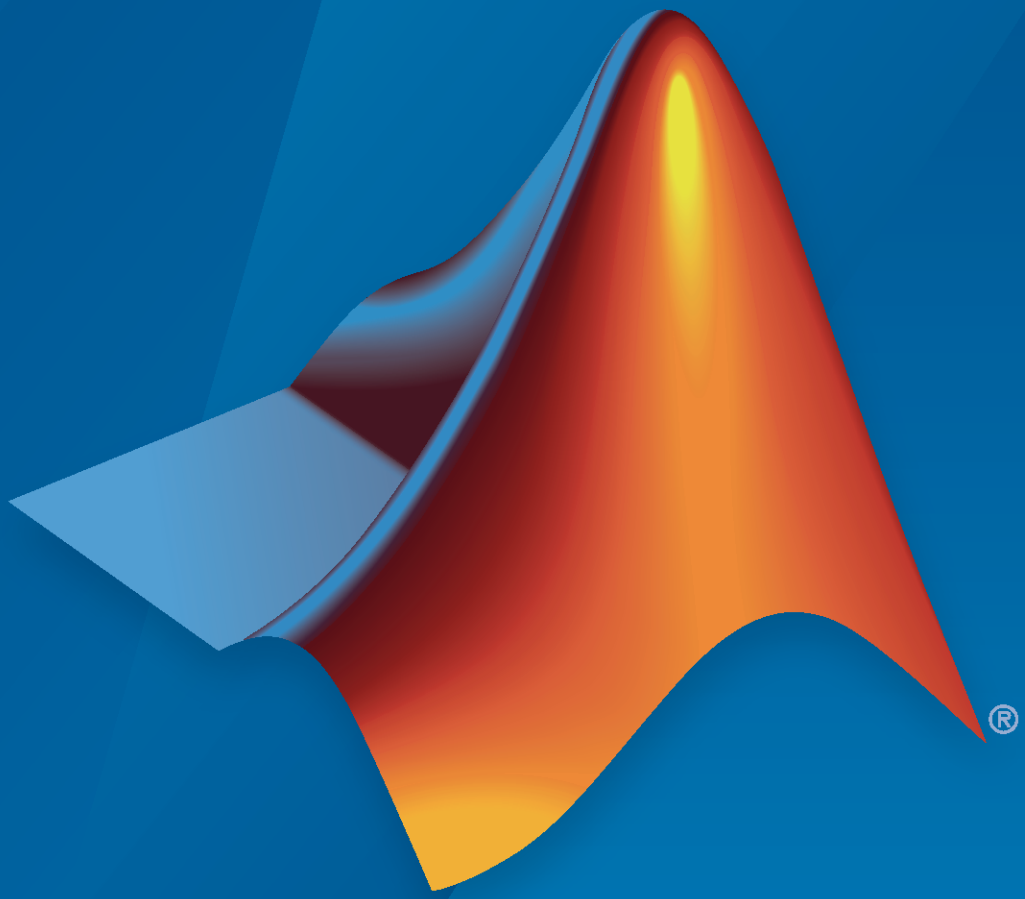
RF PCB Toolbox™

Getting Started



MATLAB®

R2023a

MathWorks®

How to Contact MathWorks

Latest news: `www.mathworks.com`

Sales and services: `www.mathworks.com/sales_and_services`

User community: `www.mathworks.com/matlabcentral`

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

Phone: 508-647-7000

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

RF PCB Toolbox™ Getting Started Guide

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

September 2021	Online only	New for Version 1.0 (R2021b)
March 2022	Online only	Revised for Version 1.1 (R2022a)
September 2022	Online only	Revised for Version 1.2 (R2022b)
March 2023	Online only	Revised for Version 1.3 (R2023a)

Contents

1 Getting Started with RF PCB Toolbox

2 RF PCB Tutorials

3 RF PCB Concepts

1

Getting Started with RF PCB Toolbox

RF PCB Toolbox Product Description

Perform electromagnetic analysis of printed circuit boards

RF PCB Toolbox™ provides functions and apps for designing, analyzing, and visualizing high-speed and RF multi-layer printed circuit boards (PCBs). You can design components with parameterized or arbitrary geometry, including distributed passive structures such as traces, bends, and vias. Using the frequency-domain method of moments and other EM techniques, you can model coupling, dispersion, and parasitic effects.

With RF PCB Toolbox, designers of RF boards, modules, MMICs, and SiPs can predict PCB performance and verify that the manufactured PCB meets specifications. For RF and antenna designers, the toolbox provides parameterized models of distributed filters, couplers, splitters, matching networks, and Gerber file generation. Toolbox support for ODB++ and databases from Cadence® Allegro®, Mentor Expedition, Altium®, and Zuken enables signal integrity engineers to analyze the high-speed portions of the PCB layout.

RF PCB Tutorials

Rat-Race Coupler - Visualize and Analyze

This example shows you how to create, visualize, and analyze a Rat-Race Coupler.

Create a rat-race coupler with default properties.

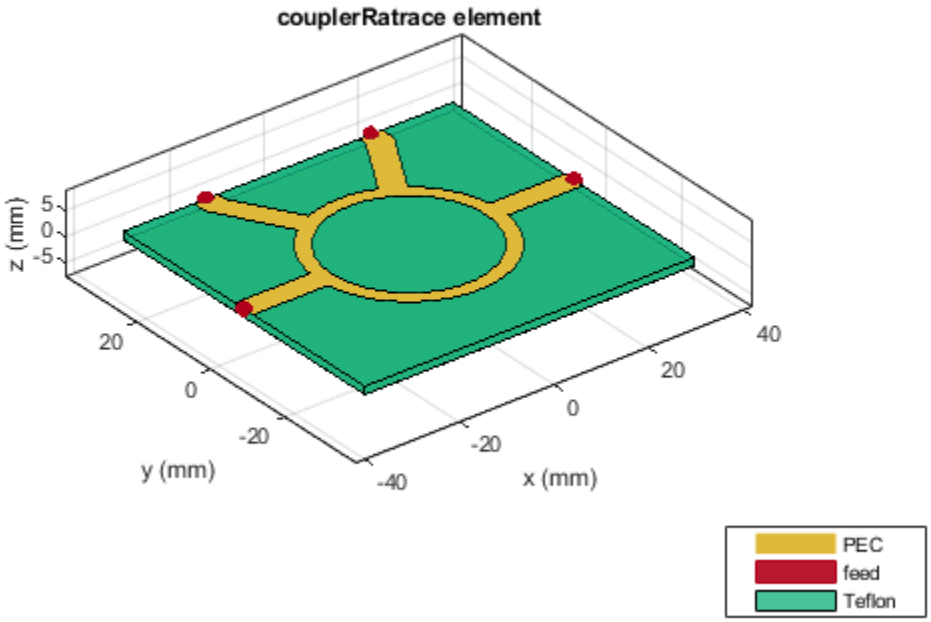
```
ratrace = couplerRatrace
```

```
ratrace =
  couplerRatrace with properties:

      PortLineLength: 0.0186
       PortLineWidth: 0.0050
    CouplerLineWidth: 0.0030
       Circumference: 0.1110
              Height: 0.0016
           Substrate: [1×1 dielectric]
           Conductor: [1×1 metal]
```

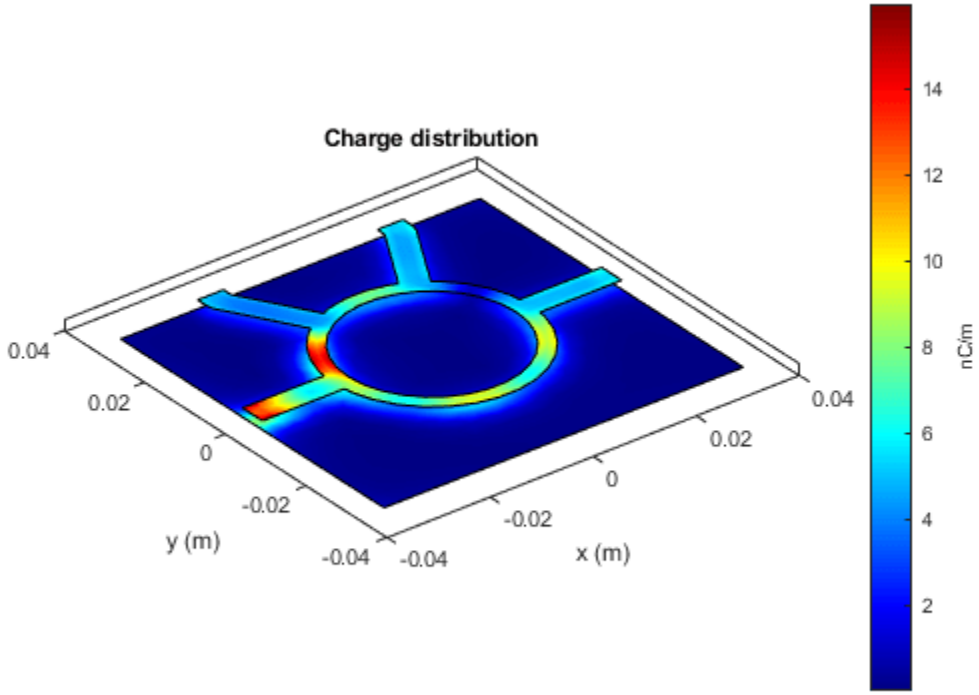
View the coupler.

```
show(ratrace)
```



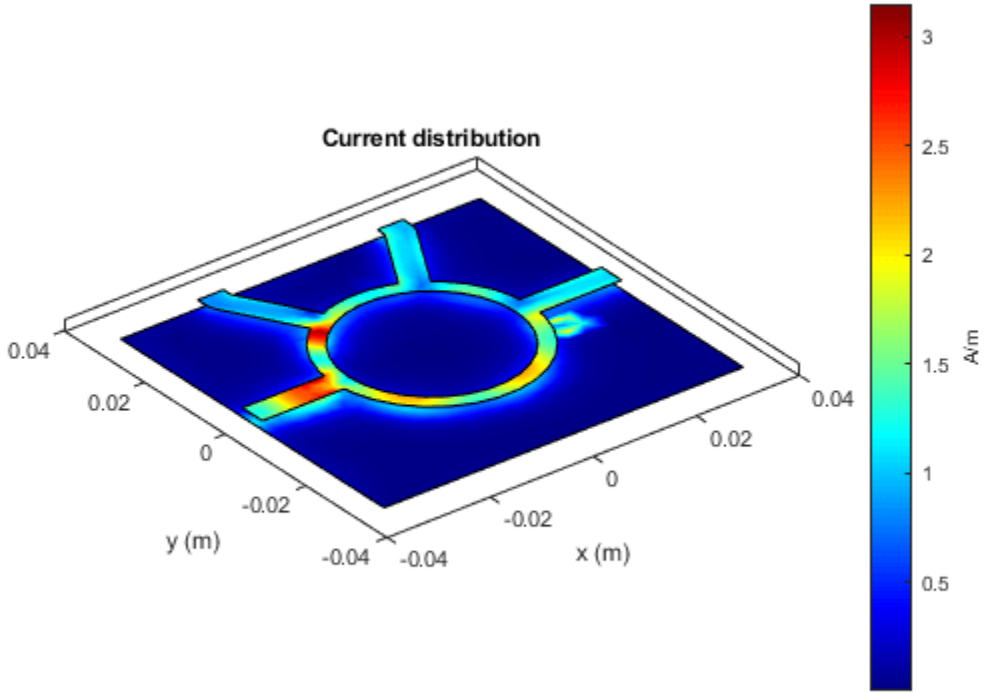
Plot the charge distribution at 5 GHz.

```
charge(ratrace, 5e9)
```



Plot the current distribution at 5 GHz.

```
figure
current(ratrace, 5e9)
```



Calculate and plot the s-parameters.

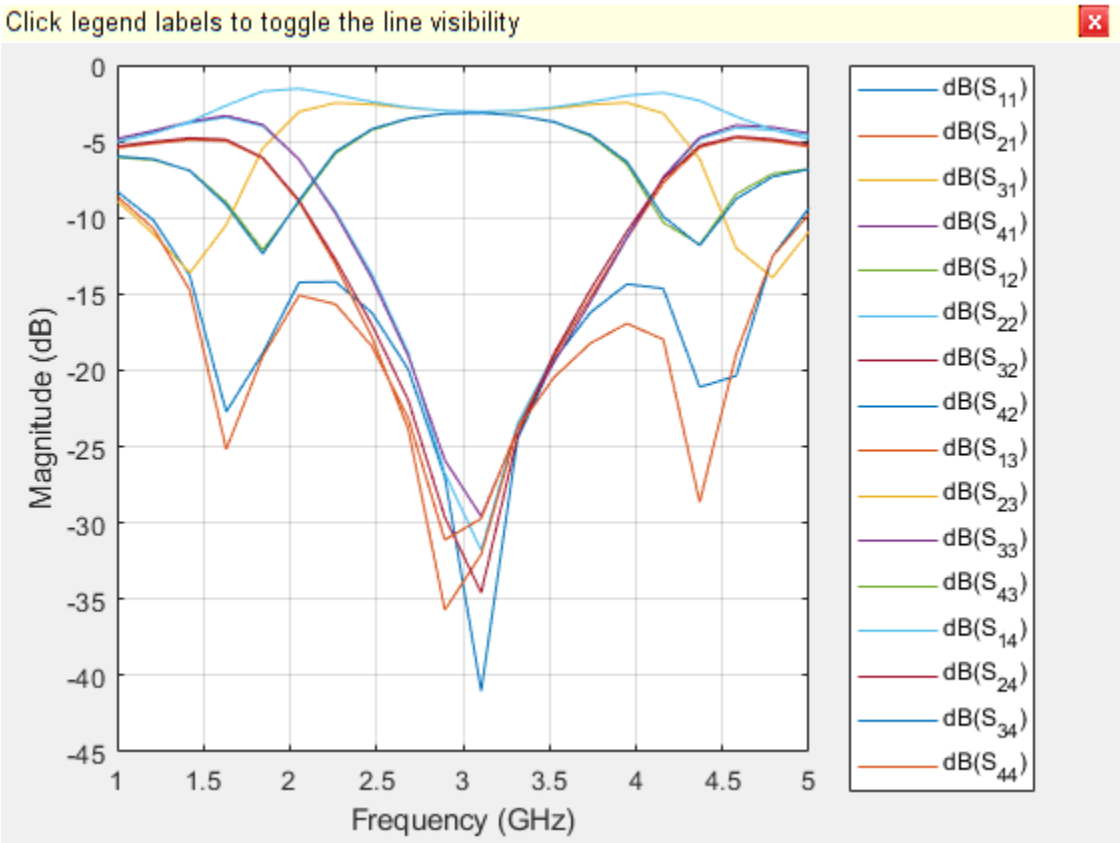
```
spar = sparameters(ratrace,linspace(1e9,5e9,20))
```

```
spar =
  sparameters: S-parameters object

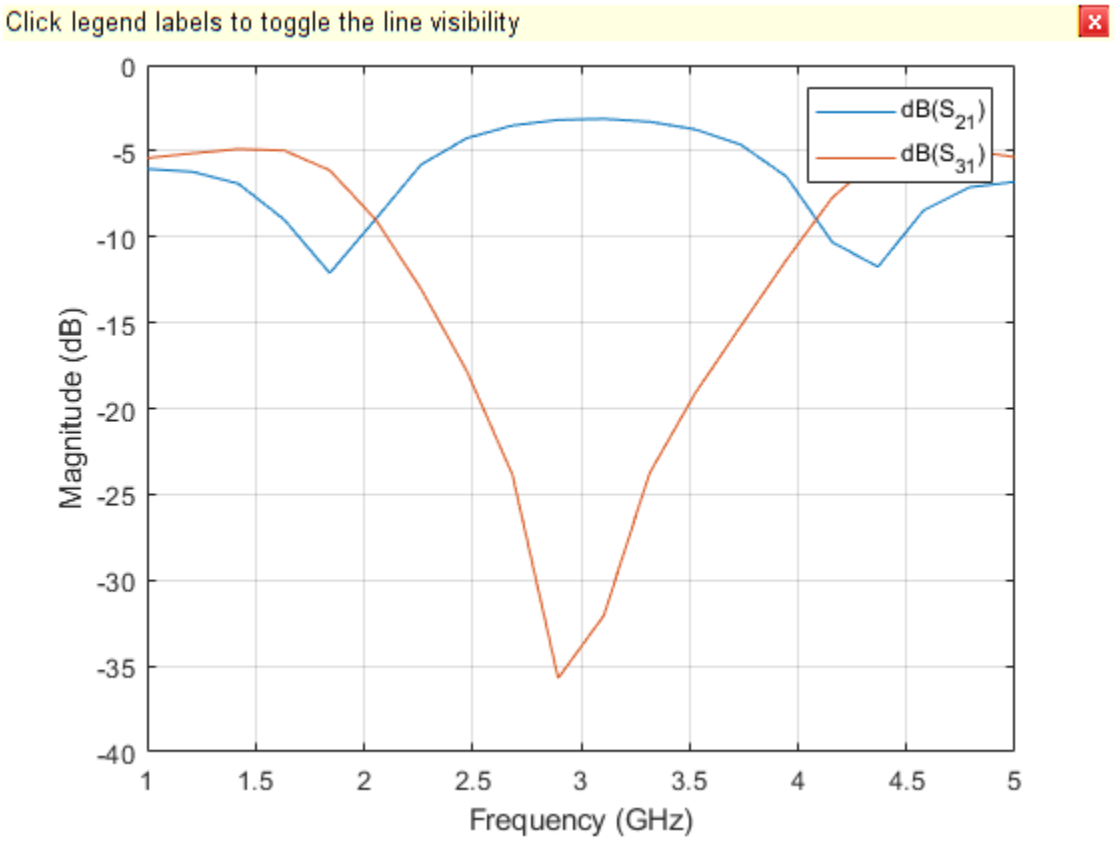
       NumPorts: 4
    Frequencies: [20×1 double]
     Parameters: [4×4×20 double]
      Impedance: 50

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

```
rfplot(spar)
```




```
figure
rfplot(spar,[2 3],1)
```



Filter Coupled Line - Visualize and Analyze

This example shows you how to create, visualize, and analyze a coupled line filter.

Create a coupled line filter with default properties.

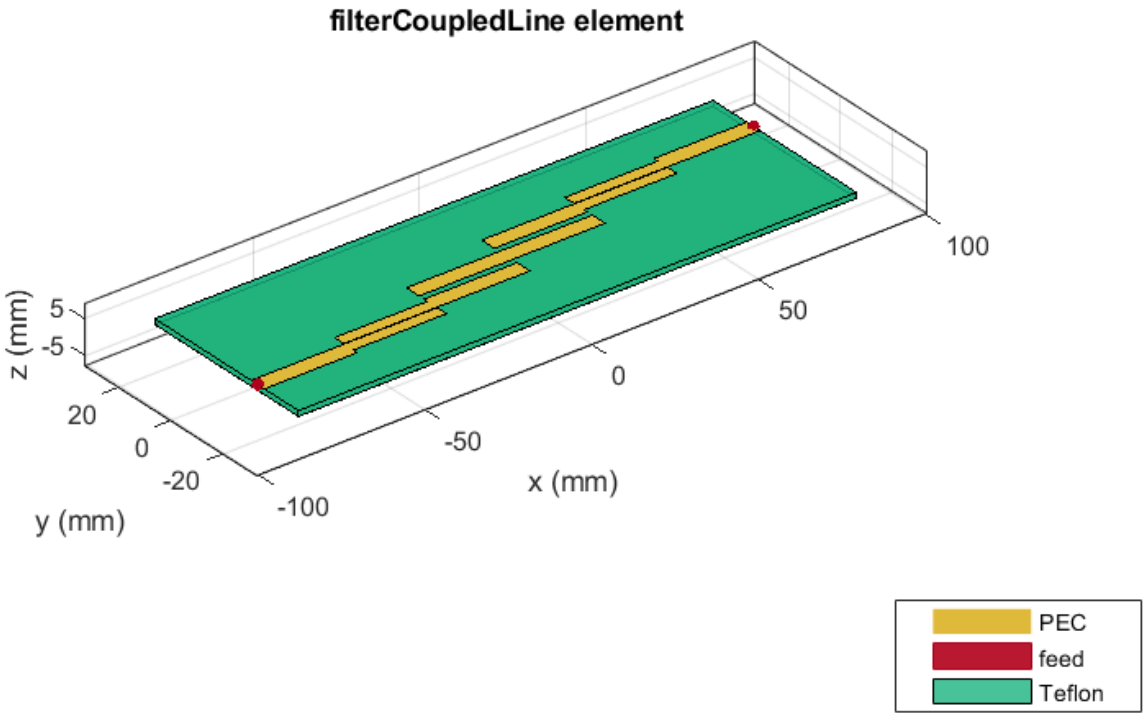
```
filter = filterCoupledLine
```

```
filter =
  filterCoupledLine with properties:

           FilterOrder: 3
        PortLineLength: 0.0279
         PortLineWidth: 0.0051
     CoupledLineLength: [0.0279 0.0279 0.0279 0.0279]
      CoupledLineWidth: [0.0036 0.0049 0.0049 0.0036]
    CoupledLineSpacing: [1.8270e-04 0.0019 0.0019 1.8270e-04]
                Height: 0.0016
      GroundPlaneWidth: 0.0551
             Substrate: [1×1 dielectric]
             Conductor: [1×1 metal]
```

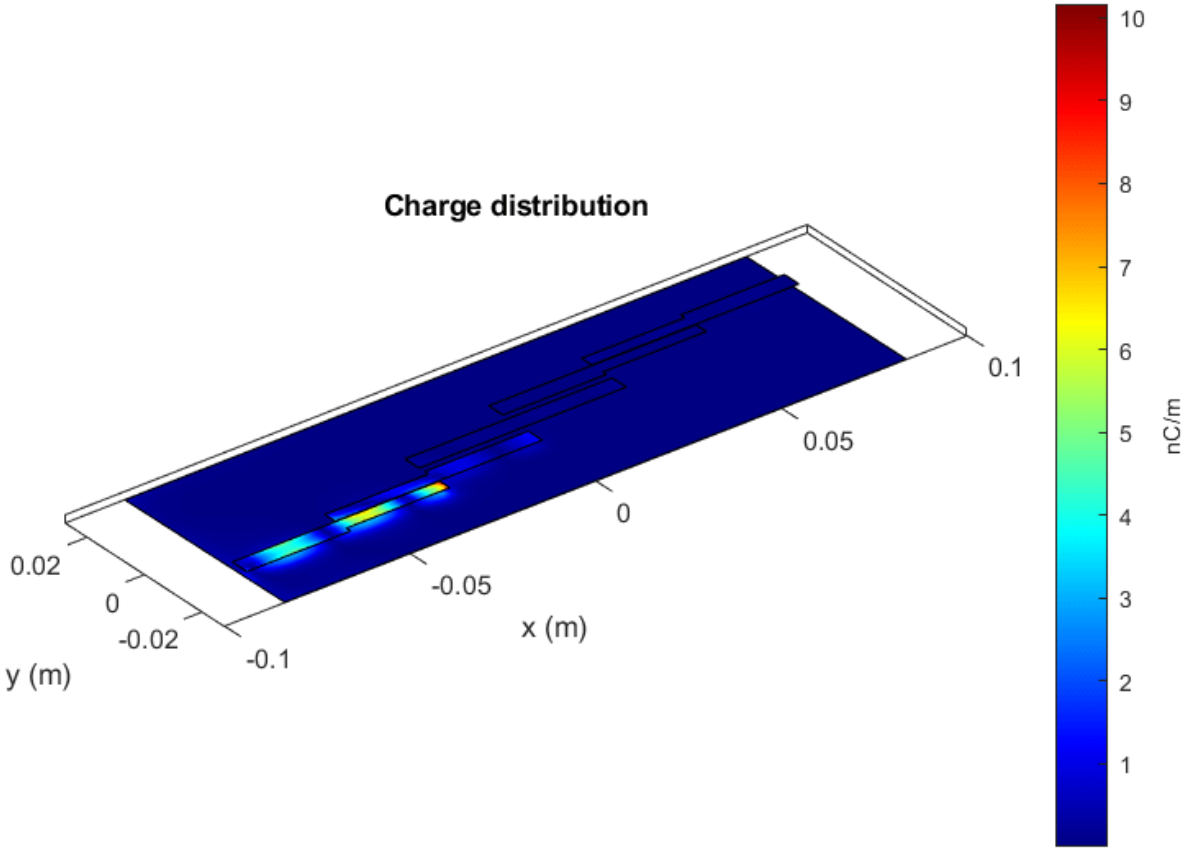
View the filter.

```
show(filter)
```



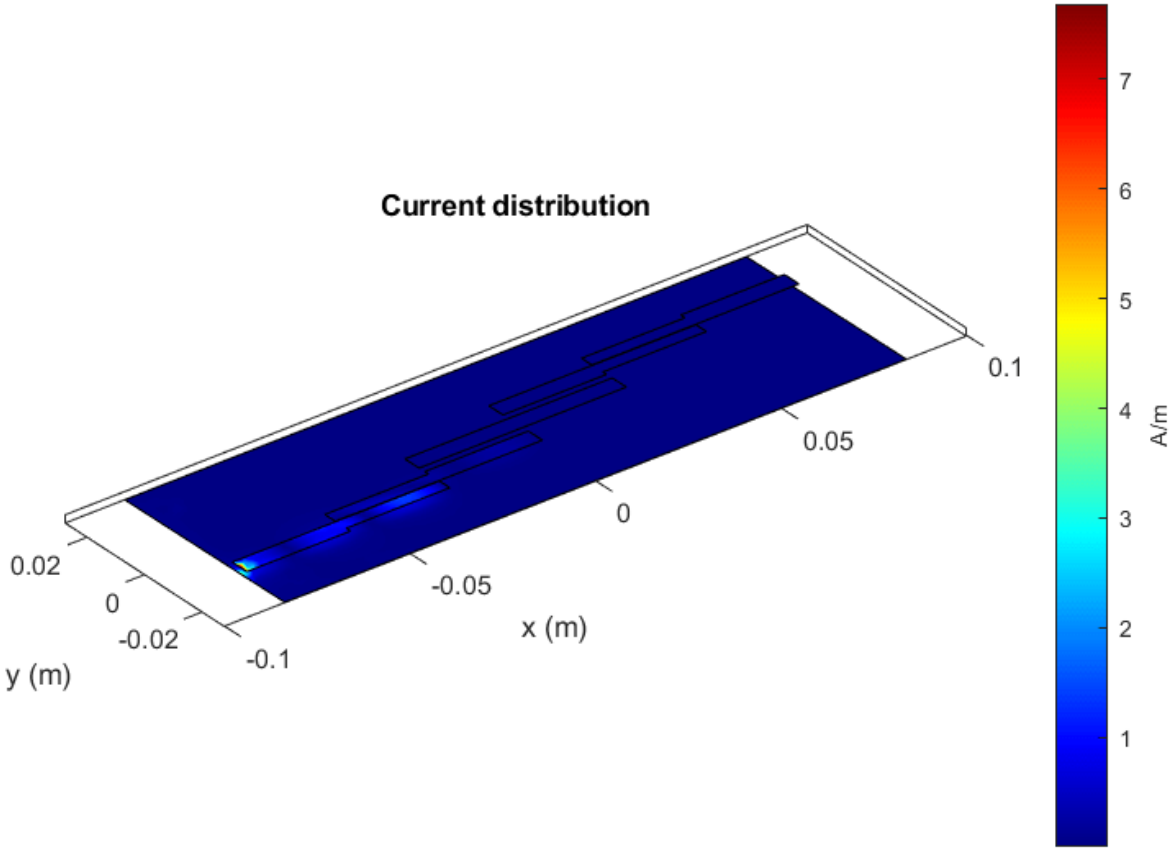
Plot the charge distribution at 5 GHz.

```
charge(filter,5e9)
```



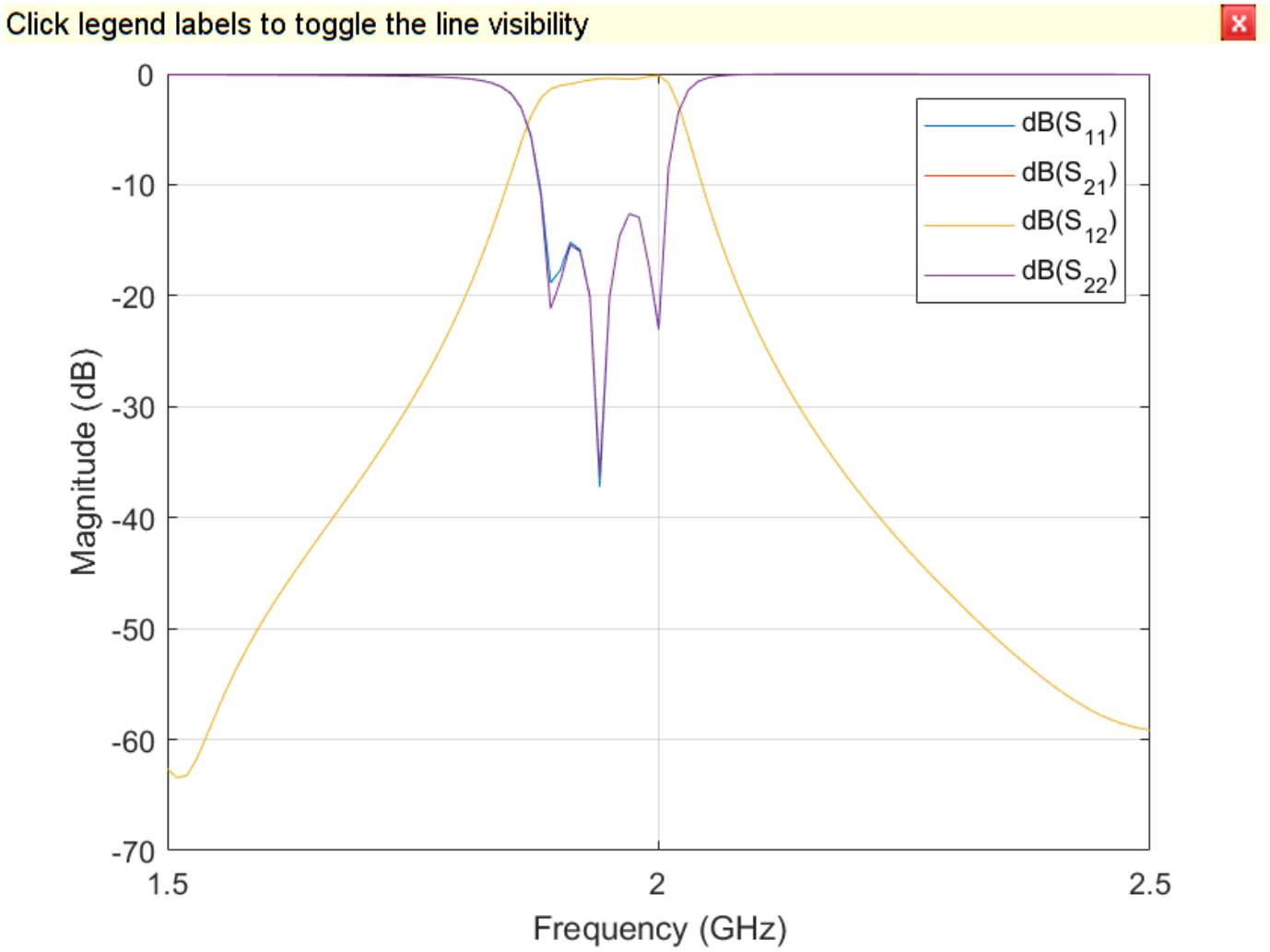
Plot the current distribution at 5 GHz.

```
figure
current(filter,5e9)
```



Calculate and plot the s-parameters.

```
spar = sparameters(filter,linspace(1.5e9,2.5e9,101));
figure,rfplot(spar)
```


Click legend labels to toggle the line visibility
0
-10
-20
-30
-40
-50
-60
-70
Magnitude (dB)
1.5
2
2.5
Frequency (GHz)
dB(S11)
dB(S21)
dB(S12)
dB(S22)

Interdigital Capacitor - Visualize and Analyze

This example shows you how to create, visualize, and analyze an interdigital capacitor.

Create an interdigital capacitor with default properties.

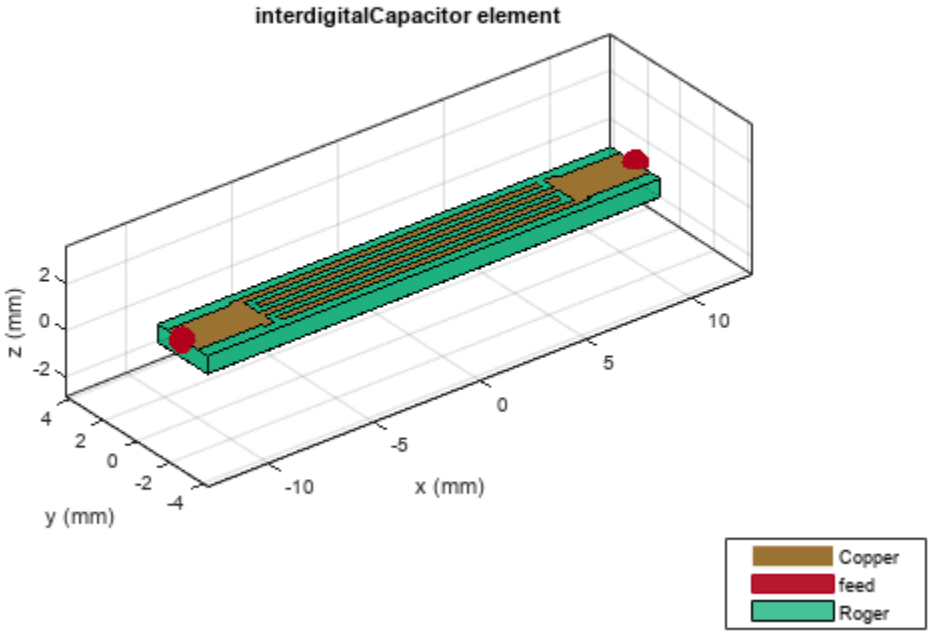
```
capacitor = interdigitalCapacitor
```

```
capacitor =
  interdigitalCapacitor with properties:

           NumFingers: 4
         FingerLength: 0.0137
          FingerWidth: 3.1600e-04
        FingerSpacing: 3.0000e-04
        FingerEdgeGap: 3.4100e-04
    TerminalStripWidth: 5.0000e-04
         PortLineWidth: 0.0019
        PortLineLength: 0.0030
                Height: 7.8700e-04
      GroundPlaneWidth: 0.0030
             Substrate: [1x1 dielectric]
             Conductor: [1x1 metal]
```

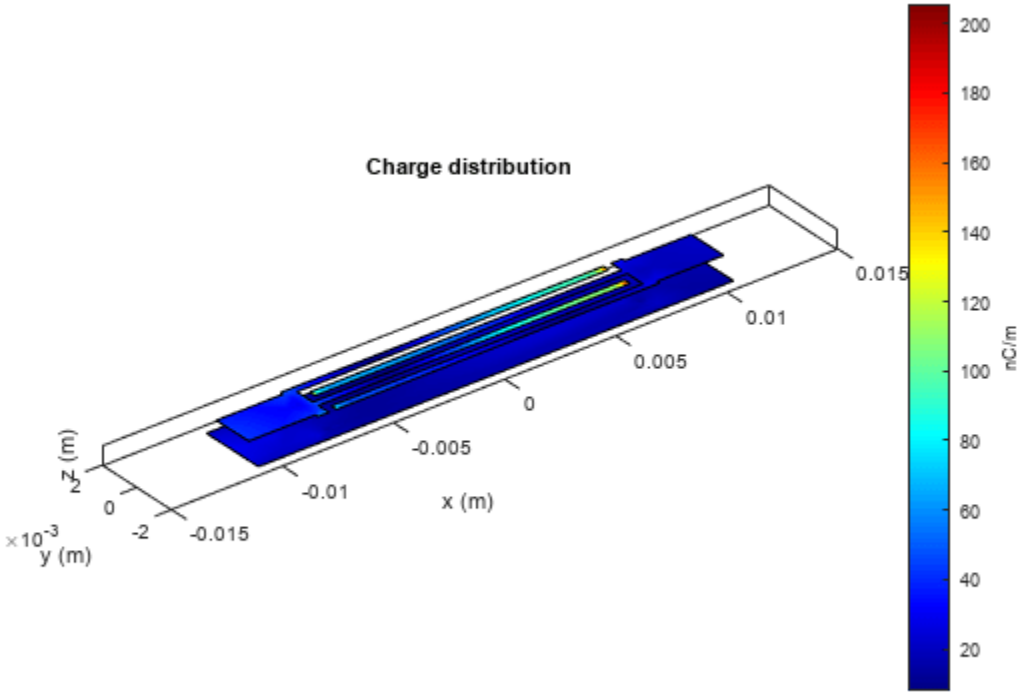
View the capacitor.

```
show(capacitor)
```



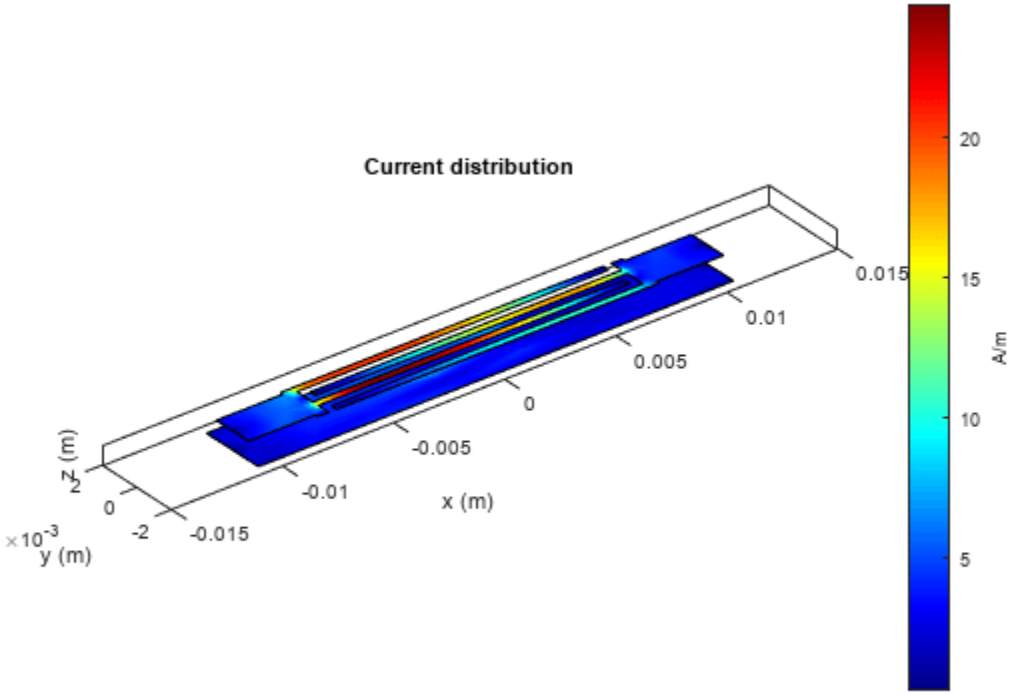
Plot the charge distribution at 5 GHz.

```
charge(capacitor, 5e9)
```



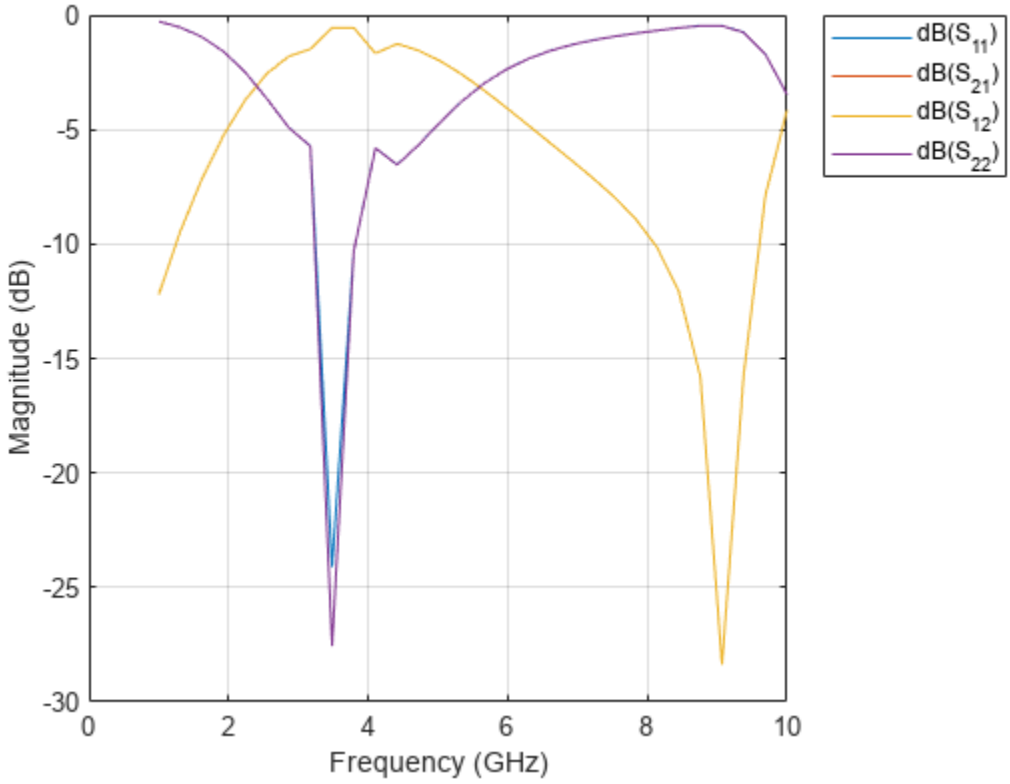
Plot the current distribution at 5 GHz.

```
figure
current(capacitor, 5e9)
```



Calculate and plot the s-parameters.

```
spar = sparameters(capacitor,linspace(1e9,10e9,30));
rfplot(spar)
```


0
-5
-10
-15
-20
-25
-30
Magnitude (dB)
0
2
4
6
8
10
Frequency (GHz)
dB(S11)
dB(S21)
dB(S12)
dB(S22)

Spiral Inductor - Visualize and Analyze

This example shows you how to create, visualize, and analyze a spiral inductor.

Create a spiral inductor with default properties.

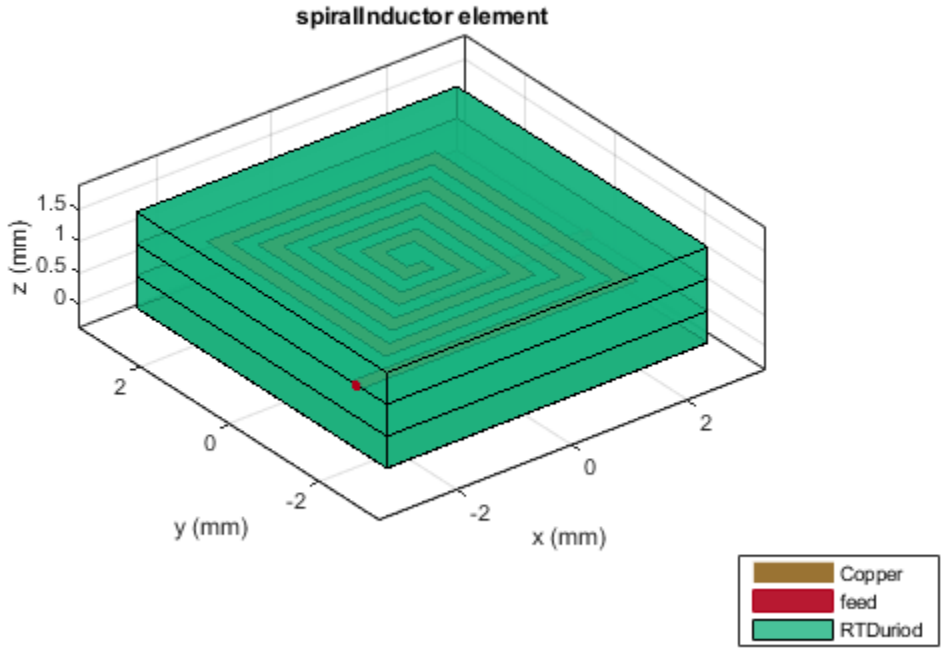
```
inductor = spiralInductor
```

```
inductor =
  spiralInductor with properties:

          SpiralShape: 'Square'
        InnerDiameter: 5.0000e-04
                Width: 2.5000e-04
              Spacing: 2.5000e-04
             NumTurns: 4
               Height: 0.0010
    GroundPlaneLength: 0.0056
     GroundPlaneWidth: 0.0056
            Substrate: [1×1 dielectric]
            Conductor: [1×1 metal]
```

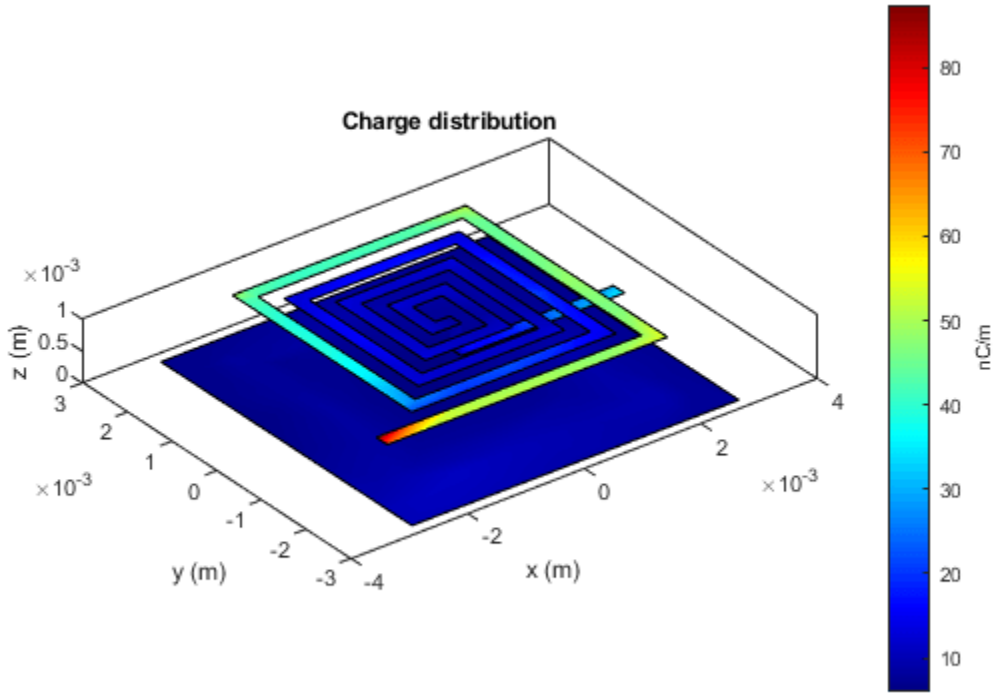
View the inductor.

```
show(inductor)
```



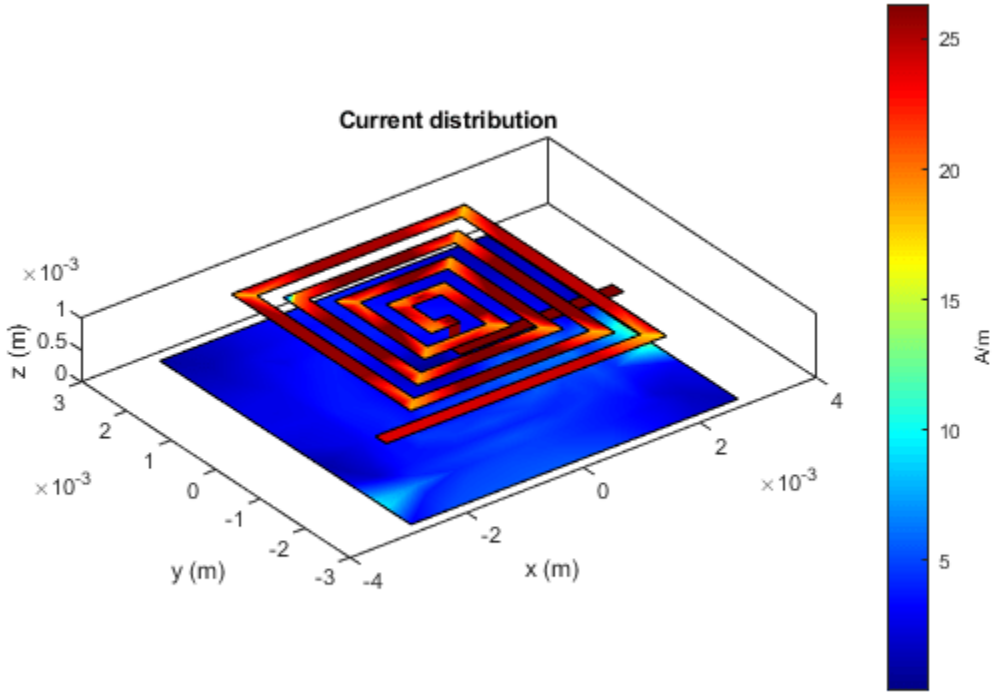
Plot the charge distribution at 500 MHz.

```
charge(inductor, 500e6)
```



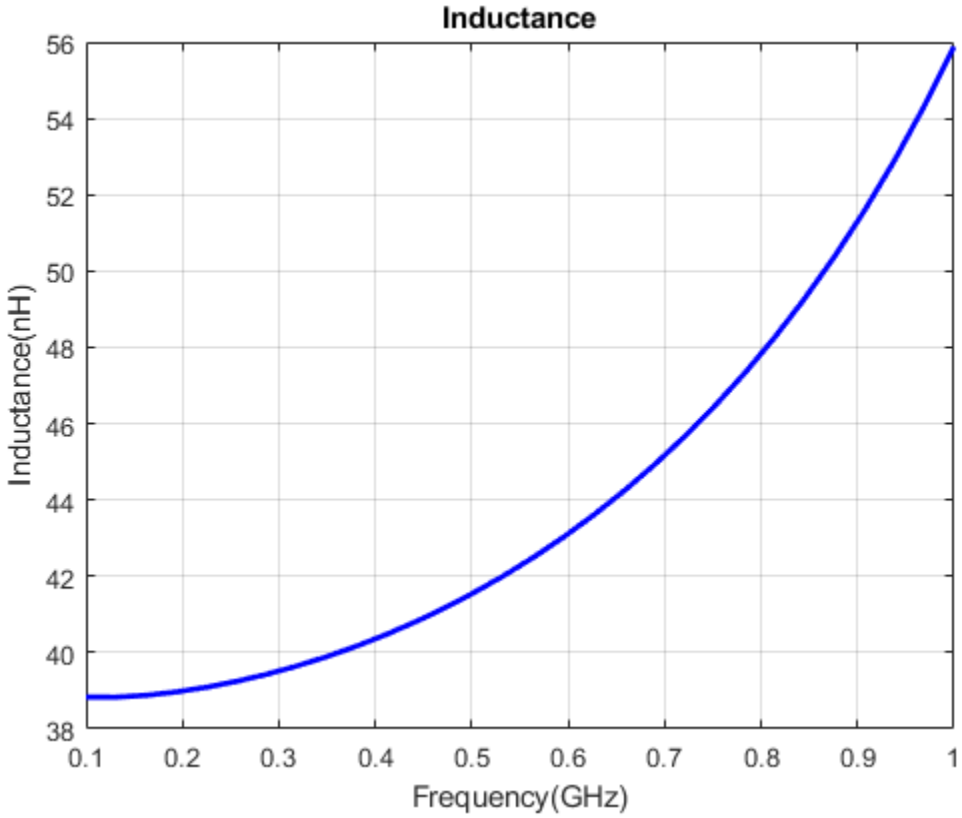
Plot the current distribution at 500 MHz.

```
figure
current(inductor, 500e6)
```



Calculate and plot the inductance.

```
figure;
inductance(inductor,linspace(100e6,1e9,30));
```


Inductance
Inductance(nH)
56
54
52
50
48
46
44
42
40
38
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Frequency(GHz)

Shunt Radial Stub - Visualize and Analyze

This example shows you how to create, visualize, and analyze a radial stub shunt on the X-Y plane.

Create a shunt radial stub with default properties.

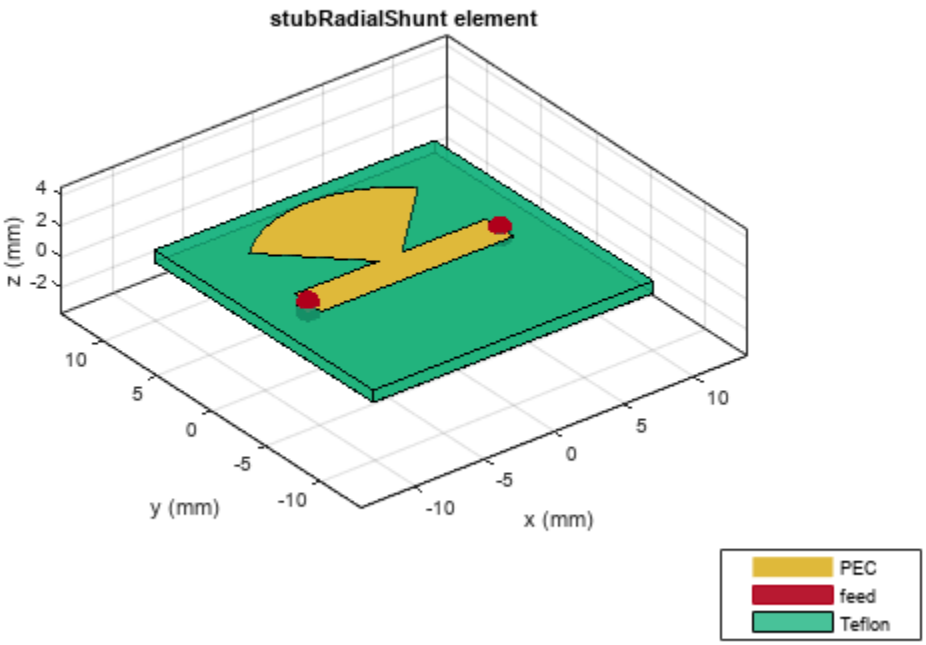
```
stub = stubRadialShunt
```

```
stub =
  stubRadialShunt with properties:

            StubType: 'Single'
         OuterRadius: 0.0085
         InnerRadius: 0.0012
               Angle: 90
      PortLineLength: 0.0137
       PortLineWidth: 0.0025
              Height: 8.0000e-04
   GroundPlaneLength: 0.0200
    GroundPlaneWidth: 0.0200
           Substrate: [1x1 dielectric]
           Conductor: [1x1 metal]
```

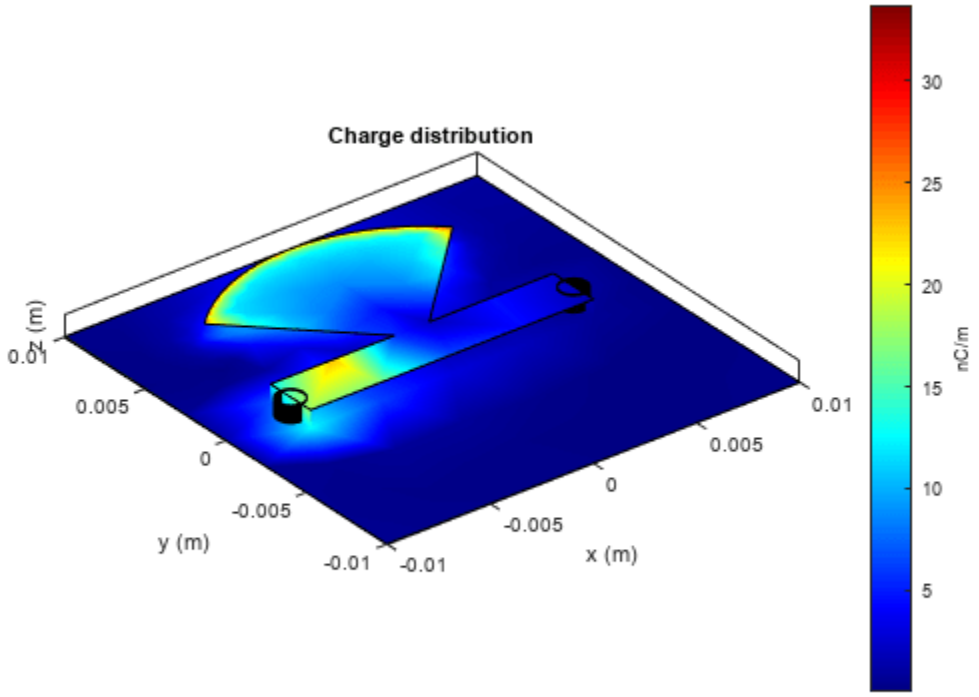
View the stub.

```
show(stub)
```



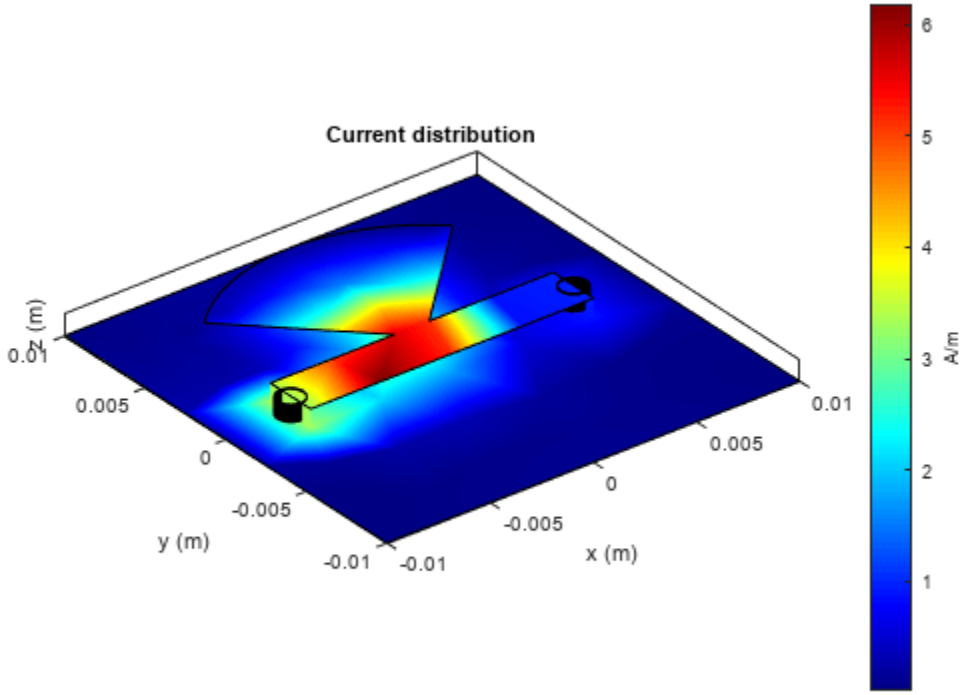
Plot the charge distribution at 5 GHz.

```
charge(stub, 5e9)
```



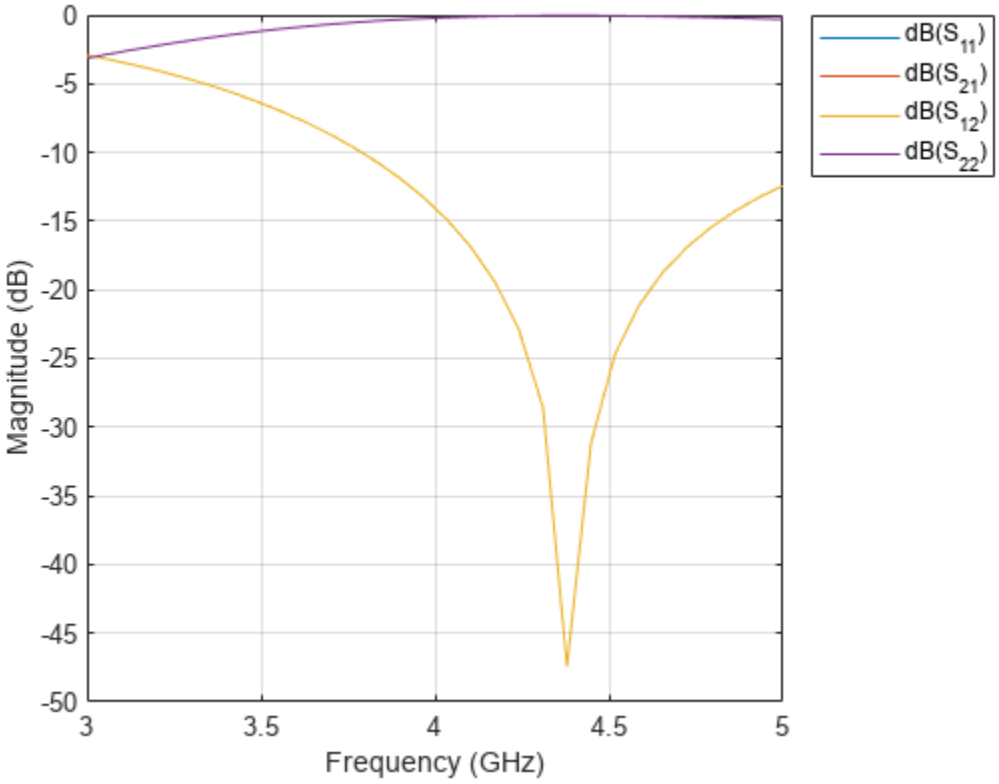
Plot the current distribution at 5 GHz.

```
figure
current(stub, 5e9)
```



Calculate and plot the s-parameters.

```
spar = sparameters(stub,linspace(3e9,5e9,30));
rfplot(spar)
```


0
-5
-10
-15
-20
-25
-30
-35
-40
-45
-50
Magnitude (dB)
3
3.5
4
4.5
5
Frequency (GHz)
dB(S11)
dB(S21)
dB(S12)
dB(S22)

RF PCB Concepts

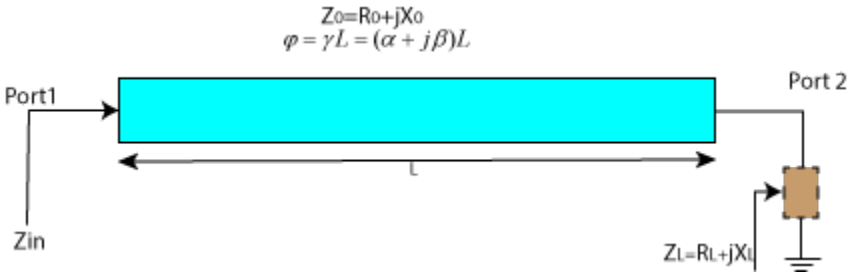
Characteristic Impedance of Transmission Lines

Characteristic Impedance

Consider an ideal, infinitely long transmission line. Exciting the input of this transmission line with the alternating voltage $V_{in}(t)$ results in the current $I_{in}(t)$. The characteristic impedance of this transmission line is given by the equation:

$$Z_0 = \frac{V_{in}(t)}{I_{in}(t)}$$

Consider a transmission line of length L terminated by load impedance of Z_L.



The complex propagation constant for this line is given by the equation:

$$\gamma = (\alpha + j\beta)$$

where α and β are the attenuation and phase constants. The complex characteristic impedance is given by the equation:

$$Z_0 = R_0 + jX_0$$

where R_0 and X_0 are the real and imaginary parts, respectively.

The input impedance of the line is given by the equation in:

Characteristic Impedances of Microstrip and Coplanar Waveguide Transmission Lines

Create microstrip and coplanar waveguide transmission lines with default properties. Use the `getZ0` from RF PCB Toolbox to calculate the characteristic impedance of following transmission lines:

- `microstripLine`

```
mline = microstripLine;
Z0 = getZ0(mline)
```

```
Z0 =

  47.6145 - 0.0004i
```

- `coplanarWaveguide`

```
waveguide = coplanarWaveguide;
Z0 = getZ0(waveguide)
```

```
Z0 =

  47.8622 + 0.0041i
```

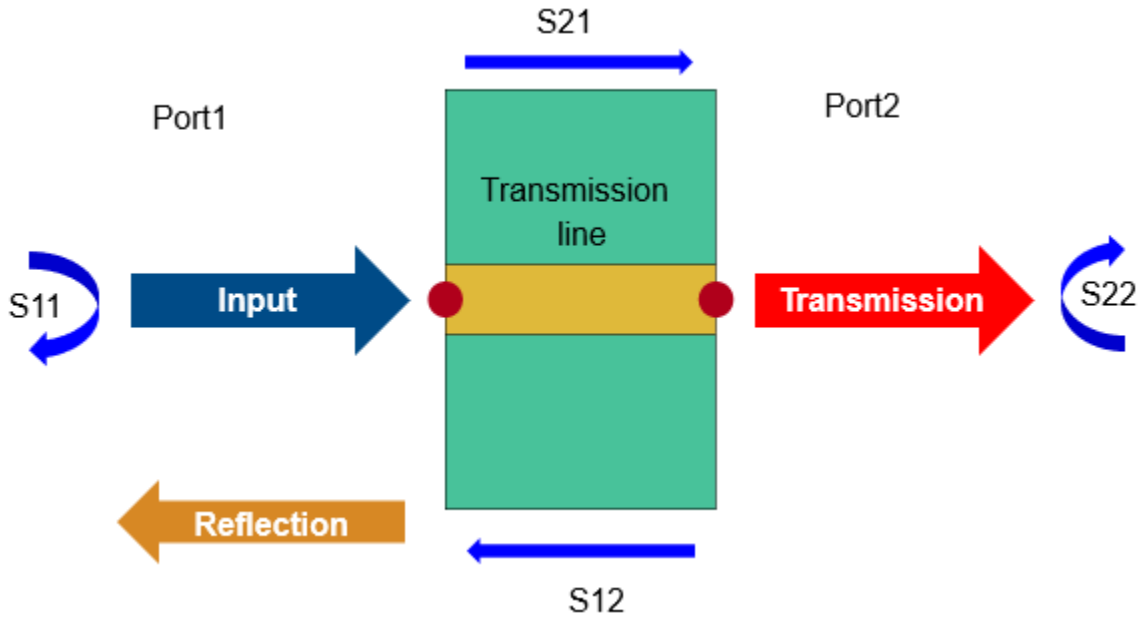
Scattering Parameters or S-Parameters

The S-parameters or scattering parameters represent the linear characteristics of radio frequency (RF) printed circuit board (PCB) components. The parameter matrix describes the changes in the reflected and transmitted waves with respect to the incident wave at an N-port PCB component. Using S-parameters you can calculate the gain, loss, phase delay, voltage standing wave ratio (VSWR), and other characteristics of any PCB component or linear network. S-parameters can also be called the behavioral representation of a network.

Basic Concepts

S-parameters are defined in terms of transmission and reflection coefficients. Consider a two-port transmission line. For a two-port device, there are four S-parameters:

- Reflected/Input = S_{11} and S_{22}
- Transmitted/Input = S_{12} and S_{21}



S_{21} is the measure of the transmitted signal from port 2 relative to the input entering port 1. S_{11} is the measure of the reflected signal from port 1 relative to the input entering port 1.

For a multiport network, the transmission and reflection coefficients are defined as S_{nn}, S_{mm}, S_{nm}, and S_{mn}, where:

- *m* is the number of input ports.
- *n* is the number of output ports.

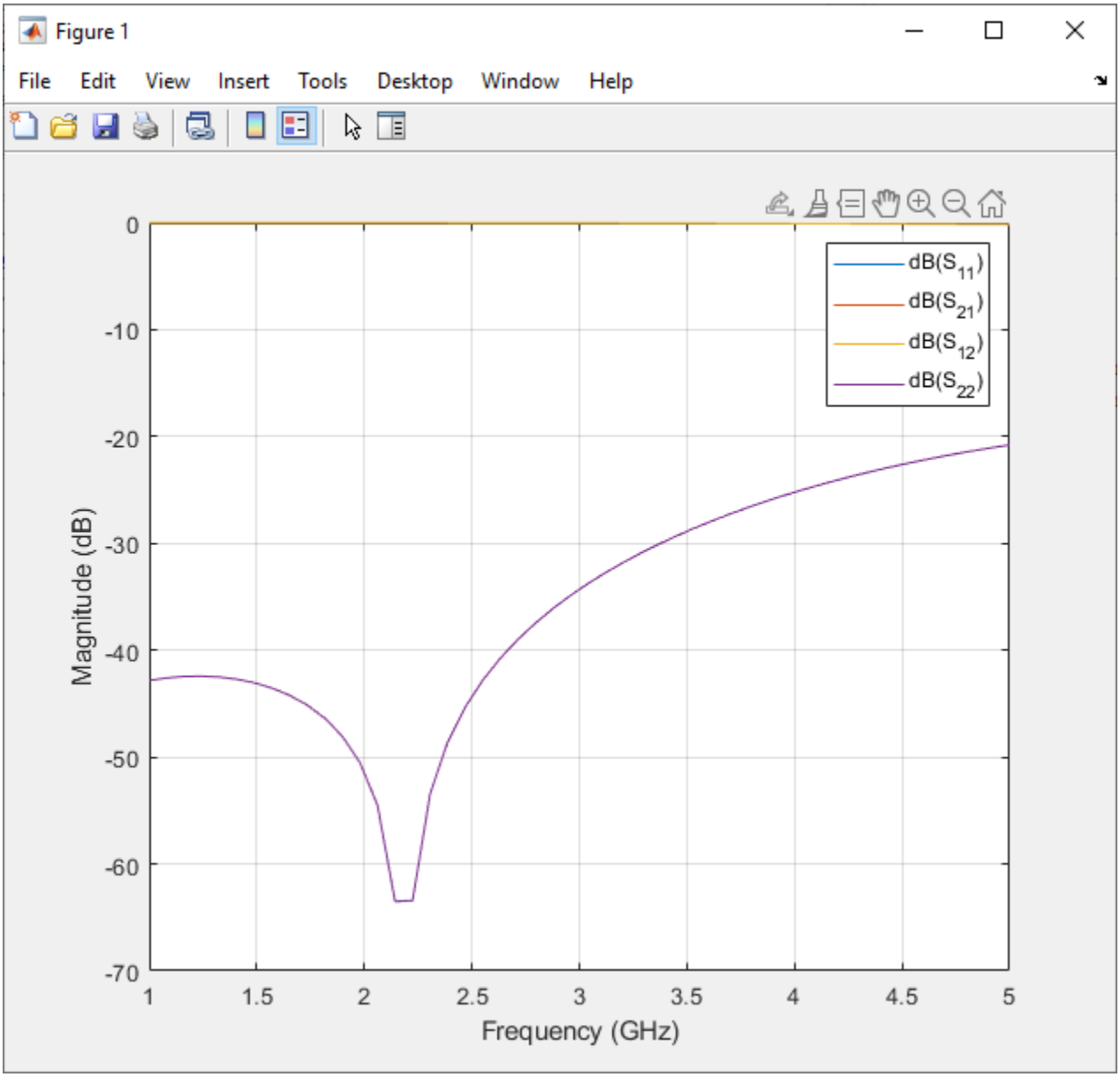
Understanding S-Parameter Plots

S-Parameters of Microstrip Transmission Lines

Consider a microstrip transmission line from “Transmission Lines”. The parameter values for this microstrip transmission line are for a frequency range of 1-5 GHz. Calculate and plot the S-

parameters at this frequency range. The operating frequency for the microstrip transmission line is around 2.5 GHz.

```
mline = microstripLine;
spar = sparameters(mline,linspace(1e9,5e9,50));
rfplot(spar)
```

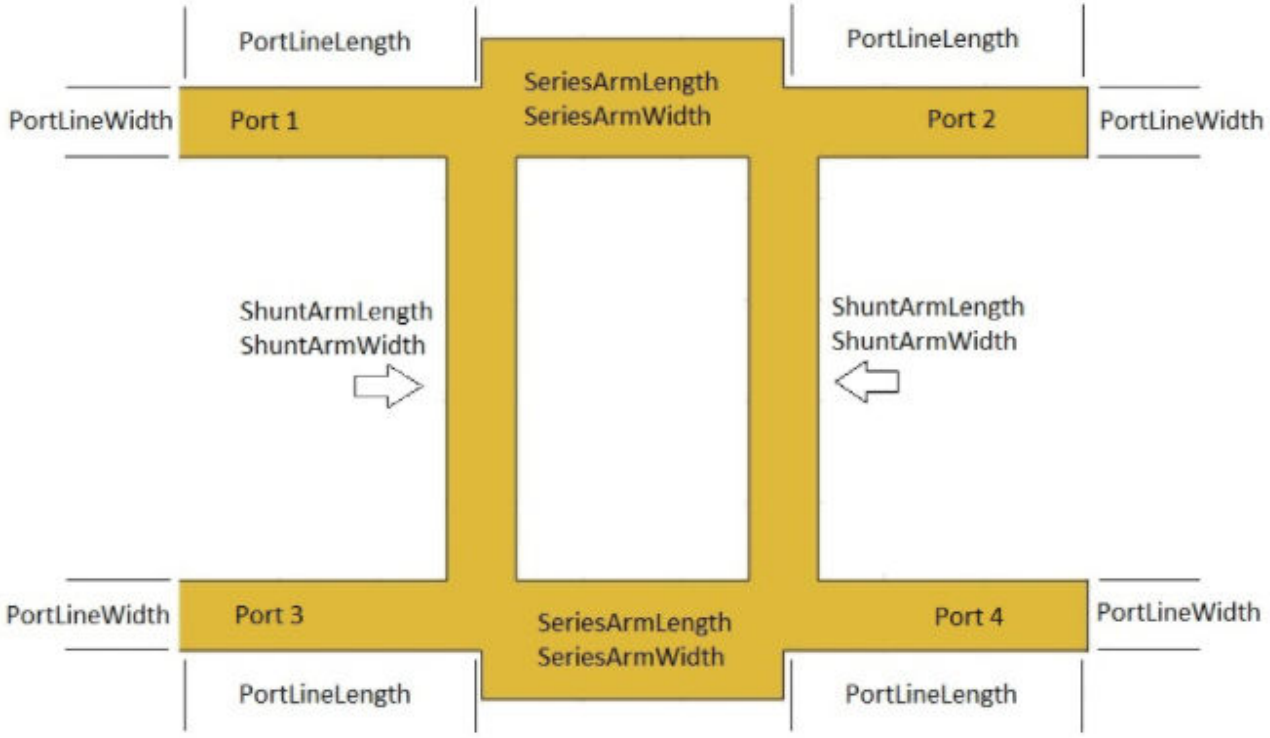


Let us look at the S_{11} and S_{21} of the plot.

- The S_{11} plot shows the return loss as almost negligible at `-50` dB at around 2.3 GHz. As the frequency increases, more signal is reflected back to the input port.
- The S_{21} plot describes the end-to-end transmission of the signal. The plot is flat throughout at 0 dB showing very little loss at all frequencies.

S-Parameters of Branchline Coupler

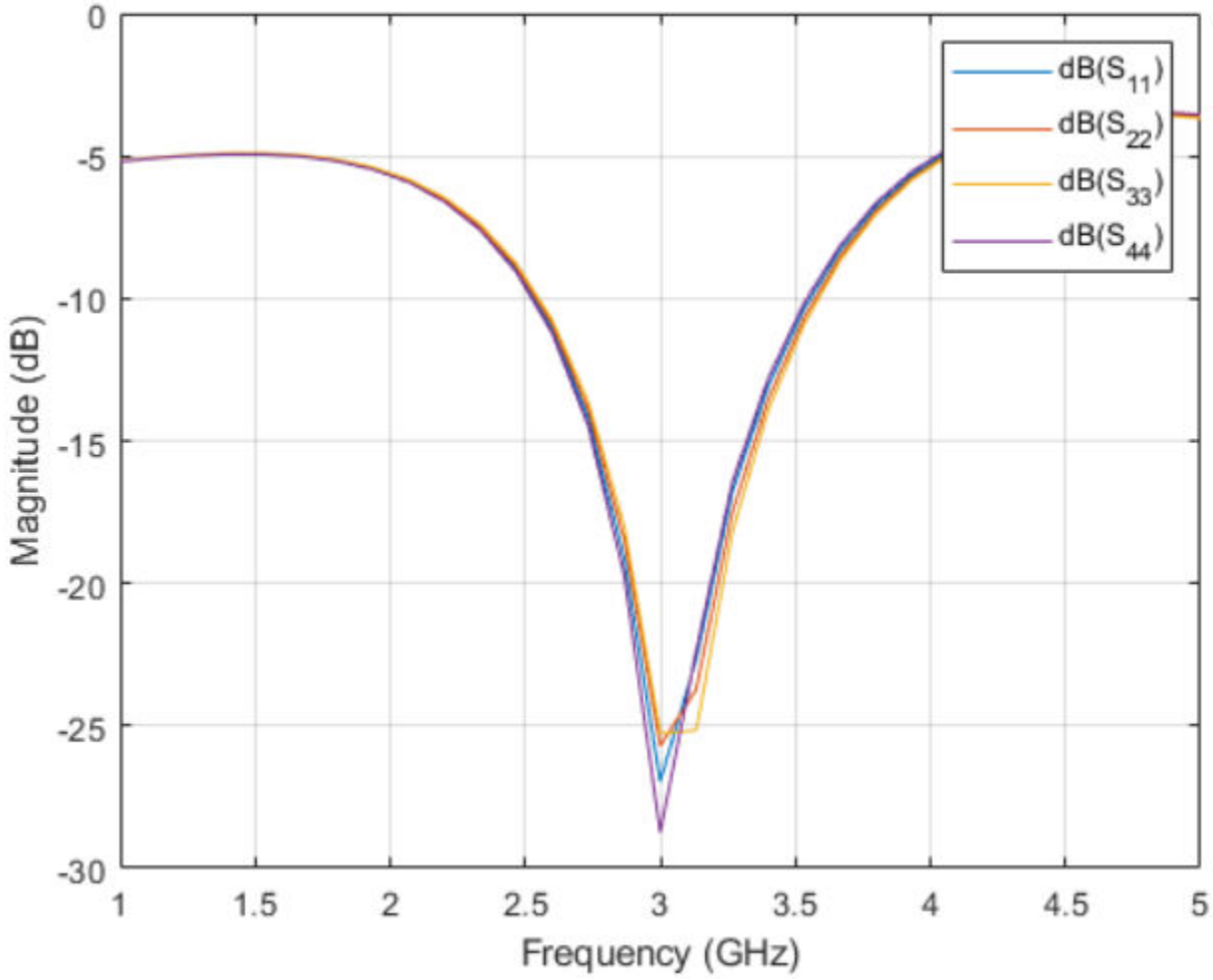
A branchline coupler is a four-Port coupler that has an input port, through port, coupled port, and an isolated port. Port 1 is the input port, port 2 is the through port, port 3 is the isolated port and port 4 is the coupled port. When the couplers are equally split, power is evenly divided between the through port and the coupled port.



Consider the branchline coupler from “Splitters and Couplers”. The parameter values of this coupler are for a design frequency of 3 GHz. Calculate and plot the S-parameters of this coupler to understand its behavior.

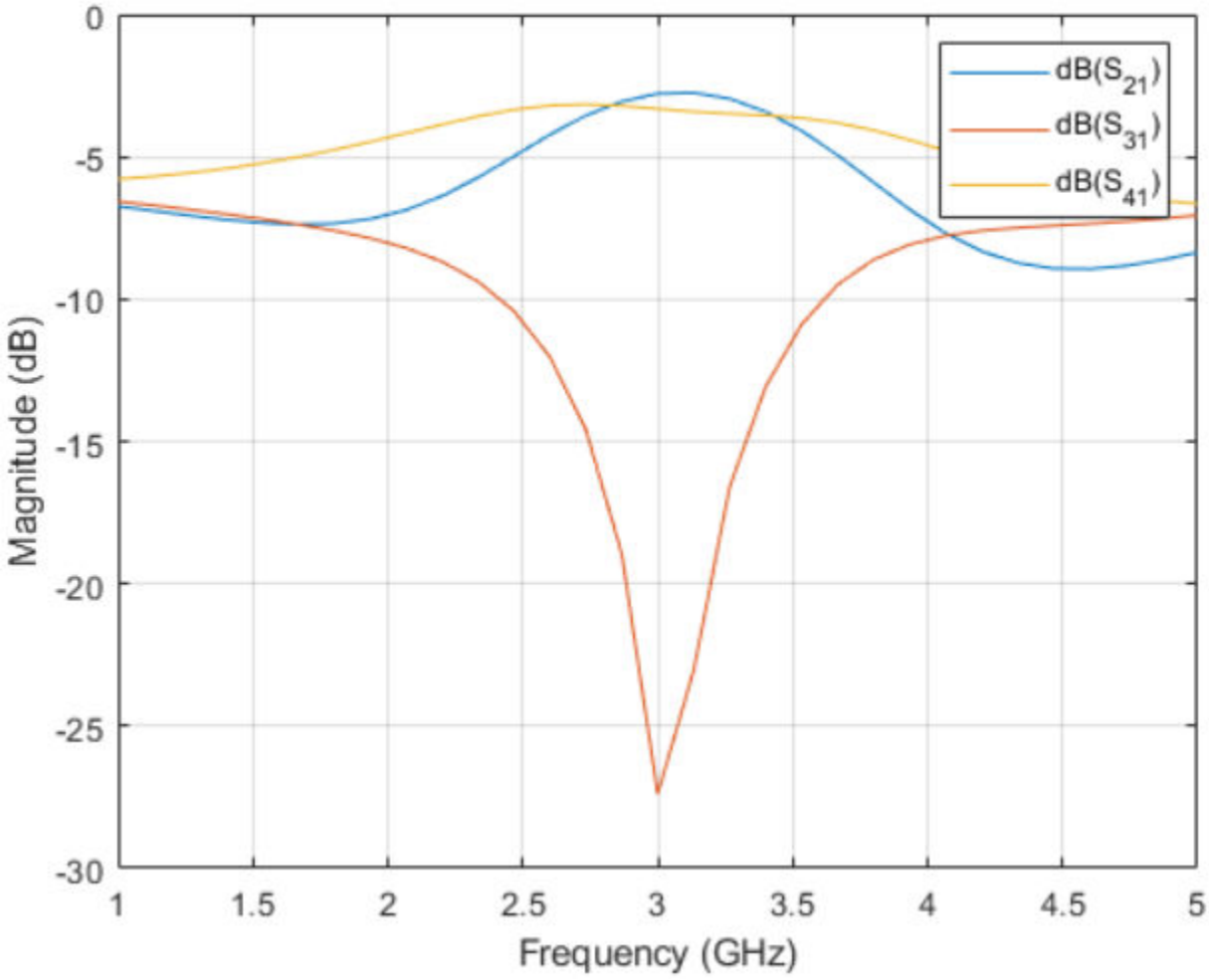
```
coupler = couplerBranchline;
spar = sparameters(coupler,linspace(1e9,5e9,50));
```

-
```
rfplot(spar,1,1)
hold on
rfplot(spar,2,2)
hold on
rfplot(spar,3,3)
hold on
rfplot(spar,4,4)
```



All the reflection coefficients S_{11}, S_{22}, S_{33}, and S_{44} are close to `-30` dB showing that the default branchline coupler has a good match at all ports.

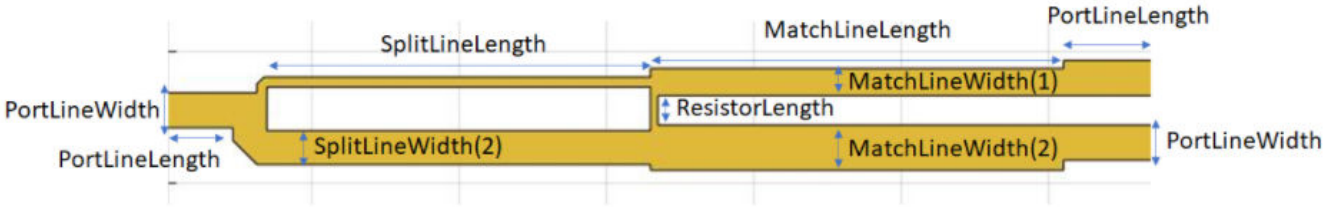
- ```
  rfplot(spar,2,1)
  hold on
  rfplot(spar,3,1)
  hold on
  rfplot(spar,4,1)
  ```
  ```



The transmission coefficients of S_{21} and S_{41} show that the power is equally divided between Port 2 and Port 4. S_{31} is `-30` dB as Port 3 is isolated.

S-Parameters of Unequal Wilkinson Splitter

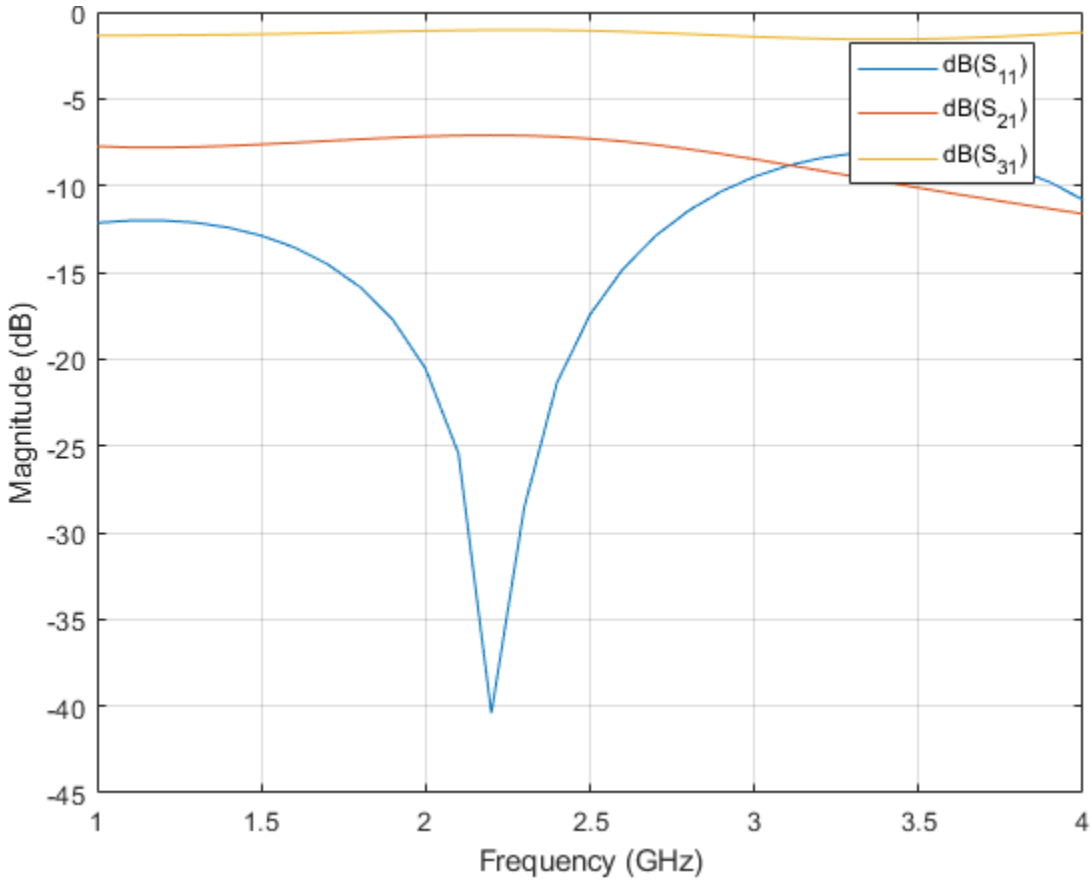
An unequal Wilkinson splitter performs arbitrary power division between three ports based on the power ratio in the component design.



Consider an unequal Wilkinson splitter from “Splitters and Couplers”. Design it for a frequency of 2.5 GHz and with a power ratio of 4.

```
splitter = wilkinsonSplitterUnequal;
splitter = design(splitter,2.5e9,'PowerRatio',2);
```

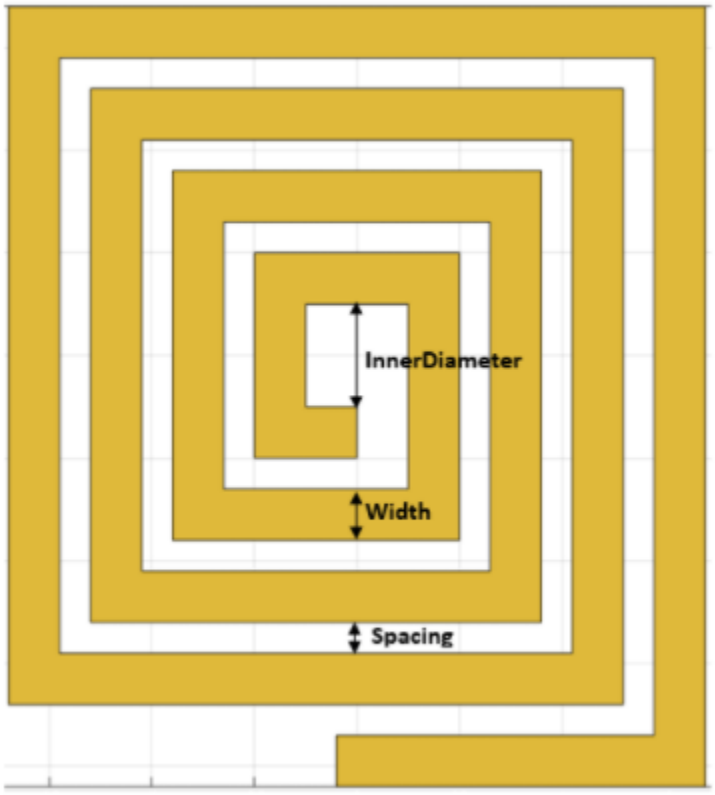
```
spar = sparameters(coupler,linspace(1e9,4e9,31));
rfplot(spar,1,1)
hold on
rfplot(spar,2,1)
hold on
rfplot(spar,3,1)
```



- The S_{11} reflection coefficient shows the value of `-30` dB at 2.5 GHz, showing very little reflection at the design frequency.
- The S_{31} value is close to `-0.97` dB and S_{21} value is `-6.99` dB indicating unequal power division at the output ports based on the power ratio of 4.

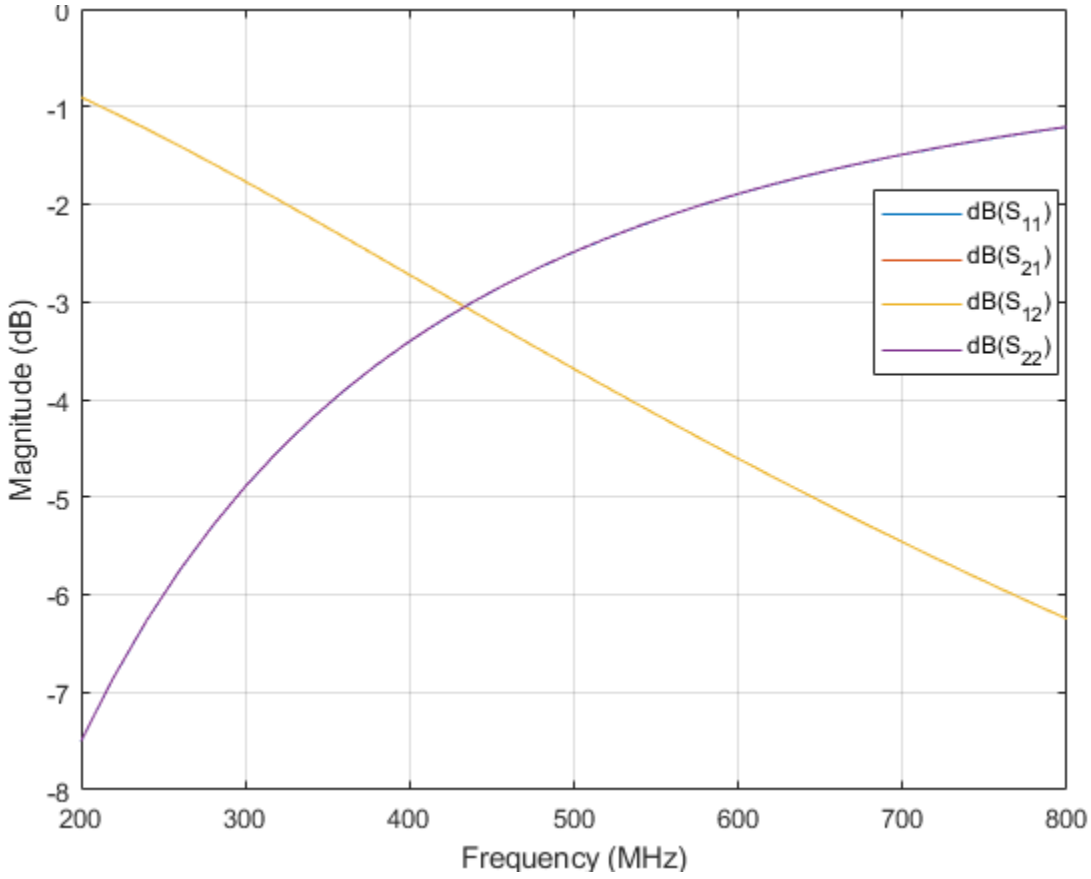
S-Parameters of Spiral Inductor

A spiral inductor is used as a microwave-resonant element in PCB circuits and as a choke in power supplies.



Consider a square spiral inductor from “Inductors and Capacitors”. Calculate and plot the S-parameters to show the power conservation at the default frequency of 600 MHz.

```
inductor = spiralInductor;
spar = sparameters(inductor,linspace(200e6,800e6,31));
rfplot(spar);
```



- The S_{11} and S_{22} values increase to 0 dB after 600 MHz..
- The S_{12} and S_{21} values decrease indicating that the energy is stored in the inductor and not radiated.
- The behavior of the S_{11} and S_{21} values satisfies the law of conservation of energy.

See Also

`sparameters`

More About

- “Behavioral Models” on page 3-12

Behavioral Models

Behavioral modeling or black-box modeling does not require you to provide the specifications of all the physical systems in your model. The type of modeling depends on the input and output measurements of the component or systems.

In RF PCB Toolbox you can:

- Perform the behavioral analysis on a PCB component by using the `sparameters` function with the `Behavioral` property set to true.
- Convert a PCB component into a behavioral PCB element using the `pcbElement` object.

The components and shapes that support behavioral modeling are:

Bends	• `bendRightAngle` • `bendCurved` • `bendMitered` **Note** The `sparameters` function does not support the behavioral model for objects with unequal widths such as `bendRightAngle`, `bendCurved`, and `bendMitered` objects.
Traces	• `traceTee` • `traceCross` **Note** The `sparameters` function does not support the behavioral model for Asymmetric tee- and cross-junction traces.
Transmission line objects	• `microstripLine` • `coplanarWaveguide`
Inductor	`spiralInductor`
Capacitor	`interdigitalCapacitor`
Splitter and Couplers	• `couplerRatrace` • `couplerBranchline` • `wilkinsonSplitter` • `wilkinsonSplitterUnequal` • `splitterTee`
Vias	• `viaSingleEnded`

Perform Behavioral Analysis of PCB Component

The main advantage of behavioral modeling as compared to full-wave electromagnetic (EM) modeling is that you can analyze the PCB component faster using behavioral modeling.

Consider an interdigital capacitor with default properties. Calculate the S-parameters of this component.

```
capacitor = interdigitalCapacitor;
tic
sparameters(capacitor,4e9)
toc

ans =

  sparameters: S-parameters object

       NumPorts: 2
    Frequencies: 4.0000e+09
     Parameters: [2×2 double]
      Impedance: 50

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

Elapsed time is 176.844515 seconds.
```

The S-parameters function takes `176.9` seconds to run.

Now calculate the S-parameters of this capacitor with the `Behavioral` property in `sparameters` function set to true.

```
capacitor = interdigitalCapacitor;
tic
sparameters(capacitor,4e9,Behavioral=true)
toc

ans =

  sparameters: S-parameters object

       NumPorts: 2
    Frequencies: 4.0000e+09
     Parameters: [2×2 double]
      Impedance: 50

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

Elapsed time is 0.295000 seconds.
```

The S-parameters function takes `0.30` seconds.

Convert PCB Component to Behavioral Circuit Element

You can convert a PCB component into a behavioral PCB element by using the `pcbElement` object, and then add this PCB element to an RF Toolbox™ circuit object. You can convert only PCB components that support behavioral analysis to behavioral PCB elements. For a complete list of objects and shapes that support behavioral analysis, see the introduction.

Create a RF Toolbox circuit object of two interdigital capacitors. Use the behavioral model in the first capacitor and the full-wave model in the second capacitor.

```
ckt = circuit;
c1 = interdigitalCapacitor;
c2 = interdigitalCapacitor('NumFingers',3);
p = pcbElement(c2,'Behavioral',false);
```

```
add(ckt,[1 2 0 0],c1) % default pcbElement created automatically
add(ckt,[2 3 0 0],p)
setports(ckt,[1 0],[3 0])
tic
S = sparameters(ckt,8e9)
toc

S =

  sparameters: S-parameters object

       NumPorts: 2
    Frequencies: 8.0000e+09
     Parameters: [2×2 double]
      Impedance: 50

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

Elapsed time is 1.997850 seconds.
```

The S-parameters function takes `1.9` seconds.

Now calculate the S-parameters of the circuit with both capacitors as full-wave model circuit elements.

```
ckt = circuit;
c1 = interdigitalCapacitor;
c2 = interdigitalCapacitor('NumFingers',3);
p1 = pcbElement(c1,'Behavioral',false);
p2 = pcbElement(c2,'Behavioral',false);
add(ckt,[1 2 0 0],p1) % default pcbElement created automatically
add(ckt,[2 3 0 0],p2)
setports(ckt,[1 0],[3 0])
tic
S = sparameters(ckt,8e9)
toc

S =

  sparameters: S-parameters object

       NumPorts: 2
    Frequencies: 8.0000e+09
     Parameters: [2×2 double]
      Impedance: 50

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

Elapsed time is 3.482307 seconds.
```

The S-parameters function takes `3.5` seconds.

Comparing the two circuits shows a small difference in the time taken to calculate the S-parameters. Now consider an entire PCB board with many PCB components. You can perform behavioral modeling on almost all noncritical components and full-wave modeling on the critical components of a PCB board.

See Also

`sparameters`

More About

- “Scattering Parameters or S-Parameters” on page 3-4

Board Thickness versus Dielectric Thickness in PCB

This example shows you how to define the BoardThickness of PCB with respect to dielectric thickness of `pcbComponent` object for different use-cases.

The BoardThickness is the property of `pcbComponent` object and its value 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 not included in the BoardThickness calculations.

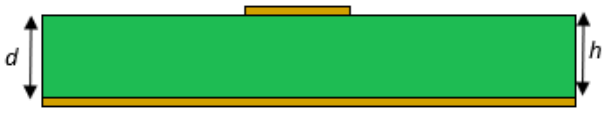
N**ote: Define BoardThickness before defining the Layers.**

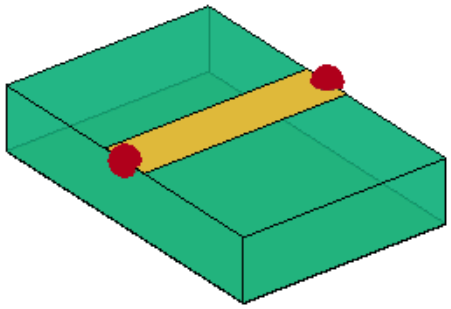
The three different sections defines the board thickness for different use-cases:

1. Section one defines the BoardThickness for a single-layered dielectric PCB.
2. Section two defines the BoardThickness for a multi-layered dielectric PCB and
3. Section three defines the BoardThickness when importing the PCB using a `pcbReader` object.

Single-Layered dielectric pcbComponent

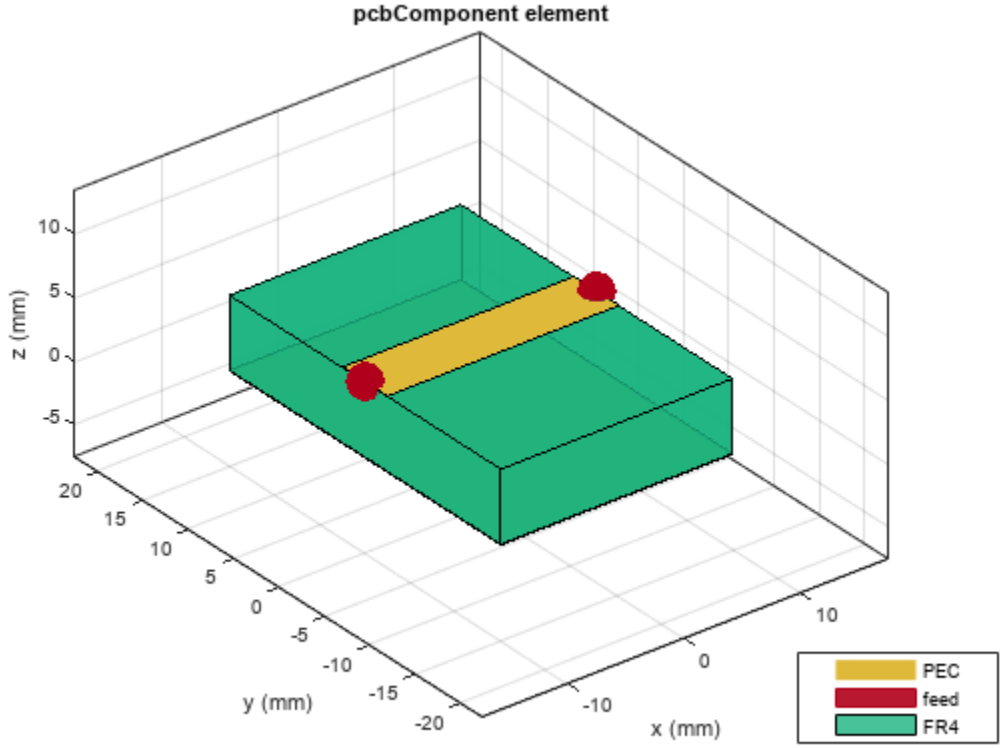
In this use case, the PCB a single dielectric layer sandwiched between the top metal layer and bottom metal layer. Here, *h* defines the BoardThickness and the *d* is the dielectric thickness.





When the pcbComponent has a single dielectric layer, the BoardThickness (*h*) should be equal to dielectric thickness (*d*). If *h* is not equal to *d*, then *d* is updated to match *h*.

```
p = pcbComponent;
sub = dielectric('Fr4');
TopMetal = p.Layers{1};
BtmMetal = p.Layers{3};
p.BoardThickness = sub.Thickness;
p.Layers = {TopMetal,sub,BtmMetal};
figure; show(p);
```

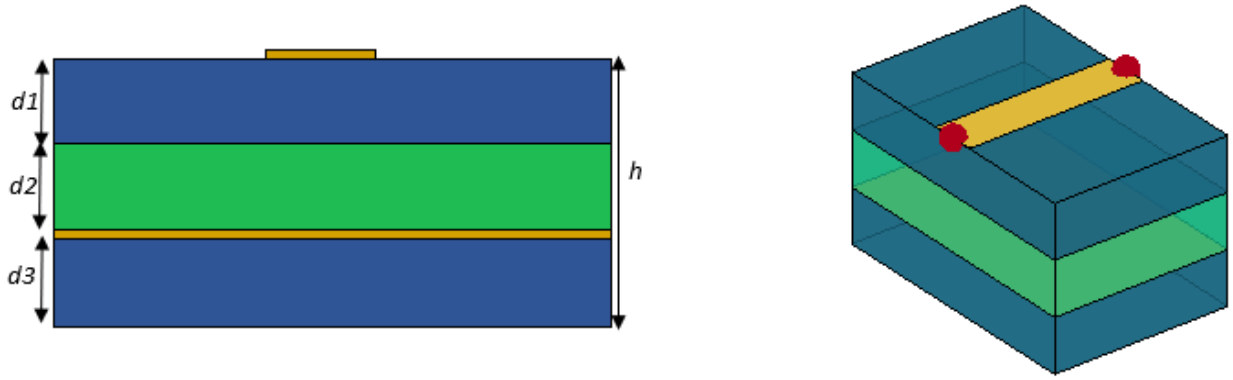


Multi-Layered dielectric pcbComponent

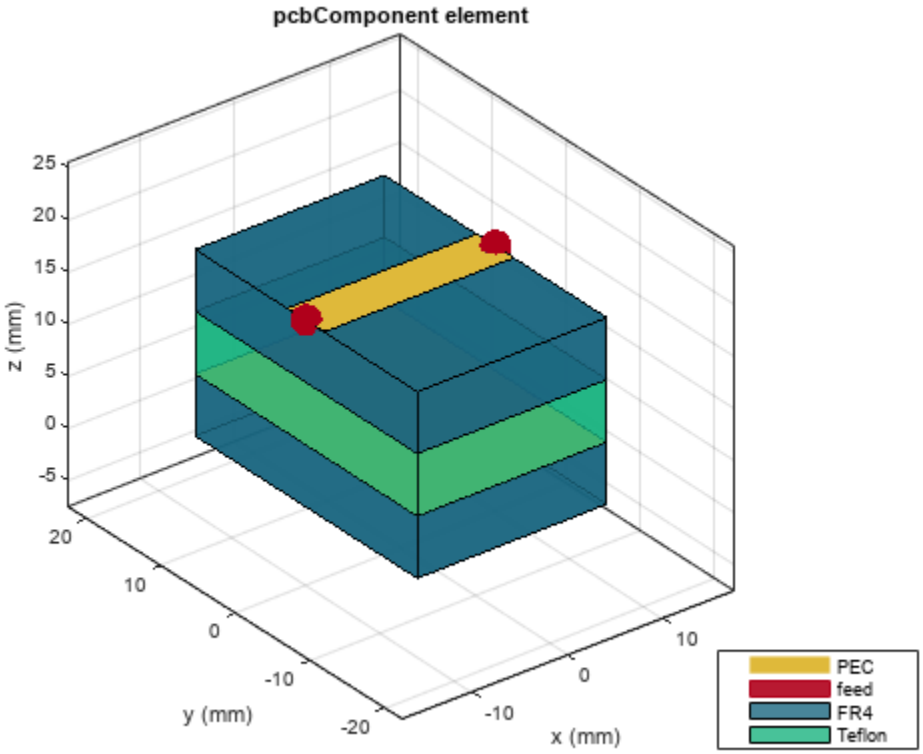
The dielectric layers are present below the top metal layer.

For multi-layered dielectrics in a PCB, the BoardThickness will be the sum of thickness of the dielectric layers below the top metal layer. The dielectrics above the top metal layers are considered as a coating and will not be included in the board thickness calculations.

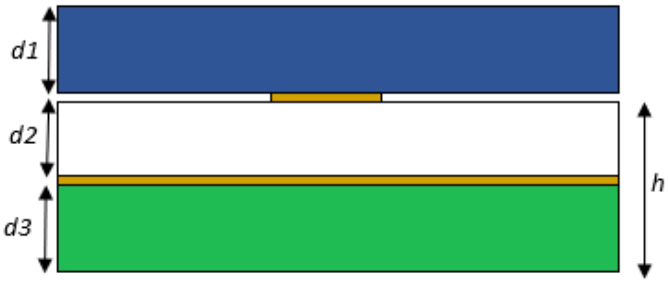
In this use case, there are two dielectric layers below the top metal layer and one dielectric layer below the bottom metal layer as shown in the figure below. The BoardThickness (h) will be the sum of dielectric layers below the top metal layer (i.e. $h = d1+d2+d3$).

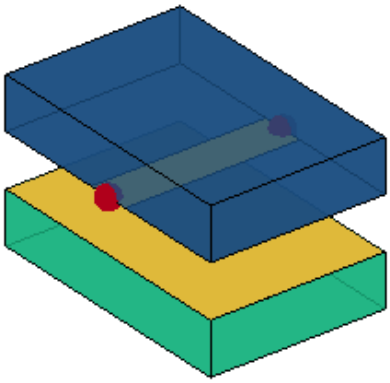


```
p = pcbComponent;
sub1 = dielectric('FR4');
sub2 = dielectric('Teflon');
TopMetal = p.Layers{1,1};
BtmMetal = p.Layers{1,3};
% Set the BoardThickness as sum of dielectric thickness below the top metal
% layer
h = 2*sub1.Thickness+sub2.Thickness;
p.BoardThickness = h;
p.Layers = {TopMetal,sub1,sub2,BtmMetal,sub1};
p.FeedLocations(:,3:4) = [1,4;1,4];
figure; show(p)
```

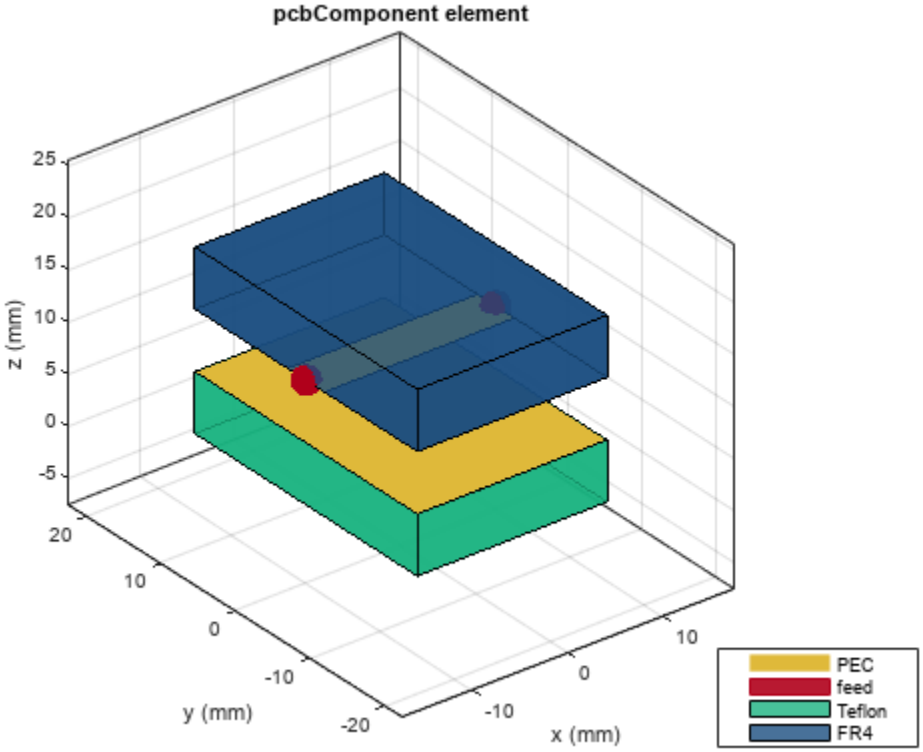


In this use case, there is a dielectric layer present above the top metal layer and air dielectric separates the top and bottom metal layers. Add another dielectric layer below the bottom metal layer. The dielectric layer which is present above the top metal layer is considered as coating and is included in the BoardThickness calculation. The BoardThickness *(h)* will be the sum of dielectric layers below the top metal layer (*ie. h= d2+d3).*





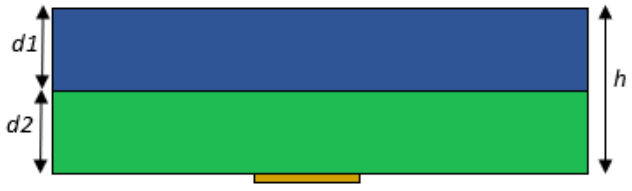
```
p = pcbComponent;
sub1 = dielectric('FR4');
sub2 = dielectric('Air');
sub3 = dielectric('Teflon');
TopMetal = p.Layers{1,1};
BtmMetal = p.Layers{1,3};
% Set the BoardThickness as sum of dielectric thickness below the top metal
% layer
h = sub2.Thickness+sub3.Thickness;
p.BoardThickness = h;
p.Layers = {sub1,TopMetal,sub2,BtmMetal,sub3};
p.FeedLocations(:,3:4) = [2,4;2,4];
figure; show(p)
```

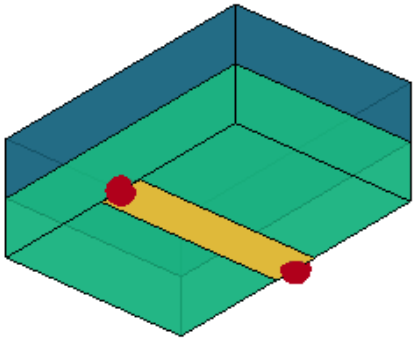


The dielectric layers are present only above the top metal layer

In this use case, PCB has multilayer dielectrics that are above the top metal layer. The BoardThickness(h) will be the sum of thickness of all the dielectric layers *(h = d1+d2)*.

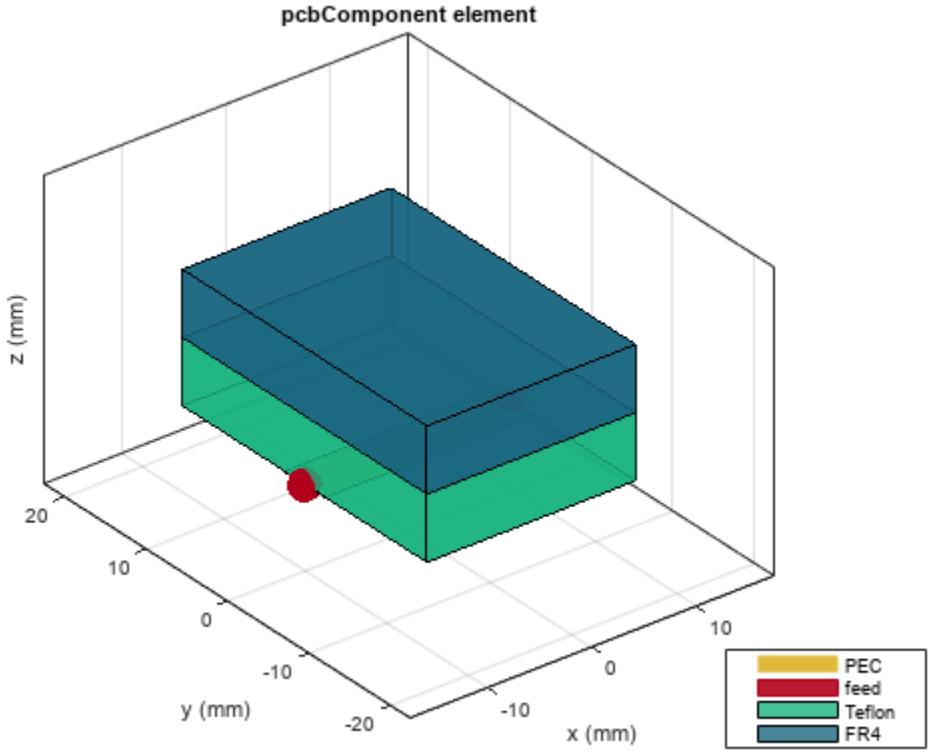
Note: This is applicable only when there are no dielectrics present below the top metal layer.





```
p = pcbComponent;
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(:,3:4) = 3;
figure; show(p);
```



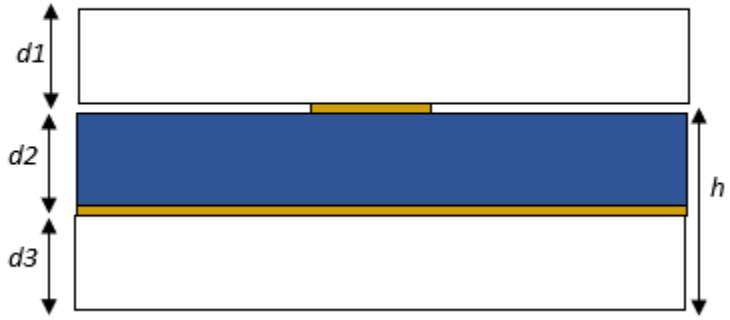
Create pcbComponent for pcbReader workflow

Import the top and bottom layers 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 = 'EBGstruct.gtl';
S.Layer4 = 'EBGstruct.gbl';
p = PCBReader ('StackUp',S);
```

Create pcbComponent of PCBReader object.

When the PCBReader object is converted as pcbComponent , the pcbComponent will read all the 5 layers of stackUp object, where air dielectric is the default top and bottom layer. This is a multilayered dielectric use case. The BoardThickness will be the sum of thickness of dielectric below the top metal layer *(ie. h = d2+d3).*

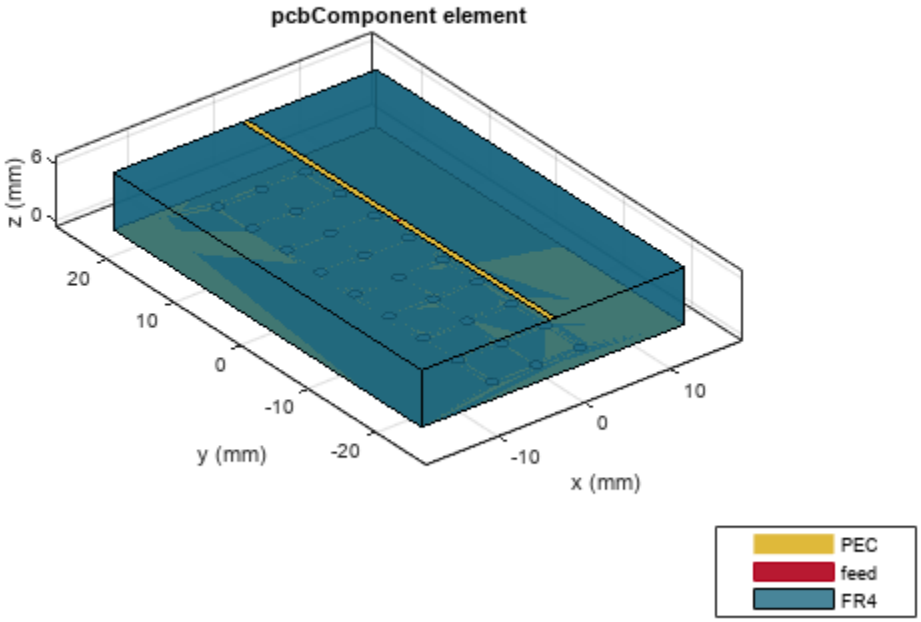



```
pcb = pcbComponent(p);
pcb.BoardThickness = pcb.Layers{3}.Thickness+pcb.Layers{5}.Thickness;
pcb.FeedLocations = [0,0,2]
```

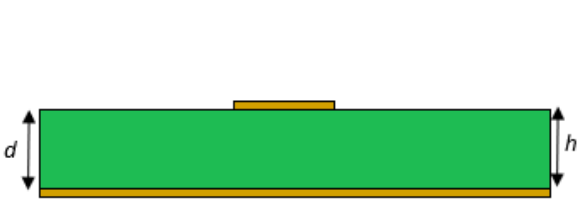
```
pcb =
  pcbComponent with properties:

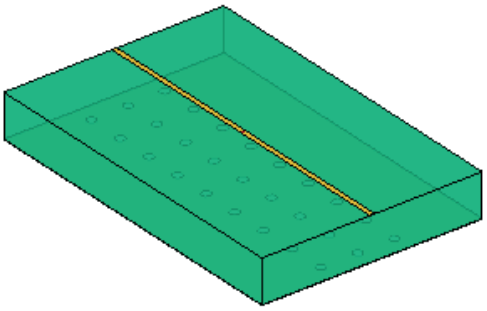
             Name: 'EBGstruct'
         Revision: 'v1.0'
       BoardShape: [1x1 antenna.Rectangle]
    BoardThickness: 0.0061
           Layers: {[1x1 dielectric]  [1x1 antenna.Polygon]  [1x1 dielectric]  [1x1 antenna.Poly
     FeedLocations: [0 0 2]
      FeedDiameter: 0.0100
      ViaLocations: []
       ViaDiameter: []
      FeedViaModel: 'square'
         Conductor: [1x1 metal]
              Tilt: 0
          TiltAxis: [0 0 1]
              Load: [1x1 lumpedElement]
```

```
pcb.FeedDiameter = 0.1e-3;
show(pcb);
```

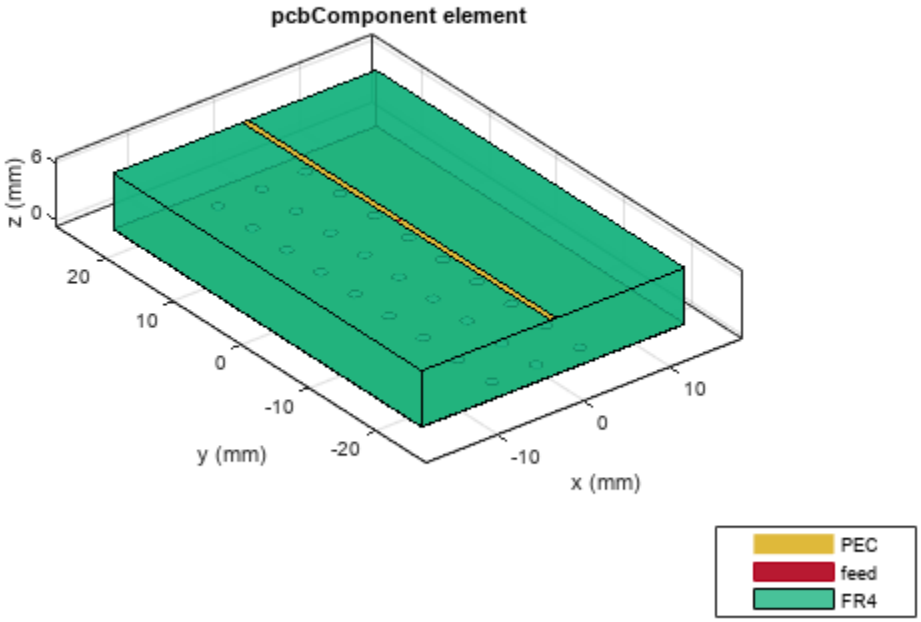


Modify the above use case by removing the top and bottom air dielectric layers as show in figure. This use case is now a single-layer dielectric pcbComponent. Now you can change the dielectric thickness by changing the BoardThickess.(h = d).





```
pcb = pcbComponent(p);
d = pcb.Layers{3};
pcb.BoardThickness = d.Thickness;
L = {pcb.Layers{2},pcb.Layers{3},pcb.Layers{4}};
pcb.Layers = L;
pcb.FeedLocations = [0,0,1];
pcb.FeedDiameter = 0.1e-3;
show(pcb);
```


pcbComponent element
z (mm)
6
0
20
10
0
-10
-20
y (mm)
-10
0
10
x (mm)
PEC
feed
FR4