices for X, Y and Z. They are combined to generate 441 points as shown below.

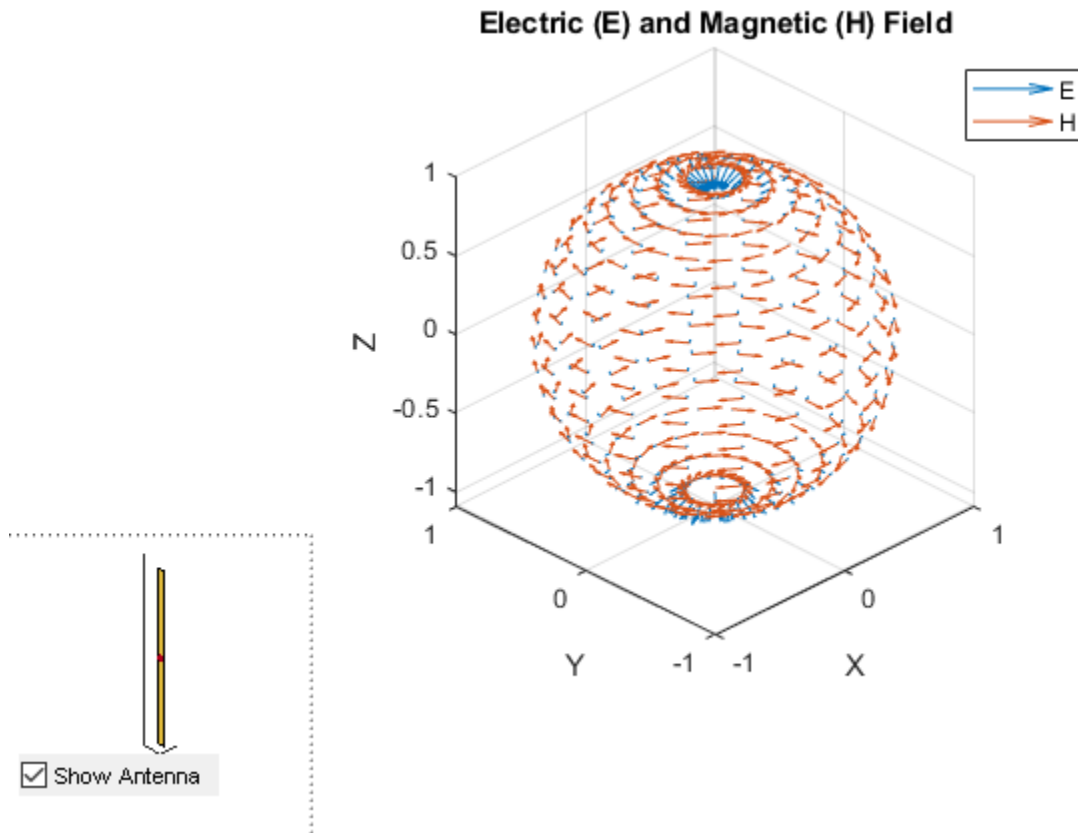
```
[X, Y, Z] = sphere(20);
Points = [X(:), Y(:), Z(:)].';
plot3(Points(1,:), Points(2,:), Points(3,:), 'x');
```



Plot Near-Fields

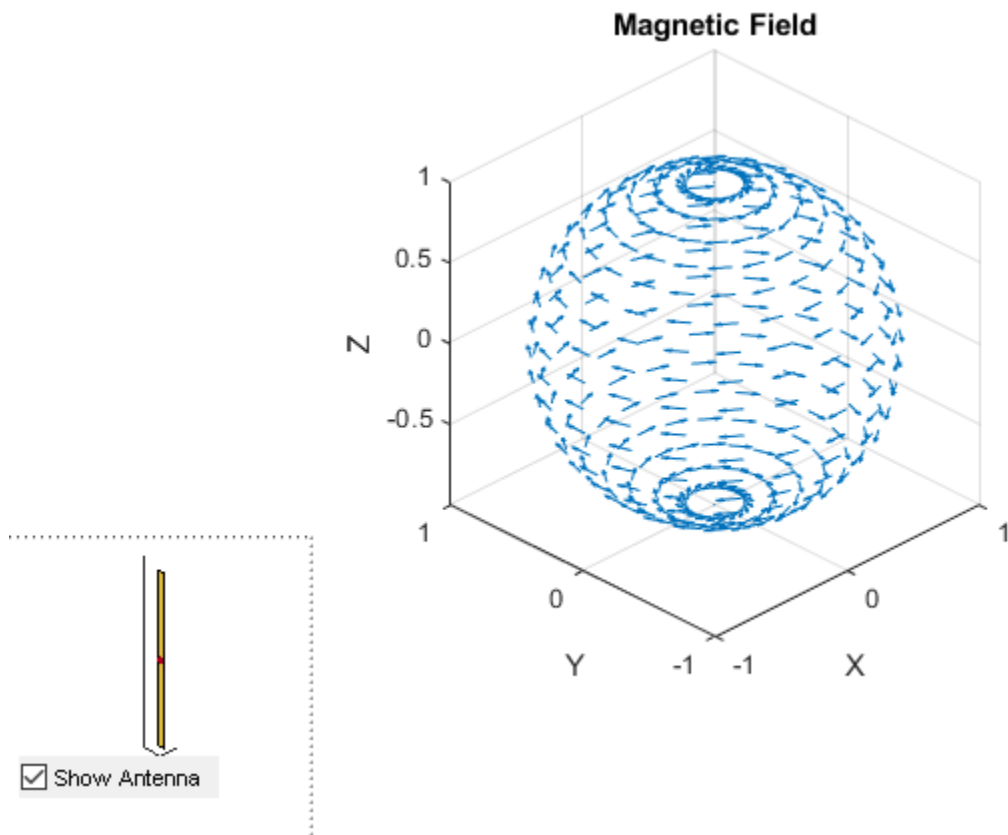
The near-fields at the various points calculated above can be plotted by using the `EHfields` function as shown. By default both the electric and magnetic fields are plotted.

```
EHfields(ant, 70e6, Points);
```



Individual fields can be plotted by using the `ViewField` flag. Below we plot only the magnetic field. As expected, the direction of the magnetic field follows the right hand rule due to the current along the length of the dipole.

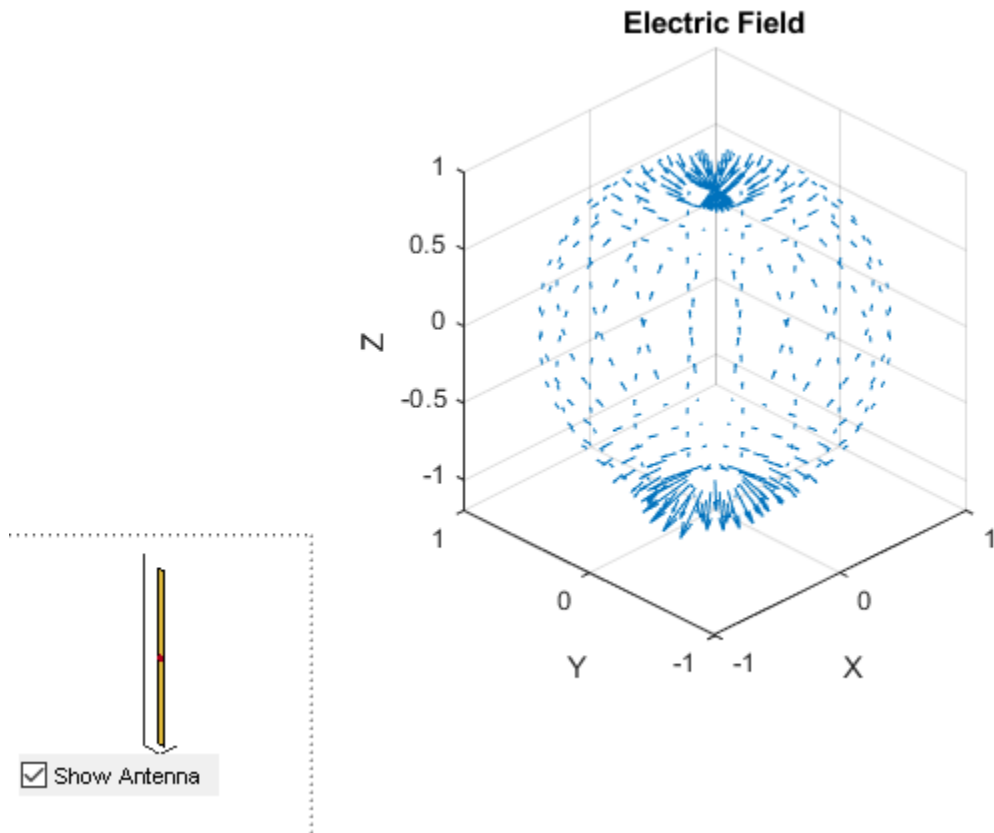
```
EHfields(ant, 70e6, Points, 'ViewField', 'H');
```



Plot Electric Field

The plot below shows the electric field distribution over the points. As expected, we have electric field lines coming out of the positive charge and into the negative charge. The electric field forms loops along the length of the dipole. The `ScaleFields` flag is used to scale the size of the field vector. In this case, the electric field vector is multiplied by 2, to artificially increase the size of the vectors so as to make them clearly visible.

```
EHfields(ant, 70e6, Points, 'ViewField', 'E', 'ScaleFields', [2,0]);
```



Output Near-Fields to Workspace

By providing RHS, the electric and magnetic fields can be outputted to the workspace to perform any other mathematical computations.

```
[E, H] = EHfields(ant, 70e6, Points);
```

Just to clarify, the EHfields function, calculates the E-H fields at any point in space. It can be used to calculate the near or the far fields.

See Also

“Modeling Resonant Coupled Wireless Power Transfer System” on page 5-209 | “Wave Impedance” on page 5-32

Wave Impedance

This example will use an elementary dipole and loop antenna and analyze the wave impedance behavior of each radiator in space at a single frequency. The region of space around an antenna has been defined in a variety of ways. The most succinct description is using a 2-or 3-region model. One variation of the 2-region model uses the terms near-field and far-field to identify specific field mechanisms that are dominant. The 3-region model, splits the near-field into a transition zone, wherein a weakly radiative mechanism is at work. Other terms that have been used to describe these zones, include, quasistatic field, reactive field, non-radiative field, Fresnel region, induction zone etc. [1]. Pinning these regions down mathematically presents further challenges as observed with the variety of definitions available across different sources [1]. Understanding the regions around an antenna is critical for both an antenna engineer as well as an electromagnetic compatibility(EMC) engineer. The antenna engineer may want to perform near-field measurements and then compute the far-field pattern. To the EMC engineer, understanding the wave impedance is required for designing a shield with a particular impedance to keep interference out.

Create Electrically Short Dipole and Small Loop Antenna

For this analysis the frequency is 1 GHz. The length and circumference of the dipole and loop are selected so that they are electrically short at this frequency.

```
f = 1e9;
c = physconst('lightspeed');
lambda = c/f;
wavenumber = 2*pi/lambda;
d = dipole;
d.Length = lambda/20;
d.Width = lambda/400;
circumference = lambda/20;
r = circumference/(2*pi);
l = loopCircular;
l.Radius = r;
l.Thickness = circumference/200;
```

Create Nodal Points in Space to Calculate Fields

The wave impedance is defined in a broad sense as the ratio of the magnitudes of the total electric and magnetic field, respectively. The magnitude of a complex vector is defined to be the length of the real vector resulting from taking the modulus of each component of the original complex vector. To examine impedance behavior in space, choose a direction and vary the radial distance R from the antenna along this direction. The spherical coordinate system is used with azimuth and elevation angles fixed at (0,0) while R is varied in terms of wavelength. For selected antennas, the maximum radiation occurs in the azimuthal plane. The smallest value of R has to be greater than the structure dimensions, i.e. field computations are not done directly on the surface.

```
N = 1001;
az = zeros(1,N);
el = zeros(1,N);
R = linspace(0.1*lambda,10*lambda,N);
x = R.*sind(90-el).*cosd(az);
y = R.*sind(90-el).*sind(az);
z = R.*cosd(90-el);
points = [x;y;z];
```

Calculate Electric and Magnetic Fields at Nodal Points

Since the antennas are electrically small at the frequency of 1 GHz, mesh the structure manually by specifying a maximum edge length. The surface mesh is a triangular discretization of the antenna geometry. Compute electric and magnetic field complex vectors.

```
md = mesh(d, 'MaxEdgeLength', 0.0003);
ml = mesh(l, 'MaxEdgeLength', 0.0003);

[Ed, Hd] = EHfields(d, f, points);
[El, Hl] = EHfields(l, f, points);
```

Total Electric and Magnetic Fields

The electric and magnetic field results from function `EHfields` is a 3-component complex vector. Calculate the resulting magnitudes of the electric and magnetic field component wise, respectively

```
Edmag = abs(Ed);
Hdmag = abs(Hd);

Elmag = abs(El);
Hlmag = abs(Hl);

% Calculate resultant E and H
Ed_rt = sqrt(Edmag(1,:).^2 + Edmag(2,:).^2 + Edmag(3,:).^2);
Hd_rt = sqrt(Hdmag(1,:).^2 + Hdmag(2,:).^2 + Hdmag(3,:).^2);

El_rt = sqrt(Elmag(1,:).^2 + Elmag(2,:).^2 + Elmag(3,:).^2);
Hl_rt = sqrt(Hlmag(1,:).^2 + Hlmag(2,:).^2 + Hlmag(3,:).^2);
```

Calculate Wave Impedance

The wave impedance can now be calculated at each of the predefined points in space as the ratio of the total electric field magnitude to the total magnetic field magnitude. Calculate this ratio for both the dipole antenna and the loop antenna.

```
ZE = Ed_rt./Hd_rt;
ZH = El_rt./Hl_rt;
```

Impedance of Free Space

The material properties of free space, the permittivity and permeability of vacuum, are used to define the free space impedance η .

```
eps_0 = 8.854187817e-12;
mu_0 = 1.2566370614e-6;
eta = round(sqrt(mu_0/eps_0));
```

Plot Wave Impedance as a Function of Distance

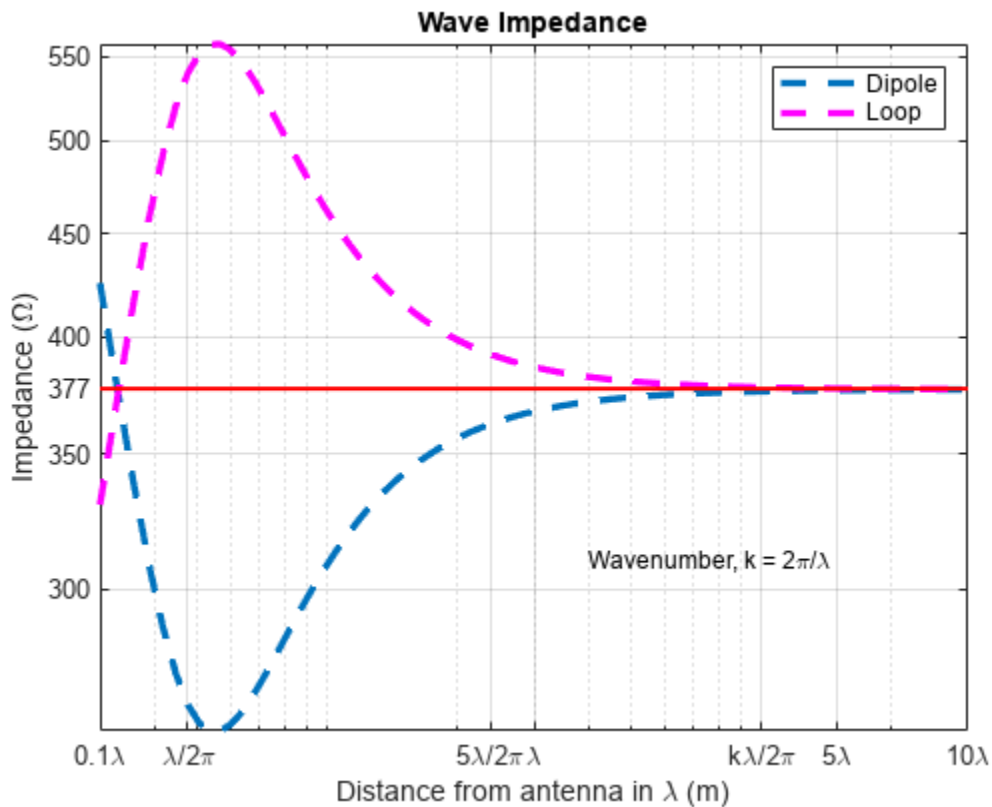
The behavior of wave impedance for both antennas is given on the same plot. The x-axis is the distance from the antenna in terms of λ and the y-axis is the impedance measured in Ω .

```
fig1 = figure;
loglog(R, ZE, '--', 'LineWidth', 2.5)
hold on
loglog(R, ZH, 'm--', 'LineWidth', 2.5)
line(R, eta.*ones(size(ZE)), 'Color', 'r', 'LineWidth', 1.5);
```

```

textInfo = 'Wavenumber, k = 2\pi/\lambda';
text(0.4,310,textInfo,'FontSize',9)
ax1 = fig1.CurrentAxes;
ax1.XTickLabelMode = 'manual';
ax1.XLim = [min(R) max(R)];
ax1.XTick = sort([lambda/(2*pi) 5*lambda/(2*pi) lambda 1 5*lambda ax1.XLim]);
ax1.XTickLabel = {'0.1\lambda'; '\lambda/2\pi'; '5\lambda/2\pi'; '\lambda'; 'k\lambda/2\pi'; '5\lambda'};
ax1.YTickLabelMode = 'manual';
ax1.YTick = sort([ax1.YTick eta]);
ax1.YTickLabel = cellstr(num2str(ax1.YTick));
xlabel('Distance from antenna in \lambda (m)')
ylabel('Impedance (\Omega)')
legend('Dipole','Loop')
title('Wave Impedance')
grid on

```



Discussion

The plot of the wave impedance variation reveals several interesting aspects.

- The wave impedance changes with distance from the antenna and shows opposing behaviors in the case of the dipole and the loop. The dipole, whose dominant radiation mechanism is via the electric field, shows a minima close to the radian sphere distance, $\lambda/2\pi$, whilst the loop which can be thought of as a magnetic dipole shows a maxima in the impedance.
- The region below the radian sphere distance, shows the first cross-over across 377Ω . This cross over occurs very close to the structure, and the quick divergence indicates that we are in the reactive near-field.

- Beyond the radian sphere distance ($\lambda/2\pi$), the wave impedance for the dipole and loop decreases and increases respectively. The impedance starts to converge towards the free space impedance value of $\eta = 377\Omega$.
- Even at a distance of $5 \lambda/2\pi - 1\lambda$ from the antennas, the wave impedance has not converged, implying that we are not yet in the far-field.
- At a distance of $k\lambda/2\pi = 1$ and beyond, the values for wave impedance are very nearly equal to 377Ω . Beyond 10λ , the wave impedance stabilizes and the region of space can be termed as the far-field for these antennas at the frequency of 1GHz.
- Note that the dependence on wavelength implies that these regions we have identified will change if the frequency is changed. Thus, the boundary will move in space.

Reference

[1] C. Capps, "Near Field or Far Field," EDN, August 16, 2001, pp. 95-102. Online at: <http://m.eet.com/media/1140931/19213-150828.pdf>

See Also

"3D Reconstruction of Radiation Pattern From 2D Orthogonal Slices" on page 5-497

Antenna Array Analysis

This example shows how to create and analyze antenna arrays in Antenna Toolbox™, with emphasis on concepts such as beam scanning, sidelobe level, mutual coupling, element patterns, and grating lobes. The analysis is on a 9-element linear array of half-wavelength dipoles

Design Frequency and Array parameters

Choose the design frequency to be 1.8 GHz, which happens to be one of the carrier frequencies for 3G/4G cellular systems. Define array size using number of elements, N and inter-element spacing, dx.

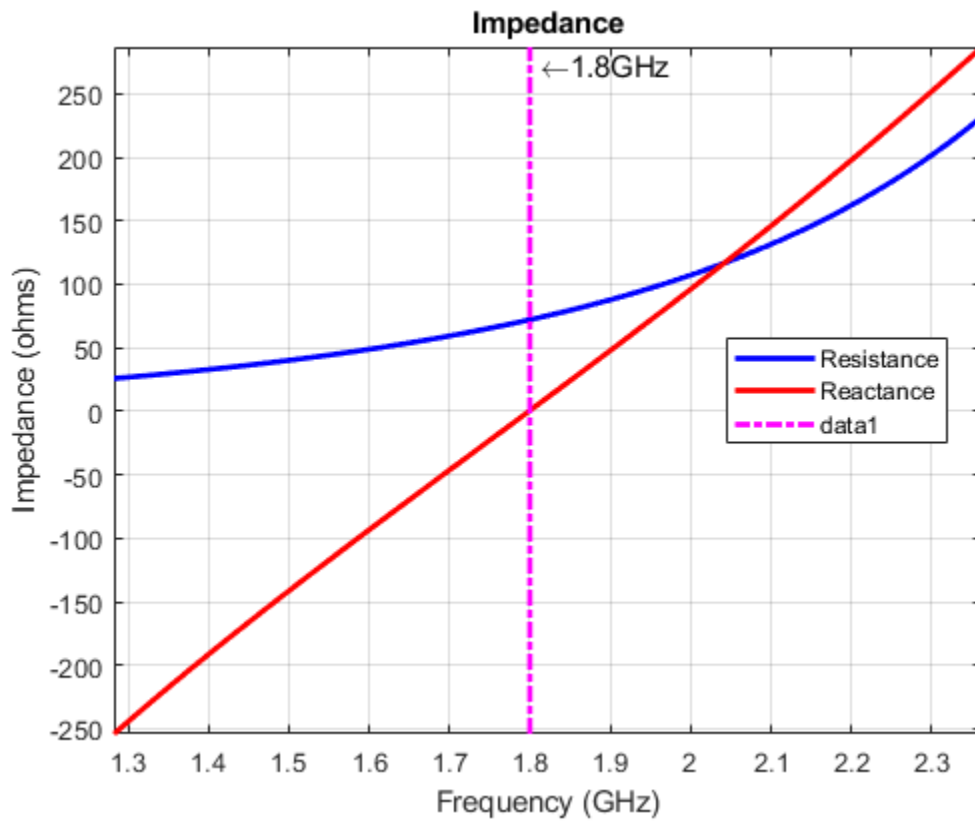
```
freq = 1.8e9;
c = physconst('lightspeed');
lambda = c/freq;
N = 9;
dx = 0.49*lambda;
```

Create Resonant Dipole

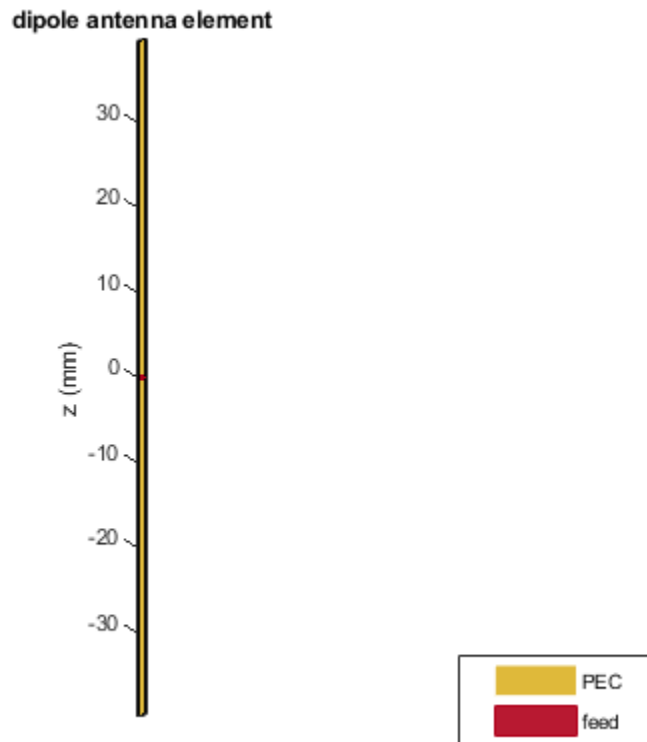
The individual element of the array is a dipole. The initial length of this dipole is chosen to be $\lambda/2$. Trim its length so as to achieve resonance ($X \sim 0 \Omega$).

```
dipole_L = lambda/2;
dipole_W = lambda/200;
mydipole = dipole;
mydipole.Length = dipole_L;
mydipole.Width = dipole_W;
mydipole.TiltAxis = 'Z';
mydipole.Tilt = 90;
fmin = freq - .05*freq;
fmax = freq + .05*freq;
minX = 0.0001;           % Minimum value of reactance to achieve
trim = 0.0005;          % The amount to shorten the length at each iteration
resonant_dipole = dipole_tuner(mydipole,freq,fmin,fmax,minX,trim);
Z_resonant_dipole = impedance(resonant_dipole,freq)
```

```
Z_resonant_dipole =
    71.8473 - 0.3583i
```



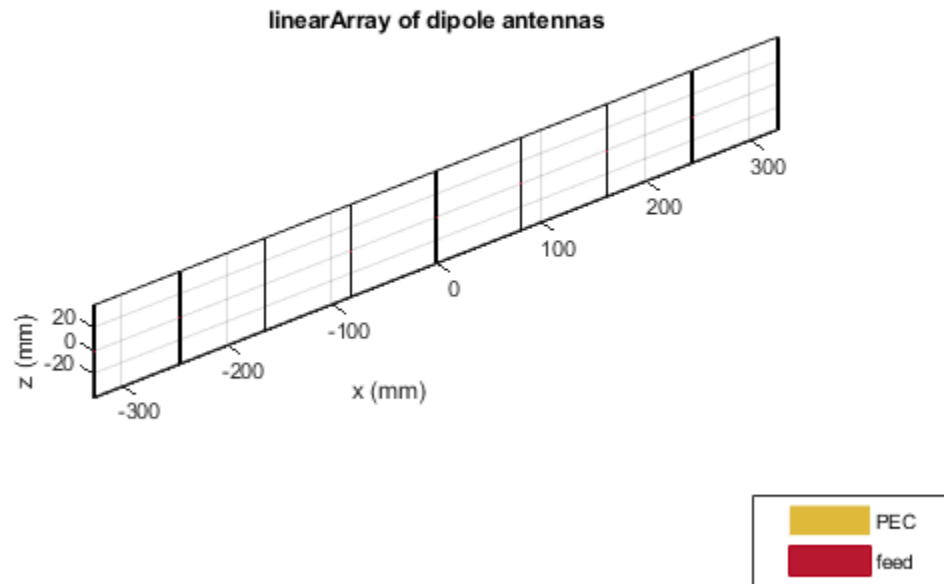
```
helement = figure;  
show(resonant_dipole)  
axis tight
```



Create Linear Array

Assign the resonant dipole as the individual radiator of a linear array. While the isolated dipole has been tuned for resonance at the design frequency, it will get detuned in the array environment. Modify the number of elements and spacing and observe the array geometry. The elements are positioned on the x-axis and are numbered from left to right.

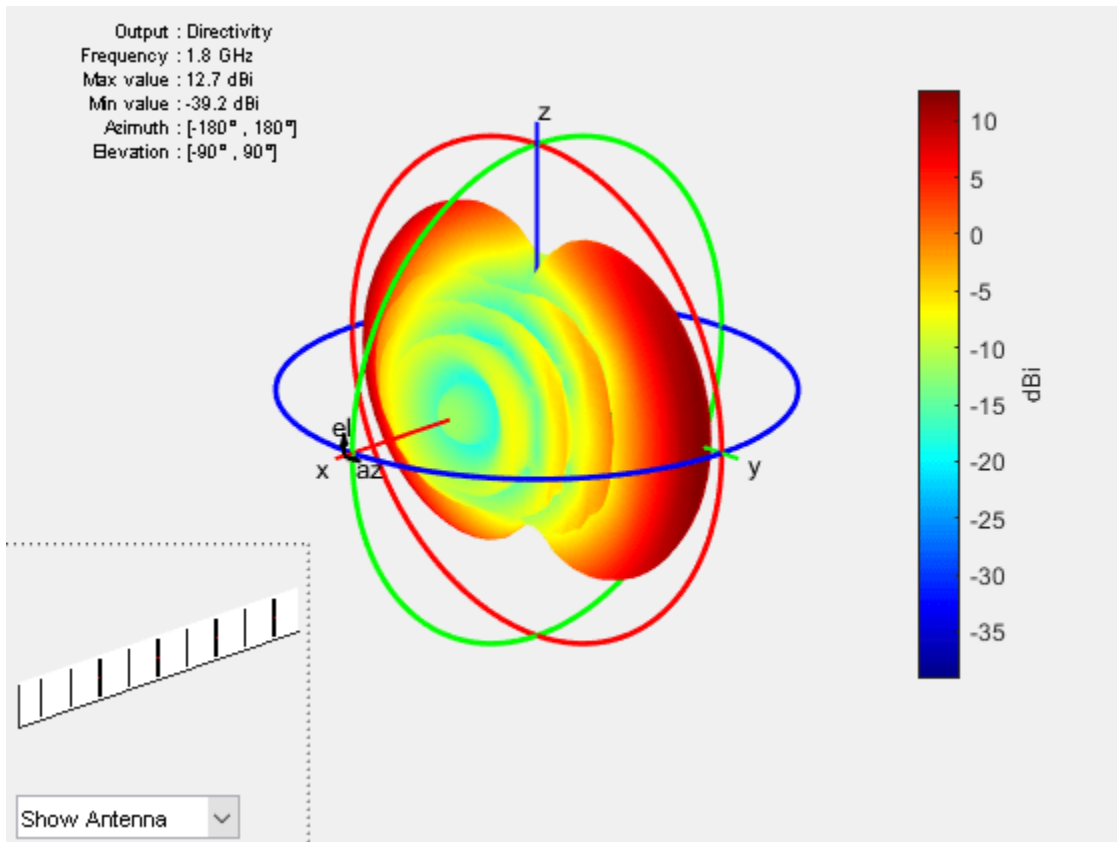
```
dipole_array = linearArray;  
dipole_array.Element = resonant_dipole;  
dipole_array.NumElements = N;  
dipole_array.ElementSpacing = dx;  
hArray = figure;  
show(dipole_array)  
axis tight
```



Plot 3D Array Pattern

Visualize the pattern for the linear array in 3D space at the design frequency.

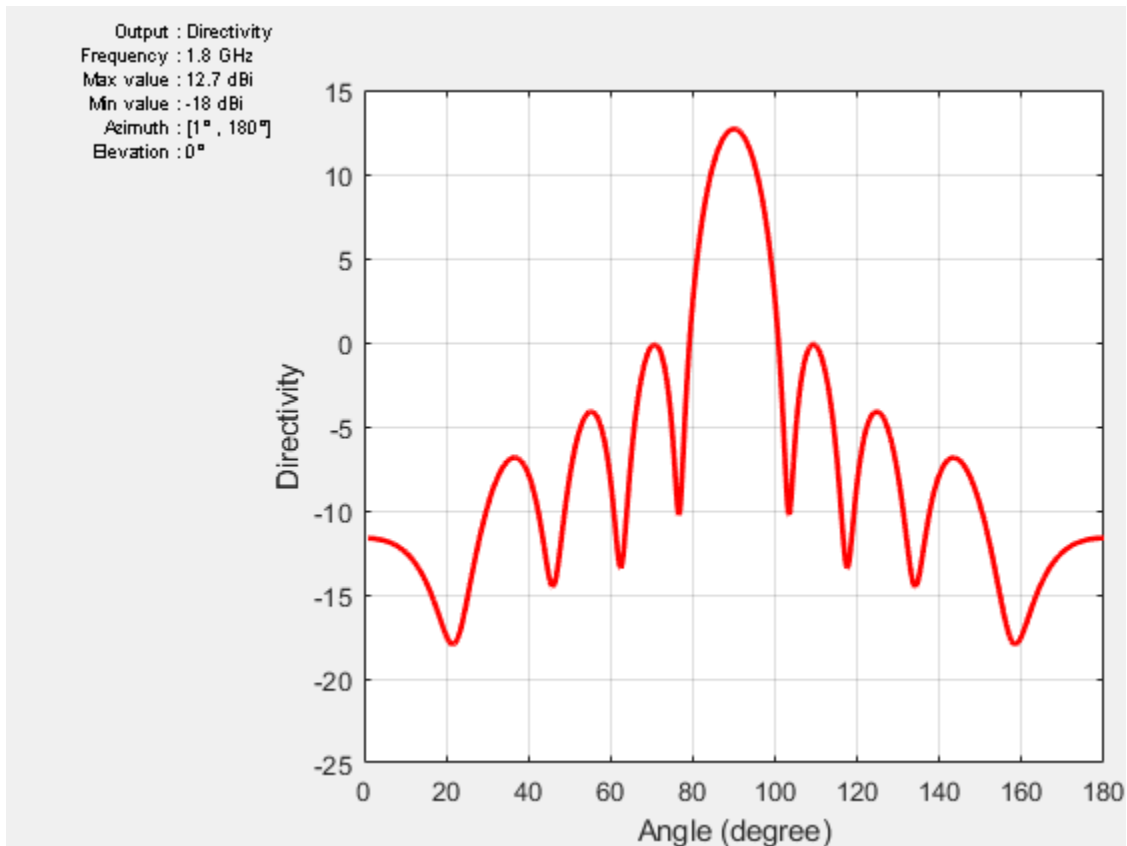
```
pattern3Dfig = figure;  
pattern(dipole_array, freq)
```



Plot 2D Radiation Pattern

The 3D pattern of the array shows the maximum of the beam at an azimuthal angle of 90 deg. Plot the 2D radiation pattern in the azimuthal plane(x-y plane) which corresponds to zero elevation angle.

```
patternazfig1 = figure;
az_angle = 1:0.25:180;
pattern(dipole_array,freq,az_angle,0,'CoordinateSystem','rectangular')
axis([0 180 -25 15])
```

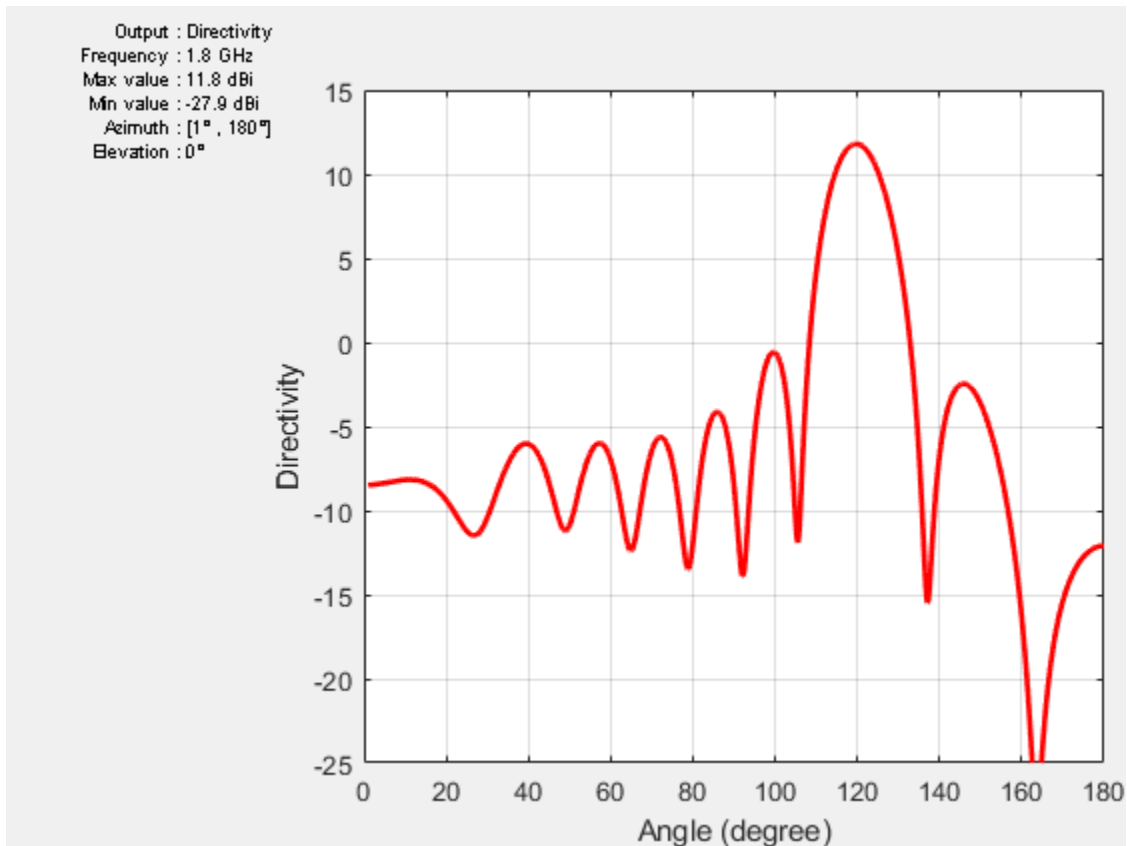


The array has a peak directivity of 12.83 dBi, and the first sidelobes on either side of the peak are approximately 13 dB down. This is because the array has a uniform amplitude taper with all elements fed at 1V. The sidelobe level can be controlled by using different amplitude tapers on the array elements such as Chebyshev and Taylor.

Beam Scanning

Choosing a set of phase shifts allows us to scan the beam to a specific angle. This linear array configuration enables scanning in the azimuthal plane (x-y plane), which corresponds to zero elevation angle. Scan the beam 30 deg off broadside (azimuthal angle of 120 deg).

```
scanangle = [120 0];
ps = phaseshift(freq,dx,scanangle,N);
dipole_array.PhaseShift = ps;
patternazfig2 = figure;
pattern(dipole_array,freq,az_angle,0,'CoordinateSystem','rectangular')
axis([0 180 -25 15])
```



The peak of the main beam is now 30 deg away from the initial peak (azimuth = 90 deg). Note the drop in directivity of about 0.9 dB. For infinite arrays, this drop increases with increasing the scan angle according to a cosine law.

Plot Element Patterns of Corner and Central Elements

In small arrays, the pattern of the individual element can vary significantly. In order to establish this fact, plot the pattern of the central elements and of the two edge elements. To obtain these patterns, excite every element alone and terminate the rest into a reference impedance. The elements are numbered from left to right, in the direction of the x-axis.

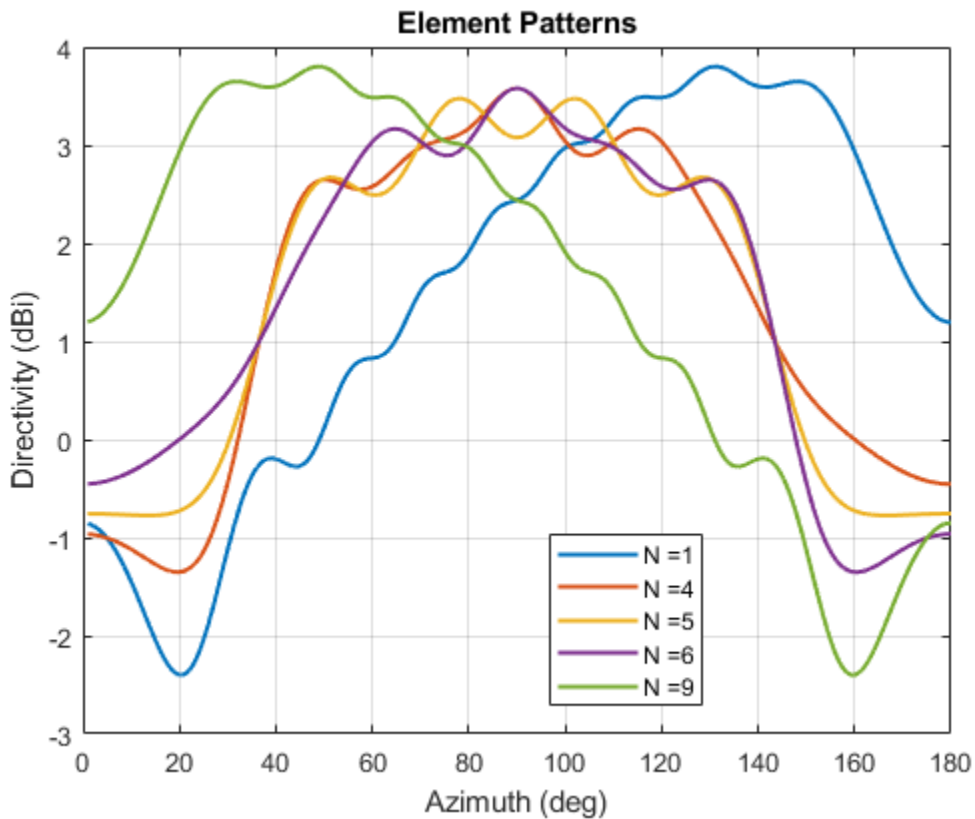
```

element_number = [1 ceil(N/2)-1 ceil(N/2) ceil(N/2)+1 N];
D_element = nan(numel(element_number),numel(az_angle));
legend_string = cell(1,numel(element_number));
for i = 1:numel(element_number)
    D_element(i,:) = pattern(dipole_array,freq,az_angle,0, ...
        'CoordinateSystem','rectangular', ...
        'ElementNumber',element_number(i), ...
        'Termination',real(Z_resonant_dipole));
    legend_string{i} = strcat('N = ',num2str(element_number(i)));
end
patternazfig3 = figure;
plot(az_angle,D_element,'LineWidth',1.5)
xlabel('Azimuth (deg)')
ylabel('Directivity (dBi)')
title('Element Patterns')

```



```
grid on
legend(legend_string, 'Location', 'best')
```



The plot of the element patterns reveals that, apart from the central element, all the others are mirror images about the center of the plot, i.e. the element pattern of the 1st element is a mirror reflection about azimuth = 90 deg of the element pattern of the 9th element, etc.

Mutual Coupling

Mutual coupling is the phenomenon whereby currents developed on each element in the array do not depend only on their own excitation, but have contributions from other elements as well. To study this effect, we will simplify the array to a 2-element case similar to [1].

```
dipole_array.NumElements = 2;
dipole_array.AmplitudeTaper = 1;
dipole_array.PhaseShift = 0;
```

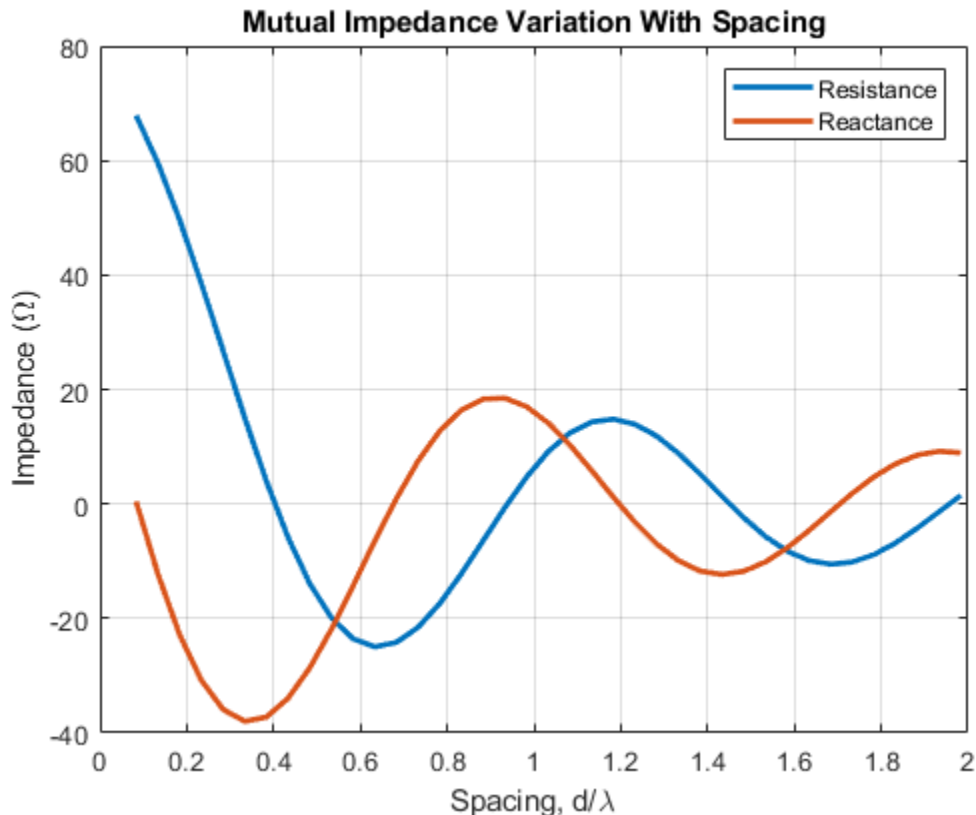
To observe the effect of mutual coupling, vary the spacing between the array elements and plot the variation in Z_{12} , mutual impedance between the pair of dipoles in the array [1]. Since the elements are parallel to each other, the coupling is strong.

```
spacing = (lambda/2:0.05:2).*lambda;
Z12 = nan(1,numel(spacing));
for i = 1:numel(spacing)
    dipole_array.ElementSpacing = spacing(i);
    s = sparameters(dipole_array,freq,real(Z_resonant_dipole));
    S = s.Parameters;
```

```

Z12(i) = 2*S(1,2)*70/((1 - S(1,1))*(1- S(2,2)) - S(1,2)*S(2,1));
end
mutualZ12fig = figure;
plot(spacing./lambda,real(Z12),spacing./lambda,imag(Z12),'LineWidth',2)
xlabel('Spacing, d/\lambda')
ylabel('Impedance (\Omega)')
grid on
title('Mutual Impedance Variation With Spacing')
legend('Resistance','Reactance')

```



Grating Lobes

Grating lobes are the maxima of the main beam predicted by the pattern multiplication theorem.

When the array spacing is less than or equal to $\lambda/2$, only the main lobe exists in the visible space with no other grating lobes. Grating lobes appear when the array spacing is greater than $\lambda/2$. For large spacing, grating lobe(s) could appear in the visible space even at zero scan angle. Investigate grating lobes for a linear array of 9 dipoles. Scan the beam 0 deg off broadside (azimuthal angle of 90 deg).

```

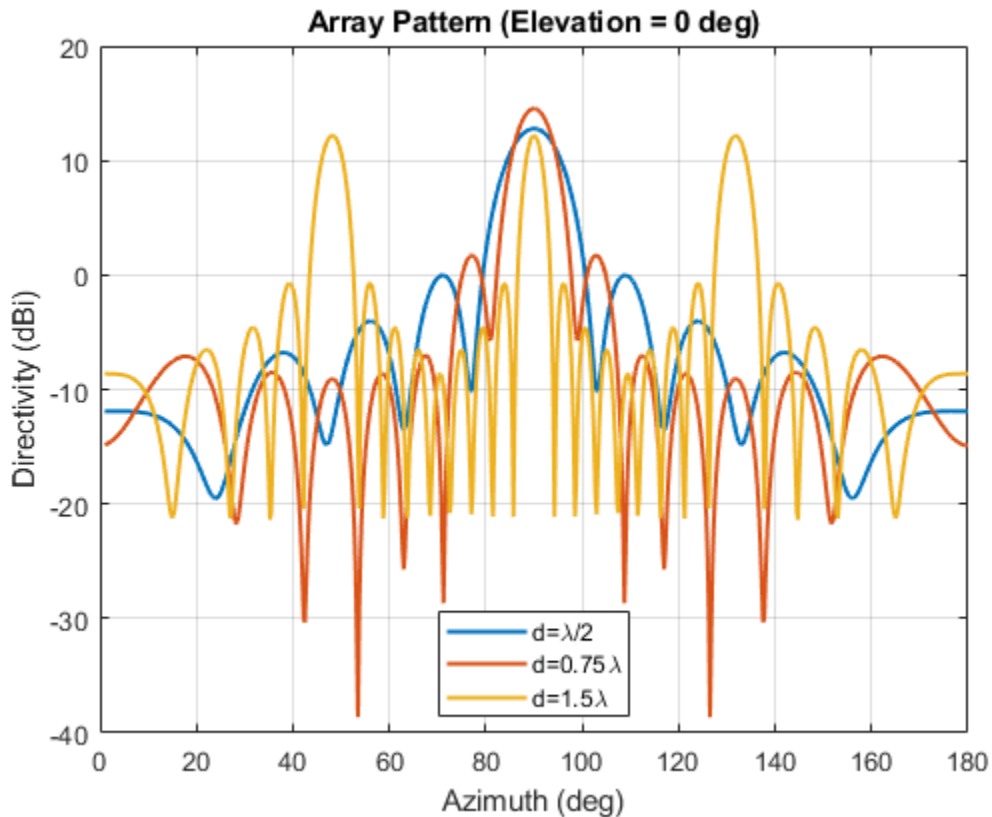
dipole_array.NumElements = 9;
dipole_array.ElementSpacing = lambda/2;
D_half_lambda = pattern(dipole_array,freq,az_angle,0,'CoordinateSystem','rectangular');
dipole_array.ElementSpacing = 0.75*lambda;
D_three_quarter_lambda = pattern(dipole_array,freq,az_angle,0,'CoordinateSystem','rectangular');
dipole_array.ElementSpacing = 1.5*lambda;
D_lambda = pattern(dipole_array,freq,az_angle,0,'CoordinateSystem','rectangular');
patterngrating1 = figure;
plot(az_angle,D_half_lambda,az_angle,D_three_quarter_lambda,az_angle,D_lambda,'LineWidth',1.5);

```

```

grid on
xlabel('Azimuth (deg)')
ylabel('Directivity (dBi)')
title('Array Pattern (Elevation = 0 deg)')
legend('d=\lambda/2','d=0.75\lambda','d=1.5\lambda','Location','best')

```



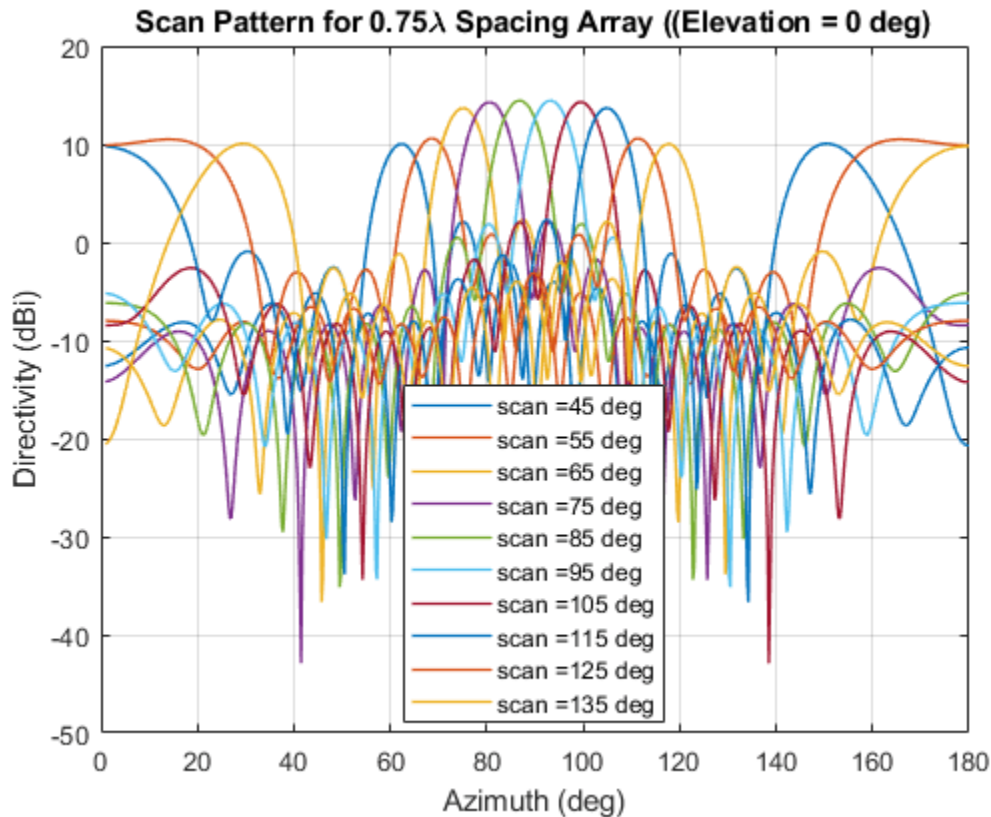
Compared to the $\lambda/2$ spaced array, the 1.5λ spaced array shows 2 additional equally strong peaks in the visible space - the grating lobes. The 0.75λ spaced array still has a single unique beam peak at zero scan off broadside (azimuthal angle of 90 deg). Scan this array off broadside to observe appearance of the grating lobes.

```

dipole_array.ElementSpacing = 0.75*lambda;
azscan = 45:10:135;
scanangle = [azscan ; zeros(1,numel(azscan))];
D_scan = nan(numel(azscan),numel(az_angle));
legend_string1 = cell(1,numel(azscan));
for i = 1:numel(azscan)
    ps = phaseshift(freq,dx,scanangle(:,i),N);
    dipole_array.PhaseShift = ps;
    D_scan(i,:) = pattern(dipole_array,freq,az_angle,0,'CoordinateSystem','rectangular');
    legend_string1{i} = strcat('scan = ',num2str(azscan(i)),' deg');
end
patterngrating2 = figure;
plot(az_angle,D_scan,'LineWidth',1)
xlabel('Azimuth (deg)')
ylabel('Directivity (dBi)')
title('Scan Pattern for 0.75\lambda Spacing Array ((Elevation = 0 deg)')

```

```
grid on
legend(legend_string1,'Location','best')
```



The 0.75λ spacing array with uniform excitation and zero phaseshift does not have grating lobes in the visible space. The peak of the main beam occurs at broadside (azimuth = 90 deg). However for the scan angle of 65 deg and lower, and for 115 deg and higher the grating lobe enters visible space. To avoid grating lobes, choose an element spacing of $\lambda/2$ or less. With smaller spacing, the mutual coupling is stronger.

Effect of Element Pattern on Array Pattern

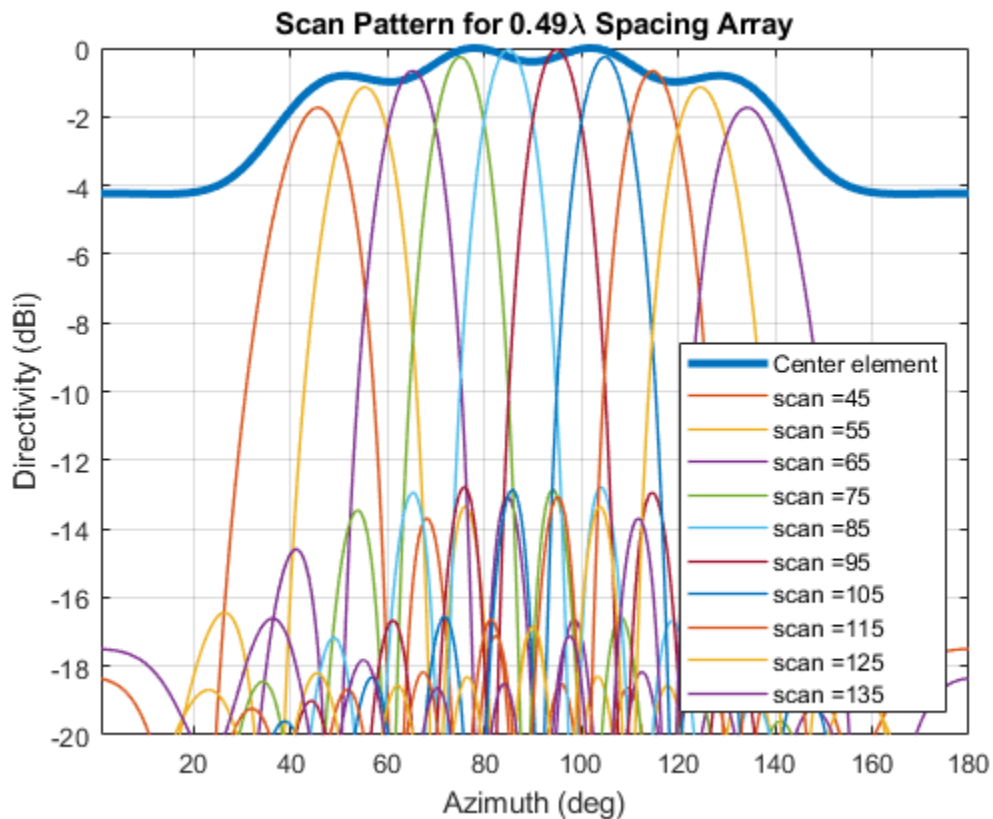
To study the effect of the element pattern on the overall array pattern, plot the normalized central element pattern versus the normalized directivity of the linear array of 9 dipoles at broadside.

```
dipole_array.ElementSpacing = 0.49*lambda;
dipole_array.PhaseShift = 0;
Dmax = pattern(dipole_array,freq,90,0);
D_scan = nan(numel(azscan),numel(az_angle)); % Pre-allocate
legend_string2 = cell(1,numel(azscan)+1);
legend_string2{1} = 'Center element';
for i = 1:numel(azscan)
    ps = phaseshift(freq,dx,scanangle(:,i),N);
    dipole_array.PhaseShift = ps;
    D_scan(i,:) = pattern(dipole_array,freq,az_angle,0,'CoordinateSystem','rectangular');
    D_scan(i,:) = D_scan(i,:) - Dmax;
    legend_string2{i+1} = strcat('scan = ',num2str(azscan(i)));
end
```

```

patternArrayVsElement = figure;
plot(az_angle,D_element(3,:) - max(D_element(3,:)), 'LineWidth',3)
hold on
plot(az_angle,D_scan,'LineWidth',1)
axis([min(az_angle) max(az_angle) -20 0])
xlabel('Azimuth (deg)')
ylabel('Directivity (dBi)')
title('Scan Pattern for 0.49\lambda Spacing Array')
grid on
legend(legend_string2,'Location','southeast')
hold off

```



Note the overall shape of the normalized array pattern approximately follows the normalized central element pattern close to broadside. The array pattern in general is the product of the element pattern and the array factor (AF).

Reference

[1] W. L. Stutzman, G. A. Thiele, Antenna Theory and Design, p. 307, Wiley, 3rd Edition, 2013.

See Also

“Design and Analysis Using Antenna Array Designer”

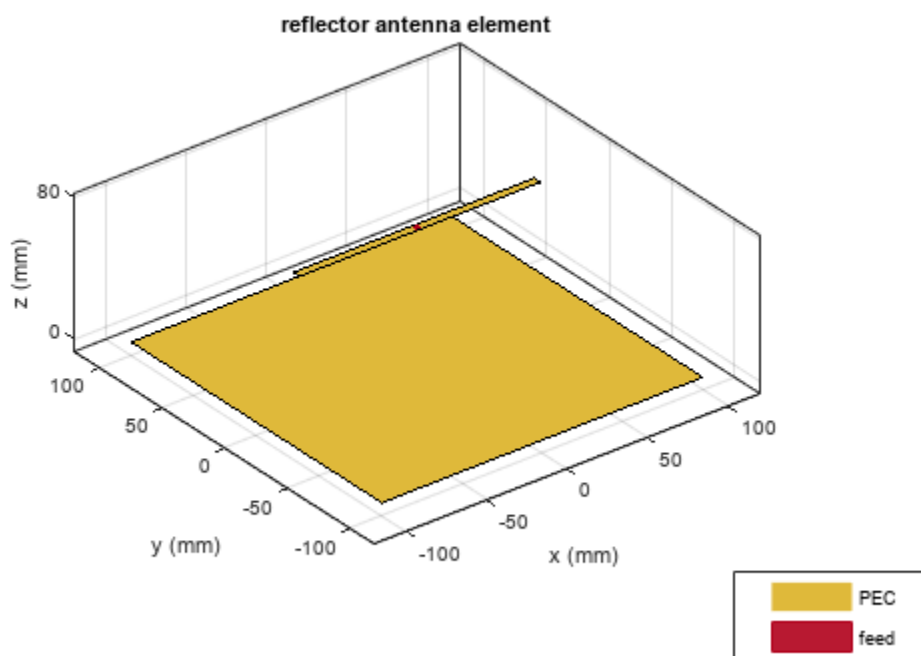
Modeling Infinite Ground Plane in Antennas and Arrays

This example illustrates modeling of antennas and arrays with infinite ground plane. The main advantage of modeling the ground plane as infinite is that the ground plane is not meshed. This helps in speeding up the solution. The structure is modelled using the method of images. Several antenna elements in Antenna Toolbox™ have a ground plane as part of the structure. For other elements, the ground plane can be introduced by placing them in front of a reflector.

Dipole Over Finite Ground Plane

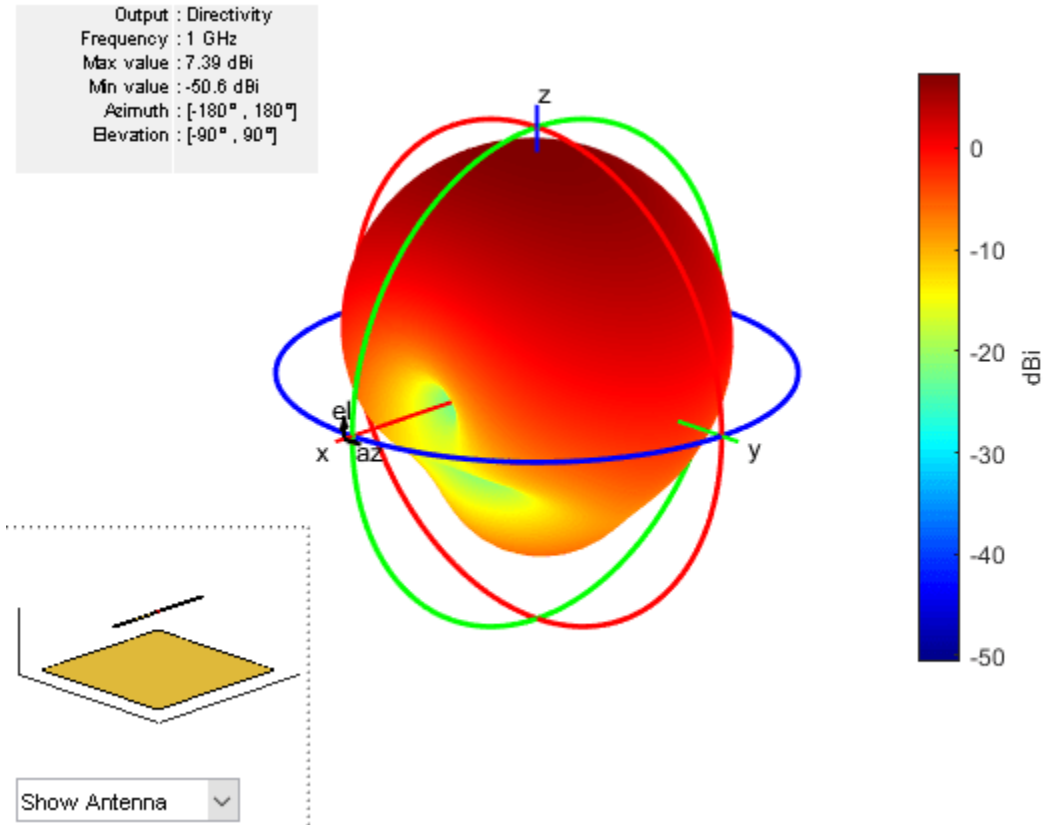
The default reflector has a dipole over a ground plane operating around 1GHz.

```
r = reflector;  
show(r);
```



Looking at the radiation pattern, we see that there is some leakage below the ground. This is because of the size of the ground plane.

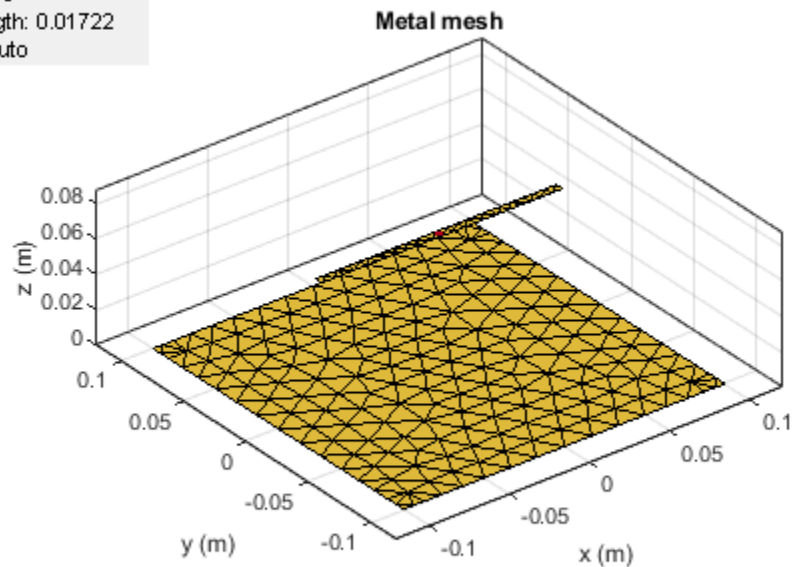
```
pattern(r, 1e9);
```



To prevent the leakage the ground plane must be made larger. However, by increasing the size of the ground plane the size of the mesh increases. The increase in mesh size increases the simulation time. Below is the mesh created for the structure above.

```
mesh(r);
```

```
NumTriangles: 382  
NumTetrahedra: 0  
NumBasis: 518  
MaxEdgeLength: 0.01722  
MeshMode: auto
```

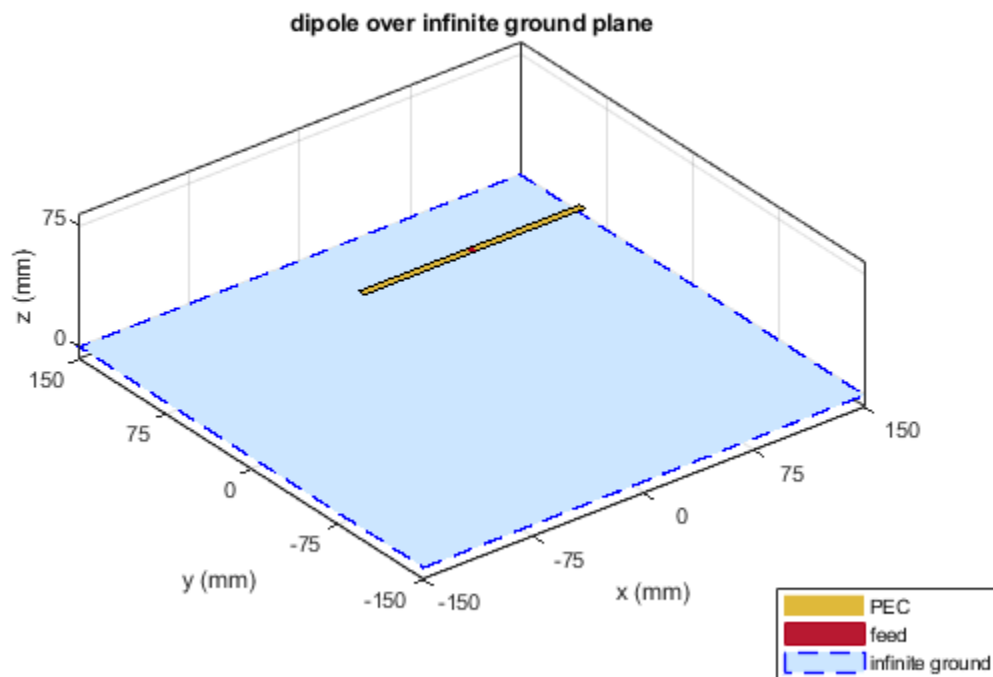


Increasing the ground plane size will put more triangles on the ground which will result in more simulation time.

Dipole Over Infinite Ground Plane

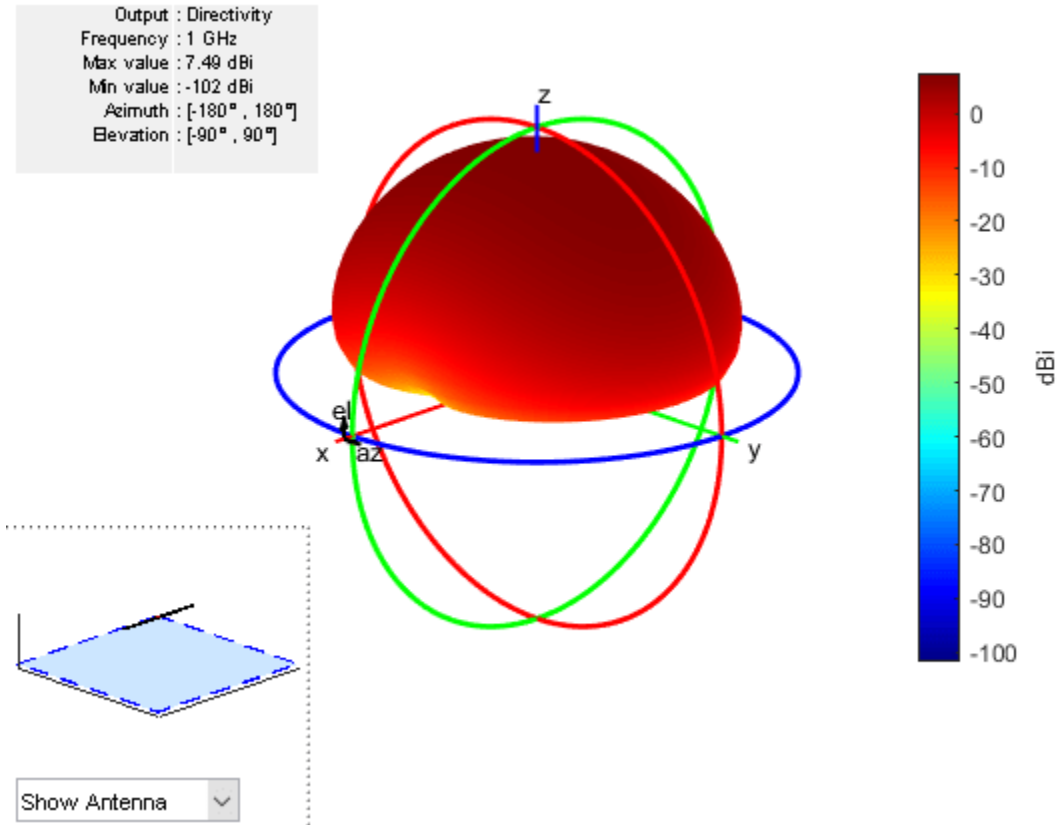
A simple way to prevent leakage below the ground plane is to make the ground plane infinite. This can be achieved by making either the `GroundPlaneLength` or `GroundPlaneWidth` or both to be infinite. In this case the ground plane is replaced by a blue sheet indicating that the ground plane is not made of metal.

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

The pattern plot shows no leakage below the ground. This result can be used as a first pass to get a general idea of the antenna. The infinite ground plane can be replaced by a large finite one at the end to look for edge effects. Another interesting factor is the increase in maximum directivity value. As there is no back lobe, all the energy is radiated above the ground plane, increasing the maximum directivity from 7.38 to 7.5dBi.

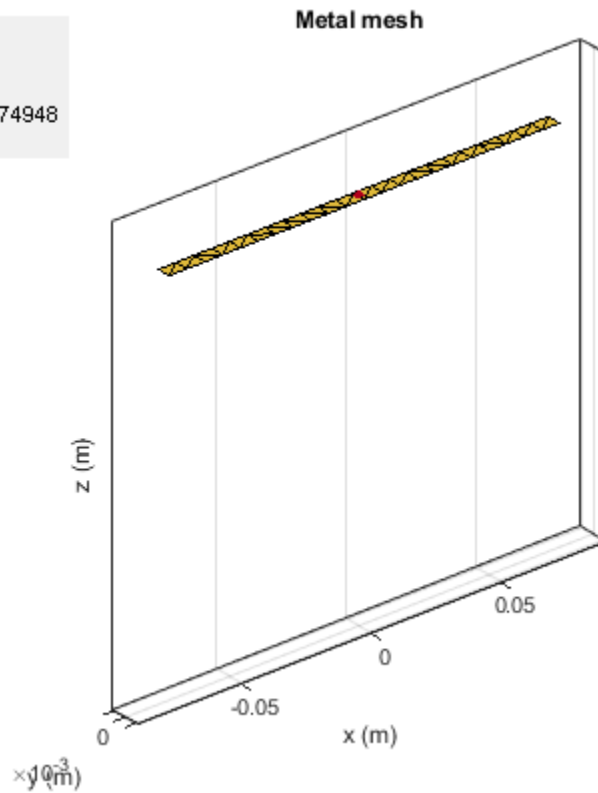
```
pattern(r, 1e9);
```



As mentioned before, the infinite ground plane is not meshed. The mesh for this structure shows only the mesh for the dipole element.

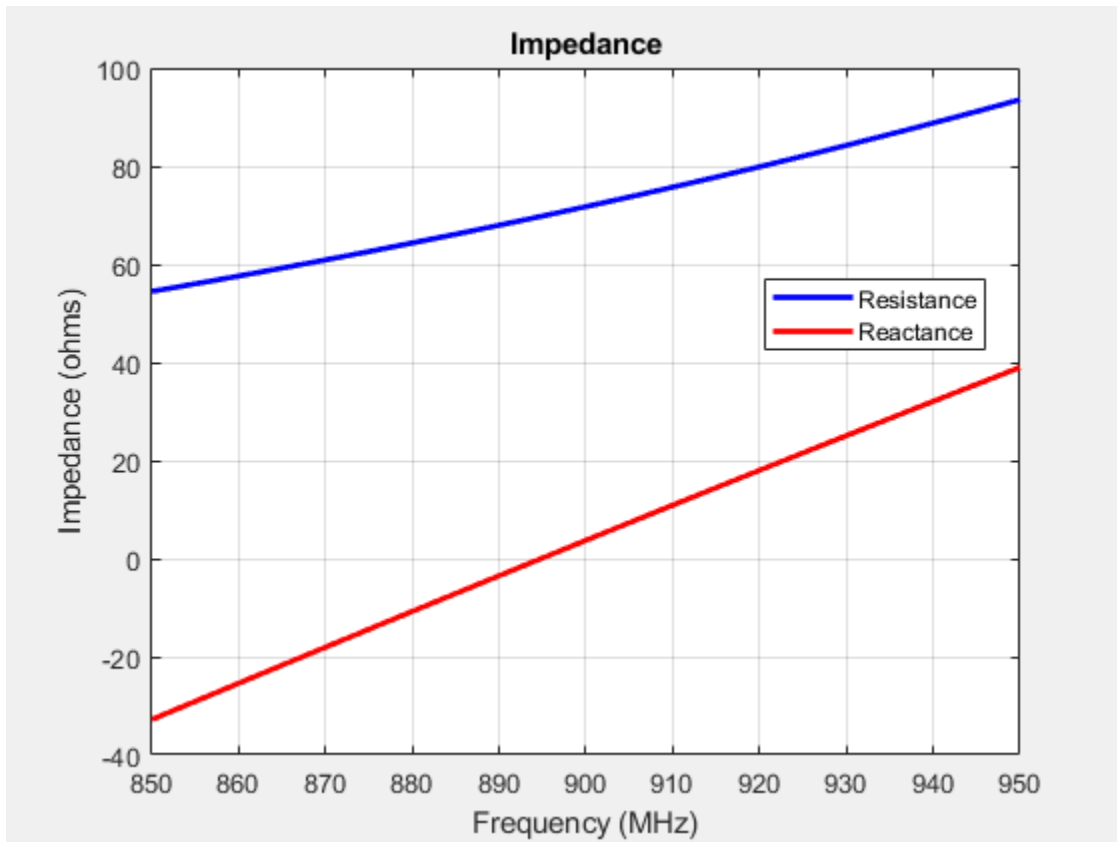
```
mesh(r);
```

NumTriangles: 44
NumTetrahedra: 0
NumBasis: 43
MaxEdgeLength: 0.0074948
MeshMode: auto



The impedance of the reflector with infinite ground plane looks similar to the one with the finite ground plane. The resonance value is shifted slightly from 880MHz for finite ground to 890MHz for the infinite case.

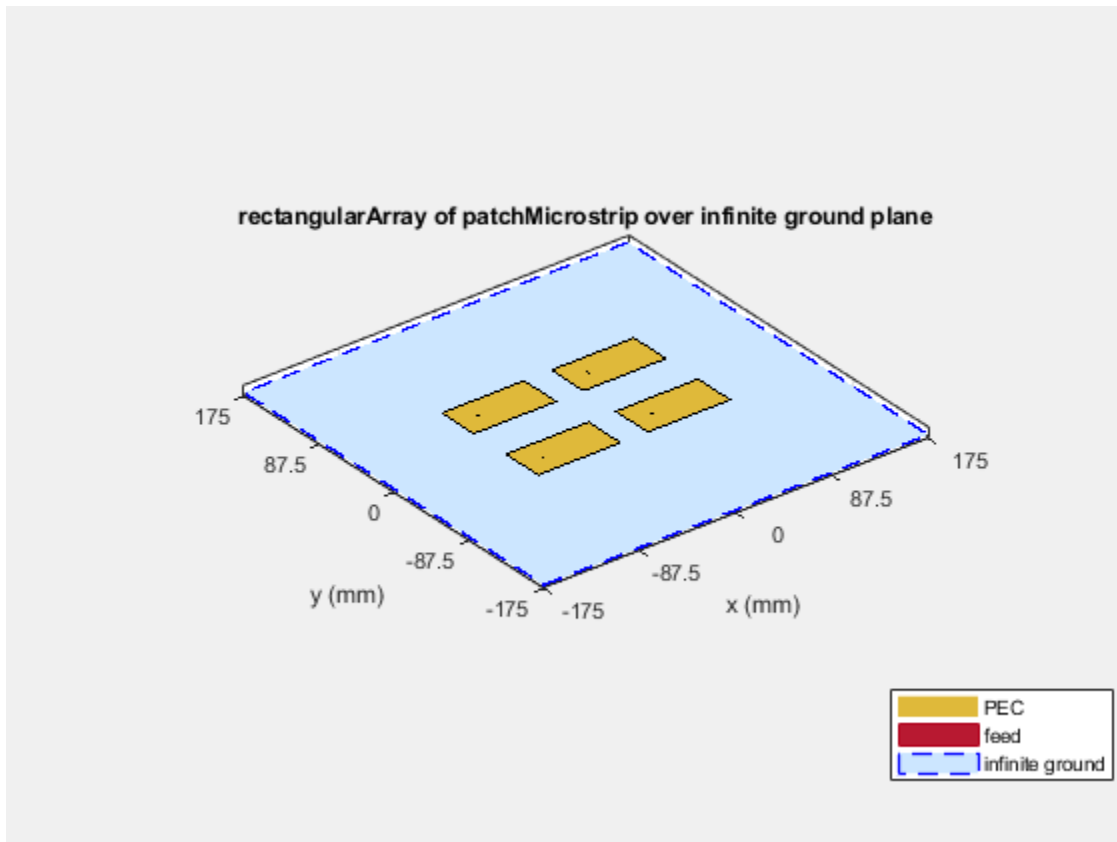
```
impedance(r, 850e6:2e6:950e6);
```



Patch Antenna Array Over Infinite Ground

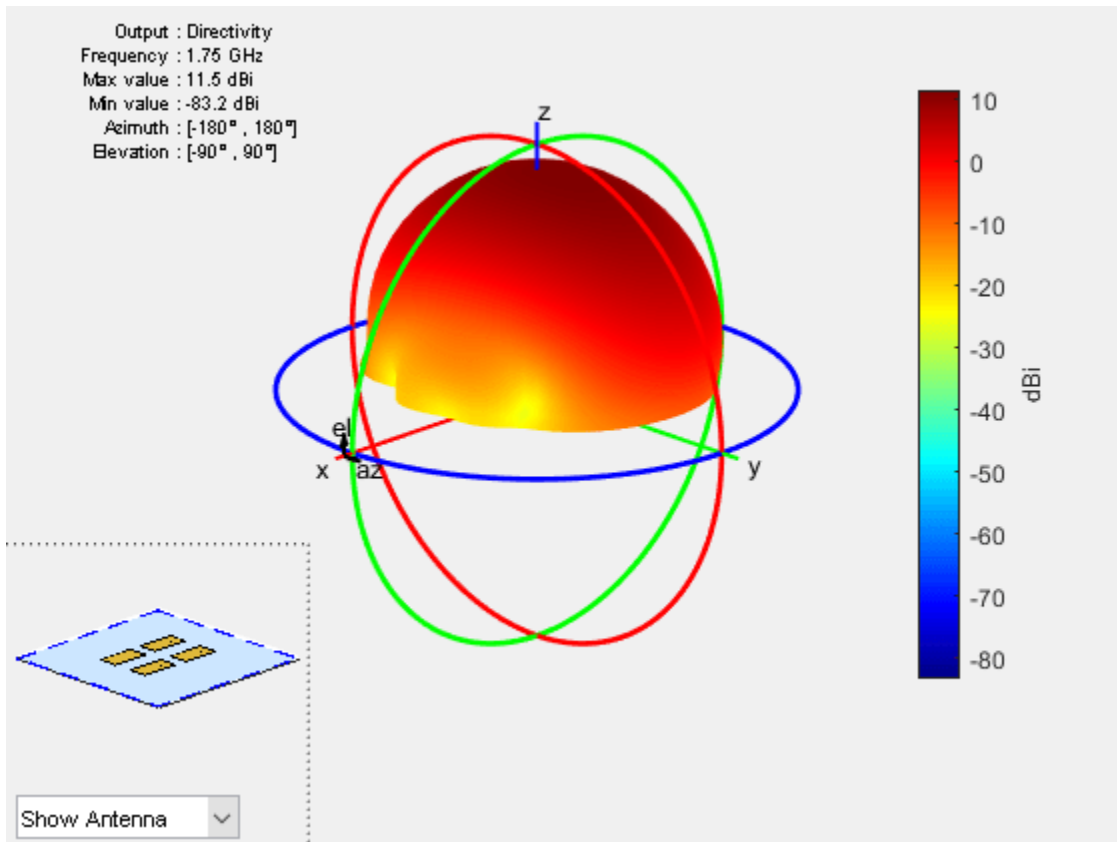
The concept of infinite ground becomes even more important for arrays. As the number of elements increase the size of the ground plane increases dramatically as the space on the ground between the elements also needs to be meshed. So choosing infinite ground plane for arrays is fairly common. Infinite ground planes in arrays are also called ground screens.

```
p = patchMicrostrip('GroundPlaneWidth', inf);  
arr = rectangularArray('Element', p);  
arr.RowSpacing = 0.075;  
arr.ColumnSpacing = 0.1;  
show(arr);
```



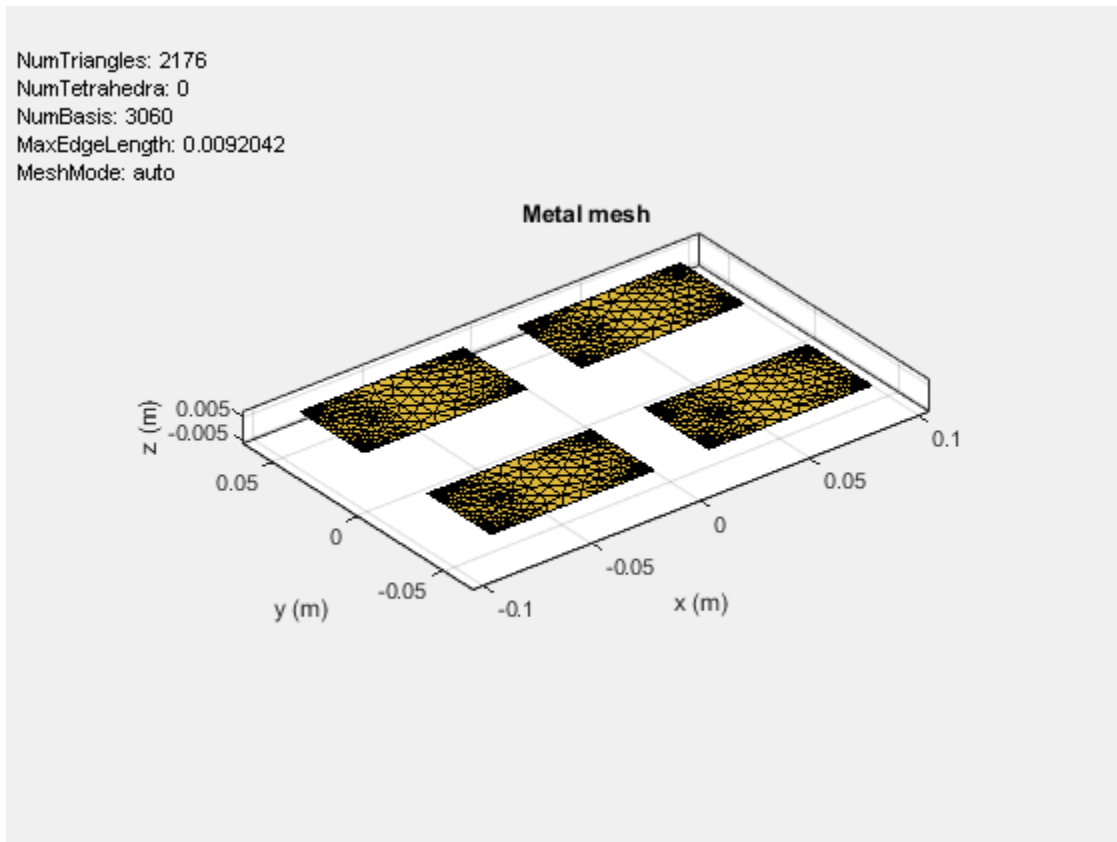
Above is the four element rectangular patch array on an infinite ground plane. The individual patches resonate around 1.75 GHz. Below is the pattern of the full array.

```
pattern(arr, 1.75e9);
```



Again the plot shows that there is no radiation below the ground. The mesh that is used to solve the array is shown. As mentioned previously, the ground is not meshed so the overall size of the system is reduced resulting in faster computation time.

```
mesh(arr);
```



References

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

See Also

"Metasurface Antenna Modeling" on page 5-381 | "Model Infinite Ground Plane for Balanced Antennas" on page 1-43

Infinite Array Analysis

This example shows the use of infinite array analysis to model the behavior of a single element - the *unit cell* embedded in the array[1] - [3]. The array is assumed to be of infinite extent in two dimensions, and located in the *XY*-plane.

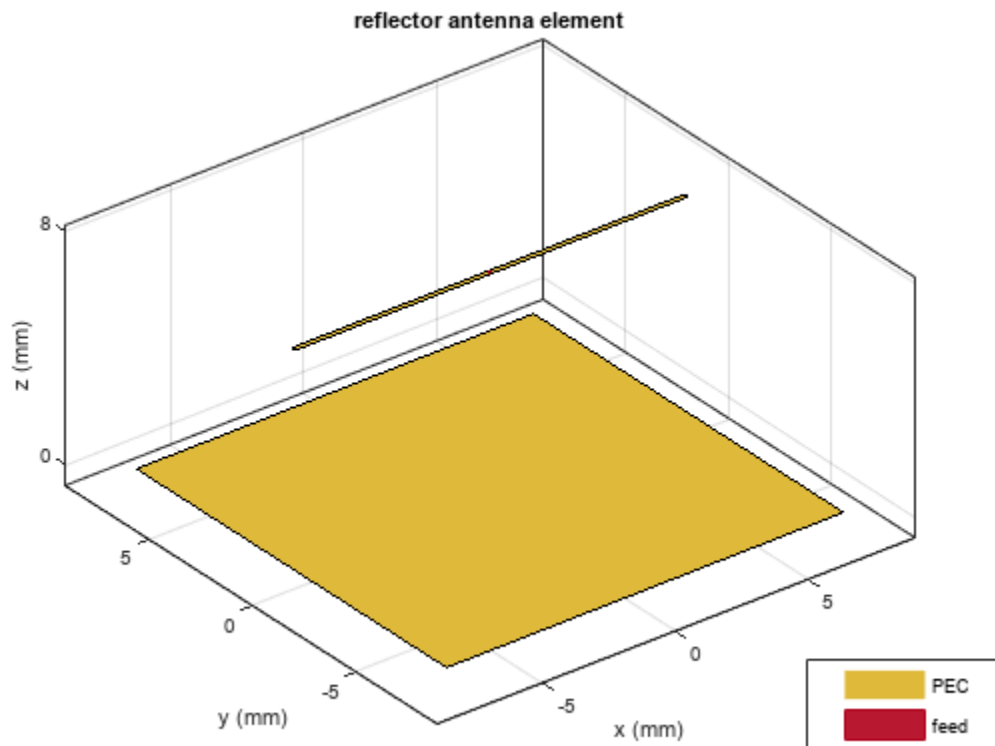
Define Unit Cell

The unit cell refers to a single element in an infinite array. The unit cell element needs a ground plane. Antennas that do not have a groundplane need to be backed by a reflector. A representative example for each case is a microstrip patch antenna having a groundplane and a dipole antenna backed by a reflector, respectively. This example uses a dipole antenna backed by a reflector as the element of the unit cell and analyzes the impedance behavior of the unit cell at 10 GHz. The unit cell has a cross-section of $\lambda/2$ by $\lambda/2$ m. To define the unit cell, create a thin dipole of length slightly less than $\lambda/2$ m and assign it as the exciter to a square reflector of dimensions $\lambda/2$ m on each side.

```
%Define constants
freq = 10e9;
vp = physconst('lightspeed');
lambda = vp/freq;
ucdx = 0.5*lambda;
ucdy = 0.5*lambda;

%Define individual element
d = dipole;
d.Length = 0.495*lambda;
d.Width = lambda/160;
d.Tilt = 90;
d.TiltAxis = [0 1 0];

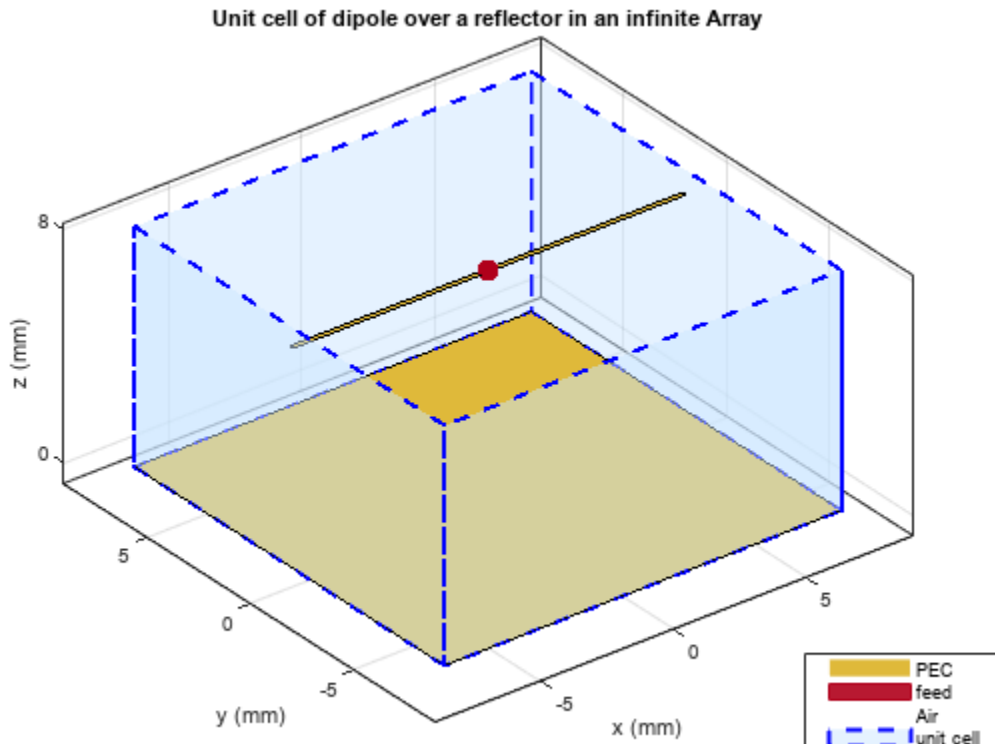
%Define reflector
r = reflector;
r.Exciter = d;
r.Spacing = lambda/4;
r.GroundPlaneLength = ucdx;
r.GroundPlaneWidth = ucdy;
figure
show(r)
```

Create Infinite Array

Create the infinite array and assign the reflector backed dipole as the element and view it.

```
infArray = infiniteArray;  
infArray.Element = r;  
infArrayFigure = figure;  
show(infArray)
```



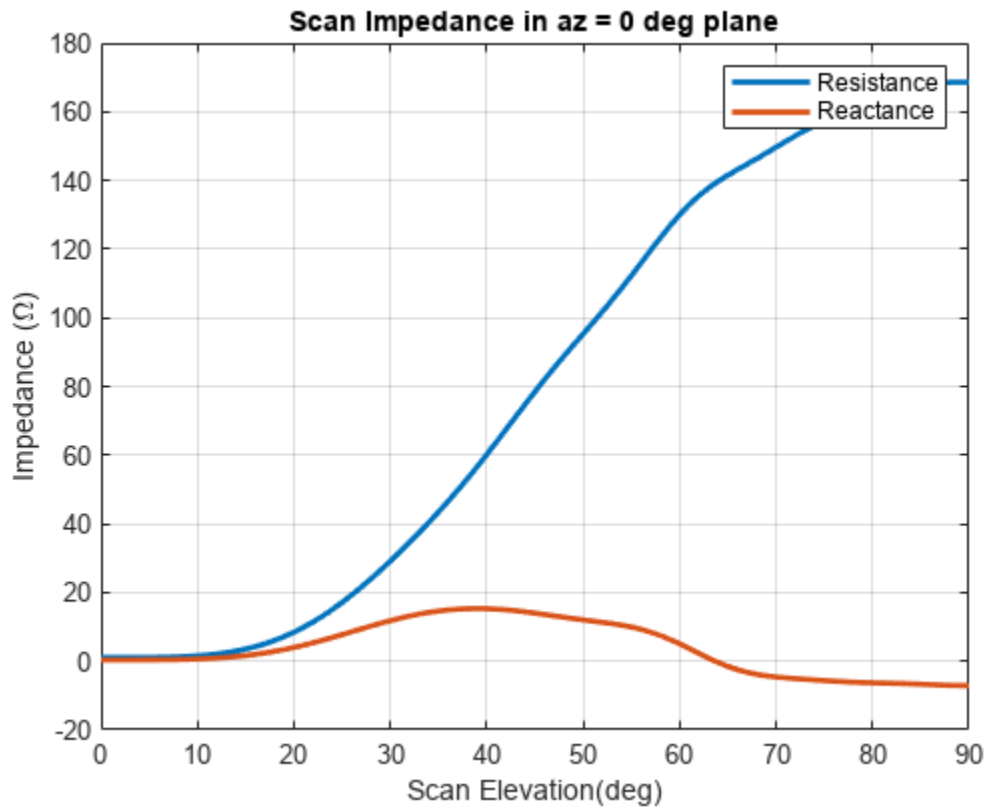
Calculate Scan Impedance

Analyze the impedance behavior of the infinite array by calculating the scan impedance. The scan impedance is the impedance variation of the unit cell element at a single frequency as a function of scan angle. Use the scan angle properties `ScanAzimuth` and `ScanElevation` on the infinite array to define the scan behavior. Calculate the scan impedance in a single plane, defined by azimuth = 0 deg and elevation varying from 0 to 90 deg in 1 deg steps.

```
% Scan plane definition
az = 0;           % E-plane
el = 0:1:90;     % elevation

% Calculate and plot
scanZ = nan(1,numel(el));
infArray.ScanAzimuth = az;
for i = 1:numel(el)
    infArray.ScanElevation = el(i);
    scanZ(i) = impedance(infArray,freq);
end

figure
plot(el,real(scanZ),el,imag(scanZ),'LineWidth',2);
grid on
legend('Resistance','Reactance')
xlabel('Scan Elevation(deg)')
ylabel('Impedance (\0mega)')
title(['Scan Impedance in az = ' num2str(az) ' deg plane'])
```



Improving Convergence Behavior

The infinite array analysis depends on a periodic Green's function which comprises of an infinite double summation. For more information about this please refer to the documentation page([infiniteArray](#)). The number of terms in this double summation has an impact on the convergence of the results. Increase the number of summation terms to improve the convergence. Execute the below command to increase the total number of terms to 101 (50 terms each for negative and positive indices, 1 term for 0th term) from the default of 21.

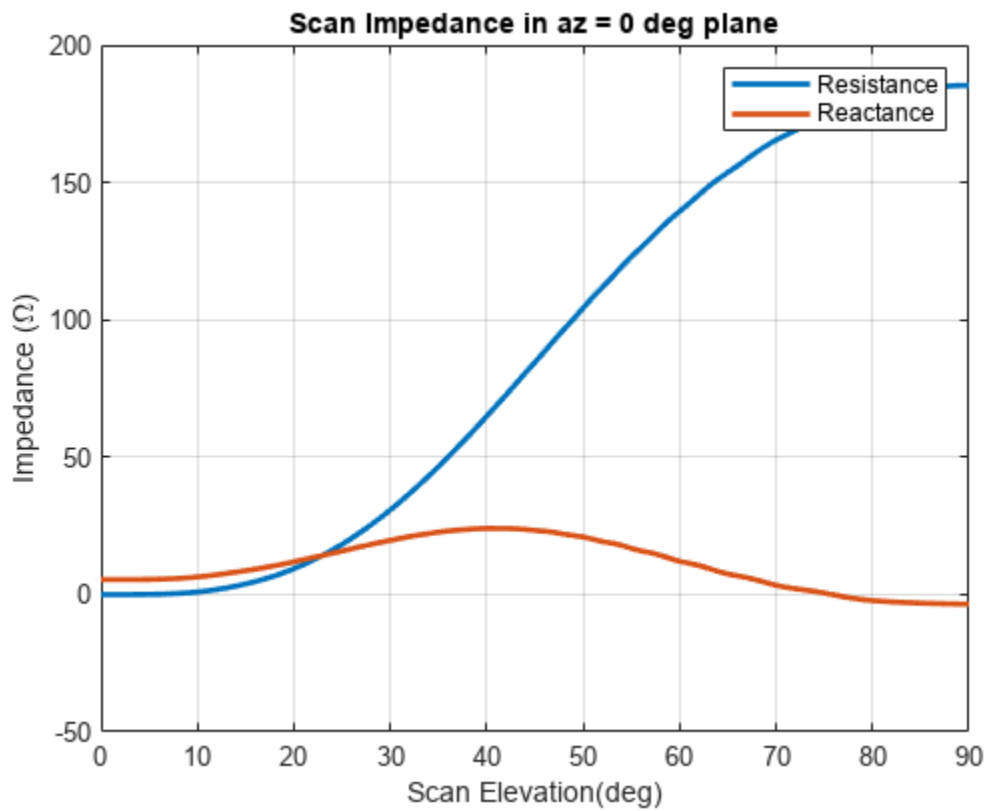
```
numSummationTerms(infArray,50);
```

Section below shows the analysis result with higher number of terms in the scan impedance in 3 planes, azimuth = 0, 45 and 90 deg respectively. It takes about 100 seconds per scan plane on a 2.4 GHz machine with 32 GB memory.

```
az = [0 45 90];           % E,D,H-plane
load scanZData
```

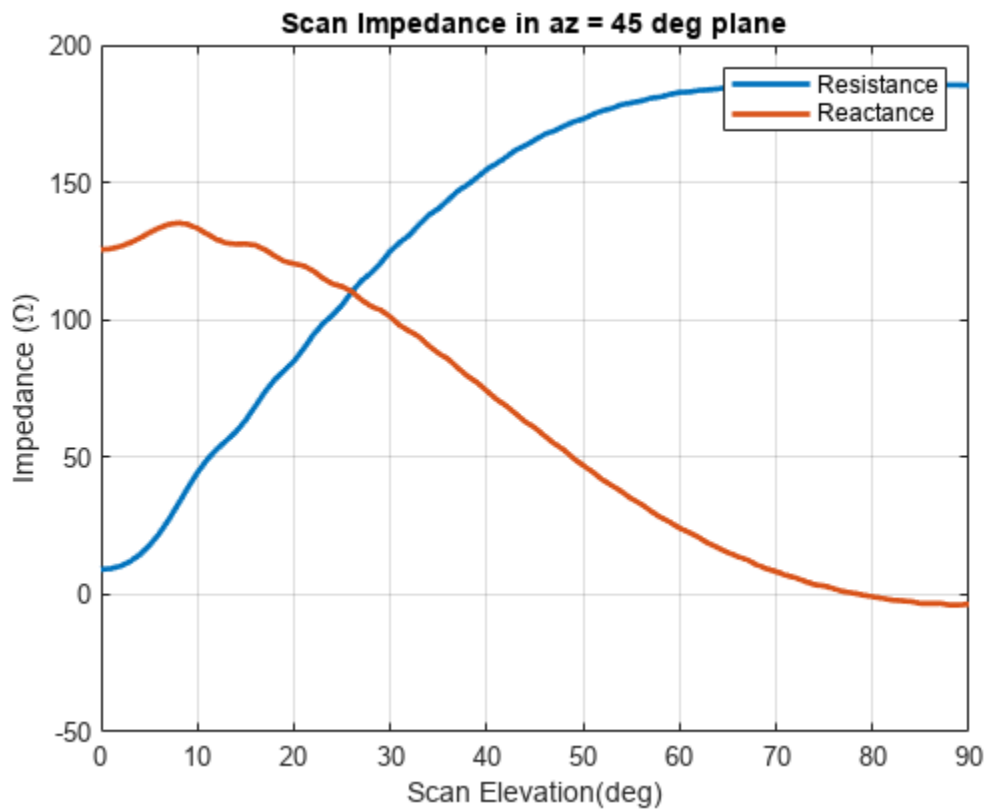
E-plane

```
figure
plot(eI,real(scanZ50terms(1,:)),eI,imag(scanZ50terms(1,:)),'LineWidth',2);
grid on
legend('Resistance','Reactance')
xlabel('Scan Elevation(deg)')
ylabel('Impedance (\0mega)')
title(['Scan Impedance in az = ' num2str(az(1)) ' deg plane'])
```



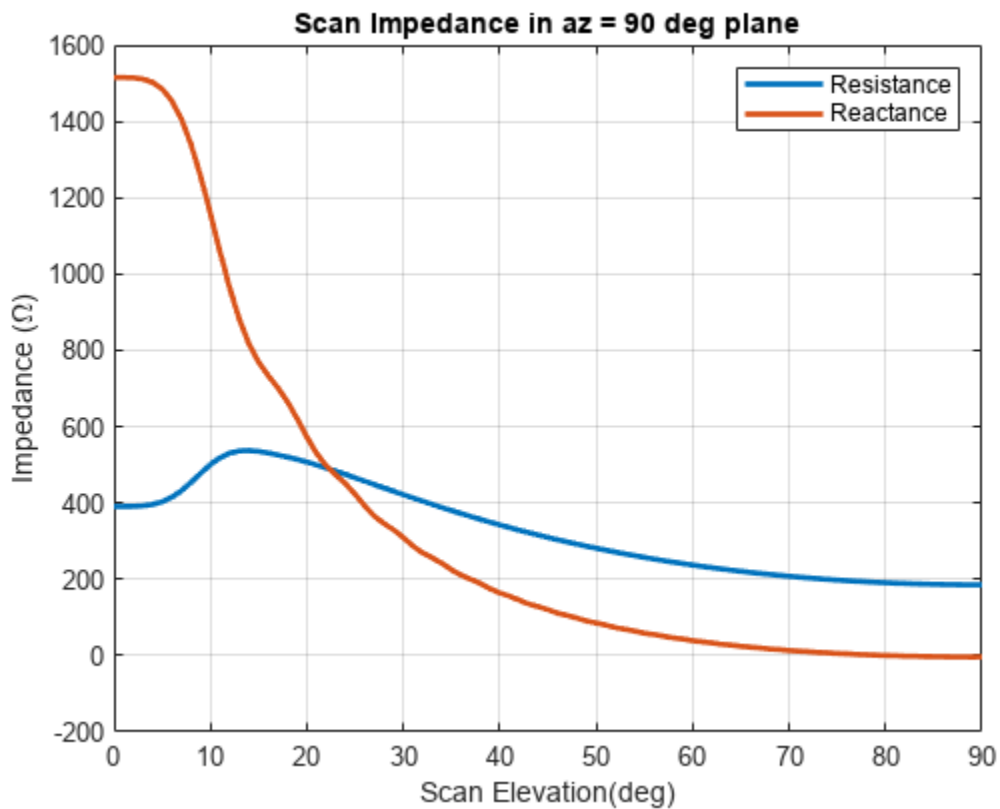
D-plane

```
figure
plot(e1,real(scanZ50terms(2,:)),e1,imag(scanZ50terms(2,:)),'LineWidth',2);
grid on
legend('Resistance','Reactance')
xlabel('Scan Elevation(deg)')
ylabel('Impedance (\Omega)')
title(['Scan Impedance in az = ' num2str(az(2)) ' deg plane'])
```



H-plane

```
figure
plot(e1,real(scanZ50terms(3,:)),e1,imag(scanZ50terms(3,:)),'LineWidth',2);
grid on
legend('Resistance','Reactance')
xlabel('Scan Elevation(deg)')
ylabel('Impedance (\Omega)')
title(['Scan Impedance in az = ' num2str(az(3)) ' deg plane'])
```



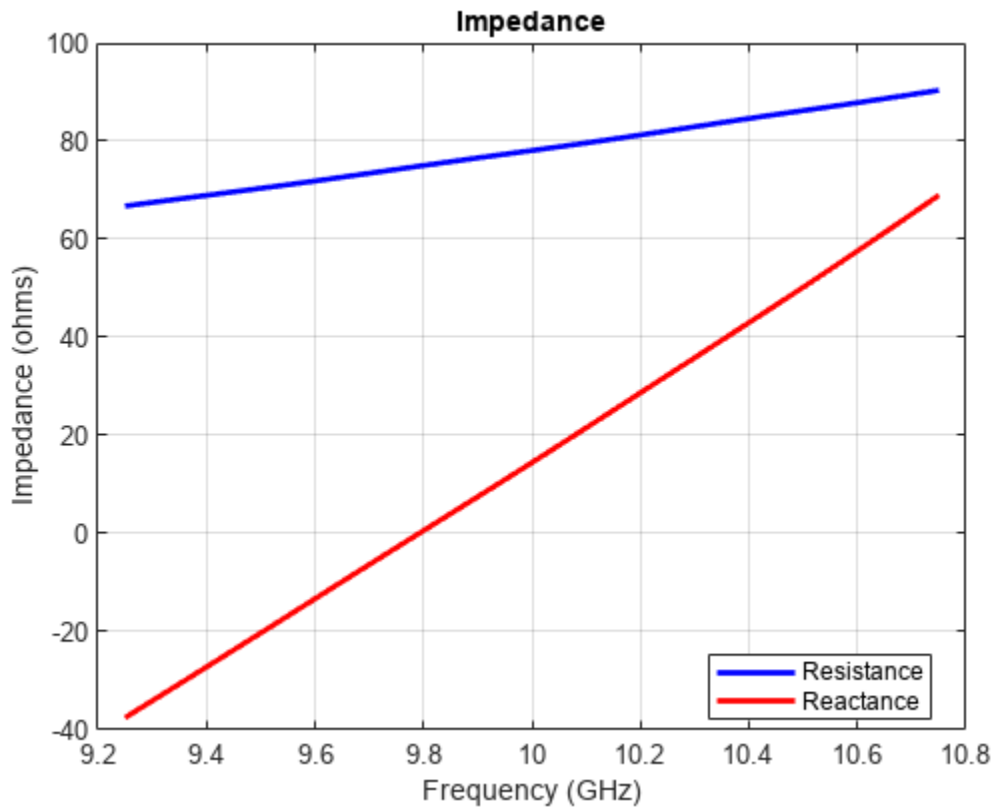
Infinite Array Impedance Variation with Frequency

Fix the scan angle to a specific value and sweep the frequency to observe the impedance behavior of this unit cell element.

```

az_scan = 0;
el_scan = 45;
percent_bw = .15;
bw = percent_bw*freq;
fmin = freq - bw/2;
fmax = freq + bw/2;
infArray.ScanAzimuth = az_scan;
infArray.ScanElevation = el_scan;
figure
impedance(infArray,linspace(fmin,fmax,51));

```



Calculate Isolated Element Pattern and Impedance

Use the scan impedance data from the infinite array analysis to derive the scan element pattern (also known as embedded/array element pattern in case of finite arrays). As indicated in [1]-[4] use the isolated element pattern and impedance to calculate it. Do this by analyzing the dipole backed by an infinite reflector and calculating its power pattern and impedance at 10 GHz.

```

r.GroundPlaneLength = inf;
r.GroundPlaneWidth = inf;
giso = nan(numel(az),numel(el));
gisodB = nan(numel(az),numel(el));
for i = 1:numel(az)
    giso(i,:) = pattern(r,freq,az(i),el,'Type','power');
    gisodB(i,:) = 10*log10(giso(i,:));
    gisodB(i,:) = gisodB(i,:) - max(gisodB(i,:));
end
Ziso = impedance(r,freq);

```

Calculate and Plot Scan Element Pattern

Define the generator impedance to calculate the scan element pattern. This example uses broadside scan resistance as the generator impedance.

```

Rg = 185;
Xg = 0;
Zg = Rg + 1i*Xg;
gs = nan(numel(az),numel(el));

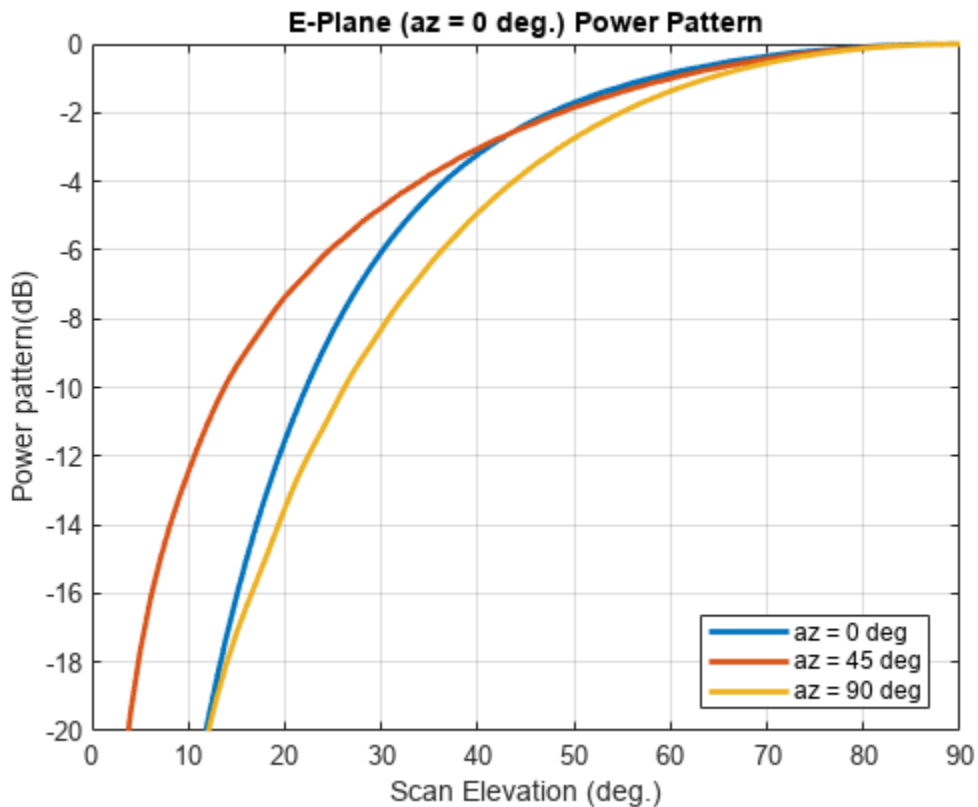
```

```

gsdB = nan(numel(az),numel(el));
for i = 1:numel(az)
    gs(i,:) = 4*Rg*real(Ziso).*giso(i,:)./(abs(scanZ50terms(i,:) + Zg)).^2;
    gsdB(i,:)= 10*log10(gs(i,:));
    gsdB(i,:)= gsdB(i,:) - max(gsdB(i,:));
end

figure;
plot(el,gsdB(1,:),el,gsdB(2,:),el,gsdB(3,:), 'LineWidth',2.0)
grid on
axis([0 90 -20 0])
xlabel('Scan Elevation (deg.)')
ylabel('Power pattern(dB)')
title(strcat('E-Plane (az = 0 deg.) Power Pattern'))
legend('az = 0 deg','az = 45 deg','az = 90 deg','Location','best')

```



Reference

- [1] J. Allen, "Gain and impedance variation in scanned dipole arrays," IRE Transactions on Antennas and Propagation, vol.10, no.5, pp.566-572, September 1962.
- [2] R. C. Hansen, Phased Array Antennas, Chapter 7 and 8, John Wiley & Sons Inc., 2nd Edition, 1998.
- [3] R. J. Mailloux, 'Phased Array Antenna Handbook', Artech House, 2nd edition, 2005

[4] W. Stutzman, G. Thiele, 'Antenna Theory and Design', John Wiley & Sons Inc., 3rd Edition, 2013.

See Also

"Modeling Infinite Ground Plane in Antennas and Arrays" on page 5-48 | "Infinite Array of Microstrip Patch Antenna on Teflon Substrate" on page 5-780

Parallelization of Antenna and Array Analyses

This example shows how to speed up antenna and array analysis using Parallel Computing Toolbox™.

This example requires the following product:

- Parallel Computing Toolbox

Frequency Sweep Analysis

Analyzing antenna performance over a frequency band is an important part of antenna design. Port analyses include impedance, returnLoss, sparameters, vswr; surface analyses include current and charge; and field analyses include pattern, EHfields, axialRatio, and beamwidth. In this example, we demonstrate the advantages of parallel computing for computing axial ratio and return loss over a frequency band.

Axial Ratio Computation without Parallel Computing

Compute the axial ratio of an Archimedean spiral over a frequency band of 0.8 to 2.5 GHz in steps of 100 MHz. This calculation is done serially, and the time taken to perform these computations is saved in variable time1.

```
sp = spiralArchimedean('Turns',4,'InnerRadius',5.5e-3,'OuterRadius',50e-3);
freq = 0.8e9:100e6:2.5e9;
AR = zeros(size(freq));
tic
for m = 1:numel(freq)
    AR(m) = axialRatio(sp, freq(m), 0, 90);
end
time1 = toc;
```

Axial Ratio Computation with Parallel Computing

Repeat the calculations using Parallel Computing Toolbox to reduce computation time. Use the function parpool to create a parallel pool cluster. Then, use parfor to calculate the axial ratio over the same frequency band. The time taken to perform the computation is saved in variable time2.

```
pardata = parpool;
```

```
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 6).
```

```
sp = spiralArchimedean('Turns',4,'InnerRadius',5.5e-3,'OuterRadius',50e-3);
ARparfor = zeros(size(freq));
tic;
parfor m = 1:numel(freq)
    ARparfor(m) = axialRatio(sp, freq(m), 0, 90);
end
time2 = toc;
```

Axial Ratio Computation Time Comparison

The table below shows the time taken for axial ratio analysis with and without Parallel Computing Toolbox. The cluster information is saved in the variable pardata.

```
cases = {'Without Parallel Computing';'With Parallel Computing'};
time = [time1; time2];
```

```
numWorkers = [1; pardata.NumWorkers];
disp(table(time,numWorkers,'RowNames', cases))
```

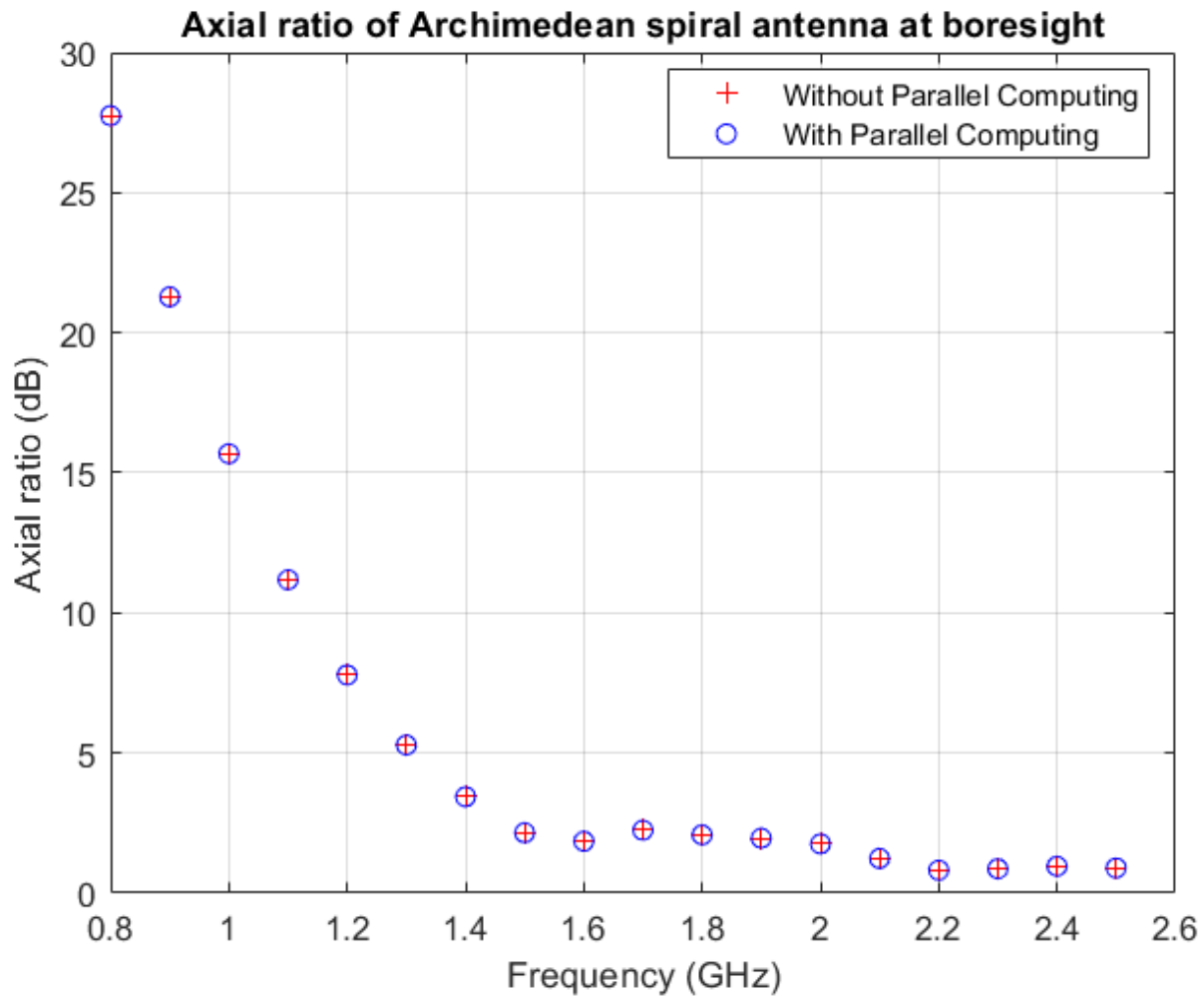
	time	numWorkers
Without Parallel Computing	46.155	1
With Parallel Computing	18.382	6

```
fprintf('Speed-up due to parallel computing = %g', time1/time2)
```

```
Speed-up due to parallel computing = 2.5109
```

The plot below shows the axial ratio data calculated for two cases. The results are identical.

```
plot(freq./1e9, AR,'r+', freq./1e9, ARparfor,'bo');
grid on;
xlabel('Frequency (GHz)');
ylabel('Axial ratio (dB)');
title('Axial ratio of Archimedean spiral antenna at boresight');
legend('Without Parallel Computing','With Parallel Computing',
       'location','Best');
```



During this analysis the antenna structure is meshed at every frequency and then the far-fields are computed at that frequency to compute the axial ratio. One way of reducing the analysis time is to mesh the structure manually by specifying a maximum edge length.

Return Loss Computation without Parallel Computing

The previous section performed a field analysis computation. All field and surface analysis computations in Antenna Toolbox™ accept only scalar frequency as input. Whereas, returnLoss and all other port analysis functions accept a frequency vector as input.

When a frequency vector is specified as an input, the antenna structure is meshed at the highest frequency. The resulting mesh is used for performing the analysis over the specified frequency band. The CPU time taken to perform the computation is saved in variable time3.

```
sp = spiralArchimedean('Turns',4,'InnerRadius',5.5e-3,'OuterRadius',50e-3);
tic;
RL = returnLoss(sp, freq);
time3 = toc;
```

Return Loss Computation with Parallel Computing

If we use the parfor loop by passing a single frequency at a time (as shown in the discussion about axialRatio), the meshing will happen at every frequency. This will limit the advantage of parallel computing. A better option is to manually mesh the structure and then use the parfor loop to run a frequency sweep. A simple way to do this is to run the analysis at the highest frequency first and get the mesh information using the mesh function. Use the maximum edge length in the mesh function to ensure that the same mesh is used for all computations. The time taken to perform the computation is saved in variable time4.

```
sp = spiralArchimedean('Turns',4,'InnerRadius',5.5e-3,'OuterRadius',50e-3);
RLparfor = zeros(size(freq));
tic;
temp = returnLoss(sp, freq(end));
meshdata = mesh(sp);
[~] = mesh(sp, 'MaxEdgeLength', meshdata.MaxEdgeLength);
parfor m = 1:numel(freq)
    RLparfor(m) = returnLoss(sp, freq(m));
end
time4 = toc;
```

Return Loss Computation Time Comparison

The table below indicated the time taken for calculating the return loss with and without Parallel Computing Toolbox. The cluster information is saved in the variable pardata.

```
cases = {'Without Parallel Computing','With Parallel Computing'};
time = [time3; time4];
numWorkers = [1; pardata.NumWorkers];
disp(table(time,numWorkers,'RowNames', cases))
```

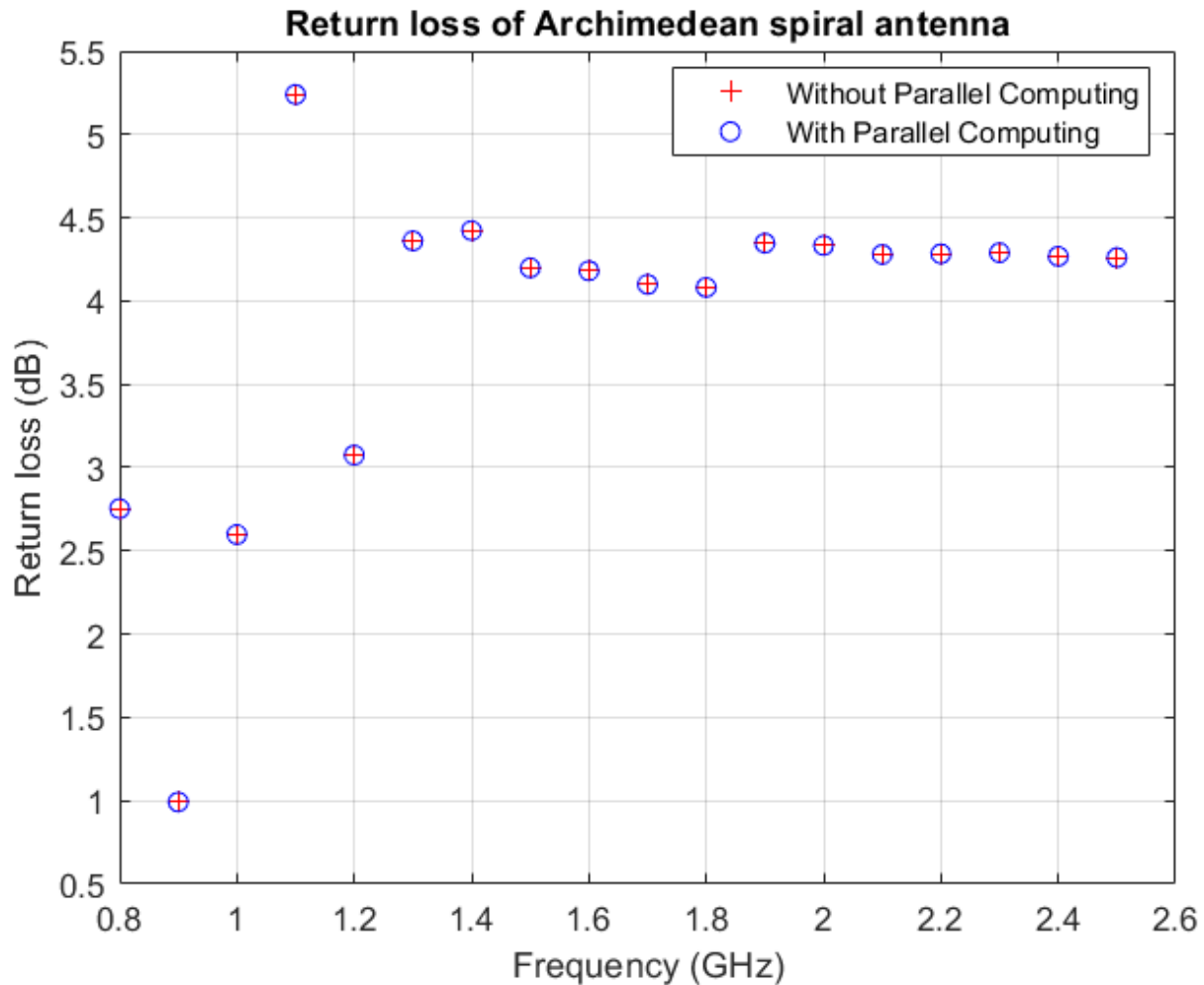
	time	numWorkers
Without Parallel Computing	5.5858	1
With Parallel Computing	3.3974	6

```
fprintf('Speed-up due to parallel computing = %g', time3/time4);
```

Speed-up due to parallel computing = 1.64412

The plot below shows the return loss data calculated for two cases. The results are identical.

```
plot(freq./1e9, RL, 'r+', freq./1e9, RLparfor, 'bo');
grid on;
xlabel('Frequency (GHz)');
ylabel('Return loss (dB)');
title('Return loss of Archimedean spiral antenna');
legend('Without Parallel Computing', 'With Parallel Computing', ...
      'location', 'Best');
```



Delete the current parallel pool.

```
delete(gcf);
```

See Also

“Port Analysis of Antenna” on page 5-4 | “Antenna Near-Field Visualization” on page 5-27

Analysis of Monopole Impedance

This example analyzes the impedance behavior of a monopole at varying mesh resolution/sizes and at a single frequency of operation. The resistance and reactance of the monopole are plotted and compared with the theoretical results. A relative convergence curve is established for the impedance.

Choose Frequency of Operation and Calculate Wavelength

Choose the operating frequency for the monopole and calculate the wavelength in free space at the frequency.

```
f = 400e6;  
speedOfLight = physconst('lightspeed');  
lambda = speedOfLight / f;
```

Select Physical Parameters of Monopole

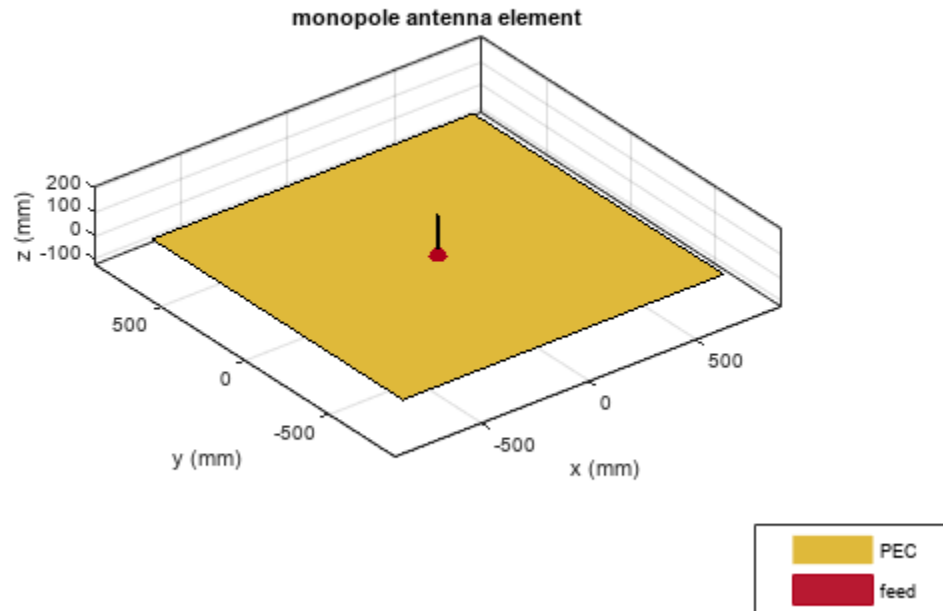
The monopole is typically fed by a coaxial transmission line of characteristic impedance 50 ohm. Define the height of the monopole to be slightly less than quarter-wavelength [1], $h/\lambda = 0.236$. The radius of the monopole also influences the impedance. Define the radius in terms of wavelength, $a/\lambda = 0.001588$. The monopole model in the Antenna Toolbox™ uses a metal strip. The width of the strip, w , is related to the radius, a , of an equivalent metal cylinder by $w = 4a$ [2]. Choose a large ground plane by specifying the length and width of the ground plane to be twice the operating wavelength.

```
h_over_lambda = 0.236;  
a_over_lambda = 0.001588;  
h = h_over_lambda*lambda;  
a = a_over_lambda*lambda;  
w = cylinder2strip(a);  
gpL = 2*lambda;  
gpW = 2*lambda;
```

Create and Modify Monopole

Create a monopole antenna and modify its properties to match the design parameters.

```
mp = monopole;  
mp.Height = h;  
mp.Width = w;  
mp.GroundPlaneLength = gpL;  
mp.GroundPlaneWidth = gpW;  
figure;  
show(mp);
```



Calculate Baseline Impedance

Calculate and store the impedance and the number of triangles in the mesh when using the default mesh size.

```
Zbaseline = impedance(mp,f);
meshdata = mesh(mp);
Nbaseline = meshdata.NumTriangles;
```

Meshing Parameter Variation

You can evaluate the accuracy of the results by changing resolution of a triangular surface mesh. The triangular surface mesh implies a discretization of the surface geometry into small planar triangles. All antenna surfaces in the Antenna Toolbox™ are discretized into triangles. To specify the resolution of the mesh, provide the size of the maximum edge length, i.e. the longest side of a triangle among all triangles in the mesh, prior to analysis. Alternatively, define a range of values for the maximum edge length.

```
maxEdgeLength = gpL./(2:2:16);
```

Set up Analysis Parameters

Create arrays to save impedance, relative change in impedance and the size of the mesh.

```
m = length(maxEdgeLength);
Zin = zeros(1,m);
numTri = zeros(1,m);
```

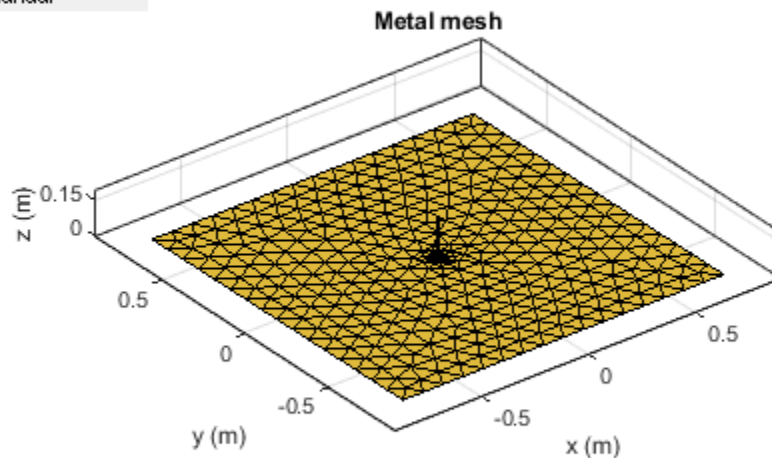
```
tolValue = .05.*ones(1,m);
tolCheck = nan*ones(1,m);
Ztemp = Zin(1);
```

Calculate Impedance Variation

For each value of the maximum edge length, update the mesh, calculate the impedance at the operating frequency and the number of triangles in the mesh. Save the impedance and number of triangles in the mesh for the convergence analysis. Finally, calculate the relative change in the impedance between subsequent mesh refinement steps.

```
for i = 1:m
    mesh(mp, 'MaxEdgeLength', maxEdgeLength(i));
    Zin(i) = impedance(mp, f);
    meshdata = meshdata(mp);
    numTri(i) = meshdata.NumTriangles;
    Zchange = abs((Zin(i)-Ztemp)/Zin(i));
    Ztemp = Zin(i);
    tolCheck(i) = Zchange;
end
```

```
NumTriangles: 712
NumTetrahedra: 0
NumBasis:
MaxEdgeLength: 0.093685
MeshMode: manual
```



Impedance of Monopole on Finer Meshes

Plot the input impedance at the operating frequency for each mesh update. Observe that the baseline values of the resistance, R_{in} and reactance, X_{in} are (for the default mesh),

$$R_{in} \approx X_{in} \approx 37 \Omega \quad 2 \Omega$$

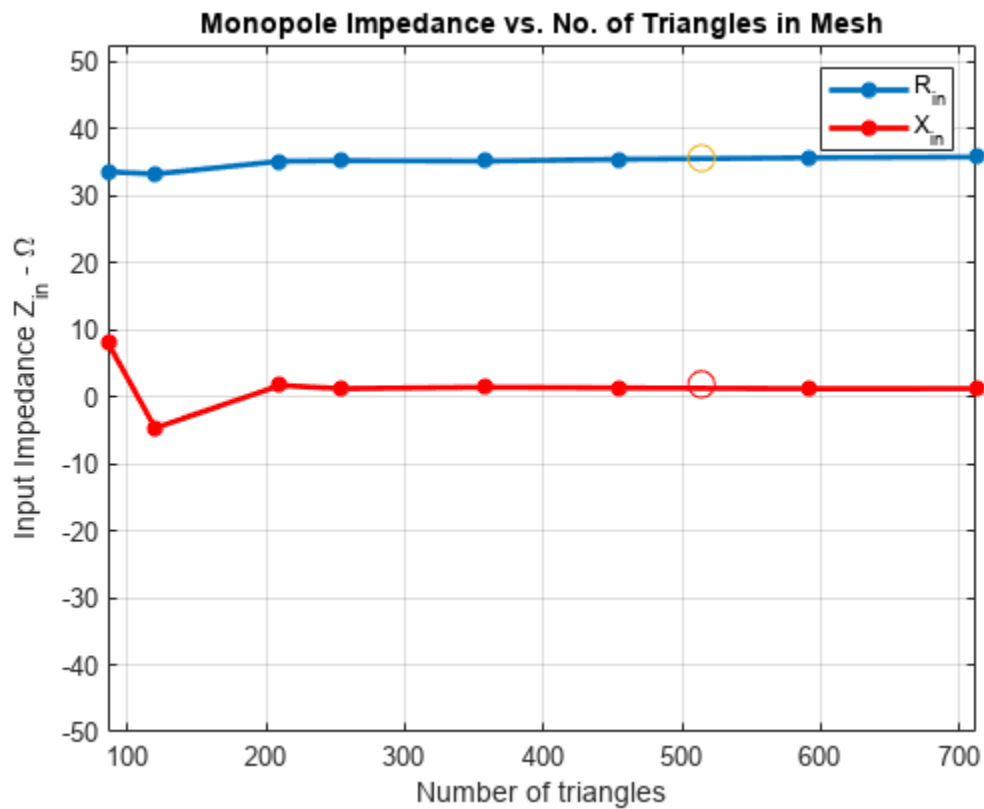
The result published in [1] for the case $a/\lambda = .001588$, $l/\lambda = 0.236$

is

$$R_{in} = 36.82 \Omega, X_{in} = 0 \Omega$$

Our result matches the resistance and suggests that a weak inductive component is present as well. The circled values for resistance and reactance are the results obtained for the default baseline mesh. Note, that the results of [1] cited above are related to cylindrical monopoles; they take into account the effect of a gap between the inner conductor and the outer conductor of the coaxial transmission line. Our geometrical model approximates the cylindrical monopole with a rectangular strip and does not account for the gap in the groundplane.

```
figure;
plot(numTri,real(Zin),'*-','LineWidth',2)
hold on
plot(numTri,imag(Zin),'r*-','LineWidth',2)
hold on
plot(Nbaseline,real(Zbaseline),'o','MarkerSize',10)
plot(Nbaseline,imag(Zbaseline),'ro','MarkerSize',10)
axis([min(numTri),max(numTri),-10*abs(round(min(imag(Zin)))),...
      1.5*floor(max(max(real(Zin),max(imag(Zin)))))]])
grid on
xlabel('Number of triangles')
ylabel('Input Impedance Z_i_n - \omega')
legend('R_i_n','X_i_n')
title('Monopole Impedance vs. No. of Triangles in Mesh')
```

**See Also**

“Analysis of Dipole Impedance” on page 5-77

“Monopole Measurement Comparison” on page 5-80

References

[1] R. Elliott, 'Antenna Theory & Design,' Chapter 8, p.353, Wiley-IEEE Press, 2003.

[2] C. A. Balanis, 'Antenna Theory. Analysis and Design,' p.514, Wiley, New York, 3rd Edition, 2005.

Analysis of Dipole Impedance

This example analyzes the impedance behavior of a center-fed dipole antenna at varying mesh resolution/size and at a single frequency of operation. The resistance and reactance of the dipole are compared with the theoretical results. A relative convergence curve is established for the impedance.

Create Thin Dipole

The default dipole has a width of 10 cm. Reduce the width to 1 cm and make the change to the dipole parameter Width.

```
mydipole = dipole;
w = 1e-2;
mydipole.Width = w;
```

Calculate Baseline Impedance

Calculate and store the impedance and the number of triangular facets in the mesh when using the default mesh. Since the dipole length is 2 m, we choose the analysis frequency as half-wavelength frequency $f = c/(2 * l)$, where c , is the speed of light.


```
c = 2.99792458e8;
f = c/(2*mydipole.Length);
Zbaseline = impedance(mydipole,f);
meshdata = mesh(mydipole);
Nbaseline = meshdata.NumTriangles;
Mbaseline = meshdata.MaxEdgeLength;
```

Define Analysis Parameters

Create parameters to store impedance, the relative change in impedance and the mesh size.

```
Zin = Zbaseline;
numTri = Nbaseline;
Ztemp = Zin(1);
```

Impact of Mesh Accuracy on Antenna Impedance

The triangular surface mesh implies a discretization of the surface geometry into small planar triangles. All antenna surfaces in the Antenna Toolbox  are discretized into triangles. You can evaluate the accuracy of the simulation results by uniformly refining the mesh. To specify the resolution of the mesh, provide the size of the maximum edge length, i.e. the longest side of a triangle among all triangles in the mesh, prior to analysis.

For each value of the maximum edge length, update the mesh, calculate the impedance at the predefined operating frequency, and calculate the number of triangles in the mesh. Store the impedance and the number of triangles in the mesh for subsequent analysis. Finally, calculate the relative change in the antenna impedance between subsequent mesh refinements until a desired convergence criterion is met.

```
exptol = .002;
tolCheck = [];
n = 1;
nmax = 12;
pretol = 1;
```

```

ShrinkFactor = 0.96;
while (n < nmax+1)&&(pretol > exptol)
    Mbaseline(n+1)=Mbaseline(n)*ShrinkFactor^(n-1);
    meshdata = mesh(mydipole,'MaxEdgeLength',Mbaseline(n+1));
    numTri(n+1) = meshdata.NumTriangles;
    % Check if mesh has changed and only then calculate impedance
    if numTri(n+1)~=numTri(n)
        Zin(n+1) = impedance(mydipole,f);
        Zchange = abs((Zin(n+1)-Ztemp)/Zin(n+1));
    else
        Zin(n+1) = Zin(n);
        Zchange = pretol;
    end
    tolCheck(n) = Zchange;           %#ok<SAGROW>
    pretol      = tolCheck(n);
    Ztemp       = Zin(n+1);
    n = n+1;
end
tolValue = exptol.*ones(1,n);
tolCheck = [nan tolCheck];

```

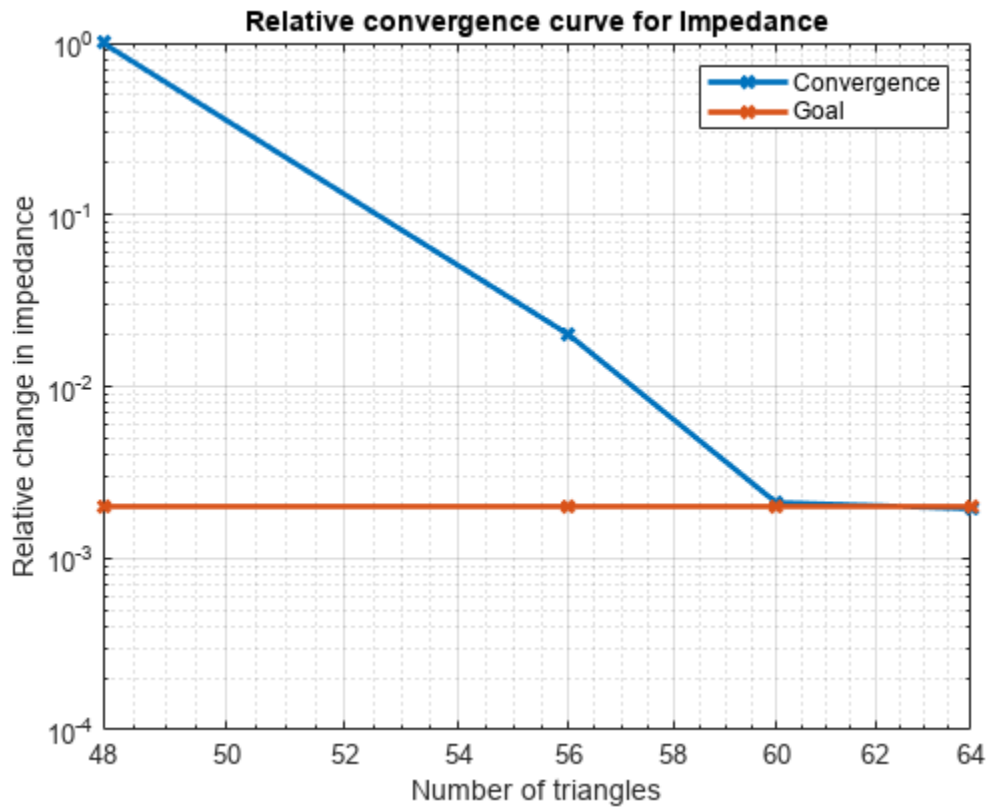
Impedance of Dipole with Finer Meshes

At the end of this analysis, the resistance, $R_{in} \approx 84\Omega$ and reactance, $X_{in} \approx 45\Omega$. This value is in good agreement with the results reported in [1], [3]. Better results are obtained with an adaptive mesh refinement algorithm, which implies selective mesh refinement in the domain of a maximum numerical error. For this case, the relative convergence curve is shown below

```

figure;
loglog(numTri,tolCheck,'-x','LineWidth',2)
hold on
loglog(numTri,tolValue,'-x','LineWidth',2)
axis([min(numTri),max(numTri),10^-4,1])
grid on
xlabel('Number of triangles')
ylabel('Relative change in impedance')
legend('Convergence','Goal')
title('Relative convergence curve for Impedance')

```



See Also

“Analysis of Monopole Impedance” on page 5-72

References

- [1] S. N. Makarov, 'Antenna and EM Modeling with MATLAB,' p.66, Wiley, 2002.
- [2] C. A. Balanis, 'Antenna Theory. Analysis and Design,' p.514, Wiley, New York, 3rd Edition, 2005
- [3] R. C. Hansen, "Carter Dipoles and Resonant Dipoles," Proceedings of the Antenna Application Symposium, Allerton Park, Monticello, IL, pp.282-284, Sep. 21-23rd 2010.

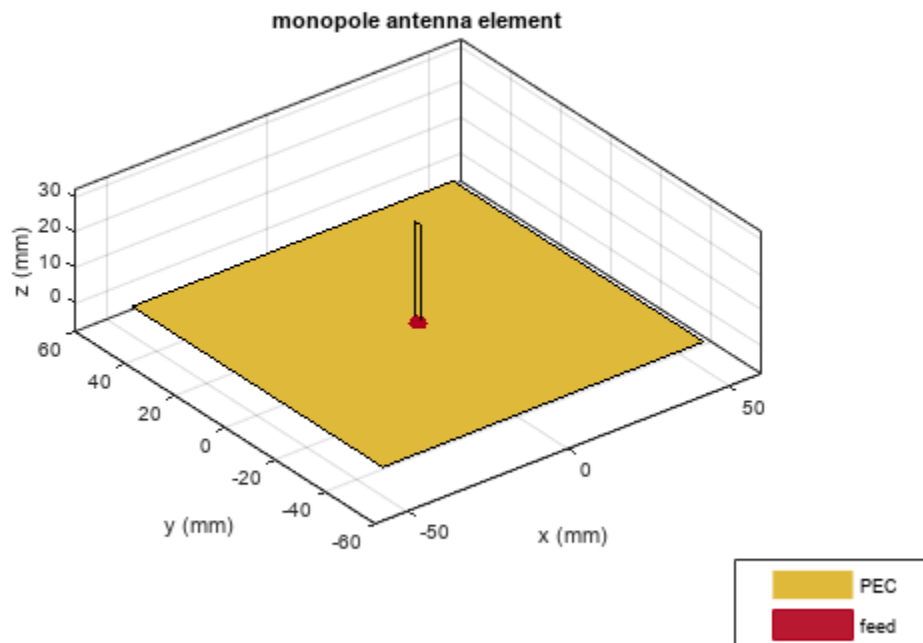
Monopole Measurement Comparison

This example compares the impedance of a monopole analyzed in Antenna Toolbox™ with the measured results. The corresponding antenna was fabricated and measured at the Center for Metamaterials and Integrated Plasmonics (CMIP), Duke University. The monopole is designed for an operating frequency of 2.5 GHz.

Create, Modify and View Monopole

Create the default monopole antenna geometry. Then, modify the height and the width of the monopole, and dimensions of the ground plane to be in agreement with the hardware prototype. Since the monopole is located at the center of the ground plane, the FeedOffset property of the antenna is not modified.

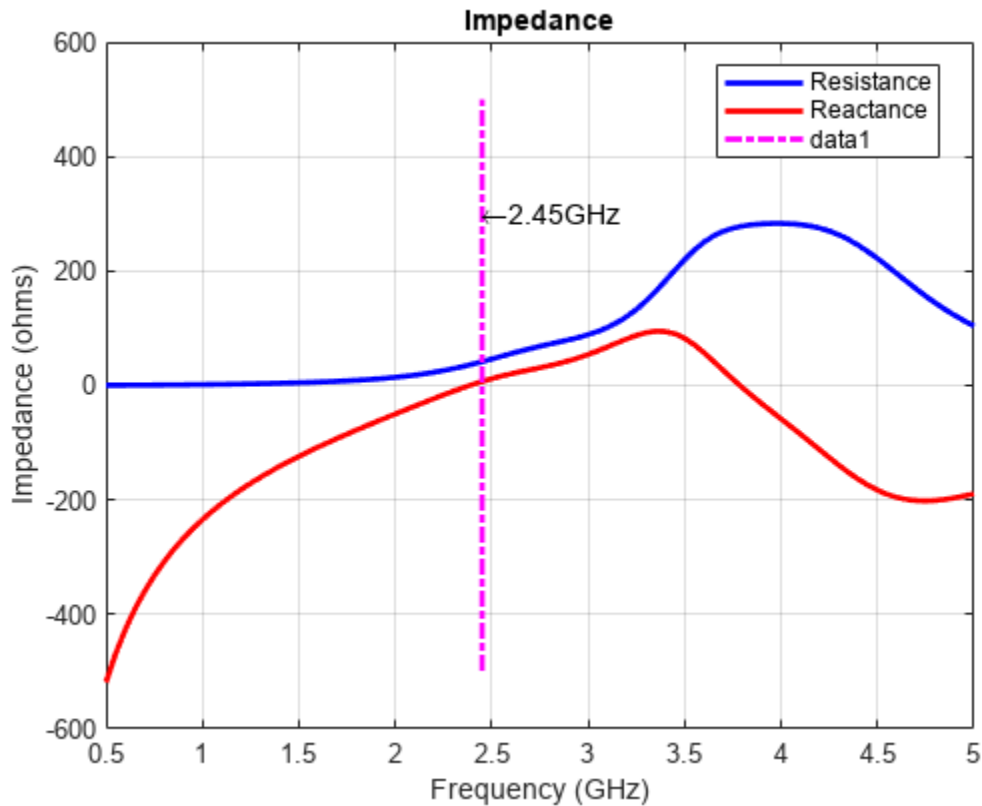
```
mp = monopole;  
mp.Height = 28.5e-3;  
mp.Width = 2.54e-3;  
mp.GroundPlaneLength = 0.1;  
mp.GroundPlaneWidth = 0.1;  
figure;  
show(mp)
```



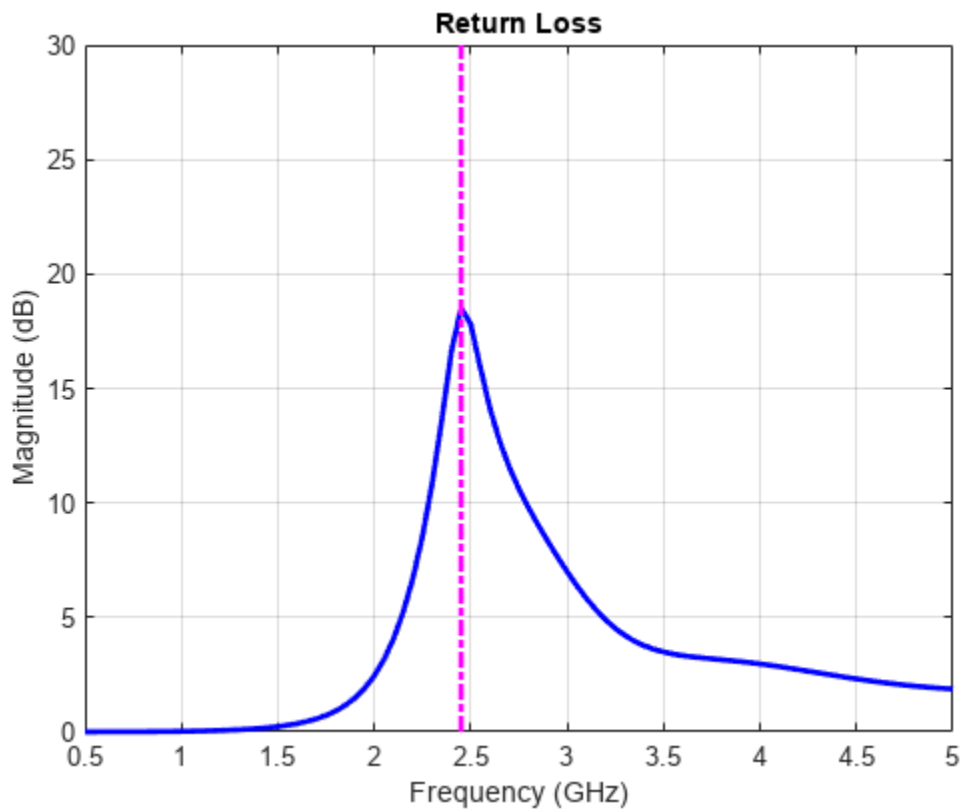
Plot Impedance and Return Loss

Define the frequency band for the analysis. The lower band frequency is 500 MHz and the upper band frequency is 5 GHz. Since the reference impedance is not specified as one of the arguments when calculating the return loss, the default value of 50 Ω is used.

```
freq = 0.5e9:50e6:5e9;
RL = returnLoss(mp,freq);
[~,ind1] = max(RL);
figure;
impedance(mp,freq);
marker1 = linspace(-500,500,21);
hold on
plot(freq(ind1).*ones(1,21)./1e9,marker1,'m-.','LineWidth',2)
textInfo = [ '\leftarrow' num2str(freq(ind1)/1e9) 'GHz'];
text(freq(ind1-1)/1e9,300,textInfo,'FontSize',11)
hold off
```



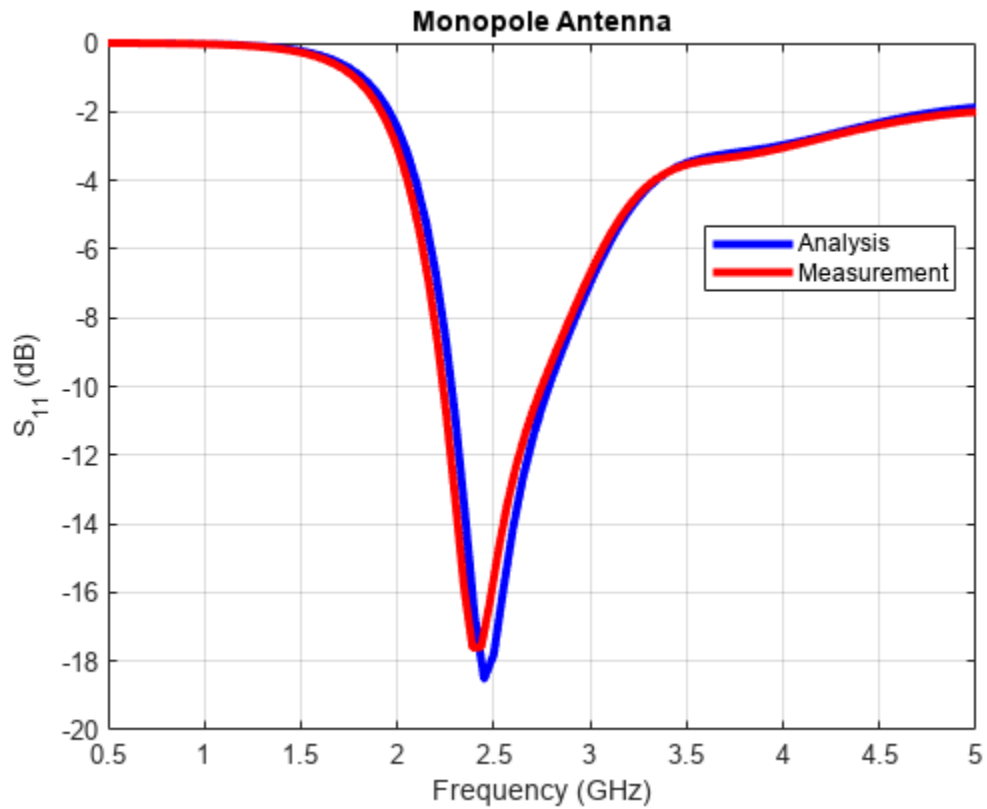
```
figure
returnLoss(mp,freq);
marker2 = linspace(0,30,21);
hold on
plot(freq(ind1).*ones(1,21)./1e9,marker2,'m-.','LineWidth',2)
hold off
```



Comparison with Measurement - Reflection Coefficient

Measurements of the fabricated monopole are taken over the same frequency band. The measured data is the reflection coefficient (S_{11}) of the monopole in decibels. To compare with measurements, plot the numerical reflection coefficient, which is equal to the negative return loss.

```
load('monopole_measured.mat');
figure;
plot(freq/1e9, -mp.returnLoss(freq), 'b', 'linewidth', 3);
hold on;
plot(f/1e9, S11dB, 'r', 'linewidth', 3);
hold off;
grid on;
legend('Analysis', 'Measurement', 'Location', 'Best');
xlabel('Frequency (GHz)');
ylabel('S_1_1 (dB)');
title('Monopole Antenna');
```

The figure indicates a good agreement between theory and measurements. Typically, a criterion such as $S_{11} < -10\text{dB}$ is used for describing a good impedance match. The analysis and measurement confirm that the monopole satisfies the criterion in a band centered about 2.5 GHz. The fabricated monopole and the measurement setup are shown below.



Fabricated monopole, and measurement setup (with permission from CMIP, Duke University)

See Also

“Monopole Measurement Comparison” on page 5-80

“Helical Antenna Design” on page 5-124

Equiangular Spiral Antenna Design Investigation

This example compares results published in [1] for a two-arm equiangular spiral antenna on foamclad backing ($\epsilon_r \approx 1$), with those obtained using the toolbox model of the spiral antenna of the same dimensions. The spiral antennas belong to the class of frequency-independent antennas. In theory, such antennas may possess an infinite bandwidth when made infinitely large. In reality, a finite feeding region has to be established and the outer extent of the spiral antenna has to be truncated.

Equiangular Spiral Antenna Parameters

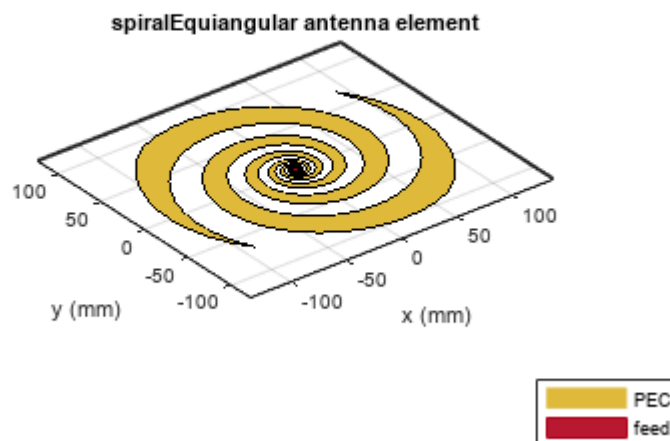
The equiangular spiral antenna is defined by a radius that grows exponentially with the winding angle. The expansion of the spiral is controlled by a factor called the growth rate. In [1], the authors use the angle between the tangent and radial vectors at any point on the spiral to define the growth rate. The inner radius of the spiral is the radius of the feeding structure, while the outer radius is the furthest extent on either arm of the spiral. Note, that the spiral arms are truncated to minimize reflections arising from the ends.

```
psi = 79*pi/180;
a = 1/tan(psi);
Ri = 3e-3;
Ro = 114e-3;
```

Create the Antenna

The parameters defined earlier are used to create an equiangular spiral antenna.

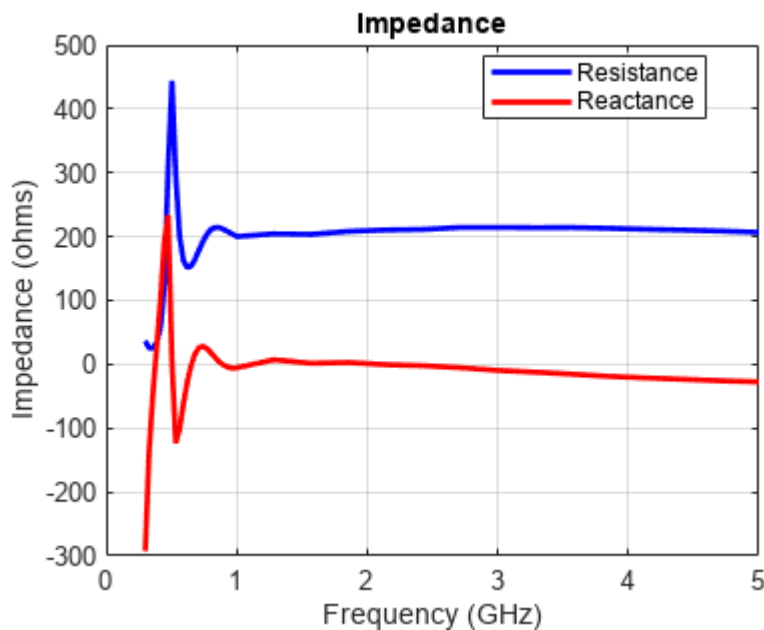
```
sp = spiralEquiangular('GrowthRate',a,'InnerRadius',Ri,'OuterRadius',Ro);
SpiralFig = figure;
show(sp)
```



Impedance Behavior of the Equiangular Spiral

The impedance behavior of the spiral antenna shows multiple resonances at the low frequency band, before achieving a relatively constant resistance and reactance with increasing frequency. To capture these resonances, split up the frequency band into two parts. Sample the lower frequency band with a finer spacing and higher frequency band with courser spacing.

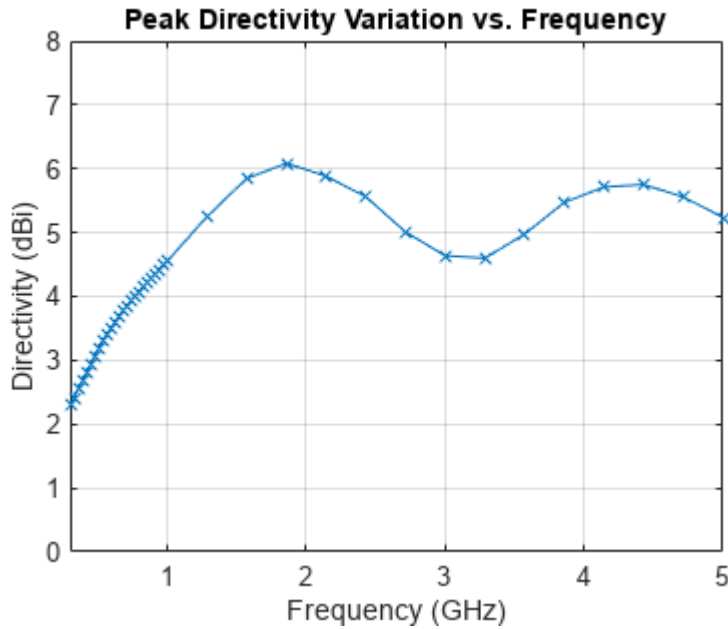
```
Nf1 = 25;
Nf2 = 15;
fband1 = linspace(0.3e9,1e9,Nf1);
fband2 = linspace(1e9,5e9,Nf2);
freq = unique([fband1,fband2]);
SpiralImpFig = figure;
impedance(sp,freq);
```



Boresight Directivity Variation with Frequency

Spiral antennas by themselves are bidirectional radiators. To suppress unwanted radiation, they are used with a ground plane and dielectric backing. The toolbox model of the equiangular spiral does not have a ground plane or the backing material.

```
SpiralDVarFig = figure;
D = zeros(1,length(freq));
for p = 1:length(freq)
    D(p) = pattern(sp,freq(p),0,90);
end
f_eng = freq./1e9;
f_str = 'G';
plot(f_eng,D,'x-')
grid on
axis([f_eng(1) f_eng(end) 0 8 ])
xlabel(['Frequency (' f_str 'Hz)'])
ylabel('Directivity (dBi)')
title('Peak Directivity Variation vs. Frequency')
```



Discussion on Results

Paper [1], compares two types of backing - Foamclad, and Rogers substrate. Since Foamclad, has relative permittivity nearly equivalent to free space, we use this for comparing results with the metal-only spiral from the toolbox. The Foamclad-backed spiral in [1] achieves a nearly constant resistance of 188Ω after 1 GHz and the reactance varies between approximately $10 - 20 \Omega$. This result agrees very well with the equiangular spiral model from the toolbox. The boresight directivity for the toolbox model of the equiangular spiral varies between 4.5 - 6 dB between 1-5 GHz. This also matches well with the results from [1].

Reference

[1] M. McFadden, W. R. Scott, "Analysis of the Equiangular Spiral Antenna on a Dielectric Substrate," IEEE Transactions on Antennas and Propagation, vol.55, no.11, pp.3163-3171, Nov. 2007.

See Also

"Reflector Backed Equiangular Spiral" on page 5-129

Archimedean Spiral Design Investigation

This example compares the results published in [1] for an Archimedean spiral antenna with those obtained using the toolbox model of the spiral antenna. The two-arm Archimedean spiral antenna ($r = R\phi$) can be regarded as a dipole, the arms of which have been wrapped into the shape of an Archimedean spiral. This idea came from Edwin Turner around 1954.

Archimedean Spiral Antenna Parameters

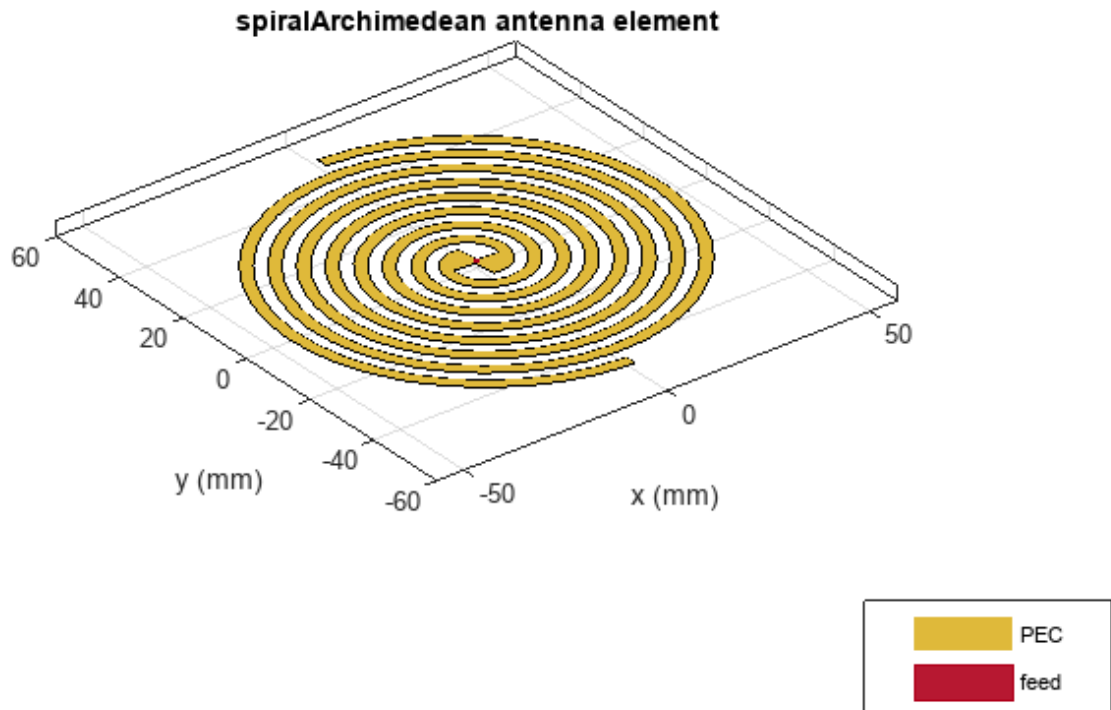
The Archimedean spiral antenna can be classified as a *frequency-independent antenna* in the sense that its input impedance and gain remain almost constant throughout the bandwidth. At low frequencies, the radiation zone is near the outermost part of the spiral, meanwhile at high frequencies it is near the center. Hence, the lowest cutoff frequency of the spiral antenna is related to its outer radius and the highest cutoff frequency is related to its inner radius. This means that the bandwidth of the antenna can be very large, only depending on size and printing accuracy. Below are the dimensions of the spiral given in [1] on page 5-94 in meters.

```
Ro    = 50e-3;  
Ri    = 5.5e-3;  
turns = 4;
```

Create and View Archimedean Spiral Antenna

Create an Archimedean spiral antenna using the parameters defined.

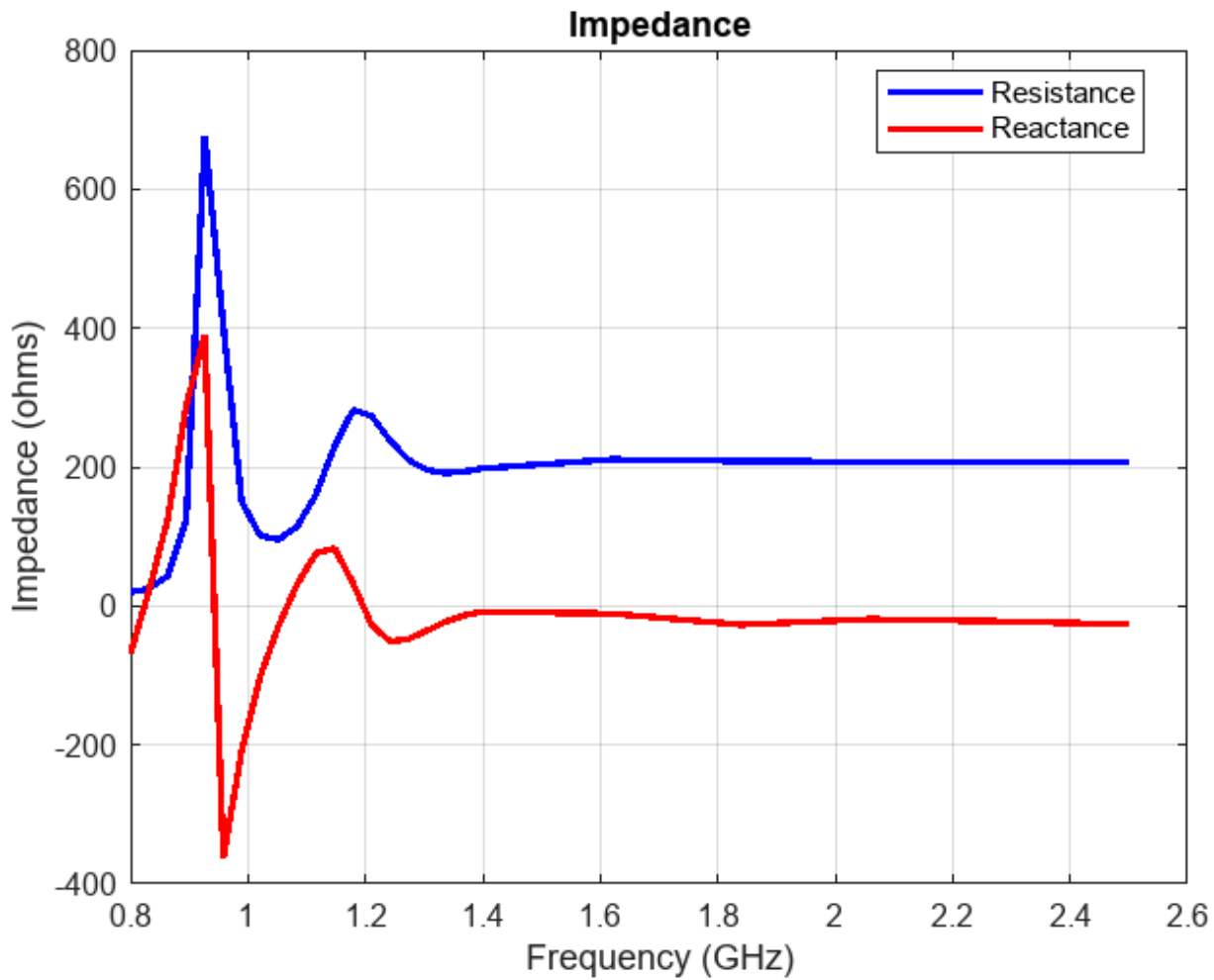
```
sp = spiralArchimedean('Turns',turns,'InnerRadius',Ri,'OuterRadius',Ro);  
figure;  
show(sp);
```



Impedance Behavior of Archimedean Spiral Antenna

The impedance behavior of the spiral antenna shows multiple resonances at the low frequency band, before achieving a relatively constant resistance and reactance behavior. To capture these resonances we split the entire frequency band into two subbands. In the lower-frequency band, we sample with a finer spacing, while at the higher frequencies we opt for coarser spacing.

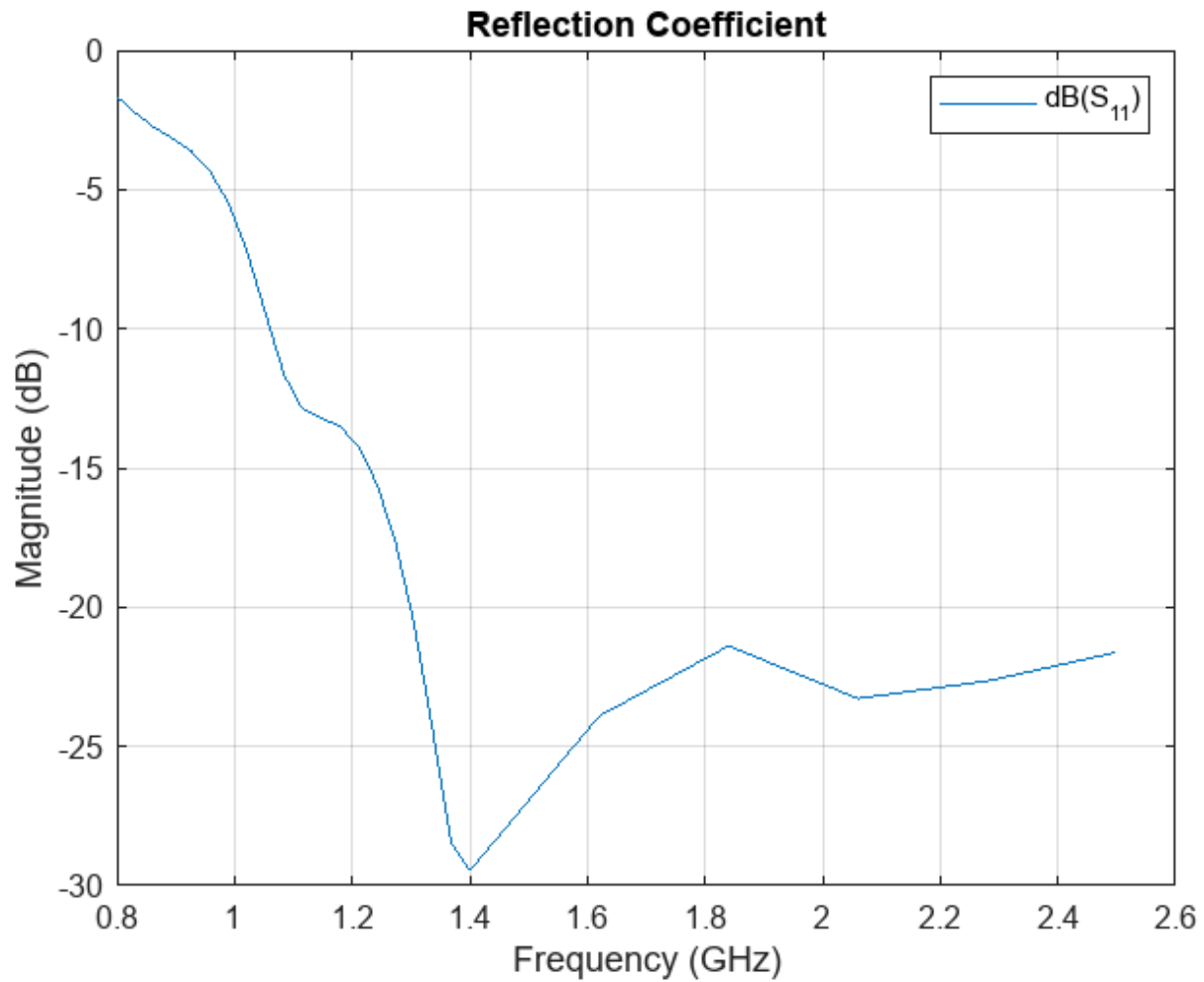
```
Nf1    = 20;
Nf2    = 6;
fband1 = linspace(0.8e9,1.4e9,Nf1);
fband2 = linspace(1.4e9,2.5e9,Nf2);
freq   = unique([fband1,fband2]);
figure;
impedance(sp, freq);
```



Reflection Coefficient

The input impedance of this spiral antenna can be obtained using the Booker's extension of the Babinet's principle for complementary structures [2]. For an antenna in free space, its input impedance equals $60\pi = 188.5 \Omega$. This value is very close to the value of the impedance observed at the higher frequencies in the plot seen above. The reflection coefficient is calculated using the reference impedance of 188Ω . From the plot below it is seen that the present antenna is well matched for frequencies higher than 1.1 GHz.

```
S = sparameters(sp, freq, 188);
figure; rfplot(S);
title('Reflection Coefficient');
```

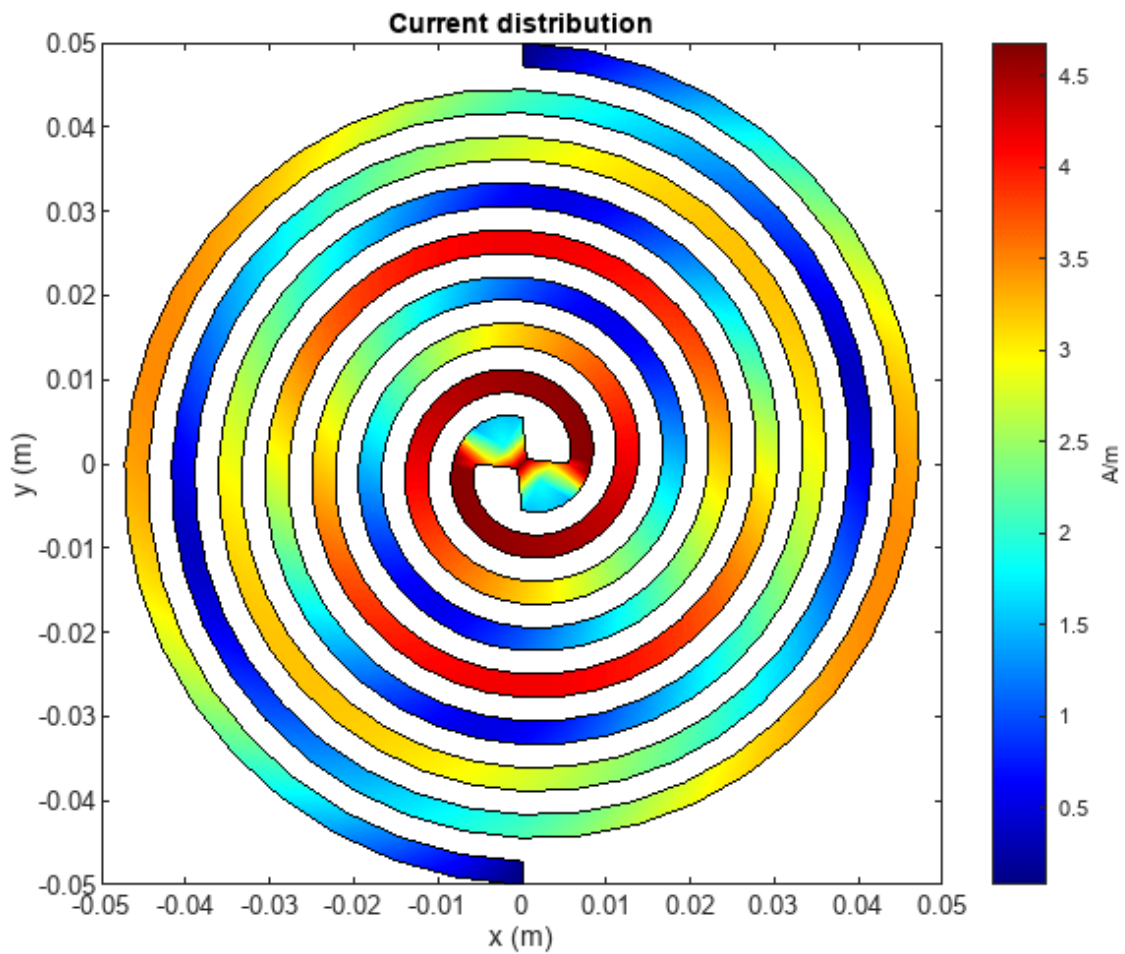



This matched frequency is higher than the approximate theoretical result of 0.96 GHz [1] since the current still reaches spiral tips as seen in the computational figure that follows.

Current Distribution

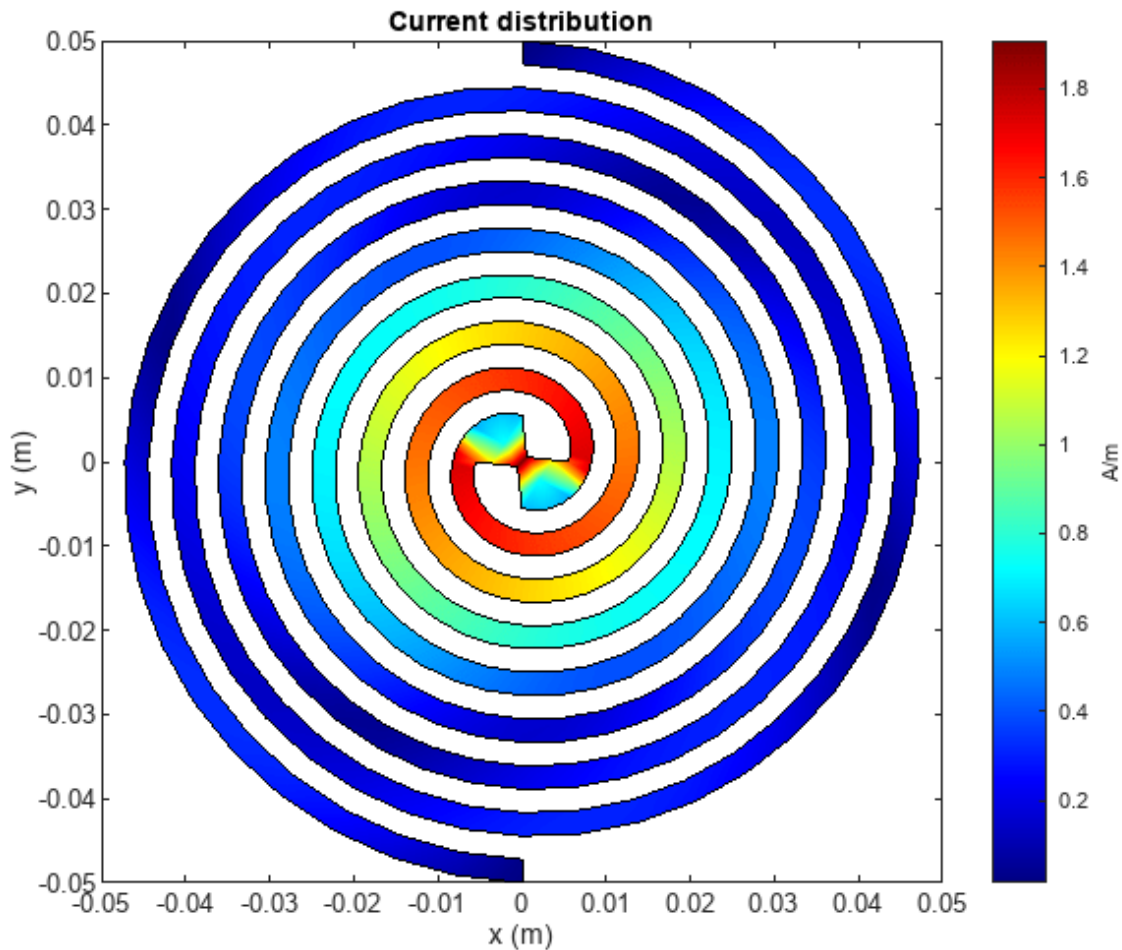
The surface current distribution at low frequencies is:

```
figure;  
current(sp, 0.85e9);  
view(0,90);
```



At higher frequencies, surface currents decay long before reaching the spiral tips. This results in a better matching.

```
figure;  
current(sp, 1.9e9);  
view(0,90);
```



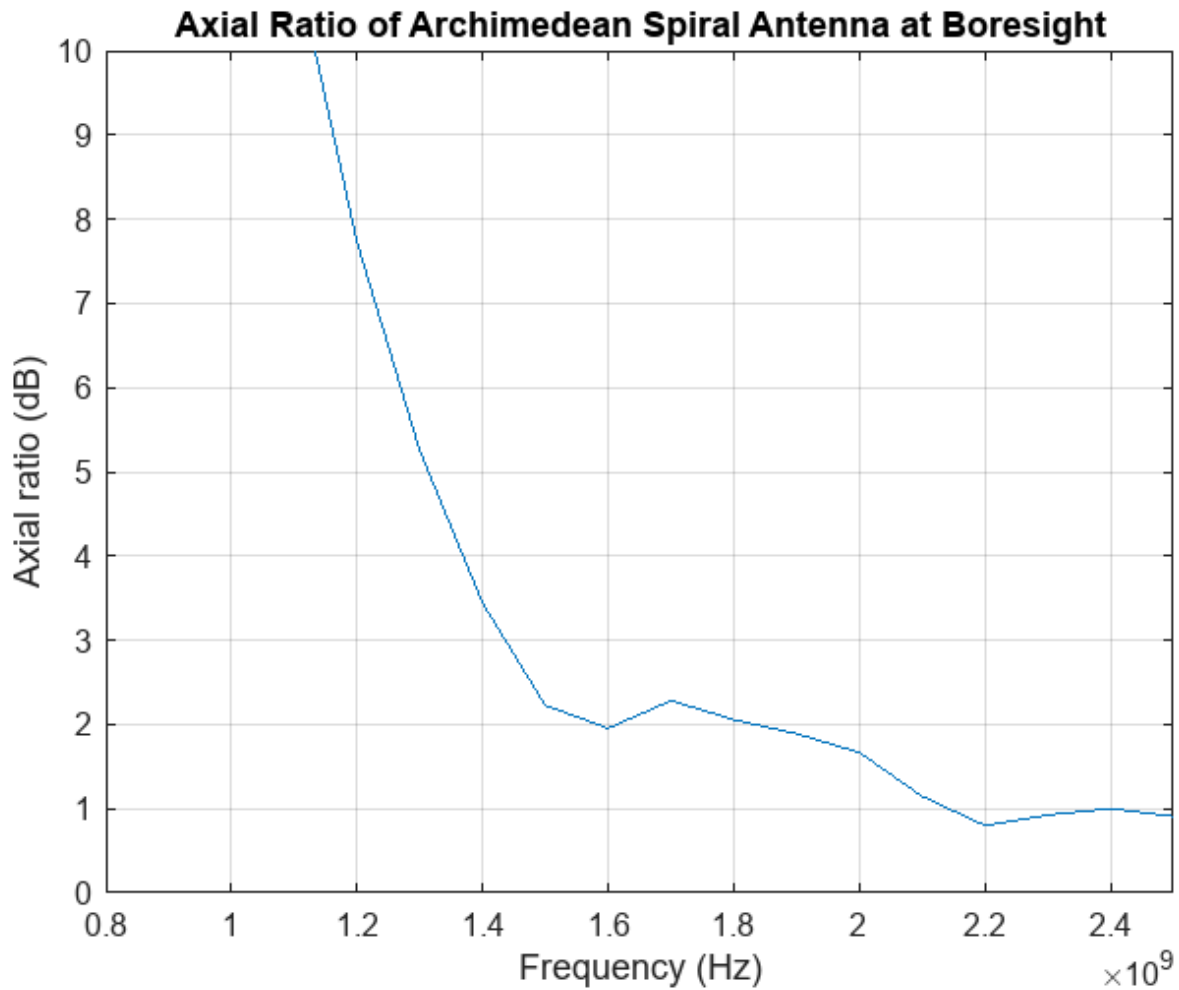
Axial Ratio

The current distribution presents a 180-degree rotational symmetry, which produces a circularly polarized radiated wave. The plot below shows the axial ratio at broadside of the spiral antenna. Observe that the spiral antenna achieves a good circular polarization for frequencies higher than 1.4 GHz.

```

freq = 0.8e9:100e6:2.5e9;
AR = zeros(size(freq));
for m=1:numel(freq)
    AR(m) = axialRatio(sp, freq(m), 0, 90);
end
figure;
plot(freq, AR);
grid on;
axis([0.8e9 2.5e9 0 10]);
xlabel('Frequency (Hz)');
ylabel('Axial ratio (dB)');
title('Axial Ratio of Archimedean Spiral Antenna at Boresight');

```



Discussion on the Results

Thesis [1] on page 5-94 designs an Archimedean spiral antenna and compares its performance to theoretical results as well as to the commercially available EM solvers. The results obtained using the Antenna Toolbox™ match very well with the results presented in [1].

Reference

[1] Israel Hinostrroza, "Conception de reseaux large bande d'antennes spirales", Other, Supelec, 2013, pp. 58-62. Online at: https://theses.hal.science/file/index/docid/830469/filename/Hinostrroza_Israel_final_final_thesis_2013.pdf

[2] C. A. Balanis, Antenna Theory. Analysis and Design, Wiley, New York, 3rd Edition, 2005.

See Also

"Modeling Resonant Coupled Wireless Power Transfer System" on page 5-209

Patch Antenna on Dielectric Substrate

This example calculates the performance of the two linearly-polarized rectangular patch antennas with results published in [1]. The first antenna has a low-epsilon thin dielectric substrate while the second antenna has a thick high-epsilon dielectric substrate.

Create a Patch Antenna on Low-Epsilon Thin Substrate

Create a rectangular patch antenna with a length of 40 mm and a width of 30 mm on a 80 mm x 60 mm ground plane. The lossless substrate has a dielectric constant of 2.33 and thickness of 1.57mm. The feed is offset by 5.5 mm from the origin along the x-axis.

```
p1 = patchMicrostrip;
p1.Length = 40e-3;
p1.Width = 30e-3;
p1.Height = 1.57e-3;
p1.GroundPlaneLength = 80e-3;
p1.GroundPlaneWidth = 60e-3;
p1.FeedOffset = [5.5e-3 0];
```

Antenna Toolbox™ has a list of substrates supported as part of its dielectric catalog. To open the catalog use the following command.

```
openDielectricCatalog
```

	Name	Relative_Permittivity	Loss_Tangent	Frequency	Comments
1	Air	1	0	1.0000e+...	
2	FR4	4.8000	0.0260	100.0000e...	
3	Teflon	2.1000	2.0000e-04	100.0000e...	
4	Foam	1.0300	1.5000e-04	50.0000e...	
5	Polystyrene	2.5500	1.0000e-04	100.0000e...	
6	Plexiglas	2.5900	0.0068	10.0000e...	
7	Fused qu...	3.7800	1.0000e-04	10.0000e...	
8	E glass	6.2200	0.0023	100.0000e...	
9	RO4725JXR	2.5500	0.0022	2.5000e+...	
10	RO4730JXR	3	0.0023	2.5000e+...	

The substrate specified for this patch is not part of the dielectric catalog. You can add it to the catalog if desired, so that the material can be used next time without specifying its electrical properties. You can also specify the substrate property directly as well as shown below.

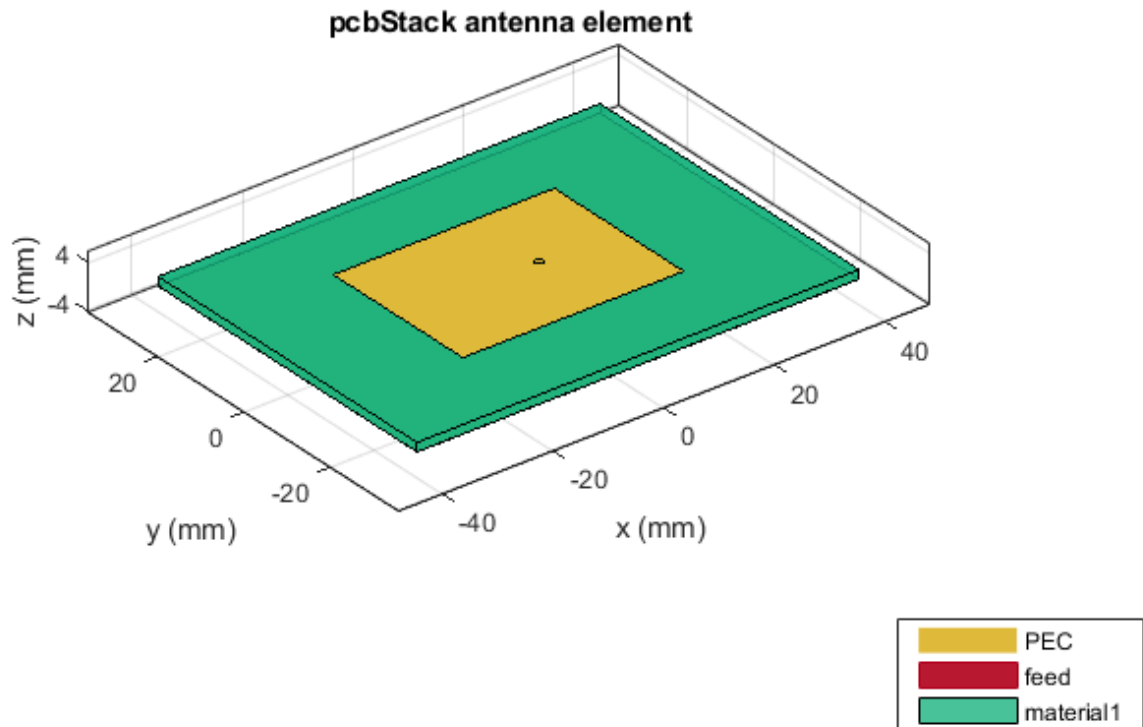
```
p1.Substrate = dielectric('Name','material1','EpsilonR',2.33);
```

As indicated in the reference [1], the feed is modeled as a square column of side 1mm. This feed model is available in the pcbStack. Convert the patch model to the stack representation and model the feed.

```
pb1 = pcbStack(p1);
pb1.FeedDiameter = sqrt(2)*1e-3;
pb1.FeedViaModel = 'square'

pb1 =
    pcbStack with properties:
        Name: 'Probe-fed rectangular microstrip patch'
        Revision: 'v1.0'
        BoardShape: [1x1 antenna.Rectangle]
        BoardThickness: 0.0016
        Layers: {[1x1 antenna.Rectangle] [1x1 dielectric] [1x1 antenna.Rectangle]}
        FeedLocations: [0.0055 0 1 3]
        FeedDiameter: 0.0014
        ViaLocations: []
        ViaDiameter: []
        FeedViaModel: 'square'
        FeedVoltage: 1
        FeedPhase: 0
        Conductor: [1x1 metal]
            Tilt: 0
            TiltAxis: [1 0 0]
            Load: [1x1 lumpedElement]
```

```
figure
show(pb1)
```

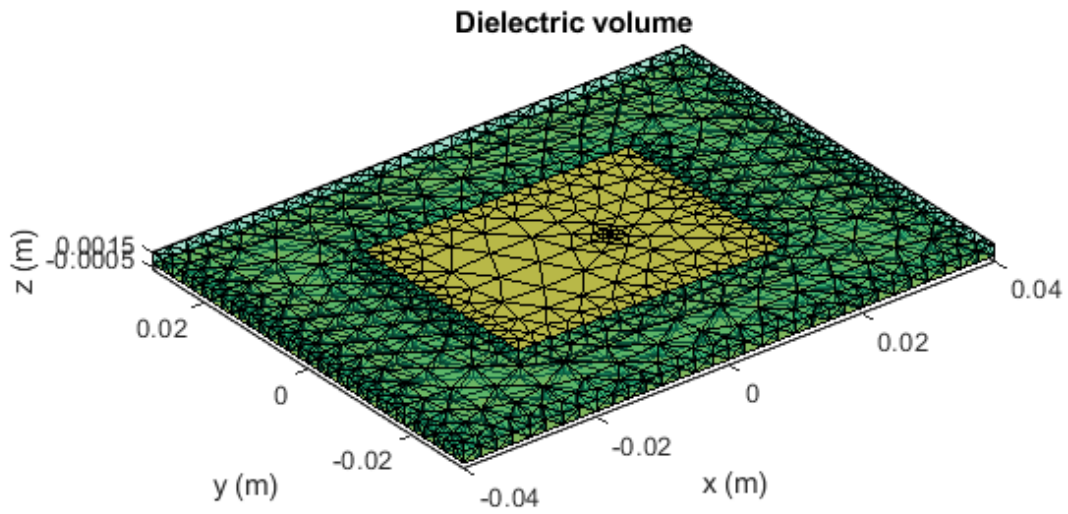


Visualize the Thin Patch Antenna Performance

Mesh the structure by specifying the maximum and the minimum edge lengths. Below is the mesh used to model the antenna. The triangles are used to discretize the metal regions of the patch, and tetrahedra are used to discretize the volume of the dielectric substrate in the patch. These are indicated by the colors yellow and green respectively. The total number of unknowns is the sum of the unknowns for the metal plus the unknowns used for the dielectric. As a result, the time to calculate the solution increases significantly as compared to pure metal antennas.

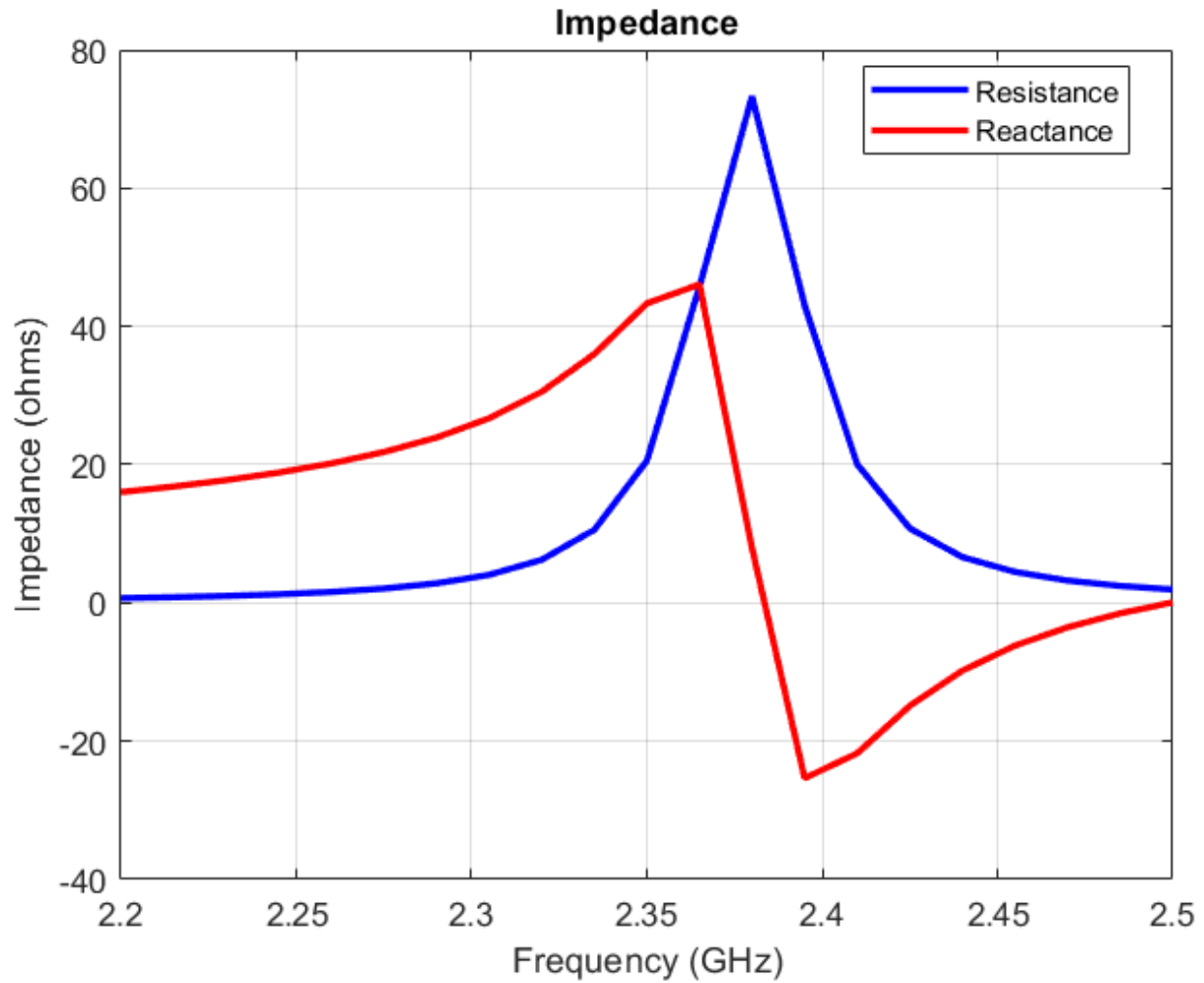
```
figure  
mesh(pb1, 'MaxEdgeLength', .01, 'MinEdgeLength', .003)
```

NumTriangles: 1364
NumTetrahedra: 3210
NumBasis:
MaxEdgeLength: 0.01
MeshMode: manual



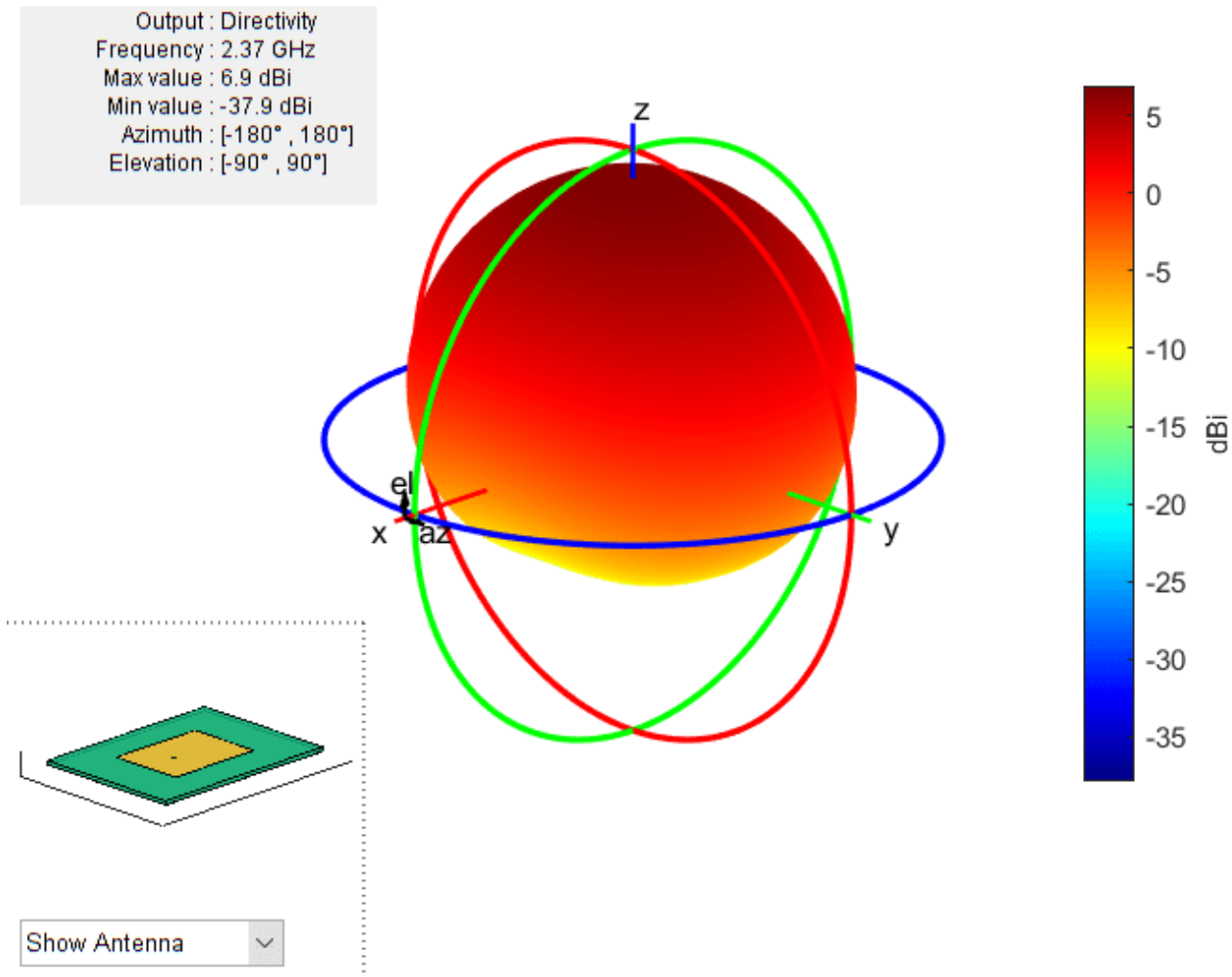
The impedance of the antenna shows a resonance at 2.37 GHz. This value is very close to the results published in the paper.

```
figure  
impedance(pb1, linspace(2.2e9, 2.5e9, 21))
```

The antenna pattern at resonance also shows uniform illumination above the ground plane with minimum leakage below it.

```
figure  
pattern(pb1, 2.37e9)
```



Modelling Thin vs. Thick Dielectric Substrates

Thickness of a dielectric substrate is measured with respect to the wavelength. In the case above the wavelength in free-space is about 126 mm. The wavelength in dielectric is approximated by dividing the above number by the square root of the dielectric constant. This value is about 85mm. So the substrate thickness is about 1/50th of the wavelength in dielectric. This is a thin substrate.

The next case considers a patch antenna on a substrate whose thickness is 1/10th of the wavelength in dielectric. This is a thick substrate. A method of moments solver requires at least 10 elements per wavelength to give an accurate solution. So if the dielectric thickness becomes greater than this, the solution accuracy is affected. In Antenna Toolbox if the substrate thickness is greater than 1/10th of the wavelength in dielectric, manual meshing is recommended for solving the antenna. As the 10 element per wavelength criteria is not satisfied, a certain amount of error in the solution is expected.

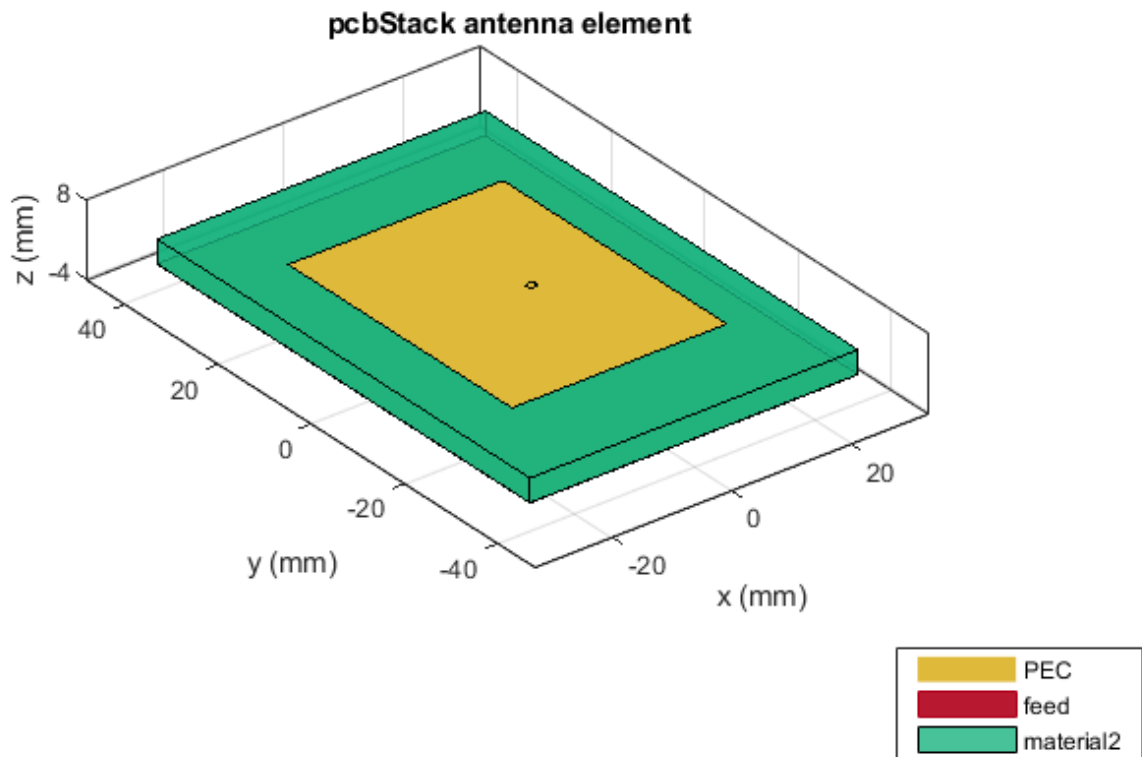
Create a Patch Antenna on a High-Epsilon Thick Substrate

Create a rectangular patch antenna with a length of 36 mm and a width of 48 mm on a 55 mm x 80 mm ground plane. The lossless substrate has a dielectric constant of 9.29 and thickness of 3.82mm. The feed is offset by 4 mm from the origin along the x-axis. Similar to the previous case, convert the model to the stack representation and change the feed model as noted in [1].

```

p2 = patchMicrostrip;
p2.Length = 36e-3;
p2.Width = 48e-3;
p2.Height = 3.82e-3;
p2.GroundPlaneLength = 55e-3;
p2.GroundPlaneWidth = 80e-3;
p2.FeedOffset = [4.0e-3 0];
p2.Substrate = dielectric('Name','material2','EpsilonR',9.29);
pb2 = pcbStack(p2);
pb2.Layers{1}.NumPoints = 40;
pb2.Layers{3}.NumPoints = 40;
pb2.FeedDiameter = sqrt(2)*1e-3;
pb2.FeedViaModel = 'square';
figure
show(pb2)

```

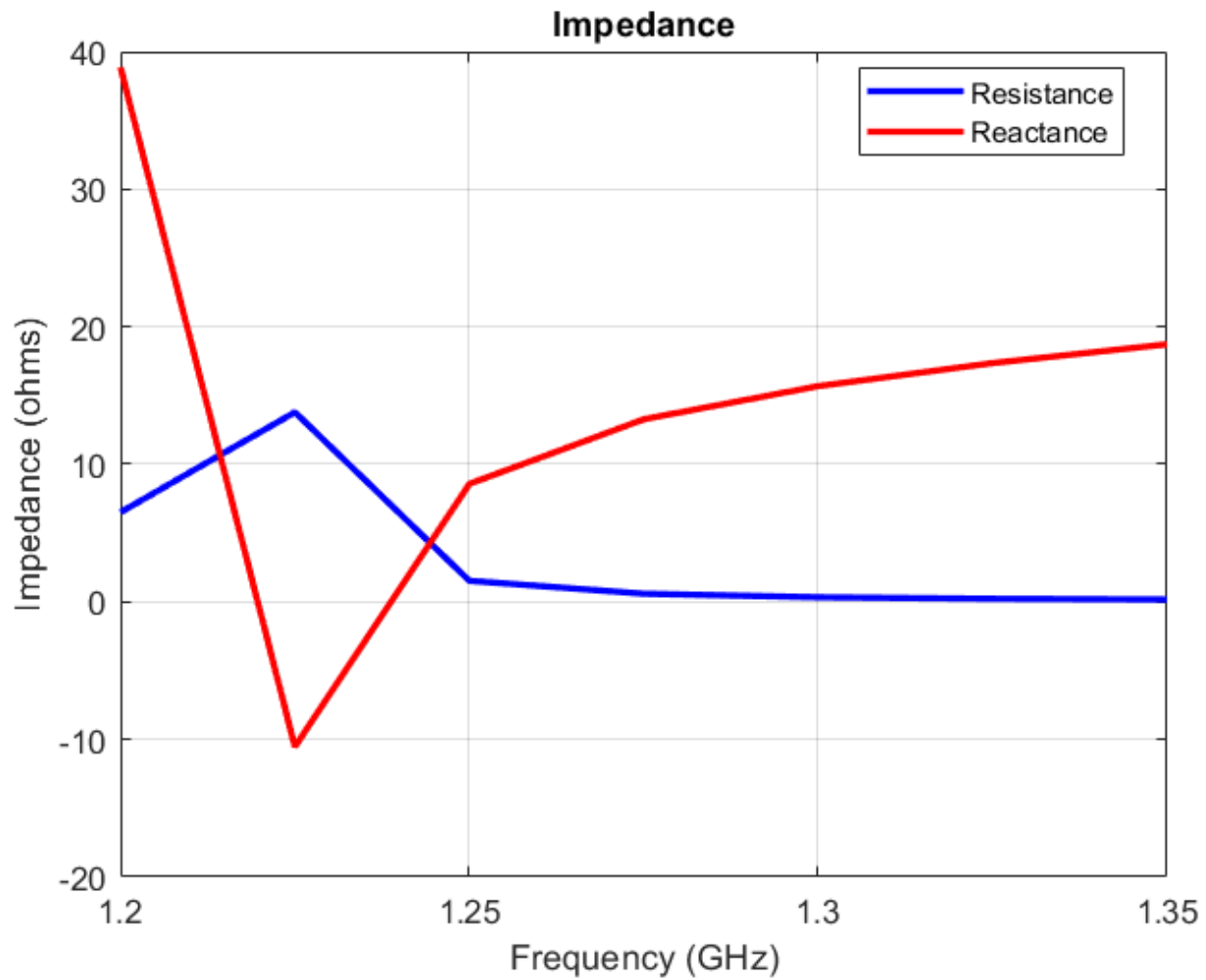


Visualize the Thick Patch Antenna Performance

This is perhaps the most complicated case from the numerical point of view - a thick high-epsilon dielectric with significant fringing fields close to the patch edges[1]. The figure below shows the

impedance plot of the antenna with resonance close to 1.22 GHz. The paper shows a resonance close to 1.27 GHz.

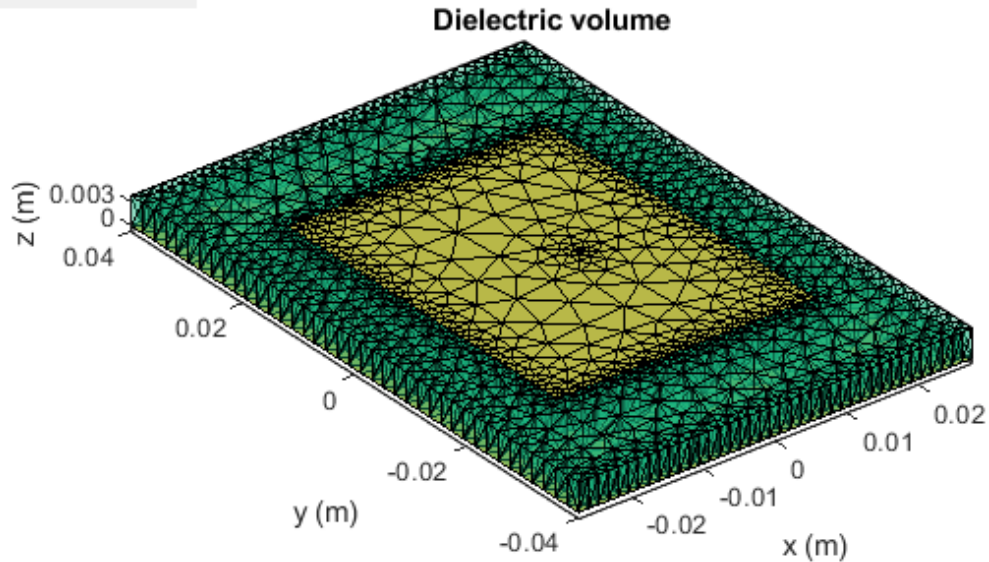
```
figure  
impedance(pb2,linspace(1.2e9,1.35e9,7))
```



Below is the mesh used for calculating the antenna performance.

```
figure  
mesh(pb2)
```

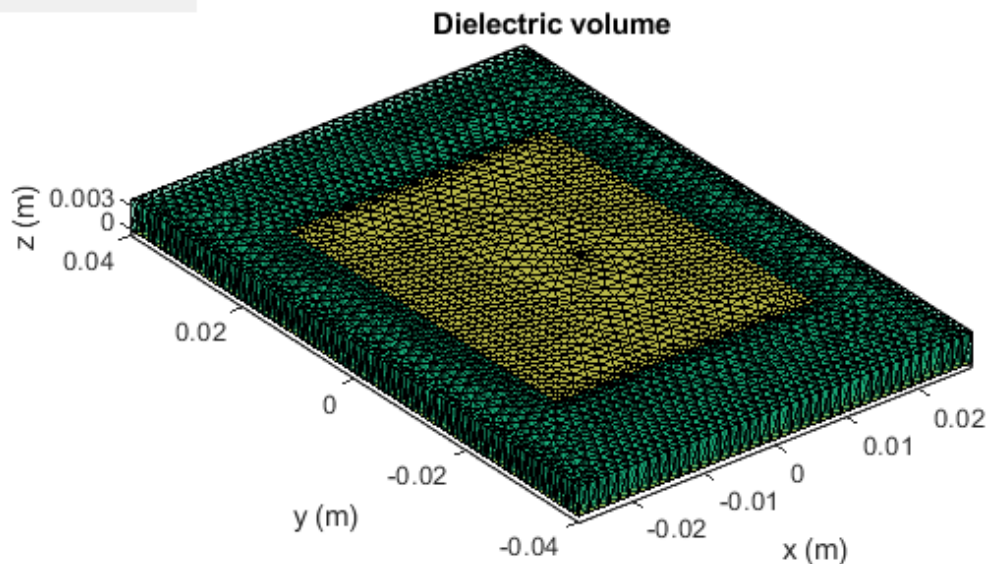
```
NumTriangles: 2880  
NumTetrahedra: 6312  
NumBasis: 12736  
MaxEdgeLength: 0.050638  
MeshMode: auto
```



To improve the result refine the mesh. The mesh can be refined using the maximum edge length criteria. In the case below the maximum edge length is set to 1.65 mm. As can be seen below, over 13000 tetrahedra are generated. As the mesh is very fine, the results take a longer time to compute and hence are saved in a .mat file.

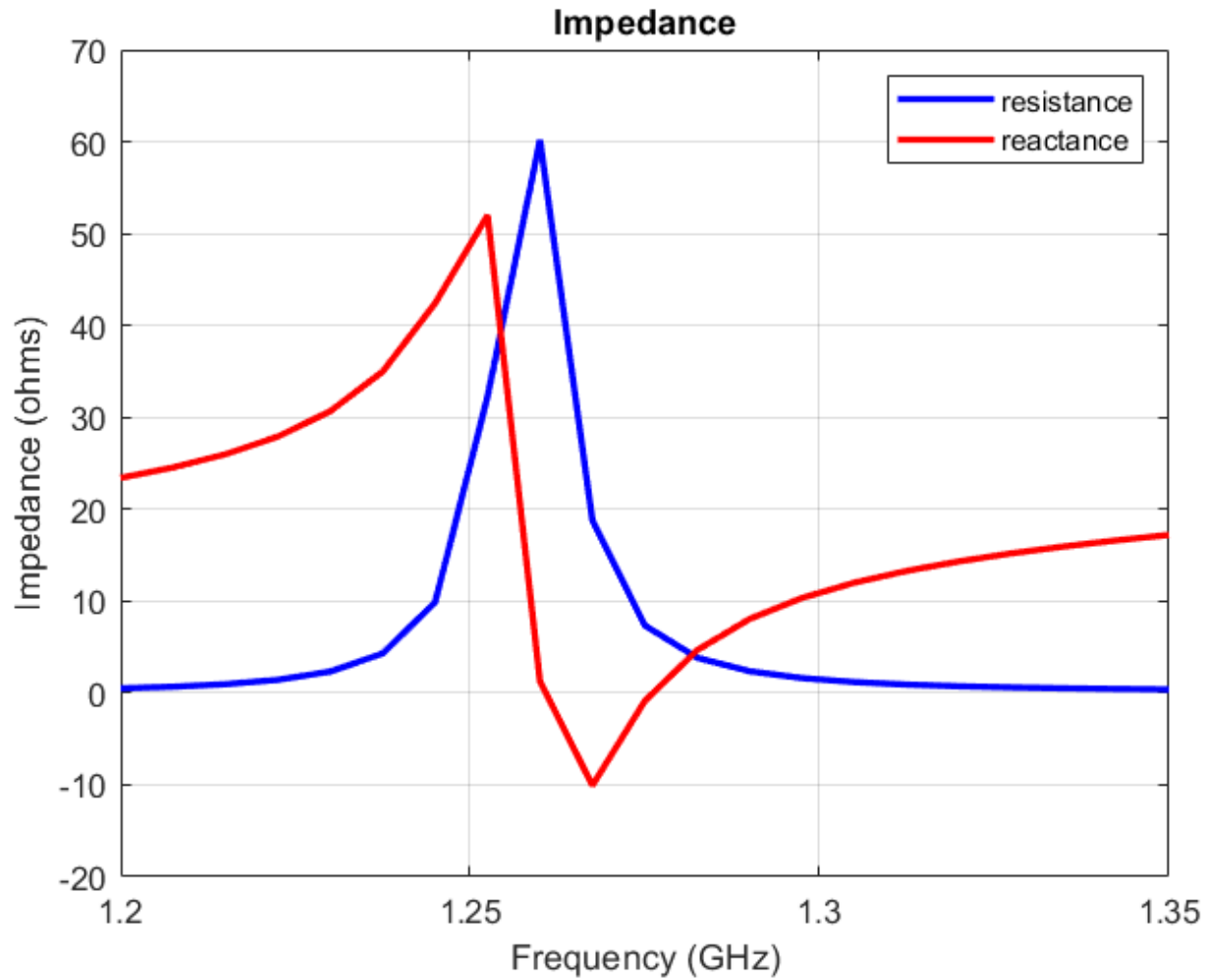
```
figure  
mesh(pb2, 'MaxEdgeLength', .00165)
```

```
NumTriangles: 6312  
NumTetrahedra: 13644  
NumBasis:  
MaxEdgeLength: 0.00165  
MeshMode: manual
```



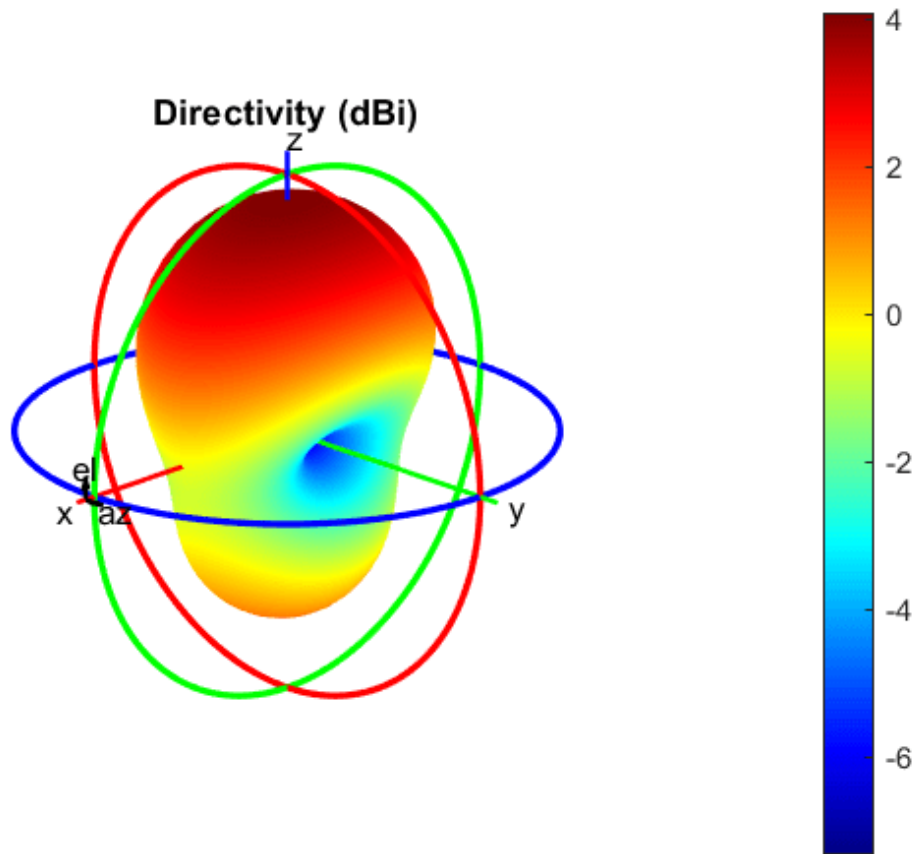
Plotting the impedance shows the resonance close to 1.27 GHz as expected.

```
load thickpatch  
figure  
plot(freq*1e-9, real(Z), 'b', freq*1e-9, imag(Z), 'r', 'LineWidth',2);  
legend('resistance', 'reactance');  
title('Impedance');  
ylabel('Impedance (ohms)');  
xlabel('Frequency (GHz)');  
grid on;
```



The directivity information for this patch is also precomputed and stored. This can be plotted using the `patternCustom` function as shown below. A significant back lobe is observed for the thick patch antenna.

```
figure
patternCustom(D.', 90-el, az);
h = title('Directivity (dBi)');
h.Position = [-0.4179, -0.4179, 1.05];
```



Reference

[1] S. N. Makarov, S. D. Kulkarni, A. G. Marut and L.C. Kempel 'Method of Moments Solution for a Printed Patch/Slot Antenna on a Thin Finite Dielectric Substrate Using the Volume Integral Equation', IEEE Trans. on Antenna and Propagation, vol. 54, No. 4, April 2006, pp 1174- 1184.

See Also

"FMCW Patch Antenna Array" on page 5-172 | "Modeling and Analysis of Probe-Fed Stacked Patch Antenna" on page 5-395

Sector Antenna for 2.4 GHz Wi-Fi™

This example models the inverted Amos sector antenna designed in [1]. A sector antenna is a type of directional antenna with a sector-shaped radiation pattern. The word 'sector' is used here in the geometric sense as a portion of the circumference of a circle measured in degrees.

Inverted Amos Sector Antenna

The antenna consists of a meander dipole (a Franklin antenna) backed by a reflector. The gain of the antenna depends on the number of vertically stacked dipoles in the meander. The current design will use 7 stacked dipoles as shown in Fig.5 in [1]. All other dimensions follow [1].

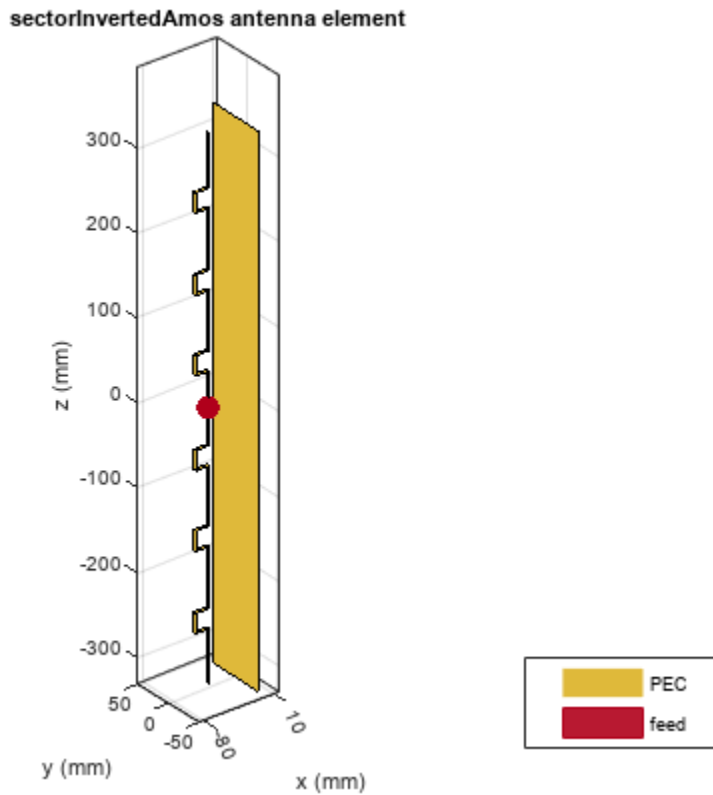
```
dipolearms = [88e-3 71e-3 73e-3 65e-3];
wirewidth  = cylinder2strip(1e-3);
notchL     = 23.8e-3;
notchW     = 17e-3;
spacing    = 35.5e-3;
GP_length  = 660e-3;
GP_Width   = 75e-3;
```

Paper [1] uses the dipole length of 84 mm with a 4 mm feeding gap. Antenna Toolbox uses the delta gap feed model. Add the feeding gap length to the length of the dipole resulting in the total length of 88 mm.

Create Antenna

Create the inverted amos sector antenna.

```
sector = sectorInvertedAmos('ArmWidth', wirewidth, 'ArmLength', dipolearms, ...
    'NotchLength', notchL, 'NotchWidth', notchW, 'Spacing', spacing, ...
    'GroundPlaneLength', GP_length, 'GroundPlaneWidth', GP_Width);
figure;
show(sector);
```



Mesh Structure

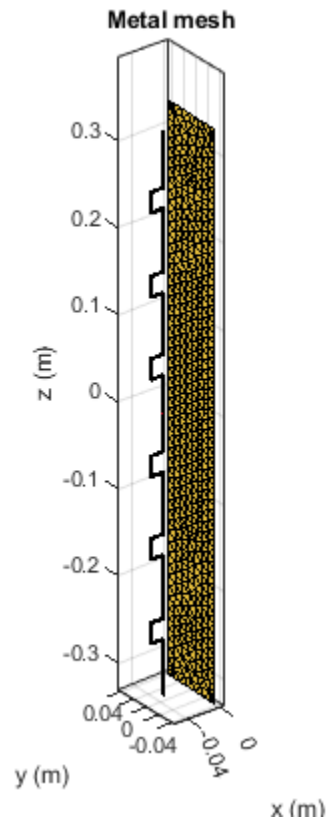
The antenna will operate between 2.4 GHz and 2.5 GHz. We manually mesh the structure by using at least 10 elements per wavelength at the highest frequency of the band.

```
lambda = 3e8/2.5e9;  
figure;  
mesh(sector, 'MaxEdgeLength', lambda/10);
```

```

NumTriangles: 1118
NumTetrahedra: 0
NumBasis:
MaxEdgeLength: 0.012
MeshMode: manual

```



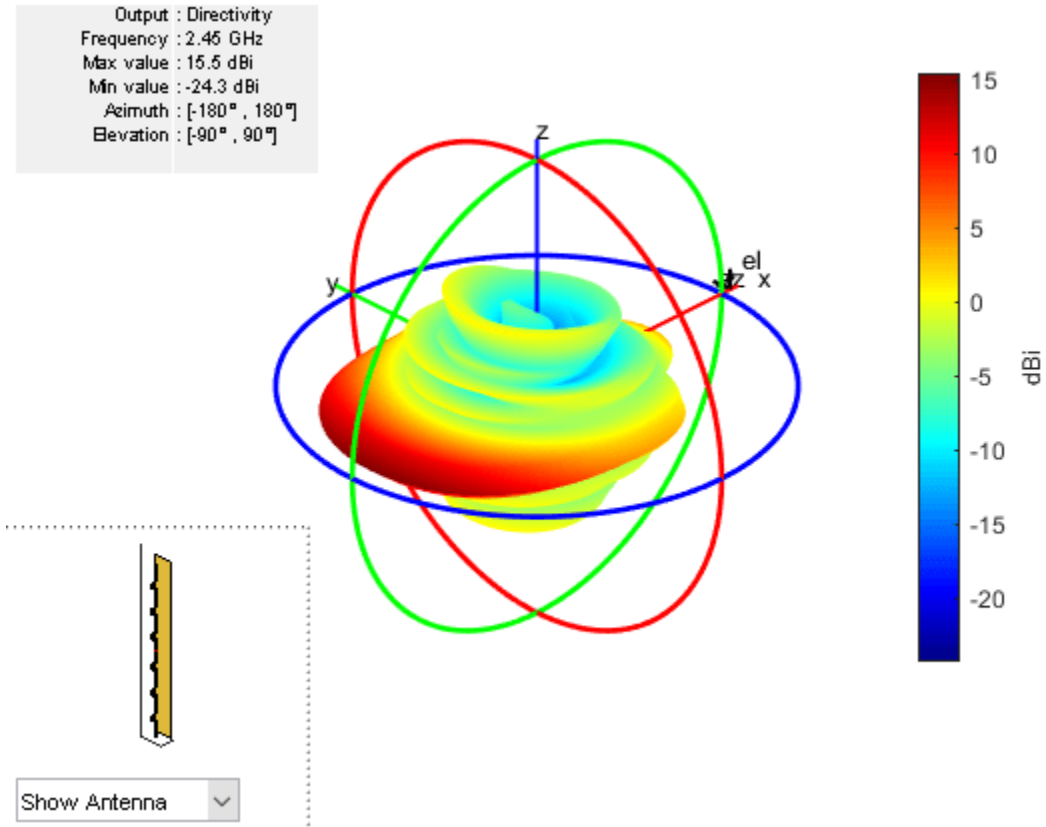
Radiation Pattern of Sector Antenna

The plots given below show the radiation patterns of the antenna at the center of the WiFi band at 2.45 GHz. As the antenna name implies, the radiation pattern illuminated a sector with minimum radiation in the back lobe. The maximum directivity is 15.5 dBi, which is slightly higher than the 15.4 dBi reported in [1].

```

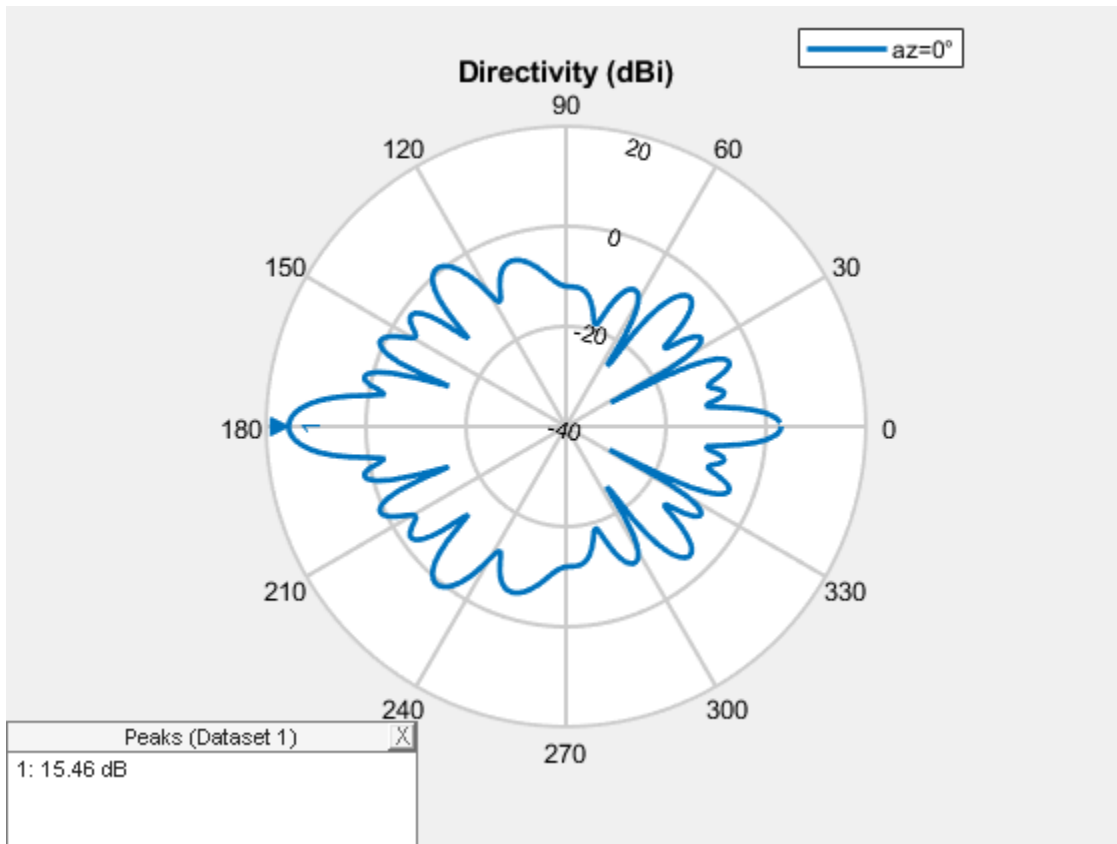
figure;
pattern(sector, 2.45e9);
view(-45,30)

```

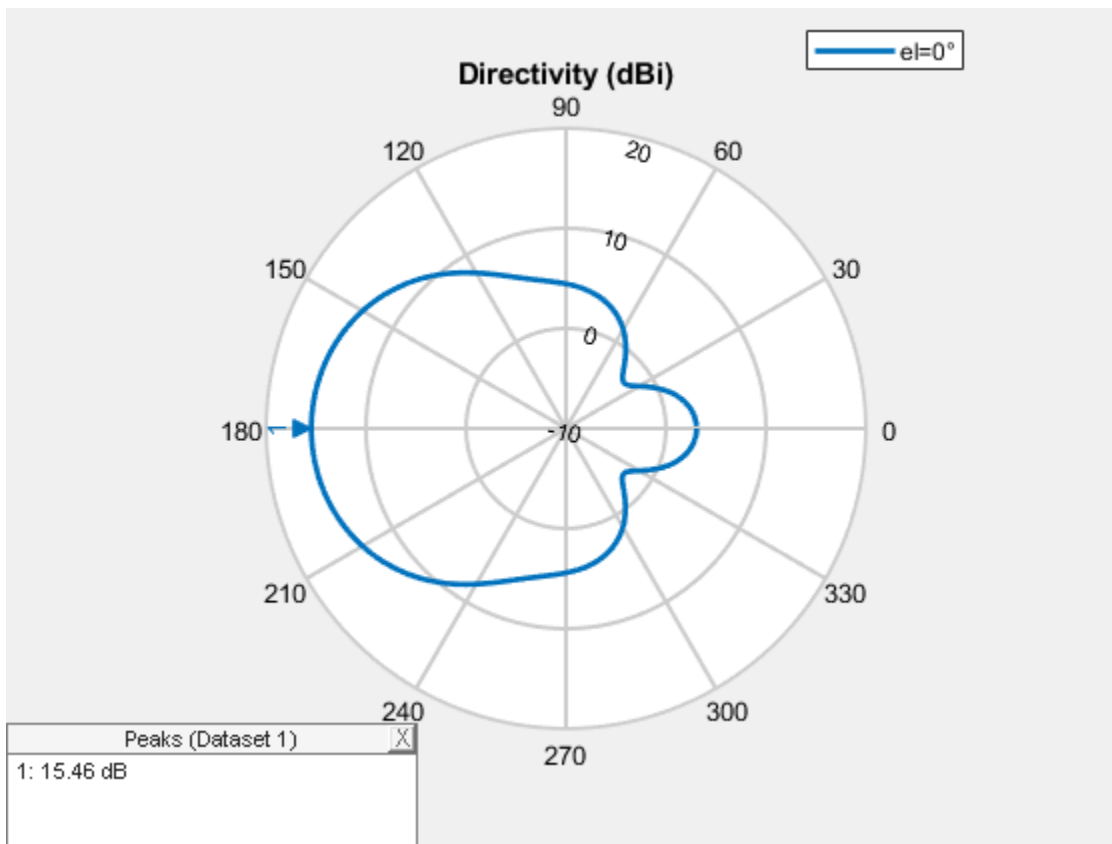


The plots below show the slices of the radiation patterns in the two principle planes.

```
figure;  
pattern(sector, 2.45e9, 0, 1:1:360);
```



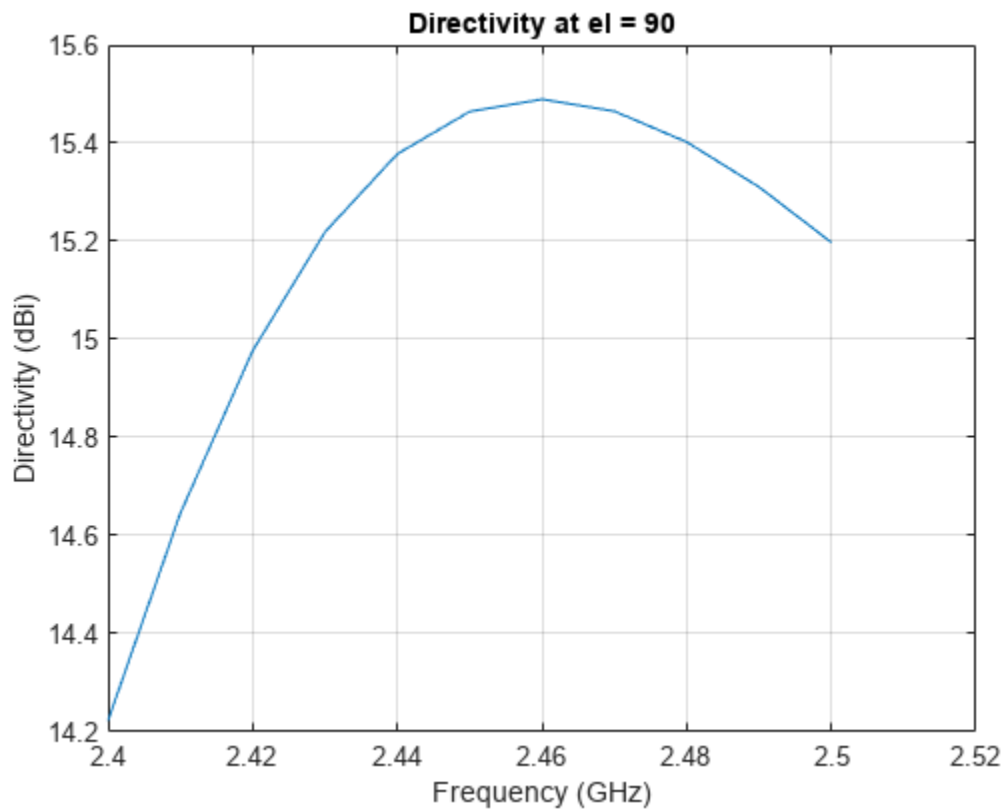
```
figure;  
pattern(sector, 2.45e9, 1:1:360,0);
```



Antenna Performance Over the Wi-Fi Band

Measure the maximum directivity at an elevation angle of 90 degrees. It would be important to have a constant value over the entire band of interest. The plot below, displays the directivity at zenith from 2.4 GHz to 2.5 GHz.

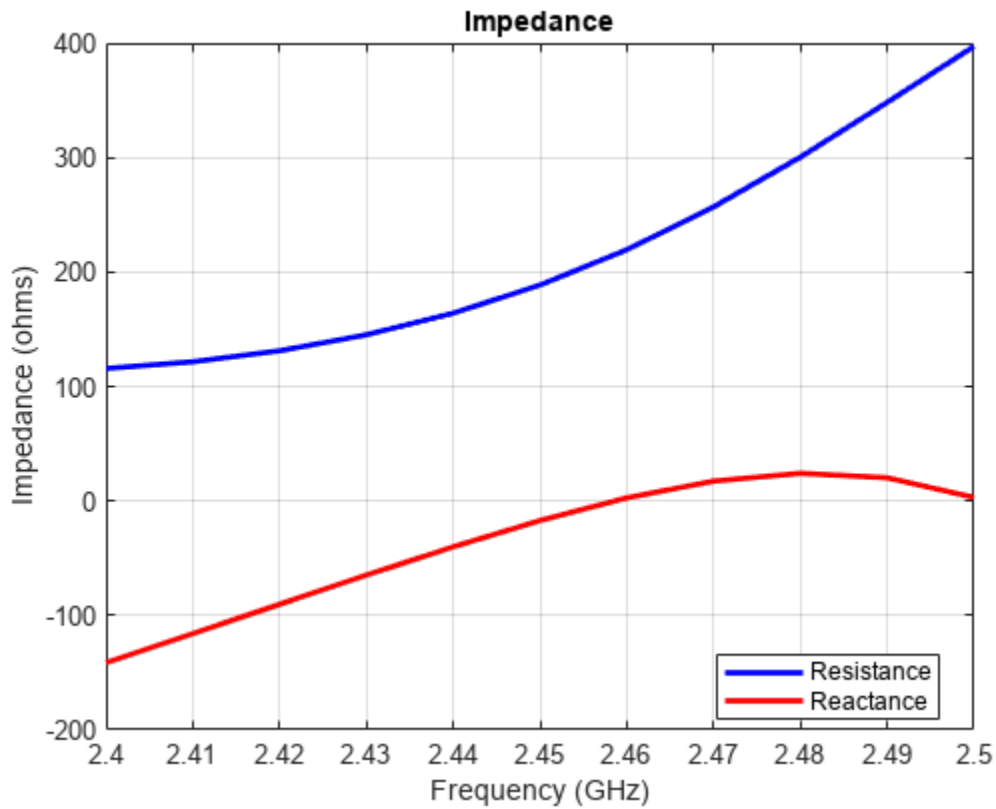
```
freq = linspace(2.4e9, 2.5e9, 11);
D = zeros(1,numel(freq));
for m=1:numel(freq)
    D(m) = pattern(sector, freq(m), 180, 0);
end
figure;
plot(freq./1e9, D);
title('Directivity at el = 90');
ylabel('Directivity (dBi)');
xlabel('Frequency (GHz)');
grid on;
```



Antenna Port Analysis

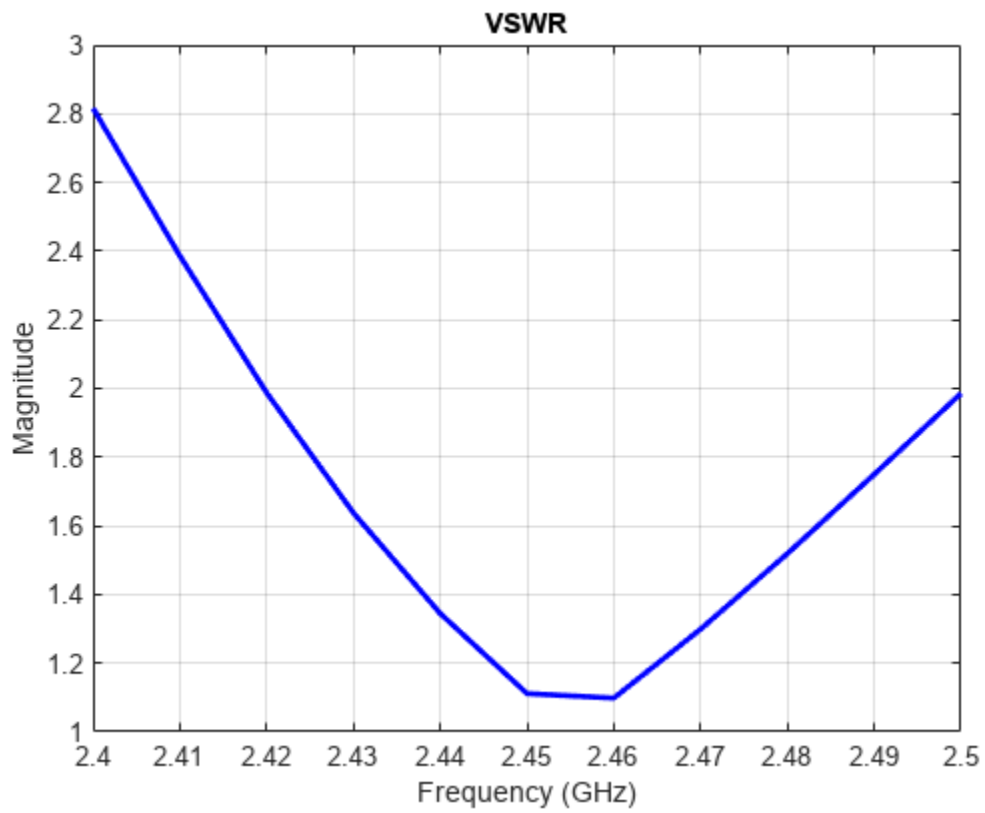
The plot given below shows the input impedance variation of the antenna over the band from 2.4 GHz to 2.5 GHz. The resistance varies between 100 to 400 ohms. To minimize reflections, the antenna is matched to a 200 ohm impedance.

```
figure  
impedance(sector, freq);
```



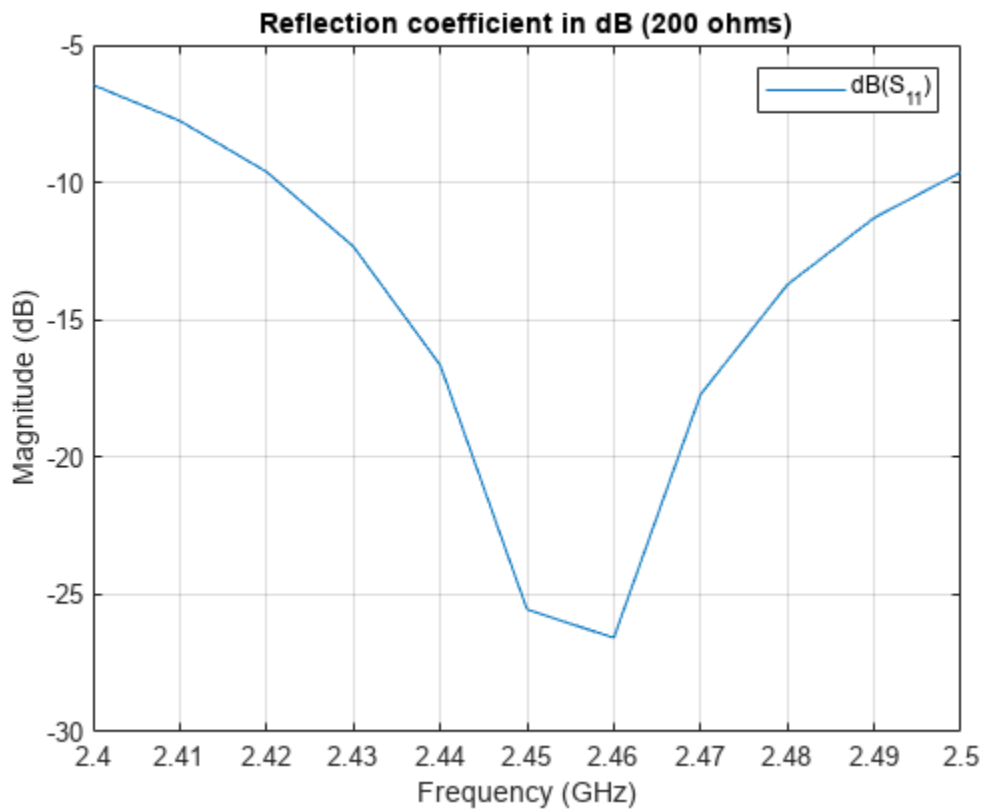
Another way to study the port-based reflection characteristics is to plot the voltage standing wave ratio (VSWR) as well as the reflection coefficient measured with regard to the reference impedance of 200 ohms over the entire WiFi band. The antenna has VSWR around 2 or less and the reflection coefficient around -10 dB or less over the entire frequency band of interest. This indicates an acceptable matching over the WiFi band.

```
figure  
vswr(sector, freq, 200);
```

Calculate the reflection coefficient using the S-parameter data.

```
S = sparameters(sector, freq, 200);  
figure;  
rfplot(S);  
title('Reflection coefficient in dB (200 ohms)');
```



See Also

“Design PIFA for WLAN Wi-Fi™ Applications” on page 5-117

Reference

[1] Inverted Amos Sector Antenna for 2.4 GHz WiFi, AntennaX Issue No. 130, February, 2008.

Design PIFA for WLAN Wi-Fi™ Applications

This example discusses the PIFA designed for Wi-Fi™ applications [1]. The Planar Inverted-F Antenna (PIFA) is basically a grounded patch antenna with the patch length of $\lambda/4$ (open-short microstrip resonator) instead of the conventional $\lambda/2$ for the patch antenna (open-open microstrip resonator). It typically consists of a ground plane, a top patch grounded at one end, and a coaxial probe feeding the top patch. The PIFA is commonly used in mobile communications since it has low manufacturing cost, smaller than the patch antenna size, and a relatively simple structure. In addition it also has reduced backward radiation towards user's head. However, the patch antenna has better polarization purity. Hundreds of PIFA varieties have recently been developed for mobile phones.

Design Parameters

The design parameters for this antenna are length, width, and height of the top patch, as well as the location of the feed probe. The shorting strip is always located at one patch edge; its length is always the edge length. We start the design process with antenna dimensions initialized below. The patch length is chose to be quarter wavelength at the upper frequency of the band (2.5 GHz).

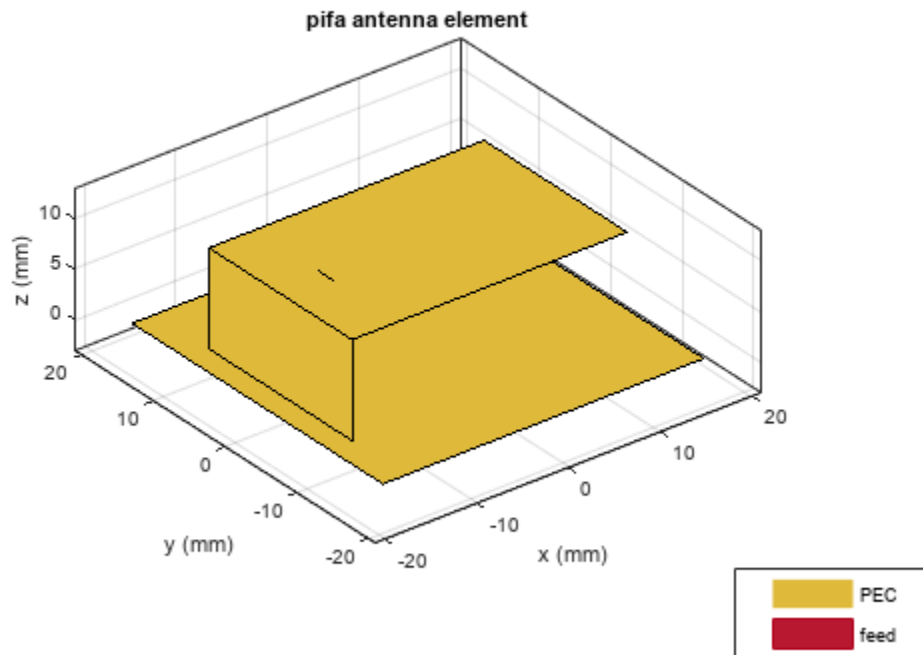
```
patchLength = 30e-3;
patchWidth  = 20e-3;
patchHeight = 10e-3;
lengthgp    = 35e-3;
widthgp     = 35e-3;
feedoffset  = [-patchLength/2+ 5e-3 0];
```

Create Planar Inverted-F Antenna

The parameters defined above are used to create the PIFA. The feed offset is so specified that the feed is 5 mm away from the shorted patch edge. The strip feed width corresponds to a wire with the radius of 1 mm.

```
ant = pifa('Length',patchLength, 'Width', patchWidth, 'Height',      ...
          patchHeight, 'GroundPlaneLength', lengthgp, 'GroundPlaneWidth', ...
          widthgp, 'ShortPinWidth', patchWidth, 'FeedOffset', feedoffset);

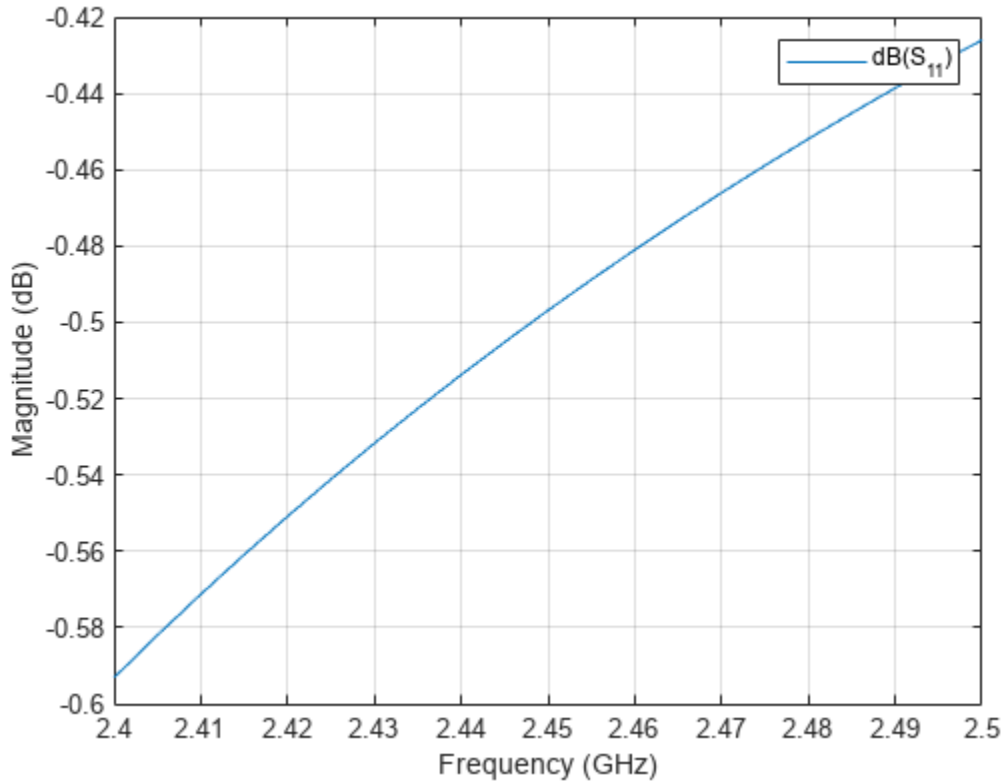
figure;
show(ant);
```



Design for WLAN Applications

This section discusses design of a compact PIFA with a suitable bandwidth for wireless application in the WiFi™ band. The bandwidth is defined as the frequency band in which VSWR is less than 2:1 which approximately corresponds to a -10 dB or lower reflection coefficient. This means that 10% or less of the incident power is reflected back to the generator.

```
freq = linspace(2.4e9, 2.5e9, 21);  
s1 = sparameters(ant,freq);  
S11Fig1 = figure;  
rfplot(s1);
```



From the plot above, it follows that the antenna is not matched. The antenna design needs to be changed to ensure that the reflection coefficient is less than -10 dB over the frequency range of interest.

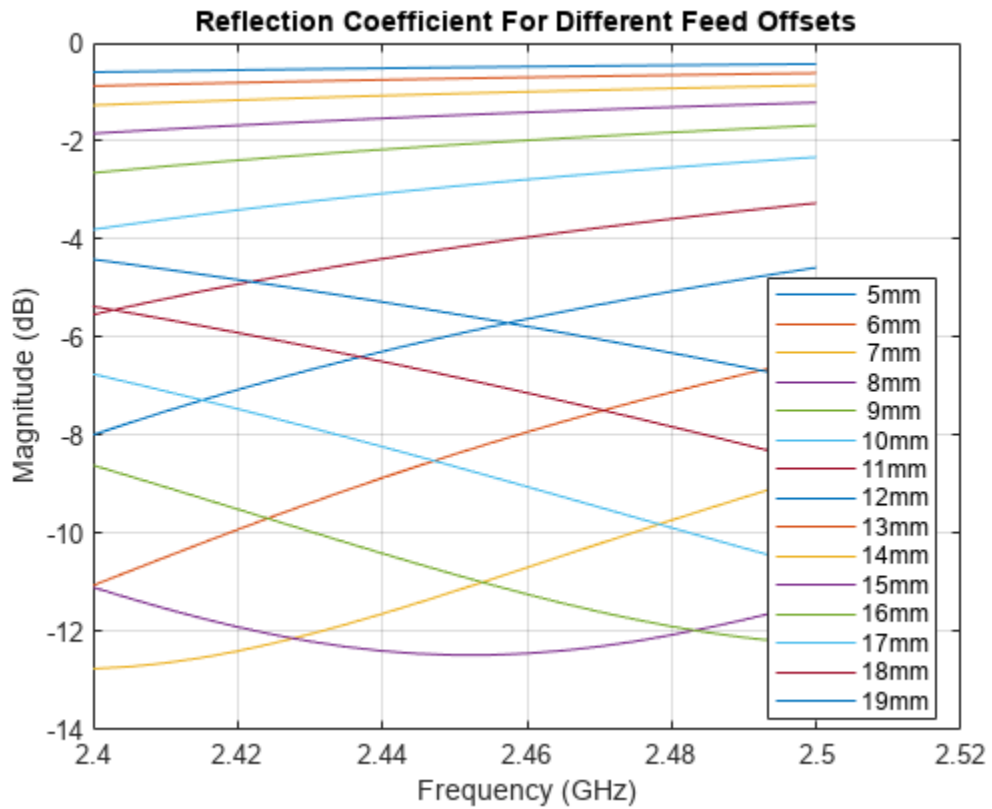
Vary Antenna Feed Location

A simple and efficient way to provide impedance match for both PIFA and patch antennas is to move the feed location. We typically move the feed along the resonating length of the patch and find the position where the best match is achieved. Use the FeedOffset parameter for this purpose and plot the reflection coefficient for every feed location.

```

feedoffsetx = 5e-3:1e-3:19e-3;
S11 = zeros(numel(freq), numel(feedoffsetx));
for m =1:length(feedoffsetx)
    feedoffset = [-patchLength/2 + feedoffsetx(m) 0];
    ant.FeedOffset = feedoffset;
    S = sparameters(ant,freq);
    S11(:,m) = 20*log10(abs(S.Parameters));
end
S11Fig2 = figure;
plot(freq./1e9, S11);
legend(strcat(num2str(feedoffsetx'.*1e3), 'mm'), 'location', 'Best');
grid on;
title('Reflection Coefficient For Different Feed Offsets');
xlabel('Frequency (GHz)');
ylabel('Magnitude (dB)');

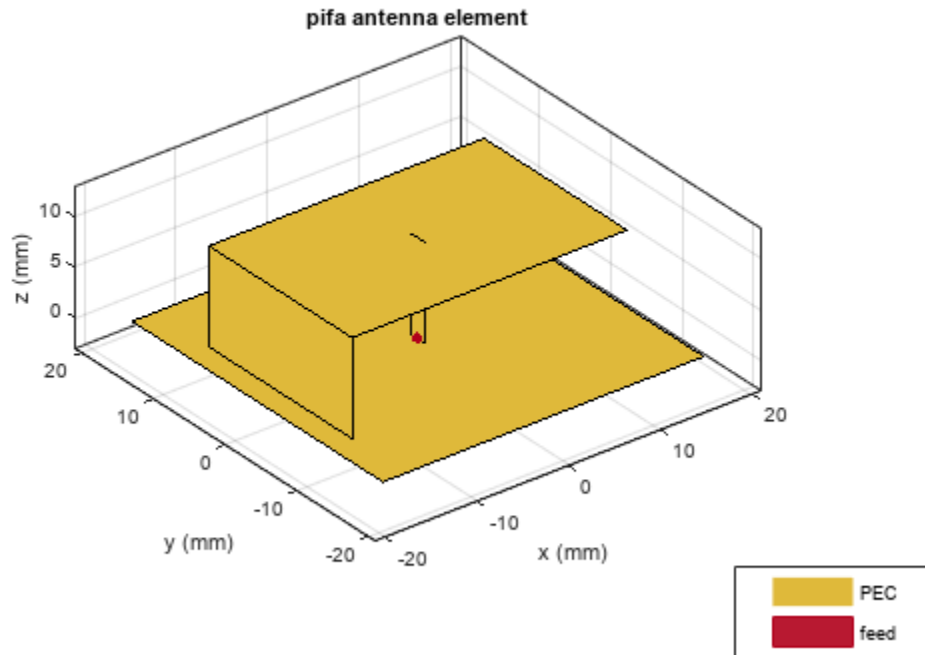
```



Choose Best Design

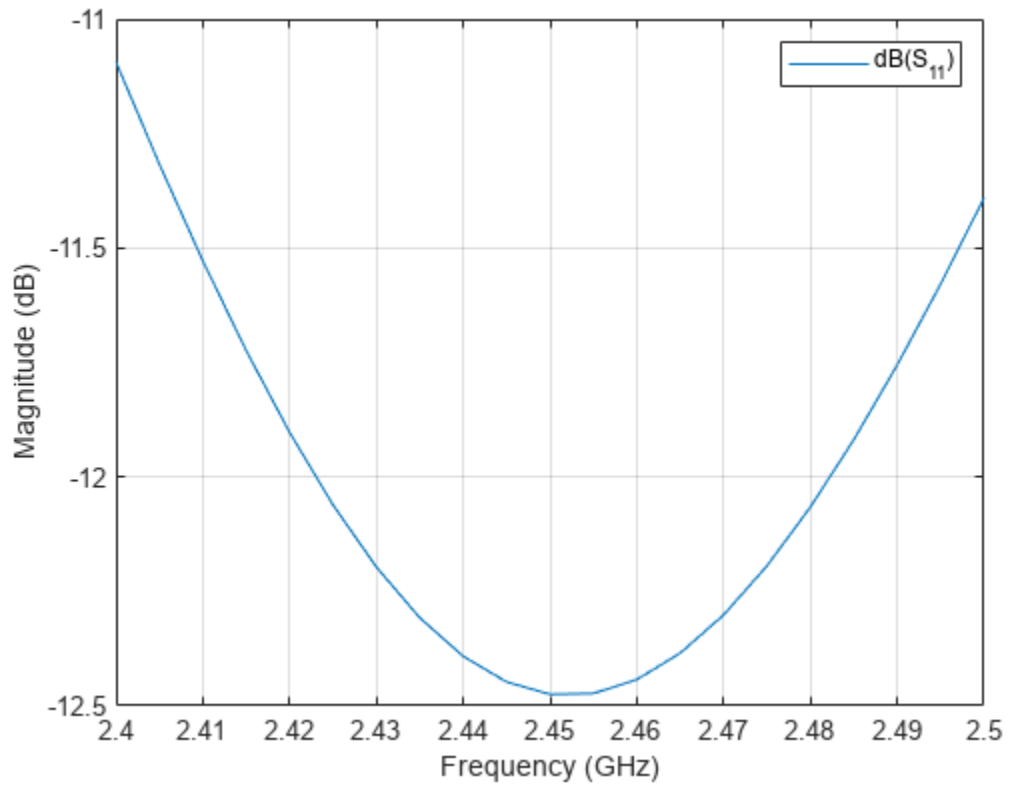
From the reflection coefficient plots we can see that the feed offset of 15 mm provides the best match. Update the feed offset and visualize the new antenna structure. The feed is now located closer to the patch center.

```
ant.FeedOffset = [-patchLength/2 + 15e-3 0];
figure;
show(ant);
```

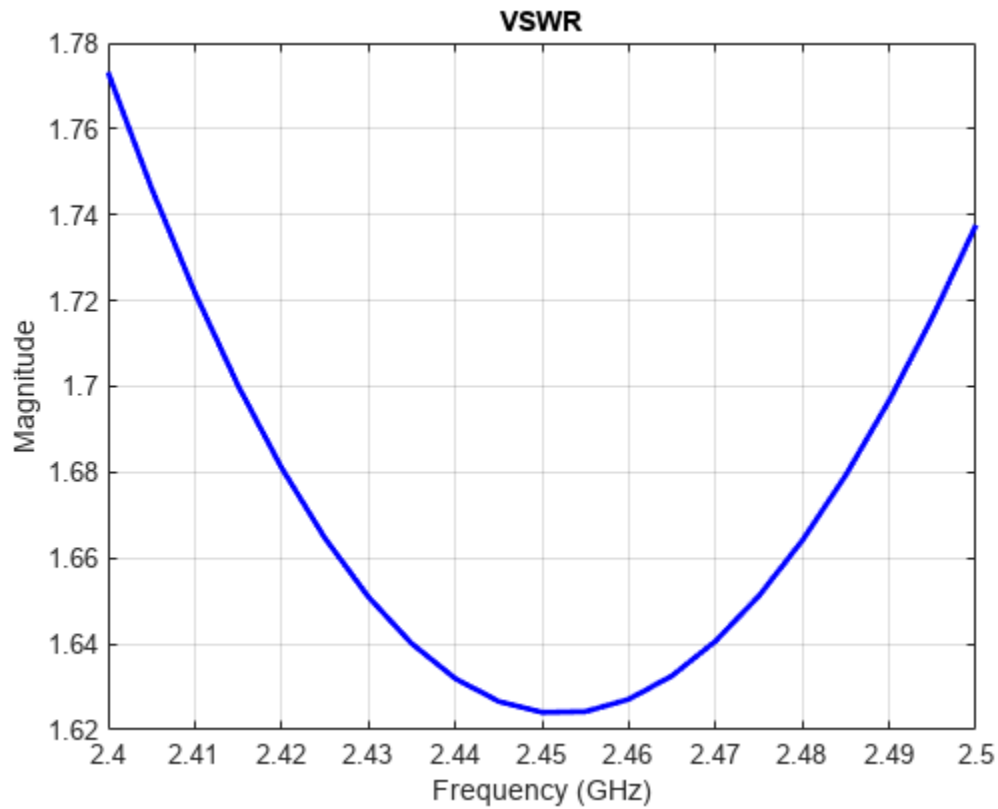


Below we also plot the reflection coefficient and VSWR for the optimal antenna design.

```
s2 = sparameters(ant, freq);  
S11Fig3 = figure;  
rfplot(s2);
```



```
VSWRfig = figure;  
vswr(ant, freq);
```


**See Also**

“Sector Antenna for 2.4 GHz Wi-Fi™” on page 5-107

Reference

[1] G. M. Khanal, "Design of a Compact PIFA for WLAN Wi-Fi Wireless Applications," Int. J. of Engineering Research and Development, www.ijerd.com Vol. 8, Issue 7, Sept. 2013, pp. 13-18, e-ISSN:2278-067X, p-ISSN: 2278-800X. Available online at:<http://www.ijerd.com/paper/vol8.issue7/C08071318.pdf>

Helical Antenna Design

This example studies a helical antenna designed in [2] with regard to the achieved directivity. Helical antennas were introduced in 1947 [1]. Since then, they have been widely used in certain applications such as mobile and satellite communications. Helical antennas are commonly used in an axial mode of operation which occurs when the circumference of the helix is comparable to the wavelength of operation. In this mode, the helical antenna has the maximum directivity along its axis and radiates a circularly-polarized wave.

Helix Design Specifications

The helical antenna design specifications are as follows ([2]):

- Frequency range: 1.3 - 2 GHz
- Gain: 13 dBi +/- 1.5 dBi
- Axial Ratio: < 1.5

Model Assumptions and Differences

Compared to Ref. [2], the helical antenna model available in the toolbox uses the following simplifying assumptions:

- Conductor - The original reference uses a cylinder of radius r while the toolbox uses a strip of width w .
- Ground plane shape - A square ground plane is used in the original reference, while the present toolbox model uses a circular shape.
- Width of Feed - The width of the feed in [2] is $r/10$ while the toolbox model uses w .
- Design validation metric - Ref. [2] used the gain to compare simulation and measurement results, whereas in the toolbox, we will use the directivity since the simulated antenna has a negligibly small loss.

Helix Design Parameters

The helix model in the toolbox uses the strip approximation which relates the width of the strip to the radius of an equivalent cylinder [3]. In addition, the helix model in the toolbox has a circular ground plane. Choose the radius of the ground plane to be half the length of side of the square ground plane.

```
r          = 0.3e-3;
width      = cylinder2strip(r);
feedheight = 3*r;
D          = 56e-3;
radius     = D/2;
turns      = 17.5;
pitch      = 11.2;
spacing    = helixpitch2spacing(pitch,radius);
side       = 600e-3;
radiusGP   = side/2;
```

Frequency of Operation and Bandwidth

The center frequency is chosen as 1.65 GHz. A relative bandwidth of 45% is chosen which provides sufficient flexibility since the operating frequency limits result in a relative bandwidth of 42.5%. Relative bandwidth is calculated as,

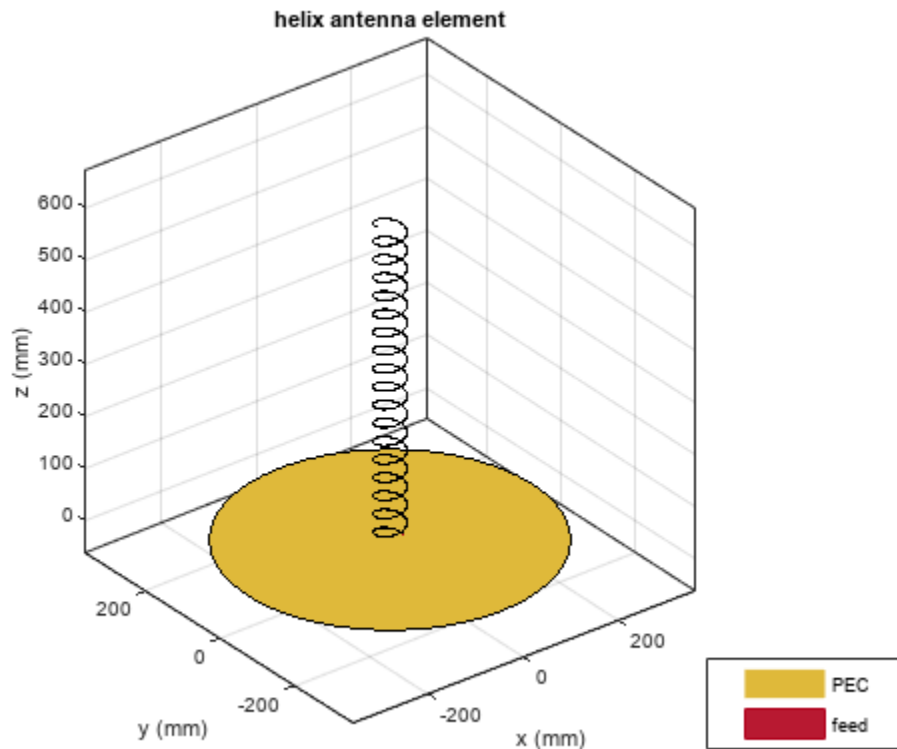
$$BW_{relative} = (f_{upper} - f_{lower})/f_c$$

```
fc          = 1.65e9;
relativeBW = 0.45;
BW          = relativeBW*fc;
```

Create Helical Antenna

Create the helix antenna with the appropriate properties as calculated before and view the structure.

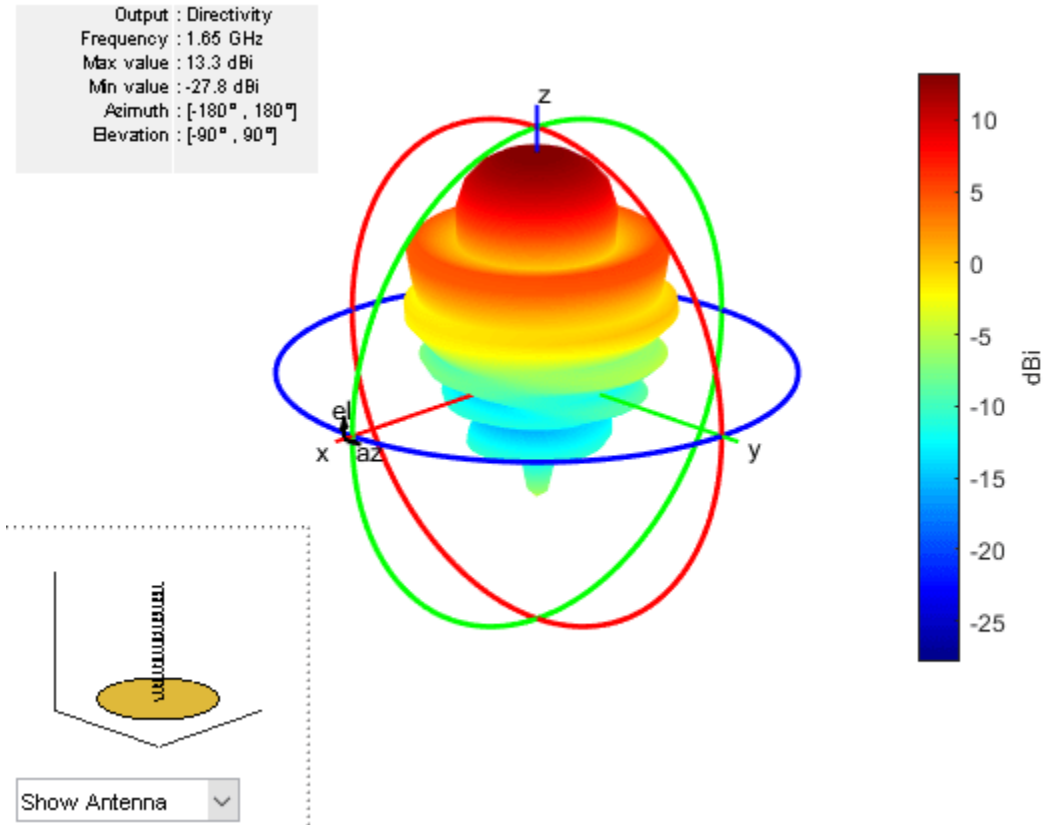
```
hx = helix('Radius',radius,'Width',width,'Turns',turns,...
           'Spacing',spacing,'GroundPlaneRadius',radiusGP,...
           'FeedStubHeight',feedheight);
figure;
show(hx);
```



Pattern Behavior

Plot the directivity radiation pattern of the helical antenna at the center frequency of 1.65 GHz. This pattern confirms the axial mode of operation for the helical antenna.

```
figure;
pattern(hx,fc);
```

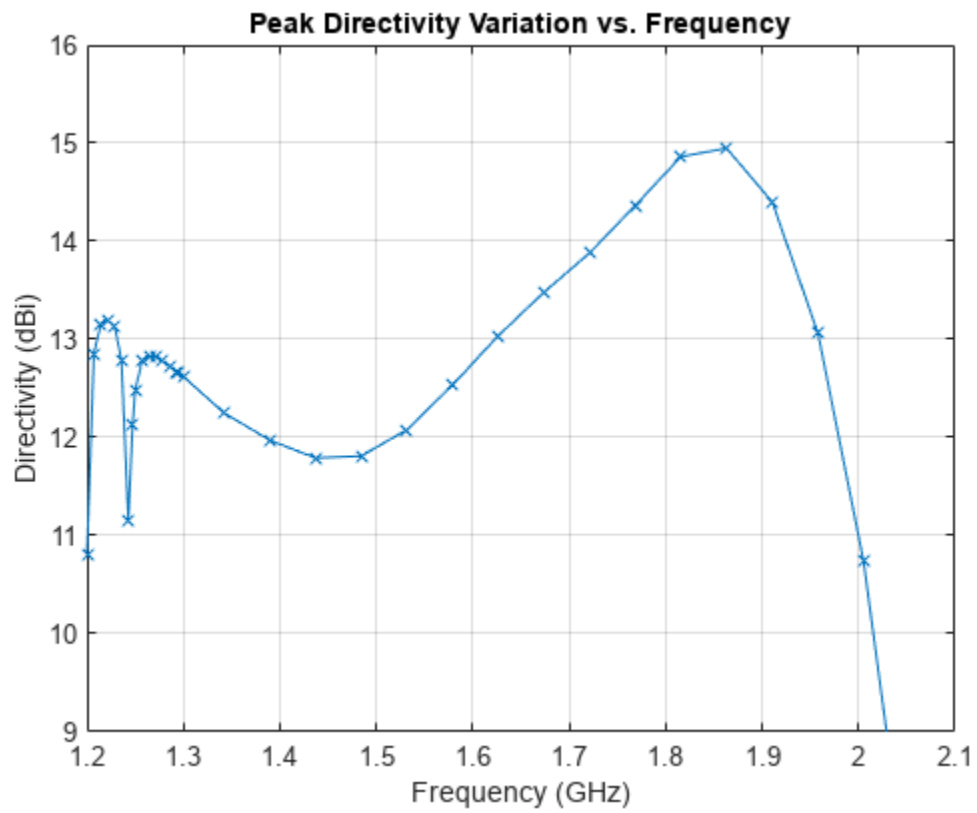


To calculate the directivity variation of the main beam as a function of frequency, choose a frequency range according to [2].

```

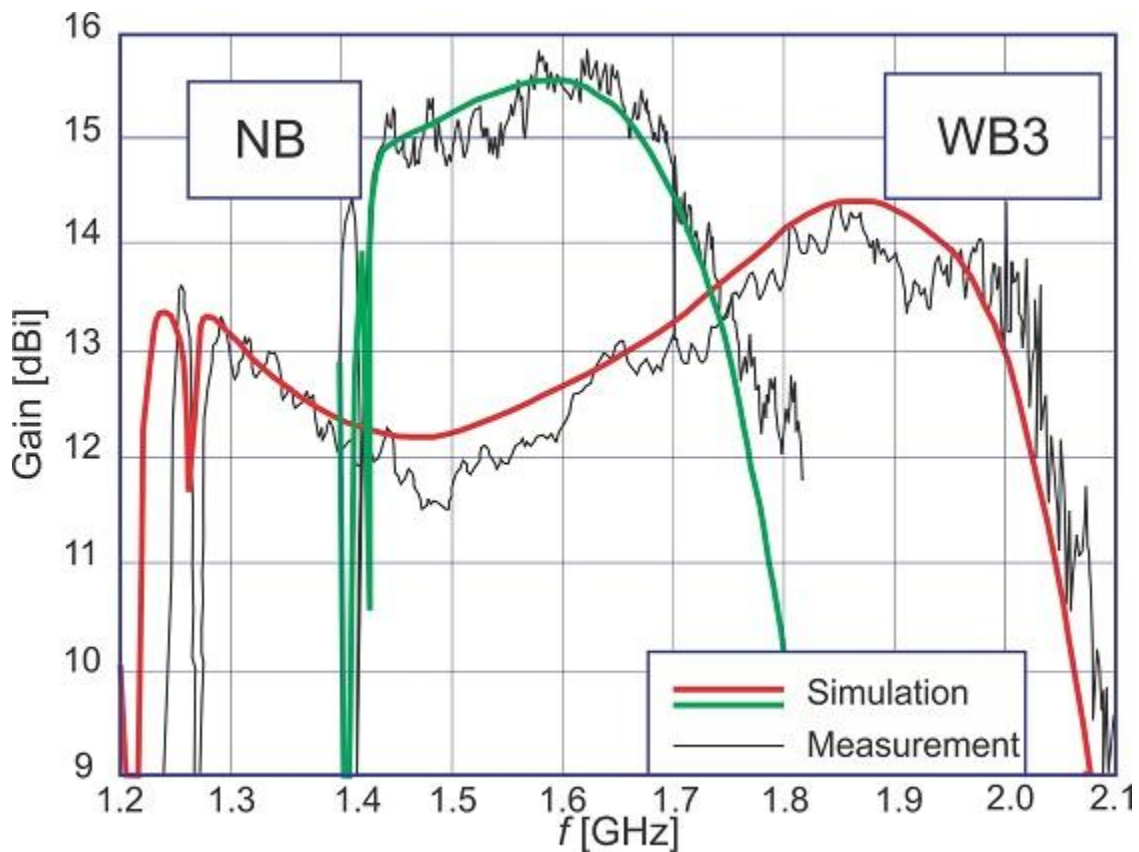
Nf1    = 15;
Nf2    = 20;
fmin   = 1.2e9;
fmax   = 2.1e9;
fstep  = 0.1e9;
fband1 = linspace(fmin,1.3e9,Nf1);
fband2 = linspace(fmin,fmax,Nf2);
freq   = unique([fband1,fband2]);
Nf     = length(freq);
D      = nan(1,Nf);
f_eng  = freq./1e9;
f_str  = 'G';
fig1 = figure;
for i = 1:length(freq)
    D(i) = pattern(hx,freq(i),0,90);
    figure(fig1)
    plot(f_eng,D,'x-')
    grid on
    axis([f_eng(1) f_eng(end) 9 16 ])
    xlabel(['Frequency (' f_str 'Hz)'])
    ylabel('Directivity (dBi)')
    title('Peak Directivity Variation vs. Frequency')
    drawnow
end

```



Discussion on the Results

Comparing this result with Fig. 11 in [2] replicated below, we establish the quantitative agreement.



Simulated and measured RHC gain for the NB and WB3 designs [2] (Reproduced with permission from IEEE)

See Also

"Monopole Measurement Comparison" on page 5-80

References

- [1] J. D. Kraus, "Helical Beam Antennas," *Electronics*, 20, April 1947, pp. 109-111.
- [2] A. R. Djordjevic, A. G. Zajic, M. M. Ilic, G. L. Stuber, "Optimization of Helical antennas [Antenna Designer's Notebook]," *IEEE Antennas and Propagation Magazine*, vol.48, no.6, pp.107-115, Dec. 2006.
- [3] C. A. Balanis, 'Antenna Theory. Analysis and Design,' p. 514, Wiley, New York, 3rd Edition, 2005.

Reflector Backed Equiangular Spiral

The spiral antenna is an inherently broadband, and bidirectional radiator. This example will analyze the behavior of an equiangular spiral antenna backed by a reflector [1]. The spiral and the reflector are Perfect Electric Conductors (PEC).

Spiral and Reflector Parameters

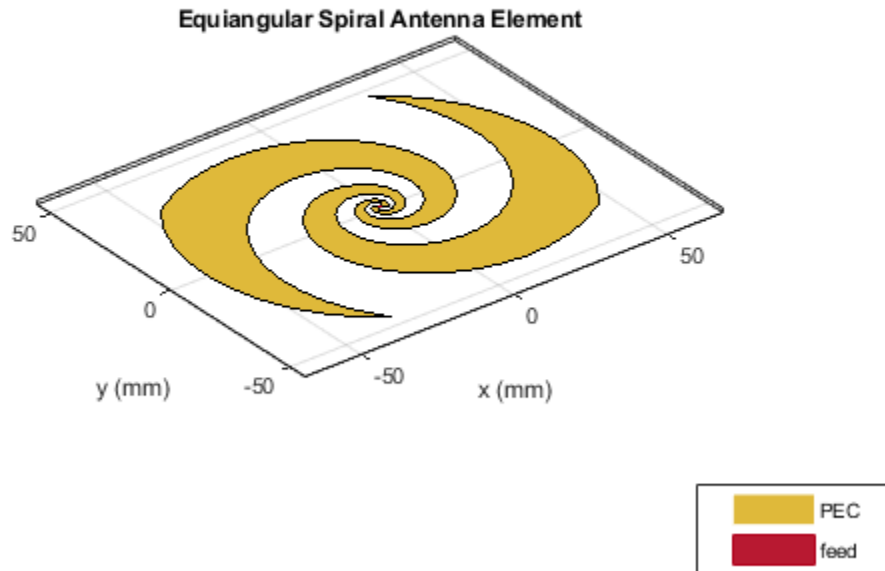
The spiral antenna parameters and reflector dimensions are provided in [1]. The design frequency range is 4-9 GHz and the analysis frequency range will span 3-10 GHz.

```
a = 0.35;
rho = 1.5e-3;
phi_start = -0.25*pi;
phi_end = 2.806*pi;
R_in = rho*exp(a*(phi_start + pi/2));
R_out = rho*exp(a*(phi_end + pi/2));
gndL = 167e-3;
gndW = 167e-3;
spacing = 7e-3;
```

Create Equiangular Spiral Antenna

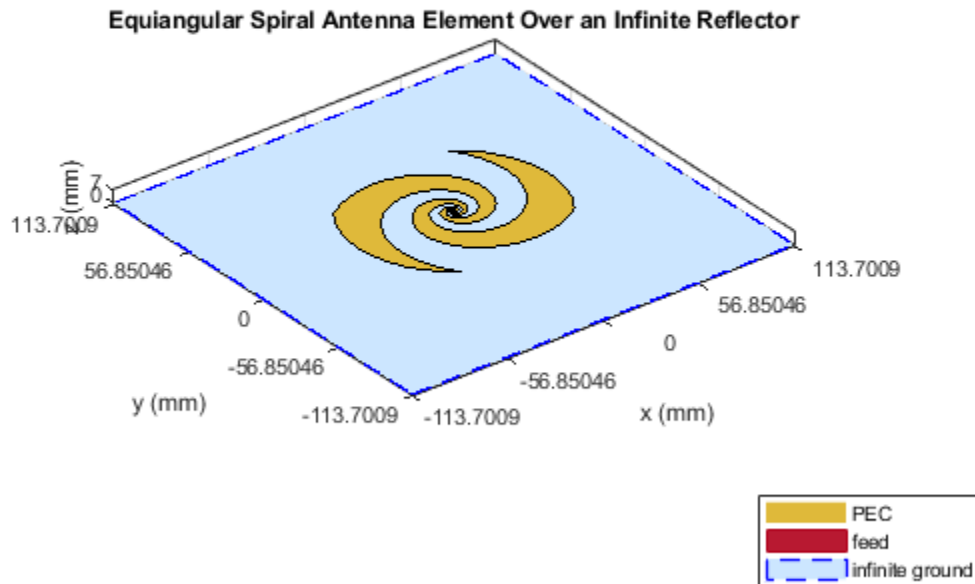
Create an equiangular spiral antenna using the defined parameters.

```
sp = spiralEquiangular;
sp.GrowthRate = a;
sp.InnerRadius = R_in;
sp.OuterRadius = R_out;
figure;
show(sp)
title('Equiangular Spiral Antenna Element');
```



Create a reflector, and assign the spiral antenna as its exciter. Adjust the spacing between the reflector and spiral to be 7 mm. The first-pass analysis will be with an infinitely large groundplane. To do this, assign the groundplane length and/or width to inf.

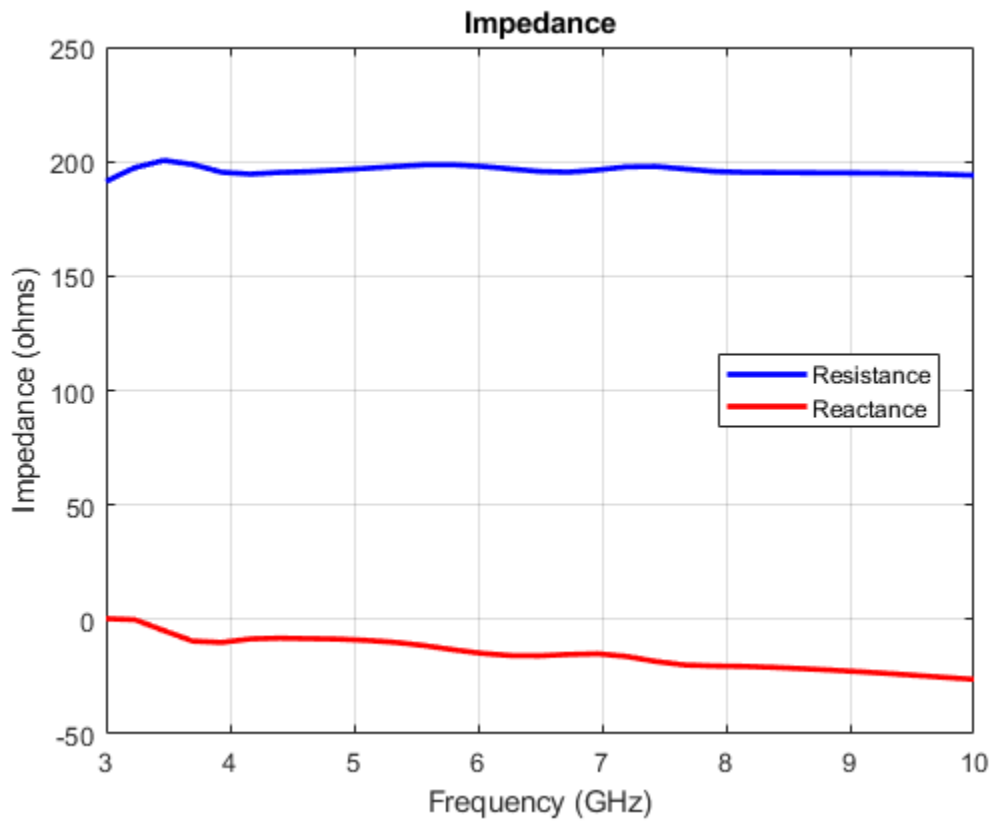
```
rf = reflector;  
rf.GroundPlaneLength = inf;  
rf.GroundPlaneWidth = inf;  
rf.Exciter = sp;  
rf.Spacing = spacing;  
figure  
show(rf)  
title('Equiangular Spiral Antenna Element Over an Infinite Reflector');
```

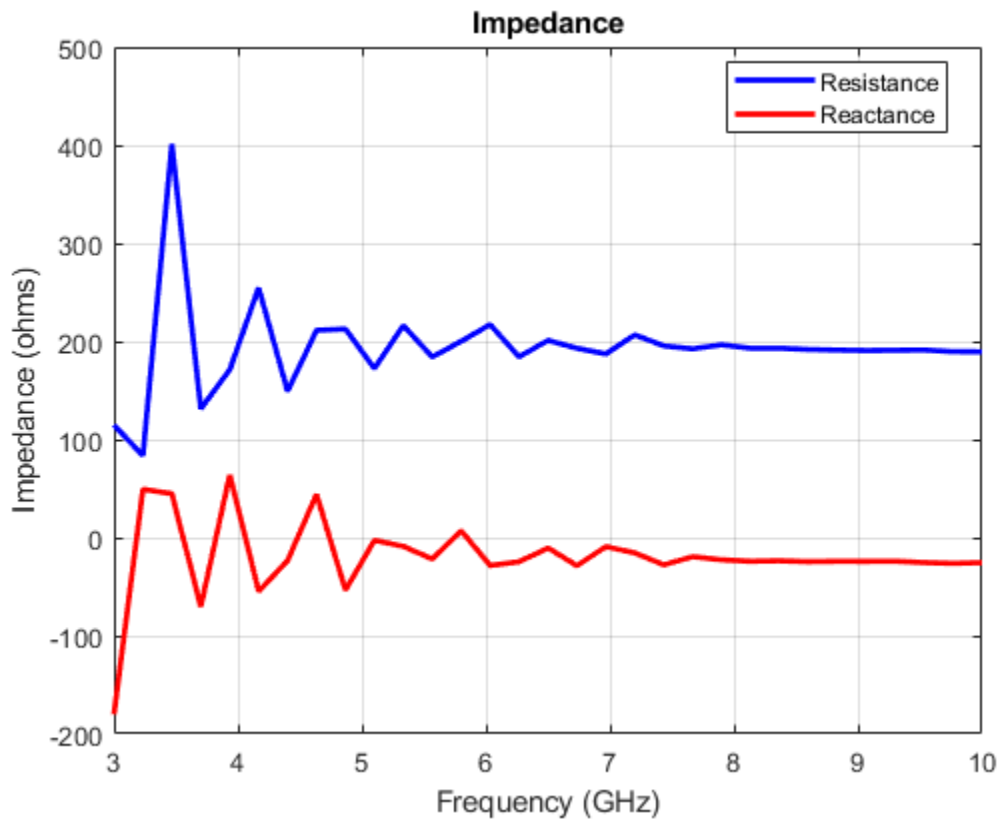
Impedance Analysis of Spiral with and without Infinite Reflector

Define a frequency range for the impedance analysis. Keep the frequency range sampling coarse to get an overall idea of the behavior. A detailed analysis will follow. Analyze the impedance of both antennas: the spiral antenna in free space and the antenna with the infinite reflector.

```
freq = linspace(3e9,10e9,31);
figure;
impedance(sp, freq)
```



```
figure;  
impedance(rf,freq);
```

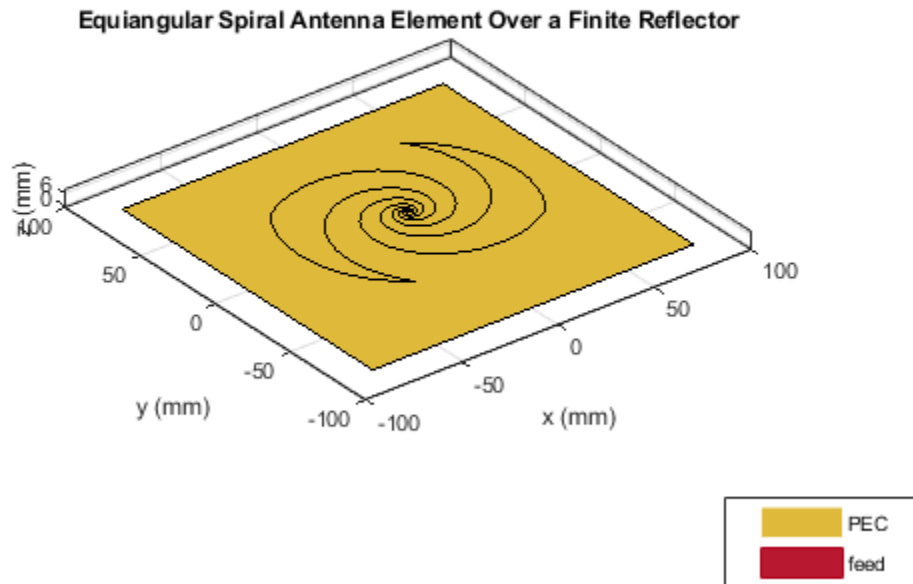


The impedance for the spiral without the reflector is smooth in both resistance and reactance. The resistance holds steady around the 184Ω value while the reactance although slightly capacitive, does not show any drastic variations. However, introducing an infinitely large reflector at a spacing of 7mm disturbs this smooth behavior for both the resistance and reactance.

Make Reflector Finite

Change the reflector to be of finite dimensions by using the dimensions specified earlier as per [1].

```
rf.GroundPlaneLength = gndL;
rf.GroundPlaneWidth = gndW;
show(rf);
title('Equiangular Spiral Antenna Element Over a Finite Reflector');
```



Mesh Different Parts Independently

To analyze the behavior of the spiral antenna backed by the finite-size reflector, mesh the antenna and reflector independently. The highest frequency in the analysis is 10 GHz, which corresponds to a λ of 3 cm. Mesh the spiral with a maximum edgelenhth of 2 mm (lower than $\lambda/10 = 3\text{mm}$ at 10 GHz). This requirement is relaxed for the reflector which is meshed with a maximum edgelenhth of 8mm (slightly lower than $\lambda/10 = 10\text{cm}$ at 3 GHz).

```
ms_new = mesh(rf.Exciter, 'MaxEdgeLength', .002)
mr_new = mesh(rf, 'MaxEdgeLength', 0.008)
```

Impedance Analysis of the Spiral Antenna with the Finite Reflector

Run the following code snippet at the prompt in the command window to create an accurate impedance plot for the finite reflector shown in the figure that follows. The impedance behavior typical for the infinitely large reflector is also observed for the finite reflector.

```
fig1 = figure;
impedance(rf, freq);
```

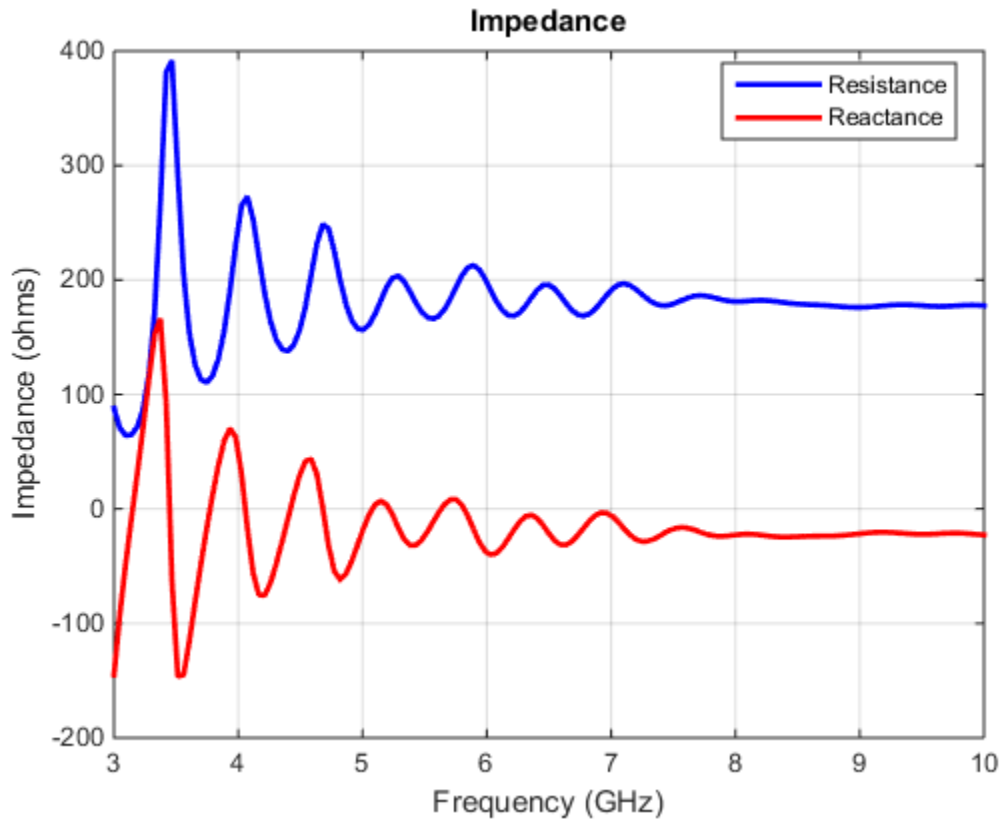


Fig. 1: Input impedance of the spiral with metal(PEC) reflector.

Surface Currents on Reflector-Backed Spiral Antenna

Run the following code snippet at the prompt in the Command Window to recreate the current distribution plots shown in the figure that follows. Since the design frequency band is 4 - 9 GHz, pick the two band edge frequencies for observing the surface currents on the spiral antenna. From the impedance analysis, the presence of multiple resonances at the lower end of the design frequency band, should manifest itself in the surface currents at 4 GHz.

```
fig2 = figure;  
current(rf,4e9)
```

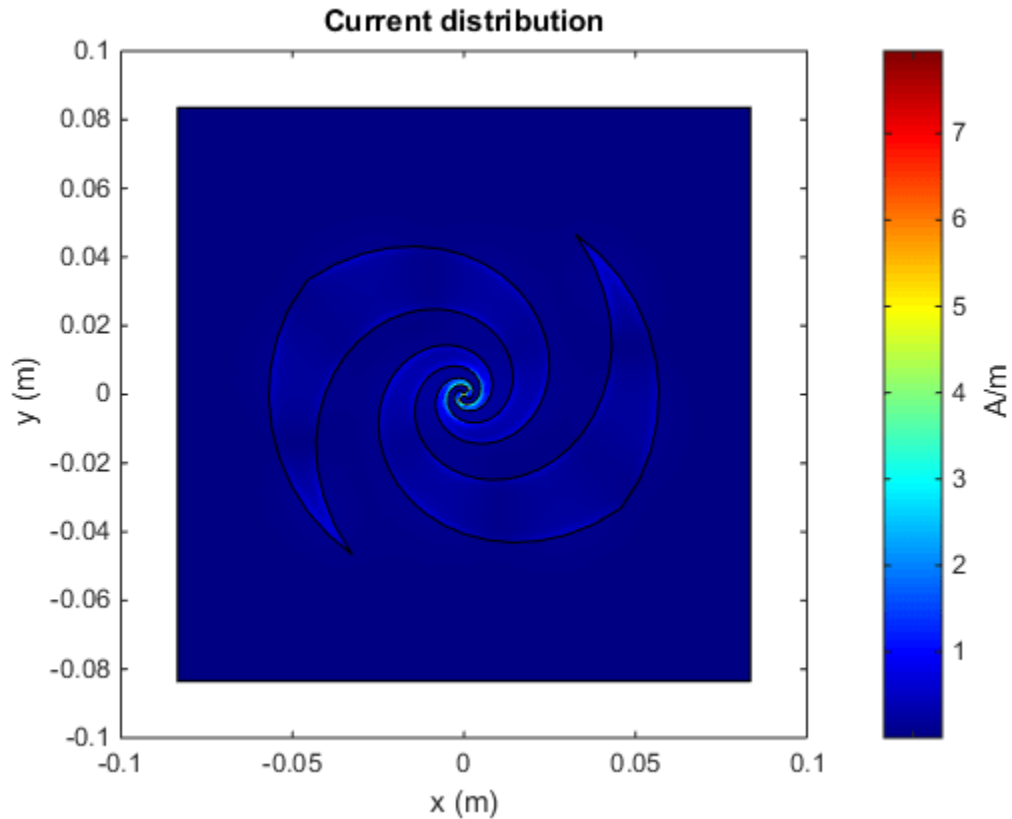


Fig. 2: Surface current density at 4 GHz.

At first glance, the surface current density plot might appear to not reveal any information. To explore further, use the colorbar on the right of the plot. This colorbar allows us to adjust the color scale interactively. To adjust the scale, move the mouse pointer to an area within the colorbar boundary, click, and drag. To adjust the range of values, move the mouse pointer outside the colorbar boundary and hover over the numeric tick marks on the colorbar or between the tick spacings, click and drag. This approach generates the following figure, which indicates presence of standing waves at 4GHz. The arrows on the plot are added separately to highlight the local current minima.

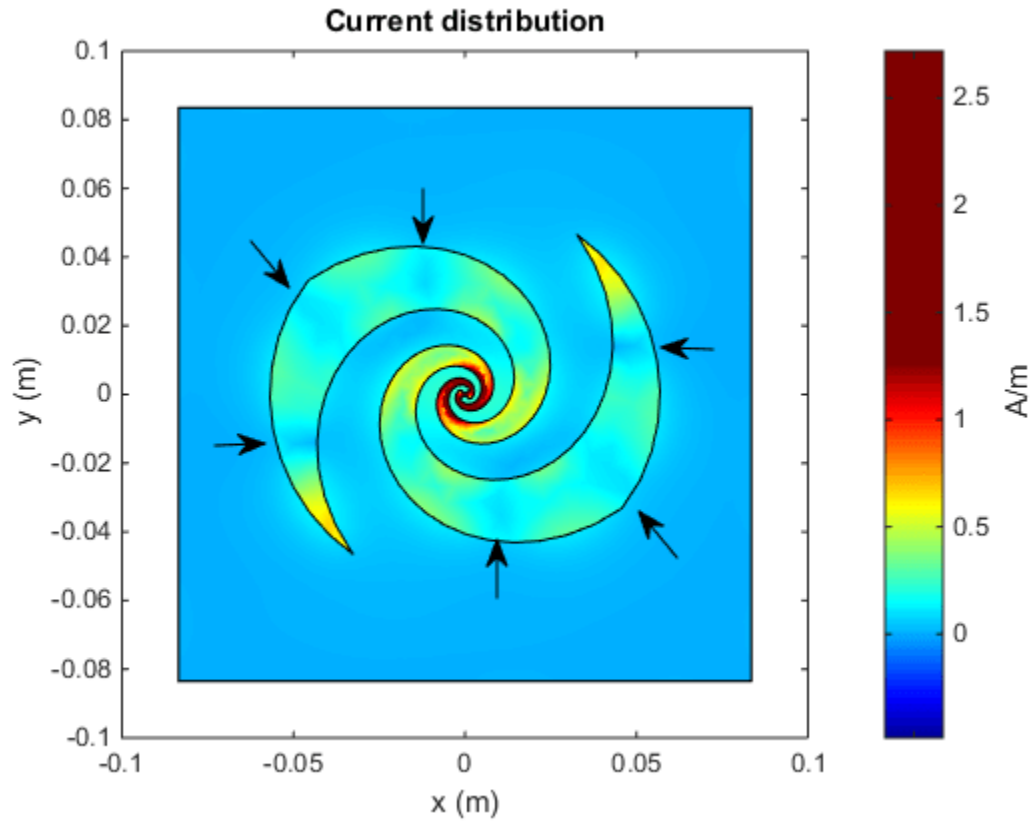


Fig. 3: Surface current density at 4 GHz after adjusting the dynamic range using the colorbar.

Plot the current density on the spiral antenna at 9 GHz. As in the previous case, we use the colorbar to adjust the scale to be similar and observe a current flow that is free of any standing waves.

```
fig4 = figure;  
current(rf,9e9)
```

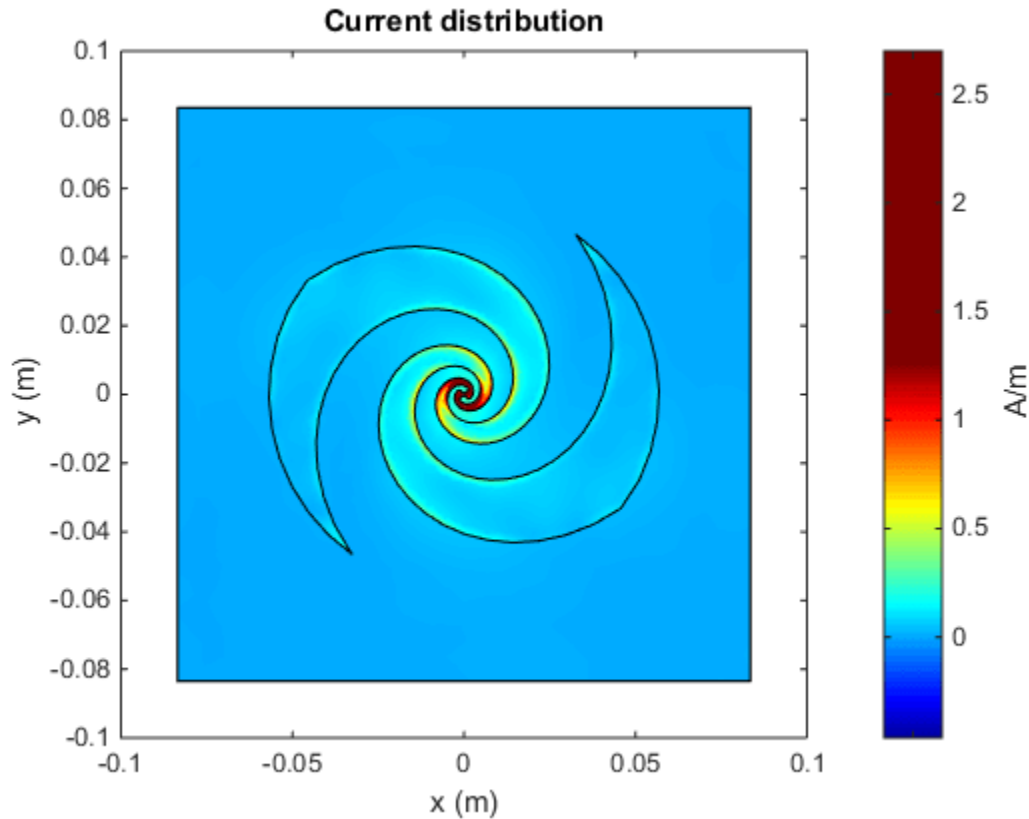


Fig. 4: Surface current density at 9 GHz.

Calculate Axial Ratio

The spiral antenna radiates circularly polarized waves. The axial ratio (AR) in a given direction quantifies the ratio of two orthogonal field components radiated in a circularly polarized wave. An axial ratio of infinity, implies a linearly polarized wave. When the axial ratio is 1, the radiated wave has pure circular polarization. Values greater than 1 imply elliptically polarized waves. Run the following code snippet at the prompt in the command window to recreate the axial ratio plot at broadside shown in the figure that follows. Note that the calculated axial ratio values are in dB, $20\log_{10}(\text{AR})$. To compare the effect of the reflector, the axial ratio is calculated for the spiral antenna with and without the reflector.

```
AR_spiral = zeros(size(freq));
AR_reflector = zeros(size(freq));

for i = 1:numel(freq);
    AR_spiral(i) = axialRatio(rf.Exciter,freq(i),0,90);
    AR_reflector(i)= axialRatio(rf,freq(i),0,90);
end

fig5 = figure;
plot(freq/1e9,AR_spiral,'LineWidth',2);
hold on
plot(freq./1e9,AR_reflector,'m','LineWidth',2);
grid on
ax1 = fig5.CurrentAxes;
```



```

ax1.YTickLabelMode = 'manual';
ax1.YLim = [0 20];
ax1.YTick = 0:2:20;
ax1.YTickLabel = cellstr(num2str(ax1.YTick));
xlabel('Frequency (GHz)')
ylabel('AR (dB)')
title('Frequency Response of Axial Ratio')
legend('Without reflector', 'With reflector');

```

The axial ratio plot of the spiral without the reflector shows that across the design frequency band, the spiral antenna radiates a nearly circularly polarized wave. The introduction of the reflector close to the spiral antenna degrades the circular polarization.

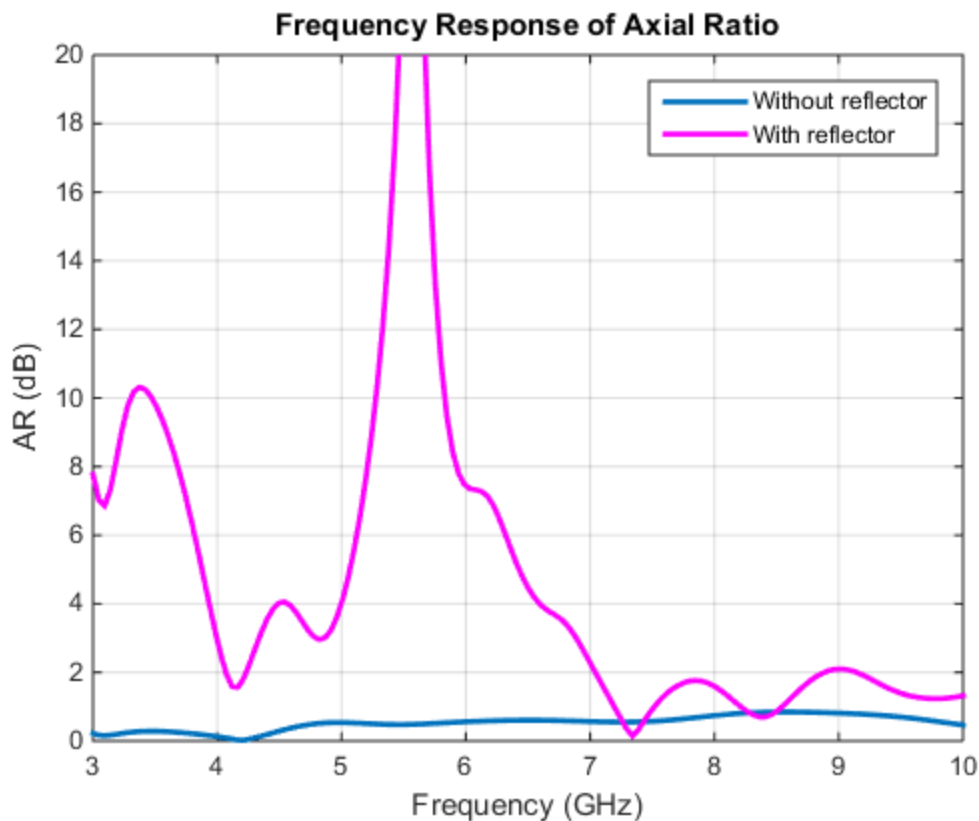


Fig. 5: Axial ratio at broadside in dB with and without reflector backing as a function of frequency [1]

Conclusion

The spiral antenna by itself has a broad impedance bandwidth and produces a bi-directional radiation pattern. It also produces a circularly polarized wave across the bandwidth. A unidirectional beam can be created by using a backing structure like reflector or cavity. Maintaining the desired performance when using a traditional metallic/PEC reflector, especially at small separation distances is difficult [1].

Reference

[1] H. Nakano, K. Kikkawa, N. Kondo, Y. Iitsuka, J. Yamauchi, "Low-Profile Equiangular Spiral Antenna Backed by an EBG Reflector," IEEE Transactions on Antennas and Propagation, vol.57, no.5, pp.1309-1318, May 2009.

See Also

"Double-Slot Cavity Patch on TMM10 Substrate" on page 5-272

Impedance Matching of Non-resonant (Small) Monopole

This example shows how to design a double tuning L-section matching network between a resistive source and capacitive load in the form of a small monopole. The L-section consists of two inductors. The network achieves conjugate match and guarantees maximum power transfer at a single frequency. This example requires the following product:

- RF Toolbox™

Create Monopole

Create a quarter-wavelength monopole antenna with the resonant frequency around 1 GHz. For the purpose of this example, we choose a square ground plane of side 0.75λ .

```
fres = 1e9;
speedOfLight = physconst('lightspeed');
lambda = speedOfLight/fres;
L = 0.25*lambda;
dp = monopole('Height',L,'Width',L/50,...
             'GroundPlaneLength',0.75*lambda,...
             'GroundPlaneWidth',0.75*lambda);
```

Calculate Monopole Impedance

Specify the source (generator) impedance, the reference (transmission line) impedance and the load (antenna) impedance. In this example, the load Z_{l0} will be the non-resonant (small) monopole at the frequency of 500 MHz, which is the half of the resonant frequency. The source has the equivalent impedance of 50 ohms.

```
f0 = fres/2;
Zs = 50;
Z0 = 50;
Zl0 = impedance(dp,f0);
Rl0 = real(Zl0);
Xl0 = imag(Zl0);
```

Define the number of frequency points for the analysis and a frequency band centered at 500 MHz.

```
Npts = 30;
fspan = 0.1;
fmin = f0*(1 - (fspan/2));
fmax = f0*(1 + (fspan/2));
freq = unique([f0 linspace(fmin,fmax,Npts)]);
```

Understand Load Behavior using Reflection Coefficient and Power Gain

Calculate the load reflection coefficient and the power gain between the source and the antenna.

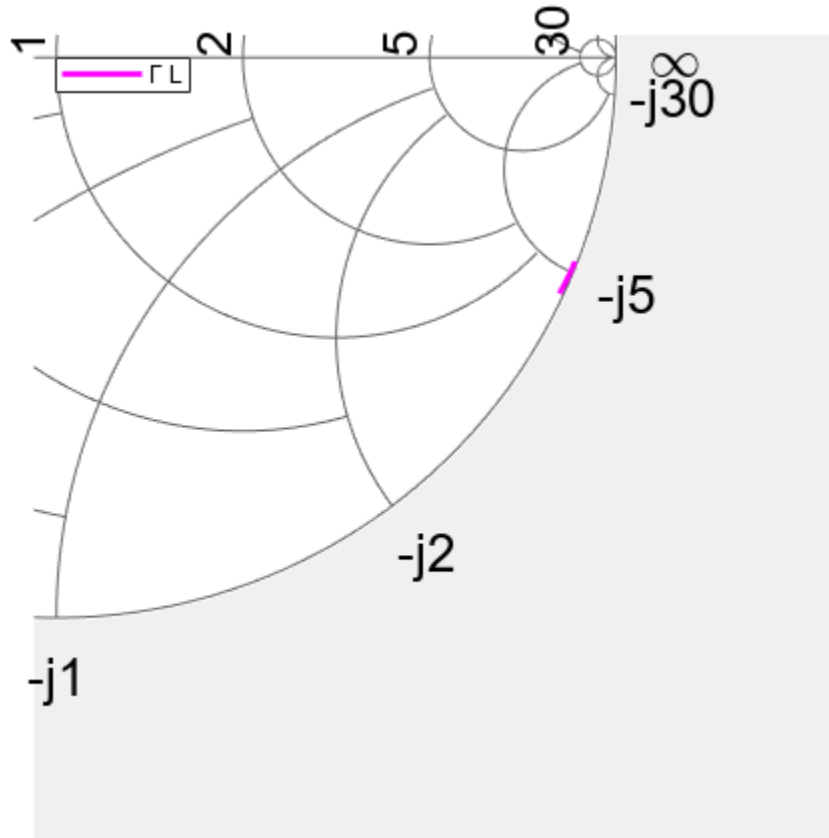
```
S = sparameters(dp, freq);
GammaL = rfparam(S,1,1);
Gt = 10*log10(1 - abs(GammaL).^2);
```

Plotting the input reflection coefficient on a Smith chart shows the capacitive behavior of this antenna around the operating frequency of 500 MHz. The center of the Smith chart represents the matched condition to the reference impedance. The location of the reflection coefficient trace around $-j5.0\Omega$ confirms that there is a severe impedance mismatch.

```

fig1 = figure;
hsm = smithplot(fig1,freq,GammaL,'LineWidth',2.0,'Color','m',...
    'View','bottom-right','LegendLabels',{'#Gamma L'});

```

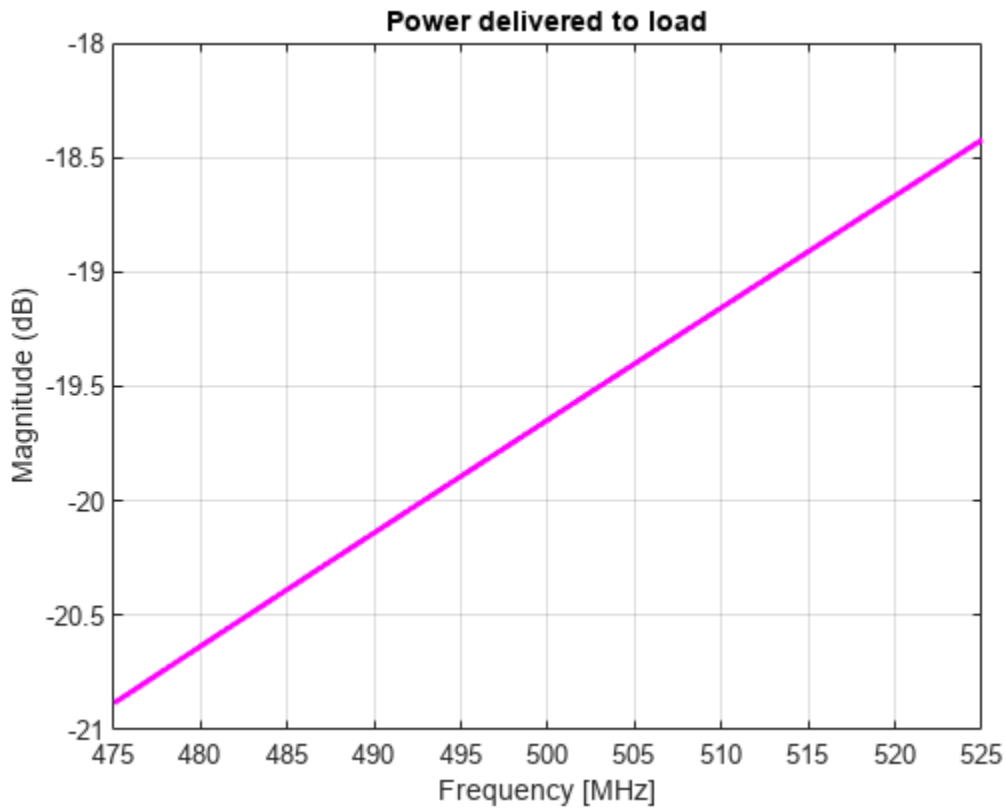


Plot the power delivered to the load.

```

fig2 = figure;
plot(freq*1e-6,Gt,'m','LineWidth',2);
grid on
xlabel('Frequency [MHz]')
ylabel('Magnitude (dB)')
title('Power delivered to load')

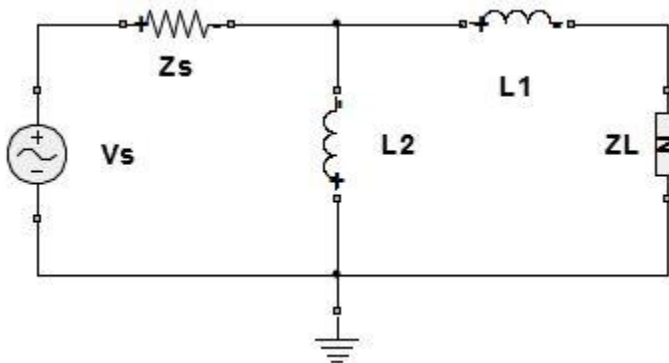
```



As the power gain plot shows, there is approximately a 20 dB power loss around the operating frequency (500 MHz).

Design Matching Network

The matching network must ensure maximum power transfer at 500 MHz. The L-section double tuning network achieves this goal [1]. The network topology, shown in the figure that follows consists of an inductor in series with the antenna, that cancels the large capacitance at 500 MHz, and a shunt inductor that further boosts the output resistance to match the source impedance of 50Ω .



```

omega0 = 2*pi*f0;
L2 = (1/omega0)*sqrt((Zs*Rl0)/(1-(Rl0/Zs)));
L1 = (-Xl0/omega0) - (L2/2) - sqrt((L2^2/4)-(((Rl0)^2)/omega0^2));

```

Create Matching Network and Calculate S-parameters

The matching network circuit consists of two inductors whose inductance values have been calculated above. Create the matching network using the RF Toolbox™ and calculate the S-parameters of this network over the frequency band centered at the operating frequency.

```

IND1 = inductor(L1,'L1');
IND2 = inductor(L2,'L2');
MatchingNW = circuit('double_tuning');
add(MatchingNW,[0 1],IND2);
add(MatchingNW,[1 2],IND1);
setports(MatchingNW,[1 0],[2 0]);
Smatchnw = sparameters(MatchingNW,freq);

```

The circuit element representation of the matching network is shown below.

```

disp(MatchingNW)

circuit: Circuit element

ElementNames: {'L2' 'L1'}
Elements: [1x2 inductor]
Nodes: [0 1 2]
Name: 'double_tuning'
NumPorts: 2
Terminals: {'p1+' 'p2+' 'p1-' 'p2-'}

```

Reflection Coefficient and Power Gain with Matching Network

Calculate the input reflection coefficient/power gain for the antenna load with the matching network.

```

Zl = impedance(dp,freq);
GammaIn = gammain(Smatchnw,Zl);
Gtmatch = powergain(Smatchnw,Zs,Zl,'Gt');
Gtmatch = 10*log10(Gtmatch);

```

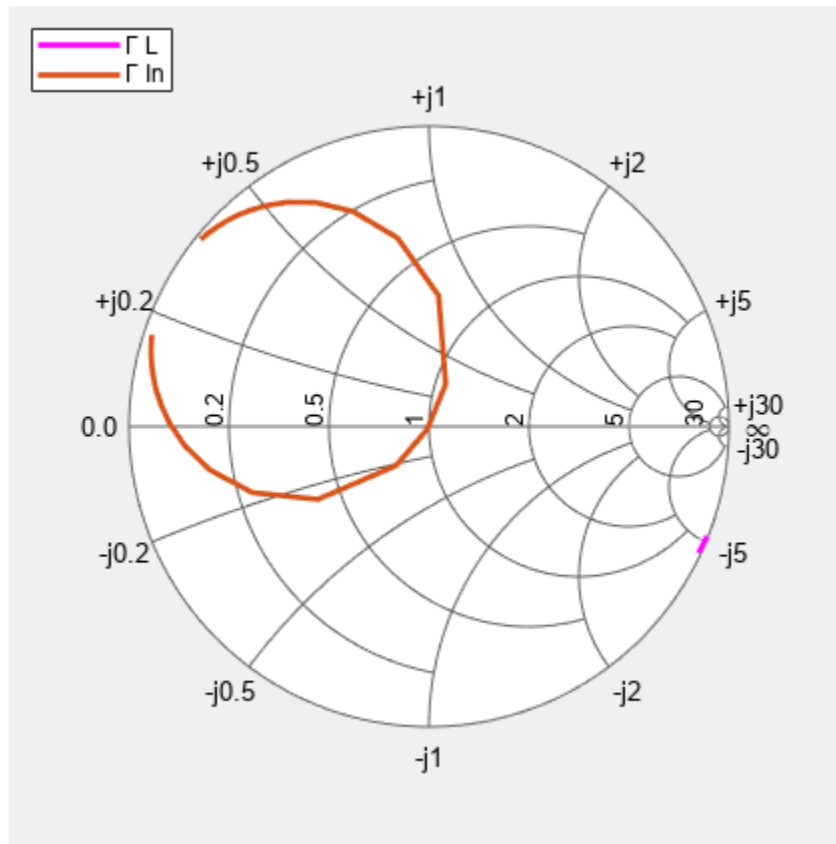
Compare Results

Plot the input reflection coefficient and power delivered to the antenna, with and without the matching network. The Smith chart plot shows the reflection coefficient trace going through its center thus confirming the match. At the operation frequency of 500 MHz, the generator transfers maximum power to the antenna. The match degrades on either side of the operating frequency.

```

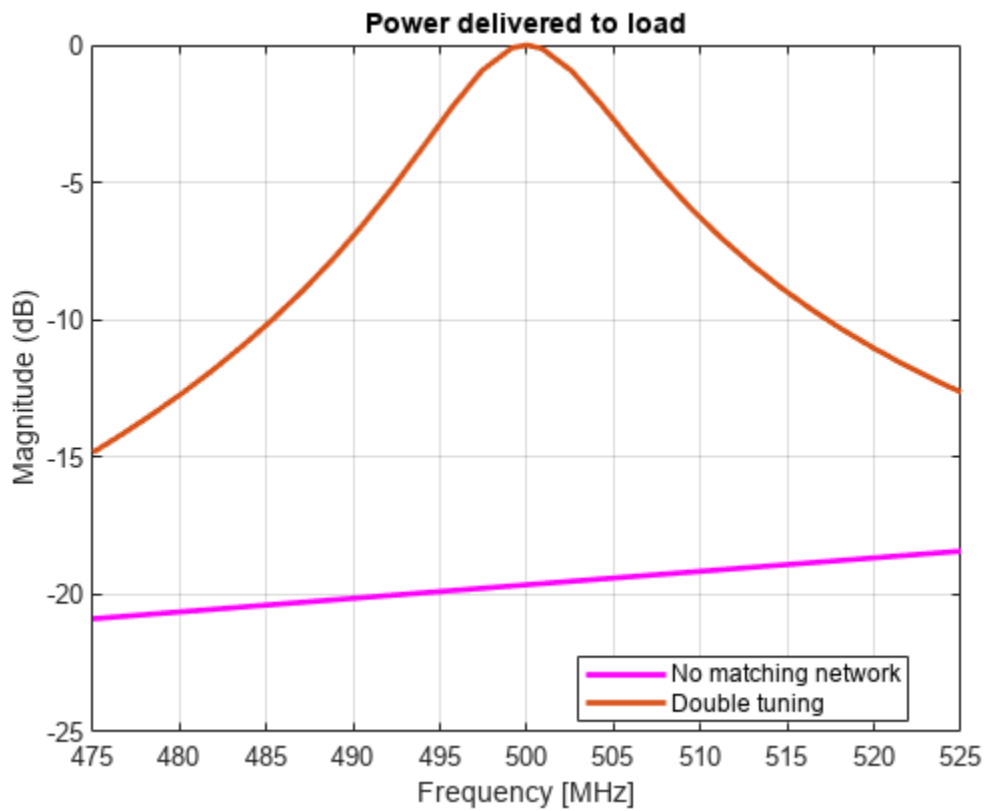
add(hsm,freq,GammaIn);
hsm.LegendLabels(2) = {'#Gamma In'};
hsm.View = 'full';

```



Plot the power delivered to the load.

```
figure(fig2)
hold on
plot(freq*1e-6,Gtmatch,'LineWidth',2);
axis([min(freq)*1e-6,max(freq)*1e-6,-25,0])
legend('No matching network','Double tuning','Location','Best');
```



References

[1] M. M. Weiner, Monopole Antennas, Marcel Dekker, Inc., CRC Press, Rev. Exp edition, New York, pp.110-118, 2003.

See Also

“Design Internally Matched Ultra-Wideband Vivaldi Antenna” on page 5-410

Direct Search Based Optimization of Six-Element Yagi-Uda Antenna

This example optimizes a 6-element Yagi-Uda antenna for both directivity and 300Ω input match using a global optimization technique. The radiation patterns and input impedance of antennas are sensitive to the parameters that define their shapes. The multidimensional surface over which such optimizations must be performed have multiple local optima. This makes the task of finding the right set of parameters satisfying the optimization goals particularly challenging and requires the use of global optimization techniques. One such technique is pattern search, a direct search based optimization technique that has yielded impressive results for antenna design optimization.

The Yagi-Uda antenna is a widely used radiating structure for a variety of applications in commercial and military sectors. This antenna has been popular for reception of TV signals in the VHF-UHF range of frequencies [1]. The Yagi is a directional traveling-wave antenna with a single driven element, usually a folded dipole or a standard dipole, which is surrounded by several passive dipoles. The passive elements form the *reflector* and *director*. These names identify the positions relative to the driven element. The reflector dipole is behind the driven element in the direction of the back lobe of the antenna radiation, while the director is in front of the driven element, in the direction where a main beam forms.

Design Parameters

Choose initial design parameters in the center of the VHF band [2]. The datasheet lists a 50Ω input impedance after taking into account a balun. Our model does not account for the presence of the balun and therefore will match to the typical folded dipole input impedance of 300Ω .

```
fc = 165e6;
wirediameter = 12.7e-3;
c = physconst('lightspeed');
lambda = c/fc;
Z0 = 300;
BW = 0.05*fc;
fmin = fc - 2*(BW);
fmax = fc + 2*(BW);
Nf = 101;
freq = linspace(fmin, fmax, Nf);
```

Create Yagi-Uda Antenna

The driven element for the Yagi-Uda antenna is a folded dipole. This is a standard exciter for such an antenna. Adjust the length and width parameters of the folded dipole. Since we model cylindrical structures as equivalent metal strips, the width is calculated using a utility function available in the Antenna Toolbox™. The length is chosen to be $\lambda/2$ at the design frequency.

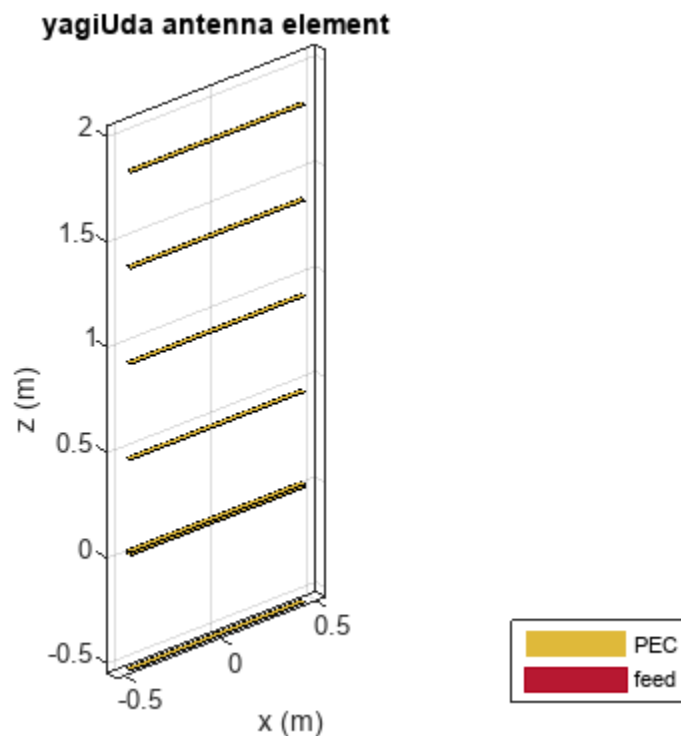
```
d = dipoleFolded;
d.Length = lambda/2;
d.Width = cylinder2strip(wirediameter/2);
d.Spacing = d.Length/60;
```

Create a Yagi-Uda antenna with the exciter as the folded dipole. Choose the reflector and director length to be $\lambda/2$. Set the number of directors to four. Choose the reflector and director spacing to be 0.3λ , 0.25λ respectively. These choices are an initial guess and will serve as a start point for the optimization procedure. Show the initial design.

```

Numdirs = 4;
refLength = 0.5;
dirLength = 0.5*ones(1,Numdirs);
refSpacing = 0.3;
dirSpacing = 0.25*ones(1,Numdirs);
exLength = d.Length/lambda;
exSpacing = d.Spacing/lambda;
initialdesign = [refLength dirLength refSpacing dirSpacing exLength exSpacing].*lambda;
yagidesign = yagiUda;
yagidesign.Exciter = d;
yagidesign.NumDirectors = Numdirs;
yagidesign.ReflectorLength = refLength*lambda;
yagidesign.DirectorLength = dirLength.*lambda;
yagidesign.ReflectorSpacing = refSpacing*lambda;
yagidesign.DirectorSpacing = dirSpacing*lambda;
show(yagidesign)

```



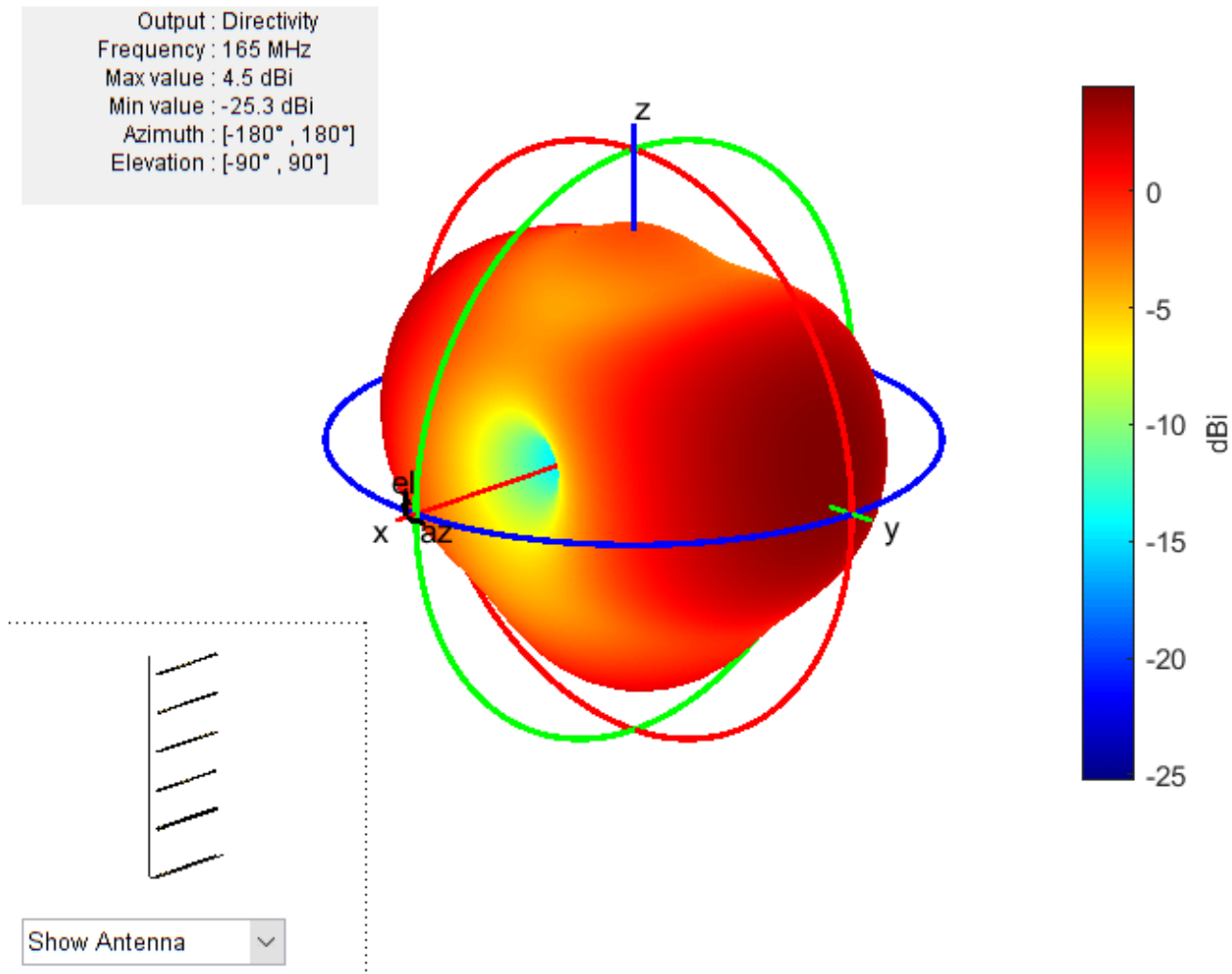
Plot Radiation Pattern at Design Frequency

Prior to executing the optimization process, plot the radiation pattern for the initial guess in 3D.

```

fig1 = figure;
pattern(yagidesign,fc);

```



This initial Yagi-Uda antenna does not have a higher directivity in the preferred direction, meaning at zenith (elevation = 90 deg) and is therefore a poorly designed radiator.

Set Up Optimization

Use the following variables as control variables for the optimization:

- Reflector length (1 variable)
- Director lengths (4 variables)
- Reflector spacing (1 variable)
- Director spacings (4 variables)
- Exciter length (1 variable)
- Exciter spacing (1 variable)

In terms of a single vector parameter `controlVals`, set

- Reflector length = `controlVals(1)`

- Director lengths = `controlVals(2:5)`
- Reflector spacing = `controlVals(6)`
- Director spacings = `controlVals(7:10)`
- Exciter length = `controlVals(11)`
- Exciter spacing = `controlVals(12)`

In terms of `controlVals`, set an objective function that aims to have a large directivity value in the 90 degree direction, a small value in the -90 degree direction, and a large value of maximum power between the elevation beamwidth angle bounds. In addition to the directivity goal an impedance match condition is also included as a constraint. Any constraint violations will penalize the objective.

type `yagi_objective_function_direct.m`

```
function objectivevalue = yagi_objective_function_direct(y,controlVals,fc,BW,ang,Z0,constraints)
% YAGI_OBJECTIVE_FUNCTION_DIRECT returns the objective for a 6 element Yagi
% OBJECTIVE_VALUE =
% YAGI_OBJECTIVE_FUNCTION_DIRECT(Y,CONTROLVALS,FREQ,ANG,Z0,constraints), assigns
% the appropriate parasitic dimensions, CONTROLVALS to the Yagi antenna Y,
% and uses the frequency FREQ, angle pair,ANG, reference impedance Z0 and
% the constraints to calculate the objective function value.

% The YAGI_OBJECTIVE_FUNCTION_DIRECT function is used for an internal example.
% Its behavior may change in subsequent releases, so it should not be
% relied upon for programming purposes.

% Copyright 2018 The MathWorks, Inc.

y.ReflectorLength = controlVals(1);
y.DirectorLength = controlVals(2:y.NumDirectors+1);
y.ReflectorSpacing = controlVals(y.NumDirectors+2);
y.DirectorSpacing = controlVals(y.NumDirectors+3:end-2);
y.Exciter.Length = controlVals(end-1);
y.Exciter.Spacing = controlVals(end);

% Unpack constraints
Gmin = constraints.Gmin;
Gdev = constraints.Gdeviation;
FBmin = constraints.FBmin;
S11min = constraints.S11min;
K = constraints.Penalty;

% Calculate antenna port and field parameters
output = analyzeAntenna(y,fc,BW,ang,Z0);

% Form objective function
output1 = output.MaxDirectivity+output.MismatchLoss;    % Directivity/Gain at zenith

% Gain constraint, e.g. G > 10
c1 = 0;
if output1<Gmin
    c1 = Gmin-output1;
end

% Gain deviation constraint, abs(G-Gmin)<0.1;
c1_dev = 0;
```

```

if abs(output1-Gmin)>Gdev
    c1_dev = -Gdev + abs(output1-Gmin);
end

% Front to Back Ratio constraint, e.g. F/B > 15
c2 = 0;
if output.FB < FBmin
    c2 = FBmin-output.FB;
end

% Reflection Coefficient, S11 < -10
c3 = 0;
if output.S11 > S11min
    c3 = -S11min + output.S11;
end

% Form the objective + constraints
objectivevalue = -output1 + max(0,(c1+c1_dev+c2+c3))*K;
end

function output = analyzeAntenna(ant,fc,BW,ang,Z0)
%ANALYZEANTENNA calculate the objective function
% OUTPUT = ANALYZEANTENNA(Y,FREQ,BW,ANG,Z0) performs analysis on the
% antenna ANT at the frequency, FC, and calculates the directivity at the
% angles specified by ANG and the front-to-back ratio. The reflection
% coefficient relative to reference impedance Z0, and impedance are
% computed over the bandwidth BW around FC.

fmin = fc - (BW/2);
fmax = fc + (BW/2);
Nf = 5;
freq = unique([fc,linspace(fmin,fmax,Nf)]);
fcIdx = freq==fc;
s = sparameters(ant,freq,Z0);
Z = impedance(ant,fc);
az = ang(1,:);
el = ang(2,:);
Dmax = pattern(ant,fc,az(1),el(1));
Dback = pattern(ant,fc,az(2),el(2));

% Calculate F/B
F_by_B = Dmax-Dback;

% Compute S11 and mismatch loss
s11 = rfparam(s,1,1);
S11 = max(20*log10(abs(s11)));
T = mean(10*log10(1 - (abs(s11)).^2));

% Form the output structure
output.MaxDirectivity= Dmax;
output.BackLobeLevel = Dback;
output.FB = F_by_B;
output.S11 = S11;
output.MismatchLoss = T;
output.Z = Z;
end

```

Set bounds on the control variables.

```

refLengthBounds = [0.3;                                % lower bound on reflector length
                  0.9];                                % upper bound on reflector spacing
dirLengthBounds = [0.3 0.3 0.3 0.3;                  % lower bound on director length
                  0.7 0.7 0.7 0.7];                  % upper bound on director length
refSpacingBounds = [0.05;                             % lower bound on reflector spacing
                  0.35];                             % upper bound on reflector spacing
dirSpacingBounds = [0.05 0.05 0.05 0.05;            % lower bound on director spacing
                  0.2 0.2 0.3 0.3];                  % upper bound on director spacing
exciterLengthBounds = [0.45;                          % lower bound on exciter length
                  0.6];                              % upper bound on exciter length
exciterSpacingBounds = [.004;
                  .009];

LB = [refLengthBounds(1),dirLengthBounds(1,:) refSpacingBounds(1) dirSpacingBounds(1,:) exciterL
UB = [refLengthBounds(2),dirLengthBounds(2,:) refSpacingBounds(2) dirSpacingBounds(2,:) exciterL
parameterBounds.LB = LB;
parameterBounds.UB = UB;
ang = [0 0;90 -90];                                % azimuth,elevation angles for main lobe and back lobe [az

```

Direct Search Based Optimization

The Global Optimization Toolbox™ provides a direct search based optimization function called `patternsearch`. We use this function with options specified with the `psoptimset` function. At every iteration, plot the best value of the objective function and limit the total number of iterations to 300. Pass the objective function to the `patternsearch` function by using an anonymous function together with the bounds and the options structure. The objective function used during the optimization process by `patternsearch` is available in the file `yagi_objective_function_direct`.

The evaluation of the directivity in different directions corresponding to the angular region defined for maximum radiation as well as maximum sidelobe and the backlobe level is given in the function `calculate_objectives` available within `yagi_objective_function_direct`.

```

% Optimizer options
optimizerparams = optimoptions(@patternsearch);
optimizerparams.UseCompletePoll = true;
optimizerparams.PlotFcns = @psplotbestf;
optimizerparams.UseParallel = true;
optimizerparams.Cache = 'on';
optimizerparams.MaxIter = 100;
optimizerparams.FunctionTolerance = 1e-2;

% Antenna design parameters
designparams.Antenna = yagidesign;
designparams.Bounds = parameterBounds;

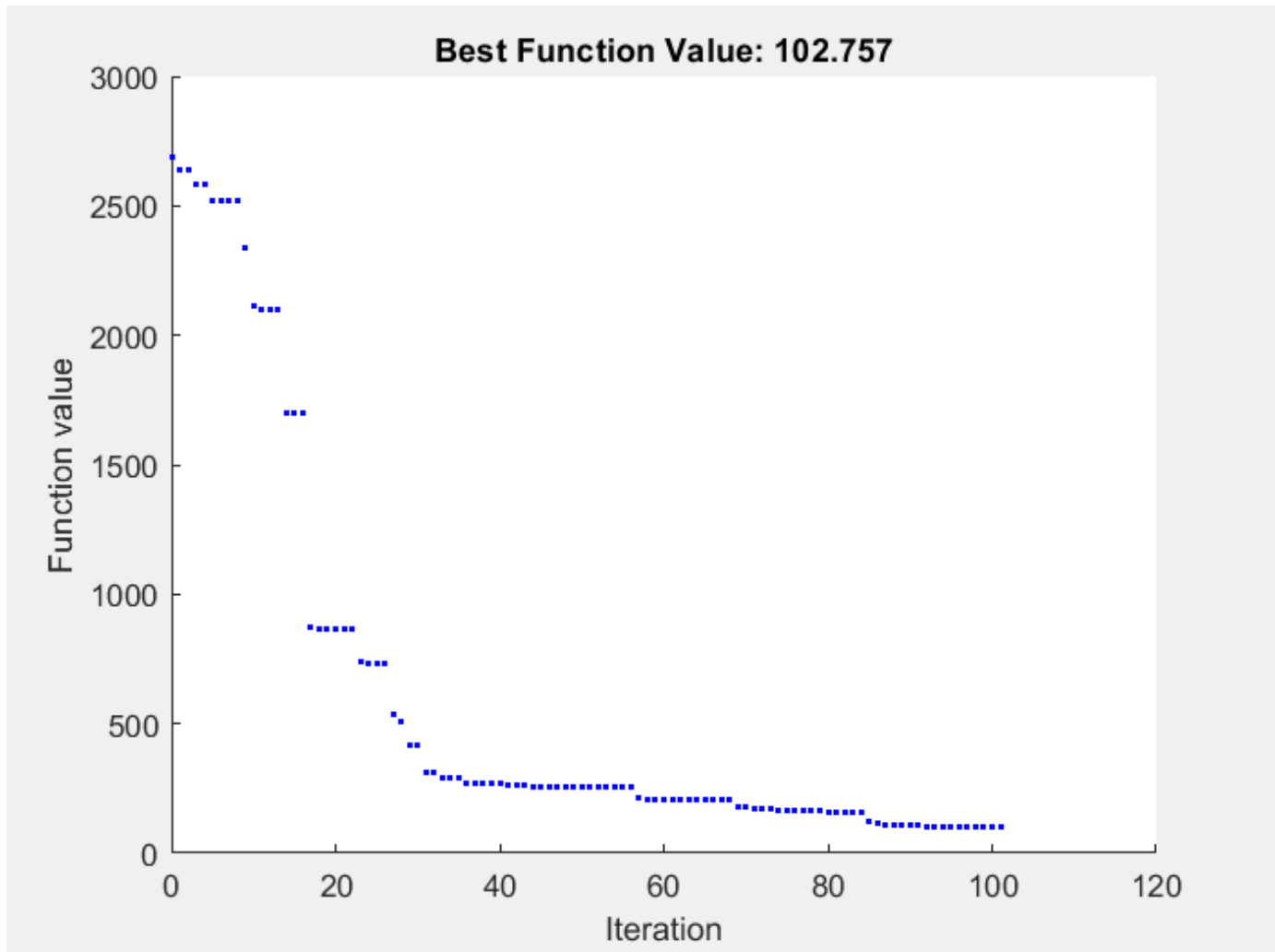
% Analysis parameters
analysisparams.CenterFrequency = fc;
analysisparams.Bandwidth = BW;
analysisparams.ReferenceImpedance = Z0;
analysisparams.MainLobeDirection = ang(:,1);
analysisparams.BackLobeDirection = ang(:,2);

% Set constraints
constraints.S11min = -10;
constraints.Gmin = 10.5;
constraints.Gdeviation = 0.1;
constraints.FBmin = 15;

```

```
constraints.Penalty = 50;
optimdesign = optimizeAntennaDirect(designparams,analysisparams,constraints,optimizerparams);
```

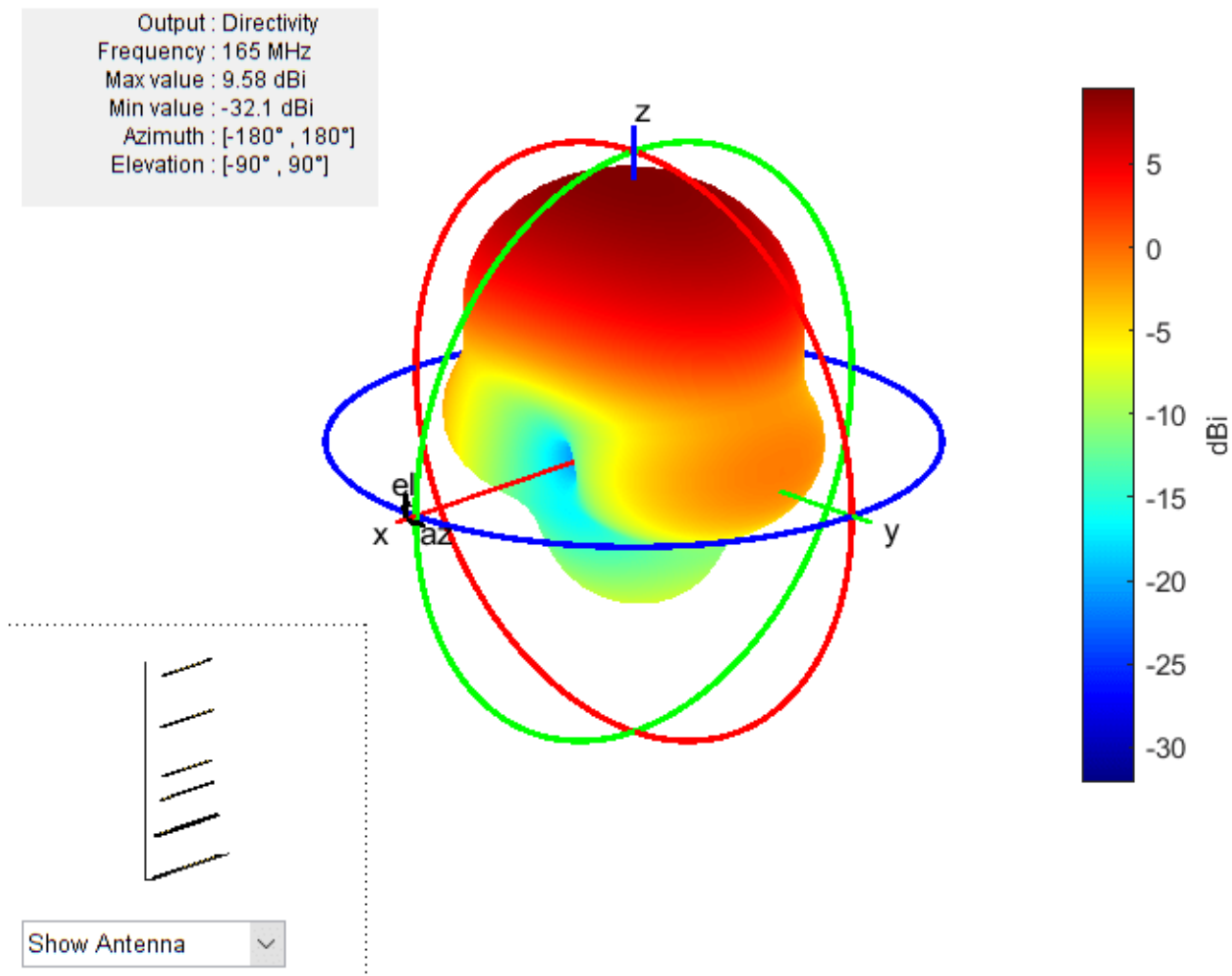
```
Starting parallel pool (parpool) using the 'Processes' profile ...
Connected to the parallel pool (number of workers: 8).
Maximum number of iterations exceeded: increase options.MaxIterations.
```



Plot Optimized Pattern

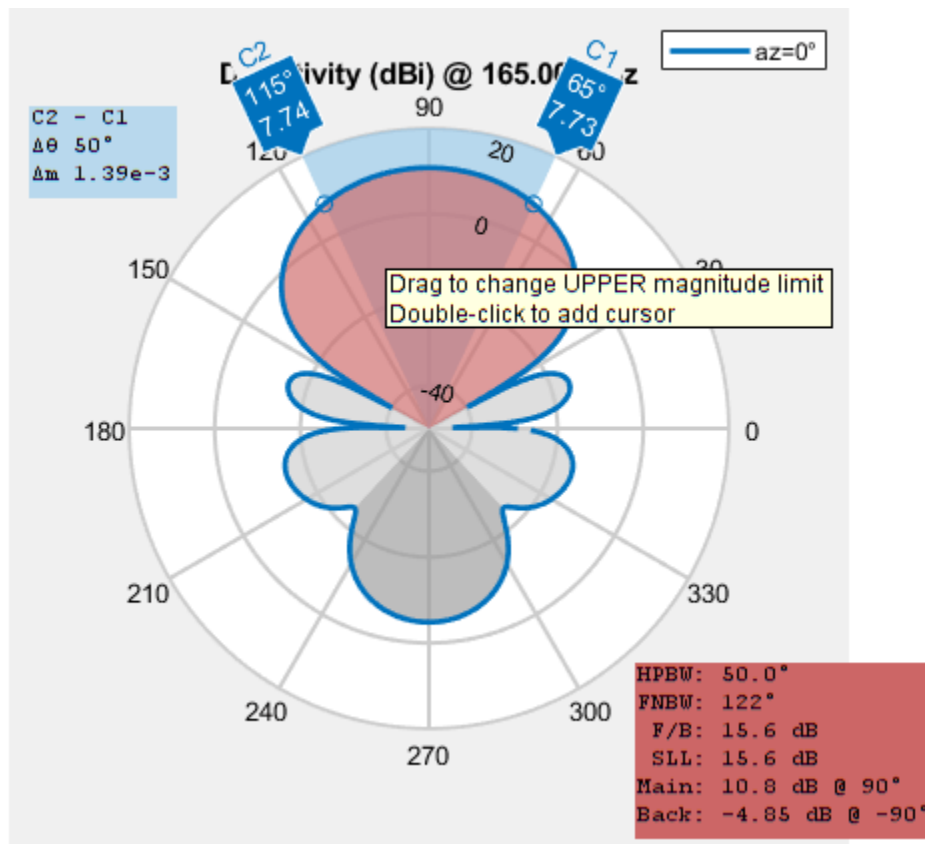
Plot the optimized antenna pattern at the design frequency.

```
yagidesign.ReflectorLength = optimdesign(1);
yagidesign.DirectorLength = optimdesign(2:5);
yagidesign.ReflectorSpacing = optimdesign(6);
yagidesign.DirectorSpacing = optimdesign(7:10);
yagidesign.Exciter.Length = optimdesign(11);
yagidesign.Exciter.Spacing = optimdesign(12);
fig2 = figure;
pattern(yagidesign,fc)
```

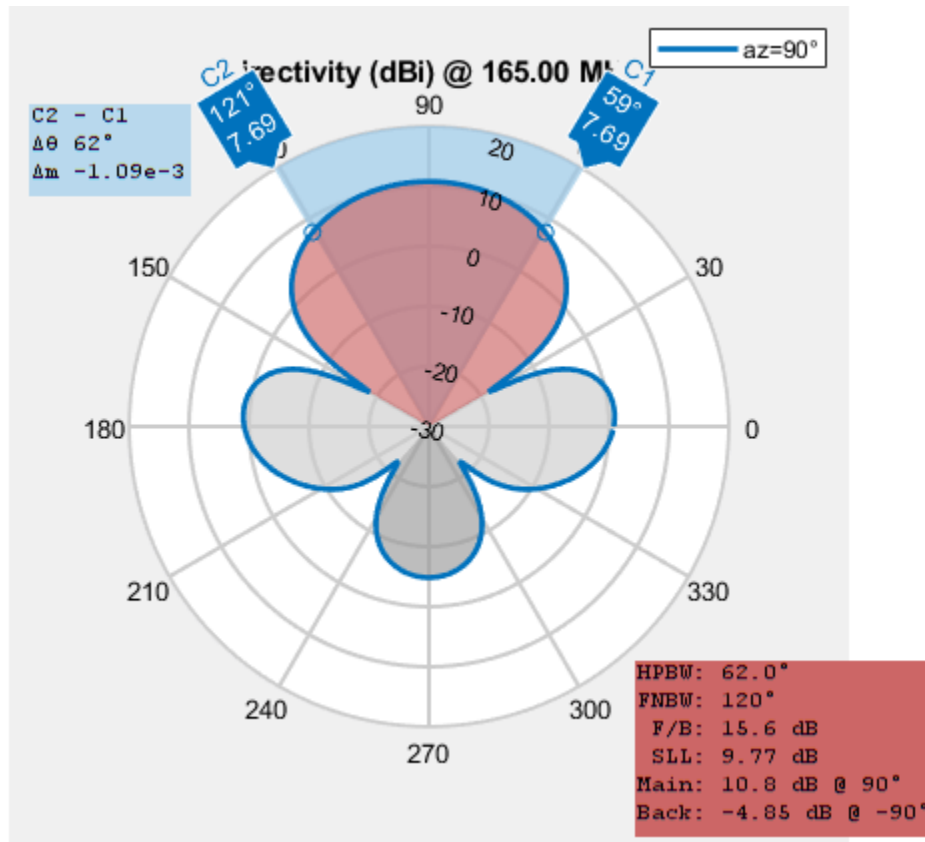


E and H-Plane Cuts of Pattern

To obtain a better insight into the behavior in the two orthogonal planes, plot the normalized magnitude of the electric field in the E and H-planes, i.e. azimuth = 0 and 90 deg respectively. Enable the antenna metrics on the polar pattern plots to establish the directivity at zenith, Front-to-Back ratio, and the beamwidth in E and H-planes.



```
% fig3 = figure;
% patternElevation(yagidesign,fc,0,'Elevation',0:1:359);
% pE = polarpattern('gco');
% pE.AntennaMetrics = 1;
```



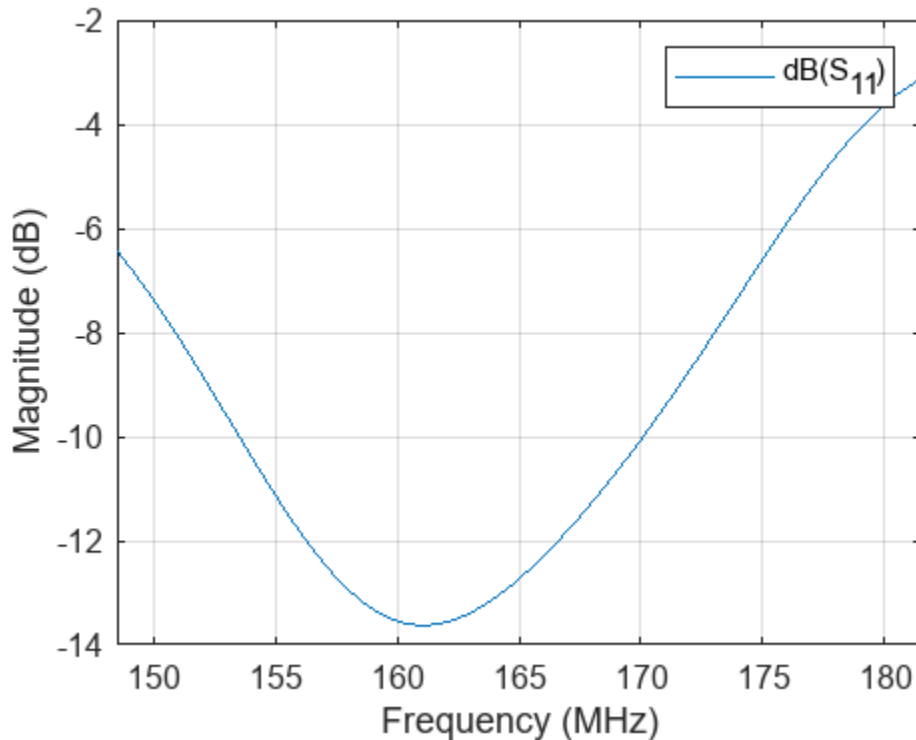
```
% fig4 = figure;
% patternElevation(yagidesign,fc,90,'Elevation',0:1:359);
% pH = polarpattern('gco');
% pH.AntennaMetrics = 1;
```

The optimized design shows a significant improvement in the radiation pattern. There is higher directivity achieved in the desired direction toward zenith. The back lobe is small resulting in a good front to back ratio for this antenna.

Input Reflection Coefficient of Optimized Antenna

The input reflection coefficient for the optimized Yagi-Uda antenna is computed and plotted relative to the reference impedance of 50Ω . A value of -10 dB or lower is considered to be a good impedance match.

```
s = sparameters(yagidesign,freq,Z0);
fig5 = figure;
rfplot(s);
```



Comparison with Manufacturer Data Sheet

The optimized Yagi-Uda antenna achieves a forward directivity greater than 10 dBi, which translates to a value greater than 8 dBd (relative to a dipole). This is close to the gain value reported by the datasheet (8.5 dBd). The F/B ratio is greater than 15 dB. The optimized Yagi-Uda antenna has a E-plane and H-plane beamwidth that compare favorably to the datasheet listed values of 54 degrees and 63 degrees respectively. The design achieves a good impedance match to 300Ω , and has a -10 dB bandwidth of approximately 8%.

```
datasheetparam = {'Gain (dBi)'; 'F/B'; 'E-plane Beamwidth (deg.)'; 'H-plane Beamwidth (deg.)'; 'Impedance Bandwidth (%)'};
datasheetvals = [10.5, 16, 54, 63, 10]';
optimdesignvals = [10.59, 15.6, 50, 62, 12.1]';
Tdatasheet = table(datasheetvals, optimdesignvals, 'RowNames', datasheetparam)
```

Tdatasheet=5x2 table

	datasheetvals	optimdesignvals
Gain (dBi)	10.5	10.59
F/B	16	15.6
E-plane Beamwidth (deg.)	54	50
H-plane Beamwidth (deg.)	63	62
Impedance Bandwidth (%)	10	12.1

Tabulating Initial and Optimized Design

Tabulate the initial design guesses and the final optimized design values.

```
yagiparam= {'Reflector Length';  
           'Director Length - 1'; 'Director Length - 2';  
           'Director Length - 3'; 'Director Length - 4';  
           'Reflector Spacing';   'Director Spacing - 1';  
           'Director Spacing - 2'; 'Director Spacing - 3';  
           'Director Spacing - 4'; 'Exciter Length';  
           'Exciter Spacing'};  
initialdesign = initialdesign';  
optimdesign = optimdesign';  
Tgeometry = table(initialdesign,optimdesign,'RowNames',yagiparam)
```

Tgeometry=12x2 table

	initialdesign	optimdesign
Reflector Length	0.90846	0.90846
Director Length - 1	0.90846	0.72096
Director Length - 2	0.90846	0.65846
Director Length - 3	0.90846	0.72096
Director Length - 4	0.90846	0.65846
Reflector Spacing	0.54508	0.42008
Director Spacing - 1	0.45423	0.36338
Director Spacing - 2	0.45423	0.23838
Director Spacing - 3	0.45423	0.51673
Director Spacing - 4	0.45423	0.53236
Exciter Length	0.90846	0.84596
Exciter Spacing	0.015141	0.016118

Reference

- [1] C. A. Balanis, Antenna Theory. Analysis and Design, p. 514, Wiley, New York, 3rd Edition, 2005
- [2] Online at: S.6Y-165

See Also

“Surrogate Based Optimization Design of Six-Element Yagi-Uda Antenna” on page 5-459

Antenna Diversity Analysis for 800 MHz MIMO

This example analyzes a 2-antenna diversity scheme to understand the effect that position, orientation and frequency have on received signals. The analysis is performed under the assumptions that impedance matching is not achieved and mutual coupling is taken into account [1].

Frequency Band Parameters

Define the operating frequency, analysis bandwidth and calculate the wavelength in free space.

```
freq = 800e6;  
c = physconst('lightspeed');  
lambda = c/freq;  
BW_frac = .1;  
fmin = freq - BW_frac*freq;  
fmax = freq + BW_frac*freq;
```

Create Two Identical Dipoles

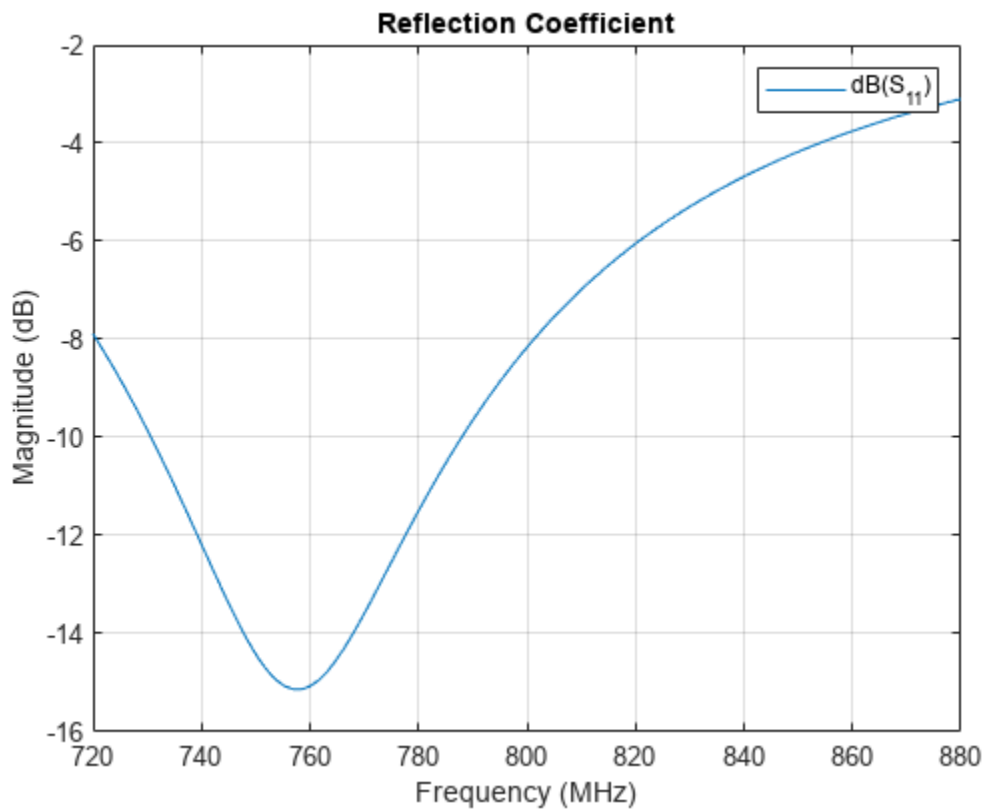
Use the dipole antenna element from the Antenna Toolbox™ library and create 2 identical thin dipoles of length $\lambda/2$.

```
d1 = dipole('Length',lambda/2,'Width',lambda/200);  
d2 = dipole('Length',lambda/2,'Width',lambda/200);
```

Plot Input Reflection Coefficient of Isolated Dipole

Calculate the input reflection coefficient of an isolated dipole and plot it to confirm the lack of impedance match at 800MHz.

```
Numfreq = 101;  
f = linspace(fmin,fmax,Numfreq);  
S = sparameters(d1,f);  
DipoleS11Fig = figure;  
rfplot(S,1,1)  
title('Reflection Coefficient')
```

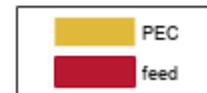
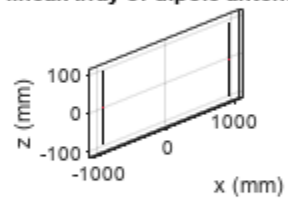


Create Two-Element Array

Create the two-element antenna diversity system and position the 2 antennas apart by 5λ .

```
range = 5*lambda;  
l = linearArray;  
l.Element = [d1 d2];  
l.ElementSpacing = range;  
show(l)  
view(-80,4)
```

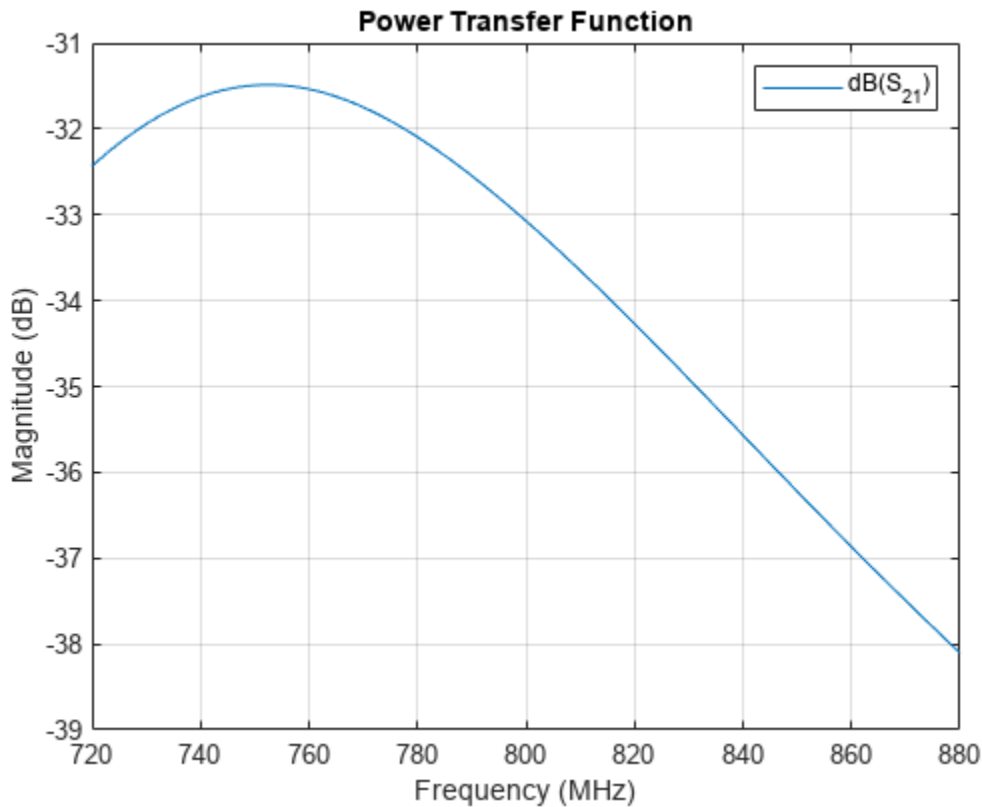
linearArray of dipole antennas



Power Transfer Function

Calculate and plot the power transfer function (S21 in dB) for two antennas. To do so, calculate the scattering parameters for the system and plot S21 over the entire frequency range.

```
S = sparameters(l,f);  
ArrayS21Fig = figure;  
rfplot(S,2,1)  
title('Power Transfer Function')
```



The response peak is clearly not at 800 MHz. In addition, note the loss in signal strength due to attenuation in free space.

Vary spatial orientation of dipole

Power transfer between the two antennas can now be investigated as a function of antenna orientation. A correlation coefficient is used in MIMO systems to quantify the system performance. Two approaches to calculate the correlation coefficient exist; using the far-field behavior and using the S-parameters. The field-based approach involves numerical integration. The calculation suggested in this example uses the function `correlation` available in the Antenna Toolbox™ and based on the S-parameters approach [1]. By rotating one antenna located on the positive x-axis, we change its polarization direction and find the correlation

```

numpos = 101;
orientation = linspace(0,90,numpos);
S21_TiltdB = nan(1,numel(orientation));
Corr_TiltdB = nan(1,numel(orientation));
fig1 = figure;
for i = 1:numel(orientation)
    d2.Tilt = orientation(i);
    l.Element(2) = d2;
    S = sparameters(l,freq);
    Corr = correlation(l,freq,1,2);
    S21_TiltdB = 20*log10(abs(S.Parameters(2,1,1)));
    Corr_TiltdB(i) = 20*log10(Corr);
    figure(fig1);
    plot(orientation,S21_TiltdB,orientation,Corr_TiltdB,'LineWidth',2)

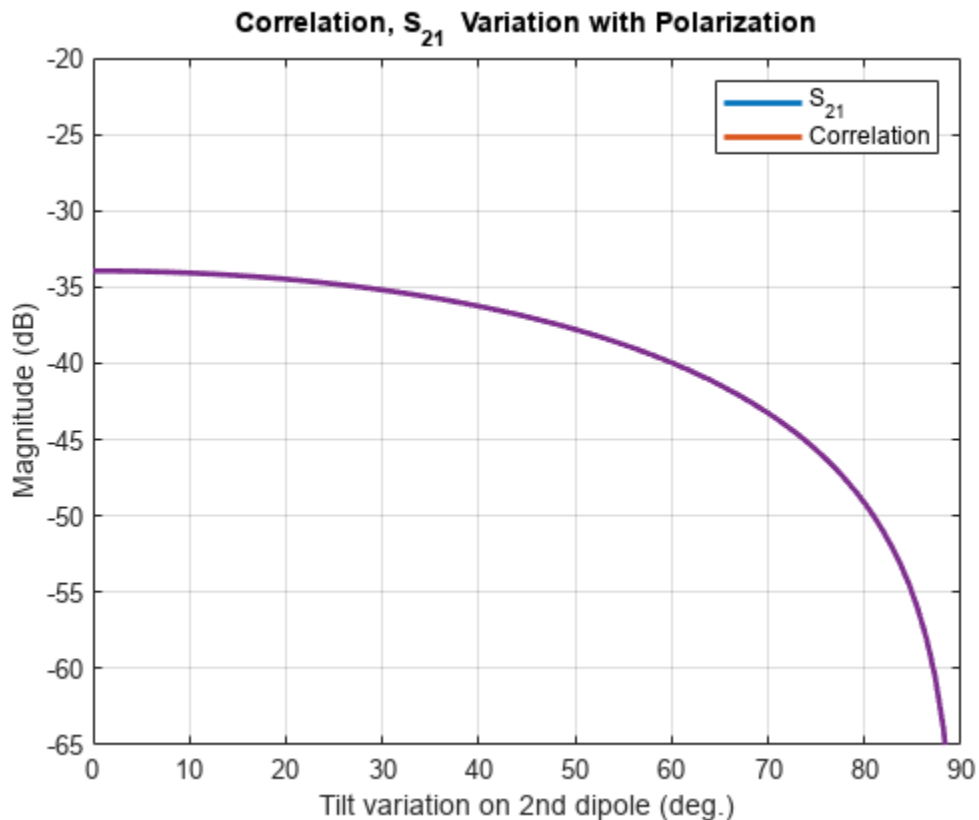
```



```

grid on
axis([min(orientation) max(orientation) -65 -20]);
xlabel('Tilt variation on 2nd dipole (deg.)')
ylabel('Magnitude (dB)')
title('Correlation, S21 Variation with Polarization')
drawnow
end
legend('S21', 'Correlation');

```



We observe that the power transfer function and the correlation function between the two antennas are identical as the antenna orientation changes for one of the dipoles.

Vary Spacing between Antennas

Restore both dipoles so that they are parallel to each other. Run a similar analysis by changing the spacing between the 2 elements.

```

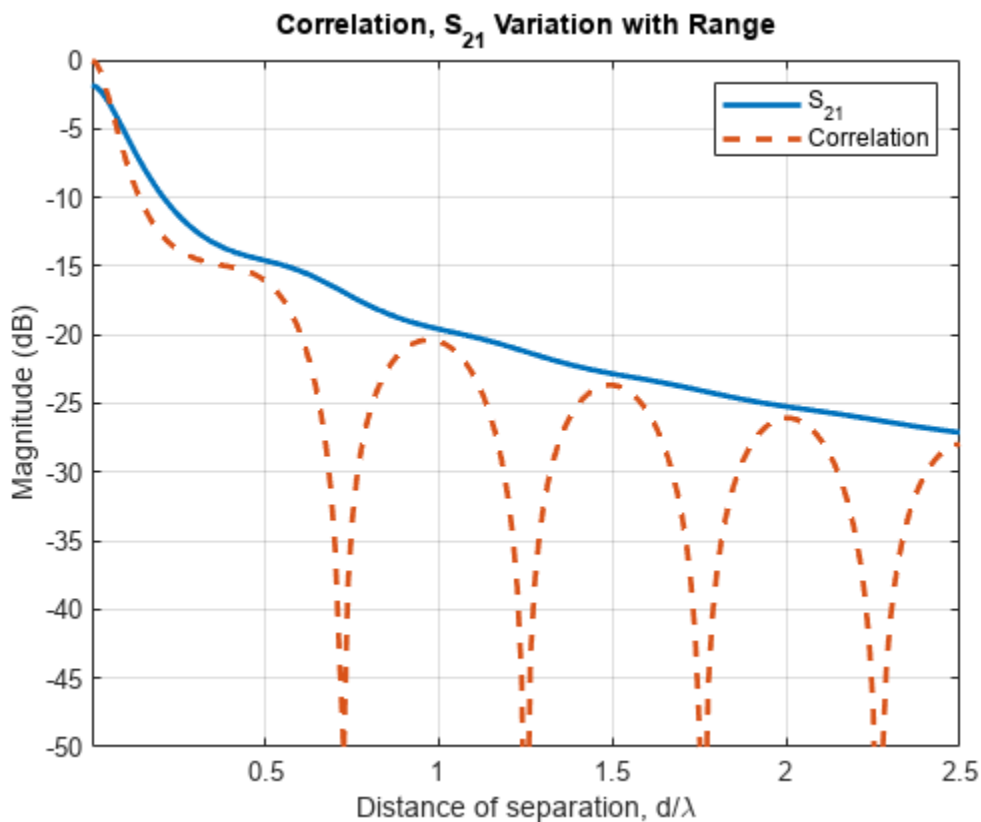
d2.Tilt = 0;
l.Element = [d1 d2];
Nrange = 201;
Rmin = 0.001*lambda;
Rmax = 2.5*lambda;
range = linspace(Rmin,Rmax,Nrange);
S21_RangedB = nan(1,Nrange);
Corr_RangedB = nan(1,Nrange);
fig2 = figure;
for i = 1:Nrange
    l.ElementSpacing = range(i);

```

```

S = sparameters(l,freq);
Corr = correlation(l,freq,1,2);
S21_RangedB(i)= 20*log10(abs(S.Parameters(2,1,1)));
Corr_RangedB(i)= 20*log10(Corr);
figure(fig2);
plot(range./lambda,S21_RangedB,range./lambda,Corr_RangedB,'--','LineWidth',2)
grid on
axis([min(range./lambda) max(range./lambda) -50 0]);
xlabel('Distance of separation, d/\lambda')
ylabel('Magnitude (dB)')
title('Correlation, S21 Variation with Range')
drawnow
hold off
end
legend('S21','Correlation');

```



The 2 curves are clearly different in their behavior when the separation distance between two antennas increases. This plot reveals the correlation valleys that exist at specific separations such as approximately 0.75λ , 1.25λ , 1.75λ , and 2.25λ .

Check Correlation Frequency Response

Pick the separation to be 1.25λ , which is one of the correlation valleys. Analyze the correlation variation for 10% bandwidth centered at 800 MHz.

```

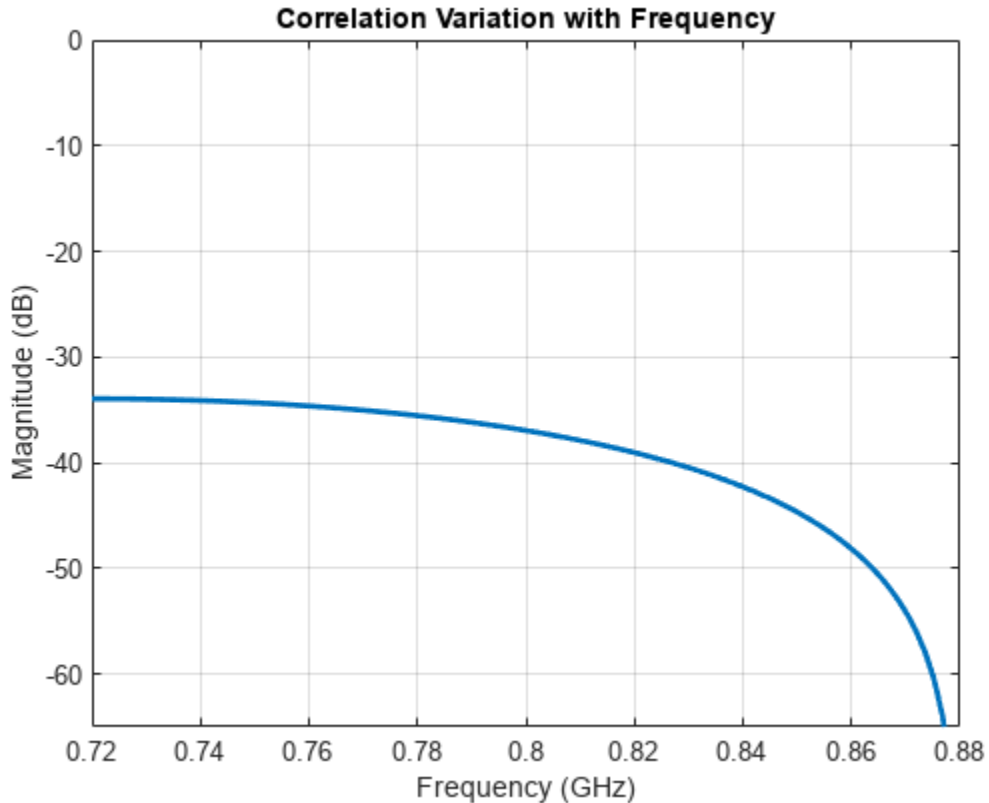
Rpick = 1.25*lambda;
f = linspace(fmin,fmax,Numfreq);

```

```

l.ElementSpacing = Rpick;
Corr_PickdB = 20.*log10(correlation(l,f,1,2));
fig2 = figure;
plot(f./1e9,Corr_TiltdB,'LineWidth',2)
grid on
axis([min(f./1e9) max(f./1e9) -65 0]);
xlabel('Frequency (GHz)')
ylabel('Magnitude (dB)')
title('Correlation Variation with Frequency')

```



The results of the analysis reveals that the two antennas have a correlation below 30 dB over the band specified.

See Also

“Effect of Mutual Coupling on MIMO Communication” on page 5-281

Reference

[1] S. Blanch, J. Romeu, and I. Corbella, "Exact representation of antenna system diversity performance from input parameter description," *Electron. Lett.*, vol. 39, pp. 705-707, May 2003. Online at: <http://upcommons.upc.edu/e-prints/bitstream/2117/10272/4/ExactRepresentationAntenna.pdf>

Polarization Analysis for X-band Microstrip Patch Antenna

In most basic form, a microstrip patch antenna consists of a radiating patch on one side of a dielectric substrate and a ground plane on the other side. Microstrip patch antennas radiate primarily as wide open-open half-wave microstrip resonators. The length L of the rectangular patch for the fundamental excitation mode is slightly less than $\lambda/2$. For a good antenna performance, a thick dielectric substrate having a low dielectric constant is usually desired since it provides a larger bandwidth and a well-defined beam. A rectangular microstrip patch antenna normally radiates a linearly polarized wave with about 6-7 dBi gain at broadside. Higher gains (up to 10 dBi) can be achieved by using different means including large patch heights and parasitic patches.

Patch Antenna

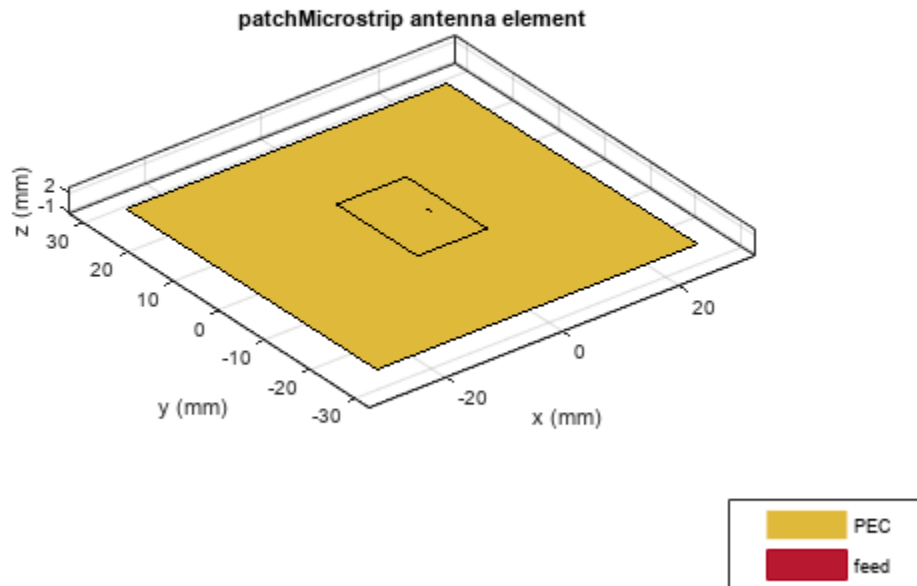
This example discusses a microstrip patch antenna using the air substrate. The dimensions are chosen from [1] for the center frequency of 10.35 GHz.

```
freq          = 10.35e9;  
patchLength  = 12e-3;  
patchWidth   = 17.73e-3;  
patchHeight  = 1.56e-3;  
lengthgp    = 55e-3;  
widthgp     = 55e-3;  
feedoffset   = [2.9e-3 0];
```

Create Antenna

The parameters defined above are used to create the microstrip patch antenna.

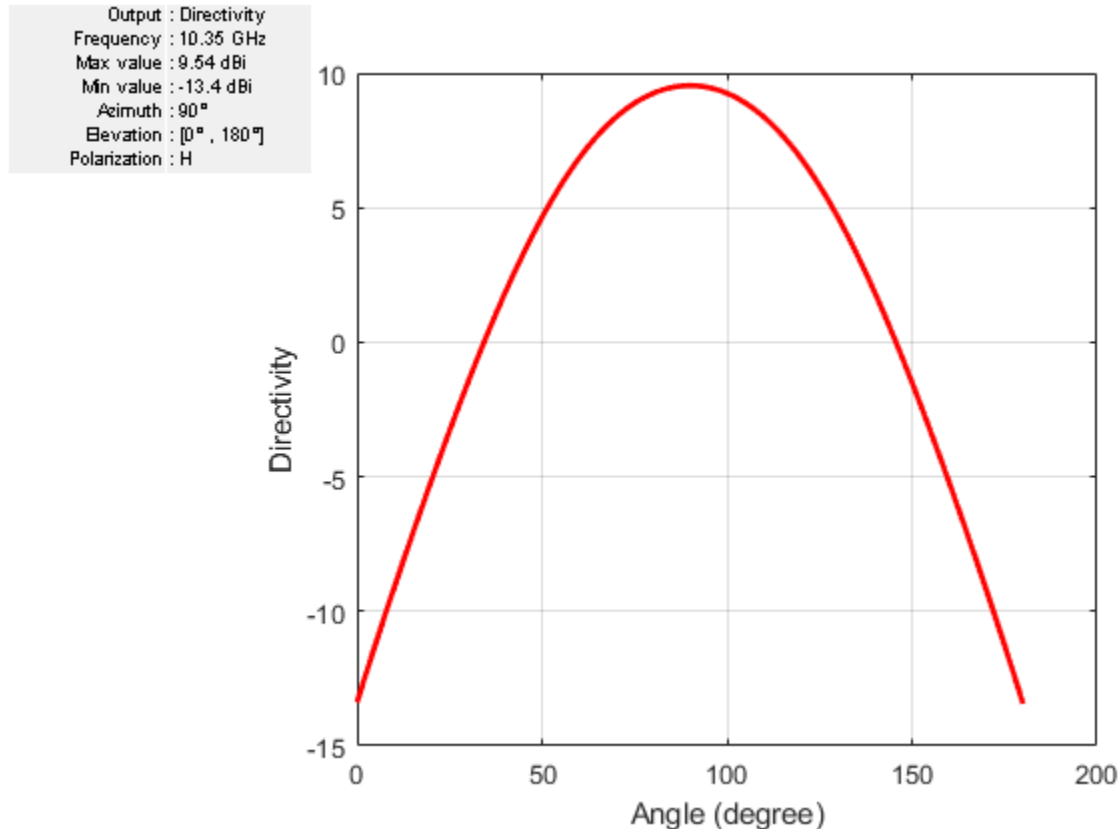
```
ant = patchMicrostrip('Length', patchLength, 'Width', patchWidth,      ...  
    'Height', patchHeight, 'GroundPlaneLength', lengthgp,            ...  
    'GroundPlaneWidth', widthgp, 'FeedOffset', feedoffset);  
figure;  
show(ant);
```



Co-Polarization Pattern in H-plane

The plane, which contains the electric-field vector and the direction of maximum radiation, is known as the E-plane [2] for linearly-polarized antennas. The E-plane is the xz -plane in the present case it is determined by the feed shift in the x -direction, but has generally nothing in common with the relative patch dimensions in the figure. The plane, which contains the magnetic-field vector and the direction of maximum radiation, is known as the H-plane for linearly-polarized antennas. The H-plane in the present case is the yz -plane. Co-polarization is the intended antenna polarization. In the H-plane, the intended polarization is the electric-field component in the x -direction (which coincides with azimuthal component). The plot given below shows the directivity of this azimuthal component in the H-plane (the co-polarization radiation pattern). As expected, the co-polarization pattern drops sharply close to the ground plane, which reflects the boundary condition of no tangential field on the metal ground plane.

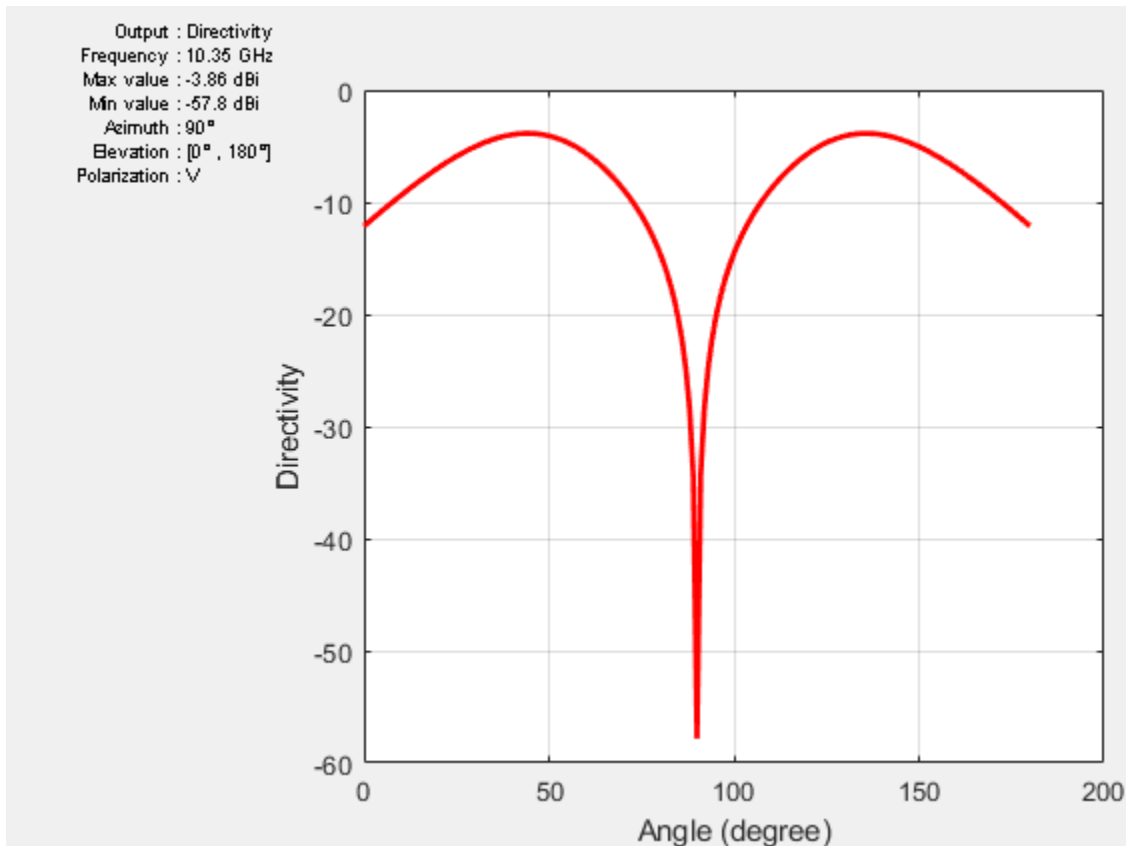
```
pattern(ant, freq, 90, 0:1:180, 'CoordinateSystem', 'rectangular', ...
        'Polarization', 'H');
```



Cross-Polarization Pattern in H-plane

Cross-polarization is the undesired antenna polarization. Cross polarized electric-field component is perpendicular to the co-polarized component. In the H-plane, the cross-polarization is given by the combined electric-field components in the y - and z -directions, which is the elevation electric-field component. The difference between the co- and cross-polarizations is called the polarization isolation. The plot given below shows the directivity of the cross-polarized E-field in the H-plane (cross-polarization radiation pattern). As expected, there is a null at zenith.

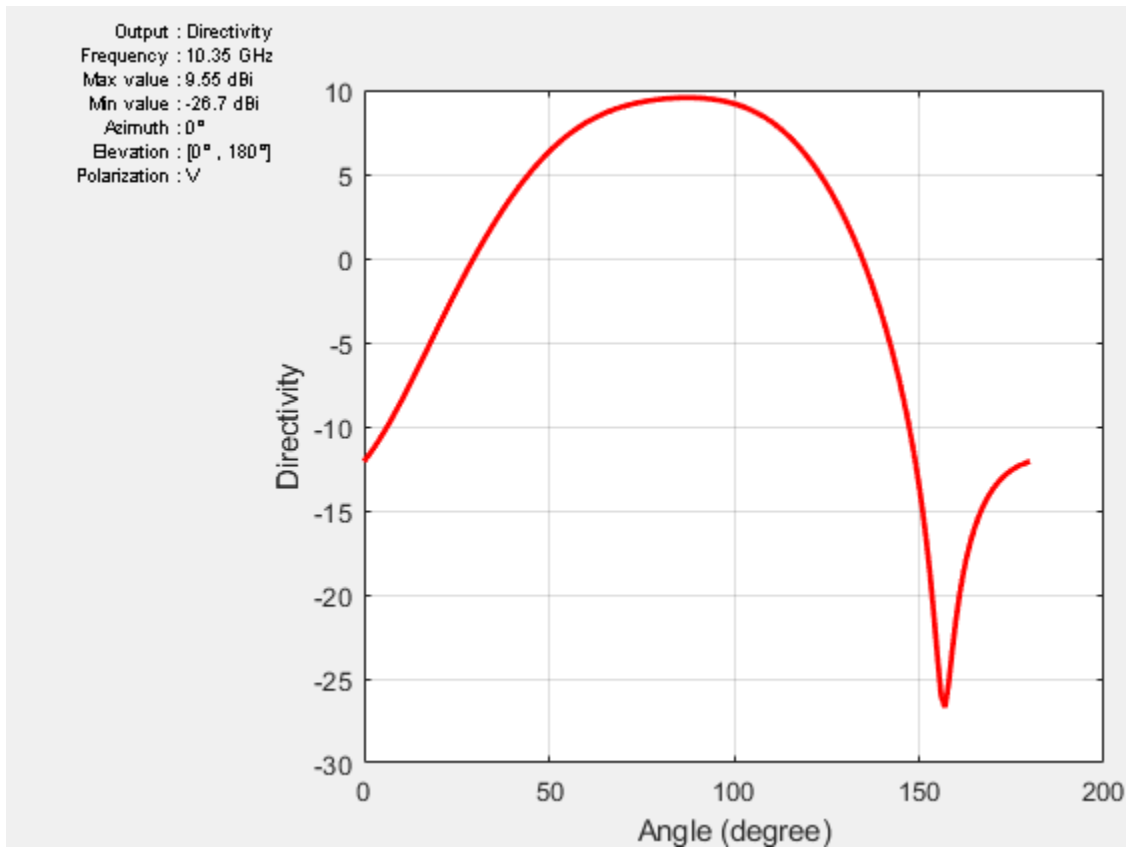
```
pattern(ant, freq, 90, 0:1:180, 'CoordinateSystem', 'rectangular', ...
        'Polarization', 'V');
```



Co-Polarization Pattern in E-plane

In the E-plane, the intended polarization is given by the combined electric-field components in the x - and z -directions, which is the elevation electric-field component. The plot given below shows the directivity of the co-polarized E-field in the E-plane (co-polarization radiation pattern). As expected, the pattern is not symmetric due to the feed effect.

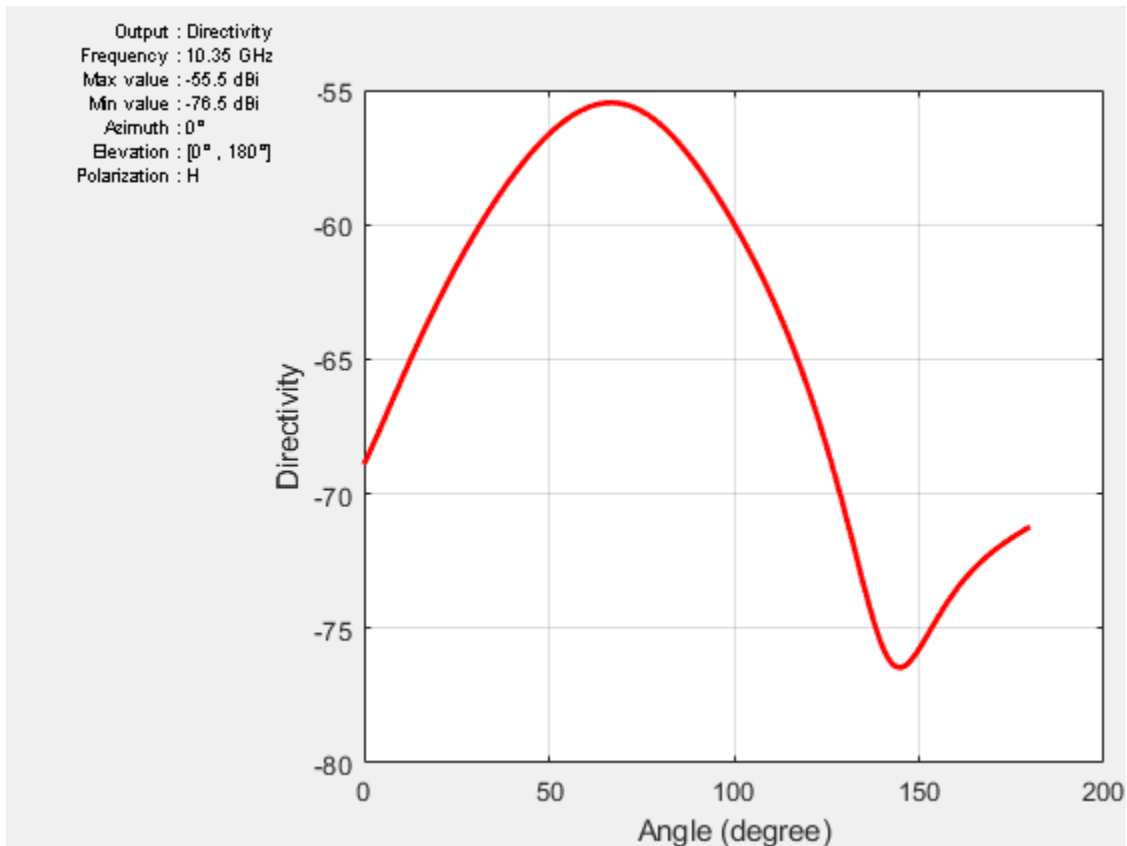
```
pattern(ant, freq, 0, 0:1:180, 'CoordinateSystem', 'rectangular', ...
        'Polarization', 'V');
```



Cross-Polarization Pattern in E-plane

In the E-plane, the cross-polarization is given by the electric-field component in the y -direction, which is the azimuthal electric-field component. The plot given below shows the directivity of the cross-polarized E-field in the E-plane (cross-polarization radiation pattern). As expected, the pattern indicates a very low cross-polarization.

```
pattern(ant, freq, 0, 0:1:180, 'CoordinateSystem', 'rectangular', ...
  'Polarization', 'H');
```

Comparison with Ref. [1]

The overall behavior of the antenna matches well with the results published in [1].

Reference

[1] D. Guha, S. Chattopadhyaya, J. Y. Siddiqui, "Estimation of gain enhancement replacing PTFE by air substrate in a microstrip patch antenna [Antenna Designer's Notebook]," IEEE Antennas and Propagation Magazine, vol.52, no.3, pp.92-95, June 2010.

[2] C. A. Balanis, 'Antenna Theory. Analysis and Design,' p.33, Wiley, New York, 3rd Edition, 2005.

See Also

"Modeling and Analysis of Single Layer Multi-band U-Slot Patch Antenna" on page 5-317

FMCW Patch Antenna Array

This example describes the modeling of a 77 GHz 2 X 4 antenna array for Frequency-Modulated Continuous-Wave (FMCW) applications. The presence of antennas and antenna arrays in and around vehicles has become a commonplace with the introduction of wireless collision detection, collision avoidance, and lane departure warning systems. The two frequency bands considered for such systems are centered around 24 GHz and 77 GHz, respectively. In this example, we will investigate the microstrip patch antenna as a phased array radiator. The dielectric substrate is air.

This example requires the following product:

- Phased Array System Toolbox™

Design Parameters

Set up the center frequency and the frequency band. The velocity of light is assumed to be that of vacuum.

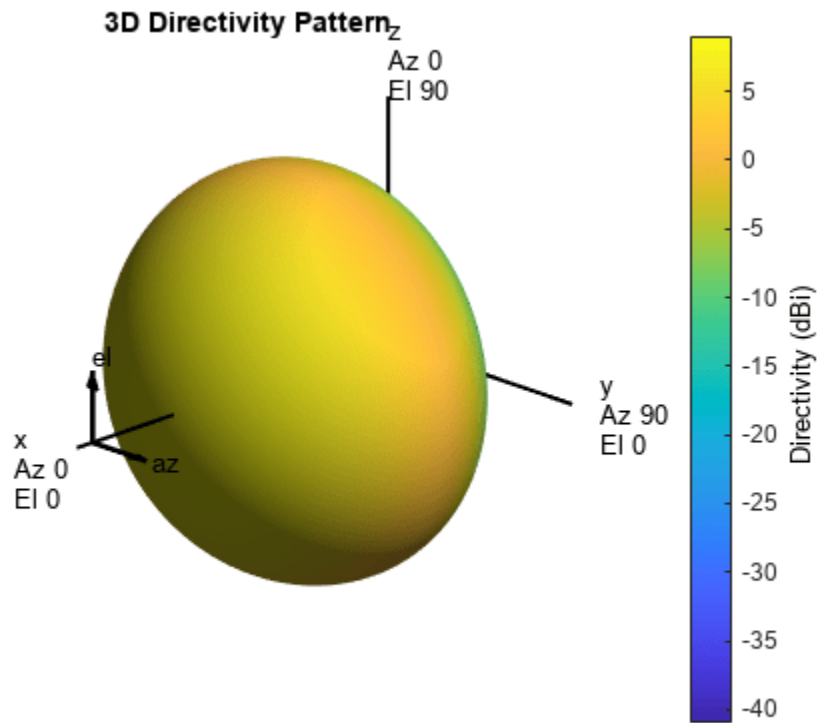
```
fc = 77e9;  
fmin = 73e9;  
fmax = 80e9;  
vp = physconst('lightspeed');  
lambda = vp/fc;
```

Create 2 X 4 Array

Hypothetic Element Pattern

The FMCW antenna array is intended for a forward radar system designed to look for and prevent a collision. Therefore, begin with a hypothetic antenna element that has the significant pattern coverage in one hemisphere. A cosine antenna element would be an appropriate choice.

```
cosineElement = phased.CosineAntennaElement;  
cosineElement.FrequencyRange = [fmin fmax];  
cosinePattern = figure;  
pattern(cosineElement,fc)
```



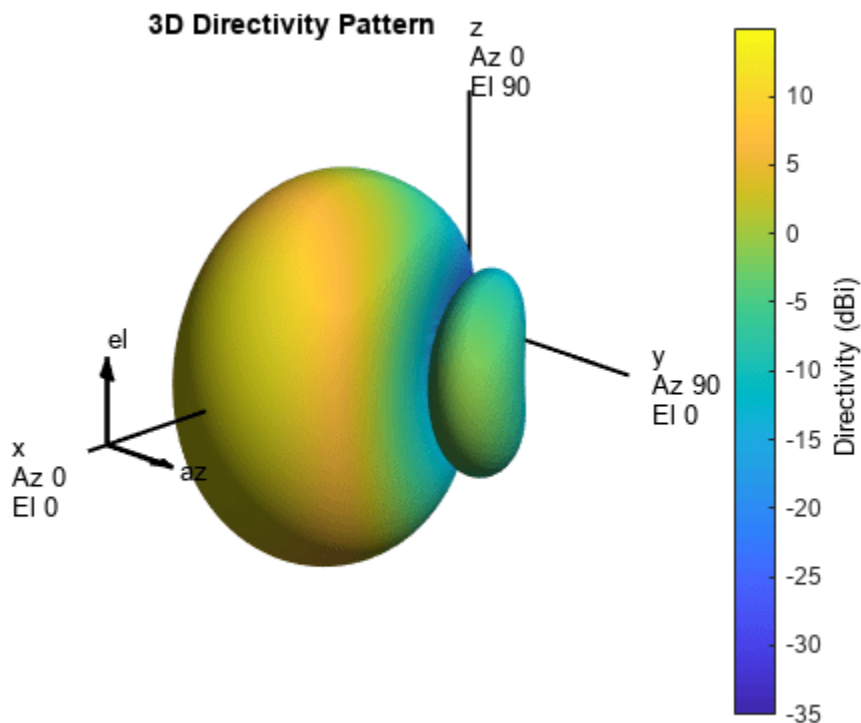
Ideal Array Pattern

The array itself needs to be mounted on or around the front bumper. The array configuration we investigate is similar to that mentioned in [1], i.e. a 2 X 4 rectangular array.

```

Nrow = 2;
Ncol = 4;
fmcwCosineArray = phased.URA;
fmcwCosineArray.Element = cosineElement;
fmcwCosineArray.Size = [Nrow Ncol];
fmcwCosineArray.ElementSpacing = [0.5*lambda 0.5*lambda];
cosineArrayPattern = figure;
pattern(fmcwCosineArray,fc);

```



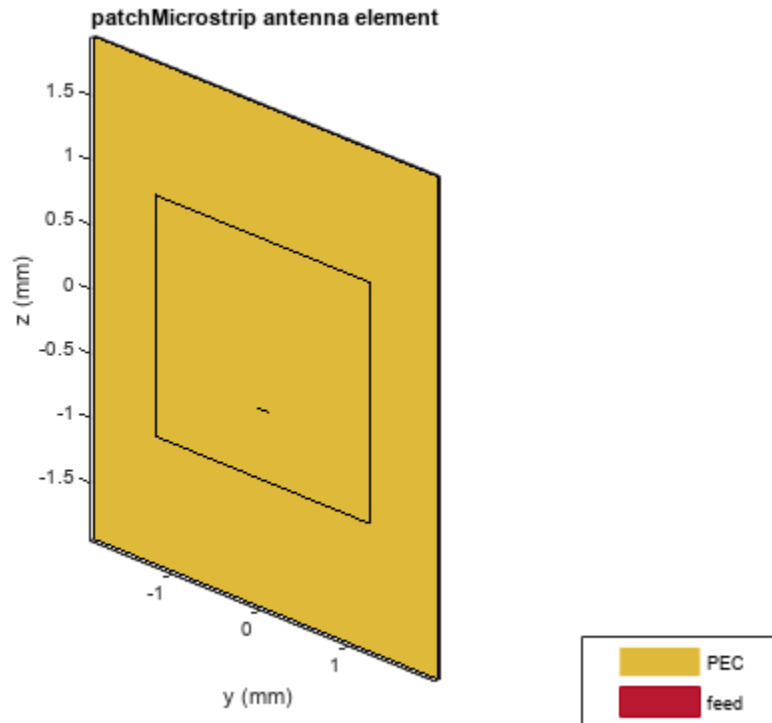
Design Realistic Patch Antenna

The Antenna Toolbox™ has several antenna elements that could provide hemispherical coverage. Choose the patch antenna element and design it at the frequency of interest. The patch length is approximately half-wavelength at 77 GHz and the width is 1.5 times the length, for improving the bandwidth.

```
patchElement = design(patchMicrostrip, fc);
```

Since the default patch antenna geometry in the Antenna Toolbox library has its maximum radiation directed towards zenith, rotate the patch antenna by 90 degrees about the y-axis so that the maximum would now occur along the x-axis. This is also the boresight direction for arrays in Phased Array System Toolbox.

```
patchElement.Tilt = 90;
patchElement.TiltAxis = [0 1 0];
figure
show(patchElement)
axis tight
view(140,20)
```

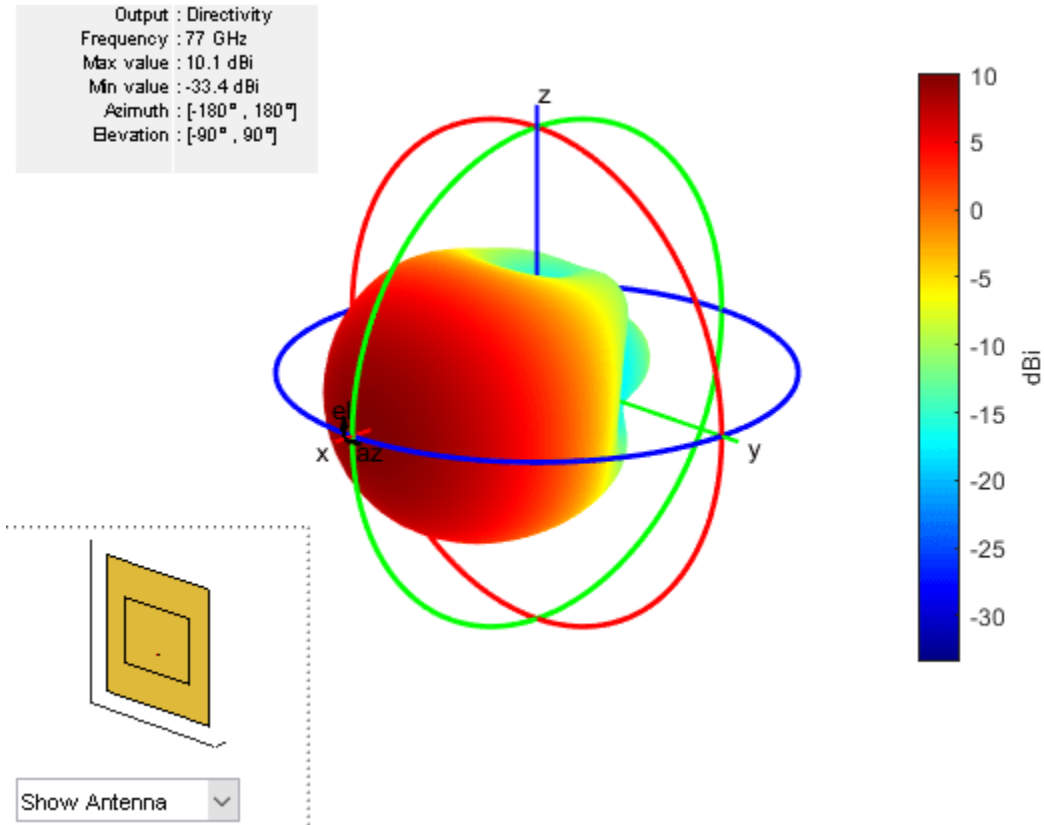


Isolated Patch Antenna 3D Pattern and Resonance

3D Directivity Pattern

Plot the pattern of the patch antenna at 77 GHz. The patch is a medium gain antenna with the peak directivity around 6 - 9 dBi.

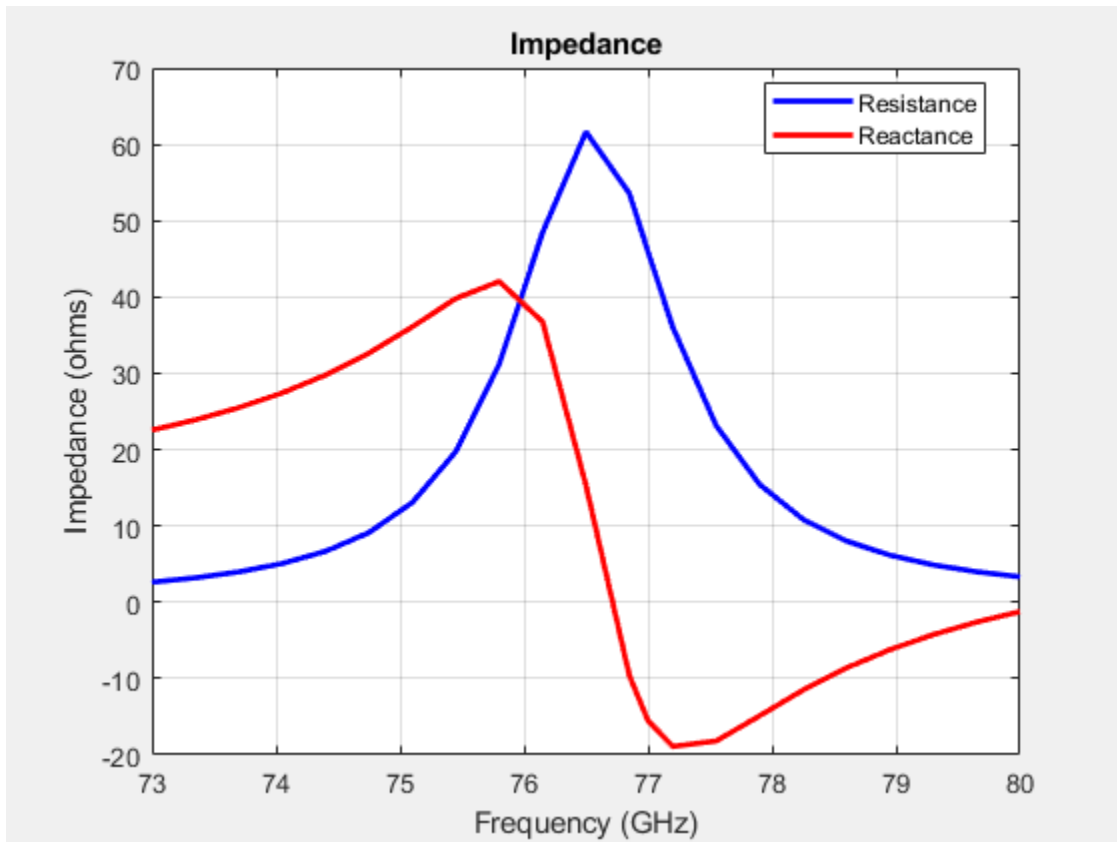
```
pattern(patchElement, fc)
```



Resonance

The patch is radiating in the correct mode with a pattern maximum at azimuth = elevation = 0 degrees. Since the initial dimensions are an approximation, check the input impedance behavior.

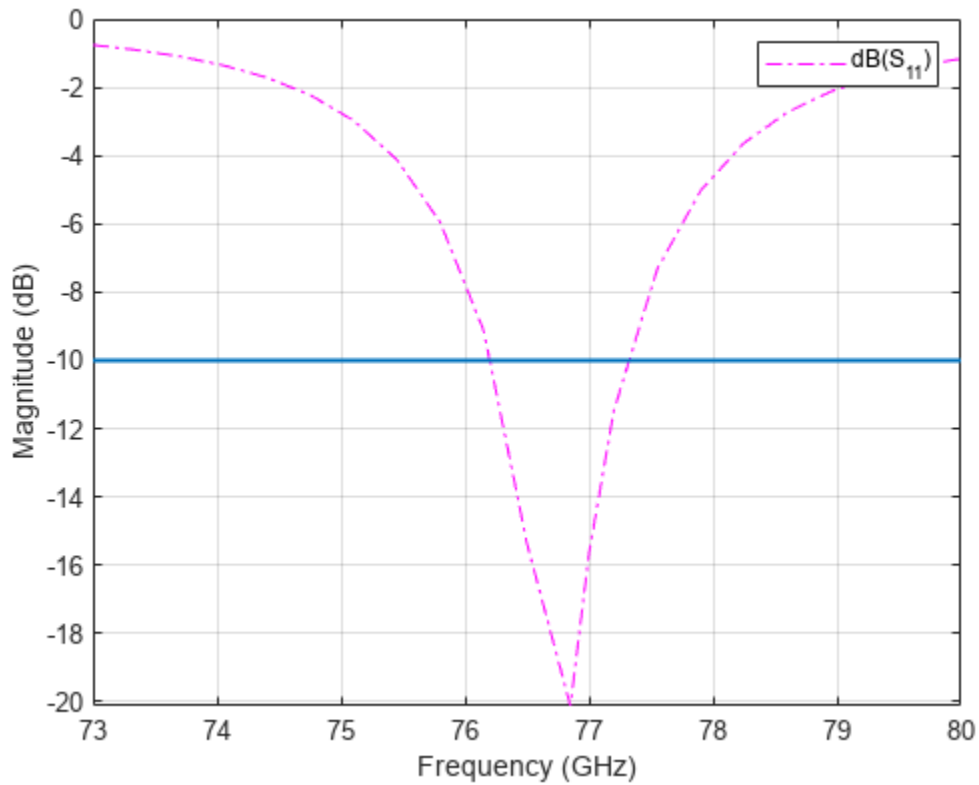
```
Numfreqs = 21;
freqsweep = unique([linspace(fmin,fmax,Numfreqs) fc]);
impedance(patchElement,freqsweep);
```



Establish Bandwidth

Plot the reflection coefficient of the patch to confirm a good impedance match. It is typical to consider the value $S_{11} = -10\text{dB}$ as a threshold value for determining the antenna bandwidth.

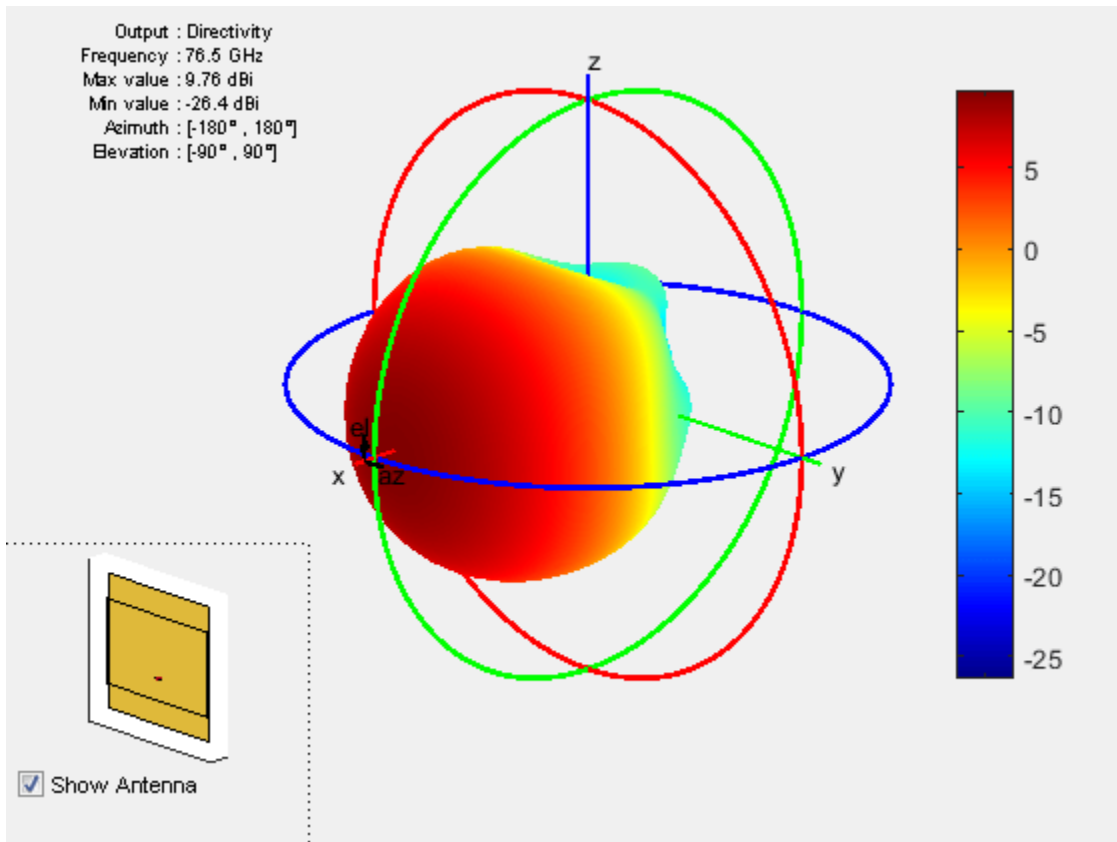
```
s = sparameters(patchElement, freqsweep);
figure
rfplot(s, 'm-.')
hold on
line(freqsweep/1e09, ones(1, numel(freqsweep))*-10, 'LineWidth', 1.5)
hold off
```

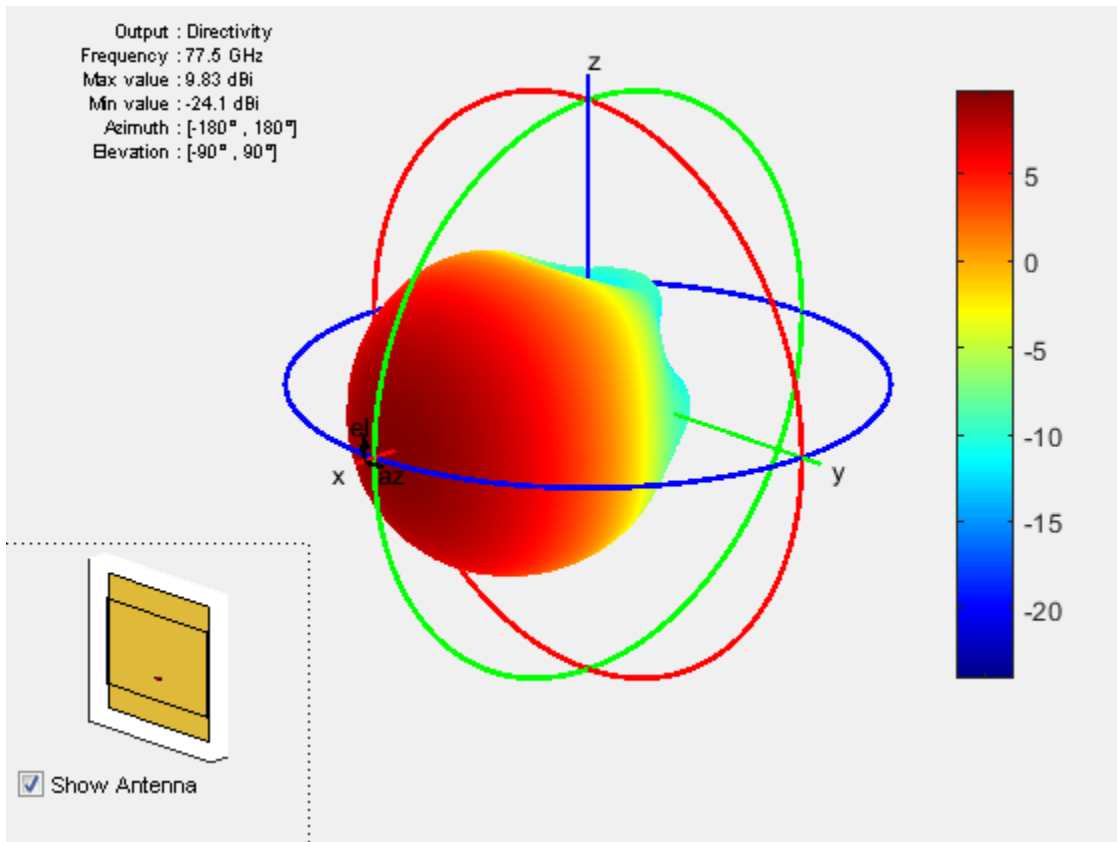


The deep minimum at 77 GHz indicates a good match to 50. The antenna bandwidth is slightly greater than 1 GHz. Thus, the frequency band is from 76.5 GHz to 77.5 GHz.

Confirm Pattern at Center and Corner Frequencies

Confirm that the pattern at the corner frequencies of the band remains nearly the same. The pattern plots at 76.5 GHz and 77.6 GHz are shown below.





It is a good practice to check pattern behavior over the frequency band of interest in general.

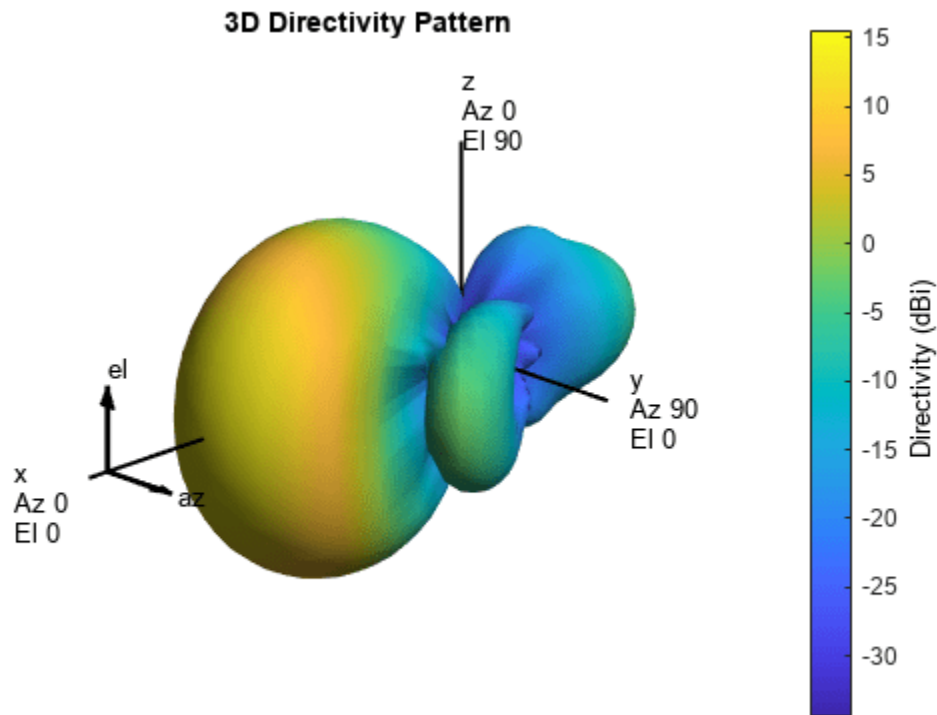
Create Array from Isolated Radiators and Plot Pattern

Create the uniform rectangular array (URA), but this time use the isolated patch antenna as the individual element. We choose spacing $\lambda/2$ at the upper frequency of the band i.e. 77.6 GHz.

```
fc2 = 77.6e9;
lambda_fc2 = vp/77.6e9;
fmcwPatchArray = phased.URA;
fmcwPatchArray.Element = patchElement;
fmcwPatchArray.Size = [Nrow Ncol];
fmcwPatchArray.ElementSpacing = [0.5*lambda_fc2 0.5*lambda_fc2];
```

Plot the pattern for the patch antenna array so constructed. Specify a 5 degree separation in azimuth and elevation to plot the 3D pattern.

```
az = -180:5:180;
el = -90:5:90;
patchArrayPattern = figure;
pattern(fmcwPatchArray,fc,az,el);
```

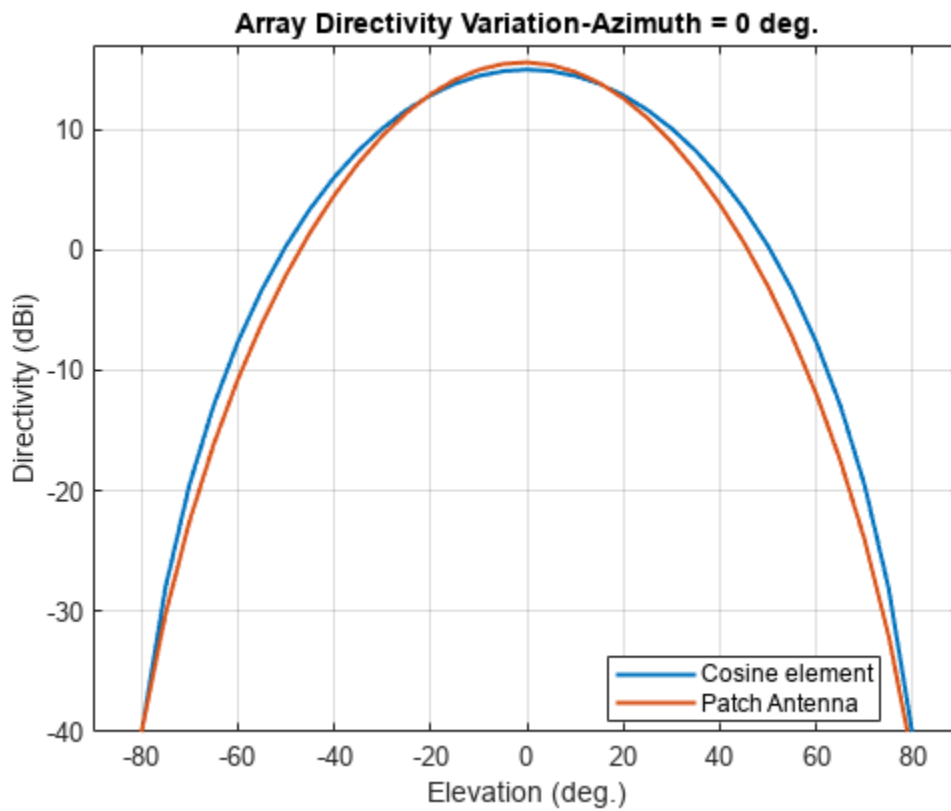


Plot Pattern Variation in Two Orthogonal Planes

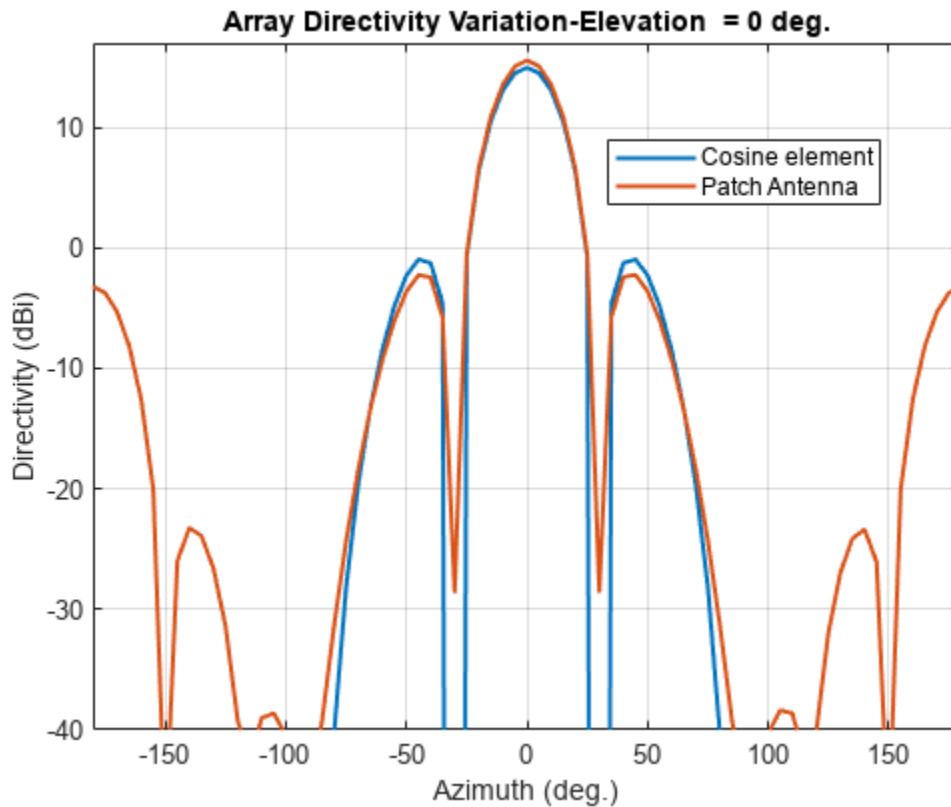
Compare the pattern variation in 2 orthogonal planes for the patch antenna array and the cosine element array. Both arrays ignore mutual coupling.

```
[Dcosine_az_zero,~,eln] = pattern(fmcwCosineArray,fc,0,el);
[Dcosine_el_zero,azn] = pattern(fmcwCosineArray,fc,az,0);
[Dpatch_az_zero,~,elp] = pattern(fmcwPatchArray,fc,0,el);
[Dpatch_el_zero,azp] = pattern(fmcwPatchArray,fc,az,0);

elPattern = figure;
plot(eln,Dcosine_az_zero,eln,Dpatch_az_zero,'LineWidth',1.5)
axis([min(eln) max(eln) -40 17])
grid on
xlabel('Elevation (deg.)')
ylabel('Directivity (dBi)')
title('Array Directivity Variation-Azimuth = 0 deg.')
legend('Cosine element','Patch Antenna','Location','best')
```



```
azPattern = figure;  
plot(azn,Dcosine_el_zero,azn,Dpatch_el_zero,'LineWidth',1.5)  
axis([min(azn) max(azn) -40 17])  
grid on  
xlabel('Azimuth (deg.)')  
ylabel('Directivity (dBi)')  
title('Array Directivity Variation-Elevation = 0 deg.')  
legend('Cosine element','Patch Antenna','Location','best')
```



Discussion

The cosine element array and the array constructed from isolated patch antennas, both without mutual coupling, have similar pattern behavior around the main beam in the elevation plane (azimuth = 0 deg). The patch-element array has a significant back lobe as compared to the cosine-element array. Using the isolated patch element is a useful first step in understanding the effect that a realistic antenna element would have on the array pattern. However, in the realistic array analysis, mutual coupling must be considered. Since this is a small array (8 elements in 2 X 4 configuration), the individual element patterns in the array environment could be distorted significantly. As a result, it is not possible to replace the isolated element pattern with an embedded element pattern. A full-wave analysis must be performed to understand the effect of mutual coupling on the overall array performance.

See Also

“Patch Antenna on Dielectric Substrate” on page 5-95

“Effect of Mutual Coupling on MIMO Communication” on page 5-281

Reference

[1] Kulke, R., S. Holzwarth, J. Kassner, A. Lauer, M. Rittweger, P. Uhlig and P. Weigand. “24 GHz Radar Sensor integrates Patch Antenna and Frontend Module in single Multilayer LTCC Substrate.” (2005).

Modeling Mutual Coupling in Large Arrays Using Infinite Array Analysis

This example uses infinite array analysis to model large finite arrays. The infinite array analysis on the unit cell reveals the scan impedance behavior at a particular frequency. This information is used with the knowledge of the isolated element pattern and impedance to calculate the scan element pattern. The large finite array is then modeled using the assumption that every element in the array possesses the same scan element pattern. The real array of finite size can be analyzed using the embedded element pattern as a next step. This approach is presented in “Modeling Mutual Coupling in Large Arrays Using Embedded Element Pattern” on page 5-195.

This example requires the following product:

- Phased Array System Toolbox™

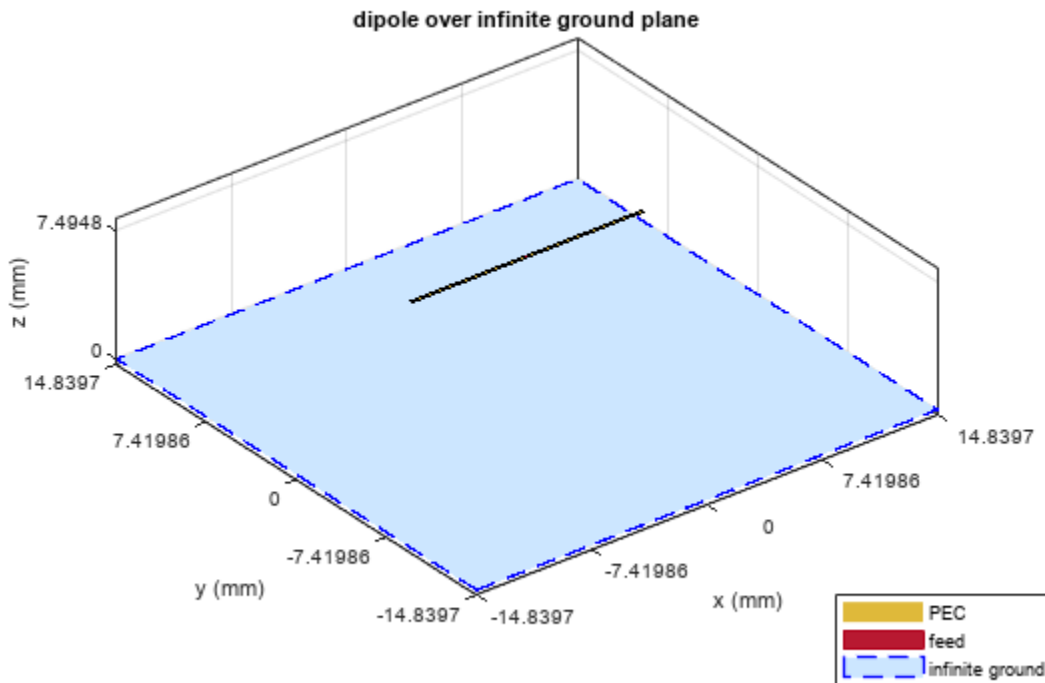
Define Individual Element

For this example, the center of the X-band is chosen as the design frequency.

```
freq = 10e9;  
vp = physconst('lightspeed');  
lambda = vp/freq;  
ucdx = 0.5*lambda;  
ucdy = 0.5*lambda;
```

Create a thin dipole of length slightly less than $\lambda(\text{lambda})/2$ and assign it as the exciter to a infinitely large reflector.

```
d = dipole;  
d.Length = 0.495*lambda;  
d.Width = lambda/160;  
d.Tilt = 90;  
d.TiltAxis = [0 1 0];  
  
r = reflector;  
r.Exciter = d;  
r.Spacing = lambda/4;  
r.GroundPlaneLength = inf;  
r.GroundPlaneWidth = inf;  
figure;  
show(r);
```



Calculate the isolated element pattern and the impedance of the above antenna. These results are used to calculate the Scan Element Pattern(SEP), also known as Array Element Pattern(AEP) or Embedded Element Pattern(EEP).

```
%Define az and el vectors
az = 0:2:360;
el = 90:-2:-90;

% Calculate power pattern
giso = pattern(r,freq,az,el,'Type','power'); % el x az

% Calculated impedance
Ziso = impedance(r,freq);
```

Calculate Infinite Array Scan Element Pattern

Unit cell

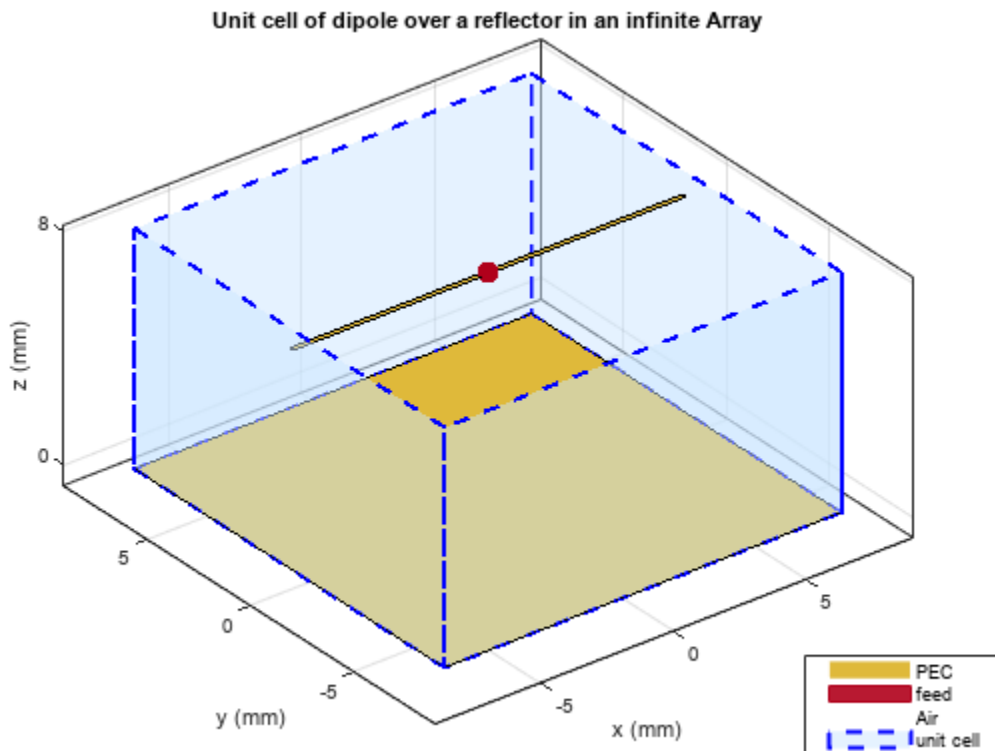
The term *unit cell* in the infinite array analysis refers to a single element in an infinite array. A dipole backed by a reflector and a microstrip patch antenna are representative examples of a unit cell in infinite arrays. This example uses a dipole backed by a reflector and analyzes the impedance behavior at 10 GHz as a function of the scan angle. The unit cell has a cross-section of $\lambda/2$ -by- $\lambda/2$.

```
r.GroundPlaneLength = ucdx;
r.GroundPlaneWidth = ucdy;
infArray = infiniteArray;
infArray.Element = r;
```

```

infArray.ScanAzimuth = 30;
infArray.ScanElevation = 45;
figure;
show(infArray);

```



Scan impedance

The scan impedance at a single frequency and single scan angle is shown.

```
scanZ = impedance(infArray, freq)
```

```
scanZ = 1.0860e+02 + 3.0442e+01i
```

For this example, the scan impedance for the full volume of scan is calculated using 50 terms in the double summation for the periodic Greens function to improve convergence behavior. For more information, refer to the following example: "Infinite Array Analysis" on page 5-58.

Scan Element Pattern /Array Element Pattern /Embedded Element Pattern

The scan element pattern (SEP) is calculated from the infinite array scan impedance, the isolated element pattern and the isolated element impedance. The expression used is shown here[1],[2]:

$$g_s(\theta) = \frac{4R_g R_i(g_i(\theta))}{|Z_s(\theta) + Z_g|^2}$$


```

load InfArrayScanZData
scanZ = scanZ.';
Rg = 185;
Xg = 0;
Zg = Rg + 1i*Xg;
gs = nan(numel(el),numel(az));
for i = 1:numel(el)
    for j = 1:numel(az)
        gs(i,j) = 4*Rg*real(Ziso).*giso(i,j)./(abs(scanZ(i,j) + Zg)).^2;
    end
end

```

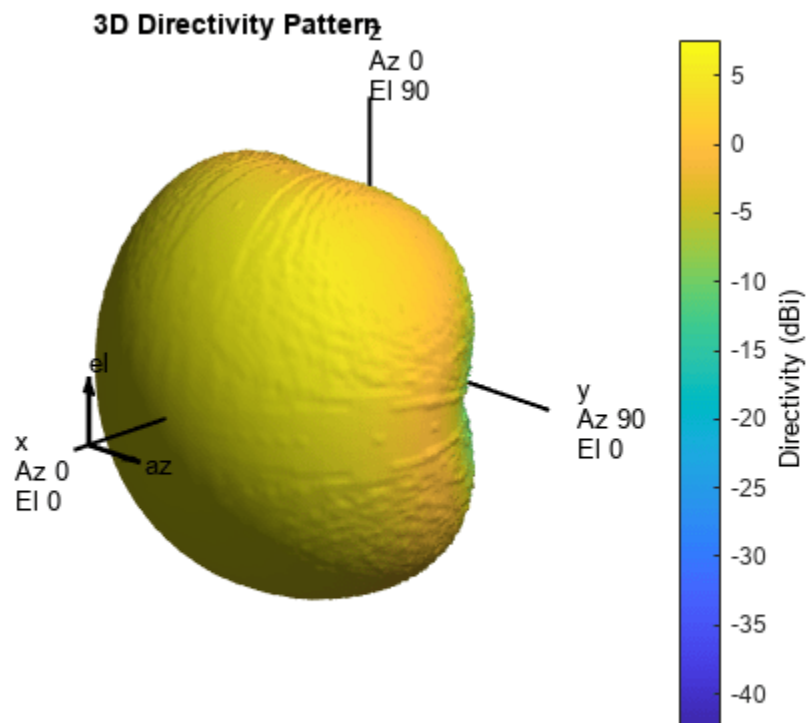
Build Custom Antenna Element

The scan element pattern obtained from full-wave analysis is now used in a system level analysis provided by the Phased Array System Toolbox™. For that, build a custom antenna element as a first step.

```

fieldpattern = sqrt(gs);
bandwidth = 500e6;
customAntInf = buildCustomAntenna(fieldpattern,freq,bandwidth,az,el);
figure;
pattern(customAntInf,freq);

```



Build 21-by-21 URA

Create a uniform rectangular array (URA) with the custom antenna element, which contains the scan element pattern. The assumption implicit in this step is that every element of this array has the same element pattern.

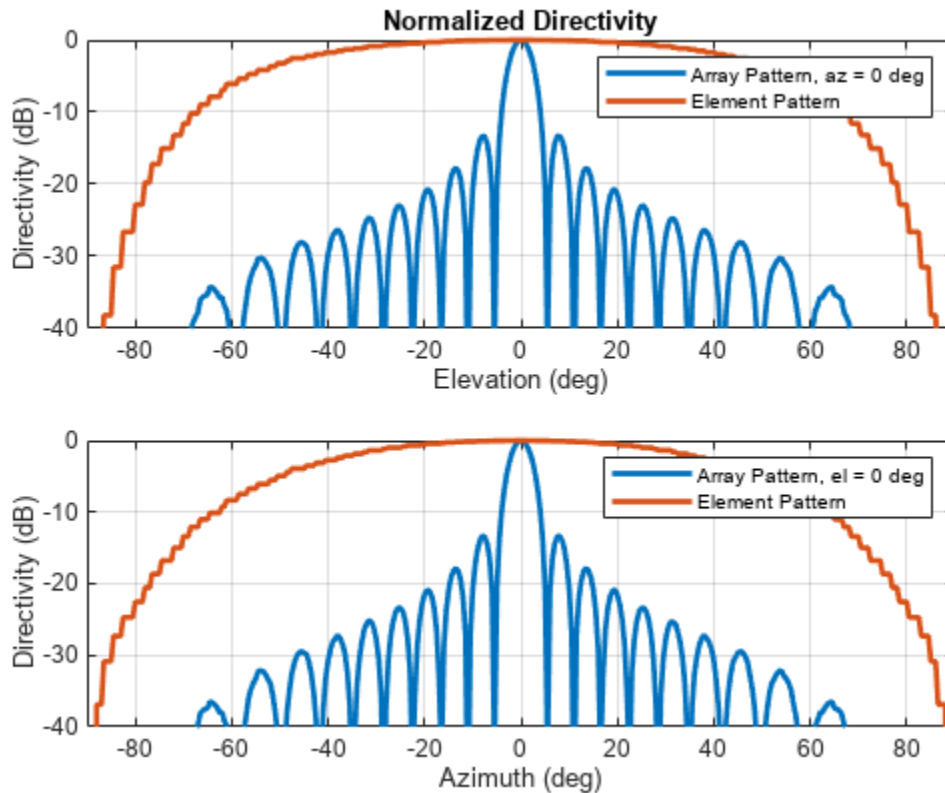
```
N = 441;
Nrow = sqrt(N);
Ncol = sqrt(N);
drow = ucdx;
dcol = ucdy;
myURA1 = phased.URA;
myURA1.Element = customAntInf;
myURA1.Size = [Nrow Ncol];
myURA1.ElementSpacing = [drow dcol];
```

Plot Slices in E and H planes

Calculate the pattern in the elevation plane (specified by azimuth = 0 deg and also called the E-plane) and azimuth plane (specified by elevation = 0 deg and called the H-plane) for the array built using infinite array analysis.

```
azang_plot = -90:0.5:90;
elang_plot = -90:0.5:90;
% E-plane
Darray1_E = pattern(myURA1,freq,0,elang_plot);
Darray1_Enormlz = Darray1_E - max(Darray1_E);
% H-plane
Darray1_H = pattern(myURA1,freq,azang_plot,0);
Darray1_Hnormlz = Darray1_H - max(Darray1_H);
% Scan element pattern in both planes
DSEP1_E = pattern(customAntInf,freq,0,elang_plot);
DSEP1_Enormlz = DSEP1_E - max(DSEP1_E);
DSEP1_H = pattern(customAntInf,freq,azang_plot,0);
DSEP1_Hnormlz = DSEP1_H - max(DSEP1_H);

figure
subplot(211)
plot(elang_plot,Darray1_Enormlz,elang_plot,DSEP1_Enormlz,'LineWidth',2)
grid on
axis([min(azang_plot) max(azang_plot) -40 0]);
legend('Array Pattern, az = 0 deg','Element Pattern')
xlabel('Elevation (deg)')
ylabel('Directivity (dB)')
title('Normalized Directivity')
subplot(212)
plot(azang_plot,Darray1_Hnormlz,azang_plot,DSEP1_Hnormlz,'LineWidth',2)
grid on
axis([min(azang_plot) max(azang_plot) -40 0]);
legend('Array Pattern, el = 0 deg','Element Pattern')
xlabel('Azimuth (deg)')
ylabel('Directivity (dB)')
```



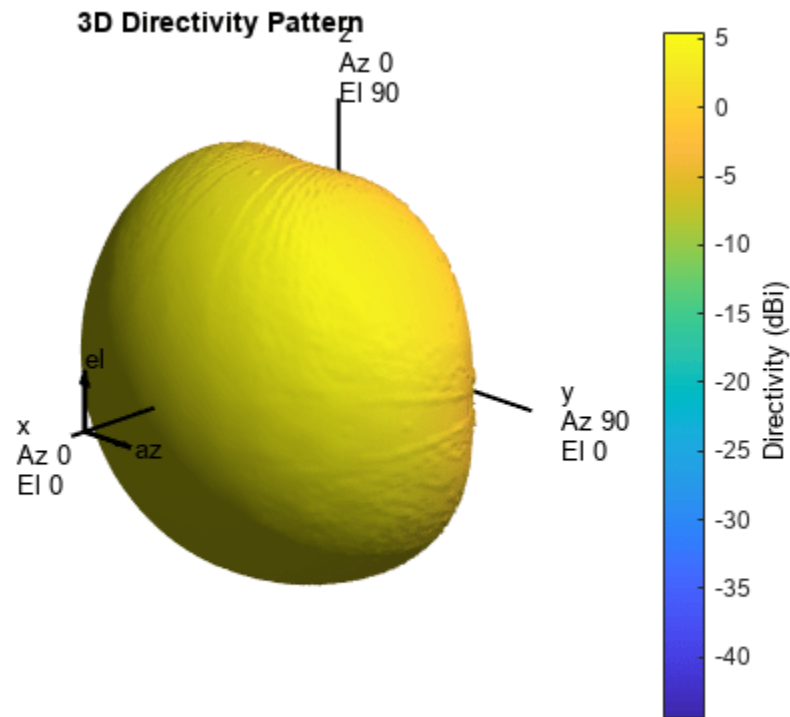
Comparison with Full Wave Finite Array Analysis

To understand the effect of the finite size of the array, we execute a full wave analysis of a 21 X 21 dipole array backed by an infinite reflector. The full wave array pattern slices in the E and H planes as well as the center element embedded element pattern is also calculated. This data is loaded from a MAT file. This analysis took approximately 630 seconds on a 2.4 GHz machine with 32 GB memory.

Load full wave data and build custom antenna

Load the finite array analysis data, and use the embedded element pattern to build a custom antenna element. Note that the pattern from the full-wave analysis needs to be rotated by 90 degrees so that it lines up with the URA model built on the YZ plane.

```
load dipolerefarray
elemfieldpatternfinite = sqrt(FiniteArrayPatData.ElemPat);
arraypatternfinite = FiniteArrayPatData.ArrayPat;
bandwidth = 500e6;
customAntFinite = buildCustomAntenna(elemfieldpatternfinite,freq,bandwidth,az,el);
figure
pattern(customAntFinite,freq)
```



Create Uniform Rectangular Array with embedded element pattern

As done before, create a uniform rectangular array with the custom antenna element.

```
myURA2 = phased.URA;
myURA2.Element = customAntFinite;
myURA2.Size = [Nrow Ncol];
myURA2.ElementSpacing = [drow dcol];
```

E and H plane slice - array With embedded element pattern

Calculate the pattern slices in two orthogonal planes - E and H for the array with the embedded element pattern and the embedded element pattern itself. In addition, since the full wave data for the array pattern is also available use this to compare results.

```
Darray2_E = pattern(myURA2,freq,0,elang_plot);
Darray2_Enormlz = Darray2_E - max(Darray2_E);
% H-plane
Darray2_H = pattern(myURA2,freq,azang_plot,0);
Darray2_Hnormlz = Darray2_H - max(Darray2_H);
```

E and H plane slice - embedded element pattern from finite array

```
DSEP2_E = pattern(customAntFinite,freq,0,elang_plot);
DSEP2_Enormlz = DSEP2_E - max(DSEP2_E);
DSEP2_H = pattern(customAntFinite,freq,azang_plot,0);
DSEP2_Hnormlz = DSEP2_H - max(DSEP2_H);
```

E and H plane slice - full wave analysis of finite array

```
azang_plot1 = -90:2:90;
elang_plot1 = -90:2:90;
```

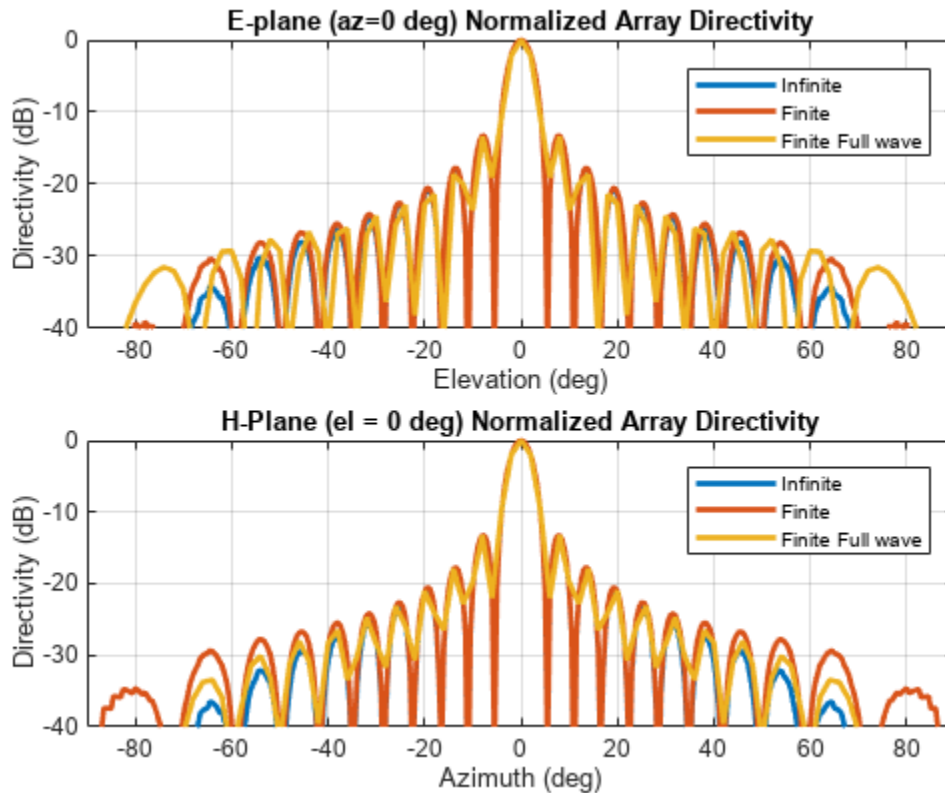
```
Darray3_E = FiniteArrayPatData.EPlane;
Darray3_Enormlz = Darray3_E - max(Darray3_E);
```

```
Darray3_H = FiniteArrayPatData.HPlane;
Darray3_Hnormlz = Darray3_H - max(Darray3_H);
```

Comparison of array patterns

The array patterns in the two orthogonal planes are plotted here.

```
figure
subplot(211)
plot(elang_plot,Darray1_Enormlz,elang_plot,Darray2_Enormlz,elang_plot1,Darray3_Enormlz,'LineWidth',2)
grid on
axis([min(elang_plot) max(elang_plot) -40 0]);
legend('Infinite','Finite','Finite Full wave','location','best')
xlabel('Elevation (deg)')
ylabel('Directivity (dB)')
title('E-plane (az=0 deg) Normalized Array Directivity')
subplot(212)
plot(azang_plot,Darray1_Hnormlz,azang_plot,Darray2_Hnormlz,azang_plot1,Darray3_Hnormlz,'LineWidth',2)
grid on
axis([min(azang_plot) max(azang_plot) -40 0]);
legend('Infinite','Finite','Finite Full wave','location','best')
xlabel('Azimuth (deg)')
ylabel('Directivity (dB)')
title('H-Plane (el = 0 deg) Normalized Array Directivity')
```

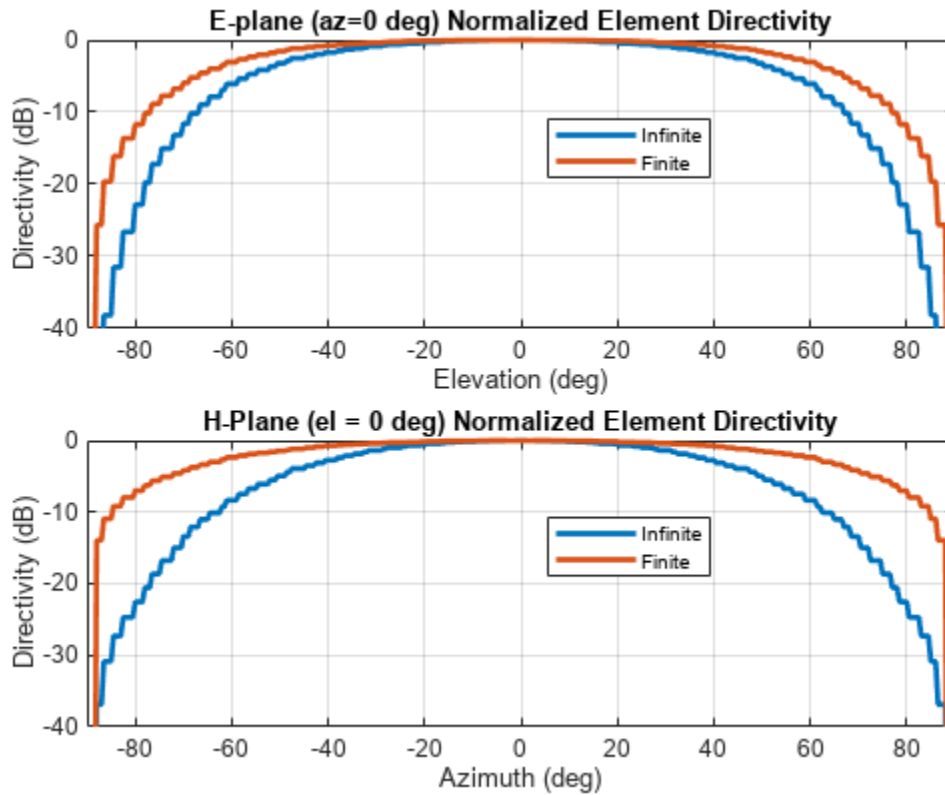


The pattern plots in the two planes reveals that all three analysis approaches suggest similar behavior out to ± 40 degree from boresight. Beyond this range, it appears that using the scan element pattern for all elements in a URA (i.e. pattern multiplication principle) underestimates the sidelobe level compared to the full wave analysis of a finite array. The one possible reason for this could be the edge effect from the finite size array.

Comparison of element patterns

The element patterns from the infinite array analysis and the finite array analysis are compared here.

```
figure
subplot(211)
plot(elang_plot,DSEP1_Enormlz,elang_plot,DSEP2_Enormlz,'LineWidth',2)
grid on
axis([min(azang_plot) max(azang_plot) -40 0]);
legend('Infinite','Finite','location','best')
xlabel('Elevation (deg)')
ylabel('Directivity (dB)')
title('E-plane (az=0 deg) Normalized Element Directivity')
subplot(212)
plot(azang_plot,DSEP1_Hnormlz,azang_plot,DSEP2_Hnormlz,'LineWidth',2)
grid on
axis([min(azang_plot) max(azang_plot) -40 0]);
legend('Infinite','Finite','location','best')
xlabel('Azimuth (deg)')
ylabel('Directivity (dB)')
title('H-Plane (el = 0 deg) Normalized Element Directivity')
```

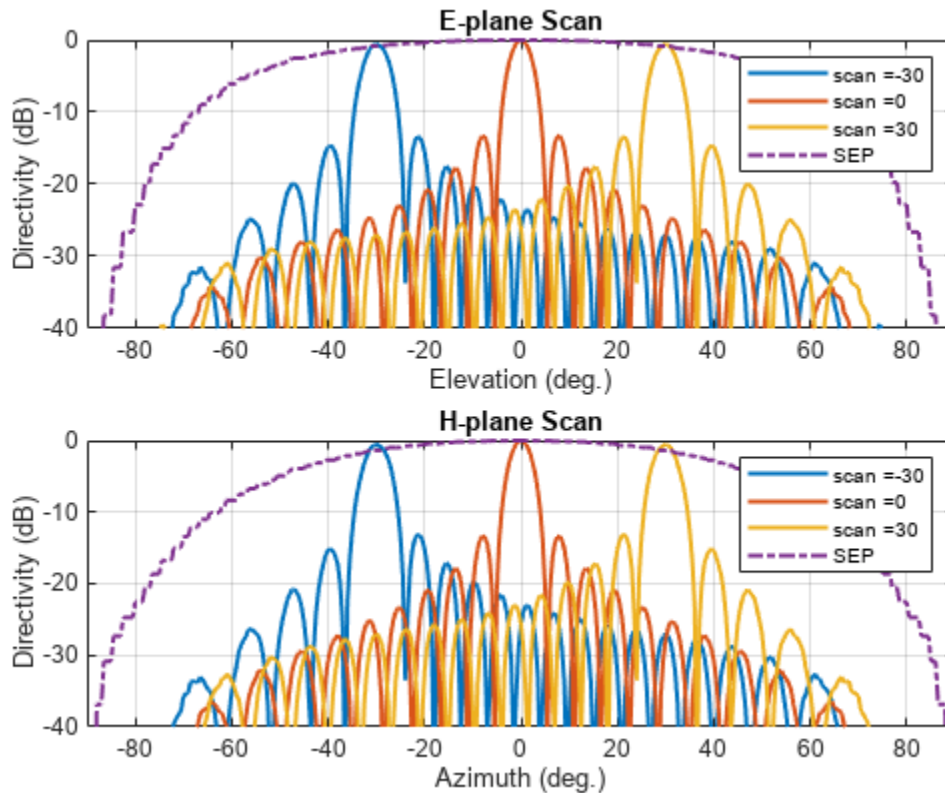


The pattern plots suggests that the behavior out to ± 40 degree from boresight is approximately the same. Outside of this range there is greater drop off in the H-plane pattern from the infinite array analysis as compared to the finite array.

Scan Behavior with Infinite Array Scan Element Pattern

Scan the array based on the infinite array scan element pattern in the elevation plane defined by azimuth = 0 deg and plot the normalized directivity. Also, overlay the normalized scan element pattern.

```
scanURA(myURA1,freq,azang_plot,elang_plot,DSEP1_Enormlz,DSEP1_Hnormlz);
```



Note the overall shape of the normalized array pattern approximately follows the normalized scan element pattern. This is also predicted by the pattern multiplication principle.

Conclusion

Infinite array analysis is one of the tools deployed to analyze and design large finite arrays. The analysis assumes that all elements are identical, have uniform excitation amplitude and that edge effects can be ignored. The isolated element pattern is replaced with the scan element pattern which includes the effect of mutual coupling.

Reference

- [1] J. Allen, "Gain and impedance variation in scanned dipole arrays," IRE Transactions on Antennas and Propagation, vol.10, no.5, pp.566-572, September 1962.
- [2] R. C. Hansen, Phased Array Antennas, Chapter 7 and 8, John Wiley & Sons Inc., 2nd Edition, 1998.

See Also

"Modeling Mutual Coupling in Large Arrays Using Embedded Element Pattern" on page 5-195 |
 "Modeling Infinite Ground Plane in Antennas and Arrays" on page 5-48

Modeling Mutual Coupling in Large Arrays Using Embedded Element Pattern

This example demonstrates the embedded element pattern approach to model large finite arrays. Such an approach is only good for very large arrays so that the edge effects may be ignored. It is common to consider an infinite array analysis as a first step for such kind of analysis. This approach is presented in “Modeling Mutual Coupling in Large Arrays Using Infinite Array Analysis” on page 5-184; Modeling Mutual Coupling in Large Arrays Using Infinite Array Analysis>. The embedded element pattern refers to the pattern of a single element embedded in the finite array, that is calculated by driving the central element in the array and terminating all other elements into a reference impedance [1]-[3]. The pattern of the driven element, referred to as the embedded element, incorporates the effect of coupling with the neighboring elements. It is common to choose the central region/element of the array for the embedded element, depending on whether the array has an even or odd number of elements (for large arrays it does not matter). The pattern of the isolated element (the radiator located in space by itself) changes when it is placed in an array due to the presence of mutual coupling. This invalidates the use of pattern multiplication, which assumes that all elements have the same pattern. To use pattern multiplication to calculate the total array radiation pattern, and improve the fidelity of the analysis, we replace the isolated element pattern with the embedded element pattern.

This example requires the following product:

- Phased Array System Toolbox

Analysis Approach

As mentioned in the introduction, it is the aim of this example to illustrate the use of the embedded element pattern when modeling large finite arrays. To do so we will model 2 arrays: first using the pattern of the isolated element, second with the embedded element pattern and compare the results of the two with the full-wave Method of Moments (MoM) based solution of the array. The array performance for scanning at broadside, and for scanning off broadside is established. Finally, we adjust the array spacing to investigate the occurrence of scan blindness and compare against reference results [3]. For this example we choose the center of the X-band as our design frequency.

```
freq = 10e9;
vp = physconst('lightspeed');
lambda = vp/freq;
```

Model an Array of Dipoles Using Isolated Element Pattern

In [4], it was discussed that the central element of a $5 \lambda \times 5 \lambda$ array starts to behave like it is in an infinite array. Such an aperture would correspond to a 10×10 array of half-wavelength spaced radiators. We choose to slightly exceed this limit and consider a 11×11 array of $\lambda/2$ dipoles.

```
Nrow = 11;
Ncol = 11;
drow = 0.5*lambda;
dcol = 0.5*lambda;
```

Dipole as Antenna Element

The individual element we choose is a dipole. Choose its length to be slightly lower than $\lambda/2$ and radius of approximately $\lambda/150$.

```
mydipole = dipole;  
mydipole.Length = 0.47*lambda;  
mydipole.Width = cylinder2strip(0.191e-3);
```

URA with an Isolated Dipole

Create a 11 X 11 URA and assign the isolated dipole as its element. Adjust the spacing to be half-wavelength at 10 GHz. The dipole tilt is now set to zero, so that its orientation matches the array geometry in the Y-Z plane.

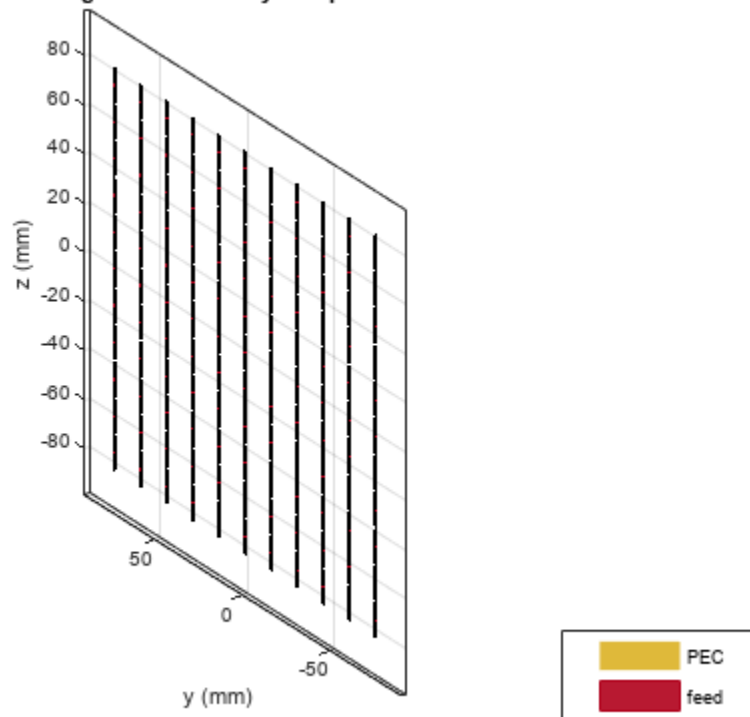
```
myURA2 = phased.URA;  
myURA2.Element = mydipole;  
myURA2.Size = [Nrow Ncol];  
myURA2.ElementSpacing = [drow dcol];
```

Create Full-Wave Model of the 11 X 11 Array

Use the Antenna Toolbox™ to create a full-wave model of the 11 x 11 array of resonant dipoles. Since the default orientation of the dipole element in the library is along z-axis, we tilt it so that the array is initially formed in the X-Y plane and then tilt the array to match the array axis of the URA.

```
myFullWaveArray = rectangularArray;  
myFullWaveArray.Element = mydipole;  
myFullWaveArray.Element.Tilt = 90;  
myFullWaveArray.Element.TiltAxis = [0 1 0];  
myFullWaveArray.Size = [Nrow Ncol];  
myFullWaveArray.RowSpacing = drow;  
myFullWaveArray.ColumnSpacing = dcol;  
myFullWaveArray.Tilt = 90;  
myFullWaveArray.TiltAxis = 'Y';  
figure;  
show(myFullWaveArray)  
title('Rectangular 11 X 11 Array of Dipole Antennas')
```

Rectangular 11 X 11 Array of Dipole Antennas



drawnow

Model Array of Dipoles Using Embedded Element Pattern

Calculate Embedded Element Pattern

Calculate the full 3D embedded element pattern in terms of the electric field magnitude. In [3], the scan resistance and scan reactance for an infinite array of resonant dipoles spaced $\lambda/2$ apart is provided. Choose the resistance at broadside as the termination for all elements. To calculate the embedded element pattern, use the `pattern` function and pass in additional input parameters of the element number(index of the center element) and termination resistance.

```
Zinf = 76 + 1i*31;
ElemCenter = (prod(myFullWaveArray.Size)-1)/2 + 1;
az = -180:2:180;
el = -90:2:90;
h = waitbar(0,'Calculating center element embedded pattern...');
embpattern = pattern(myFullWaveArray,freq,az,el, ...
                    'ElementNumber',ElemCenter, ...
                    'Termination',real(Zinf), ...
                    'Type','efield');
waitbar(1,h,'Pattern computation complete');
delete(h);
```

URA with Embedded Element Pattern

Import this embedded element pattern into the custom antenna element.

```
embpattern = 20*log10(embpattern);  
fmin = freq - 0.1*freq;  
fmax = freq + 0.1*freq;  
freqVector = [fmin fmax];  
EmbAnt = phased.CustomAntennaElement('FrequencyVector',freqVector,...  
    'AzimuthAngles',az,'ElevationAngles',el,...  
    'MagnitudePattern',embpattern,'PhasePattern',zeros(size(embpattern)));
```

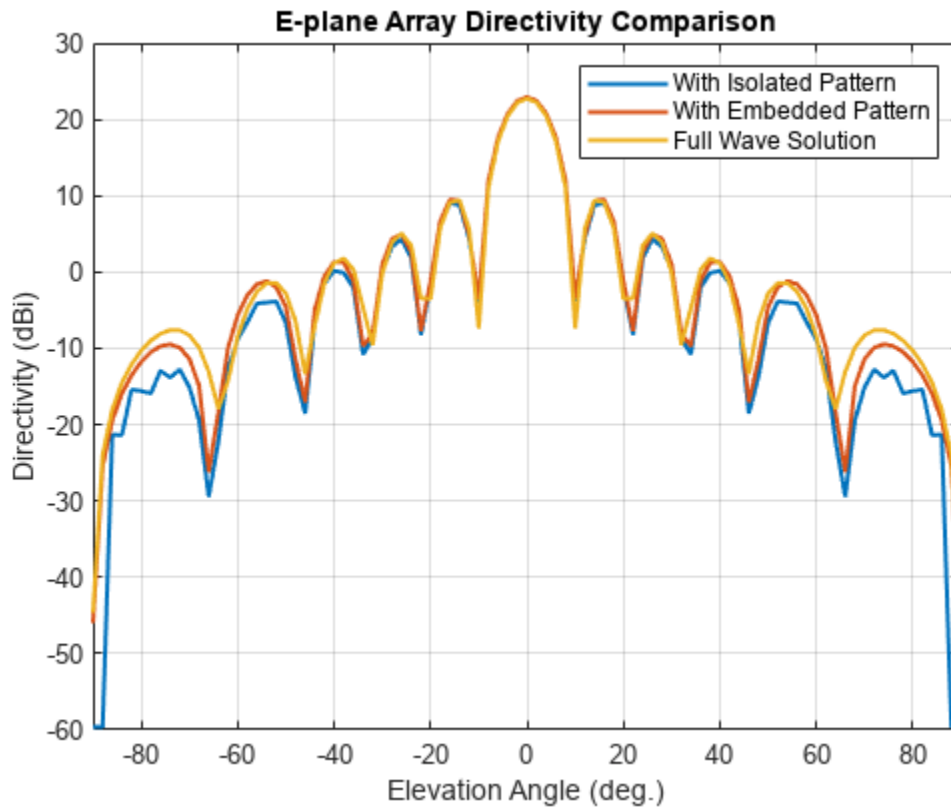
Create a uniform rectangular array(URA), with the custom antenna element, which has the embedded element pattern.

```
myURA1 = phased.URA;  
myURA1.Element = EmbAnt;  
myURA1.Size = [Nrow Ncol];  
myURA1.ElementSpacing = [drow dcol];
```

Compare Array Pattern in Elevation and Azimuth Plane

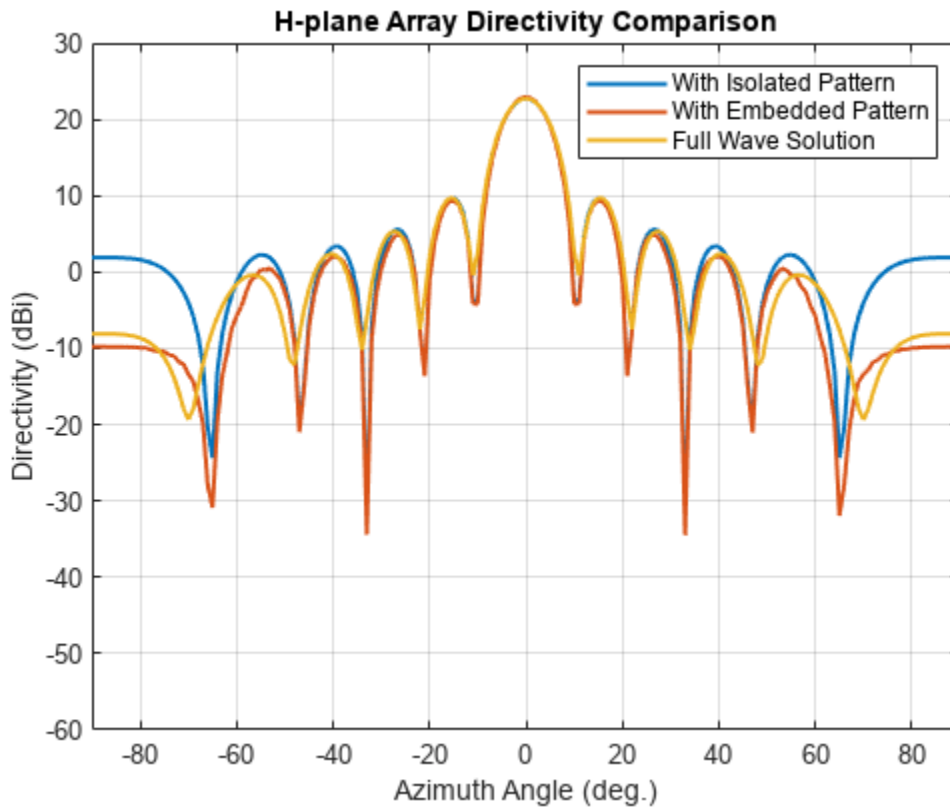
Calculate the pattern in the elevation plane (specified by azimuth = 0 deg and also called the E-plane) and azimuth plane (specified by elevation = 0 deg and called the H-plane) for the three arrays : based on the isolated element pattern, based on the embedded element pattern, and based on the full-wave model.

```
Eplane1 = pattern(myURA1,freq,0,el);  
Eplane2 = pattern(myURA2,freq,0,el);  
[Eplane3,~,el3e] = pattern(myFullWaveArray,freq,0,el);  
figure;  
plot(el,Eplane2,el,Eplane1,el3e,Eplane3,'LineWidth',1.5);  
axis([min(el) max(el) -60 30])  
grid on  
xlabel('Elevation Angle (deg.)');  
ylabel('Directivity (dBi)');  
title('E-plane Array Directivity Comparison')  
legend('With Isolated Pattern','With Embedded Pattern','Full Wave Solution')
```



```
drawnow
```

```
Hplane1 = pattern(myURA1,freq,az/2,0);
Hplane2 = pattern(myURA2,freq,az/2,0);
Hplane3 = pattern(myFullWaveArray,freq,az/2,0);
figure;
plot(az/2,Hplane2,az/2,Hplane1,az/2,Hplane3,'LineWidth',1.5);
axis([min(az/2) max(az/2) -60 30])
grid on
xlabel('Azimuth Angle (deg.)');
ylabel('Directivity (dBi)');
title('H-plane Array Directivity Comparison')
legend('With Isolated Pattern','With Embedded Pattern','Full Wave Solution')
```

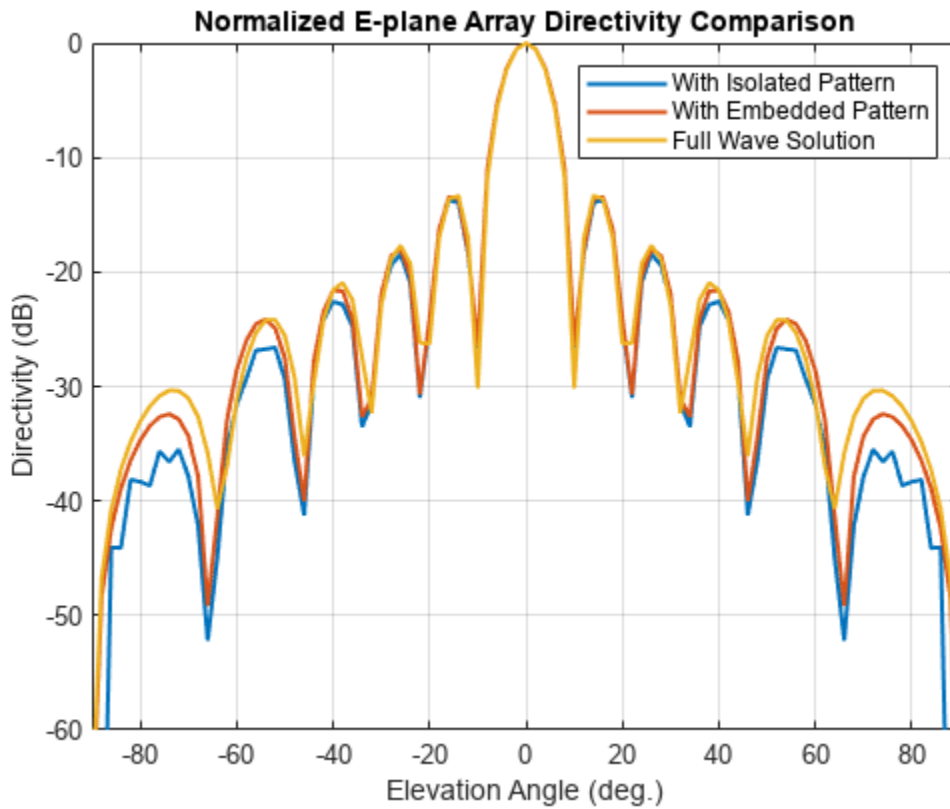


drawnow

The array directivity is approximately 23 dBi. This result is close to the theoretical calculation for the peak directivity [5] after taking into account the absence of a reflector, $D = 4 \pi A / \lambda^2 N_{row} N_{col}$, $A = d_{row} * d_{col}$.

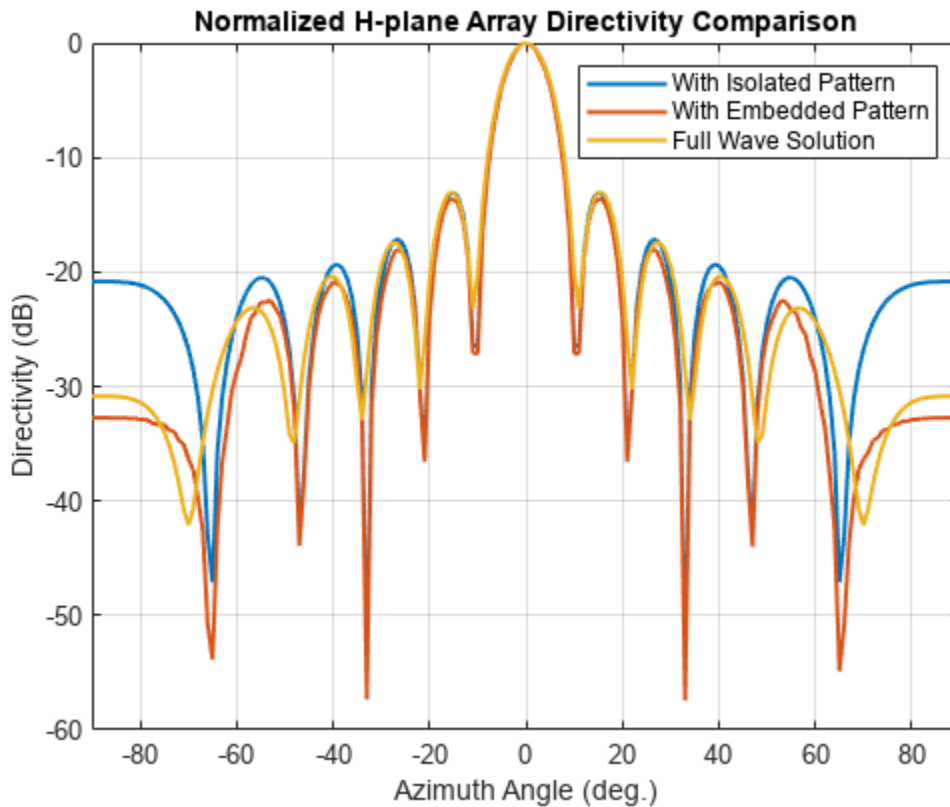
Normalize the directivity for the three arrays and plot it for comparison.

```
figure;
Eplanenormlz1 = Eplane1 - max(Eplane1);
Eplanenormlz2 = Eplane2 - max(Eplane2);
Eplanenormlz3 = Eplane3 - max(Eplane3);
plot(e1,Eplanenormlz2,e1,Eplanenormlz1,e1,Eplanenormlz3,'LineWidth',1.5);
axis([min(e1) max(e1) -60 0])
grid on
xlabel('Elevation Angle (deg.)');
ylabel('Directivity (dB)');
title('Normalized E-plane Array Directivity Comparison')
legend('With Isolated Pattern','With Embedded Pattern','Full Wave Solution')
```



```
drawnow
```

```
figure;
Hplanenormlz1 = Hplane1 - max(Hplane1);
Hplanenormlz2 = Hplane2 - max(Hplane2);
Hplanenormlz3 = Hplane3 - max(Hplane3);
plot(az/2,Hplanenormlz2,az/2,Hplanenormlz1,az/2,Hplanenormlz3,'LineWidth',1.5);
axis([min(el) max(el) -60 0])
grid on
xlabel('Azimuth Angle (deg.)');
ylabel('Directivity (dB)');
title('Normalized H-plane Array Directivity Comparison')
legend('With Isolated Pattern','With Embedded Pattern','Full Wave Solution')
```



drawnow

The pattern comparison suggests that the main beam and the first sidelobes are aligned for all three cases. Moving away from the main beam shows the increasing effect of coupling on the sidelobe level. As expected, the embedded element pattern approach suggests a coupling level in between the full-wave simulation model and the isolated element pattern approach.

Comparison with 25 X 25 Array

The behavior of the array pattern is intimately linked to the embedded element pattern. To understand how our choice of a 11 X 11 array impacts the center element behavior, we increase the array size to a 25 X 25 array ($12.5 \lambda \times 12.5 \lambda$ aperture size). Note that the triangular mesh size for the full wave Method of Moments (MoM) analysis with 625 elements increases to 25000 triangles (40 triangles per dipole) and the computation for the embedded element pattern takes approximately 12 minutes on a 2.4 GHz machine with 32 GB memory. This time can be reduced by lowering the mesh size per element by meshing manually using a maximum edge length of $\lambda/20$.

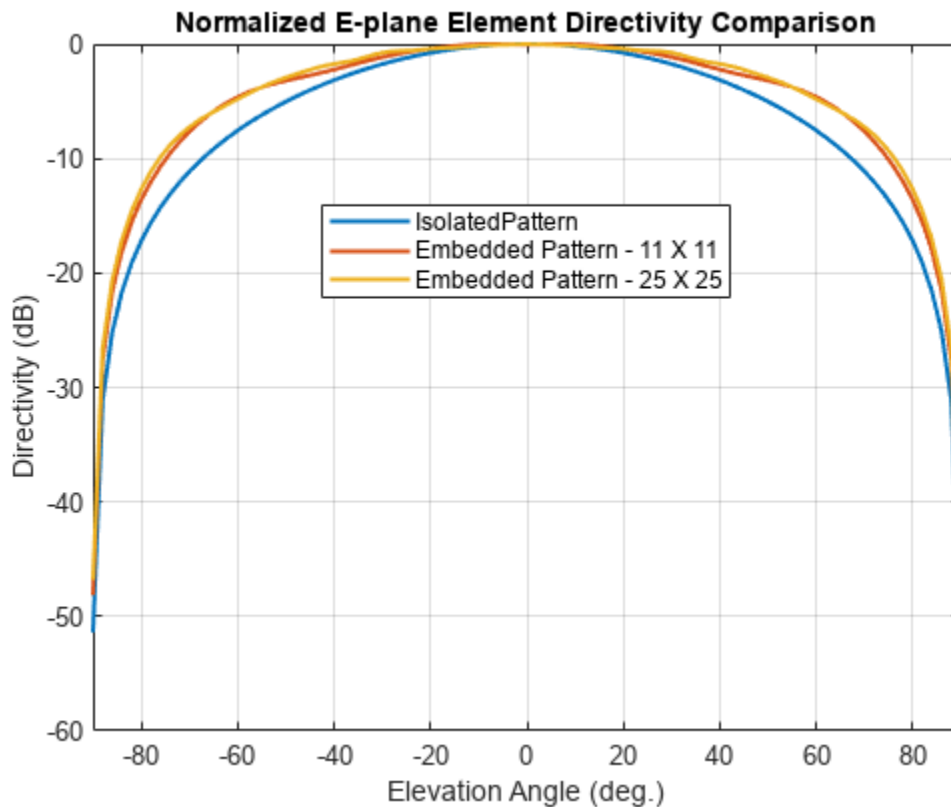
```
load dipolearray
embpattern = 20*log10(DipoleArrayPatData.ElemPat);
EmbAnt2 = clone(EmbAnt);
EmbAnt2.AzimuthAngles = DipoleArrayPatData.AzAngles;
EmbAnt2.ElevationAngles = DipoleArrayPatData.ElAngles;
EmbAnt2.MagnitudePattern = embpattern;
Eplane1 = pattern(EmbAnt2,freq,0,el);
Eplane1 = Eplane1 - max(Eplane1);
Eplane2 = pattern(mydipole,freq,0,el);
Eplane2 = Eplane2 - max(Eplane2);
```



```

embpatE = pattern(EmbAnt, freq, 0, el);
embpatE = embpatE - max(embpatE);
figure;
plot(el, Eplane2, el, embpatE, el, Eplane1, 'LineWidth', 1.5);
axis([min(el) max(el) -60 0])
grid on
xlabel('Elevation Angle (deg.)');
ylabel('Directivity (dB)');
title('Normalized E-plane Element Directivity Comparison')
legend('IsolatedPattern', 'Embedded Pattern - 11 X 11', 'Embedded Pattern - 25 X 25', 'location', 'l')

```

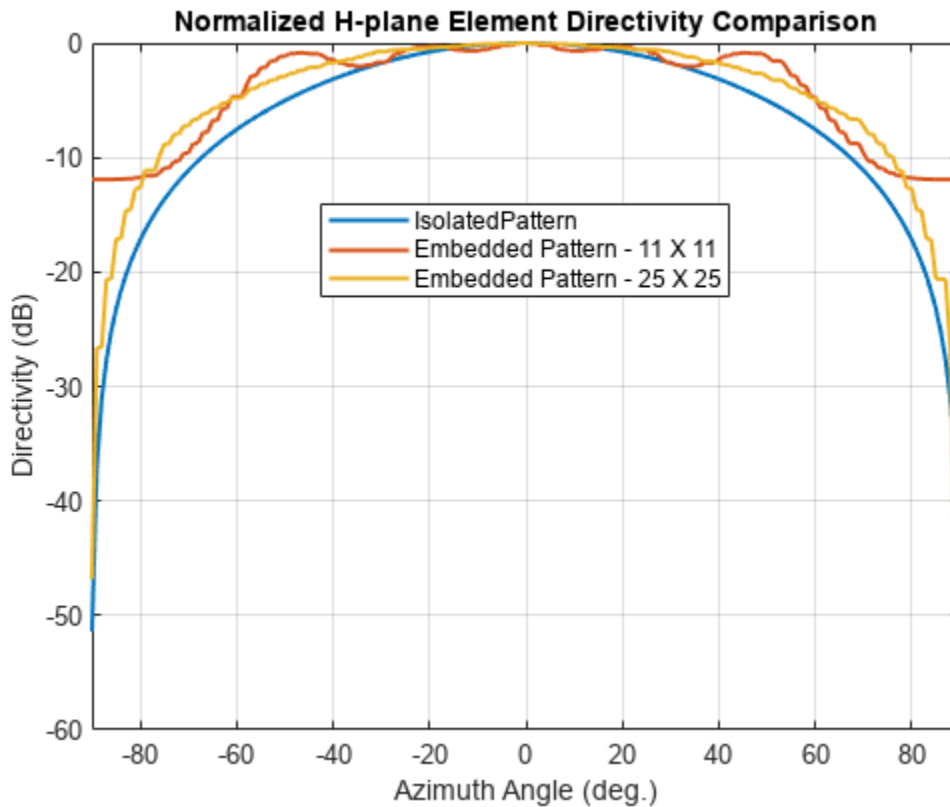


```

drawnow

Hplane1 = pattern(EmbAnt2, freq, 0, az/2);
Hplane1 = Hplane1 - max(Hplane1);
Hplane2 = pattern(mydipole, freq, 0, az/2);
Hplane2 = Hplane2 - max(Hplane2);
embpatH = pattern(EmbAnt, freq, az/2, 0);
embpatH = embpatH - max(embpatH);
figure;
plot(az/2, Hplane2, az/2, embpatH, az/2, Hplane1, 'LineWidth', 1.5);
axis([min(el) max(el) -60 0])
grid on
xlabel('Azimuth Angle (deg.)');
ylabel('Directivity (dB)');
title('Normalized H-plane Element Directivity Comparison')
legend('IsolatedPattern', 'Embedded Pattern - 11 X 11', 'Embedded Pattern - 25 X 25', 'location', 'l')

```



drawnow

The plot above reveals that the difference between embedded element patterns of the 11 X 11 and the 25 X 25 array, respectively, is less than 0.5 dB, in the E-plane. However, the H-plane shows more variation for the 11 X 11 array as compared with the 25 X 25 array.

Scan Behavior and Embedded Element Pattern

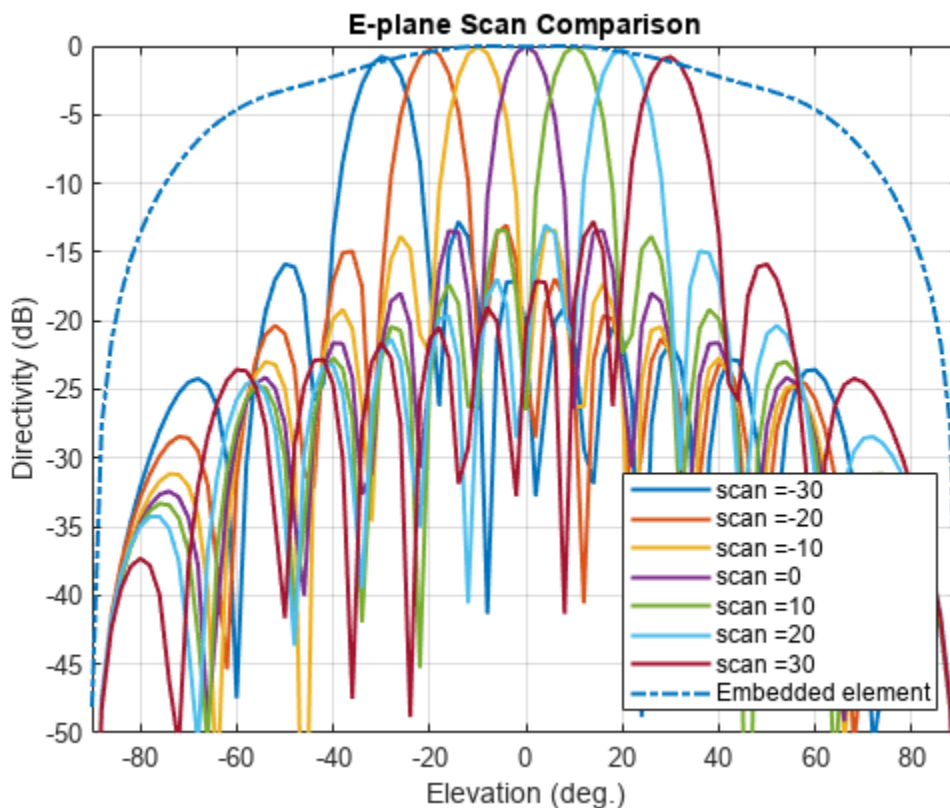
Scan the array based on the embedded element pattern in the elevation plane defined by azimuth = 0 deg and plot the normalized directivity. Also, overlay the normalized embedded element pattern. Note the overall shape of the normalized array pattern approximately follows the normalized embedded element pattern. This is also predicted by the pattern multiplication principle.

```
eplane_indx = find(az==0);
scan_el1 = -30:10:30;
scan_az1 = zeros(1,numel(scan_el1));
scanEplane = [scan_az1;scan_el1];
hsv = phased.SteeringVector;
hsv.SensorArray = myURA1;
hsv.IncludeElementResponse = true;
weights = step(hsv,freq,scanEplane);
legend_string1 = cell(1,numel(scan_el1)+1);
legend_string1{end} = 'Embedded element';
scanEPat = nan(numel(el),numel(scan_el1));
for i = 1:numel(scan_el1)
    scanEPat(:,i) = pattern(myURA1,freq,scan_az1(i),el,'Weights',weights(:,i)); % -23.13;
    legend_string1{i} = strcat('scan = ',num2str(scan_el1(i)));
```

```

end
scanEPat = scanEPat - max(max(scanEPat));
figure;
plot(e1,scanEPat,'LineWidth',1.5);
hold on
grid on
plot(e1,embpatE,'-.','LineWidth',1.5);
axis([min(e1) max(e1) -50 0])
xlabel('Elevation (deg.)')
ylabel('Directivity (dB)')
title('E-plane Scan Comparison')
legend(legend_string1,'Location','southeast')
hold off

```



drawnow

Scan Blindness

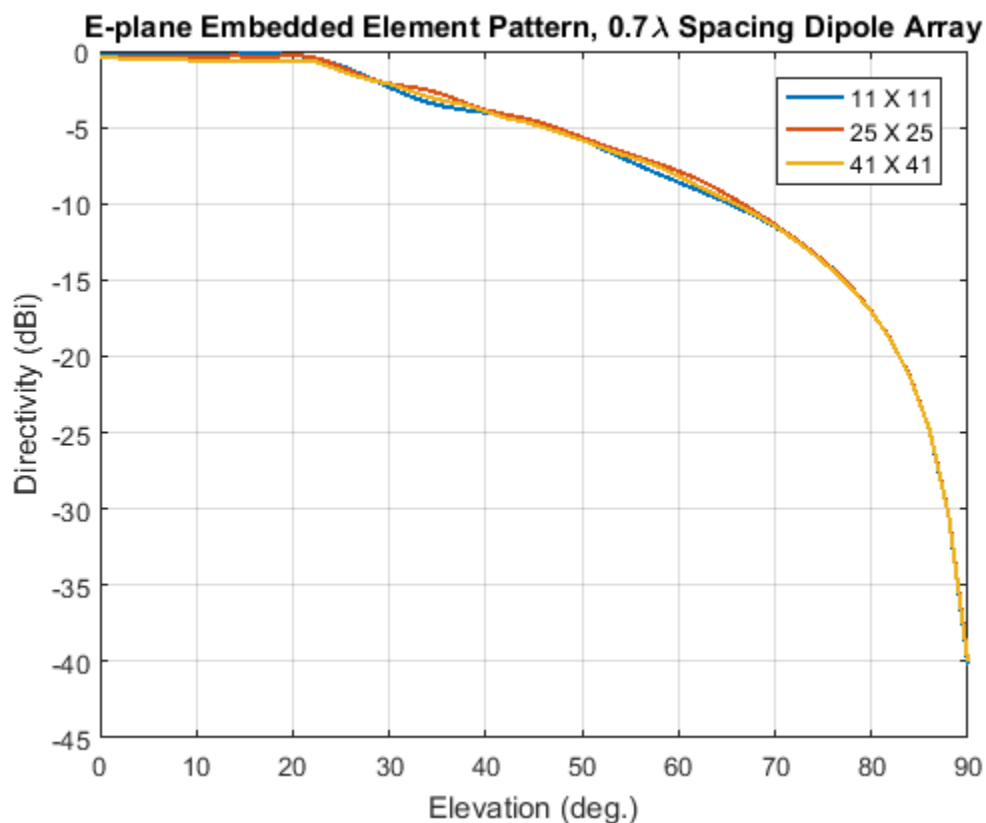
In large arrays, it is possible that the array directivity will reduce drastically at certain scan angles. At these scan angle, referred to as the blind angles, the array does not radiate the power supplied at its input terminals [3]. Two common mechanisms under which blindness conditions occur are

- Surface Wave Excitation
- Grating Lobe Excitation

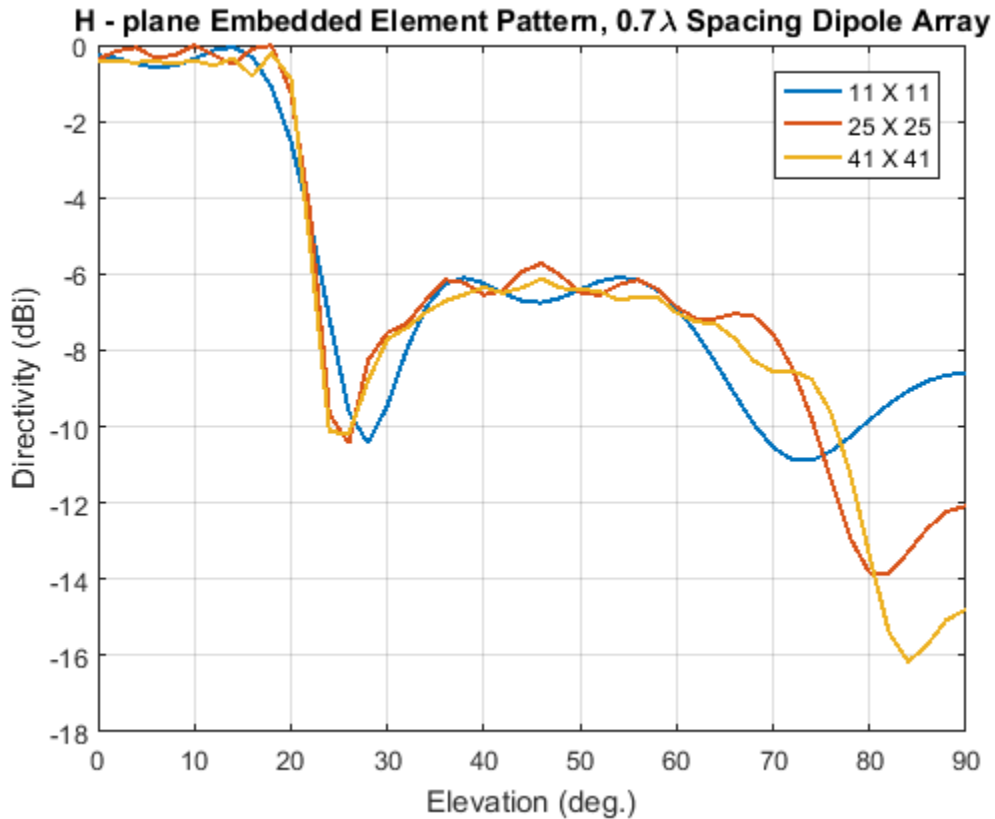
It is possible to detect scan blindness in large finite arrays by studying the embedded element pattern (also known as array element pattern in the infinite array analysis). The array being investigated in

this example does not have a dielectric substrate/ground plane, and therefore the surface waves are eliminated. However we can investigate the second mechanism, i.e. the grating lobe excitation. To do so, let us increase the spacing across rows and columns of the array to be 0.7λ . Since this spacing is greater than the half-wavelength limit we should expect grating lobes in the visible space beyond a specific scan angle. As pointed out in [3], to accurately predict the depth of grating lobe blind angles in the finite array of dipoles, we need to have an array of the size 41×41 or higher. We will compare 3 cases, namely the 11×11 , 25×25 and the 41×41 size arrays and check if the existence of blind angles can at least be observed in the 11×11 array. As mentioned earlier, the results were precomputed in Antenna Toolbox™ and saved in a MAT file. To reduce the computational time, the elements were meshed with maximum edge length of $\lambda/20$.

```
load dipolearrayblindness.mat
```



The normalized E-plane embedded element pattern for arrays of three sizes



The normalized H-plane embedded element pattern for arrays of three sizes. Notice the blind angle around 24-26 deg.

Conclusion

The embedded element pattern approach is one possible way of performing the analysis of large finite arrays. They need to be so large that the edge effects can be ignored. The isolated element pattern is replaced with the embedded element pattern which includes the effect of mutual coupling.

Reference

- [1] R. J. Mailloux, 'Phased Array Antenna Handbook', Artech House, 2nd edition, 2005
- [2] W. Stutzman, G. Thiele, 'Antenna Theory and Design', John Wiley & Sons Inc., 3rd Edition, 2013.
- [3] R. C. Hansen, Phased Array Antennas, Chapter 7 and 8, John Wiley & Sons Inc., 2nd Edition, 1998.
- [4] H. Holter, H. Steyskal, "On the size requirement for finite phased-array models," IEEE Transactions on Antennas and Propagation, vol.50, no.6, pp.836-840, Jun 2002.

[5] P. W. Hannan, "The Element-Gain Paradox for a Phased-Array Antenna," IEEE Transactions on Antennas Propagation, vol. 12, no. 4, July 1964, pp. 423-433.

See Also

"Verification of Far-Field Array Pattern Using Superposition with Embedded Element Patterns" on page 5-372

Modeling Resonant Coupled Wireless Power Transfer System

This example shows how to create and analyze a resonant coupling type wireless power transfer (WPT) system with an emphasis on concepts such as resonant mode, coupling effect, and magnetic field pattern. The analysis is based on a two-element system of spiral resonators.

Design Frequency and System Parameters

Choose the design frequency to be 30 MHz. This is a popular frequency for compact WPT system design. Also specify the frequency for broadband analysis, and the points in space to plot near fields.

```
fc=30e6;
fcmin = 28e6;
fcmax = 31e6;
fband1 = 27e6:1e6:fcmin;
fband2 = fcmin:0.25e6:fcmax;
fband3 = fcmax:1e6:32e6;
freq = unique([fband1 fband2 fband3]);
pt=linspace(-0.3,0.3,61);
[X,Y,Z]=meshgrid(pt,0,pt);
field_p=[X(:)';Y(:)';Z(:)'];
```

The Spiral Resonator

The spiral is a very popular geometry in a resonant coupling type wireless power transfer system for its compact size and highly confined magnetic field. We will use such a spiral as the fundamental element in this example.

Create Spiral Geometry

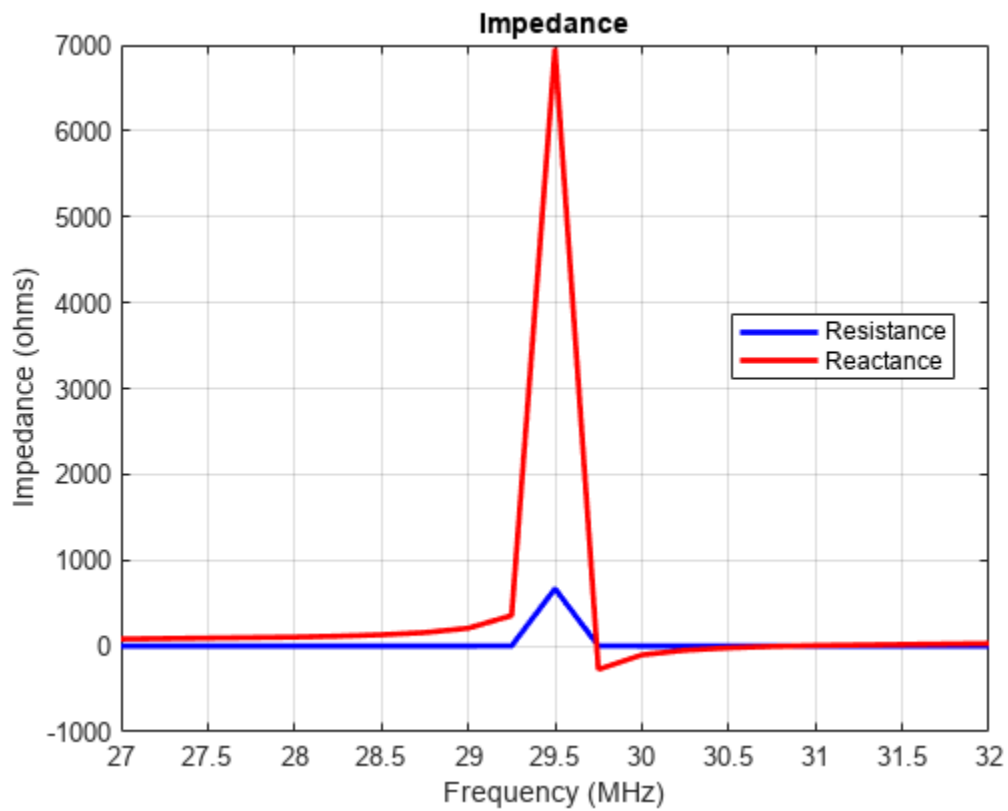
The spiral is defined by its inner and outer radius, and number of turns.

```
Rin = 0.05;
Rout = 0.15;
N = 6.25;
spiralobj = spiralArchimedean('NumArms', 1, 'Turns', N,
    'InnerRadius', Rin, 'OuterRadius', Rout, 'Tilt', 90, 'TiltAxis', 'Y');
```

Resonance Frequency and Mode

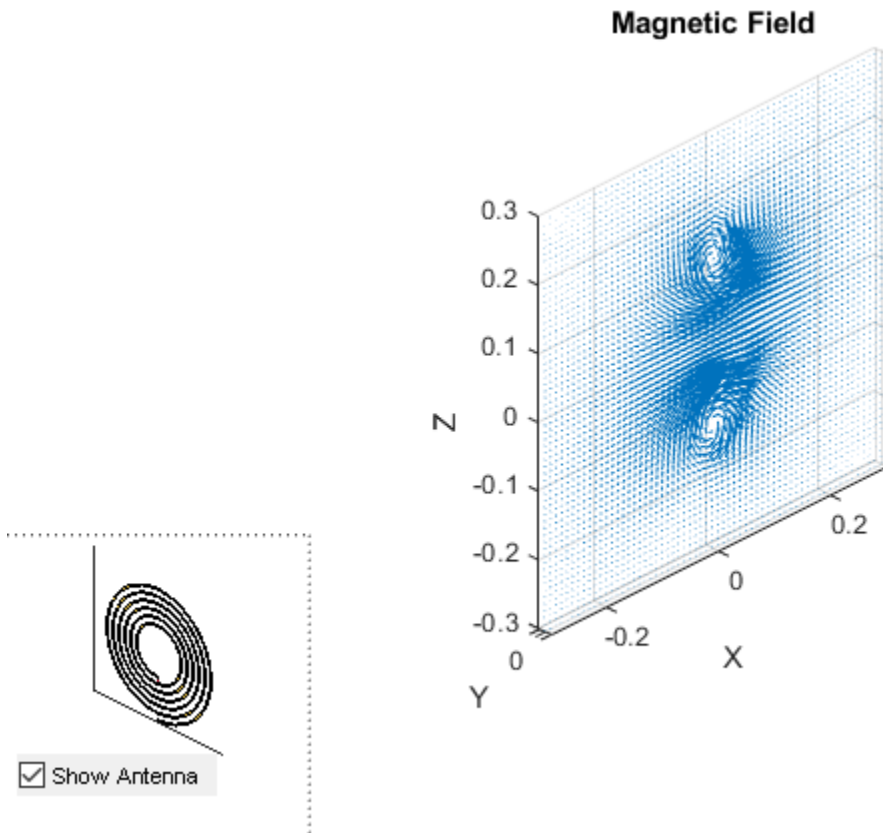
It is important to find the resonant frequency of the designed spiral geometry. A good way to find the resonant frequency is to study the impedance of the spiral resonator. Since the spiral is a magnetic resonator, a lorentz shaped reactance is expected and observed in the calculated impedance result.

```
figure
impedance(spiralobj, freq);
```



Since the spiral is a magnetic resonator, the dominant field component of this resonance is the magnetic field. A strongly localized magnetic field is observed when the near field is plotted.

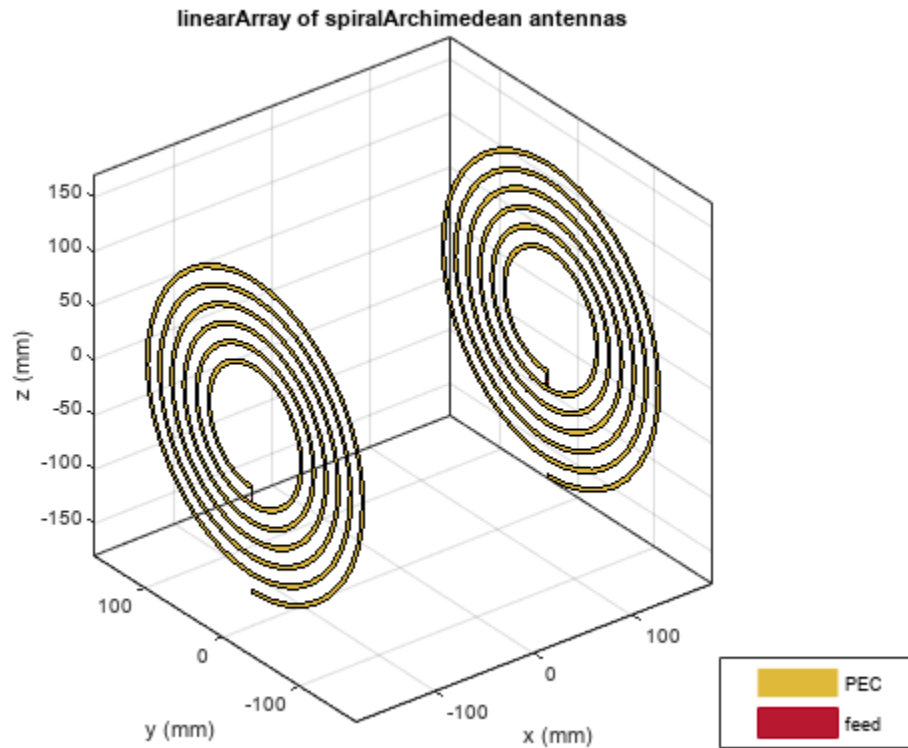
figure
`EHfields(spiralobj,fc,field_p,'ViewField','H','ScaleFields',[0 5]);`



Create Spiral to Spiral Power Transfer System

The complete wireless power transfer system is composed of two parts: the transmitter(Tx) and receiver(Rx). Choose identical resonators for both transmitter and receiver to maximize the transfer efficiency. Here, the wireless power transfer system is modeled as a linear array.

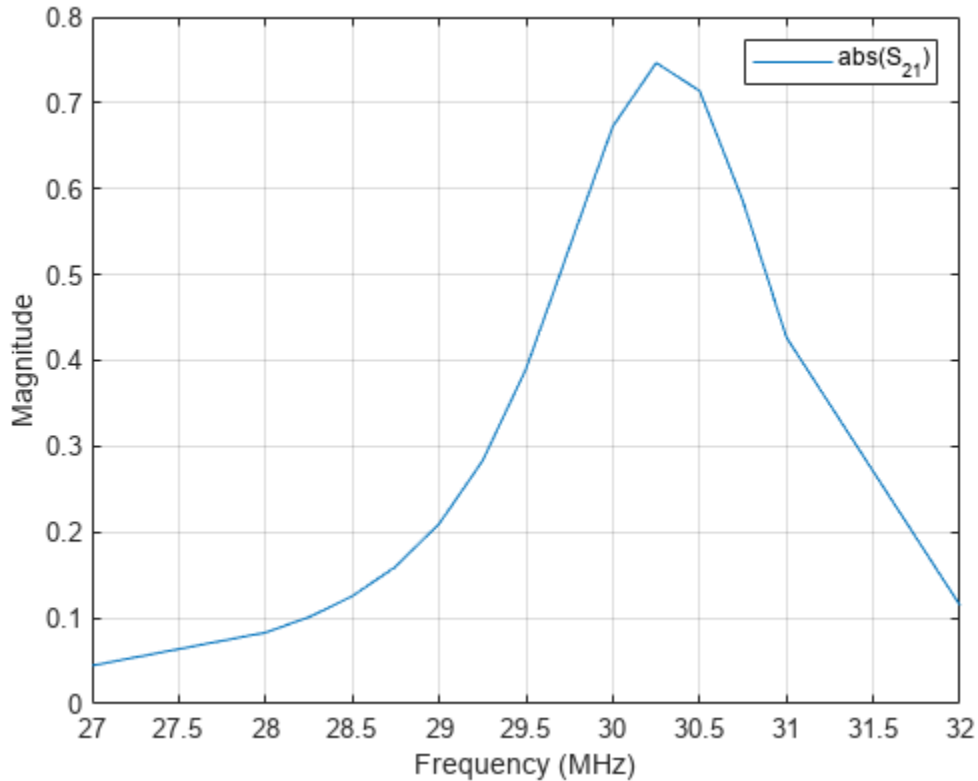
```
wptsys = linearArray('Element',[spiralobj spiralobj]);
wptsys.ElementSpacing = Rout*2;
figure
show(wptsys);
```



Variation of System Efficiency with Transfer Distance

One way to evaluate the efficiency of the system is by studying the S21 parameter. As presented in [[1]], the system efficiency changes rapidly with operating frequency and the coupling strength between the transmitter and receiver resonator. Peak efficiency occurs when the system is operating at its resonant frequency, and the two resonators are strongly coupled.

```
sparam = sparameters(wptsys, freq);  
figure  
rfplot(sparam,2,1,'abs');
```



Critical Coupled Point

The coupling between two spirals increases with decreasing distance between two resonators. This trend is approximately proportional to $1/d^3$. Therefore, the system efficiency increases with shorter transfer distance until it reaches the critical coupled regime [1]. When the two spirals are over coupled, exceeding the critical coupled threshold, system efficiency remains at its peak, as shown in Fig.3 in [1]. We observe this critical coupling point and over coupling effect during modeling of the system. Perform a parametric study of the system s-parameters as a function of the transfer distance. The transfer distance is varied by changing the ElementSpacing. It is varied from half of the spiral dimension to one and half times of the spiral dimension, which is twice of the spiral's outer radius. The frequency range is expanded and set from 25 MHz to 36 MHz.

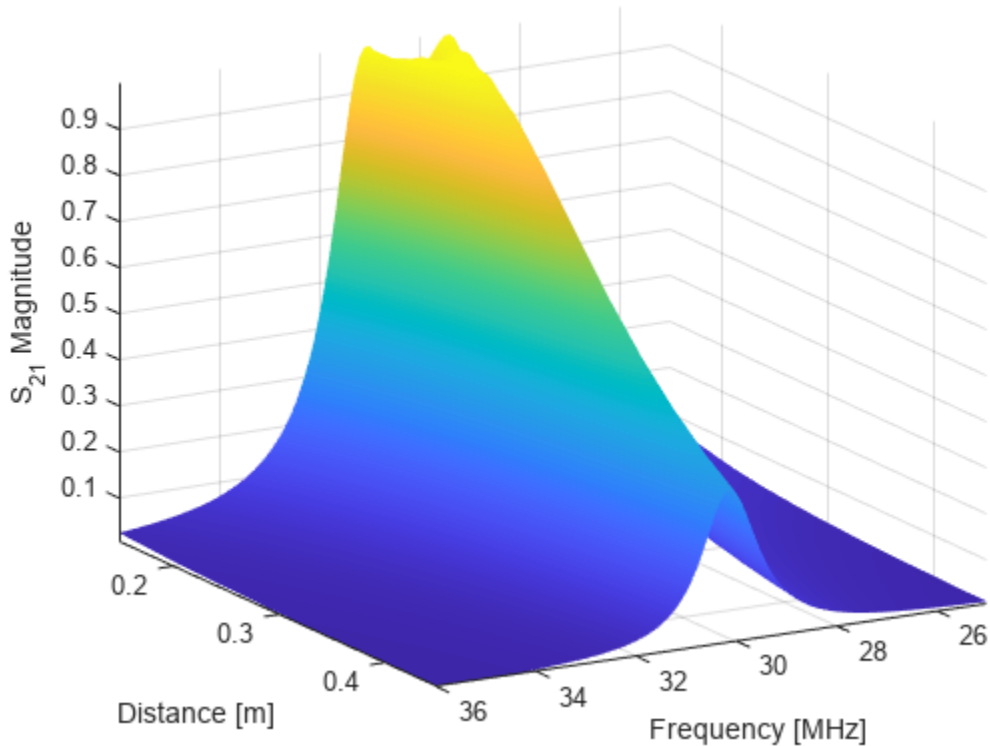
```

freq = (25:0.1:36)*1e6;
dist = Rout*2*(0.5:0.1:1.5);

load('wptData.mat');
s21_dist = zeros(length(dist),length(freq));
for i = 1:length(dist)
    s21_dist(i,:) = rfparam(sparam_dist(i),2,1);
end

figure;
[X,Y] = meshgrid(freq/1e6,dist);
surf(X,Y,abs(s21_dist),'EdgeColor','none');
view(150,20);
shading(gca,'interp');
```

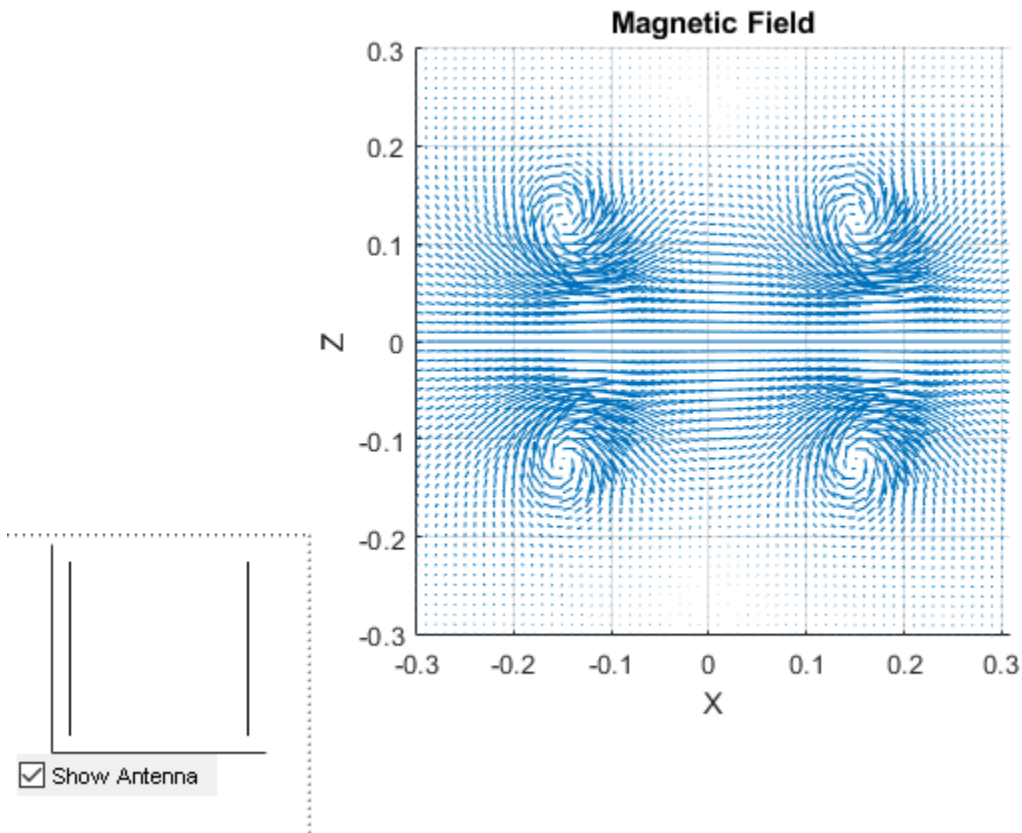
```
axis tight;
xlabel('Frequency [MHz]');
ylabel('Distance [m]');
zlabel('S_{21} Magnitude');
```



Coupling Mode between Two Spiral Resonator

The dominant energy exchange mechanism between the two spiral resonators is through the magnetic field. Strong magnetic fields are present between the two spirals at the resonant frequency.

```
wptsys.ElementSpacing = Rout*2;
figure
EHfields(wptsys,fc,field_p,'ViewField','H','ScaleFields',[0 5]);
view(0,0);
```

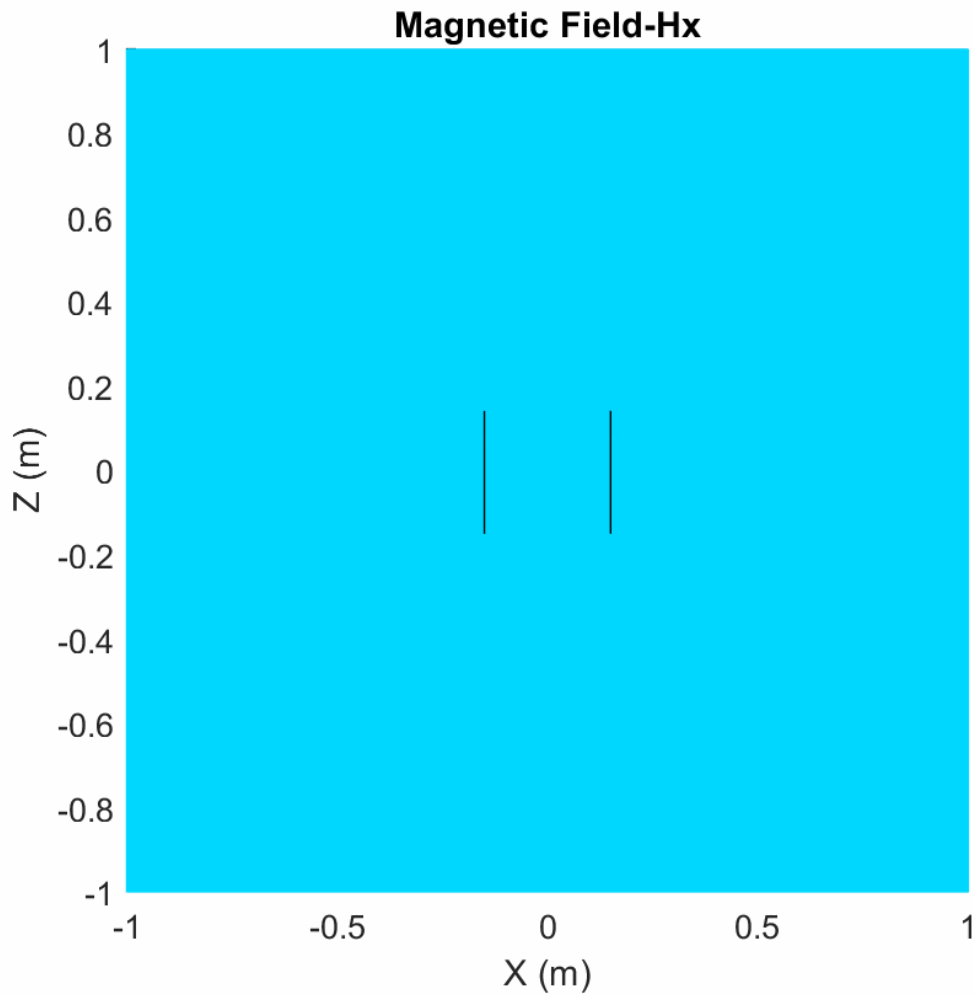


Animate the H-field Coupling

The analysis from the solver provides us with the steady state result. The field calculation provides the 3 components of the magnetic field in phasor form in cartesian coordinate system. Animate this field by using a propagator defined by the frequency, f_c to bring the result back to time-harmonic form and visualize on the same slice plane XZ either of the components H_x or H_z . To do so use the helper `AnimateEHFields` function as shown below. Define the spatial extent, point density as well as the total number of time periods defined in terms of the frequency f_c . If the sampling time is not defined, the function automatically chooses a sampling time based on 10 times the center frequency. There is also an option available to save the animation output to a gif file with an appropriate name specified.

```
% Excite the first resonator only
wptsys.AmplitudeTaper = [1 0];
% Define Spatial Plane Extent
SpatialInfo.XSpan = 2;
SpatialInfo.YSpan = 2;
SpatialInfo.ZSpan = 2;
SpatialInfo.NumPointsX = 200;
SpatialInfo.NumPointsY = 200;
SpatialInfo.NumPointsZ = 200;
% Define Time Extent
Tperiod = 1/fc;
T = 2*Tperiod;
TimeInfo.TotalTime = T;
TimeInfo.SamplingTime = []; % Leave it empty for picking automatically
% Define Field Data details
```

```
DataInfo.Component = 'Hx';  
DataInfo.PlotPlane = 'XZ';  
DataInfo.DataLimits = [-0.25 0.5];  
DataInfo.ScaleFactor = 1;  
DataInfo.GifFileName = 'wptcpl.gif';  
DataInfo.CloseFigure = true;  
%% Animate  
helperAnimateEHFields(wptsys,fc,SpatialInfo, TimeInfo, DataInfo)
```



Conclusion

The results obtained for the wireless power transfer system match well with the results published in [[1]].

See Also

More About

- “Archimedean Spiral Design Investigation” on page 5-88

References

- [1] Sample, Alanson P, D A Meyer, and J R Smith. "Analysis, Experimental Results, and Range Adaptation of Magnetically Coupled Resonators for Wireless Power Transfer." *IEEE Transactions on Industrial Electronics* 58, no. 2 (February 2011): 544-54. <https://doi.org/10.1109/TIE.2010.2046002>.

Crossed-Dipole (Turnstile) Antenna and Array

The turnstile antenna invented in 1936 by Brown [1] is a valuable tool to create a circularly-polarized pattern (RHCP or LHCP). It is commonly used in mobile communications.

Turnstile Antenna Parameters

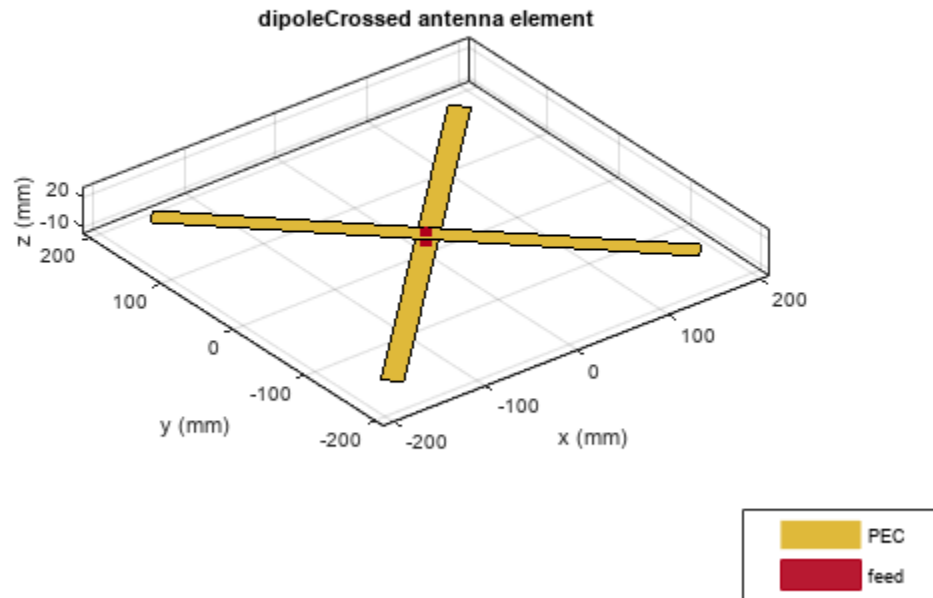
The turnstile antenna usually operates at the fundamental (series) resonance of the dipole-like antenna. In order to achieve circular polarization, the turnstile antenna has either an external quadrature hybrid as a 90 degree power divider/combiner or an internal built-in phase shifting network. In this example, the antenna is designed for 300 MHz. The spacing between the two crossed dipoles is of the order of $\lambda/50$.

```
freq    = 300e6;  
lambda  = 3e8/freq;  
offset  = lambda/50;  
spacing = lambda/2;  
length  = lambda/2.1;  
width   = lambda/50;  
anglevar= 0:10:180;  
freqrange = 200e6:2e6:400e6;  
gndspacing = lambda/4;
```

Turnstile Antenna

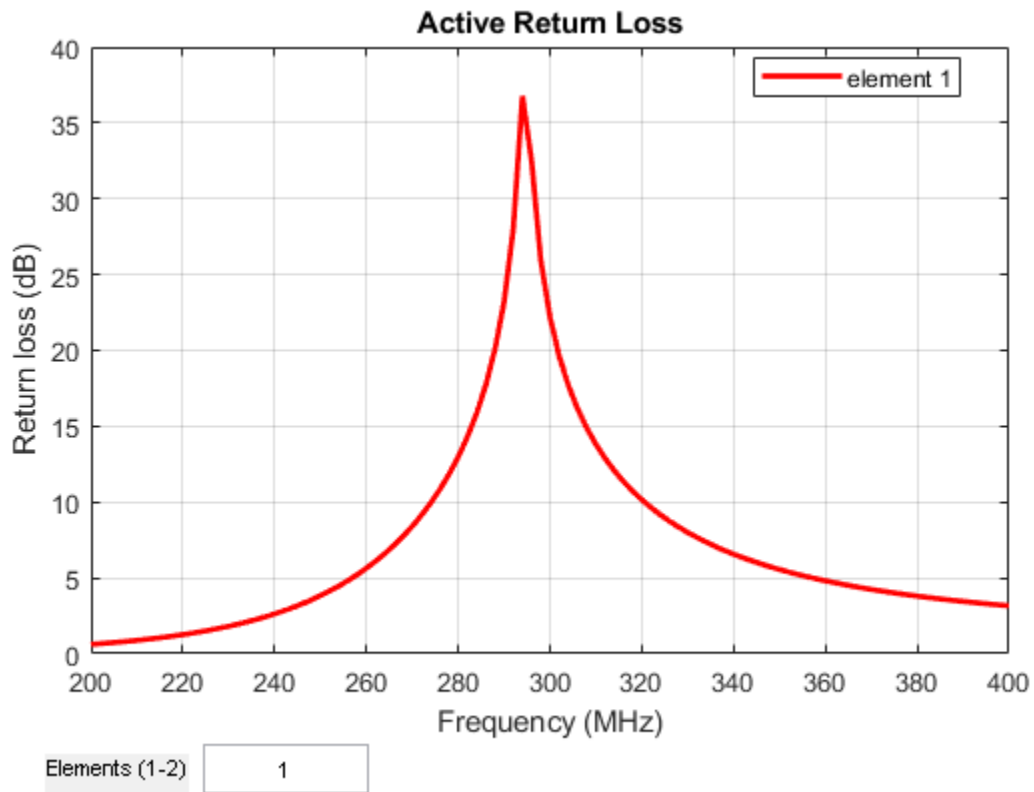
The turnstile antenna is created by using two identical dipoles oriented at right angles to each other. The default crossed dipole catalog element is rotated by 90 degrees to set it up in the X-Y plane. The desired 90 degree phase shift is obtained by specifying the phase shift of the second dipole to 90 degree

```
d = dipole('Length',length,'Width',width);  
ant= dipoleCrossed('Element',d,'Tilt',90,'TiltAxis',[0 1 0]);  
figure; show(ant);
```

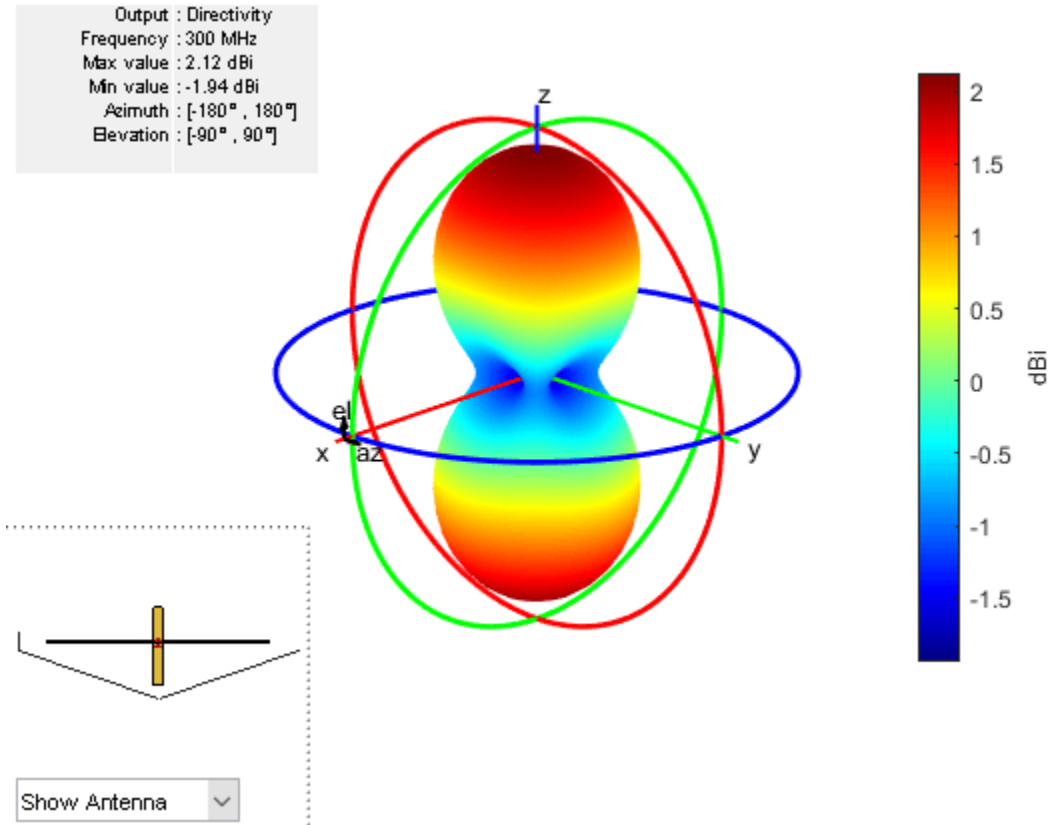
The plot below shows the return loss of the first element of the turnstile antenna. As the two elements are identical, the return loss of the second dipole should be the same. The elements are well matched to a 75 ohm system.

```
figure;  
returnLoss(ant, freqrange, 75);
```



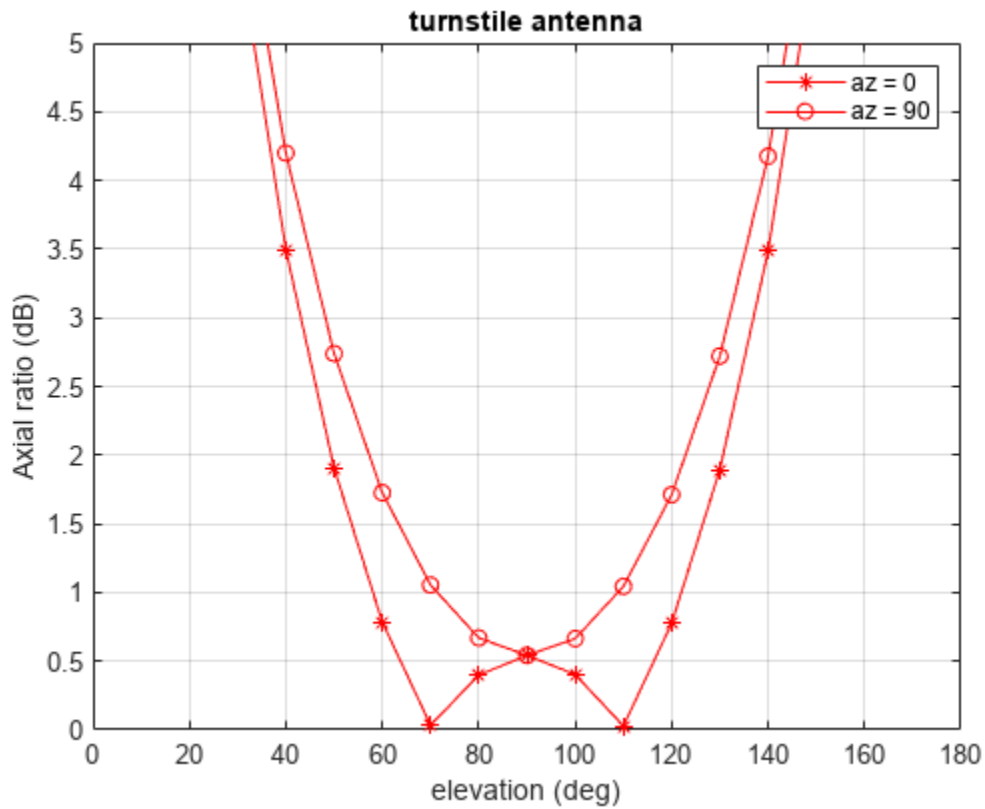
The radiation pattern of the crossed-dipole is symmetric about the x-y plane and the peak value is close to 2.1dBi.

```
pattern(ant, freq);
```



The axial ratio of the turnstile is calculated and plotted in the two principal planes. As can be seen in the plot, the axial ratio is less than 3dB, around 45 degrees on either side of boresight. This indicates that the antenna gives close to circular polarization in the 90 degree region around the boresight.

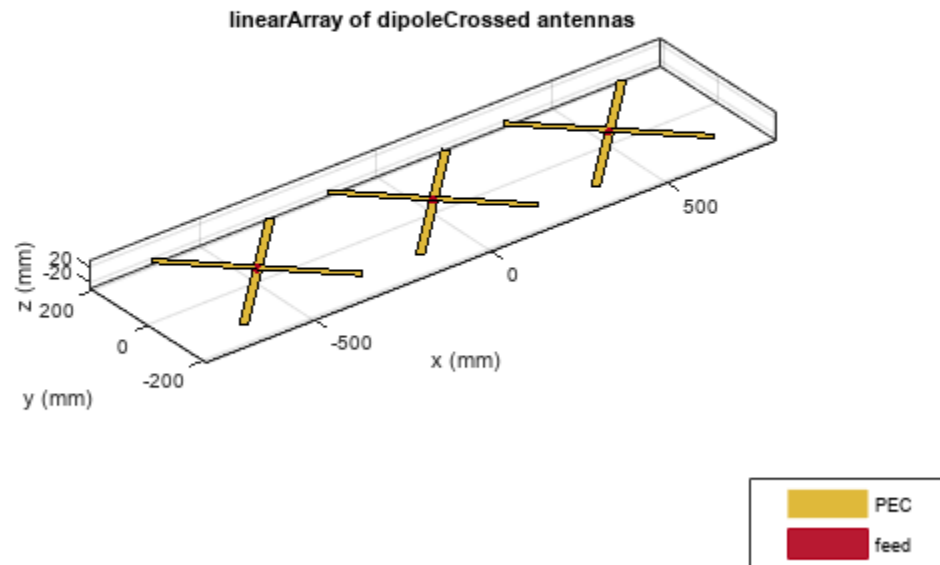
```
AR1 = axialRatio(ant, freq, 0, anglevar);
AR2 = axialRatio(ant, freq, 90, anglevar);
figure;
plot(anglevar, AR1, 'r*- ', anglevar, AR2, 'ro- ');
axis([0 180 0 5]);
grid on;
xlabel('elevation (deg)')
legend('az = 0', 'az = 90')
ylabel('Axial ratio (dB)');
title('turnstile antenna')
```



3 Element Turnstile Array

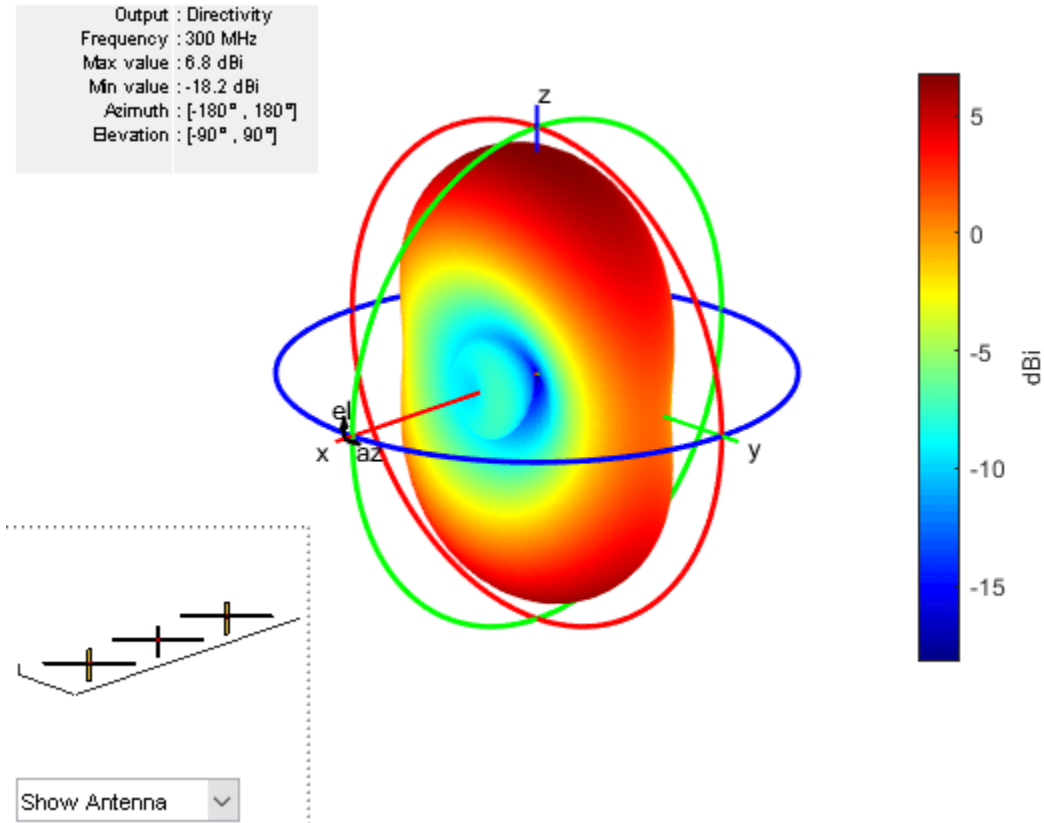
The figure below shows a three element turnstile array. The spacing between the elements is chosen to be $\lambda/2$. The first turnstile element is at the origin while the other two elements are one half wavelength away.

```
arr=linearArray('Element',ant,'ElementSpacing',spacing,'NumElements',3);  
show(arr);
```



The plot below shows the directivity of the resultant array. The peak value is close to 6.8 dBi. The pattern is still symmetric about the X-Y plane.

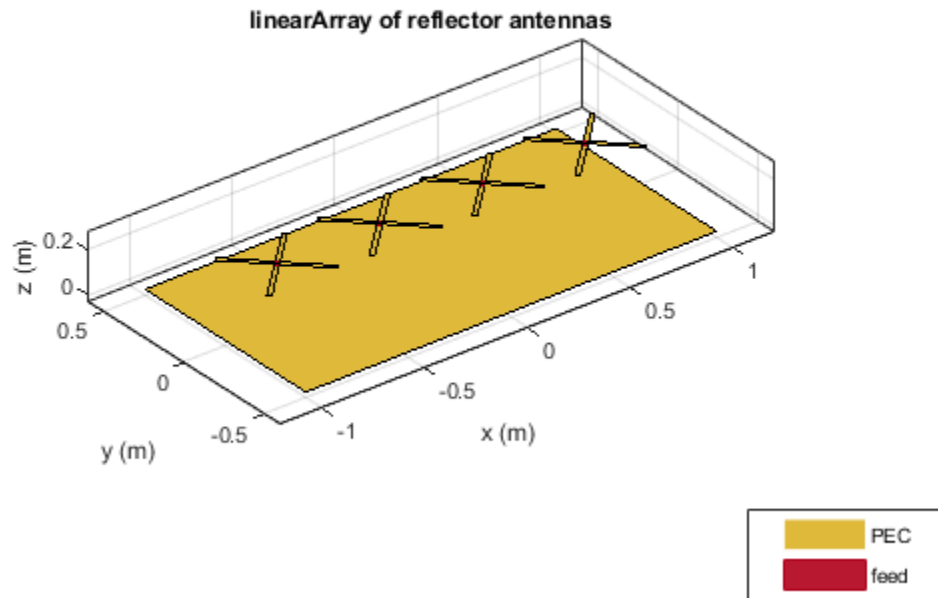
```
pattern(arr, freq);
```



4 Element Turnstile Array with Reflector

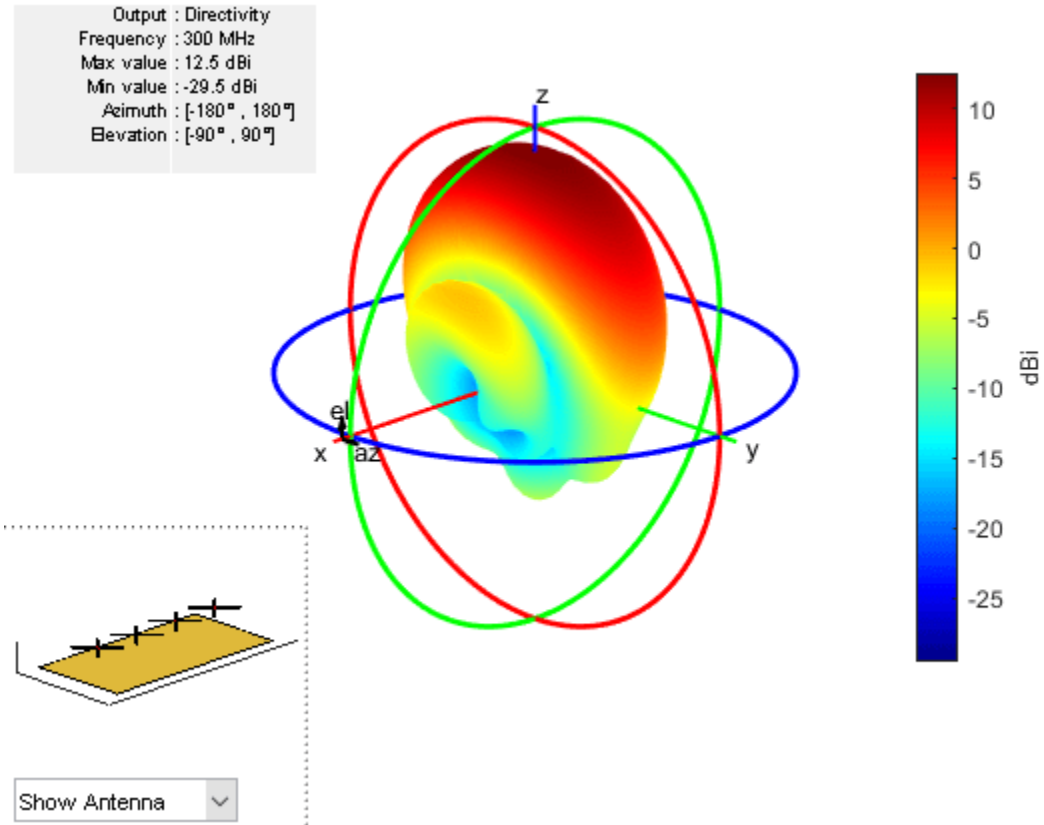
To prevent loss of radiation below the x-y plane, a reflector can be added. Below is a four element turnstile array backed by a reflector. The reflector length is twice the wavelength while the width is a wavelength. The spacing between the reflector and the antenna is quarter wavelength. The array is symmetric about the origin, so there is no element at the origin.

```
r = reflector('Exciter',ant,'GroundPlaneLength',spacing, ...
    'GroundPlaneWidth', lambda,'Spacing',gndspacing);
refarray =linearArray('Element',r,'ElementSpacing',spacing,'NumElements',4);
show(refarray);
```



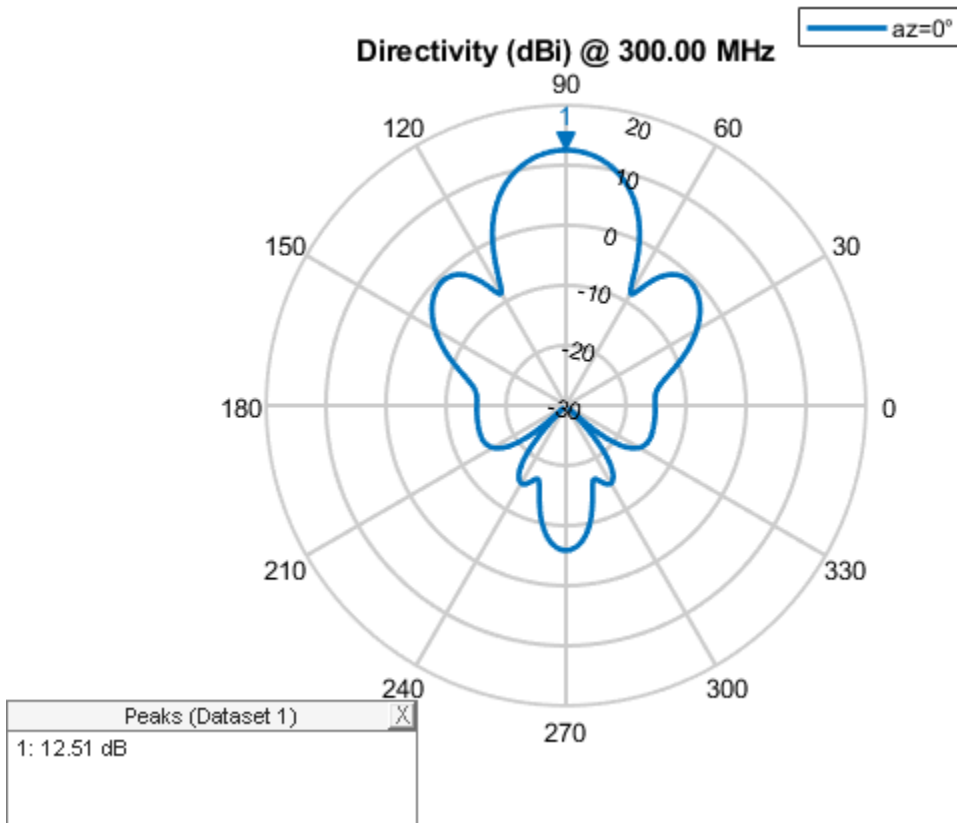
The plot below shows the radiation pattern of the four element array. The peak value is close to 12.6 dBi. The presence of the reflector ensures that most of the energy is radiated along the positive z-axis.

```
figure;  
pattern(refarray, freq);
```



The plot below shows the pattern slice at zero azimuth angle.

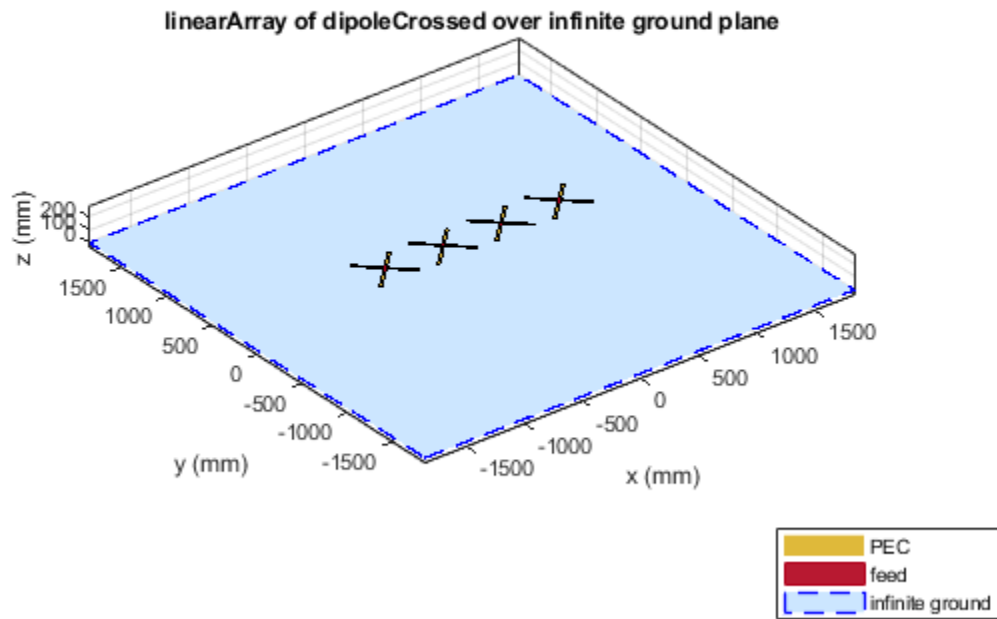
```
figure;
patternElevation(refarray, freq);
```

4 Element Turnstile Array over Infinite Ground Plane

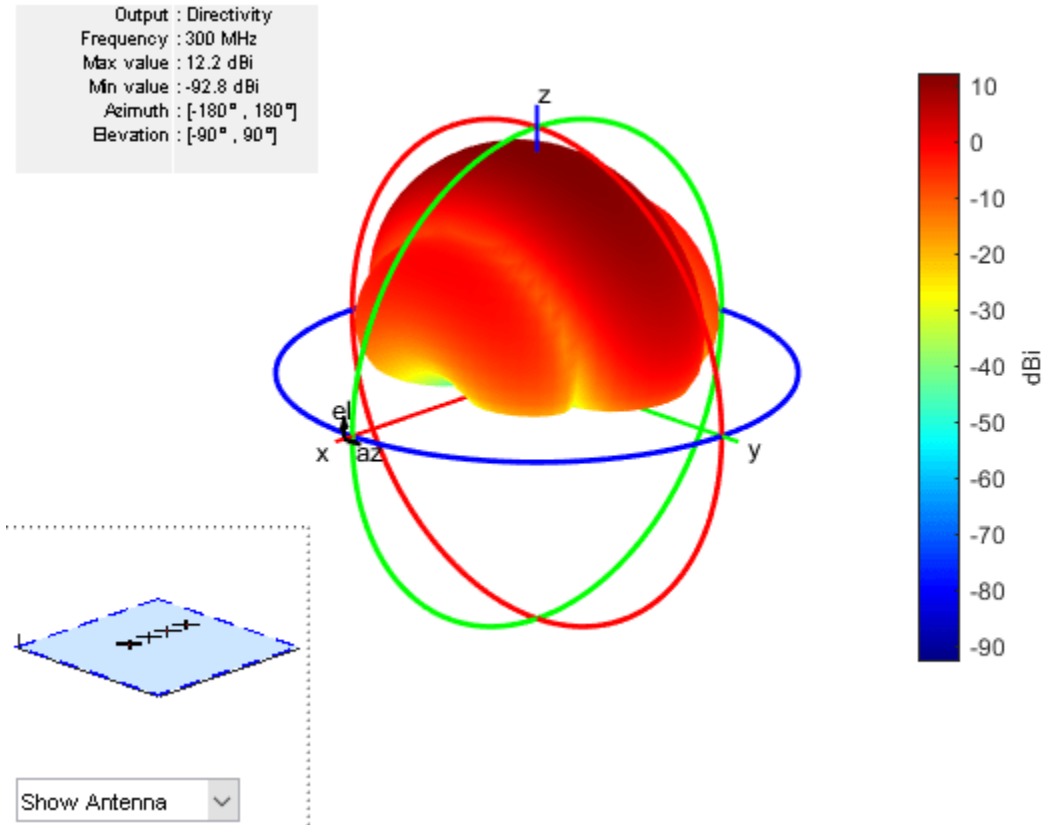
A turnstile array over an electrically large structure can be approximated by placing it over an infinite ground plane. This can be achieved by making the ground plane length of the reflector infinite.

```
refarray.Element.GroundPlaneLength = inf;
show(refarray);
```



The plot below shows the radiation pattern of the turnstile array over an infinite ground plane. As expected, no energy is leaked below the ground.

```
pattern(refarray, freq);
```



Reference

- [1] G. H. Brown, "The turnstile antenna," *Electronics*, April 1936, pp. 14-17.

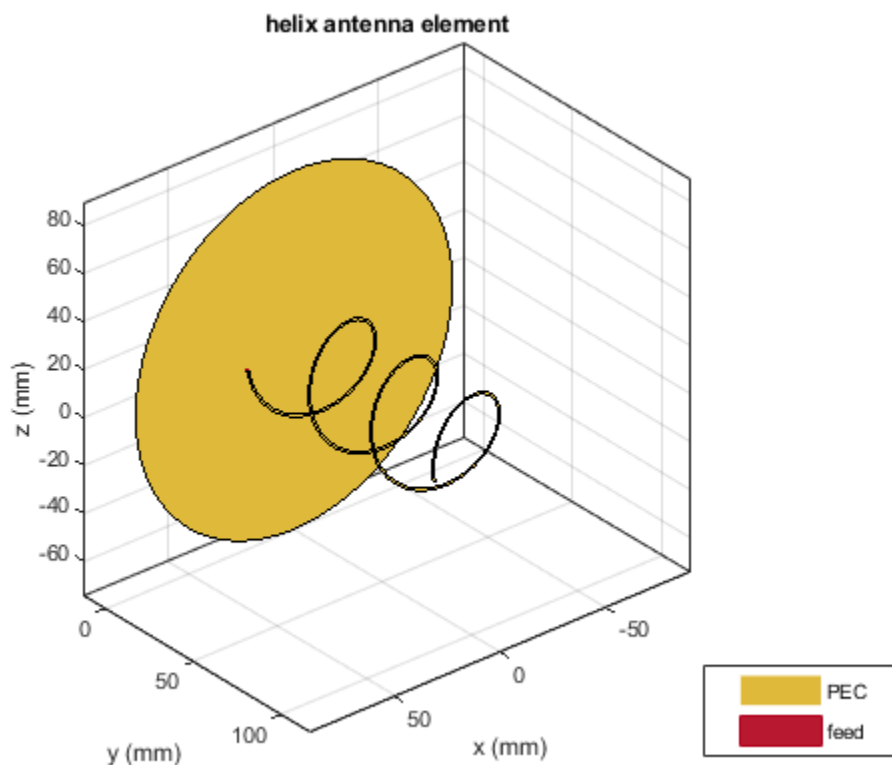
Visualize Antenna Field Strength Map on Earth

This example shows how to calculate an antenna's field strength on flat earth and display it on a map.

Create Helix

Use the default helix, and rotate it by 90 degrees so that most of its field is along the XY-plane, i.e. on the Earth's surface.

```
f0 = 2.4e9;
ant = helix('Tilt',-90);
f = figure;
show(ant)
view(-220,30)
```



Calculate the Electric Field at Various Points in Space

Assume the origin of the tilted antenna is located 15 m above the ground. Calculate the electric field in a rectangular region covering 500-by-500 km.

```
z = -15;
x = (-250:4:250)*1e3;
y = (-100:4:400)*1e3;

[X,Y] = meshgrid(x,y);
numpoints = length(x)*length(y);
```

```
points = [X(:)'; Y(:)'; z*ones(1,numel(X))];
```

```
E = EHfields(ant,f0,points); % Units: V/m
```

Calculate the Magnitude of Electric Field

Compute the magnitude of the electric field.

```
Emag = zeros(1,numpoints);
```

```
for m=1:numpoints
    Emag(m) = norm(E(:,m)/sqrt(2));
end
```

```
Emag = 20*log10(reshape(Emag,length(y),length(x))); % Units: dBV/m
```

```
Emag = Emag + 120; % Units: dBuV/m
```

Plot the Electric Field

Plot the magnitude of the electric field as a function of the x and y distance.

```
d_min = min(Emag(:));
```

```
d_max = max(Emag(:));
```

```
del = (d_max-d_min)/12;
```

```
d_vec = round(d_min:del:d_max);
```

```
if isvalid(f)
```

```
    close(f)
```

```
end
```

```
figure
```

```
contourf(X*1e-3,Y*1e-3,Emag,d_vec,'showtext','on')
```

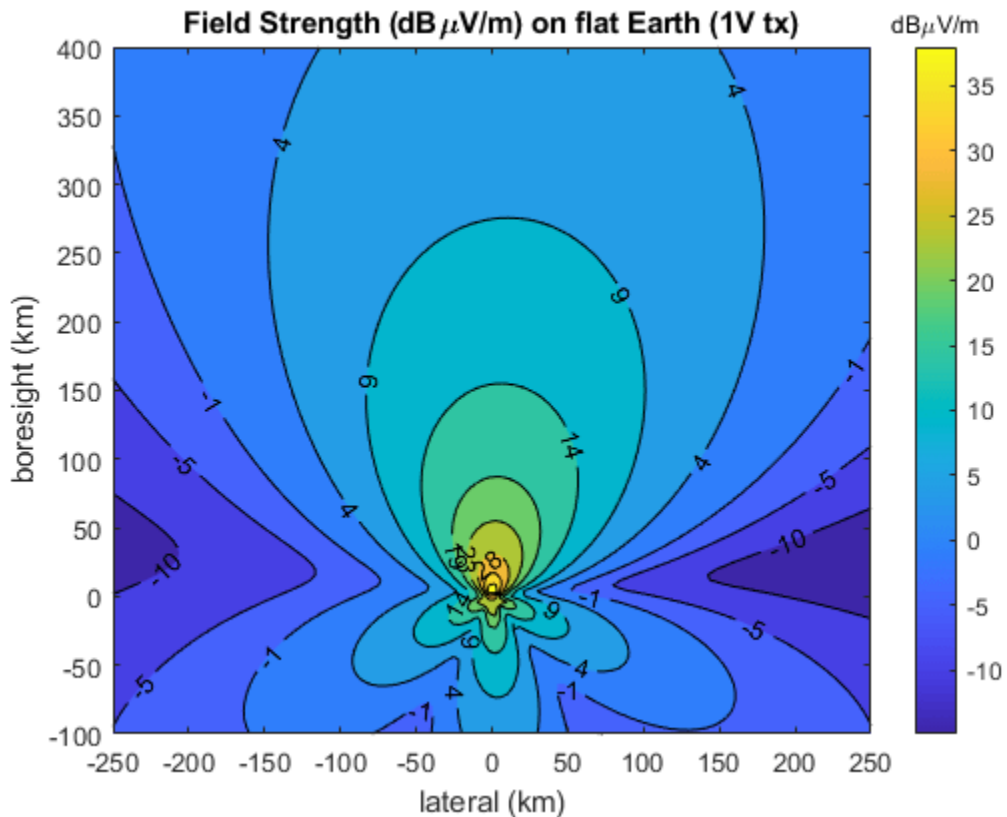
```
title('Field Strength (dB\muV/m) on flat Earth (1V tx)')
```

```
xlabel('lateral (km)')
```

```
ylabel('boresight (km)')
```

```
c = colorbar;
```

```
set(get(c,'title'),'string','dB\muV/m')
```



Define Transmitter Site with Antenna

Specify location and orientation of the antenna. Assign a location of 42 degrees north and 73 degrees west, along with antenna height of 15 meters. Orient the antenna toward the southwest at an angle of -150 degrees (or, equivalently, +210 degrees) measured counterclockwise from east.

```
lat = 42;
lon = -73;
h = 15;
az = -150;
```

Because the antenna points along the local Y-axis (not the X-axis), subtract 90 degrees to determine the rotation of the local system with respect to xEast-yNorth. (This is the angle to the local X-axis from xEast measured counterclockwise.)

```
xyrot = wrapTo180(az - 90);
```

Define a transmitter site with the antenna and orientation defined above.

```
tx = txsite('Name','Antenna Site', ...
    'Latitude',lat, ...
    'Longitude',lon, ...
    'Antenna',ant, ...
    'AntennaHeight',h, ...
    'AntennaAngle',xyrot, ...
    'TransmitterFrequency',f0);
```

Calculate Transmitter Power

Calculate root mean square value of power corresponding to antenna E-fields calculation and set as transmitter power.

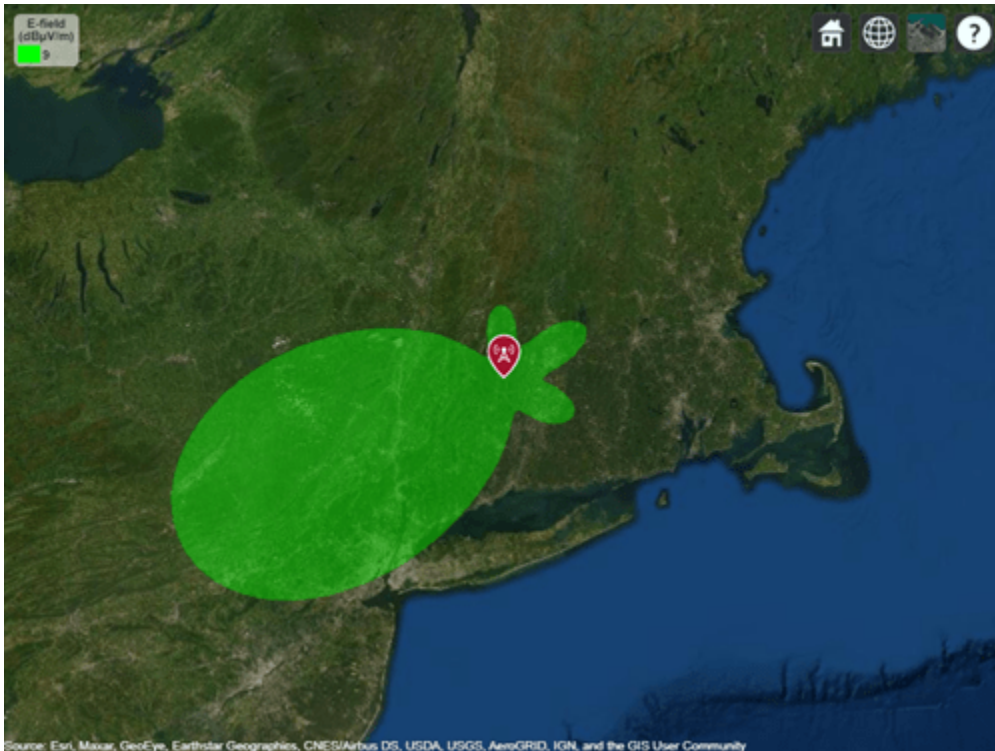
```
Z = impedance(tx.Antenna,tx.TransmitterFrequency);
If = feedCurrent(tx.Antenna,tx.TransmitterFrequency);
Irms = norm(If)/sqrt(2);
Ptx = real(Z)*(Irms)^2;
tx.TransmitterPower = Ptx;
```

Display Electric Field Coverage Map with Single Contour

Show transmitter site marker on a map and display coverage zone. Specify a signal strength of 9 dBuV/m to display a single electric field contour. Specify the propagation model as 'freespace' to model idealized wave propagation in free space, which disregards obstacles due to curvature of the Earth or terrain.

```
% Launch Site Viewer with no terrain
viewer = siteviewer("Terrain", "none");
```

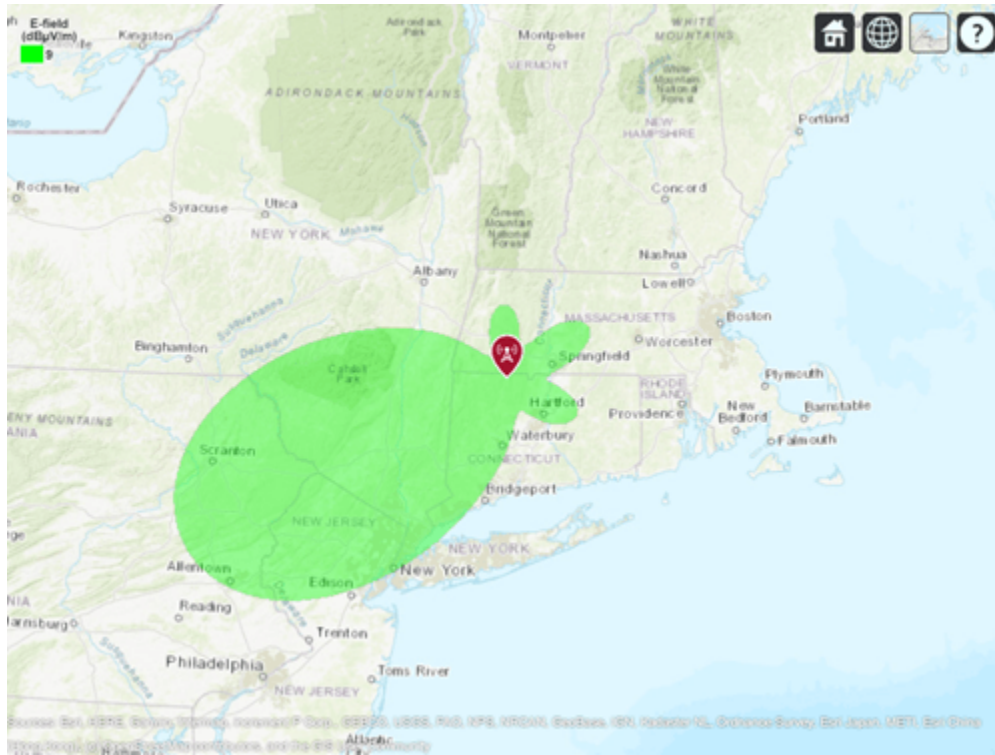
```
% Plot coverage
coverage(tx,'freespace', ...
        'Type','efield', ...
        'SignalStrengths',9)
```



Customize Site Viewer

Set the map imagery using the Basemap property. Alternatively, open the map imagery picker in Site Viewer by clicking the second button from the right. Select "Topographic" to see topography and labels on the map.

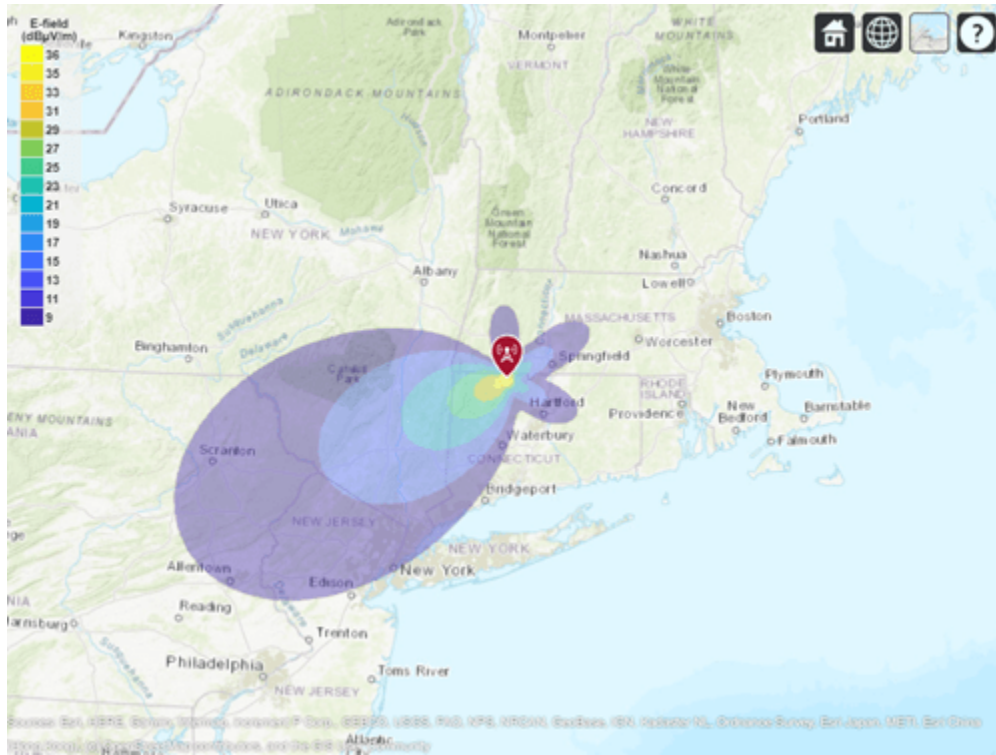
```
viewer.Basemap = 'topographic';
```



Display Electric Field Coverage Map with Multiple Contours

Specify multiple signal strengths to display multiple electric field contours.

```
sigStrengths = [9 14 19 24 29 36];
coverage(tx,' freespace', ...
    'Type','efield', ...
    'SignalStrengths',sigStrengths, ...
    'Colormap','parula', ...
    'ColorLimits',[9 36])
```

See Also

Functions

[EHfields](#) | [coverage](#)

Objects

[helix](#) | [siteviewer](#) | [txsite](#)

Related Examples

- “Visualize Antenna Coverage Map and Communication Links” on page 5-309
- “Read, Visualize and Write MSI Planet Antenna Files” on page 5-256

RFID Antenna Design

This example creates a commercially available RFID tag operating at 915 MHz. The planar antenna is drawn and meshed using PDE Toolbox™. The meshed structure is then imported into Antenna Toolbox and solved for its EM properties.

This example requires the following product:

- Partial Differential Equation Toolbox™

RFID Antenna

Below is a commercially available RFID antenna. These tags are typically used on boxes for inventory tracking. The main requirement for these tags is that they should be cheap to manufacture and have a very narrow band. The antenna below has an overall dimension of 22 mm x 22 mm.

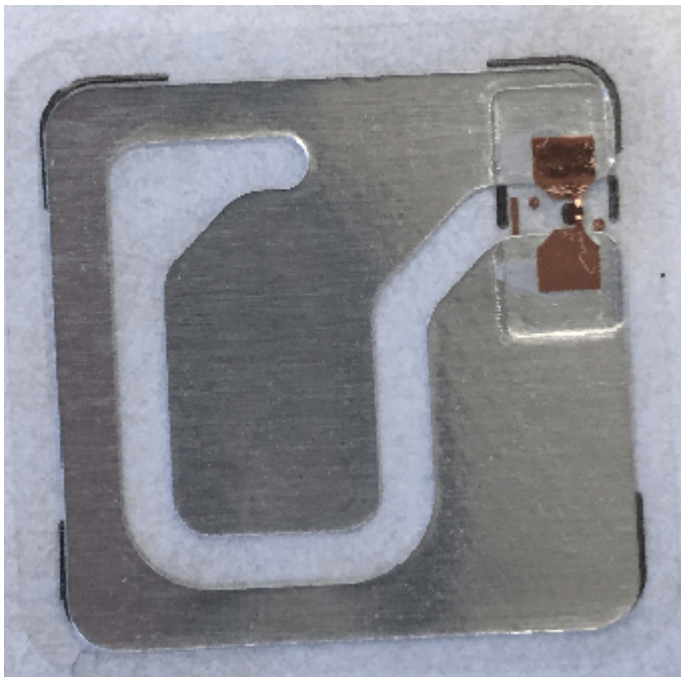


Fig.1: RFID antenna tag.

Drawing the Antenna Geometry

The antenna geometry can be drawn using `pdetool` in PDE Toolbox. The `pdetool` provides a graphical user interface to draw and mesh a 2D geometry. Call `pdetool` without arguments to start the application. The figure below is the snapshot of the `pdetool` with a basic layout of the antenna. Rectangle R1 is used to create the 22 mm square. Various slots in the structure are created using rectangles R2 to R5 as shown below. The rectangles R5 can be connected to R6 using a polygon shape. Similar idea can be used to connect R2 with R3.

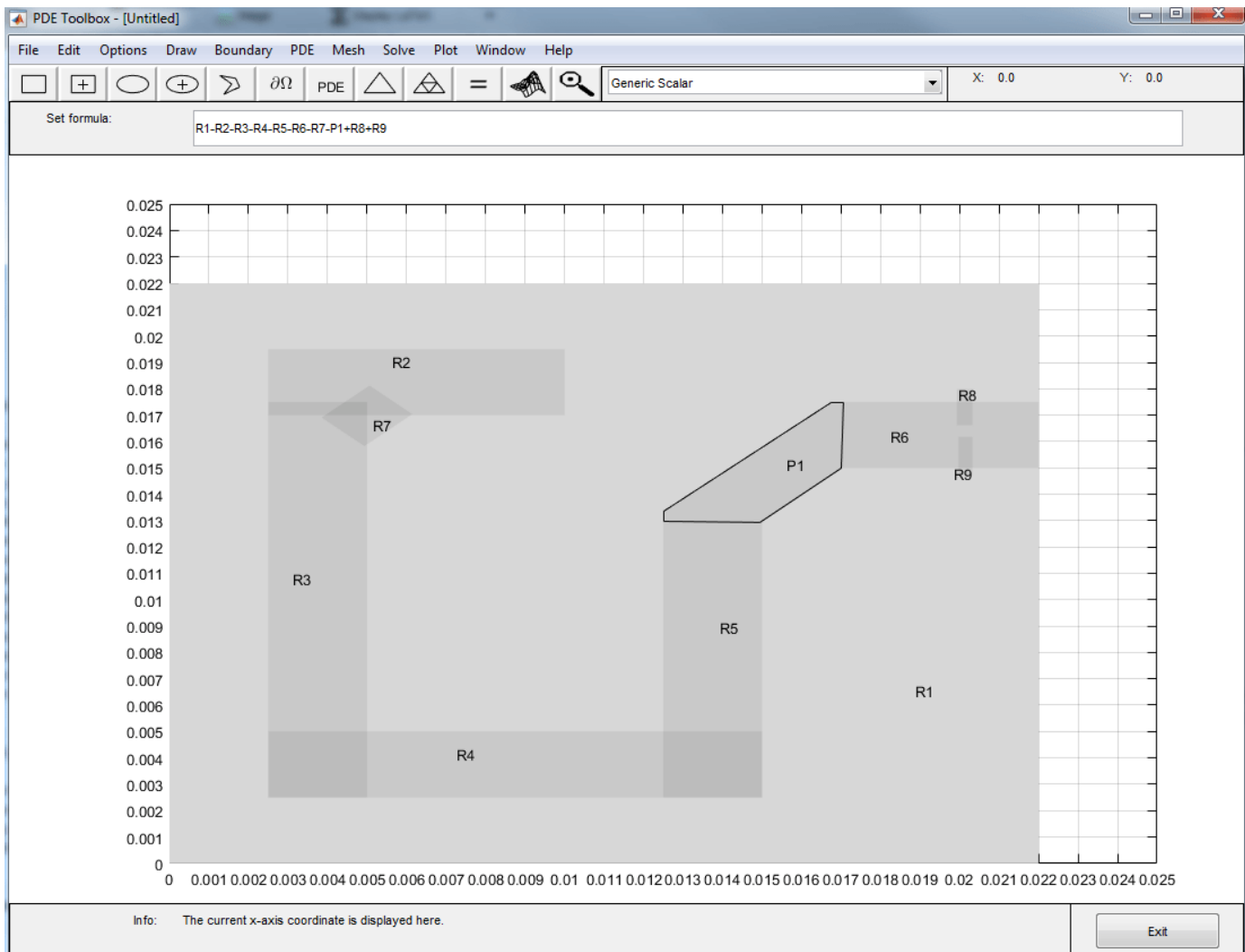


Fig.2. Geometry layout of the antenna tag.

Once the desired antenna geometry is set up, Boolean operations can be performed on the various shapes. In this example, the rectangles R2 to R7 need to be subtracted from the rectangle R1 to create the slots. The two rectangles R8 and R9 are added so that the antenna can be fed between them. The Boolean operation can be performed in the Set formula tab.

Meshing the Antenna

Once the Set formula tab is filled with the correct Boolean operation, press the triangle button, to mesh the structure. Maximum edge length in the mesh can be modified by setting the Max edge size value under Mesh->Parameters tab. The mesh below is generated with a maximum edge length of 2 mm.

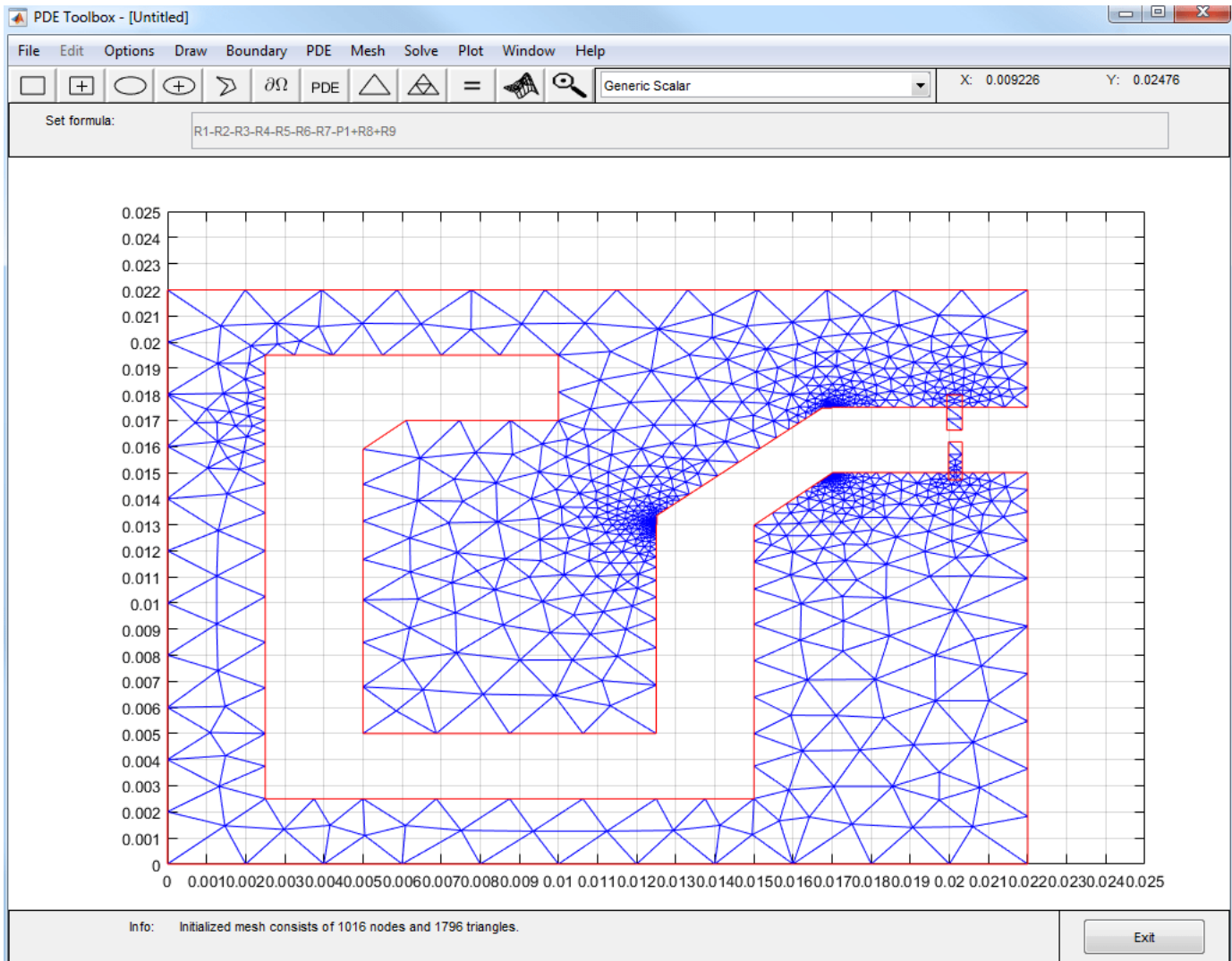


Fig.3. Meshed antenna tag

The generated mesh can be exported to the MATLAB workspace by selecting the Mesh-> Export Mesh tab. The p (Points), e (boundary edges) and t (triangles) are exported to the workspace. The data is saved in the file RFIDtag.mat. The complete antenna geometry is saved in the file RFIDtag.m.

Creating a Custom Antenna

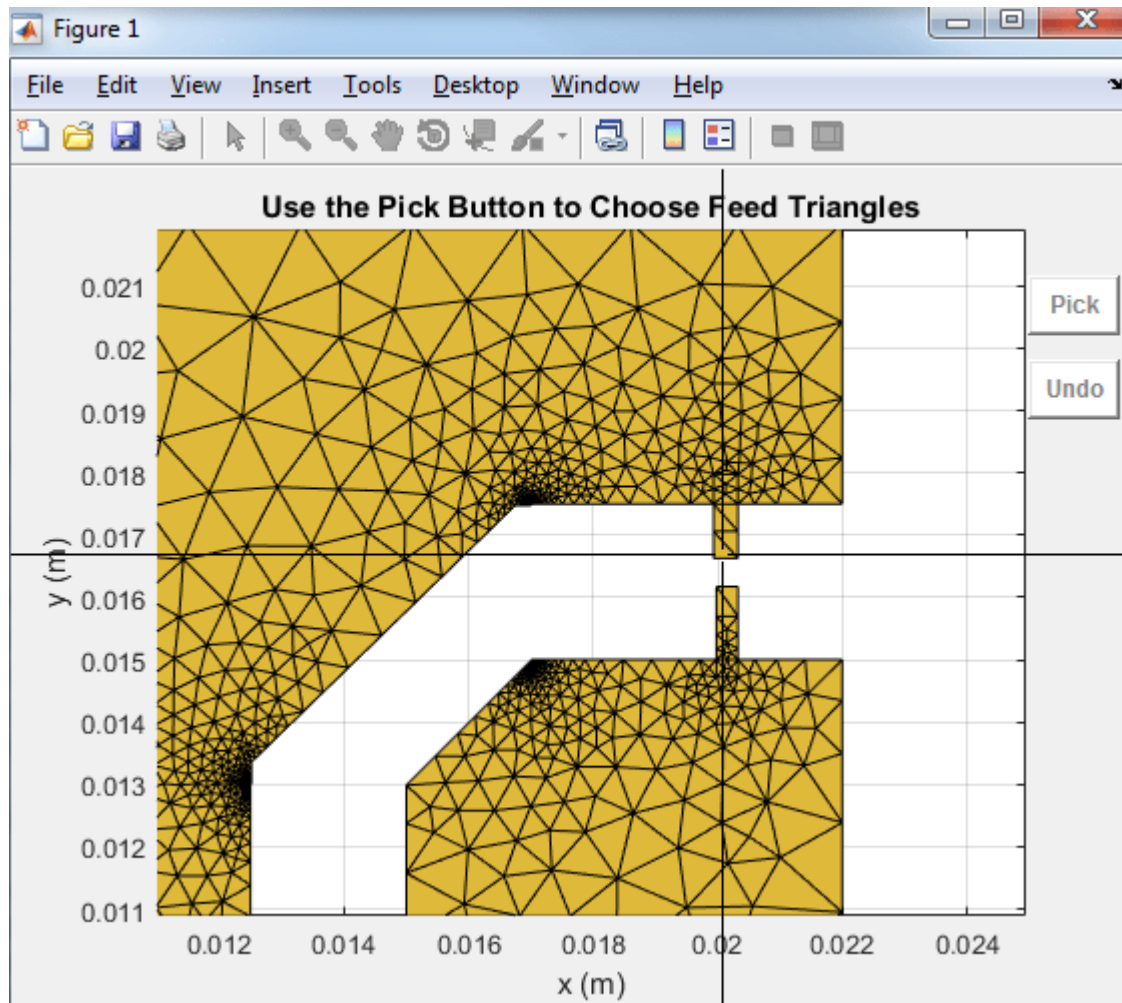
A custom planar mesh can be imported into Antenna Toolbox using the `customAntennaMesh` function. To convert this mesh to an antenna, a feed needs to be defined. This determines where the antenna is excited.

```
load RFIDtag
ant = customAntennaMesh(p, t);
```

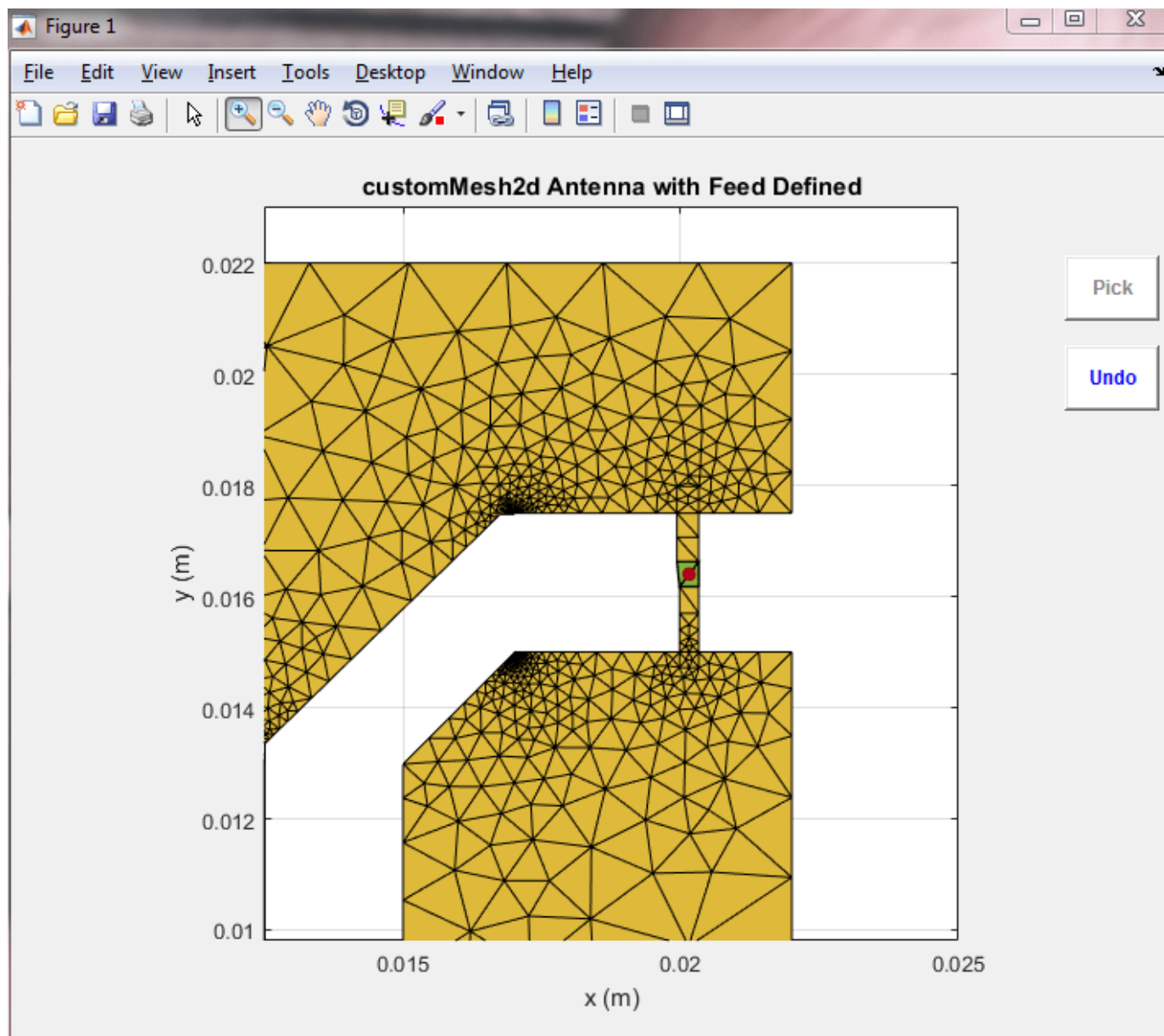
The feed can be created by using a GUI or by specifying the two points across which the feed is defined. We will look at both these approaches in this example. Calling `createFeed` with just the antenna as the input parameter opens the feed creating GUI.

createFeed(ant)

Pick a gap in the metal to feed the antenna. The interactive figure allows to select two triangles across the gap. Click the pick button which provides a cross hair to pick the two triangles. The figure below shows the way to select the first triangle. Bring the cross hair over the triangle as shown below and press the left mouse button.

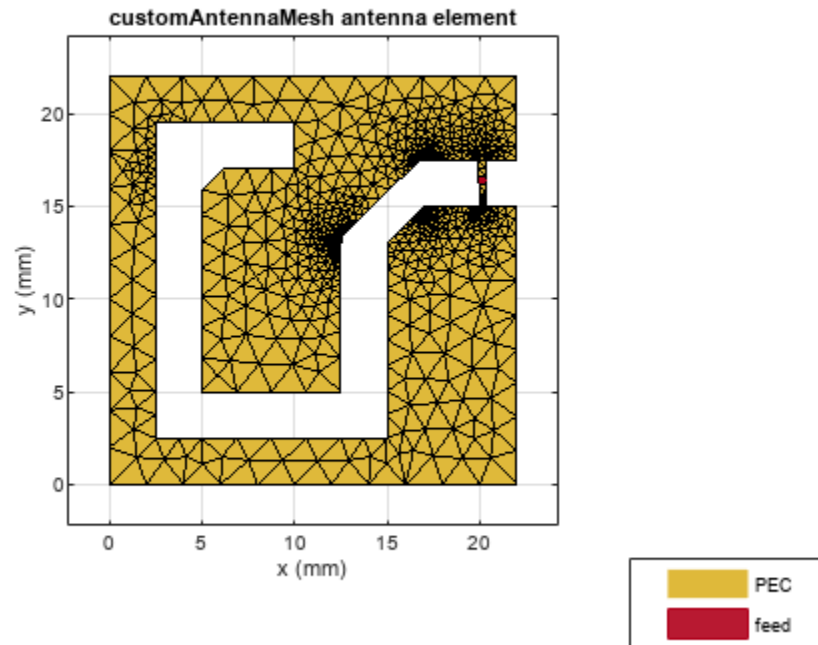


Repeat the process for the second triangle across the gap. Once the two feed triangles are selected the figure will update to show the feed with green triangles as shown below. The red dot shows the feeding edge. To change the feed location, click the undo button and start again.



The other way to create the feed is by specifying the two points between which the feed needs to be created as shown below. Use the show function to visualize the antenna.

```
createFeed(ant, [0.0201 0.0168], [0.0201, 0.0161]);  
show(ant);  
view(2);
```



Impedance of the RFID Antenna

The custom antenna can now be solved by calling all the analysis functions of the Antenna Toolbox. We are interested in the operation of the RFID tag at 915 MHz.

```
z = impedance(ant, 915e6)
```

```
z = 7.0565e-01 + 2.9162e+02i
```

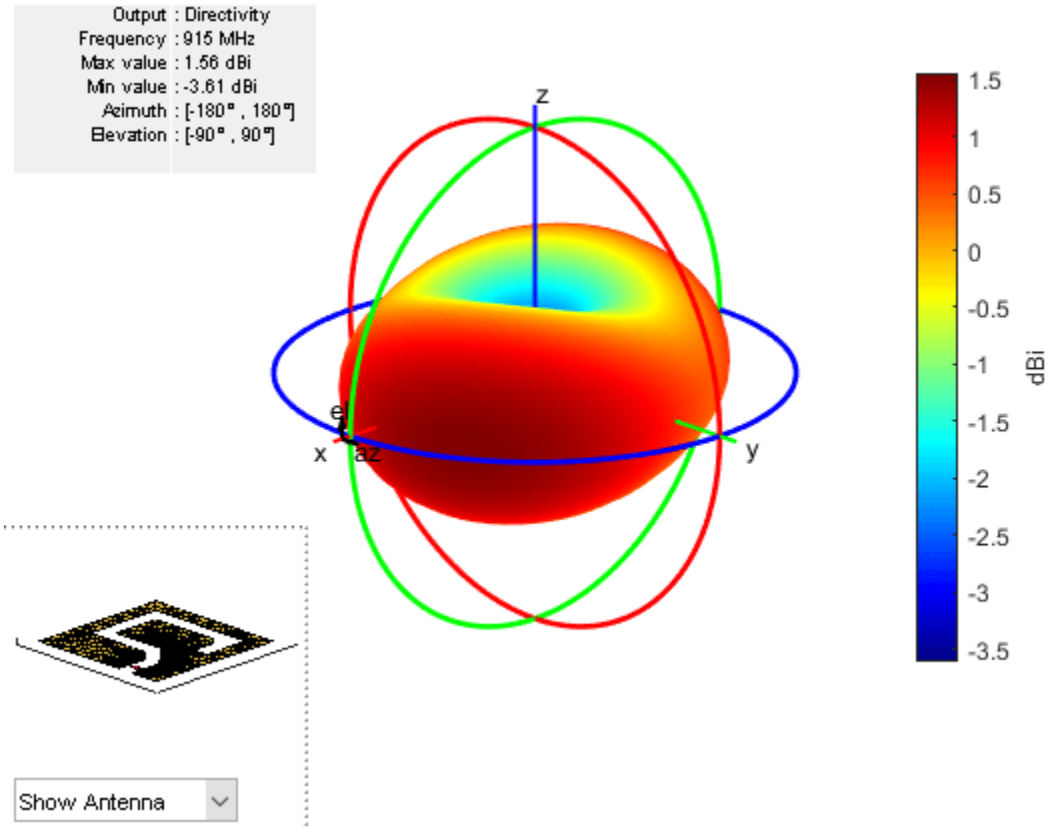
The impedance values above indicate that the antenna is highly inductive with a small resistance value. This is typical for RFID tags. A matching network at 915MHz need to be designed for successful operation of the tag.

More information on designing a matching network can be obtained in the following demo. "Impedance Matching of Non-resonant (Small) Monopole" on page 5-141;

Radiation Pattern of the RFID Antenna

The directivity of the RFID antenna is shown below. The tag has a null at the zenith but good coverage in the x-y plane and a maximum value of about 1.5dBi. This shows that the antenna is operational over a short distance from the reader, which is a requirement for an RFID tag.

```
pattern(ant, 915e6);
```



See Also

“Antenna Model Generation and Full-Wave Analysis From A Photo” on page 5-294

Wideband Blade Dipole Antenna and Array

This example shows the analysis of a blade dipole as a single element and in a 2 X 2 array with emphasis on its wideband behavior.

The Blade Dipole

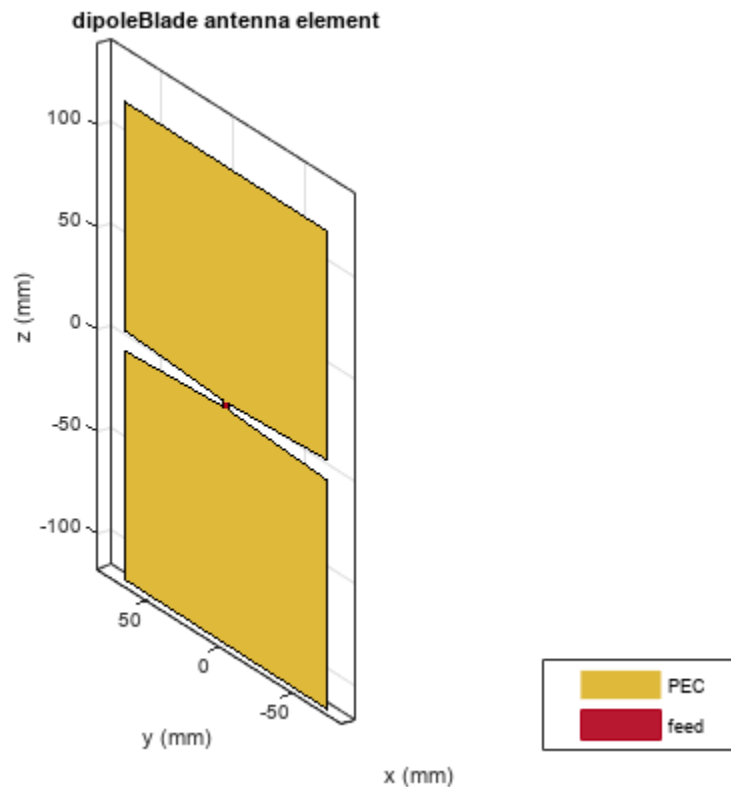
Strip dipoles or the cylindrical counterparts have a significant capacitive component to its impedance and typically possess a narrow impedance bandwidth. It is well known that widening the dipole or thickening the cylindrical cross-sections makes it easier to impedance match over a broader frequency range [1]. These wide dipoles, are known as *blade dipoles*. We will model such a blade dipole in this example [2].

```
L = 117e-3;      % Total half-length of blade dipole
W = 140e-3;      % Width of blade dipole
Ld = 112e-3;     % Half-length excluding impedance match taper
fw = 3e-3;       % Feed width
g = 3e-3;        % Feed gap
```

```
bladeDipole = dipoleBlade('Length', L, 'width', W);
```

The blade dipole consists of two identical metallic arms that include a tapered section close to the feed. The tapered section ensures a better impedance match to 50Ω .

```
figure
show(bladeDipole)
```



Calculate Reflection Coefficient of Blade Dipole

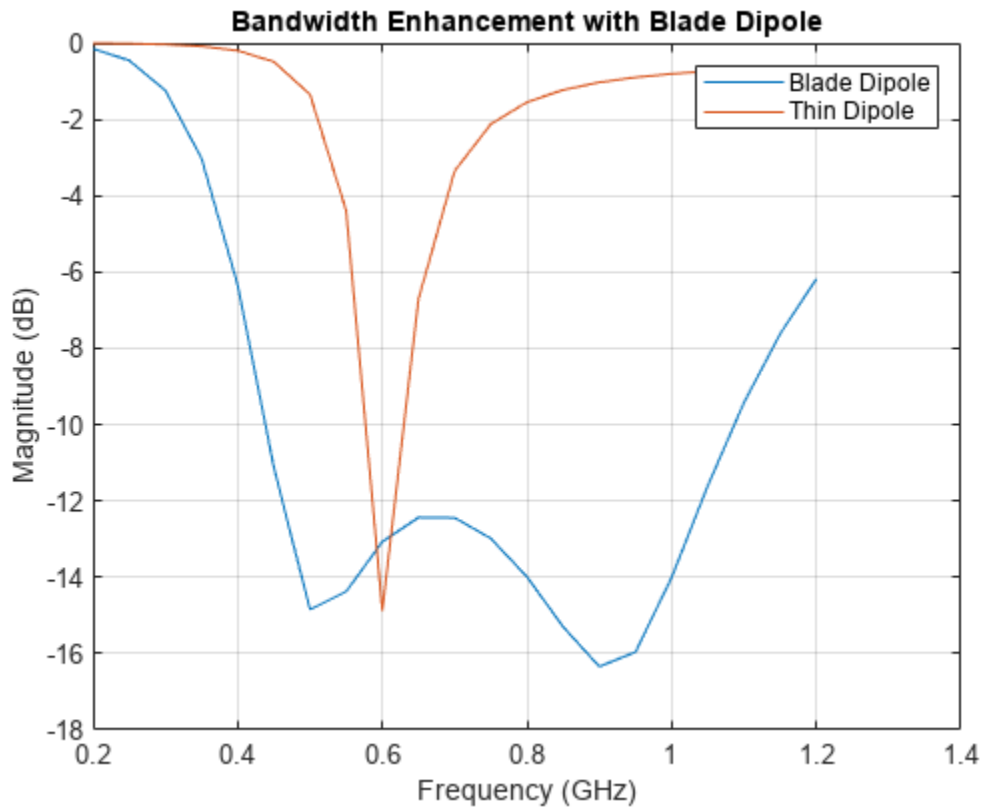
A good way to understand the quality of impedance matching, in this case to 50Ω , is to study the reflection coefficient of the antenna. We choose the frequency range of 200 MHz to 1.2 GHz and calculate the S-parameters of the antenna.

```
fmin = 0.2e9;  
fmax = 1.2e9;  
Nfreq = 21;  
freq = linspace(fmin,fmax,Nfreq);  
s_blade = sparameters(bladeDipole,freq);
```

Compare with Thin Dipole

To understand the difference between the performance of a blade dipole and of a traditional thin dipole, create a typical dipole from the Antenna Toolbox library with the same total length as the blade dipole but with the width being the same as the feedwidth, i.e. 3 mm. Calculate and plot the reflection coefficient of both dipoles.

```
thinDipole = dipole;  
thinDipole.Length = 2*L;  
thinDipole.Width = fw;  
s_thin = sparameters(thinDipole,freq);  
figure  
rfplot(s_blade,1,1);  
hold on  
rfplot(s_thin,1,1)  
legend('Blade Dipole','Thin Dipole')  
title('Bandwidth Enhancement with Blade Dipole')
```

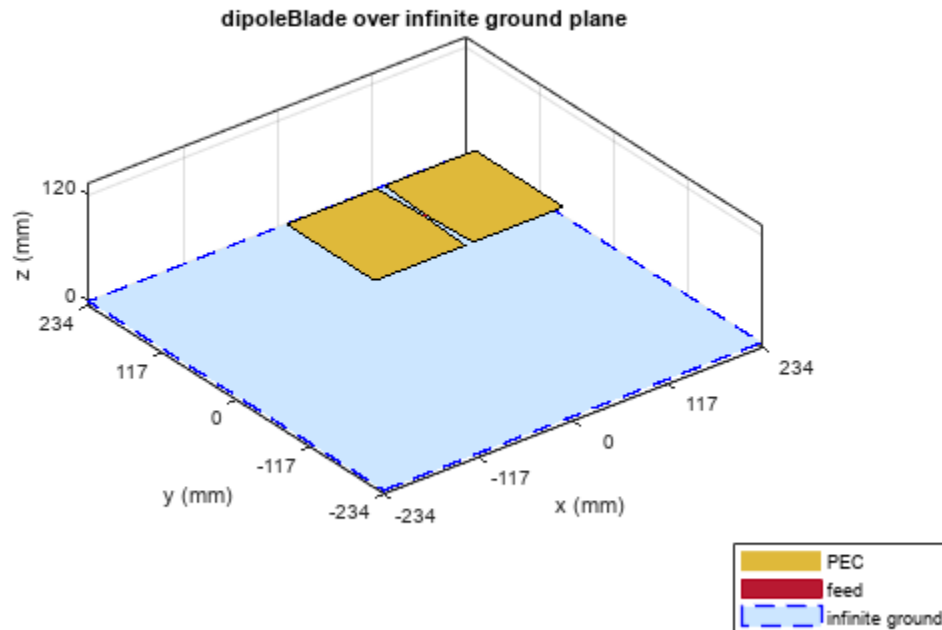


Build 2 X 2 Array of Blade Dipoles on Infinite GroundPlane

Such blade dipoles can be used as building blocks for modular arrays [2].

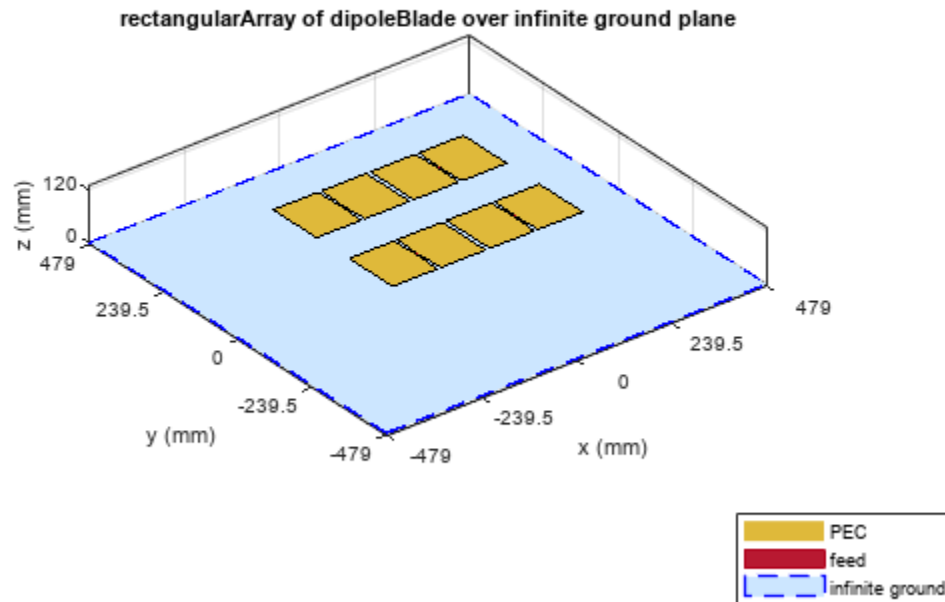
Create blade dipole backed by an infinite reflector Assign the blade dipole as a the exciter to a reflector of infinite extent. The dipole is positioned on the X-Y plane($z=0$).

```
bladeDipole.Tilt = 90;
bladeDipole.TiltAxis = [0 1 0];
ref = reflector;
ref.Exciter = bladeDipole;
ref.GroundPlaneLength = inf;
ref.GroundPlaneWidth = inf;
ref.Spacing = 120e-3;
figure
show(ref);
```



Create rectangular array with the reflector backed dipole Make a 2 X 2 rectangular array with the reflector backed blade dipole. The spacings between elements are chosen for the center of the band. This might introduce grating lobes since the spacing becomes approximately 0.81λ at 1 GHz. However, for non-scanning applications this larger spacing should not cause problems.

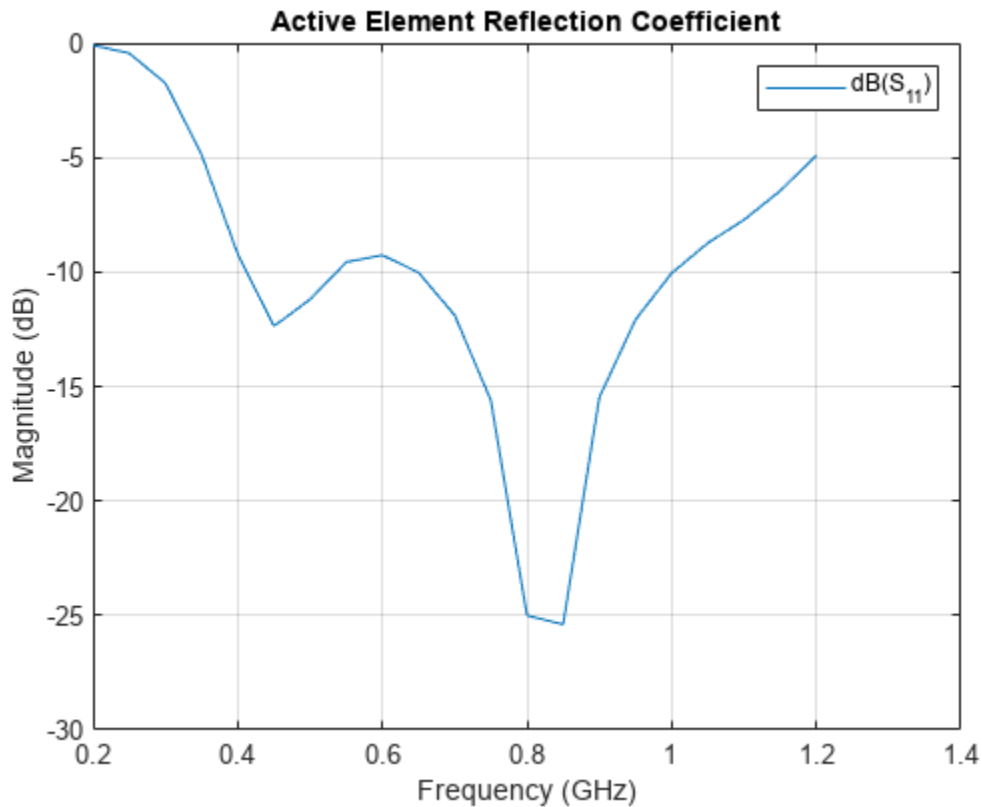
```
ref_array = rectangularArray;
ref_array.Element = ref;
ref_array.RowSpacing = 245e-3;
ref_array.ColumnSpacing = 245e-3;
figure
show(ref_array);
```



Calculate Active Reflection Coefficient

For a small 2 X 2 array, each element experiences the same environment. Therefore the active reflection coefficient of any of the elements should suffice to understand the matching performance.

```
s_array = sparameters(ref_array, freq);  
figure  
rfplot(s_array, 1, 1)  
title('Active Element Reflection Coefficient')
```



The blade dipole in a 2 X 2 array continues to show broadband performance. The bandwidth is approximately 2.6:1, measured as the ratio of the highest to the lowest frequency at which the reflection coefficient curve crosses -10 dB.

References

- [1] C. A. Balanis, *Antenna Theory. Analysis and Design*, Wiley, New York, 3rd Edition, 2005.
- [2] V. Iyer, S. Makarov, F. Nekoogar, "On wideband modular design of small arrays of planar dipoles," *IEEE Antennas and Propagation Society International Symposium (APSURSI)*, pp.1-4, 11-17 July 2010.

See Also

"Crossed-Dipole (Turnstile) Antenna and Array" on page 5-218

Analysis of Inset-Feed Patch Antenna on Dielectric Substrate

This example showcases the analysis of an inset-feed patch antenna on a low-epsilon, low-loss, thin dielectric substrate. The results are compared with the reflection coefficient and surface currents around the 2.4 GHz, Wi-Fi band for a reference design [1]. The Antenna Toolbox™ library of antenna elements includes a patch antenna model that is driven with a coaxial probe. Another way to excite the patch is by using an inset-feed. The inset-feed is a simple way to excite the patch and allows for planar feeding techniques such as a microstrip line.

Inset-Feed Patch Antenna

The inset-feed patch antenna typically comprises of a notch that is cut from the non-radiating edge of the patch to allow for the planar feeding mechanism. Typical feeding mechanism involves a microstrip line coplanar with the patch. The notch size, i.e. the length and the width are calculated to achieve an impedance match at the operating frequency. A common analytical expression that is used to determine the inset feed position (x_0, y_0), the distance from the edge of the patch along its length, is shown below [2]. The patch length and width are L, W respectively.

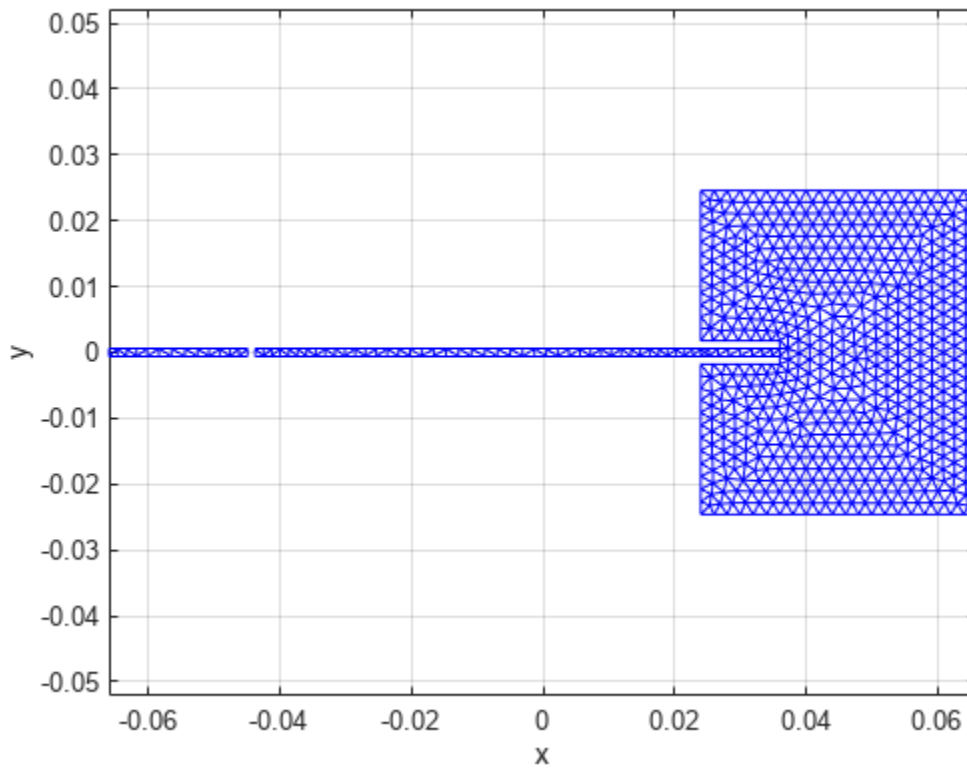
$$R = R_{edge} \cos^4(\pi x_0 / L)$$

It is also common to excite the patch along the center line ($y = W/2$), where W is the width of the patch antenna, which makes the y -coordinate zero. This is the reason, the expression shown above, is only in terms of the x -coordinate.

Create Geometry and Mesh

The PDE toolbox™ has been used to create the geometry and mesh the structure. Use the `triangulation` function in MATLAB™ to visualize the mesh.

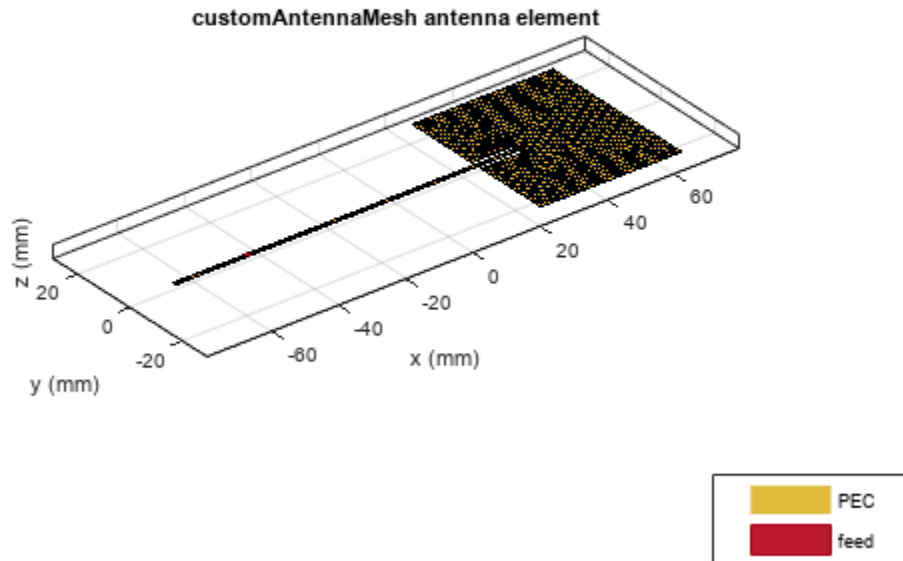
```
load insetfeedpatchmesh
T = triangulation(t(1:3,:), 'p');
figure
triplot(T)
axis equal
grid on
xlabel('x')
ylabel('y')
```



Create Patch Antenna

Use the `customAntennaMesh` from Antenna Toolbox™ to transform this mesh into an antenna. Use the `createFeed` function to define the feed.

```
c = customAntennaMesh(p,t);  
createFeed(c,[-0.045 0 0],[-0.0436 0 0])  
show(c)
```

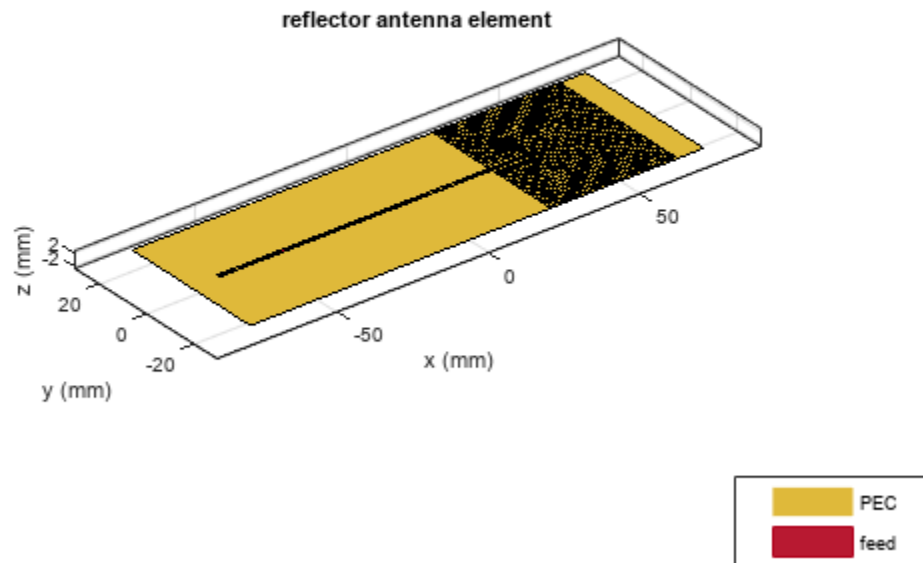



- A note about the feeding point on the microstrip line* The feed point chosen is such that it is $\lambda_g/4$ from the open circuit end of the line [3]. In addition a relatively long line length is chosen to facilitate an accurate calculation of the s-parameters. The wavelength in the dielectric is approximated relative to the wavelength in free-space as

$$\lambda_g \approx \lambda_0 / \sqrt{\epsilon_r}$$

Provide groundplane backing to radiator Patch antennas are backed by a ground plane. Do this by assigning the customAntennaMesh as an exciter for a reflector.

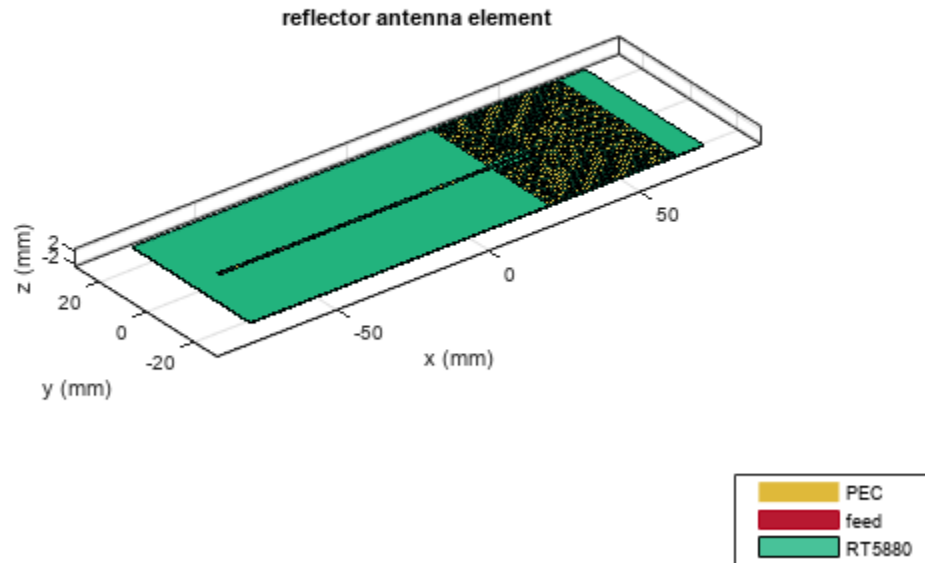
```
r = reflector;
r.Exciter = c;
r.GroundPlaneLength = 15e-2;
r.GroundPlaneWidth = 5e-2;
r.Spacing = 0.381e-3;
show(r)
```



Assign Substrate to Patch Antenna

Define a dielectric material with $\epsilon_r = 2.2$ and loss tangent of 0.0009. Assign this dielectric material to the Substrate property in the reflector.

```
d = dielectric;  
d.Name = 'RT5880';  
d.EpsilonR = 2.2;  
d.LossTangent = 0.0009;  
r.Substrate = d;  
show(r)
```



Analysis of Patch Antenna

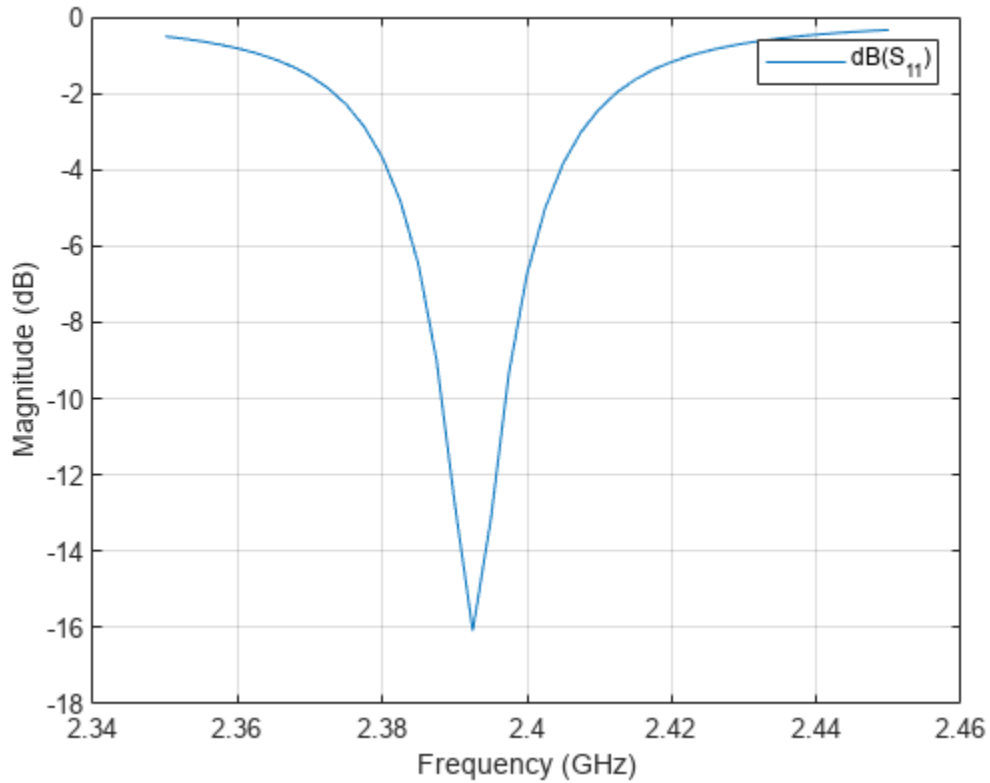
The overall dimensions of this patch antenna are large and therefore will result in a relatively large mesh (dielectric + metal). The structure is analyzed by meshing it with a maximum edge length of 4 mm and solved for the scattering parameters. The maximum edge length was chosen to be slightly lesser than the default maximum edge length computed at the highest frequency in the analysis range of 2.45 GHz, which is about 4.7 mm. The analyzed antennas are loaded from a MAT file to the workspace.

load `insetfeedpatch`

Calculate S-Parameters of Patch Antenna

The center frequency of analysis is approximately 2.4 GHz. Define the frequency range

```
freq = linspace(2.35e9,2.45e9,41);
s = sparameters(r,freq,50);
figure
rfplot(s,1,1)
```

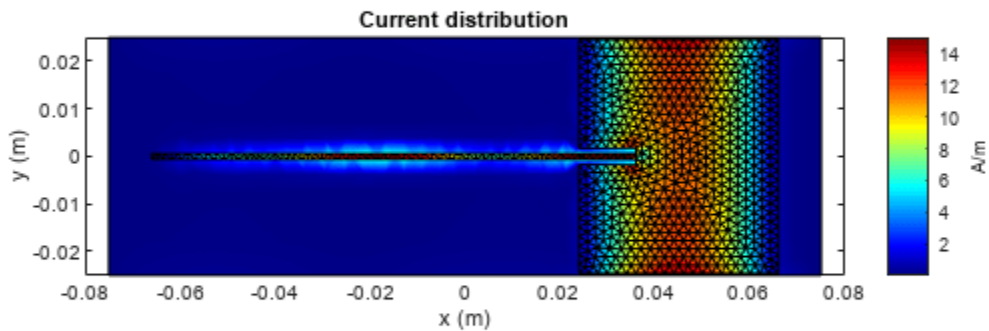


The plot shows the expected dip in the reflection coefficient close to 2.4 GHz. The default reference impedance is set to 50Ω . Use the -10 dB criterion to determine the reflection coefficient bandwidth for this patch to be less than 1%.

Visualize Current Distribution

Use the function `current`, to plot the surface current distribution for this patch antenna at approximately 2.4 GHz. The current is minimum at the edges along its length and maximum in the middle.

```
figure
current(r,2.4e9)
view(0,90)
```



Discussion

The results for this patch antenna are in good agreement with the reference results reported in [1], pg.111 - 114.

Reference

- [1] Jagath Kumara Halpe Gamage, "Efficient Space Domain Method of Moments for Large Arbitrary Scatterers in Planar Stratified Media", Dept. of Electronics and Telecommunications, Norwegian University of Science and Technology.
- [2] Lorena I. Basilio, M. A. Khayat, J. T. Williams, S. A. Long, "The dependence of the input impedance on feed position of probe and microstrip line-fed patch antennas," IEEE Transactions on Antennas and Propagation, vol. 49, no. 1, pp.45-47, Jan 2001.
- [3] P. B. Katehi and N. G. Alexopoulos, "Frequency-dependent characteristics of microstrip discontinuities in millimeter wave integrated circuits," IEEE Transactions on Microwave Theory and Techniques, vol. 33, no. 10, pp. 1029-1035, 1985.

See Also

"Modeling and Analysis of Single Layer Multi-band U-Slot Patch Antenna" on page 5-317

Read, Visualize and Write MSI Planet Antenna Files

This example shows how to read a MSI Planet antenna file (.MSI or .PLN). You can read a MSI file using the `msiread` function and visualize the data using the `polarpattern` function. You can also write the data back into the MSI Planet format using the `msiwrite` function.

Read MSI Planet Antenna File

Read the .pln file

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

The MSI Planet file reader returns 3 structures. Horizontal and Vertical structures contains the 7 fields: 1) PhysicalQuantity: Describes the nature of the data in the file. Typically, MSI Planet files contains antenna gain data. 2) Magnitude: Gain data 3) Units: Units of the data 4) Azimuth: Azimuthal angle(s) in degrees at which the data was measured. 5) Elevation: Elevation angle(s) in degrees at which the data was measured. 6) Frequency: Frequency of operation in Hertz. 7) Slice: Cut plane of the data measured. This plane can be 'Azimuth' or 'Elevation'.

Optional structure contains all the other fields from the file such as: name, make, frequency, h_width, v_width, front_to_back, gain, tilt, polarization, comment. There can also be other fields in the file.

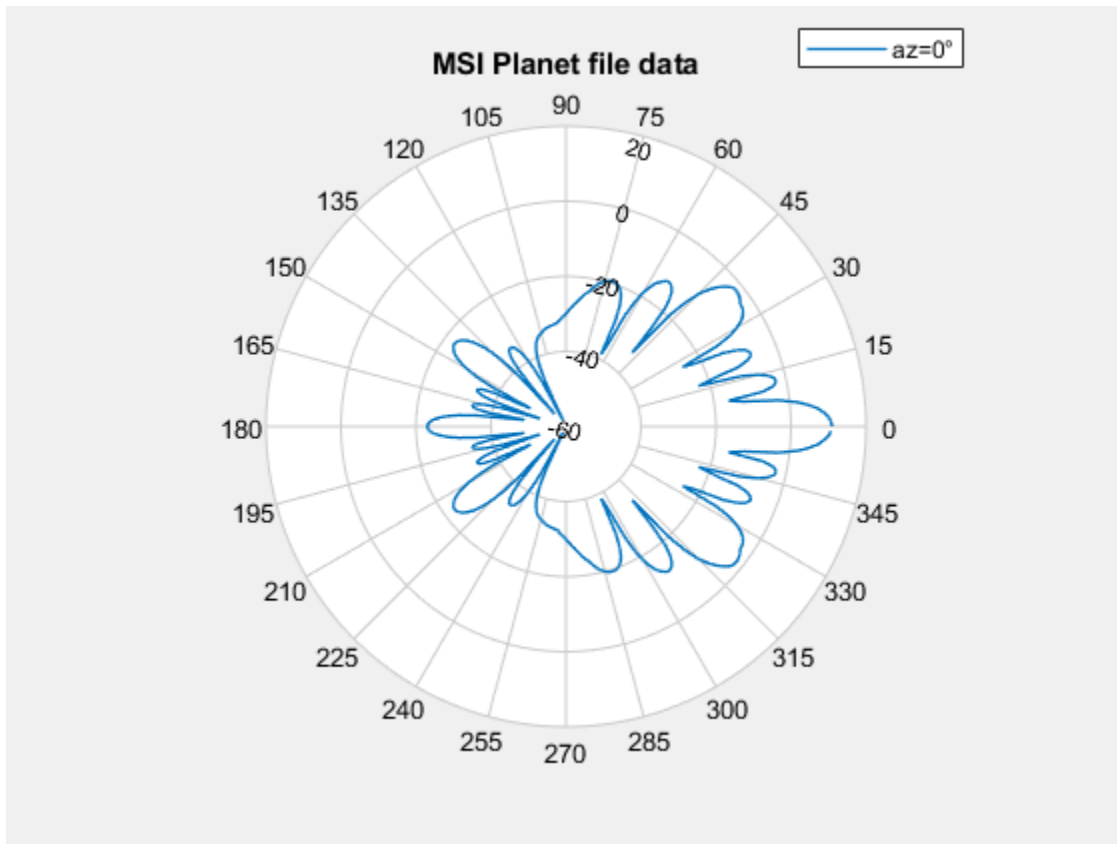
The horizontal and vertical gain data in the MSI Planet file is specified as loss data and is relative to the gain specified in the file. This data is converted to gain data. After reading the file, the relativity to gain is removed in the horizontal and vertical structures.

The front_to_back ratio in the MSI Planet file is calculated either from the horizontal gain data or vertical gain data. The results can reflect one of these possibilities: 1) Ratio between the peak value and the value in the opposite hemisphere (180 degrees from the peak value angle). 2) Ratio between the peak value and the worst-case value in the 60-degree sector ((peak angle + 180) +/- 30 degrees). 2) Ratio between the peak value and the worst-case value in the 40-degree sector ((peak angle + 180) +/- 20 degrees).

Visualize Data from MSI Planet Antenna File

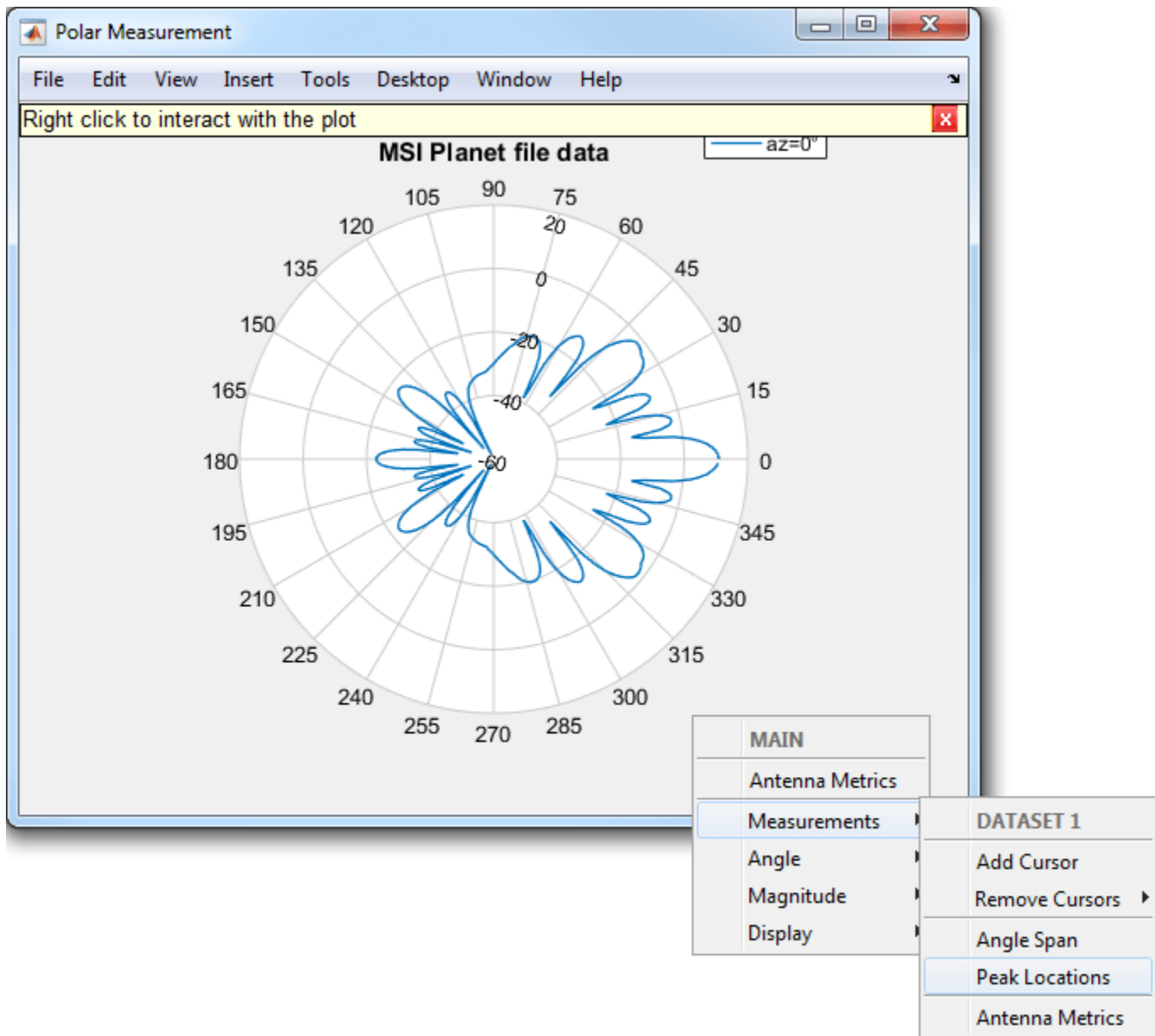
Visualize the vertical gain data in an interactive 2D polar plot. Right click in the figure window to interact with the plot.

```
P = polarpattern(Vertical.Elevation, Vertical.Magnitude);  
P.TitleTop = 'MSI Planet file data';  
createLabels(P, 'az=0#deg');
```

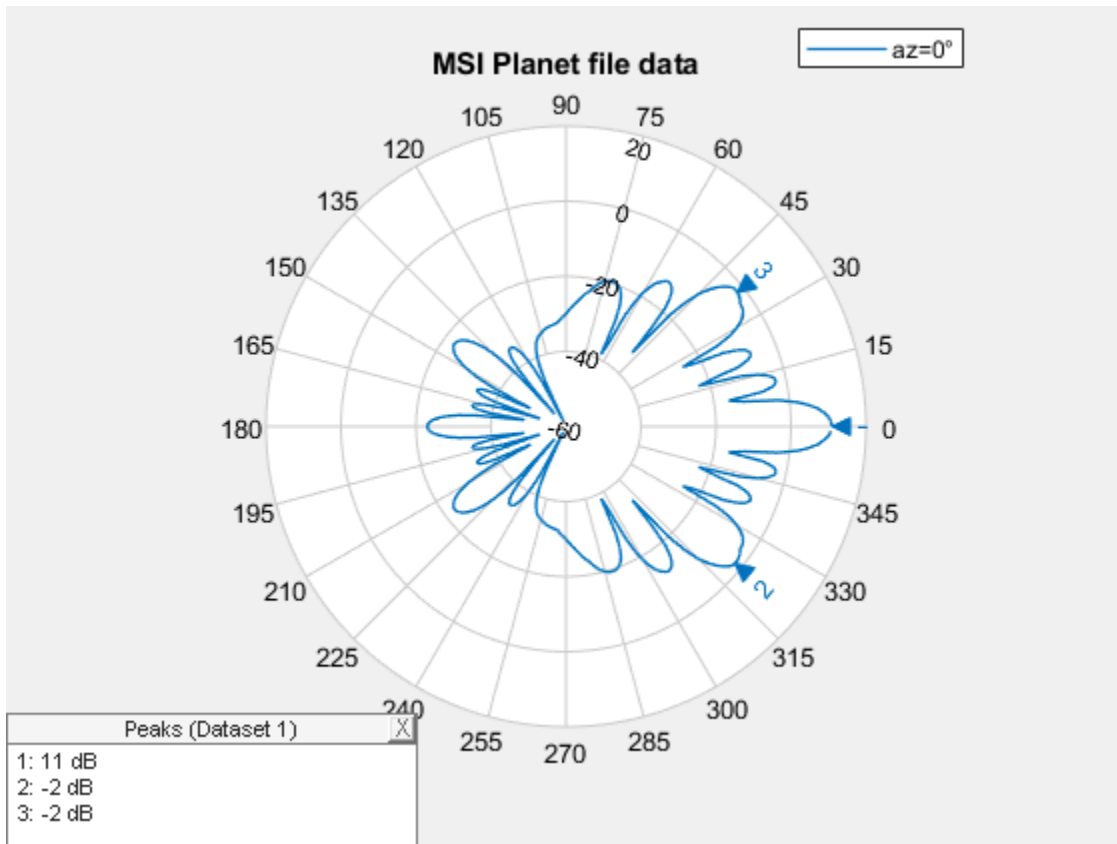


Enabling Peaks, Cursors, and Angle Span

Peaks and cursors can be enabled using the interactive plot or the command line. To turn on peaks, right click on the plot. Click on the **Measurements** tab and then check **Peak Locations** from the context menu. To control the number of peaks using the plot, right click on any of the peak triangles and use the **Num Peaks** from the context menu.

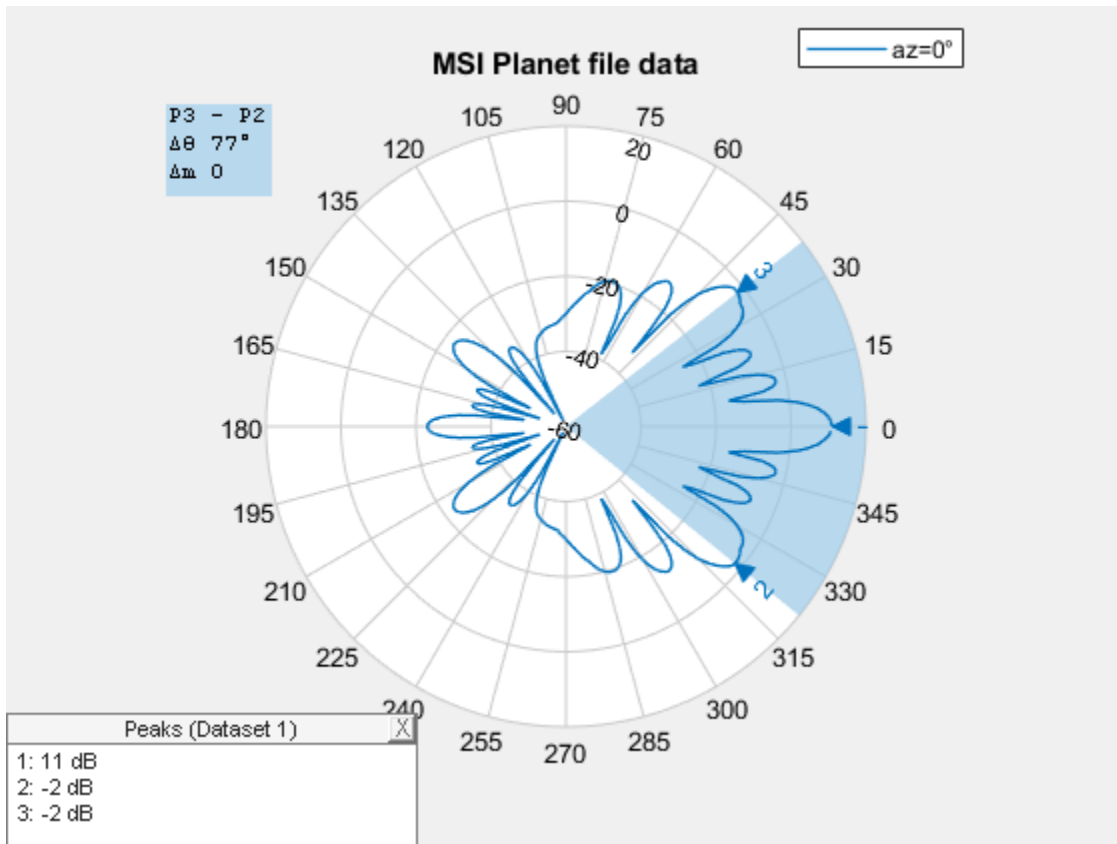


```
% The peaks can also be turned on using the command line.
P.Peaks = 3;
```

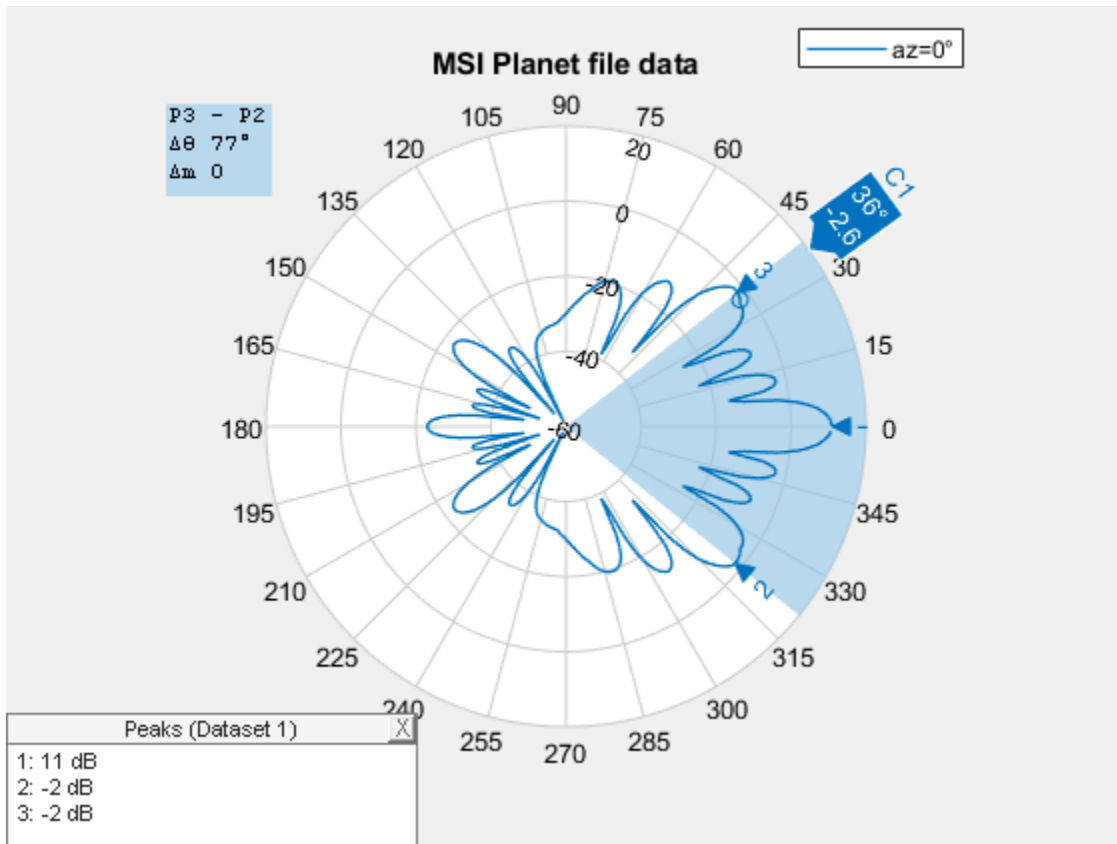
To turn on the angle span, right click on the plot. Click on Measurements tab and then check Angle Span context menu. To turn on angle span between peaks 2 and 3 the user can use the following function.

```
showSpan(P, 'P2', 'P3', 1);
```



To add a cursor, double click on the plot at the intended location. The cursor can be moved using the mouse. A cursor can be added at 36 degrees using the following function

```
addCursor(P,36,1);
```

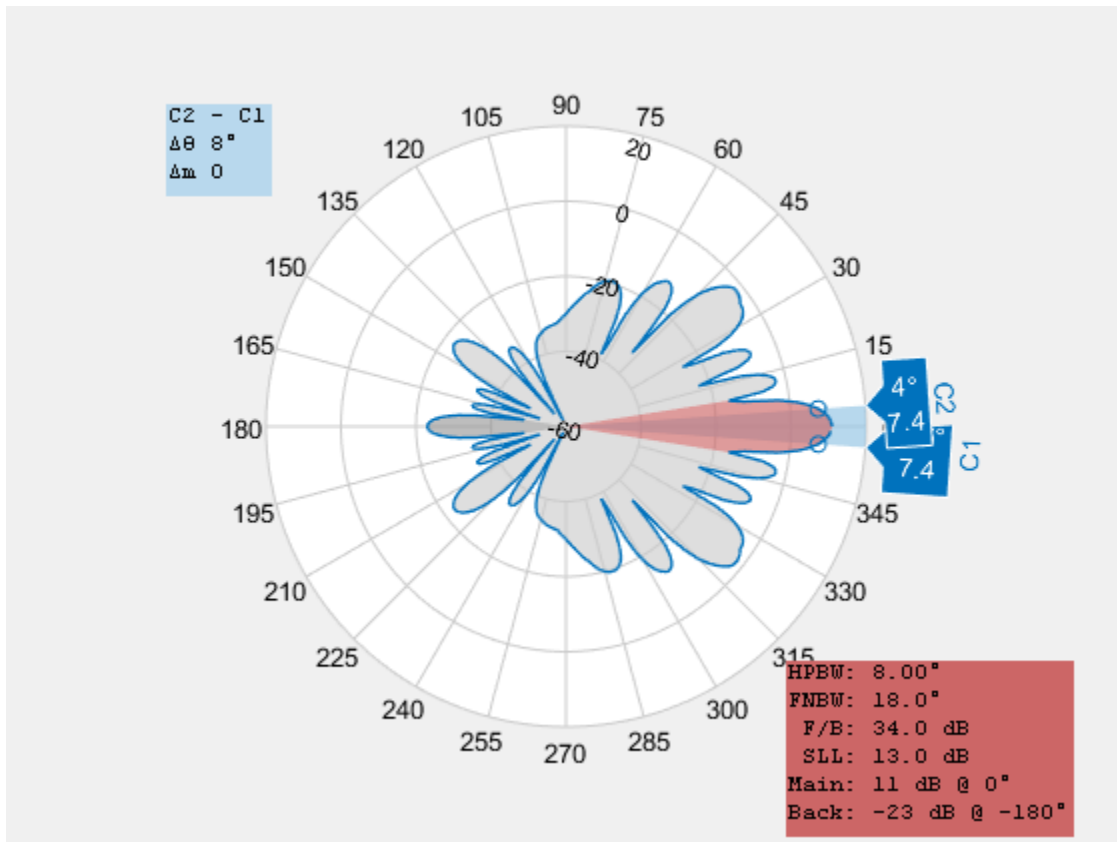


Turning on Antenna Metrics for Lobes Visualization and Measurements

Turning on antenna metrics displays the main, back and side lobes on the plot. It also gives a text box with antenna measurements: 1) Half-power Beamwidth 2) First-Null Beamwidth 3) Front-to-Back ratio 4) Side-lobe level 5) Main lobe peak value and corresponding angle 6) Back lobe peak value and corresponding angle.

Some of these measurements like the Half-power Beamwidth and the Front-to-Back ratio are available in the MSI Planet files. These results can be compared and verified between the files and the plot.

```
P = polarpattern(Vertical.Elevation, Vertical.Magnitude);
P.AntennaMetrics = 1; % Turning on antenna metrics on the plot
```



Write Data in MSI Planet File Format

Call the `msiwrite` function to take the 3 output structures from the `msiread` function and write the data in the MSI Planet file format.

```
msiwrite('Test_file_demo_write.pln',Horizontal,Vertical,Optional);
```

The first three inputs to `msiwrite` function are mandatory. These are the filename, the horizontal data structure and the vertical data structure. The user can also provide an optional structure as the 4th argument. Horizontal and Vertical data structures should contain the 7 fields discussed before.

The gain data is made relative to the gain value and is converted to dB loss when it is written to the file.

`msiwrite` functionality can also accept an antenna or array object as an input. Other inputs are frequency, filename and two name-value pairs: 'Name' and 'Comment'. Using this functionality you can first design the antenna. You can then input the antenna object along with the desired frequency and filename into the MSI Planet file format.

```
h = helix; % Creating helix in default configuration
msiwrite(h,2e9,'test_helix','Name','Test_Antenna','Comment', ...
        'Designed Helix Antenna in MATLAB');
```

See Also

“Custom Radiation Pattern and Fields” on page 5-263

Custom Radiation Pattern and Fields

This example shows how to visualize a radiation pattern and vector fields from user data. To plot 3D field data, use the `patternCustom` function. This function also allows the user to slice the data and see it. To visualize just 2D polar data use the `polarpattern` function. The `polarpattern` function allows you to interact with the data as well as perform antenna specific measurements. The user can also plot the vector fields at a point in space using the `fieldsCustom` function.

Import 3D Pattern Data

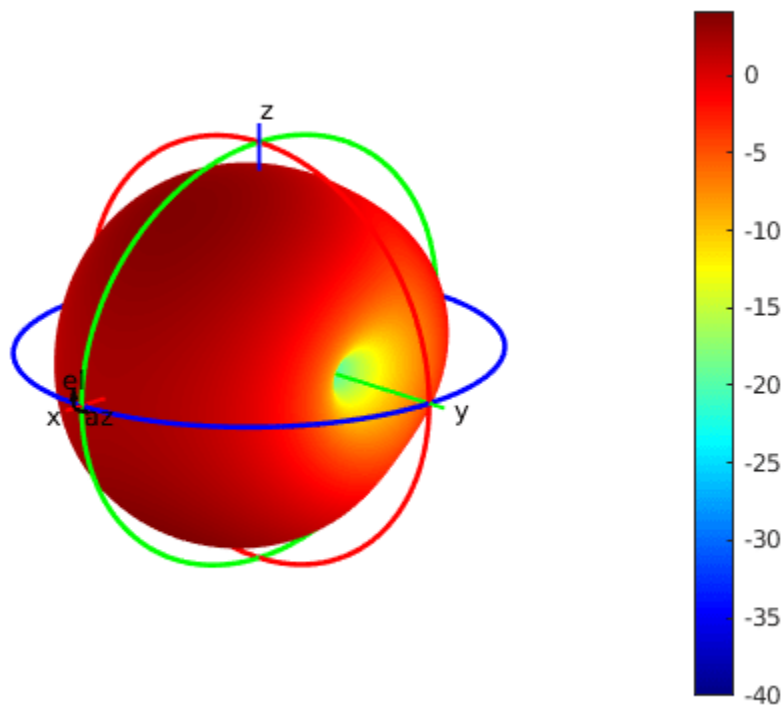
3D Radiation pattern data stored in a csv file format can be read using the `csvread` function. In the first part of this example we use the `patternCustom` function to visualize the 3D data. The function can be used to visualize 2D slices of the 3D data as well.

```
M = csvread('CustomPattern_testfile.csv',1,0);
```

Plot 3D Radiation Pattern on Polar Coordinate System

To plot the 3D radiation pattern on a polar coordinate system, specify the `MagE` vector/matrix and `theta` and `phi` vectors. If `MagE` is a matrix, it should be of size `phi x theta`. If `MagE` is a vector, all 3 arguments `MagE`, `phi` and `theta` should be of the same size.

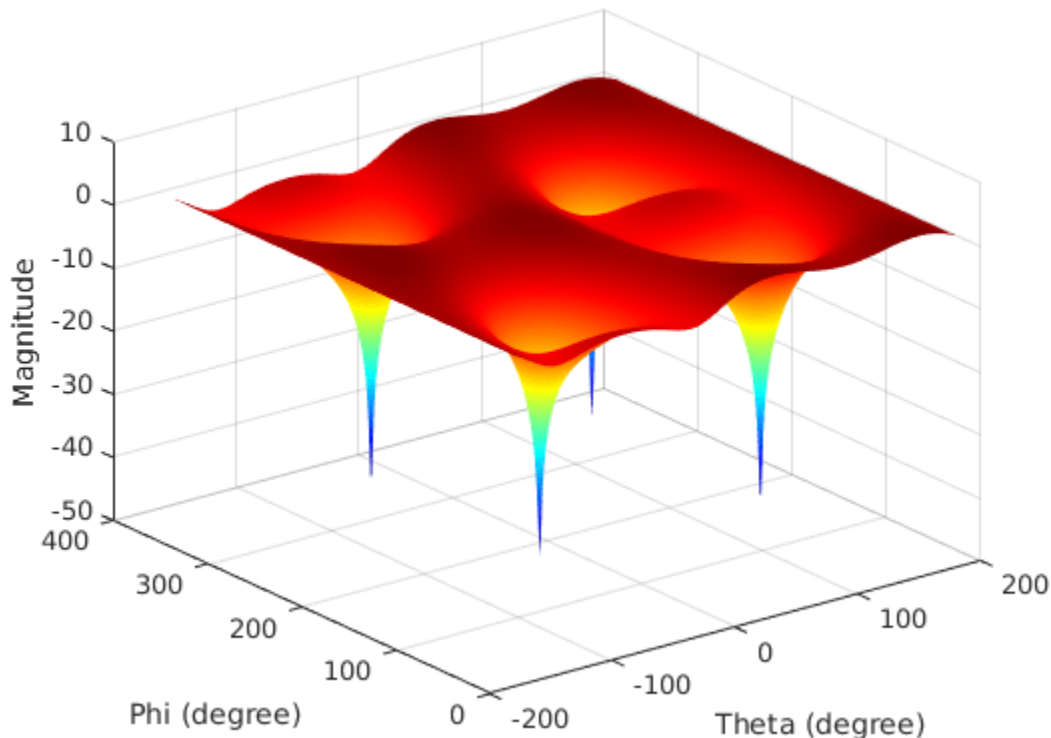
```
patternCustom(M(:,3),M(:,2),M(:,1));
```



Plot 3D Radiation Pattern on Rectangular Coordinate System

To plot the 3D radiation pattern on a rectangular coordinate system, you modify the `CoordinateSystem` flag. By default, the flag is set to polar. Change it to rectangular to visualize the data in the rectangular coordinate system.

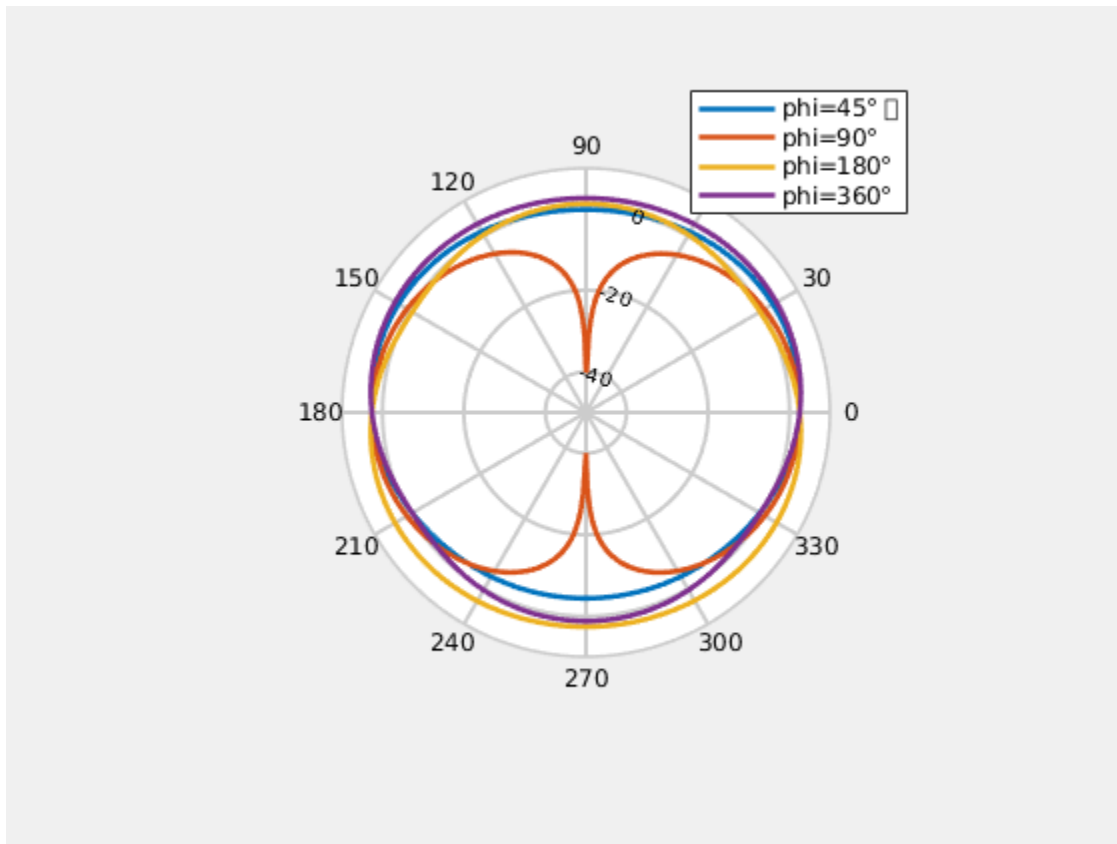
```
patternCustom(M(:,3),M(:,2),M(:,1),'CoordinateSystem','rectangular');
```



Visualize 2D Slices from the 3D Data

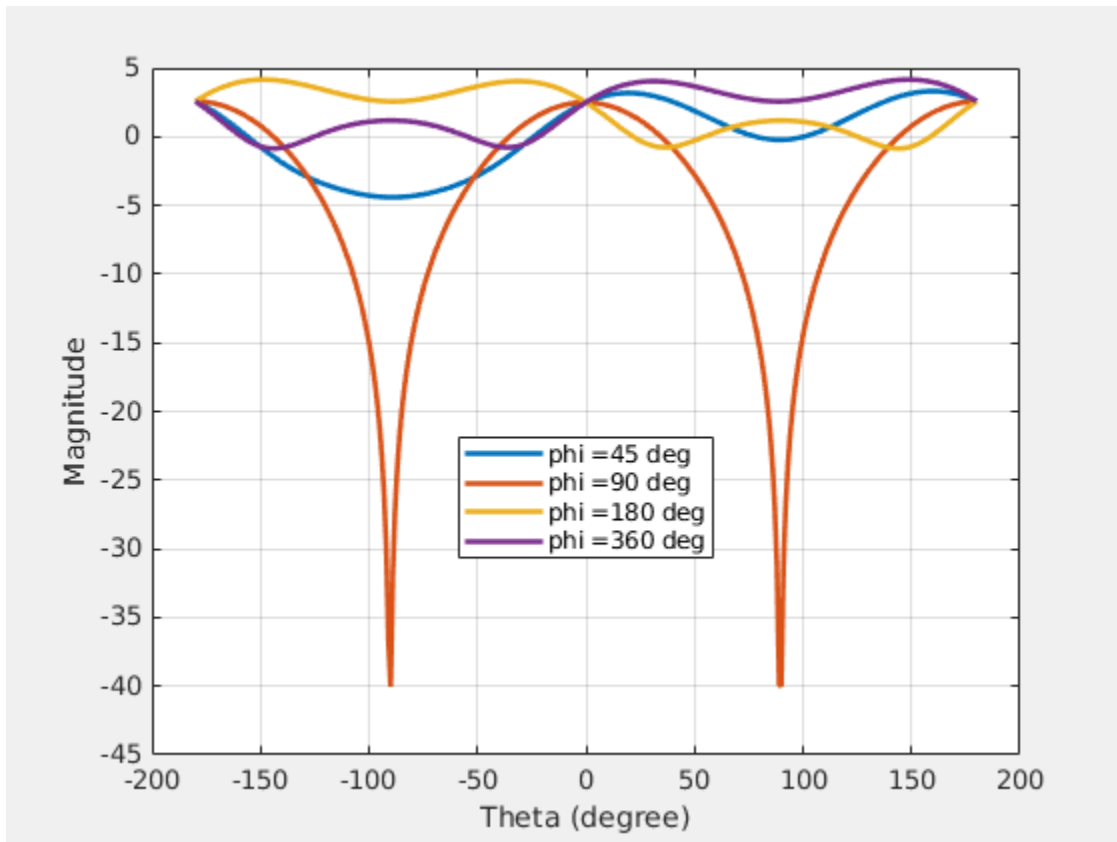
To plot a 2D slice on polar coordinate system, modify the `Slice` flag to either 'phi' or 'theta', depending on the plane you want to view data in. You should also modify the `SliceValue` flag to give a vector of phi or theta values for the slices. The slice values should be in the input data. Specify the `CoordinateSystem` flag as polar to view using a polar plot.

```
patternCustom(M(:,3),M(:,2),M(:,1),'CoordinateSystem','polar','Slice', ...
    'phi','SliceValue',[45 90 180 360]);
```



Specify the `CoordinateSystem` flag as `rectangular` to view the above case using a rectangular plot.

```
patternCustom(M(:,3),M(:,2),M(:,1),'CoordinateSystem','rectangular', ...
    'Slice','phi','SliceValue',[45 90 180 360]);
```

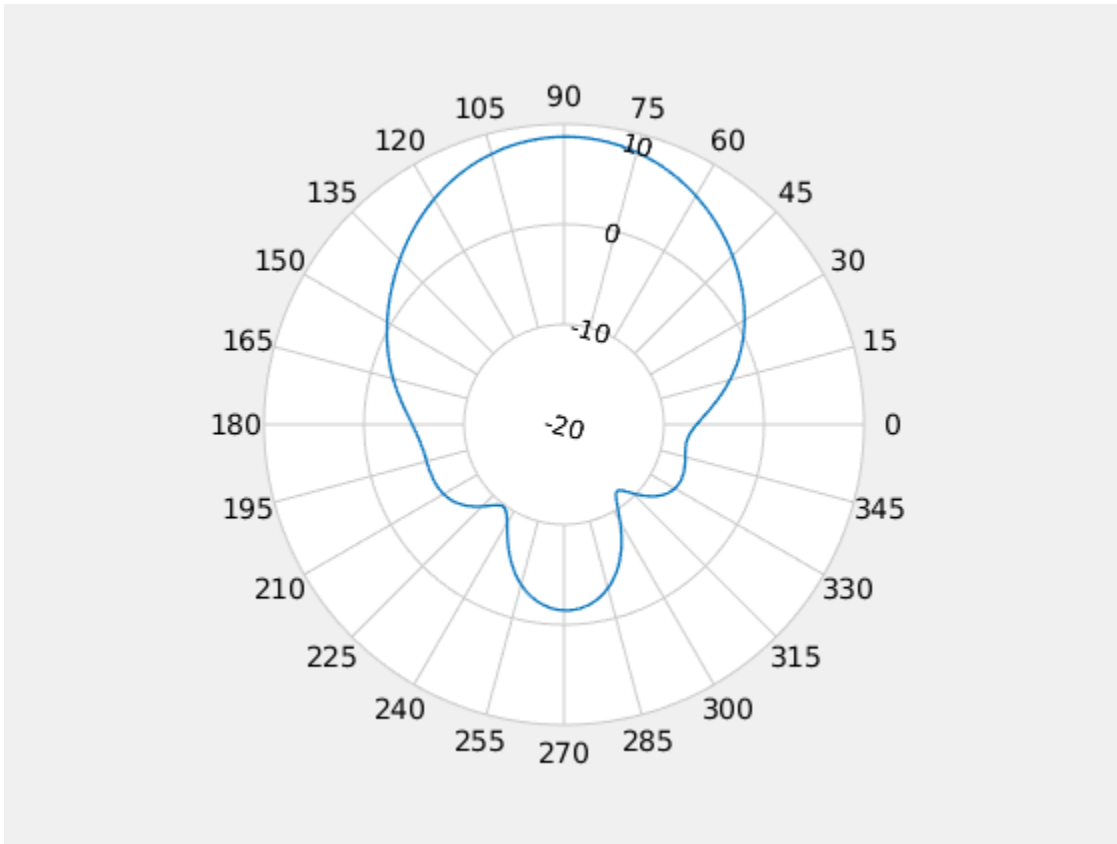


Plot 2D Polar Data

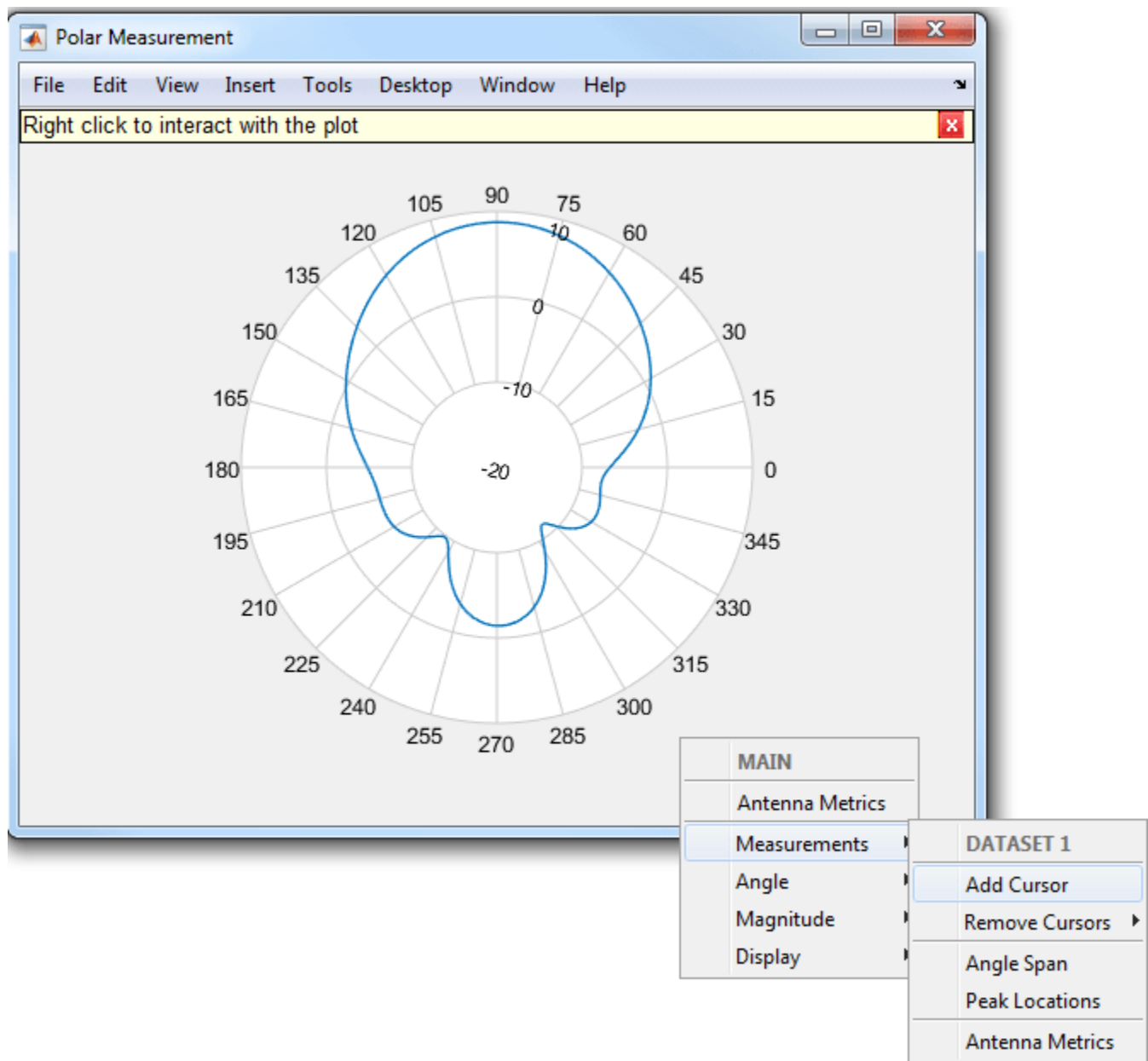
To plot a 2D polar data, you can use the `polarpattern` function as shown below. The plot generated is an interactive plot that allows the user to perform antenna specific measurements as well. The data in this case is stored in a `.mat` file. The file contains directivity values calculated over 360 degrees with one degree separation.

```
load polardata
```

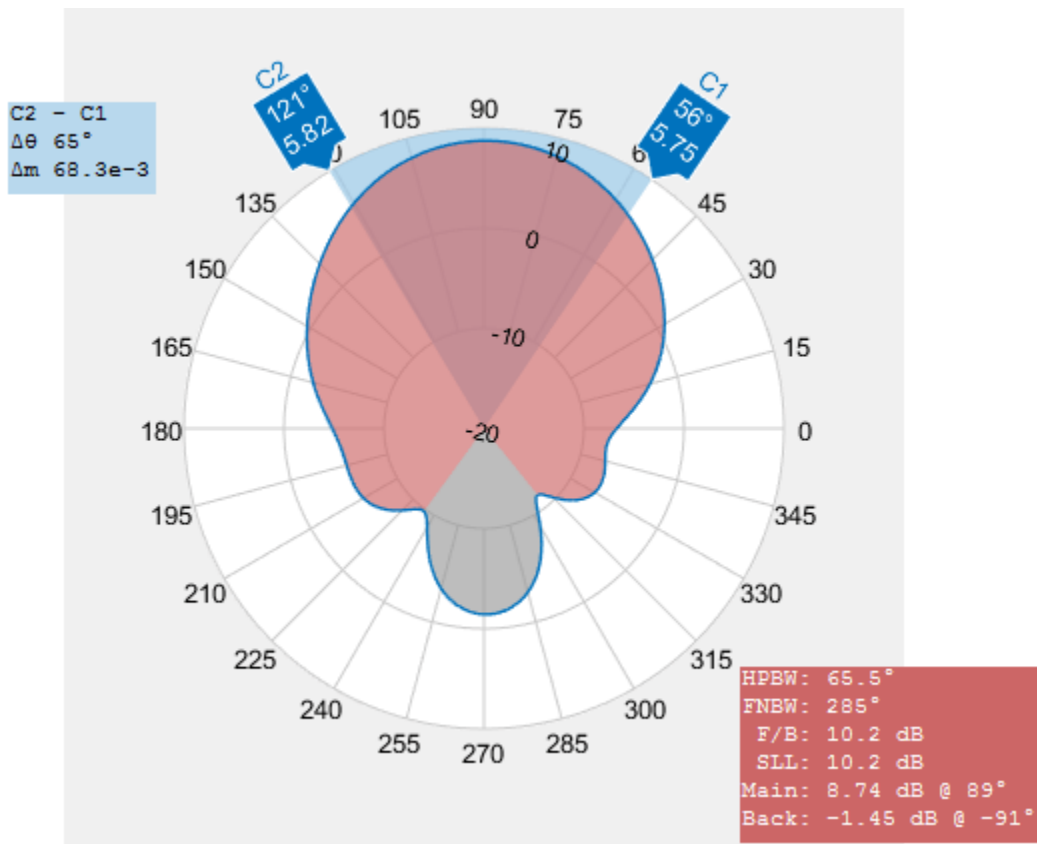
```
p = polarpattern(ang, D);
```

Right click in the figure window to interact with the plot. The figure below shows a screen shot of the context menu. The context menus can be used to do measurements such as peak detection, beamwidth calculation etc. You can also add a cursor by right clicking inside the polar circle.



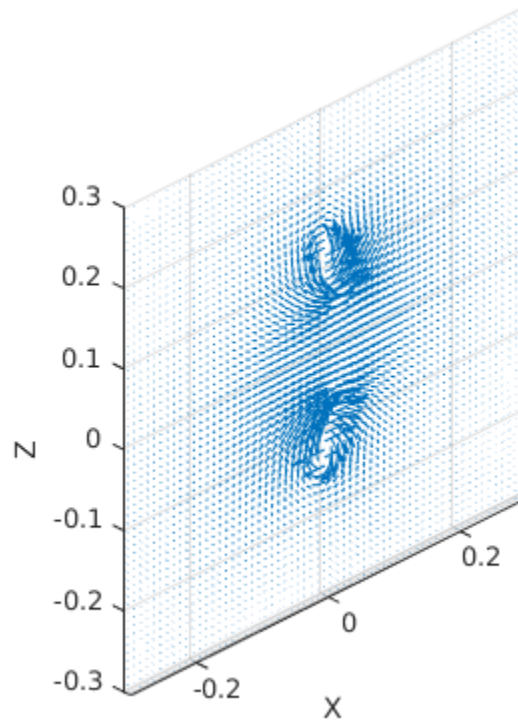
Select the Antenna Metrics option in the context menu shown above, to visualize the antenna specific measurements as shown below.



Plot Vector Field Data at a Point in Space

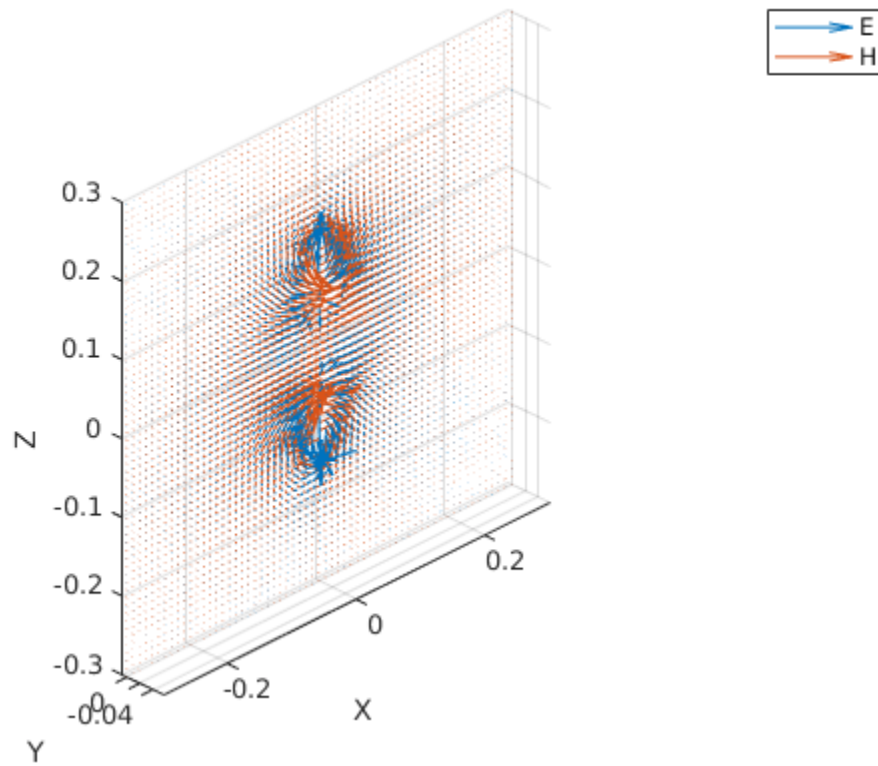
To plot vector electric and/or magnetic fields at any point in space, use the `fieldsCustom` function as shown below. The mat file `EHfielddata` contains the E and H field data as well as the points in space specified as x, y and z coordinates. The electric and magnetic fields are complex quantities and have x, y and z components at every point in space. The fields can be artificially scaled for better visualization.

```
load EHfielddata;
figure;
fieldsCustom(H, points, 5);
```



The function is used to plot one field quantity at a time. To plot both E and H fields on the same plot, use the hold on command.

```
figure;  
fieldsCustom(gca, E, points, 5);  
hold on;  
fieldsCustom(gca, H, points, 5);  
hold off;  
legend('E', 'H');
```

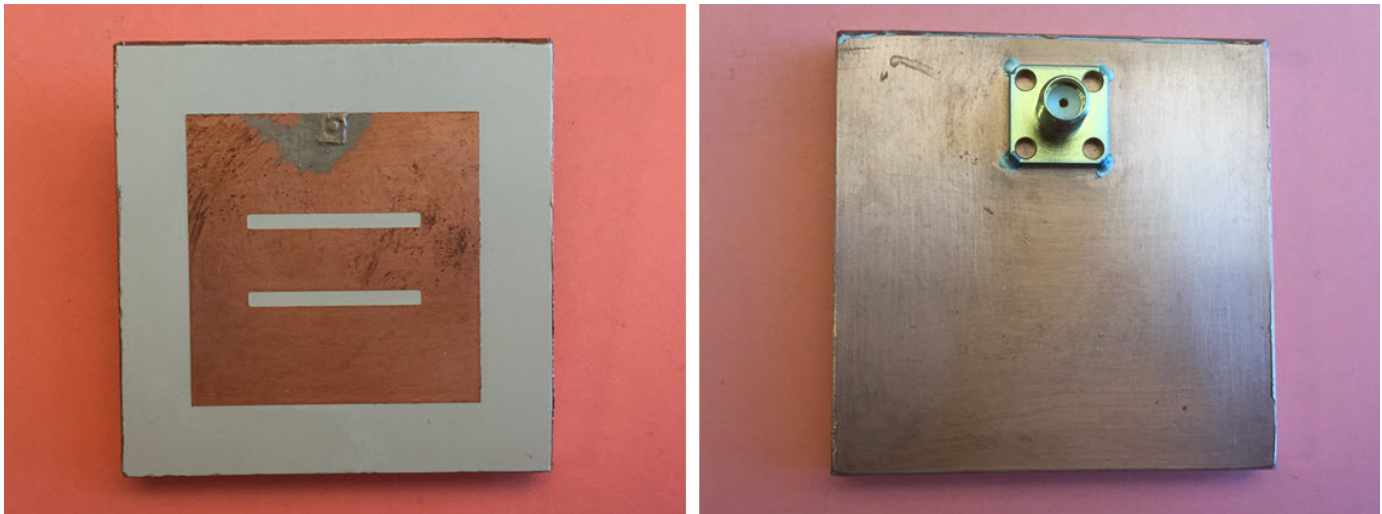
**See Also**

“3D Reconstruction of Radiation Pattern From 2D Orthogonal Slices” on page 5-497 | “Antenna Array Beam Scanning Visualization on a Map” on page 5-335

Double-Slot Cavity Patch on TMM10 Substrate

This example shows you how to create a custom slot cavity patch using custom antenna geometry and thick dielectric substrate. A double slot cavity patch consists of a double slot patch, backed by a cavity and probe-fed. The cavity is filled with TMM10 substrate. The cavity backing helps to reduce back radiation. You can use this antenna for microwave imaging by placing the antenna close to the human body.

Below is the picture of the fabricated slotted patch antenna.



Fabricated slotted patch antenna(with permission from Antenna Lab, WPI)

Create a Double Slot Patch

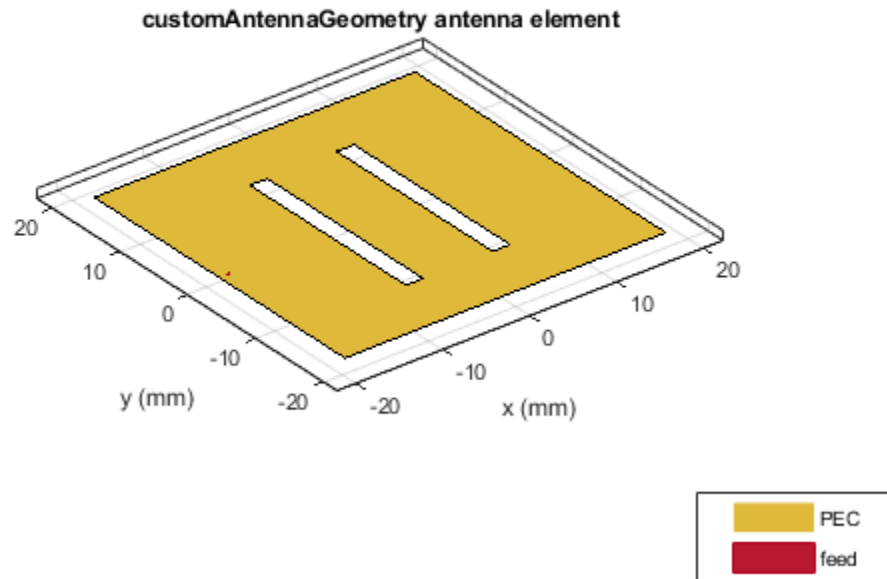
A double slot patch is not available as part of the Antenna Toolbox Library. However, you can create the geometry using basic rectangle shape primitive. You can put this information into a customAntennaGeometry antenna object and Boolean operation is performed to create the slots.

```

rect1 = antenna.Rectangle('Length', 37e-3, 'Width', 37e-3);
p1 = getShapeVertices(rect1);
slot1 = antenna.Rectangle('Length', 2e-3, 'Width', 23e-3, ...
    'Center', [-5e-3, 0], 'NumPoints', [5 10 5 10]);
p2 = getShapeVertices(slot1);
slot2 = antenna.Rectangle('Length', 2e-3, 'Width', 23e-3, ...
    'Center', [ 5e-3, 0], 'NumPoints', [5 10 5 10]);
p3 = getShapeVertices(slot2);
feed1 = antenna.Rectangle('Length', 0.5e-3, 'Width', 0.5e-3, ...
    'Center', [-17.25e-3 0]);
p4 = getShapeVertices(feed1);

ant = customAntennaGeometry;
ant.Boundary = {p1,p2,p3,p4};
ant.Operation = 'P1-P2-P3+P4';
ant.FeedLocation = [-17.5e-3,0,0];
ant.FeedWidth    = 0.5e-3;
figure
show(ant);

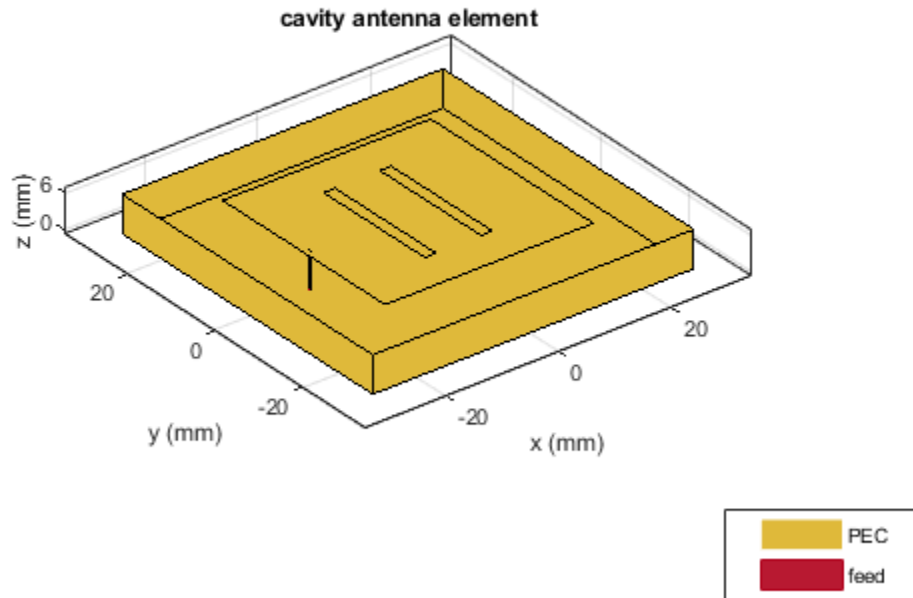
```



Provide Cavity Backing with Probe Feed

Use the slot patch created as an exciter for the cavity and enable probe feed. Below you see the patch antenna structure on an air substrate.

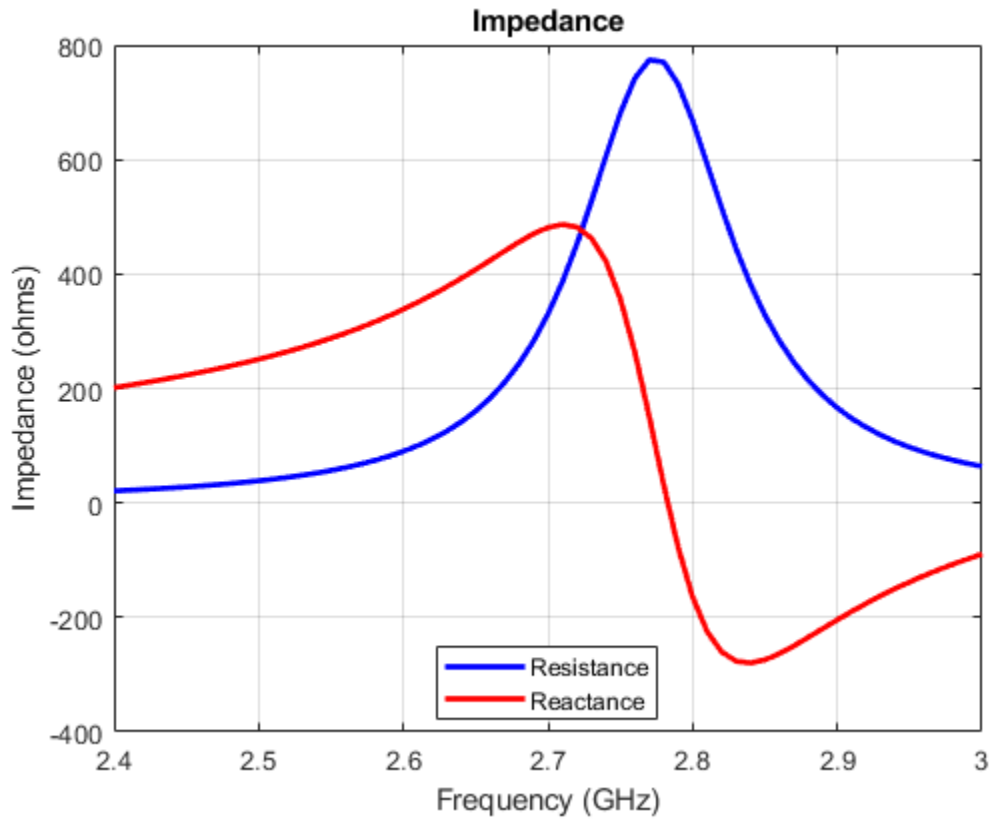
```
c = cavity('Exciter', ant, 'Length', 57e-3, 'Width', 57e-3, 'Height', ...  
        6.35e-3, 'Spacing', 6.35e-3, 'EnableProbeFeed', 1);  
figure;  
show(c);
```



Calculate the Antenna Impedance

Calculate the antenna impedance over the range of 2.4 GHz to 3 GHz. From the figure, observe that the antenna resonates around 2.76 GHz.

```
figure;  
impedance(c, linspace(2.4e9, 3.0e9, 61));
```

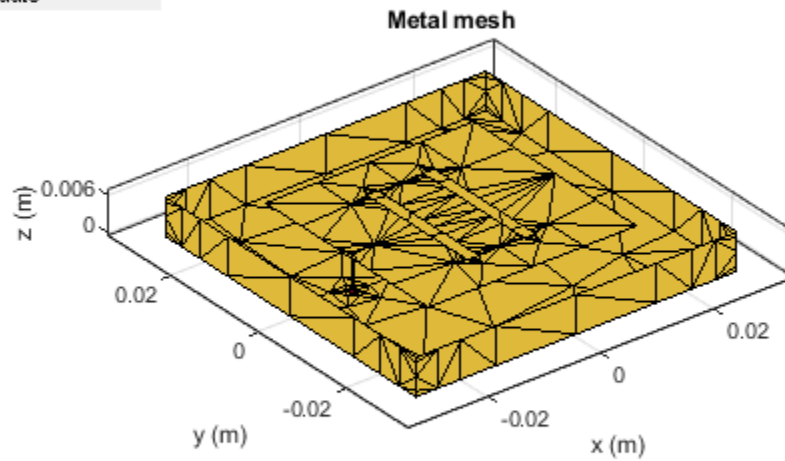



Visualize the Antenna Mesh

At the highest frequency of 2.2 GHz, the wavelength (λ) in TMM10 dielectric is 43.6 mm. So the substrate thickness is $\lambda/7$. So to make the thick substrate accurate thick substrate, two layers of tetrahedra are automatically generated.

```
figure;  
mesh(c);
```

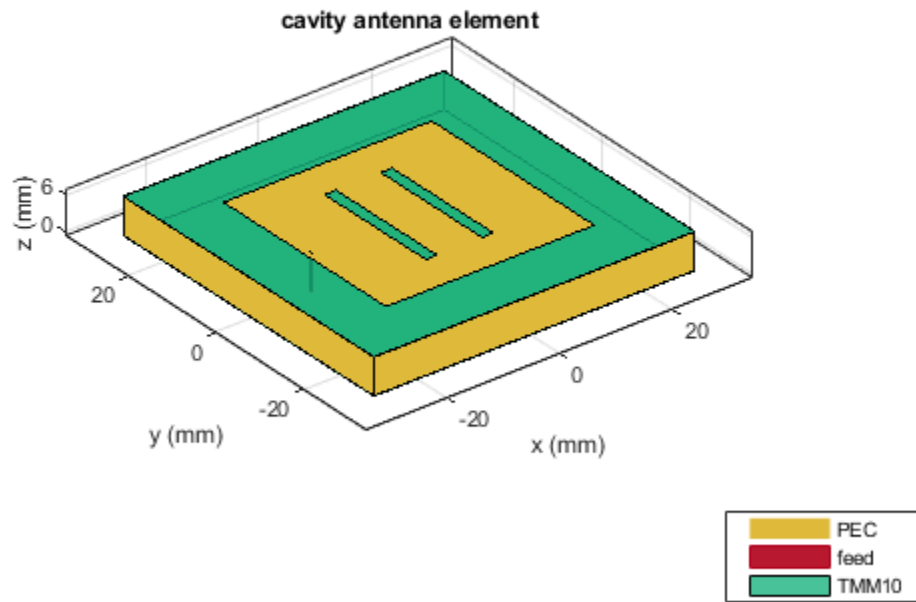
NumTriangles: 326
NumTetrahedra: 0
NumBasis: 416
MaxEdgeLength: 0.015186
MeshMode: auto



Add a Dielectric Substrate

Fill the space between the cavity and the patch with Rogers TMM10 substrate from the dielectric catalog.

```
c.Substrate = dielectric('TMM10');  
show(c);
```



Meshing the Antenna

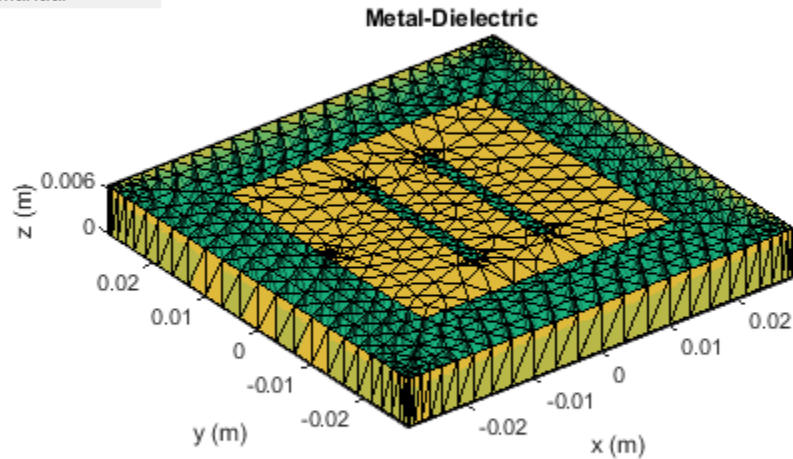
Mesh the antenna with a maximum edgelenhth of 3.5 mm.

```
mesh(c, 'MaxEdgeLength', 3.5e-3);
```

```

NumTriangles: 1513
NumTetrahedra: 2886
NumBasis:
MaxEdgeLength: 0.0035
MeshMode: manual

```



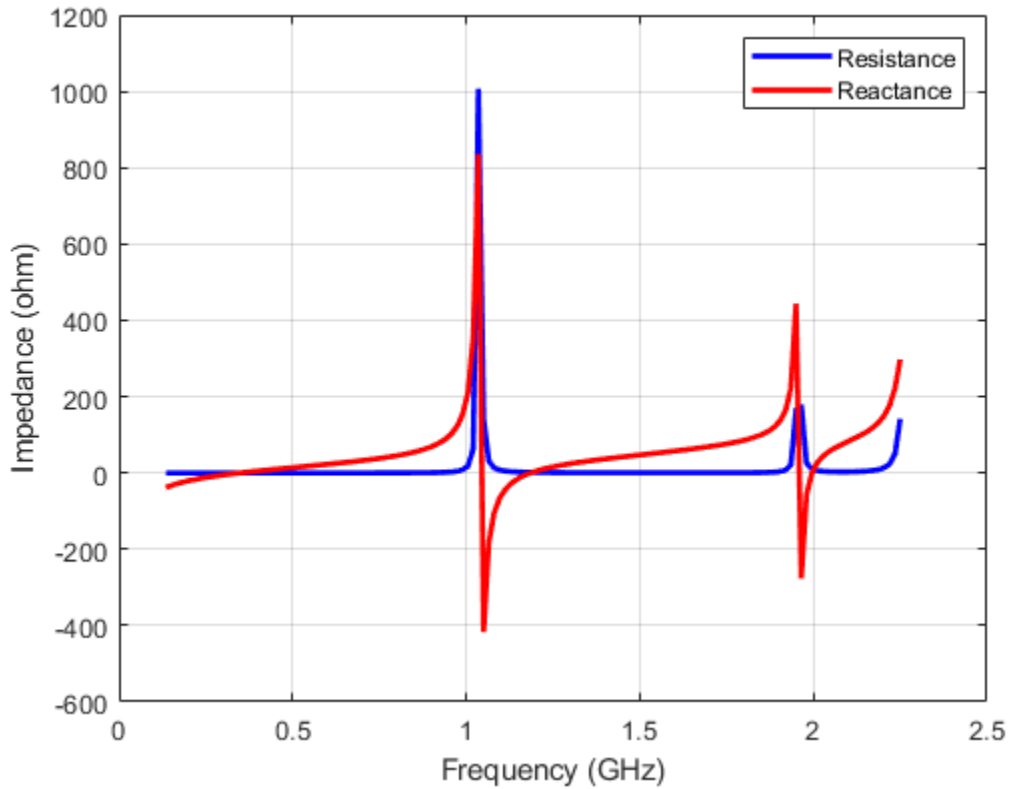
Calculate the Antenna Impedance

The effect of the dielectric constant is to move the resonance by a factor of $\sqrt{9.8} \sim 3$, approximately. So antenna miniaturization is achieved by adding a dielectric substrate. However, as the dielectric constant of the substrate increases the antenna higher Q-factor creates a sharp resonance. Due to the large numbers of frequency steps involved, the results are pre-computed and stored. Only one of the highest frequency computations is shown.

```

zl = impedance(c, 2.2e9);
load cavitypatch;
figure;
plot(freq./1e9, real(Z), 'b', freq./1e9, imag(Z), 'r', 'LineWidth',2);
xlabel('Frequency (GHz)');
ylabel('Impedance (ohm)');
legend('Resistance','Reactance');
grid on;

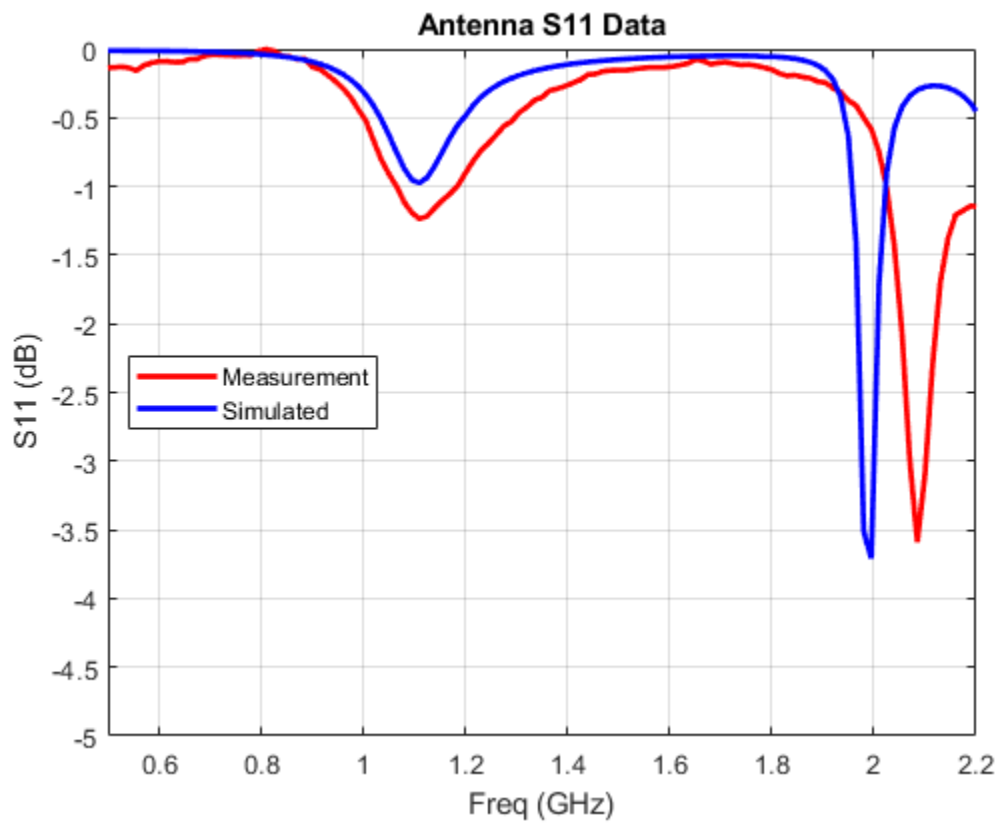
```



Fabricated Slot-patch Antenna

The double slot patch antenna was manufactured and its reflection coefficient was measured at the Antenna Lab in Worcester Polytechnic Institute (WPI). As seen from the plot below, a very good agreement is achieved at the lower frequency. At the upper frequency the difference in the reflection coefficient is around 3.5%. This could be due to the SMA connector present on the actual antenna or the frequency variation in the dielectric constant for the substrate.

```
figure
plot(freq./1e9,s11_meas,'-r','LineWidth',2); grid on;
hold on
plot(freq./1e9,s11_sim,'-b','LineWidth',2); grid on;
xlabel('Freq (GHz)')
ylabel('S11 (dB)')
axis([0.5,2.2,-5,0])
title('Antenna S11 Data')
legend('Measurement','Simulated','Location','best')
```



See Also

“Design, Analysis, and Prototyping of Microstrip-Fed Wide-Slot Antenna” on page 5-356

Effect of Mutual Coupling on MIMO Communication

This example shows how the antenna mutual coupling affects the performance of an orthogonal space-time block code (OSTBC) transmission over a multiple-input multiple-output (MIMO) channel. The transmitter and receiver have two dipole antenna elements each. The BER vs. SNR curves are plotted under different correlation and coupling scenarios. To run this example, you need Antenna Toolbox™.

System Parameters

A QPSK modulated Alamouti OSTBC is simulated over a 2x2 quasi-static frequency-flat Rayleigh channel [1]. The system operates at 2.4 GHz. The SNR range to be simulated is 0 to 10 dB.

```
fc = 2.4e9;           % Center frequency
Nt = 2;              % Number of Tx antennas
Nr = 2;              % Number of Rx antennas
blkLen = 2;         % Alamouti code block length
snr = 0:10;         % SNR range
maxNumErrs = 3e2;   % Maximum number of errors
maxNumBits = 5e4;   % Maximum number of bits
```

Create objects to perform QPSK modulation and demodulation, Alamouti encoding and combining, AWGN channel as well as BER calculation.

```
qpskMod = comm.QPSKModulator;
qpskDemod = comm.QPSKDemodulator;
alamoutiEnc = comm.OSTBCEncoder( ...
    'NumTransmitAntennas', Nt);
alamoutiDec = comm.OSTBCCombiner( ...
    'NumTransmitAntennas', Nt, ...
    'NumReceiveAntennas', Nr);
awgnChanNC = comm.AWGNChannel( ... % For no coupling case
    'NoiseMethod', 'Signal to noise ratio (SNR)', ...
    'SignalPower', 1);
berCalcNC = comm.ErrorRate;        % For no coupling case

% Clone objects for mutual coupling case
awgnChanMC = clone(awgnChanNC);
berCalcMC = clone(berCalcNC);
```

Antenna Arrays and Coupling Matrices

A two-element resonant dipole array is used at both transmit (Tx) and receive (Rx) side. At Tx, the dipoles are spaced a half-wavelength apart. At Rx, the spacing is a tenth of a wavelength.

```
txSpacing = 0.5;
rxSpacing = 0.1;
lambda = physconst('lightspeed')/fc;
antElement = dipole( ...
    'Length', lambda/2, ...
    'Width', lambda/100);
txArray = linearArray( ...
    'Element', antElement, ...
    'NumElements', Nt, ...
    'ElementSpacing', txSpacing*lambda);
rxArray = linearArray( ...
```

```

'Element',      antElement,...
'NumElements', Nr,...
'ElementSpacing', rxSpacing*lambda);

```

The coupling matrix is calculated based on a circuit model of the array as per [[2]]. The s-parameter calculation is performed for the transmit and receive arrays and from this the impedance matrix representation of the array is derived.

```

txMCMtx = helperCalculateCouplingMatrix(txArray, fc, [1 Nr]);
rxMCMtx = helperCalculateCouplingMatrix(rxArray, fc, [1 Nr]);

```

Spatial Correlation Matrices

The transmit and receive spatial correlation matrices capture the propagation environment of the channel. Without coupling, it is assumed that the two elements at Tx are uncorrelated and the two elements at Rx have high correlation. The combined/overall correlation matrix for the whole channel is their Kronecker product.

```

txCorrMtx = eye(2);
rxCorrMtx = [1 0.9; 0.9 1];
combCorrMtx = kron(txCorrMtx, rxCorrMtx);

```

With coupling, we use the approach in [[3]] to modify the Tx and Rx correlation matrices by pre and post-multiplying them by the corresponding coupling matrices. This is valid under the assumption that the correlation and coupling can be modeled independently.

```

txMCCorrMtx = txMCMtx * txCorrMtx * txMCMtx';
rxMCCorrMtx = rxMCMtx * rxCorrMtx * rxMCMtx';

```

The combined spatial correlation with coupling is `kron(txMCCorr, rxMCCorr)`. Alternatively, we can treat the Tx/Rx coupling matrix as being "absorbed" into the Tx/Rx correlation matrix and derive the combined correlation matrix as follows:

```

txSqrtCorrMtx = txMCMtx * sqrtm(txCorrMtx);
rxSqrtCorrMtx = rxMCMtx * sqrtm(rxCorrMtx);
combMCCorrMtx = kron(txSqrtCorrMtx, rxSqrtCorrMtx);
combMCCorrMtx = combMCCorrMtx * combMCCorrMtx';

```

MIMO Channel Modeling

Create two `comm.MIMOChannel` objects to simulate the 2x2 MIMO channels with and without coupling. The combined spatial correlation matrix is assigned in each case. The `MaximumDopplerShift` property of the objects is set to 0 to model a quasi-static channel.

```

mimoChanNC = comm.MIMOChannel( ... % For no coupling case
    'MaximumDopplerShift',      0, ...
    'SpatialCorrelationSpecification', 'Combined', ...
    'SpatialCorrelationMatrix',  combCorrMtx,...
    'PathGainsOutputPort',     true);

% Clone objects for mutual coupling case
mimoChanMC = clone(mimoChanNC);
mimoChanMC.SpatialCorrelationMatrix = combMCCorrMtx;

```

Simulations

Simulate the QPSK modulated Alamouti code for each SNR value with and without antenna coupling. One Alamouti code is simulated through the MIMO channel in each iteration. To model a quasi-static

channel, we reset the `comm.MIMOChannel` object to obtain a new set of channel gains for each code transmission (iteration).

```

% Set up a figure to visualize BER results
h1 = figure; grid on; hold on;
ax = gca;
ax.YScale = 'log';
xlim([snr(1), snr(end)]); ylim([1e-3 1]);
xlabel('SNR (dB)'); ylabel('BER');
h1.NumberTitle = 'off';
h1.Name = 'Orthogonal Space-Time Block Coding';
title('Alamouti-coded 2x2 System - High Coupling, High Correlation');

s = rng(108); % For repeatability
[berNC, berMC] = deal(zeros(3,length(snr)));

% Loop over SNR values
for idx = 1:length(snr)
    awgnChanNC.SNR = snr(idx);
    awgnChanMC.SNR = snr(idx);
    reset(berCalcNC);
    reset(berCalcMC);

    while min(berNC(2,idx),berMC(2,idx)) <= maxNumErrs && (berNC(3,idx) <= maxNumBits)
        % Generate random data
        txData = randi([0 3], blkLen, 1);

        % Perform QPSK modulation and Alamouti encoding
        txSig = alamoutiEnc(qpskMod(txData));

        % Pass through MIMO channel
        reset(mimoChanNC); reset(mimoChanMC);
        [chanOutNC, estChanNC] = mimoChanNC(txSig);
        [chanOutMC, estChanMC] = mimoChanMC(txSig);

        % Add AWGN
        rxSigNC = awgnChanNC(chanOutNC);
        rxSigMC = awgnChanMC(chanOutMC);

        % Perform Alamouti decoding with known channel state information
        decSigNC = alamoutiDec(rxSigNC, squeeze(estChanNC));
        decSigMC = alamoutiDec(rxSigMC, squeeze(estChanMC));

        % Perform QPSK demodulation
        rxDataNC = qpskDemod(decSigNC);
        rxDataMC = qpskDemod(decSigMC);

        % Update BER
        berNC(:, idx) = berCalcNC(txData, rxDataNC);
        berMC(:, idx) = berCalcMC(txData, rxDataMC);
    end

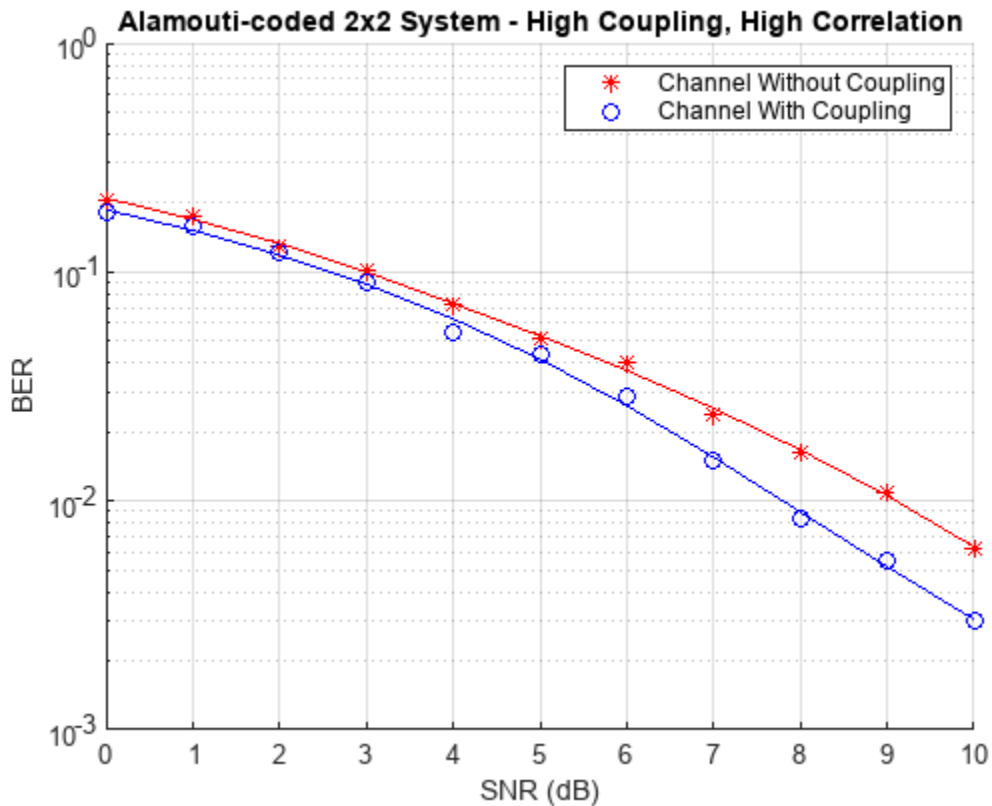
    % Plot results
    semilogy(snr(1:idx), berNC(1,1:idx), 'r*');
    semilogy(snr(1:idx), berMC(1,1:idx), 'bo');
    legend({'Channel Without Coupling', 'Channel With Coupling'});
    drawnow;
end

```

```

% Perform curve fitting
fitBERNC = berfit(snr, berNC(1,:));
fitBERMC = berfit(snr, berMC(1,:));
semilogy(snr, fitBERNC, 'r', snr, fitBERMC, 'b');
legend({'Channel Without Coupling', 'Channel With Coupling'});

```

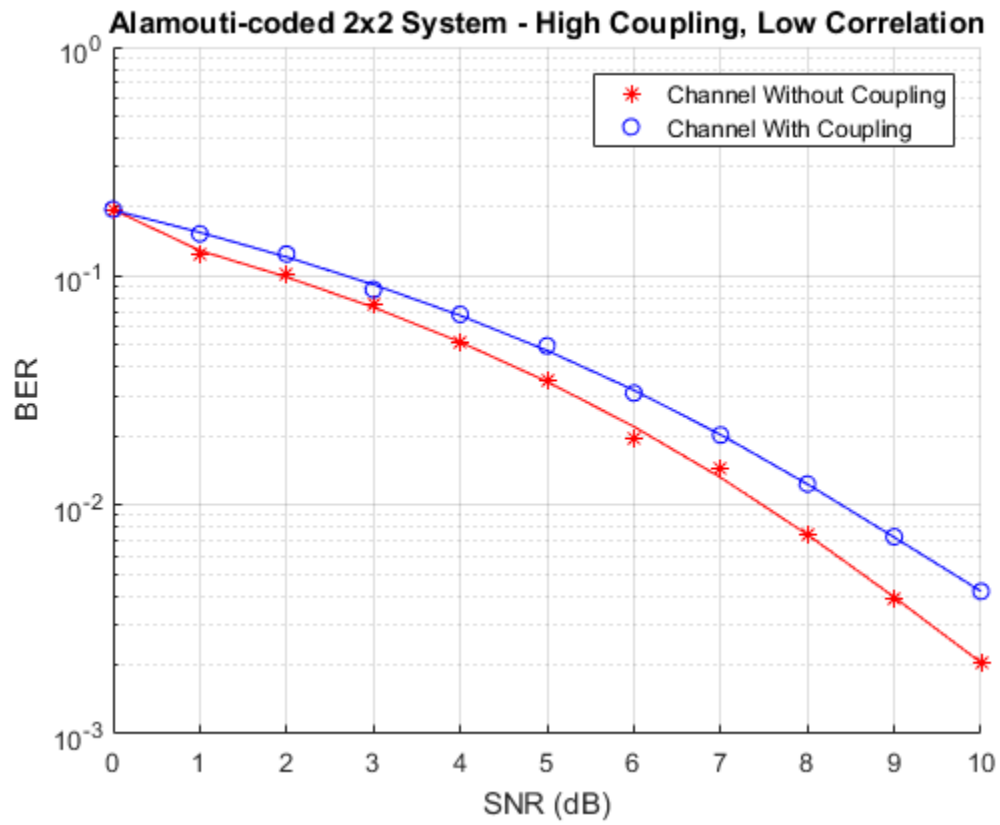


```
rng(s); % Restore RNG
```

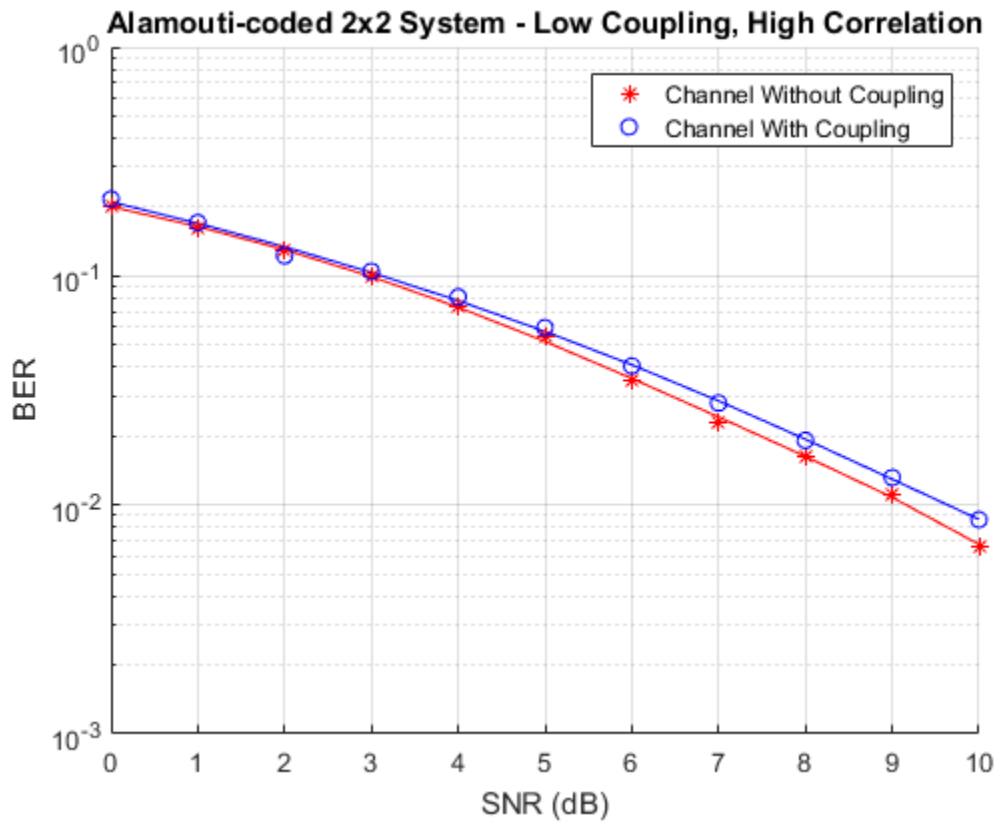
Further Exploration

The effect of correlation and mutual coupling on the BER performance can be further studied by modifying the correlation coefficient and/or by changing the spacing between the elements. The smaller the spacing is, the higher the coupling is. Similar to what has been done above for high correlation (0.9) and high coupling (spacing = 0.1λ) at Rx, we now show the BER vs. SNR results for low correlation (0.1) and/or low coupling (spacing = 0.5λ).

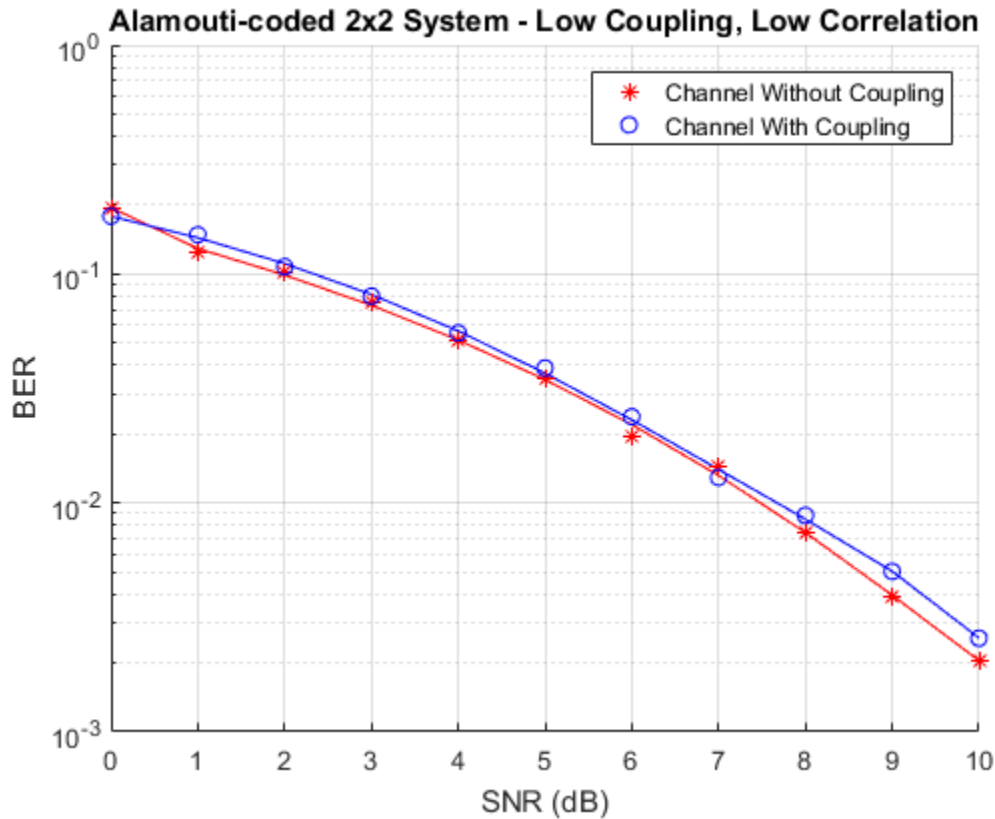
- High Coupling (spacing = 0.1λ), Low Correlation (0.1)



- Low Coupling (spacing = 0.5λ), High Correlation (0.9)



- Low Coupling (spacing = 0.5λ), Low Correlation (0.1)



Conclusion

The simulation results are similar to those reported in the first reference. A spacing of 0.5λ has a negligible impact on BER under both high and low correlation conditions. For the case with high coupling, i.e., 0.1λ element spacing, the results indicate that depending on the correlation conditions, the BER could be either higher or lower than if coupling were not considered.

Appendix

This example uses the following helper functions:

- `helperCalculateCouplingMatrix.m`

See Also

More About

- "FMCW Patch Antenna Array" on page 5-172
- "Antenna Diversity Analysis for 800 MHz MIMO" on page 5-159

References

- [1] A. A. Abouda, H. M. El-Sallabi, and S. G. Haggman, "Effect of Mutual Coupling on BER Performance of Alamouti Scheme," *IEEE International Symposium on Antennas and Propagation*, July 2006.

- [2] Gupta, I., and A. Ksienski. "Effect of Mutual Coupling on the Performance of Adaptive Arrays." *IEEE Transactions on Antennas and Propagation* 31, no. 5 (September 1983): 785-91. <https://doi.org/10.1109/TAP.1983.1143128>.
- [3] Y. Wu, J. P. Linnartz, J. W. M. Bergmans, and S. Attallah, "Effects of Antenna Mutual Coupling on the Performance of MIMO Systems," *Proc. 29th Symposium on Information Theory in the Benelux*, May 2008.

Switched Beam Array with Butler Matrix

This example shows a switched beam array of 4 resonant dipoles. The beam switching is accomplished by using a 4-by-4 Butler matrix. The effect of the beam switching is shown by observing the outputs of 4 receiving antennas that are placed in the far-field of the array at the approximate azimuthal angles corresponding to the beam peaks. The effect of mutual coupling between the array elements on the transmit side is accounted for using S-parameters.

This example requires the following products:

- RF Blockset

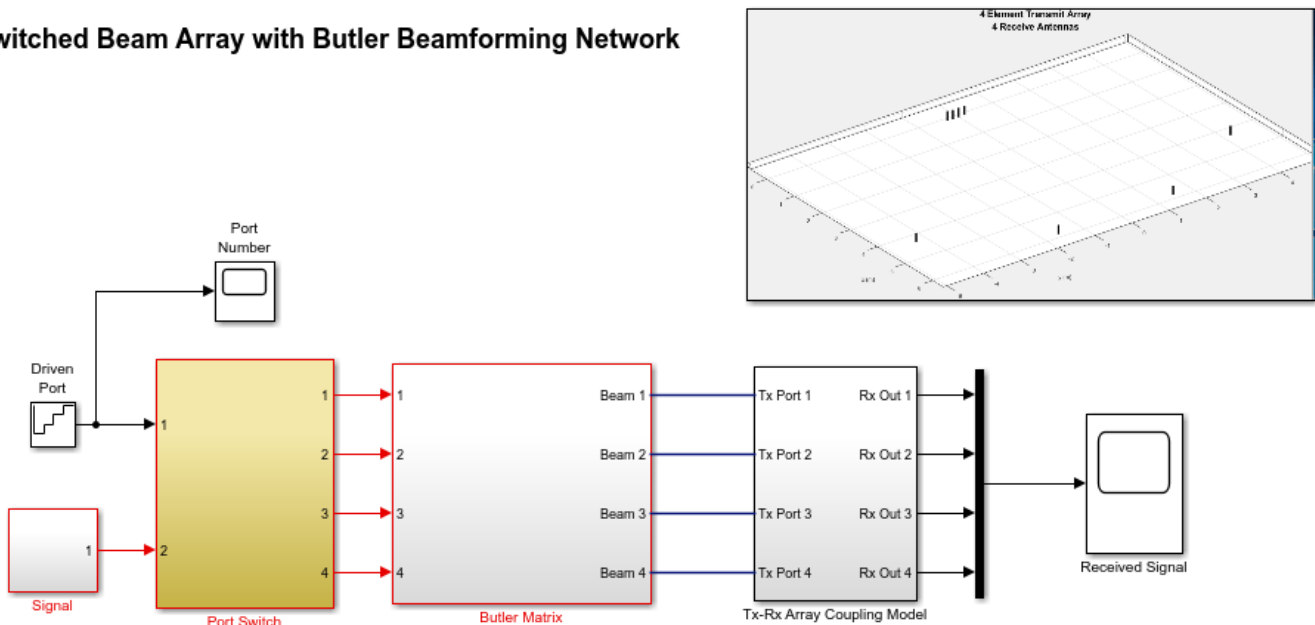
System Description

The system comprises of the following subsystems

- A Signal source and port switch
- A 4 X 4 Butler matrix implemented using directional couplers and phase shifters
- Array coupling model to capture the mutual coupling between the dipole linear array
- Channel transfer function model between linear array and the receiving antennas in the far-field

```
model = 'AtxSwitchedBeamArrayWithButlerMatrixModel';
open_system(model);
```

Switched Beam Array with Butler Beamforming Network



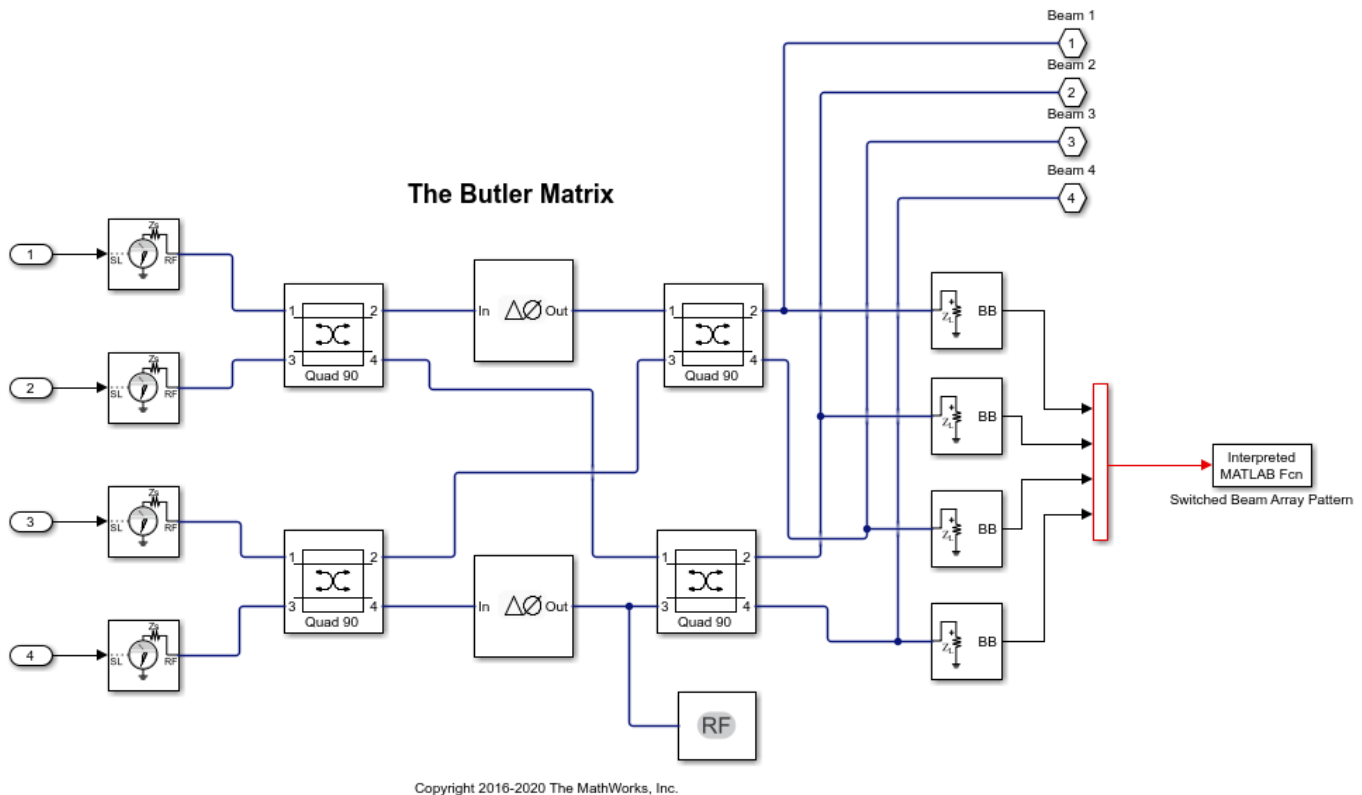
Copyright 2016-2020 The MathWorks, Inc.

Butler Matrix

The Butler matrix is a type of analog beamforming network that can be constructed from purely passive devices like directional couplers and phase shifters. The number of input and output ports in

a Butler matrix are equal. The output ports are connected directly to each antenna element. Depending on the input port which is excited, the signals on the output ports are phase-shifted such that the beam switches in direction.

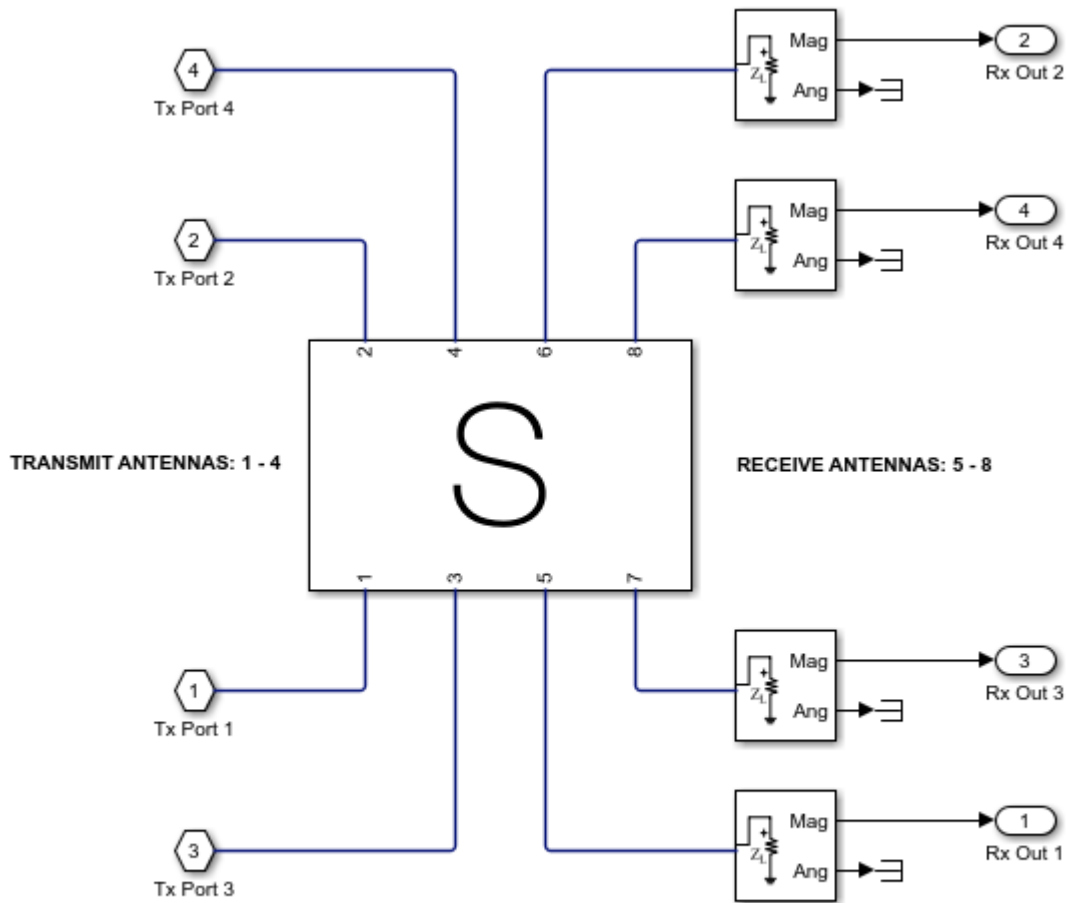
```
open_system([model '/Butler Matrix'], 'force');
```



Array Mutual Coupling Model, Channel, and Receiving Antenna Output

The transmit side comprises of a 4 element linear array of resonant dipoles spaced half-wavelength apart. There are 4 receiving antennas, resonant dipoles, placed in the far-field of this array. These 4 dipoles are placed in the azimuthal plane at 15 degrees, 45 degrees, 315 degrees and 345 degrees. The locations correspond to the expected beam peaks from the switched beam array output. The response at each of these far-field receiving antenna elements is computed as a superposition of the contributions from each antenna element in the 4-element transmit array. The channel is assumed to be free-space. To capture the interactions on the transmit side and the transfer function from the transmit to receive antennas, we compute the overall scattering parameters for the transmit-receive system. This is done by using the `conformalArray` in Antenna Toolbox. The resulting 8-port S-parameter matrix is loaded into the S-parameters block in RF Blockset. The first 4 ports correspond to the transmit side antenna elements and the ports 5-8 belong to the individual receiving antennas in the far-field.

```
open_system([model '/Tx-Rx Array Coupling Model'], 'force');
```

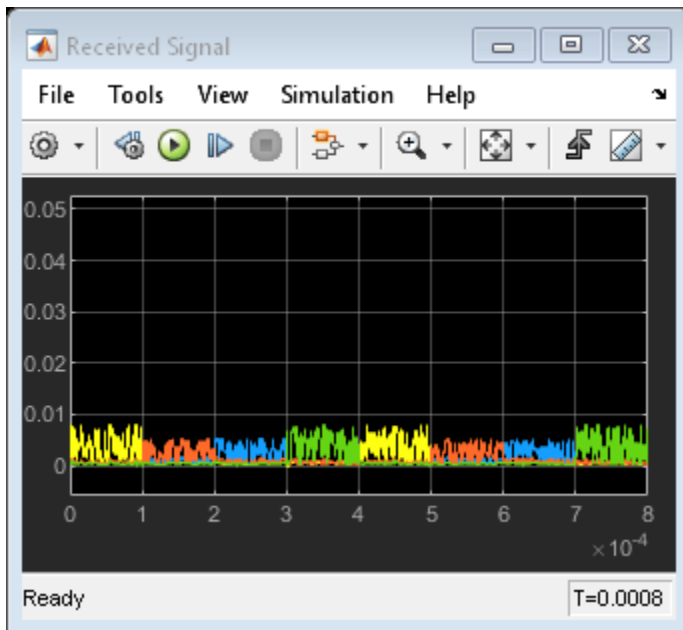
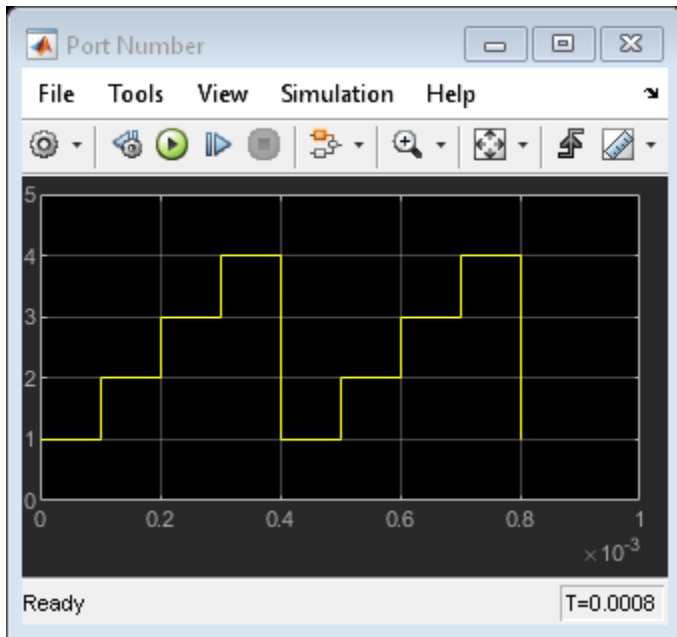



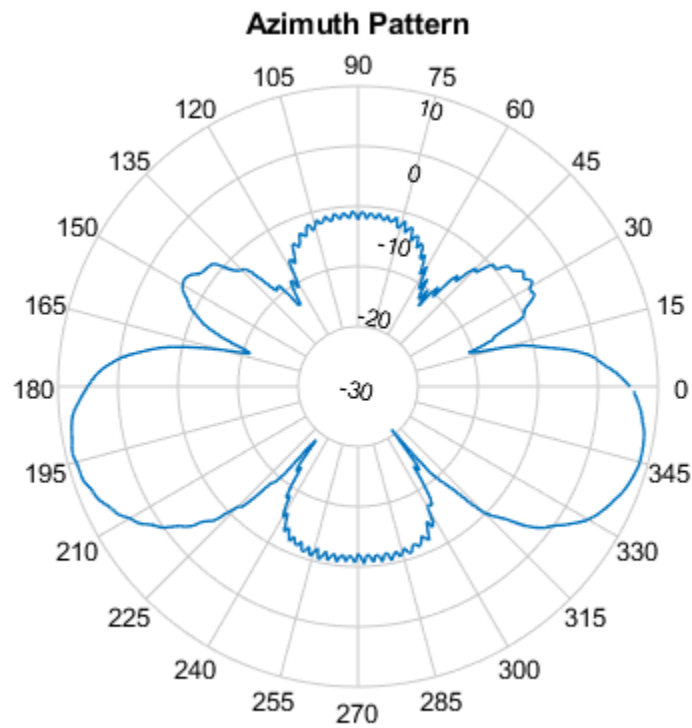
Copyright 2016-2020 The MathWorks, Inc.

Run Simulation

In the simulation, switch the input port that gets driven with the signal by using the repeating staircase sequence generator. Each port is excited for 0.1 ms. Depending on the port that gets excited, the array main beams switch in direction. This can be seen in the output signal levels at each receiving antenna. Within a 0.1 ms block of time, 1 out of the 4 received signals dominates the remaining three indicating the effect of the beamforming.

```
sim(model)
```





```
bdclose(model);  
clear model;
```

See Also

“Impedance Matching of Non-resonant (Small) Monopole” on page 5-141

Antenna Model Generation and Full-Wave Analysis From A Photo

This example demonstrates the process of using a photograph of a planar antenna to generate a viable antenna model and its subsequent analysis for port, surface and field characteristics. The Image Segmenter app will be used to perform segmentation on the image of an RFID tag, and the resulting boundaries will be used to set up the antenna model in Antenna Toolbox™. An initial impedance analysis will be done over a frequency range to understand the port characteristics of the antenna. After determining the resonance frequency, the current and far-field pattern will be calculated and plotted.

- This example depends on Image Processing Toolbox™

The RFID Tag

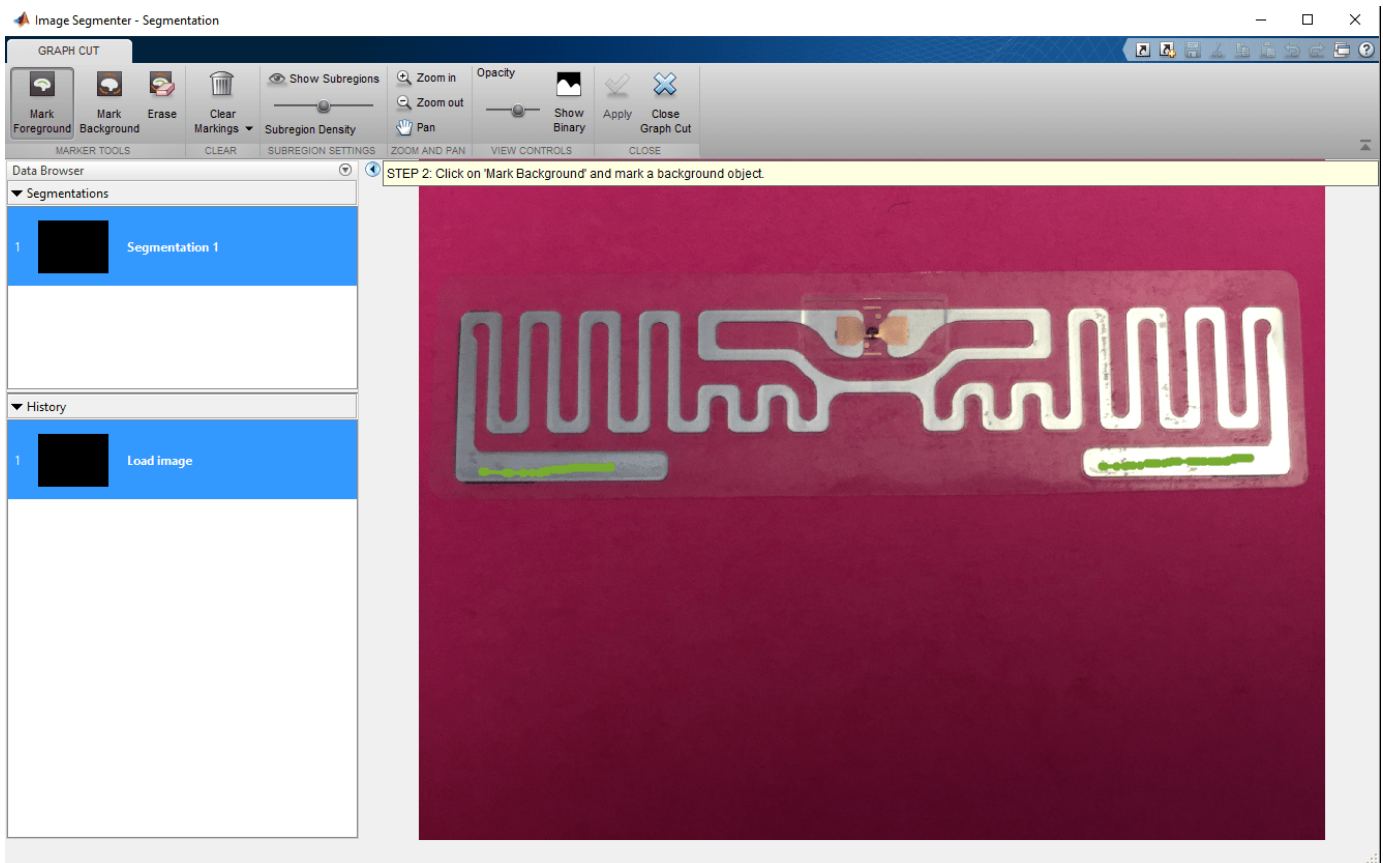
Begin by taking a photo of an RFID tag against a high-color contrast background. The camera is positioned directly over the antenna. This photo was taken with a smartphone.

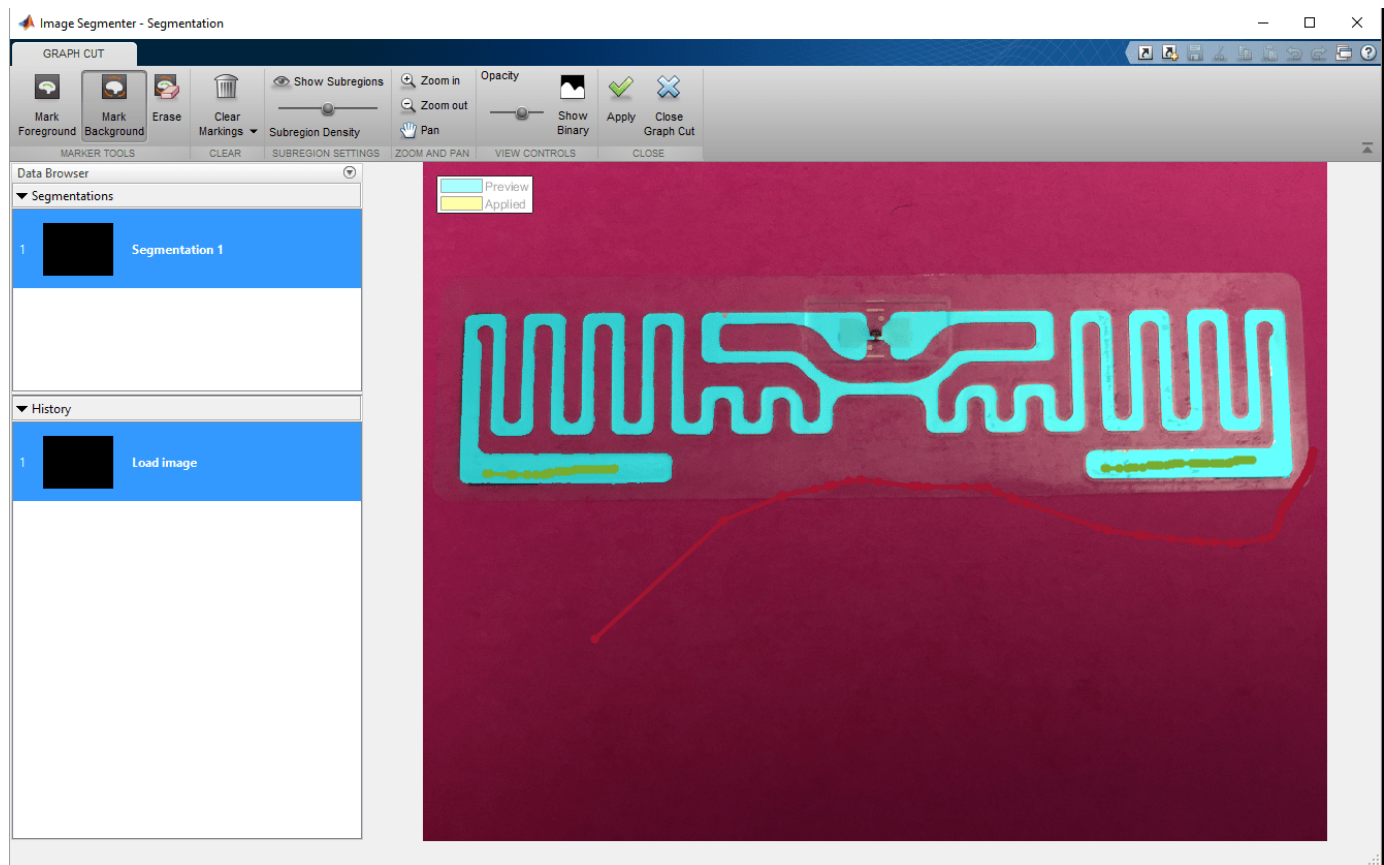


Image Segmentation Using Image Segmenter App

Choose Foreground and Background

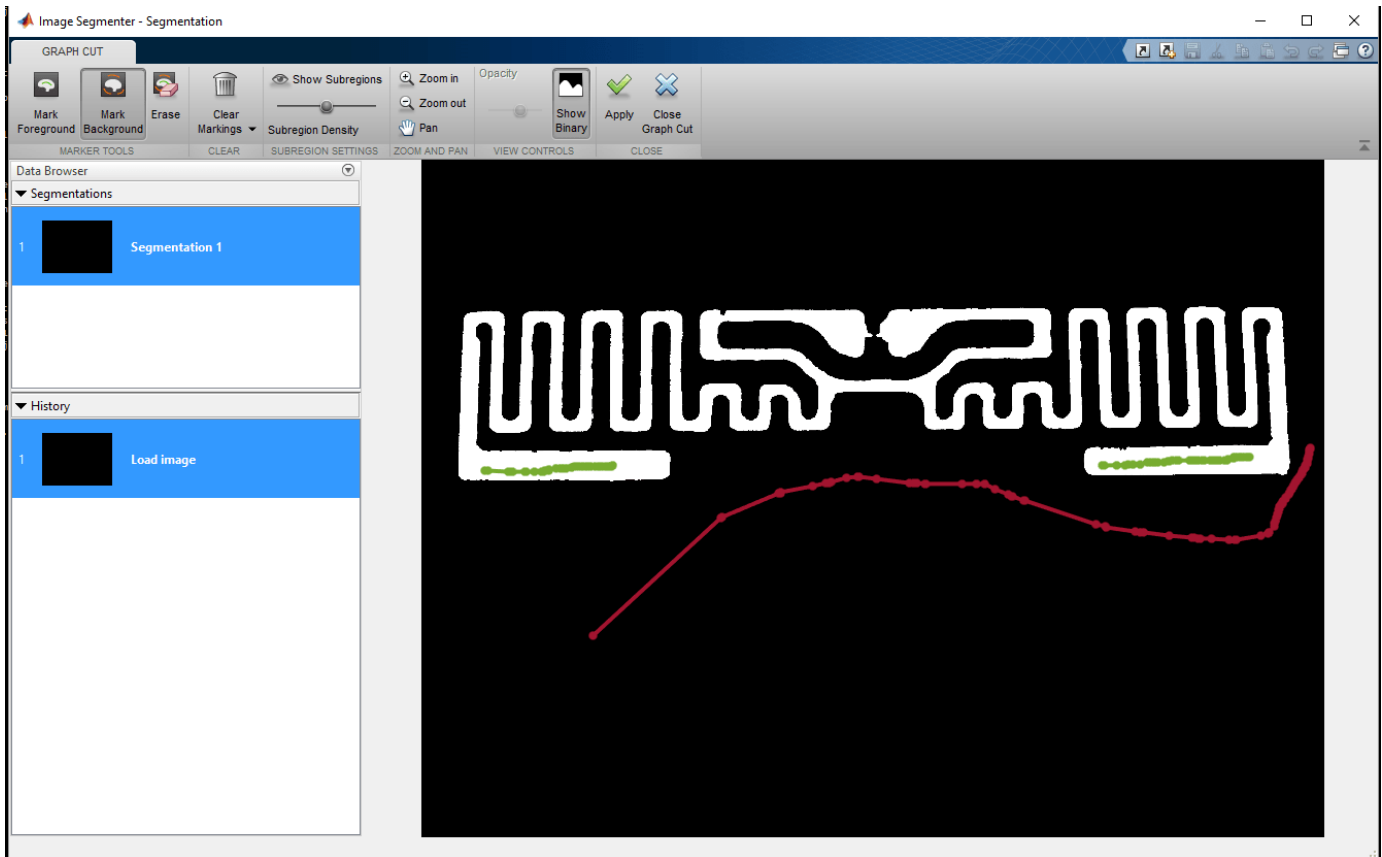
Using the app, import this antenna and choose the graph cut option on the toolstrip. Pick the foreground and background regions on the image. For this example, the foreground region is the metallized regions of the RFID tag and the background is the colored region.



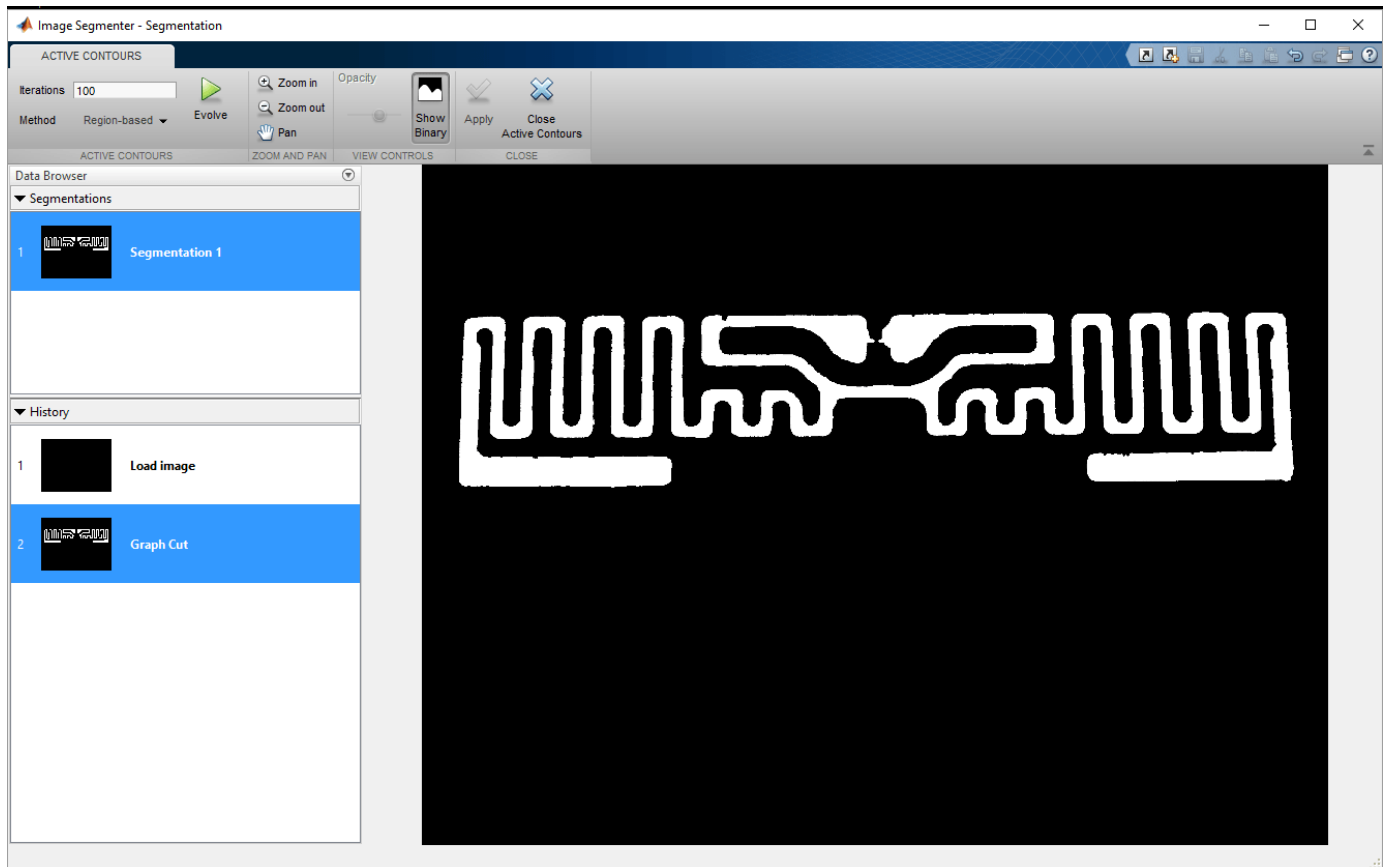


Improving segmentation quality

Choosing the background in the app results in an initial segmentation of the image into the foreground and background portions. If this segmentation is sufficient, apply the changes and proceed to the next part of the process by closing the graph cut tab.



If however, all parts of the antenna have not been identified yet, continue marking up the foreground and background regions. This allows the segmentation algorithm to improve upon the results. It may also be of use to adjust the subregion density to refine the quality of segmentation. After making the required adjustments, apply the changes and close the graph-cut segmentation tab in the app. On returning to the main tab, there are several options to further improve the segmentation. In this example we use the Active Contours option and evolve the existing segmentation to fill in any imperfections in the boundaries.

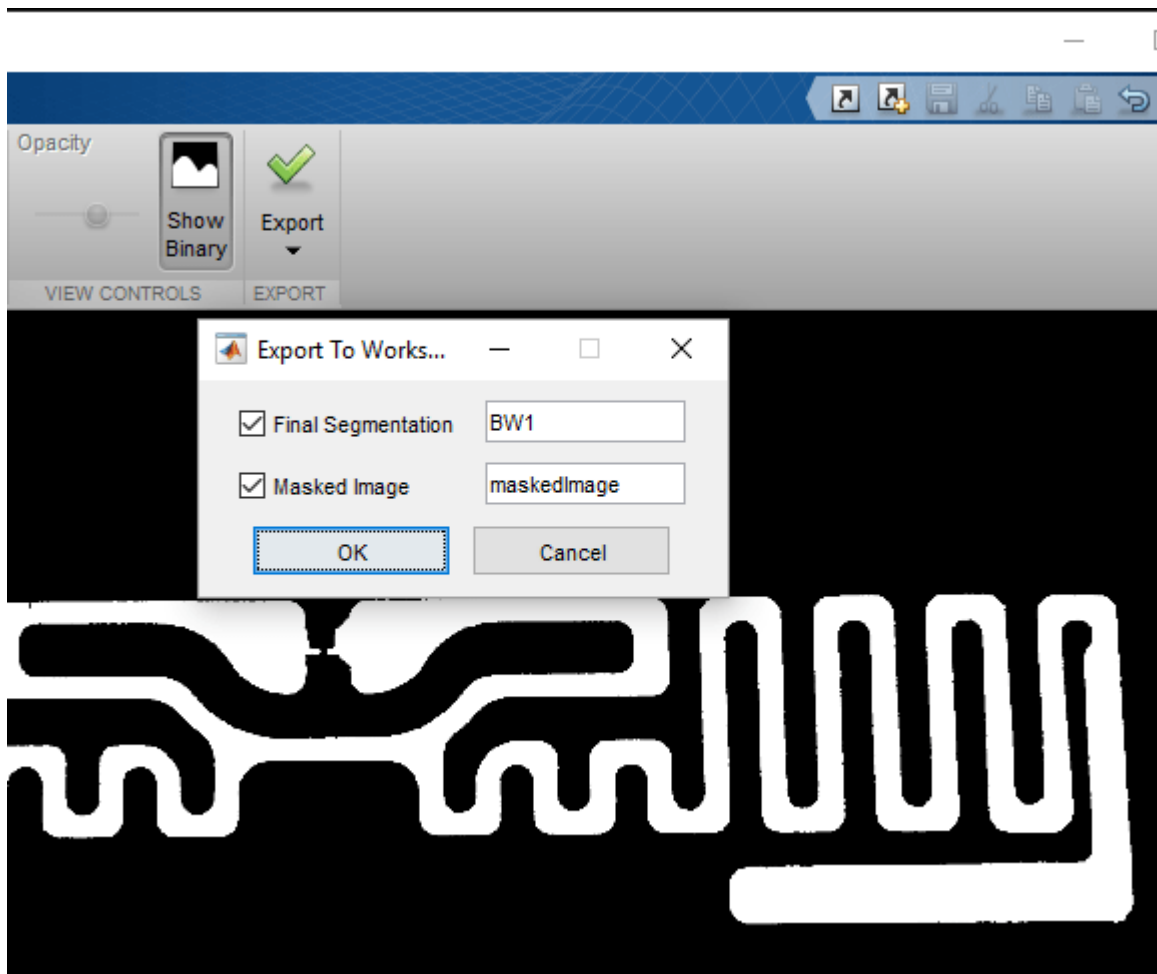


Notice that after several iterations of the active contours algorithm has executed, the boundary is much smoother and free of notch like artifacts. An example region is shown for comparison.



Export Code and Boundary

The color based segmentation process yields the mask of the antenna image. Use the export option on the app, to obtain a function and the boundary information which can then be used in a script for further processing (such as this one).

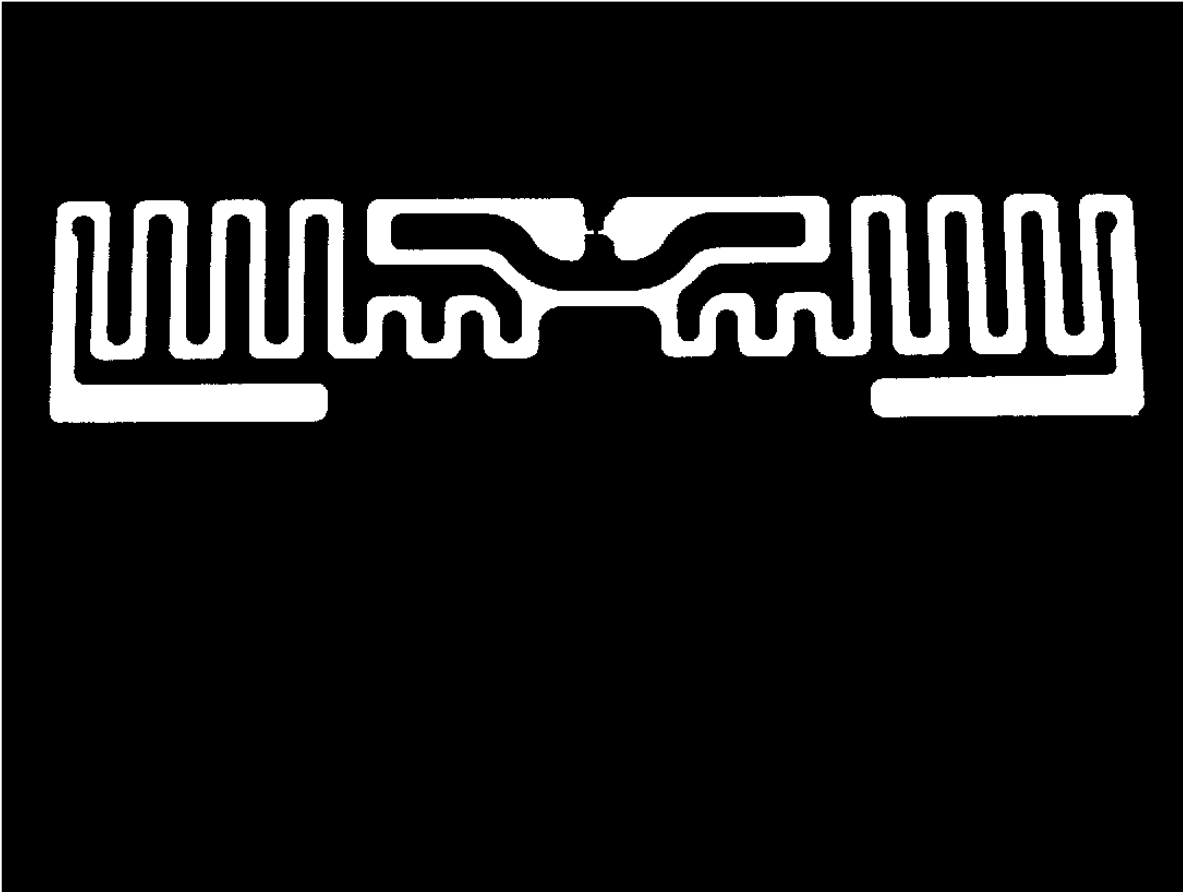


Boundary Clean-up

Read Image, Create Mask and Visualize

The image of the RFID tag is imported into the workspace and the boundary is generated by using the exported code from the Image Segmenter app.

```
I = imread('IMG_2151.JPG');  
BWf = createMask_2151(I);  
figure  
imshow(BWf)
```



Calculate Boundaries in Cartesian Space

For performing full-wave analysis on this structure the next step is to convert the pixel space representation of the boundary to a cartesian space representation. To do this we extract the maximum and minimum pixel indices in the x, y dimensions and scale it based on overall tag dimensions in terms of its length and width.

```

B = bwboundaries(BWf);
xmax = max(B{1}(:,1));
xmin = min(B{1}(:,1));
ymax = max(B{1}(:,2));
ymin = min(B{1}(:,2));

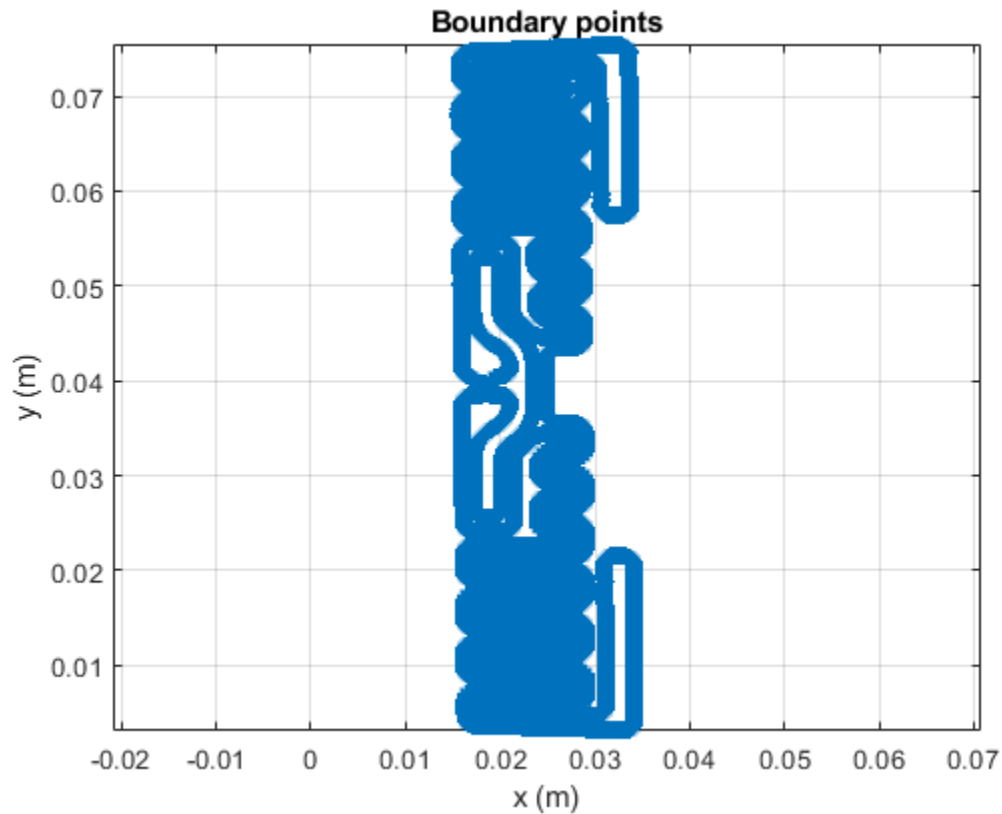
% Scale per pixel based on tag dimensions
L = 18.61e-3;
W = 72.27e-3;
LperColpixel = L/(xmax-xmin);
WperRowpixel = W/(ymax-ymin);
Bp = B;
for i = 1:length(Bp)
    Bp{i} = [Bp{i}(:,1).*LperColpixel Bp{i}(:,2).*WperRowpixel zeros(size(Bp{i},1),1)];
end

```

```

p = cell2mat(Bp);
x = p(:,1);
y = p(:,2);
figure
plot(x,y,'*')
grid on
axis equal
xlabel('x (m)')
ylabel('y (m)')
title('Boundary points')

```



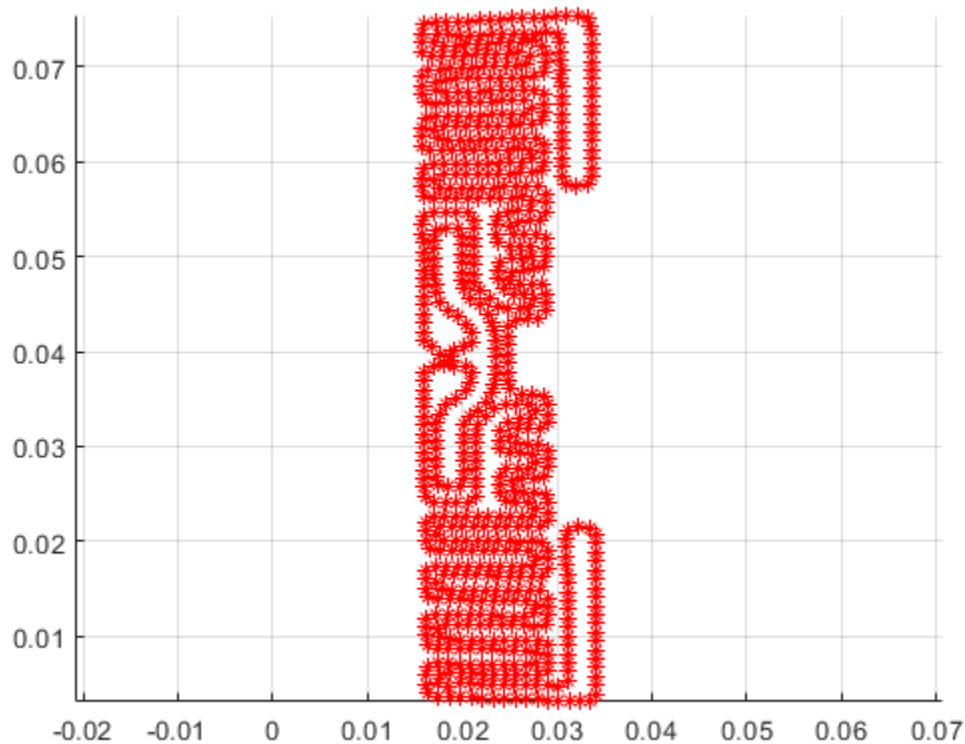
Reduce Boundary Points

The boundary has 28000 points, and this will result in a very large mesh size. Downsample this boundary by a factor of 39. The downsample factor was chosen since it still represented the boundary details accurately based on a simple visual inspection.

```

D = 39;
xD = x(1:D:end);
yD = y(1:D:end);
BpD{1} = Bp{1}(1:D:end,:);
figure
hold on
plot(xD,yD,'r*')
shg
grid on
axis equal

```



Create Layer For PCB Stack

Create a shape from the boundary

```
pol = antenna.Polygon('Vertices', BpD{1});
```

Creating the Antenna Feed The feed region of the tag still has some sharp artifacts in the boundary. This must be cleaned up prior to defining the feed. We use a boolean subtract operation is done by creating a rectangle to remove this artifact.

```
rect1 = antenna.Rectangle('Length', 5e-3, 'Width', 2e-3, ...
    'Center', [0.019 0.0392]);
```

The final step, is to define the feeding strip. Add a feed in the form of a rectangle.

```
rect2 = antenna.Rectangle('Length', 0.25e-3, 'Width', 2e-3, ...
    'Center', [0.0185 0.0392]);
```

Resultant shape

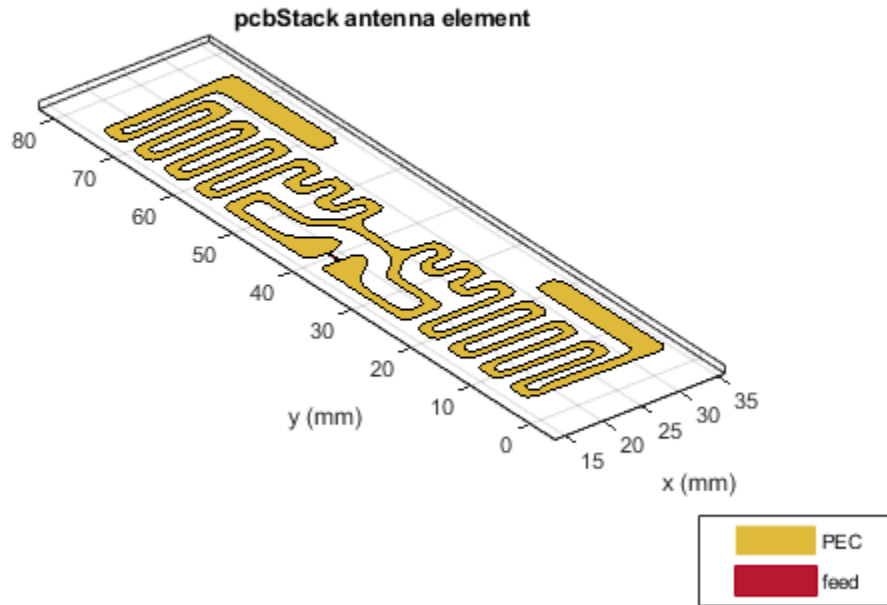
```
final_shape = pol-rect1+rect2;
```

Create a PCB Stack

Assign the resultant shape generated as the layer for pcb stack. FeedLocation is specified appropriately so that it lies on this layer

```
p = pcbStack;
p.Layers = {final_shape};
```

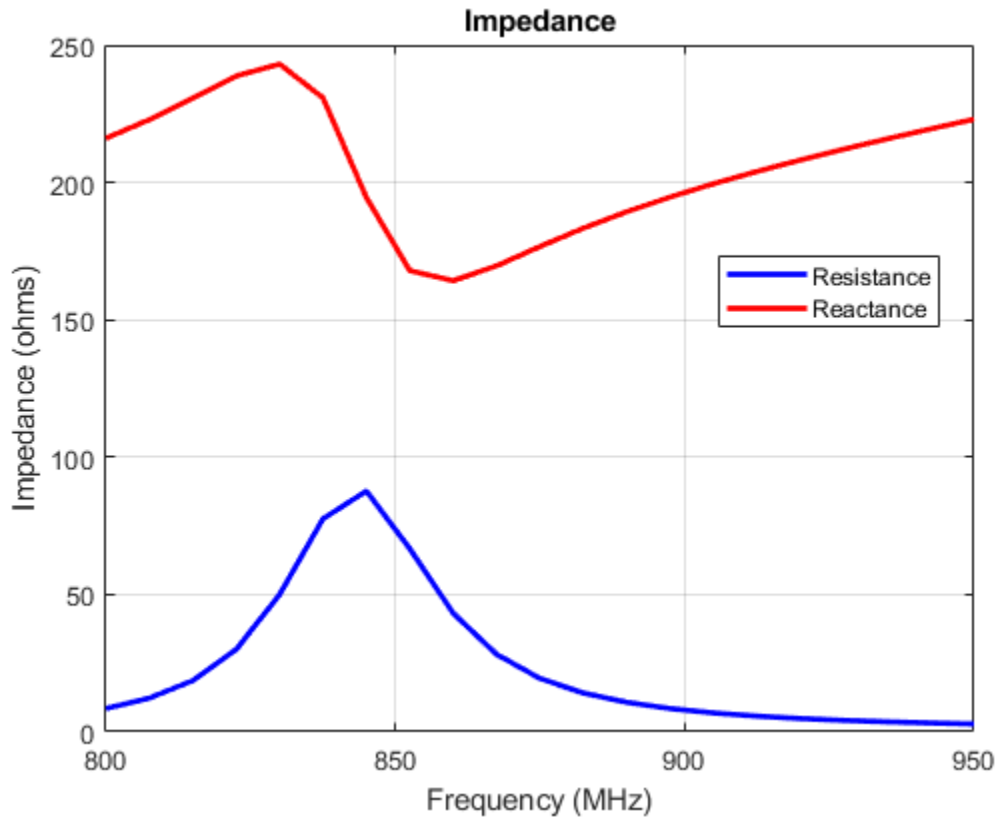
```
p.BoardShape = antenna.Rectangle('Length',1,'Width',1);  
p.FeedLocations = [0.0185 0.0392 1 ];  
p.FeedDiameter = 1.25e-4;  
show(p);
```



Port Analysis - Impedance Behavior vs. Frequency

Determine the port characteristics of this antenna by executing an impedance analysis over a coarse sampled frequency range. The tag is expected to operate in the UHF band, between 800 - 900 MHz. Our frequency range will extend slightly past 900 MHz.

```
f_coarse = linspace(0.8e9,0.95e9,21);  
figure  
impedance(p,f_coarse)
```



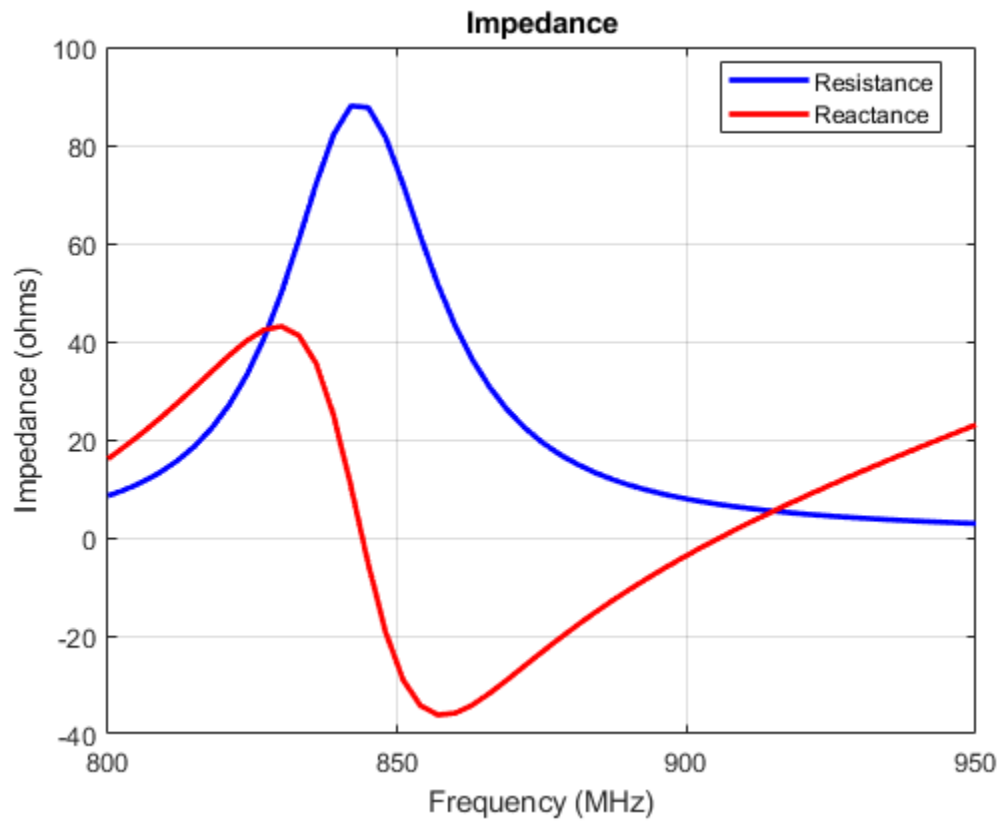
Tuning the Tag for Resonance

The tag is inductive and has a good resistive component at approximately 854 MHz. Moreover the reactance shows the classic parallel resonance curve around that frequency. Typically, the input impedance of the chip would be complex, to match to the tag. Use Load property on the antenna to cancel the inductive component. Since the reactance is about 200Ω create a load with reactance of -200Ω and add it to the antenna model.

```
X = -1i*200;
zl = lumpedElement;
zl.Impedance = X;
p.Load = zl;
```

Recalculate impedance With the load in place at the feed, the inductive part of the reactance should be canceled at 854 MHz. Confirm this by analyzing the impedance over a fine frequency range. The reactance at 854 MHz should be approximately 0 ohms.

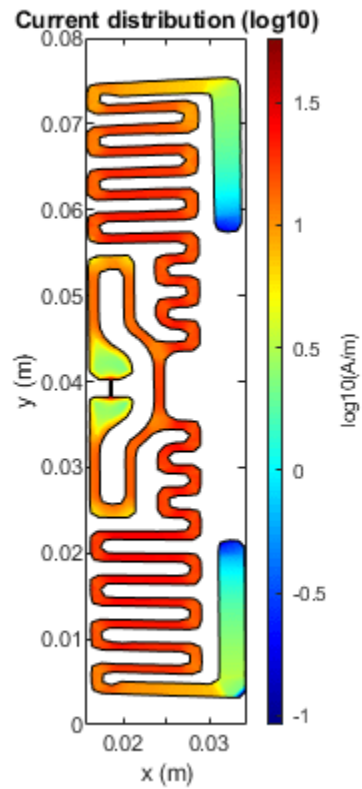
```
f_fine = linspace(0.8e9,0.95e9,51);
figure
impedance(p,f_fine)
```



Surface Analysis - Current behavior at Center Frequency

At the center frequency visualize the current distribution on the antenna surface.

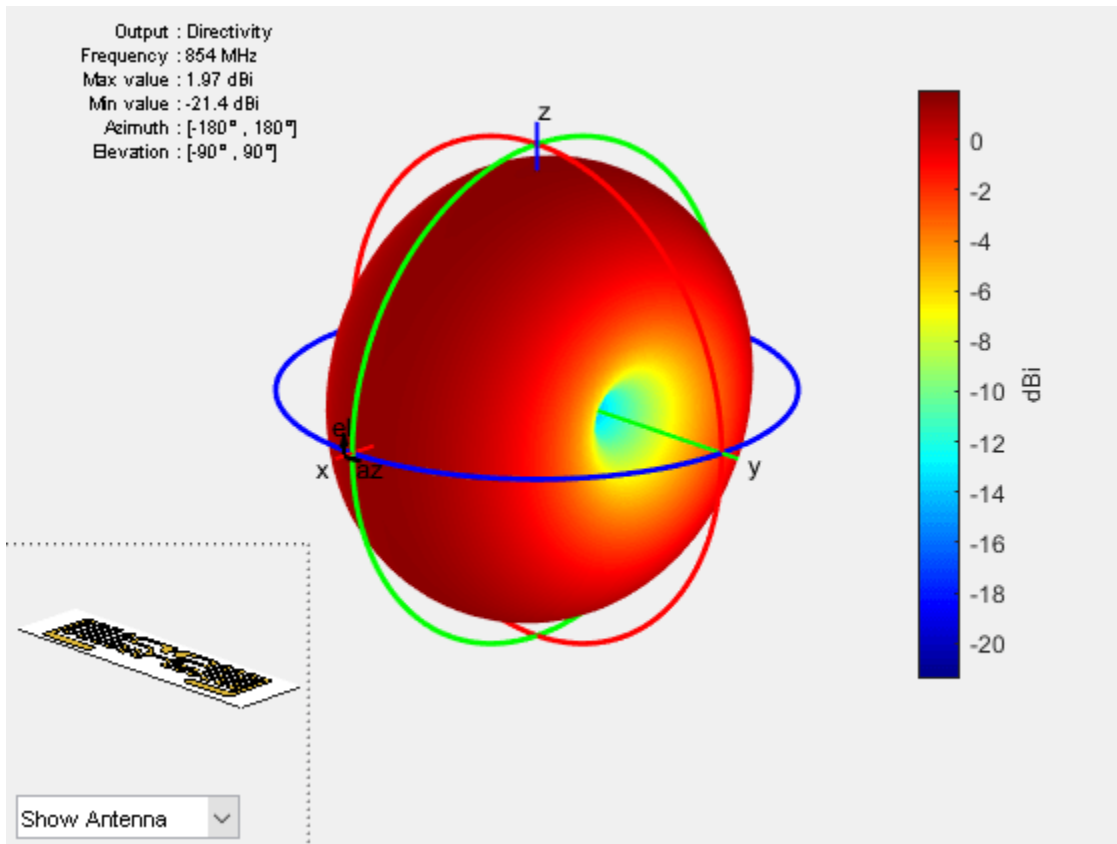
```
figure
current(p,854e6,'scale','log10');
view(0,90);
```



Field Analysis - Pattern at Center Frequency

RFID tags typically have an omnidirectional far-field pattern in one plane. Visualize the far-field radiation pattern of the tag.

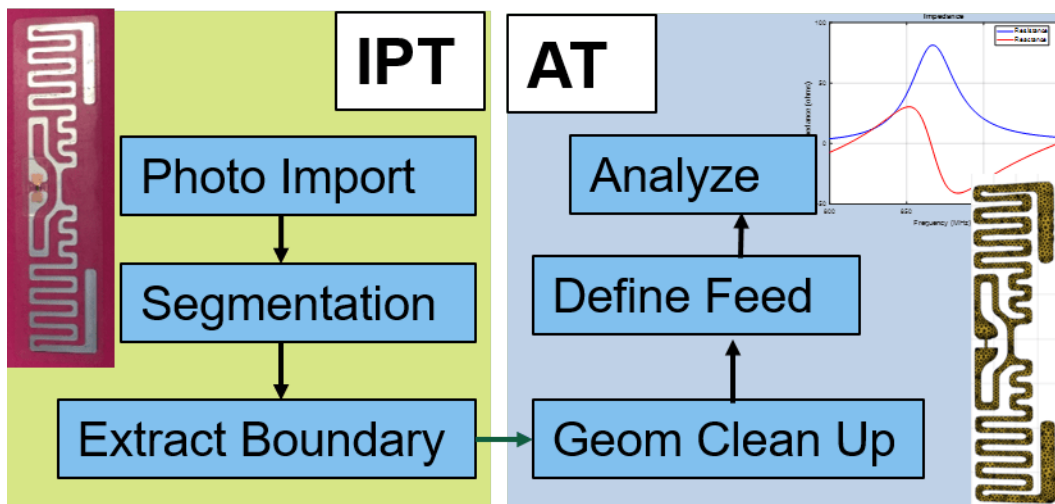
```
figure  
pattern(p,854e6)
```

The tag has a gain of approximately 2dBi at 854 MHz.

Conclusion

A procedure for identifying the antenna boundary from a photograph, conversion into a geometric model of the antenna and its subsequent full-wave analysis has been detailed in this example. These steps are graphically depicted as shown:



The antenna performance characteristics obtained from this procedure are good enough for further integration into a larger system simulation.

See Also

“Design, Analysis, and Prototyping of Microstrip-Fed Wide-Slot Antenna” on page 5-356

Visualize Antenna Coverage Map and Communication Links

This example shows how to calculate and visualize signal strength between a transmitter and multiple receivers. The visualizations include an area coverage map and colored communication links. The example also shows selection of a directional antenna in order to achieve a communication link to a specific location.

Define Transmitter Site

```
% Define transmitter site at MathWorks (3 Apple Hill Dr, Natick, MA)
fq = 6e9; % 6 GHz
tx = txsite("Name","MathWorks", ...
    "Latitude",42.3001, ...
    "Longitude",-71.3504, ...
    "Antenna",design(dipole,fq), ...
    "AntennaHeight",60, ... % Units: meters
    "TransmitterFrequency",fq, ... % Units: Hz
    "TransmitterPower",15); % Units: Watts
```

Define Receiver Sites

```
% Define receiver sites in several surrounding towns and cities
rxNames = [...
    "Boston, MA","Lexington, MA","Concord, MA","Marlborough, MA", ...
    "Hopkinton, MA","Holliston, MA","Foxborough, MA","Quincy, MA"];

rxLocations = [...
    42.3601 -71.0589; ... % Boston
    42.4430 -71.2290; ... % Lexington
    42.4604 -71.3489; ... % Concord
    42.3459 -71.5523; ... % Marlborough
    42.2287 -71.5226; ... % Hopkinton
    42.2001 -71.4245; ... % Holliston
    42.0654 -71.2478; ... % Foxborough
    42.2529 -71.0023]; % Quincy

% Define receiver sensitivity. Sensitivity is the minimum signal strength in
% power that is necessary for the receiver to accurately detect the signal.
rxSensitivity = -90; % Units: dBm

rxs = rxsite("Name",rxNames, ...
    "Latitude",rxLocations(:,1), ...
    "Longitude",rxLocations(:,2), ...
    "Antenna",design(dipole,tx.TransmitterFrequency), ...
    "ReceiverSensitivity",rxSensitivity); % Units: dBm
```

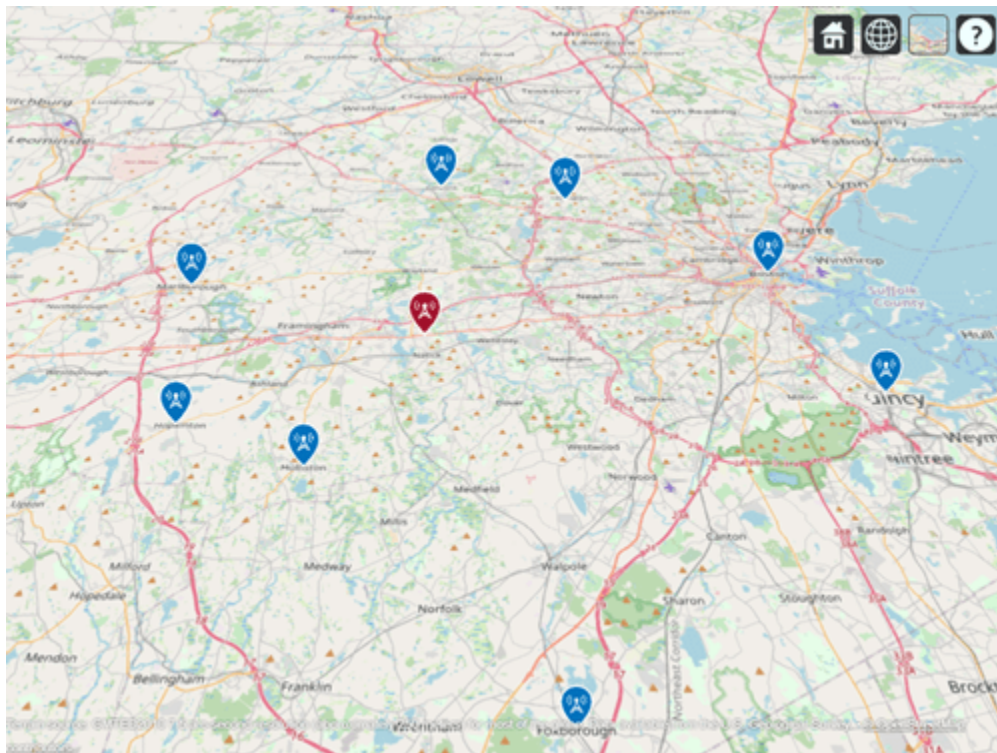
Show Sites on a Map

Show the transmitter and receiver sites on a map. You can display information about a site by clicking on a marker.

```
viewer = siteviewer;
show(tx)
show(rxs)
```

Set the map imagery by using the `Basemap` property. Alternatively, open the map imagery picker in Site Viewer by clicking the second button from the right. Select "OpenStreetMap" to see streets and labels on the map. Rotate the view to show an overhead perspective.

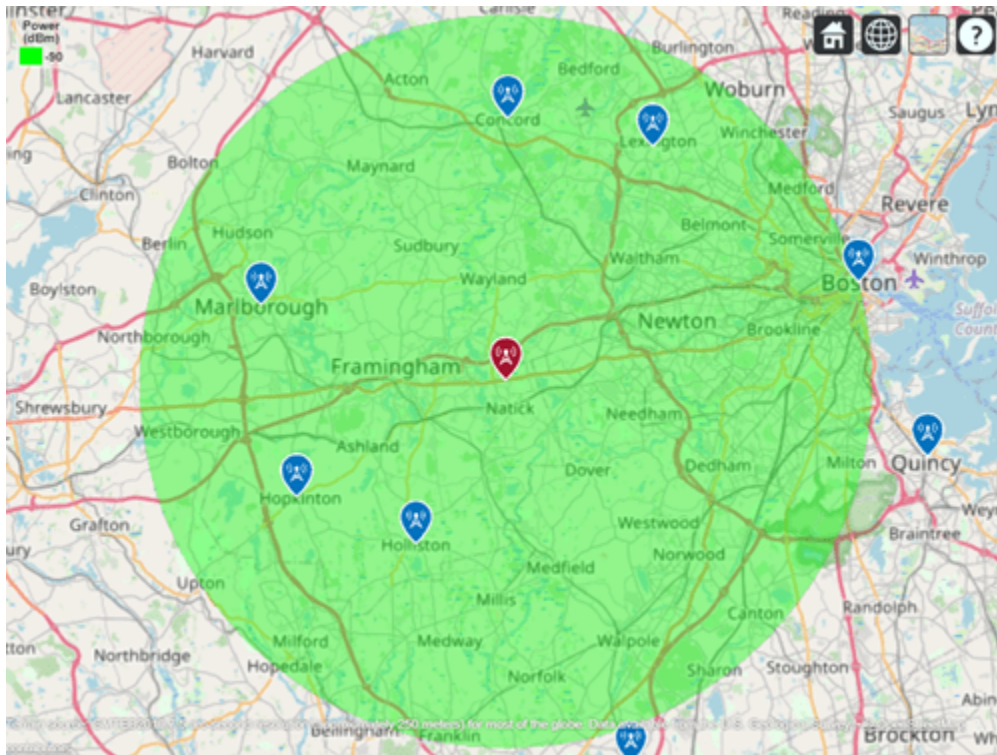
```
viewer.Basemap = "openstreetmap";
```



Display Idealized Coverage Map using Dipole Antenna

Display coverage map. A coverage map shows the geographic area where a receiver will obtain good reception, which is where transmitted signal strength meets or exceeds the receiver's sensitivity. Transmitted signal strength in power (dBm) is computed using a free-space propagation model, which disregards terrain, obstacles, and atmospheric effects. As a result, the coverage map shows idealized coverage area in the absence of any path loss impairments beyond free space loss.

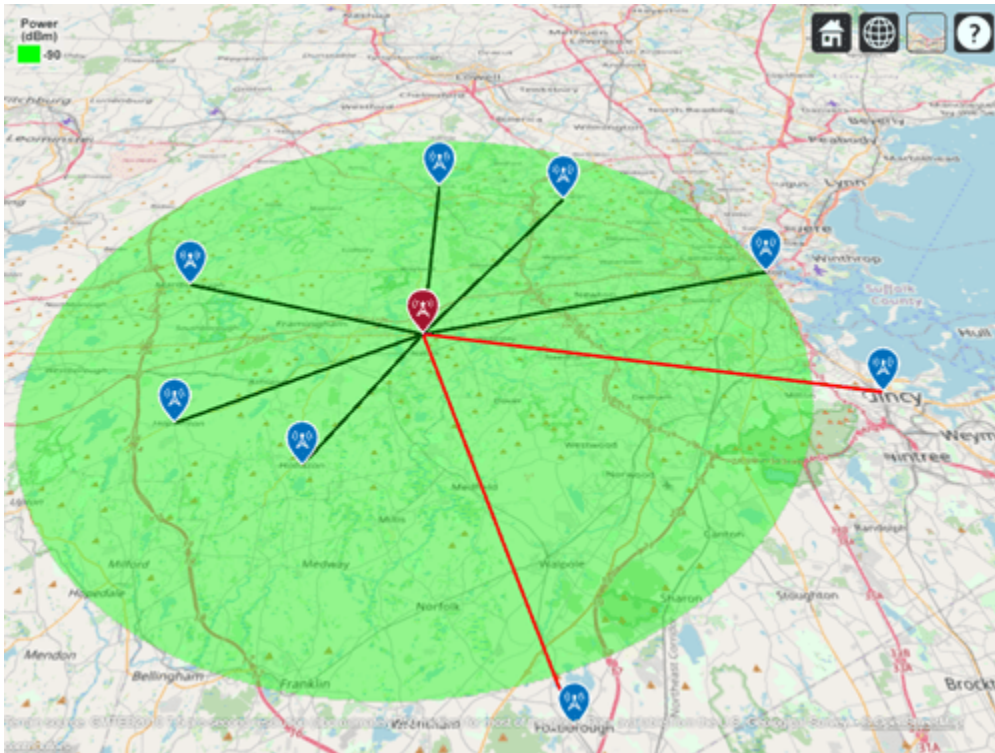
```
coverage(tx, "freespace", ...  
        "SignalStrengths", rxSensitivity)
```



Plot Communication Links using Dipole Antenna

Plot communication links on the map. Red links appear where the receiver is outside of the coverage zone, and green links appear where the receiver is within the coverage zone. Link lines may be clicked to display link statistics. To contrast the colors of the coverage zone and successful links, specify the color of successful links as dark green.

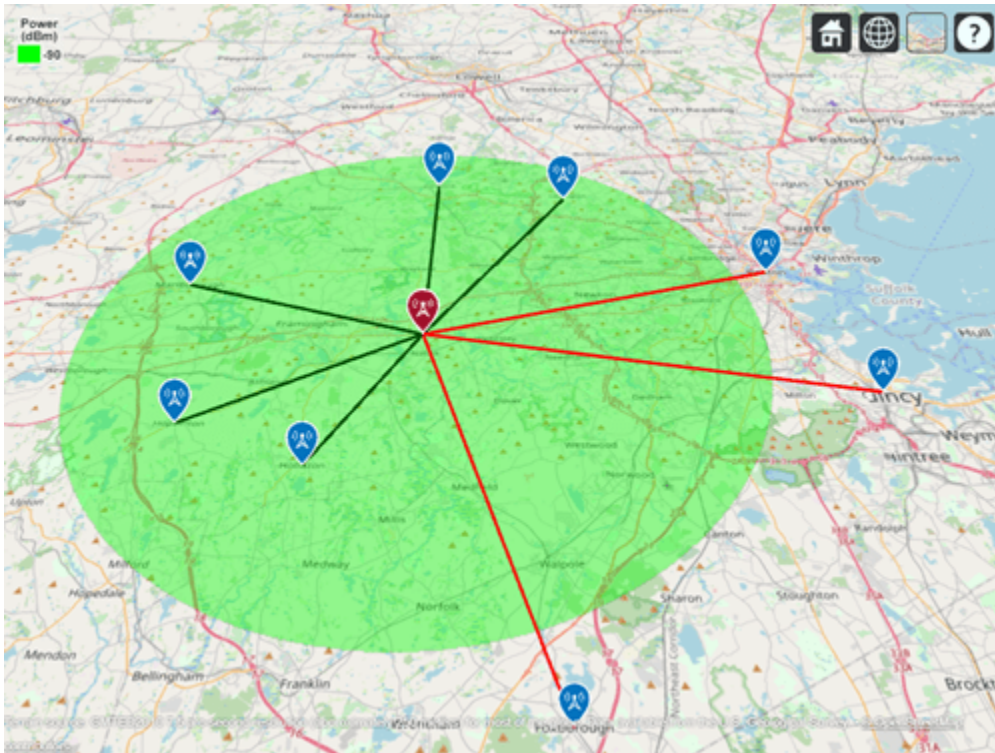
```
sc = [0 0.3 0];
link(rxs,tx,"freespace","SuccessColor",sc)
```



Use Rain Propagation Model

Update the coverage map and links to include path loss due to rain. Note that Boston, MA is no longer inside the coverage zone.

```
coverage(tx,"rain","SignalStrengths", rxSensitivity)  
link(rxs,tx,"rain","SuccessColor",sc)
```



Define Directional Antenna

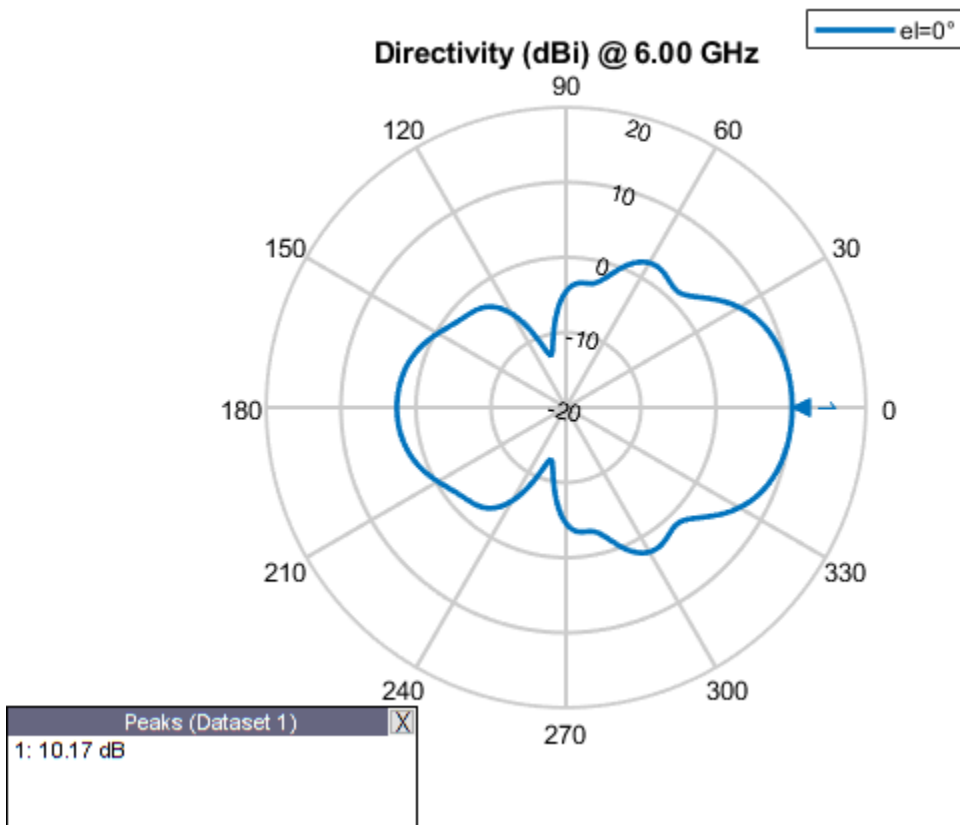
The dipole antenna transmitter results in a few receiver sites outside of the coverage zone, including the receiver in Boston, MA. Now assume a requirement of the transmitter is to achieve a communication link with Boston. Define a directional antenna that can increase antenna gain in that direction.

```
% Define Yagi-Uda antenna designed for transmitter frequency
yagiAnt = design(yagiUda,tx.TransmitterFrequency);

% Tilt antenna to direct radiation in XY-plane (i.e. geographic azimuth)
yagiAnt.Tilt = 90;
yagiAnt.TiltAxis = "y";

f = figure;

% Show directivity pattern
patternAzimuth(yagiAnt,tx.TransmitterFrequency)
```



```
%Close the previous figure
if (isvalid(f))
    close(f);
end
```

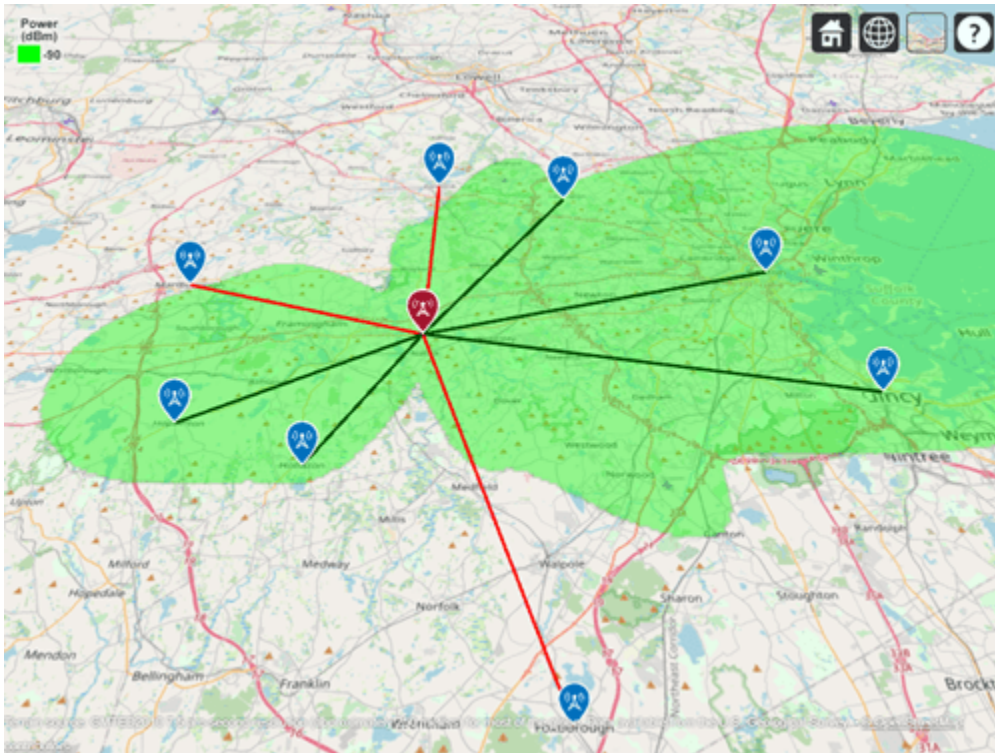
Display Coverage Map using Yagi-Uda Antenna

Update the coverage map and links. Boston is now within the coverage zone, but communication links with receivers in other directions are lost.

```
% Update transmitter antenna
tx.Antenna = yagiAnt;

% Point main beam toward Boston, MA by assigning azimuth angle between
% transmitter location and Boston receiver location
tx.AntennaAngle = angle(tx, rx(1));

% Update visualizations, using "rain" propagation model
coverage(tx,"rain","SignalStrengths",rxSensitivity)
link(rxs,tx,"rain","SuccessColor",sc)
```

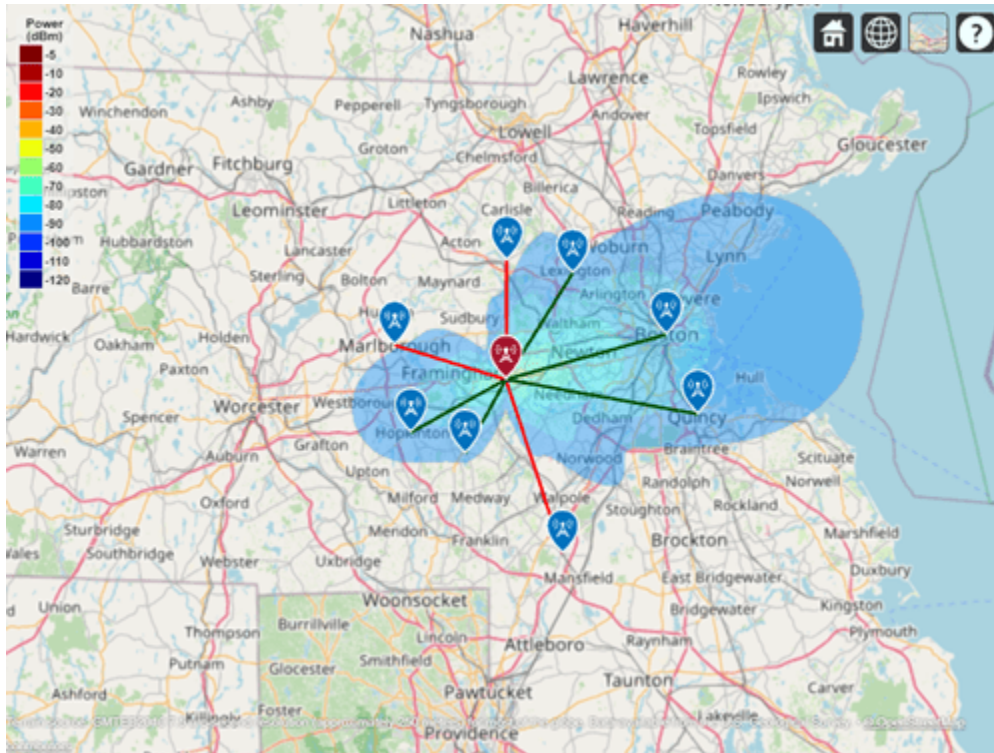



Display Contoured Coverage Map using Multiple Signal Strengths

When a single signal strength is specified, the coverage map is green for the coverage region. Specify multiple signal strengths to generate a coverage map with contours for different signal levels.

```
% Define signal strengths from sensitivity to -60 dB  
sigStrengths = rxSensitivity:5:-60;
```

```
% Update coverage map  
coverage(tx, "rain", "SignalStrengths", sigStrengths)
```



See Also

Functions

[coverage](#) | [design](#) | [link](#)

Objects

[txsite](#) | [rxsite](#) | [siteviewer](#)

Related Examples

- “Urban Link and Coverage Analysis Using Ray Tracing” on page 5-612
- “VHF/UHF Biconical Antenna for Testing Applications” on page 5-714
- “Discone Antenna for TV Broadcasting System” on page 5-595

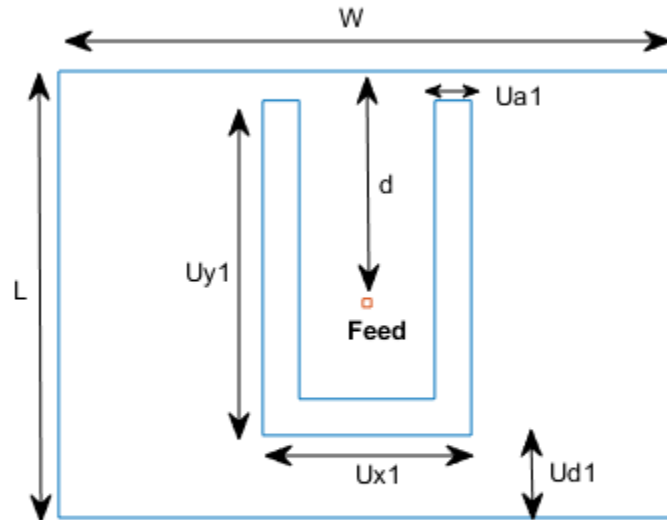
Modeling and Analysis of Single Layer Multi-band U-Slot Patch Antenna

The standard rectangular microstrip patch is a narrowband antenna and provides 6-8 dBi Gain with linear polarization. This example based on the work done in [1],[2], models a broadband patch antenna using a slot in the radiator and develops a dual-band and a tri-band variation from it. In the process, the single wide response has been split into multiple narrow band regions catering to specific bands in the WiMAX standard. These patch antennas have been probe-fed.

Building the Single U-Slot Patch

Define parameters The basic U-slot patch antenna consists of a rectangular patch radiator within which a U-shaped slot has been cut out. As discussed in [1], the patch itself is on an air substrate and thick so as to enable higher bandwidths to be achieved. The presence of the slot structure achieves additional capacitance within the structure which combines with the inductance of the long probe feed to create a double resonance within the band. The geometry parameters based on [2] are defined and shown in a drawing below.

$L = 26e-3;$
 $W = 35.5e-3;$
 $Uy1 = 19.5e-3;$
 $Ux1 = 12e-3;$
 $Ua1 = 2.1e-3;$
 $Ud1 = 4.8e-3;$
 $d = 13.5e-3;$
 $h = 6e-3;$



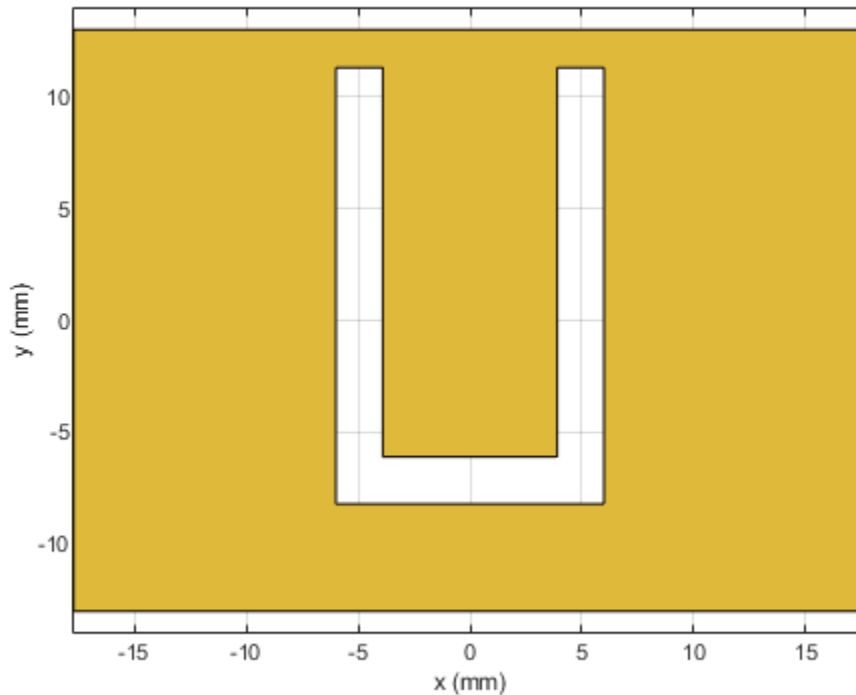
Define radiator shape - Single U-slot

Use the rectangle shape primitives in Antenna Toolbox™ to create the U-slot patch radiator shape. Boolean subtraction operation is used among the shape primitives for this purpose.

```

N1 = 2; %50;
N2 = 2; %10;
s = antenna.Rectangle('Length',W,'Width',L,'NumPoints',60);
h1 = antenna.Rectangle('Length',Ua1,'Width',Uy1,'NumPoints',[N2 N1 N2 N1],...
    'Center',[-Ux1/2 + Ua1/2, -L/2 + Ud1 + Uy1/2]);
h2 = antenna.Rectangle('Length',Ua1,'Width',Uy1,'NumPoints',[N2 N1 N2 N1],...
    'Center',[Ux1/2 - Ua1/2, -L/2 + Ud1 + Uy1/2]);
h3 = antenna.Rectangle('Length',Ux1,'Width',Ua1,'NumPoints',[N1 N2 N1 N2],...
    'Center',[0,-L/2 + Ud1 + Ua1/2]);
Uslot_patch = s-h1-h2-h3;
figure
show(Uslot_patch)

```



Define ground shape

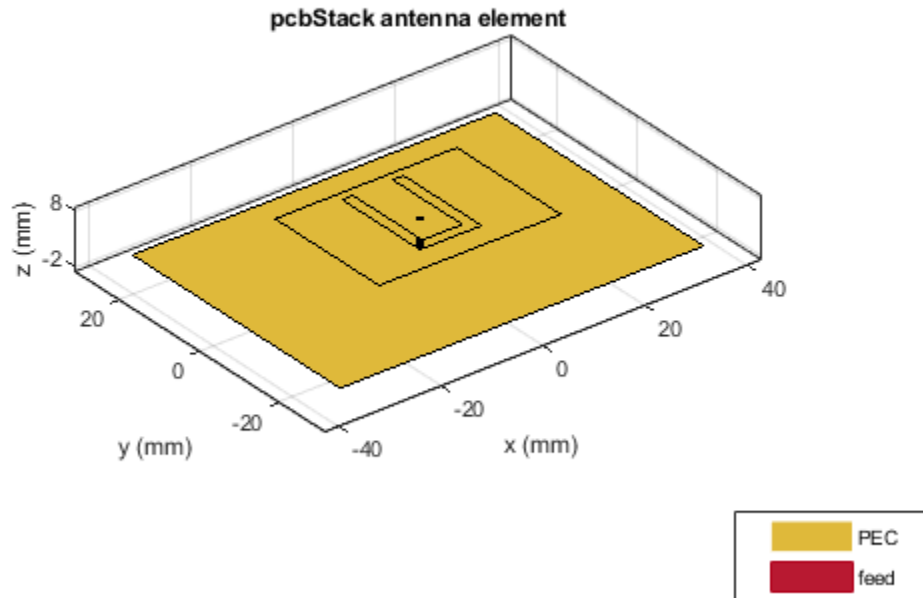
Create the ground plane shape for the antenna. The groundplane in this case is rectangular and 71 mm x 52 mm in size.

```
Lgp = 71e-3;
Wgp = 52e-3;
p2 = antenna.Rectangle('Length',Lgp,'Width',Wgp,'NumPoints',10);
```

Define stack

Use the pcbStack to define the metal and dielectric layers and the feed for the single U-slot patch antenna. The layers are defined top-down. In this case, the top-most layer is a metal layer defined by the U-slot patch shape. The second layer is a dielectric material, air in this case, and the third layer is the metal ground plane.

```
d1 = dielectric('Air');
slotPatch = pcbStack;
slotPatch.Name = 'U-Slot Patch';
slotPatch.BoardThickness = h;
slotPatch.BoardShape = p2;
slotPatch.Layers = {Uslot_patch,d1,p2};
slotPatch.FeedLocations = [0 L/2-d 1 3];
slotPatch.FeedDiameter = 0.9e-3;
figure
show(slotPatch)
```

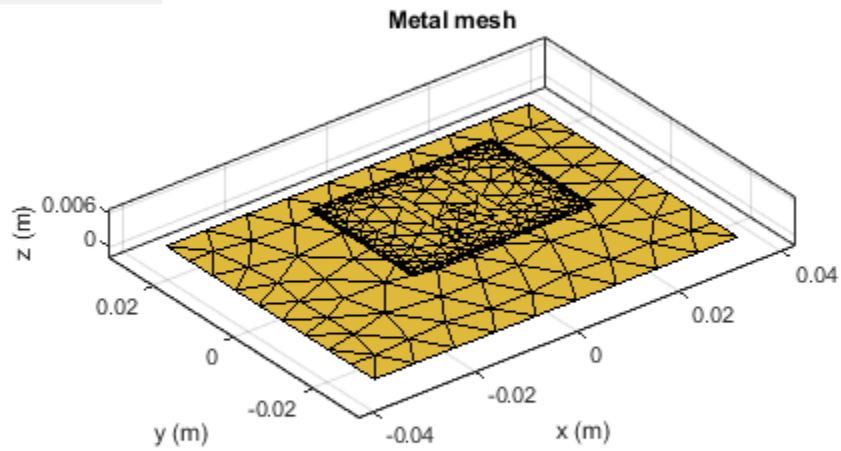


Calculate and Plot Reflection Coefficient

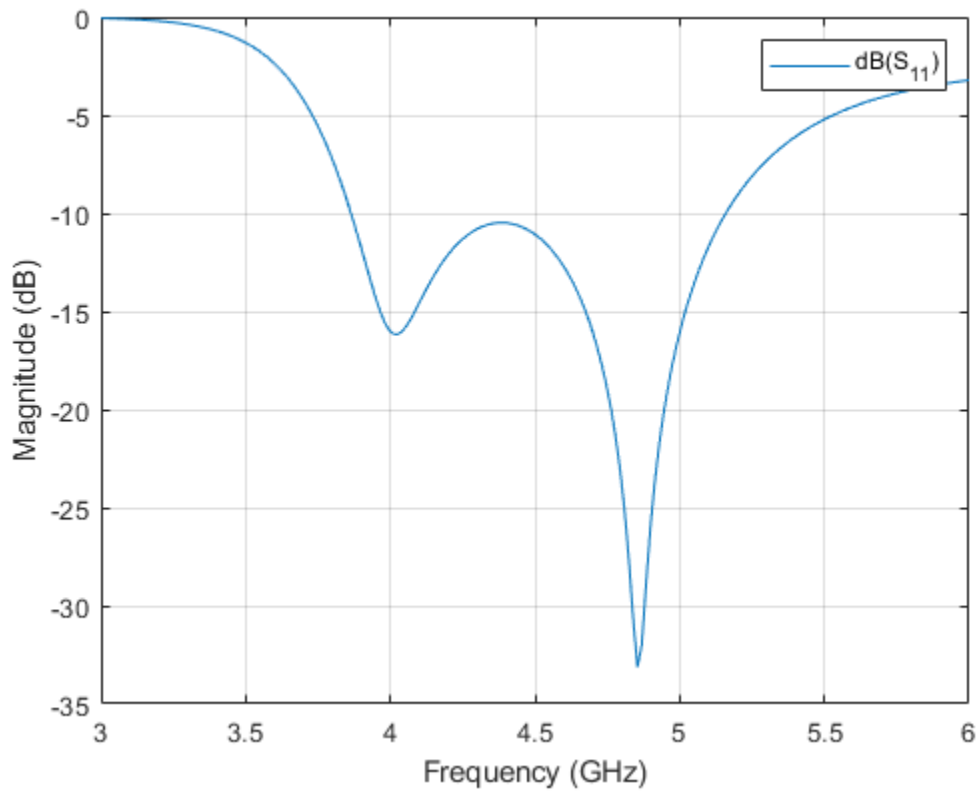
Mesh the structure by using a maximum edge length which is one-tenth the wavelength at the highest frequency of operation which is 6 GHz for this example. Compute and plot the reflection coefficient for this antenna over the band. The reflection coefficient is plotted with a reference impedance of 50 ohms.

```
figure  
mesh(slotPatch, 'MaxEdgeLength', .01, 'MinEdgeLength', .001, 'GrowthRate', 0.7)
```

NumTriangles: 753
NumTetrahedra: 0
NumBasis:
MaxEdgeLength: 0.01
MeshMode: manual



```
freq = linspace(3e9,6e9,200);  
s1 = sparameters(slotPatch,freq);  
s11Fig = figure;  
rfplot(s1,1,1)  
s11Ax = gca(s11Fig);  
hold(s11Ax,'on');
```



Calculate and plot pattern

Plot the radiation pattern for this antenna at the frequencies of best match in the band.

```
figure  
pattern(slotPatch,3.9e9)
```

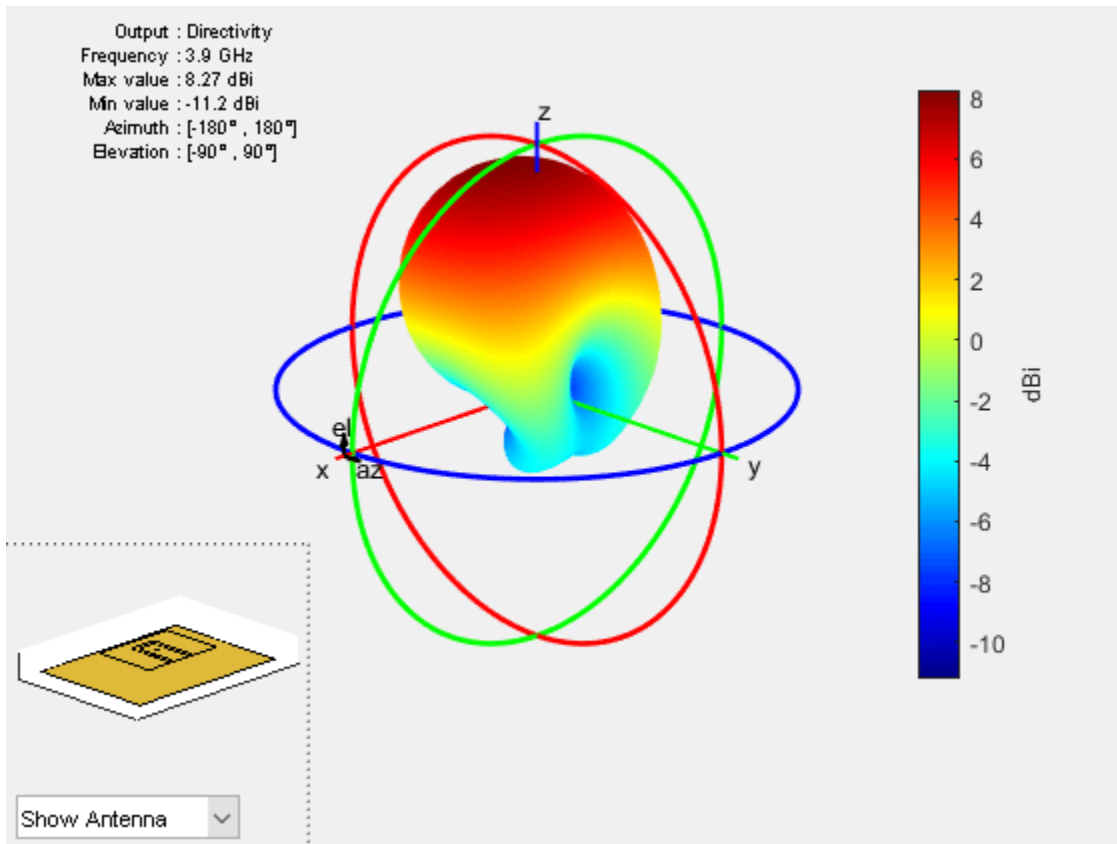
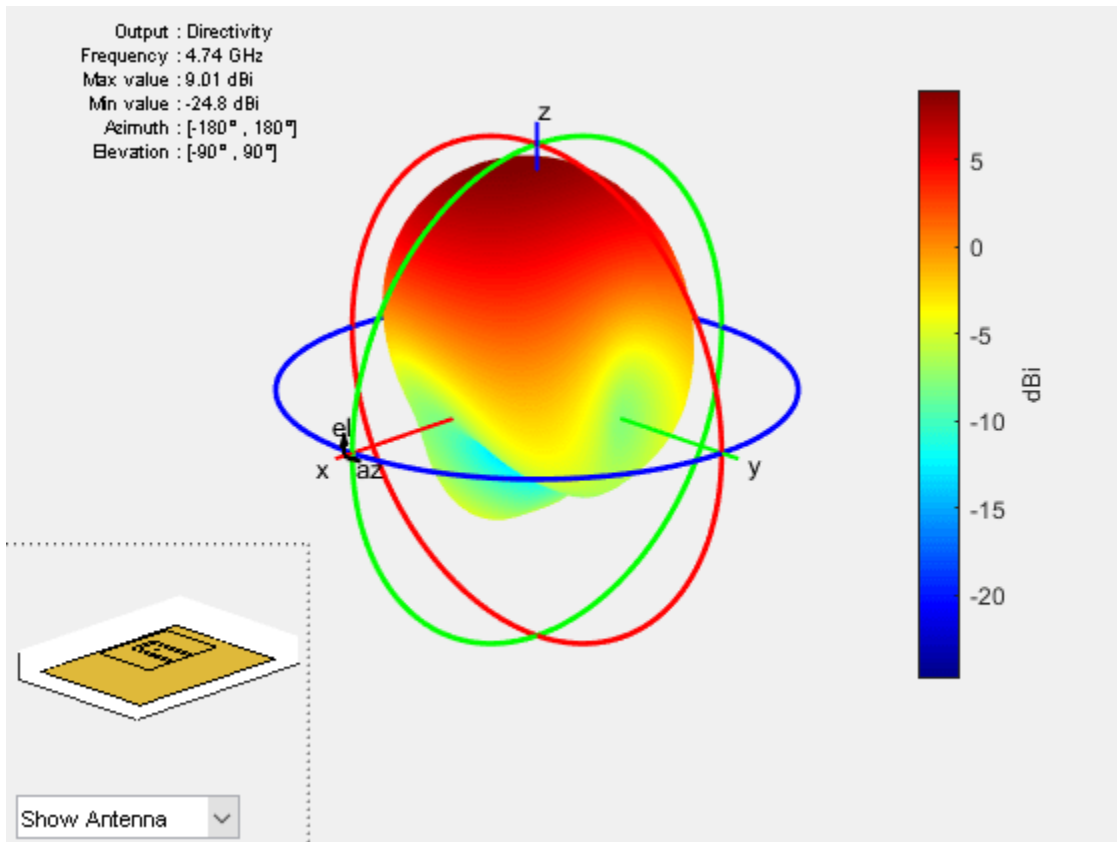



figure
 pattern(slotPatch,4.74e9)



Dual-band U-Slot Patch Antenna

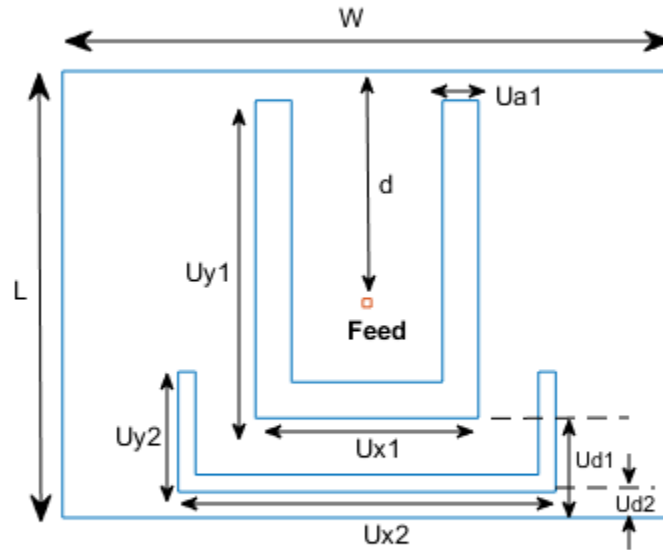
Define Parameters

To achieve dual-band behavior as shown in [1], [2], the double resonance is modified such that the two contributing resonances, i.e. from the patch and from the slot do not merge. To do so the existing slot parameters are adjusted and a second slot is introduced into the structure. The parameters for the double U-slot are listed below as per [2] and a figure annotated with the variables used is shown.

```

Ux1 = 13e-3;
Ux2 = 22e-3;
Uy1 = 18.5e-3;
Uy2 = 7e-3;
Ua2 = 1e-3;
Ud1 = 5.8e-3;
Ud2 = 1.5e-3;

```



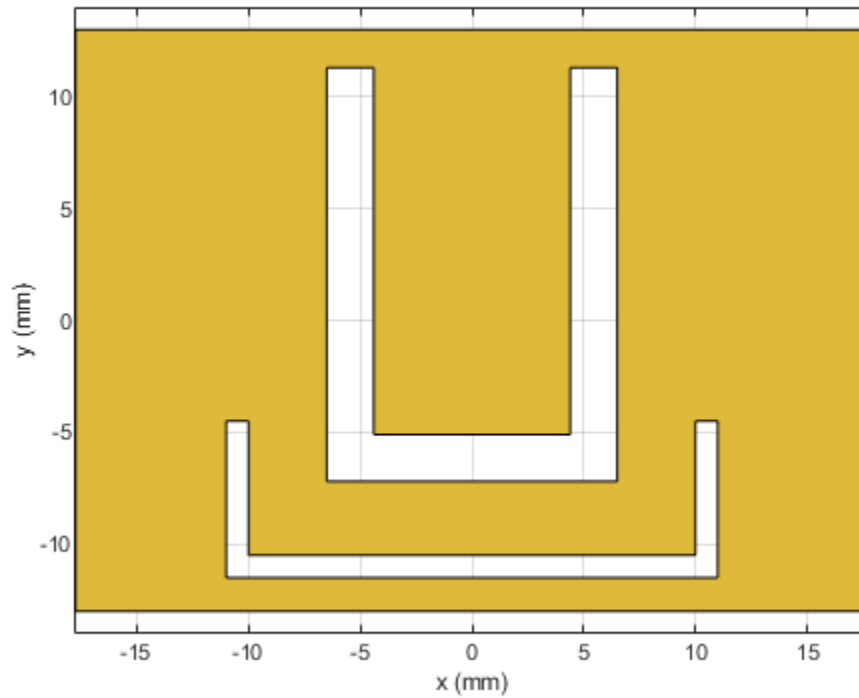
Create Double U-slot radiator

As before use the shape primitives, to create the geometry by using Boolean operations.

```

h1 = antenna.Rectangle('Length',Ua1,'Width',Uy1,'NumPoints',[N2 N1 N2 N1],'Center',[-Ux1/2 + Ua1/2,Uy1/2]);
h2 = antenna.Rectangle('Length',Ua1,'Width',Uy1,'NumPoints',[N2 N1 N2 N1],'Center',[Ux1/2 - Ua1/2,Uy1/2]);
h3 = antenna.Rectangle('Length',Ux1,'Width',Ua1,'NumPoints',[N1 N2 N1 N2],'Center',[0,-L/2 + Ud1]);
Uslot_patch = s-h1-h2-h3;
h4 = antenna.Rectangle('Length',Ua2,'Width',Uy2,'NumPoints',[N2 N1 N2 N1],'Center',[-Ux2/2 + Ua2/2,Uy2/2]);
h5 = antenna.Rectangle('Length',Ua2,'Width',Uy2,'NumPoints',[N2 N1 N2 N1],'Center',[Ux2/2 - Ua2/2,Uy2/2]);
h6 = antenna.Rectangle('Length',Ux2,'Width',Ua2,'NumPoints',[N1 N2 N1 N2],'Center',[0,-L/2 + Ud2]);
DoubleUslot_patch = Uslot_patch-h4-h5-h6;
figure
show(DoubleUslot_patch)

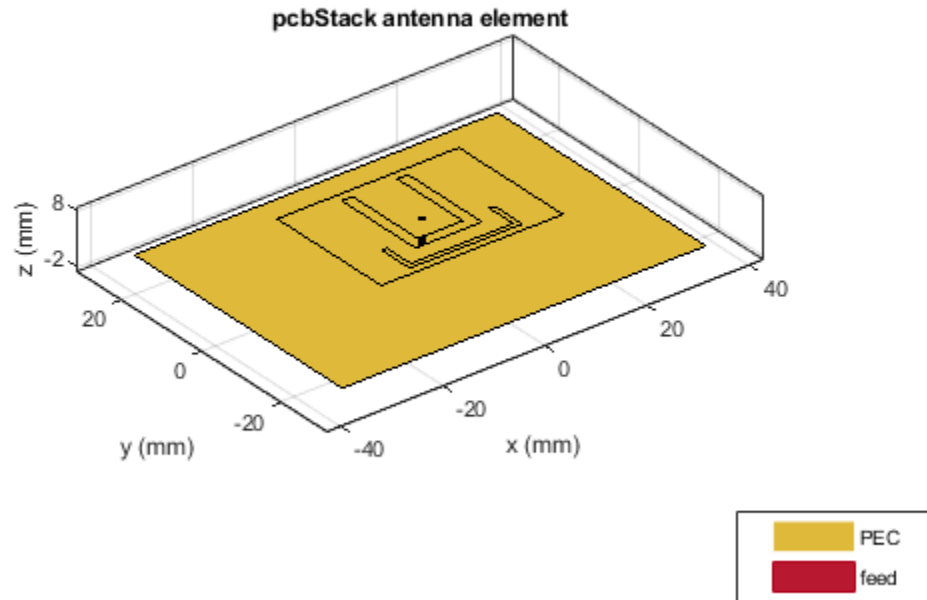
```



Modify Layers in Stack

Modify the existing stack by introducing the new radiator in the Layers property.

```
slotPatch.Layers = {DoubleUsSlot_patch,d1,p2};  
figure  
show(slotPatch)
```



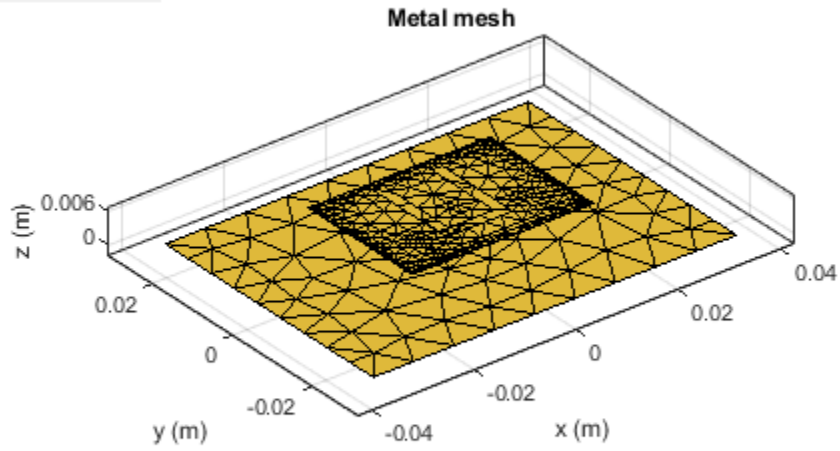
Mesh and Plot Reflection Coefficient

Mesh the structure at the highest frequency of operation and calculate the reflection coefficient.

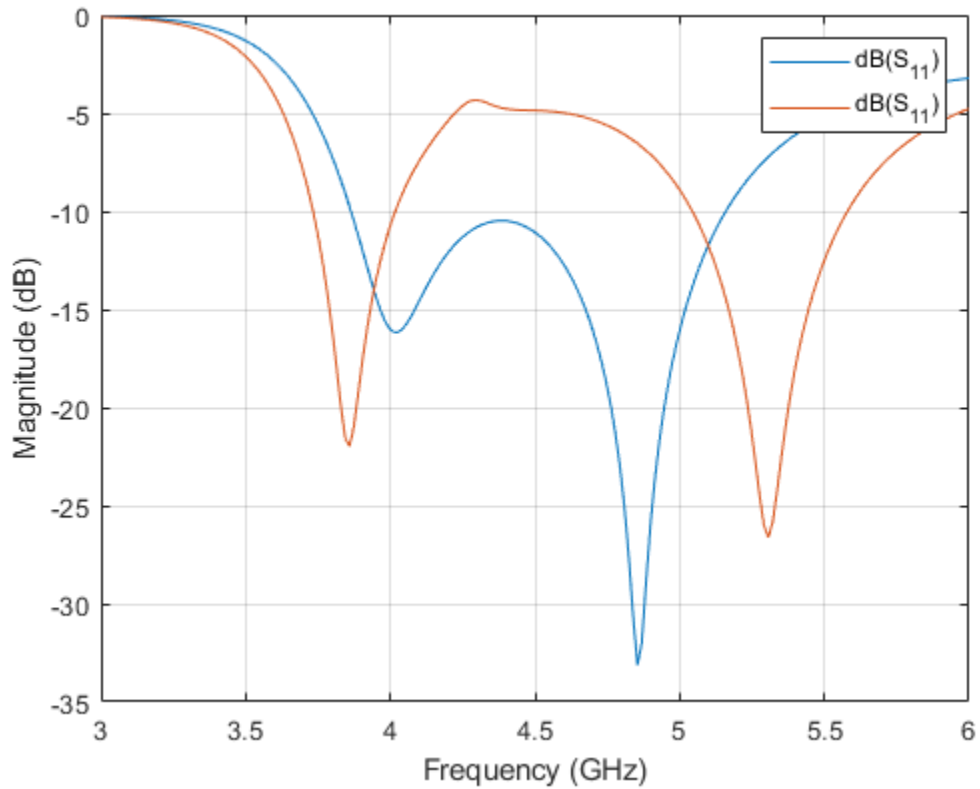
figure

```
mesh(slotPatch, 'MaxEdgeLength', .01, 'MinEdgeLength', .001, 'GrowthRate', 0.5)
```

NumTriangles: 856
NumTetrahedra: 0
NumBasis:
MaxEdgeLength: 0.01
MeshMode: manual



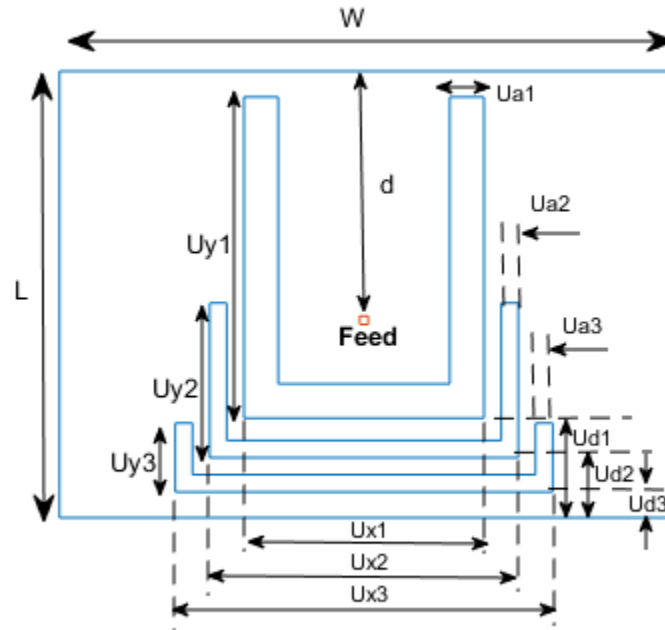
```
s2 = sparameters(slotPatch,freq);  
figure(s11Fig);  
rfplot(s2,1,1);
```



Triple-Band U-slot Patch Antenna Parameters

For triple-band operation a third U-slot is introduced and the existing slot parameters are adjusted. The parameters are shown below based on [2].

$d = 14.5e-3;$
 $Ux1 = 14e-3;$
 $Ux2 = 18e-3;$
 $Ux3 = 22e-3;$
 $Uy1 = 18.7e-3;$
 $Uy2 = 9e-3;$
 $Uy3 = 4e-3;$
 $Ud2 = 3.5e-3;$
 $Ud3 = 1.5e-3;$
 $Ua1 = 2e-3;$
 $Ua3 = 1e-3;$

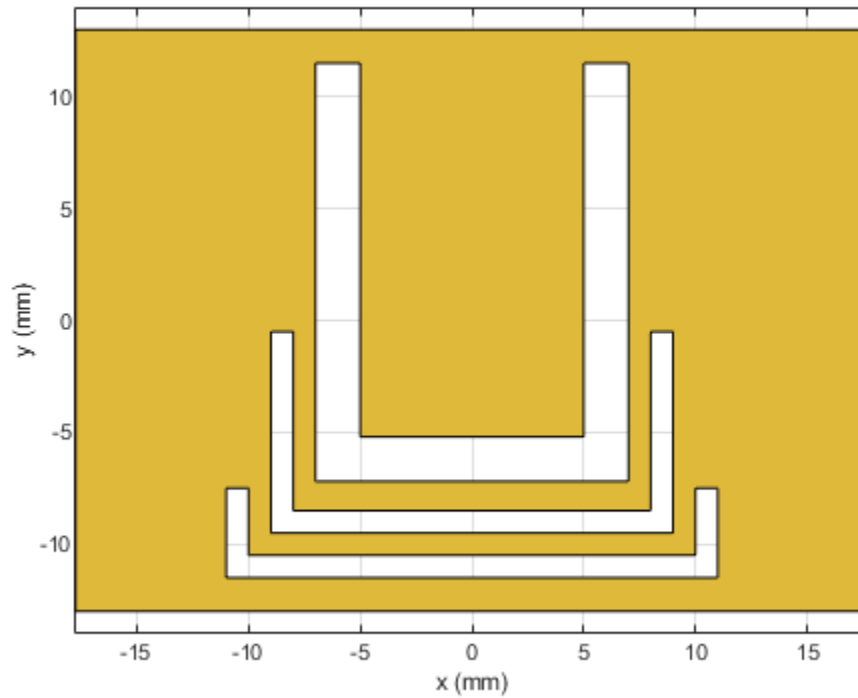


Create Triple U-slot radiator

```

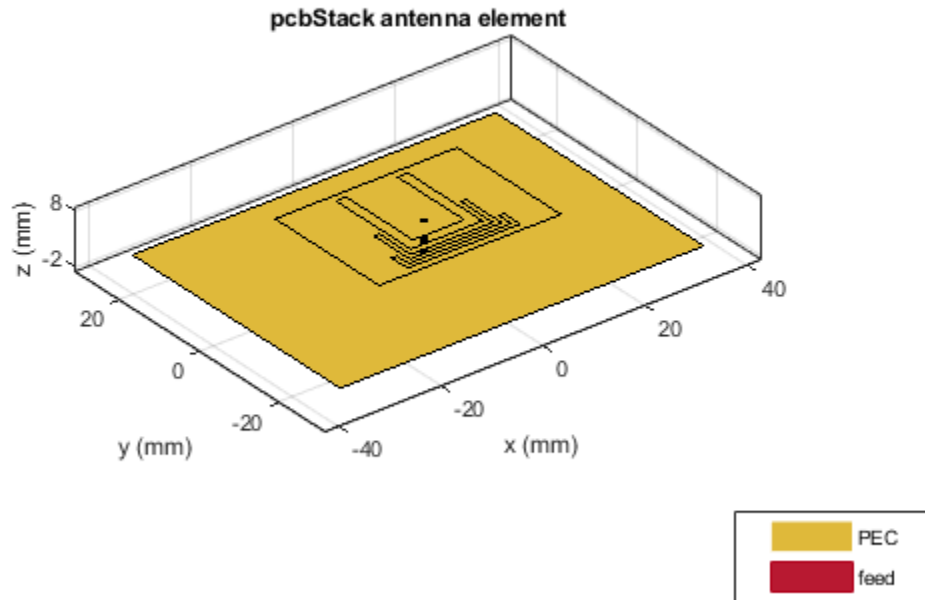
N1 = 20; %50;
N2 = 10; %10;
h1 = antenna.Rectangle('Length',Ua1,'Width',Uy1,'NumPoints',[N2 N1 N2 N1],'Center',[-Ux1/2 + Ua1/2,0]);
h2 = antenna.Rectangle('Length',Ua1,'Width',Uy1,'NumPoints',[N2 N1 N2 N1],'Center',[Ux1/2 - Ua1/2,0]);
h3 = antenna.Rectangle('Length',Ux1,'Width',Ua1,'NumPoints',[N1 N2 N1 N2],'Center',[0,-L/2 + Ud1]);
Uslot_patch = s-h1-h2-h3;
h4 = antenna.Rectangle('Length',Ua2,'Width',Uy2,'NumPoints',[N2 N1 N2 N1],'Center',[-Ux2/2 + Ua2/2,0]);
h5 = antenna.Rectangle('Length',Ua2,'Width',Uy2,'NumPoints',[N2 N1 N2 N1],'Center',[Ux2/2 - Ua2/2,0]);
h6 = antenna.Rectangle('Length',Ux2,'Width',Ua2,'NumPoints',[N1 N2 N1 N2],'Center',[0,-L/2 + Ud2]);
DoubleUslot_patch = Uslot_patch-h4-h5-h6;
h7 = antenna.Rectangle('Length',Ua3,'Width',Uy3,'NumPoints',[N2 N1 N2 N1],'Center',[-Ux3/2 + Ua3/2,0]);
h8 = antenna.Rectangle('Length',Ua3,'Width',Uy3,'NumPoints',[N2 N1 N2 N1],'Center',[Ux3/2 - Ua3/2,0]);
h9 = antenna.Rectangle('Length',Ux3,'Width',Ua3,'NumPoints',[N1 N2 N1 N2],'Center',[0,-L/2 + Ud3]);
TripleUslot_patch = DoubleUslot_patch-h7-h8-h9;
figure
show(TripleUslot_patch)

```

Modify Layers in Stack

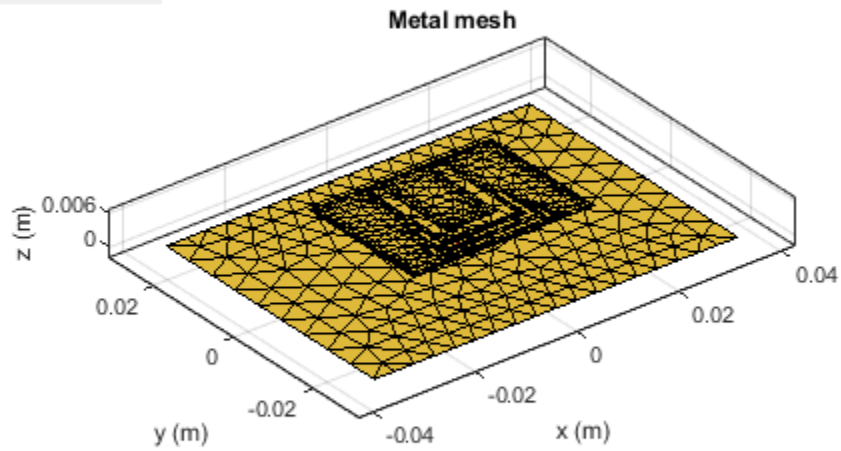
```
slotPatch.Layers = {TripleUslot_patch,d1,p2};  
slotPatch.FeedLocations = [0 L/2-d 1 3];  
figure  
show(slotPatch)
```



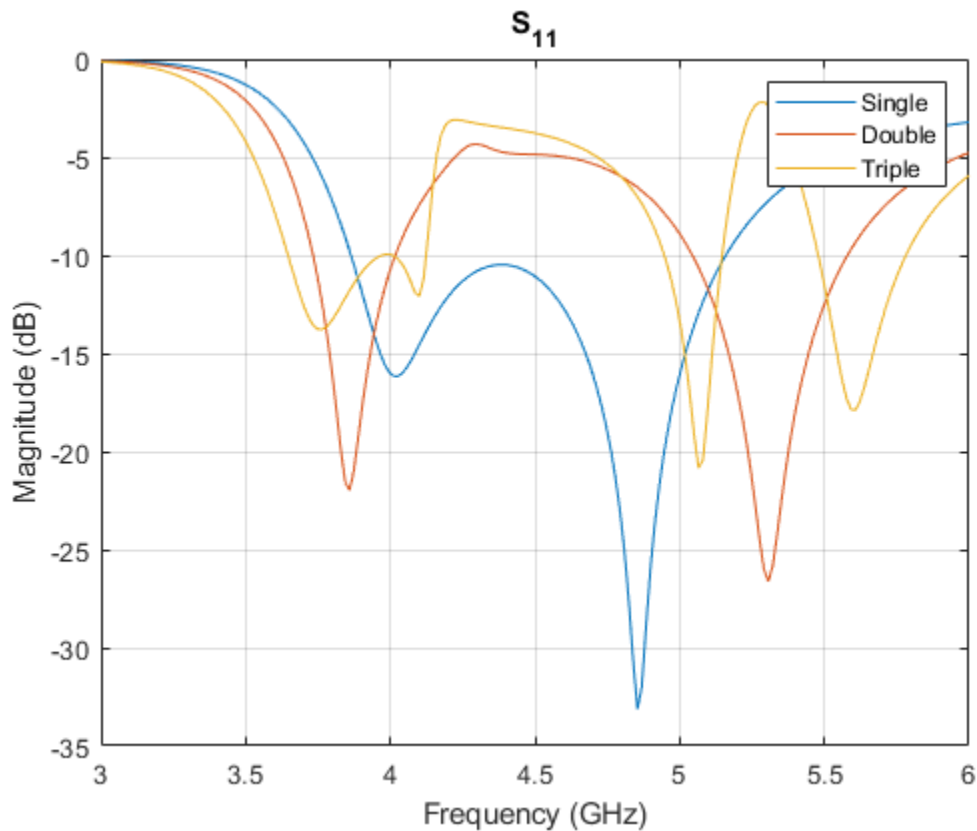
Mesh and Plot Reflection Coefficient

```
figure  
mesh(slotPatch, 'MaxEdgeLength', .005, 'MinEdgeLength', .001, 'GrowthRate', 0.7)
```

NumTriangles: 1665
NumTetrahedra: 0
NumBasis:
MaxEdgeLength: 0.005
MeshMode: manual



```
s3 = sparameters(slotPatch,freq);  
figure(s11Fig)  
rfplot(s3,1,1)  
legend('Single','Double','Triple')  
title('S_1_1')
```



Conclusion

The models of the multi-band single layer U-slot patch antenna as discussed in [1], and [2] have been built and analyzed and agree well with results reported.

Reference

[1] K. F. Lee, S. L. S. Yang and A. Kishk, "The versatile U-slot patch antenna," 2009 3rd European Conference on Antennas and Propagation, Berlin, 2009, pp. 3312-3314.

[2] W. C. Mok, S. H. Wong, K. M. Luk and K. F. Lee, "Single-Layer Single-Patch Dual-Band and Triple-Band Patch Antennas," in IEEE Transactions on Antennas and Propagation, vol. 61, no. 8, pp. 4341-4344, Aug. 2013.

See Also

"Model and Analyze Dual Polarized Patch Microstrip Antenna" on page 5-476

Antenna Array Beam Scanning Visualization on a Map

This example shows how to visualize the changing pattern and coverage map of an antenna array as it scans a sweep of angles. The antenna array is created using Antenna Toolbox™ and Phased Array System Toolbox™. The array is designed to be directional and radiate in the xy-plane to generate a maximum coverage region in the geographic azimuth. Transmitter and receiver sites are created and shown on a map, and the pattern and coverage map are displayed as the antenna array is steered.

Design a Reflector-Backed Dipole Antenna Element

Use Antenna Toolbox to design a reflector-backed dipole antenna element. Design the element and its exciter for 10 GHz, and specify tilt to direct radiation in the xy-plane, which corresponds to the geographic azimuth.

```
% Design reflector-backed dipole antenna element
fq = 10e9; % 10 GHz
myelement = design(reflector,fq);
myelement.Exciter = design(myelement.Exciter,fq);

% Tilt antenna element to radiate in xy-plane, with boresight along x-axis
myelement.Tilt = 90;
myelement.TiltAxis = "y";
myelement.Exciter.Tilt = 90;
myelement.Exciter.TiltAxis = "y";
```

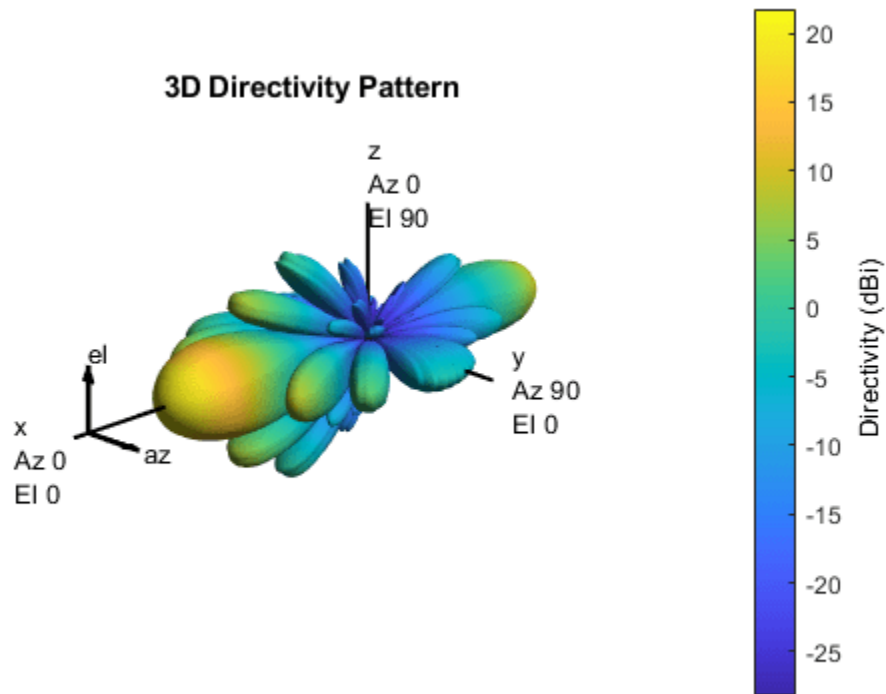
Create a 7-by-7 Rectangular Antenna Array

Use Phased Array System Toolbox to create a 7-by-7 rectangular array from the antenna element. Specify the array normal to direct radiation in the x-axis direction.

```
% Create 7-by-7 antenna array
nrow = 7;
ncol = 7;
myarray = phased.URA("Size",[nrow ncol],"Element",myelement);

% Define element spacing to be half-wavelength at 10 GHz, and specify
% array plane as yz-plane, which directs radiation in x-axis direction
lambda = physconst("lightspeed")/fq;
drow = lambda/2;
dcol = lambda/2;
myarray.ElementSpacing = [drow dcol];
myarray.ArrayNormal = "x";

% Display radiation pattern
f = figure;
az = -180:1:180;
el = -90:1:90;
pattern(myarray,fq,az,el)
```



Create Transmitter Site at Washington Monument

Create a transmitter site at the Washington Monument in Washington, DC, using the antenna array. The transmitter frequency matches the antenna's design frequency, and the transmitter output power is 1 W. Set antenna height to 169 m, which is the height of the monument.

```
tx = txsite("Name","Washington Monument",...
    "Latitude",38.88949, ...
    "Longitude",-77.03523, ...
    "Antenna",myarray,...
    "AntennaHeight",169', ...
    "TransmitterFrequency",fq,...
    "TransmitterPower",1);
```

Show Transmitter Site on a Map

Launch Site Viewer and show the transmitter site, which centers the view at the Washington Monument. The default map shows satellite imagery, and the site marker is shown at the site's antenna height.

```
if isvalid(f)
    close(f)
end
viewer = siteviewer;
show(tx)
```



Show Antenna Radiation Pattern on a Map

Visualize the orientation of the antenna by showing the radiation pattern in Site Viewer.

pattern(tx)

Select the site marker to view the color legend of the pattern.



Create Receiver Sites

Create an array of receiver sites in the Washington, DC, area. These are used as place markers for sites of interest to assess the coverage of the transmitter site.

```
% Define names for receiver sites
rxNames = [...
    "Brentwood Hamilton Field" ...
    "Nationals Park" ...
    "Union Station" ...
    "Georgetown University" ...
    "Arlington Cemetery"];

% Define coordinates for receiver sites
rxLocations = [...
    38.9080 -76.9958; ...
    38.8731 -77.0075; ...
    38.8976 -77.0062; ...
    38.9076 -77.0722; ...
    38.8783 -77.0685];

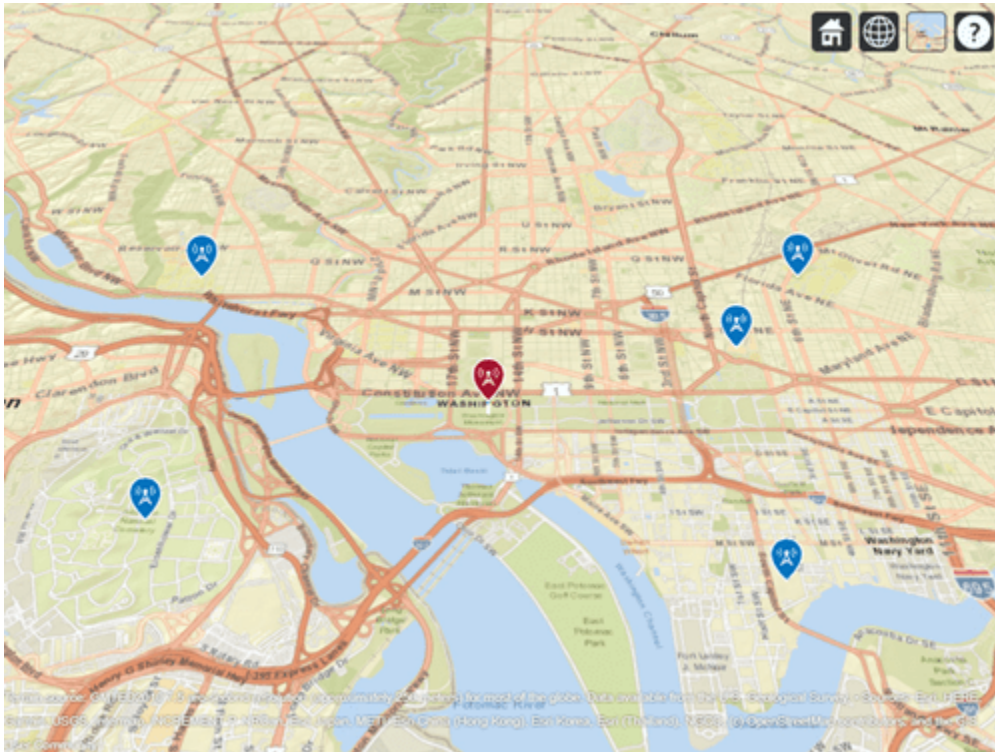
% Create array of receiver sites. Each receiver has a sensitivity of -75 dBm.
rxs = rxsite("Name",rxNames, ...
    "Latitude",rxLocations(:,1), ...
    "Longitude",rxLocations(:,2), ...
    "ReceiverSensitivity",-75);
```

Show receiver sites on a map.

```
show(rxs)
```


Set the map imagery using the Basemap property. Alternatively, open the map imagery picker in Site Viewer by clicking the second button from the right. Select "Streets" to see streets and labels on the map.

```
viewer.Basemap = "streets";
```



Scan the Array and Update the Radiation Pattern

Scan the antenna beam by applying a taper for a range of angles. For each angle, update the radiation pattern in Site Viewer. This approach of scanning the beam produces different patterns than physically rotating the antenna, as could be achieved by setting AntennaAngle of the transmitter site. This step is used to validate the orientation of the antenna's main beam.

```
% Get the starting array taper
startTaper = myarray.Taper;

% Define angles over which to perform sweep
azsweep = -30:10:30;

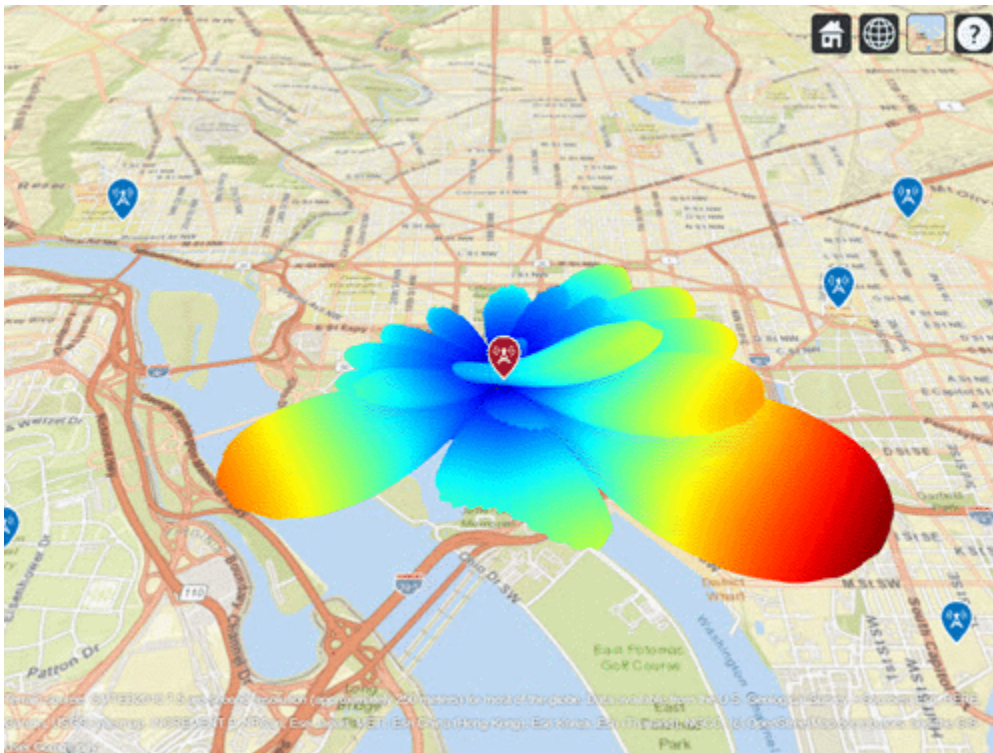
% Set up tapering window and steering vector
N = nrow*ncol;
nbar = 5;
sll = -20;
sltaper = taylorwin(N,nbar,sll)';
steeringVector = phased.SteeringVector("SensorArray",myarray);

% Sweep the angles and show the antenna pattern for each
for az = azsweep
    sv = steeringVector(fq,[az; 0]);
    myarray.Taper = sltaper.*sv';
```

```

    % Update the radiation pattern. Use a larger size so the pattern is visible among the antenna
    pattern(tx, "Size", 2500, "Transparency", 1);
end

```



Display Transmitter Coverage Map

Define three signal strength levels and corresponding colors to display on the coverage map. Each color is visible where the received power for a mobile receiver meets the corresponding signal strength. The received power includes the total power transmitted from the rectangular antenna array.

The default orientation of the transmitter site points the antenna x-axis east, so that is the direction of maximum coverage.

```

% Reset the taper to the starting taper
myarray.Taper = startTaper;

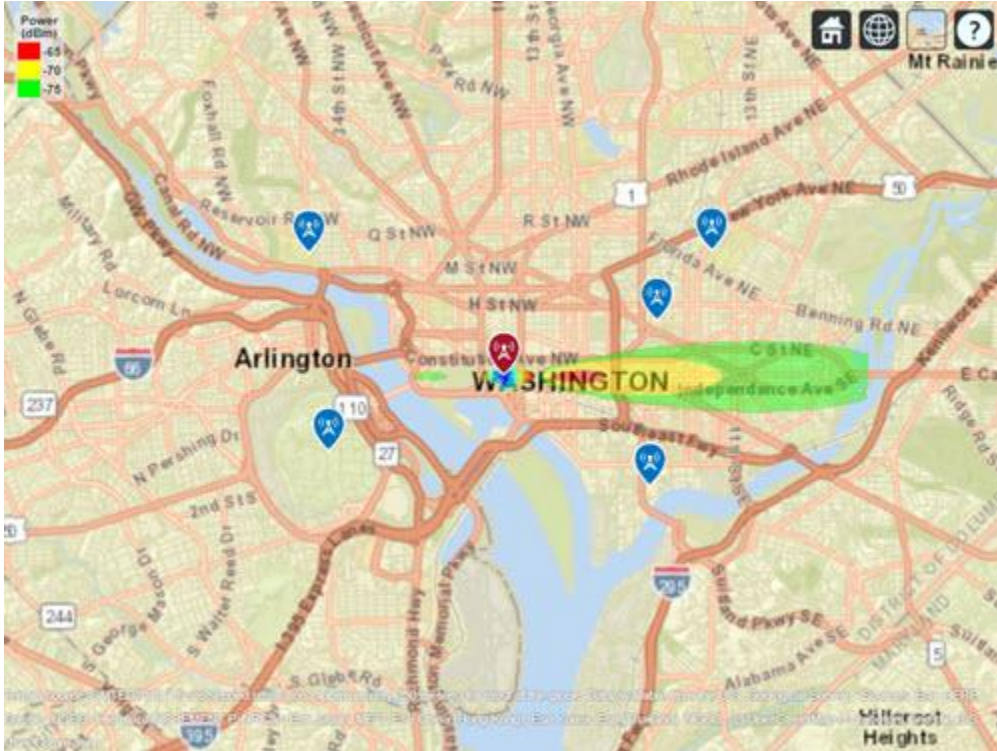
% Define signal strength levels (dBm) and corresponding colors
strongSignal = -65;
mediumSignal = -70;
weakSignal = -75;
sigstrengths = [strongSignal mediumSignal weakSignal];
sigcolors = ["red" "yellow" "green"];

% Show the tx pattern
pattern(tx, "Size", 500)

% Display coverage map out to 6 km
maxRange = 6000;

```

```
coverage(tx, ...
         "SignalStrengths", sigstrengths, ...
         "Colors", sigcolors, ...
         "MaxRange", maxRange)
```



The coverage map shows no coverage at the transmitter site and a couple of pockets of coverage along the boresight direction before the main coverage area. The radiation pattern provides insight into the coverage map by showing how the antenna power projects onto the map locations around the transmitter.



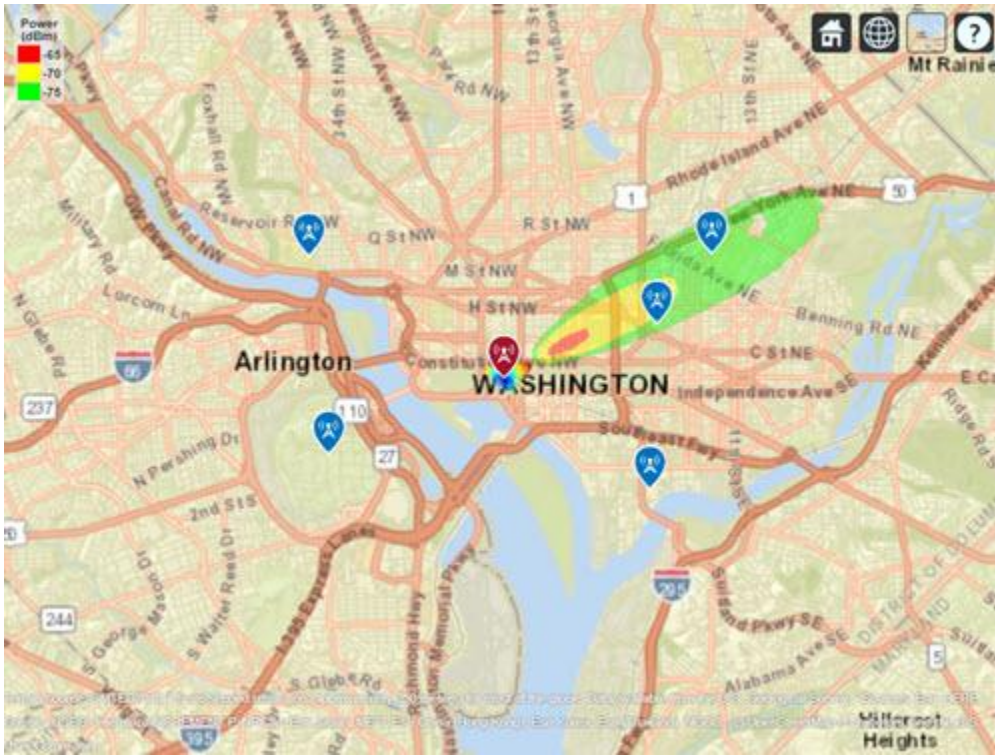
Scan the Array and Update the Coverage Display

Scan the antenna beam by applying a taper for a range of angles. For each angle, update the coverage map. This method of beamscanning is the same method used above. The final map includes two receiver sites of interest within the coverage region.

```
% Repeat the sweep but show the pattern and coverage map
for az = azsweep
    % Calculate and assign taper from steering vector
    sv = steeringVector(fq,[az; 0]);
    myarray.Taper = sltaper.*sv';

    % Update the tx pattern
    pattern(tx,"Size",500)

    % Update coverage map
    coverage(tx, ...
        "SignalStrengths",sigstrengths, ...
        "Colors",sigcolors, ...
        "MaxRange",maxRange)
end
```



See Also

Functions

[design](#) | [pattern](#) | [coverage](#)

Objects

[siteviewer](#) | [txsite](#) | [rxsite](#) | [phased.URA](#) | [phased.SteeringVector](#)

Related Examples

- “Parallelization of Antenna and Array Analyses” on page 5-68
- “Subarrays in Large Finite Array For Hybrid Beamforming” on page 5-562

Modeling Planar Photonic Band Gap Structure

This example shows how to create and analyze microwave planar Photonic Band Gap (PBG) structures in Antenna Toolbox™. Photonic Band Gap structures consist of a periodic lattice which provides effective and flexible control of the Electromagnetic wave propagation in one or multiple directions. Microwave planar PBG structures were first introduced around the year 2000 by Prof. Itoh and his group. These structures create a stop band over a certain frequency range and are easy to implement by cutting periodic patterns on the metal ground plane.

Design Frequency and System Parameters

The design shown in this example is the same as in [1]. A 50-ohm conventional microstrip is designed on a RT/Duroid 6010 substrate, with dielectric constant of 10.5 and 25 mil thickness. The strip width is 27 mil. The period of the lattice on the back is kept at 200 mil. The overall PCB board size is 6 periods by 9 periods.

```
period = 200*1e-3*0.0254; % period = 200mil;

boardLength = period*6;
boardWidth  = period*9;
boardThick  = 25*1e-3*0.0254; % board is 25mil thick
boardPlane = antenna.Rectangle('Length',boardLength,...
    'Width',boardWidth);

sub = dielectric('Name','Duroid6010','EpsilonR',10.5,...
    'Thickness',boardThick);

stripWidth = 27*1e-3*0.0254;% 27mil;
stripLength = boardWidth;

strip = antenna.Rectangle('Length',stripWidth,...
    'Width',stripLength,'Center',[0,0]);
```

In our first study, a circle with radius of 25 mil is used as the unit etch shape on the ground. A lattice of size 3 by 9 are etched out from the ground plane. The constructed ground plane is shown below. Later, we also study performance of the microstrip with different circles with larger radius, radius equals to 50 mil and 90 mil.

```
gnd = boardPlane;

radius = 25*1e-3*0.0254; % hole radius = 25mil;

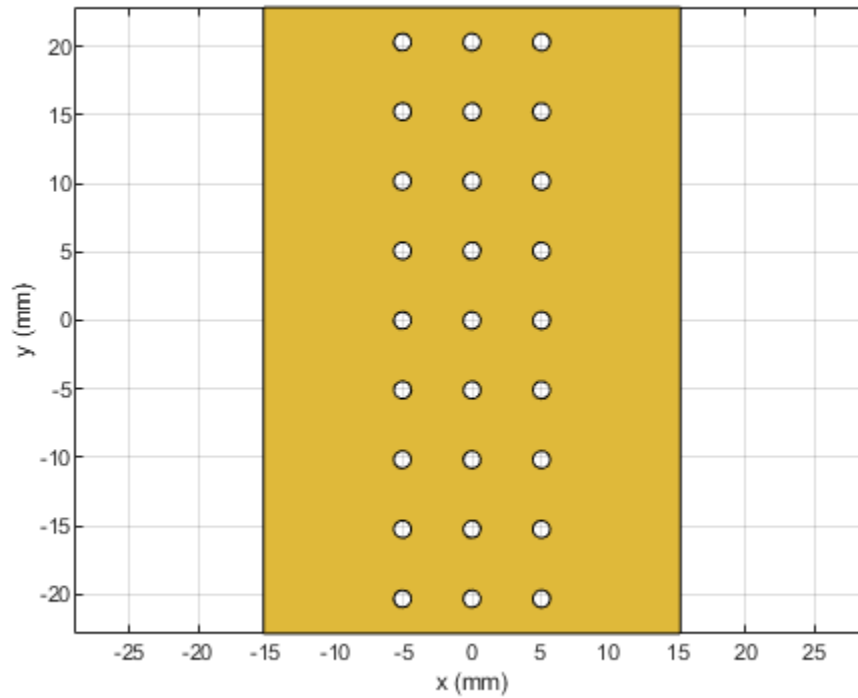
posStart = [-period, -stripLength/2+period/2];

for i = 1:3
    for j = 1:9

        pos = posStart+[(i-1)*period,(j-1)*period];
        circle = antenna.Circle('Radius',radius,...
            'Center',pos,'NumPoints',16);
        gnd = gnd - circle;

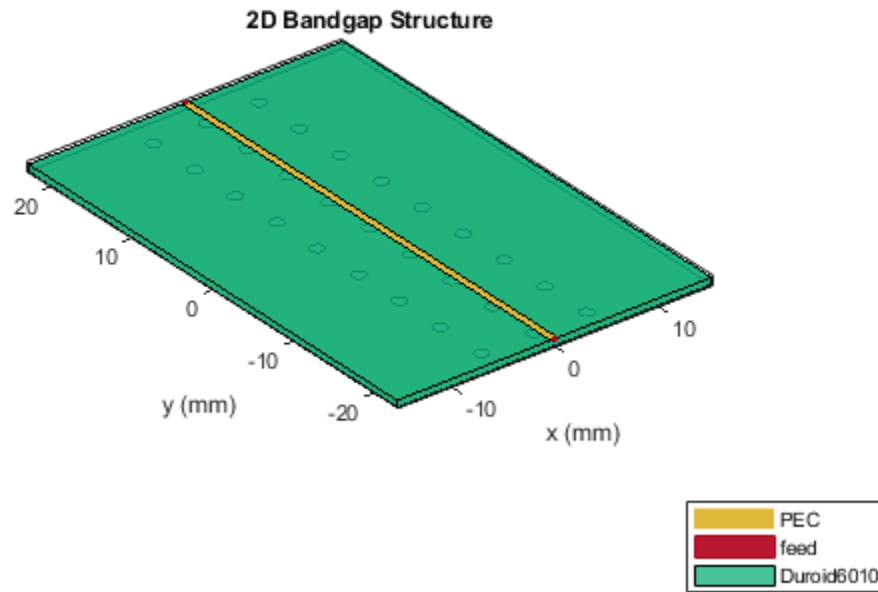
    end
end
```

```
figure;
show(gnd);
axis equal;
```



Here we combine the top microstrip, substrate and etched ground plane into `pcbStack` object for meshing and full wave analysis. The final constructed geometry is shown below:

```
obj = pcbStack('Name','2D Bandgap Structure');
obj.BoardShape = boardPlane;
obj.BoardThickness = boardThick;
obj.Layers = {strip,sub,gnd};
obj.FeedLocations = [0,-boardWidth/2,1,3;0,boardWidth/2,1,3];
obj.FeedDiameter = stripWidth/2;
figure;
show(obj);
axis equal;
title(obj.Name);
```



We manually mesh the structure using manual mesh mode in the toolbox to better control the output triangles and tetrahedra.

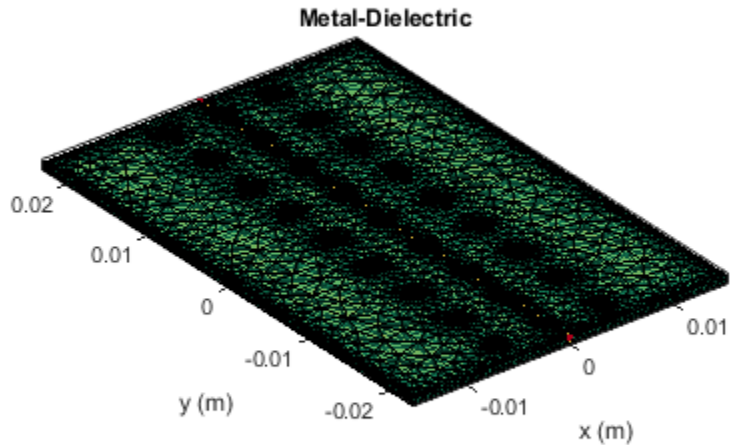
```
figure;  
mesh(obj, 'MaxEdgeLength', 12*stripWidth, ...  
      'MinEdgeLength', stripWidth, 'GrowthRate', 0.85);
```



```

NumTriangles: 5922
NumTetrahedra: 18636
NumBasis:
MaxEdgeLength: 0.0082296
MeshMode: manual

```



In order to observe the band gap effect, we compute the S-parameters for the 2-port system. The band gap effect is shown in the S21 parameter. In the analysis, we calculated the S-parameters from 2 GHz to 16 GHz, and plot the S21 and S11 for the three different circle radii. The results for all s-parameter analysis has been precomputed and stored in a MAT-file.

```

freq= linspace(2e9,16e9,141);
sparam = sparameters(obj,freq);
figure;
rfplot(sparam,1,1,'-o');
hold on;
rfplot(sparam,2,1,'--o');

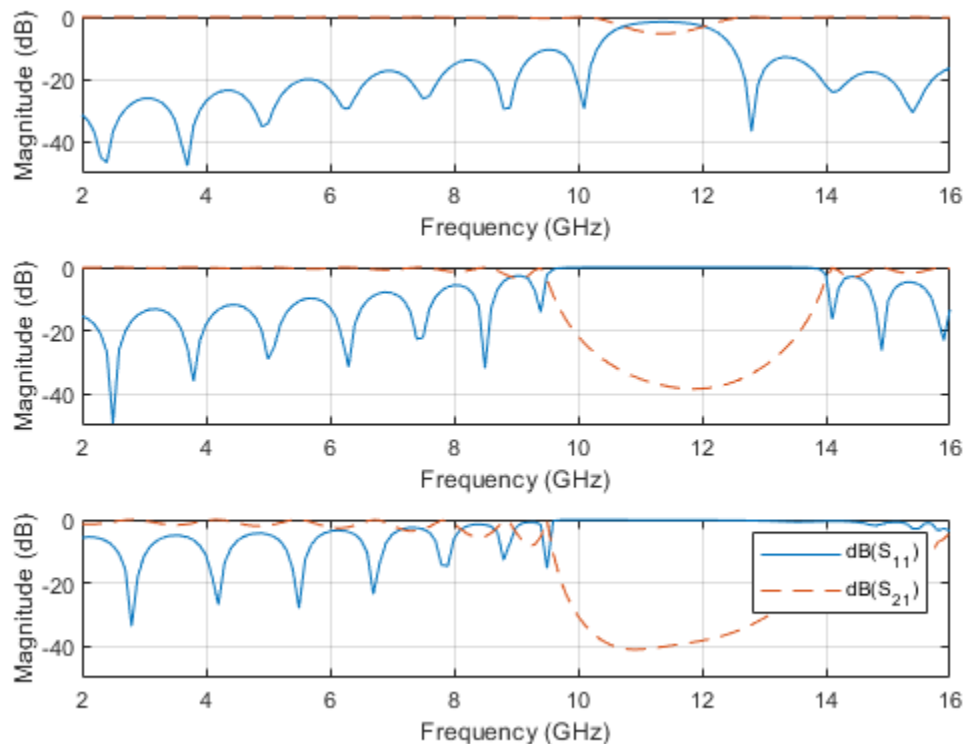
load('atx_bandgap_data.mat','sparam_25mil');
load('atx_bandgap_data.mat','sparam_50mil');
load('atx_bandgap_data.mat','sparam_90mil');
figure;
subplot(3,1,1);
rfplot(sparam_25mil,1,1,'-');
text(4e9,-40,'Radius = 25 mil', 'FontSize',8, 'Color', 'm')
hold on;
rfplot(sparam_25mil,2,1,'--');
legend off;

```

```

subplot(3,1,2);
rfplot(sparam_50mil,1,1,'-');
text(4e9,-40,'Radius = 50 mil', 'FontSize',8, 'Color', 'm')
hold on;
rfplot(sparam_50mil,2,1,'--');
legend off;
subplot(3,1,3);
rfplot(sparam_90mil,1,1,'-');
text(4e9,-40,'Radius= 90 mil', 'FontSize',8, 'Color', 'm')
hold on;
rfplot(sparam_90mil,2,1,'--');

```



From the calculated s-parameters, it is clearly seen that there is a stop band around 11GHz, which converts the 50-ohm matched transmission line into a bandstop filter. By varying the etching shape on the groundplane, different filter structures such as low pass or high pass, etc. can be realized.

90-degree Bend Microstrip Structure

As shown below, a compensated right-angle microstrip bend with patterned groundplane is created. The etched circles on the ground plane follow the right-angle bend.

```

bendboardLength = period*9;
bendboardWidth = period*9;
boardThick = 25*1e-3*0.0254; % board is 25mil thick
bendboardPlane = antenna.Rectangle('Length',bendboardLength,'Width',bendboardWidth);
bendgnd = bendboardPlane;

```

```

stripLength = bendboardWidth/2;
strip_1 = antenna.Rectangle('Length',stripLength,'Width',stripWidth,'Center',[stripLength/2,0]);
strip_2 = antenna.Rectangle('Length',stripWidth,'Width',stripLength+stripWidth/2,'Center',[0,-st

bendstrip = strip_1+strip_2;

radius = 50*1e-3*0.0254; % hole radius = 50mil;

posStart = [-period, -stripLength+period/2];

pos = zeros(27,2);

for i = 1:3
    for j =1:6

        pos = posStart+[(i-1)*period,(j-1)*period];
        circle = antenna.Circle('Radius',radius,'Center',pos,'NumPoints',15);
        bendgnd = bendgnd - circle;

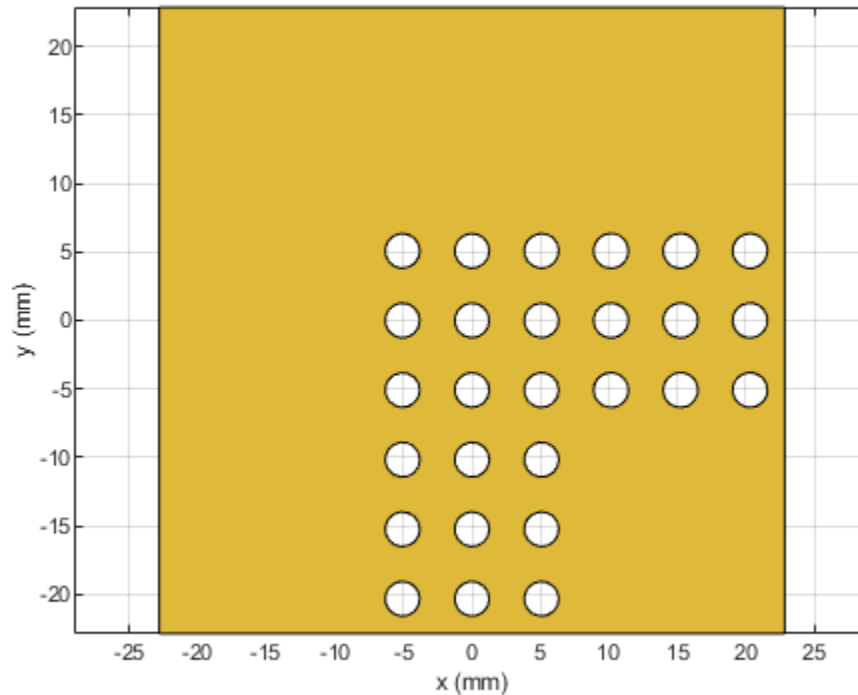
    end
end

posStart = [period, -period];

for i = 1:3
    for j = 1:3
        pos = posStart+[(i)*period,(j-1)*period];
        circle = antenna.Circle('Radius',radius,'Center',pos);
        bendgnd = bendgnd - circle;
    end
end

figure;
show(bendgnd);
axis equal;

```



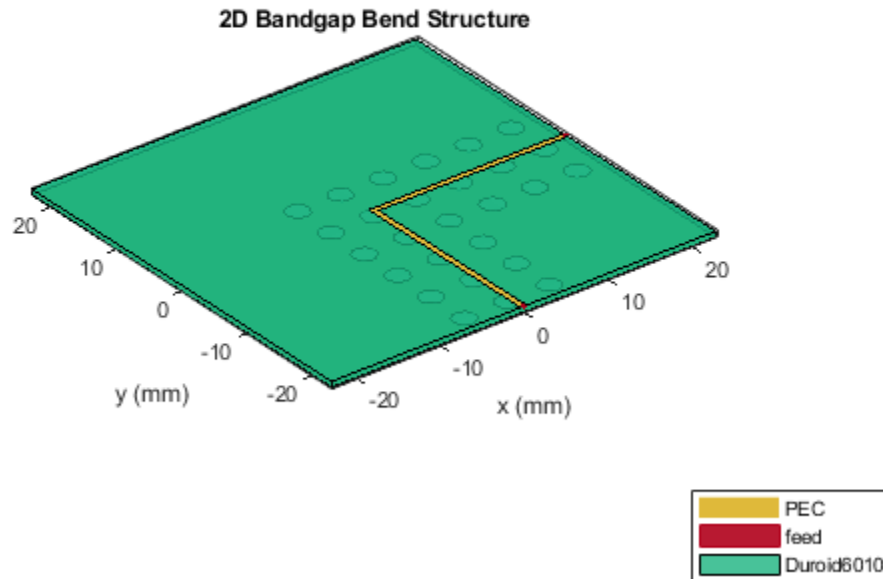
PCB Stack Creation

Create the layer stack up by organizing the PCB layers, setting the feed locations for the ports.

```

bendobj = pcbStack('Name', '2D Bandgap Bend Structure');
bendobj.BoardShape = bendboardPlane;
bendobj.BoardThickness = boardThick;
bendobj.Layers = {bendstrip, sub, bendgnd};
bendobj.FeedLocations = [0, -bendboardLength/2, 1, 3; bendboardWidth/2, 0, 1, 3];
bendobj.FeedDiameter = stripWidth/2;
figure;
show(bendobj);
axis equal;
title(bendobj.Name);

```

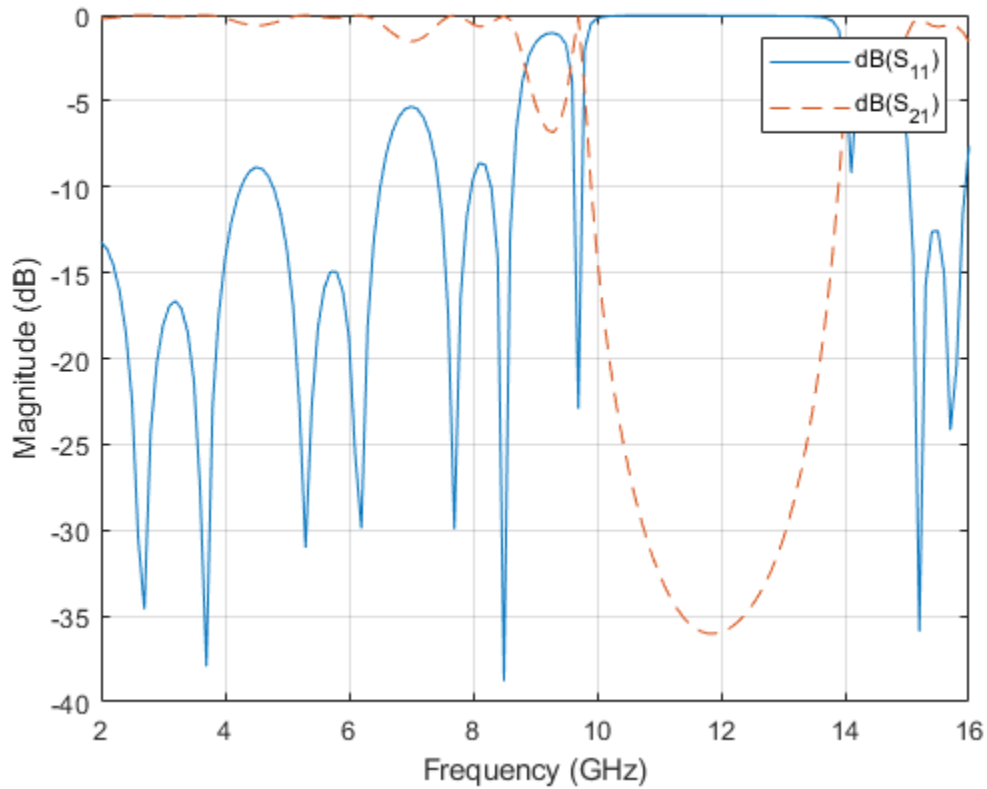


The S-parameters for the structure is calculated. The analysis results compare favorably with the measured results reported in[1] and the PBG properties of the structure are captured effectively. It is noted here that the analysis predicts almost perfect reflection within the stop band while the measured results shown in Fig. 3 of [1] reveal the presence of a loss mechanism that improves the impedance match slightly.

```

freq= linspace(2e9,16e9,141);
sparam = sparameters(obj,freq);
figure;
rfplot(sparam,1,1,'-o');
hold on;
rfplot(sparam,2,1,'--o');
load('atx_bandgap_data.mat','sparam_bend_50mil');
figure;
rfplot(sparam_bend_50mil,1,1,'-');
hold on;
rfplot(sparam_bend_50mil,2,1,'--');

```



Conclusion

The result obtained for the three designs matches well with the result published in [1].

Reference

[1] V. Radisic, Y. Qiang, R. Coccioli, and T. Itoh, "Novel 2-D Photonic Bandgap Structure for Microstrip Lines", *IEEE Microwave and Guided Wave Letters*, vol. 8, No. 2, 1998;

See Also

"Metasurface Antenna Modeling" on page 5-381

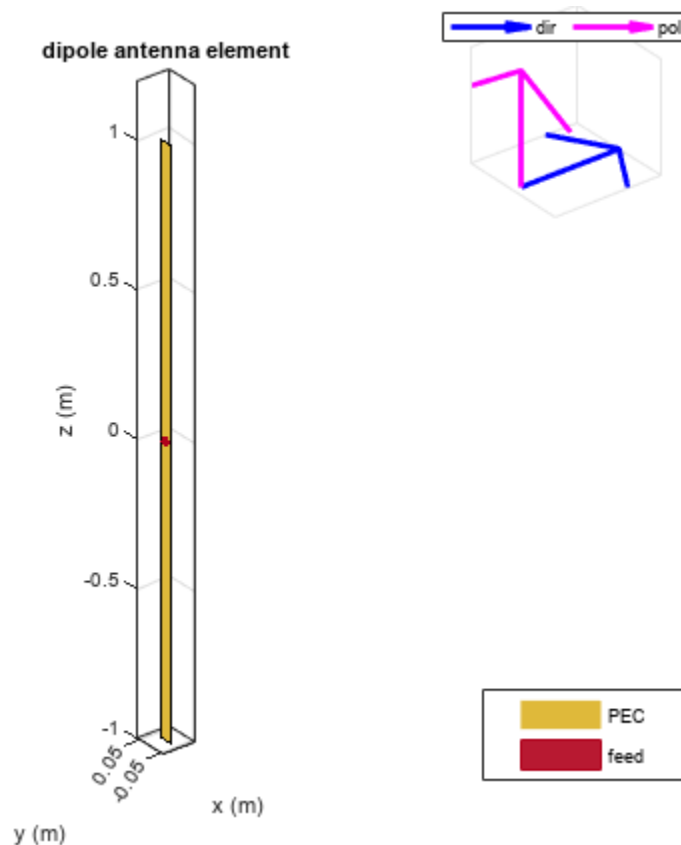
Plane Wave Excitation - Scattering Solution

This example explains how to excite an antenna using a plane wave. The antenna in this case can be thought of as a receiving antenna. A receiving antenna may be viewed as any metal object that scatters an incident electromagnetic field. As a result of scattering an electric current appears on the antenna's surface. The current in turn creates a corresponding electric field. This produces a voltage difference across the feed. This voltage constitutes the received signal. [1]

Incident Plane Wave on Dipole Antenna

Consider the dipole antenna of length 2 m and width of 50 mm oriented along the z-axis. It is excited by a plane-wave directed along the positive x-axis and having a z-polarization. The inset figure in the top right hand corner shows the direction and polarization of the plane wave w.r.t the antenna geometry.

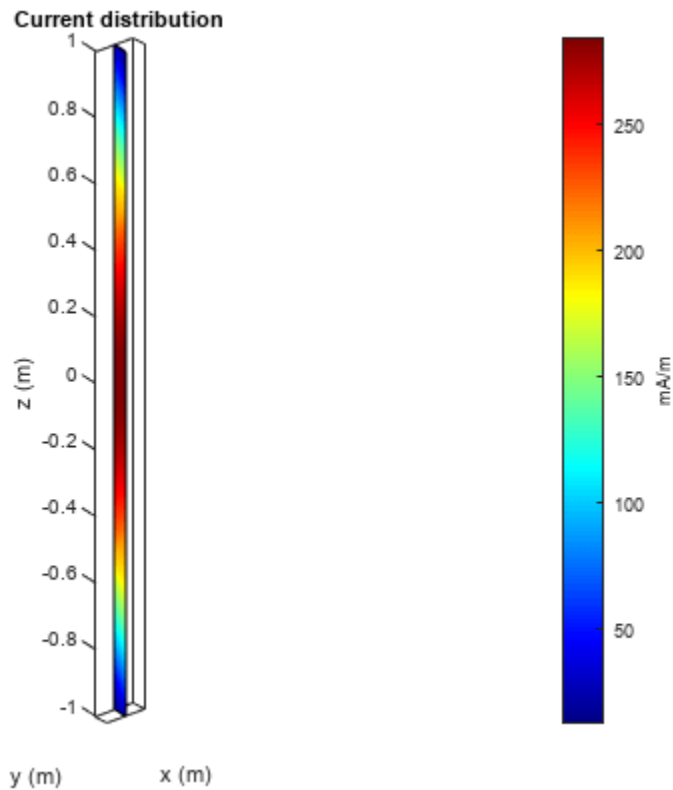
```
d = dipole('Length', 2, 'Width', 50e-3);
p = planeWaveExcitation('Element', d, 'Direction', [1 0 0], ...
    'Polarization', [0 0 1]);
show(p);
```



Visualize Current Distribution

The current function displays the resulting current distribution on the antenna surface. The z-component of the current, along the dipole axis, dominates. The maximum is in the middle of the dipole.

```
current(p, 75e6);
```



Measure Feed Current

The current at the antenna feed is calculated by determining the current density at the antenna feed and multiplying it with the length of the feed. In this case the current is about 13.9 mA.

```
I = feedCurrent(p, 75e6);
magI = abs(I);
```

Calculate Voltage at Feed

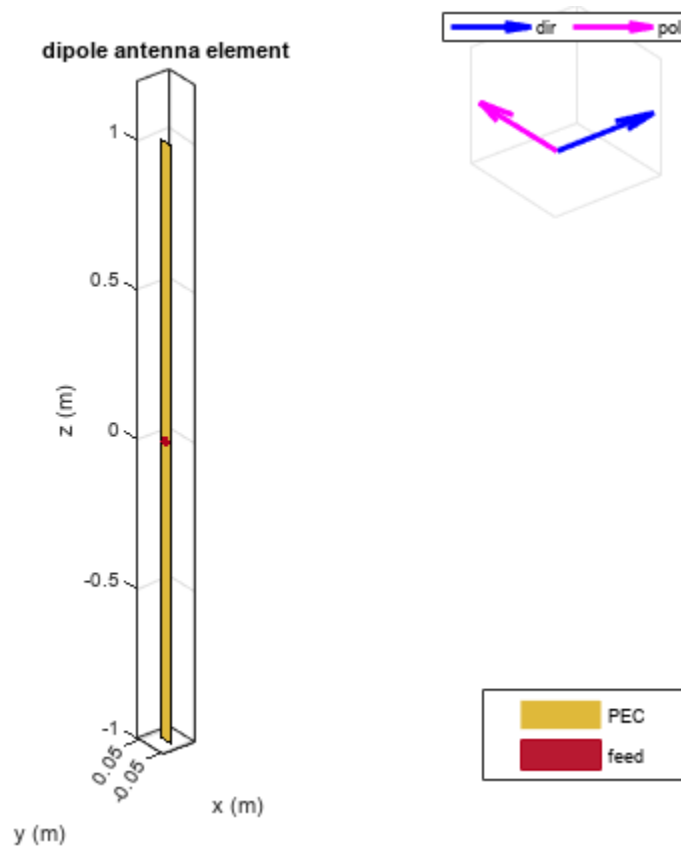
The feed voltage can be calculated as the product of the feed current and the impedance of the antenna at that frequency. In this case the feed voltage is about 1.45 V. This is the received signal.

```
Z = impedance(d, 75e6);
feedV = abs(Z*I);
```

Change Polarization of Plane Wave

Change the polarization of the plane wave so that it is y-polarized.

```
p.Polarization = [0 1 0];
show(p);
```

In this case feed voltage is about 4.7 mV. The received signal is down by a factor of 300, so the antenna is receiving very little as compared to the previous case.

```
Icross = feedCurrent(p, 75e6);
feedVcross = abs(Z*Icross);
```

This indicates that the dipole antenna is capable of receiving signals whose E-field has a component parallel to the dipole axis. So dipole is a linearly polarized antenna.

Reference

[1] S. N. Makarov, *Antenna and EM Modeling with MATLAB*, chapter 2, Wiley, New York, 2002.

See Also

“Radar Cross Section Benchmarking” on page 5-579

Design, Analysis, and Prototyping of Microstrip-Fed Wide-Slot Antenna

This example builds a model of a microstrip-fed printed wide slot antenna on FR4, analyzes it and finally enables prototyping by generating Gerber files. The design is intended for operation in the L-band and has a bandwidth of about 17% over the band 1.6 - 1.8 GHz.

Design parameters

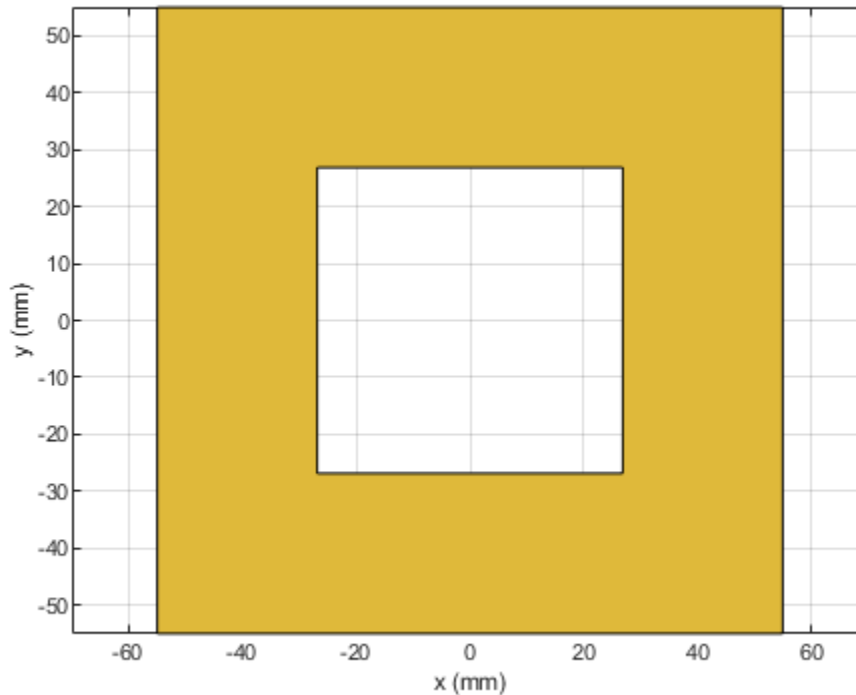
Define the design parameters of the antenna as provided in [1].

```
pcbThickness = 0.8e-3;  
pcbMaterial = 'FR4';  
pcbEpsilonR = 4.4;  
pcbTraceWidth = 1.5e-3;  
gndLength = 110e-3;  
gndWidth = 110e-3;  
slotLength = 53.7e-3;  
slotWidth = 53.7e-3;  
offsetTraceLength = 28e-3;  
pcbTraceLength = (gndWidth/2 - slotWidth/2) + offsetTraceLength;
```

Create Layer Shapes

Using the design parameters, create the basic shape primitives. The antenna has two metal layers on either side of a single-layer PCB. The first metal layer is the microstrip feedline and the second layer of metal is the groundplane with a wide slot cut out from it. After defining the shapes, use Boolean subtraction to create the slot in the ground plane.

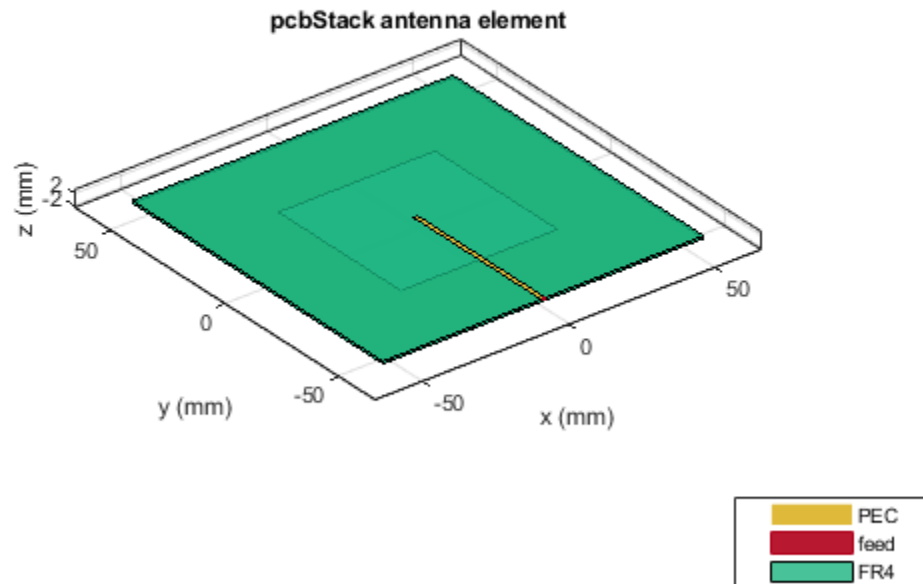
```
feed = antenna.Rectangle('Length',pcbTraceWidth,'Width',pcbTraceLength,...  
                        'Center',[0, -gndWidth/2+pcbTraceLength/2],...  
                        'NumPoints',[2 40 2 40]);  
gnd = antenna.Rectangle('Length',gndLength,'Width',gndWidth);  
gndslot = antenna.Rectangle('Length',slotLength,'Width',slotWidth);  
gndPlane = gnd - gndslot;  
figure;  
show(gndPlane)
```



Create Stack

Create the PCB stack by defining the dielectric material and arranging the layers in a top-down description starting with the top-most layer of metal. Define a feed location and the feed diameter as well. This antenna has the microstrip feedline brought out to the edge of the board.

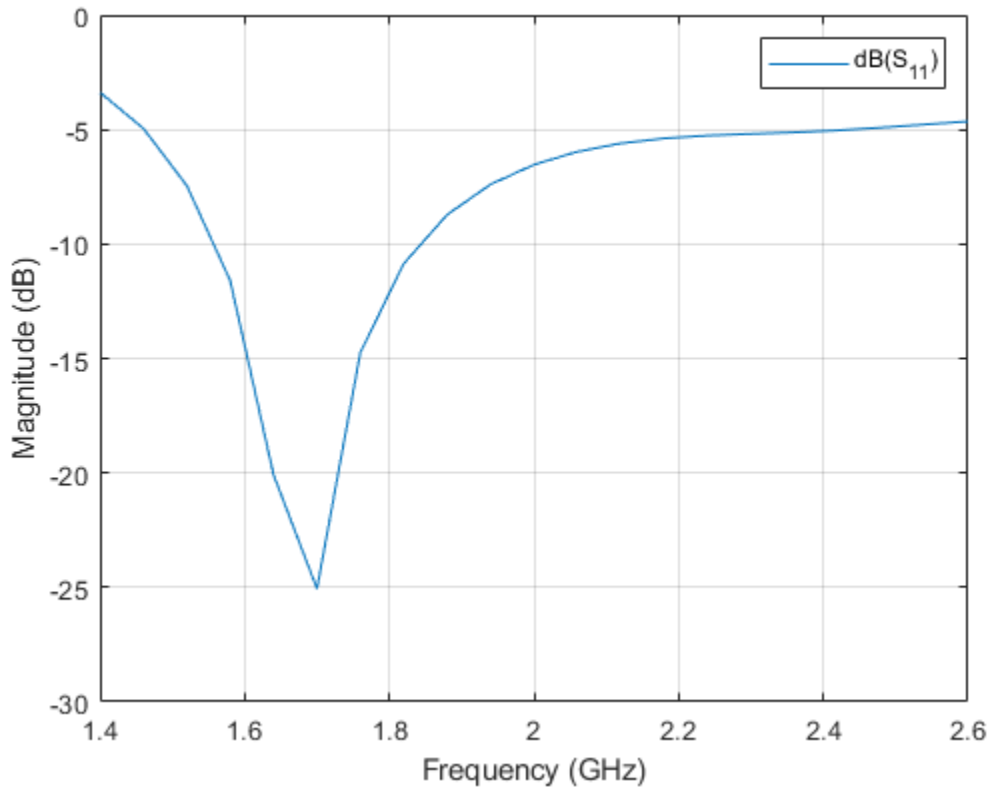
```
d = dielectric(pcbMaterial);
d.EpsilonR = pcbEpsilonR;
d.Thickness = pcbThickness;
p = pcbStack;
p.Name = 'Strip-fed slot';
p.BoardShape = gnd;
p.BoardThickness = pcbThickness;
p.Layers = {feed,d,gndPlane};
p.FeedLocations = [0,-gndWidth/2,1,3];
p.FeedDiameter = pcbTraceWidth/2;
figure;
show(p)
```



Compute S-parameters

The antenna reflection coefficient reveals how well it responds to stimulus at any particular frequency. Typically, $S_{11} \leq -10$ dB is considered to be good from an impedance matching perspective. The reference impedance here is the default of 50-ohms.

```
fmin = 1.4e9;  
fmax = 2.6e9;  
Z0 = 50;  
N = 21;  
freq = linspace(fmin, fmax, N);  
s = sparameters(p, freq, Z0);  
s11Fig = figure;  
rfplot(s, 1, 1)
```



The plot of the S_{11} reveals a good match to 50-ohms within the band 1.6 - 1.85 GHz.

Gerber Files

Gerber files are a commonly used format to export the geometry information of a PCB. To generate these files, two additional pieces of information are required apart from the PCB itself. The first is the type of connector to be used and the second is the PCB manufacturing service/viewer service. The type of RF connector determines the pad layouts on the PCB. The Antenna Toolbox™ provides a catalog of PCB services and RF connectors. The PCB services catalog supports configuring the Gerber file generation process for manufacturing as well as for online viewer-only.

Gerber generation The collection of Gerber files that describe a Printed Circuit Board (PCB) have each a different role. Each file describes a specific aspect of the PCB design. As an example, on the PCB there are metal regions corresponding to the signal and ground that are filled with Copper. This information is captured in the .gtl and .gbl files. Information about the solder mask, which is applied to protect and insulate the metal regions, is captured in the .gts and .gbs files. Design information is encoded into the silkscreen layer designated by .gto and .gbo files. To understand the generation process for these files use a PCB manufacturing service with an online viewer to render the design.

Online Gerber Viewer Use the MayhewWriter to configure the Gerber file generation process for the Mayhewlabs free online 3D Gerber viewer. Select an SMA edge connector from the catalog and modify it for this particular design. Use the PCB antenna model, the service and the RF connector to create a PCBWriter.

```
W = PCBServices.MayhewWriter;
W.FileName = 'Microstrip-fed slot patch-MH';
```

```













C = SMAEdge_SamtecCustom;
C.EdgeLocation = 'south';
C.ExtendBoardProfile = false;
Am = PCBWriter(p,W,C);

```

Execute the gerberWrite command to generate the Gerber files.

```
gerberWrite(Am)
```

The files are generated and placed in zipped folder with the same name as assigned in the Filename property of the particular PCB service that was chosen. The location of the folder is the current working directory. The files in the folder are shown in the image below.

Name	Date modified	Type	Size
 Microstrip_fedSlotPatch_MH.dri	7/6/2017 1:24 PM	DRI File	1 KB
 Microstrip_fedSlotPatch_MH.gbl	7/6/2017 1:24 PM	GBL File	1 KB
 Microstrip_fedSlotPatch_MH.gbo	7/6/2017 1:24 PM	GBO File	384 KB
 Microstrip_fedSlotPatch_MH.gbp	7/6/2017 1:24 PM	GBP File	1 KB
 Microstrip_fedSlotPatch_MH.gbs	7/6/2017 1:24 PM	GBS File	1 KB
 Microstrip_fedSlotPatch_MH.gpi	7/6/2017 1:24 PM	GPI File	2 KB
 Microstrip_fedSlotPatch_MH.gtl	7/6/2017 1:24 PM	GTL File	2 KB
 Microstrip_fedSlotPatch_MH.gto	7/6/2017 1:24 PM	GTO File	489 KB
 Microstrip_fedSlotPatch_MH.gtp	7/6/2017 1:24 PM	GTP File	1 KB
 Microstrip_fedSlotPatch_MH.gts	7/6/2017 1:24 PM	GTS File	1 KB
 Microstrip_fedSlotPatch_MH.ipc	7/6/2017 1:24 PM	IPC File	1 KB
 Microstrip_fedSlotPatch_MH	7/6/2017 1:24 PM	Text Document	1 KB

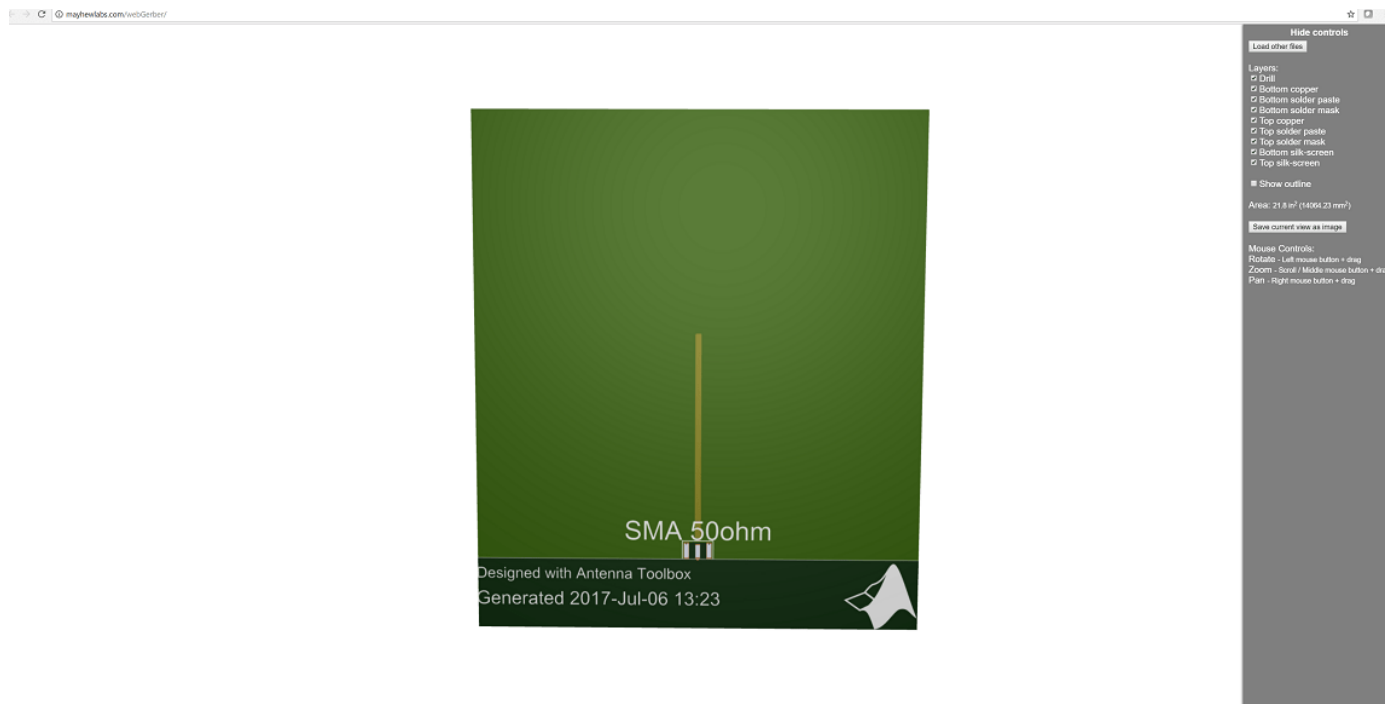
In addition to this, if internet access is available, a browser window will open for the Mayhewlabs free 3D online Gerber viewer. Select and drag all the generated files into the browser window. The files and their purpose are organized as shown below. When ready click on 'Done'.

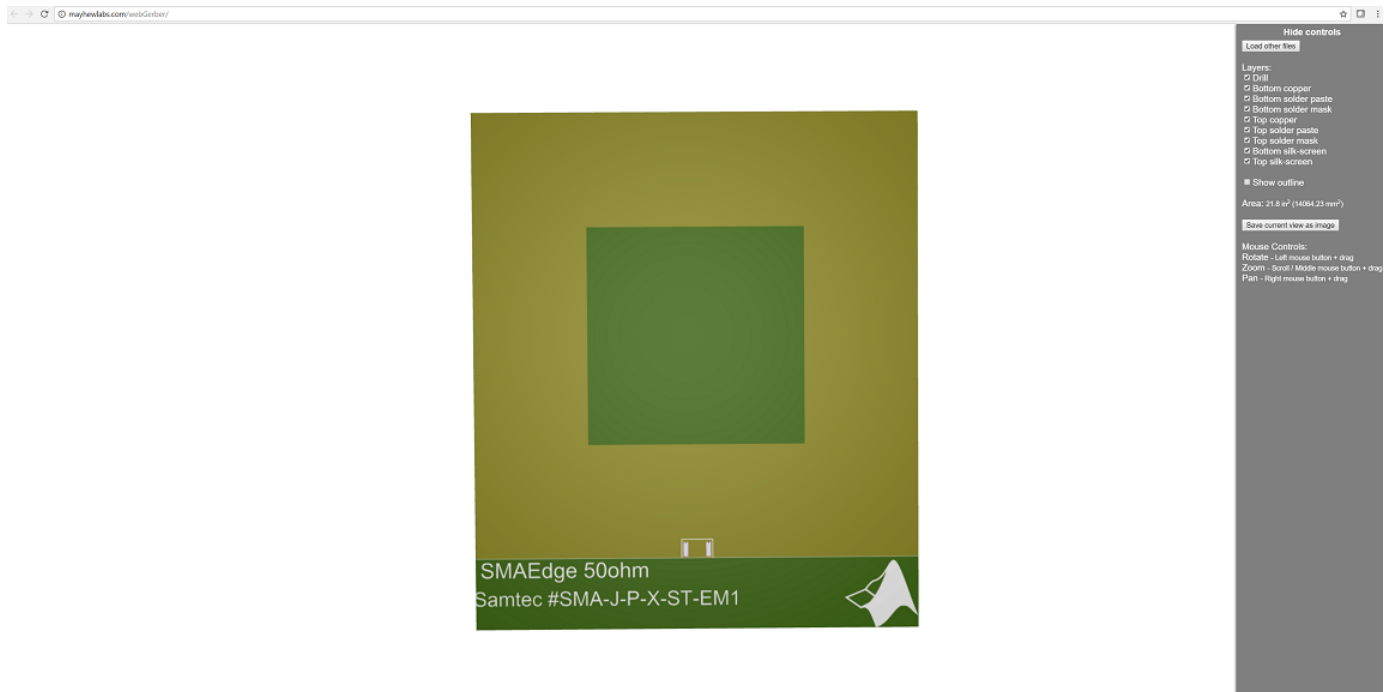
Step 2:
Select the layers corresponding to the gerber files

Microstrip_fedSlotPatch_MH.txt	Drill
Microstrip_fedSlotPatch_MH.dri	No layer
Microstrip_fedSlotPatch_MH.gbl	Bottom copper
Microstrip_fedSlotPatch_MH.gbo	Bottom silk-screen
Microstrip_fedSlotPatch_MH.gbp	Bottom solder paste
Microstrip_fedSlotPatch_MH.gbs	Bottom solder mask
Microstrip_fedSlotPatch_MH.gpi	No layer
Microstrip_fedSlotPatch_MH.gtl	Top copper
Microstrip_fedSlotPatch_MH.gto	Top silk-screen
Microstrip_fedSlotPatch_MH.gtp	Top solder paste
Microstrip_fedSlotPatch_MH.gts	Top solder mask
Microstrip_fedSlotPatch_MH.ipc	No layer

Done

The PCB design described by the set of Gerber files is now rendered in the browser window. Use the mouse to orient and position the design. The menus on the right of the screen enables selective viewing of different parts of the Gerber file such as the soldermask, copper layers and silkscreen.





PCB Service for Manufacturing Use the SeedWriter to configure the Gerber file generation process for the Seed Fusion PCB manufacturing services. Regenerate the PCBWriter with the new service.

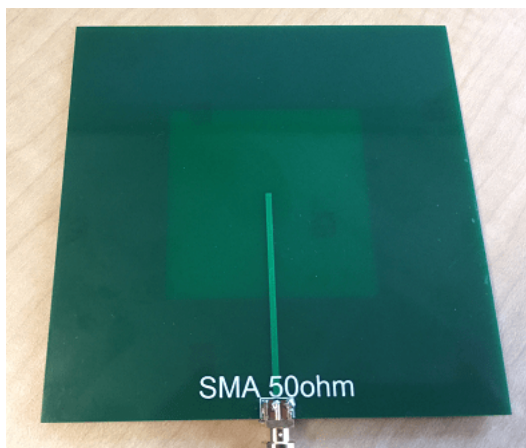
```
W = PCBServices.SeedWriter;  
W.FileName = 'Microstrip-fed slot patch-SS';  
As = PCBWriter(p,W,C);
```

Execute the gerberWrite command to generate the Gerber files.

```
gerberWrite(As)
```

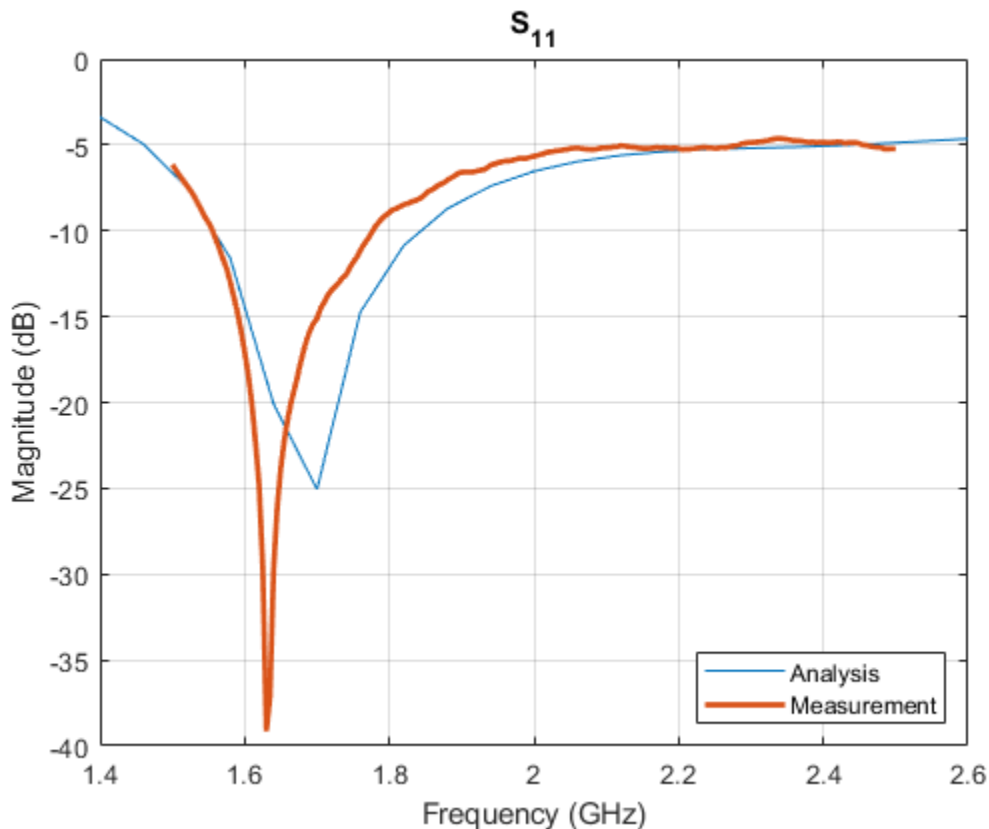
Manufactured Antenna and Measurements

Submit the order on the Seed Fusion website together with the generated Gerber files. The manufactured antenna is mailed out in a few weeks.



The reflection coefficient of the prototype antenna was measured in the Antenna Lab at Worcester Polytechnic Institute (WPI). The results are plotted as shown below.

```
s11_1 = csvread('TRACE01.CSV',3,0);
freq_trace = s11_1(:,1)./1e9;
s11Data = s11_1(:,2);
figure(s11Fig);
hold on
plot(freq_trace, s11Data,'LineWidth',2);
legend('Analysis','Measurement','Location','SouthEast')
title('S_1_1')
hold off
```



Conclusion

The agreement between the analyzed and measured results is reasonable with approximately 4% absolute error in the S_{11} minimum.

Reference

[1] Jia-Yi Sze and Kin-Lu Wong, "Bandwidth enhancement of a microstrip-line-fed printed wide-slot antenna," in *IEEE Transactions on Antennas and Propagation*, vol. 49, no. 7, pp. 1020-1024, Jul 2001.

See Also

"Model and Analyze Dual Polarized Patch Microstrip Antenna" on page 5-476

Comparison of Antenna Array Transmit and Receive Manifold

This example calculates and compares the transmit and receive manifolds for a basic half-wavelength dipole antenna array. The array manifold is a fundamental property of antenna arrays, both in transmit and receive configurations. The transmit and receive manifolds are theoretically the same due to the reciprocity theorem. This example validates this equality thus providing an important verification of the calculations performed by the Antenna Toolbox™.

Analysis Setup

Define the variables necessary for calculating the transmit and receive manifolds of an antenna. The frequency for the analysis is 300 MHz which results in a free-space wavelength of approximately 1. Define the sweep angles for azimuth and elevation. These variables will be used to calculate the positions in the far-field of the antenna at which the electric and magnetic fields are computed.

```
fc      = 3e8;
lambda = physconst('lightspeed')/fc;
R       = 100*lambda;
M       = 4;
ZL      = [];
phi     = 5:5:175;
psi     = 0:5:80;
nphi    = length(phi);
npsi    = length(psi);
np      = nphi*npsi;
PHI     = kron(ones(1,npsi),pi/180*phi);
PSI     = kron(pi/180*psi,ones(1,nphi));
```

Create Cartesian Co-ordinates for Observation and Unit Vectors

The azimuth and elevation angle variables are used to calculate the points on a sphere with radius of 100λ and the unit vectors.

```
antennapos = [(-(M-1)/2:(M-1)/2)*lambda/2;zeros(2,M)]';
x          = R*cos(PSI).*cos(PHI);
y          = R*cos(PSI).*sin(PHI);
z          = R*sin(PSI);
```

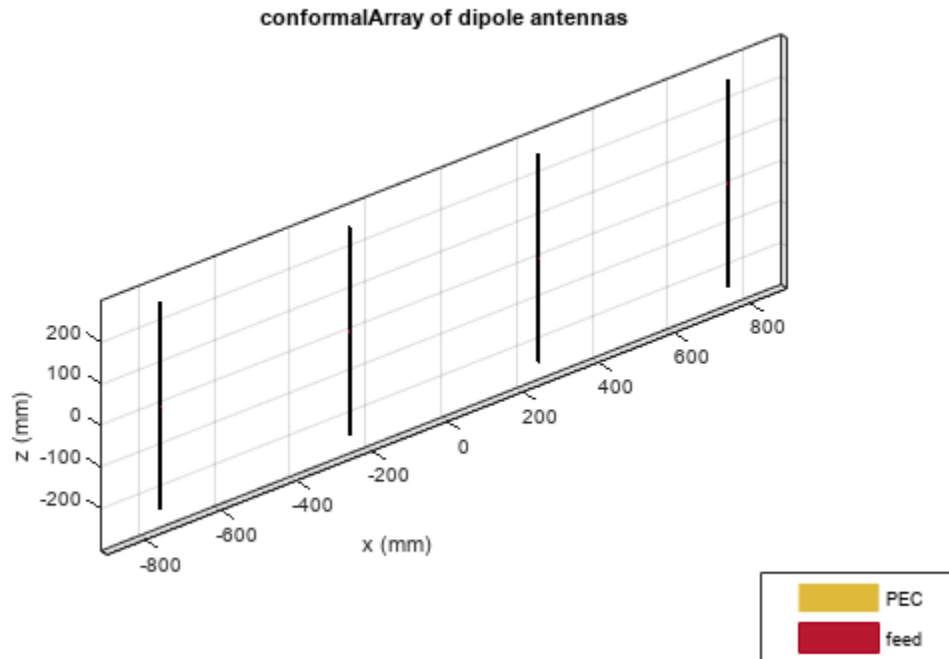
```
Points = [x;y;z];
```

```
h = [-sin(PHI);cos(PHI);zeros(size(PHI))];
v = [-cos(PHI).*sin(PSI);-sin(PHI).*sin(PSI);cos(PSI)];
u = [cos(PHI).*cos(PSI);sin(PHI).*cos(PSI);sin(PSI)];
```

Create Antenna Array

Create a half-wavelength dipole antenna, positioned vertically along the z-axis. Use this element to build a conformal array of dipoles arranged in a linear configuration with half-wavelength spacing between elements.

```
d = dipole('Length',lambda/2,'Width',lambda/200);
elem = [d,d,d,d];
dA = conformalArray('Element',elem,'ElementPosition',antennapos);
figure;
show(dA)
```



Compute Array Transmit Manifold

Use `EHfields` to calculate the electric and magnetic field vectors in the transmit mode for the array. The field calculation at 300 MHz is done at the observation points specified on the far-field sphere and considering the vertical polarization contribution explicitly since the dipole orientation is vertical. The analysis is conducted such that the element under excitation has a 50Ω load in series with the voltage source, and all other elements, which are not under excitation, do not have any internal impedance.

```

nomLoad = lumpedElement('Impedance',ZL);
actLoad = lumpedElement('Impedance',50);
for i=[1,1:M]
    % setup loads
    for m=1:M
        dA.Element(m).Load = nomLoad;
    end
    dA.Element(i).Load = actLoad;
    % setup active element
    ampTaper = zeros(1,M);
    ampTaper(i) = 1;
    dA.AmplitudeTaper = ampTaper;
    [E,H] = EHfields(dA,fc,Points);
    Etx(i,:) = sum(E.*v);
end

```

Compute Array Receive Manifold

To calculate the receive manifold, consider a plane wave incident upon the array with the same electric field polarization but opposite in direction to the transmit mode. Use the `planeWaveExcitation` object and the `feedCurrent` function to compute the current passing through the feeds in response to the impinging plane wave.

```
nomLoad = lumpedElement('Impedance',ZL);
for m=1:M
    dA.Element(m).Load = nomLoad;
end

for n = 1:np
    dirVec = -u(:,n);
    polVec = v(:,n);
    p = planeWaveExcitation('Element',dA,'Direction',dirVec,'Polarization',polVec);
    Erx(:,n) = feedCurrent(p,fc).';
end
```

Compute Normalized Error Between Manifolds

Calculate the normalized error between the transmit and receive manifolds for each element in the array. Reshape the error to plot as a function of the azimuth and elevation angles respectively

```
for i=1:M
    a(i) = Erx(i,:)/Etx(i,:);
    err(i,:) = abs(Erx(i,:)-a(i)*Etx(i,:));
    mse      = sqrt(mean(abs(a(i)*Etx(i,:)).*abs(Erx(i,:))));
    err(i,:) = err(i,+)/mse;
    Etx(i,:) = a(i)*Etx(i,:);
end

ETX = reshape(Etx,M,nphi,npsi);
ERX = reshape(Erx,M,nphi,npsi);

for i = 1:M
    ETXmag(i,,:) = abs(squeeze(ETX(i,:,:)))';
    ETXphase(i,,:) = 180/pi*angle(squeeze(ETX(i,:,:)))';
    ERXmag(i,,:) = abs(squeeze(ERX(i,:,:)))';
    ERXphase(i,,:) = 180/pi*angle(squeeze(ERX(i,:,:)))';
end
ERR = 20*log10(reshape(mean(err),nphi,npsi));
```

Plot the Transmit, Receive Manifold and the Error Pattern

The transmit and receive manifolds show a great degree of similarity in the magnitude and phase plots. This is confirmed by the error plot with a maximum error of approximately -40 dB over the span of azimuth and elevation angles.

```
for i=1:M
    figure;
    subplot(221)
    imagesc(phi,psi,squeeze(ETXmag(i,:,:)));
    colorbar
    colormap('jet')
    xlabel('Azimuth [deg]','LineWidth',6);
    ylabel('Elevation [deg]','LineWidth',6);
    title(['Tx magnitude pattern, ant ',num2str(i)],'FontSize',10)
```

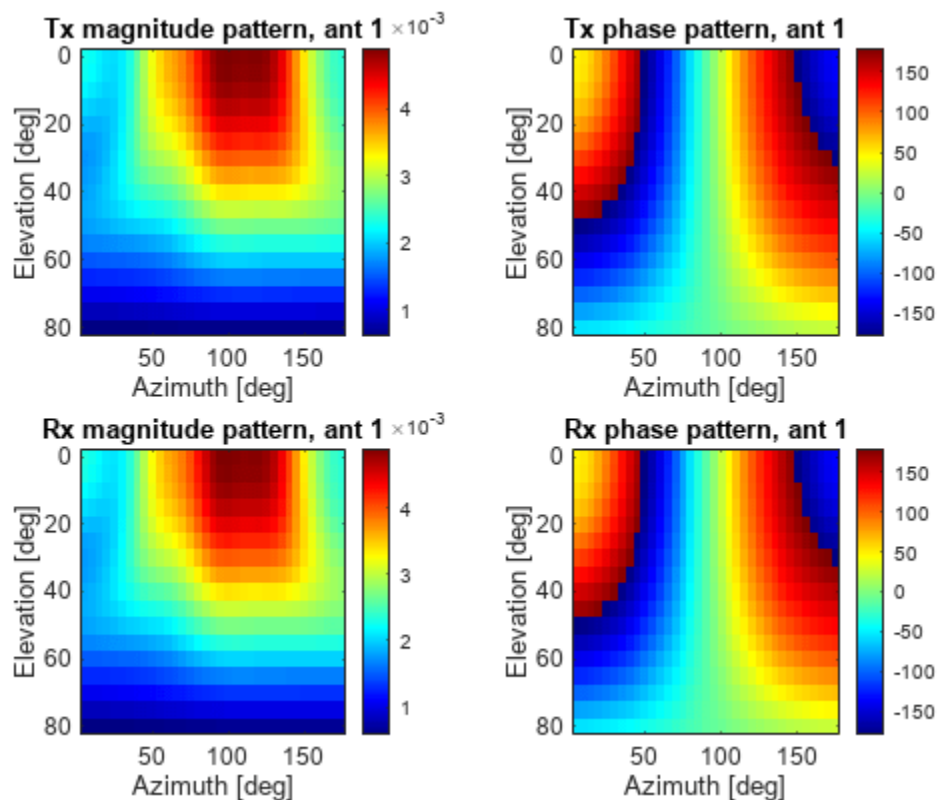
```

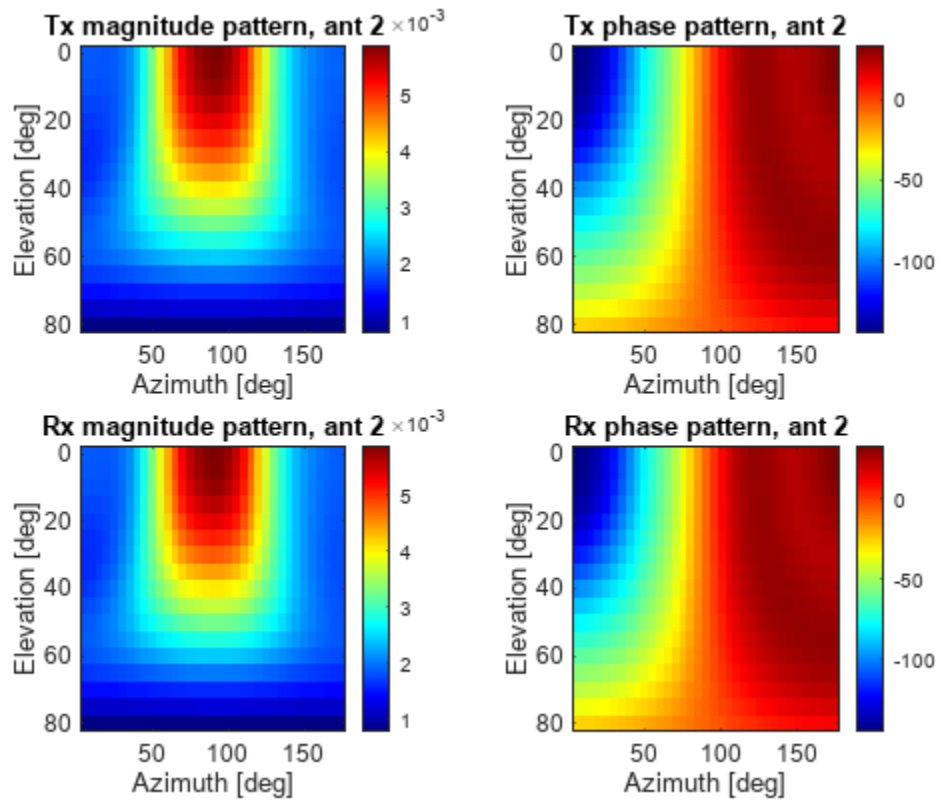
subplot(222)
imagesc(phi,psi,squeeze(ETXphase(i,:,:)));
colorbar
colormap('jet')
xlabel('Azimuth [deg]','LineWidth',6);
ylabel('Elevation [deg]','LineWidth',6);
title(['Tx phase pattern, ant ',num2str(i)],'FontSize',10)

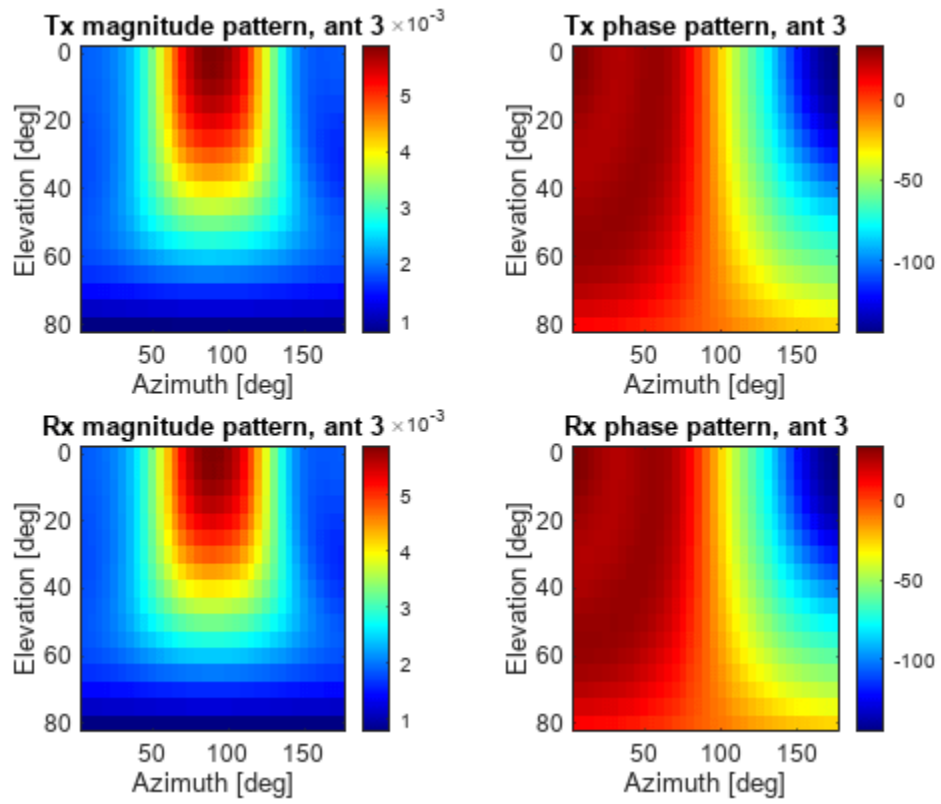
subplot(223)
imagesc(phi,psi,squeeze(ERXmag(i,:,:)));
colorbar
colormap('jet')
xlabel('Azimuth [deg]','LineWidth',6);
ylabel('Elevation [deg]','LineWidth',6);
title(['Rx magnitude pattern, ant ',num2str(i)],'FontSize',10);

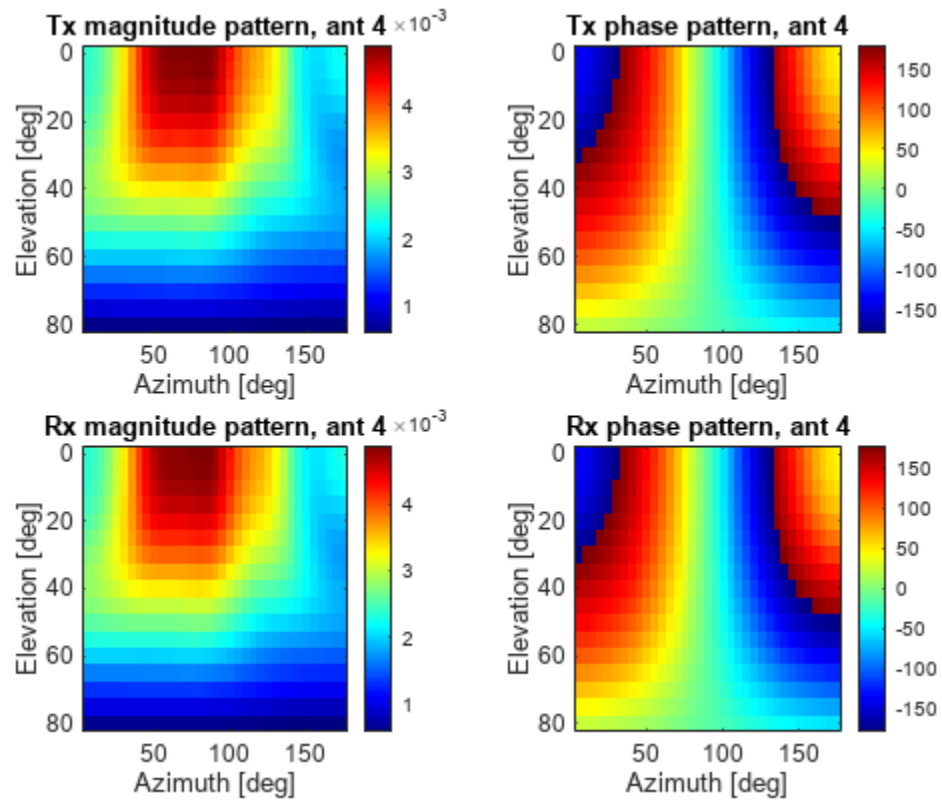
subplot(224)
imagesc(phi,psi,squeeze(ERXphase(i,:,:)));
colorbar
colormap('jet')
xlabel('Azimuth [deg]','LineWidth',6);
ylabel('Elevation [deg]','LineWidth',6);
title(['Rx phase pattern, ant ',num2str(i)],'FontSize',10);
end

```

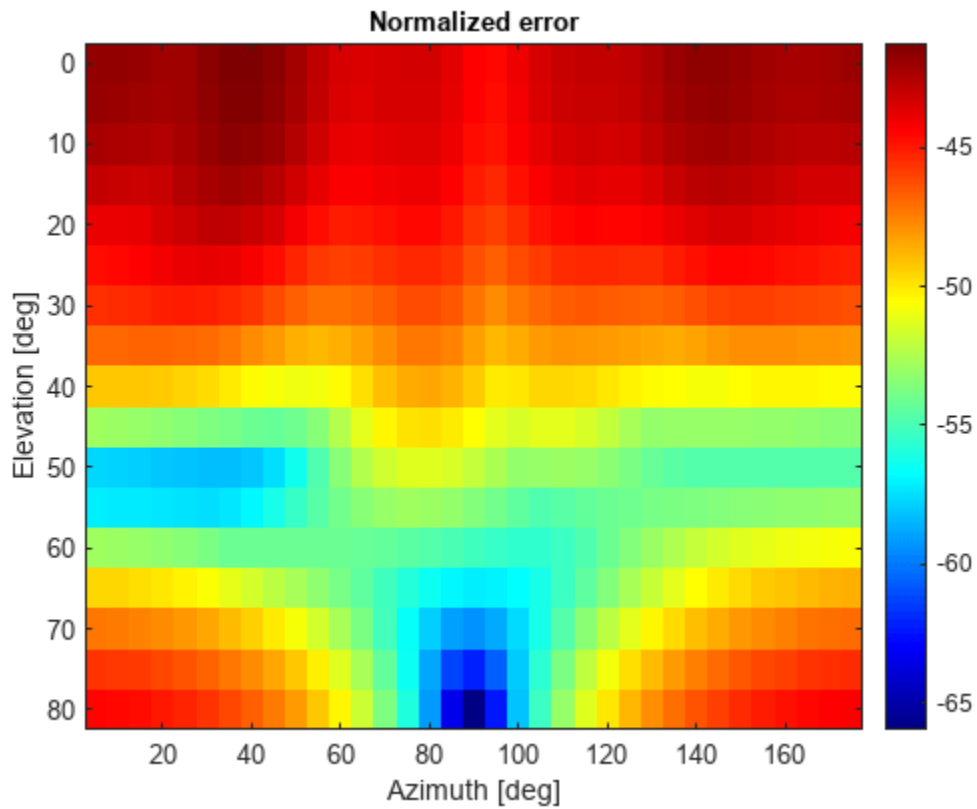








```
figure;  
imagesc(phi,psi,ERR');  
colorbar  
colormap('jet')  
xlabel('Azimuth [deg]','LineWidth',6);  
ylabel('Elevation [deg]','LineWidth',6);  
title('Normalized error','FontSize',10)
```

Acknowledgement

This example was developed in collaboration with Prof. Benjamin Friedlander at the University of California, Santa Cruz.

See Also

“Subarrays in Large Finite Array For Hybrid Beamforming” on page 5-562 | “Planning a 5G Fixed Wireless Access Link over Terrain” on page 5-432

Verification of Far-Field Array Pattern Using Superposition with Embedded Element Patterns

This example shows that the far-field radiation pattern of a fully excited array can be recreated from the superposition of the individual embedded patterns of each element. The pattern multiplication theorem in array theory states that the far-field radiation pattern of an array is the product of the individual element pattern and the array factor. In the presence of mutual coupling, the individual element patterns are not identical and therefore invalidates the result from pattern multiplication. However, by computing the embedded pattern for each element and using superposition, we can show the equivalence to the array pattern under full excitation.

Set up Frequency and Array parameters

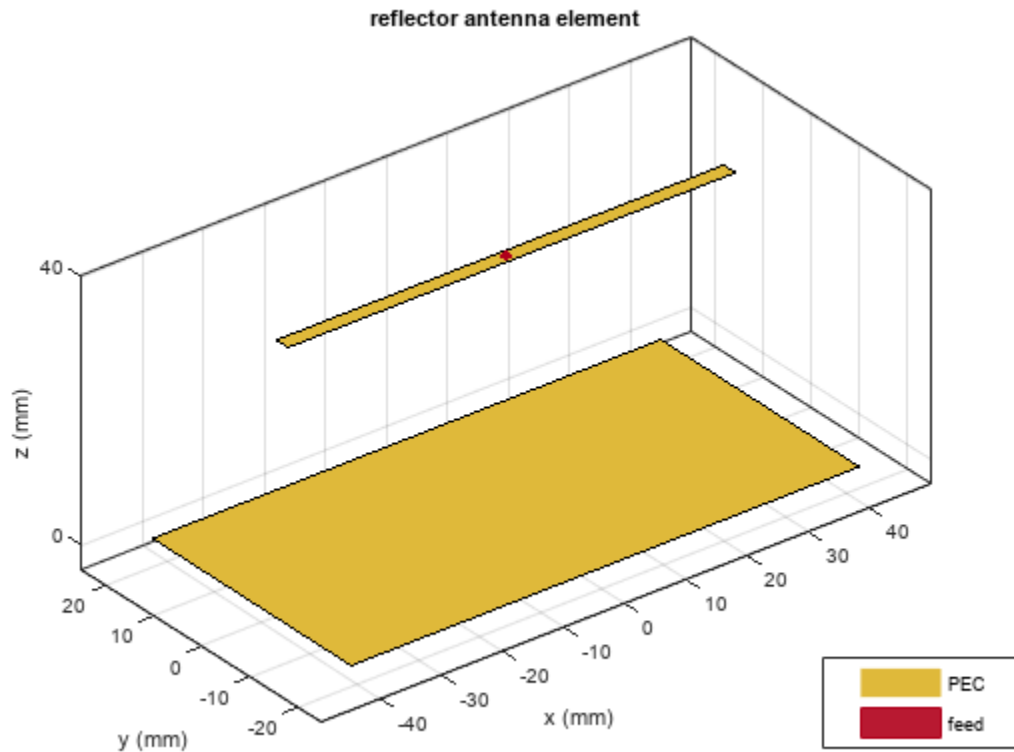
Choose the design frequency to be 1.8 GHz, which happens to be one of the carrier frequencies for 3G/4G cellular systems. Define array size using number of elements, N and inter-element spacing, dx .

```
fc = 1.8e9;  
vp = physconst('lightspeed');  
lambda0 = vp/fc;  
N = 4;  
dx = lambda0/2;
```

Design Antenna Element and Create the Array

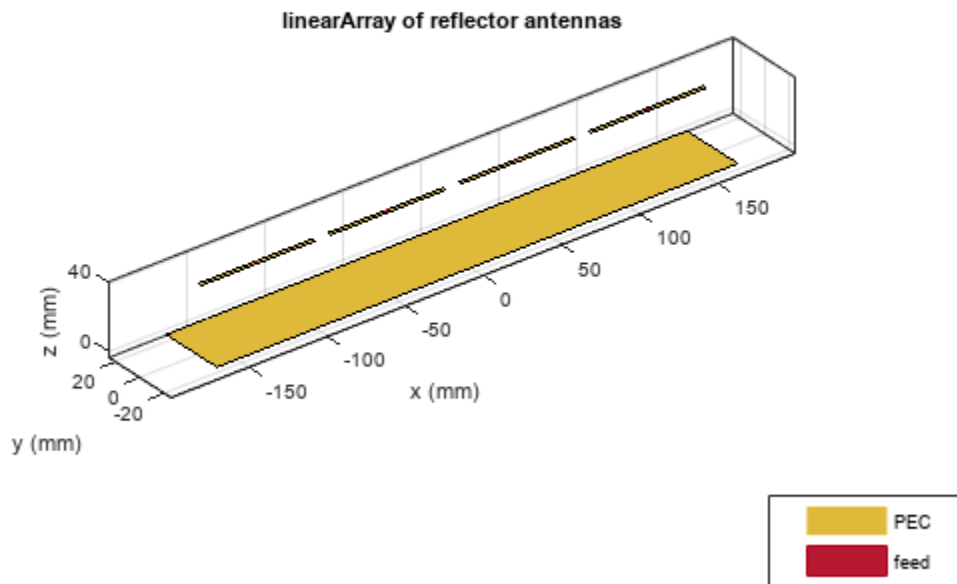
For this example, we design a reflector backed half-wavelength dipole antenna. The reflector is half-wavelength in length along the x-axis and a quarter-wavelength in width, along the y-axis.

```
r = design(reflector,fc);  
r.GroundPlaneLength = lambda0/2;  
r.GroundPlaneWidth = lambda0/4;  
figure  
show(r)
```



Use the reflector backed dipole as the individual element for the linear array. Use the NumElements property to change the linear array to have 4 elements instead of the default of 2. Change the element spacing to be half-wavelength.

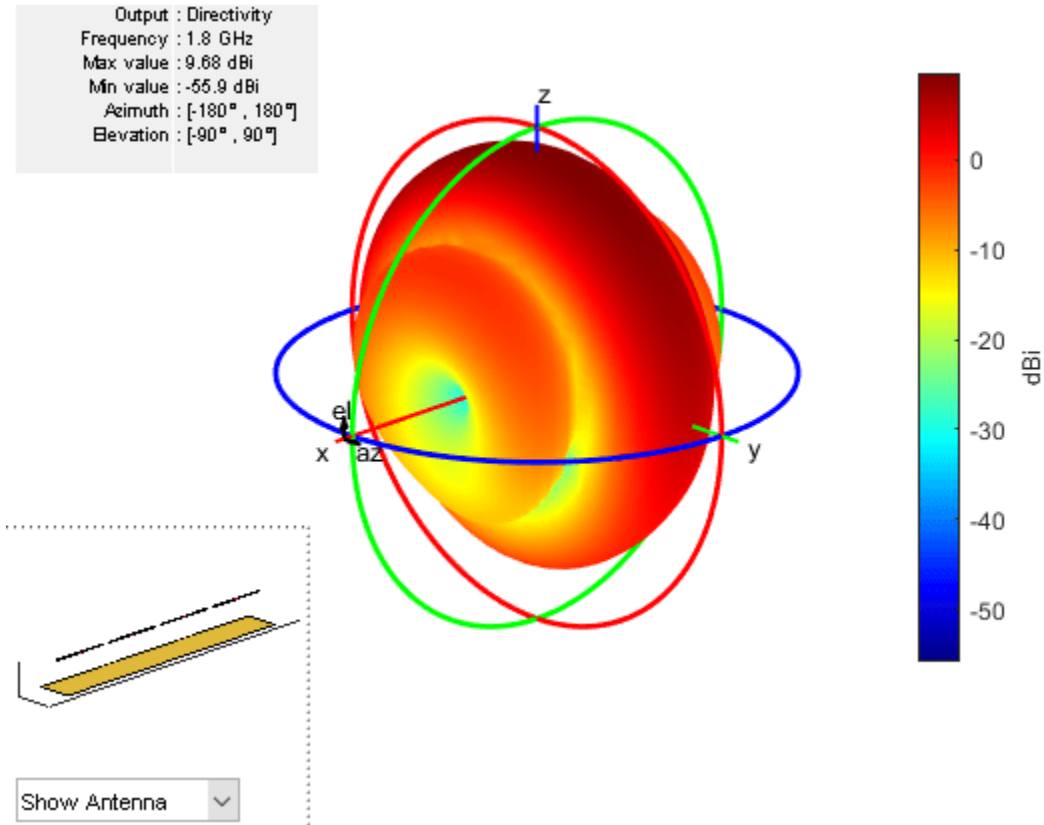
```
lA = linearArray;  
lA.Element = r;  
lA.ElementSpacing = dx;  
lA.NumElements = N;  
figure  
show(lA)
```



Calculate and Plot the 3D Array Pattern

By default all four elements in this array are excited with a voltage of 1V at a phase of 0 deg. Compute the far-field directivity pattern of this uniformly excited array at the center frequency.

```
figure  
pattern(lA,fc)
```

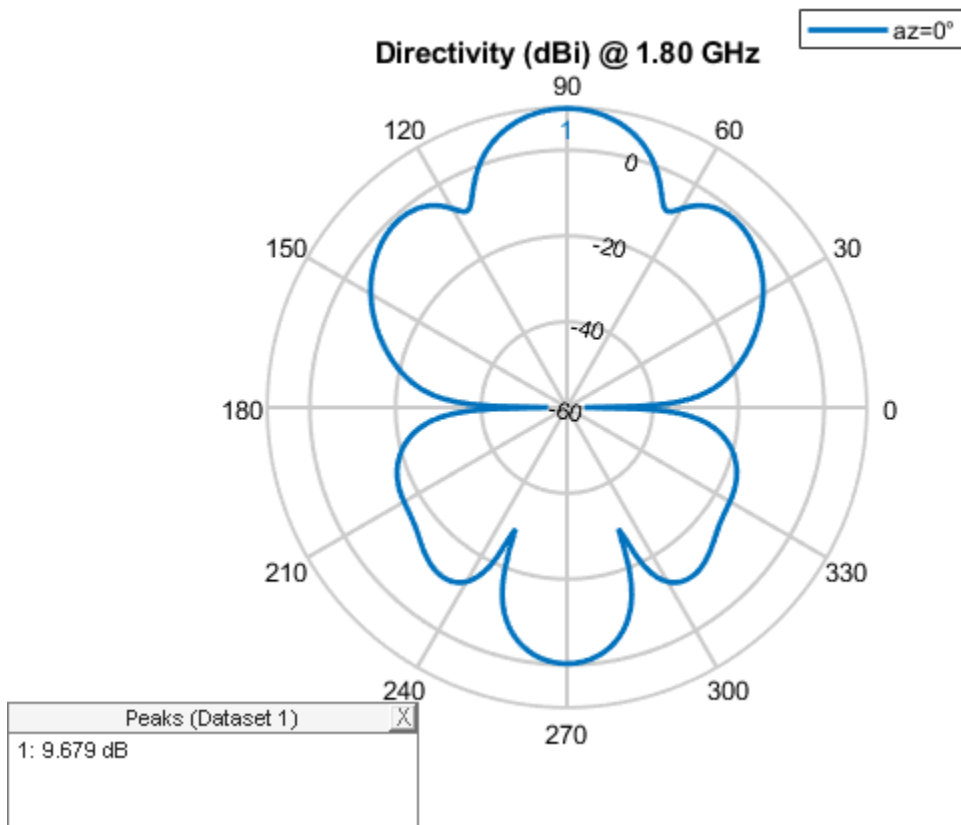


E and H-Plane Pattern Variation of the Fully Excited Array

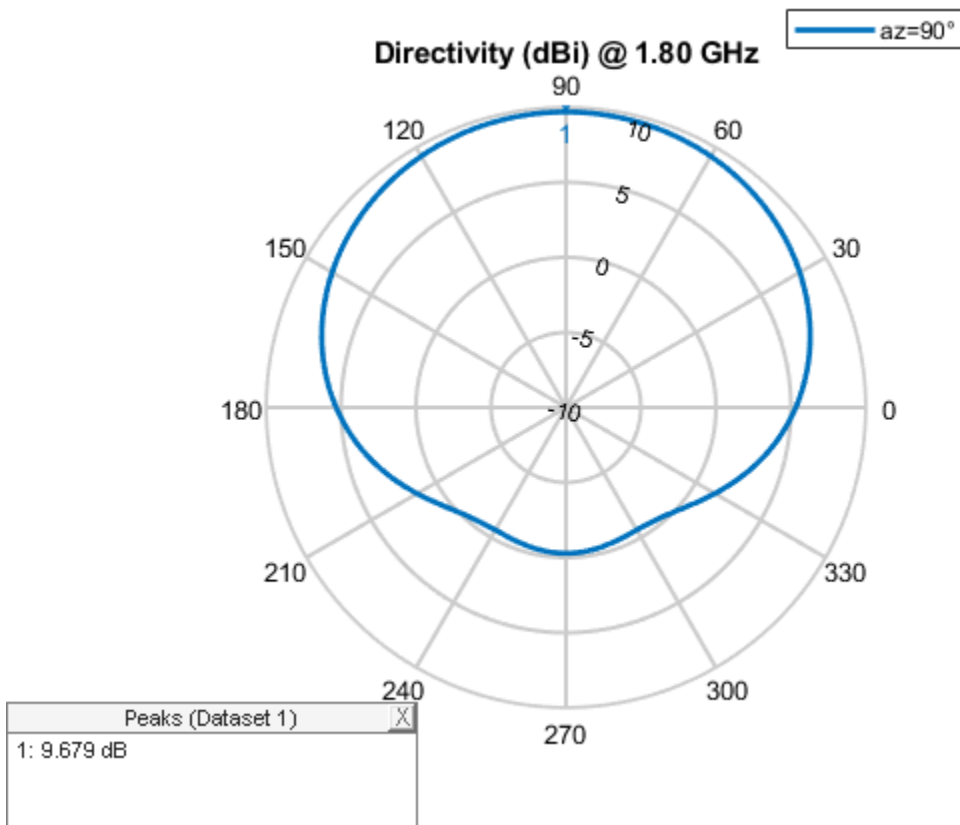
The array being situated in the x-y plane results in most of the radiation being directed towards the zenith. The array pattern variations along the elevation angles can be captured along two orthogonal azimuth slices; at azimuth of 0° and at 90° . Visualize the directivity variation with elevation angle in these two planes is using the `polarpattern` function.

```
az = 0:5:360;  
el = -180:1:180;
```

```
figure  
patternElevation(lA,fc,theta);
```



```
pE = polarpattern('gco');  
figure  
patternElevation(lA,fc,90);
```



```
pH = polarpattern('gco');
```

Calculate Embedded Element Complex Far-Fields

The embedded element pattern refers to the pattern of a single element embedded in the finite array, that is calculated by driving the central element in the array and terminating all other elements into a reference impedance [1]-[3]. The pattern of the driven element, referred to as the embedded element, incorporates the effect of coupling with the neighboring elements. In the Antenna Toolbox™, an ideal voltage source is used as excitation. To recreate the far-field pattern from superposition of the complex far-fields, use a very small value of resistance to terminate the remaining elements. Secondly, the superposition must be done on the complex far-field. Use the EHfields function to calculate the complex electric and magnetic fields at different points in space due to each excited element. For this example, choose a spherical arrangement of points in the E and H-plane angles defined earlier. The far-field points are computed at a radius of 100λ .

```
R = 100*299792458/min(fc);
phil = az;
thetal = 90-el;
[theta, phi] = meshgrid(thetal, phil);
phi = phi(:);
theta = theta(:);
X = R.*sind(theta).*cosd(phi);
Y = R.*sind(theta).*sind(phi);
Z = R.*cosd(theta);
Points = [X';Y';Z'];
N = \A.NumElements;
E = zeros(3,size(Points,2),N);
```

```

for i = 1:N
    E(:, :, i) = EHfields(lA, fc, Points, 'ElementNumber', i, 'Termination', 1e-12);
end

```

Superposition of Embedded Element Pattern Fields

Combine the individual embedded element electric field patterns in the far-field. For the sake of comparison with the pattern of the fully excited array, compute the magnitude. This will be used to calculate the total directivity in the E and H-plane respectively.

```

arrayEfieldpat = sum(E, 3);
MagEsquare = dot(arrayEfieldpat, arrayEfieldpat);
MagE = sqrt(MagEsquare);
MagE = reshape(MagE, length(az), length(el));

```

Compute Directivity of Array

Directivity is a measure of the power projection ability of an antenna or array as a function of different angles in space. It defines the overall shape of the power projection capability of the radiating structure. To calculate this, find the radiation intensity in particular directions and divide it by the total radiated power from the structure over all directions. The total radiated power is computed as a product of the radiation efficiency and the input power. Each element of the array is assumed to be excited by a 1 Volt excitation source for computing the input power. The radiation efficiency of the array is computed using the `efficiency` function.

```

RadEff = efficiency(lA, fc);
InputPower = sum(0.5*real(1./conj(impedance(lA, fc))));
RadiatedPower = RadEff*InputPower;
eta = sqrt(1.25663706e-06/8.85418782e-12);
U = R^2*MagE.^2/(2*eta);
D = 10*log10(4*pi*U/RadiatedPower);

```

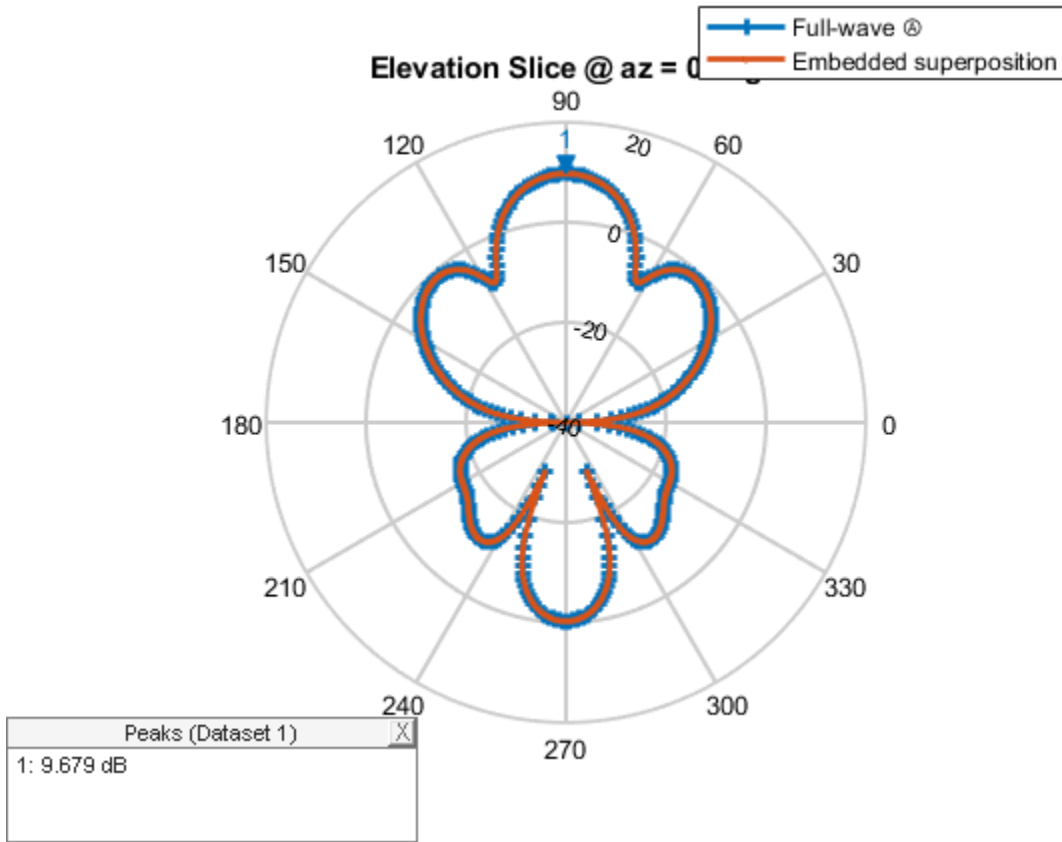
Comparison of Patterns

Overlay the directivity result from the superposition of the embedded element patterns on the result from the computation for the fully excited array.

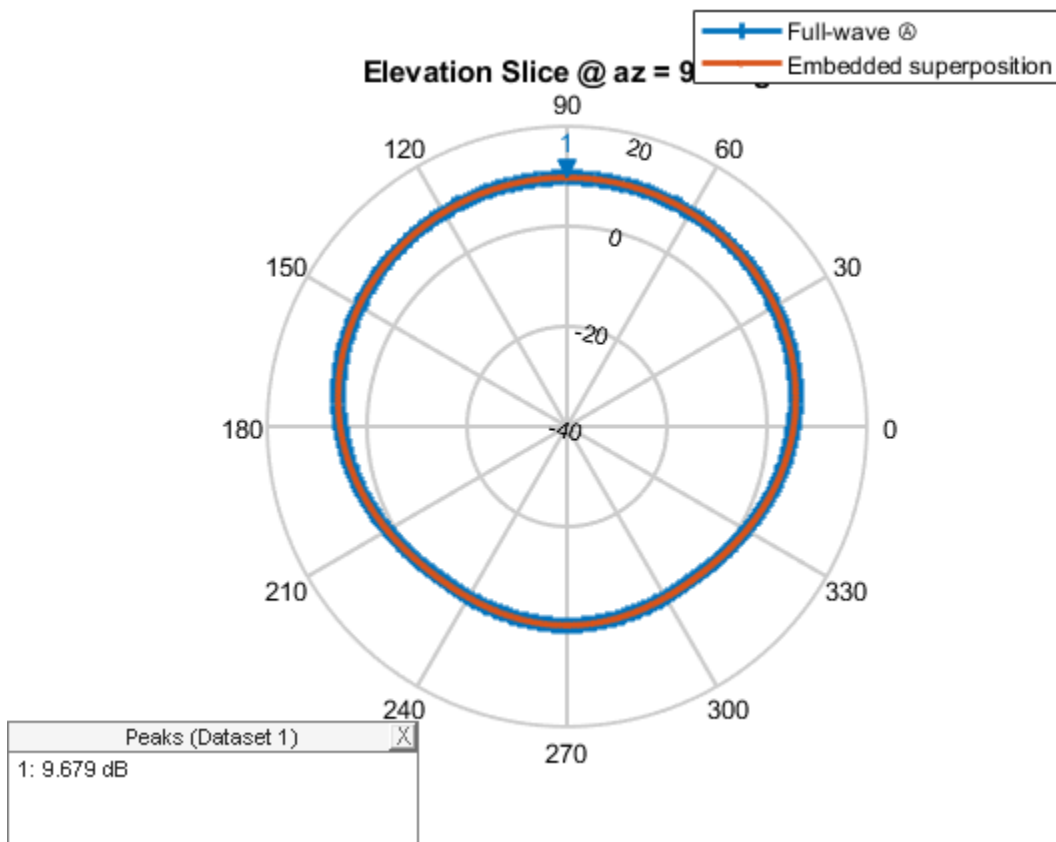
```

idphi0 = find(az==0);
idphi90 = find(az==90);
Dphi = D(idphi0, :);
Dphi90 = D(idphi90, :);
add(pE, el, Dphi);
pE.LegendLabels = {'Full-wave', 'Embedded superposition'};
pE.MagnitudeLim = [-40 20];
pE.Marker = {'+', '.'};
pE.TitleTop = 'Elevation Slice @ az = 0 deg';

```

```
add(pH,e1,Dphi90);
pH.LegendLabels = {'Full-wave','Embedded superposition'};
pH.MagnitudeLim = [-40 20];
pH.Marker = {'+','.'};
pH.TitleTop = 'Elevation Slice @ az = 90 deg';
```



Summary

The use of superposition on the complex far-fields produced by the individual elements of an array generates the same pattern as the one from the uniformly excited array.

See Also

“Modeling Mutual Coupling in Large Arrays Using Embedded Element Pattern” on page 5-195

Reference

- [1] R. J. Mailloux, 'Phased Array Antenna Handbook', Artech House, 2nd edition, 2005.
- [2] W. Stutzman, G. Thiele, 'Antenna Theory and Design', John Wiley & Sons Inc., 3rd Edition, 2013.
- [3] R. C. Hansen, Phased Array Antennas, Chapter 7 and 8, John Wiley & Sons Inc., 2nd Edition, 1998.

Metasurface Antenna Modeling

This example shows how to design, model, and analyze a metasurface antenna based on reference [1]. Metasurface antennas take advantage of the periodic boundary of each radiation unit cell to group the radiation unit in a more compact space.

You construct an unit radiation element with Antenna Toolbox™ objects and functions. You also use `infiniteArray` and `rectangularArray` objects to construct the large antenna array, on which you perform pattern and current analysis at the designed frequency.

Construct Unit Radiator

The unit radiator in [1] is a cross I-beam structure with an optimized probe feed in one arm. The unit cell has a dimension of 3 cm and gap of 0.5 mm. Specify the geometry of the antenna.

```
length = 14.75*1e-3;
width = 3*1e-3;
s = 0.25*1e-3;
thickness = 1.5*1e-3;
viaDia = 0.5*1e-3*2;
feedWidth = viaDia/2;
gndLength = length + 2*s;
gndWidth = gndLength;
```

Use the `customAntennaGeometry` object to construct the top cross I-beam radiator and place it into a reflector to form the unit radiator structure. In this simplified geometry, you remove the substrate from the design. Removing the substrate increases the radiation frequency to 9.14 GHz, which is higher than the value in [1].

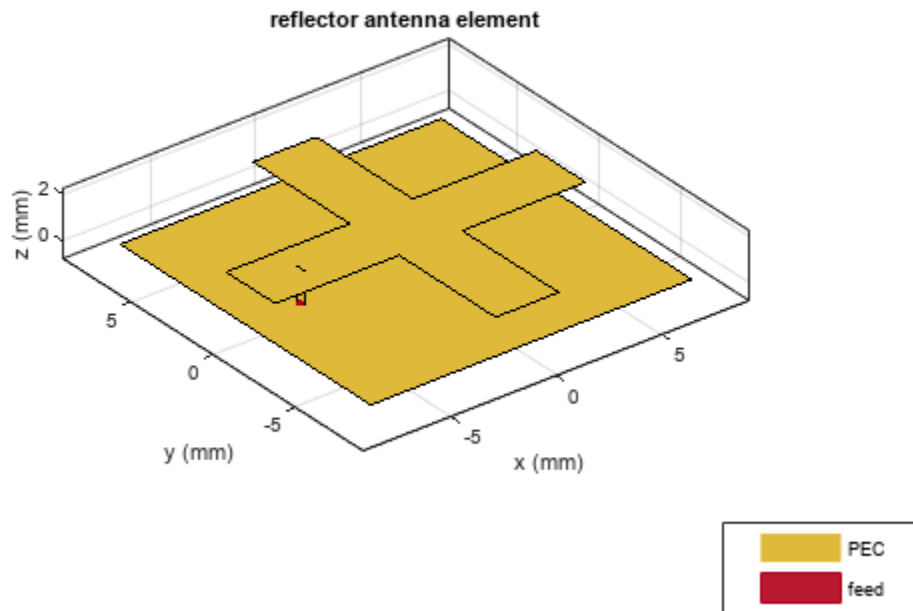
```
s1 = antenna.Rectangle(Length=length, Width=width);
s2 = antenna.Rectangle(Length=width, Width=length);
s3 = antenna.Rectangle(Length=feedWidth, Width=feedWidth);
[~] = translate(s3,[-5*1e-3+feedWidth/2 0 0]);

pr1 = getShapeVertices(s1);
pr2 = getShapeVertices(s2);
pr3 = getShapeVertices(s3);

radiator = customAntennaGeometry(Boundary={pr1,pr2,pr3}, Operation="P1+P2+P3");
radiator.FeedLocation = [-5*1e-3 0 0];
radiator.FeedWidth = feedWidth;

ant = reflector(Exciter=radiator, GroundPlaneLength=gndLength, GroundPlaneWidth=gndWidth, ...
    Spacing=thickness, EnableProbeFeed=true);

figure
show(ant)
```

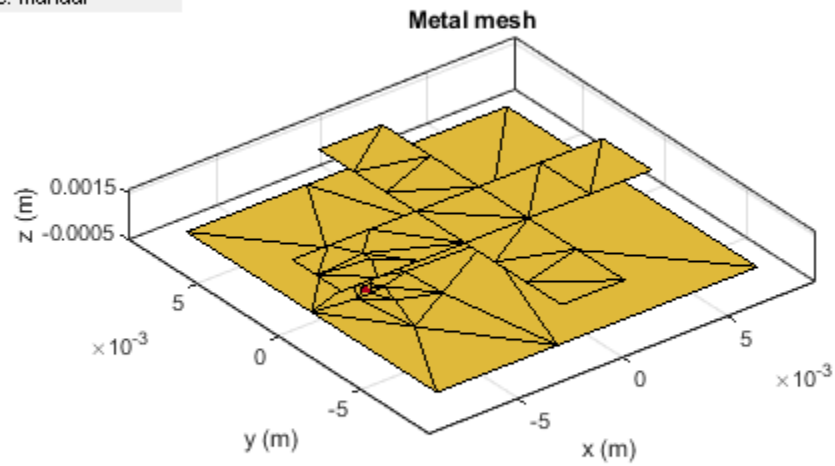


Mesh and Analyze Unit Structure

Generate a mesh with a maximum edge length of 0.1 m.

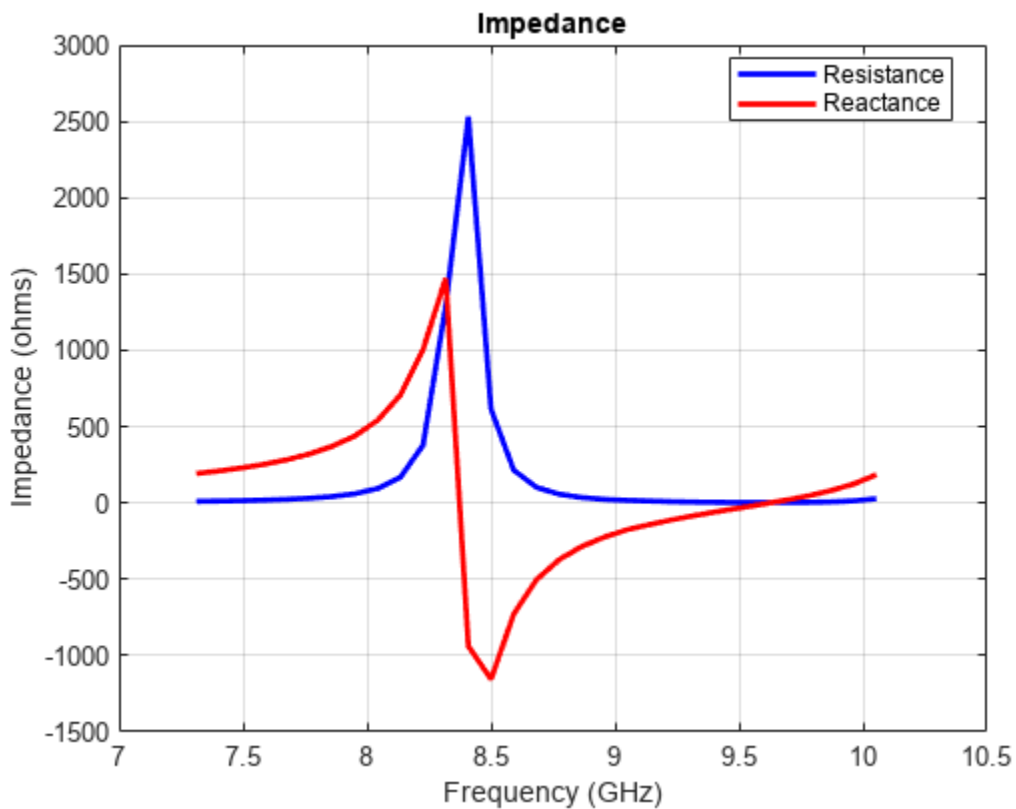
```
figure;  
mesh(ant,MaxEdgeLength=0.1);
```

NumTriangles: 60
NumTetrahedra: 0
NumBasis:
MaxEdgeLength: 0.1
MeshMode: manual



Calculate the impedance as a function of frequency.

```
freq = 9.14e9;  
figure;  
impedance(ant, freq*[0.8:0.01:1.1]);
```



Implement Periodic Boundary and Analyze Antenna

To include the periodic boundary effect on the unit radiator, specify the radiator as the element in the `infiniteArray` object. The object constructs an infinite antenna array with the designed radiator. Numerically, the object uses a Green function with a method of moments (MOM) and models the coupling effect between adjacent unit radiators.

```
infArray = infiniteArray(Element=ant);  
figure  
show(infArray)
```

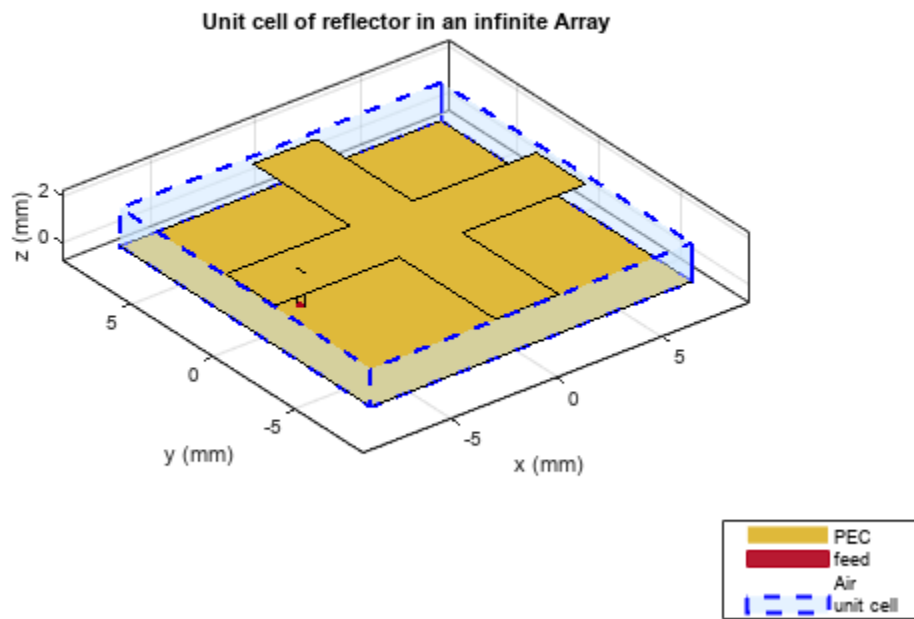
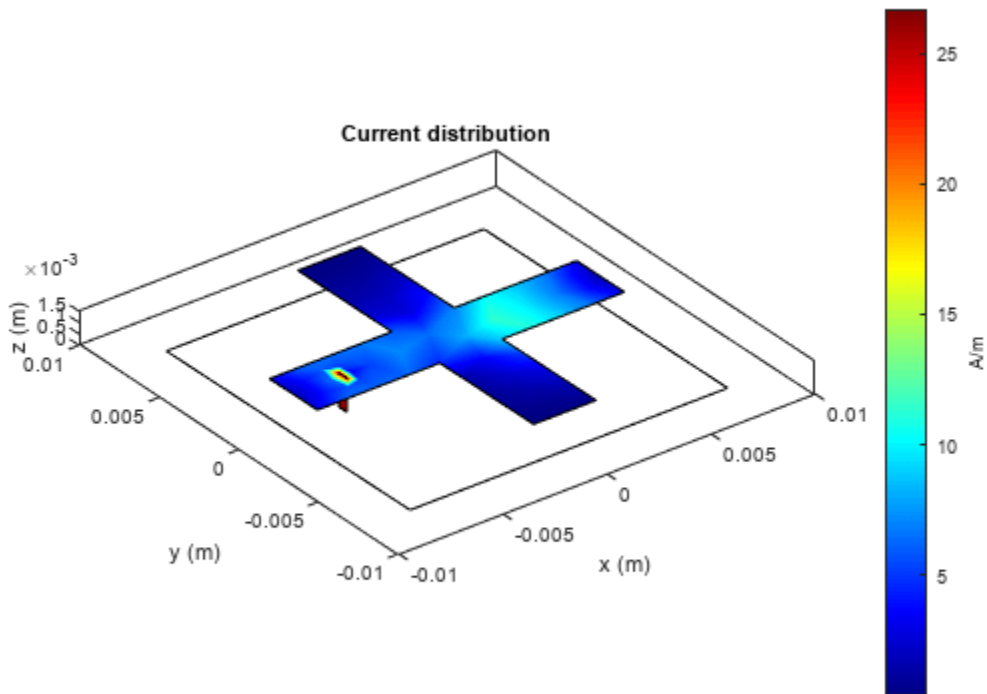


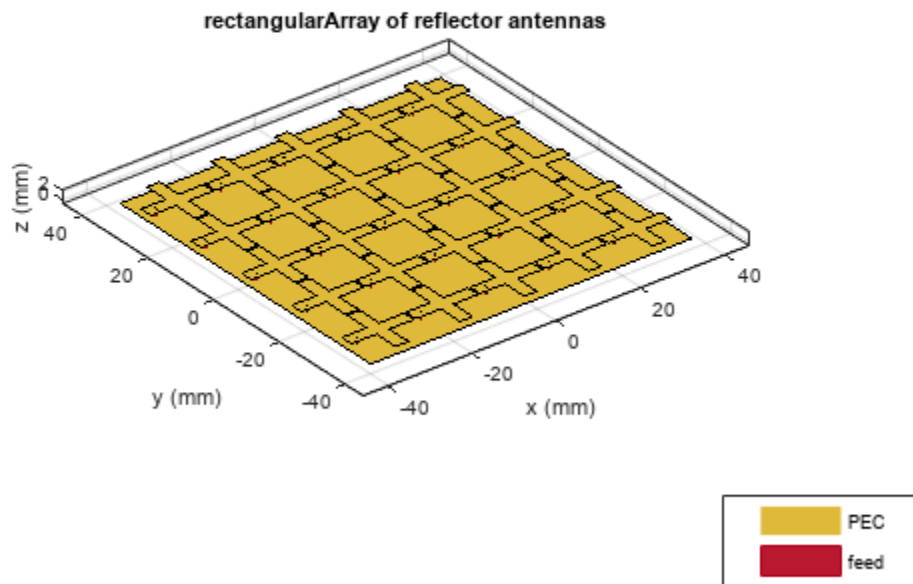
figure
current(infArray, freq)



Analyze Finite Array

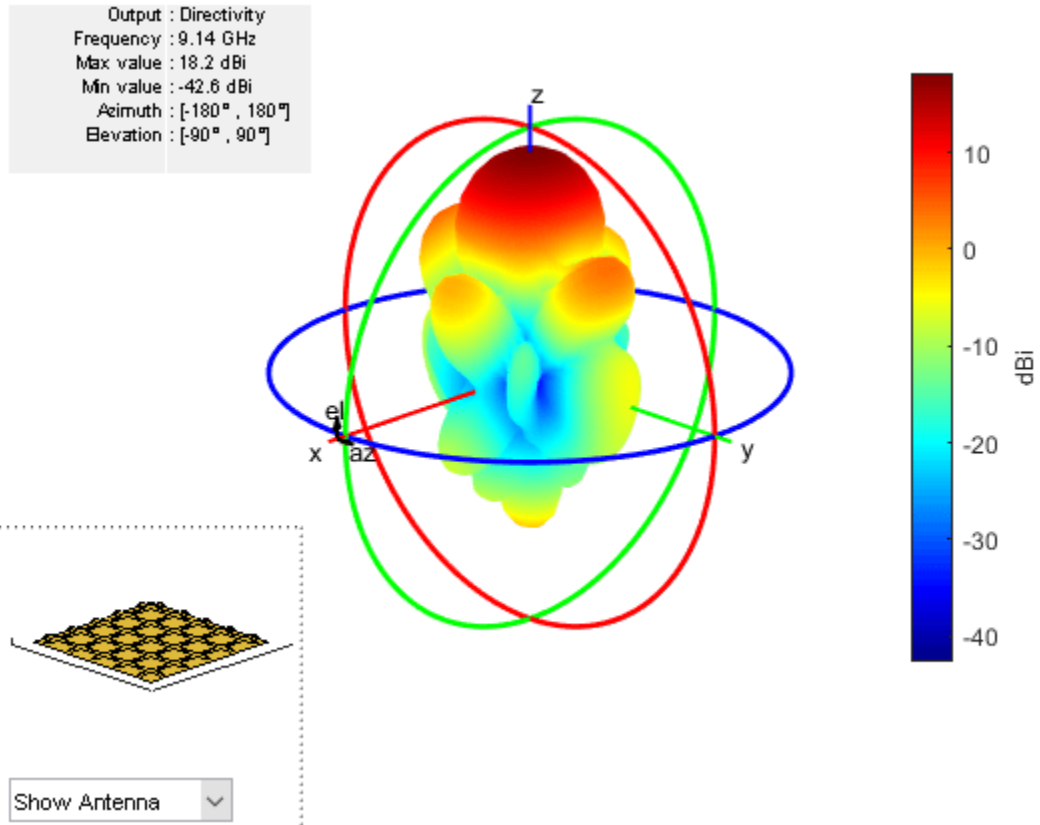
To analyze the coupling effect on a finite array, construct a 5-by-5 element rectangular array with the designed radiator.

```
array = rectangularArray(Element=ant);  
array.Size = [5 5];  
array.RowSpacing = gndLength;  
array.ColumnSpacing = gndWidth;  
figure  
show(array)
```

Plot the pattern and current of the array as functions of the frequency.

```
figure  
pattern(array, freq)
```



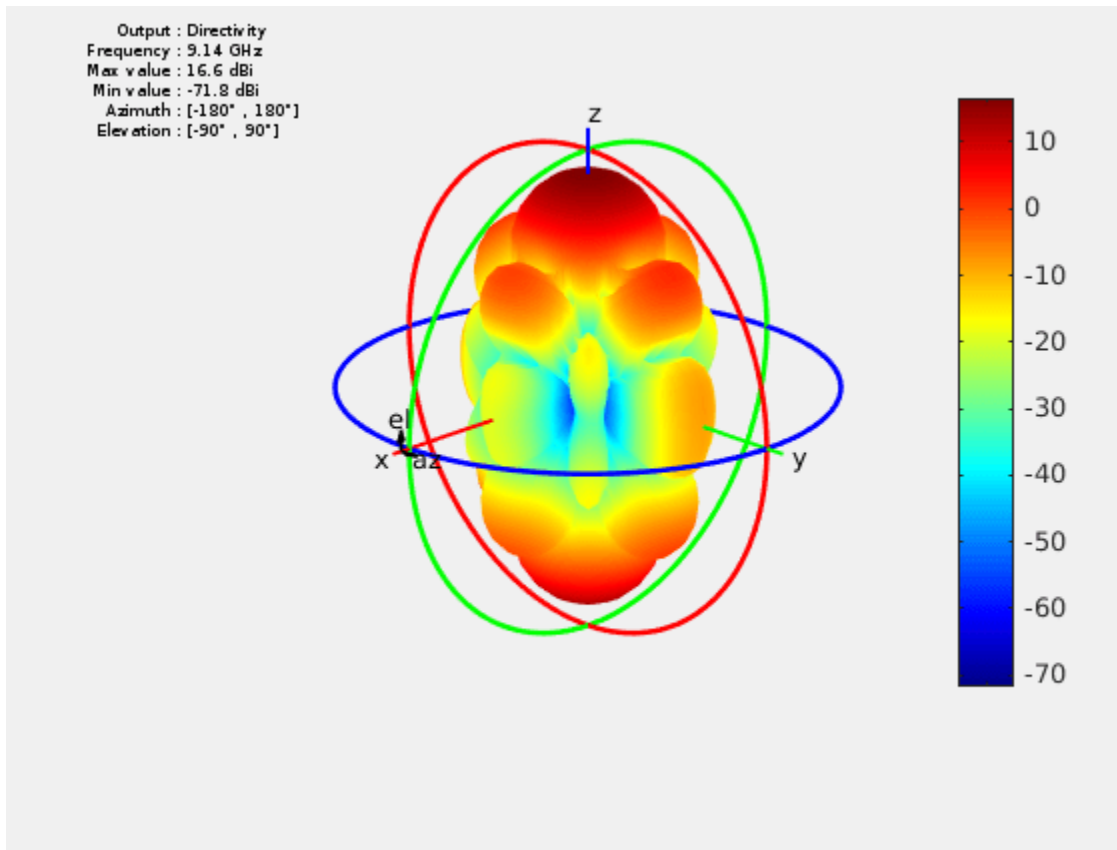
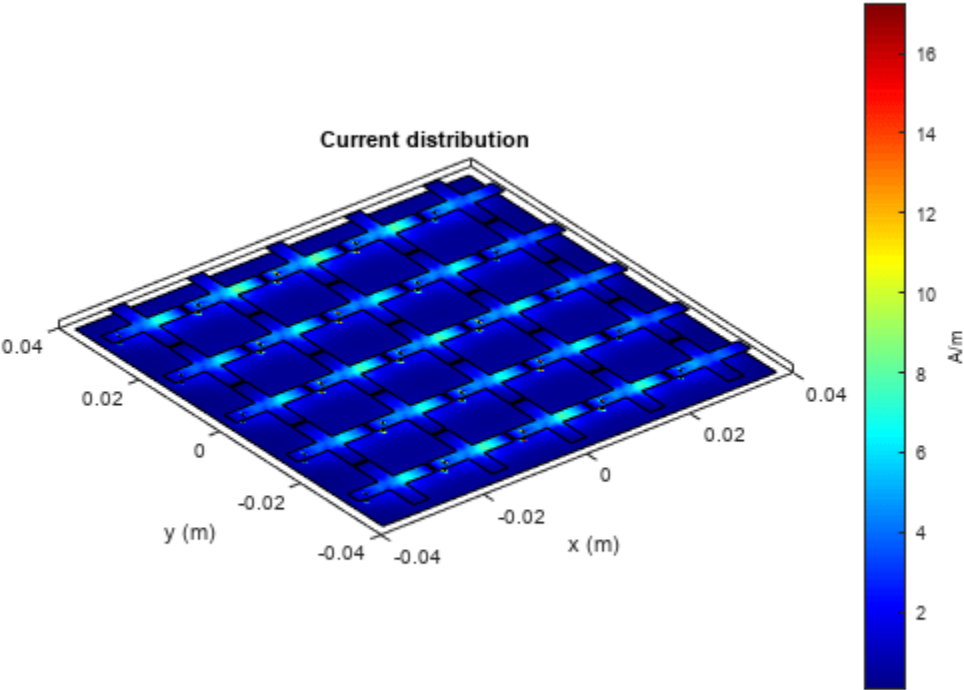
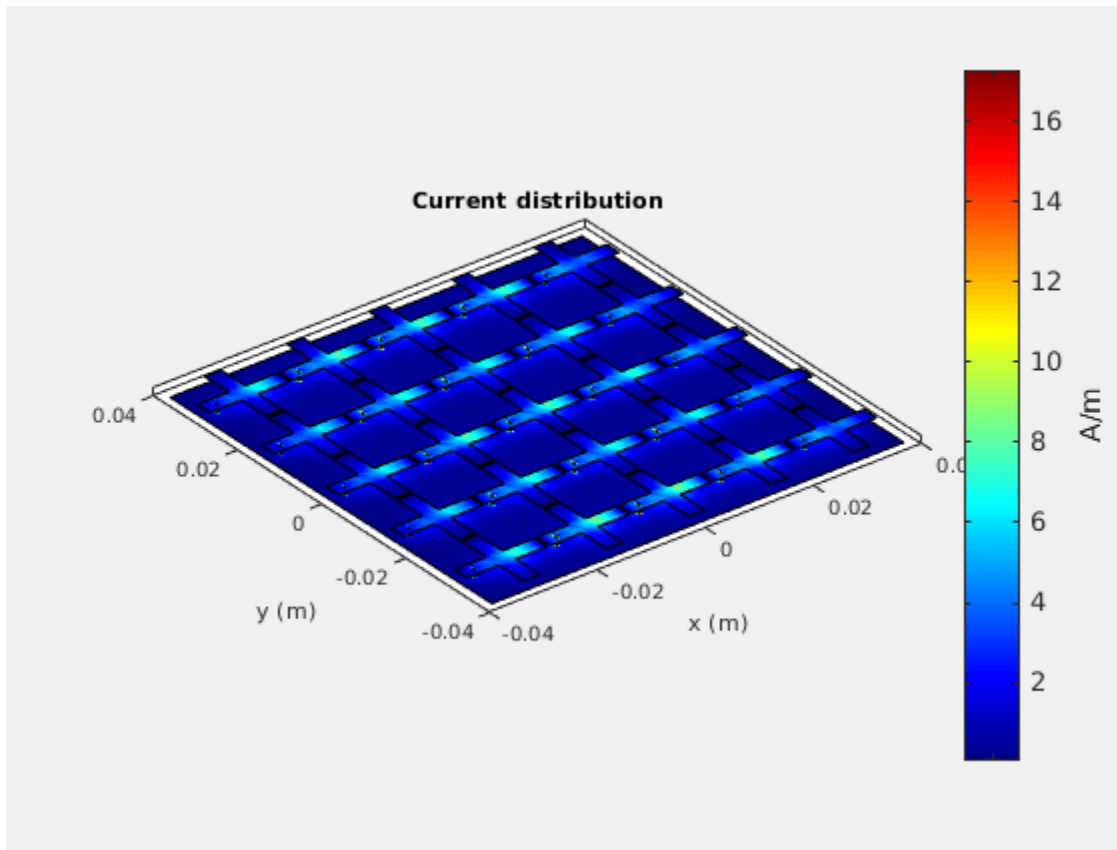


figure
current(array, freq)

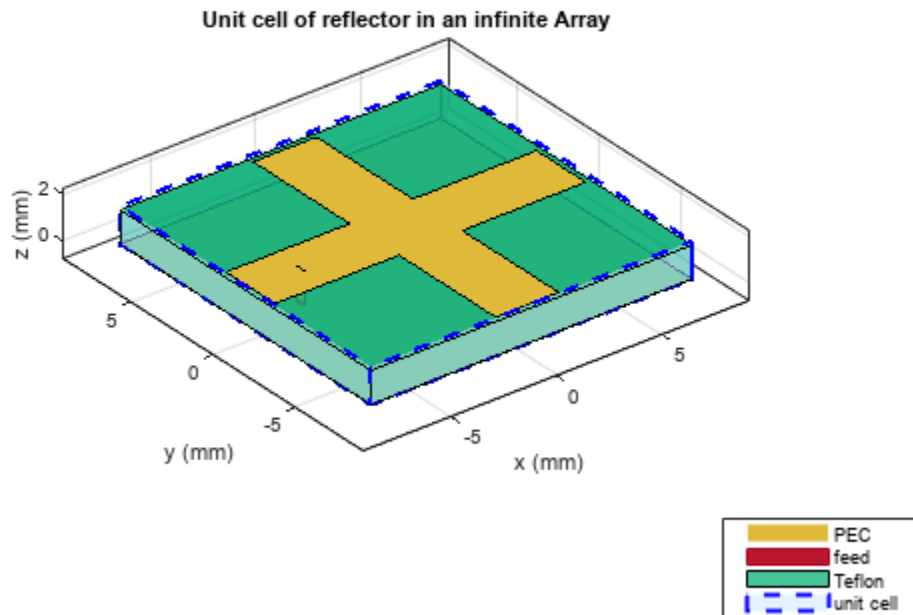




Assign Substrate to Infinite Array

Specify Teflon™ as the substrate of the unit cell element and visualize the geometry of the infinite array.

```
infArray.Element.Substrate = dielectric("Teflon");  
figure  
show(infArray)
```

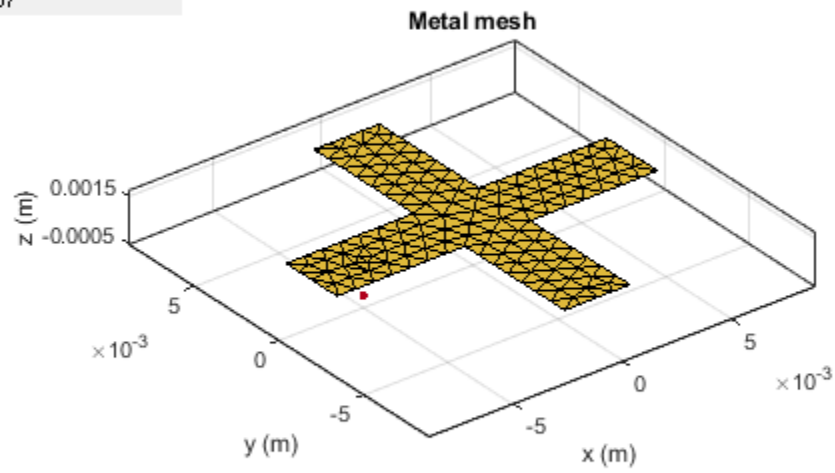


Mesh Unit Cell

Mesh the unit cell, specifying a maximum edge length of $\lambda/40$, where λ is the free-space wavelength at 9.14 GHz. The software captures the dielectric effect in the infinite array analysis in the equivalent periodic Green's function, so use the MOM over the top layer metal mesh only.

```
lambda = 3e8/9.14e9;
figure
mesh(infArray,MaxEdgeLength=lambda/40)
```

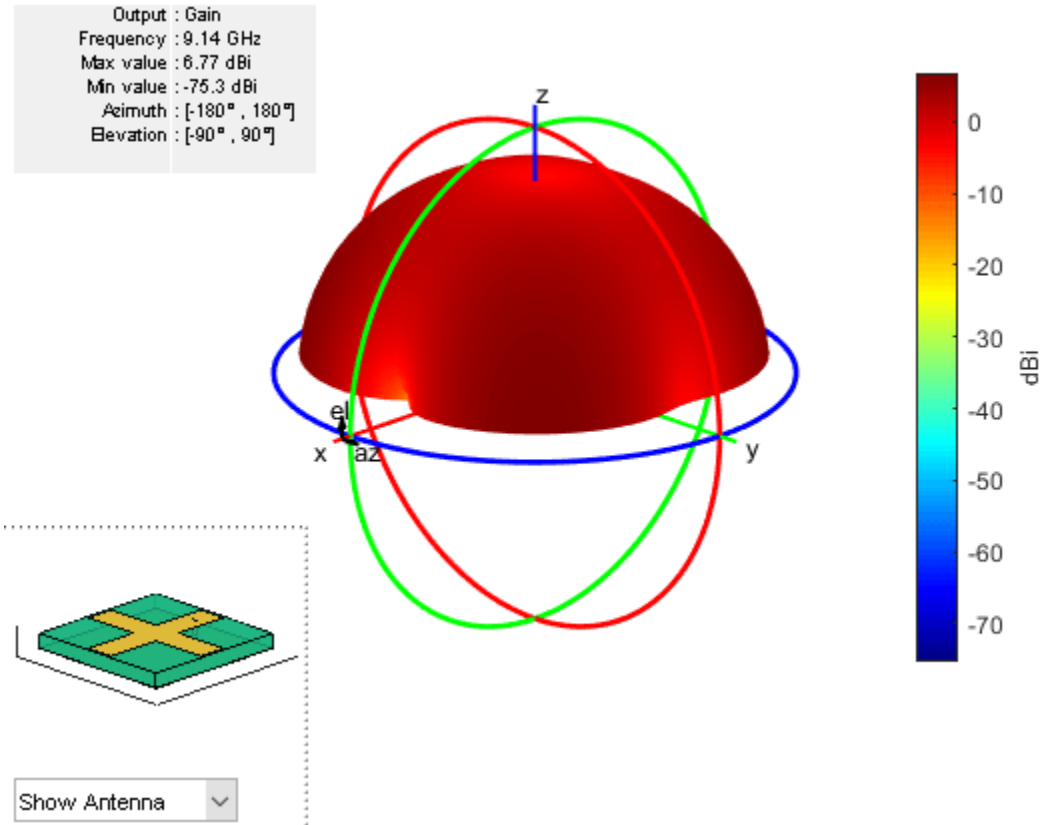
NumTriangles: 128
NumTetrahedra: 0
NumBasis:
MaxEdgeLength:
0.00082057



Compute Pattern

With the meshed geometry, perform the full-wave radiation pattern analysis at 9.14 GHz.

```
figure  
pattern(infArray,9.14e9);
```



Conclusion

The current and pattern modeling result obtained from the finite array are qualitatively similar to the results reported in [1]. For other types of metasurface geometry, you can construct the unit cell by using an antenna catalog element, as the “Infinite Array of Microstrip Patch Antenna on Teflon Substrate” on page 5-780 example shows, or using a PCB stack.

References

[1] Badawe, Mohamed El, Thamer S. Almoneef, and Omar M. Ramahi. “A True Metasurface Antenna.” Scientific Reports 6, no. 1 (January 13, 2016): 19268. <https://doi.org/10.1038/srep19268>.

See Also

“Modeling Resonant Coupled Wireless Power Transfer System” on page 5-209

More About

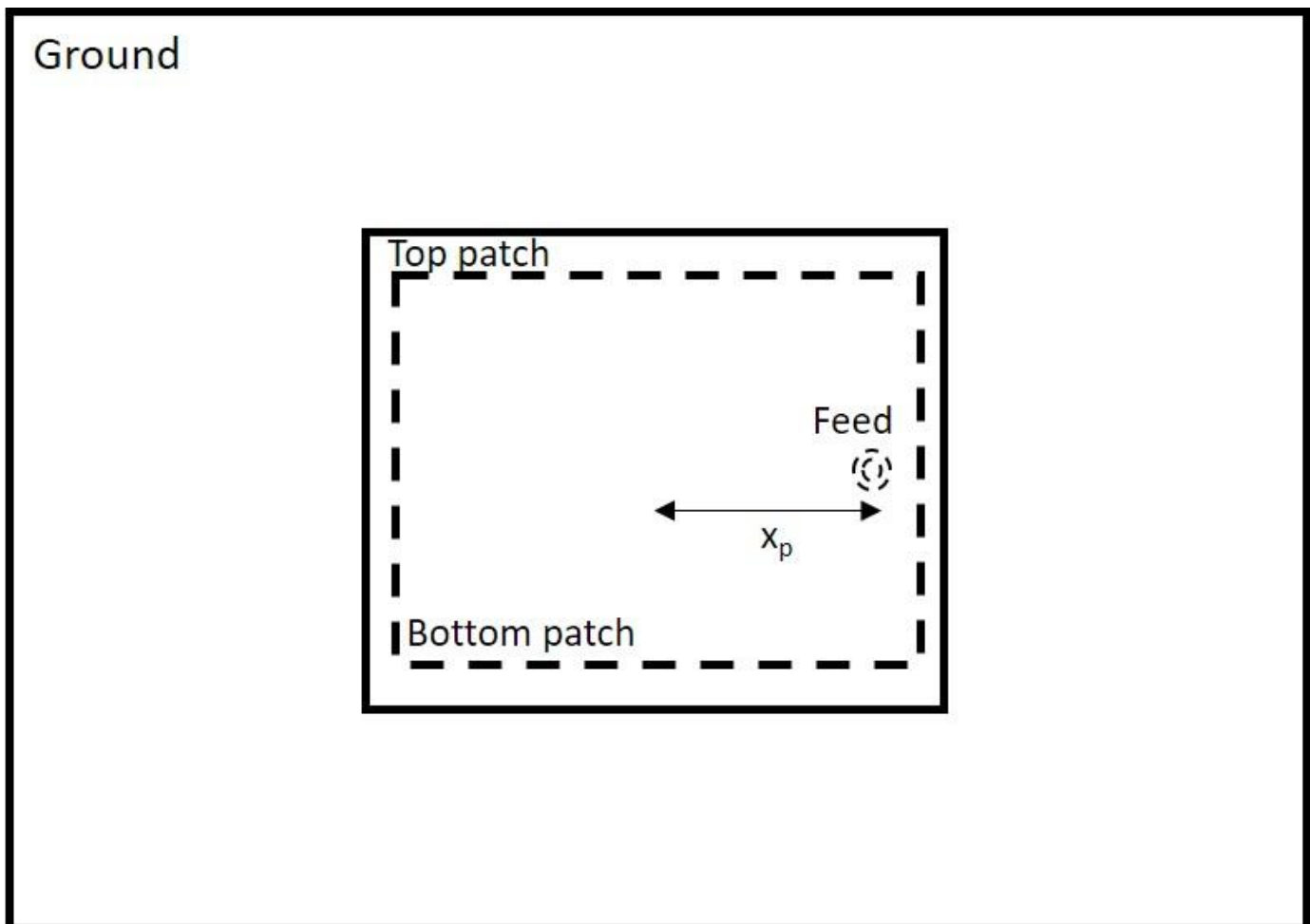
- “What Are Infinite Arrays?” on page 2-22

Modeling and Analysis of Probe-Fed Stacked Patch Antenna

This example shows the steps to model and analyze a probe-fed stacked patch antenna. The standard rectangular microstrip patch antenna has a narrow impedance bandwidth typically lesser than 5%. The stacked patch configuration is one of the ways of increasing the impedance bandwidth of these antennas to be greater than 25% [1]. There are different ways of designing stacked patches, primarily differing in the way their feed is designed [2]. The two types of feeding mechanisms are probe-feed and aperture coupled. These two mechanisms have a role in the impedance bandwidth behavior as well as the radiation characteristics of the antenna.

Probe-Fed Stacked Patch Geometry

The stacked patch comprises of two patches of slightly different sizes positioned over each other along the z-axis and separated by a dielectric material. Both patches are centered relative to the groundplane. The gap between the lower patch and the groundplane is also filled with a dielectric material. Either the top or the bottom patch is driven with a coaxial probe when used in a single feed configuration. A plan view of the geometry is shown in the sketch here.



Define Units

Define standard units for distance, frequency and resistance as well as their multiplicative equivalents for this example.

```
meter = 1;  
hertz = 1;  
ohm = 1;  
mm = 1e-3*meter;  
GHz = 1e9*hertz;
```

Antenna Dimensions

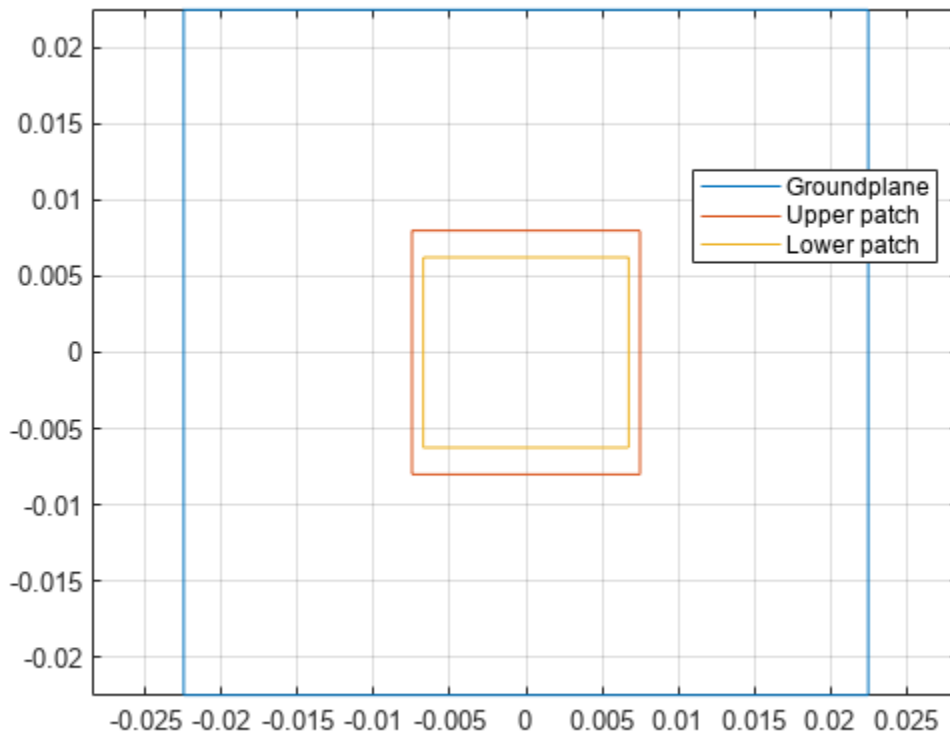
The antenna dimensions are provided in [1] for a probe-fed rectangular stacked patch with two substrate layers. The variables names are identical to those mentioned in [1] barring those of the groundplane. For this example, a square groundplane is chosen with size of three times the length of the top patch. The dimensions of the two patches are chosen to maximize the impedance bandwidth and guidelines are provided in [1] for designing such patch antennas together with a sensitivity analysis. For the geometry being modeled, the upper patch is slightly larger than the lower one.

```
L1 = 13.5*mm;  
W1 = 12.5*mm;  
L2 = 15*mm;  
W2 = 16*mm;  
d1 = 1.524*mm;  
d2 = 2.5*mm;  
xp = 5.4*mm;  
r_0 = 0.325*mm;  
Lgnd = 3*L2;  
Wgnd = 3*L2;
```

Create Layer Shapes and Substrates

Layers: Use the rectangle shape from the catalog to create the three metal layers required for the stacked patch, namely for the upper patch, the lower patch and the groundplane. All the layers are centered about the coordinate axis origin. Plot the layer boundaries to confirm their sizes and positions.

```
pU = antenna.Rectangle(Length=L2,Width=W2);  
pL = antenna.Rectangle(Length=L1,Width=W1);  
pGnd = antenna.Rectangle(Length=Lgnd,Width=Wgnd);  
figure  
plot(pGnd)  
hold on  
plot(pU)  
plot(pL)  
grid on  
legend('Groundplane','Upper patch','Lower patch',Location='best')
```



Dielectric Substrates: The stacked patch antenna in this example has a dielectric substrate between the upper and lower patches as well as between the lower patch and the groundplane. The lower patch has higher relative permittivity than the upper patch. This implies a loose electrical coupling between the two patches.

```

epsr_1 = 2.2;
tandelta_1 = 0.001;
dL = dielectric;
dL.Name = 'Lower sub';
dL.EpsilonR = epsr_1;
dL.LossTangent = tandelta_1;
dL.Thickness = d1;

```

```

epsr_2 = 1.07;
tandelta_2 = 0.001;
dU = dielectric;
dU.Name = 'Upper sub';
dU.EpsilonR = epsr_2;
dU.LossTangent = tandelta_2;
dU.Thickness = d2;

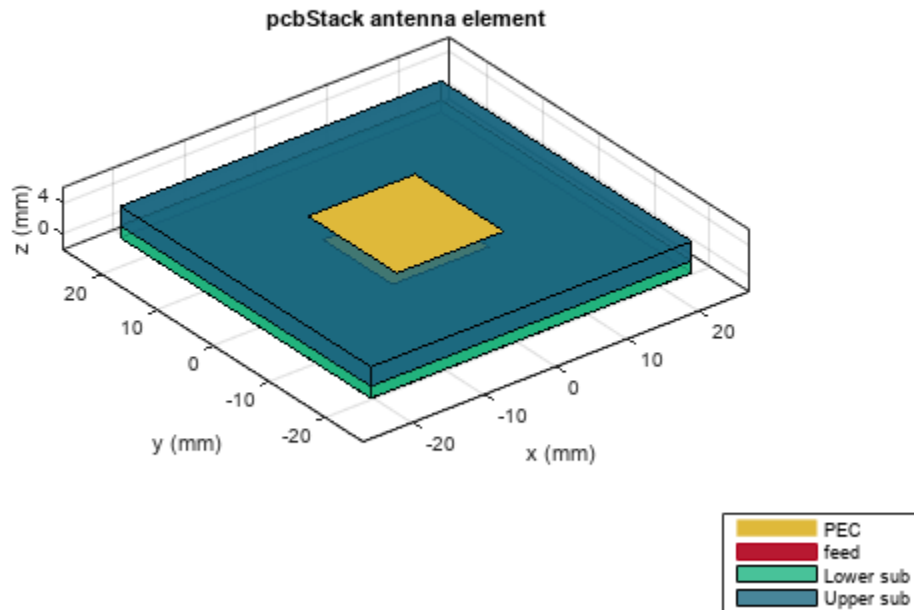
```

Create Stacked Patch Model

Create the stacked patch antenna model by using `pcbStack`. Assign the layers starting from the top-most layer, in this case the metal layer for the upper patch and proceed to the lowest layer which is the groundplane. The probe-feed is specified between the lower patch and the ground plane. To

improve the accuracy of the model, we switch the feed model to be a solid column approximated to a square shape. The default feed model is a strip wherein the strip approximation to a cylinder is used.

```
p = pcbStack;
p.Name = 'Stacked patch - Waterhouse';
p.BoardShape = pGnd;
p.BoardThickness = d1+d2;
p.Layers = {pU,dU,pL,dL,pGnd};
p.FeedLocations = [xp 0 3 5];
p.FeedDiameter = 2*r_0;
p.FeedViaModel = 'square';
figure
show(p)
```



Impedance Analysis

Analyze the stacked patch impedance over the frequency range 6-9 GHz. The stacked patch structure over this range should show two closely spaced parallel resonances. Prior to analysis, mesh the structure

```
fmax = 9*GHz;
fmin = 6*GHz;
deltaf = 0.125*GHz;
freq = fmin:deltaf:fmax;
mesh(p,MaxEdgeLength=0.01,MinEdgeLength=0.003)
```

NumTriangles: 870
NumTetrahedra: 3444
NumBasis:
MaxEdgeLength: 0.01
MeshMode: manual

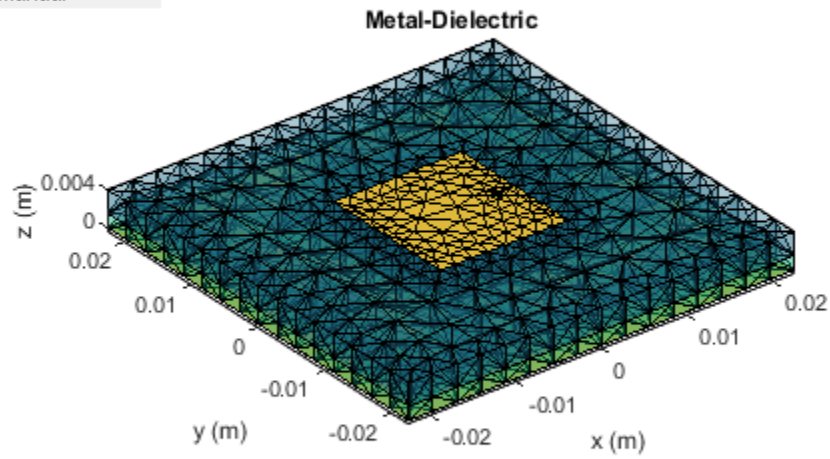
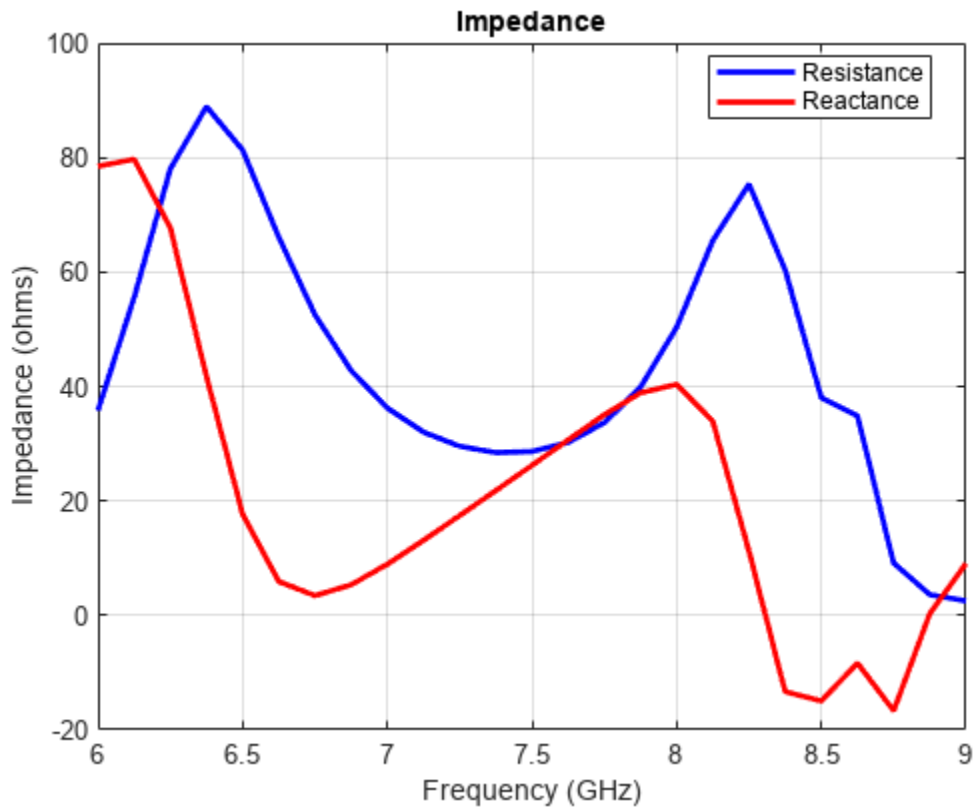


figure
impedance(p, freq)

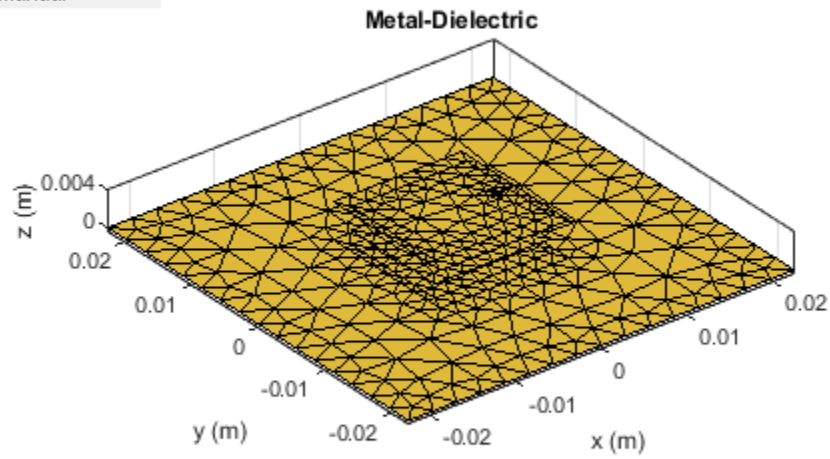


Stacked Patch Mesh

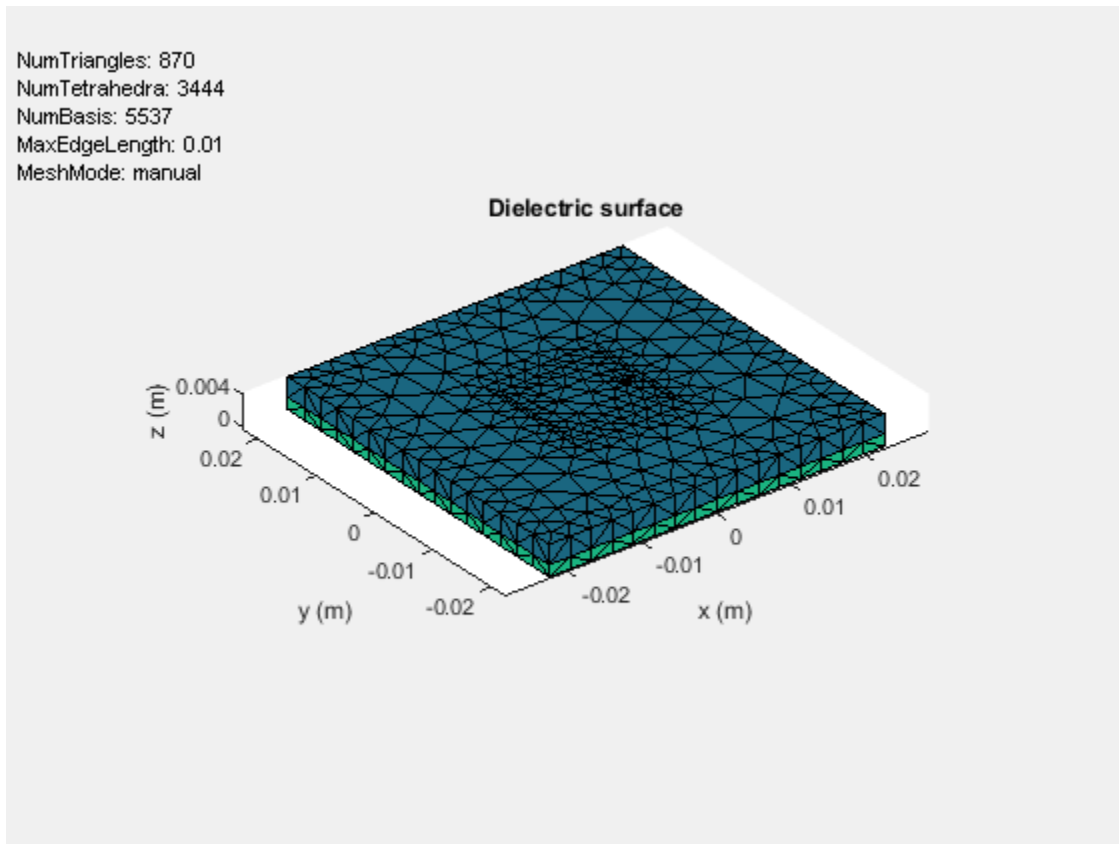
The impedance analysis over the frequency range 6 - 9 GHz, results in automatic mesh generation at the highest frequency. The mesh consists of triangles, which discretize all metal surfaces of the antenna and tetrahedra, which discretize the volume of the dielectric substrates. Plot the mesh for the metal surfaces and dielectric surfaces.

```
figure  
mesh(p,view='metal')
```

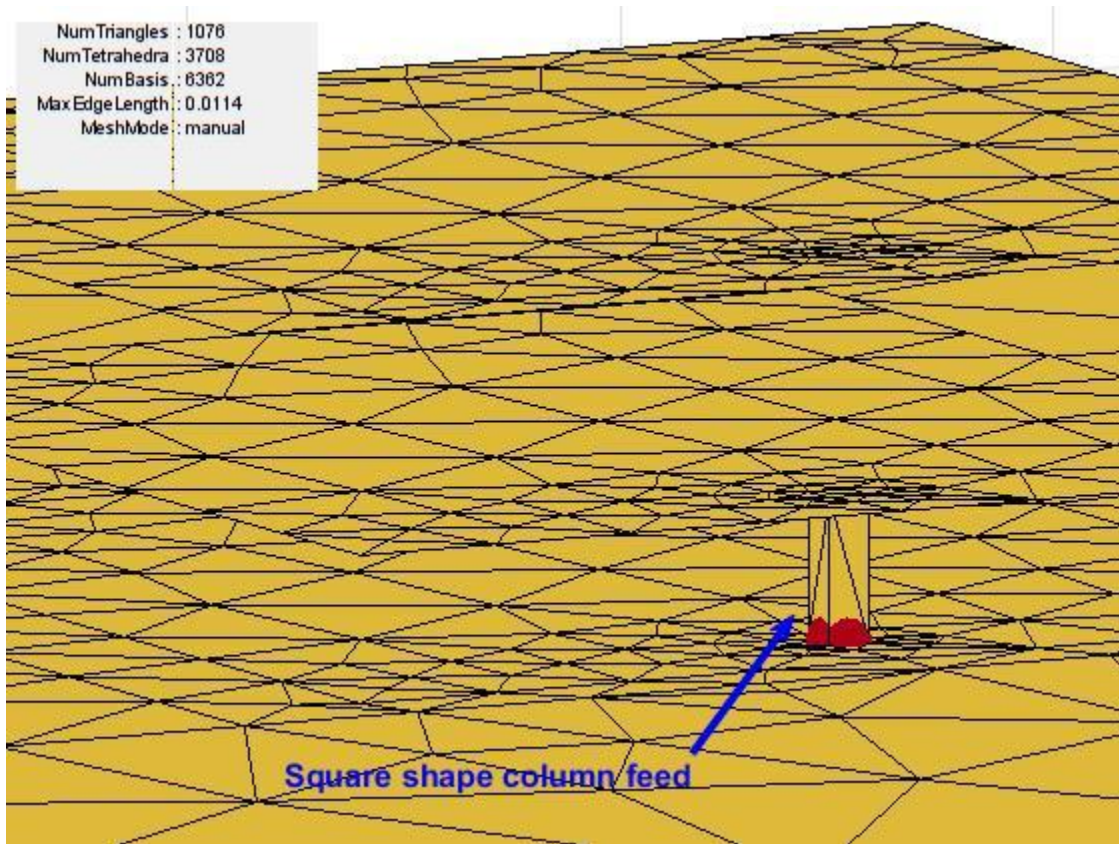
NumTriangles: 870
NumTetrahedra: 3444
NumBasis: 5537
MaxEdgeLength: 0.01
MeshMode: manual



```
figure  
mesh(p,view='dielectric surface')
```



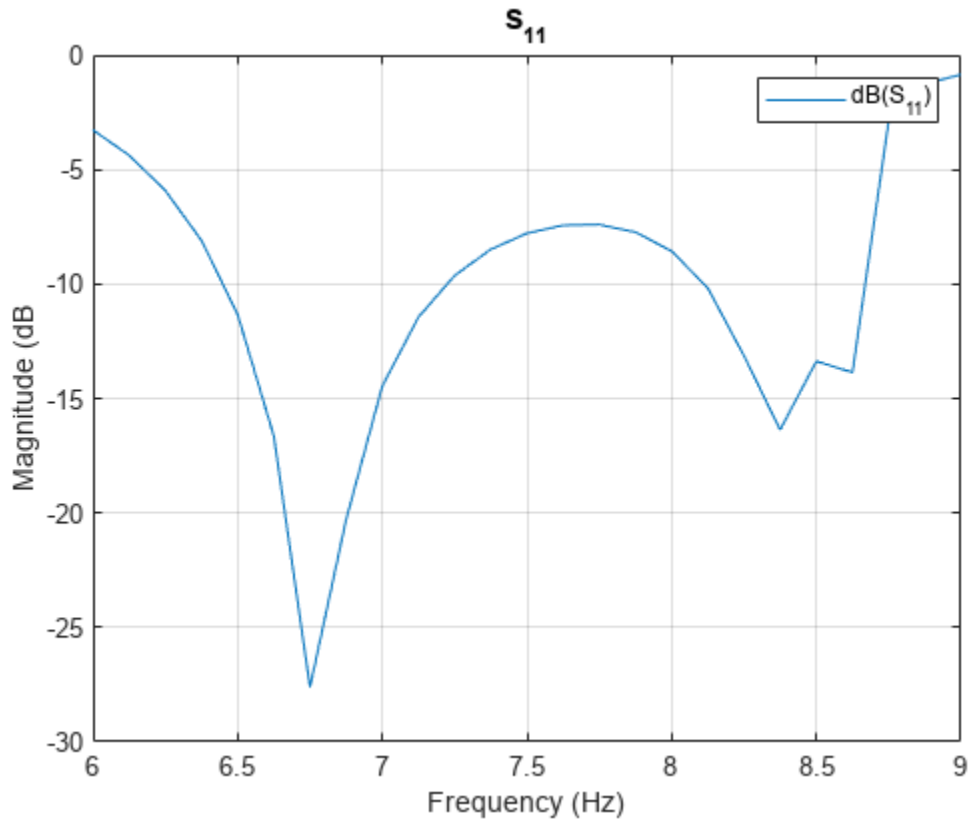
Note that a column feed model with square shaped side walls approximating a cylindrical feed is used in this patch antenna. Use the metal mesh to get a closer view of this feed structure.



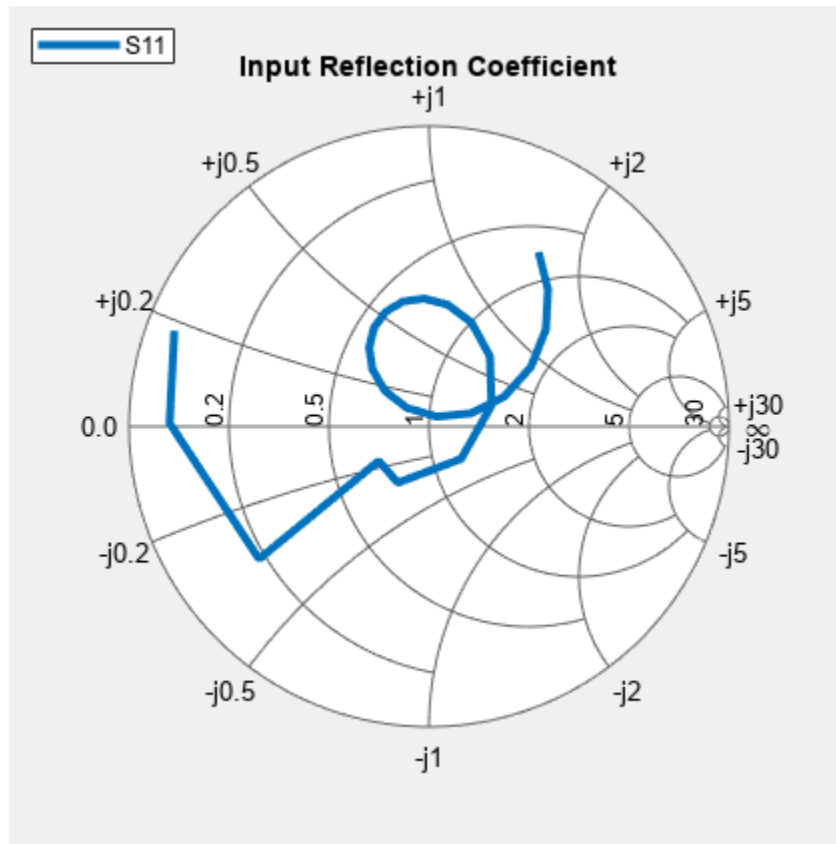
Reflection Coefficient

Since the antenna is excited by a coaxial probe, calculate the reflection coefficient at the input relative to 50-ohm reference impedance.

```
Zref = 50*ohm;
s = sparameters(p, freq, Zref);
figure
rfplot(s,1,1)
title('S11')
xlabel('Frequency (Hz)')
ylabel('Magnitude (dB)')
```



```
figure
smplot = smithplot(s);
smplot.TitleTop = 'Input Reflection Coefficient';
smplot.LineWidth = 3;
```

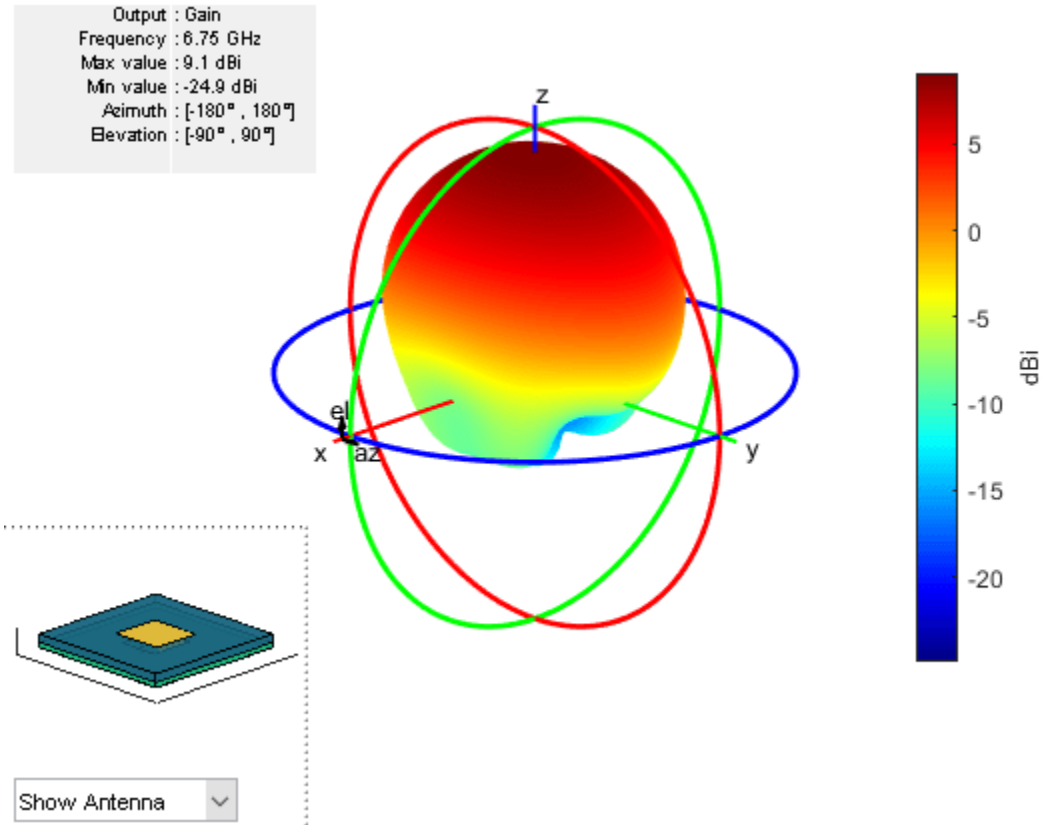


The results for the reflection coefficient match very well with the experiment results reported in [1]. The presence of the double resonance in the impedance behavior, has implications on the radiation pattern behavior of the antenna.

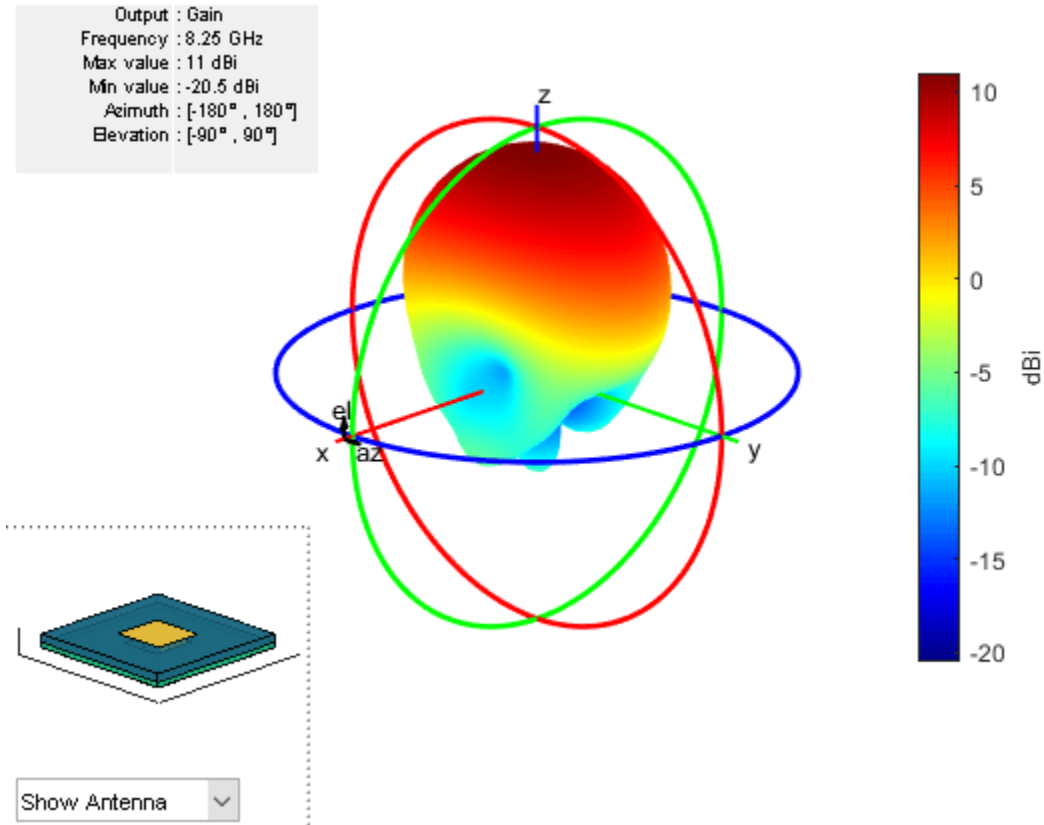
Pattern Variations Across Band

The wide impedance bandwidth observed from the port analysis of the stacked patch will have an impact on the far-field radiation pattern. To understand this, plot the radiation pattern in the far-field of this antenna at the 2 notches in the reflection coefficient plot - 6.75 GHz and 8.25 GHz.

```
patternfreqs = [6.75*GHz, 8.25*GHz];
freqIndx = arrayfun(@(x) find(freq==x),patternfreqs);
figure
pattern(p,freq(freqIndx(1)))
```



```
figure
pattern(p, freq(freqIndx(2)))
```



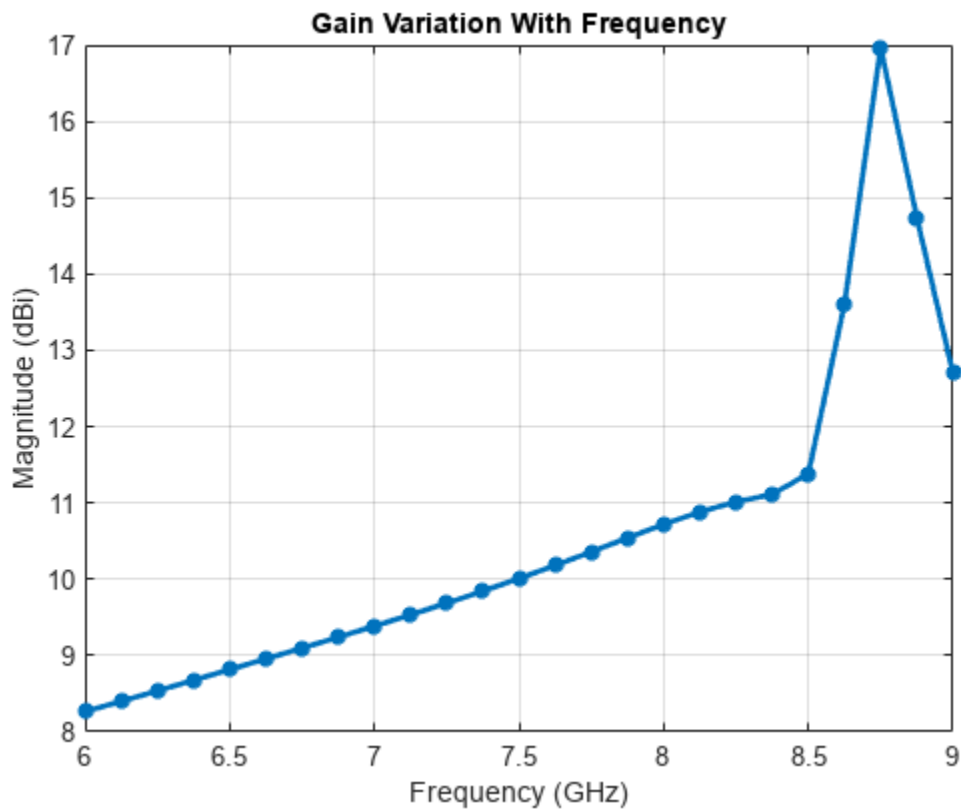
Realized Gain Variation Across Band

The pattern is relatively stable at higher elevation angles close to zenith. However, note that the radiation towards the horizon and backlobe seems to grow at the higher frequency end of the 6-9 GHz band. These results account for the losses in the dielectric but not for the impedance mismatches that might exist at the feed point. To understand the effect of the impedance mismatch, calculate the realized gain at zenith and compare it with the gain.

```
D = zeros(1,numel(freq));
az = 0;
el = 90;
for i = 1:numel(freq)
    D(i) = pattern(p,freq(i),az,el);
end
```

Plot Gain

```
h = figure;
plot(freq./GHz,D,'-*',LineWidth=2)
xlabel('Frequency (GHz)')
ylabel('Magnititude (dBi)')
grid on
title('Gain Variation With Frequency')
```

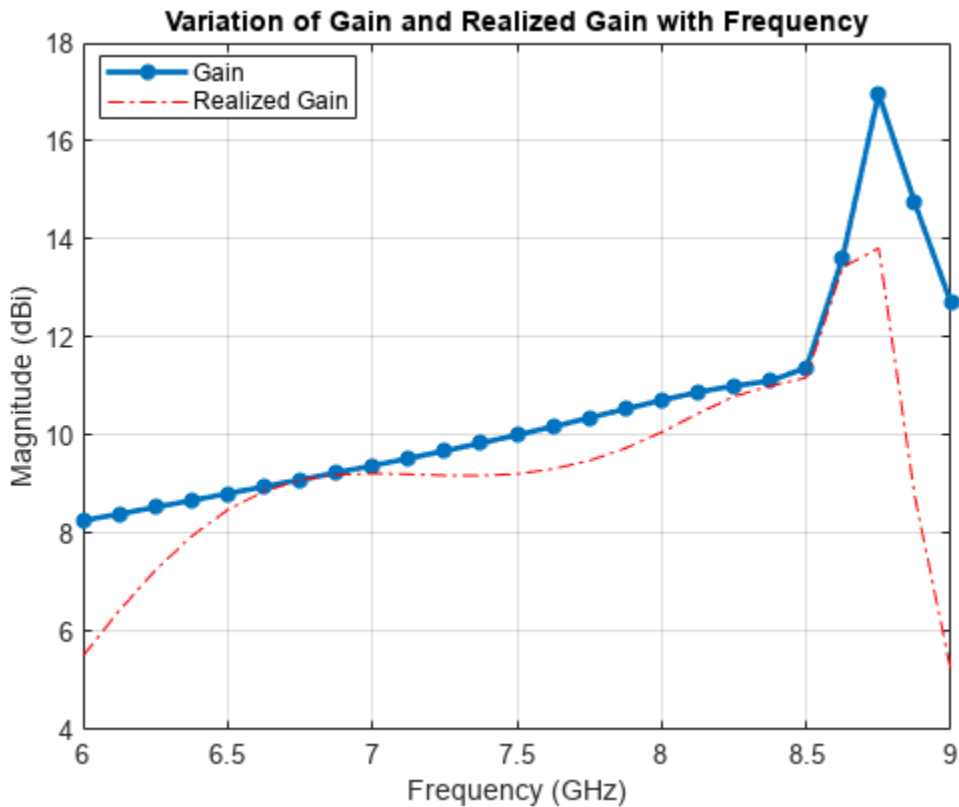


Calculate mismatch factor

```
gamma = rfparam(s,1,1);
mismatchFactor = 10*log10(1 - abs(gamma).^2);
```

Compute Realized Gain

```
Gr = mismatchFactor.' + D;
figure(h)
hold on
plot(freq./GHz,Gr,'r-.')
legend('Gain','Realized Gain',Location='best')
title('Variation of Gain and Realized Gain with Frequency')
hold off
```



Summary

The experimental results of the stacked patch design reported in [1] agree well with the analysis results shown in this example. In addition the antenna exhibits good stability in gain variation close to zenith with higher variations in shape near the horizon and the backlobe regions. The maximum realized gain at zenith is achieved at the lower and upper frequency ends of the 6 - 9 GHz band, especially at the notches in the input reflection coefficient where the match is the best. Within the 7 - 9 GHz range observe that the realized gain only drops off by about 0.6 dB. The reduced values of the realized gain below 6.5 GHz and above 8.5 GHz are due to the impedance mismatch.

References

- [1] R. B. Waterhouse, "Design of probe-fed stacked patches," in *IEEE Transactions on Antennas and Propagation*, vol. 47, no. 12, pp. 1780-1784, Dec 1999.
- [2] D.Orban and G.J.K.Moernaut, *The Basics of Patch Antennas*, Updated, Orban Microwave Products.
- [3] C. A. Balanis, 'Antenna Theory. Analysis and Design,' p.514, Wiley, New York, 3rd Edition, 2005

See Also

"FMCW Patch Antenna Array" on page 5-172

Design Internally Matched Ultra-Wideband Vivaldi Antenna

This example will model and analyze a vivaldi antenna with an internal matching circuit. The vivaldi is also known as an exponentially tapered slot antenna. The antenna possesses wideband characteristics, low cross polarization and a highly directive pattern. The design will be implemented on a single layer dielectric substrate with 2 layers of metal; one for a flared slot line, and the feed line with the matching circuit on the other layer. The substrate is chosen as a low cost FR4 material of thickness 0.8 mm. The design is intended for operation over the frequency band 3.1 - 10.6 GHz [1].

Antenna Dimensions

The vivaldi antenna is designed to operate between 3 to 11 GHz with dimensions of 45×40 mm. At the highest frequency of operation, the structure is approximately $1\lambda \times 1\lambda$. Define the design parameters of the antenna as provided.

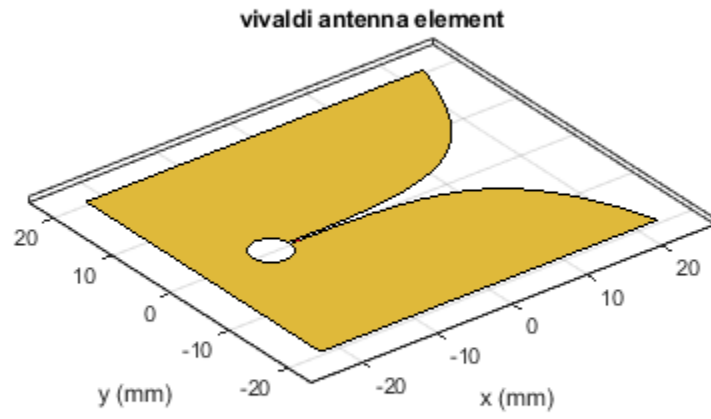
```
Lgnd = 45e-3;
Wgnd = 40e-3;
Ls = 5e-3;
Ltaper = 28.5e-3;
Wtaper = 39.96e-3;
s = 0.4e-3;
d = 5e-3;
Ka = (1/Ltaper)*(log(Wtaper/s)/log(exp(1)));
```

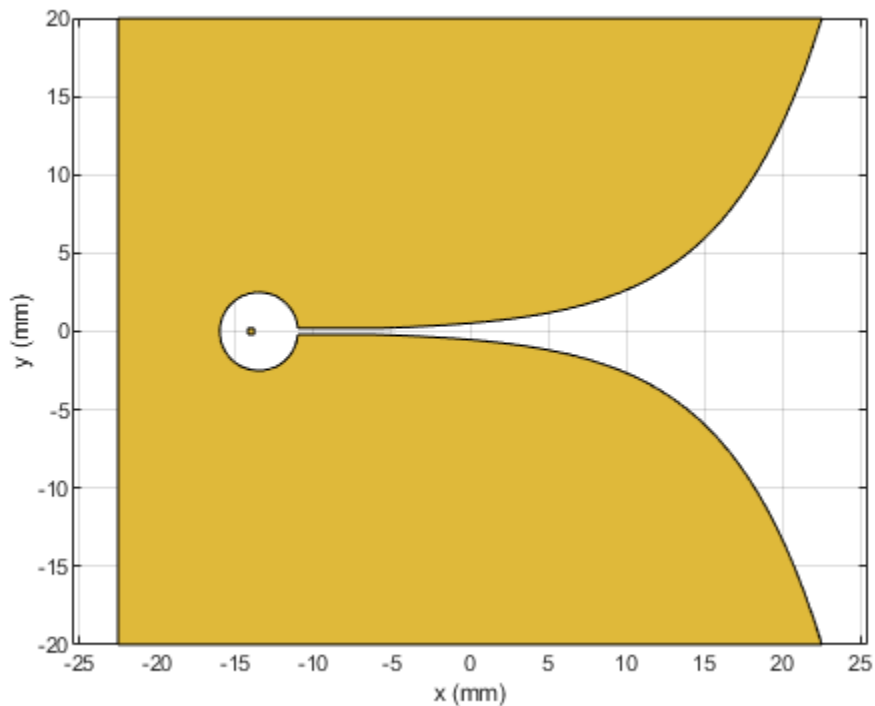
Create Top Layer Shape

This design consists of three layers; the top layer is the exponentially tapered slot shape. This is the same shape as the vivaldi in the Antenna Toolbox catalog. The bottom layer consists of the feed and the matching circuit. The middle layer is the FR4 substrate. The function `pcbStack(ant)` converts any 2D or 2.5D antenna from the catalog into a pcb antenna for further modeling and analysis. Create the vivaldi antenna from the catalog and visualize it. Thereafter, move the feed and convert it to the stack representation and access the layer geometry for further modifications.

```
vivaldiant = vivaldi('TaperLength',Ltaper, 'ApertureWidth', Wtaper, ...
                    'OpeningRate', Ka,'SlotLineWidth', s, ...
                    'CavityDiameter',d,'CavityToTaperSpacing',Ls, ...
                    'GroundPlaneLength', Lgnd, 'GroundPlaneWidth', Wgnd,...
                    'FeedOffset',-10e-3);

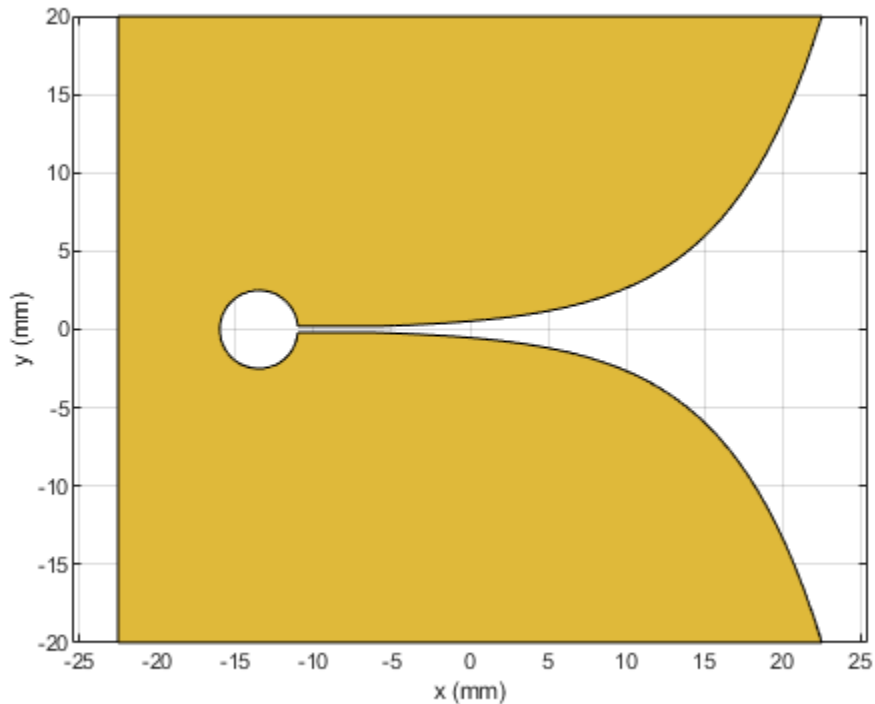
figure
show(vivaldiant);
vivaldiant.FeedOffset = -14e-3;
ewant = pcbStack(vivaldiant);
topLayer = ewant.Layers{1};
figure
show(topLayer)
```



Remove Feed Strip from Vivaldi Structure The default vivaldi antenna structure in the catalog has an internal feed and the associated feed strip specified at the center of the antenna. In this example we are using edge feed model. Remove the strip from the vivaldi structure.

```
cutout = antenna.Rectangle('Length',1e-3,'Width',4e-3,'Center',[-0.014,0]);  
topLayer = topLayer-cutout;  
figure;  
show(topLayer);
```



Create a Matching Circuit for the Vivaldi Antenna

A stepped microstrip line is used as a matching circuit with a 90 degree bend terminating into a radial bowtie stub. Use the rectangle shape primitive in Antenna Toolbox™ to create the stepped microstrip line. Boolean addition operation is used among the shape primitives for this purpose.

```
L1 = 8e-3;
L2 = 4.1e-3;
L3 = 9.1e-3;
W1 = 1.5e-3;
W2 = 1e-3;
W3 = 0.75e-3;
H = 0.8e-3;
fp = 11.2e-3;
th = 90;
```

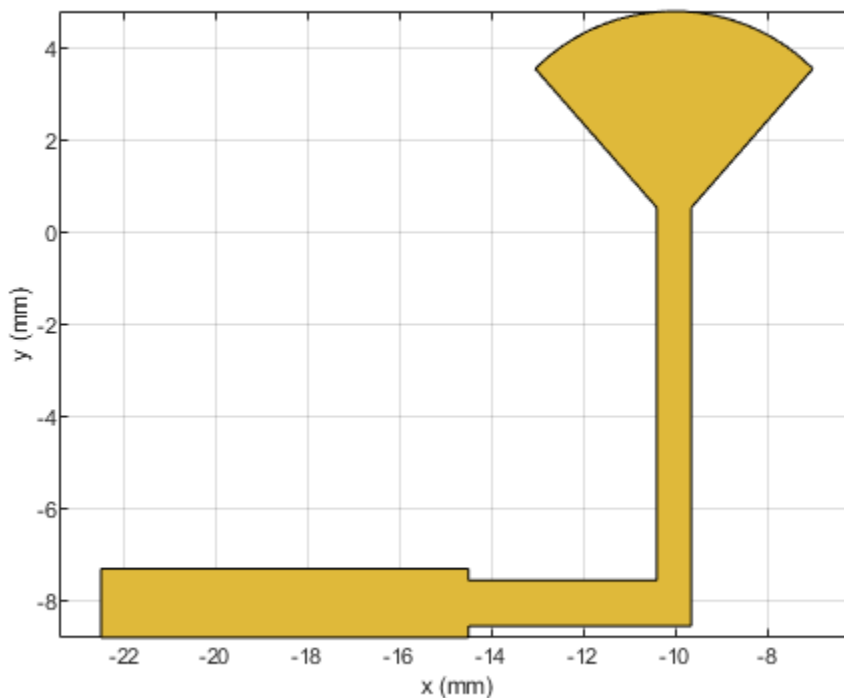
```
patch1 = antenna.Rectangle('Length',L1,'Width',W1,...
    'Center',[-(Lgnd/2 - L1/2), -(Wgnd/2 - fp - W1/2)],...
    'NumPoints', [10,2,10,2]);
patch2 = antenna.Rectangle('Length',L2,'Width',W2,...
    'Center',[-(Lgnd/2 - L1 - L2/2), -(Wgnd/2 - fp - W1/2)],...
    'NumPoints', [5,2,5,2]);
patch3 = antenna.Rectangle('Length',W3,'Width',L3,...
    'Center',[-(Lgnd/2 - L1 - L2 - W3/2), -(Wgnd/2 - fp - W1/2 + W2/2 - L3),
    'NumPoints', [2,10,2,10]);
```

Create a Radial Stub Matching circuit To create a radial stub matching circuit we use the function `makebowtie`. It provides inputs for the radius, neck width, flare angle, center, shape of the bowtie and finally number of points to create a shape for bowtie.

```
Bowtie = em.internal.makebowtie(8.55e-3, W3, th, [0 0 0], 'rounded', 20);
rotatedBowtie = em.internal.rotateshape(Bowtie, [0 0 1], [0 0 0], 90);
p = antenna.Polygon('Vertices', rotatedBowtie);

radialStub = translate(p, [-(Lgnd/2 - L1 - L2 - W3/2) -(Wgnd/2 - fp - W1/2 + W2/2 - L3) 0]);

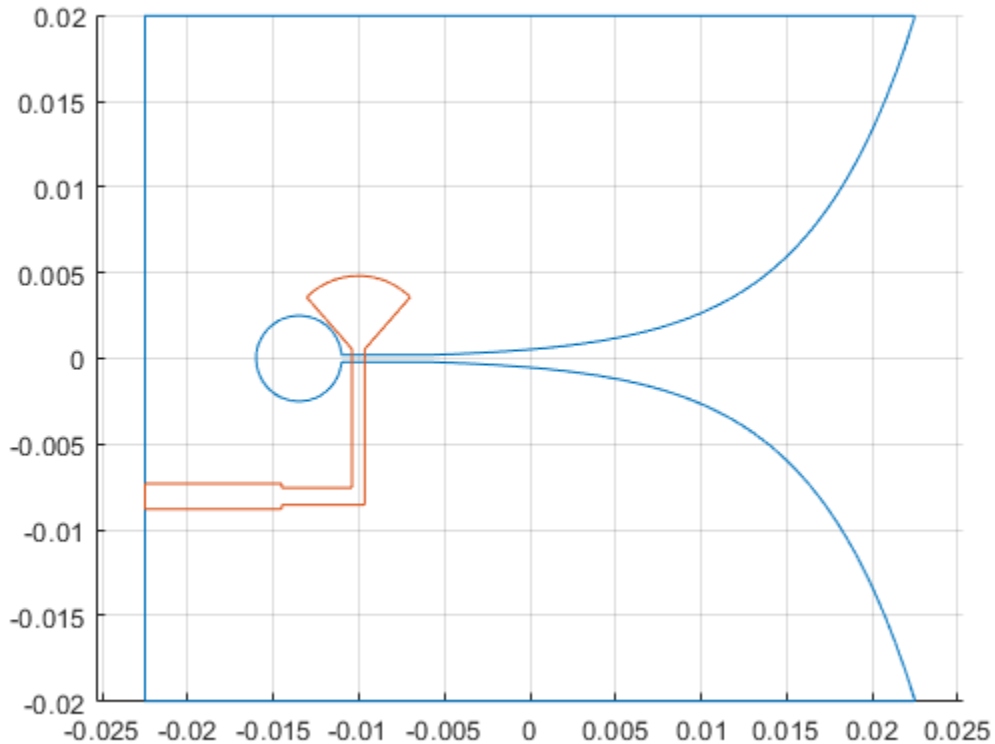
bottomLayer = patch1+patch2+patch3+radialStub;
figure;
show(bottomLayer);
```



Create the PCB stack

Create the board shape for the antenna. The board in this case is rectangular and 45 mm x 40 mm in size.

```
boardShape = antenna.Rectangle('Length', Lgnd, 'Width', Wgnd);
figure;
hold on;
plot(topLayer)
plot(bottomLayer)
grid on
```

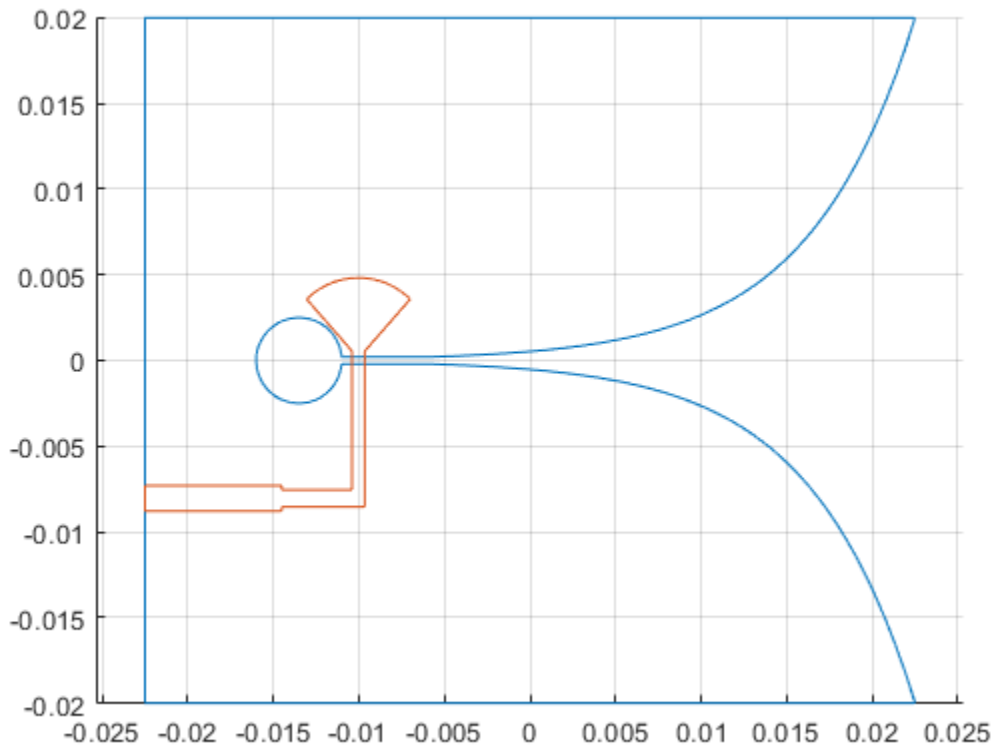


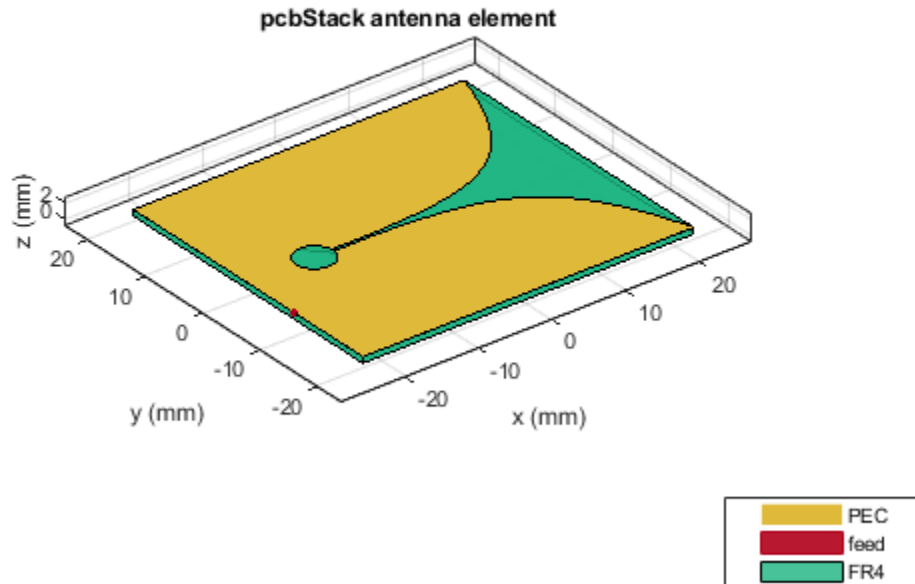
Define the Dielectric Substrate The vivaldi antenna is constructed using an FR4 substrate with relative permittivity of 4.4 and height 0.8 mm.

```
substrate = dielectric('Name', 'FR4', 'EpsilonR', 4.4, 'Thickness', H);
```

Assign the Layers and Define the Feed Assign the layers starting from the top-layer, in this case vivaldi structure, followed by the FR4 dielectric substrate and finally the lowest layer, which is the matching circuit. The edge-feed is specified between the vivaldi and the matching circuit on the bottom layer. Having the feed line with the matching circuit on the bottom layer reduces any spurious radiation. Define the feed location and the feed diameter as well.

```
vivaldi_Notch = pcbStack;
vivaldi_Notch.Name = 'vivaldiNotch';
vivaldi_Notch.BoardThickness = H;
vivaldi_Notch.BoardShape = boardShape;
vivaldi_Notch.Layers = {topLayer, substrate, bottomLayer};
vivaldi_Notch.FeedLocations = [-(Lgnd/2), -(Wgnd/2 - fp - W1/2), 1, 3];
vivaldi_Notch.FeedDiameter = W1/2;
figure;
show(vivaldi_Notch);
```





Impedance Analysis

Calculate the antenna impedance over the range of 2.5 GHz to 11 GHz. For the purpose of executing this example, the impedance analysis has been precomputed and save in a MAT-file. The analysis was performed with the mesh generation in the auto mode. Execute the info method on the antenna to get information about the meshing/solution status, analysis frequencies and estimate of the memory required for analysis.

```
freq = linspace(2.5e9, 11e9,41);
bandfreqs = [3.1e9, 10.6e9];
freqIndx = nan.*(ones(1,numel(bandfreqs)));
for i = 1:numel(bandfreqs)
    df = abs(freq-bandfreqs(i));
    freqIndx(i) = find(df==min(df));
end
load vivaldi_Notch_auto_mesh
vivaldiInfo = info(vivaldi_Notch)
figure;
impedance(vivaldi_Notch, freq);
```

```
vivaldiInfo =
```

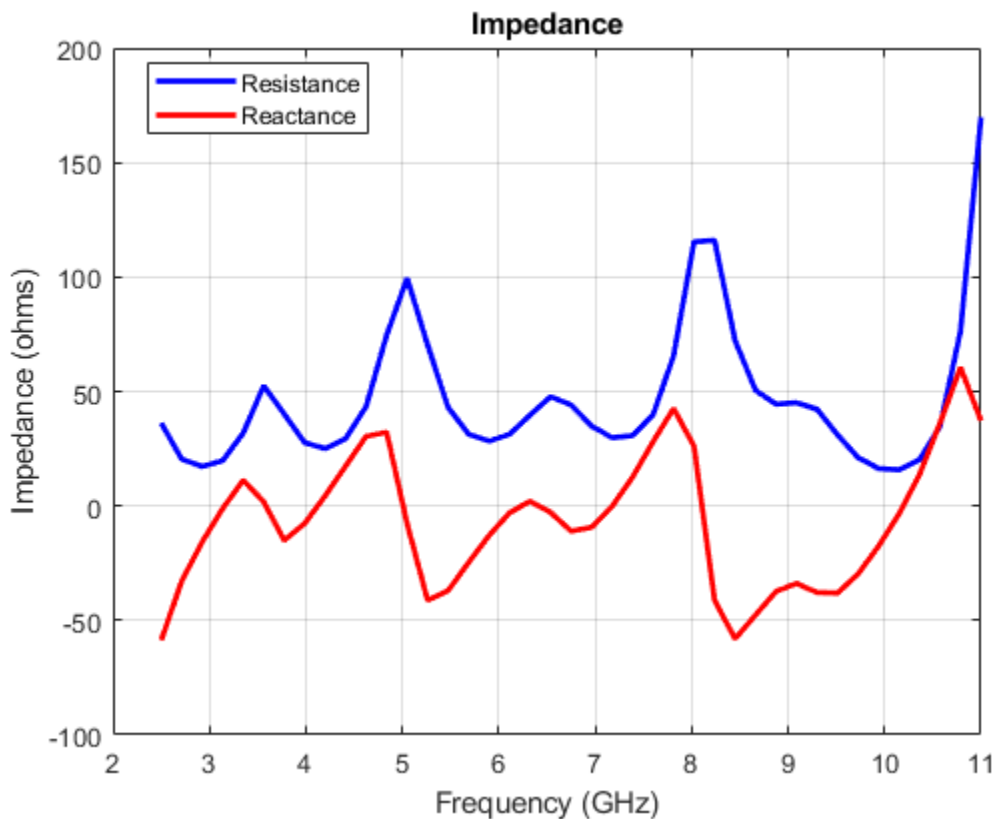
```
struct with fields:
```

```
IsSolved: "true"
IsMeshed: "true"
```

```

MeshingMode: "auto"
HasSubstrate: "true"
HasLoad: "false"
PortFrequency: [1×41 double]
FieldFrequency: []
MemoryEstimate: "2.2 GB"

```



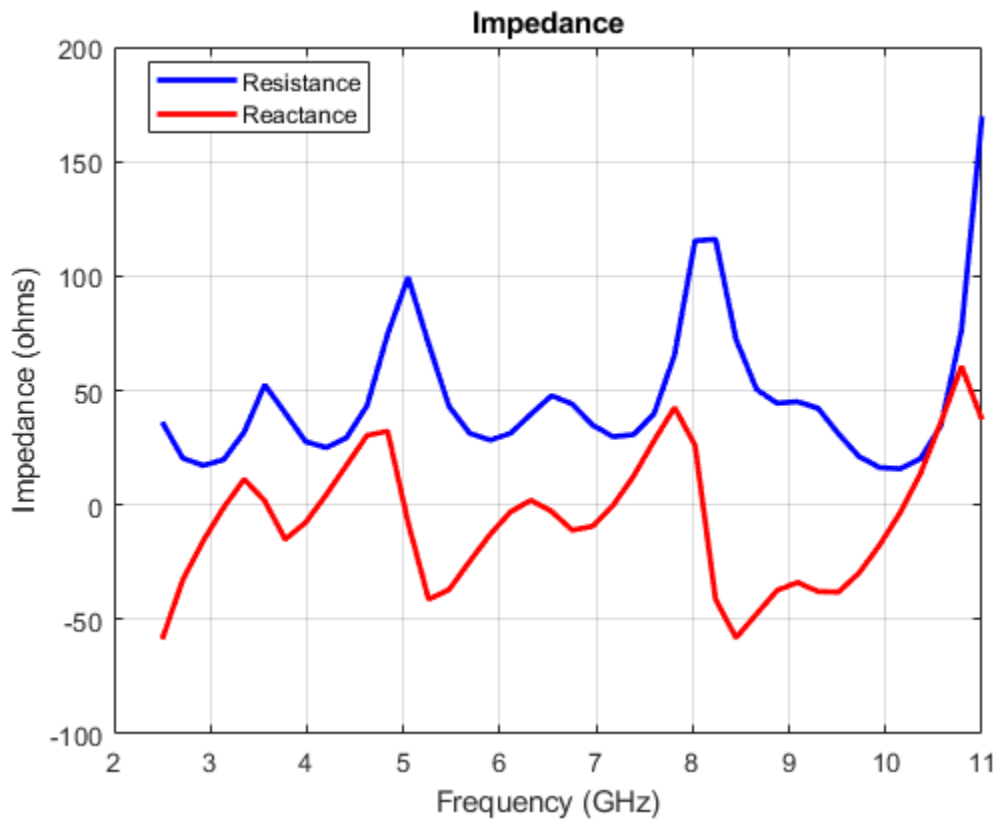
Refine the Antenna Mesh

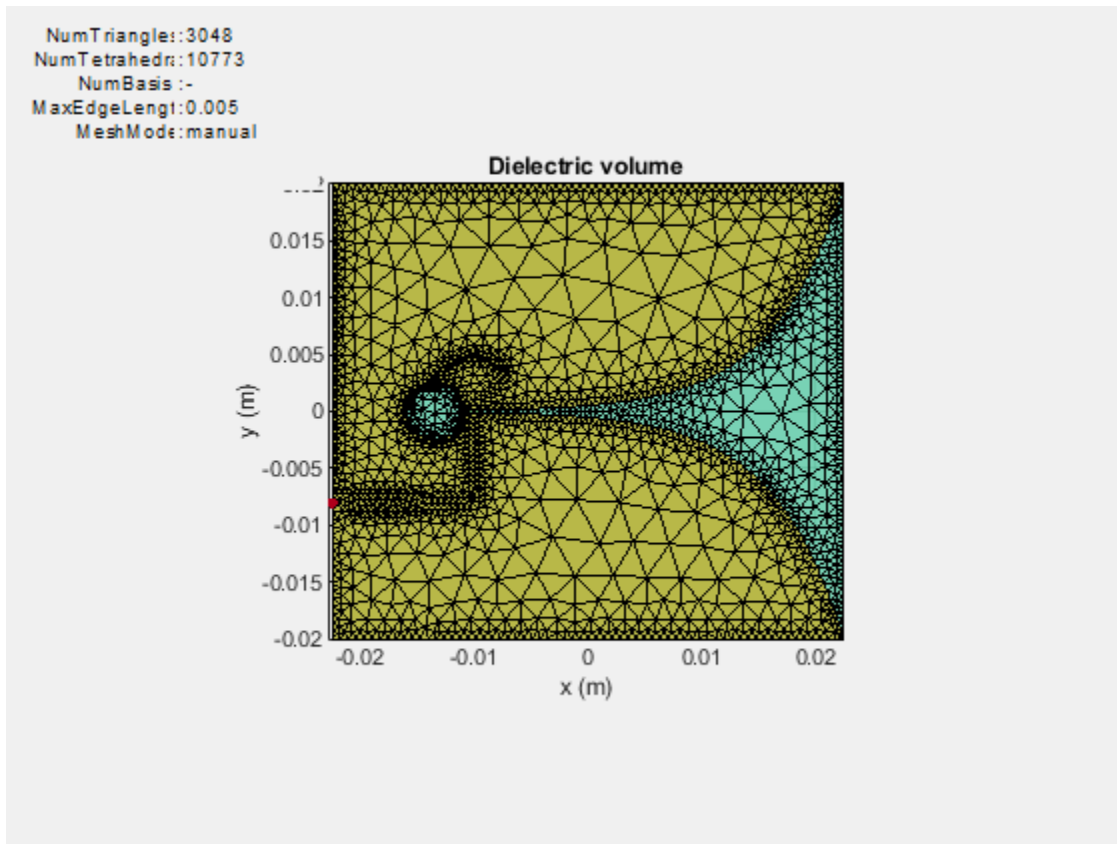
Refine the mesh to check for convergence with the impedance variation over the band. The automatically generated mesh has a maximum edge length of approximately 2 cm and a minimum edge length of 3 mm. The highest frequency in the analysis range is 11 GHz which corresponds to a wavelength in free space of 27.3 mm. Considering 10 elements per wavelength would give us an edge length of approximately 2.7 mm which is lower than the both the maximum and the minimum edge length chosen by the automatic mesher. After a few attempts, using a maximum edgelenhth of 5 mm and a minimum edge length of 0.8 mm resulted in a good solution.

```

figure
mesh(vivaldi_Notch, 'MaxEdgeLength',5e-3, 'MinEdgeLength',0.8e-3);
view(0,90)

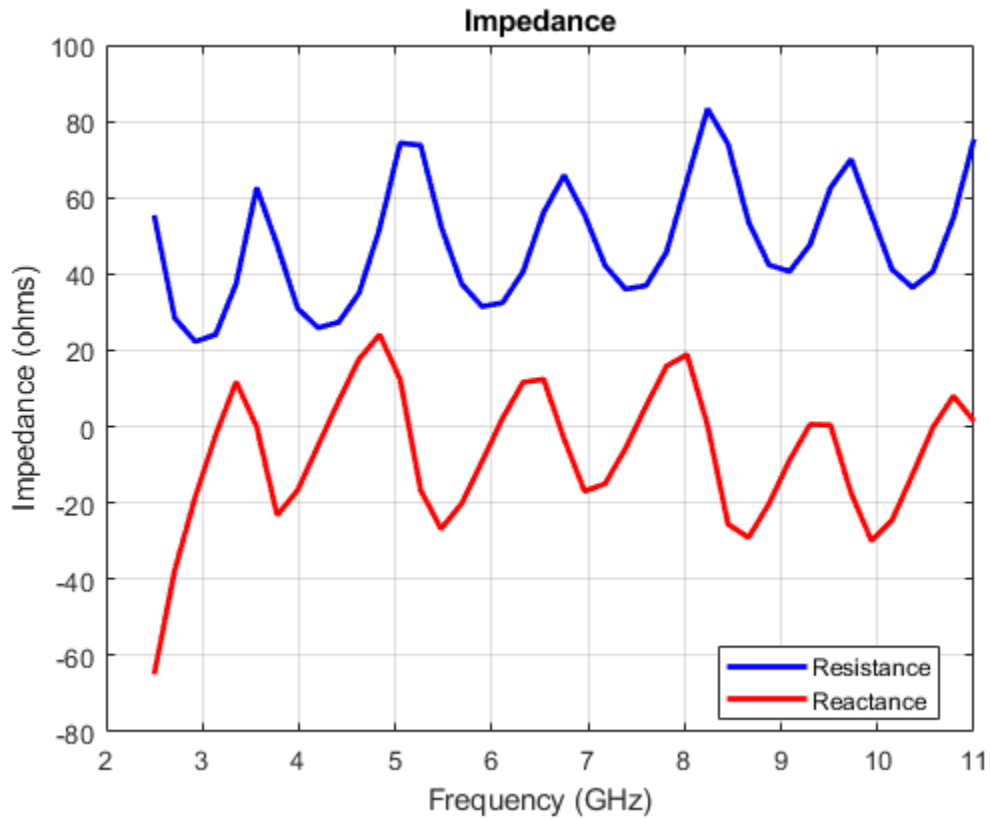
```



Due to the size of the mesh, the number of unknowns increases to obtain an accurate solution. As before the solved structure has been saved to a MAT-file and is loaded here for further analysis.

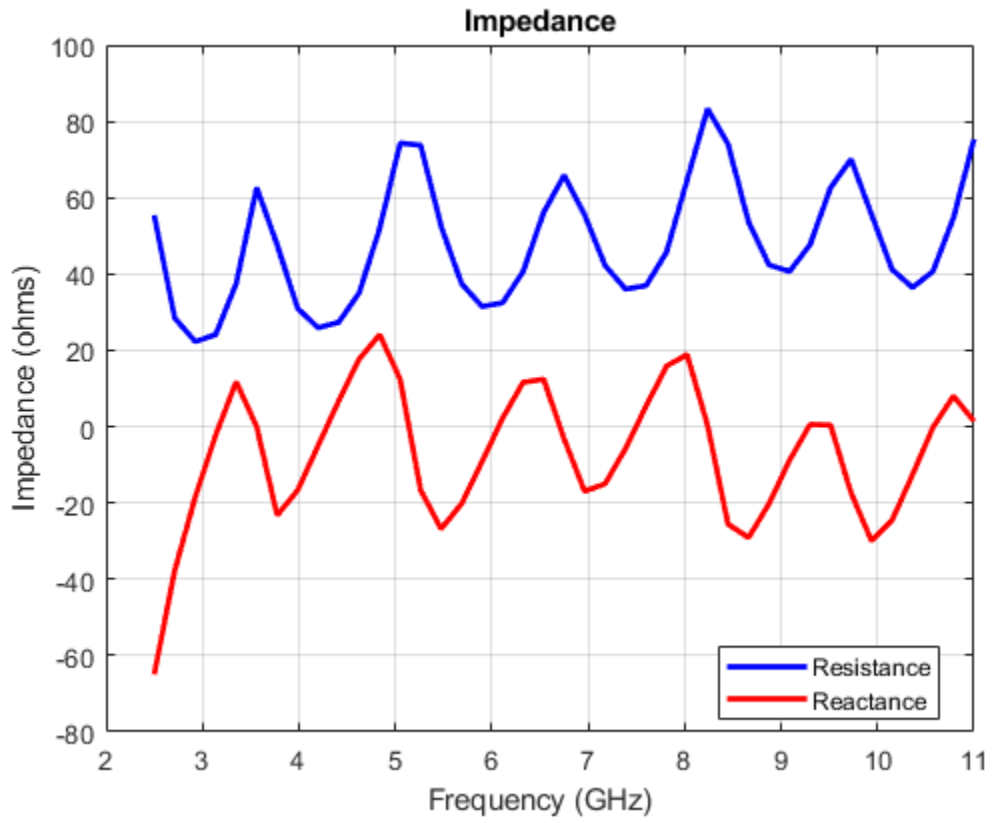
```
load vivaldi_Notch_manual_mesh.mat  
figure;  
impedance(vivaldi_Notch, freq);
```

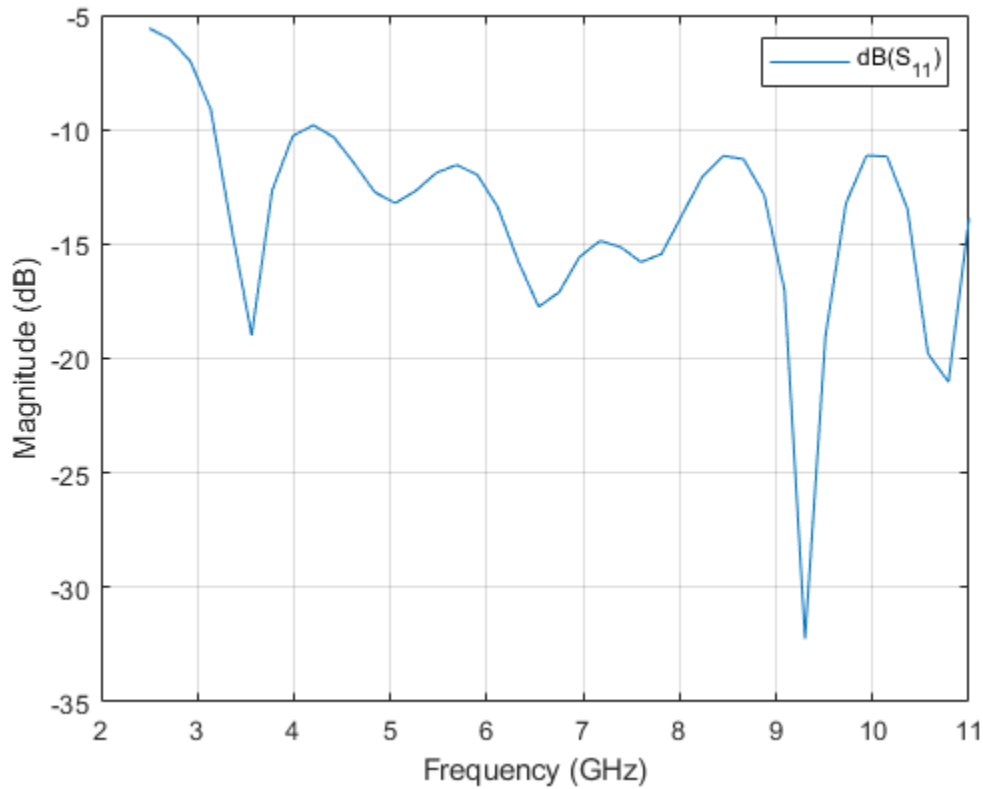


Reflection Coefficient

Calculate the reflection coefficient at the input relative to 50-ohm reference impedance. The reflection coefficient is below -10 dB for the frequency range from 3.1 GHz to 11 GHz. Save the reflection coefficient for use later on for calculating the realized gain.

```
figure;
s = sparameters(vivaldi_Notch, freq);
rfplot(s);
gamma = rfparam(s,1,1);
```

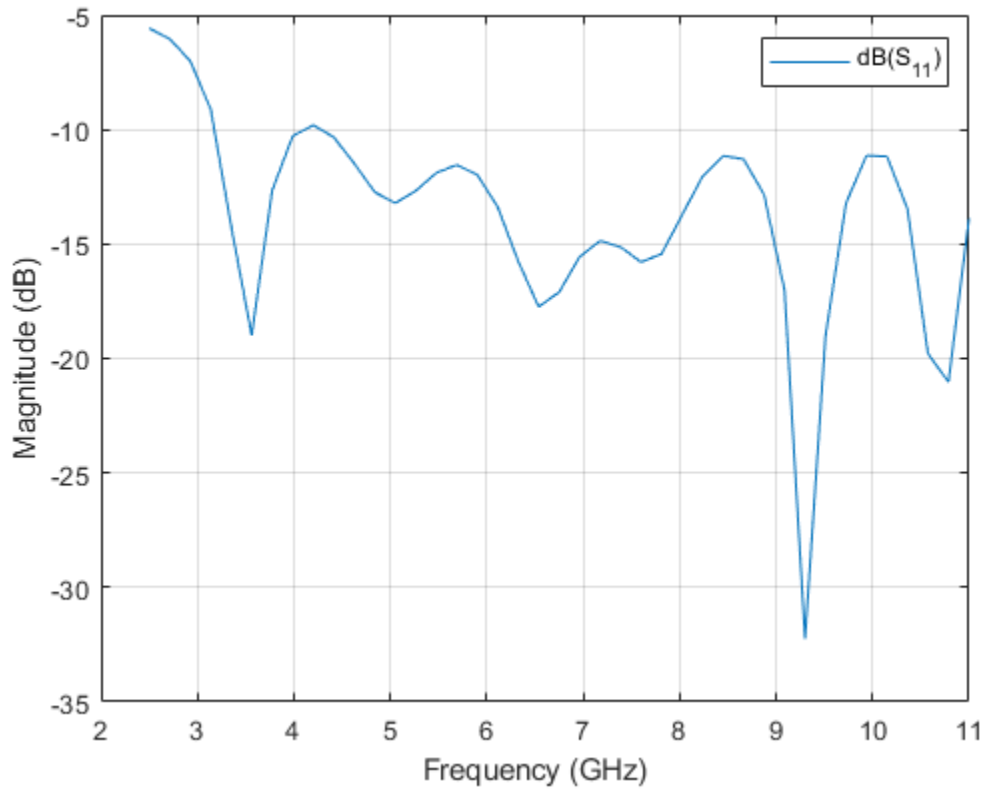


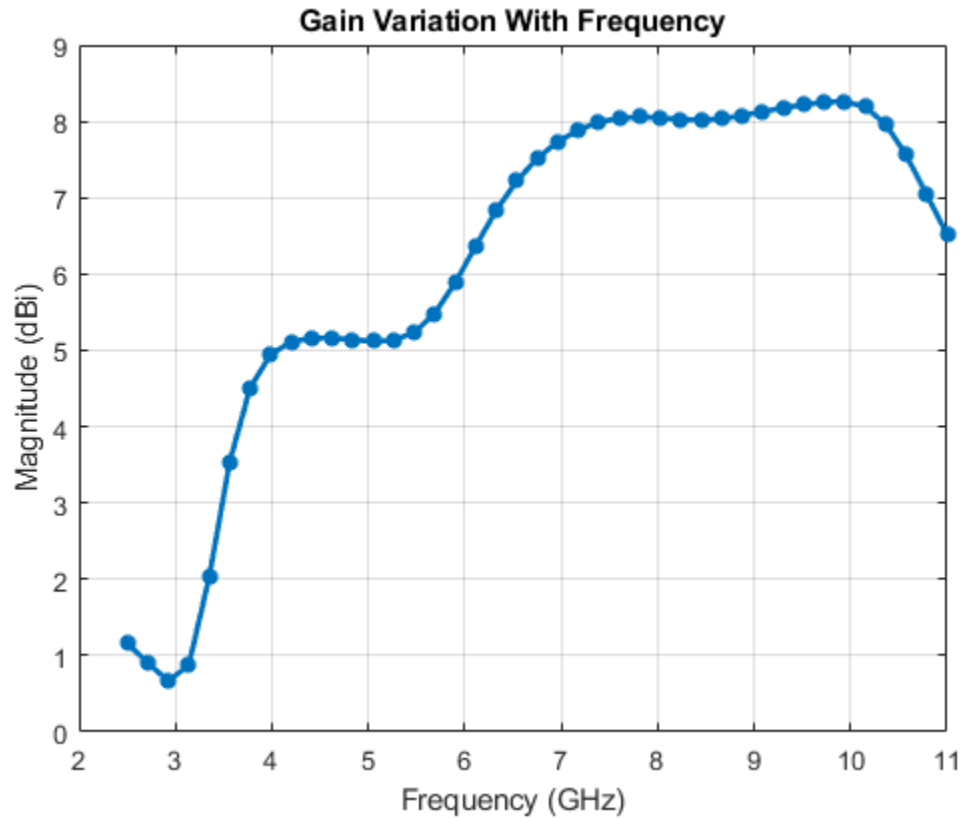


Realized Gain

The antenna realized gain includes losses in the dielectric and due to any impedance mismatch. Plot the variation in realized gain with frequency at the antenna boresight at $(az,el) = (0,0)$ deg.

```
G = zeros(1,numel(freq));
az = 0;
el = 0;
for i = 1:numel(freq)
    G(i) = pattern(vivaldi_Notch,freq(i),az,el);
end
g = figure;
plot(freq./1e9,G,'-*','LineWidth',2);
xlabel('Frequency (GHz)');
ylabel('Magnitude (dBi)');
grid on;
title('Gain Variation With Frequency');
```



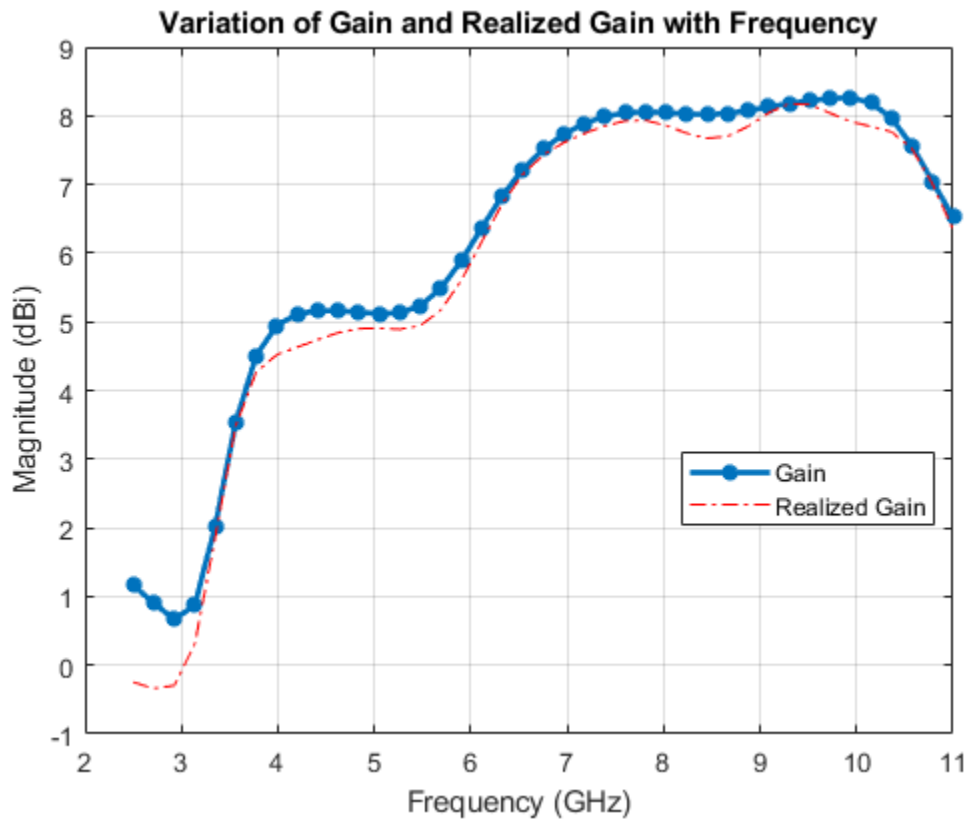


Compute Mismatch and calculate Realized Gain

```

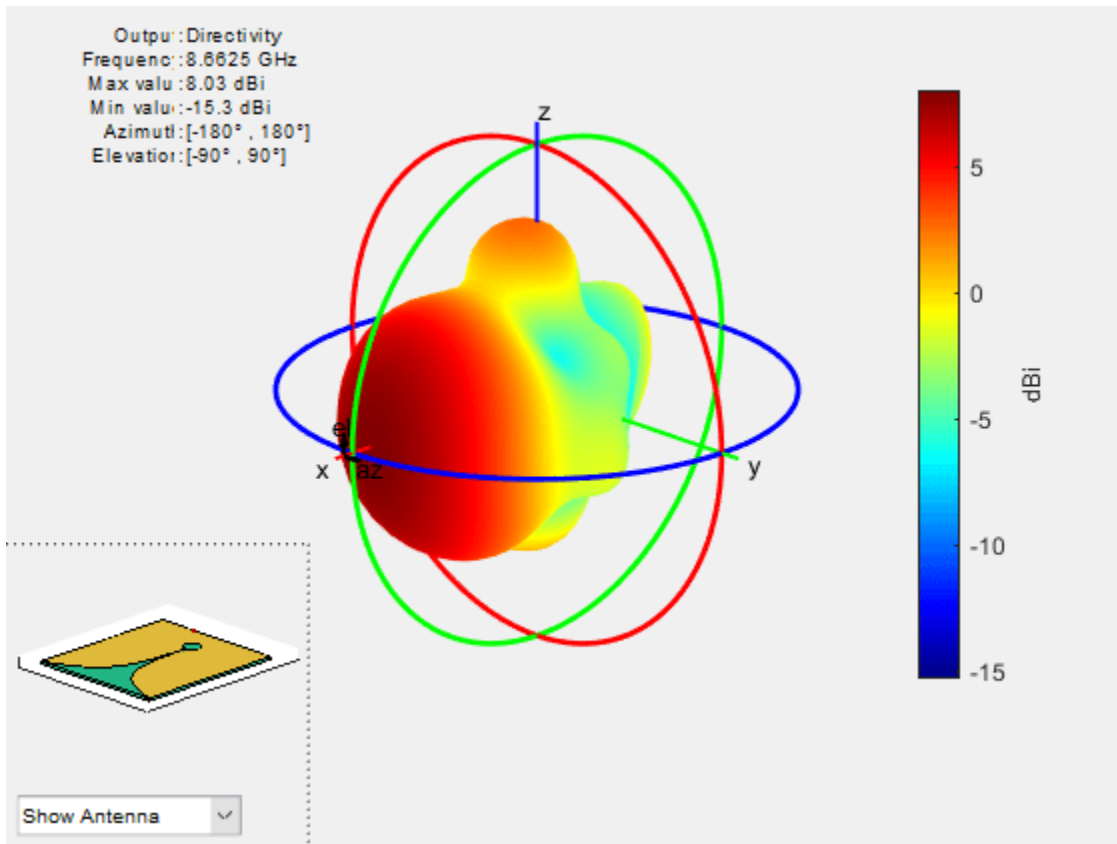
mismatchFactor = 10*log10(1 - abs(gamma).^2);
Gr = mismatchFactor.' + G;
figure(g);
hold on
plot(freq./1e9,Gr,'r-.');
legend('Gain','Realized Gain','Location','best')
title('Variation of Gain and Realized Gain with Frequency')
hold off

```



The wide impedance bandwidth does not necessarily translate to a wide gain/pattern bandwidth. The highest gain is achieved over the 7 - 10.4 GHz range at boresight of approximately 9.5 dBi. Plot the 3D pattern in the middle of this sub-band to understand the overall radiation characteristics.

```
dfsub = abs(freq - (10.4e9+7e9)/2);  
subfreqIndx = find(dfsub==min(dfsub));  
figure;  
pattern(vivaldi_Notch, freq(subfreqIndx));
```

Phase Center Variation of the Antenna

The phase center of an antenna is the local center of curvature of the far-field phase front [2]. It can vary with frequency and observation angle. An analysis of the phase center variation is critical for positioning systems. This is because variations in the phase center directly translate to variations in time delay, which can impact range estimates between a transmitter and a receiver. To understand this, calculate the maximum possible variation in time delay due to a harmonic signal at f_{\min} and another at f_{\max} over a set of observation angles in the far-field. Choose the angles over 2 orthogonal planes; the first specified at elevation = 0 degrees, i.e. the xy-plane and the other at az = 0 degrees, i.e. xz plane. In the xy-plane we will use the E_{ϕ} component of the electric field for analysis while in the xz-plane we will use the E_{θ} component of the electric field.

Create Points in Far-Field and Calculate Electric Field Define the far-field sphere radius and the set of observation angles in azimuth and elevation. Choose the two harmonic signal frequencies to be at 3 and 11 GHz respectively.

```
az = -180:5:180;
el = -90:5:90;
fmin = freq(freqIndx(1));
fmax = freq(freqIndx(2));
R = 100*299792458/fmin;
coord = 'sph';
phi = 0;
theta = 90 - el;
[Points, ~, ~] = em.internal.calcpointsinspace( phi, theta, R, coord);
```

Calculate Local Phase Variation of E-field Find the electric field at the two frequencies and convert to the spherical component E_θ . Since we are interested in the maximum time delay variation, we first calculate the maximum phase variation between the two frequencies over the set of points in XZ-plane.

```
E_at_fmin = EHfields(vivaldi_Notch,fmin,Points);
E_at_fmax = EHfields(vivaldi_Notch,fmax,Points);
Eth_at_fmin = helperFieldInSphericalCoordinates(E_at_fmin,phi,theta);
Eth_at_fmax = helperFieldInSphericalCoordinates(E_at_fmax,phi,theta);
phase_at_fmin = angle(Eth_at_fmin);
phase_at_fmax = angle(Eth_at_fmax);
```

Calculate Time Delay Variation Between the Two Harmonic Signals

```
delta_phase = max(phase_at_fmin-phase_at_fmax) - min(phase_at_fmin-phase_at_fmax);
delta_omega = 2*pi*(fmax-fmin);
delta_time = pi*delta_phase/180/delta_omega;
delta_timeXZ = delta_time*1e12;
sprintf("The time delay variation in the XZ-plane is: %2.2f %s",delta_timeXZ,'ps')
```

```
ans =
```

```
"The time delay variation in the XZ-plane is: 2.18 ps"
```

Repeat the process for the points on the XY-plane and calculate the time delay variation due to the E_θ variation.

```
phi = az;
theta = 0;
delta_omega = 2*pi*(fmax-fmin);
[Points, ~, ~] = em.internal.calcpointsinspace(phi,theta,R,coord);
E_at_fmin = EHfields(vivaldi_Notch,fmin,Points);
E_at_fmax = EHfields(vivaldi_Notch,fmax,Points);
[~,Ephi_at_fmin] = helperFieldInSphericalCoordinates(E_at_fmin,phi,theta);
[~,Ephi_at_fmax] = helperFieldInSphericalCoordinates(E_at_fmax,phi,theta);
phase_at_fmin = angle(Ephi_at_fmin);
phase_at_fmax = angle(Ephi_at_fmax);
delta_phase = max(phase_at_fmin-phase_at_fmax) - min(phase_at_fmin-phase_at_fmax);
delta_time = pi*delta_phase/180/delta_omega;
delta_timeXY = delta_time*1e12;
sprintf("The time delay variation in the XY-plane is: %2.2f %s",delta_timeXY,'ps')
```

```
ans =
```

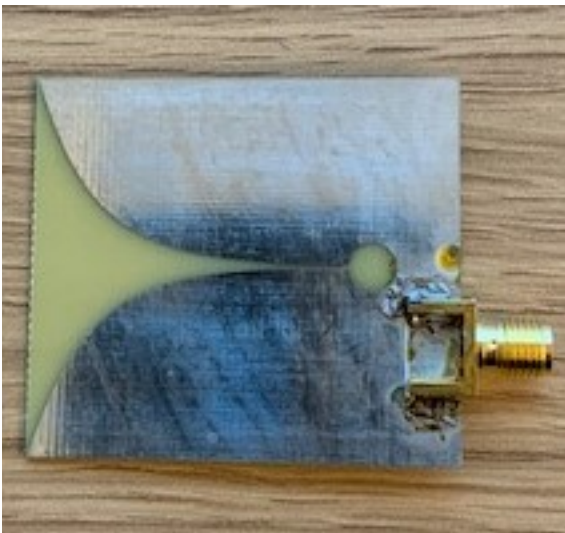
```
"The time delay variation in the XY-plane is: 2.71 ps"
```

Observation The time delay variation in the two planes that bisect the antenna boresight, reveal that the phase center is relatively stable. The average time delay variation of approximately 2 ps translates to a maximum range error of less than 1 mm.

Generate Gerber Files for Prototyping

The vivaldi antenna can be fabricated by using the gerber file generation capability in the toolbox. For this example and SMA Edge connector from Amphenol [3] was chosen and Advanced Circuits [4]

was used as the manufacturing service. In addition, on the PCBWriter object, we chose to not enable the layer of solder mask. The fabricated antenna, is shown below.



Performance in the 3-6 GHz Range

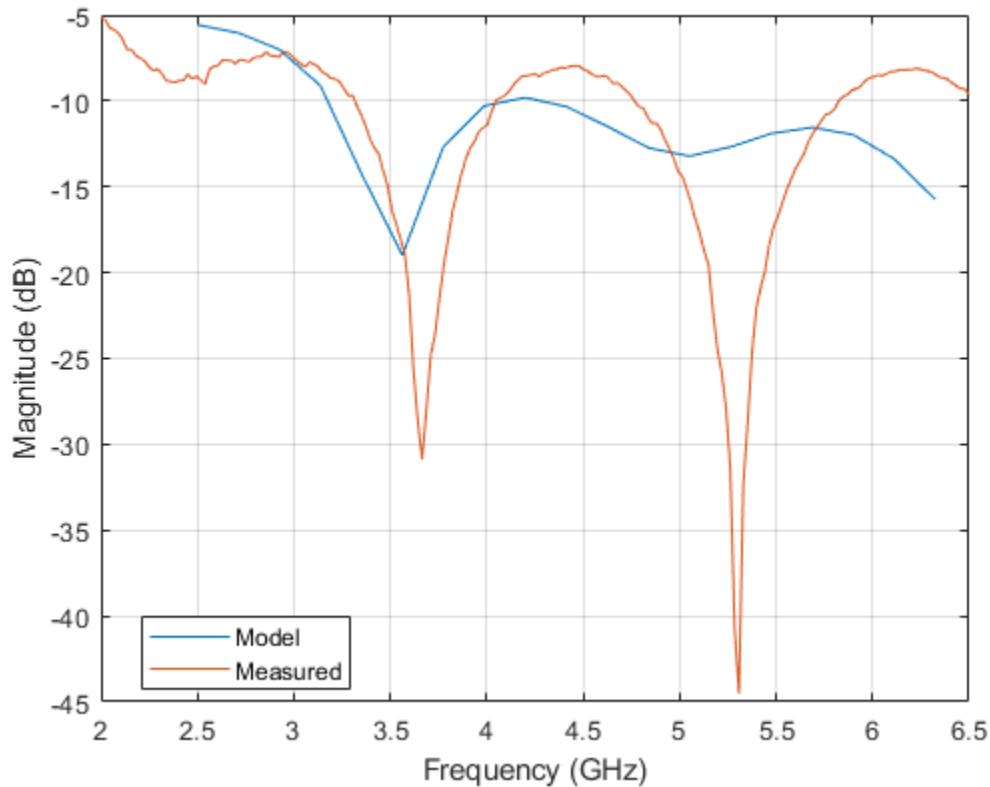
The fabricated antenna was tested using a desktop network analyzer. Since the upper limit of the analyzer was 6.5 GHz we compare the results of the antenna with the analysis from the model.

```
fLim = 6.5e9;  
findx = find(freq>fLim);  
freq2 = freq(1:findx(1)-1);  
s_model = sparameters(vivaldi_Notch, freq2);  
rfplot(s_model);
```

```

s_proto = sparameters('UWB2.s1p');
hold on
rfplot(s_proto)
legend('Model','Measured','Location','best')

```



Conclusion

The proposed antenna covers the Federal Communications Commission defined UWB spectrum and has more than 3.5:1 impedance bandwidth (from 3 GHz to more than 11 GHz). The antenna realized gain achieved over the band 3-10 GHz at boresight is very close to the gain result. Comparing the reflection coefficient in 3 - 6 GHz range of the fabricated prototype and the model reveals an acceptable performance. The reflection coefficient between 4-4.75 GHz does degrade to about -8 dB.

Reference

- [1]. High Gain Vivaldi Antenna for Radar and Microwave Imaging Applications International Journal of Signal Processing Systems Vol. 3, No. 1, June 2015 G. K. Pandey, H. S. Singh, P. K. Bharti, A. Pandey, and M. K. Meshram.
- [2]. Vishwanath Iyer, Andrew Cavanaugh, Sergey Makarov, R. J. Duckworth, 'Self-Supporting Coaxial Antenna with an Integrated Balun and a Linear Array Thereof', Proceedings of the Antenna Application Symposium, Allerton Park, Monticello, IL, pp.282-284, Sep. 21-23rd 2010.
- [3]. https://www.mouser.com/datasheet/2/18/2985-6037.PDD_0-918701.pdf

[4]. <https://www.4pcb.com/>

See Also

“Design and Analysis Using Antenna Designer”

Planning a 5G Fixed Wireless Access Link over Terrain

This example shows how to plan a fixed wireless access (FWA) link over terrain using 5G technologies. FWA is a use case for 5G to enable broadband service to homes or enterprises where wireline services are either unavailable or underperforming. FWA connects a base station to a user's fixed wireless terminal (FWT) [1 on page 5-445]. At the high frequencies required for 5G, terrain and path loss impairments like foliage and weather play an important role in determining link success.

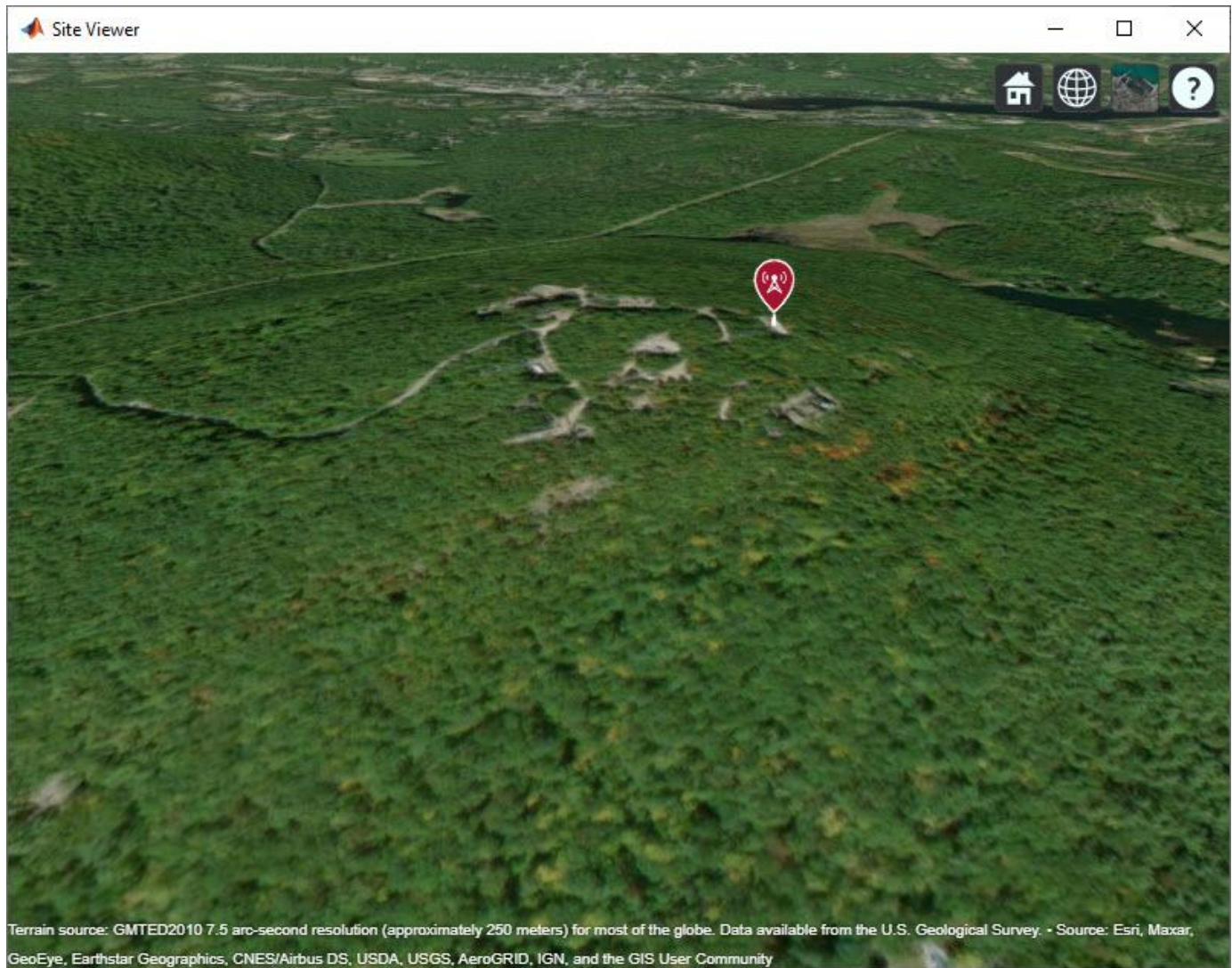
The example creates a base station and multiple receiver sites in a suburban environment, situating the antennas to achieve line-of-sight visibility over intermediate terrain. A multi-user multiple-input, multiple-output (MU-MIMO) system with high gain antennas is designed using Antenna Toolbox™ and Phased Array System Toolbox™. Signal strength at the receiver sites is assessed for two frequencies in the presence of path loss impairments.

Create Base Station Site in 28 GHz Band

Create a transmitter site on South Uncanoonuc Mountain in Goffstown, New Hampshire, US. The mountain is home to several transmitting facilities that serve the area. Define the transmitter site to represent a base station transmitting at 28 GHz with 1 Watt of power. Show the site in Site Viewer and rotate the view to visualize the site with surrounding terrain.

```
fq = 28e9; % 28 GHz

tx = txsite("Name","South Uncanoonuc (BS)", ...
    "Latitude",42.983723, ...
    "Longitude",-71.587173, ...
    "TransmitterPower",1, ...
    "TransmitterFrequency",fq);
show(tx)
```



Create Receiver Sites

Create three receiver sites in the area and show the sites on the map. Each receiver site represents a site where a user's fixed wireless terminal is placed.

```
rxBedford = rxsite("Name","Bedford Town Center", ...
    "Latitude",42.946193, ...
    "Longitude",-71.516234);
```

```
rxStA = rxsite("Name","St. Anselm College", ...
    "Latitude",42.987386, ...
    "Longitude",-71.507475);
```

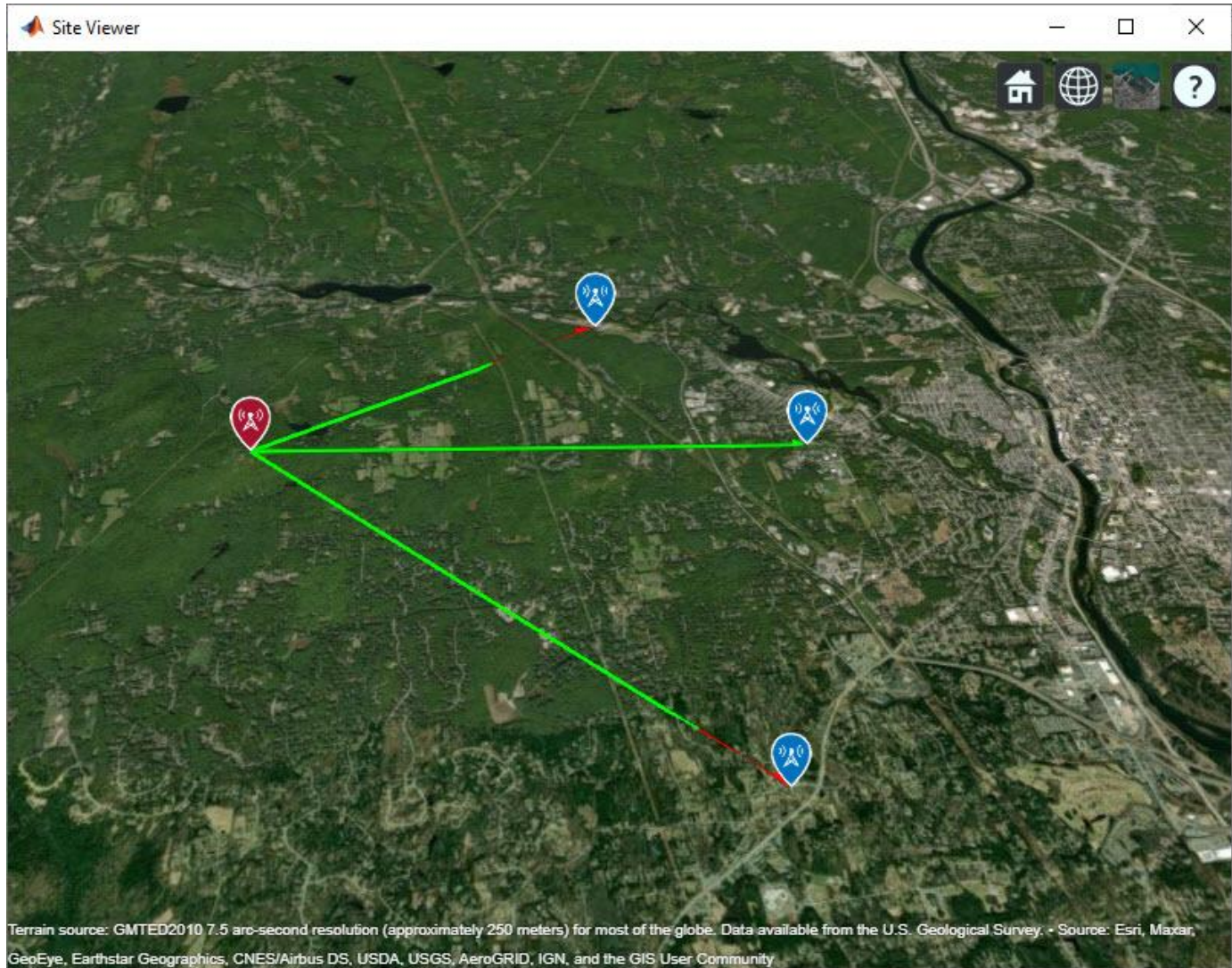
```
rxGPD = rxsite("Name","Goffstown Police Dept", ...
    "Latitude",43.009335, ...
    "Longitude",-71.539083);
```

```
rxs = [rxBedford, rxStA, rxGPD];
show(rxs)
```

Achieve Line-of-Sight Link Visibility

A challenge for 5G communication is achieving a successful link in the presence of terrain and other obstacles, since propagation losses increase at high frequency. A visible line-of-sight path is required for optimal propagation conditions. In the suburban environment considered here, terrain is the dominant obstacle to achieving line-of-sight visibility. Plot the line-of-sight propagation paths between the base station and receiver sites. The line-of-sight calculation includes terrain but no other obstacles and reveals obstructed line-of-sight with two of the three receiver sites.

los(tx,rxs)



Adjust antenna heights in order to achieve line-of-sight visibility.

% Place antennas on structures at receiver sites. Assume 6 m utility poles for Bedford
% and St. Anselm sites, and 15 m antenna pole at Goffstown Police Department.

```
rxBedford.AntennaHeight = 6;  
rxStA.AntennaHeight = 6;  
rxGPD.AntennaHeight = 15;
```



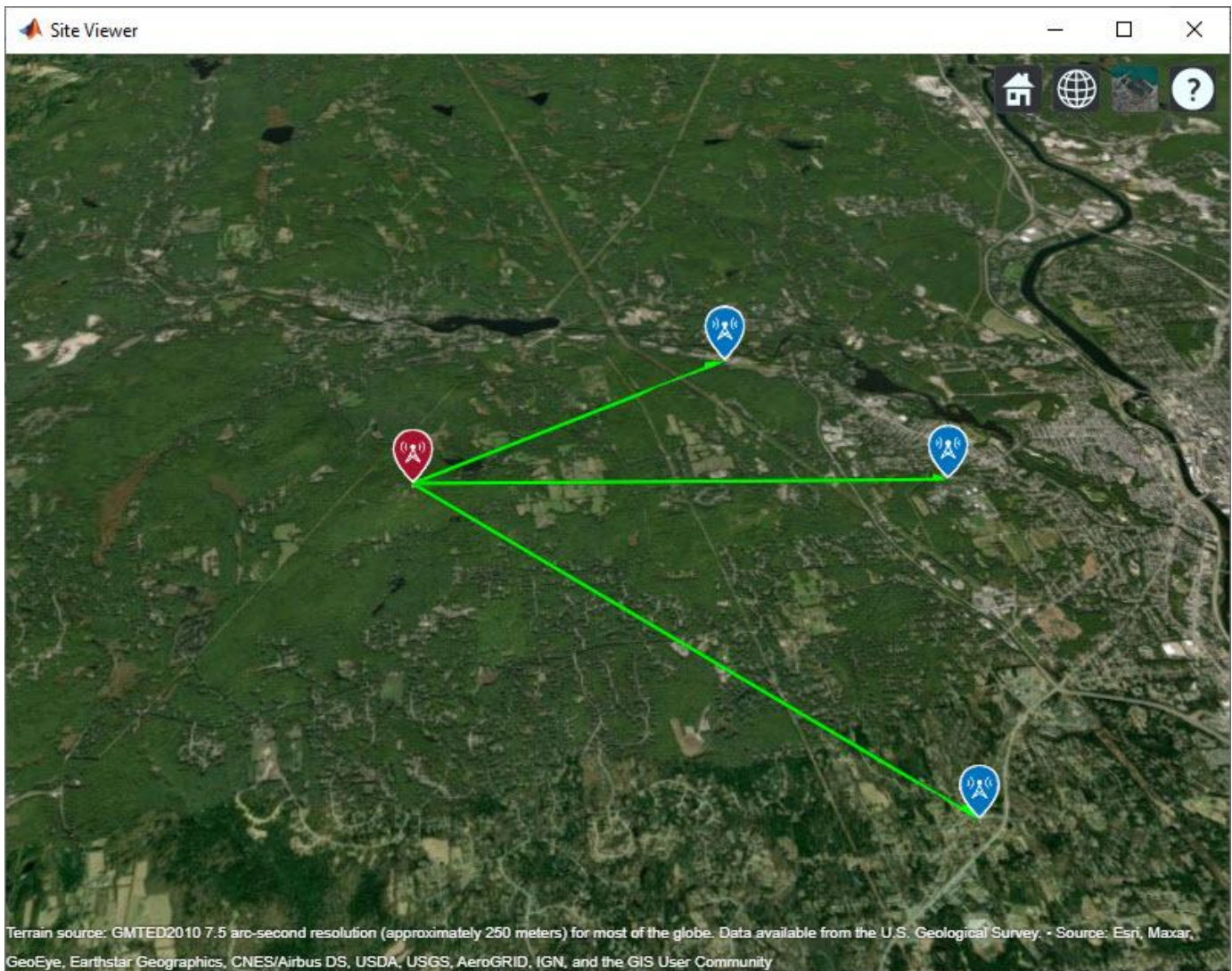
```

% Increase height of antenna at base station until line-of-sight is achieved with all receiver s
tx.AntennaHeight = 10;
while ~all(los(tx,rxs))
    tx.AntennaHeight = tx.AntennaHeight + 5;
end

% Display line-of-sight
los(tx,rxs)
disp("Antenna height required for line-of-sight: " + tx.AntennaHeight + " m")

Antenna height required for line-of-sight: 70 m

```



Create 8-by-12 Base Station Antenna Array

Design an 8-by-12 antenna array of crossed dipole antenna elements to generate a highly directive beam. This system implements a 5G concept utilizing MU-MIMO [1 on page 5-445]. Plot the radiation pattern on the map, using the default antenna orientation so that the antenna array is physically oriented in the east direction.

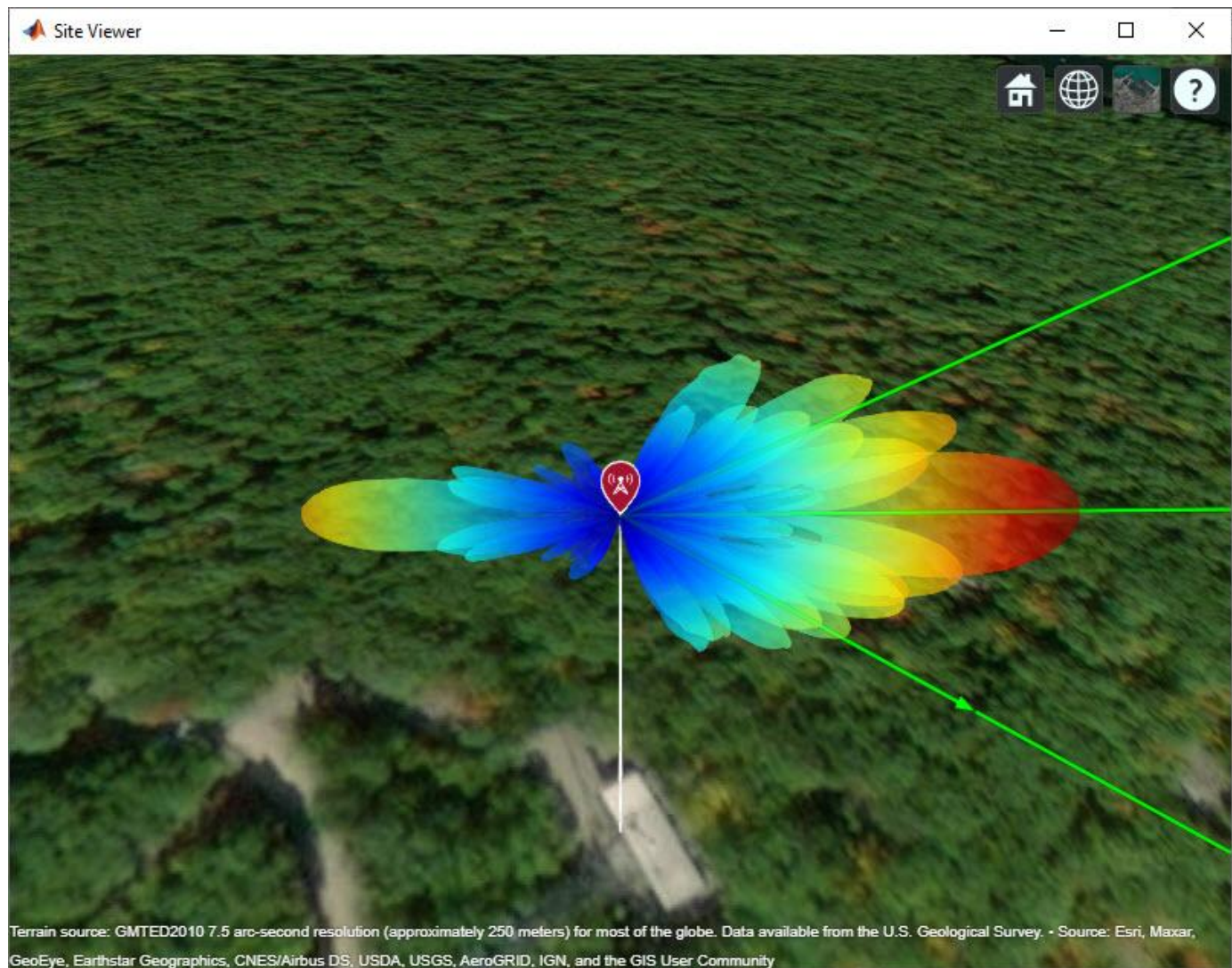
```
% Design reflector-backed crossed dipole antenna
txElement = reflectorCrossedDipoleElement(fq);

% Define array size
ntxrow = 8;
ntxcol = 12;

% Define element spacing
lambda = physconst("lightspeed")/fq;
drow = lambda/2;
dcol = lambda/2;

% Create 8-by-12 antenna array
tx.Antenna = phased.URA("Size",[ntxrow ntxcol], ...
    "Element",txElement, ...
    "ElementSpacing",[drow dcol]);

% Plot pattern on the map
pattern(tx)
```



Create 3-by-3 Receiver Site Antenna Array

Create a 3-by-3 rectangular array from a reflector-backed vertical dipole antenna element. At each receiver site, point the array toward the base station and plot the radiation pattern on the map.

```
rxElement = reflectorDipoleElement(fq);

% Define array size
nrxrow = 3;
nrxcoll = 3;

% Define element spacing
lambda = physconst("lightspeed")/fq;
drow = lambda/2;
dcol = lambda/2;

% Create antenna array
rxarray = phased.URA("Size",[nrxrow nrxcoll], ...
    "Element",rxElement, ...
    "ElementSpacing",[drow dcol]);

% Assign array to each receiver site and point toward base station
for rx = rxS
    rx.Antenna = rxarray;
    rx.AntennaAngle = angle(rx, tx);
    pattern(rx,fq)
end
```



Predict Signal Strength in Free Space with Beamforming

Use the free space propagation model to compute received signal strength for each receiver site. For each site, steer the base station beam to optimize directivity for the link. The favorable conditions assumed by free space produce strong signals at the receiver sites, assuming a receiver sensitivity of -84 dBm [2 on page 5-445].

```
steeringVector = phased.SteeringVector("SensorArray",tx.Antenna);
for rx = rxS
    % Compute steering vector for receiver site
    [az,el] = angle(tx,rx);
    sv = steeringVector(fq,[az;el]);

    % Update base station radiation pattern
    tx.Antenna.Taper = conj(sv);
    pattern(tx)

    % Compute signal strength (dBm)
    ss = sigstrength(rx,tx,"freespace");
```

```
disp("Signal strength at " + rx.Name + ":")  
disp(ss + " dBm")  
end
```

Signal strength at Bedford Town Center:

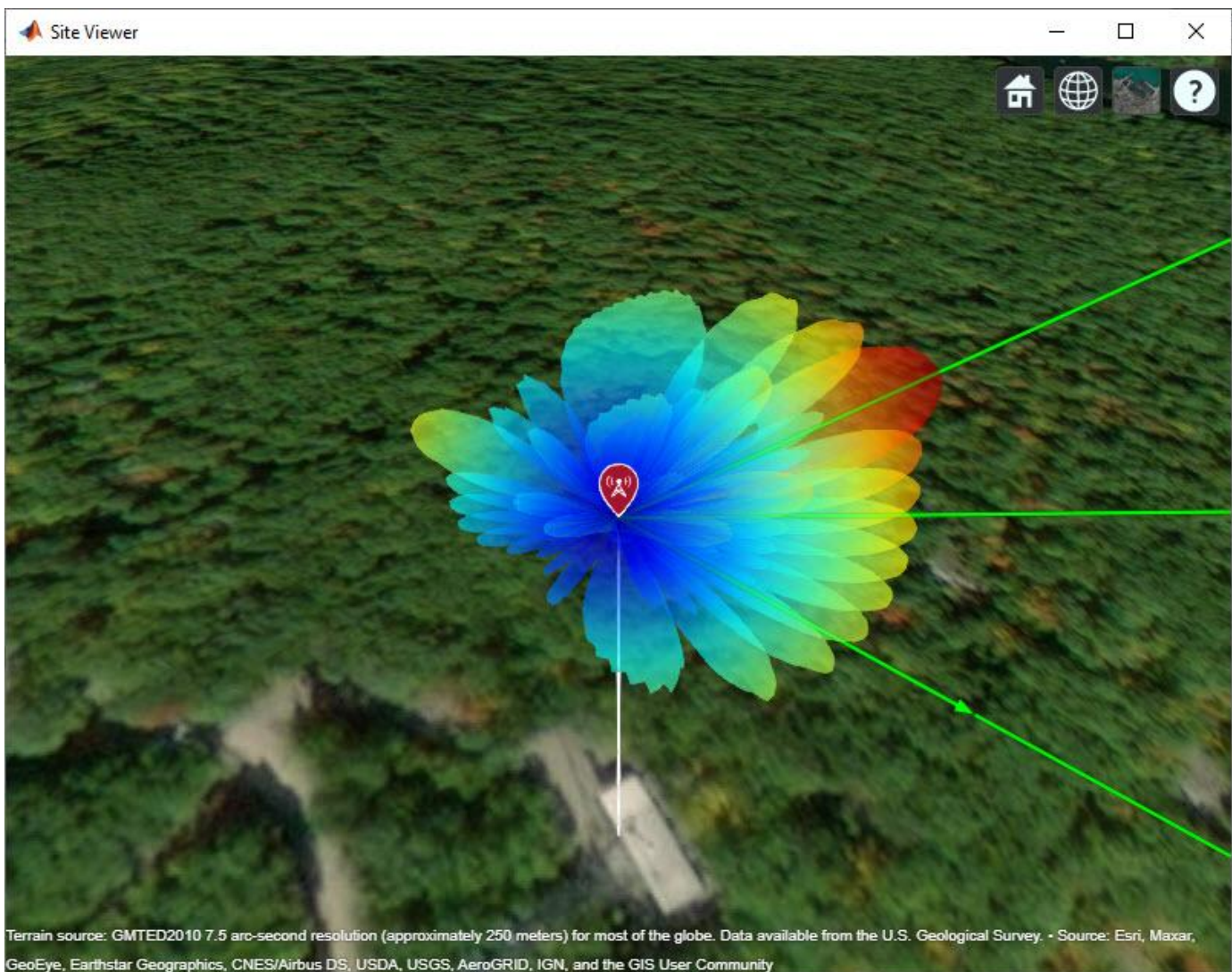
-69.6743 dBm

Signal strength at St. Anselm College:

-68.0441 dBm

Signal strength at Goffstown Police Dept:

-66.3306 dBm



Simultaneous Transmission

Instead of steering the base station antenna beam to each receiver site in turn, generate a single beam that can transmit to all receiver sites simultaneously. The single beam generates radiation lobes

toward the three receiver sites. The signal strength drops at each receiver site with simultaneous transmission but still meets the receiver sensitivity.

```
steeringVector = phased.SteeringVector("SensorArray",tx.Antenna);
```

```
% Compute steering vector for receiver site
```

```
[az,el] = angle(tx,rxs);  
sv = steeringVector(fq,[az el]');
```

```
% Update base station radiation pattern
```

```
tx.Antenna.Taper = conj(sum(sv,2));  
pattern(tx)
```

```
% Compute signal strength (dBm)
```

```
for rx = rxs  
    ss = sigstrength(rx,tx,"freespace");  
    disp("Signal strength at " + rx.Name + ":")  
    disp(ss + " dBm")  
end
```

```
Signal strength at Bedford Town Center:
```

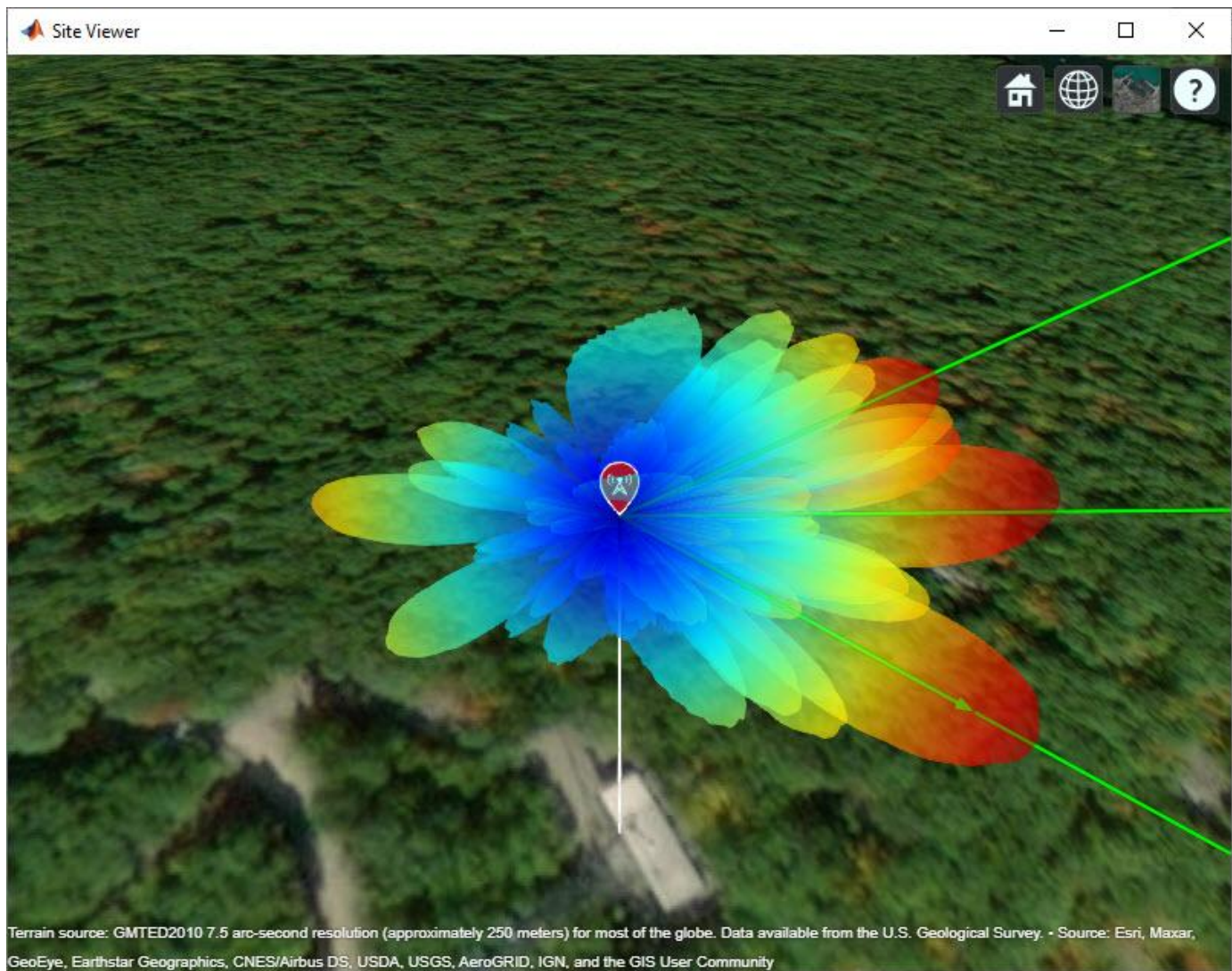
```
-75.2896 dBm
```

```
Signal strength at St. Anselm College:
```

```
-72.2968 dBm
```

```
Signal strength at Goffstown Police Dept:
```

```
-72.0328 dBm
```



Add Path Loss Impairments

Additional attenuation of the signal occurs due to foliage and weather. Use Weissberger's model [3 on page 5-445] to estimate path loss due to foliage, and use the gas and rain propagation models to estimate signal strength due to weather. In the presence of path loss impairments, the estimated signal strength becomes weak and drops below the receiver sensitivity of -84 dBm.

```
% Assume that propagation path travels through 25 m of foliage
foliageDepth = 25;
L = 1.33*((fq/1e9)^0.284)*foliageDepth^0.588; % Weissberger model for d > 14
disp("Path loss due to foliage: " + L + " dB")
```

```
Path loss due to foliage: 22.7422 dB
```

```
% Assign foliage loss as static SystemLoss on each receiver site
for rx = rxS
    rx.SystemLoss = L;
end
```

```

% Compute signal strength with foliage loss
for rx = rxS
    rx.SystemLoss = L;
    ss = sigstrength(rx,tx,"freespace");
    disp("Signal strength at " + rx.Name + ":")
    disp(ss + " dBm")
end

```

Signal strength at Bedford Town Center:

-98.0318 dBm

Signal strength at St. Anselm College:

-95.0391 dBm

Signal strength at Goffstown Police Dept:

-94.775 dBm

```

% Compute signal strength including propagation through gas and rain. Use
% the "+" operator to add the propagation models to create a composite
% model including both atmospheric effects.

```

```

weatherpm = propagationModel("gas") + propagationModel("rain");
for rx = rxS
    ss = sigstrength(rx,tx,weatherpm);
    disp("Signal strength at " + rx.Name + ":")
    disp(ss + " dBm")
end

```

Signal strength at Bedford Town Center:

-114.4897 dBm

Signal strength at St. Anselm College:

-110.4526 dBm

Signal strength at Goffstown Police Dept:

-107.3242 dBm

Performance in the 3.5 GHz Band

The 3.5 GHz band is a prominent band under consideration for 5G radio [1 on page 5-445]. Redesign the MU-MIMO system for this lower frequency to achieve more favorable path loss and achieve the required signal strength.

```
fq = 3.5e9; % 3.5 GHz
```

```

% Create antenna array for base station
lambda = physconst("lightspeed")/fq;
drow = lambda/2;
dcol = lambda/2;
tx.TransmitterFrequency = fq;
tx.Antenna = phased.URA("Size",[ntxrow ntxcol], ...
    "Element",reflectorCrossedDipoleElement(fq), ...
    "ElementSpacing",[drow dcol]);

```

```

% Create antenna array for receiver sites

```



```

lambda = physconst("lightspeed")/fq;
drow = lambda/2;
dcol = lambda/2;
rxarray = phased.URA("Size",[nrxrow nrxcol], ...
    "Element",reflectorDipoleElement(fq), ...
    "ElementSpacing",[drow dcol], ...
    "ArrayNormal","x");
for rx = rxS
    rx.Antenna = rxarray;
end

```

In addition to computing signal strength at each receiver site, generate a coverage map using the Longley-Rice propagation model with weather impairments. The Longley-Rice model, which is also known as the Irregular Terrain Model (ITM), estimates path loss based on diffraction and other losses derived from terrain. The Longley-Rice model is valid from 20 MHz to 20 GHz and is therefore available for 3.5 GHz but not for 28 GHz.

```

% Compute steering vector for receiver site
steeringVector = phased.SteeringVector("SensorArray",tx.Antenna);
[az,el] = angle(tx,rxS);
sv = steeringVector(fq,[az el]');

% Update base station radiation pattern
tx.Antenna.Taper = conj(sum(sv,2));
pattern(tx,'Size',4000)

% Recompute loss due to foliage
L = 1.33*((fq/1e9)^0.284)*foliageDepth^0.588; % Weissberger model for d > 14

% Assign foliage loss as static SystemLoss on each receiver site
for rx = rxS
    rx.SystemLoss = L;
end
disp("Path loss due to foliage: " + L + " dB")

Path loss due to foliage: 12.5996 dB

% Add weather-based path loss to the Longley-Rice propagation model
pm = propagationModel('longley-rice') + weatherpm;

% Compute receiver gain from peak antenna gain and system loss
G = pattern(rxarray, fq);
rxGain = max(G(:)) - L;

coverage(tx, ...
    'PropagationModel',pm, ...
    'ReceiverGain',rxGain, ...
    'ReceiverAntennaHeight',6, ...
    'SignalStrengths',-84:-50)

% Compute signal strength with foliage loss and weather
for rx = rxS
    ss = sigstrength(rx,tx,pm);
    disp("Signal strength at " + rx.Name + ":")
    disp(ss + " dBm")
end

Signal strength at Bedford Town Center:

```

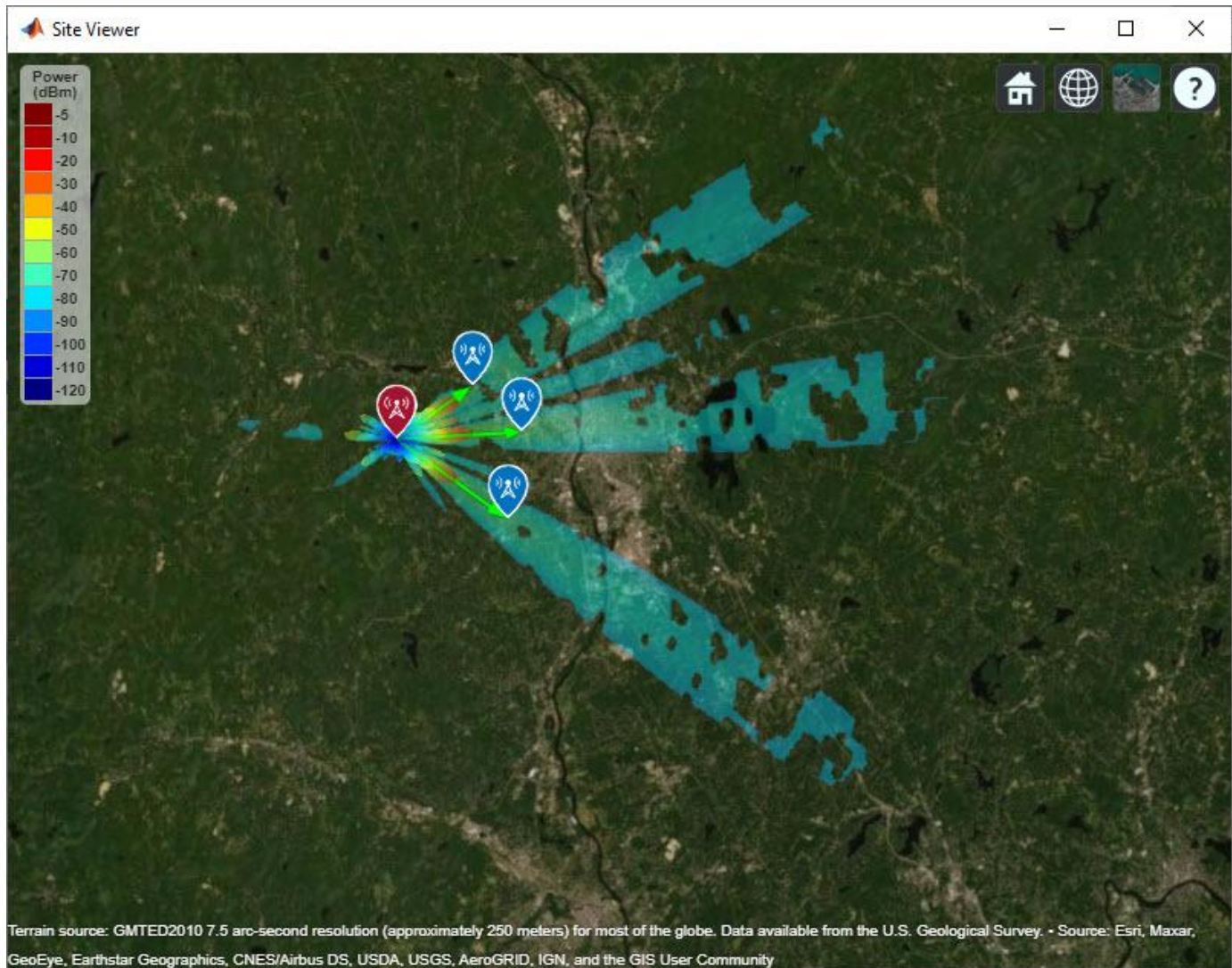
-69.9048 dBm

Signal strength at St. Anselm College:

-66.8941 dBm

Signal strength at Goffstown Police Dept:

-66.6094 dBm



Summary

This example shows how to plan a fixed wireless access link over terrain using 5G technologies in a multi-user suburban scenario. While line-of-sight propagation is achieved over the terrain, path loss impairments render the 28 GHz carrier frequency unsuitable for the links despite the use of high gain antennas and beamforming. The addition of foliage loss alone drops the signal strength below the receiver sensitivity of -84 dBm, and the addition of weather loss significantly drops it further. The lower frequency of 3.5 GHz is required to achieve successful links at the multi-kilometer ranges considered here. As a result, this example illustrates the sensitivity of high 5G carrier frequencies to common path loss impairments.

References

[1] Ericsson Technology Review, Fixed wireless access on a massive scale with 5G, Anders Furuskär, Kim Laraqui, Sibel Tombaz, Ala Nazari, Björn Skubic, Elmar Trojer, December, 2016

[2] Microwave Journal, Pre-5G and 5G: Will The mmWave Link Work?, Andreas Roessler, December, 2017

[3] John Seybold, Introduction to RF Propagation, Wiley, 2005

```
function element = reflectorCrossedDipoleElement(fq, showAntenna)
%reflectorCrossedDipoleElement  Design reflector-backed crossed dipole antenna element

if nargin < 2
    showAntenna = false;
end

lambda = physconst("lightspeed")/fq;
offset = lambda/50;
gndspacing = lambda/4;
gndLength = lambda;
gndWidth = lambda;

% Design crossed dipole elements
d1 = design(dipole,fq);
d1.Tilt = [90,-45];
d1.TiltAxis = ["y","z"];
d2 = copy(d1);
d2.Tilt = 45;
d2.TiltAxis = "x";

% Design reflector
r = design(reflector,fq);
r.Exciter = d1;
r.GroundPlaneLength = gndLength;
r.GroundPlaneWidth = gndWidth;
r.Spacing = gndspacing;
r.Tilt = 90;
r.TiltAxis = "y";
if showAntenna
    show(r)
end

% Form the crossed dipole backed by reflector
refarray = conformalArray;
refarray.ElementPosition(1,:) = [gndspacing 0 0];
refarray.ElementPosition(2,:) = [gndspacing+offset 0 0];
refarray.Element = {r, d2};
refarray.Reference = "feed";
refarray.PhaseShift = [0 90];
if showAntenna
    show(refarray);
    view(65,20)
end
```

```
% Create custom antenna element from pattern
[g,az,el] = pattern(refarray,fq);
element = phased.CustomAntennaElement;
element.AzimuthAngles = az;
element.ElevationAngles = el;
element.MagnitudePattern = g;
element.PhasePattern = zeros(size(g));
end

function element = reflectorDipoleElement(fq)
%reflectorDipoleElement  Design reflector-backed dipole antenna element

% Design reflector and exciter, which is vertical dipole by default
element = design(reflector,fq);
element.Exciter = design(element.Exciter,fq);

% Tilt antenna element to radiate in xy-plane, with boresight along x-axis
element.Tilt = 90;
element.TiltAxis = "y";
element.Exciter.Tilt = 90;
element.Exciter.TiltAxis = "y";
end
```

See Also

Functions

los | pattern

Objects

txsite | rxsite | phased.URA | phased.SteeringVector

Related Examples

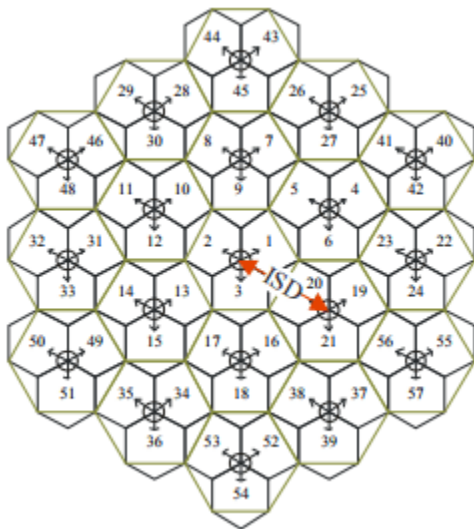
- “SINR Map for a 5G Urban Macro-Cell Test Environment” on page 5-447

SINR Map for a 5G Urban Macro-Cell Test Environment

This example shows how to construct a 5G urban macro-cell test environment and visualize the signal-to-interference-plus-noise ratio (SINR) on a map. The test environment is based on the guidelines defined in Report ITU-R M.[IMT-2020.EVAL] [1 on page 5-458] for evaluating 5G radio technologies. This report defines several test environments and usage scenarios in Section 8.2. The test environment in this example is based on the urban environment with high user density and traffic loads focusing on pedestrian and vehicular users (Dense Urban-eMBB). The test environment includes a hexagonal cell network as well as a custom antenna array that is implemented using Phased Array System Toolbox™.

Define Network Layout

The test environment guidelines for 5G technologies reuse the test network layout for 4G technologies defined in Section 8.3 of Report ITU-R M.2135-1 [2 on page 5-458], which is shown below. The layout consists of 19 sites placed in a hexagonal layout, each with 3 cells. The distance between adjacent sites is the inter-site distance (ISD) and depends on the test usage scenario. For the Dense Urban-eMBB test environment, the ISD is 200 m.



Create the locations corresponding to cell sites in the network layout, using MathWorks® Glasgow as the center location.

```
% Define center location site (cells 1-3)
centerSite = txsite('Name','MathWorks Glasgow', ...
    'Latitude',55.862787,...
    'Longitude',-4.258523);

% Initialize arrays for distance and angle from center location to each cell site, where
% each site has 3 cells
numCellSites = 19;
siteDistances = zeros(1,numCellSites);
siteAngles = zeros(1,numCellSites);

% Define distance and angle for inner ring of 6 sites (cells 4-21)
isd = 200; % Inter-site distance
```

```

siteDistances(2:7) = isd;
siteAngles(2:7) = 30:60:360;

% Define distance and angle for middle ring of 6 sites (cells 22-39)
siteDistances(8:13) = 2*isd*cosd(30);
siteAngles(8:13) = 0:60:300;

% Define distance and angle for outer ring of 6 sites (cells 40-57)
siteDistances(14:19) = 2*isd;
siteAngles(14:19) = 30:60:360;

```

Define Cell Parameters

Each cell site has three transmitters corresponding to each cell. Create arrays to define the names, latitudes, longitudes, and antenna angles of each cell transmitter.

```

% Initialize arrays for cell transmitter parameters
numCells = numCellSites*3;
cellLats = zeros(1,numCells);
cellLons = zeros(1,numCells);
cellNames = strings(1,numCells);
cellAngles = zeros(1,numCells);

% Define cell sector angles
cellSectorAngles = [30 150 270];

% For each cell site location, populate data for each cell transmitter
cellInd = 1;
for siteInd = 1:numCellSites
    % Compute site location using distance and angle from center site
    [cellLat,cellLon] = location(centerSite, siteDistances(siteInd), siteAngles(siteInd));

    % Assign values for each cell
    for cellSectorAngle = cellSectorAngles
        cellNames(cellInd) = "Cell " + cellInd;
        cellLats(cellInd) = cellLat;
        cellLons(cellInd) = cellLon;
        cellAngles(cellInd) = cellSectorAngle;
        cellInd = cellInd + 1;
    end
end
end

```

Create Transmitter Sites

Create transmitter sites using parameters defined above as well as configuration parameters defined for Dense Urban-eMBB. Launch Site Viewer and set the map imagery using the **Basemap** property. Alternatively, open the basemap picker in Site Viewer by clicking the second button from the right. Select "Topographic" to choose a basemap with topography, streets, and labels.

```

% Define transmitter parameters using Table 8-2 (b) of Report ITU-R M.[IMT-2020.EVAL]
fq = 4e9; % Carrier frequency (4 GHz) for Dense Urban-eMBB
antHeight = 25; % m
txPowerDBm = 44; % Total transmit power in dBm
txPower = 10.^((txPowerDBm-30)/10); % Convert dBm to W

% Create cell transmitter sites
txs = txsite('Name',cellNames, ...
            'Latitude',cellLats, ...

```

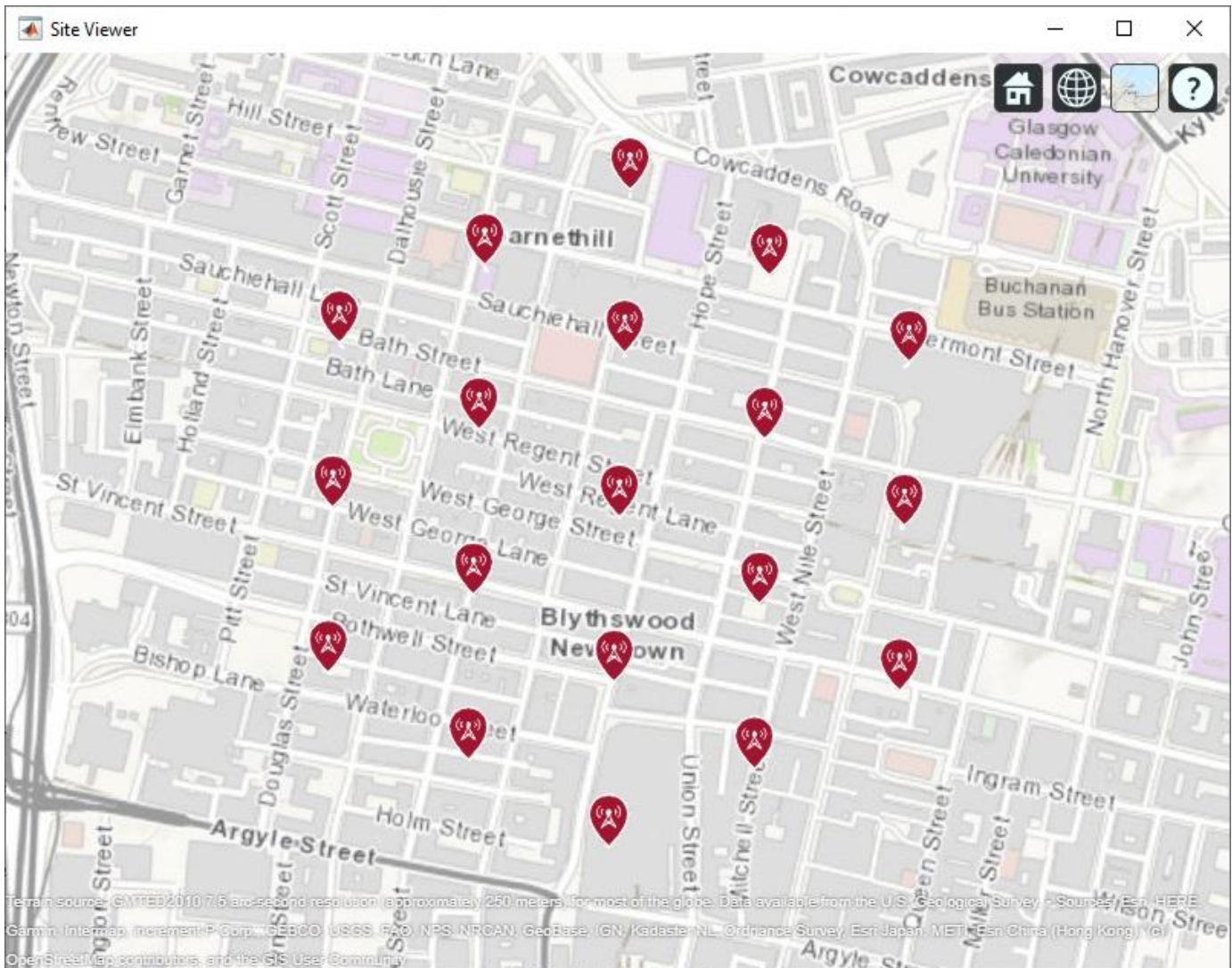
```

'Longitude',cellLons, ...
'AntennaAngle',cellAngles, ...
'AntennaHeight',antHeight, ...
'TransmitterFrequency',fq, ...
'TransmitterPower',txPower);

% Launch Site Viewer
viewer = siteviewer;

% Show sites on a map
show(txs);
viewer.Basemap = 'topographic';

```



Create Antenna Element

Section 8.5 of ITU-R report [1 on page 5-458] defines antenna characteristics for base station antennas. The antenna is modeled as having one or more antenna panels, where each panel has one or more antenna elements. Use Phased Array System Toolbox to implement the antenna element pattern defined in the report.

```

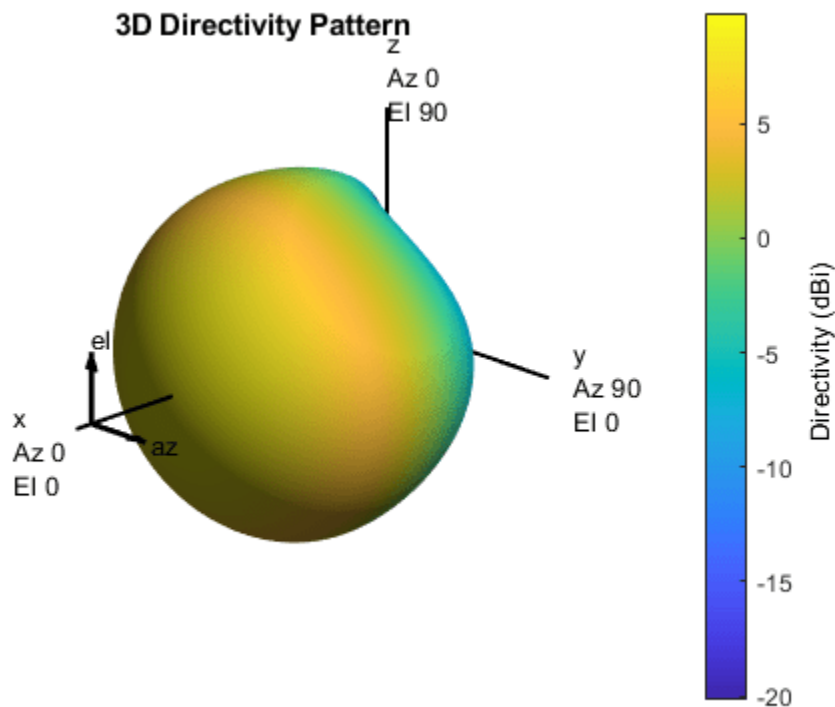
% Define pattern parameters
azvec = -180:180;
elvec = -90:90;
Am = 30; % Maximum attenuation (dB)
tilt = 0; % Tilt angle
az3dB = 65; % 3 dB bandwidth in azimuth
el3dB = 65; % 3 dB bandwidth in elevation

% Define antenna pattern
[az,el] = meshgrid(azvec,elvec);
azMagPattern = -12*(az/az3dB).^2;
elMagPattern = -12*((el-tilt)/el3dB).^2;
combinedMagPattern = azMagPattern + elMagPattern;
combinedMagPattern(combinedMagPattern<-Am) = -Am; % Saturate at max attenuation
phasepattern = zeros(size(combinedMagPattern));

% Create antenna element
antennaElement = phased.CustomAntennaElement(...
    'AzimuthAngles',azvec, ...
    'ElevationAngles',elvec, ...
    'MagnitudePattern',combinedMagPattern, ...
    'PhasePattern',phasepattern);

% Display radiation pattern
f = figure;
pattern(antennaElement,fq);

```



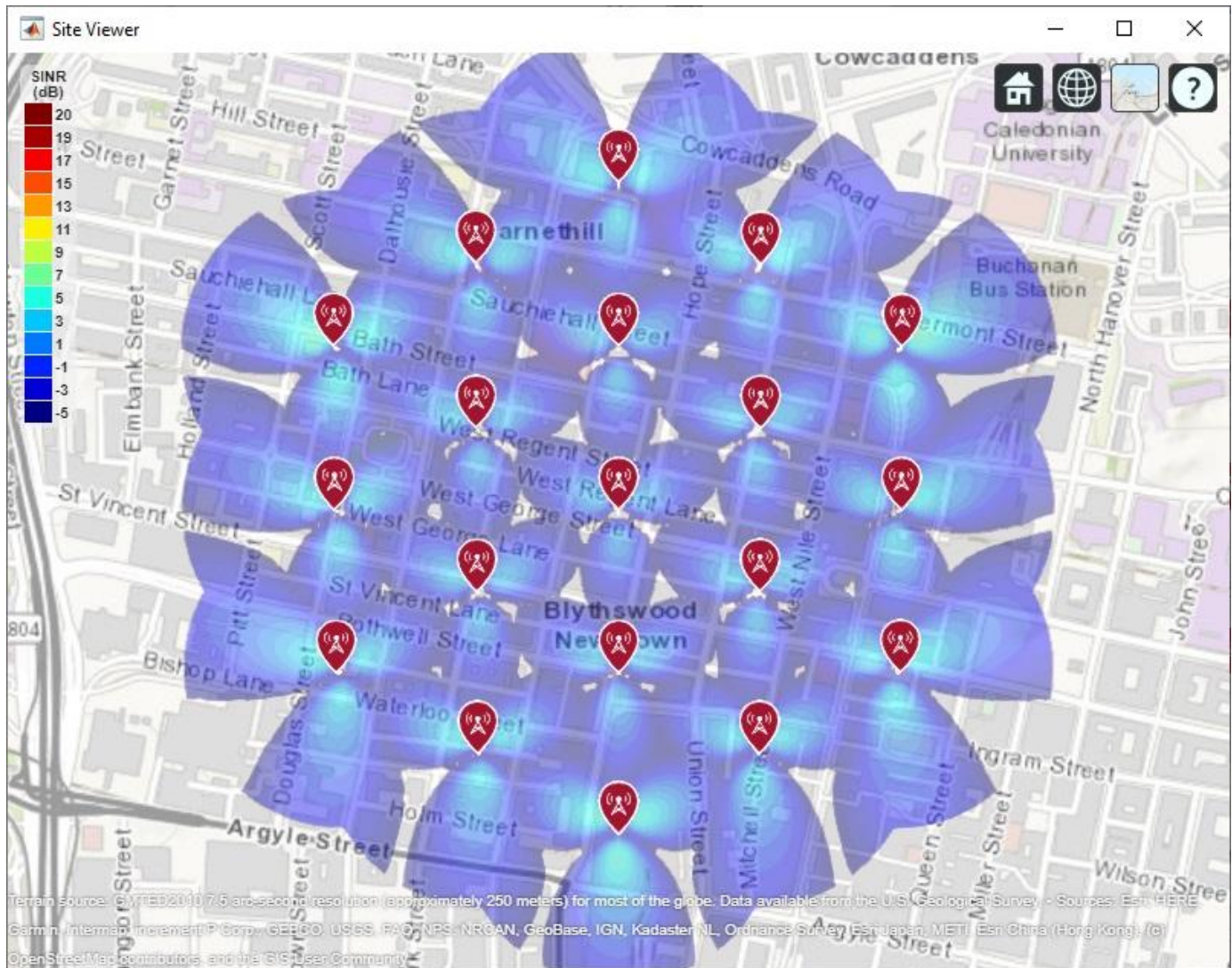
Display SINR Map for Single Antenna Element

Visualize SINR for the test scenario using a single antenna element and the free space propagation model. For each location on the map within the range of the transmitter sites, the signal source is the cell with the greatest signal strength, and all other cells are sources of interference. Areas with no color within the network indicate areas where the SINR is below the default threshold of -5 dB.

```
% Assign the antenna element for each cell transmitter
for tx = txs
    tx.Antenna = antennaElement;
end

% Define receiver parameters using Table 8-2 (b) of Report ITU-R M.[IMT-2020.EVAL]
bw = 20e6; % 20 MHz bandwidth
rxNoiseFigure = 7; % dB
rxNoisePower = -174 + 10*log10(bw) + rxNoiseFigure;
rxGain = 0; % dBi
rxAntennaHeight = 1.5; % m

% Display SINR map
if invalid(f)
    close(f)
end
sinr(txs,'freespace', ...
    'ReceiverGain',rxGain, ...
    'ReceiverAntennaHeight',rxAntennaHeight, ...
    'ReceiverNoisePower',rxNoisePower, ...
    'MaxRange',isd, ...
    'Resolution',isd/20)
```



Create 8-by-8 Rectangular Antenna Array

Define an antenna array to increase directional gain and increase peak SINR values. Use Phased Array System Toolbox to create an 8-by-8 uniform rectangular array.

```
% Define array size
```

```
nrow = 8;
```

```
ncol = 8;
```

```
% Define element spacing
```

```
lambda = physconst('lightspeed')/fq;
```

```
drow = lambda/2;
```

```
dcol = lambda/2;
```

```
% Define taper to reduce sidelobes
```

```
dBdown = 30;
```

```
taperz = chebwin(nrow,dBdown);
```

```
tapery = chebwin(ncol,dBdown);
```

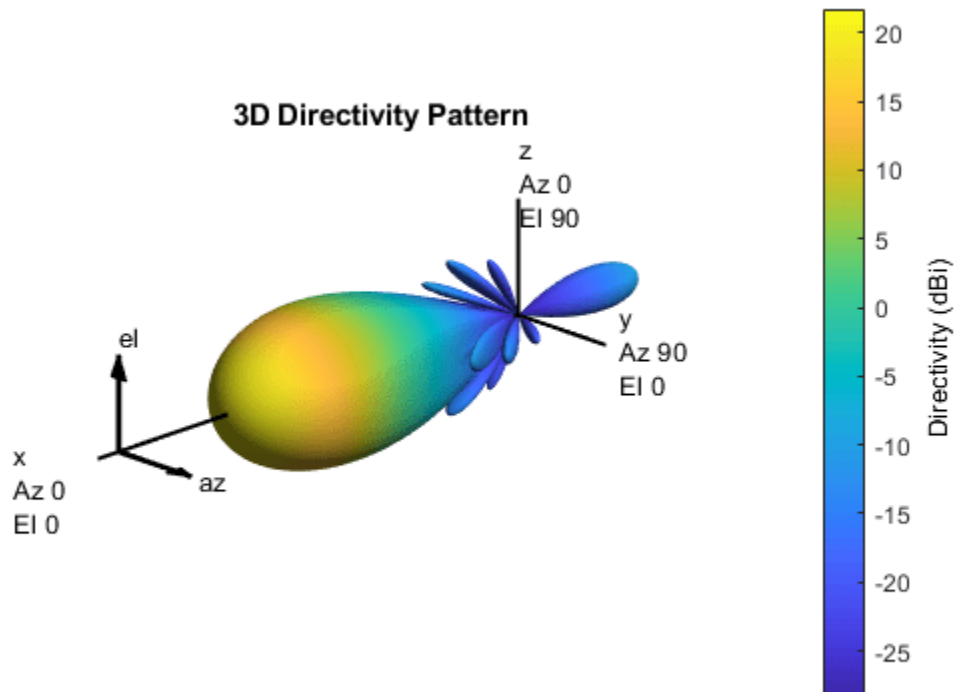
```
tap = taperz*tapery.'; % Multiply vector tapers to get 8-by-8 taper values
```

```

% Create 8-by-8 antenna array
cellAntenna = phased.URA('Size',[nrow ncol], ...
    'Element',antennaElement, ...
    'ElementSpacing',[drow dcol], ...
    'Taper',tap, ...
    'ArrayNormal','x');

% Display radiation pattern
f = figure;
pattern(cellAntenna,fq);

```



Display SINR Map for 8-by-8 Antenna Array

Visualize SINR for the test scenario using a uniform rectangular antenna array and the free space propagation model. Apply a mechanical downtilt to illuminate the intended ground area around each transmitter.

```

% Assign the antenna array for each cell transmitter, and apply downtilt.
% Without downtilt, pattern is too narrow for transmitter vicinity.
downtilt = 15;
for tx = txs
    tx.Antenna = cellAntenna;
    tx.AntennaAngle = [tx.AntennaAngle; -downtilt];
end

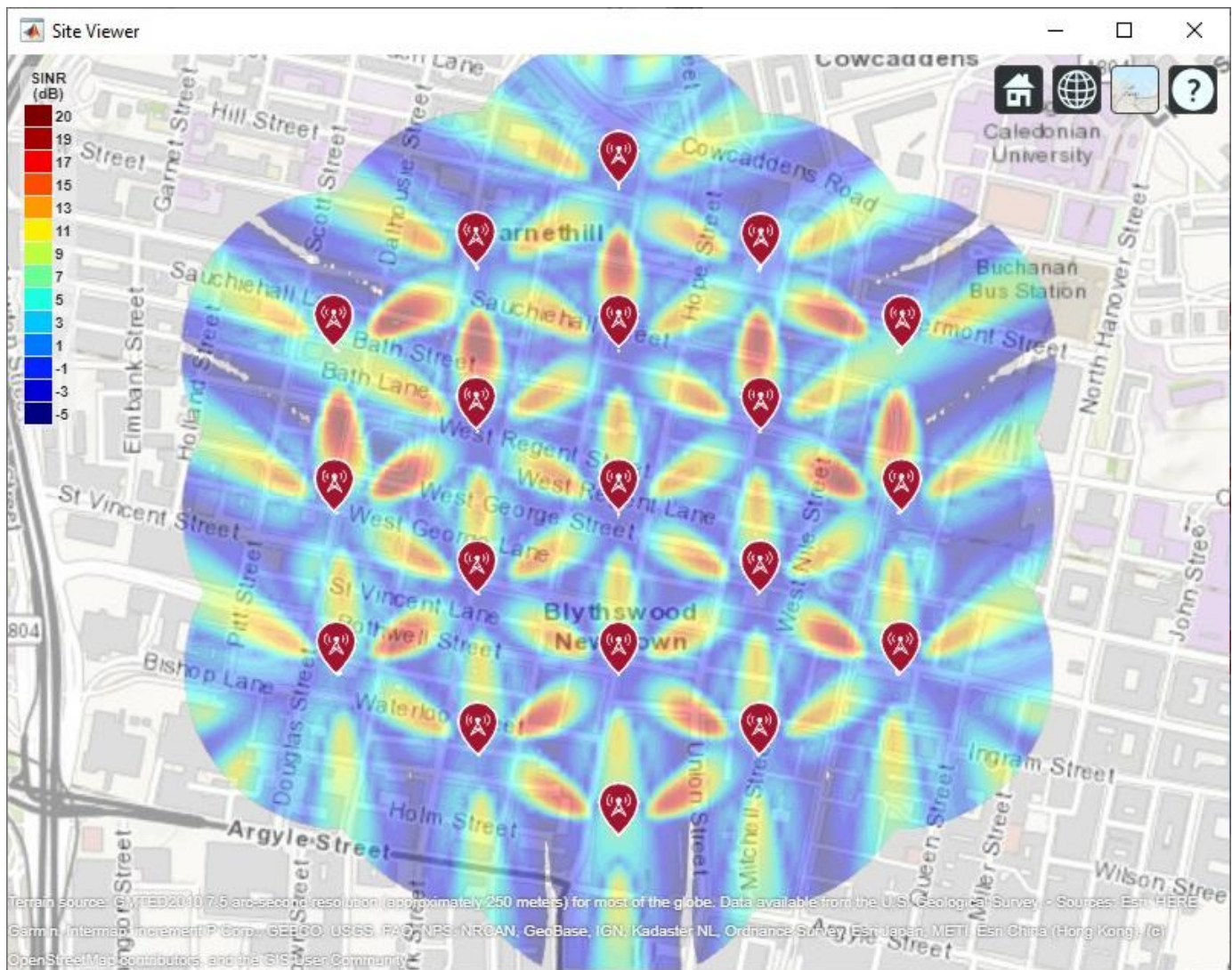
% Display SINR map
if invalid(f)

```

```

close(f)
end
sinr(txs,'freespace', ...
'ReceiverGain',rxGain, ...
'ReceiverAntennaHeight',rxAntennaHeight, ...
'ReceiverNoisePower',rxNoisePower, ...
'MaxRange',isd, ...
'Resolution',isd/20)

```



Display SINR Map Using Close-In Propagation Model

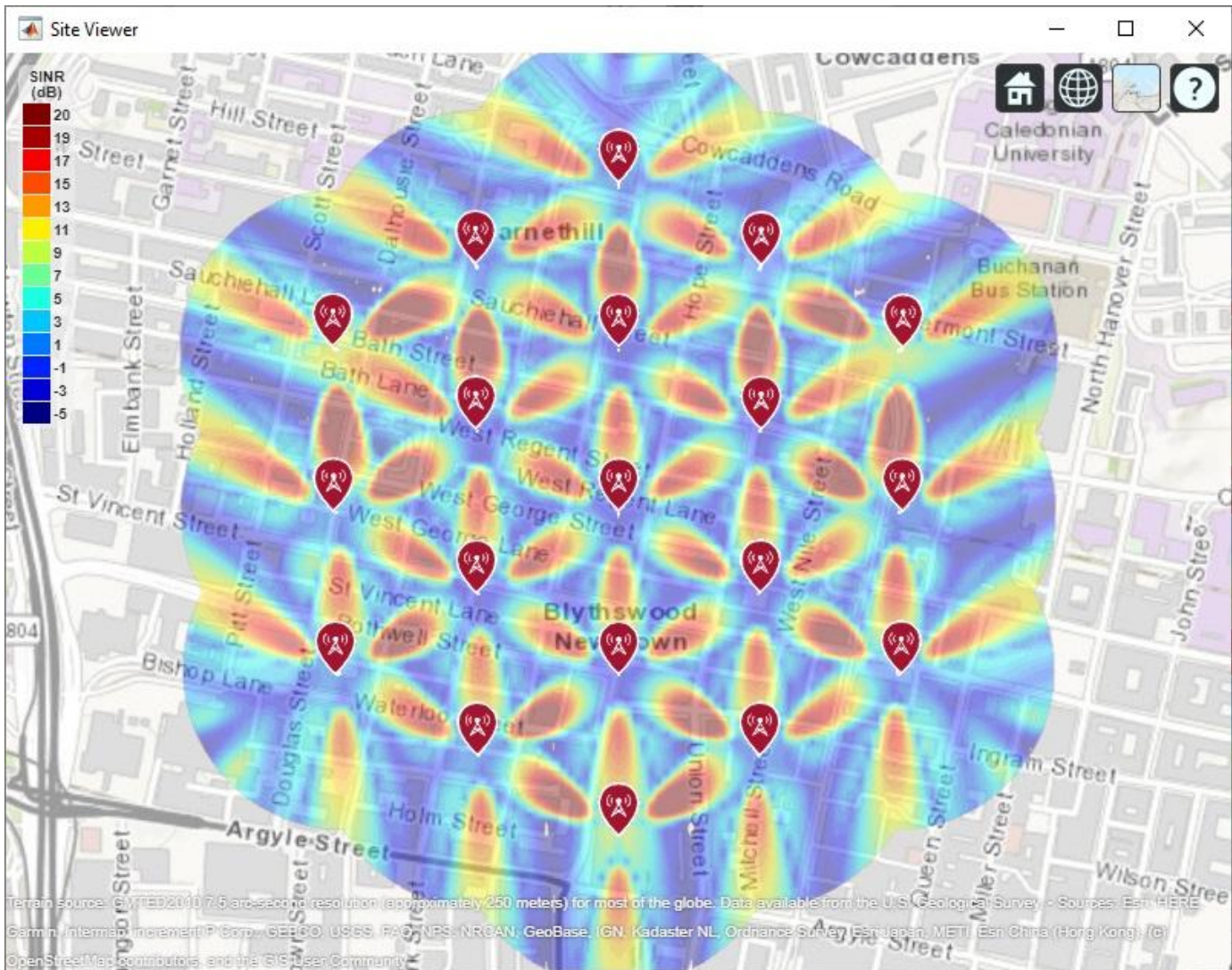
Visualize SINR for the test scenario using the Close-In propagation model [3 on page 5-458], which models path loss for 5G urban micro-cell and macro-cell scenarios. This model produces an SINR map that shows reduced interference effects compared to the free space propagation model.

```

sinr(txs,'close-in', ...
'ReceiverGain',rxGain, ...
'ReceiverAntennaHeight',rxAntennaHeight, ...
'ReceiverNoisePower',rxNoisePower, ...

```

```
'MaxRange',isd, ...
'Resolution',isd/20)
```



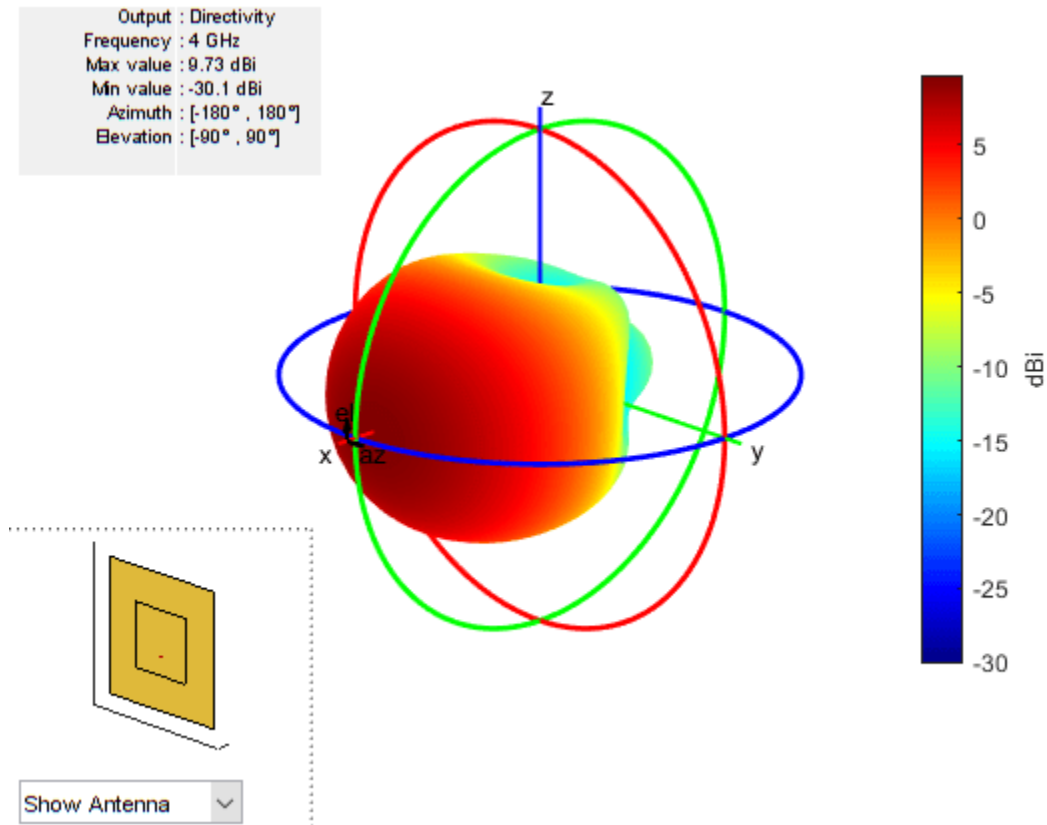
Use Rectangular Patch Antenna as Array Element

The analysis above used an antenna element that was defined using the equations specified in the ITU-R report [1 on page 5-458]. The antenna element needs to provide a maximum gain of 9.5 dBi and a front-to-back ratio of approximately 30 dB. Now replace the equation-based antenna element definition with a real antenna model using a standard half-wavelength rectangular microstrip patch antenna. The antenna element provides a gain of about 9 dBi, although with a lower front-to-back ratio.

```
% Design half-wavelength rectangular microstrip patch antenna
patchElement = design(patchMicrostrip,fq);
patchElement.Width = patchElement.Length;
patchElement.Tilt = 90;
patchElement.TiltAxis = [0 1 0];
```

```
% Display radiation pattern
```

```
f = figure;
pattern(patchElement, fq)
```



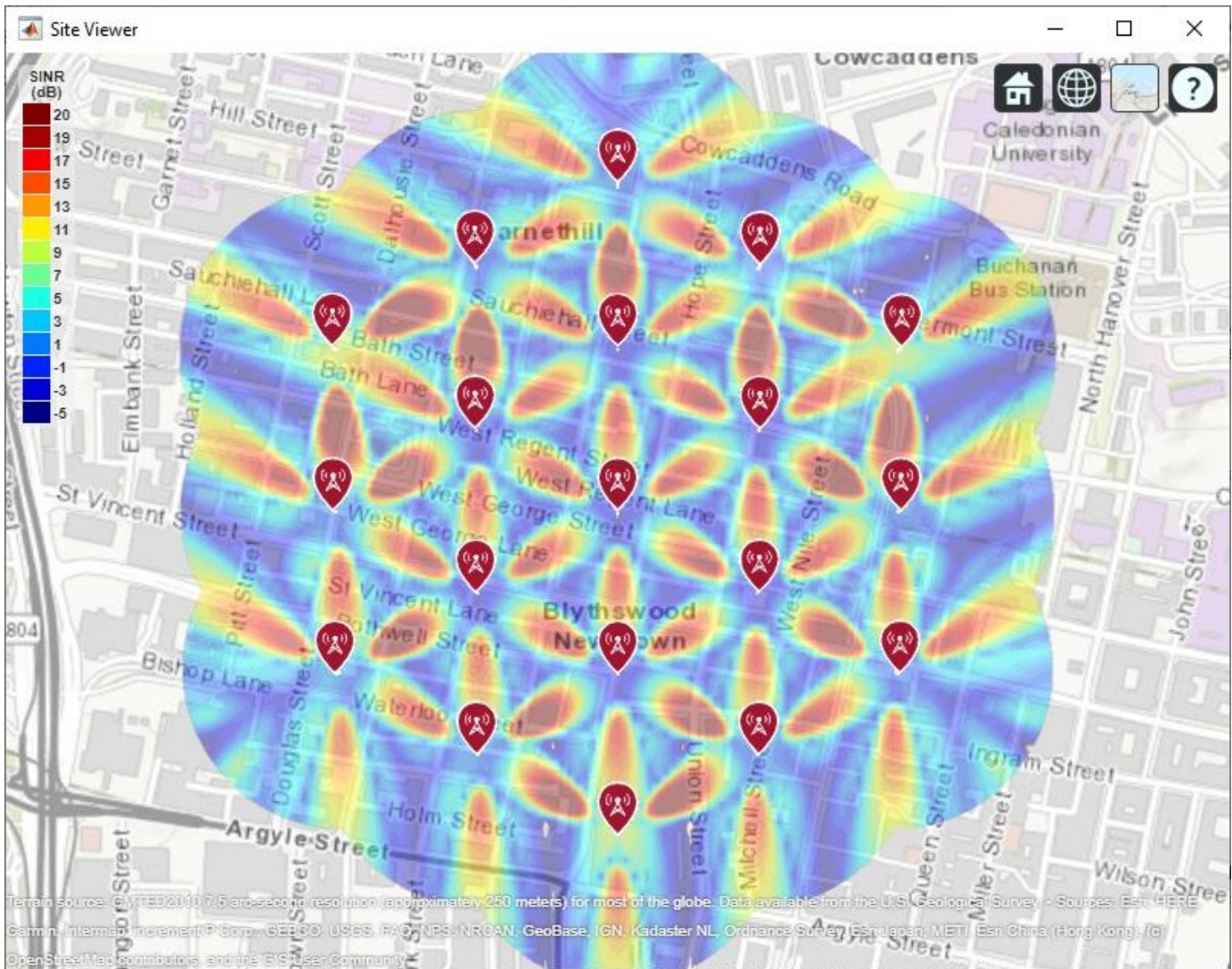
Display SINR Map Using the Patch Antenna Element in the 8-by-8 Array

Update the SINR map for the Close-In propagation model [3 on page 5-458] using the patch antenna as the array element. This analysis should capture the effect of deviations from an equation-based antenna specification as per the ITU-R report [1 on page 5-458], including:

- Variations in peak gain
- Variations in pattern symmetry with spatial angles
- Variations in front-to-back ratios

```
% Assign the patch antenna as the array element
cellAntenna.Element = patchElement;
```

```
% Display SINR map
if invalid(f)
    close(f)
end
sinr(txs, 'close-in', ...
    'ReceiverGain', rxGain, ...
    'ReceiverAntennaHeight', rxAntennaHeight, ...
    'ReceiverNoisePower', rxNoisePower, ...
    'MaxRange', isd, ...
    'Resolution', isd/20)
```



Summary

This example shows how to construct a 5G urban macro-cell test environment consisting of a hexagonal network of 19 cell sites, each containing 3 sectored cells. The signal-to-interference-plus-noise ratio (SINR) is visualized on a map for different antennas. The following observations are made:

- A rectangular antenna array can provide greater directionality and therefore peak SINR values than use of a single antenna element.
- The outward-facing lobes on the perimeter of the SINR map represent areas where less interference occurs. A more realistic modelling technique would be to replicate, or wrap around, cell sites to expand the geometry so that perimeter areas experience similar interference as interior areas.
- Using a rectangular antenna array, a propagation model that estimates increased path loss also results in higher SINR values due to less interference.
- Two antenna elements are tried in the antenna array: an equation-based element using Phased Array System Toolbox and a patch antenna element using Antenna Toolbox™. These produce similar SINR maps.

References

[1] Report ITU-R M.[IMT-2020.EVAL], "Guidelines for evaluation of radio interface technologies for IMT-2020", 2017. <https://www.itu.int/md/R15-SG05-C-0057>

[2] Report ITU-R M.2135-1, "Guidelines for evaluation of radio interface technologies for IMT-Advanced", 2009. https://www.itu.int/dms_pub/itu-r/opb/rep/R-REP-M.2135-1-2009-PDF-E.pdf

[3] Sun, S., Rapport, T.S., Thomas, T., Ghosh, A., Nguyen, H., Kovacs, I., Rodriguez, I., Koymen, O., and Prartyka, A. "Investigation of prediction accuracy, sensitivity, and parameter stability of large-scale propagation path loss models for 5G wireless communications." *IEEE Transactions on Vehicular Technology*, Vol 65, No.5, pp.2843-2860, May 2016.

See Also

Functions

`sinr` | `pattern`

Objects

`txsite` | `siteviewer` | `phased.CustomAntennaElement` | `phased.URA`

Related Examples

- "Effect of Mutual Coupling on MIMO Communication" on page 5-281

Surrogate Based Optimization Design of Six-Element Yagi-Uda Antenna

This example optimizes a 6-element Yagi-Uda antenna for both directivity and 50Ω input match using a global optimization technique called surrogate optimization. The radiation patterns and input impedance of antennas are sensitive to the parameters that define their shapes. The multidimensional surface over which such optimizations must be performed have multiple local optima. This makes the task of finding the right set of parameters satisfying the optimization goals particularly challenging and requires the use of global optimization techniques. This Yagi-Uda antenna is intended for operation as part of a repeater station in a HAM radio setup.

Design Parameters

Choose initial design parameters in the center of the VHF band.

```
fc = 144.5e6;
wirediameter = 12e-3;
c = physconst('lightspeed');
lambda = c/fc;
Z0 = 50;
BW = 0.015*fc;
fmin = fc - 2*(BW);
fmax = fc + 2*(BW);
Nf = 101;
freq = linspace(fmin, fmax, Nf);
```

Create Yagi-Uda Antenna

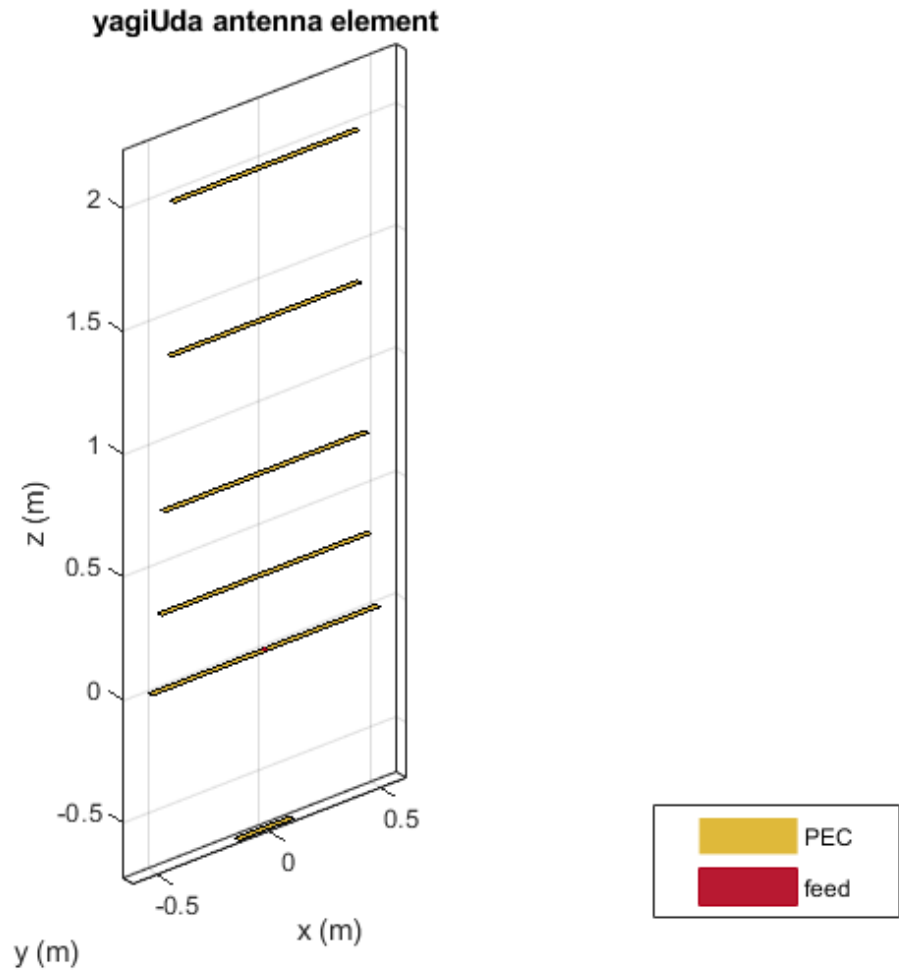
The driven element for this Yagi-Uda antenna is a dipole. This is a standard exciter for such an antenna. Adjust the length and width parameters of the dipole. Since we model cylindrical structures as equivalent metal strips, the width is calculated using a utility function available in the Antenna Toolbox™. The length is chosen to be approximately $\lambda/2$ at the design frequency.

```
d = dipole;
d.Length = 0.982.*(lambda/2);
d.Width = cylinder2strip(wirediameter/2);
d.Tilt = 90;
d.TiltAxis = 'Y';
```

Create a Yagi-Uda antenna with the exciter as the dipole. Set the number of directors to four. The choices for the lengths of the elements and spacing between the elements are an initial guess and will serve as a start point for the optimization procedure. Show the initial design.

```
Numdirs = 4;
refLength = 0.25;
dirLength = [0.940 0.910 0.850 0.830];
refSpacing = 0.35;
dirSpacing = [0.15 0.2 0.3 0.3 ];
initialdesign = [refLength refSpacing dirSpacing].*lambda;
yagidesign = yagiUda;
yagidesign.Exciter = d;
yagidesign.NumDirectors = Numdirs;
yagidesign.ReflectorLength = refLength;
yagidesign.DirectorLength = dirLength;
yagidesign.ReflectorSpacing = refSpacing*lambda;
```

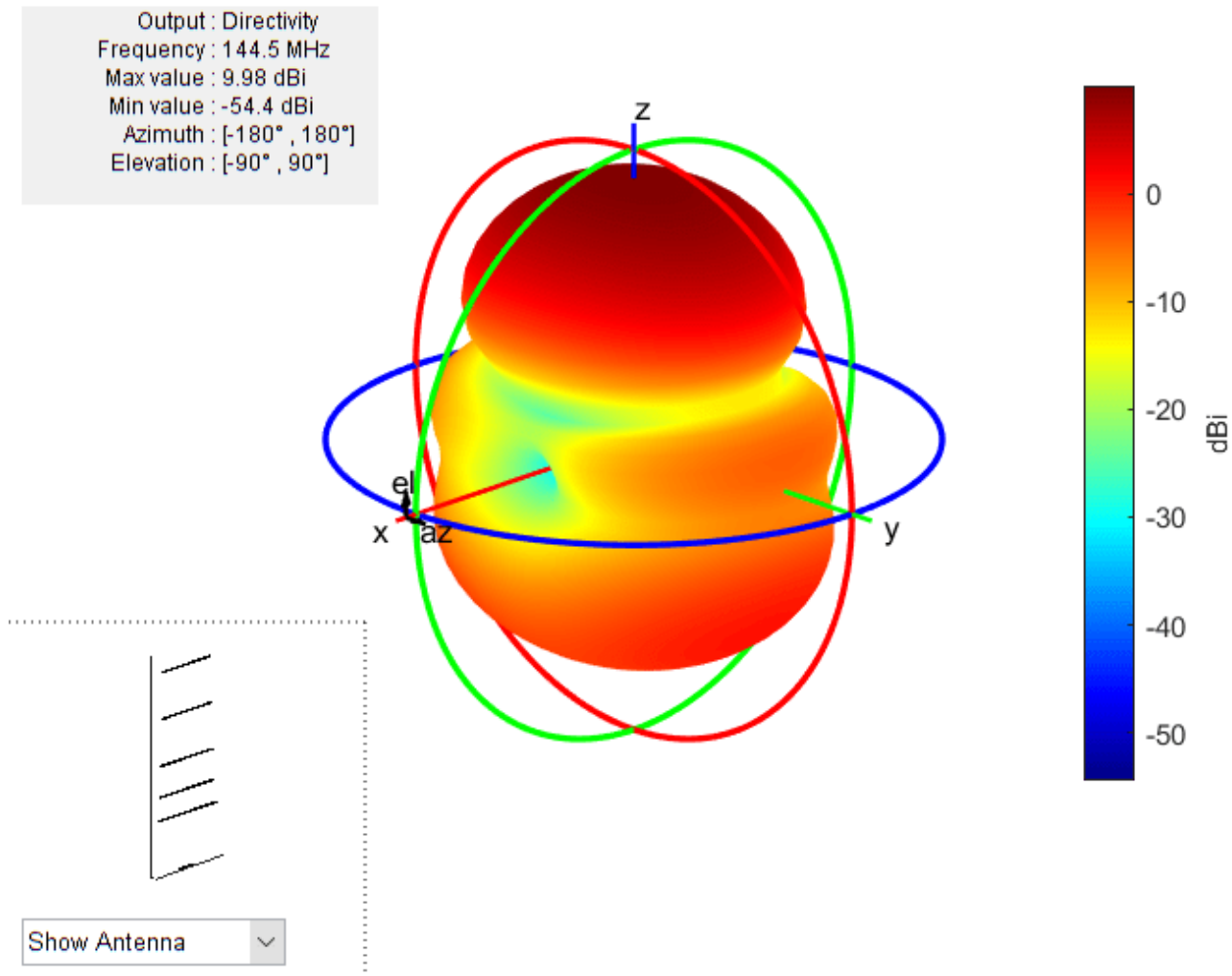
```
yagidesign.DirectorSpacing = dirSpacing*lambda;  
show(yagidesign)
```



Plot Radiation Pattern at Design Frequency

Prior to executing the optimization process, plot the radiation pattern for the initial guess in 3D.

```
pattern(yagidesign,fc);
```



This initial Yagi-Uda antenna does not have a higher directivity in the preferred direction, meaning at zenith (elevation = 90 deg) and is therefore a poorly designed radiator.

Set Up Optimization

Use the following variables as control variables for the optimization:

- Reflector length (1 variable)
- Reflector spacing (1 variable)
- Director spacings (4 variables)

In terms of a single vector parameter `controlVals`, set

- Reflector length = `controlVals(1)`
- Reflector spacing = `controlVals(2)`
- Director spacings = `controlVals(3:6)`

In terms of `controlVals`, set an objective function that aims to have a large directivity value in the 90 degree direction, a small value in the -90 degree direction, and a large value of maximum power

between the elevation beamwidth angle bounds. In addition to the directivity goal an impedance match condition is also included as a constraint. Any constraint violations will penalize the objective.

type `yagi_objective_function_surrogate.m`

```
function objectivevalue = yagi_objective_function_surrogate(y,controlVals,fc,BW,ang,Z0,constraints)
% YAGI_OBJECTIVE_FUNCTION returns the objective for a 6 element Yagi
% OBJECTIVE_VALUE =
% YAGI_OBJECTIVE_FUNCTION(Y,CONTROLVALS,FREQ,ANG,Z0,constraints), assigns
% the appropriate parasitic dimensions, CONTROLVALS to the Yagi antenna Y,
% and uses the frequency FREQ, angle pair,ANG, reference impedance Z0 and
% the constraints to calculate the objective function value.

% The YAGI_OBJECTIVE_FUNCTION function is used for an internal example.
% Its behavior may change in subsequent releases, so it should not be
% relied upon for programming purposes.

% Copyright 2018 The MathWorks, Inc.

y.ReflectorLength = controlVals(1);
y.ReflectorSpacing = controlVals(2);
y.DirectorSpacing = controlVals(3:end);

% Unpack constraints
Gmin = constraints.Gmin;
Gdev = constraints.Gdeviation;
FBmin = constraints.FBmin;
S11min = constraints.S11min;
K = constraints.Penalty;

% Calculate antenna port and field parameters
output = analyzeAntenna(y,fc,BW,ang,Z0);

% Form objective function
output1 = output.MaxDirectivity+output.MismatchLoss;      % Directivity/Gain at zenith
Gain1 = output.MaxDirectivity1+output.MismatchLoss1;      % Directivity/Gain at zenith
Gain2 = output.MaxDirectivity2+output.MismatchLoss2;      % Directivity/Gain at zenith
% Gain constraint, e.g. G > 10
c1 = 0;
if output1<Gmin
    c1 = abs(Gmin-output1)+abs(Gain1-Gmin)+abs(Gain2-Gmin);
end

% Gain deviation constraint, abs(G-Gmin)<0.1;
c1_dev = 0;
c1_dev_temp = 0;
c1_dev_temp1 = 0;
c1_dev_temp2 = 0;
if abs(output1-Gmin)>Gdev
    c1_dev_temp = -Gdev + abs(output1-Gmin);
end
if abs(Gain1-Gmin)>Gdev
    c1_dev_temp1 = -Gdev + abs(Gain1-Gmin);
end
if abs(Gain2-Gmin)>Gdev
    c1_dev_temp2 = -Gdev + abs(Gain2-Gmin);
end
```

```

c1_dev = (c1_dev_temp+c1_dev_temp1+c1_dev_temp2)/3;

% Front to Back Ratio constraint, e.g. F/B > 15
c2 = 0;
% if output.FB < FBmin
%     c2 = FBmin-output.FB;
% end
c2 = (abs(FBmin-output.FB)+abs(FBmin-output.FB1)+abs(FBmin-output.FB2))/3;

% Reflection Coefficient, S11 < -10
c3 = 0;
if output.S11 > S11min
    c3 = -S11min + output.S11;
end

% Form the objective + constraints
objectivevalue = -output1 + max(0, (c1+c1_dev+c2+c3))*K;
end

function output = analyzeAntenna(ant,fc,BW,ang,Z0)
%ANALYZEANTENNA calculate the objective function
% OUTPUT = ANALYZEANTENNA(Y,FREQ,BW,ANG,Z0) performs analysis on the
% antenna ANT at the frequency, FC, and calculates the directivity at the
% angles specified by ANG and the front-to-back ratio. The reflection
% coefficient relative to reference impedance Z0, and impedance are
% computed over the bandwidth BW around FC.

fmin = fc - (BW/2);
fmax = fc + (BW/2);
Nf = 5;
freq = unique([fc,linspace(fmin,fmax,Nf)]);
fcIdx = freq==fc;
fcIdx1 = freq==fmin;
fcIdx2 = freq==fmax;
s = sparameters(ant,freq,Z0);
Z = impedance(ant,fc);
Z1 = impedance(ant,fmin);
Z2 = impedance(ant,fmax);
az = ang(1,:);
el = ang(2,:);
Dmax = pattern(ant,fc,az(1),el(1));
Dmax1 = pattern(ant,fmin,az(1),el(1));
Dmax2 = pattern(ant,fmax,az(1),el(1));
Dback = pattern(ant,fc,az(2),el(2));
Dback1 = pattern(ant,fmin,az(2),el(2));
Dback2 = pattern(ant,fmax,az(2),el(2));
% Calculate F/B
F_by_B = Dmax-Dback;
F_by_B1 = (Dmax1-Dback1);
F_by_B2 = (Dmax2-Dback2);
% Compute S11 and mismatch loss
s11 = rfparam(s,1,1);
S11 = max(20*log10(abs(s11)));
T = mean(10*log10(1 - (abs(s11)).^2));
T1 = max(10*log10(1 - (abs(s11(fcIdx1))).^2));
T2 = max(10*log10(1 - (abs(s11(fcIdx2))).^2));
% Form the output structure
output.MaxDirectivity= Dmax;

```

```

output.BackLobeLevel = Dback;
output.MaxDirectivity1= Dmax1;
output.BackLobeLevel1 = Dback1;
output.MaxDirectivity2= Dmax2;
output.BackLobeLevel2 = Dback2;
output.FB = F_by_B;
output.FB1 = F_by_B1;
output.FB2 = F_by_B2;
output.S11 = S11;
output.MismatchLoss = T;
output.MismatchLoss1 = T1;
output.MismatchLoss2 = T2;
output.Z = Z;
output.Z1 = Z1;
output.Z2 = Z2;
end

```

Set bounds on the control variables.

```

refLengthBounds = [0.1; % lower bound on reflector length
                   0.6]; % upper bound on reflector spacing
dirLengthBounds = [0.3 0.3 0.3 0.3; % lower bound on director length
                   0.7 0.7 0.7 0.7]; % upper bound on director length
refSpacingBounds = [0.25; % lower bound on reflector spacing
                    0.65]; % upper bound on reflector spacing
dirSpacingBounds = [0.01 0.1 0.1 0.1; % lower bound on director spacing 0.2 0.25 0.3 0.3
                    0.2 0.25 0.35 0.35]; % upper bound on director spacing
exciterLengthBounds = [0.45; % lower bound on exciter length
                       0.6]; % upper bound on exciter length
exciterSpacingBounds = [.004;
                        .008];
LB = [refLengthBounds(1) refSpacingBounds(1) dirSpacingBounds(1,:) ].*lambda;
UB = [refLengthBounds(2) refSpacingBounds(2) dirSpacingBounds(2,:) ].*lambda;
parameterBounds.LB = LB;
parameterBounds.UB = UB;
ang = [0 0;90 -90]; % azimuth,elevation angles for main lobe and back

```

Surrogate Based Optimization

The Global Optimization Toolbox™ provides a surrogate based optimization function called `surrogate`. We use this function with options specified with the `optimoptions` function. At every iteration, plot the best value of the objective function and limit the total number of iterations to 300. Pass the objective function to the surrogate function by using an anonymous function together with the bounds and the options structure. The objective function used during the optimization process by `surrogate` is available in the file `yagi_objective_function`.

```

% Optimizer options
optimizer = 'Surrogate';

if strcmpi(optimizer,'PatternSearch')
    optimizerparams = optimoptions(@patternsearch);
    optimizerparams.UseCompletePoll = true;
    optimizerparams.PlotFcns = @psplotbestf;
    optimizerparams.UseParallel = true;
    optimizerparams.Cache = 'on';
    optimizerparams.MaxFunctionEvaluations = 1200;
    optimizerparams.FunctionTolerance = 1e-2;
elseif strcmpi(optimizer,'Surrogate')

```

```
optimizerparams = optimoptions(@surrogateopt);
optimizerparams.UseParallel = true;
optimizerparams.MaxFunctionEvaluations = 600;
optimizerparams.MinSurrogatePoints = 12;
optimizerparams.InitialPoints = initialdesign;
else
    error('Optimizer not supported');
end

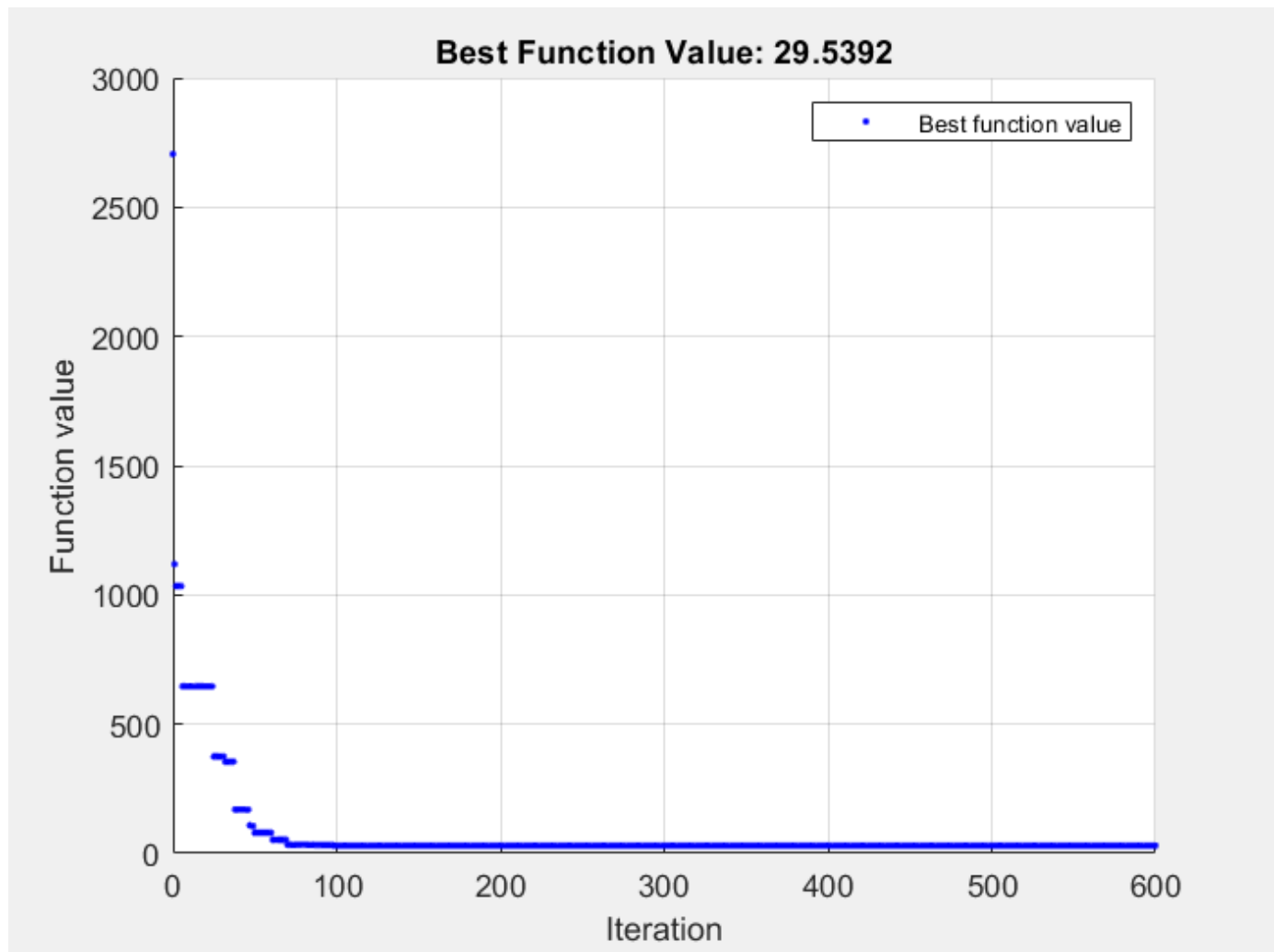
% Antenna design parameters
designparams.Antenna = yagidesign;
designparams.Bounds = parameterBounds;

% Analysis parameters
analysisparams.CenterFrequency = fc;
analysisparams.Bandwidth = BW;
analysisparams.ReferenceImpedance = Z0;
analysisparams.MainLobeDirection = ang(:,1);
analysisparams.BackLobeDirection = ang(:,2);

% Set constraints
constraints.S11min = -15;
constraints.Gmin = 10;
constraints.Gdeviation = 0.1;
constraints.FBmin = 20;
constraints.Penalty = 75;
poolobj = gcp;

Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 6).

optimdesign = optimizeAntennaSurrogate(designparams,analysisparams,constraints,optimizerparams);
```

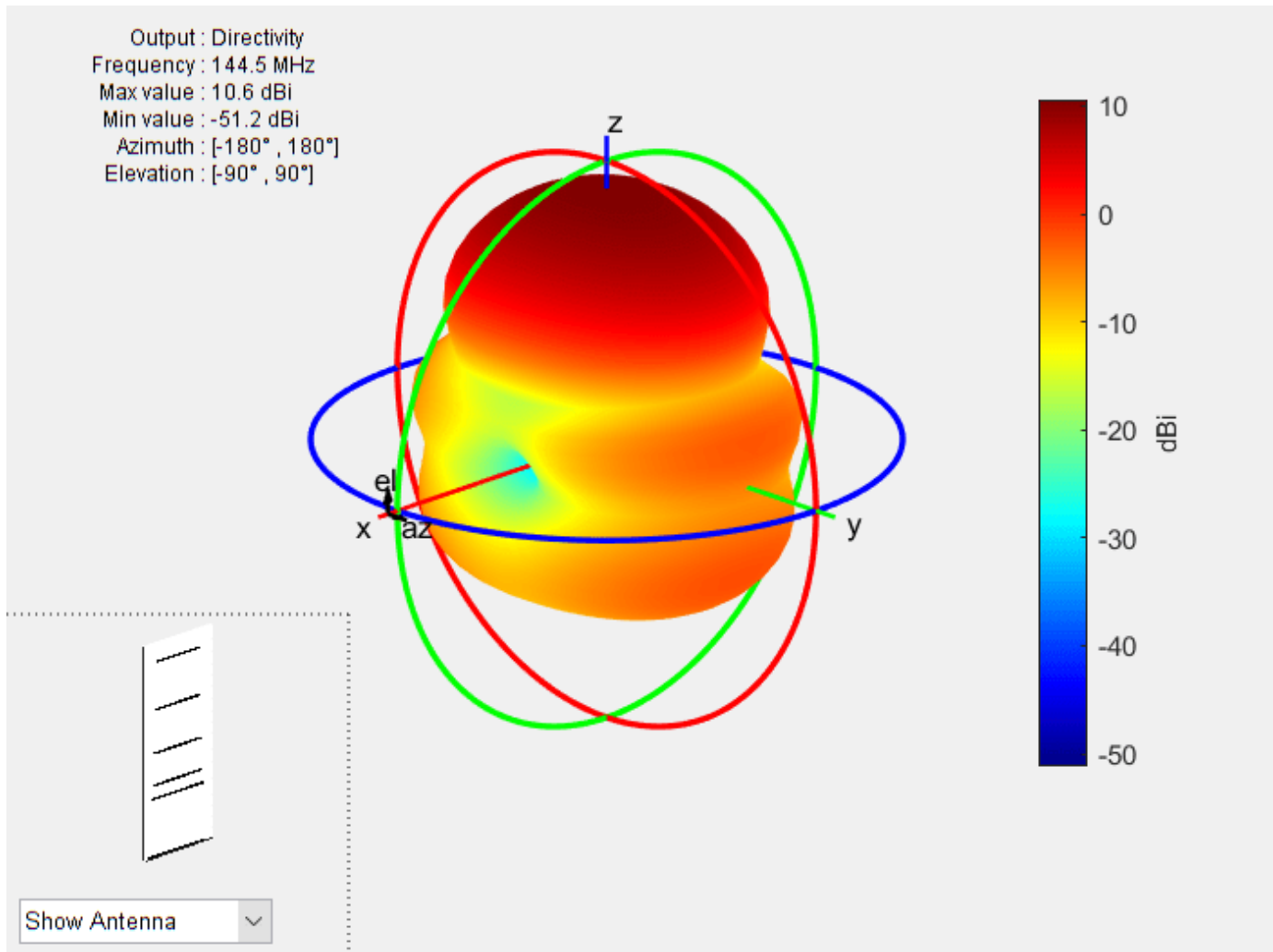


surrogateopt stopped because it exceeded the function evaluation limit set by 'options.MaxFunctionEvaluations'.

Plot Optimized Pattern

Plot the optimized antenna pattern at the design frequency.

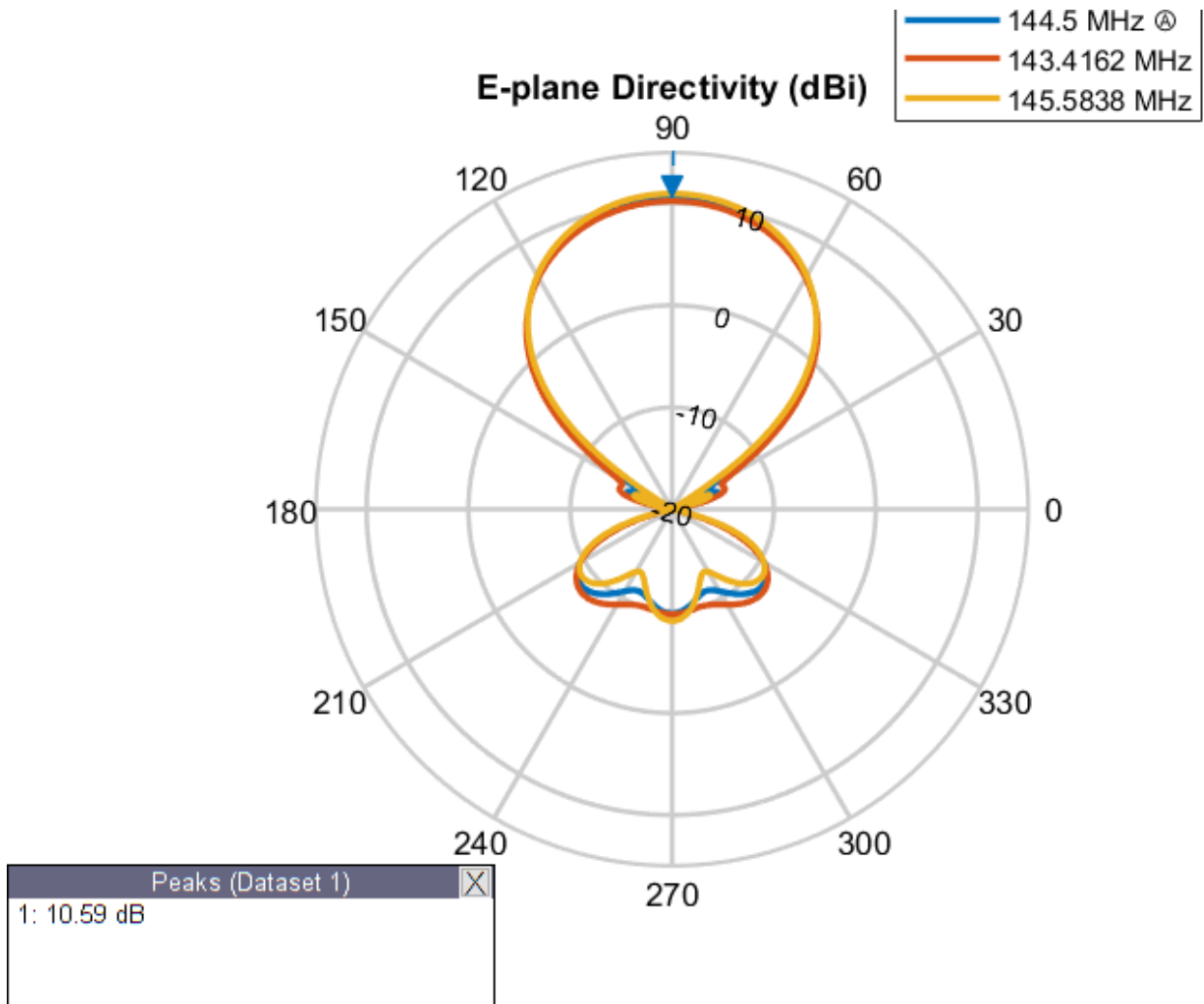
```
yagidesign.ReflectorLength = optimdesign(1);  
yagidesign.ReflectorSpacing = optimdesign(2);  
yagidesign.DirectorSpacing = optimdesign(3:6);  
pattern(yagidesign,fc)
```

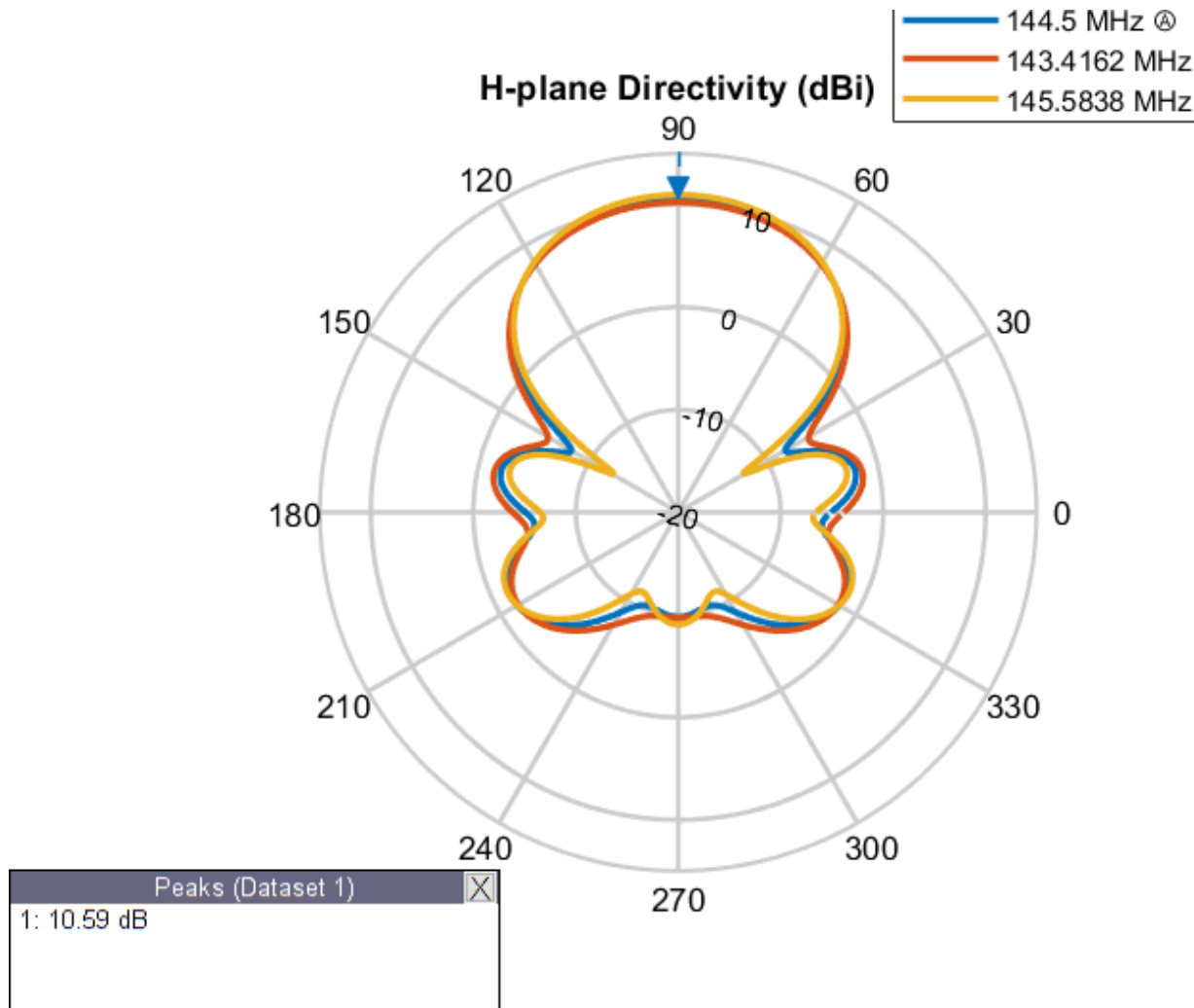
E and H-Plane Cuts of Pattern

To obtain a better insight into the behavior in the two orthogonal planes, plot the normalized magnitude of the electric field in the E and H-planes, i.e. azimuth = 0 and 90 deg respectively. Enable the antenna metrics on the polar pattern plots to establish the directivity at zenith, Front-to-Back ratio, and the beamwidth in E and H-planes.

```
fU = fc+ BW/2;
fL = fc-BW/2;
figure;
patternElevation(yagidesign,fc,0,'Elevation',0:1:359);
pE = polarpattern('gco');
DE_fL = patternElevation(yagidesign,fL,0,'Elevation',0:1:359);
DE_fU = patternElevation(yagidesign,fU,0,'Elevation',0:1:359);
add(pE,[DE_fL, DE_fU])
pE.MagnitudeLim = [-20 15];
pE.TitleTop = 'E-plane Directivity (dBi)';
pE.LegendLabels = {[num2str(fc./1e6), ' MHz'],[num2str(fL./1e6), ' MHz'],[num2str(fU./1e6), ' MHz']}
```



```
figure;
patternElevation(yagidesign,fc,90,'Elevation',0:1:359);
pH = polarpattern('gco');
DH_fL = patternElevation(yagidesign,fL,90,'Elevation',0:1:359);
DH_fU = patternElevation(yagidesign,fU,90,'Elevation',0:1:359);
add(pH,[DH_fL, DH_fU])
pH.MagnitudeLim = [-20 15];
pH.TitleTop = 'H-plane Directivity (dBi)';
pH.LegendLabels = {[num2str(fc./1e6), ' MHz'],[num2str(fL./1e6), ' MHz'],[num2str(fU./1e6), ' MHz']}
```

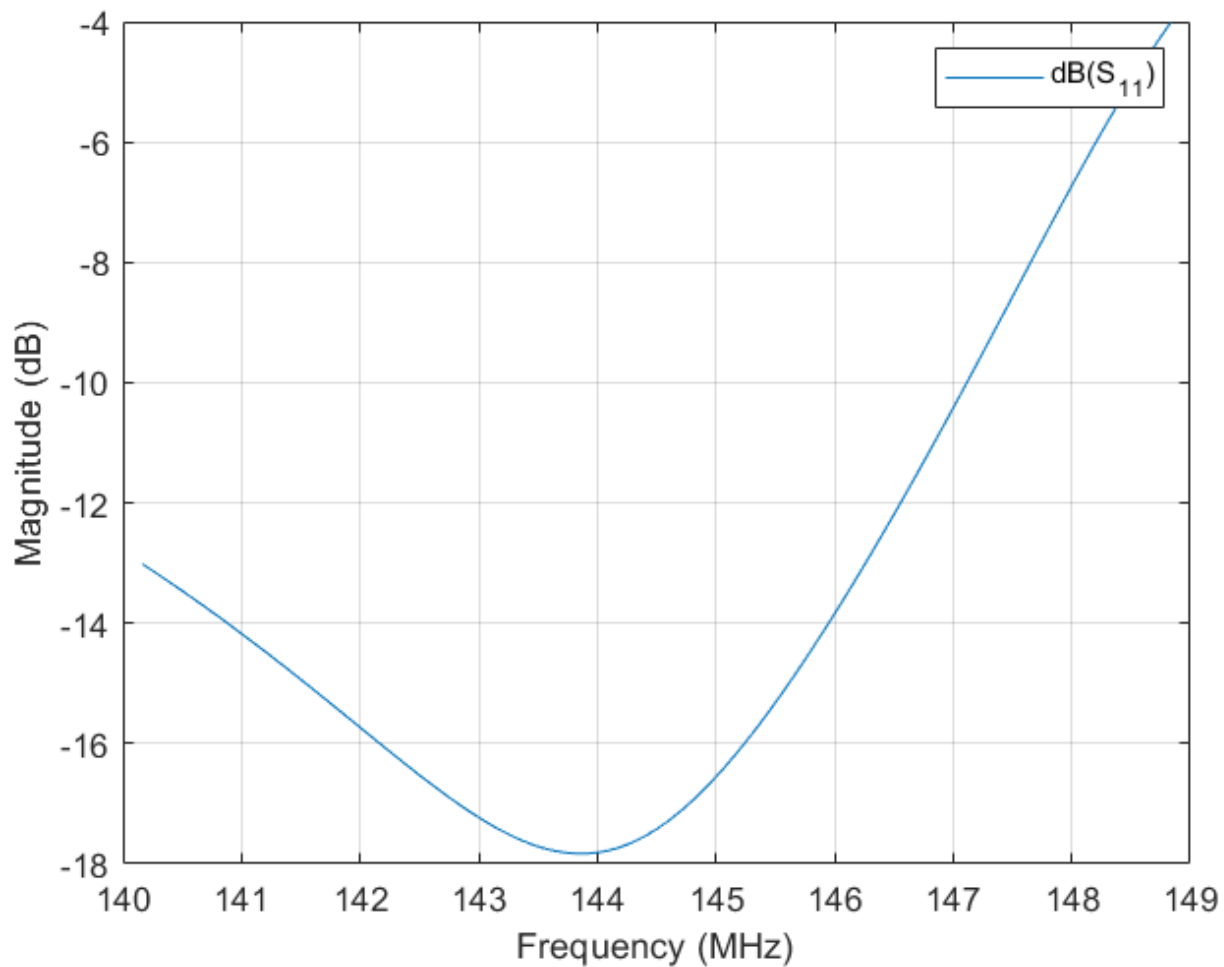


The optimized design shows a significant improvement in the radiation pattern. There is higher directivity achieved in the desired direction toward zenith. The back lobe is small, resulting in a good front to back ratio for this antenna.

Input Reflection Coefficient of Optimized Antenna

The input reflection coefficient for the optimized Yagi-Uda antenna is computed and plotted relative to the reference impedance of 50Ω . A value of -10 dB or lower is considered as a good impedance match.

```
s = sparameters(yagidesign, freq, Z0);
figure;
rfplot(s);
```



Tabulating Initial and Optimized Design

Tabulate the initial design guesses and the final optimized design values.

```
yagiparam= {'Reflector Length';'Reflector Spacing';'Director Spacing - 1';
            'Director Spacing - 2';'Director Spacing - 3';
            'Director Spacing - 4'};
initialdesign = initialdesign';
optimdesign = optimdesign';
Tgeometry = table(initialdesign,optimdesign,'RowNames',yagiparam)
```

Tgeometry=6×2 table

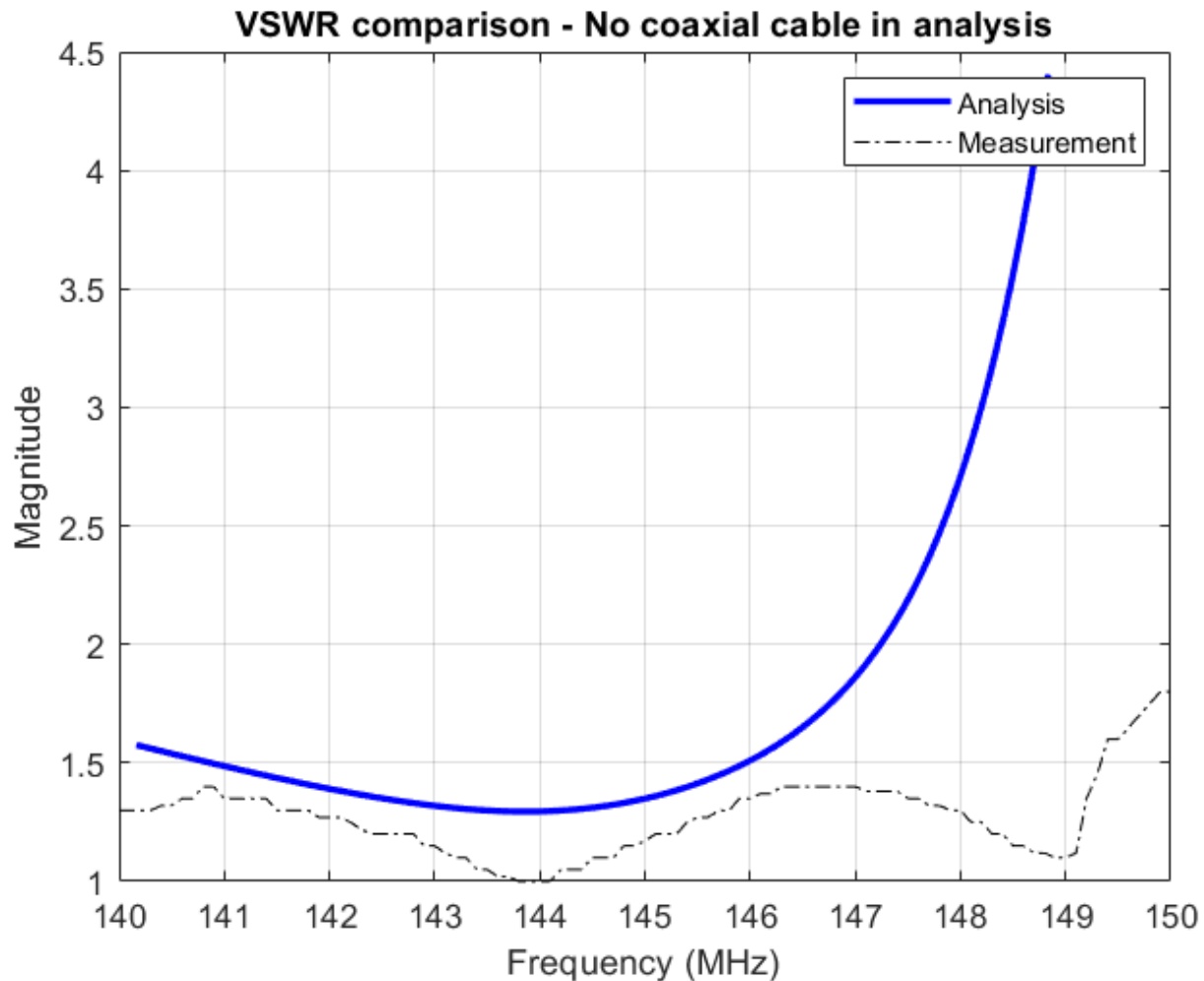
	initialdesign	optimdesign
Reflector Length	0.51867	1.1327
Reflector Spacing	0.72614	0.87689
Director Spacing - 1	0.3112	0.20547
Director Spacing - 2	0.41494	0.4871
Director Spacing - 3	0.62241	0.66081

Director Spacing - 4 0.62241 0.72399

Fabricated Antennas

The optimized yagi design was fabricated. The real yagi needs a supporting element along the longitudinal axis to impart mechanical rigidity. This supporting element is called a boom and is usually manufactured out of a non-metallic material. In this case pVC tubing was used to make the boom. Note that the effect of this boom has not been modeled in the Antenna Toolbox yagiUda element. Another technique of analyzing the input match is to calculate and plot the VSWR (Voltage Standing Wave Ratio). Calculate and plot the predicted VSWR from the optimized design. The fabricated antennas VSWR was measured as well using an SWR meter. This data was saved in CSV file. Overlay the measured results with the analysis.

```
vswr_measured = csvread('SWR_Values_Sep_15.csv',1,0);  
figure  
vswr(yagidesign,freq,Z0)  
hold on  
plot(vswr_measured(:,1),vswr_measured(:,2),'k-.')  
legend('Analysis','Measurement')  
title('VSWR comparison - No coaxial cable in analysis')  
ylabel('Magnitude')
```



Model the Effect of Coaxial Cable

The coaxial cable connected to the fabricated yagi antenna is a RG-58/U with a characteristic impedance of 50Ω . Create a model of this coaxial cable using RF Toolbox.

```

out_radius = 3.51e-3;
in_radius = 0.91e-3;
eps_r = 2.95;
line_length = 5.05*lambda;
coax_cable = rfckt.coaxial;
coax_cable.OuterRadius = out_radius;
coax_cable.InnerRadius = in_radius;
coax_cable.EpsilonR = eps_r;
coax_cable.LossTangent = 2e-4;
coax_cable.LineLength = line_length;

```

Analyze the coaxial cable at the range of frequencies intended for operation and use the yagi's impedance as the load. Compute the input VSWR for the coaxial cable and yagi antenna.

```

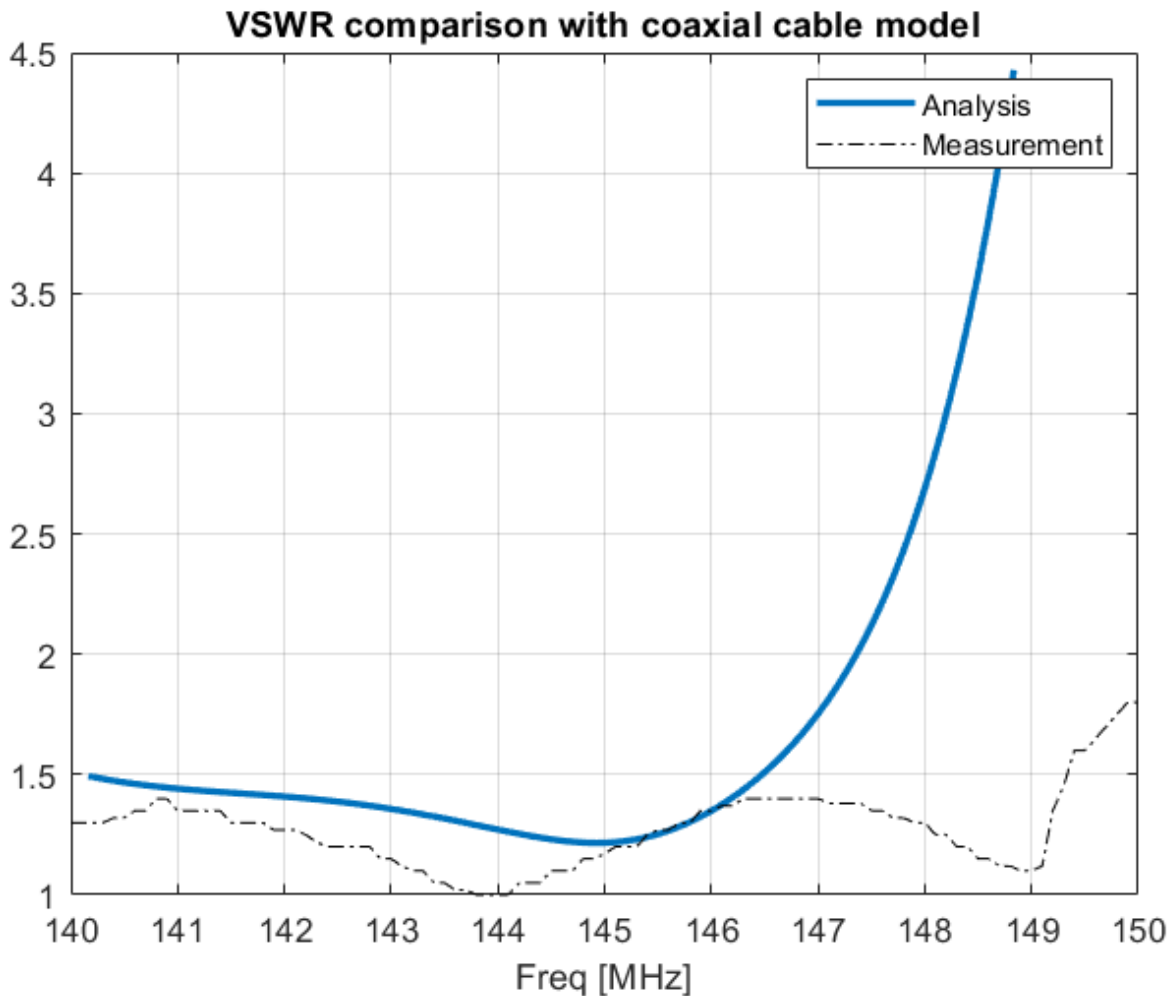
Zyagi =impedance(yagidesign, freq);
analyze(coax_cable, freq, Zyagi);

```

```

figure
hline = plot(coax_cable, 'VSWRin', 'None');
hline.LineWidth = 2;
hold on
plot(vswr_measured(:,1),vswr_measured(:,2),'k-.')
legend('Analysis','Measurement')
title('VSWR comparison with coaxial cable model')

```



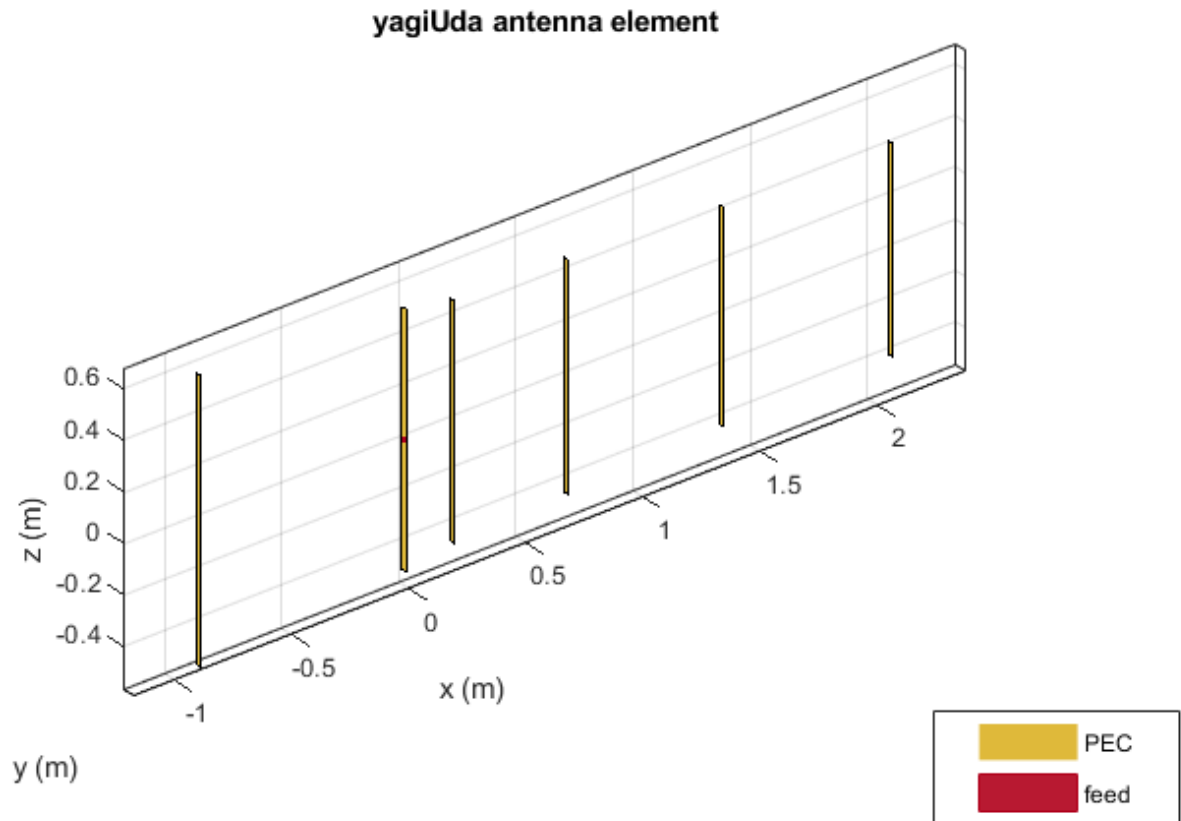
The analysis VSWR curve for the coax and yagi antenna combination compares favorably with the measured data.

The results of the optimized design compare favorably with the fabricated antenna. This antenna will be used as part of a repeater station operating at 145 MHz.

```

figure
yagidesign.Tilt = 90;
yagidesign.TiltAxis = [0 1 0];
show(yagidesign)

```





%%

See Also

“Direct Search Based Optimization of Six-Element Yagi-Uda Antenna” on page 5-147

Model and Analyze Dual Polarized Patch Microstrip Antenna

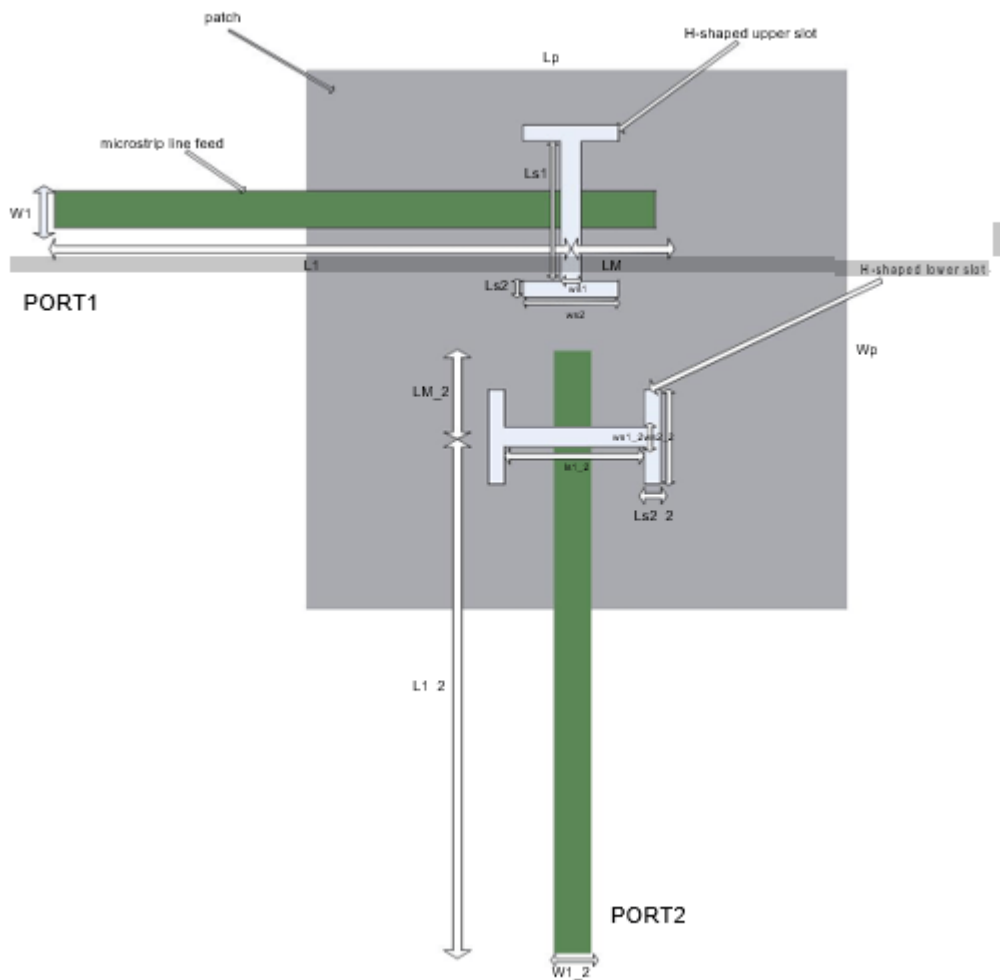
This example shows how to design and measure a wideband dual polarized microstrip antenna that finds its use at the base station of a cellular system. In order to achieve the wideband characteristic, this design considers a slot coupled patch antenna structure.

Building Aperture Coupled Antenna

Define Parameters

Parameters given below defines the offsets for upper and lower slots and the stublengths.

```
of_1=12e-3;
of_2=-6e-3;
LM=7e-3;
LM_2=7e-3;
```



taken from reference page.no:55

Define Patch

In this antenna, the radiating element is patchMicrostrip. Above the patch, a dielectric substrate with EpsilonR of 3.38, acts like a radome. Below the patch there is a foam material of EpsilonR 1.025.

```
Lp=50e-3;
patch=antenna.Rectangle('Length',Lp,'Width',Lp,'Center',[0 0]);
```

Define H-shaped Slots

The double slots perform the dual polarized operation. Each slot is H-shaped and positioned on the ground plane in "T" formation. This formation provides a good isolation level among port1 and port2.

Define Upper H-shape Slot

```
Ls1=12e-3;
Ws1=0.5e-3;
Ls2=1e-3;
Ws2=22e-3;
f1=antenna.Rectangle('Length',Ws1,'Width',Ls1,'Center',[0 of_1]);
f2=antenna.Rectangle('Length',Ws2,'Width',Ls2,'Center',[0 of_1+(Ls1/2)+(Ls2/2)]);
f3=antenna.Rectangle('Length',Ws2,'Width',Ls2,'Center',[0 of_1-(Ls1/2)-(Ls2/2)]);
f4=f1+f2+f3;
```

Define Lower H-shape Slot

```
Ls1_2=17e-3;
Ls2_2=1e-3;
Ws1_2=0.5e-3;
Ws2_2=17e-3;
f5=antenna.Rectangle('Length',Ls1_2,'Width',Ws1_2,'Center',[0 of_2]);
f6=antenna.Rectangle('Length',Ls2_2,'Width',Ws2_2,'Center',[ (Ls1_2/2)+(Ls2_2/2) of_2]);
f7=antenna.Rectangle('Length',Ls2_2,'Width',Ws2_2,'Center',[ -(Ls1_2/2)+(Ls2_2/2) of_2]);
f8=f5+f6+f7;
```

Define Ground Plane

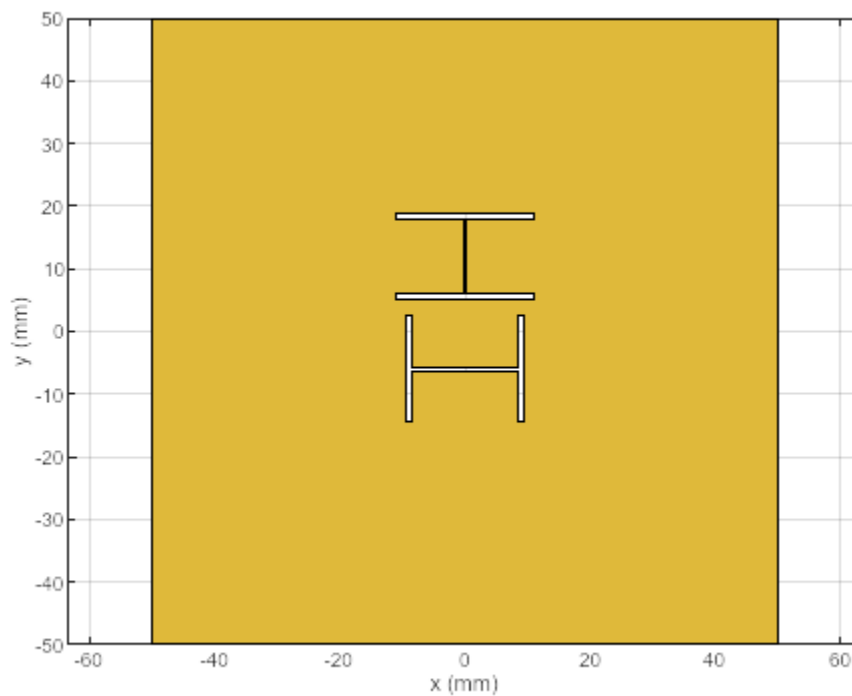
Create the ground plane shape for the antenna. The ground plane in this case is a square of size 100mm x 100mm.

```
LGp=100e-3;
Ground_plane=antenna.Rectangle('Length',LGp,'Width',LGp,'Center',[0 0]);
```

Define Slotted Ground Plane

Use the rectangle shape primitives to create the H-slots. Use the Boolean subtraction operation for slotting the Ground plane.

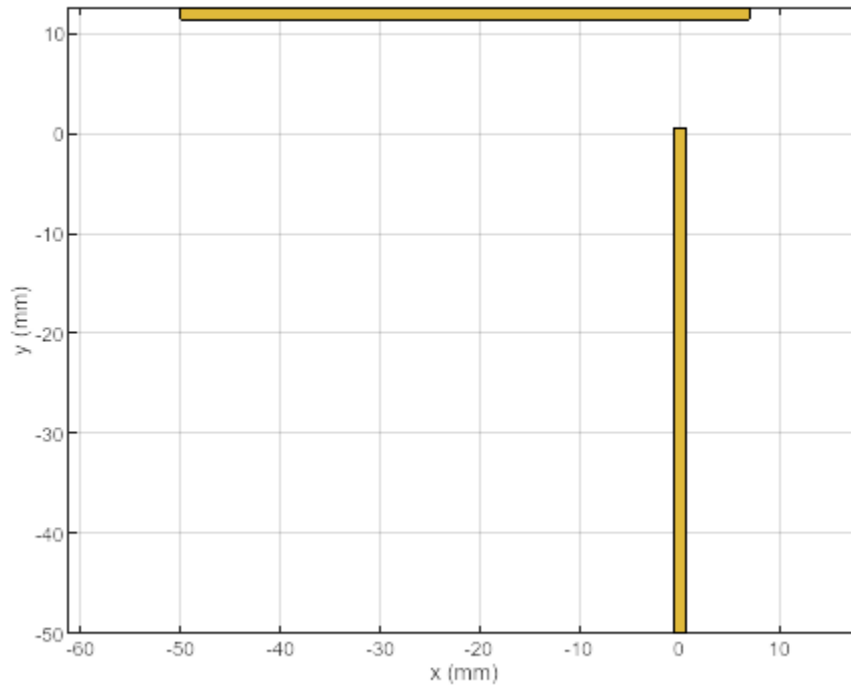
```
Gp_slot=Ground_plane-f4-f8;
figure;
show(Gp_slot);
```



Define Feed Lines

Use a feed line of size 50mm x 1.181mm for the upper H-slot. Use a feed line of size 44mm x 1.181mm for the lower H-slot. Connect the stubs at the end of the feed lines.

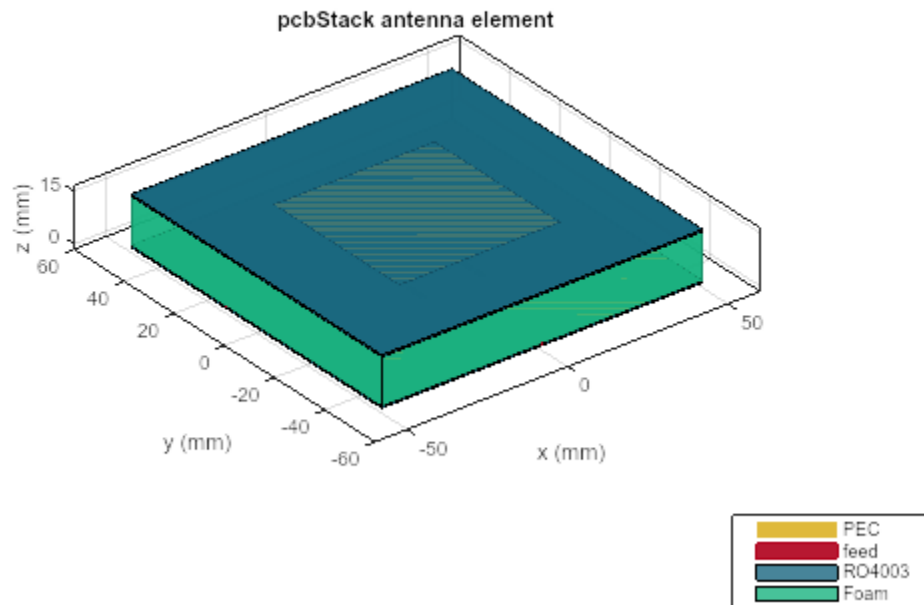
```
L1=50e-3;
W1=1.181e-3;
L1_2=44e-3;
W1_2=1.181e-3;
feed_1=antenna.Rectangle('Length',L1,'Width',W1,'Center',[-(L1/2) of_1]);
feed_2=antenna.Rectangle('Length',W1_2,'Width',L1_2,'Center',[0 -(L1_2/2)]+(of_2));
feed_1_2=feed_1+feed_2;
stub_1=antenna.Rectangle('Length',LM,'Width',W1,'Center',[(LM/2) of_1]);
stub_2=antenna.Rectangle('Length',W1,'Width',LM_2,'Center',[0 of_2/2]);
stub=stub_1+stub_2;
feed=feed_1_2+stub;
figure;
show(feed);
```



Define PCB Stack

Use the `pcbStack` to define the metal and dielectric layers and the feed for the aperture coupled patch antenna. The layers are defined top-down. In this case, the top-most layer is a dielectric layer. The second layer is a patch of square shape, and the third layer is another dielectric, followed by a fourth layer which is the ground plane. Fifth Layer is again the same dielectric which is used as first layer. Sixth layer is related to feed lines.

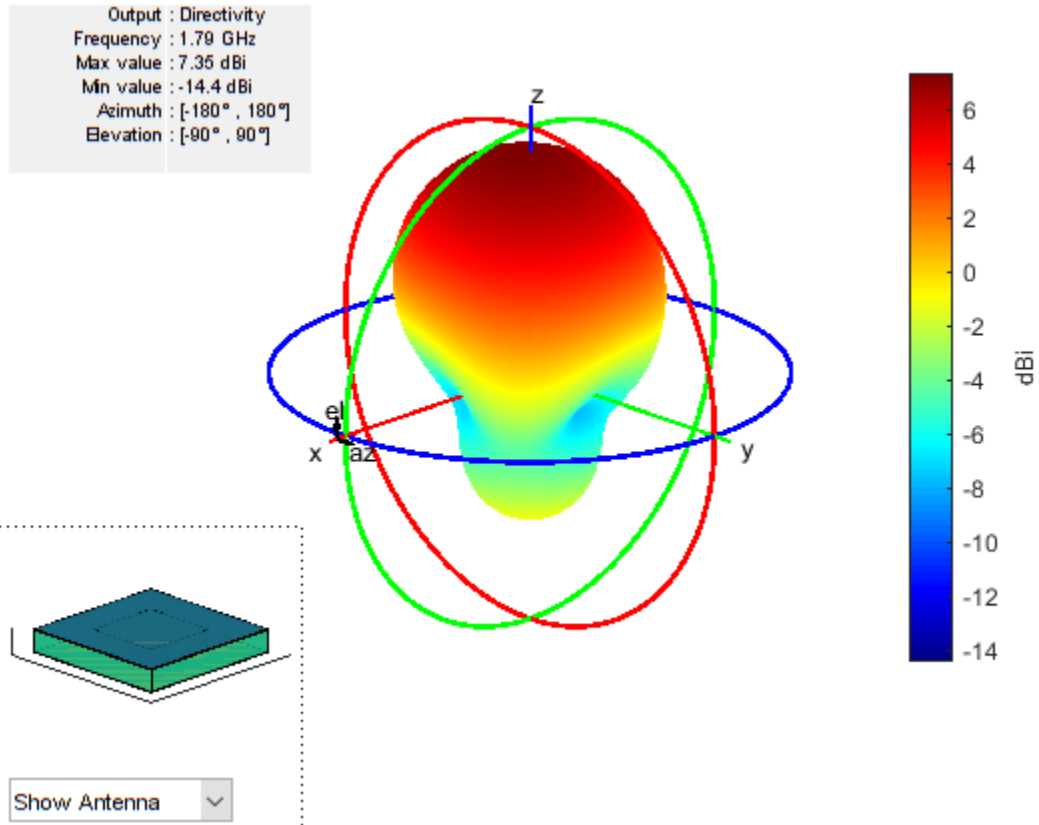
```
p = pcbStack;
d1=dielectric('EpsilonR',3.38,'Thickness',0.51e-3,'Name','R04003');
d2=dielectric('EpsilonR',1.025,'Thickness',14e-3,'Name','Foam');
p.BoardThickness=d1.Thickness+d2.Thickness+d1.Thickness;
p.BoardShape.Length=LGp;
p.BoardShape.Width=LGp;
p.Layers={d1,patch,d2,Gp_slot,d1,feed};
p.FeedLocations=[-L1 of_1 4 6;0 -L1_2+of_2 4 6];
p.FeedDiameter=feed_1.Width/3;
figure;
show(p);
```



Plot Radiation Pattern

Plot the radiation pattern of the antenna at the frequencies of best match. Use a resonant frequency of 1.79 GHz to plot the radiation pattern.

```
figure;  
pattern(p,1.79e9);
```

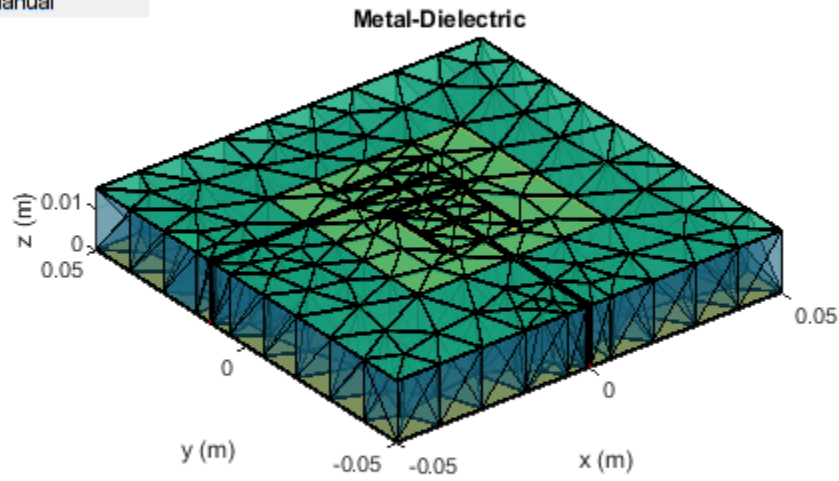


Meshing Antenna

Mesh the antenna with maximum edge length of 0.036m.

```
figure;
mesh(p, 'MaxEdgeLength', 0.036);
```

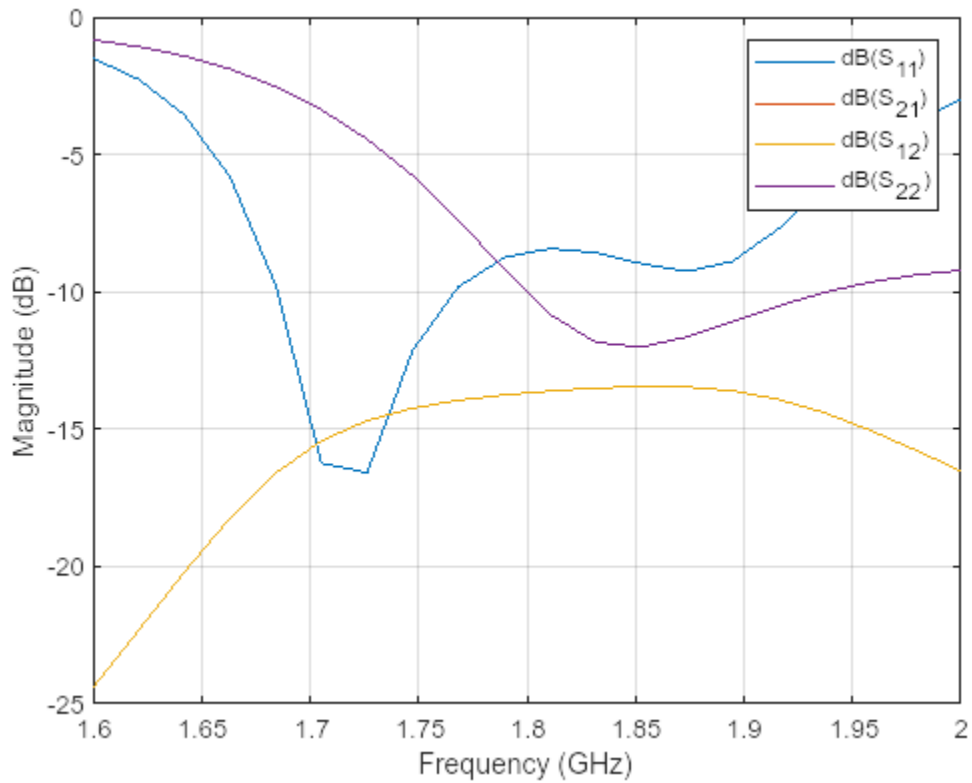
NumTriangles: 503
NumTetrahedra: 2988
NumBasis:
MaxEdgeLength: 0.036
MeshMode: manual



Calculate and Plot S-parameters

The plot shows return loss characteristics(S_{11}, S_{22}) and isolation(S_{12}) between ports.

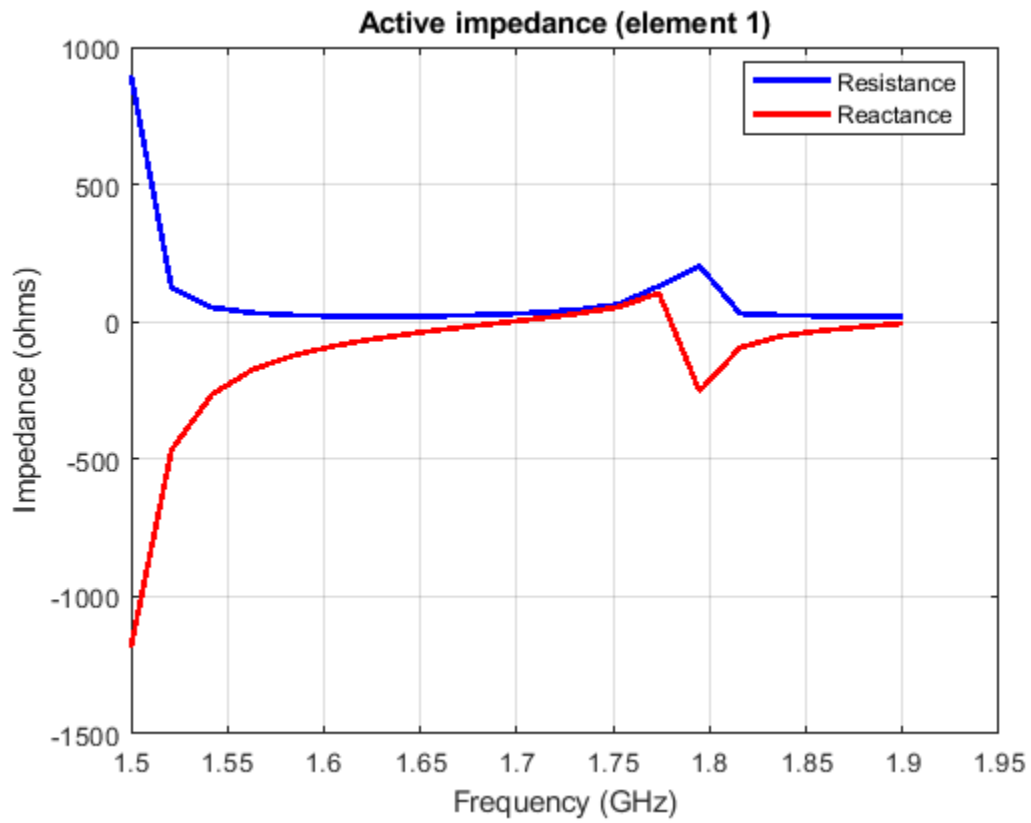
```
figure;  
sf=sparameters(p,linspace(1.6e9,2e9,20));  
rfplot(sf);
```

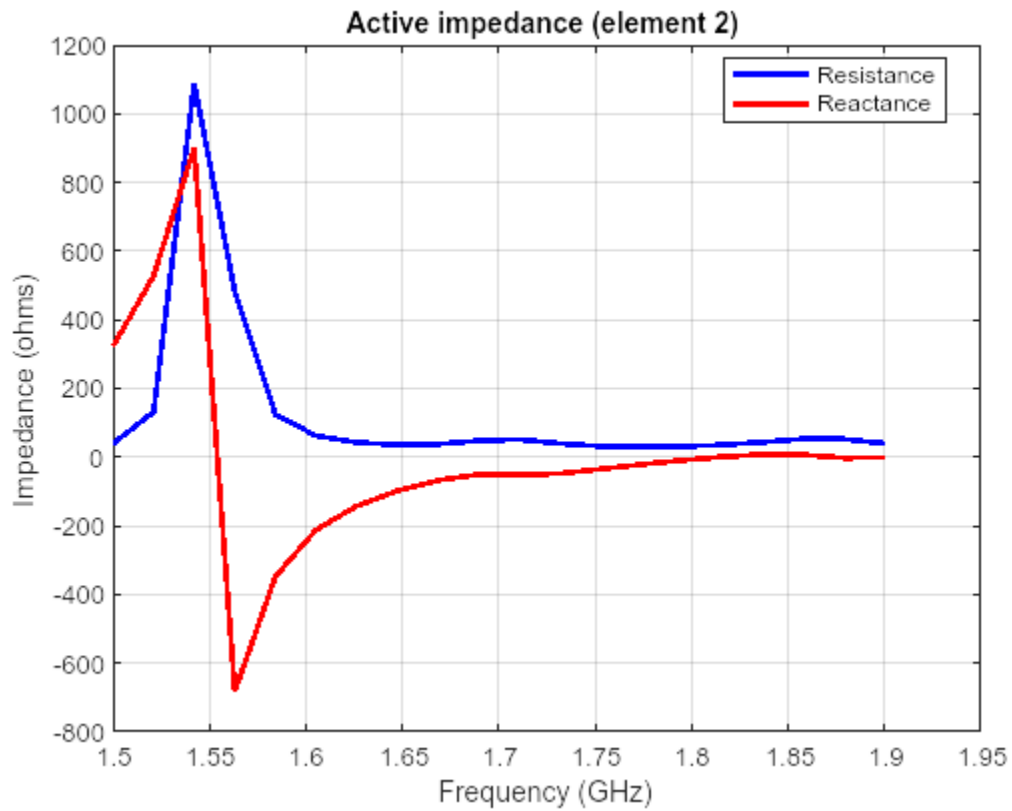
Plot Impedance Pattern

Use a frequency range from 1.5 GHz to 1.9 GHz with 20 frequency points to plot the impedance pattern.

```
figure;  
impedance(p,linspace(1.5e9,1.9e9,20),1);
```



```
figure;  
impedance(p,linspace(1.5e9,1.9e9,20),2);
```



Conclusion

The design and analysis of the dual polarized aperture coupled antenna using Antenna Toolbox agrees well with the referred results.

References

[1] Meltem Yildirim, "Design of Dual Polarized Wideband Microstrip Antennas", pp. 54-70.

See Also

"RFID Antenna Design" on page 5-236

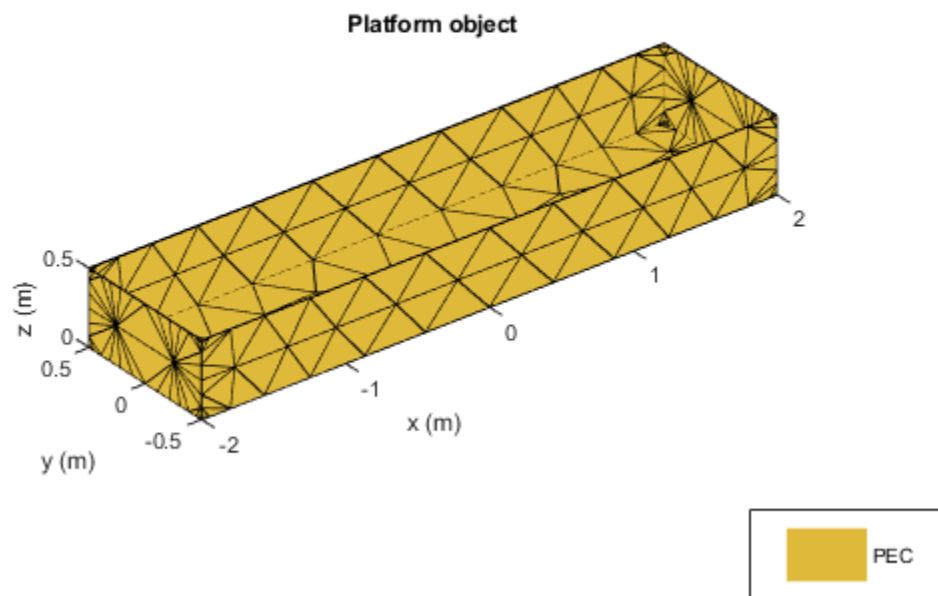
Installed Antenna Analysis - Modelling Antennas with Platforms

The example models an antenna element in presence of a large platform. The antenna is modelled accurately using the Method of Moments (MoM) while the effect of the electrically large platform is considered using physical optics (PO).

Create a platform

The electrically large platform can be imported in Antenna Toolbox as an STL file. These STL files can be used to describe ships, planes or any other kind of structures on which the antenna element is mounted. In the present case the `rectcavity.stl` file is a large rectangular cavity created using the `stlwrite` function and then deleting the dipole antenna manually. `h = cavity('Length', 4, 'Width', 1, 'Height', 0.5); z = impedance(h, 1e8); stlwrite(h, 'rectcavity.stl');`

```
base = platform('FileName', 'rectcavity.stl', 'Units', 'm');  
show(base);
```



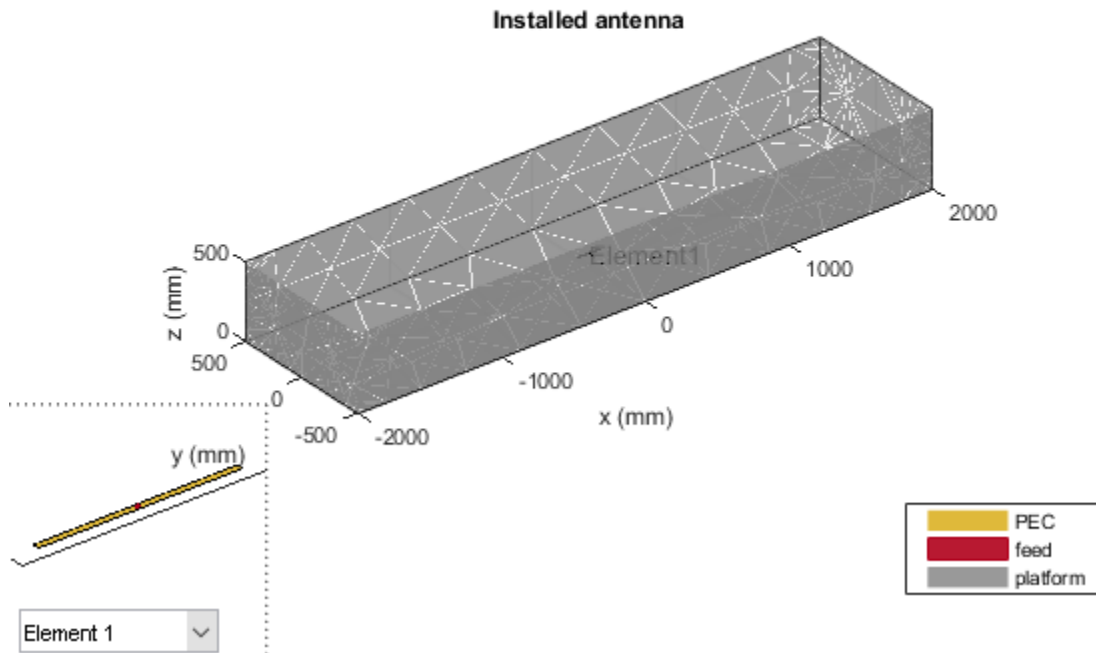
Set-up the installed antenna analysis

The user defined platform can be specified in the installed antenna analysis. The user can choose the antenna element and its location with respect to the platform.

```

ant = installedAntenna;
ant.Platform = base;
show(ant);

```



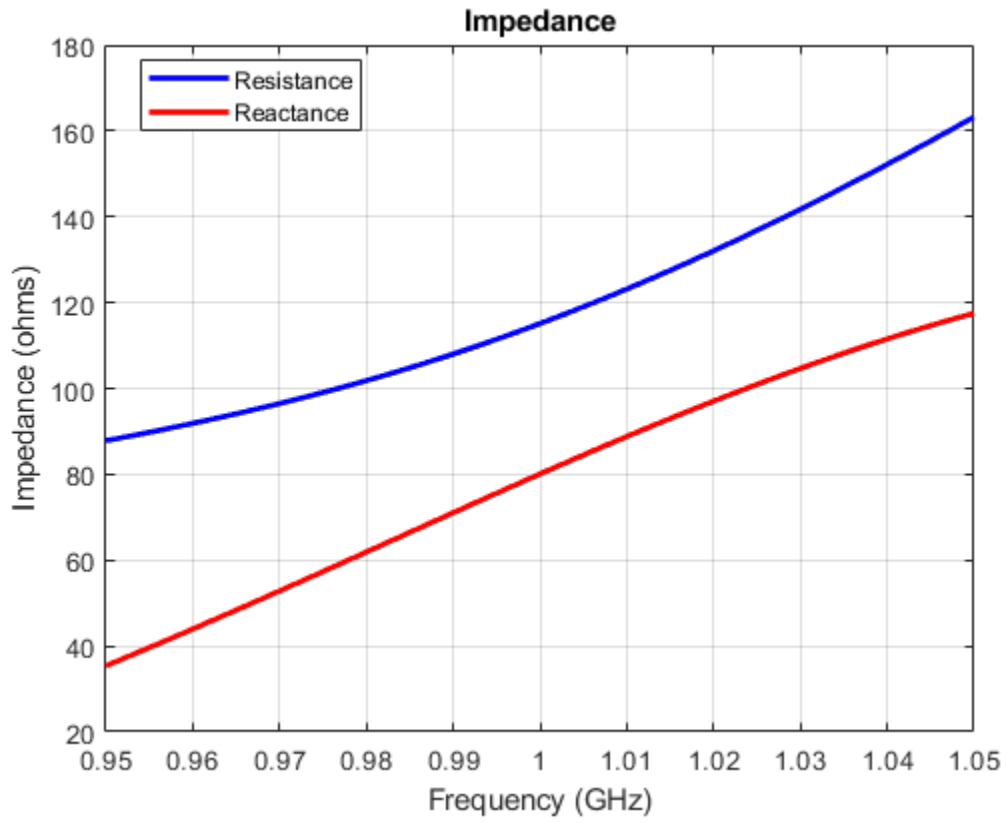
Analyze the installed antenna setup

All the analysis that can be performed on the single antenna element can be performed in case of an installed antenna.

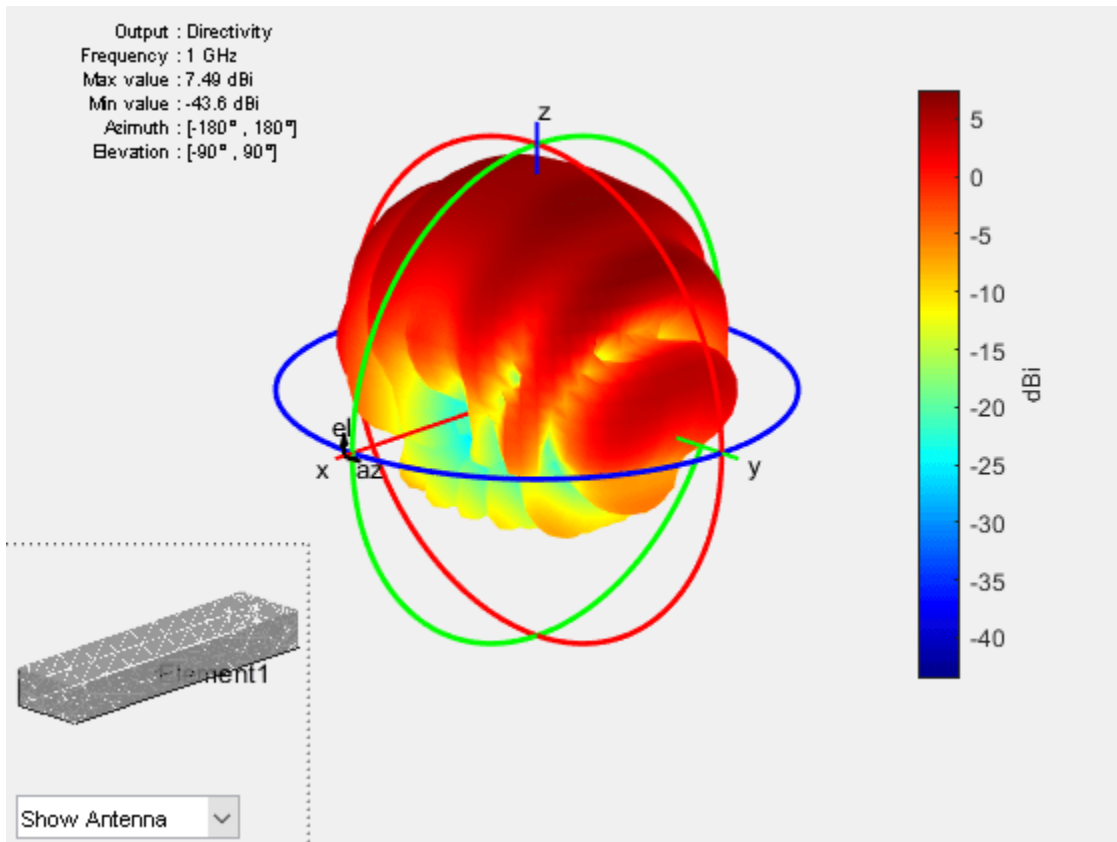
```

figure;
impedance(ant, linspace(950e6, 1050e6, 51));

```

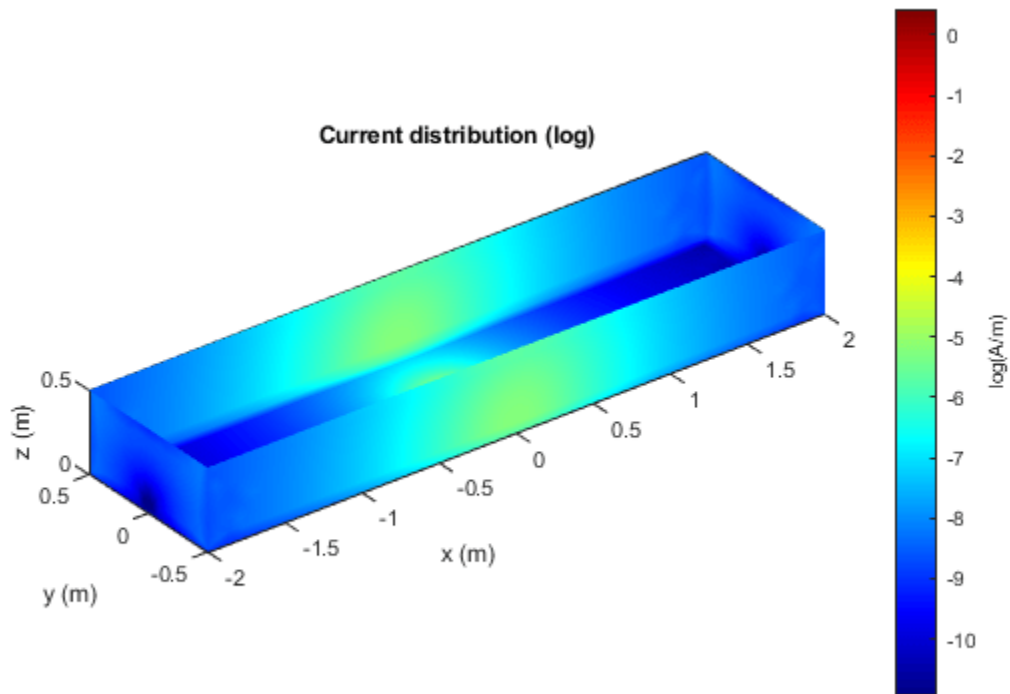


```
figure;  
pattern(ant, 1e9);
```



To better visualize the current distribution on the platform it is better to select the log scale.

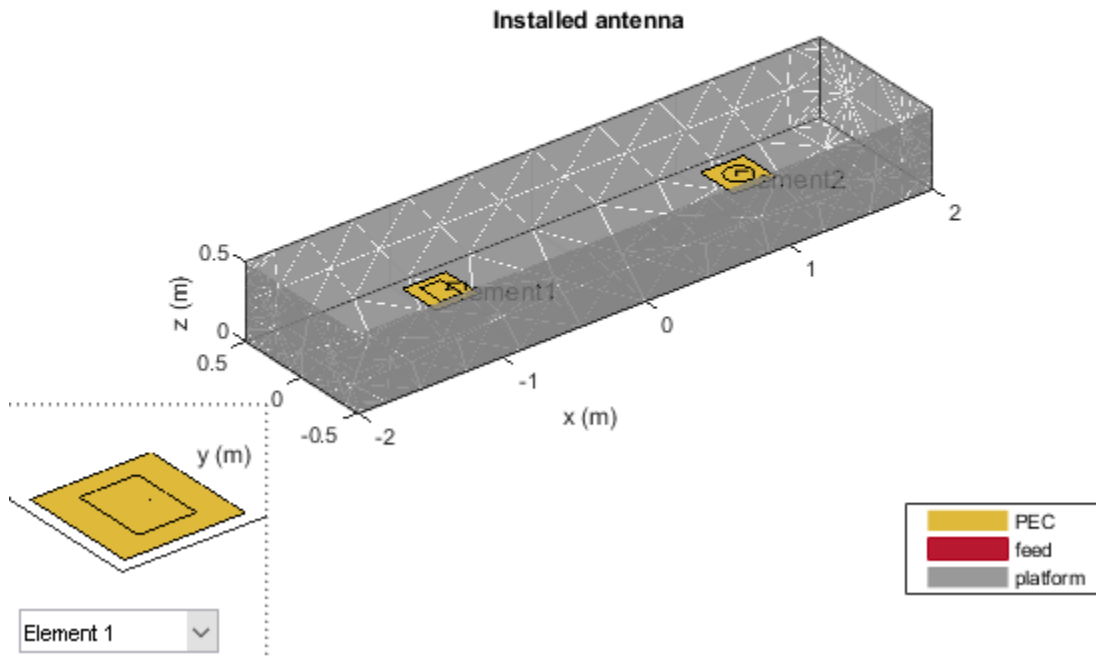
```
figure;  
current(ant, 1e9, 'scale', 'log');
```



Add multiple antennas on the platform

Installed antenna analysis can be performed with multiple antenna elements. In this case a rectangular patch microstrip antenna and a circular patch microstrip antenna are designed at 1 GHz and are placed inside the cavity structure 2 meters apart.

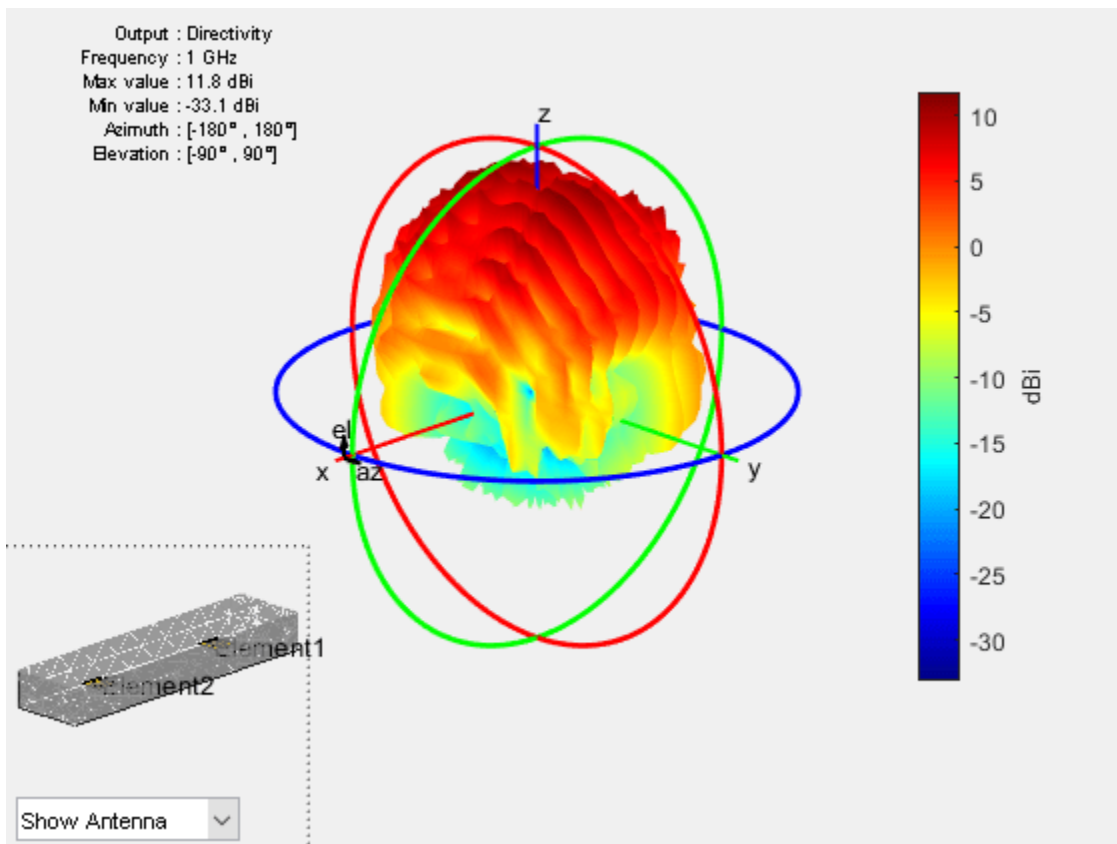
```
elem1 = design(patchMicrostrip, 1e9);  
elem2 = design(patchMicrostripCircular, 1e9);  
ant.ElementPosition = [-1 0 0.2; 1 0 0.2];  
ant.Element = {elem1, elem2};  
show(ant);
```

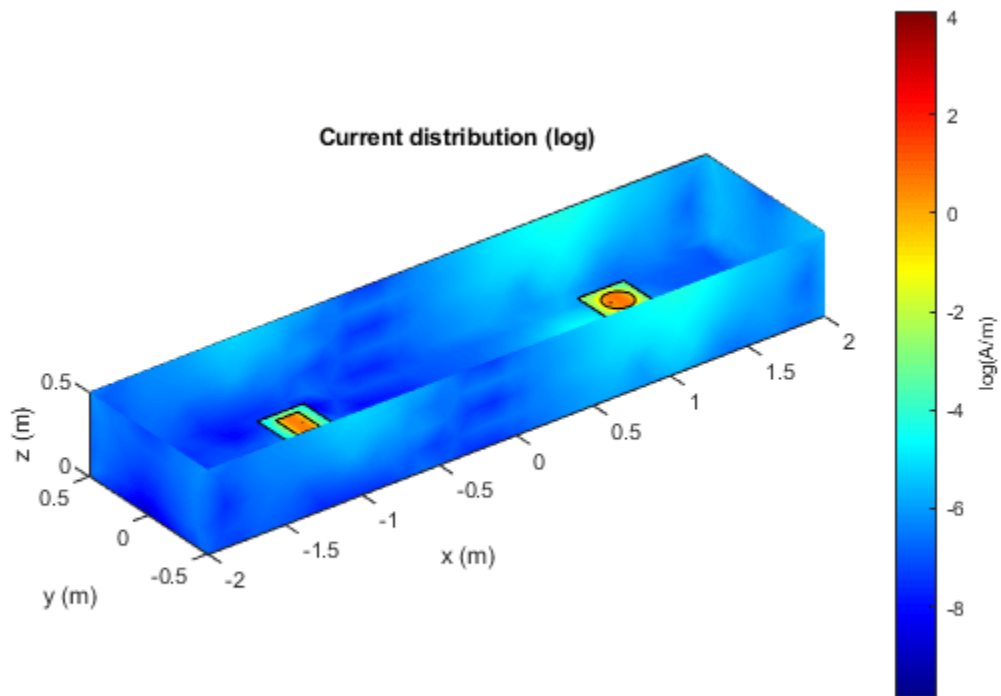
Analyze multiple antenna elements on a platform

All the analysis performed on a single antenna element can also be performed on multiple antenna elements.

```
figure;
pattern(ant, 1e9);
```

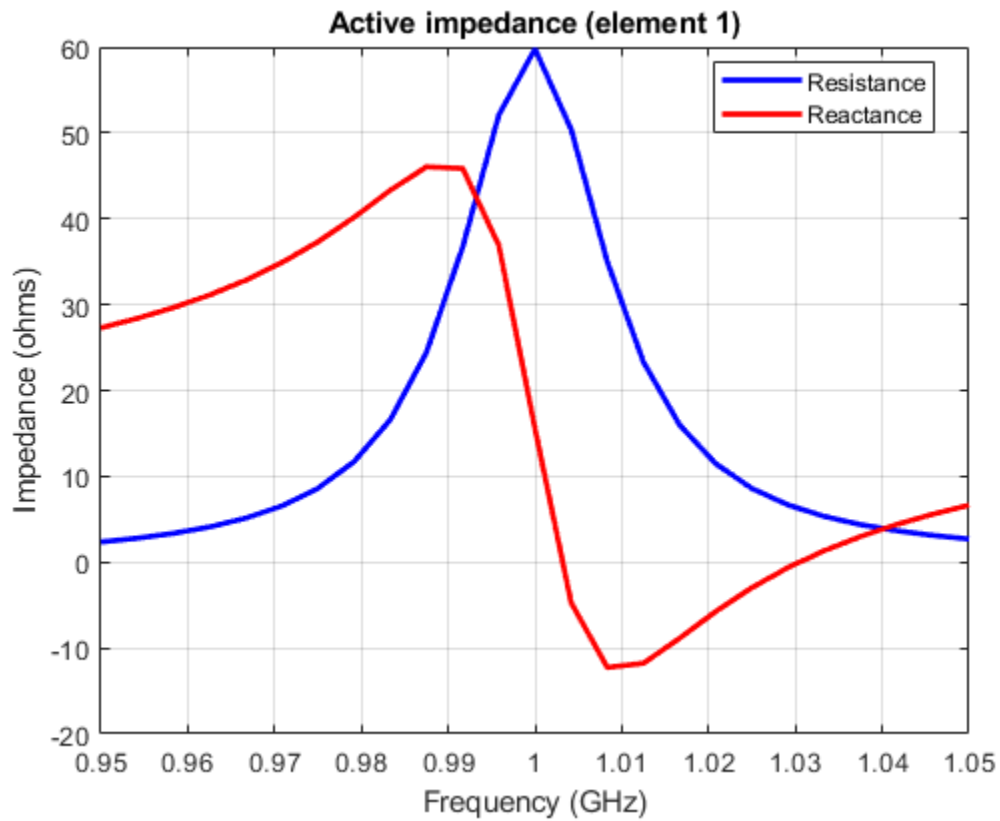


```
figure;  
current(ant, 1e9, 'scale', 'log');
```



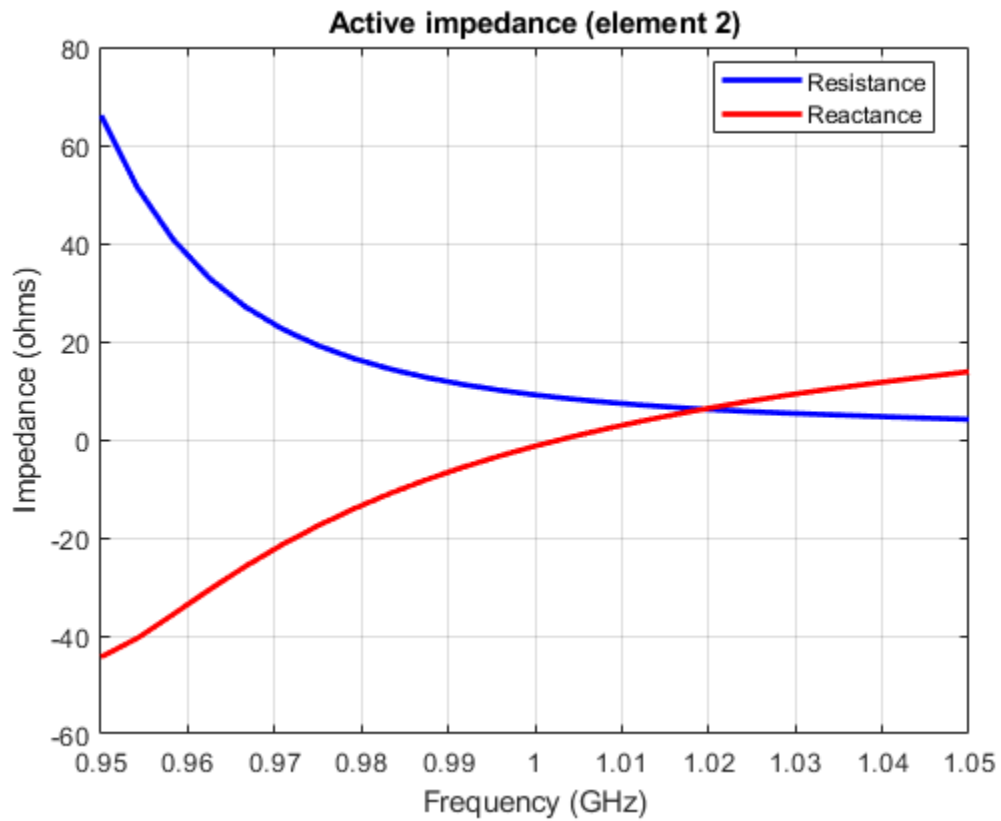
The plot below displays the impedance of the rectangular patch microstrip antenna.

```
figure;  
impedance(ant, linspace(950e6, 1050e6, 25), 1);
```



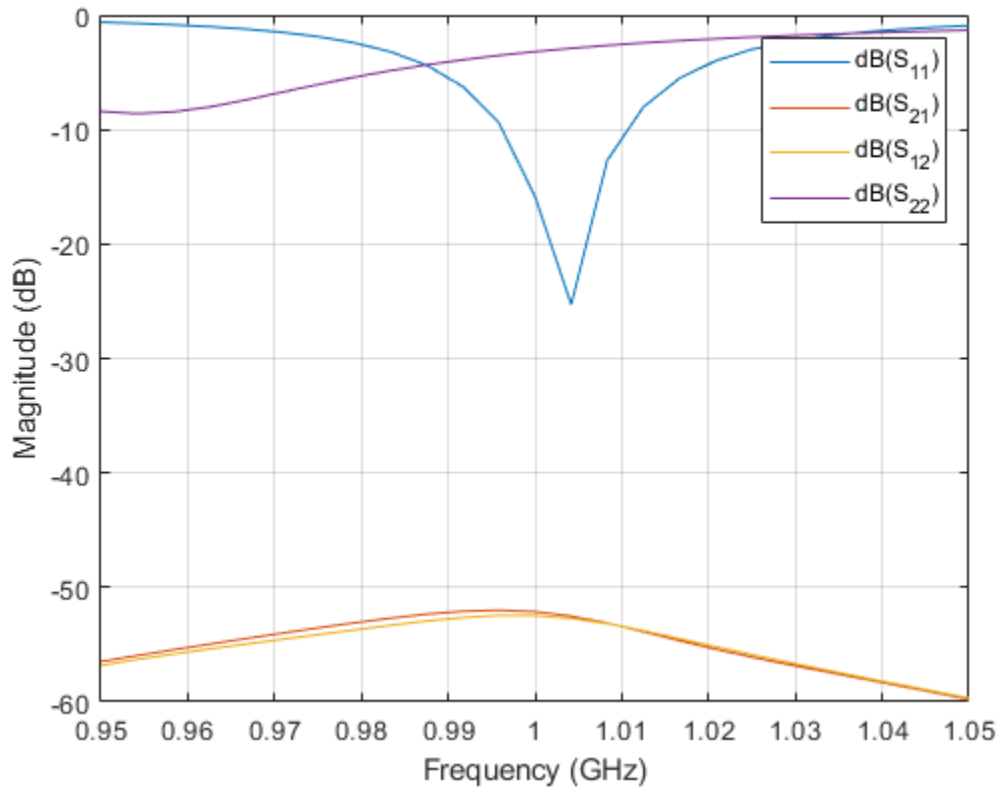
The plot below displays the impedance of the circular patch microstrip antenna.

```
figure;  
impedance(ant, linspace(950e6, 1050e6, 25), 2);
```



The coupling between the antenna elements can be calculated using the s-parameters.

```
S = sparameters(ant, linspace(950e6, 1050e6, 25));  
figure;  
rfplot(S);
```



See Also

“Pattern Analysis of the Symmetric Parabolic Reflector” on page 5-570

3D Reconstruction of Radiation Pattern From 2D Orthogonal Slices

This example shows how to reconstruct 3D radiation pattern using `patternFromSlices` function. A 3D radiation pattern is a very important tool for antenna analysis, characterization, design, planning, and applications. This example will show reconstruction of 3-D radiation from 2 orthogonal slices. Pattern reconstruction for an omni-directional and directional antenna will be considered.

Omni-Directional Antenna

Define an omni-directional antenna such as a dipole with a specific frequency and required elevation and azimuth angle.

```
ant = dipole; freq = 70e6;  
ele = -90:5:90;  
azi = -180:1:180;
```

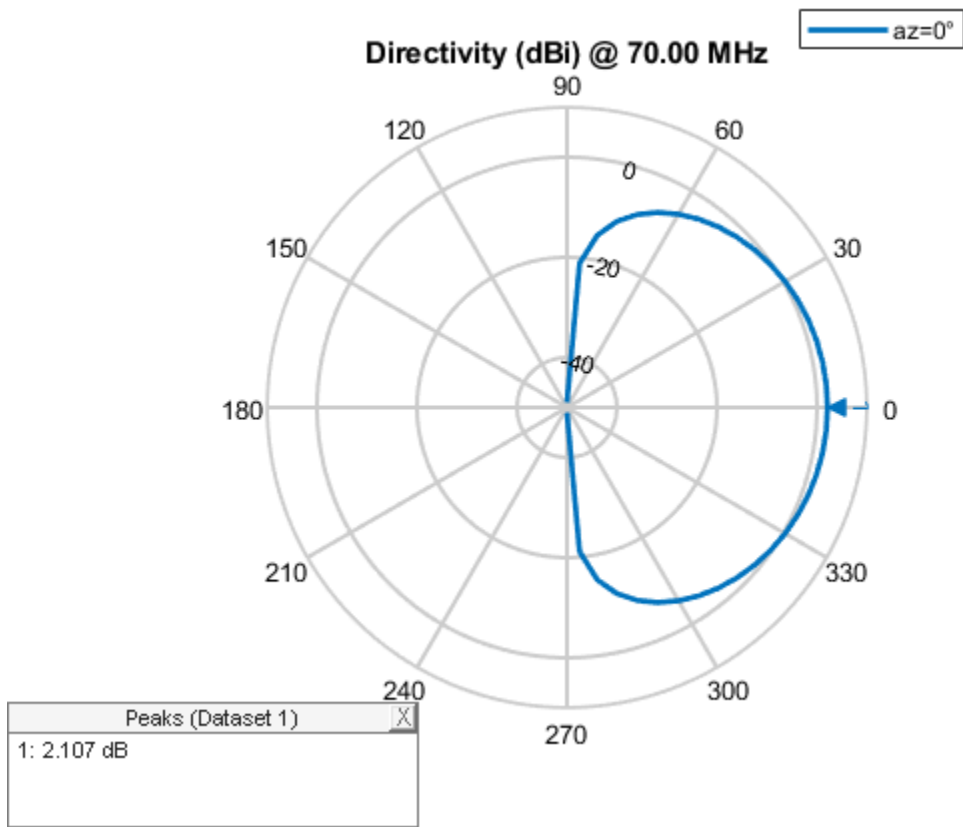
Generate Orthogonal 2-D Slices.

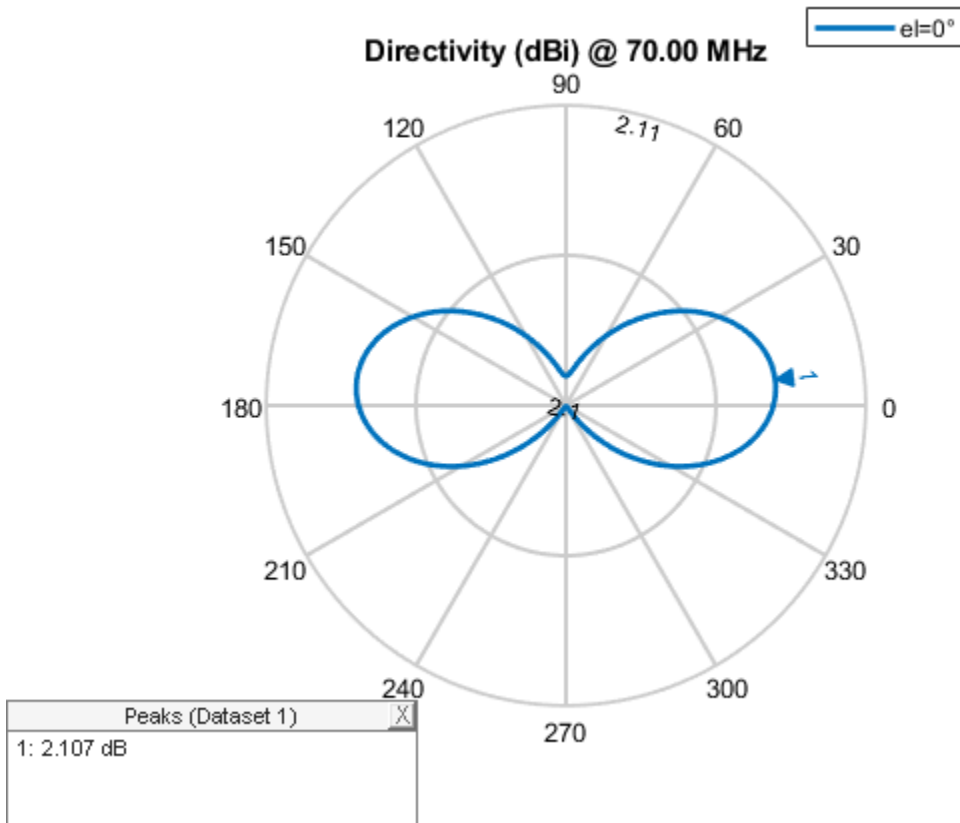
The slice is along the vertical direction using `patternElevation` function. Here we can give other 2-D pattern data also.

```
vertSlice = patternElevation(ant, freq, 0, 'Elevation', ele);  
theta = 90 - ele;
```

The two orthogonal slices can also be visualized.

```
figure;  
patternElevation(ant, freq, 0, 'Elevation', ele);  
figure;  
patternAzimuth(ant, freq, 0, 'Azimuth', azi);
```

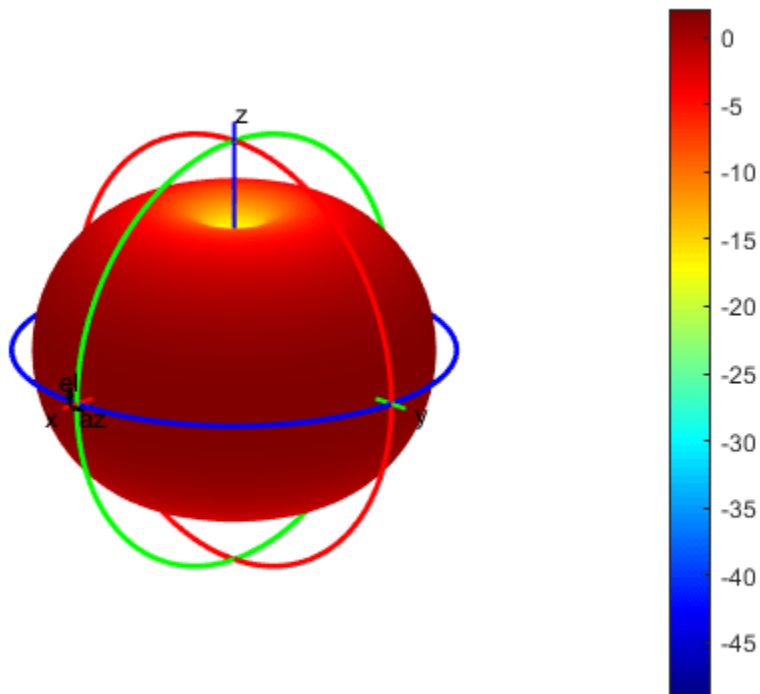




Reconstruction Of 3-D Radiation Pattern

For an omni-directional antenna, we can reconstruct the 3-D pattern using `vertSlice` alone. When only the elevation pattern data is provided, function assumes omnidirectionality of the antenna with symmetry about the z-axis (i.e., azimuthal symmetry).

```
patternFromSlices(vertSlice, theta);
```



Reconstruction using both `vertSlice` & `horizSlice` data points can also be done for the above case. The reconstructed pattern will not vary. Thus, for any omni-directional antenna 3-D pattern can be reconstructed with enough data points from orthogonal slices along the theta direction. The reconstructed radiation pattern looks like the 3-D radiation pattern which can be obtained using the pattern function.

Discarding of Data Points

Discarding of data points during reconstruction of 3-D pattern happens when both data points span across 360 degrees in 2-D plane. Since the algorithm need maximum span of 360 degree in one plane and a span of 180 degree in the other plane, extra data points are discarded.

```
vertSlice = patternElevation(ant,freq);
theta = 90 - (-180:1:180);
```

Dimension of `pat3-D` won't be equal to the `length(phi)*length(theta)` in this case. The size of `thetaout` also vary from that of `theta` dimension. Also, `thetaout` data will show the values for which data points have been considered during reconstruction.

```
[pat3D,thetaout]=patternFromSlices(vertSlice,theta);
dim_theta = size(thetaout);
disp(dim_theta);
```

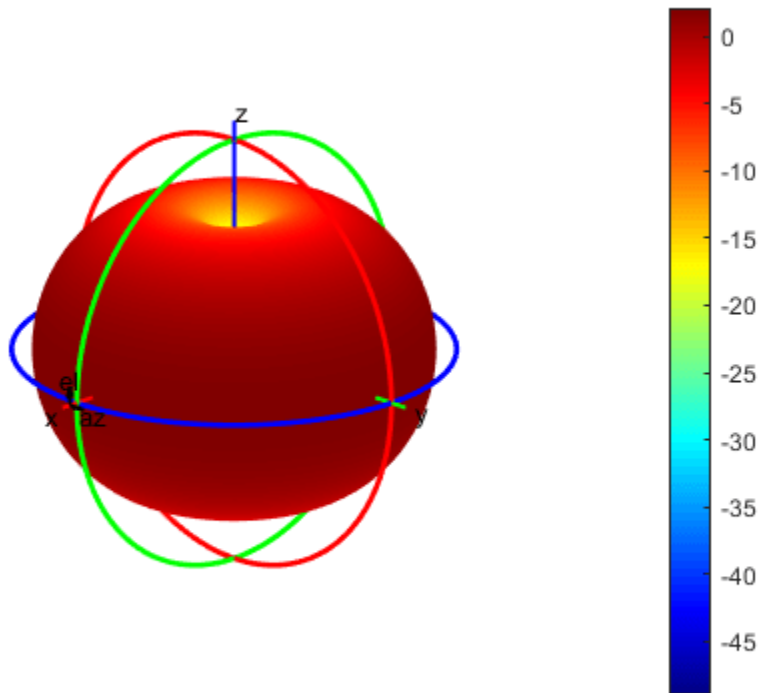
Warning: Vertical pattern slice data from backplane for theta greater than 180 degrees is discarded.

```
1 181
```

3-D radiation pattern will not get affected by data discarding since there will be enough data points along both the orthogonal plane for reconstruction of 3-D pattern. This result will be the same as that of the above reconstructed 3-D radiation pattern.

```
patternFromSlices(vertSlice,theta);
```

Warning: Vertical pattern slice data from backplane for theta greater than 180 degrees is discarded.



Directional Antenna

Define a directional antenna such as helix with specific frequency and values for elevation and azimuth angles.

```
ant_dir = helix('Tilt',90,'TiltAxis',[0 1 0]); freq = 2e9;
ele = -90:5:90;
azi = -180:5:180;
```

Orthogonal 2-D Slices

The slice along the vertical direction using patternElevation function.

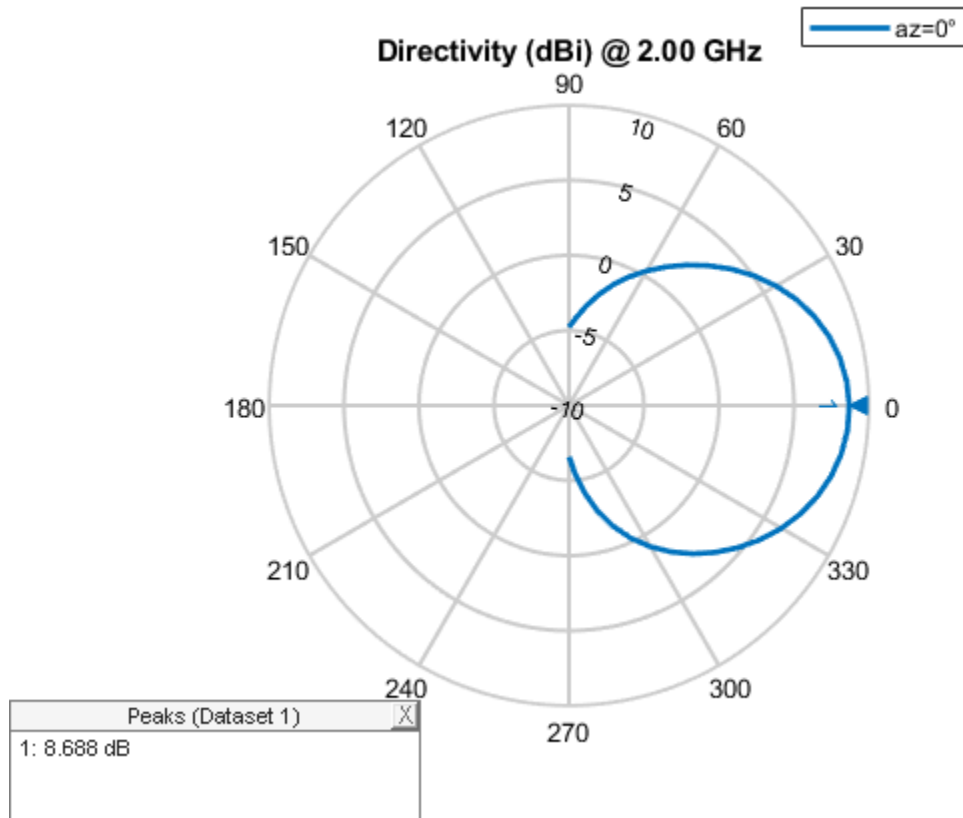
```
vertSlice = patternElevation(ant_dir,freq,0,'Elevation',ele);
theta = 90 - ele;
```

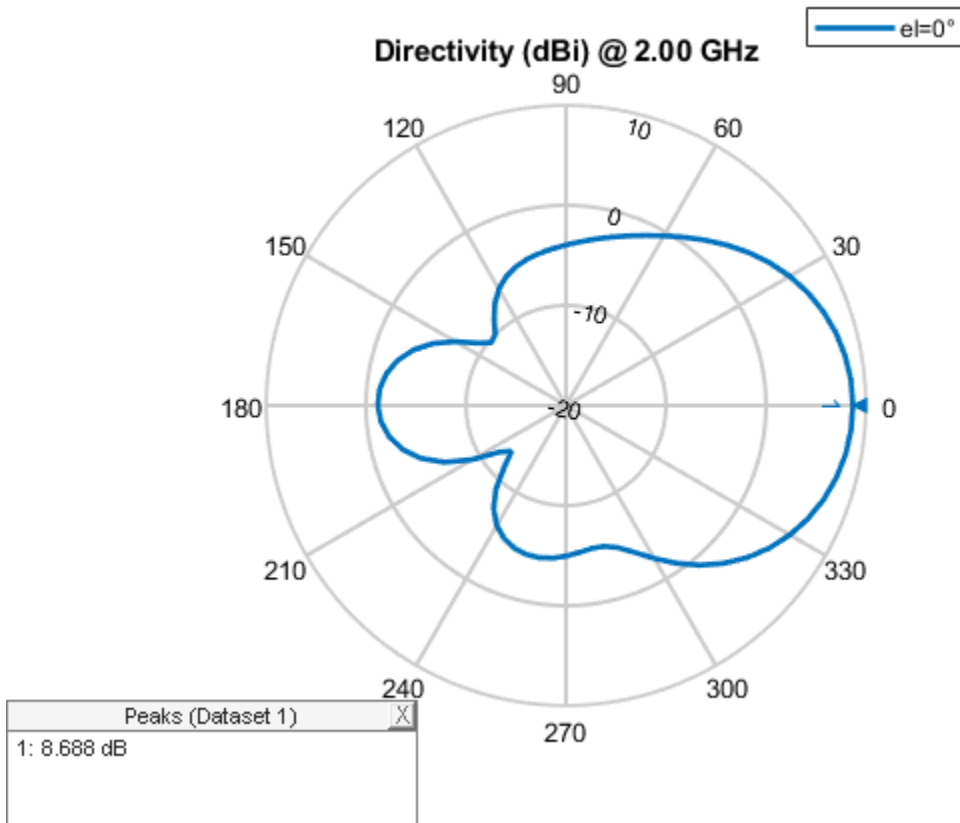
The slice along the horizontal direction using patternAzimuth function.

```
horizSlice = patternAzimuth(ant_dir,freq,0,'Azimuth',azi);  
phi = azi ;
```

The two orthogonal slices can also be visualized.

```
figure;  
patternElevation(ant_dir,freq,0,'Elevation',ele);  
figure;  
patternAzimuth(ant_dir,freq,0,'Azimuth',azi);
```





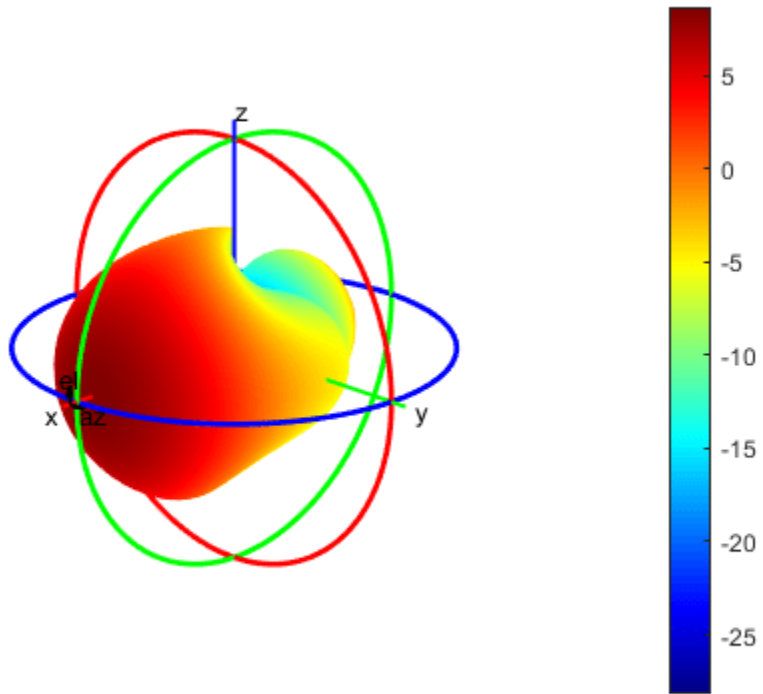
Reconstruction Of 3-D Radiation Pattern

For a directional antenna pattern, both the horizontal and vertical slice must be provided for accurate pattern reconstruction. Two separate algorithms are implemented for pattern reconstruction and will consider both below.

Summing Method.

The "classic" summing algorithm is the default method. This algorithm can be used for near-perfect reconstruction of omni-directional antennas than for directional antenna.

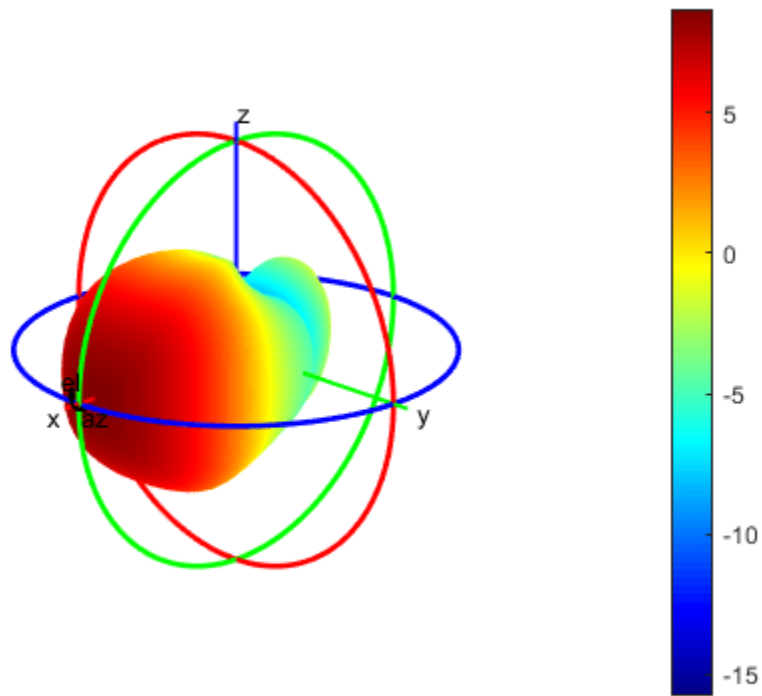
```
patternFromSlices(vertSlice,theta,horizSlice,phi);
```



CrossWeighted Method.

In this algorithm, the normalization parameter can be changed to obtain different results for the reconstructed pattern about the estimated directivity/gain

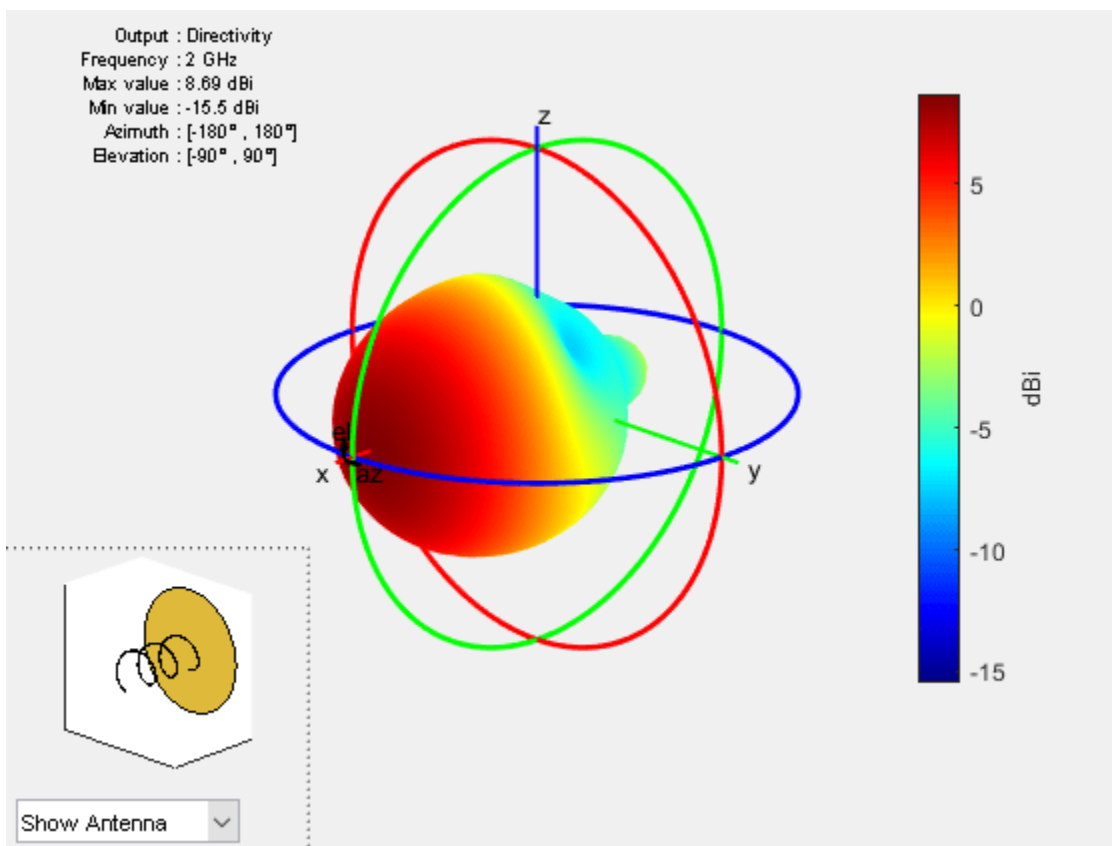
```
patternFromSlices(vertSlice,theta,horizSlice,phi,'Method','CrossWeighted');
```



3-D Radiation Using Pattern Function

Originally 3-D radiation pattern using pattern function for helix

```
figure;  
pattern(ant_dir, freq);
```



From comparing the above 3-D radiation pattern using pattern function and the reconstructed 3-D pattern it is clear that front plane of the 3-D pattern is reconstructed well compared to the back plane of it. Also, when reconstruction done using CrossWeighted method is more accurate than summing method for this case.

Read and Visualize Antenna Data from Manufacturer

Antenna manufacturers typically provide details of the antennas that they supply together with the two orthogonal slices of the radiation pattern. The pattern data is available in a variety of formats. One such format that is supported in the Antenna Toolbox is the MSI file format (extension .msi or .pln). Use the msiread function to read the data into the workspace.

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

Adjust To dBi if data is in dBd

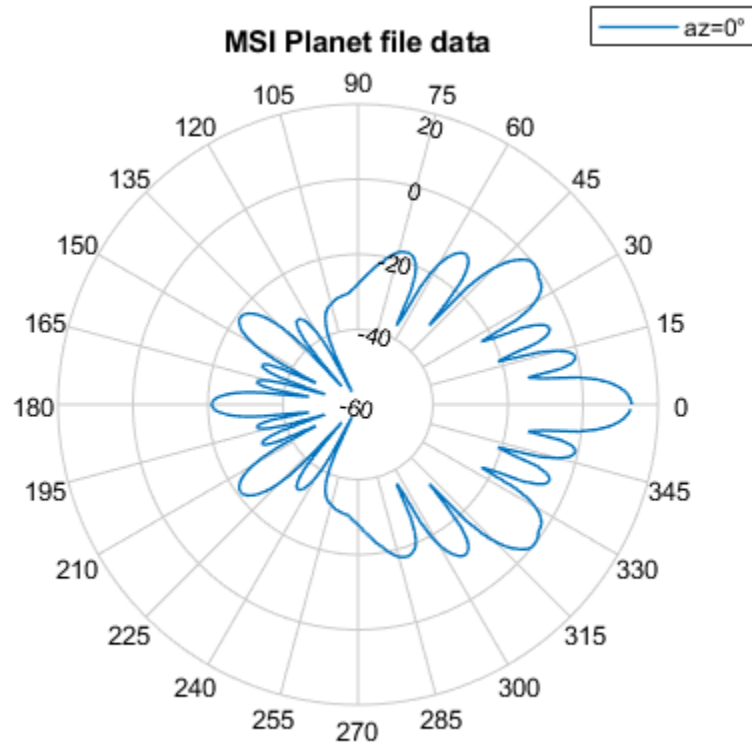
```
if strcmpi(Optional.gain.unit,'dBd')
    Horizontal.Magnitude = Horizontal.Magnitude + 2;
    Vertical.Magnitude = Vertical.Magnitude + 2;
end
```

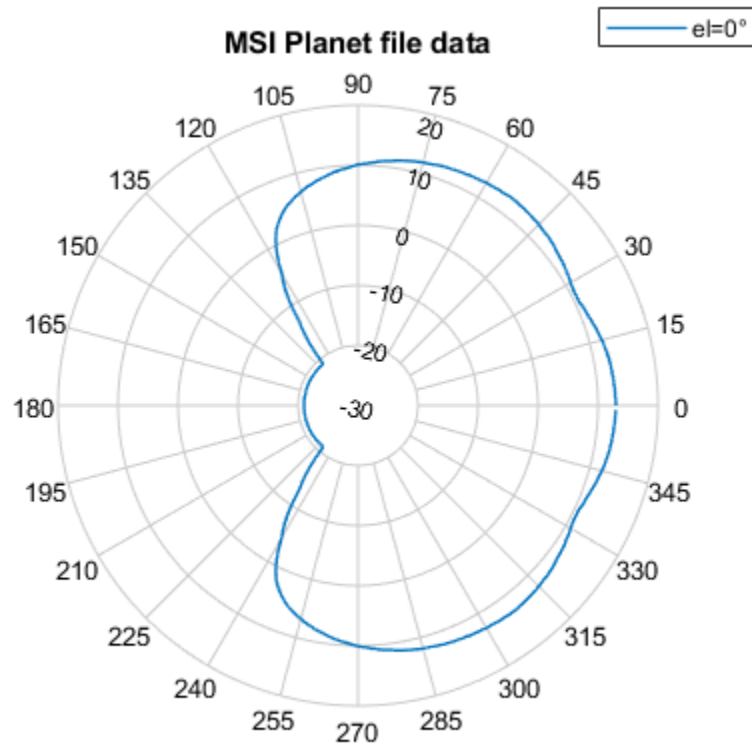
Visualize the vertical and horizontal gain data in an interactive 2-D polar plot.

```
figure
P = polarpattern(Vertical.Elevation, Vertical.Magnitude);
P.TitleTop = 'MSI Planet file data';
createLabels(P, 'az=0#deg');
```



```
figure
Pel = polarpattern(Horizontal.Azimuth, Horizontal.Magnitude);
Pel.TitleTop = 'MSI Planet file data';
createLabels(Pel, 'el=0#deg');
```



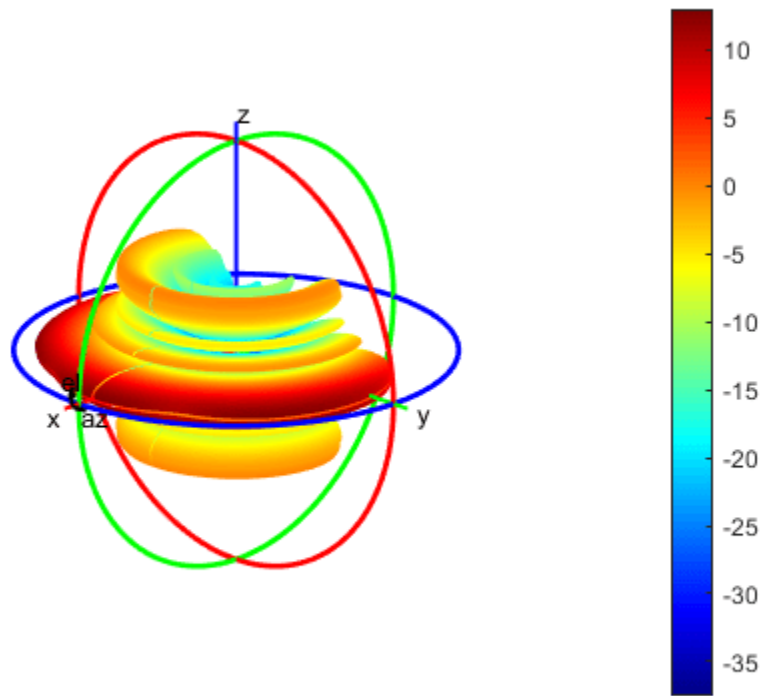


Reconstruction Of 3-D Radiation Pattern

Extract the pattern slice magnitude data from the two output structures as well as the azimuth and elevation angle data. Note, that the angle data should be adjusted for the phi-theta convention. The azimuth angles maps to phi but the elevation angle is adjusted by 90 degrees to map to theta.

```
vertSlice = Vertical.Magnitude;
theta = 90-Vertical.Elevation;
horizSlice = Horizontal.Magnitude;
phi = Horizontal.Azimuth;
patternFromSlices(vertSlice,theta,horizSlice,phi,'Method','CrossWeighted');
```

Warning: Vertical pattern slice data from backplane for theta greater than 180 degrees is discarded.



See Also

“Visualize Antenna Coverage Map and Communication Links” on page 5-309

Loading Using Lumped Elements

This example shows how to load an antenna using a `lumpedElement`. You can load an antenna to make it smaller, enable matching at the feedline, or to make antennas larger to obtain higher performance. The example below demonstrates creating a simple matching network by adding a lumped load at the feed.

Define a `lumpedElement`

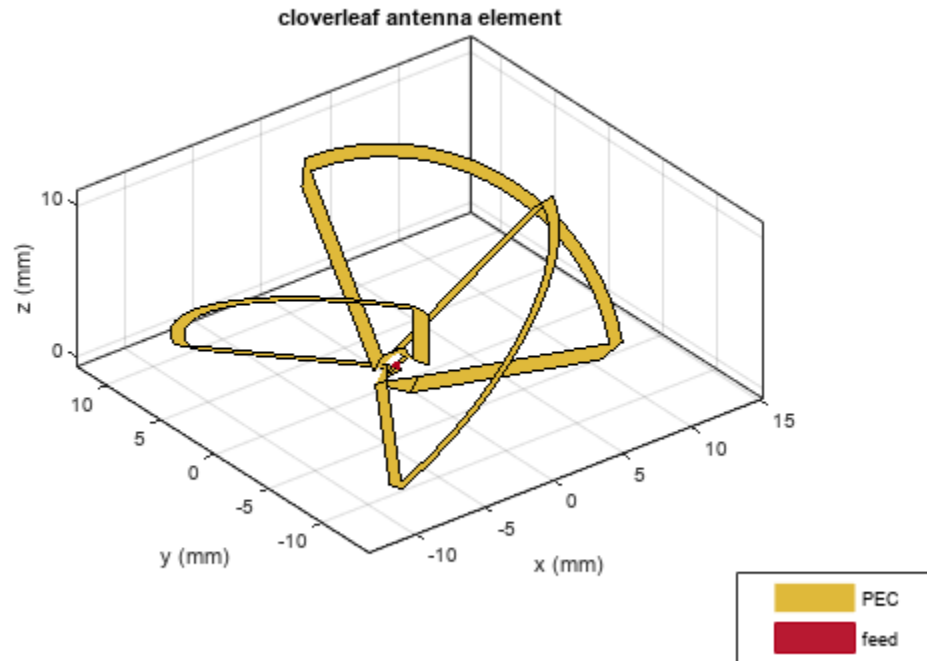
The `lumpedElement` allows the user to specify a complex load. The load can be frequency independent (scalar) or dependent (vector). You can define the frequency variation as a vector using the `Frequency` property. You can also choose the location on the antenna surface where the load needs to be specified. To perform impedance matching the load is applied to the feed.

```
le = lumpedElement
le =
  lumpedElement with properties:
    Impedance: []
    Frequency: []
    Location: 'feed'
```

Cloverleaf antenna

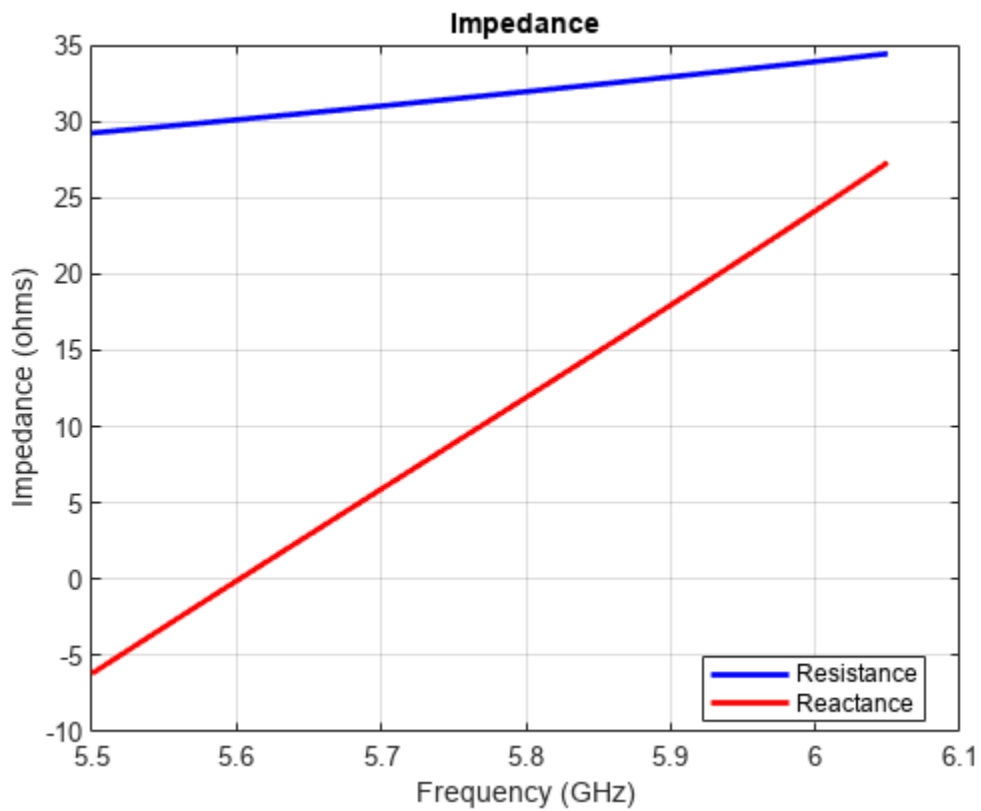
Pick a cloverleaf antenna from the catalog. A cloverleaf antenna is typically used on drones for wireless communication between 5.5-6.05GHz.

```
ant = cloverleaf;
freq = linspace(5.5e9, 6.05e9, 51);
figure; show(ant);
```



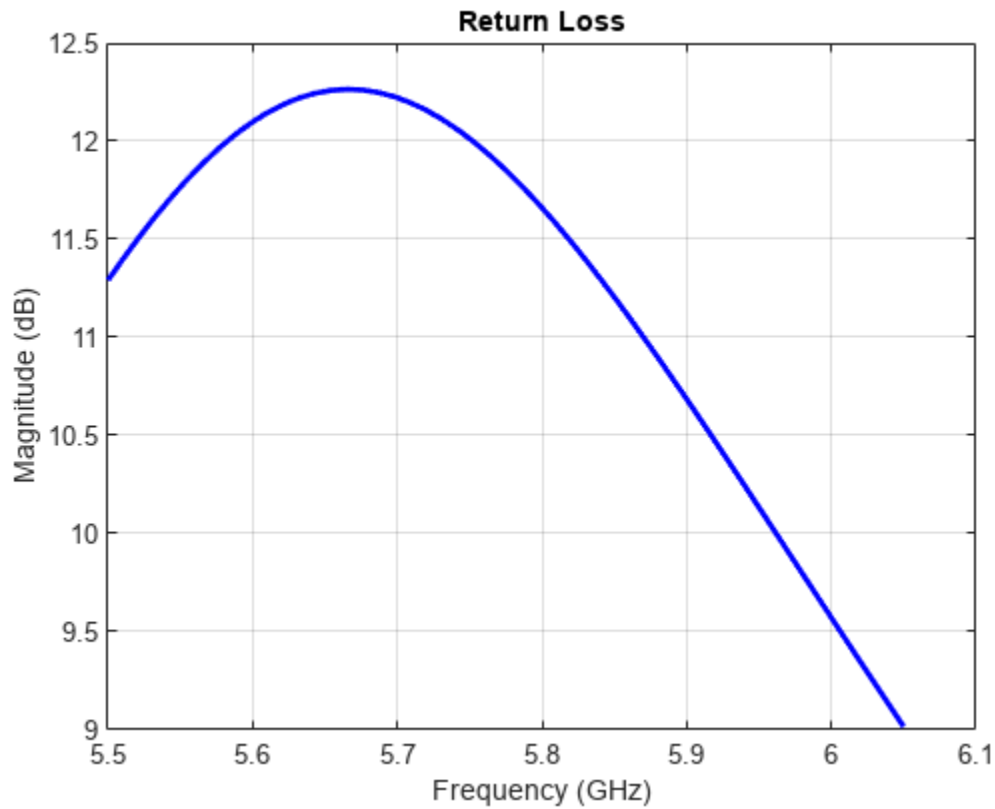
As seen from the impedance plot, the antenna resonates at 5.6GHz. The resistance and reactance value increase at higher frequencies. The impedance at 5.8 GHz is $32 + j12$.

```
figure; impedance(ant, freq);
```



The matching at the lower end seems pretty good but at the higher frequencies we do not have a good match beyond 5.95 GHz. This prevents the antenna from matching the specifications required for successful operation over the entire frequency range.

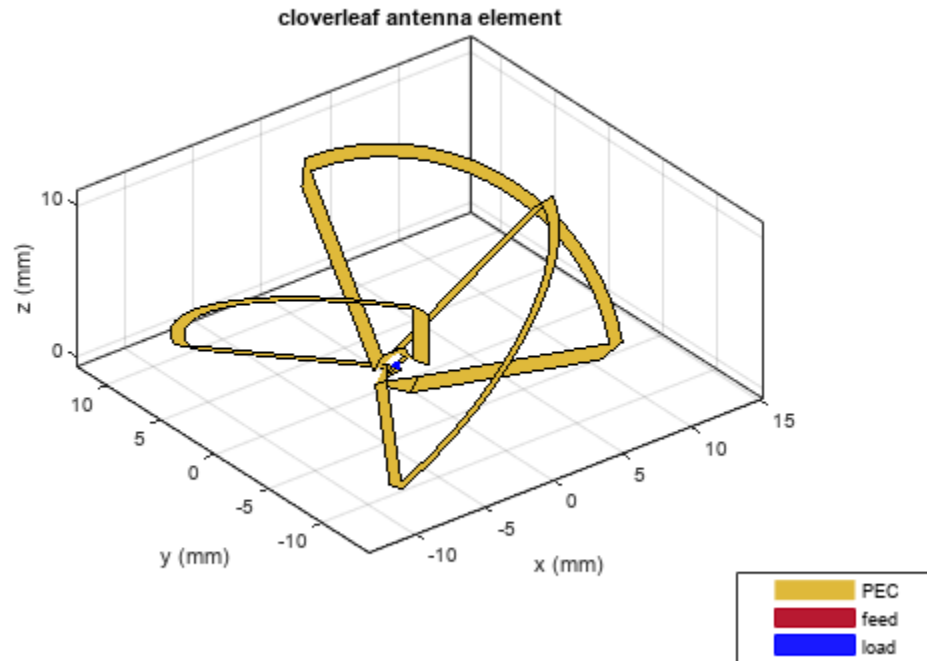
```
figure; returnLoss(ant, freq);
```



Impedance matching - Adding load at the feed

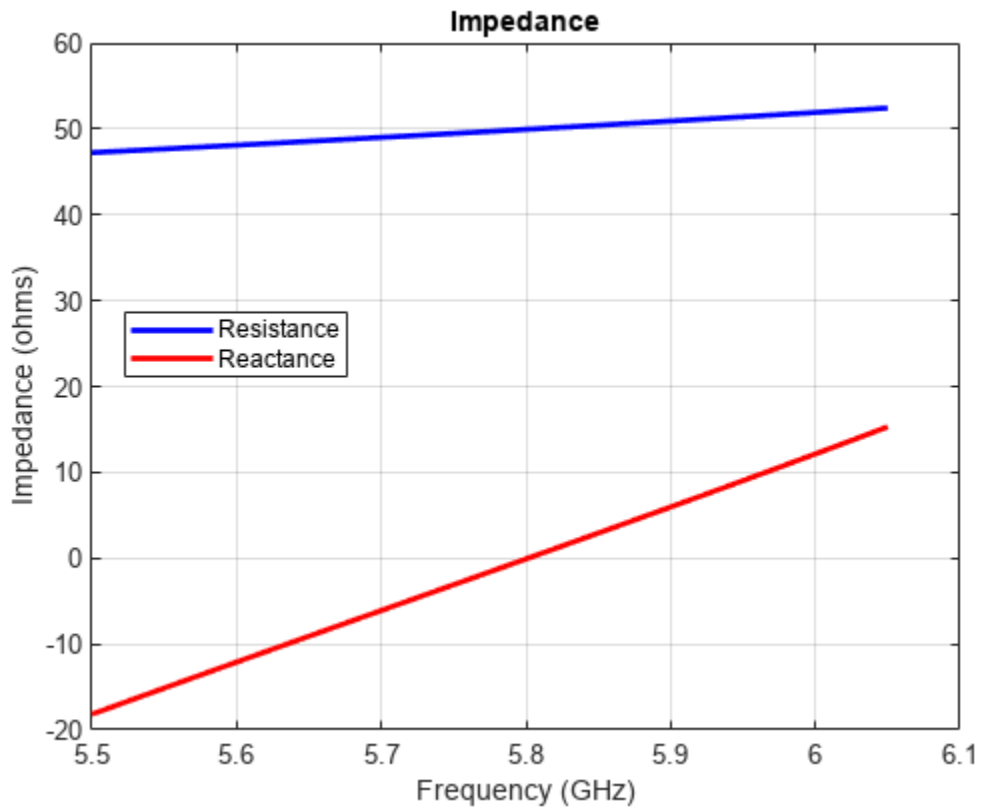
You can obtain better matching over the entire frequency range by adding some impedance at the feed. As impedance variation is quite smooth a single impedance value might be enough to obtain a good match over the entire frequency range. We select the impedance at 5.8 GHz and try to match it exactly to 50 ohms. A blue dot is added at the feed indicating the location of the lumped load.

```
le.Impedance = complex(18, -12);  
ant.Load = le;  
figure; show(ant);
```



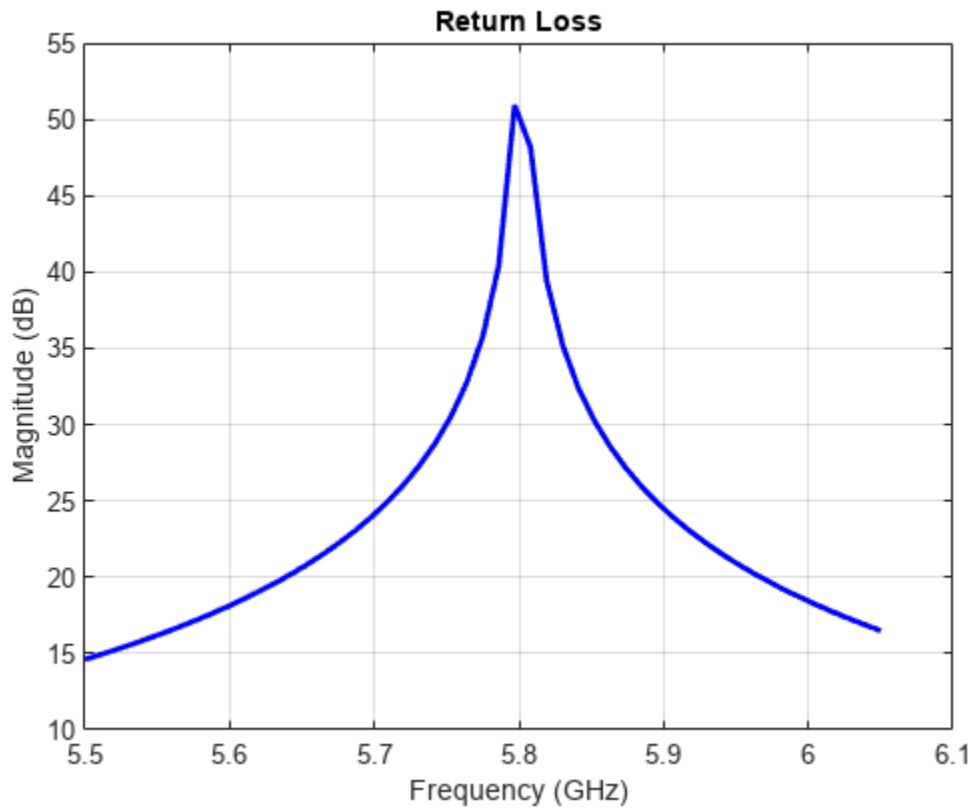
Calculating the impedance, indicates that a constant 18 ohms is added to the resistance while a capacitive reactance of 12 ohms is added to the reactance over the entire frequency range. This also changes the resonant frequency of the antenna.

```
figure; impedance(ant, freq);
```

However, changing the impedance over the entire frequency range does help with the impedance bandwidth. A value greater than 10 dB of return loss is seen over the entire frequency range of operation.

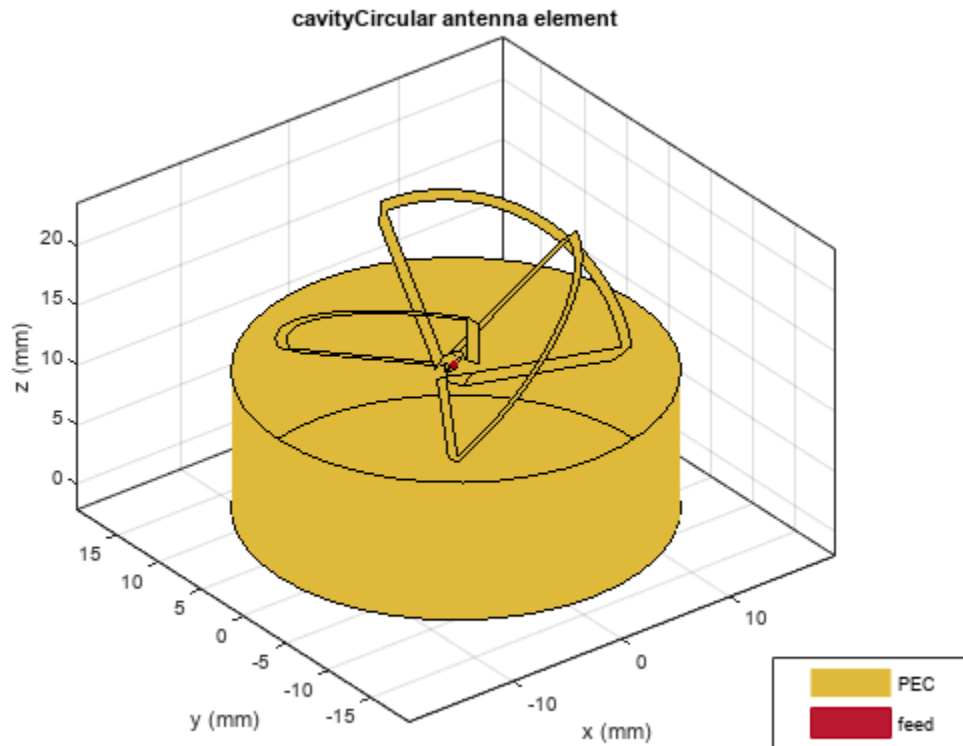
```
figure; returnLoss(ant, freq);
```



Adding load at arbitrary location on the antenna surface

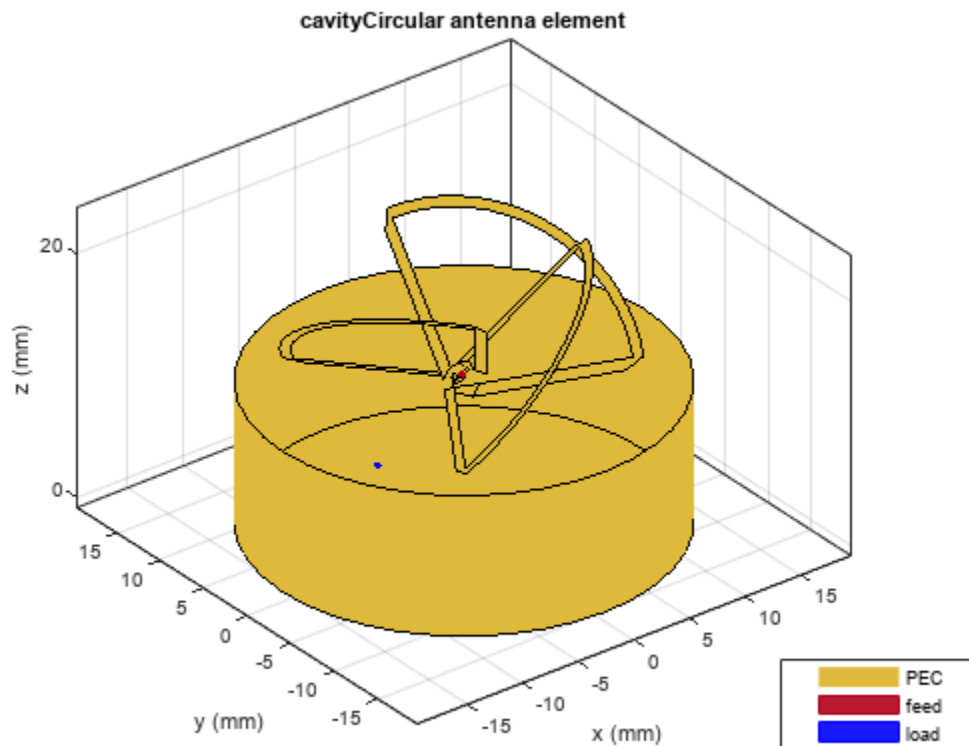
You can also load the antenna at an arbitrary location on the surface by specifying the x, y and z coordinates of the location. Consider the same cloverleaf antenna but backed by a circular cavity as shown below.

```
ref = design(cavityCircular, 5.5e9);  
ref.Exciter = cloverleaf;  
figure; show(ref);
```



You can add some load to the cavity base to add more loss into the system. The blue dot added to the cavity base is the location of the lumped load.

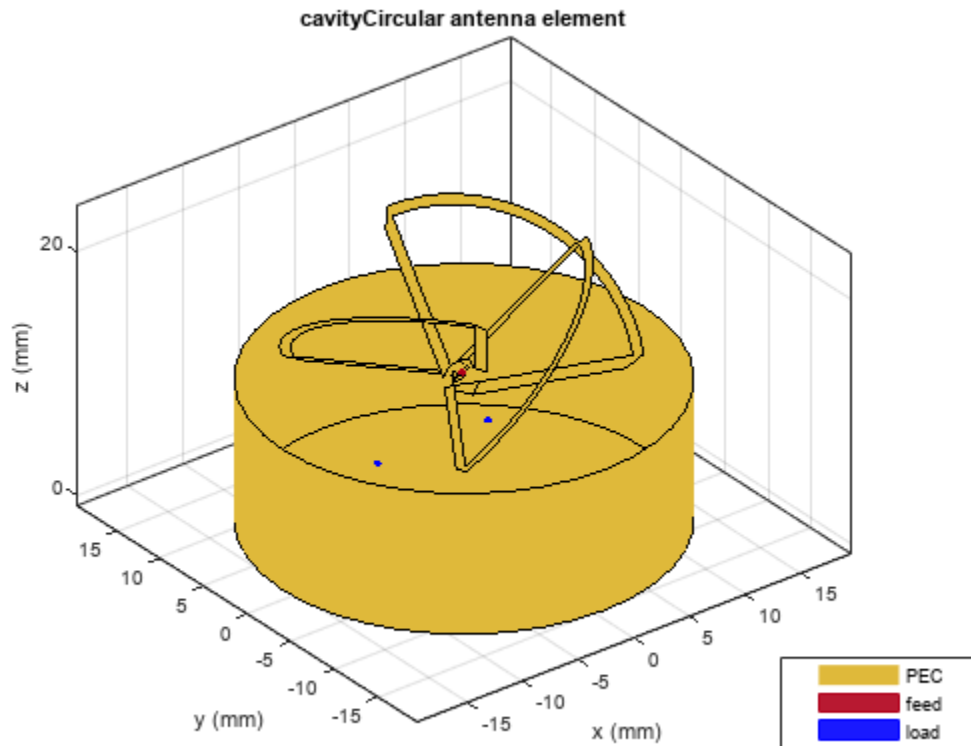
```
reload = lumpedElement('Impedance',complex(20, 20), 'Location', [0 10e-3 0]);  
ref.Load = reload;  
figure; show(ref);
```



Adding multiple loads

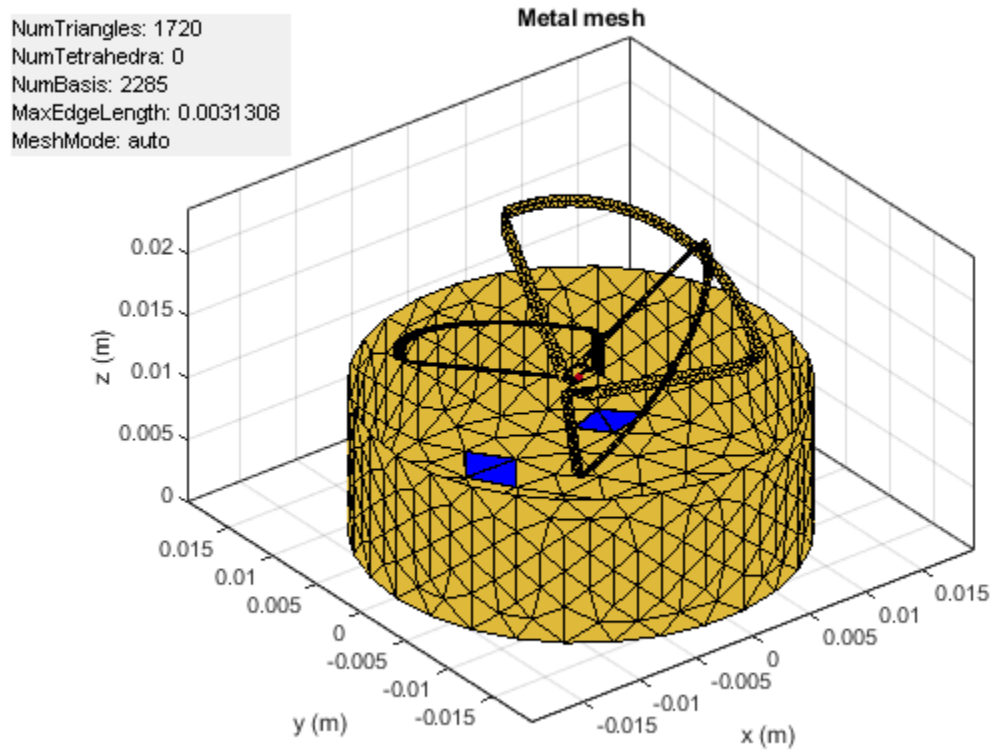
Multiple loads can be added to an antenna by specifying multiple `lumpedElement`. You can specify these either at the feed or on the antenna surface. Multiple blue dots are observed on the antenna surface indicating the location of the load.

```
refload2 = lumpedElement('Impedance', complex(30, -10), 'Location', [10e-3, 10e-3, 0]);  
ref.Load = [refload, refload2];  
figure; show(ref);
```



All the analysis performed on the above antenna will take into account the effect of the lumped load. This is done by adding the impedance value specified in the lumped load to the basis function in the interaction matrix of Method of Moments. The edge at which the load is added can be visualized by looking at the mesh. The common edge shared by the two blue triangles is the edge at which the impedance value gets added.

```
z = impedance(ref, 5.5e9);
figure; mesh(ref);
```



See Also

“Comparison of Antenna Array Transmit and Receive Manifold” on page 5-364 | “Impedance Matching of Non-resonant (Small) Monopole” on page 5-141

Planning Radar Network Coverage over Terrain

This example shows how to plan a radar network using propagation modelling over terrain. DTED level-1 terrain data is imported for a region that contains five candidate monostatic radar sites. The radar equation is used to determine whether target locations can be detected, where additional path loss is calculated using either the Longley-Rice propagation model or Terrain Integrated Rough Earth Model™ (TIREM™). The best three sites are selected for detection of a target that is flying at 500 meters above ground level. The scenario is updated to model a target that is flying at 250 meters above ground level. Radar coverage maps are shown for both scenarios.

Import Terrain Data

Import DTED-format terrain data for a region around Boulder, Colorado, US. The terrain file was downloaded from the SRTM Void Filled data set available from the United States Geological Survey (USGS). The file is DTED level-1 format and has a sampling resolution of about 90 meters. A single DTED file defines a region that spans 1 degree in both latitude and longitude.

```
dtedfile = "n39_w106_3arc_v2.dt1";
attribution = "SRTM 3 arc-second resolution. Data available from the U.S. Geological Survey.";
addCustomTerrain("southboulder",dtedfile,"Attribution",attribution)
```

Open Site Viewer using the imported terrain. Visualization with high-resolution satellite map imagery requires an Internet connection.

```
viewer = siteviewer("Terrain","southboulder");
```

Show Candidate Radar Sites

The region contains mountains to the west and flatter areas to the east. Radars will be placed in the flat area to detect targets over the mountainous region. Define five candidate locations for placing the radars and show them on the map. The candidate locations are chosen to correspond to local high points on the map outside of residential areas.

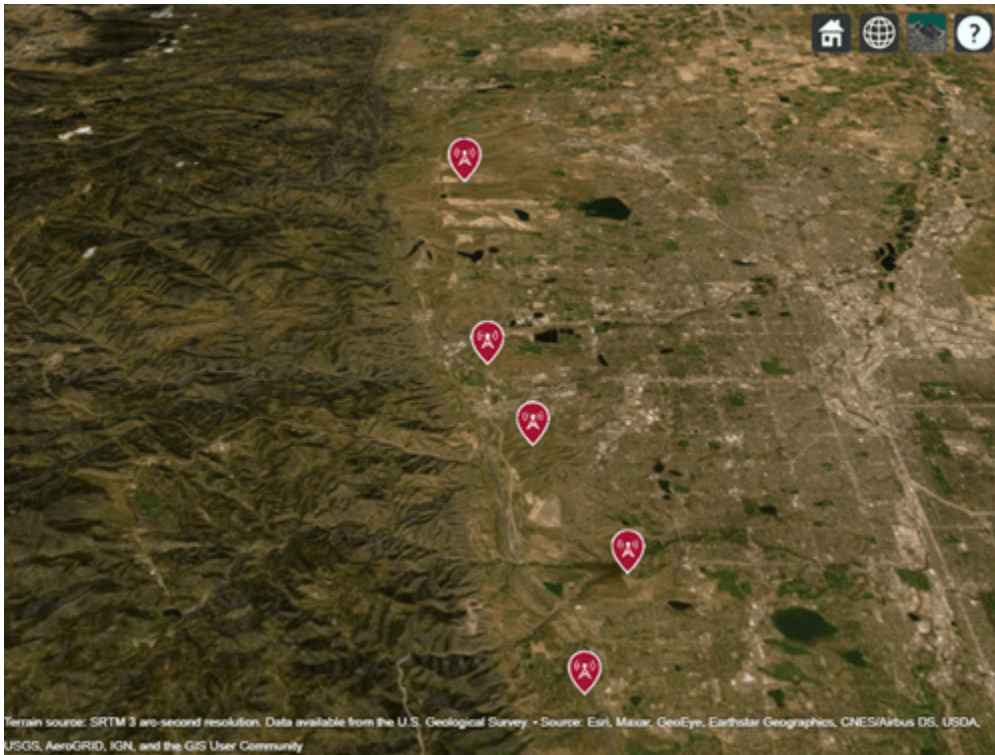
Create collocated transmitter and receiver sites at each location to model monostatic radars, where the radar antennas are assumed to be 10 meters above ground level.

```
names = "Radar site" + (1:5);
rdrlats = [39.6055 39.6481 39.7015 39.7469 39.8856];
rdrlons = [-105.1602 -105.1378 -105.1772 -105.2000 -105.2181];
```

```
% Create transmitter sites associated with radars
rdrtxs = txsite("Name",names, ...
    "AntennaHeight",10, ...
    "Latitude",rdrlats, ...
    "Longitude",rdrlons);
```

```
% Create receiver sites associated with radars
rdrrxs = rxsite("Name",names, ...
    "AntennaHeight",10, ...
    "Latitude",rdrlats, ...
    "Longitude",rdrlons);
```

```
% Show just the radar transmitter sites
show(rdrtxs);
```



Zoom and rotate the map to view the 3-D terrain around the candidate radar sites. Select a site to view the location, antenna height, and ground elevation.



Design Monostatic Radar System

Design a basic monostatic pulse radar system to detect nonfluctuating targets with 0.1 square meter radar cross section (RCS) at a distance up to 35,000 meters from the radar with a range resolution of 5 meters. The desired performance index is a probability of detection (Pd) of 0.9 and probability of false alarm (Pfa) below $1e-6$. The radars are assumed to be rotatable and support the same antenna gain in all directions, where the antenna gain corresponds to a highly directional antenna array.

```
pd = 0.9;           % Probability of detection
pfa = 1e-6;        % Probability of false alarm
maxrange = 35000; % Maximum unambiguous range (m)
rangeres = 5;     % Required range resolution (m)
tgtrcs = .1;      % Required target radar cross section (m^2)
```

Use pulse integration to reduce the required SNR at the radar receiver. Use 10 pulses and compute the SNR required to detect a target.

```
numpulses = 10;
snrthreshold = albersheim(pd, pfa, numpulses); % Unit: dB
disp(snrthreshold);

4.9904
```

Define radar center frequency and antenna gain assuming a highly directional antenna array.

```
fc = 10e9; % Transmitter frequency: 10 GHz
antgain = 38; % Antenna gain: 38 dB
c = physconst('LightSpeed');
lambda = c/fc;
```

Calculate the required peak pulse power (Watts) of the radar transmitter using the radar equation.

```
pulsebw = c/(2*rangeres);
pulsewidth = 1/pulsebw;
Ptx = radareqpow(lambda,maxrange,snrthreshold,pulsewidth,...
    'RCS',tgtrcs,'Gain',antgain);
disp(Ptx)

3.1521e+05
```

Define Target Positions

Define a grid containing 2500 locations to represent the geographic range of positions for a moving target in the region of interest. The region of interest spans 0.5 degrees in both latitude and longitude and includes mountains to the west as well as some of the area around the radar sites. The goal is to detect targets that are in the mountainous region to the west.

```
% Define region of interest
latlims = [39.5 40];
lonlims = [-105.6 -105.1];

% Define grid of target locations in region of interest
tgtlatv = linspace(latlims(1),latlims(2),50);
tgtlonv = linspace(lonlims(1),lonlims(2),50);
[tgtlons,tgtlats] = meshgrid(tgtlonv,tgtlatv);
tgtlons = tgtlons(:);
tgtlats = tgtlats(:);
```

Compute the minimum, maximum, and mean ground elevation for the target locations.

```
% Create temporary array of sites corresponding to target locations and query terrain
Z = elevation(txsite("Latitude",tgtlats,"Longitude",tgtlons));
[Zmin, Zmax] = bounds(Z);
Zmean = mean(Z);

disp("Ground elevation (meters): Min    Max    Mean" + newline + ...
     "          " + round(Zmin) + "    " + round(Zmax) + "    " + round(Zmean))

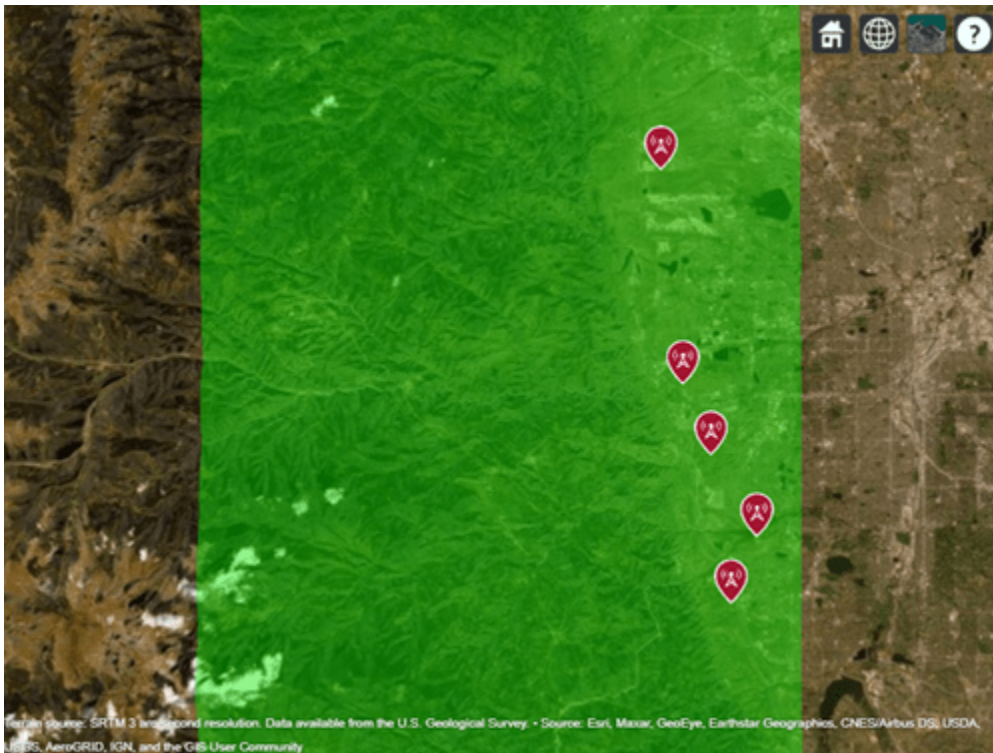
Ground elevation (meters): Min    Max    Mean
                          1257   3953   2373
```

Target altitude can be defined with reference to mean sea level or to ground level. Use ground level as the reference and define a target altitude of 500 meters.

```
% Target altitude above ground level (m)
tgtalt = 500;
```

Show the region of interest as a solid green area on the map.

```
viewer.Name = "Radar Coverage Region of Interest";
regionData = propagationData(tgtlats,tgtlons,'Area',ones(size(tgtlats)));
contour(regionData,'ShowLegend',false,'Colors','green','Levels',0)
```



Calculate SNR for Target Positions with Terrain

The radar equation includes free space path loss and has a parameter for additional losses. Use a terrain propagation model to predict the additional path loss over terrain. Use TIREM™ from Huntington Ingalls Industries if it is available, or else use the Longley-Rice (ITM) model. TIREM™ supports frequencies up to 1000 GHz, whereas Longley-Rice is valid up to 20 GHz. Compute total additional loss including propagation from the radar to the target and then back from the target to the receiver.

```

% Create a terrain propagation model, using TIREM or Longley-Rice
tiremloc = tiremSetup;
if ~isempty(tiremloc)
    pm = propagationModel('tirem');
else
    pm = propagationModel('longley-rice');
end

% Compute additional path loss due to terrain and return distances between radars and targets
[L, ds] = helperPathlossOverTerrain(pm, rdrtxs, rdrxs, tglats, tglons, tgtalt);

```

Use the radar equation to compute the SNR at each radar receiver for the signal reflected from each target.

```

% Compute SNR for all radars and targets
numtgts = numel(tglats);
numrdrs = numel(rdrtxs);
rawsnr = zeros(numtgts,numrdrs);
for tgtind = 1:numtgts
    for rdrind = 1:numrdrs
        rawsnr(tgtind,rdrind) = radareqsnr(lambda,ds(tgtind,rdrind),Ptx,pulsewidth, ...
            'Gain',antgain,'RCS',tgtrcs,'Loss',L(tgtind,rdrind));
    end
end

```

Optimize Radar Coverage

A target is detected if the radar receiver SNR exceeds the SNR threshold computed above. Consider all combinations of radar sites and select the three sites that produce the highest number of detections. Compute the SNR data as the best SNR available at the receiver of any of the selected radar sites.

```

bestsitenums = helperOptimizeRadarSites(rawsnr, snrthreshold);
snr = max(rawsnr(:,bestsitenums),[],2);

```

Display radar coverage showing the area where the SNR meets the required threshold to detect a target. The three radar sites selected for best coverage are shown using red markers.

The coverage map shows straight edges on the north, east, and south sides corresponding to the limits of the region of interest. The coverage map assumes that the radars can rotate and produce the same antenna gain in all directions and that the radars can transmit and receive simultaneously so that there is no minimum coverage range.

The coverage map has jagged portions on the western edge where the coverage areas are limited by terrain effects. A smooth portion of the western edge appears where the coverage is limited by the design range of the radar system, which is 35000 meters.

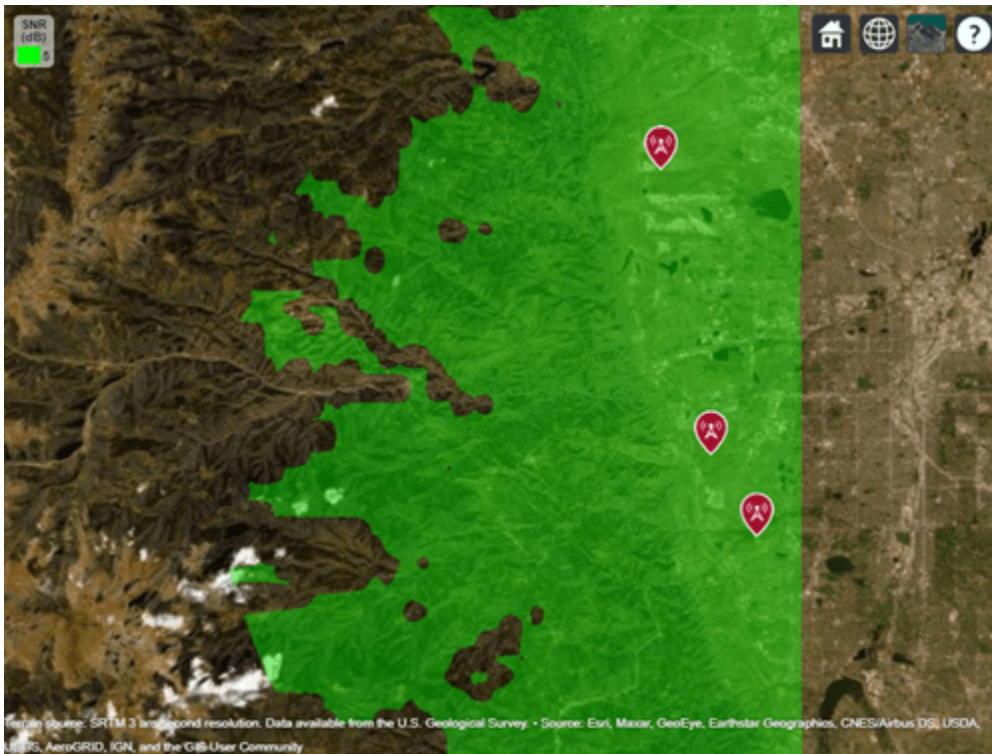
```

% Show selected radar sites using red markers
viewer.Name = "Radar Coverage";
clearMap(viewer)
show(rdrtxs(bestsitenums))

% Plot radar coverage
rdrData = propagationData(tglats,tglons,"SNR",snr);
legendTitle = "SNR" + newline + "(dB)";
contour(rdrData, ...
    "Levels",snrthreshold, ...

```

```
"Colors","green", ...
"LegendTitle",legendTitle)
```

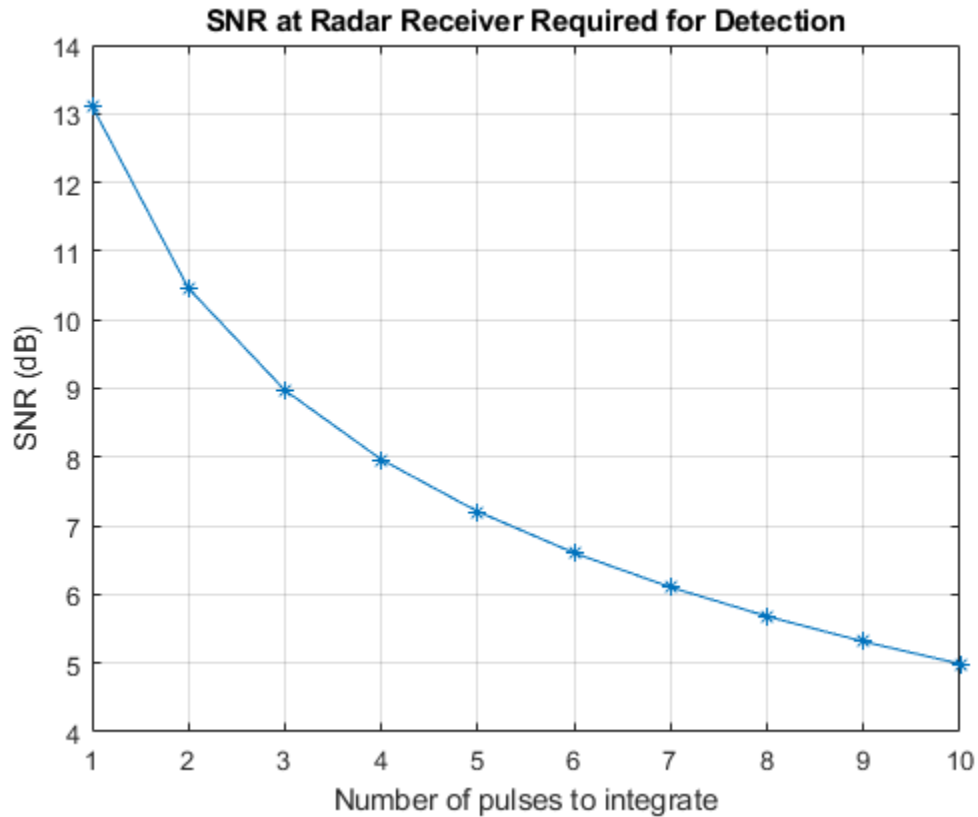


Vary the Number of Pulses to Integrate

The previous analysis optimized radar transmitter power and site locations based on a system that integrates 10 pulses. Now investigate the impact on radar coverage for different modes of operation of the system, where the number of pulses to integrate is varied. Compute the SNR thresholds required to detect a target for varying number of pulses.

```
% Calculate SNR thresholds corresponding to different number of pulses
numpulses = 1:10;
snrthresholds = zeros(1,numel(numpulses));
for k = 1:numel(numpulses)
    snrthresholds(k) = albersheim(pd, pfa, numpulses(k));
end

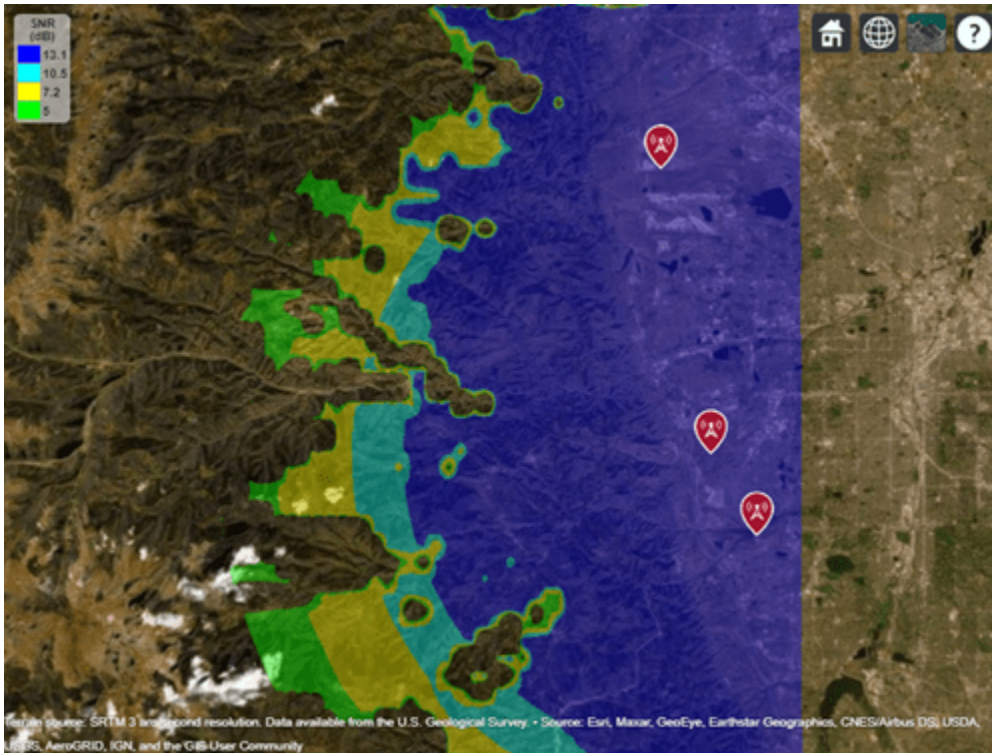
% Plot SNR thresholds vs number of pulses to integrate
plot(numpulses,snrthresholds,'-*')
title("SNR at Radar Receiver Required for Detection")
xlabel("Number of pulses to integrate")
ylabel("SNR (dB)")
grid on;
```



Show the radar coverage map for SNR thresholds corresponding to a few different numbers of pulses to integrate. Increasing the number of pulses to integrate decreases the required SNR and therefore produces a larger coverage region.

```
% Show best sites
viewer.Name = "Radar Coverage for Multiple SNR Thresholds";
show(rdrtxs(bestsitenums))

colors = jet(4);
colors(4,:) = [0 1 0];
contour(rdrData, ...
    "Levels",snrthresholds([1 2 5 10]), ...
    "Colors",colors, ...
    "LegendTitle",legendTitle)
```



Update Target Altitude

Update the scenario so that target positions are 250 meters above ground level instead of 500 meters above ground level. Rerun the same analysis as above to select the three best radar sites and visualize coverage. The new coverage map shows that reducing the visibility of the targets also decreases the coverage area.

```
% Target altitude above ground (m)
tgtalt = 250;
[L, ds] = helperPathlossOverTerrain(pm, rdrtxs, rdrrxs, tgtlats, tgtlons, tgtalt);

% Compute SNR for all radars and targets
numrdrs = numel(rdrtxs);
rawsnr = zeros(numtgs,numrdrs);
for tgtind = 1:numtgs
    for rdrind = 1:numrdrs
        rawsnr(tgtind,rdrind) = radareqsnr(lambda,ds(tgtind,rdrind),Ptx,pulsewidth, ...
            'Gain',antgain,'RCS',tgtcrs,'Loss',L(tgtind,rdrind));
    end
end

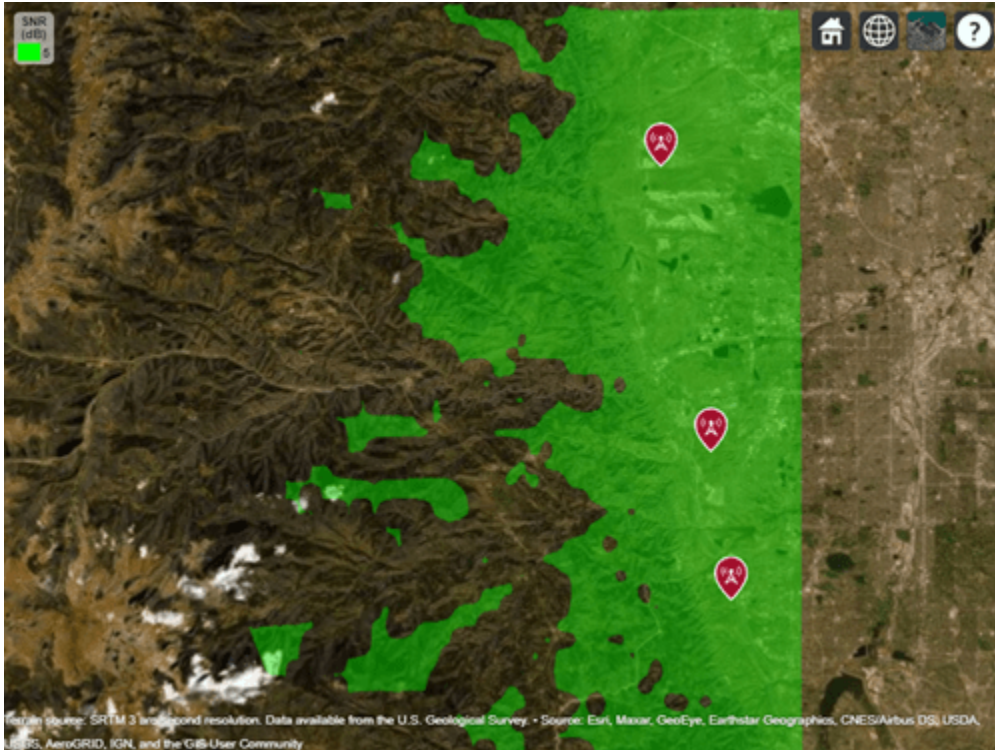
% Select best combination of 3 radar sites
bestsitenum = helperOptimizeRadarSites(rawsnr, snrthreshold);
snr = max(rawsnr(:,bestsitenum), [], 2);

% Show best sites
viewer.Name = "Radar Coverage";
clearMap(viewer);
show(rdrtxs(bestsitenum))
```

```

% Plot radar coverage
rdrData = propagationData(tgtlats,tgtlons,"SNR",snr);
contour(rdrData, ...
    "Levels",snrthreshold, ...
    "Colors","green", ...
    "LegendTitle",legendTitle)

```



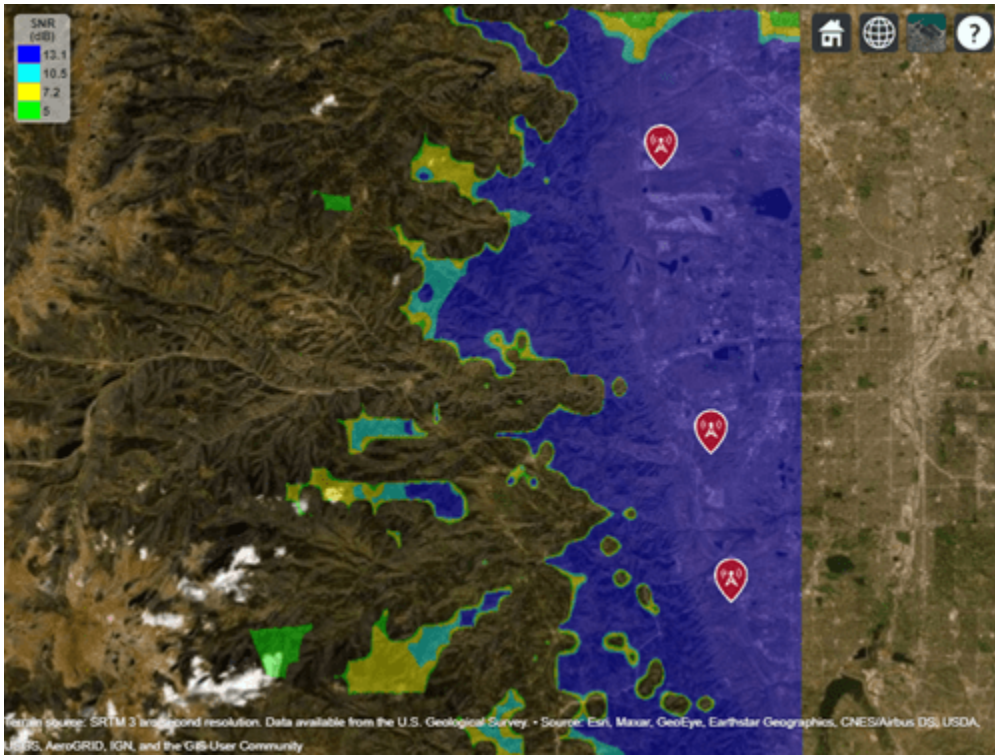
Show the radar coverage map for multiple SNR thresholds.

```

% Show best sites
viewer.Name = "Radar Coverage for Multiple SNR Thresholds";
show(rdrtxs(bestsitenums))

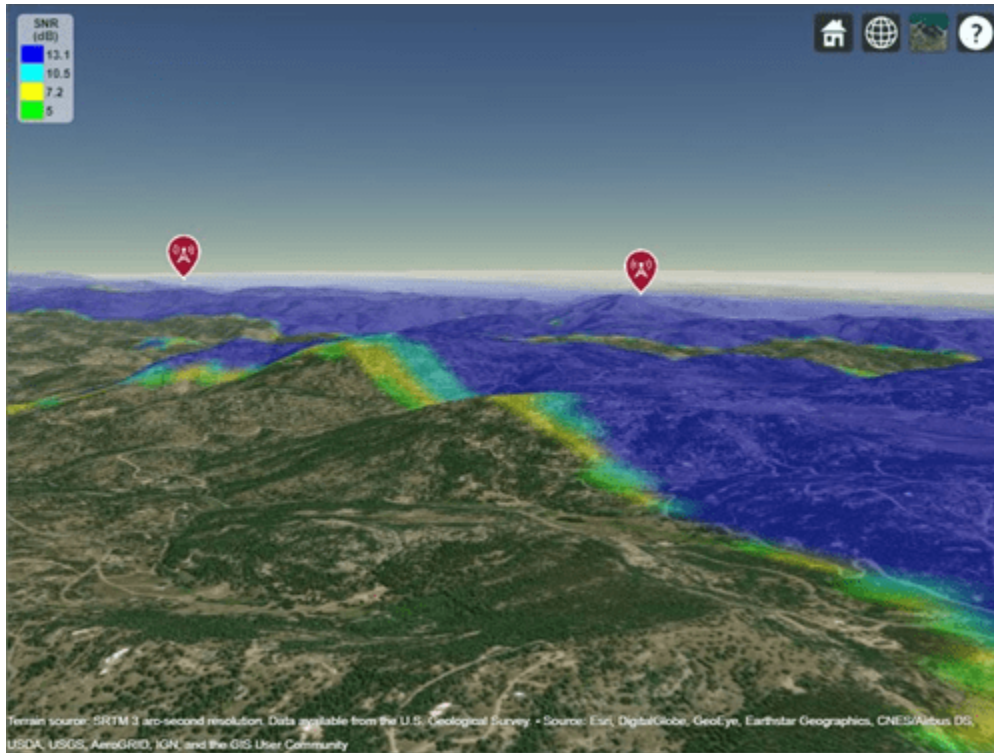
contour(rdrData, ...
    "Levels",snrthresholds([1 2 5 10]), ...
    "Colors",colors, ...
    "LegendTitle",legendTitle)

```



Conclusion

A monostatic radar system was designed to detect non-fluctuating targets with 0.1 square meter radar cross section (RCS) at a distance up to 35000 meters. Radar sites were selected among five candidate sites to optimize number of detections over a region of interest. Two target altitudes were considered: 500 meters above ground level and 250 meters above ground level. The coverage maps suggest the importance of line-of-sight visibility between the radar and target in order to achieve detection. The second scenario results in targets that are closer to ground level and therefore more likely to be blocked from line-of-sight visibility with a radar. This can be seen by rotating the map to view terrain, where non-coverage areas are typically located in shadow regions of the mountains.



Clean up by closing Site Viewer and removing the imported terrain data.

```
close(viewer)  
removeCustomTerrain("southboulder")
```

See Also

Functions

[addCustomTerrain](#) | [contour](#)

Objects

[siteviewer](#)

More About

- “Infinite Array Analysis” on page 5-58

Visualize Viewsheds and Coverage Maps Using Terrain

Display viewsheds and coverage maps for a region of interest using satellite imagery and terrain data. A *viewshed* includes areas that are within line of sight (LOS) of a set of points, and a *coverage map* includes received power for the areas within range of a set of wireless transmitters. You can use a viewshed as a first-order approximation of a coverage map.

By combining the mapping capabilities in Mapping Toolbox™ with the RF propagation capabilities in Antenna Toolbox™, you can create contoured coverage maps and use the contour data for both visualization and analysis. This example shows you how to:

- View imagery and terrain on 2-D maps
- View imagery, terrain, and viewsheds on 3-D relief maps
- View coverage maps in 2-D using contour plots and bar graphs
- Determine whether nearby points are within coverage
- View coverage maps for other regions of interest

View Imagery on 2-D Map

You can view imagery for a region of interest by plotting data into geographic axes. Geographic axes enable you to plot 2-D data over basemaps, explore regions interactively, and add data tips to points of interest.

Specify the geographic coordinates and heights of two cell towers in the San Fernando Valley region. Specify the heights using meters above the terrain.

```
lat1 = 34.076389;
lon1 = -118.468944;
towerHeight1 = 17.1;
text1 = "Tower 1";
```

```
lat2 = 34.386389;
lon2 = -118.329778;
towerHeight2 = 30.5;
text2 = "Tower 2";
```

Display the locations of the cell towers over a satellite basemap.

```
figure
basemap = "satellite";
geobasemap(basemap)

geoplot([lat1 lat2],[lon1 lon2], ...
       LineStyle="none",Marker="o",MarkerSize=10, ...
        MarkerEdgeColor="k",MarkerFaceColor="c")
```

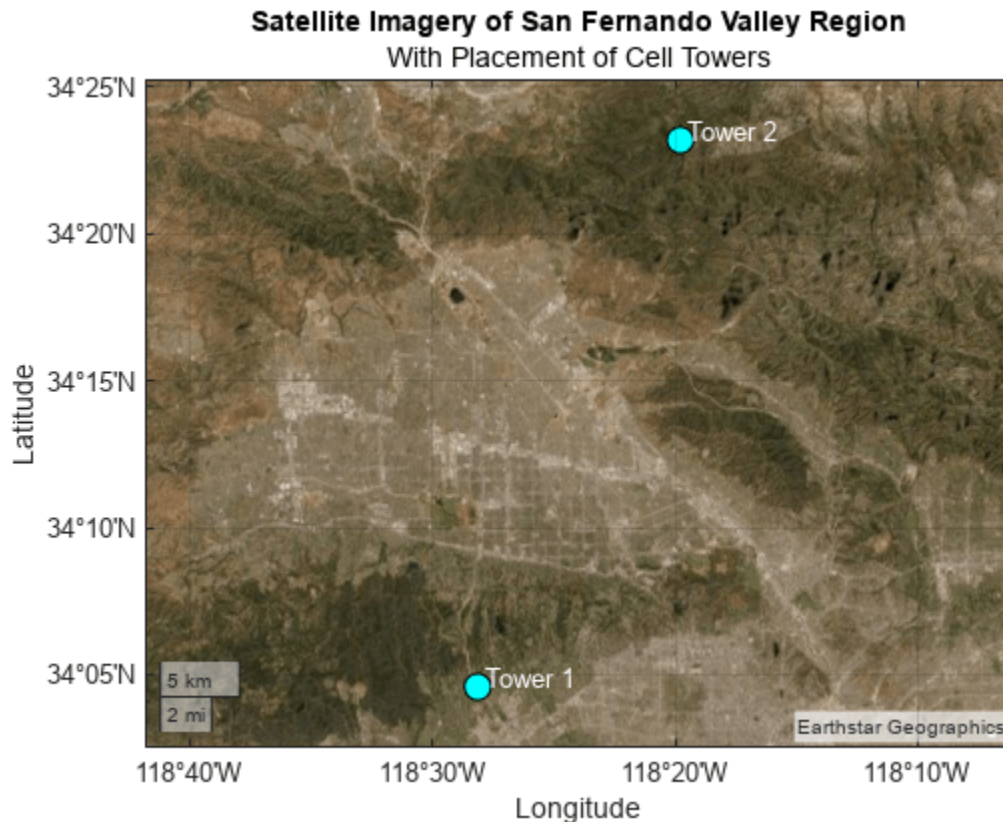
Prepare to create a 3-D relief map for the same region by querying the zoom level, latitude limits, and longitude limits of the geographic axes. To ensure that the relief map covers the same region, change the zoom level of the geographic axes to an integer.

```
gx = gca;
zoomLevel = floor(gx.ZoomLevel);
gx.ZoomLevel = zoomLevel;
```

```
[latlim,lonlim] = geolimits(gx);
```

Customize the map by labeling the cell towers and adding a title.

```
txtDelta = 0.005;
text(lat1 + txtDelta,lon1 + txtDelta,text1,Color="w")
text(lat2 + txtDelta,lon2 + txtDelta,text2,Color="w")
title("Satellite Imagery of San Fernando Valley Region")
subtitle("With Placement of Cell Towers")
```



View Terrain on 2-D Map

Visualize the terrain for the same region of interest by plotting imported elevation data into an axesm-based map. axesm-based maps enable you to display geographic data using a map projection.

Read Terrain Elevation Data

Read terrain elevation data for the region of interest from a Web Map Service (WMS) server. WMS servers provide geospatial data such as elevation, temperature, and weather from web-based sources.

Search the WMS Database for terrain elevation data hosted by MathWorks®. The `wmsfind` function returns search results as layer objects. A *layer* is a data set that contains a specific type of geographic information, in this case elevation data.

```
layer = wmsfind("mathworks",SearchField="serverurl");
layer = refine(layer,"elevation");
```

Read the elevation data from the layer into the workspace as an array and a spatial reference object in geographic coordinates. Get quantitative data by specifying the image format as BIL.

```
[Z,RZ] = wmsread(layer,Latlim=latlim,Lonlim=lonlim,ImageFormat="image/bil");
```

Prepare the elevation data for plotting by converting the data type to double.

```
Z = double(Z);
```

Find Tower Elevations

Calculate the terrain elevation at the base of each cell tower. The elevations are in meters above mean sea level.

```
height1 = geointerp(Z,RZ,lat1,lon1,"nearest")
```

```
height1 = 177
```

```
height2 = geointerp(Z,RZ,lat2,lon2,"nearest")
```

```
height2 = 1422
```

Visualize Elevation and Tower Locations

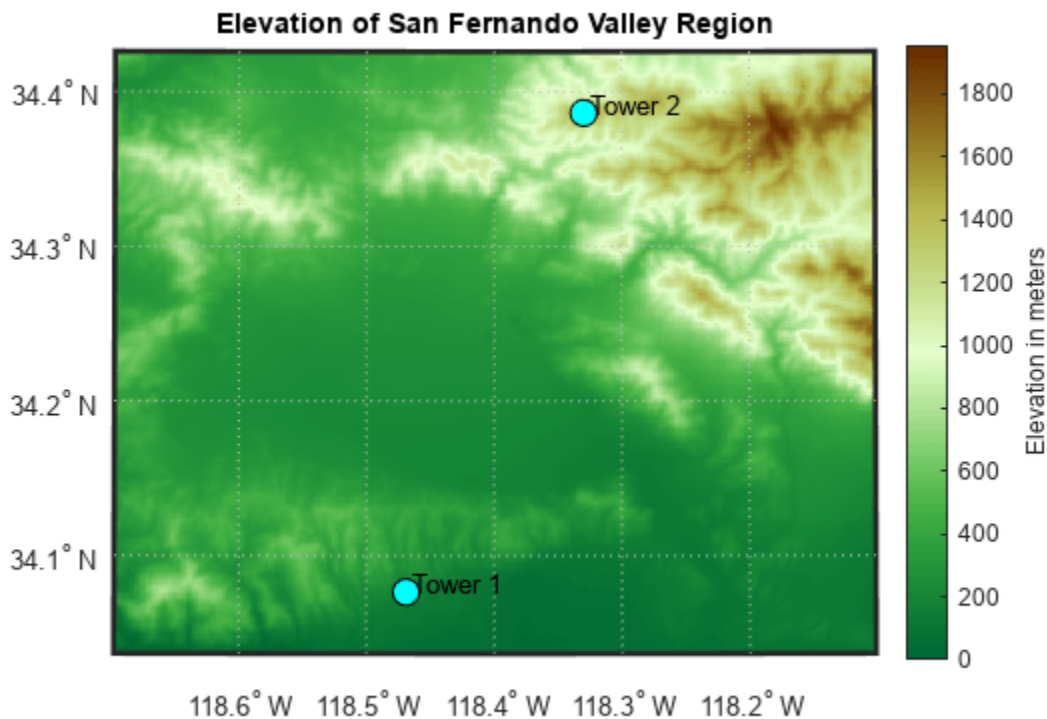
Create an axesm-based map with an appropriate projection for the region. Then, display the elevation data as a surface using an appropriate colormap. Note that the areas of highest elevation are in the northeast corner.

```
figure
usamap(Z,RZ)
geoshow(Z,RZ,DisplayType="texturemap")
demcmap(Z,256)

geoshow(lat1,lon1,DisplayType="point",ZData=height1, ...
        MarkerEdgeColor="k",MarkerFaceColor="c",MarkerSize=10,Marker="o")
geoshow(lat2,lon2,DisplayType="point",ZData=height2, ...
        MarkerEdgeColor="k",MarkerFaceColor="c",MarkerSize=10,Marker="o")

textm(lat1 + txtDelta,lon1 + txtDelta,text1)
textm(lat2 + txtDelta,lon2 + txtDelta,text2)
title("Elevation of San Fernando Valley Region")

cb = colorbar;
cb.Label.String = "Elevation in meters";
```



View Imagery, Terrain, and Viewshed on 3-D Relief Map

Visualize the imagery, terrain, and viewshed for the region by plotting the terrain data over a basemap image in a standard axes. Basemap images enable you to provide geographic context for plot types that geographic axes do not support, such as 3-D surfaces.

Read Basemap Image

Read a basemap image using the same basemap, geographic limits, and zoom level as the geographic axes. The `readBasemapImage` function reads the image into the workspace as an array, a spatial reference object in Web Mercator (WGS 84/Pseudo-Mercator) coordinates, and an attribution string.

```
[A,RA,attrib] = readBasemapImage(basemap,latlim,lonlim,zoomLevel);
```

To display the cell tower locations over the basemap image, you must first project the geographic coordinates to Web Mercator coordinates.

```
proj = RA.ProjectedCRS;
[x1,y1] = projfwd(proj,lat1,lon1);
[x2,y2] = projfwd(proj,lat2,lon2);
```

Plot the projected coordinates over the basemap image. Note that this visualization is similar to the geographic axes visualization.

```
figure
axis off
hold on
```

```

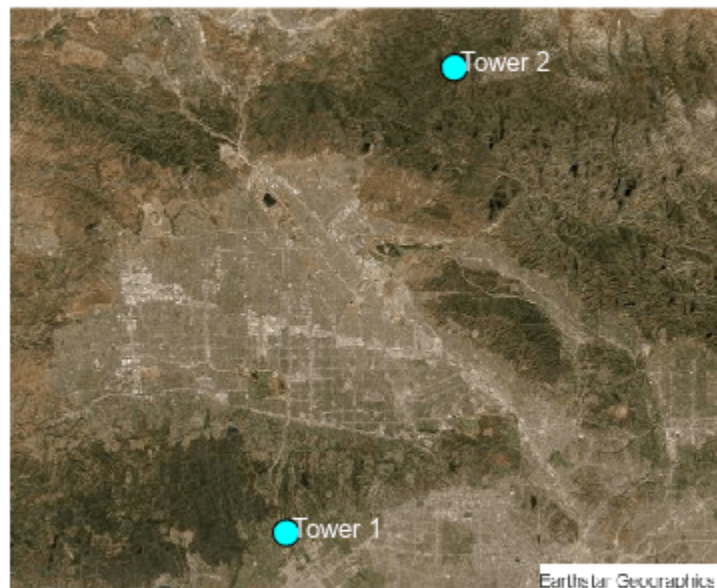
mapshow(A,RA)
plot([x1 x2],[y1 y2], ...
    LineStyle="none",Marker="o",MarkerSize=10, ...
     MarkerEdgeColor="k",MarkerFaceColor="c")

txtDeltaXY = 500;
text(x1 + txtDeltaXY,y1 + txtDeltaXY,text1,Color="w")
text(x2 + txtDeltaXY,y2 + txtDeltaXY,text2,Color="w")
title("Satellite Basemap Imagery of San Fernando Valley Region")
subtitle("Attribution: " + attrib)

```

Satellite Basemap Imagery of San Fernando Valley Region

Attribution: Earthstar Geographics



Calculate Viewsheds from Towers

Read the terrain data again, this time using array dimensions and geographic limits that match the dimensions and limits of the basemap image.

```

[latlim,lonlim] = projinv(proj,RA.XWorldLimits,RA.YWorldLimits);
szA = RA.RasterSize;

[Z,RZ] = wmsread(layer,Latlim=latlim,Lonlim=lonlim, ...
     ImageHeight=szA(1),ImageWidth=szA(2),ImageFormat="image/bil");
Z = double(Z);

```

Calculate the viewshed for each tower by specifying the terrain data and the cell tower coordinates. The first argument returned by each viewshed call indicates visible areas using 1 values and obscured areas using 0 values.

```
[vis1,visR1] = viewshed(Z,RZ,lat1,lon1,towerHeight1);
[vis2,visR2] = viewshed(Z,RZ,lat2,lon2,towerHeight2);
```

Calculate the combined viewshed for both towers. To avoid plotting the obscured areas, replace the 0 values with NaN values.

```
visCombined = vis1 | vis2;
vis = visCombined;
vis = double(vis);
vis(vis == 0) = NaN;
visR = visR2;
```

Extract the geographic coordinates of the elevation data from its spatial reference object. Then, project the coordinates to Web Mercator coordinates.

```
[latz,lonz] = geographicGrid(RZ);
[xz,yz] = projfwd(proj,latz,lonz);
```

View Individual and Combined Viewsheds

Display the viewshed for each tower and the combined viewshed for both towers in a tiled layout.

Define a colormap for the viewshed. Use turquoise for visible areas and black for obscured areas.

```
cmapt = [0 0 0];
covcolor = [0.1034 0.8960 0.7150];
cmapt(2, :, :) = covcolor;
```

Create a tiled layout. Display the viewshed for the individual towers on the left.

```
figure
clim([0 1])
colormap(cmapt)
tiledlayout(2,3,Padding="tight",TileSpacing="tight")

nexttile
geoshow(vis1,visR1,EdgeColor="none",DisplayType="surface")
axis off
title("Viewshed")
subtitle("Tower 1")

nexttile(4)
geoshow(vis2,visR2,EdgeColor="none",DisplayType="surface")
axis off
title("Viewshed")
subtitle("Tower 2")
```

Display the basemap image and the combined viewshed on the right. Specify the z-coordinates for the labels using the terrain height (above mean sea level) and the tower height (above the terrain).

```
nexttile([2,2])
mapshow(A,RA)
hold on
mapshow(xz,yz,vis,EdgeColor="none",DisplayType="surface")
axis off

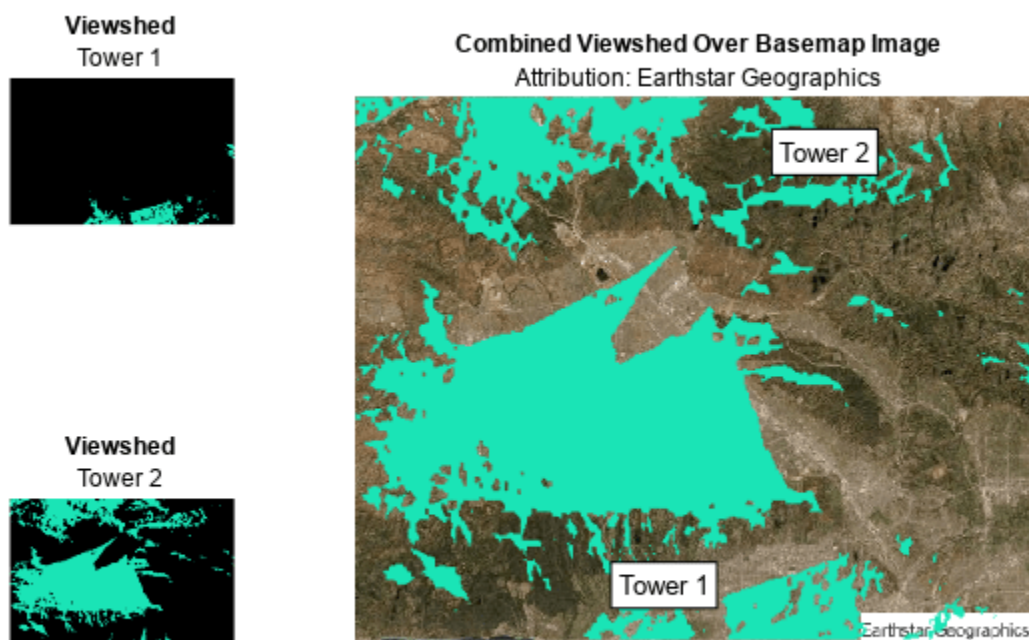
zdelta = 150;
z1 = height1 + towerHeight1 + zdelta;
```

```

z2 = height2 + towerHeight2 + zdelta;

text(x1,y1,z1,text1,Color="k",EdgeColor="k",BackgroundColor="w")
text(x2,y2,z2,text2,Color="k",EdgeColor="k",BackgroundColor="w")
title("Combined Viewshed Over Basemap Image")
subtitle("Attribution: " + attrib)

```



View Basemap Image and Viewshed on 3-D Relief Map

Prepare the combined viewshed for plotting.

- Find the indices of visible areas in the combined viewshed.
- Replace the visible areas in the combined viewshed with the terrain data (the corresponding values of Z).

```

[row,col] = find(visCombined);
sz = size(visCombined);
ind = sub2ind(sz,row,col);
vis(ind) = Z(ind);

```

Create a new figure that uses the same colormap as the tiled layout. Remove the axes lines and enable the plot to extend to the edges of the axes.

```

figure
colormap(covcolor)

```

```

hold on

```



```
axis off
axis tight
```

Visualize the basemap image, viewshed, and terrain using surfaces.

- Display the terrain by draping the basemap image over the elevation data. Specify the z-coordinates using the terrain data, and specify the surface colors using the image data.
- Display the combined viewshed.
- Customize the plot by adding text labels and a title.
- View the plot in 3-D. Adjust the data aspect ratio so that elevation scale is comparable to the xy-scale.

```
mapshow(xz,yz,Z,DisplayType="surface",CData=A)
```

```
mapshow(xz,yz,vis,EdgeColor="none",DisplayType="surface")
```

```
text(x1,y1,z1,text1,Color="k",EdgeColor="k",BackgroundColor="w")
```

```
text(x2,y2,z2,text2,Color="k",EdgeColor="k",BackgroundColor="w")
```

```
title("Satellite Basemap Imagery with Terrain and Viewshed")
```

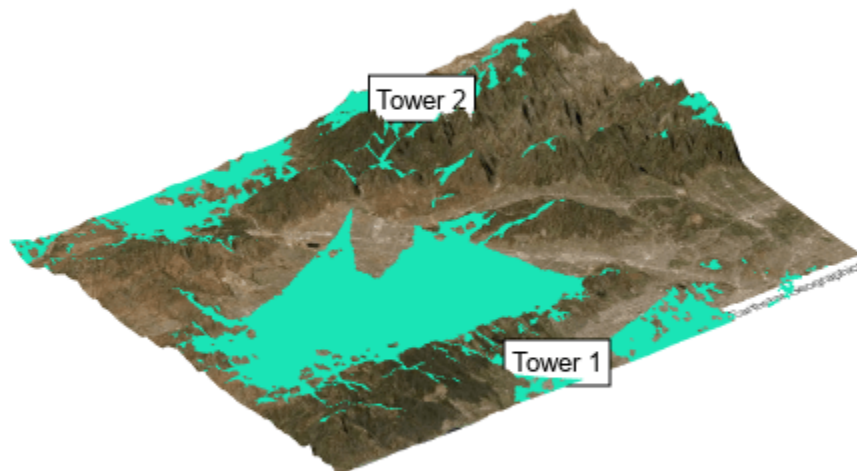
```
subtitle("Attribution: " + attrib)
```

```
daspect([1 1 0.25])
```

```
view(3)
```

Satellite Basemap Imagery with Terrain and Viewshed

Attribution: Earthstar Geographics



Visualize Coverage Maps in 2-D

A viewshed is comparable to a wireless coverage map that includes only line-of-sight (LOS) propagation. You can create more detailed coverage maps and calculate received power by using transmitter antenna sites and propagation models.

Calculate Coverage Map

Create a transmitter site for each cell tower. Then, calculate the coverage map and return the result as a `propagationData` object. By default, the `coverage` function uses a Longley-Rice propagation model and incorporates diffraction over terrain.

```
txTower1 = txsite(Name=text1,Latitude=lat1,Longitude=lon1);
txTower2 = txsite(Name=text2,Latitude=lat2,Longitude=lon2);
txsites = [txTower1; txTower2];
pd = coverage(txsites);
```

Extract latitude and longitude values, power values, and power levels from the `propagationData` object by using the `propagationDataGrid` on page 5-555 helper function. Find the limits of the quadrangle that bounds the coverage map.

```
dataVariableName = "Power";
maxrange = [30000 30000];
[lats,lons,data,levels] = propagationDataGrid(pd,dataVariableName,maxrange,txsites);
[covlatlim,covlonlim] = geoquadline(lats,lons);
```

Define a colormap using the power levels. Create a new figure that uses the power levels colormap.

```
cmap = turbo(length(levels));
climLevels = [min(levels) max(levels)+10];

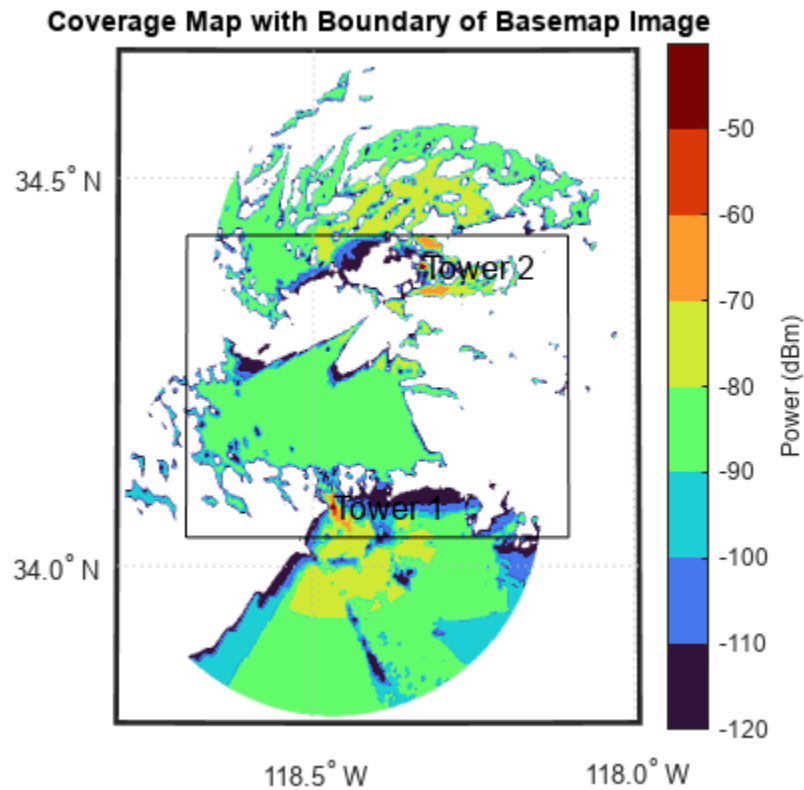
figure
colormap(cmap)
clim(climLevels)
hold on
```

Create an `axesm`-based map with an appropriate projection for the region. Display the coverage map, a rectangle that represents the San Fernando Valley region, and the transmitter sites on the map. Note that the coverage map extends beyond the San Fernando Valley region.

```
usamap(covlatlim,covlonlim)
geoshow(lats,lons,data,DisplayType="surface")
geoshow(latlim([1 2 2 1 1]),lonlim([1,1,2,2,1]),Color="k")

textm(lat1,lon1,text1,Color="k",FontSize=12)
textm(lat2,lon2,text2,Color="k",FontSize=12)
title("Coverage Map with Boundary of Basemap Image")

cb = colorbar;
cb.Label.String = "Power (dBm)";
cb.Ticks = levels;
```



View Contoured Coverage Map on Basemap Image

Project the geographic coordinates of the coverage map to Web Mercator coordinates.

```
[cx,cy] = projfwd(proj,lats,lons);
```

Create a figure that uses the power levels colormap.

```
figure
colormap(cmap)
clim(climLevels)
axis off
hold on
```

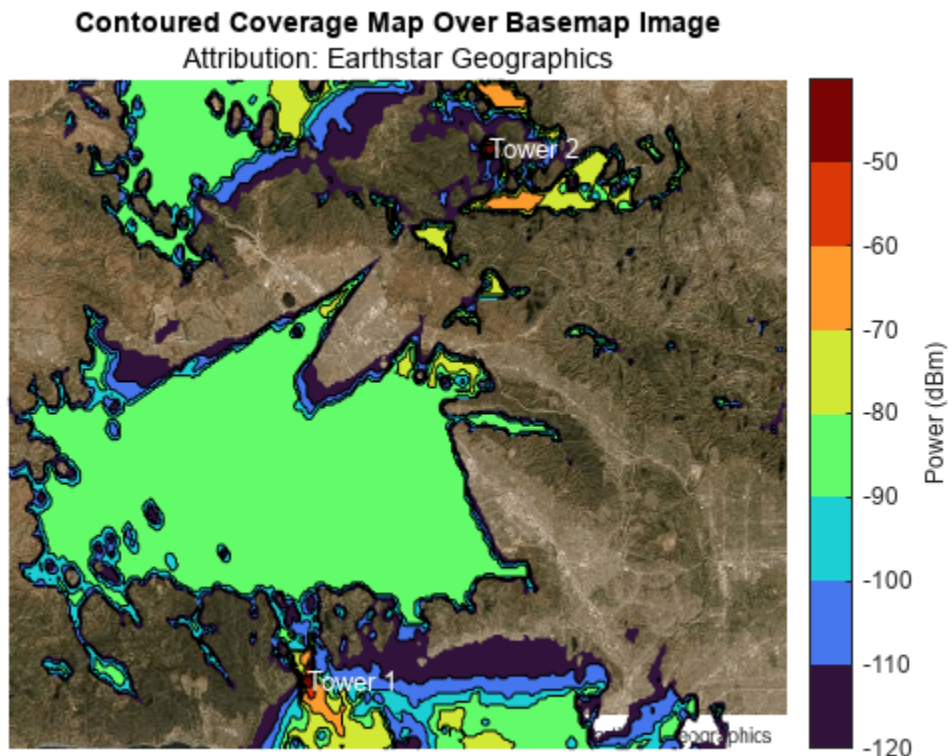
Contour the coverage map, specifying the contour levels as the power levels, and display the result over the basemap image. Change the limits of the map to match the San Fernando Valley region.

```
mapshow(A,RA)
contourf(cx,cy,data,LevelList=levels)

xlim(RA.XWorldLimits)
ylim(RA.YWorldLimits)

text(x1,y1,text1,Color="w")
text(x2,y2,text2,Color="w")
title("Contoured Coverage Map Over Basemap Image")
subtitle("Attribution: " + attrib)
```

```
cb = colorbar;
cb.Label.String = "Power (dBm)";
cb.Ticks = levels;
```



View Contoured Coverage Map and Viewshed on Basemap Image

Project the geographic coordinates of the combined viewshed to Web Mercator coordinates.

```
[vlat,vlon] = geographicGrid(visR);
[vx,vy] = projfwd(proj,vlat,vlon);
```

Display the coverage map and viewshed over the basemap image by using contour plots. You can create a map with two colorbars by overlaying two axes objects.

Use the first axes to display the basemap image, the coverage map, and a colorbar for the coverage map.

```
figure
ax1 = axes;
hold(ax1,"on")
axis(ax1,"off")
clim(ax1,climLevels)
colormap(ax1,cmap)

mapshow(ax1,A,RA)
contourf(ax1,cx,cy,data,LevelList=levels);

xlim(ax1,RA.XWorldLimits)
```

```
ylim(ax1,RA.YWorldLimits)
title(ax1,"Contoured Coverage Map and Viewshed on Basemap Image")
subtitle(ax1,"Attribution: " + attrib)
```

```
cb1 = colorbar(ax1);
cb1.Label.String = "Power (dBm)";
cb1.Ticks = levels;
```

Use the second axes to display the viewshed and a colorbar for the viewshed.

```
ax2 = axes(Visible="off");
hold(ax2,"on")
colormap(ax2,covcolor)
```

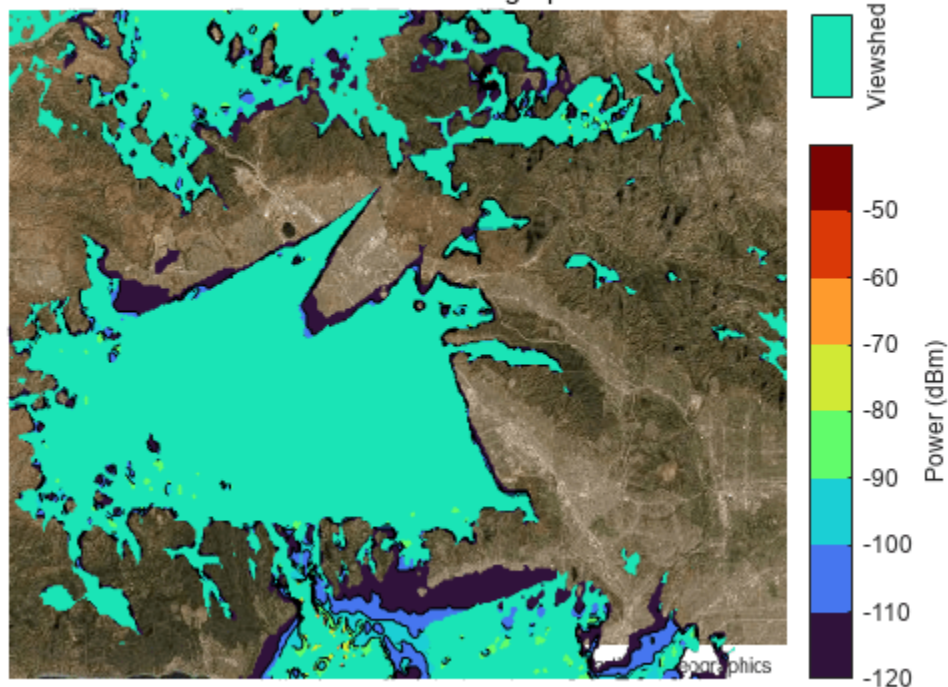
```
[~,visContourMap] = contourf(ax2,vx,vy,vis,LevelList=0.5:1:1.5);
```

```
cb2 = colorbar(ax2);
cb2.Label.String = "Viewshed";
cb2.Ticks = [];
```

Reposition the axes objects and the colorbars.

```
ax1.Position = [ax1.Position(1) ax1.Position(2)-0.04 ax1.Position(3) ax1.Position(4)];
ax2.Position = ax1.Position;
cb1.Position = [cb1.Position(1) cb1.Position(2) cb1.Position(3) cb1.Position(4)*0.8];
cb2.Position = [cb2.Position(1) 0.76 cb2.Position(3) 0.1];
```

Contoured Coverage Map and Viewshed on Basemap Image
Attribution: Earthstar Geographics



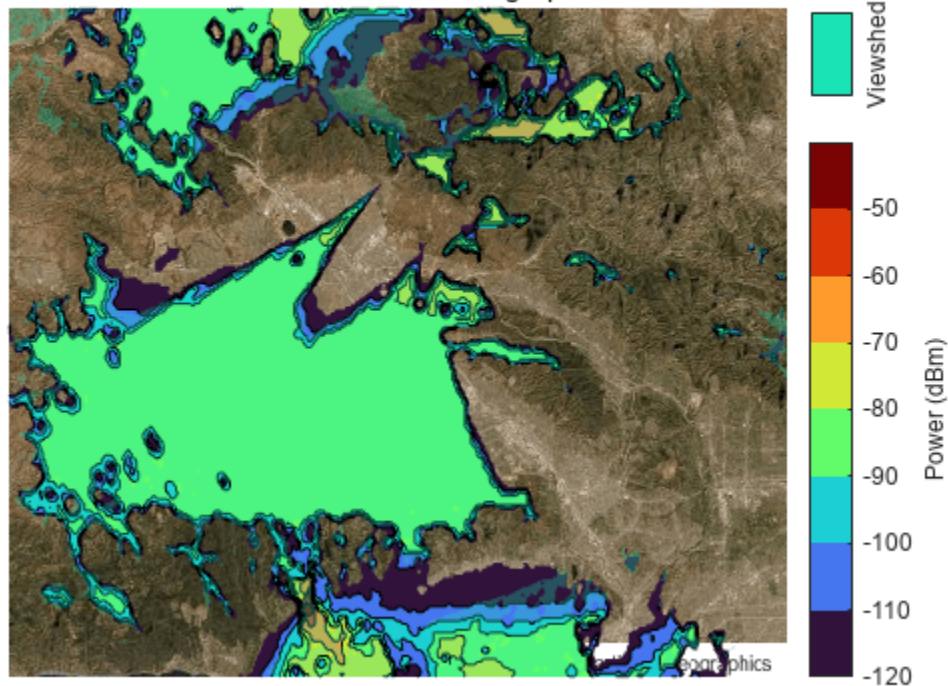
In addition to LOS analysis, the coverage map incorporates diffraction over terrain. As a result, the coverage map extends beyond the viewshed.

Adjust the transparency of the viewshed. When you decrease the value of `faceAlpha`, the viewshed becomes more transparent and you can see the coverage map beneath.

```
faceAlpha = 0.3  ;
visContourMap.FaceAlpha = faceAlpha;
```

Contoured Coverage Map and Viewshed on Basemap Image

Attribution: Earthstar Geographics



View Contoured Coverage Map on Geographic Axes

To display a contoured coverage map over geographic axes, contour the coverage map using the `contourDataGrid` on page 5-556 helper function. Each row of the returned geospatial table corresponds to a power level and includes the contour shape, the area of the contour shape in square kilometers, the minimum power for the level, and the power range for the level. The area value for each contour shape includes the regions that are covered by other contour shapes.

```
GT = contourDataGrid(lats,lons,data,levels);
disp(GT)
```

Shape	Area (sq km)	Power (dBm)	Power Range (dBm)	
geopolyshape	2.5889e+06	-120	-120	-110
geopolyshape	2.314e+06	-110	-110	-100
geopolyshape	2.1344e+06	-100	-100	-90

```

geopolyshape 1.7129e+06 -90 -90 -80
geopolyshape 3.0979e+05 -80 -80 -70
geopolyshape 13563 -70 -70 -60
geopolyshape 1894.9 -60 -60 -50
geopolyshape 676.29 -50 -50 -40

```

Plot the contoured coverage map and a rectangle that represents the San Fernando Valley region into geographic axes.

```

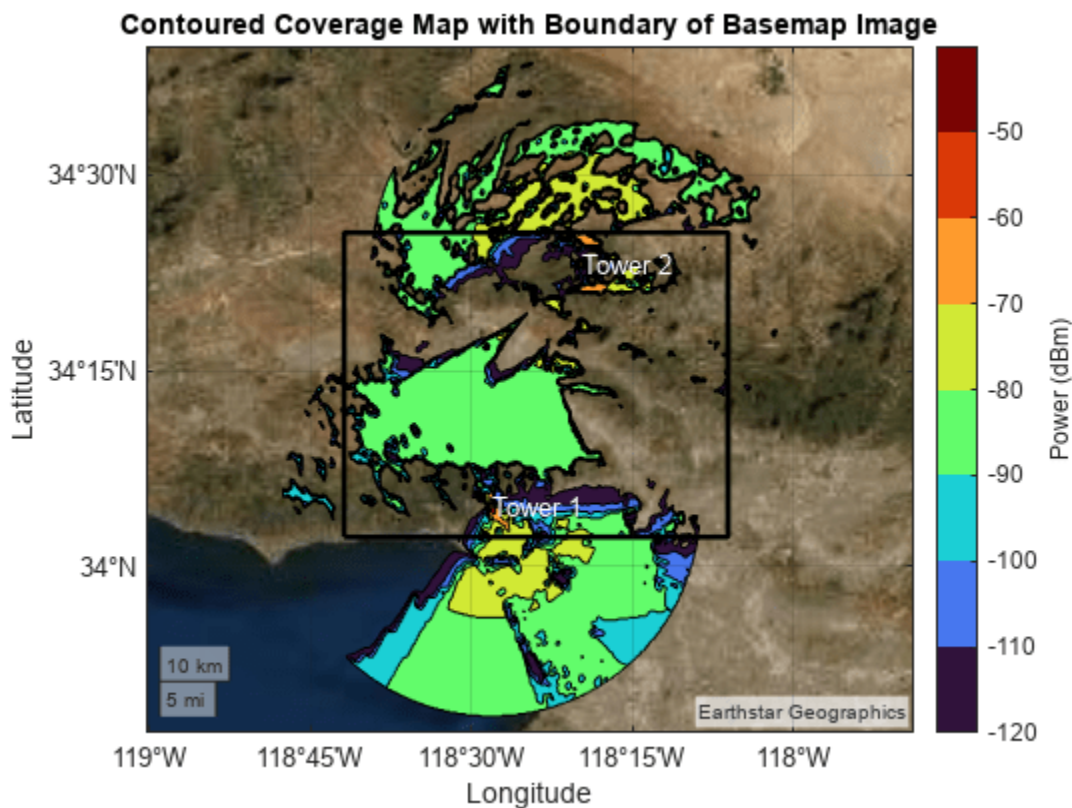
figure
geobasemap satellite
colormap(cmap)
clim(climLevels)
hold on

hp = geoplot(GT,ColorVariable="Power (dBm)",EdgeColor="k",FaceAlpha=1);
boundary = geoplot(latlim([1 2 2 1 1]),lonlim([1,1,2,2,1]),Color="k",LineWidth=2);

text(lat1,lon1,text1,Color="w")
text(lat2,lon2,text2,Color="w")
title("Contoured Coverage Map with Boundary of Basemap Image")

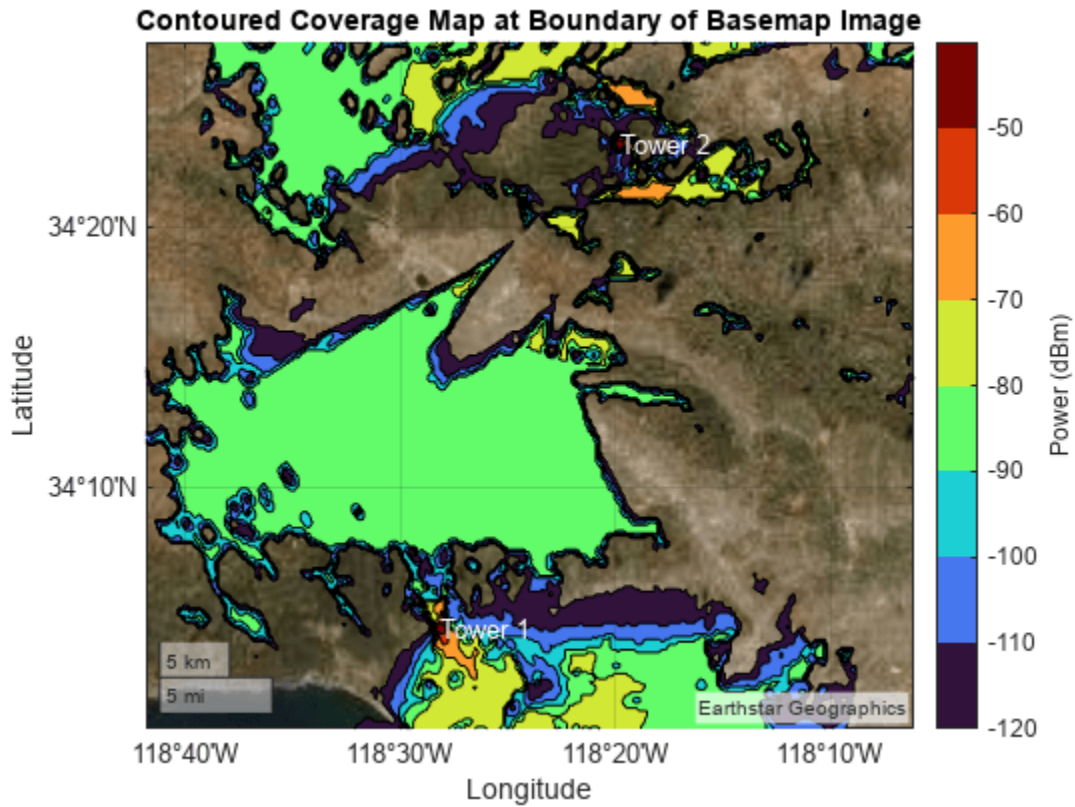
cb = colorbar;
cb.Label.String = "Power (dBm)";
cb.Ticks = levels;

```



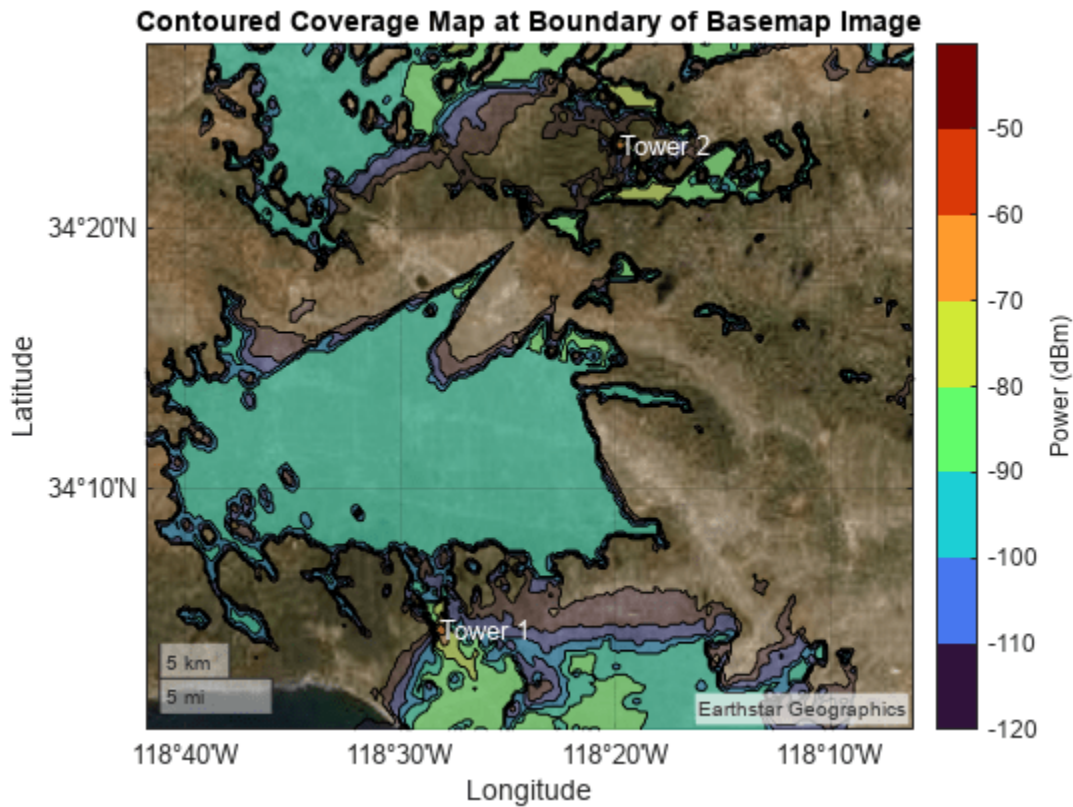
Zoom the map into the San Fernando Valley region and delete the boundary.

```
geolimits(latlim,lonlim)
delete(boundary)
title("Contoured Coverage Map at Boundary of Basemap Image")
```



Adjust the transparency of the coverage map. When you decrease the value of `faceAlpha`, the coverage map becomes more transparent and you can see the basemap beneath.

```
faceAlpha = 0.3  ;
hp.FaceAlpha = faceAlpha;
```

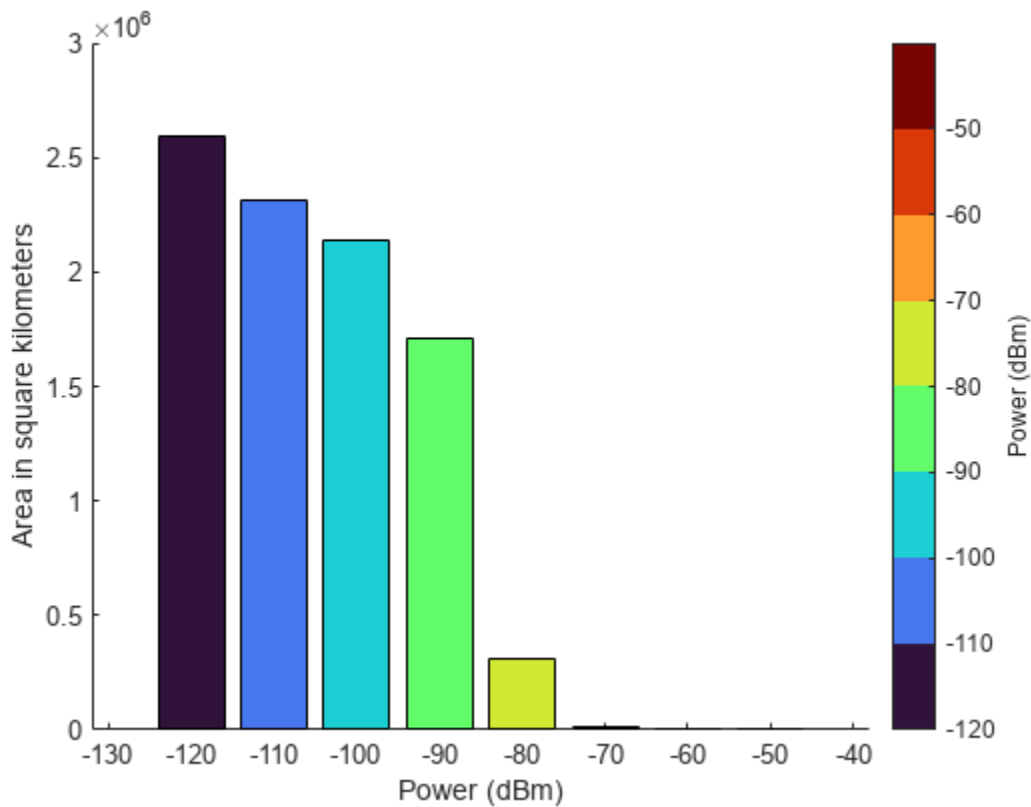
View Bar Graph of Coverage Area

Create a bar graph that shows the coverage area in square kilometers for each power level. As the power level increases, the coverage area decreases.

```
figure
colormap(cmap)
clim(climLevels)
hold on

bar(GT("Power (dBm)"),GT("Area (sq km)"),FaceColor="flat",CData=cmap);
ylabel("Area in square kilometers")
xlabel("Power (dBm)")

cb = colorbar;
cb.Label.String = "Power (dBm)";
cb.Ticks = levels;
```



Determine If Points Are Within Coverage

Determine whether points within the San Fernando Valley region are within coverage.

Calculate Coverage at Point Locations

Specify the locations of two points near the cell towers. Determine whether the points are within coverage by using the `incov` on page 5-559 helper function. The result indicates the first point is in coverage and the second point is not in coverage.

```
locationLats = [34.2107 34.1493];
locationLons = [-118.4545 -118.1744];

[tf,powerLevels] = incov(locationLats,locationLons,GT);
disp(tf)

1
0
```

Display the power levels. The result `-Inf` indicates that the second point is not in coverage.

```
disp(powerLevels)

-90
-Inf
```

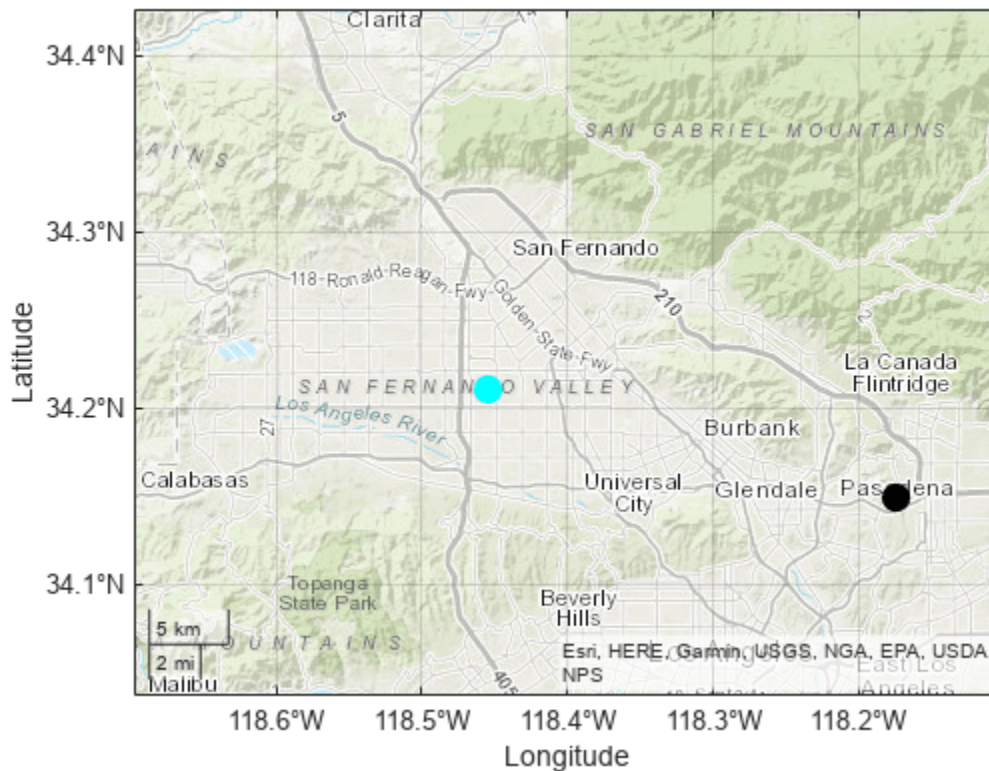
Display the point that is in coverage using a cyan marker and the point that is not in coverage using a black marker.

```

figure
geobasemap topographic
hold on
geotickformat dd
geolimits(latlim,lonlim)

geoplot(locationLats(tf),locationLons(tf),"o", ...
        MarkerEdgeColor="c",MarkerFaceColor="c",MarkerSize=10)
geoplot(locationLats(~tf),locationLons(~tf),"o", ...
        MarkerEdgeColor="k",MarkerFaceColor="k",MarkerSize=10)

```



Interactively Select Points and Calculate Coverage at Point Locations

Select points and determine whether they are in coverage.

Set up a map that uses the power levels colormap. Then, plot the contoured coverage map.

```

figure
geobasemap topographic
geolimits(latlim,lonlim)
geotickformat dd
hold on
colormap(cmap)
clim(climLevels)

contourPlot = geoplot(GT,ColorVariable="Power (dBm)",EdgeColor="k",FaceAlpha=1);

cb = colorbar;

```

```
cb.Label.String = "Power (dBm)";
cb.Ticks = levels(1:end);
```

Select the points. Use predefined points by specifying `interactivelySelectPoints` as `false`. Alternatively, you can interactively select points by specifying `interactivelySelectPoints` as `true`. Specify the number of points to interactively select by specifying a value for `numberOfPoints`.

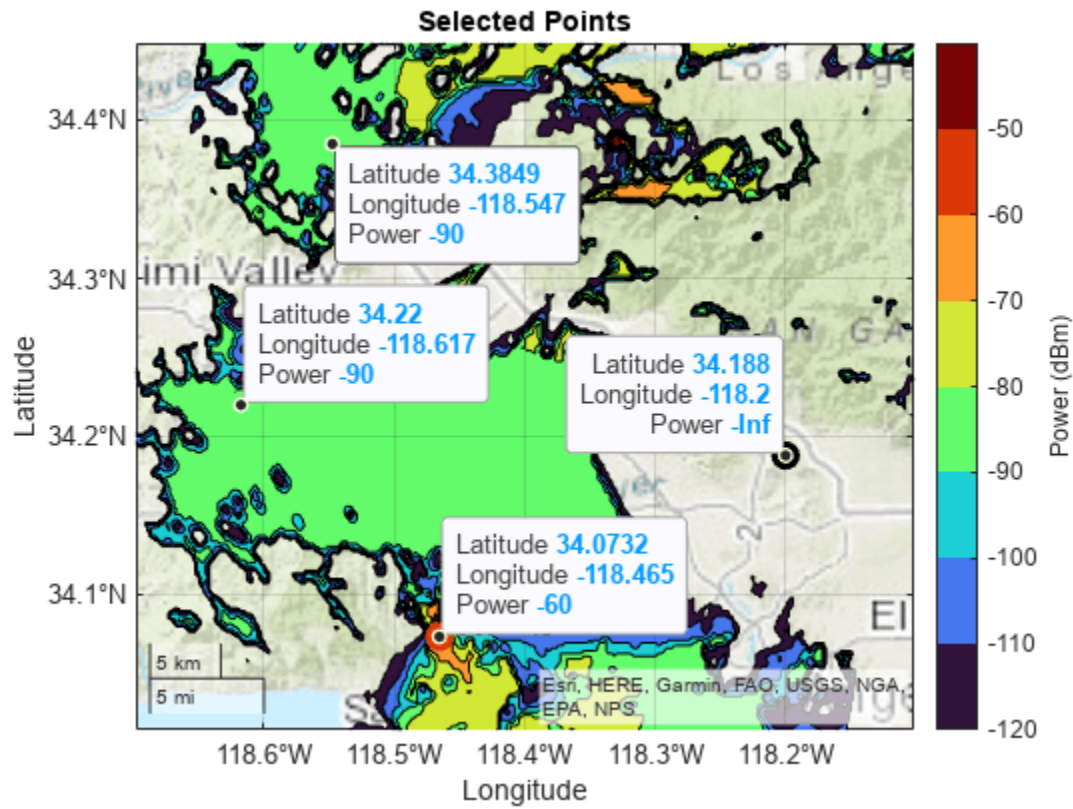
```
interactivelySelectPoints =  ;
if interactivelySelectPoints
    numberOfPoints =  ; %#ok<*UNRCH>
    title("Select " + string(numberOfPoints) + " Points")
    pts = ginput(numberOfPoints);
else
    pts = [ ...
        34.0732 -118.4652
        34.1880 -118.2000
        34.2200 -118.6172
        34.3849 -118.5472];
end
```

Determine whether each point is within coverage. If the point is not within coverage, display it on the map in black. If the point is within coverage, display it on the map using the color associated with its power level.

```
for k = 1:size(pts,1)
    latpoint = pts(k,1);
    lonpoint = pts(k,2);
    [tf,powerLevel] = incoverage(latpoint,lonpoint,GT);

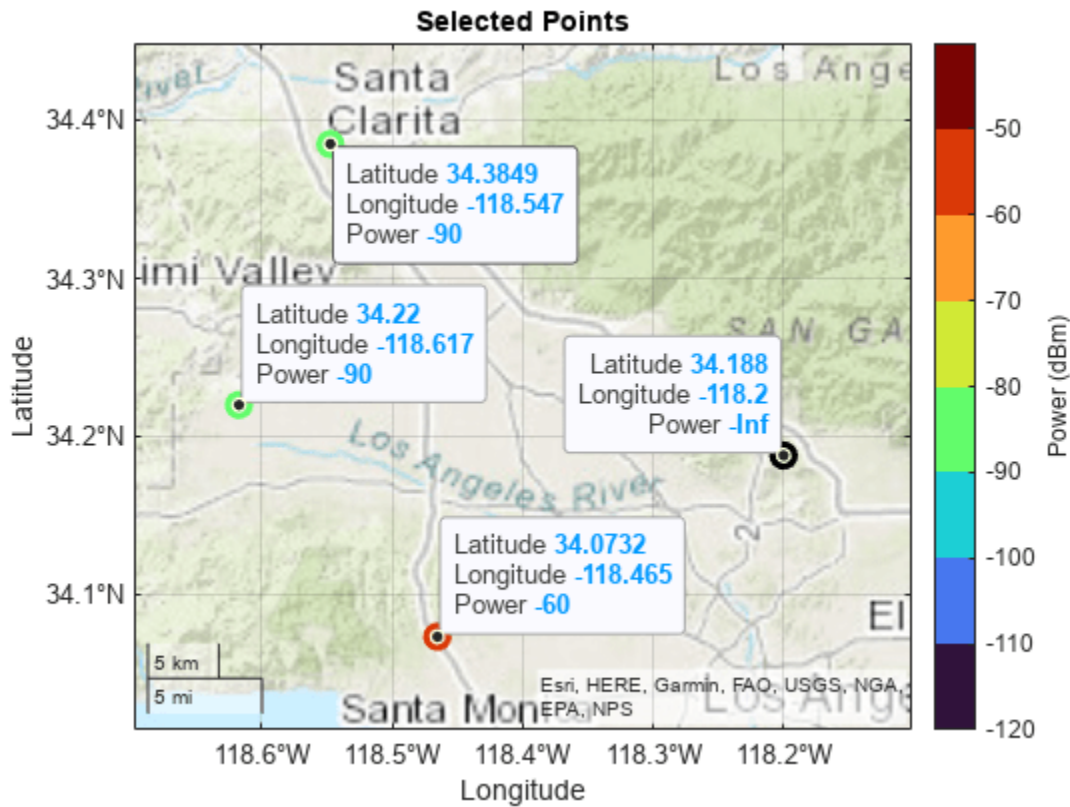
    if ~tf
        color = [0 0 0];
    else
        index = GT("Power (dBm)") == powerLevel;
        color = cmap(index,:,:);
    end

    h = geoplot(latpoint,lonpoint,"o",MarkerEdgeColor=color,MarkerFaceColor=color,MarkerSize=10)
    dt = datatip(h);
    dtrow = dataTipTextRow("Power",powerLevel);
    h.DataTipTemplate.DataTipRows(end+1) = dtrow;
end
title("Selected Points")
```



Toggle off the contour plot by setting `showContourPlot` to `false`. The color of each point indicates its power level.

```
showContourPlot =  ;
contourPlot.Visible = showContourPlot;
```



Create Geographic Coverage Maps for Other Regions

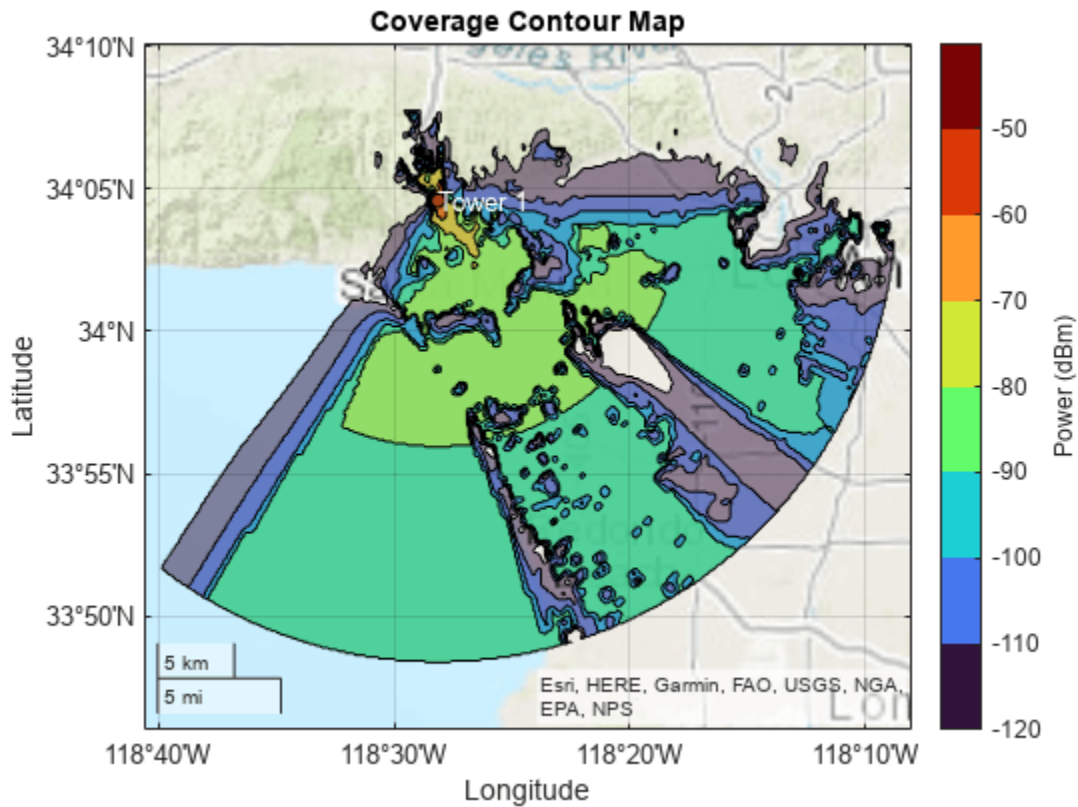
The `geocoverage` on page 5-559 helper function calculates or displays a coverage map for the specified transmitter sites. When you specify an output argument, the function returns a geospatial table.

```
T = geocoverage(txsites);
disp(T)
```

Shape	Area (sq km)	Power (dBm)	Power Range (dBm)	
geopolyshape	2.5889e+06	-120	-120	-110
geopolyshape	2.314e+06	-110	-110	-100
geopolyshape	2.1344e+06	-100	-100	-90
geopolyshape	1.7129e+06	-90	-90	-80
geopolyshape	3.0979e+05	-80	-80	-70
geopolyshape	13563	-70	-70	-60
geopolyshape	1894.9	-60	-60	-50
geopolyshape	676.29	-50	-50	-40

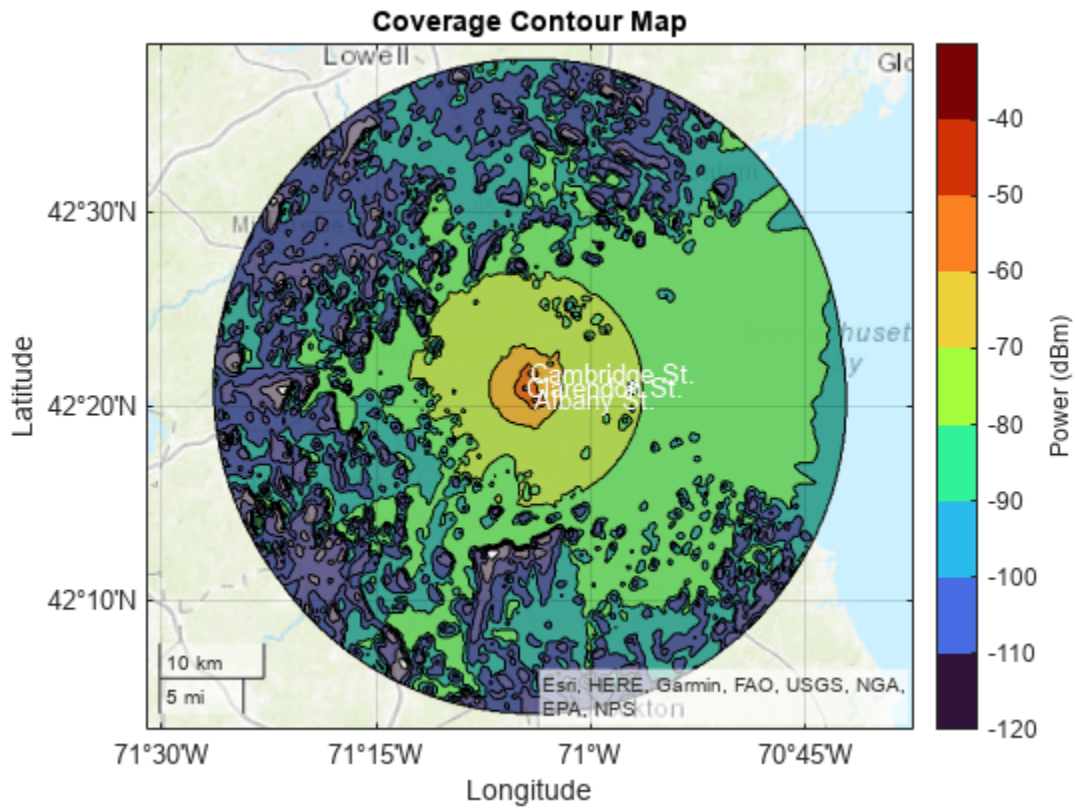
When you omit the output argument, the function contours the coverage map and displays the result over a basemap.

```
basemap = "topographic";
geocoverage(txsites(1),basemap)
```



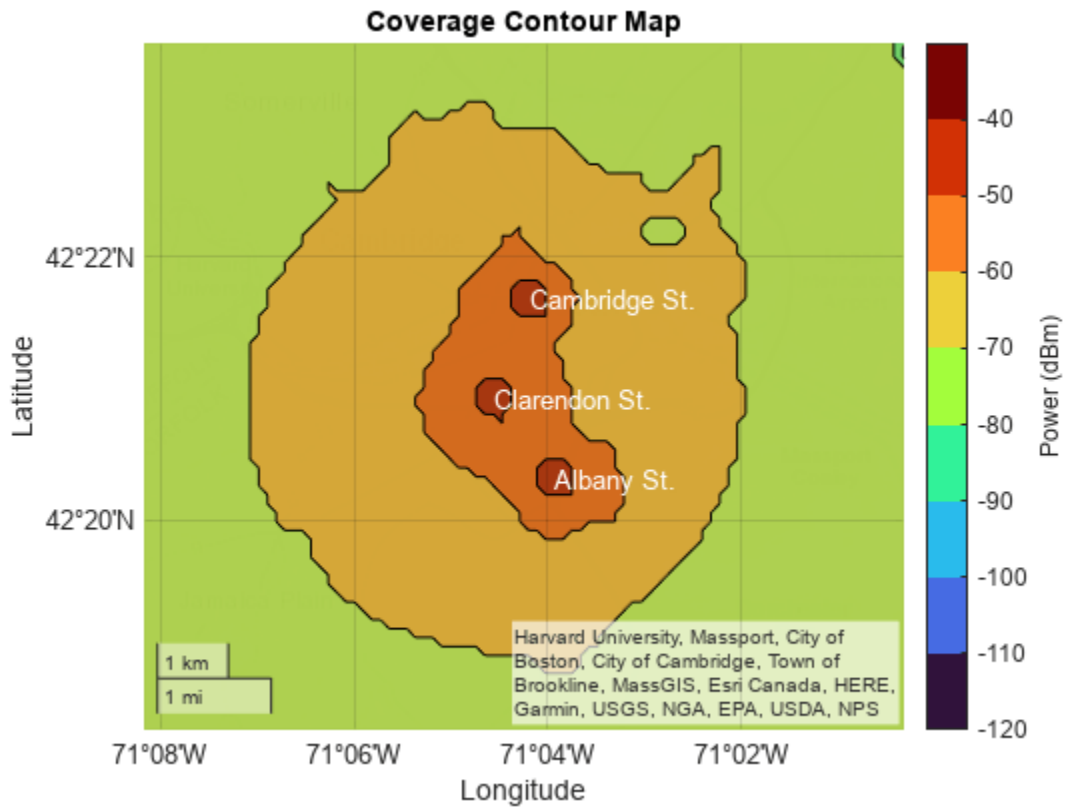
You can use the `geocoverage` helper function to create coverage maps for other locations. For example, create a coverage map for cell towers near Boston. The `geocoverage` function calculates coverage data for up to 30 kilometers from the transmitter sites.

```
names = ["Clarendon St.", "Cambridge St.", "Albany St."];
bostonLat = [42.348722 42.361222 42.338444];
bostonLon = [-71.075889 -71.069778 -71.065611];
bostonH = [260 30 23];
freq = [852.637e6 862.012e6 862.012e6];
txs = txsite(Name=names, Latitude=bostonLat, Longitude=bostonLon, ...
             AntennaHeight=bostonH, TransmitterFrequency=freq);
geocoverage(txs, basemap)
```



The cell towers are close together. View the area immediately around the towers by zooming in on the coverage map.

```
gx = gca;
gx.ZoomLevel = 12;
```

Helper Functions

The `propagationDataGrid` helper function extracts the latitude and longitude values `clats` and `clons`, the power values `cdata`, and the power levels `levels` from the propagation data object `pd` with data variable `dataVariableName`, using the transmitter sites `txsites` and the maximum range in meters from the transmitter sites `maxrange`. Within the function:

- Create regularly spaced grids of latitude and longitude values from the propagation data object.
- Determine which latitude and longitude values are within range of the transmitter sites.
- Create a grid of power values by interpolating the propagation data. Interpolate only the values that are within range of the transmitter sites.
- Determine the power levels by discretizing the data grid.

```
function [clats,clons,cdata,levels] = propagationDataGrid(pd,dataVariableName,maxrange,txsites)
% Extract propagation data grid from propagationData object

% Create grids of latitude and longitude values
imageSize = [512 512];
lats = pd.Data.Latitude;
lons = pd.Data.Longitude;
data = pd.Data.(dataVariableName);

[lonmin,lonmax] = bounds(lons);
[latmin,latmax] = bounds(lats);
```

```

lonsv = linspace(lonmin,lonmax,imageSize(2));
latsv = linspace(latmin,latmax,imageSize(1));
[clons,clats] = meshgrid(lonsv,latsv);

% Determine which latitude and longitude values are within range of the
% transmitters
txlat = [txsites.Latitude]';
txlon = [txsites.Longitude]';

numTx = size(txlat,1);
gc = nan(numel(clons),numTx);
for txInd = 1:numTx
    gc(:,txInd) = distance(txlat(txInd),txlon(txInd),clats(:),clons(:),wgs84Ellipsoid);
end
isInRange = any(gc <= maxrange,2);

% Create grid of power values
cdata = nan(imageSize);
cdata(isInRange) = interp(pd,clats(isInRange),clons(isInRange), ...
    DataVariableName=dataVariableName);

% Determine power levels
[bmin,bmax] = bounds(data);
levels = max(bmin,-120):10:bmax;
datalevels = sort(levels);
maxBin = max(max(datalevels(:)),max(cdata(:))) + 1; % Need max bin edge to include all values
bins = [datalevels(:); maxBin];
cdata = discretize(cdata,bins,datalevels);
end

```

The `contourDataGrid` helper function contours a data grid created using the `propagationDataGrid` helper function and returns the result as a geospatial table. Each row of the table corresponds to a power level and includes the contour shape, the area of the contour shape in square kilometers, the minimum power for the level, and the power range for the level. Within the function:

- Ensure that the function contours power values below the minimum power level by replacing NaN values in the power grid with a large negative number.
- Calculate accurate areas by projecting the latitude and longitude coordinates using an equal-area projection, in this case the North America Albers Equal Area Conic projection.
- Contour the projected coordinates using the `contourf` function. Specify the contour levels as the power levels. The `contourf` function returns a matrix that contains, for each power level, the coordinates of the contour lines.
- Prepare to analyze the data for each contour line by initializing several variables.
- Calculate the area within each contour line. Remove areas that correspond to holes and ignore negative areas. Unproject the coordinates.
- Create contour shapes from the unprojected coordinates. Include the contour shapes, the areas, and the power levels in a geospatial table.
- Condense the geospatial table into a new geospatial table with one row per power level. Each row contains one combined contour shape (that includes all contour shapes for the power level), the area of the combined contour shape, the minimum power for the level, and the power range for the level.

- Clean up the geospatial table. Convert the area of each shape to square kilometers, ensure that small contours appear on top of large contours by sorting the rows, and rename the table variables.

```
function GT = contourDataGrid(latd,lond,data,levels)
% Contour data grid created from propagationDataGrid helper function

% Replace NaN values in power grid with a large negative number
data(isnan(data)) = min(levels) - 1000;

% Project the coordinates using an equal-area projection
proj = projcrs(102008,"Authority","ESRI");
[xd,yd] = projfwd(proj,latd,lond);

% Contour the projected data
fig = figure("Visible","off");
obj = onCleanup(@()close(fig));
c = contourf(xd,yd,data,LevelList=levels);

% Initialize variables
x = c(1,:);
y = c(2,:);
n = length(y);
k = 1;
index = 1;
levels = zeros(n,1);
latc = cell(n,1);
lonc = cell(n,1);
Area = zeros(n,1);

% Calculate the area within each contour line. Remove areas that
% correspond to holes and ignore negative areas.
while k < n
    level = x(k);
    numVertices = y(k);
    yk = y(k+1:k+numVertices);
    xk = x(k+1:k+numVertices);
    k = k + numVertices + 1;

    [first,last] = findNanDelimitedParts(xk);
    jindex = 0;
    jy = {};
    jx = {};
    sumpart = 0;

    for j = 1:numel(first)
        % Process the j-th part of the k-th level
        s = first(j);
        e = last(j);
        cx = xk(s:e);
        cy = yk(s:e);
        if cx(end) ~= cx(1) && cy(end) ~= cy(1)
            cy(end+1) = cy(1); %#ok<*AGROW>
            cx(end+1) = cx(1);
        end

        if j == 1 && ~ispolycw(cx,cy)
```

```

        % First region must always be clockwise
        [cx,cy] = poly2cw(cx,cy);
    end

    jindex = jindex + 1;
    jy{jindex,1} = cy(:)';
    jx{jindex,1} = cx(:)';

    a = polyarea(cx,cy);
    if ~ispolycw(cx,cy)
        % Remove areas that correspond to holes
        a = -a;
    end

    sumpart = sumpart + a;
end

% Add a part when its area is greater than 0. Unproject the
% coordinates.
[jx,jy] = polyjoin(jx,jy);
if length(jy) > 2 && length(jx) > 2 && sumpart > 0
    [jlat,jlon] = projinv(proj,jx,jy);
    latc{index,1} = jlat(:)';
    lonc{index,1} = jlon(:)';
    levels(index,1) = level;
    Area(index,1) = sumpart;
    index = index + 1;
end
end

% Create contour shapes from the unprojected coordinates. Include the
% contour shapes, the areas, and the power levels in a geospatial
% table.
latc = latc(1:index-1);
lonc = lonc(1:index-1);
Shape = geopolyshape(latc,lonc);

Area = Area(1:index-1);

levels = levels(1:index-1);
Levels = levels;

allPartsGT = table(Shape,Area,Levels);

% Condense the geospatial table into a new geospatial table with one
% row per power level.
GT = table.empty;
levels = unique(allPartsGT.Levels);
for k = 1:length(levels)
    gtLevel = allPartsGT(allPartsGT.Levels == levels(k),:);
    tLevel = geotable2table(gtLevel,["Latitude","Longitude"]);
    [lon,lat] = polyjoin(tLevel.Longitude,tLevel.Latitude);
    Shape = geopolyshape(lat,lon);
    Levels = levels(k);
    Area = sum(gtLevel.Area);
    GT(end+1,:) = table(Shape,Area,Levels);
end

```

```

maxLevelDiff = max(abs(diff(GT.Levels)));
LevelRange = [GT.Levels GT.Levels + maxLevelDiff];
GT.LevelRange = LevelRange;

% Clean up the geospatial table
GT.Area = GT.Area/1000;
GT = sortrows(GT,"Area","descend");
GT.Properties.VariableNames = ...
    ["Shape","Area (sq km)","Power (dBm)","Power Range (dBm)"];
end

```

The `incoverage` helper function determines whether the points with latitude values `lat` and longitude values `lon` are within the coverage map defined by the geospatial table `T`. Within the function:

- Determine whether the points are within the coverage map and return the results in `in`. The elements of `in` are 1 (true) when the corresponding points are within coverage.
- Determine the power levels for the points and return the results in `powerLevels`. A value of `-Inf` indicates that the corresponding point is not within coverage.

```

function [in,powerLevels] = incoverage(lat,lon,T)
% Query points in coverage

% Determine whether points are within coverage
tf = false(length(T.Shape),length(lat));
for k = 1:length(T.Shape)
    tf(k,:) = isinterior(T.Shape(k),geopointshape(lat,lon));
end

% Determine the power levels for the points
in = false(length(lat),1);
powerLevels = -inf(length(lat),1);
for k = 1:length(in)
    v = tf(:,k);
    in(k) = any(v);
    if in(k)
        powerLevels(k) = max(T{v,"Power (dBm)"});
    end
end
end

```

The `geocoverage` helper function calculates or displays a coverage map for the transmitter sites `txsites`. Within the function:

- Calculate the coverage map by using the `coverage` function. Then, contour the coverage map by using the `proagationDataGrid` and `contourDataGrid` helper functions.
- When you do not specify an output argument, the function displays the contoured coverage map over geographic axes using the `basemap` `basemap`.
- When you do specify an output argument, the function returns the result as a geospatial table.

```

function varargout = geocoverage(txsites,basemap)
% Display or calculate contoured coverage map

```

```
% Calculate and contour the coverage map
pd = coverage(txsites);
dataVariableName = "Power";
maxrange = 30000*ones(1,length(txsites));
[lats,lons,data,levels] = propagationDataGrid(pd,dataVariableName,maxrange,txsites);
GT = contourDataGrid(lats,lons,data,levels);

% If you do not specify an output argument, display the coverage map
if nargin == 0
    if nargin < 2
        basemap = "satellite";
    end

    cmap = turbo(length(levels));
    figure
    geobasemap(basemap)

    geoplot(GT,ColorVariable="Power (dBm)",EdgeColor="k",FaceAlpha=0.5)
    colormap(cmap)
    clim([min(levels) max(levels)+10])

    h = colorbar;
    h.Label.String = "Power (dBm)";
    h.Ticks = levels;

    for k = 1:length(txsites)
        latt = txsites(k).Latitude;
        lont = txsites(k).Longitude;
        text(latt,lont,txsites(k).Name,Color="w")
    end

    title("Coverage Contour Map")
% If you specify an output argument, return a geospatial table
else
    varargout{1} = GT;
end
end
```

The `findNaNDelimitedParts` helper function finds the indices of the first and last elements of each NaN-separated part of the array `x`. The `contourDataGrid` helper function uses `findNaNDelimitedParts`.

```
function [first,last] = findNaNDelimitedParts(x)
% Find indices of the first and last elements of each part in x.
% x can contain runs of multiple NaNs, and x can start or end with
% one or more NaNs.

n = isnan(x(:));

firstOrPrecededByNaN = [true; n(1:end-1)];
first = find(~n & firstOrPrecededByNaN);

lastOrFollowedByNaN = [n(2:end); true];
```

```
    last = find(~n & lastOrFollowedByNaN);  
end
```

See Also

Functions

viewshed | contourf | coverage | geoplot | readBasemapImage

Objects

txsite | LongleyRice

Related Examples

- “Create Geospatial Tables” (Mapping Toolbox)
- “Visualize Antenna Coverage Map and Communication Links” on page 5-309

Subarrays in Large Finite Array For Hybrid Beamforming

This example shows how to design subarrays in large finite array for hybrid beam forming. Hybrid beamforming combines analog beamforming with digital precoding to intelligently form the patterns transmitted from a large antenna array.

Array Parameters

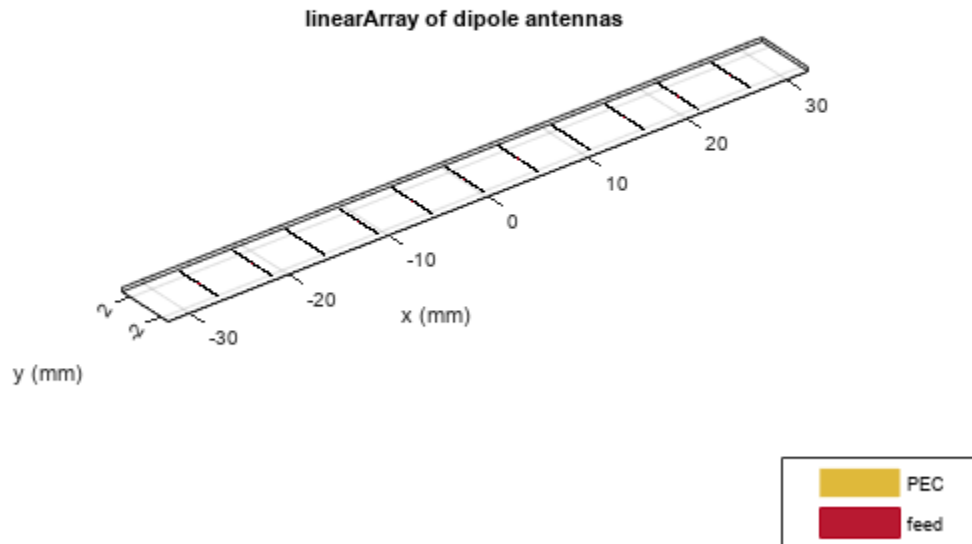
Choose number of elements, frequency of operation and the azimuth and elevation angle to steer the main beam of the array.

```
N = 11;  
fc = 28e9;  
az = 30;  
el = 20;
```

Find Phase Shifts for Azimuth Control

Design a linear array at the desired frequency. The default element is a dipole. Find the phase shifts to apply on each element of the linear array for controlling the main beam in the azimuthal direction. Note, that the distance of separation is chosen to be half-wavelength to ensure no grating lobes.

```
l = design(linearArray,fc);  
elem = l.Element;  
elem.Tilt = 90;  
l.NumElements = N;  
figure  
show(l)
```

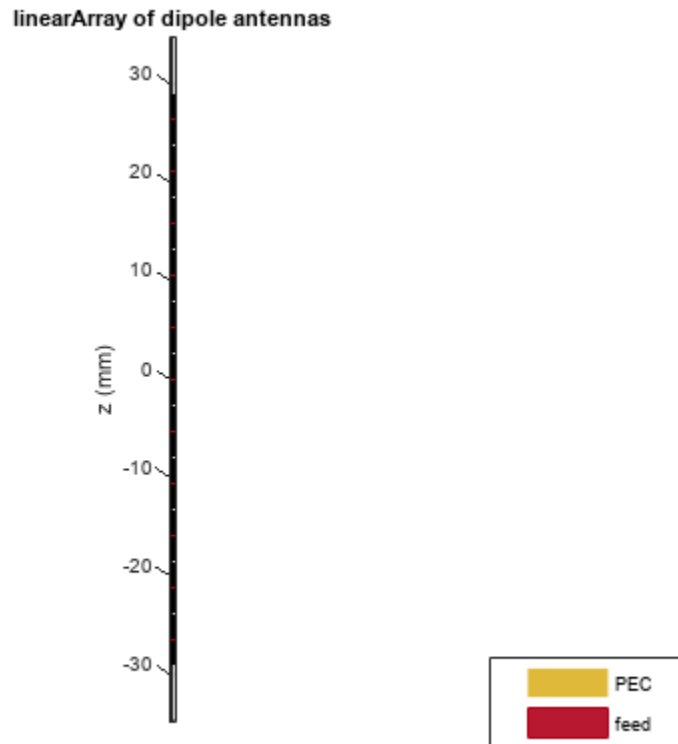



```
ps_az = phaseShift(l,fc,[az;0]);
```

Find Phase Shifts for Elevation Control

Compute the phase shifts for the steering in elevation. To do this, we modify the geometry of our initial linear array for the echelon configuration along z .

```
elem.Tilt = 90;
elem.TiltAxis = [0 1 0];
l.Tilt = 90;
l.TiltAxis = [0 1 0];
l.ElementSpacing = 1.05*(elem.Length) ;
figure
show(l)
```

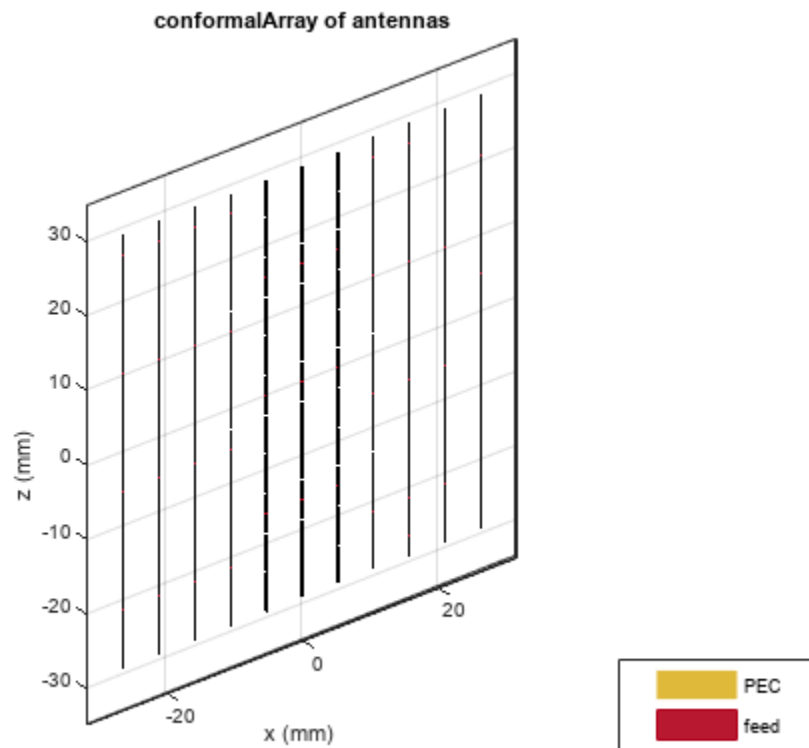


```
ps_el = phaseShift(l,fc,[0;el]);
```

Create Subarrays

Create a $N \times N$ rectangular array comprising of N , $1 \times N$ linear arrays stacked along the positive and negative z -directions.

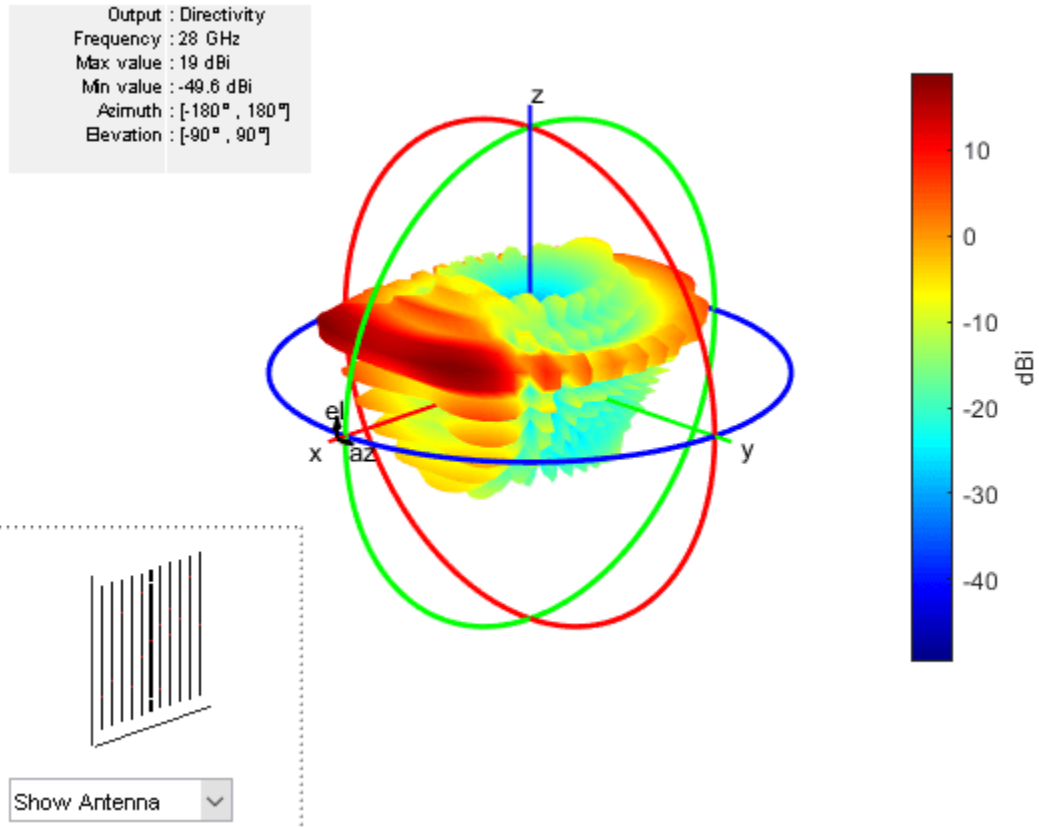
```
l.Tilt = 0;
elem.Tilt = 0;
l.PhaseShift = ps_az;
c = conformalArray;
zposn = fliplr((-N+1)/2:1:(N-1)/2);
for i = 1:N
    c.Element{i} = l;
    c.ElementPosition(i,:) = [0,0,zposn(i)*l.ElementSpacing];
end
figure
show(c)
```



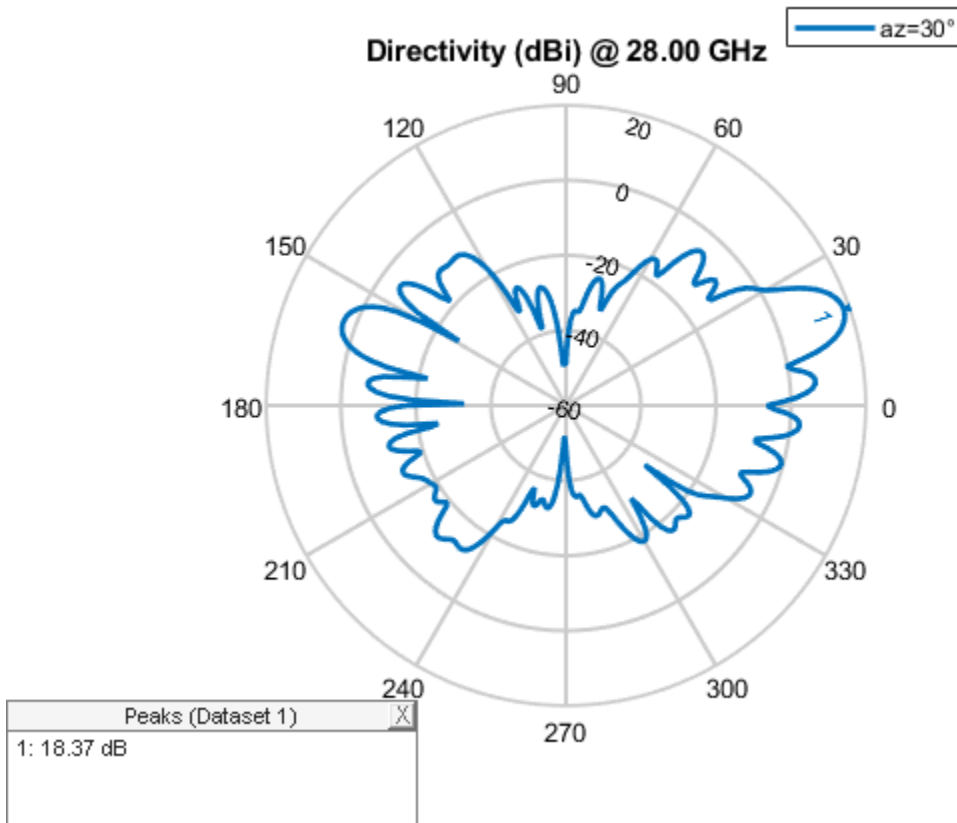
Assign Phase Shifts and Plot Pattern

Assign the sub-array level phase shifts and compute pattern

```
c.PhaseShift = ps_el;  
figure  
pattern(c,fc);
```



```
figure
patternElevation(c,fc,az);
```

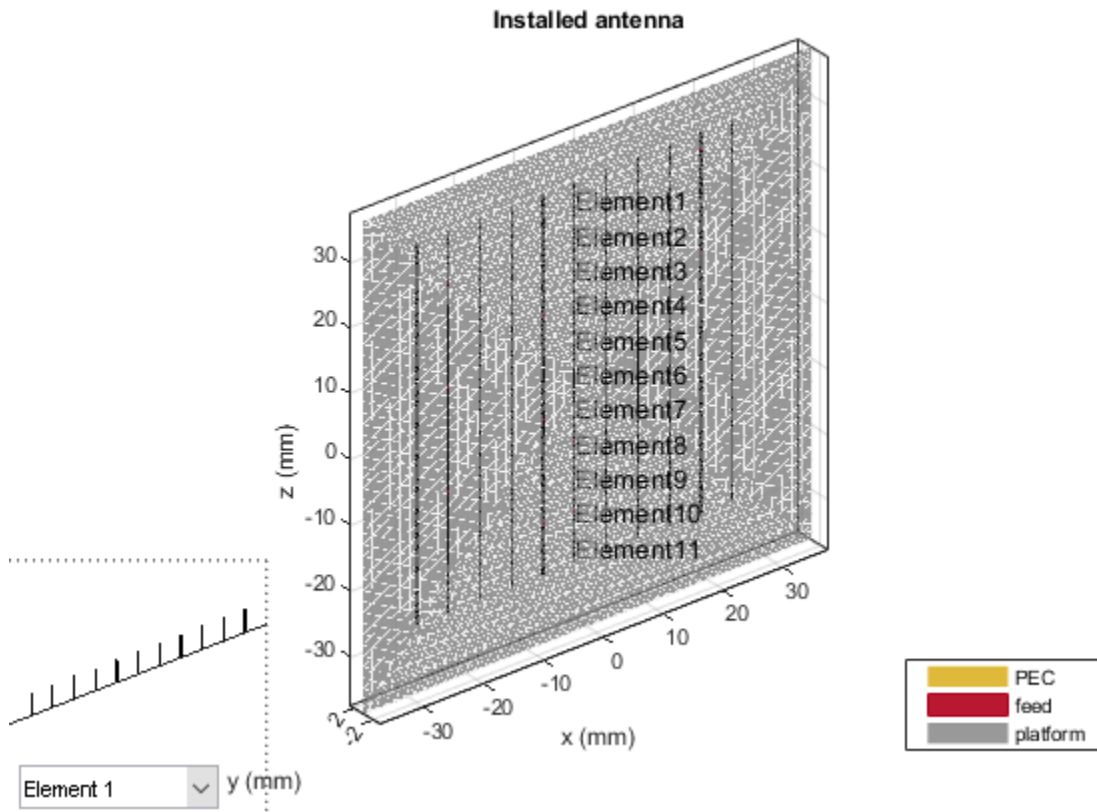


Note the actual peak location varies from the theoretical computed due to mutual coupling

Array with Large Reflector Backing

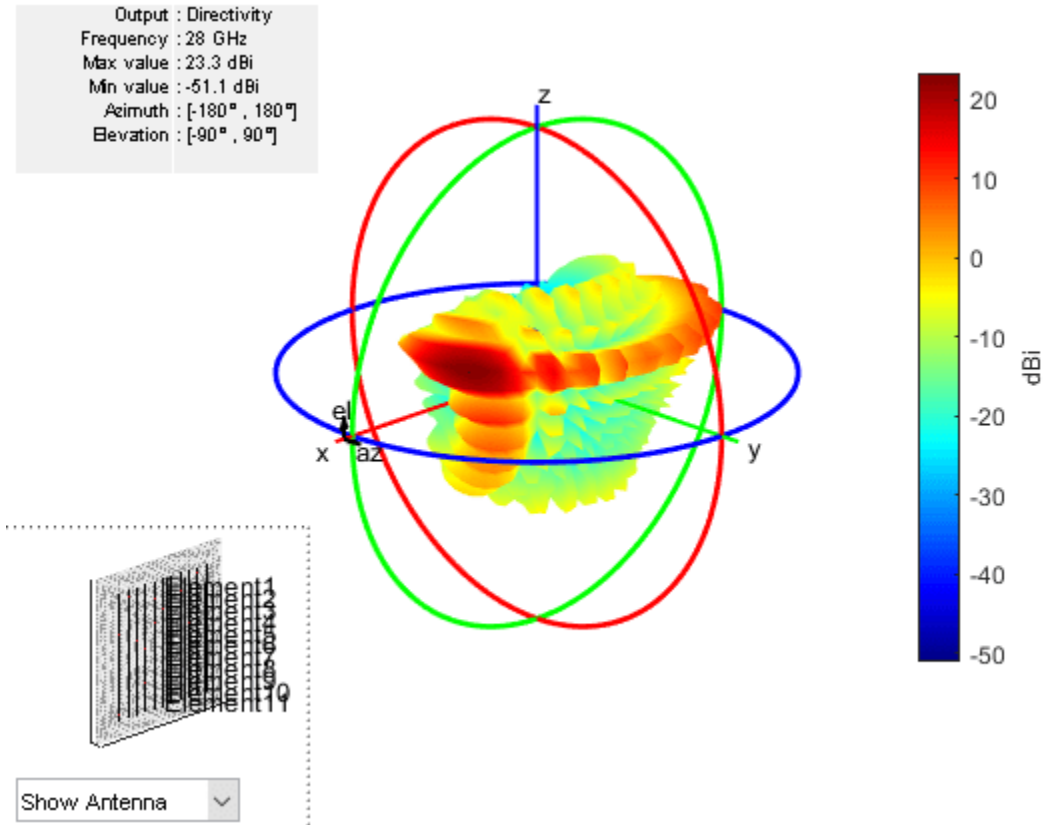
Using the installed antenna capability, allows for an initial approximate analysis of the antenna array by including a large structure in its vicinity. For this example provide an STL file of a large metallic reflector positioned a quarter-wavelength away from the array. The analysis treats the array using a full-wave Method of Moments (MoM) approach and the large reflector is handled using the Physical Optics (PO) approximation.

```
lambda = physconst('lightspeed')/fc;
ref_offset = lambda/4;
p = platform;
p.FileName = 'GroundPlane.stl';
p.Units = 'm';
p.Tilt = 90;
f = installedAntenna;
f.Platform = p;
f.Element = c.Element;
f.ElementPosition = c.ElementPosition;
f.ElementPosition(:,2) = ref_offset;
f.FeedPhase = ps_el;
figure
show(f)
```



Approximate Array Pattern

figure
pattern(f,fc)



See Also

“Antenna Array Beam Scanning Visualization on a Map” on page 5-335

Pattern Analysis of the Symmetric Parabolic Reflector

This example investigates the effect of feed position and reflector surface geometry on the far-field radiation pattern of a half-wavelength dipole-fed symmetric parabolic reflector.

The symmetric parabolic reflector also commonly referred to as a 'dish' is a simple and widely used high gain antenna. These antennas are commonly used for satellite communications, in both civilian and military applications. The high gain of these antennas is achieved due to the electrical size of the antenna, also referred to as the aperture. The symmetric parabolic reflector has a circular aperture and its electrical size is typically reported in terms of the diameter. Depending on the application the diameter of the reflector could range from 10-30 λ (VSAT terminals), or upwards of 100 λ (radio astronomy).

Reflector Parameters

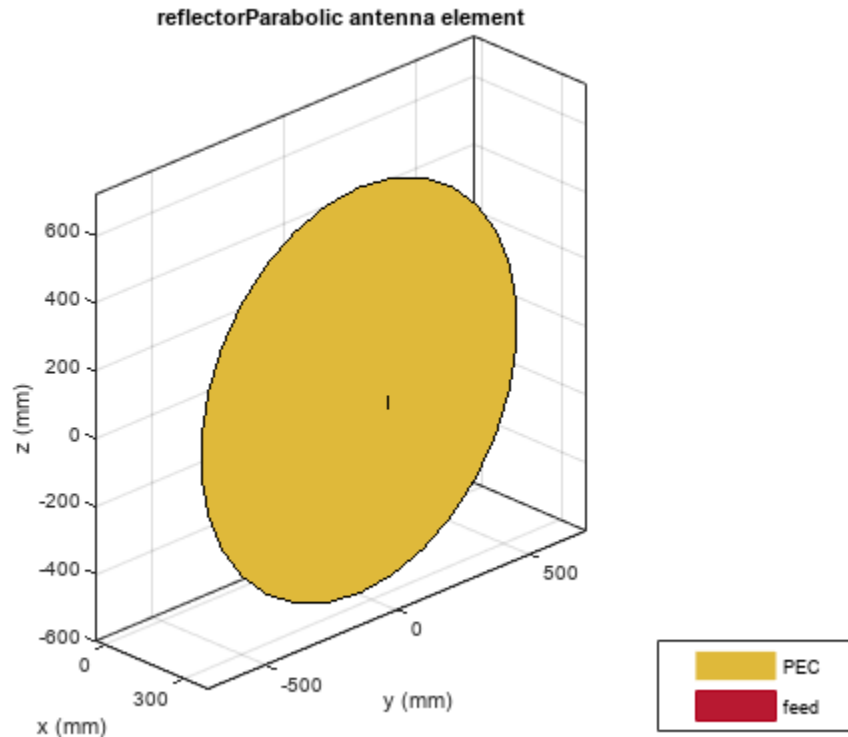
For this example, we will consider a common C-band downlink frequency used by satellites such as the Intelsat-30 serving the Americas region [1]. Also, we will target a Very Small Aperture Terminal (VSAT) application and therefore, limit the diameter of the reflector to be 1.2 m. At the upper end the electrical size of the reflector will be about 15 λ . Finally, the F/D ratio is chosen to be 0.3.

```
C_band = [3.4e9 3.7e9];
vp = physconst('lightspeed');
C_band_lambda = vp./C_band;
D = 1.2;
D_over_lambda_C = D./C_band_lambda;
F_by_D = 0.3;
```

Design Reflector

Design the reflector at the selected frequency of 3.5 GHz and adjust the parameters as needed for the example. Re-orient the parabolic reflector to have the boresight align with x-axis.

```
f = 3.5e9;
lambda = vp/f;
p = design(reflectorParabolic,f);
p.Radius = D/2;
p.FocalLength = F_by_D*D;
p.Tilt = 90;
p.TiltAxis = [0 1 0];
figure
show(p)
view(45,25)
```

Obtain an Estimate of the Memory Requirements

Since the parabolic reflector is an electrically large structure, it is good to estimate the amount of RAM needed to solve a given structure at the frequency of design. Use the `memoryEstimate` function to do this.

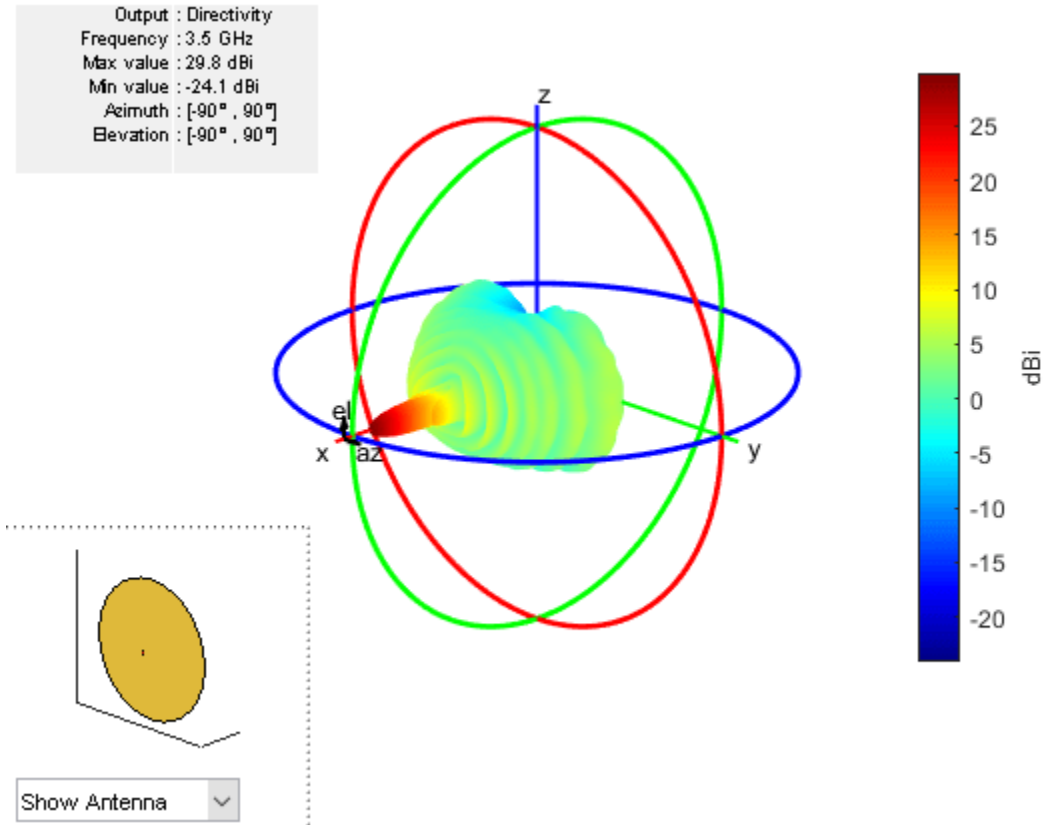
```
m = memoryEstimate(p, f)
```

```
m =  
'880 MB'
```

3D Pattern

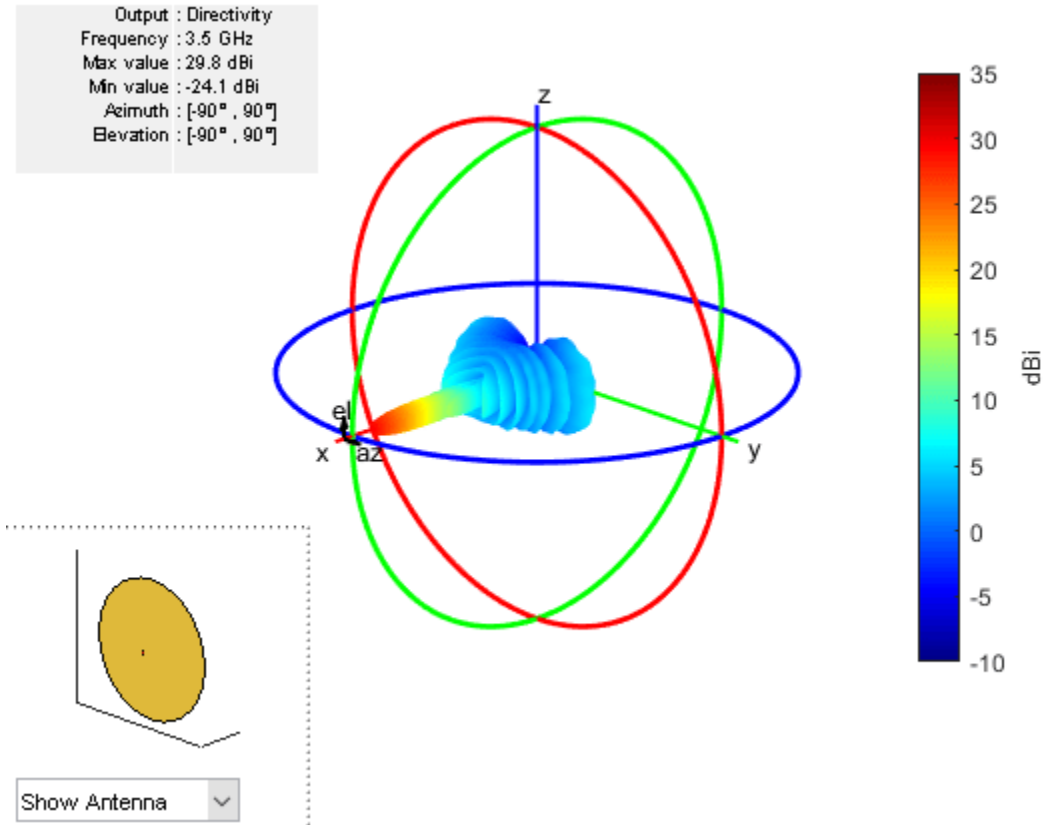
Calculate the 3D far-field directivity pattern for the forward-half plane including the boresight. In addition, we rescale the magnitude to enhance the features in the pattern using the `PatternPlotOptions`.

```
az = -90:1:90;  
el = -90:1:90;  
figure  
pattern(p, f, az, el)
```



Create a `PatternPlotOptions` object and rescale the magnitude for the plot.

```
patOpt = PatternPlotOptions;
patOpt.MagnitudeScale = [-10 35];
figure
pattern(p,f,az,el,'patternOptions',patOpt)
```



Calculating the Aperture Efficiency

The maximum gain from the parabolic reflector is achieved under uniform illumination of the aperture (amplitude, phase). A feed pattern that compensates for the spherical spreading loss with the angle off from the axis and at the same time becoming zero at the rim to avoid spillover related losses would achieve this ideal efficiency of unity [2]. In reality we have different types of antennas that are used as feeds such as dipoles, waveguides, horns etc. Using the pattern analysis, we can numerically estimate the aperture efficiency. This calculation yields an aperture efficiency of approximately 50% for a dipole feed.

```
Dmax = pattern(p, f, 0, 0);
eta_ap = (10^(Dmax/10)/(pi^2))*(lambda/D)^2
```

```
eta_ap = 0.4942
```

Effect of Axial Displacement of the Feed

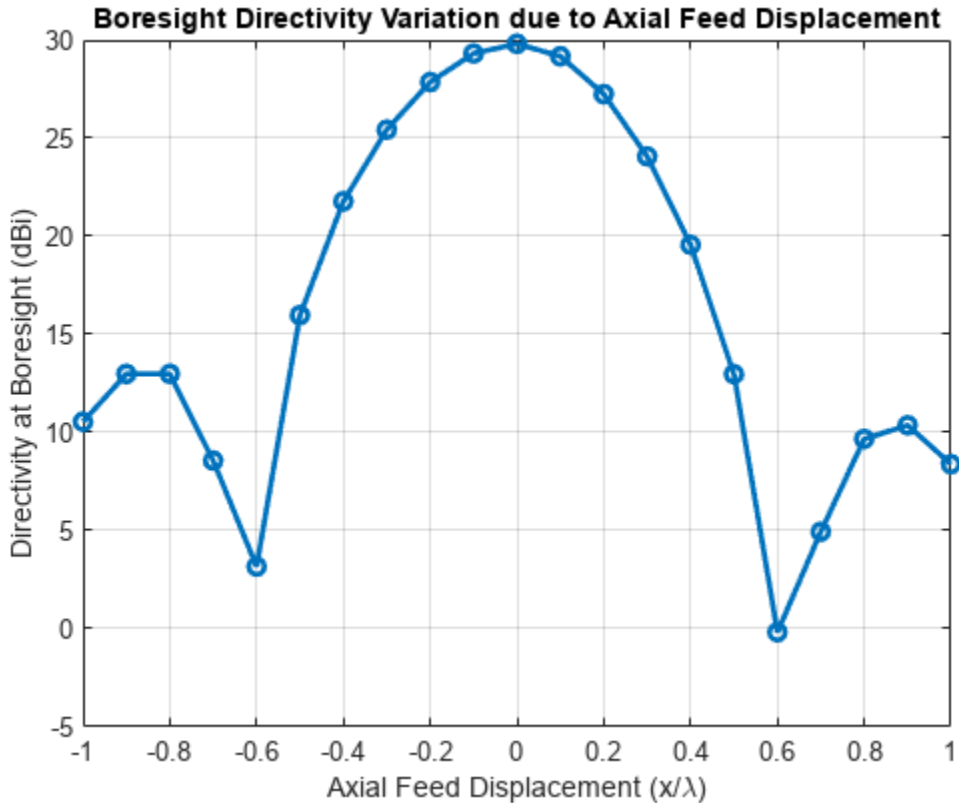
In certain applications it might be necessary to position the feed away from the focal point of the reflector. As expected such a configuration will introduce phase aberrations which will translate to a pattern degradation. Investigate the effect of an axial displacement of the feed, both - towards and away from the focus on the peak gain at boresight, i.e. $(az, el) = (0, 0)$ degrees. To do so, vary the x-coordinate of the FeedOffset property on the parabolic reflector.

```
feed_offset = -lambda:0.1*lambda:lambda;
Dmax_offset = zeros(size(feed_offset));
for i = 1:numel(feed_offset)
    p.FeedOffset = [feed_offset(i), 0, 0];
```

```

Dmax_offset(i) = pattern(p,f,0,0);
end
figure
plot(feed_offset./lambda,Dmax_offset,'o-','LineWidth',2)
xlabel('Axial Feed Displacement (x/\lambda)')
ylabel('Directivity at Boresight (dBi)')
grid on
title('Boresight Directivity Variation due to Axial Feed Displacement')

```



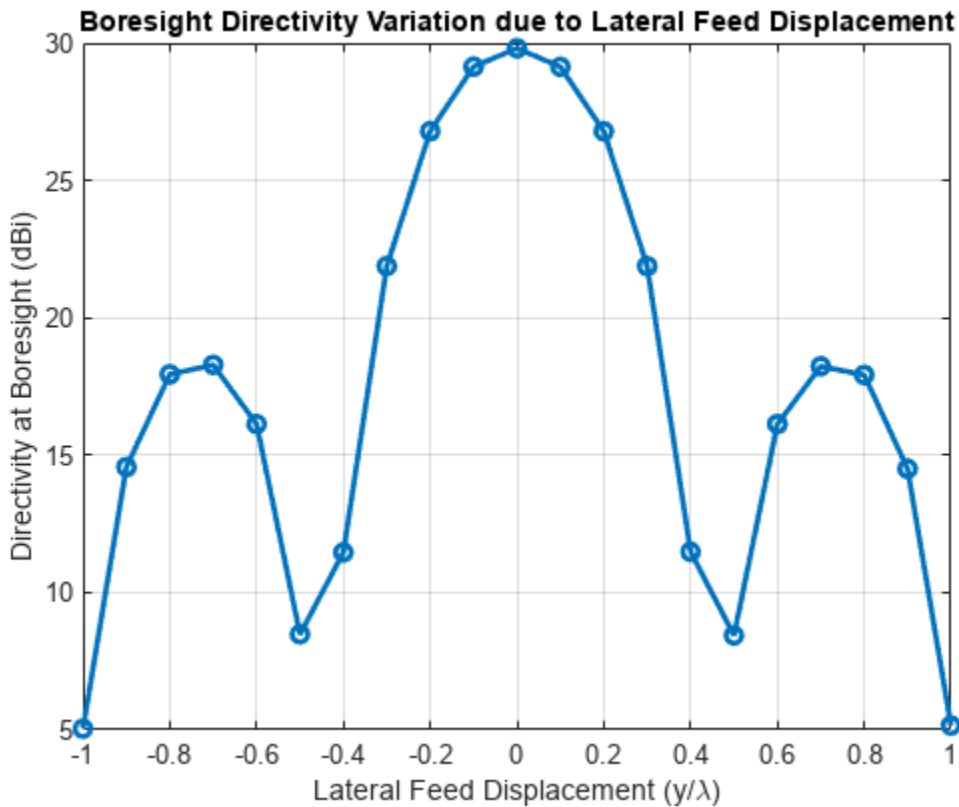
Effect of the Lateral Displacement Feed

Displacement of the feed way from the axis, laterally results in beam scan. For symmetric parabolic reflectors this effect is limited. Similar to the previous section, we continue to look at the bore sight gain variation as a function of the feed being displaced along the y-axis.

```

Dmax_offset = zeros(size(feed_offset));
for i = 1:numel(feed_offset)
    p.FeedOffset = [0,feed_offset(i),0];
    Dmax_offset(i) = pattern(p,f,0,0);
end
figure
plot(feed_offset./lambda,Dmax_offset,'o-','LineWidth',2)
xlabel('Lateral Feed Displacement (y/\lambda)')
ylabel('Directivity at Boresight (dBi)')
grid on
title('Boresight Directivity Variation due to Lateral Feed Displacement')

```



Effect of Random Surface Errors on the Reflector Surface

Ideally the surface of parabolic reflector will be perfectly smooth without any surface imperfections. Manufacturing processes and mechanical stresses result in a surface that deviates from the perfect paraboloid. Use an RMS surface error term for each co-ordinate and analytically estimate the gain degradation due to surface errors [3].

```
epsilon_rms = lambda/25;
chi = (4*F_by_D)*sqrt(log(1 + 1/(4*F_by_D)^2));
Gmax_est = 10*log10(eta_ap*(pi*D/lambda)^2*exp(-1*(4*pi*chi*epsilon_rms/lambda)^2))
```

```
Gmax_est = 28.9771
```

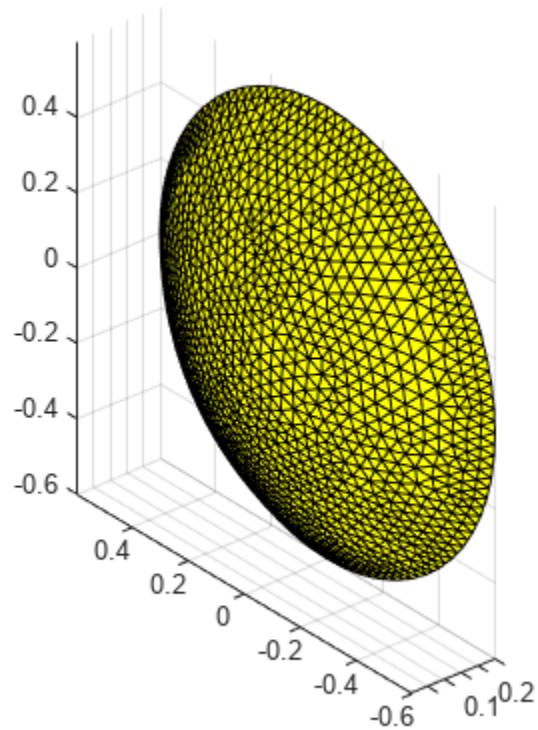
Next, we build a geometric model of the reflector with surface errors. To do so we isolate the mesh for the reflector alone and perturb the points on the surface with a zero-mean Gaussian random process. The standard deviation of this process is assigned to be the RMS surface error. After perturbing the points, we compute the rms surface error to confirm that the process deviation is indeed close to what we set.

```
p.FeedOffset = [0,0,0];
[Pt,t] = exportMesh(p);
idrad = find(Pt(:,1)>=p.FocalLength);
idref = find(Pt(:,1)<p.FocalLength);
removeTri = [];
for i = 1:size(t,1)
    if any(t(i,1)==idrad)||any(t(i,2)==idrad)||any(t(i,3)==idrad)
        removeTri = [removeTri,i];
    end
end
```

```

    end
end
tref = t;
tref(removeTri,:) = [];
figure
patch('Faces',tref(:,1:3),'Vertices',Pt,'FaceColor','yellow');
axis equal;
axis tight;
grid on;
hfig = gcf;
ax = findobj(hfig, 'type','axes');
z = zoom;
z.setAxes3DPanAndZoomStyle(ax,'camera');
view(-40, 30)

```



Create gaussian noise for perturbing surface mesh

```

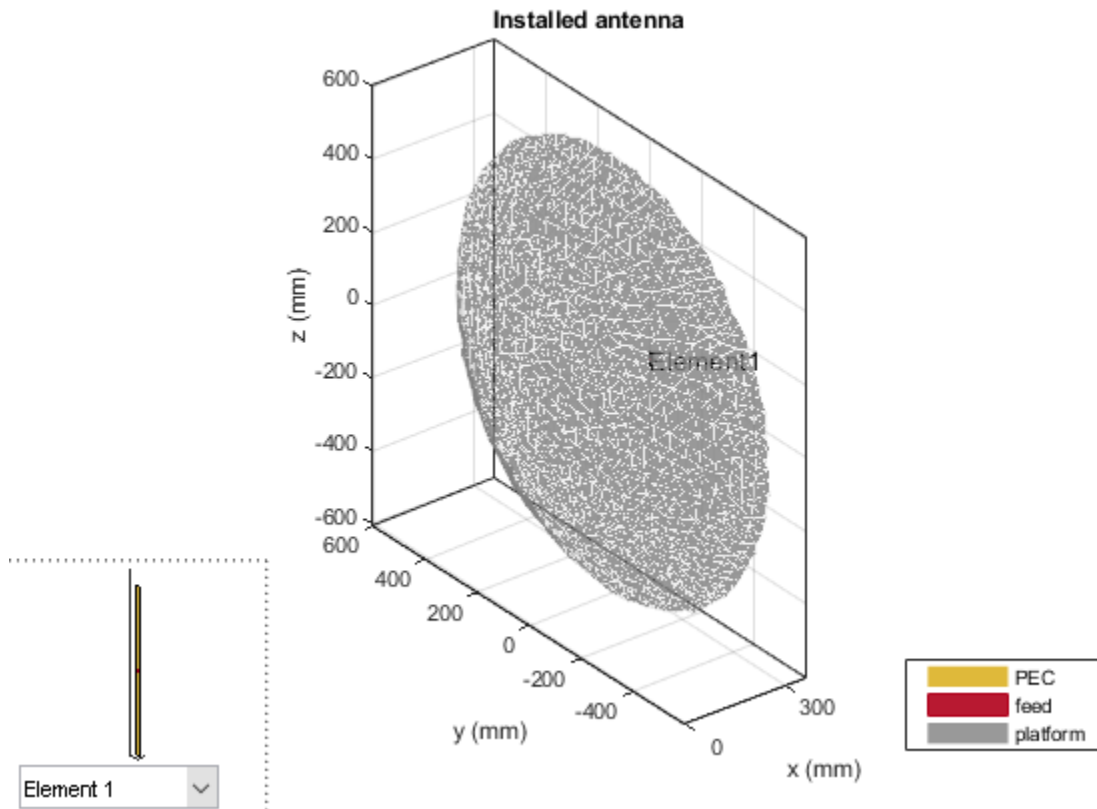
n = epsilon_rms*randn(numel(idref),3);
Ptnoisy = Pt(idref,:) + n;
rms_model_error = sqrt(mean((Pt(idref,:)-Ptnoisy).^2,1))

rms_model_error = 1×3
    0.0034    0.0034    0.0034

```

Create an STL file out of the reflector surface and make it the platform for an installed antenna analysis as shown. The excitation element is the same as before. Assign the position of the element using the feedlocation property on the parabolic reflector.

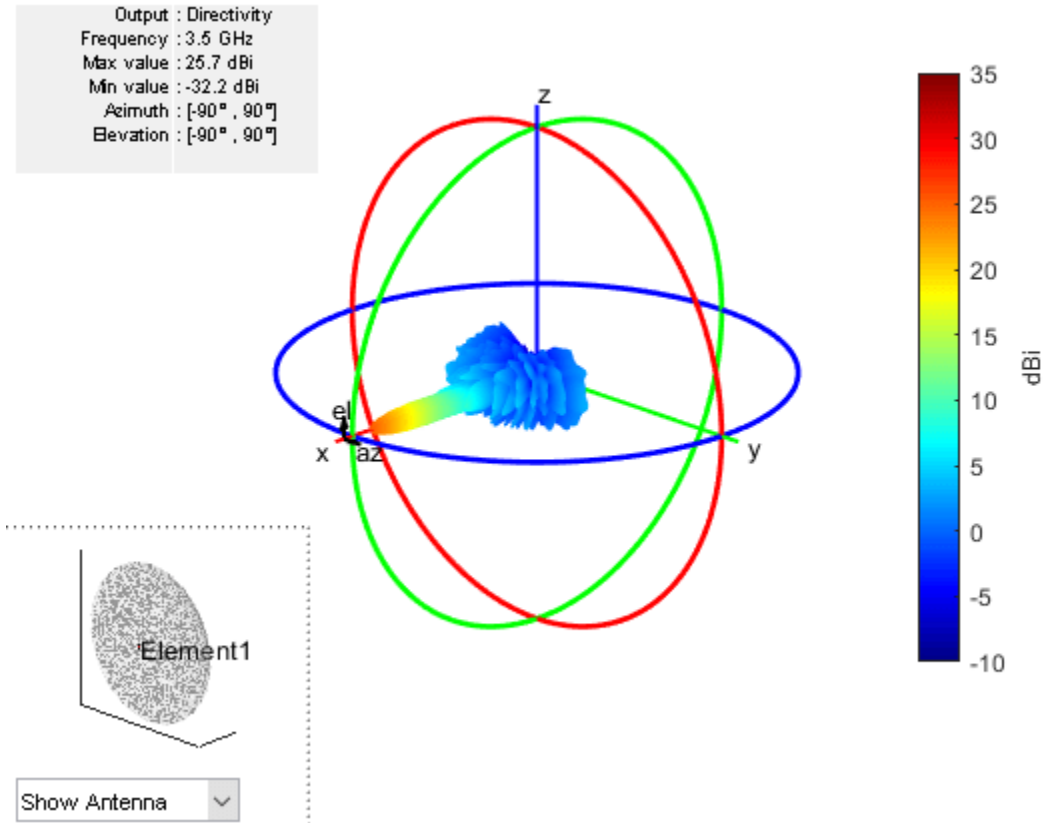
```
TR = triangulation(tref(:,1:3),Ptnoisy);
stlwrite(TR,'noisyref.stl')
pn = installedAntenna;
pl = platform;
exciter = p.Exciter;
exciter.Tilt = 0;
exciter.TiltAxis = [0 1 0];
pl.FileName = 'noisyref.stl';
pl.Units = 'm';
pn.Platform = pl;
pn.Element = exciter;
pn.ElementPosition = [p.FeedLocation(1),0,0];
figure
show(pn)
```



Far-field 3D Pattern for Reflector surface with Errors

The effect of the surface errors on the reflector result in a 3 dB boresight gain reduction. This effect is particularly important to consider at the Ka, Ku and higher bands

```
patnOpt = PatternPlotOptions;
patnOpt.MagnitudeScale = [-10 35];
figure
pattern(pn,f,az,el,'patternOptions',patnOpt)
```



References

- [1] "Satellite Coverage Maps | Intelsat." Accessed May 26, 2022. <https://www.intelsat.com/fleetmaps/?s=G-13>.
- [2] W. L. Stutzman, G. A. Thiele, Antenna Theory and Design, p. 307, Wiley, 3rd Edition, 2013.
- [3] J. Ruze, "Antenna tolerance theory-a review," Proc. of IEEE, vol. 54, no.4. pp.633-640, April, 1966.

See Also

"Sector Antenna for 2.4 GHz Wi-Fi™" on page 5-107

Radar Cross Section Benchmarking

This example performs benchmarking of the Radar Cross Section computation on three structures: a square plate, a circular plate and the NASA almond. The benchmarking for the square and circular plate is done against the analytical physical optics based solution and in the case of the NASA Almond, the comparison is with the Method of Moments (MoM) solution.

Square Plate RCS Parameters Setup

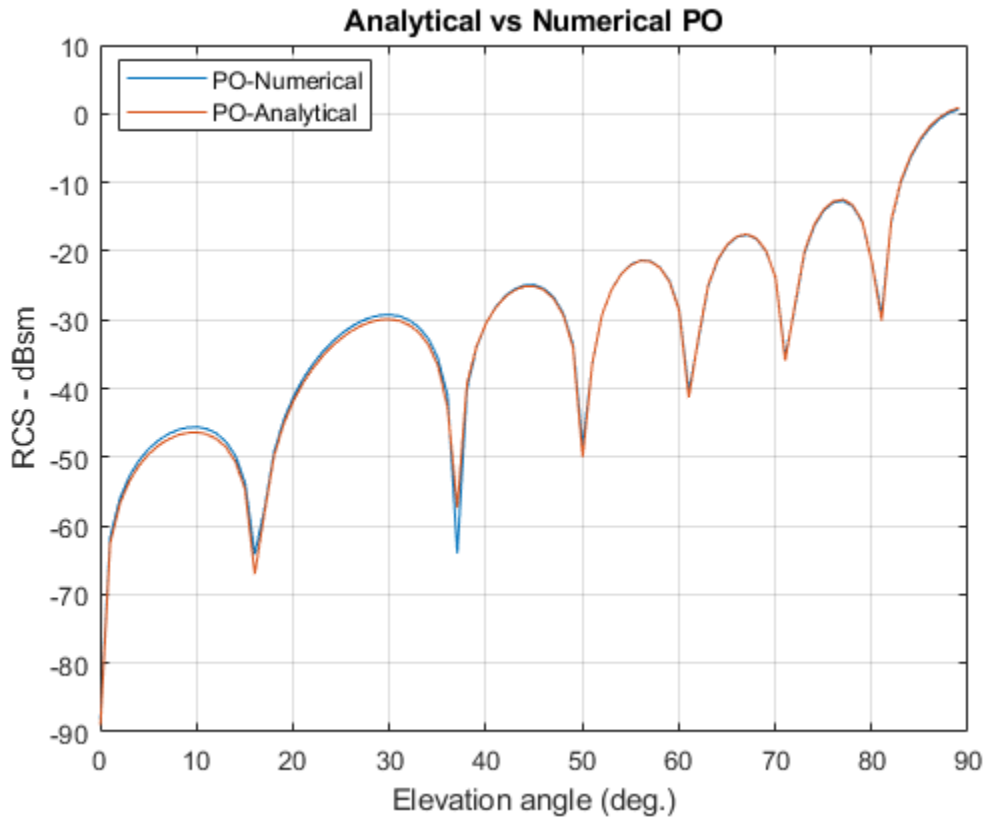
Define the physical dimensions for the square plate, the wavelength and frequency for analysis. The plate is defined using an STL file.

```
lambda = 3.25e-2;  
f1 = physconst('lightspeed')/lambda;  
L = 10.16e-2;  
W = 10.16e-2;  
p = platform;  
p.FileName = 'square_plate.stl';  
p.Units = 'm';
```

Analyze and Compare with Analytical result

The RCS computation is done in the elevation plane at azimuth = 0 deg. The electric-field polarization vector is set to HH. This implies a horizontal component on transmit and horizontal component on receive is used for the RCS calculation. The results for the RCS from the toolbox are compared with the analytical results provided in [1].

```
az = 0;  
el = 0.05:1:90;  
sigma = rcs(p,f1,az,el,'Polarization','HH');  
asigma1 = rectPlateRCS(L,W,f1,az,90-el);  
figure  
plot(el,sigma,el,asigma1)  
grid on  
xlabel('Elevation angle (deg.)')  
ylabel('RCS - dBsm')  
title('Analytical vs Numerical PO')  
legend('PO-Numerical','PO-Analytical','Location','best')
```



Circular Plate RCS Parameters Setup

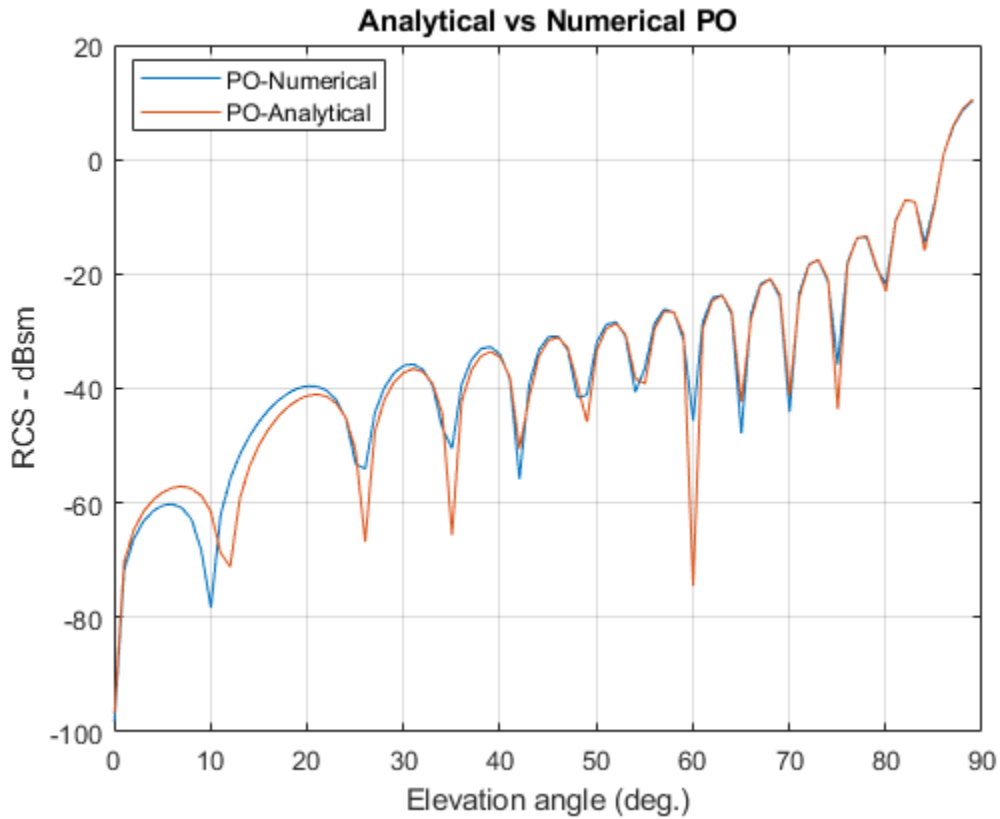
Define the physical dimensions of the circular plate. The circular plate is described using an STL file. All dimensions are in meters.

```
R = 10.16e-2;
pc = platform;
pc.FileName = 'circular_plate.stl';
pc.Units = 'm';
```

Analyze and Compare with analytical result

As before compare the results from the RCS function in the toolbox with the analytical expression provided in [1]. The RCS calculation is done in the elevation plane between 0 and 90 deg.

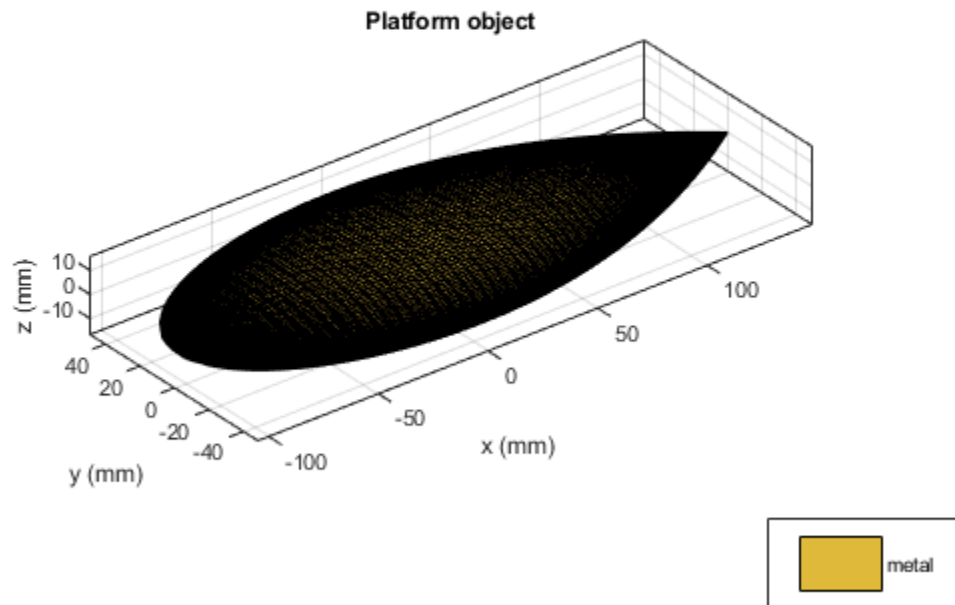
```
az = 0;
el = 0.05:1:90;
sigmaV = rcs(pc,f1,az,el,'Polarization','HH');
asignal = circPlateRCS(R,f1,90-el);
figure
plot(el,sigmaV,el,asignal)
grid on
xlabel('Elevation angle (deg.)')
ylabel('RCS - dBsm')
title('Analytical vs Numerical PO')
legend('PO-Numerical','PO-Analytical','Location','best')
```



NASA Almond Setup

The third structure is the NASA almond shape described in [2]. This is a classic shape for benchmarking the performance of high-frequency electromagnetic solvers. The mathematical expressions in [2] have been used to create the STL file that describes the almond shape.

```
p = platform;  
p.FileName = 'NASA-Almond.stl';  
p.Units = 'm';  
figure  
show(p)
```



Analysis parameters

Since the physical optics solver is only applicable at large ka values, we compare the results produced by the Antenna Toolbox with those published in [2]. The results produced by the toolbox will be from both solvers, the physical optics (PO) and the Method-of-Moments (MoM). The wavelength at 7 GHz is approximately 4.3 cm. We refine the mesh to be slightly finer than this using the $\lambda/10$ criterion.

```
f2 = 7e9;
m = mesh(p, 'MaxEdgeLength', .0035)
az = 0:1:180;
el = 0;
```

```
m =
```

```
struct with fields:
    NumTriangles: 7878
    NumTetrahedra: 0
    NumBasis: []
    MaxEdgeLength: 0.0035
    MeshMode: 'manual'
```

RCS Calculation with HH-Polarization

Calculate the RCS for the HH-polarization condition in which the transmitted and received field is horizontally polarized.

```
sigmahh_po = rcs(p,f2,az,el,'Solver','PO',...
                'EnableGPU', false,...
                'Polarization','HH');

sigmahh_mom = rcs(p,f2,az,el,'Solver','MoM',      ...
                'Polarization','HH');
```

RCS Calculation with VV-Polarization

Calculate the RCS for the VV-polarization condition in which the transmitted and received field is vertically polarized.

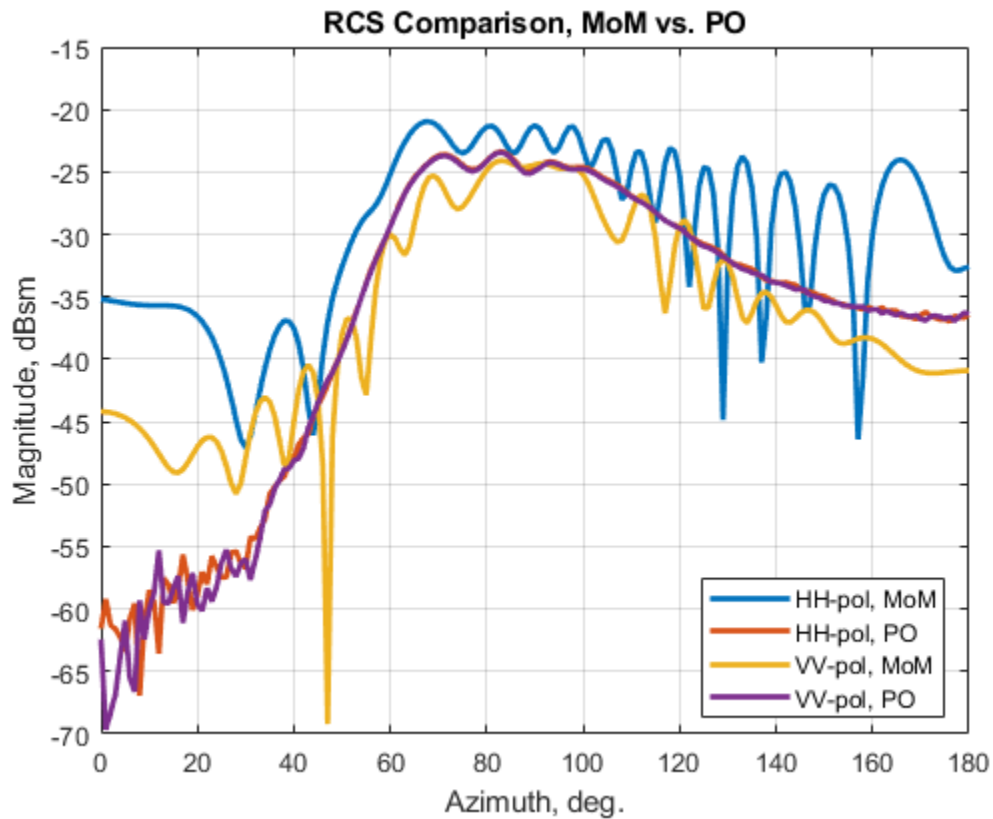
```
sigmavv_po = rcs(p,f2,az,el,'Solver','PO',...
                'EnableGPU', false,...
                'Polarization','VV');

sigmavv_mom = rcs(p,f2,az,el,'Solver','MoM',      ...
                'Polarization','VV');
```

Plot the results

Overlay the plots from both solvers for both polarizations to compare. Notice that the full-wave MoM solver extracts all the phenomenon that contribute to the scattered field. In contrast the PO solver being a first-order approximation is able to predict the level of RCS but averages out the variations at the different angles. This is expected since the PO solver assumes that the surface current density outside the illuminated region, i.e. the shadow region is zero, thus not contributing to the scattered field.

```
figure
plot(az,sigmahh_mom,az,sigmahh_po,az,sigmavv_mom,az,sigmavv_po,'LineWidth',2)
ax = gca;
ax.YLim = [-70,-15];
title('RCS Comparison, MoM vs. PO')
xlabel('Azimuth, deg.')
ylabel('Magnitude, dBsm')
grid on
legend('HH-pol, MoM','HH-pol, PO', 'VV-pol, MoM','VV-pol, PO','Location','best')
```



Summary

The RCS benchmarking results compare favorably with published results using analytical techniques as well as other numerical solvers.

References

[1] Radar System Analysis and Design Using MATLAB, Bassem R. Mahafza, Chapman&Hall/CRC,2000.

[2]A. C. Woo, H. T. G. Wang, M. J. Schuh and M. L. Sanders, "EM programmer's notebook-Benchmark radar targets for the validation of computational electromagnetics programs," in IEEE Antennas and Propagation Magazine, vol. 35, no. 1, pp. 84-89, Feb. 1993.

See Also

"Planning Radar Network Coverage over Terrain" on page 5-521

Design and Analyze Cassegrain Antenna

This example shows how to create a cassegrain antenna. A typical parabolic antenna consists of a parabolic reflector with a small feed antenna at its focus. Parabolic reflectors used in dish antennas have a large curvature and short focal length and the focal point is located near the mouth of the dish, to reduce the length of the supports required to hold the feed structure. In more complex designs, such as the cassegrain antenna, a sub reflector is used to direct the energy into the parabolic reflector from a feed antenna located away from the primary focal point. Cassegrain provides an option to increase focal length, reducing side lobes. Such type of antennas can be used in satellite communications as well as Astronomy and other emerging modes of communications.

Define Parameters

Parameters given below helps designing of the cassegrain antenna

R_p = Radius of main reflector

f_p = Focal Length of main reflector

R_{sub} = sub reflector radius

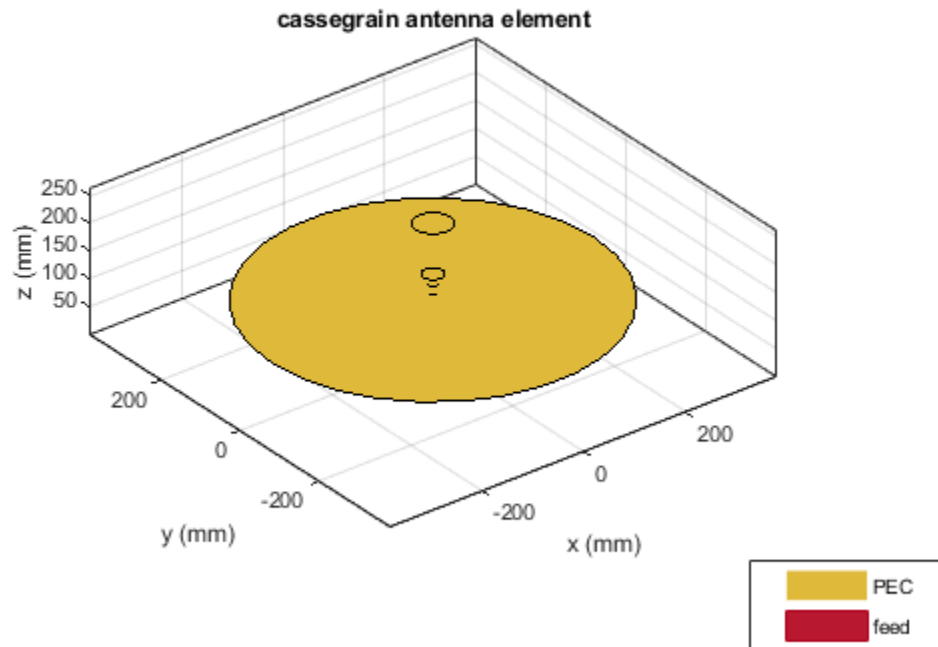
f_{hyp} = focal length of hyperbola

```
Rp=0.3175;
fp=0.2536;
Rsub=0.033;
fhyp=0.1416;
```

Model Cassegrain Antenna

Cassegrain antenna typically has three structures. First is the main reflector which is parabolic, second is the subreflector which is hyperbolic, third is the exciter element. Focus of the main reflector and the near focus of the hyperbolic subreflector coincides. The energy is directed from the subreflector towards main reflector. The parabolic reflector converts a spherical wavefront into a plane wavefront as the energy directed towards it appears to be coming from focus.

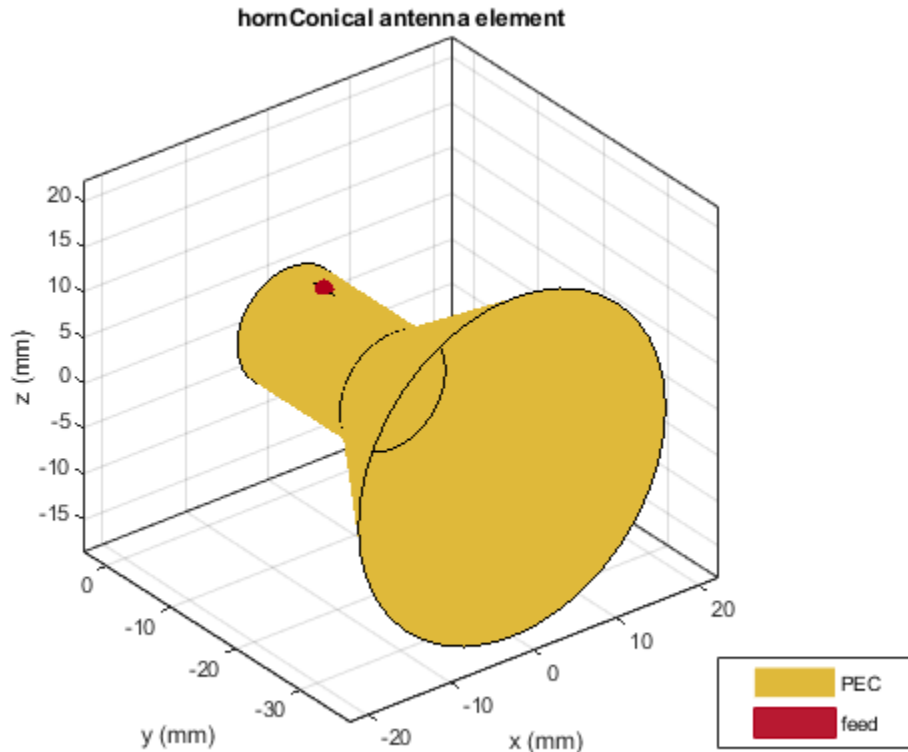
```
ant=cassegrain('Radius',[Rp Rsub],'FocalLength',[fp fhyp]);
show(ant);
```



Define Exciter Element

Conical horn is used as the default exciter for cassegrain. It is oriented towards the sub reflector. It is designed at a frequency which gives the desired performance. The aperture diameter is chosen analytically to give the desired co-planar pattern beam width.

```
Exciter=design(hornConical,17.7e9);
Exciter.FeedWidth=3.4e-3;
Exciter.Tilt=270;
Exciter.TiltAxis=[0 1 0];
show(Exciter);
```

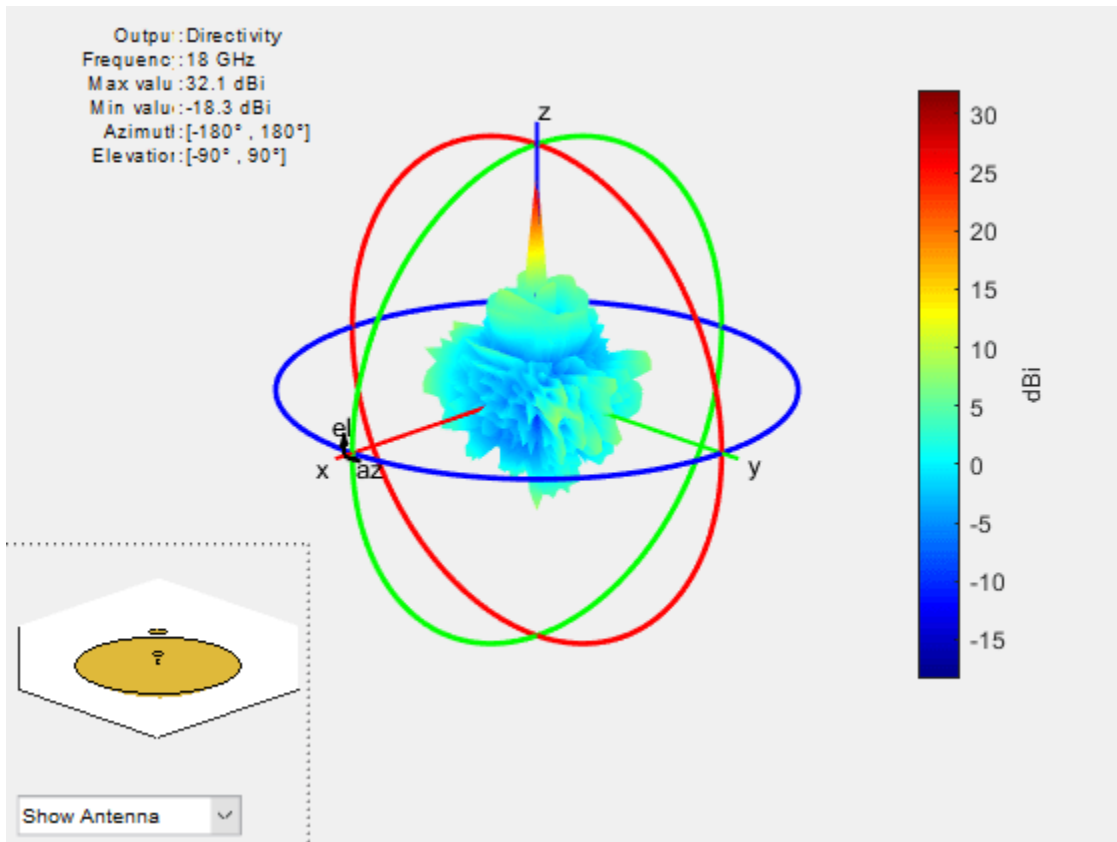
Solve Structure

Default cassegrain antenna uses PO solver for main reflector and MoM solver for sub reflector and exciter. Diameter of main reflector is 39λ and that of sub reflector is 4λ . Solvers are used according to the size of main reflector and subreflector. Typically, if the diameter sub reflector is greater than 5λ , then it is recommended to use PO solver for it. Normally the expected diameter of subreflector is to be less than 20% of main reflector to minimize the blockage by the subreflector.

Plot Radiation Pattern and Mesh Antenna

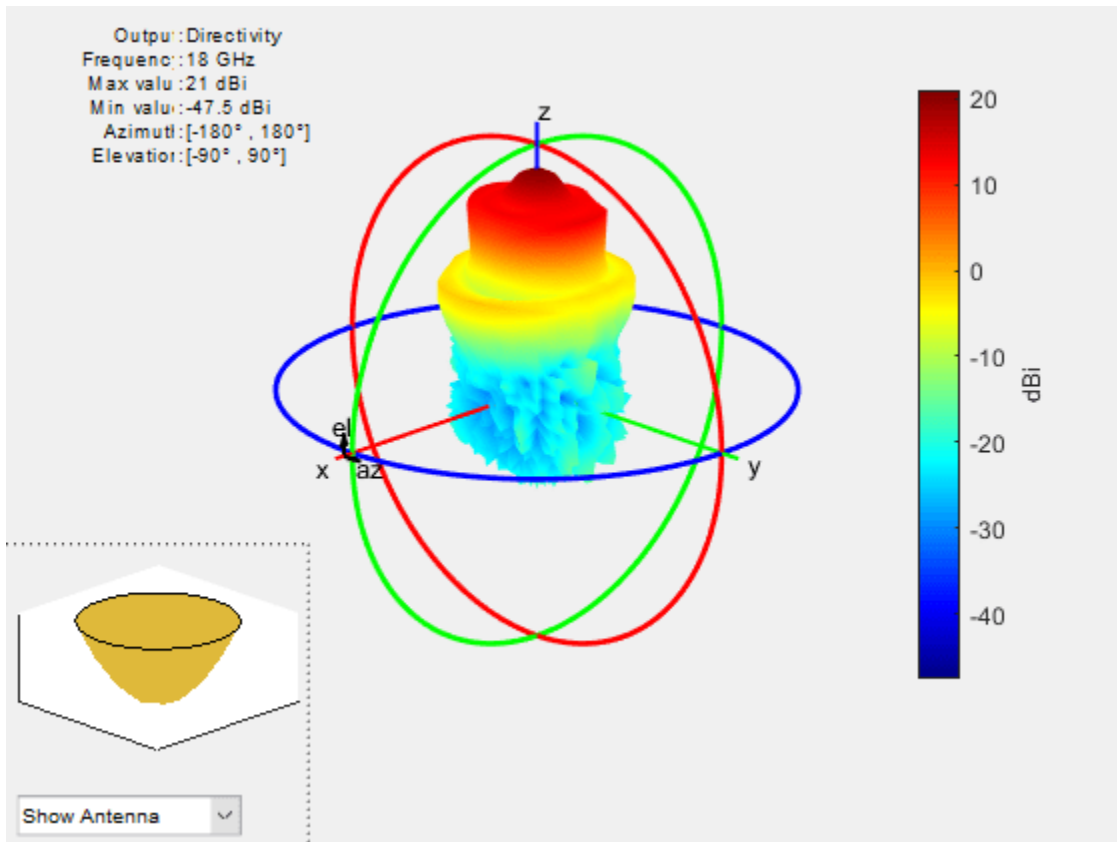
Plot the radiation pattern of cassegrain antenna at 18GHz. The expected pattern need to generate a pencil like beam with less spill over and minor lobe radiation. Default Cassegrain antenna generates more number of triangles, so it can be meshed manually for different mesh edge lengths for the required numbers of triangular generation. This solves the structure quickly, but the gain decreases compared to default in such cases. Mesh can be controlled for exciters and reflectors separately with different mesh edge lengths.

```
ant=cassegrain;
ant.Exciter=design(hornConical,17.7e9);
ant.Exciter.Tilt=270;
[~]=mesh(ant,'MaxEdgeLength',15e-3);
figure;
pattern(ant,18e9);
```



Comparing with parabolic reflector which does not have subreflector shows that cassegrain configuration increases directivity.

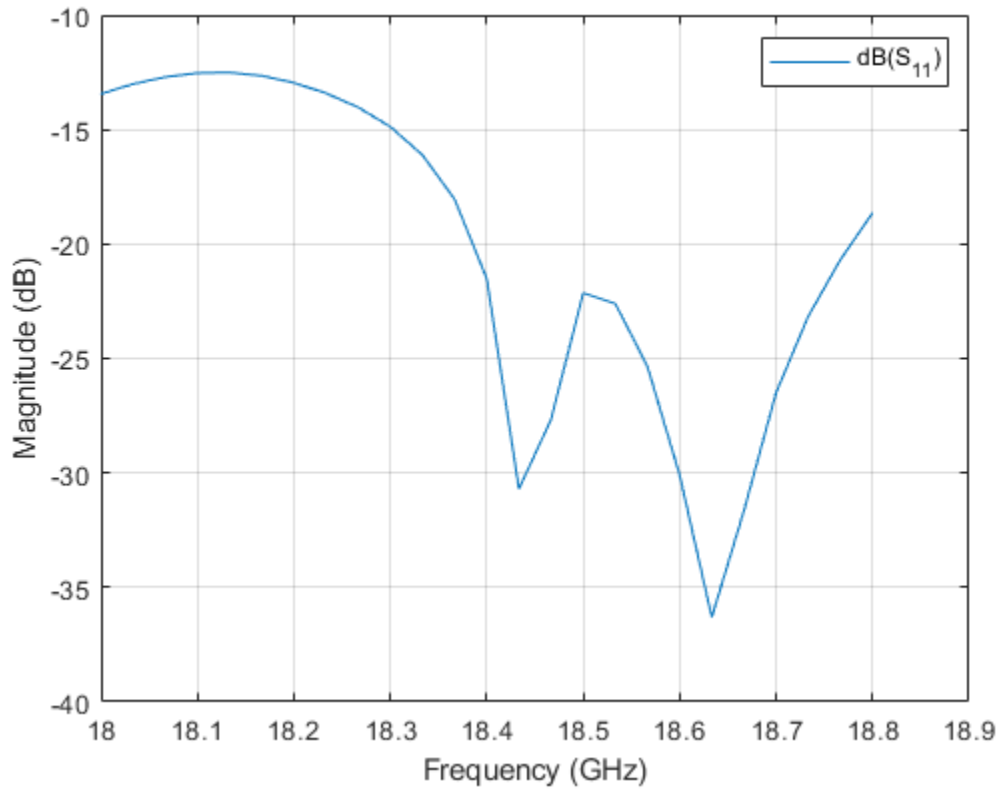
```
ant=reflectorParabolic('Radius',0.3175);
ant.Exciter=design(hornConical,17.7e9);
ant.Exciter.Tilt=90;
figure;
pattern(ant,18e9);
```



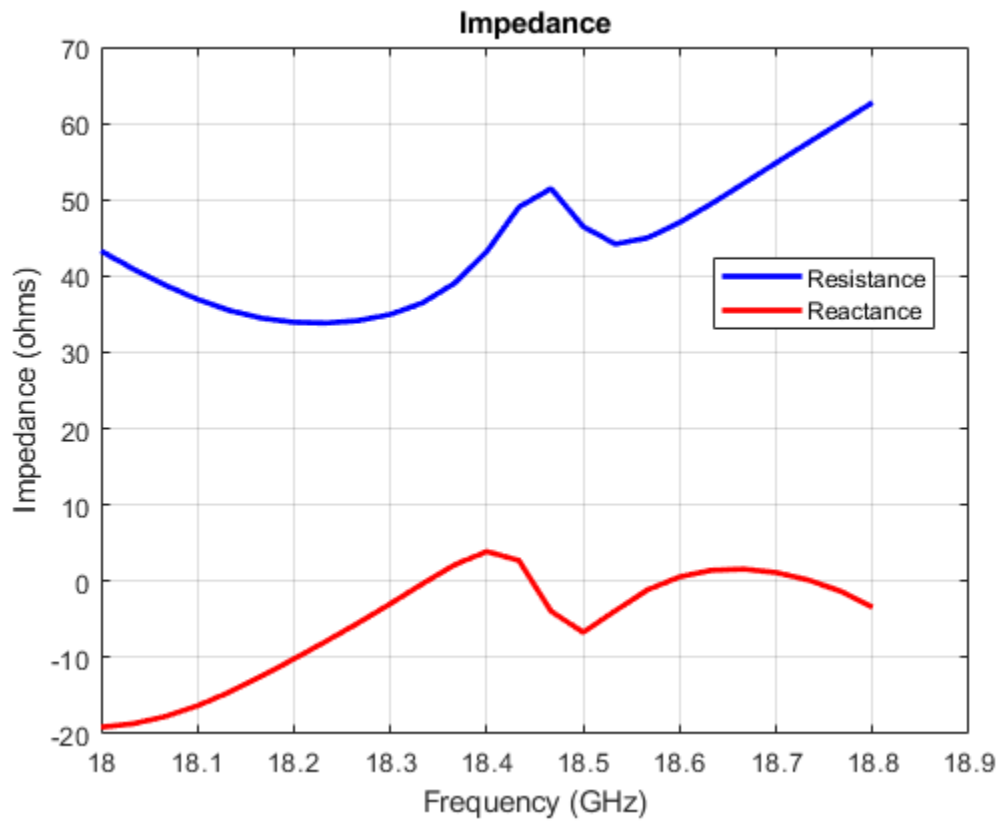
S-parameters and Impedance Plot

Plot the s-parameters and impedance of cassegrain antenna over a frequency range of 18 GHz to 18.8 GHz. It provides a bandwidth of about 100MHz and the structure resonates at 18.51 GHz.

```
ant=cassegrain;
s=sparameters(ant,linspace(18e9,18.8e9,25));
figure;rfplot(s);
```



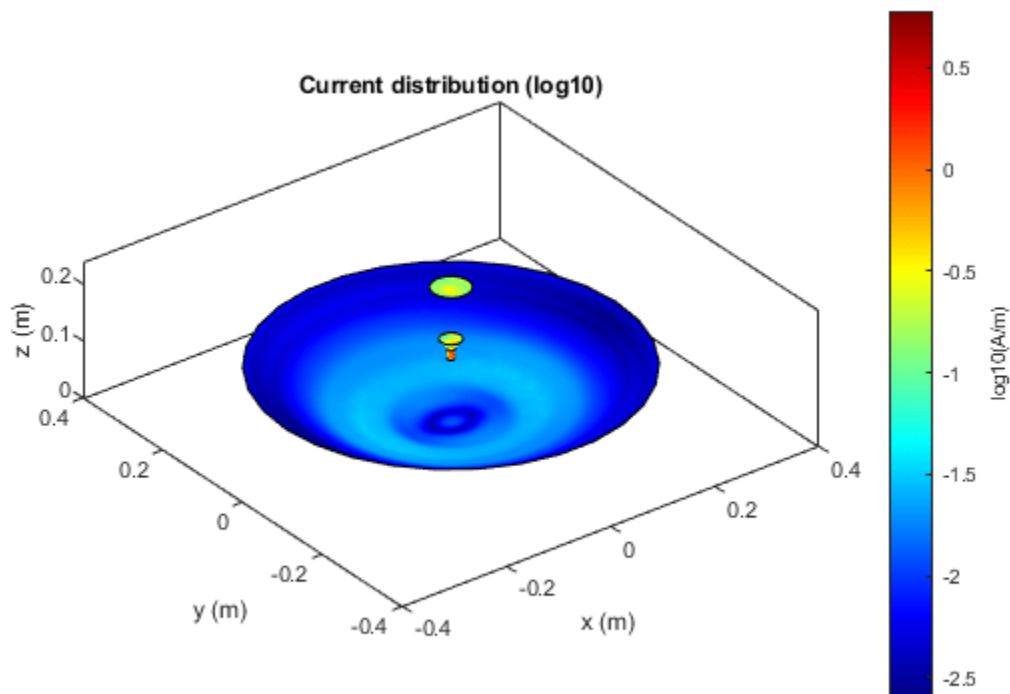
```
figure;  
impedance(ant,linspace(18e9,18.8e9,25));
```



Current Distribution

Analyze the current distribution of cassegrain at a specified frequency.

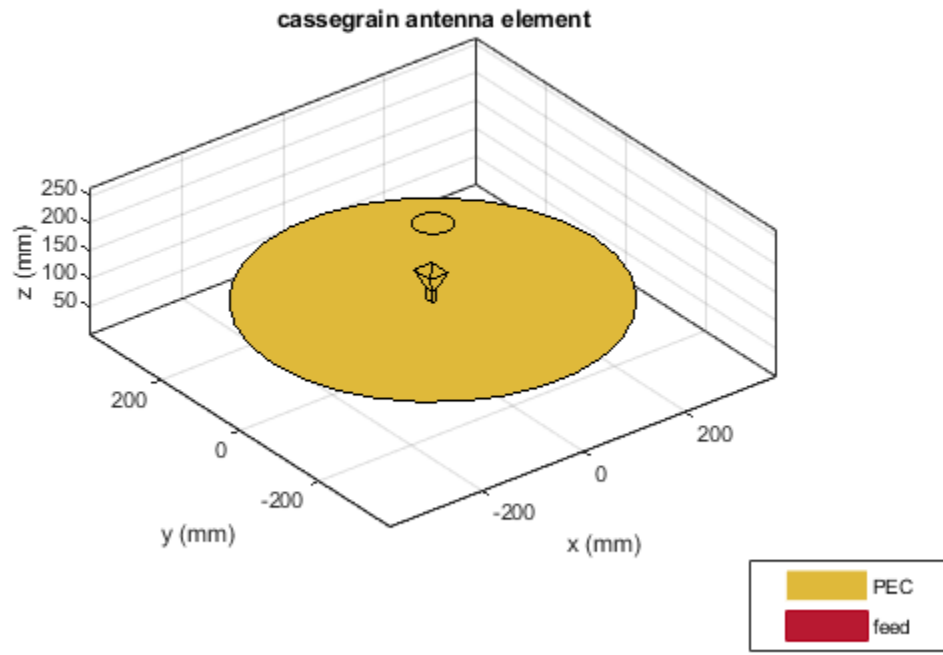
```
current(ant,18e9,'scale','log10');
```

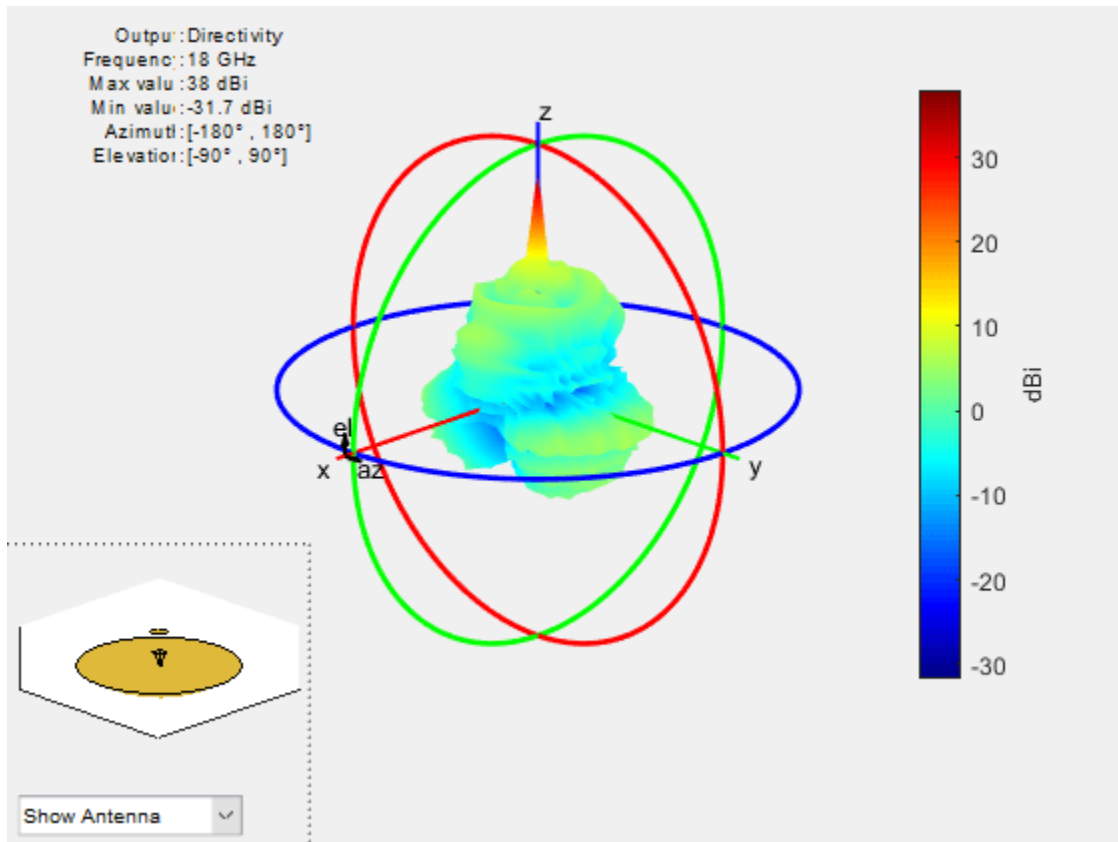


Use Different Exciters with Tilts

Several other elements are supported as exciters like horn, waveguides and so on. Exciter can be tilted in the required direction based on the need.

```
Exciter=design(horn,16.2e9);  
Exciter.Tilt=270;  
Exciter.TiltAxis=[0 1 0];  
ant.Exciter=Exciter;  
show(ant);  
figure;  
pattern(ant,18e9);
```





Conclusion

Using the Cassegrain design increases antenna performance as the feed antenna is directed forward rather than towards a dish in front fed antenna. This orientation reduces side lobes and the use of dual reflectors helps in tailoring radiation pattern for maximum antenna performance.

References

Dandu, Obulesu. "Optimized Design of Axially Symmetric Cassegrain Reflector antenna using Iterative Local Search Algorithm." (2013).

See Also

"Design and Analyze Curved Reflectors" on page 5-705

Discone Antenna for TV Broadcasting System

This example shows how to design and implement a discone antenna for indoor use in digital TV receiving and transmitting systems. Discone antennas are wide bandwidth and omnidirectional radiation antennas that are widely used in VHF and UHF broadcasting systems. The antenna consists of a circular disc and a cone whose apex approaches the centre of the disc.

Define Parameters

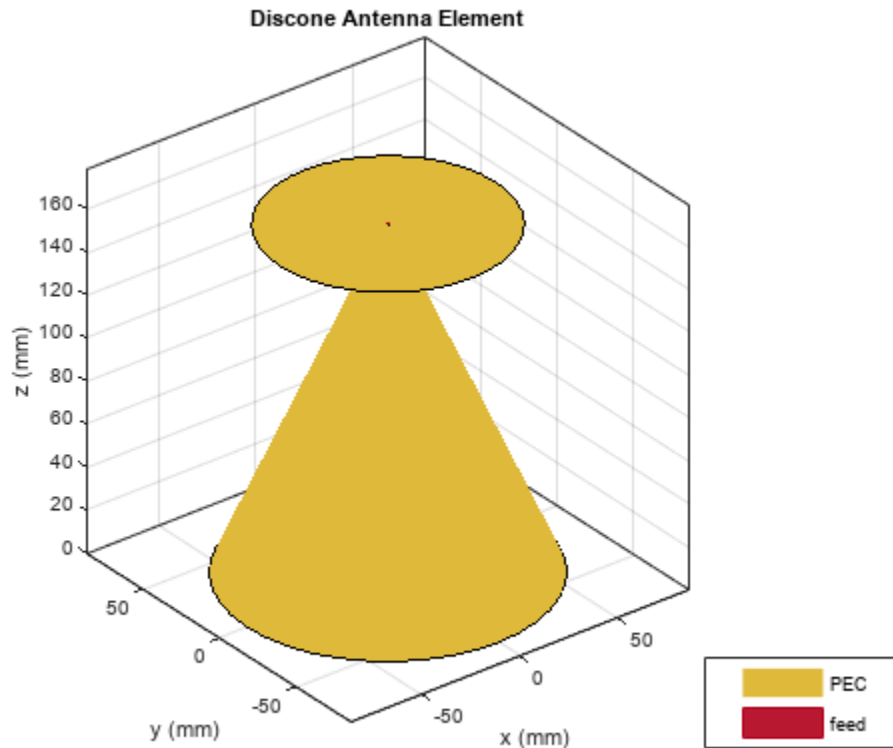
```
Rd = 55e-3;           % Radius of disc
Rc1 = 72.1e-3;       % Broad Radius of cone
Rc2 = 1.875e-3;     % Narrow Radius of cone
Hc = 160e-3;        % Vertical height of cone
Fw = 1e-3;          % Feed Width
S = 1.75e-3;        % Spacing between cone and disc
```

The above dimensions provided in [1] helps to design discone antenna to cover frequencies between 470 and 862 MHz, which is the Ultra High Frequency band of TV broadcasting.

Create Discone Antenna

Create a discone antenna using the defined parameters.

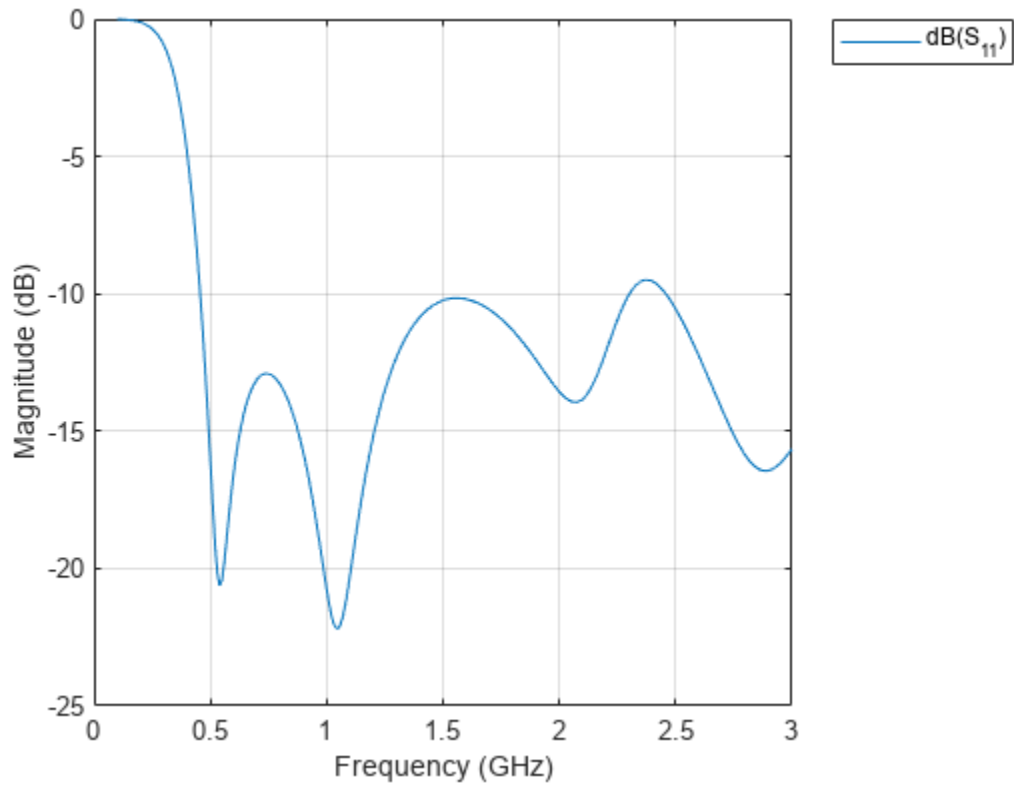
```
ant = discone;
ant.Height = Hc;
ant.ConeRadii = [Rc2 Rc1];
ant.DiscRadius = Rd;
ant.FeedHeight = S;
ant.FeedWidth = Fw;
figure;
show(ant);
title('Discone Antenna Element');
```



S-Parameters

The operational characteristics of the discone antenna are mainly dependent on the broad and narrow radius and the height of the cone. The disc radius and the height of the cone are related to the minimum frequency of operation of the antenna, where the disc should have an overall diameter of 0.7 times a quarter wavelength of the antenna's minimum working frequency. In this example, the minimum operating frequency is 470 MHz. The higher frequency of the discone antenna is mainly determined by the size of the narrow radius of the cone and the gap between the cone and the disc. These dimensions will control the input impedance and is used to guarantee impedance match. Small variations on these dimensions can impose a significant change in the antenna matching.

```
freq = (0.1:0.01:3)*1e9;
[~] = mesh(ant, 'MaxEdgeLength', 10e-3);
s1 = sparameters(ant, freq);
rfplot(s1);
```

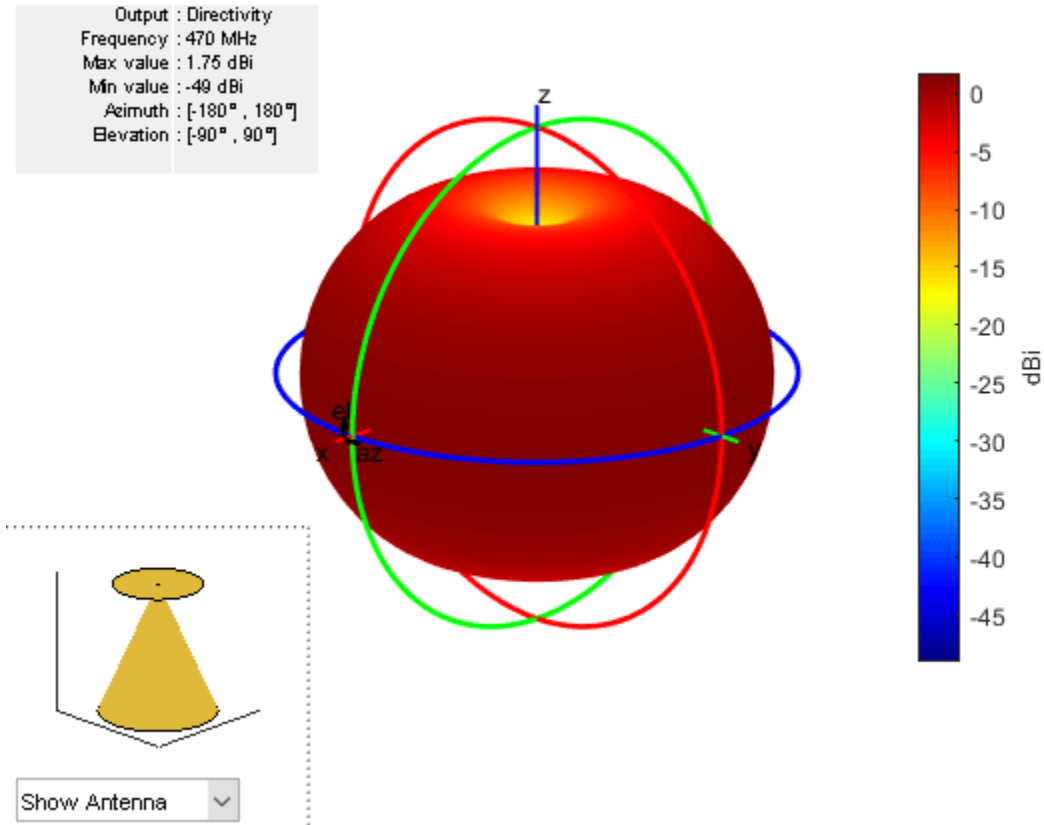


Antenna is said to be in good performance when the reflection coefficient is below -10 dB. There is a good impedance match from 460 MHz to 2.3 GHz. When you plot the S-parameters over a wide range of frequencies, you can observe wideband characteristics of disccone antenna.

Radiation Pattern

Plot radiation pattern at 470MHz

```
f = 470e6;  
figure;  
pattern(ant, f);
```

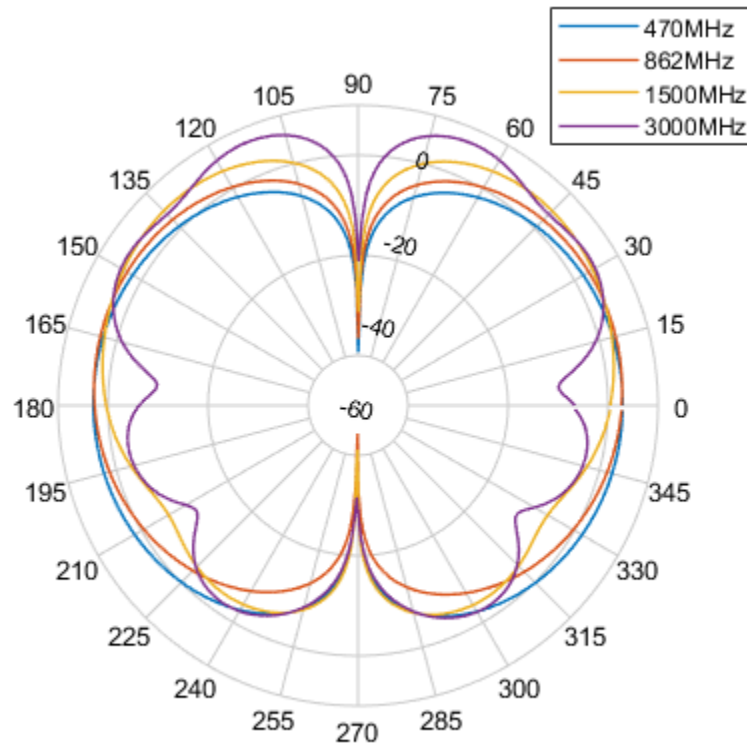


The pattern is omnidirectional from 470 MHz to 862 MHz, with the gain ranging from 1.75 to 2.3 dBi. Due to omnidirectional characteristics, discone antenna is used in TV broadcasting systems.

Elevation Pattern

Elevation pattern is the 2D radiation pattern of the antenna.

```
p1 = patternElevation(ant,470e6);
p2 = patternElevation(ant,862e6);
p3 = patternElevation(ant,1.5e9);
p4 = patternElevation(ant,3e9);
figure;
polarpattern(p1);
hold on;
polarpattern(p2);
polarpattern(p3);
polarpattern(p4);
legend({'470MHz' '862MHz' '1500MHz' '3000MHz'});
```



The radiation pattern is omnidirectional in azimuth plane and bidirectional in elevation plane.

Conclusion

The antenna provides matched bandwidth, below -10 dB of return loss, between 460 MHz and 2.3 GHz, and provides omnidirectional radiation pattern within the considered TV band, from 470 MHz to 862 MHz. There is good agreement between the simulation results and the results from [1] in terms of directivity of radiation pattern.

References

- [1] R.Goncalves, P.Pinho and N.B.Carvalho, "Design and implementation of a 3D printed discone antenna for TV broadcasting system," 2015 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting, Vancouver, BC, 2015, pp. 314-315.
- [2] Sarita Verma, Abhilasha Mishra, Rukhsana khan, "Analysis of Variation of Various Parameters on Design of Discone Antenna", Advanced Computational Techniques in Electromagnetics, Volume 2012(2012),1-5.

See Also

"VHF/UHF Biconical Antenna for Testing Applications" on page 5-714

Analysis of Biquad Yagi for Wi-Fi Applications

This example shows how to analyze the performance of a customized Yagi-Uda antenna. Biquad Yagi antenna is popularly used in Wi-Fi applications.

Define Parameters

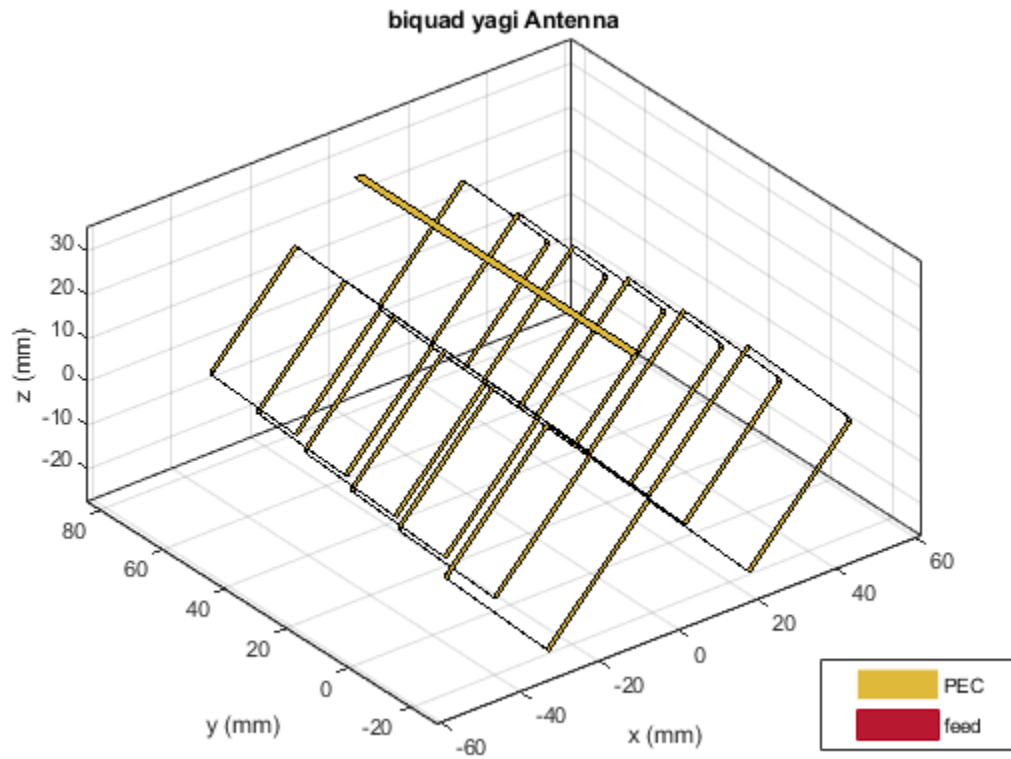
Design the biquad yagi antenna to operate at 2.4 GHz. Use dimensions of 30 mm element for the first parasitic element followed by 31 mm, 32 mm, 33 mm, then 34 mm driven element and a 36 mm reflector at the rear. Define the design parameters of the antenna as provided.

```
ref = biquad('Tilt',90,'ArmLength',36e-3); % Reflector
exct = biquad('Tilt',90,'ArmLength',34e-3); % Driven element
direct1 = biquad('Tilt',90,'ArmLength',33e-3); % Director1
direct2 = biquad('Tilt',90,'ArmLength',32e-3); % Director2
direct3 = biquad('Tilt',90,'ArmLength',31e-3); % Director3
direct4 = biquad('Tilt',90,'ArmLength',30e-3); % Director4
```

Create Biquad Yagi Antenna

Space the parasitic elements 17 mm from driven element and the reflector 19 mm from the driven element. You can increase and decrease the length of the Boom and, you can move the Boom by changing the BoomOffset property. Create a quadCustom antenna using the parameters defined.

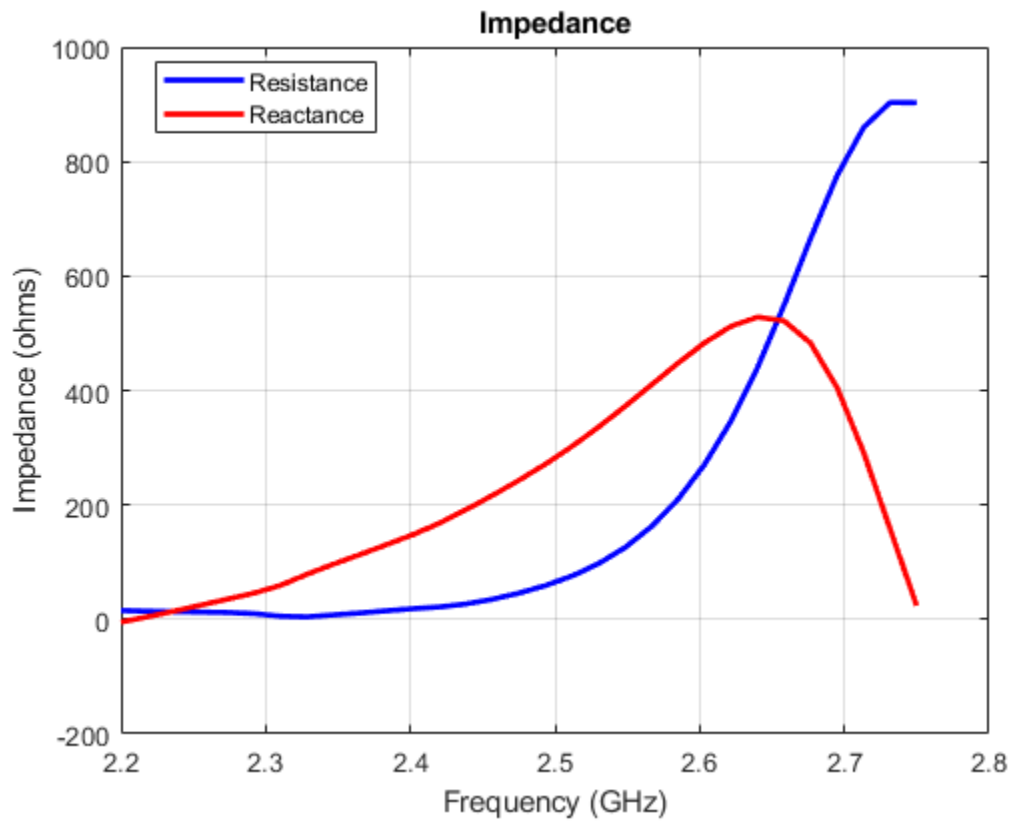
```
ant = quadCustom('Exciter',exct,'Director',{direct1,direct2,direct3,direct4},...
    'DirectorSpacing',17e-3,'Reflector',{ref},'ReflectorSpacing',19e-3,...
    'BoomOffset',[0 0.03 0.030],'BoomLength',0.09);
figure;
ant.Tilt = 180;
ant.TiltAxis = [0 1 1];
show(ant);
% view(-13,17);
title('biquad yagi Antenna');
```



Calculate Antenna Impedance

Calculate the antenna impedance over the frequency range of 2.3 GHz to 2.6 GHz. From the figure, observe the antenna resonates around 2.4 GHz.

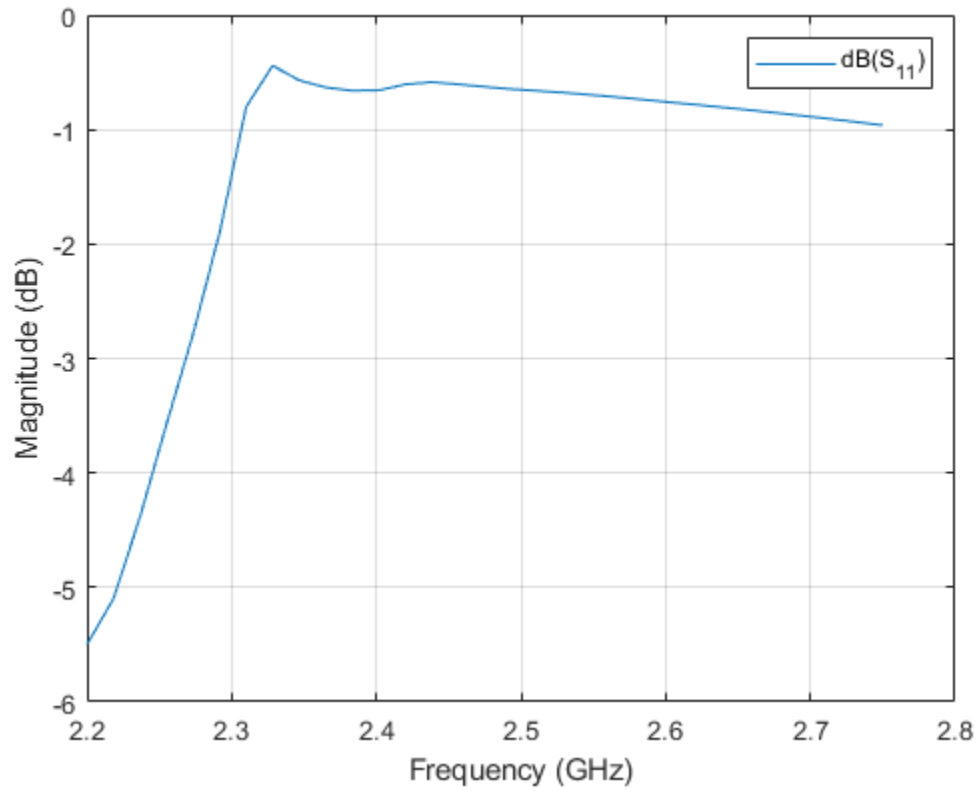
```
figure;  
impedance(ant, linspace(2.2e9, 2.75e9, 31));
```



Plot Reflection Coefficient

Plot the reflection coefficient for this antenna over the band and a reference impedance of 50 ohms.

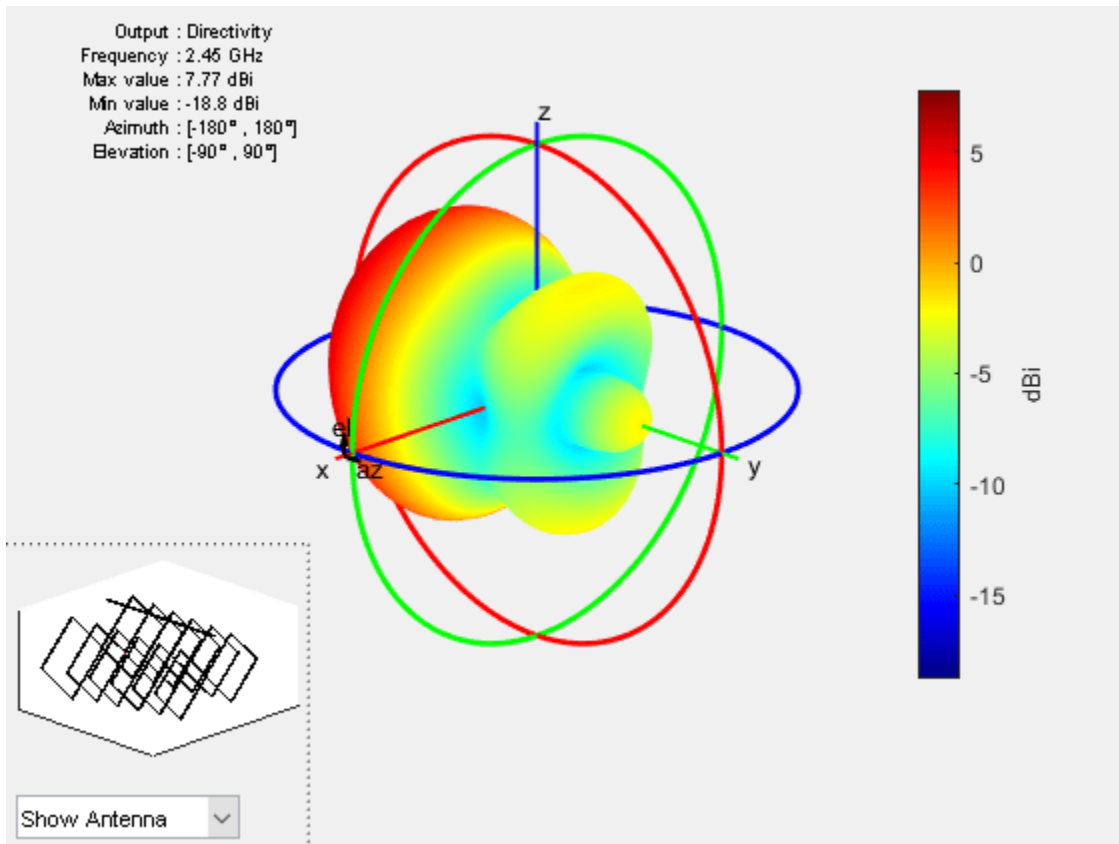
```
figure;  
s = sparameters(ant, linspace(2.2e9, 2.75e9, 31));  
rfplot(s);
```

Calculate and Plot Pattern

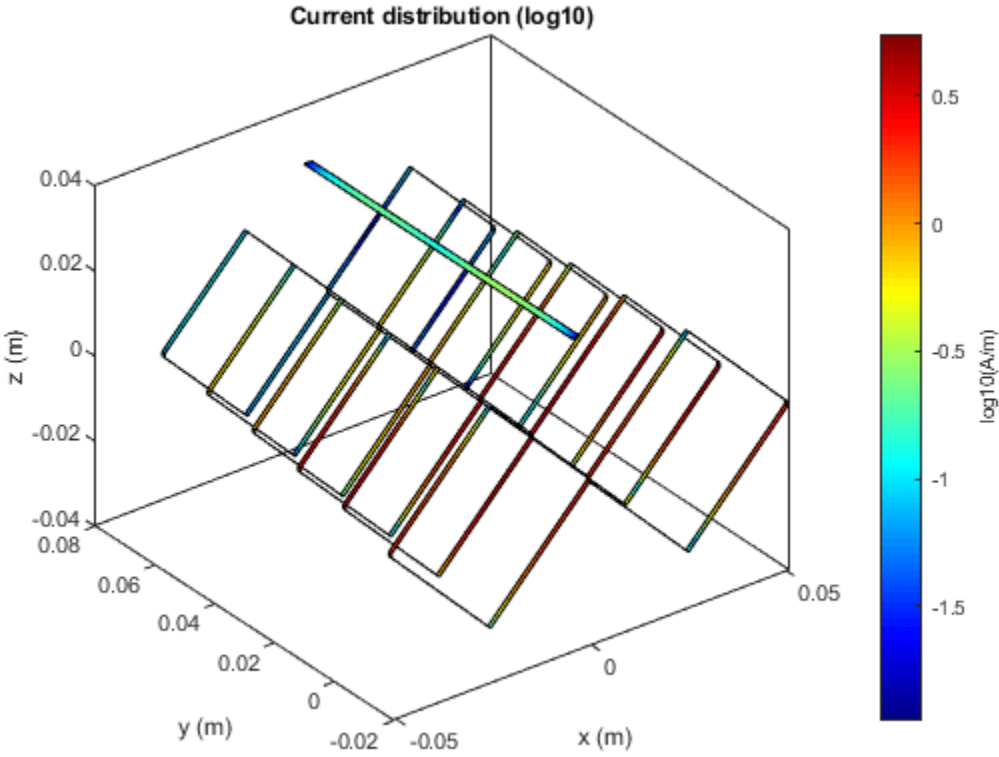
Plot the radiation pattern for this antenna at the frequency of best match in the band.

```
figure;  
pattern(ant,2.45e9);
```



Current Distribution

```
figure;  
current(ant, 2.45e9, 'scale', 'log10');
```



See Also

“Direct Search Based Optimization of Six-Element Yagi-Uda Antenna” on page 5-147

Analysis of Broad-wall Slotted Array Waveguide for High Frequency Applications

This example shows how to analyze the performance of a slotted waveguide antenna.

Create Slotted Waveguide

Create a slotted waveguide using defined geometry parameters.

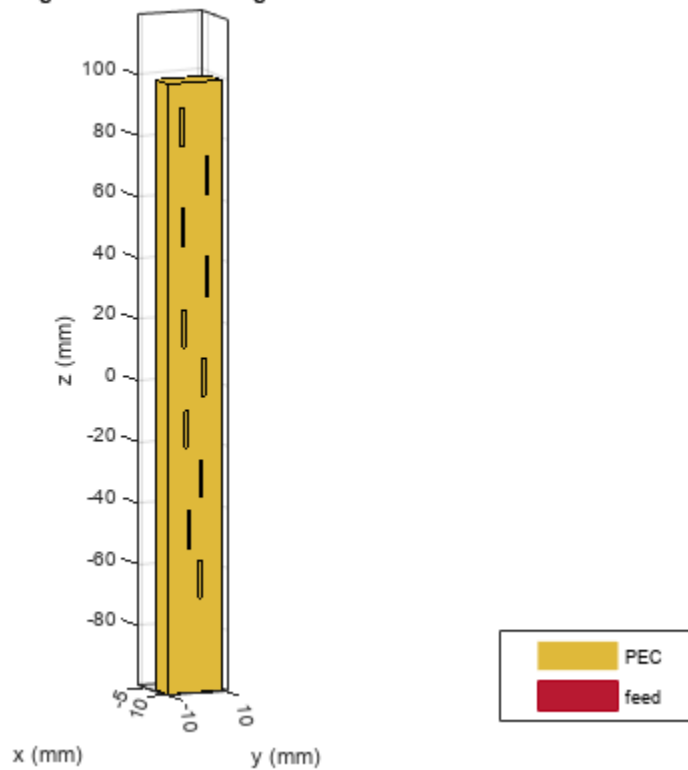
```
WG_w = 94e-3;      % Width of the waveguide
WG_h = 44e-3;      % Height of the waveguide
slot_L = 53e-3;    % Length of the slot
slot_w = 6.5e-3;   % Width of the slot
slot_off = 10e-3;  % Slot offset
r = antenna.Rectangle('Length',slot_L,'Width',slot_w); % Rectangular Slot
a = waveguideSlotted;
a.Tilt = 180;
a.TiltAxis = [1 0 1];
```

Longitudinal Linear Traveling Wave Slotted Waveguide

The array consists of radiating slots of different electrical lengths. Due to inherent property, each of them will resonate at their own individual resonance frequency. If the slot length and positions are chosen in such a way that the lower cut-off frequency and higher cut-off frequency of the n th slot overlaps with the higher and lower cut-off frequencies of the $(n-1)$ th and $(n+1)$ th slots respectively, then the complete array is expected to give a wide band response resulting in log-periodic dipole array. Slot Offset on broad wall slots are parallel to the waveguide centerline and they are blocking the transversal current components on the waveguide's broad wall. The polarization of these slots is vertical when the waveguide is held parallel to the ground and the transversal current component is zero on the centerline of the broad wall, however, if one slot moves closer to the narrow walls, the transversal current component increases. Therefore, the radiation amplitude of these slots increases as they move away from the centerline and that is why they are called offset slots. This is the most widely used slot type and this design is taken from [1].

```
r1 = antenna.Rectangle('Length',12.7e-3,'Width',1e-3);
r2 = antenna.Rectangle('Length',12.6e-3,'Width',1e-3);
r3 = antenna.Rectangle('Length',12.5e-3,'Width',1e-3);
r4 = antenna.Rectangle('Length',12.6e-3,'Width',1e-3);
r5 = antenna.Rectangle('Length',12.4e-3,'Width',1e-3);
r6 = antenna.Rectangle('Length',12.3e-3,'Width',1e-3);
r7 = antenna.Rectangle('Length',12.2e-3,'Width',1e-3);
r8 = antenna.Rectangle('Length',12e-3,'Width',1e-3);
r9 = antenna.Rectangle('Length',12e-3,'Width',1e-3);
r10 = antenna.Rectangle('Length',12.2e-3,'Width',1e-3);
%create antenna
ant1 = waveguideSlotted('Length',199.37e-3,'Width',19e-3, 'NumSlots',10,...
    'Height',9.5e-3,'Slot',[r1 r2 r3 r4 r5 r6 r7 r8 r9 r10] , 'SlotToTop',14.476e-3,...
    'SlotSpacing',16.45e-3, 'SlotOffset', ...
    [4.5e-3 4.3e-3 4.1e-3 4.5e-3 3.7e-3 3.3e-3 2.9e-3 2.3e-3 1.9e-3 1.9e-3], ...
    'FeedHeight',6.6e-3, 'ClosedWaveguide',0, 'FeedOffset',[-91.5e-3 0], ...
    'FeedWidth',0.04e-3, 'Tilt',180, 'TiltAxis',[1 0 1]);
figure
show(ant1);
title('Traveling wave slotted waveguide Antenna');
```

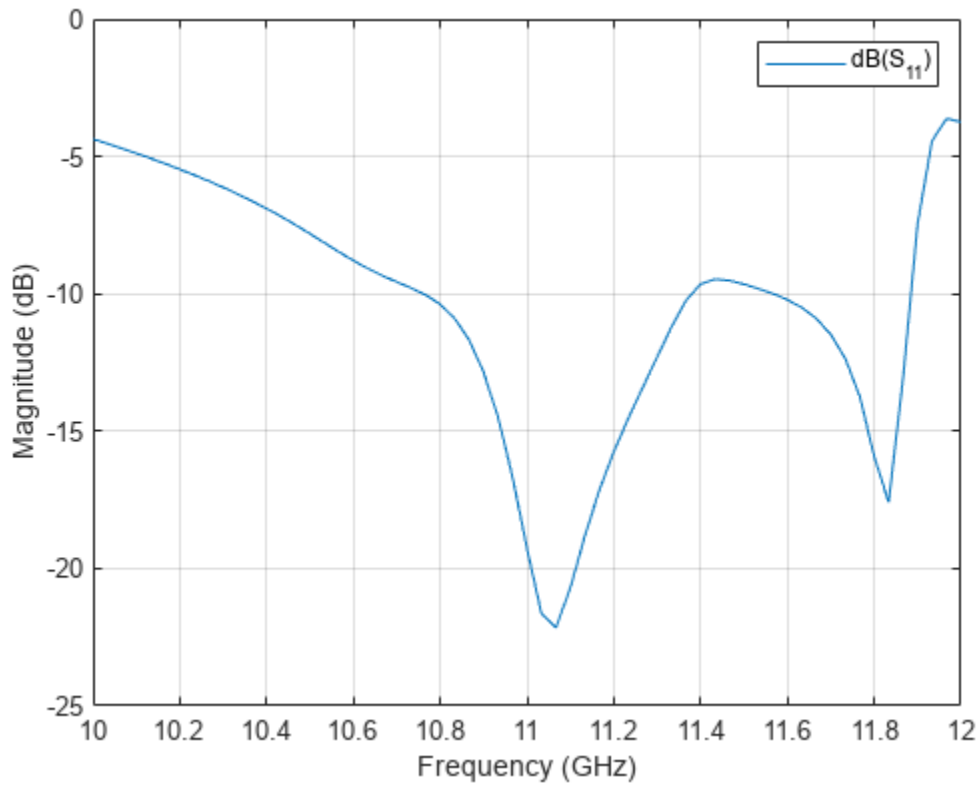
Traveling wave slotted waveguide Antenna



Plot Reflection Coefficient

Plot the reflection coefficient for this antenna over the frequency band of 10 GHz to 12 GHz and a reference impedance of 50 ohms.

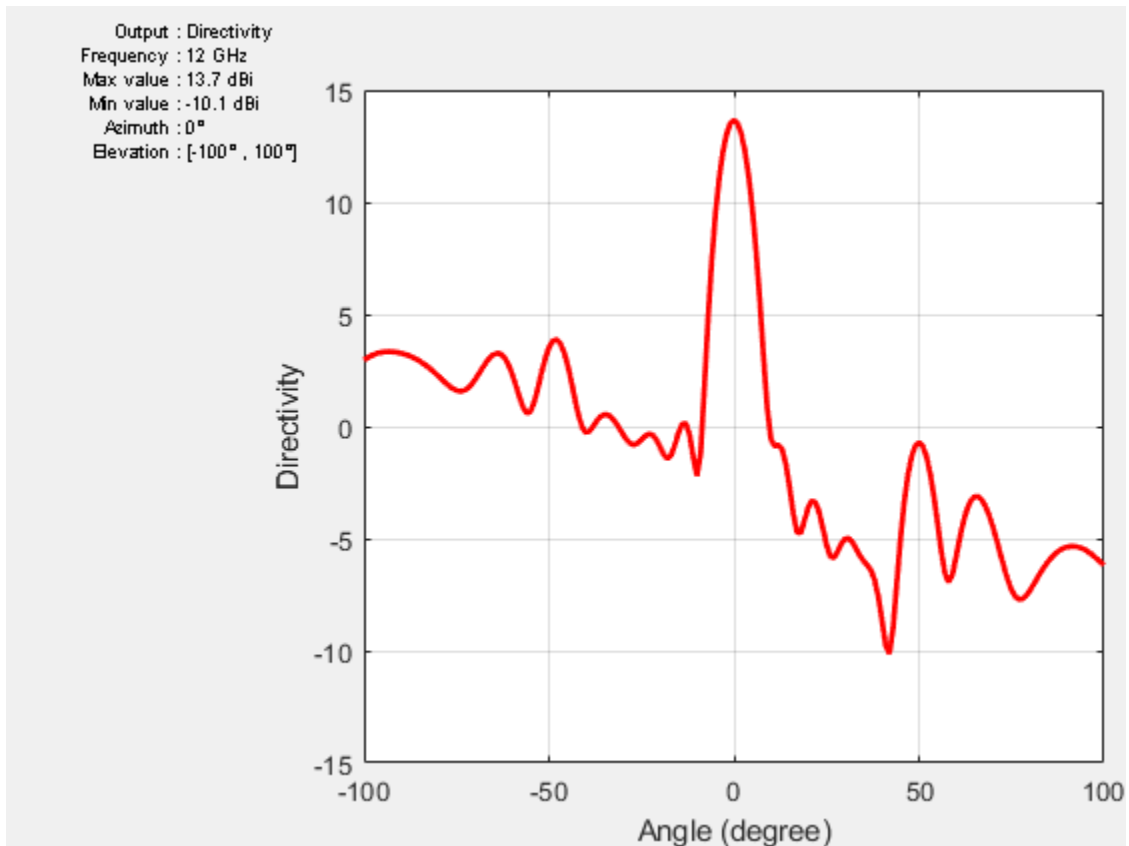
```
s = sparameters(ant1, linspace(10e9, 12e9, 61));  
figure  
rfplot(s);
```



Radiation pattern

The most significant effect to be considered in the design process are internal and external mutual coupling between slots. The internal mutual couplings are caused by the partial reflections of the incident electromagnetic wave from succeeding slots in a waveguide. These partial reflections cause a considerable displacement of the EM field inside the waveguide.

```
figure  
pattern(ant1,12e9,0,-100:100,'CoordinateSystem','Rectangular');
```



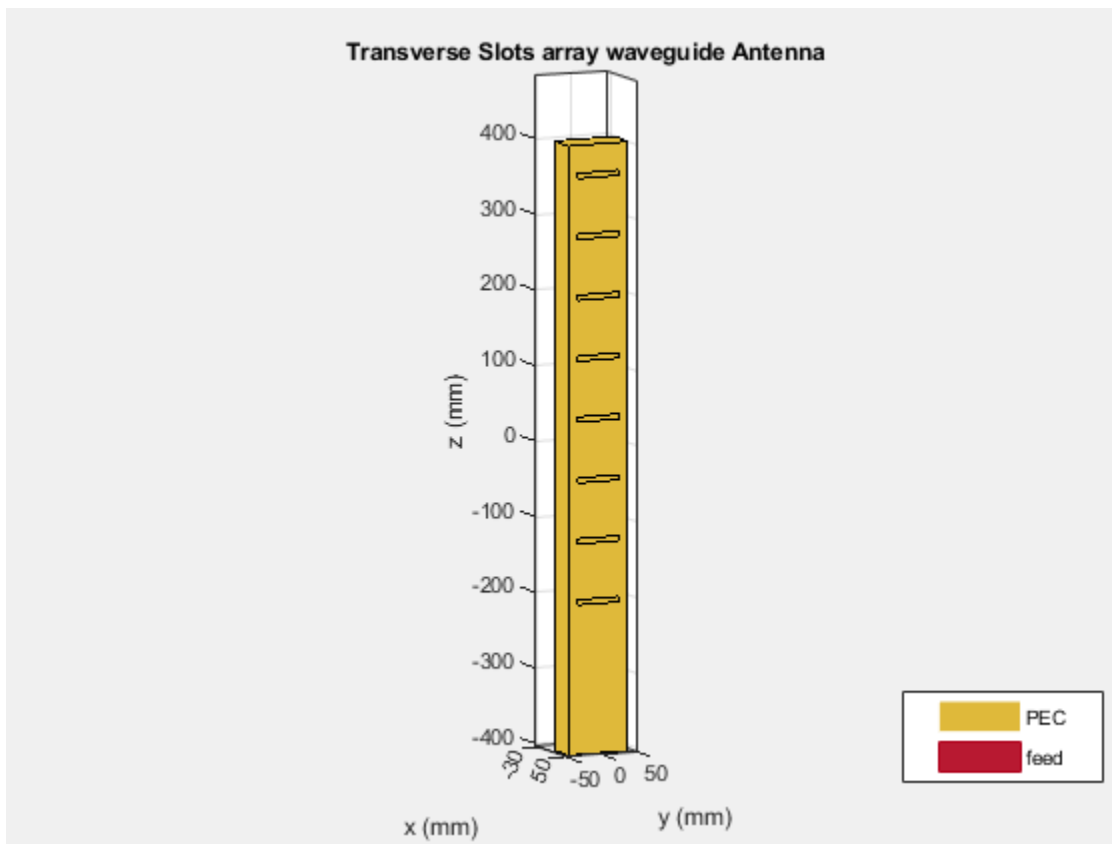
Create custom slots in waveguide

Create Transverse and X-shaped slots.

Transverse Slots Array

Transverse slots result in a very high value of their normalized resistance and they cannot be matched to the characteristic waveguide impedance. So, they have no practical importance.

```
a.SlotAngle = 90;
a.SlotOffset = 0;
show(a);
title('Transverse Slots array waveguide Antenna');
```



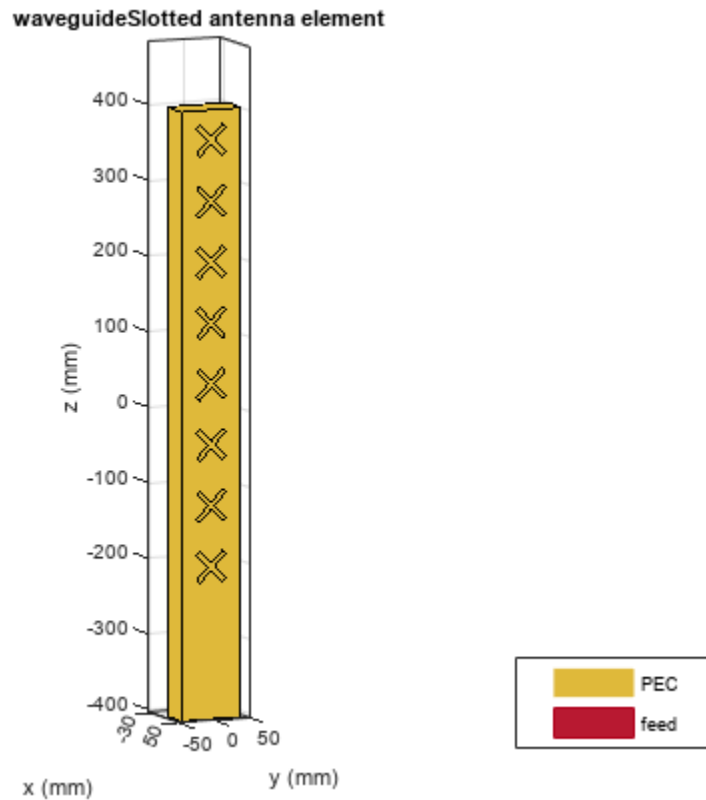
X-shaped Slots

The cross(X) slots can generate a circular polarized wave with a good axial ratio performance. The slots are oriented to form an orthogonal pair of slots which eventually generate a circular polarized wave. The theory of cross slots also suggests that the slots should be ideally equal to half of the free space wavelength.

```

r1 = antenna.Rectangle('Length',0.053,'Width',0.0065);
r1 = rotateZ(r1,45);
r2 = antenna.Rectangle('Length',0.053,'Width',0.0065);
r2 = rotateZ(r2,-45);
temp2 = r1+r2;
a.Slot = temp2;
a.SlotAngle = 90;
a.SlotOffset = 0;
figure
show(a);

```

Conclusion

The models of the Slotted waveguide antenna have been built and analyzed and agree well with results reported from [1].

References

- [1] Montesinos Ortego, "Contribution to the design of waveguide fed compound slot arrays by means of equivalent circuit modeling".
- [2] Zunnurain Ahmad, "Design and Implementation of Quasi Planar K-Band Array Antenna Based on Travelling Wave Structures".

See Also

"Sector Antenna for 2.4 GHz Wi-Fi™" on page 5-107

Urban Link and Coverage Analysis Using Ray Tracing

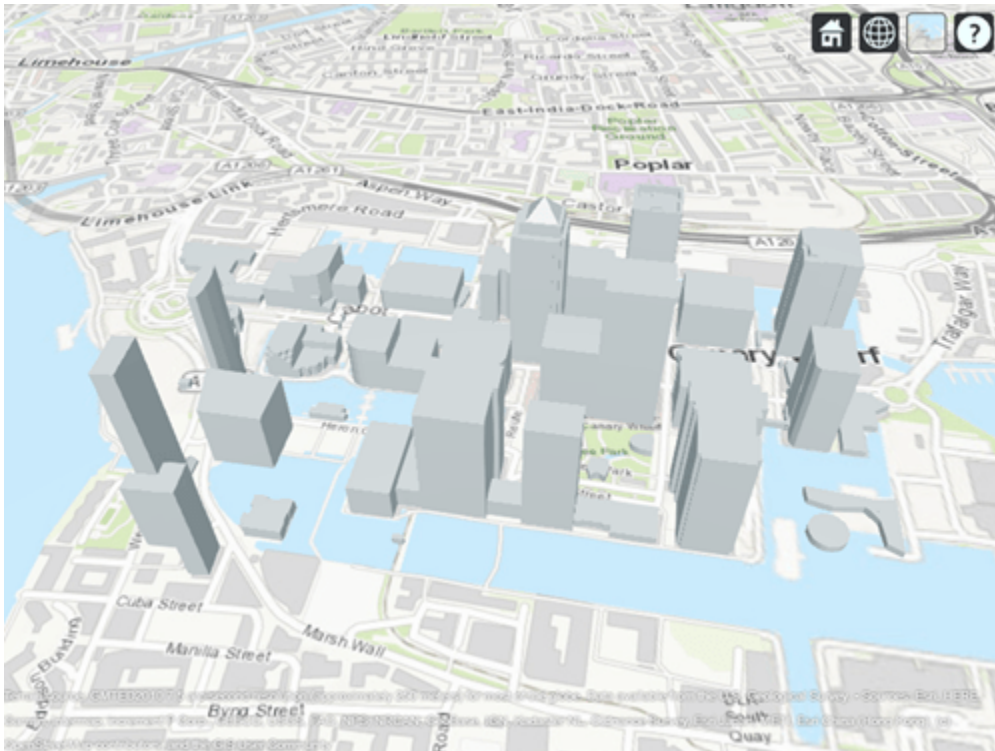
This example shows how to use ray tracing to analyze communication links and coverage areas in an urban environment. Within the example:

- Import and visualize 3-D buildings data into Site Viewer
- Define a transmitter site and ray tracing propagation model corresponding to a 5G urban scenario
- Analyze a link in non-line-of-sight conditions
- Visualize coverage using the shooting and bouncing rays (SBR) ray tracing method with different numbers of reflections, diffractions, and launched rays
- Optimize a non-line-of-sight link using beam steering and Phased Array System Toolbox™

Import and Visualize Buildings Data

Import an OpenStreetMap (.osm) file corresponding to Canary Wharf in London, UK. The file was downloaded from <https://www.openstreetmap.org>, which provides access to crowd-sourced map data all over the world. The data is licensed under the Open Data Commons Open Database License (ODbL), <https://opendatacommons.org/licenses/odbl/>. The buildings information contained within the OpenStreetMap file is imported and visualized in Site Viewer.

```
viewer = siteviewer("Buildings","canarywharf.osm","Basemap","topographic");
```

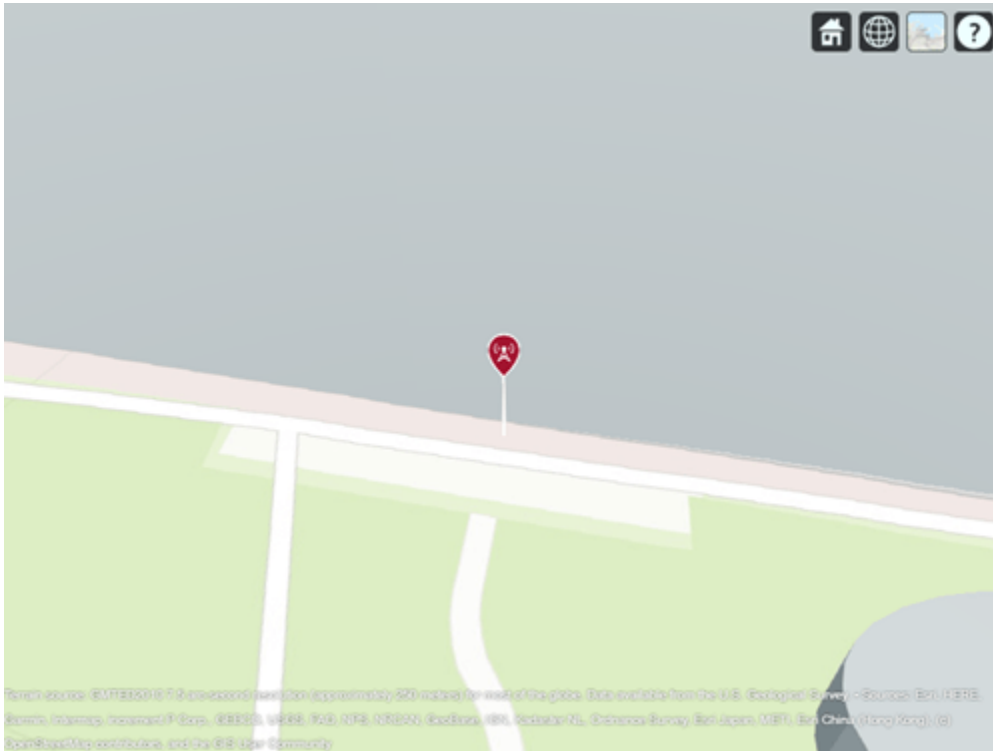


Define Transmitter Site

Define a transmitter site to model a small cell scenario in a dense urban environment. The transmitter site represents a base station that is placed on a pole servicing the surrounding area

which includes a neighboring park. The transmitter uses the default isotropic antenna, and operates at a carrier frequency of 28 GHz with a power level of 5 W.

```
tx = txsite("Name","Small cell transmitter", ...
  "Latitude",51.50375, ...
  "Longitude",-0.01843, ...
  "AntennaHeight",10, ...
  "TransmitterPower",5, ...
  "TransmitterFrequency",28e9);
show(tx)
```



View Coverage Map for Line-of-Sight Propagation

Create a ray tracing propagation model using the shooting and bouncing rays (SBR) method. The SBR propagation model uses ray tracing analysis to compute propagation paths and their corresponding path losses. Path loss is calculated from free-space loss, reflection and diffraction loss due to interactions with materials, and antenna polarization loss.

Limit the initial analysis to only line-of-sight propagation paths by setting the maximum number of reflections to 0. Set the building and terrain material types to model perfect electrical conductors.

```
rtpm = propagationModel("raytracing", ...
  "Method","sbr", ...
  "MaxNumReflections",0, ...
  "BuildingsMaterial","perfect-reflector", ...
  "TerrainMaterial","perfect-reflector");
```

View the corresponding coverage map for a maximum range of 250 meters from the base station. The coverage map shows received power for a receiver at each ground location but is not computed for building tops or sides.

```
coverage(tx,rtpm, ...
    "SignalStrengths",-120:-5, ...
    "MaxRange",250, ...
    "Resolution",3, ...
    "Transparency",0.6)
```



Define Receiver Site in Non-Line-of-Sight Location

The coverage map for line-of-sight propagation shows shadowing due to obstructions. Define a receiver site to model a mobile receiver in an obstructed location. Plot the line-of-sight path to show the obstructed path from the transmitter to the receiver.

```
rx = rxsite("Name","Small cell receiver", ...
    "Latitude",51.50216, ...
    "Longitude",-0.01769, ...
    "AntennaHeight",1);
```

```
los(tx,rx)
```



Plot Propagation Path Using Ray Tracing

Adjust the ray tracing propagation model to include single-reflection paths, then plot the rays. The result shows signal propagation along a single-reflection path. View the corresponding propagation characteristics, including the received power, phase change, distance, and angles of departure and arrival, by clicking on the plotted path.

```

rtpm.MaxNumReflections = 1;
clearMap(viewer)
raytrace(tx,rx,rtpm)

```



Analyze Signal Strength and Effect of Materials

Compute the received power.

```
ss = sigstrength(rx,tx,rtpm);
disp("Received power using perfect reflection: " + ss + " dBm")
```

Received power using perfect reflection: -70.392 dBm

Update the model to use concrete for the buildings and terrain materials. Then, update the rays shown in Site Viewer.

```
rtpm.BuildingsMaterial = "concrete";
rtpm.TerrainMaterial = "concrete";
raytrace(tx,rx,rtpm)
```

Recalculate the received power. The use of realistic material reflection results in approximately 8 dB of loss compared to perfect reflection.

```
ss = sigstrength(rx,tx,rtpm);
disp("Received power using concrete materials: " + ss + " dBm")
```

Received power using concrete materials: -78.4999 dBm

Include Weather Loss

Adding weather impairments to the propagation model and re-computing the received power results in another 1.5 dB of loss.

```
rtPlusWeather = ...
    rtpm + propagationModel("gas") + propagationModel("rain");
```

```
raytrace(tx, rx, rtPlusWeather)
```

```
ss = sigstrength(rx, tx, rtPlusWeather);
disp("Received power including weather loss: " + ss + " dBm")
```

Received power including weather loss: -80.0172 dBm

Plot Propagation Paths Including Two Reflections

Expand the point-to-point analysis to include two-reflection paths and choose a smaller angular separation between launched rays for the SBR method. The visualization shows two clusters of propagation paths. The total received power for two-reflection paths is similar to the total received power for single-reflection paths.

```
rtPlusWeather.PropagationModels(1).MaxNumReflections = 2;
rtPlusWeather.PropagationModels(1).AngularSeparation = "low";
```

```
ss = sigstrength(rx, tx, rtPlusWeather);
disp("Received power with two-reflection paths: " + ss + " dBm")
```

Received power with two-reflection paths: -79.6847 dBm

```
clearMap(viewer)
raytrace(tx, rx, rtPlusWeather)
```



Plot Propagation Paths Including Two Reflections and One Diffraction

Expand the point-to-point analysis to include paths with one edge diffraction. The visualization shows the two stronger clusters of propagation paths due to reflections with no diffraction, plus weaker clusters involving one diffraction. The total received power is not changed significantly by the inclusion of one diffraction.

```

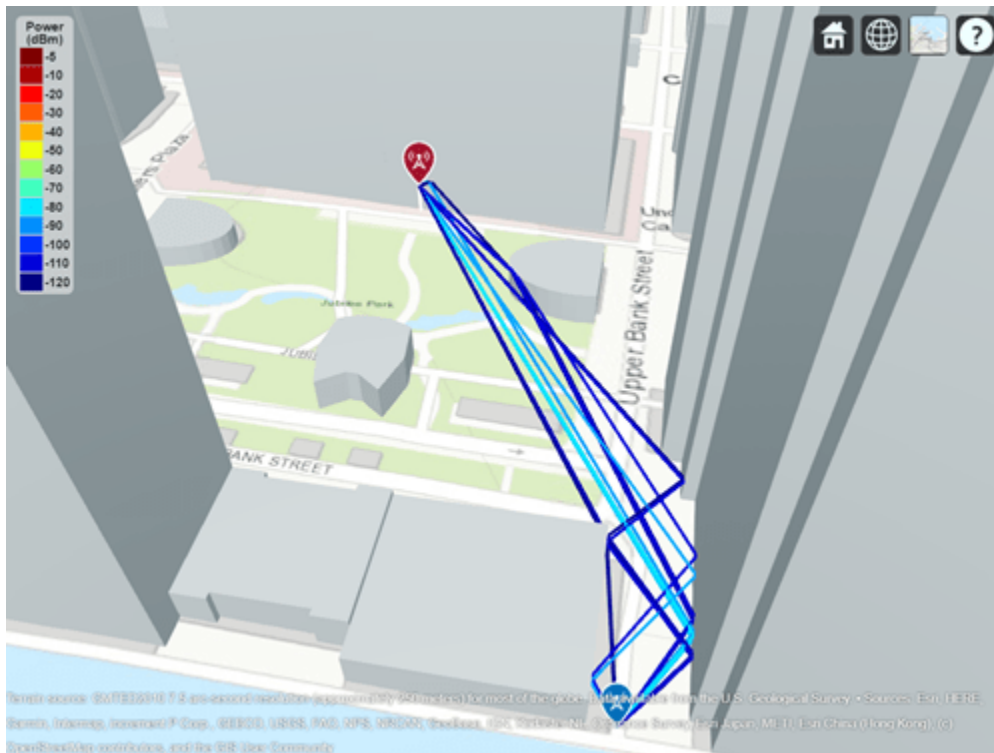
rtPlusWeather.PropagationModels(1).MaxNumDiffractions = 1;

ss = sigstrength(rx,tx,rtPlusWeather);
disp("Received power with two-reflection and one-diffraction paths: " + ss + " dBm")

Received power with two-reflection and one-diffraction paths: -79.8526 dBm

raytrace(tx,rx,rtPlusWeather)

```



View Coverage Map with Single-Reflection Paths

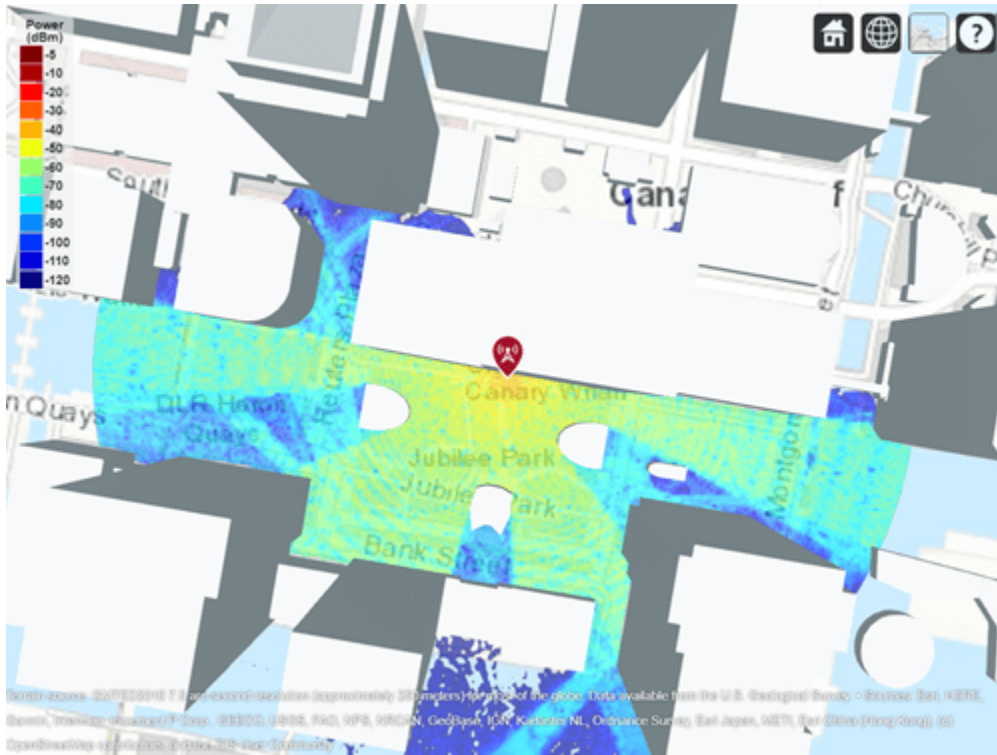
Use the configured propagation model and generate a coverage map that includes single-reflection paths and weather impairments.

```

rtPlusWeather.PropagationModels(1).MaxNumReflections = 1;
rtPlusWeather.PropagationModels(1).MaxNumDiffractions = 0;
clearMap(viewer)
show(tx)

coverage(tx,rtPlusWeather, ...
    "SignalStrengths",-120:-5, ...
    "MaxRange", 250, ...
    "Resolution",2, ...
    "Transparency",0.6)

```

Use Beam Steering to Enhance Received Power

Many modern communications systems use techniques to steer the transmitter antenna to achieve optimal link quality. This section uses Phased Array System Toolbox™ to optimally steer a beam to maximize received power for a non-line-of-sight link.

Define a custom antenna from Report ITU-R M.2412 [1] on page 5-623 for evaluating 5G radio technologies. Create an 8-by-8 uniform rectangular array with half-wavelength element spacing using the pattern defined in Section 8.5 of the report.

```
azvec = -180:180; % Azimuth angles (deg)
elvec = -90:90; % Elevation angles (deg)
SLA = 30; % Maximum side-lobe level attenuation (dB)
tilt = 0; % Tilt angle (deg)
az3dB = 65; % 3 dB beamwidth in azimuth (deg)
el3dB = 65; % 3 dB beamwidth in elevation (deg)
lambda = physconst("lightspeed")/tx.TransmitterFrequency; % Wavelength (m)

[az,el] = meshgrid(azvec,elvec);
azMagPattern = -min(12*(az/az3dB).^2,SLA);
elMagPattern = -min(12*((el-tilt)/el3dB).^2,SLA);
combinedMagPattern = -min(-(azMagPattern + elMagPattern),SLA); % Relative antenna gain (dB)

antennaElement = phased.CustomAntennaElement("MagnitudePattern",combinedMagPattern);
tx.Antenna = phased.URA("Size",[8 8], ...
    "Element",antennaElement, ...
    "ElementSpacing",[lambda/2 lambda/2]);
```

Calculate the peak directivity of the array.

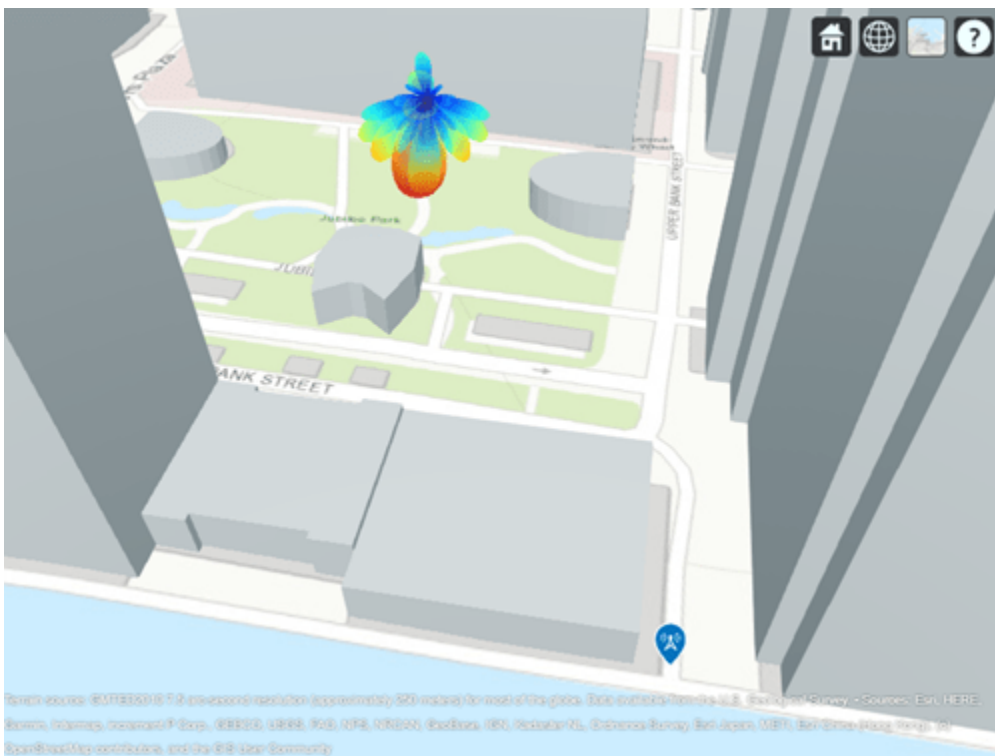
```
antennaDirectivity = pattern(tx.Antenna, tx.TransmitterFrequency);
antennaDirectivityMax = max(antennaDirectivity(:));
disp("Peak antenna directivity: " + antennaDirectivityMax + " dBi")
```

Peak antenna directivity: 23.4449 dBi

Point the antenna south and view the radiation pattern.

```
tx.AntennaAngle = -90;
```

```
clearMap(viewer)
show(rx)
pattern(tx, "Transparency", 0.6)
hide(tx)
```



Set the maximum number of reflections to 1 and maximum number of diffractions to 0 to perform analysis on the dominant single-reflection path. Call `raytrace` with an output to access the rays that were computed. The returned `comm.Ray` objects include both the geometric and propagation-related characteristics of each ray.

```
rtPlusWeather.PropagationModels(1).MaxNumReflections = 1;
rtPlusWeather.PropagationModels(1).MaxNumDiffractions = 0;
ray = raytrace(tx, rx, rtPlusWeather);
disp(ray{1})
```

Ray with properties:

```
PathSpecification: 'Locations'
CoordinateSystem: 'Geographic'
TransmitterLocation: [3x1 double]
ReceiverLocation: [3x1 double]
```

```
LineOfSight: 0
Interactions: [1x1 struct]
  Frequency: 2.8000e+10
PathLossSource: 'Custom'
  PathLoss: 117.0069
  PhaseShift: 4.0976
```

```
Read-only properties:
  PropagationDelay: 6.6488e-07
  PropagationDistance: 199.3261
  AngleOfDeparture: [2x1 double]
  AngleOfArrival: [2x1 double]
  NumInteractions: 1
```

Get the angle-of-departure for the single-reflection path and apply this angle to steer the antenna in the optimal direction to achieve higher received power. The angle-of-departure azimuth is offset by the physical antenna angle azimuth to convert it to the steering vector azimuth defined in the local coordinate system of the phased array antenna.

```
aod = ray{1}.AngleOfDeparture;
steeringaz = wrapTo180(aod(1)-tx.AntennaAngle(1));
steeringVector = phased.SteeringVector("SensorArray",tx.Antenna);
sv = steeringVector(tx.TransmitterFrequency,[steeringaz;aod(2)]);
tx.Antenna.Taper = conj(sv);
```

Show the antenna energy directed along the propagation path by plotting the radiation pattern. The new received power increases by more than 20 dB. The increased received power corresponds to the peak directivity of the antenna.

```
pattern(tx,"Transparency",0.6)
raytrace(tx,rx,rtPlusWeather);
hide(tx)

ss = sigstrength(rx,tx,rtPlusWeather);
disp("Received power with beam steering: " + ss + " dBm")

Received power with beam steering: -57.0575 dBm
```


[1] Report ITU-R M.2412, "Guidelines for evaluation of radio interface technologies for IMT-2020", 2017. <https://www.itu.int/pub/R-REP-M.2412>

See Also

Functions

propagationModel | raytrace | coverage | contour | pattern

Objects

siteviewer | txsite | rxsite

Related Examples

- "Ray Tracing for Wireless Communications" on page 4-12
- "3D Reconstruction of Radiation Pattern From 2D Orthogonal Slices" on page 5-497

Maximizing Gain and Improving Impedance Bandwidth of E-Patch Antenna

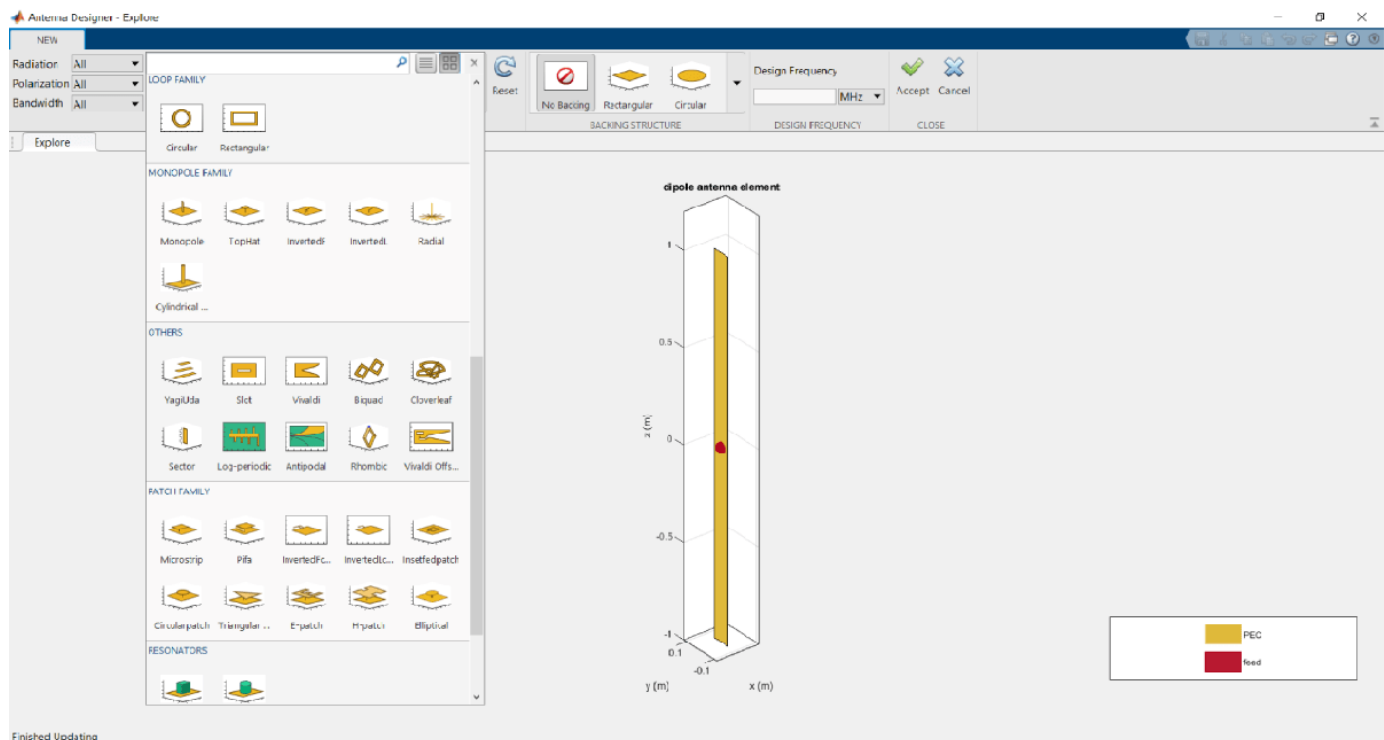
This example shows how to optimize an E-patch antenna in the **Antenna Designer** app using the SADEA optimizer.

Antenna Selection

Enter `antennaDesigner` at the MATLAB command prompt to open the app.

In the blank canvas, click **NEW**.

From the **ANTENNA GALLERY dropdown**, under **PATCH FAMILY**, select an E-patch antenna.

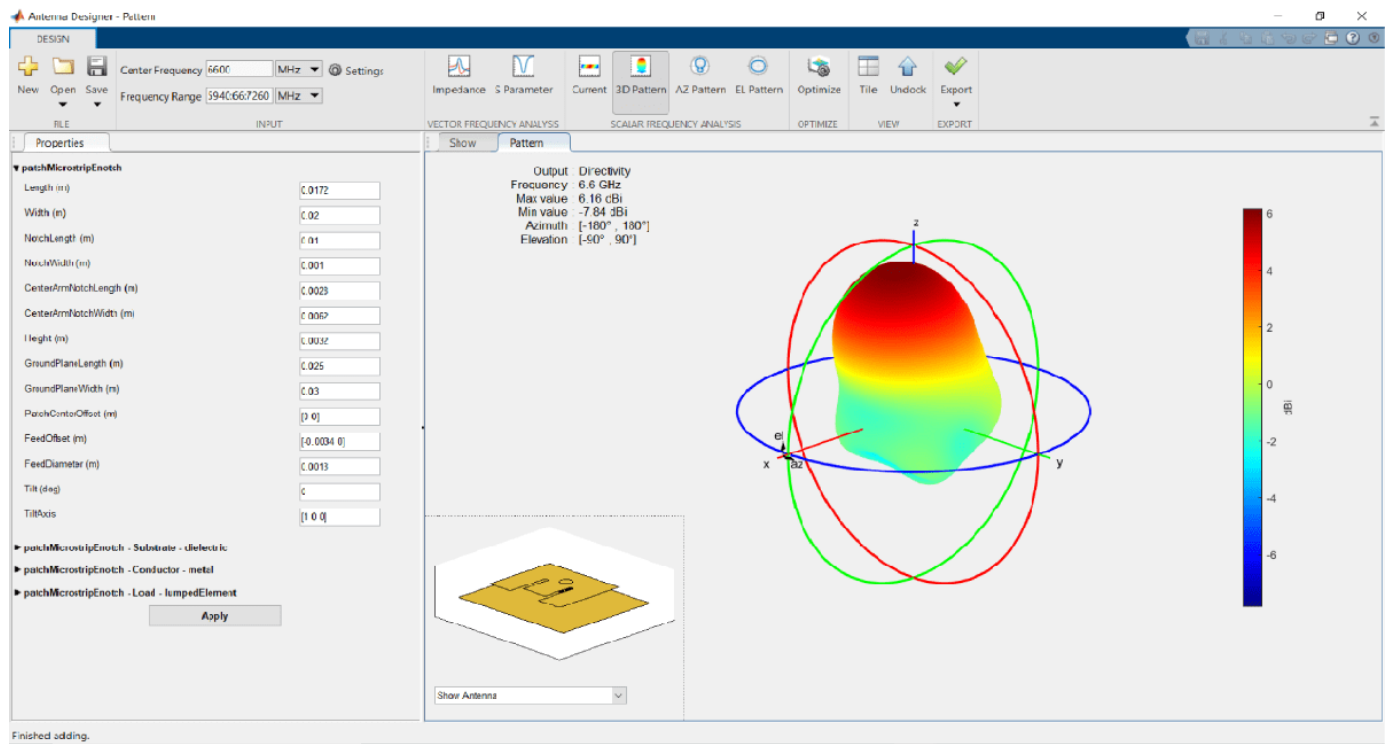


To analyze the antenna, click **Accept**.

Analyze 3D Radiation Pattern

Click on **3D Pattern**. Observe the radiation pattern and the maximum gain.

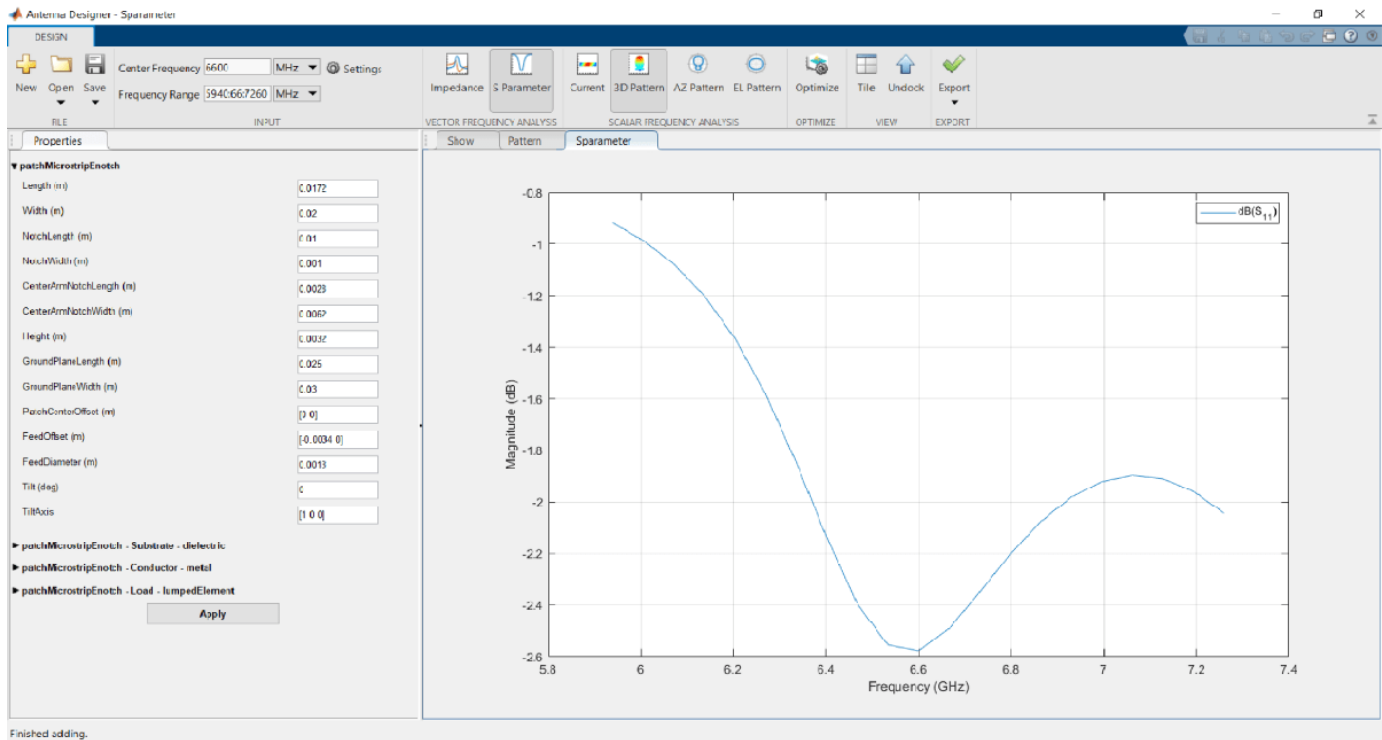
5 Antenna Toolbox Examples



The maximum gain of the antenna is **6.16 dBi** as shown on the top left side of the pattern figure.

Analyze S11

Click **S Parameter**.



Observe that the S11 of the antenna do not show any impedance bandwidth within the frequency range and that the current E-patch antenna design might not be an optimized one. To improve the gain and the impedance bandwidth, optimize the antenna using the optimizer tab.

Setup Optimization Problem

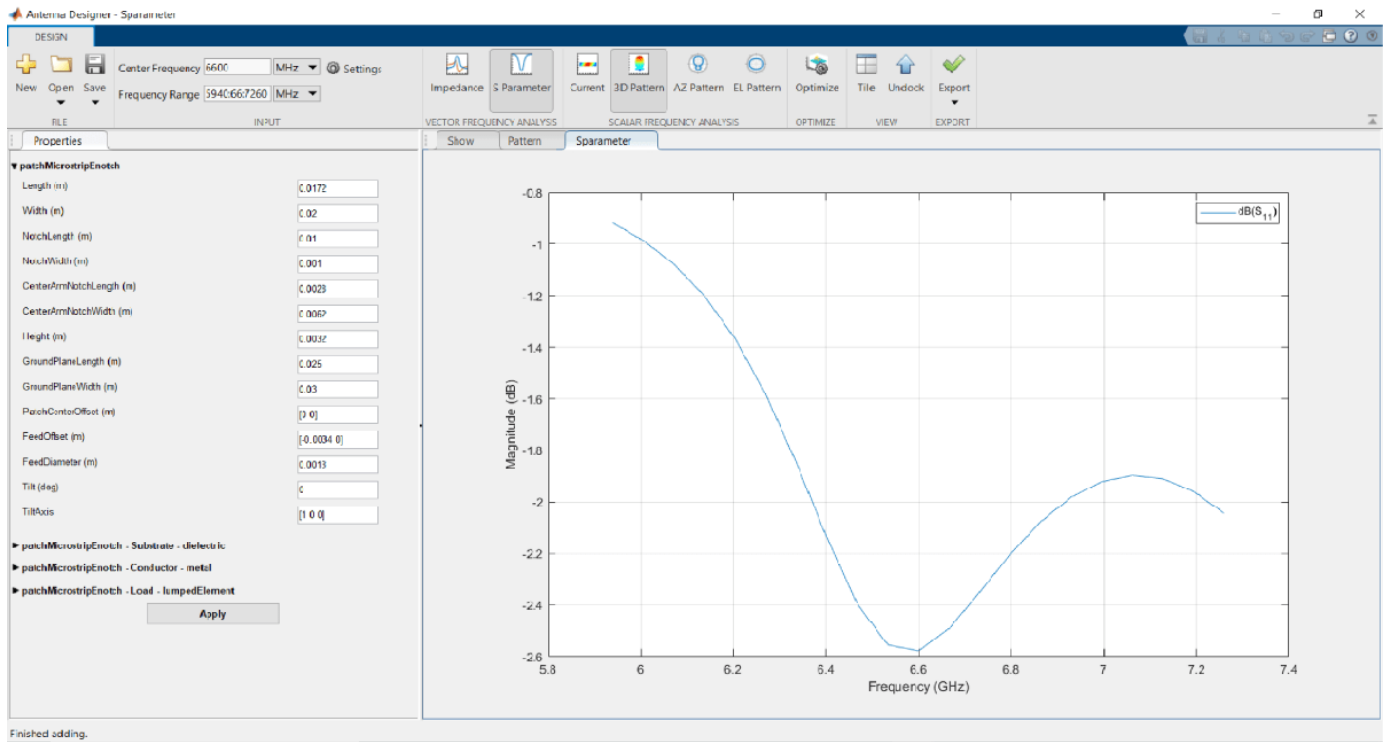
Any optimization problem typically requires following inputs.

- 1 Objective function: Main goal of the optimization. It evaluates the analysis function and minimizes or maximizes the output of the function. **In this example, maximizing the gain of the antenna is the objective function.**
- 2 Design variables: The input variables to the objective function. These variables are changed by the optimizer within a pre-set range of values called as the bounds of the variables. **In this example, the dimensions of the E-patch are the design variables.**
- 3 Constraint functions (if necessary): Functions which restrict a desired analysis function value on the antenna. **In this example, the constraint function is S11 less than -10 db to obtain an impedance bandwidth.**
- 4 Other inputs: Other inputs may include the number of iterations, the input center frequency, and the input frequency, the number of iterations, the frequency at which the analysis is performed, etc.

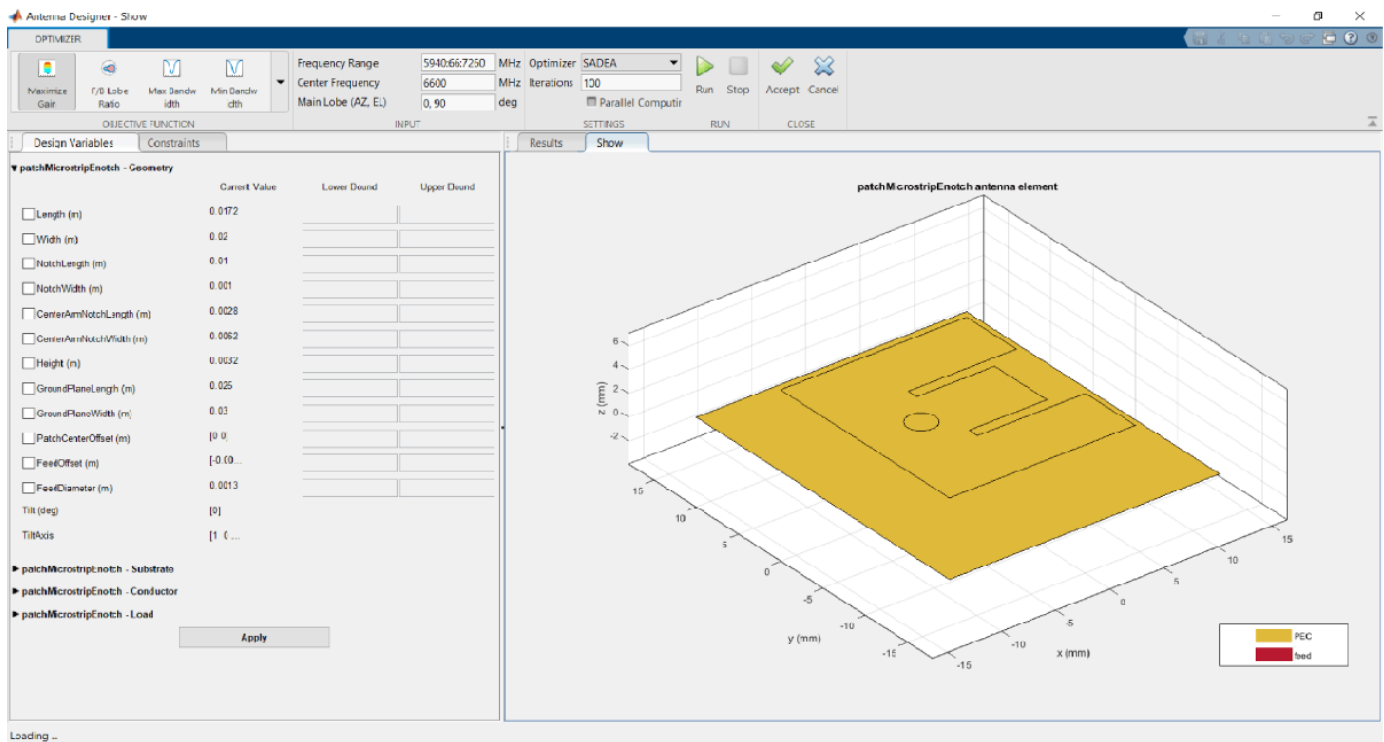
Optimization Function

To optimize the E patch, click on the **Optimize** button.

5 Antenna Toolbox Examples



To select an objective function, use the **OBJECTIVE FUNCTION** Gallery drop down. Since the goal is to maximize the gain of the antenna, click on **Maximize Gain**.



Design Variables

To set up the design variables, click on the **Design Variables** tab. Click on the checkboxes present on the left-hand side of the properties to choose the required design variables. The optimizer would change these chosen properties to obtain a maximum gain for the antenna.

Enter bounds for the design variables as follows:

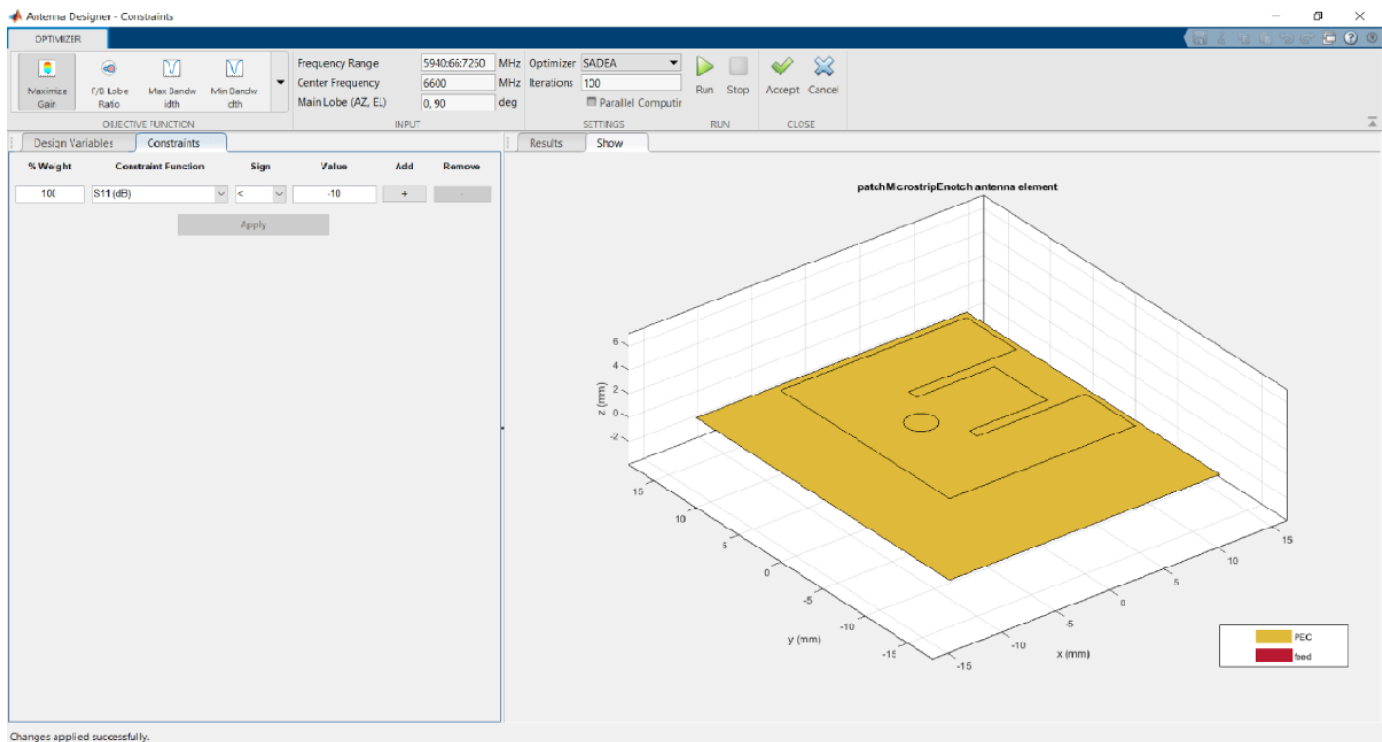
	Lower Bound	Upper Bound
Length	0.016	0.02
Width	0.018	0.024
NotchLength	0.0089	0.019
NotchWidth	0.001	0.002
CenterArmNotchLength	0.002	0.003
CenterArmNotchWidth	0.0058	0.007

Click **Apply** to set the variables.

Constraints

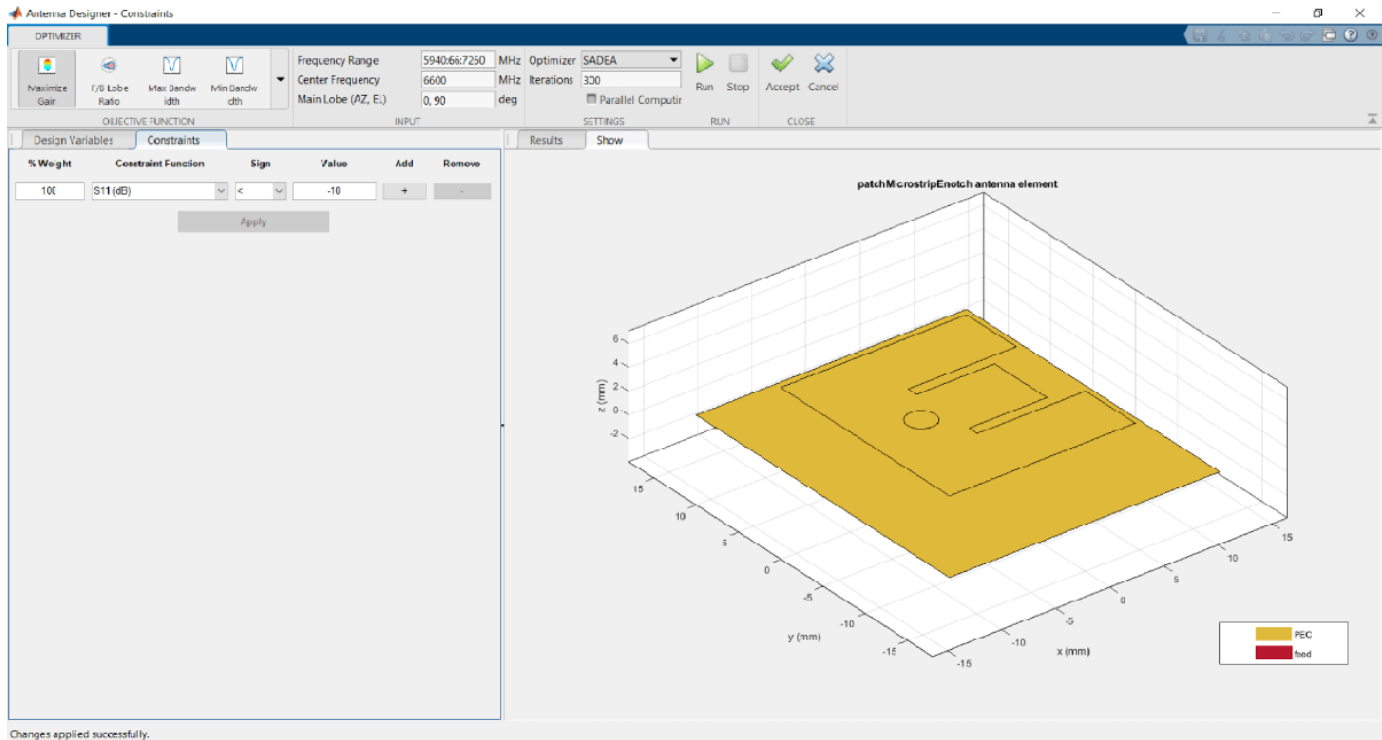
To set up the constraints, click on the **Constraints** tab.

Select **S11 (dB)** from the **Constraint Function**. Select < operator from **sign** and enter value as **-10**.



Click **Apply**.

In the **SETTINGS** section, enter **Iterations** as **300** to give the number of iterations to run for the optimizer. Click **Parallel Computing** in **Settings** section if you have Parallel Computing Toolbox™. To start the start optimization, click the **Run** button.



Optimization

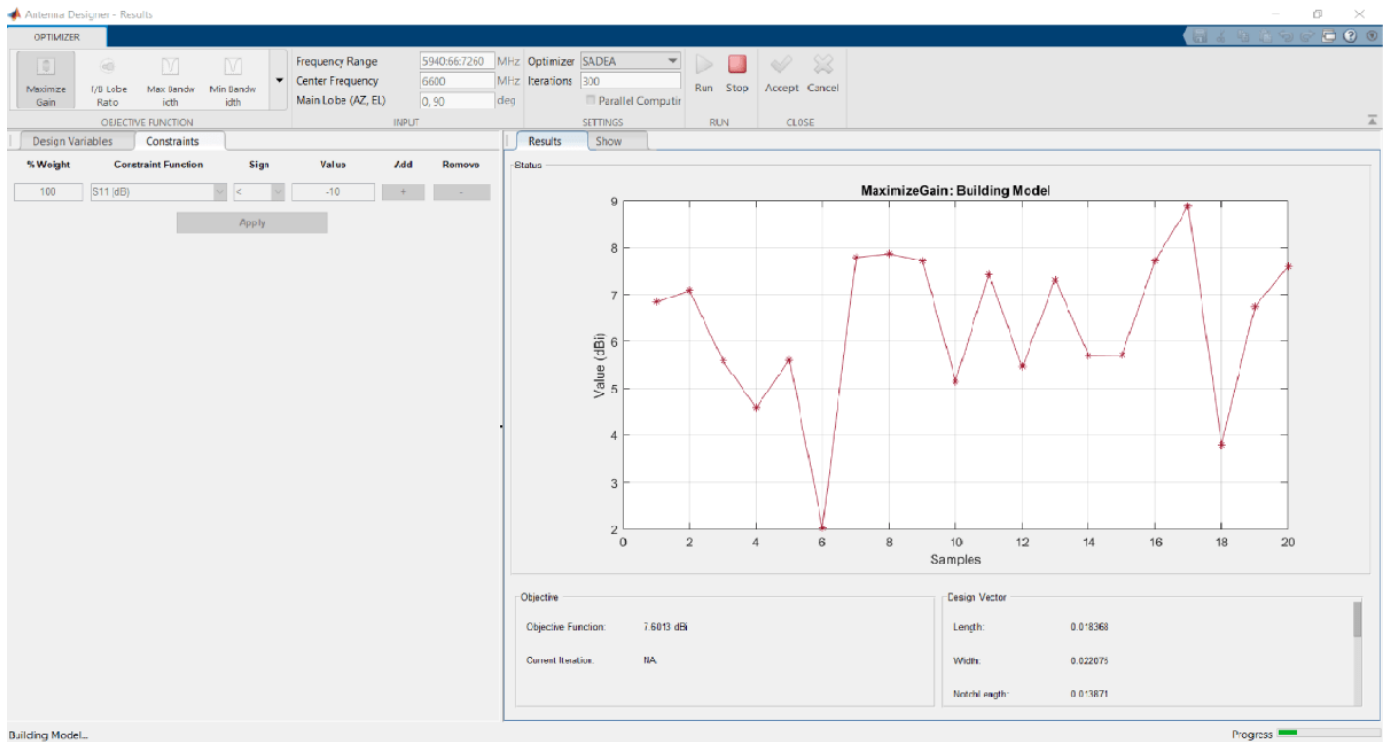
The **SADEA** is a surrogate model-based optimization which contains two stages

- 1 Building model
- 2 Optimizing

Building Model:

In the model building stage, the optimizer makes a surrogate model from the design space, and the specified objective and the constraints function. It diversely goes through the design space and performs analysis on these sample points.

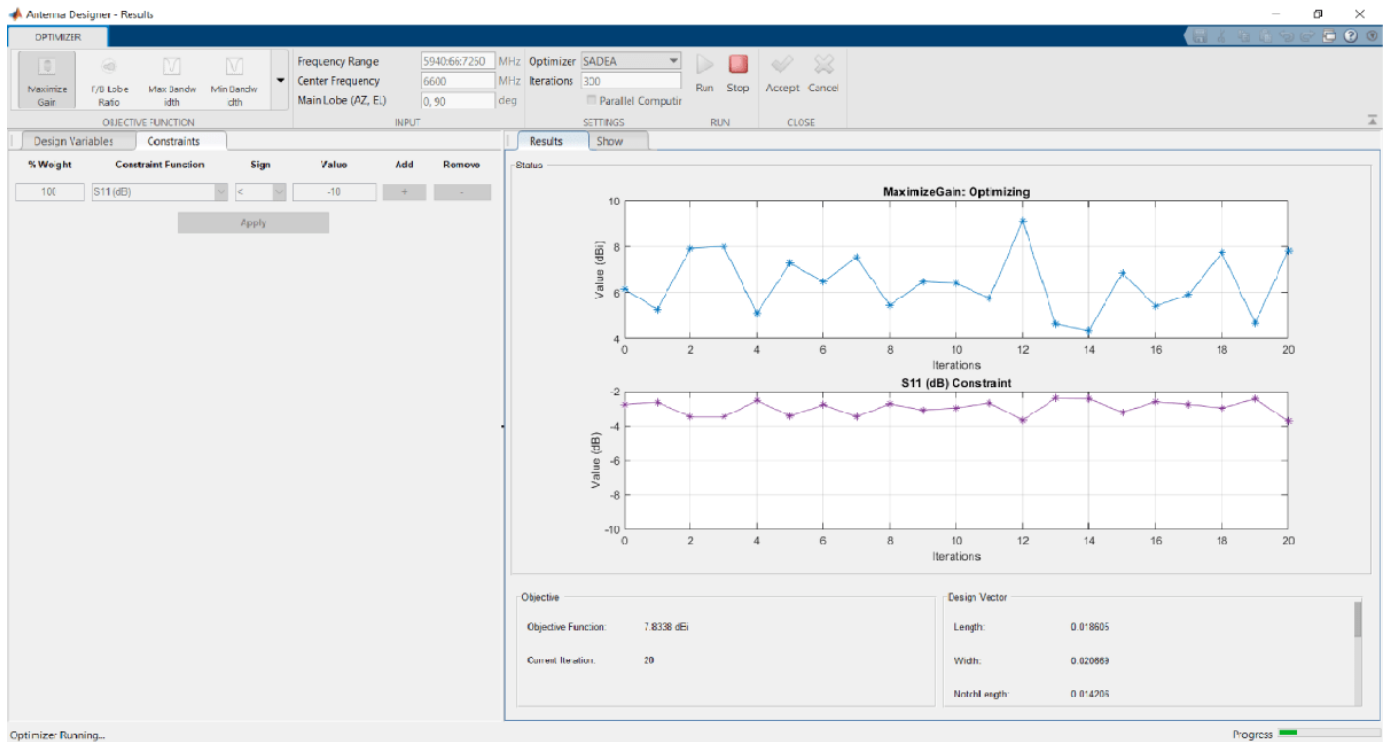
So, the X-axis shows the number of samples and the Y-axis shows the value of the analysis function value at that sample. The bottom left side show the current sample value and the bottom right side shows the design variables. The optimizer within decides and takes appropriate number of samples to build the model. After the model is built, the optimizer starts running iterations.



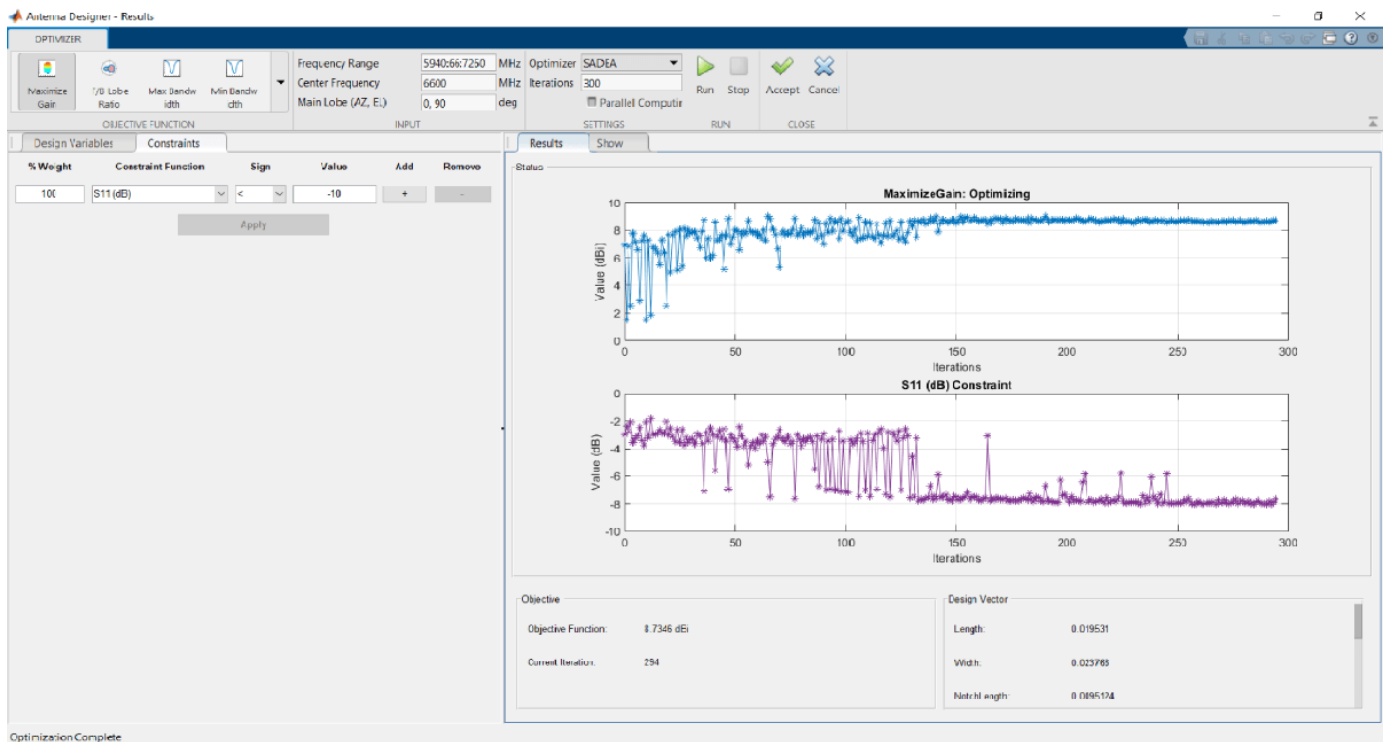
Optimizing

In the optimizing stage, the X-axis shows the number of iterations and the Y-axis shows the objective function values. From the plots shown on the optimizing stage, you can understand the trend of convergence.

5 Antenna Toolbox Examples

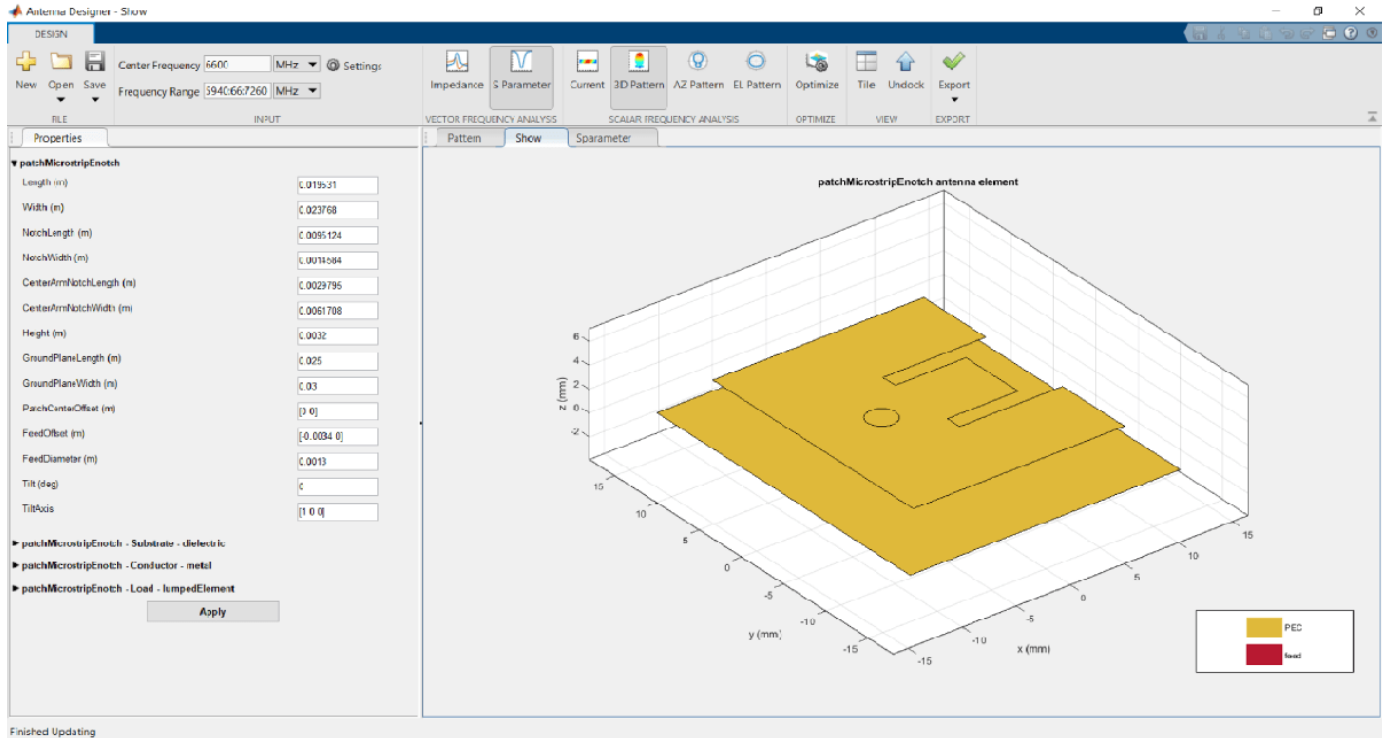


Notice that, the objective as well as the constraint plots show fewer deviations from after the 150th iteration. This indicates that the objective is **converging**.



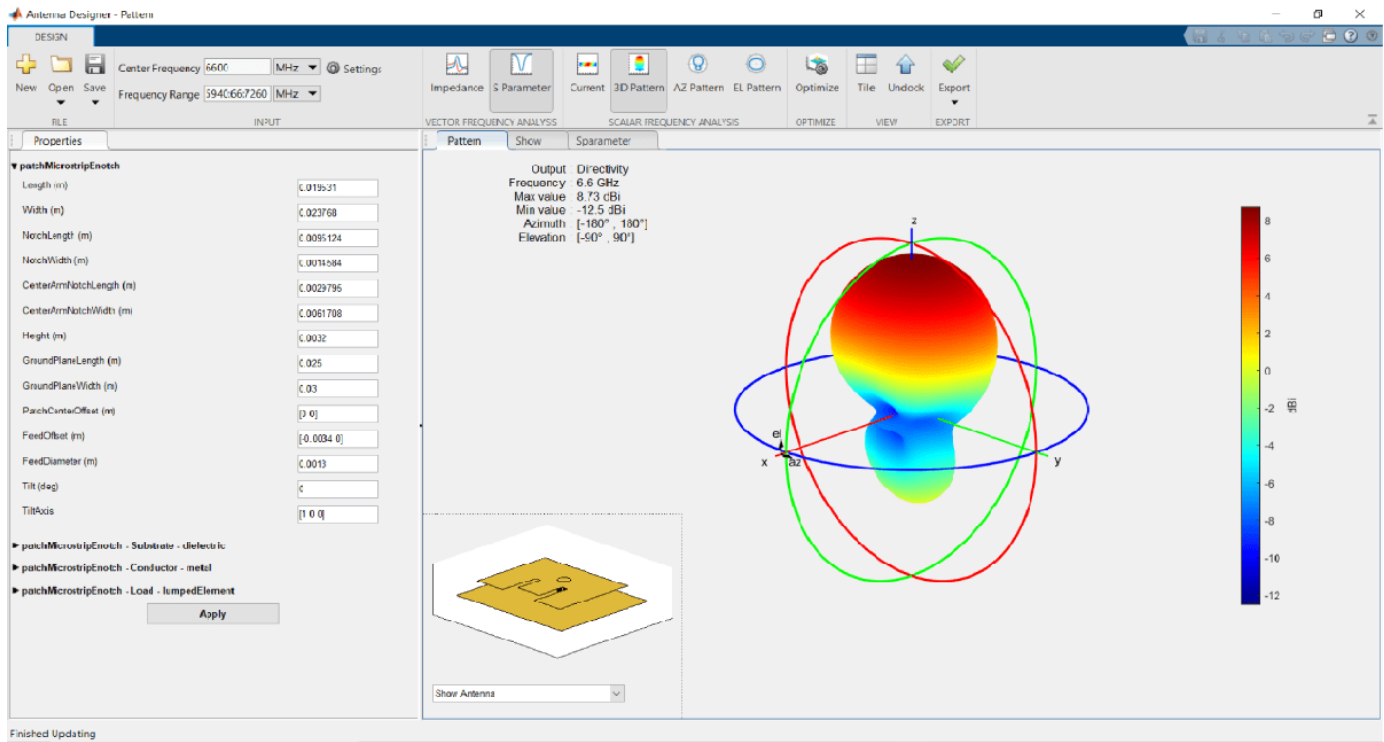
Optimized 3D Pattern and Bandwidth

Once the optimization is complete, click on **Accept**. This takes you back to the previous page.

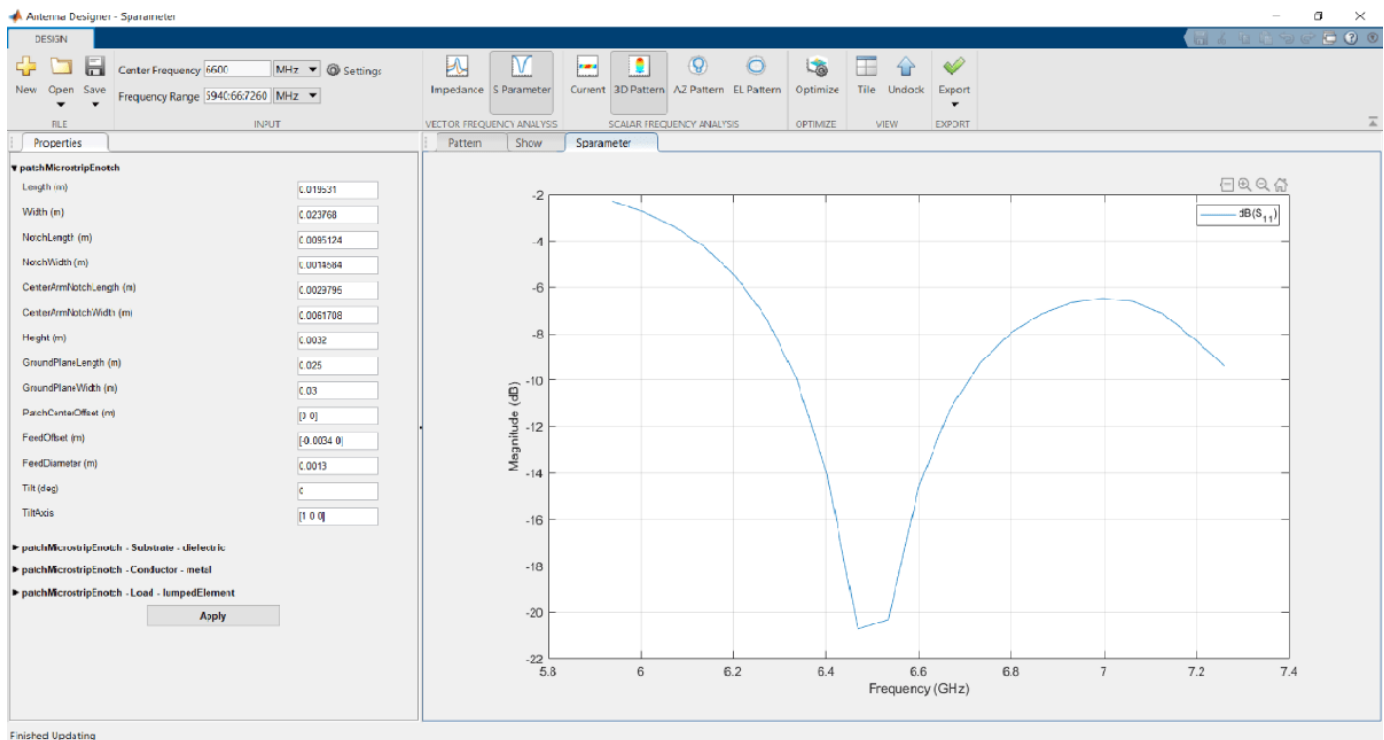


Click on **Pattern** to observe the 3D radiation pattern of the optimized antenna. Now, the maximum directivity is 8.73 dBi.

5 Antenna Toolbox Examples



Click on **S Parameter** to observe the impedance bandwidth of the antenna. The optimized antenna shows a bandwidth of around **6.25 GHz to 6.669 GHz**.



Before Optimization	After Optimization
Max. Directivity: 6.16 dBi	Max. Directivity: 8.73 dBi
Impedance Bandwidth: No Bandwidth	6.25 GHz to 6.669 GHz
Length: 0.0172 m Width: 0.02 m NotchLength: 0.01 m NotchWidth: 0.001 m CenterArmNotchLength: 0.0028 m CenterArmNotchWidth: 0.0062 m	Length: 0.01953 m Width: 0.02377 m NotchLength: 0.00951 m NotchWidth: 0.00146 m CenterArmNotchLength: 0.00298 m CenterArmNotchWidth: 0.00617 m

See Also

“Surrogate Based Optimization Design of Six-Element Yagi-Uda Antenna” on page 5-459

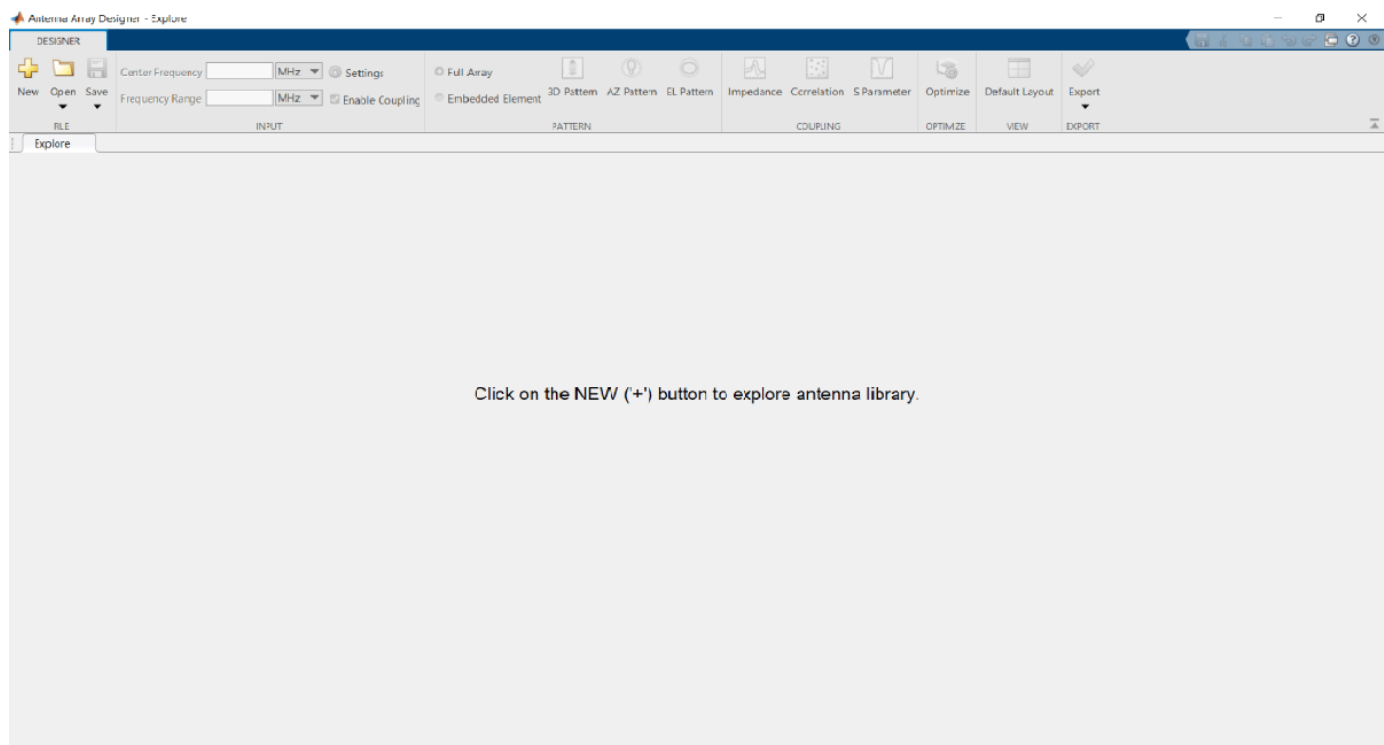
Optimization of Antenna Array Elements Using Antenna Array Designer App

This example demonstrates how to maximize the gain of 2-by-2 patch array antenna elements using a surrogate model assisted differential evolution for antenna synthesis (SADEA) optimizer. The design and analysis are performed at 2.4 GHz.

Open Antenna Array Designer App

Enter `antennaArrayDesigner` at the MATLAB® command prompt to open the app.

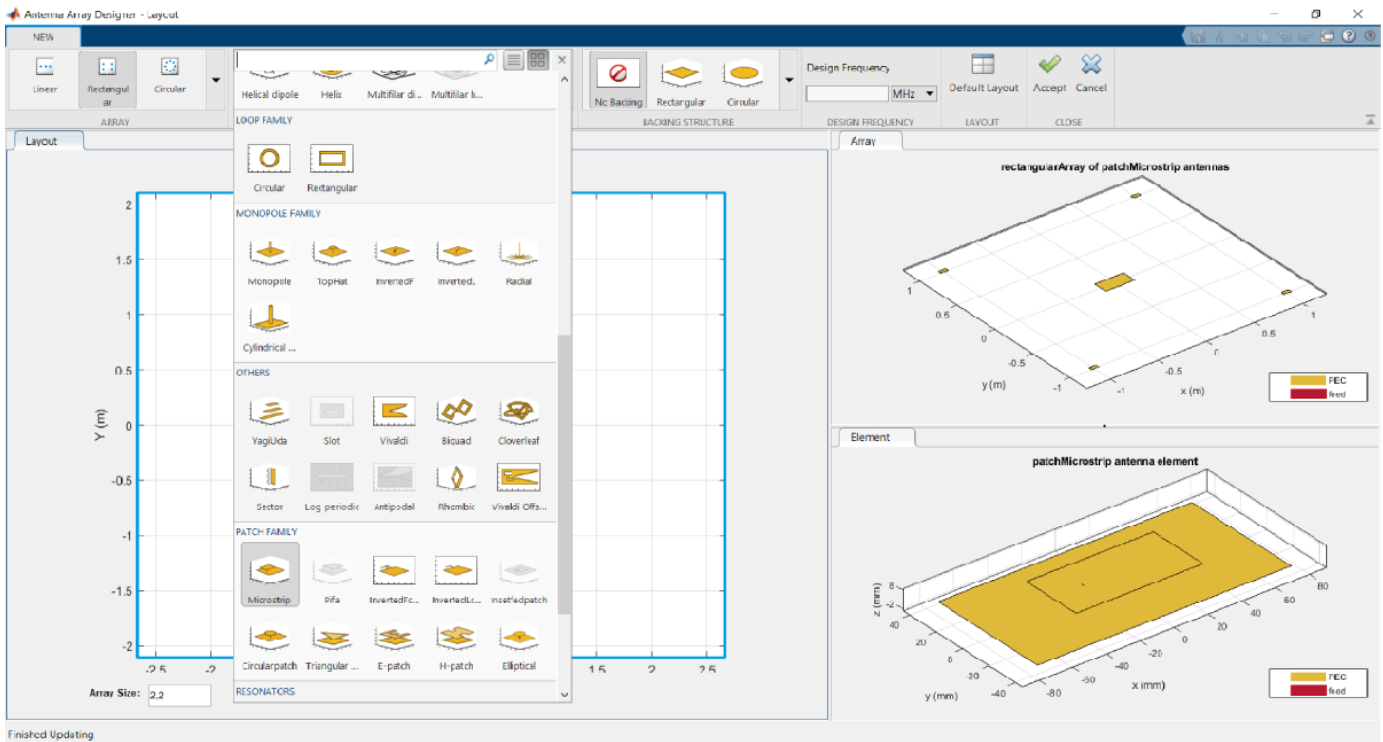
The command opens a blank canvas. In the blank canvas, click **New**.



Design Rectangular Microstrip Patch Array

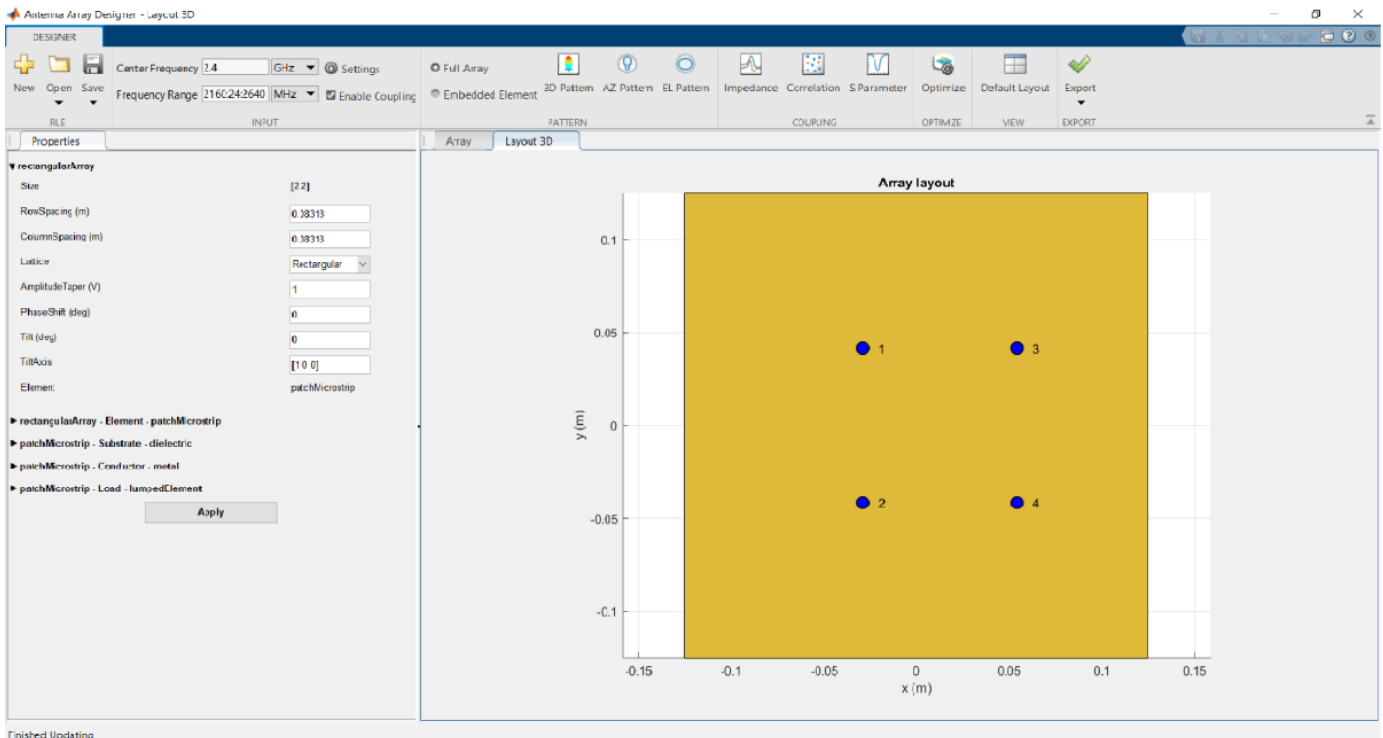
In the **New** tab, select **Rectangular** from **Array Gallery**. Select **Antenna Gallery > Microstrip** under **Patch family**.

Select **No Backing** under Backing Structure Gallery section.



Set the **Design Frequency** value to 2.4 GHz. Set **Array size** to 2, 2.

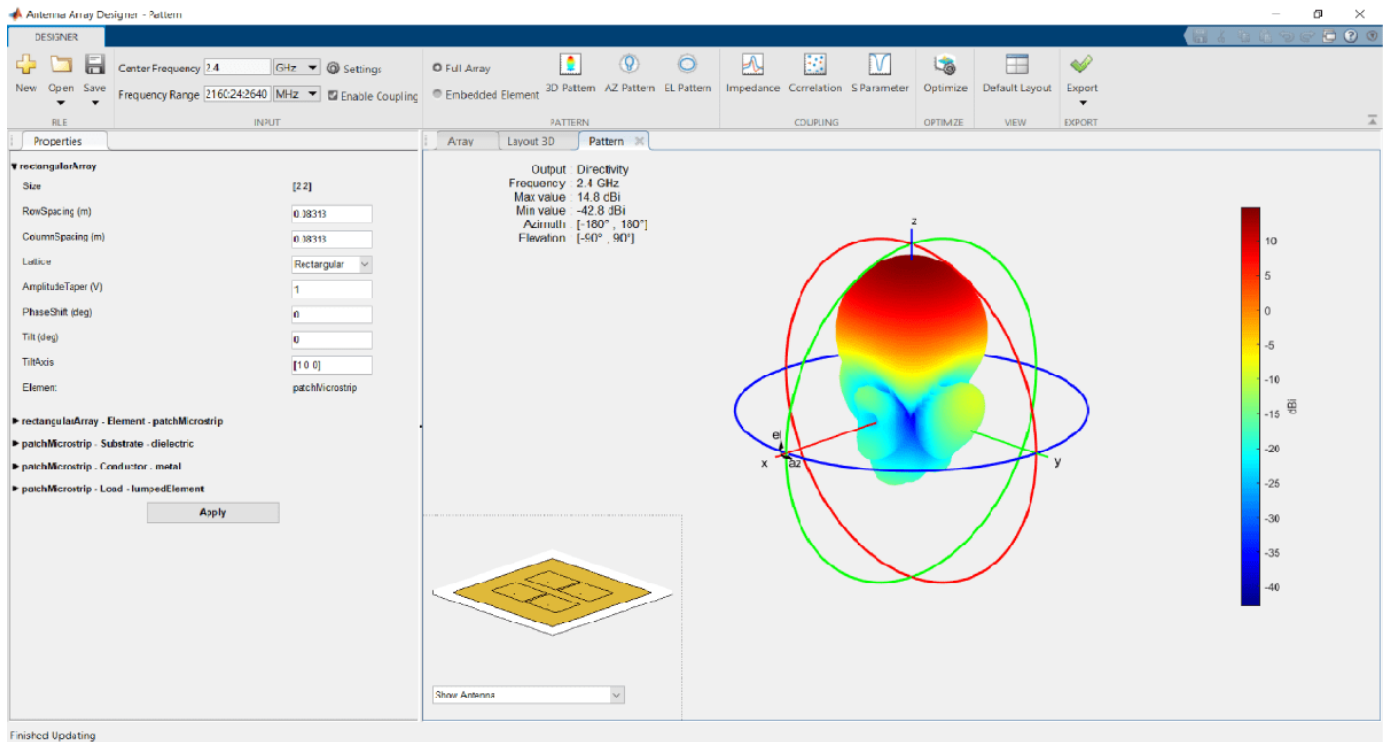
To analyze this antenna array, click **Accept**.



Plot 3-D Radiation Pattern

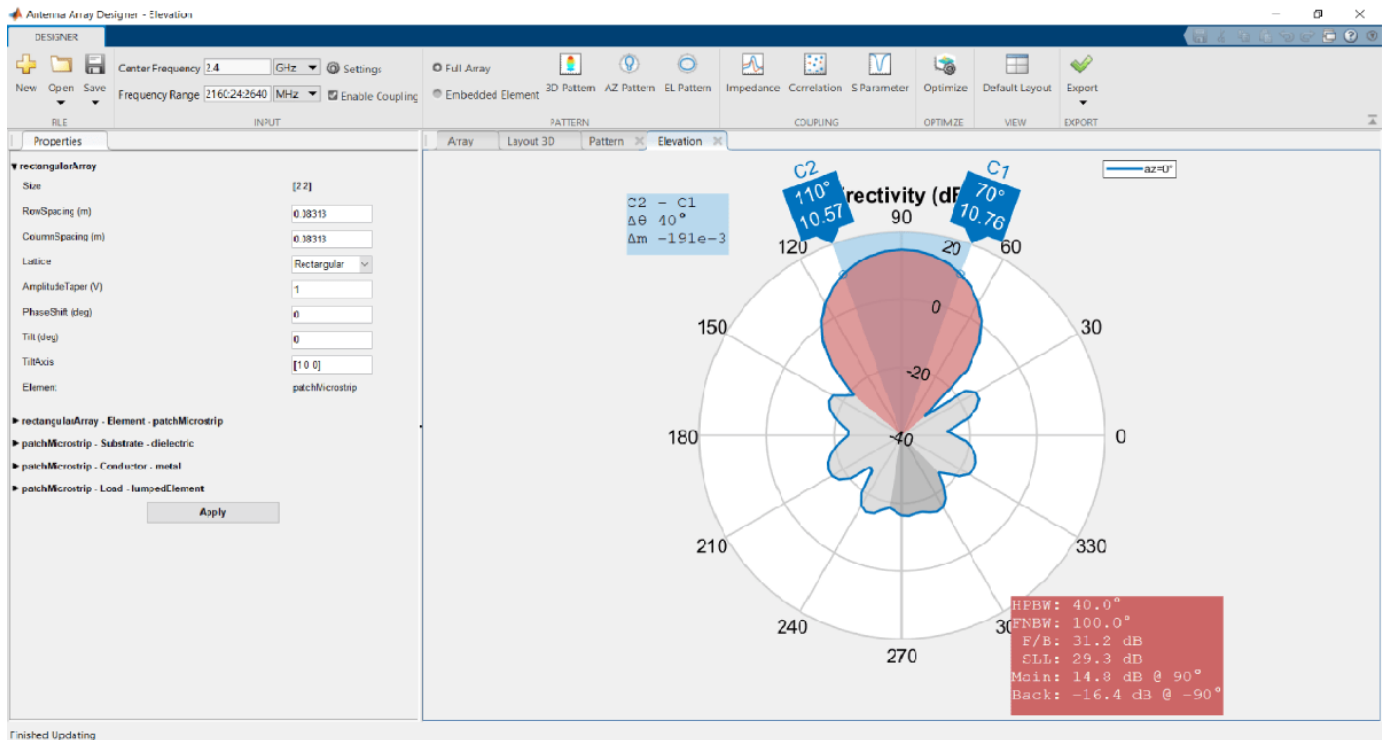
Observe the rectangular array of microstrip patch antenna and the layout of the geometry at 2.4 GHz in the **Array** and **Layout 3D** figure tabs.

In the toolbar, under the **PATTERN** section, click **3-D Pattern** to visualize the radiation pattern. The maximum gain of array is 14.8 dBi.



Plot 2-D Elevation Pattern to Measure Front to Back Lobe Ratio

In the toolbar, under the **PATTERN** section, click **EL Pattern** to visualize the front to back (F/B) lobe ratio. Right click on the plot and select **Measurements > Antenna Metrics**. A dialog box appears with the message: **Existing markers will be removed**. Select **Remove**.



The F/B(dB) is 31.2 dB.

Define Optimization Goal

To optimize an antenna array using the Antenna Array Designer App, these inputs are required:

- **Objective function:** The main goal of the optimization. The objective function evaluates the analysis function and minimizes or maximizes the output of the function.
- **Design variables:** The input variables that have to be optimized to achieve the objective function under certain constraints. These variables are changed by the optimizer within a preset range of values called the bounds of the variables.
- **Constraints:** The conditions under analysis that must be satisfied. The constraints are optional. If there are multiple constraints, then the user can prioritize the constraints using the **% Weight** parameter.
- **Other inputs:** These inputs might include number of iterations, center frequency, and frequency range at which the analysis is performed.

Optimization Goal: To maximize gain of the rectangular patch array antenna using the F/B lobe ratio as a constraint to maintain the desired direction of the main lobe.

In this example, inputs are:

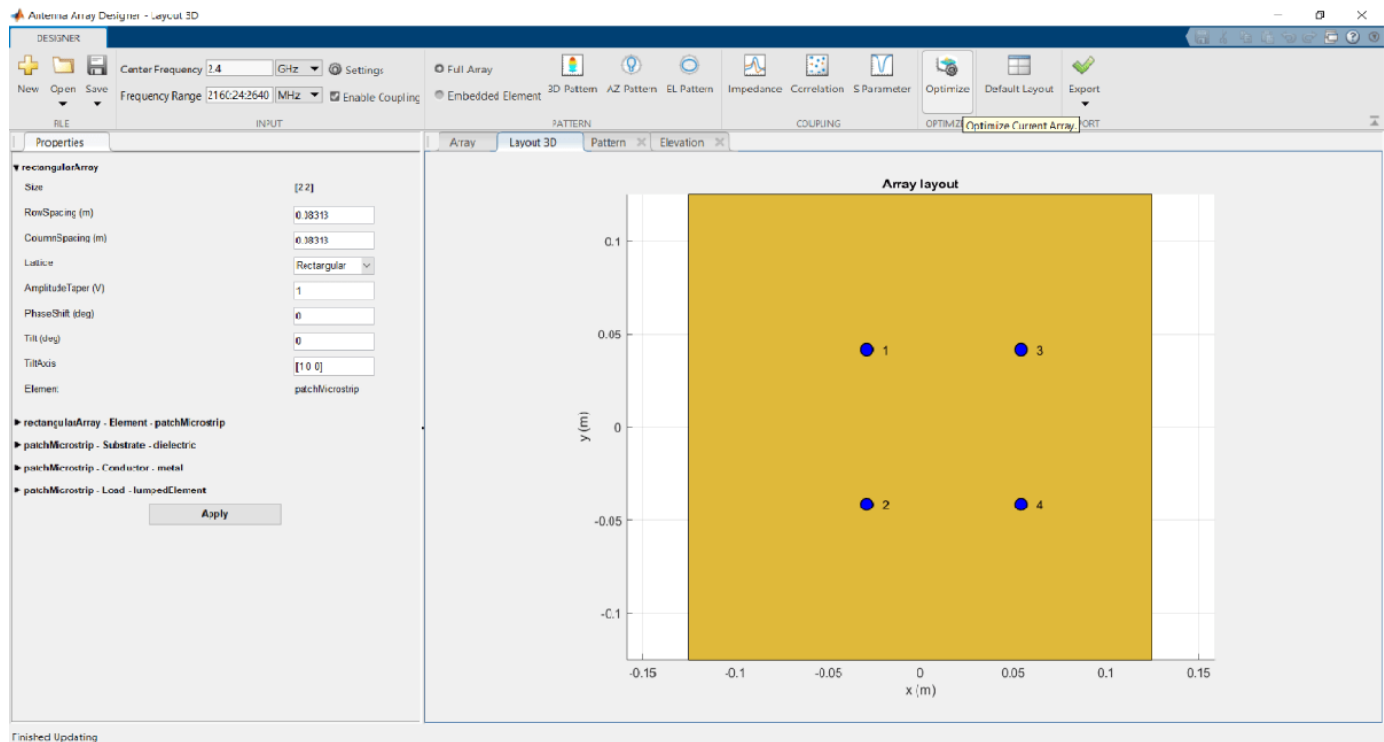
Objective function : Maximize Gain (dBi).

Design variables : RowSpacing, ColumnSpacing, GroundPlaneLength and GroundPlaneWidth.

Constraints : F/B lobe ratio (dB).

Set Optimizer

To optimize the rectangular patch array antenna, click **Optimize** under **Optimize section**.



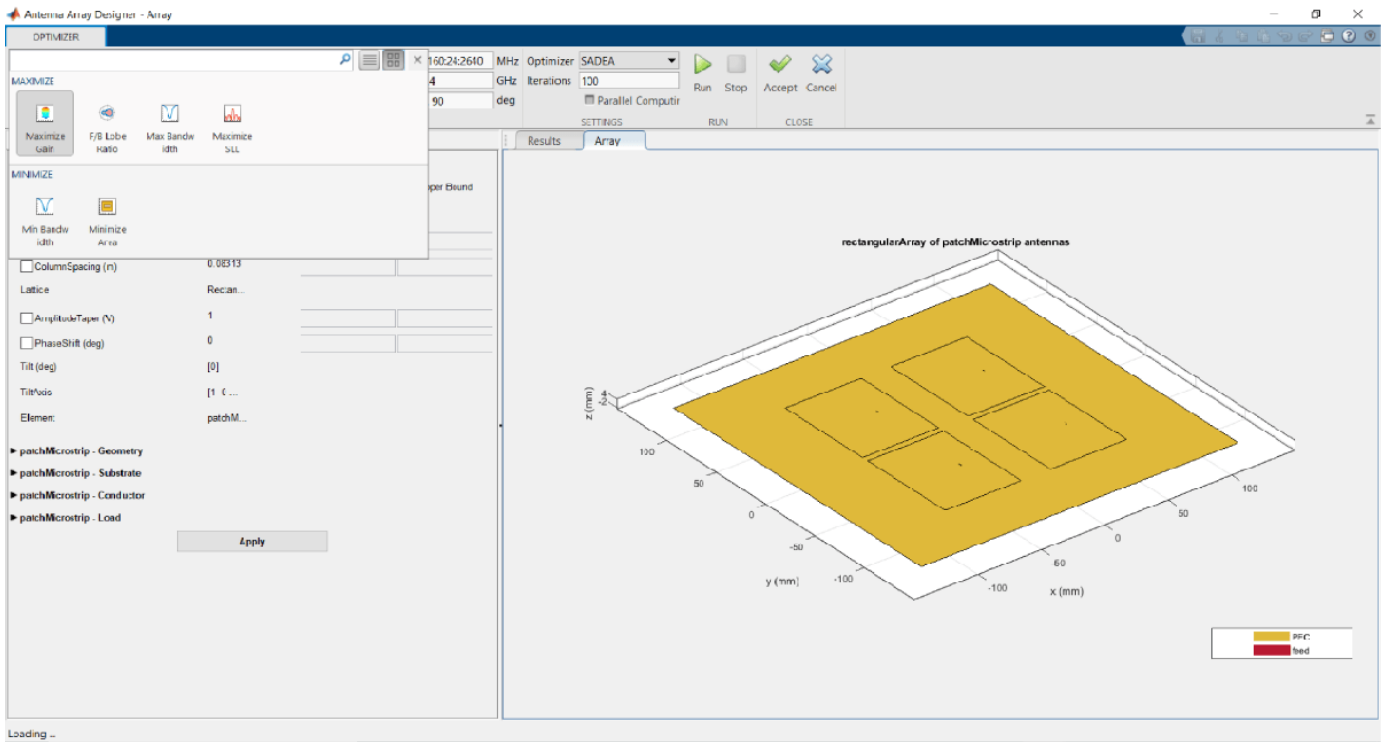
The optimizer supports several objective functions. This example uses maximizing gain as a objective function.

In this example, the optimizer can take up to seven hours to converge. To meet the objectives, these machine configurations are preferred:

- **Processor:** Intel® Xeon® CPU E5-1650 v4 @3.60GHz.
- **RAM:** 64GB.
- **System type:** 64-bit operating system.

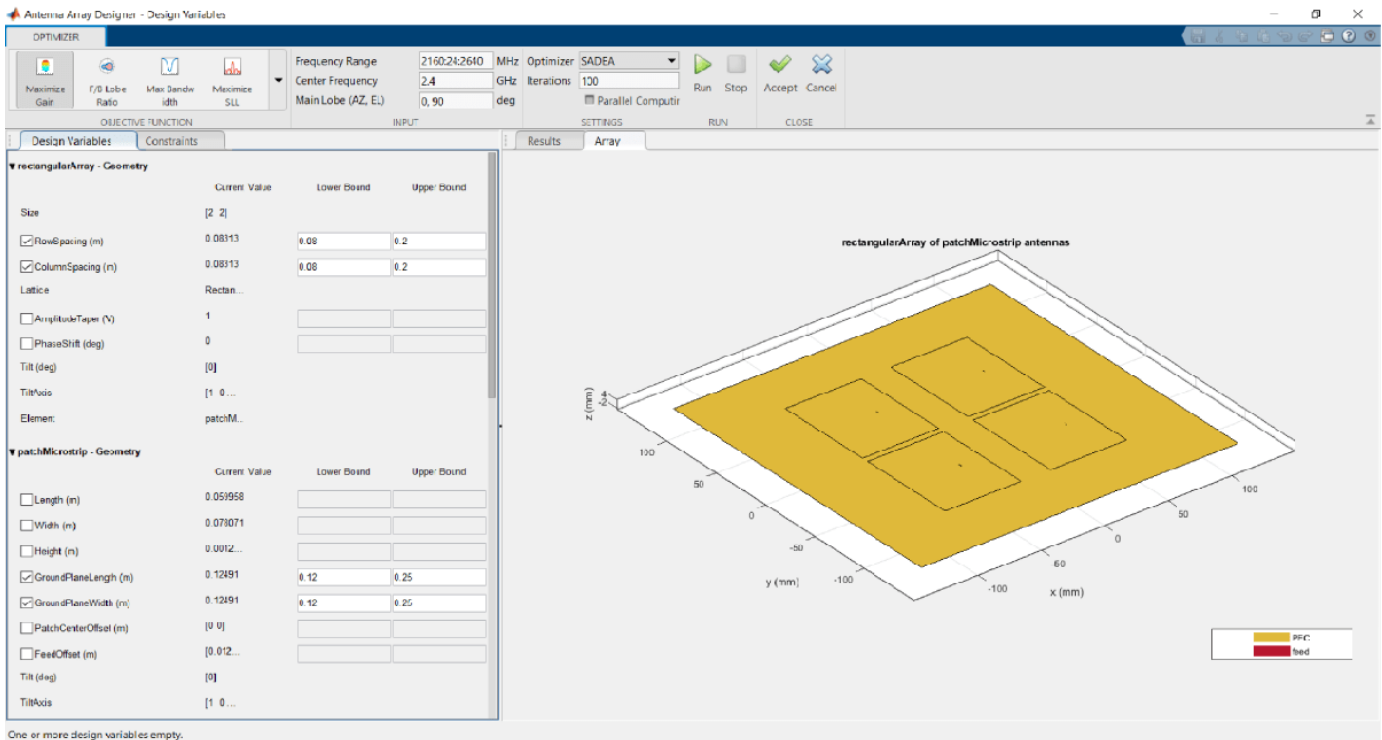
Set Objective Function

To select the objective function, select **Maximize Gain**.



Set Design Variables

To set design variables, select the **Design Variables** tab. Select the checkboxes to select the design variables. These variables are optimized to obtain maximum gain of the antenna.



In this example, select the check boxes corresponding to **RowSpacing** and **ColumnSpacing** under **rectangularArray-Geometry** and **GroundPlaneLength** and **GroundPlaneWidth** under **patchMicrostrip -Geometry**.

Design variables are set based on the values shown in the table.

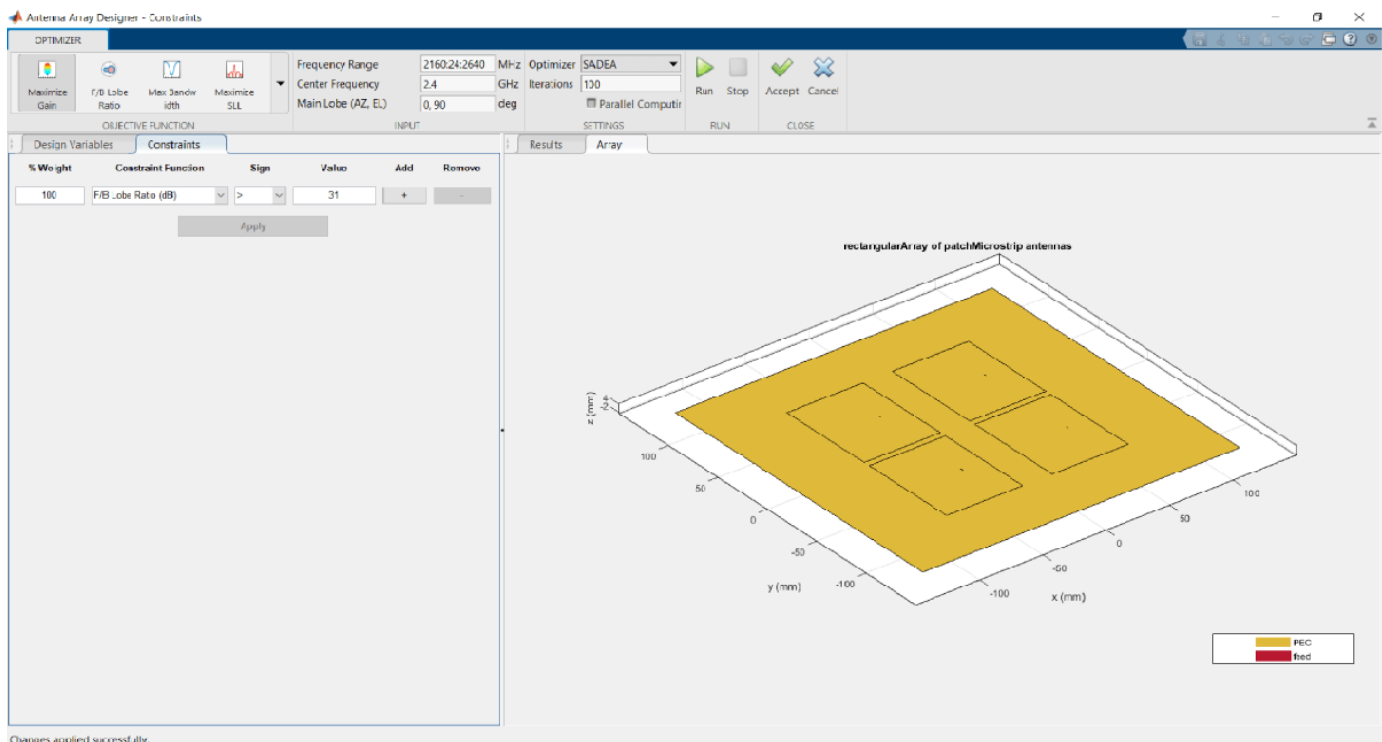
Design Variables	Lower Bound	Upper Bound
RowSpacing	0.08	0.2
ColumnSpacing	0.08	0.2
GroundPlaneLength	0.12	0.25
GroundPlaneWidth	0.12	0.25

Click **Apply** to set the variables.

Set Constraints to Optimization Goal

Choose the **Constraint Function** under **Constraint** tab.

To set constraints, choose **F/B Lobe Ratio (dB)** as the **Constraint Function** from **Constraint** pane select **'>'** operator under **sign** and set **Value** as 31.



Click **Apply** to accept the Constraints.

To input the number of iterations, in **Settings** section, set **Iterations** to 200 and select **Parallel Computing** if you have Parallel Computing Toolbox™.

Run SADEA Optimization

To start optimization, click **Run**.

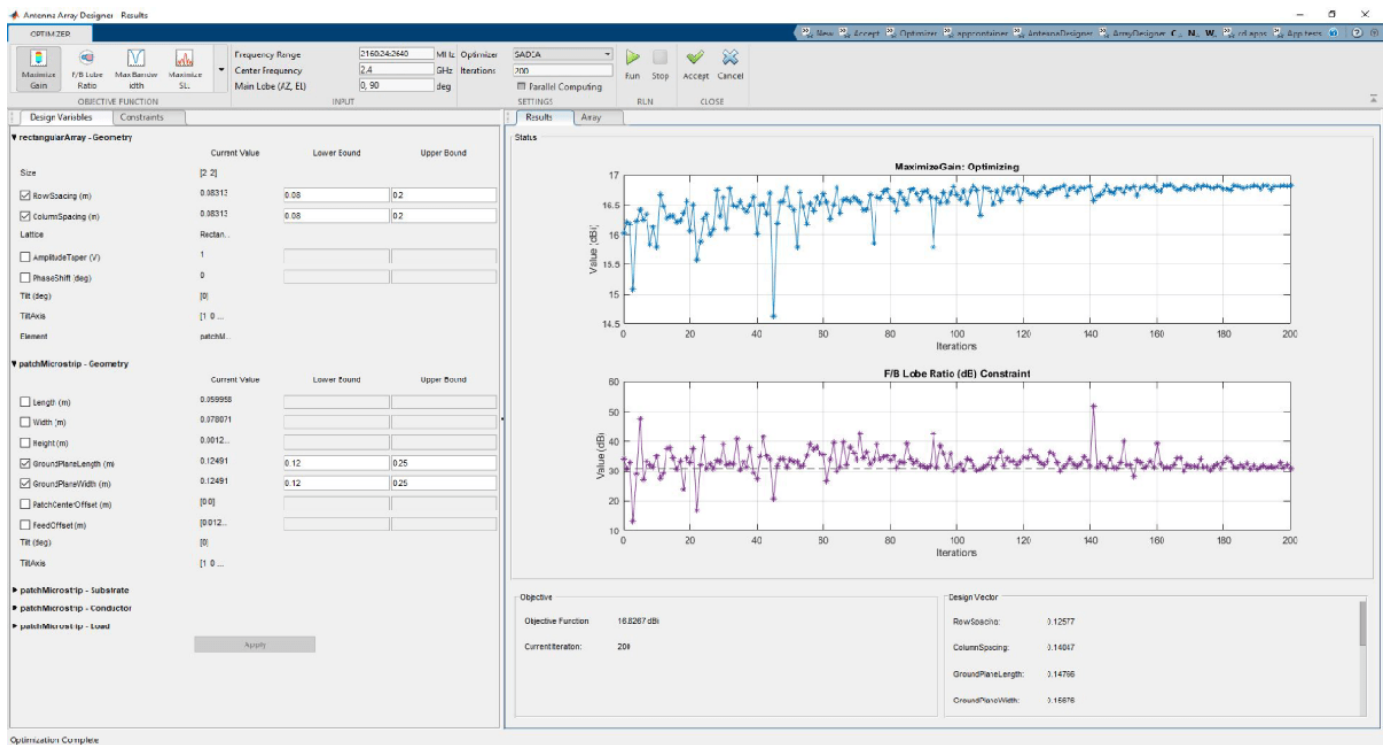
The SADEA algorithm contains two stages

- 1 Building model
- 2 Optimization

In the building model stage, the optimizer makes a surrogate model from the design space, specified objective, and the constraints function. In the design space analysis are performed on sample points.

As a result, the X-axis shows the number of samples, and the Y-axis shows the value of the analysis function value at that sample. The bottom left side of the app window shows the current sample value, and the bottom right side of the app window shows the design variables. The optimizer takes an appropriate number of samples to build the model. After the model is built, the optimizer starts running iterations.

In the optimization stage, the X-axis shows the number of iterations, and the Y-axis shows the objective function values. From the plots shown on the optimizing stage, you can understand the trend of convergence.

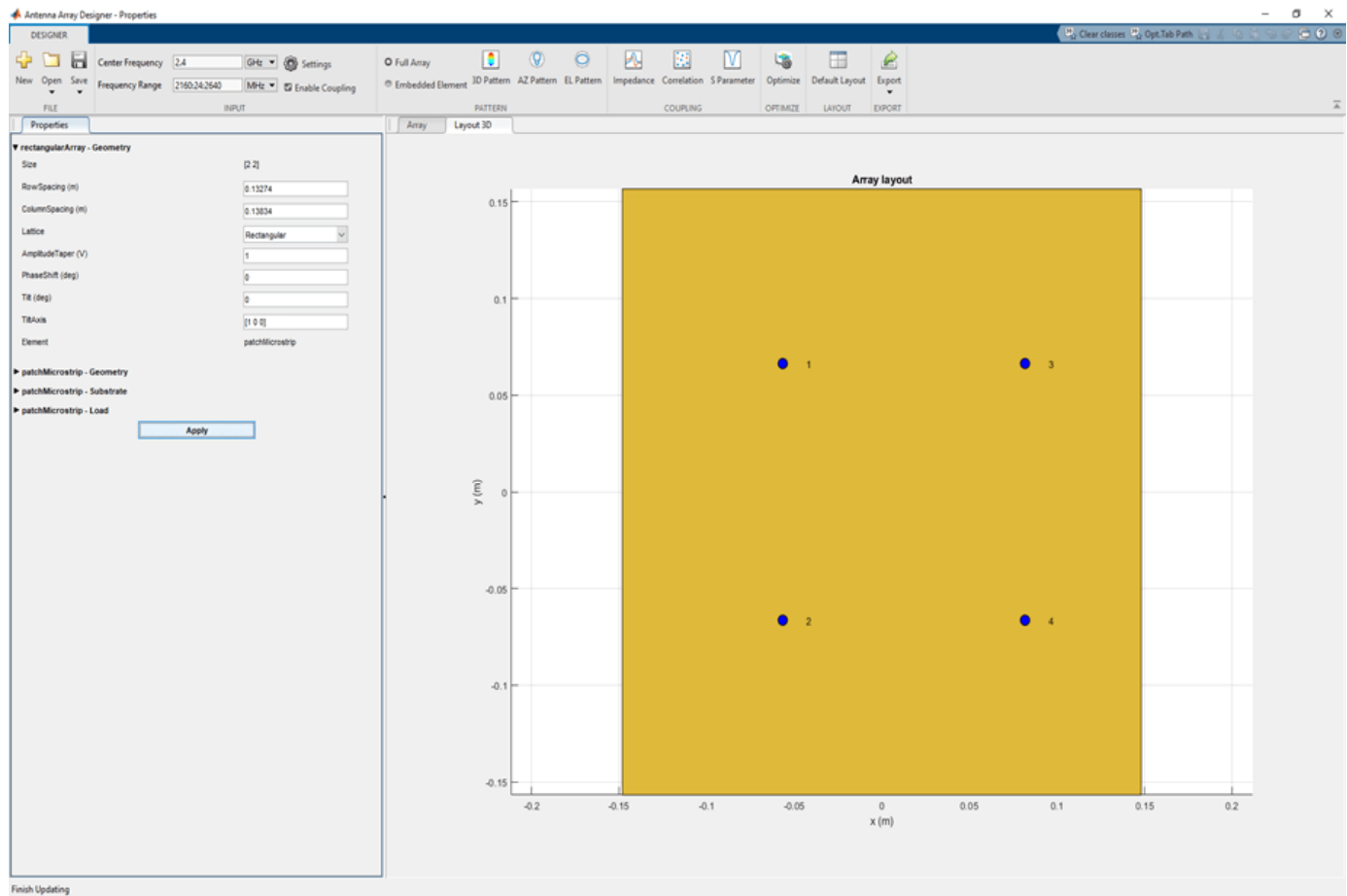


Note: The objective and the constraint plots show fewer deviations. After the 170th iteration, the two plots converge.

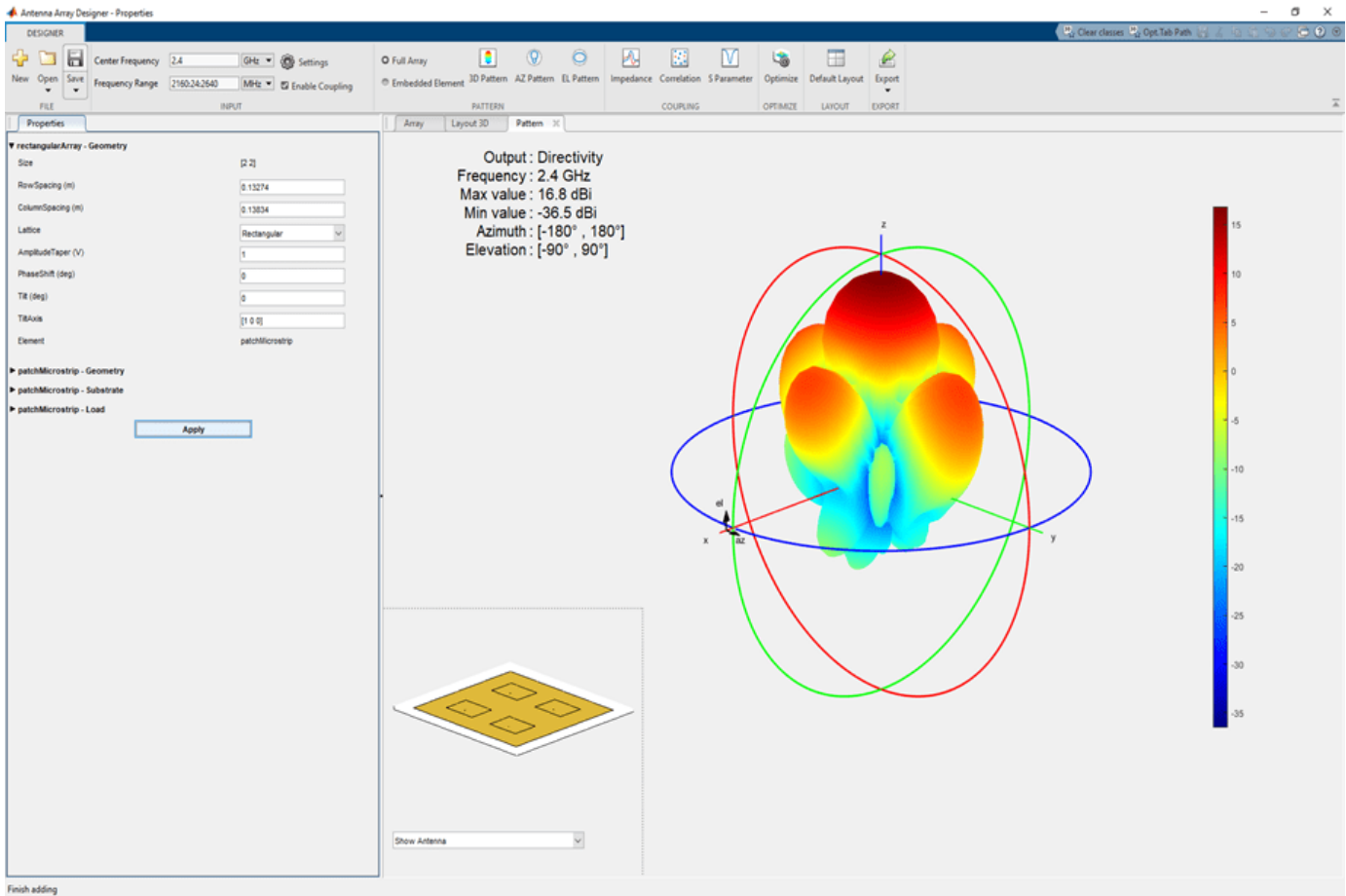
Optimized 3-D Pattern and Front to Back Lobe Ratio

Once optimization is complete, click **Accept**. Click **Apply** to analyze the 3-D and 2-D patterns.

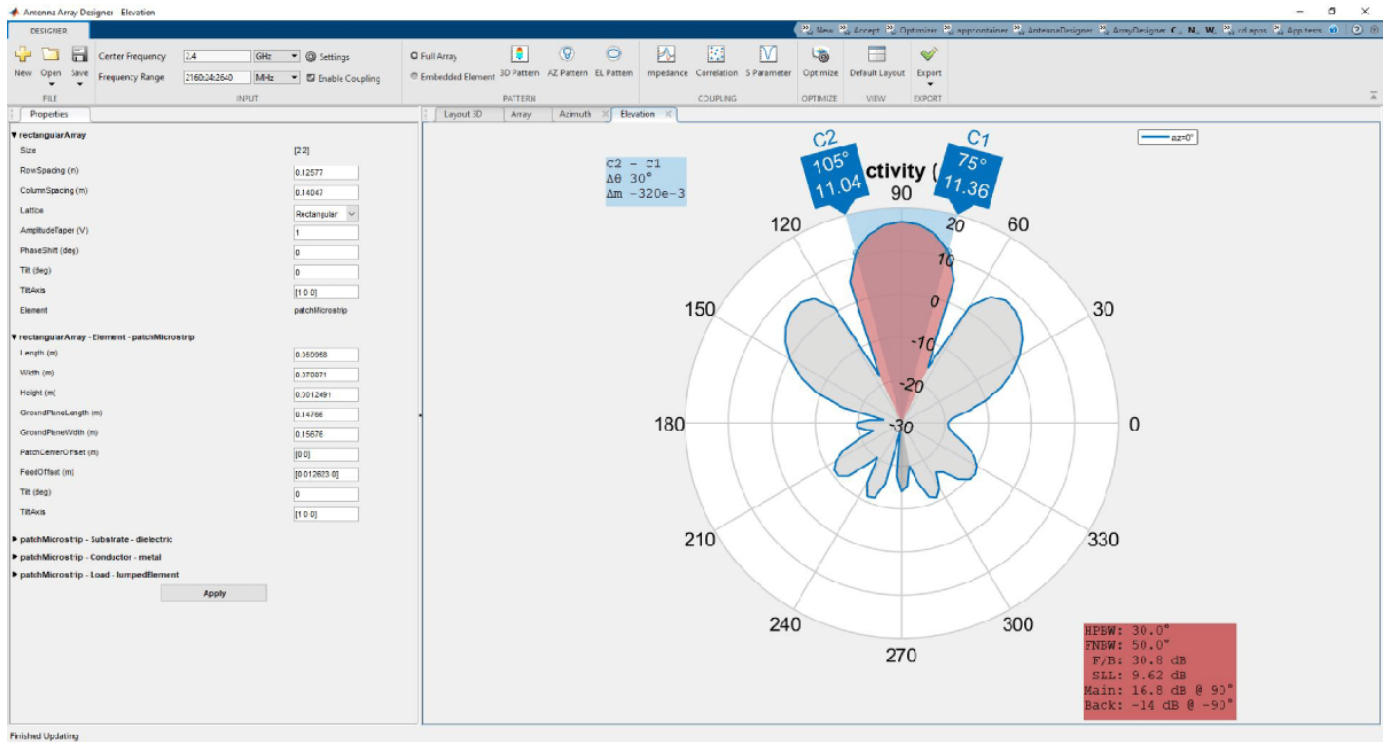
5 Antenna Toolbox Examples



To observe the 3-D radiation pattern of the optimized antenna, click **3D Pattern** in the Pattern section. The maximum directivity is observed as 16.8 dBi.



Click **EL Pattern** in Pattern section to observe the F/B lobe ratio. The optimized antenna F/B lobe ratio is 31.4 dB.



The table shows a comparison of the results.

Before Optimization		After Optimization	
Max directivity	14.8 dBi	Max directivity	16.8 dBi
F/B lobe ratio	31.2 dBi	F/B lobe ratio	31.4 dBi
RowSpacing (m)	0.08313	RowSpacing (m)	0.13274
ColumnSpacing (m)	0.08313	ColumnSpacing (m)	0.13834
GroundPlaneLength (m)	0.12491	GroundPlaneLength (m)	0.14813
GroundPlaneWidth (m)	0.12491	GroundPlaneWidth (m)	0.15668

See Also

“Maximizing Gain and Improving Impedance Bandwidth of E-Patch Antenna” on page 5-625

Import and Analyze Custom 3-D Antenna Geometry

This example shows how to simulate a 3-D custom antenna geometry from an STL (stereolithography) file. To simulate a 3-D custom antenna geometry, use the `customAntennaStl` object. This object allows the user to simulate a custom 3-D geometry from an *.stl file. An STL file is a tessellation of a structure in 3-D space using triangles.

Create customAntennaStl Object

Create a `customAntennaStl` object with `FileName` 'antenna3D.stl'.

```
c = customAntennaStl('FileName','antenna3D.stl')
```

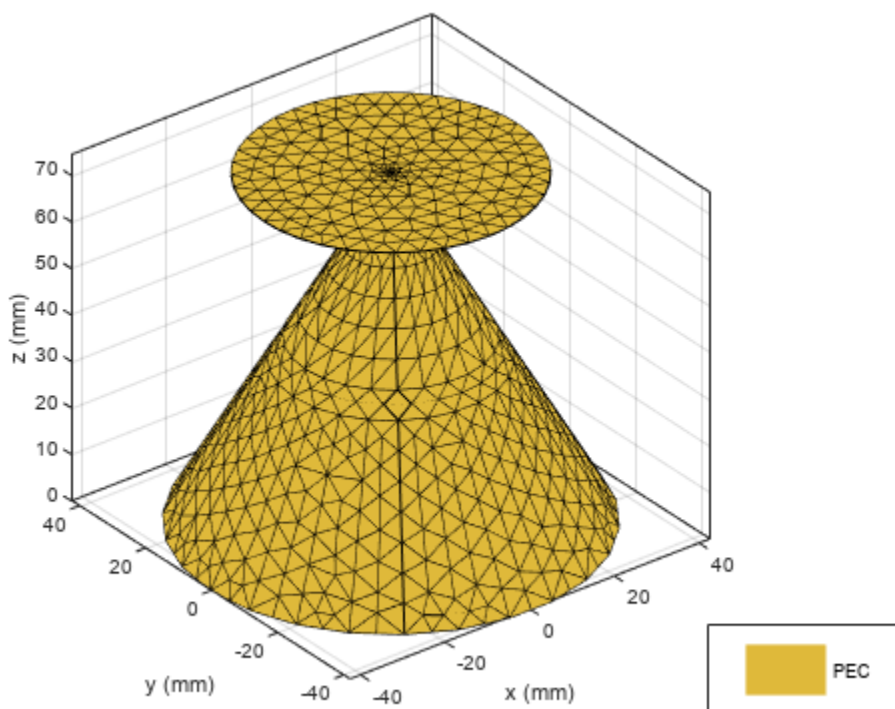
```
c =  
  customAntennaStl with properties:
```

```
      FileName: 'antenna3D.stl'  
      Units: 'm'  
      FeedLocation: []  
      AmplitudeTaper: 1  
      PhaseShift: 0  
      UseFileAsMesh: 0  
      Tilt: 0  
      TiltAxis: [1 0 0]
```

View STL Geometry

View the 3-D antenna geometry in the STL file.

```
figure  
show(c)
```



Create Feed

The STL file holds information about the 3-D geometry. However, It does not contain information about feeding edges or the feed location, Therefore, a feed is required to excite the antenna

An antenna feed in `customAntennaStl` object can be created in two ways:

- Create a feed using command line interface.
- Create a feed using UI figure window which allows you to select the edges.

Create Feed Using Command Line Interface

To create a feed on the antenna, select an edge located at the origin at a height of 74.36 mm.

The edges are selected based on the distance between the feed location and the midpoints of the edges. The edges surrounding the feed are selected using `NumEdges` property.

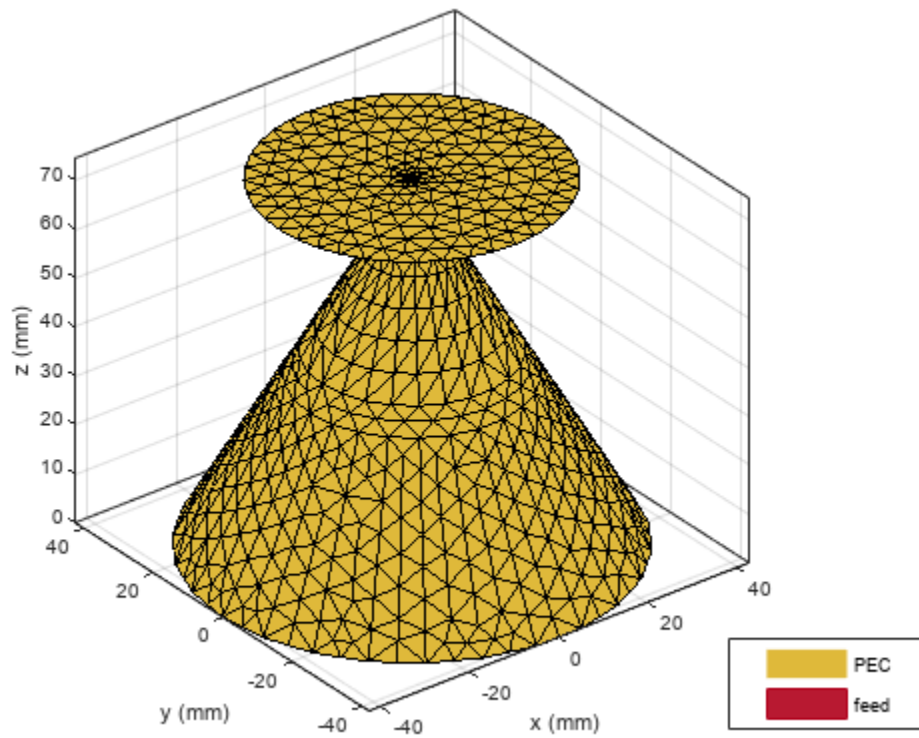
In this example, the `NumEdges` property is set to 1. The closest single edge is selected as the feed.

```
c.createFeed('FeedLocation',[0,0,0.07436],'NumEdges',1);
```

View STL Geometry with Feed

View the 3-D antenna geometry with feed. The feed is created at the specified `FeedLocation`.

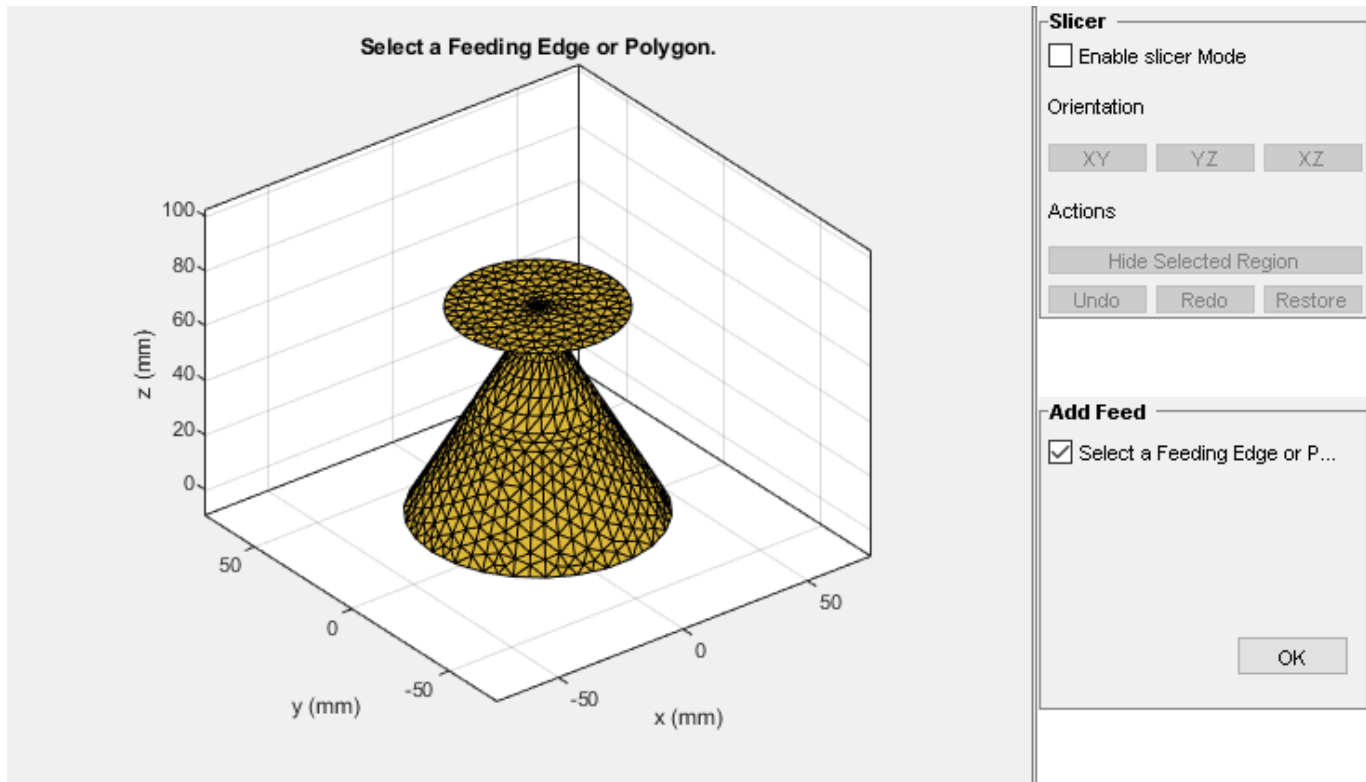
```
figure
show(c)
```



Create Feed Using UI Figure Window

For complex 3-D antenna geometries, the feed could be set using UI figure window. To select the feeding edges, open the UI figure window using `createFeed()` function.

`c.createFeed()`

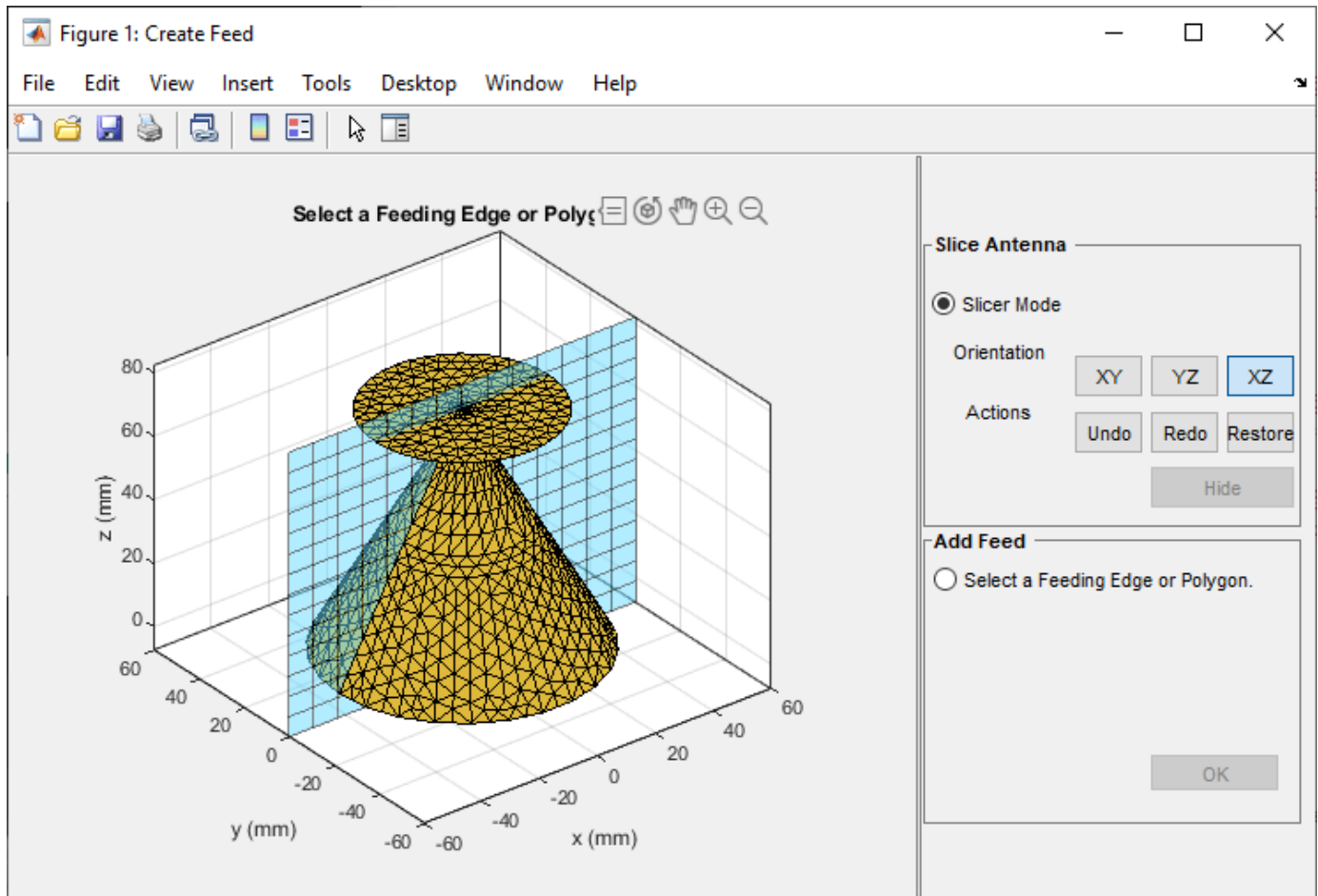


Select Slicer Mode

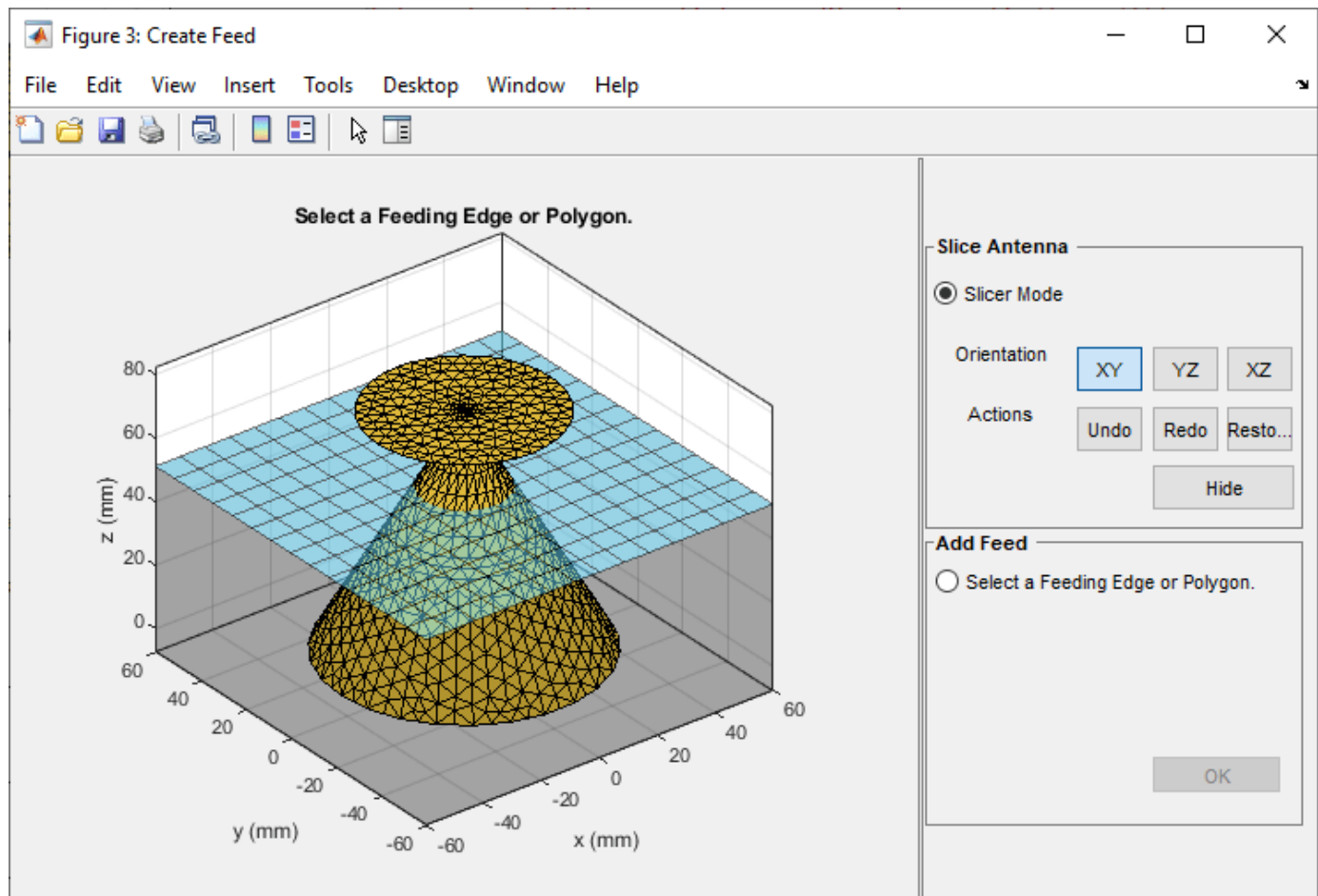
To select the feeding edge, the region of interest needs to be exposed in your 3-D structure where the feeding edge is located.

The UI figure window consists of two panes, the **Slice Antenna** pane and the **Add Feed** pane. The **Slicer Mode** when selected, creates a slicing plane that can be used to slice the 3-D structure. The orientation of the slicing plane can be shifted using the **XY**, **YZ**, and **XZ** buttons.

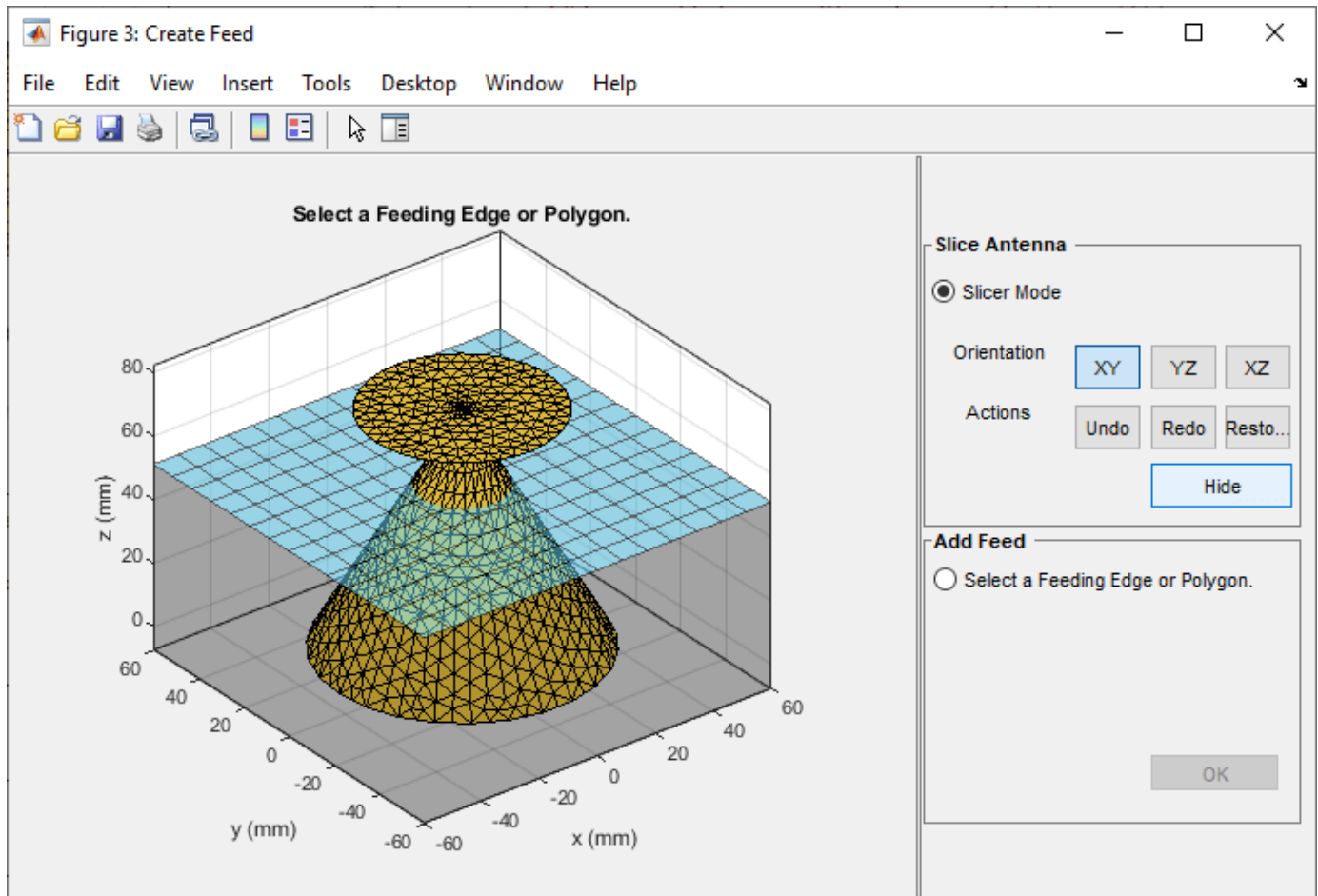
Select the **Slicer Mode**, then click **XZ** to select that as the plane along which to slice your antenna.

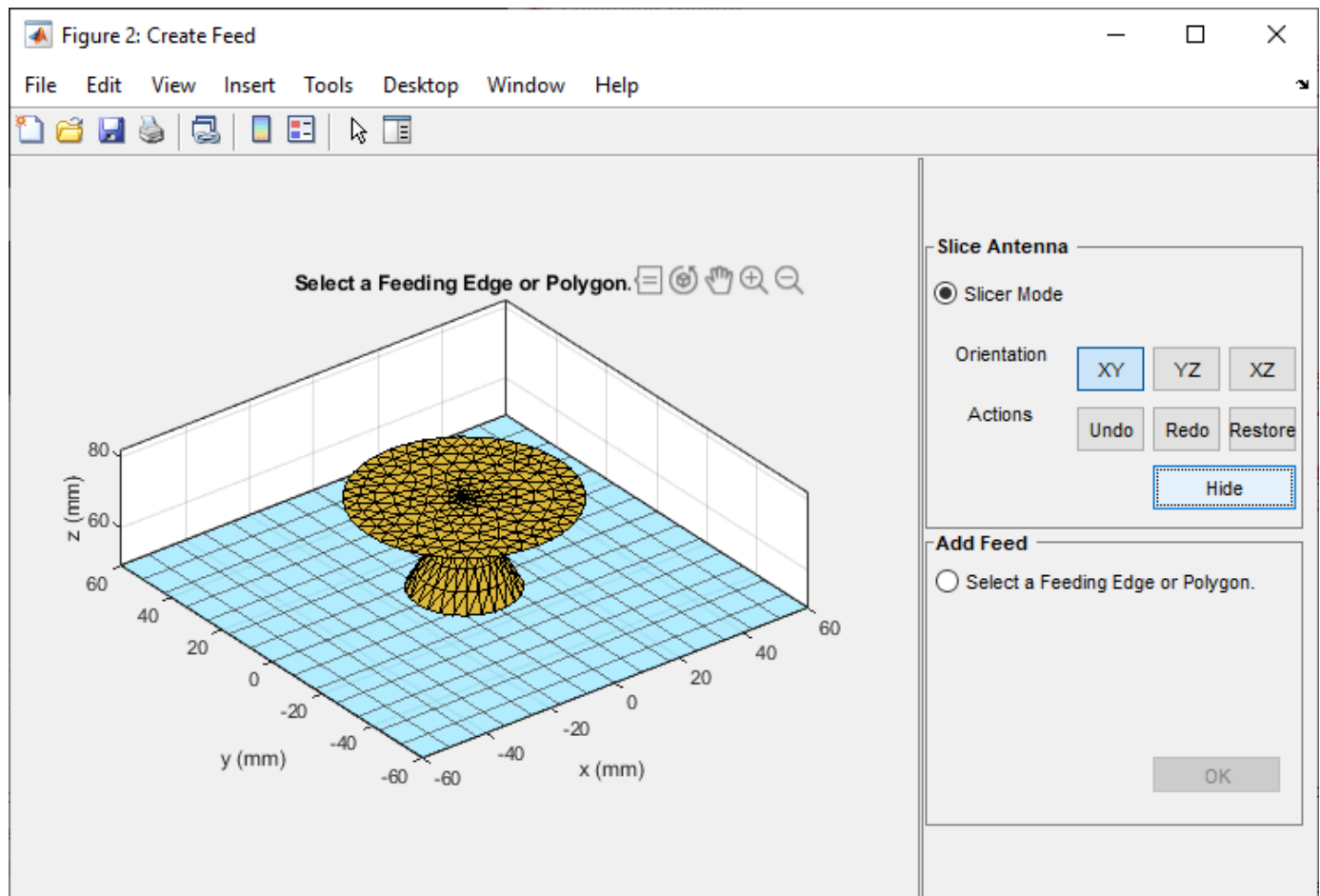


Select the region you want to hide.

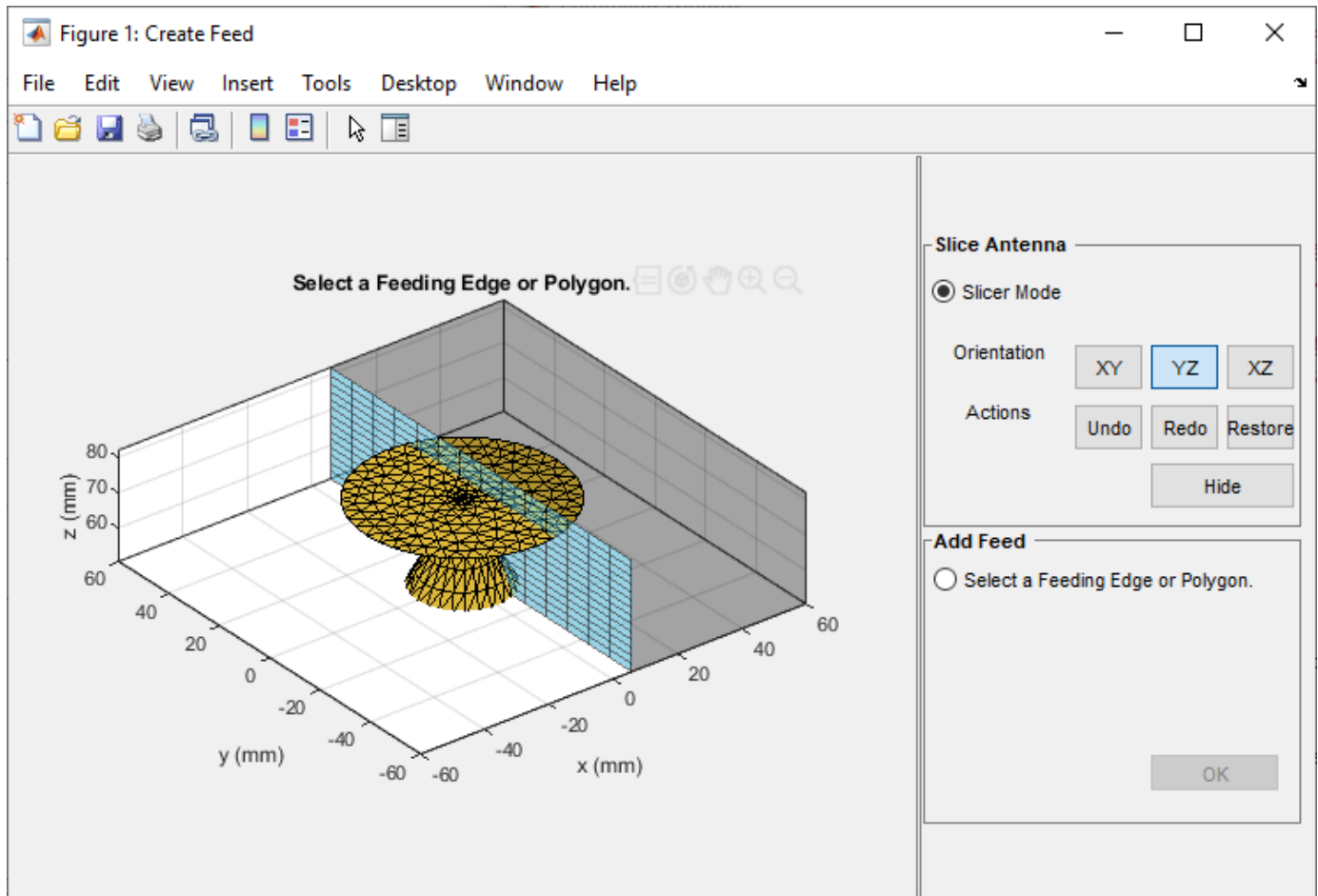


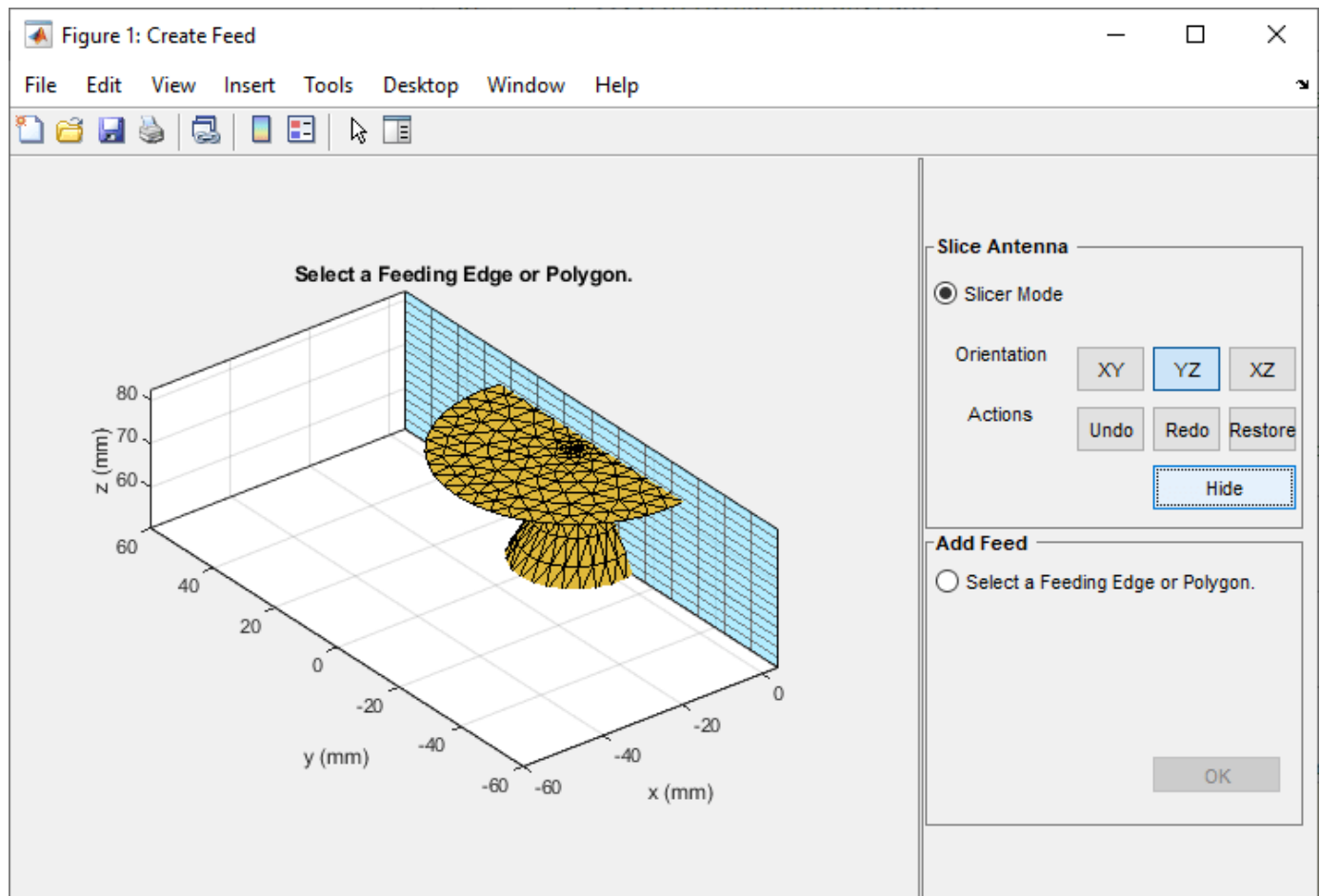
Click **Hide** to hide the selected region. The selected region is grayed out.

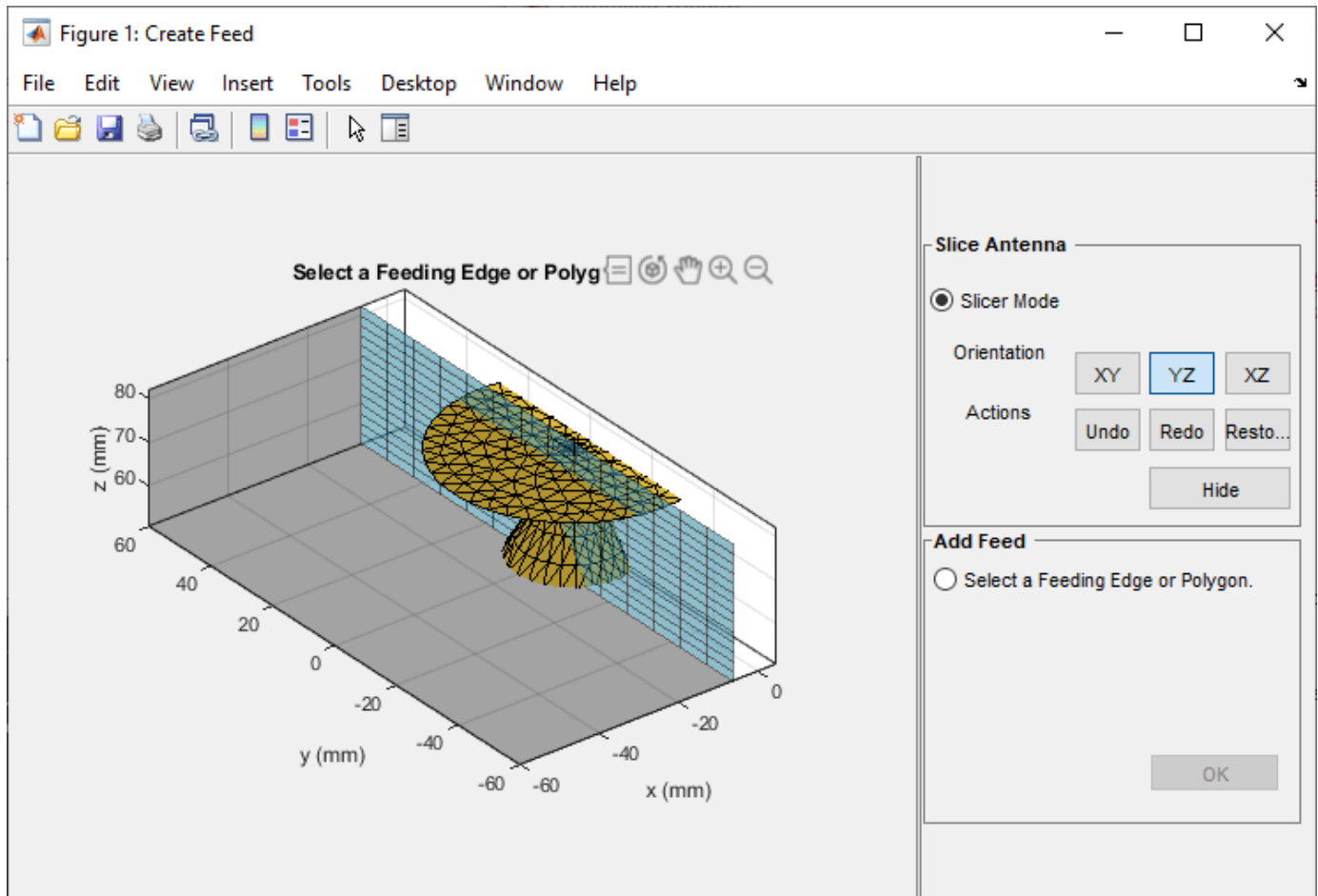


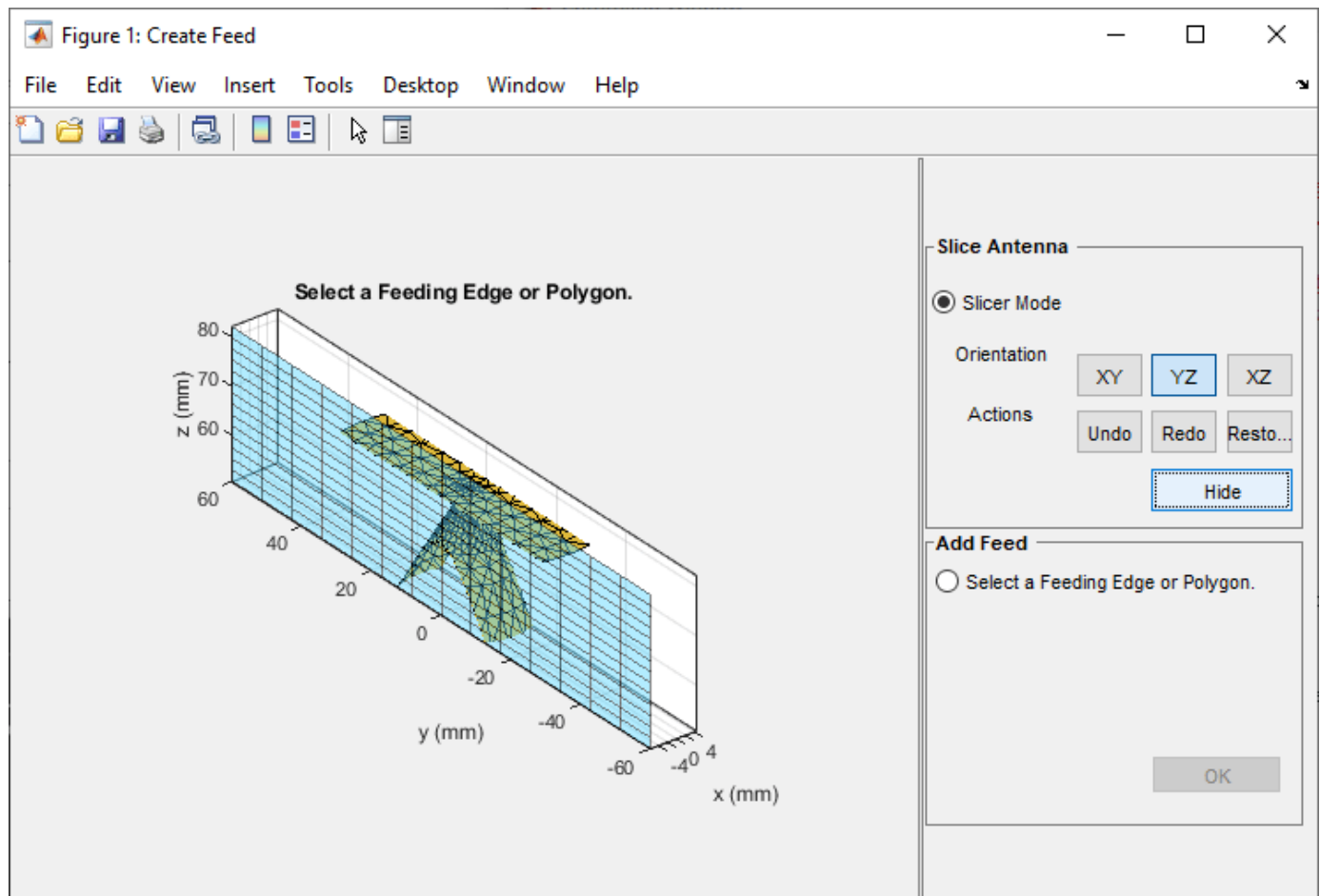


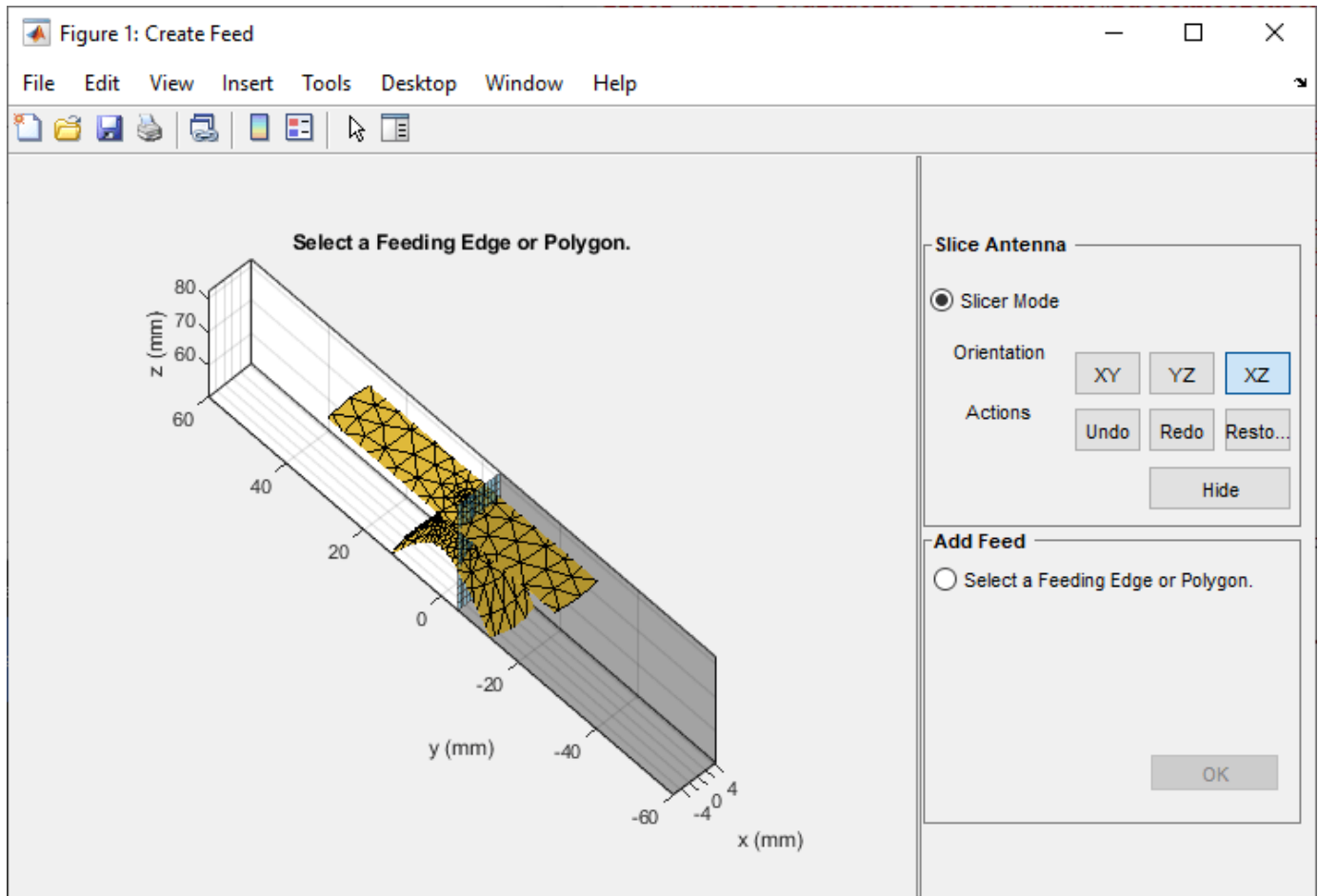
Repeat the process until you reach the region of interest.

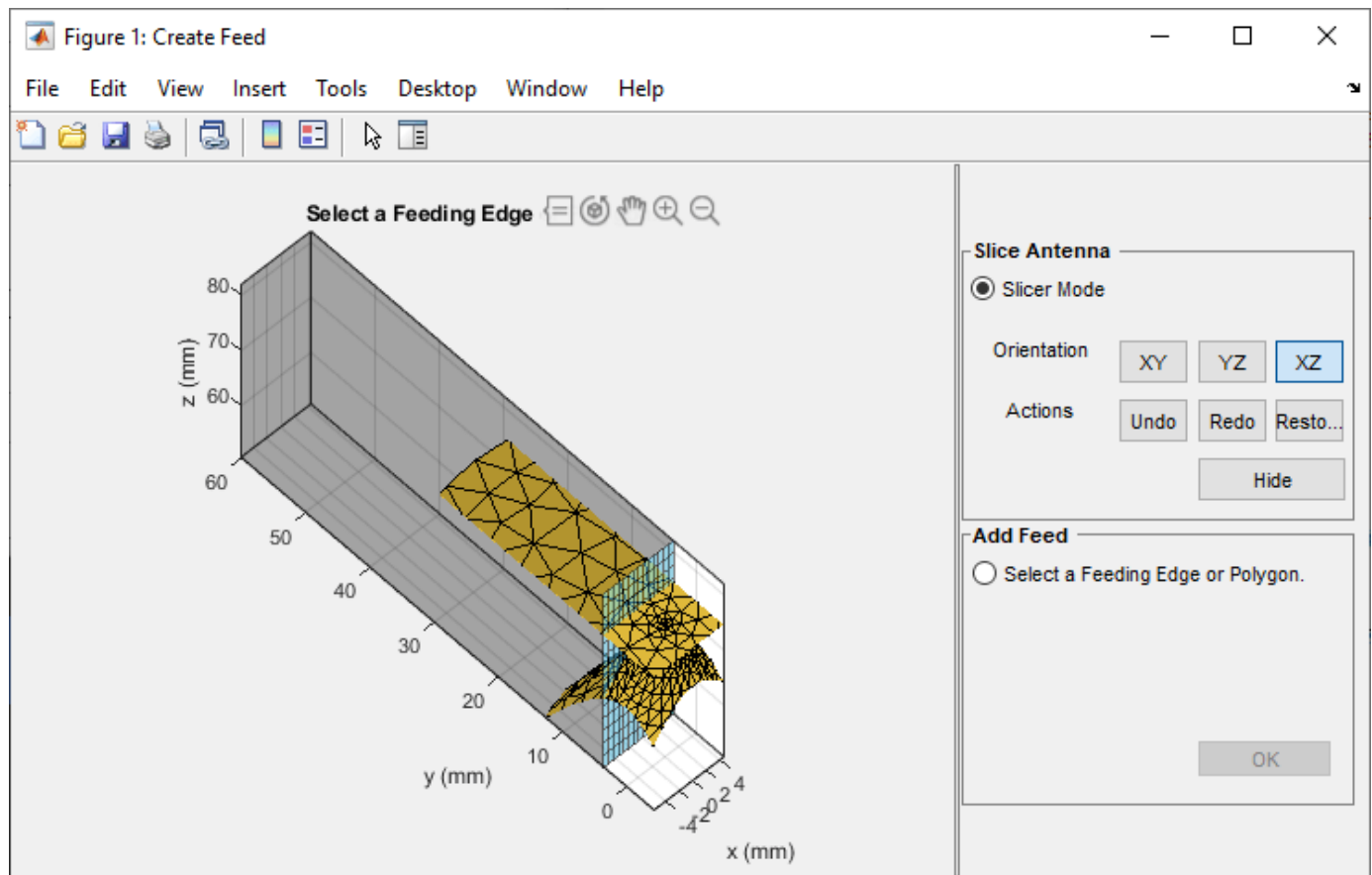


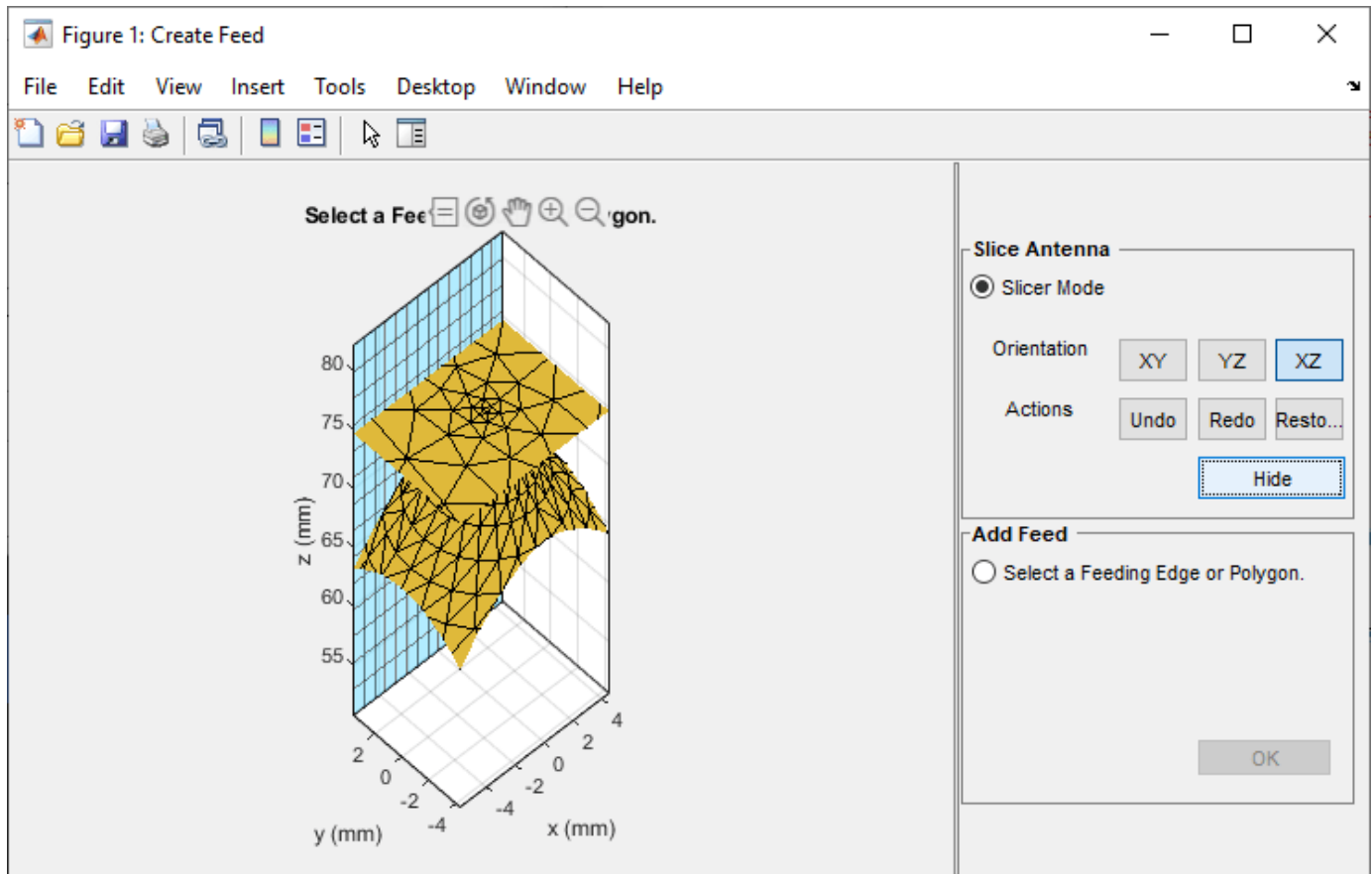


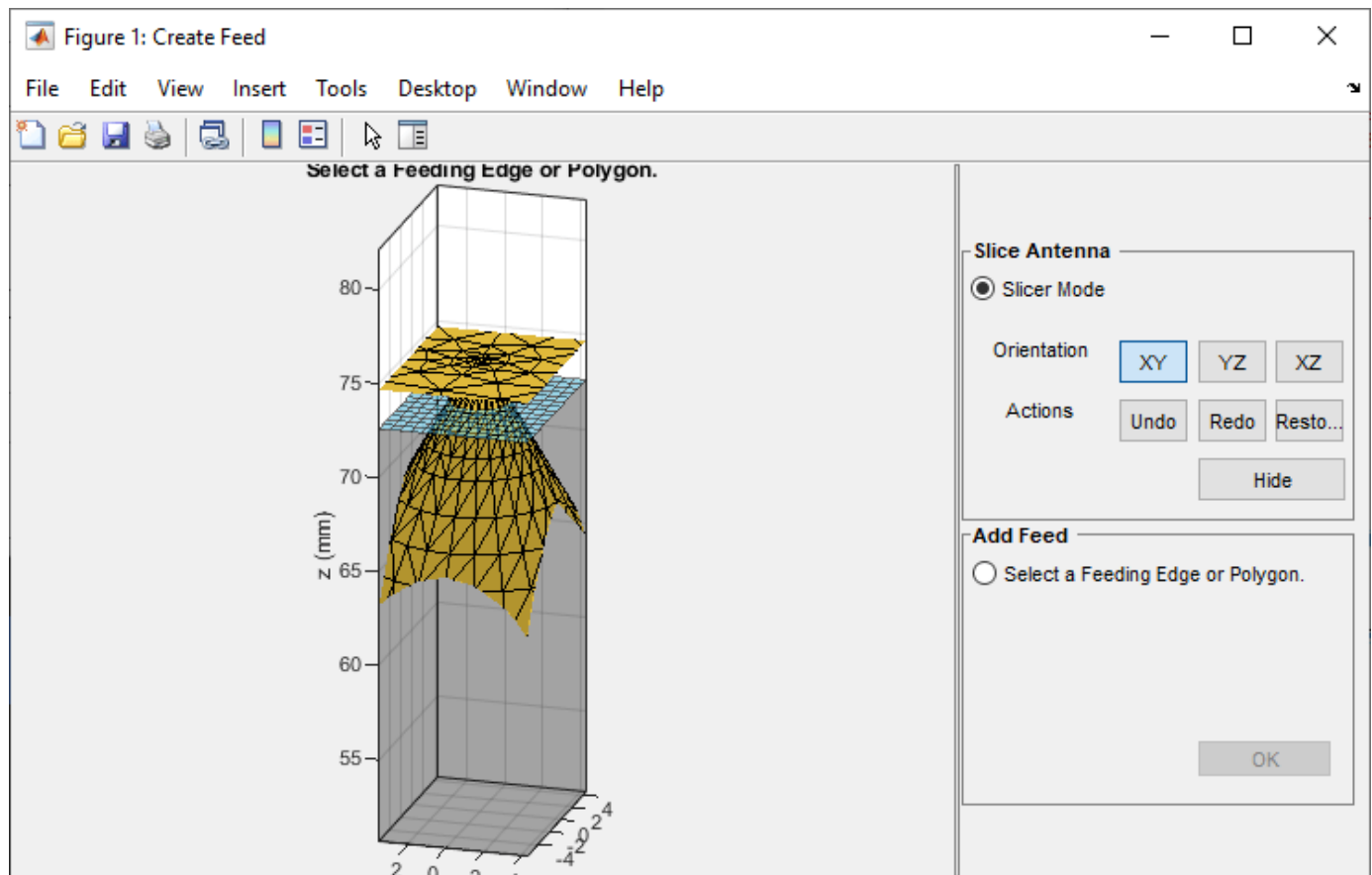


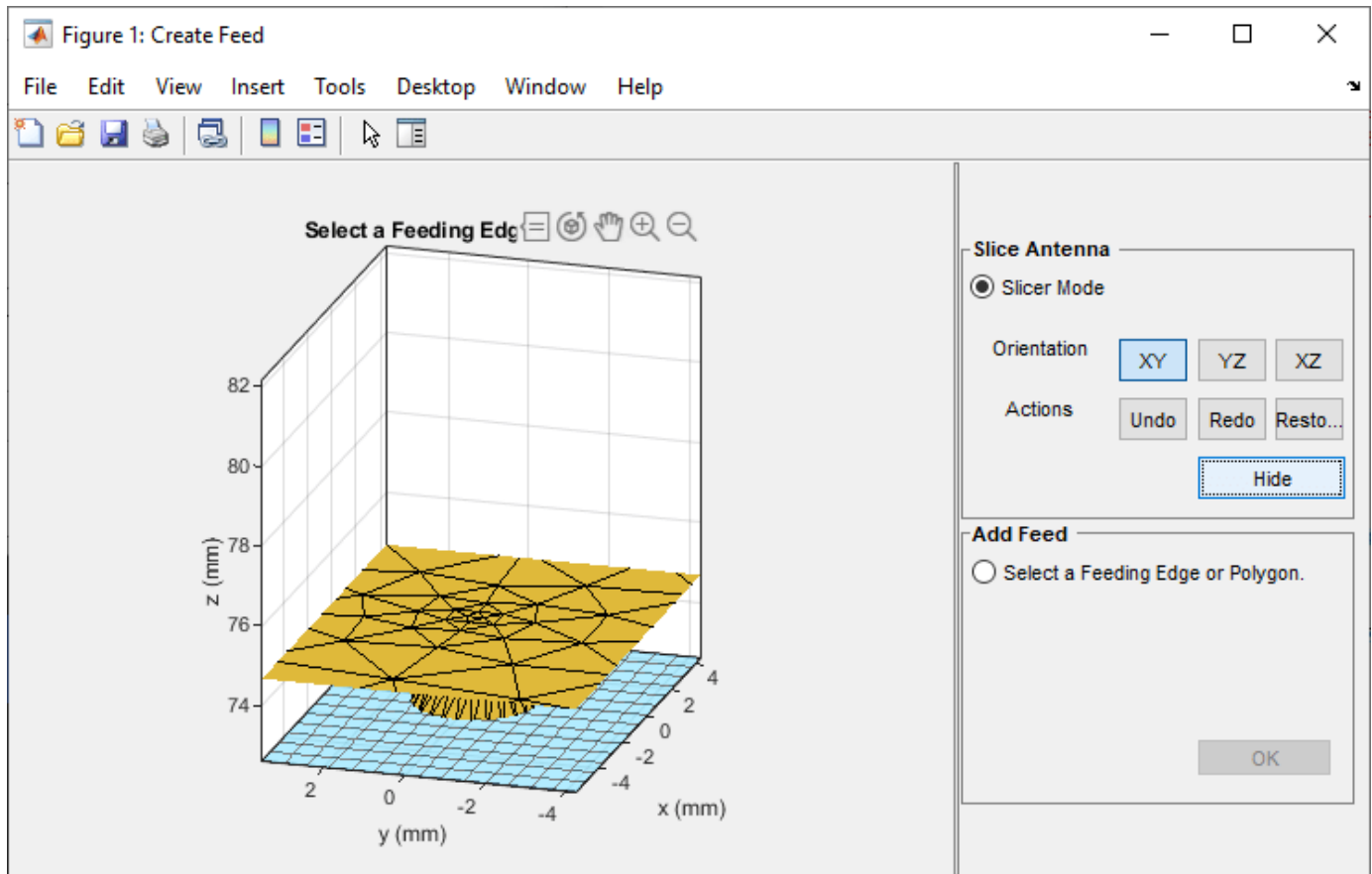


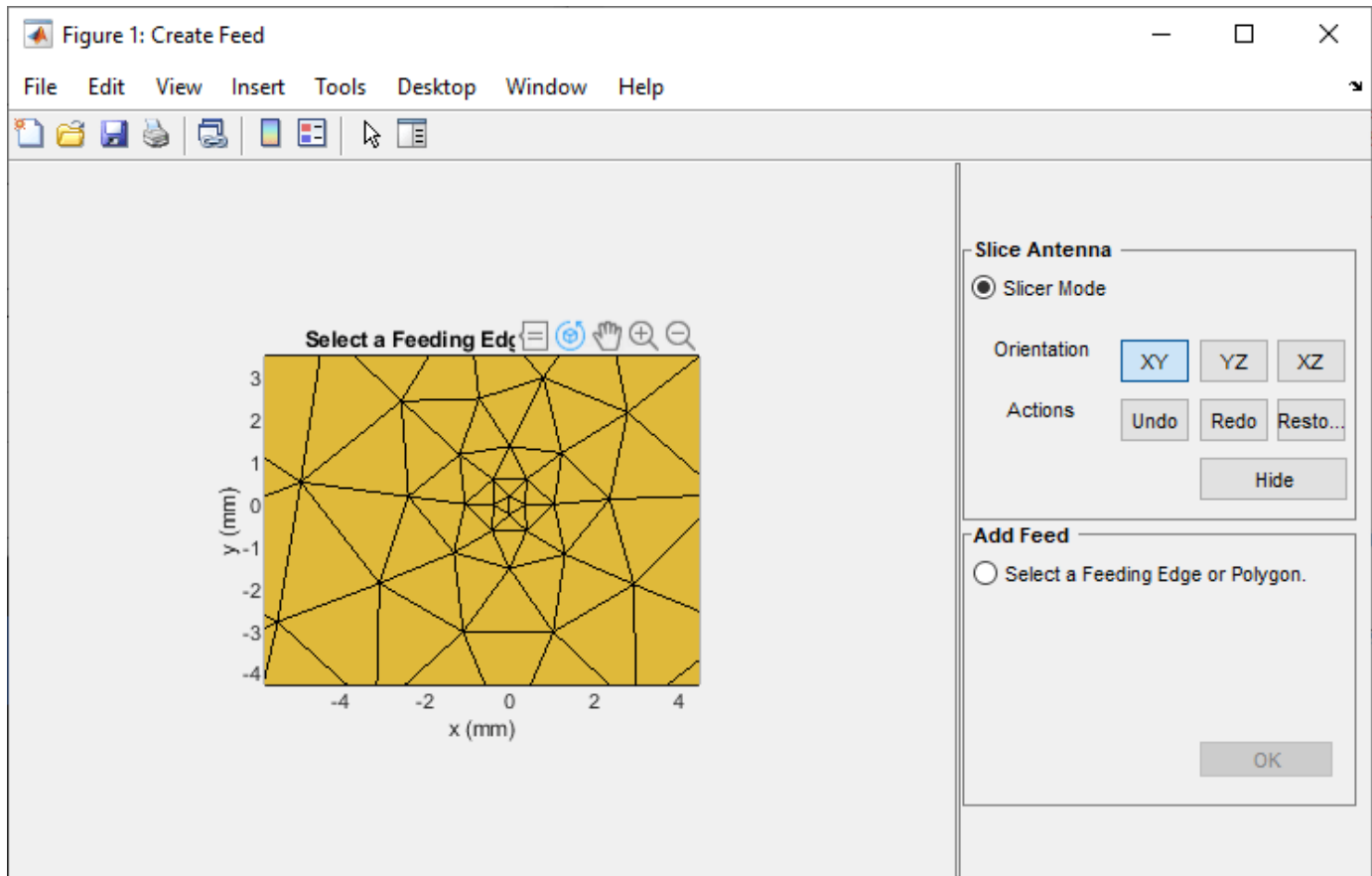












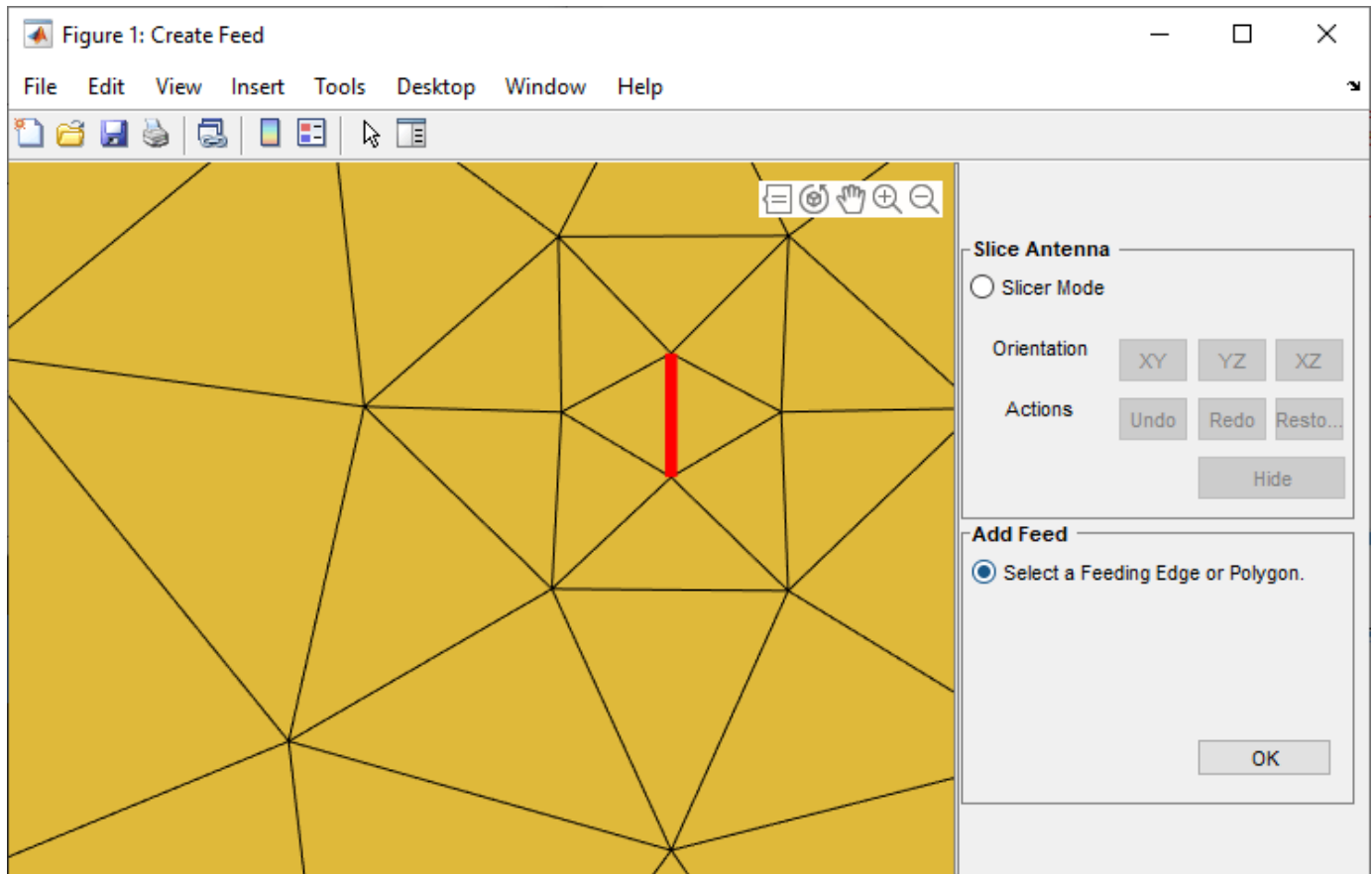
Use the **Undo**, **Redo** and **Restore** buttons to undo the changes created using hide option, redo the changes, and restore to the initial state, respectively.

Select Feeding Edge or Polygon

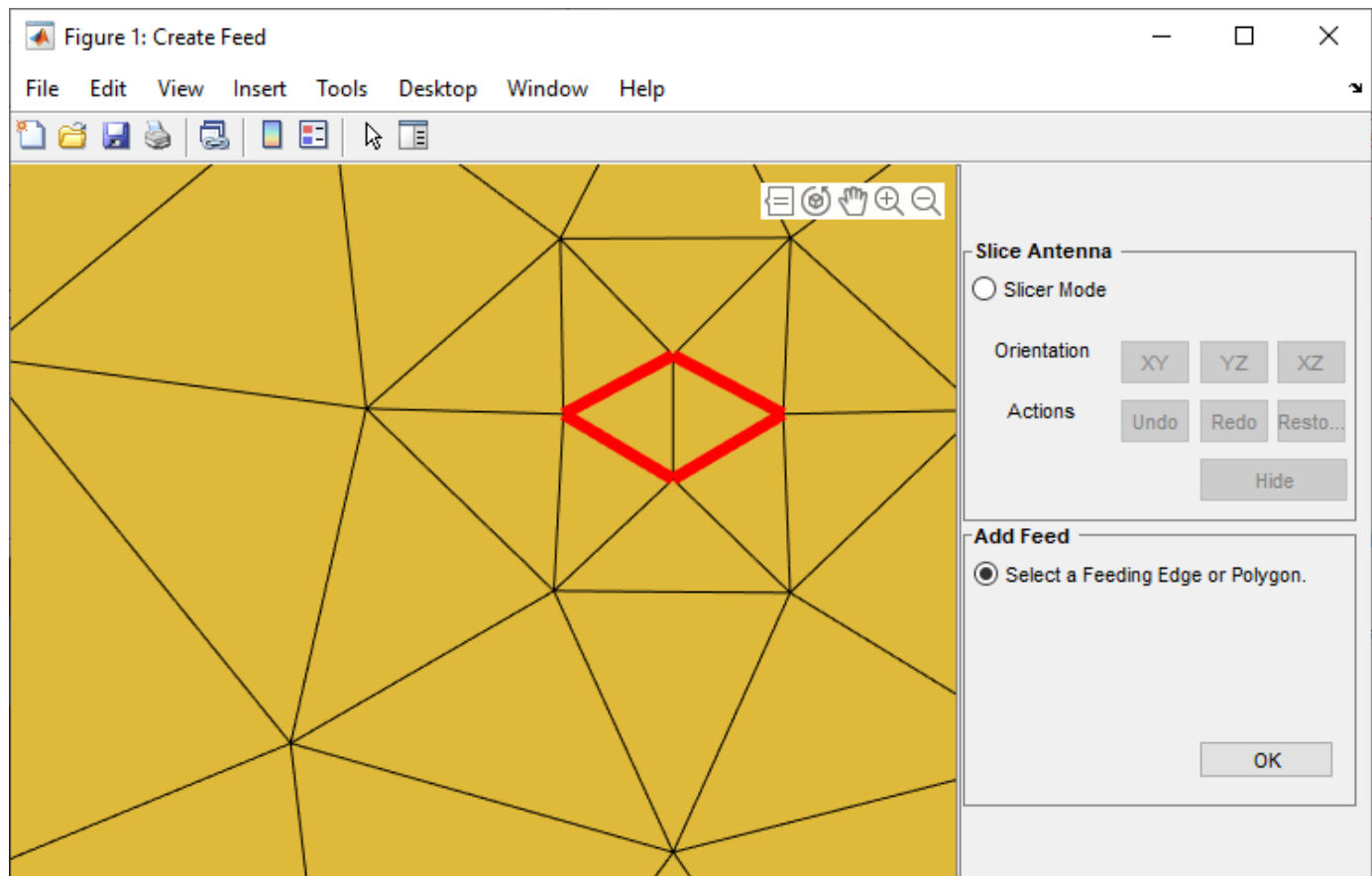
Select **Select a Feeding Edge or Polygon** in the **Add Feed** pane to select the desired feeding edge or feeding polygon. Click the edge to select it, and click it a second time to deselect it.

The feed can be created either selecting a single edge or selecting a group of edges that forms a polygon.

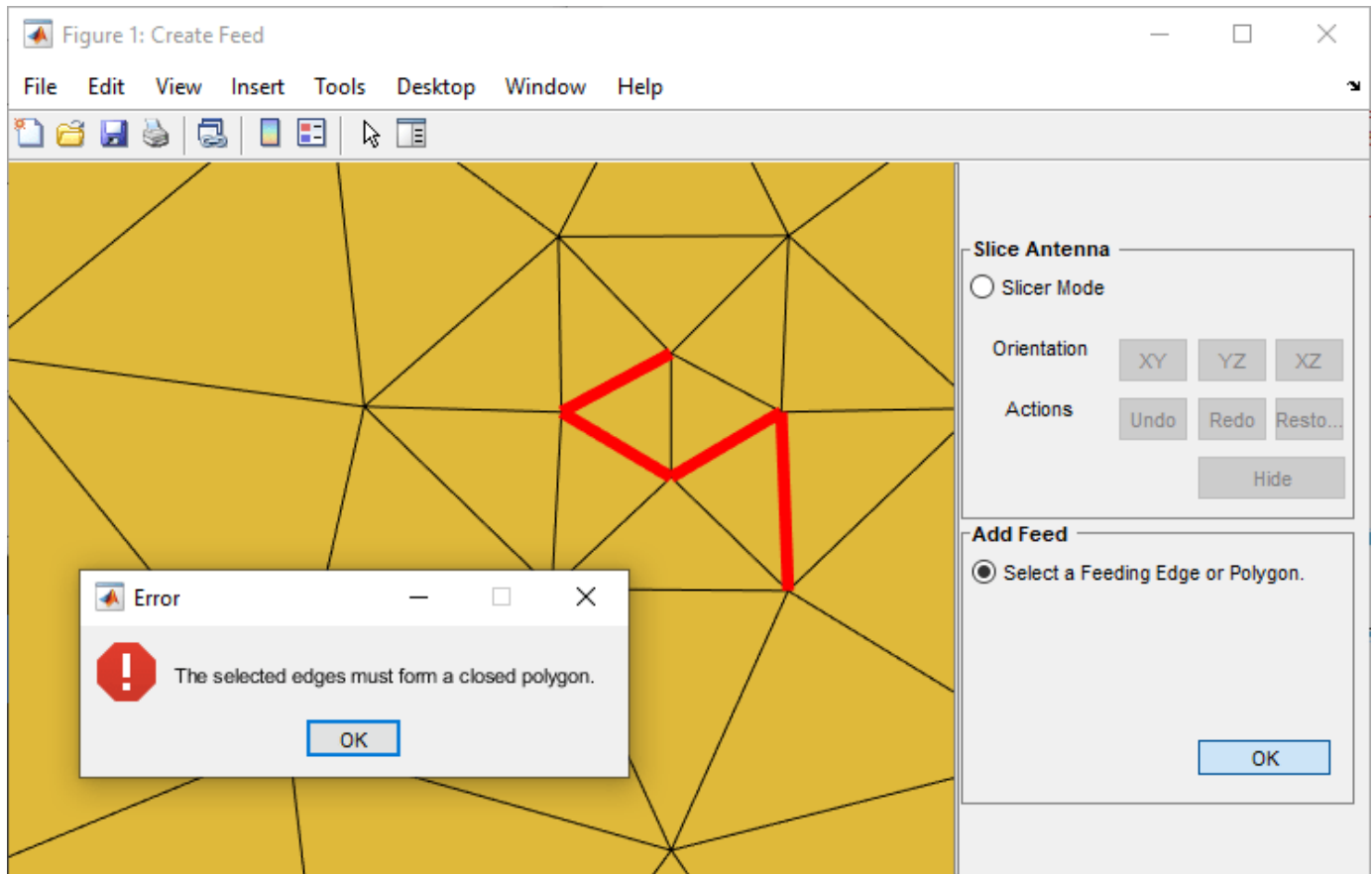
The image below shows how to select a feed as a single edge.



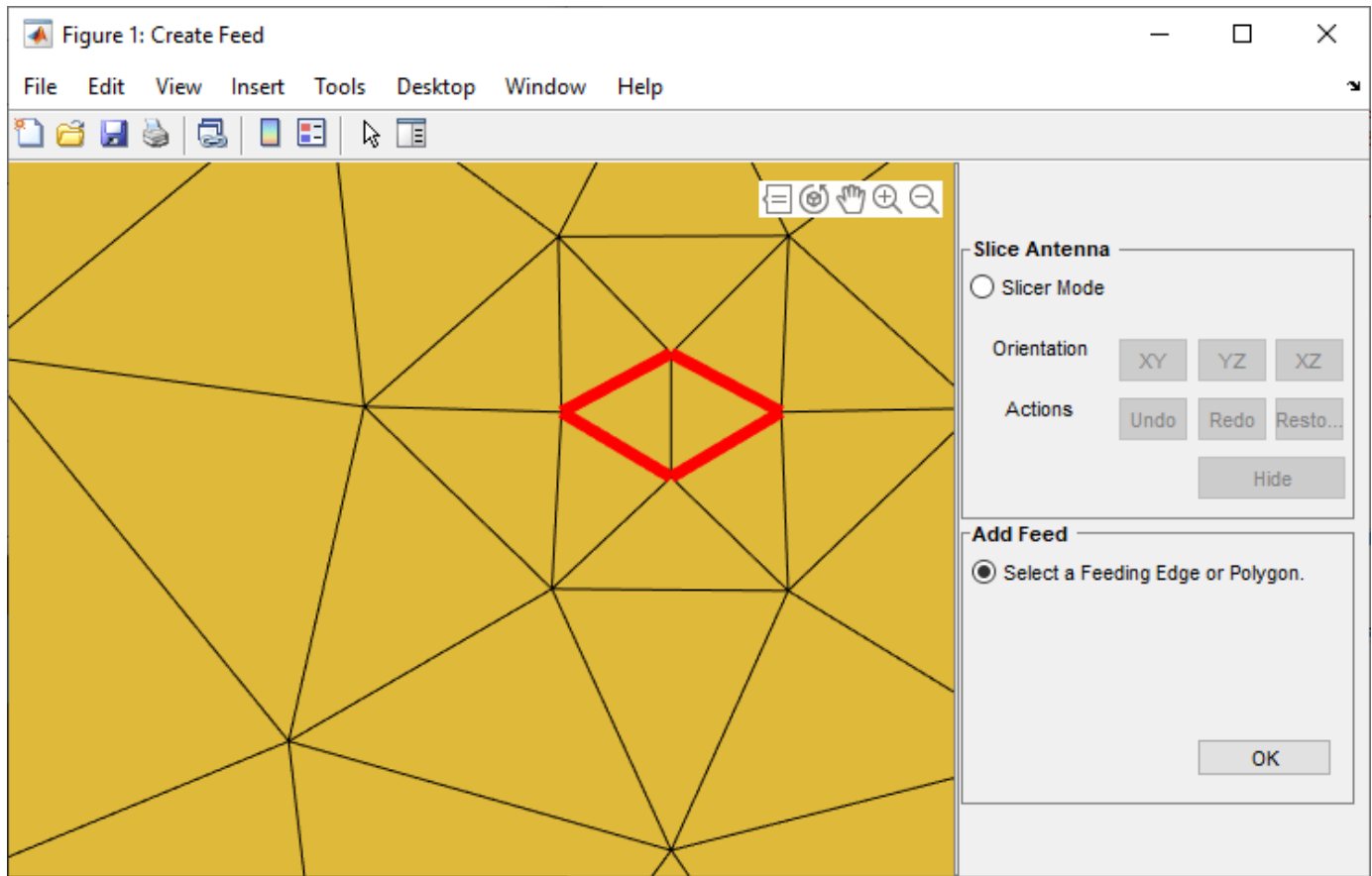
Select the edges that form a closed polygon.



The selected edges must be connected to other edges. If they are not, the UI figure window displays an error when you click **OK**.

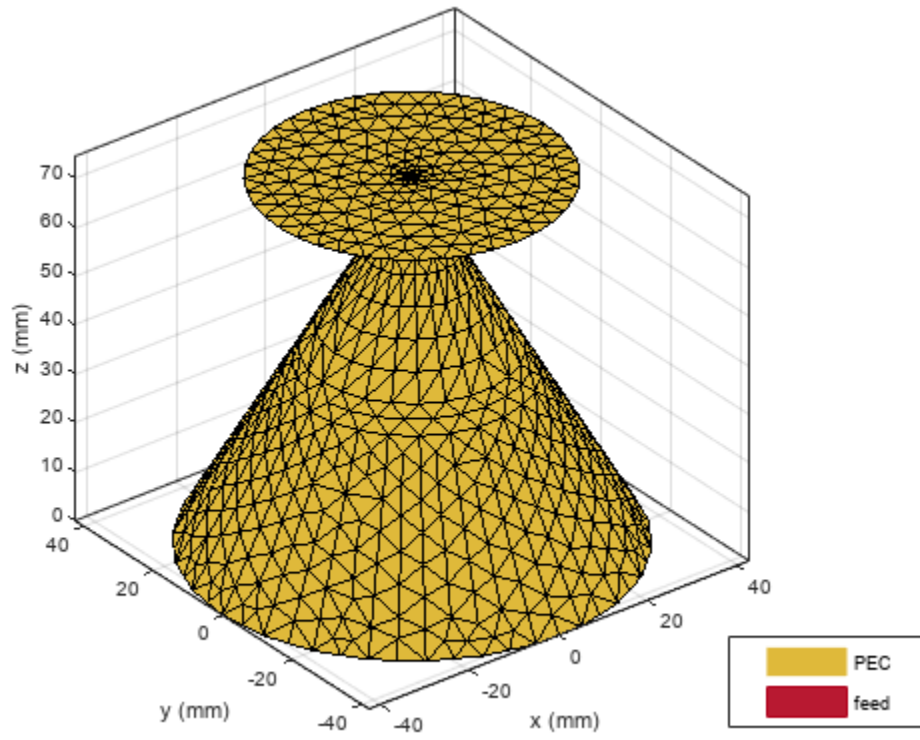


In this example, the feed is created by selecting a polygon.

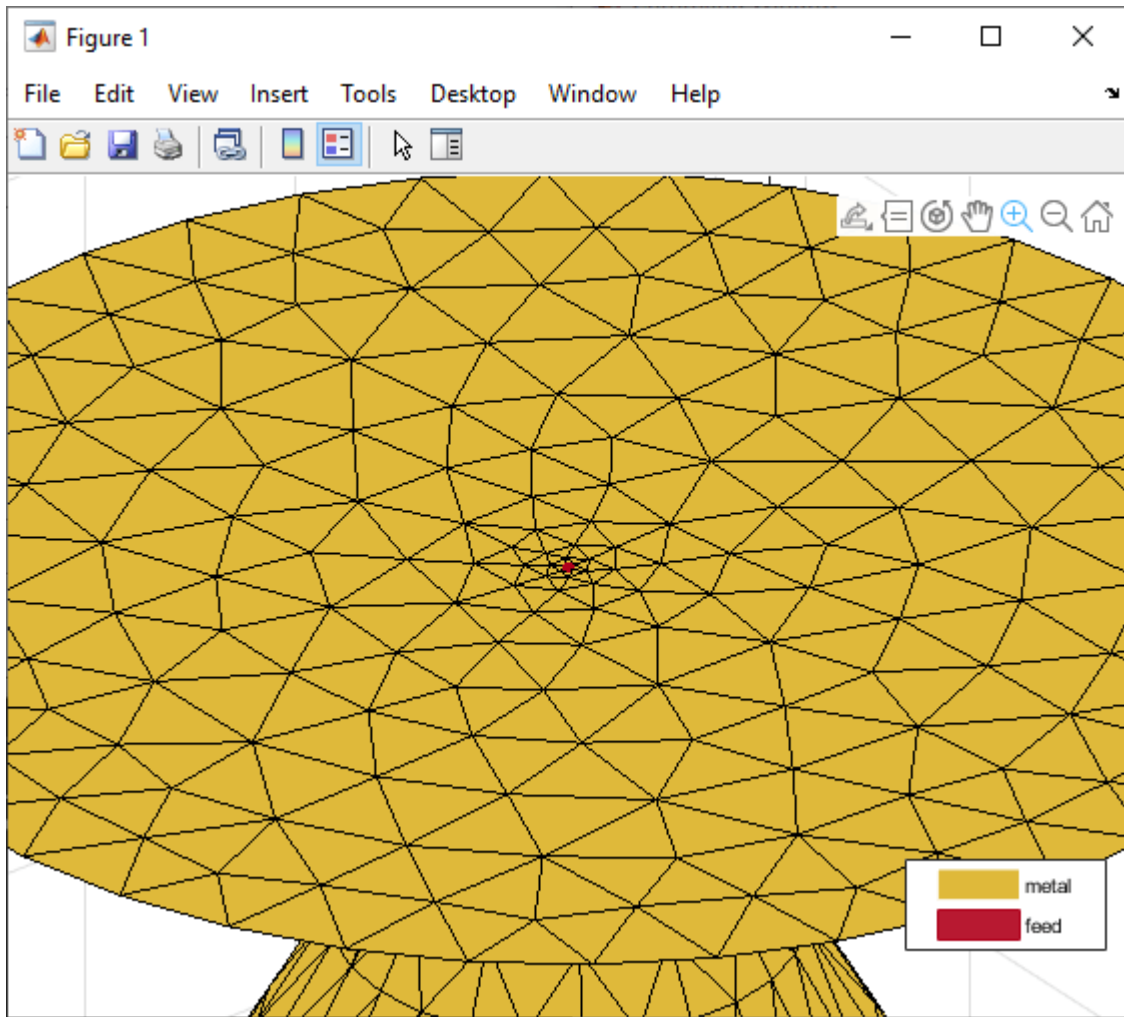


Click **OK** to define the selected edges as feeding edges and the structure with the feed is displayed

```
figure  
show(c)
```



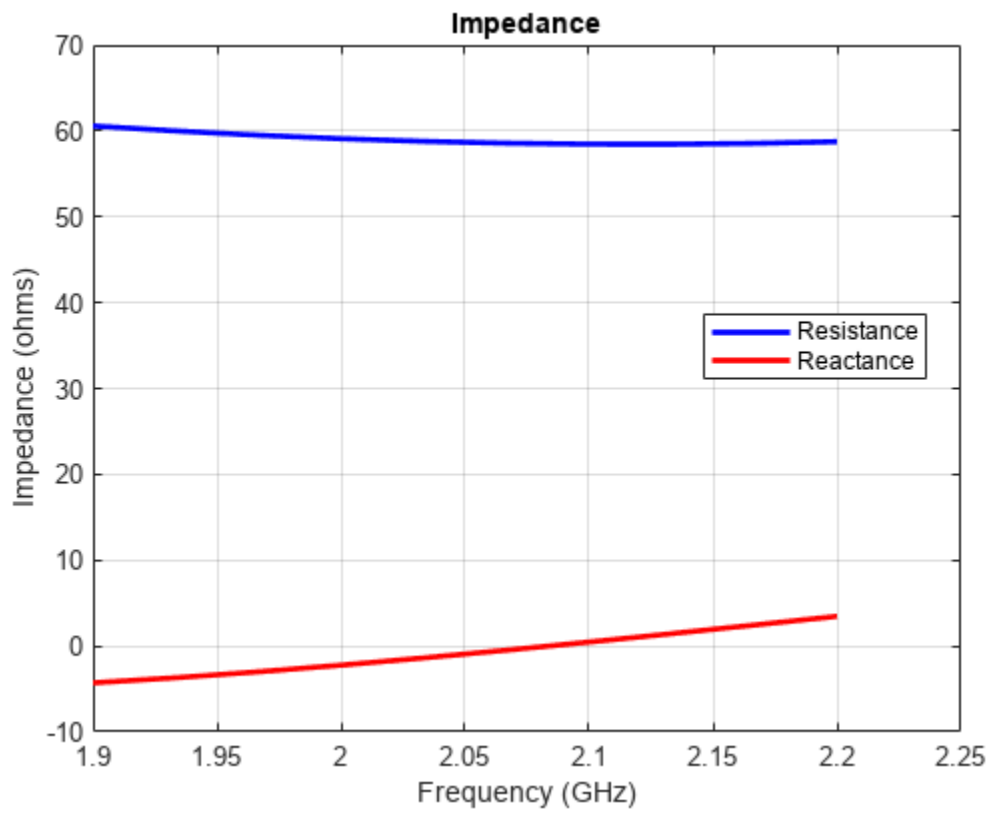
The feed is created at the mean of all the points of the polygon.



Analyze Antenna

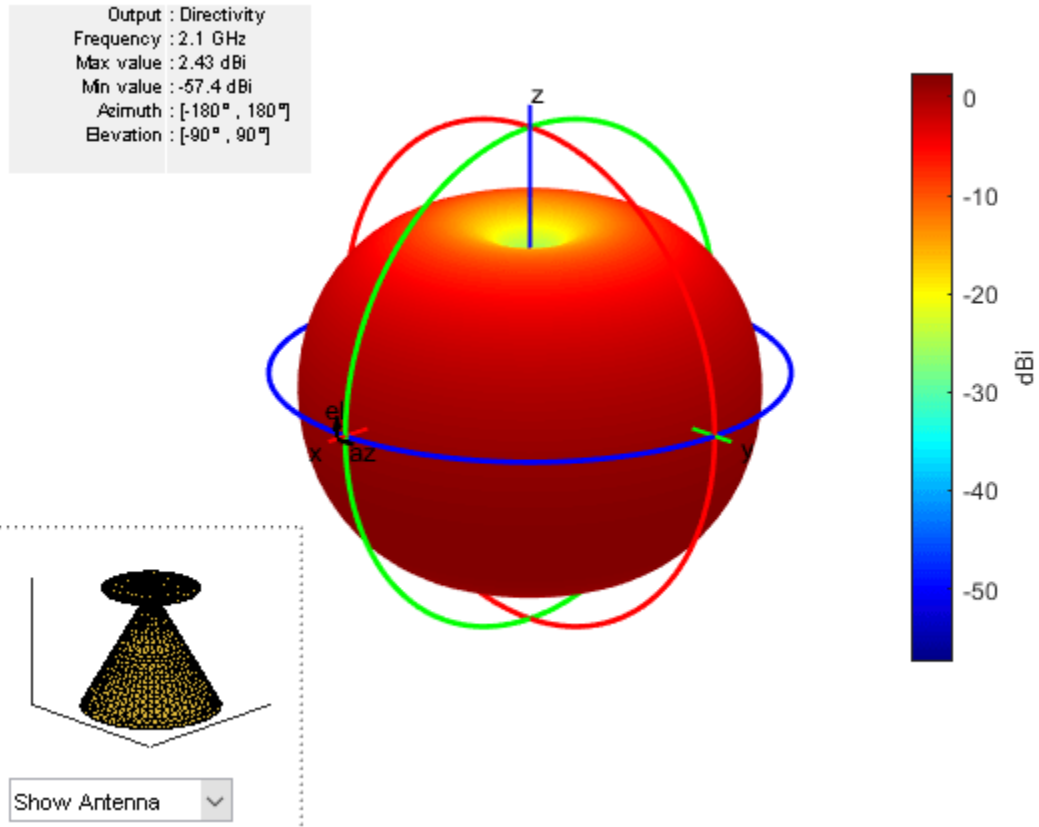
Analyze the antenna geometry using impedance and directivity plots.

```
figure  
impedance(c,linspace(1.9e9,2.2e9,10))
```



The antenna structure resonates at 2.1 GHz. Plot the directivity of the antenna at resonant frequency.

```
figure  
pattern(c,2.1e9)
```



See Also

“Pattern Analysis of the Symmetric Parabolic Reflector” on page 5-570

Model, Analyze and Compare Horn Antennas

This example shows how to model, analyze the pattern and compare the gains of different types of horn antennas. Horn antennas are essentially a section of a waveguide where the open end is flared to provide a transition to the areas of free space. This antenna is a simple development of a waveguide transmission line. It is possible to leave a waveguide open and let signal radiate, but this is not efficient. Signals passing along the waveguide see a sudden transition from the waveguide to free space, which cause the signals to be reflected back along the waveguide as standing waves. To overcome this issue the waveguide can be tapered out or flared, for providing a gradual transition from the impedance of the waveguide to that of free space.

Rectangular Horns

This type of horn antennas has a rectangular shaped waveguide, typically fed by a monopole. In such horns, flaring is done in both E-plane & H-plane walls of the rectangular waveguide. Aperture of the horn is rectangular. The parameters listed below will be used in modeling this type of horn.

FL = Flare Length of the horn

FW = Flare Width of the horn

FH = Flare Height of the horn

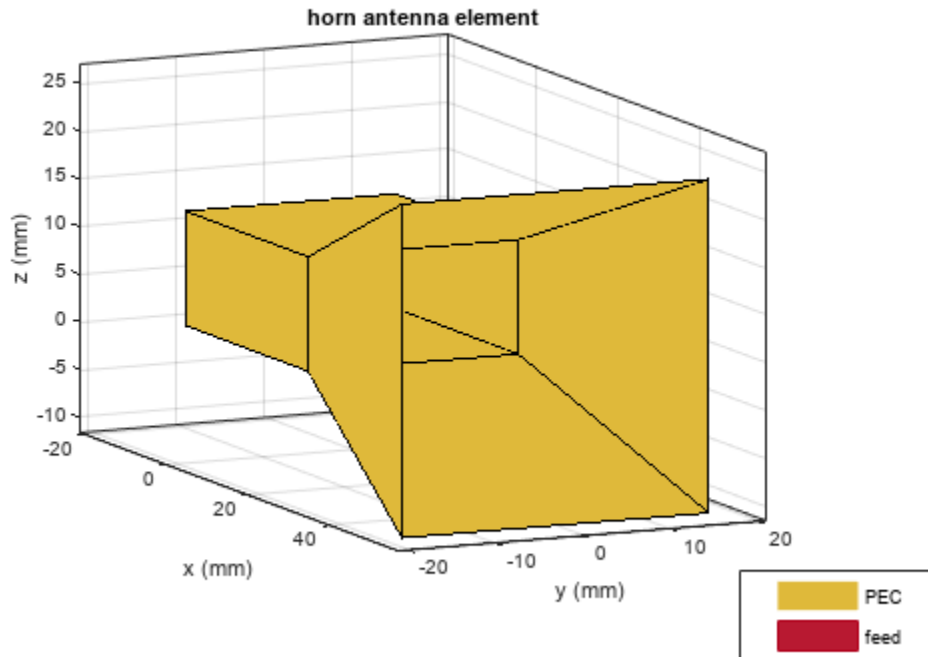
L = Length of the waveguide

W = Width of the waveguide

H = Height of the waveguide

For example:

```
FL = 0.0348;
FW = 0.035;
FH = 0.035;
L = 0.03;
W = 0.024;
H=0.012;
F0=[0.01 0];
ant1 = horn('FlareLength',FL,'FlareWidth',FW,'FlareHeight',FH,'Length',L,...
            'Width',W,'Height',H,'FeedOffset',F0);
show(ant1);
```



Conical Horn

This conical horn has a circular shaped waveguide. Aperture of this antenna is circular rather than rectangular with a cone shaped structure coming out of the waveguide. The parameters listed below will be used in modeling this type of horn.

R = Radius of the circular waveguide

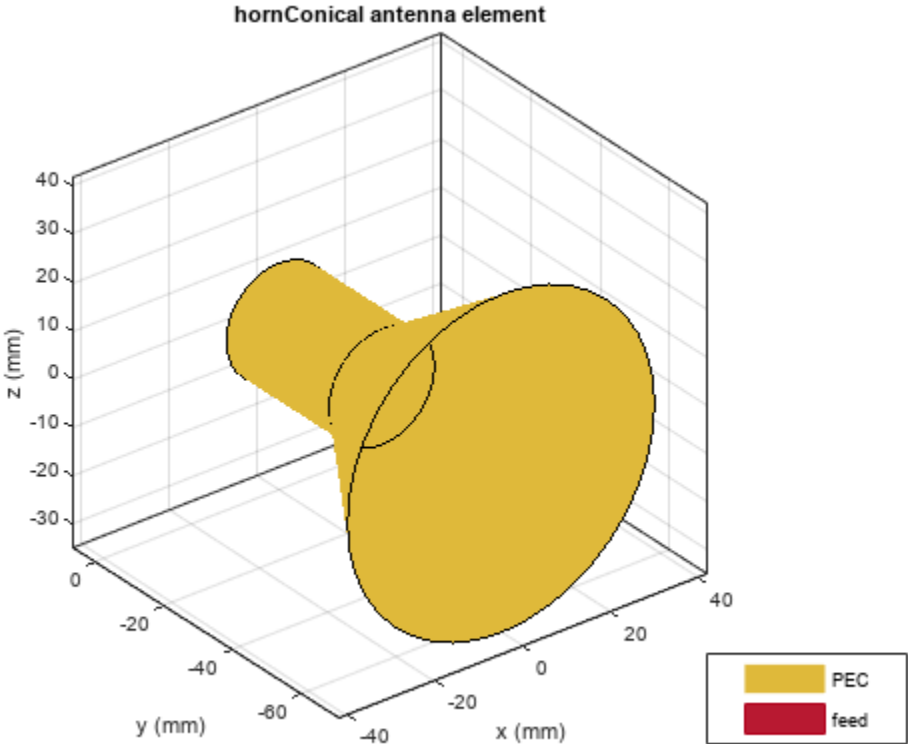
WH = Height of the circular waveguide

CH = Height of the cone

AR = ApertureRadius of the cone

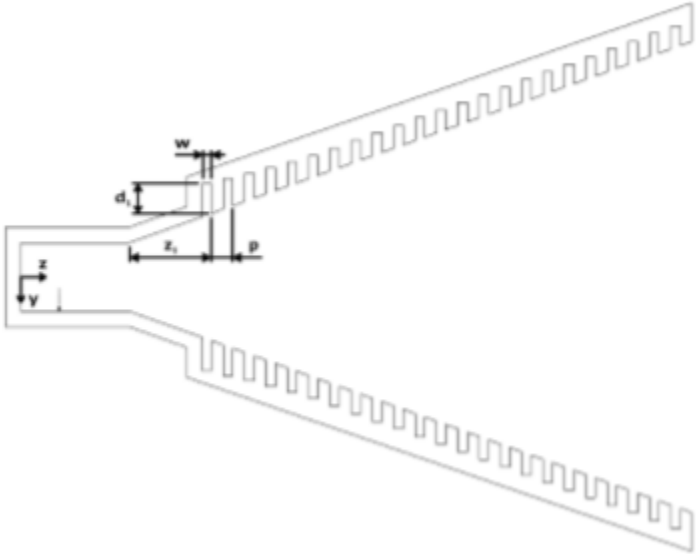
For example:

```
R = 0.012;
WH = 0.03;
CH = 0.0348;
AR = 0.035;
ant2 = hornConical('Radius',R,'WaveguideHeight',WH,'ConeHeight',CH,'ApertureRadius',AR);
show(ant2);
```

Conical Corrugated Horn

The corrugated horn antenna is like an extension for the conical horn with some additional features. This antenna consists of slots or grooves covering the inner surface of the cone. Typically, such type of horns has 5 to 6 corrugations per wavelength.



$z1$ = FirstCorrugateDistance, Distance from end of the waveguide to the first corrugate. Corrugations on the surface of the cone start with a metal followed by a groove and repeats itself along the length of the cone. So, the distance is measured from the end of the waveguide to the end of the first groove

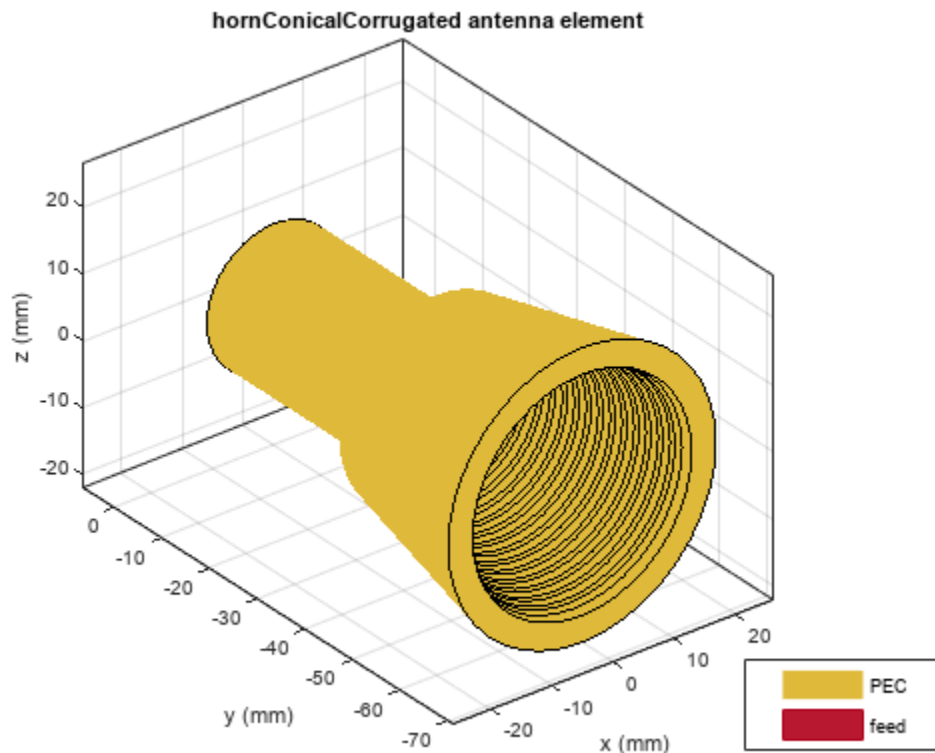
p = Pitch, Distance between two successive corrugations. Number of corrugations can be controlled with this property

w = CorrugateWidth, Width of the groove or corrugation.

$d1$ =CorrugateDepth, Depth of the corrugation. Typically, depth is selected such that it is equal to 0.25λ

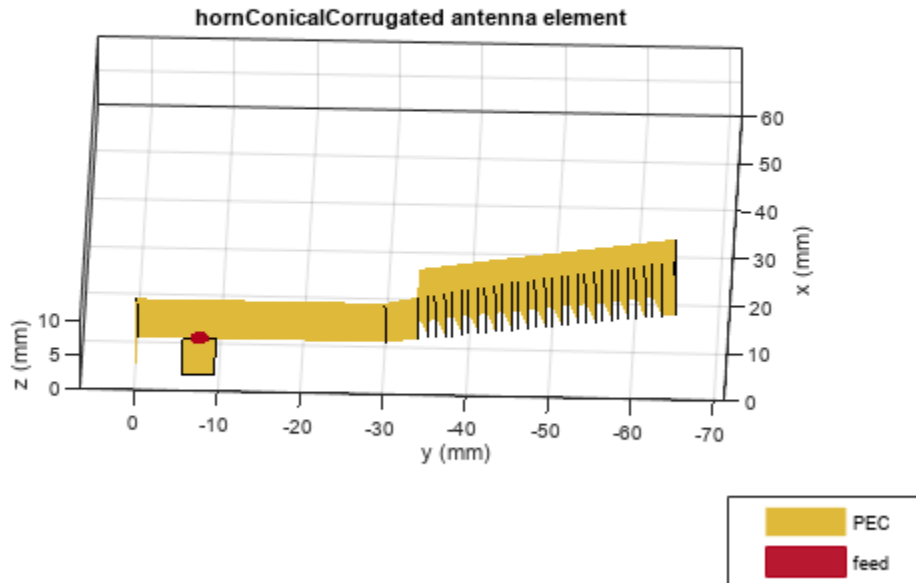
For example:

```
z1=0.006;
p=0.002;
w=0.0008;
d1=0.004;
ar=0.018;
ch=0.0348;
ant3 = hornConicalCorrugated('FirstCorrugateDistance',z1,'Pitch',p,'CorrugateWidth'...
                             ,w,'CorrugateDepth',d1,'ConeHeight',ch,'ApertureRadius',ar);
show(ant3);
```



Cross sectional view of the corrugated horn

```
xlim([0 60]);
zlim([0 10]);
view(-88,35);
```



Rectangular Corrugated Horn

Rectangular corrugated horn is an extension for horn antenna with corrugations along the width and height of the flare.

FL = Flare Length of the horn

FW = Flare Width of the horn

FH = Flare Height of the horn

L = Length of the waveguide

W = Width of the waveguide

H = Height of the waveguide

FCD = First Corrugate Distance

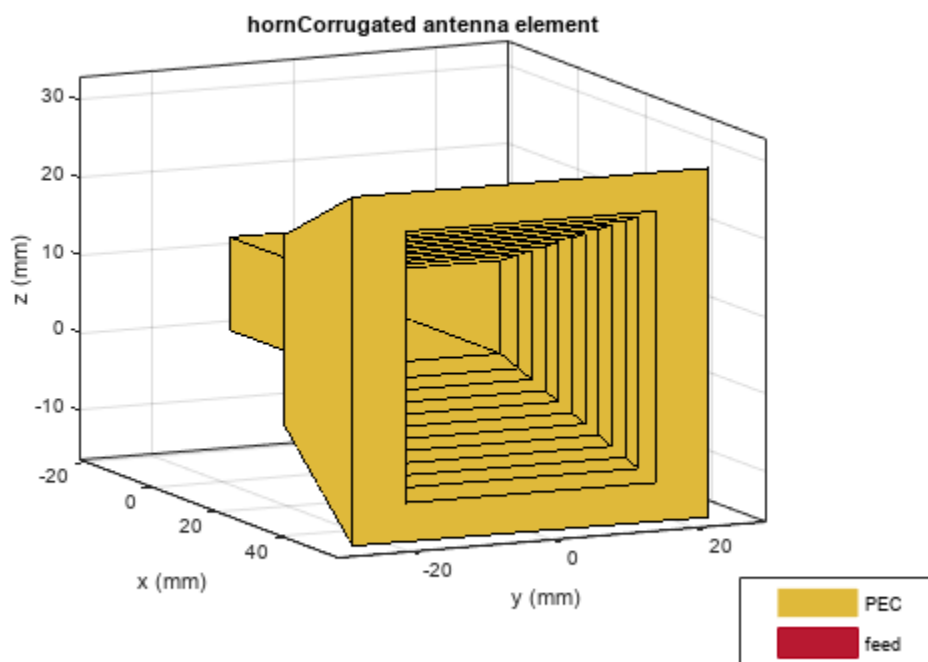
CD = CorrugateDepth

For example:

```

FL = 0.0348;
FW = 0.035;
FH = 0.035;
L = 0.03;
W = 0.024;
H=0.012;
FCD=0.01;
CD=[0.005 0.0075];
ant4 = hornCorrugated('FlareLength',FL,'FlareWidth',FW,'FlareHeight',FH,'Length',L,...
    'Width',W,'Height',H,'FirstCorrugateDistance',FCD,'CorrugateDepth',CD);
show(ant4);

```



Applications

Horn antennas are widely used in satellite communications as feed antennas in cassegrain, parabolic reflector antennas. These antennas direct the beam towards the reflector. Let us compare the gain provided by these antennas in such applications.

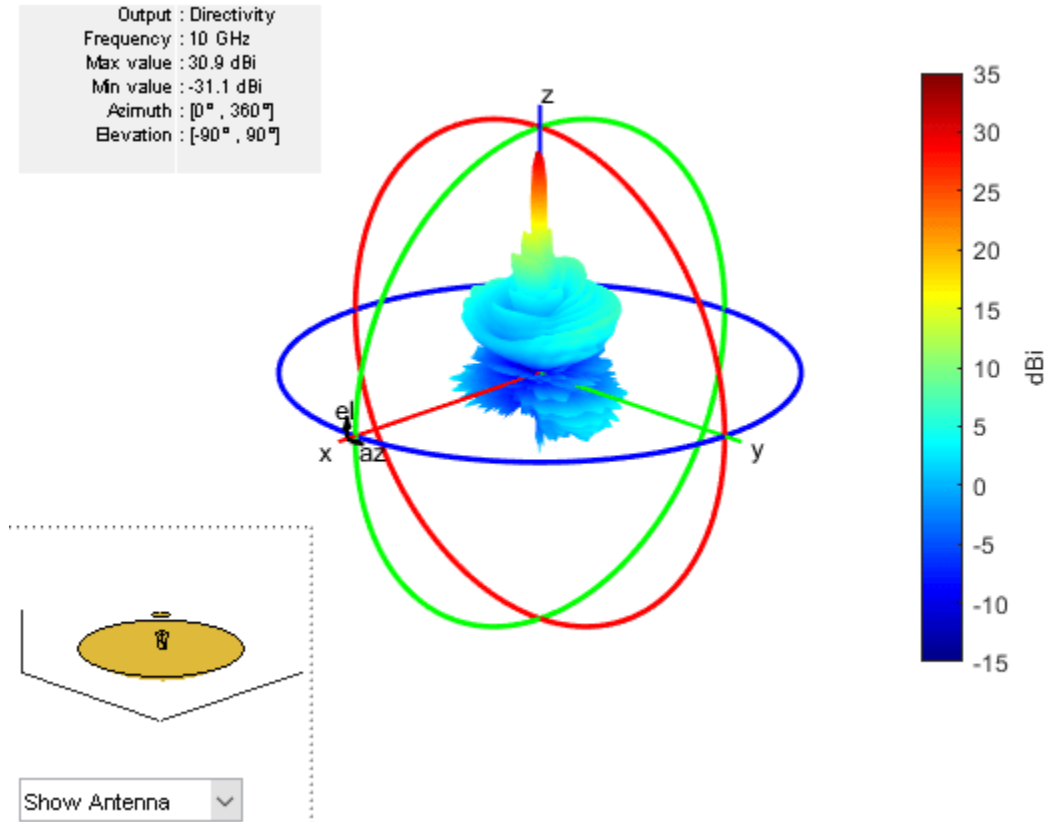
For the cassegrain antenna with the horn as the exciter, analyze the pattern at 10 GHz. Create a `PatternPlotOptions` object to rescale the magnitude for the plot.

```

ant5=cassegrain;
ant5.Exciter=ant1;
ant5.Exciter.Tilt=270;
ant5.Exciter.TiltAxis=[0 1 0];
az = 0:2:360;
el = -90:1:90;

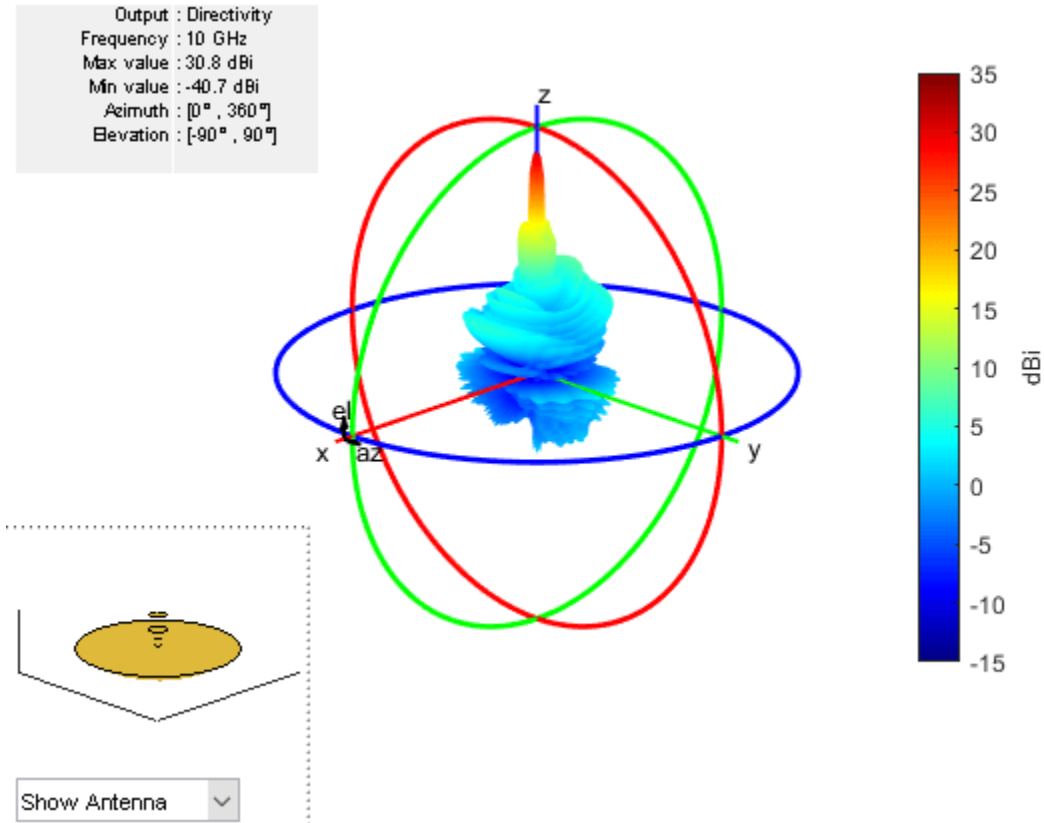
```

```
patOpt = PatternPlotOptions;
patOpt.MagnitudeScale = [-15 35];
pattern(ant5,10e9,az,el, 'patternOptions',patOpt);
```



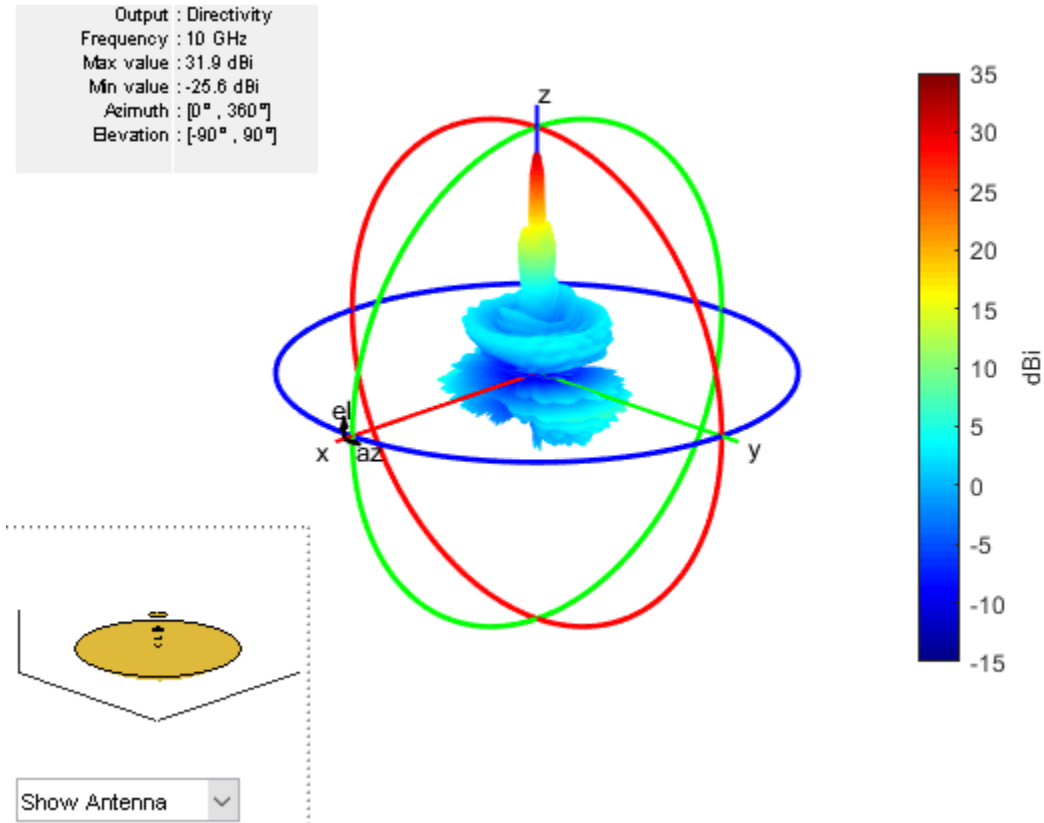
For the cassegrain antenna with the hornConical as the exciter,analyze the pattern at 10 GHz

```
ant5.Exciter=ant2;
ant5.Exciter.Tilt=-90;
figure;
pattern(ant5,10e9,az,el, 'patternOptions',patOpt);
```



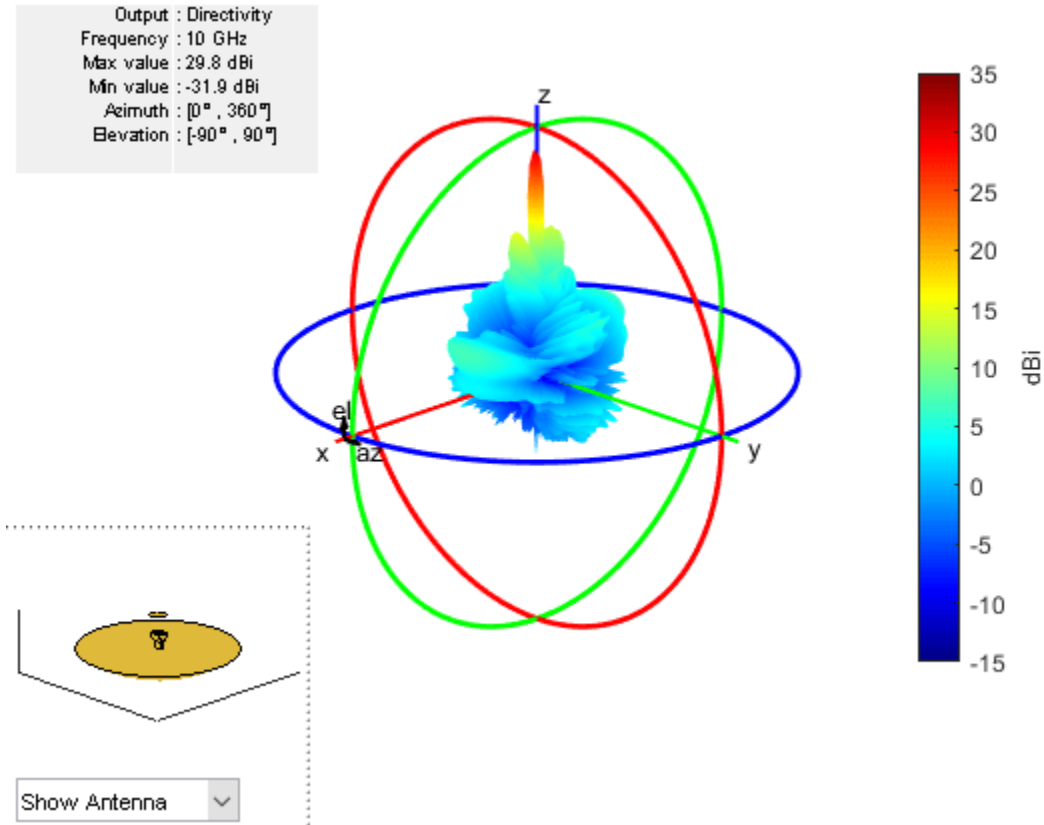
For the cassegrain antenna with the hornConicalCorrugated as the exciter, analyze the pattern at 10 GHz. The gain provided by the corrugated horn antenna is more compared to other horns due to corrugate depth and pitch of the corrugations which result in lower side lobes.

```
ant5.Exciter=ant3;
ant5.Exciter.Tilt=-90;
figure;
pattern(ant5,10e9,az,el, 'patternOptions',pat0pt);
```



For the cassegrain antenna with the hornCorrugated as the exciter, plot the radiation pattern at 10 GHz.

```
ant5.Exciter=ant4;
ant5.Exciter.Tilt=270;
ant5.Exciter.TiltAxis=[0 1 0];
figure;
pattern(ant5,10e9,az,el, 'patternOptions',pat0pt);
```



Conclusion

Horn antennas often have a directional radiation pattern with a higher antenna gain and are relatively easy to manufacture.

References

- [1] P. Piksa, "Comparison of conical horn with optimized corrugated surface and corrugated horn," Proceedings of 21st International Conference Radioelektronika 2011, Brno, 2011, pp. 1-3.
- [2] R. P. Jadhav, V. Javnakash Dongre and A. Heddallikar, "Design of X-Band Conical Horn Antenna Using Coaxial Feed and Improved Design Technique for Bandwidth Enhancement," 2017 International Conference on Computing, Communication, Control and Automation (ICCCUBEA), Pune, 2017, pp. 1-6.

See Also

"Design and Analyze Cassegrain Antenna" on page 5-585

Multiband Nature and Miniaturization of Fractal Antennas

This example shows how to analyze the multi band characteristics of fractal antennas and compare the size of fractal antennas with conventional antennas. The term fractal means irregular fragment. A Fractal is a rough or fragmented geometric shape that can be subdivided in parts, each of which is a reduced size of the whole. Fractal geometries are applied to antenna elements which aids in creating antennas of compact size and multi band behavior. These characteristics are required in small and complex circuits. Fractal antennas are multiband, high gain, low profile antennas. They are compact relative to the conventional antennas because of their efficient space filling nature.

The geometry generating process of a fractal begins with a basic geometry referred to as the initiator. The final fractal geometry is an infinitely intricate underlying structure such that no matter how closely the structure is viewed, the fundamental building blocks cannot be differentiated because they are scaled versions of the initiator. Different fractal antennas are koch dipole, koch loop, snowflake, seirpinski carpet, seirpinski gasket and fractal Island.

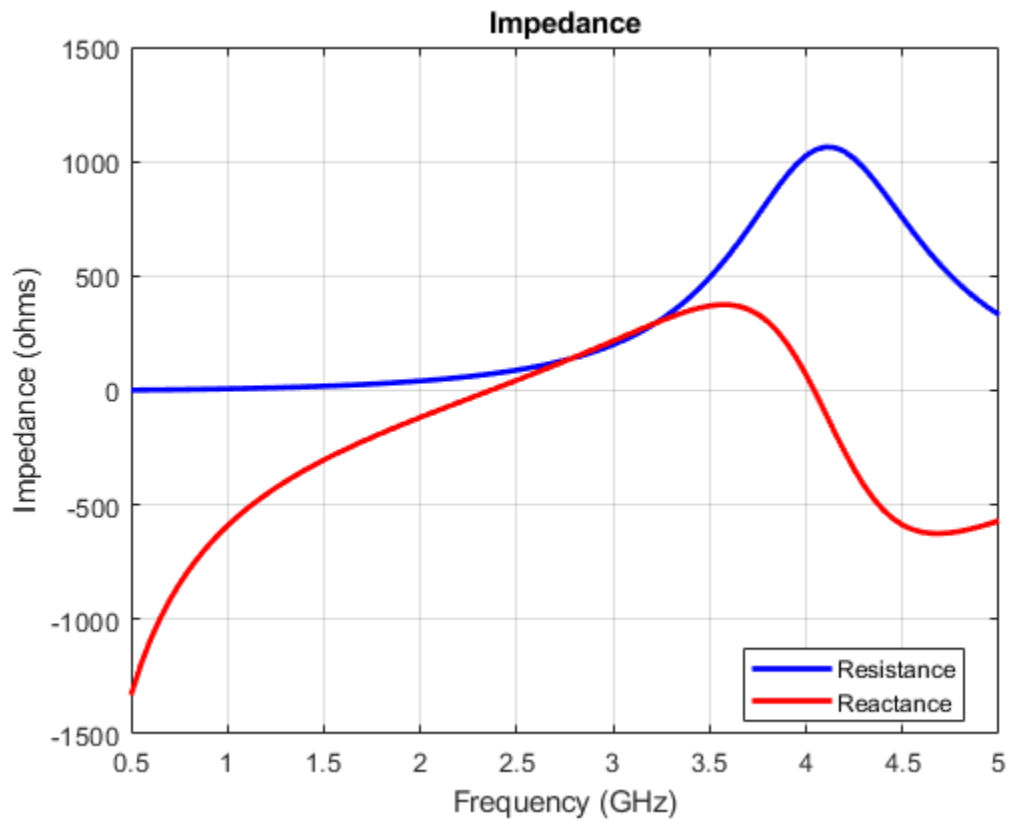
Koch Curve

A line of unit length is taken, middle third is removed and replaced with two lines of same length as removed one. These two lines and the segment removed should make up an equilateral triangle. This is first iteration. Same procedure is repeated with all the lines for next iterations. Each smaller segment is exact replica of the whole curve. Each iteration adds length to the total curve which results in total length that is $(4/3)^n$ times the original length, where n is number of iterations.

Fractal Koch Dipole Vs Dipole Antenna

Create a dipole with length 60 mm and plot the impedance over the frequency range of 0.5 GHz-5 GHz.

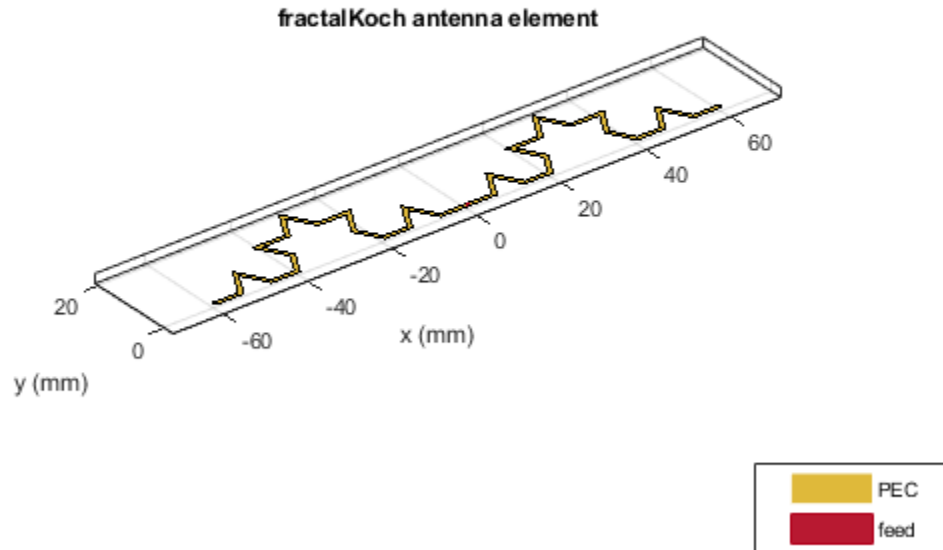
```
d = dipole('Length',60e-3,'Width',1e-3);
figure;impedance(d,(0.5:0.05:5)*1e9);
```



The resonant frequency of dipole antenna is 2.35 GHz.

Create a fractal koch dipole. By default, the length is 60 mm and number of iterations are two.

```
k = fractalKoch;  
figure; show(k);
```

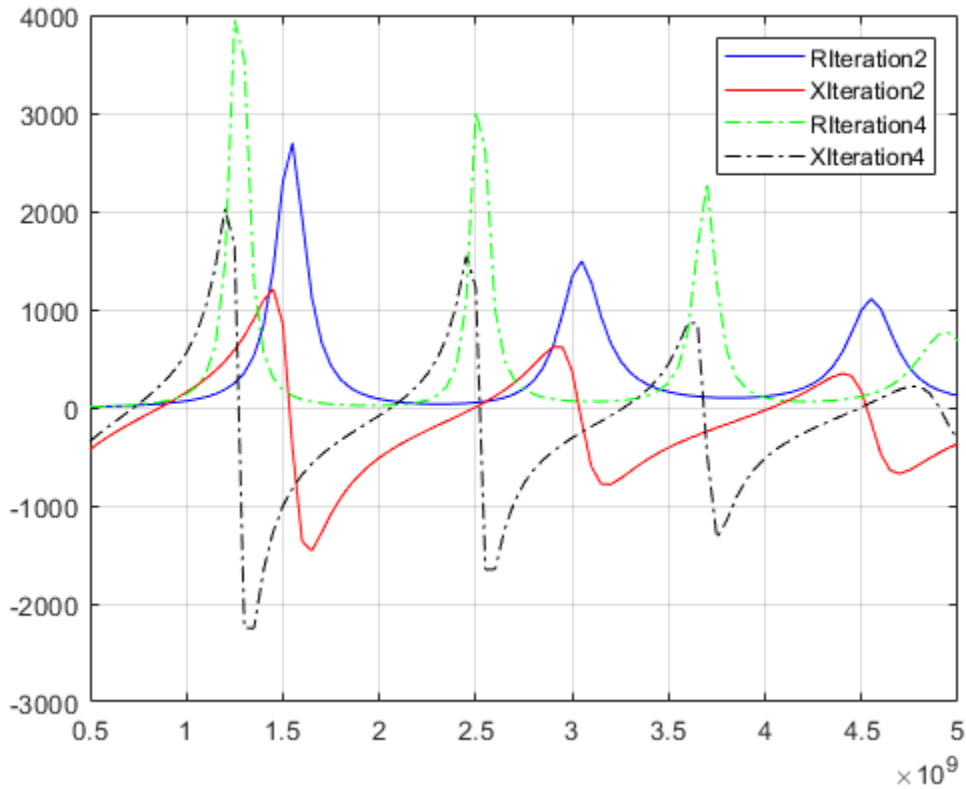


Calculate the impedance over the frequency range of 0.5 GHz-5 GHz.

```
zk1 = impedance(k, (0.5:0.05:5)*1e9);
```

Increase the number of iterations to four and calculate the impedance. Plot the impedance for second and fourth iterations.

```
k.NumIterations = 4;
k.Width = 0.5*k.Width;
zk2 = impedance(k, (0.5:0.05:5)*1e9);
figure; plot((0.5:0.05:5)*1e9, real(zk1), 'b', (0.5:0.05:5)*1e9, imag(zk1), 'r'); grid on;
hold on; plot((0.5:0.05:5)*1e9, real(zk2), '-.g', (0.5:0.05:5)*1e9, imag(zk2), '-.k');
legend('RIteration2', 'XIteration2', 'RIteration4', 'XIteration4');
```



In the legend, 'R' represents resistance curve and 'X' represents Reactance curve.

Tabulation of resonant frequencies, electrical lengths of dipole, fractal koch with second and fourth iteration.

```
Iteration = [0;2;4]; % Zeroth iteration corresponds to dipole antenna.
ResonantFrequencies_GHz = {'2.35';'0.87,1.54, 2.5, 3.04, 4.04, 4.53';'0.7, 1.25, 2.05, 2.5, 3.25, 3.7, 4.5'};
PhysicalLength = [60e-3;60e-3;60e-3];
ElectricalLength = [60e-3; 106.7e-3; 189.6e-3];
table(Iteration,ResonantFrequencies_GHz,PhysicalLength,ElectricalLength)
```

ans =

3x4 table

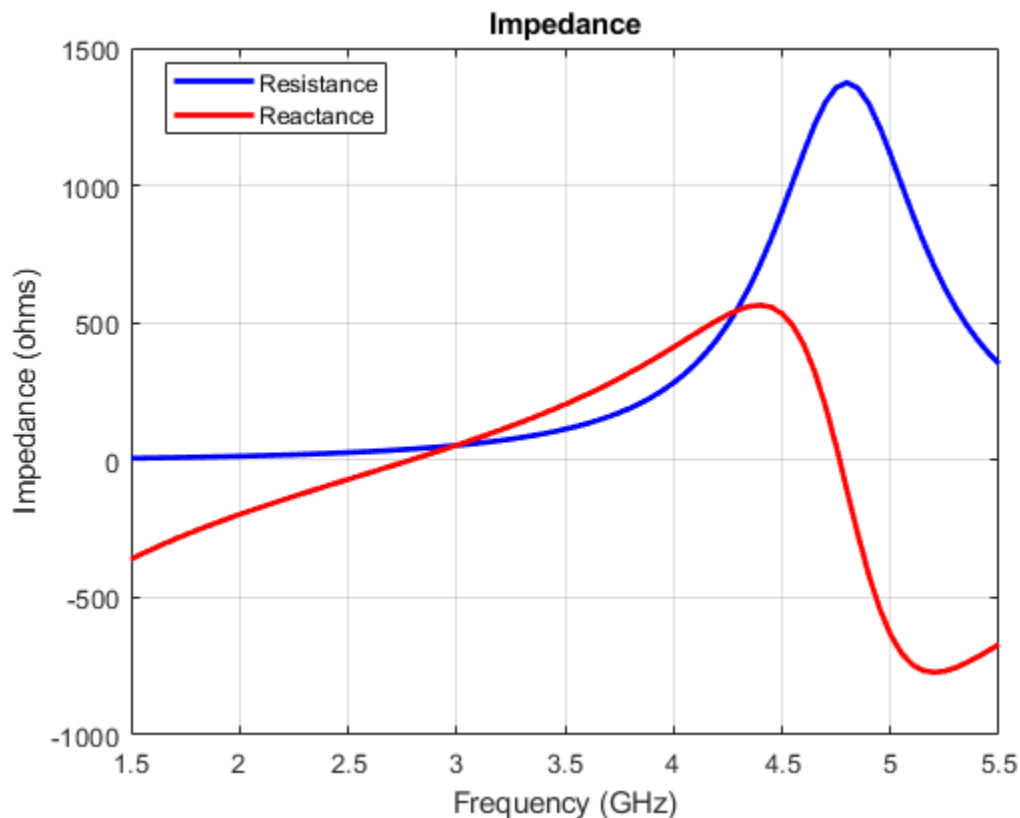
Iteration	ResonantFrequencies_GHz	PhysicalLength	ElectricalLength
0	{'2.35' }	0.06	0.06
2	{'0.87,1.54, 2.5, 3.04, 4.04, 4.53' }	0.06	0.1067
4	{'0.7, 1.25, 2.05, 2.5, 3.25,3.7,4.5' }	0.06	0.1896

With number of iterations equal to two, the antenna is resonating at six frequencies where the first resonating frequency is 0.87 GHz with the electrical length of koch 106.7 mm. The first resonant frequency is shifted left when compared to the resonant frequency of dipole antenna. This helps us to

create a koch dipole of physical length less than the length of dipole. With number of iterations equal to four, the antenna is resonating at eight frequencies where the first resonant frequency is 0.7 GHz. There is an increase in number of resonating frequencies and there is a left shift in the resonating frequencies when number of iterations are increased. The total length of the curve is 189.6 mm $((4/3)^n * (k.Length), n = 4)$.

Create a koch dipole antenna of four iterations with physical length of 19 mm, which makes electrical length 60 mm $((4/3)^4 * 19e-3 = 60e-3)$. And plot impedance over the frequency range of 0.5 GHz - 5 GHz.

```
antK = fractalKoch('Length', 19e-3);
figure; impedance(antK, (1.5:0.05:5.5)*1e9);
```

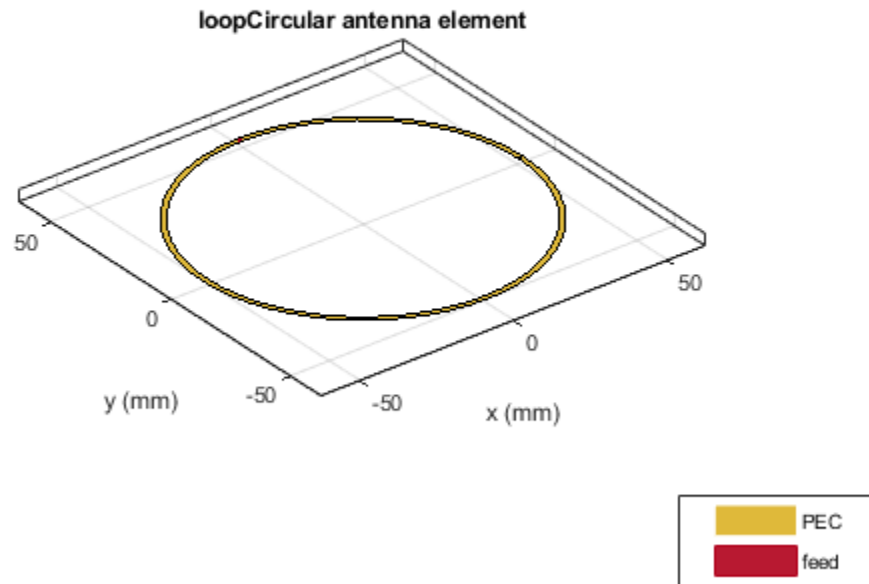


The impedance plot shows that the koch dipole antenna is resonating at 2.75 GHz which is closer to the dipole antenna resonating frequency. But the physical length of koch dipole is 19 mm where as the dipole length is 60 mm. So, the size of the antenna is reduced by 68 percentage. Miniaturization is one of the advantages of fractals.

Fractal Koch Loop Vs Circular Loop Antenna

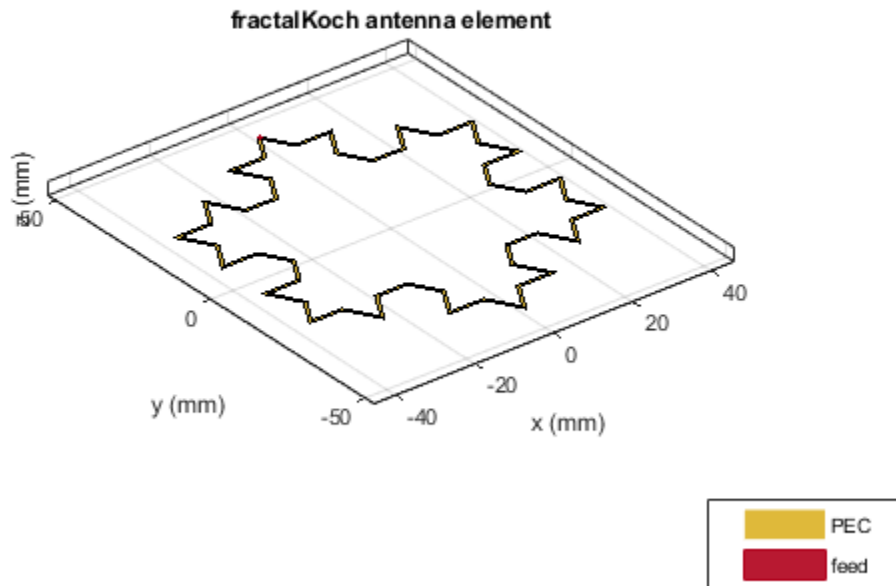
Create a circular loop antenna which resonates at 1 GHz.

```
antL = design(loopCircular, 1e9);
figure; show(antL);
```



Design a fractal koch loop antenna which resonates at 1 GHz. Koch loop antenna is constructed by using koch curve.

```
antkL = design(fractalKoch('Type','loop'),1e9);  
figure;show(antkL);
```



The radius of circular loop antenna is 52.1 mm. The perimeter is 327.2 mm. The radius of circle which circumscribes the koch loop of side 74 mm is $\sqrt{3} \cdot 74 \text{e-}3$, which is equal to 42.8 mm. This value is less than the radius of circular loop antenna for the same resonant frequency 1GHz. The perimeter with two iterations of koch loop is 395.4 mm ($3 \cdot \sqrt{3} \cdot (4/3)^n \cdot \text{radius}$). Further, the perimeter of the koch loop can be increased by increasing number of iterations. With number of iterations equal to four, the perimeter of loop is 702.9 mm which is 2.14 times longer than the perimeter of circular loop. The advantage is the perimeter increases while maintaining the same volume occupied.

Fractal Carpet Antenna

A square is taken and virtually (zeroth iteration) divided into 9 smaller congruent squares, each of which is one-third of the original square and the center square was removed to get the first iteration. Similarly, for the second iteration subdivide each of the eight remaining solid squares into 9 congruent squares and remove the center square. Continue in the same fashion to obtain further iterations of the carpet.

Create a fractal carpet antenna with a substrate of dielectric constant 4.4 and loss tangent 0.03 and with number of iterations equal to one.

```
antC = fractalCarpet;
antC.NumIterations = 1;
sub = dielectric('EpsilonR',4.4,'LossTangent',0.03);
antC.Length = 27.9e-3;
antC.Width = 37.25e-3;
antC.Substrate = sub;
```

```

antC.Height = 1.59e-3;
antC.GroundPlaneLength = 48e-3;
antC.GroundPlaneWidth = 57e-3;
antC.FeedOffset = [-0.5*antC.GroundPlaneLength -6e-3];

```

Mesh the antenna with a maximum edge length of 20e-3.

```

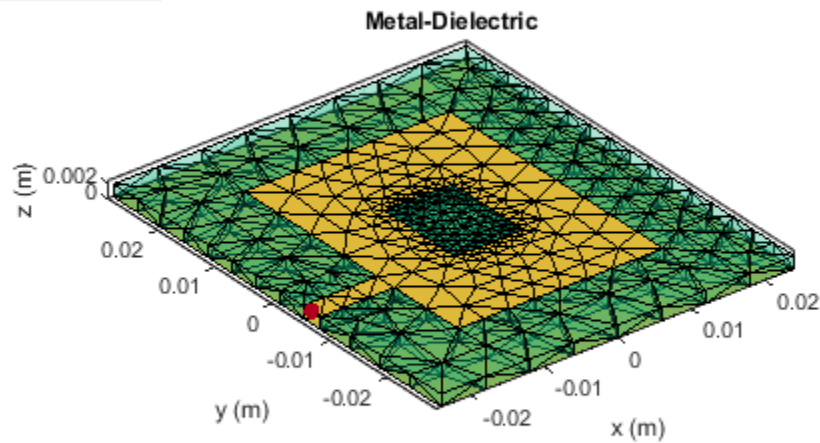
mesh(antC, 'MaxEdgeLength', 20e-3);

```

```

NumTriangles: 816
NumTetrahedra: 1665
NumBasis:
MaxEdgeLength: 0.02
MeshMode: manual

```



Calculate impedance, s-parameters over the frequency range of 1.5 GHz to 5 GHz.

```

freqRange = ((1.5:0.02:5)*1e9);
z1 = impedance(antC, freqRange);
s1 = sparameters(antC, freqRange);

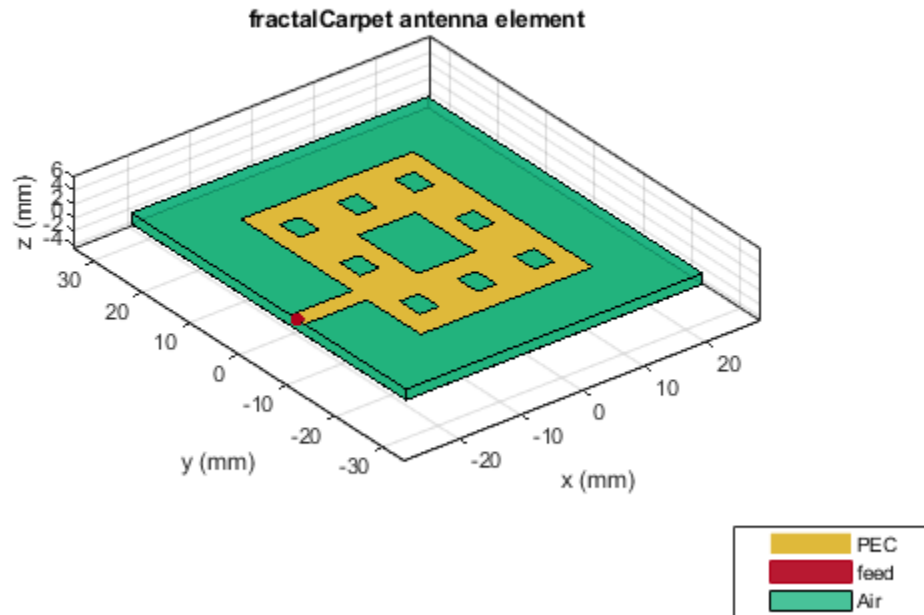
```

Change number of iterations to two.

```

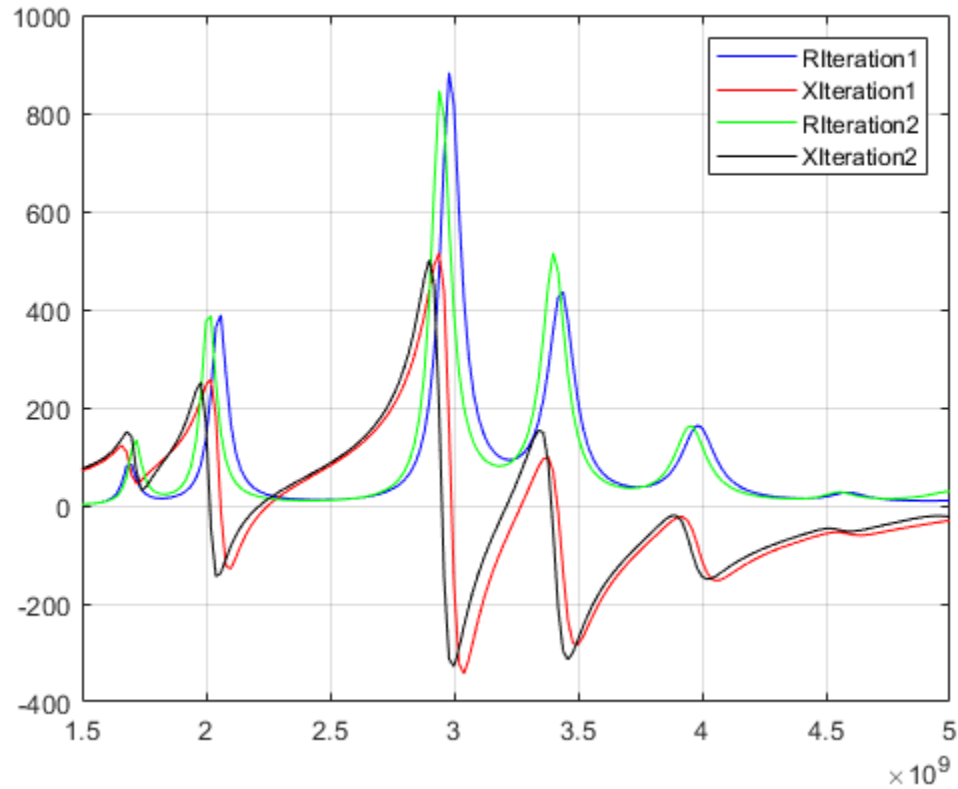
antC.NumIterations = 2;
figure; show(antC);

```

Calculate the impedance with second iteration and plot the impedance of first and second iterations.

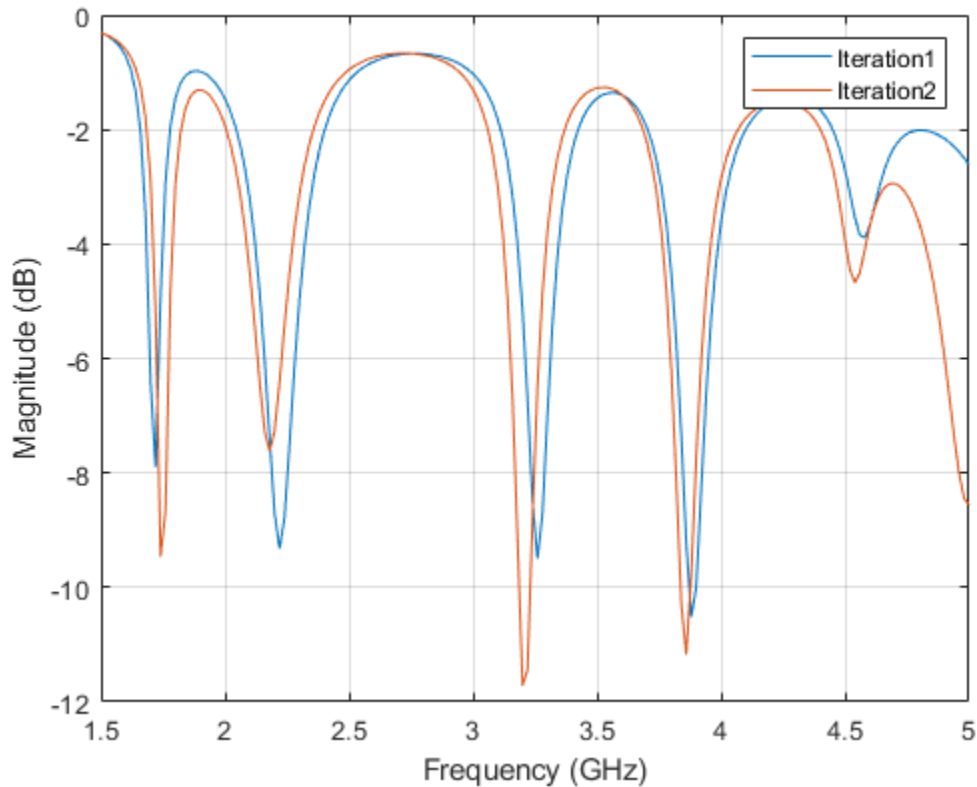
```
z2 = impedance(antC, freqRange);
figure; plot(freqRange, real(z1), 'b', freqRange, imag(z1), 'r'); grid on;
hold on; plot(freqRange, real(z2), '-g', freqRange, imag(z2), '-k');
legend('RIteration1', 'XIteration1', 'RIteration2', 'XIteration2');
```



The fractal carpet antenna is resonating at multiple frequencies. This is due to the self-similarity in the geometry of the antenna. The resonant frequencies for fractal carpet with number of iterations two are shifted left when compared to the number of iterations equal to one.

Calculate the s-parameters in second iteration and plot for first and second iterations.

```
s2 = sparameters(antC, freqRange);
figure; rfplot(s1);
hold on; rfplot(s2);
legend('Iteration1', 'Iteration2');
```



Tabulation of bandwidths and magnitudes of reflection coefficient with first and second iterations.

```
Iteration = [1;2;2];
FrequencyRange_GHz = {'3.84-3.92'; '3.18-3.24'; '3.8-3.88'};
ReflectionCoefficients_dB = [-14.22; -11.95; -15.64];
Bandwidth_MHz = [80;60;80];
table(Iteration,FrequencyRange_GHz,Bandwidth_MHz,ReflectionCoefficients_dB)
```

ans =

3x4 table

Iteration	FrequencyRange_GHz	Bandwidth_MHz	ReflectionCoefficients_dB
1	{'3.84-3.92'}	80	-14.22
2	{'3.18-3.24'}	60	-11.95
2	{'3.8-3.88' }	80	-15.64

With number of iterations one, the reflection coefficient is less than -10dB in one band. With number of iterations two, there are two bands of frequencies as mentioned in the table. The self-similarity property of fractals gives the multi band behavior. So, fractal antennas can be used in different bands of frequencies. The area of fractal carpet gets decreased by $(8/9)^n$ times the original area of square patch, with the increase in number of iterations(n). With $n = 1$, there is 11.1 percentage reduction in area and with $n = 2$, there is 21 percent reduction in area compared to the area with zeroth iteration.

Conclusion

Applying fractal geometries is one of the ways to miniaturize antennas. With the increase in number of iterations, the electrical length of dipole, perimeter of koch loop increases. This increase helps in fitting large electrical length into smaller volume. There is a decrease in the dimensions of the fractal antennas, compared to the conventional antennas. The self-similarity in the geometry of the fractal antennas, helps them operate in multi bands. Miniaturization and multi band characteristics can be observed with the other fractals also like fractal gasket, fractal snowflake, and fractal Island antennas.

References

- [1] Werner, D. H. and S. Ganguly, "An overview of fractal antenna engineering research", IEEE Antennas and Propagation Magazine, Vol. 45, No. 1, 38-57, 2003.
- [2] "Design and Implementation of Sierpinski Carpet Fractal Antenna for Wireless Communication. Rahul Batra, P.L.Zade, Dipika.

See Also

"Modeling Planar Photonic Band Gap Structure" on page 5-344

ISM Band Patch Microstrip Antennas and Mutually Coupled Patches

This example shows how to design and implement a rectangular, circular, triangular and elliptical patch microstrip antennas complying with the ISM (Industrial Scientific and Medical) band.

Define Parameters

All these microstrip antennas consisting of PCB with 6.6 mm thickness with dielectric constant EpsilonR 4.2, and loss-tangent of 0.02, and square ground plane of 100 mm x 100 mm, fed by 1.3 mm diameter coaxial probe, are designed to comply with the ISM band (2.4 - 2.5 GHz).

```
LGp = 100e-3;      % Ground plane length
WGp = 100e-3;      % Ground plane width
h = 6.6e-3;        % Height of the substrate
```

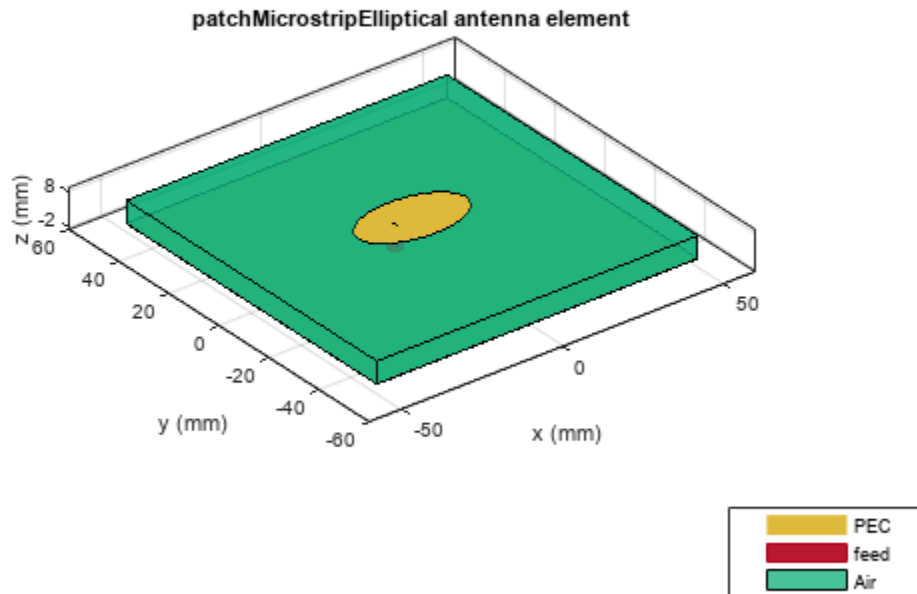
Define Parameters Of Elliptical Patch Antenna

Design a probe feed `patchMicrostripElliptical` antenna using a dimension of 33.5 mm major axis, 18.8 mm minor axis. The feed is offset by 11.6 mm from the origin along the X-axis.

```
a = 33.5e-3;      % Major Axis
b = 18.8e-3;      % Minor Axis
f = 11.6e-3;      % Feed Offset
d = dielectric('EpsilonR',4.2,'LossTangent',0.02);
```

Create a `patchMicrostripElliptical` antenna using the defined parameters.

```
p_Ellipse= patchMicrostripElliptical('MajorAxis',a,'MinorAxis',b,...
    'Height',h,'Substrate',d,'GroundPlaneLength',LGp,'GroundPlaneWidth',...
    WGp,'FeedOffset',[-(a/2-f) 0]);
figure;
show(p_Ellipse);
```



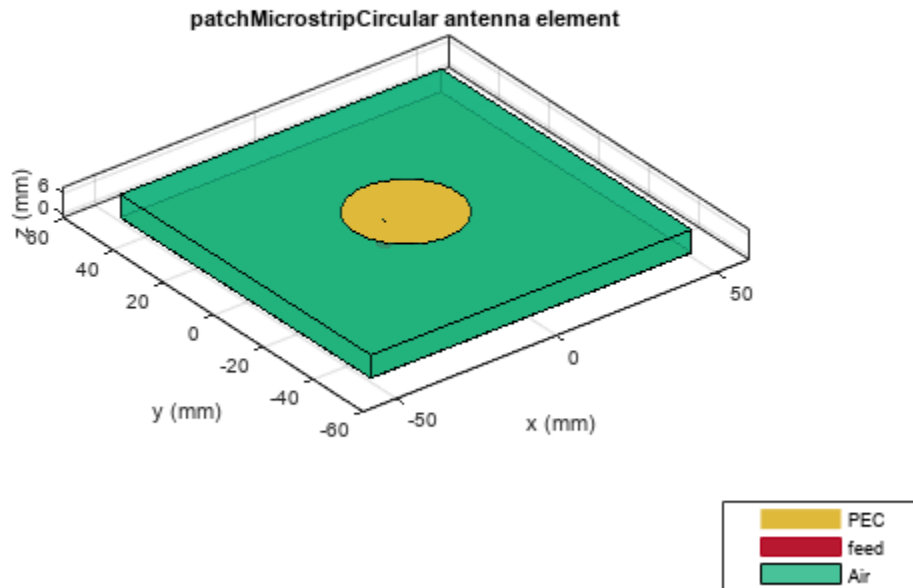
Define Parameters Of Circular Patch Antenna

Design a probe feed patchMicrostripCircular antenna using a dimension of 16 mm Radius. The feed is offset by 9.25 mm from the origin along the X-axis.

```
r = 16e-3;           % Radius
f1 = 9.25e-3;       % Feed Offset
```

Create a patchMicrostripCircular antenna using the defined parameters.

```
p_Circle= patchMicrostripCircular('Radius',r,'Height',h,'Substrate',d,...
    'GroundPlaneLength',LGp,'GroundPlaneWidth',WGp,'FeedOffset',[-(r-f1) 0]);
figure;
show(p_Circle);
```



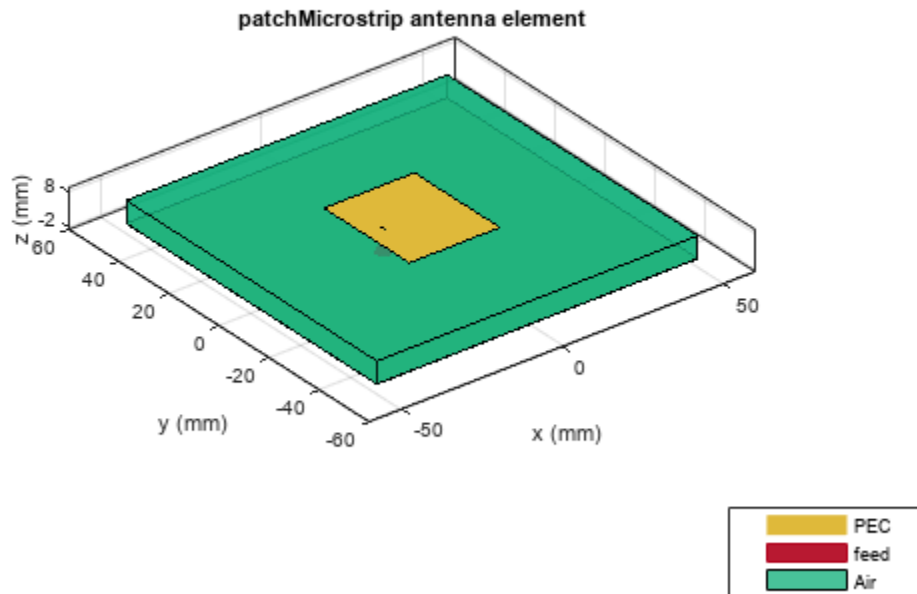
Define Parameters Of Rectangular Patch Antenna

Design a probe-fed rectangular `patchMicrostrip` antenna using a dimension of 28.20 mm length, 34.06 mm width. The feed is offset by 5.3 mm from the origin along the X-axis.

```
rect1 = 28.20e-3; % length
rect2 = 34.06e-3; % width
f2 = 5.3e-3;     % feedoffset
```

Create a `patchMicrostrip` rectangular antenna using the defined parameters.

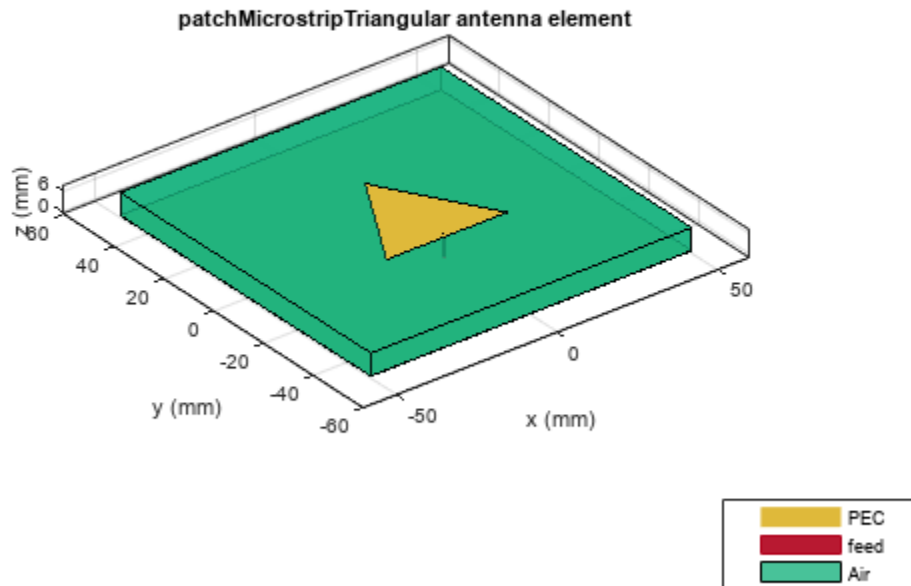
```
p_rect= patchMicrostrip('Length',rect1,'Width',rect2,'Height',h,'Substrate',d,...
    'GroundPlaneLength',LGp,'GroundPlaneWidth',WGp,...
    'FeedOffset',[-(rect1/2-f2) 0]);
figure;
show(p_rect);
```



Define Parameters of Triangular Patch Antenna

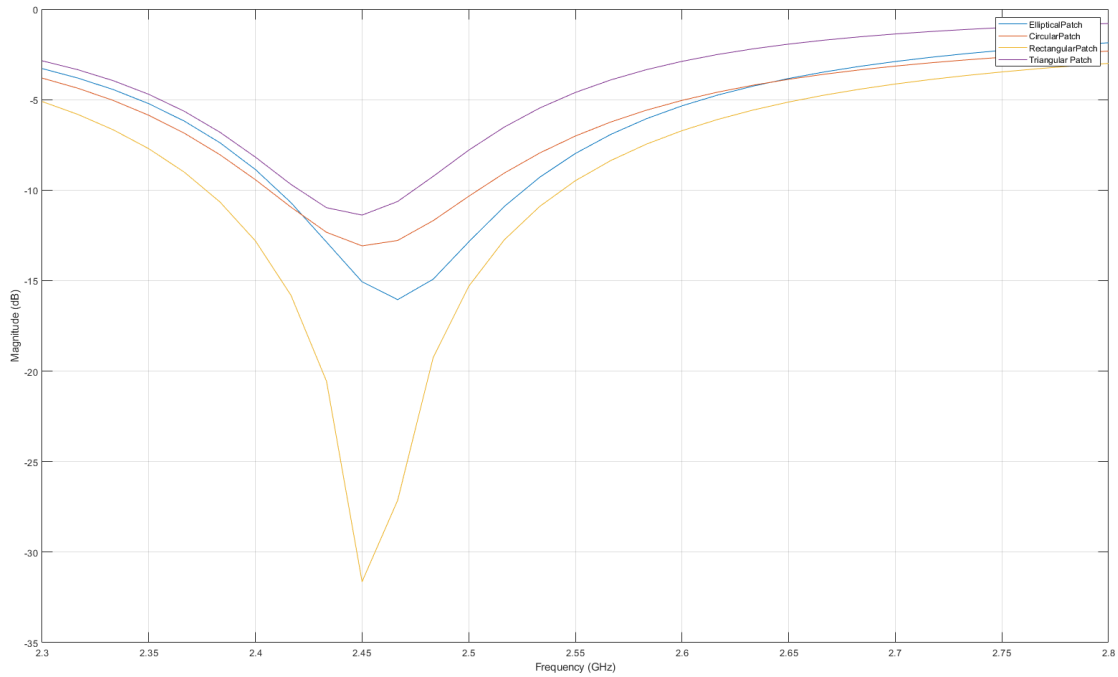
Design an equilateral patchMicrostripTriangular antenna using a dimension of 37.63 mm side. The feed is offset by 3.8 mm from the origin along the Y-axis.

```
side = 37.63e-3;
f_off = 3.8e-3;
p_triangular = patchMicrostripTriangular('side',side,'Height',h,'Substrate',d,...
    'GroundPlaneLength',LGp,'GroundPlaneWidth',WGp,...
    'FeedOffset',[0 -side/2+f_off]);
figure;
show(p_triangular);
```

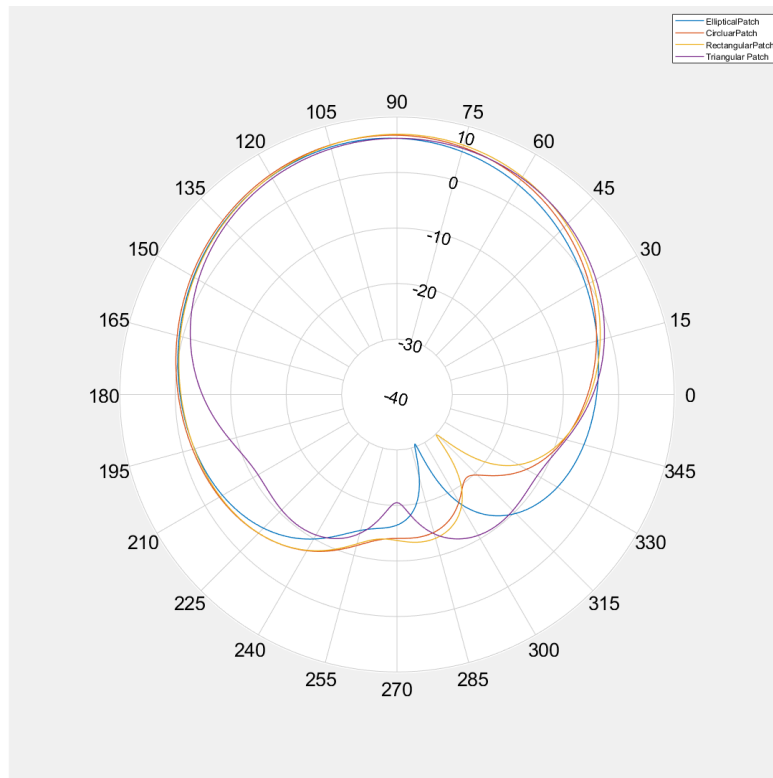
Visualize Reflection Coefficient Magnitude of Patches

Plot the reflection coefficient for these antennas over the band and a reference impedance of 50 ohms. Curves for the reflection coefficient magnitude are shown in the below figure. Manually mesh of patch antennas with different edge lengths.



Visualize Radiation Pattern

The directivity of the antennas are around 6.37 dB for Elliptical patch, 7 dB for circular patch, 7.37 dB for rectangular patch and 6.16 dB for triangular patch.



Mutually Coupling Between Rectangular and Triangular Patches

This section is devoted to the study of cases, focusing on configurations with the lowest mutual couplings. The relative displacement 'd' is fixed as $\lambda/2$, two patches placed side-by-side configuration and the patches positioned in the center of the rectangular 160 mm x 100 mm ground plane with a substrate of FR4.

```

LGp1 = 160e-3;      % Ground plane length
WGp1 = 100e-3;     % Ground plane width
Ground_plane1=antenna.Rectangle('Length',LGp1,'Width',WGp1);% Ground plane
dis = 0.0612;      % distance between two patches (d = lambda/2)

```

Create a patchMicrostrip rectangular antenna using the defined parameters.

```

r_ant = pcbStack(p_rect);
rect_p = r_ant.Layers{1};
rect_p.Center = [-dis/2 0];

```

Create a patchMicrostripTriangular antenna using the defined parameters.

```

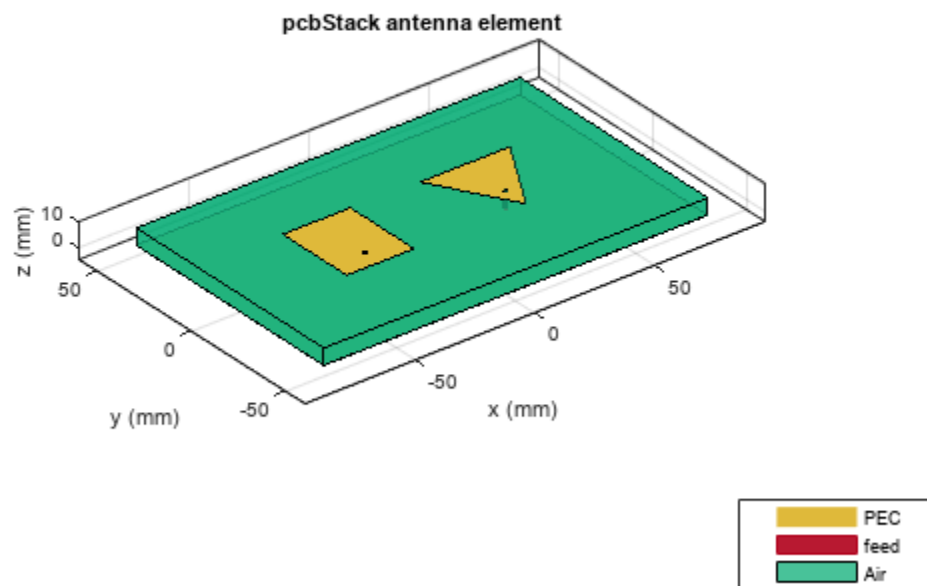
t_ant = pcbStack(p_triangular);
triangle_p = t_ant.Layers{1};
triangle_p= rotateZ(triangle_p,180);
triangle_p= translate(triangle_p,[dis/2, 0, 0]);
patch = rect_p+triangle_p; % adding patches

```

Define PCB Stack

Use the `pcbStack` to define the metal and dielectric layers for mutually coupled patch antenna. the top-most layer is a patch layer, the second layer is dielectric layer, and the third layer is the ground plane.

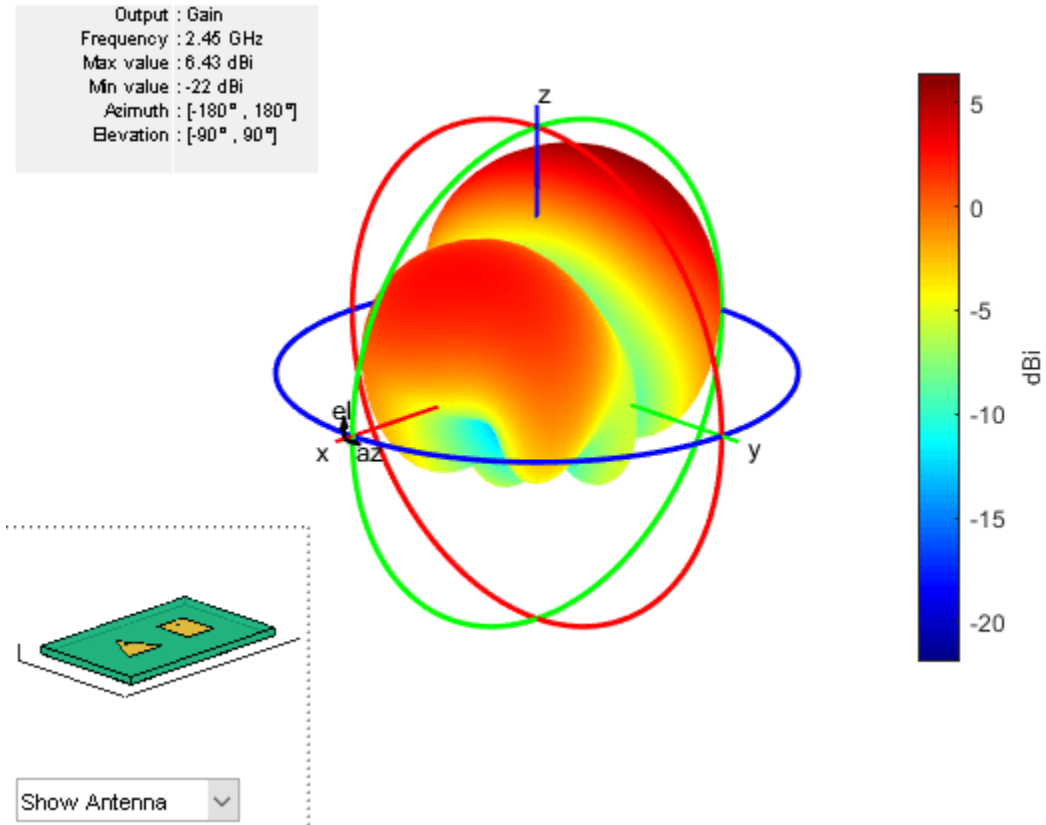
```
p_mc=pcbStack;
d4=dielectric('EpsilonR',4.2,'Thickness',h,'LossTangent',0.02);
p_mc.BoardThickness=d4.Thickness;
p_mc.BoardShape.Length=LGp1;
p_mc.BoardShape.Width=WGp1;
p_mc.Layers={patch,d4,Ground_plane1};
p_mc.FeedLocations=[-dis/2 -(rect1/2-f2) 1 3; dis/2 -5.3075e-3 1 3];
p_mc.FeedDiameter=1.3e-3;
figure;
show(p_mc);
```



Radiation Pattern of Mutually Coupled Patches

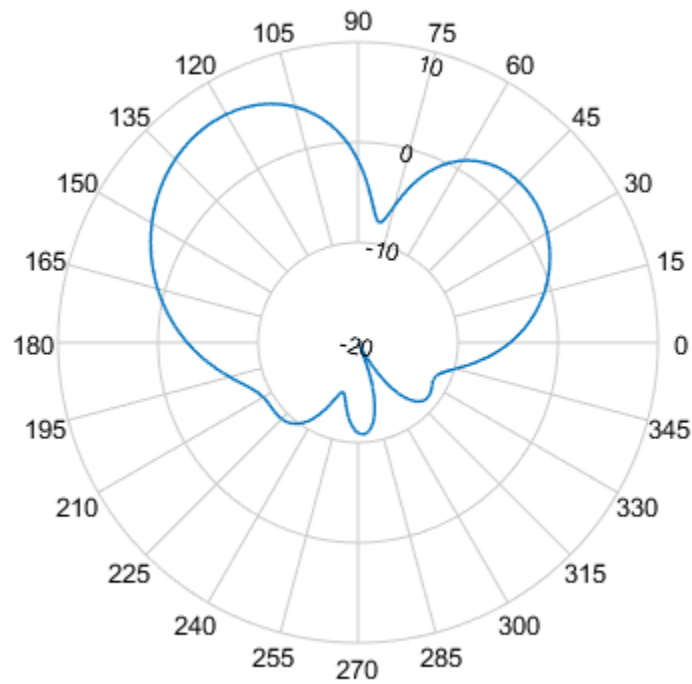
The side-by-side configuration directivity is 7.4 dB

```
figure;
pattern(p_mc,2.45e9);
```



Pattern Magnitude

```
figure;
[fm,~,t10] = pattern(p_mc,2.45e9,0,0:360);
polarpattern(t10,fm);
```



Reference

- 1) <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7225105>
- 2) Probe - Fed Linearly - Polarized Electrically Equivalent Microstrip Antennas on FR4 Substrates

See Also

“Multiband Nature and Miniaturization of Fractal Antennas” on page 5-683 | “Modeling and Analysis of Probe-Fed Stacked Patch Antenna” on page 5-395

Design and Analyze Curved Reflectors

This example shows how to design and analyze the curved reflectors. Curved reflectors are an upgraded version of rectangular reflectors. The curved surface on these reflectors reflects and focuses waves, which provides a stronger signal. Some of the notable characteristics of the curved reflectors are high gain, low cross-polarization, and reasonable bandwidth.

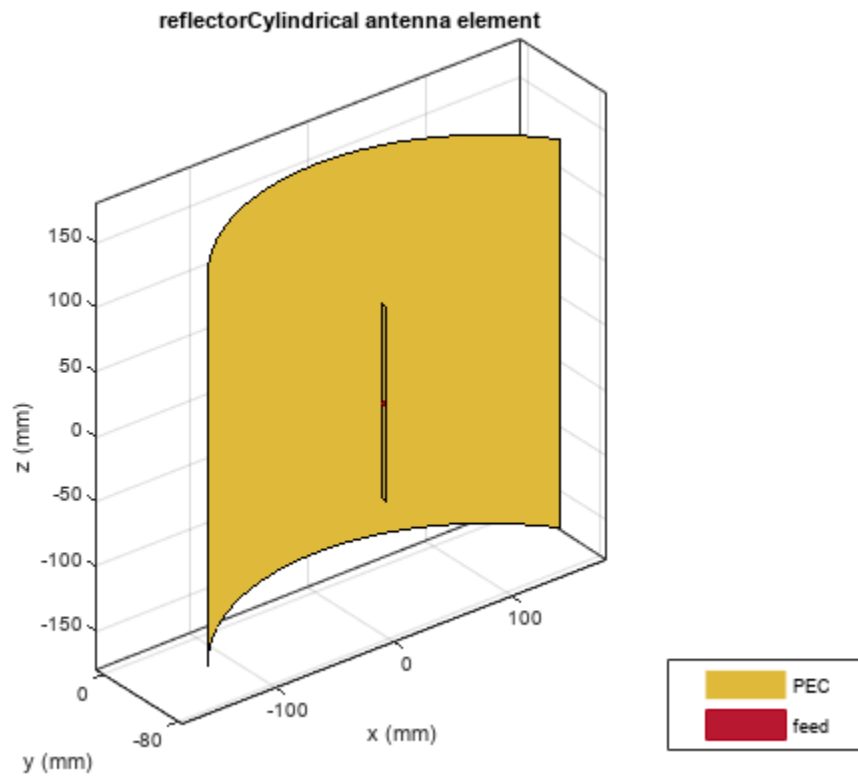
In this example, you will design and analyze two types of curved reflectors, the cylindrical and the spherical. Then, you will compare their radiation patterns with other types of backing structures.

Design and Analyze Cylindrical Reflector

Cylindrical reflector uses a cylindrical shaped surface as a reflector. This reflector coexist with an antenna in various wireless transmission applications such as radar, general satellite communications, and in astronomical applications. The function of the reflector is to focus a beam of signal towards a specific direction for an enhanced antenna gain. The following properties of cylindrical reflector object, `reflectorCylindrical` are used in this example to define the cylindrical reflector and an exciter system.

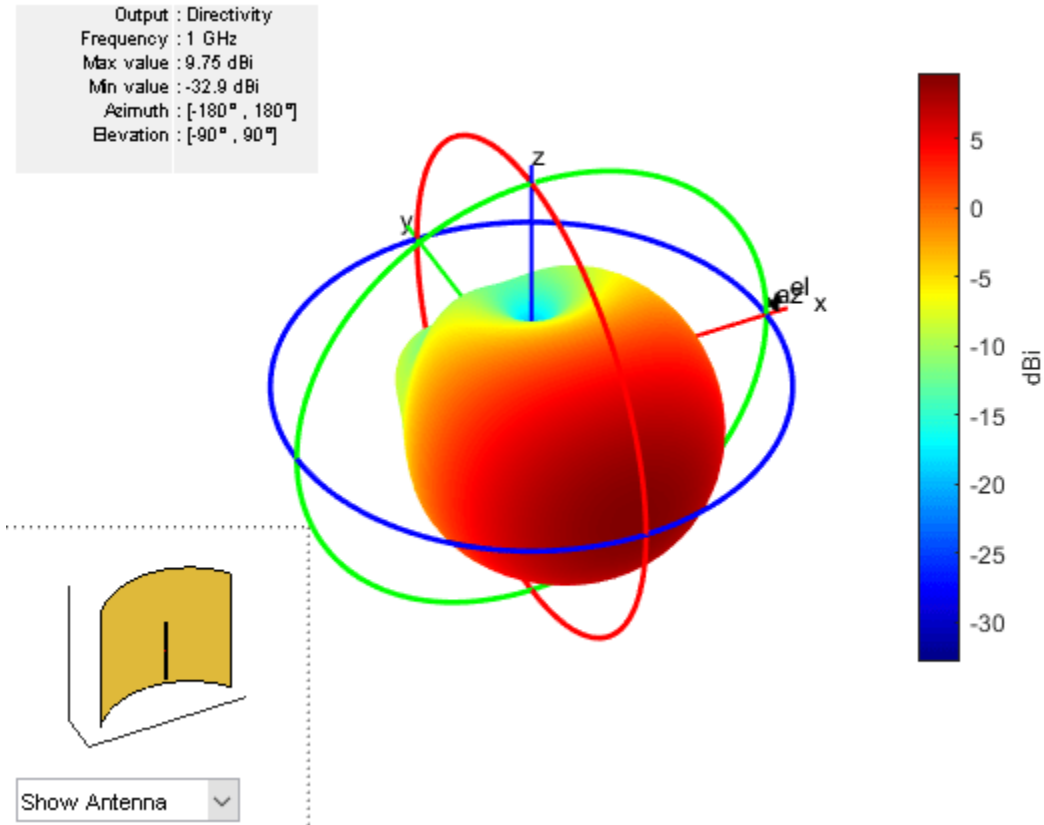
- `GroundPlaneLength` — Length of the ground plane in meters
- `GroundPlaneWidth` — Width of the ground plane in meters
- `Depth` — Perpendicular distance from origin to cylindrical reflector aperture in meters

```
GL = 0.3;  
GW = 0.3;  
D = 0.075;  
cref = reflectorCylindrical('GroundPlaneLength',GL,'GroundPlaneWidth',GW,'Depth',D);  
cref.Tilt = 90;  
cref.Exciter.TiltAxis = [1 0 0];  
cref.Exciter.Tilt = 90;  
show(cref)
```



Plot the radiation pattern of the cylindrical reflector at 1 GHz.

```
pattern(cref,1e9)  
view(-26,39)
```

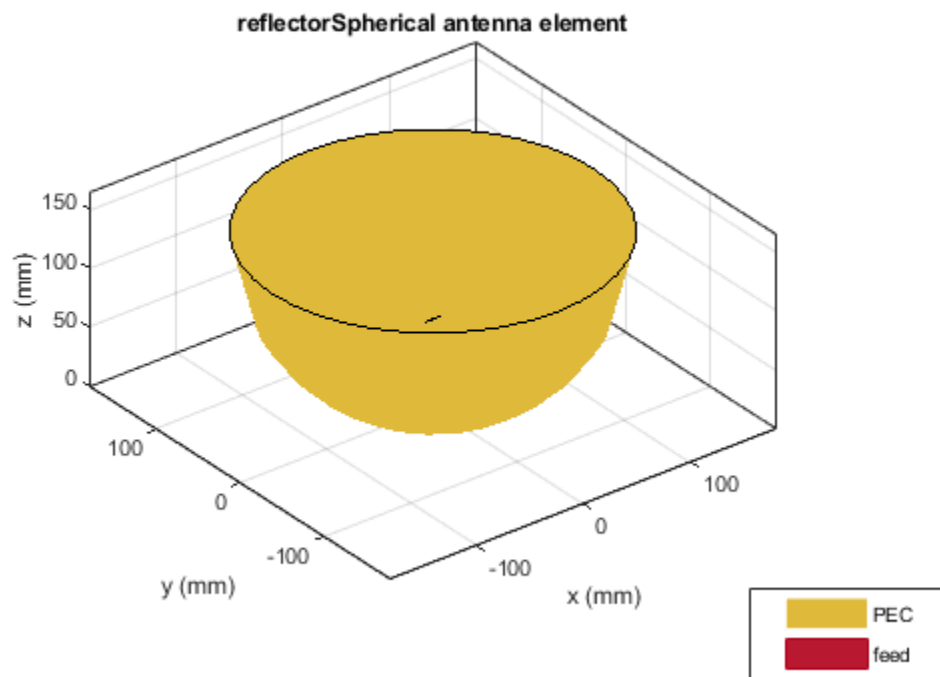



Design and Analyze Spherical Reflector

This type reflector are created by taking a cross section from the sphere and suitable for wide-angle scanning. The spherical reflector can provide beam scanning from a fixed reflecting surface without any distortion. This reduces scanning system cost in compared to a conventional reflector antenna systems. The cost reduction comes in eliminating the need to move the primary reflecting surface at all elevation angles. The following properties of the spherical reflector object, `reflectorSpherical` are used in this example to define spherical reflector and an exciter system.

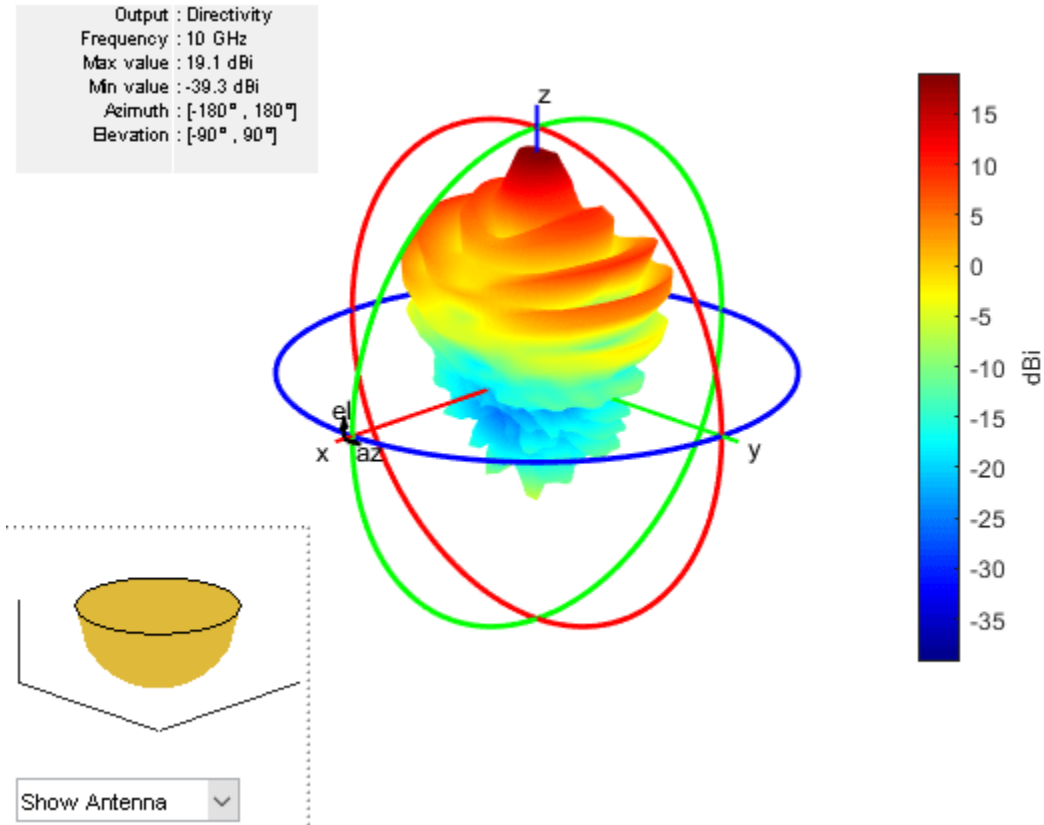
- Radius — Radius of the aperture in meters
- Depth — Perpendicular distance between origin and the aperture of the antenna in meters

```
R = 0.15;
D = 0.15;
sref = reflectorSpherical('Radius',R,'Depth',D);
show(sref)
```



Plot the radiation pattern of the spherical reflector backed dipole.

```
pattern(sref,10e9)
```

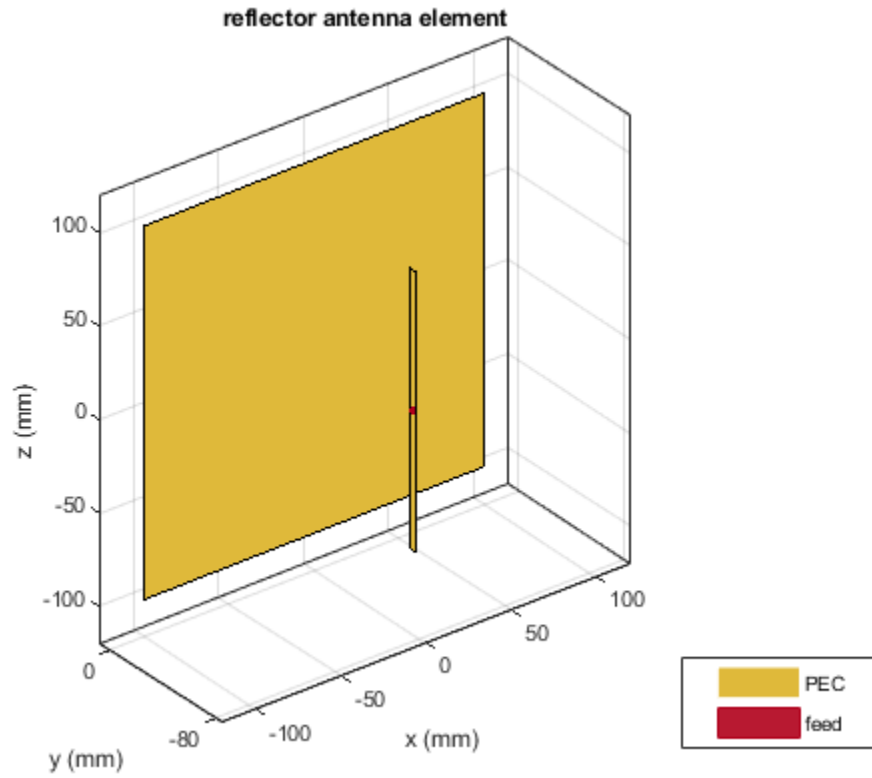


Azimuth Plots of Rectangular and Cylindrical Reflectors

Plot the azimuth patterns of rectangular and cylindrical reflectors to compare the radiation characteristics of the two reflectors.

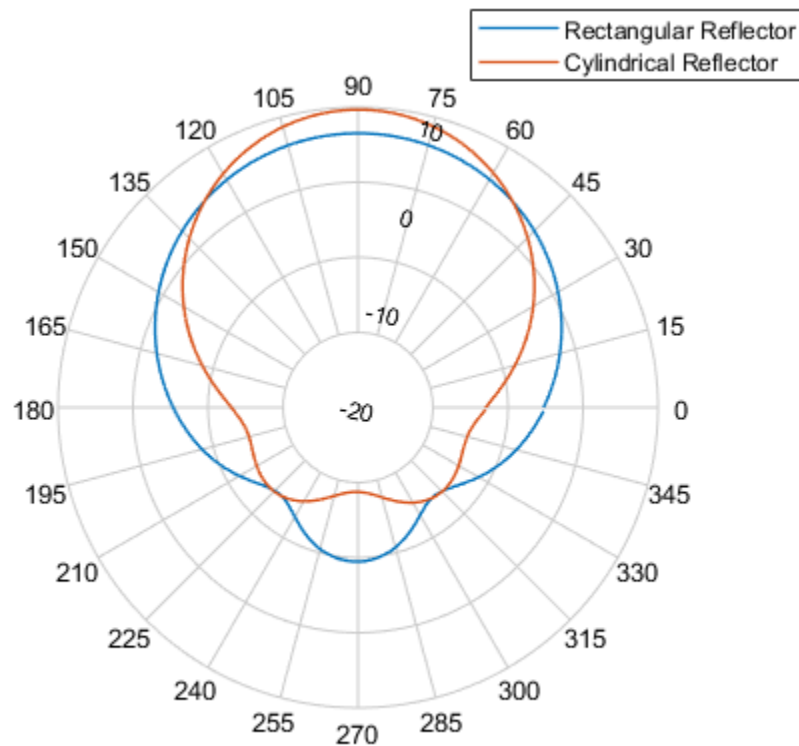
Create the rectangular reflector antenna object.

```
antR = reflector;  
antR.Tilt = 90;  
antR.Exciter.TiltAxis = [1 0 0];  
antR.Exciter.Tilt = 90;  
show(antR)
```



Plot the azimuth pattern plots for the rectangular and the cylindrical reflectors and compare their results.

```
pa1 = patternAzimuth(anrR,1e9,0,"Azimuth",0:-1:-360);
pa2 = patternAzimuth(crf,1e9,0,"Azimuth",0:-1:-360);
figure;
polarpattern(pa1);
hold on;
polarpattern(pa2);
hold off;
l = legend('Rectangular Reflector','Cylindrical Reflector');
l.Position = [0.6 0.8877 0.2996 0.0869];
```



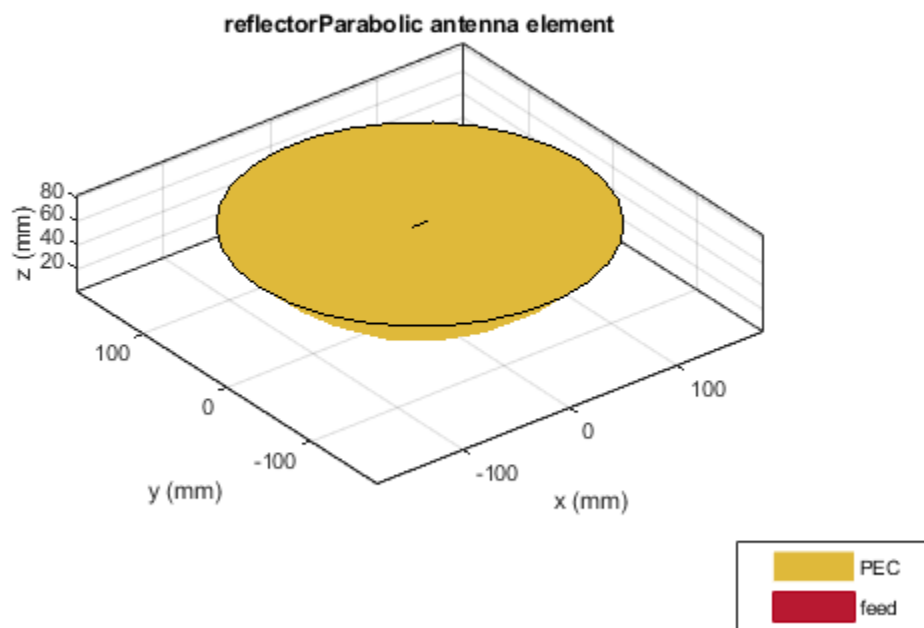
Cylindrical reflectors, on account of their curvature, provide higher gain compared to the rectangular reflectors.

Elevation Plots of Spherical and Parabolic Reflectors

Plot the elevation patterns of spherical and parabolic reflectors to compare the radiation characteristics of the two reflectors.

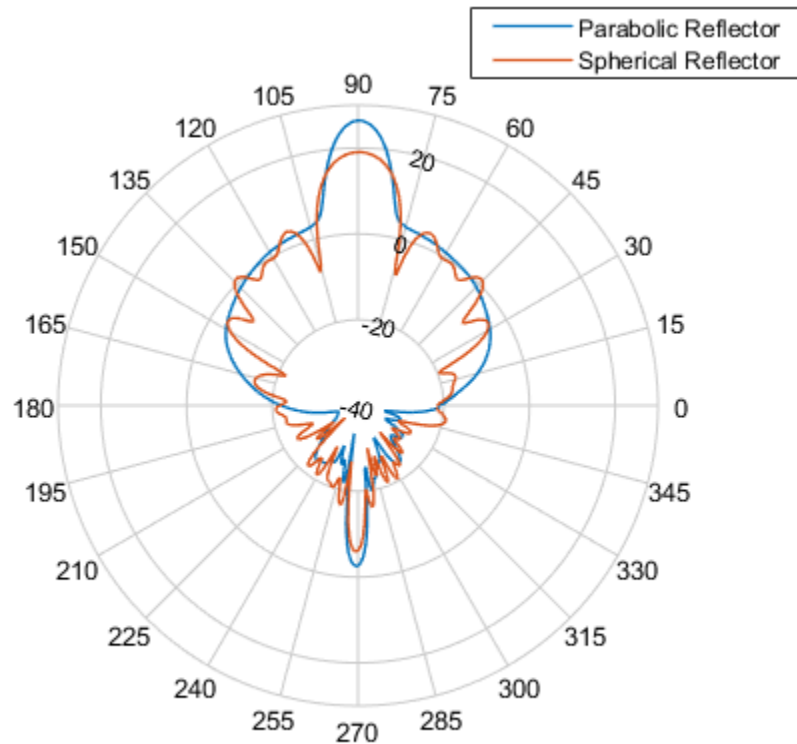
Create a parabolic reflector.

```
pref = reflectorParabolic;
show(pref)
```



Compare the elevation pattern plots of the parabolic and the spherical reflectors.

```
pe1 = patternElevation(pref,10e9,0,'Elevation',0:360);  
pe2 = patternElevation(sref,10e9,0,'Elevation',0:360);  
figure  
polarpattern(pe1)  
hold on;  
polarpattern(pe2)  
l = legend('Parabolic Reflector','Spherical Reflector');  
l.Position = [0.6 0.8877 0.2996 0.0869];
```



The elevation pattern plot of the spherical reflector shows more peaks compared to the parabolic reflector. This means that a spherical reflector scans a wider area.

Reference

[1] Balanis, Constantine A. *Antenna Theory: Analysis and Design*. 3rd ed. Hoboken, NJ: John Wiley, 2005.

[2] Zali H.M., M.T.Ali, I.Pasya, N.A.Halili, H.Ja'afar, M.Hilmi. "Design of a Cylindrical Parabolic Reflector on Monopole Plasma Antenna". *IEEE International RF and Microwave Conference (RFM)*, Penang, 2013, pp. 344-348.

See Also

"Design and Analyze Cassegrain Antenna" on page 5-585

VHF/UHF Biconical Antenna for Testing Applications

This example shows how to design and analyze a wire biconical antenna that can be used in various testing applications. Wire biconical antennas are used as a standard victim in any compliance test lab in a variety of testing applications such as immunity testing, emission testing, spectrum monitoring, and shielding effectiveness etc.

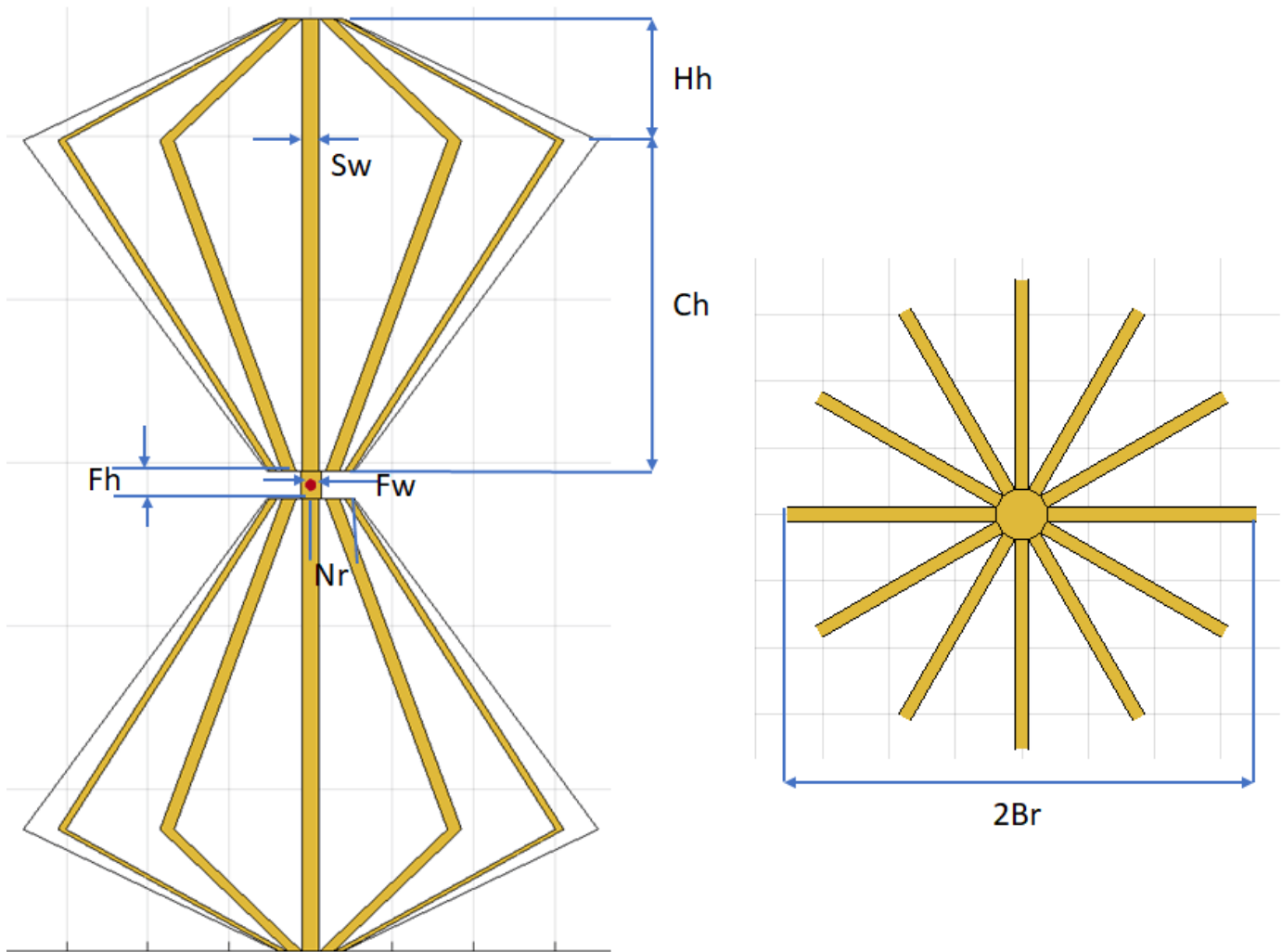
- **Immunity Testing:** Radiated immunity testing requires the device under test (DUT) to operate when illuminated with high fields of electromagnetic energy. Wire biconical antennas are capable of generating those high fields at a lower range from 20MHz to 80MHz and in a higher range from 80MHz to 300MHz. A larger wire biconical antenna can produce higher fields.
- **Emission Testing:** Radiation emission testing is the most common type of compliance testing. It can measure the field strength of the DUT generally in 30MHz to 1GHz. Wire biconical antennas are an excellent choice for portable applications.
- **Shielding Effectiveness:** Wire biconical antenna have a coaxial wound balun that can handle high radio frequency fields.
- **Spectrum Monitoring:** Radio surveillance covers a broad operating range. Wire biconical antennas are the ideal solution for these applications because of their broad bandwidth and omnidirectional radiation pattern.

Half-wave dipole antennas were used in such applications earlier. However, these antennas require more time in testing as the length of the antenna has to be adjusted for each frequency of interest. Whereas, wire biconical antennas are advantageous in that respect because of their broad bandwidth and omnidirectional pattern.

This example designs a wire biconical antenna with stable impedance over a wide frequency band of 300MHz to 1GHz. This frequency band has flat gain characteristics, which is omnidirectional in the H-plane and bidirectional in the E-plane like the half-wave dipole antenna.

Define Parameters

The dimensions shown in this figure are used to create a wire biconical antenna to cover the frequency range 300MHz to 1GHz, which is the UHF range for EMI and EMC testing application.



$N = 12;$
 $Sw = 12e-3;$
 $Hh = 150e-3;$
 $Ch = 405.2e-3;$
 $Nr = 35e-3;$
 $Br = 353.2e-3;$
 $Fh = 25e-3;$
 $Fw = 18e-3;$

Create Wire Biconical Antenna

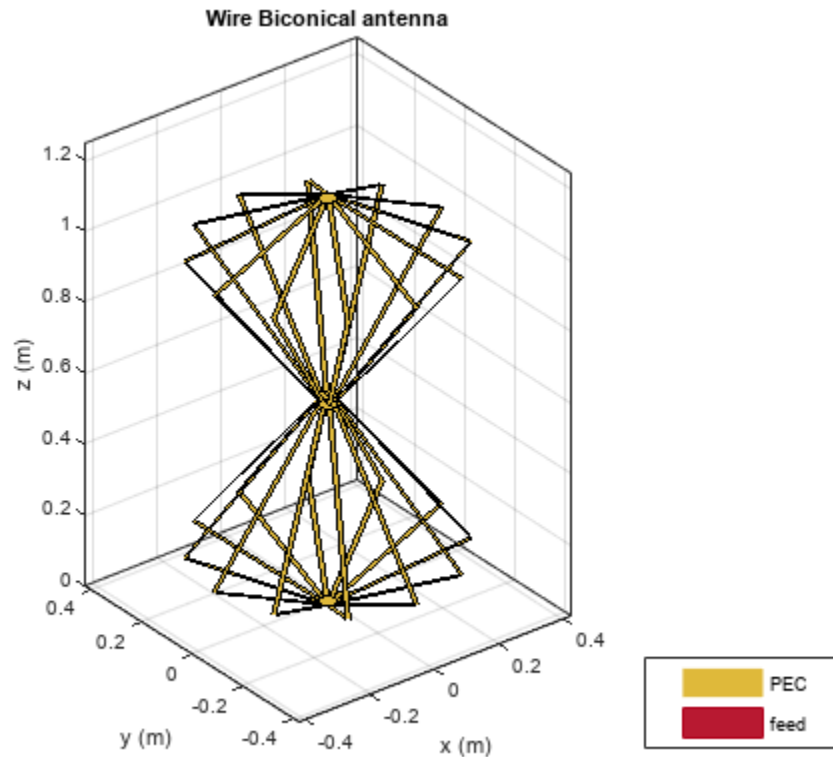
Create a wire biconical antenna with the defined parameters.

```

ant = biconeStrip;
ant.NumStrips      = N;
ant.StripWidth     = Sw;
ant.HatHeight      = Hh;
ant.ConeHeight     = Ch;
ant.NarrowRadius   = Nr;
ant.BroadRadius    = Br;
ant.FeedHeight     = Fh;

```

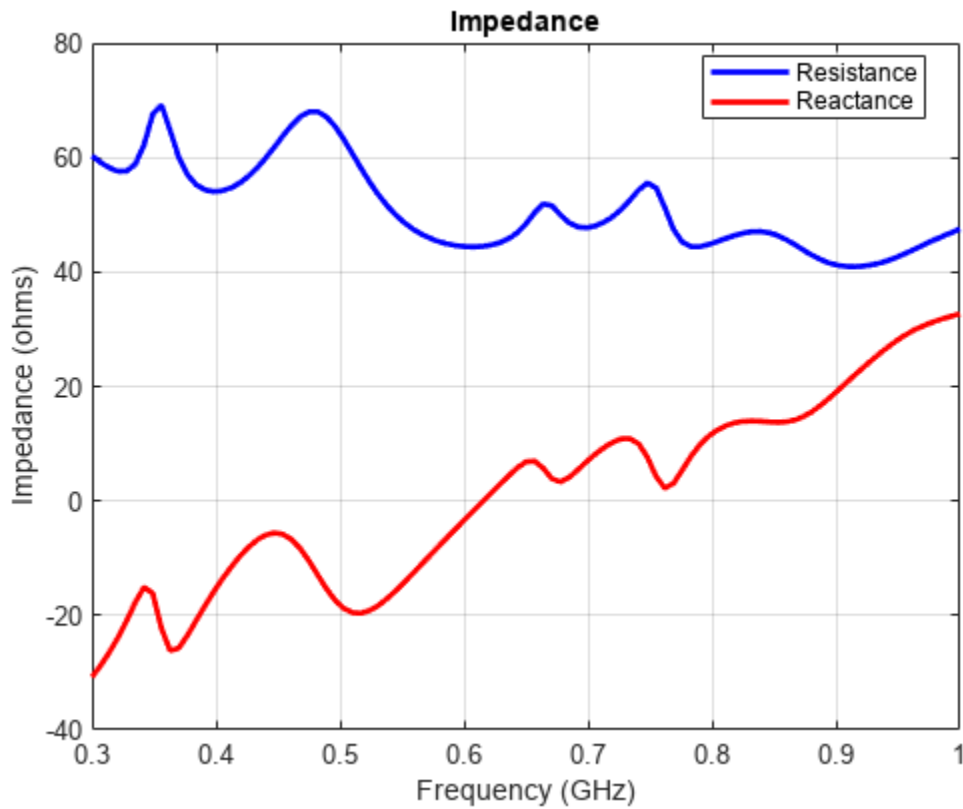
```
ant.FeedWidth = Fw;  
figure;  
show(ant);  
title('Wire Biconical antenna')
```



Impedance

The wire biconical antenna operates like the dipole antenna except that its skeleton has a conical-shaped element. This antenna exhibits a stable impedance for the wide frequency band ranging from 300MHz to 1000MHz. A balanced feed system should be used between the antenna and transmitter and receiver system.

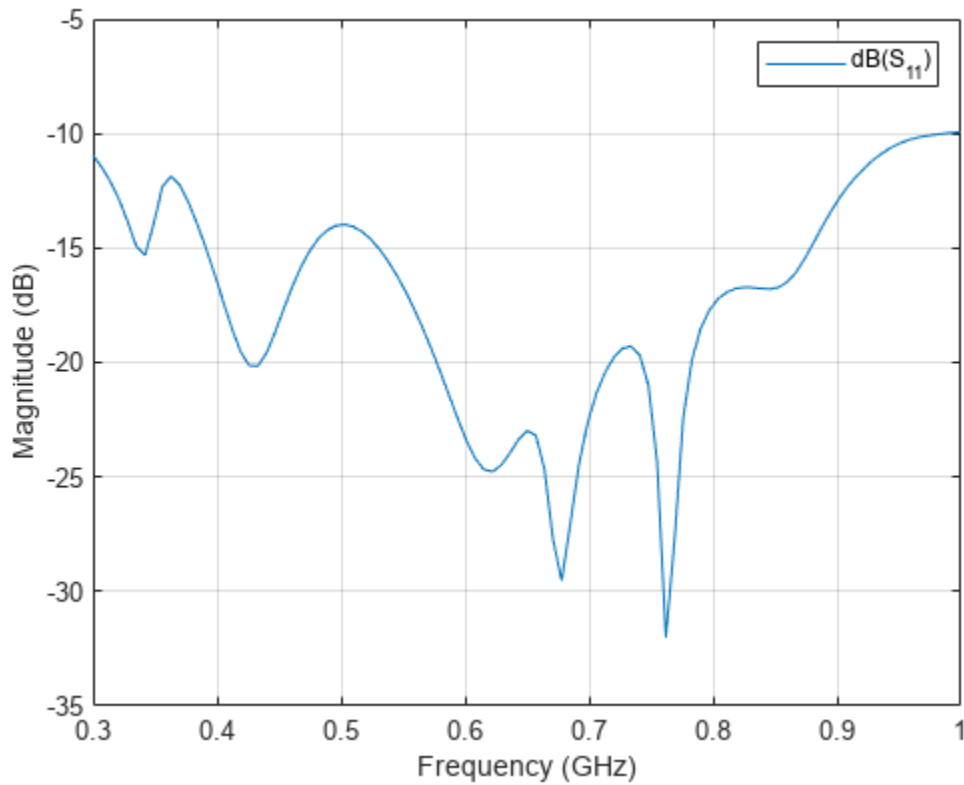
```
freq = linspace(300e6,1e9,101);  
figure;  
impedance(ant,freq);
```



Reflection Coefficient

The designed wire biconical antenna provides a reflection coefficient of less than -10dB for the desired operational frequency, without the matching network or balun. Thus, the designed antenna is preferred in EMC applications. Broadband coverage and compact size are prime advantages of this antenna. We choose the frequency range of 0.3 GHz to 1 GHz and calculate the S-parameters of the antenna.

```
s11 = sparameters(ant, freq);  
figure;  
rfplot(s11);
```

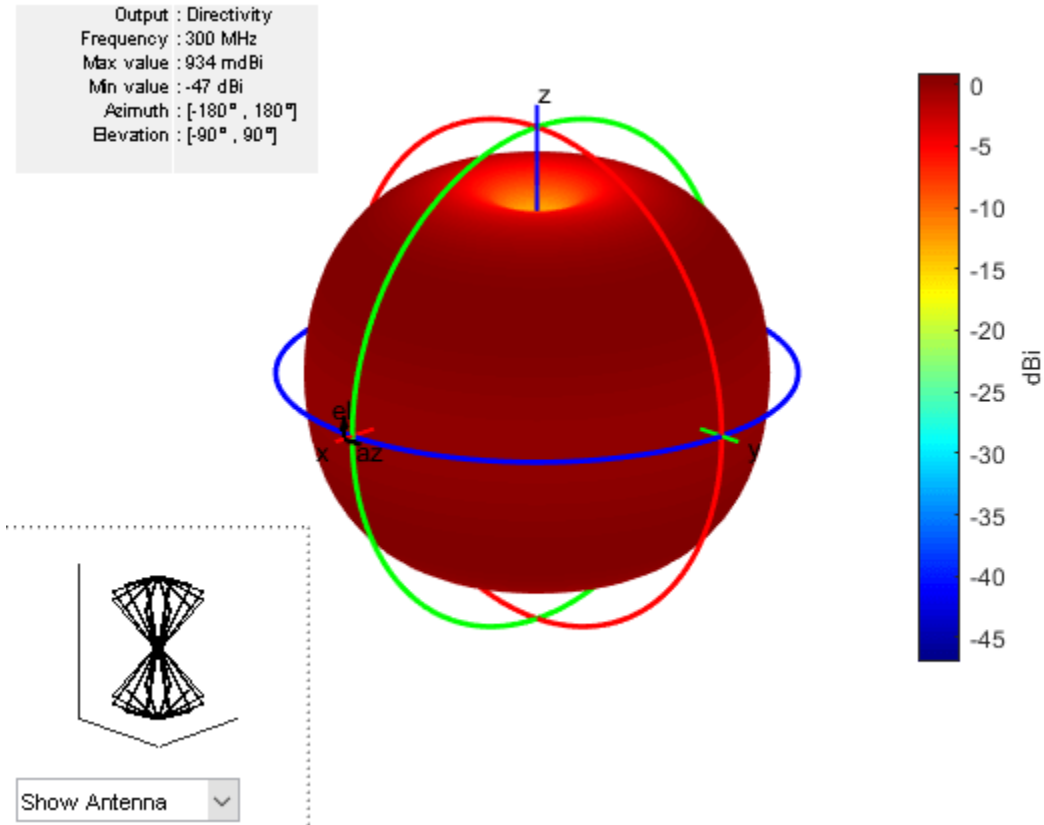


Radiation Pattern

The radiation pattern of a wire biconical antenna is similar that of a half-wave dipole antenna. The wire biconical antenna displays an omnidirectional pattern (circular shape) in the H-plane and a bidirectional (eight-shaped) pattern in the E-plane, as shown in this figure. The antenna is used in field surveillance and spectrum monitoring applications because of its H-plane beam width.

Plot the 3-D radiation pattern of the antenna at 300MHz.

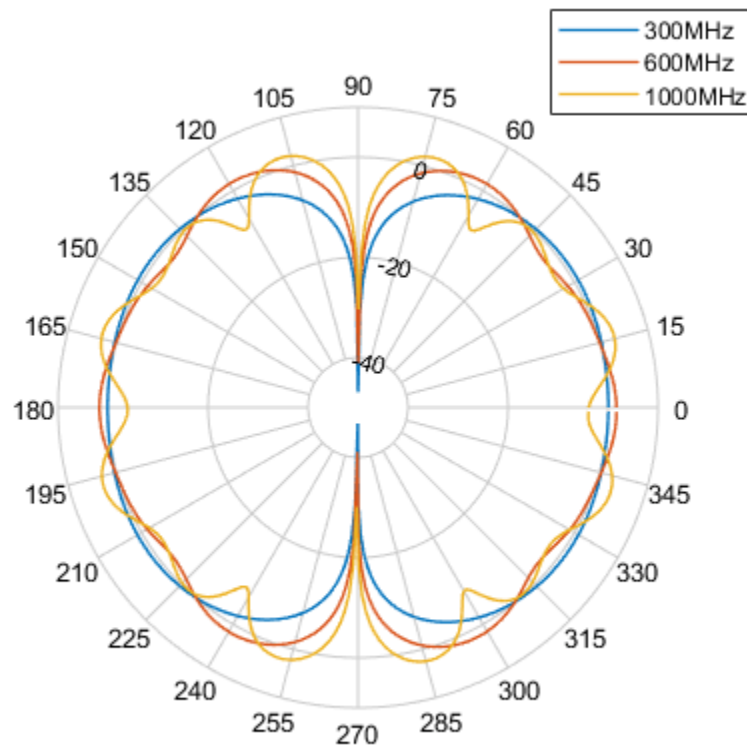
```
f = 300e6;  
figure;  
pattern(ant,f);
```



The wire biconical antenna shows the omnidirectional pattern for the frequencies ranging from 300MHz to 1000MHz with the gain varying between 2.04dB to 3.283dB.

Plot the elevation pattern of the antenna at different frequencies.

```
p1 = patternElevation(ant,300e6);
p2 = patternElevation(ant,600e6);
p3 = patternElevation(ant,1000e6);
figure; polarpattern(p1);
hold on; polarpattern(p2);
hold on; polarpattern(p3);
legend 300MHz 600MHz 1000MHz;
```



Conclusion

The wire biconical antenna plays a major role in testing applications due to its advantages over other antennas. Its chief advantages are compact size and broadband characteristics along with omnidirectional radiation pattern. Regardless of the type of test (standard compliance test or simple field monitoring test), the antenna displays efficient performance characteristics.

Reference

"SCHWARZBECK MESS- ELEKTRONIK", [Online]. <https://www.yumpu.com/en/document/read/35274071/schwarzbeck-mess-elektronik>.

See Also

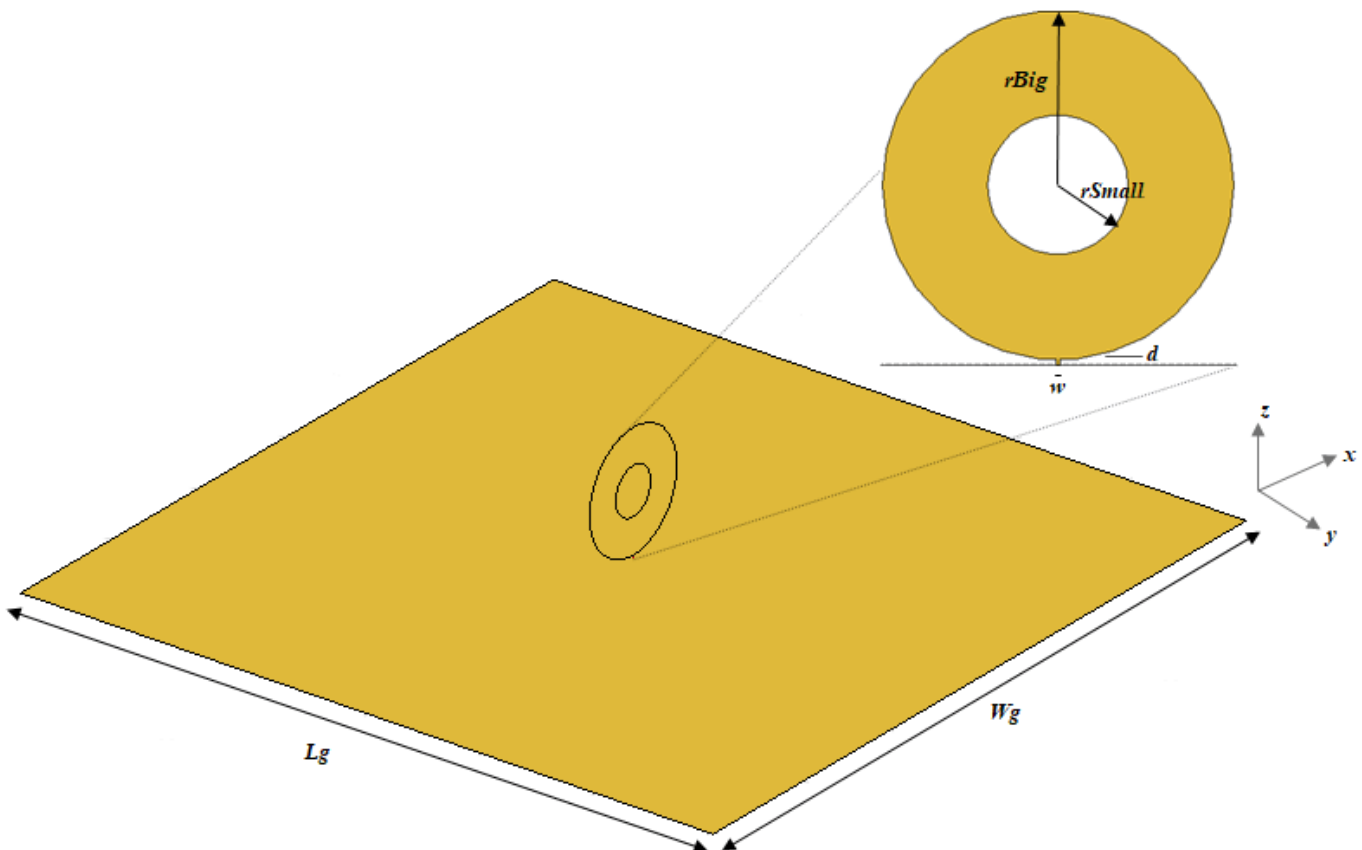
"Wideband Blade Dipole Antenna and Array" on page 5-243 | "Design Internally Matched Ultra-Wideband Vivaldi Antenna" on page 5-410

Ultra-Wideband (UWB) Planar Monopole Antennas

This example shows how to design UWB planar monopole antennas. Planar monopole antennas are simple in geometry and provide an ultrawideband (UWB) operating bandwidth. To create a planar monopole, replace a conventional wire monopole with planar elements in a variety of shapes to increase the surface area of the monopole. To design a planar monopole antenna, mount a planar metal plate on the ground plane. The ground plane can also have various shapes.

Annular Planar Monopole Antenna

This figure shows the geometry and dimensions of an annular planar monopole antenna. The radiating element is an annular ring, with an outer radius (r_{Big}) of 25 mm and an inner radius (r_{Small}) of 10 mm, located vertically above a square ground plane with a side length of 305 mm. The feedgap (d) between the feedpoint in the radiator and the ground plane and the width of the feedstrip (w) are set to 0.8 mm and 0.6 mm, respectively, to enhance the impedance bandwidth. The offset value is set to 0.2 mm to make a perfect connection between the feedstrip and the annular ring radiator.



Create Ground Plane

Create the square ground plane with a side length of 305 mm using the antenna.Rectangle function.

```
Lg = 305e-3; Wg = 305e-3;
groundPlane = antenna.Rectangle("Length",Lg,"Width",Wg);
```

Create Radiator

Create the feedstrip using the `antenna.Rectangle` function. To create an outer and inner circle, use the `antenna.Circle` function. To create the annular ring radiator, subtract the `innerCircle` from the `outerCircle` and add in the `FeedStrip`.

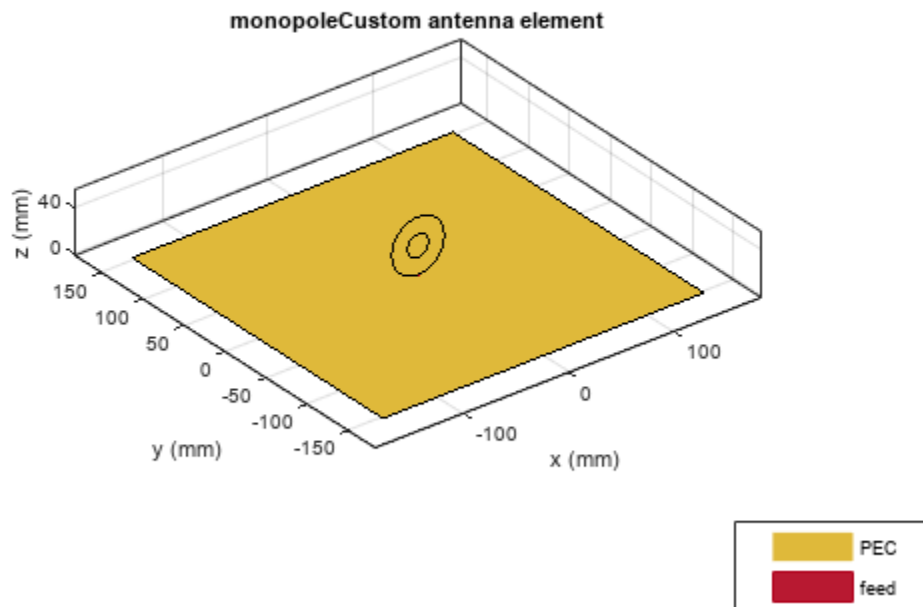
```
d = 0.8e-3; w = 0.6e-3;
offset = 0.2e-3; % offset value when a vertex touches the feed.
FeedStrip = antenna.Rectangle("Length",w,"Width",d,"Center",[0 d/2]);

rBig = 25e-3; rSmall = 10e-3;
outerCircle = antenna.Circle("Radius",rBig,"Center",[0 rBig+d-offset]);
innerCircle = antenna.Circle("Radius",rSmall,"Center",[0 rBig+d-offset]);
radiator = outerCircle-innerCircle+FeedStrip;
```

Create Planar Monopole

To create the annular ring planar monopole antenna, use the `monopoleCustom` object.

```
ant = monopoleCustom ("Radiator",radiator,"GroundPlane",groundPlane);
show(ant)
```

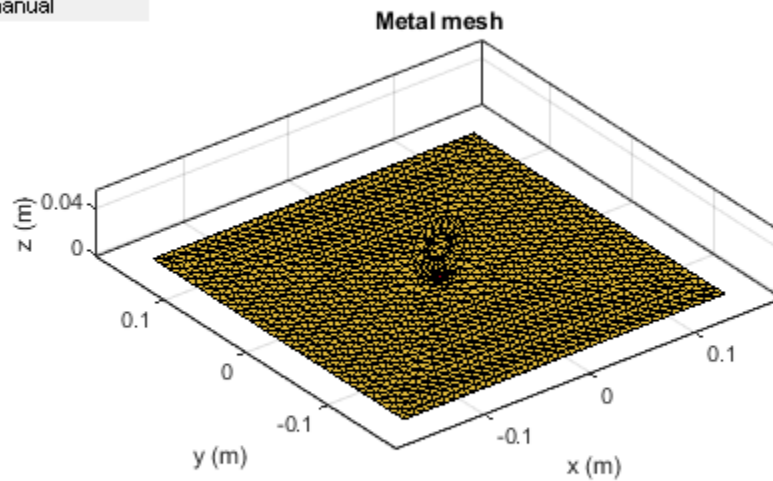


Create Antenna Mesh

Mesh the antenna manually by using at least 10 elements per wavelength at 3 GHz before running the analysis. Set the MaxEdgeLength to $\lambda/10$.

```
figure;
mesh(ant, "MaxEdgeLength", 0.01)
```

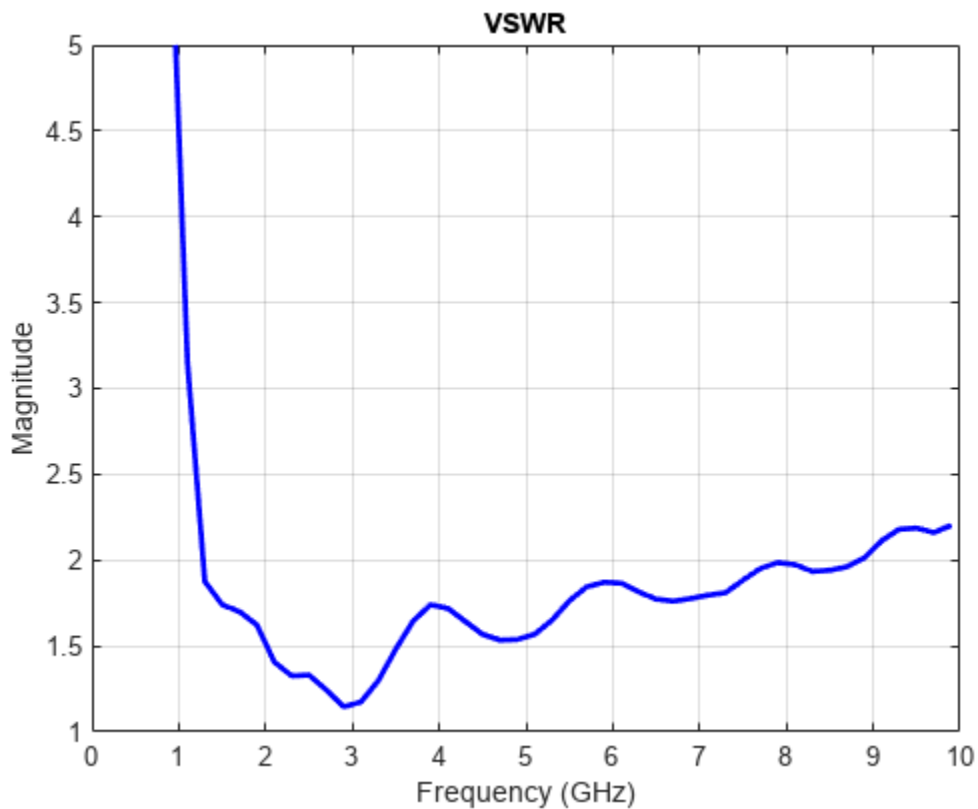
```
NumTriangles: 2350
NumTetrahedra: 0
NumBasis:
MaxEdgeLength: 0.01
MeshMode: manual
```



Calculate VSWR

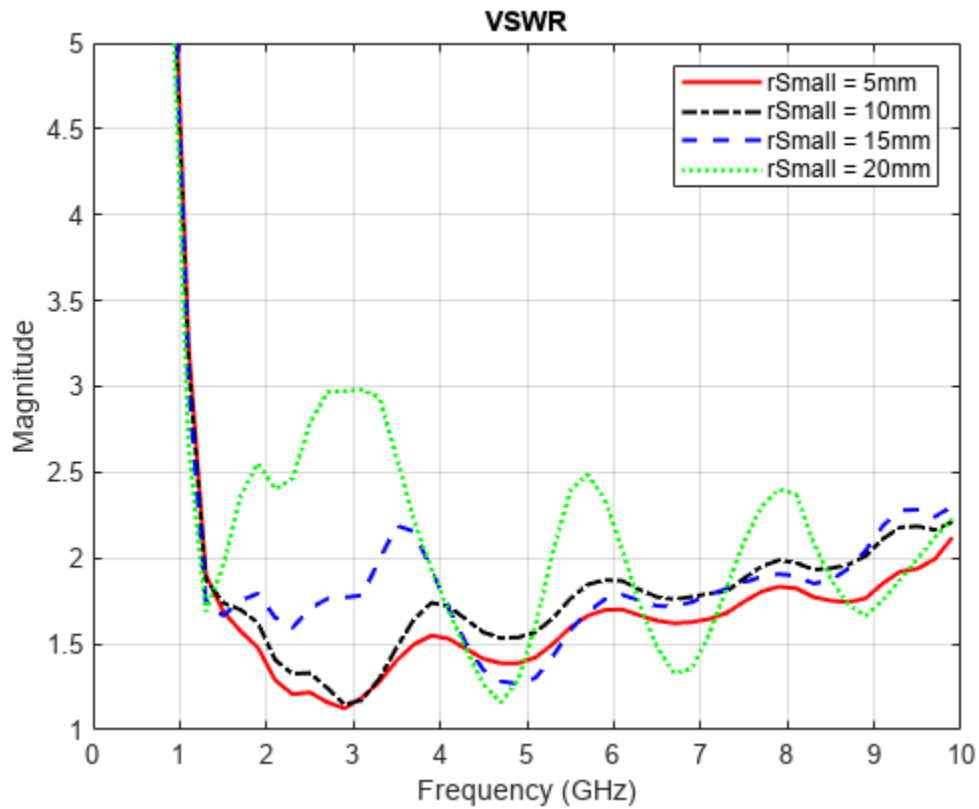
Calculate the voltage standing wave ratio (VSWR) of the annular ring planar monopole antenna to find the impedance bandwidth of the antenna. The VSWR is less than 2 in the the frequency range of 1.2 GHz to 8.7 GHz.

```
figure;
vswr(ant, (0.5:0.2:10)*1e9, 50);
ylim([1 5]);
```



Plot VSWR for different planar antennas where the `rSmall` is set to 5 mm, 10 mm, 15 mm and 20 mm.

```
load annularRingVSWR.mat % load data from annularRingVSWR.mat file
figure; plot(freq,r5,'LineStyle','-',"Color",'r','LineWidth',1.5); grid on; hold on;
plot(freq,r10,'LineStyle','-."Color",'k','LineWidth',1.5);
plot(freq,r15,'LineStyle',"--","Color",'b','LineWidth',1.5);
plot(freq,r20,'LineStyle'," ":"Color",'g','LineWidth',1.5);
ylim([1 5]);
xlabel('Frequency (GHz)');
ylabel('Magnitude');
title('VSWR');
legend('rSmall = 5mm', 'rSmall = 10mm', 'rSmall = 15mm', 'rSmall = 20mm')
```

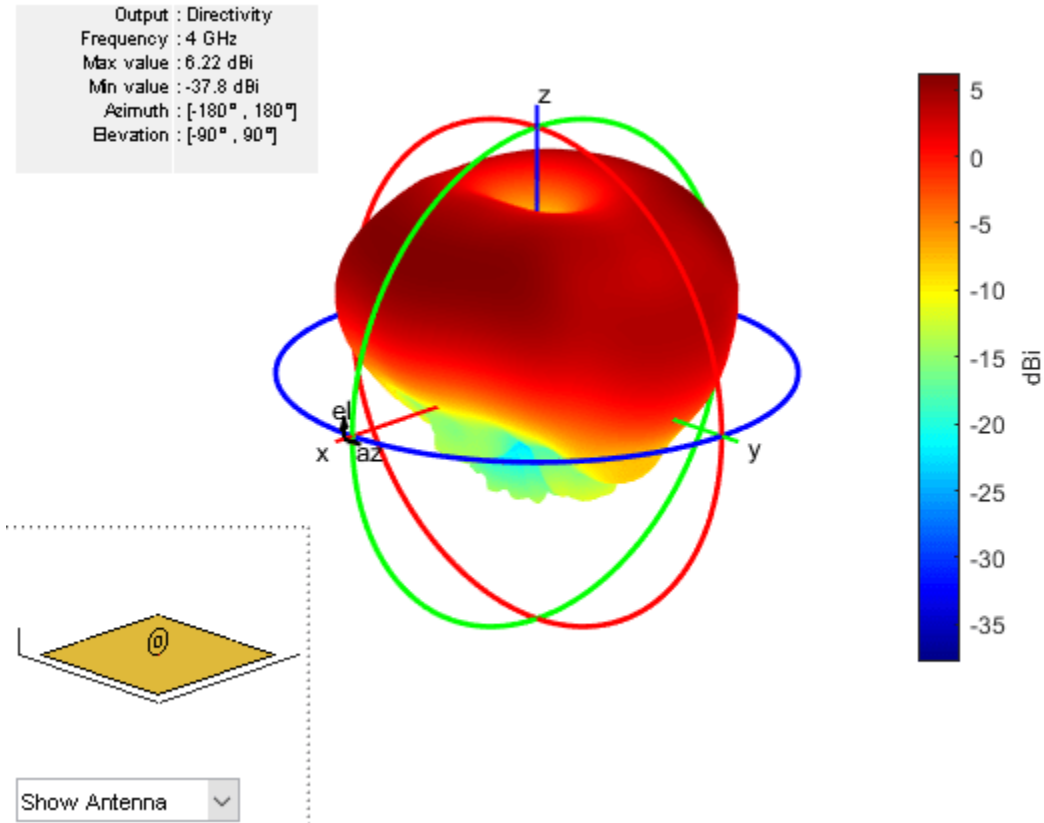


You can see that as the value of `rSmall` increases from 5 mm to 10 mm, there is change in the upper band edge frequency from 9.7 GHz to 8.7 GHz. As this increases further, the antenna becomes a multiband antenna instead of a wideband antenna. In all these cases, the lower band edge frequency is the same. In this example, the `rSmall` or the `innerCircle` radius value is set to 10 mm.

Plot Antenna Radiation Pattern

Analyze the radiation pattern of the antenna is at 4 GHz.

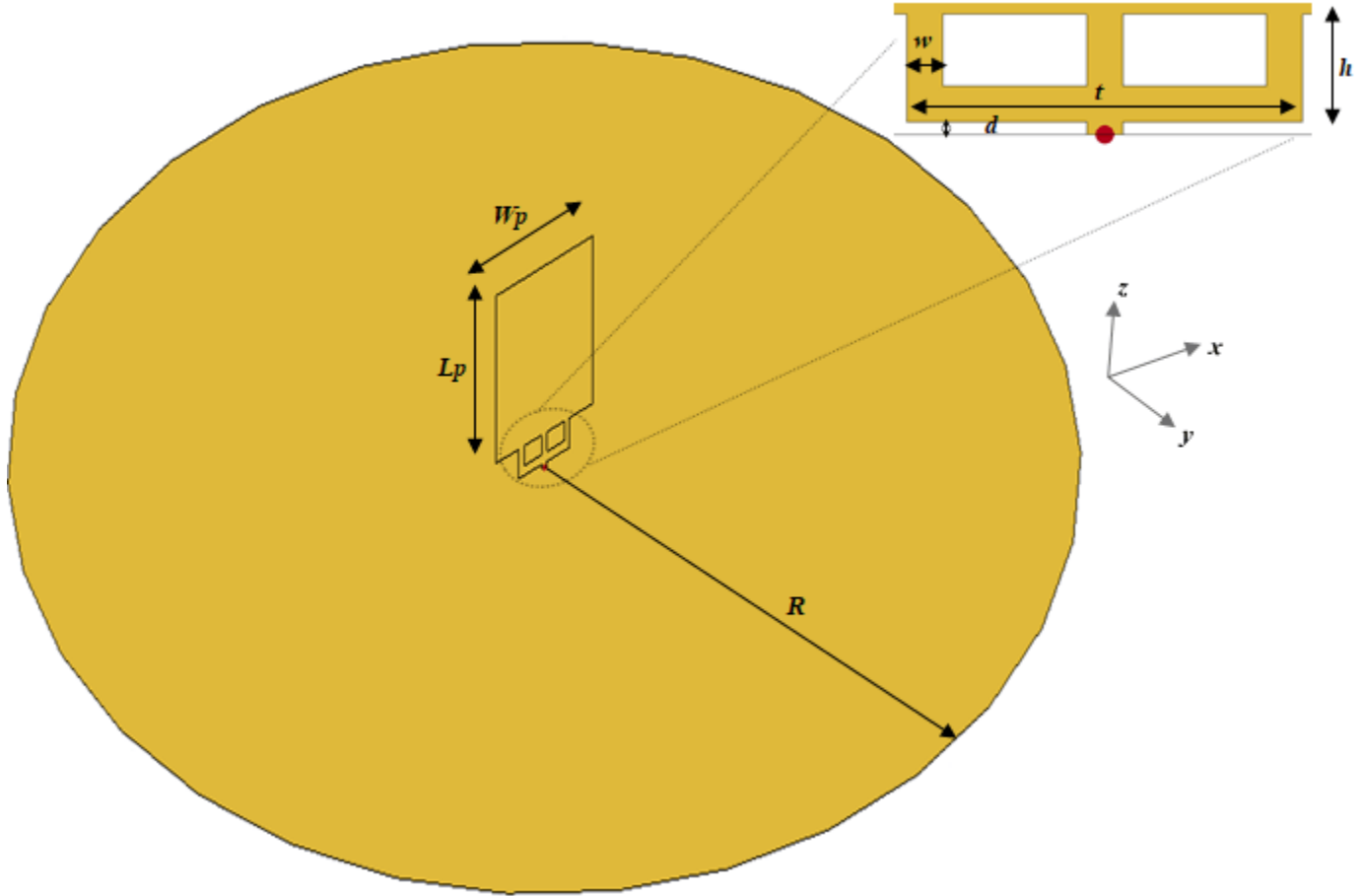
```
figure; pattern(ant,4e9);
```



At 4 GHz, the annular ring monopole antenna exhibits typical omnidirectional monoplanar radiation pattern with a maximum gain of 6.23 dBi. The simulation results agree with the results presented in [1] on page 5-735.

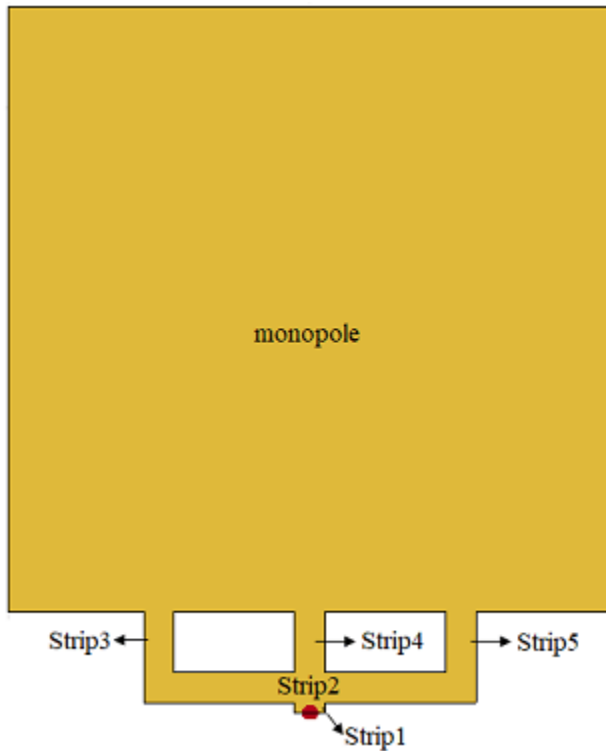
Square Planar Monopole Antenna with Trident-Shaped Feeding Strip

This figure shows the geometry and dimensions of a square planar antenna with the square plane has a side length of 34 mm. The trident-shaped feeding strip parameters are: distance between the ground plane and the feeding strip (d) is 1 mm, height of the feeding strip from d (h) is 4 mm, the horizontal length of the feeding strip (t) is 18 mm, and the width (w) of all the three strips that make the trident is 2 mm. The radius of the ground plane (R) is 150 mm. The idea of the trident-fed square monopole is taken from [2] on page 5-735, but in this example uses a circular ground plane.



Create Radiator

The radiator comprises of a trident-shaped feeding strip and square monopole. Create the trident-shaped feeding strip object, Feed, by adding the Strip1, Strip2, Strip3, Strip4 and Strip5 shapes. To create radiator, add the Feed with monopole shape.



```
Lp = 34e-3;
Wp = 34e-3;
h = 4e-3;
d = 1e-3;
w = 2e-3;
t = 18e-3;
```

```
Strip1 = antenna.Rectangle("Length",w,"Width",d,"Center",[0 d/2]);
Strip2 = antenna.Rectangle("Length",t,"Width",w,"Center",[0 d+w/2]);
Strip3 = antenna.Rectangle("Length",w,"Width",h,"Center",[-(t/2-w/2) d+w+h/2]);
Strip4 = antenna.Rectangle("Length",w,"Width",h,"Center",[0 d+w+h/2]);
Strip5 = antenna.Rectangle("Length",w,"Width",h,"Center",[t/2-w/2 d+w+h/2]);
Feed = Strip1+Strip2+Strip3+Strip4+Strip5;
monopole = antenna.Rectangle("Length",Lp,"Width",Wp,"Center",[0 d+w+h+Wp/2]);
radiator = Feed + monopole;
```

Create Ground Plane

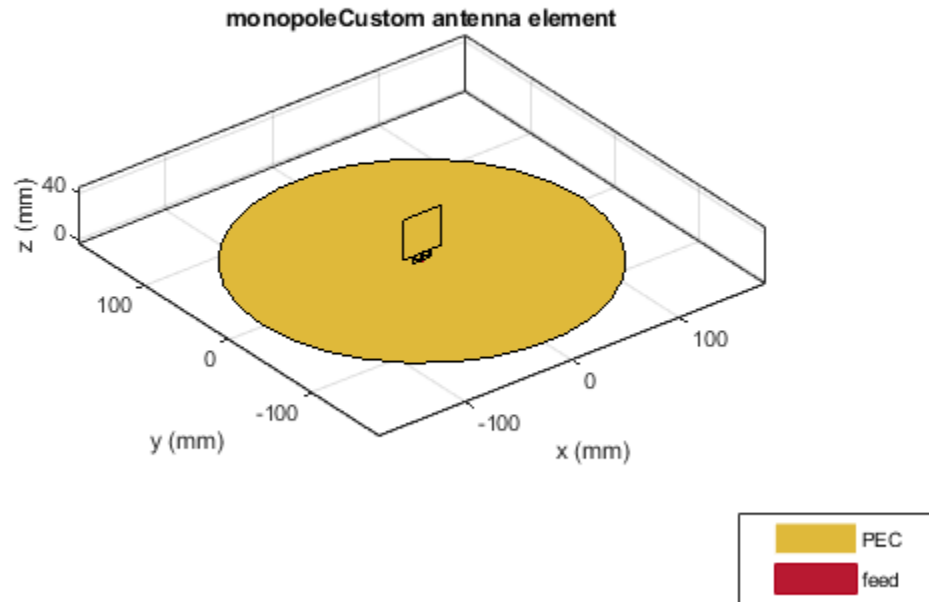
Create the circular ground plane using the `antenna.Circle` function with radius of 150 mm.

```
R = 150e-3;
groundPlane = antenna.Circle("Radius",R);
```

Create Planar Monopole

To create a trident-fed square planar monopole antenna, use the `monopoleCustom` object.

```
ant = monopoleCustom("Radiator",radiator,"GroundPlane",groundPlane);
show(ant)
```

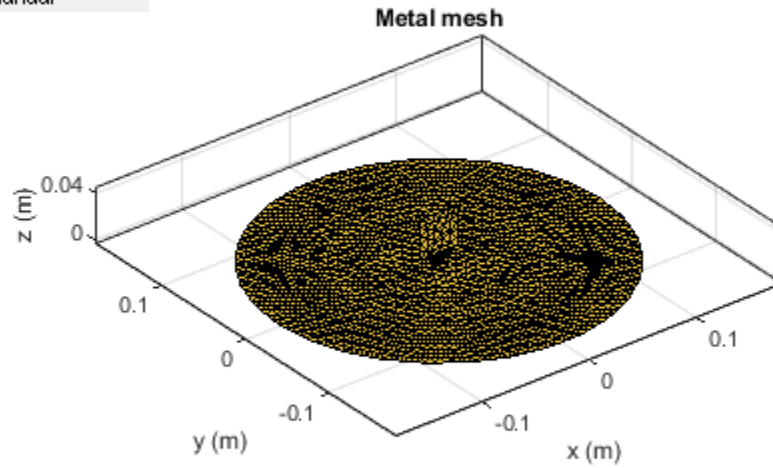


Meshing antenna

Mesh the antenna manually by using at least 10 elements per wavelength at 4 GHz before running the analysis. Set the `MaxEdgeLength` to $\lambda/10$.

```
figure;  
mesh(ant, "MaxEdgeLength", 0.0075);
```

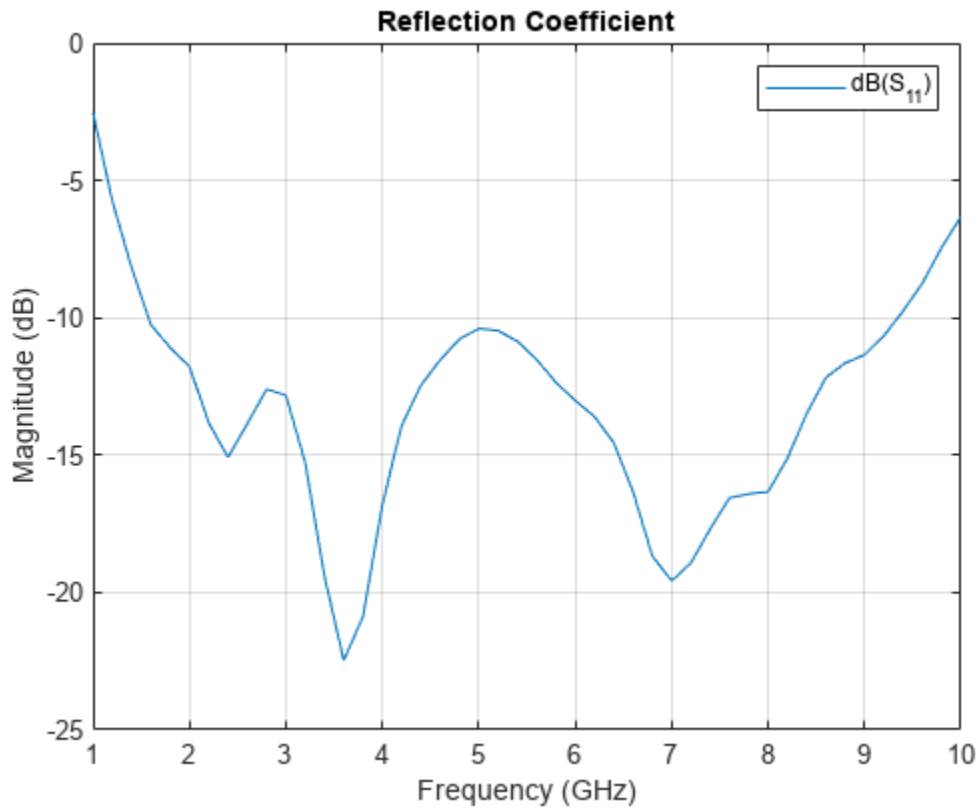
```
NumTriangles: 3080  
NumTetrahedra: 0  
NumBasis:  
MaxEdgeLength: 0.0075  
MeshMode: manual
```



Calculate Reflection Coefficient

Calculate the reflection coefficient of the planar monopole antenna to find the impedance bandwidth of the antenna. The reflection coefficient is less than -10 dB in the frequency range 1.6 GHz to 9.3 GHz.

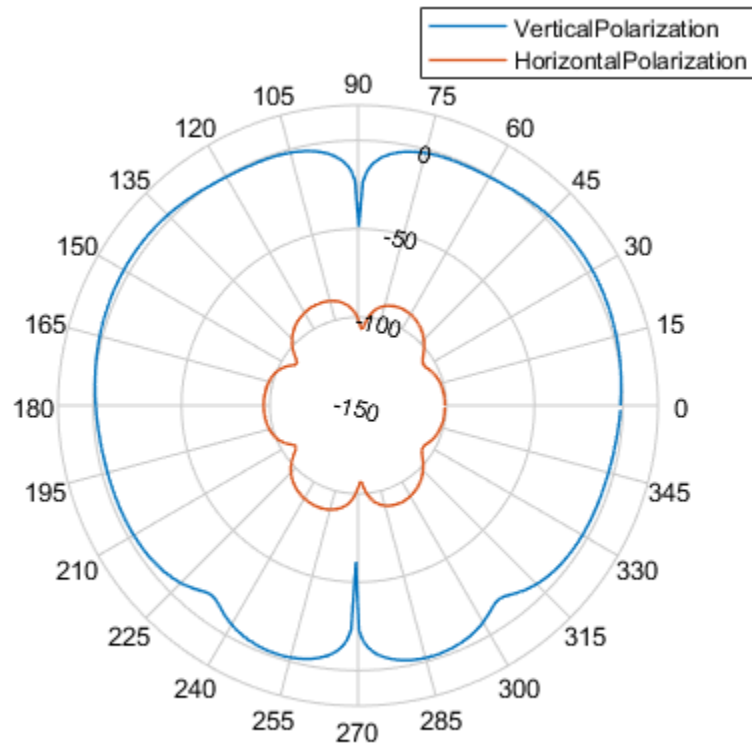
```
freq = (1:0.2:10)*1e9;  
s = sparameters(ant,freq);  
figure;  
rfplot(s);  
title('Reflection Coefficient');
```

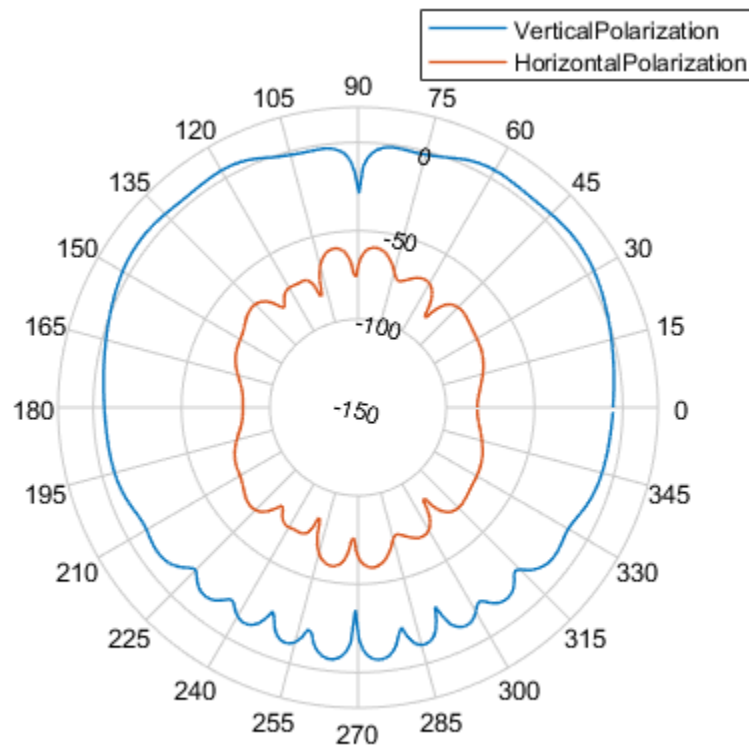
Plot Radiation Pattern

Analyzing the radiation pattern of the antenna at 2 GHz, 6 GHz, and 9 GHz shows the vertical polarization as stable in the entire range of frequencies. But as the frequency increases, there is an increase in the horizontal polarization component.

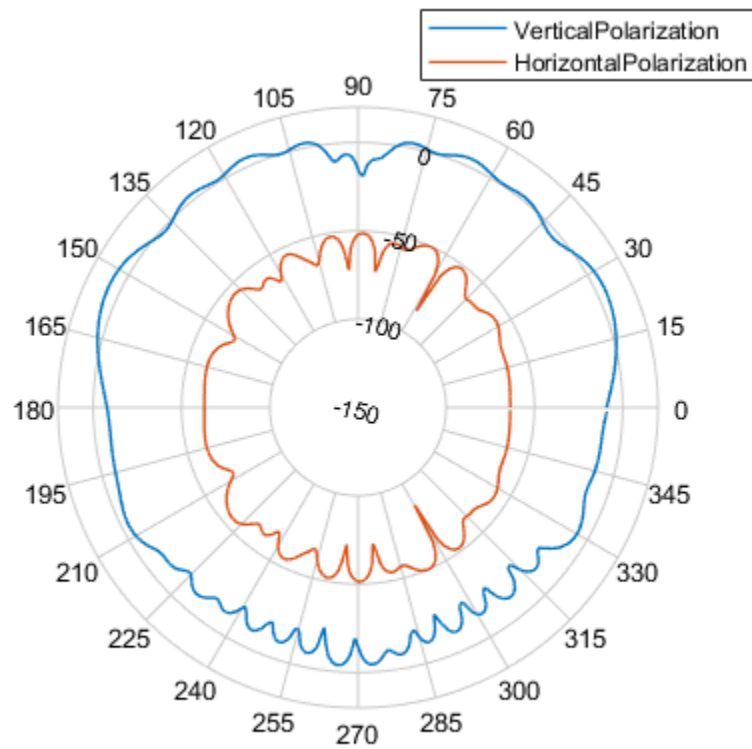
```
pV_1 = pattern (ant, 2e9, 0, 0:1:360, 'Polarization', 'V');
pH_1 = pattern (ant, 2e9, 0, 0:1:360, 'Polarization', 'H');
figure;
polarpattern(pV_1, 'MagnitudeLim', [-150 20]);
hold on;
polarpattern(pH_1, 'MagnitudeLim', [-150 20]);
legend VerticalPolarization HorizontalPolarization;
```



```
pV_2=pattern (ant, 6e9, 0, 0:1:360,'Polarization','V');  
pH_2=pattern (ant, 6e9, 0, 0:1:360,'Polarization','H');  
figure; polarpattern(pV_2,'MagnitudeLim',[-150 20]);  
hold on; polarpattern(pH_2,'MagnitudeLim',[-150 20]);  
legend VerticalPolarization HorizontalPolarization;
```



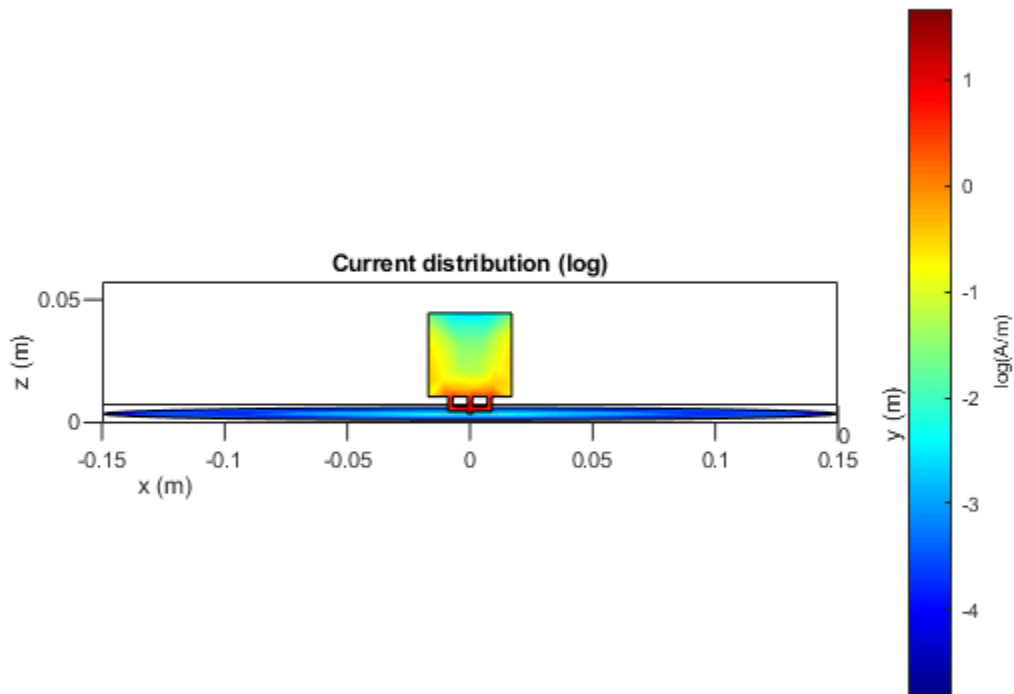
```
pV_3 = pattern (ant, 9e9, 0, 0:1:360, 'Polarization', 'V');  
pH_3 = pattern (ant, 9e9, 0, 0:1:360, 'Polarization', 'H');  
figure;  
polarpattern(pV_3, 'MagnitudeLim', [-150 20]);  
hold on;  
polarpattern(pH_3, 'MagnitudeLim', [-150 20]);  
legend VerticalPolarization HorizontalPolarization;
```



View Current Distribution

The current distribution of the antenna is analyzed at 2.5 GHz. The trident-shaped feeding structure results in uniform current distribution in the lower part of the square monopole.

```
current(ant,2.5e9,'scale','log');  
view(0,1)
```



Conclusion

Planar monopole antennas have simple geometry, and you can define their radiators and the ground planes in variety of different shapes. These antennas offer ultrawide impedance bandwidth with an azimuthal radiation pattern that is nearly omnidirectional.

Reference

- [1] Ammann, M. J. Chen, Z. N. Chia, M. Y. W. and See, T. S. P. . "Annular Planar Monopole Antennas." *IEE Proceedings - Microwaves, Antennas and Propagation*, Vol. 149, No. 4, August 2002, pp. 200-203.
- [2] Kin-Lu Wong, Chih-Hsien Wu and Saou-Wen Su, "Ultrawide-band square planar metal-plate monopole antenna with a trident-shaped feeding strip," *IEEE Transactions on Antennas and Propagation*, Vol. 53, No. 4, April 2005, pp. 1262-1269.

See Also

"Design Internally Matched Ultra-Wideband Vivaldi Antenna" on page 5-410

Maximize Impedance Bandwidth of Triangular Patch Antenna

This example shows how to optimize a triangular microstrip patch antenna to maximize its bandwidth such that its gain remains constant.

Define Antenna Geometry

Define the initial geometry of the antenna.

```
% Define Antenna
ant = patchMicrostripTriangular;

% Center frequency and frequency range
fc = 10e9;
% For BW ~20%
minFreq = fc * 0.8;
maxFreq = fc * 1.2;
freqRange = linspace(minFreq, maxFreq, 35);

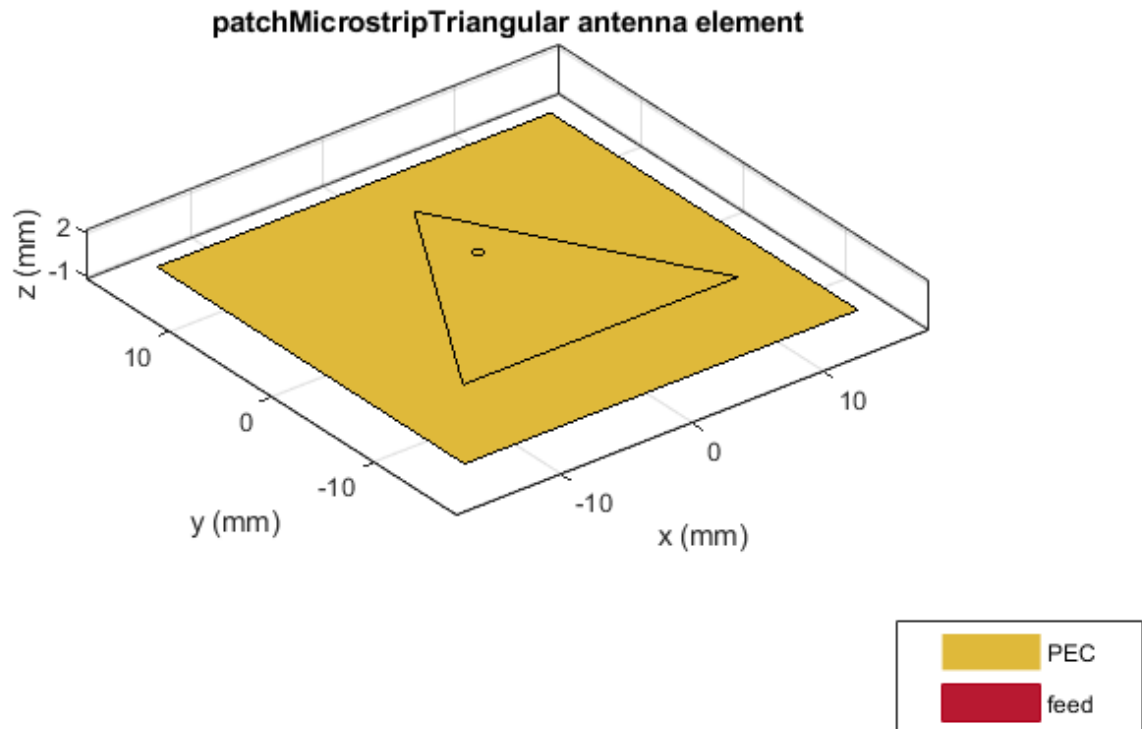
% Initial point
ant = design(ant, fc)

ant =
    patchMicrostripTriangular with properties:

        Side: 0.0210
        Height: 0.0012
        Substrate: [1x1 dielectric]
    GroundPlaneLength: 0.0300
    GroundPlaneWidth: 0.0300
    PatchCenterOffset: [0 0]
        FeedOffset: [0 0.0029]
    FeedDiameter: 3.7474e-04
        Conductor: [1x1 metal]
        Tilt: 0
        TiltAxis: [1 0 0]
        Load: [1x1 lumpedElement]
```

Use `show` function to view the antenna.

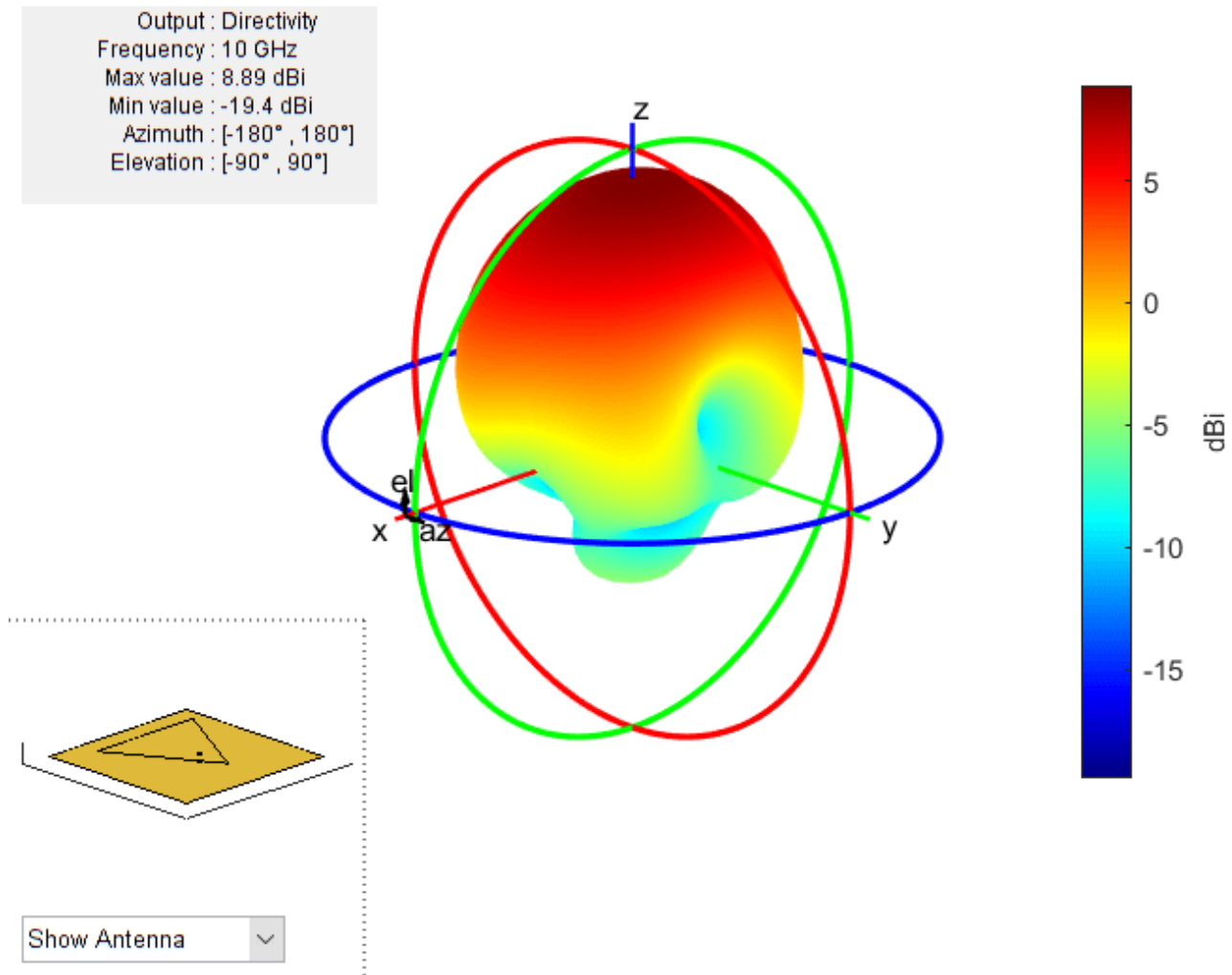
```
show(ant)
```



Initial Analysis

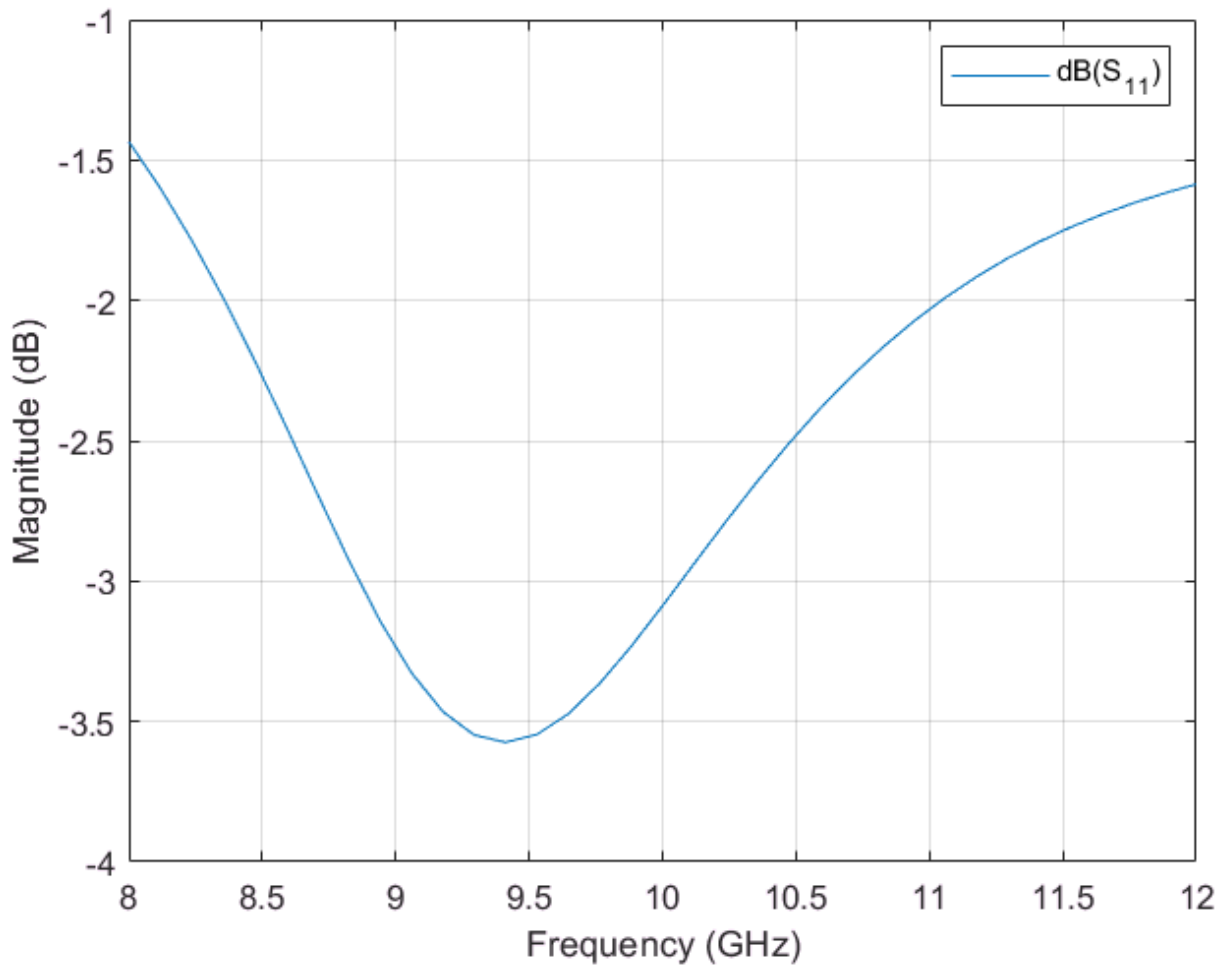
Analyze the gain of the antenna. Record the maximum gain of the antenna before optimization.

```
pattern(ant, fc);
```



Analyze the impedance bandwidth of the antenna using S-parameters.

```
rfplot(sparameters(ant, freqRange));
```

Set Up Optimization Problem

You will need the following inputs for the optimization problem:

1. Objective Function: Choose 'maximizeBandwidth' as the objective function. The main goal of this example is maximizing the bandwidth of the antenna.
2. Design variables: Select side length, height of the triangular patch antenna, and feed offset as control variables for the objective function. The side length of the triangular patch will affect the front lobe gain. The height and the feed location will enhance the impedance bandwidth of the antenna and help improve its matching.
3. Constraints: The constraint function is gain more than 8.5 dBi.

```
% Design Variables
% Property Names
propNames = {'Side', 'Height', 'FeedOffset'};

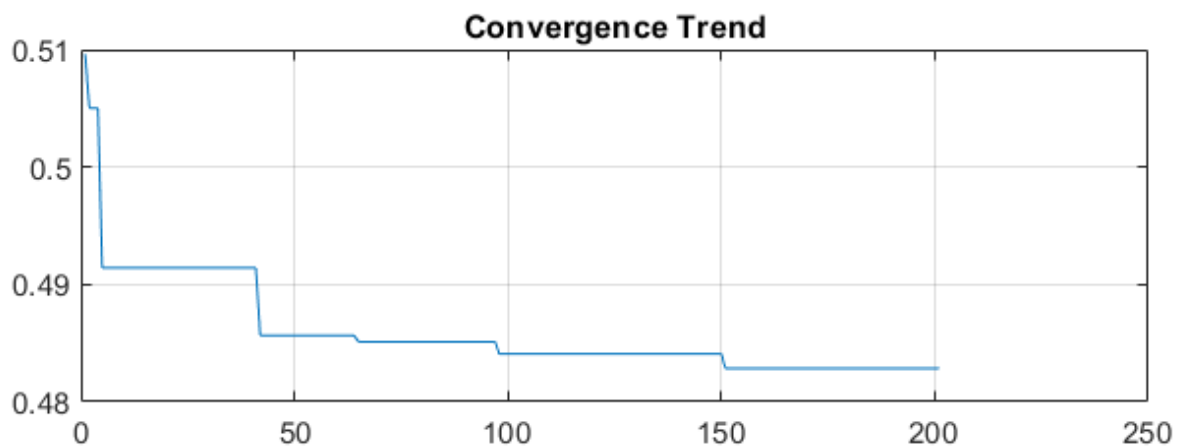
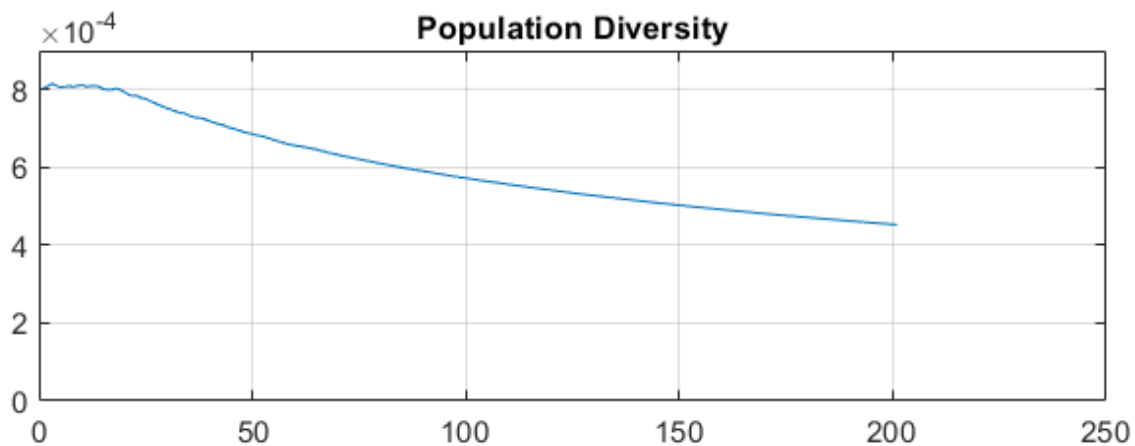
% Bounds for the properties {lower bounds; upper bounds}
bounds = {0.017 0.001 [0 0.0017]; ... % Specify the FeedOffset vector to move the feed along [x,
0.023 0.004 [0.0002 0.0034]]};
```

Optimization Function

To optimize the antenna, use the `optimize` function.

```
figure; % Figure to display the convergence trend
% Optimize
optAnt = optimize(ant, fc, 'maximizeBandwidth', ...
    propNames, bounds, ...
    'Constraints', {'Gain > 8.5'}, ...
    'FrequencyRange', freqRange, ...
    'UseParallel', true);
```

Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 6).



Set 'EnableLog' as true, in order to print the iteration number and best value of convergence on the command line.

```
Iteration: 2, Best: 0.508140, -0.158702
Iteration: 3, Best: 0.508140, -0.158702
.
.
.
Iteration: 199, Best: 0.490964, -0.001916
Iteration: 200, Best: 0.490964, -0.001916
Iteration: 201, Best: 0.490964, -0.001916
```

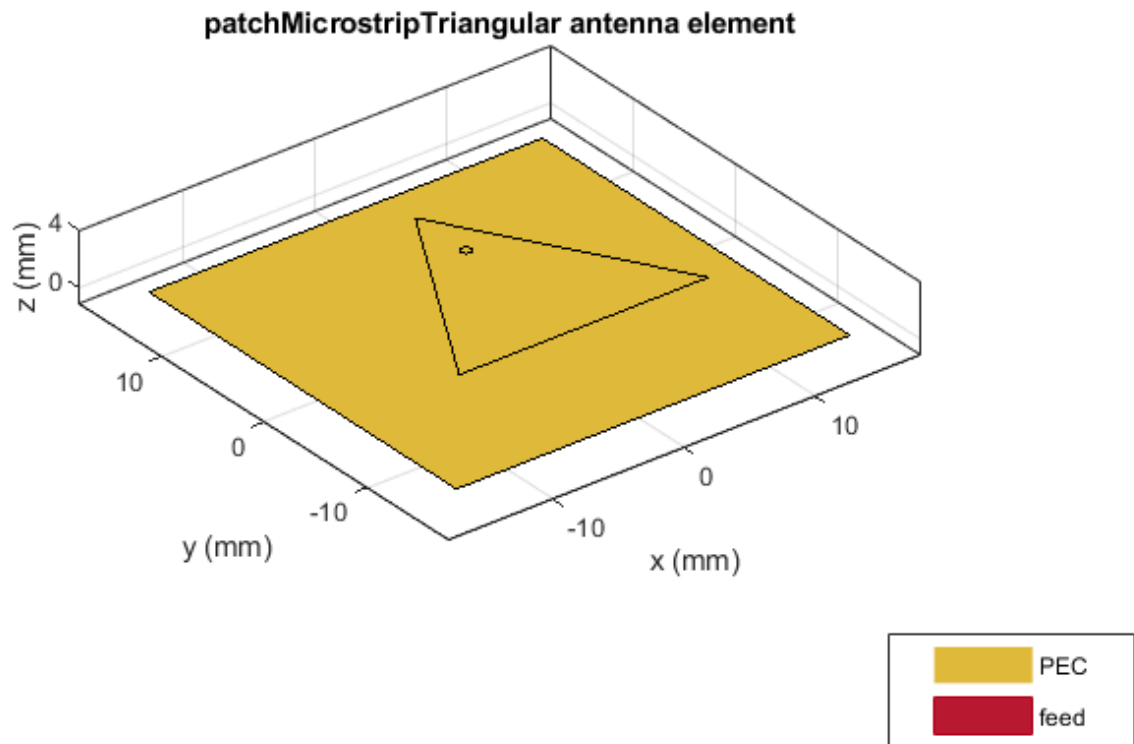
Optimization Result

Compare the analysis parameters of the antenna from before and after the optimization.

```
optAnt
```

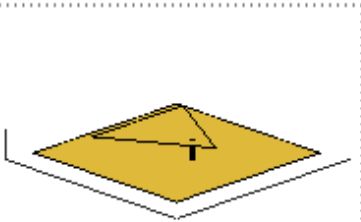
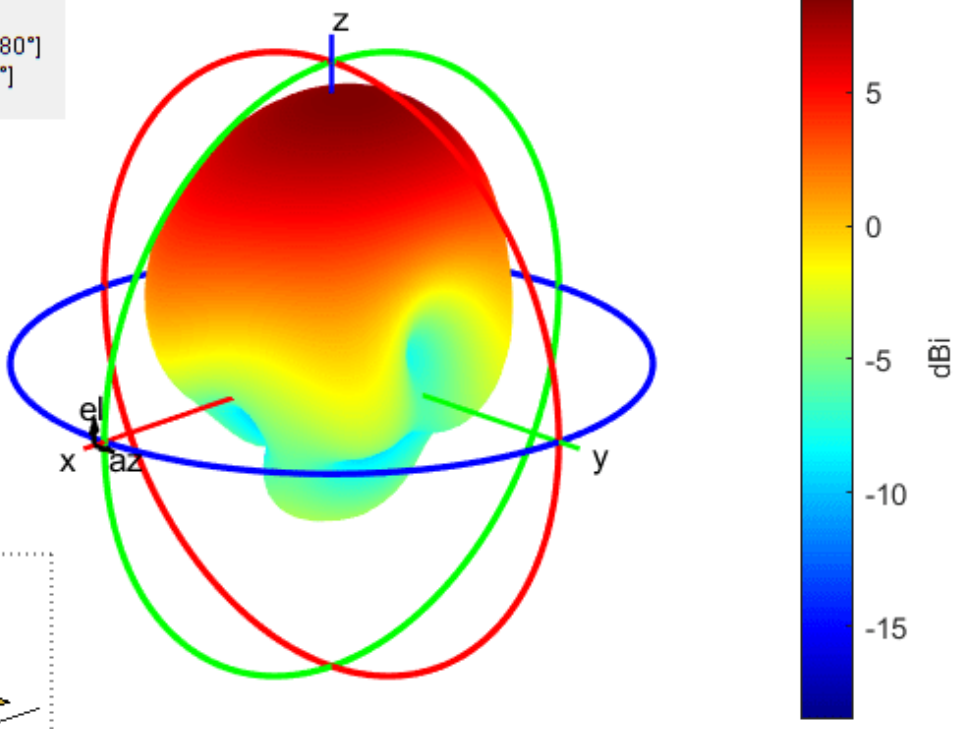
```
optAnt =
  patchMicrostripTriangular with properties:
    Side: 0.0190
    Height: 0.0029
    Substrate: [1x1 dielectric]
    GroundPlaneLength: 0.0300
    GroundPlaneWidth: 0.0300
    PatchCenterOffset: [0 0]
    FeedOffset: [2.4368e-05 0.0033]
    FeedDiameter: 3.7474e-04
    Conductor: [1x1 metal]
    Tilt: 0
    TiltAxis: [1 0 0]
    Load: [1x1 lumpedElement]
```

```
show(optAnt)
```



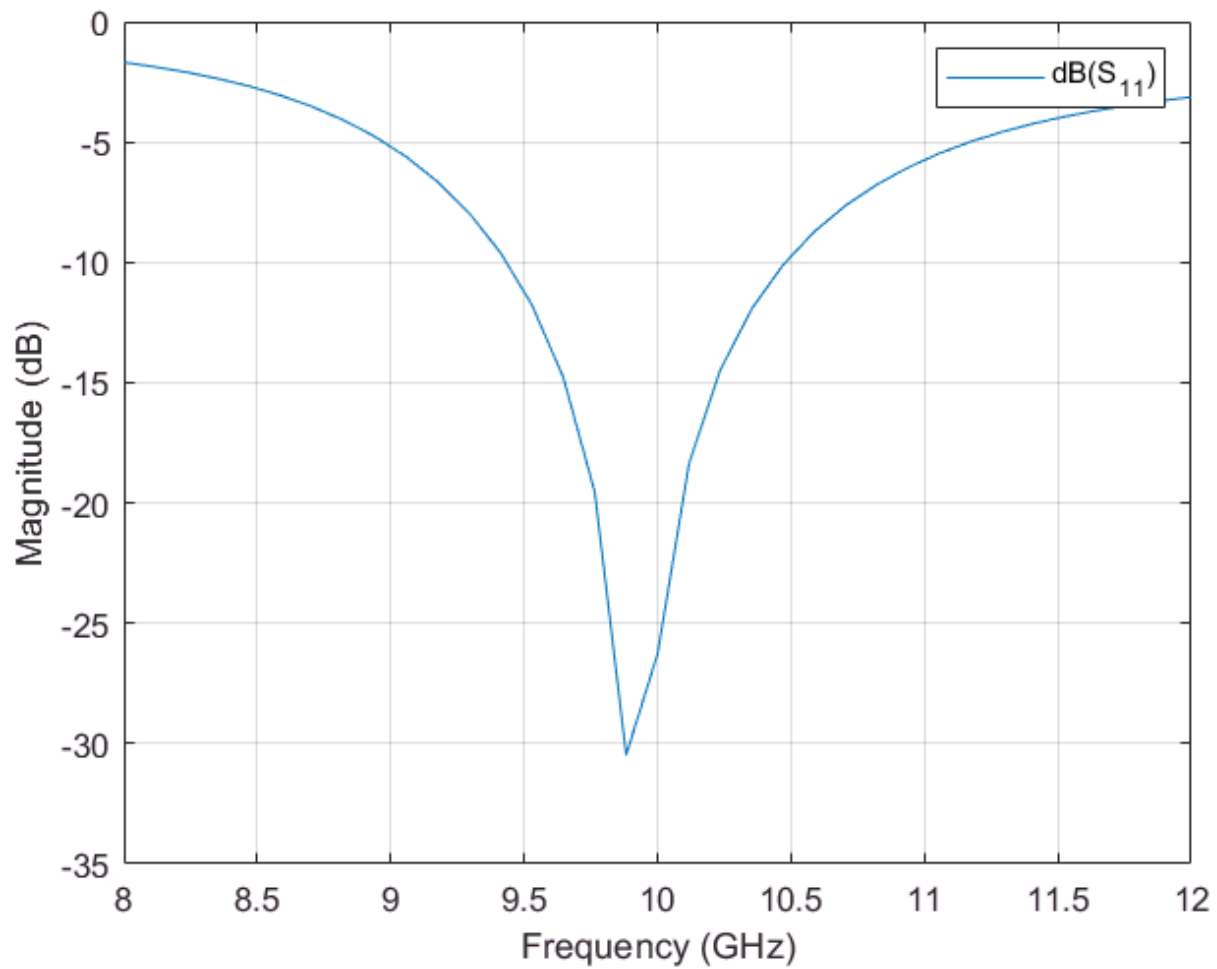
```
pattern(optAnt, fc)
```

Output : Directivity
 Frequency : 10 GHz
 Max value : 8.64 dBi
 Min value : -18.5 dBi
 Azimuth : [-180°, 180°]
 Elevation : [-90°, 90°]



Show Antenna

```
rfplot(parameters(optAnt, freqRange))
```



Observe the bandwidth of the antenna after optimization.

See Also

"Modeling and Analysis of Probe-Fed Stacked Patch Antenna" on page 5-395

Create Antenna Model from Gerber Files

This example shows you how to create an antenna model from Gerber files and subsequently analyze the antenna. The Gerber file format is used in printed circuit board (PCB) manufacturing and is available in two formats: RS-274D, which was the initial release standard, and RS-274X, which is the newer extended Gerber format. The Antenna Toolbox™ supports the newer RS-274X format both to generate Gerber files from an antenna model as well as to create an antenna model from a set of Gerber files.

Introduction

A set of Gerber files includes information about layer geometry, layer mask, solder paste usage on the layers, drill file and so on. To create a PCB antenna model out of these files, you need layer files that specify the antenna geometry, and if available a drill file to specify any plated through-hole (PTH) vias. The geometry of the layers is specified through either a top and a bottom layer file, with extensions `.gtl` and `.gbl`, or a Gerber file, with the extension `.gbr`. The Antenna Toolbox supports the Excellon format to specify drill information with file extensions `.txt` or `.drl`. To create an antenna model, import up to two layers and an optional drill file.

This example will generate an antenna model using a single layer file and a two-layer file.

Import Single Layer Design and Analysis

While you can specify several antennas that are specified on a single side of a PCB, this example uses the design for an inverted-L antenna. The first step is to import a top-layer Gerber file into the workspace using the `gerberRead` function. This will create a `PCBReader` object. The `PCBReader` object provides access to the stackup that holds the information on the metal and dielectric layers and also any drill files that describe a PTH via from one metal layer to another. The GTL file that this example provides is held in the stackup as `Layer2`. If a second layer is present, you can specify it as `Layer4` directly through the `stackUp` object or by passing it to the `gerberRead` function as the second input.

```
P1 = gerberRead('ILA_coplanar.gtl');
P1.StackUp
```

```
ans =
    stackUp with properties:

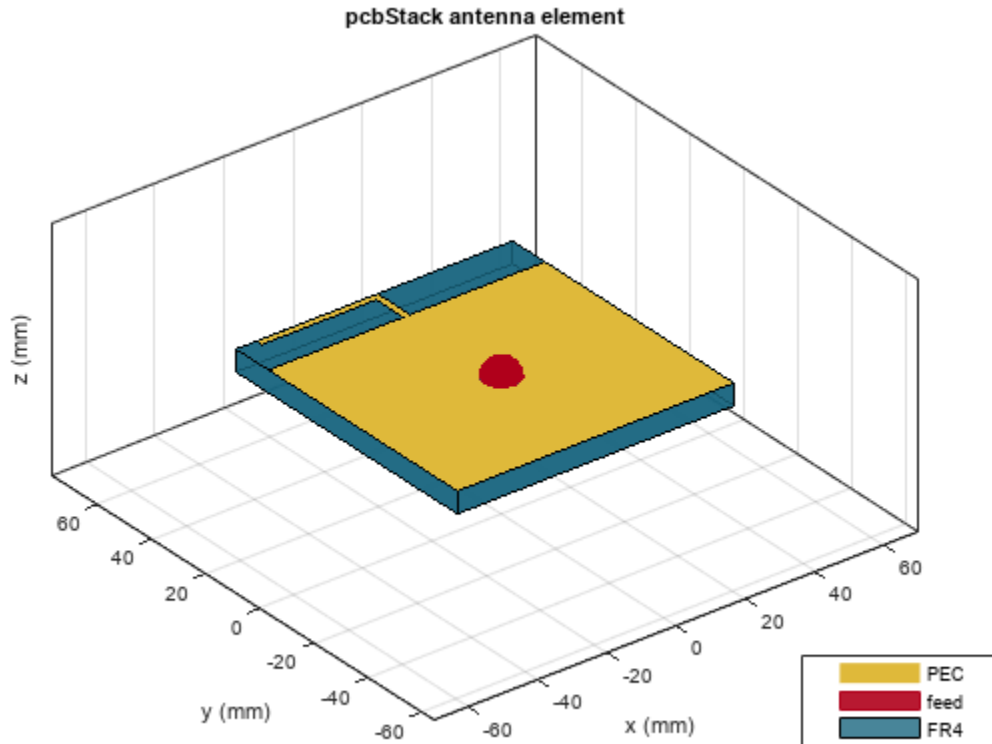
        NumLayers: 5
        Layer1: [1x1 dielectric]
        Layer2: 'ILA_coplanar.gtl'
        Layer3: [1x1 dielectric]
        Layer4: []
        Layer5: [1x1 dielectric]
```

The `PCBReader` object also provides a property to control the discretization on any curved segments in the imported layer. By default, the value of this property, `NumPointsOnCurves`, is set to 50 in this example.

Model Creation

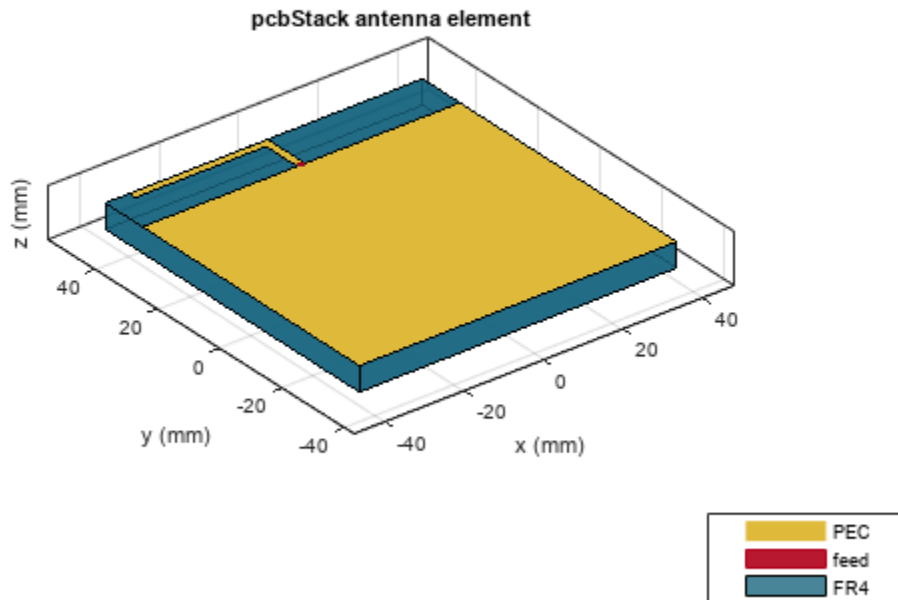
The next step is to create the antenna model. To do so, pass the `PCBReader` object as an input to the `pcbStack` object.

```
pb = pcbStack(P1);  
figure  
show(pb)
```



The Gerber file format does not provide information regarding a feed. By default, the feed is located at the origin in the created model. In order to make this model usable, modify the feed location on the layer.

```
pb.FeedDiameter = .001;  
pb.FeedLocations(1:2) = [0,0.035];  
figure  
show(pb)
```

Import Two-Layer Design and Analysis

Import a two-layer design to create the antenna model. As before, you can use the `gerberRead` function to create the `PCBReader` object. Pass in the top and bottom layers as inputs to the function.

```
P2 = gerberRead('UWBVivaldi.gtl','UWBVivaldi.gbl');
P2.StackUp
```

```
ans =
    stackUp with properties:

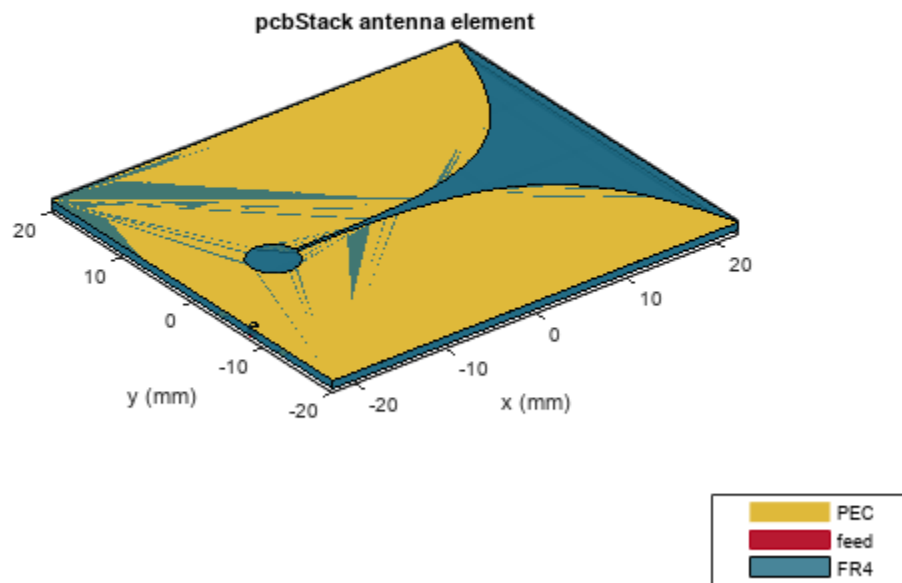
    NumLayers: 5
    Layer1: [1x1 dielectric]
    Layer2: 'UWBVivaldi.gtl'
    Layer3: [1x1 dielectric]
    Layer4: 'UWBVivaldi.gbl'
    Layer5: [1x1 dielectric]
```

Modify the third layer in the stack, which is the dielectric layer between the top and bottom metal layers.

```
S = P2.StackUp;
S.Layer3 = dielectric('Name','FR4','EpsilonR', 4.4, 'Thickness', 0.8e-3);
P2.StackUp = S;
```

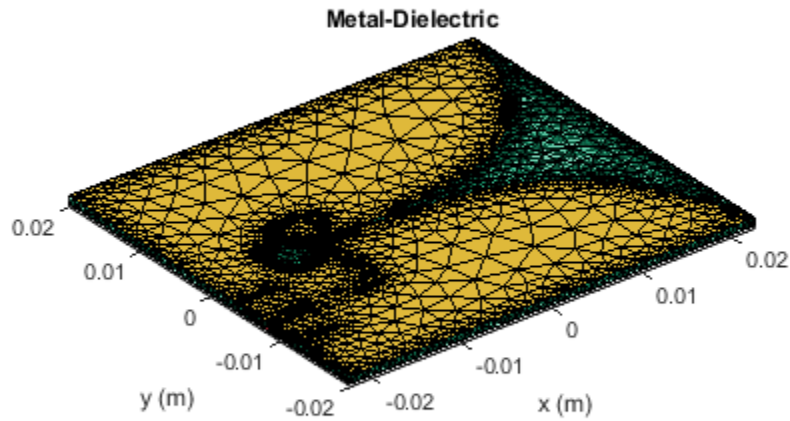
Create the antenna model by passing `pcbStack` object as an input to the `PCBReader` object. In addition, modify the feed information such as the coordinates, layer, and the feed diameter.

```
pb2 = pcbStack(P2);  
pb2.BoardThickness = 0.8e-3;  
pb2.FeedLocations=[-(44e-3/2), -(40e-3/2 - 11.2e-3 - 1.5e-3/2), 2, 4];  
pb2.FeedDiameter = 1.5e-3/2;  
figure  
show(pb2)  
axis equal;
```



```
figure  
mesh(pb2, 'MaxEdgeLength',5e-3, 'MinEdgeLength',0.8e-3);
```

NumTriangles: 3061
NumTetrahedra: 10821
NumBasis:
MaxEdgeLength: 0.005
MeshMode: manual



Summary

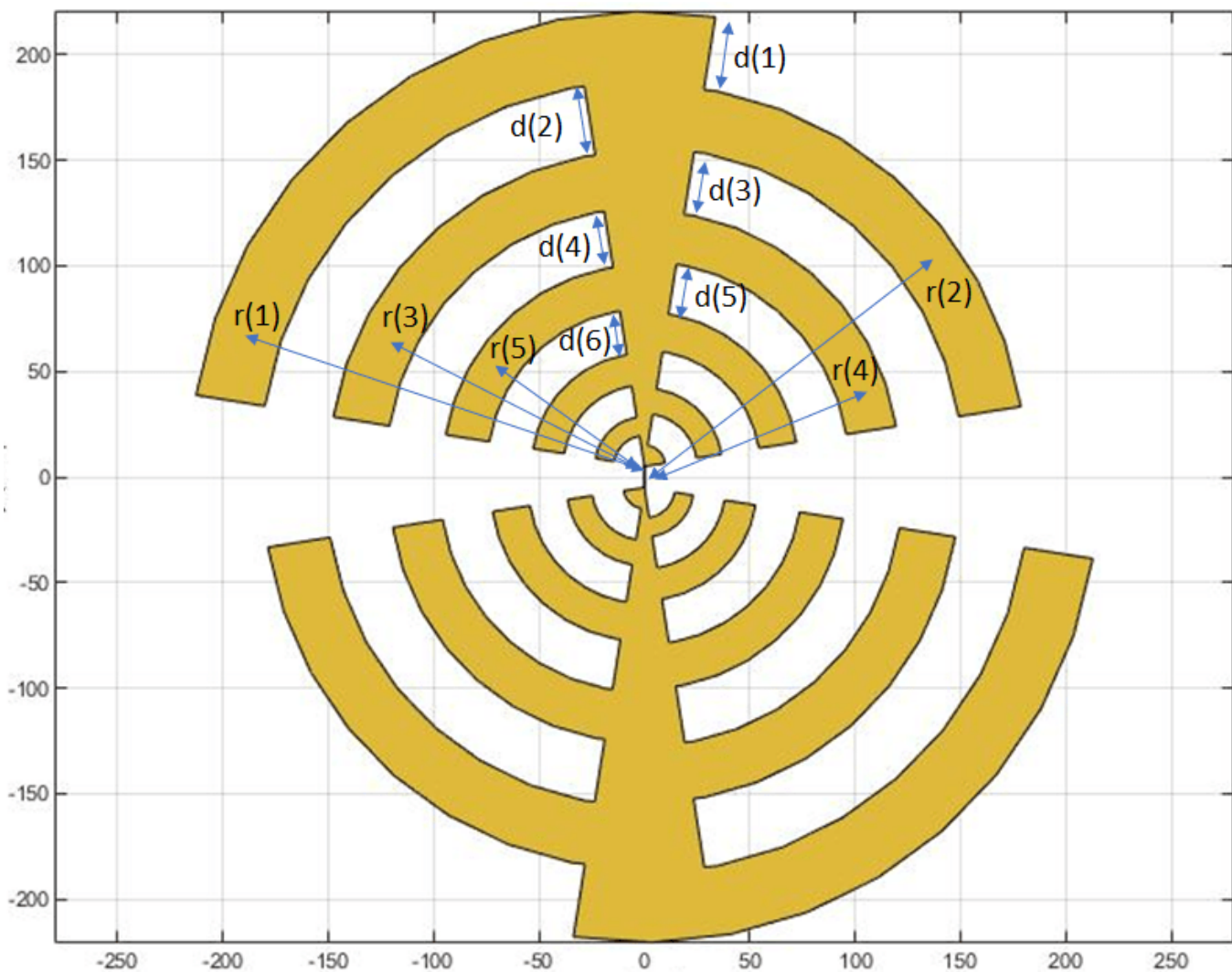
Using the `gerberRead` function, you can create a `PCBReader` object, and subsequently use that object to generate the antenna model using a `pcbStack` object.

See Also

`PCBReader` | `gerberRead` | `shapes` | `stackUp` | “Design, Analysis, and Prototyping of Microstrip-Fed Wide-Slot Antenna” on page 5-356

Design Log-Periodic Sawtooth Planar Antenna for UHF Ultra-Wideband Applications

This example shows how to create, model, and analyze two-arm log-periodic sawtooth planar microstrip antenna [1] on page 5-758. It is a bowtie-like antenna that consists of two arms located on the square-shaped FR4 dielectric substrate. This example designs a log-periodic sawtooth planar antenna with stable impedance over a wide bandwidth from 300 MHz to 1200 GHz. This antenna is widely used in the ultra high frequency (UHF) range and for ultra-wideband (UWB) systems such as ground penetrating radar systems (GPRs). The non-invasive and nondestructive technique in GPRs uses electromagnetic radiation in the microwave band. GPRs use high-frequency radio waves, usually in the 10 MHz to 2.6 GHz range.



Define Parameters

The dimensions and the variables used in this example are taken from the paper [1]. The variables r and d specify the radius and width of the arms of the log-periodic antenna. The Angle variable is

used as an `ArcAngle` for creating the arms of the antenna. The `r`, `d`, and `Angle` variables are used to design the arms on either side of the center arc. The `NumPoints` variable is used to specify the number of discretization points.

```
Angle      = 18;
r          = [199 166 136 109 85 64 46 31 19 5]*1e-3;
d          = [33 30 27 24 21 18 15 12 9 10]*1e-3;
NumPoints = 8;
```

Create Log-Periodic Shape

Use the `generatePointsforCurve` function to create the vertices for the arms of the antenna. This function takes the angle, radius, width, and number of discretization points as the inputs and generates the boundary vertices for the curves. These vertices are used to form an arm using the `antenna.polygon` function. All the arms on either side of the center arc can be generated by using `r` and `d` variables. All the polygons formed are combined to form the upper half of the log-periodic antenna.

```
NumShapes = numel(1,r);
p1 = generatePointsforCurve([90-Angle/2,90+Angle/2],(205e-3)/2,205e-3,NumPoints/2);
shp = antenna.Polygon('Vertices',p1);
for i=1:NumShapes
    if rem(i,2)==1
        tempPoints = generatePointsforCurve([90-Angle/2,180-Angle/2],r(i),d(i),NumPoints);
        tempShape = antenna.Polygon('Vertices',tempPoints);
        shp = shp+tempShape;
    else
        tempPoints = generatePointsforCurve([Angle/2,90+Angle/2],r(i),d(i),NumPoints);
        tempShape = antenna.Polygon('Vertices',tempPoints);
        shp = shp+tempShape;
    end
end
```

Shift the shape along the y-axis to create a gap for the feed using the `translate` function. The total gap given in [1] is 10 mm, so shift the upper half of the shape by 5 mm.

```
shp = translate(shp,[0 5e-3 0]);
```

Copy the shape into a variable `shp1` and use it to create a symmetrical shape. Use the `rotateZ` function to rotate the shape replica by 180 degrees.

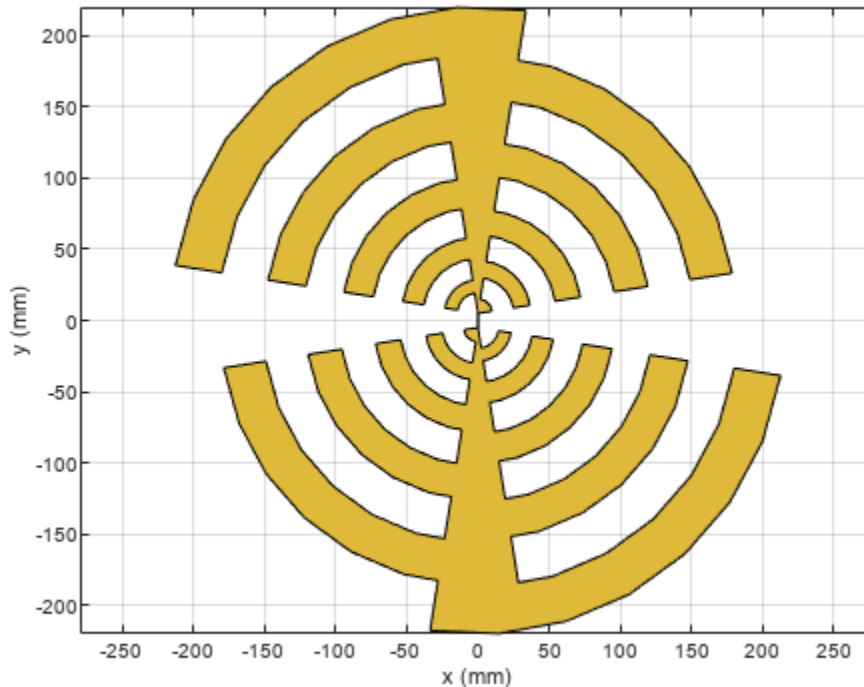
```
shp1 = copy(shp);
shp1 = rotateZ(shp1,180);
```

Create a rectangle for the feed 0.5 mm in length and 15 mm in width using the `antenna.Rectangle` shape object.

```
feed = antenna.Rectangle('Length',0.5e-3,'Width',15e-3);
```

Unify shapes `shp` and `shp1` with the `feed` to form sawtooth-shaped antenna structure.

```
shape = shp+shp1+feed;
figure;
show(shape)
```



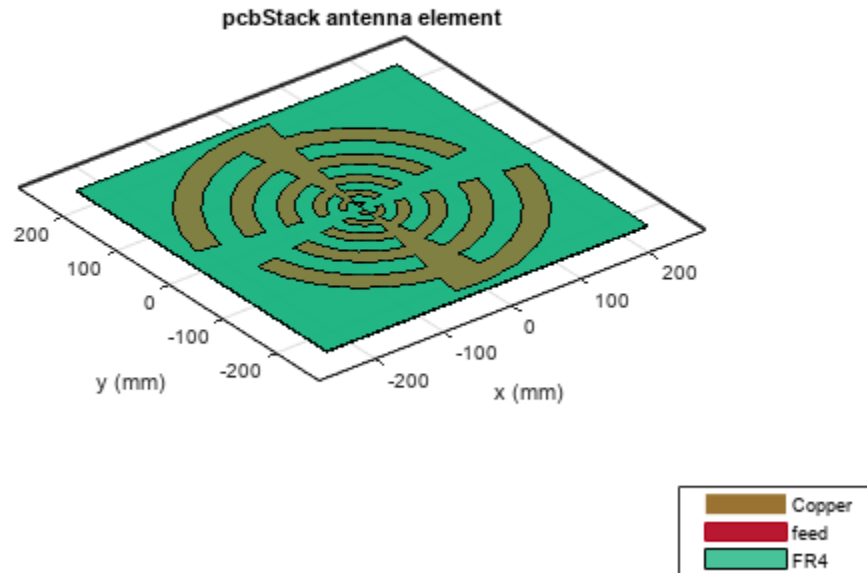
Create Log-Periodic Antenna Using PCB Stack

Create a `pcbStack` with two layers. The top metal layer is the log-periodic shape antenna and the bottom layer is the square-shaped dielectric substrate FR4 of length 470 mm. The substrate has a dielectric constant of 4.6, an attenuation coefficient of 0.02, and the substrate thickness of 1.6 mm. The top conductor is set to copper with a thickness of 35 μm .

```
ant          = pcbStack;
d            = dielectric('FR4');
d.Thickness = 1.6e-3;
ant.BoardThickness = 1.6e-3;
BoardDim    = antenna.Rectangle('Length',470e-3,'Width',470e-3);
ant.Name    = 'LogPeriodic';
ant.Layers  = {shape,d};
ant.BoardShape = BoardDim;
ant.FeedLocations = [0 0 1];
ant.FeedDiameter = 0.25e-3;
ant.FeedViaModel = 'strip';
ant.Conductor = metal('copper');
ant.Conductor.Thickness = 35e-6;
```

Use the `show` function to visualize the antenna.

```
figure;
show(ant)
```

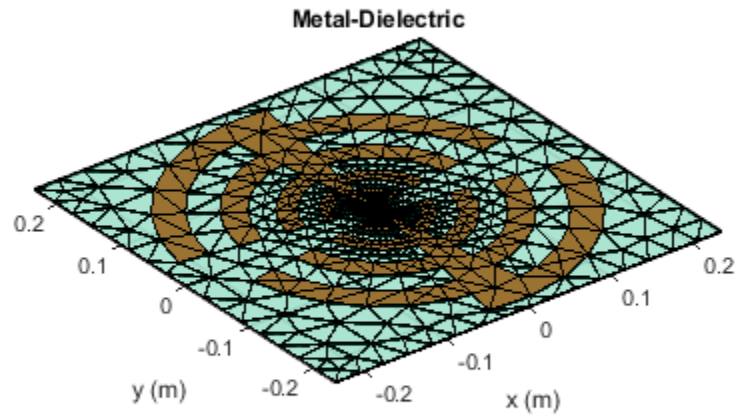


Meshing Antenna

Mesh the structure by specifying the maximum edge length. Below is the mesh used to model the antenna. The triangles are used to discretize the metal regions of the patch, and tetrahedra are used to discretize the volume of the dielectric substrate in the antenna. The triangles and tetrahedra are indicated by the colors yellow and green, respectively. The total number of unknowns is the sum of the unknowns for the metal plus the unknowns used for the dielectric. As a result, the time to calculate the solution increases significantly as compared to pure metal antennas. Set the maximum edge length to 0.05. This value can be decreased to generate a denser mesh.

```
figure;  
mesh(ant, 'MaxEdgeLength', 0.05)
```

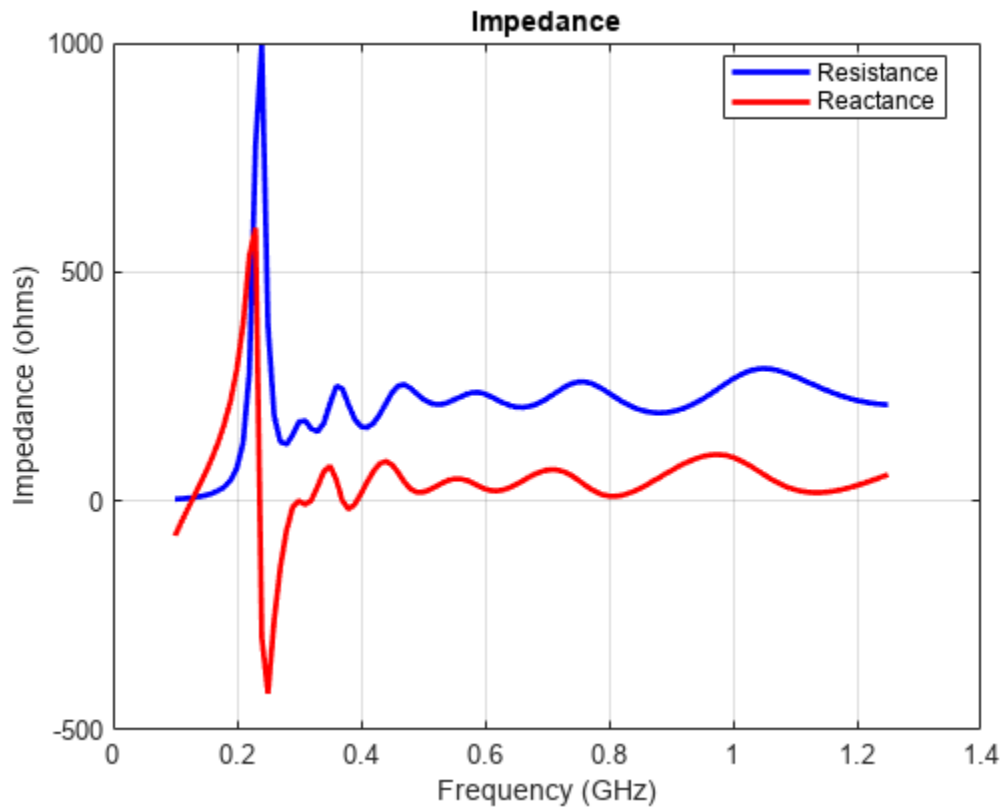
NumTriangles: 466
NumTetrahedra: 3426
NumBasis:
MaxEdgeLength: 0.05
MeshMode: manual



Analyze Antenna

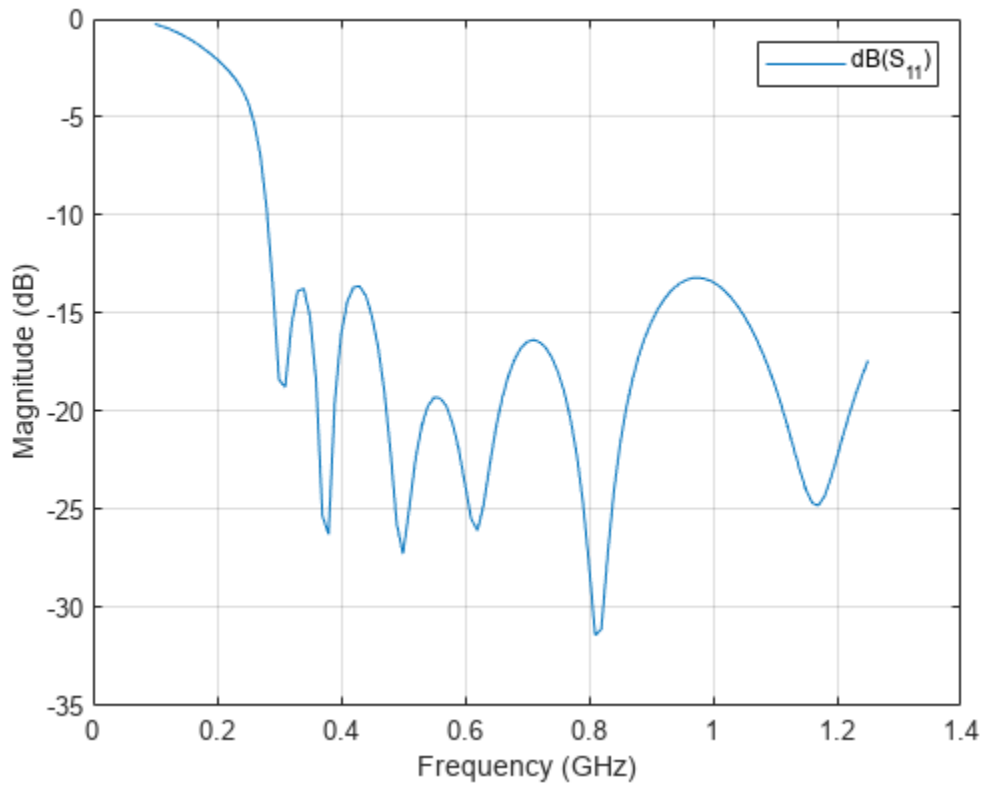
Plot the impedance of the antenna using the `impedance` function over a frequency range of 100-1250 MHz.

```
figure;  
impedance(ant,100e6:10e6:1250e6);
```

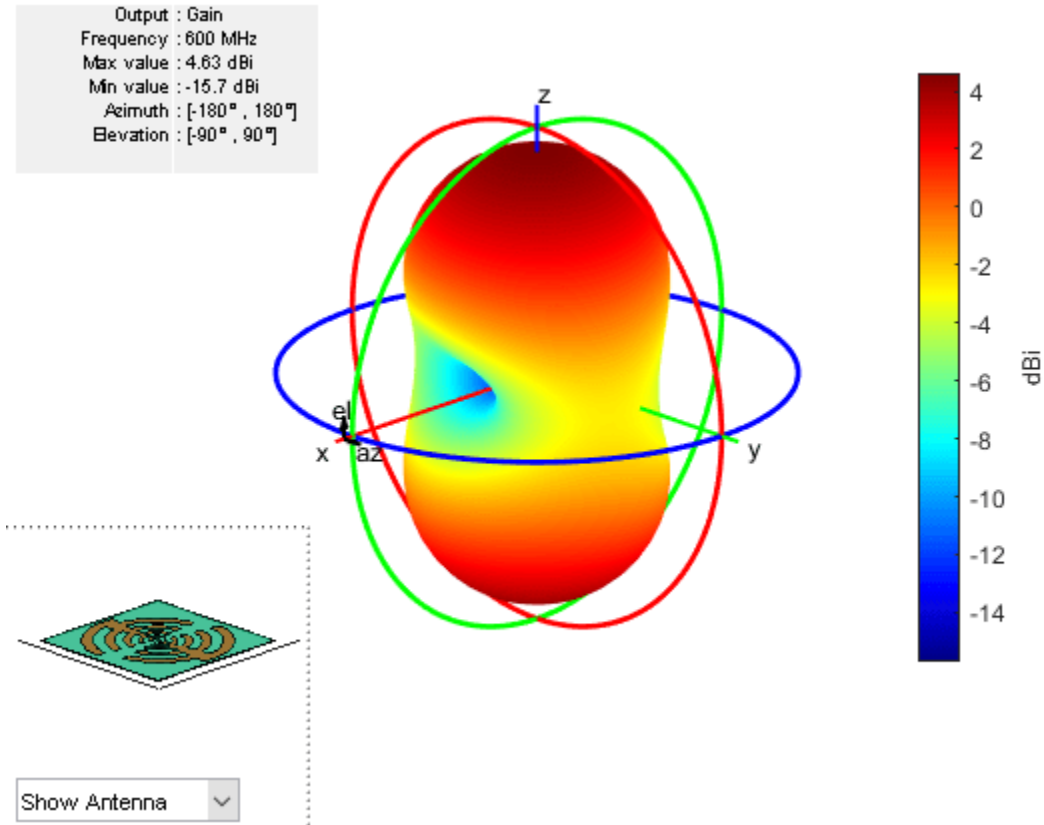
The impedance plot shows that real part of the impedance is approximately 220 ohms over the specified range. And the imaginary part of the impedance is almost 0. The results in [1] are obtained by using a balun to transform the impedance to 50 ohms. Set the reference impedance as 220 ohms to obtain results similar to [1]. Plot S-parameters over a frequency 300-1250MHz with the reference impedance of 220 ohms using the `sparameters` function. The antenna has an S11 of value less than -10 dB over a frequency range of 300-1200 MHz.

```
spar = sparameters(ant,100e6:10e6:1250e6,220);
figure;
rfplot(spar)
```



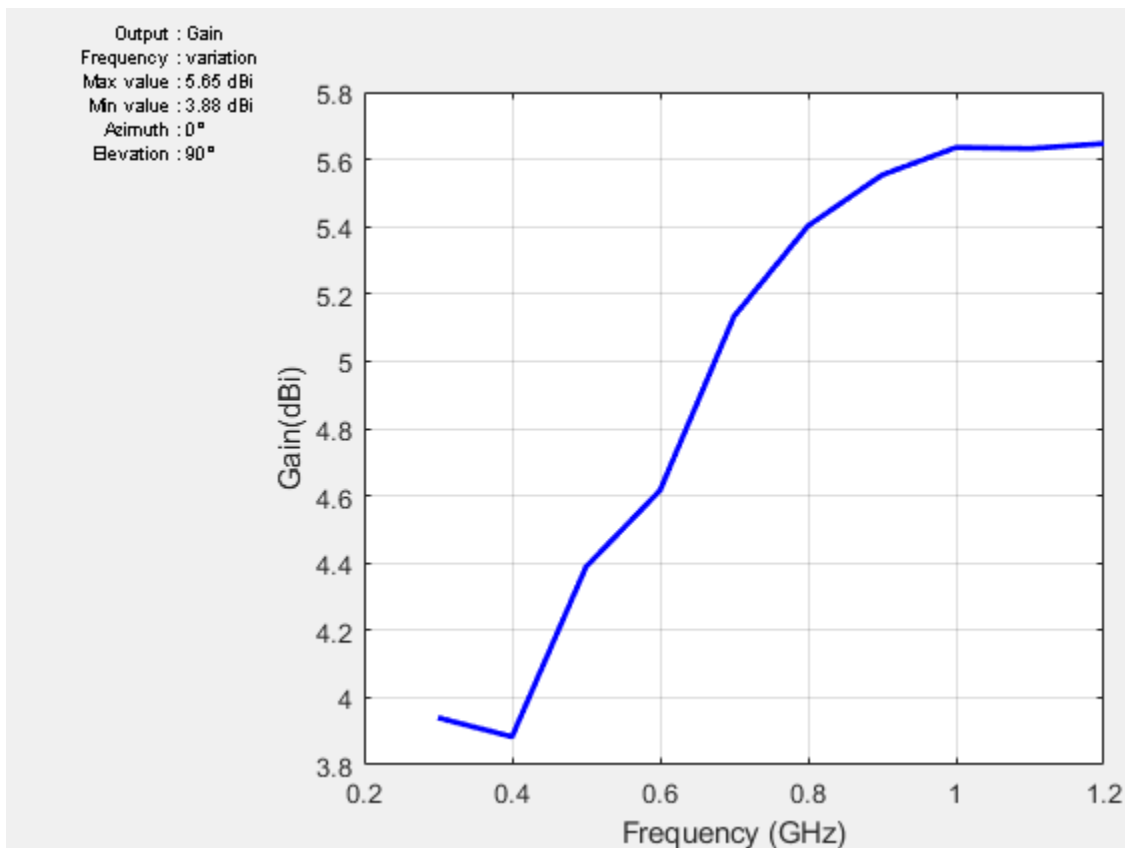
Use the pattern function to plot the radiation pattern of the antenna at a frequency of 600 MHz.

```
figure;  
pattern(ant,600e6)
```



Plot the radiation pattern of the antenna in the rectangular cartesian coordinate system over a frequency range of 300-1200 MHz

```
figure;  
pattern(ant,300e6:100e6:1200e6,0,90,'CoordinateSystem','rectangular')
```



Conclusion

Modelling and analysis of a log-periodic antenna using the Antenna Toolbox is completed and its performance matches the results given in [1]. The return loss (S11) of the antenna indicates a wideband performance over a frequency range of 300-1200 MHz. The antenna has an approximate gain of 4.94 dBi at 600 MHz and the gain increases with the increase in frequency.

Reference

[1]. Phu B. H., Quang P. M., Phuoc D. T. and Duy L. N., "A log-periodic saw-toothed planar antenna for UHF ultra-wideband applications," 2013 International Conference on Advanced Technologies for Communications, 2013, pp. 689-692.

Calculate Radiation Efficiency of Antenna

This example shows you how to calculate the radiation efficiency of an antenna or antenna array from the Antenna Toolbox™. The radiation efficiency of an antenna is defined as the ratio of the power radiated by an antenna to the power fed to the excitation port of the antenna. The power loss due to port impedance mismatch is not considered here.

The input power to the antenna can be written as

$$P_{in} = \frac{1}{2} V_{in} I_{in}^* \quad (1)$$

Here, the input voltage and input current are represented by V_{in} and I_{in} , respectively. I_{in}^* is the complex conjugation of the input current. Calculate the power P_{rad} radiated by the antenna by integrating the radiation intensity ($U(\theta, \phi)$) over the infinite radiation sphere (S_∞)

$$P_{rad} = \int_{S_\infty} U(\theta, \phi) \sin\theta d\theta d\phi \quad (2)$$

The azimuth and elevation angles are denoted by ϕ and θ , respectively. The radiation efficiency (η_r) is defined as

$$\eta_r = \frac{P_{rad}}{P_{in}} \quad (3)$$

The difference between the input power and the radiated power is due to conduction loss in the metal-only antennas and due to both conduction loss and dielectric loss in the metal-dielectric antennas. The radiation efficiency is also alternately defined as the gain and directivity of the antenna. In other words,

$$G(\theta, \phi) = \eta_r D(\theta, \phi) \quad (4)$$

In an ideal lossless antenna, the radiation efficiency (η_r) is 1.

Metal-Only Antenna

This example uses a Yagi-Uda antenna with the same dimensions as given in [1] on page 5-773.

Create Geometry

Create the geometry of the Yagi-Uda antenna with two director elements of the length 131.9 mm and 126.5 mm, respectively. The spacing dimensions for the directors are 65.95 mm and 80.34 mm. The length of the exciter is 139.1 mm. The length and spacing values of the reflector are 141.5 mm and 88.13 mm, respectively. In [1], all the elements are thin wires with the radius of 0.6745 mm. However, this example uses the `cylinder2strip` function to model equivalent strips.

```
d=design(dipole,1e9);
radius=6.7450e-04; %Radius of thin wires
d.Width=cylinder2strip(radius); %Converting into equivalent stripwidth
d.Length=139.1e-03;
d.TiltAxis=[0 1 0];
d.Tilt=-90;
ant=design(yagiUda,1e9);
```

```

ant.Exciter=d;
ant.NumDirectors=2;
ant.DirectorLength=[131.9e-03;126.5e-03];
ant.DirectorSpacing=[65.95e-03;80.34e-03];
ant.ReflectorLength=141.5e-03;
ant.ReflectorSpacing=88.13e-03;

```

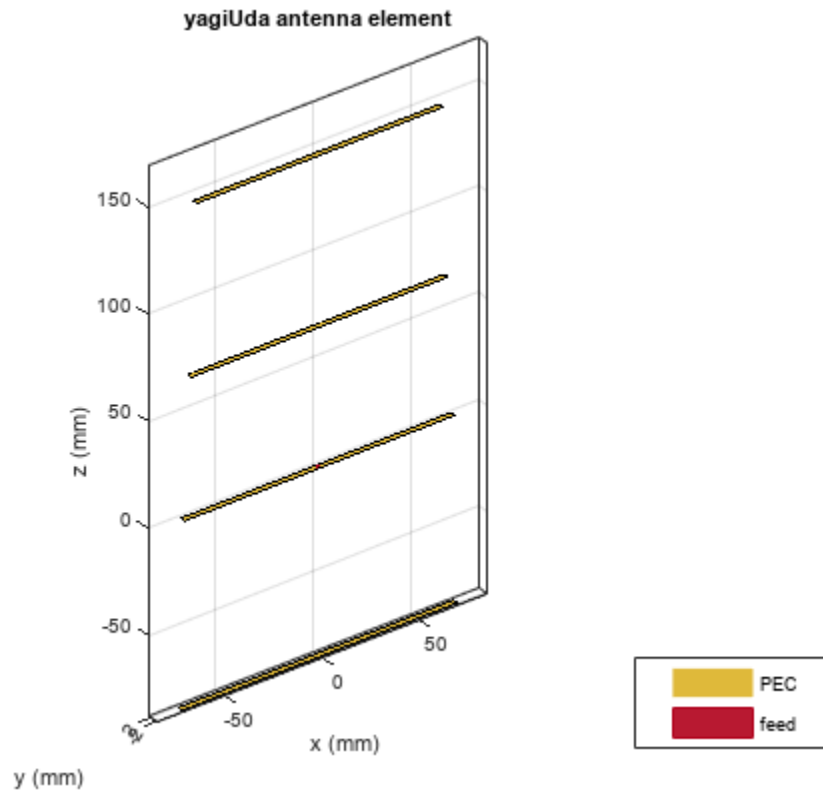
Visualize Antenna

Visualize the antenna, which is a default perfect electrical conductor (PEC) antenna with conductivity and thickness values of infinity and zero, respectively.

```

figure;
show(ant)

```



Using `show` function provides the name of the conductor and the feed location using different colors in the above figure.

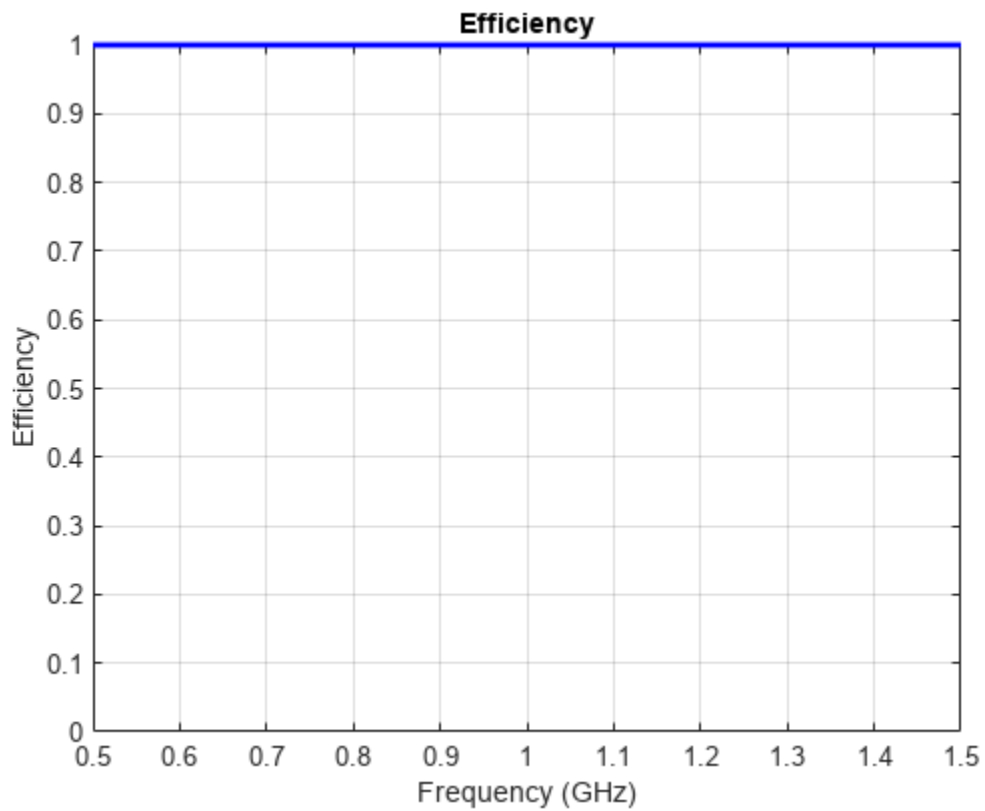
Visualize Radiation Efficiency of PEC Antenna

Plot the radiation efficiency of the PEC yagi-uda antenna using the `efficiency` function within the frequency range of 0.5-1.5 GHz with 31 sampling points. As the PEC antenna has no loss, it shows a radiation efficiency of 1 over the specified frequency range.

```

f=linspace(0.5e9, 1.5e9, 31);
efficiency(ant,f)

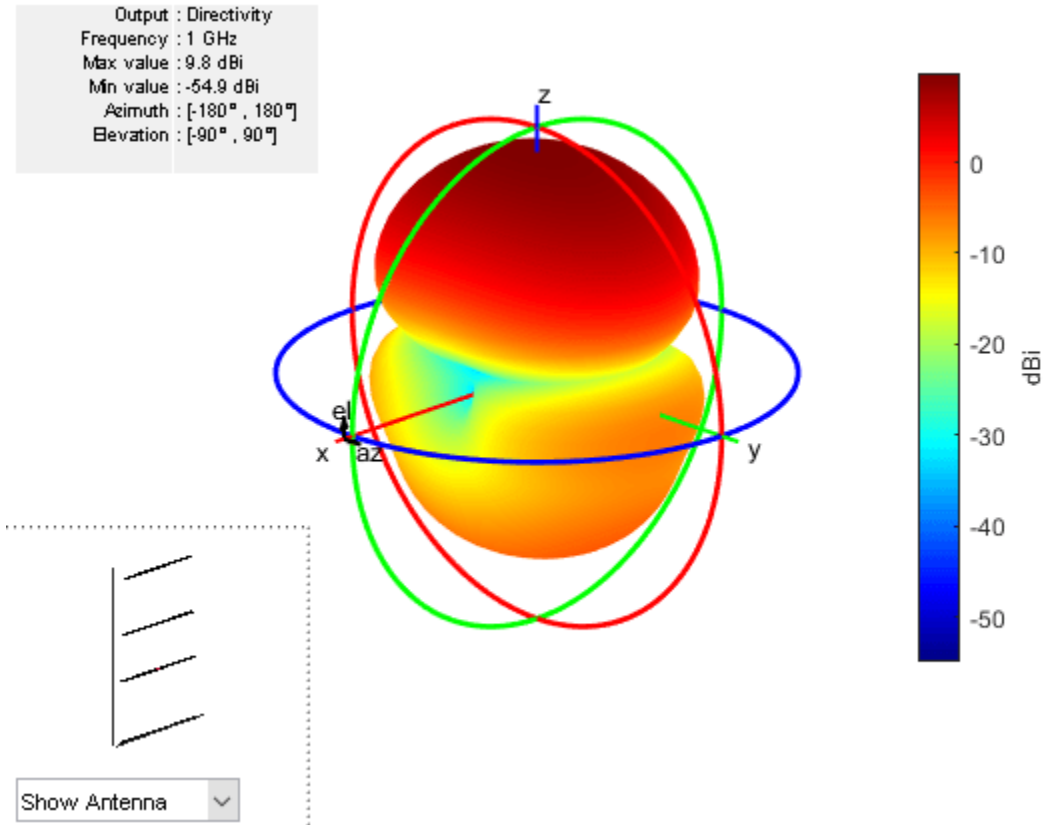
```



Visualize Directivity of PEC antenna

Calculate the directivity of the PEC Yagi-Uda antenna at a frequency of 1 GHz. Due to the absence of loss, the directivity and gain will be the same in the PEC antenna.

```
figure;  
pattern(ant, 1e9)
```



Use Copper in Antenna Design

Set the conductor as copper from the metal catalog of the Antenna Toolbox. Modify the conductivity of the finite metal Yagi-Uda antenna using the properties of the metal object.

Use Copper Metal for Antenna Design

```
ant.Exciter.Conductor = metal('Copper');%Choose conductor from the metal catalog
ant.Exciter.Conductor.Conductivity = 1e5;%Same value used in the reference paper
```

Specify Metal Properties of Conductor

```
ant.Conductor = metal('Copper');%Choose the conductor from the metal catalog
ant.Conductor.Conductivity = 1e5;%same value used in the reference paper
```

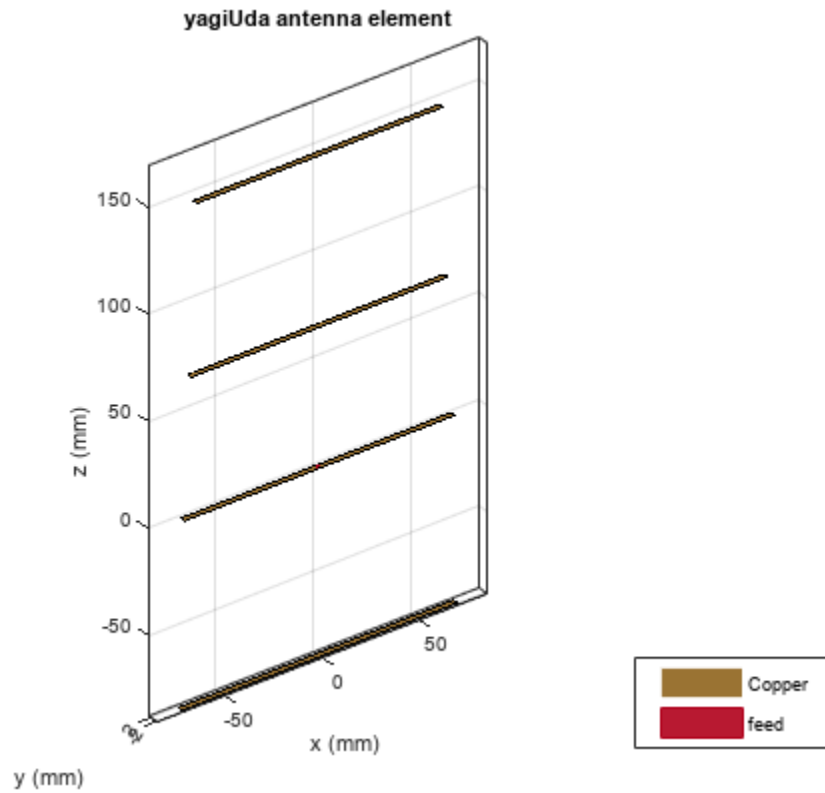
Modify the conductivity and thickness of the finite metal Yagi-Uda antenna.

```
ant.Exciter.Conductor.Thickness = 700*1e-6;
ant.Conductor.Thickness = 700*1e-6;
```

Visualize Finite Metallic Antenna

Visualize the metallic Yagi-Uda antenna using the show function.

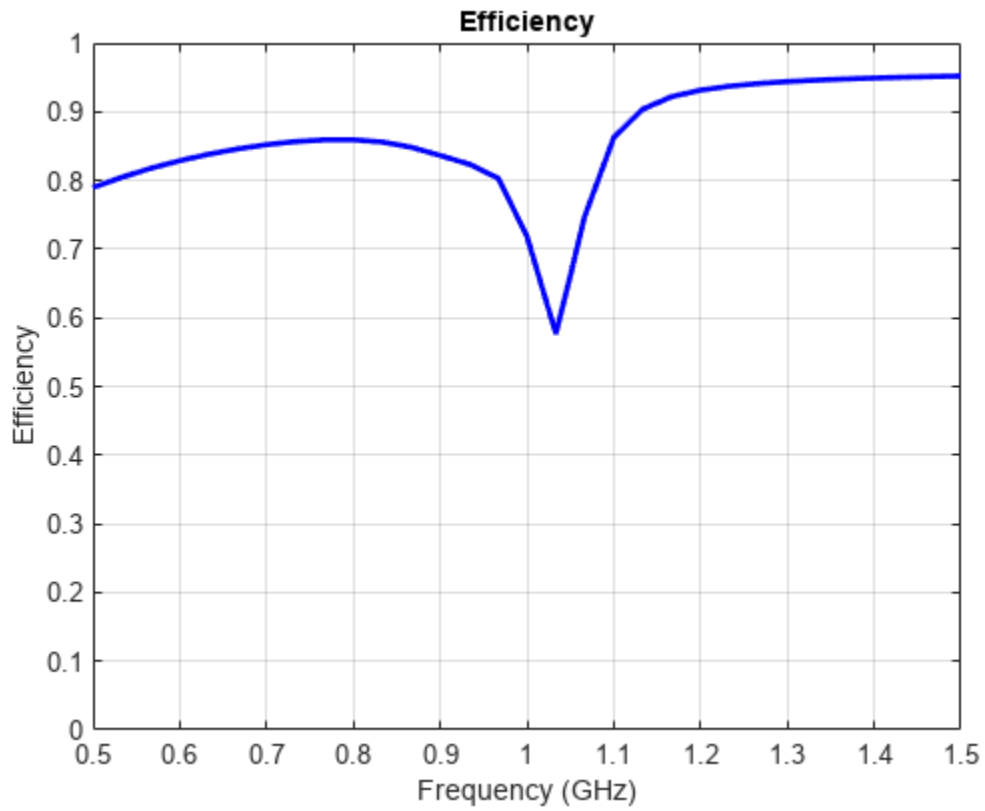
```
figure;
show(ant)
```

Plot Radiation Efficiency of Finite Metallic Antenna

Plot the radiation efficiency visualization of the metallic Yagi-Uda antenna at the frequency range of 0.5-1.5 GHz.

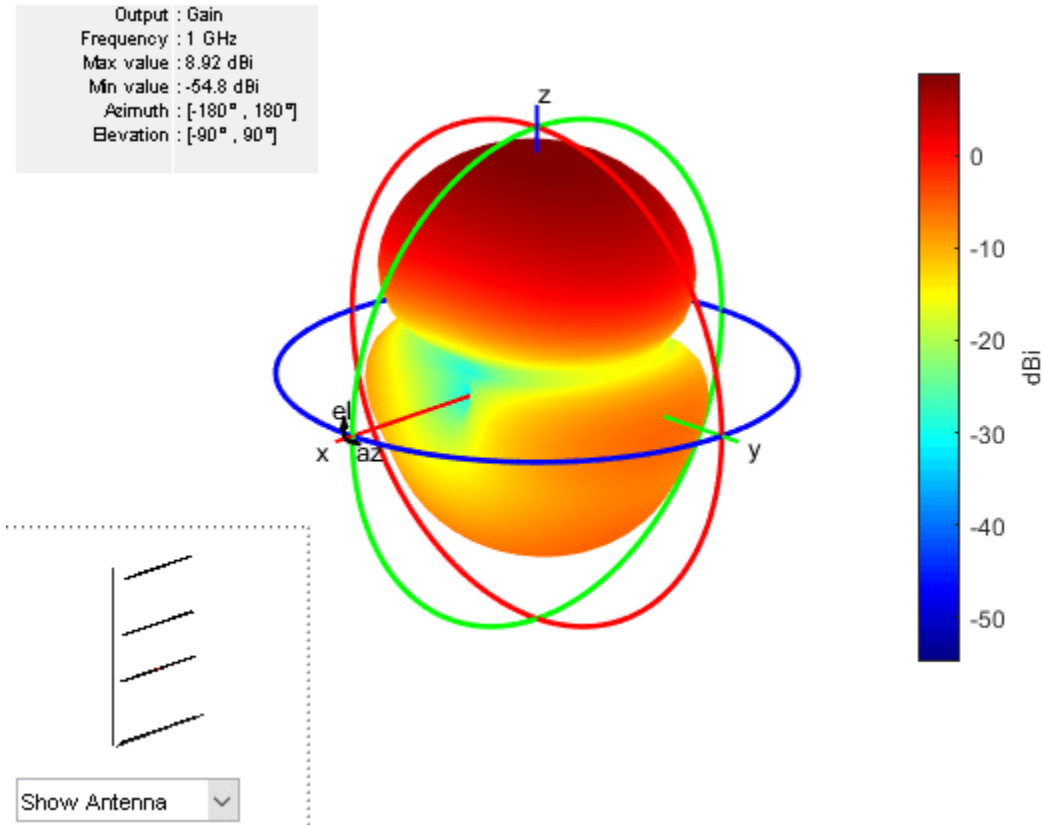
```
f = linspace(0.5e9, 1.5e9, 31);  
figure;  
efficiency(ant,f)
```



Plot Gain Pattern of Finite Metallic Antenna

Plot the gain pattern of the metallic Yagi-Uda antenna at a frequency of 1 GHz.

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



This shows that the gain of the antenna is reduced by 0.86 dB due to finite conduction loss. The efficiency value closely matches the analytical results in [1] on page 5-773.

Metal-Dielectric Antenna

Create a microstrip patch antenna from [2] on page 5-773. The numerical analysis in [2] uses the finite-difference time-domain (FDTD) technique. However, this example uses the method of moments (MoM) solver from the Antenna Toolbox to analyze the antenna.

Create Geometry

Create the geometry of the microstrip patch antenna with a PEC conductor and lossy substrate of 1.57 mm thickness.

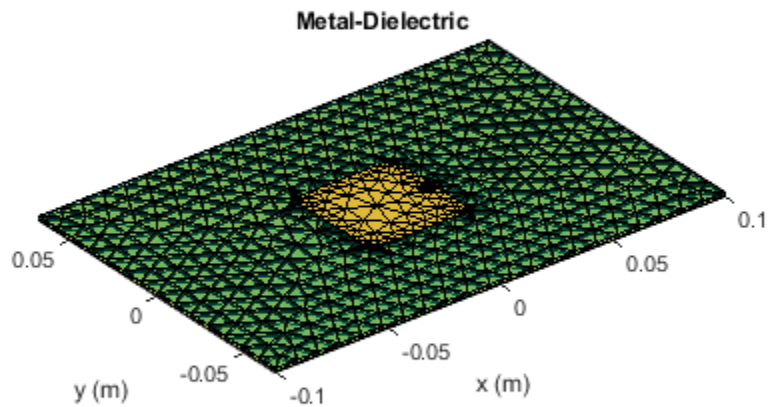
```
f = 1.59e9; %Solution frequency
lambda = 3e8/f;
d = dielectric('FR4'); %Selecting the substrate from the dielectric catalog
d.EpsilonR = 4.36;
d.LossTangent = 2/100;%Indicates the lossy substrate
ant = patchMicrostrip('Substrate',d);
ant.Height = 1.57e-3;
ant.Substrate.Thickness = 1.57e-3;
ant.Length = 45e-3;
ant.Width = 45e-3;
ant.GroundPlaneLength = 20e-2;
ant.GroundPlaneWidth = 13.5e-2;
ant.FeedOffset = [20e-3 0];
ant.FeedWidth = lambda/200;
```

Manual Meshing of Microstrip Patch Antenna

Mesh the antenna by using the maximum edgelenhth of the RWG basis functions as $\lambda/20$ where the free-space wavelength at the solution frequency of 1.5 GHz is λ .

```
figure;  
mesh(ant, 'MaxEdgeLength', lambda/20)
```

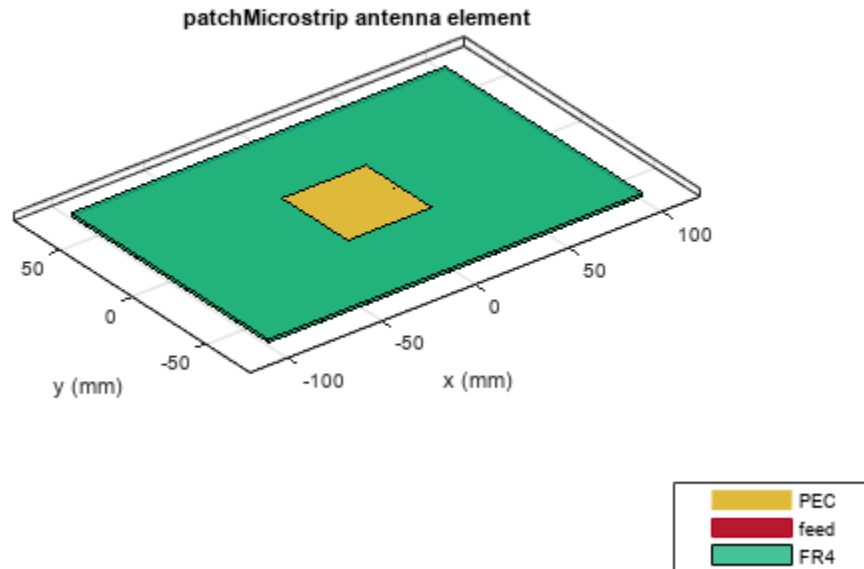
```
NumTriangles: 1232  
NumTetrahedra: 2940  
NumBasis:  
MaxEdgeLength: 0.009434  
MeshMode: manual
```



Visualize Microstrip Patch Antenna

Visualize the antenna.

```
figure;  
show(ant)
```



Calculate Radiation Efficiency of Microstrip Antenna with PEC Metal and Lossy Substrate

Calculate the radiation efficiency in absolute and logarithmic values. As the metal is PEC, the resultant loss is due to the lossy substrate.

```
E1 = efficiency(ant,f)
```

```
E1 = 0.3116
```

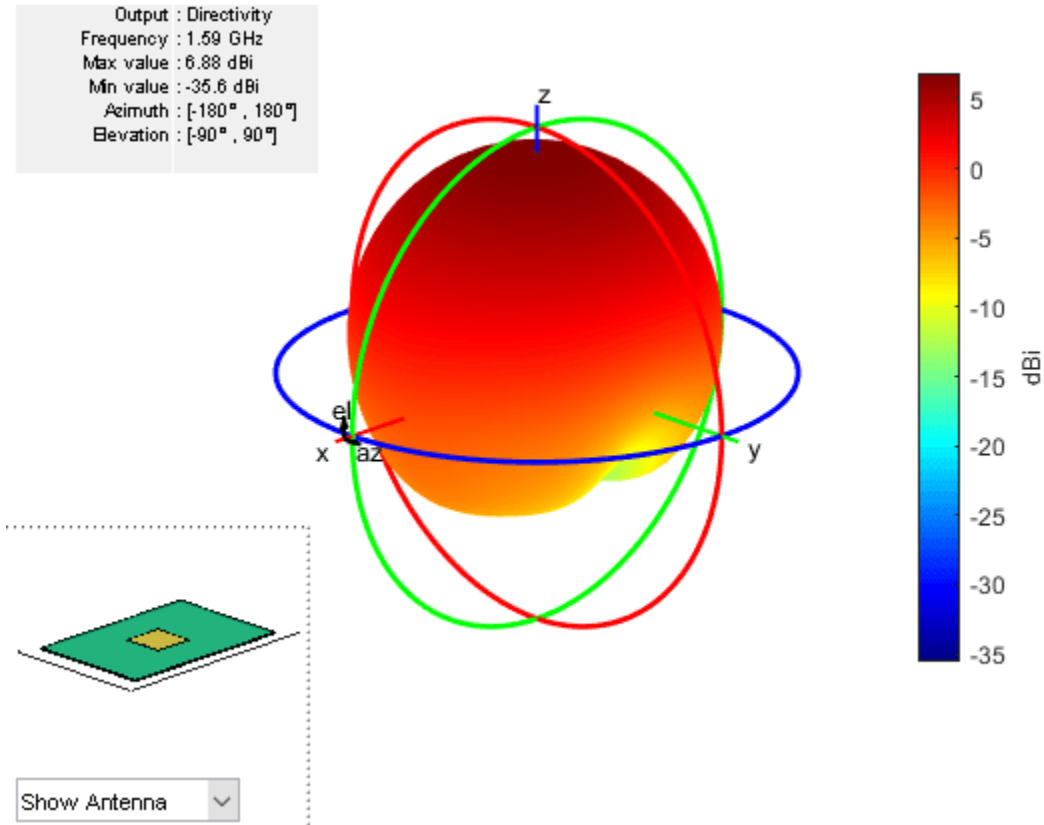
```
E1_log = 10*log10(E1)
```

```
E1_log = -5.0647
```

Plot Directivity of Microstrip Antenna with PEC Metal and Lossy Substrate

Plot the directivity of the microstrip antenna using `pattern` function. The directivity does not depend on the conduction and dielectric losses.

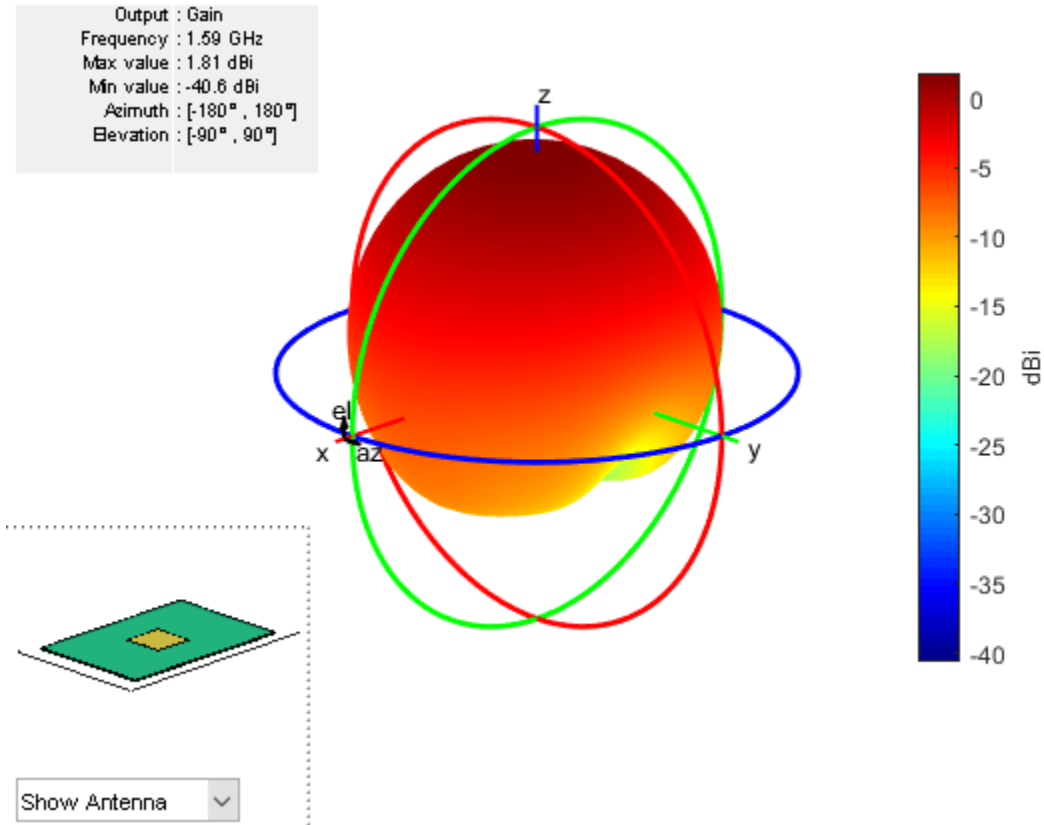
```
figure
pattern(ant,f,'Type','Directivity')
```



Plot Gain of Microstrip Antenna with PEC Metal and Lossy Substrate

Plot the gain of the microstrip antenna with PEC metal and lossy substrate. The gain value is less than the directivity value due to the dielectric loss. The difference in the gain and directivity values matches closely with the log value of the radiation efficiency E_{1log} .

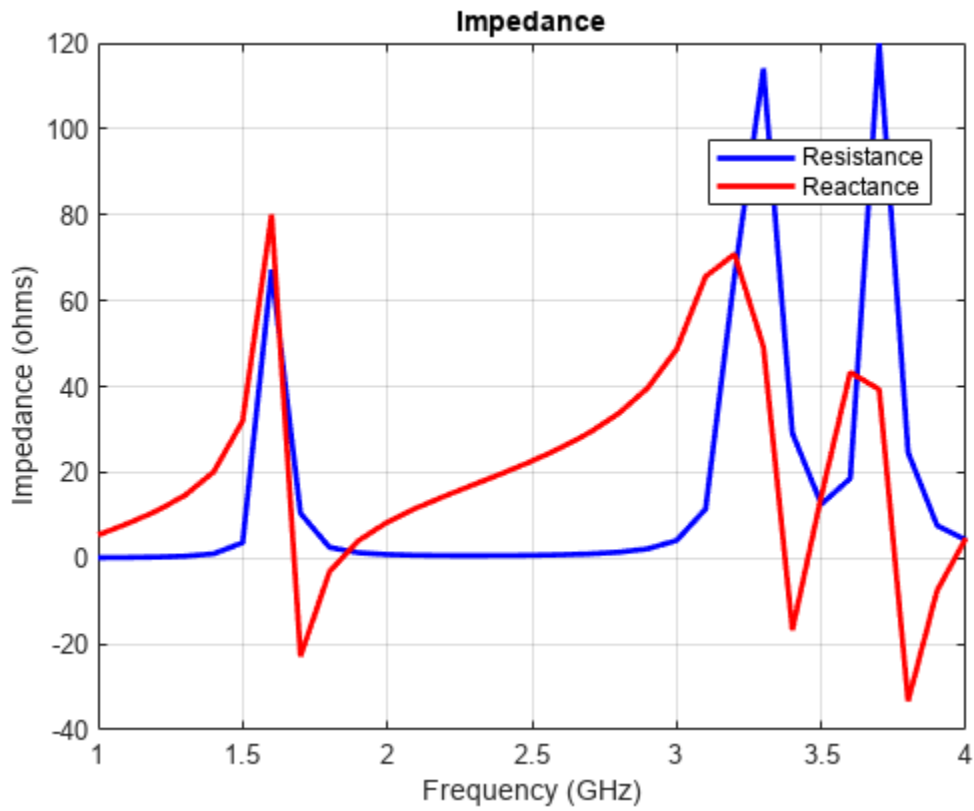
```
figure;  
pattern(ant,f,'Type','Gain')
```



Impedance of Microstrip Antenna with PEC Metal and Lossy Substrate

Plot the impedance variation of the microstrip antenna with PEC metal and lossy substrate in the frequency range of 1-4 GHz.

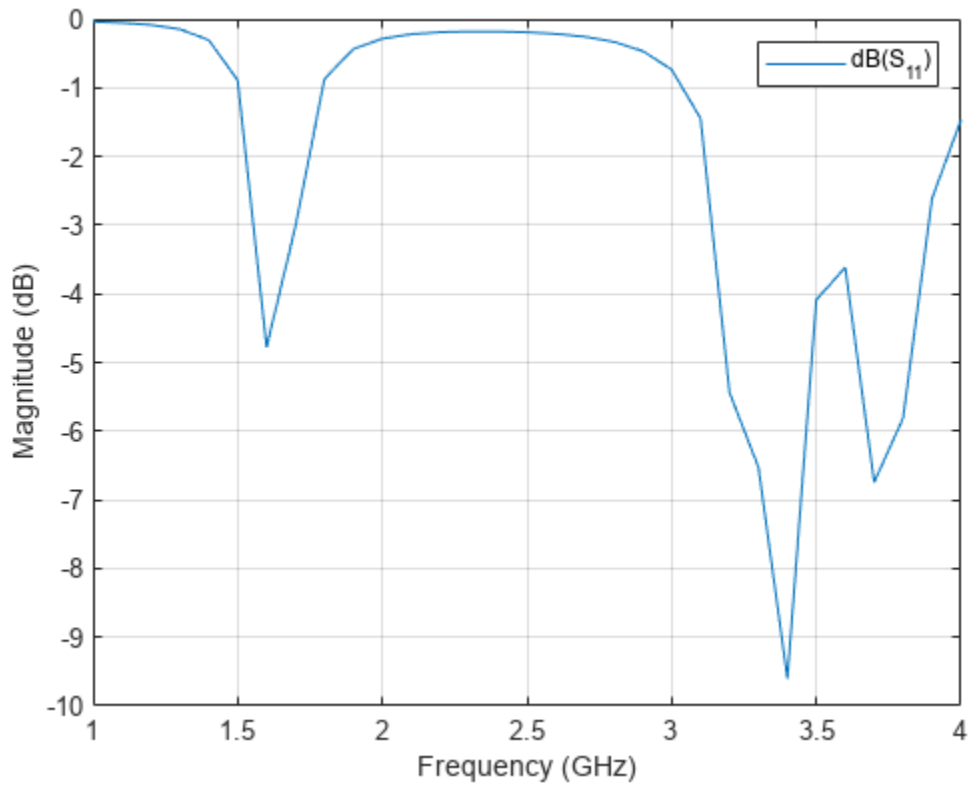
```
f1 = linspace(1e9,4e9,31);  
figure  
impedance(ant,f1)
```



Return Loss of Microstrip Antenna with PEC Metal and Lossy Substrate

Plot the return loss variation of the microstrip antenna with PEC metal and lossy substrate.

```
figure;  
s1 = sparameters(ant,f1,50);  
rfplot(s1);
```

Change Conductor Properties of Microstrip Antenna

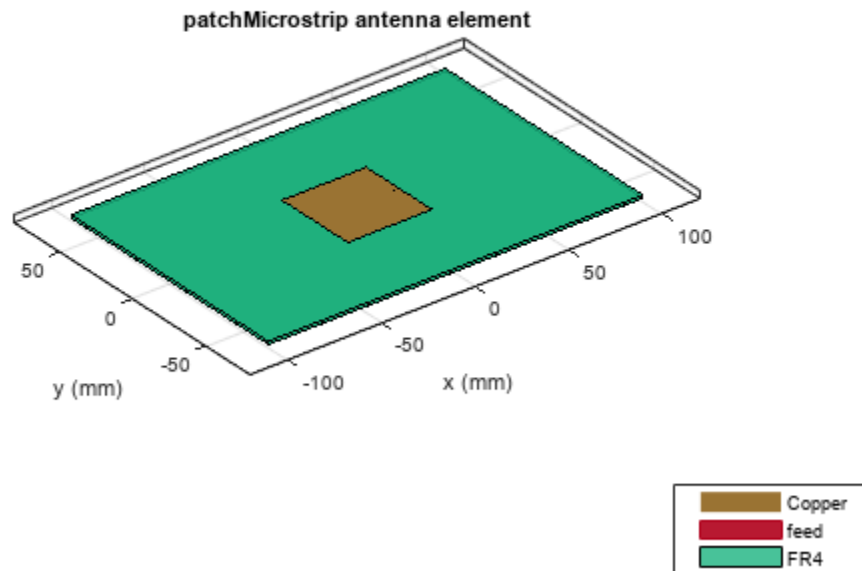
Set conductor of antenna as a lossy metal. Use metal object to change conductor to copper.

```
ant.Conductor = metal('Copper');
```

Visualize Microstrip Antenna with Lossy Metal and Lossy Substrate

Visualize of the microstrip antenna with a copper conductor and FR4 substrate.

```
figure;  
show(ant)
```



Calculate Radiation Efficiency of Microstrip Antenna with Lossy Metal and Lossy Substrate

Calculate the radiation efficiency in absolute and logarithmic values. The radiation efficiency is lower due to conduction loss in addition to the dielectric loss.

```
E2 = efficiency(ant,f)
```

```
E2 = 0.2255
```

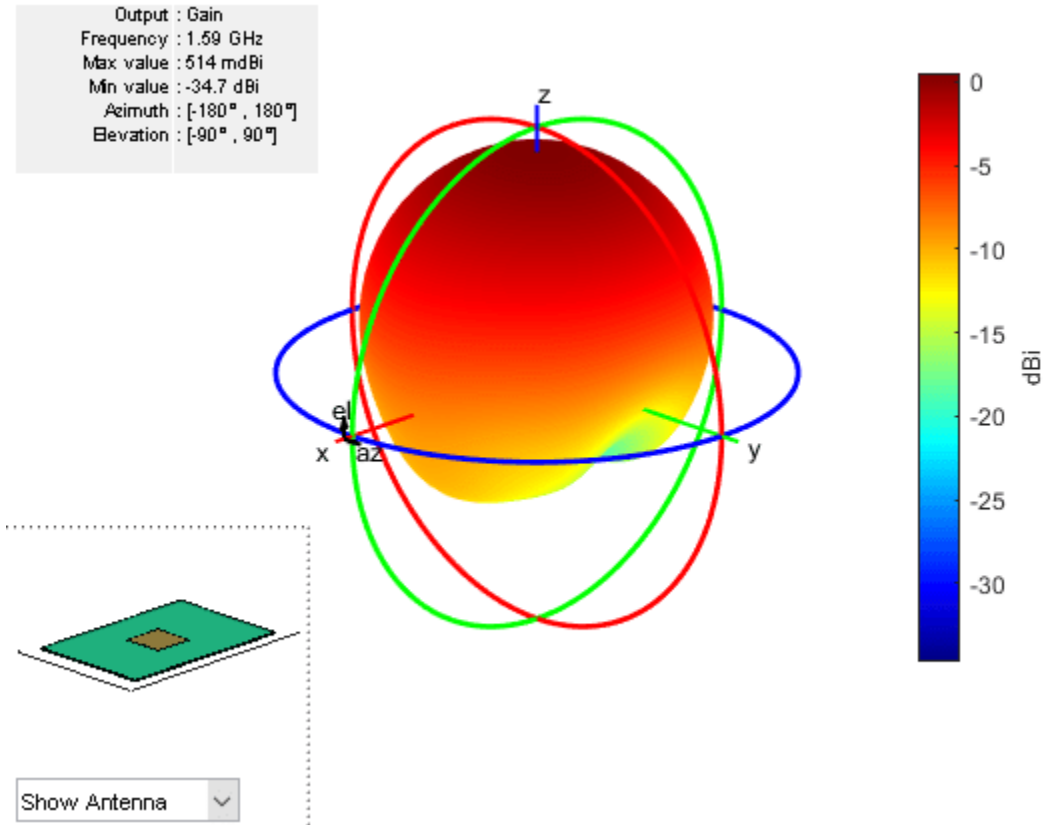
```
E2_log = 10*log10(E2)
```

```
E2_log = -6.4692
```

Plot Gain of Microstrip Antenna with Lossy Metal and Lossy Substrate

Plot the gain of the microstrip antenna with a copper conductor and FR4 substrate.

```
figure;  
pattern(ant,f,'Type','Gain')
```



The gain value is less than the directivity value due to both the conduction loss and the dielectric losses. The difference in the gain and directivity matches closely with the log value of the radiation efficiency $E2_{log}$.

Conclusion

The radiation efficiencies computed using the Antenna Toolbox for both the metallic and metal-dielectric antennas are found to match closely as reported in the references that use different analytic [1] or numerical techniques [2].

References

[1] Shahpari, Morteza, and David V. Thiel. "Fundamental limitations for antenna radiation efficiency," IEEE Transactions on Antennas and Propagation, Vol. 66, No. 8, 2018.

[2] Ph. Leveque, A. Reineix and B. Jecko, "Modelling of Dielectric Losses in Microstrip Patch Antennas: Application of FDTD Method", Electronics Letters, Vol. 28, No. 6, March 1992.

Design 77 GHz Patch Microstrip for Automotive Radar Receiver

This example shows how to create, model, and analyze an inset-fed patch microstrip antenna at high frequencies. As the frequency of operation increases to millimeter waves, the antenna sizes decrease and the antennas are fabricated on printed circuit boards (PCBs). Such printed antennas are of light weight, are inexpensive, easy to integrate, and are widely used as components in a radar. The antenna designed in this example operates at a frequency of 77 GHz and is used in an automotive radar receiver.

Define Parameters

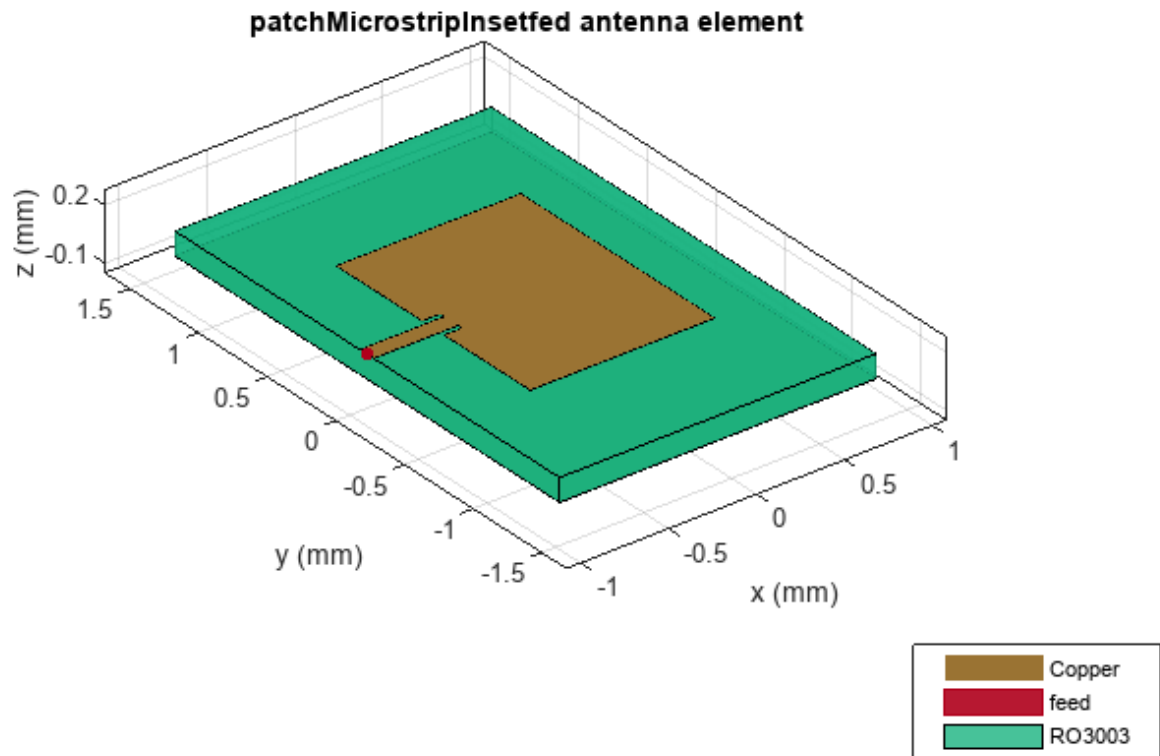
The length and the width of the patch are as provided in [1] on page 5-779. The antenna is designed on a Rogers RO3003™ substrate, with the dielectric constant of 3, loss tangent of 0.0013, and 130 μm thickness. A copper conductor of thickness 17 μm is used as the patch.

```
PL = 1049e-6;  
PW = 1412e-6;  
sub = dielectric(Name='RO3003',EpsilonR=3,LossTangent=0.0013,Thickness=130e-6);  
con = metal(Name='Copper',Conductivity=5.96e7,Thickness=17e-6);
```

Create Patch Microstrip Antenna

Create an inset-fed patch microstrip antenna using the `patchMicrostripInsetfed` object. The ground plane dimensions are 1800 μm by 2800 μm . The length and the width of the notch are specified as 100 μm and 160 μm , respectively. The strip line width is 100 μm . The antenna feed is located at the end of the strip.

```
ant = patchMicrostripInsetfed(Length=PL,Width=PW,Height=sub.Thickness,...  
    Substrate=sub,Conductor=con,GroundPlaneLength=1800e-6,...  
    GroundPlaneWidth=2800e-6,NotchLength=100e-6,NotchWidth=160e-6,...  
    StripLineWidth=100e-6,FeedOffset=[-900e-6 0]);  
show(ant)
```

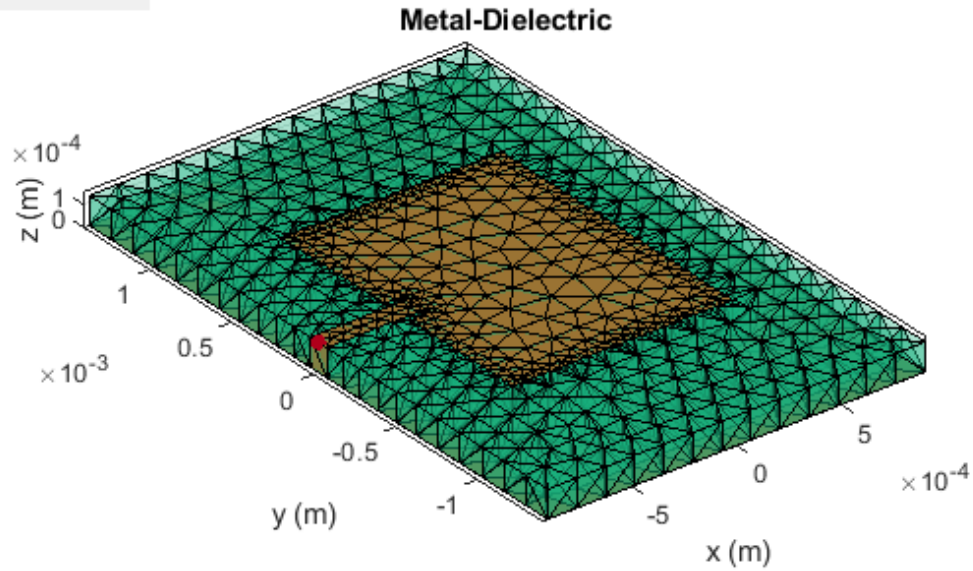


Analyze Antenna

Use the mesh function to create and display the mesh structure of the patch microstrip antenna. Mesh the antenna with a maximum edge length of $140e-6$ m.

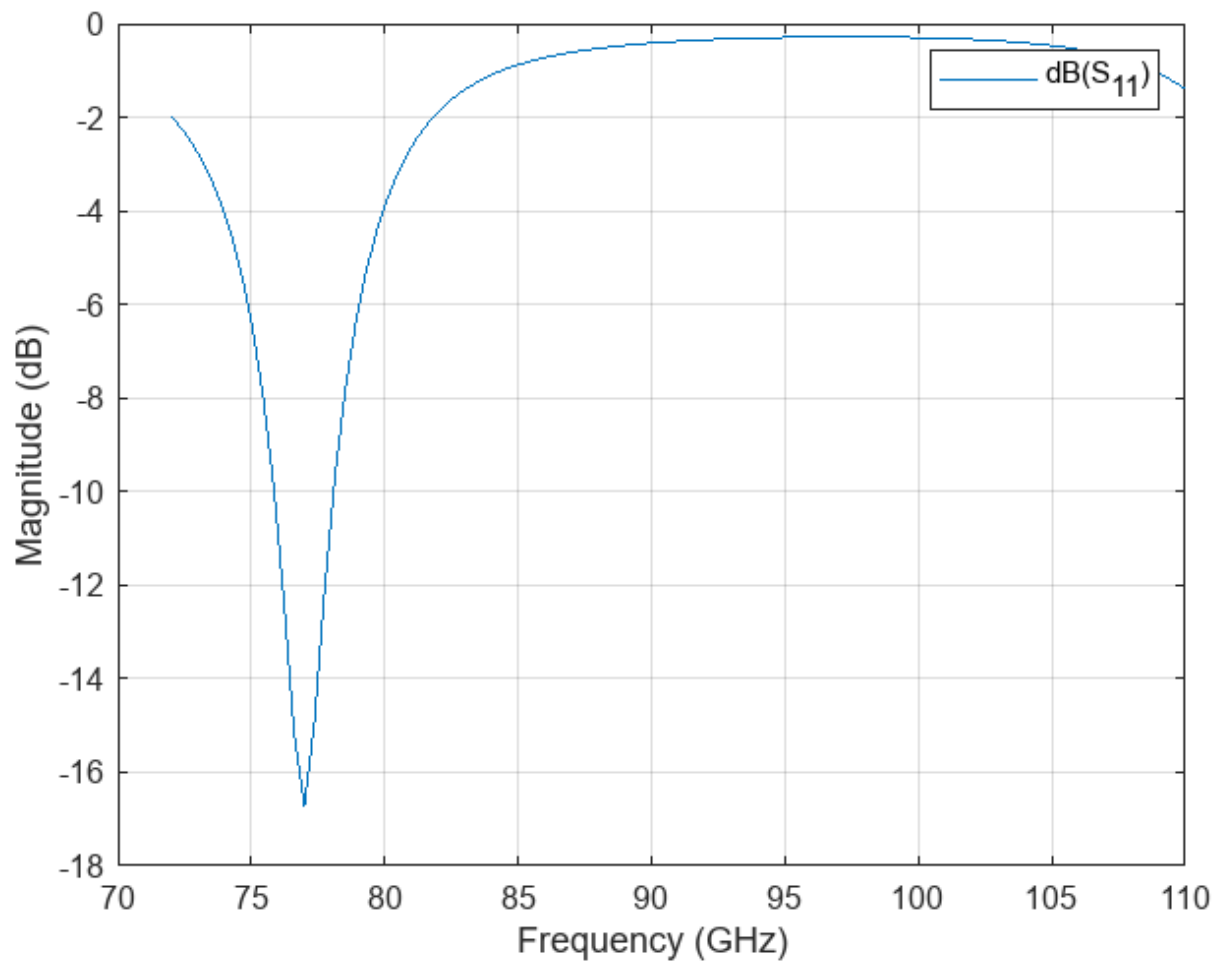
```
mesh(ant,MaxEdgeLength=140e-6)
```

```
NumTriangles: 1669  
NumTetrahedra: 3369  
NumBasis:  
MaxEdgeLength: 0.00014  
MeshMode: manual
```



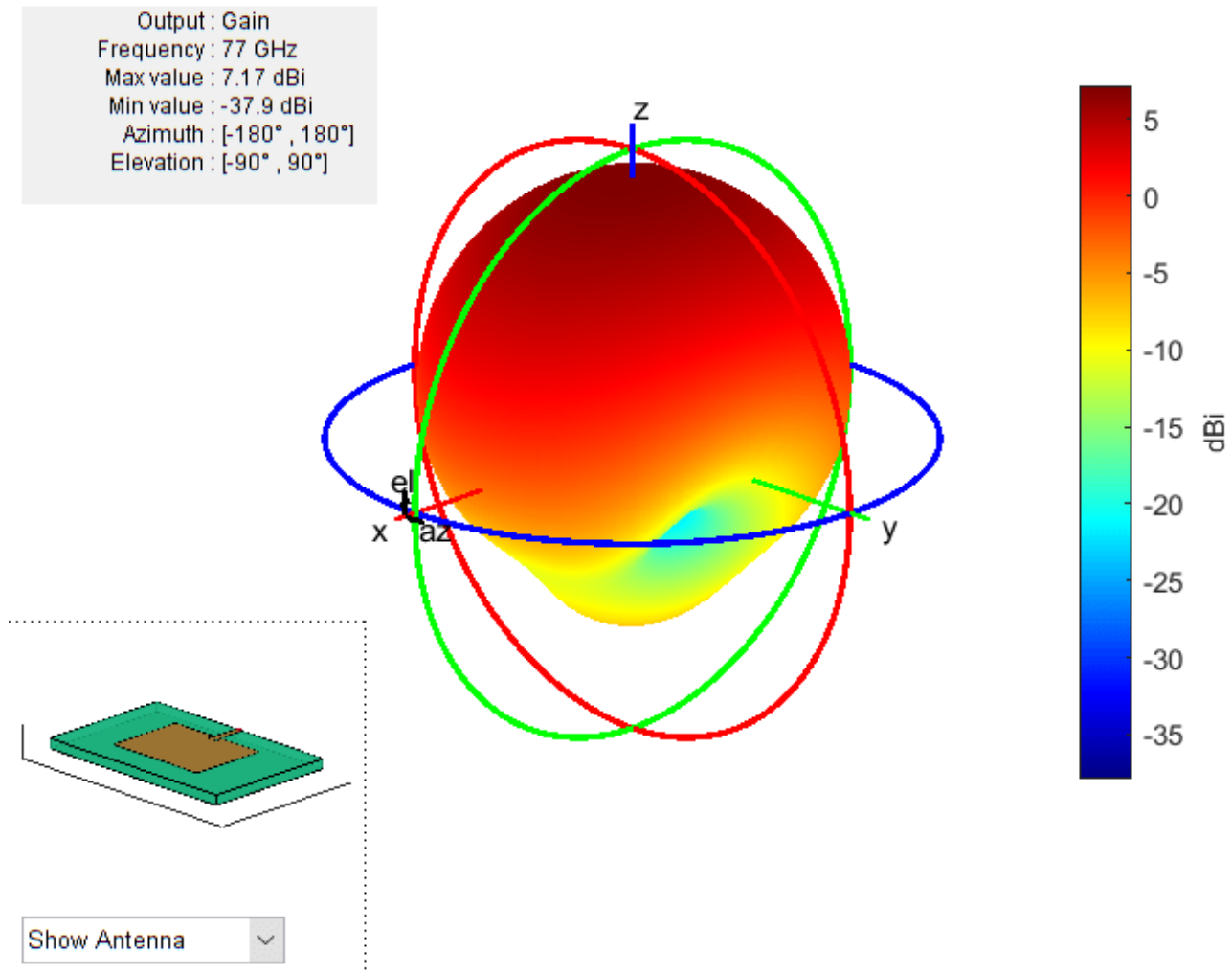
Use the `sparameters` function to plot the s-parameters of patch antenna over a frequency range of 72 -110 GHz. The antenna resonates at 77 GHz, with an approximate bandwidth of 2.7 GHz

```
sf = sparameters(ant, linspace(72e9, 110e9, 100));  
figure;  
rfplot(sf)
```



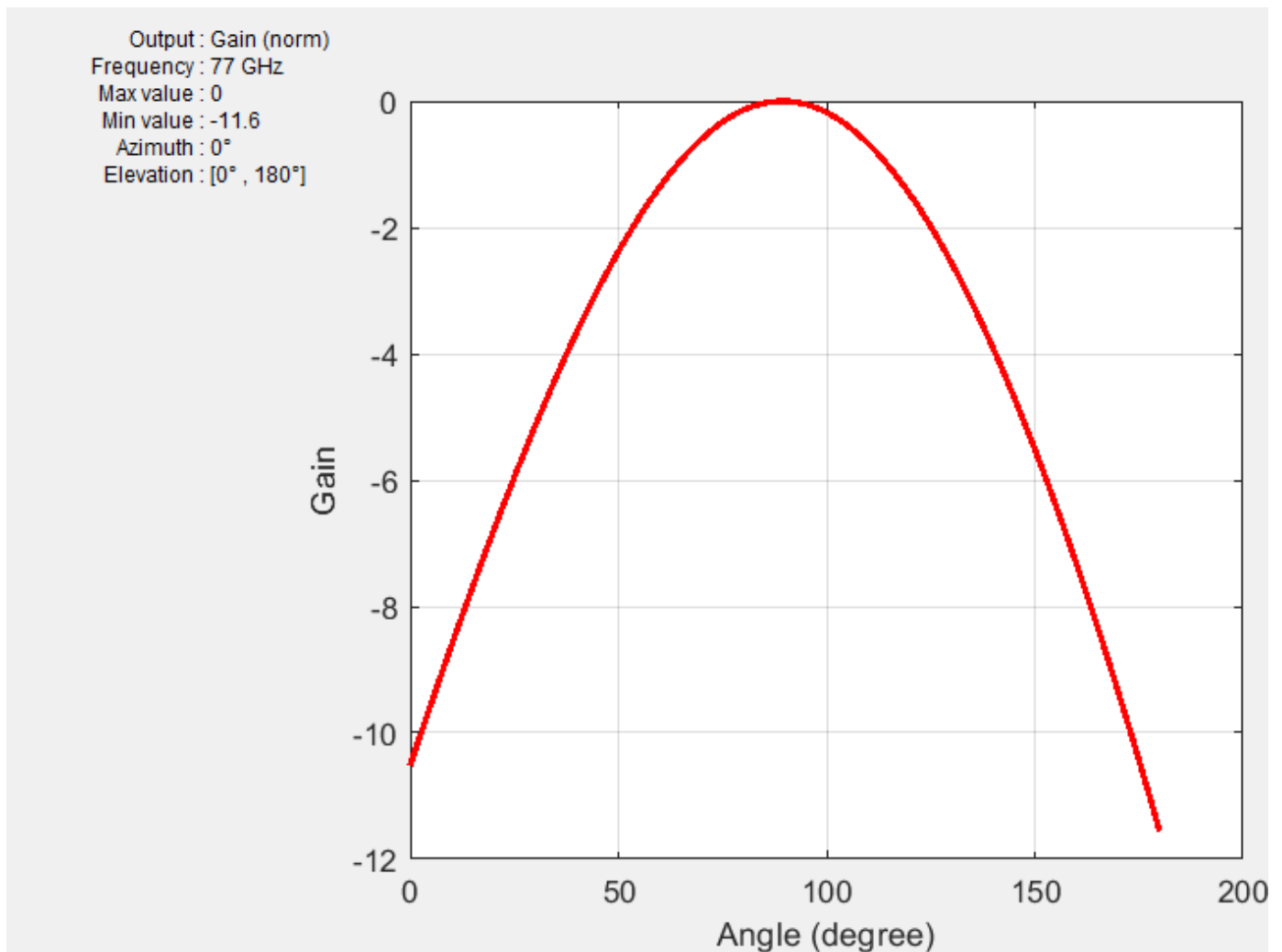
Use the `pattern` function to plot the 3-D radiation pattern of the antenna at 77 GHz. The gain obtained from the antenna is over 7 dBi.

```
pattern(ant,77e9)
```



Plot the copolarization plot in the H -plane for the antenna at 77 GHz in a rectangular coordinate system with normalization.

```
pattern(ant,77e9,0,0:1:180,CoordinateSystem='rectangular',Normalize=true)
```

Conclusion

The analysis results of the 77 GHz microstrip patch antenna show that it has good bandwidth and gain and is suitable for use in applications like automotive radar receivers.

References

[1] Seyyedesfahlan, Mehdi and I. Tekin. "77 GHz PCB Patch Antenna." URSI (International Union of Radio Science) Türkiye Ulusal Komitesi (2016).

Infinite Array of Microstrip Patch Antenna on Teflon Substrate

This example shows how to use infinite array analysis to model a probe-fed microstrip patch element used as a unit cell in an infinite array. Assume that the array is infinitely extended along two horizontal dimensions and located in the XY plane. The dimensions and physical parameters of array are taken from the reference example in [1].

Define Unit Cell

A unit cell refers to a single element of an infinite array. This example uses a probe-fed microstrip patch element as a unit cell. The infinite array of a metal dielectric antenna must have a perfectlyconducting ground plane. Specify the ground plane length as 22.2 mm and width as 23.7 mm.

```
freq = 5e9;  
vp = physconst('lightspeed');  
lambda = vp/freq;  
ucdx = 22.2e-3;  
ucdy = 23.7e-3;
```

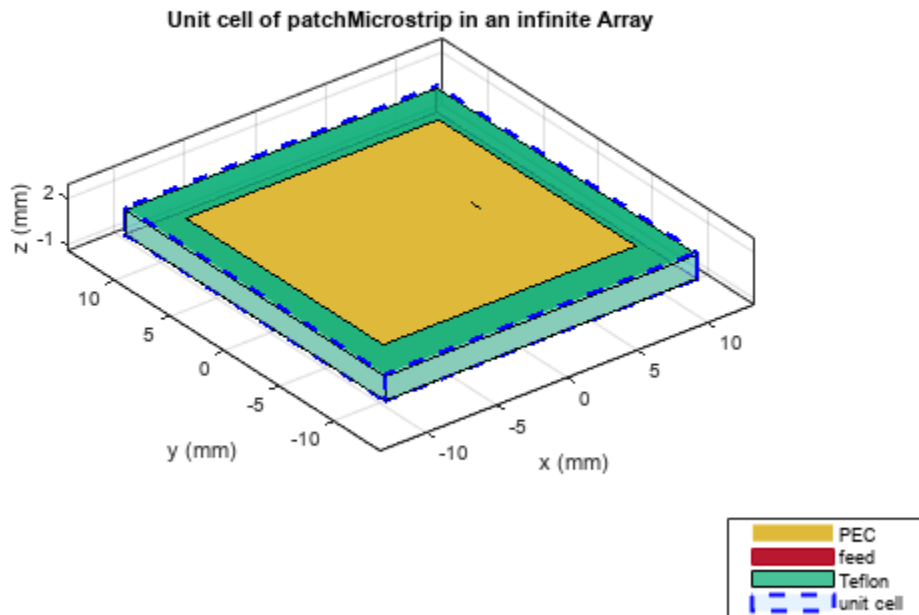
Create a probe-fed microstrip patch antenna with a lossy dielectric substrate.

```
elem = patchMicrostrip;  
elem.Length = 18*1e-3;  
elem.Width = 18*1e-3;  
elem.Height = 1.59e-3;  
elem.Substrate = dielectric('Teflon');  
elem.Substrate.EpsilonR = 2.33;  
elem.Substrate.LossTangent = 0.001;  
elem.Substrate.Thickness = 1.59*1e-3;  
elem.GroundPlaneLength = ucdx;  
elem.GroundPlaneWidth = ucdy;  
elem.FeedOffset = [9*1e-3-4.3e-3 0];
```

Create and Visualize Infinite Array

Create an infinite array and assign the probe-fed microstrip patch antenna as an element to the infinite array. Visualize the antenna.

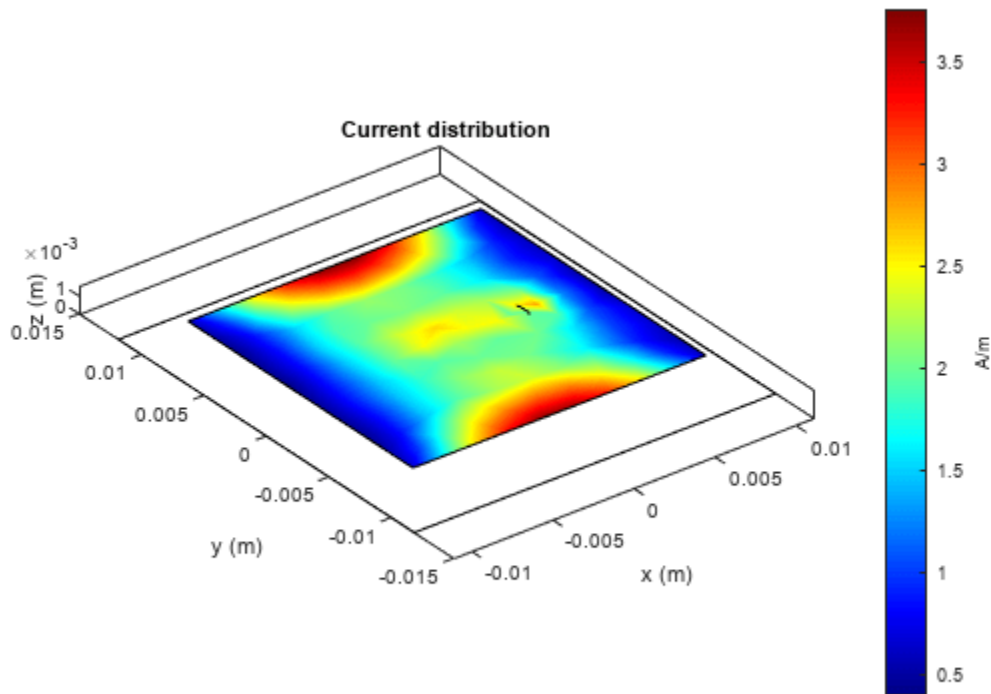
```
ant = infiniteArray;  
ant.Element = elem;  
figure;  
show(ant)
```



Visualize Current

In an Infinite array of metal dielectric structure with infinitely conductor backed dielectric substrate, only the upper metal layer is meshed. The impacts of dielectric and the infinite ground plane are considered in the dyadic Green's function. Calculate and visualize the current distribution on the top metal layer of the unit cell at the frequency of 5 GHz.

```
figure;
current(ant, freq)
```



The current distribution looks similar to the standard dominant current distribution of a rectangular patch antenna with maxima near the two opposite edges.

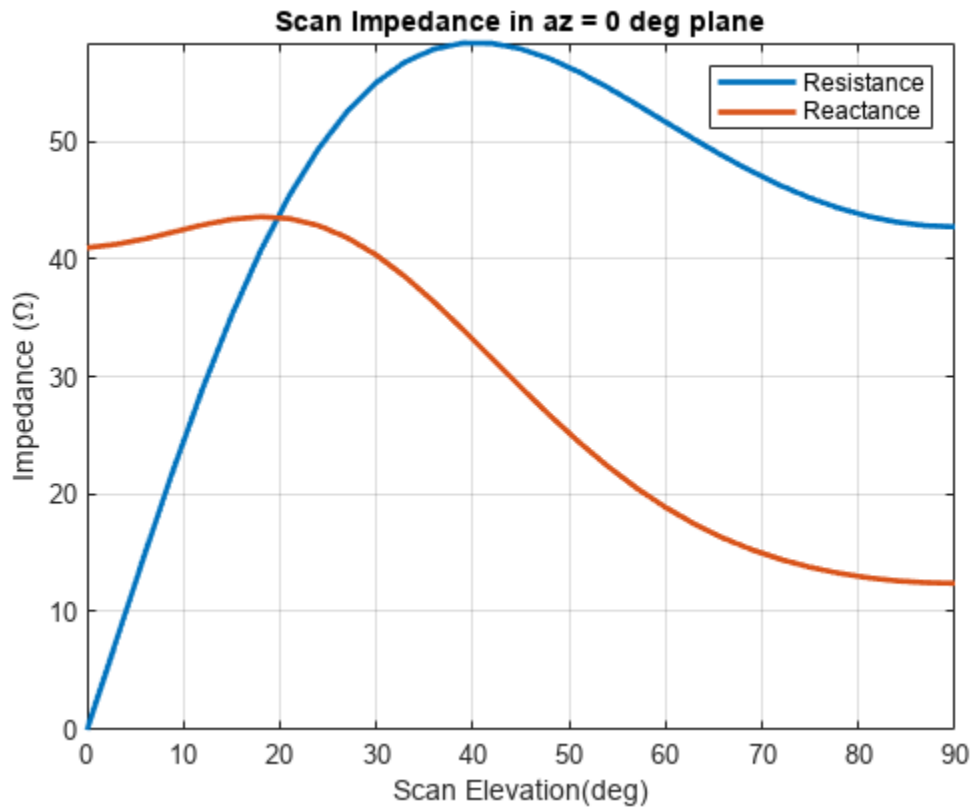
Compute and Visualize Scan Impedance

Compute and plot the scan impedance for varying elevation scan angles and a constant azimuth scan angle of zero degrees (E-plane). In an infinite array, the active reflection behavior and the scan element characteristics depend on the scan impedance behavior. For more information, see “Infinite Array Analysis” on page 5-58. As shown below, near the elevation angle of zero degrees the scan resistance is very low, indicating high reflection loss.

```

az = 0;           % azimuth, E-plane
el = 0:3:90;     % elevation
scanZ = nan(1,numel(el));
ant.ScanAzimuth = az;
for i = 1:numel(el)
    ant.ScanElevation = el(i);
    scanZ(i) = impedance(ant,freq);
end
figure
plot(el,real(scanZ),el,imag(scanZ),'LineWidth',2);
grid on
legend('Resistance','Reactance')
xlabel('Scan Elevation(deg)')
ylabel('Impedance (\Omega)')
title(['Scan Impedance in az = ' num2str(az) ' deg plane'])
axis tight

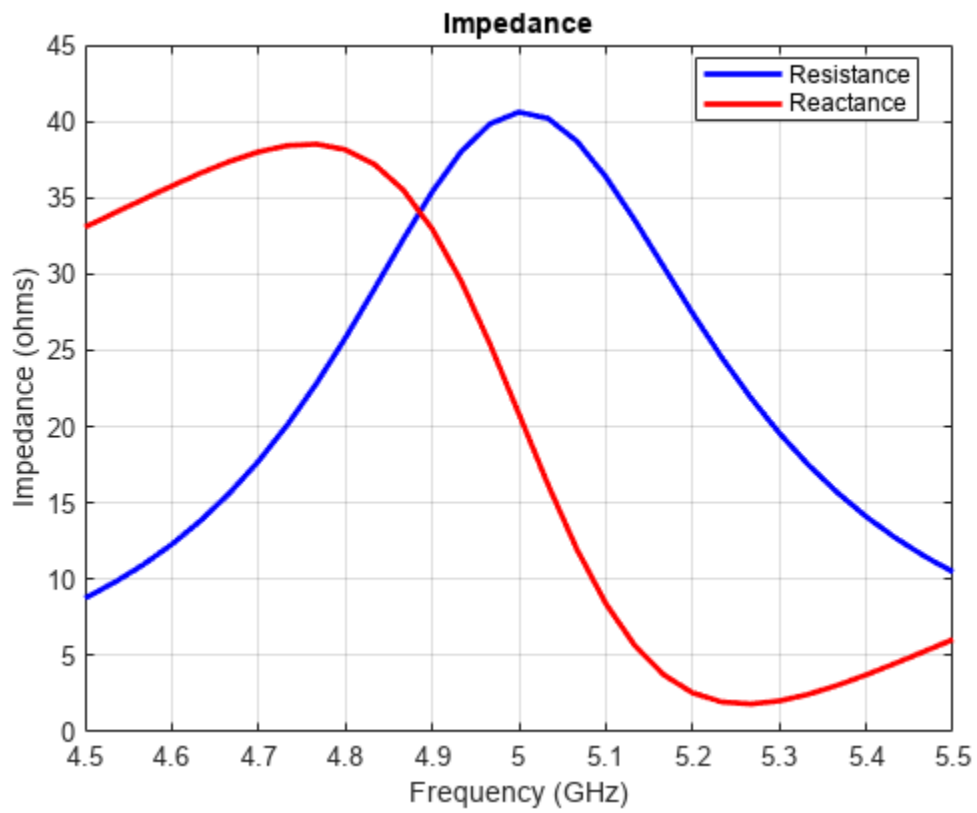
```



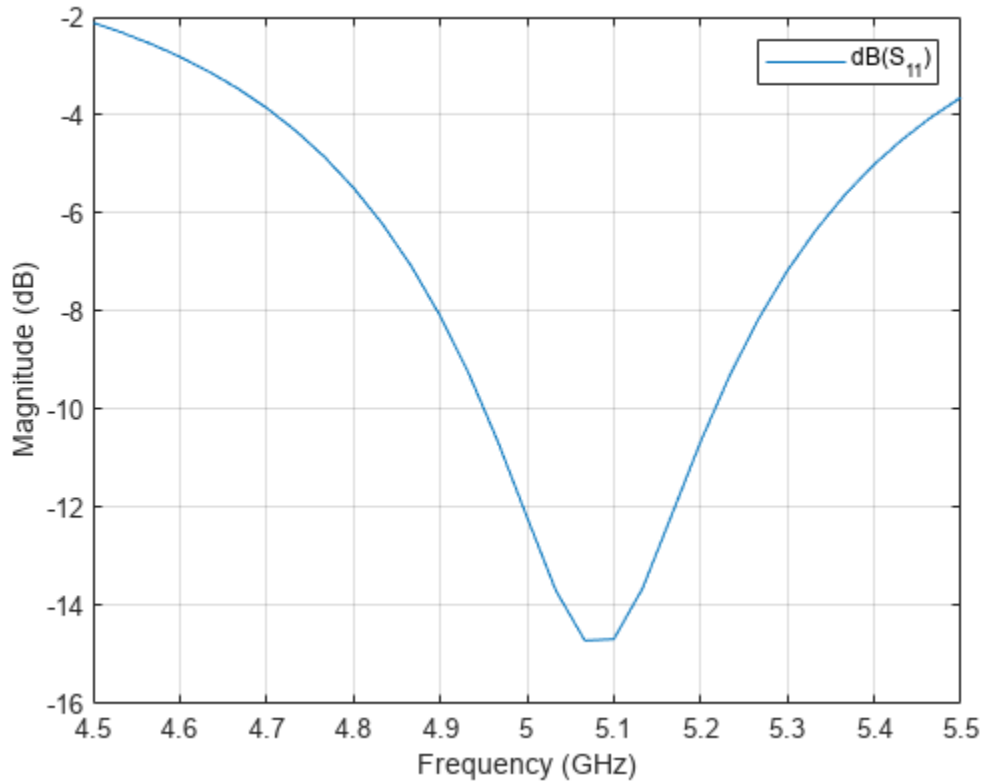
Scan Impedance and S-parameter Variation with Frequency

Fix the scan angles and sweep the frequency to observe the impedance behavior and the S-parameters of the unit cell element.

```
ant.ScanAzimuth = 0;
ant.ScanElevation = 90;
impedance(ant, linspace(4.5e9, 5.5e9, 31))
```



```
s=sparameters(ant,linspace(4.5e9,5.5e9,31));  
figure;  
rfplot(s,1,1)
```



The port reflection shows a dip of around 5.1 GHz, which closely matches the results in [1].

Reference

[1] Deshpande, M., and P. Prabhakar. "Analysis of Dielectric Covered Infinite Array of Rectangular Microstrip Antennas." *IEEE Transactions on Antennas and Propagation* 35, no. 6 (June 1987): 732-36. <https://doi.org/10.1109/TAP.1987.1144169>.

See Also

"Infinite Arrays" on page 2-22 | "Infinite Ground Plane" on page 1-39

Related Examples

- "Infinite Array Analysis" on page 5-58
- "Modeling Infinite Ground Plane in Antennas and Arrays" on page 5-48

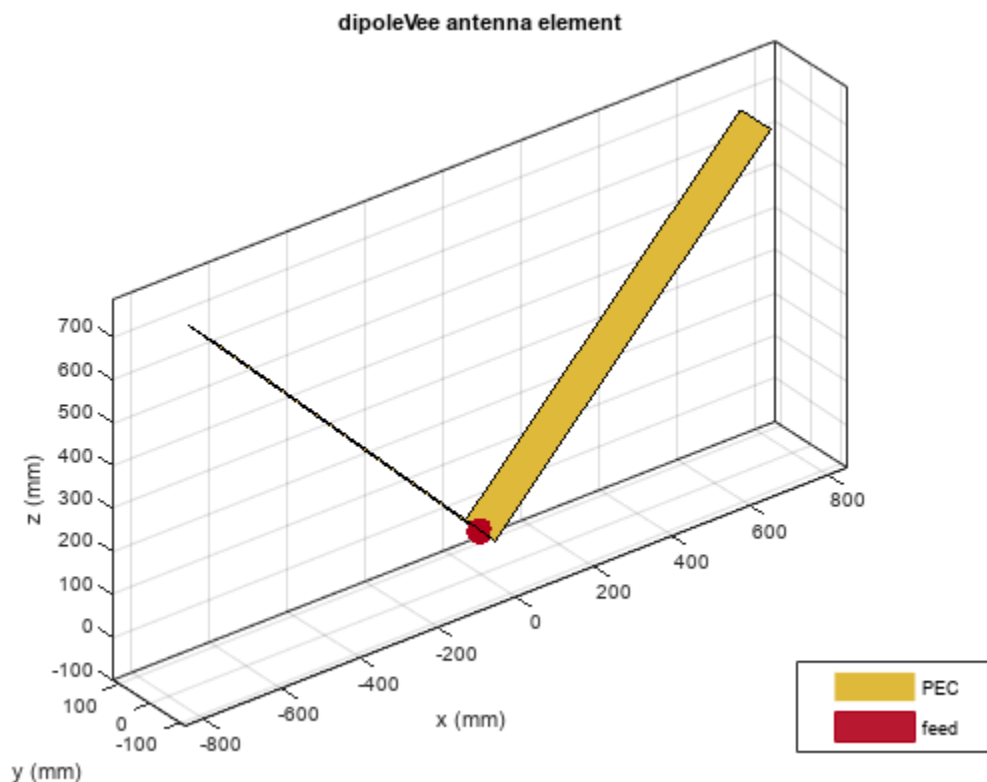
Modeling Wire Antenna and Arrays

This example shows how to model V-dipole wire antennas and arrays using the wire basis function `wireStack`. This workflow eliminates the wire to strip approximation or the strip basis function you need to perform when converting strip structures to wire antennas. You can also use this workflow to model all the antenna elements from the dipole and loop antenna catalog families.

Create Strip Model V-Dipole Antenna

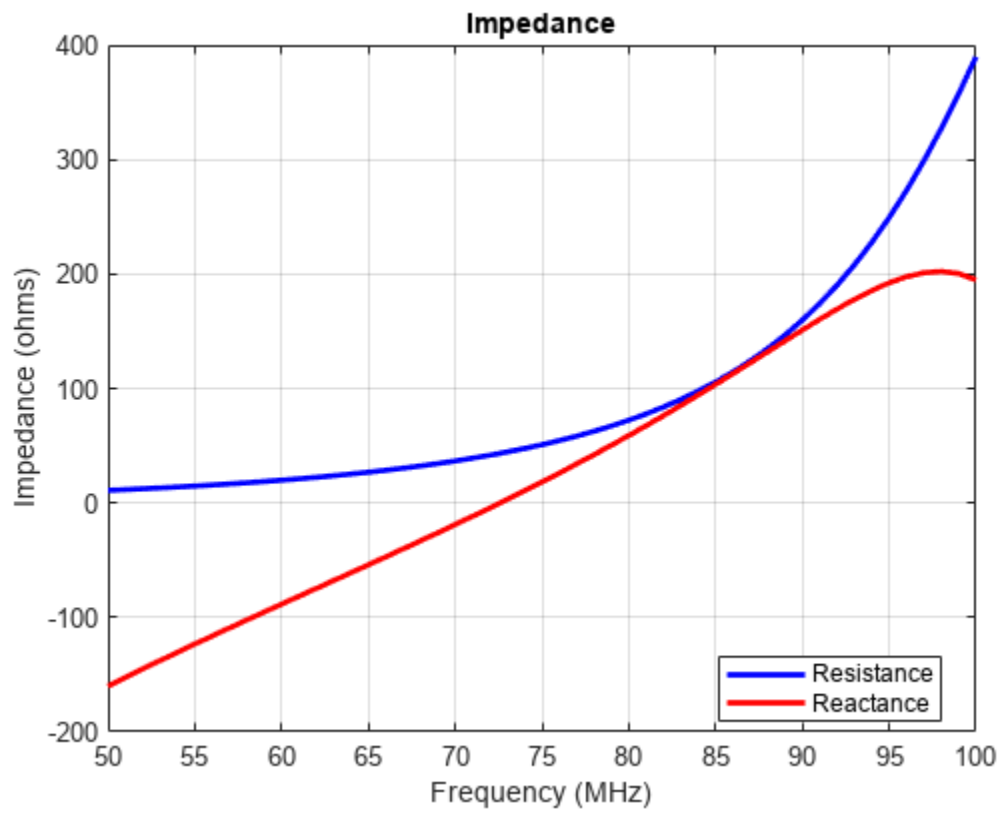
Create and display a `dipoleVee` antenna object. The default V-dipole resonates close to 75 MHz.

```
antStrip = dipoleVee;
show(antStrip)
```



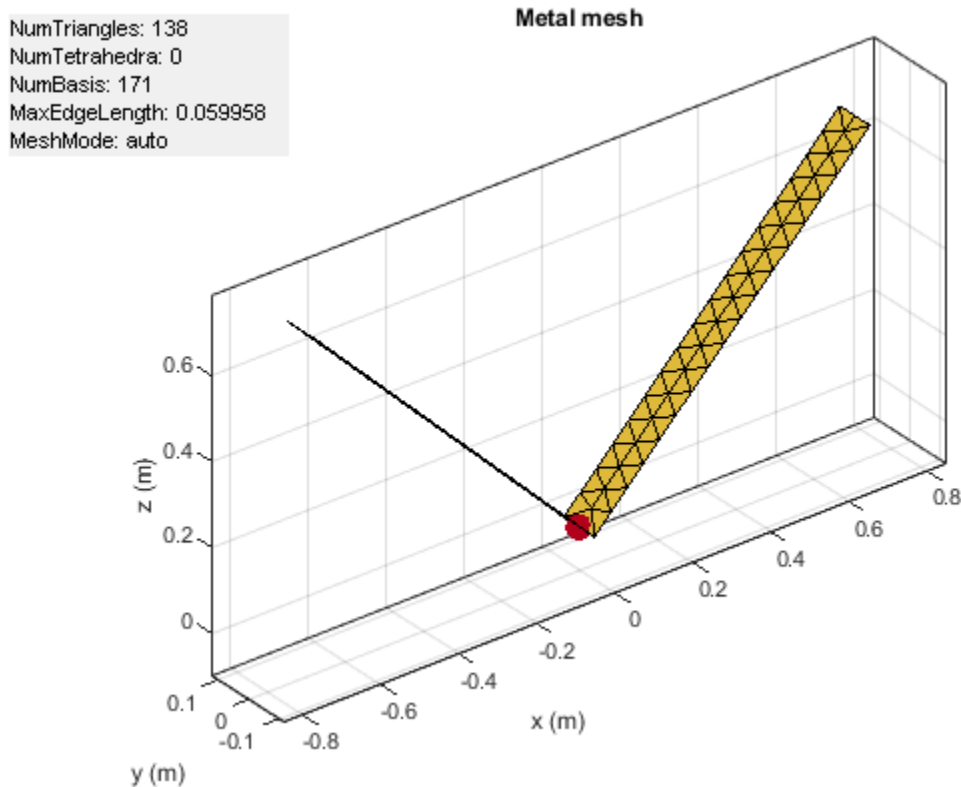
Calculate the impedance of the V-dipole antenna. The impedance plot shows that the V-dipole antenna resonates at 73 MHz.

```
freq = linspace(50e6,100e6,51);
impedance(antStrip, freq)
```

Display the antenna mesh plot. The antenna mesh plot shows that the strip surface is divided into triangles.

```
figure;  
mesh(antStrip)
```



Create Wire Model for V-Dipole Antenna

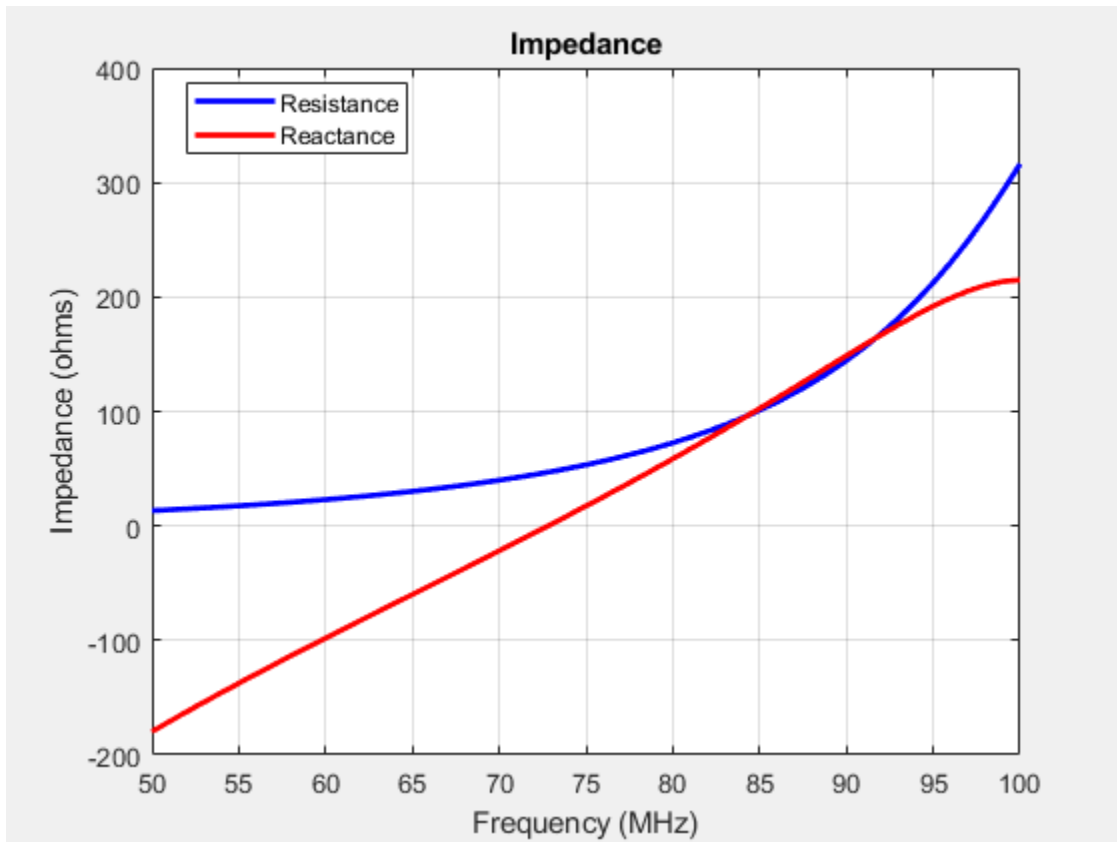
A V-dipole antenna is typically constructed using wires. Use the `wireStack` function to convert the strip model to a wire model. You cannot change the geometric properties of the antenna. However, you can set the `FeedLocation`, `FeedVoltage`, `Tilt`, and `TiltAxis` properties. To update the geometrical properties such as `ArmLength` and `Width`, you must convert the wire model back to the strip model.

```
antwire = wireStack(antStrip)

antwire =
  wireStack with properties:
      Name: 'V-dipole'
  FeedLocation: [-0.1250 0 0.1250]
  FeedVoltage: 1
  FeedPhase: 0
  Tilt: 0
  TiltAxis: [1 0 0]
```

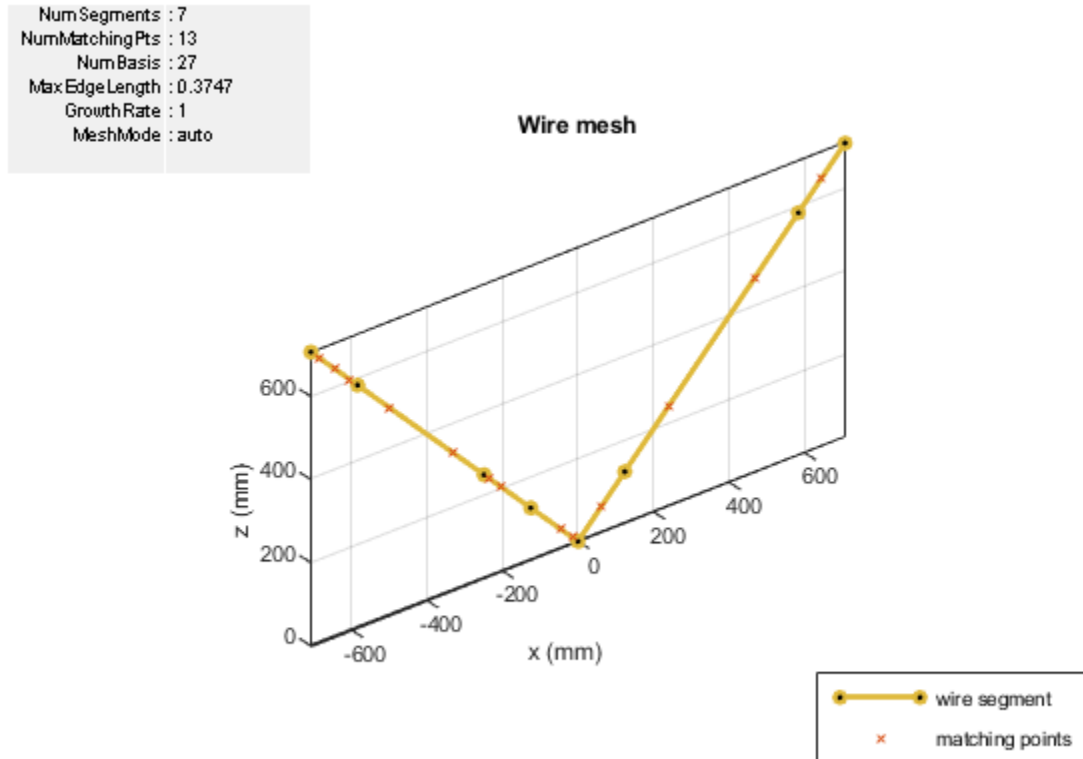
As seen from the impedance plot, the behavior of the wire antenna is similar to the strip antenna.

```
impedance(antwire, freq)
```



Plot the mesh of the wire model of the V-dipole antenna.

```
figure;  
mesh(antwire)
```



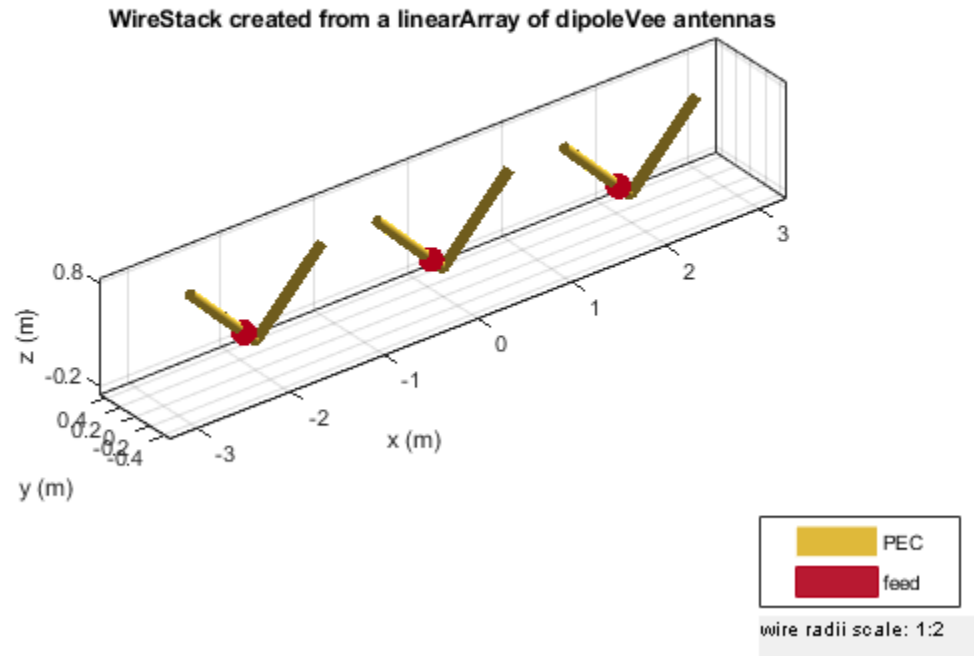
The meshes of the strip and the wire models of the V-dipole antenna are different. The mesh of the wire V-dipole antenna is based on the thin-wire approximation, that is, the wire is discretized into one-dimensional wire segments on which the current is approximated as a polynomial of order N . The currents at each segment are obtained by solving a set of $N+1$ equations per segment, two equations at the segment edges, and $N-1$ equations at matching points dispensed along the segment. The segments and matching points are displayed in the mesh plot to give a complete picture of the discretization choices made to solve the structure.

Create Array of Wire Antennas

To create an array of wire antennas, first create an array of the catalog element. This workflow applies to linear and conformal arrays.

Create and display a three-element V-dipole linear antenna array.

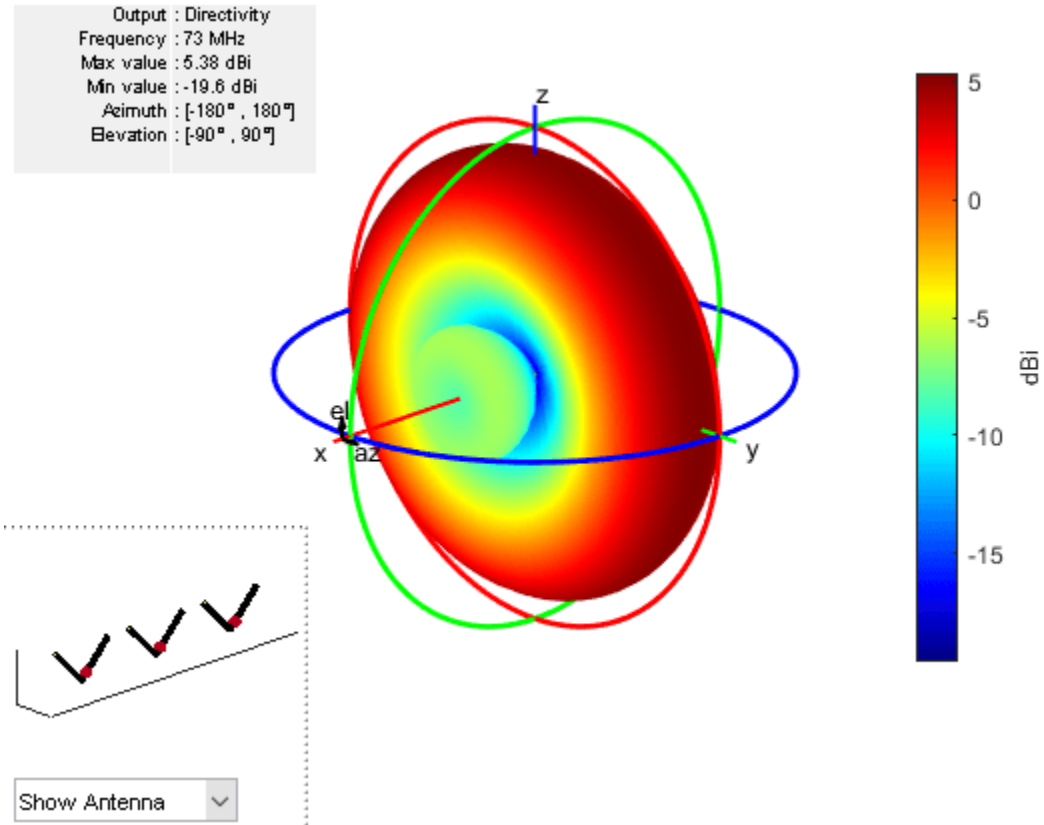
```
l = linearArray("Element",antStrip,"NumElements",3);
arr = wireStack(l);
show(arr)
```



The feed point of the wire V-dipole is offset by a small amount from the antenna origin. This is the outcome of an implementation-based requirement that the feed be placed on a straight portion of the wire. The offset is roughly 7-wire radii.

All the analysis supported for catalog elements are supported for wire arrays. Calculate the radiation pattern of the three-element V-dipole linear antenna array.

```
pattern(arr, 73e6)
```



See Also

Related Examples

- “Wideband Blade Dipole Antenna and Array” on page 5-243
- “Analysis of Biquad Yagi for Wi-Fi Applications” on page 5-600

Analysis of Electrically Large Structures Using Hybrid MoM and FMM

This example shows how to analyze electrically large antennas using two solvers in Antenna Toolbox™: the hybrid method of moments (MOM) and the fast multipole method (FMM). The overall size of an antenna can be expressed in electrical terms, as a function of the frequency of operation or wavelength λ . Doing so can enable you to choose the electromagnetic solver to deploy for analysis.

Typical antennas such dipole antennas and microstrip patch antennas have a radiating element size of $\frac{\lambda}{2}$. Including a ground plane can increase the overall dimensions up to 3λ . However, certain classes of antennas, by their intended application are electrically large. A common example of such antennas is the parabolic reflector. These antennas have an electrical size in the range of 5λ to 100λ .

To solve these structures, you must discretize the geometry. In Antenna Toolbox, metal surfaces are discretized into a mesh consisting of triangles. The immediate effect of a larger electrical size is an increase in the size of the mesh results in more unknowns to solve for, during analysis by the solver.

You can use one of two approaches to solving electrically large structures: the first approach is hybrid MoM, which is an approximate technique that relies on the physical optics (PO) method to couple an electrically large portion of the geometry being analyzed to the relatively smaller radiating element being handled by a full-wave MoM technique.

The second approach uses an iterative FMM algorithm to avoid building the interaction matrix, thereby avoiding the storage constraints of a typical direct solver (relying on the explicit matrix inverse), but still produces a solution with full-wave accuracy.

The hybrid MoM approach will trade-off full-wave accuracy in order to be faster whereas the FMM based solver removes the matrix storage requirements of a direct solver thus enabling solution of a electrically large structures with full-wave accuracy.

Define Reflector Specifications

Define the frequency and physical dimensions of the parabolic reflector. The reflector consists of a $\frac{\lambda}{2}$ dipole as the exciter. The electrical size of this parabolic reflector is $\approx 20\lambda$.

```
Fc = 3.2975e9;
lambda = physconst('lightspeed')/Fc;
D = 2;
F_over_D = 0.51;
```

Create Parabolic Reflector

Use the design object function to design a parabolic reflector at the desired center frequency and adjust the properties to align with the physical specifications.

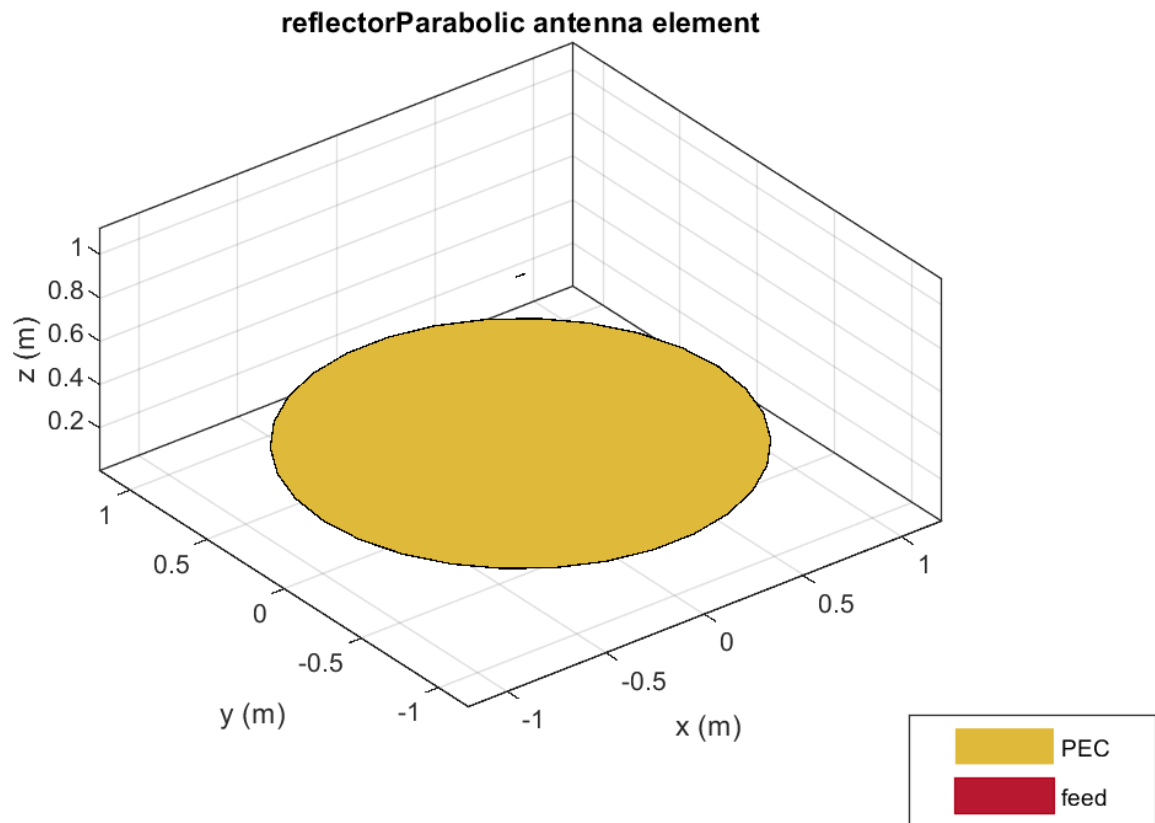
```
p = design(reflectorParabolic,Fc);
p.Radius = D/2;
p.FocalLength = D*F_over_D

p =
    reflectorParabolic with properties:
```

```
Exciter: [1x1 dipole]
Radius: 1
FocalLength: 1.0200
FeedOffset: [0 0 0]
Tilt: 0
TiltAxis: [1 0 0]
Load: [1x1 lumpedElement]
SolverType: 'MoM-PO'
```

Visualize the structure using the `show` object function. Notice that one of the properties in the object display is `SolverType`. By default, the hybrid solver is the solver for antennas of the reflector family.

```
figure
show(p)
```

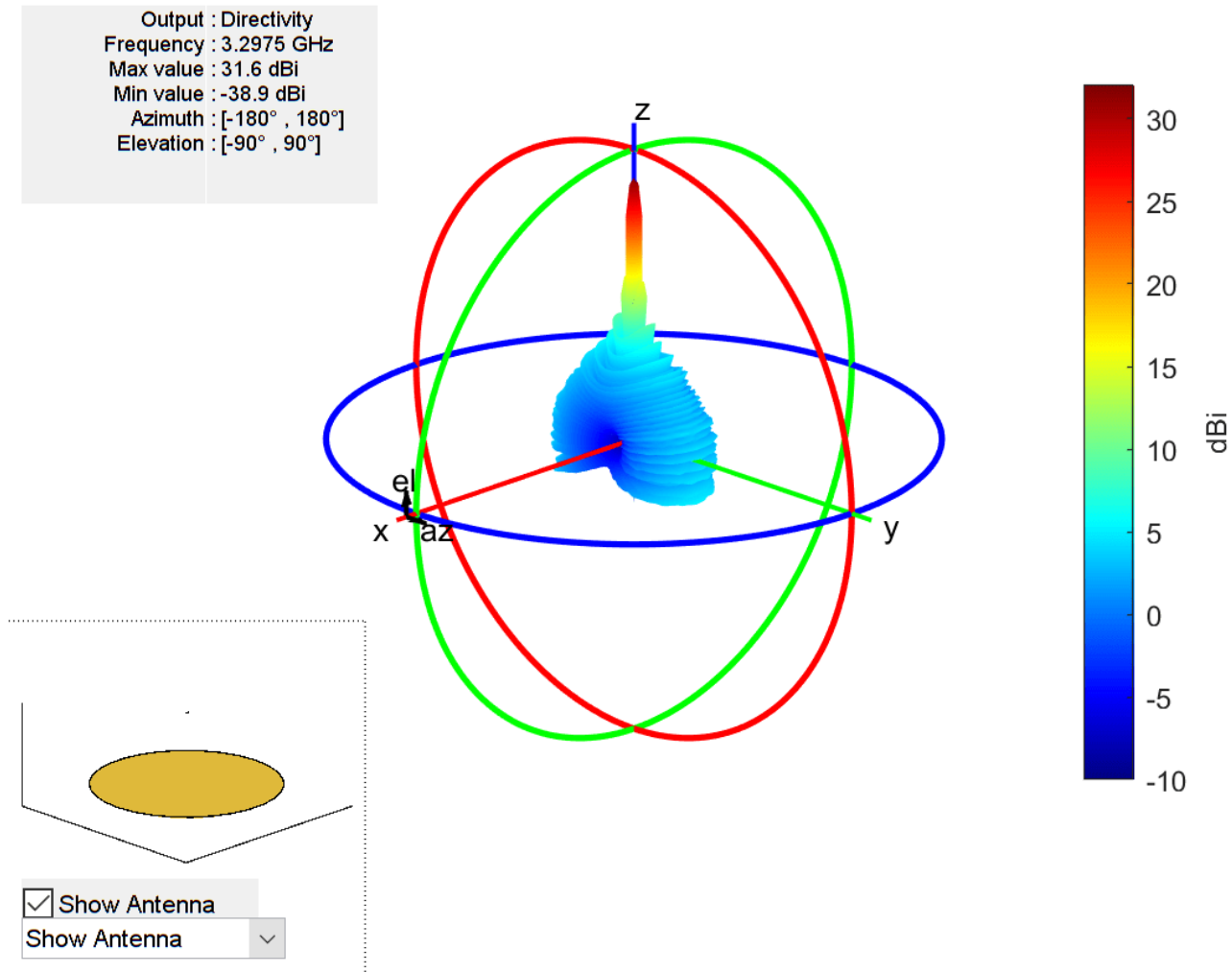


Plot Radiation Pattern

Set up a far-field analysis by using the `pattern` function. The default angular spacing for this function in both the azimuth and elevation planes is 5 degrees. Since the antenna is an electrically large reflector antenna, expect a narrow main beam towards the zenith. For this example, specify a

finer angular discretization of 1 degrees. in both planes. Use the `PatternPlotOptions` class to adjust the magnitude limits on the overall plot output.

```
az = -180:2:180;
el = -90:1:90;
P = PatternPlotOptions;
P.MagnitudeScale = [-10 32];
figure
pattern(p,Fc,az,el,patternOptions=P)
```



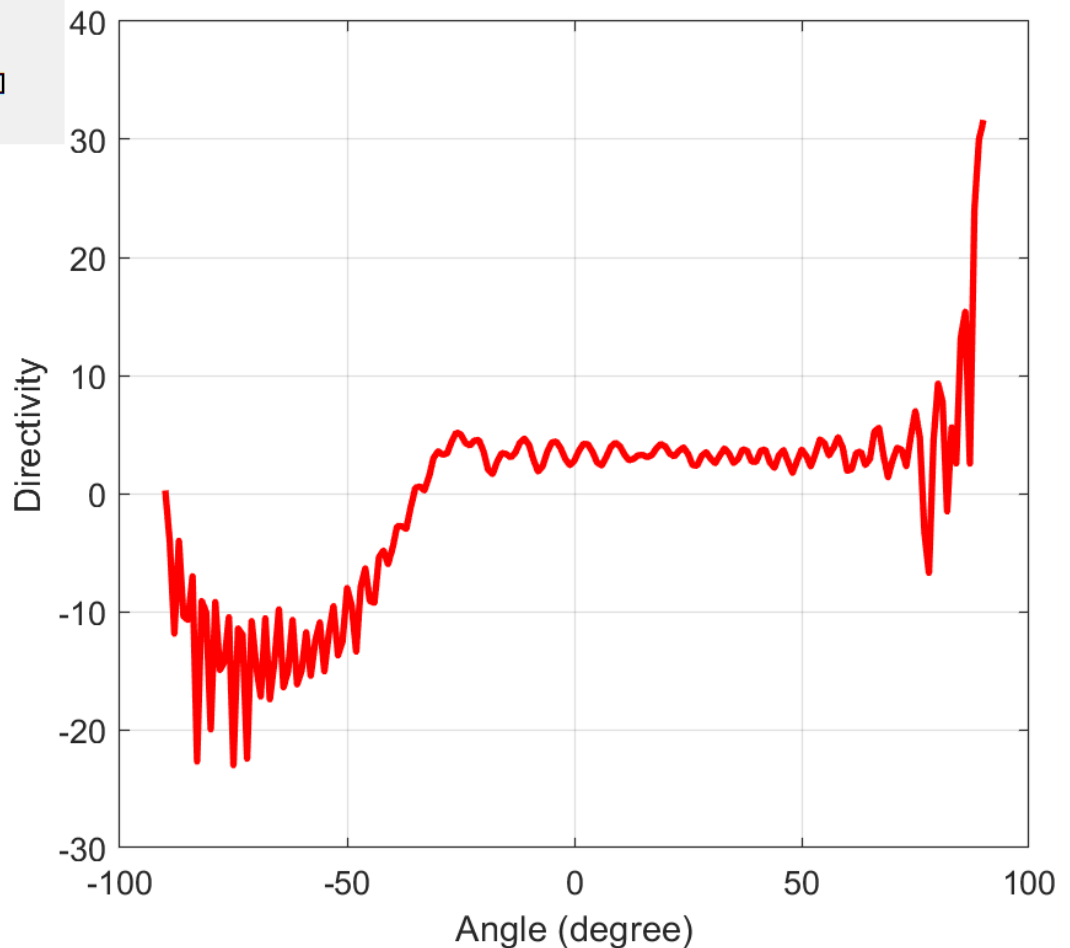
Fix the azimuth angle to 90 degrees and change the coordinate system to rectangular to generate a pattern plot that varies with the elevation plane.

```
figure
pattern(p,Fc,90,el,CoordinateSystem='rectangular');
```

```

Output : Directivity
Frequency : 3.2975 GHz
Max value : 31.6 dBi
Min value : -23 dBi
Azimuth : 90°
Elevation : [-90° , 90°]

```



Switch to FMM Solver

Use the `SolverType` property on the reflector and switch it to FMM. The FMM uses an iterative solver to arrive at the final solution. Invoking the object function, `solver`, on the antenna object gives you access to the properties. One of the key advantages with a hybrid solver like MoM-PO is that you can relax the meshing criterion on the structure.

```

p.SolverType = 'FMM';
s = solver(p)

```

```

s =
  EFIE with properties:

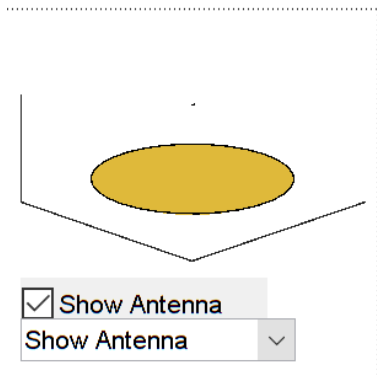
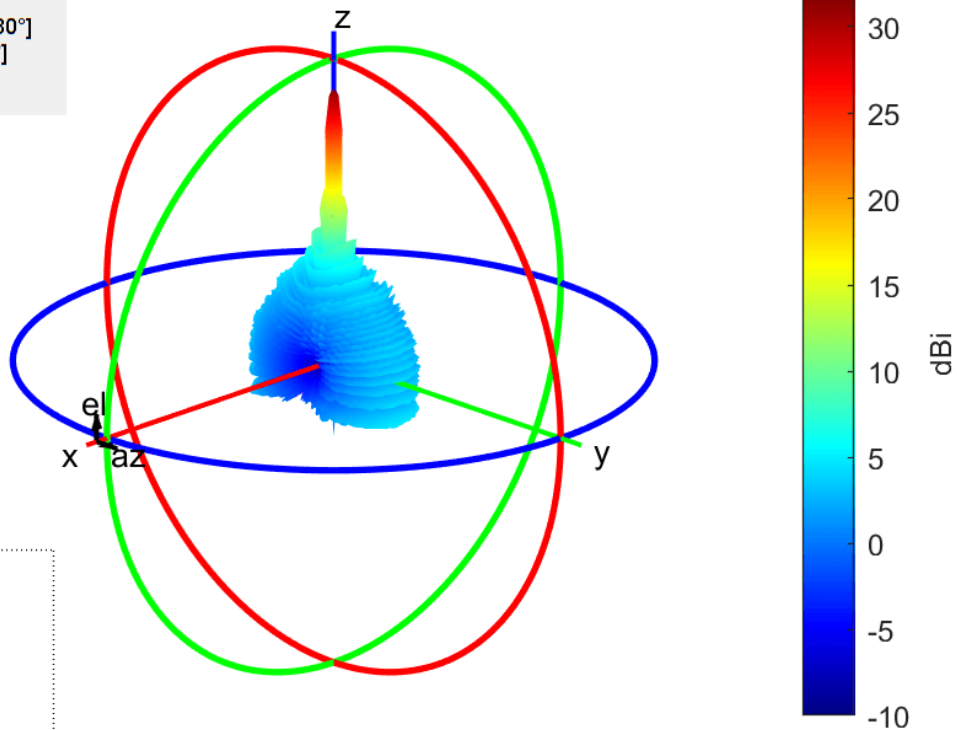
    IterativeSolver: 'gmres'
    Iterations: 100
    RelativeResidual: 1.0000e-04
    Precision: 2.0000e-04

```

Because the FMM is a full-wave solver, you must refine the mesh. To understand this, change the default number of iterations to 20 and calculate and visualize the pattern as before.

```
s.Iterations = 20;
figure
pattern(p,Fc,az,el,patternOptions=P)
```

```
Output : Directivity
Frequency : 3.2975 GHz
Max value : 31.5 dBi
Min value : -35.8 dBi
Azimuth : [-180° , 180°]
Elevation : [-90° , 90°]
```

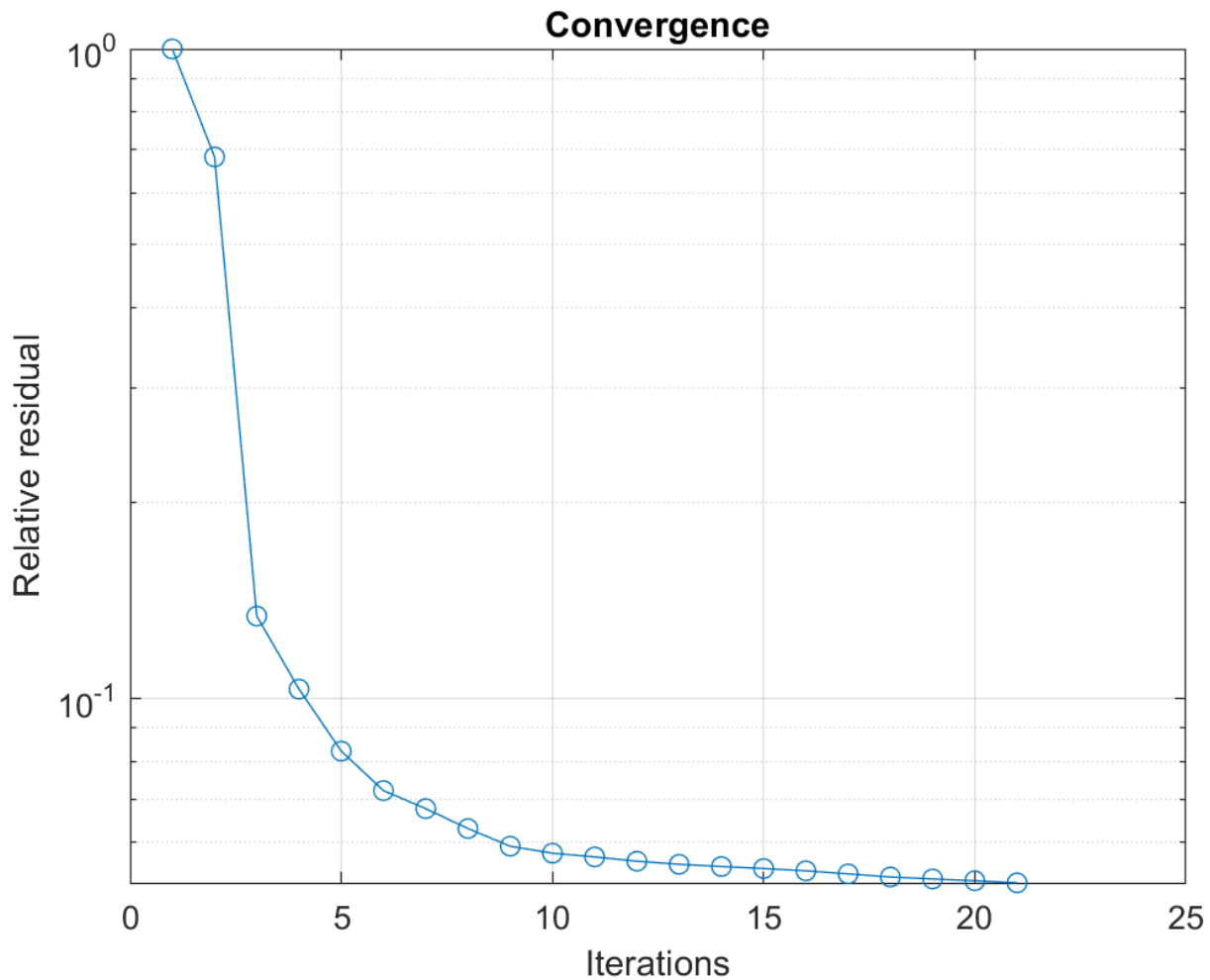


Plot Solver Convergence

The far-field pattern appears similar to that computed from the hybrid MoM method. Plot the convergence of the solver as a function of the number of iterations. The requested `RelativeResidual` in the solver properties is set to $1e-4$, whereas the convergence plot reveals that past the fifth iteration, the convergence slows down dramatically and eventually fails to achieve the desired residual.

The iterative solver solves the system of equations where the V is known and the Z , the interaction matrix, is never formed and stored. The left-hand side is instead computed as a matrix-vector product using the estimated I for each iteration by the GMRES algorithm. The convergence trend observed in the plot suggests that the mesh might need to be refined.

```
figure
convergence(s)
```



Improve Mesh and Solve

Before refining the mesh, check the ratio of the wavelength to the maximum edge length.

```
M = mesh(p)
```

```
M = struct with fields:
    NumTriangles: 4656
    NumTetrahedra: 0
    NumBasis: 6895
    MaxEdgeLength: 0.0391
    MeshMode: 'auto'
```

```
mesh_ratio = lambda/M.MaxEdgeLength
```

```
mesh_ratio = 2.3265
```

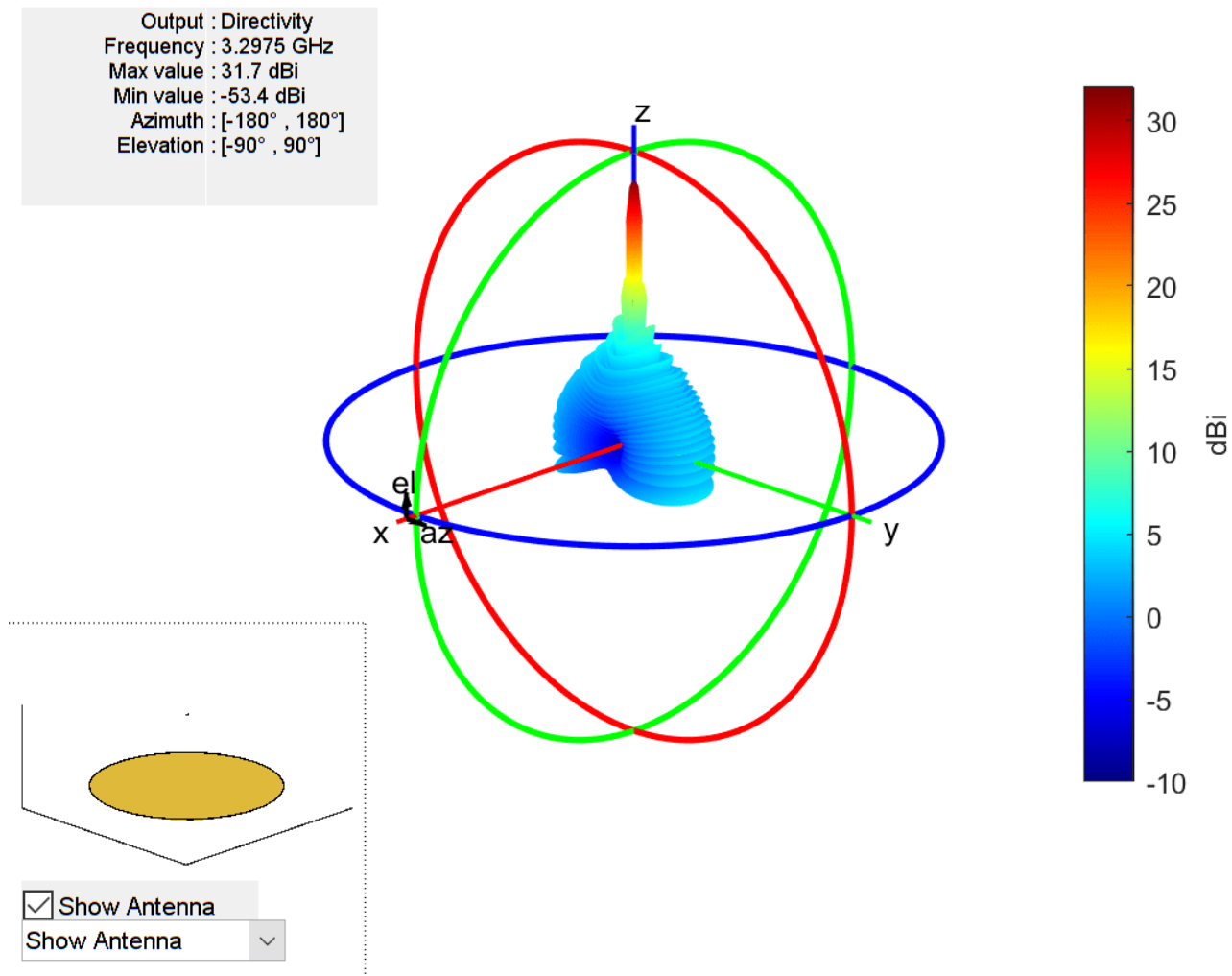
Use the `mesh` function to refine the mesh and then recompute the pattern. The solver properties are retained.

```
M = mesh(p,MaxEdgeLength=lambda/6)
```

```
M = struct with fields:
  NumTriangles: 33076
  NumTetrahedra: 0
  NumBasis: []
  MaxEdgeLength: 0.0152
  MeshMode: 'manual'
```

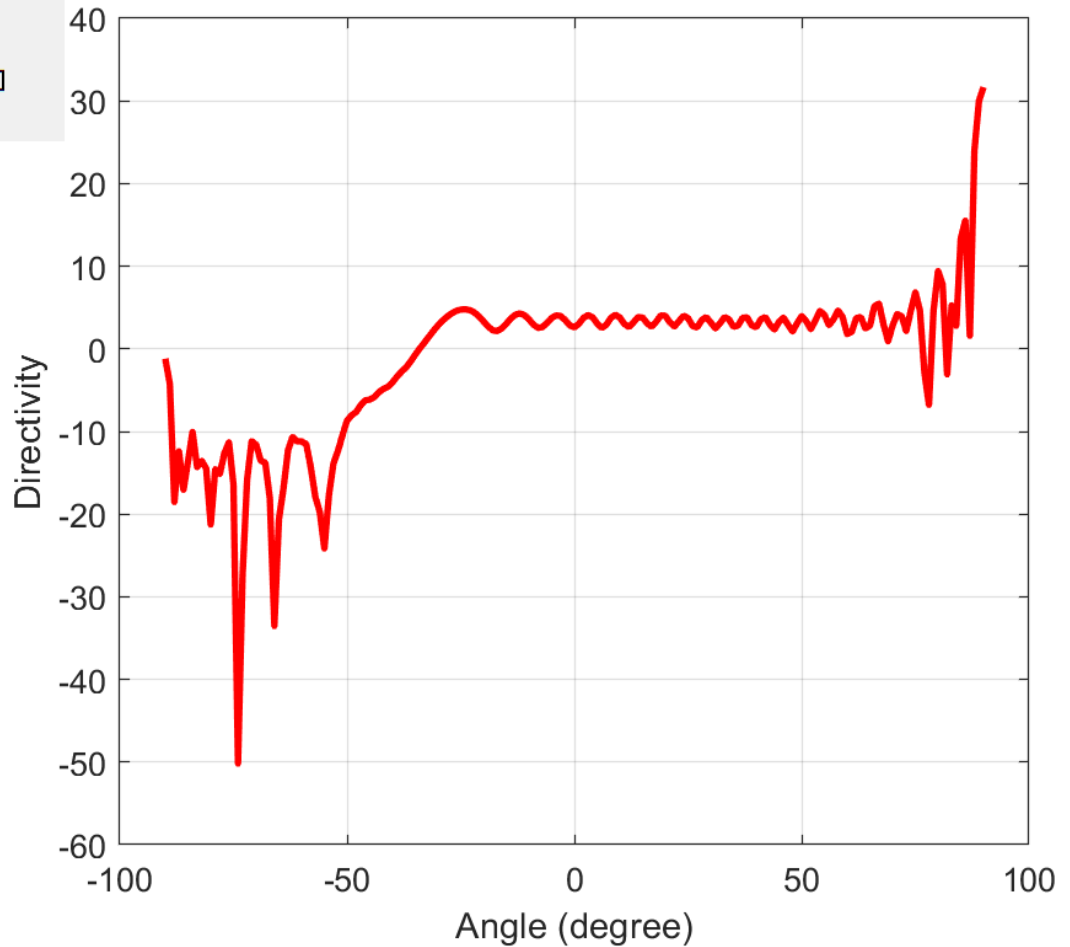
Plot the 3-D pattern and a slice at the azimuth of 90 degrees to show the variation along the elevation plane.

```
figure
pattern(p,Fc,az,el,patternOptions=P)
```



```
figure
pattern(p,Fc,90,el,CoordinateSystem='rectangular');
```

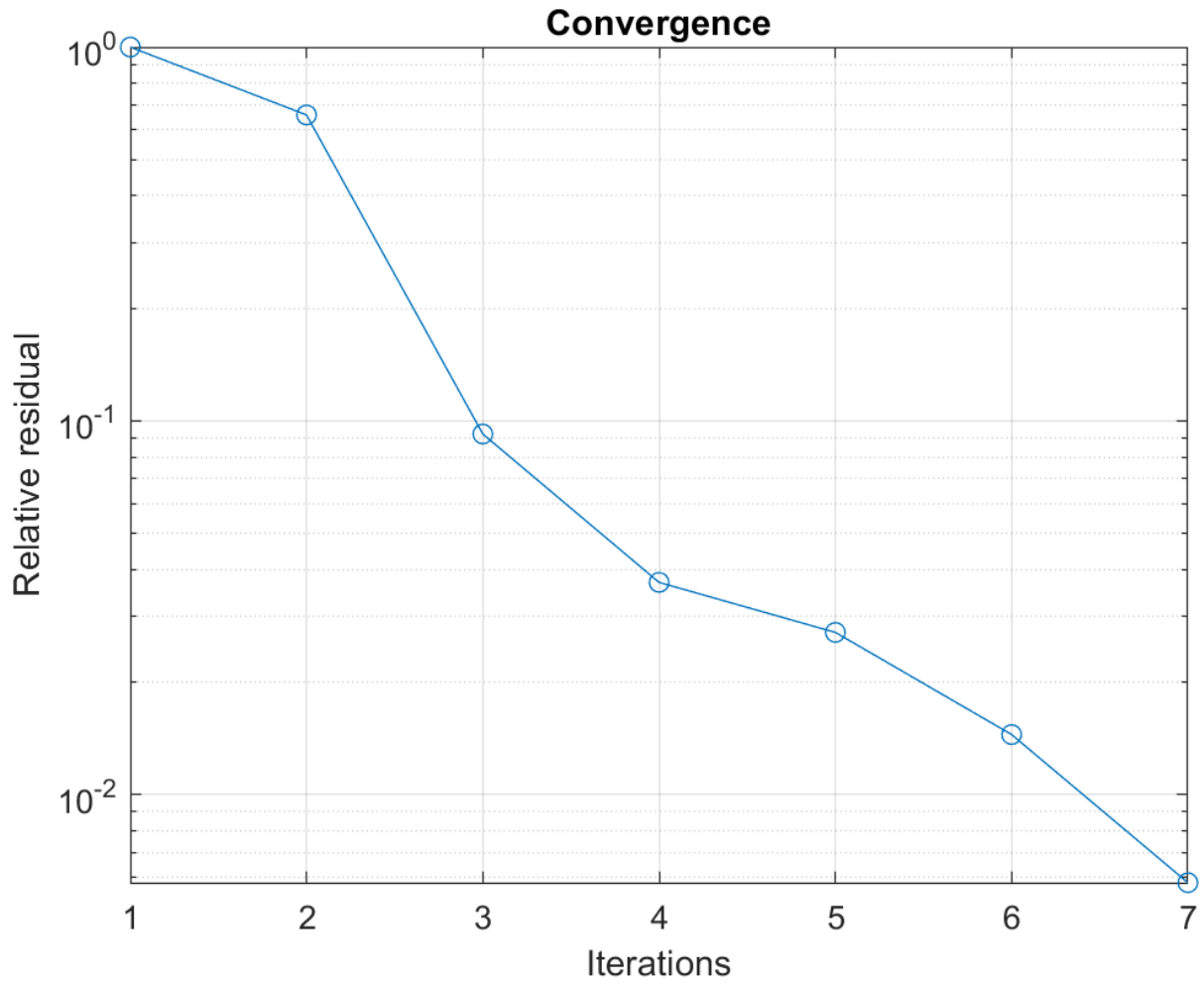
Output : Directivity
Frequency : 3.2975 GHz
Max value : 31.7 dBi
Min value : -50.2 dBi
Azimuth : 90°
Elevation : [-90° , 90°]



The plot shows that although the maximum value for the directivity does not change by much, the nulls in the pattern are resolved more accurately. Note the approximately 20 dB drop in minimum value of directivity registered.

Plot the convergence of the solver. Notice that, the relative residual is achieved this time in fewer iterations, but the computation takes more time, since mesh refinement results in a greater number of unknowns to solve.

figure
convergence(s)



See Also

Related Examples

- "Infinite Array Analysis" on page 5-58

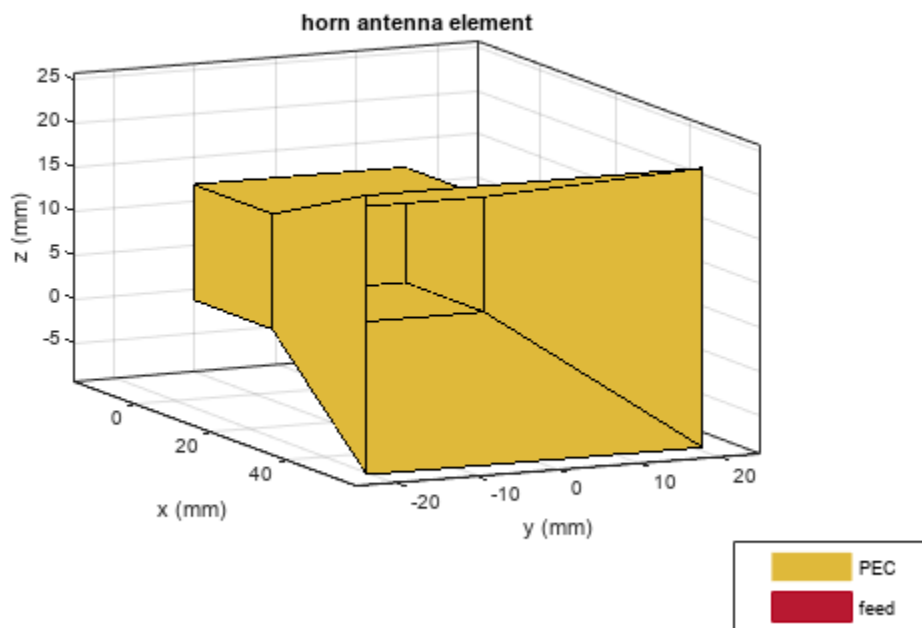
Analyze Cylindrical Reflector Antenna with Horn Array Feed

This example shows how to analyze an antenna consisting of a cylindrical reflector with a linear array of horns.

Create Horn Antenna

Create a horn antenna resonating at 10 GHz. Set `lambda` to the free space wavelength at the design frequency of 10 GHz.

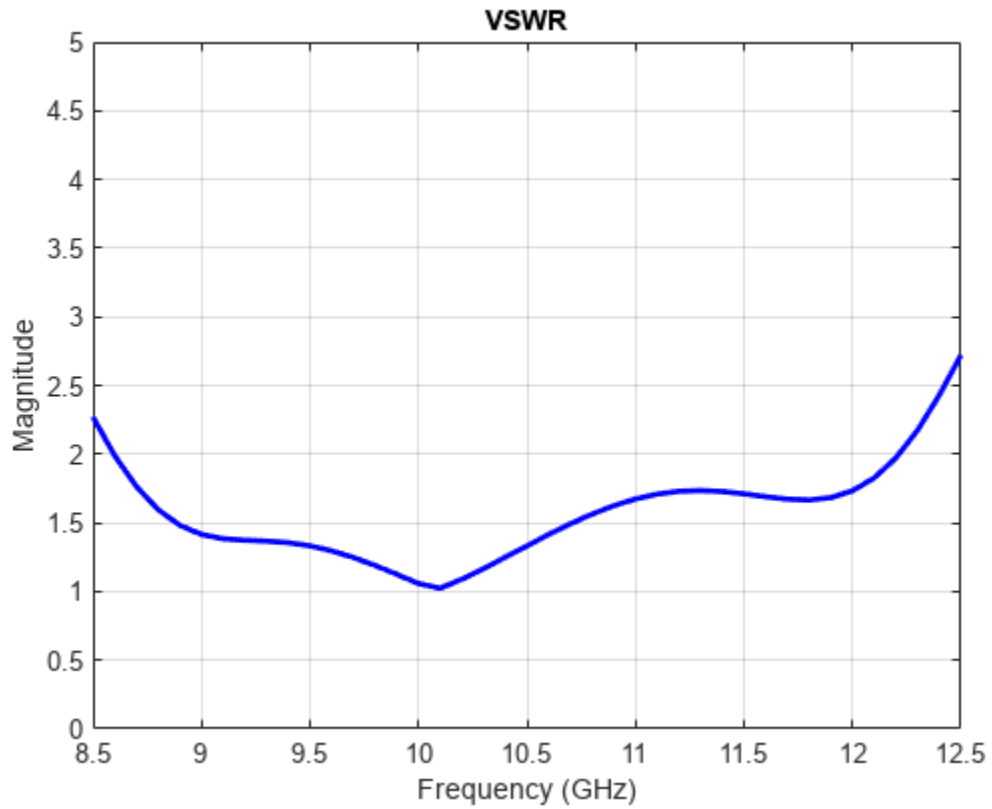
```
freq = 10e9;  
lambda = physconst('light')/freq;  
h = design(horn,10e9);  
h.FlareWidth = 0.0416;  
h.FlareLength = 0.0416;  
h.FlareHeight = 31.8e-3;  
h.Length = 21e-3;  
h.FeedOffset(1) = -3.6e-3;  
figure;  
show(h)
```



Plot VSWR and Radiation Pattern of Horn Antenna

Plot the voltage standing wave ratio (VSWR) of the horn antenna over the frequency range of 8.5–12.5 GHz.

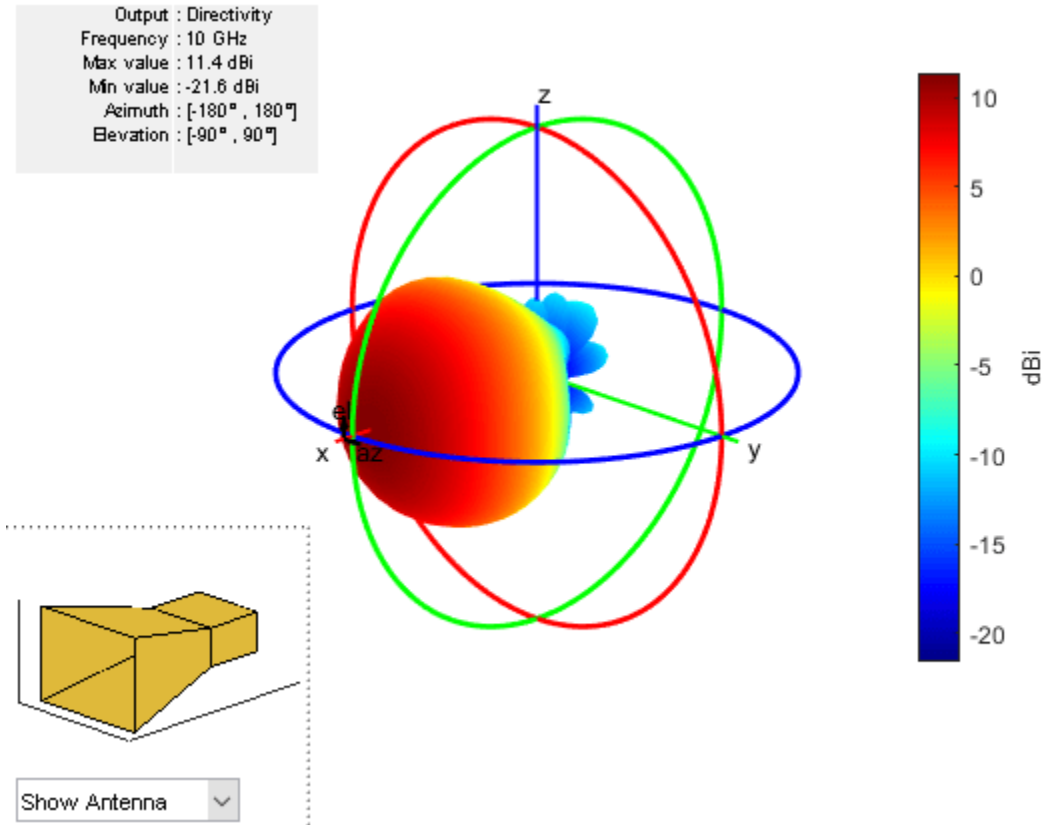

```
range = (0.85:0.01:1.25)*freq;  
figure;  
vswr(h,range);  
ylim([0 5]);
```



The VSWR of the horn antenna is less than 2 between 8.6–12.2 GHz.

Plot the radiation pattern of the horn antenna.

```
figure;  
pattern(h,freq);
```



The directivity of the antenna is 11.4 dB.

Animate the Field in the E-plane and H-plane

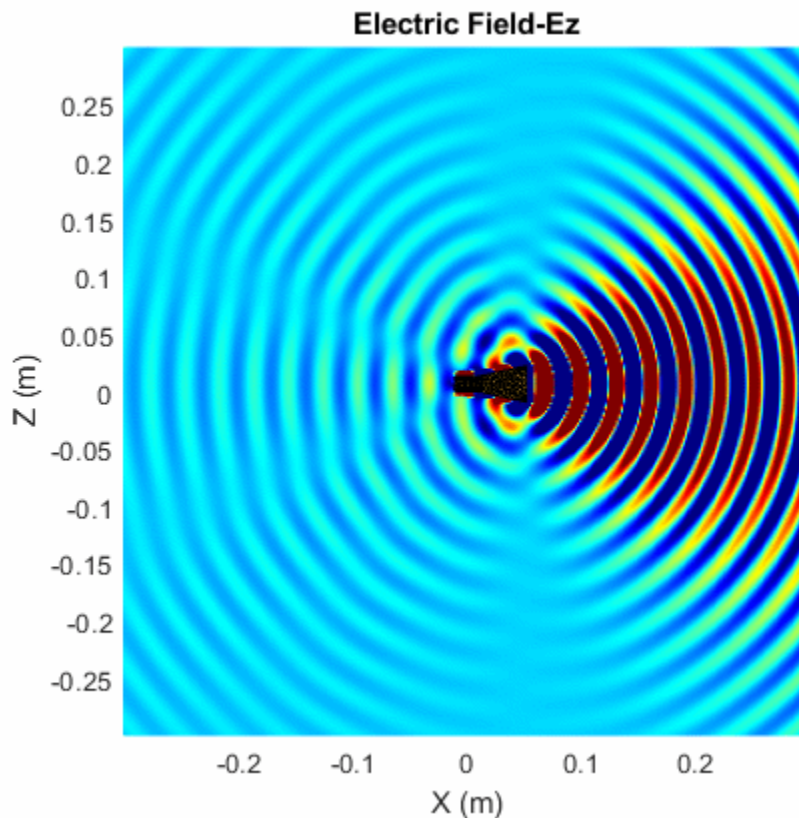
The analysis from the solver provides us with the steady state result. The field calculation provides the 3 components of the electric and magnetic field in phasor form in the cartesian coordinate system. Animate this field by using a propagator defined by the frequency, f_c to bring the result back to time-harmonic form and visualize on the E-plane and H-plane. In the case of the horn, E-plane is the XZ plane and H-plane is the XY plane. As noted in the far-field pattern plot shown in the previous section, the main lobe is along the positive x-axis. Use the helper `AnimateEHFields` function as shown below. Define the spatial extent, point density as well as the total number of time periods defined in terms of the frequency f_c . Choose the E_z component with the plotting plane defined as XZ for the E-field and the H_y component with the plotting plane defined as XY for the animation. If the sampling time is not defined, the function automatically chooses a sampling time based on 10 times the center frequency. There is also an option available to save the animation output to a gif file with an appropriate name specified.

```
% Define Spatial Plane Extent
SpatialInfo.XSpan = 20*lambda;
SpatialInfo.YSpan = 20*lambda;
SpatialInfo.ZSpan = 20*lambda;
SpatialInfo.NumPointsX = 300;
SpatialInfo.NumPointsY = 300;
SpatialInfo.NumPointsZ = 300;
% Define Time Extent
Tperiod = 1/freq;
```

```

T = 1*Tperiod;
TimeInfo.TotalTime = T;
TimeInfo.SamplingTime = [];
% Define E-Field Data details
DataInfo.Component = 'Ez';
DataInfo.PlotPlane = 'XZ';
DataInfo.DataLimits = [-5 10];
DataInfo.ScaleFactor = 1;
DataInfo.GifFileName = 'hornEz.gif';
DataInfo.CloseFigure = true;
%% Animate
helperAnimateEHFields(h,freq,SpatialInfo, TimeInfo, DataInfo)

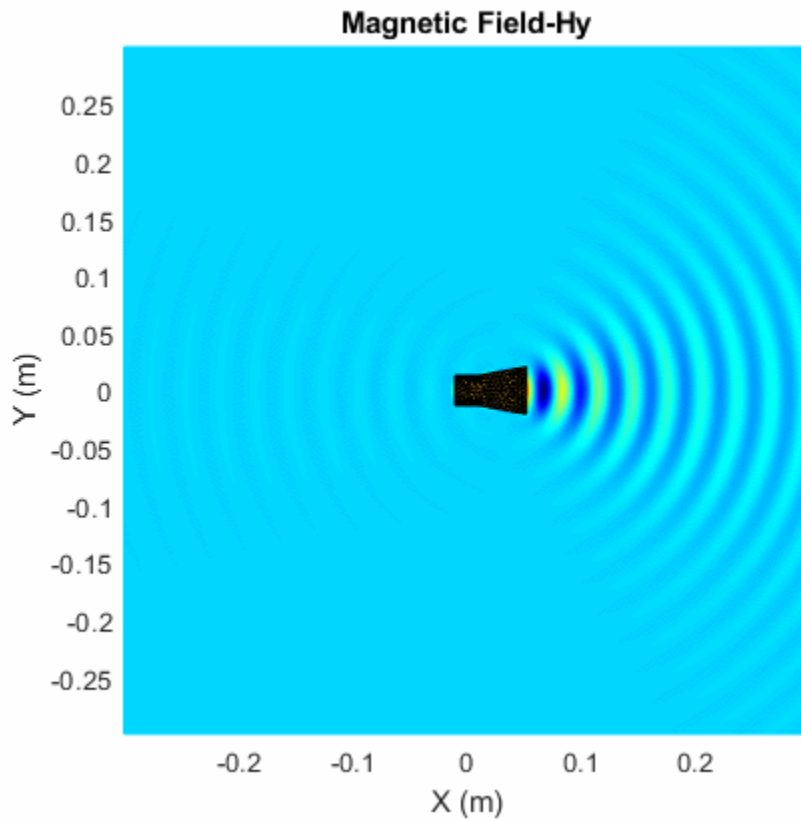
```



```

% Define H-Field Data details
DataInfo.Component = 'Hy';
DataInfo.PlotPlane = 'XY';
DataInfo.DataLimits = [-0.25 0.5];
DataInfo.ScaleFactor = 1;
DataInfo.GifFileName = 'hornHy.gif';
%% Animate
helperAnimateEHFields(h,freq,SpatialInfo, TimeInfo, DataInfo)

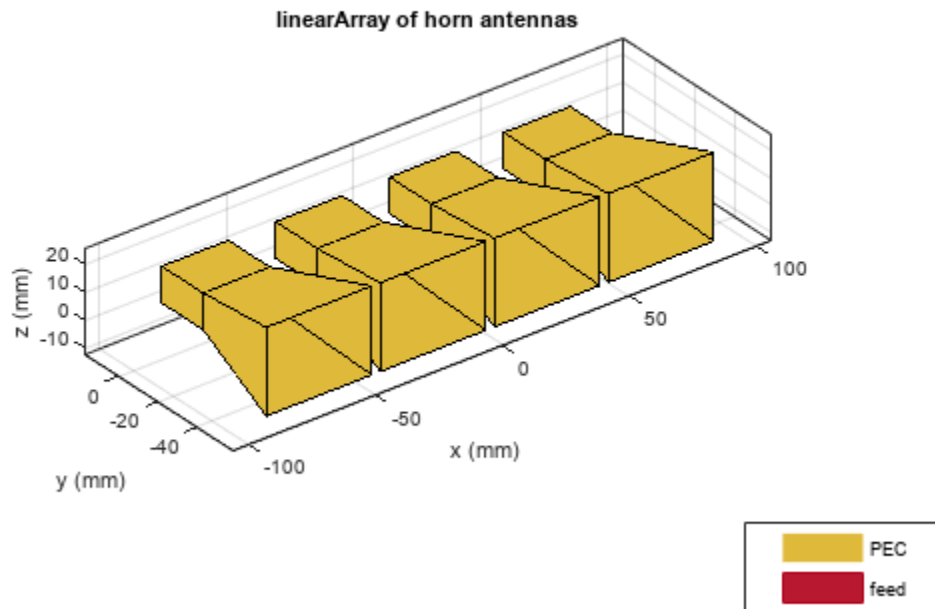
```



Create Linear Array of Horns

Create a linear array of horns. Set the number of elements in the array to 4 and the spacing between the elements to 1.5 times of lambda.

```
arr = linearArray('Element',h,'NumElements',4);  
arr.Element.Tilt = -90;  
arr.Element.TiltAxis = 'Z';  
arr.ElementSpacing = 1.5*lambda;  
figure;  
show(arr)
```

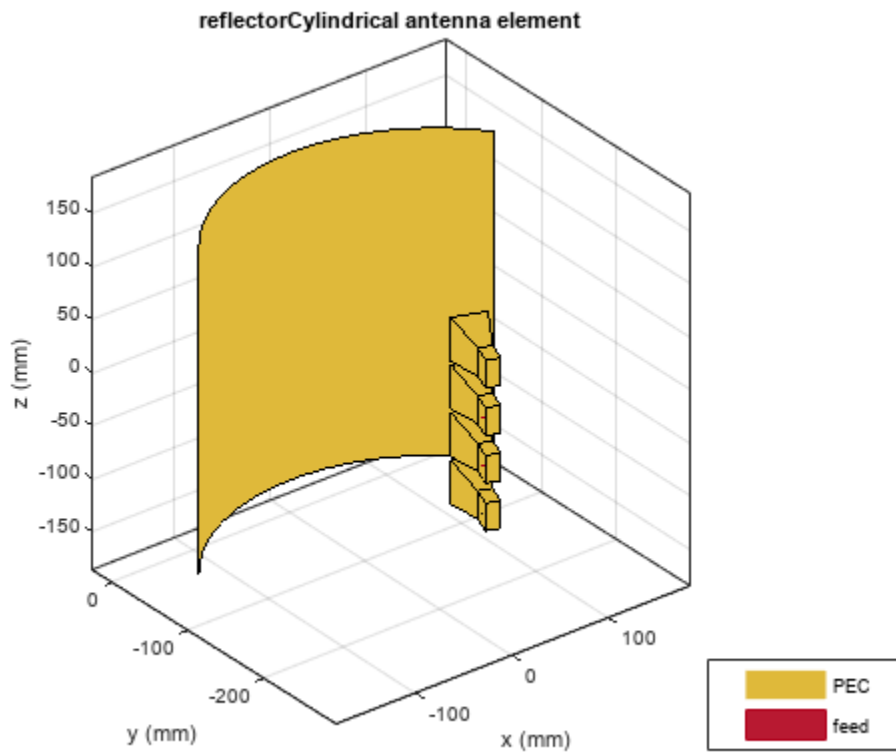


Create Cylindrical Reflector with Horn Array

Create a cylindrical reflector with the linear array of horns as the exciter. Tilt the exciter at an angle of 90 degrees along the Z-axis and 70 degrees along the Y-axis.

```

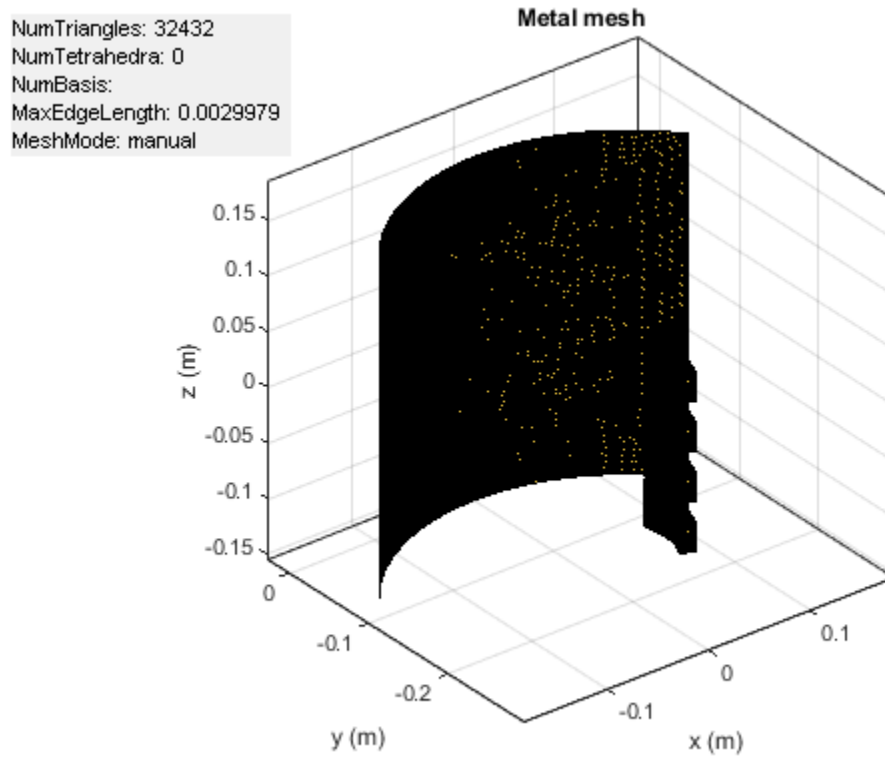
ref = reflectorCylindrical;
ref.GroundPlaneLength = 10.2*lambda;
ref.GroundPlaneWidth = 10.3*lambda;
ref.Spacing = 8.5*lambda;
ref.Tilt = 90;
ref.Exciter = arr;
ref.Exciter.Tilt = [90 70];
ref.Exciter.TiltAxis = ['Z', 'Y'];
figure;
show(ref)
  
```



Mesh and Plot Radiation Pattern of Cylindrical Reflector with Horn Array

Mesh the reflector manually with the maximum edge length of $\lambda/10$.

```
figure;  
mesh(ref, 'MaxEdgeLength', lambda/10);
```

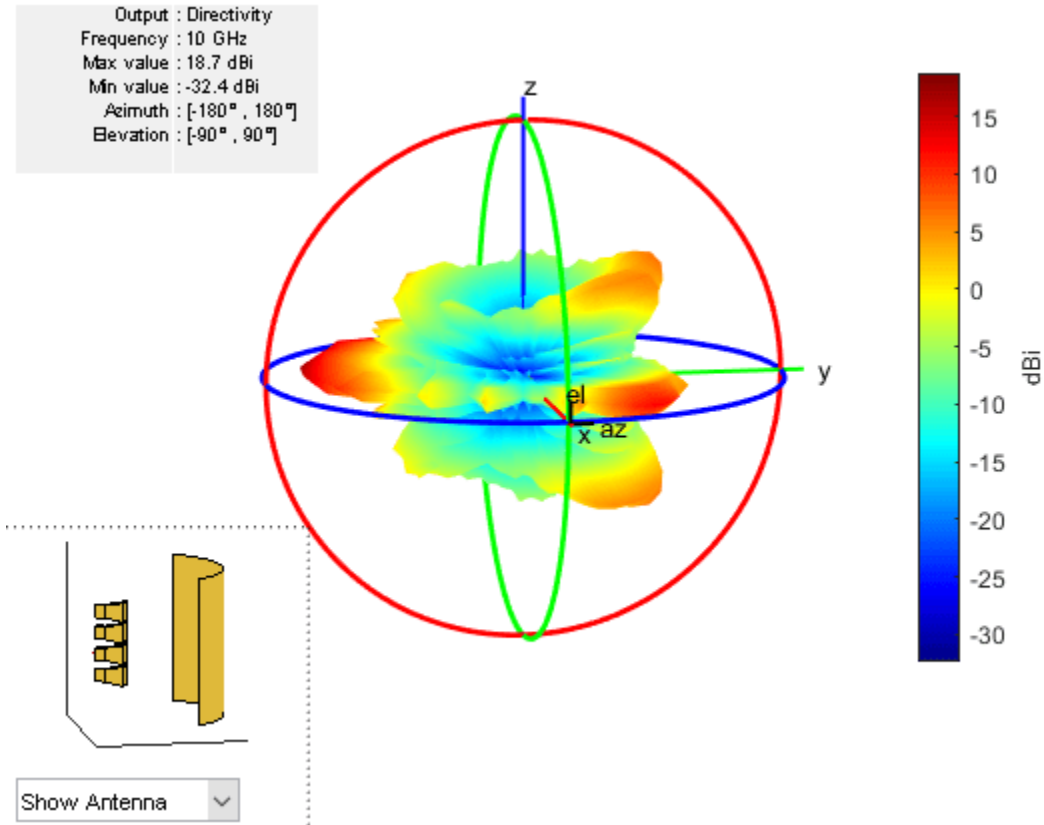


Load the solved reflector file ref0bj.mat.

```
load("ref0bj.mat");
```

Plot the radiation pattern.

```
figure;  
pattern(ref,freq);  
view([80 10]);
```



You can also plot the radiation pattern without using the MAT file, but the `pattern(ref, freq)` command takes up to 18 minutes to plot the radiation pattern. To run the `pattern(ref, freq)` command, these machine configurations are preferred:

- **Processor:** Intel® Xeon® CPU E5-1650 v4 @3.60GHz.
- **RAM:** 64GB.
- **System type:** 64-bit operating system.

The directivity of the cylindrical reflector antenna with a horn array feed is 18.7 dB.

See Also

`reflectorCylindrical`

Related Examples

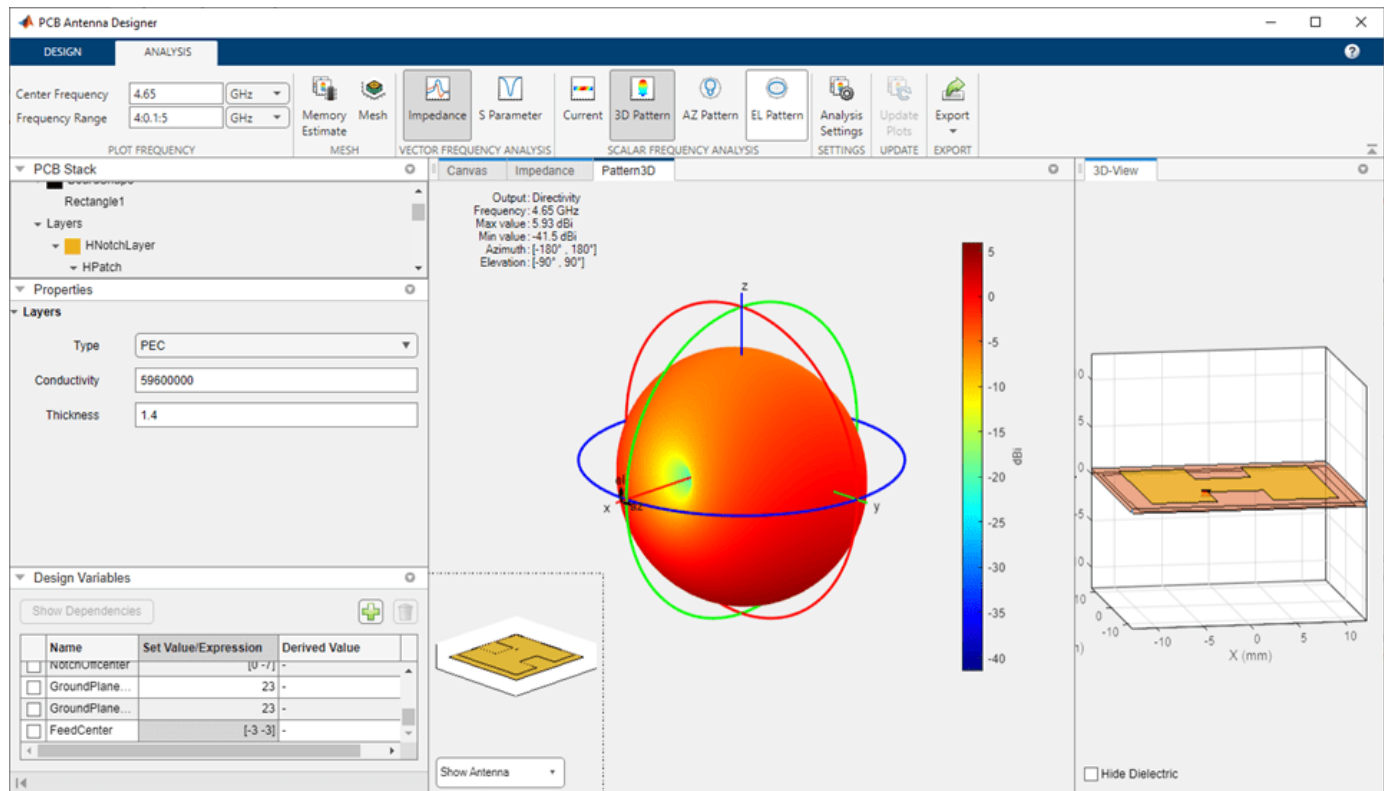
- “Design and Analyze Curved Reflectors” on page 5-705

References

- [1] Zhang, R, Z. Xie and Q. Chu. "A New Horn Array Feed for Parabolic Cylindrical Reflector Antenna." *2007 Asia-Pacific Microwave Conference, 2007*, pp.1-4. doi: 10.1109/APMC.2007.4555169.

Design and Analysis Using PCB Antenna Designer

This five part tutorial shows how to design, validate, analyze, optimize and export a 1-by-2 linear array of H-notch patch antennas resonating at 4.65 GHz using the PCB Antenna Designer app. The first four parts of this tutorial are sequenced as: define the PCB layers, design and analyze the unit element, create a linear array, and export the design. The fifth part of this tutorial allows you to recreate the same H-notch patch using variables for property values and optimize the design for maximum gain.



To get started, follow these steps:

- 1 Define PCB Antenna layers:** Define the metal patch, substrate, and ground plane layers for the linear array of 1-by-2 H-notch patch PCB antenna. For more information, see “Define PCB Antenna Layers” on page 5-813.
- 2 Design H-notch patch unit element:** Design the H-notch patch unit element by subtracting the top and bottom notches from the substrate. Add a feed to the H-notch patch unit element and validate the board shape, layers, feed, via, and load. For more information, see “Design H-Notch Patch Unit Element” on page 5-818.
- 3 Analyze H-notch patch unit element:** Perform vector frequency analysis on the H-notch patch unit element to calculate the impedance and the S-parameters over its frequency range of operation. Perform scalar frequency analysis on the H-notch patch unit element to calculate its current distribution, 3-D pattern, and azimuth and elevation patterns. For more information, see “Analyze H-Notch Unit Element” on page 5-831.
- 4 Create, analyze, and export 1-by-2 linear array of H-notch patch:** Create a 1-by-2 linear array with the H-notch patch as unit element. Add the feed and validate the design before

analysis. Analyze the design to confirm that it meets the design goals and export the design to Gerber files for the fabrication of the PCB patch antenna. For more information, see “Design, Analyze, and Export 1-by-2 H-Notch Linear Array” on page 5-836.

- 5 **Design, analyze, and optimize H-notch patch using design variables:** Design the same H-notch patch using variables for the properties of the board, metal layers, and patch. Create these variables using the **Design Variables** tab. Analyze the design to see that the results match the previous patch. Optimize the design for maximum gain. For more information see, “Design, Analyze, and Optimize H-Notch Patch Using Design Variables” on page 5-852.

See Also

Apps

PCB Antenna Designer

Objects

pcbStack

Related Examples

- “Antenna Optimization Algorithm” on page 3-29
- “Minimize Area of Dipole Antenna to Optimize Gain Using Surrogate Optimization Method”
- “Optimization of Antenna Array Elements Using Antenna Array Designer App” on page 5-636

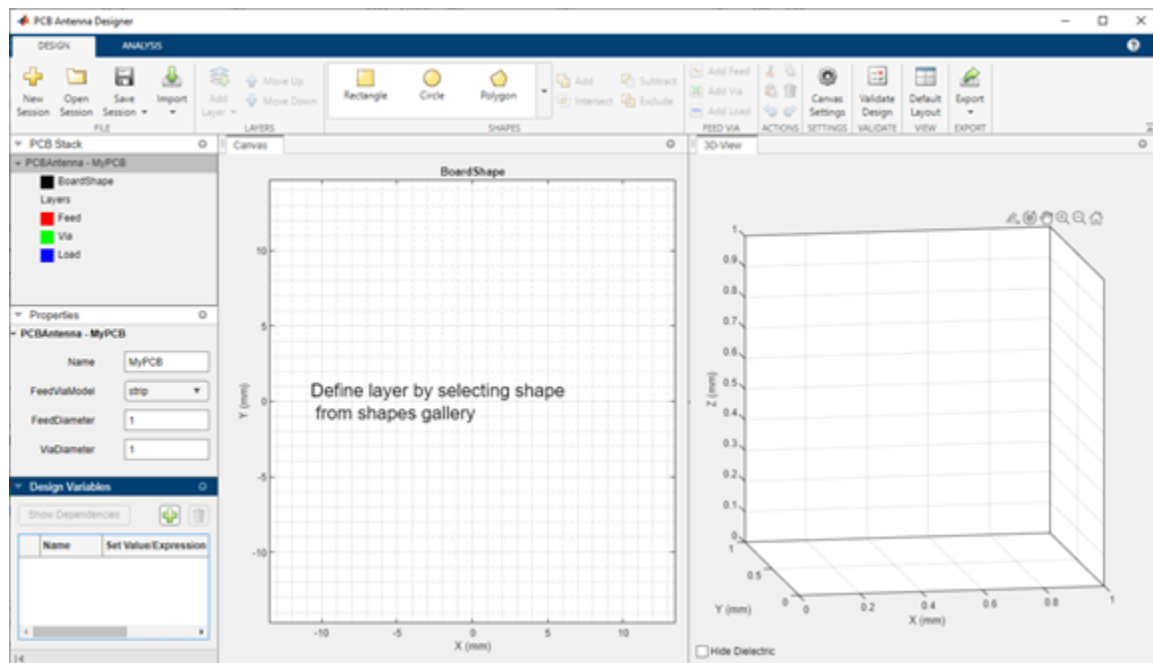
Define PCB Antenna Layers

This step of the tutorial shows how to define the metal patch, dielectric, and ground plane layers in a PCB antenna using the PCB Antenna Designer app.

Open New Session

Type this command at the command line to open the **PCB Antenna Designer** app.

```
pcbAntennaDesigner
```



On the **Design** tab, click **New Session** to start a new session and open a blank canvas.

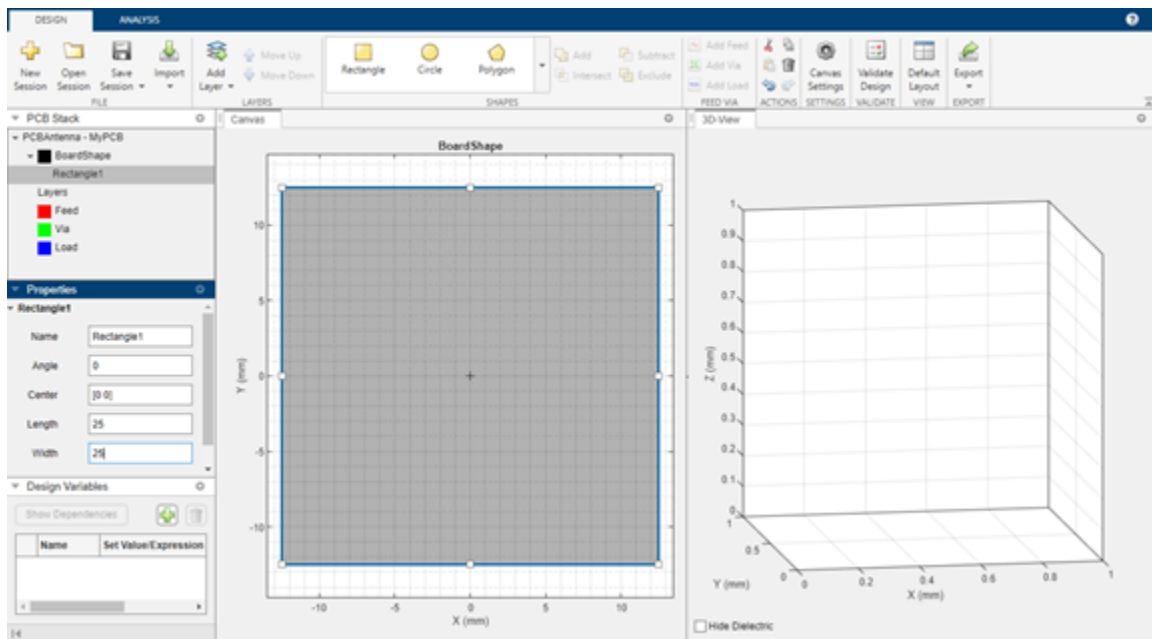
The default units for the canvas are millimeters (mm). The app uses global units. You can change the units by clicking the **Canvas Settings** button.

Define Board Shape

To design a PCB stack, you must first define a board shape. Select **Rectangle** from the Shapes section on the toolbar. Drag the shape on the canvas to create a rectangle.

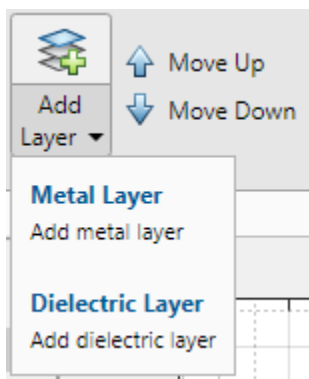
Change the properties of Rectangle1 in the **Properties** pane to the following:

- **Center** — $[0, 0]$
- **Length** — 25
- **Width** — 25

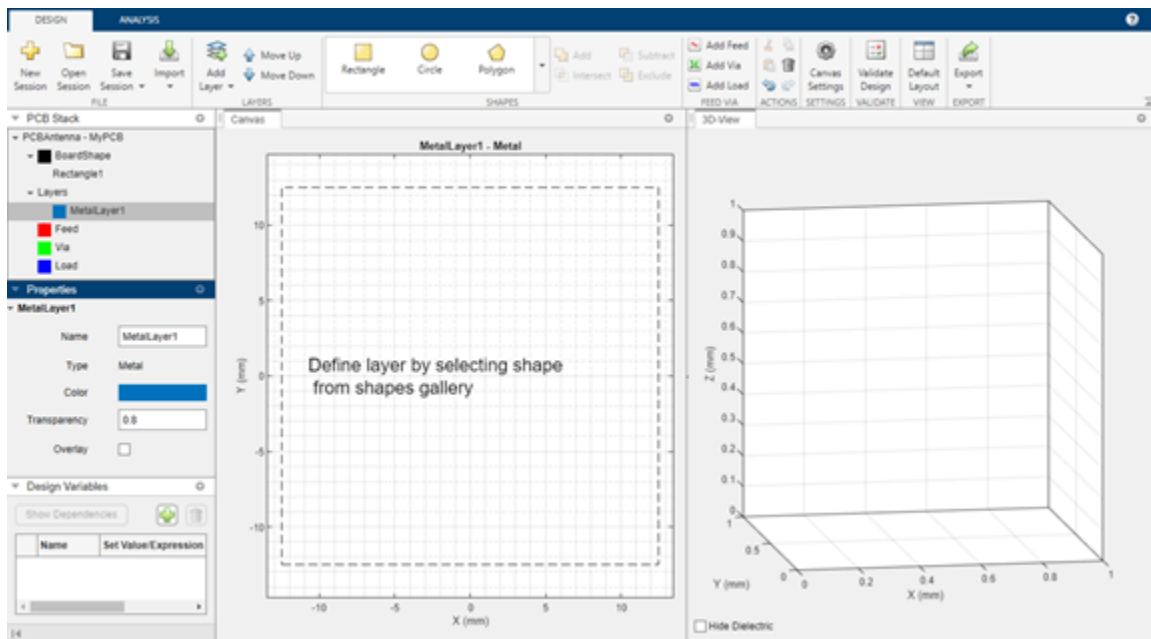


Add Ground Plane

Click **Add Layer** on the toolbar and then select **Metal Layer** to add a metal layer.



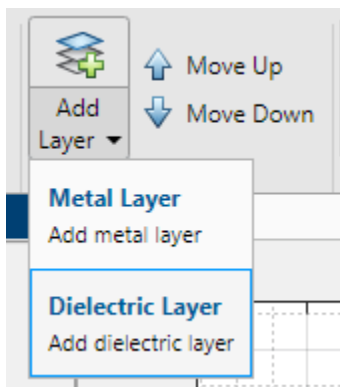
The board shape is represented as a dotted line on the canvas.

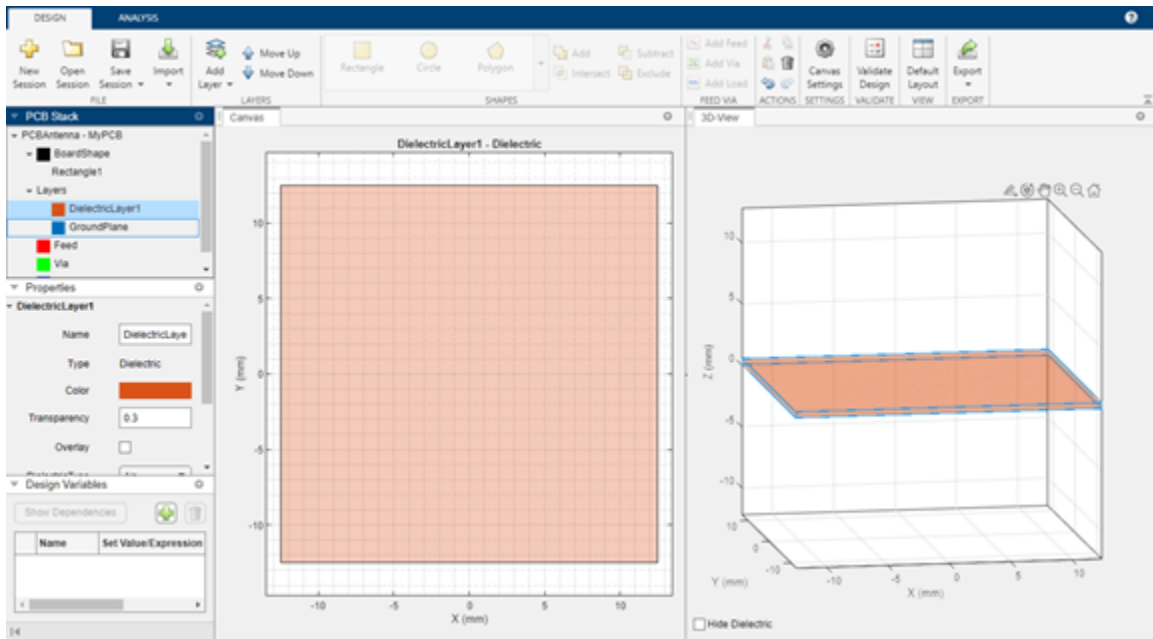


Set the **Name** of the metal layer to GroundPlane.

Add Dielectric Layer

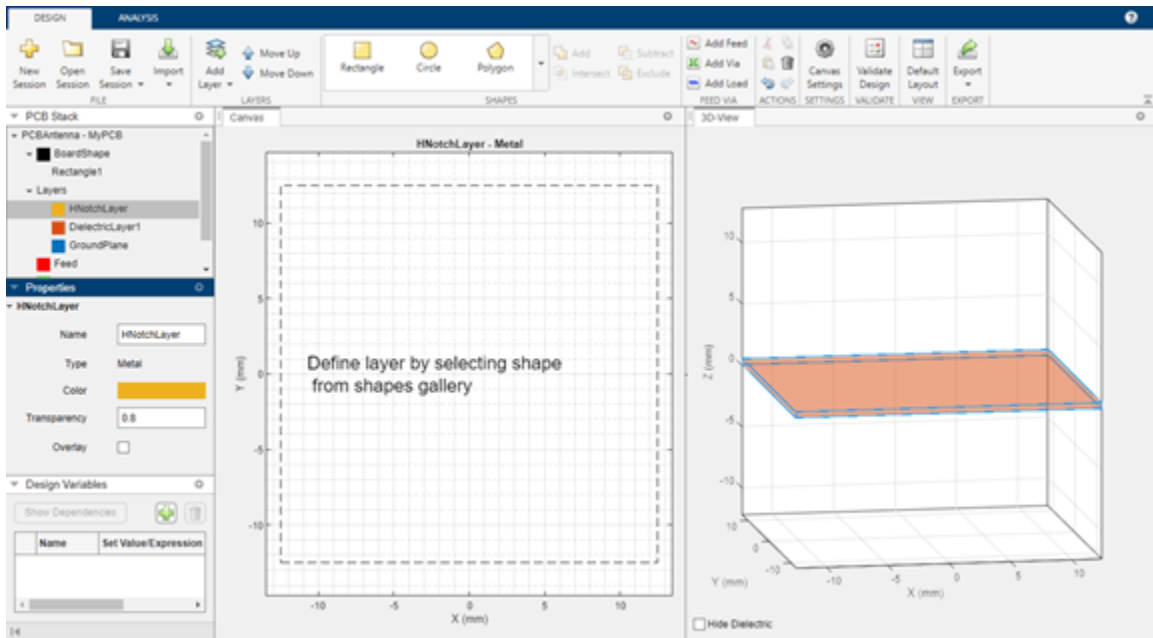
Click **Add Layer** on the toolbar and then select **Dielectric Layer** to add a dielectric layer DielectricLayer1.





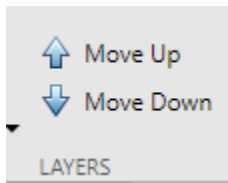
Add Metal Patch Layer

Click **Add Layer** on the toolbar and then select **Metal Layer** to add a metal layer. Set the name of this layer to HNotchLayer.



Move Layers

You can move the layers using the **Move Up** and **Move Down** buttons on the toolbar.



See Also

Objects

`antenna.Rectangle` | `pcbStack`

Related Examples

- “Dielectric Catalog”
- “Metal Catalog”

Design H-Notch Patch Unit Element

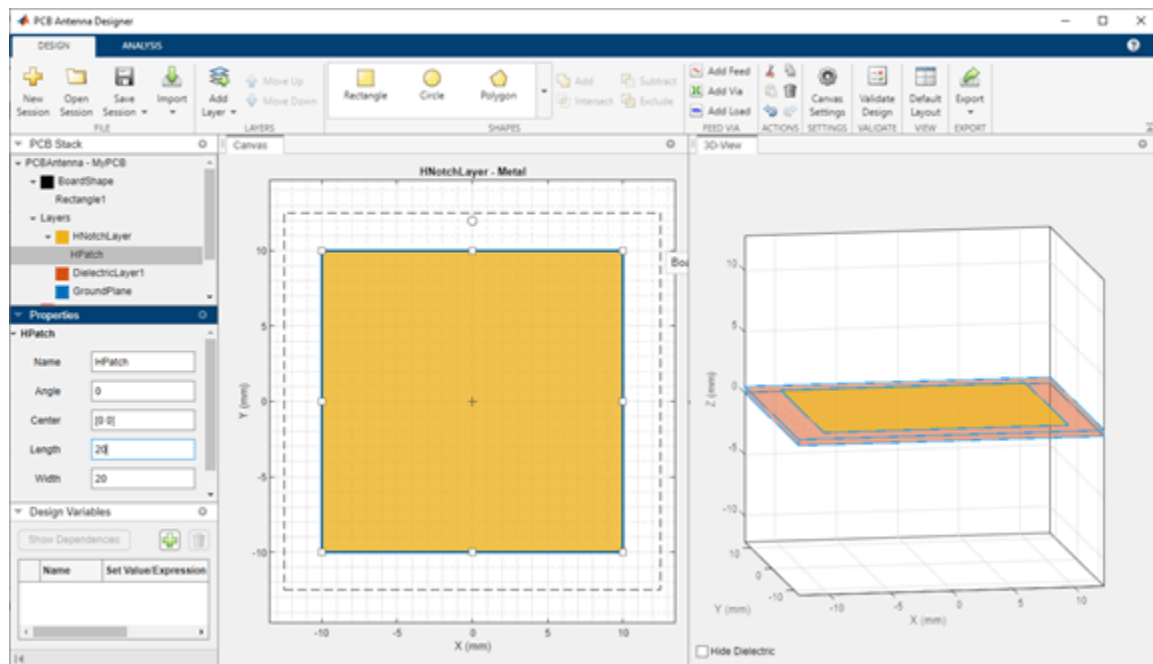
This step of the tutorial shows how to design an unit element for a 1-by-2 H-notch linear array. You design the H-notch unit element by subtracting the top and bottom notches from the substrate.

Create H-Notch

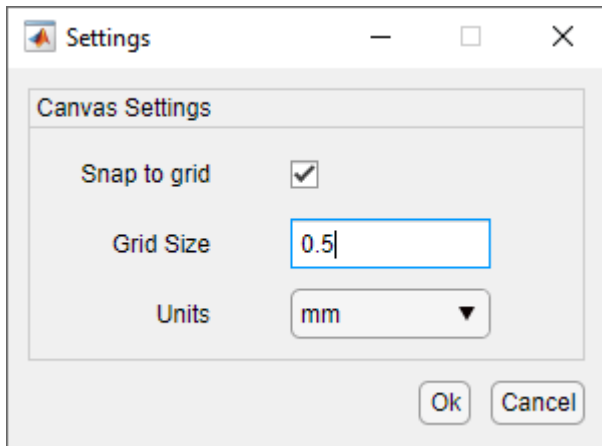
Select the HNotchLayer on the PCB Stack navigation tree and then select **Rectangle** from the **Shapes** section on the toolbar. Drag the shape onto the canvas to create a rectangular patch.

Set the properties of Rectangle2 to the following:

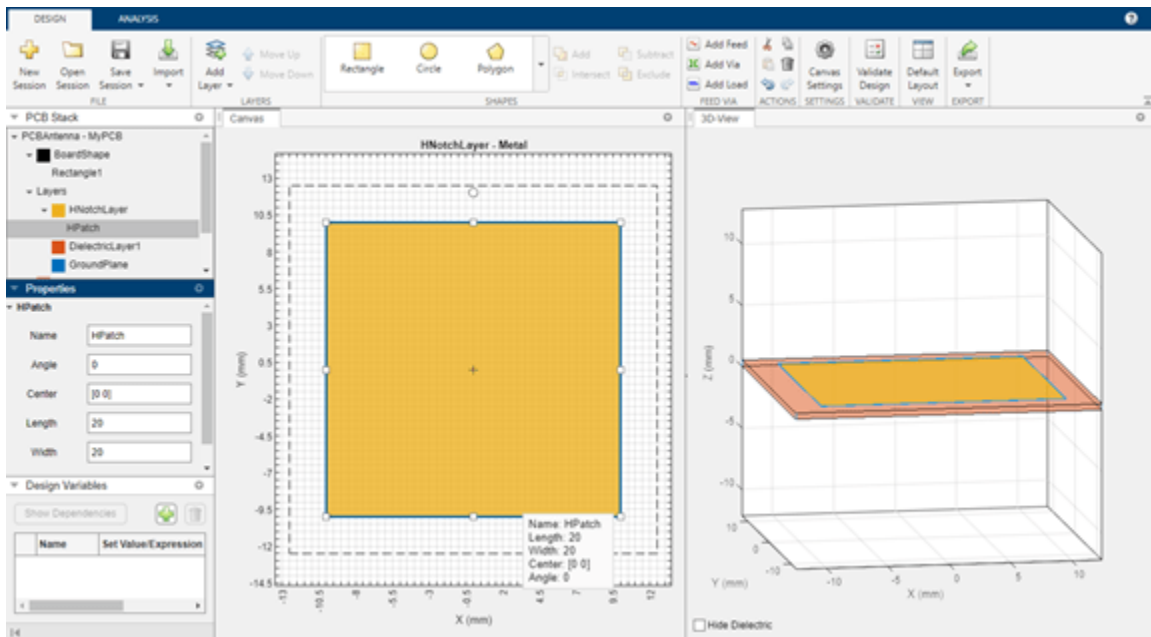
- **Name** — HPatch
- **Center** — [0, 0]
- **Length** — 20
- **Width** — 20



Click the **Canvas Settings** button and select **Snap to grid** to allow easier and more accurate drawing. Then set the **Grid Size** to 0.5 mm.



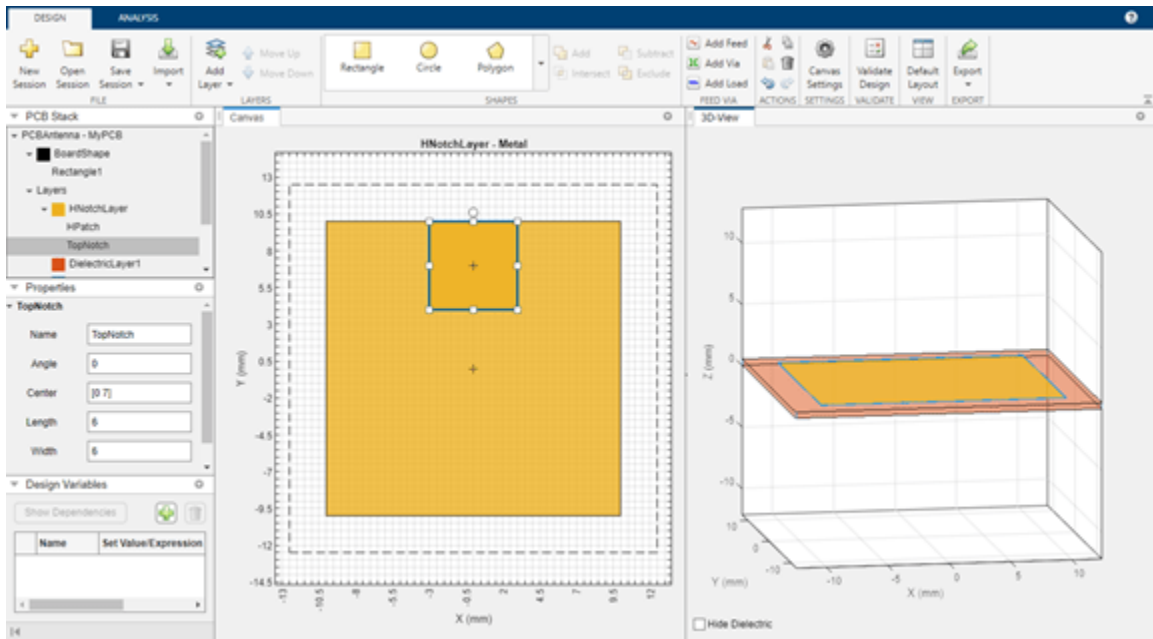
The app displays the drawn shape.



Create Top Notch

Select **Rectangle** from the **Shapes** section and drag the shape to the top of the shape to create a top notch in the patch layer. Set the properties of **Rectangle3** to the following:

- **Name** — TopNotch
- **Center** — [0, 7]
- **Length** — 6
- **Width** — 6

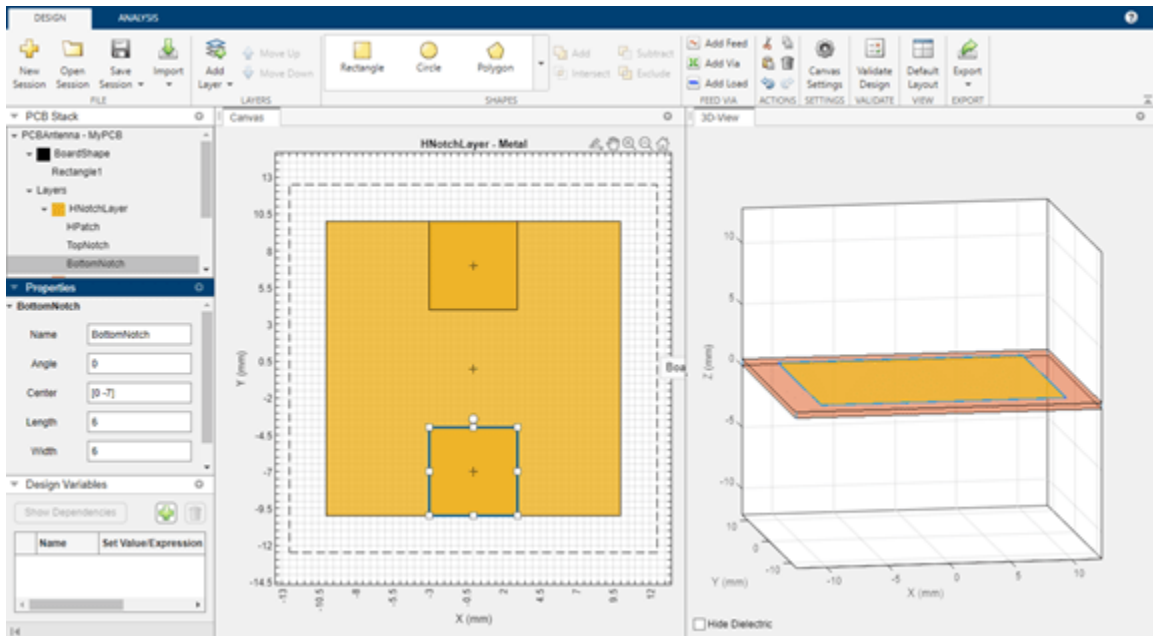


Create Bottom Notch

Select **TopNotch** from the canvas and then copy the layer by clicking **Copy** in the **Actions** section. Click **Paste** to paste a copy of **TopNotch**. Doing so creates a rectangle **TopNotch_Copy_1** with identical dimensions to the **TopNotch** layer.

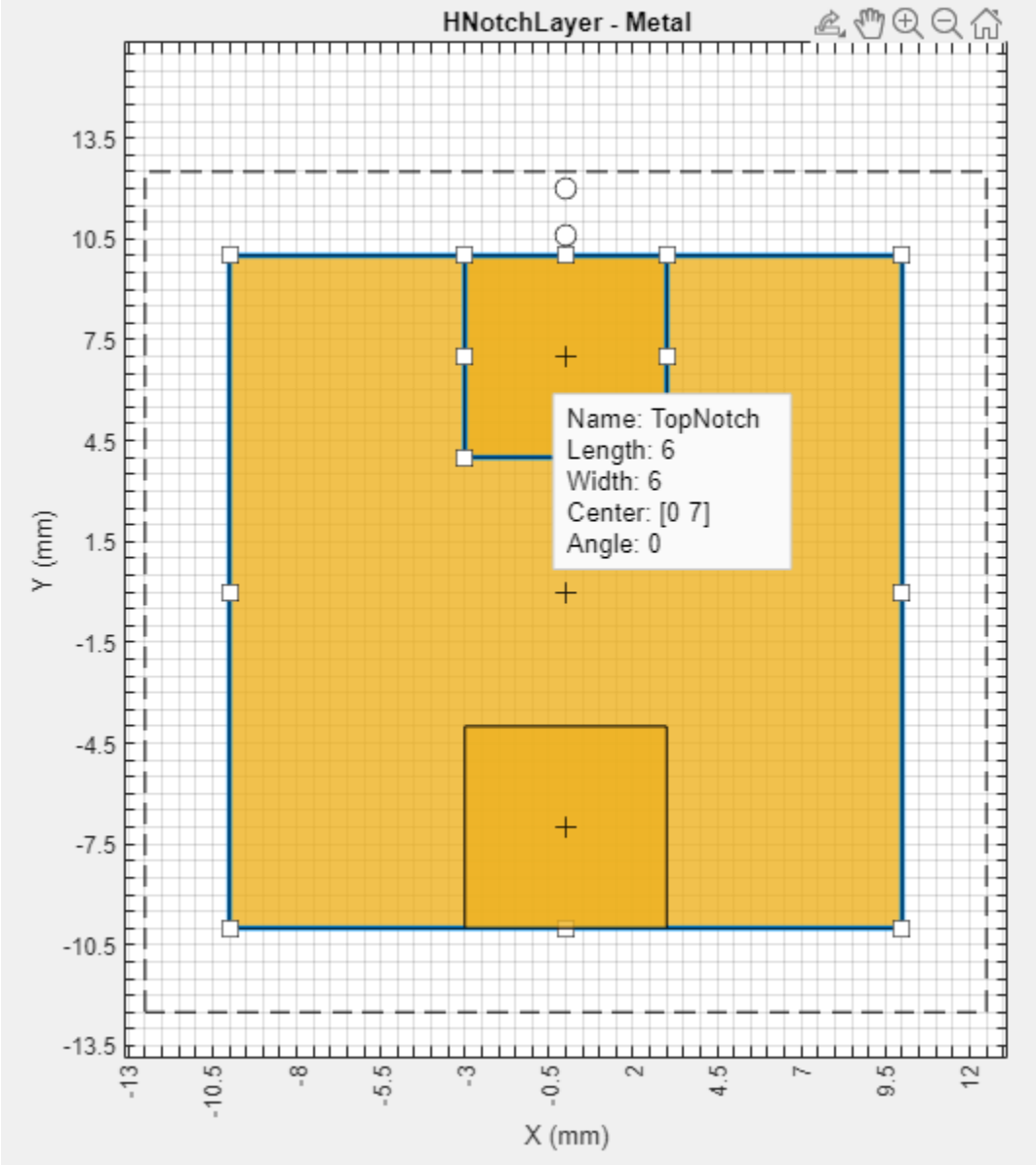
Set the properties of **TopNotch_Copy_1** to the following:

- **Name** – BottomNotch
- **Center** – [0, -7]



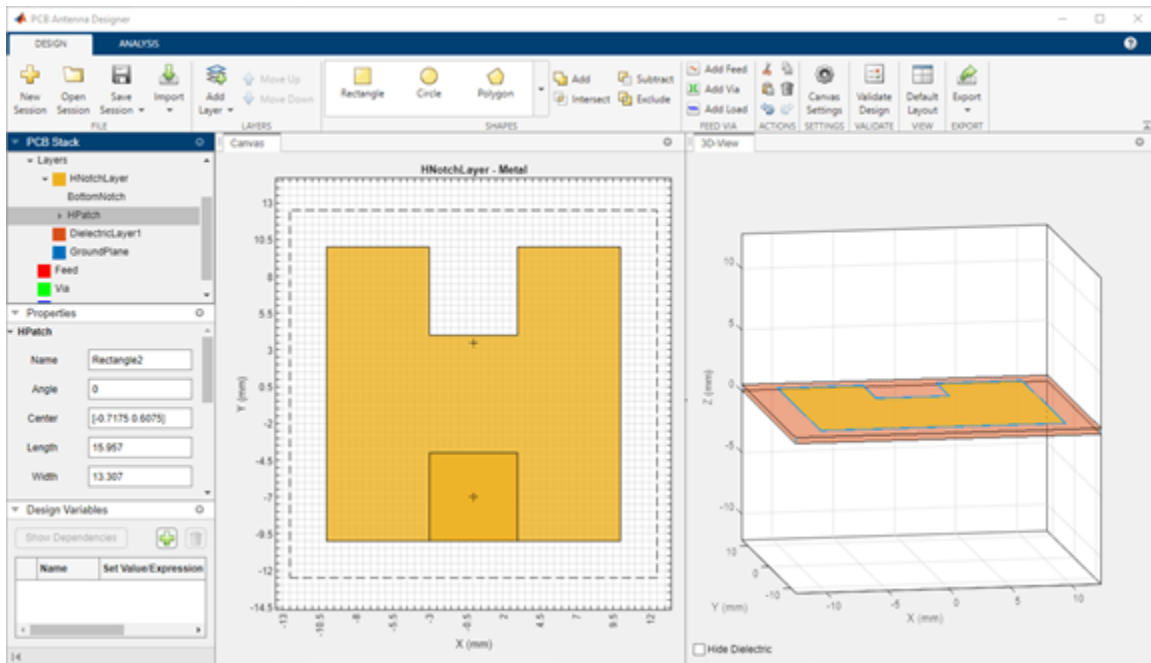
Subtract Top and Bottom Notch Layers from Metal Patch

Select HPatch and TopNotch and then click **Subtract** in the **Shapes** section of the toolbar to remove both rectangles.

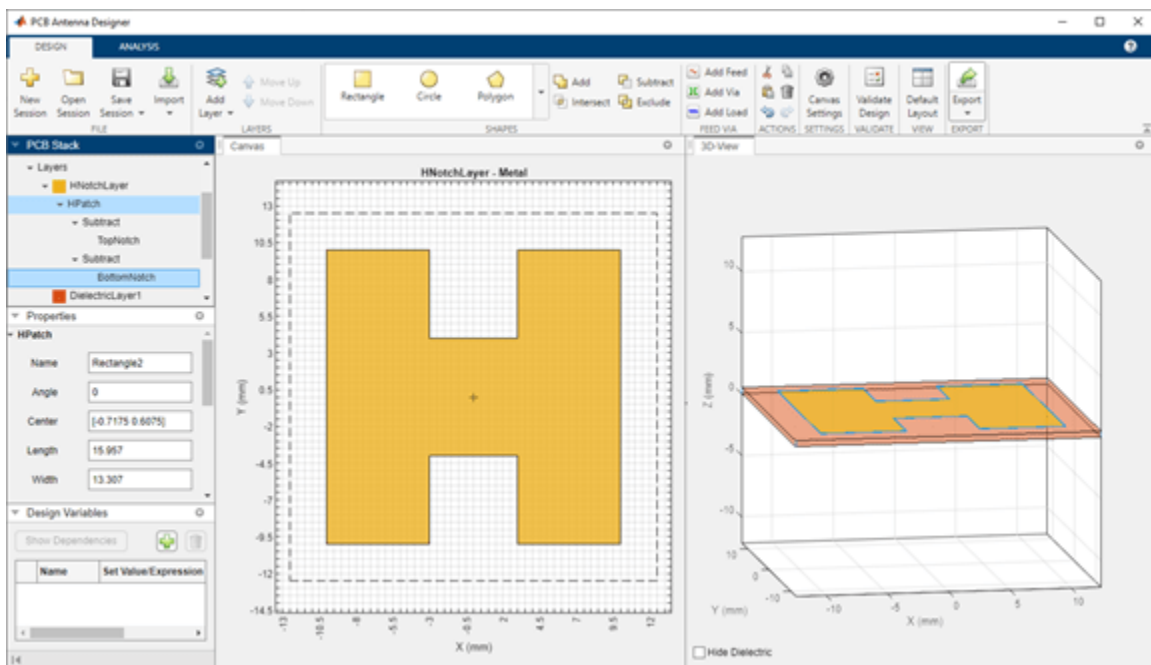


The app updates the HPatch layer.

5 Antenna Toolbox Examples

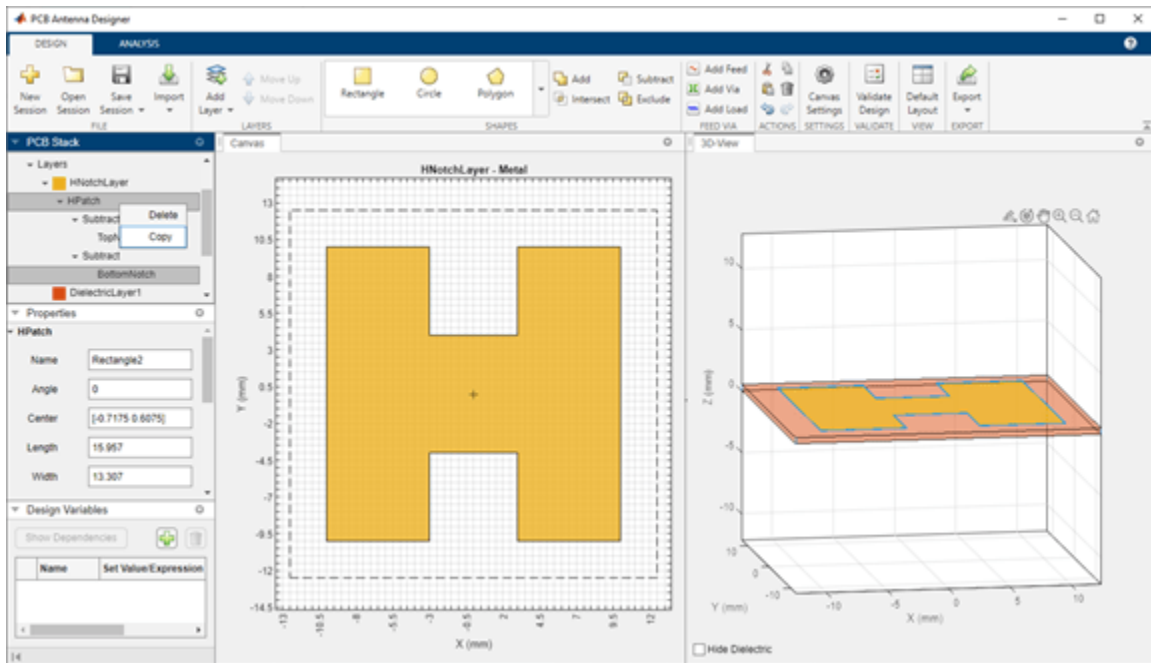


Select the HPatch and BottomNotch and click **Subtract** in the **Shapes** section of the toolbar to remove both rectangles.

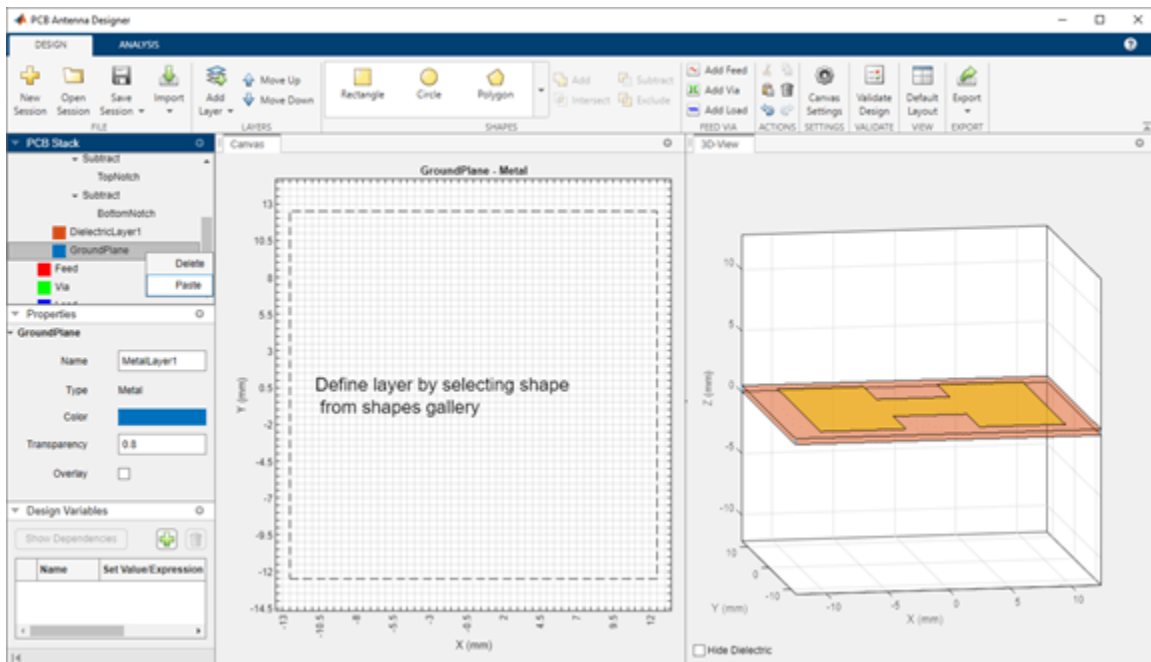


Set Dimensions for Ground Plane

The ground plane must be same size as the HNotchLayer. To make the ground plane the same size as HPatch, right-click HPatch and select **Copy**.



Right-click the GroundPlane and click **Paste** on the toolbar to paste the rectangle in the GroundPlane.

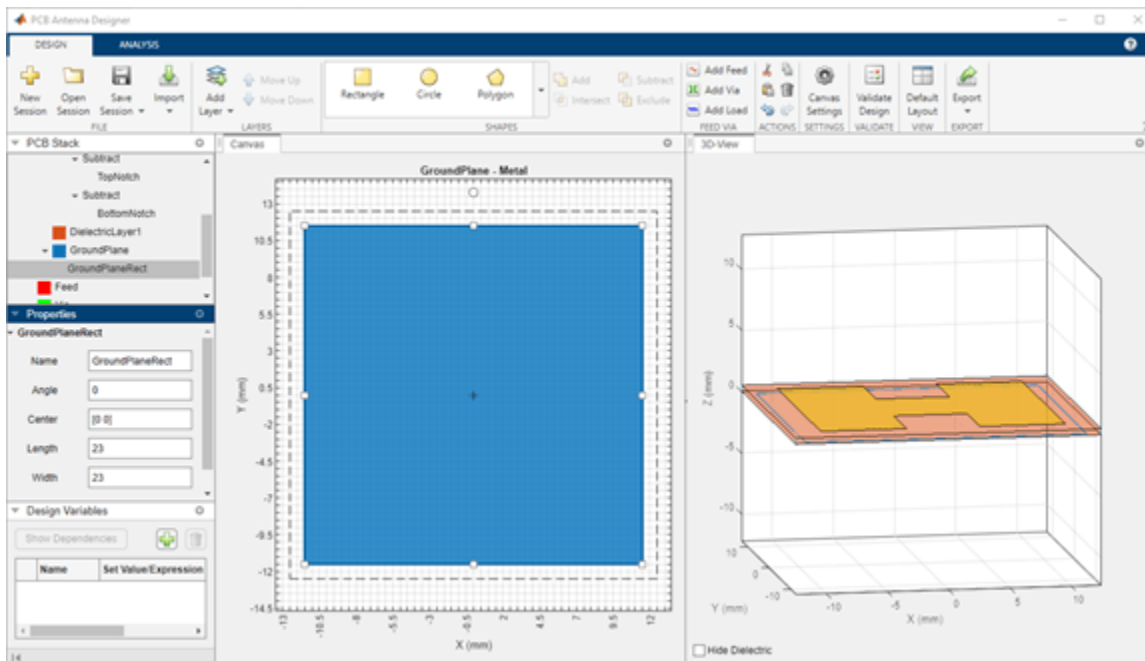


The app copies the rectangle to the ground plane layer with the name HPatch_Copy(1). HPatch_Copy(1) and HPatch have the same dimensions.

Set the properties of HPatch_Copy(1) to the following:

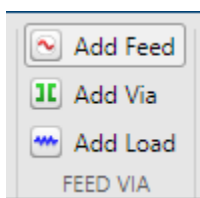
- **Name** — GroundPlaneRect

- **Center** — [0,7]
- **Length** — 23
- **Width** — 23

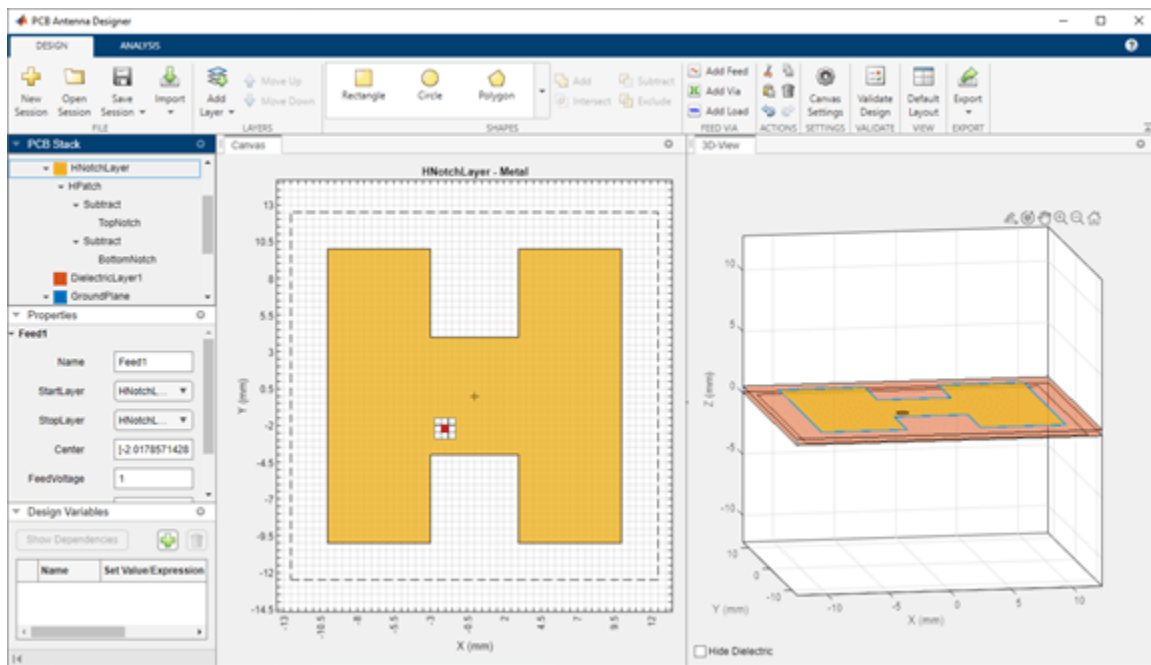


Add Feed

Select the HNotchLayer metal layer on the **PCB Stack** navigation tree and then click **Add Feed** in the **Feed Via** section on the toolbar. You can add a feed only to a metal layer.



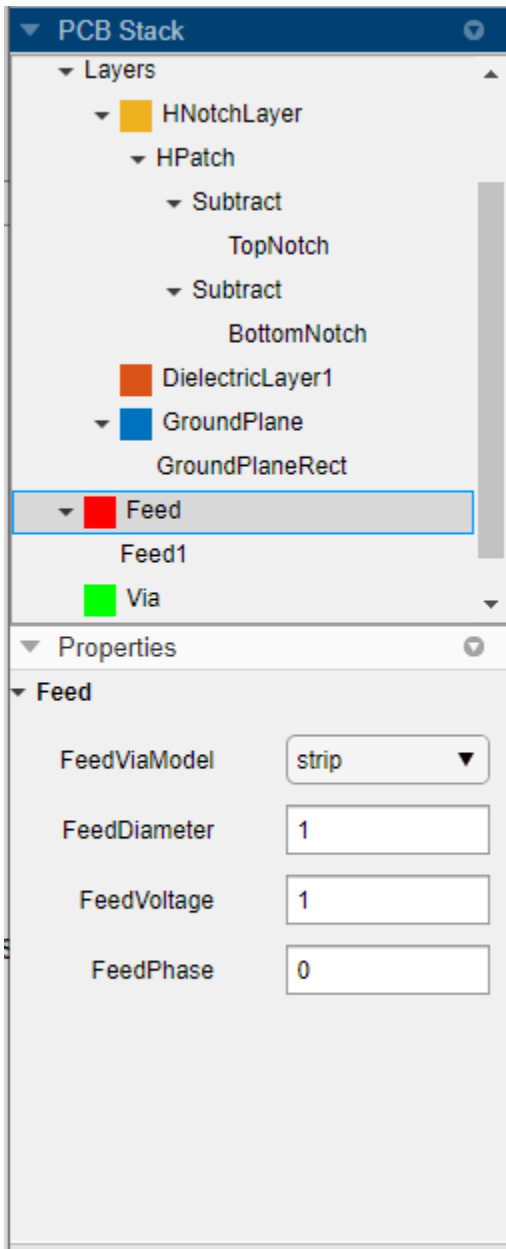
As the PCB Stack navigation tree shows, the app adds the feed to HNotchLayer as Feed1.



For more information on feeds and vias, refer to the **FeedLocation** and **ViaLocation** properties in the documentation of the **pcbStack** object.

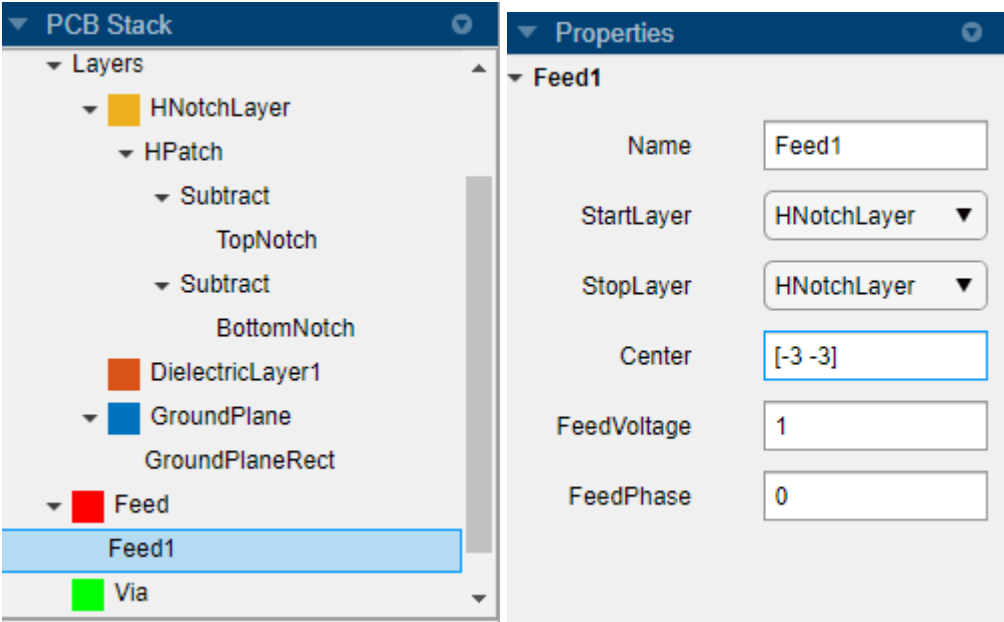
Change Properties of Feed

Select the parent **Feed** node on the **PCB Stack** navigation tree, and then set the **FeedDiameter** to 1 and **FeedViaModel** to **Strip**. **FeedDiameter** is a global property.

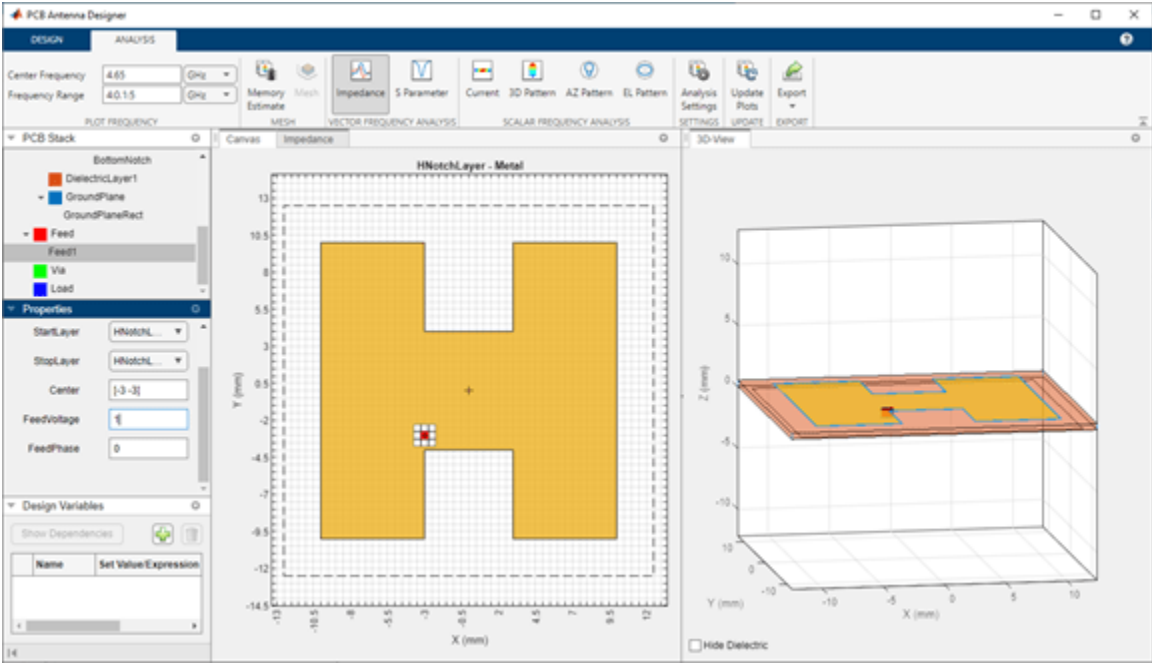


Change Position of Feed

Select Feed1 on the PCB Stack navigation tree. In the **Properties** pane, set **Center** to [-3, -3].

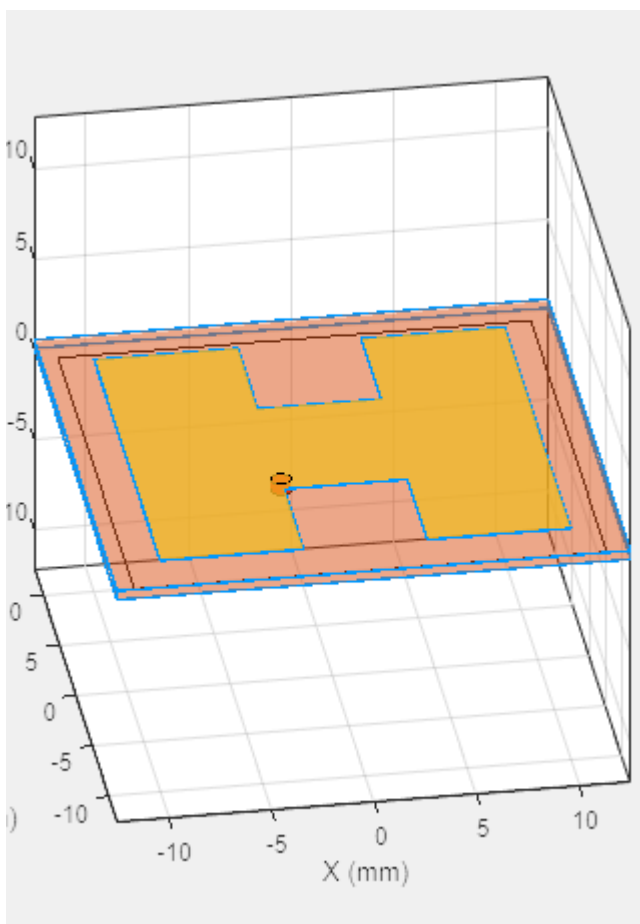
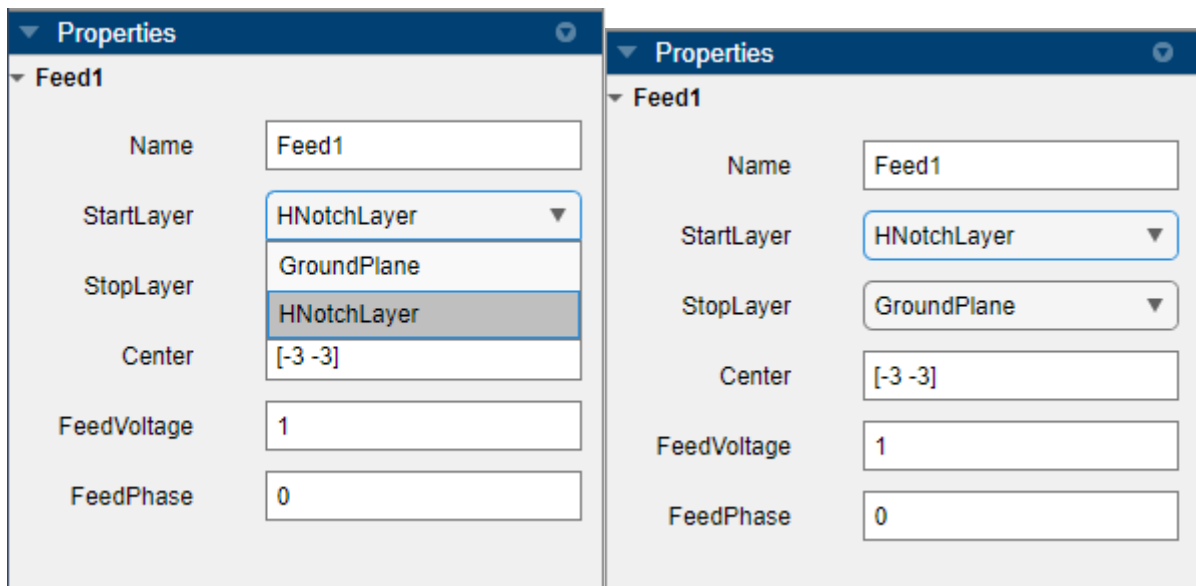


In the **3D-View** pane, you can see that the feed is connected only to HNotchLayer.



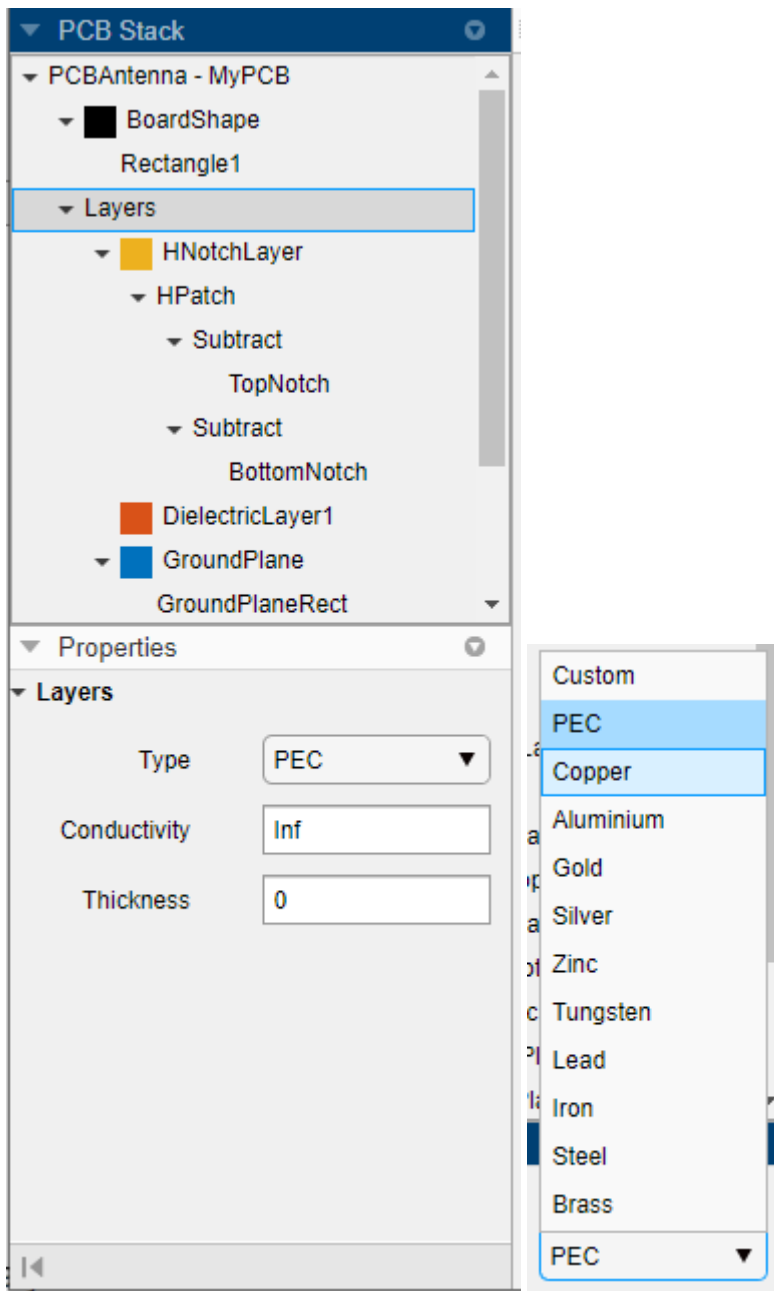
Change Start Layer and Stop Layer

Connect the feed from HNotchLayer to GroundPlane by setting **StartLayer** to HNotchLayer and **StopLayer** to GroundPlane. This creates a feed on the GroundPlane which connects to the HNotchLayer.



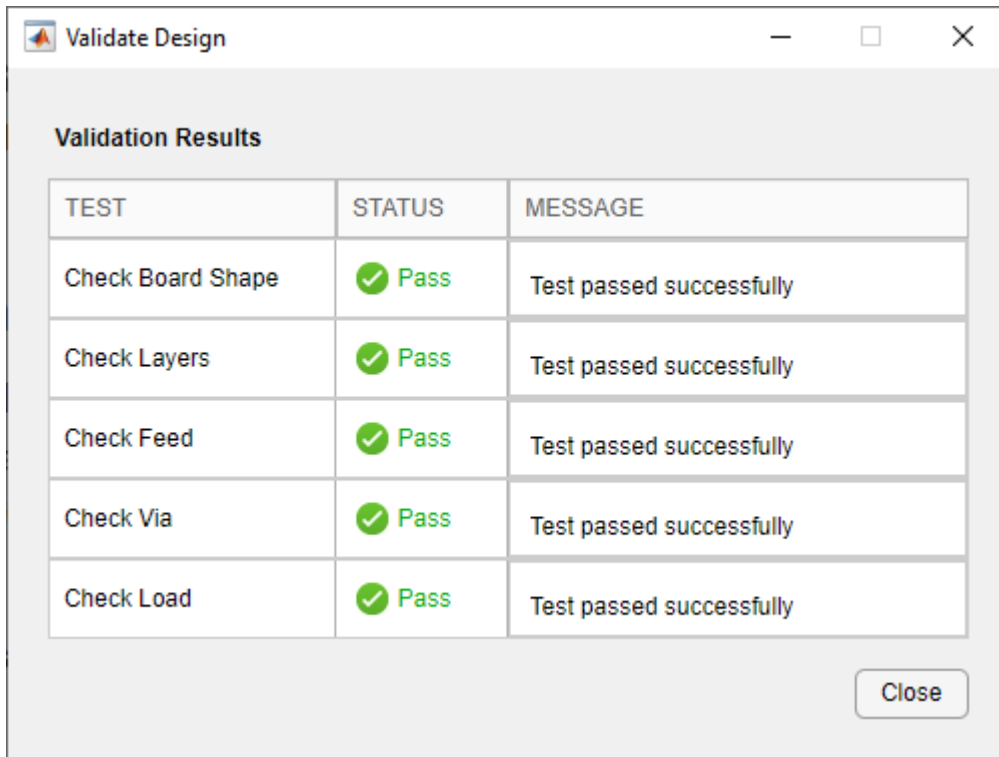
Change Metal Properties

Select the Layers in on the PCB Stack navigation tree and set the **Type** of the metal layer to Copper in the **Properties** pane.



Validate Design

Click **Validate Design** on the toolbar to validate your board shape, layers, feed, via, and load.



See Also

Objects

`antenna.Rectangle` | `pcbStack`

Functions

`subtract`

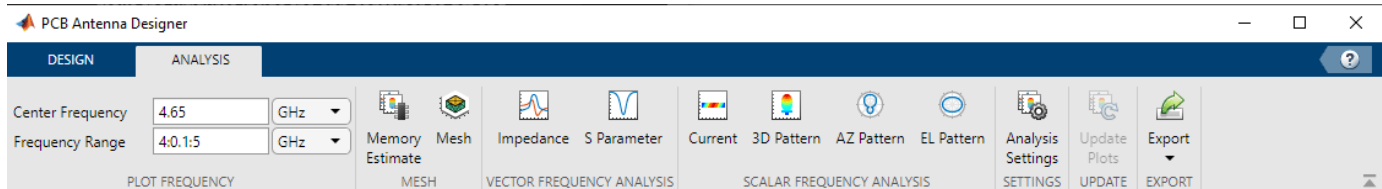
Related Examples

- "Dielectric Catalog"
- "Metal Catalog"

Analyze H-Notch Unit Element

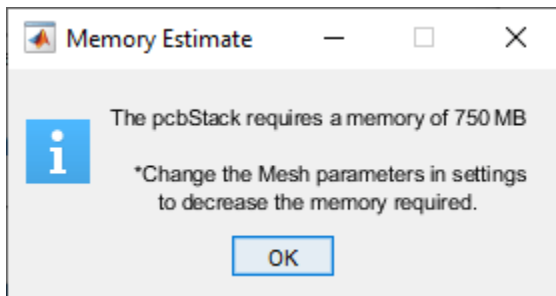
This step of the tutorial shows how to perform vector and scalar frequency analysis on a 1-by-2 H-notch linear array.

In the **Analysis** tab, set the **Center Frequency** to 4.65 GHz and **Frequency Range** to 4:0.1:5 GHz .



Memory Estimate

Click the **Memory Estimate** button from the **Mesh** section to estimate the memory you will require to run the analysis.

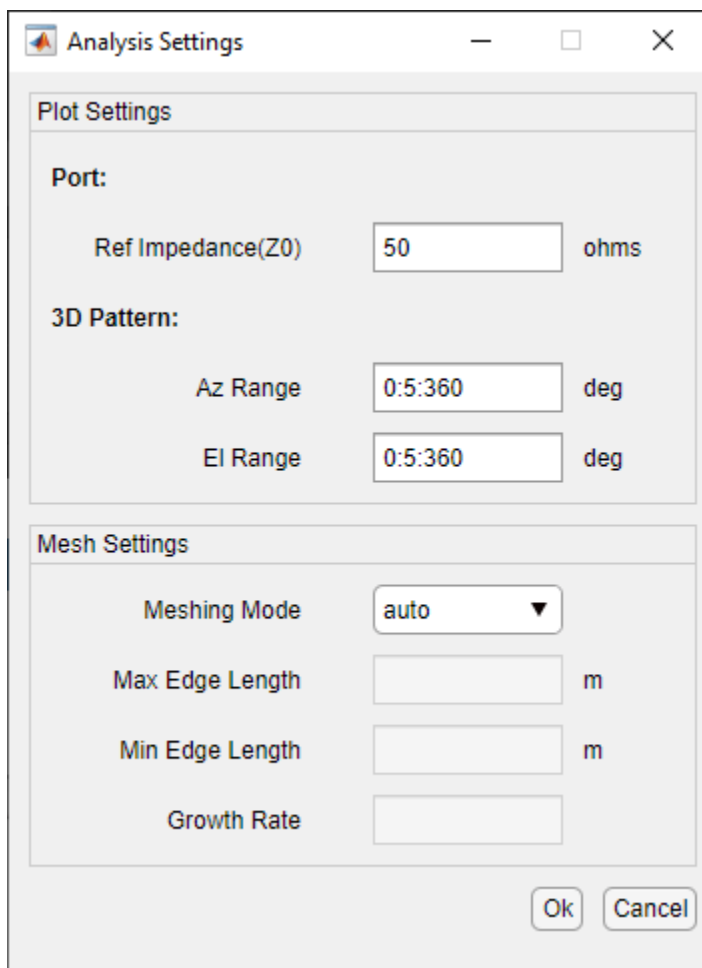


You will need a memory of 750 MB (RAM) to analyze the antenna design. You can alter the memory requirements by changing the mesh parameters. Select **Mesh** on the toolbar to alter the mesh settings.

Analysis Settings

Click **Analysis Settings** from the **Analysis** section on the toolbar to set the the plot and mesh parameters to the following:

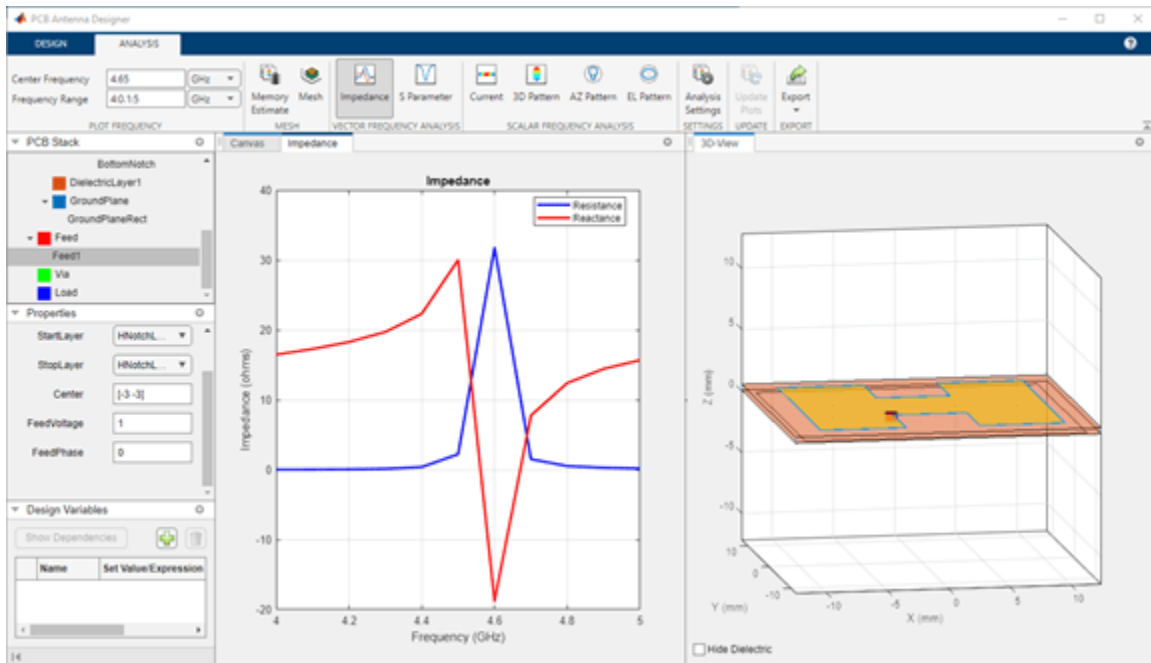
- **Ref Impedance(Z0)** — 50 ohms
- **Az Range** — 0:5:360 deg
- **El Range** — 0:5:360 deg
- **Meshing Mode** — auto



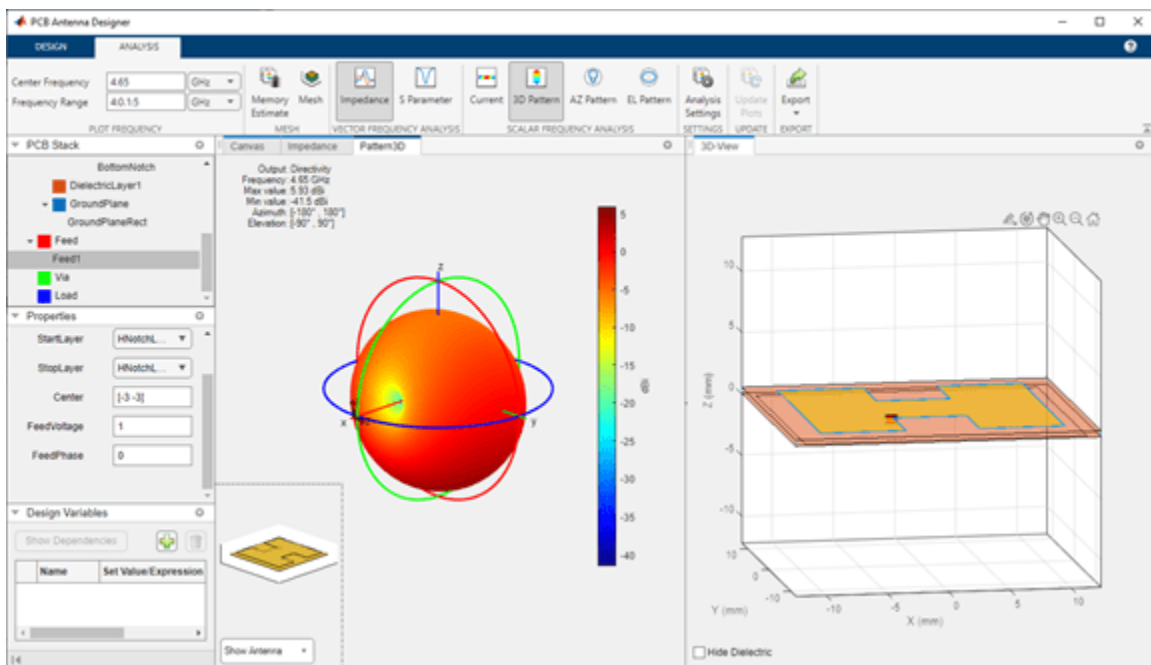
Click **OK** to save the settings.

Run Analysis

Select **Impedance** from the **Vector Frequency Analysis** section on the toolbar to plot the impedance plot. The impedance plot shows that the PCB stack antenna resonates at 4.6 GHz.

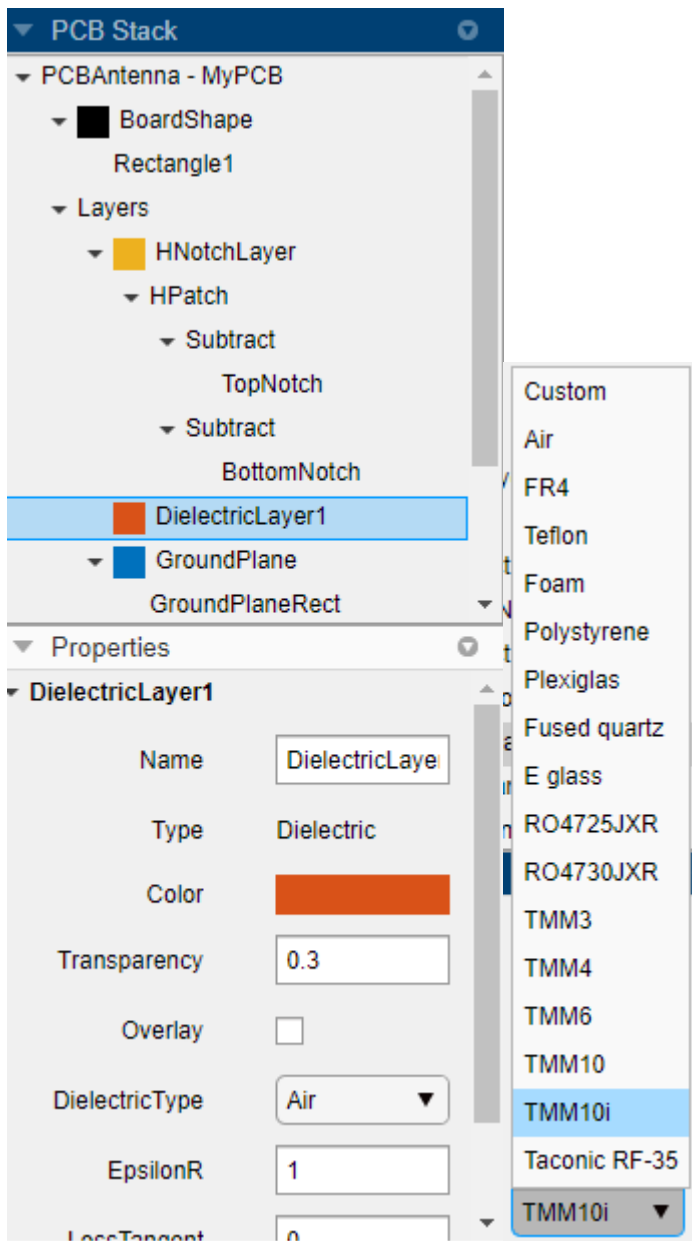


Click **3D Pattern** from the **Scalar Frequency Analysis** section on the toolbar to plot the 3-D far-field radiation pattern of the antenna. The directivity of the H-notch patch unit element is 5.93 dBi.

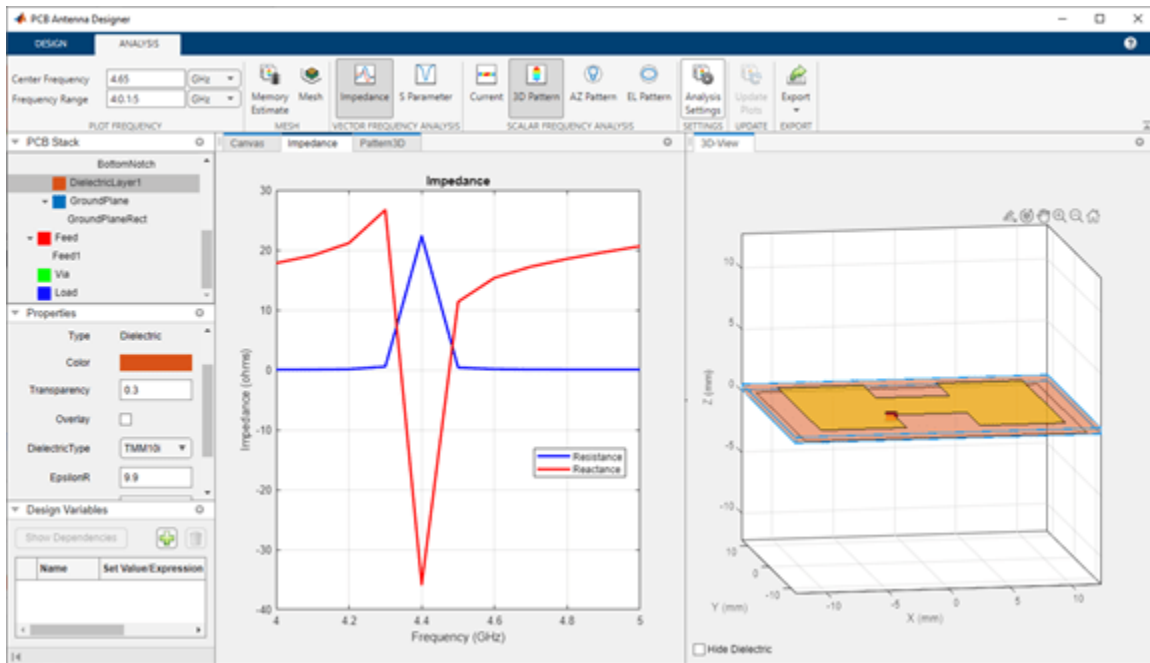


Update Plots

Select **DielectricLayer1** from the PCB Stack navigation tree and set the **DielectricType** to **TMM10i** in the **Properties** pane.



Click **Update Plots** from the **Update** section on the **Analysis** tab to update the analysis results. The impedance plot now shows that the PCB stack antenna resonates at 4.4 GHz.



See Also

Functions

mesh | impedance | pattern

Related Examples

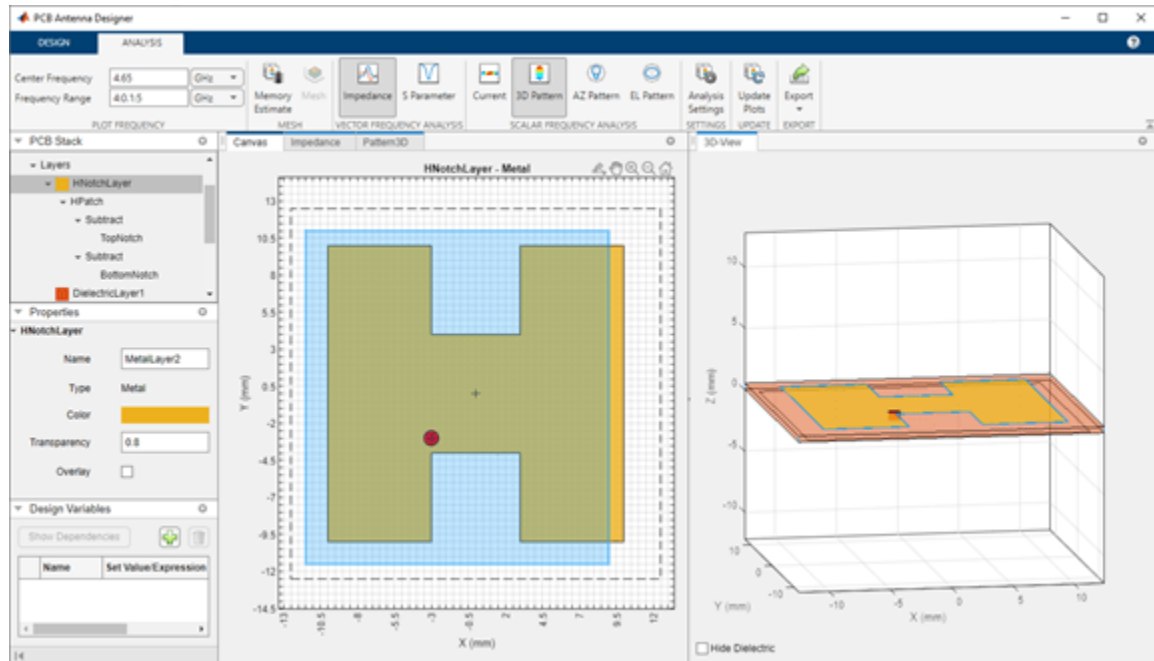
- “Design and Analyze X-Band Custom PCB Patch Antenna”

Design, Analyze, and Export 1-by-2 H-Notch Linear Array

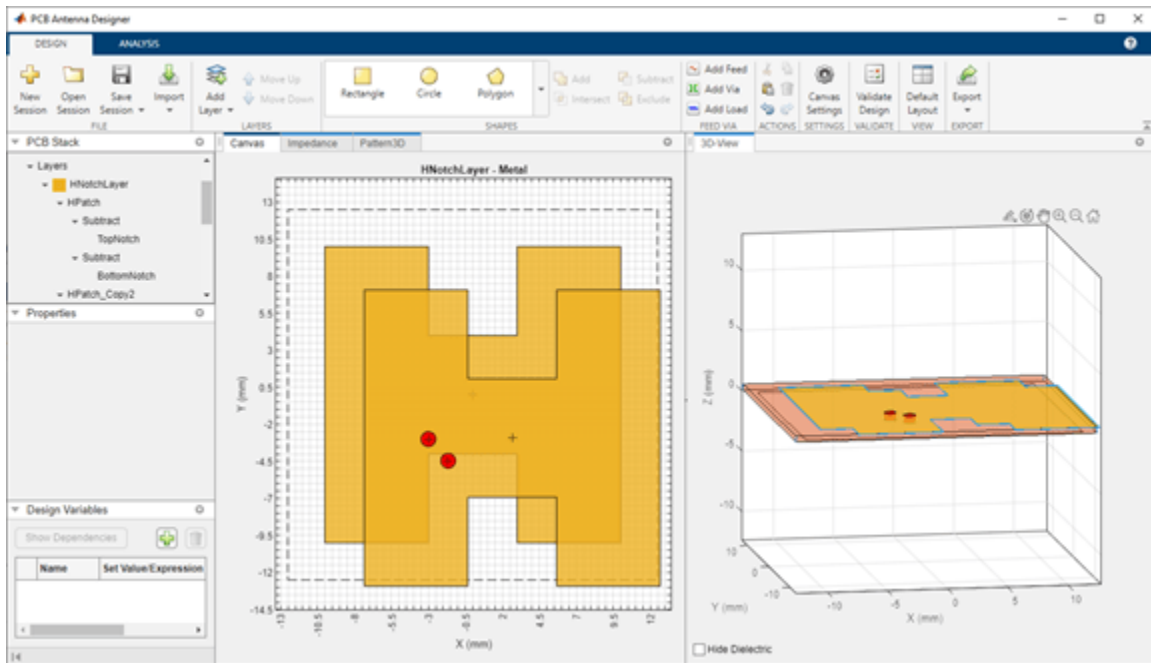
This step of the tutorial shows how to design, analyze, and export a 1-by-2 H-notch linear array.

Design 1-by-2 H-Notch Linear Antenna Array

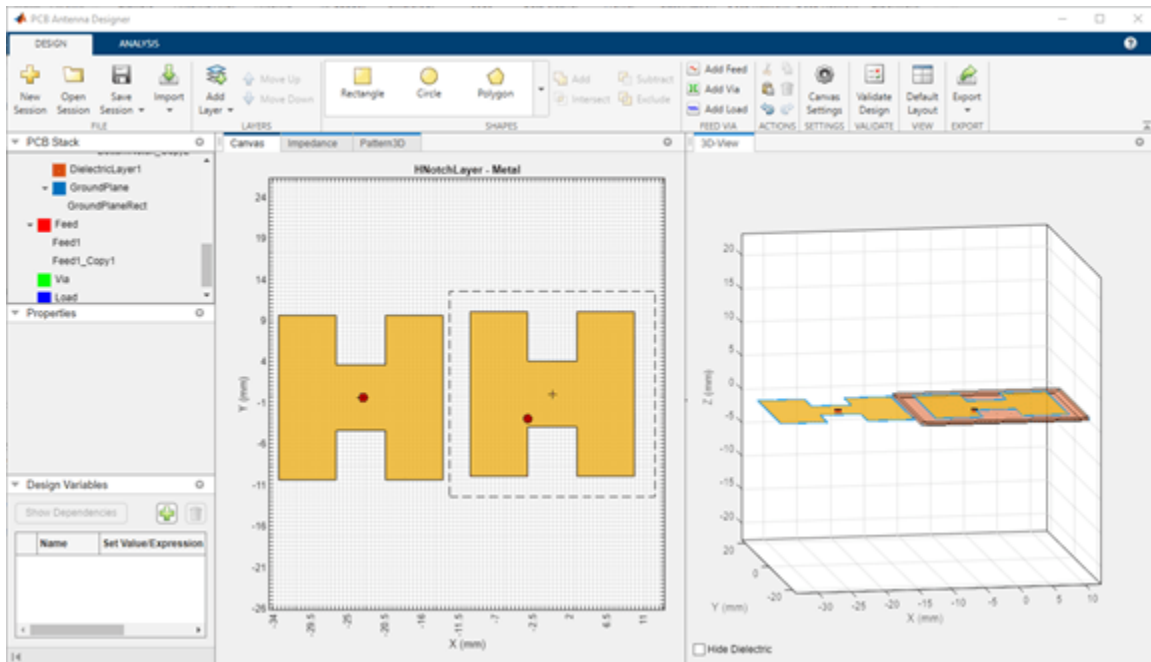
Select HNotchLayer in the **PCB Stack** pane and drag the axes to create a selection box. Drag the axes until the selection box encloses Feed and HNotchLayer.



Click **Copy** on the **Design** tab to copy the layers enclosed in the selection box and click **Paste** to create the HPatch_Copy (1) layer.

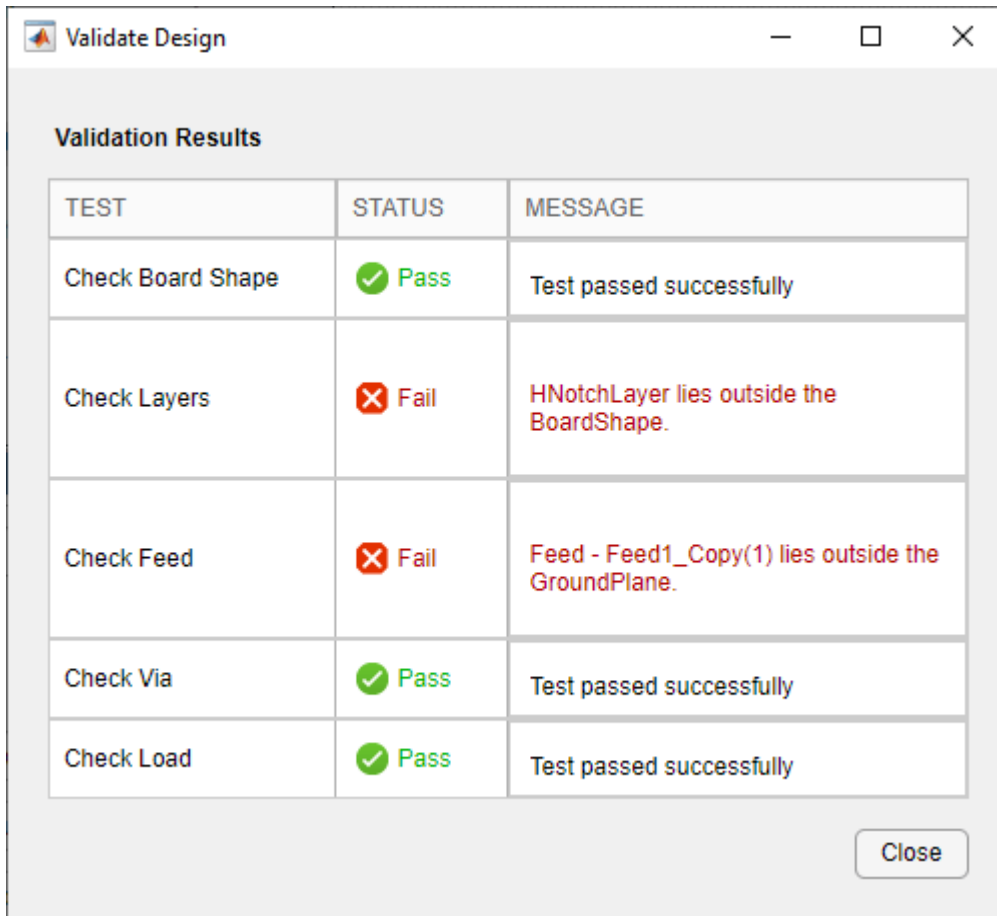


Select the HPatch_Copy (1) layer and drag it to place it to adjacent to the HPatch layer.



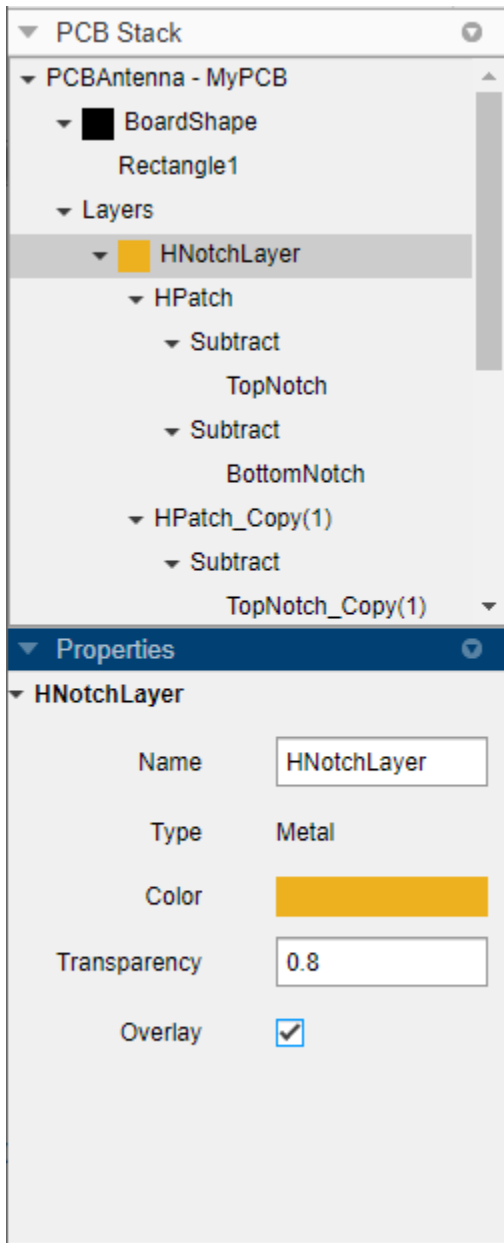
Validate Design for Errors

Click **Validate Design** on the toolbar to validate your board shape, layers, feed, via, and load. The **Check Layers** and **Check Feed** validation failed because HNotchLayer and Feed1_Copy (1) lie outside BoardShape and GroundPlane, respectively.



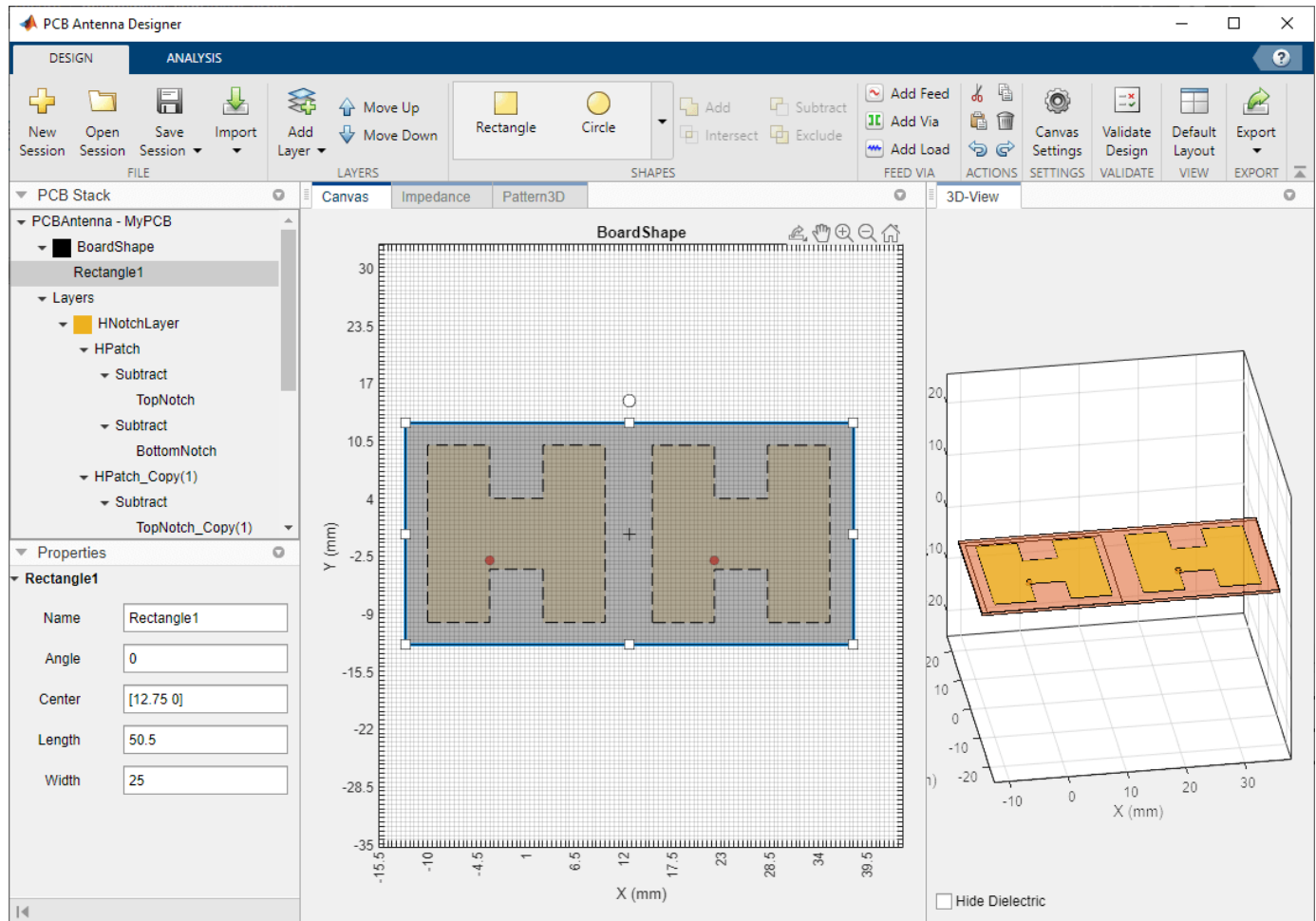
Overlay Layers

Select HNotchLayer and then select the **Overlay** check box in the **Properties** pane to create an image of HNotchLayer in the canvas.

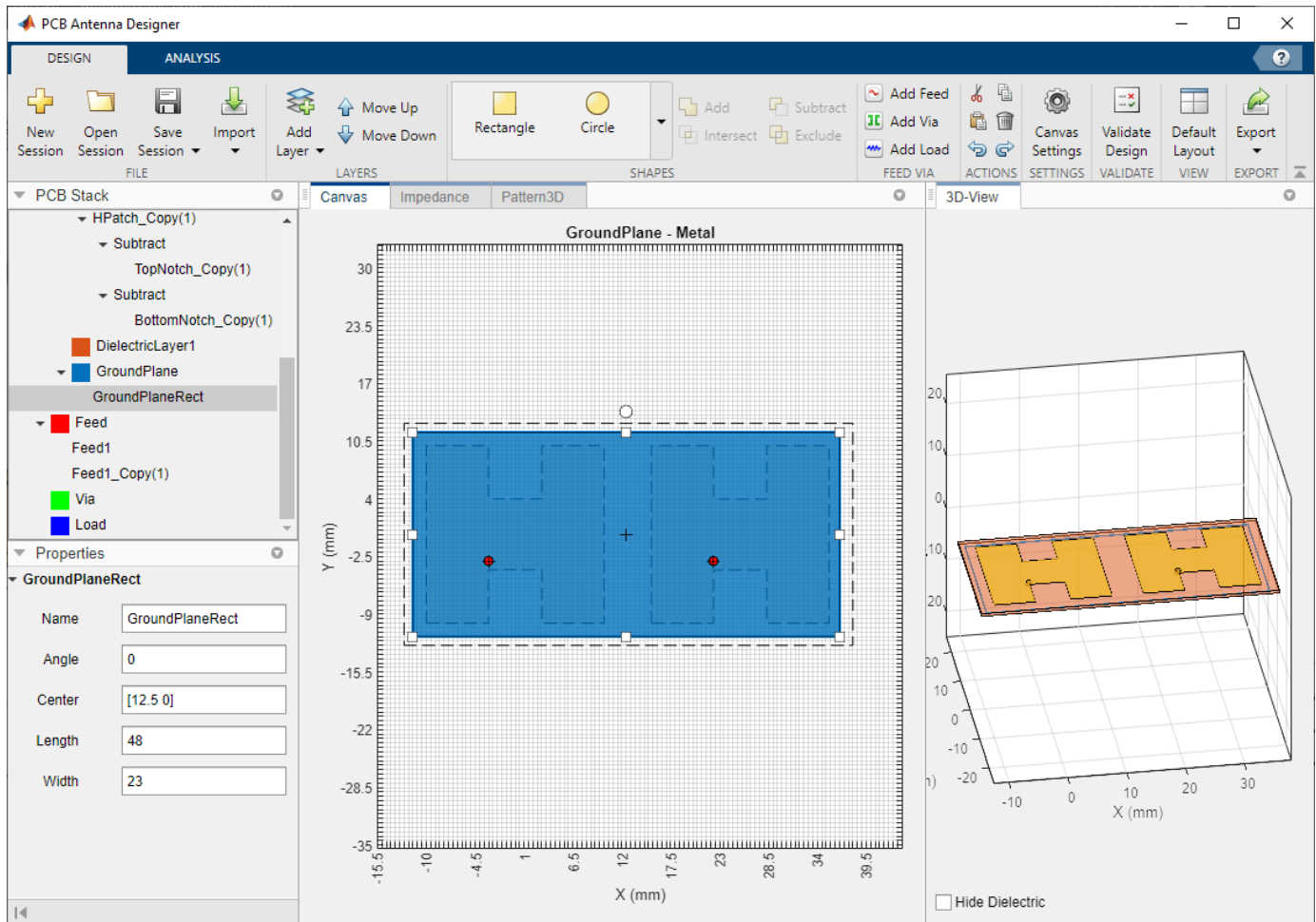


Select Rectangle1 under **BoardShape** on the canvas and use the marker on the right to change the shape of BoardShape such that it encloses the HNotch image.

5 Antenna Toolbox Examples



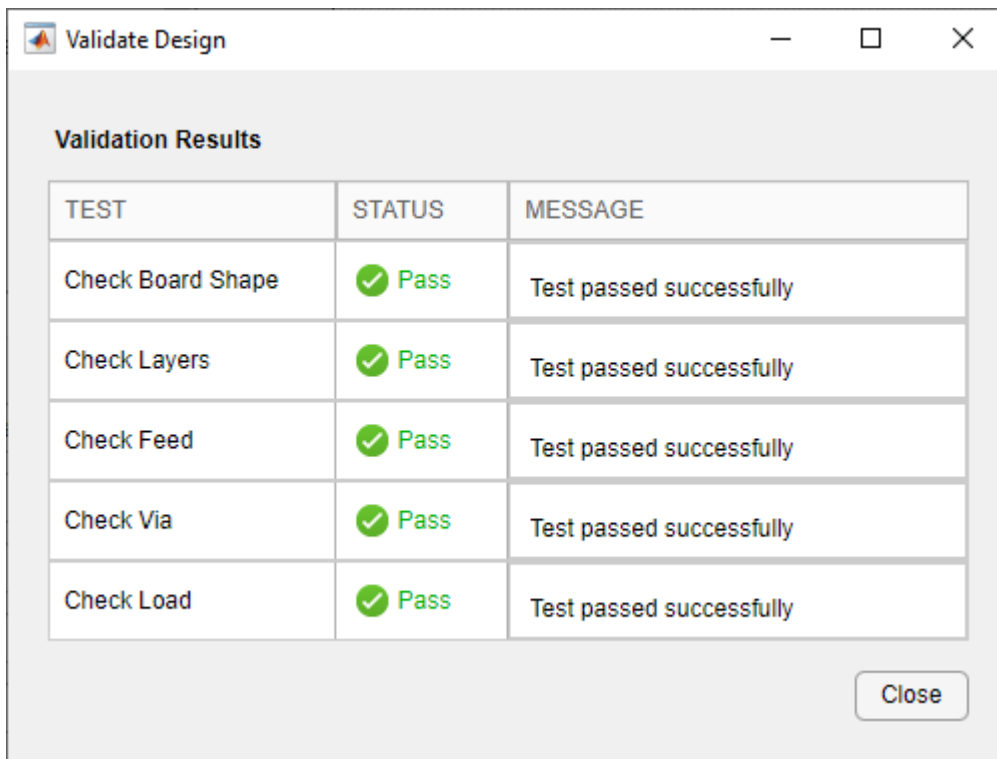
Repeat the procedure with the GroundPlane layer.



The BoardShape now encloses the HNotchLayer. You can change the layer dimensions using the **Properties** pane.

Validate Design for Errors

Click **Validate Design** to check for issues in the design. The design is validated with no errors.

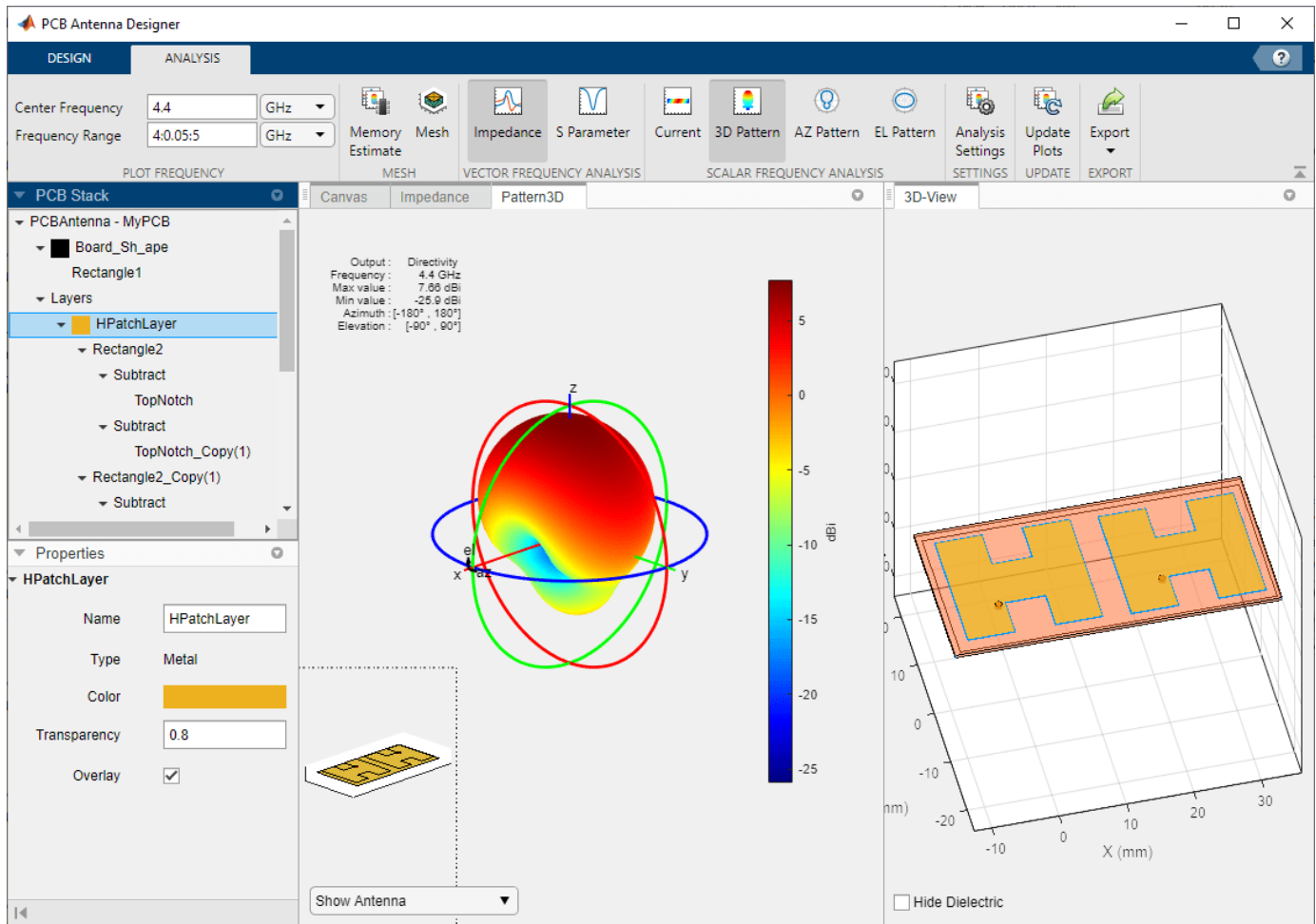


Run Analysis

Click **Update Plots** in the Analysis tab to update the analysis results. This updates all the analysis plots in the app for the design of a 1-by-2 H-notch linear array.

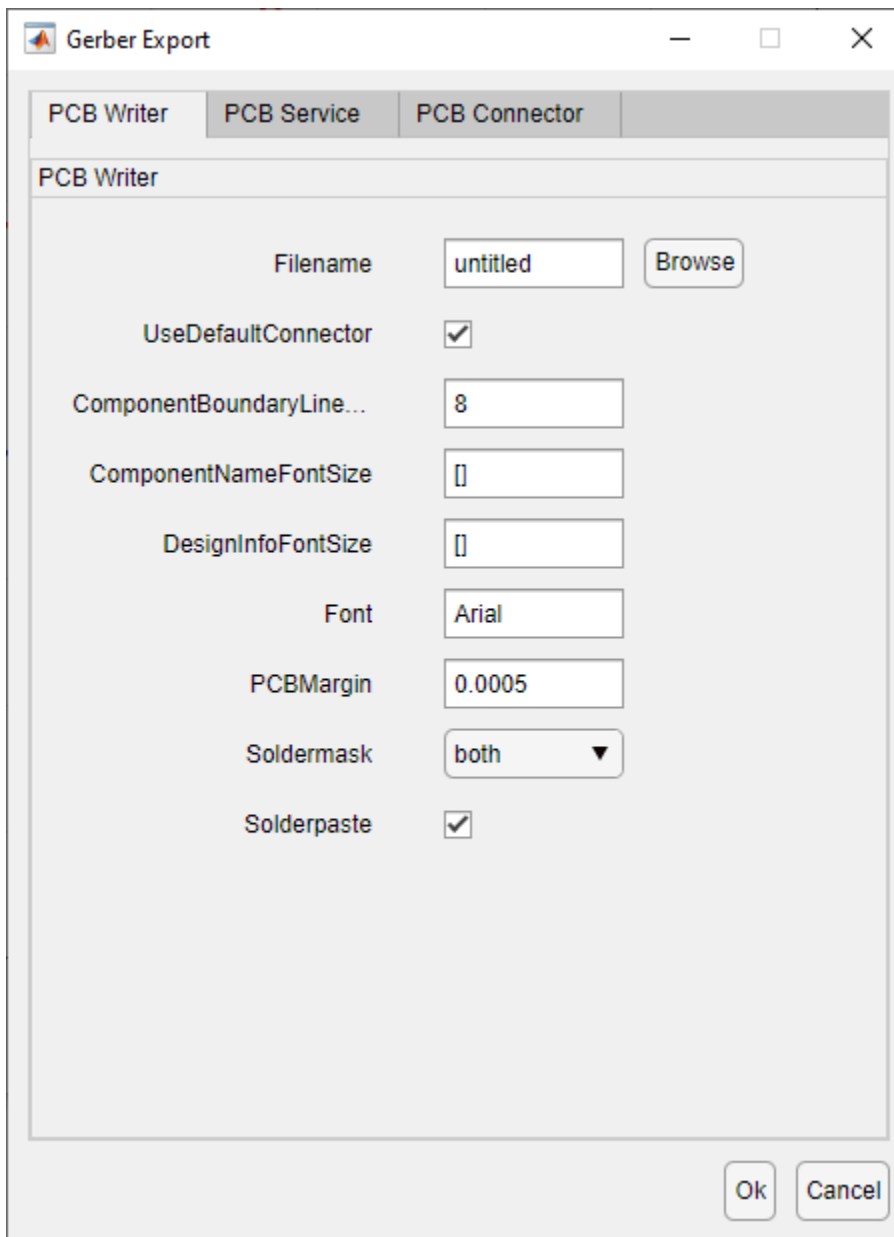
3-D Pattern

The pattern for the antenna is more directional with a higher gain compared to the H-notch patch unit element. The directivity of the 1-by-2 H-notch linear array is 7.66 dBi.

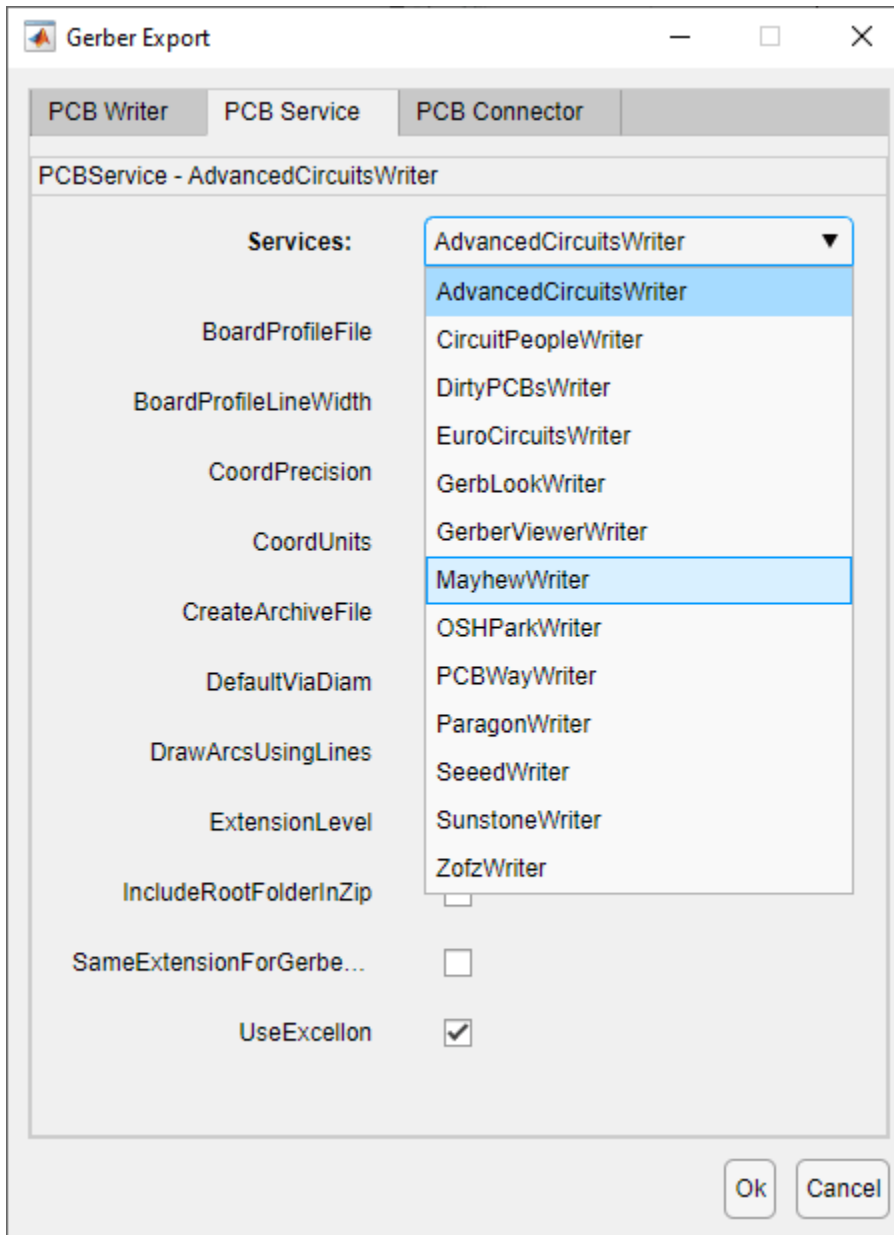


Export Your Design to Gerber Files

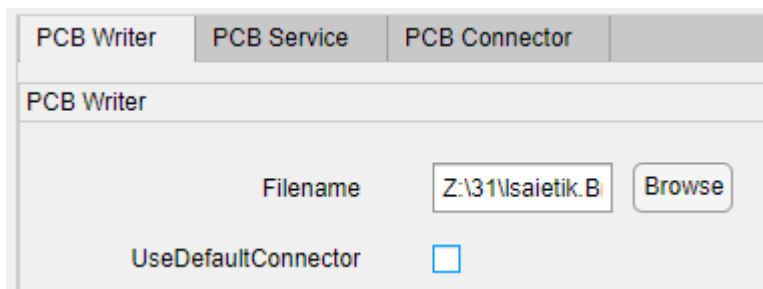
In the **Analysis** tab, click **Export** and then select **Export as Gerber File**. In the Gerber Export dialog box, select **Browse** in the **PCB Writer** tab to select the directory in which you want the Gerber files to be stored. Clear the **UseDefaultConnector** check box.



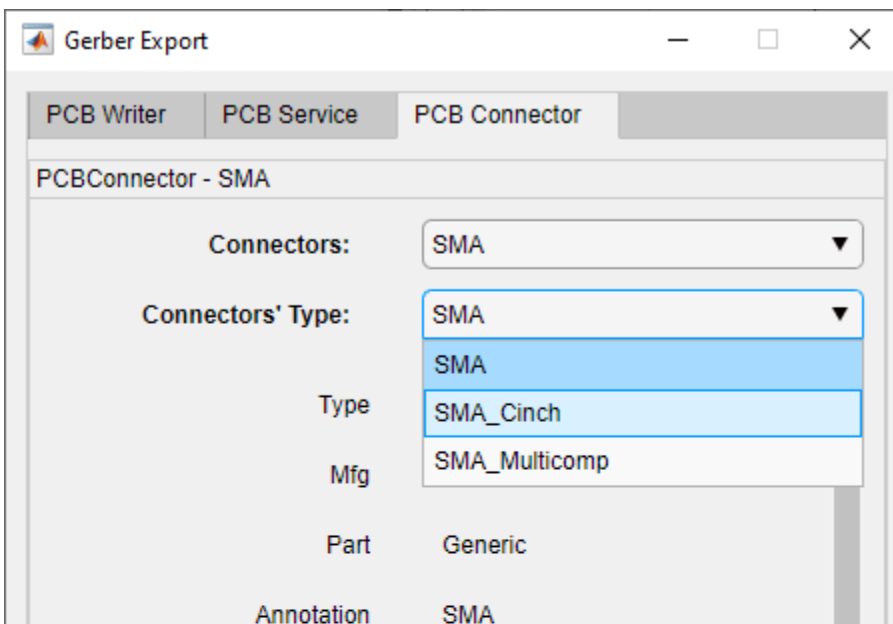
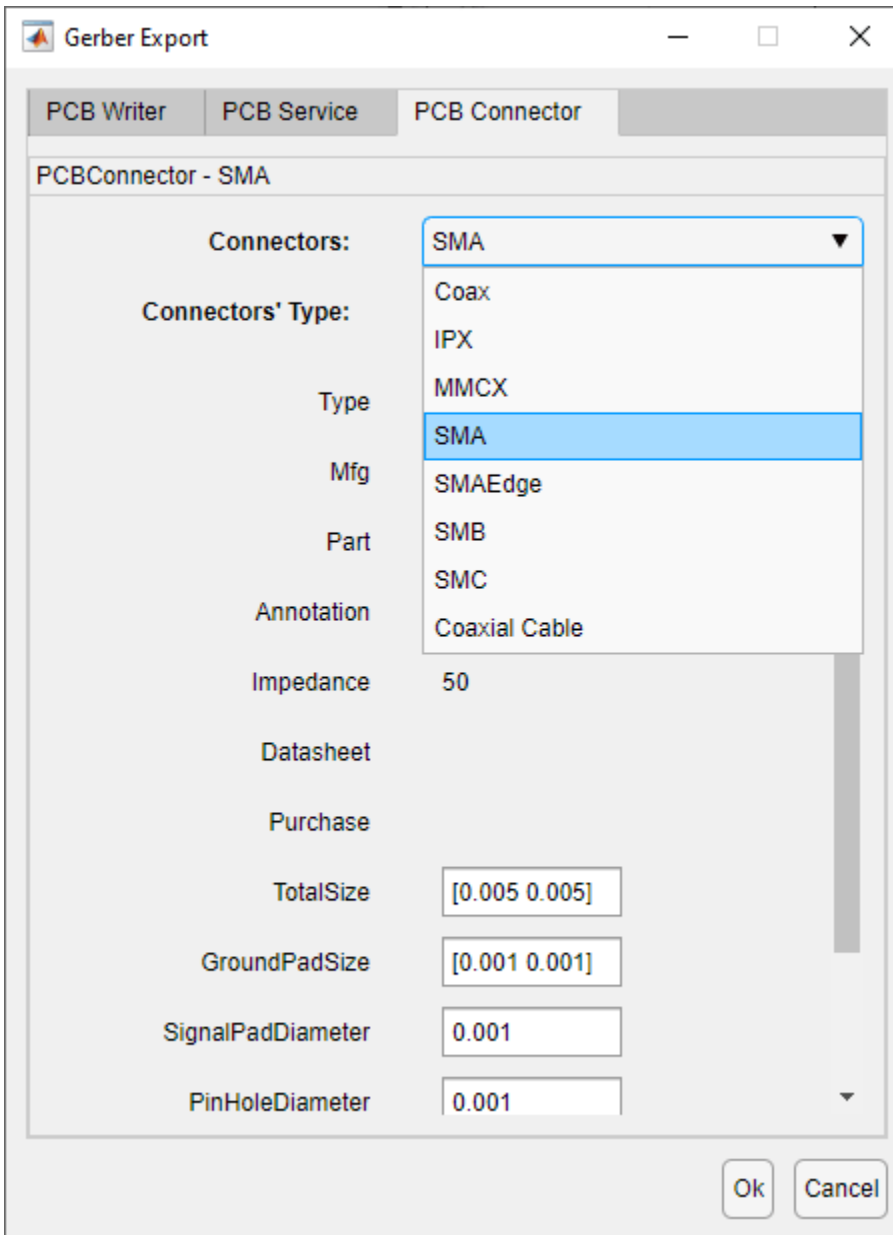
In the **PCB Services** tab, under **Services**, select **MayhewWriter**. The PCB writer uses the selected service to create the Gerber files.



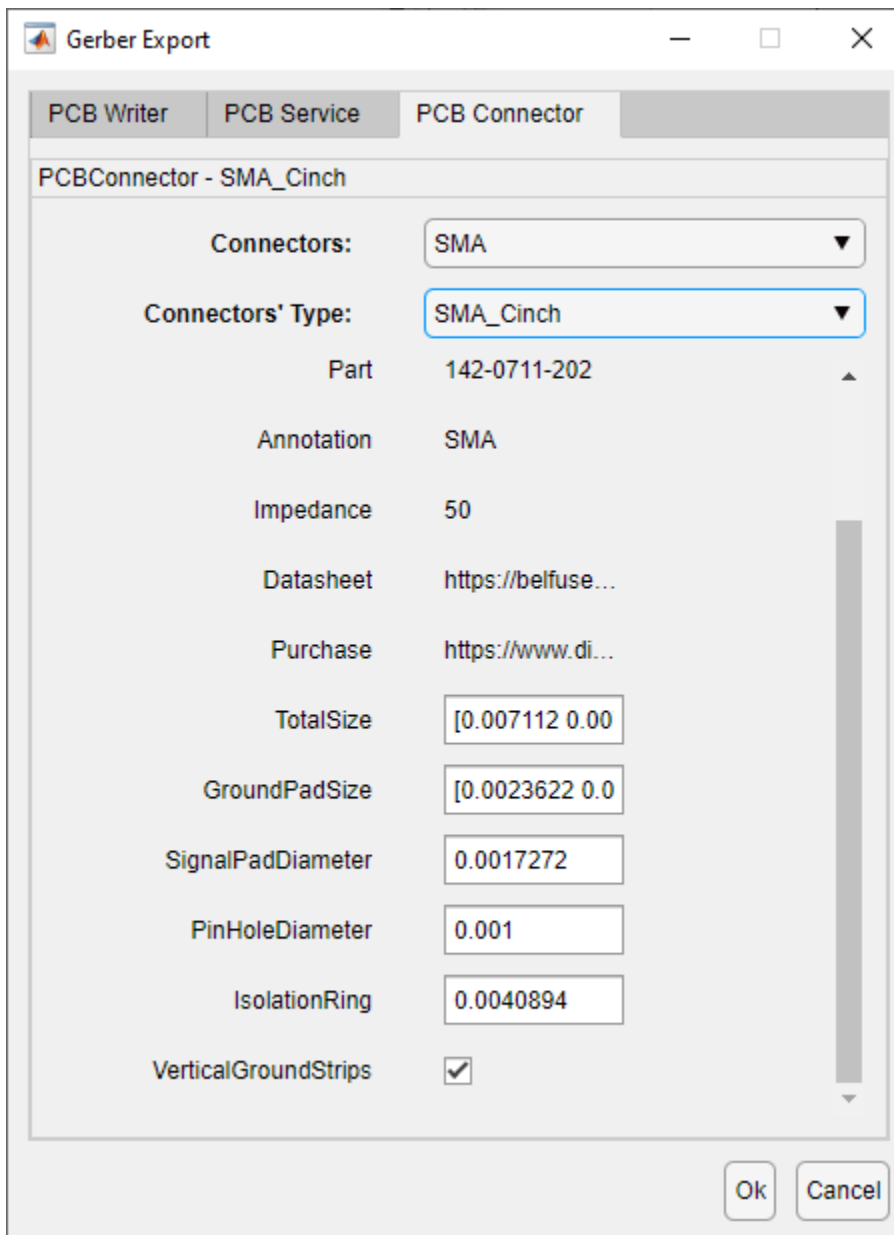
In the **PCB Connector** tab, clear the **UseDefaultConnector** check box.














In the **PCB Connector** tab, set **Connectors** to SMA and **Connectors' Type** to SMA_Cinch.



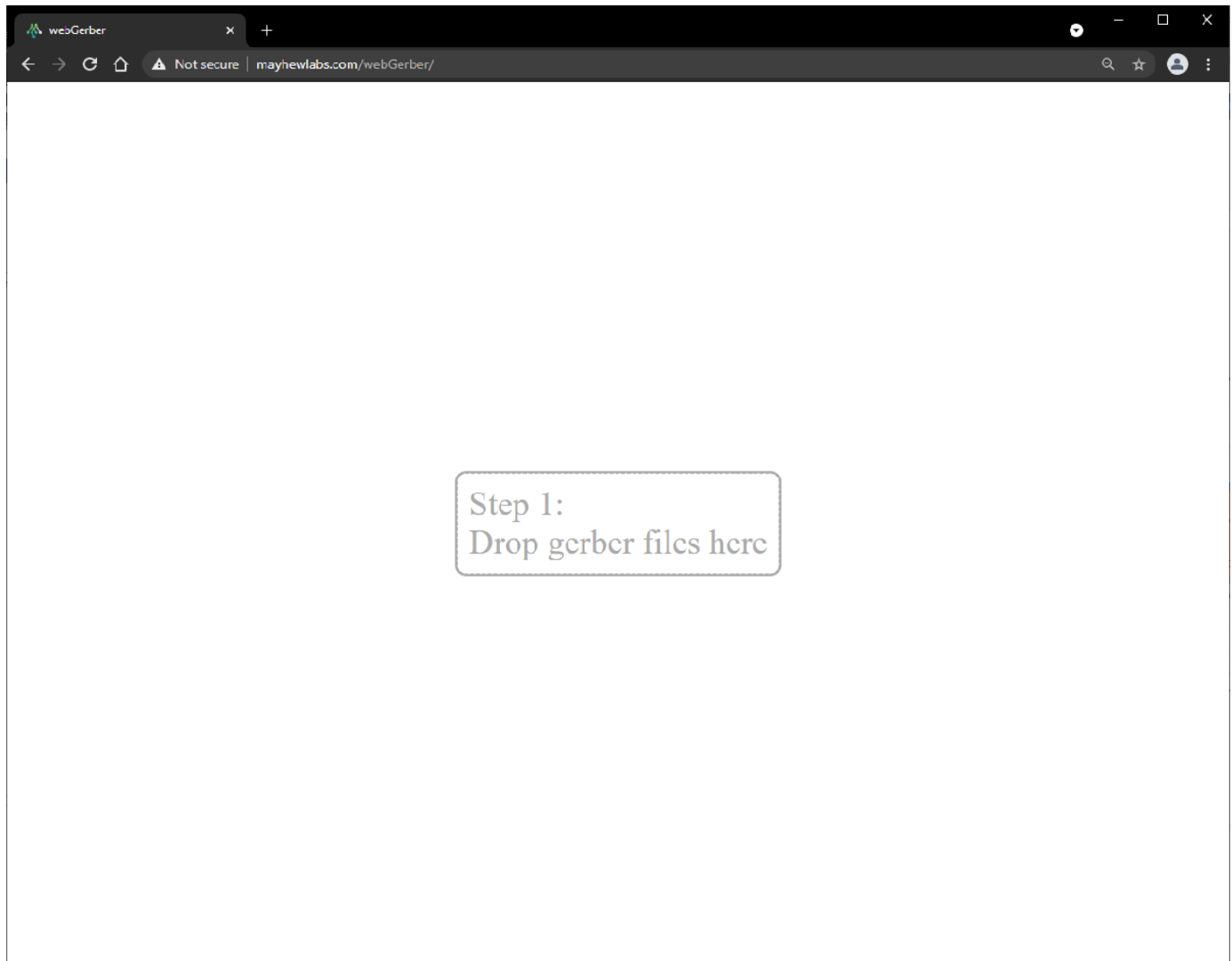
The **PinHoleDiameter** is of the same value as the **FeedDiameter**. Click **OK** to generate the Gerber files.



The generated Gerber files are stored in the directory you selected in the **PCB Writer** tab.

<input type="checkbox"/> Name	Type	Size
 HNotchArray.dri	DRI File	1 KB
 HNotchArray.gbl	GBL File	2 KB
 HNotchArray.gbo	GBO File	184 KB
 HNotchArray.gbp	GBP File	2 KB
 HNotchArray.gbs	GBS File	1 KB
 HNotchArray.gpi	GPI File	2 KB
 HNotchArray.gtl	GTL File	1 KB
 HNotchArray.gto	GTO File	1 KB
 HNotchArray.gts	GTS File	1 KB
 HNotchArray.ipc	IPC File	1 KB
 HNotchArray	Text Document	1 KB

Open mayhewlabs.com/webGerber from your browser. Drag and drop the Gerber files from the directory and select **Done**.

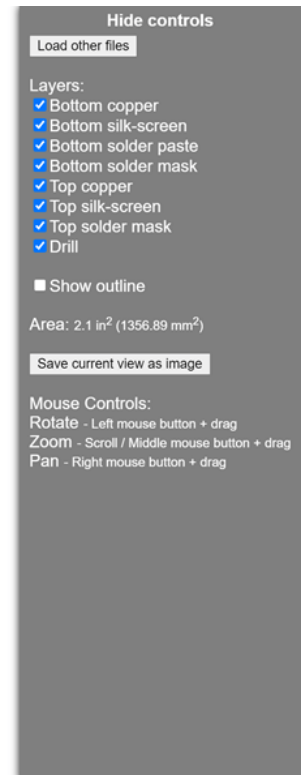


Step 2:
Select the layers corresponding to the gerber files

- HNotchGerberFiles.gbp Bottom solder paste ▾
- HNotchGerberFiles.gbs Bottom solder mask ▾
- HNotchGerberFiles.gpi No layer ▾
- HNotchGerberFiles.gtl Top copper ▾
- HNotchGerberFiles.gto Top silk-screen ▾
- HNotchGerberFiles.gts Top solder mask ▾
- HNotchGerberFiles.ipc No layer ▾
- HNotchGerberFiles.txt Drill ▾
- HNotchGerberFiles.dri No layer ▾
- HNotchGerberFiles.gbl Bottom copper ▾
- HNotchGerberFiles.gbo Bottom silk-screen ▾

Done

The Gerber files are displayed in the Gerber viewer. You can use the Gerber files to fabricate of your 1-by-2 H-notch linear array.



You can also export your design to the MATLAB® workspace or MATLAB script by selecting **Export to MATLAB Workspace** or **Export to MATLAB Script** under **Export** in the **Design** or **Analysis** tab.

See Also

Objects

pcbStack | PCBWriter | PCBServices | PCBConnectors

Functions

gerberWrite

Design, Analyze, and Optimize H-Notch Patch Using Design Variables

This tutorial shows how to define the metal patch, dielectric, and ground plane layers in H-notch using design variables in the PCB Antenna Designer app and optimize the design for maximum gain.

Open New Session

Type this command at the command line to open the **PCB Antenna Designer** app.

```
pcbAntennaDesigner
```

On the **Design** tab, click **New Session** to start a new session and open a blank canvas.

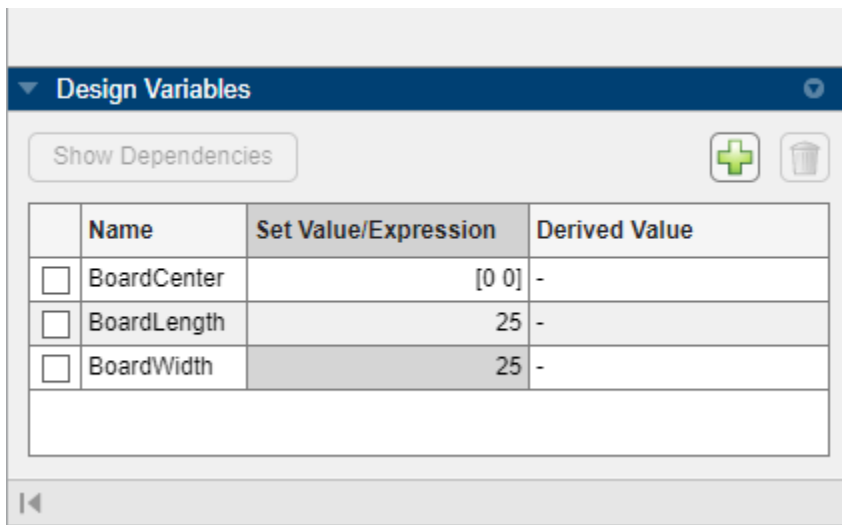
Define Board Shape

To design a PCB stack, you must first define a board shape. Select **Rectangle** from the Shapes section on the toolbar. Drag the shape on the canvas to create a rectangle.

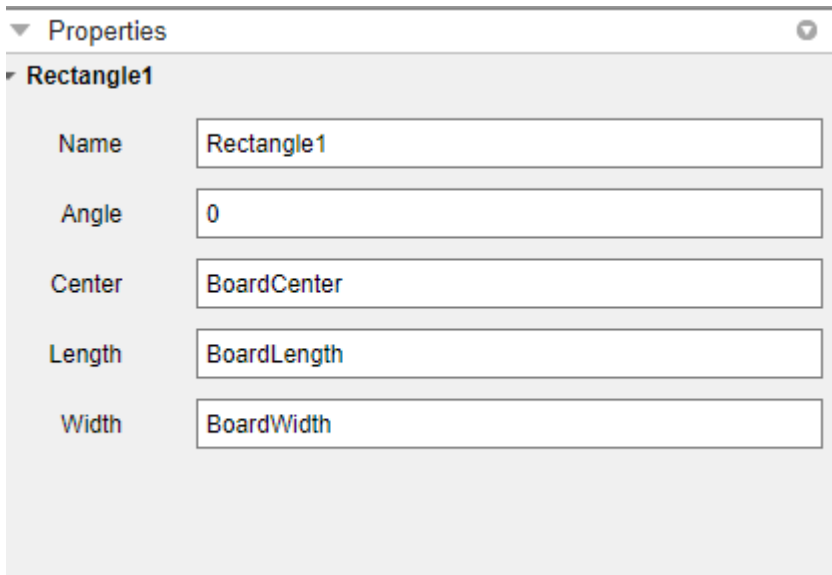
Use the **Design Variables** tab to add variables for the board center, board length, and board width.

To add a new variable, click . Add the following to the table:

- **BoardCenter** — [0, 0]
- **BoardLength** — 25
- **BoardWidth** — 25



Add the variables to the Rectangle1 properties tab.



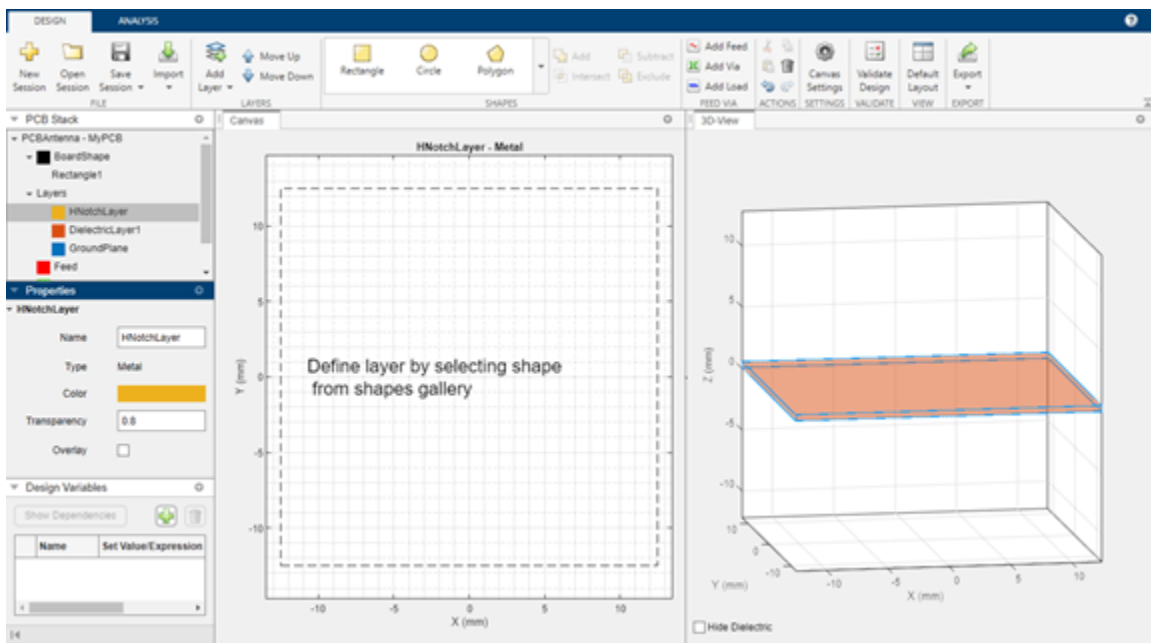
Add Ground, Metal, and Dielectric Layer

Click **Add Layer** on the toolbar and then select **Metal Layer** to add a metal layer.

Set the **Name** of the metal layer to GroundPlane.

Click **Add Layer** on the toolbar and then select **Dielectric Layer** to add a dielectric layer DielectricLayer1.

Click **Add Layer** on the toolbar and then select **Metal Layer** to add a metal layer. Set the name of this layer to HNotchLayer.

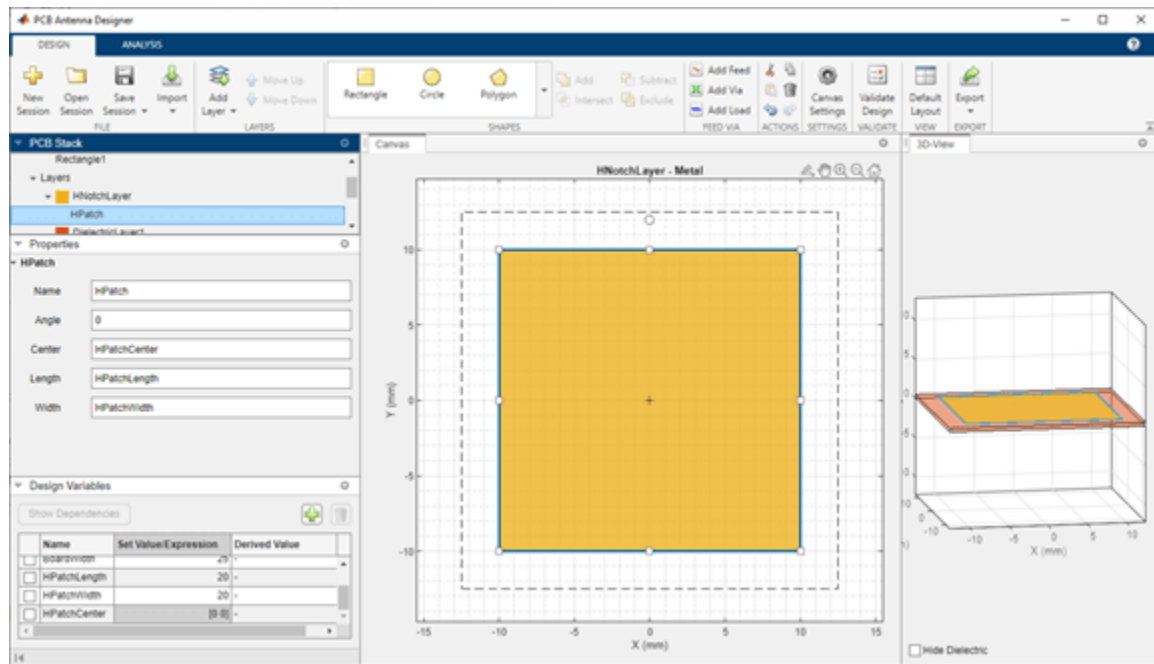


Create H-Notch

Select the HNotchLayer on the PCB Stack navigation tree and then select **Rectangle** from the **Shapes** section on the toolbar. Drag the shape onto the canvas to create a rectangular patch.

Use the **Design Variables** tab to set the properties of the HPatch.

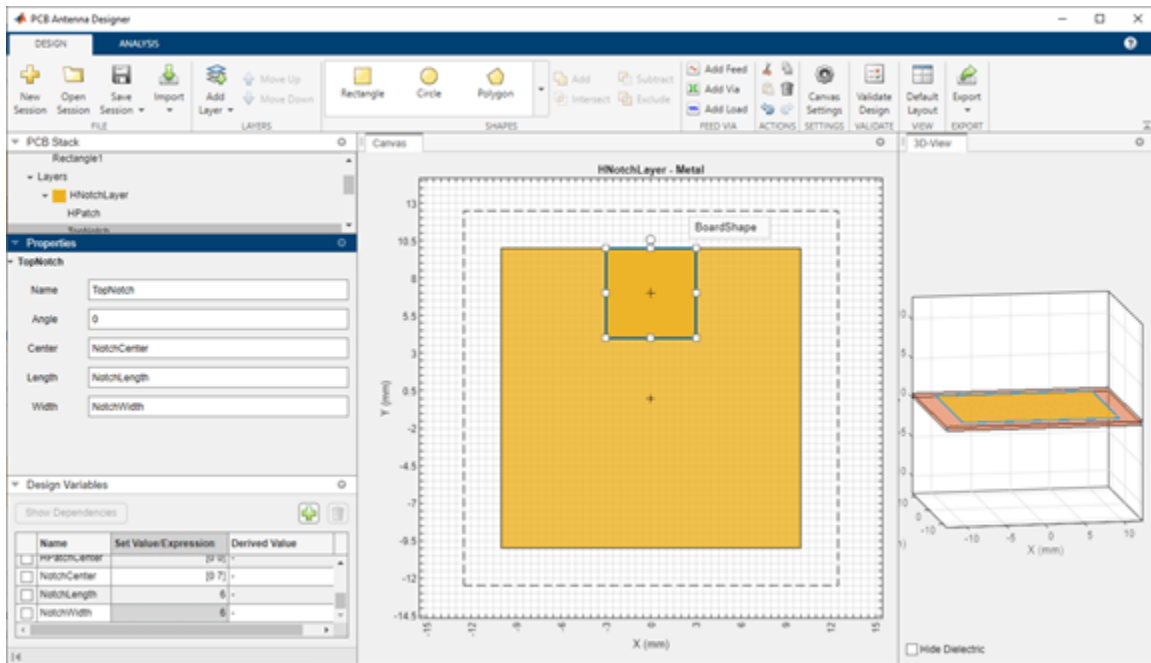
- **Center** — $[0, 0]$
- **Length** — 20
- **Width** — 20



Create Top Notch

Select **Rectangle** from the **Shapes** section and drag the shape to the top of the shape to create a top notch in the patch layer. Set the properties of Rectangle3 to the following using the **Design Variable** tab.

- **Name** — TopNotch
- **Center** — $[0, 7]$
- **Length** — 6
- **Width** — 6

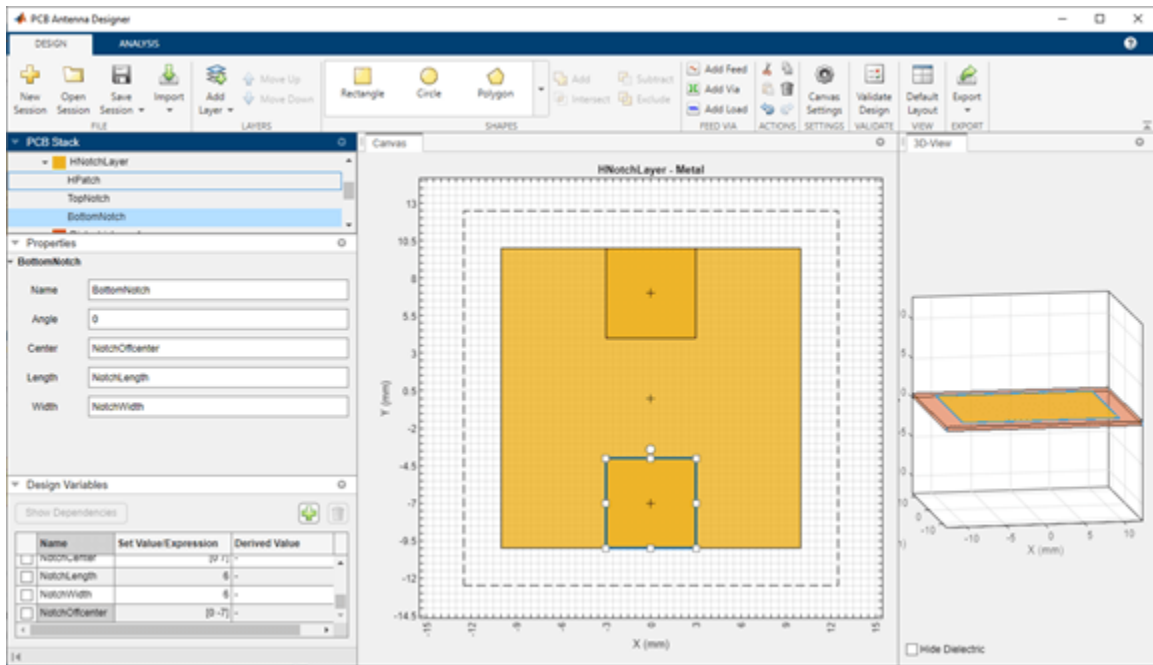


Create Bottom Notch

Select TopNotch from the canvas and then copy the layer by clicking **Copy** in the **Actions** section. Click **Paste** to paste a copy of TopNotch. Doing so creates a rectangle TopNotch_Copy_1 with identical dimensions to the TopNotch layer.

Set the properties of TopNotch_Copy_1 to the following:

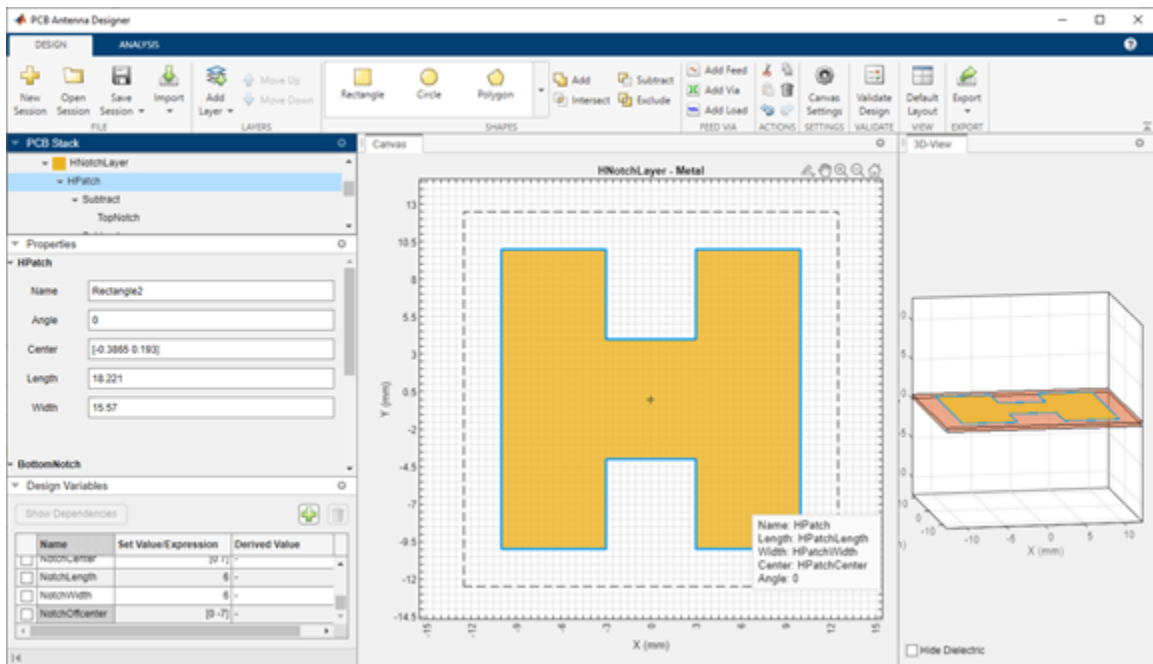
- **Name** — BottomNotch
- **Center** — [0, -7]



Subtract Top and Bottom Notch Layers from Metal Patch

Select HPatch and TopNotch and then click **Subtract** in the **Shapes** section of the toolbar to remove both rectangles.

Select the HPatch and BottomNotch and click **Subtract** in the **Shapes** section of the toolbar to remove both rectangles.



Set Dimensions for Ground Plane

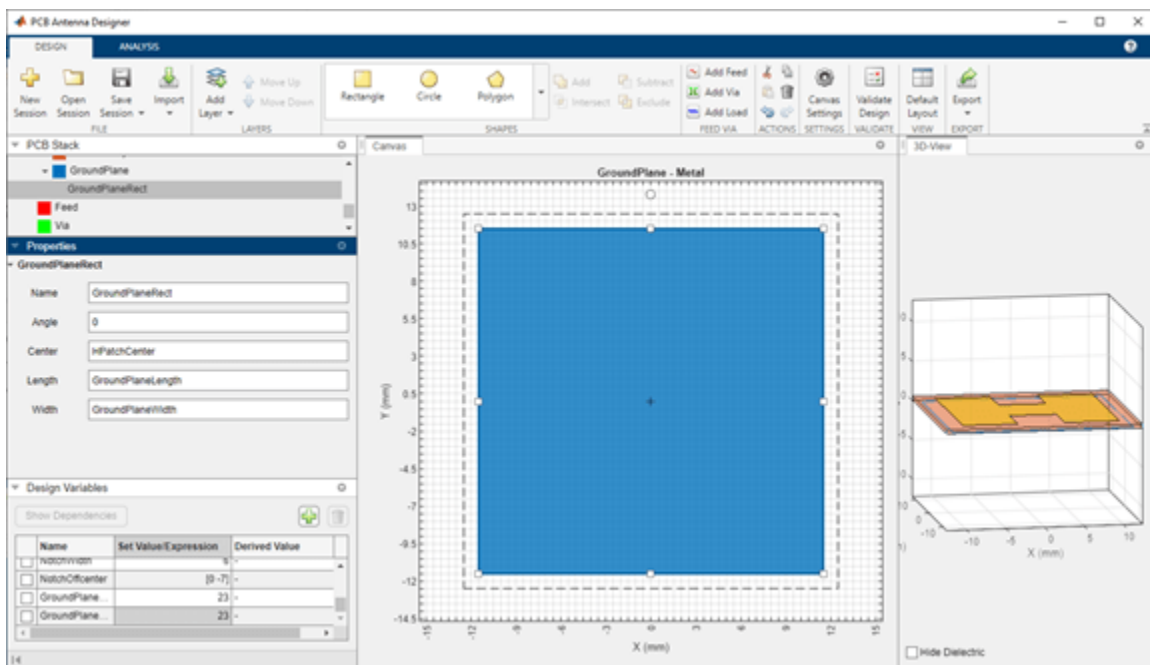
The ground plane must be same size as the HNotchLayer. To make the ground plane the same size as HPatch, right-click HPatch and select **Copy**.

Right-click the GroundPlane and click **Paste** on the toolbar to paste the rectangle in the GroundPlane.

The app copies the rectangle to the ground plane layer with the name HPatch_Copy(1). HPatch_Copy(1) and HPatch have the same dimensions.

Set the properties of HPatch_Copy(1) to the following:

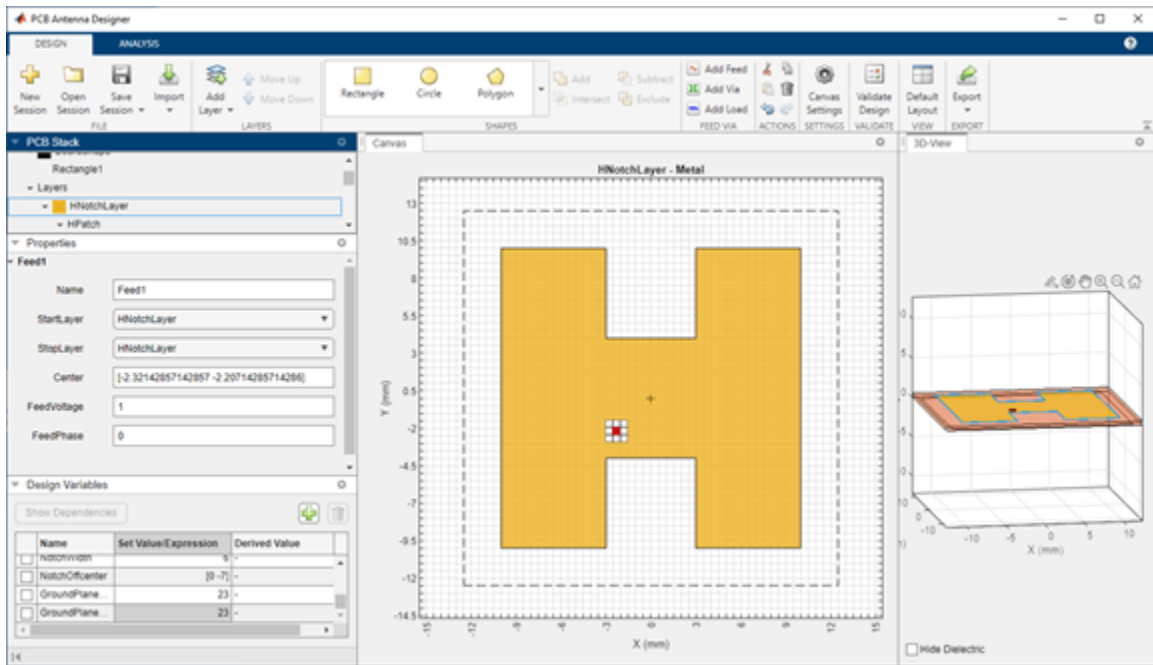
- **Name** — GroundPlaneRect
- **Center** — [0,0]
- **Length** — 23
- **Width** — 23



Add Feed

Select the HNotchLayer metal layer on the **PCB Stack** navigation tree and then click **Add Feed** in the **Feed Via** section on the toolbar. You can add a feed only to a metal layer.

As the PCB Stack navigation tree shows, the app adds the feed to HNotchLayer as Feed1.

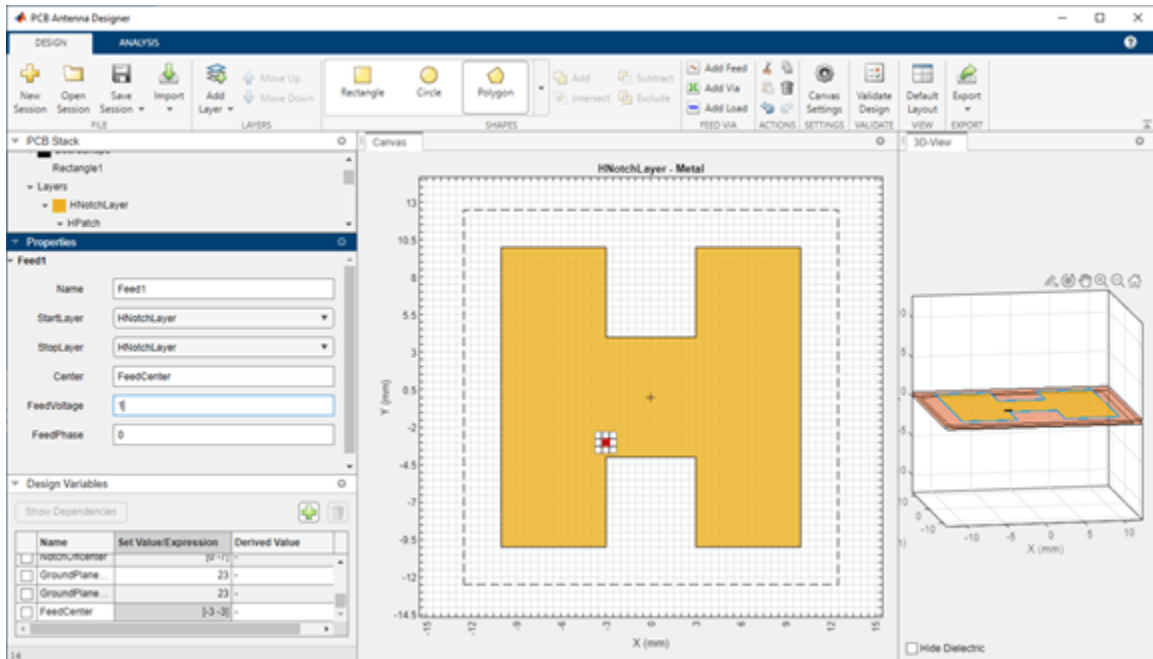


Select the parent Feed node on the PCB Stack navigation tree, and then set the **FeedDiameter** to 1 and **FeedViaModel** to Strip. **FeedDiameter** is a global property.

Change Position of Feed

Select Feed1 on the PCB Stack navigation tree. In the **Properties** pane, set **Center** to [- 3, - 3].

In the **3D-View** pane, you can see that the feed is connected only to HNotchLayer.



For more information on feeds and vias, refer to the **FeedLocation** and **ViaLocation** property descriptions in the **pcbStack** object documentation.

Change Start Layer and Stop Layer

Connect the feed from **HNotchLayer** to **GroundPlane** by setting **StartLayer** to **HNotchLayer** and **StopLayer** to **GroundPlane**. This creates a feed on the **GroundPlane** which connects to the **HNotchLayer**.

Change Metal Properties

Select the **Layers** in on the **PCB Stack** navigation tree and set the **Type** of the metal layer to **Copper** in the **Properties** pane.

Validate Design

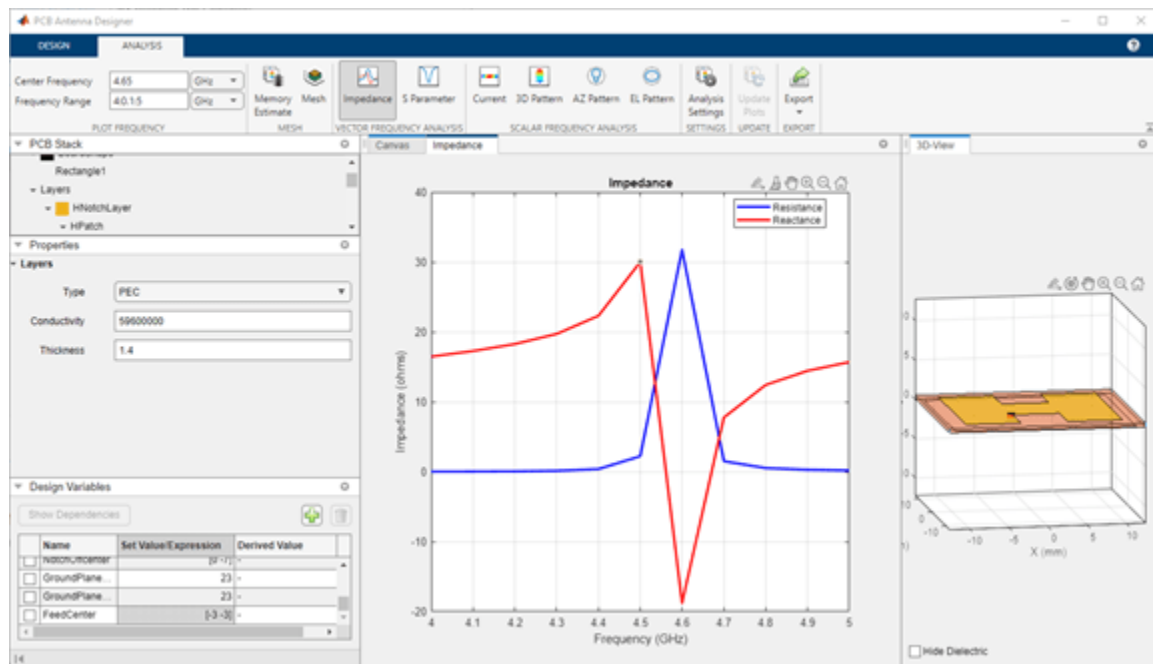
Click **Validate Design** on the toolbar to validate your board shape, layers, feed, via, and load.

Analyze H-Notch Unit Element

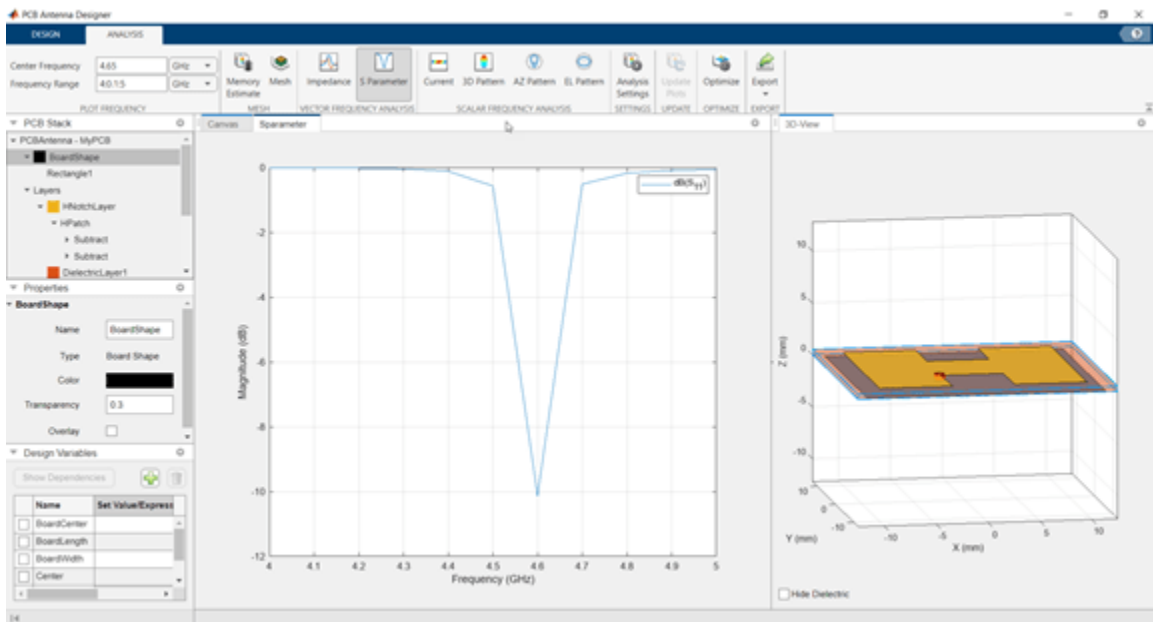
In the **Analysis** tab, set the **Center Frequency** to 4.65 GHz and **Frequency Range** to 4:0.1:5 GHz

Run Analysis

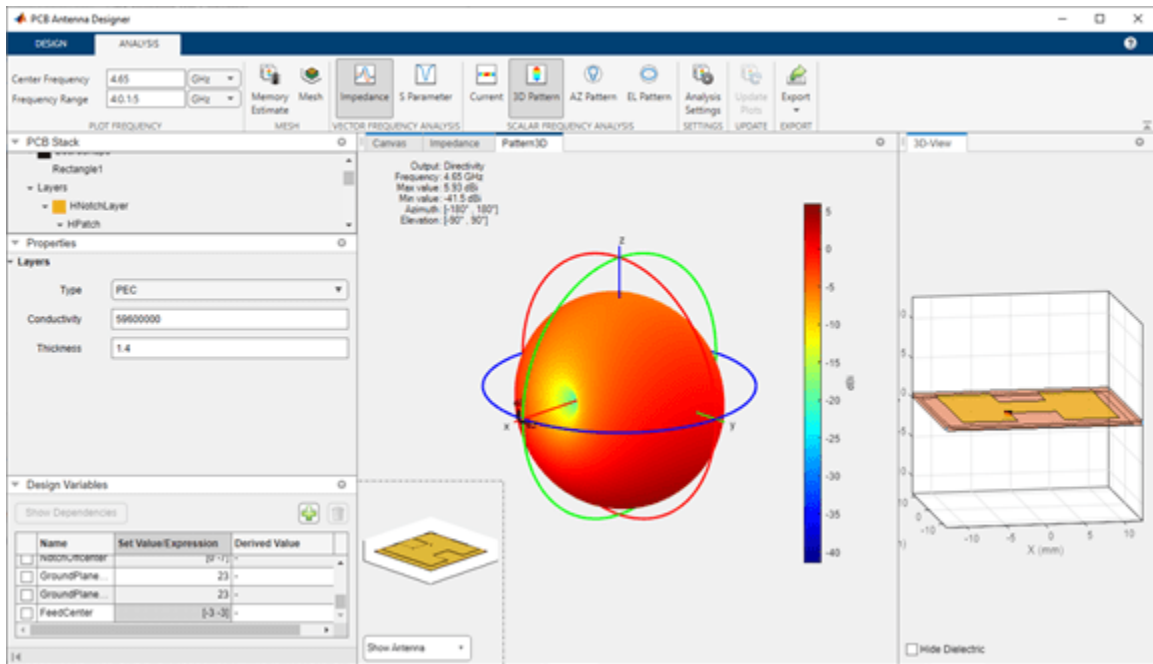
Select **Impedance** from the **Vector Frequency Analysis** section on the toolbar to plot the impedance plot. The impedance plot shows that the PCB stack antenna resonates at 4.6 GHz.



Select **S Parameter** from the **Vector Frequency Analysis** section on the toolbar to plot the **S11**.



Click **3D Pattern** from the **Scalar Frequency Analysis** section on the toolbar to plot the 3-D far-field radiation pattern of the antenna. The directivity of the H-notch patch unit element is 5.93 dBi.



Setup Optimization Problem

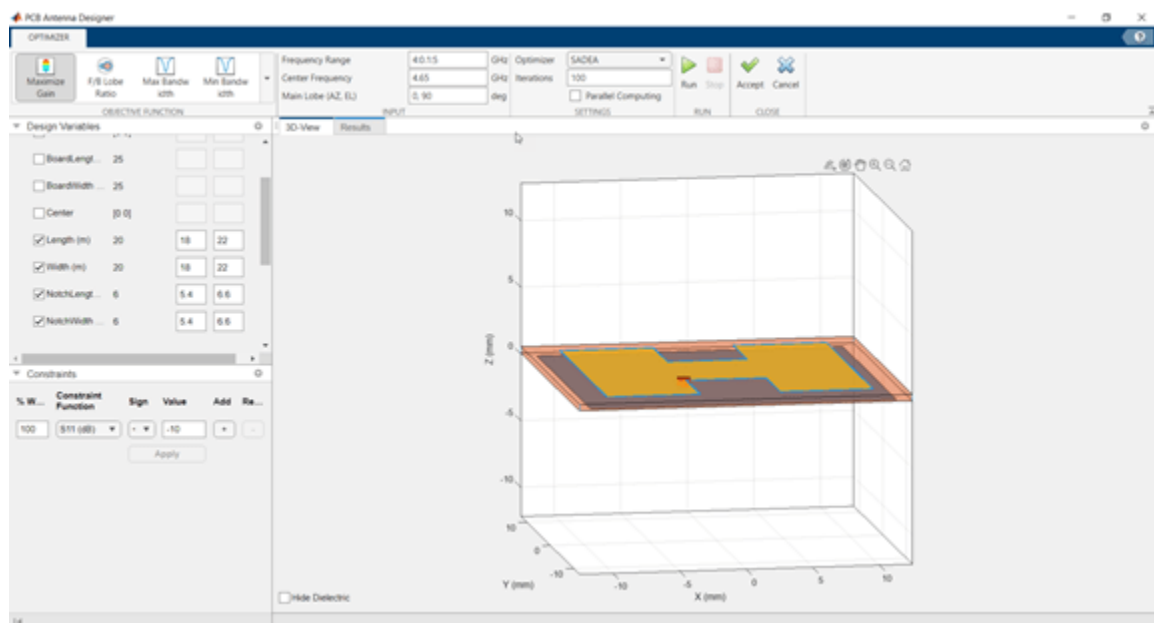
Any optimization problem typically requires following inputs.

- 1 Objective function: Main goal of the optimization. It evaluates the analysis function and minimizes or maximizes the output of the function. **In this tutorial, maximizing the gain of the antenna is the objective function.**

- 2 Design variables: The input variables to the objective function. These variables are changed by the optimizer within a pre-set range of values called as the bounds of the variables. **In this tutorial, the dimensions of the H-Notch patch are the design variables.**
- 3 Constraint functions (if necessary): Functions which restrict a desired analysis function value on the antenna. **In this tutorial, the constraint function is S11 less than -10 db to obtain an impedance bandwidth.**
- 4 Other inputs: Other inputs may include the number of iterations, the input center frequency, and the input frequency, the number of iterations, the frequency at which the analysis is performed, etc.

Optimization Function, Design Variables, and Constraints

To optimize the H-Notch patch, click on the **Optimize** button. To select an objective function, use the **OBJECTIVE FUNCTION** Gallery drop down. Since the goal is to maximize the gain of the antenna, click on **Maximize Gain**. To set up the design variables, click on the **Design Variables** tab. Click on the checkboxes present on the left-hand side of the properties to choose the required design variables. The optimizer would change these chosen properties to obtain a maximum gain for the antenna. To set up the constraints, click on the Constraints tab and select S11 (dB) from the Constraint Function. Select < operator from sign and enter value as -10.



Click **Apply**.

In the **SETTINGS** section, enter the number of **Iterations** to run for the optimizer. To start the optimization, click the **Run** button.

Optimization

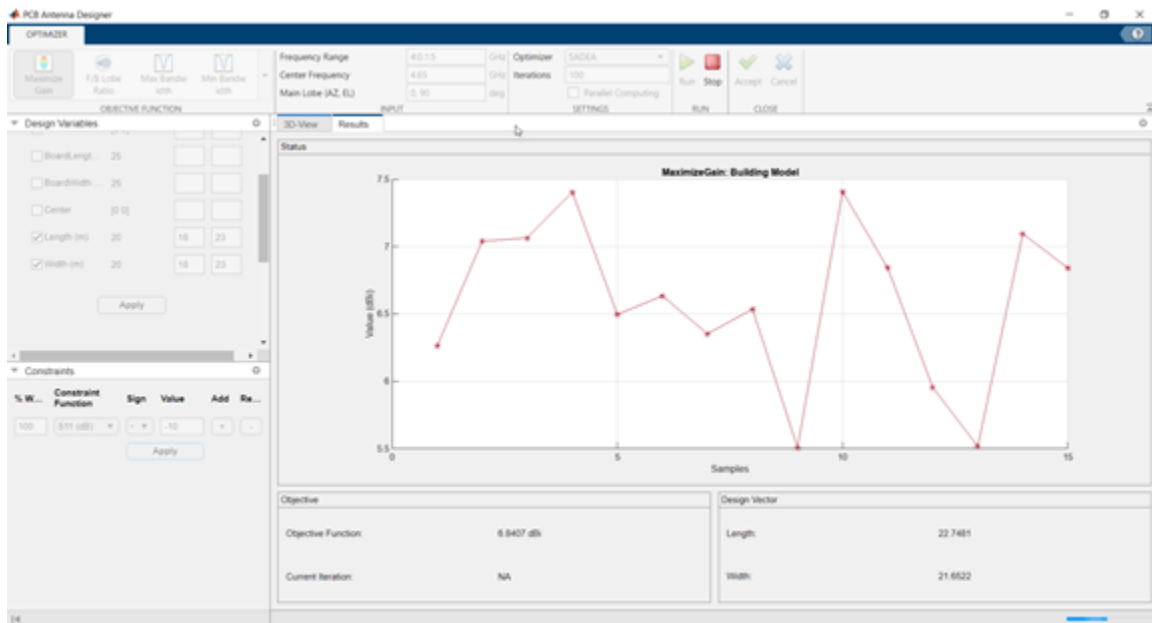
The **SADEA** optimization contains two stages:

- 1 Building model
- 2 Optimizing

Building Model

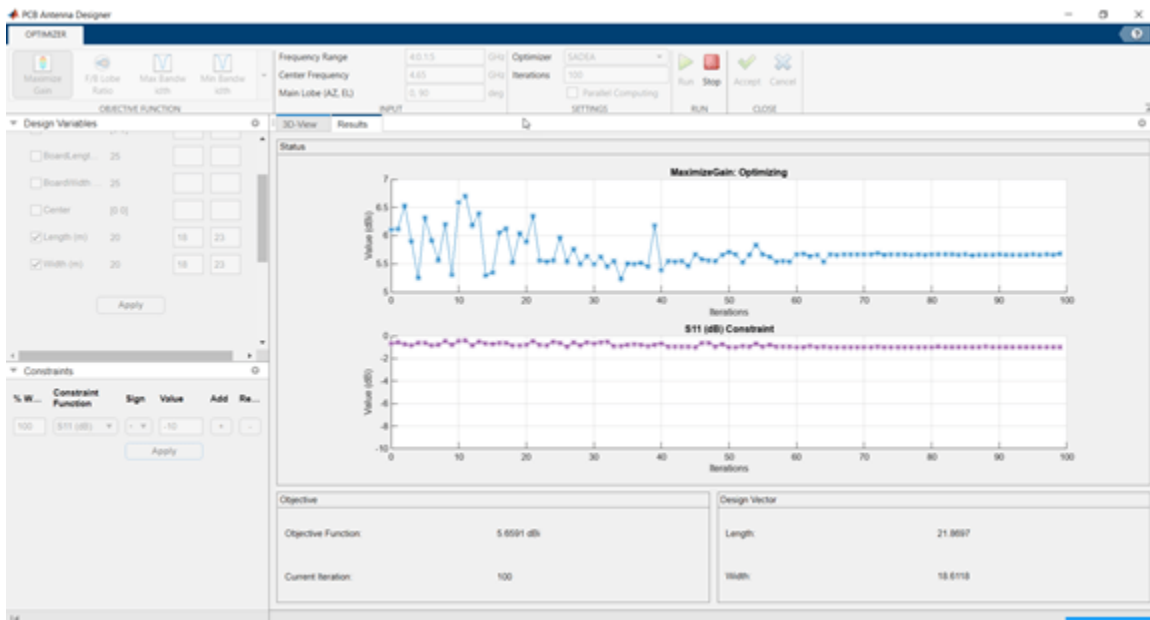
In the model building stage, the optimizer makes a surrogate model from the design space, and the specified objective and the constraints function. It diversely goes through the design space and performs analysis on these sample points.

So, the X-axis shows the number of samples and the Y-axis shows the value of the analysis function value at that sample. The bottom left side show the current sample value and the bottom right side shows the design variables. The optimizer within decides and takes appropriate number of samples to build the model. After the model is built, the optimizer starts running iterations.



Optimizing

In the optimizing stage, the X-axis shows the number of iterations and the Y-axis shows the objective function values. From the plots shown on the optimizing stage, you can understand the trend of convergence.



Once the optimization is complete, click on **Accept**. This takes you back to the **Analysis** tab. And the dimensions are updated with the optimized values. Run the **Impedance**, **S Parameters**, and **3D Pattern** analysis again to compare optimization results with the original results.

See Also

Objects

pcbStack | antenna.Rectangle

Functions

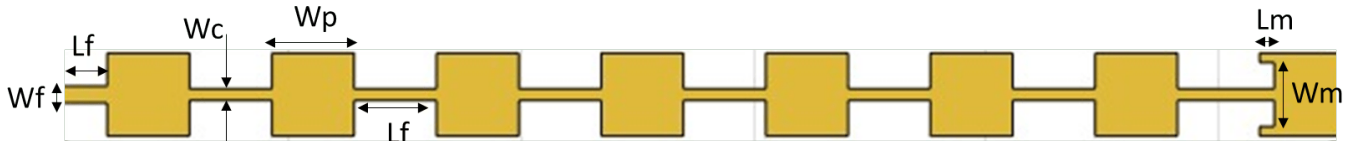
subtract | impedance | sparameters | pattern

Related Examples

- “Dielectric Catalog”
- “Metal Catalog”
- “Antenna Optimization Algorithm” on page 3-29
- “Maximizing Gain and Improving Impedance Bandwidth of E-Patch Antenna” on page 5-625

Design Series-Fed Patch Antenna Array for 5G Base Station

This example shows how to design and analyze a series fed patch antenna array at 28 GHz. The array has 64 antenna elements arranged in 8-by-8 configuration and is used as a 5G mobile base station antenna at 28 GHz. The phased array steers its beam along the horizontal axis to provide coverage in different directions.



Create Variables

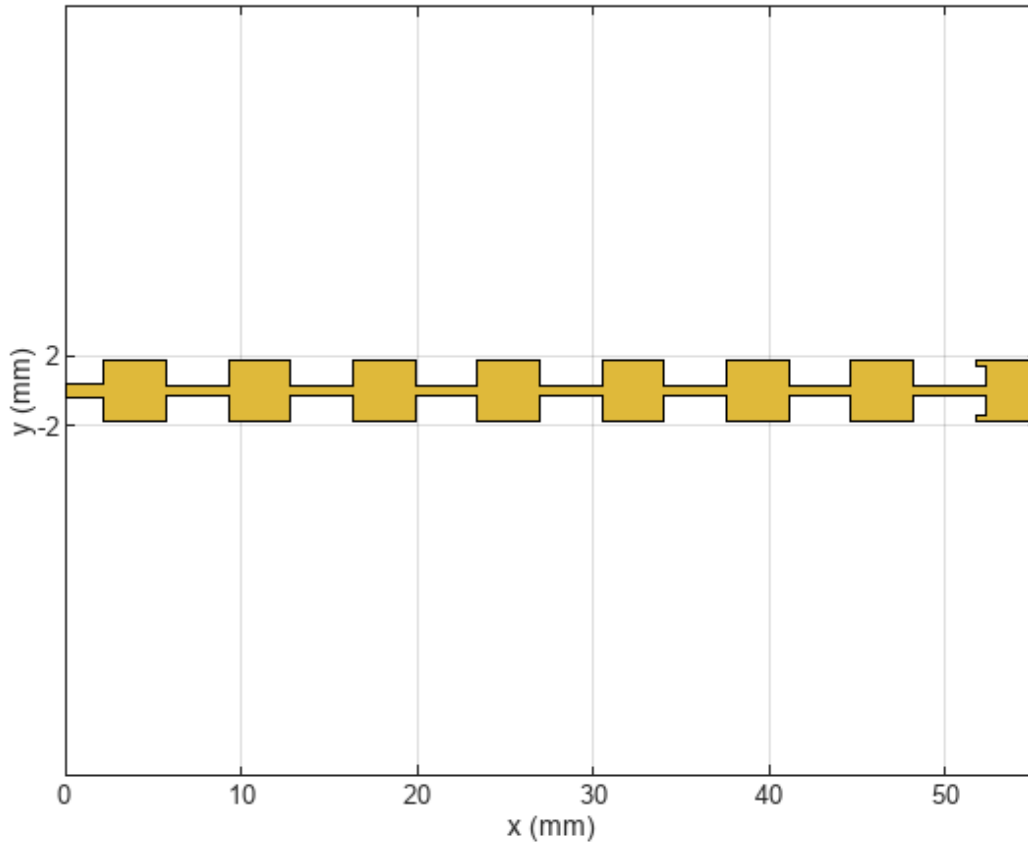
Create the variables and assign the values as per [2].

```
Wp = 3.539e-3;
Wc = 0.494e-3;
Lc = 3.539e-3;
Wm = 2.727e-3;
Lm = 0.6269e-3;
Wf = 0.72e-3;
Lf = 2.215e-3;
```

Create Geometry

Use the `antenna.Rectangle` shape object to create all the rectangles in the structure and then join them. Visualize the structure using the `show` function.

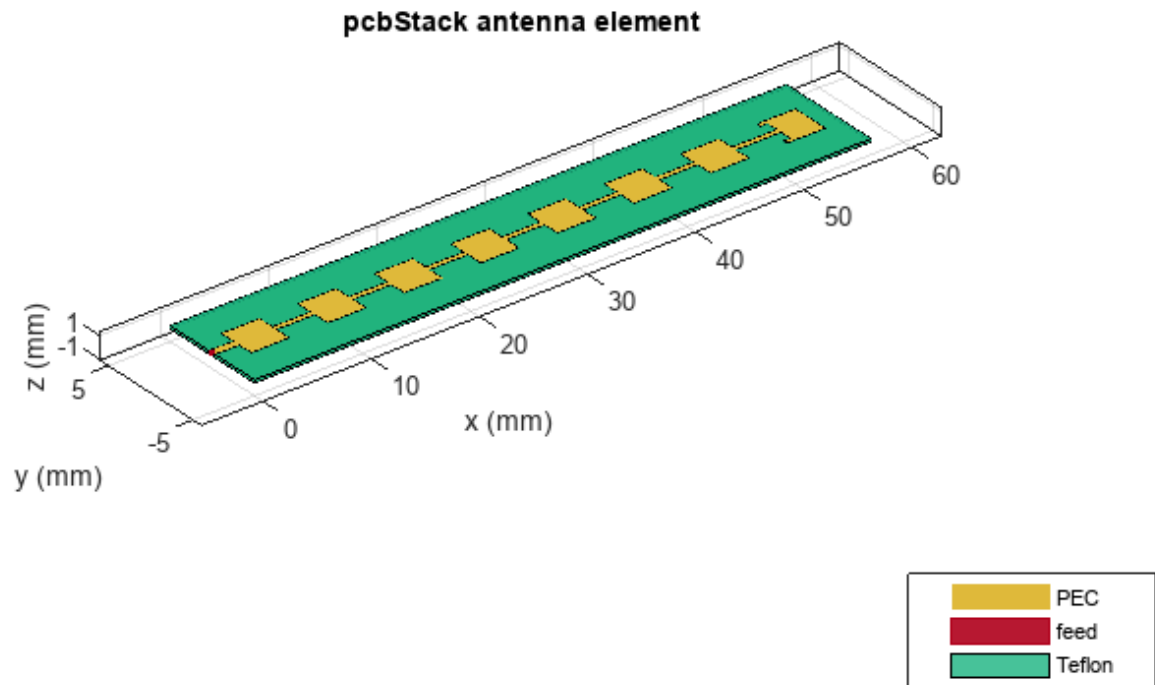
```
a = antenna.Rectangle(Length=Lf,Width=Wf,Center=[Lf/2,0]);
transVec = Lf;
for i =1:7
b = antenna.Rectangle(Length=Wp,Width=Wp,Center=[transVec+Wp/2,0]);
transVec = transVec+Wp;
c = antenna.Rectangle(Length=Lc,Width=Wc,Center=[transVec+Lc/2,0]);
transVec = transVec+Lc;
a = a+b+c;
end
last = antenna.Rectangle(Length=Wp,Width=Wp,Center=[transVec+Wp/2,0]);
lastsub = antenna.Rectangle(Length=Lm,Width=Wm,Center=[transVec+Lm/2,0]);
last = last-lastsub;
conn = antenna.Rectangle(Length=Lc,Width=Wc,Center=[transVec+Lc/2,0]);
a = a+last+conn;
figure;
show(a)
```



Design PCB Board

Use the `pcbStack` object to create the PCB stack. Create the ground plane and dielectric layers, and set the `BoardShape` to be the same shape as the ground plane. Visualize the PCB stack.

```
ant = pcbStack;
gnd = antenna.Rectangle(Length=57e-3,Width=10e-3,Center=[57e-3/2,0]);
d = dielectric('Teflon');
d.EpsilonR = 2.2;
ant.BoardThickness = 0.254e-3;
d.Thickness = ant.BoardThickness;
ant.Layers={a,d,gnd};
ant.BoardShape = gnd;
ant.FeedDiameter = Wf/2;
ant.FeedLocations = [0,0,1,3];
figure;
show(ant)
```



Analyze Array Performance

Use the `mesh` function to manually generate the mesh and set the `MaxEdgeLength` property to 0.4 mm to ensure there are 30 triangles per wavelength.

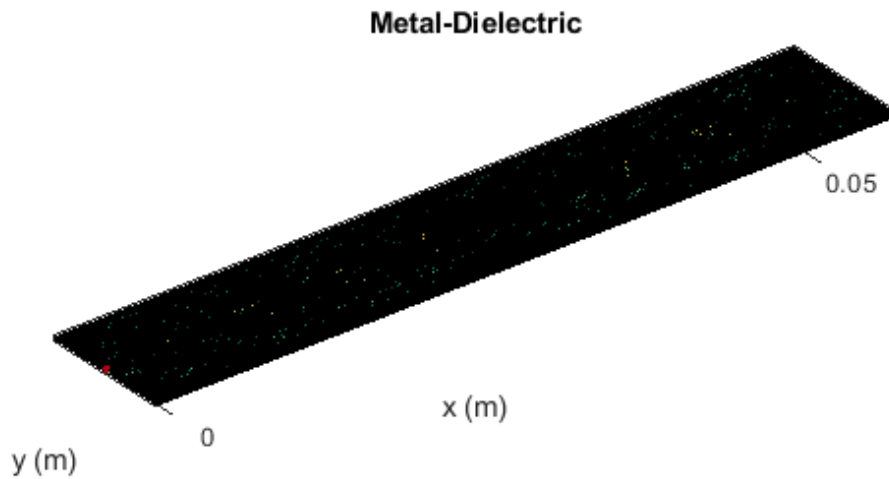
```
figure;  
mesh(ant,MaxEdgeLength=0.4e-3);
```



```

NumTriangles: 12966
NumTetrahedra: 31881
NumBasis:
MaxEdgeLength: 0.0004
MeshMode: manual

```



Use the `memoryEstimate` function to calculate the memory required to solve the structure.

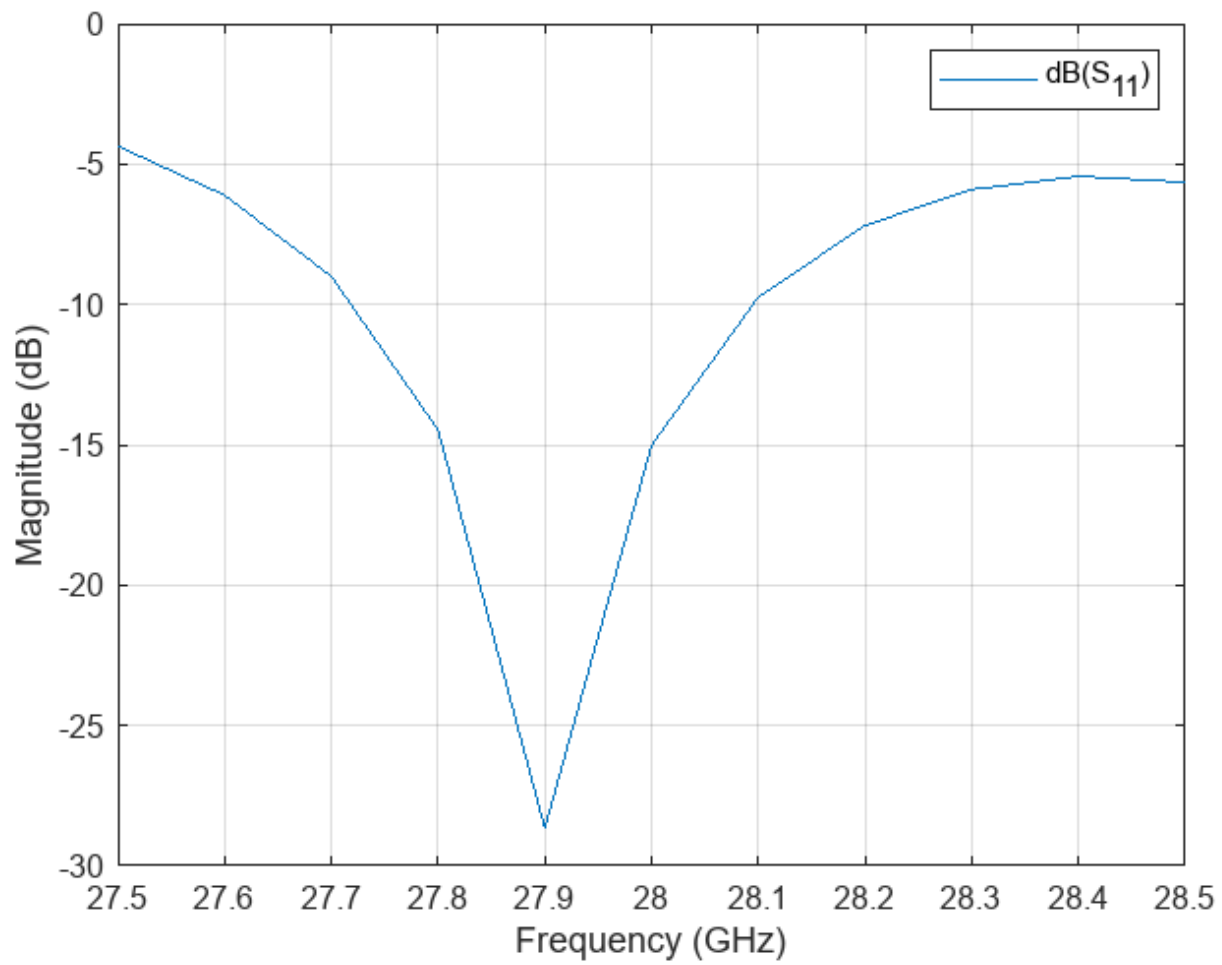
```
memEst = memoryEstimate(ant,28e9)
```

```
memEst =
'38 GB'
```

As the memory requirement is very high, time required to calculate the S-parameters and plot the antenna pattern is quite high. This example includes a MAT file containing the return loss and pattern data computations. The code used to compute the results is available in the **Return Loss and Pattern Data Computations** section. On the machine with 64 GB of RAM and Intel(R) Xeon(R) W-2133 CPU processor it takes around 50 minutes to solve one frequency.

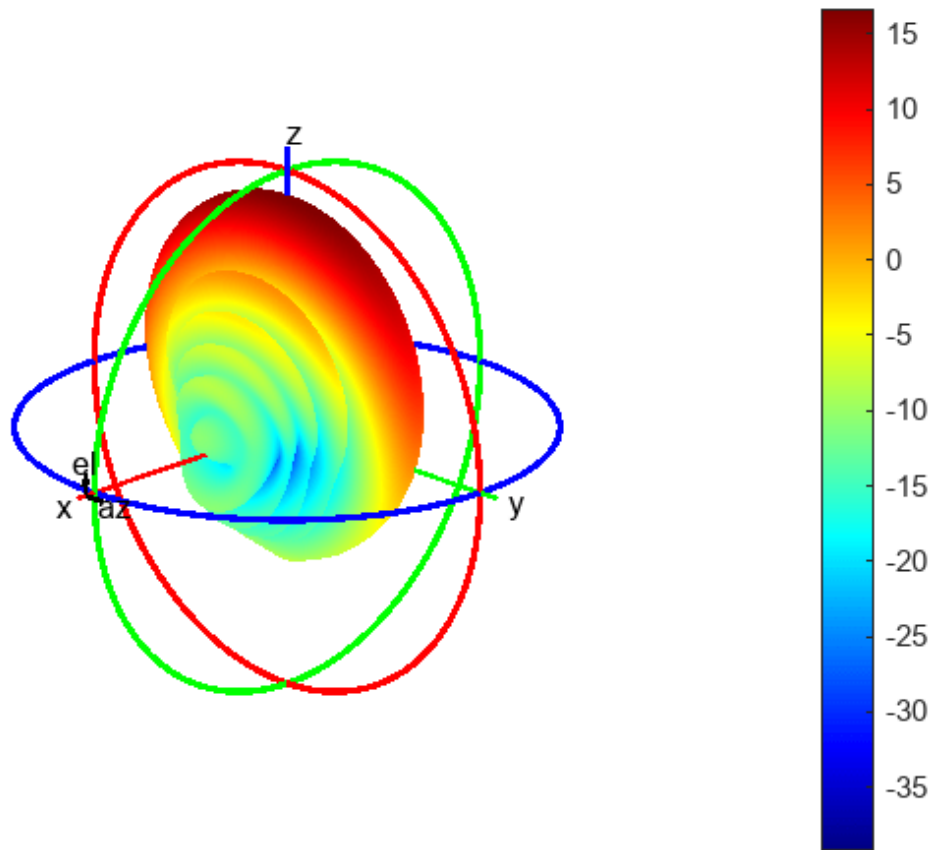
Load the MAT file and plot the return loss using the `rfplot` function.

```
load seriesPatchData.mat
figure;
rfplot(out);
```



Use the `patternCustom` to plot the 2-D or 3-D radiation pattern of the antenna.

```
phi = az';  
theta = (90-el);  
MagE = pat';  
figure;  
patternCustom(MagE,theta,phi);
```



Plot Radiation Pattern Using Pattern Multiplication

Use pattern multiplication to plot the radiation pattern of the 8-by-8 array instead of solving the entire structure using the MoM solver. You need a license to the Phased Array System Toolbox™ to use pattern multiplication.

Use the `phased.CustomAntennaElement` System object™ to build the custom antenna element using the pattern data from the loaded MAT file.

```
antenna = phased.CustomAntennaElement;
antenna.FrequencyVector = [0 28e9];
antenna.AzimuthAngles = az;
antenna.ElevationAngles = el;
antenna.MagnitudePattern = pat;
antenna.PhasePattern = zeros(size(pat));
```

Use the `phased.ULA` System object™ to create the 8-by-8 array. Specify the custom element as the array element. Set the number of elements to 8, and set the element spacing to 3.5 mm.

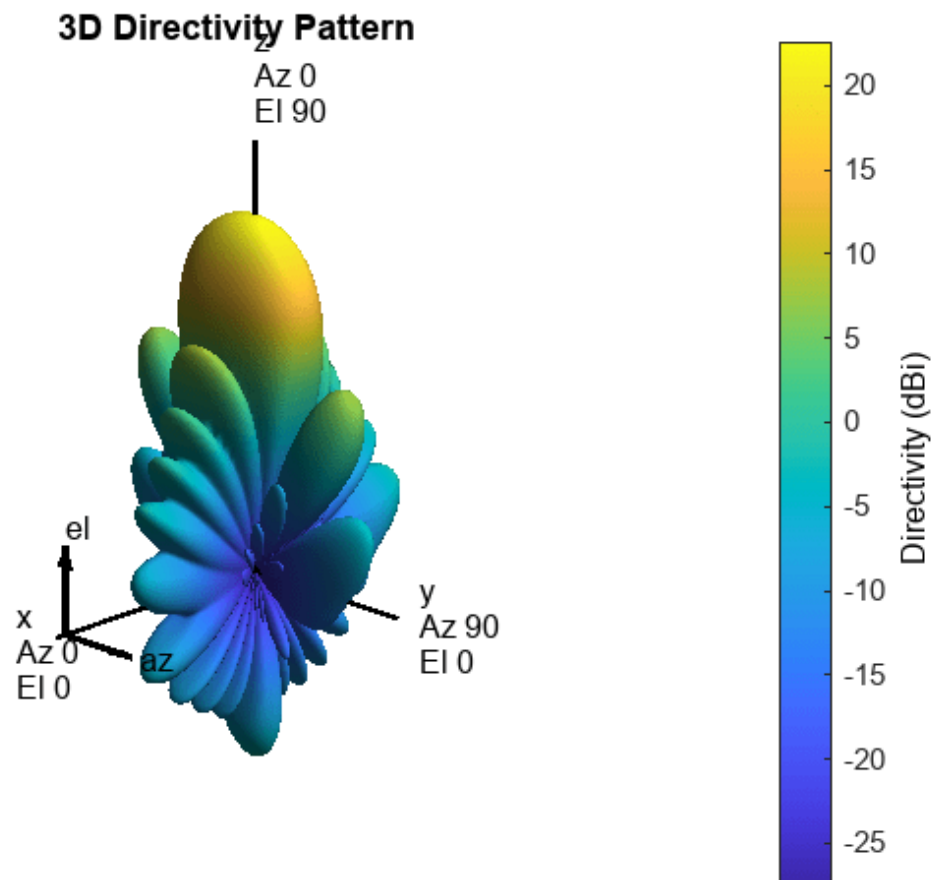
```
array = phased.ULA;
array.Element = antenna;
```

```
array.NumElements = 8;
array.ElementSpacing = 3.5e-3;
```

Pattern Multiplication

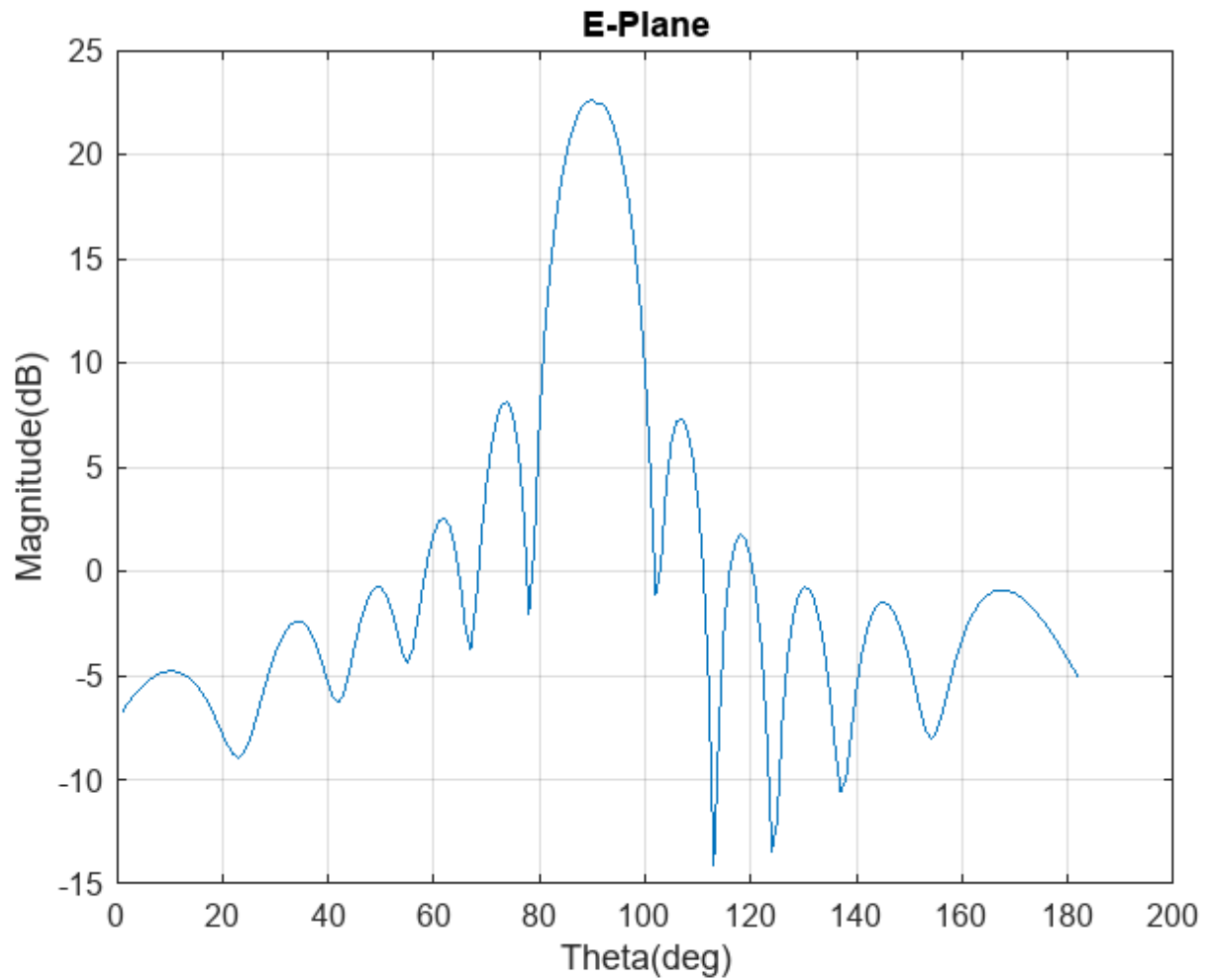
Use the `pattern` function to plot the 3-D radiation pattern.

```
figure;
pattern(array,28e9)
```



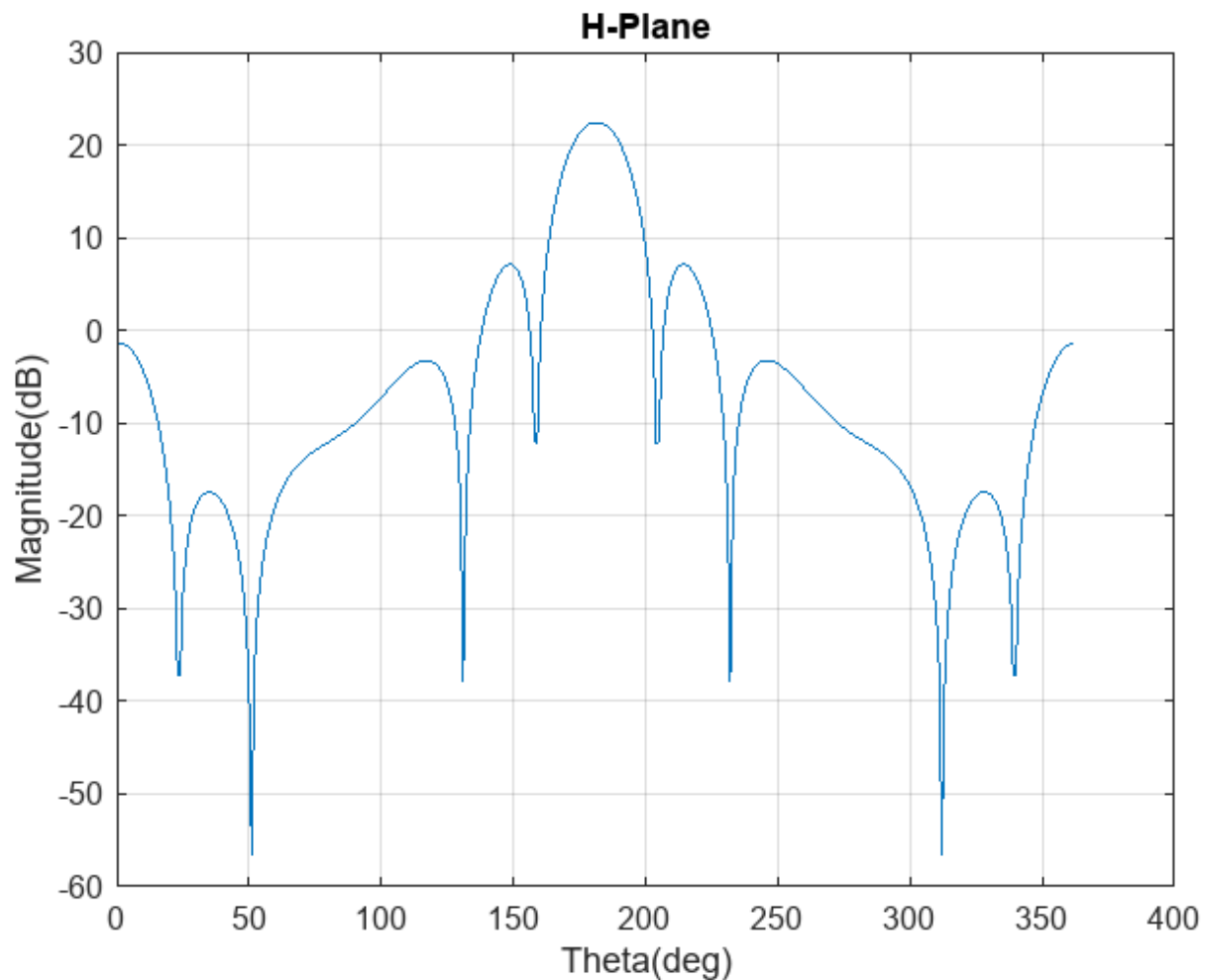
Use the `pattern` function to plot the radiation pattern in the E-plane

```
pat1 = pattern(array,28e9,0,0:1:90,CoordinateSystem='rectangular');
pat2 = pattern(array,28e9,180,0:1:90,CoordinateSystem='rectangular');
data1 = [pat1;flip(pat2)];
figure,plot(data1);
grid on;
xlabel('Theta(deg)');
ylabel('Magnitude(dB)');
title('E-Plane');
```



Use the pattern function to plot the radiation pattern in the H-plane

```
pat3 = pattern(array,28e9,90,-90:1:90,CoordinateSystem='rectangular');
data2 = [pat3;flip(pat3)];
figure,plot(data2);
grid on;
xlabel('Theta(deg)');
ylabel('Magnitude(dB)');
title('H-Plane');
```



Return Loss and Pattern Data Computations

The following code was used to generate 'seriesPatchData.mat' file used in this example.

```
type RLpatDataComputation.m

out = sparameters(ant,linspace(27.5e9,28.5e9,11));
figure;
rfplot(out);
figure;
pattern(ant, 28e9)
figure;
impedance(ant,linspace(27.5e9,28.5e9,11));
```

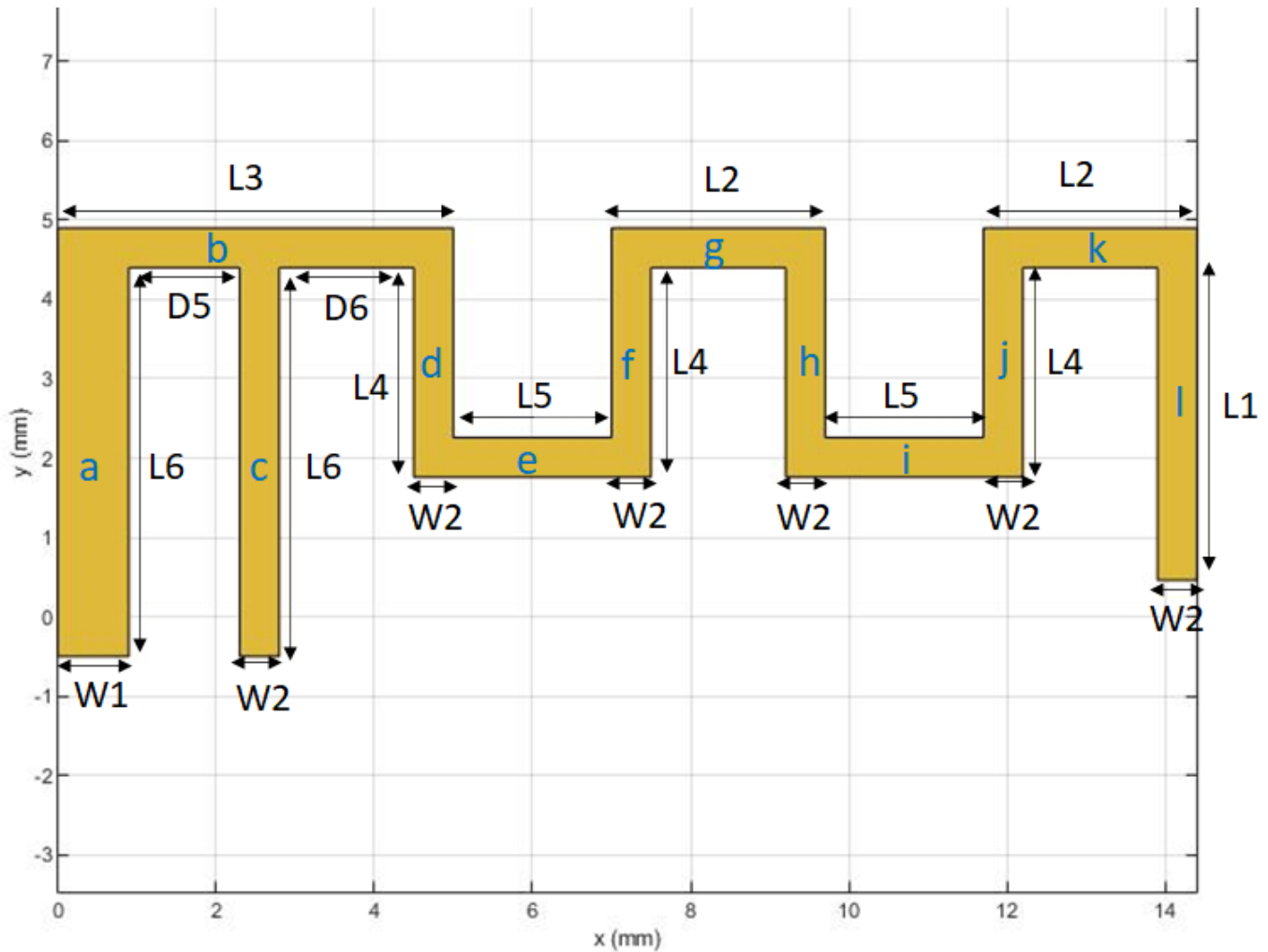
References

[1] "EM Simulation of 28 GHz Series-Fed Patch Antenna Array for 5G | 2019-02-01 | Microwave Journal." Accessed January 21, 2022. <https://www.microwavejournal.com/articles/31731-em-simulation-of-28-ghz-series-fed-patch-antenna-array-for-5g>.

[2] Ishfaq, Muhammad Kamran, Tharek Abd Rahman, Yoshihide Yamada, and Kunio Sakakibara. "8×8 Phased Series Fed Patch Antenna Array at 28 GHz for 5G Mobile Base Station Antennas." In *2017 IEEE-APS Topical Conference on Antennas and Propagation in Wireless Communications (APWC)*, 160-62. Verona, Italy: IEEE, 2017. <https://doi.org/10.1109/APWC.2017.8062268>.

PCB Antenna for USB Dongle and BLE Applications

This example shows how to design and analyze a small PCB antenna usually used on a USB dongle and in BLE applications. The antenna design and dimensions are as specified in the application note in [1]. This antenna design requires no more than 15.2 by 5.7 mm of space and ensures a Voltage Standing Wave Ratio (VSWR) of less than 2 across the 2.4 GHz Industrial, Scientific, and Medical frequency (ISM) band when you connect antenna to a 50 ohm source.



Create Variables

Assign the variables to create the top metal layer.

```
L1 = 3.94e-3;
L2 = 2.70e-3;
L3 = 5e-3;
L4 = 2.64e-3;
L5 = 2e-3;
L6 = 4.90e-3;
```



```

W1 = 0.90e-3;
W2 = 0.5e-3;

D4 = 0.5e-3;
D5 = 1.4e-3;
D6 = 1.70e-3;

Lgp = 15.2e-3; % Length of the Groundplane
Wgp = 30e-3;   % Width of the Groundplane

LBS = 15.2e-3; % Length for the Board Shape
WBS = 5.2e-3; % Width of the Board Shape

```

Create Geometry

Use the `antenna.Rectangle` object to create all the rectangular sections as shown in the schematic above.

```

a = antenna.Rectangle('Length',W1,'Width',L6,'Center',[W1/2,L6/2-D4]);
b = antenna.Rectangle('Length',L3,'Width',W2,'Center',[L3/2,W2/2+L6-D4]);
c = antenna.Rectangle('Length',W2,'Width',L6,'Center',[W2/2+D5+W1,L6/2-D4]);
d = antenna.Rectangle('Length',W2,'Width',L4,'Center',[W2/2+W1+D5+W2+D6,-L4/2+L6-D4]);
e = antenna.Rectangle('Length',L5,'Width',W2,'Center',[L5/2+L3,-W2/2+L6-L4]);
f = antenna.Rectangle('Length',W2,'Width',L4,'Center',[W2/2+W1+D5+W2+D6+W2+L5,-L4/2+L6-D4]);
g = antenna.Rectangle('Length',L2,'Width',W2,'Center',[L2/2+W1+D5+W2+D6+W2+L5,W2/2+L6-D4]);
h = antenna.Rectangle('Length',W2,'Width',L4,'Center',[W2/2+W1+D5+D6+W2+L5+L2,-L4/2+L6-D4]);
i = antenna.Rectangle('Length',L5,'Width',W2,'Center',[L5/2+L3+L5+L2,-W2/2+L6-L4]);
j = antenna.Rectangle('Length',W2,'Width',L4,'Center',[W2/2+W1+D5+W2+D6+W2+L5+L2+L5,-L4/2+L6-D4]);
k = antenna.Rectangle('Length',L2,'Width',W2,'Center',[L2/2+W1+D5+W2+D6+W2+L5+L2+L5,W2/2+L6-D4]);
l = antenna.Rectangle('Length',W2,'Width',L1,'Center',[W1+D5+W2+D6+W2/2+L5+L2+L5+L2,-L1/2+L6-D4]);

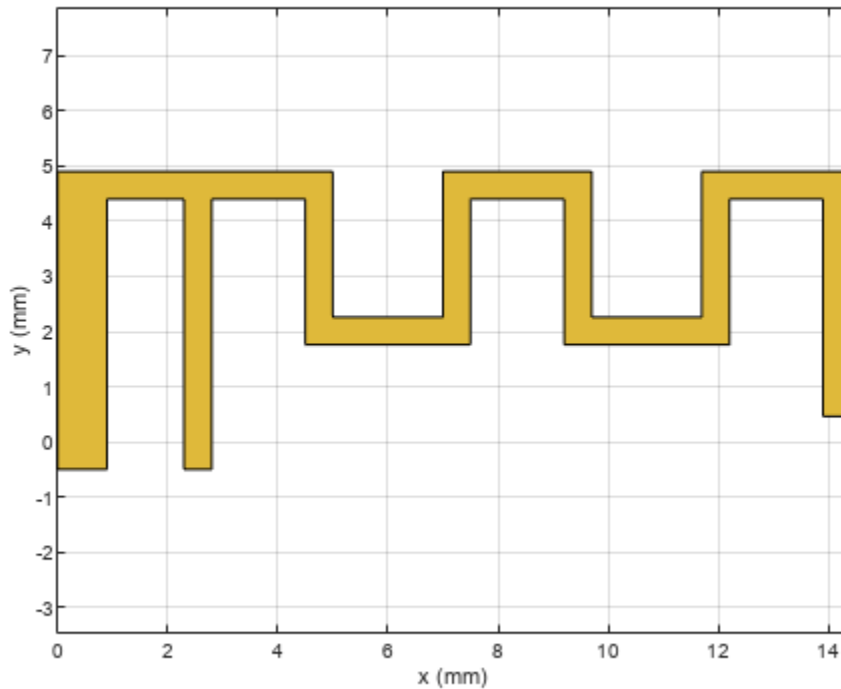
```

Join all the rectangles and visualize the shape.

```

Newobj = a+b+c+d+e+f+g+h+i+j+k+l;
figure;
show(Newobj)

```



Design PCB Board

Design the PCB board as per the specifications in section 4.3 of the application note in [1]. Information given in [1] is also given below.

1-2 LAYER PCB 0.25 MM NOMINAL

2-3 LAYER PCB 0.50 MM NOMINAL

3-4 LAYER PCB 0.25 MM NOMINAL

THICKNESS FR4 WITH 35um Cu PER LAYER

Dielectric constant for FR4 is 4.5

The ground plane dimensions are taken as 30 mm and the dielectric substrate is FR4 with a total board height of 1 mm. The PCB antenna has four layers and the details of the layers are given below.

1) Radiating Layer - Conductor

2) Dielectric Layer- 0.25 mm

3) Ground Layer - Conductor

4) Dielectric Layer- 0.75 mm - This height is taken after combining the 2-3 layer and 3-4 layer as given in the PCB description

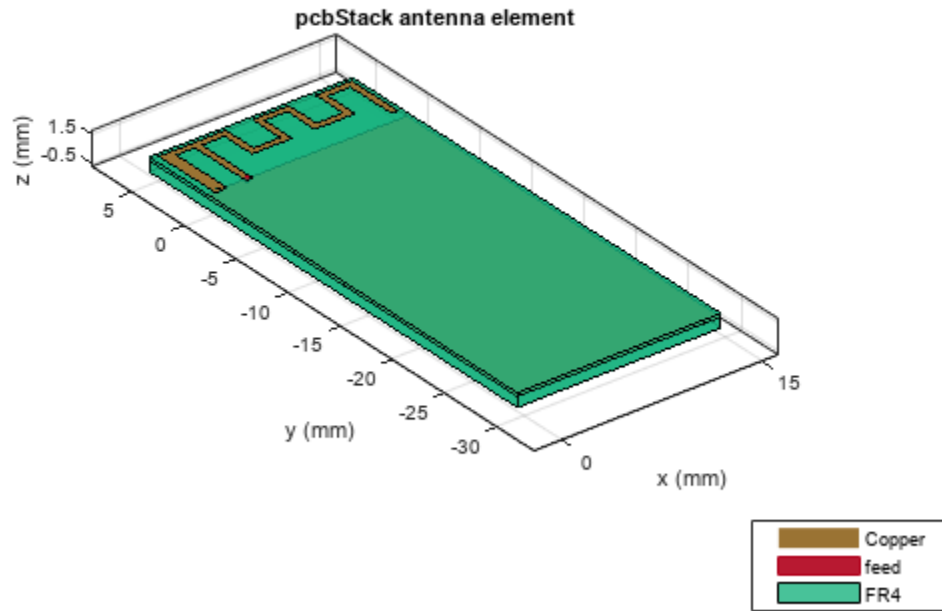
Define the dielectric substrate for the antenna. The antenna has the top radiating layer followed by a dielectric layer of 0.25 mm in thickness. The third layer is the ground layer, which is followed by another dielectric layer 0.75 mm in thickness.

```
d = dielectric('FR4');
d.Thickness = 0.25e-3;
d.EpsilonR = 4.5;
d1 = dielectric('FR4');
d1.Thickness = 0.75e-3;
d1.EpsilonR = 4.5;
```

Use the `pcbStack` object to create the PCB stack of the antenna. Use the `antenna.Rectangle` object to create the board and the ground plane. Assign the board, dielectrics, and ground plane to the `Layers` property of `pcbStack`. Set the `FeedLocations` and `ViaLocations` property to create a feed and via from the first layer to the third layer, and set the `BoardThickness` to 1 mm.

Use the `show` function to visualize the antenna.

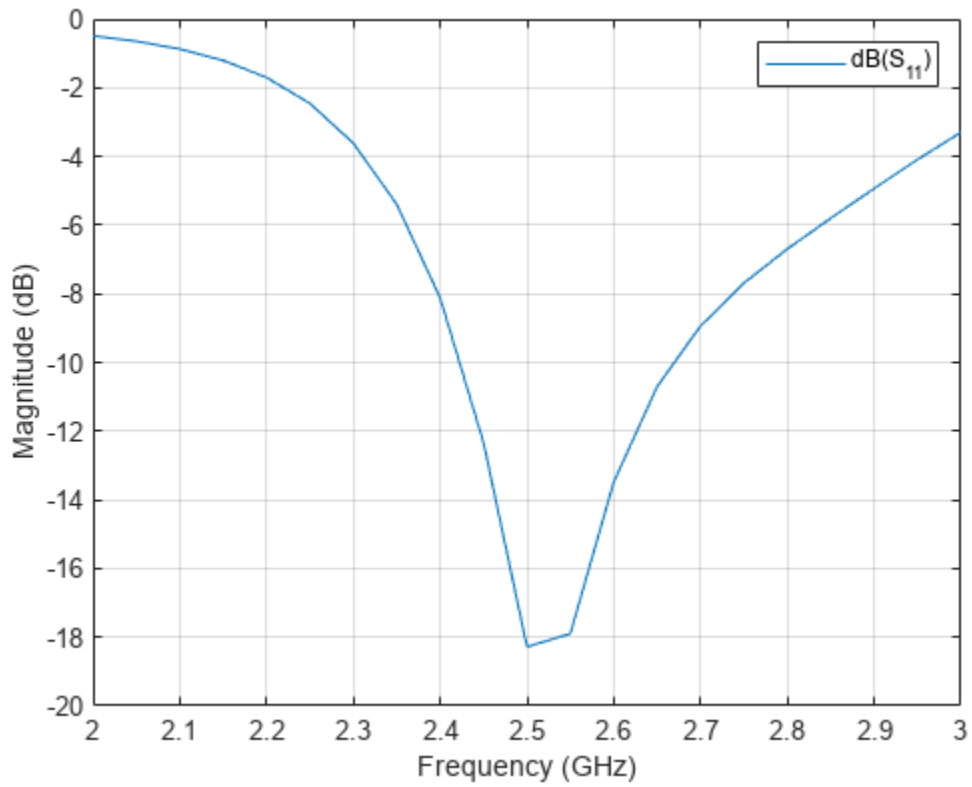
```
ant = pcbStack;
boardshape = antenna.Rectangle('Length',LBS,'Width',WBS+Wgp,'Center',[LBS/2-0.5e-3,(WBS-Wgp)/2]);
ant.BoardShape = boardshape;
ant.BoardThickness = d.Thickness + d1.Thickness;
gnd = antenna.Rectangle('Length',Lgp,'Width',Wgp,'Center',[Lgp/2-0.5e-3,-Wgp/2]);
ant.Layers = {Newobj,d,gnd,d1};
ant.FeedDiameter = W2/2;
ant.ViaDiameter = W1/4;
ant.FeedLocations = [W1+1.4e-3+W2/2,-D4/2,3,1];
ant.ViaLocations = [W1/2,-D4/2,3,1];
ant.FeedViaModel = 'square';
ant.Conductor = metal('Copper');
ant.Conductor.Thickness = 35e-6;
figure;
show(ant)
```



Analyze Antenna Performance

Use the `sparameters` function to calculate the S-parameters and plot them using the `rfplot` function.

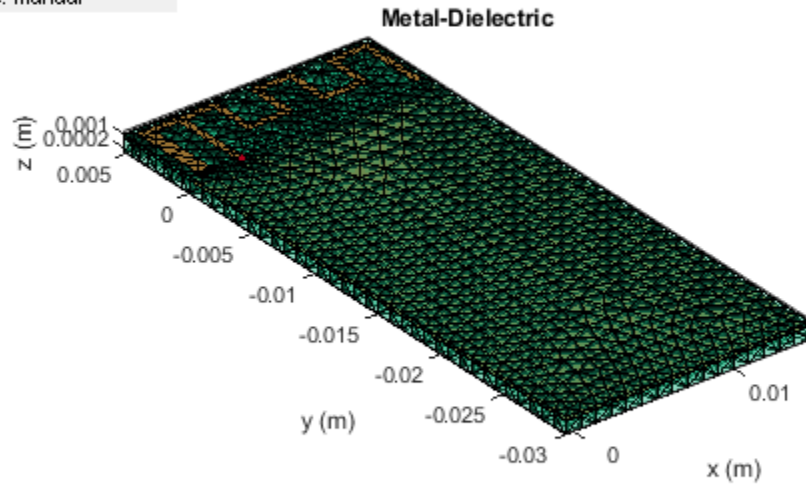
```
spar = sparameters(ant, linspace(2e9, 3e9, 21));  
figure;  
rfplot(spar);
```



Use the mesh function with a MaxEdgeLength of 1.2 mm to generate a denser mesh, to ensure around 50 triangles per wavelength.

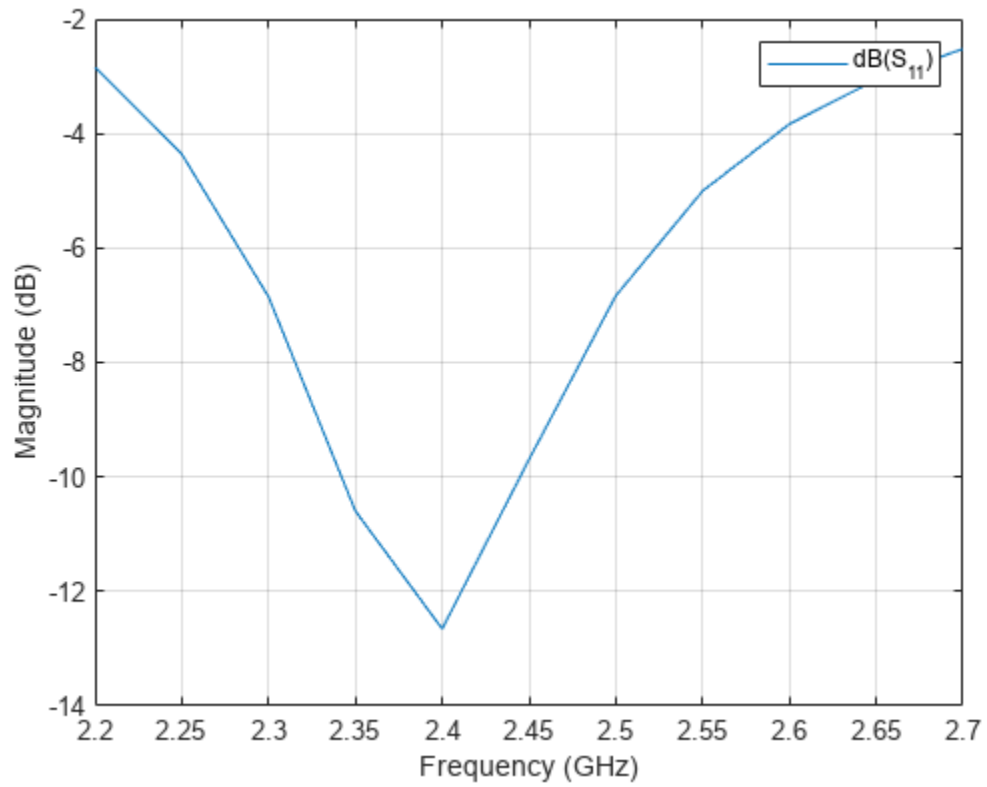
```
figure;  
mesh(ant, 'MaxEdgeLength', 1.2e-3);
```

NumTriangles: 1096
NumTetrahedra: 7728
NumBasis:
MaxEdgeLength: 0.0012
MeshMode: manual



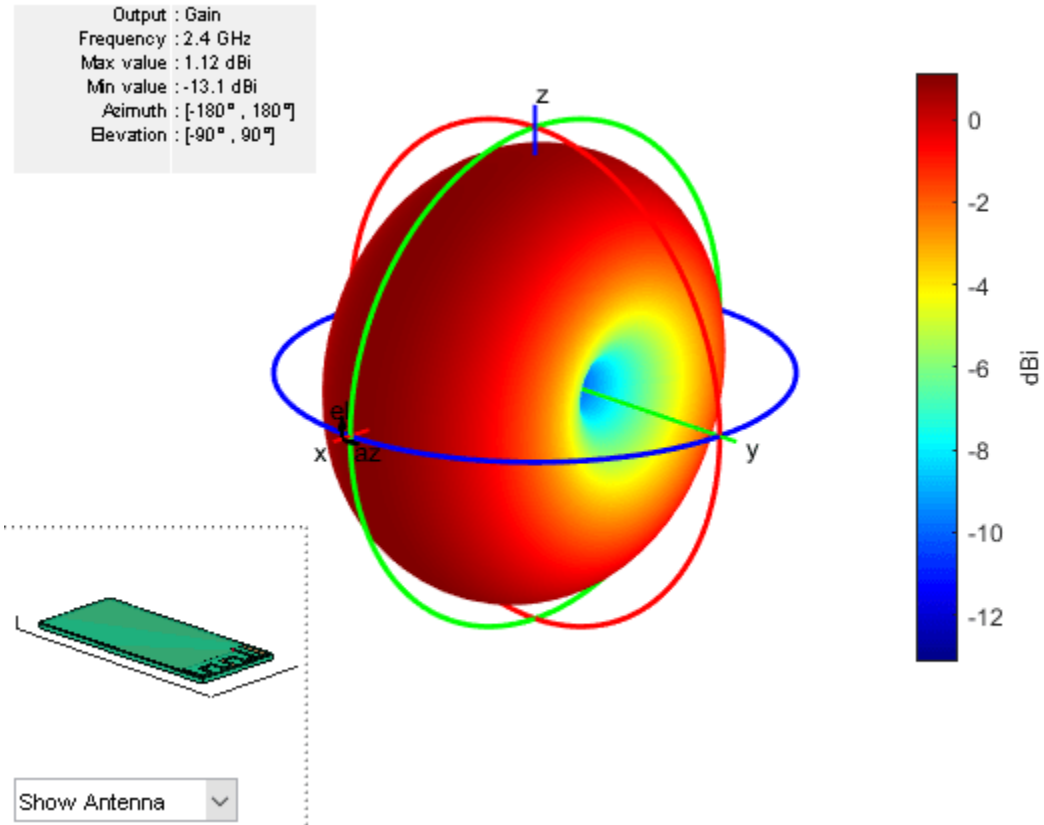
Use the `sparameters` function to calculate the S-parameters for the improved mesh and plot them using the `rfplot` function.

```
spar = sparameters(ant,linspace(2.2e9,2.7e9,11));  
figure;  
rfplot(spar)
```



Use the pattern function to plot the 3-D pattern of the antenna.

```
figure;  
pattern(ant,2.4e9)
```



References

- [1] Andersen, Audun. "Small Size 2.4 GHz PCB Antenna ." Texas Instruments. Accessed January 14, 2022. <https://www.ti.com/lit/an/swra117d/swra117d.pdf>.

Miniaturize Patch Antennas Using Metamaterial-Inspired Technique

This example shows how to design a miniaturized patch antenna at 2.4 GHz with a complementary split-ring resonator (CSRR) loading plane and evaluate its performance, with results as published in [1]. The antenna is miniaturized by shrinking its circular patch structure from the original radius of 23.1 mm to 6 mm.

Define Patch Dimensions

The microstrip patch in this example has 3 layers: top layer, loading plane, and ground plane. The board is a square with the side length B_d . r_0 is the original radius of the resonant patch in the top layer, which is shrunk to r . The CSRR has N complementary split rings in the loading plane, with the split width of w and the inner and outer radius of $R2_in$ and $R2_out$, respectively.

```
clc
r_0=23.1e-3;
r = 6e-3;
B_d =2.2*r_0;
N=7;
w = 0.011*N;
R2_in=0.3521;
R2_out=(R2_in*N/1.9079) -w;
```

Design Top Layer

Create a circular patch with the radius of r .

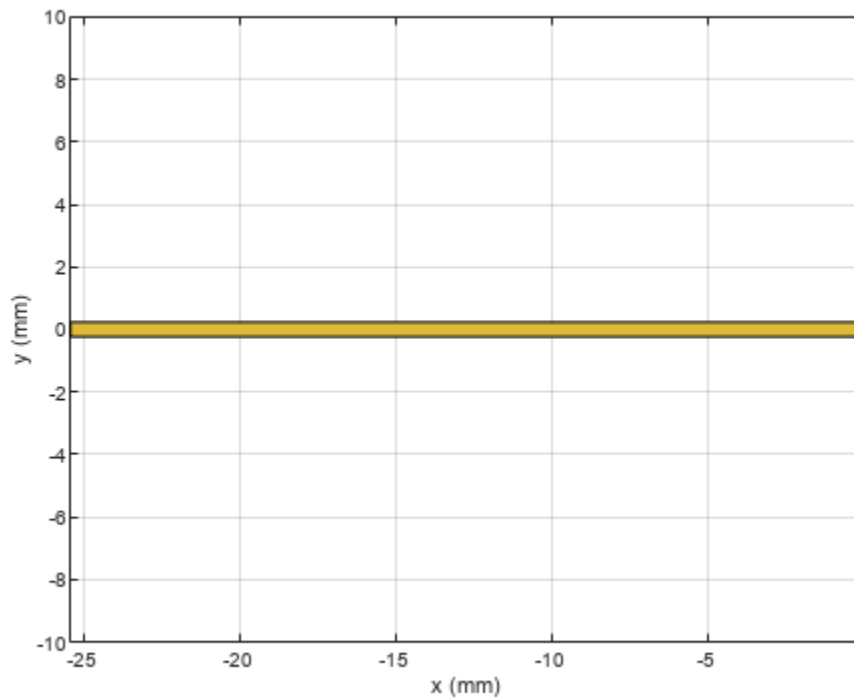
```
circle_L1= antenna.Circle(Center=[0 0],Radius=r);
```

Create the feed as a rectangle.

```
rect_L1 = antenna.Rectangle(Length=B_d/2,Width=w*r);
```

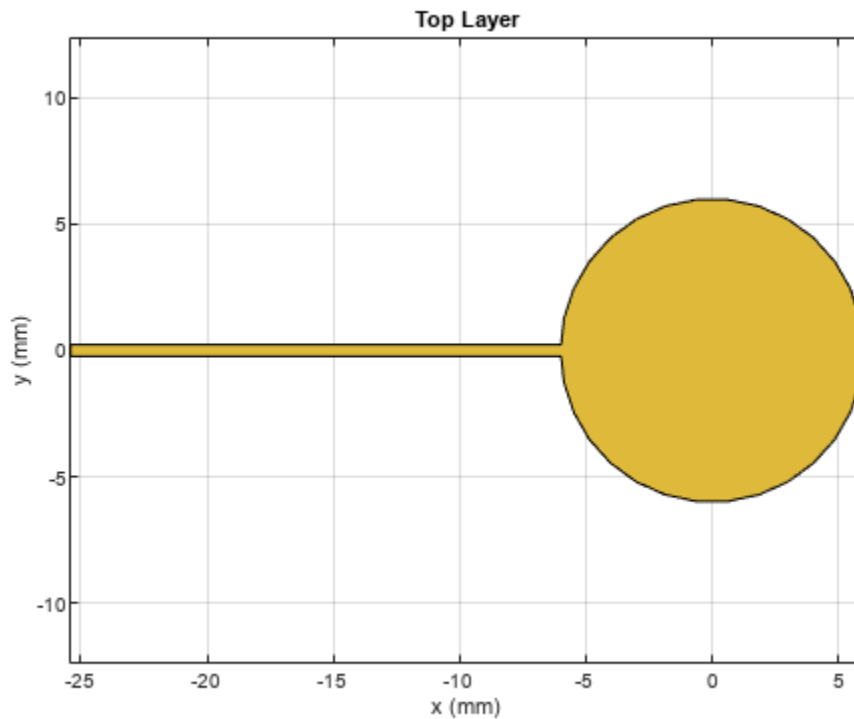
Move the feed in the x - y plane to connect it to the circular patch.

```
translate(rect_L1, [-B_d/4 0 0]);
```



Create a polygon combining the patch and the feed, and then plot the resulting shape.

```
polygon_L1 = circle_L1+rect_L1;  
show(polygon_L1);  
title('Top Layer')
```



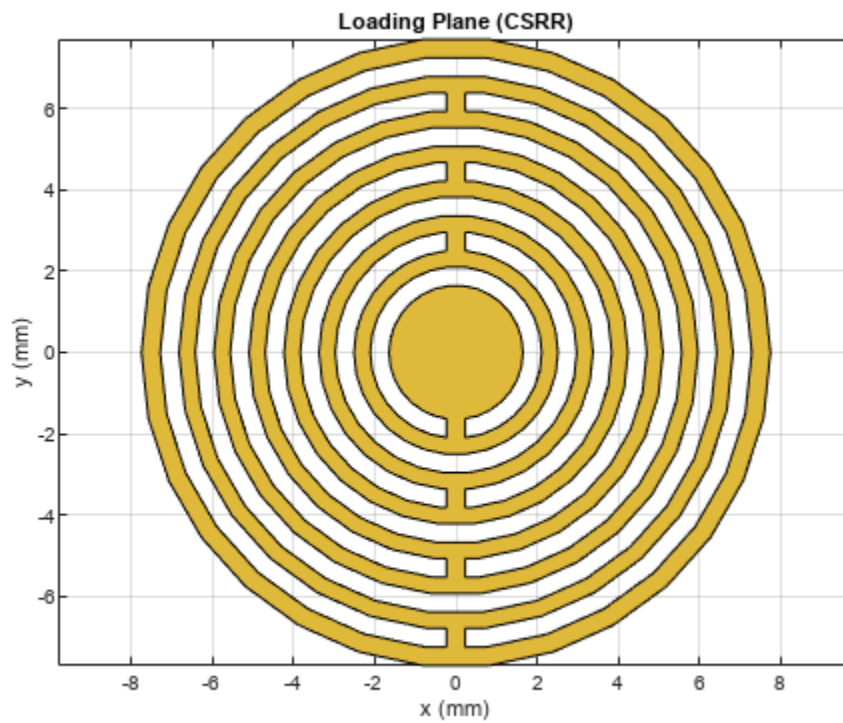
Design Loading Plane

To create a CSRR with seven rings, first, create a variable `r_rad` which generates multiplying factors for each of the seven rings. Then use a MATLAB for loop to iteratively add and remove circles to create the ring slots that make up the CSRR.

```

r_rad=linspace(R2_out,R2_in,N);
sign=-1;
circ_outer_L2 = antenna.Circle(Center=[0 0],Radius=(R2_out+w)*r);
for i=1:length(r_rad)
    circle_Minus = antenna.Circle(Center=[0 0],Radius=r_rad(i)*r);
    circle_Plus = antenna.Circle(Center=[0 0],Radius=(r_rad(i)-w)*r);
    rect_Minus = antenna.Rectangle(Center=[0,sign*(r_rad(i)-w/2)*r],Length=w*r,Width=(w+w/1.3)*r);
    if i==1
        spliti=circ_outer_L2-circle_Minus+circle_Plus+rect_Minus;
        CSRR_L2=spliti;
    end
    CSRR_L2=CSRR_L2-circle_Minus+circle_Plus+rect_Minus;
    sign=sign*-1;
end
show(CSRR_L2);title('Loading Plane (CSRR)')

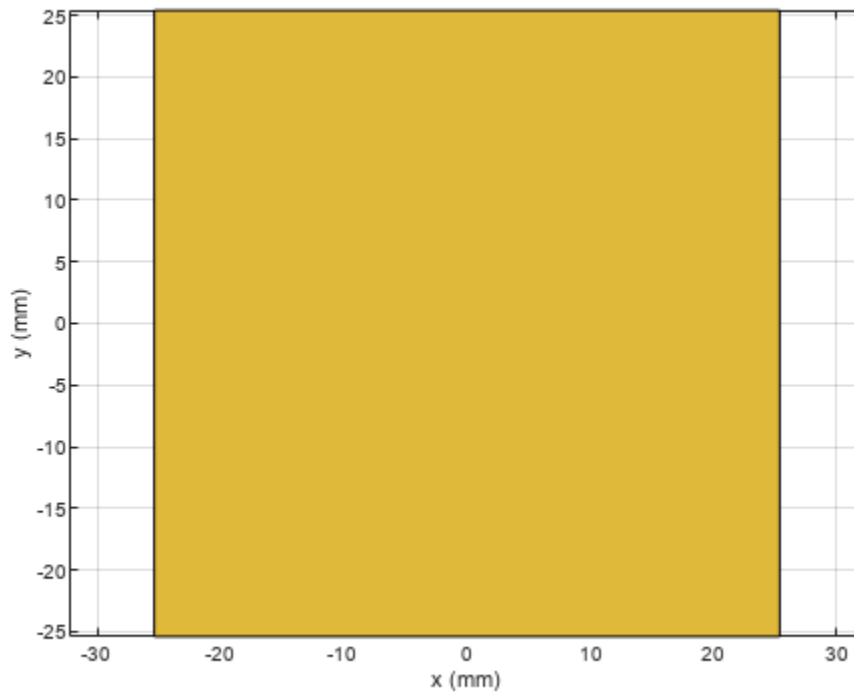
```



Design Ground Plane

Design the ground plane as a rectangle with the same dimension as that of the board.

```
rect_L3 = antenna.Rectangle(Length=B_d,Width=B_d);  
show(rect_L3)
```



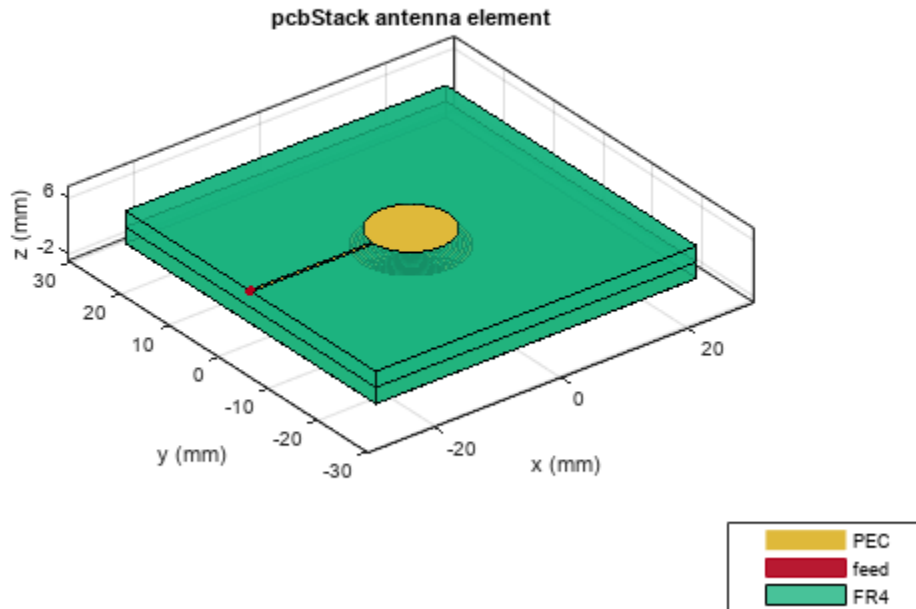
Design Board

Define the shape of the board and create the stack with the layers mentioned above and specified dielectric layers in between.

```
boardShape = antenna.Rectangle(Length=B_d,Width=B_d);
```

Create PCB stack using previously defined layers and two dielectric layers.

```
p = pcbStack;
p.BoardShape = boardShape;
d1 = dielectric('FR4');
% d1 = dielectric('RT5870');
d1.Thickness = 2.34e-3;
% d2 = dielectric('RT5870');
d2 = dielectric('FR4');
d2.Thickness = 2.34e-3;
p.BoardThickness = d1.Thickness+d2.Thickness;
p.Layers = {polygon_L1,d1,CSRR_L2,d2,rect_L3};
p.FeedLocations = [-B_d/2 0 1 3];
figure
show(p)
```



Analyze Performance of Miniature Antenna

Before evaluating the board's performance, estimate the memory required to solve the mesh structure. You can do this by using the memory estimator.

```
memoryEstimate(p,2.4e9)
```

```
ans =  
'4.4 GB'
```

Based on the memory estimate, you can utilize parallel computing in MATLAB to accelerate the simulation. For the purpose of this example, load a MAT file containing the return loss computations. You can view the code used to compute the return loss in the **Compute Return Loss** section.

```
load("RL_linear.mat");
```

Plot the return loss of the patch. The patch is resonant at 2.396 GHz, even though its dimension is much smaller than its natural resonant size. Its area is 16 times smaller than the resonant area, as reported in [1].

```
plot(freq,RL_parfor,Marker='o',LineWidth=2);
```

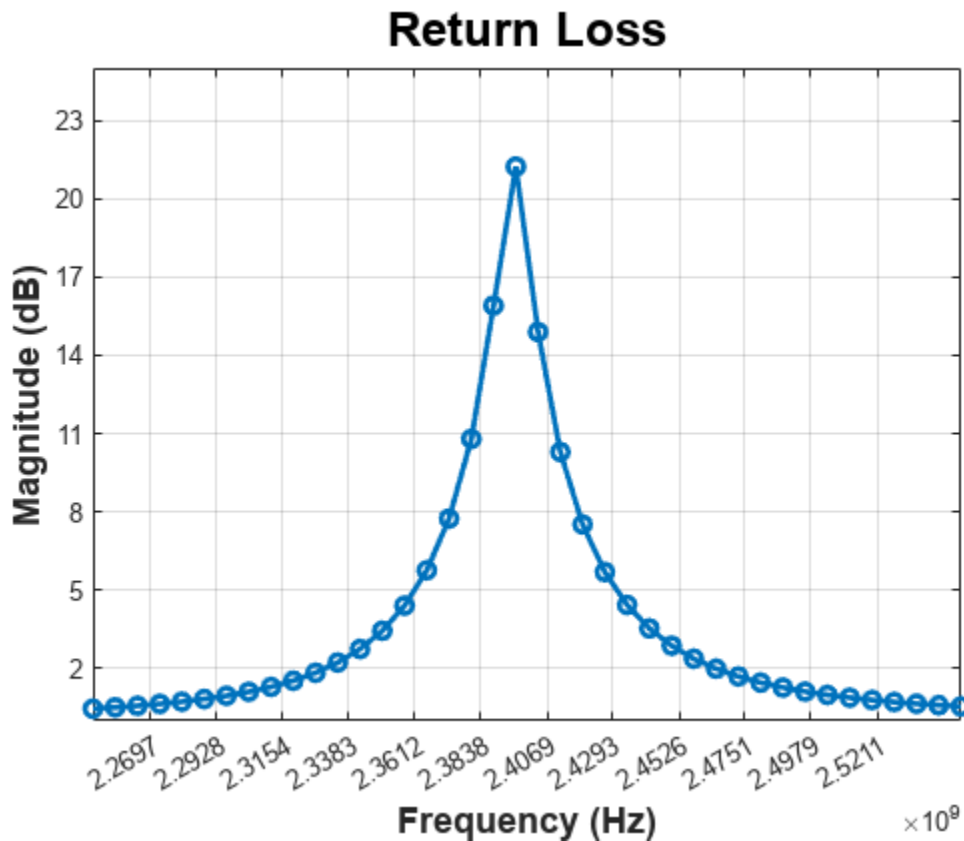
```
grid on  
% Create ylabel  
ylabel({'Magnitude (dB)'},FontWeight='bold',FontSize=14);
```

```
% Create xlabel
```

```

xlabel({'Frequency (Hz)',FontWeight='bold',FontSize=14);
% Create title
title({'Return Loss'},FontWeight='bold',FontSize=18);
% Set XTicks and Yticks
set(gca,XTick=decimate(freq,3))
set(gca,YTick=floor(min(RL_parfor))-1:3:ceil(max(RL_parfor)+1))

```



The linearly spaced frequency points cannot capture the varying resonance between 2.36 GHz and 2.43 GHz. Alternatively, you can use the `densespace` function to generate the 40 frequency points that are dense around the resonant frequency. Call this function instead of the `linspace` in return loss computation. For the purpose of this example, load a MAT file containing the return loss computations done using `densespace` function.

```
load("resR23521.mat");
```

Plot the return loss with the dense space.

```

f = figure;
f.Position = [100 100 1820 500];
plot(freq,RL_parfor,Marker='o',LineWidth=2);
% Create ylabel
ylabel({'Magnitude (dB)',FontWeight='bold',FontSize=14);
% Create xlabel

```

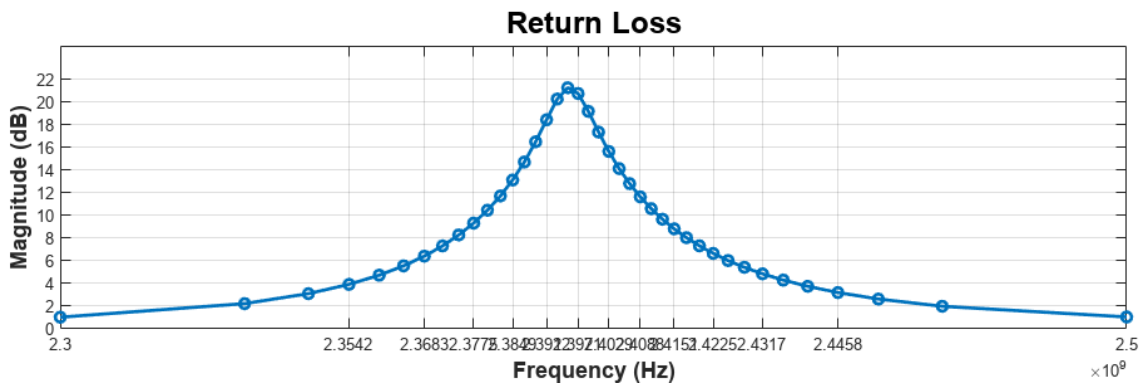
```

xlabel({'Frequency (Hz)'},FontWeight='bold',FontSize=14);

% Create title
title({'Return Loss'},FontWeight='bold',FontSize=18);

% Set XTicks and Yticks
f_points=round(downsample(freq,3),3);
% set(gca,'XTick',round(downsample(freq,3),10))
set(gca,XTick=f_points)
set(gca,YTick=floor(min(RL_parfor))-1:2:ceil(max(RL_parfor)+1))
grid on

```



Plot Radiation Pattern

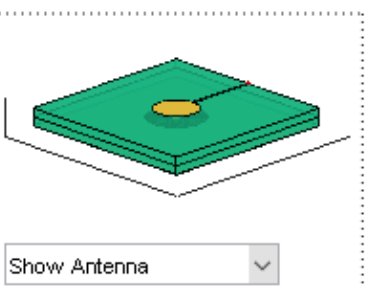
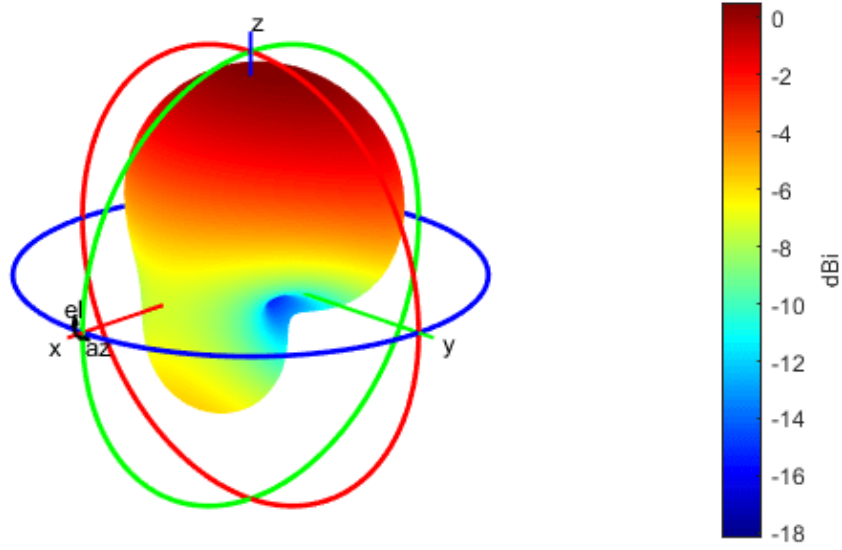
Plot the 3-D radiation pattern of the patch.

```

f = figure;
f.Position = [100 100 720 400];
pattern(p,2.396e9)

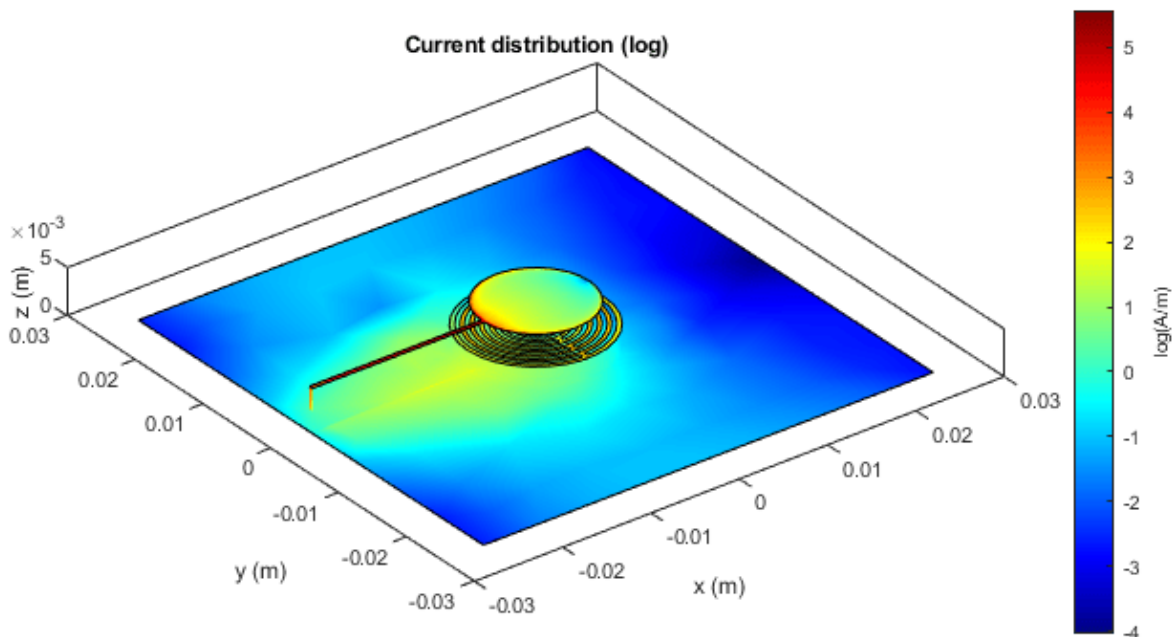
```


Output : Gain
 Frequency : 2.396 GHz
 Max value : 519 mdB
 Min value : -18.2 dBi
 Azimuth : [-180°, 180°]
 Elevation : [-90°, 90°]



Visualize the current density of the patch.

```
current(p, 2.396e9, scale='log')
```



Compute Return Loss

This example uses the following commented code to compute the return loss of the designed patch antenna.

```
%returnLoss(p,linspace(2.25e9,2.55e9,31));
% freq = linspace(2.25e9,2.55e9,40);

% RL_parfor = zeros(size(freq));
%
% tic;
% temp = returnLoss(p, freq(end));
% meshdata = mesh(p);
% [~, ~] = mesh(p, 'MaxEdgeLength', meshdata.MaxEdgeLength);
% parfor m = 1:numel(freq)
%     m
%     RL_parfor(m) = returnLoss(p, freq(m));
% end
% time4 = toc
```

Reference

[1] Ouedraogo, Raoul O., Edward J. Rothwell, Alejandro R. Diaz, Kazuko Fuchi, and Andrew Temme. "Miniaturization of Patch Antennas Using a Metamaterial-Inspired Technique." *IEEE Transactions on Antennas and Propagation* 60, no. 5 (May 2012): 2175–82. <https://doi.org/10.1109/TAP.2012.2189699>.

Design and Analysis of Compact Ultra-Wideband MIMO Antenna Array

This example shows the design and analysis of a compact ultra-wideband (UWB) Multiple-Input Multiple-Output (MIMO) antenna array as referred in [1]. The American Federal Communications Commission (FCC) allowed commercial use in the 3.1 GHz to 10.6 GHz frequency range from the year 2002. Since then, there has been a significant amount of research done to develop antenna technology for this ultra-wide frequency range. The main challenge in designing a UWB antenna with such wide bandwidth is the multipath fading. A popular solution to overcome this fading effect is to use the MIMO antenna array technology which also increases the channel capacity of UWB systems.

Create Upper Patches of Monopole Antennas

The MIMO array consists of two orthogonally placed planar monopole antennas lying in the horizontal plane. The size of the upper conductor of each monopole antenna is 10 mm by 16 mm. The dimension of the conductor backed dielectric substrate is 40 mm by 26 mm. The square radiator of the first monopole is at an offset of (7 mm, 14 mm, 0 mm) from one of the corners of the substrate. The second monopole's radiator is at an offset of (14 mm, 9 mm, 0 mm) from the diagonally opposite corner of the substrate.

```
% Specify the geometry dimensions in m
% The dimension of the square patch of each monopole radiator
Lp=10e-3;
% The length of the substrate along X-axis
Xg=40e-3;
% The width of the substrate along Y-axis
Yg=26e-3;
% The offset of the first monopole radiator
px1=-Xg/2+7e-3;
py1=-Yg/2+14e-3;
% Lower left patch
patch1 = antenna.Rectangle(Center=[0,0],Length=Lp,Width=Lp,Center=[px1, py1]);
% The offset of the second monopole radiator
px2=Xg/2-9e-3-Lp/2;
py2=Yg/2-8.1e-3-0.9e-3;
% Upper right patch
patch2 = antenna.Rectangle(Center=[0,0],Length=Lp,Width=Lp,Center=[px2, py2]);
```

Create Feed Lines of Monopole Antennas

Feed both monopole antennas using two orthogonally oriented identical planar feedlines with length and width of 9 mm and 1.8 mm, respectively.

```
% Upper left patch feedline
feed1 = antenna.Rectangle(Center=[0,0],Length=1.8e-3,Width=9e-3,Center=[px1, py1-Lp/2-9e-3/2]);
% Upper right patch feedline
feed2 = antenna.Rectangle(Center=[0,0],Length=9e-3,Width=1.8e-3,Center=[px2+Lp/2+9e-3/2, py2]);
```

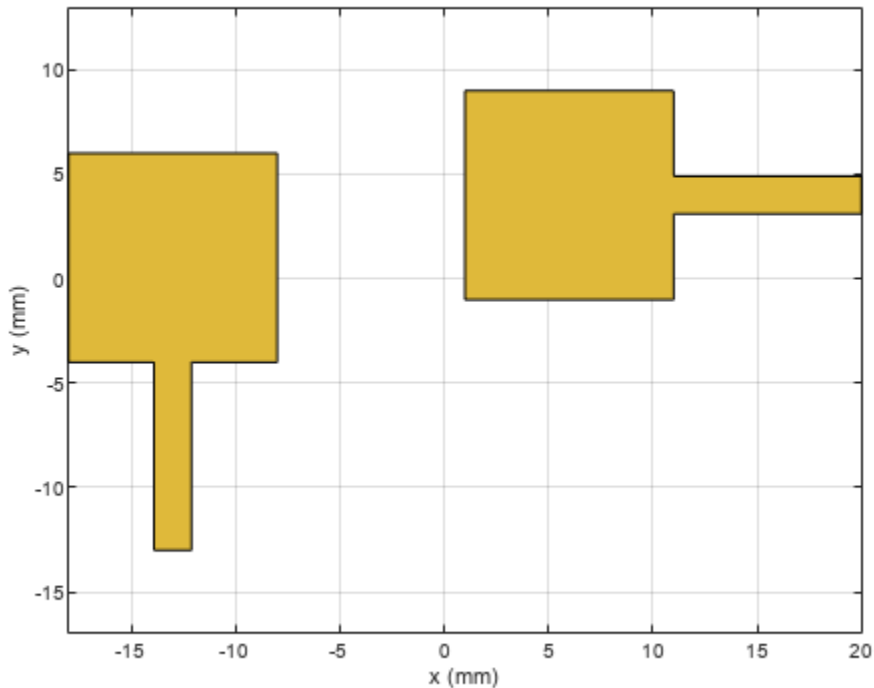
Create and Visualize Upper Layer

Join the patches to create the upper layer of the MIMO array which consists of two square radiators and two feed lines of the monopole antennas. Use the show function to visualize the uppermost layer of the array.

```

% Join four components
patchUpper=patch1+patch2+feed1+feed2;
figure;
show(patchUpper)

```



Create and Visualize Ground Layer

The ground layer consists of two rectangular shapes, two rectangular slots and three stubs. The purpose of introducing such perturbation in the ground plane is to achieve good impedance matching across the entire UWB band and to minimize the mutual coupling between the two monopole antennas of the MIMO array.

```

% Create left ground plane
% Left rectangular ground block
ground1 = antenna.Rectangle(Center=[0,0],Length=29e-3,Width=8e-3,Center=[-Xg/2+29e-3/2, -Yg/2+4e-3]);
% Slot in the left rectangular ground block
groundcut1 = antenna.Rectangle(Center=[0,0],Length=4e-3,Width=1e-3,...
    Center=[-Xg/2+5e-3+4e-3/2, -Yg/2+8e-3-1e-3/2]);
% Subtract the slot from the ground in the left ground block
ground1=ground1-groundcut1;

% Create right ground plane
% Right rectangular ground block
ground2 = antenna.Rectangle(Center=[0,0],Length=8e-3,Width=26e-3,Center=[Xg/2-4e-3, 0]);
% Slot in the right rectangular ground block
groundcut2 = antenna.Rectangle(Center=[0,0],Length=1e-3,Width=4e-3,Center=[Xg/2-8e-3+1e-3/2, py2]);
% Subtract the slot from the ground in the right ground block

```

```

ground2=ground2-groundcut2;

% Ground interconnect
ground3 = antenna.Rectangle(Center=[0,0],Length=3e-3,Width=1e-3,Center=[Xg/2-8e-3-3e-3/2, -Yg/2+...

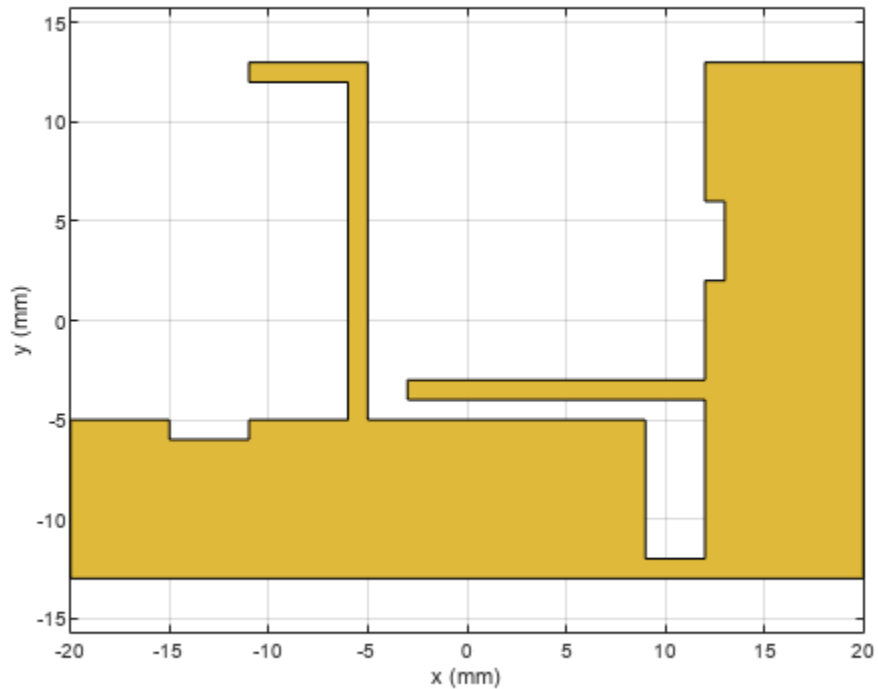
% Connect the three ground blocks
ground=ground1+ground2+ground3;

% First ground stub
groundstub1= antenna.Rectangle(Center=[0,0],Length=1e-3,Width=18e-3,...
    Center=[-Xg/2+14e-3+1e-3/2, Yg/2-18e-3/2]);
% Second ground stub
groundstub2= antenna.Rectangle(Center=[0,0],Length=5e-3,Width=1e-3,Center=[-Xg/2+14e-3-5e-3/2, Y...

% Third ground stub
groundstub3= antenna.Rectangle(Center=[0,0],Length=16e-3,Width=1e-3,...
    Center=[Xg/2-8e-3+1e-3-16e-3/2, Yg/2-16e-3-1e-3/2]);

% Connect all stubs with the ground block
ground=ground+groundstub1+groundstub2+groundstub3;
figure;
show(ground)

```



Assign Dielectric Properties

Design the entire MIMO array on a dielectric substrate with the relative permittivity and loss tangent values of 3.5 and 0.004, respectively. The thickness of the dielectric substrate is assumed to be 0.8 mm.

```
% Dielectric
d = dielectric;
d.EpsilonR = 3.5;
d.LossTangent=0.004;
h=0.8e-3;
```

Create pcbStack Object

Create a `pcbStack` object using the same dielectric substrate. Assign `patchUpper` and `ground` to the upper and bottom layers of the `pcbStack` object, respectively.

```
ps=pcbStack;
ps.BoardThickness=h;
d.Thickness = h;
r1bottom=antenna.Rectangle(Center=[0,0],Length=Xg,Width=Yg);
ps.BoardShape =r1bottom;
ps.Layers = {patchUpper,d,ground};
```

Excite Both Antennas

Define feed locations at the edge of the feedline of each antenna. Set the feed diameter manually to half of the feedline width. Excite each antenna with a unit amplitude and zero phase voltage source.

```
% Feed location of the first antenna
fx1=-Xg/2+6.1e-3+1.8e-3/2;
fy1=-Yg/2;
% Feed location of the second antenna
fx2=Xg/2;
fy2=Yg/2-8.1e-3-1.8e-3/2;
% Assign excitation to both feed locations
ps.FeedLocations=[fx1 fy1 1 3; fx2 fy2 1 3];
ps.FeedDiameter=1.8e-3/2;% in m
ps.FeedVoltage=[1 1];% in Volt
ps.FeedPhase=[0 0];% in degree
```

Mesh Geometry

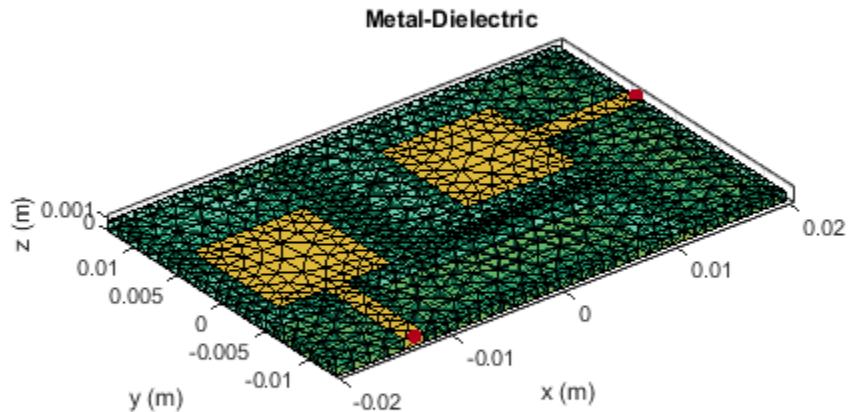
Manually mesh the array geometry with the maximum edge length of 1.8 mm.

```
ax=figure;
mesh(ps,MaxEdgeLength=1.8e-3)
```

```

NumTriangles: 902
NumTetrahedra: 4062
NumBasis:
MaxEdgeLength: 0.0018
MeshMode: manual

```



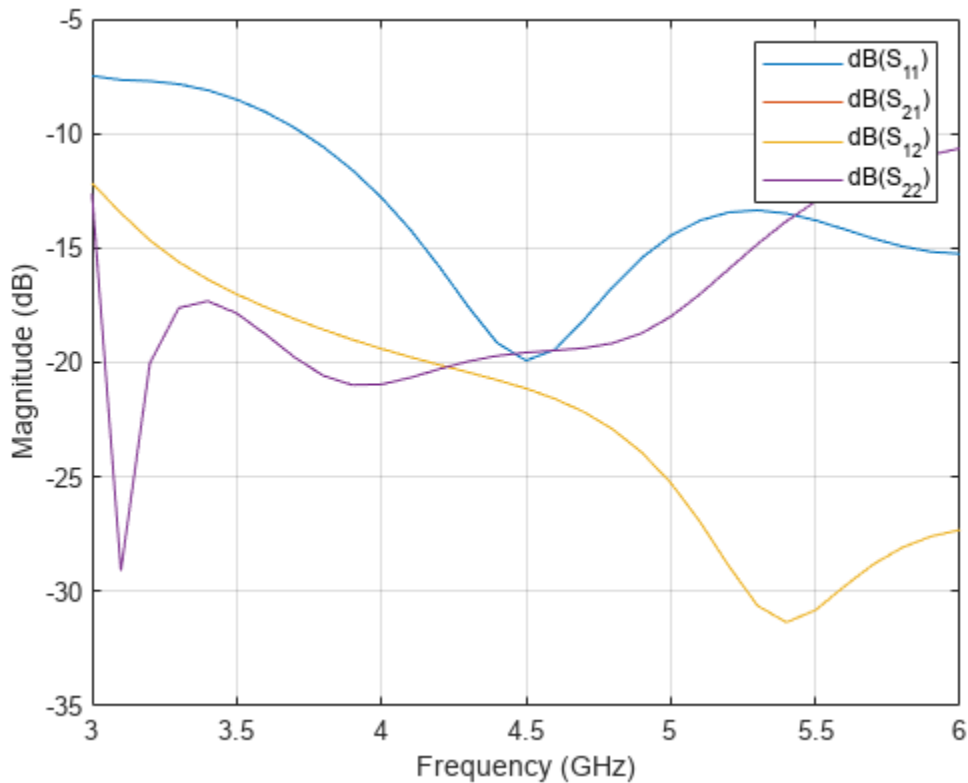
Visualize S-parameters of Array

Use the `sparameters` function to calculate the scattering parameters of the array. As it is a two-port array, it has four S-parameters, among which the S11 and S22 represent the reflection coefficient at the port 1 and 2, respectively. The mutual coupling between two ports is characterized by S12 and S21. Use the `rfplot` function to visualize the port parameters. Due to the large number of basis functions, simulation of the frequency variation can take several minutes. By default, simulation of the frequency variation is disabled and a precomputed model is used. To enable the simulation of the frequency variation, set `runModel` to true.

```

% Define the frequency range in Hz
f1=linspace(3,6,31)*1e9;
runModel = false;
if runModel
% Compute the S-parameters
s=sparameters(ps,f1);
else
% Load precomputed data if runModel is set to false
load('frequencyVariationData.mat')
end
% Plot the s-parameters
figure;
rfplot(s)

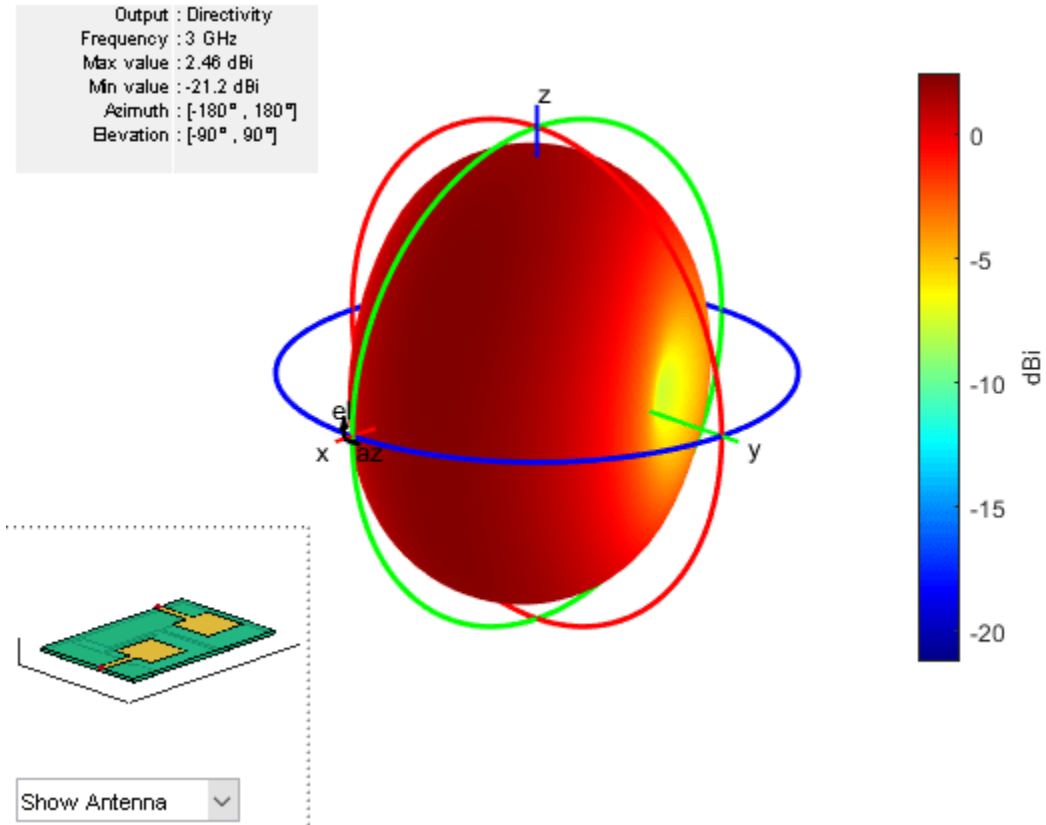
```



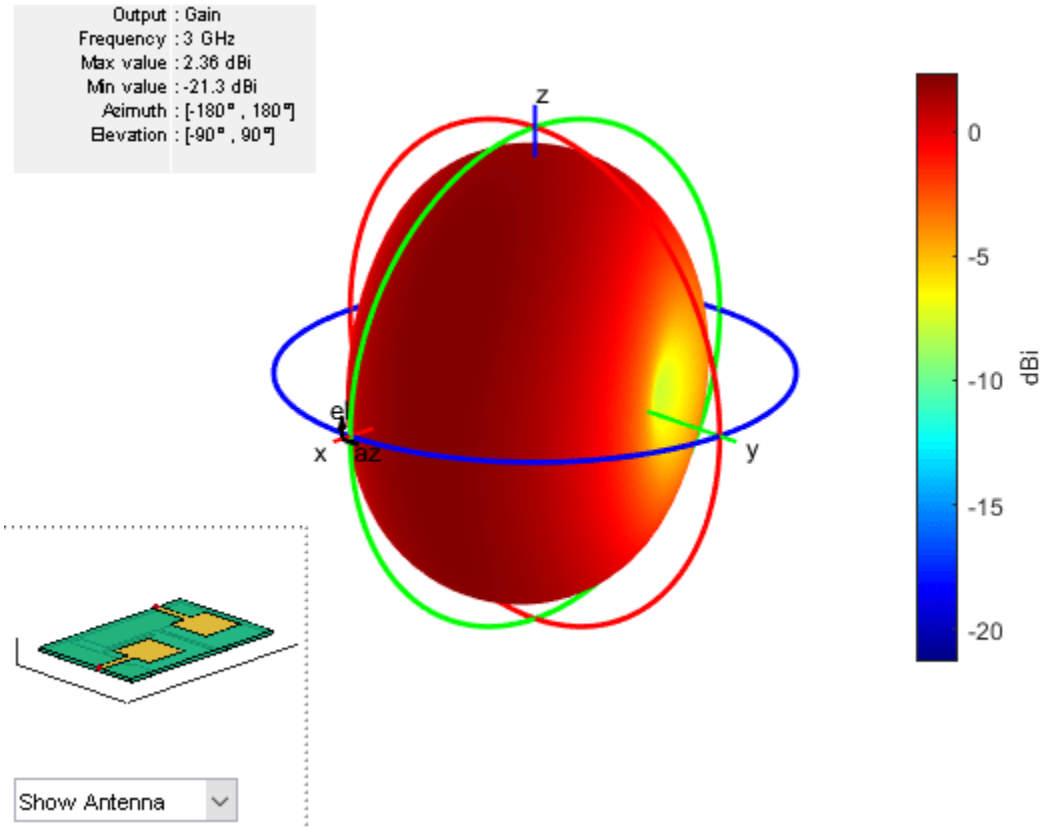
Plot Directivity, Gain, and Realized Gain Patterns with Both Excitations

Use the `pattern` function to visualize different radiation patterns of the array with excitation in both antennas. The input argument 'Type' of `pattern` will determine whether it computes directivity, gain or realized gain pattern. By default, 'Type' is 'directivity' for a lossless antenna and 'gain' for a lossy antenna. The directivity is a ratio of the radiation intensity in a specific angular direction to the average radiated power. It does not depend on material loss or impedance mismatch. The gain is a product of the directivity and radiation efficiency. The radiation efficiency is the ratio of the total radiated power to the total input power going into the antenna's/array's input port(/s). The material losses like finite conduction loss and dielectric loss are accounted in the radiation efficiency and gain parameter. As a part of a complete RF system, antenna/array is commonly connected with an impedance matching network. The impedance mismatch of the antenna or the array with such impedance network leads to finite port mismatch loss which is characterized by the port efficiency. Port efficiency is a ratio of the input power provided by the external excitation port(/s) and the power received at the antenna's/array's input feeding edge(/s). The multiplication of gain with the port efficiency provides the realized gain. The directivity, gain, and realized gain patterns of the MIMO array at 3 GHz are shown below.

```
figure;
pattern(ps,3e9,Type='directivity');
```

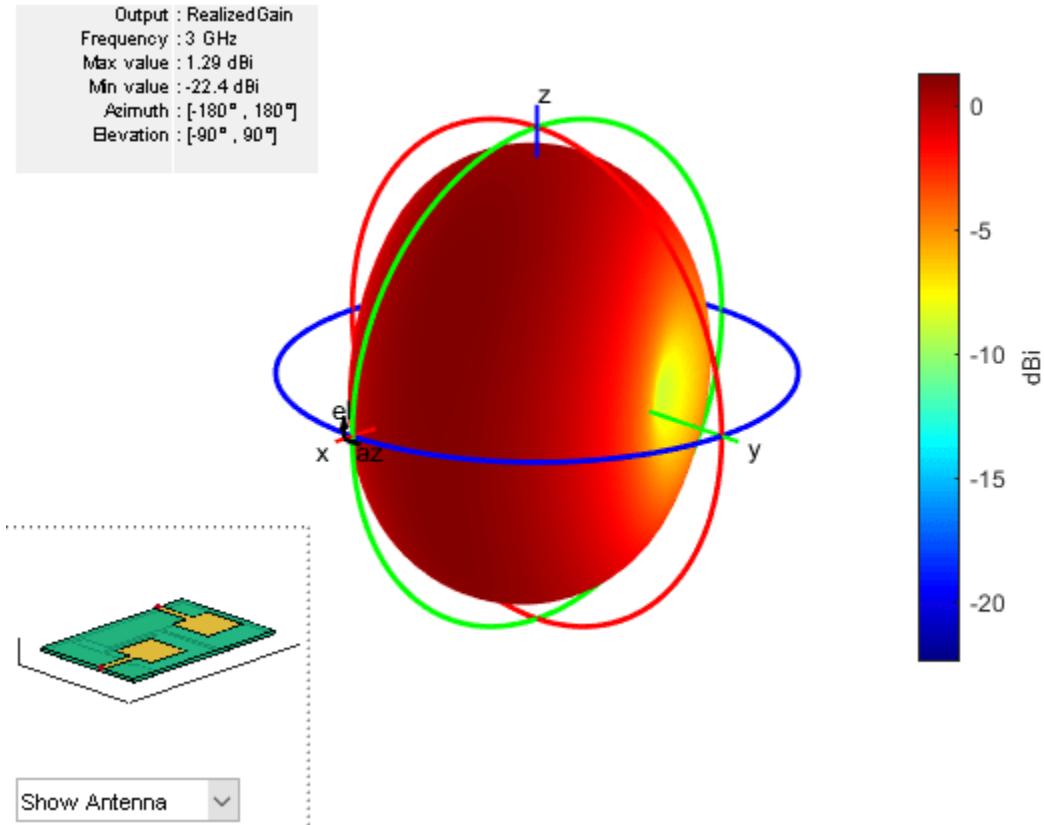



```
figure;  
pattern(ps,3e9,Type='gain');
```



```
figure;  

pattern(ps,3e9,Type='realizedgain');
```



Comparison of the radiation patterns shows that the gain is lower than the directivity as the radiation efficiency is less than 1. Similarly, the realized gain is lower than the gain value due to finite impedance mismatch loss. Use the `efficiency` function to calculate the radiation efficiency of the array.

```
Efficiency=efficiency(ps,3e9);
```

Visualize Directivity, Gain, and Realized Gain Patterns with Single Excitation

Similar to [1], if only one antenna of the array is excited and the other is shorted, a corresponding excitation model is introduced through the `FeedVoltage` and `FeedPhase`. Only the first antenna is assumed to be excited here.

```
ps.FeedVoltage=[1 0];
ps.FeedPhase=[0 0];
```

Compute and plot the directivity, gain, and realized gain values over a broad frequency range.

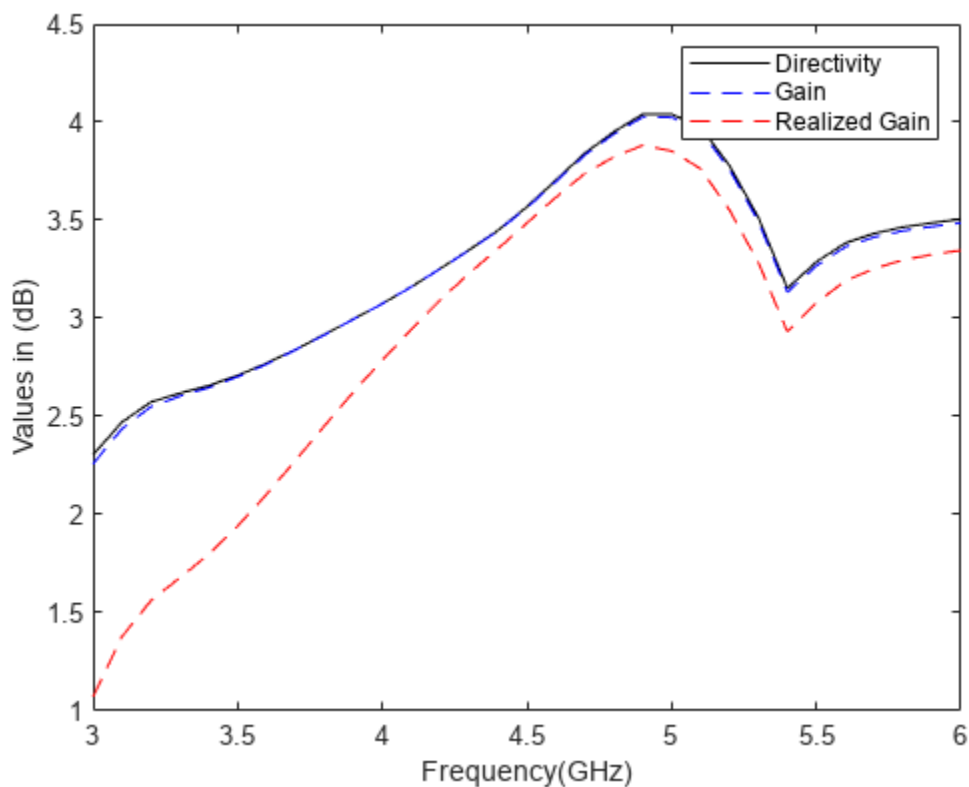
```
if runModel
% Compute directivity, gain, and realized gain
for m=1:length(f1)
patD1=pattern(ps,f1(m),Type='directivity');
patG1=pattern(ps,f1(m),Type='gain');
patRG1=pattern(ps,f1(m),Type='realizedgain');
D1(1,m)=max(max(patD1));
G1(1,m)=max(max(patG1));
RG1(1,m)=max(max(patRG1));
```

```

end
end

% Visualize the directivity, gain, and realized gain patterns
figure;
plot(f1./1e9,D1,'k')
hold on
plot(f1./1e9,G1,'--b')
plot(f1./1e9,RG1,'--r')
xlabel('Frequency(GHz)')
ylabel('Values in (dB)')
legend('Directivity', 'Gain', 'Realized Gain')

```



The gain values are observed to increase with the frequency. This indicates that the array radiation pattern becomes more directional as frequency increases.

Conclusion

This example shows how to efficiently design any advanced complex PCB antenna array using Antenna Toolbox™ and evaluate its impedance and radiation characteristics using Method of Moment (MoM) solver based full-wave simulation.

References

[1] L. Liu, S. W. Cheung, T. I. Yuk and D. Wu. "A Compact Ultrawideband MIMO Antenna," 2013 7th European Conference on Antennas and Propagation (EuCAP), Sweden, 2013.

Direction of Arrival Determination Using Full-Wave Electromagnetic Analysis

This example shows how to determine the Direction of Arrival (DoA) when the transmission source is an antenna located in the far-field region or by assuming that the incident signal behaves like a plane wave incident on the receive array. DoA can be denoted by two angles phi and theta in the spherical co-ordinate system. It is a common practice to use a receive antenna array to scan for any incoming signals and to calculate its angles of arrival with respect to the center of the array. The incoming signal is commonly assumed to arrive from a far-field transmission source. Proper accounting of receive array's electromagnetic behavior is one of the key elements in determining the angles of arrival of the incident signal.

The receiving array for DoA determination can either be a linear or a rectangular array. As the transmit antenna lies in the far-field with respect to the receive array, the mutual interaction between the transmit antenna and the receive array can be neglected. However, there is a finite amount of mutual coupling between the respective elements of the receive array. Mutual coupling is one of the key elements that impact the DoA determination. Mutual coupling depends on various factors such as the type of antenna elements selected used in the receive array, the orientation and separation of the respective array elements, along with the material properties (like conduction loss and dielectric loss) of the antenna elements. For determining the operational boundaries of the the DoA system, it is essential to consider the configuration of the receive array first. You can use the `doa` function from Antenna Toolbox™ to perform full-wave electromagnetic analysis and calculate angles of arrival for a given receive array configuration and type of antenna elements. This aids the designers in selecting the optimal receive array configuration and the optimal antenna elements for a DoA system design. The `doa` function consists of two input arguments, the first argument is either a `conformalArray` object or a `planeWaveExcitation` object, while the second argument is the operating frequency of the system (either a single frequency value or vector of values). The output argument of `doa` provides the calculated phi and theta angles in spherical co-ordinates.

DoA Computation for Conformal Array

This example shows a use case where the input argument is a `conformalArray` object. The `conformalArray` object consists of two elements. The first element is a single feed transmit antenna lying in the far-field with respect to the receive array. The second element is a homogenous uniformly spaced linear or rectangular receive array, with an even number of antenna elements. In this example, the receive array is implemented as a rectangular array.

Assign Operational Frequency

Set the desired frequency (F0) to 2.4 GHz and compute the corresponding wavelength (lambda) using the speed of light and frequency.

```
F0 = 2.4e9;
lambda=physconst('LightSpeed')/F0;
```

Create Transmit Antenna Center Offset Location

Set the location of the transmit antenna in the far-field region by providing arbitrary azimuth and elevation angle values of 0° and 30°, respectively.

```
r=100*lambda;% far-field region
az=0;
```

```

el=30;

tx=r*cosd(az)*sind(el);
ty=r*sind(az)*sind(el);
tz=r*cosd(el);

% Offset of Tx antenna center from the origin
Tx_center_offset=[tx,ty,tz];

```

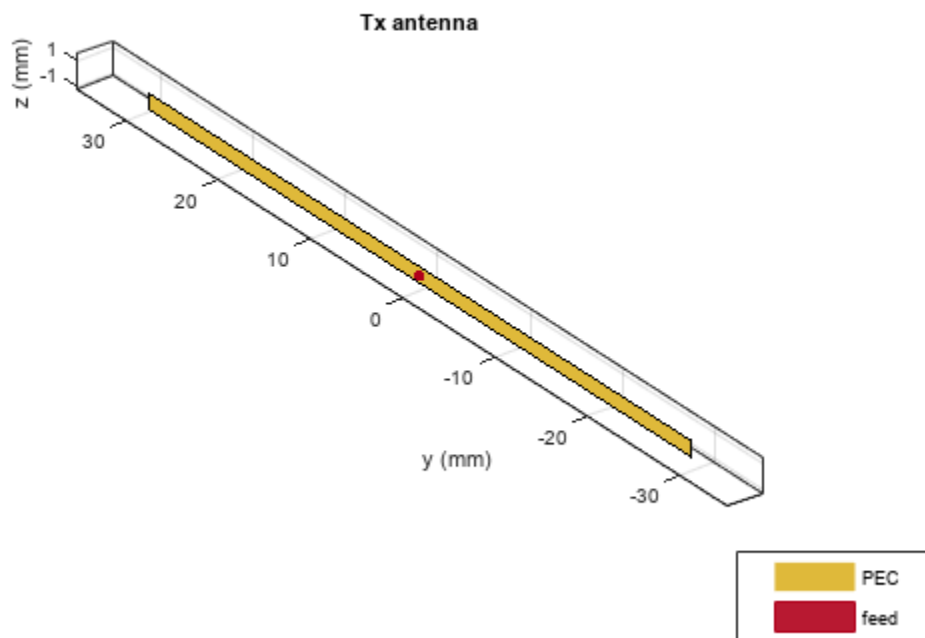
Create and Visualize Transmit Antenna Geometry

Create the geometry of the single feed transmit antenna element and visualize it using the `show` function. In this example, a dipole antenna element is used and placed along the Z-axis. To set the orientation of the dipole antenna as horizontal, set the parameter `Tilt` to 90° . Generate the physical layout of the dipole antenna using the design function from Antenna Toolbox™.

```

ant1=design(dipole,F0);
ant1.Tilt=90; %To make it horizontal
figure;
show(ant1);
title('Tx antenna')

```



Create Receive Array Center Offset Location

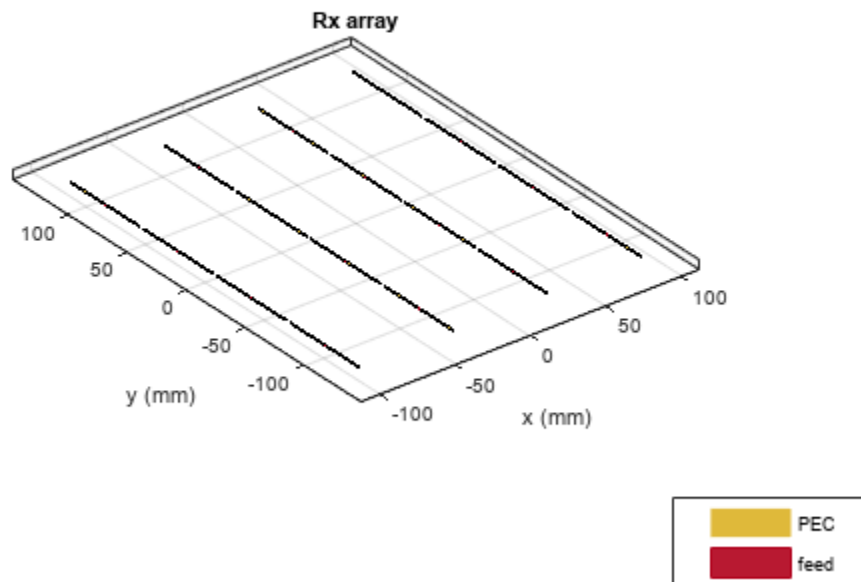
The offset location of the receive array is assumed at the origin of the co-ordinate system.

```
Rx_center_offset=[0,0,0];
```

Create and Visualize 4 by 4 Rectangular Receive Array Geometry

The homogenous uniform receive array is constructed using the catalog element `rectangularArray` from Antenna Toolbox™. The rectangular array consists of dipole antenna elements. Set the size of the array to 4 by 4. Specify even number of elements along each axis of the array. Set the spacing between the two adjacent elements to $\lambda/2$ along each axis. Visualize the receive array using the `show` Function.

```
RxArray2=rectangularArray('Size',[4 4]);
RxArray2.Element = design(dipole,F0);
RxArray2.RowSpacing=0.5*lambda;
RxArray2.ColumnSpacing=0.5*lambda;
ant2=RxArray2;
ant2.Element.Tilt=90; %To make it horizontal
figure;
show(ant2)
title('Rx array')
```



Create Conformal Array Geometry

Create a `conformalArray` object and assign the transmit and receive elements to it. The transmit element is the first element and the receive element is the second element. Here, the transmit element is a single dipole antenna and the receive element is a uniform homogenous 4 by 4 rectangular array of dipole antennas.

```
c=conformalArray;
c.ElementPosition=[Tx_center_offset;Rx_center_offset];
```

```
%1st element of conformal array is Tx  
%2nd element of conformal array is Rx  
c.Element={ant1 ant2};
```

Call doa for Conformal Array Object

Call the `doa` function on the `conformalArray` object.

```
[phiArrival1, thetaArrival1]= doa(c, F0)  
  
phiArrival1 = -2.9205e-05  
  
thetaArrival1 = 29.1207
```

The results show a slight difference in the calculated angles `phiArrival1` and `thetaArrival1` than the previously specified location of the transmitter. You can do this analysis by using alternate receive array configurations and antenna elements to select the optimum configuration that fits the accuracy requirements for your DoA system.

DoA Computation for Incident Plane Wave

This example shows the DoA computation for an incident plane wave signal.

Create Single Element of Homogenous Receive Array

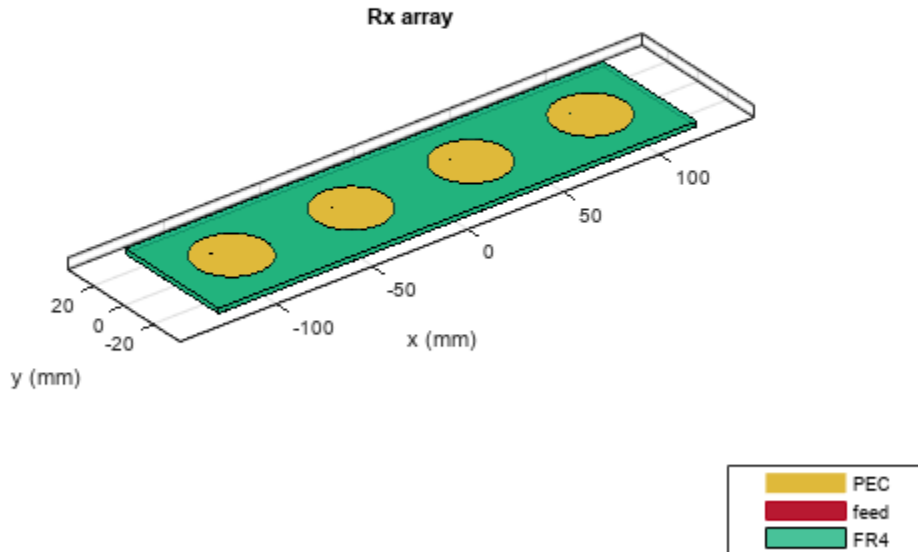
Create a circular microstrip patch antenna operating at 2.4 GHz as the receive array element by using the `design` function from Antenna Toolbox™. Assign the ground plane dimensions to it. Add a lossy FR4 dielectric substrate to it. Adjust the thickness of the dielectric substrate and the radius of the circular microstrip patch to achieve a resonant frequency of 2.4 GHz.

```
RxElement = design(patchMicrostripCircular, F0);  
RxElement.GroundPlaneLength = lambda/2; % in m  
RxElement.GroundPlaneWidth = lambda/2; % in m  
RxElement.Radius = 0.018;% in m  
prx_con.Height = 0.0016;% in m  
RxElement.FeedOffset = [ -0.006 0.006];% in m  
RxElement.Substrate = dielectric('FR4'); % lossy substrate
```

Create and Visualize Linear Array as Receive Array

Construct a homogenous uniform receive array using the `linearArray` element in the array catalog. Set the total number of array elements to 4. Set the spacing between adjacent antenna elements to $\lambda/2$. View the array geometry using the `show` function.

```
RxArray1 = linearArray;  
RxArray1.Element = RxElement;  
RxArray1.NumElements = 4;  
RxArray1.ElementSpacing = lambda/2;  
ant2=RxArray1;  
figure;  
show(ant2)  
title('Rx array')
```

Create Direction and Polarization Vectors for Plane Wave Excitation

Set the position angles of the incident plane wave to 10° azimuth and 30° elevation. Using these angles, determine the direction and polarization vectors of the incident plane wave.

```
az_ref=10;
el_ref=30;
dir=[sind(el_ref)*cosd(az_ref) sind(el_ref)*sind(az_ref) -cosd(el_ref)];
pol=[cosd(el_ref)*cosd(az_ref) cosd(el_ref)*sind(az_ref) sind(el_ref)];
```

Create the planeWaveExcitation Object

Create the planeWaveExcitation Object with the previously created linear array as its receive element and direction and polarization vectors as calculated above.

```
obj1 = planeWaveExcitation('Element', ant2, 'Direction', ...
    dir, 'Polarization', pol);
```

Call doa for the planeWaveExcitation Object

Call the doa function on the planeWaveExcitation object to compute the angles of arrival of the incident plane wave.

```
[phiArrival2, thetaArrival2]= doa(obj1, F0)
```

```
phiArrival2 = 10
```

```
thetaArrival2 = 31.2847
```

Like the computation in the previous example, the computed `phiArrival2` and `thetaArrival2` values in this example slightly differ from the specified azimuth and elevation angles. These values can be used to study the trade-offs between different array elements and configurations and to select the optimum configuration for your use case. This analysis can be used either for calibrating the DoA system or as an input to signal processing-based DoA algorithms. The finite difference between the actual and predicted angles depends on the electromagnetic properties of the receiving array. Antenna Toolbox™ uses full-wave electromagnetic solution of Maxwell's equations to model the behavior of antenna systems and as shown, can be used for predicting direction of arrival using the `doa` function.

Conclusion

With Antenna Toolbox™, you can perform a similar analysis with antenna elements in the antenna catalog or custom antenna elements constructed from the shape primitives, `pcbstack` or imported via the `stl` function. This workflow facilitates the study of DoA system configurations from strictly a full-wave electromagnetic analysis viewpoint. This analysis is useful to initially assess the types of antenna elements and antenna array configuration that are used to achieve a desired overall system performance for a DoA system.

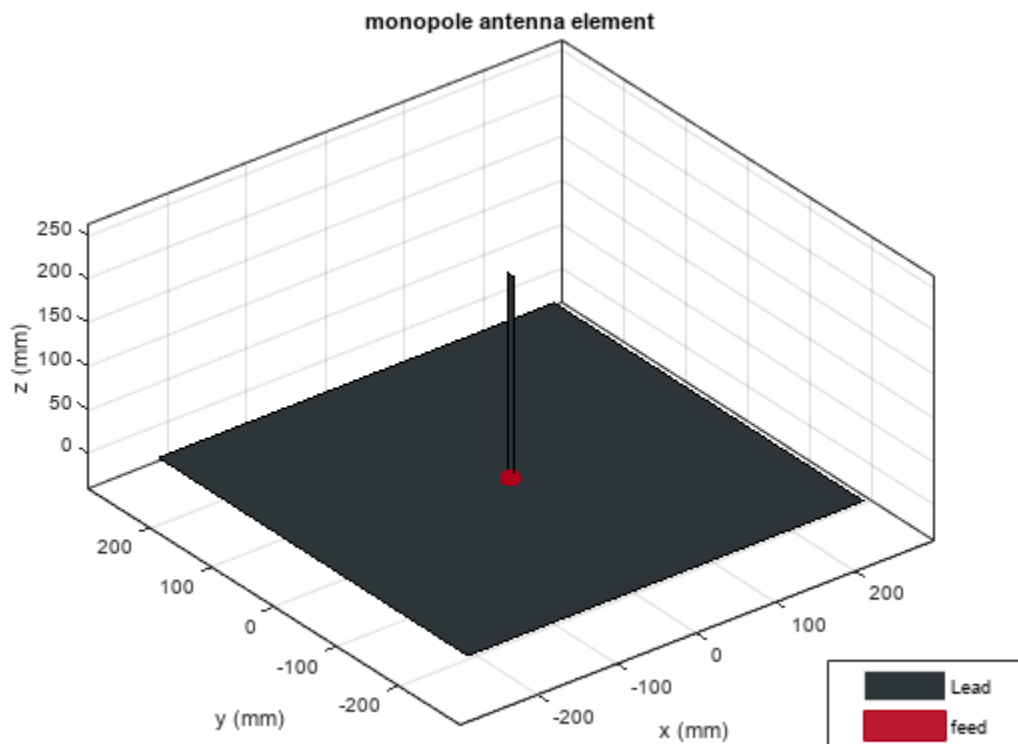
Field Analysis of Monopole Antenna

This example shows how to conduct the field analysis of monopole antenna using `pattern` function.

Construct and Visualize Antenna

Use the design function to create a monopole antenna at 300 MHz. Use lead as the conductor material for the antenna.

```
f=300e6;
ant=design(monopole(Conductor='Lead'),f);
figure;
show(ant)
```

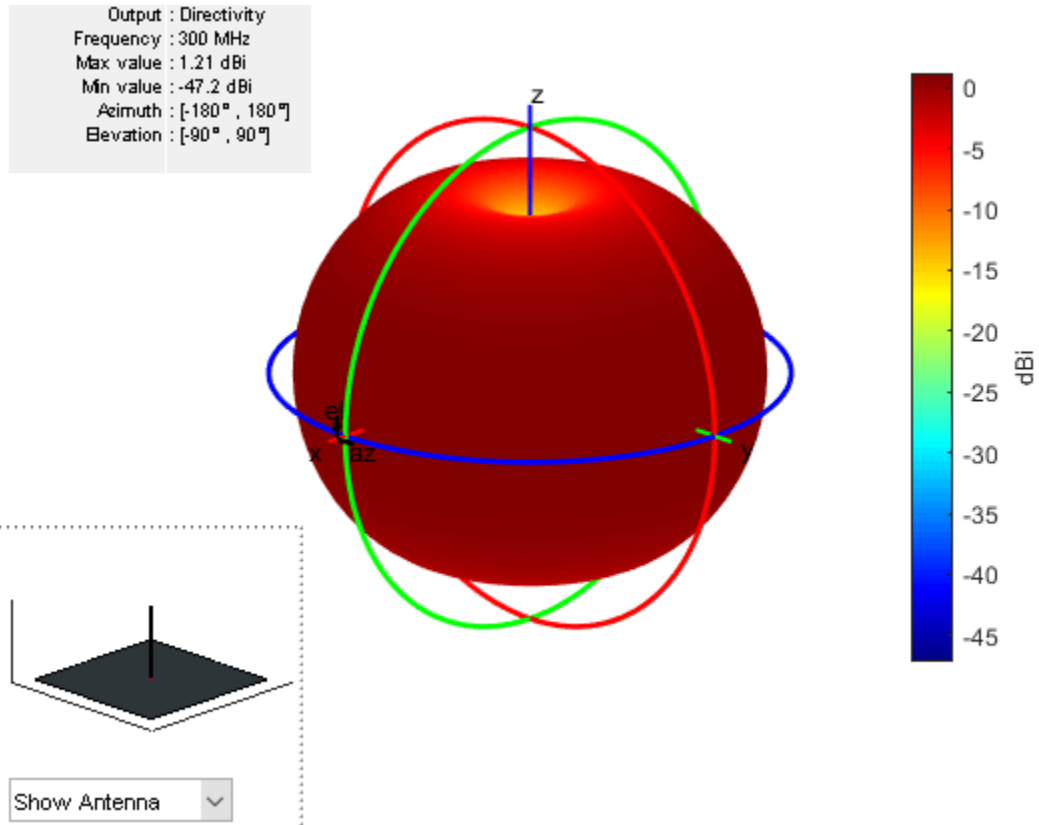


Compute Directivity, Gain, and Realized Gain

Use the Type name-value pair in the pattern function to visualize the directivity, gain, and realized gain of the monopole antenna.

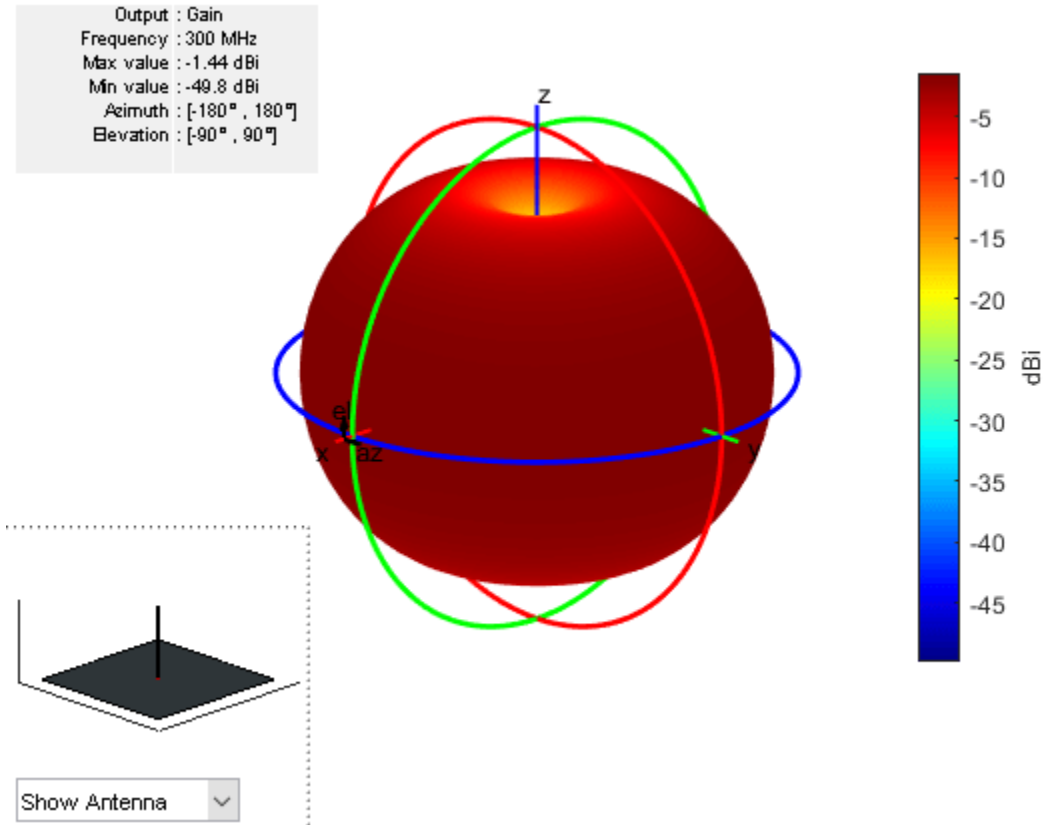
- `directivity`

```
figure;
pattern(ant,f,Type='directivity')
```



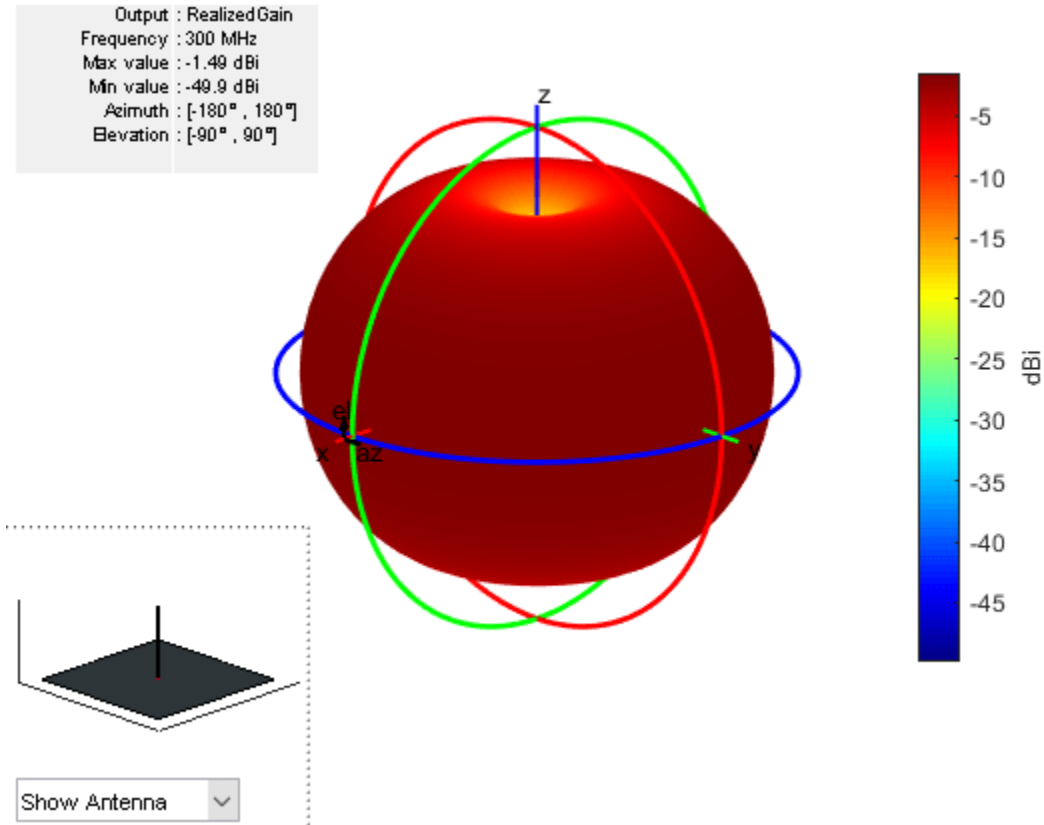
- gain

```
figure;  
pattern(ant, f, Type='gain')
```



- realized gain

```
figure;  
pattern(ant, f, Type='realizedgain')
```



Use the `max` function in dB scale to compute the maximum value of the corresponding 3D patterns.

```
Dmax=max(max(pattern(ant,f,Type='directivity')))
```

```
Dmax = 1.2060
```

```
Gmax=max(max(pattern(ant,f,Type='gain')))
```

```
Gmax = -1.4405
```

```
RGmax=max(max(pattern(ant,f,Type='realizedgain')));
```

Calculate the radiation efficiency using the `efficiency` function and the port reflection loss using the `S-parameters` function.

```
DiffGminusD=Gmax-Dmax
```

```
DiffGminusD = -2.6465
```

```
EfficiencyValue=10*log10(efficiency(ant,f))
```

```
EfficiencyValue = -2.6465
```

```
s=sparameters(ant,f);
```

```
s11=s.Parameters;
```

```
ReflectionLoss=10*log10(1-(abs(s11))^2)
```

```
ReflectionLoss = -0.0501
```

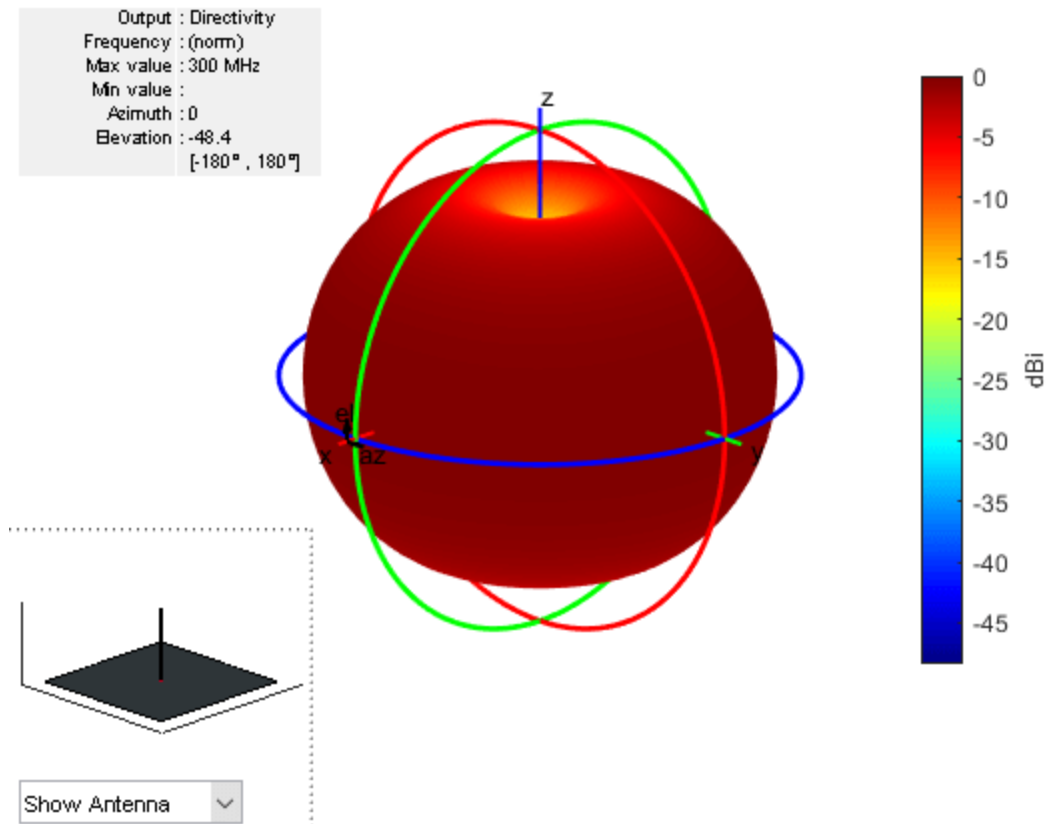
```
DiffRGminusG=RGmax-Gmax
```

```
DiffRGminusG = -0.0501
```

You see that the efficiency value is the difference between the maximum gain and maximum directivity. The difference between the realized gain and gain values is equal to the port efficiency.

To visualize the maximum amplitude normalized pattern, set `Normalize` name-value pair in the `pattern` function to `true`.

```
figure;
pattern(ant,f,Type='directivity',Normalize=true)
```

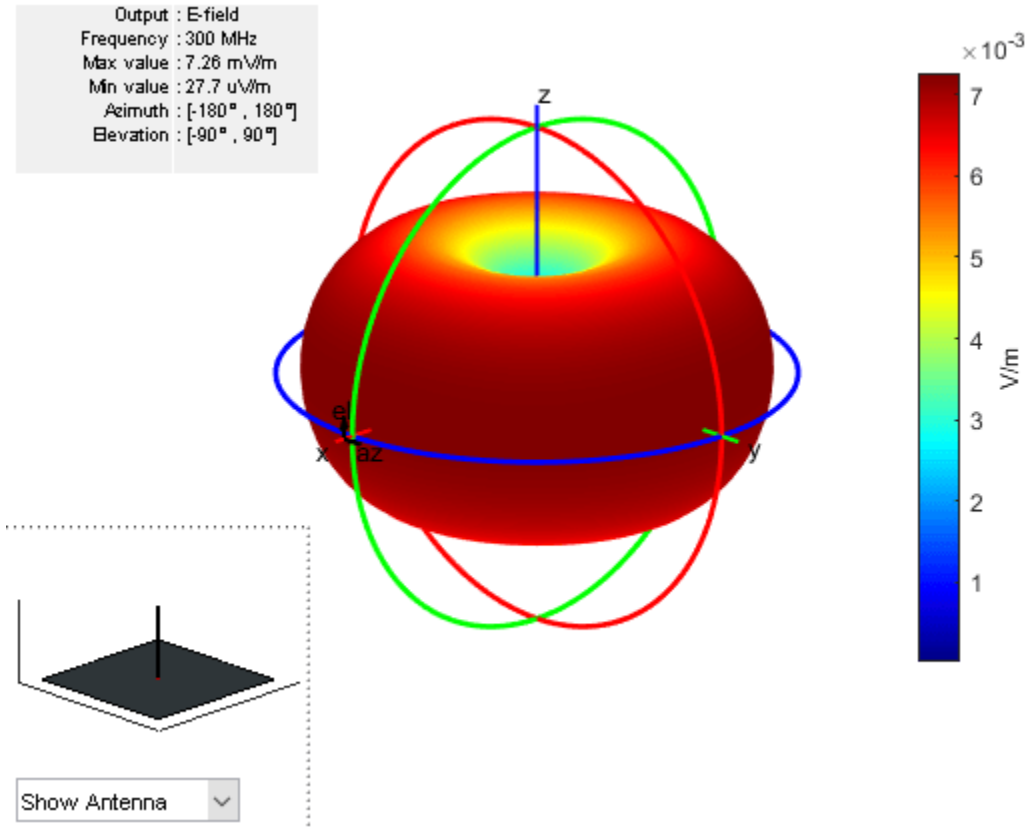


You can see in the figure that with `Normalize` name-value pair set to `true` the maximum gain value is 0 dB.

Calculate Electric Field Magnitude and Norm

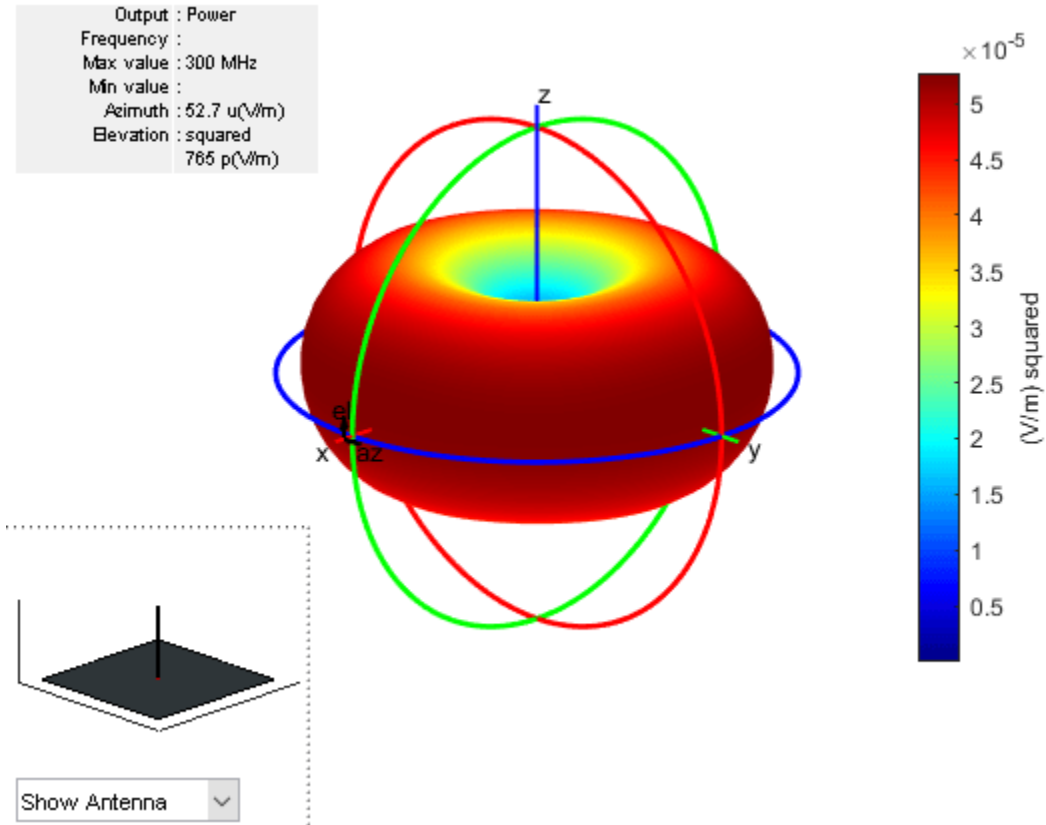
To visualize the electric field magnitude set the `Type` name-value pair to `efield`.

```
figure;
pattern(ant,f,Type='efield')
```



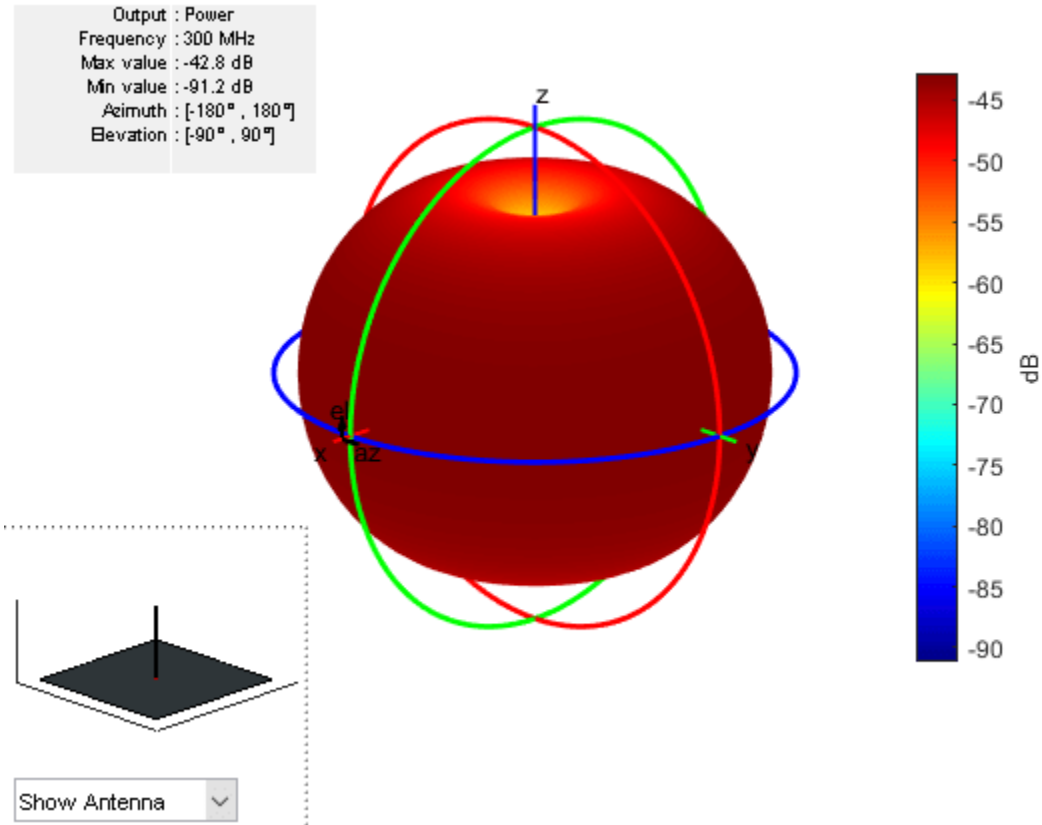
To visualize the norm of the electric field in absolute scale set the Type name-value pair to power.

```
figure;  
pattern(ant, f, Type='power')
```

To visualize the norm of the electric field in dB scale set the Type name-value pair to `powerdB`.

```
figure;
pattern(ant,f,Type='powerdB')
```



The unit of power when Type is set to power is Watt and when set to powerdB, it is dBW. The Type set to power provides the norm of the electric field at each observation location over the far field radiation sphere. The radiated power of the antenna or the array is different from the power obtained when the Type is set to power . The second one is $|E_s^{\text{abs}}(R, \theta, \phi)|^2$ in volt per meter squared. However, the radiated power of the antenna or array is

$$P_{\text{rad}} = \int_{\phi=0}^{\pi} \int_{\theta=0}^{2\pi} U(\theta, \phi) \sin\theta d\theta d\phi = \int_{\phi=0}^{\pi} \int_{\theta=0}^{2\pi} \frac{R^2}{2\eta_0} |E_s^{\text{abs}}(R, \theta, \phi)|^2 \sin\theta d\theta d\phi$$

Compute Electric Field Phase

The electric field computed at each observation point over the radiation sphere is a 3-by-1 complex vector. In Cartesian coordinates this vector is written as

$$E_s = \begin{bmatrix} E_x \\ E_y \\ E_z \end{bmatrix}$$

In the far-field region, the radial component of the electric field becomes zero. The equivalent component of the electric field in the spherical coordinate is:

$$E_{\theta} = E_x * \cos\theta * \cos\phi + E_y * \cos\theta * \sin\phi - E_z * \sin\theta$$

$$E_\phi = -E_x \sin\phi + E_y \cos\phi$$

For a spherical coordinate system, you can use azimuth and elevation angles instead of theta and phi.

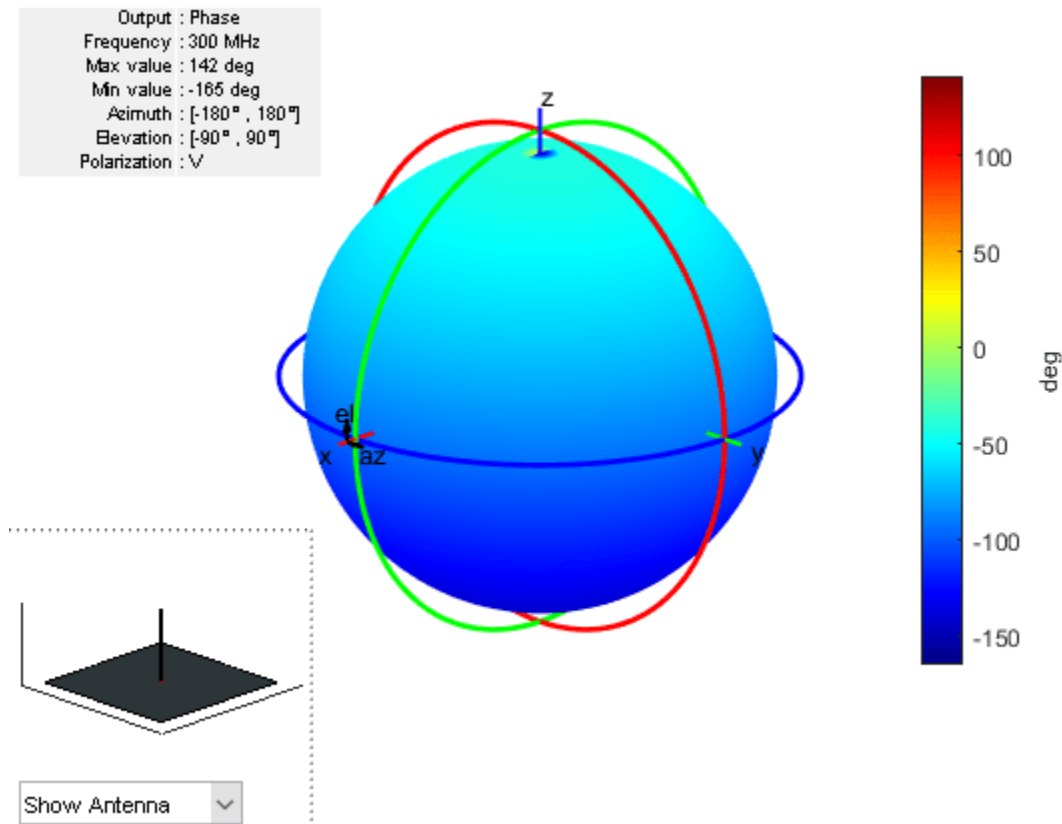
azimuth = θ

elevation = $90 - \theta$

To compute the phase, you need to specify the polarization in the pattern.

Compute the phase of the electric field with vertical polarization.

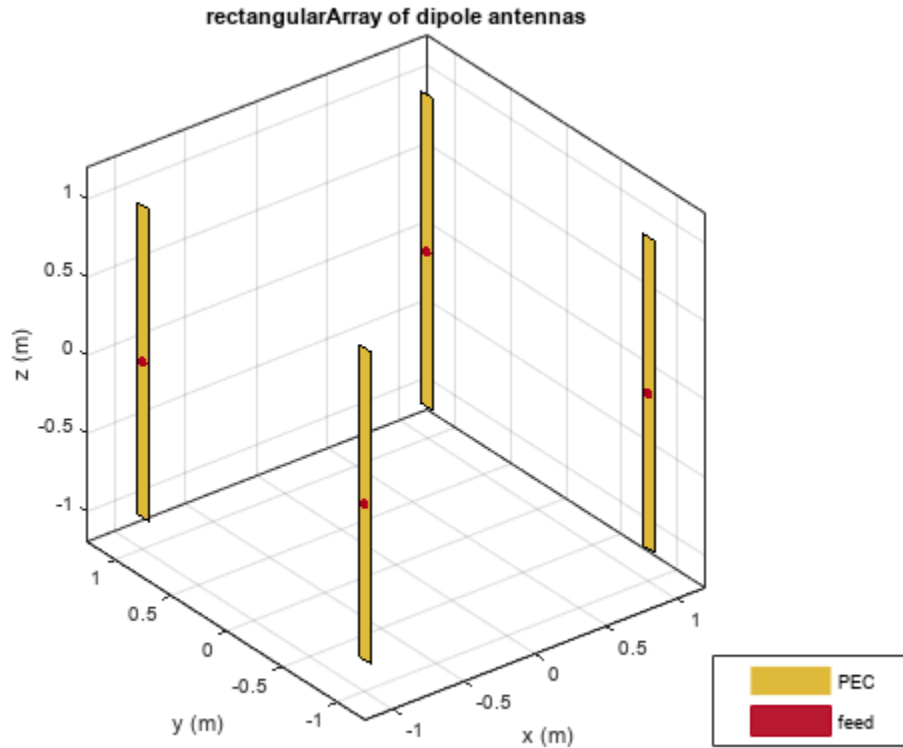
```
figure;
pattern(ant,f,Type='phase',Polarization='V')
```



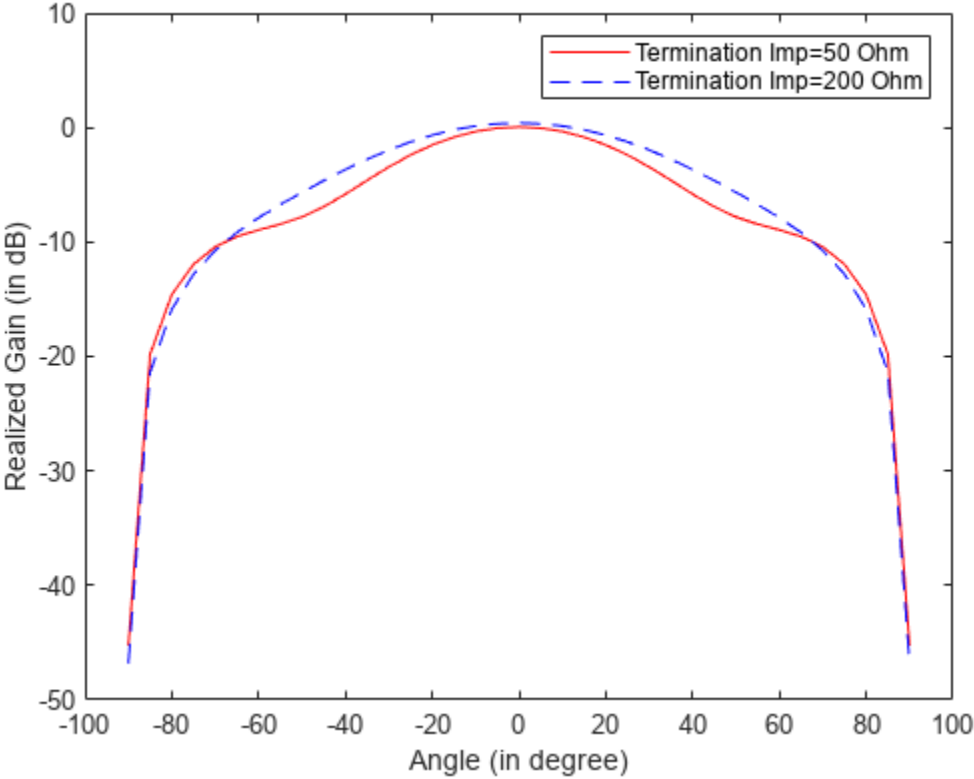
Embedded Pattern in Antenna Array

For antenna arrays, the embedded pattern is the pattern obtained by exciting one antenna port and terminating the other antenna ports to a certain termination impedance. To plot the embedded element pattern of an array, set the element index `ElementNumber` property in `pattern` function. While computing the embedded pattern, you can terminate the antenna's excitation port to a customized impedance using the `Termination` property in `pattern` function. The termination impedance will influence the port efficiency ($PE = 1 - |S_{11}|^2$ for a single port antenna) and the realized gain depends on the termination impedance value. To have a comparative overview, consider a default four-element rectangular array with two different termination impedance values.

```
l=rectangularArray;
figure;
show(l)
```



```
pat1=pattern(l,70e6,0,-90:5:90,ElementNumber=1, Termination=50,...
    Type='realizedgain',CoordinateSystem='rectangular');
pat2= pattern(l,70e6,0,-90:5:90,ElementNumber=1, Termination=200,...
    Type='realizedgain',CoordinateSystem='rectangular');
figure;
plot(-90:5:90,pat1,'r')
hold on
plot(-90:5:90,pat2,'--b')
xlabel('Angle (in degree)')
ylabel('Realized Gain (in dB)')
legend('Termination Imp=50 Ohm','Termination Imp=200 Ohm')
```



Design and Analysis of a Diamond-Shaped Antenna for Ultra-Wideband Applications

This example shows how to create the diamond-shaped ultra-wideband (UWB) antenna described in [1] with a modified ground plane. UWB antennas are in high demand for various applications, such as wireless communications, medical imaging, radar, and indoor positioning. This demand is due to their ability to enable a high data transmission rate and low power consumption.

The planar diamond-shaped monopole antenna structure described in [1] achieves an ultra-wide bandwidth by using its features, such as its small size, ease of fabrication, low power consumption, and easy integration with the RF system. These features make the antenna a suitable candidate for use in compact devices.

In this example, you design the proposed antenna on a 1.6 mm FR4 substrate with a dielectric constant of 4.4. This antenna consists of a diamond-shaped radiating patch, a modified feedline, and a semicircular ground plane. You then modify the conventional rectangular monopole antenna with beveled edges to form a diamond-shaped radiating patch.

Create Antenna Model

Define the length and width of the ground plane.

```
gndL = 32e-3;
gndW = 40e-3;
% Define the board thickness.
h = 1.6e-3;
R_gnd = 15.6e-3;
% Define the substrate.
sub = dielectric("Name",{ 'FR4' }, "EpsilonR", 4.4, 'Thickness', h);
% Specify the widths of the traces.
Wf = 3.1e-3;
Wf1 = 1.2e-3;
W3 = 13e-3;
W2 = 29e-3;
W1 = 22e-3;
% Specify the lengths of the traces.
Lf = 10e-3;
Lf1 = 6e-3;
L3 = 2e-3;
L2 = 11e-3;
L1 = 5e-3;
```

Create the top-layer diamond-shaped antenna by passing the parameters and vertices to the `antenna.Polygon` object.

```
xver = round([-Wf/2 -Wf/2 -Wf1/2 -W3/2 -W2/2 -W1/2 W1/2 W2/2 W3/2 Wf1/2 Wf/2 Wf/2],12);
yver = round([-gndW/2 -gndW/2+Lf -gndW/2+Lf+Lf1 -gndW/2+Lf+Lf1+L3 -gndW/2+Lf+Lf1+L3+L2 ...
-gndW/2+Lf+Lf1+L3+L2+L1 -gndW/2+Lf+Lf1+L3+L2+L1 -gndW/2+Lf+Lf1+L3+L2 ...
-gndW/2+Lf+Lf1+L3 -gndW/2+Lf+Lf1 -gndW/2+Lf -gndW/2],12);
zver = zeros(1,size(yver,2));
ver = [xver',yver',zver'];
UWB_ant = antenna.Polygon('Vertices',ver);
```

Create the semicircular ground layer.

```

cir = antenna.Circle('Radius',R_gnd,'Center',[0,-gndW/2]);
rec = antenna.Rectangle('Length',gndL+2e-3,'Width',R_gnd,'Center',[0,-gndW/2-R_gnd/2]);
rec1 = antenna.Rectangle('Length',0.5*R_gnd,'Width',0.5*R_gnd,'Center',[-gndL/2-(0.5*R_gnd)/2,-gndW/2]);
rec2 = antenna.Rectangle('Length',0.5*R_gnd,'Width',0.5*R_gnd,'Center',[gndL/2+(0.5*R_gnd)/2,-gndW/2]);
gnd = cir - (rec+rec1+rec2);

```

Generate the UWB antenna model by using the pcbStack object with the assigned ground layer.

```

ant = pcbStack;
ant.BoardThickness = h;
ant.BoardShape = antenna.Rectangle('Length',gndL,'Width',gndW);
ant.Layers = {UWB_ant,sub,gnd};
ant.FeedLocations = [0,-gndW/2,1,3];
ant.FeedDiameter = Wf/2;

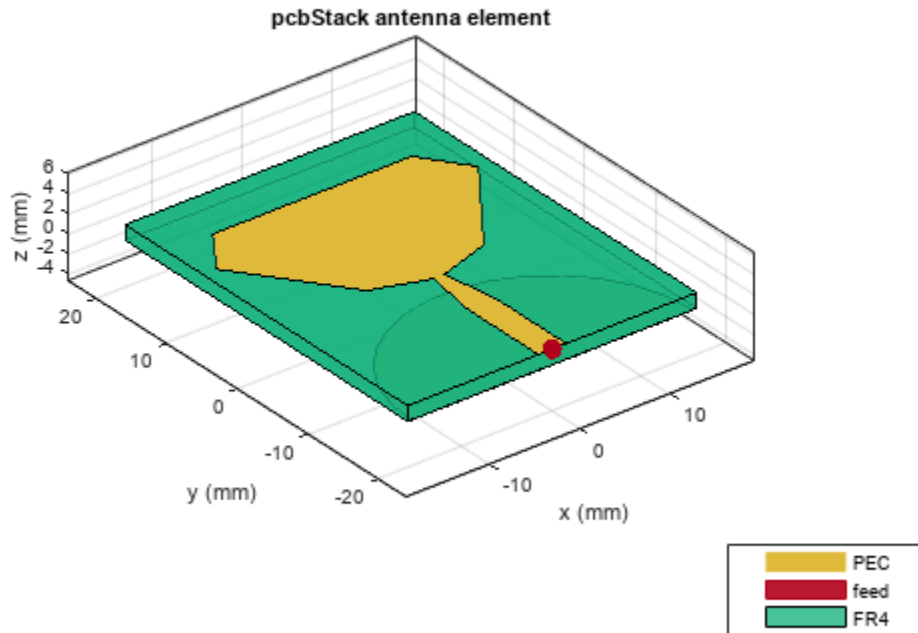
```

Show the antenna.

```

figure;
show(ant);

```



Analyze Return Loss and Gain of Antenna

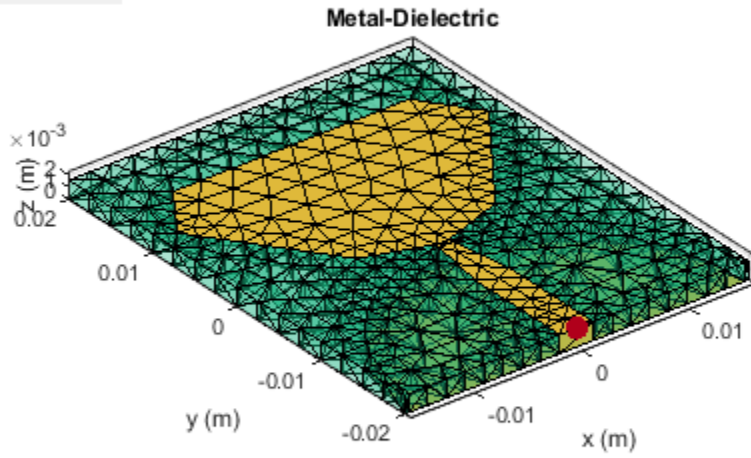
Mesh the structure using manual mesh mode with a maximum edge length of 0.003 m.

```

figure;
mesh(ant,'MaxEdgeLength',0.003);

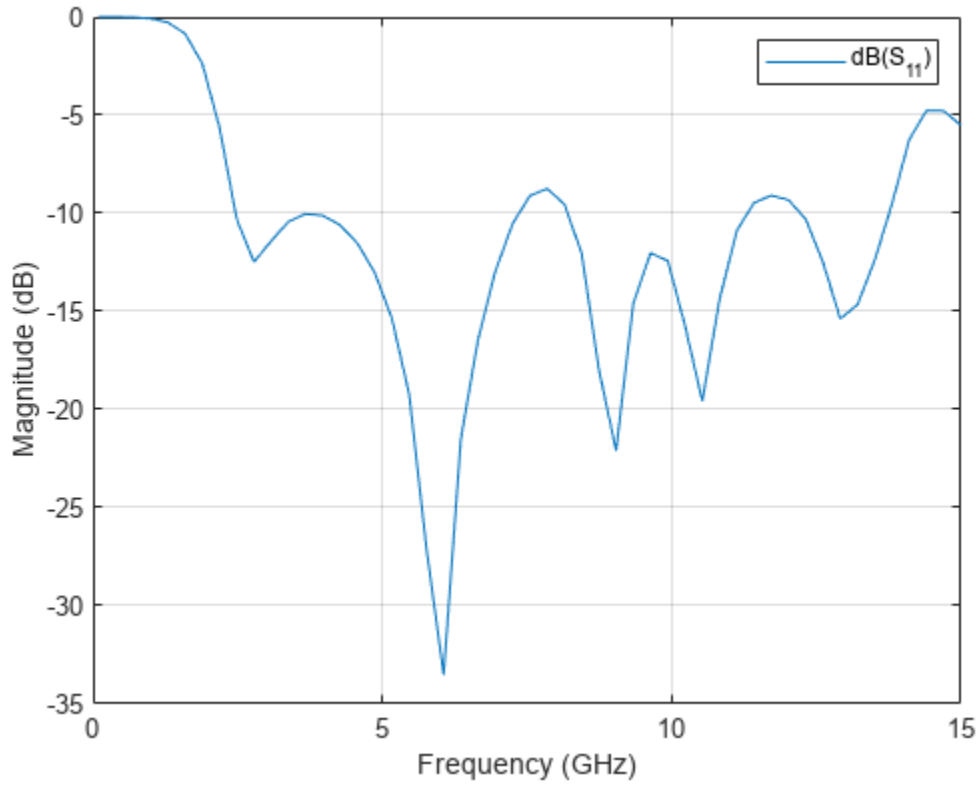
```

NumTriangles: 403
 NumTetrahedra: 1899
 NumBasis:
 MaxEdgeLength: 0.003
 MeshMode: manual



Use the `sparameters` function to analyze the return loss of the structure in the frequency range of 0.1 GHz to 15 GHz.

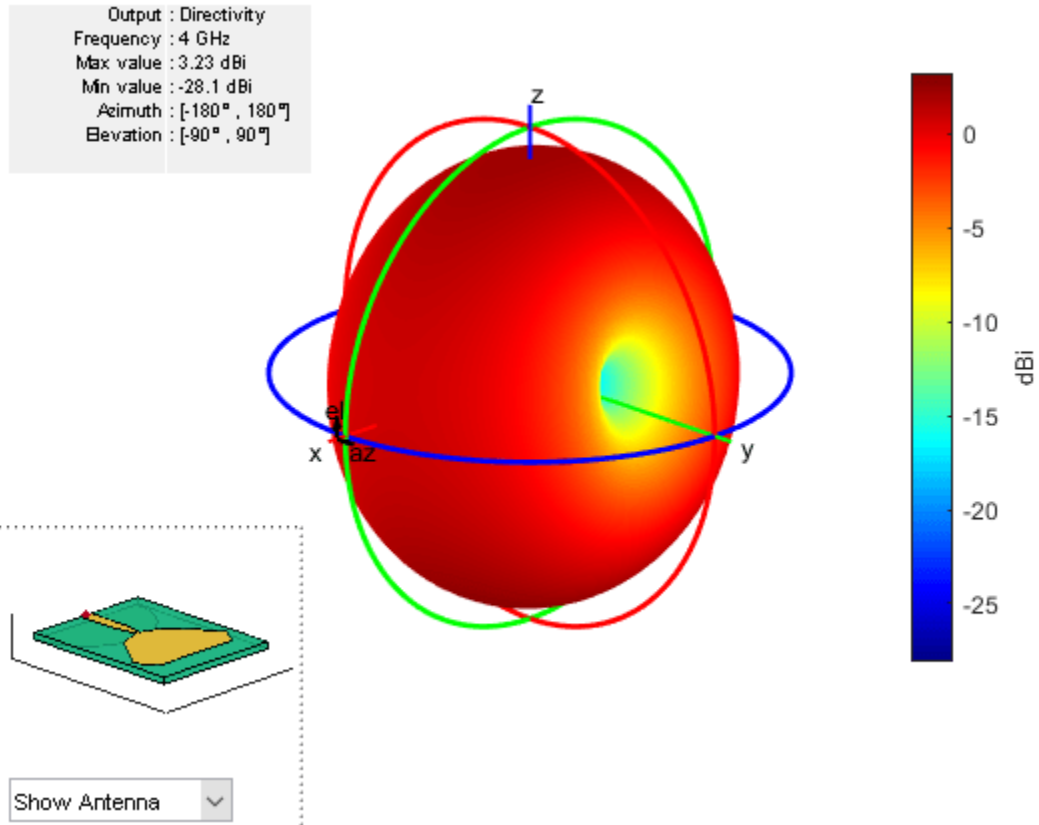
```
spar = sparameters(ant,linspace(0.1e9,15e9,51));
figure;
rfplot(spar);
```

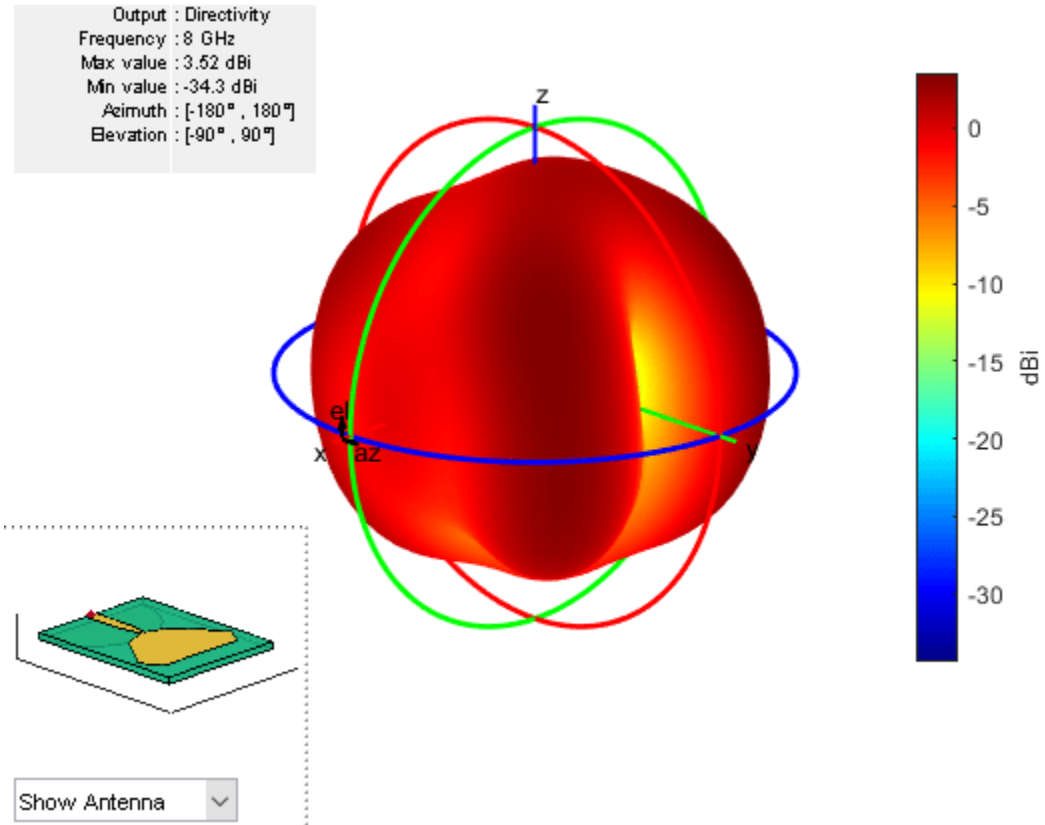
The UWB diamond-shaped antenna exhibits a passband behavior in the frequency range of 2.5 GHz to 15 GHz with a return loss greater than -10 dB.

Analyze the radiation pattern of the wideband antenna at 4 GHz, 8 GHz, and 15 GHz within the frequency spectrum.

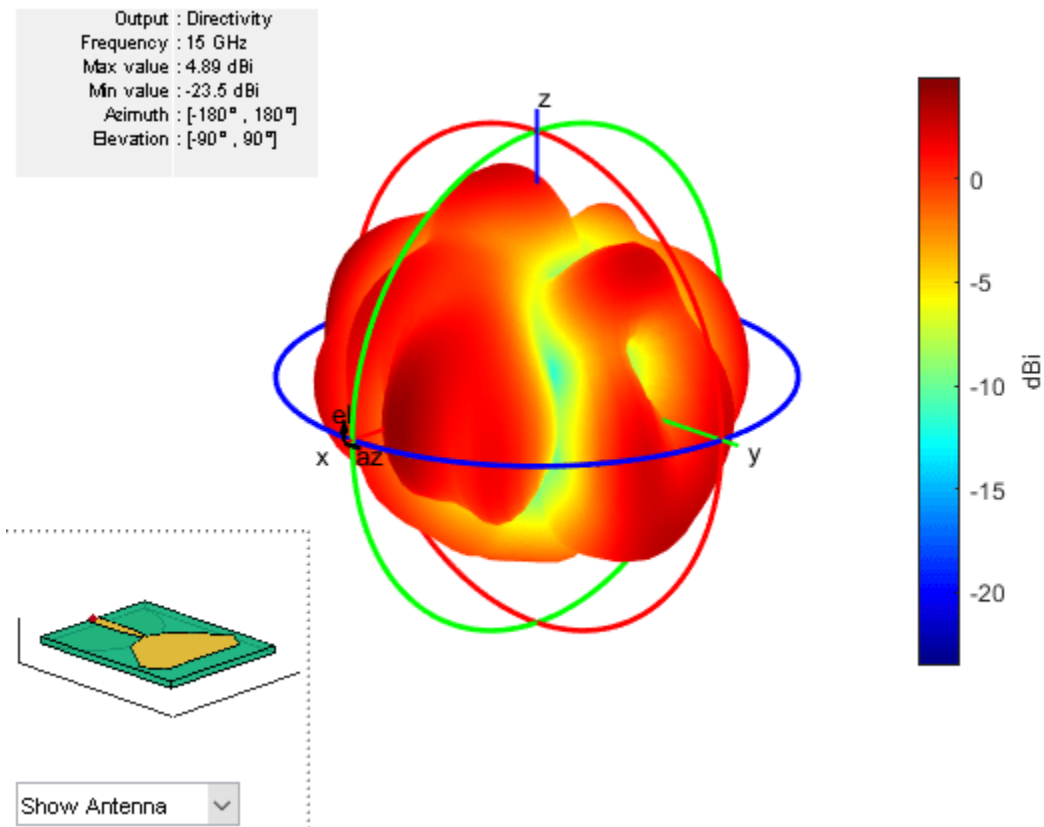
```
figure;  
pattern(ant,4e9);
```



```
figure;  
pattern(ant,8e9);
```



```
figure;  
pattern(ant,15e9);
```



The proposed antenna exhibits stable radiation characteristics with a gain of 3.3 dBi, 3.26 dBi, and 3.99 dBi at 4, 8, and 15 GHz, respectively.

Conclusion

The diamond-shaped antenna achieves the enhanced ultra-wide bandwidth, in which the UWB frequency spectrum covers the range from 3.1 GHz to 10.6 GHz, with stable radiation characteristics. Because the proposed antenna has good UWB characteristics and a compact size, it is suitable for wireless communications systems.

Reference

[1] Singhal, Sarthak, Tushar Goel, and Amit Kumar Singh. "Novel diamond shaped UWB monopole antenna." 2013 Annual IEEE India Conference (INDICON) (January 2014): 1-6. <https://doi.org/10.1109/INDCON.2013.6726048>.

Board Thickness versus Dielectric Thickness in PCB

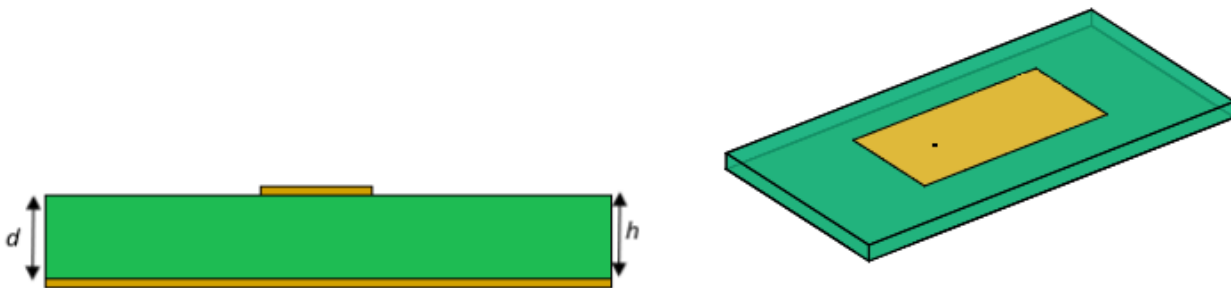
This example shows how to define the board thickness of a PCB with respect to the dielectric thickness of `pcbStack` object for different use-cases. This example has three sections for three different use cases as below:

- 1 Single-layered dielectric PCB
- 2 Multi-layered dielectric PCB
- 3 Importing the PCB design using a `pcbReader` object.

The value of the `BoardThickness` property of the `pcbStack` object is the sum of thicknesses of all the dielectric layers that lie below the top metal layer. Dielectric layers above the top metal layer are considered as coating and are not included in the `BoardThickness` calculations. **You must define the `BoardThickness` before defining the Layers.**

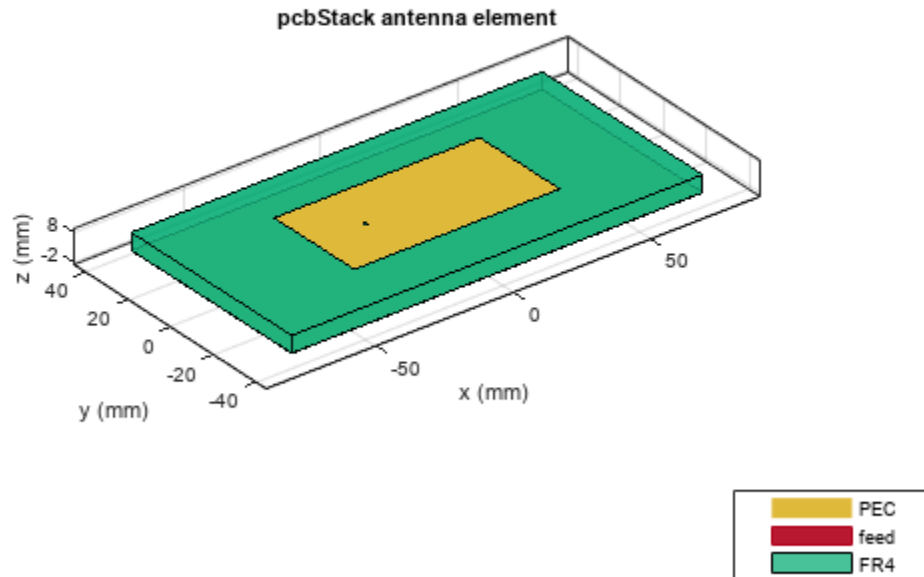
Single-Layered dielectric `pcbStack`

The PCB in this use case has a single dielectric layer sandwiched between the top and bottom metal layers with `BoardThickness` defined by h and the dielectric Thickness defined by d .



When the `pcbStack` has a single dielectric layer, the `BoardThickness` (h) must be equal to the dielectric Thickness (d). Otherwise, d is updated to match h .

```
p = pcbStack;
sub = dielectric('Fr4');
TopMetal = p.Layers{1};
BtmMetal = p.Layers{2};
p.BoardThickness = sub.Thickness;
p.Layers = {TopMetal,sub,BtmMetal};
p.FeedLocations = [-0.0187 0 1 3];
figure;
show(p)
```



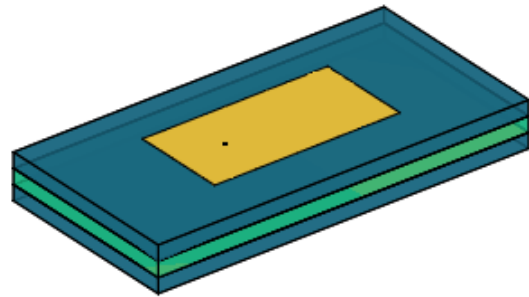
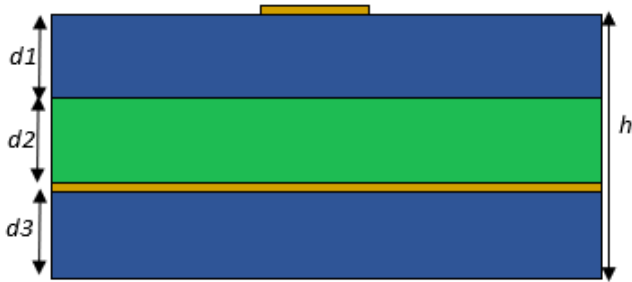
Multi-Layered dielectric pcbStack

There are two use cases for the PCBs with multiple dielectric layers; PCBs with dielectric layers present below the top metal layer or only above the top metal layer. Sections below describe how to define the `BoardThickness` for these two use cases.

Dielectric layers present below the top metal layer

The `BoardThickness` of a PCB with multiple dielectric layers is the sum of thicknesses of all the dielectric layers present below the top metal layer. The dielectric layers above the top metal layer are considered as a coating and are not included in the board thickness calculations.

In the below use case, there are two dielectric layers between the top and bottom metal layers and one dielectric layer below the bottom metal layer as shown in the figure. The `BoardThickness` (h) for this PCB is the sum of the thicknesses ($d1+d2+d3$) of all the dielectric layers below the top metal layer.

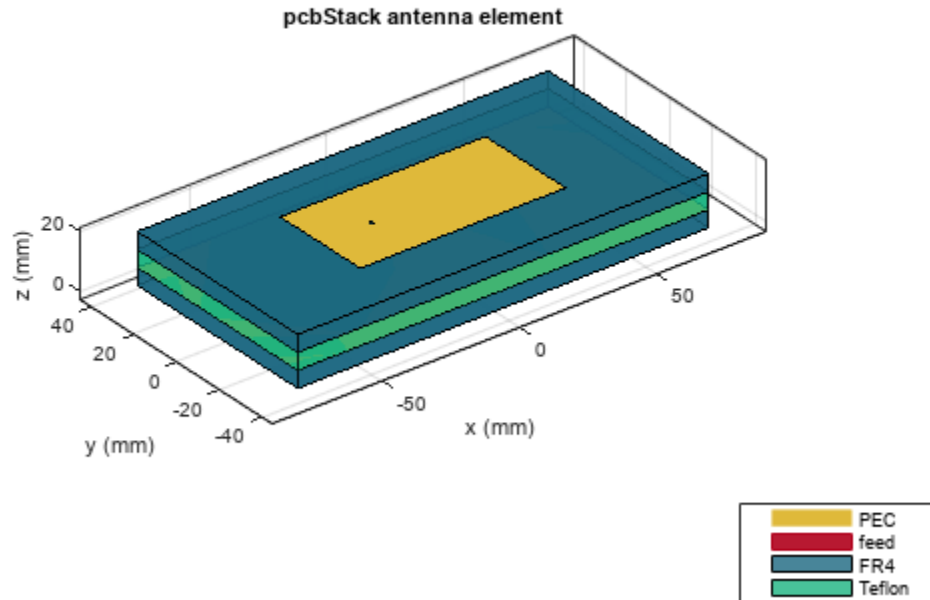


```

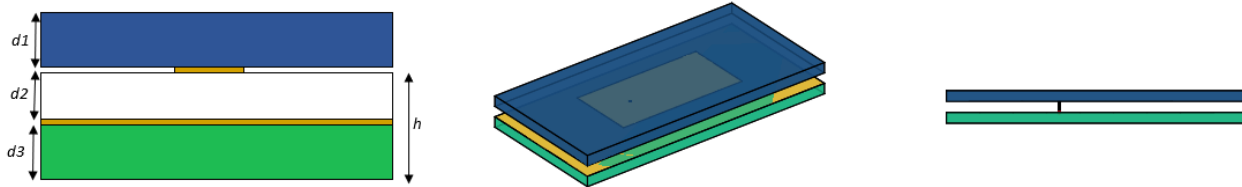
p = pcbStack;
sub1 = dielectric('FR4');
sub2 = dielectric('Teflon');
TopMetal = p.Layers{1,1};
BtmMetal = p.Layers{1,2};

% Set the BoardThickness as sum of dielectric thickness below the top metal
% layer
h = 2*sub1.Thickness+sub2.Thickness;
p.BoardThickness = h;
p.Layers = {TopMetal,sub1,sub2,BtmMetal,sub1};
p.FeedLocations(1,3:4) = [1,4];
figure;
show(p)

```



In the below use case, there is a dielectric layer present above the top metal layer and air dielectric separates the top and the bottom metal layers. There is another dielectric layer below the bottom metal layer. The dielectric layer which is present above the top metal layer is considered as coating and is not included in the `BoardThickness` calculation. The `BoardThickness` (h) is the sum of the thicknesses ($d2+d3$) of all the dielectric layers below the top metal layer.

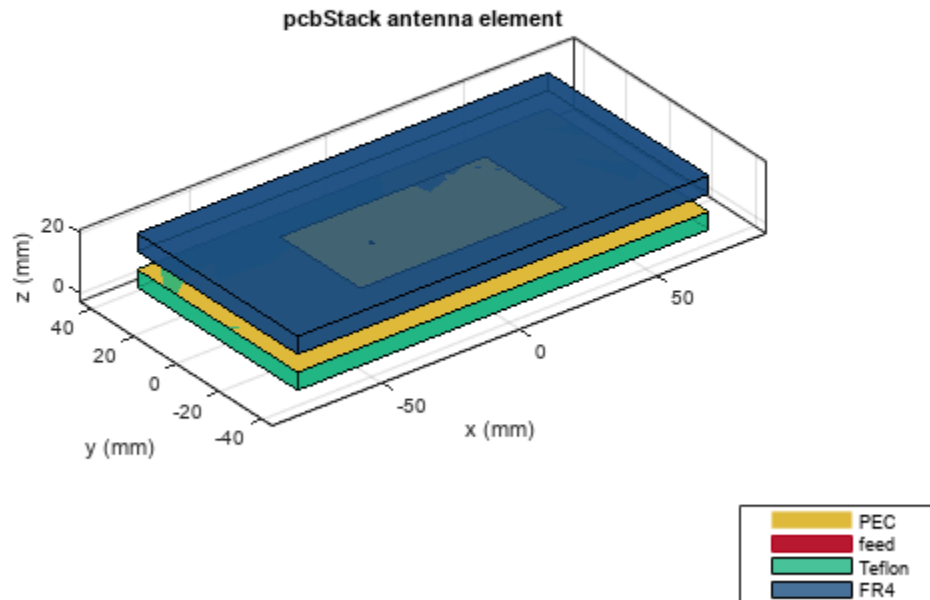


```
p = pcbStack;
sub1 = dielectric('FR4');
sub2 = dielectric('Air');
sub3 = dielectric('Teflon');
TopMetal = p.Layers{1,1};
BtmMetal = p.Layers{1,2};

% Set the BoardThickness as sum of dielectric thickness below the top metal
% layer
h = sub2.Thickness+sub3.Thickness;
p.BoardThickness = h;
p.Layers = {sub1,TopMetal,sub2,BtmMetal,sub3};
```

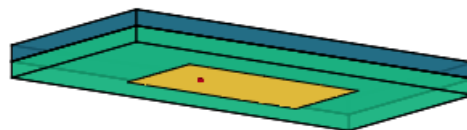


```
p.FeedLocations(1,3:4) = [2,4];
figure;
show(p)
```



Dielectric layers present only above the top metal layer

In the below use case, the PCB has multiple dielectric layers only above the top metal layer. The BoardThickness (h) for this PCB is the sum of thicknesses ($d1+d2$) of all the dielectric layers. **This is applicable only when there are no dielectric layers below the top metal layer.**



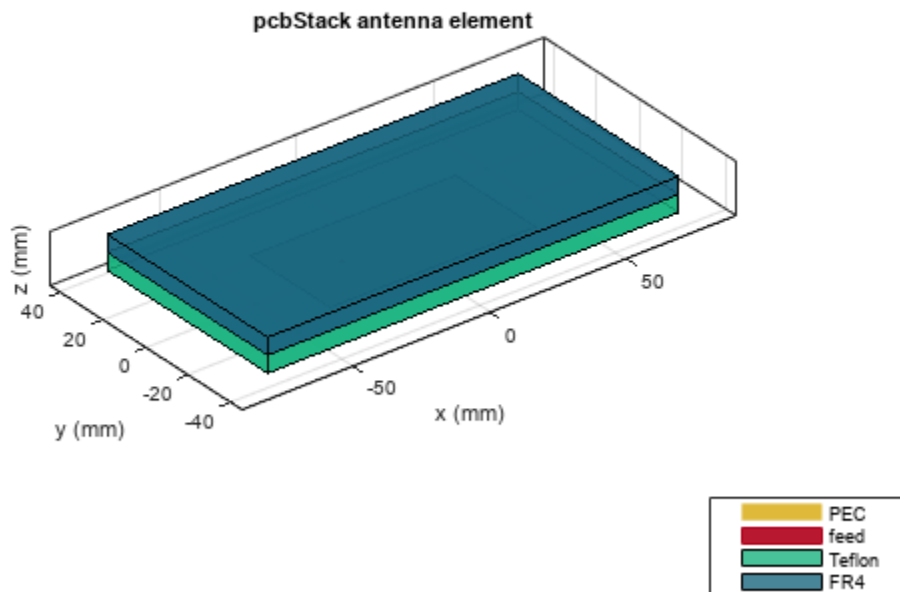
```
p = pcbStack;
sub1 = dielectric('FR4');
sub2 = dielectric('Teflon');
```

```

TopMetal = p.Layers{1,1};

% Set the BoardThickness as sum of dielectric thickness below the top metal
% layer
h = sub1.Thickness+sub2.Thickness;
p.BoardThickness = h;
p.Layers = {sub1,sub2,TopMetal};
p.FeedLocations(1,3:4) = 3;
figure;
show(p)

```



Importing PCB design using pcbReader object

Import the top and bottom layers of the PCB from the Gerber file by setting .gtl and .gbl file to Layer2 and Layer4 in stackUp function. Pass the stackUp object to the PCBReader object.

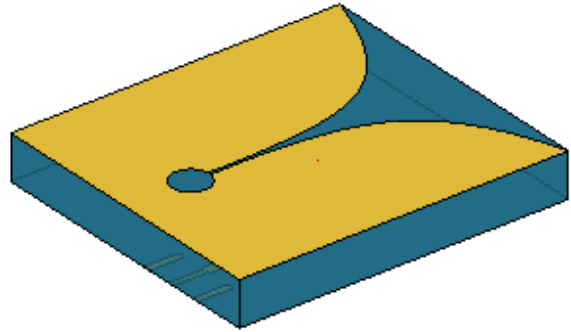
```

S = stackUp;
S.Layer2 = 'UWBVivaldi.gtl';
S.Layer4 = 'UWBVivaldi.gbl';
p = PCBReader ('StackUp',S);

```

Create pcbStack of PCBReader object

When the PCBReader object is converted into pcbStack, the pcbStack reads all the 5 layers of the stackUp object, where air dielectric is the default top and bottom layer. This is a multiple dielectric layers use case. Hence, the BoardThickness (h) is the sum of thicknesses (d_2+d_3) of the dielectric layers below the top metal layer.



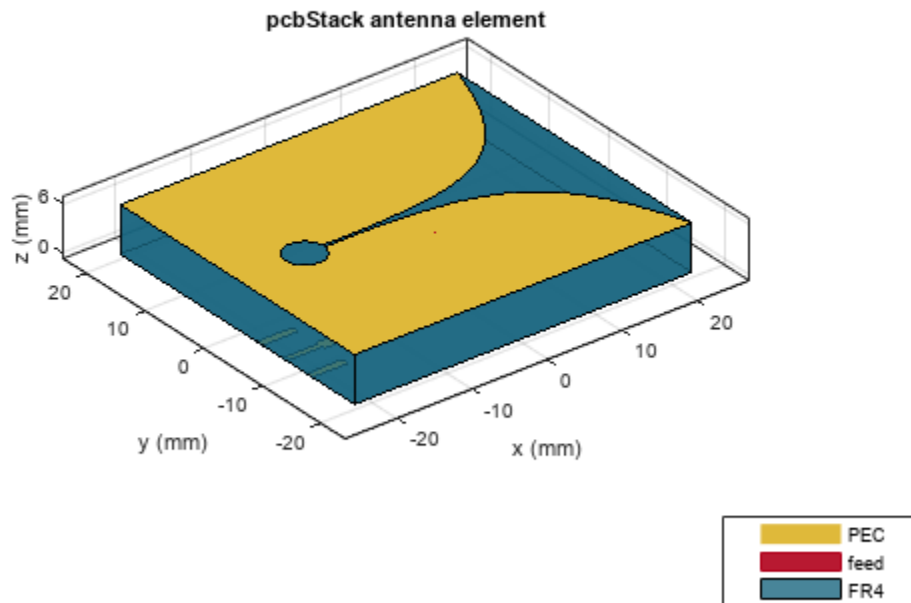
```
pcb = pcbStack(p);
pcb.BoardThickness = pcb.Layers{3}.Thickness+pcb.Layers{5}.Thickness;
pcb.FeedLocations = [0,-5e-3,2]
```

```
pcb =
```

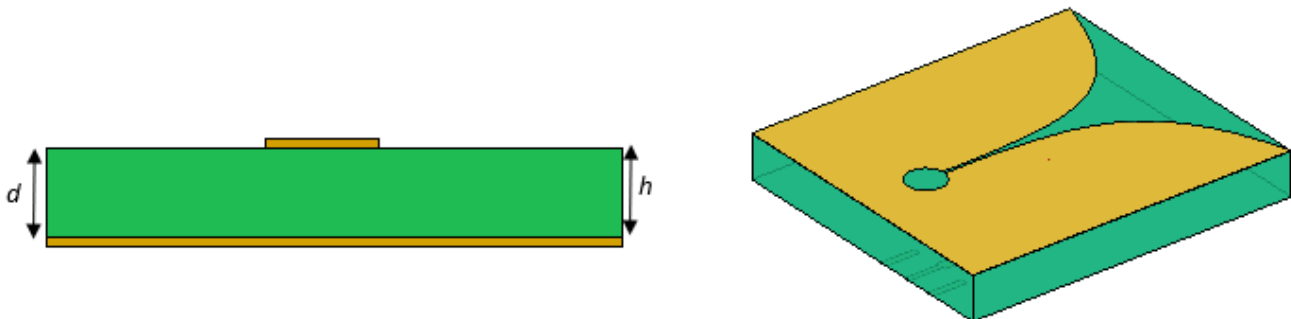
```
pcbStack with properties:
```

```
    Name: 'UWBVivaldi'
    Revision: 'v1.0'
    BoardShape: [1x1 antenna.Rectangle]
    BoardThickness: 0.0061
    Layers: {[1x1 dielectric] [1x1 antenna.Polygon] [1x1 dielectric] [1x1 antenna.Polygon]}
    FeedLocations: [0 -0.0050 2]
    FeedDiameter: 0.0100
    ViaLocations: []
    ViaDiameter: []
    FeedViaModel: 'square'
    FeedVoltage: 1
    FeedPhase: 0
    Conductor: [1x1 metal]
    Tilt: 0
    TiltAxis: [1 0 0]
    Load: [1x1 lumpedElement]
```

```
pcb.FeedDiameter = 0.1e-3;
show(pcb);
```



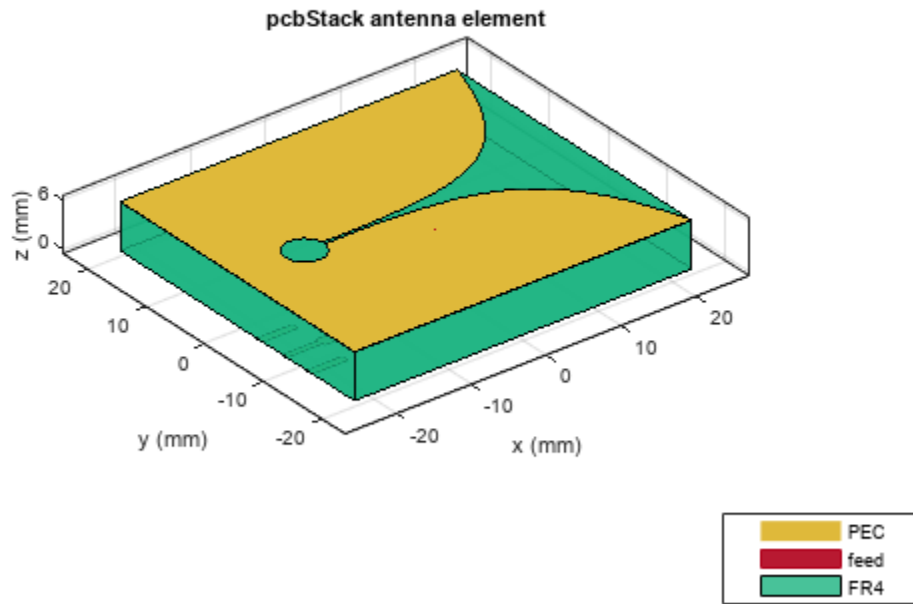
Modify the above use case by removing the top and bottom air dielectric layers as shown in figure below. This use case is now a single-layer dielectric pcbStack. Now you can change the dielectric Thickness (d) by changing the BoardThickness (h).



```

pcb = pcbStack(p);
d = pcb.Layers{3};
pcb.BoardThickness = d.Thickness;
L = {pcb.Layers{2},pcb.Layers{3},pcb.Layers{4}};
pcb.Layers = L;
pcb.FeedLocations = [0,-5e-3,1];
pcb.FeedDiameter = 0.1e-3;
show(pcb);

```



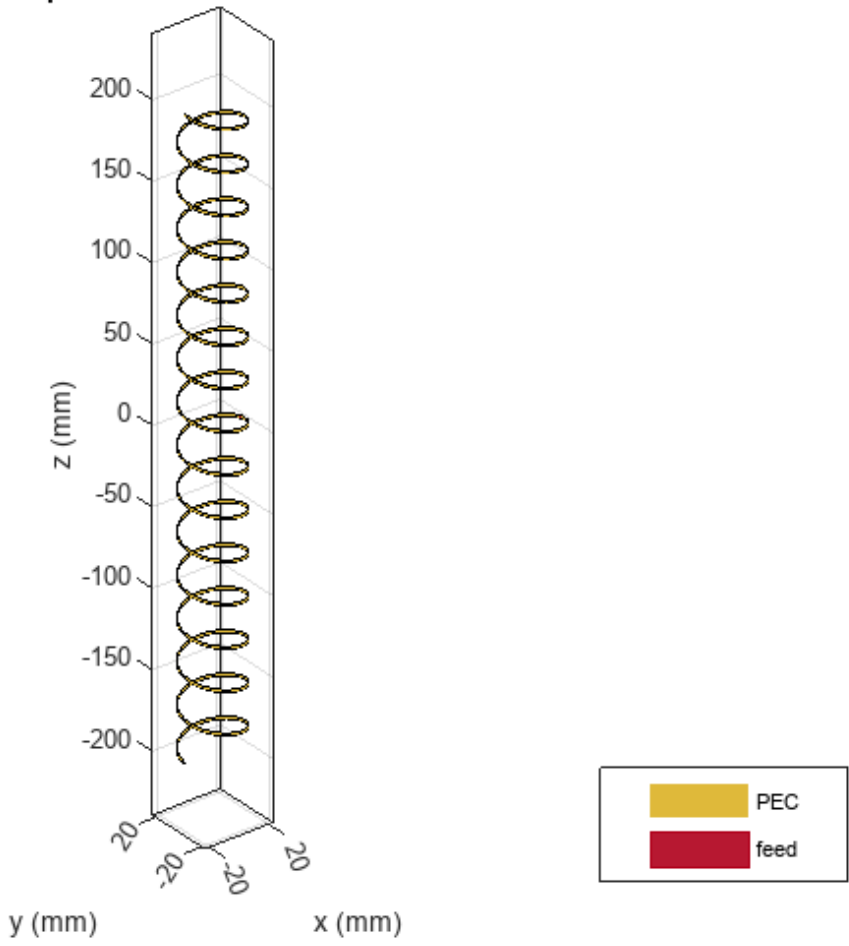
Analyze Metal Conductors in Helical Dipole Antenna

This example shows how to design dipole helix antennas using different conductors and analyze their characteristics as functions of the conductor thickness and conductivity.

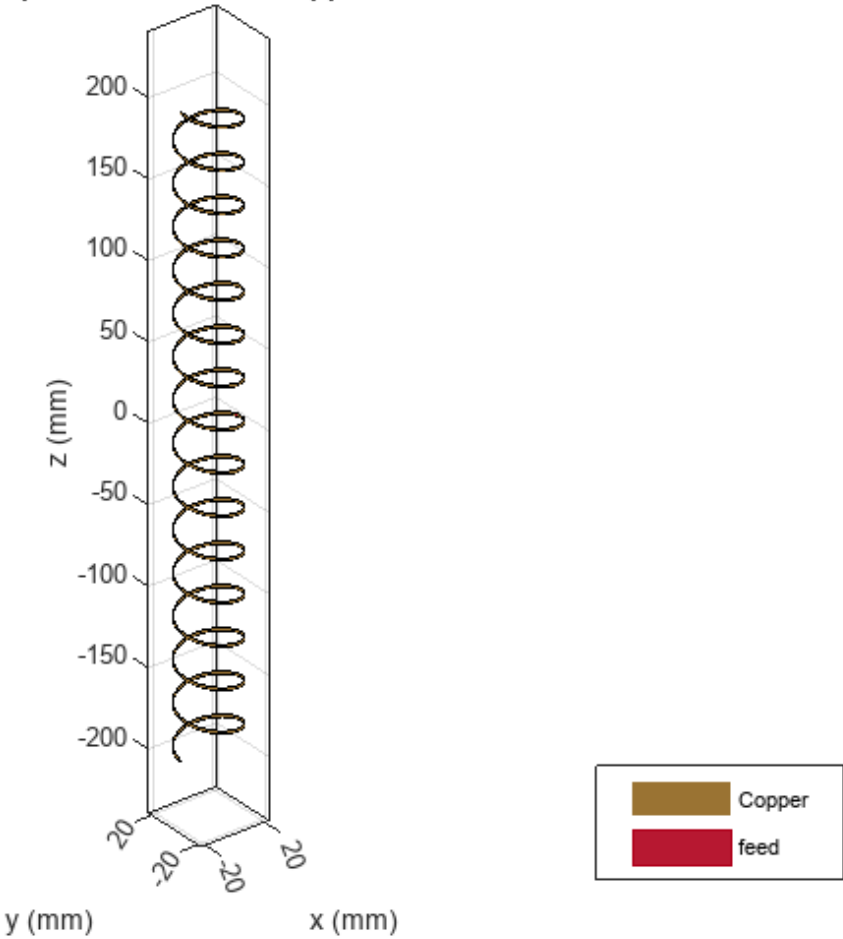
Create antennas with perfect electrical conductor (PEC), copper, aluminium, gold, silver, and zinc as the conducting materials. Create a row vector with the metal names and use the `design` function to create the antennas resonating at 1.8 GHz. Store the antennas in a row vector, in which the conductor of each antenna in the `antennas` vector corresponds to the material in the `metals` vector with the same index.

```
designFrequency = 1.8e9;
metals = ["PEC" "Copper" "Aluminium" "Gold" "Silver" "Zinc"];
antennas = repmat(dipoleHelix,size(metals));
for idx = 1:length(metals)
    helix = dipoleHelix(Conductor=metal(metals(idx)));
    antennas(idx) = design(helix,designFrequency);
    figure
    show(antennas(idx))
    title(sprintf("Helical Dipole Antenna with %s Conductor",metals(idx)))
end
```

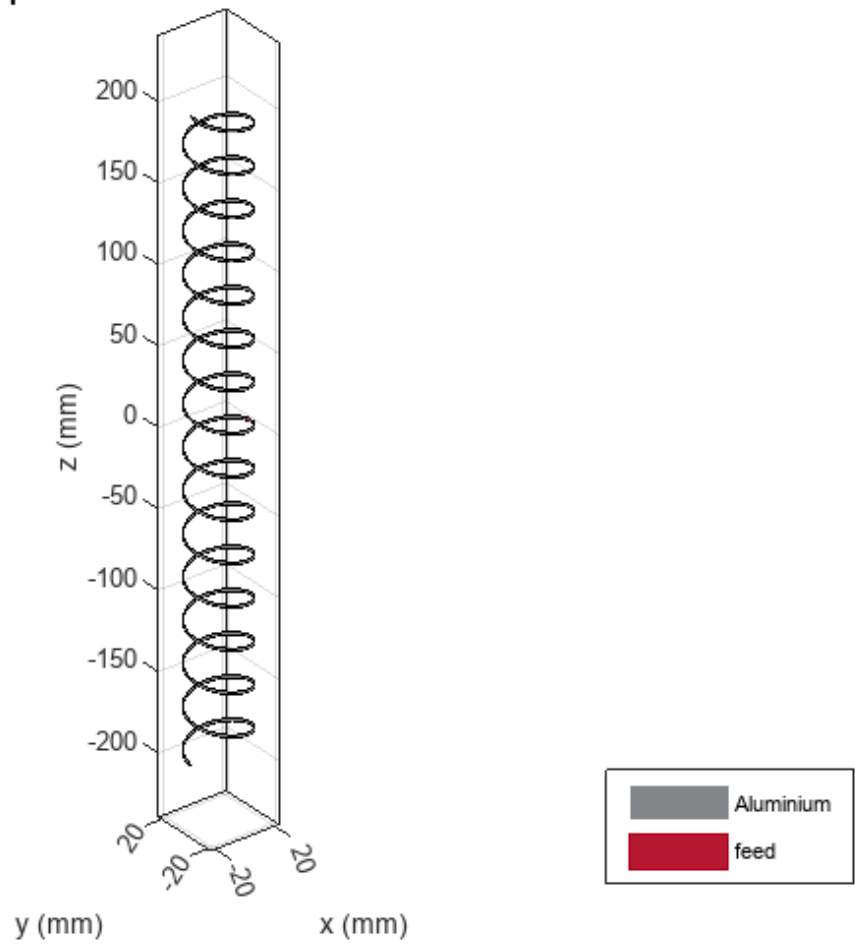
Helical Dipole Antenna with PEC Conductor



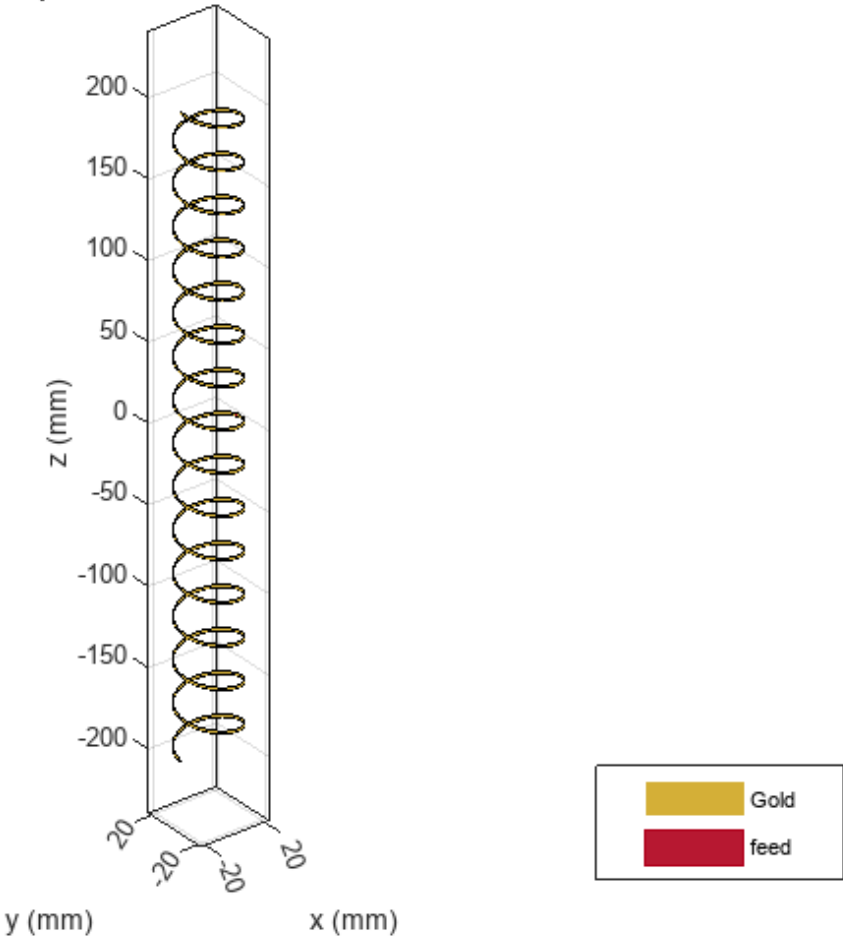
Helical Dipole Antenna with Copper Conductor



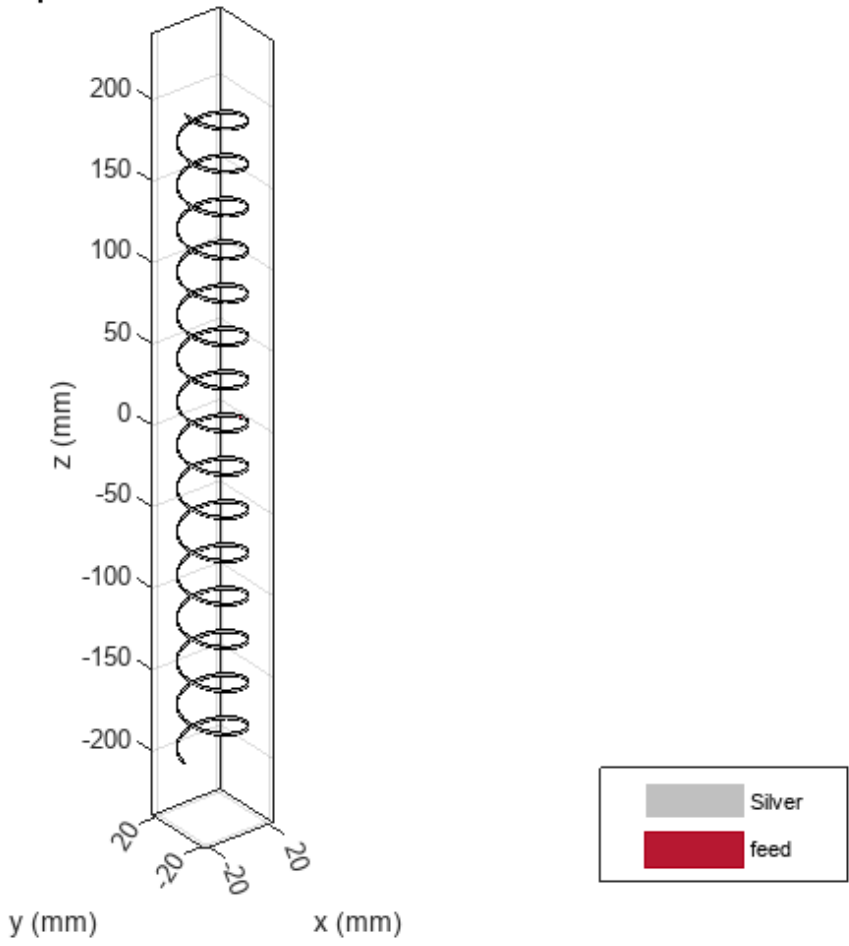
Helical Dipole Antenna with Aluminium Conductor



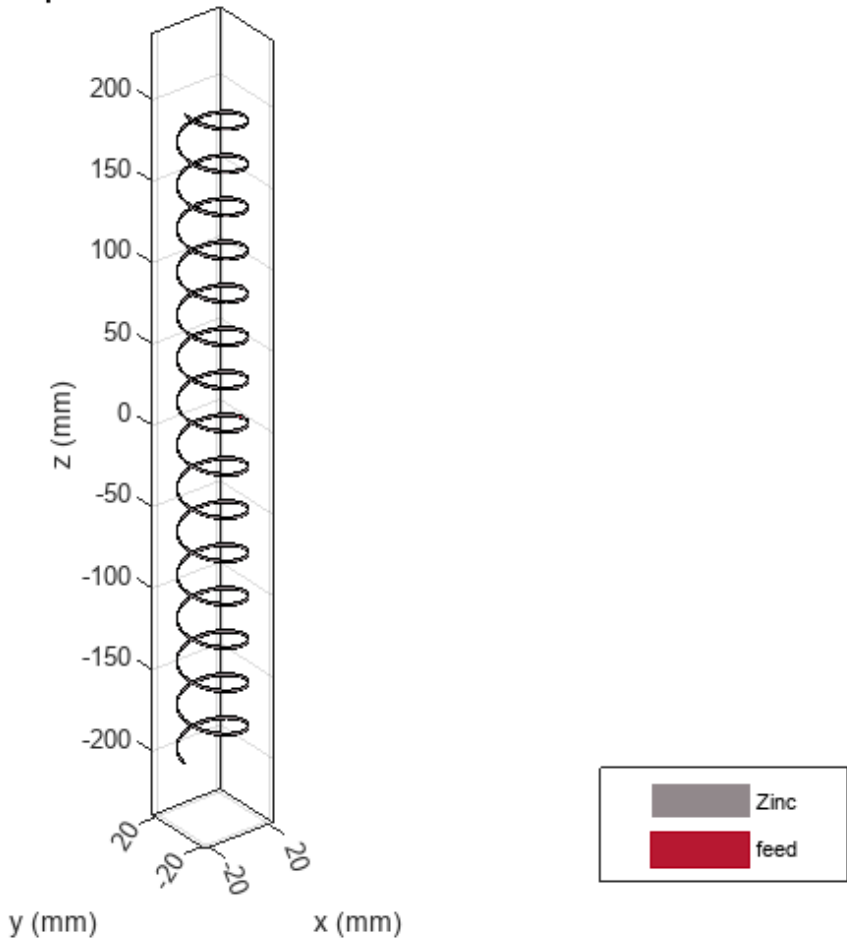
Helical Dipole Antenna with Gold Conductor



Helical Dipole Antenna with Silver Conductor



Helical Dipole Antenna with Zinc Conductor



Calculate Antenna Characteristics

Compute the impedance, directivity, gain, realized gain, return loss, and efficiency as functions of frequency for all the antennas. Store the values in a matrix in which each row corresponds to an antenna and each column corresponds to a frequency value.

```

freq = 1e9:50e6:3e9;
angles = 1:1:360;
z = zeros(length(antennas),length(freq));
loss = zeros(length(antennas),length(freq));
res = [1.4e9,1.8e9];
pat = zeros(length(antennas),length(res),length(angles));
eff = zeros(length(antennas),length(freq));
direct = zeros(length(antennas),length(freq));
gain = zeros(length(antennas),length(freq));
realizedgain = zeros(length(antennas),length(freq));
for idx = 1:length(antennas)
    ant = antennas(idx);
    z(idx,:) = impedance(ant,freq);
    [pat(idx,:,:),~,~] = pattern(ant,res,0,angles);
    direct(idx,:) = pattern(ant,freq,0,90,Type="directivity)';

```

```

gain(idx,:) = pattern(ant,freq,0,90,Type="gain");
realizedgain(idx,:) = pattern(ant,freq,0,90,Type="realizedgain");
loss(idx,:) = returnLoss(ant,freq);
eff(idx,:) = efficiency(ant,freq);
end

```

Vary Number of Turns

Vary the number of turns and determine whether the gain changes at azimuth and elevation angles of 0.

```

turns = 1:0.1:30;
pat2 = zeros(length(antennas),length(turns));
for idx = 1:length(metals)
    for jj = 1:length(turns)
        ant1 = dipoleHelix(Conductor=metal(metals(idx)));
        ant1.Turns = turns(jj);
        ant1 = design(ant1,designFrequency);
        pat2(idx,jj) = pattern(ant1,designFrequency,0,0);
    end
end

```

Vary Thickness

Vary the thickness of the conductor to compute impedance and efficiency. You cannot change the thickness of the PEC, because it has infinite conductivity and zero thickness, so omit PEC from the metals vector.

```

thicknesses = linspace(2e-7,1e-5,200);
z2 = zeros(length(antennas),length(thicknesses));
eff2 = zeros(length(antennas),length(thicknesses));
for idx = 2:length(metals)
    ant2 = dipoleHelix(Conductor=metal(metals(idx)));
    ant2 = design(ant2,designFrequency);
    for jj = 1:length(thicknesses)
        ant2.Conductor.Thickness = thicknesses(jj);
        z2(idx,jj) = impedance(ant2,designFrequency);
        eff2(idx,jj) = efficiency(ant2,designFrequency);
    end
end

```

Vary Conductivity

Vary the conductivity and calculate the return loss and gain. You cannot change the conductivity of the PEC, because it has infinite conductivity and zero thickness, so omit PEC from the metals vector.

```

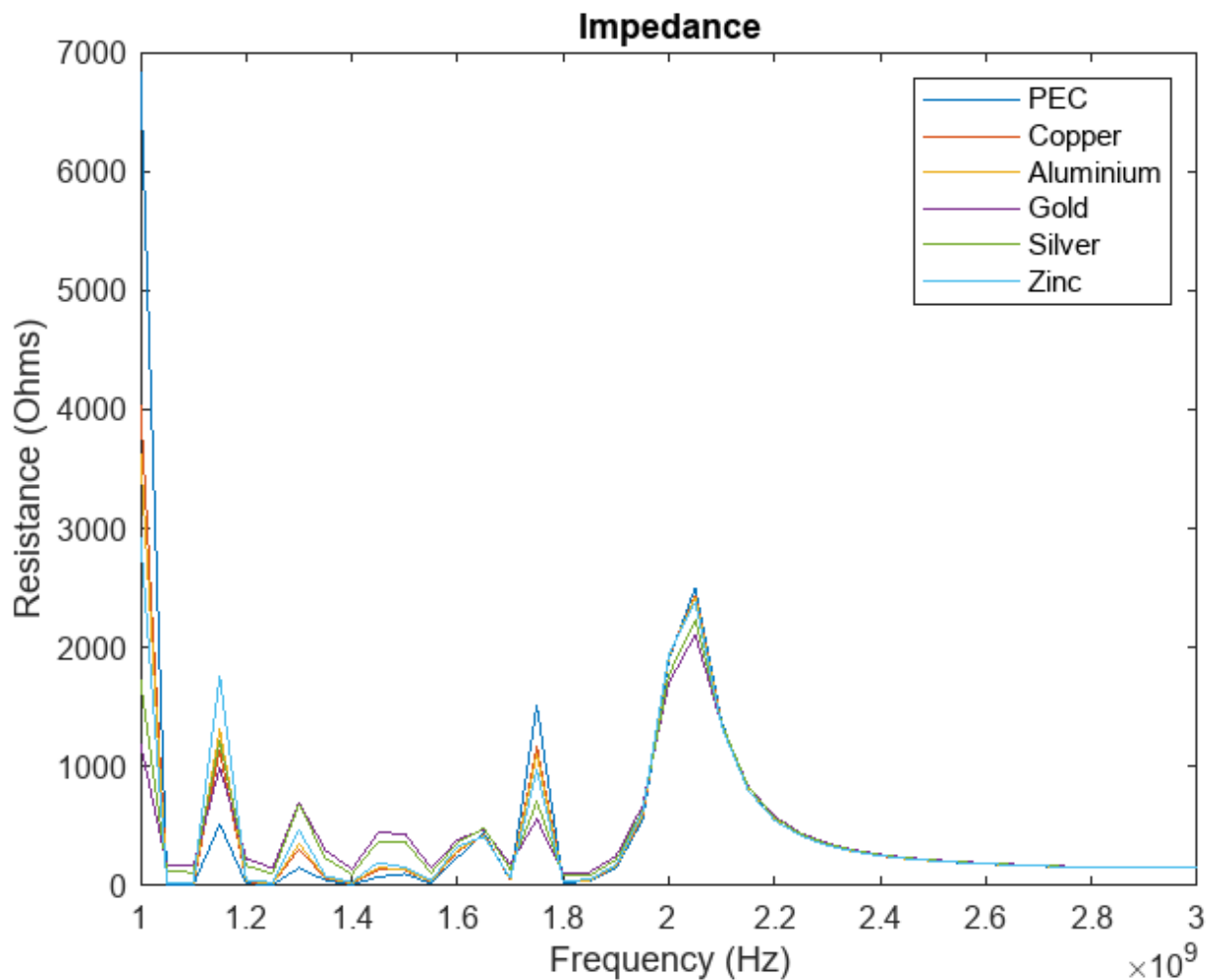
conductivity = linspace(10e6,100e6,20);
loss2 = zeros(length(antennas),length(conductivity));
gain2 = zeros(length(antennas),length(conductivity));
for idx = 2:length(metals)
    for jj = 1:length(conductivity)
        ant3 = dipoleHelix(Conductor=metal(metals(idx)));
        ant3.Conductor.Conductivity = conductivity(jj);
        ant3 = design(ant3,designFrequency);
        loss2(idx,jj) = returnLoss(ant3,designFrequency);
        gain2(idx,jj) = pattern(ant3,designFrequency,0,90,Type="gain");
    end
end

```

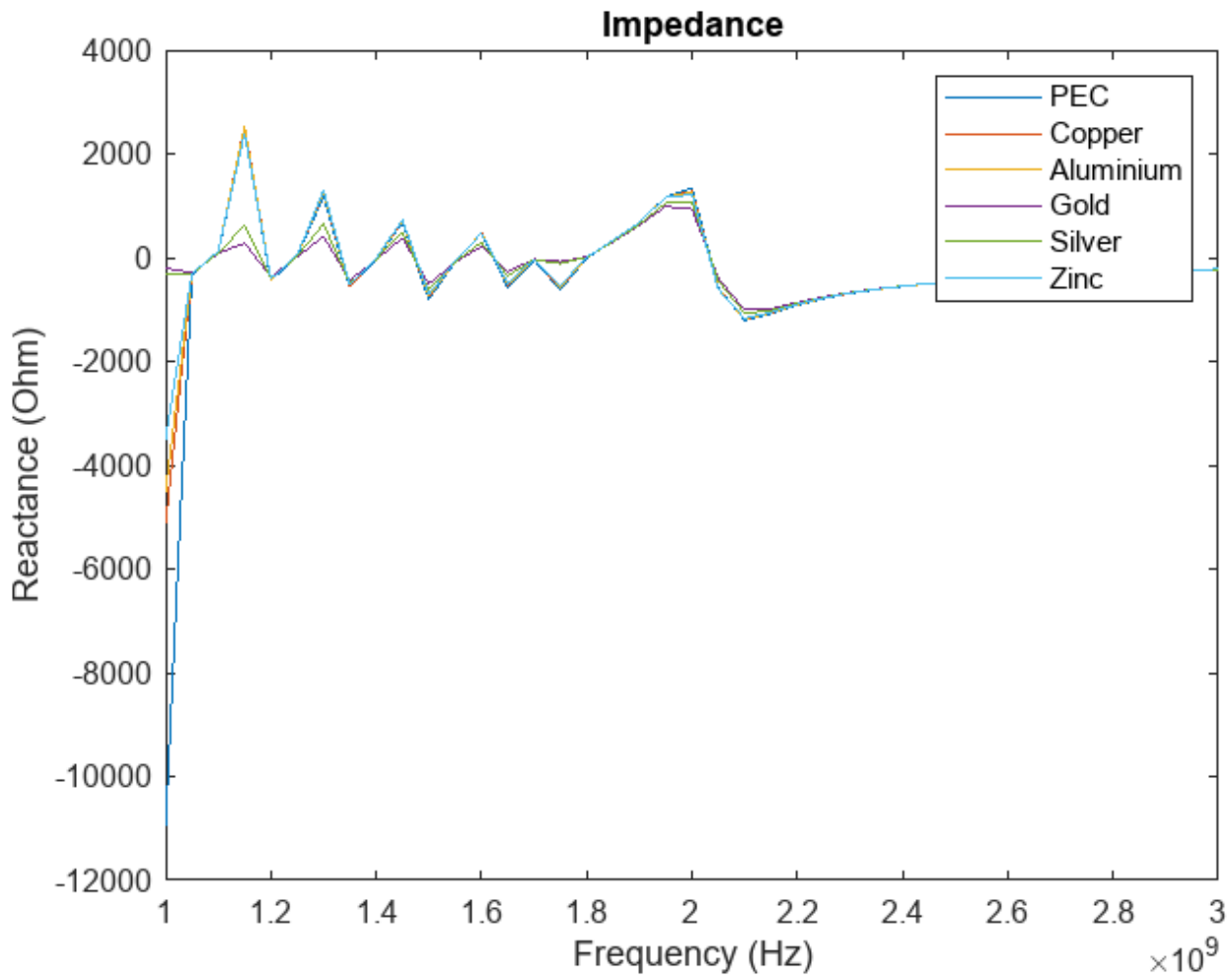
Plotting Impedance Against Frequency

Plot the impedance, directivity, gain, realized gain, return loss, and efficiency as functions of frequency. Because the frequency varies along the columns and the conductor varies along the rows, using the `plot` function results in 6 lines, one for each conductor.

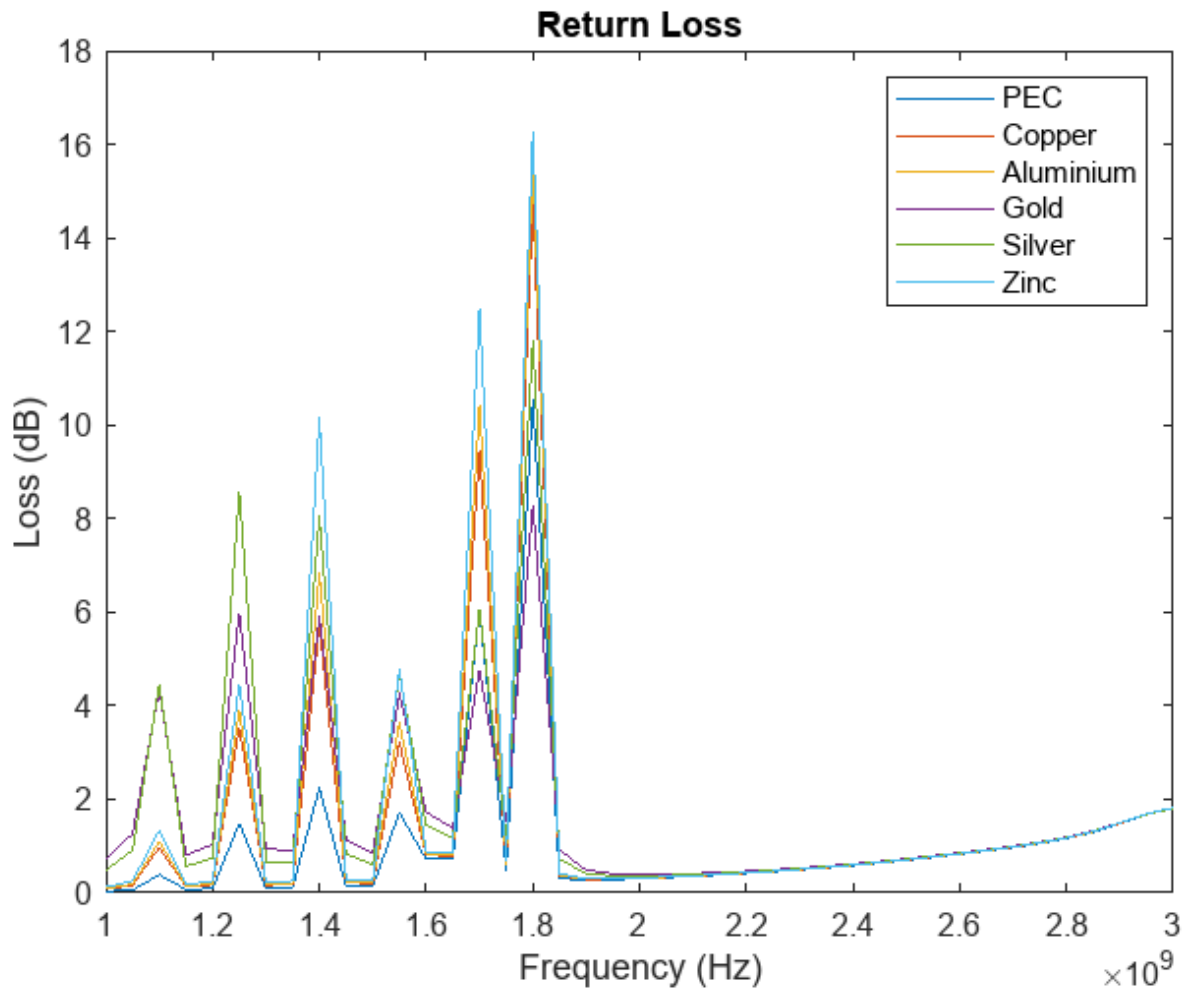
```
figure
plot(freq,real(z))
title("Impedance")
xlabel("Frequency (Hz)")
ylabel("Resistance (Ohms)")
legend(metals)
```



```
figure
plot(freq,imag(z))
title("Impedance")
xlabel("Frequency (Hz)")
ylabel("Reactance (Ohm)")
legend(metals)
```

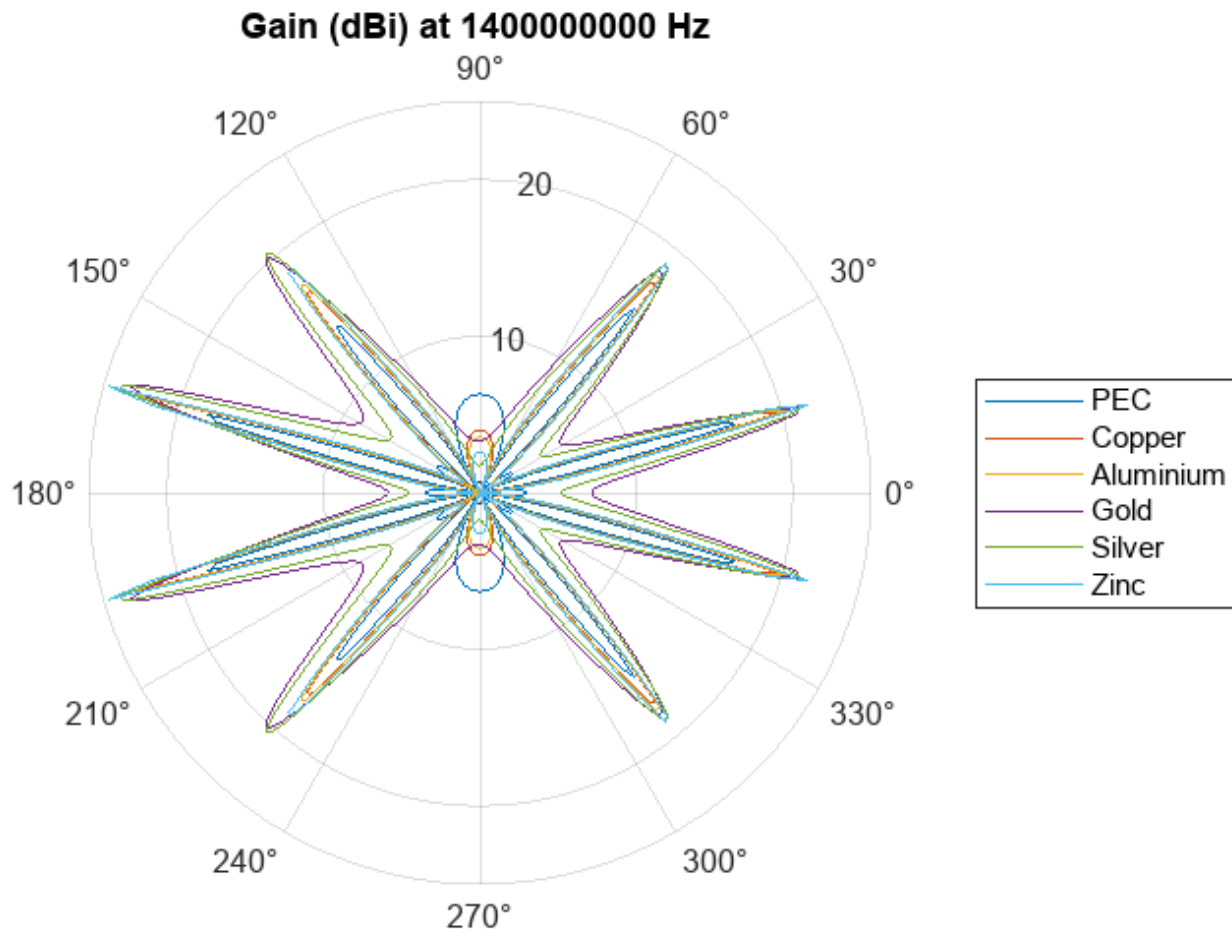


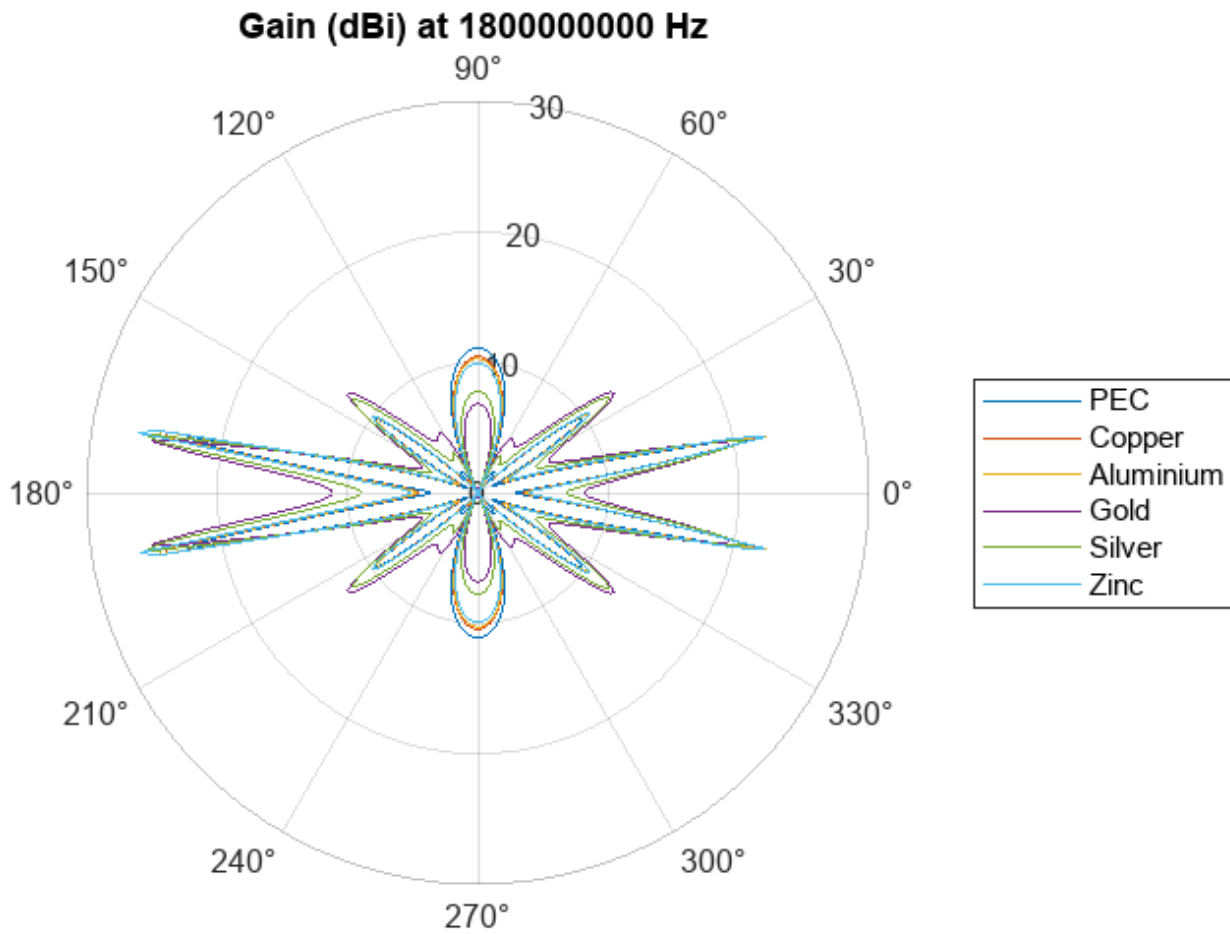
```
figure
plot(freq,loss)
title("Return Loss")
xlabel("Frequency (Hz)")
ylabel("Loss (dB)")
legend(metals)
```



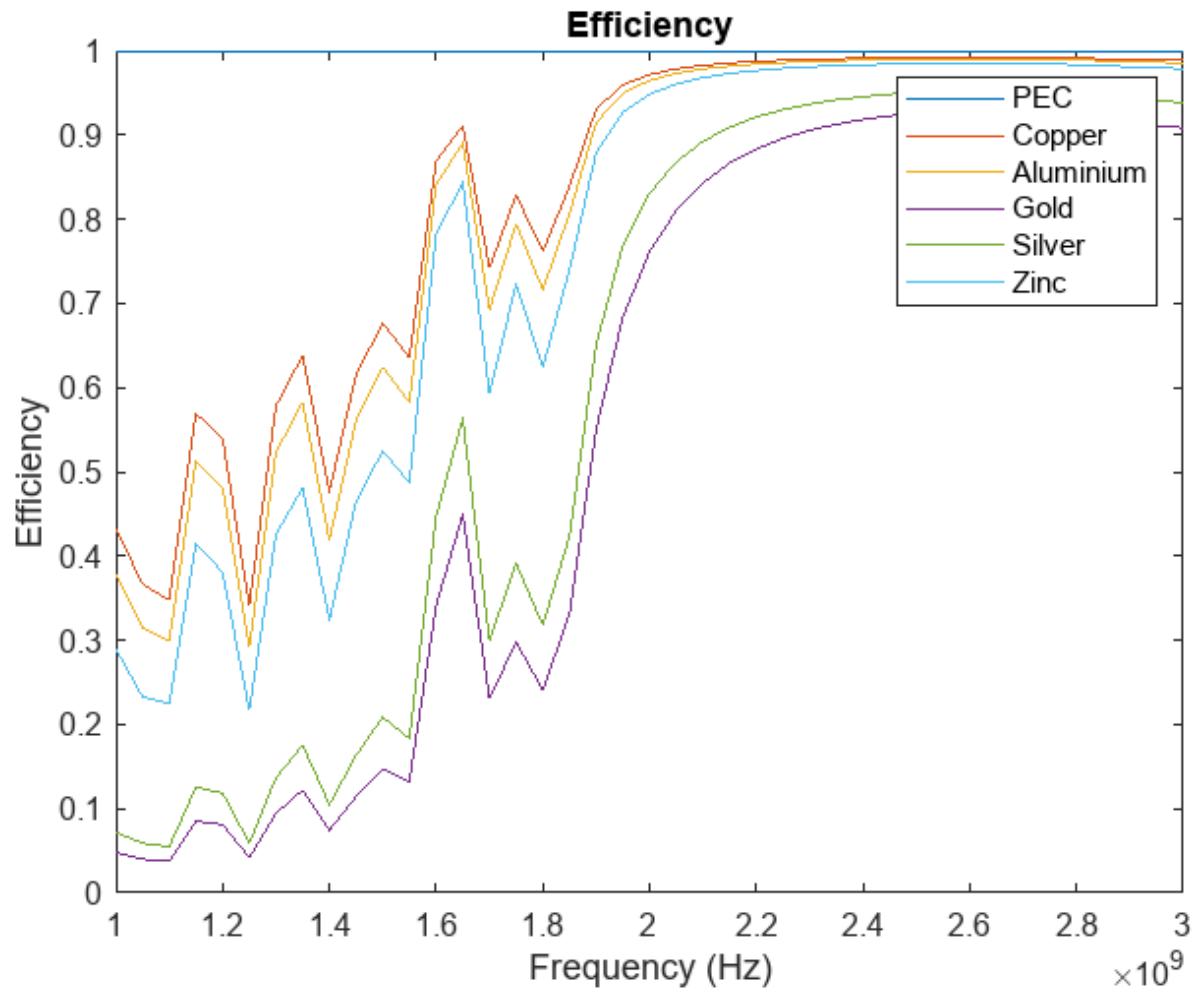
To plot the gain, use the reshape function to convert the 6-by-1-by-360 pattern arrays to 6-by-360 pattern arrays.

```
for idx = 1:length(res)
    figure
    polarplot(angles*pi/180,reshape(pat(:,idx,:),[length(antennas),length(angles)]))
    title(sprintf("Gain (dBi) at %d Hz ",res(idx)));
    legend(metals)
end
```

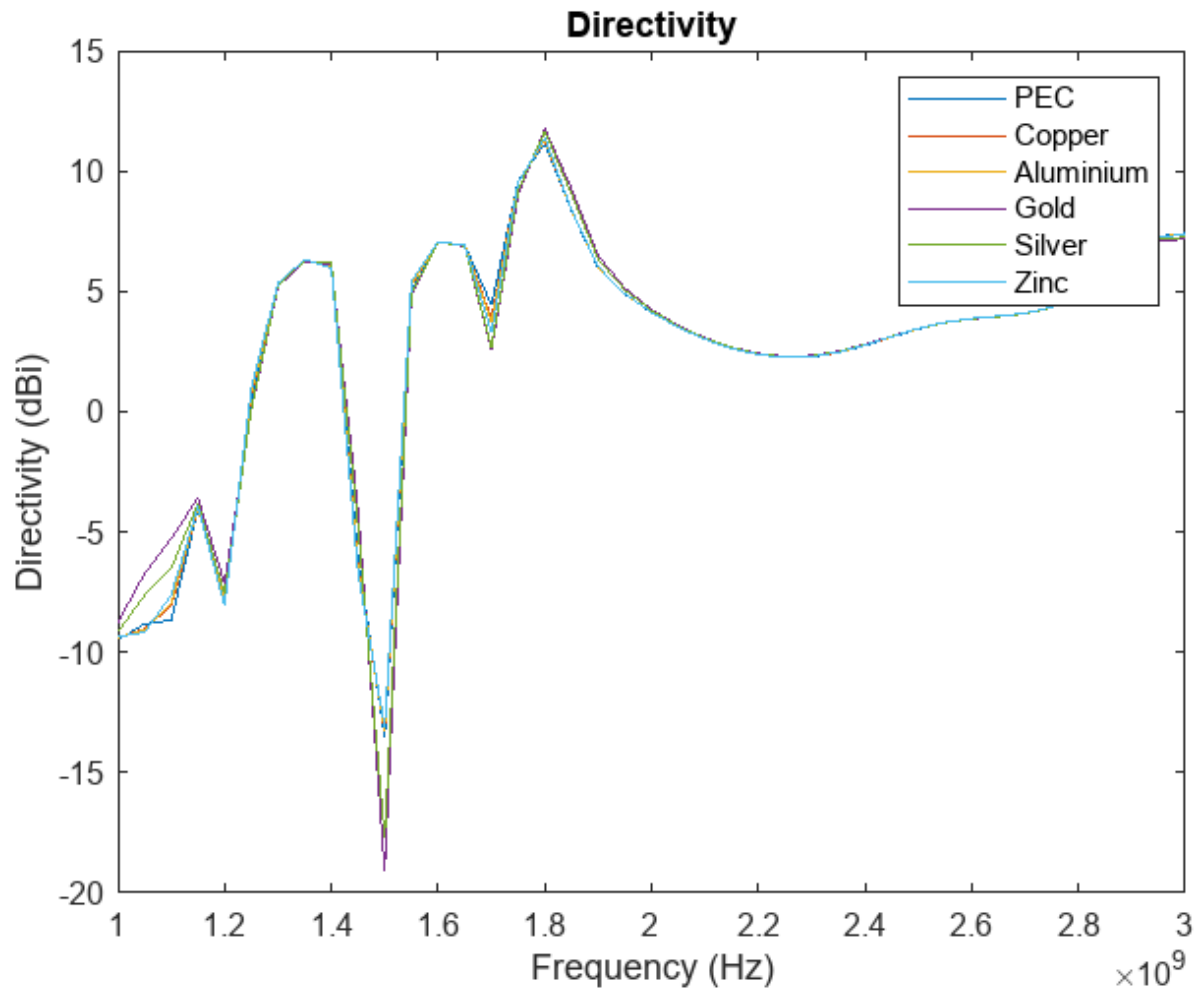





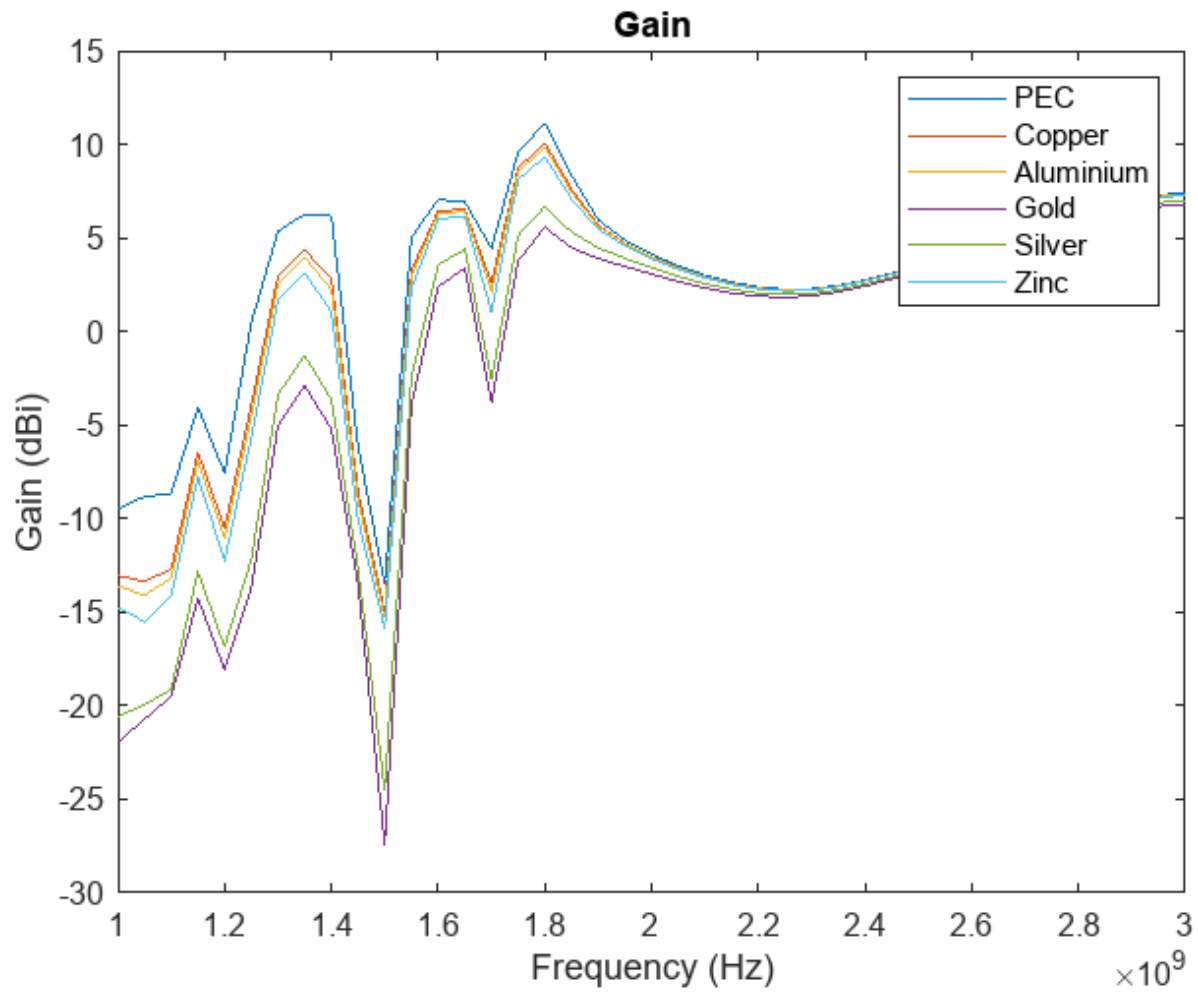
```
figure
plot(freq,eff)
title("Efficiency")
xlabel("Frequency (Hz)")
ylabel("Efficiency")
legend(metals)
```



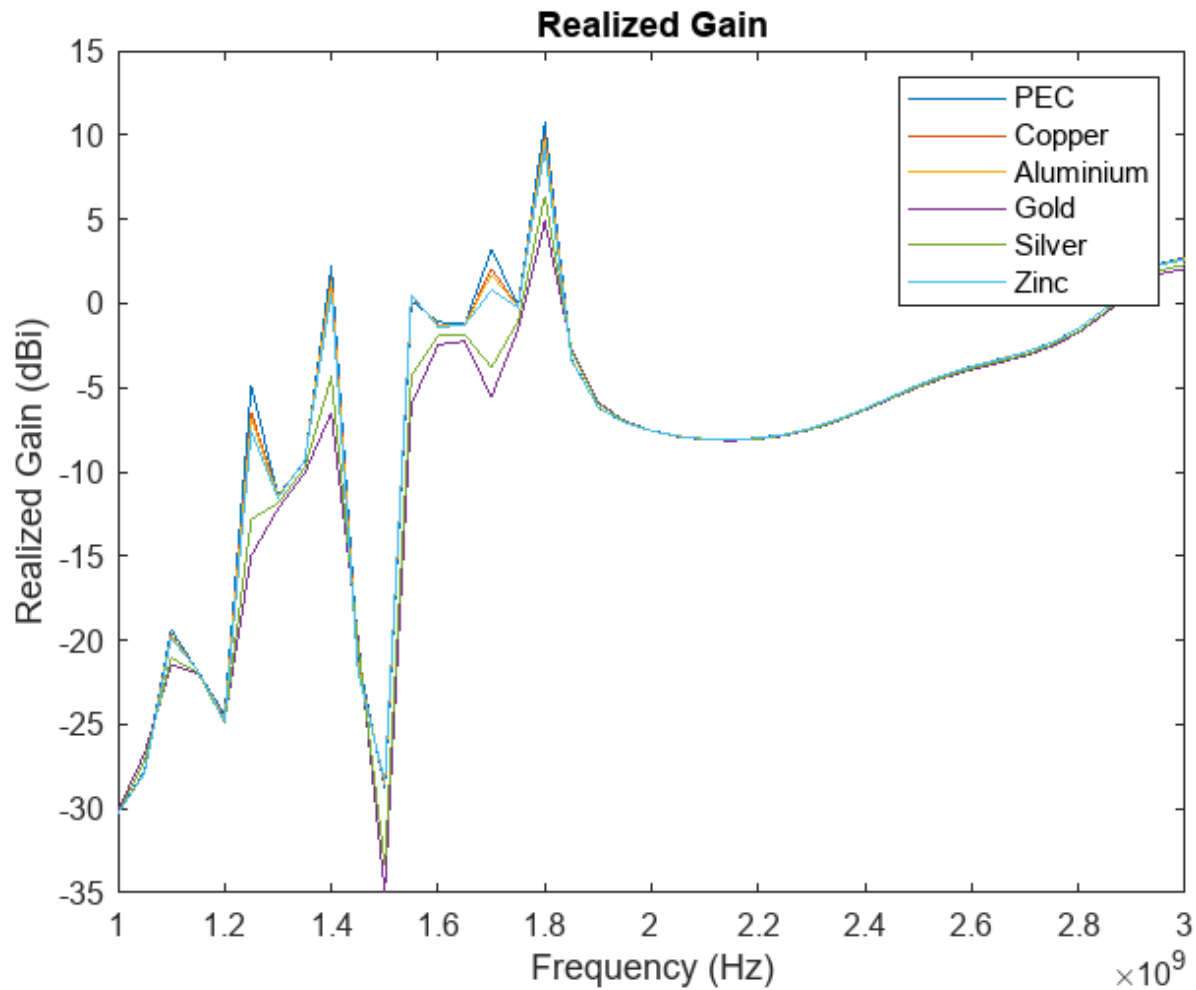
```
figure
plot(freq,direct)
title("Directivity")
xlabel("Frequency (Hz)")
ylabel("Directivity (dBi)")
legend(metals)
```



```
figure
plot(freq,gain)
title("Gain")
xlabel("Frequency (Hz)")
ylabel("Gain (dBi)")
legend(metals)
```



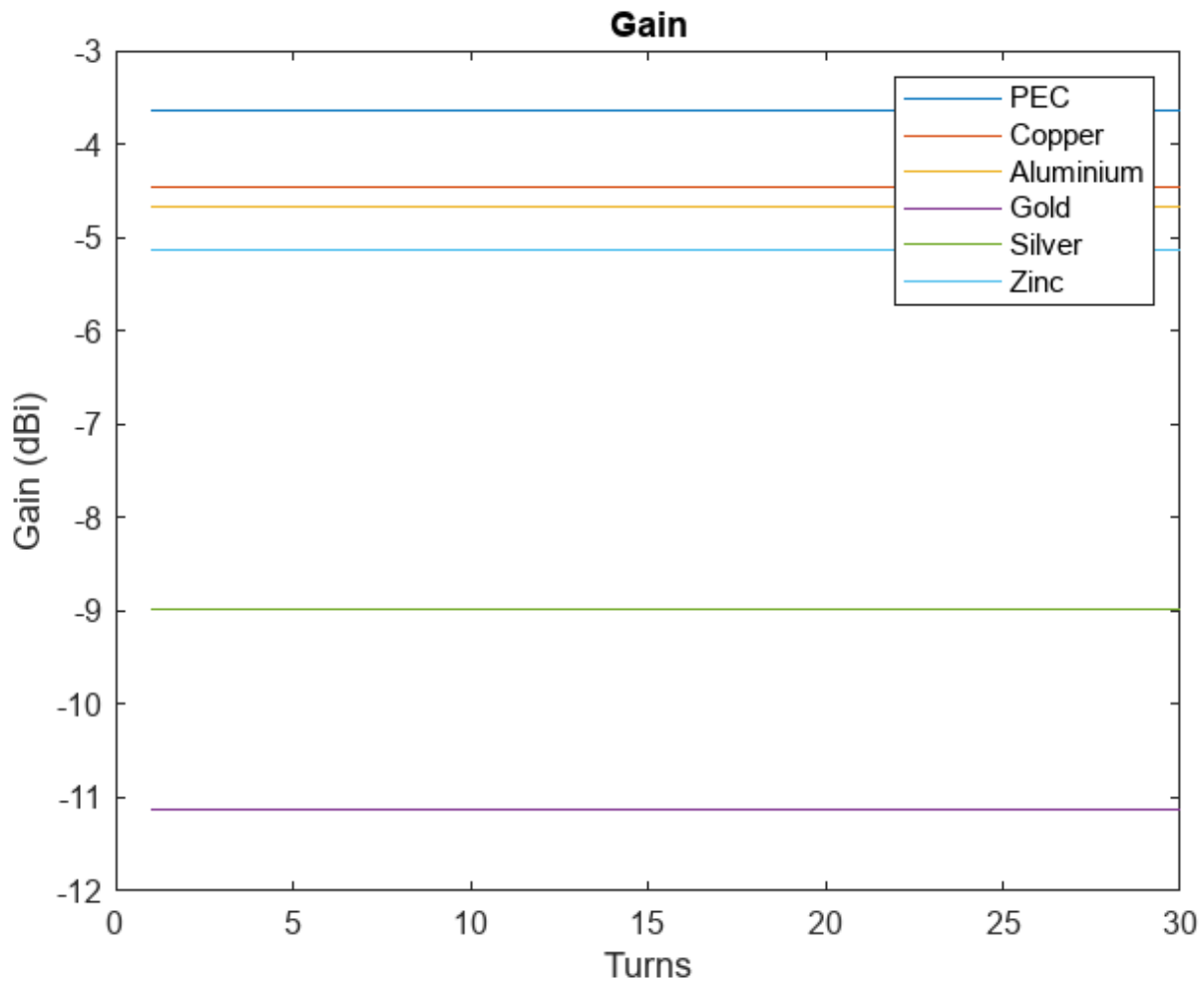
```
figure
plot(freq,realizedgain)
title("Realized Gain")
xlabel("Frequency (Hz)")
ylabel("Realized Gain (dBi)")
legend(metals)
```



Plot Radiation Pattern Against Number of Turns

Plot the radiation pattern of each antenna as a function of the number of turns. The number of turns does not affect the gain of the antenna. The gain is constant for each conductor, regardless of the number of turns.

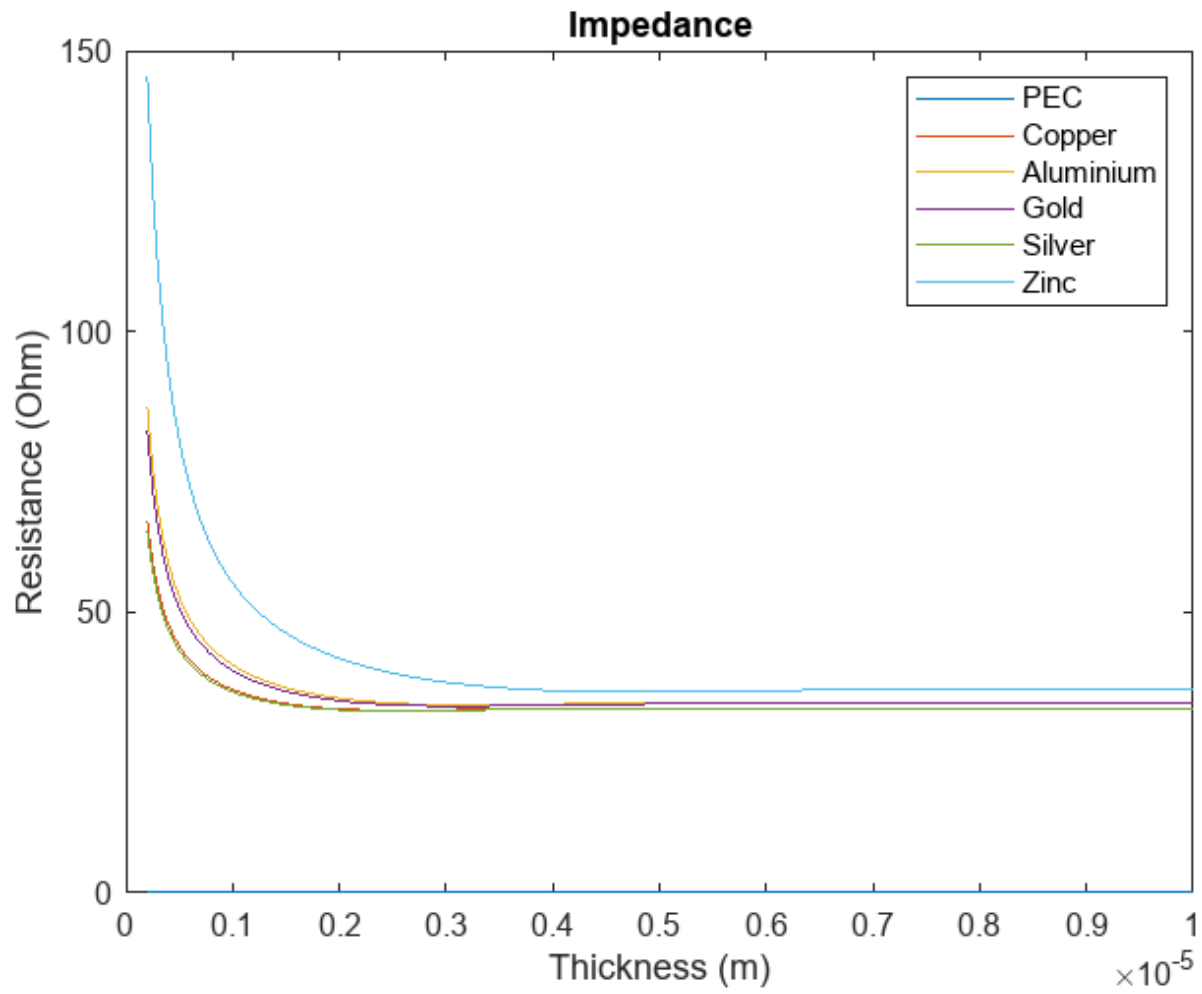
```
figure
plot(turns,pat2)
title("Gain")
xlabel("Turns")
ylabel("Gain (dBi)")
legend(metals)
```



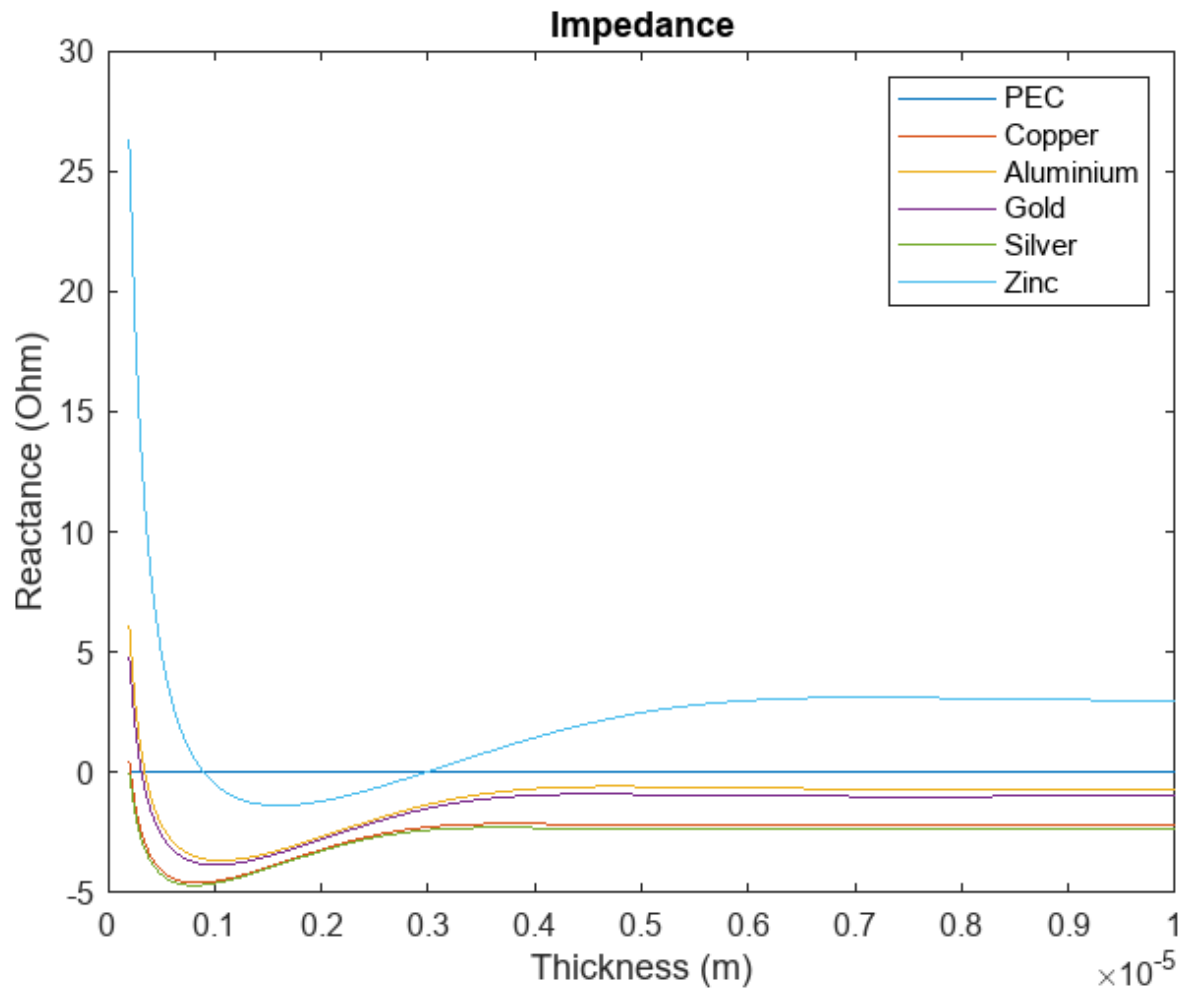
Plot Impedance and Efficiency Against Thickness

Plot the impedance and efficiency as functions of the thickness of the conductors.

```
figure
plot(thicknesses, real(z2))
title("Impedance")
xlabel("Thickness (m)")
ylabel("Resistance (Ohm)")
legend(metals)
```



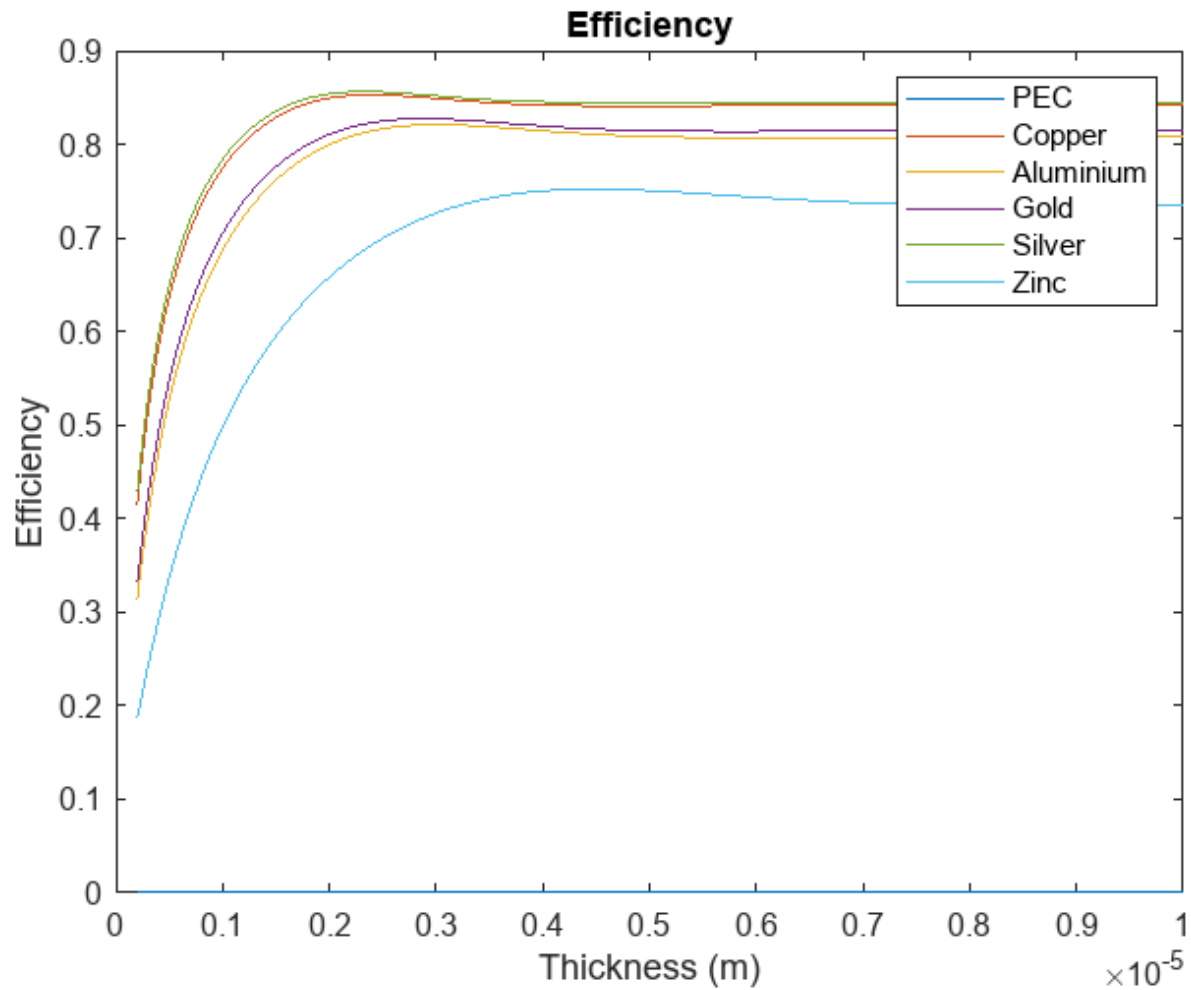
```
figure
plot(thicknesses, imag(z2))
title("Impedance")
xlabel("Thickness (m)")
ylabel("Reactance (Ohm)")
legend(metals)
```

```

figure
plot(thicknesses,eff2)
title("Efficiency")
xlabel("Thickness (m)")
ylabel("Efficiency")
legend(metals)

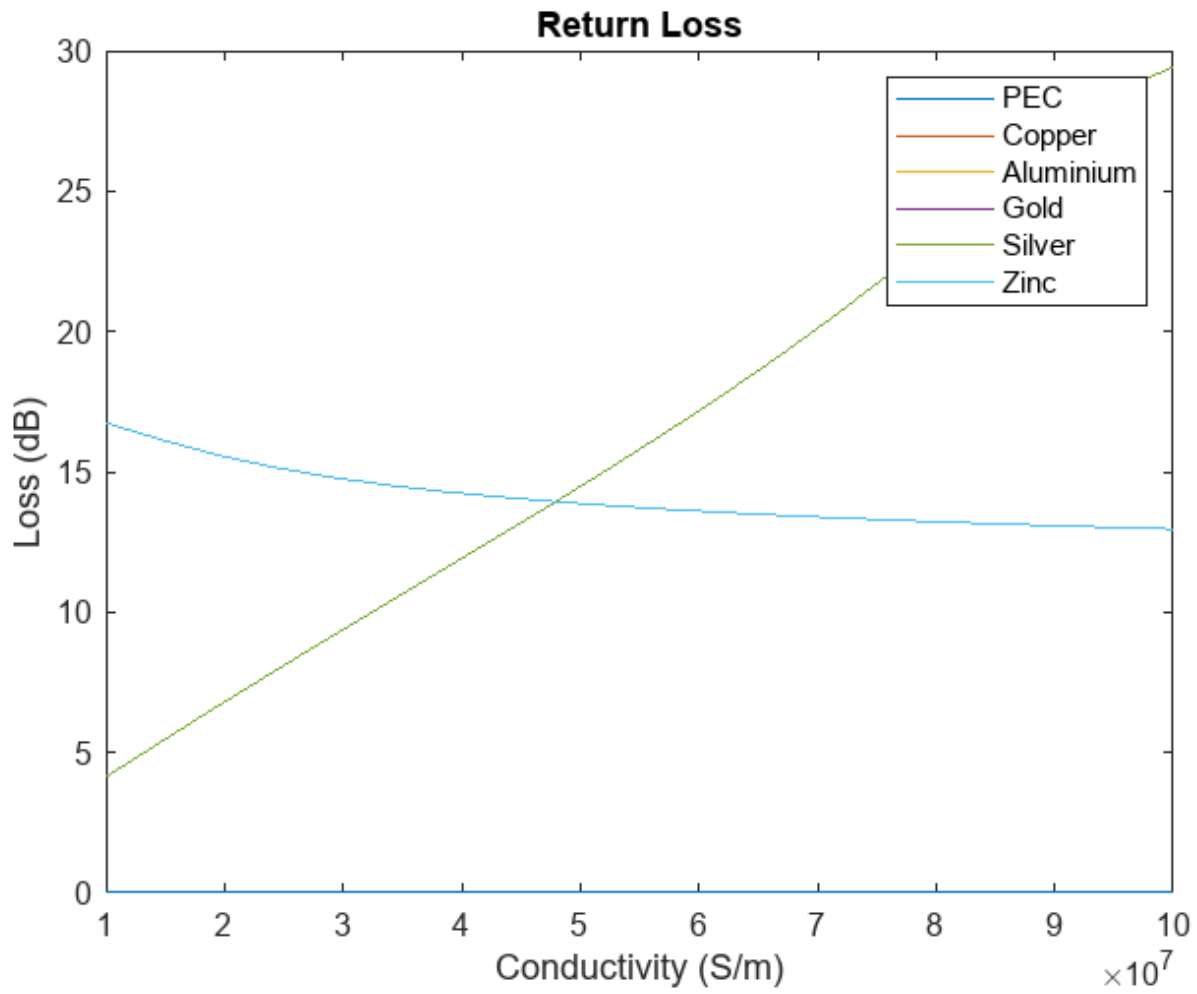
```



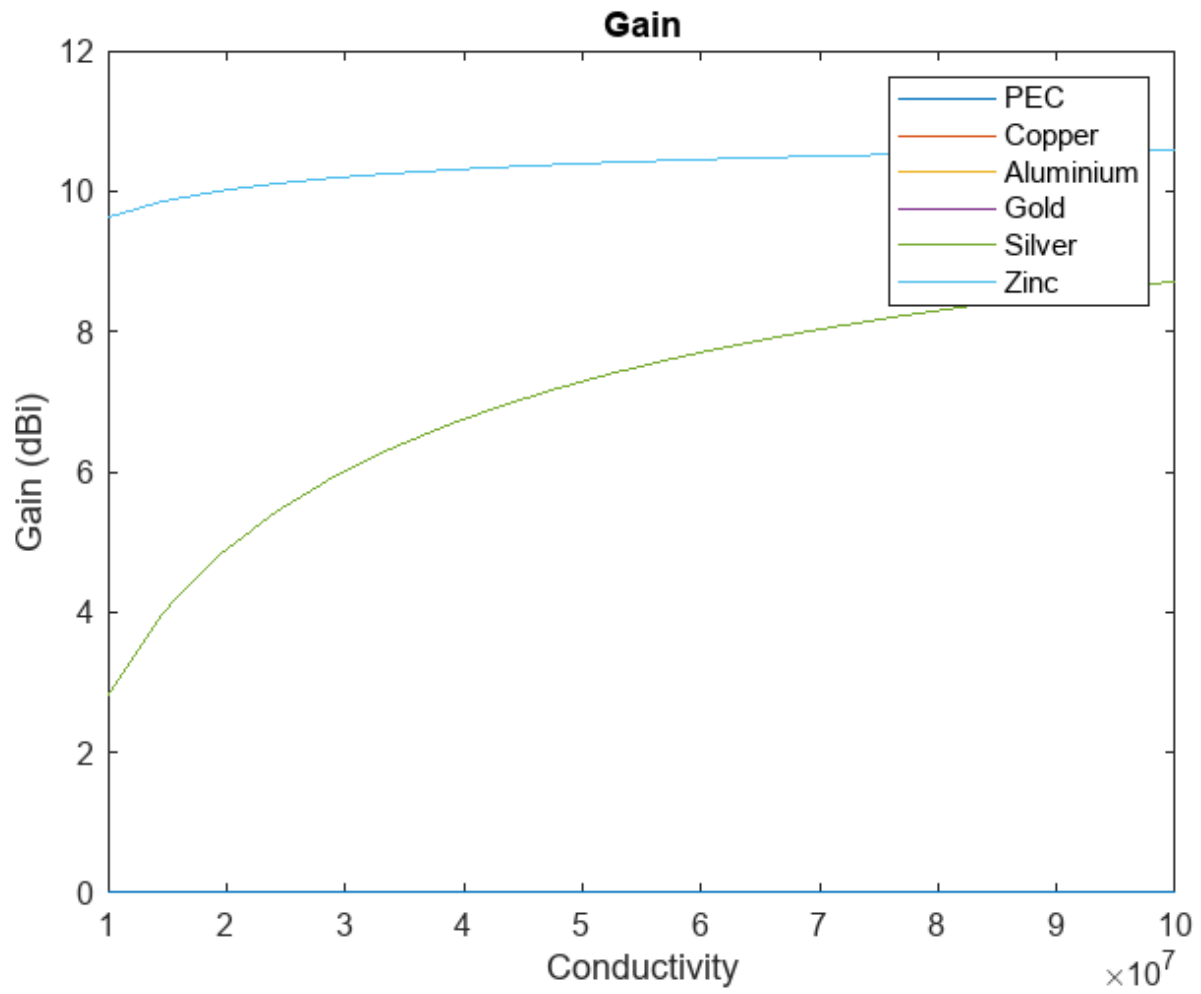
Plot Return Loss and Gain Against Conductivity

Plot the return loss and gain as functions of the conductivity of the conductors. Copper, aluminium, and zinc overlap on both plots. Gold and silver also overlap on both plots.

```
figure
plot(conductivity,loss2)
title("Return Loss")
xlabel("Conductivity (S/m)")
ylabel("Loss (dB)")
legend(metals)
```



```
figure
plot(conductivity,gain2)
title("Gain")
xlabel("Conductivity")
ylabel("Gain (dBi)")
legend(metals)
```



Design, Analyze, and Prototype 2x2 Patch Array Antenna

This example shows how to create a 2x2 patch array on FR4 substrate, analyze the antenna, and generate Gerber files of the PCB for prototyping. The design operates at around 2.4 GHz.

Design Parameters

Set up the dielectric according to [1] and specify the physical constants.

```
freq = 2.4e9;
freqRange = linspace(2e9,3e9);
c = physconst("lightspeed");
d = dielectric("FR4");
d.EpsilonR = 4.3;
d.Thickness = 1.6e-3;
```

Calculate Patch Dimensions

Find the dimensions of the patch microstrip using the equations in [2]. The width and length are based on the effective permittivity ϵ_R and effective wavelength λ_{eff} in the substrate.

```
W = c/(2*freq*sqrt((d.EpsilonR+1)/2));
epsilonEff = (d.EpsilonR+1)/2 + (d.EpsilonR-1)/2/sqrt(1+12*d.Thickness/W);
lambdaEff = c/(freq*sqrt(epsilonEff));
Leff = lambdaEff/2;
deltaL = 0.412*d.Thickness*(epsilonEff+0.3)*(W/d.Thickness+0.264)/(epsilonEff-0.258)/(W/d.Thickness);
L = Leff - 2*deltaL;
```

Create Patch

Create a square patch microstrip antenna element using the ground plane length and width described in [1].

```
GroundPlaneLength = 0.12;
GroundPlaneWidth = 0.12;
patch = patchMicrostrip(Substrate=d, Height=d.Thickness, Length=L, Width=L, ...
    GroundPlaneLength=GroundPlaneLength/2, GroundPlaneWidth=GroundPlaneWidth/2, ...
    FeedOffset=[0,0])
```

```
patch =
    patchMicrostrip with properties:

        Length: 0.0298
        Width: 0.0298
        Height: 0.0016
        Substrate: [1x1 dielectric]
        GroundPlaneLength: 0.0600
        GroundPlaneWidth: 0.0600
        PatchCenterOffset: [0 0]
        FeedOffset: [0 0]
        Conductor: [1x1 metal]
        Tilt: 0
        TiltAxis: [1 0 0]
        Load: [1x1 lumpedElement]
```

Create Array

Set the spacing between the elements in the array to be greater than half of the effective wavelength and create the rectangular array.

```
spacing = lambdaEff*0.6;  
arr = rectangularArray(Element=patch, RowSpacing=spacing, ColumnSpacing=spacing)
```

```
arr =  
    rectangularArray with properties:  
  
        Element: [1x1 patchMicrostrip]  
        Size: [2 2]  
        RowSpacing: 0.0375  
        ColumnSpacing: 0.0375  
        Lattice: 'Rectangular'  
        AmplitudeTaper: 1  
        PhaseShift: 0  
        Tilt: 0  
        TiltAxis: [1 0 0]
```

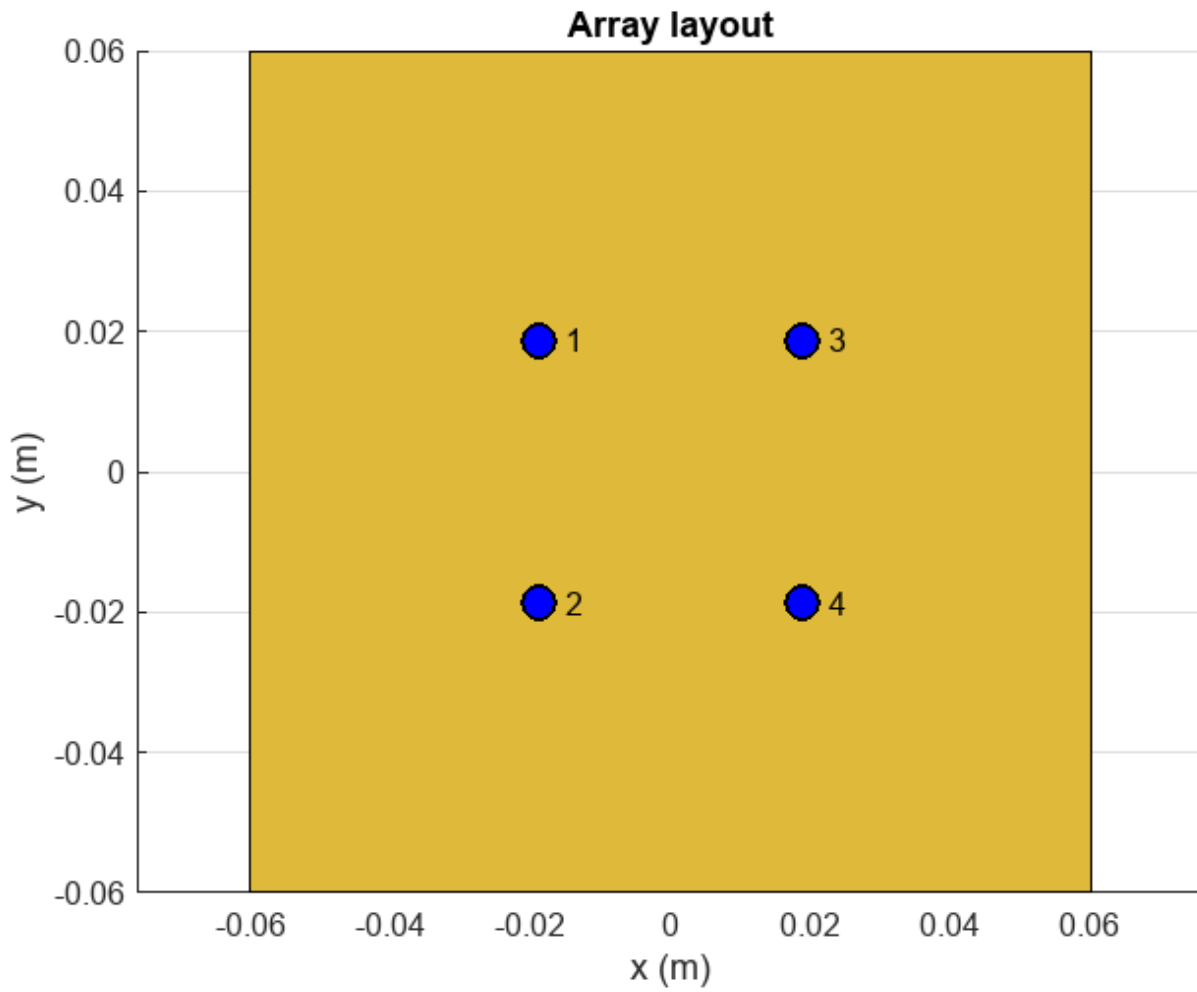
`arr.Element`

```
ans =  
    patchMicrostrip with properties:  
  
        Length: 0.0298  
        Width: 0.0298  
        Height: 0.0016  
        Substrate: [1x1 dielectric]  
        GroundPlaneLength: 0.0600  
        GroundPlaneWidth: 0.0600  
        PatchCenterOffset: [0 0]  
        FeedOffset: [0 0]  
        Conductor: [1x1 metal]  
        Tilt: 0  
        TiltAxis: [1 0 0]  
        Load: [1x1 lumpedElement]
```

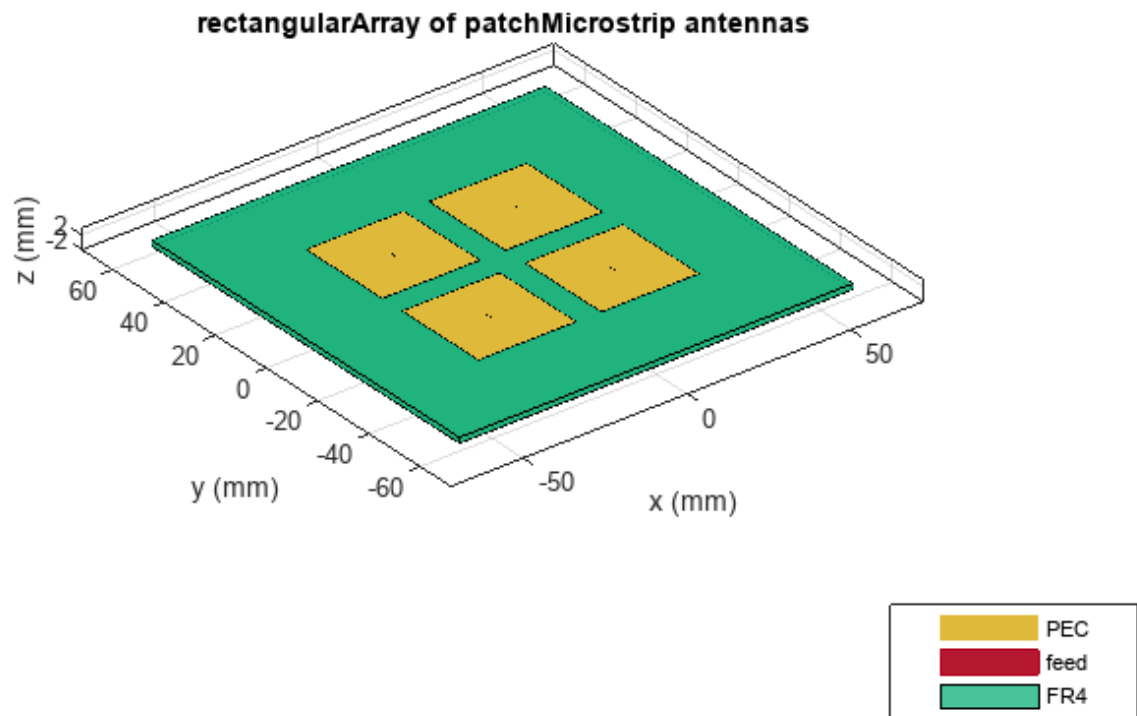
Display the layout and show the array without the traces. You can change this configuration so the bottom of the PCB has only one feedpoint.

```
layout(arr)
```

```
ans=1x2 cell array  
    {1x1 Patch}    {1x1 Patch}
```

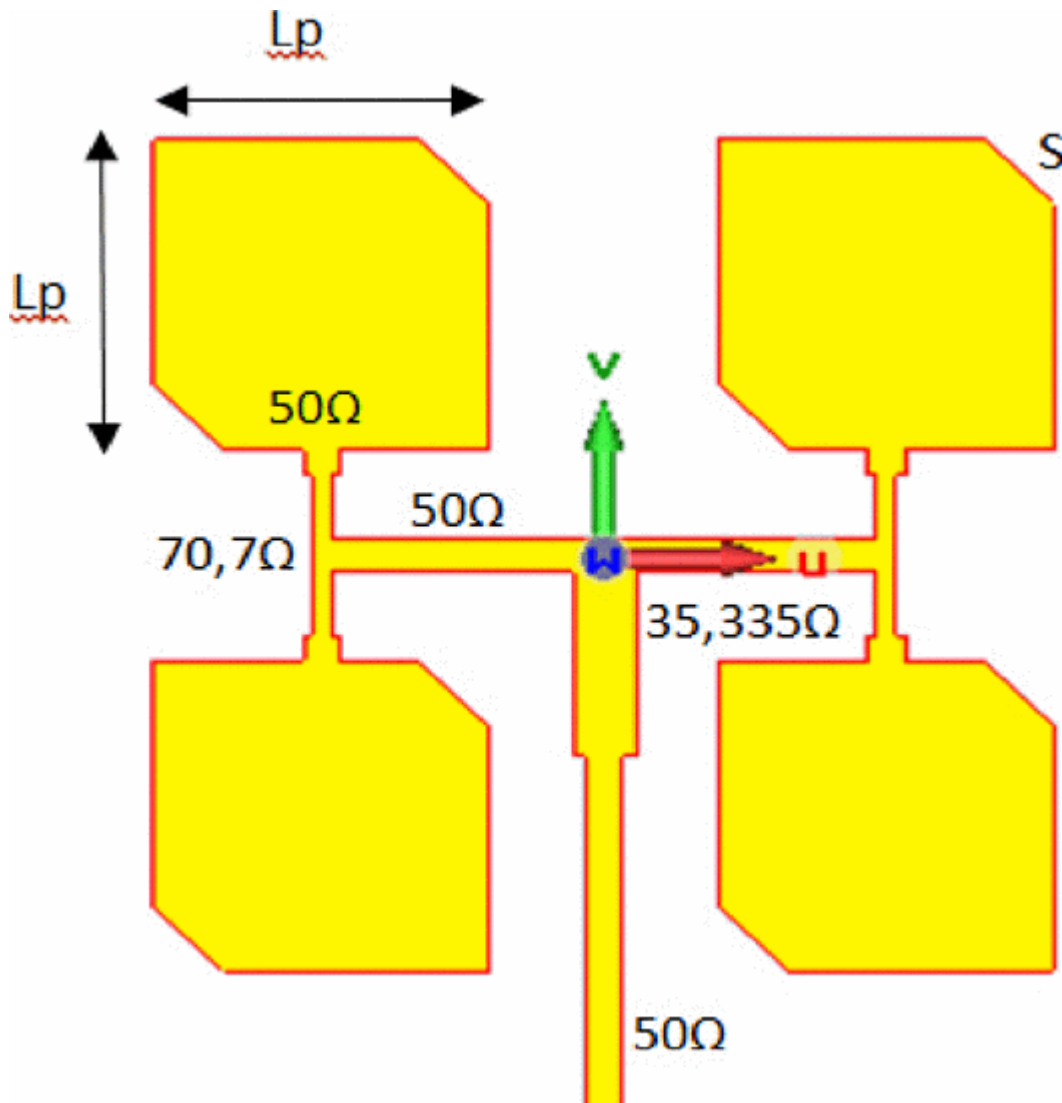


```
figure  
show(arr)
```



Create Feed Trace

Create the trace from [1]. The authors use T-junction Wilkinson power dividers to impose a resistance of 50Ω at the feedpoints and at the edge of the patches. This figure from paper [1] shows the impedances of the traces.



```

z0 = 50;
traceWidth = traceThickness(z0,d)

traceWidth = 0.0031

z1 = real(z0)*sqrt(2);
traceWidth2 = traceThickness(z1,d)

traceWidth2 = 0.0017

z2 = real(z0)/sqrt(2);
traceWidth3 = traceThickness(z2,d)

traceWidth3 = 0.0053

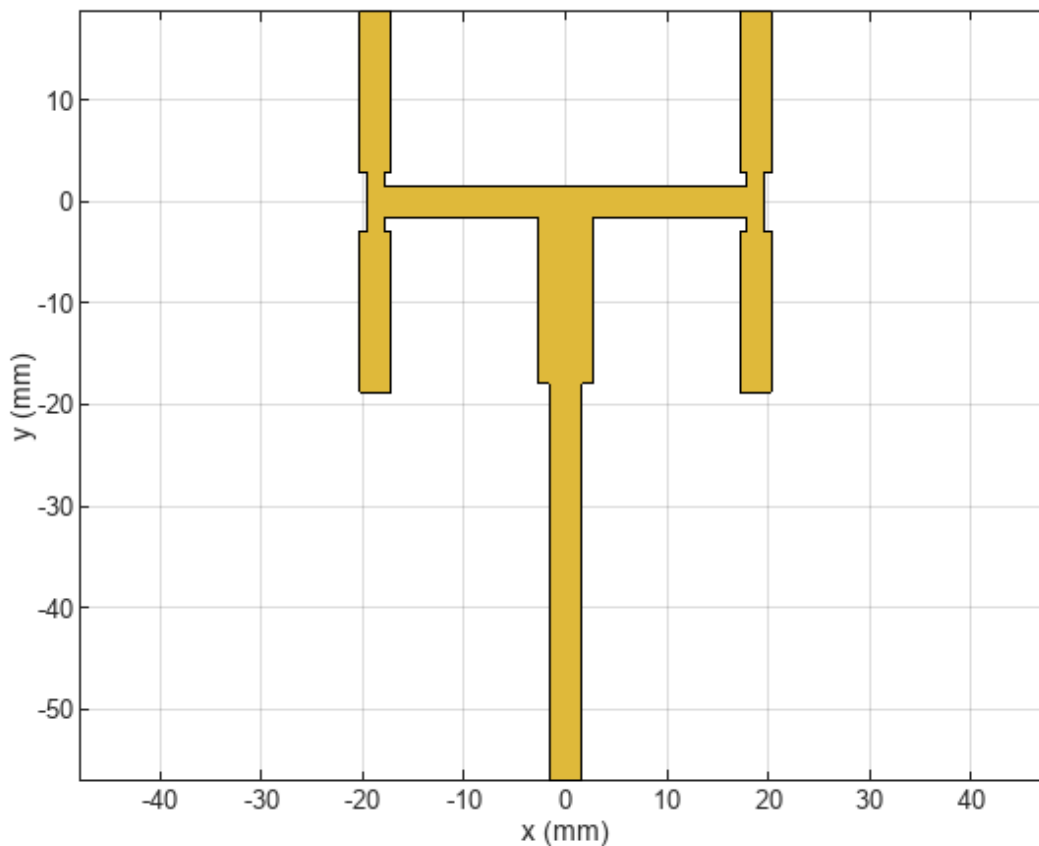
offset = 3e-3;
feedLocation = -arr.GroundPlaneWidth/2 + offset;
x = abs(arr.FeedLocation(1,1));
y = abs(arr.FeedLocation(1,2));

```

```

firstTLength = 2*y;
secondTLength = 2*x + traceWidth2;
feedLength = traceWidth/2 - feedLocation;
feedTraceCenter = feedLocation + feedLength/2;
yLower = y - L/2;
patchFeedLength = L/2 + yLower/4;
patchFeedCenter = y - patchFeedLength/2;
feedTLength = feedLength/3;
feedTCenter = traceWidth/2 - feedTLength/2;
t1 = antenna.Rectangle(Length=traceWidth2, Width=firstTLength, Center=[x 0]);
t2 = antenna.Rectangle(Length=traceWidth2, Width=firstTLength, Center=[-x 0]);
t3 = antenna.Rectangle(Length=secondTLength, Width=traceWidth, Center=[0 0]);
t4 = antenna.Rectangle(Length=traceWidth, Width=feedLength, Center=[0 feedTraceCenter]);
t5 = antenna.Rectangle(Length=traceWidth, Width=patchFeedLength, Center=[x patchFeedCenter]);
t6 = antenna.Rectangle(Length=traceWidth, Width=patchFeedLength, Center=[-x patchFeedCenter]);
t7 = antenna.Rectangle(Length=traceWidth, Width=patchFeedLength, Center=[x -patchFeedCenter]);
t8 = antenna.Rectangle(Length=traceWidth, Width=patchFeedLength, Center=[-x -patchFeedCenter]);
t9 = antenna.Rectangle(Length=traceWidth3, Width=feedTLength, Center=[0 feedTCenter]);
feedTrace = t1 + t2 + t3 + t4 + t5 + t6 + t7 + t8 + t9;
figure(Name="Feed Trace")
show(feedTrace)

```



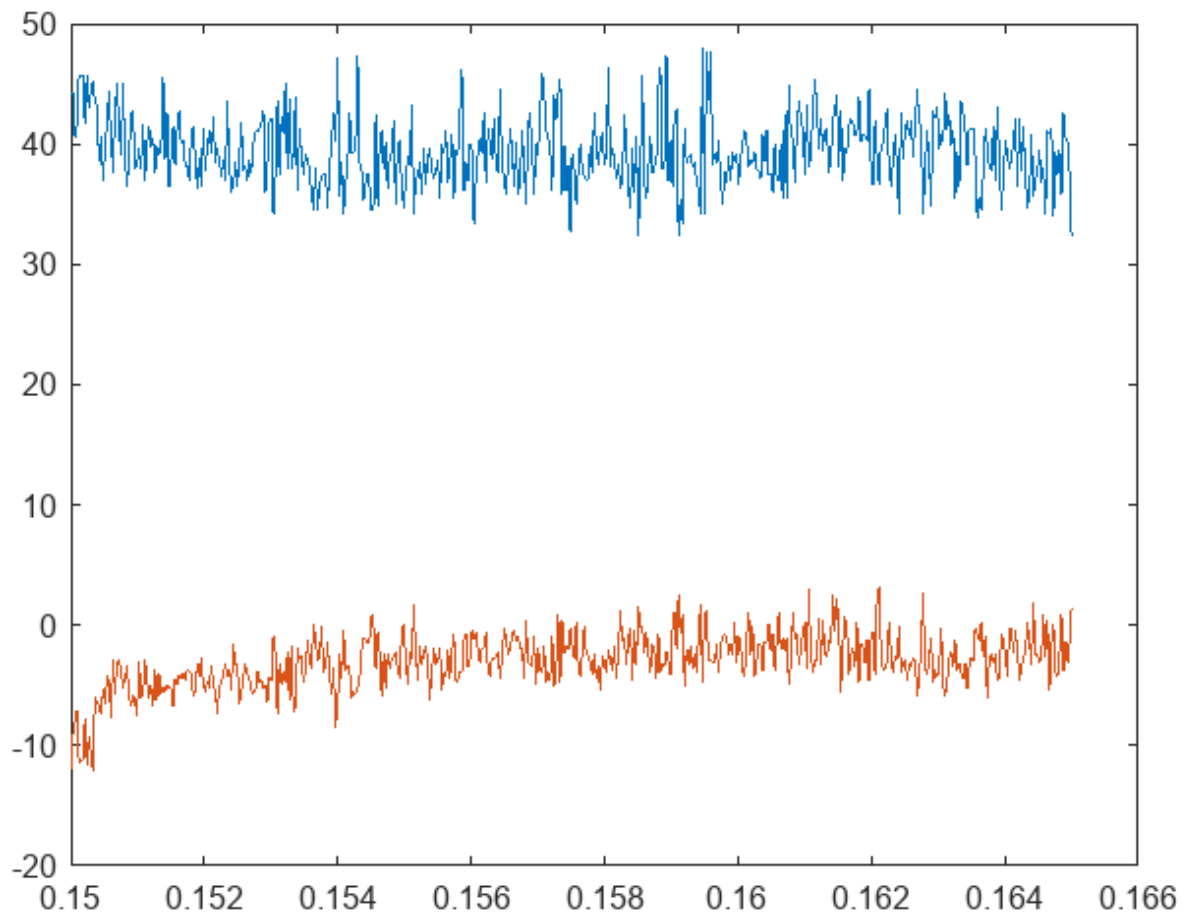
Sweep Side Lengths of Corners

Determine how much to cut off the corners by sweeping through possible side lengths of the triangle to cut and determine the impedance z of the PCB. S represents the ratio of triangle side length to the patch's length, so S is the side length of the triangle that must be cut from the patch. Truncating the corners helps match the impedance to the desired value of 50Ω and the resonant frequency to 2.4 GHz.

The `impedanceSweep` MAT file contains a saved value of the impedance z of a PCB with truncated corners whose side lengths are equal to $S \cdot L$. Because computing the impedances for all the different values of S takes several hours, using the stored value in the MAT file speeds up the example. To calculate the value of z using the `sweepImpedance` function, set the `calculateImpedance` variable to `true`. The `sweepImpedance` function loops through values of S to generate a PCB with truncated corners whose side lengths correspond to those values.

Plot the real and imaginary impedance as functions of S .

```
S = linspace(0.15,0.165,500);
calculateImpedance = false;
if calculateImpedance
    z = sweepImpedance(arr,S);
else
    load("impedanceSweep.mat");
end
figure
plot(S,real(z))
hold on
plot(S,imag(z))
hold off
```



Create Truncated Corners

Find the value of S that results in the PCB having impedance values closest to 50Ω resistance and 0Ω reactance. For this design, cut off isosceles right triangles with leg lengths equal to roughly 16% of the side length of a patch, or 4.7 mm.

```
[zmin,zidx] = min(abs(z-z0))
```

```
zmin = 2.5609
```

```
zidx = 318
```

```
z(zidx)
```

```
ans = 47.6976 + 1.1212i
```

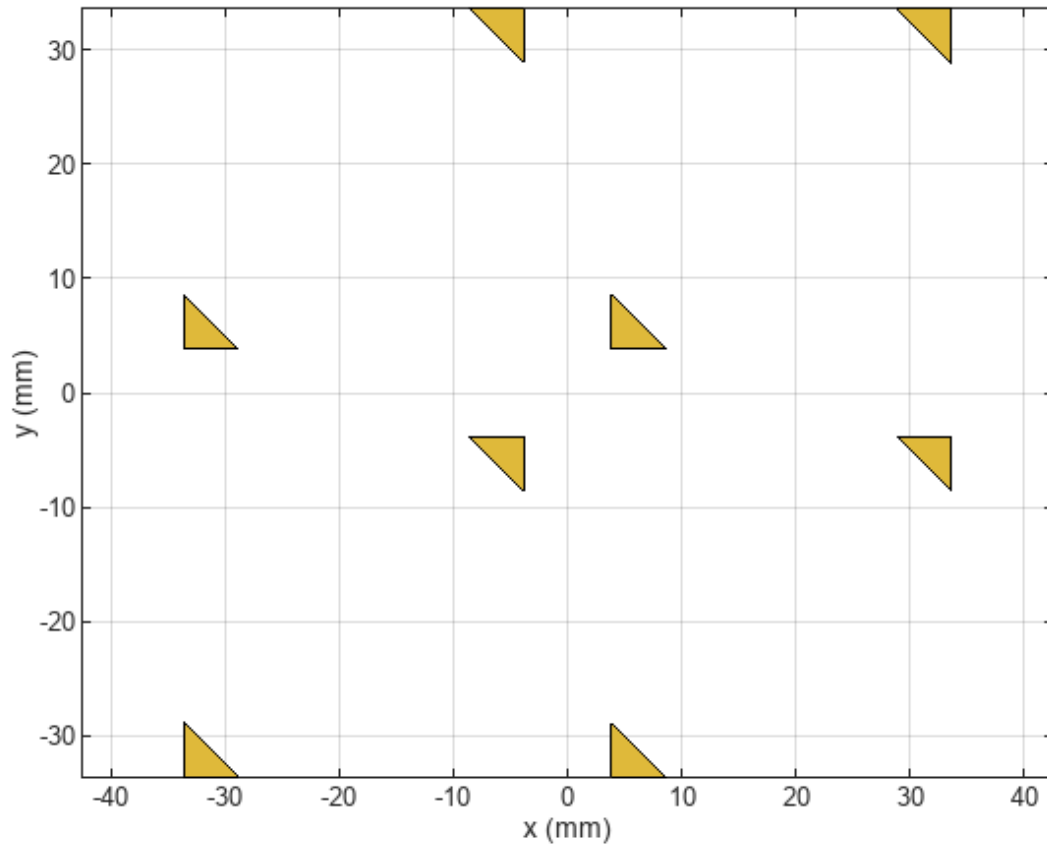
```
S(zidx)
```

```
ans = 0.1595
```

```
s = S(zidx)*L
```

```
s = 0.0047
```

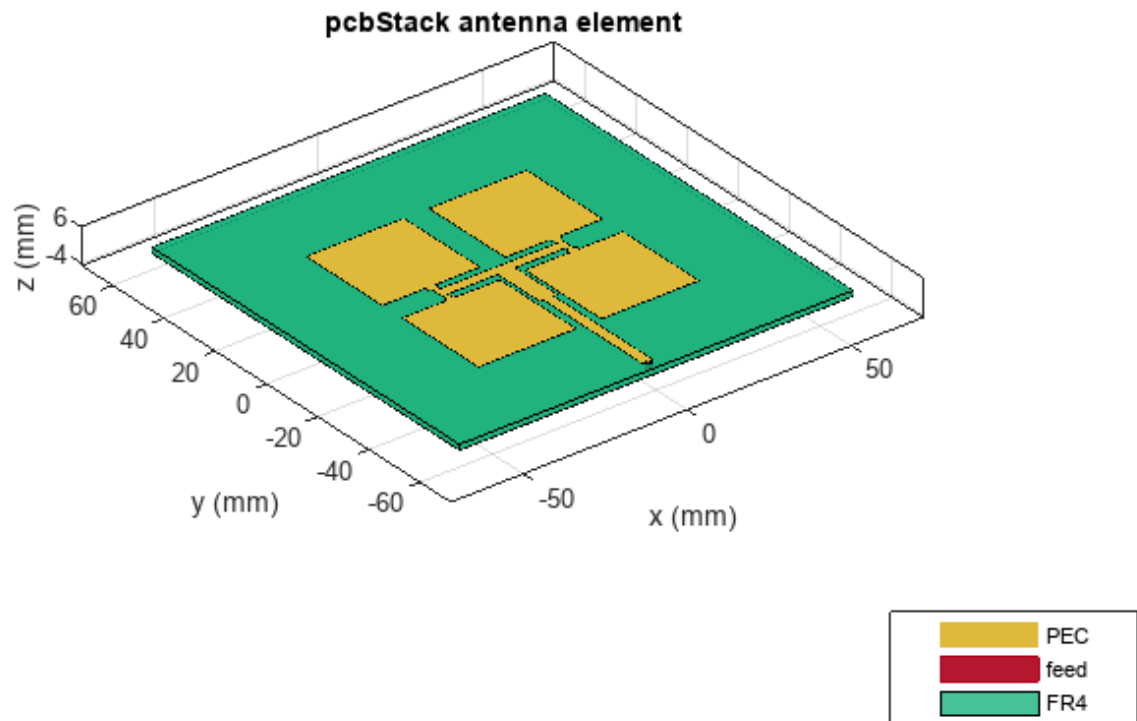
```
truncatedCorners = createTruncatedCorners(arr,s);
show(truncatedCorners)
```



Create PCB Stack Without Truncated Corners

Convert the array to a PCB stack and join the feed trace with the array. Then, set the feed and via locations and properties. Ensure that the PCB dielectric is as big as the groundplane.

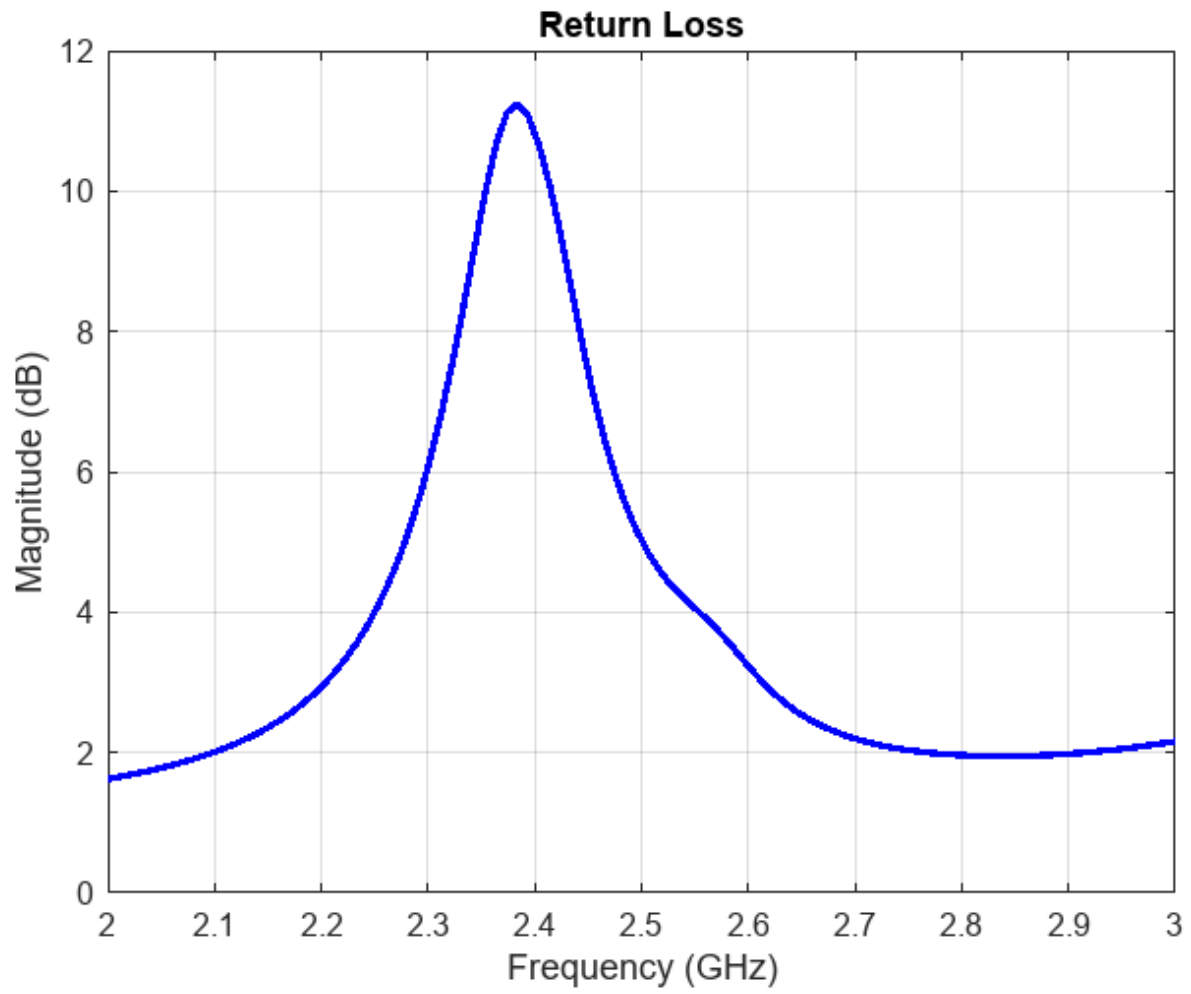
```
arrPCB = pcbStack(arr);
arrPCB.Layers{1,1} = arrPCB.Layers{1,1} + feedTrace;
arrPCB.FeedLocations = [0 feedLocation 1 3];
arrPCB.ViaLocations = arrPCB.FeedLocations(1,:);
arrPCB.FeedDiameter = traceWidth/2;
arrPCB.ViaDiameter = arrPCB.FeedDiameter;
arrPCB.Layers{1,2}.Length = GroundPlaneLength;
arrPCB.Layers{1,2}.Width = GroundPlaneWidth;
figure;
show(arrPCB);
```



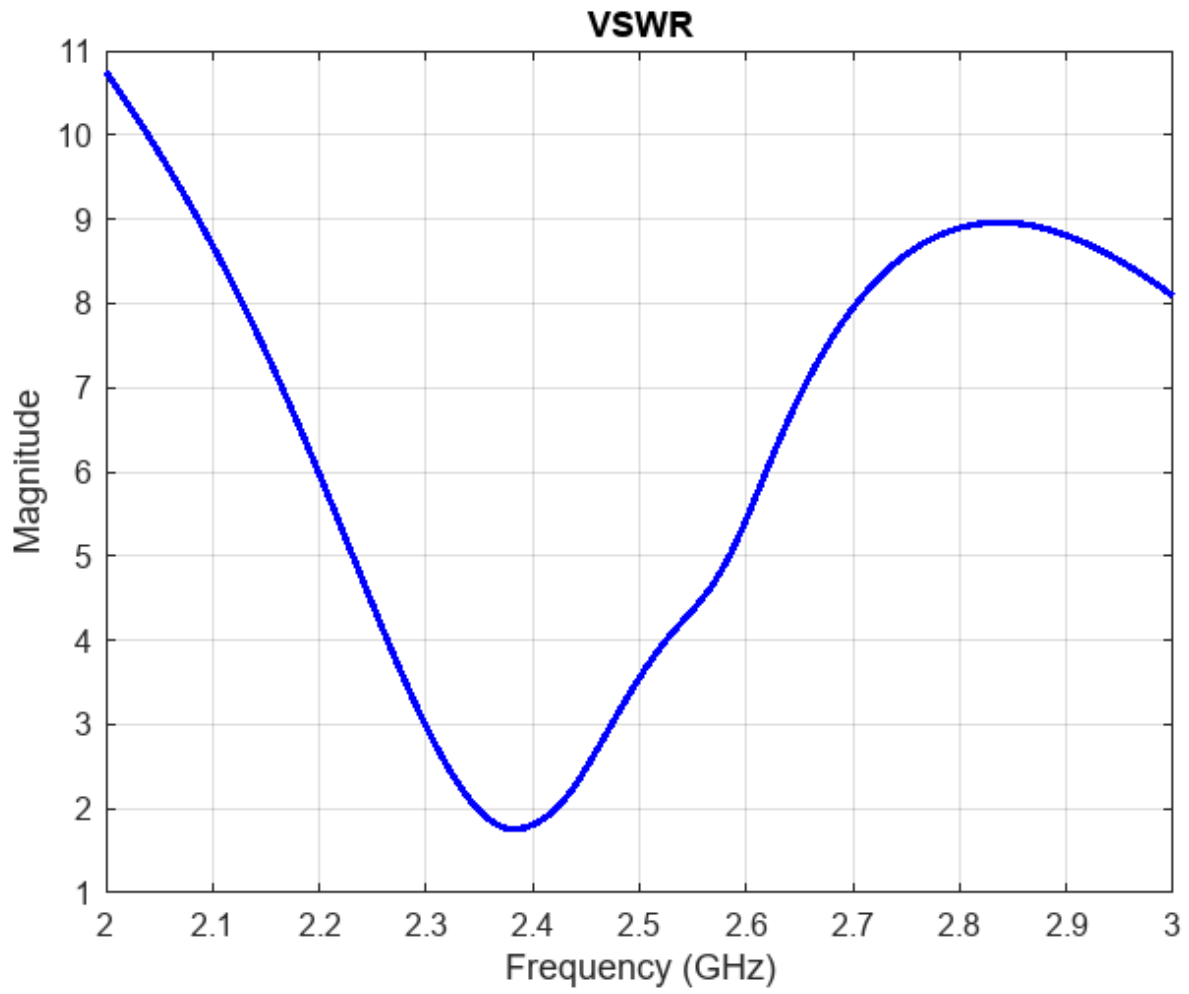
Analyze PCB Stack Without Truncated Corners

Analyze the PCB stack of the array that does not have truncated corners. Determine the return loss, the voltage standing wave ratio, the impedance, and the S-parameters.

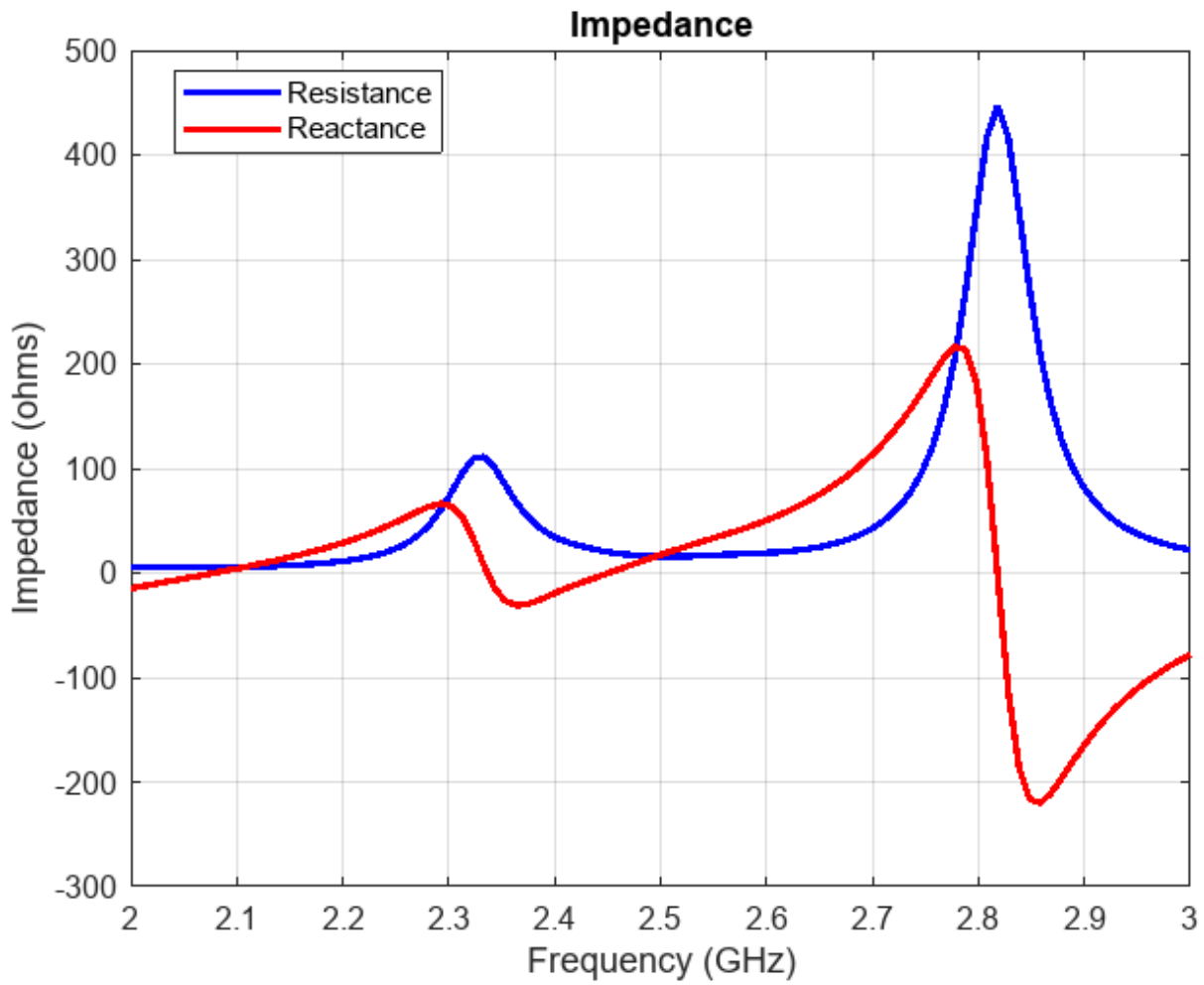
```
returnLoss(arrPCB, freqRange, z0);
```



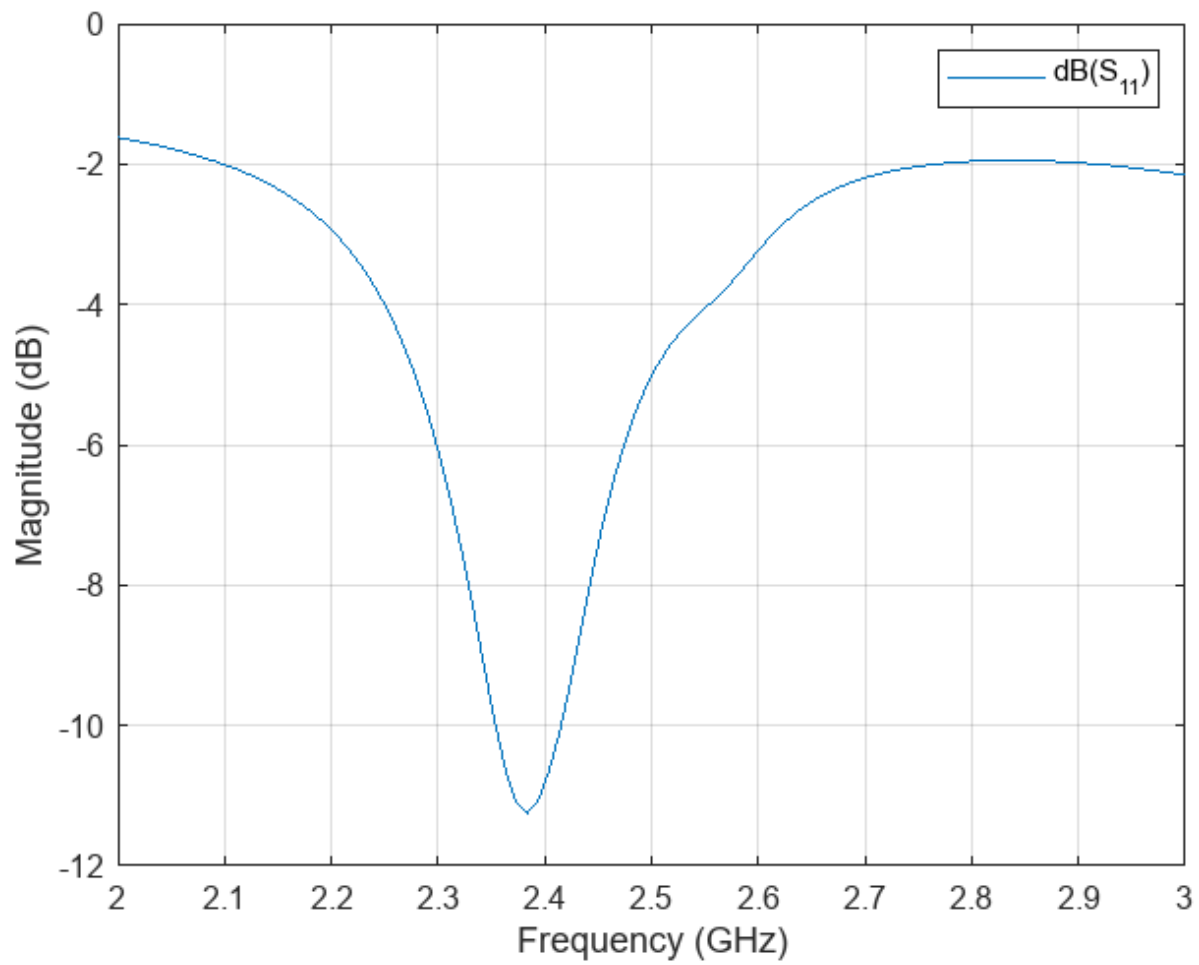
```
vswr(arrPCB, freqRange, z0);
```



```
impedance(arrPCB, freqRange);
```

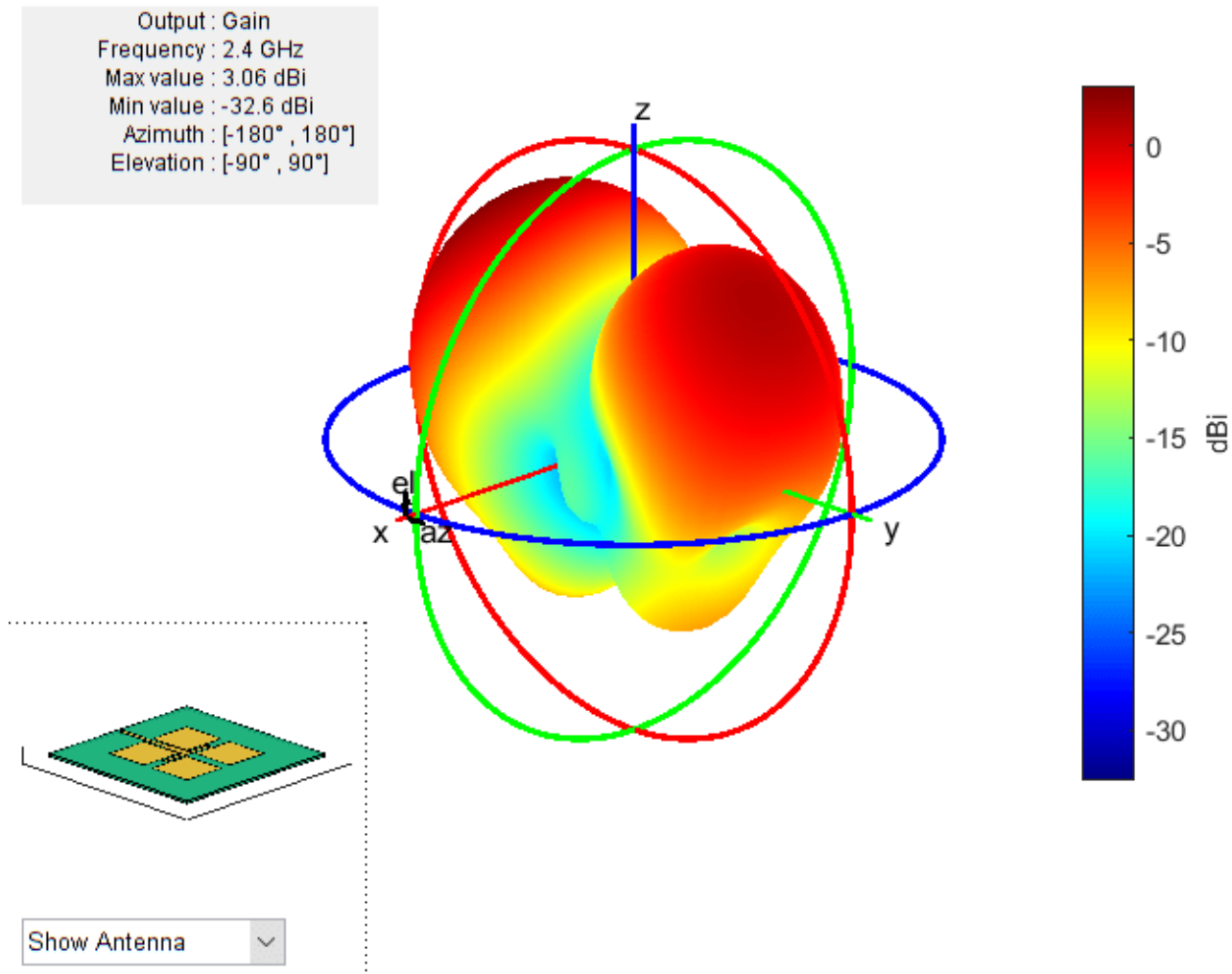



```
spar = sparameters(arrPCB, freqRange, z0);  
rfplot(spar)
```



Show the PCB radiation pattern at 2.4 GHz.

```
pattern(arrPCB, freq)
```

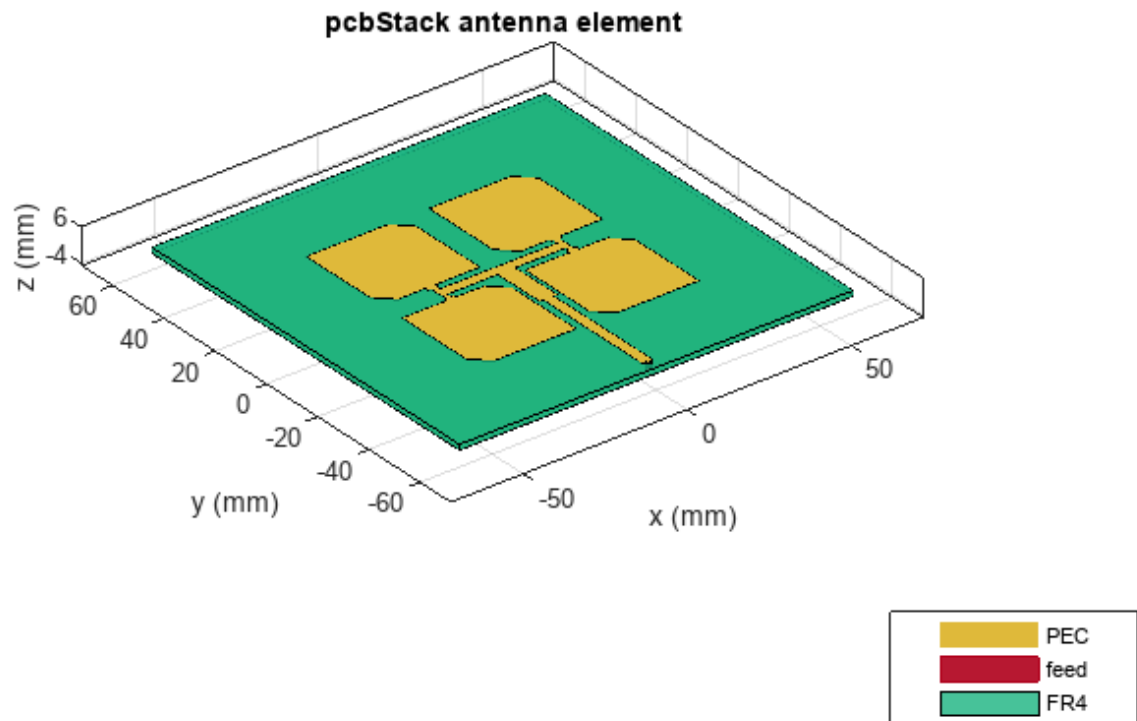


Create PCB Stack with Truncated Corners

Convert the array to a PCB stack and join the feed trace with the array. Truncate the corners from the patches. Then, set the feed and via locations and properties. Ensure that the PCB's dielectric is as big as the groundplane.

```

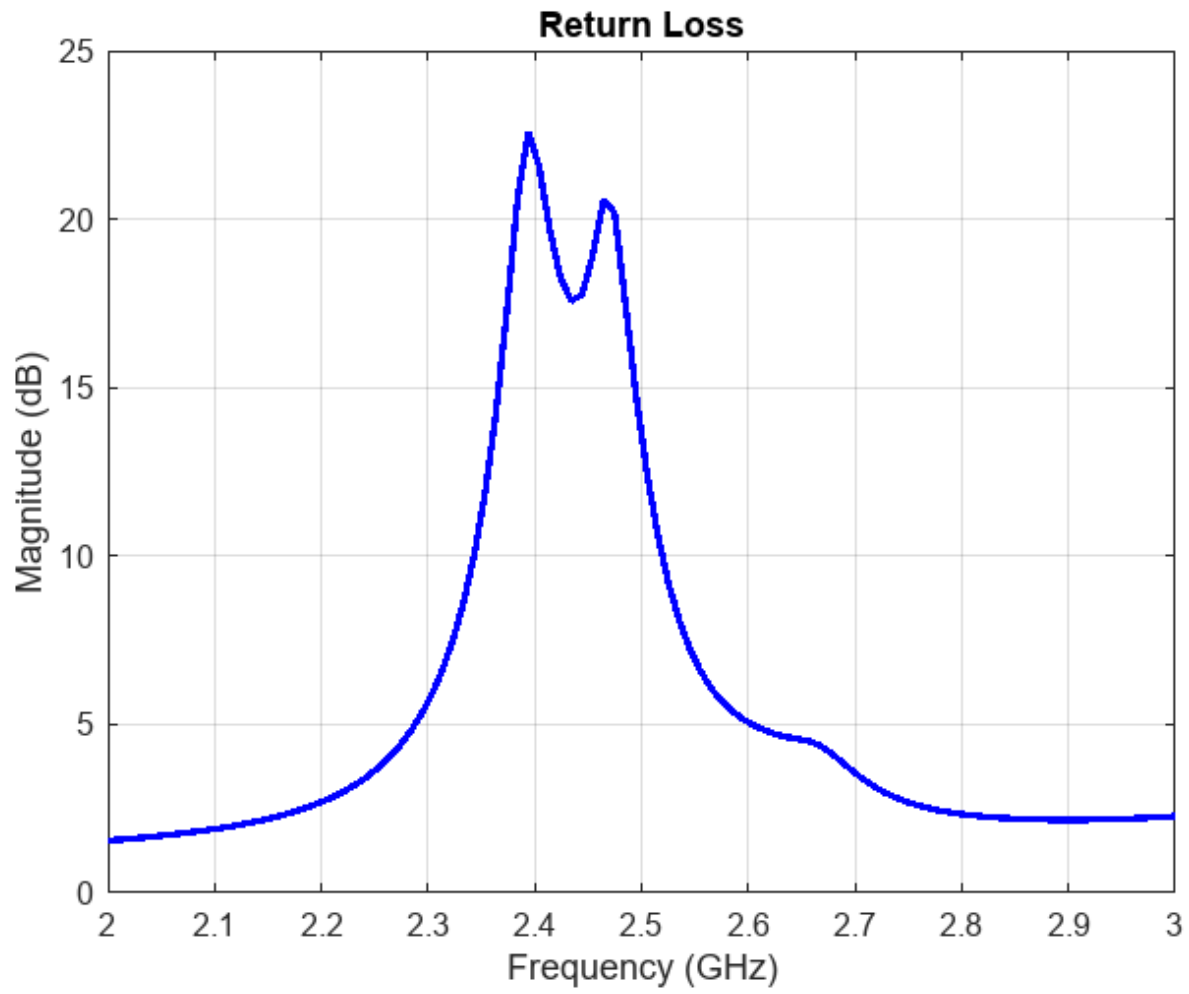
arrPCB_tc = pcbStack(arr);
arrPCB_tc.Layers{1,1} = arrPCB_tc.Layers{1,1} + feedTrace - truncatedCorners;
arrPCB_tc.FeedLocations = [0 feedLocation 1 3];
arrPCB_tc.ViaLocations = arrPCB_tc.FeedLocations(1,:);
arrPCB_tc.FeedDiameter = traceWidth/2;
arrPCB_tc.ViaDiameter = arrPCB_tc.FeedDiameter;
arrPCB_tc.Layers{1,2}.Length = GroundPlaneLength;
arrPCB_tc.Layers{1,2}.Width = GroundPlaneWidth;
figure;
show(arrPCB_tc);
  
```



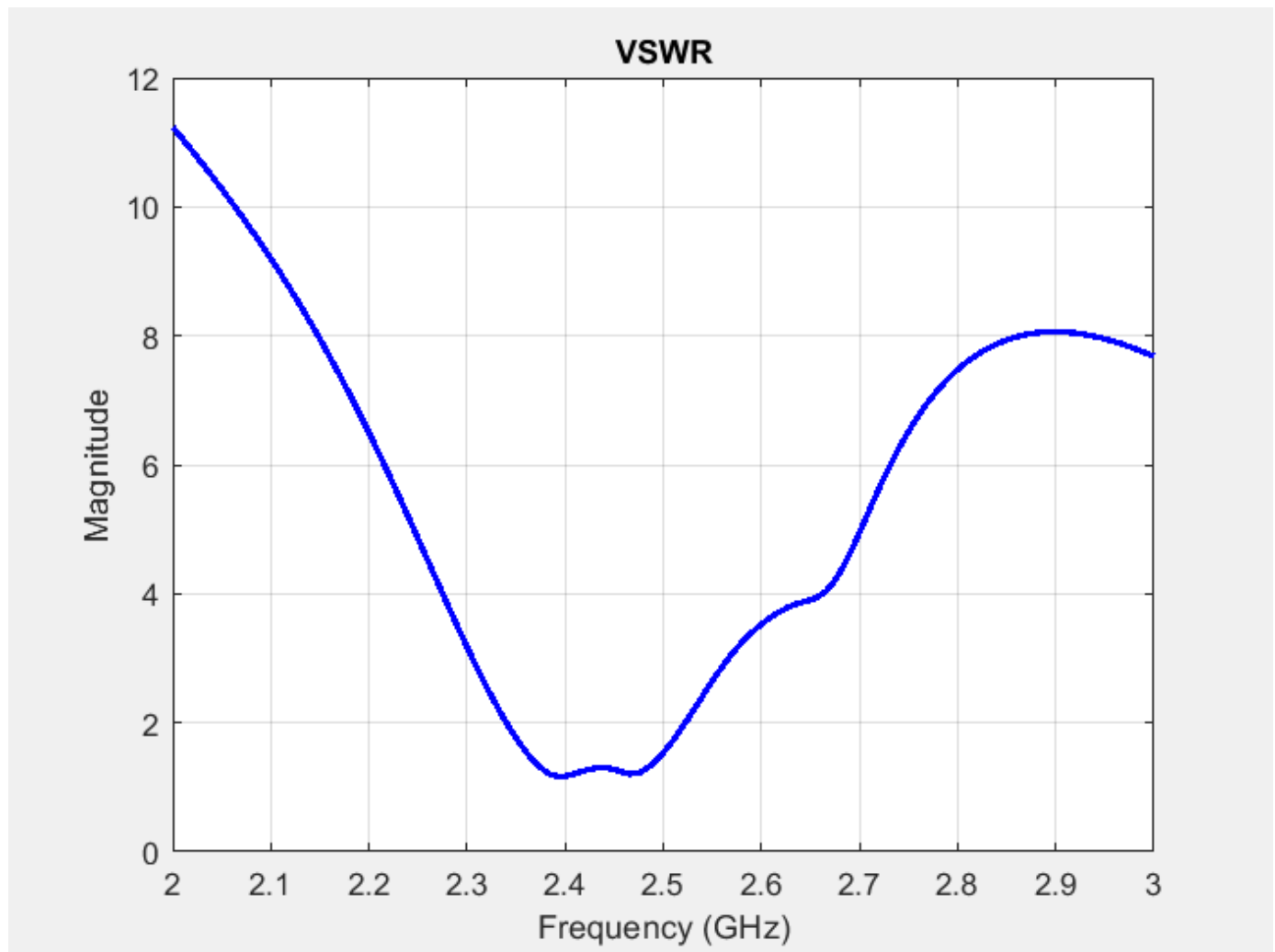
Analyze PCB Stack with Truncated Corners

Analyze the PCB Stack of the array with truncated corners. Determine the return loss, the voltage standing wave ratio, the impedance, and the S-parameters.

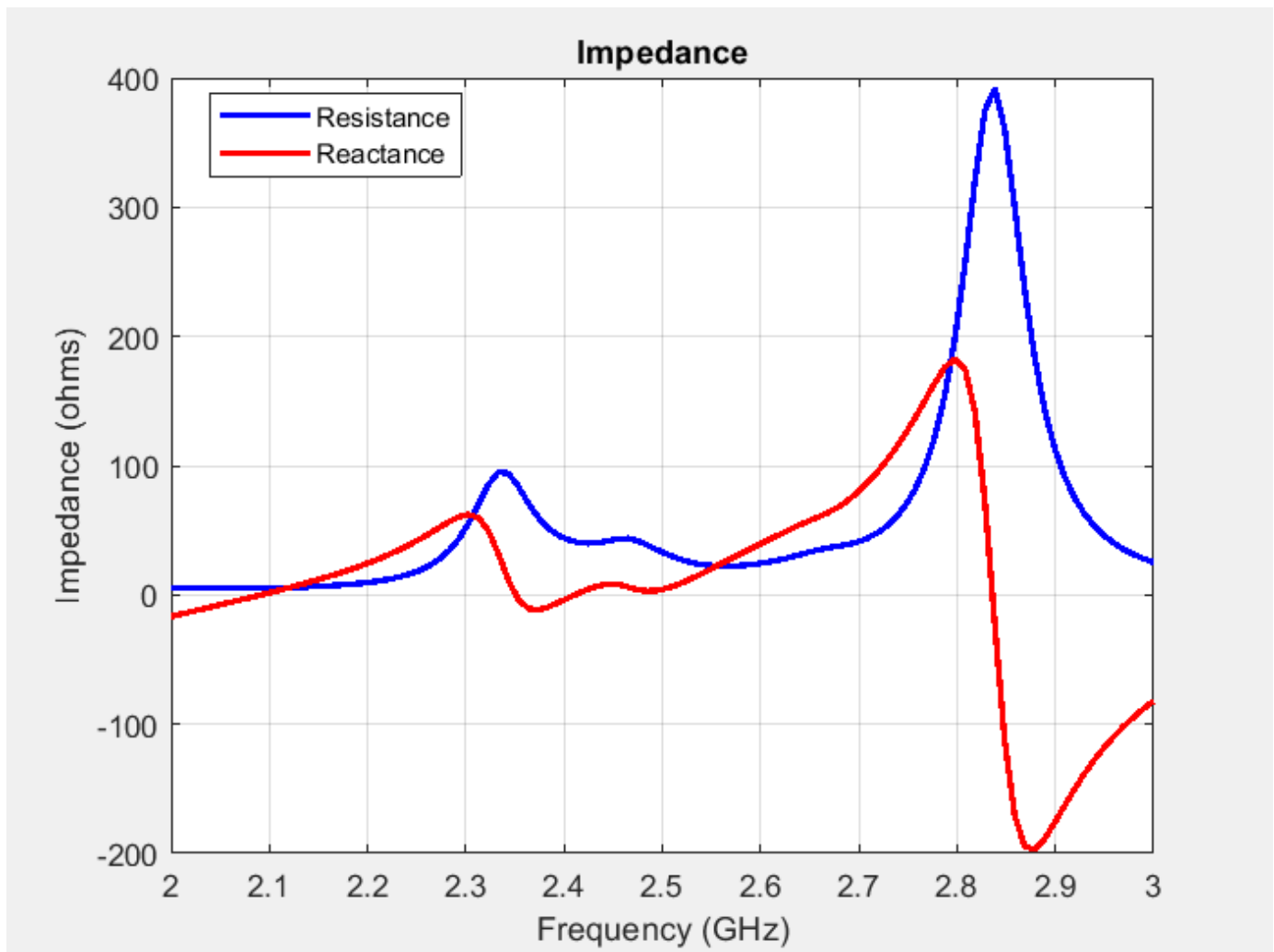
```
returnLoss(arrPCB_tc, freqRange, z0);
```



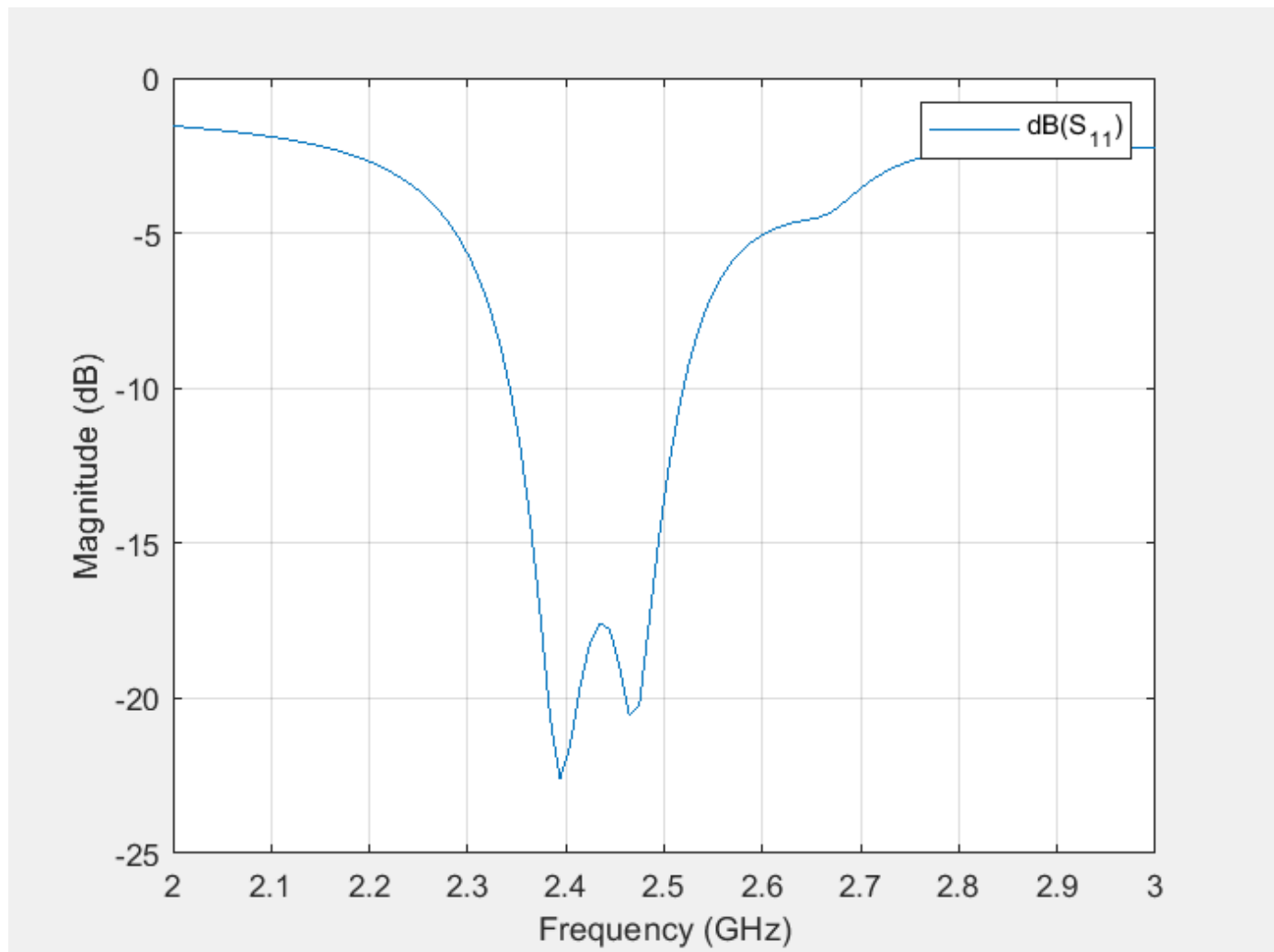
```
vswr(arrPCB_tc, freqRange, z0);
```



```
impedance(arrPCB_tc, freqRange);
```

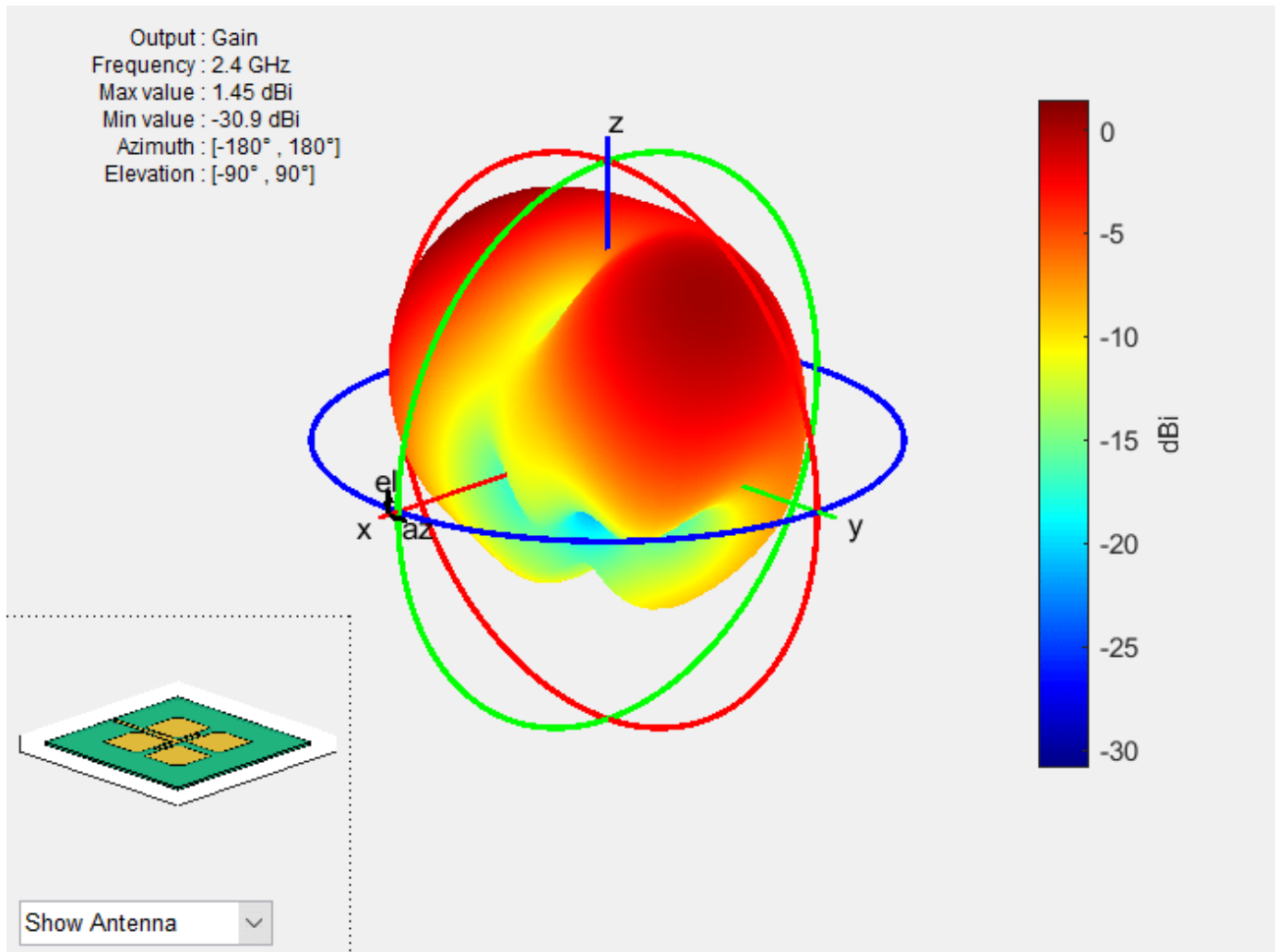


```
spar = sparameters(arrPCB_tc, freqRange, z0);  
rfplot(spar)
```



Show the PCB radiation pattern at 2.4 GHz.

```
pattern(arrPCB_tc, freq)
```

Generate Gerber Files

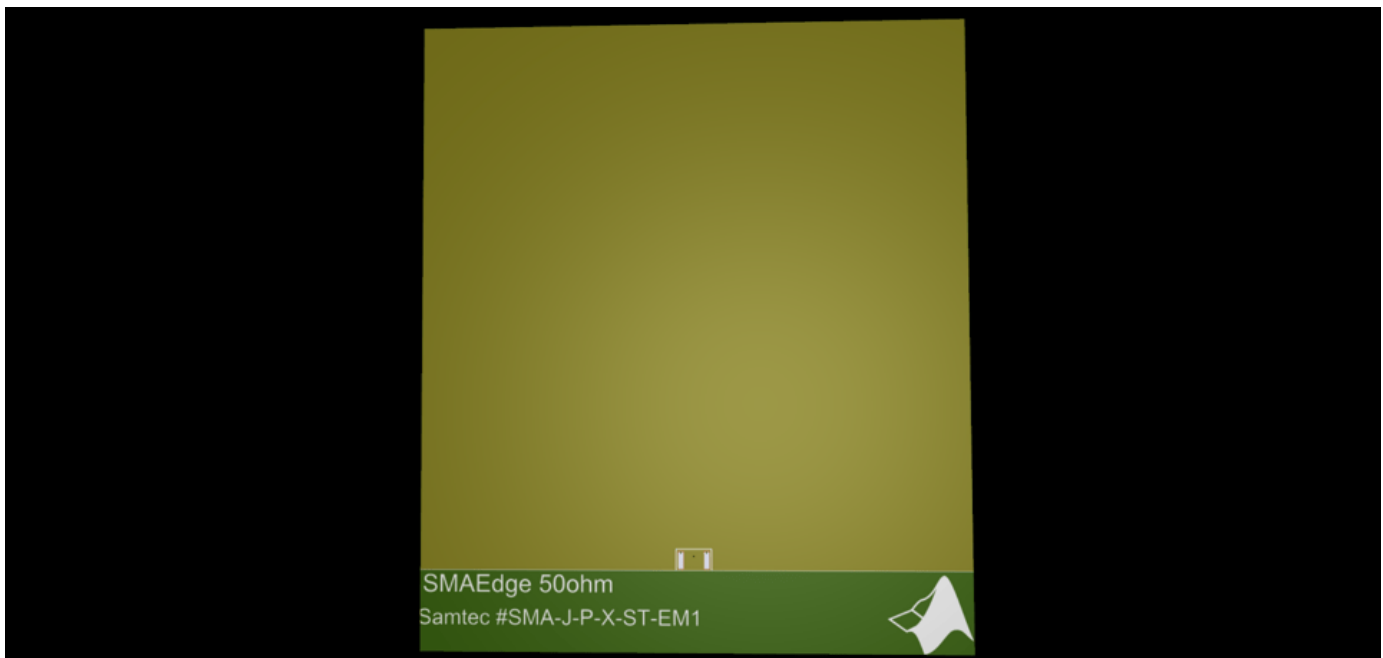
You can use Gerber files to export the geometry information of a PCB. Use a PCB writer and an RF connector to generate the Gerber files.

```
W = PCBServices.OSHParkWriter;
W = PCBServices.MayhewWriter;
W.Filename = 'patch array 2x2';
C = SMAEdge_SamtecCustom;
C.EdgeLocation = "south";
C.ExtendBoardProfile = false;
Am = PCBWriter(arrPCB_tc,W,C);
gerberWrite(Am)
```

If you have an internet connection, a browser window opens and displays the Mayhewlabs free 3D online Gerber viewer. After you drag and drop the files into the Mayhewlabs viewer, the software renders the PCB design. This figure shows the front view of the board.



This figure shows the back view of the board.



Conclusion

In this example, you create a 2x2 patch array on FR4, analyze the antenna, and generate Gerber files of the PCB for prototyping.

Without truncated corners, the array has a single resonant frequency at 2.38 GHz. The reactance reaches 0Ω at around 2.45 GHz.

With the truncated corners, the array has multiple frequencies with $S_{11} \leq -10$ dB in the S-parameter plot, one minimum at 2.39 GHz, and another minimum at 2.48 GHz. The reactance reaches 0Ω at around 2.41 GHz.

Supporting Functions

Determine Trace Widths

The `traceThickness` function determines the thickness of the traces on the PCB for input impedance and dielectric based on the equation from [1].

```
function width = traceThickness(z,d)
    A = z/60*sqrt((d.EpsilonR+1)/2)+(d.EpsilonR-1)/(d.EpsilonR+1)*(0.23+0.11/d.EpsilonR);
    width = d.Thickness*8*exp(A)/(exp(2*A)-2);
end
```

Create the Truncated Corners

The `truncatedCorners` function creates the truncated corners for the input array and leg length of an isosceles right triangle.

```
function truncatedCorners = createTruncatedCorners(arr,s)
    x11 = -arr.ColumnSpacing/2 + arr.Element.Length/2;
    y11 = arr.RowSpacing/2 + arr.Element.Width/2;
    x12 = -arr.ColumnSpacing/2 - arr.Element.Length/2;
    y12 = arr.RowSpacing/2 - arr.Element.Width/2;
    s1 = antenna.Polygon(Vertices=[x11-s y11 0; x11 y11-s 0; x11 y11 0]);
    s2 = antenna.Polygon(Vertices=[x12+s y12 0; x12 y12+s 0; x12 y12 0]);

    x21 = x11;
    y21 = -arr.RowSpacing/2+arr.Element.Width/2;
    x22 = x12;
    y22 = -arr.RowSpacing/2-arr.Element.Width/2;
    s3 = antenna.Polygon(Vertices=[x21-s y21 0; x21 y21-s 0; x21 y21 0]);
    s4 = antenna.Polygon(Vertices=[x22+s y22 0; x22 y22+s 0; x22 y22 0]);

    x31 = arr.ColumnSpacing/2 + arr.Element.Length/2;
    y31 = y11;
    x32 = arr.ColumnSpacing/2 - arr.Element.Length/2;
    y32 = y12;
    s5 = antenna.Polygon(Vertices=[x31-s y31 0; x31 y31-s 0; x31 y31 0]);
    s6 = antenna.Polygon(Vertices=[x32+s y32 0; x32 y32+s 0; x32 y32 0]);

    x41 = x31;
    y41 = y21;
    x42 = x32;
    y42 = y22;
    s7 = antenna.Polygon(Vertices=[x41-s y41 0; x41 y41-s 0; x41 y41 0]);
    s8 = antenna.Polygon(Vertices=[x42+s y42 0; x42 y42+s 0; x42 y42 0]);

    truncatedCorners = s1 + s2 + s3 + s4 + s5 + s6 + s7 + s8;
end
```

Calculate Impedance

For each element of the ratio S of truncated corner side length to patch length, the `sweepImpedance` function creates the truncated corners. The function then creates the PCB stack of the array, adds the feed trace, and removes the corners. You specify the feed and via locations and properties, ensuring that the PCB's dielectric is as big as the groundplane. Finally, the function computes the impedance of the PCB.

```
function z = sweepImpedance(arr,S)
    z = zeros(size(S));
    for i = 1:length(S)
        s = S(i)*L;
        truncatedCorners = createTruncatedCorners(arr,s);

        pcb = pcbStack(arr);
        pcb.Layers{1,1} = pcb.Layers{1,1} + feedTrace - truncatedCorners;
        pcb.FeedLocations = [0 feedLocation 1 3];
        pcb.ViaLocations = pcb.FeedLocations(1,:);
        pcb.FeedDiameter = traceWidth/2;
        pcb.ViaDiameter = pcb.FeedDiameter;
        pcb.Layers{1,2}.Length = GroundPlaneLength;
        pcb.Layers{1,2}.Width = GroundPlaneWidth;

        z(i) = impedance(pcb,freq);
    end
end
```

Reference

[1] Muludi, Zainal, and Budi Aswoyo. "Truncated Microstrip Square Patch Array Antenna 2×2 Elements with Circular Polarization for S-Band Microwave Frequency." In 2017 International Electronics Symposium on Engineering Technology and Applications (IES-ETA), 87-92. Surabaya: IEEE, 2017. <https://doi.org/10.1109/ELECSYM.2017.8240384>.

[2] Balanis, Constantine A. *Antenna Theory: Analysis and Design*. Fourth edition. Hoboken, New Jersey: Wiley, 2016.

Copyright 2023 The MathWorks, Inc.

Modified Sierpinski Monopole Fractal Antenna for Dual-Band Application

This example shows how to model and analyze a second-order modified Sierpinski fractal antenna in a monopole configuration using Antenna Toolbox™ software. A single triangular Sierpinski cell is the basic building block of the Sierpinski fractal antenna. In this example, you construct a second order modified Sierpinski fractal antenna by fractalizing a single Sierpinski cell embedded with a diamond shape in two iterations. The triangle and the diamond have the same ratios throughout the process. The modified design has a reduced footprint in terms of the overall dimensions and the quantity of metal conductor compared to a normal second-order Sierpinski fractal antenna. The order of fractalization determines the number of resonant frequencies of the antenna.

The antenna you design in this example can be used in applications that require a dual band operation. You use the dimensions specified in [1] for the design to model the antenna on an FR4 substrate with a dielectric constant of 4.4, a loss tangent of 0.025, and height of 1.5 mm. You then analyze the antenna for operation in the S and C bands.

Create Antenna Model

The antenna geometry consists of three triangular shapes, each with a side length of 30.23 mm and nine diamond shapes. Use `antenna.Triangle` to create these shapes. First, create the three triangular shapes.

```
side = 30.23e-3;
tol = 0.95;
S = [side side/2 side/4];
shapeTriangle = cell(1,5);
thetaA = zeros(1,3);
for i = 1:3
    if i == 1
        shapeTriangle{i} = antenna.Triangle(Side=S(1,i));
    else
        shapeTriangle{i} = antenna.Triangle(Side=0.95*S(1,i));
    end
    a = S(1,i);
    b = a;
    c = a;
    X = (c^2-a^2-b^2)/(-2*a*b);
    thetaA(1,i) = acosd(X);
    shapeTriangle{i}.StartVertex = [-S(1,i)/2+(0.05*S(1,i))/2 -sind(thetaA(1,i))*b];
end
shapeTriangle{1}.StartVertex = [-S(1,1)/2 -sind(thetaA(1,1))*S(1,1)];
shapeTriangle{2} = mirrorX(shapeTriangle{2});
shapeTriangle{2} = translate(shapeTriangle{2},[0 -2*sind(thetaA(1,2))*(S(1,2)) 0]);
shapeTriangle{3} = mirrorX(shapeTriangle{3});
shape3 = copy(shapeTriangle{3});
shapeTriangle{3} = translate(shapeTriangle{3},[0 -2*sind(thetaA(1,3))*(S(1,3)) 0]);
shapeTriangle{4} = translate(copy(shape3),[-S(1,3) -4*sind(thetaA(1,3))*(S(1,3)) 0]);
shapeTriangle{5} = translate(copy(shape3),[S(1,3) -4*sind(thetaA(1,3))*(S(1,3)) 0]);

triangle = (shapeTriangle{2} + shapeTriangle{3} + shapeTriangle{4} + shapeTriangle{5});
```

Then, create the nine diamond shapes.

```

diamSide = [1.17e-3 2.0742e-3 2.0742e-3];
diaHalf1 = antenna.Triangle(Side=diamSide);
diaHalf2 = mirrorX(copy(diaHalf1));

diaShape = rotateZ(diaHalf1+diaHalf2,90);

DiaShape1 = translate(copy(diaShape),[0 -0.75*sind(thetaA(1,3))*S(1,3) 0]);
DiaShape2 = translate(copy(diaShape),[-S(1,3)/2 -1.75*sind(thetaA(1,3))*S(1,3) 0]);
DiaShape3 = translate(copy(diaShape),[S(1,3)/2 -1.75*sind(thetaA(1,3))*S(1,3) 0]);

DiaShape4 = translate(copy(diaShape),[-S(1,3) -2.75*sind(thetaA(1,3))*S(1,3) 0]);
DiaShape5 = translate(copy(diaShape),[-S(1,3)/2 -3.75*sind(thetaA(1,3))*S(1,3) 0]);
DiaShape6 = translate(copy(diaShape),[-1.5*S(1,3) -3.75*sind(thetaA(1,3))*S(1,3) 0]);

DiaShape7 = translate(copy(diaShape),[S(1,3) -2.75*sind(thetaA(1,3))*S(1,3) 0]);
DiaShape8 = translate(copy(diaShape),[S(1,3)/2 -3.75*sind(thetaA(1,3))*S(1,3) 0]);
DiaShape9 = translate(copy(diaShape),[1.5*S(1,3) -3.75*sind(thetaA(1,3))*S(1,3) 0]);

DiamondShape = DiaShape1 + DiaShape2 + DiaShape3 + DiaShape4 + DiaShape5 + DiaShape6 + DiaShape7

```

Perform boolean operations on these shapes to create the antenna geometry.

```

CustomFractal = shapeTriangle{1} - (triangle+DiamondShape);
CustomFractal = mirrorX(CustomFractal);

```

Create the feed line and add it to the antenna geometry.

```

feedL = 16.5e-3;
feedW = 0.4e-3;
feed = antenna.Rectangle(Length=feedW, Width=feedL, Center=[0,-7.5e-3]);

radiator = CustomFractal + feed;
radiator = translate(radiator,[0 -7e-3 0]);

```

Define the ground plane and board dimensions.

```

gndL = 50e-3;
gndW = 15.50e-3;
BoardL = 50e-3;
BoardW = 45.50e-3;

gndPlane = antenna.Rectangle(Length=gndL, Width=gndW, Center=[0 -BoardW/2+gndW/2]);
boardShape = antenna.Rectangle(Length=BoardL, Width=BoardW);

```

Specify the height of the substrate.

```

h = 1.5e-3;
substrate = dielectric(Name="FR4", EpsilonR=4.4, LossTangent=0.025, Thickness=h);

```

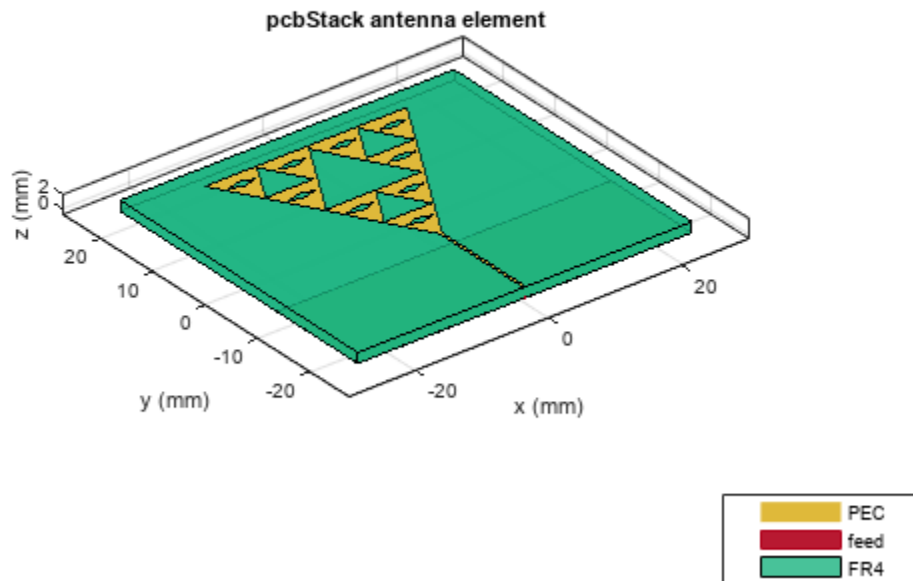
Convert the antenna geometry into a radiating antenna with a dielectric using `pcbStack` and view the antenna.

```

pcb = pcbStack;
pcb.BoardThickness = h;
pcb.BoardShape = boardShape;
pcb.Layers = {radiator,substrate,gndPlane};
pcb.FeedLocations = [0 -BoardW/2 3 1];
pcb.FeedDiameter = feedW/2;

```

```
figure
show(pcb)
```



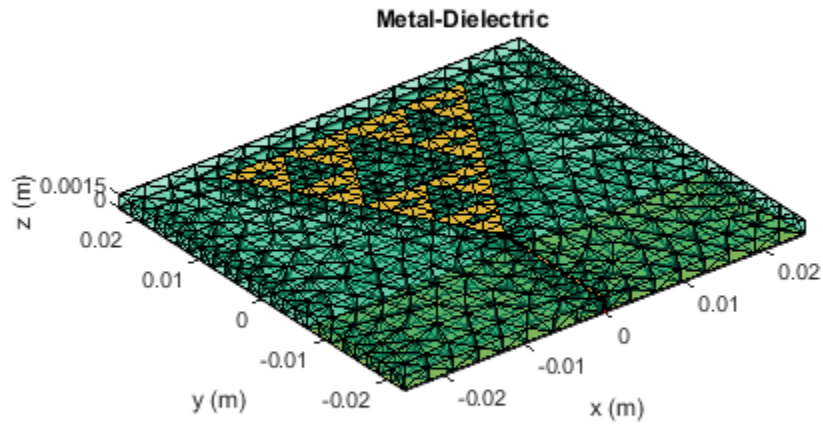
Analyze Return Loss and Gain of Antenna

Manually mesh the antenna structure. The wavelength λ at the center frequency is 0.0286 m., so set `MaxEdgeLength` to $\lambda/8$. Eight triangles per wavelength are enough to simulate the structure.

Calculate the wavelength for a resonant frequency of 5 GHz. The relative permittivity of the FR4 is 4.4.

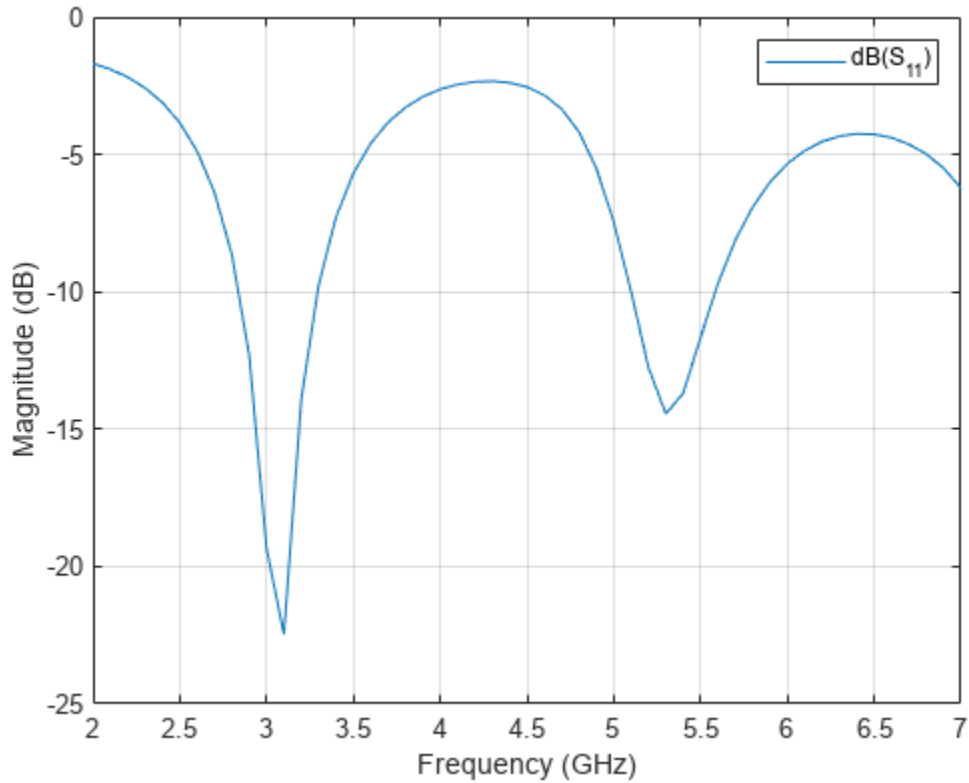
```
lambda = physconst("lightspeed)/(5e9*sqrt(4.4));
figure
mesh(pcb,MaxEdgeLength=lambda/8);
```

```
NumTriangles: 423  
NumTetrahedra: 2655  
NumBasis:  
MaxEdgeLength: 0.003573  
MeshMode: manual
```



Use the `sparameters` function to calculate the S-parameters of the antenna over a frequency range of 2 to 7 GHz. Plot the S-parameters using the `rfplot` function.

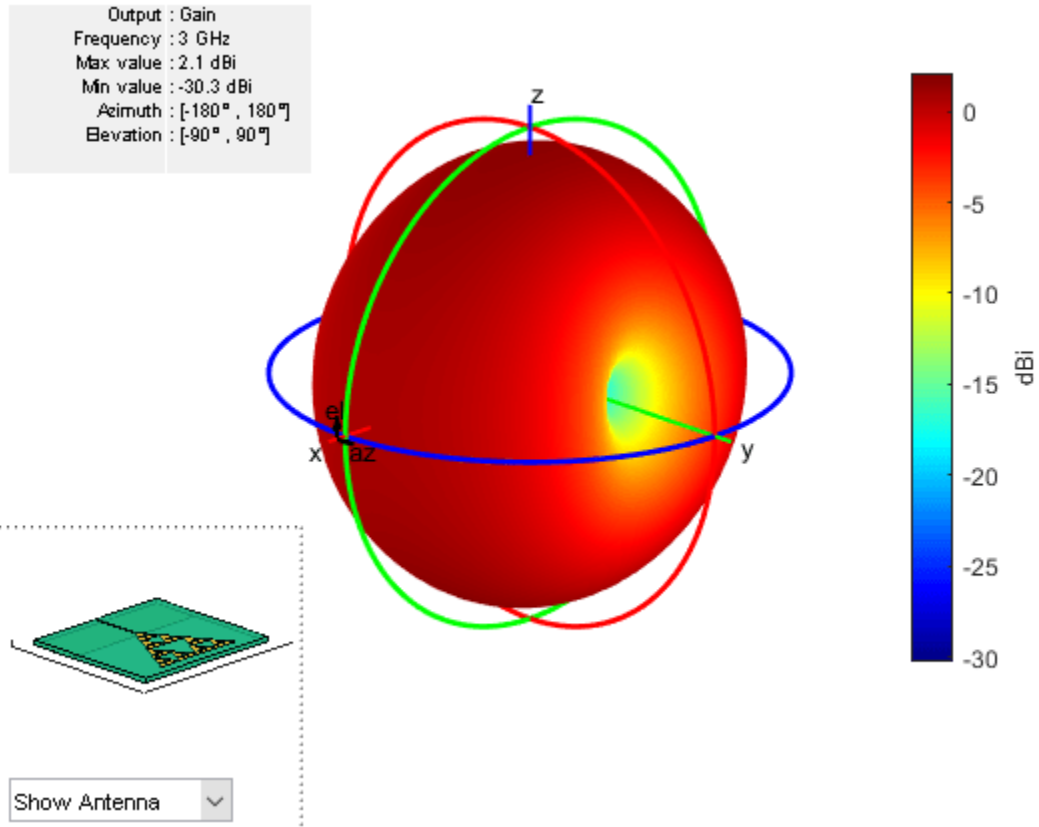
```
spar = sparameters(pcb,linspace(2e9,7e9,51));  
figure  
rfplot(spar)
```

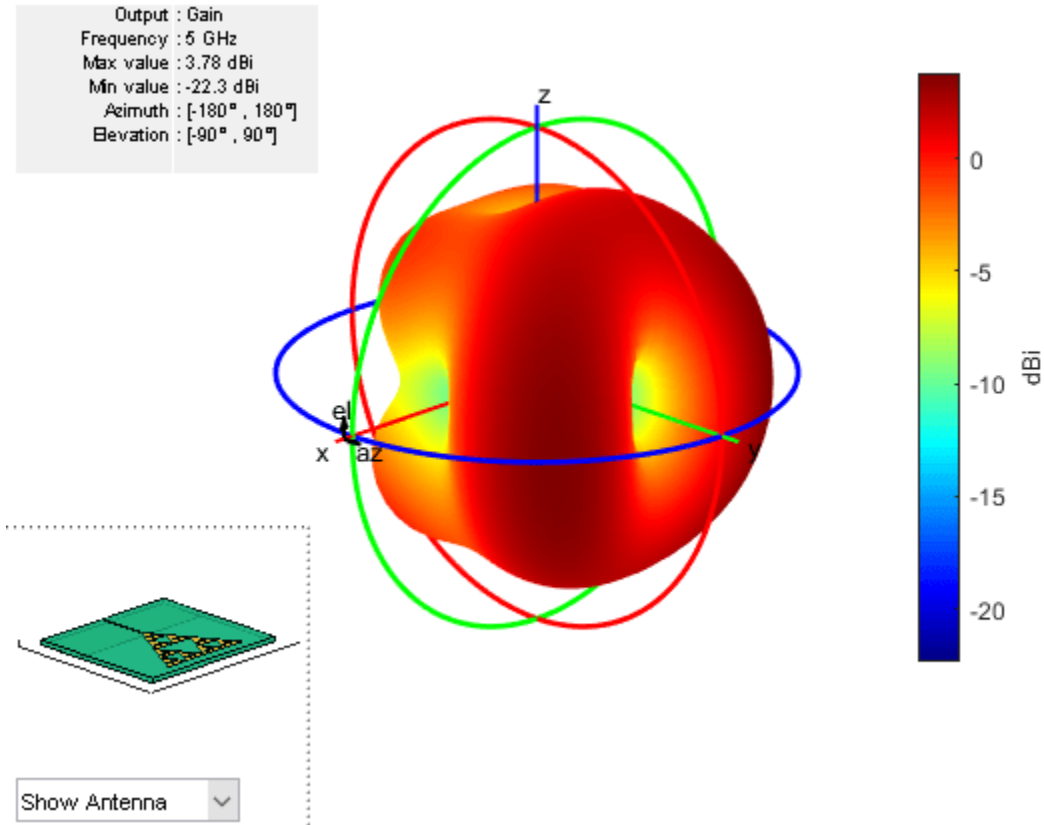
The Modified Sierpinski fractal monopole antenna exhibits its first resonance point centered at 3 GHz, for which the corresponding value of S_{11} is -22.48 dB. The impedance bandwidth for which $S_{11} < -10$ dB is 2.7 to 3.3 GHz. The second resonance point is centered at 5.2 GHz, for which the corresponding value of S_{11} is -14.5 dB. The impedance bandwidth such that $S_{11} < -10$ dB is 4.9 to 5.6 GHz.

Plot the radiation pattern of this antenna at the first (3 GHz) and second (5 GHz) resonant frequency. The pattern plots are in the S and C frequency bands, with peak gains of 2.1 dB at 3 GHz and 3.78 dB at 5 GHz.

```
figure
pattern(pcb,3e9);
```



```
figure  
pattern(pcb,5e9)
```



Reference

- [1] Islam, Zain Ul, Muhibur Rahman, Niaz Muhammad, Zeeshan Ahmed, Faiz Khalid Lodhi, and Muhammad Haneef. "Dual Band Second Order Modified Sierpinski Monopole Reduced Size Fractal Antenna." In 2016 International Conference on Emerging Technologies (ICET), 1-5. Islamabad, Pakistan: IEEE, 2016. <https://doi.org/10.1109/ICET.2016.7813244>.

Design and Analyze Parabolic-Reflector-Backed Wideband Eggcrate Array

This example shows how to create an eggcrate array of `vivaldiOffsetCavity` antenna elements backed by a parabolic reflector. The Vivaldi antenna is widely used in radar and electronic countermeasures because of its simple structure and wide operating bandwidth. The eggcrate array consists of Vivaldi antenna elements oriented in a grid. The eggcrate array in Antenna Toolbox™ is a fully metallic array, which is capable of handling a high average power. Vivaldi phased arrays are used for a wide range of applications at various frequencies and bandwidths. The eggcrate array of vivaldi antenna elements provides a wide bandwidth. You can back the array with any Antenna Toolbox backing structure. In this example, you analyze the Vivaldi element, eggcrate array, and eggcrate array backed with a parabolic reflector over 5G frequency bands.

Create Vivaldi Offset Cavity Antenna

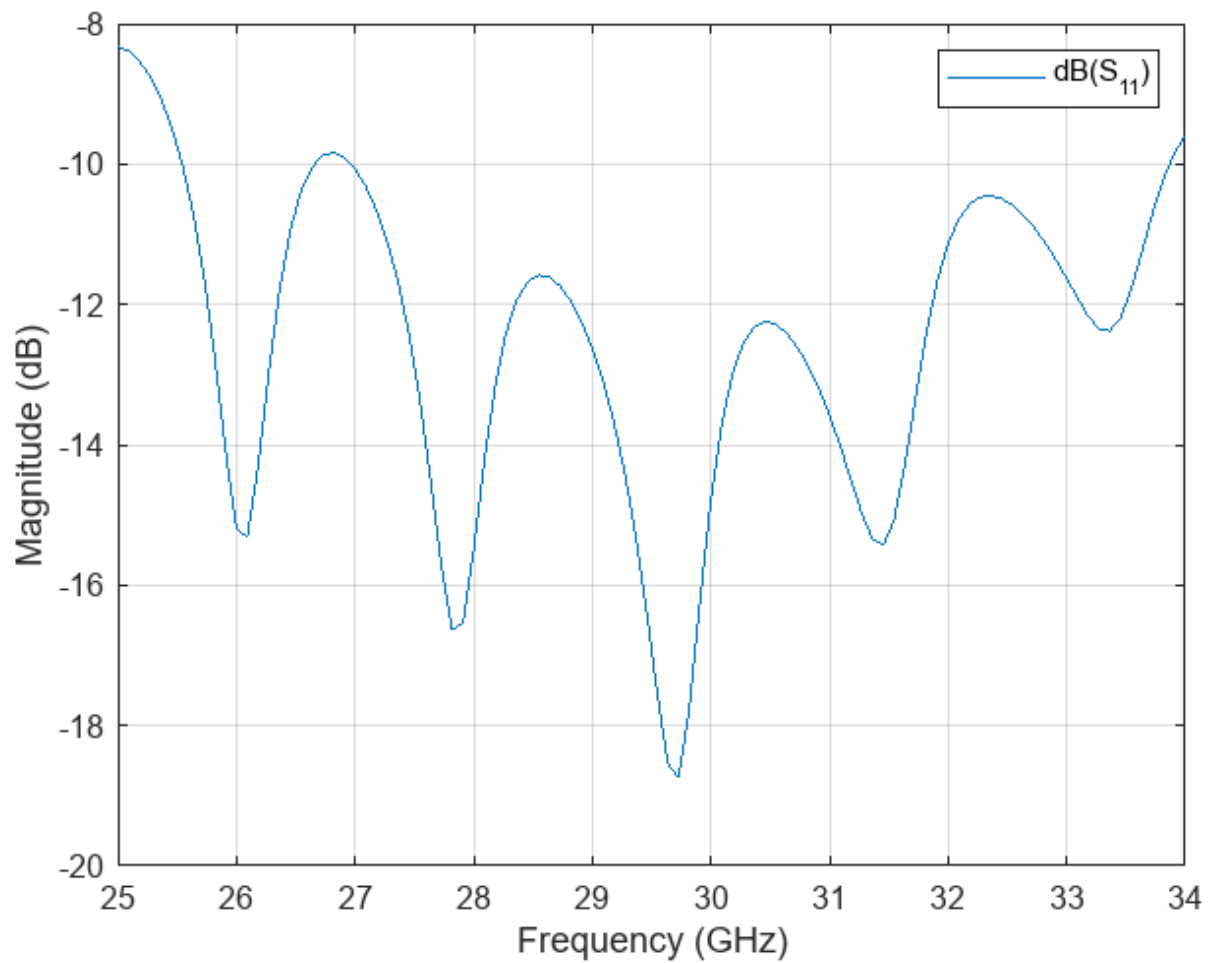
The all-metal vivaldi antenna is lightweight, has a high structural strength, and is suitable for airborne applications. The antenna consists of a tapered slot, a feed slot, and a rectangular cavity.

Create a `vivaldiOffsetCavity` antenna element by using the `design` function. Choose a 5G frequency of 26 GHz to perform analysis on the antenna element and the eggcrate array.

```
vc = vivaldiOffsetCavity;  
freq = 26e9;  
lambda = physconst("lightspeed")/26e9;  
ant = design(vc,freq);
```

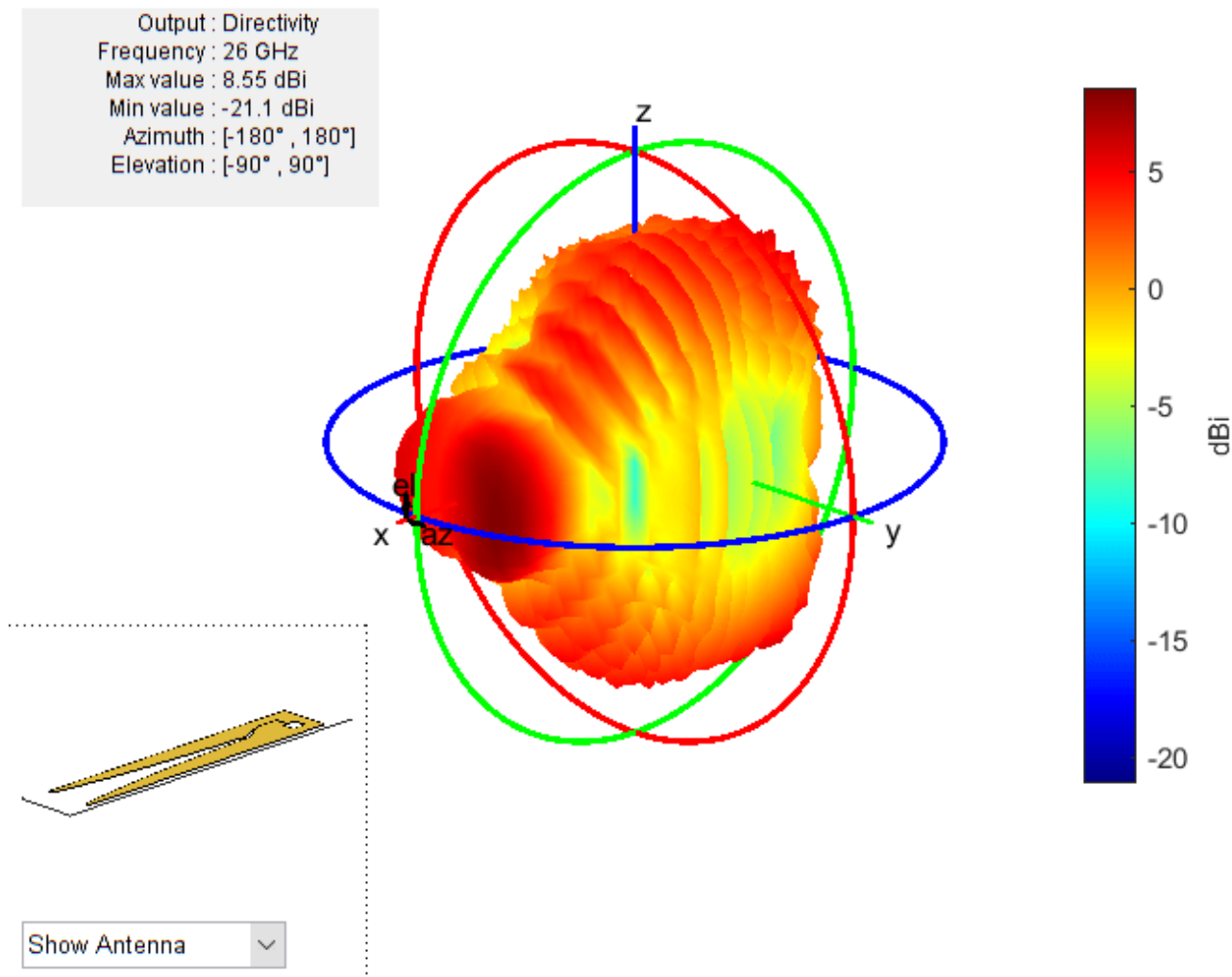
Plot the S-parameters of the antenna element over the 25 to 34 GHz frequency range. The magnitude of the reflection coefficient is less than -10 dB over the 27 to 33.9 GHz frequency range, so the antenna has a bandwidth of 6.9 GHz.

```
freqRange = linspace(25e9,34e9,100);  
s = sparameters(ant,freqRange);  
figure  
rfplot(s)
```



Plot the radiation pattern of the antenna at 26 GHz. The maximum gain of the antenna element is 8.55 dB.

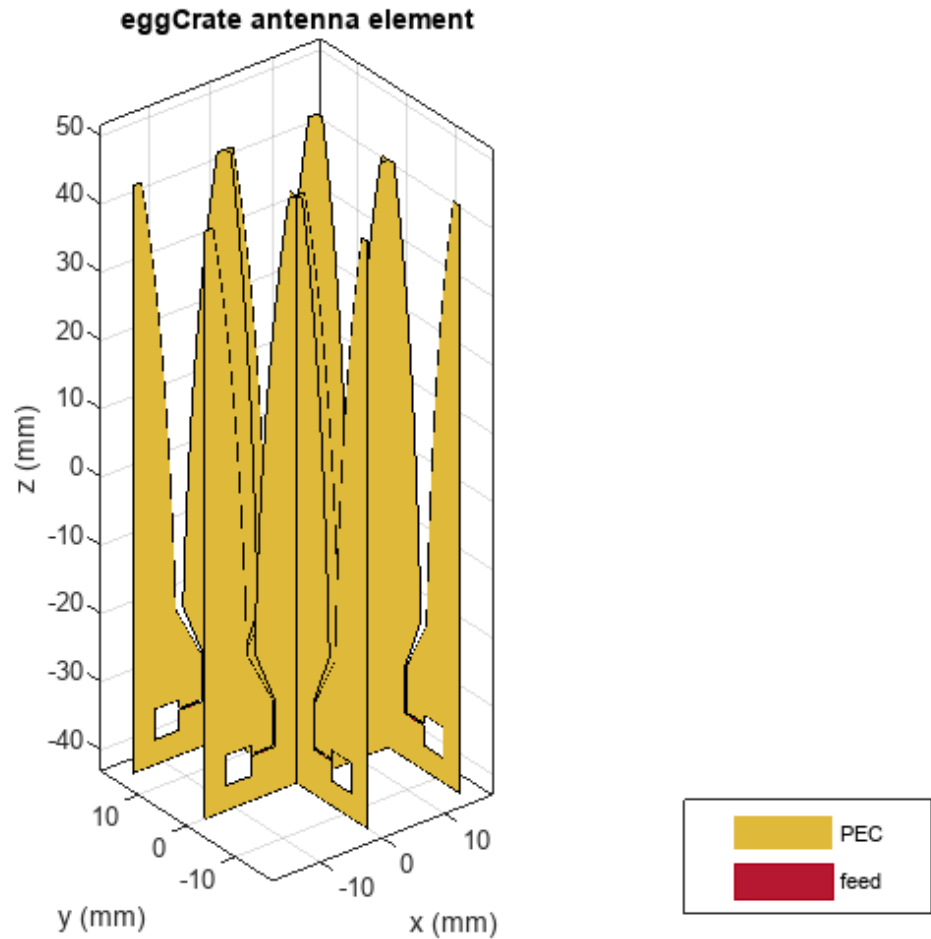
```
freq = 26e9;  
figure;  
pattern(ant, freq);
```



Create and Analyze Eggcrate Array

Create an eggcrate array with offset Vivaldi cavity elements. You can also change the element of the eggcrate array to a Vivaldi notch antenna on a ground plane. The default eggcrate array size is 2-by-2. The total number of elements in an eggcrate of size N -by- N is $2N^2$, so a 2-by-2 array contains eight elements. You can also create a linear array of Vivaldi antennas by defining the size of the eggcrate array as 1-by- N or N -by-1.

```
earray = eggCrate;  
earray.Element = ant;  
figure  
show(earray)
```



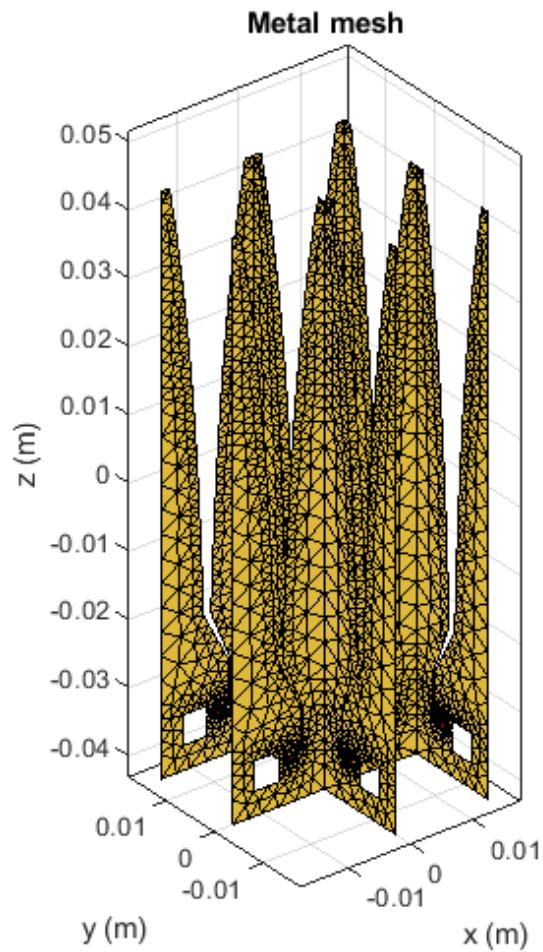
Mesh the eggcrate array with a maximum edge length equal to half the free-space wavelength. You can vary the maximum edge length to make the mesh coarser or denser. The computation time for solving an antenna or array depends on the number of triangles in the mesh.

```
figure  
mesh(earray,MaxEdgeLength=lambda/2);
```

```

NumTriangles: 5312
NumTetrahedra: 0
NumBasis:
MaxEdgeLength: 0.0057652
MeshMode: manual

```

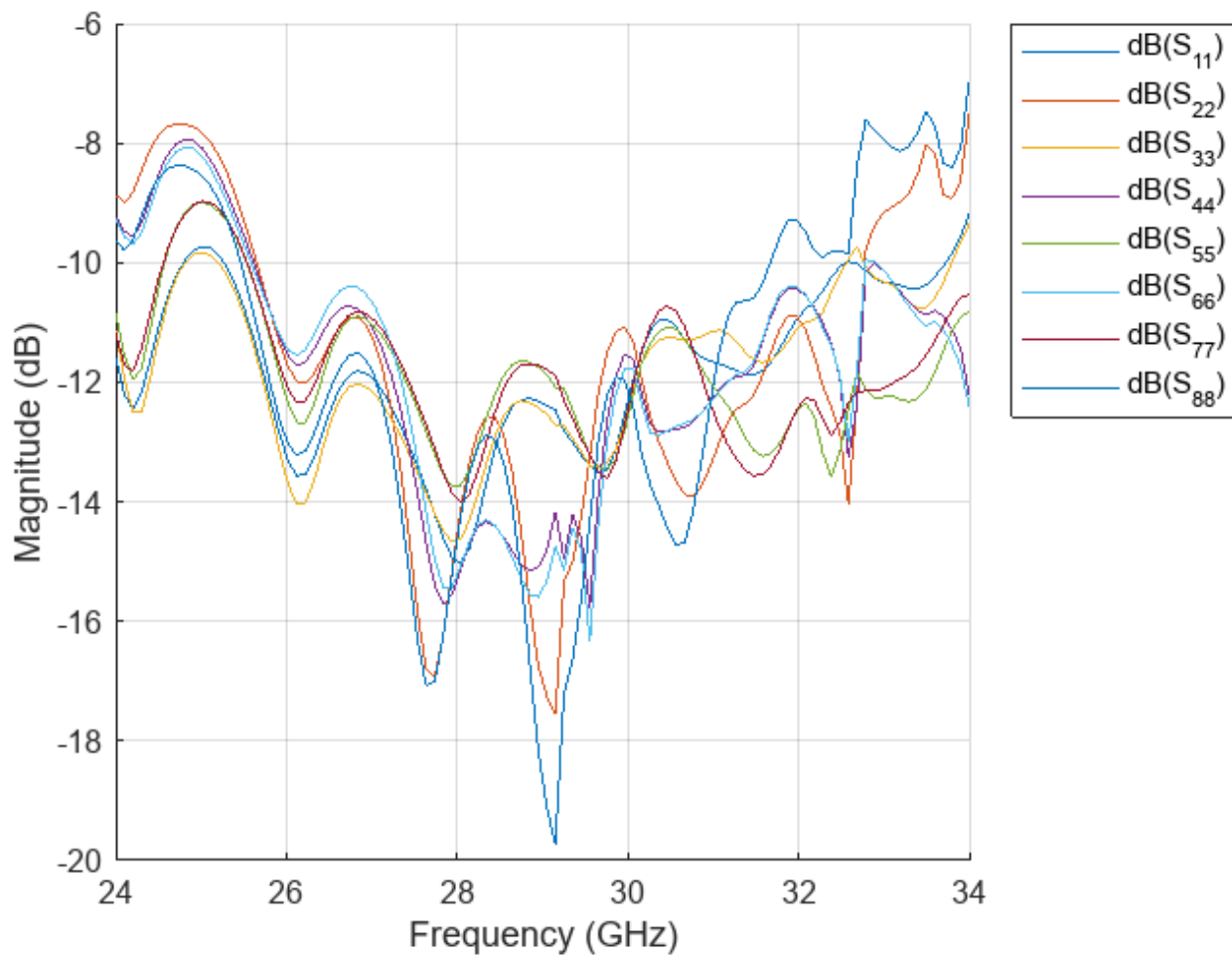


Compute and plot the S-parameters of the eggcrate array over the 24 to 34 GHz frequency range. The magnitude of the reflection coefficient is less than -10 dB between 25.6 and 31.6 GHz.

```

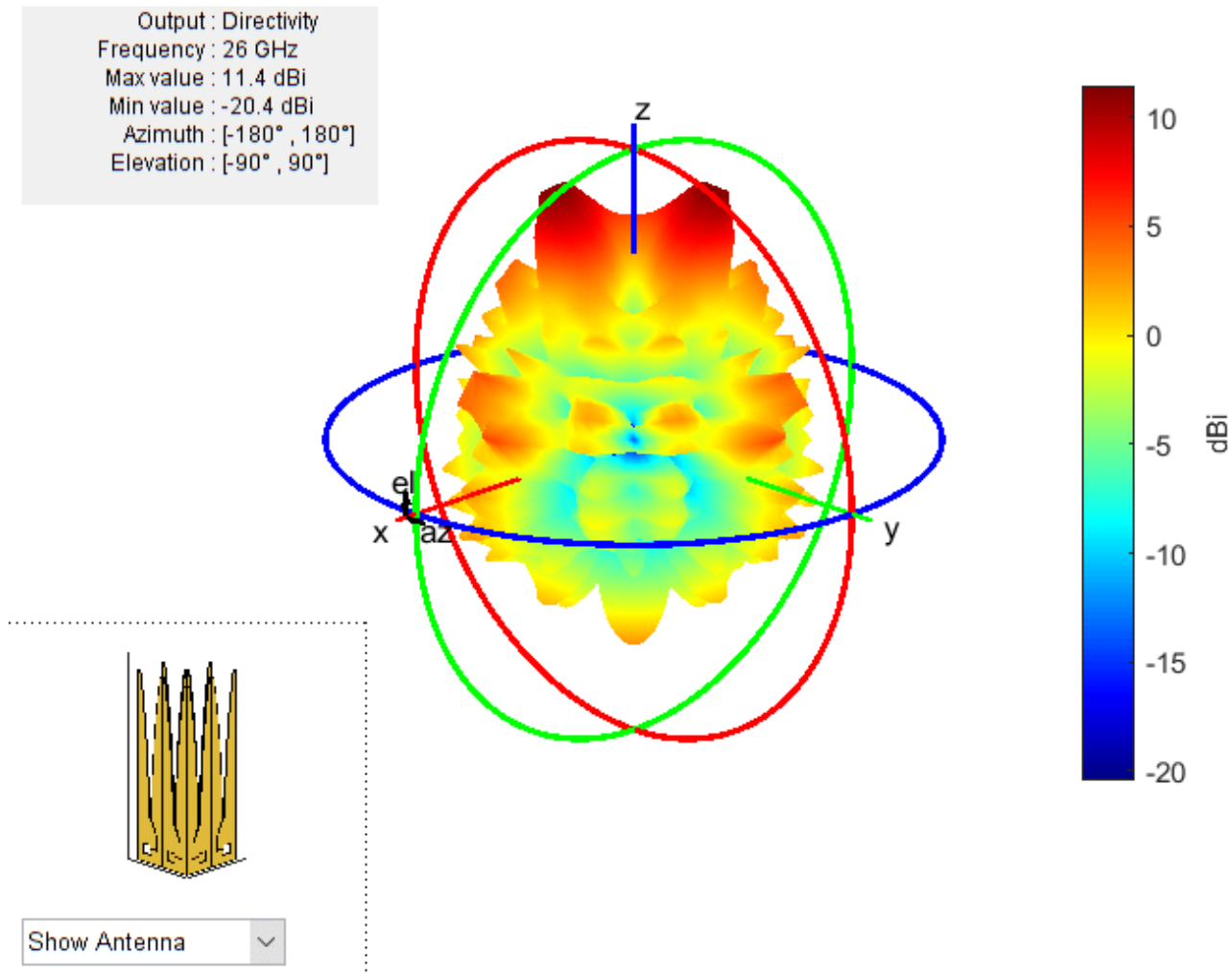
freqRange = linspace(24e9,34e9,100);
s = sparameters(earray,freqRange);
figure
for i = 1:earray.NumElements
    hold on
    rfplot(s,i,i)
end
lg = legend(Location="bestoutside");

```

Plot the radiation pattern of the eggcrate array at 26 GHz. The maximum gain of the eggcrate array at 26 GHz is 11.4 dB.

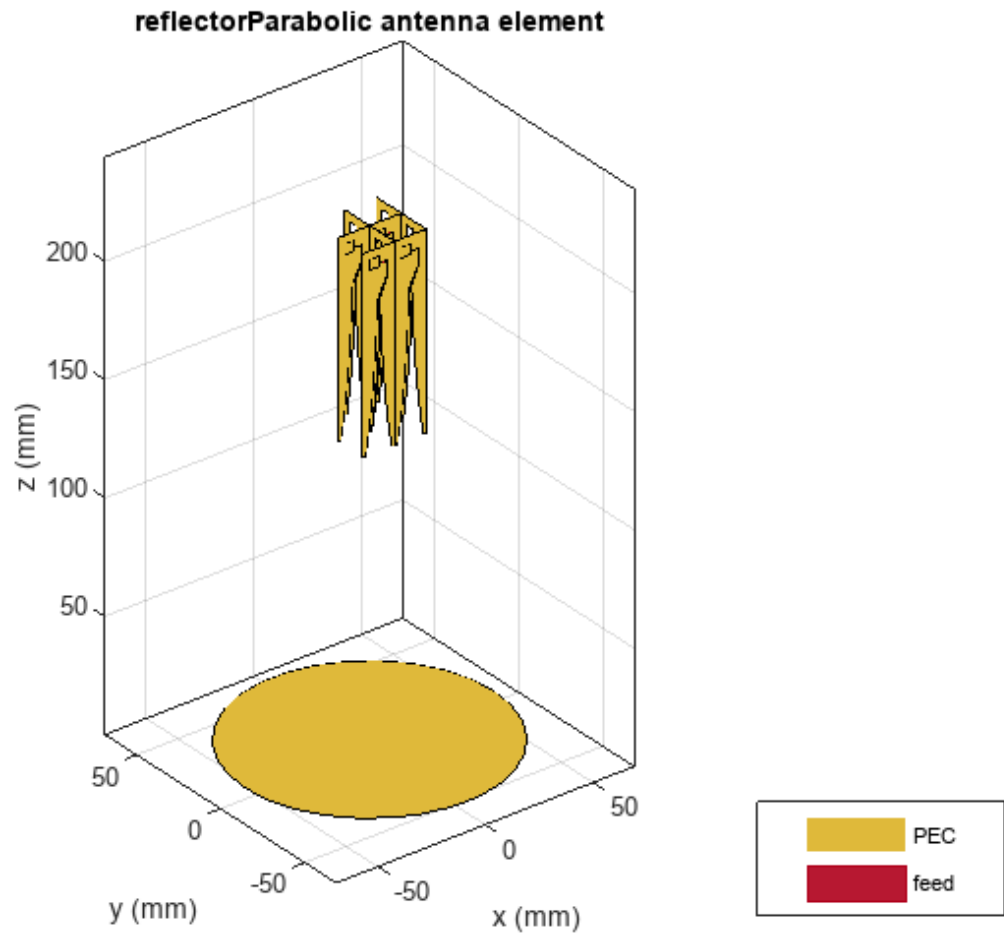
```
figure  
pattern(earray, freq)
```



Create and Analyze Eggcrate Array Backed with Parabolic Reflector

Design the parabolic reflector at 26 GHz and position the eggcrate array at a focal length of 15.5λ , where λ is the free-space wavelength. Choose the method of moments (MOM) and physical optics (PO) hybrid solver (MoM-PO). Orient the eggcrate array towards the backing reflector.

```
earray.Tilt = 180;
rp = design(reflectorParabolic,26e9);
rp.Exciter = earray;
rp.SolverType = "MoM-PO";
rp.FocalLength = 15.5*lambda;
figure
show(rp)
```

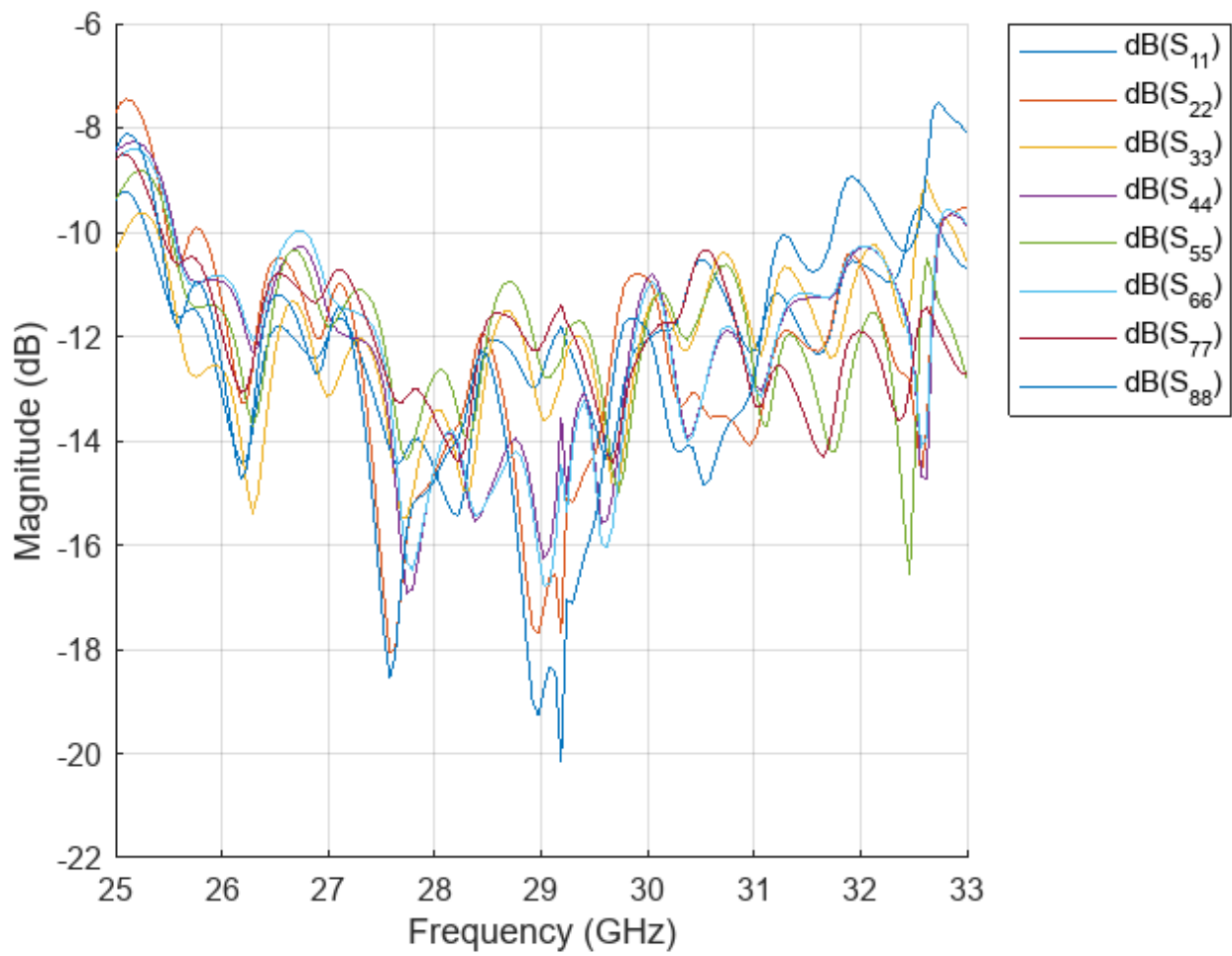


Compute and plot the reflection coefficients of each element of the array over the 25 to 33 GHz frequency range. The magnitude of the reflection coefficients is less than -10 dB between 25.8 and 31.7 GHz.

```

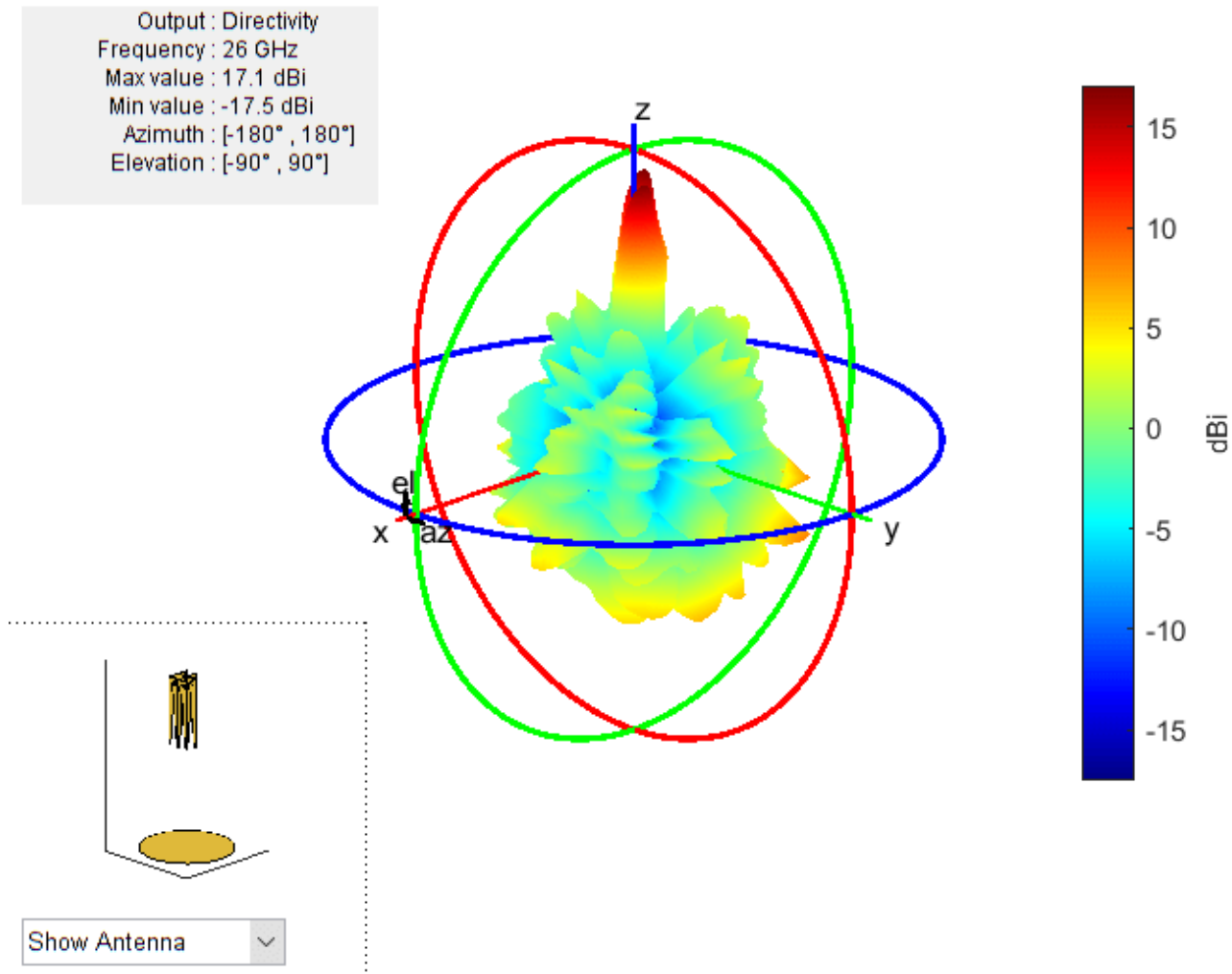
freqRange = linspace(25e9,33e9,150);
srp = sparameters(rp,freqRange);
figure
for i = 1:earray.NumElements
    hold on
    rfplot(srp,i,i)
end
lg = legend(Location="bestoutside");

```



Plot the radiation pattern of parabolic reflector backed eggcrate array at 26 GHz.

```
figure  
pattern(rp, freq)
```



Conclusion

The reflection coefficient of the eggcrate array and the parabolic-reflector-backed eggcrate array are less than -10 dB in the 25.8 to 31 GHz frequency range. The gain of the eggcrate array is 11.5 dB and increases to 17.1 dB when you back it with a parabolic reflector. You can further explore the radiation pattern behavior by varying the design parameters like the focal length of the reflector and the size of the eggcrate array.

References

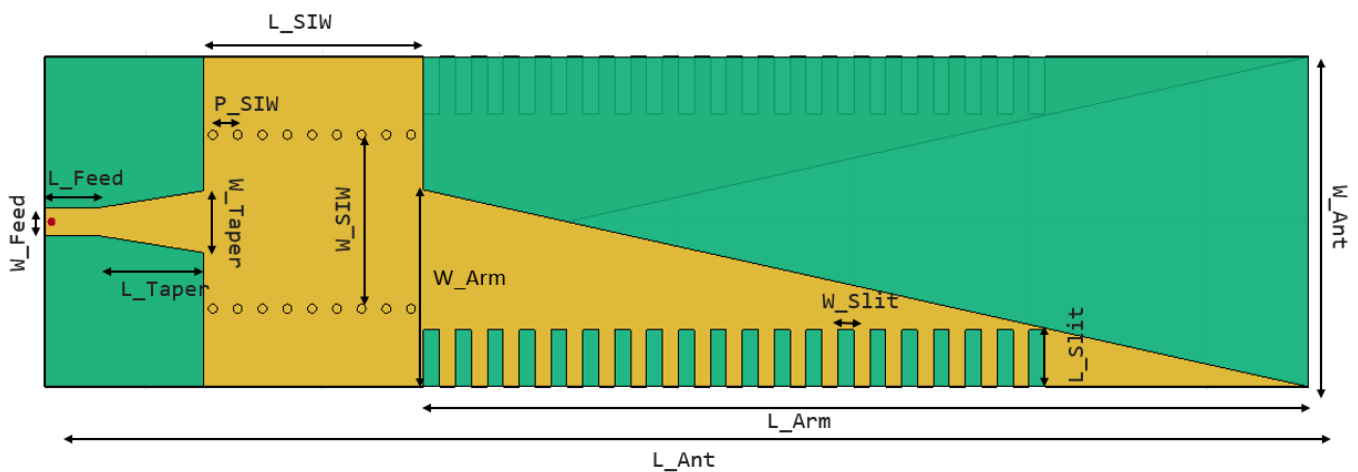
- [1] Ma, Xin, Shunlian Chai, Ke Xiao, and Liang Ding. "Design of All-Metal Vivaldi Phased Array Antenna." In 2018 IEEE 3rd International Conference on Signal and Image Processing (ICSIP), 547-51. Shenzhen: IEEE, 2018. <https://doi.org/10.1109/SIPROCESS.2018.8600487>.
- [2] Kindt, Rick W, and William R Pickles. "Ultrawideband All-Metal Flared-Notch Array Radiator." IEEE Transactions on Antennas and Propagation 58, no. 11 (November 2010): 3568-75. <https://doi.org/10.1109/TAP.2010.2071360>.

Design and Analyze Tapered-Slot SIW Filtenna

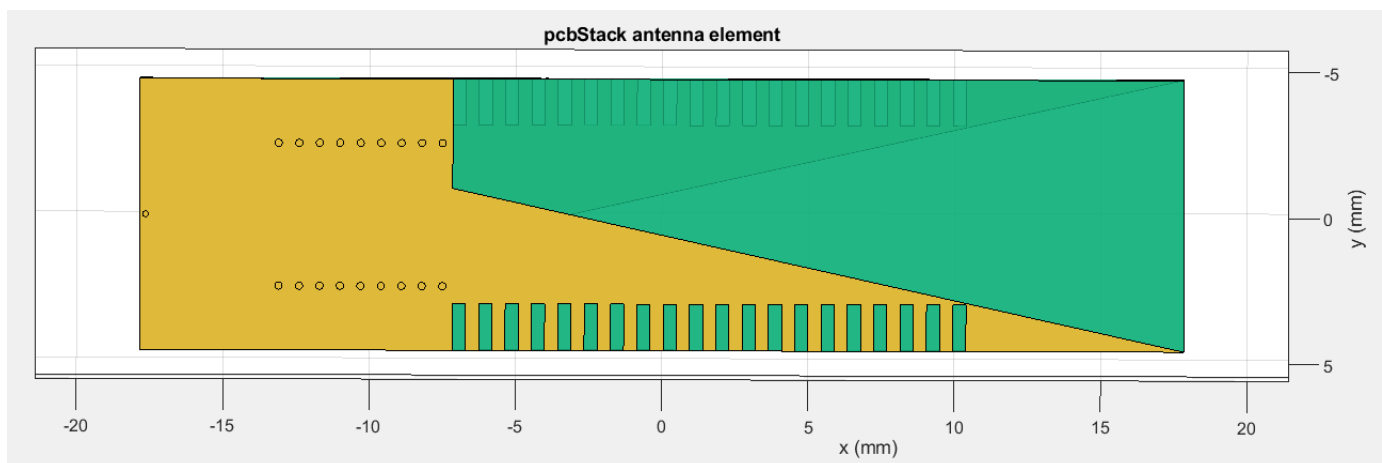
This example shows how to create the substrate-integrated-waveguide-based (SIW-based) antipodal filtenna described in [1] with a modified feeding network. A *filtenna* is a planar antenna with a built-in filter, which can be used to provide frequency agility for communicating at different frequencies without causing any interference to the adjacent bands. You can also use a filtenna to avoid undesired frequencies when you use it as a receiver.

The SIW filtenna design in this example has an operating range of 23.8 GHz to 40 GHz, maximum gain of 10.25 dB, and a maximum out-of-band suppression of 20.55 dB.

This figure shows the top view of the filtenna.



This figure shows the bottom view of the filtenna.



Create SIW Filtenna

Define the SIW patch, radiator, taper, feedline, ground plane, and slit dimensions as well as slit and via locations.

```

l_siw = 6.2e-3; % SIW patch length
l_arm = 25e-3; % Radiator length
l_taper = 3e-3; % Taper length
l_feed = 1.5e-3; % Feedline length
w_feed = 0.7826e-3; % Feedline width
w_taper = 1.745e-3; % Taper width
w_ant = 9.3e-3; % Antenna width
w_siw = 4.9e-3; % SIW patch width
w_arm = 5.55e-3; % Radiator and ground plane width
l_gnd = l_siw + l_taper + l_feed; % Ground plane length in the botto layer
w_gnd = w_ant; % Ground plane width
l_ant = l_feed + l_taper + l_siw + l_arm; % Total length of the antenna
viaX = -l_ant/2 + l_feed + l_taper + 0.25e-3; % Via start point location on x-axis
viaY = w_siw/2; % Via start point location on y-axis
p_siw = 0.7e-3; % Via pitch
l_slit = 0.45e-3; % Side slit length
w_slit = 1.6e-3; % Side slit width
slitX = -l_ant/2 + l_feed + l_taper + l_siw + l_slit/2; % X-coordinate of slit position
slitY = -w_ant/2 + w_slit/2; % Y-coordinate of slit position

```

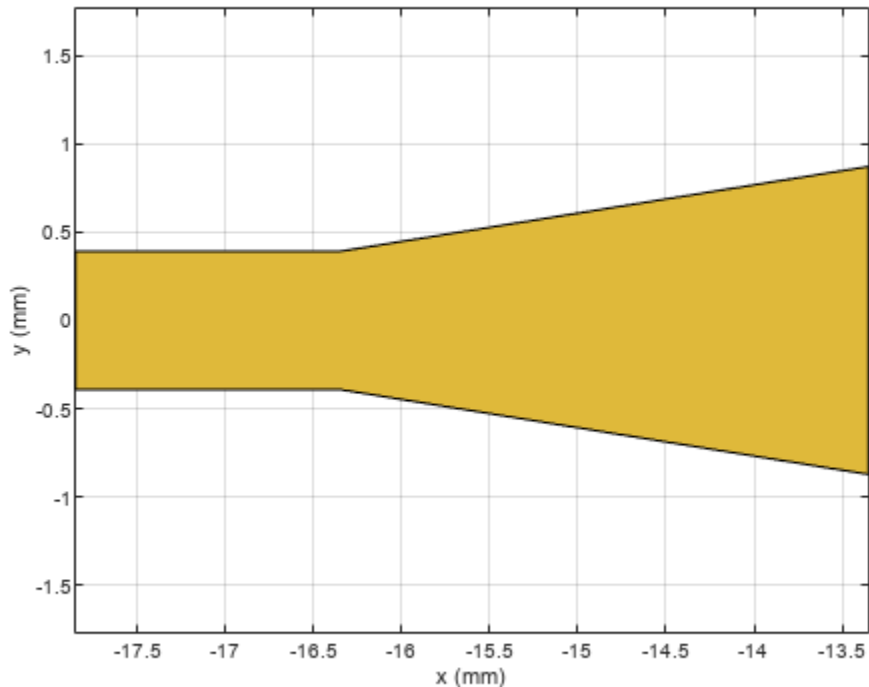
Create Tapered Microstrip Line

Use the specified dimesions to create a microstrip transmission line. Use a tapered section to match the impedance with a 50Ω microstrip line in which the quasi-TEM and TE₁₀ modes are dominant. Create a microstrip line up to the SIW transition trace. Visualize the transition using the show method.

```

feedline = antenna.Rectangle(Length=l_feed,Width=w_feed,Center=[-l_ant/2+l_feed/2 0]);
taper = antenna.Polygon;
taper.Name = "taper";
taper.Vertices = [-l_ant/2+l_feed w_feed/2;-l_ant/2+l_feed+l_taper w_taper/2;-l_ant/2+l_feed+l_t
transition = taper + feedline;
figure(Name="Tapered Micostrip Line")
show(transition);

```



Create SIW

Use `antenna.Rectangle` object to create the SIW. This object adds high-pass filter characteristics to the lower bound of the antenna resonance. The SIW operates at a frequency of 25 GHz.

Create series of vias to the SIW transmission line.

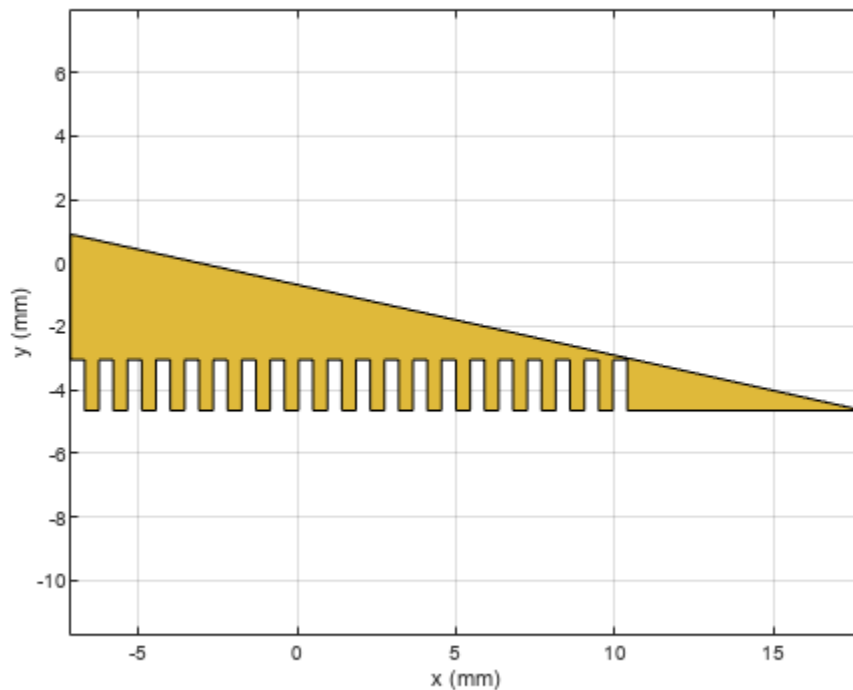
```
siw = antenna.Rectangle(Length=l_siw, Width=w_ant, Center=[-l_ant/2+l_feed+l_taper+l_siw/2 0]);
for i=1:9
    viapointX(i) = viaX + (i-1)*(p_siw);
    viapointY(i) = -viaY;
    layer1(i) = 1;
    layer2(i) = 3;
    vialoc1 = [viapointX' viapointY' layer1' layer2'];
end
for i=1:9
    viapointX(i) = viaX + (i-1)*(p_siw);
    viapointY(i) = viaY;
    layer1(i) = 1;
    layer2(i) = 3;
    vialoc2 = [viapointX' viapointY' layer1' layer2'];
end
```

Create Radiator

Create the antipodal antenna with two arms named as `topArm` and `bottomArm`. Use `antenna.Polygon` and `antenna.Rectangle` functions to create the arms of antipodal antenna with slits. Visualize the arms using the `show` function.

Create the top radiating arm.

```
topArm = antenna.Polygon;
topArm.Vertices = [-l_ant/2+l_feed+l_taper+l_siw -w_ant/2;...
    -l_ant/2+l_feed+l_taper+l_siw -w_ant/2+w_arm;...
    -l_ant/2+l_feed+l_taper+l_siw+l_arm -w_ant/2];
for i = 1:20
    Slit = antenna.Rectangle(Length=l_slit, Width=w_slit, Center=[slitX slitY]);
    slitX = slitX + 2*l_slit;
    topArm = topArm - Slit;
end
figure(Name="Top Arm")
show(topArm)
```



Create the top layer.

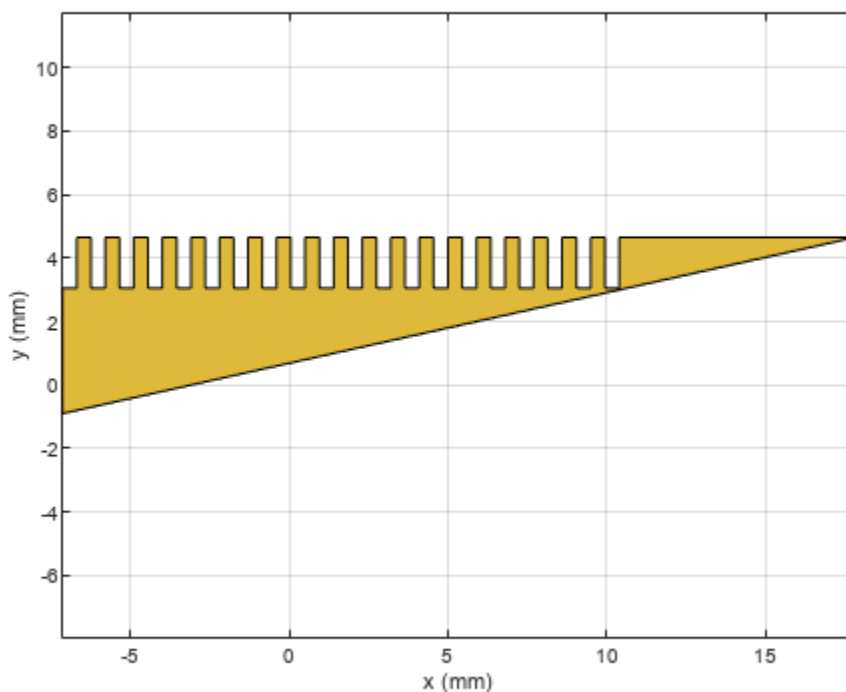
```
topLayer = siw + feedline + taper + topArm;
```

Create the bottom radiating arm.

```
bottomArm = copy(topArm);
bottomArm = mirrorX(bottomArm);
gnd = antenna.Rectangle(Length=l_gnd, Width=w_ant, Center=[-l_ant/2+l_gnd/2 0]);
```

Create the bottom layer.

```
Bottomlayer = gnd + bottomArm;
figure(Name="Bottom Arm")
show(bottomArm)
```



Create Printed Circuit Board (PCB) Antenna

Use the `pcbStack` object to create a PCB antenna using the SIW `filtenna` geometry. You can extract the shape of the top layer by querying the `Layers` property.

Visualize the antenna using the `show` function.

```
siwFiltenna = pcbStack ;
substrate = dielectric("Teflon");
substrate.EpsilonR = 2.2;
substrate.Thickness = 0.254e-3;
siwFiltenna.BoardThickness = 0.254e-3;
siwFiltenna.Layers = {topLayer,substrate,BottomLayer};
boardShape = antenna.Rectangle(Length=l_ant, Width=w_ant, Center=[0 0]);
siwFiltenna.BoardShape = boardShape;
siwFiltenna.FeedDiameter = w_feed/4;
siwFiltenna.FeedLocations = [-l_ant/2+w_feed/4 0 3 1];
siwFiltenna.ViaLocations = [vialoc1; vialoc2]

siwFiltenna =
    pcbStack with properties:

        Name: 'MyPCB'
        Revision: 'v1.0'
        BoardShape: [1x1 antenna.Rectangle]
        BoardThickness: 2.5400e-04
        Layers: {[1x1 antenna.Polygon] [1x1 dielectric] [1x1 antenna.Polygon]}
        FeedLocations: [-0.0177 0 3 1]
```

```

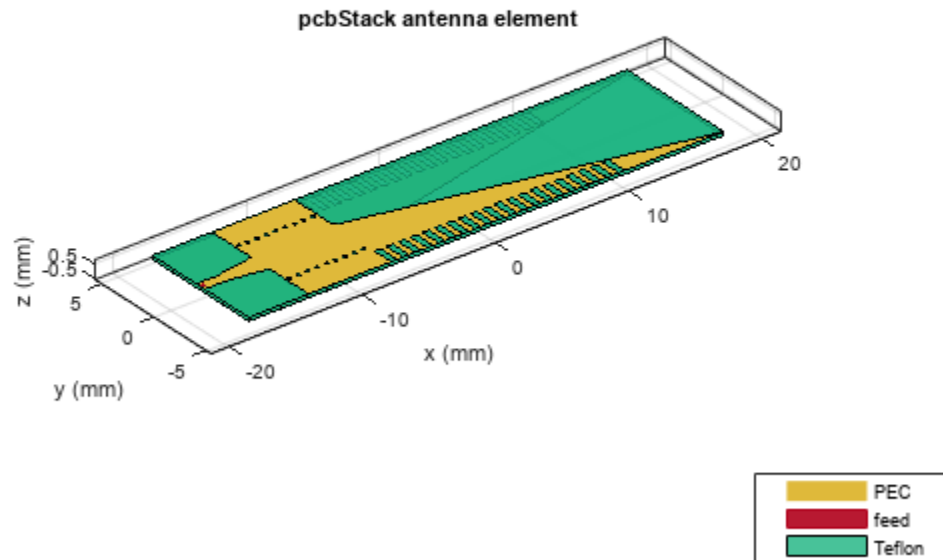
FeedDiameter: 1.9565e-04
ViaLocations: [18x4 double]
ViaDiameter: []
FeedViaModel: 'strip'
FeedVoltage: 1
FeedPhase: 0
Conductor: [1x1 metal]
Tilt: 0
TiltAxis: [1 0 0]
Load: [1x1 lumpedElement]

```

```

siwFiltenna.ViaDiameter = 0.00025;
siwFiltenna.FeedViaModel = "square";
siwFiltenna.FeedVoltage = 1;
siwFiltenna.FeedPhase = 0;
show(siwFiltenna)

```



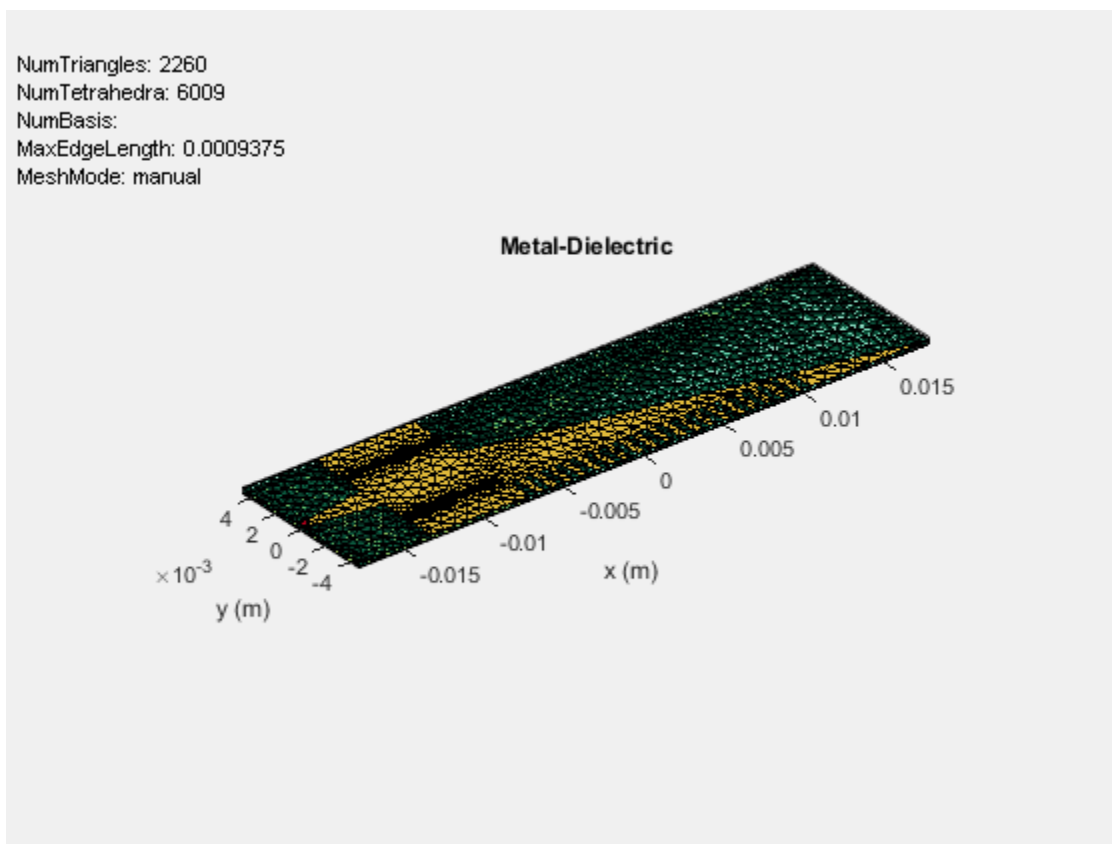
Analyze Filtenna

Use the mesh function to mesh the SIW filtenna. Set the maximum edge length to one eighth of the wavelength to get a fine mesh.

```

plotfrequency = 40*1e9;
lambda = 3e8/plotfrequency; % Wavelength
mesh(siwFiltenna, MaxEdgeLength=lambda/8);

```



Estimate memory requirement

Use the `memoryEstimate` function to calculate the memory required to solve the structure.

```
memEst = memoryEstimate(siwFiltenna,25e9)
```

```
memEst =
'2.4 GB'
```

Calculate S-Parameters

To calculate and plot the S-parameters, set the `plotSParams` variable to true. Calculating and plotting the S-Parameters can take a long time. To save time, set `plotSParams` to false and use the precomputed results in the `FilterAntennaData` MAT file attached to this example for the analysis.

```

plotfrequency = 25*1e9;
frequencyRange = (20:0.1:40)*1e9;
plotSParams = false;
if plotSParams
    s = sparameters(siwFiltenna,frequencyRange);
    rfplot(s)
end

```

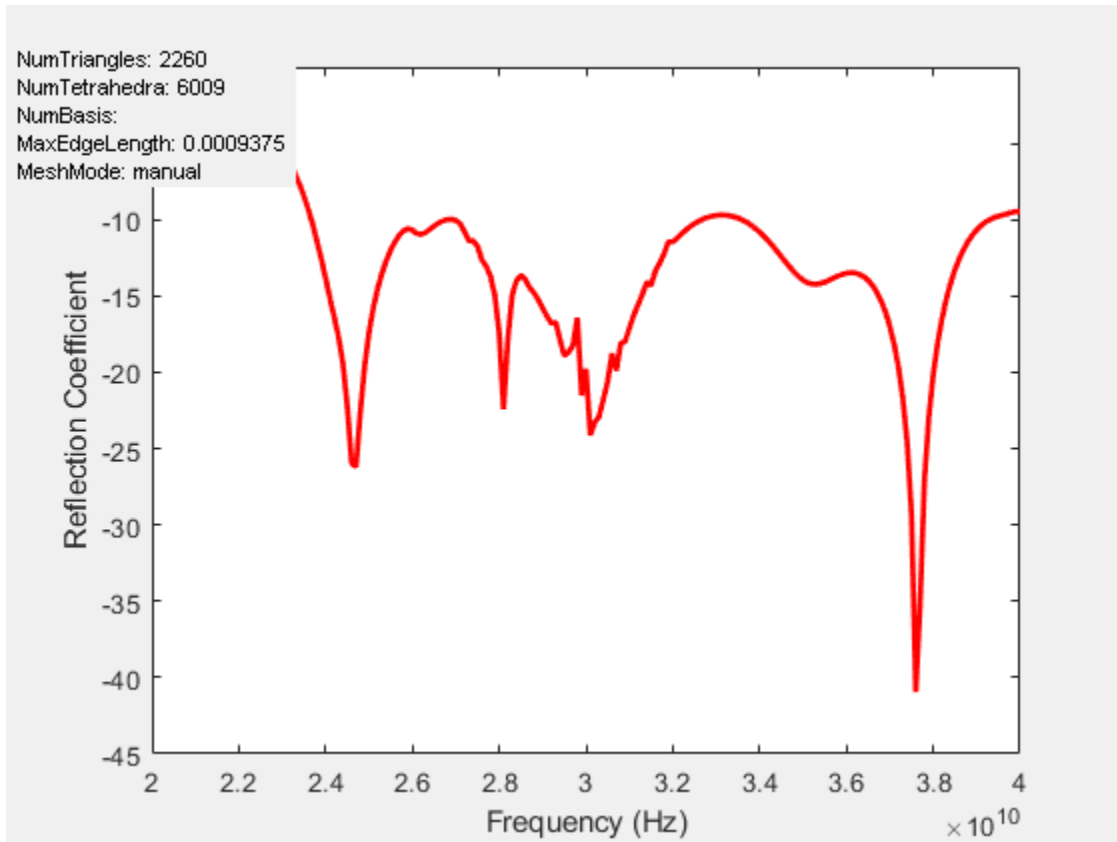
Load the data from the `FiltennaData` MAT file, which contains the simulated and measured data in the `S11_simulated`, `S11_measured`, `pattern_simulated`, and `pattern_measured` variables. Use `S11_simulated` to plot the S-Parameters.

```

load FiltennaData.mat
plot(frequencyRange,S11_simulated,"r",LineWidth=2);

```

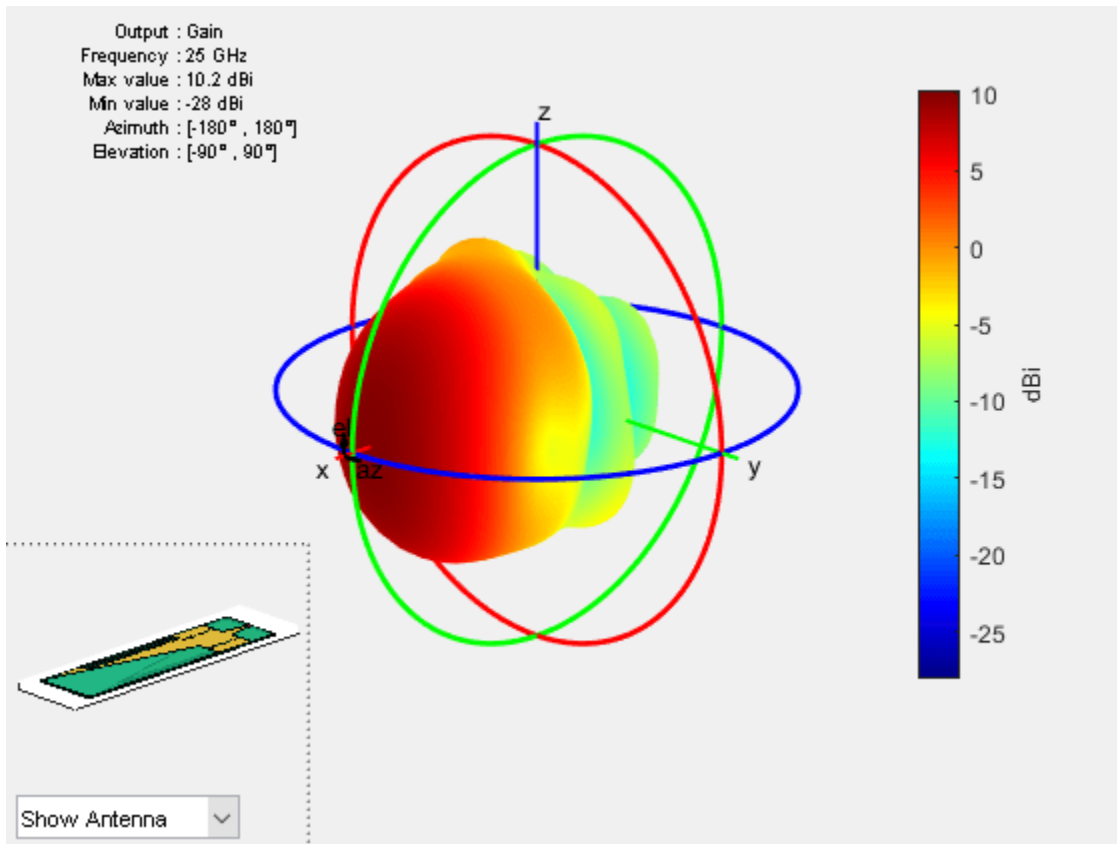
```
xlabel("Frequency (Hz)");  
ylabel("Reflection Coefficient");
```



Plot Radiation Pattern

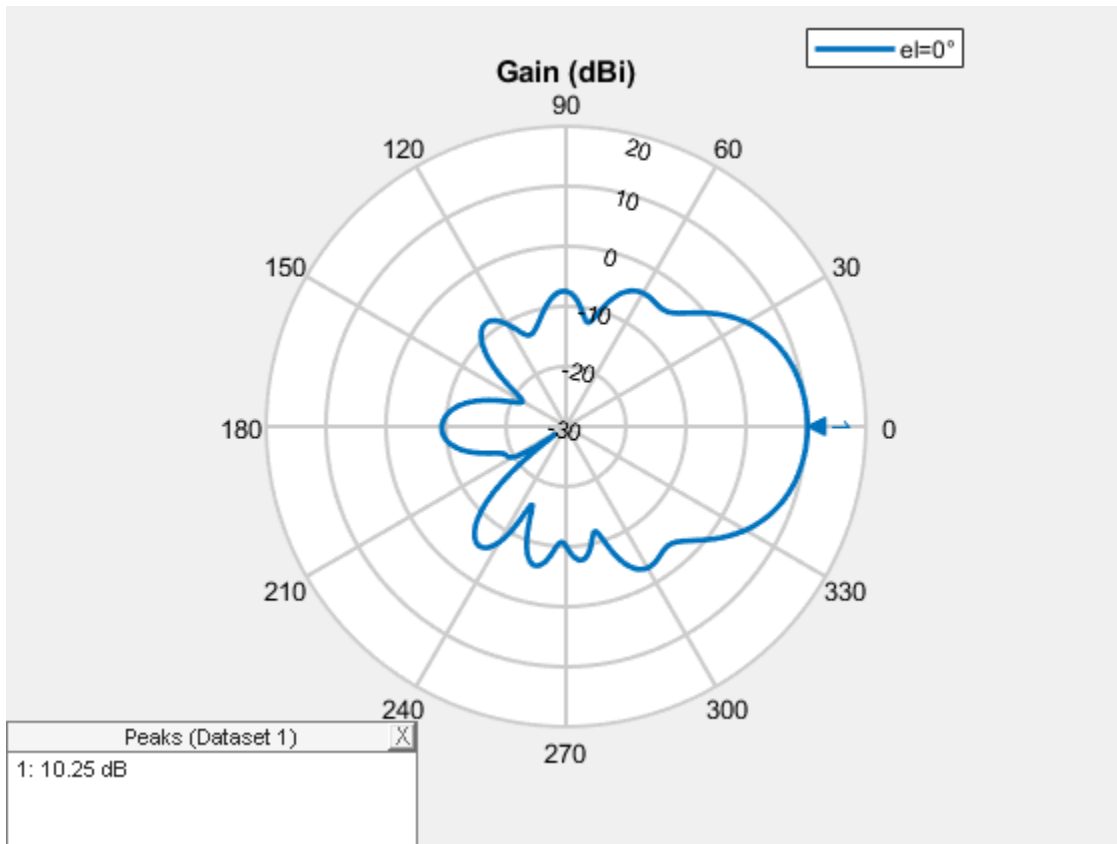
Use the `pattern` function to plot the 3-D radiation pattern at 25 GHz.

```
pattern(siwFiltenna,25e9);
```



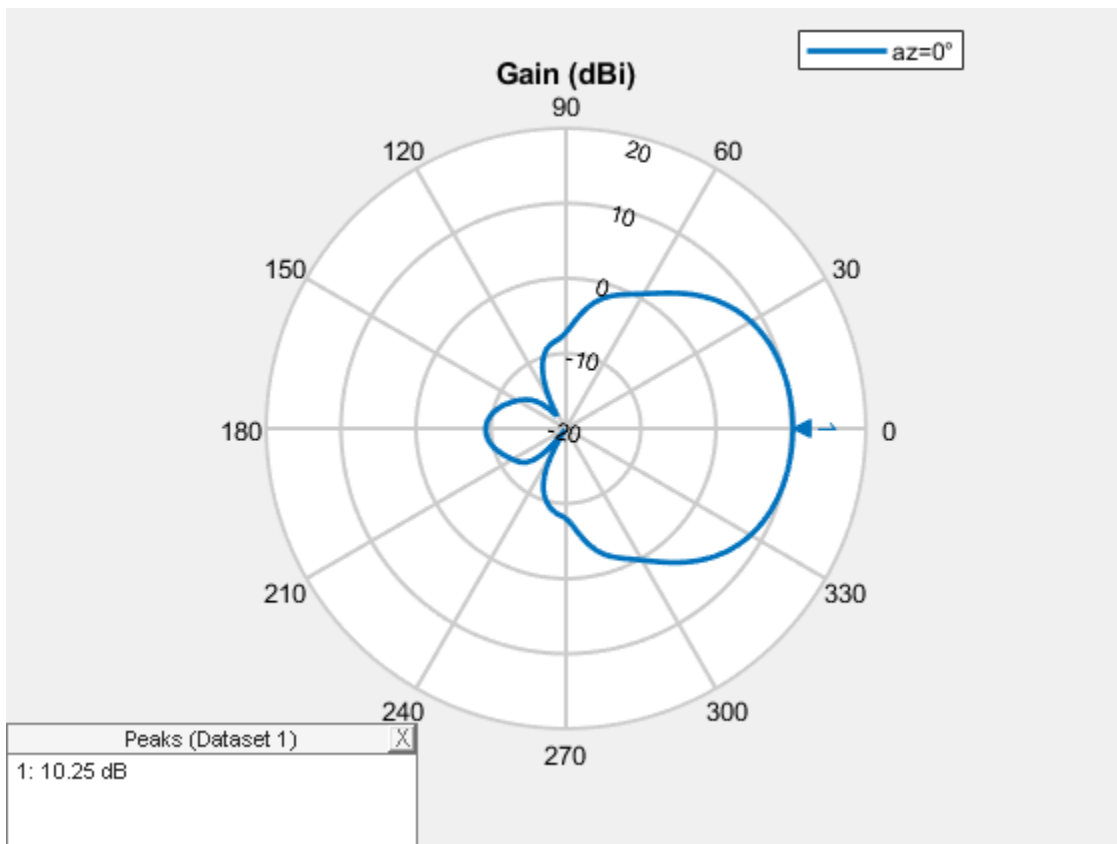
Use the pattern function to plot the 2-D radiation pattern at an elevation angle of 0 degrees .

```
pattern(siwFiltenna,25e9,0:1:360,0);
```



Use the `pattern` function to plot the 2-D radiation pattern at an azimuth angle of 0 degrees.

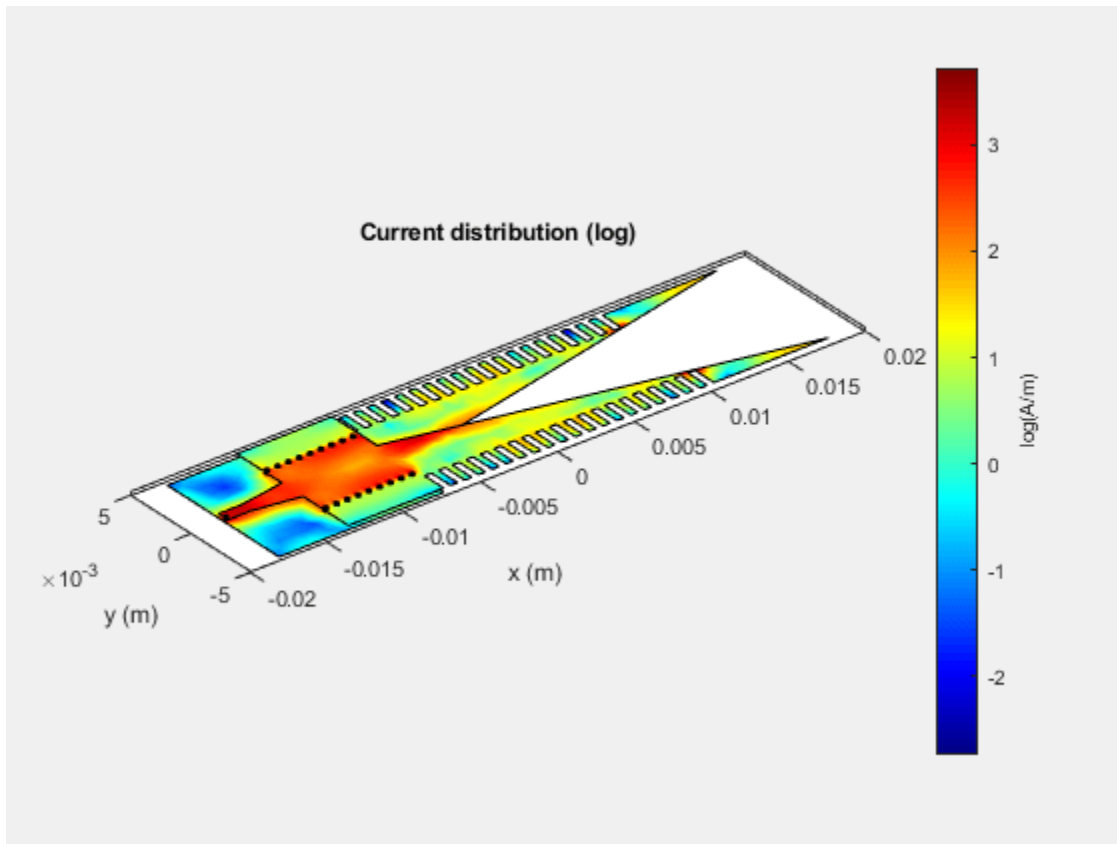
```
pattern(siwFiltenna,25e9,0,0:1:360);
```



Surface Current Distribution

Use the `current` function to plot the current distribution of the antenna at 25 GHz.

```
current(siwFiltenna,25e9, scale="log");
```

Compare Simulated and Measured Results

The SIW filtenna was built and tested for the reflection coefficient and the radiation pattern. The reflection coefficient was measured on a Keysight® N5224B PNA Network Analyzer. The radiation pattern measurements were performed in an anechoic chamber.

This figure shows the top view of the antenna.

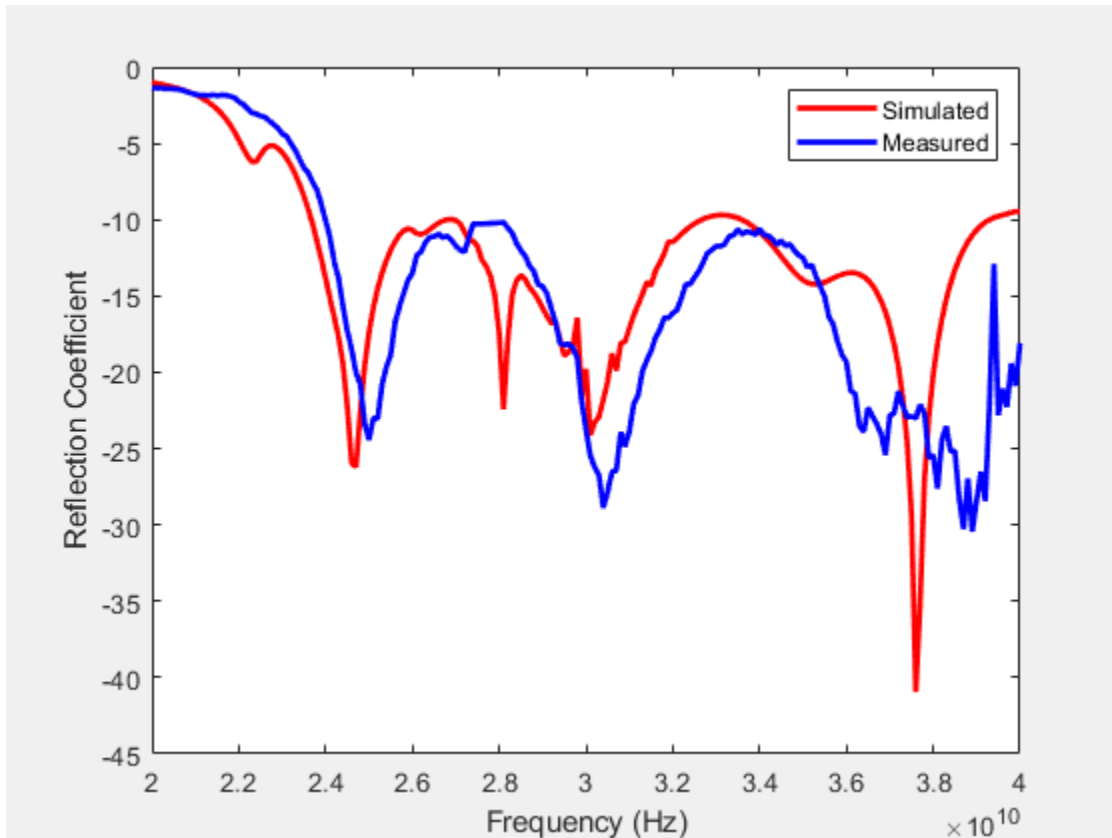


This figure shows the bottom view of the antenna.



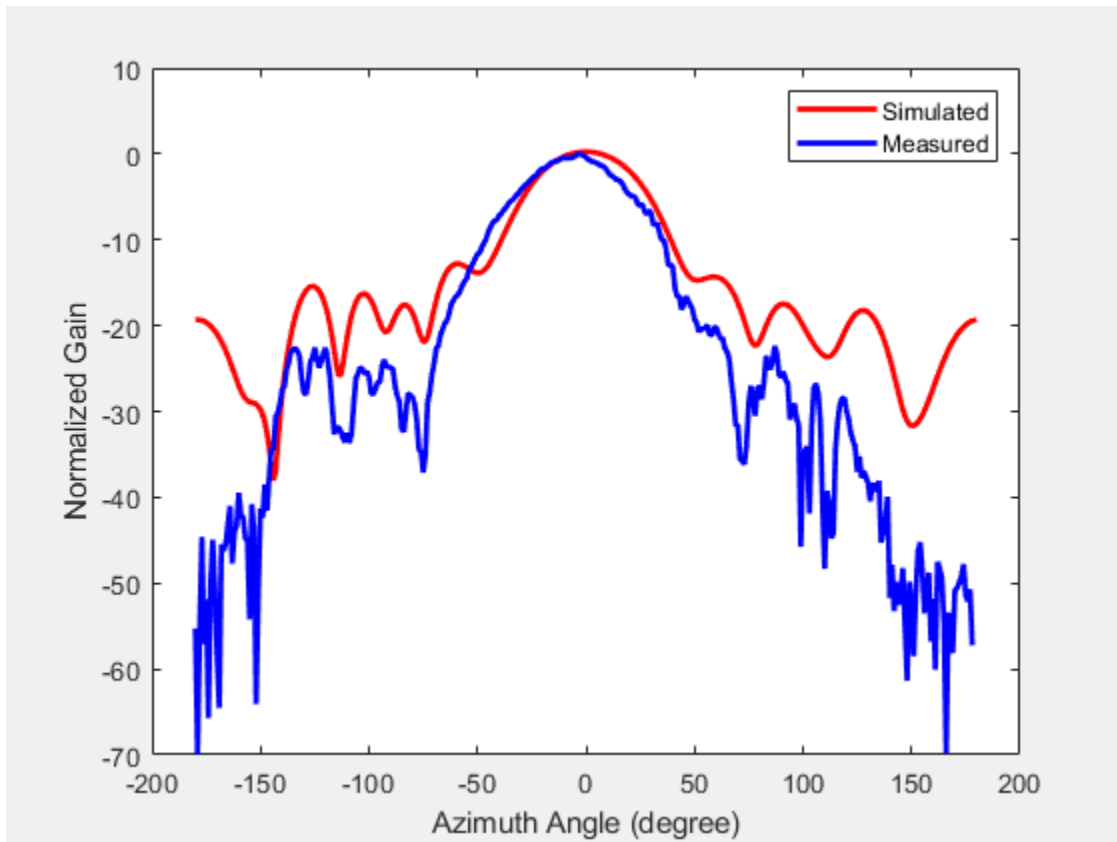
Compare the simulated and measured reflection coefficients as functions of frequency.

```
plot(frequencyRange,S11_simulated,"r",LineWidth=2);
hold on
plot(frequencyRange,S11_measured',"b",LineWidth=2);
hold off
xlabel("Frequency (Hz)");
ylabel("Reflection Coefficient");
legend("Simulated","Measured");
```



Compare the simulated and measured radiation patterns as functions of azimuth angles. The proposed antenna exhibits stable radiation characteristics with a gain of 10.25 dB at 25 GHz.

```
plot(-180:1:180,pattern_simulated,"r",LineWidth=2);
hold on
plot(-180:1:180,pattern_measured',"b",LineWidth=2);
hold off
xlabel("Azimuth Angle (degree)");
ylabel("Normalized Gain");
legend("Simulated","Measured");
```



Conclusion

The designed antenna has a maximum gain of 10.25 dB and a bandwidth of 23.8 GHz to 40 GHz. You verify the efficiency of the filtenna by comparing the simulated value against a prototype that has been fabricated and measured. The measured and simulated results agree well. The filtenna is a promising candidate for compact 5G/B5G millimeter-wave communication system.

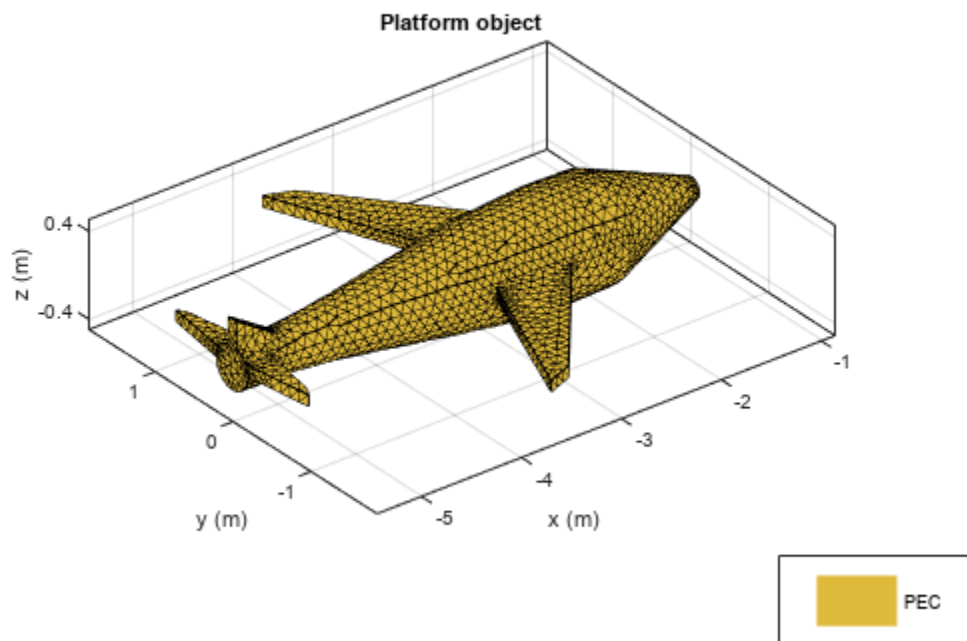
Reference

[1] Mengyun Hu, Zhiqiang Yu, Jun Xu, Ji Lan, Jianyi Zhou, and Wei Hong. "Diverse SRRs Loaded Millimeter-Wave SIW Antipodal Linearly Tapered Slot Filtenna With Improved Stopband," in *IEEE Transactions on Antennas and Propagation*, 69, no. 12, (December 2021): 8902-7. <https://doi.org/10.1109/TAP.2021.3090854>.

Antennas on A Glider as Platform

Create a platform from the STL file of a glider.

```
plat = platform('FileName', 'glider.stl', 'Units', 'm');  
figure;  
show(plat);
```



Install a dipole antenna element on the above platform. Design the dipole to operate at a frequency of 1 GHz.

```
ant = installedAntenna;  
ant.Platform = plat;  
ant.Element = design(dipole, 1e9)  
  
ant =  
    installedAntenna with properties:  
  
        Platform: [1x1 platform]  
        Element: [1x1 dipole]  
        ElementPosition: [0 0 0.0750]  
        Reference: 'feed'  
        FeedVoltage: 1  
        FeedPhase: 0  
        Tilt: 0  
        TiltAxis: [1 0 0]
```

```
SolverType: 'MoM-P0'
```

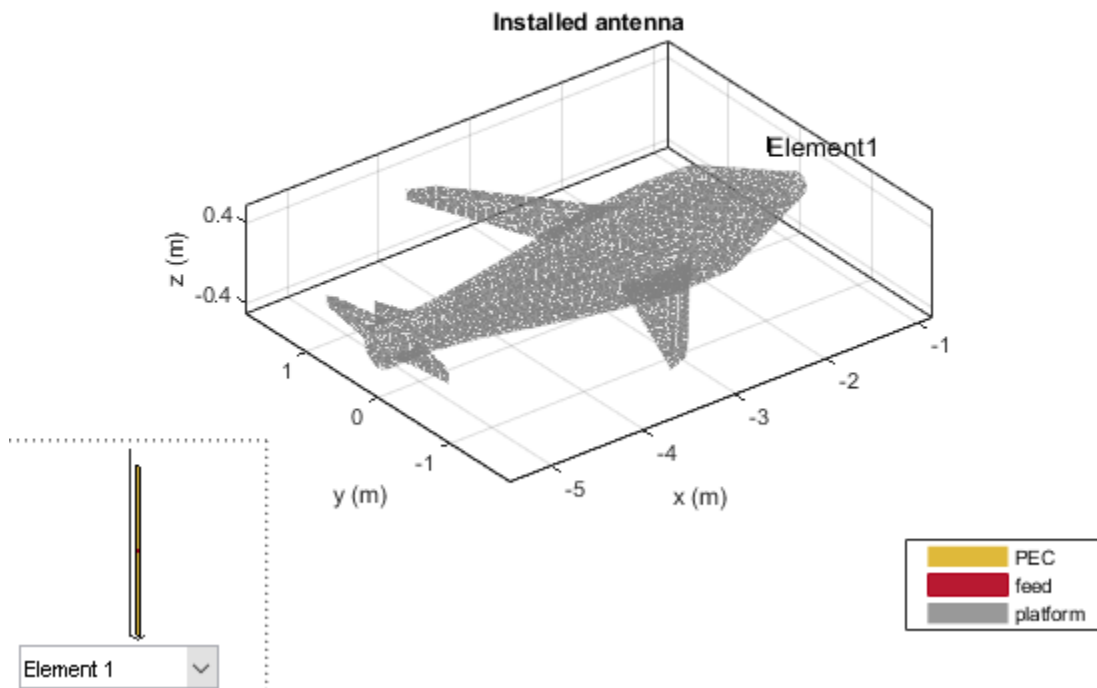
```
ant.ElementPosition = [-1.2 0 0.5]
```

```
ant =
```

```
  installedAntenna with properties:
```

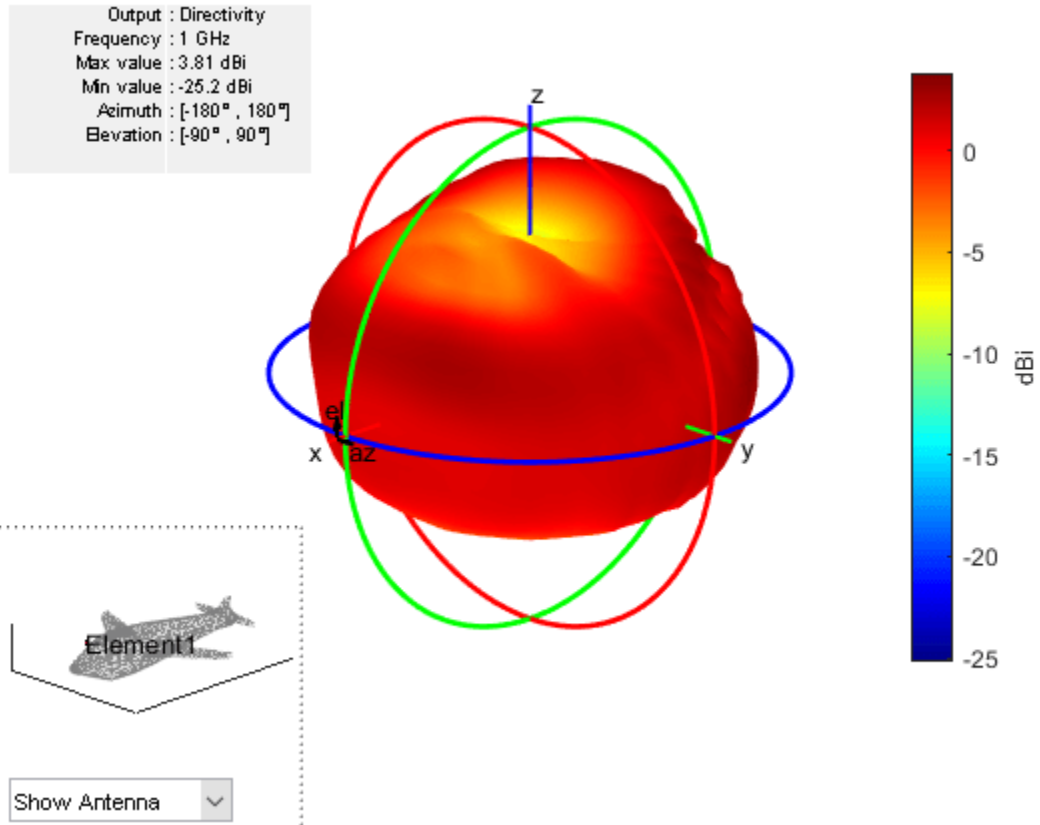
```
    Platform: [1x1 platform]
    Element: [1x1 dipole]
    ElementPosition: [-1.2000 0 0.5000]
    Reference: 'feed'
    FeedVoltage: 1
    FeedPhase: 0
    Tilt: 0
    TiltAxis: [1 0 0]
    SolverType: 'MoM-P0'
```

```
figure;
show(ant);
```



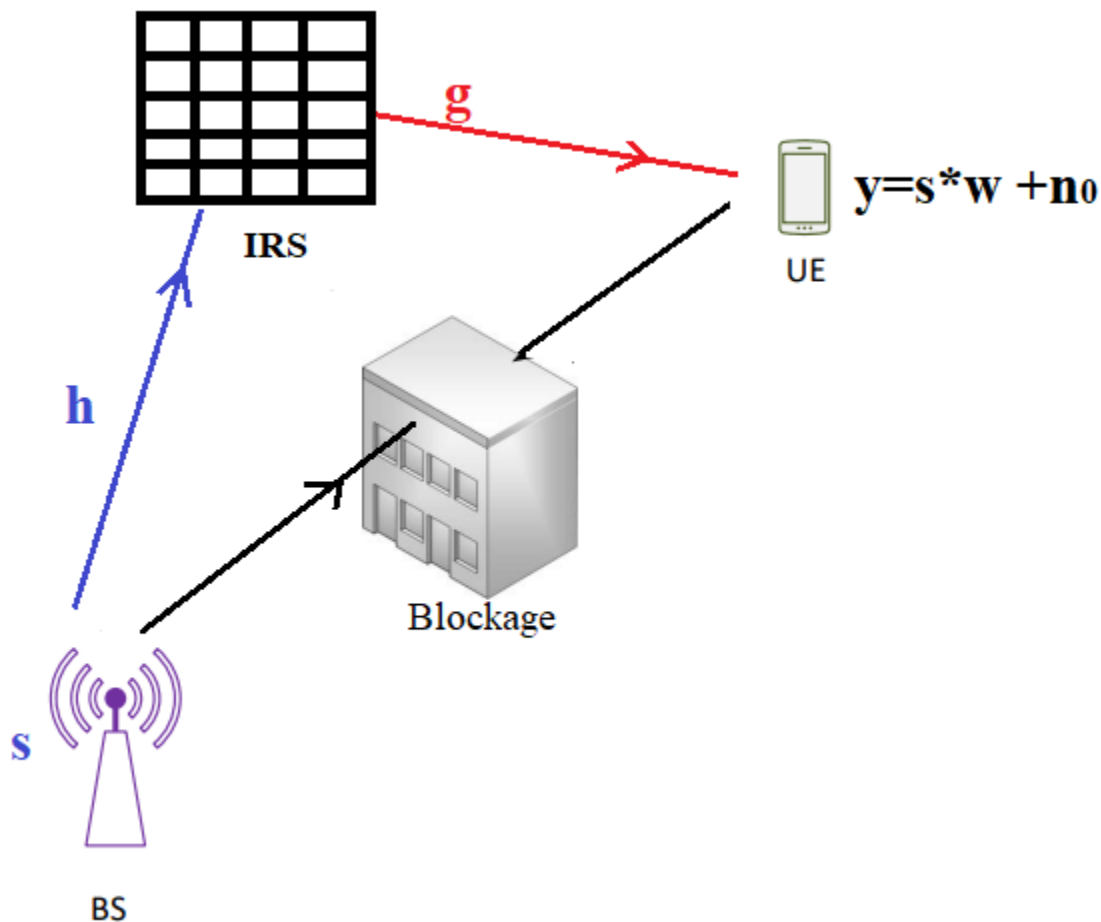
Visualize the pattern of this installation at 1 GHz.

```
figure;
pattern(ant, 1e9)
```



Electromagnetic Analysis of Intelligent Reflecting Surface

This example shows how to model the response of an intelligent reflecting surface (IRS) using full-wave electromagnetic simulation. The IRS, also known as the reconfigurable intelligent surface (RIS), large intelligent surface (LIS), or metasurface, has garnered significant interest in the wireless communication community. The IRS consists of a large array of subwavelength unit cell elements which can manipulate the phase, amplitude, polarization, or frequency of the incident signal. If an obstacle blocks the direct path between the base station (BS) and the user equipment (UE), the parasitic reflection of the IRS can provide an alternative path for signal transmission, as this figure shows. An external phase control mechanism can be integrated to control the reflection characteristics of the IRS.



The inclusion of the RIS provides a two-segment indirect path to the signal traveling between the BS and the UE. The first segment covers the path between the BS and the IRS. The second segment covers the path between the IRS and the UE. The channel matrix of these two signal paths depends on the geometry and the electromagnetic reflection properties of the IRS.

The signal received at the UE is represented mathematically in [1] using this equation.

$$r = sw + n_0 = s \frac{\sqrt{G_r G_u G_t A_r A_u}}{4\pi} \sum_{n=1}^N \frac{\sqrt{\tilde{F}_n} R_n}{d_{tn} d_{rn}} e^{-j2\pi(d_{tn} + d_{rn})/\lambda} + n_0,$$

where

- $R = [R_1 R_2 \dots R_n]^T$.
- G_u is the embedded gain of the n^{th} IRS element.
- G_t is the embedded gain of the BS transmitter.
- G_r is the embedded gain of the UE receiver.
- $F_n = F_n^t F_n^r F_n^{tx} F_n^{rx}$ is the product of the normalized power patterns of the BS, UE, and IRS.
- d_t is the distance between the BS and the n^{th} IRS element.
- d_r is the distance between the UE and the n^{th} IRS element.
- A_r is the effective aperture area of the UE.
- A_u is the effective aperture area of the n^{th} IRS element.
- λ is the free-space wavelength.

For this example, assume that the IRS has N unit cell elements. The complex reflection coefficient of the n^{th} IRS element is denoted by $R_n = R_n^{\text{mag}} e^{j\tau_n}$.

The reflection coefficients of the IRS depend on these factors:

- Direction and polarization of the incoming signal towards the IRS.
- Direction and polarization of the outgoing signal from the IRS.
- Material properties and geometry of the IRS.

This example focuses on varying the incident signal configurations and determining the corresponding reflection characteristics.

Assume that d_{tn} and d_{rn} are in the far-field range. You can abstract the IRS as `infiniteArray` catalog element with a `planeWaveExcitation` model.

Assign Operational Frequency

Set the operational frequency `F0` to 5.8 GHz and compute the corresponding wavelength `lambda` using the speed of light and frequency.

```
F0 = 5.8e9;
lambda = physconst("LightSpeed")/F0;
k = 2*pi/lambda;
```

Specify Transmitter Location for Incident Wave

Set the location of the transmit source in the far-field region by specifying arbitrary azimuth angle, elevation angle, and radial distance values. To maintain the far-field criterion, the radial distance is 100 times the wavelength. Compute the direction of the incident plane wave signal in the Cartesian coordinate system.


```

phi_inc = 0;
theta_inc = -45:1:45;
elevation_inc = 90 + theta_inc;
radius_inc = 100*lambda;
[x_inc,y_inc,z_inc] = sph2cart(phi_inc*pi/180,elevation_inc*pi/180,radius_inc);
inc_loc = [x_inc' y_inc' z_inc'];

```

Specify Receiver Location for Reflected Wave

Set the location of observation in the far-field region by specifying arbitrary azimuth angle, elevation angle, and radial distance values. Compute the receiving location in the Cartesian coordinate system.

```

phi_obs = 0;
theta_obs = 20;
elevation_obs = 90 - theta_obs;
radius_obs = 100*lambda;
[x_obs,y_obs,z_obs] = sph2cart(phi_obs*pi/180,elevation_obs*pi/180,radius_obs);
obs_loc = [x_obs, y_obs, z_obs];

```

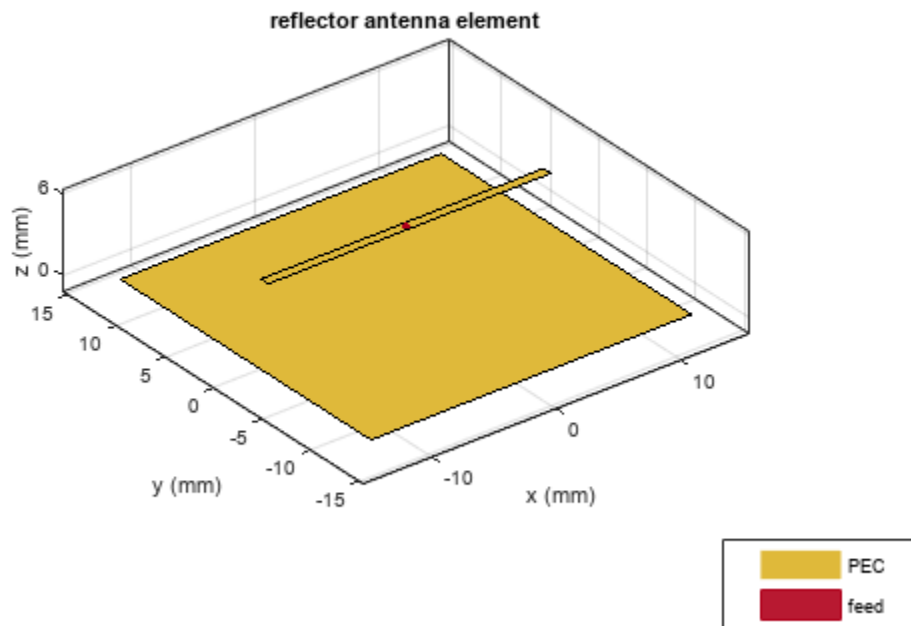
Create and Visualize Geometry of Unit Cell

Design a reflector-backed dipole antenna at the operation frequency F_0 using the `reflector` and `dipole` objects. Tilt the dipole to make it horizontal to the ground plane. Visualize the reflector element using the `show` function. The ground plane length and reflector width determine the periodicity of the IRS.

```

rf = design(reflector,F0);
rf.Spacing = lambda/10;
rf.GroundPlaneLength = 0.5*lambda;
rf.GroundPlaneWidth = 0.5*lambda;
figure;
show(rf)

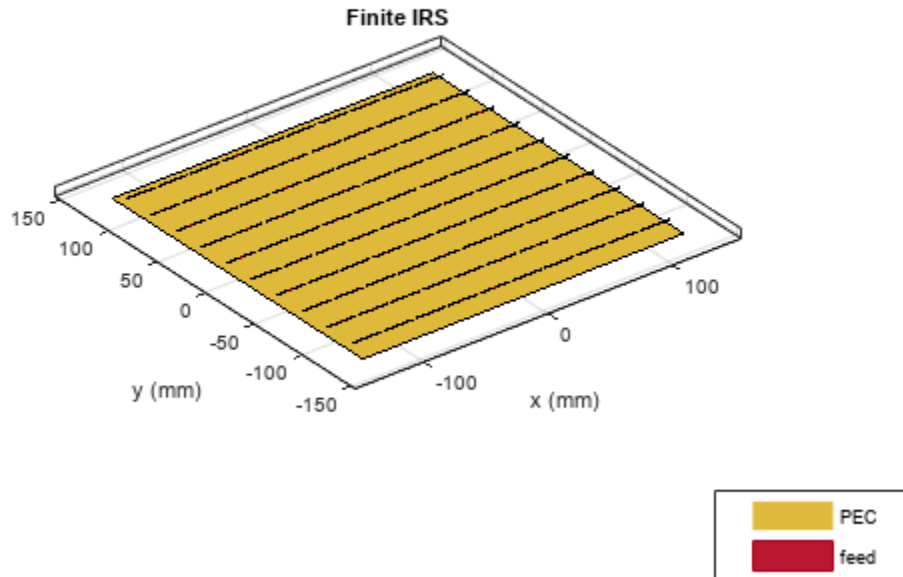
```



Visualize Finite IRS

Create a 10-by-10 finite IRS using the `rectangularArray` object and visualize it using the `show` function. Each unit cell element of the periodic IRS comprises a reflector object with a horizontal bowtie antenna as the exciter.

```
ifa = rectangularArray(Element=rf, Size=[10 10],...  
    ColumnSpacing=rf.GroundPlaneLength,...  
    RowSpacing=rf.GroundPlaneWidth);  
figure;  
show(ifa)  
title("Finite IRS")
```

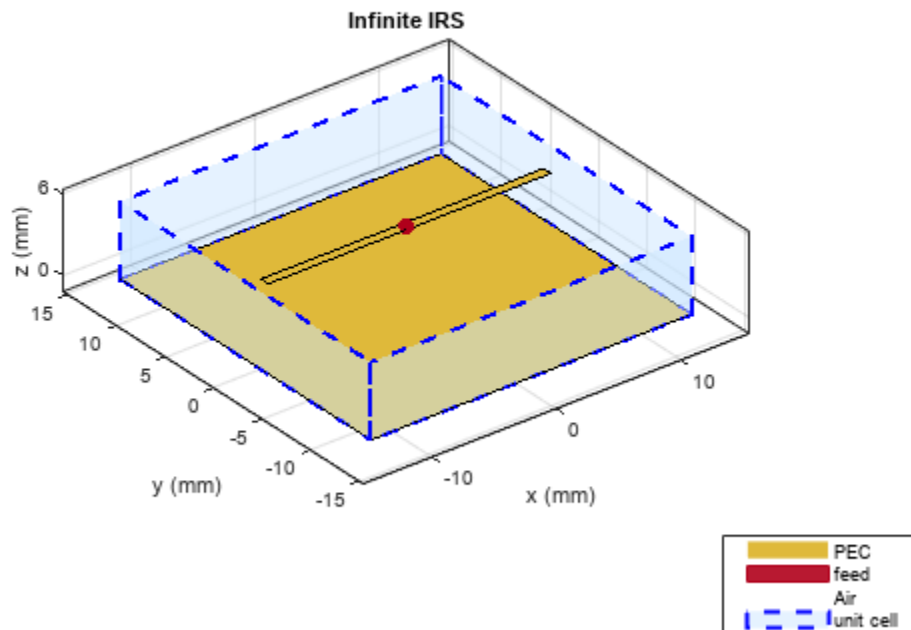


Create and Visualize the Infinite IRS

Commonly, IRS configurations contain a very large number of array elements. Considering the long simulation time of such a large array, infinite array approximation using a periodic Green's function is a suitable alternative. In the "Modeling Mutual Coupling in Large Arrays Using Infinite Array Analysis" on page 5-184 example, you design the IRS using an `infiniteArray` object.

Assign the reflector as the unit cell element of the IRS. Specify the scan angles of the IRS based on the receiver direction. The accuracy of infinite array modeling depends on the number of summation terms in the periodic Green's function, which you can see using `numSummationTerms`. Use the `show` function to visualize the IRS.

```
irs = infiniteArray(Element=rf);
irs.ScanAzimuth = phi_obs;
irs.ScanElevation = elevation_obs;
numSummationTerms(irs,20);
figure;
show(irs)
title("Infinite IRS")
```



```
% Compute the Array factor
AF=hArrayFactorCalc(irs,F0);
```

Assign Direction and Polarization to IRS

Initialize the magnitude and phase of the field reflection coefficient for different incident configurations. To excite the IRS, specify it as the Element of planeWaveExcitation object. Specify the polarization and direction of the incident wave.

Initialize the magnitude and phase of the reflection coefficients.

```
MagReflection=zeros(1,numel(theta_inc));
PhaseReflection=zeros(1,numel(theta_inc));
```

Calculate the reflection coefficients for each value of the incidence angles.

```
for mm = 1:numel(theta_inc)

    % Construct a planeWaveExcitation object with the IRS as element
    [pw,pol,dir] = hcalcPlaneWaveIncidence(theta_inc(mm),phi_inc,irs);

    % Compute incident field upon the IRS
    Ein = pol;
    Einc = dot(Ein,pol/norm(pol));

    % Compute the outward scattered field from the IRS
    [Eo,~] = EHfields(pw,F0,obs_loc');
    Eo = Eo*AF;
```

```

Eobs = dot(Eo,pol/norm(pol));

% Compute the reflection coefficient of the IRS
MagReflection(1,mm) = abs(Eobs/Einc);
PhaseReflection(1,mm) = (angle(Eobs/Einc))*180/pi;
end

```

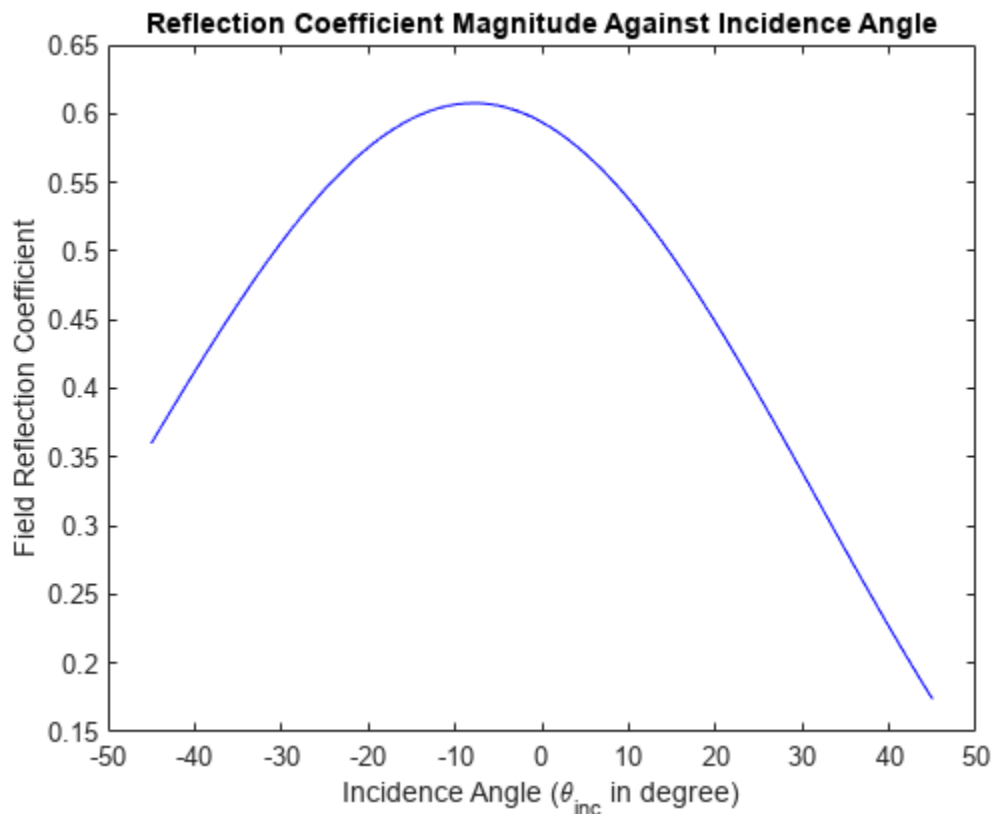
Visualize Reflection Characteristics

Visualize the magnitude and phase of the reflection coefficients. Both the parameters depend on the polarization of incident and reflected signals. Based on the phase of the reflection coefficients, the signal at the receiver end can be constructively or destructively interfered to enhance or reduce the received signal strength, respectively, in a specific observation direction.

```

figure;
plot(theta_inc,MagReflection,"b")
title("Reflection Coefficient Magnitude Against Incidence Angle")
xlabel("Incidence Angle (\theta_{inc} in degree)")
ylabel("Field Reflection Coefficient")

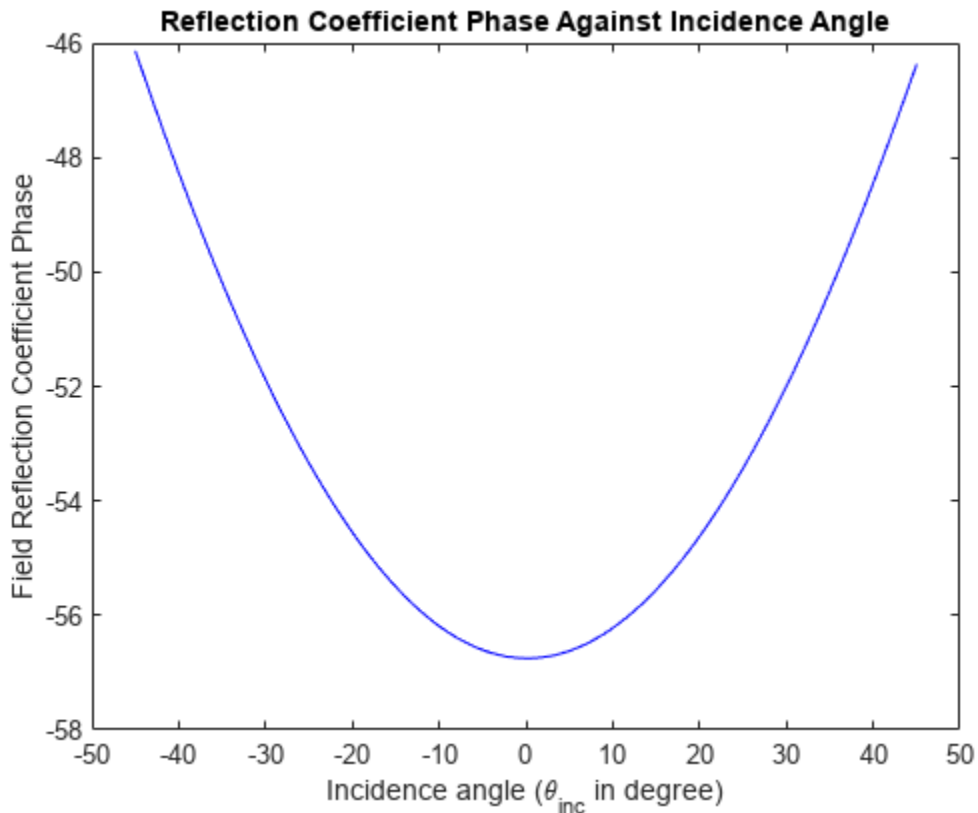
```



```

figure;
plot(theta_inc,PhaseReflection,"b")
title("Reflection Coefficient Phase Against Incidence Angle")
xlabel("Incidence angle (\theta_{inc} in degree)")
ylabel("Field Reflection Coefficient Phase")

```



You can do this analysis by using alternate infinite array configurations and different unit cell element to achieve desired IRS channel matrix behavior. Based on the phase of the reflection coefficients, the signal at the receiver's end can be constructively/destructively interfered to enhance/reduce the received signal strength in a specific observation direction.

Further Exploration

Using the full-wave electromagnetic solution of Maxwell's equations from Antenna Toolbox™, you can perform a similar analysis with different IRS configurations constructed from the `infiniteArray` and `planeWaveExcitation` objects to design a physics-based smart propagation environment. The geometry of an IRS can be further integrated with Communication Toolbox™ and Optimization Toolbox™ software to obtain an accurate, physics-based characterization of IRS-aided wireless communication scenarios.

Reference

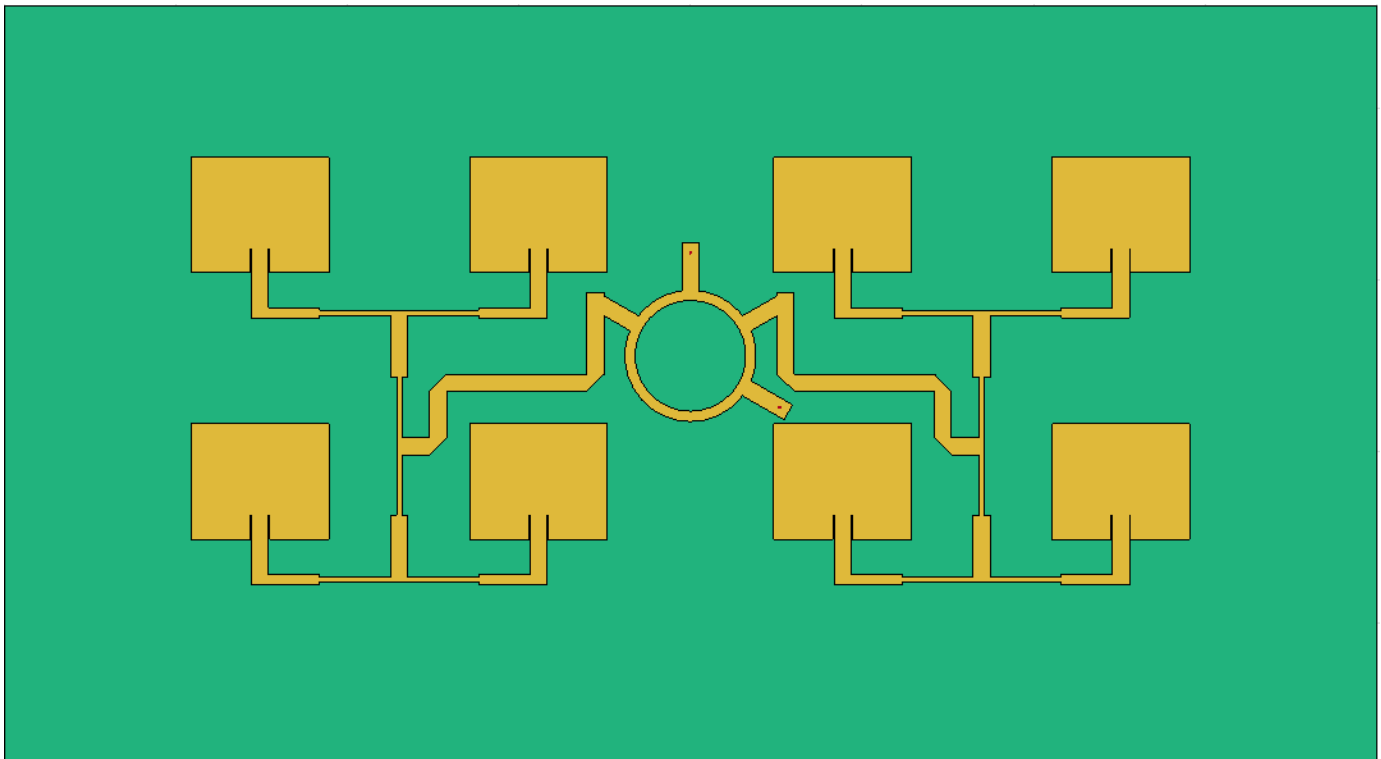
[1] Cui, Yaoshen, Haifan Yin, Li Tan, and Marco Di Renzo., " A 3D Positioning-based Channel Estimation Method for RIS-aided mmWave Communications." arXiv:2203.14636 (April 21, 2022). <https://doi.org/10.48550/arXiv.2203.14636>.

Design S-Band Monopulse Tracking RADAR Antenna

This example shows the integration of a 2-by-4 Microstrip patch array and a Rat-race coupler to implement a monopulse operation.

The system is designed for an operating frequency of 3 GHz and the simulated and measured results are compared in this example. The maximum gain of 15.3 dB is achieved at the sum port.

Below diagram shows the top view of the antenna.



Create Antenna Array Geometry

Use `couplerRatrace` object and assign appropriate values to the properties such that it resonates at 3 GHz.

```
%create RatRace by assigning the values to couplerRatrace component
%create RatRace
cup = couplerRatrace;
d = cup.Substrate;
cup.PortLineLength = 18.6e-3;
cup.PortLineWidth = 0.0050;
cup.CouplerLineWidth = 0.0030;
cup.Circumference = 0.111;
```

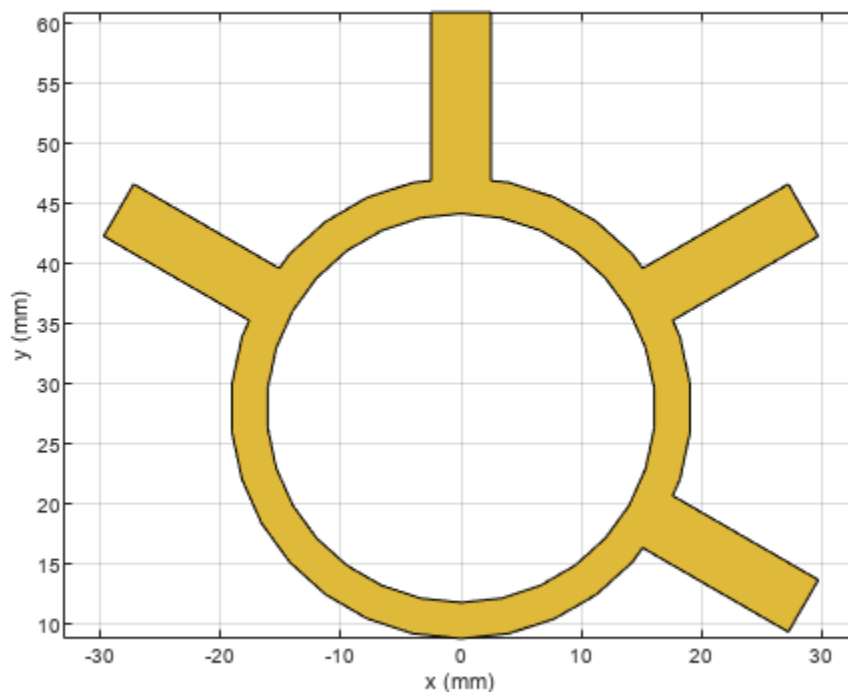
Use the `pcbComponent` object to form a pcb stack of the Rat-race coupler such that the ground plane can be extended to accommodate the patch antenna array.

```
RR = pcbComponent(cup);
RR.BoardShape = antenna.Rectangle("Center",[0 0.02],"Length",0.4,"Width",0.22);
```

```
RR.Layers{3} = RR.BoardShape;
ground = RR.BoardShape;
```

Re-alignment of the ports of the Rat-race coupler is required to arrange the ratrace coupler at the center of the patch antennas such that the distance between the patch antennas on both sides is same. Use `antenna.Rectangle` to create the the rectangle shapes and use Boolean add and subtract operations to modify the Rat-race coupler as required in this design. Use `show` method to visualize the structure.

```
top = RR.Layers{1};
cut1 = antenna.Rectangle("Center", [34.4e-3 0e-3], "Length", 3e-3, "Width", 5e-3);
cut2 = antenna.Rectangle("Center", [-34.4e-3 0e-3], "Length", 3e-3, "Width", 5e-3);
top = top-cut1-cut2;
top = rotateZ(top, -30);
cutfine1 = antenna.Rectangle("Center", [0e-3, 35.9e-3], "Length", 7e-3, "Width", 6e-3);
top = top-cutfine1;
top = rotateZ(top, -30);
cutfine2 = antenna.Rectangle("Center", [35.9e-3, 0], "Length", 6e-3, "Width", 7e-3);
top = top-cutfine2;
top = rotateZ(top, 30);
top = translate(top, [0, 28e-3, 0]);
figure
show(top);
```



Create patch antenna

Use `patchMicrostripInsetfed` object to create the patch antenna array. Assign the values for the properties of the antenna such that it resonates at 3 GHz.

```
ant = patchMicrostripInsetfed;
ant.Length = 33.6309e-3;
ant.Width = 40.1597e-3;
ant.Height = 1.6e-3;
ant.Substrate = d;
ant.NotchLength = 7e-3;
ant.GroundPlaneLength = 0.0600

ant =
    patchMicrostripInsetfed with properties:

        Length: 0.0336
        Width: 0.0402
        Height: 0.0016
        Substrate: [1x1 dielectric]
        PatchCenterOffset: [0 0]
        FeedOffset: [-0.0300 0]
        StripLineWidth: 1.0000e-03
        NotchLength: 0.0070
        NotchWidth: 0.0030
        GroundPlaneLength: 0.0600
        GroundPlaneWidth: 0.0600
        Conductor: [1x1 metal]
        Tilt: 0
        TiltAxis: [1 0 0]
        Load: [1x1 lumpedElement]

ant.StripLineWidth = 5e-3;
ant.NotchWidth = 5.577e-3;
```

Create antenna array

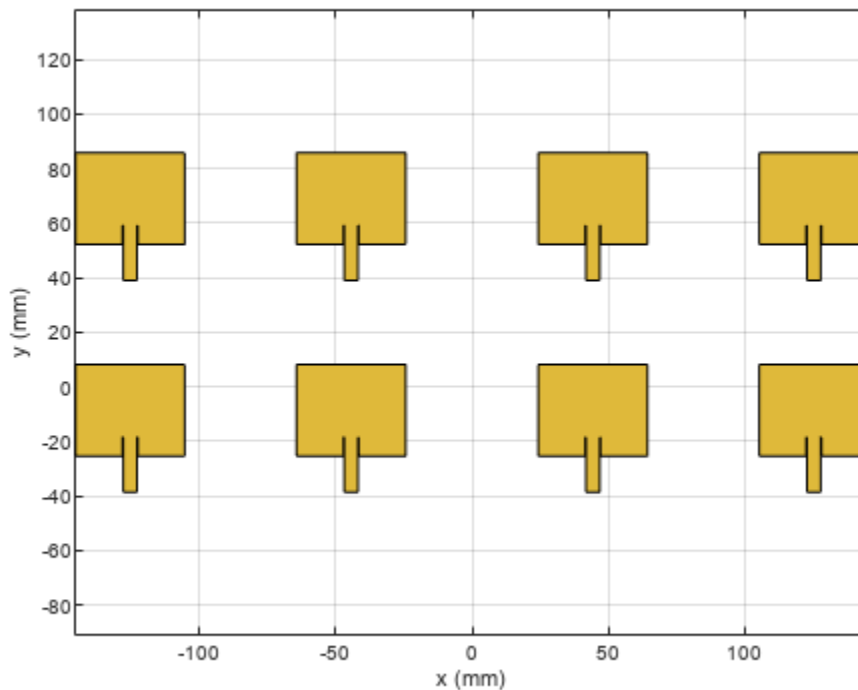
Use `pcbStack` to convert the antenna into a PCB stack and copy the designed antenna element to create 8 similar elements. Use `translate` function to position the patch antennas on both the sides of the Rat-race coupler to create the a 2-by-4 array antenna elements and visualize it using the `show` function.

```
array = pcbStack(ant);
a1 = array.Layers{1};
a1 = rotateZ(a1,90);
a2 = copy(a1);
a3 = copy(a1);
a4 = copy(a1);
ant1 = translate(a1,[-125.5202e-3,69e-3,0]);
ant2 = translate(a2,[-125.5202e-3,-0.008650,0]);
ant3 = translate(a3,[-44.3591e-3,69e-3,0]);
ant4 = translate(a4,[-44.3591e-3,-0.008650,0]);
antArrayleft = ant1+ant3+ant2+ant4;
antArrayright = copy(antArrayleft)

antArrayright =
    Polygon with properties:
```

```
Name: 'mypolygon'
Vertices: [331x3 double]
```

```
antArrayright = mirrorY(antArrayright);
antArray = antArrayleft+antArrayright;
show(antArray);
```



Use `antenna.Rectangle` to create a part of the power divider network. Create the power divider using `traceTee` shape. Use `copy` and `mirrorX` operation to create 4 power dividers with same dimensions.

```
inline1 = antenna.Rectangle("Center", [-54.29e-3, 0.0405], "Length", 15e-3, "Width", 2.9e-3);
inline2 = antenna.Rectangle("Center", [-115.6e-3, 0.0405], "Length", 15e-3, "Width", 2.9e-3);
inline3 = antenna.Rectangle("Center", [-54.29e-3, -37.1475e-3], "Length", 15e-3, "Width", 2.9e-3);
inline4 = antenna.Rectangle("Center", [-115.6e-3, -37.1475e-3], "Length", 15e-3, "Width", 2.9e-3);
tee = traceTee;
tee.Length = [46.3746e-3 18e-3];
tee.Width = [1.492e-3, 5e-3];
tee1 = copy(tee);
tee2 = copy(tee);
tee3 = copy(tee);
tee.ReferencePoint = [84.92e-3, 0.0405];
tee2 = mirrorX(tee2);
tee3 = mirrorX(tee3);
tee1.ReferencePoint = [-84.92e-3, 0.0405];
tee2.ReferencePoint = [84.92e-3, 37.1475e-3];
```

```

tee3.ReferencePoint = [-84.92e-3,37.1475e-3];
teecut = antenna.Circle("Center",[12e-3,12e-3],"Radius",3e-3,NumPoints=3);
ytee = traceTee;
ytee.Length = [45.3746e-3 9.12456e-3];
ytee.Width = [1.492e-3,5e-3];
yltee = copy(ytee);
ytee.ReferencePoint = [-1.675e-3,84.92e-3];
ytee = rotateZ(ytee,-90);
yltee.ReferencePoint = [1.675e-3,84.92e-3];
yltee = rotateZ(yltee,90);
fifine = antenna.Rectangle("Center", [-73.585e-3,9.175e-3],"Length",5e-3,"Width",20e-3);
fifine1 = antenna.Rectangle("Center", [-52.57e-3,20.105e-3],"Length",47e-3,"Width",5e-3);
fifine2 = antenna.Rectangle("Center", [-27.7e-3,32.6e-3],"Length",5e-3,"Width",30e-3);
curve = antenna.Polygon('Vertices',[-84.085e-3 -0.825e-3 0;-79.085e-3 -0.825e-3 0; ...
-79.085e-3 4.175e-3 0]);
curve = translate(curve,[8e-3,0,0]);
curve1 = antenna.Polygon('Vertices',[-84.085e-3 22.705e-3 0;-84.085e-3 17.605e-3 0; ...
-79.085e-3 22.705e-3 0]);
curve1 = translate(curve1,[8e-3,0,0]);
curve2 = antenna.Polygon('Vertices',[-33.2e-3 17.6e-3 0;-33.2e-3 22.605e-3 0; ...
-38.4e-3 17.6e-3 0]);
curve2=translate(curve2,[8e-3,0,0]);
sqcut = antenna.Rectangle("Center", [-26.7e-3,47.518e-3],"Length",7e-3,"Width",2.1640e-3);
sqcut2 = copy(sqcut);
sqcut2 = mirrorY(sqcut2);
RRline = fifine+fifine1+fifine2-curve-curve1-curve2;
feedLeft = teel+tee3+inlined1+inlined2+inlined3+inlined4+y1tee+RRline;
feedRight = copy(feedLeft);
feedRight = mirrorY(feedRight);
totaltop = top+feedLeft+feedRight+antArray-sqcut-sqcut2;

```

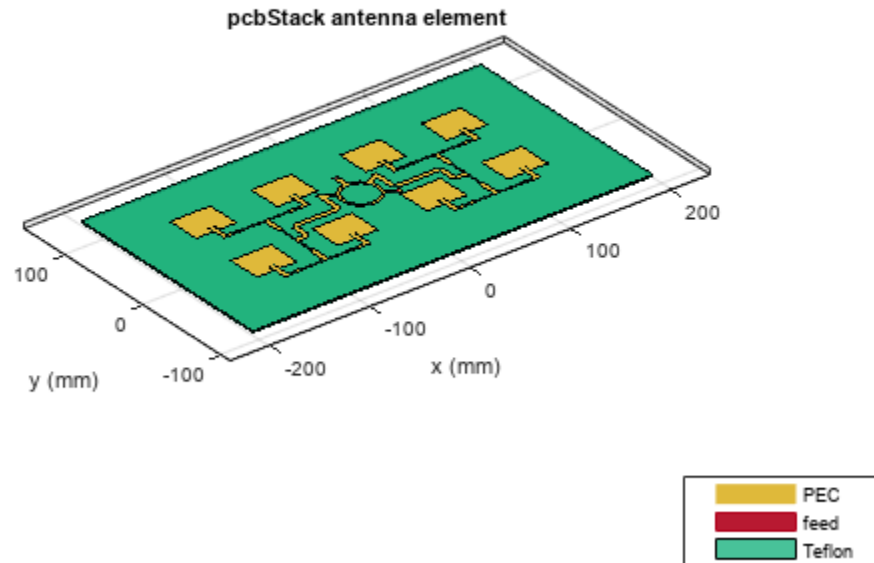
Create PCB antenna from geometry

Use the pcbStack object to create a stack of the layers of the antenna and assign the shape created in the previous section to the top layer of the pcbStack. Thus the design of the patch antenna is complete.

```

RRmicrostrip = pcbStack;
RRmicrostrip.Layers{1} = totaltop;
RRmicrostrip.BoardThickness = 0.0016;
RRmicrostrip.BoardShape = antenna.Rectangle("Center",[0 0.02],"Length",0.4,"Width",0.22);
RRmicrostrip.Layers = {totaltop,d,ground};
feedloc = [[0 58e-3 3 1];[26e-3 13e-3 3 1]];
RRmicrostrip.FeedLocations = feedloc;
RRmicrostrip.FeedDiameter = 0.005/8;
RRmicrostrip.FeedViaModel = 'strip';
RRmicrostrip.FeedPhase = 0;
show(RRmicrostrip )

```



Analyze the Antenna

Set the `FeedVoltage` property of `pcbStack` to `[1 0]` to feed a single port amongst two.

```
%To get the sum port characteristics
RRmicrostrip.FeedVoltage = [1 0];
%To get the Difference port characteristics
%RRmicrostrip.FeedVoltage = [0 1];
```

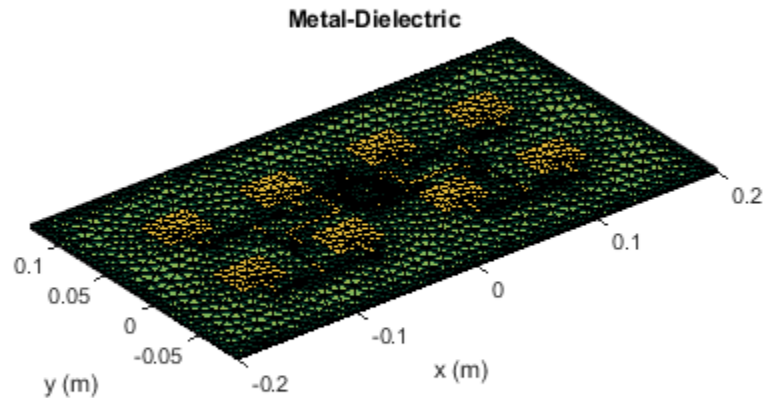
Use `mesh` function to manually mesh the array antenna. Use `MaxEdgeLength` equal to $\lambda/8$ times the center frequency to get a finer mesh where λ represents the wavelength.

```
plotfrequency = 3.3*1e9;
lambda = 3e8/plotfrequency;
mesh(RRmicrostrip, 'MaxEdgeLength', lambda/8);
```

```

NumTriangles: 3994
NumTetrahedra: 9444
NumBasis:
MaxEdgeLength: 0.011364
MeshMode: manual

```



Estimate memory requirement

Use the `memoryEstimate` function to calculate the memory required to solve the structure.

```
memEst = memoryEstimate(RRmicrostrip,3e9);
```

Calculate S-parameters

Use the `sparameters` function to calculate the S-parameters.

```

frequencyRange = (2.5:0.1:3.5)*1e9;
%s = sparameters(RRmicrostrip,frequencyRange);
%rfplot(s)

```

To save simulation time, the code for calculating and plotting the s-parameters is commented out and a pre-computed result is provided in the `RRmicrostripdata.mat` file. You can uncomment the code and use the freshly computed results instead of the MAT-file for the analysis.

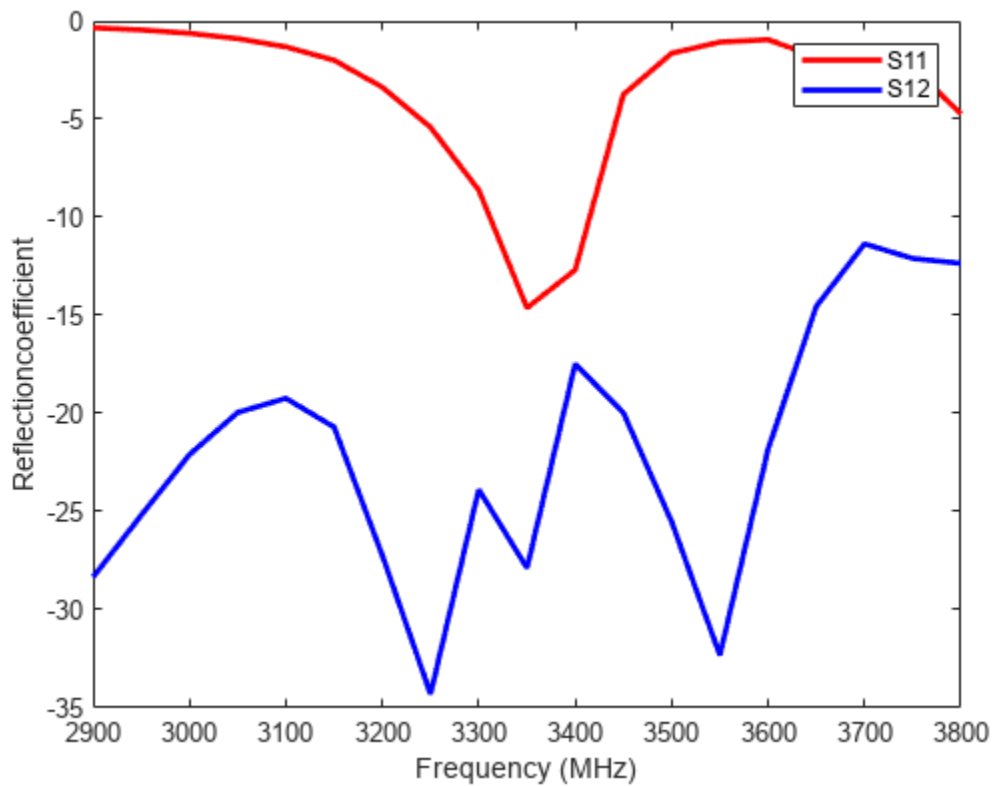
Load the `RRmicrostripdata.mat` file. It contains the simulated and measured data stored in four variables namely `S11_simulated`, `S11_measured`, `patternsum_simulated`, `patternsum_measured`, `patterndiff_simulated` and `patterndiff_measured`. Use `S11_simulated` to plot the s-parameters.

```

load sBandmonopulse.mat
figure;
plot(simufreq,S11_simulated,'r',LineWidth=2);
hold on;

```

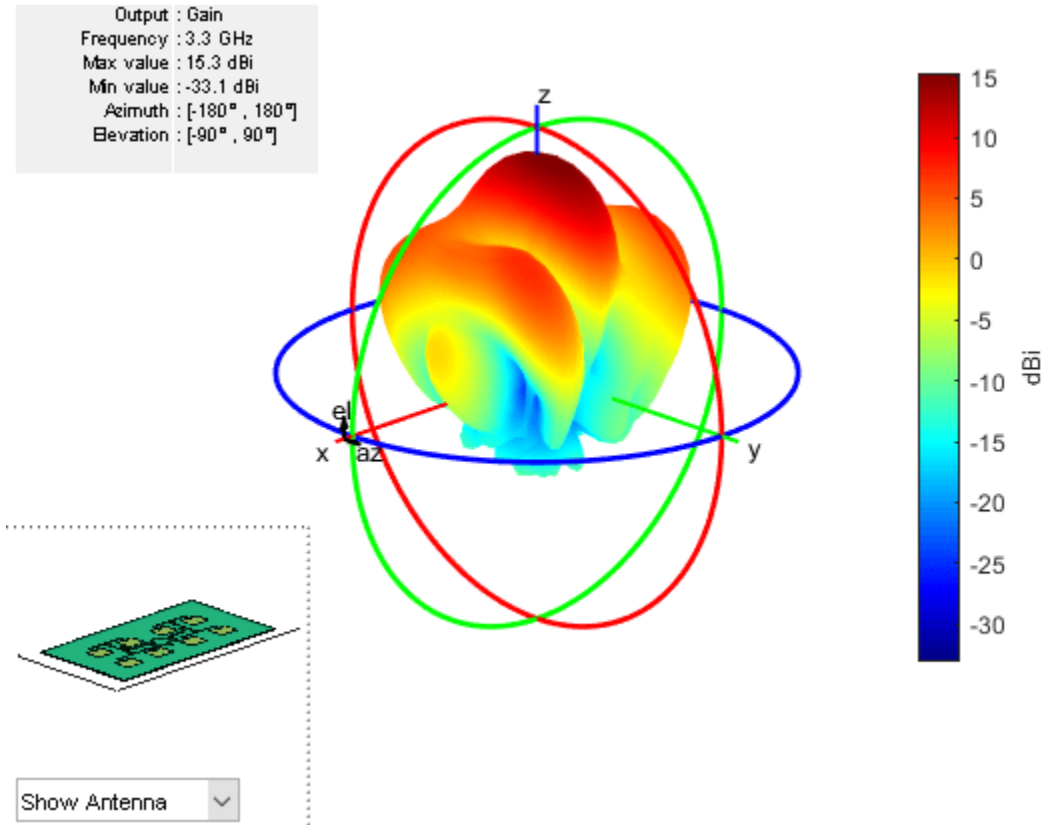
```
plot(simufreq,S12_simulated,'b',LineWidth=2);  
hold off;  
legend('S11','S12');  
xlabel('Frequency (MHz)');  
ylabel('Reflectioncoefficient');
```



Plot radiation pattern

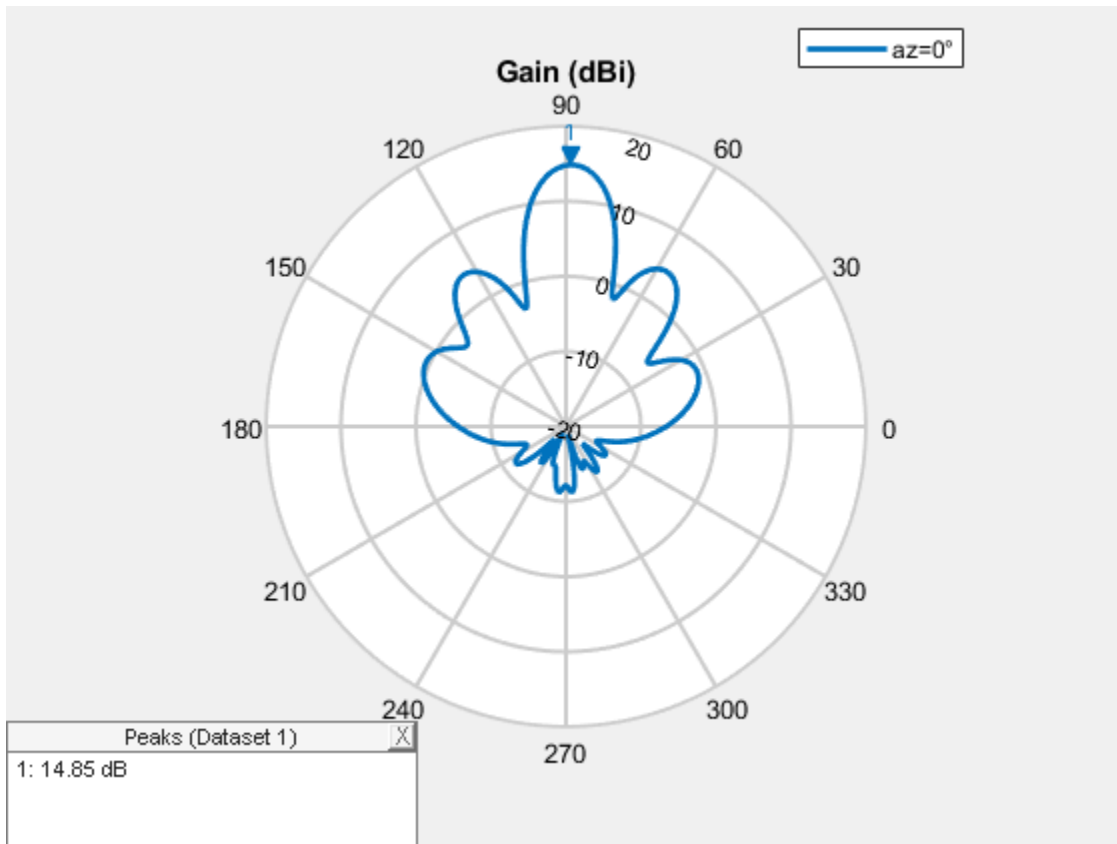
Use the pattern function to plot the 3D radiation Pattern at 3.3GHz.

```
figure;  
pattern(RRmicrostrip,3.3e9);
```



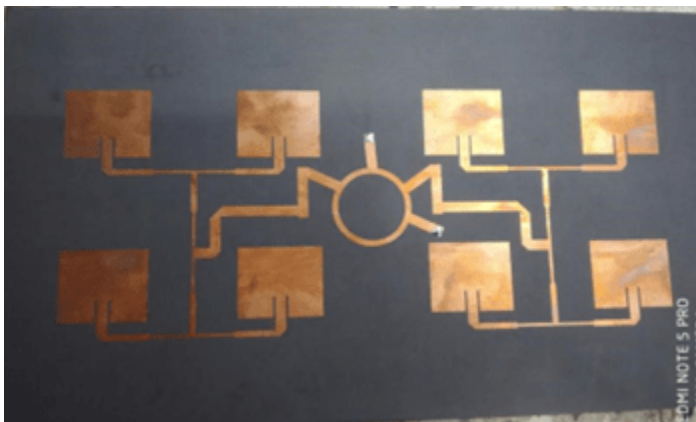
Use the `Pattern` function to plot the 2D radiation pattern at an elevation angle of 0 deg.

```
figure;  
pattern(RRmicrostrip,plotfrequency,0,0:1:360);
```



The designed array antenna is fabricated as shown below and it is tested for the reflection coefficient and the radiation pattern at both sum and difference ports. Reflection coefficient of fabricated array antenna is measured on Network Analyzer. Radiation pattern measurements are performed at an anechoic chamber.

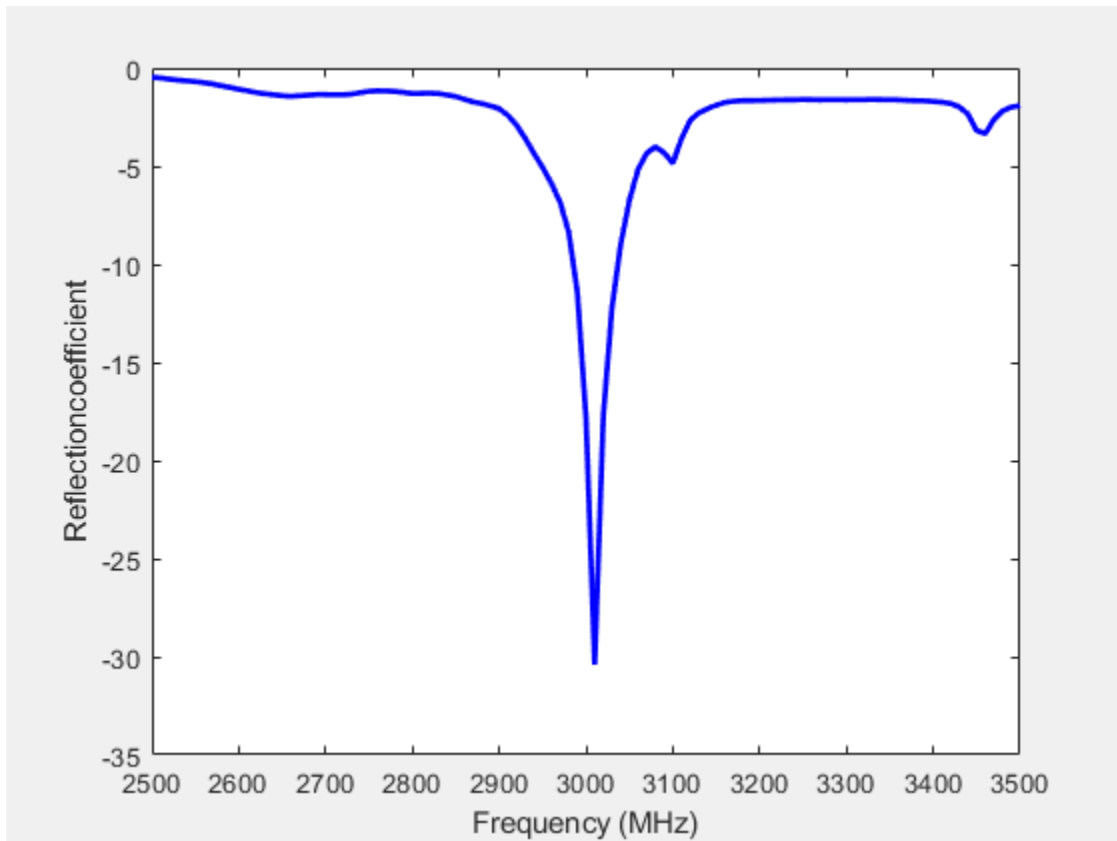
Top view of the fabricated antenna is shown below.



Measured results

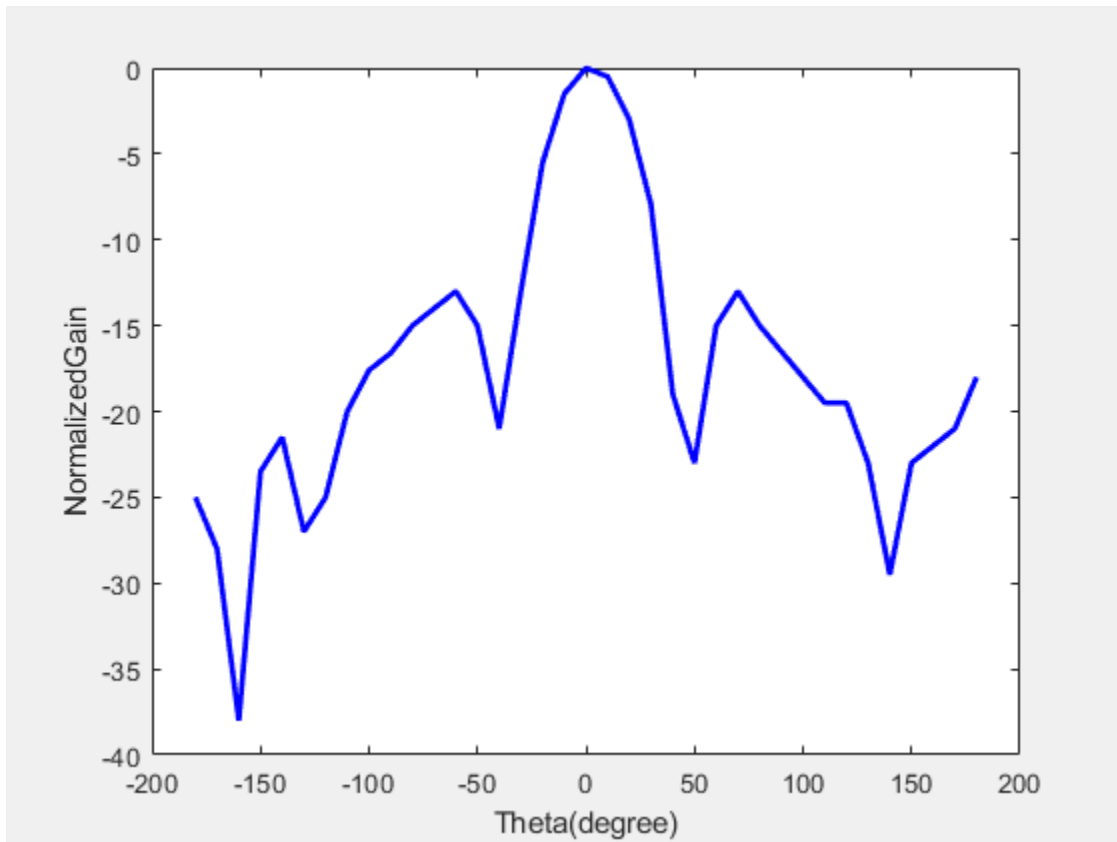
Create the comparison graph for measured reflection coefficient.


```
plot(measfreq,S11_measured','b',LineWidth=2);  
xlabel('Frequency (MHz)');  
ylabel('Reflectioncoefficient');
```



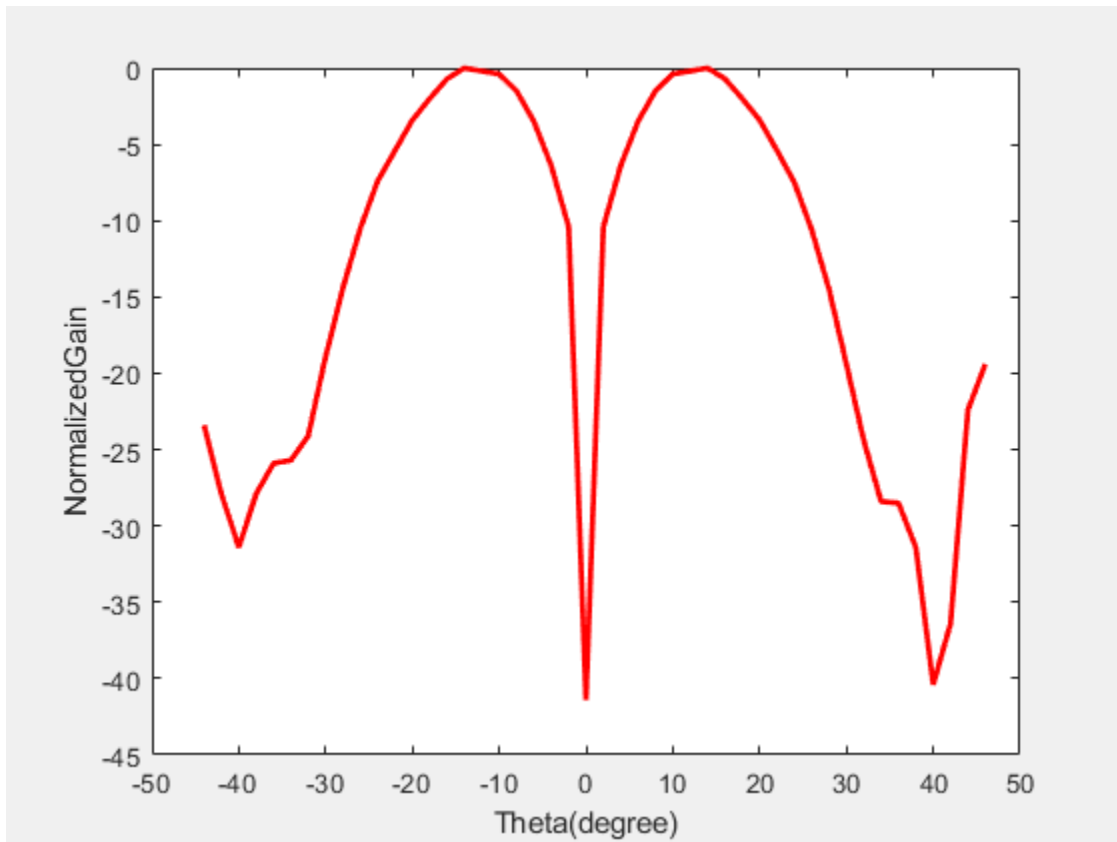
Create the graph for measured sum radiation pattern.

```
plot(-180:10:180,sumpattern_measured','b',LineWidth=2);  
xlabel('Theta (degree)');  
ylabel('NormalizedGain');
```



Create the graph for measured difference radiation pattern.

```
plot(-44:2:46,diffpattern_measured,'r',LineWidth=2);  
xlabel('Theta(degree)');  
ylabel('NormalizedGain');
```



Conclusion

The designed antenna has a maximum gain of 15.3 dB sum port gain and 20 dB null depth at difference port at 3GHz. The efficiency of the designed array antenna is confirmed by measurements. The measured and simulated results agree well. The designed array antenna is a promising candidate for monopulse tracking radar at S-band.

Reference

[1] H. Kumar and G. Kumar, "Microstrip antenna array with ratrace comparator at X-band for monopulse tracking radar," *2016 IEEE International Symposium on Antennas and Propagation (APSURSI)*, 2016, pp. 513-514, doi: 10.1109/APS.2016.7695965.

