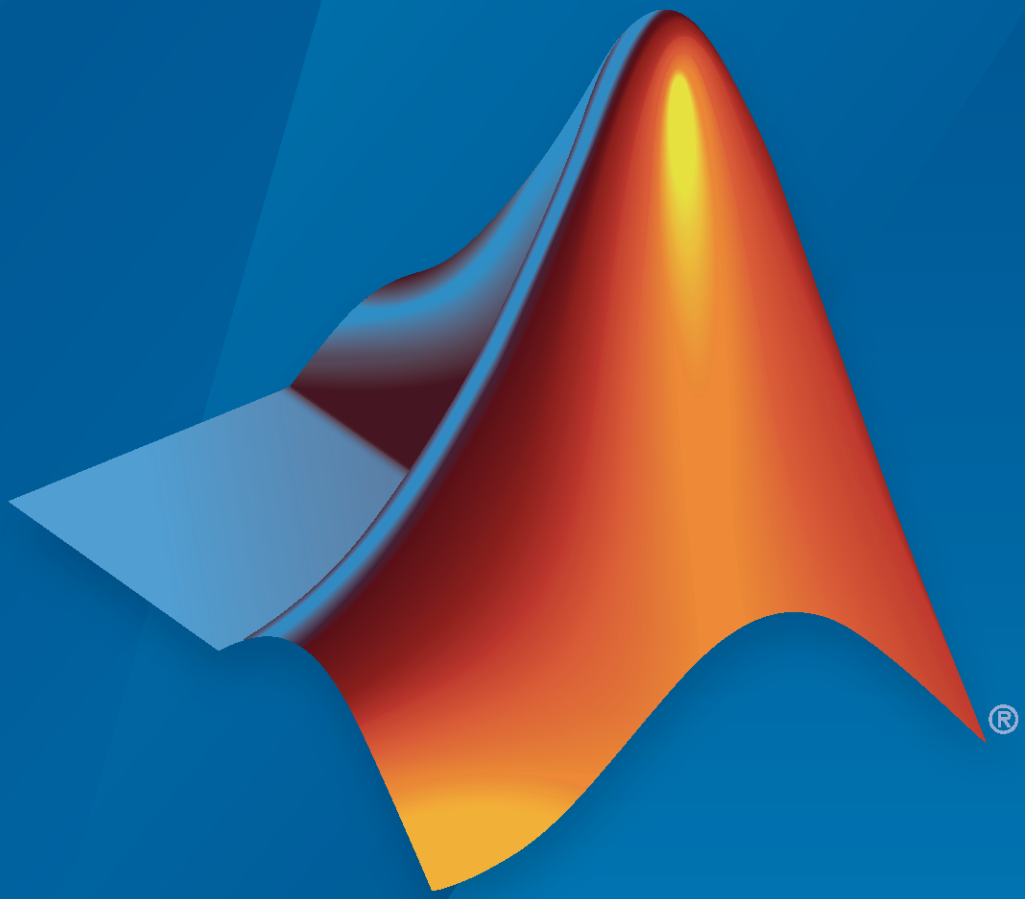
RF PCB Toolbox™

User's Guide



MATLAB®

R2022b

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™ User's Guide

© COPYRIGHT 2021–2022 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

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)

Contents

1 RF PCB Examples

Stepped Impedance Maximally Flat Lowpass Filter for Microwave Applications **1-2**

On-Chip Square Spiral Inductor for Si RFIC Application **1-8**

Bandstop and Bandpass Filters with Open Microstrip Line Stubs Using Behavioral and EM Simulation **1-14**

Dual-Fed Square Microstrip Patch Antenna for BLE Applications **1-22**

Create and Analyse Dual Characteristics of Complementary Split Ring Resonators from Gerber Files **1-37**

Prototype, Design, and Analysis of SIW based Microstrip Tapered Transmission Line **1-44**

Model and Analyze Microstrip Interdigital Capacitor as Bandpass filter **1-50**

Introduction to Passive Planar Spiral Inductors **1-55**

Design and Analysis of Hairpin Micro-Strip Line Bandpass Filter **1-66**

Analyzing Crosstalk Between PCB Traces **1-77**

Create and Analyze PCB Interconnects using Custom Traces **1-88**

Routing Discipline to Reduce Crosstalk **1-99**

Introduction to Equal and Unequal Split Wilkinson Power Splitter . . . **1-106**

Introduction to 4-Port Couplers **1-117**

Design Two-Stage Low Noise Amplifier Using Microstrip Transmission Line Matching Network **1-137**

Design and Analyze Compact UWB Low Pass Filter Using pcbComponent **1-152**

Design and Analyze Band Stop Filter using pcbComponent **1-165**

Design and Analyze HighPass Filter Using pcbComponent **1-177**

Design of Quarter-Wave Transformer for Impedance Matching Applications **1-190**

Corporate Feed Divider Network for a Linear Patch Antenna array . . . **1-197**

Microstrip Composite Bandpass Filters for Ultra-Wideband (UWB) Wireless Communications **1-204**

UWB Bandpass Filter Using Open and Short-Circuited Stubs **1-224**

Comparison of Lumped and Distributed EM Models for Low Pass Filters **1-231**

Variations of Open and Short Stub Filters **1-237**

Microstrip Ultra-Wideband Bandpass Filter with Cascaded Broadband Bandpass and Bandstop Filters **1-244**

2 Computational Techniques

Method of Moments Solver for Metal and Dielectric Structures **2-2**
MoM Formulation **2-2**
Neighbor Region **2-6**
Singularity Extraction **2-7**

2-D Field Solver **2-9**

RF PCB Examples

- “Stepped Impedance Maximally Flat Lowpass Filter for Microwave Applications” on page 1-2
- “On-Chip Square Spiral Inductor for Si RFIC Application” on page 1-8
- “Bandstop and Bandpass Filters with Open Microstrip Line Stubs Using Behavioral and EM Simulation” on page 1-14
- “Dual-Fed Square Microstrip Patch Antenna for BLE Applications” on page 1-22
- “Create and Analyse Dual Characteristics of Complementary Split Ring Resonators from Gerber Files” on page 1-37
- “Prototype, Design, and Analysis of SIW based Microstrip Tapered Transmission Line” on page 1-44
- “Model and Analyze Microstrip Interdigital Capacitor as Bandpass filter” on page 1-50
- “Introduction to Passive Planar Spiral Inductors ” on page 1-55
- “Design and Analysis of Hairpin Micro-Strip Line Bandpass Filter” on page 1-66
- “Analyzing Crosstalk Between PCB Traces” on page 1-77
- “Create and Analyze PCB Interconnects using Custom Traces” on page 1-88
- “Routing Discipline to Reduce Crosstalk” on page 1-99
- “Introduction to Equal and Unequal Split Wilkinson Power Splitter” on page 1-106
- “Introduction to 4-Port Couplers” on page 1-117
- “Design Two-Stage Low Noise Amplifier Using Microstrip Transmission Line Matching Network” on page 1-137
- “Design and Analyze Compact UWB Low Pass Filter Using pcbComponent” on page 1-152
- “Design and Analyze Band Stop Filter using pcbComponent” on page 1-165
- “Design and Analyze HighPass Filter Using pcbComponent” on page 1-177
- “Design of Quarter-Wave Transformer for Impedance Matching Applications” on page 1-190
- “Corporate Feed Divider Network for a Linear Patch Antenna array” on page 1-197
- “Microstrip Composite Bandpass Filters for Ultra-Wideband (UWB) Wireless Communications” on page 1-204
- “UWB Bandpass Filter Using Open and Short-Circuited Stubs” on page 1-224
- “Comparison of Lumped and Distributed EM Models for Low Pass Filters” on page 1-231
- “Variations of Open and Short Stub Filters” on page 1-237
- “Microstrip Ultra-Wideband Bandpass Filter with Cascaded Broadband Bandpass and Bandstop Filters” on page 1-244

Stepped Impedance Maximally Flat Lowpass Filter for Microwave Applications

This example shows you how to design a stepped-impedance lowpass filter for X-band applications.

Microstrip filter plays an important role in microwave applications. Almost all communication systems contain an RF front end which performs signal processing using RF filters. Microwave lowpass filters attenuate the unwanted signals above the cutoff frequency. These filters have very low insertion loss and are easy to fabricate in compact sizes.

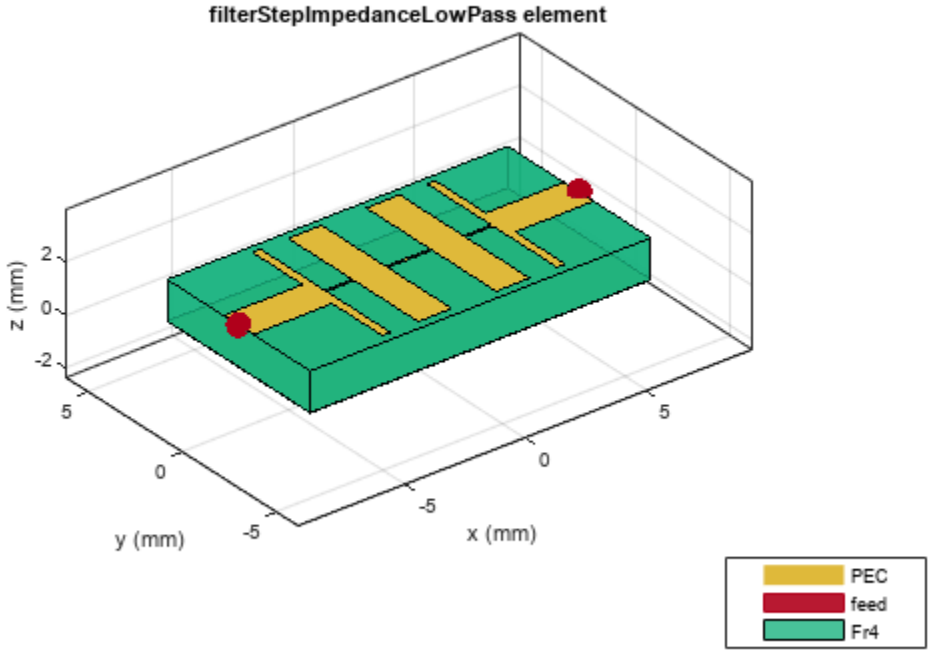
The stepped impedance low-pass microstrip filters is a cascaded structure of alternating high and low impedance transmission lines. These lines act as semi lumped elements as they are much shorter than the associated guided wavelength. The high-impedance lines act as series inductors and the low-impedance lines act as shunt capacitors. Therefore, this filter structure is directly realizing the L-C ladder type (LC-Pi) of low-pass filters.

Here the stepped impedance low-pass filter and the lumped element model is analyzed for a cut off frequency of 9 GHz. The results of lumped element model and the simulation model are compared and validated [1].

Design Stepped Impedance Low-Pass Filter

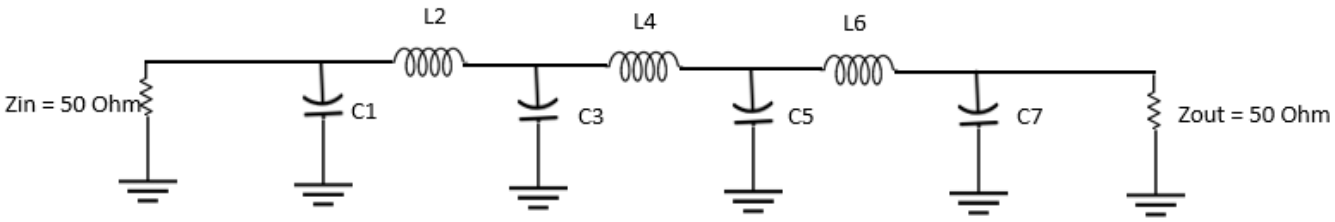
Design a stepped impedance low-pass filter at 9 GHz on an aluminium substrate and visualize it.

```
freq = 9e9;
sub  = dielectric("Name",{'Fr4'},"EpsilonR",9.6,"LossTangent",0,"Thickness",1.6e-3);
obj  = filterStepImpedanceLowPass;
obj.FilterOrder = 7;
obj.Substrate = sub;
d   = design(obj,freq,'Z0',50,'HighZ',120,'LowZ',20);
figure; show(d);
```



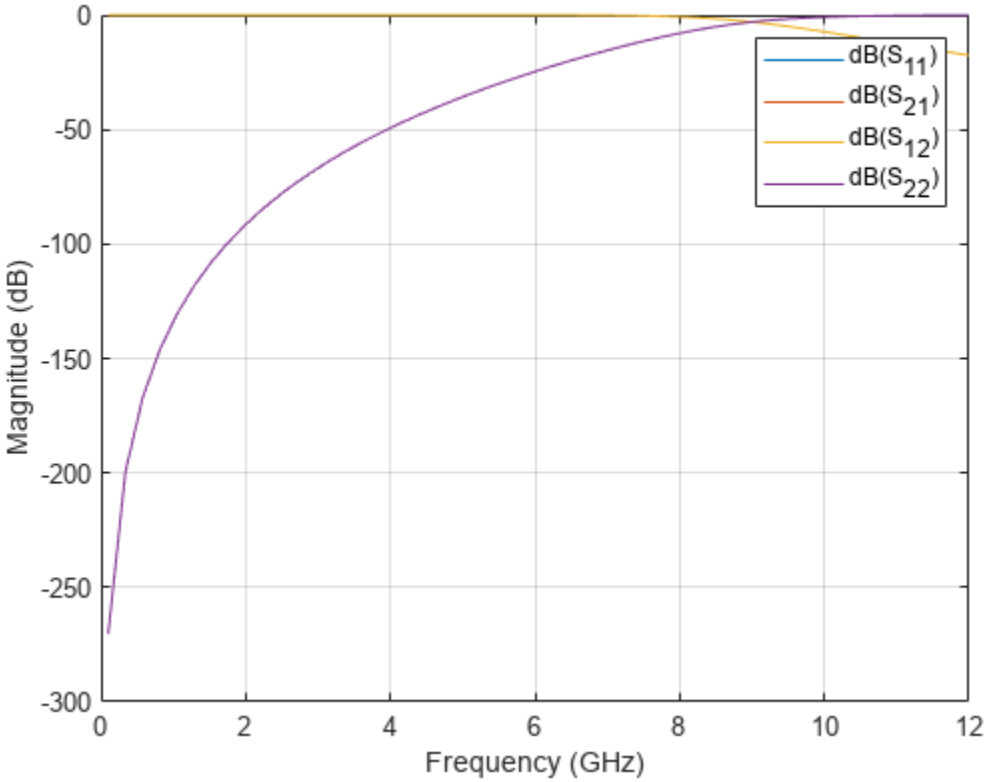
Analyze Using Lumped Element Model

The figure shows the schematic of lumped element model for a 7th order low-pass LC-Pi configuration.



Model the 7th order maximally flat low-pass filter at 9 GHz using the `rffilter` object from RF Toolbox. Analyze its s-parameter behavior with input and output impedance at 50 ohm.

```
Lpfilter = rffilter('FilterType','Butterworth','ResponseType','LowPass', ...
                'Implementation','LC pi','FilterOrder',obj.FilterOrder,'PassbandFrequency',freq,
                'PassbandAttenuation',3.0103,'Zin',50,'Zout',50,'Name','Filter');
spar = sparameters(Lpfilter,linspace(0.1e9,12e9,51));
figure; rfplot(spar);
```

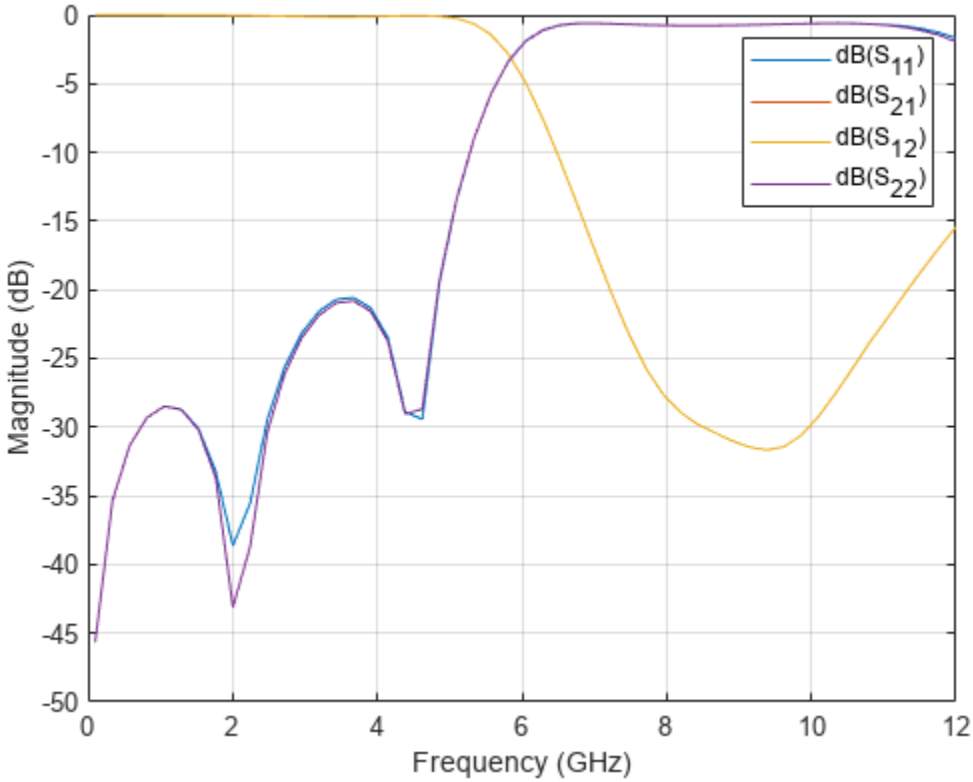


It is clear from the scattering parameters that the cut-off frequency is at 9 GHz and the stop band attenuation is more than 20dB using the lumped element model.

Analyze Using Simulation Model

Analyze the simulation model for its s-parameter behavior at 50 ohm reference impedance.

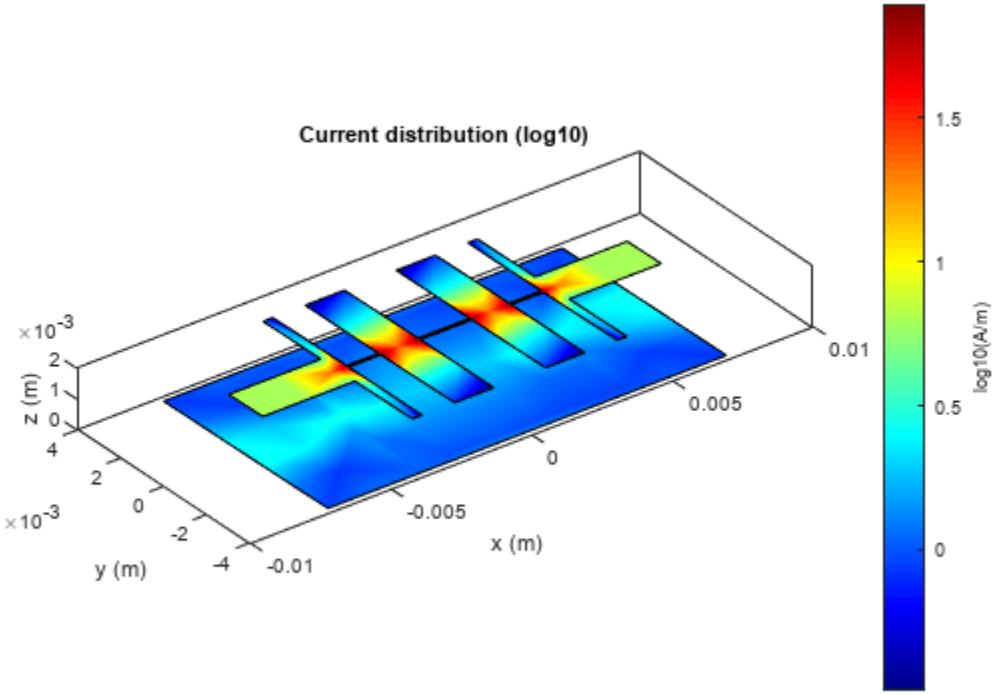
```
s11 = sparameters(d,linspace(0.1e9,12e9,51));
figure; rfplot(s11)
ylim([-50,0]);
```



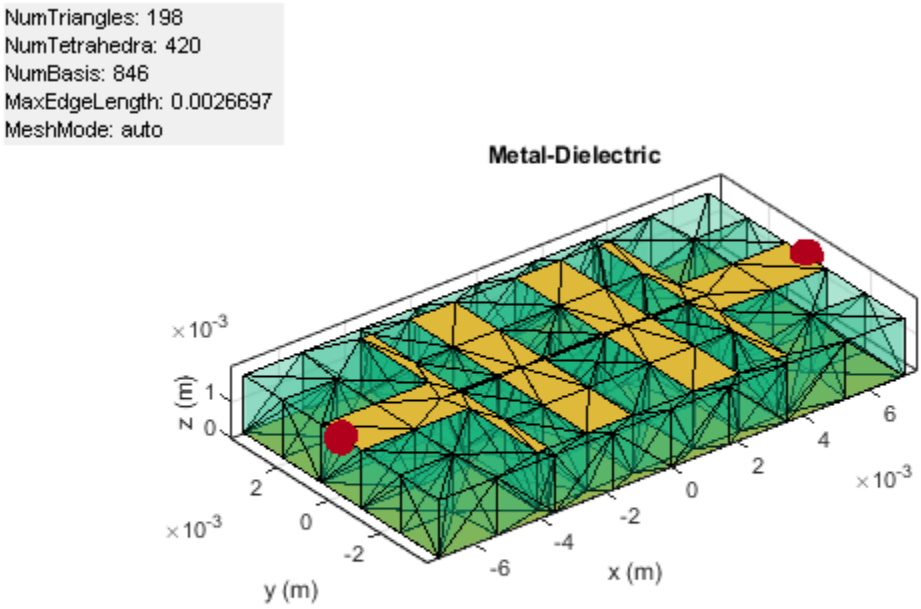
The scattering parameters obtained from the filter layout realization using simulation model are more accurate because the coupling effect between microstrip line sections are considered along with the losses. The s-parameters plot shows that the cut-off frequency is reduced to 5.8 GHz. This reduction is due to two reasons. One is the coupling effect between each section and the second is the design method for modeling the filter which uses the analytical equations where those effects are not considered. Hence, the design will give approximate results, and you should optimize the length and width of the High and Low impedance lines to shift the cut-off frequency to the desired value.

Use the `current` function to plot the current distribution on the surface of the stepped impedance low-pass filter below the cut off frequency and view its mesh.

```
figure;
current(d,3e9,'scale','log10');
```




```
figure;
mesh(d);
```



Designing the stepped impedance low-pass filter at S-band and below will exhibit less shift in frequency nearly about 0.5 GHz when compared with the shift in design at high frequencies.

Conclusion

The stepped impedance low-pass filter is designed and analyzed using the `design` function at 9 GHz cut off frequency. The design is compared with both lumped model and the simulation model to understand the difference in shift in the frequency. It is seen that the shift in the frequency for simulation model is about 2.9 GHz which is due to the coupling effect. Hence, the design parameters of the stepped impedance low-pass filter should be optimized to achieve the desired cut-off frequency.

Reference

1. Salama,Y Battah,A Abuelhaija, 'Stepped Impedance 7th order Maximally Flat Low-Pass Filter Using Microstrip Line for X-Band Applications', *Journal of Physics: Conference Series (*ICERIA *)*.

2. Sheetal.Mitra, 'Stepped Impedance Microstrip Low-Pass Filter Implementation for S-band Application', *International Journal of Latest Trends in Engineering and Technology (IJLTET)*, Vol. 5, Issue 3, May 2015.

On-Chip Square Spiral Inductor for Si RFIC Application

This examples shows you how to model a square spiral inductor for Si (silicon) RFIC application.

The Silicon substrate is chosen for its low cost of Si IC fabrication over GaAs IC fabrication. The potential for integration with baseband circuits makes Si the process of choice in many RFIC applications.

The RF planar inductors play a vital role in wireless communication to meet the desired requirements like low supply voltage, low cost, low power dissipation, low noise, high frequency of operation, and low distortion.

The square spiral inductors are the most common in Si RF IC's. Design the square spiral inductor over the Si substrate and the SiO2 insulator at 3 GHz.

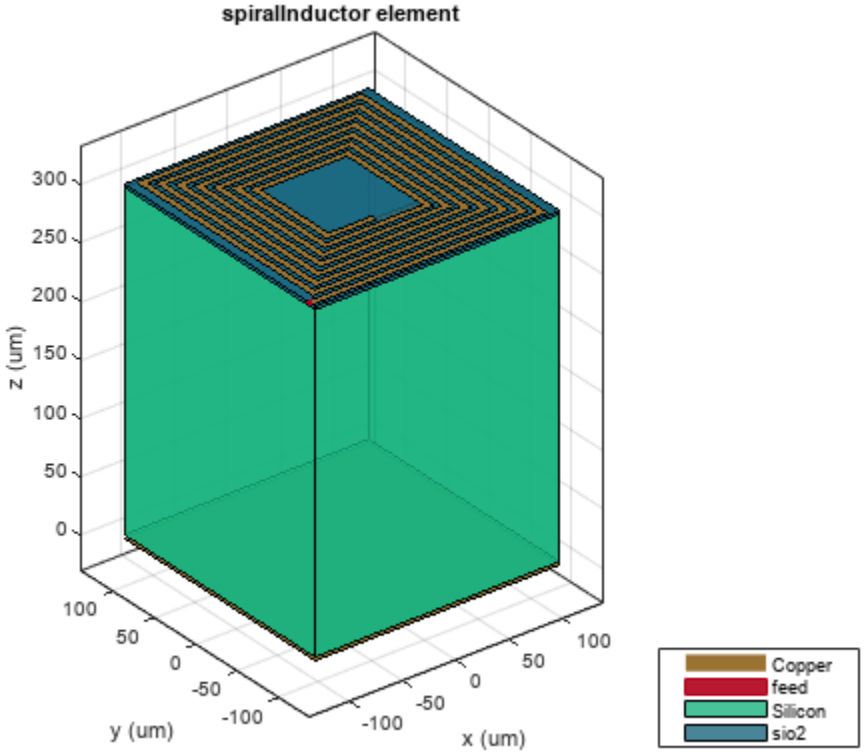
Define Parameters

```
N  = 7;
ID = 81e-6;
W  = 6e-6;
S  = 4e-6;
% Height from the ground plane to the spiral
H  = 303e-6;
GL = 230e-6;
GW = 230e-6;
```

Create Planer Square Spiral Inductor on Si and Sio2 Substrate

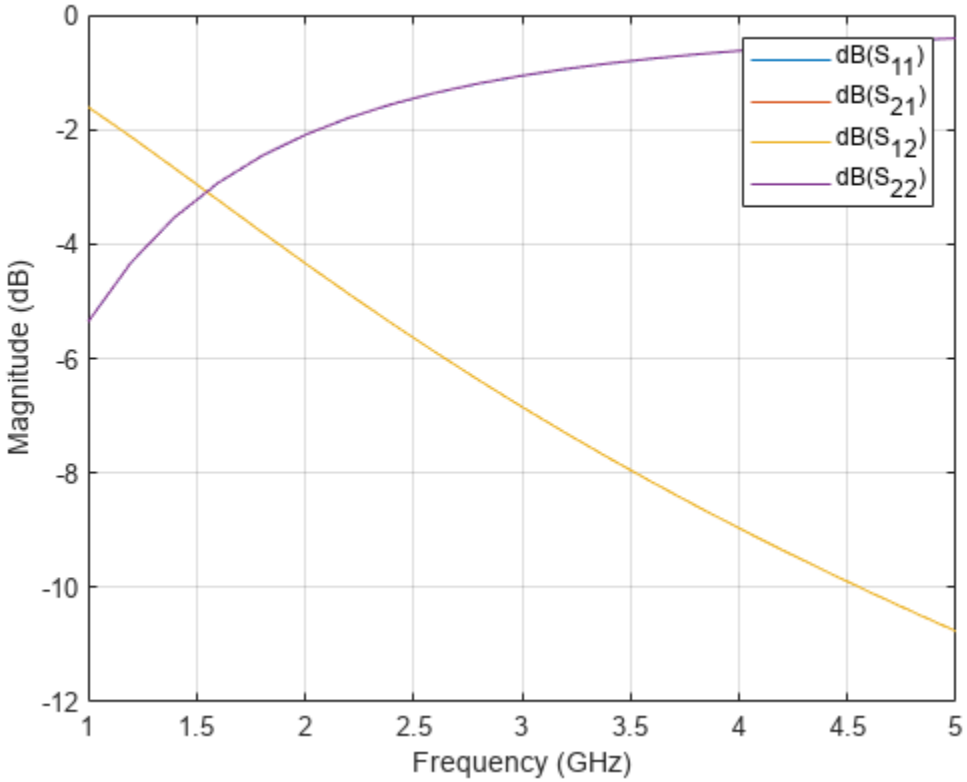
Model the 7 turn square spiral inductor is modelled using the defined parameters in microstrip line form on Silicon and Sio2 substrate

```
Ind = spiralInductor;
Ind.NumTurns          = N;
Ind.InnerDiameter     = ID;
Ind.Width             = W;
Ind.Spacing           = S;
Ind.Height            = H;
Ind.GroundPlaneLength = GL;
Ind.GroundPlaneWidth  = GW;
d = dielectric('Name',{'Silicon','sio2'},'EpsilonR',[11.9,4.1],'LossTangent',[0.005,0],'Thickness
Ind.Substrate = d;
figure; show(Ind);
```



Use the `sparameters` function to compute the s-parameters and plot it.

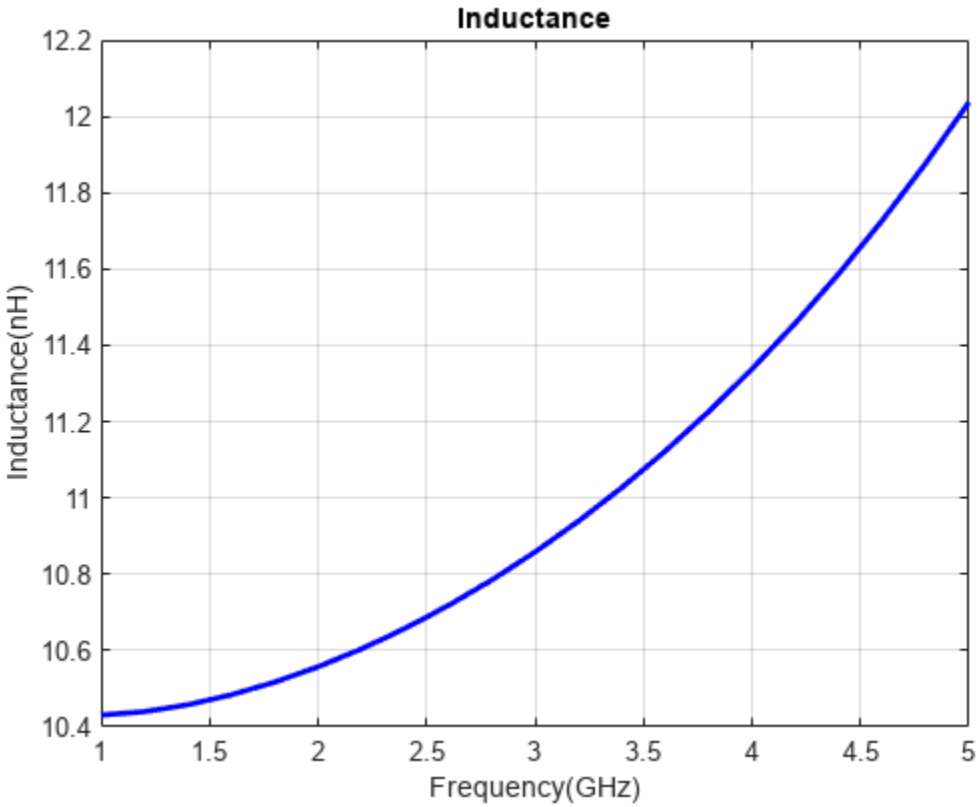
```
spar = sparameters(Ind,linspace(1e9,5e9,21));
figure; rfplot(spar);
```



The s-parameters plot shows that after 430 MHz the S11 and S22 monotonically increases towards 0 dB. S12 and S21 decreases showing that energy is stored in inductor and not radiated. The behavior of S11 and S21 satisfies the law of conservation of energy.

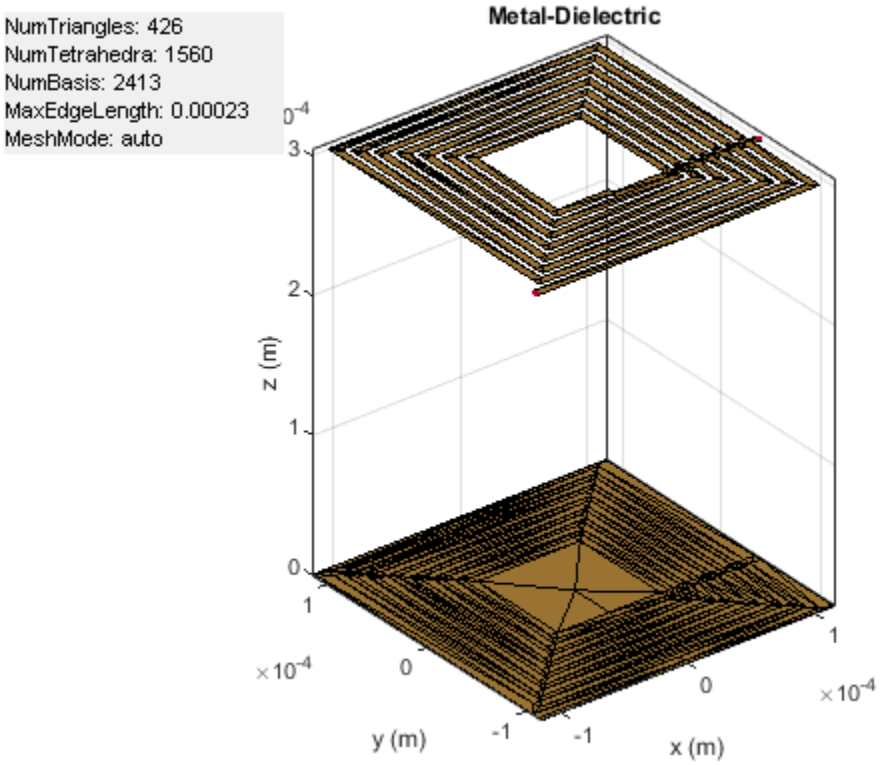
Use the inductance function to calculate the measure of inductance of the square spiral inductor at 3 GHz.

```
figure; inductance(Ind,linspace(1e9,5e9,21));
```



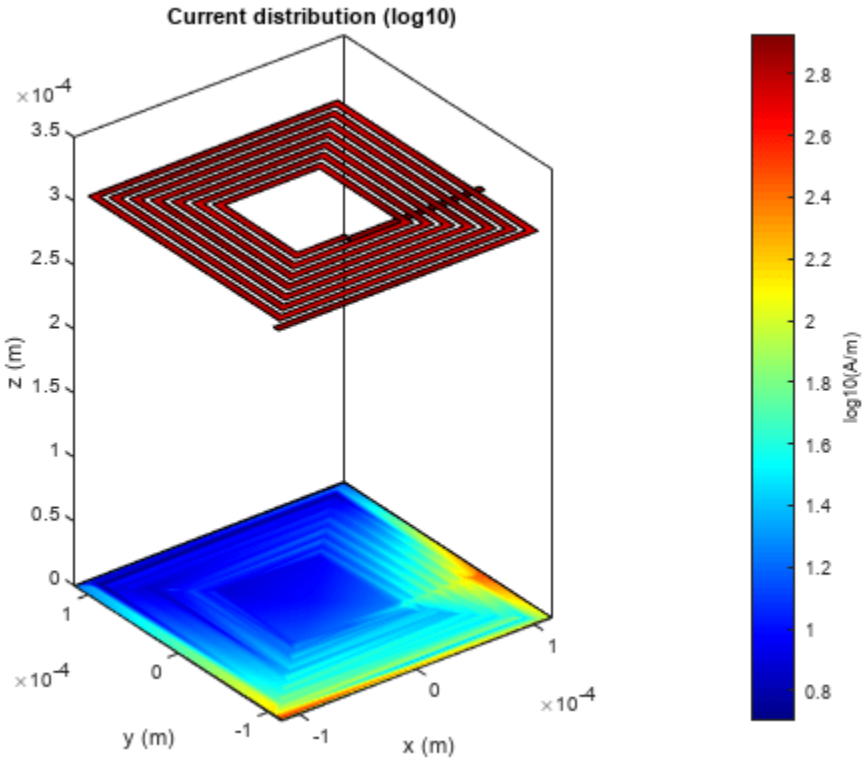
It is observed from the plot that the inductance of the spiral inductor increases with increase in frequency. Based on analyzing the Q- factor the peak in Q will be the design frequency of the inductor [2] , According to the reference the design frequency is around 3 GHz and the calculated inductance of the spiral is around 11.09 nH which is suitable for RFIC application.

```
figure;
mesh(Ind,'View','Metal');
```



Use the current function to plot the current distribution on the surface of the square spiral Inductor.

```
figure;
current(Ind,3e9,'scale','log10');
```



Reference

1 *Beeresha R Sa, A M Khanb, Manjunatha Reddy H Vc ,'' The Design and EM-Simulation of Square Spiral Inductor Using SimpleEquations''*,*Materials Today: Proceedings 5 (2018) 10875–10882.*

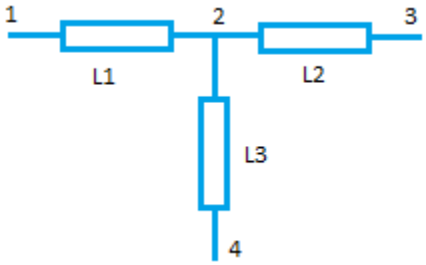
2 *Tuan Huu Bui, "Design and Optimization of a 10 nH Square-Spiral Inductor for Si RF Ics",* University of North Carolina at Charlotte, October, 1999

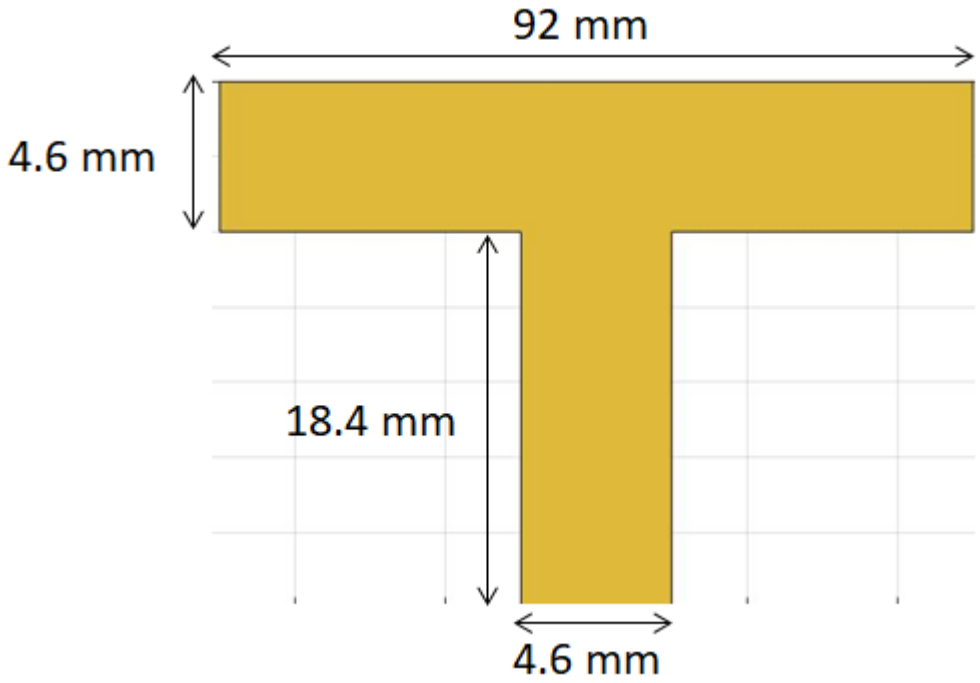
Bandstop and Bandpass Filters with Open Microstrip Line Stubs Using Behavioral and EM Simulation

This example shows different ways to create microstrip Line bandpass and bandstop filters using open circuit stub.

Convert the RF PCB catalog into a circuit element using a `pcbElement` object and build a Tee shaped circuit. Analyze the circuit using the behavioral model and EM simulation. Use the `traceTee` shape object in RF PCB Toolbox to create a tee shaped trace. Convert the shape into a PCB component and analyze it. Analyze the PCB using EM Simulation and compare the results with the circuit counterpart.

The dimensions of the microstrip line and the stub in the figure are taken from the reference. The left image shows the circuit representation with the port numbers and the right side image shows the `traceTee` structure with same dimensions.





Circuit Element with Behavioral Model for Microstrip Line

Create a `circuit` object and three `microstripLine` objects to build the circuit as shown in the figure. Update the value of EpsilonR and Height to 2.33 and 1.57 mm respectively.

```
ckt = circuit;
c1 = microstripLine('Length',46e-3,'Width',4.6e-3,'Height',1.57e-3);
c1.Substrate.EpsilonR = 2.33;

c2 = microstripLine('Length',46e-3,'Width',4.6e-3,'Height',1.57e-3);
c2.Substrate.EpsilonR = 2.33;

c3 = microstripLine('Length',18.4e-3,'Width',4.6e-3,'Height',1.57e-3);
c3.Substrate.EpsilonR = 2.33;
```

Use the `pcbElement` object to convert the microstrip lines to circuit objects and set the `Behavioral` flag to `true` so that the computation is done using the analytical equations. In the circuit model, the microstrip discontinuities are not modelled, hence the results might be slightly different from EM analysis.

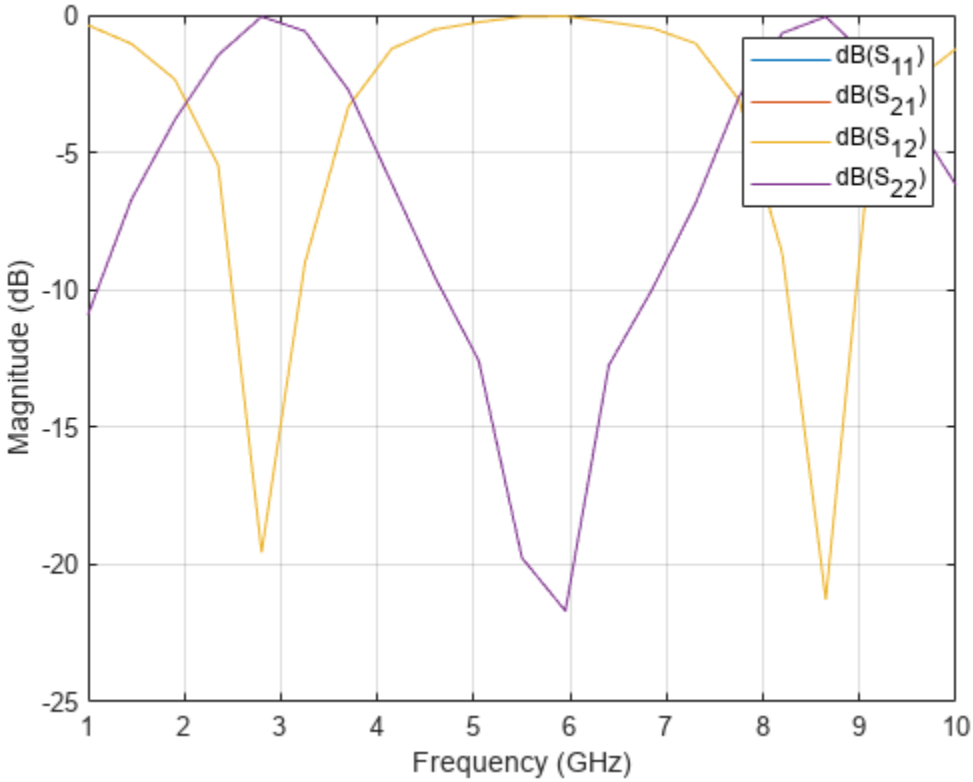
```
p1  = pcbElement(c1,'Behavioral',true);
p2 = pcbElement(c2,'Behavioral',true);
p3 = pcbElement(c3,'Behavioral',true);
```

Use the add function to form the circuit and use the ports numbers as shown in the circuit above. Use port numbers 1 and 2 for the first microstrip line and connect another microstrip line from ports 2 to 3 which forms a series connection. Connect the third microstrip line from the port 2 to 4 which forms a tee network.

```
add(ckt,[1 2 0 0],p1);
add(ckt,[2 3 0 0],p2);
add(ckt,[2 4 0 0],p3);
setports(ckt,[1 0],[3 0]);
```

Use the `sparameters` function to calculate the s-parameters for the circuit and plot it using the `rfplot` function.

```
S = sparameters(ckt,linspace(1e9,10e9,21));
figure,rfplot(S);
```



Circuit Element with EM simulation for Microstrip Line

Use the `pcbElement` object to convert the microstrip lines to circuit objects and set the `Behavioral` flag to `false` so that the computation is done using EM Simulation.

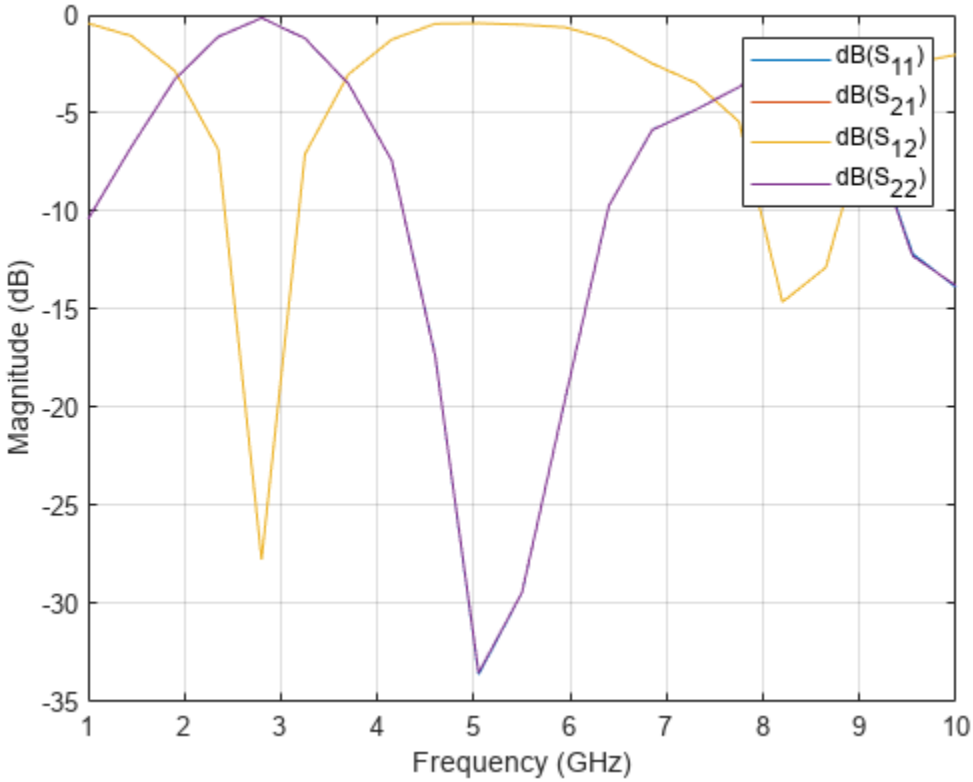
```
ckt1 = circuit;
p1   = pcbElement(c1,'Behavioral',false);
p2   = pcbElement(c2,'Behavioral',false);
p3   = pcbElement(c3,'Behavioral',false);
```

Use the add function to form the circuit and use the port numbers as shown in the circuit above. Use port numbers 1 and 2 for the first microstrip line and connect another microstrip line from ports 2 to 3 which forms a series connection. Connect the third microstrip line from the port 2 to 4 which forms a tee network.

```
add(ckt1,[1 2 0 0],p1);
add(ckt1,[2 3 0 0],p2);
add(ckt1,[2 4 0 0],p3);
setports(ckt1,[1 0],[3 0]);
```

Use the `sparameters` function to calculate the s-parameters for the circuit and plot it using the `rfplot` function. Each microstripLine section is solved using the EM solver and the s-parameters calculated are combined to give the combined response of the circuit.

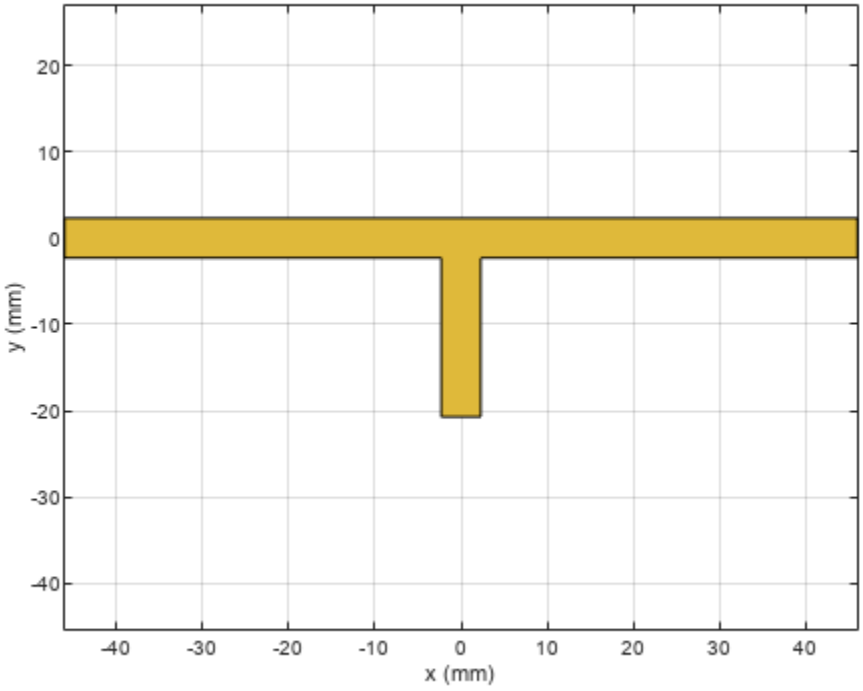
```
S = sparameters(ckt1,linspace(1e9,10e9,21));
figure;
rfplot(S);
```



Creating PCB Stack of traceTee Shape with EM Simulation

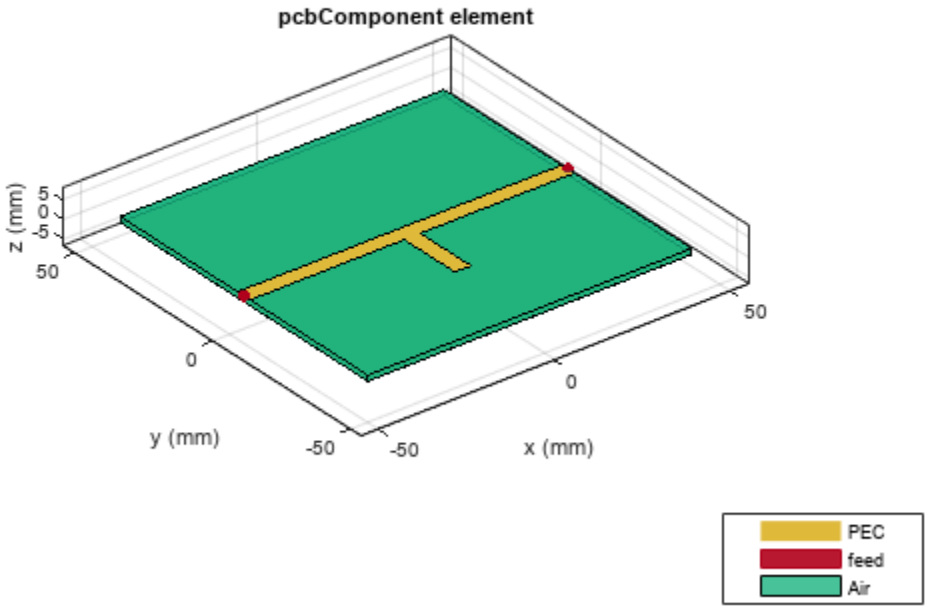
Use the `traceTee` shape to create the open circuit stub and use the dimensions as shown in the figure above and visualize it.

```
obj = traceTee;
obj.Length = [92e-3 18.4e-3];
obj.Width  = [4.6e-3 4.6e-3];
figure;
show(obj);
```



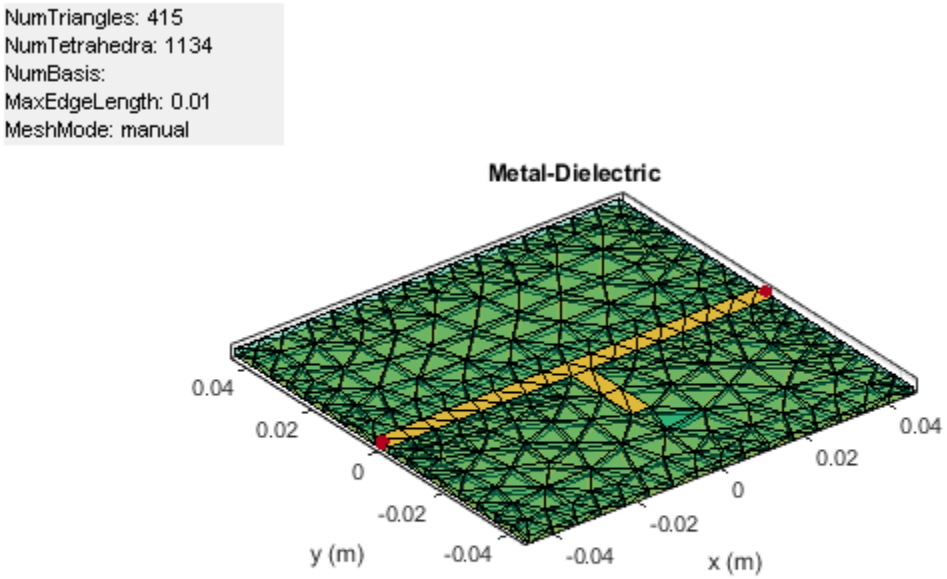
Use the `pcbComponent` to convert the shape into a PCB stack. The `pcbComponent` creates the PCB stack up for traceTee shape and assigns the FeedLocations at the three open ends of the shape. Assign the dielectric and groundplane to the `Layers` property of the `pcbComponent`. Also assign the `BoardShape` to the grounplane. In the current design, the stub is required as open circuit, hence delete the feed at the third location.

```
pcb = pcbComponent(obj);
gnd = traceRectangular('Length',obj.Length(1),'Width',90e-3);
d   = dielectric('EpsilonR',2.33);
d.Thickness = 1.57e-3;
pcb.BoardThickness = 1.57e-3;
pcb.Layers{2}      = d;
pcb.Layers{3}      = gnd;
pcb.BoardShape     = gnd;
pcb.FeedLocations(3,:)=[];
figure;
show(pcb);
```



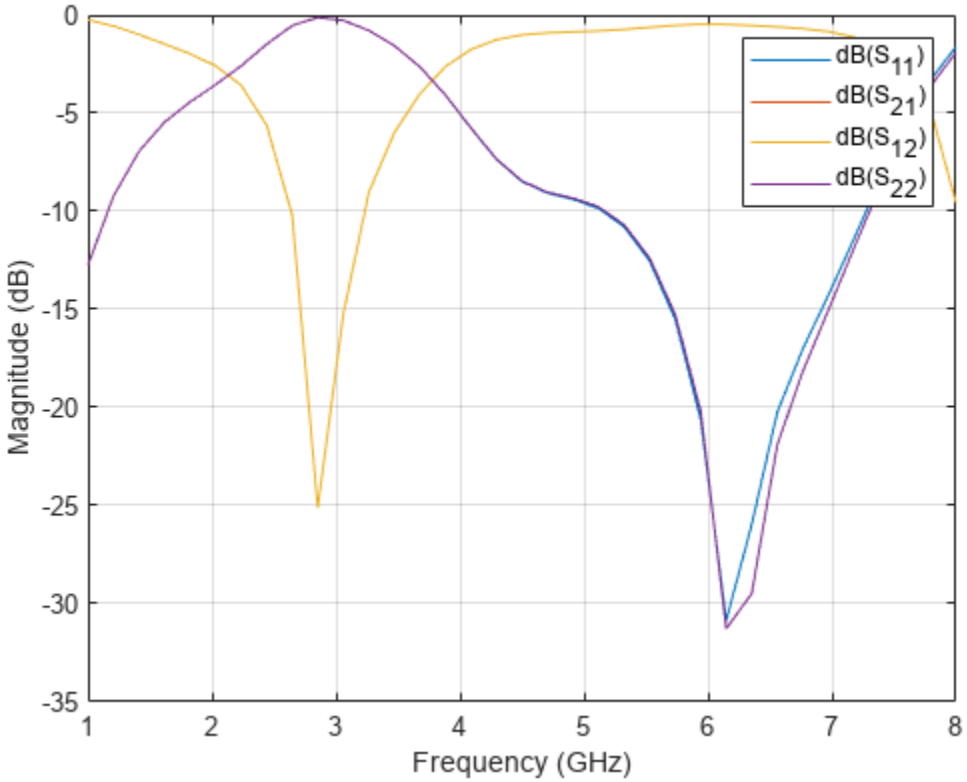
Use the `mesh` function to manually mesh the structure and set the `MaxEdgeLength` to 10 mm.

```
figure,mesh(pcb,'MaxEdgeLength',10e-3);
```



Use the `sparameters` function to calculate the s-parameters of the structure and plot it using `rfplot` function.

```
S1 = sparameters(pcb,linspace(1e9,8e9,35));
figure;
rfplot(S1);
```



Design at Specific Frequency

The resonance of the bandstop filter depends on the stub length which needs to be quarter wavelength at the resonant frequency. In the above example the stub is quarter wavelength at 2.9 GHz and hence is resonance is observed at that frequency. If the resonance is expected at any other frequency then the stub length needs to be quarter wavelength at that frequency.

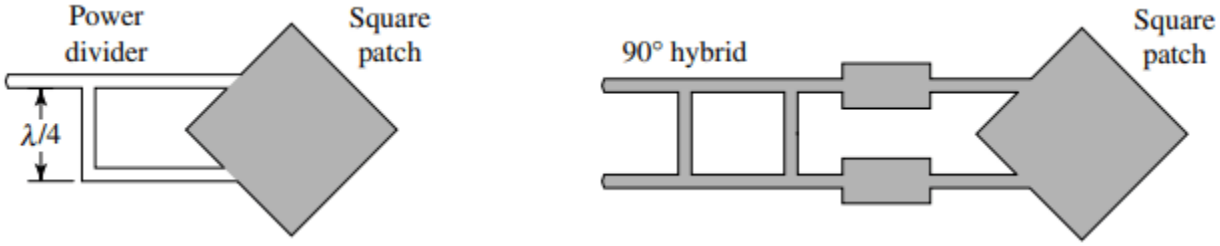
References

1 Alexander B. Yakovlev, Ahmed I. Khalil , Efficient MOM-based generalized scattering matrix method for the integrated circuit and multilayered structures in waveguide

Dual-Fed Square Microstrip Patch Antenna for BLE Applications

This example shows how to use the `pcbComponent` and `pcbcascade` functionality of RF PCB Toolbox™ to design and analyze the dual-fed square microstrip patch antenna for Bluetooth Low Energy (BLE) applications.

You can use different feed arrangements to achieve circular and elliptical polarization or by using two orthogonal modes excited with a 90 degree phase difference between them. The two orthogonal modes can be accomplished by adjusting the physical dimensions of the patch and using either single or multiple feeds. For a square patch element, the easiest way to excite for circular polarization is to feed the element at two adjacent edges, to excite the two orthogonal modes as shown in the figure. For a quadrature phase difference, feed the element with a 90 degree power divider or a 90 degree hybrid as shown in the figure below.



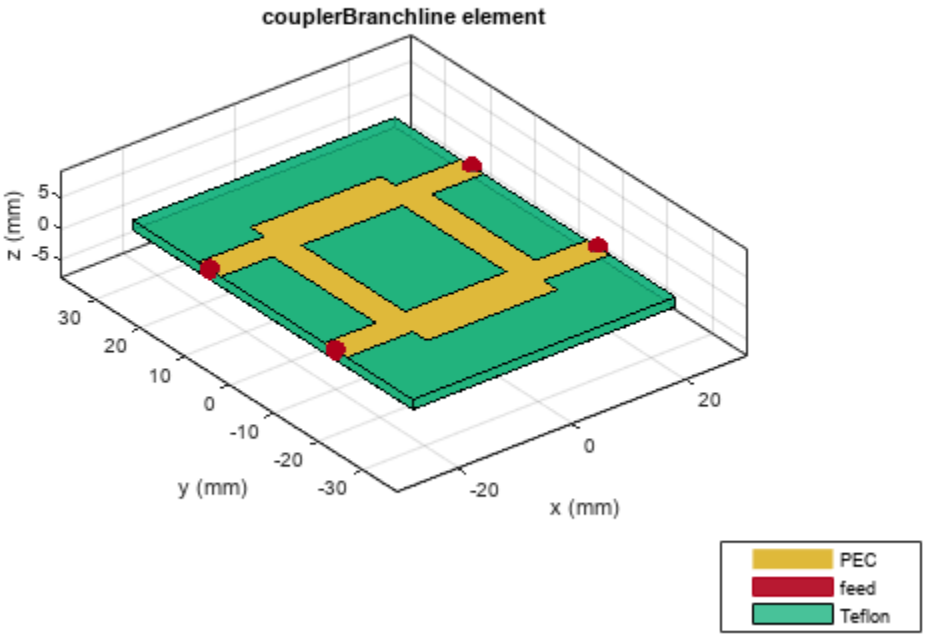
Create Variables

```
Centerfreq = 2.4e9;
freq       = linspace(2e9,3e9,41);
```

Design of Branchline Coupler

Use the `design` function on the `couplerBranchline` object to create a branchline coupler at the desired frequency and visualize it. The default substrate for branchline coupler is Teflon.

```
coupler = design(couplerBranchline,Centerfreq);
figure;
show(coupler);
```



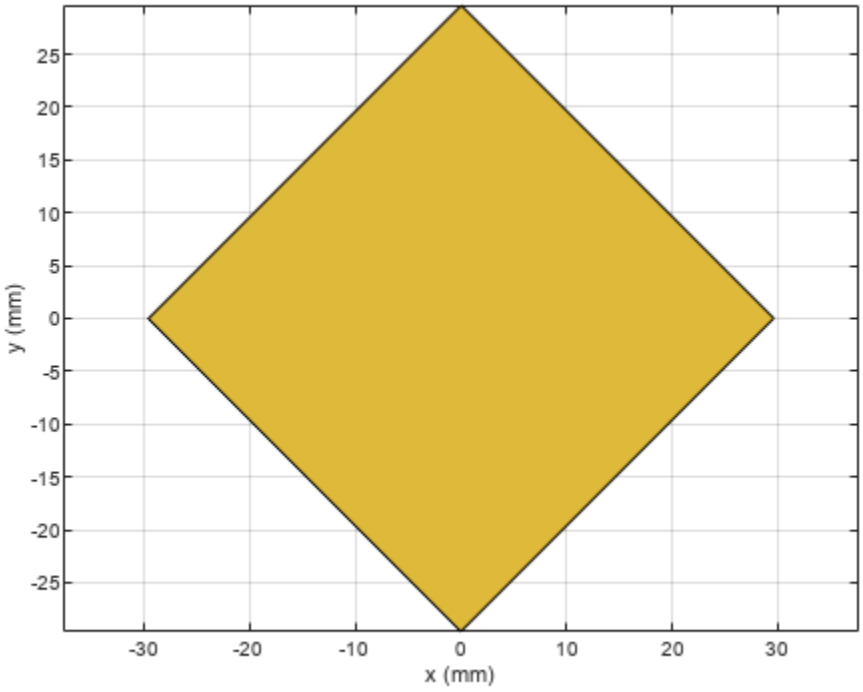
Create Variables for Square Patch and Connecting Feed Lines

The `patchLength` variable creates a square patch. The `feedLineWidth` and `feedLineLength` variable creates the feed lines to the antenna. The `portSpacing` matches the gap between the feed lines to that of the output ports of the branchline coupler. The `patchLength and feedLineLength` are close to half wavelength at design frequency.

```
patchLength    = 41.9e-3;
feedLineLength = 65.7e-3;
feedLineWidth  = 3e-3;
portSpacing    = coupler.ShuntArmLength/2+coupler.PortLineWidth/2;
```

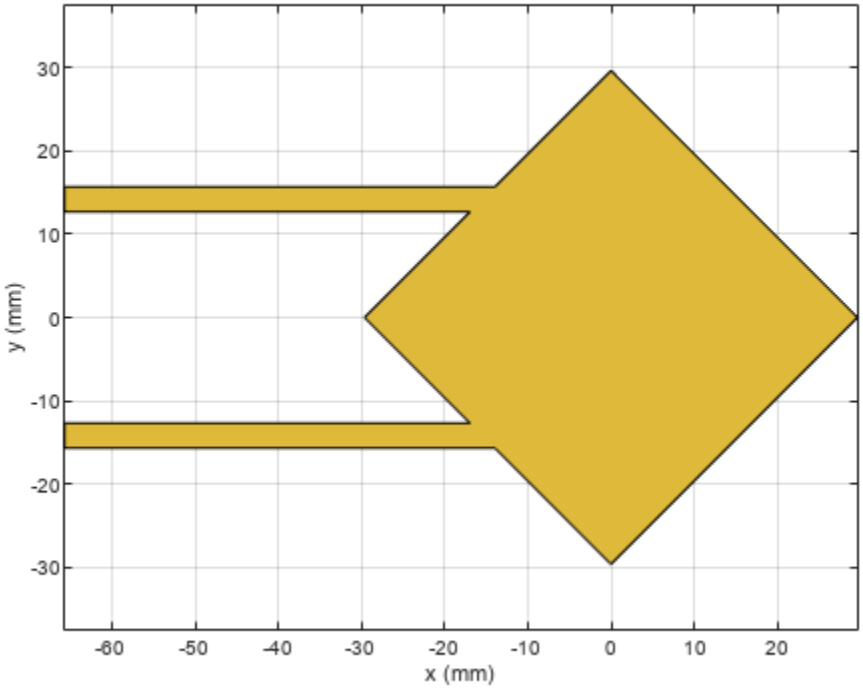
Use the `traceRectangular` object to create a square patch with side length as `patchLength`. Use the `rotateZ` function to rotate the shape by 45 degrees and visualize it.

```
patch    = traceRectangular('Length',patchLength,'Width',patchLength);
patch    = rotateZ(patch,45);
figure;
show(patch);
```



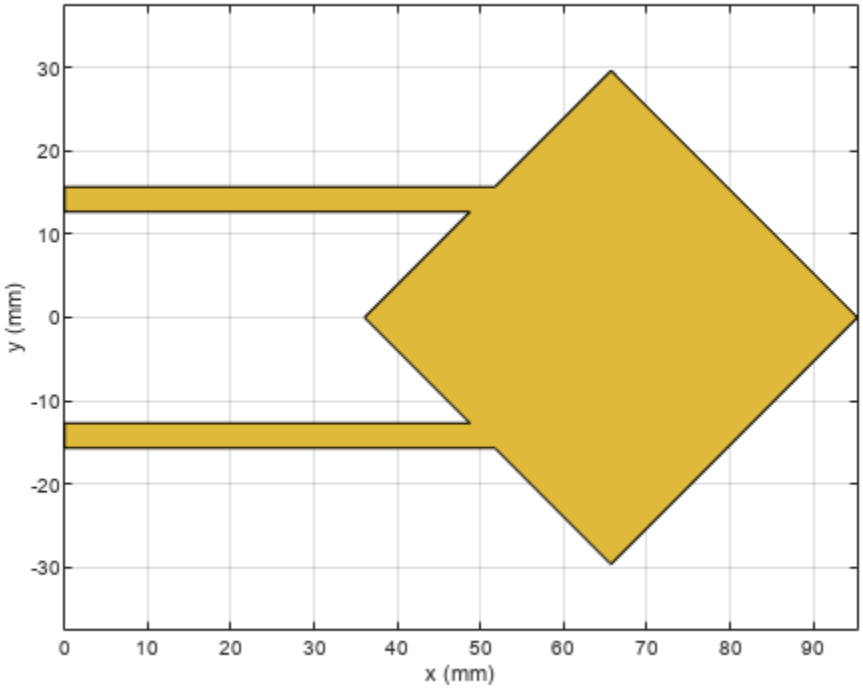
Use the `traceRectangular` object to create two feed lines with same length, width, and equal spacing on either side of the X-axis. Perform a Boolean add operation for the shapes `patch`, `feedline1` and `feedline2` and visualize it.

```
feedLine1 = traceRectangular('Length',feedLineLength,'Width',feedLineWidth,'Center',[-feedLineLe
feedLine2 = traceRectangular('Length',feedLineLength,'Width',feedLineWidth,'Center',[-feedLineLe
antShape = patch+ feedLine1+ feedLine2;
figure;
show(antShape);
```



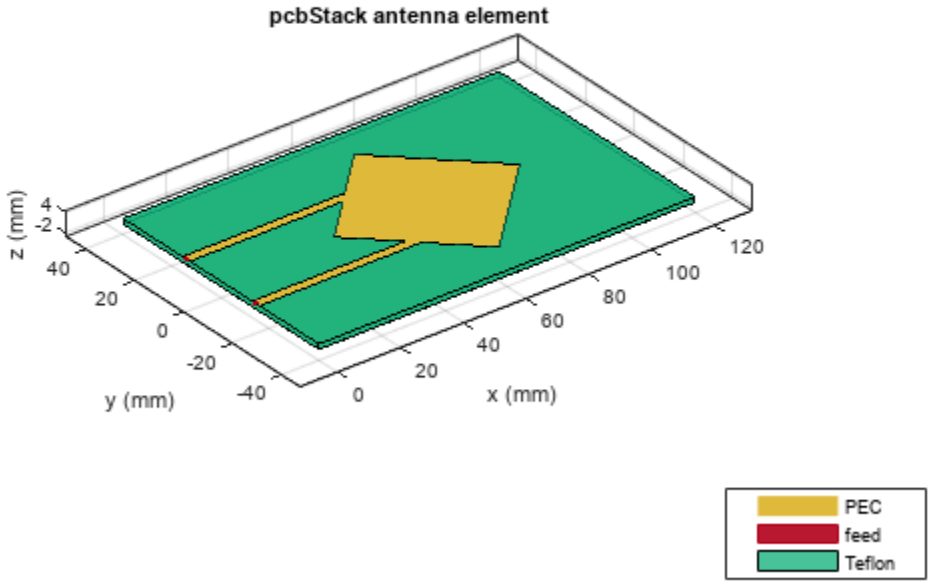
Translate the shape along the X-axis such that the feedlines are aligning at x = 0.

```
translate(antShape,[feedLineLength,0,0]);
```



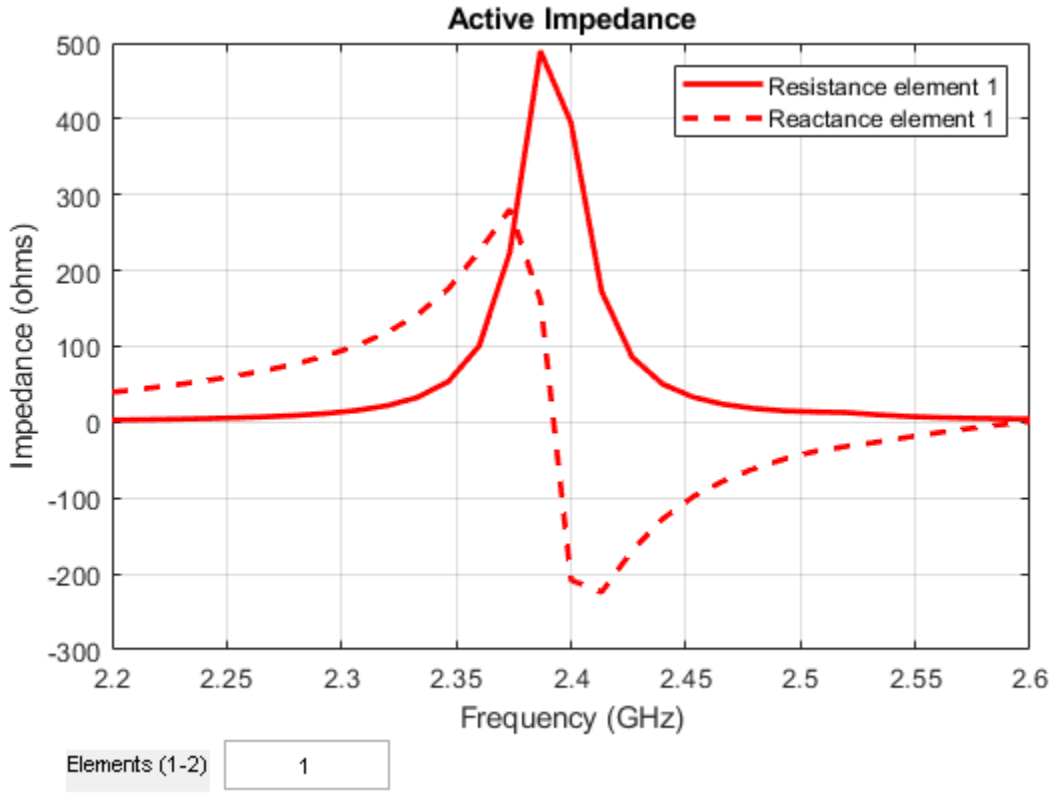
Define the substrate parameters and create a dielectric object. Create a groundplane using the `antenna.Rectangle` object and use `pcbStack` to create a PCB antenna. Assign the dielectric and ground plane to the `Layers` property on `pcbStack`. Assign the `FeedLocations` to the edge of the feedline and set the `BoardThickness` to `Height` on the pcbStack and visualize the antenna.

```
EpsilonR     = coupler.Substrate.EpsilonR;    % Dielectric EpsilonR
Height       = coupler.Height;                % Height of the Substrate
LossTangent  = coupler.Substrate.LossTangent; % Loss Tangent of the Substate

d1  = dielectric('Name',{'Teflon'},'EpsilonR',EpsilonR,'LossTangent',LossTangent,'Thickness',Hei
Gnd = antenna.Rectangle('Length',120e-3,'Width',80e-3,'Center',[120e-3/2,0]);


ant = pcbStack;
ant.BoardThickness = Height;
ant.Layers = {antShape,d1,Gnd};
ant.BoardShape    = Gnd;
ant.FeedLocations = [0 portSpacing 1 3;0,-portSpacing 1 3];
figure;
show(ant);
```



Use the `impedance` function to plot the impedance of the antenna from 2.2 GHz to 2.6 GHz

```
figure;
impedance(ant,linspace(2.2e9,2.6e9,31));
```



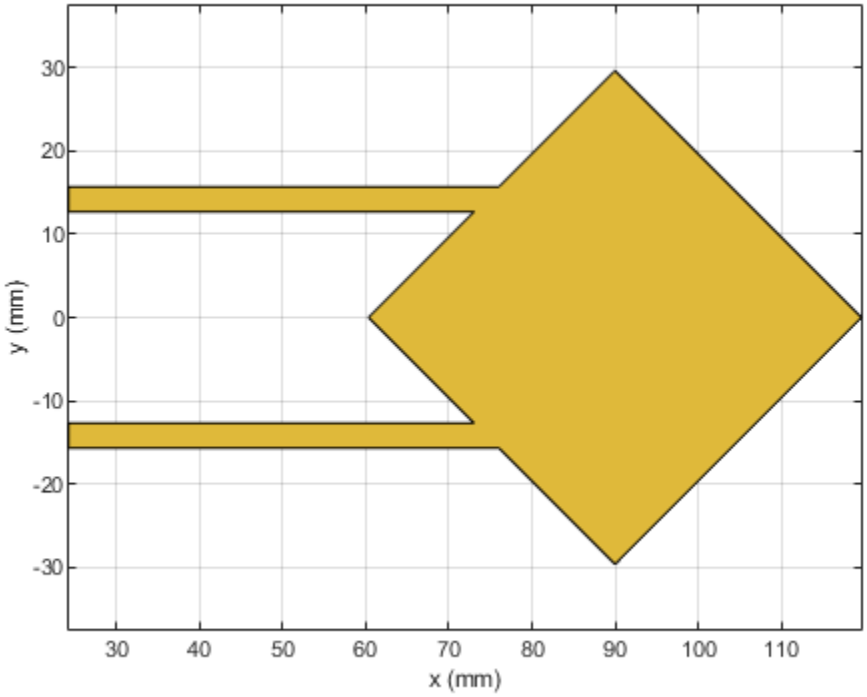
The result shows that the impedance of the patch is around 400 ohms at 2.4 GHz. In order to match the impedance of the patch with the 50 ohm line, connect a quarter wavelength transformer at the end of the feedline to transform the impedance to 50 ohm. The impedance of the quarterwave transformer is the geometric mean of the impedances to be matched. Hence the geometric mean of 50 ohm and 400 ohm value comes out to be 141 ohms.

Use the `microstripLine` design to calculate the width of the line with a Z0 of 141 ohm.

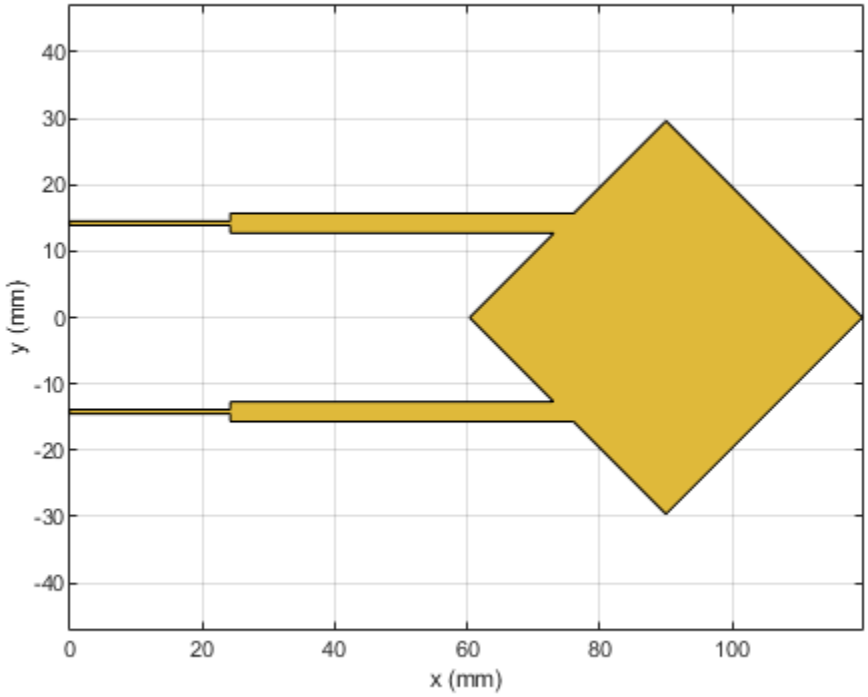
```
line = microstripLine;
line.Height = coupler.Height;
line = design(line,Centerfreq,"LineLength",0.25,"Z0",141);
```

Use the `traceRectangular` object to create two quarter wave transformer lines with same length, width, and equal spacing on either side of the X-axis. Perform a Boolean add operation for the shapes `patch`, `Line1` and `Line2` and visualize it.

```
Line1     = traceRectangular('Length',line.Length,'Width',line.Width,'Center',[line.Length/2,por
Line2     = traceRectangular('Length',line.Length,'Width',line.Width,'Center',[line.Length/2,-po
translate(antShape,[line.Length,0,0]);
```



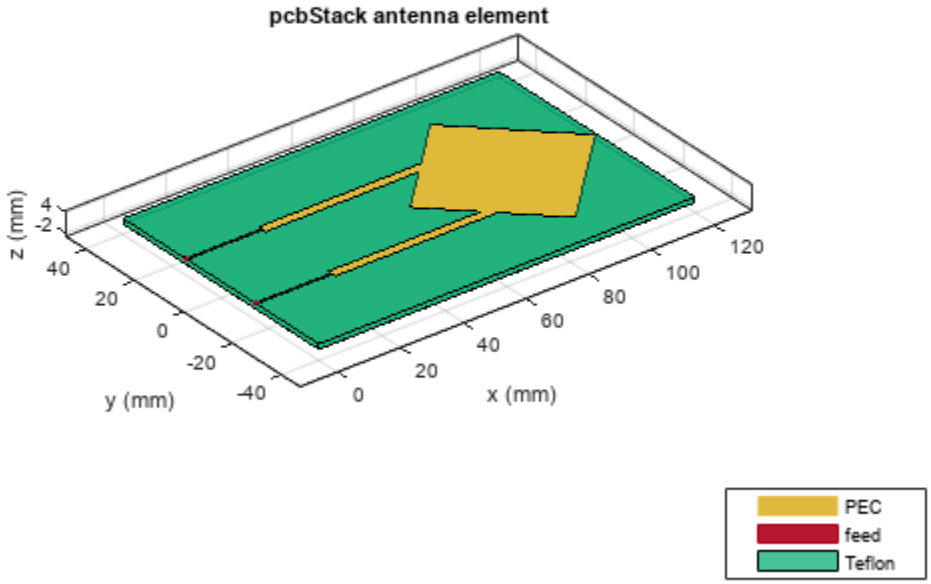

```
antShape  = antShape + Line1+ Line2;
show(antShape);
```



Define the substrate parameters and create a dielectric to use in the PCB stack of the antenna. Create a groundplane using the `antenna.Rectangle` shape.

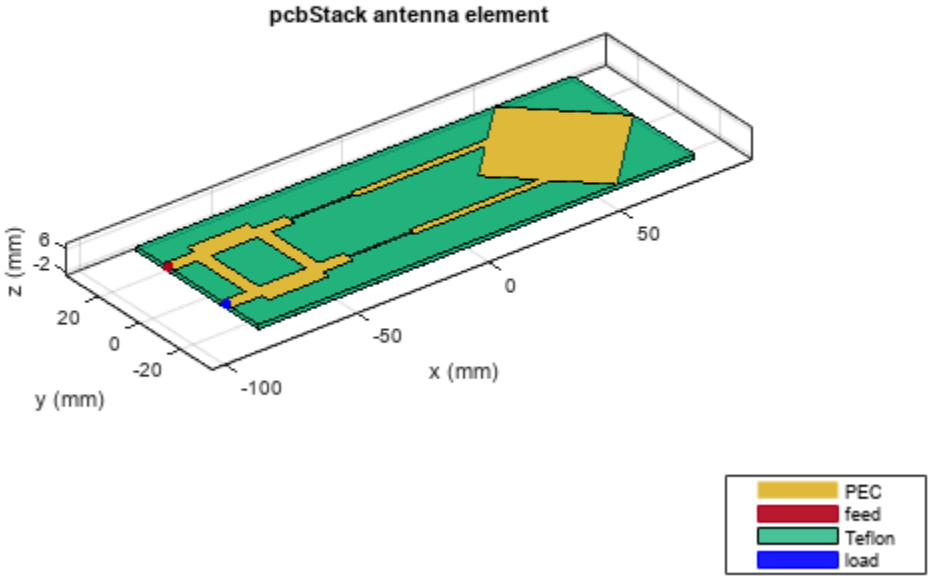
Use the `pcbStack` to create a PCB antenna and assign the dielectric and ground plane to the `Layers` property on `pcbStack`. Assign the `FeedLocations` to the edge of the feedline and set the `BoardThickness` to `Height` on the `pcbStack` and visualize the antenna. The below code performs these operations and creates the PCB antenna.

```
EpsilonR     = coupler.Substrate.EpsilonR;    % Dielectric EpsilonR
Height       = coupler.Height;                % Height of the Substrat
LossTangent  = coupler.Substrate.LossTangent; % Loss Tangent of the Substate

d1  = dielectric('Name',{'Teflon'},'EpsilonR',EpsilonR,'LossTangent',LossTangent,'Thickness',Heig
Gnd = antenna.Rectangle('Length',120e-3,'Width',80e-3,'Center',[120e-3/2,0]);
ant               = pcbStack;
ant.BoardThickness = Height;
ant.Layers        = {antShape,d1,Gnd};
ant.BoardShape    = Gnd;
ant.FeedLocations = [0 portSpacing 1 3;0,-portSpacing 1 3];
figure,show(ant);
```



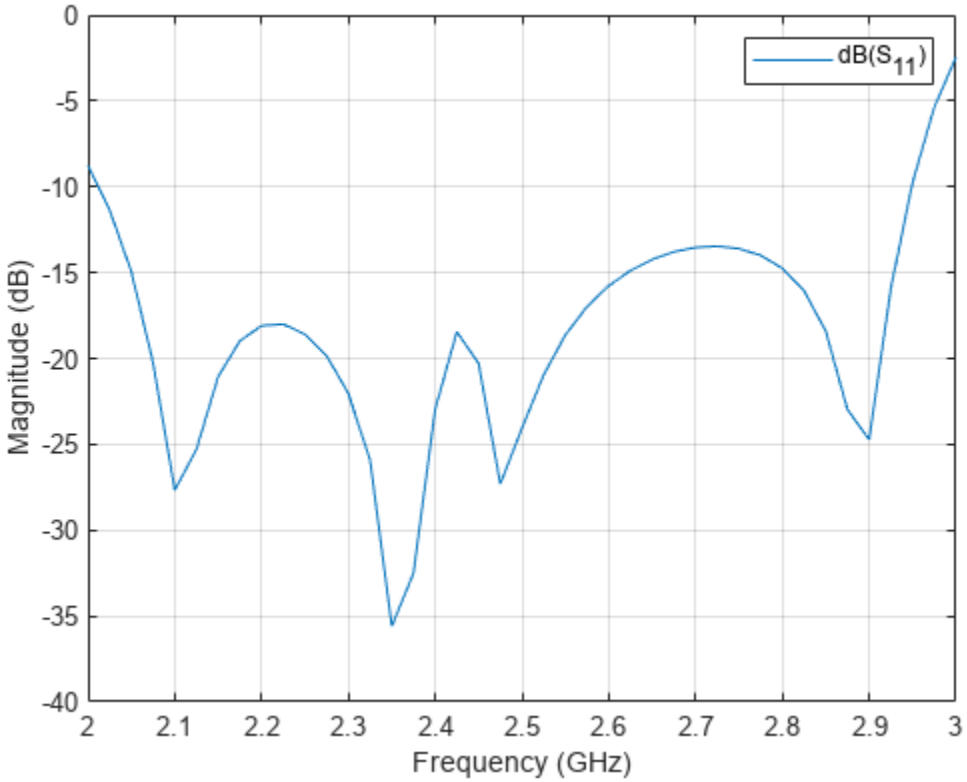
Use the `pcbcascade object` to join the coupler and the patch antenna. To obtain the circular polarization, connect the isolated port on the branchline coupler to a matched load of 50 ohm. The isolated port feed location in `FeedLocations` property of the pcbStack is copied into the `ViaLocations.` The feed location on the isolated port is deleted. The `lumpedElement` object is used to create an impedance of 50 ohms and the location of the Lumped Element is given at the `ViaLocations`. Assign this `lumpedElement` to the `Load` property on the `pcbStack` and visualize the antenna. The below code performs these operations.

```
pcbAntenna                    = pcbcascade(coupler,ant);
pcbAntenna.ViaLocations       = pcbAntenna.FeedLocations(2,:);
pcbAntenna.ViaDiameter        = pcbAntenna.FeedDiameter;
pcbAntenna.FeedLocations(2,:) = [];
r = lumpedElement;
r.Impedance = 50;
r.Location  = [pcbAntenna.ViaLocations(1:2),pcbAntenna.BoardThickness];
pcbAntenna.Load = r;
figure;
show(pcbAntenna);
```



Use the `sparameters` function to calculate the s-parameters of the antenna and plot the result using the `rfplot` function.

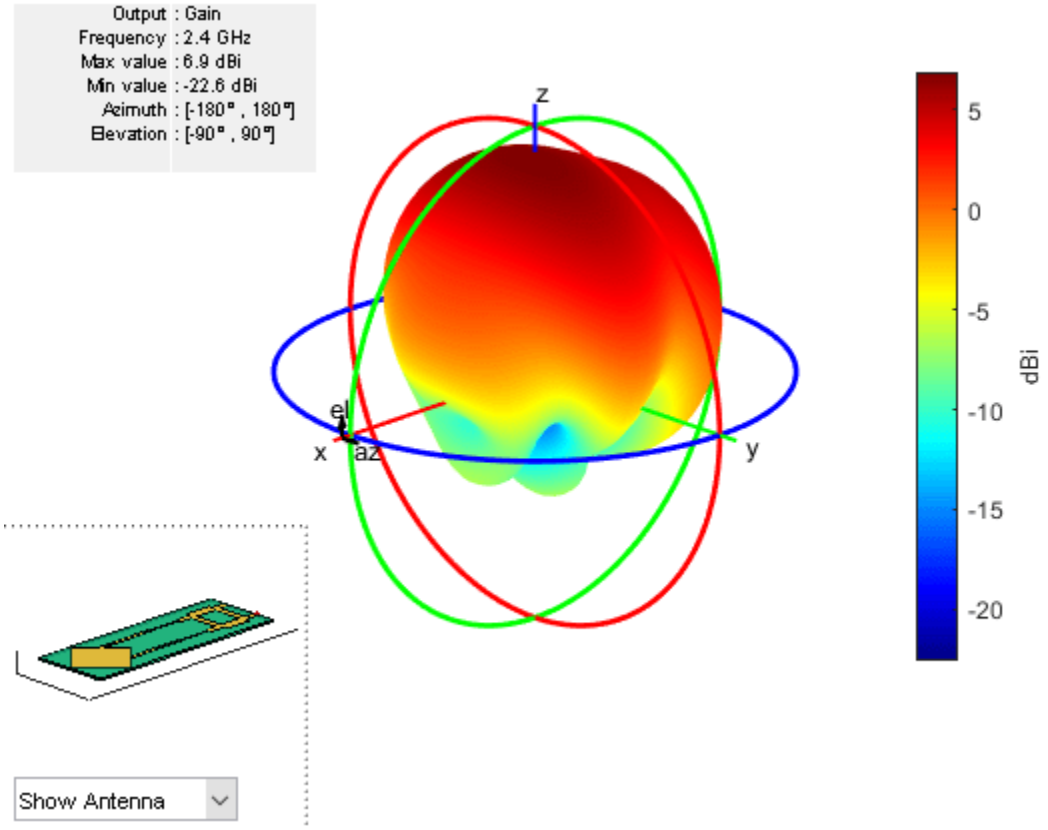
```
spar = sparameters(pcbAntenna,freq);
figure;
rfplot(spar);
```



The result shows that the antenna is a good match from 2 GHz to 3 GHz.

Use the pattern function to plot the 3D radiation pattern of the antenna.

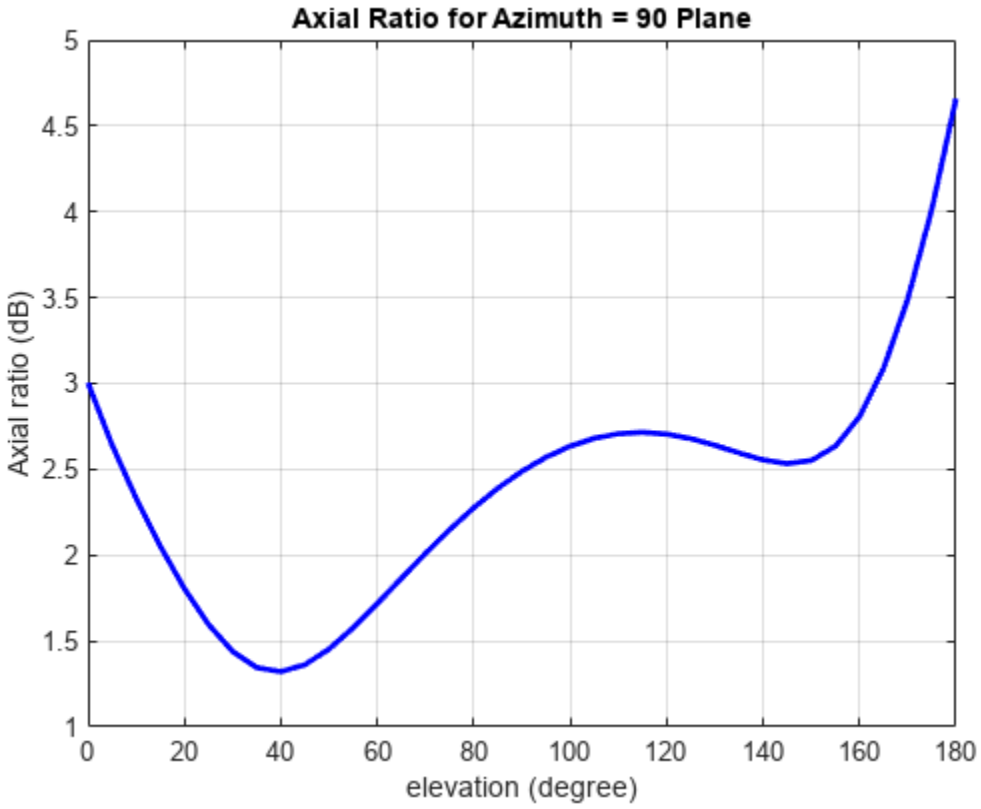
```
figure;
pattern(pcbAntenna,Centerfreq);
```



The antenna has a gain of around 7 dBi at the design frequency.

Use the `axialRatio` function to plot the axial ratio of the antenna at 2.45 GHz. Use the axial ratio to determine the polarization of the antenna. For the circular polarization, the axial ratio must be less than 3 dB.

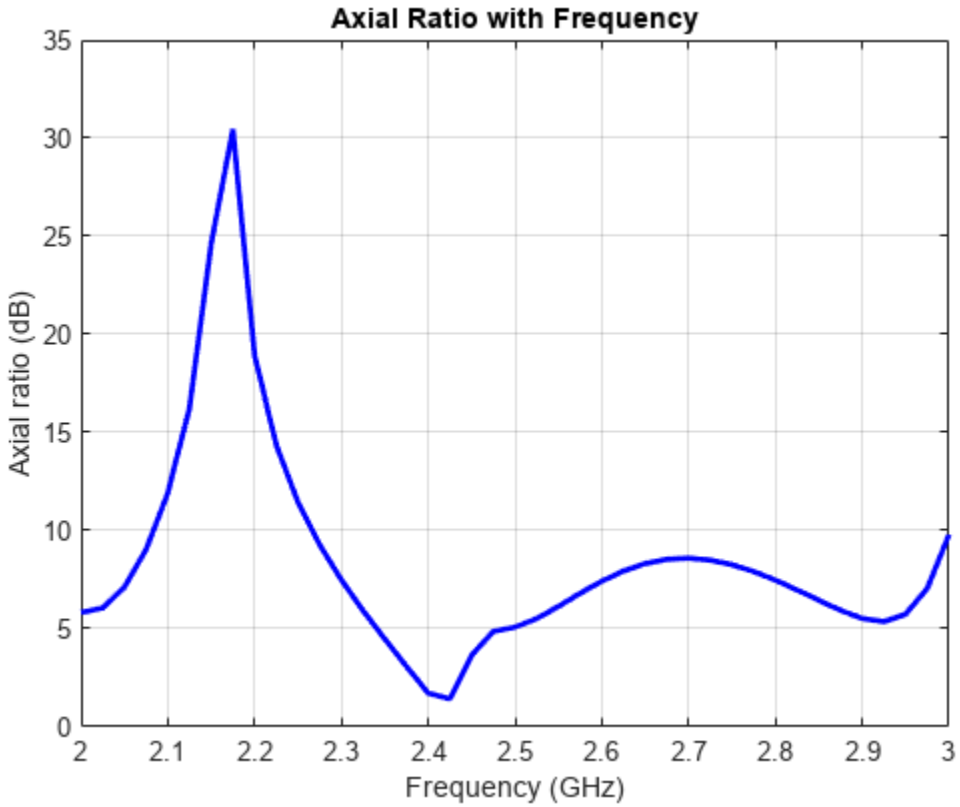
```
figure;
axialRatio(pcbAntenna,Centerfreq,90,0:5:180);
title('Axial Ratio for Azimuth = 90 Plane');
```



The result shows that the axial ratio is below 3 dB for all the elevation angles and hence the antenna exhibits a circular polarization. In the orthogonal plane when azimuth angle is 0 degrees, the circular polarization is obtained between the elevation angles of 60 to 90 degrees.

Use the `axialRatio` function to plot the axial ratio of the antenna with frequency.

```
figure;
axialRatio(pcbAntenna,freq,0,90);
title('Axial Ratio with Frequency');
```



The axial ratio with frequency shows that the antenna exhibits circular polarization around 2.4 GHz.

References

[1] Balanis, C.A. "Antenna Theory. Analysis and Design", p. 860, Wiley, New York, 3rd Edition, 2005.

Create and Analyse Dual Characteristics of Complementary Split Ring Resonators from Gerber Files

This example shows how to create a complementary split ring resonator (CSRR) from Gerber files and subsequently analyze the model. 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 RF-PCB Toolbox™ supports the newer RS-274X format both to generate Gerber files from a PCB model and to create the PCB model from a set of Gerber files.

The CSRRs have been proposed for the synthesis of metamaterials. The basic unit cell that exhibits negative permittivity is imported using the gerber files in the first section, which shows band stop response at 0.9 GHz, and later the left-handed (LH) behavior of the metamaterial is achieved which exhibits pass band at the same frequency by creating the slit on the top layer of the transmission line and the same is analyzed [1].

This example shows how to model the above-described PCB structure from the gerber file.

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 model out of these files, you need layer files that specify the PCB 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 RF-PCB Toolbox supports the Excelling format to specify drill information with file extensions .txt or .drl.

The complimentary metamaterial resonator structure contains three layers on which the top layer is the transmission line in microstrip form and the bottom layer is the CSRR. The CSRR model is created from the gerber file in first section. Later, a gap capacitor is introduced by modifying the top layer of CSRR in second section and further, their behaviors are analyzed in the third section .

Import Top and Bottom Layer from Gerber file

The first step is to import a top-layer and bottom-layer Gerber file from the workspace using the gerberRead function. This will create a PCBReader object. The PCBReader object provides access to the stack up that holds the information of the metal and dielectric layers and also any drill files that describe a PTH via from one metal layer to another if available.

The CSRR model imported using StackUp method contains three layers corresponding to the model of CSSR and returns the model with additional two layers of air dielectric as the first and last layer by default.

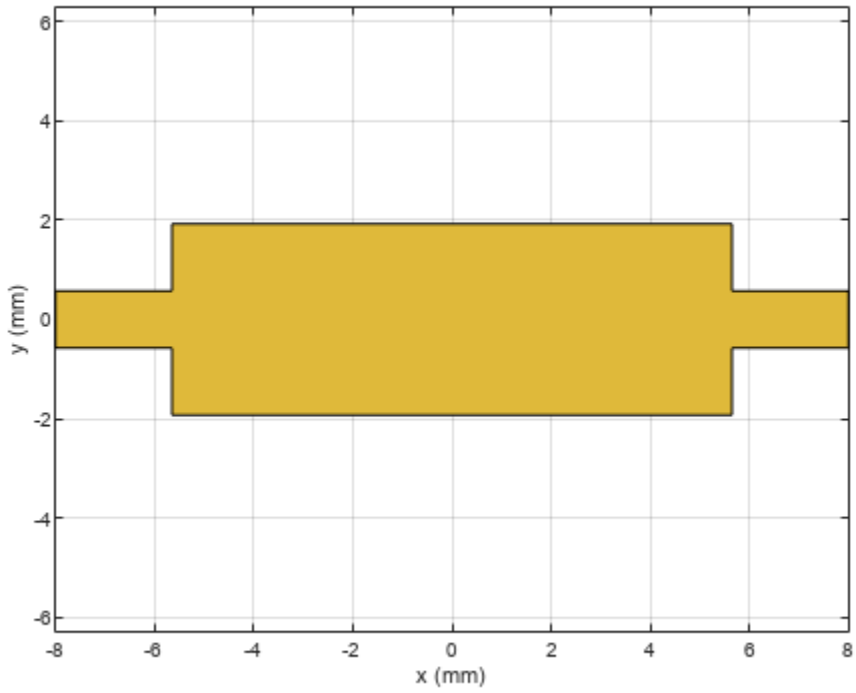
Import the top and bottom layers from the gerber file by passing .gtl and .gbl file as inputs to the function gerberRead and use stack up method to create the layers.

```
P1 = gerberRead('CSRRWOSlit.gtl','CSRRWOSlit.gbl');
P1.StackUp;
```

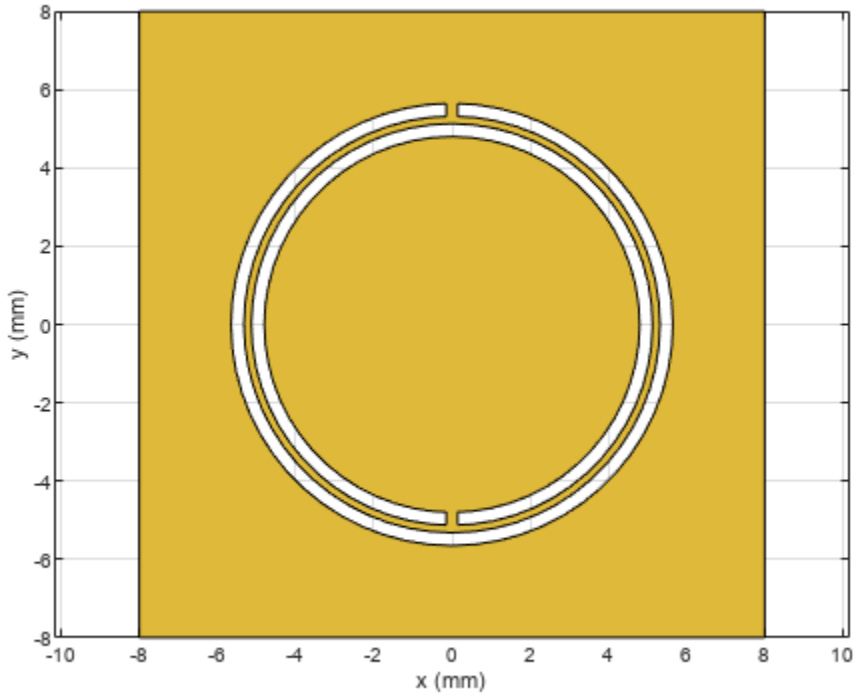
Pass the created PCB reader object as input to pcbComponent and clean up the layer vertices by rounding it off up to some tolerance. Here 6 is picked as the tolerance as the layer dimensions are in mm units. Later, view the top and bottom layer of the structure to measure its feed width and board dimensions.

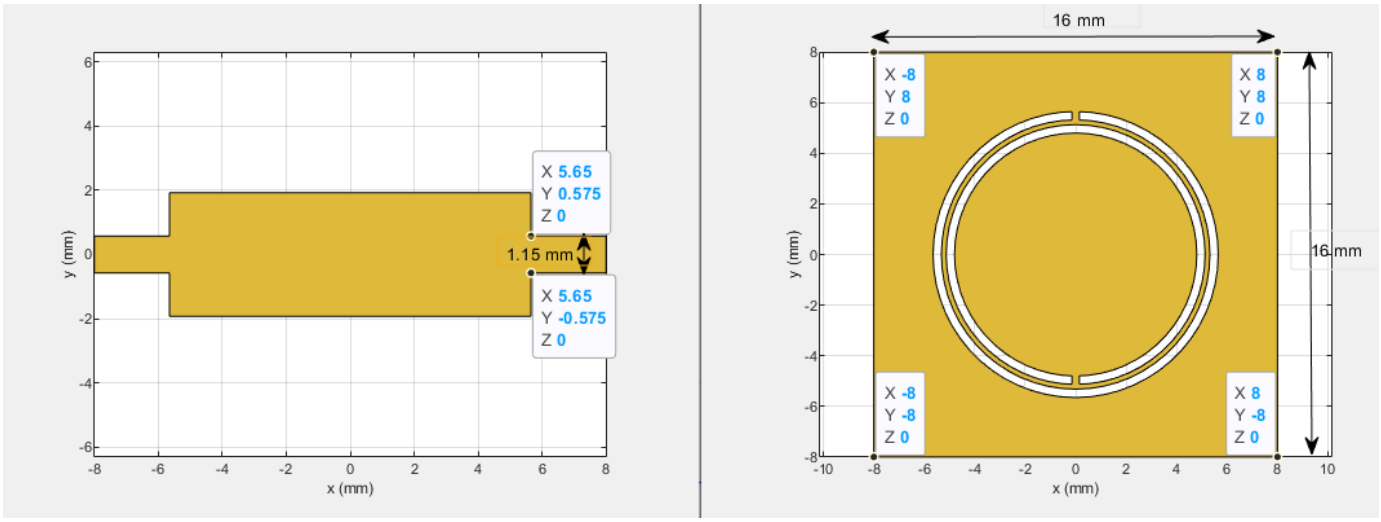
The imported PCBReader object contains five layers, where layer2 and layer4 holds the top and bottom layer of the CSRR structure.

```
pb = pcbComponent(P1);

pb.Layers{2}.Vertices =  round(pb.Layers{2}.Vertices,6);
pb.Layers{4}.Vertices =  round(pb.Layers{4}.Vertices,6);

% Visualize the top and bottom layer of the PCB structure
figure;
show(pb.Layers{2});
```




```
figure;
show(pb.Layers{4});
```

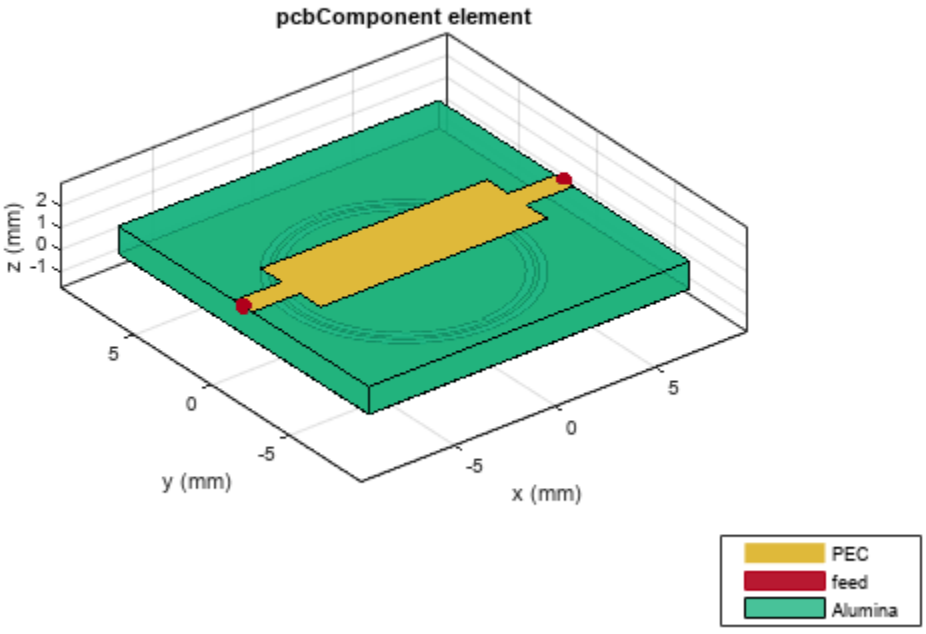




The pointers are placed to measure the dimensions of Feed Width and the board shape as shown above. It is observed that, the Feed Width is 1.15 mm and ground plane length and width are 16 mm each.

Next step is to modify the layers of pcb object. Set the PCB Board dimensions, feed diameter and feed location from the measured values accordingly and visualize the PCB model.

```
%Define substrate
h   = 1.27e-3;
sub = dielectric('Name',{'Alumina'},'EpsilonR',10.2,'LossTangent',0.0023,'Thickness',h);

% Define Board thickness
pb.BoardThickness = h;

% Define pcb Layers
pb.Layers = {pb.Layers{2},sub,pb.Layers{4}};

% Define Board Shape
gndL = 16e-3;
gndW = 16e-3;
gnd  = antenna.Rectangle('Length',gndL,'Width',gndW);
pb.BoardShape = gnd;

% Define Feed Diameter and set Feed location from the measured values
Fw = 1.15e-3;
pb.FeedDiameter = Fw/2;
pb.FeedViaModel = 'strip';
pb.FeedLocations =  [-gndL/2,0,1,3;gndL/2,0,1,3];

% Visualize pcb Component (pb)
figure;
show(pb)
```

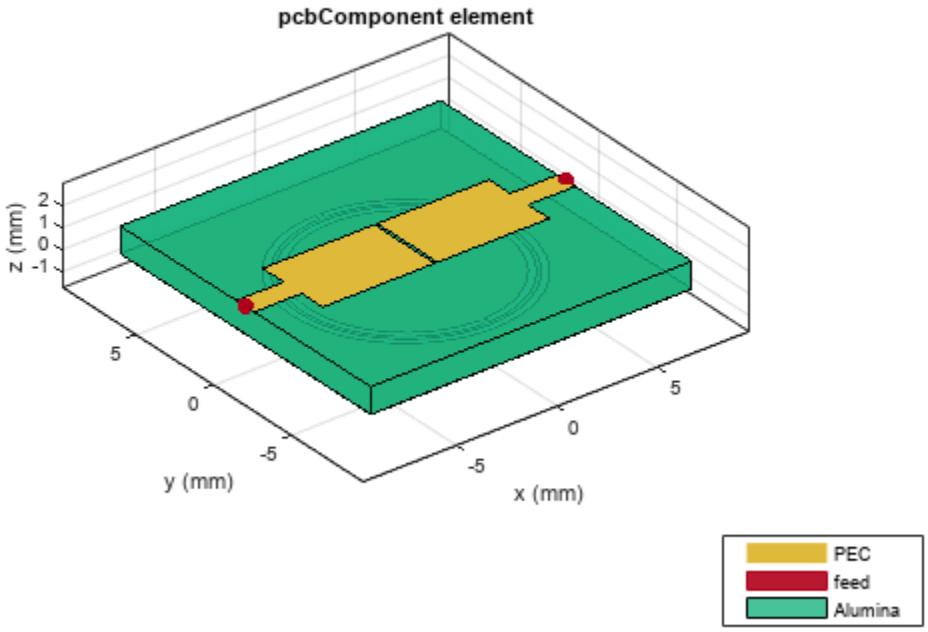


Create Slit Model of CSRR

Create the slit complimentary split ring resonator by introducing the slit in the microstrip line which exhibits gap capacitor due to which the structure can show its characteristics as LH Metamaterial.

Modify the top layer from the pcbComponent object (pb) by introducing the slit in the top layer of CSSR structure and visualize it.

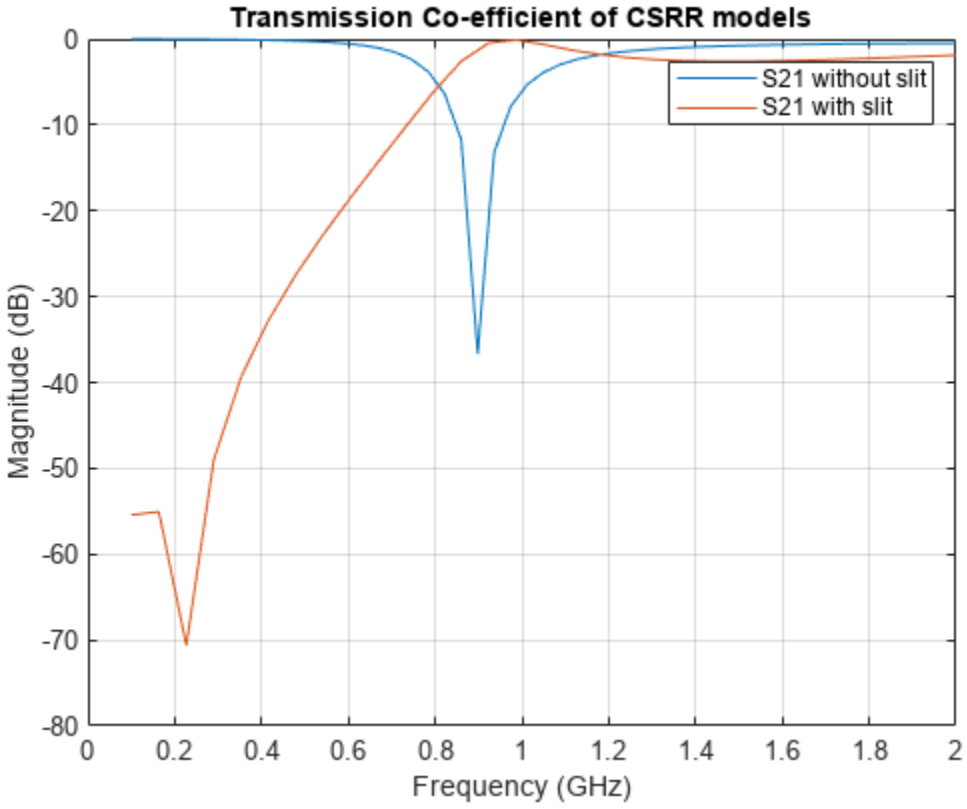
```
pbs = copy(pb);
g = 0.15e-3;
W = 3.85e-3;
% Create the rectangle to introduce the gap
trace = traceRectangular('Length',g,'Width',W,'Center',[0,0]);
toplayer = pbs.Layers{1};
% Substract the rectangle from the top layer
slit  = toplayer-trace;
% Reassign the top layer
pbs.Layers{1} = slit;
% Visualize pcb Component (pbs)
figure;
show(pbs)
```



Analyze Above CSRR Model With and Without Slit

The CSRR model shown without slit (pb) and with slit (pbs) on the top layer is analyzed in the frequency range of 0.1 GHz to 2 GHz using the sparameters method to compute the transmission coefficient and is plotted using the `rfplot` method.

```
% Analyse sparameters of CSRR without slit
spar = sparameters(pb,linspace(0.1e9,2e9,51));
figure;
rfplot(spar,2,1);
% Analyse sparameters of CSRR with slit
spar = sparameters(pbs,linspace(0.1e9,2e9,31));
hold on;
rfplot(spar,2,1)
title('Transmission Co-efficient of CSRR models');
legend('S21 without slit','S21 with slit');
```



It is observed from the above graph that the transmission coefficient S21 for the CSRR metamaterial of negative permittivity lines exhibits the band stop response at resonant frequency of 0.9 GHz, and the Left-handed behavior which is achieved by introducing gap capacitors in microstrip line exhibits the band pass response at the same frequency.

Conclusion

The microstrip lines loaded with CSRRs has been analyzed in two different forms on the same substrate and is observed that they exhibit stop and pass band at same desired frequency.

The `gerberRead` function is used to import the gerber files of CSRR. One can create, a `PCBReader` object, and subsequently use that object to generate the PCB model using a `pcbComponent` object.

Reference

1 Jordi Bonache, Marta Gil, Ignacio Gil, Joan García, *and* Ferran Martín, "On the Electrical Characteristics of Complementary Metamaterial Resonators," *IEEE MICROWAVE AND WIRELESS COMPONENTS LETTERS, VOL. 16, NO. 10, OCTOBER 2006.*

Prototype, Design, and Analysis of SIW based Microstrip Tapered Transmission Line

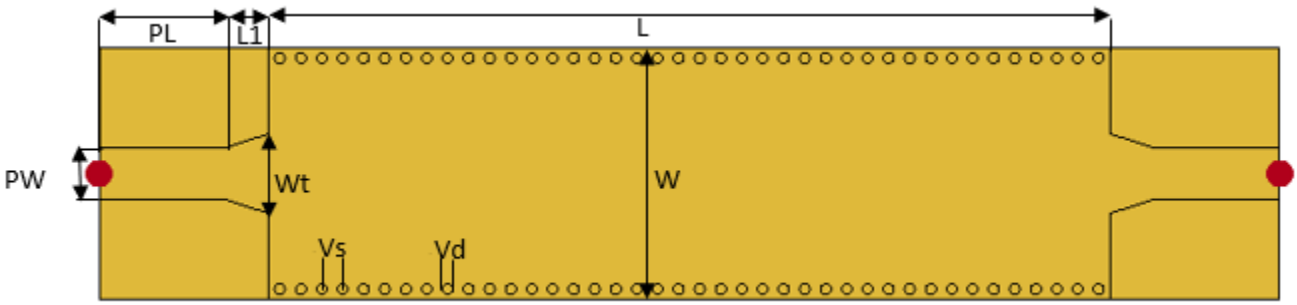
This example shows how to design and analyze a Substrate Integrated Waveguide (SIW) microstrip transmission line using Gerber files.

High performance microwave components can be designed using a novel approach called the Substrate Integrated Waveguide (SIW) technology. This approch combines the advantages of planar technology, such as low fabrication costs, with the low loss inherent to the waveguide solution.

This SIW tapered transmission line model is intended to operate in the microwave frequency range of 10GHz to 15 GHz [1] .

Define Parameters

Below is the schematic diagram of SIW based microstrip tapered transmission line model.




```
L   = 40.016e-3;
Wt  = 3.81e-3;
W   = 12e-3;
PW  = 2.46e-3;
L1  = 2.1e-3;
PL  = 6e-3;
gndW = W;
gndL = L+(2*L1)+(2*PL);
% define substrate
h = 0.8e-3;
sub = dielectric('Name',{'duriod'},'EpsilonR',2.2,'LossTangent',0,'Thickness',h);
```

Create and Analyze SIW Based Tapered Transmission Line

Use the defined parameters to create the microstrip transmisison line. A tapered section is used to match the impedance between a 50 Ω microstrip line in which quasi-TEM and TE10 mode are the dominant mode and their electric current distributions are approximate in the profile of the structure are shown.

Create the model using pcbComponent object and visualize it using show function.

```
% Create the tapered planar microstrip line trace
trace1 = traceRectangular('Length',L,'Width',W);
```

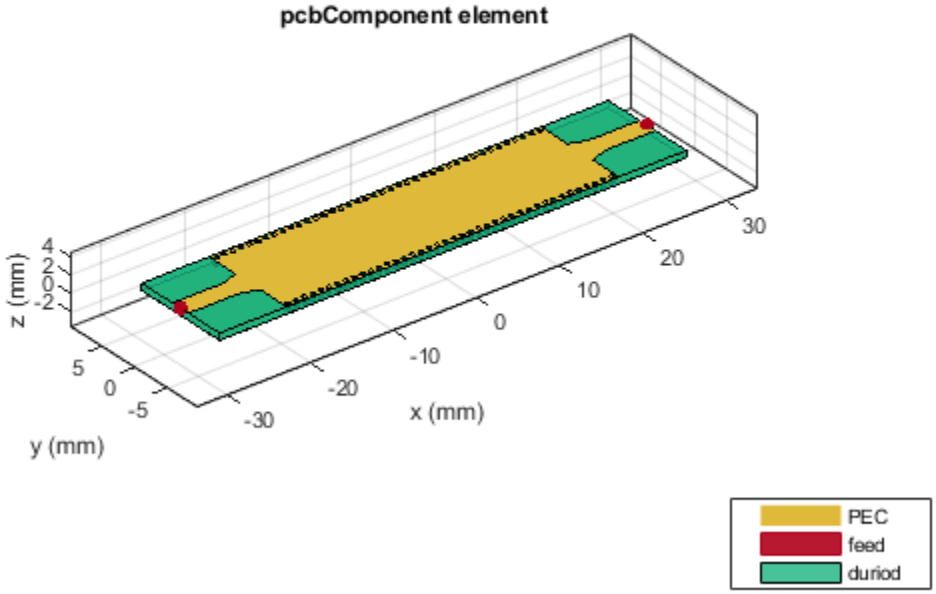
```
v = [-L/2, Wt/2, 0; -(L/2+L1), PW/2, 0; -(L/2+L1),-PW/2,0;-L/2, -Wt/2, 0];
trace2 = antenna.Polygon('Vertices',v);
trace3 = antenna.Polygon('Vertices',-v);
Port1  = traceRectangular('Length',PL,'Width',PW,'Center',[-gndL/2+PL/2+1e-12,0]);
Port2  = traceRectangular('Length',PL,'Width',PW,'Center',[gndL/2-PL/2-1e-12,0]);
Mtrace = trace1+trace2+trace3+Port1+Port2;
```

In order to build the PCB model, use pcbComponent object. Mtrace created above will be the top layer on the rectangular SIW.

```
p = pcbComponent;
p.BoardThickness = h;
gnd = traceRectangular('Length',gndL,'Width',gndW,'Center',[0,0]);
p.BoardShape = gnd;
p.Layers = {Mtrace,sub,gnd};
```

Define feed and via locations. The FeedLocations and ViaLocations are the property with an array of four elements, in which the 1st and 2nd elements define the x and y co-ordinates, 3rd and 4th elements define the layers of connection.

```
% Define Feed location
p.FeedDiameter = PW/2;
p.FeedLocations = [-gndL/2,0,1,3;gndL/2,0,1,3];
p.ViaDiameter = 1e-3/2;
% Define Via location
offset = 0.5e-3;
viax = -L/2+offset;
viay = -W/2+offset;
for i=1:40
    viaSp = 0.5e-3;
    ViapointX(i) = viax + (i-1)*(viaSp)+(i-1)*(p.ViaDiameter);
    ViapointY(i) = viay;
    layer1(i) = 1;
    layer2(i) = 3;
    Viapoint1     = [ViapointX' ViapointY' layer1' layer2'];
end
for i=1:40
    viaSp = 0.5e-3;
    ViapointX1(i) = viax + (i-1)*(viaSp)+(i-1)*(p.ViaDiameter);
    ViapointY1(i) = -viay;
    layer11(i) = 1;
    layer21(i) = 3;
    Viapoint2     = [ViapointX1' ViapointY1' layer11' layer21'];
end
p.ViaLocations = [Viapoint1;Viapoint2];
figure; show(p);
```



Use the `sparameters` function to compute the s-parameters in the frequency range of 10 - 15 GHz and plot it.

```
freq = linspace(10e9,15e9,31);
spar = sparameters(p,freq);
figure; rfplot(spar,[2 1],1)
axis tight
```

It is obseved from the plot that the reflection coefficient S11 is less than -15 dB over the frequency band and the transmission coefficient S21 is around -0.3 dB to -0.7 dB across the entire band.

Prototype the model using Gerber files

Gerber files are open ASCII vector format files that contain information on each physical board layer of the PCB design. Circuit board objects, like copper traces, vias, pads, solder mask and silkscreen images, are all represented by series of coordinates. These files are used by PCB manufacturers to translate the details of the design into the physical properties of the PCB.

To generate these files, two additional pieces of information is required apart from the PCB design. 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 RF PCB 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.

Generation of Gerber files

Gerber files contains the collection of files that describe a Printed Circuit Board (PCB) . Each file describes a specific aspect of the PCB design. For example, .gtl and .gbl files contains the information about the metal regions corresponding to the signal and ground that are filled with copper traces. Similarly, .gts and .gbs files holds the information about the solder mask, which is applied to protect and insulate the metal regions. 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 and render the design.

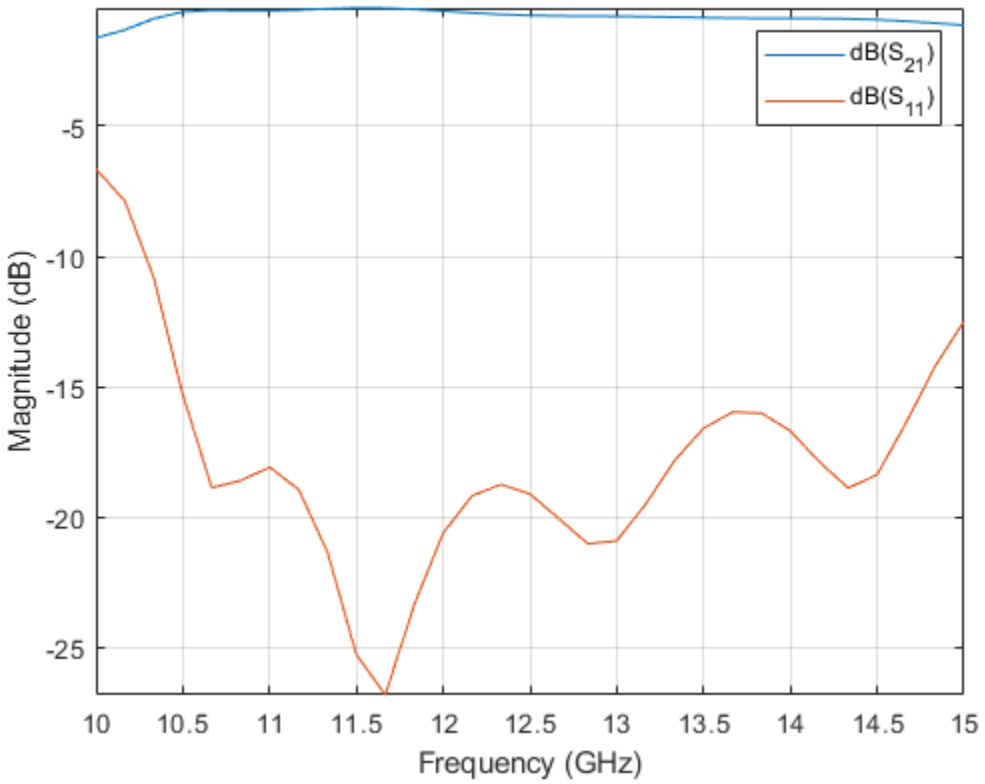
Online viewer for Gerber files

The PCB services catalog contains different manufacturing servies type which can accesed using the object `PCBServices`. In this example, MayhewWriter is used to configure the Gerber file generation process and view them using the Mayhewlabs free online 3D Gerber viewer. Select an SMA edge connector from the available catalog and modify it accordingly. The PCB model, PCB services, and the RF PCB connector are used to create a PCBWriter.

```
 s = PCBServices.MayhewWriter;
s.Filename = 'SIWMicrostripLine';
%
PW = PCBWriter(p,s);
PW.UseDefaultConnector = 0;
```

Using the above created PCBWriter, execute the gerberWrite command to generate the Gerber files for the SIW Model.

```
gerberWrite(PW)
```

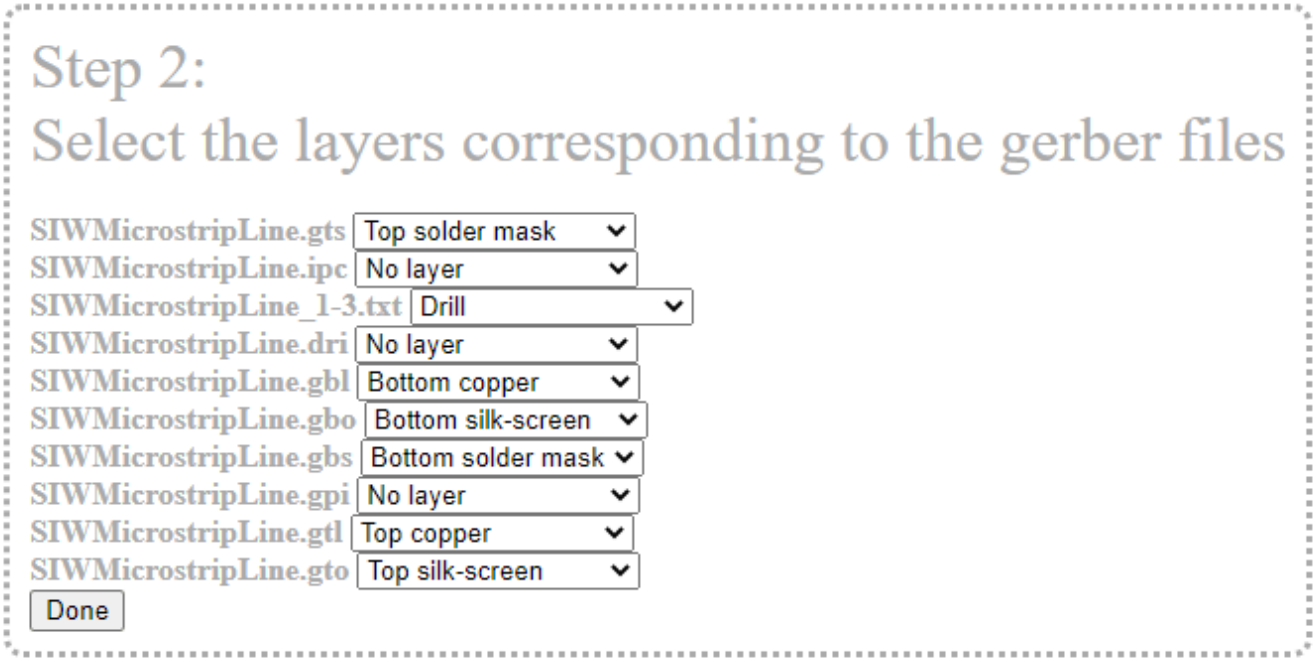


The collection of files are generated and placed in a folder with the same name as assigned in the Filename property of the particular PCB services. This folder is located in the current working directory. The files in the folder are shown in the image below.

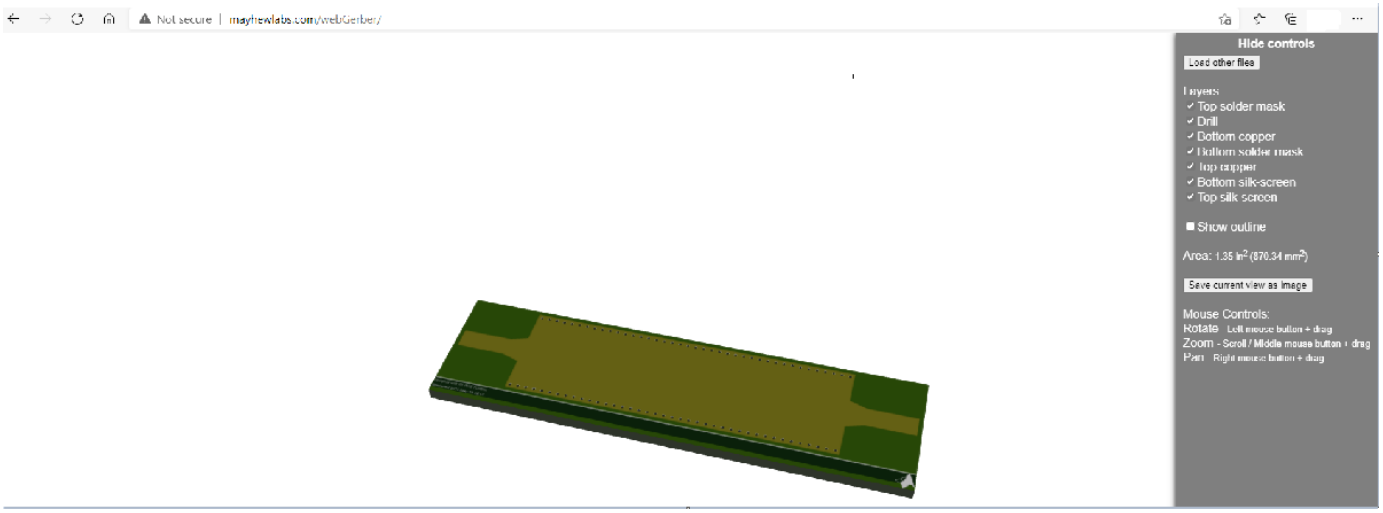
Name	Date modified	Type	Size
SIWMicrostripLine.dri	5/10/2021 4:12 PM	DRI File	1 KB
SIWMicrostripLine.gbl	5/10/2021 4:12 PM	GBL File	1 KB
SIWMicrostripLine.gbo	5/10/2021 4:12 PM	GBO File	138 KB
SIWMicrostripLine.gbs	5/10/2021 4:12 PM	GBS File	1 KB
SIWMicrostripLine.gpi	5/10/2021 4:12 PM	GPI File	1 KB
SIWMicrostripLine.gtl	5/10/2021 4:12 PM	GTL File	1 KB
SIWMicrostripLine.gto	5/10/2021 4:12 PM	GTO File	301 KB
SIWMicrostripLine.gts	5/10/2021 4:12 PM	GTS File	1 KB
SIWMicrostripLine.ipc	5/10/2021 4:12 PM	IPC File	1 KB
SIWMicrostripLine_1-3	5/10/2021 4:12 PM	Text Document	2 KB

If the file name is not specified then the folder name and the files in the folder are named as "untitled".

Running the above example automatically opens up the Mayhew Labs PCB manufacturing service in the default internet browser. Further to view the 3D PCB model drag and drop the gerber files as shown below.



Click on Done to to view the proposed PCB design as shown below.



Reference

1 Bouchra Rahal,*et.al*,"*SUBSTRATE INTEGRATED WAVEGUIDE POWER DIVIDER, CIRCULATOR AND COUPLER IN [10-15] GHZ BAND*", *International Journal of Information Sciences and Techniques,* DOI:10.5121/ijist.2014.4201 , March, 2015.

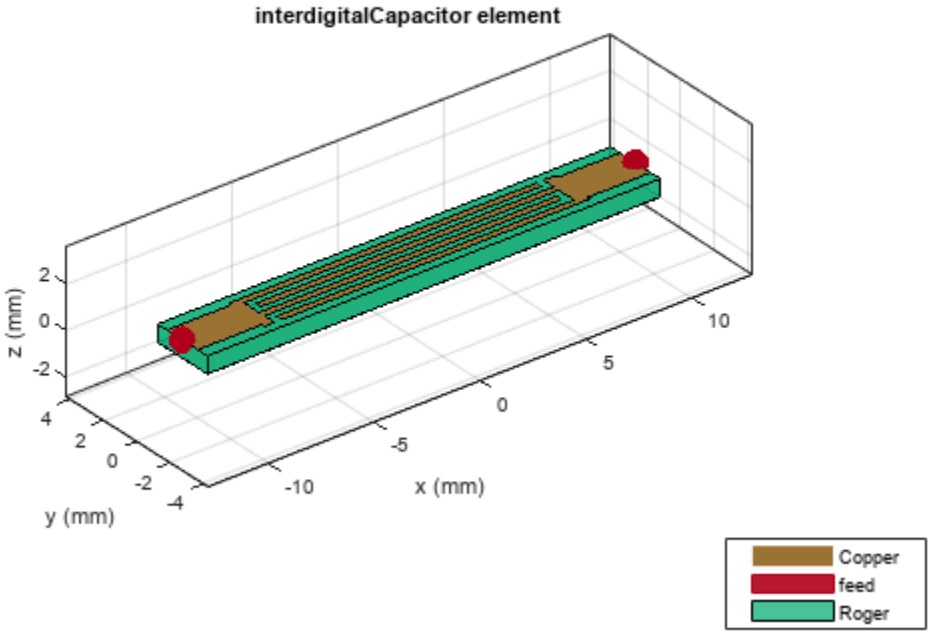
Model and Analyze Microstrip Interdigital Capacitor as Bandpass filter

The interdigital capacitor (IDC) is a multi-finger periodic structure and is frequently used in the microstrip microwave integrated circuits for RF or microwave development. The interdigital capacitors use the capacitance that occurs across a narrow gap between conducting fingers. They are optimized based on applications.

This example shows the analysis of interdigital capacitor that shows band pass response in the frequency range of 3.5 GHz to 3.9 GHz. The capacitance is calculated in the desired range with and without de-embedding the effect of microstrip feeder line.

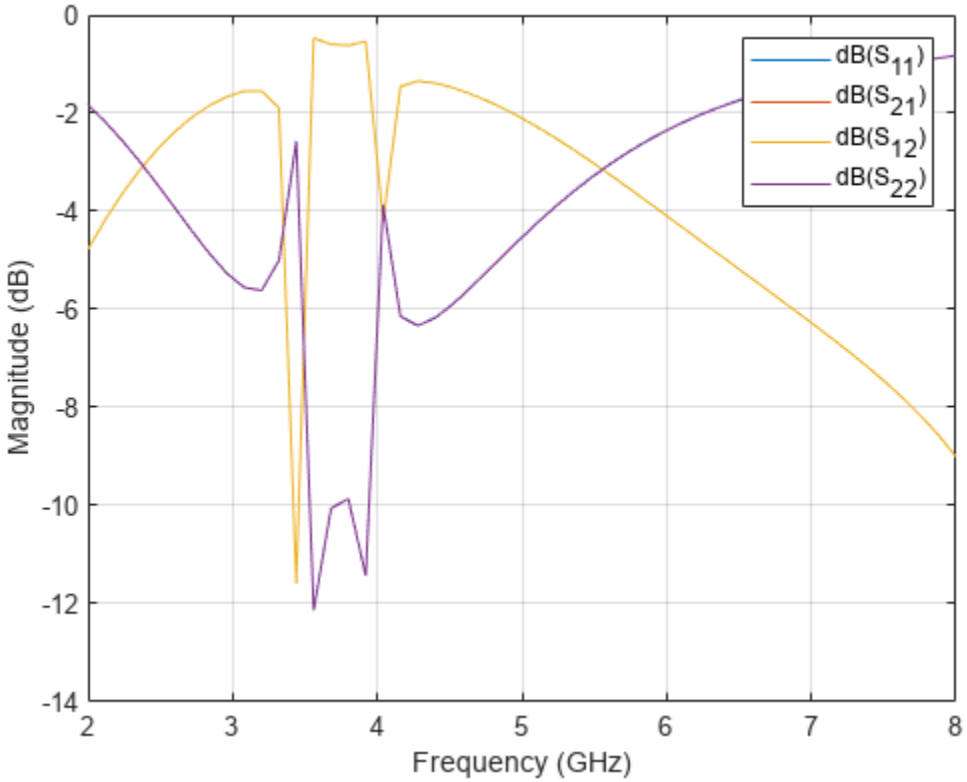
Create an interdigital capacitor and visualize it.

```
obj = interdigitalCapacitor;
figure;
show(obj);
```



Use the `sparameters` function to compute the s-parameters and plot it using the `rfplot` function from RF Toolbox to understand the behavior of the designed capacitor.

```
freq = linspace(2e9,8e9,51);
spar = sparameters(obj,freq);
figure;
rfplot(spar);
```



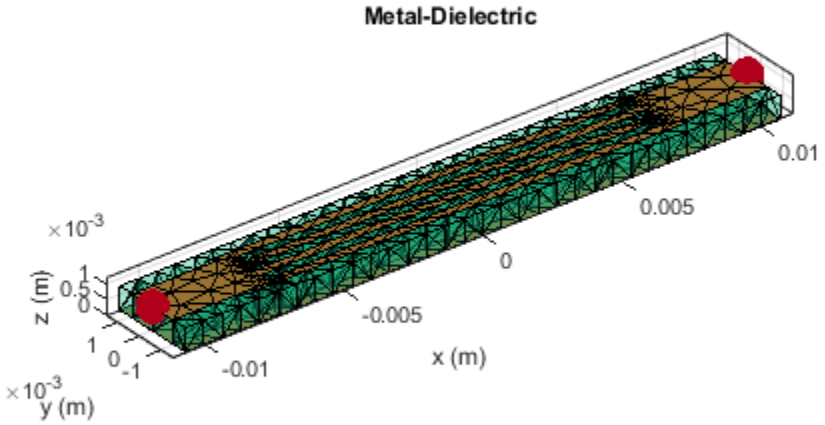
The s-parameters plot shows that the insertion loss is minimal in the frequency range of 3.5 GHz to 3.9 GHz which exhibits band pass like behavior. The capacitor exhibits low attenuation in this frequency range.

These kinds of capacitor are used in RF couple or DC blocks where the capacitor should have minimal attenuation and reflection when the signal goes through them.

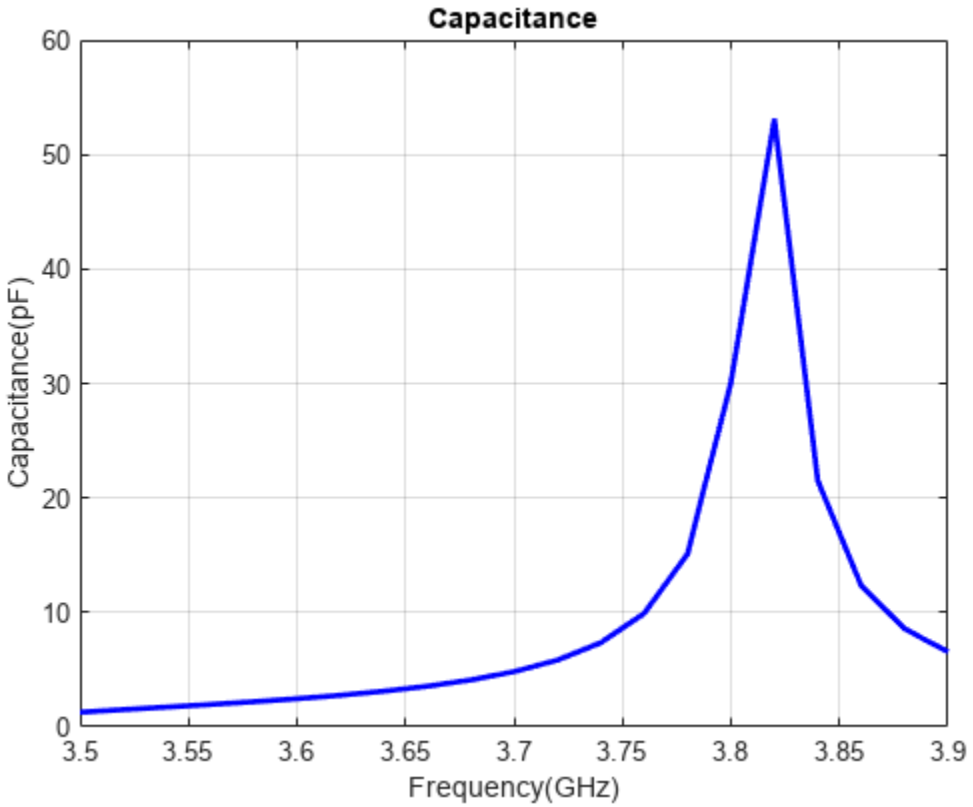
Analyze the measure of capacitor without de-embedding the feeder line effect using the `capacitance` function from 3.5 GHz to 3.9 GHz. Use the `mesh` function to manually mesh the structure.

```
figure;
mesh(obj,'MaxEdgeLength',0.0017)
```

```
NumTriangles: 778
NumTetrahedra: 1584
NumBasis:
MaxEdgeLength: 0.0017
MeshMode: manual
```

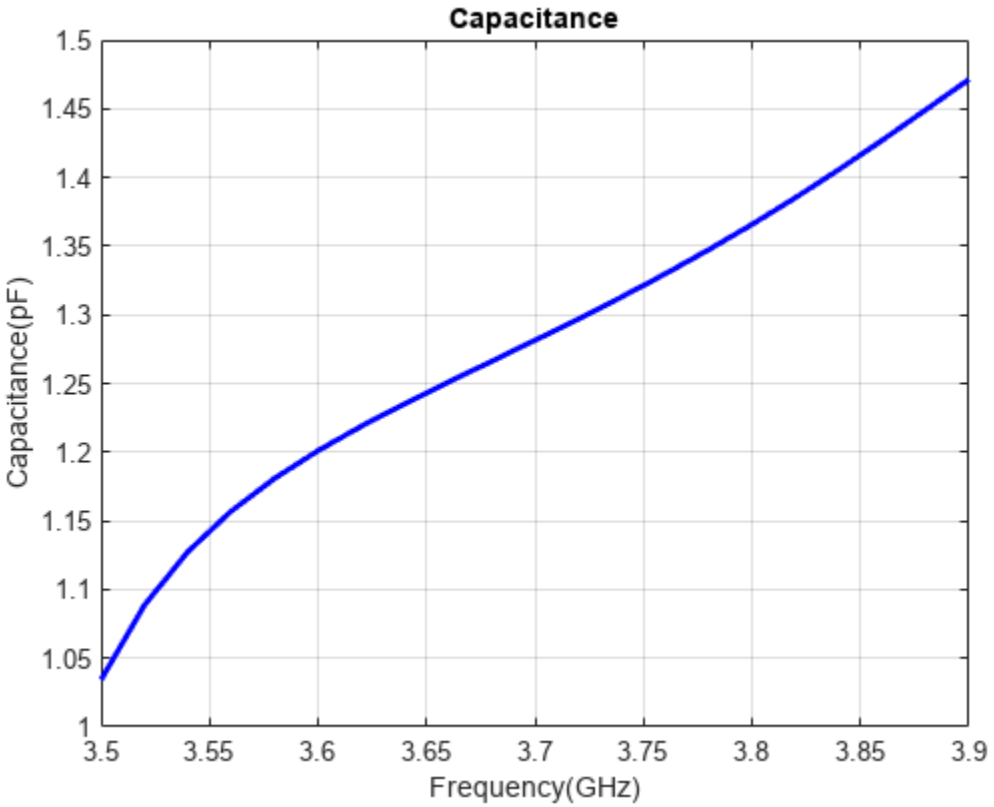



```
freq = linspace(3.5e9,3.9e9,21);
figure;
capacitance(obj,freq,'DeEmbed',0)
```



It is observed that there is a sudden spurious spike at 3.82 GHz which is due to the effect of the feeder line. Further to eliminate that effect the `DeEmbed` option in the capacitance method has to be enabled.

```
figure;
capacitance(obj,freq,'DeEmbed',1)
```



Hence, by de-embeddeding the effect of feeder line it subtracts the additional phase that is added due to the feed line. The capacitance of the capacitor is around 1 pF to 1.5 pF in the frequency range of 3.5 to 3.9 GHz.

The capacitance of the capacitor can be analyzed over the desired frequency range using different options such as de-embed the effect of the feeder line and include the parasitic effect by enabling the options in capacitance method.

Reference

[1] Dib, Nihad, Qiu Zhangb, and Ulrich Rohde. “New Cad Model of the Microstrip Interdigital Capacitor.” *Active and Passive Electronic Components* 27, no. 4 (2004): 237–45. https://doi.org/10.1080/08827510310001648870.

Introduction to Passive Planar Spiral Inductors

This example shows you how to design, visualize, and analyze different types of spiral inductors.

The modern wireless communication system requires low cost, small size, and higher efficiency circuits design. The circuit need considerable amount of passive (inductor and capacitor) components for designing RF components like power amplifier, oscillators, microwave switches, combiners, and splitters circuits.

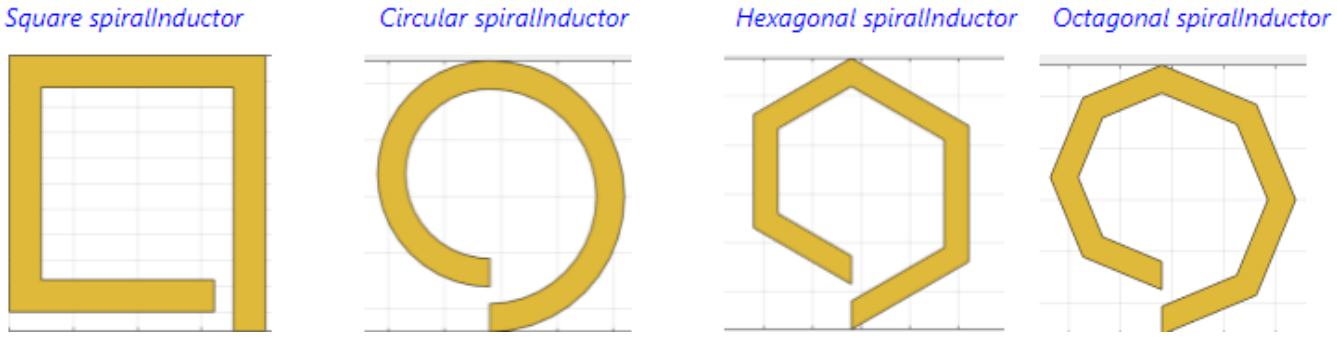
In radio frequency integrated circuits (RFIC) the spiral inductors dominates overall circuit performance and it is frequently used as a passive component in modern RFIC’s design technology.

Here the passive planar spiral inductor is modeled and analysed. This example shows the analyses of different types of spiral inductors.

Model Spiral Inductor

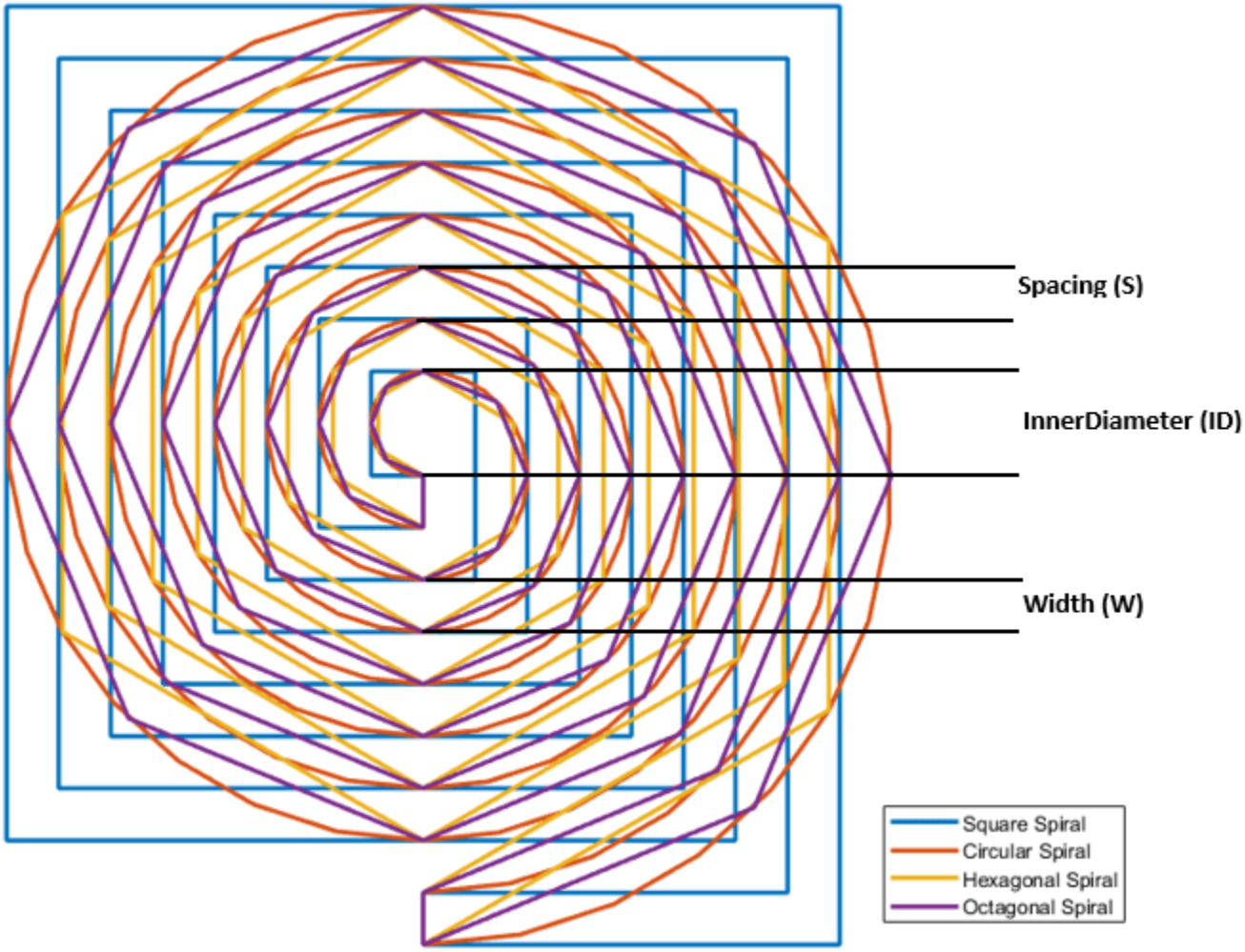
The passive spiral inductor can be realized in different structure like square, circular, hexagonal, and octagonal with multiturns. The multi-turn inductors are used for higher inductance value design.

The spiral Inductor is a two port multiturn planar inductor with multiple dielectric layers. A turn in the spiral inductor is the length of a complete 360-degree revolution.



The width and spacing of the strips in the spiral inductor are uniform throughout. You can feed the inductor in such a way that the inner end is routed out from the layer below the inductor by a via hole or the input port and the outer end or output port is extended to the end of the board at the same layer .

The figure shown below consists of four spiral inductors along with their physical parameters to model spiral inductors



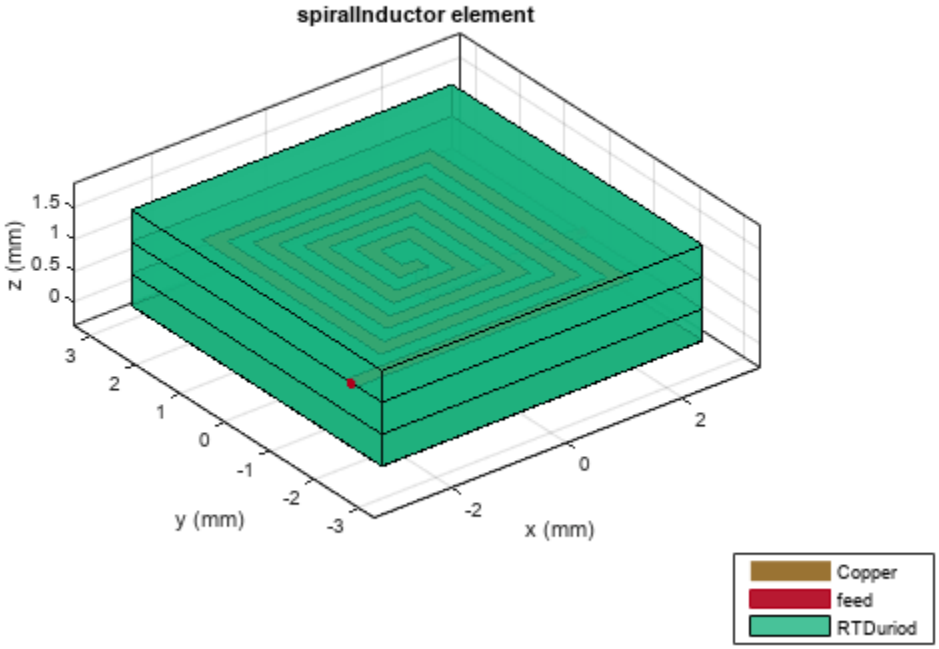
Planar Spiral Inductor in Embedded Microstrip Form with Homogeneous Substrate

Create and analyze a multiturn spiral inductor with different configurations like square, circular, hexagonal, and octagonal by changing the property `SpiralShape`. Calculate the measure of inductance at 600 MHz.

Square Spiral Inductor

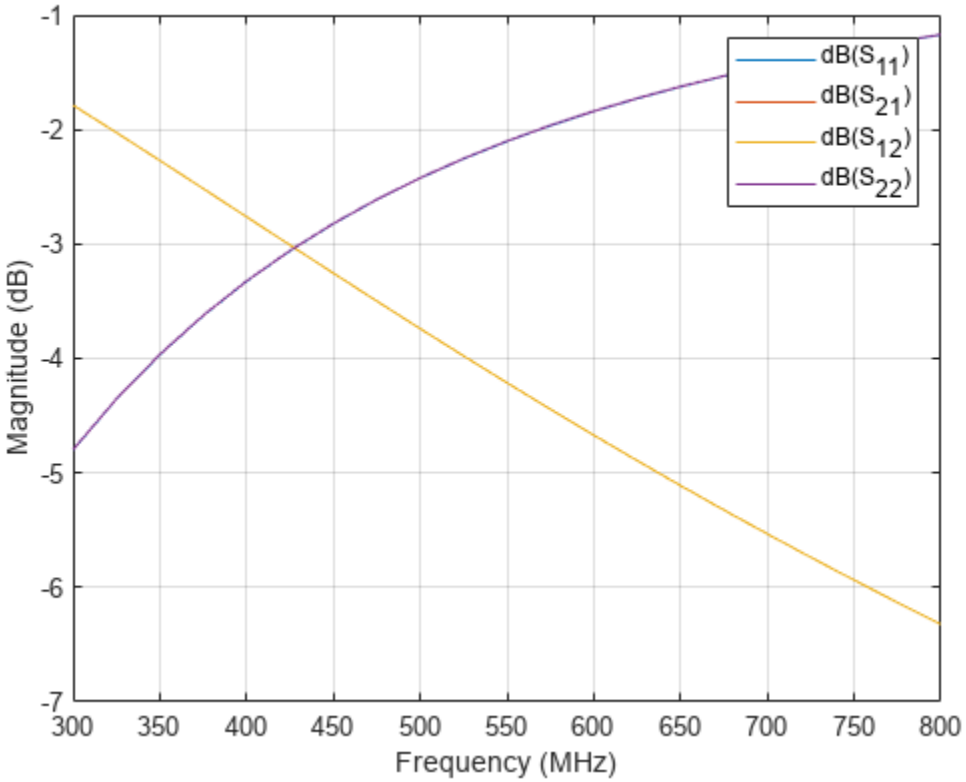
Create and analyze a multiturn square spiral inductor.

```
Ind = spiralInductor('SpiralShape','square');
figure; show(Ind);
```



Use the `sparameters` function to compute the s-parameters and plot it using the `rfplot function`.

```
spar = sparameters(Ind,linspace(300e6,800e6,21));
figure;
rfplot(spar);
```



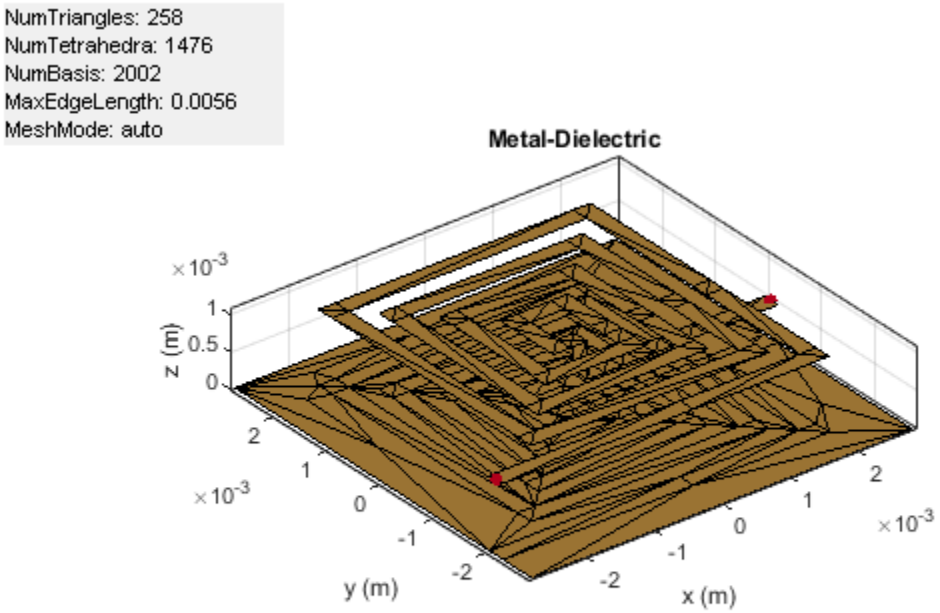
The s-parameters plot shows that after 430 MHz the S11/S22 is monotonically increasing towards 0 dB and S12/S21 is decreasing accordingly. This plot tells that the energy is stored in inductor and not radiated. The behavior of S11 and S21 satisfies the law of conservation of energy.

Use the inductance function to plot the measure of inductance of the square spiral inductor at 600 MHz and view the mesh.

```
Ls = inductance(Ind,600e6)
```

```
Ls = 4.3781e-08
```

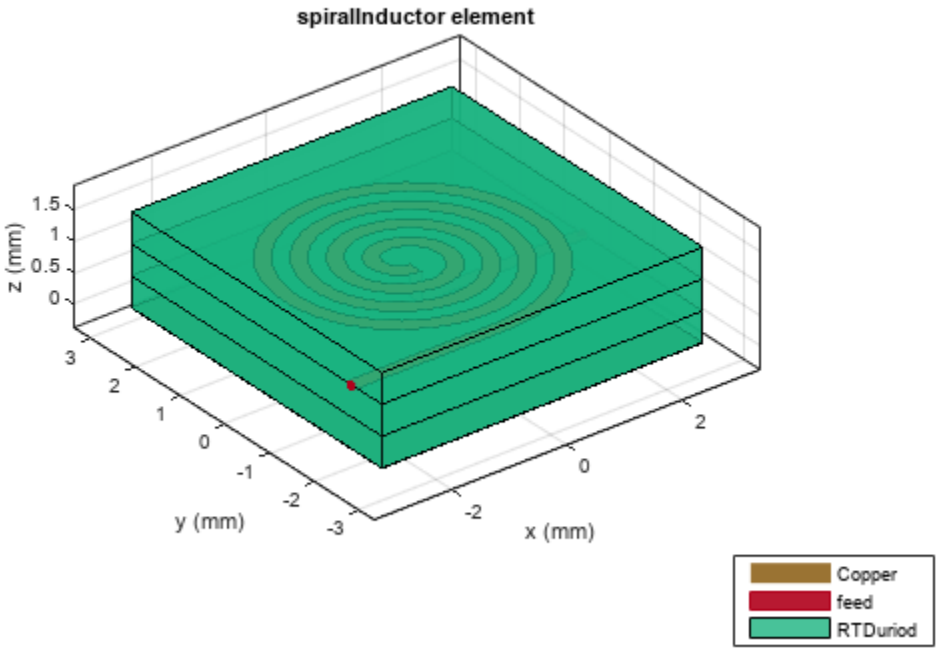
```
figure; mesh(Ind,'View','Metal');
```



Circular Spiral Inductor

Create and analyze multiturn circular spiral inductor.

```
Ind = spiralInductor('SpiralShape','circle');
figure; show(Ind);
```



Use the inductance function to calculate the measure of inductance of the circular spiral inductor at 600 MHz.

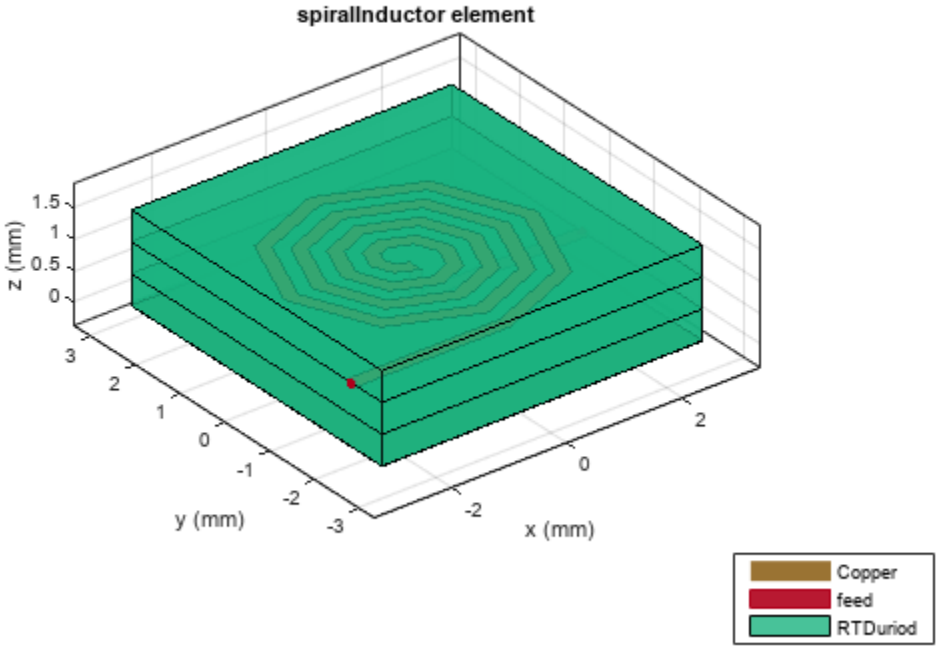
```
Lc = inductance(Ind,600e6)
```

```
Lc = 3.8092e-08
```

Octagonal Spiral Inductor

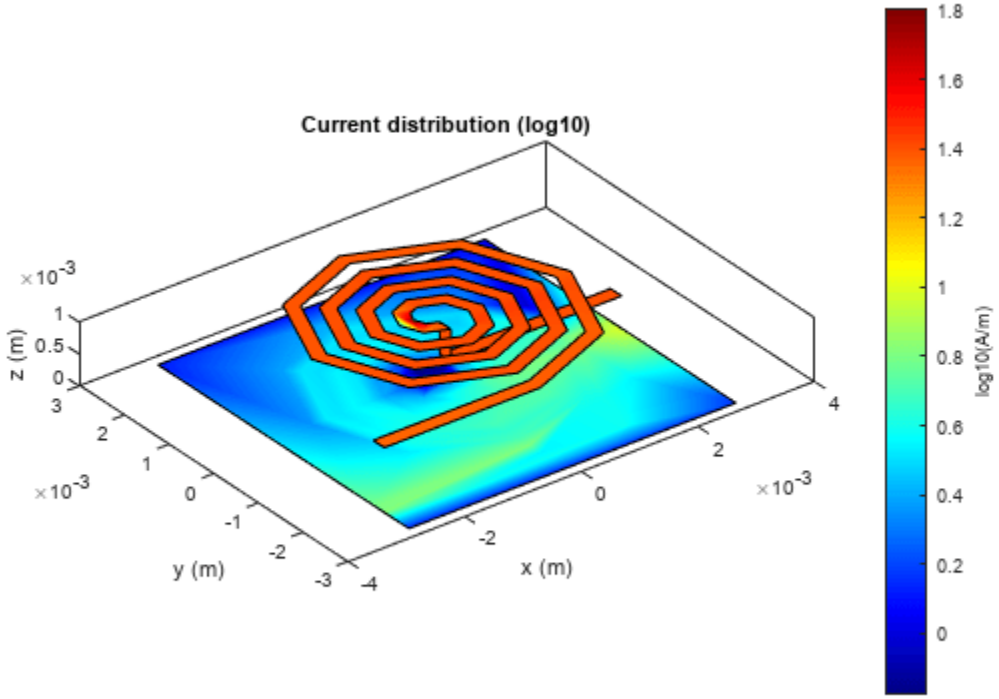
Create and analyze a multiturn octagonal spiral inductor.

```
Ind = spiralInductor('SpiralShape','Octagon');
figure; show(Ind);
```



Use the current function to plot the current distribution on the surface of the octagonal spiral Inductor.

```
figure;
current(Ind,600e6,'scale','log10');
```



Use the inductance function to plot the measure of inductance of the octagonal spiral inductor at 600 MHz.

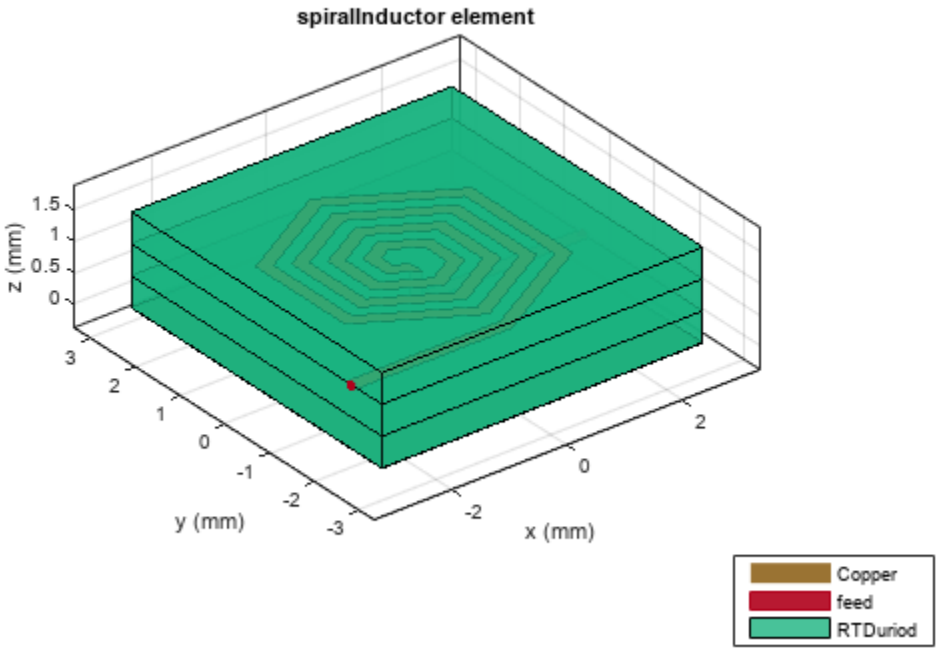
```
Lo = inductance(Ind,600e6)
```

```
Lo = 3.7661e-08
```

Hexagonal spiral Inductor

Create and analyze multiturn hexagonal spiral inductor.

```
Ind = spiralInductor('SpiralShape','Hexagon');
figure; show(Ind);
```



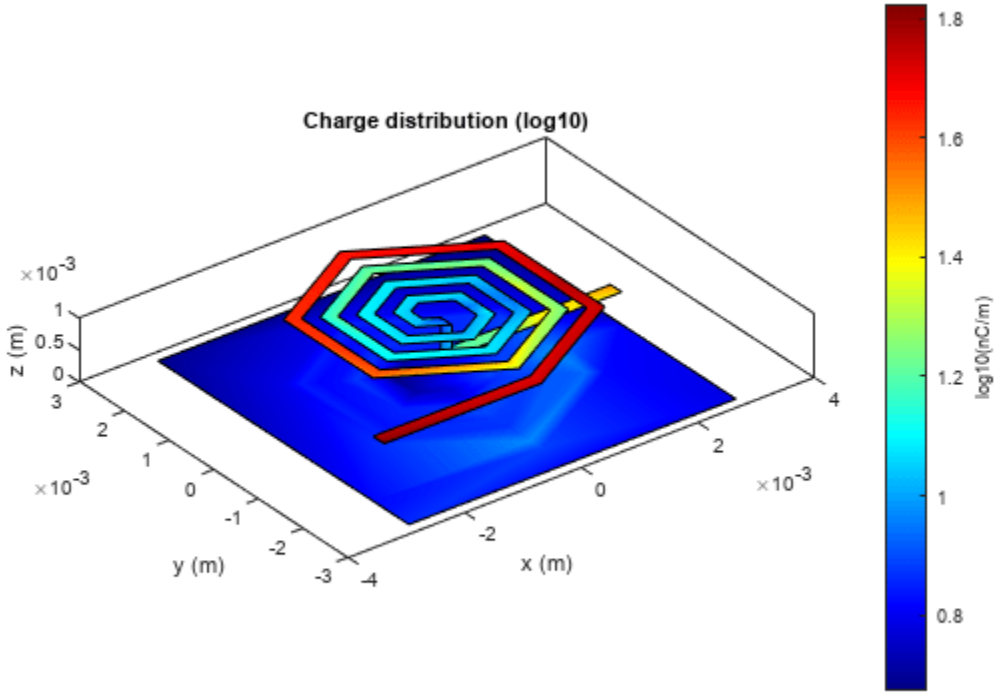
Use the inductance function to calculate the measure of inductance of the hexagonal spiral inductor at 600 MHz.

```
Lh = inductance(Ind,600e6)
```

```
Lh = 3.6937e-08
```

Use the charge function to plot the charge distribution on the surface of the octagonal spiral Inductor.

```
figure;
charge(Ind,600e6,'scale','log10');
```



The Magnitude of inductance for the different configurations of the four turn spiral inductors are shown in the table below. It is observed that the inductance of the square spiral inductor is more when compared to the rest of the spiral shapes.

Sl.no	Spiral Shape	Inductance (nH)
1	Square	43.11
2	Circle	38.29
3	Octagon	37.67
4	Hexagon	36.28

The magnitude of the inductance increases as the length of the inductor and area increases. It is evident from the figures shown for the four configurations of spiral inductors, that the inductance of spiral inductors increases with the increase in trace length and area of the inductors such that: Square> Circular>Octagonal>Hexagonal. The circular and square structures are most popular of the spiral inductors because of less complex analytical design and ease of fabrication. However, other polygonal spirals have also been used in circuit design. Some designers prefer polygons with more than four sides to improve performance of quality factor of the spiral inductors.

Reference

1 *Sunderarajan S. Mohan, Maria del Mar Hershenson, Stephen P. Boyd, and Thomas H. Lee, "Simple Accurate Expressions for Planar SpiralInductances," IEEE Journal Of Solid-State Circuits, Vol. 34, No. 10, pp. 1419-1424, October 1999.*

Design and Analysis of Hairpin Micro-Strip Line Bandpass Filter

This example shows how to design and analyze Hairpin filter in RF PCB Toolbox™.

Most wireless applications require high quality, low cost, and compact sized RF or microwave filters. Planar filters can be used to fulfill this purpose. Among planar filters hairpin filters are available in reduced size as compared to parallel coupled line structures. These hairpin filters may conceptually be obtained by folding the resonators of parallel-coupled half-wavelength resonator filters. The Hairpin filters are widely used as bandpass filters in transmitters and receivers at various frequencies.

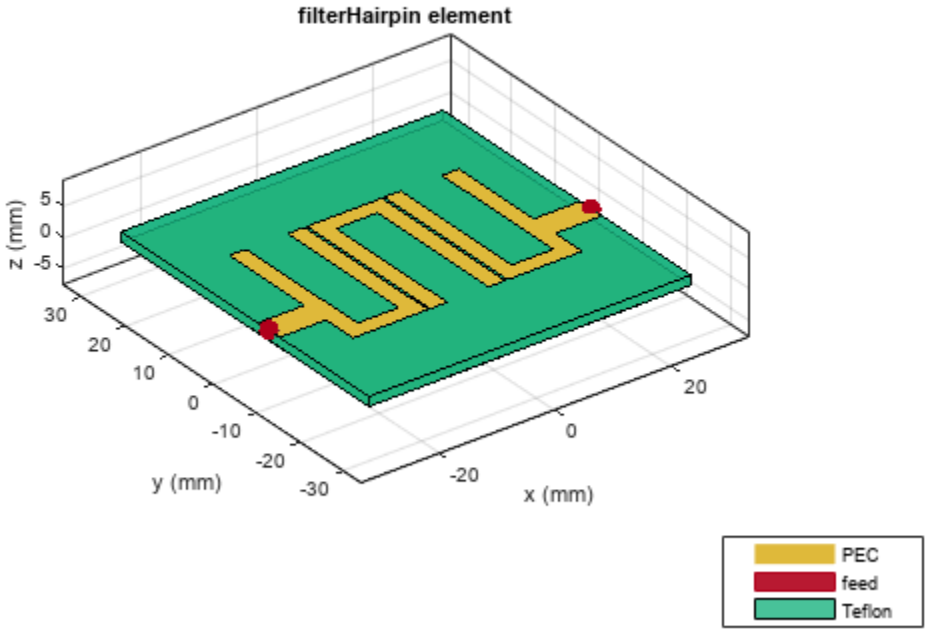
Create Variables

```
N = 3;
Ripple = 0.1;
BandWidth = 20;
Z0 = 50;
f = linspace(1.5e9,2.5e9,51);
EpsilonR = 6.15;
Height = 1.27e-3;
```

Tapped-Input Hairpin Filter

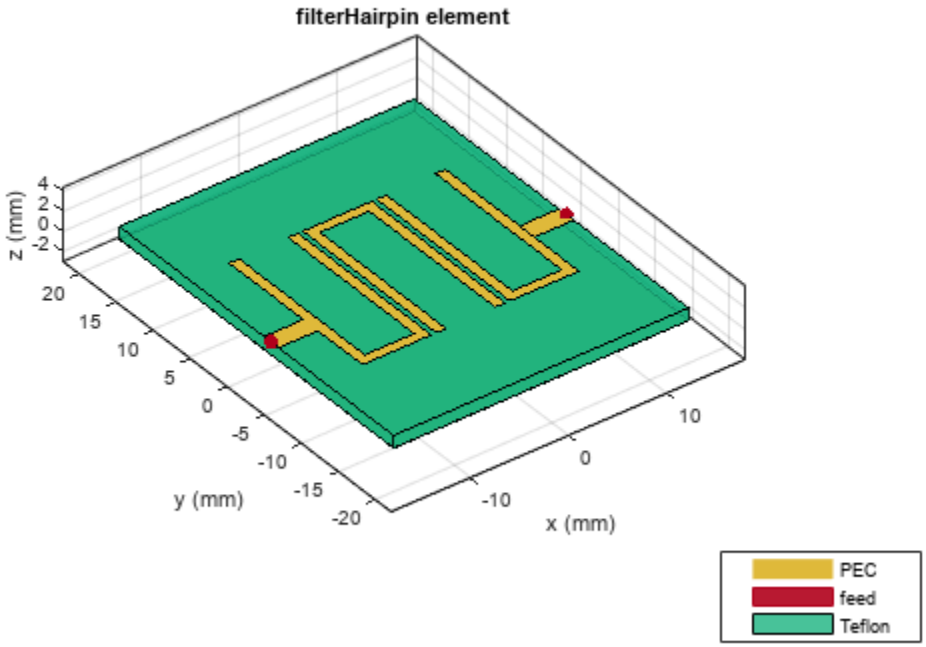
Use the `filterHairpin` object to create the hairpin filter and visualize it.

```
filter = filterHairpin;
figure;
show(filter);
view(-37,30);
```



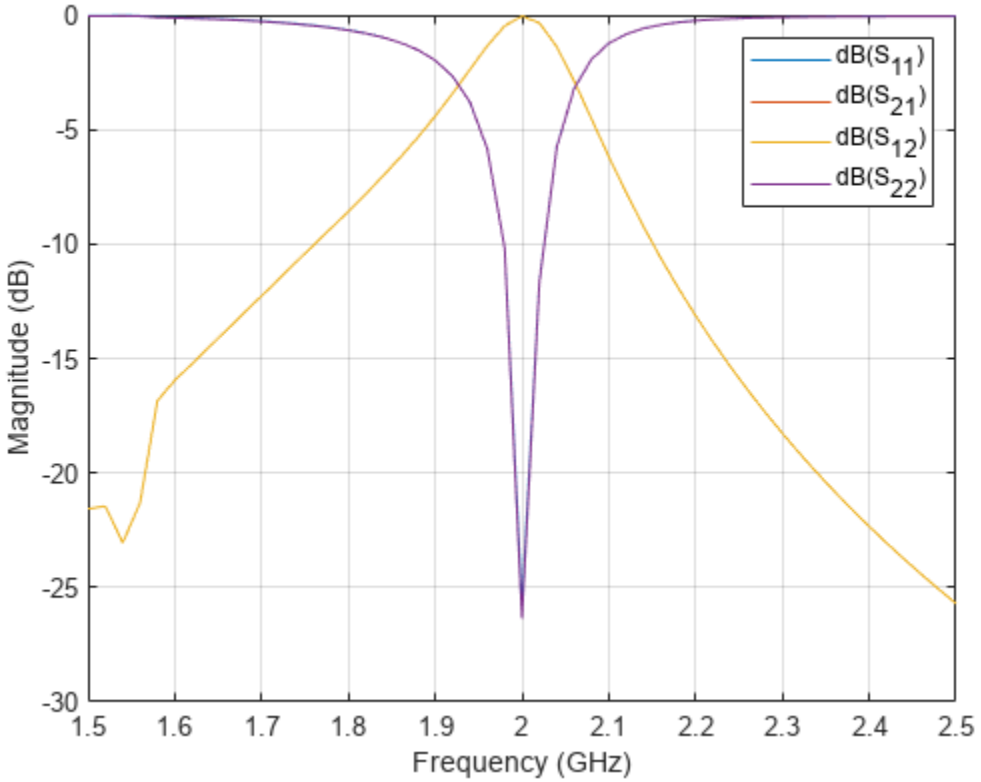
Use the `design` function to design a 3rd order hairpin filter with a `RippleFactor` of 0.1 dB and 20% fractional bandwidth and visualize it.

```
filter.FilterOrder = N;
filter.Height = Height;
filter.Substrate.EpsilonR = EpsilonR;
filter = design(filter,2e9,'FBW',BandWidth,'RippleFactor',Ripple);
figure;
show(filter);
view(-37,35);
```



Use the `sparameters` function to calculate the s-parameters for the hairpin filter and plot it using `rfplot` function.

```
spar = sparameters(filter,f);
figure;
rfplot(spar);
```



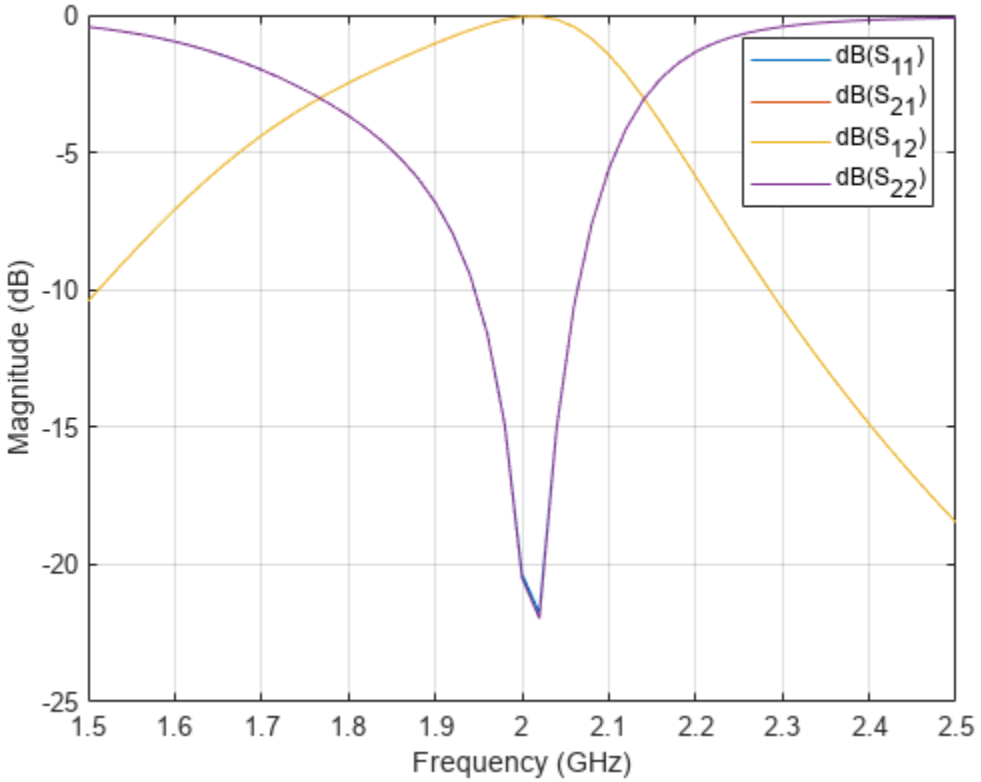
The result shows that the filter resonates close to the design frequency of 2 GHz. The design function uses analytical equations and the spacing between the resonators is generally calculated using the practical approach. Hence the design function sets the spacing to 0.5 mm between all the resonators. To get the accurate results, the spacing between the resonators needs to be adjusted. Increase the bandwidth of the filter by reducing the spacing between the resonators.

Set the `Spacing` between the resonators to 0.05 mm.

```
filter.Spacing = [0.05e-3 0.05e-3];
```

Use the `sparameters` function to calculate the s-parameters for the hairpin filter and plot it using `rfplot` function.

```
spar = sparameters(filter,f);
figure;
rfplot(spar);
```

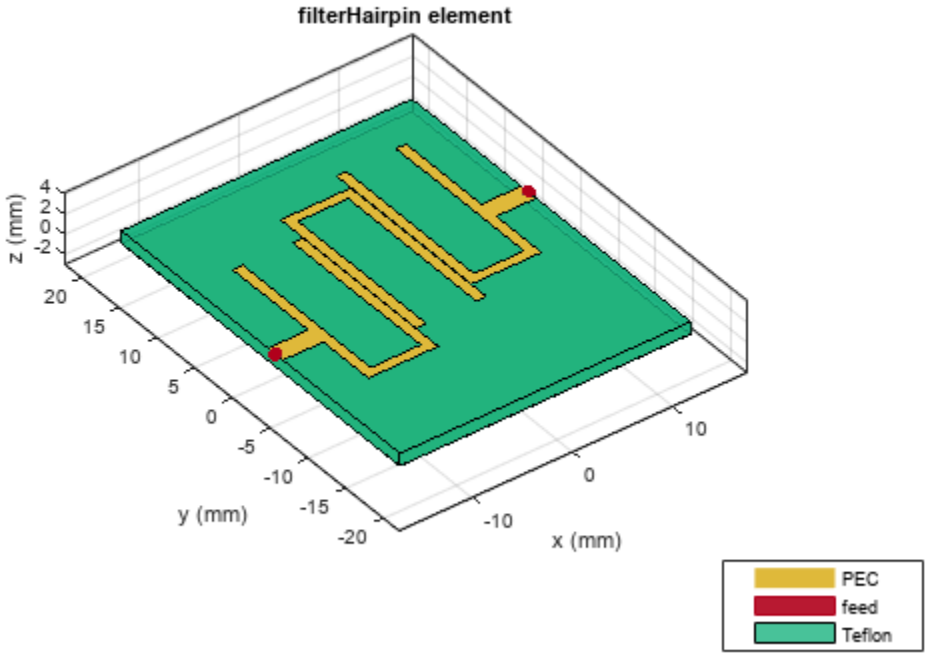


Shifting Resonators and Feed Lines

You can also shift the resonators and the feeds using the `ResonatorOffset` and `FeedOffset` properties of `filterHairpin.`

Change the `ResonatorOffset` and `FeedOffset` and visualize it.

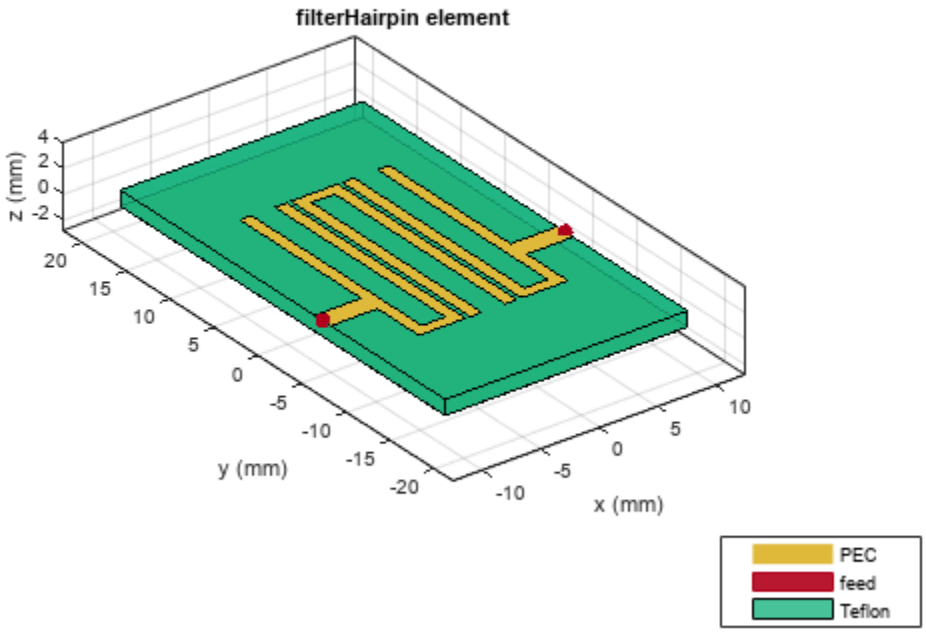
```
filter.ResonatorOffset = [0e-3 3e-3 5e-3];
filter.FeedOffset(2) =  filter.FeedOffset(2) + 5e-3;
figure;
show(filter);
view(-37,37);
```



Comparing Results with the Paper

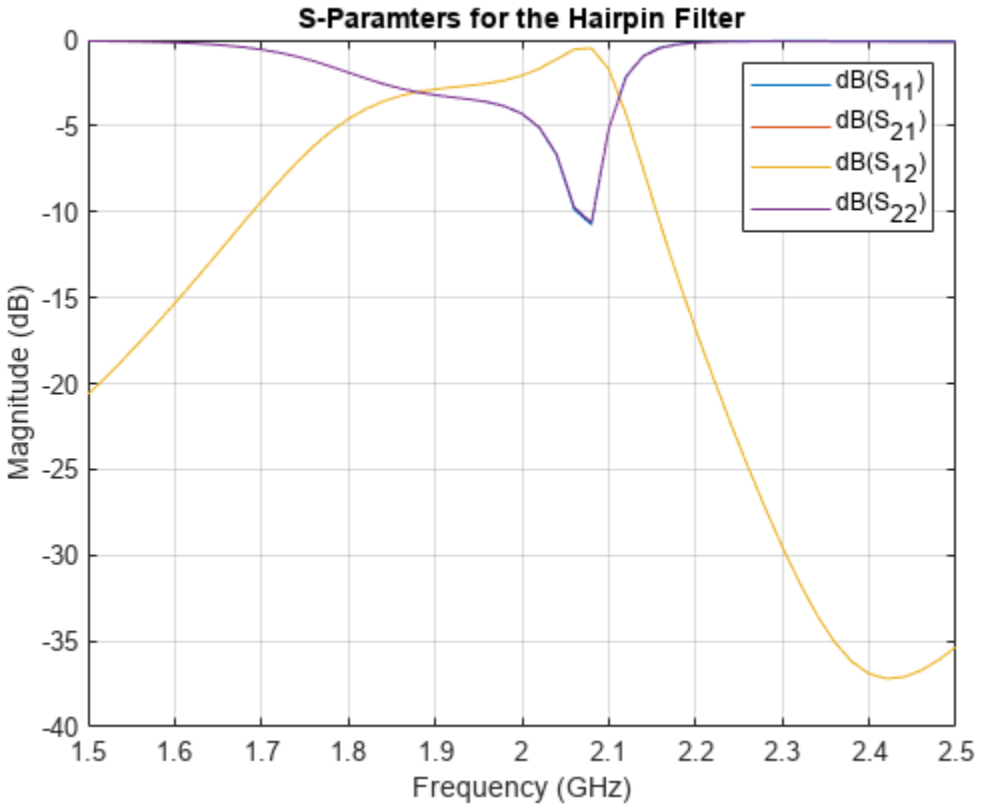
Change the dimensions of the filter as given in [1]. Change the `spacing` on the `filterHairpin` and set the spacing to 0.4 mm and set the length of the U-Shaped resonator to 2 mm.

```
filter.ResonatorOffset = [0e-3 0e-3 0e-3];
filter.Resonator(1).Length(1) = 20.4e-3;
filter.Resonator(1).Length(2) = 2e-3;
filter.Resonator(1).Length(3) = 20.4e-3;
filter.Spacing = [0.4e-3 0.4e-3];
filter.PortLineWidth = 1.85e-3;
filter.FeedOffset = [-4.565e-3 -4.565e-3];
figure;
show(filter);
view(-37,29);
```



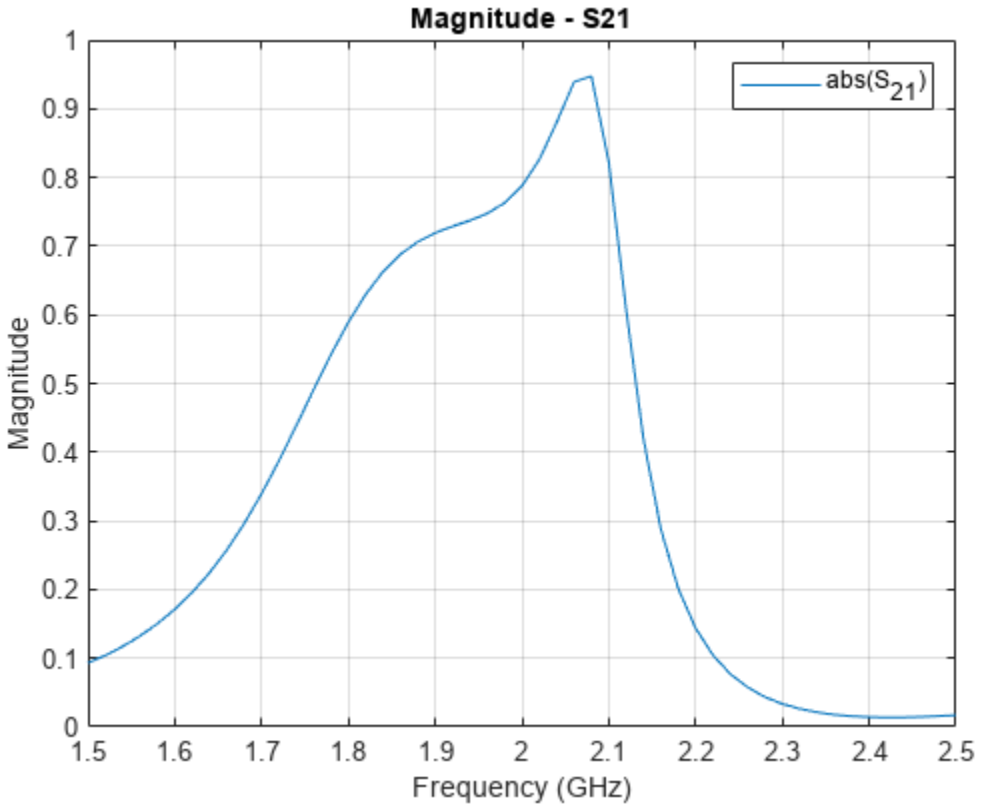
Use the `sparameters` function to calculate the s-parameters for the hairpin filter and plot it using `rfplot` function.

```
spar = sparameters(filter,f);
figure;
rfplot(spar);
title('S-Paramters for the Hairpin Filter');
```



Plot the absolute values of S21 over the frequency range using `rfplot` function.

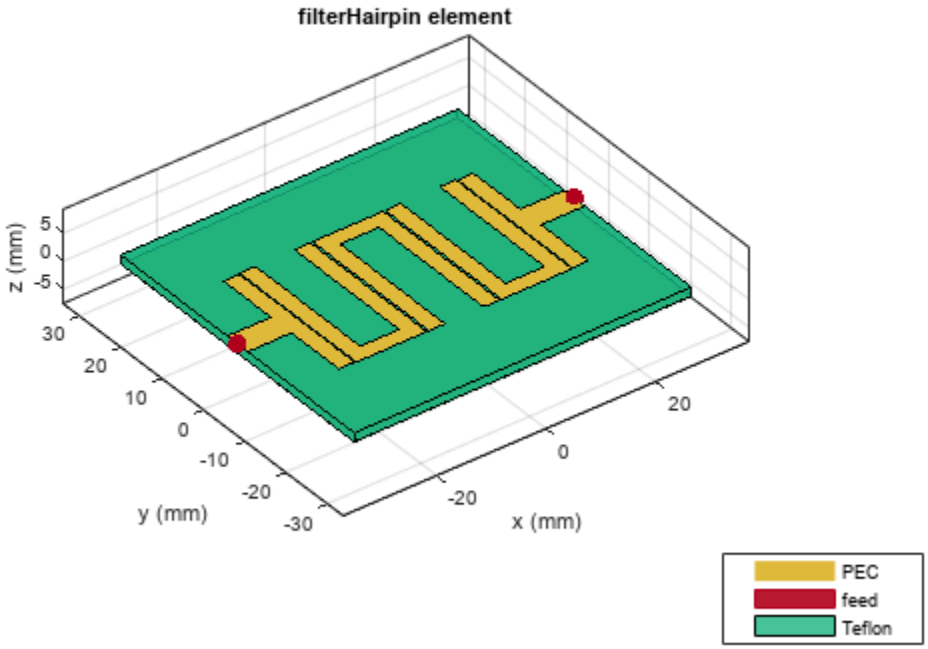
```
figure;
rfplot(spar,2,1,'abs');
title('Magnitude - S21');
```



Coupled-Input Hairpin Filter

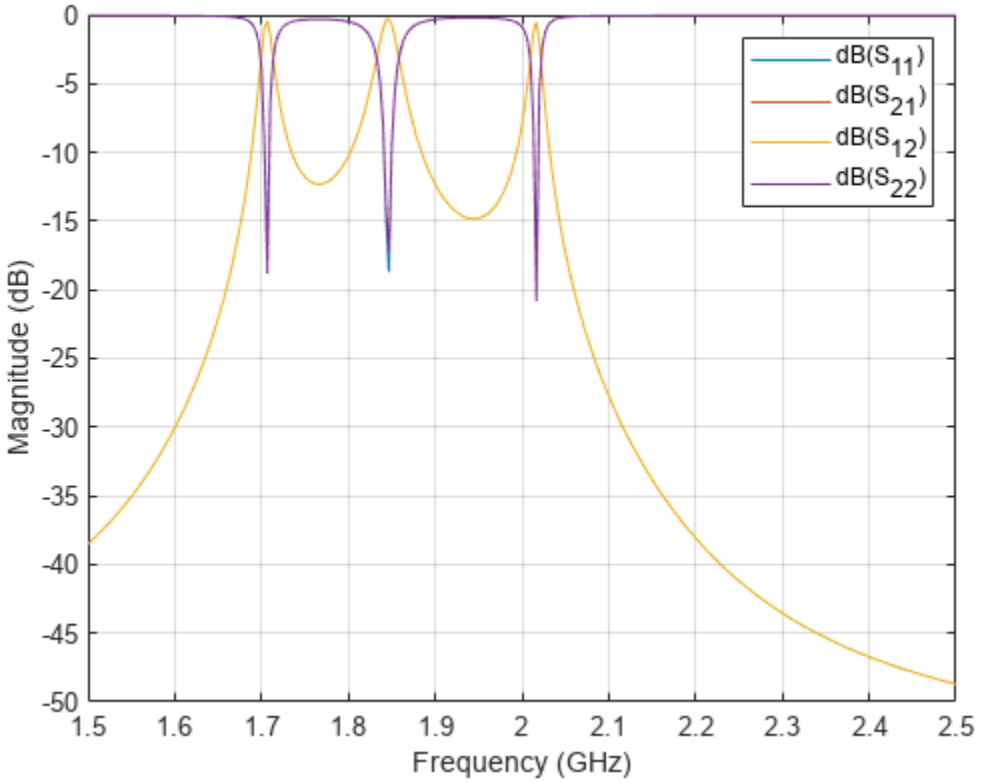
Use the `filterHairpin` object to create the hairpin filter. Change the `FeedType` property to `Coupled` to create the hairpin filter with a coupled Input.

```
filterC = filterHairpin;
filterC.FeedType = 'Coupled';
filterC.FeedOffset = [0 0];
figure;
show(filterC);
view(-37,35);
```



Use the `sparameters` function to calculate the s-parameters for the hairpin filter and plot it using `rfplot` function. The frequency range for the simulation is taken as 1.5 GHz to 2.5 GHz with 301 points. As this below simulation takes more time to run, load the sparameters from the s2p file and then plot the S-Parameters.

```
sparC = sparameters('sparams.s2p');
figure;
rfplot(sparC);
```



The result shows narrow band resonances at 1.7 GHz, 1.85 GHz, and 2 GHz. For this type of Input, the `ResonatorOffset` and `FeedOffset` can be used to create different types of Hairpin Filters.

References

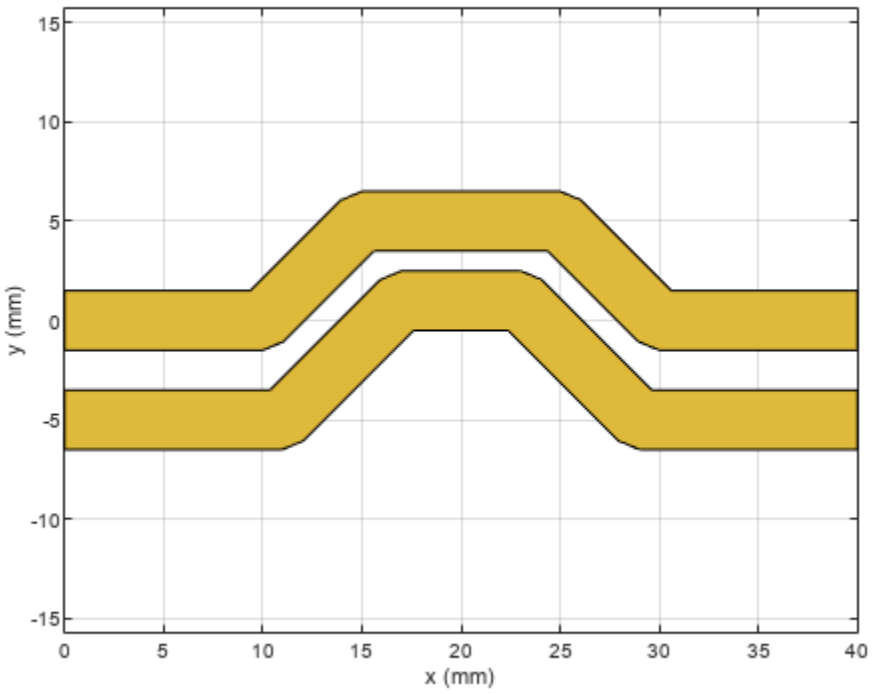
1 Nikunj Parikh , Pragya Katare , Ketan Kathal , Nandini Patel , Gaurav Chaitanya, Design and Analysis of Hairpin Micro-Strip Line Band Pass Filter.

Analyzing Crosstalk Between PCB Traces

This example shows how to create a pair of close traces using `traceLine` object and performs analysis on those traces to calculate the amount of unwanted coupling to the passive trace.

Create traces

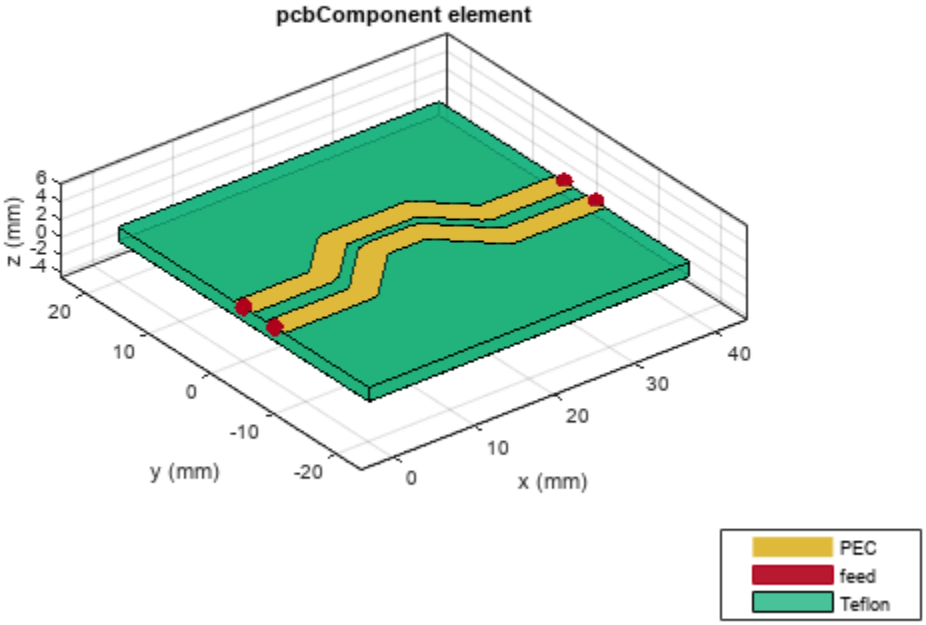
```
trace1 = traceLine;
trace1.Length = [10 5*sqrt(2) 10 5*sqrt(2) 10]*1e-3;
trace1.Angle  = [0 45 0 -45 0];
trace1.Width  = 3e-3;
trace1.Corner = "Miter";
trace2 = copy(trace1);
trace2.Length = [11 6*sqrt(2) 6 6*sqrt(2) 11]*1e-3;
trace2 = translate(trace2, [0,-5e-3,0]);
trace = trace1 + trace2 ;
figure;
show(trace);
```



Create PCB Component

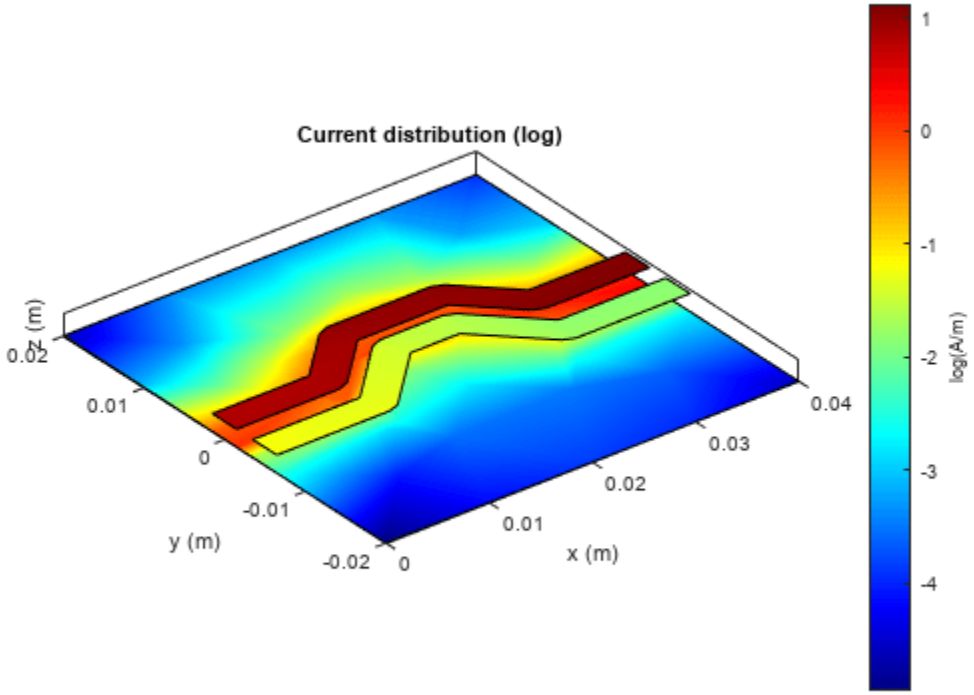
Use the `pcbComponent` to create the PCB stack for the shape. For creating a PCB stack, the trace created above is used as a top layer. The middle layer is a dielectric and the bottom layer is a ground plane. Use the `dielectric` object to create the Teflon dielectric. Use `traceRectangular` object to create a rectangular ground plane. Assign the trace, dielectric(d), and groundplane to the `Layers` property of the `pcbComponent`. Assign the `FeedLocations` at the ends of the trace and visualize it.

```
pcb = pcbComponent;
d = dielectric('Teflon');
d.Thickness = pcb.BoardThickness;
groundplane = traceRectangular('Length', 40e-3,'Width',40e-3,'Center',[40e-3/2,0]);
pcb.Layers = {trace,d,groundplane};
pcb.FeedLocations = [0,0,1,3;40e-3,0,1,3;40e-3,-5e-3,1,3;0e-3,-5e-3,1,3];
pcb.BoardShape = groundplane;
pcb.FeedDiameter = trace1.Width/2;
figure;
show(pcb);
```



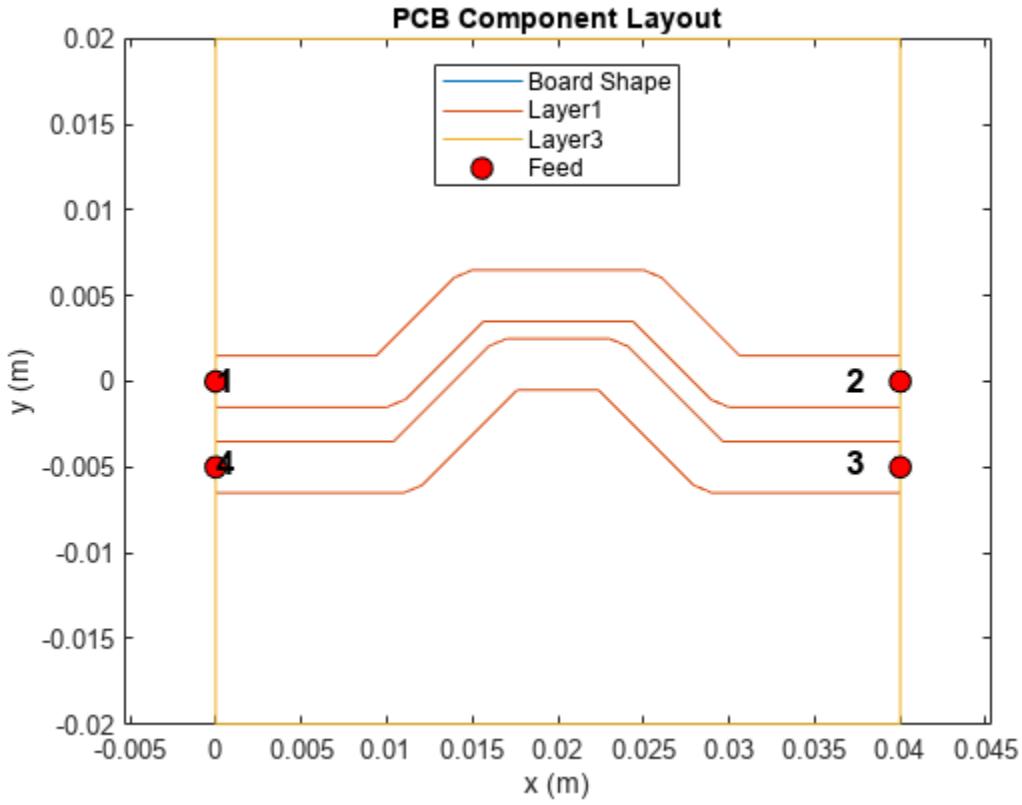
Use `current` function to plot the current distribution on the trace

```
figure;
current(pcb,1e9,'scale','log');
```



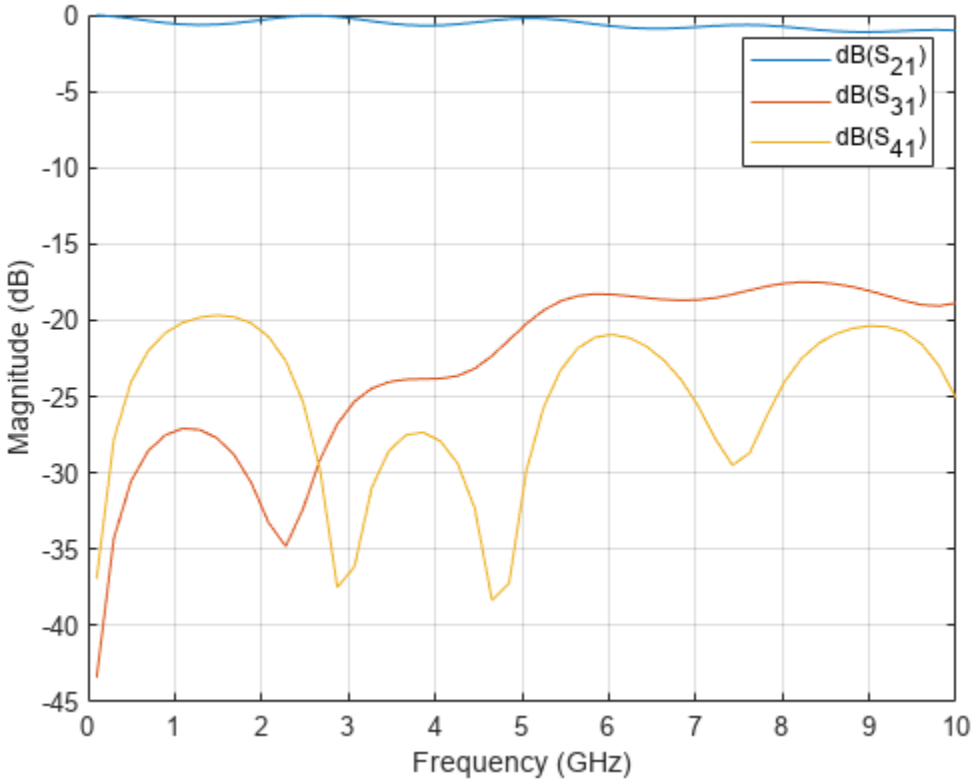
Use the `layout` function to show the layout of the pcbComponent.

```
figure;
layout(pcb);
```



Use `sparameters` function to calculate the leakage power from Trace 1 to Trace 2.

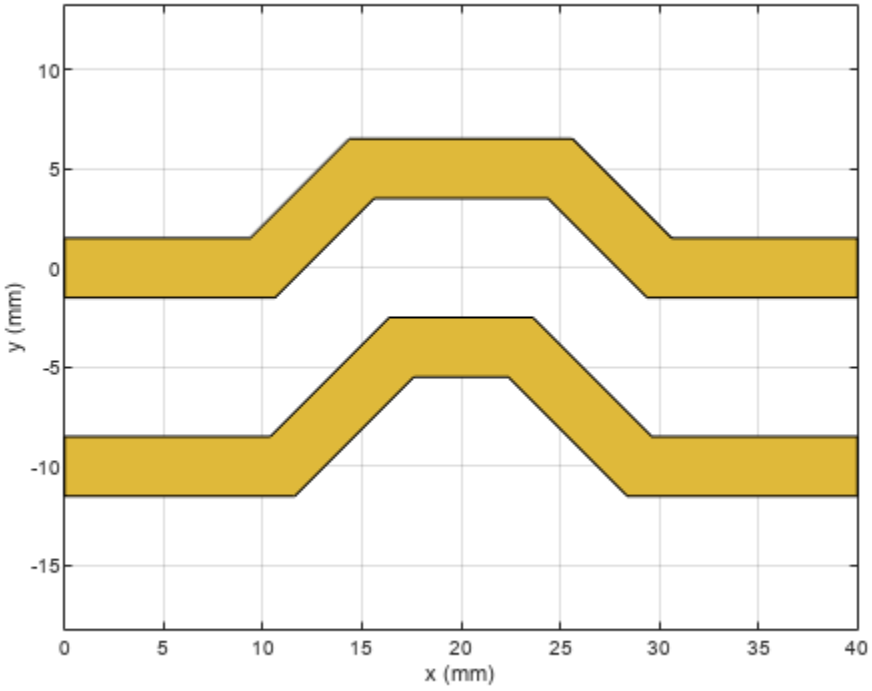
```
spar = sparameters(pcb,linspace(0.1e9,10e9,51));
figure;
rfplot(spar,2:4,1)
```



The S31 shows the power coupled from the top trace to the bottom trace. This is the unwanted coupling from the top line to the bottom line and must be reduced. To reduce the coupling, increase the spacing between the traces.

Increase the Spacing Between Traces

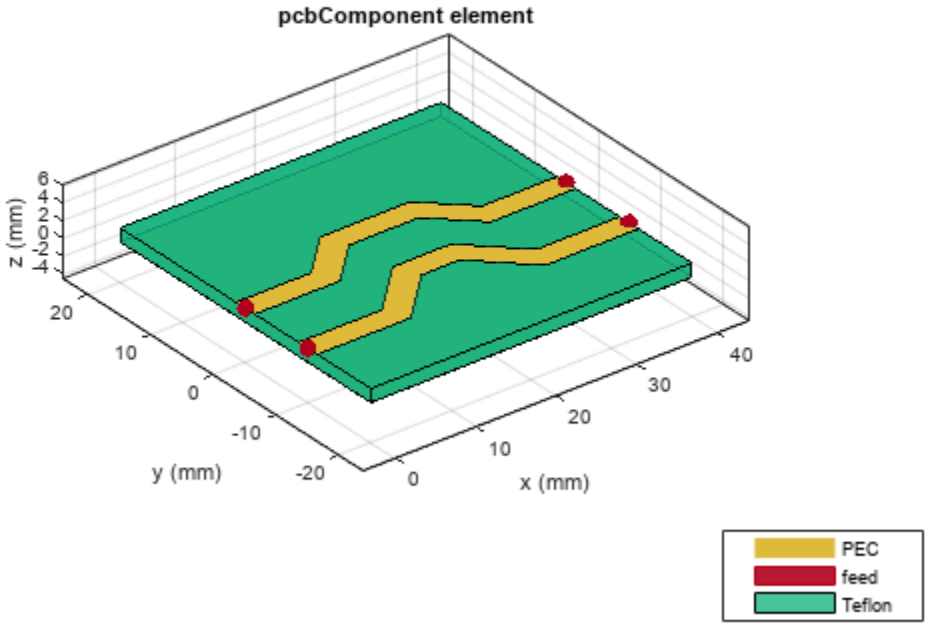
```
trace1 = traceLine;
trace1.Length = [10 5*sqrt(2) 10 5*sqrt(2) 10]*1e-3;
trace1.Angle  = [0 45 0 -45 0];
trace1.Width  = 3e-3;
trace1.Corner = "Sharp";
trace2 = copy(trace1);
trace2.Length = [11 6*sqrt(2) 6 6*sqrt(2) 11]*1e-3;
trace2 = translate(trace2, [0,-10e-3,0]);
trace = trace1 + trace2 ;
figure;
show(trace);
```



Create PCB Component

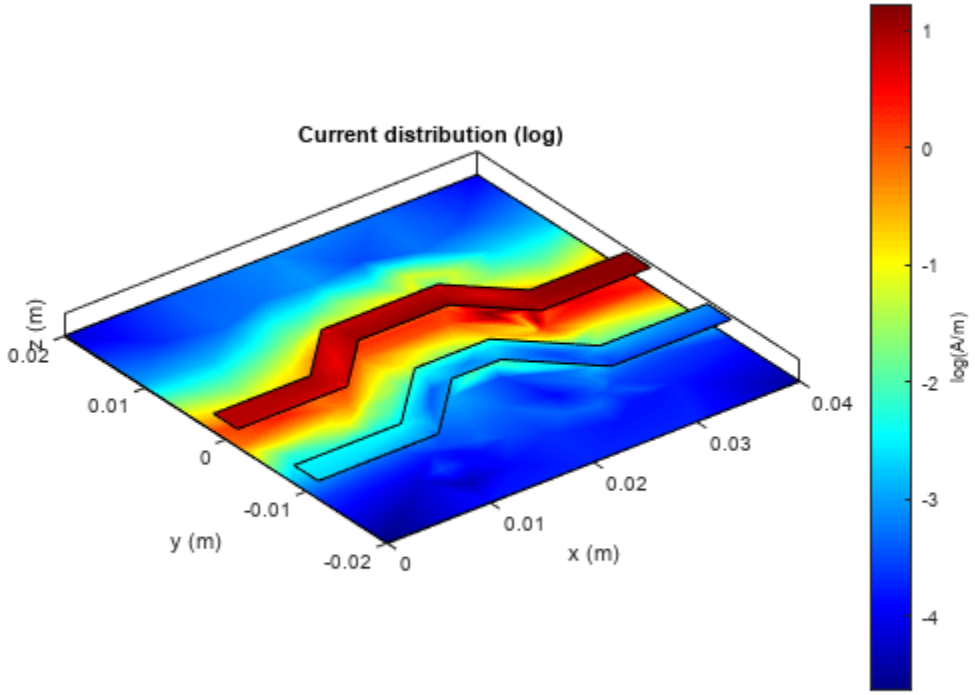
Update the properties of the created earlier and visualize it.

```
groundplane = traceRectangular('Length', 40e-3,'Width',40e-3,'Center',[40e-3/2,0]);
pcb.Layers = {trace,d,groundplane};
pcb.FeedLocations = [0,0,1,3;40e-3,0,1,3;40e-3,-10e-3,1,3;0e-3,-10e-3,1,3];
figure;
show(pcb);
```



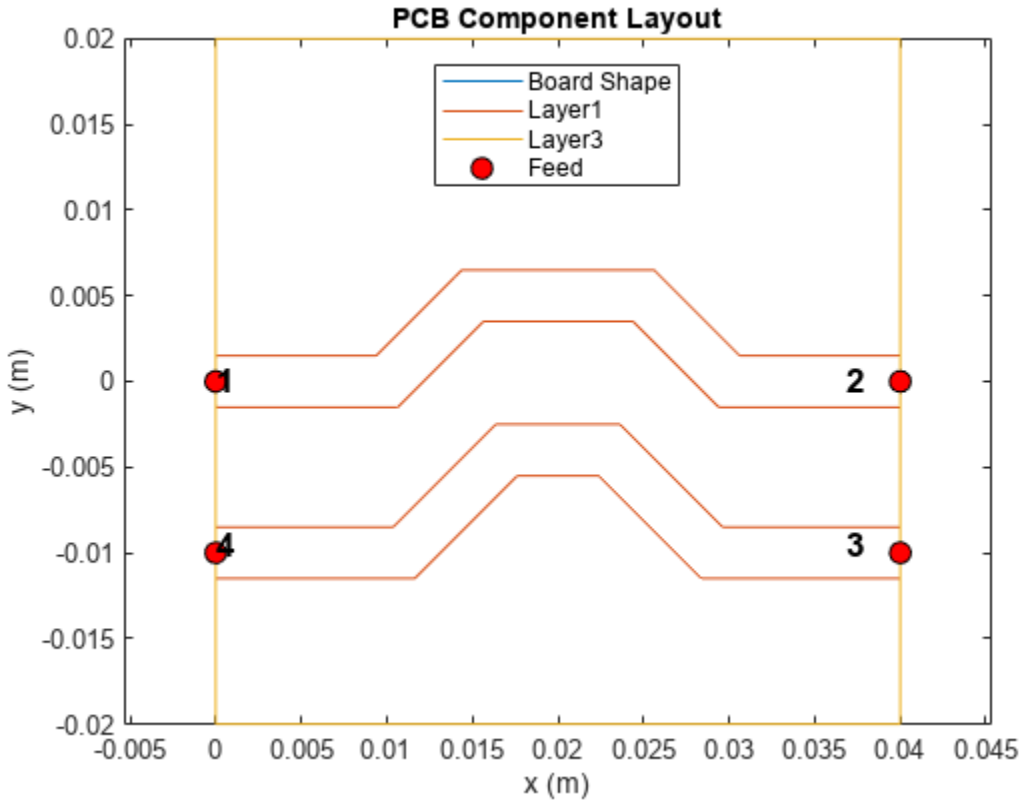
Use `current` function to plot the current distribution on the trace.

```
figure;
current(pcb,1e9,'scale','log');
```



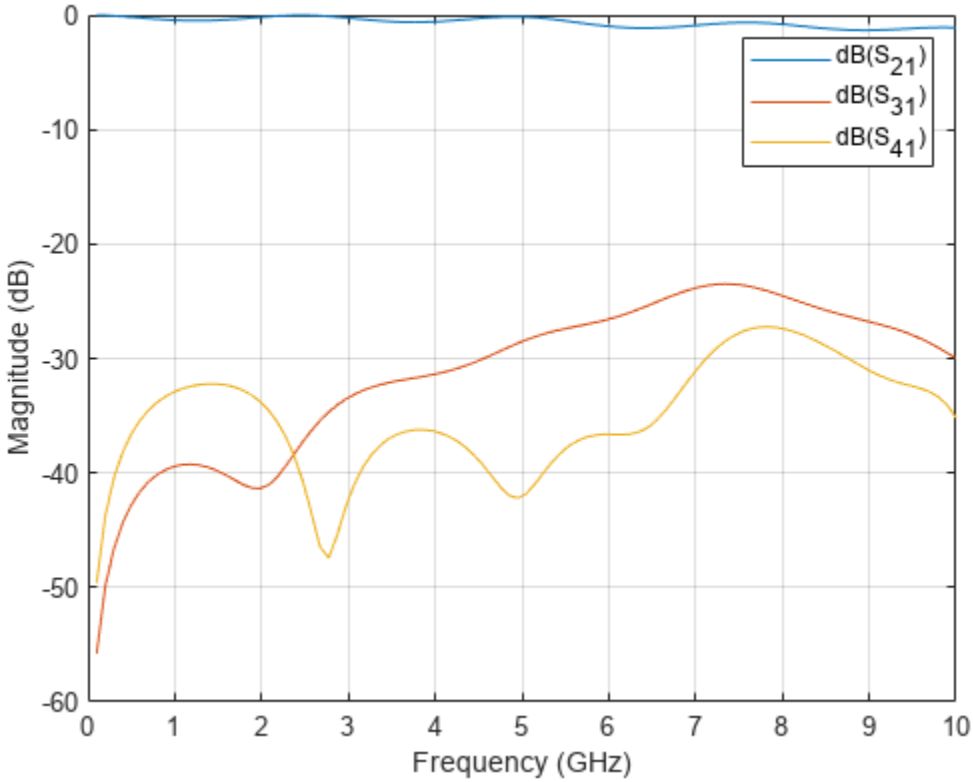
Use the `layout` function to show the layout of the `pcbComponent`.

```
figure;
layout(pcb);
```



Use `sparameters` function to calculate the leakage power from Trace 1 to Trace 2.

```
spar = sparameters(pcb,linspace(0.1e9,10e9,101));
figure;
rfplot(spar,2:4,1)
```

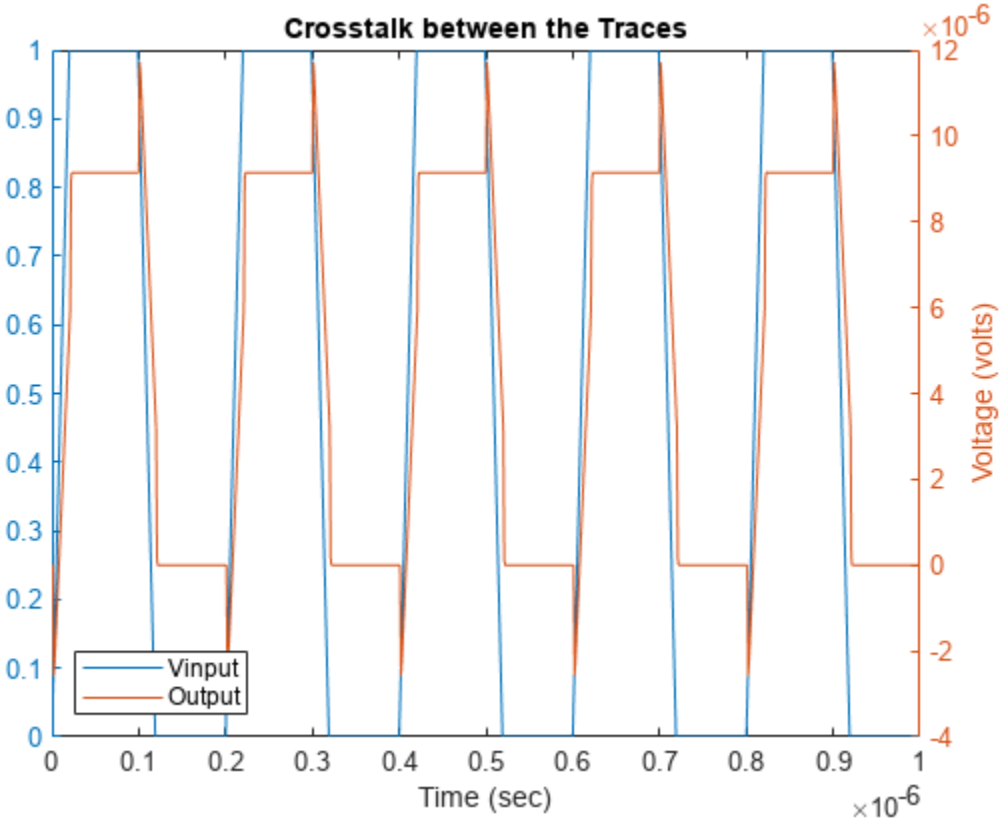


When the spacing between the traces is doubled to 10 mm, the coupling also reduces by around 10 dB and reaches to less than -25 dB.

Plot the Crosstalk Induced Voltage

Define a pulse signal with rise time and fall time and use the `timeresp` function to plot the output signal at port 2 for the input pulse signal.

```
sampleTime = 1e-9;
t = (0:999)'*sampleTime;
input = [0.05*(0:20)';ones(1,78)'; 0.05*(20:-1:0)'; zeros(1,80)'];
input = repmat(input,5,1);
fit   = rationalfit(spar);
output = timeresp(fit(3,1),input,sampleTime);
figure;
yyaxis left;
plot(t,input);
yyaxis right;
plot(t,output);
title('Crosstalk between the Traces');
xlabel('Time (sec)');
ylabel('Voltage (volts)');
legend('Vinput','Output','Location','SouthWest');
```



The output signal magnitude is around 0.1 mV as seen from the result.

References

1) https://incompliancemag.com/article/visualizing-crosstalk-in-pcbs/

2) Basic Principles of Signal Integrity, Altera

Create and Analyze PCB Interconnects using Custom Traces

This example shows how to use the custom traces `traceLine` and `tracePoint` in RF PCB Toolbox and create PCB traces with different shapes and orientations. The `traceLine` uses the lengths and angles properties to create a trace and the `tracePoint` uses points to create the trace. For both the traces, corner property can be set to `Sharp`, `Miter,` and `Smooth` to create specific corners at all the bends.

Line Trace

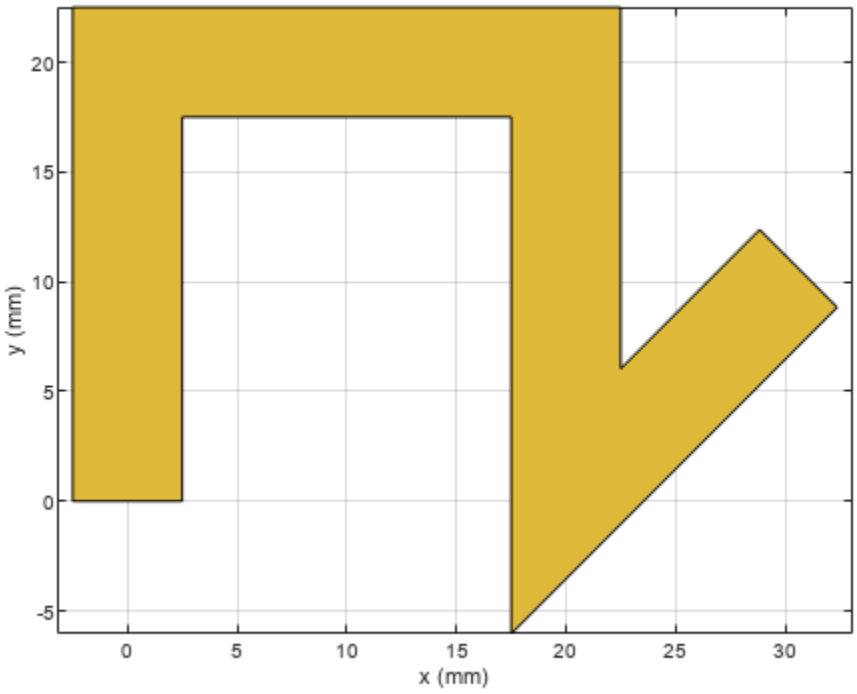
Use the `traceLine` object to create a line trace. Observe that the traceLine has Length, Angle, Width, and Corner as properties. The line trace is a combination of all the rectangles which are created based on the `Length, Angle`, and the `Corner` property is used to specify the type of corner at each bend. You can specify the corners as sharp, miter, or smooth . By default the `Corner` property is set to `Sharp`.

```
trace = traceLine
```

```
trace =
  traceLine with properties:

          Name: 'mytraceLine'
    StartPoint: [0 0]
        Length: [0.0200 0.0200 0.0200 0.0150]
         Width: 0.0050
         Angle: [90 0 -90 45]
        Corner: "Sharp"
```

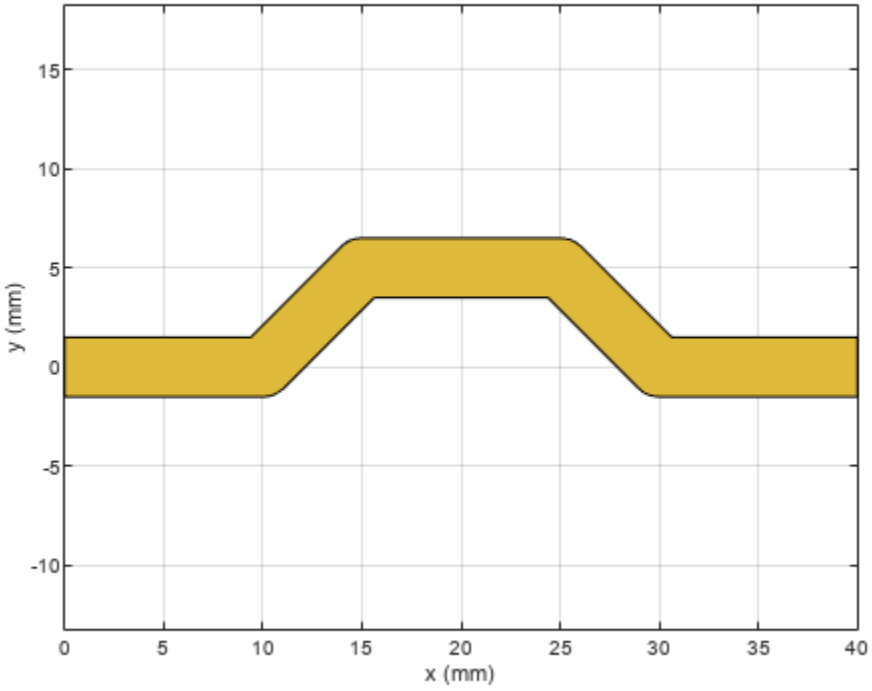
Visualize the traceLine using the `show` function.

```
figure;
show(trace);
```



Change the `Length` and `Angle` property on the traceLine and set the `Corner` to `smooth`.

```
trace.Length = [10 5*sqrt(2) 10 5*sqrt(2) 10]*1e-3;
trace.Angle  = [0 45 0 -45 0];
trace.Width  = 3e-3;
trace.Corner = "smooth";
figure;
show(trace);
```

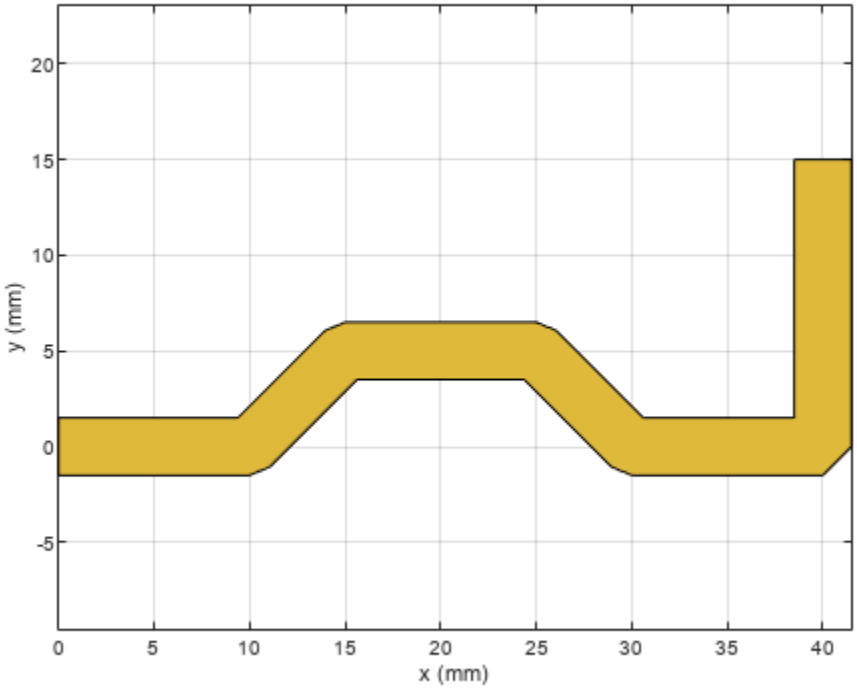


The output shows that the trace is created using the angle specified for each corresponding length input. The corners are specified as smooth and hence it creates curved edges.

Extend the Line Trace

Add a 15 mm line perpendicular to the earlier trace. So add 15 mm to the `Length` property and 90 deg to the `Angle` property of the trace and set the corner to `Miter`.

```
trace.Length = [10 5*sqrt(2) 10 5*sqrt(2) 10 15]*1e-3;
trace.Angle  = [0 45 0 -45 0 90];
trace.Width  = 3e-3;
trace.Corner = "Miter";
figure;
show(trace);
```

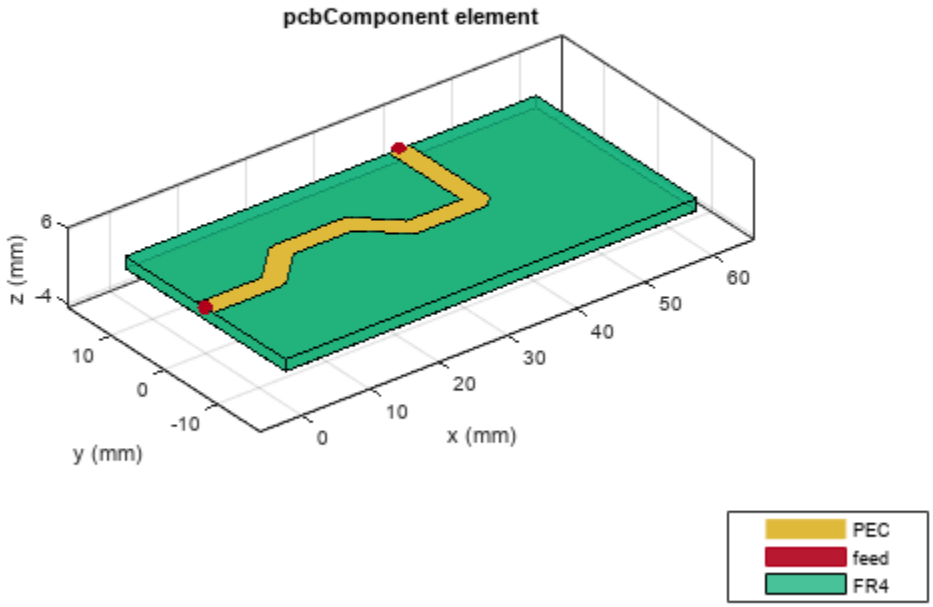


The output shows that the trace is created using the angle specified for each corresponding length input. The corners is specified as Miter and hence it creates mitered edges. Observe that the ends of the trace are at [0,0] and [40,15] mm where the feed points are assigned in the next section.

Create PCB Component

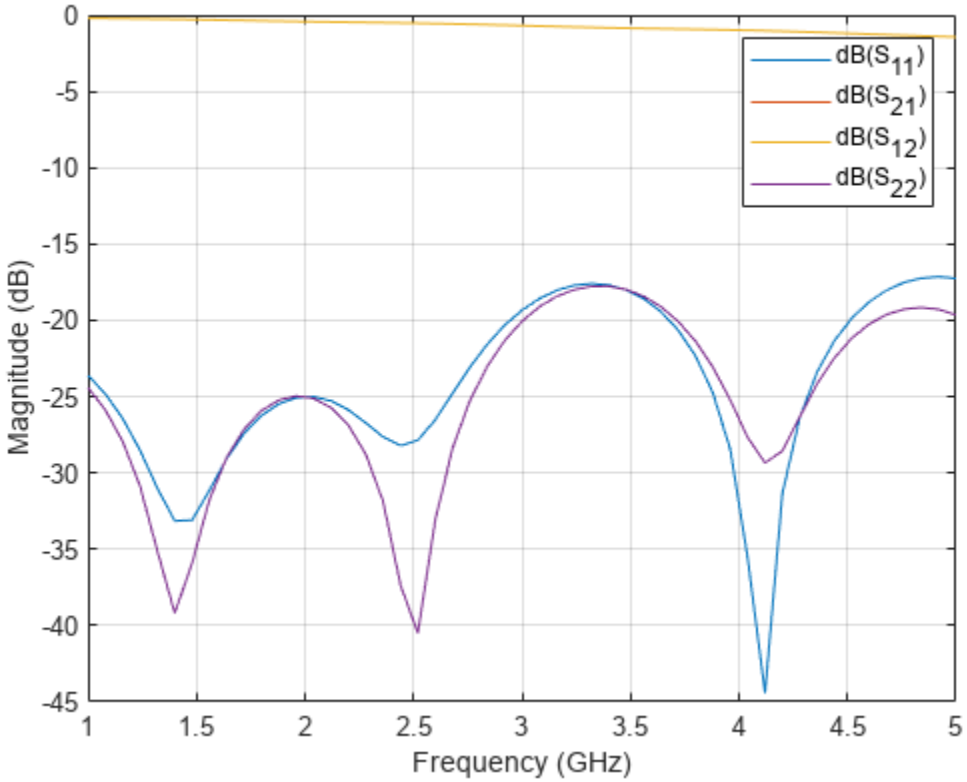
Use the `pcbComponent` to create the PCB stack for the shape. For creating a PCB stack, use the trace created as the top layer. The middle layer is a dielectric and the bottom layer is ground plane. Use the `dielectric` object to create the FR4 dielectric. Use `traceRectangular` object to create a rectangular ground plane. Assign the `trace`, dielectric(`d`), and `groundplane` to the `Layers` property on the `pcbComponent`. Assign the `FeedLocations` at the ends of the trace and visualize it.

```
pcb = pcbComponent;
d = dielectric('FR4');
d.Thickness = pcb.BoardThickness;
groundplane = traceRectangular('Length', 60e-3,'Width',30e-3,'Center',[60e-3/2,0]);
pcb.Layers = {trace,d,groundplane};
pcb.BoardShape = groundplane;
pcb.FeedDiameter = trace.Width/2;
pcb.FeedLocations = [0,0,1,3;40e-3,15e-3,1,3];
figure;
show(pcb);
```



Use `sparameters` to calculate the s-parameters of the trace and plot it using `rfplot` function.

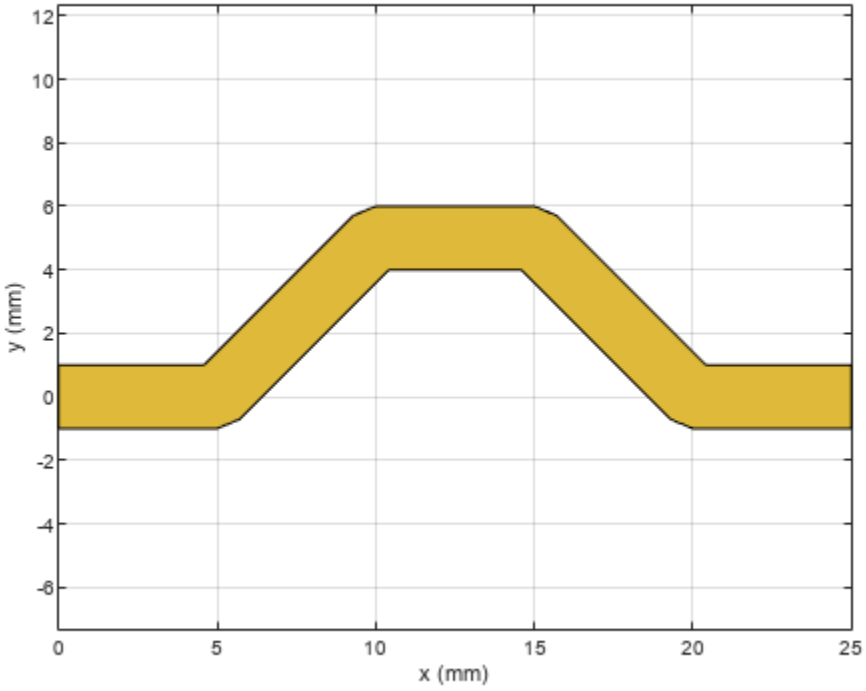
```
spar = sparameters(pcb,linspace(1e9,5e9,51));
figure;
rfplot(spar);
```



Point Trace

Use the tracePoint object to create a Point trace. Observe that the `tracePoint` has `TracePoints`, `Width,` and `Corner` as properties. Based on the Points, the tracePoint creates the trace and the Corner property is used to specify the type of corner at each bend i.e. you can specify either sharp, miter or smooth. Change the `TracePoints` and set the corner poperty to `Miter`

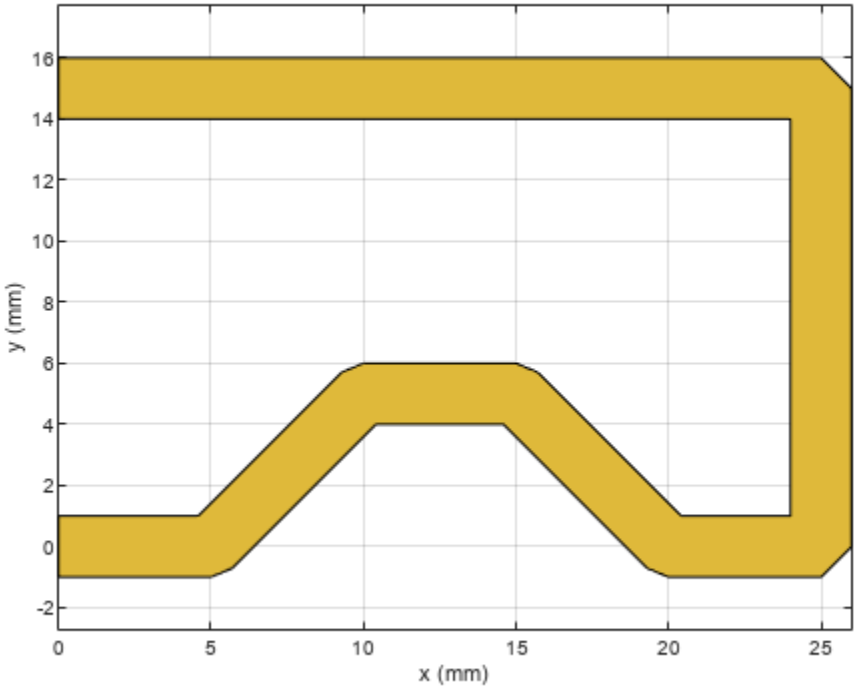
```
trace = tracePoint;
trace.TracePoints = [0 0;5 0;10 5;15 5;20 0;25 0]*1e-3;
trace.Corner = "Miter";
figure;
show(trace);
```



Extend the trace

Add an addional L-Shaped trace to the earlier trace so that the feed points are aligned to the same edge. Add two points in the `TracePoints` property as shown below. Set the `Corner` to `Miter`.

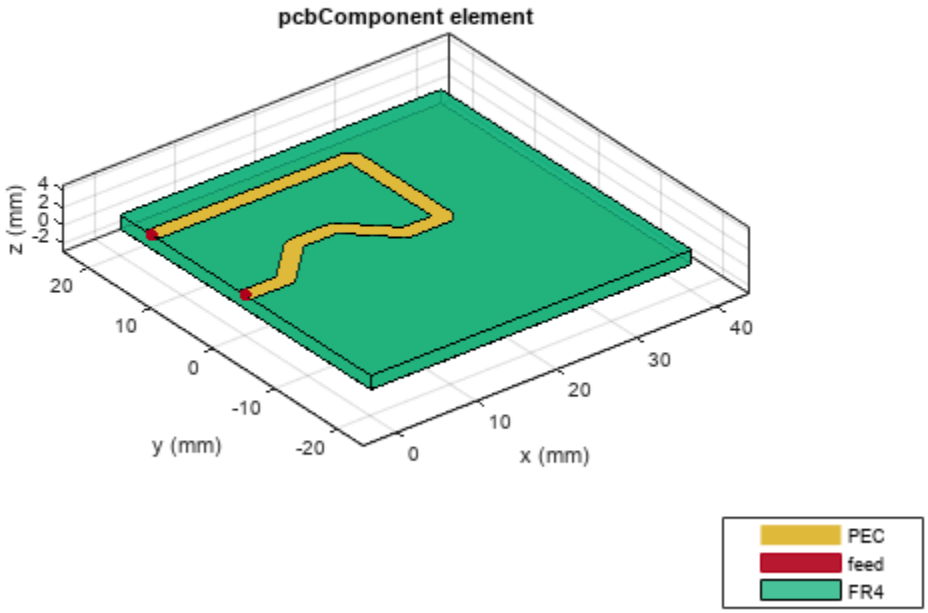
```
trace.TracePoints = [0 0;5 0;10 5;15 5;20 0;25 0;25 15;0 15]*1e-3;
trace.Corner = "Miter";
figure;
show(trace);
```



Create PCB Component

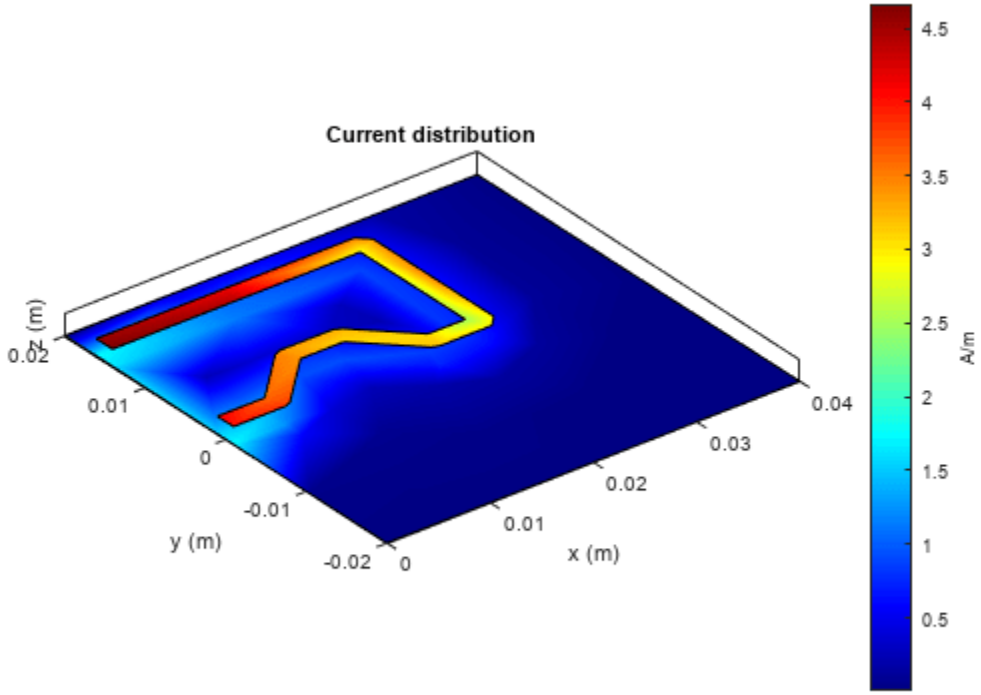
Use the `pcbComponent` to create the PCB stack for the shape. For creating a PCB stack, use the trace created as a top layer. The middle layer is a dielectric and the bottom layer is ground plane. Use the `dielectric` object to create the FR4 dielectric. Use `traceRectangular` object to create a rectangular ground plane. Assign the `trace`, dielectric(d) and `groundplane` to the `Layers` property on the `pcbComponent`. Assign the `FeedLocations` at the ends of the trace and visualize it.

```
pcb = pcbComponent;
d = dielectric('FR4');
d.Thickness = pcb.BoardThickness;
groundplane = traceRectangular('Length', 40e-3,'Width',40e-3,'Center',[40e-3/2,0]);
pcb.Layers = {trace,d,groundplane};
pcb.FeedLocations = [0,0,1,3;0e-3,15e-3,1,3];
pcb.BoardShape = groundplane;
pcb.FeedDiameter = trace.Width/2;
figure;
show(pcb);
```



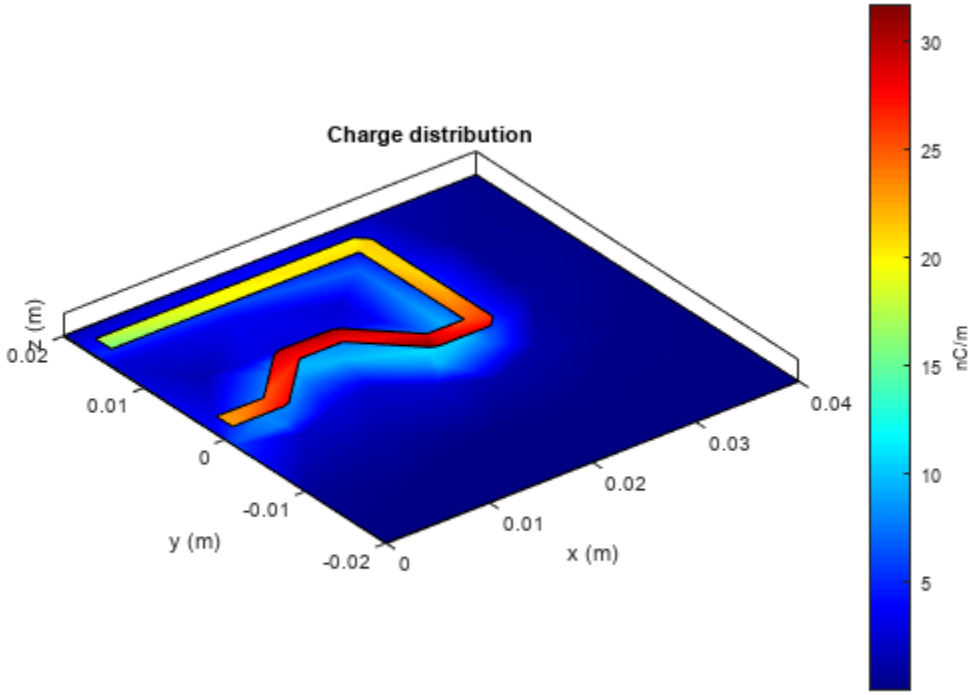
Use `current` function to plot the current distribution on the trace

```
figure;
current(pcb,1e9);
```



Use `charge` function to plot the charge on the trace

```
figure;
charge(pcb,1e9);
```


Charge distribution
z (m)
0.02
0.01
0
-0.01
-0.02
y (m)
0
0.01
0.02
0.03
0.04
x (m)
30
25
20
15
10
5
nC/m

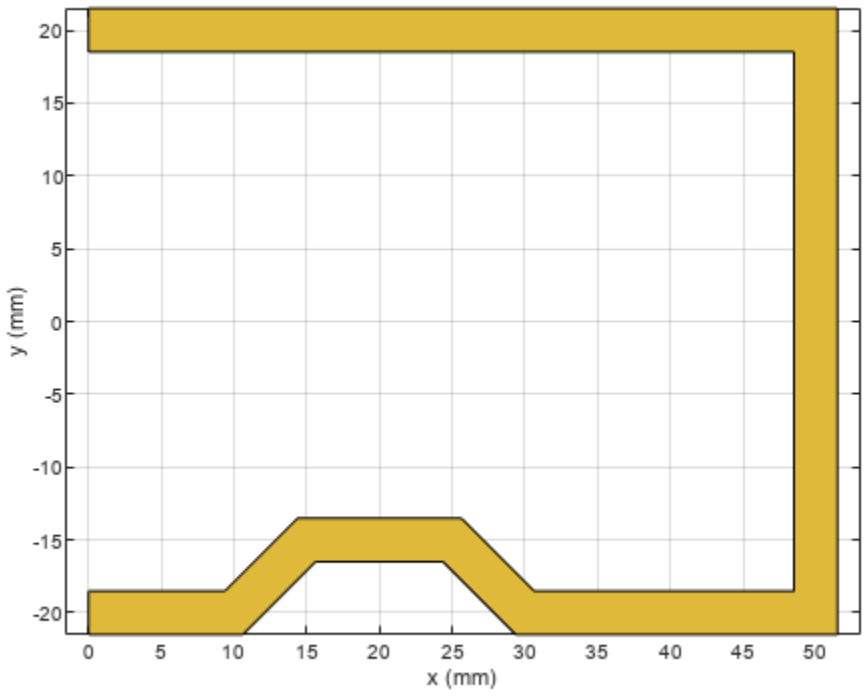
Routing Discipline to Reduce Crosstalk

This example shows how to use the RF PCB Toolbox to analyze the surface currents on the trace and focuses on the routing discipline that needs to be followed while designing the Mixed Signal PCB layout.

Build Trace

To analyze the currents on the trace, build a trace using `traceLine` object and visualize it.

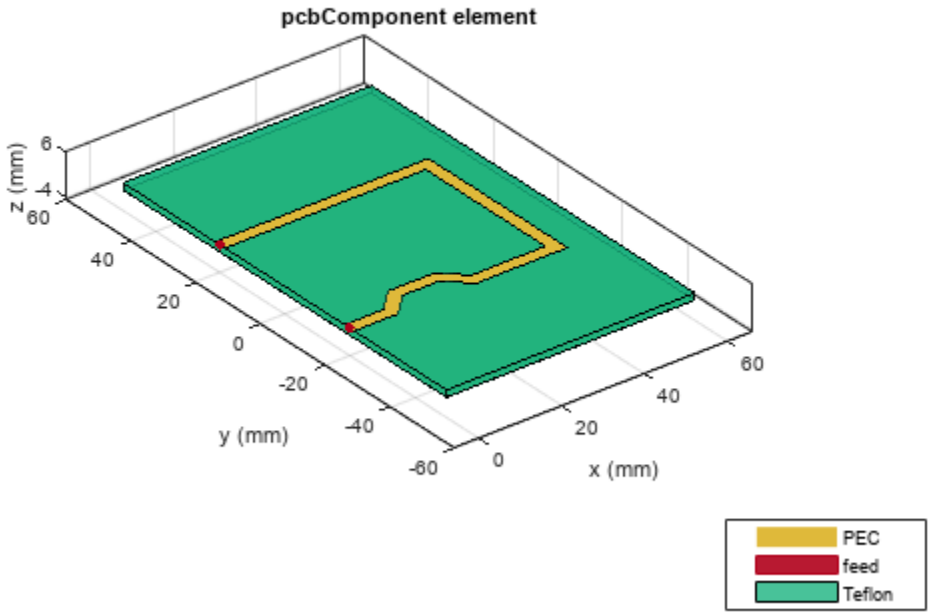
```
trace = traceLine;
trace.StartPoint = [0 -20e-3];
trace.Length = [10 5*sqrt(2) 10 5*sqrt(2) 20 40 50]*1e-3;
trace.Angle  = [0 45 0 -45 0 90 180];
trace.Width  = 3e-3;
trace.Corner = "Sharp";
figure;
show(trace)
```



Create PCB Component of Trace

Use the `pcbComponent` object to create the PCB stack up for the trace. Use the `dielectric` object function and create a teflon dielectric. Use the `traceRectangular` object to create the ground plane for the PCB. Assign the trace, dielectric, and the ground plane to the `Layers` property of the `pcbComponent`. Visualize the PCB stack using the `show` function.

```
pcb = pcbComponent;
d = dielectric('Teflon');
groundplane = traceRectangular('Length', 60e-3,'Width',100e-3,'Center',[60e-3/2,0]);
pcb.Layers = {trace,d,groundplane};
pcb.FeedLocations = [0e-3,-20e-3,1,3;0e-3,20e-3,1,3];
pcb.BoardShape = groundplane;
pcb.FeedDiameter = trace.Width/2;
pcb.ViaDiameter = trace.Width/2;
figure;
show(pcb);
```

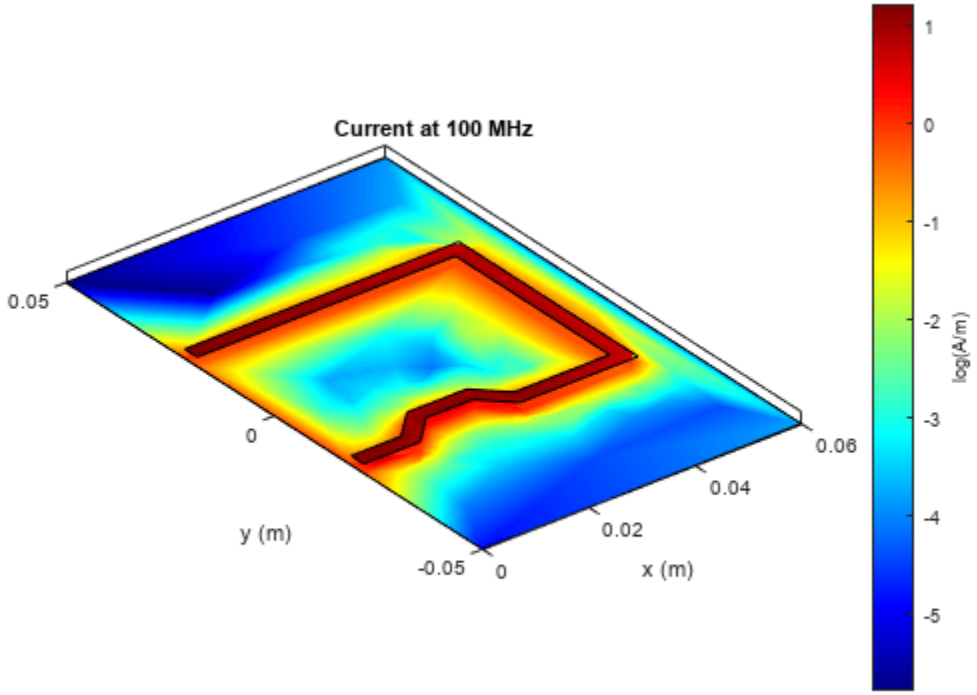


Surface Currents on PCB Trace and Ground Plane

Electromagnetic simulations for signals of different frequencies are shown here to visualize the paths in which the currents flow. The forward signal currents for each case are constrained to the trace. However, the return ground currents can flow anywhere on the ground plane.

Use the `current` function to plot the current distribution at 100 MHz.

```
figure;
current(pcb,0.1e9,'scale','log');
title('Current at 100 MHz');
```

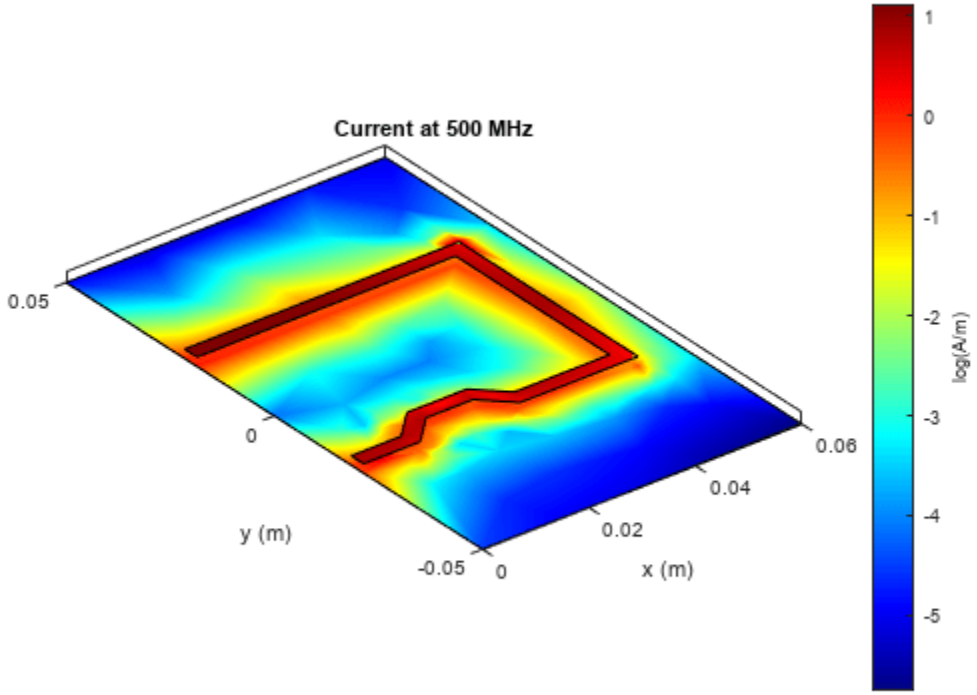


The figure shows that there is a red line between the two feed locations i.e, Source and Load, which means the current on the ground plane choses the path of least resistance and returns directly from the source to the load. A small amount of the ground current flows along the signal path (light red), while even smaller amounts flow in between these two paths as indicated by the blue color of much of the plane.

As the frequency increases, the return currents on the ground plane flow exactly beneath the trace.

Use the `current` function to plot the current distribution at 500 MHz.

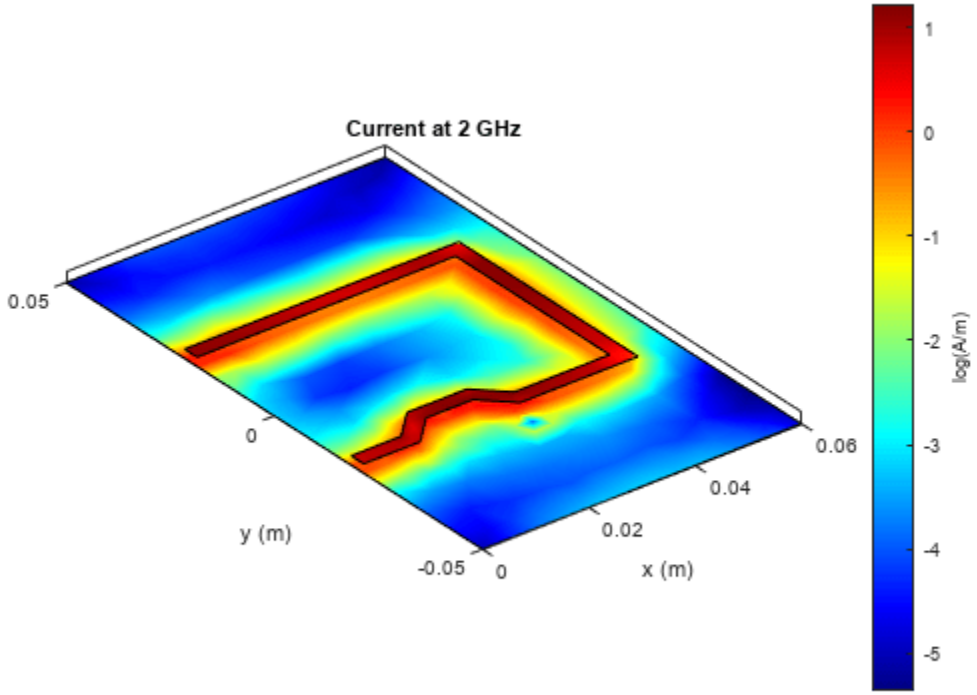
```
figure;
current(pcb,0.5e9,'scale','log');
title('Current at 500 MHz');
```



This figure shows that the return current on the ground plane flows only along the trace and doesn't flow between the source and load. As the frequency increases, the return currents on the ground plane flow exactly beneath the trace and this fact can be seen in the subsequent figures.

Use the `current` function to plot the current distribution at 2 GHz.

```
figure;
current(pcb,2e9,'scale','log');
title('Current at 2 GHz');
```



This results shows that the high frequency current on the ground plane flows beneath the trace and the routing of the PCB traces on the board becomes very critical and must be followed carefully to avoid cross talk with other traces.

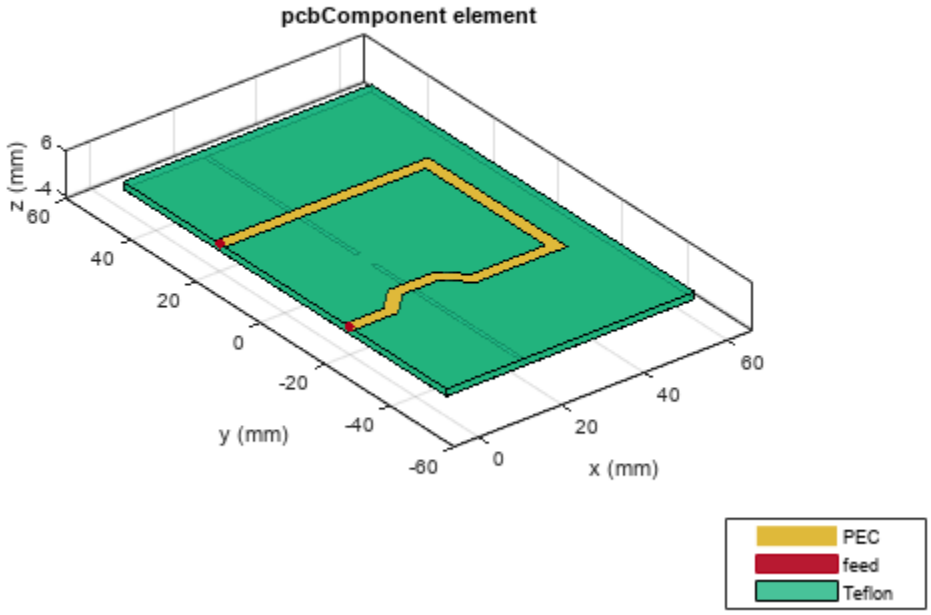
Routing with Gap in Ground Plane

For mixed signal PCB layout, separate the ground plane into analog ground and digital ground to reduce the interference between these traces. But splitting the ground plane and not following the routing discipline causes severe problems in the mixed signal PCB design. To understand this, use the same trace created in the above section but with a slit in the ground plane

Use the `pcbComponent` object to build the PCB Stack of the trace. Use `traceRectangular` to build two rectangular traces and subtract these traces from the ground plane to create the slit in the ground plane and visualise it.

```
pcb = pcbComponent;
d = dielectric('Teflon');
groundplane1 = traceRectangular('Length', 60e-3,'Width',100e-3,'Center',[60e-3/2,0]);
slit1 = traceRectangular('Length', 1e-3,'Width',48e-3,'Center',[20e-3,27e-3]);
slit2 = traceRectangular('Length', 1e-3,'Width',50e-3,'Center',[20e-3,-27e-3]);
groundplane = groundplane1-slit1-slit2;
pcb.Layers  = {trace,d,groundplane};
pcb.FeedLocations = [0e-3,-20e-3,1,3;0e-3,20e-3,1,3];
pcb.BoardShape = groundplane1;
pcb.FeedDiameter = trace.Width/2;
pcb.ViaDiameter = trace.Width/2;
```

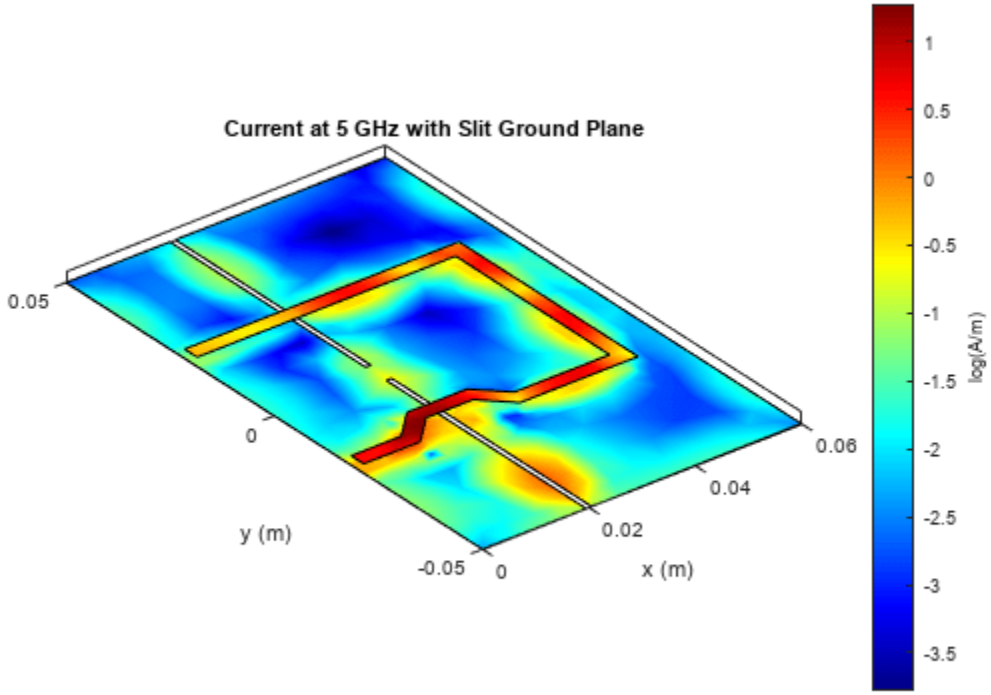
```
figure;
show(pcb);
```



The slit in the ground plane is such that the trace above crosses the slit perpendicularly as shown in the figure.

Use the `current` function to plot the current distribution at 5 GHz. As there is a slit in the ground plane the currents cannot return directly beneath the trace and takes a longer route to return through the small joint where both the ground planes are attached.

```
figure;
current(pcb,5e9,'scale','log');
title('Current at 5 GHz with Slit Ground Plane');
```



The result shows that the return currents take a longer route and might interfere with the components placed in that area and introduce noise at that component. So even if the ground is separated as an analog and digital ground, the routing of the traces is very important to design a PCB with less crosstalk and noise.

References

1 Mark Fortunato, Successful PCB Grounding with Mixed-Signal Chips - Follow the Path of Least Impedance

2 Henry W. Ott, Partitioning and Layout of a Mixed-Signal PCB.

Introduction to Equal and Unequal Split Wilkinson Power Splitter

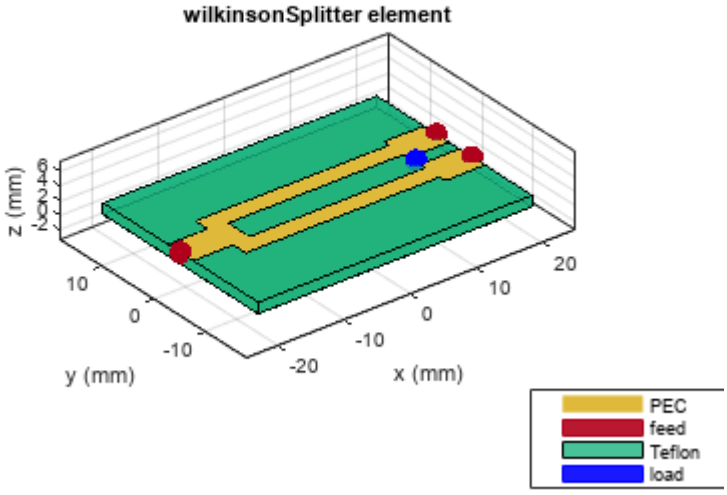
This example shows you how to design, analyze, and compare an equal and unequal Wilkinson splitter.

A Wilkinson power splitter is a device that is matched at all ports. This splitter is lossless when excited at the input port and the output ports are isolated. You can also make this splitter to perform arbitrary power division which is the unequal Wilkinson splitter.

Equal Split Wilkinson with Rectangular Shape

Use the `wilkinsonSplitter` object to create an equal split Wilkinson splitter. For this case the reference impedance is 50 Ohms. The split lines have an impedance of 70.7 ohms and the Isolation Resistor between the two output ports is 100 Ohms. The split lines have the same width and have a length equal to a quarter wavelength. The port lines have an impedance of 50 Ohms.

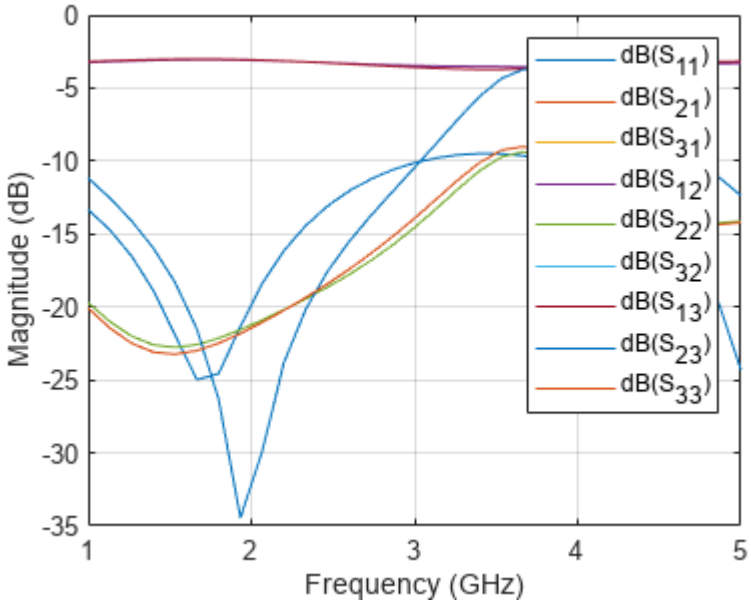
```
splitter = wilkinsonSplitter;
figure;
show(splitter);
```



The Shape property on the wilkinson splitter is used to create the geometry with rectangular bends or circular rings. The default shape property is Rectangular which creates split lines with rectangular bends.

Use the `sparameters` function to compute the S-parameters and plot it using the `rfplot` function.

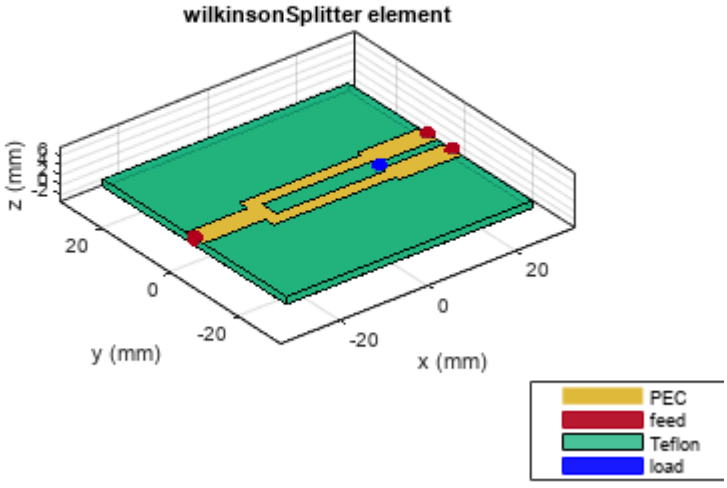
```
spar = sparameters(splitter,linspace(1e9,5e9,31));
figure;
rfplot(spar);
```



The result shows an S11 value of -25 dB at 1.8 GHz. The S21 and S31 are overlapping each other with a value of -3.1 dB. This overlap indicates an equal power division to the output ports. The Isolation is also higher than -25 dB at 1.8 GHz. The output ports are matched over the entire simulated frequency range and S22 and S33 value is higher than -25 dB at 1.8 GHz.

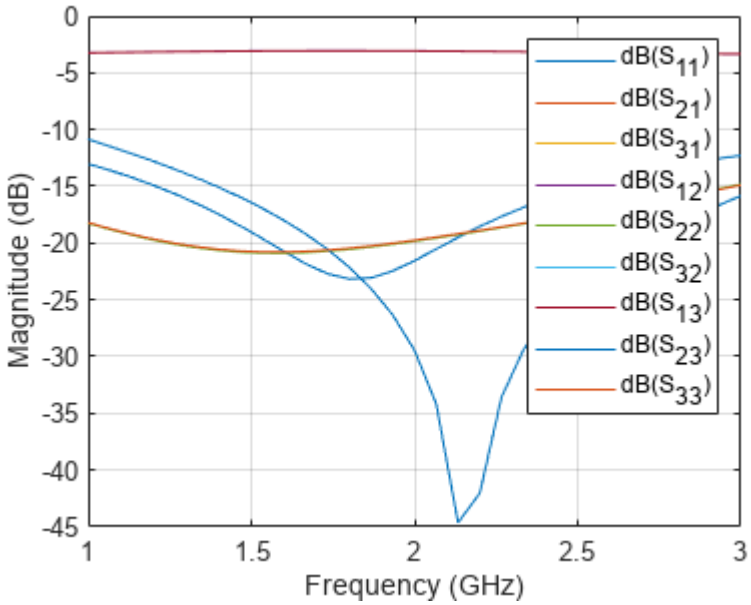
Use the `design` method to design Wilkinson splitter at 2 GHz and visualize it.

```
splitter = design(splitter,2e9);
figure;
show(splitter);
```



Compute and plot the S-parameters of the 2 GHz splitter.

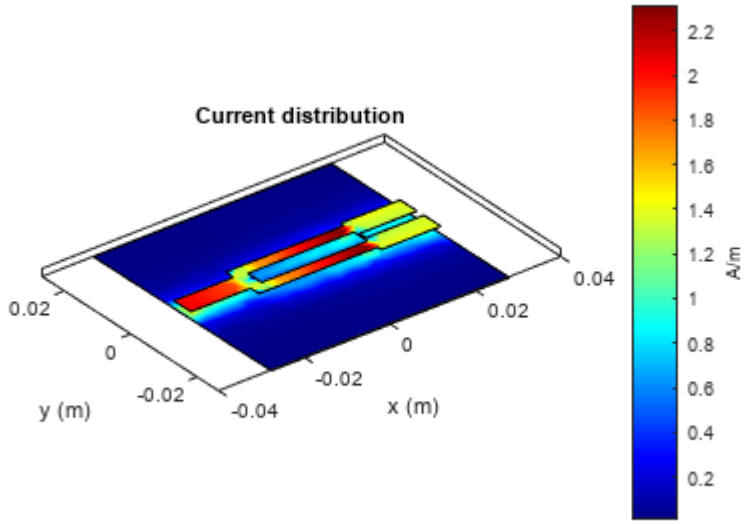
```
spar = sparameters(splitter,linspace(1e9,3e9,31));
figure;
rfplot(spar);
```



The result shows an S11 value of -30 dB close to the design frequency of 2 GHz. The S21 and S31 overlaps each other indicating an equal power division to output ports. The isolation is also higher than -30 dB at the design frequency. The output ports are matched over the entire simulated frequency range and the S22 and S33 value is higher than -20 dB at the design frequency.

Use the `current` function to plot the current distribution on the surface of the Wilkinson power splitter.

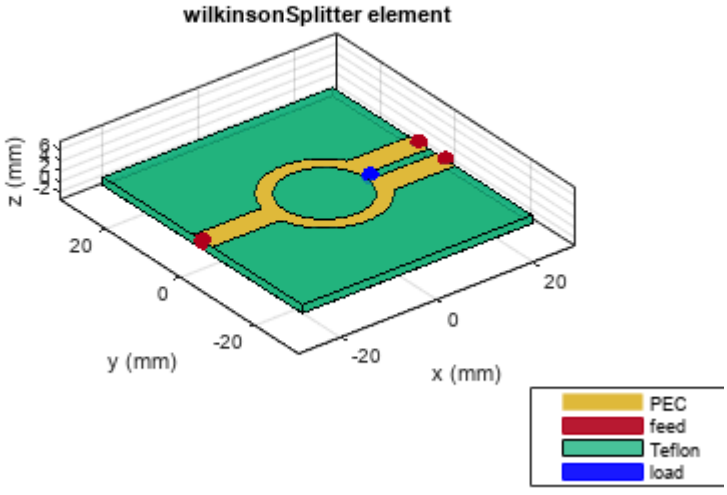
```
figure;
current(splitter,2e9);
```



Equal Split wilkinson with Circular Shape

Use the Shape property of the Wilkinson to change the splitter shape to Circular.

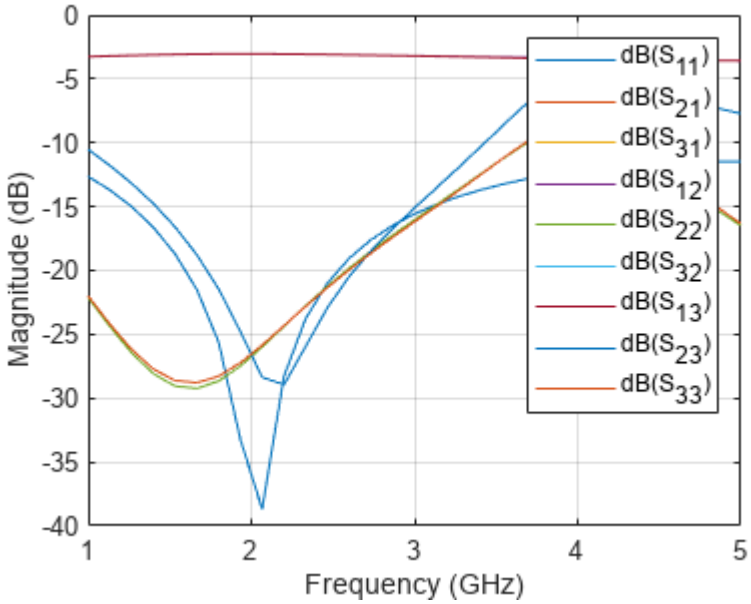
```
splitter.Shape = 'Circular';
figure;
show(splitter);
```



For a reference impedance of 50 Ohms, the input PortLine and two output PortLines have an impedances of 50 ohms. Both the SplitLines have the same width and the same impedance of 70.7 ohms. The resistance between the two lines is represented by the 100 ohm load. Since the Shape property is Circular, the SplitLines are created using an annular ring.

Compute and plot the S-parameters of the splitter.

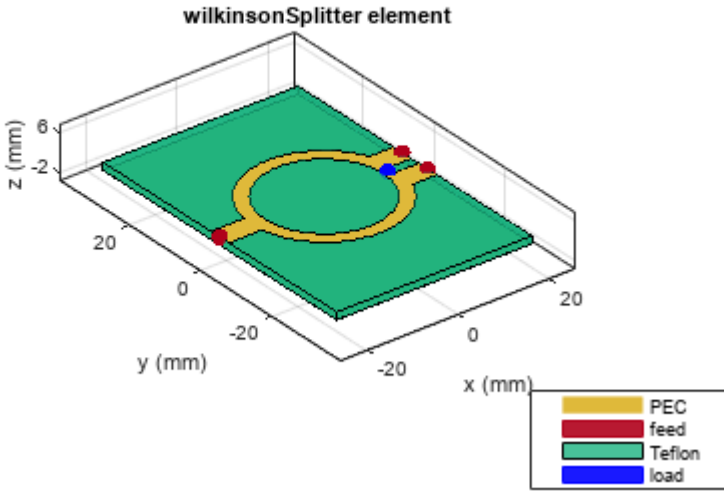
```
spar = sparameters(splitter,linspace(1e9,5e9,31));
figure;
rfplot(spar);
```



The result shows an S11 value of -25 dB at 2 GHz. The S21 and S31 overlap each other with a value of -3.1 dB at the resonant frequency. This indicates an equal power division to output ports. The Isolation is also better than -25 dB at 1.8 GHz. The output ports are matched over the entire simulated frequency range and the S22 and S33 value is higher than -25 dB at 1.8 GHz.

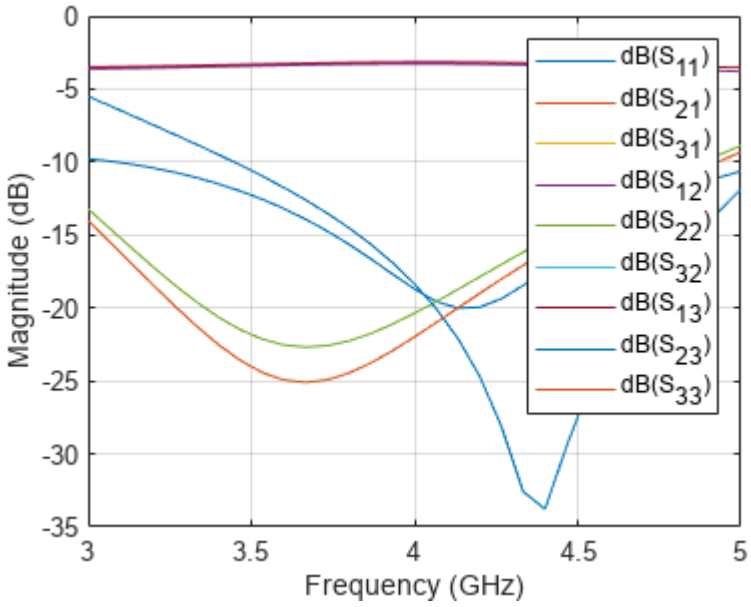
Design the Wilkinson splitter at 4 GHz.

```
splitter = design(splitter,4e9);
figure;
show(splitter);
```



Compute and plot the S-parameters.

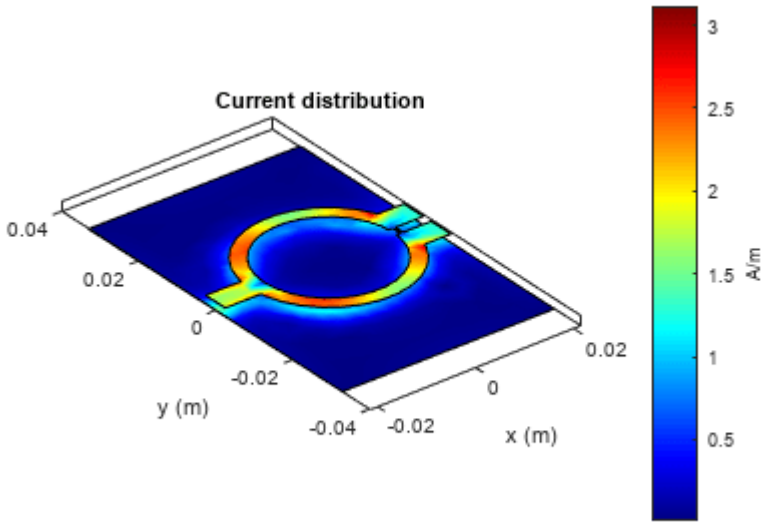
```
spar = sparameters(splitter,linspace(3e9,5e9,31));
figure;
rfplot(spar);
```



The result shows an S11 value of -20 dB close to the design frequency of 4 GHz. The S21 and S31 overlap on each other indicating an equal power division to output ports. The Isolation is also better than -20 dB at the design frequency. The output ports are matched over the entire simulated frequency range and the S22 and S33 value is -20 dB at the design frequency.

Compute and plot the current distribution on the surface of the splitter.

```
figure;
current(splitter,4e9);
```

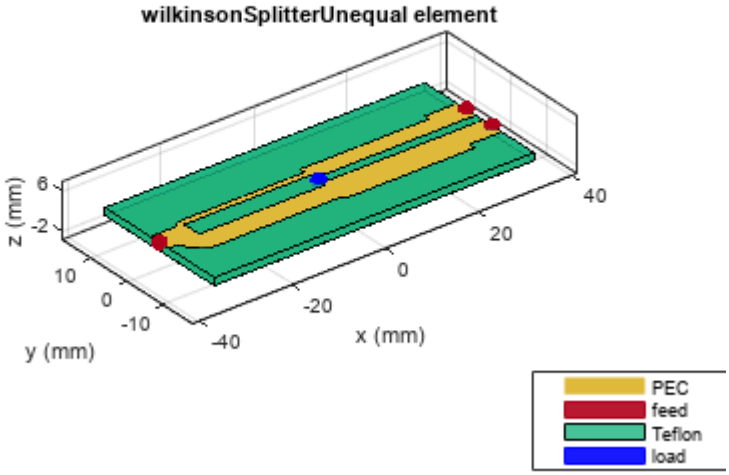


Design and Analysis of the Unequal Split wilkinson Power Splitter

The unequal Wilkinson splitter divides the power unequally to the output ports. Because of the unequal division, the impedance of both the arms of the power splitter and width of the split lines are different. Two more microstrip lines are used to match the split lines to the reference impedance. Both the SplitLine and the MatchLine are quarter wavelength long.

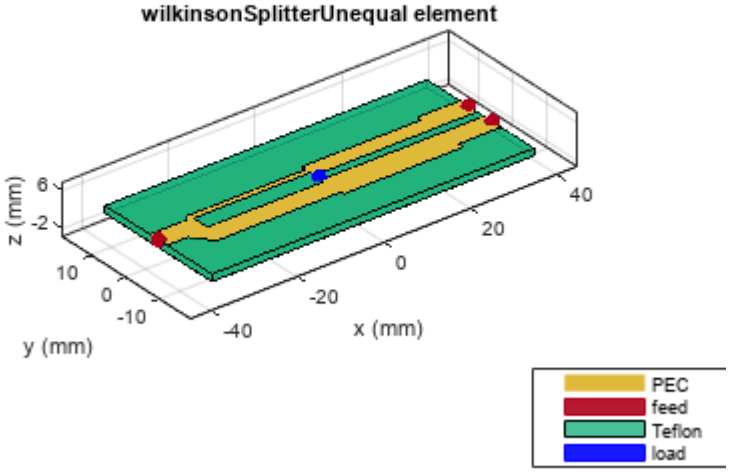
Use the wilkinsonSplitterUnequal object to create a Wilkinson splitter and use the show function to visualize it.

```
splitter = wilkinsonSplitterUnequal;
figure;
show(splitter);
```



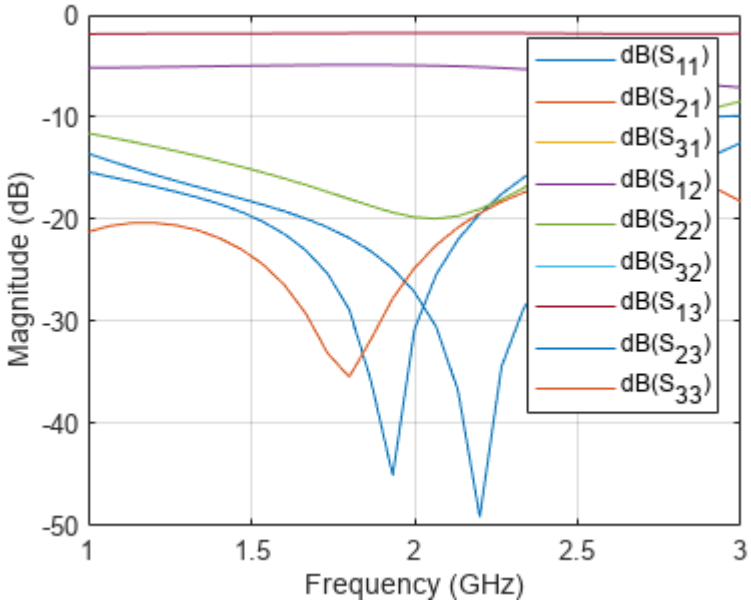
Design the Wilkinson splitter at 2 GHz and specify the power ratio between the ports as 2.

```
splitter = design(splitter,2e9,'PowerRatio',2);
figure;
show(splitter);
```



Compute and plot the S-parameters.

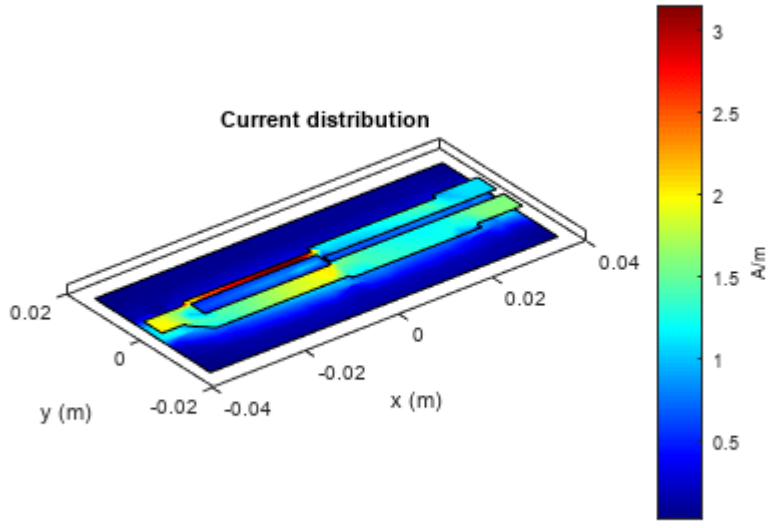
```
spar = sparameters(splitter,linspace(1e9,3e9,31));
figure;
rfplot(spar);
```



The result shows an S11 value higher than -30 dB close to the design frequency of 2 GHz. The S21 value is close to the expected value of -1.76 dB and S31 value is close to the expected value of -4.77 dB. These values indicate an unequal division to output ports. The Isolation is also higher than -30 dB at the design frequency. The output ports are matched over the entire simulated frequency range and S22 and S33 value is better than -20 dB at the design frequency.

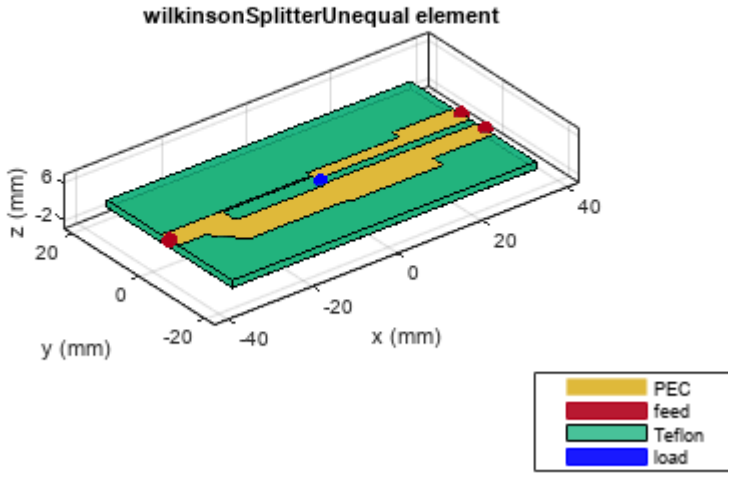
Compute and plot the current distribution on the surface of the Wilkinson power splitter.

```
figure;
current(splitter,2e9);
```



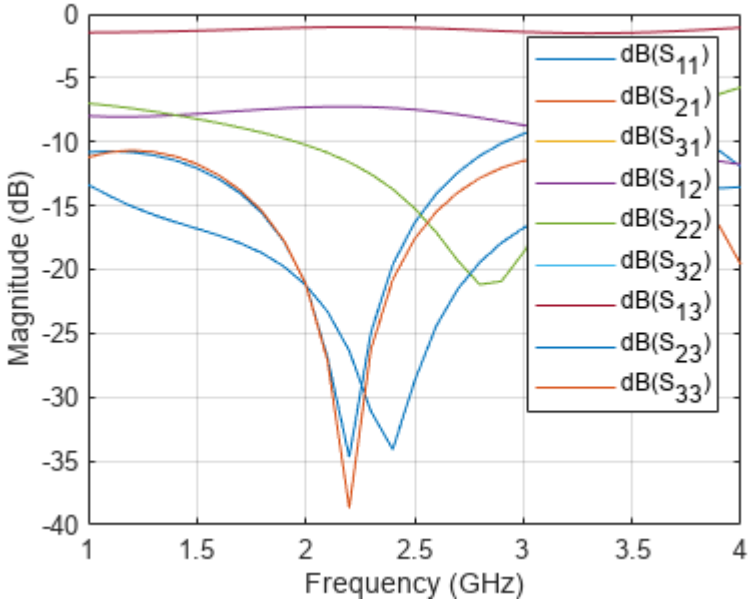
Change the PowerRatio to 4 in the design method.

```
splitter = design(splitter,2.5e9,'PowerRatio',4);
figure;
show(splitter);
```



Computer and plot the S-parameters.

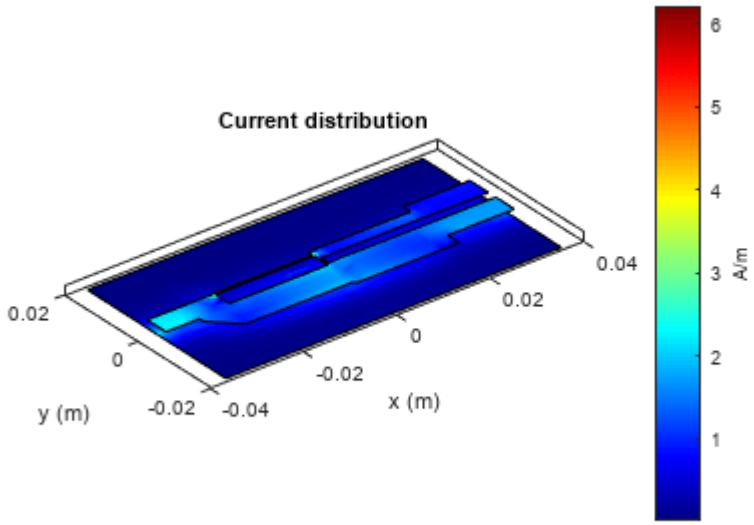
```
spar = sparameters(splitter,linspace(1e9,4e9,31));
figure;
rfplot(spar);
```



The result shows an S11 value higher than -30 dB close to the design frequency of 2.5 GHz. The S21 value is close to the expected value of -0.97 dB and S31 value is close to the expected value of -6.99 dB. These values indicate an unequal division to output ports with the power ratio of 4. The Isolation is also higher than -20 dB at the design frequency. The output ports are matched over the entire simulated frequency range and S22 and S33 value is better than -15 dB at the design frequency.

Compute and plot current distribution on the surface of the Wilkinson power splitter.

```
figure;
current(splitter,2.5e9);
```



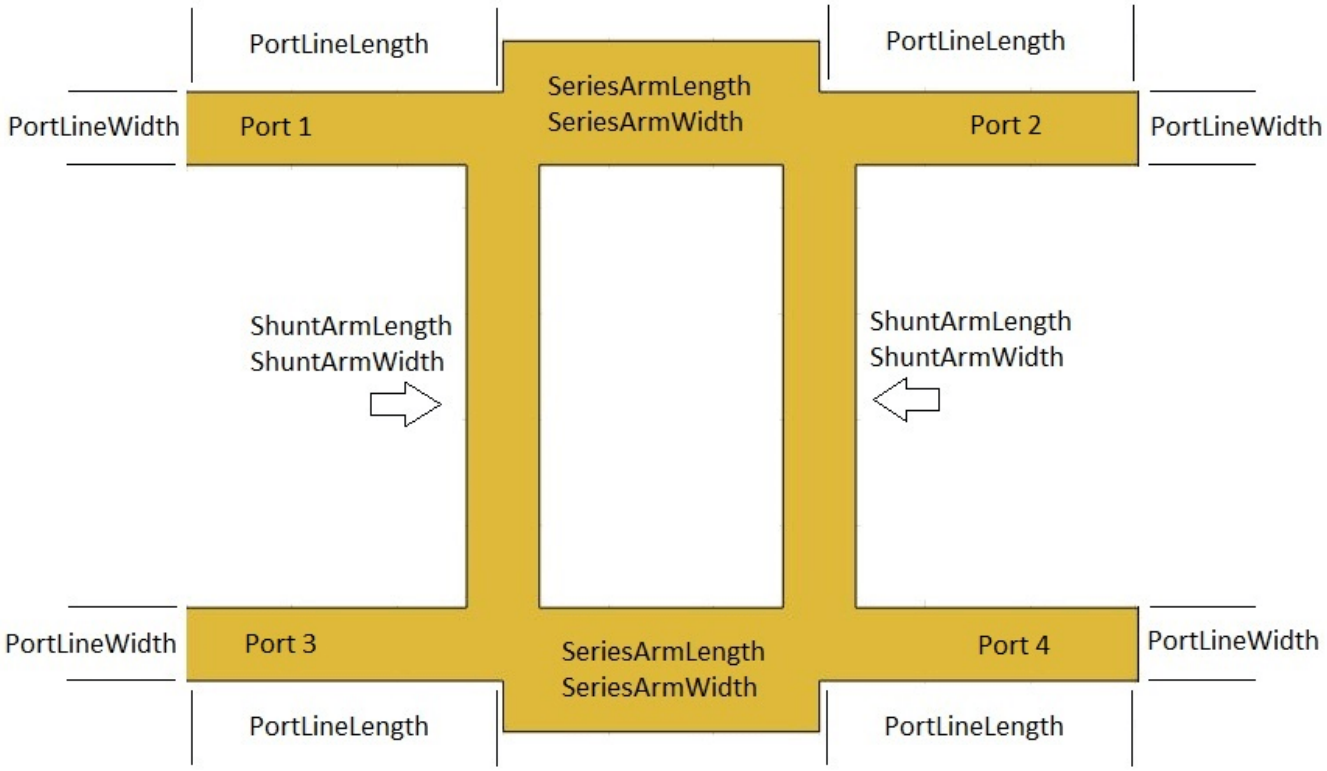
Introduction to 4-Port Couplers

This example shows how to design and analyze 4-Port Couplers.

4-Port couplers have a through port, coupled port, and an isolated port. For the equal split couplers, power is evenly split between the through port and the coupled port and the ideal coupling factor is 3 dB. No power is coupled to the isolated port. In this example, you will design and analyze a Branchline coupler and Ratrace coupler.

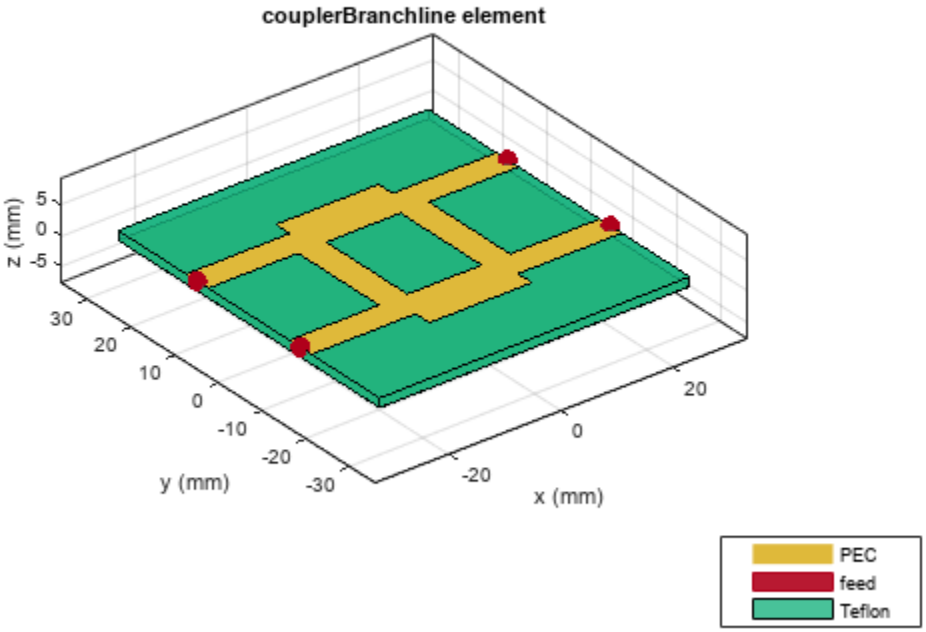
Design and Analyze Branchline Coupler

The Branchline coupler is a 3-dB coupler and divides power equally between the through and coupled arms. It is also called Quadrature Hybrid because of the 90 degree phase difference in the outputs from the through and coupler arms. With all ports matched, power entering port 1 is evenly divided between ports 2 and 4, with a 90 degree phase shift between the outputs. No power is coupled to the Port 3 (the isolated port).



Use the object couplerBranchline to create a Branchline coupler and visualize it.

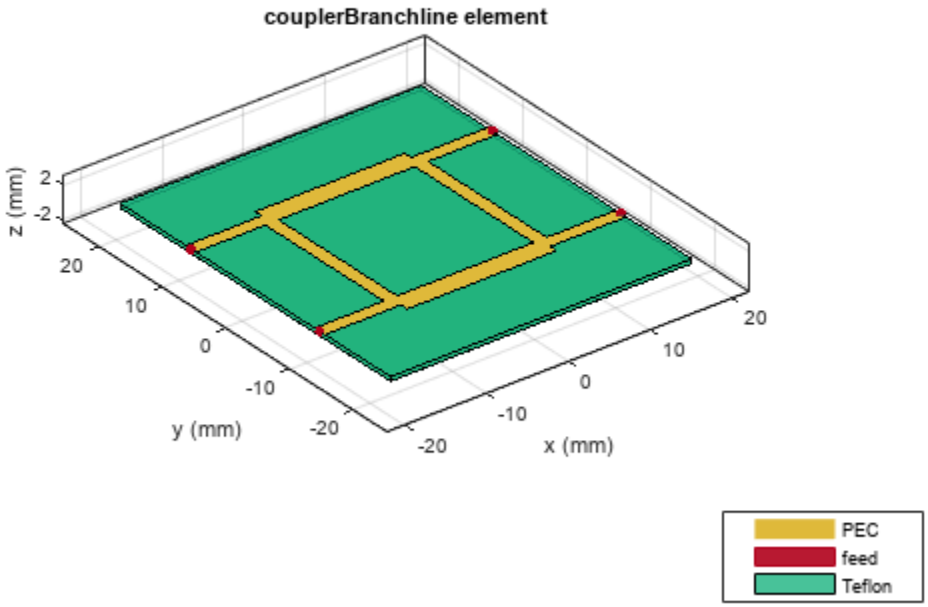
```
coupler = couplerBranchline;
figure;
show(coupler);
```



The `Height` property controls the substrate thickness for the RF PCB Catalog elements. Changing the Height property changes the substrate thickness.

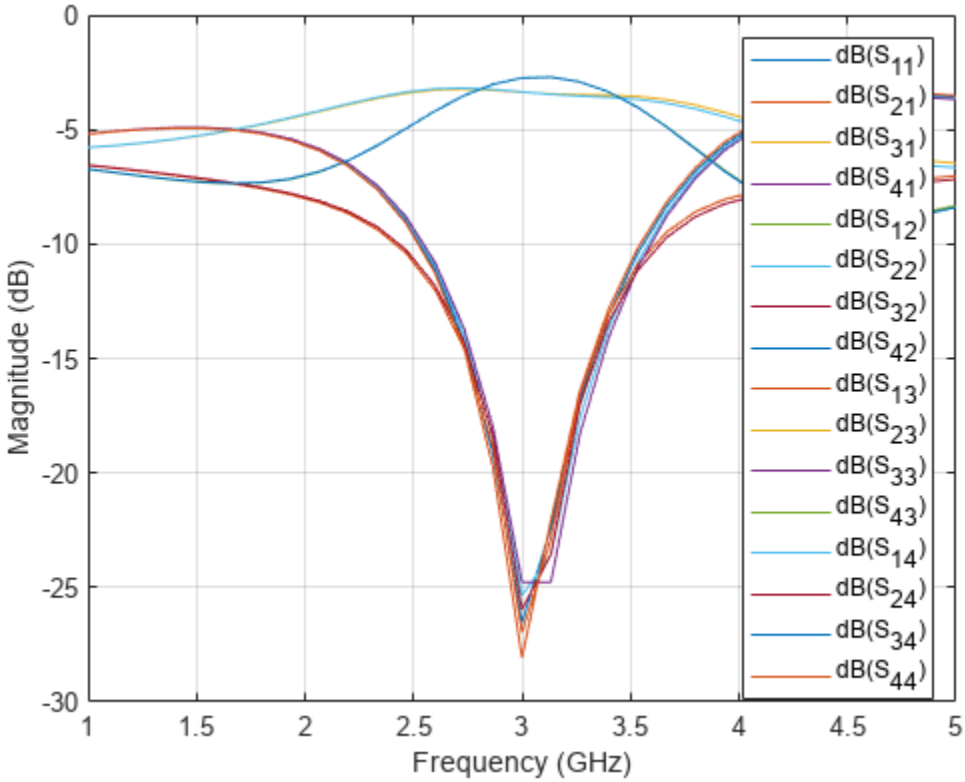
Change the `Height` property of the Branchline coupler to 0.508 mm . Use the `design` function to design the Branchline coupler at 3 GHz and visualize it.

```
coupler.Height = 0.508e-3;
coupler = design(coupler,3e9);
figure;
show(coupler);
```



Use the `sparameters` function to calculate the S-Parameters and plot it using the `rfplot` function.

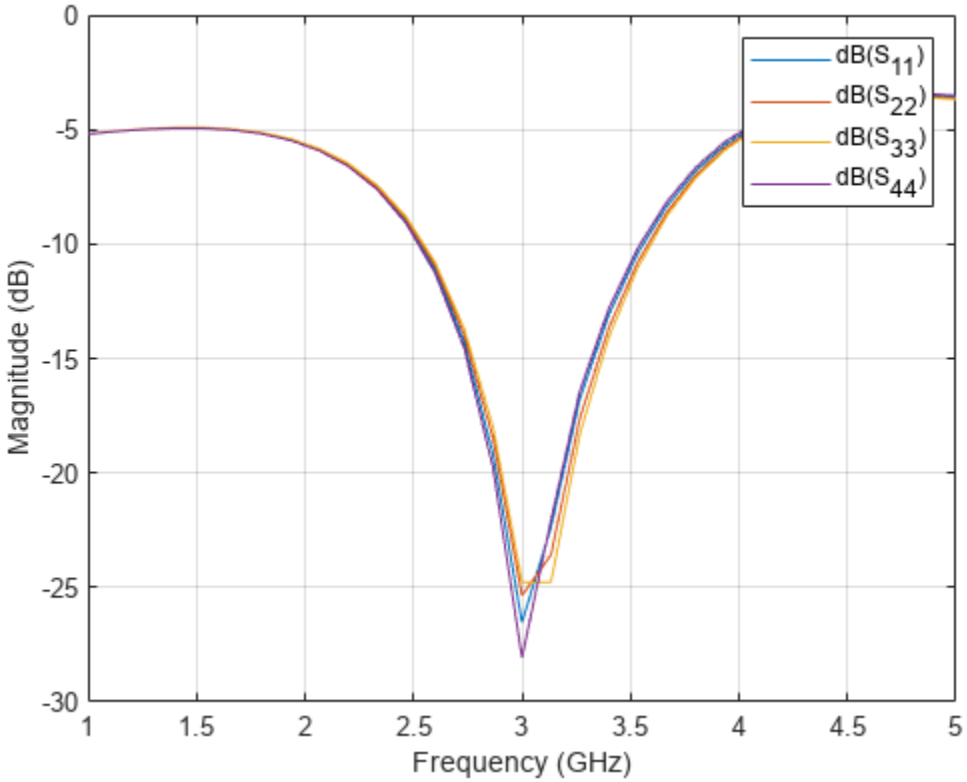
```
spar = sparameters(coupler,linspace(1e9,5e9,31));
figure
rfplot(spar)
```



As there are 16 curves in the result, let us analyze the result in three different parts.

Plot Return Loss Values at All Ports

```
figure
rfplot(spar,'diag')
```

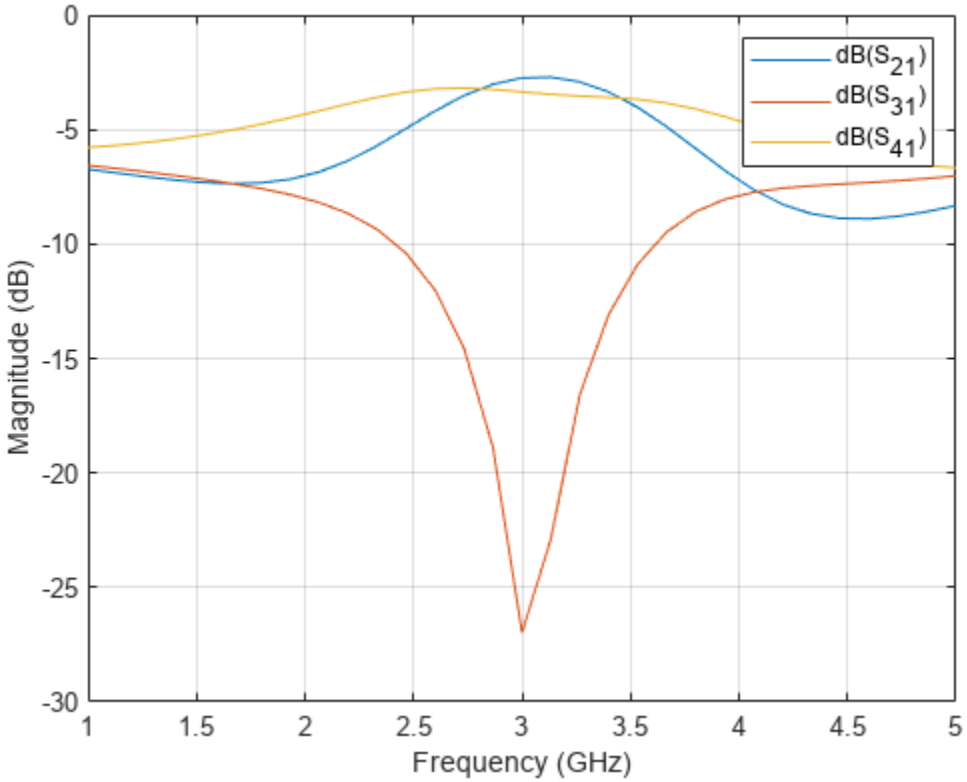


The result shows that all the ports are a good match at 3 GHz and the value of S11, S22, S33, and S44 is close to -30 dB.

Plot S-Parameters When Input is Applied at Port 1

Analyze the values of S21, S31, and S41 to understand the behavior of Branchline coupler when input is applied to port 1.

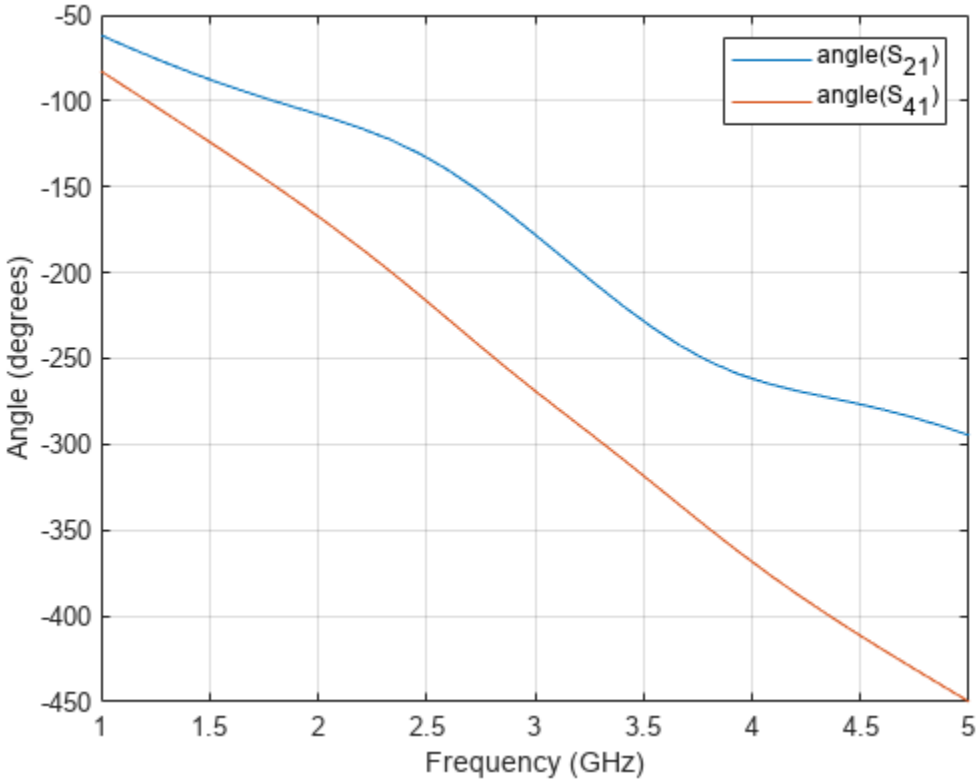
```
figure
rfplot(spar,2:4,1)
```



The result shows that the power is equally divided between Port 2 and Port 4. Port 3 is isolated.

Plot the phase of the S21 and S41 to understand the phase different between the output ports.

```
figure, rfplot(spar,[2 4],1,'angle')
```

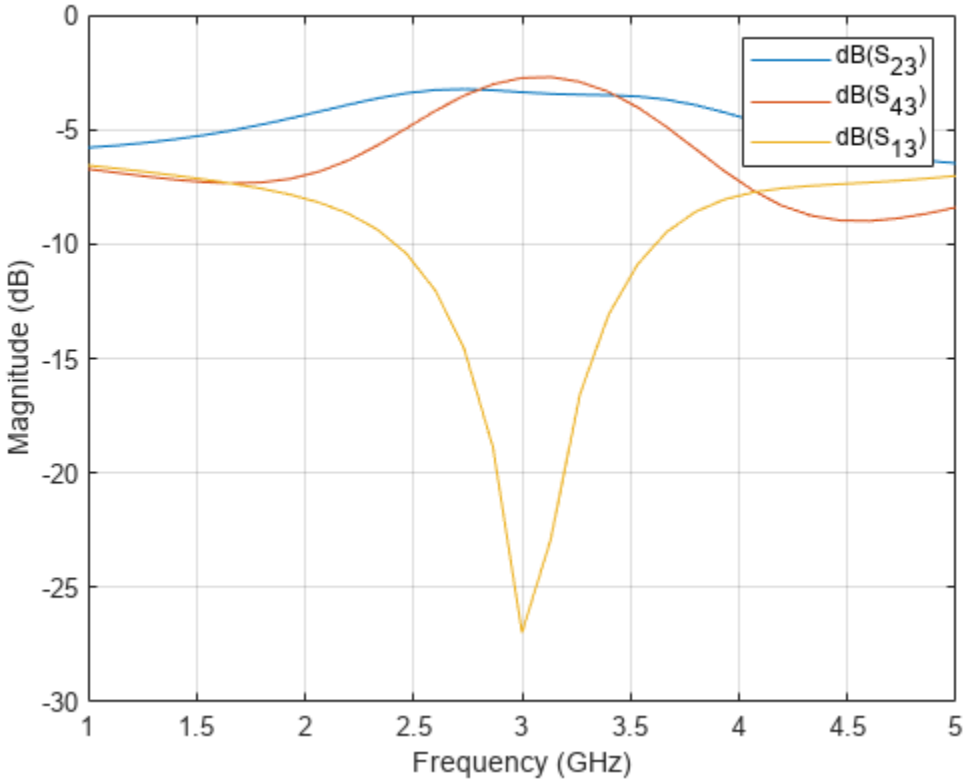


The result shows that the phase difference between the output ports is close to 90 degrees at the design frequency. Hence, the outputs at Port 2 and Port 4 is out of phase by 90 degrees when the input is applied to Port 1.

Plot S-Parameters When Input is Applied at Port 3

Analyze the values of S13, S23, and S43 to understand the behavior of Branchline coupler when input is applied to Port 3.

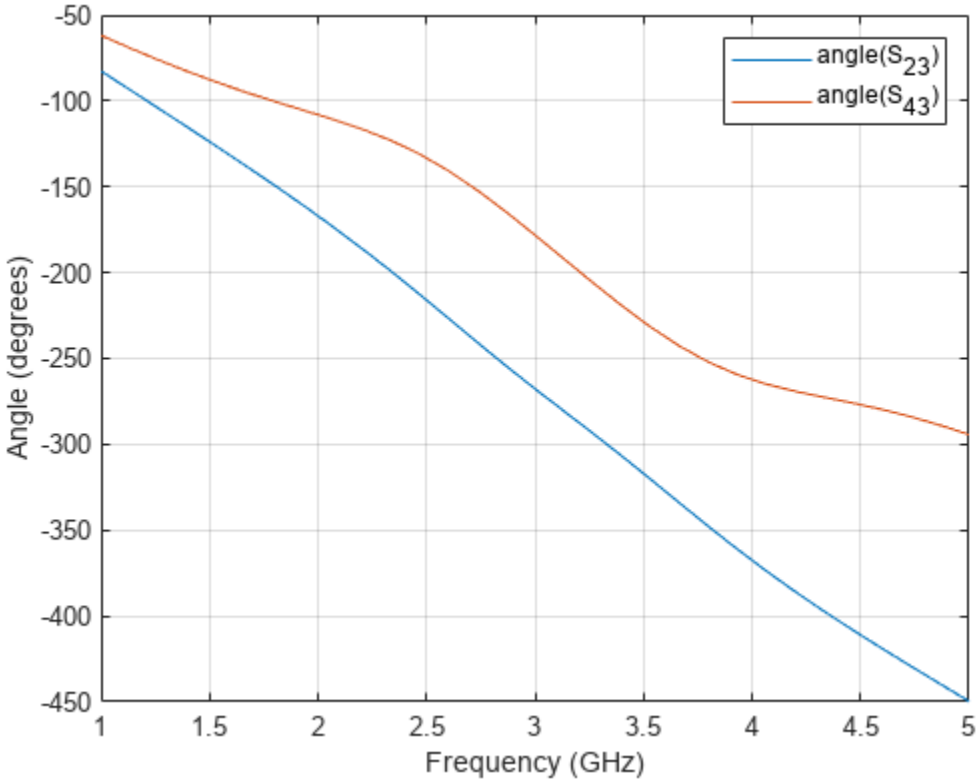
```
figure
rfplot(spar,[2 4 1],3)
```



The result shows that the power is equally divided to Port 2 and Port 4. Port 1 is isolated.

Plot the phase of the S23 and S43 to understand the phase different between the output ports.

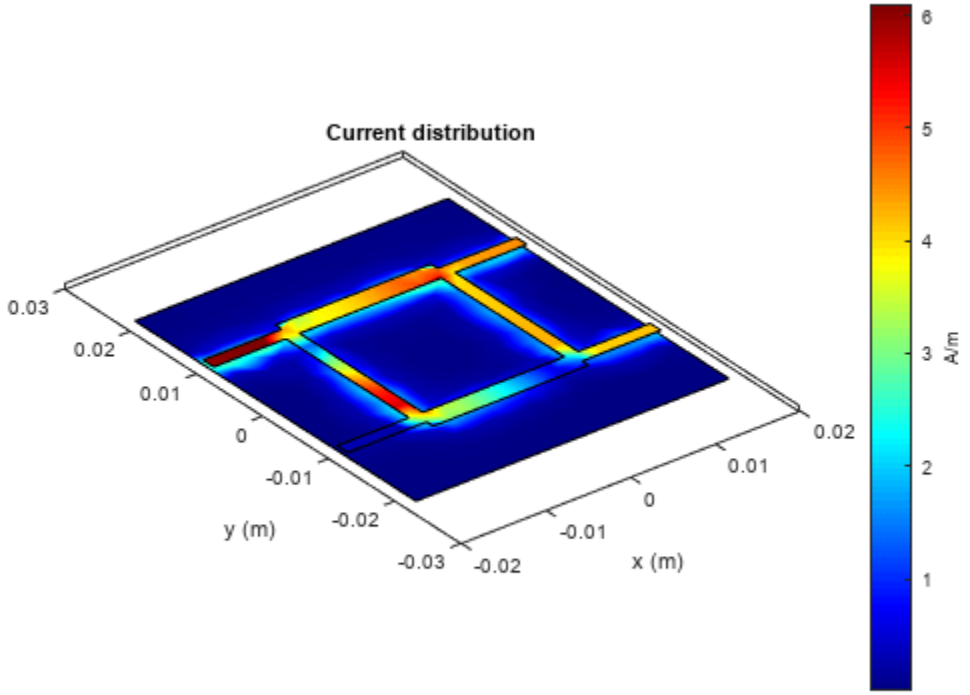
```
figure
rfplot(spar,[2 4],3,'angle')
```



The result shows that the phase difference between the output ports is close to 90 degrees at the design frequency of 3 GHz. Hence, when the input is applied to Port 3, the output at Port 2 and Port 4 is out of phase by 90 degree.

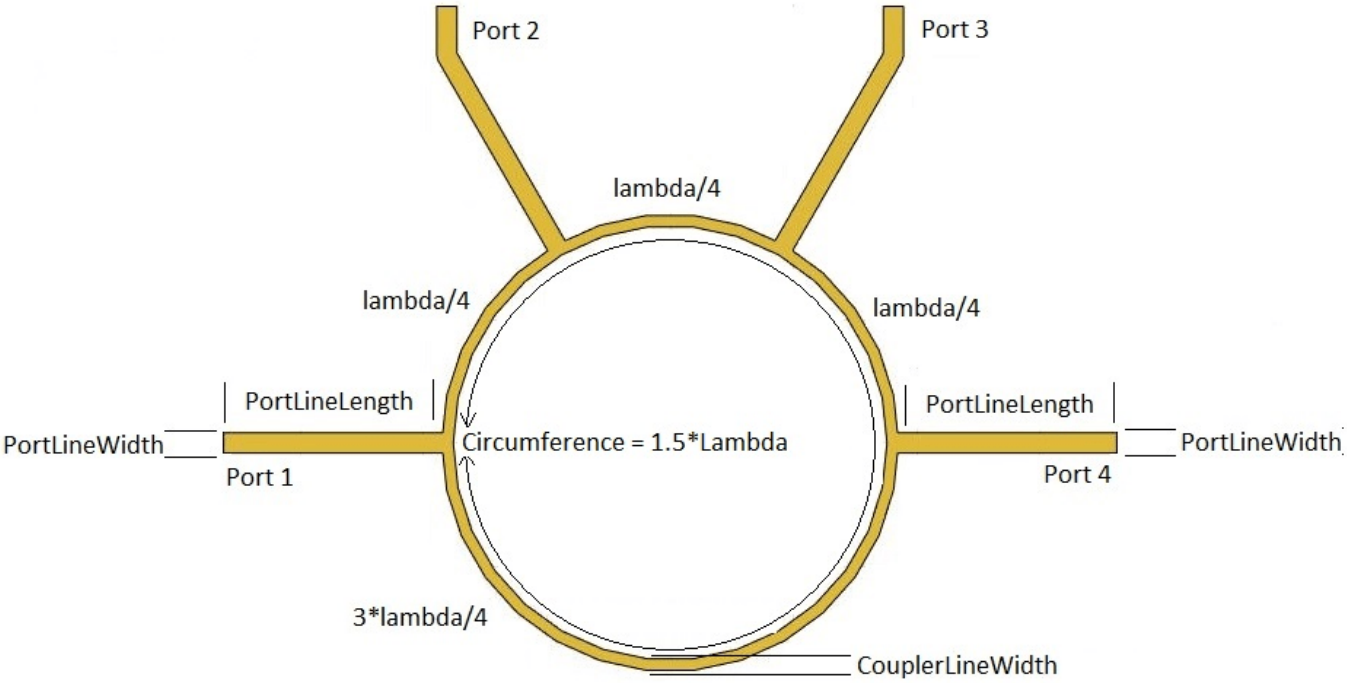
Use the `current` method to visualize the current distribution on the Branchline coupler

```
figure
current(coupler,3e9)
```



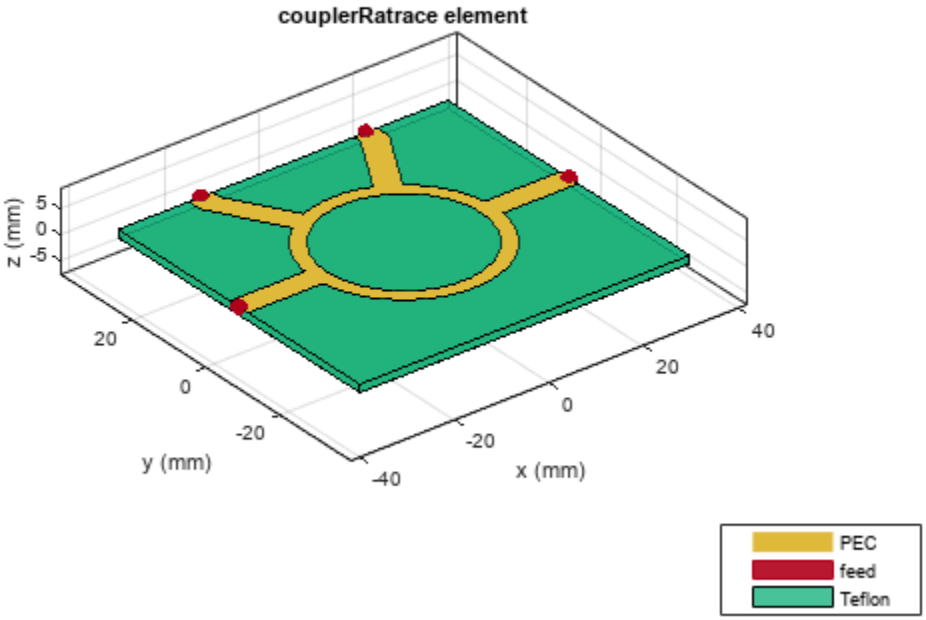
Design and Analyze Ratrace Coupler

The Ratrace Coupler is a 4-port coupler and is also called a 180 deg ring hybrid as the output ports exhibit a phase shift of 180 deg. With reference to the 180 deg hybrid shown below, a signal applied at Port 3 will be evenly split into two in-phase components at Ports 2 and 4. Port 1 is isolated. If the input is applied at Port 1, signal is equally split into two components with a 180 deg phase difference at Ports 2 and 4. Port 3 is isolated. When operated as a combiner, with input signals applied at Ports 2 and 4, the sum of the inputs forms at Port 3, while the difference forms at Port 1. Hence Port 3 and 1 are referred to as the sum and difference ports respectively.



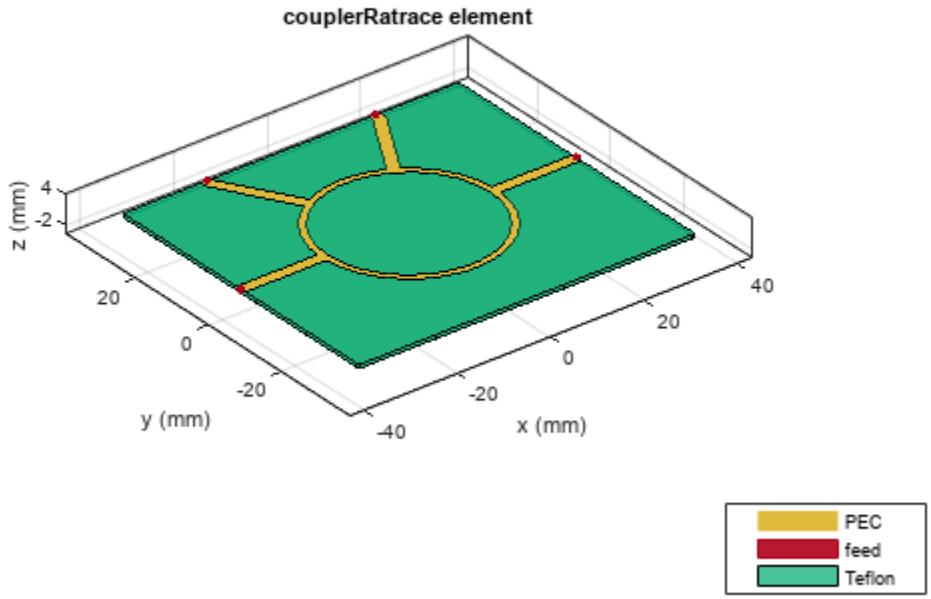
Use the couplerRatrace object to create a Ratrace coupler and visualize it.

```
coupler = couplerRatrace;
figure
show(coupler)
```



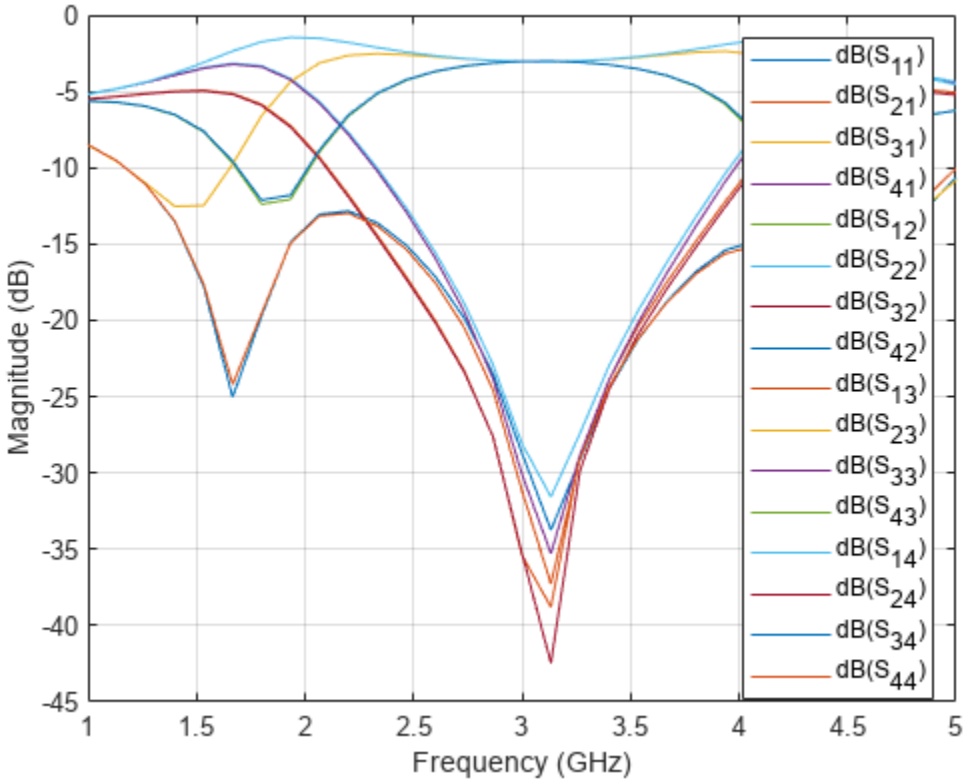
Change the Height property for the Ratrace coupler. Use the design function to design the Ratrace coupler at 3 GHz and visualize it.

```
coupler.Height = 0.762e-3;
coupler = design(coupler,3e9);
figure
show(coupler)
```



Use the `sparameters` function to calculate the S-Parameters for the Ratrace coupler and plot it using the `rfplot` function.

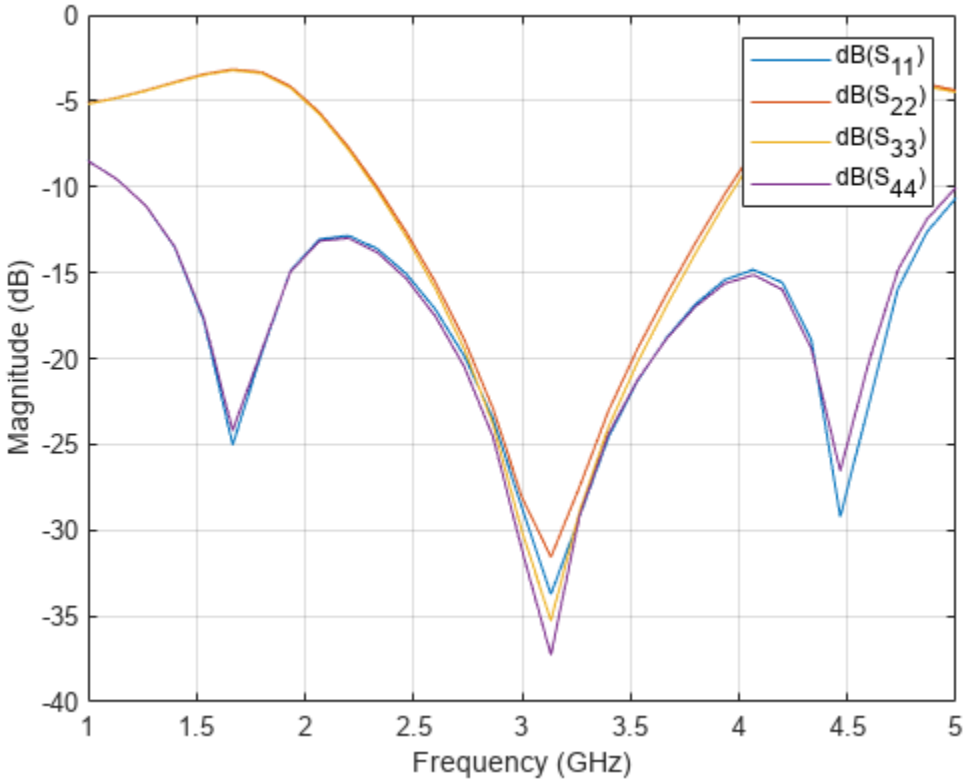
```
spar = sparameters(coupler,linspace(1e9,5e9,31));
figure
rfplot(spar)
```



As there are 16 curves in the result, let us analyze the result in three different parts.

Plot Return Loss Values at All Ports

```
figure
rfplot(spar,'diag')
```

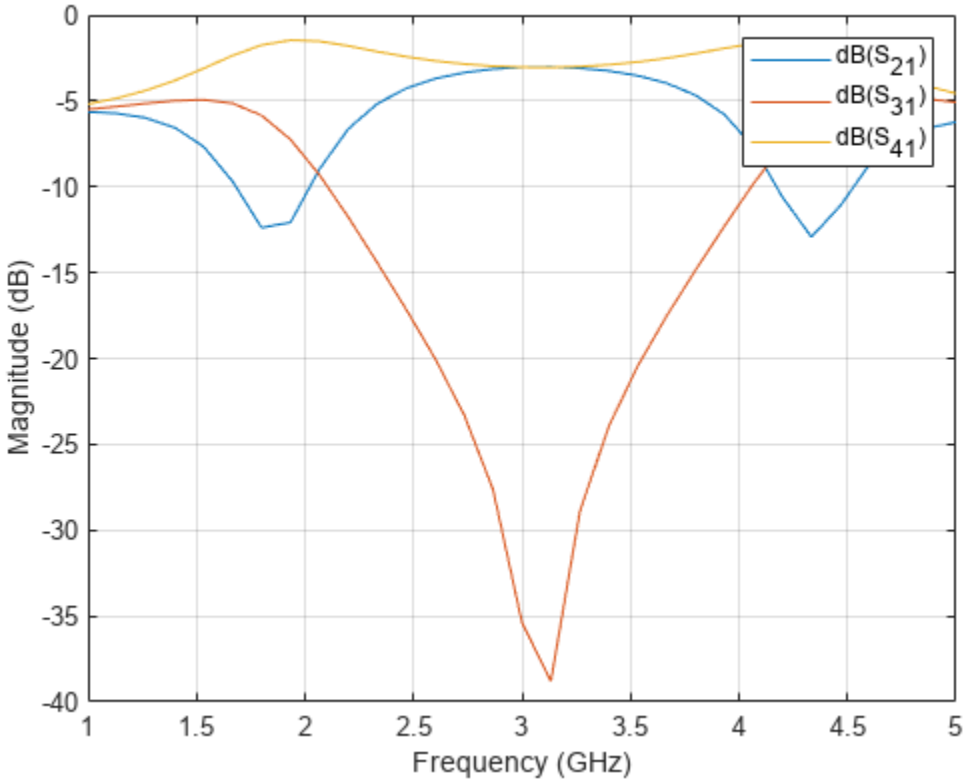


The result shows that all the ports are a good match at 3 GHz and the values of S11, S22, S33, and S44 is higher than -30 dB.

Plot S-Parameters When Input is Applied at Port 1

Analyze the values of S21, S31, and S41 to understand the behavior of Ratrace coupler when input is applied to port 1.

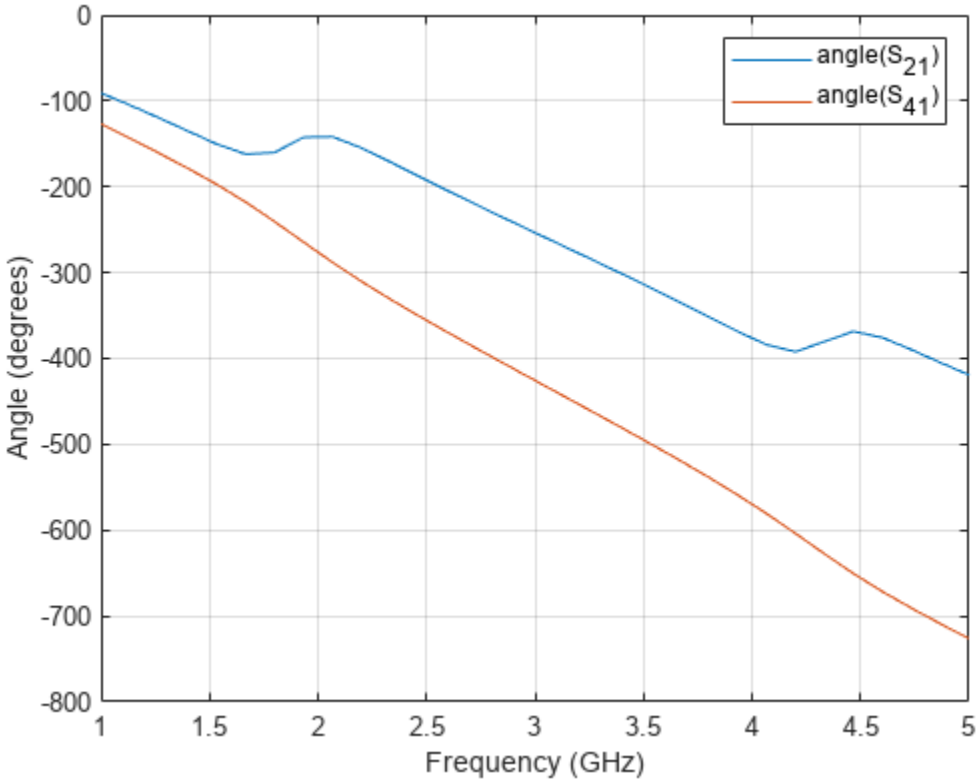
```
figure
rfplot(spar,2:4,1)
```



The result shows that the power is equally divided between Port 2 and Port 4. Port 3 is isolated when the input is applied to Port 1.

Plot the phase of S21 and S41 to understand the phase difference between the output ports.

```
figure
rfplot(spar,[2 4],1,'angle')
```

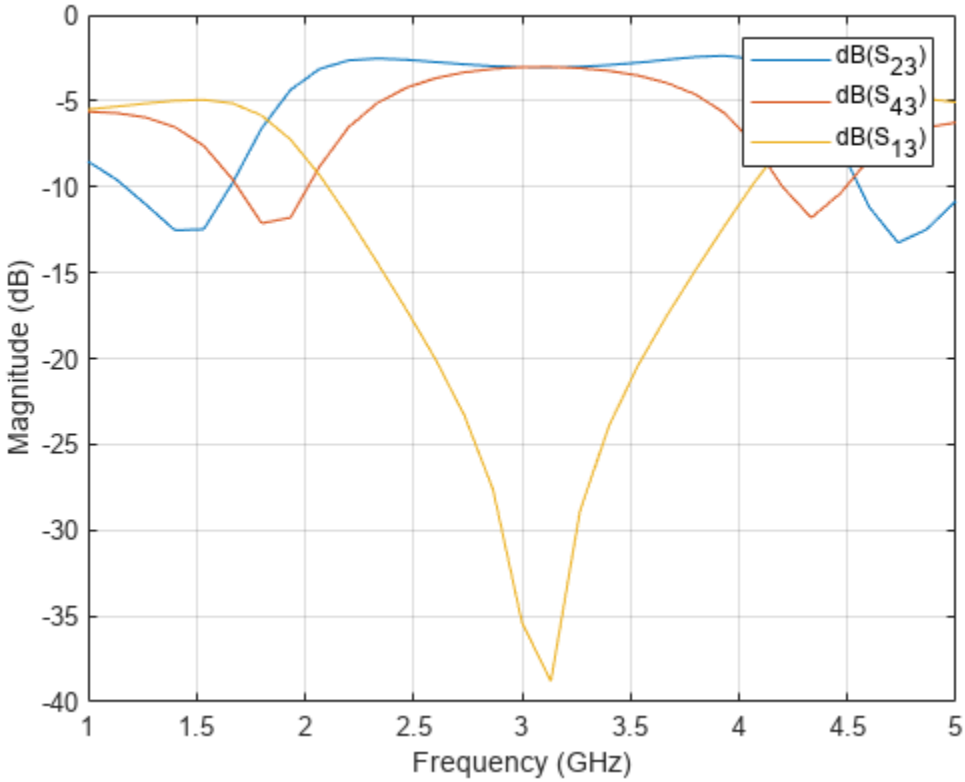


The result shows that the phase difference between the output ports is close to 180 degrees at the design frequency. Hence, when the input is applied at Port 1, the output at Port 2 and Port 4 is out of phase by 180 degrees.

Plot S-Parameters When Input is Applied at Port 3

Analyze the values of S23, S43, and S13 to understand the behavior of Ratrace coupler when input is applied to Port 3.

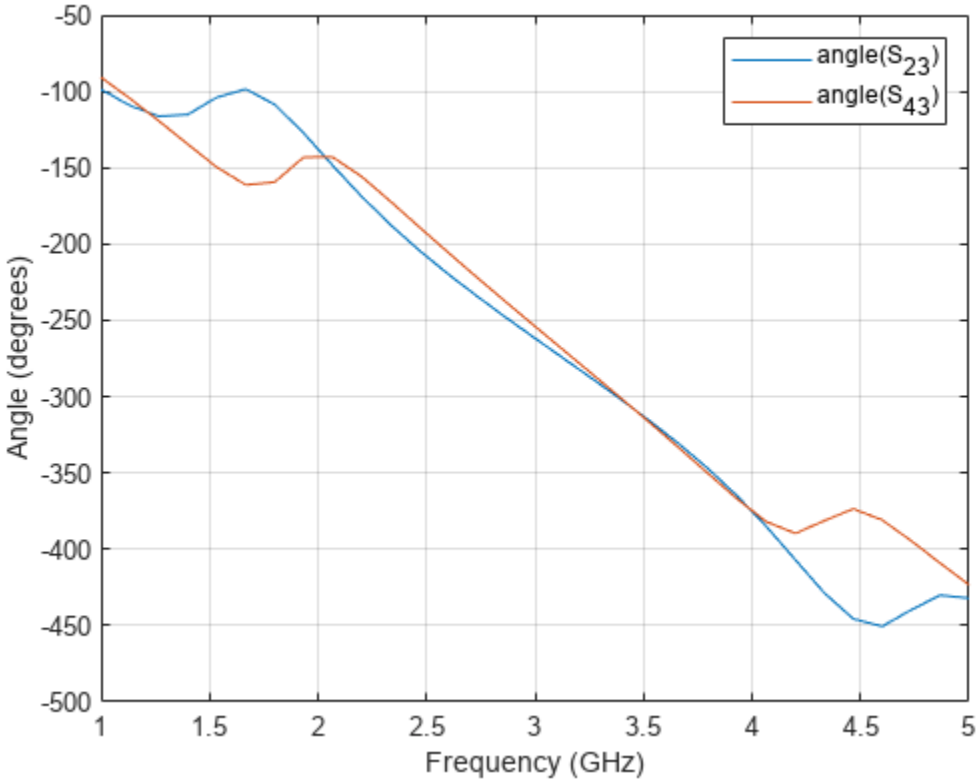
```
figure
rfplot(spar,[2 4 1],3)
```



The result shows that the power is equally divided between Port 2 and Port 4. Port 1 is isolated when the input is applied to Port 3.

Plot the phase of the S23 and S43 to understand the phase difference between the output ports.

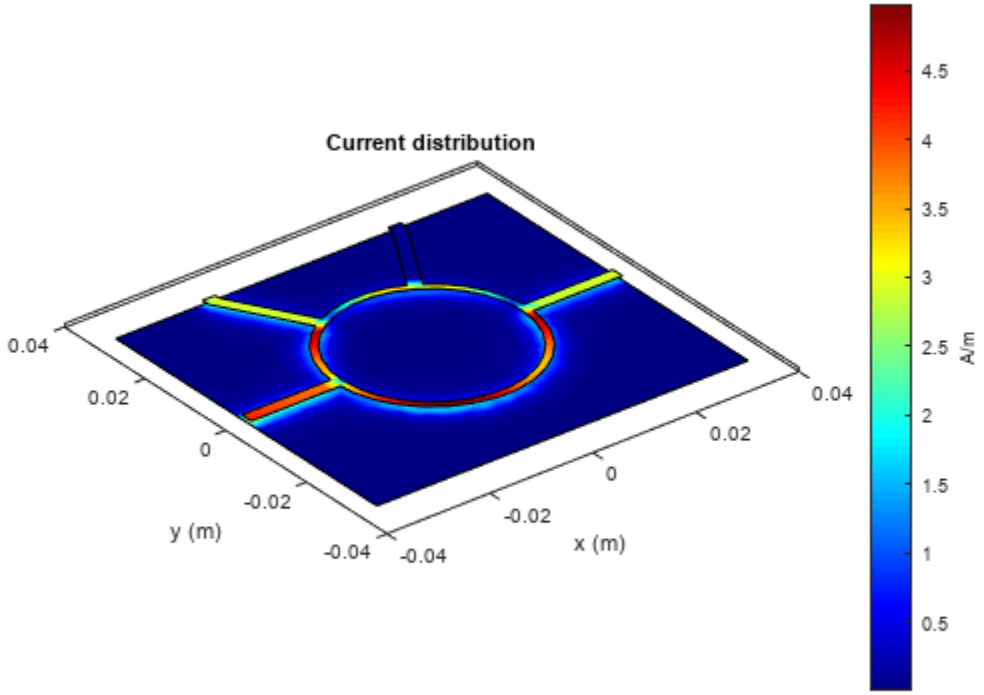
```
figure
rfplot(spar,[2 4],3,'angle')
```



The result shows that the phase difference between the output ports is close to 0 degrees at the design frequency of 3 GHz. Hence, when the input is applied at Port 3, the output at Port 2 and Port 4 is in phase.

Use the `current` method to visualize the current distribution on the Ratrace coupler

```
figure
current(coupler,3e9)
```

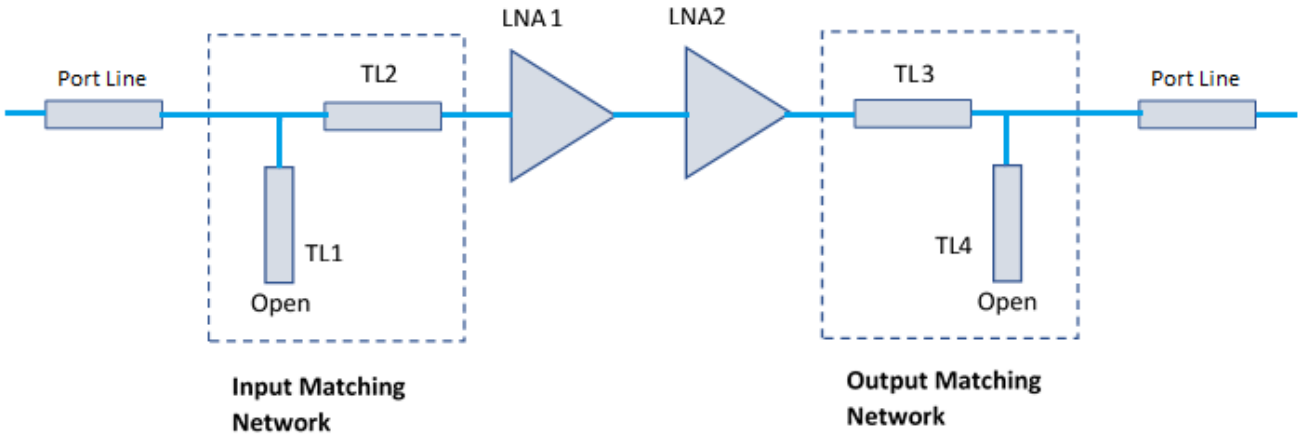

Current distribution
0.04
0.02
0
-0.02
-0.04
y (m)
-0.04
-0.02
0
0.02
0.04
x (m)
4.5
4
3.5
3
2.5
2
1.5
1
0.5
A/m

Design Two-Stage Low Noise Amplifier Using Microstrip Transmission Line Matching Network

This example shows how to use the RF PCB Toolbox™ shapes and the `pcbComponent` object functionality to design a two-stage low noise amplifier (LNA) for a wireless local area network (WLAN) with an input and output matching network (MNW) to maximize the power delivered to the 50 ohm load.

Designing an input and output MNW is an important part of an amplifier design. The amplifier in this example has high gain and low noise. This example uses the traceTee RF PCB shape to create the input and output matching network.

The diagram shows the matching network topology used for both the input and output matching networks. The lengths and widths given in the paper[1] are for the lines TL2 and TL1 for Input MNW and TL3 and TL4 for the output MNW. Additional Port lines will be added as shown in the figure which will have same length as the series transmission lines(TL2 and TL3) so that we can connect the ports and analyze the structure. As the MNW forms the tee shape at both input and output, you can use the `traceTee` shape for creating the MNWs.



Define Microstrip Transmission Line Parameters

The microstrip transmission line parameters are chosen as follows.

Physical Height of conductor or dielectric thickness — 1.524 mm

Relative permittivity of dielectric — 3.48

Loss angle tangent of dielectric — 0.0037

Physical thickness of microstrip transmission line — 3.5 um

Create the Variables

The variables below are for creating the input and output MNWs. The `Length_Line1` and `Length_Line2` store the length of the series matching line and the length of the portline. The paper

gives the TL2 as 14.7 mm and hence the portline at input also is taken as 14.7 mm. Hence the `Length_Line1` becomes 29.4 mm. At the output MNW, the line TL3 is 22.5 mm and hence the portline at the output is also 22.5 mm. Hence `Length_Line2` becomes 45 mm. The stub lengths Length_Stub1 and Length_Stub2 are also taken from the paper[1].

```
Length_Line1 = 29.4e-3; % Length of the Input Matching Network
Length_Stub1 = 8.9e-3;  % Length of the Stub

Length_Line2 = 45e-3;   % Length of the Output Matching Network
Length_Stub2 = 5.66e-3; % Length of the Stub
Width_Line   = 3.5e-3;  % Width of the line

EpsilonR     = 3.48;    % Dielectric EpsilonR
Height       = 1.524e-3;% Height of the Substrate
LossTangent  = 0.0037;  % Loss Tangent of the Substrate

fmin   = 2e9;
fmax   = 3e9;
points = 50;
freq   = linspace(fmin,fmax,points);
```

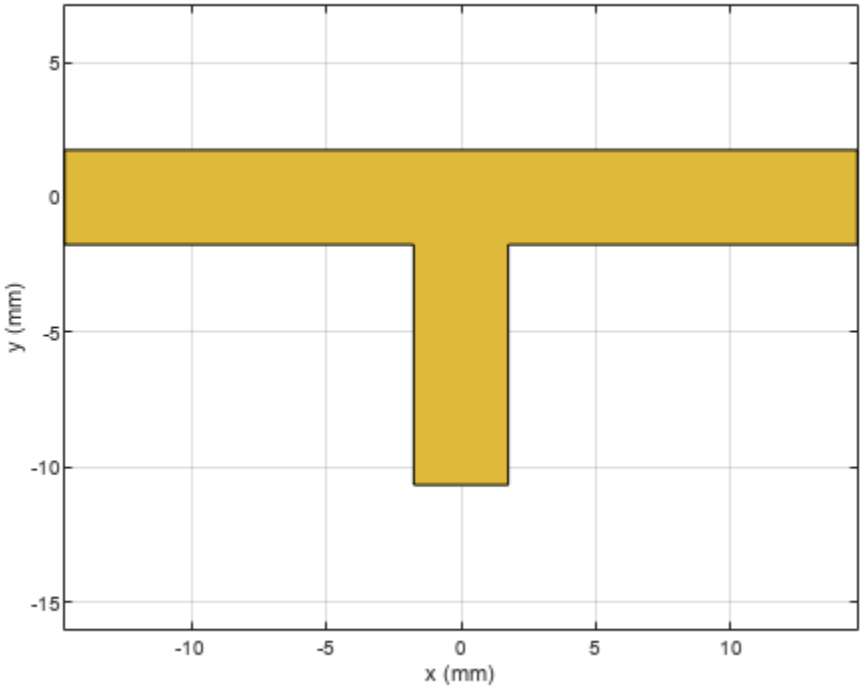
Use the `nport` function from RF Toolbox to read the sparameters of the amplifier from an s2p file for the amplifier and store it in the variables Amp1 and Amp2.

```
Amp1 = nport('f551432p.s2p');
Amp2 = nport('f551432p.s2p');
```

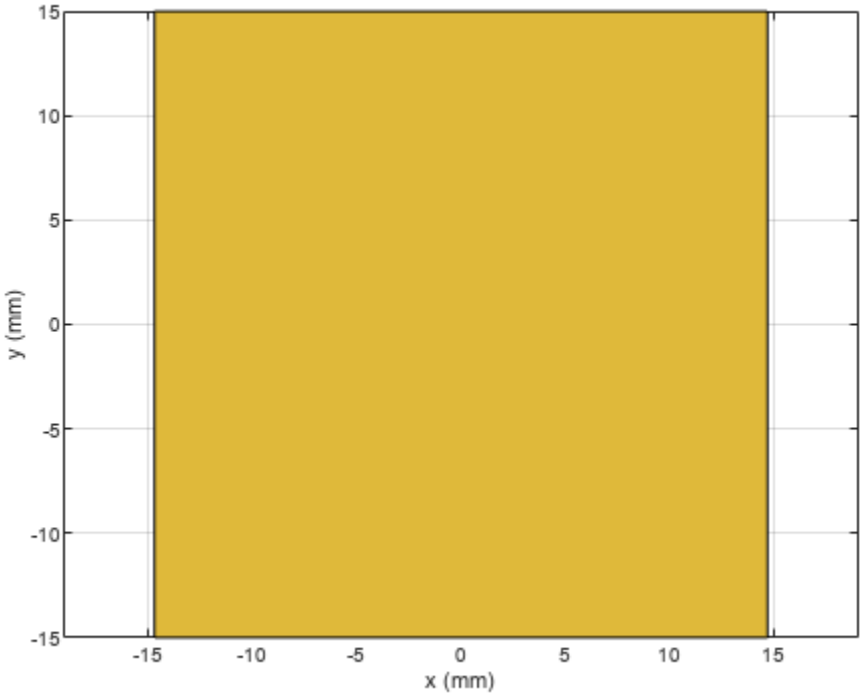
Design Input Matching Network Using tracetee shape

The input matching network consists of one shunt stub and one series microstrip transmission line. The dimensions of the series transmission line is Length_Line1 and the length of the stub is Length_Stub1. The width of the lines is Width_Line.

```
InputMatching = traceTee('Length',[Length_Line1 Length_Stub1],'Width',[Width_Line Width_Line]);
show(InputMatching);
```




```
Gnd1 = antenna.Rectangle('Length',Length_Line1,'Width',30e-3);
show(Gnd1);
```

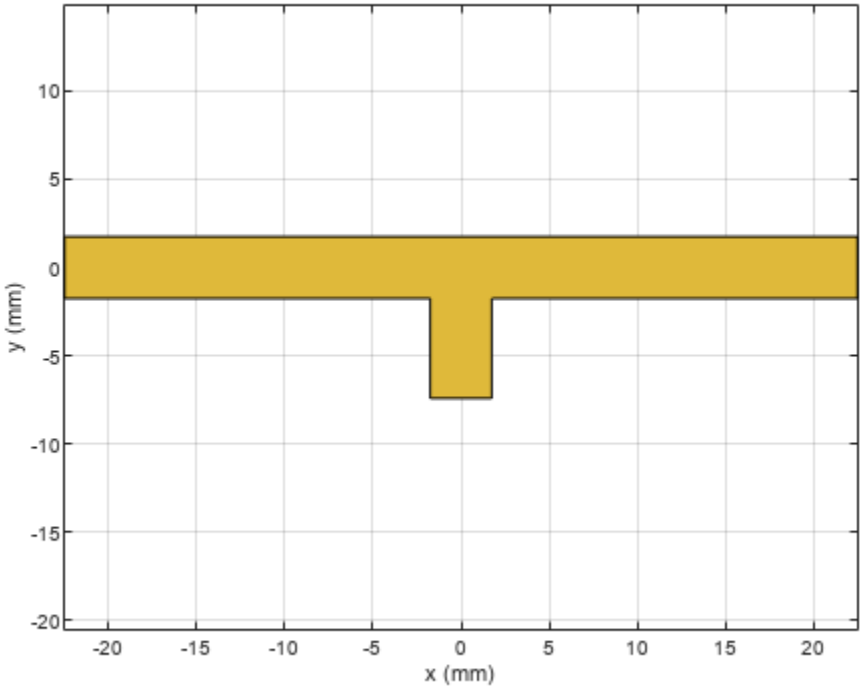


The output figures show the tee shaped matching network and the ground plane for creating the PCB stack.

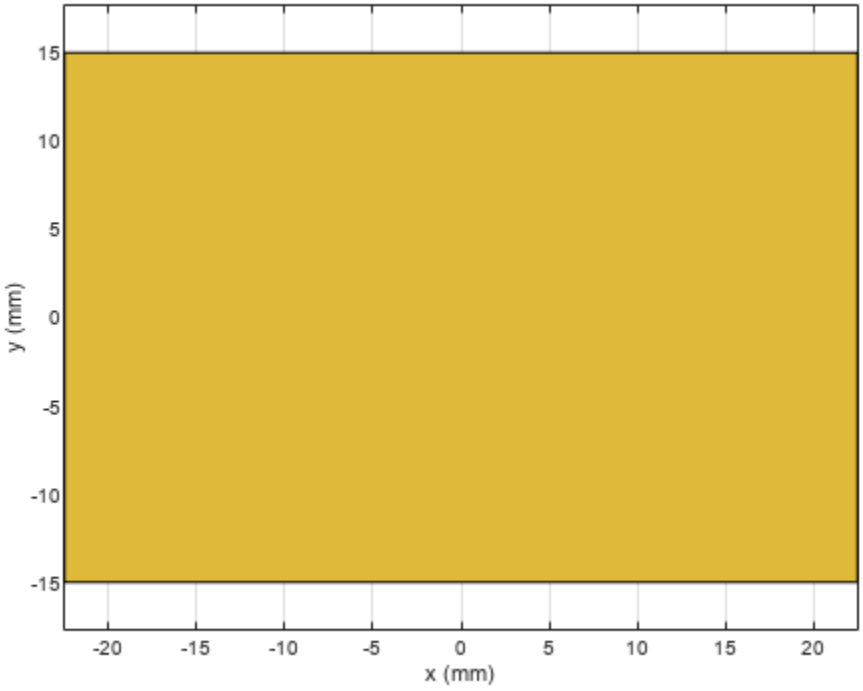
Design Output Matching Network Using traceTee shape

The output matching network also consists of one shunt stub and one series microstrip transmission line. The dimensions of the series transmission line is Length_Line2 and the length of the stub is Length_Stub2. The width of the lines is Width_Line.

```
OutputMatching = traceTee('Length',[Length_Line2 Length_Stub2],'Width',[Width_Line Width_Line]);
figure;
show(OutputMatching);
```




```
Gnd2 = antenna.Rectangle('Length',Length_Line2,'Width',30e-3);
figure;
show(Gnd2);
```



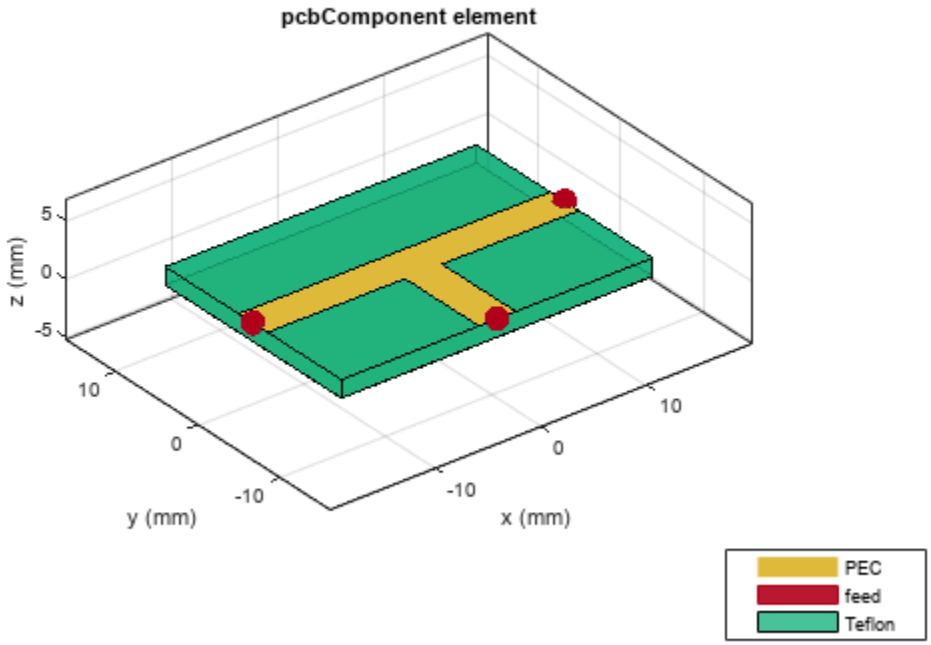
The output figures show the tee shaped matching network and the ground plane for creating the PCB stack up.

Create pcbComponent for Input and Output Matching Networks

The `pcbComponent` method creates the PCB stack for any of the RF PCB shapes and it assigns the layers for dielectric, ground plane and feeds at the open ends of the shape.

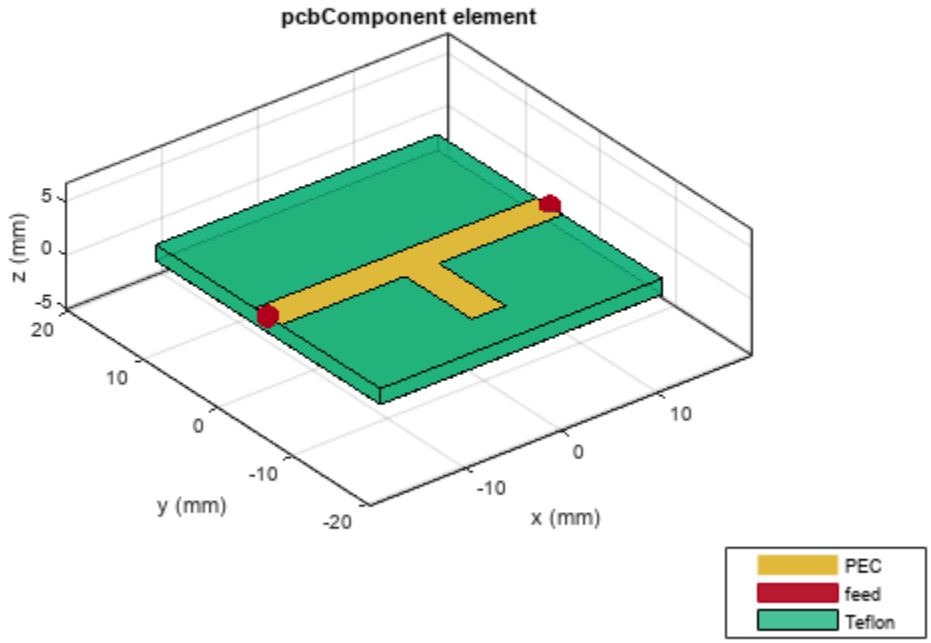
Use the `pcbComponent` method to create the PCB stack of the `InputMatching traceTee` shape. Use the show method to visualize the PCB stack up of the traceTee. Notice that the `pcbComponent` method assigns the `InputMatching` shape, dielectric, ground plane to `Layers` property of `pcbComponent` and assigns FeedLocations at three open ends of the shape.

```
InputMatchingNw = pcbComponent(InputMatching);
figure;
show(InputMatchingNw);
```



The output shows the top and the bottom metal with the dielectric in the middle and ports at all the open ends of the shape. However the stub needs to be an open circuit. For this you can remove the feed at the third port, increase the ground plane width and change the dielectric values as per the paper[1] using the below code.

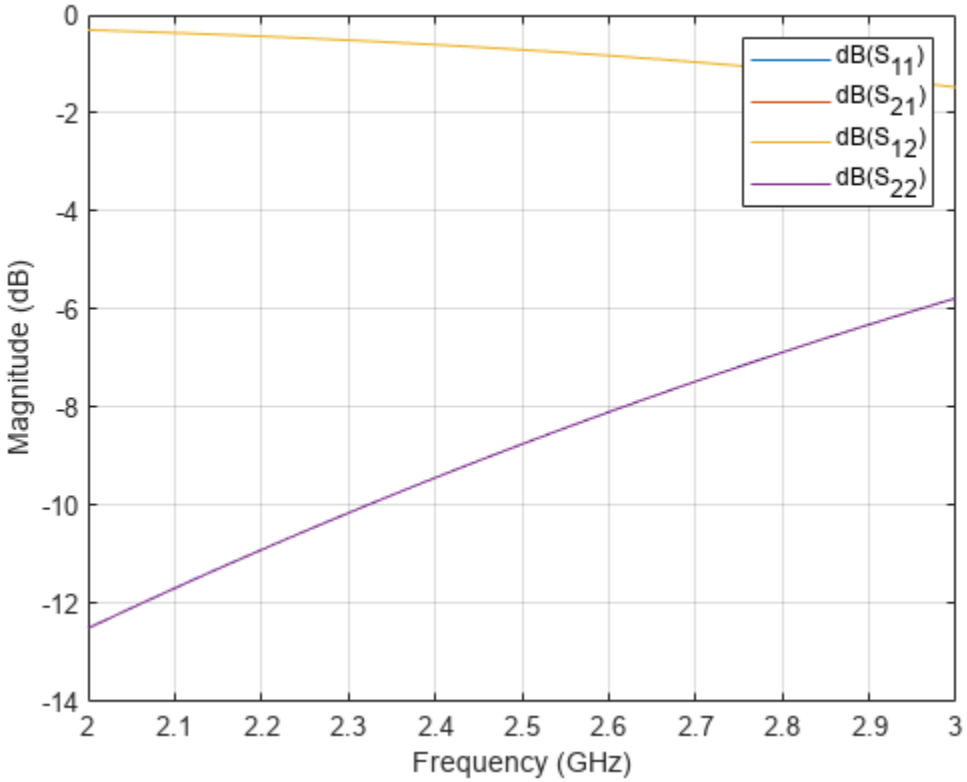
```
d=dielectric('Name',{'Teflon'},'EpsilonR',EpsilonR,'LossTangent',LossTangent,'Thickness',0.0016)
InputMatchingNw.Layers(2:3)          = {d,Gnd1};
InputMatchingNw.BoardShape           = Gnd1;
InputMatchingNw.FeedLocations(3,:)   = [];
InputMatchingNw.BoardThickness       = Height;
figure;
show(InputMatchingNw);
```



Observe the changes in the PCB stack up after removal of the feed and increasing the ground plane width.

Use the `sparameters` function to calculate the s-parameters of the input matching network.

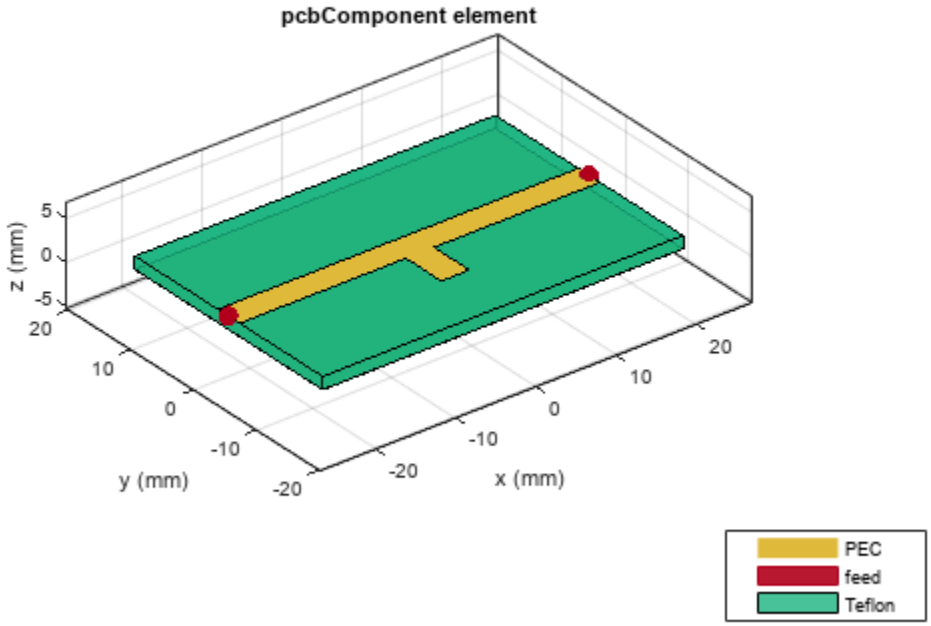
```
s_in=sparameters(InputMatchingNw,freq);
figure;
rfplot(s_in);
```




```
rfwrite(s_in,'s_in.s2p');
```

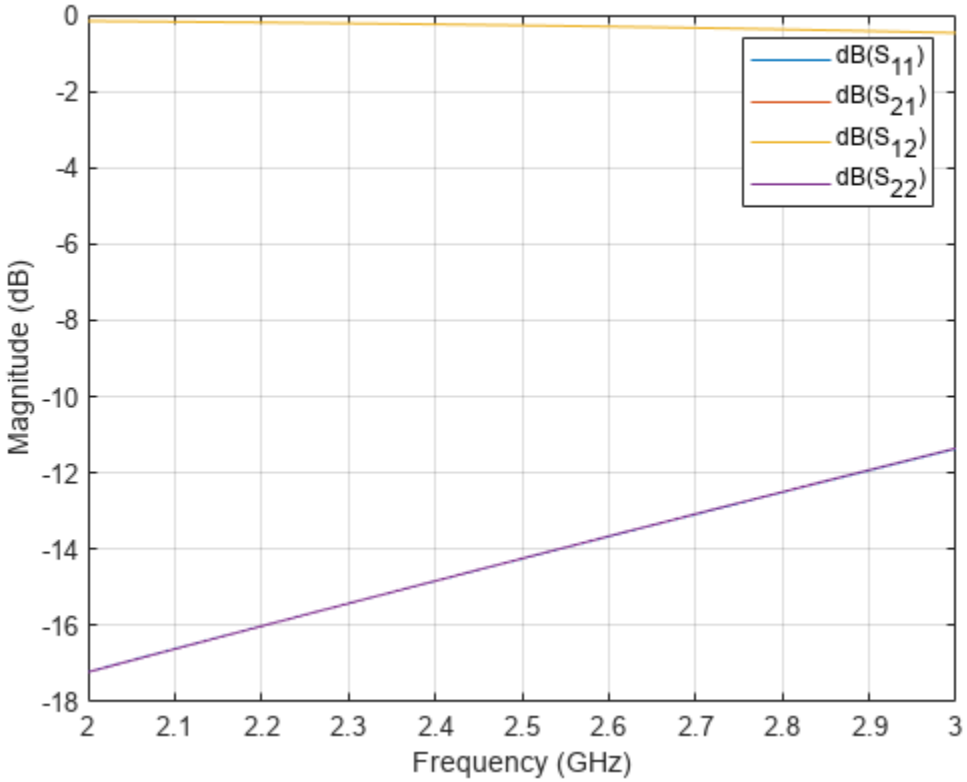
Perform the same steps for the output matching network and build the required PCB stack.

```
OutputMatchingNw                    = pcbComponent(OutputMatching);
OutputMatchingNw.Layers(2:3)        = {d Gnd2};
OutputMatchingNw.BoardShape         = Gnd2;
OutputMatchingNw.FeedLocations(3,:) = [];
OutputMatchingNw.BoardThickness     = Height;
figure;
show(OutputMatchingNw);
```



The output shows the top and the bottom metal with dielectric in the middle layer. Use the `sparameters` function to calculate the s-parameters of the output matching network.

```
s_out=sparameters(OutputMatchingNw,freq);
figure;
rfplot(s_out);
```




```
rfwrite(s_out,'s_out.s2p');
```

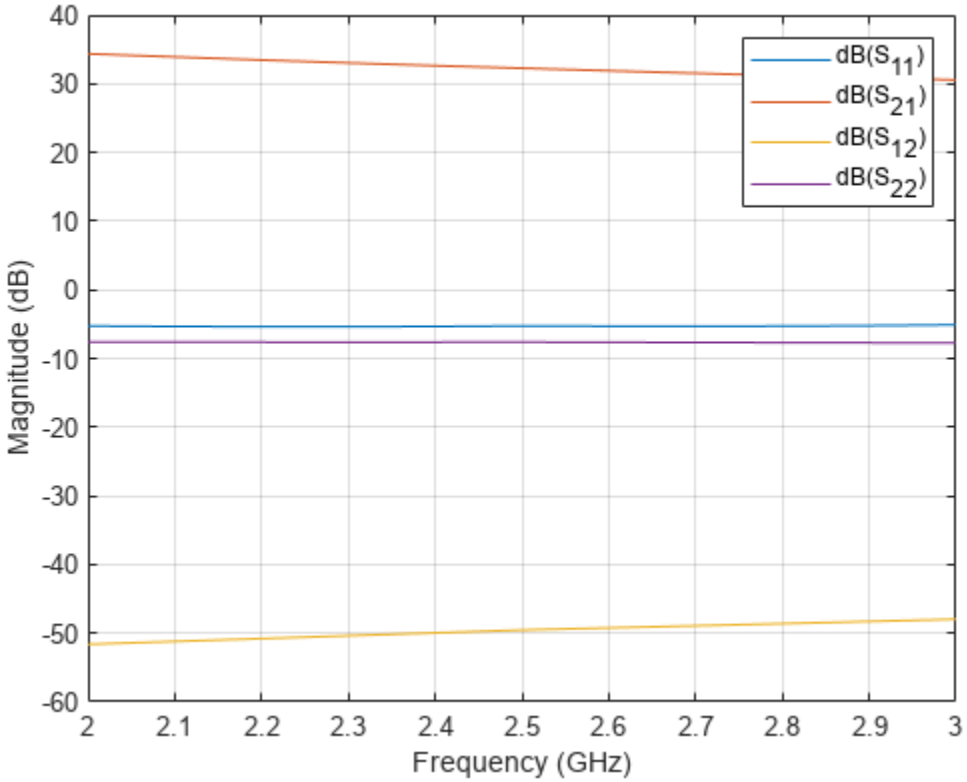
Use `nport` function from RF Toolbox to import the s-parameter files for the input and output matching networks.

```
IMNW = nport('s_in.s2p');
OMNW = nport('s_out.s2p');
```

Cascaded Amplifier S-Parameters

Use the `circuit` function to cascade the two amplifiers and use the `sparameters` function to calculate the s-parameters and plot it using the `rfplot` method.

```
casamp = circuit([Amp2,clone(Amp2)],'amplifiers');
S2 = sparameters(casamp,freq);
figure;
rfplot(S2);
```

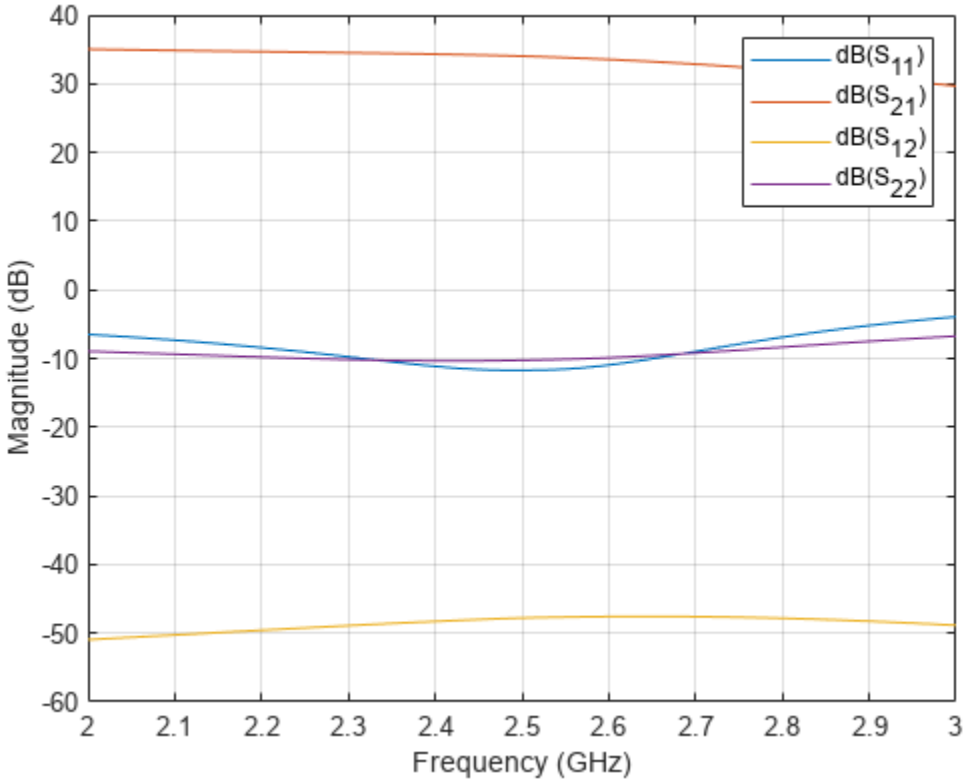


The plot shows that the S21 is around 35 dB but the S11 and S22 values are around -5 dB mark which is not sufficient as the expected values are -10 dB. So the MNWs that are designed will improve the S11 and S22 values.

Cascade Input and Output Matching Networks with Two Amplifiers

Use the `circuit` function to cascade the Input and output matching networks' s-parameters and the two amplifiers and plot the s-parameters. The S11 and S22 are below -10 dB indicating a proper matching at the input and output ports of the amplifier.

```
casamp1 = circuit([IMNW ,clone(Amp1),clone(Amp2), OMNW],'amplifiersWithMatching');
S3 = sparameters(casamp1,freq);
figure;
rfplot(S3);
```

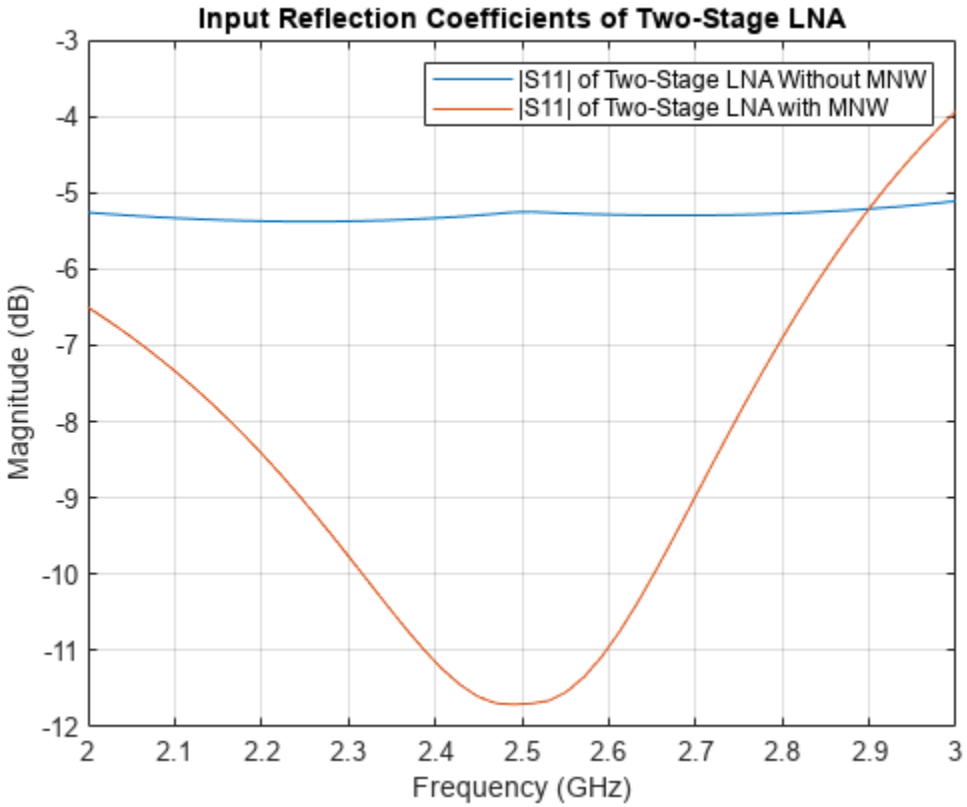


The plot shows that the S11 and S22 are below -10 dB and hence the reflected power is reduced at Port1 and Port2.

Compare Input Reflection Coefficients of Two-Stage LNA

To verify the simultaneous conjugate match at the input of the amplifier, plot the input reflection coefficients in dB for the amplifier circuit with and without a matching network. Plot the s-parameters with and without the matching networks over the frequency range of 2 – 3 GHz.

```
figure;
rfplot(S2,1,1);
hold on;
rfplot(S3,1,1);
legend('|S11| of Two-Stage LNA Without MNW','|S11| of Two-Stage LNA with MNW');
title('Input Reflection Coefficients of Two-Stage LNA');
grid on;
```

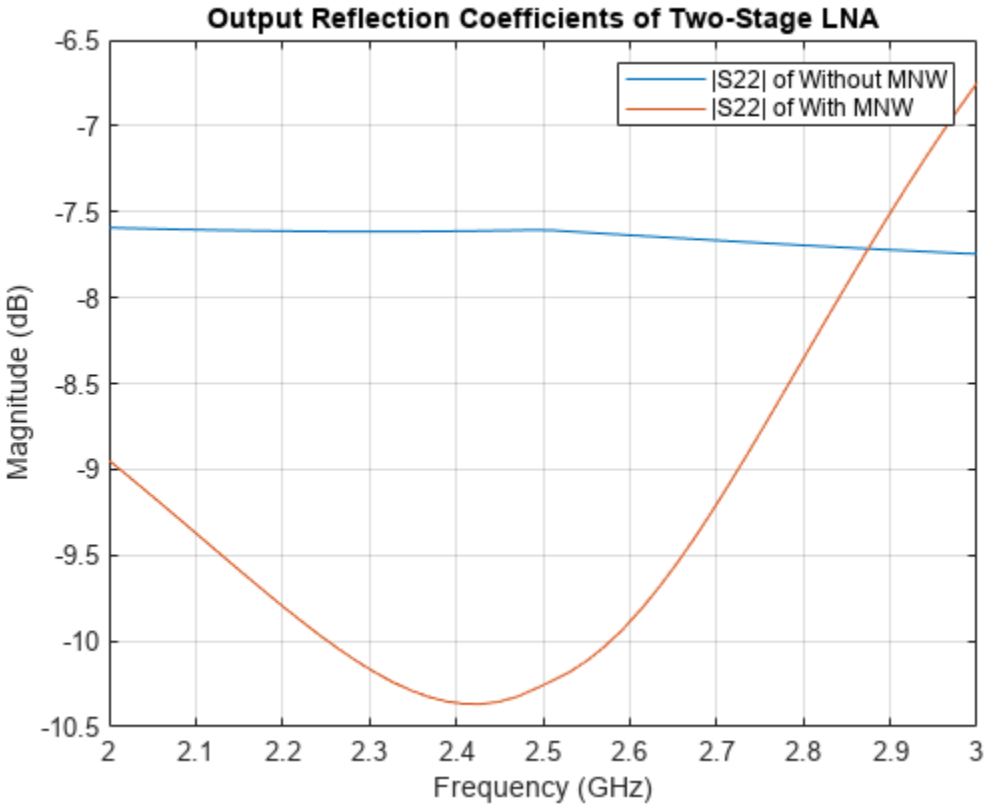


The input return loss for the two-stage LNA with the input MNW is around –12 dB.

Compare Output Reflection Coefficients of Two-Stage LNA

To verify the simultaneous conjugate match at the output of the amplifier, plot output reflection coefficients in dB for both the two-stage LNA with and without a MNW.

```
figure;
rfplot(S2,2,2);
hold on;
rfplot(S3,2,2);
legend('|S22| of Without MNW','|S22| of With MNW');
title('Output Reflection Coefficients of Two-Stage LNA');
grid on;
```



The calculated output return loss for the two-stage LNA with the output MNW is around -10 dB.

This matching network design can be used for matching any input impedance and convert that impedance to 50 ohm so that you get good matching at input and output.

Reference

[1] Maruddani, B, M Ma'sum, E Sandi, Y Taryana, T Daniati, and W Dara. "Design of Two Stage Low Noise Amplifier at 2.4 - 2.5 GHz Frequency Using Microstrip Line Matching Network Method." *Journal of Physics: Conference Series* 1402 (December 2019): 044031.

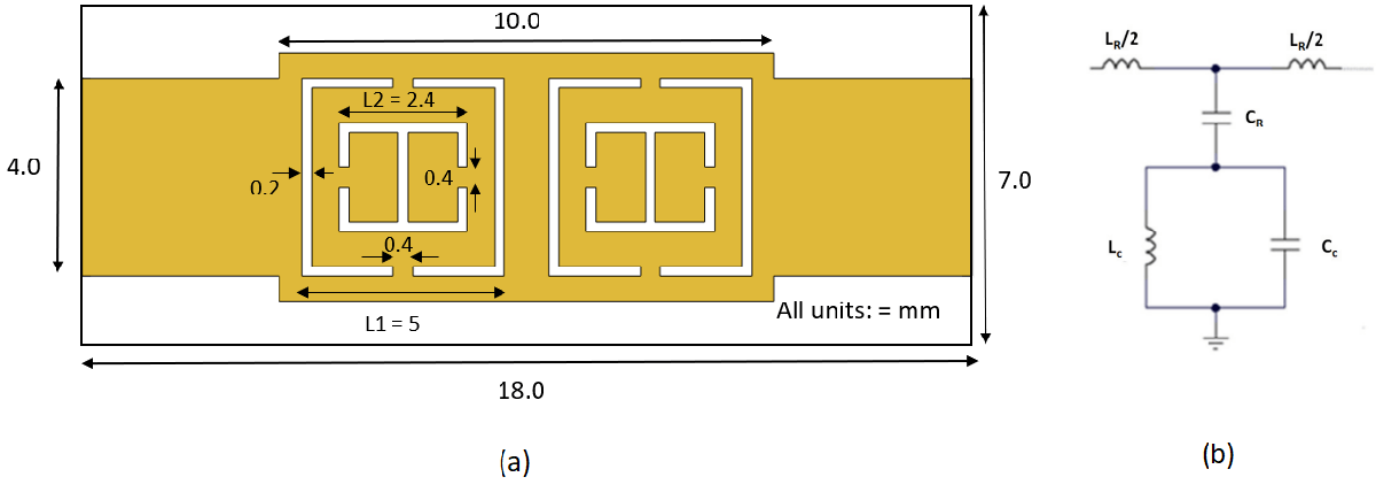
Design and Analyze Compact UWB Low Pass Filter Using pcbComponent

This example shows how to design and analyze a compact low pass ultra-wide band (UWB) filter based on a U-shaped complementary split ring resonator using the pcbComponent object. The filter is designed to have very low insertion loss over a wide band of frequency from low 0.1 GHz to 10.8 GHz. The design of the filter is taken from the reference [1].

Design and Analyze UWB Low Pass Filter

This compact filter design employs U-shaped complementary split ring resonator (U-CSRR). U-CSRR is a uniplanar configuration of complementary split ring resonator (CSRR) as described in [1]. This structure has an advantage of simpler fabrication as it is formed on the top metal layer. The U-CSRR is formed using two concentric split square rings of outer length L_1, inner length L_2 as shown in Figure (a) below. This U-CSRR element is created on top metal layer of hosting transmission line. The equivalent circuit of the single U-CSRR cell shown in Figure (b) is represented by a parallel resonant circuit with inductance L_c and capacitance C_c. The inductance L_R and capacitance C_R represent the inductance and capacitance of the host transmission. The U-CSRR particle is electrically coupled to the host transmission line.

The equivalent circuit of the U-CSRR particle suggests that at low frequency, the impedance of the parallel tank circuit is small and the circuit has passband characteristics. Figure (a) shows the schematic diagram of such a filter employing two U-CSRR cells in microstrip [1] representing various feature dimensions. As in [1], the choice of two U-CSRR particles in filter design is to have a compact size and ensure high attenuation in stop band.



Use the `traceRectangular` object to create feeding transmission line `ZA` and rectangular unit cell `Cell_A`. Perform the Boolean `add` operation for the microstrip shapes `ZA`, `Cell_A` and create `LeftSection`.

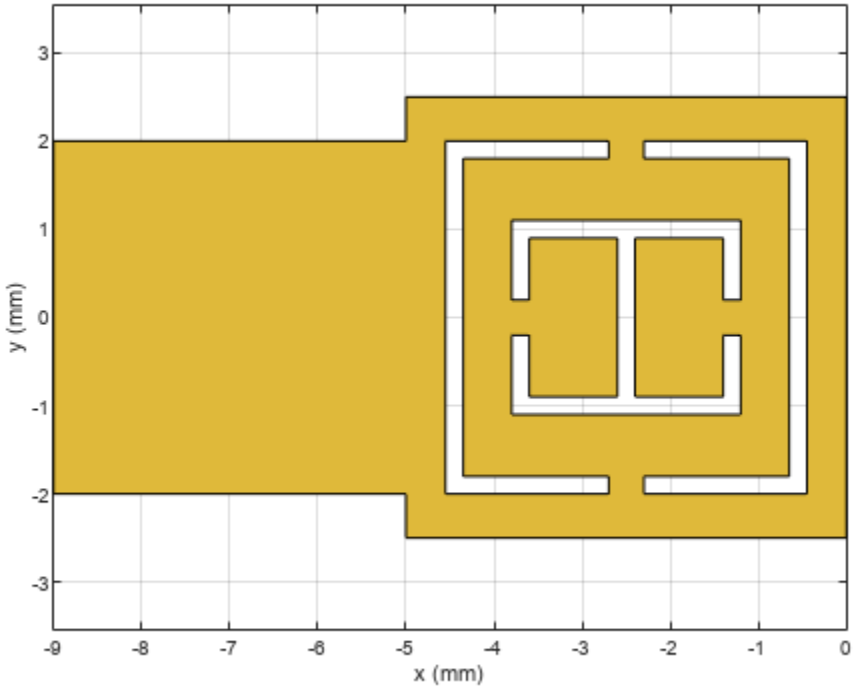
```
% Set variables for ground plane
gndL = 18e-3;
gndW = 7e-3;
```

```
% Set variables for feeding transmission line
ZA_Width = 4e-3;
ZA_Length = 4e-3;

% Define unit cell length
Cell_Length = 5e-3;

% Create feeding microstrip line
ZA = traceRectangular("Length",ZA_Length,"Width",ZA_Width,...
    "Center",[-ZA_Length/2-Cell_Length 0]);

% Create rectangular unit cell
Cell_A = traceRectangular("Length",Cell_Length,"Width",Cell_Length,...
    "Center",[-Cell_Length/2 0]);

% Join feeding line and rectangular unit cell
LeftSection = ZA + Cell_A;
```

Use `traceLine` object to create shapes `s1, s2, s3, s4`. Use `traceRectangular` object to create shape `s5.`

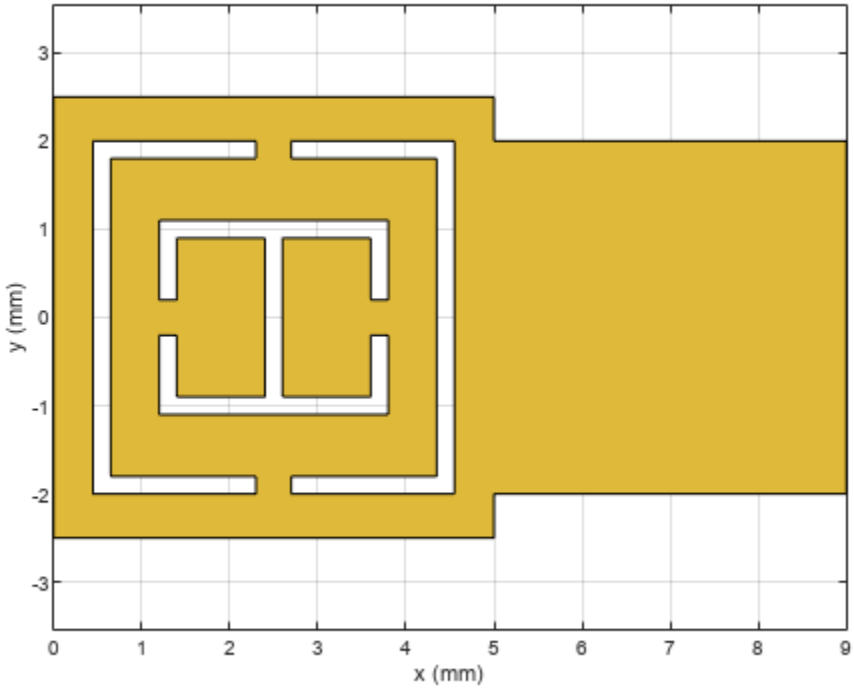
Subtract shapes `s1,s2,s3,s4,` and `s5` from `LeftSection`. This operation creates various slots seen on U-CSRR particle. Visualize `LeftSection` using the `show`

```
% Create shapes for various slots
s1 = traceLine('StartPoint',[-Cell_Length/2-0.2e-3  -1.9e-3],...
    'Angle',[-180 -270 0],'Length',[1.75e-3 3.8e-3 1.75e-3],'Width',0.2e-3);

s2 = traceLine('StartPoint',[-Cell_Length/2+0.2e-3  -1.9e-3],...
    'Angle',[0 90 180],'Length',[1.75e-3 3.8e-3 1.75e-3],'Width',0.2e-3);

s3 = traceLine('StartPoint',[-Cell_Length/2-1.2e-3  -0.2e-3],...
    'Angle',[-90 0 90],'Length',[0.8e-3 2.4e-3 0.8e-3],'Width',0.2e-3);

s4 = traceLine('StartPoint',[-Cell_Length/2-1.2e-3   0.2e-3],...
    'Angle',[90 0 -90],'Length',[0.8e-3 2.4e-3 0.8e-3],'Width',0.2e-3);

s5 = traceRectangular("Length",0.2e-3,"Width",1.8e-3,...
    "Center",[-Cell_Length/2 0]);

% Create slots of U-CSRR on hosted micrsotrip line
LeftSection = LeftSection -s1 -s2 -s3 -s4 -s5;
figure;
show(LeftSection);
```



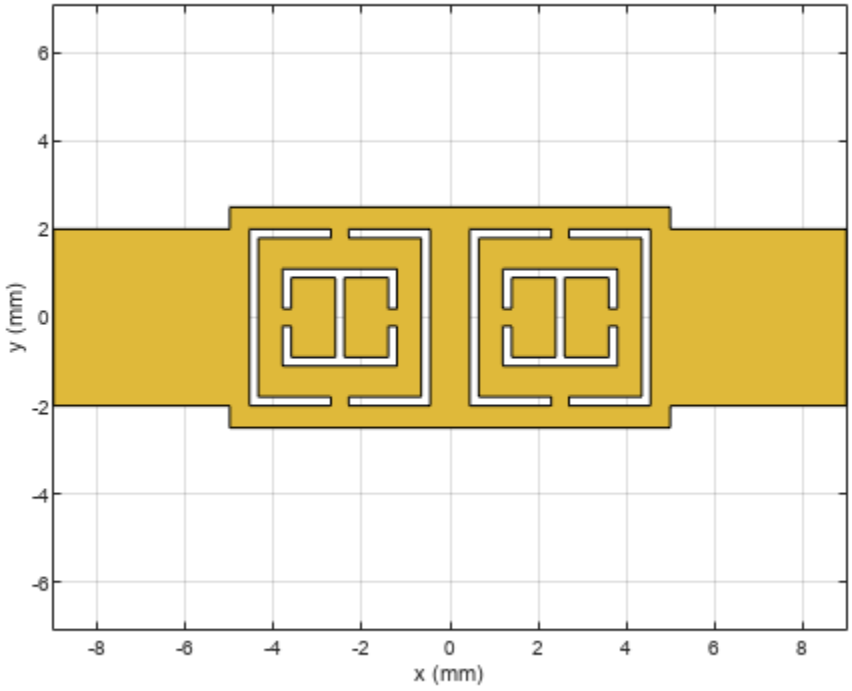
Use the `copy` and `mirrorY` methods on the `LeftSection` object to create a `RightSection`. This creates right portion of filter having another U-CSRR hosted transmission line. Visualize `RightSection` using `show function.`

```
RightSection = copy(LeftSection);
RightSection = mirrorY(RightSection);
figure;
show(RightSection);
```



Perform theBoolean `add` operation for the shapes `LeftSection`, `RightSection` to create a `filter`. Visualize the filter.

```
filter = LeftSection + RightSection;
show(filter);
```

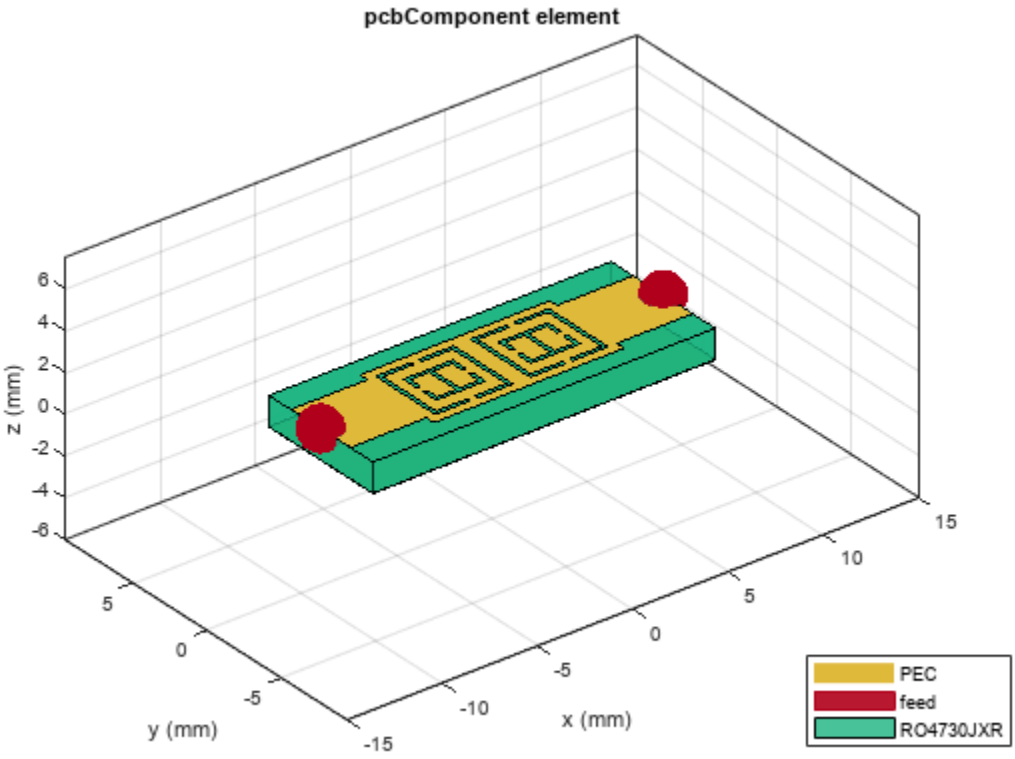


Define the substrate parameters and create a dielectric to use in the `pcbComponent` of the designed filter. Create a groundplane using the `traceRectangular` shape. Use the `pcbComponent` to create a filter PCB. Assign the dielectric and ground plane to the `Layers` property on `pcbComponent`. Assign the `FeedLocations` to the edge of the feed ports. Set the `BoardThickness` to 1.52 mm on the `pcbComponent` and visualize the filter. The below code performs these operations and creates the filter PCB.

```
% Define Substrate and its thickness
substrate = dielectric("RO4730JXR");
substrate.Thickness = 1.52e-3;

% Define bottom ground plane
ground = traceRectangular("Length",gndL,"Width",gndW,...
   "Center",[0,0]);
```

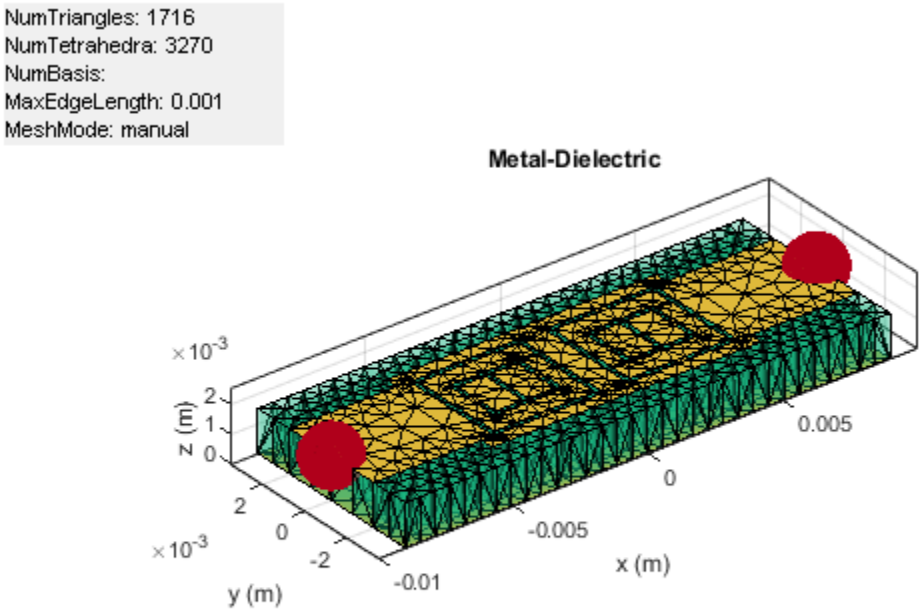
Use pcbComponent to create a filter pcb

```
pcb = pcbComponent;
pcb.BoardShape = ground;
pcb.BoardThickness = 1.52e-3;
pcb.Layers ={filter,substrate,ground};
pcb.FeedDiameter = ZA_Width/2;
pcb.FeedLocations = [-gndL/2 0 1 3;gndL/2 0 1 3];
figure;
show(pcb);
```



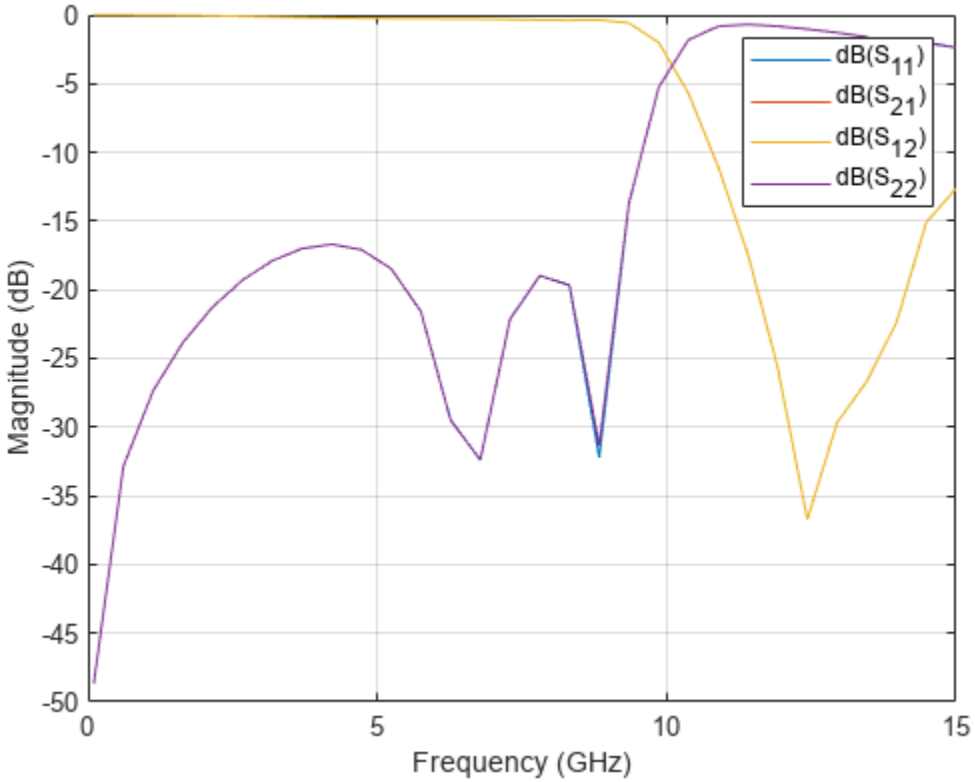
Use the `mesh` function to have fine meshing and set `MaxEdgeLength to 1mm.`

```
figure;
mesh(pcb,'MaxEdgeLength',1e-3);
```



Use the `sparameters` function to calculate the S-parameters for the low pass filter and plot it using the `rfplot` function.

```
spar = sparameters(pcb,linspace(0.1e9,15e9,30));
figure;
rfplot(spar);
```

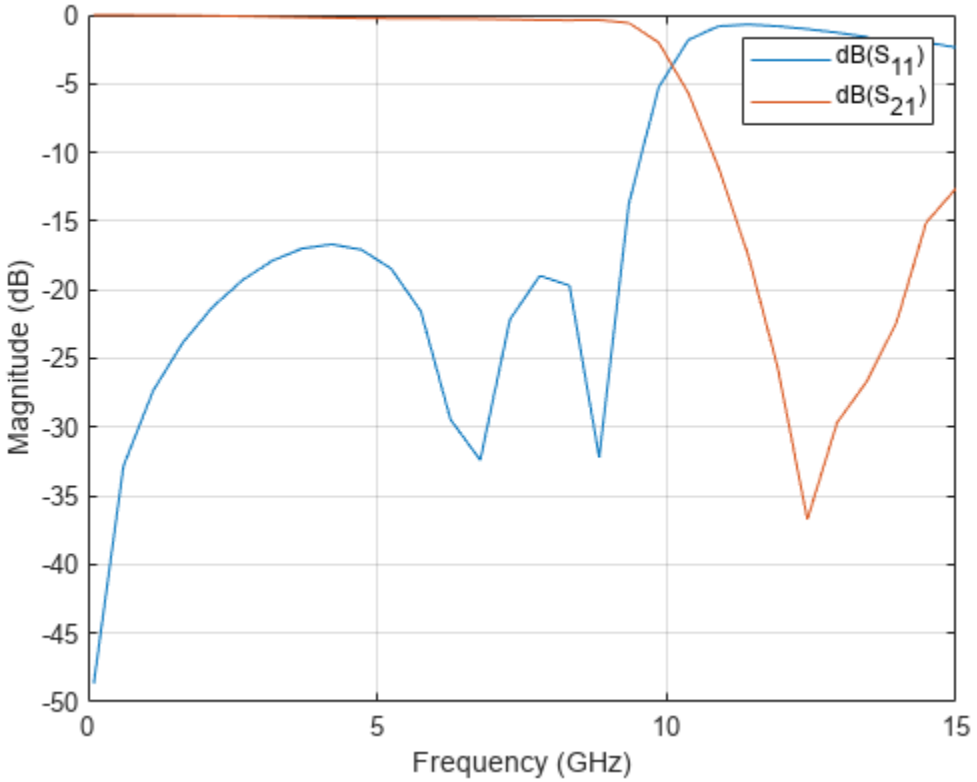


As there are four curves in the result, let us analyze the results.

Plot S-Parameters

Analyze the values of S_{21}, and S_{11} to understand the behavior of low pass filter.

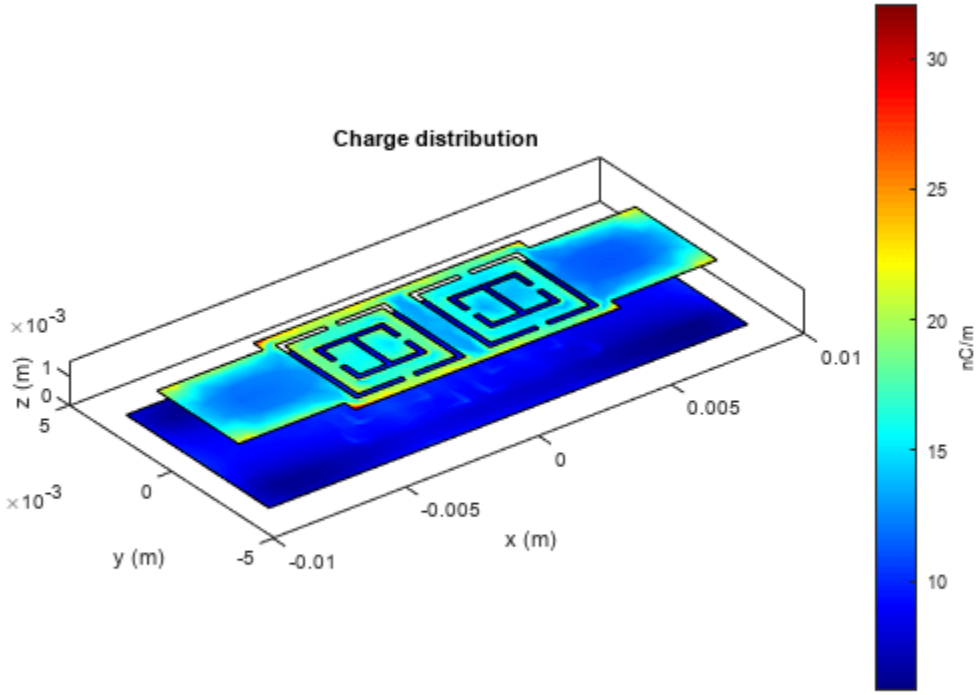
```
figure;
rfplot(spar,[1 2],1);
```



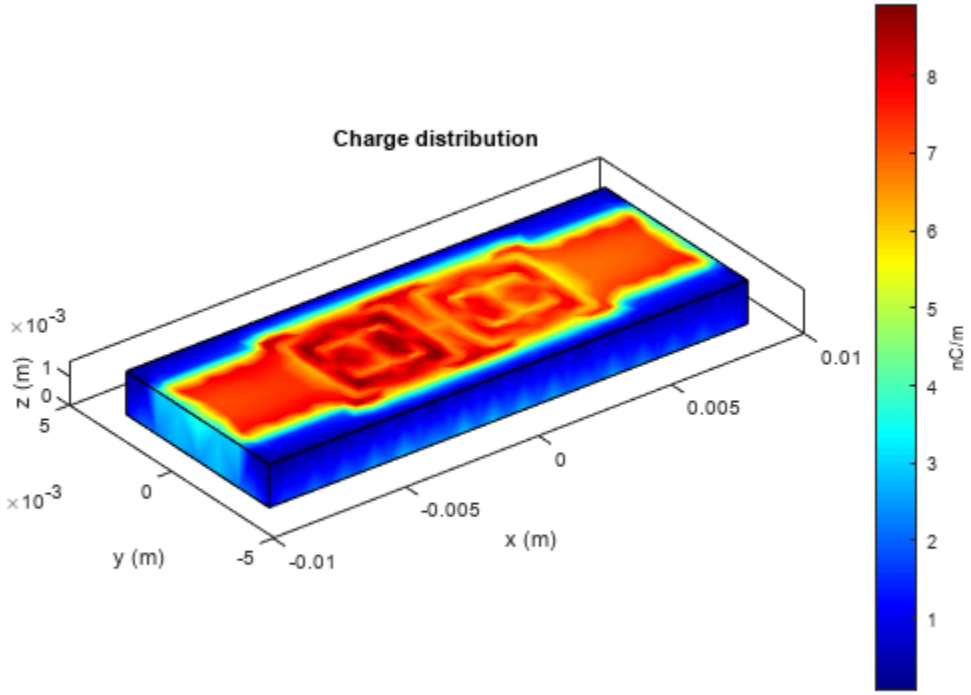
The result shows that the filter has S_{21} values close to 0 dB and S_{11} values less than -15 dB between wide band of frequencies f_1 = 0.1 GHz and f_2 = 10.0 GHz. The designed filter therefore has ultra-wide passband response. For frequencies greater than 10.8 GHz, S_{21} values are less than -10 dB indicating stopband response.

Use the `charge` function to visualize the charge distribution on the metal surface and dielectric of low pass filter.

```
figure;
charge(pcb,5e9);
```

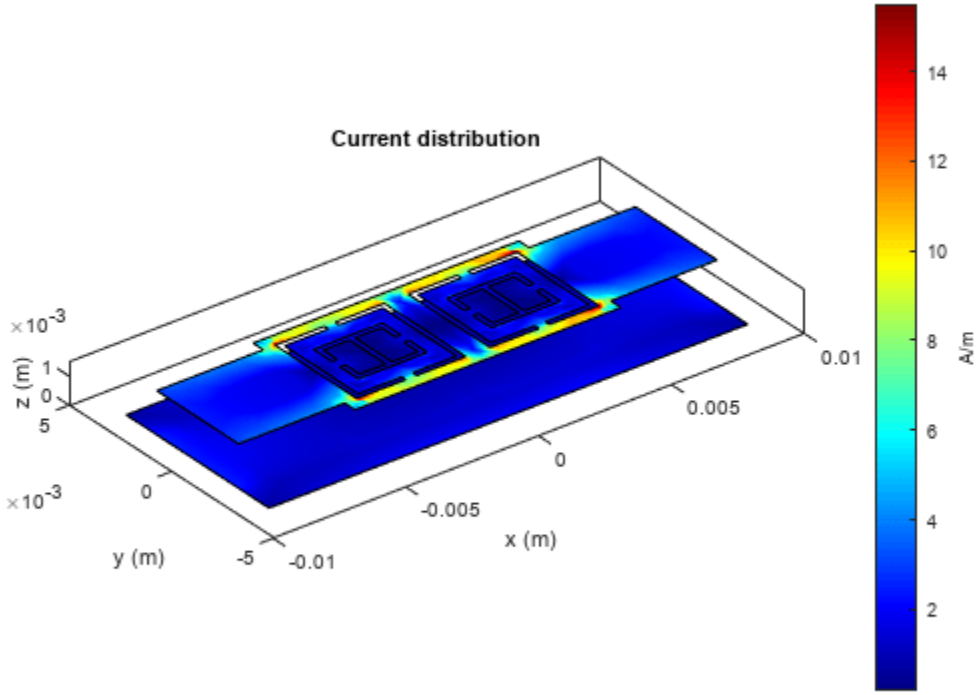



```
figure;
charge(pcb,5e9,'dielectric');
```

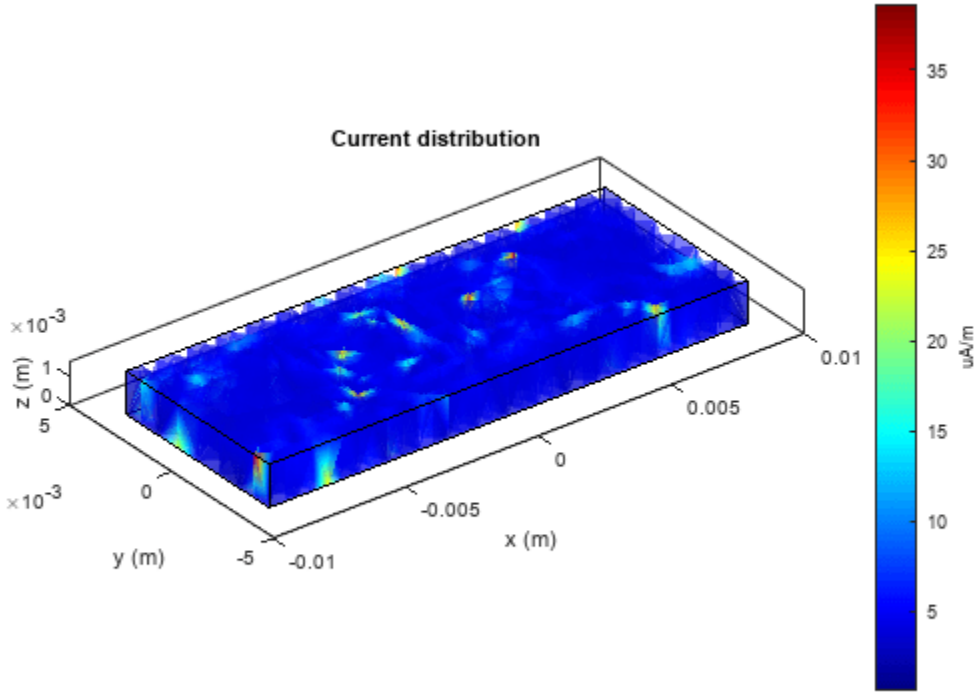


Use the `current` function to visualize the current distribution on the metal surface and volume polarization currents on dielectric of low pass filter.

```
figure;
current(pcb,5e9);
```




```
figure;
current(pcb,5e9,'dielectric');
```



References

[1] Abdalla, M. A., G. Arafa, and M. Saad, “Compact UWB LPF based on uni-planar metamaterial complementary split ring resonator,” Proceedings of 10th International Congress on Advanced Electromagnetic Materials in Microwaves and Optics (METAMATERIALS), 10–12, Chania, Greece, Sep. 2016

Design and Analyze Band Stop Filter using pcbComponent

This example shows you how to design and analyze band stop filter using the `pcbComponent` object. Design the band stop filter with a fractional bandwidth (FBW) of 1.0 at a midband frequency f_0 of 2.5 GHz for the band-edge frequencies of f_1 = 1.25 GHz and f_2 = 3.75 GHz as defined in Figure 6.11b of reference [1].

Design Band Stop Filter

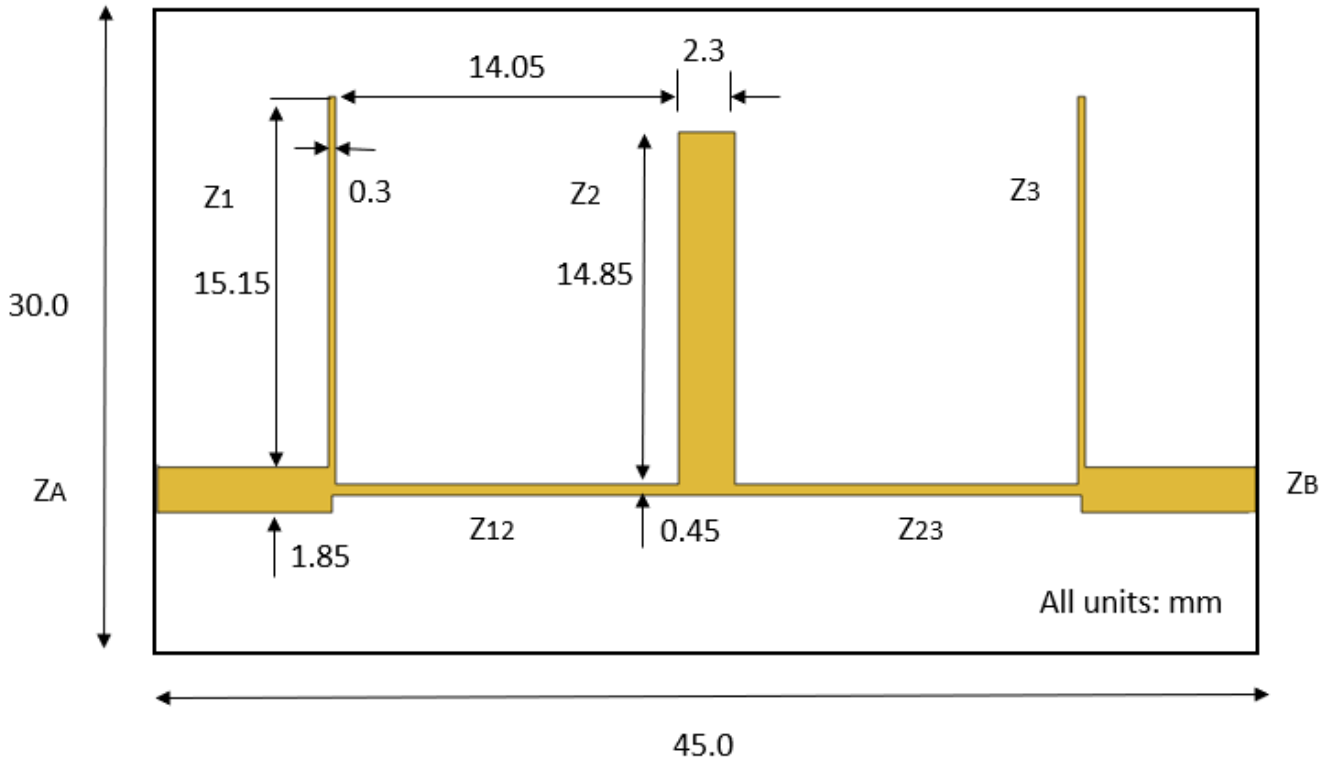
Design the microstrip band stop filter based on a three-pole (n = 3) Chebyshev lowpass prototype with 0.05 dB passband ripple. The element values of the lowpass prototype are

- $g_0 = g_4 = 1.0$
- $g_1 = g_3 = 0.8794$
- $g_2 = 1.1132$

Using the design equations Eq.(6.30) given in refrence [1] on page no. 182 for n = 3 and Z_0 = 50 Ω as , you can obtain

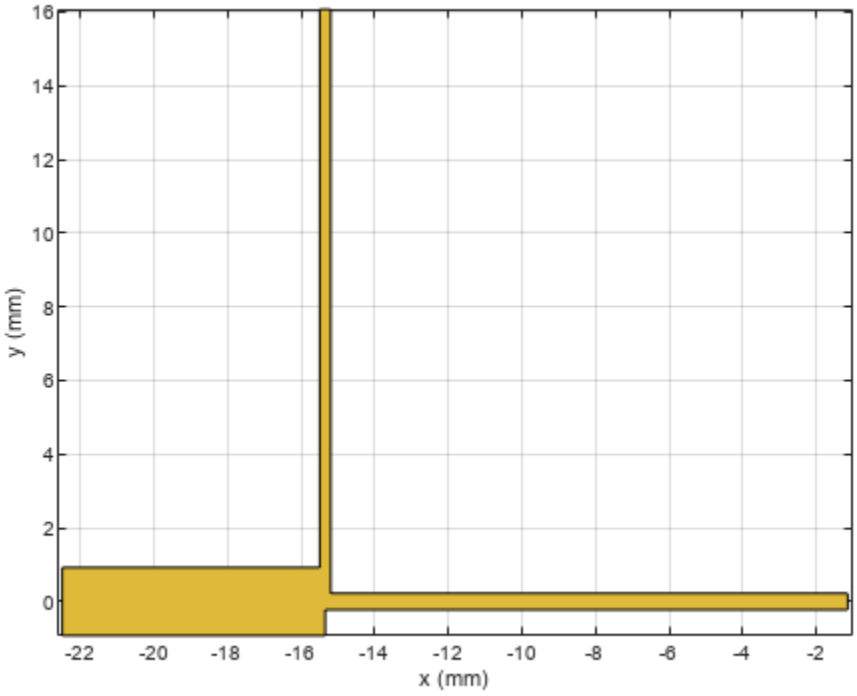
- $Z_A = Z_B$ = 50 ohm
- $Z_1 = Z_3$ = 106.8544 ohm
- $Z_{12} = Z_{23}$ = 93.9712 ohm
- Z_2 = 44.9169 ohm

Choose a commercial substrate `(RT/D 6006)` with a relative dielectric constant of 6.15 and thickness of 1.27 mm. Calculate the microstrip widths using the microstrip design equations given in Chapter 4 of reference [1]. The figure shows the schematic diagram of the microstrip band stop filter [1] representing various feature dimensions.



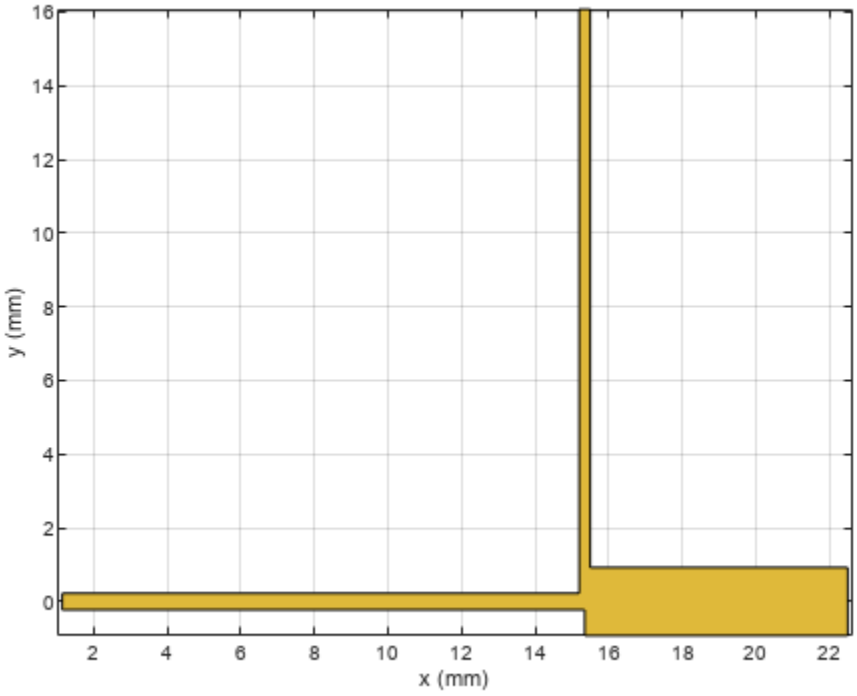
Use the `traceRectangular` object to create `ZA`, `Z1`, `Z12`. Perform a Boolean add operation for the microstrip shapes `ZA`, `Z1`, `Z12` and create a `LeftSection` object. Visualize the `LeftSection` using `show` function.

```
ZA_Width = 1.85e-3;
ZA_Length = 7e-3;
Z1_Length = 0.3e-3;
Z1_Width = 15.15e-3;
Z12_Length = 14.05e-3;
Z12_Width = 0.45e-3;
Z2_Length = 2.3e-3;
Z2_Width = 14.85e-3;
gndL = 45e-3;
gndW = 30e-3;

ZA = traceRectangular(Length = ZA_Length+Z1_Length/2,Width = ZA_Width,...
    Center = [-gndL/2+ZA_Length/2+Z1_Length/4 0]);
Z1 = traceRectangular(Length = Z1_Length,Width = Z1_Width+ZA_Width/2,...
    Center = [-gndL/2+ZA_Length+Z1_Length/2 (Z1_Width/2+ZA_Width/4)]);
Z12 = traceRectangular(Length = Z12_Length+Z1_Length,Width = Z12_Width,...
    Center = [-gndL/2+ZA_Length+Z1_Length/2+Z12_Length/2 0]);
LeftSection = ZA+Z1+Z12;
figure;
show(LeftSection);
```



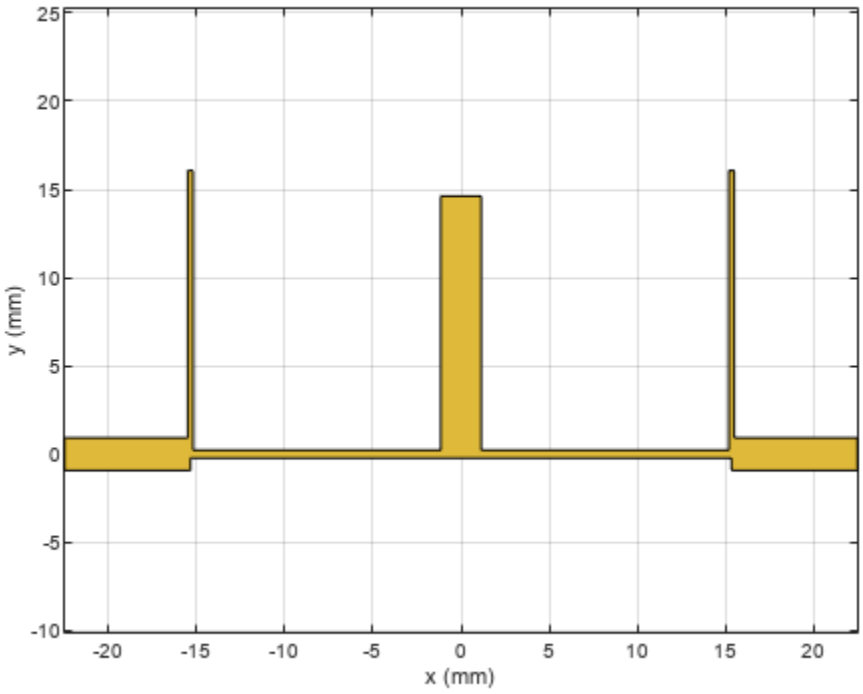
Use the `copy`, `rotateZ` and `rotateX` functions on `LeftSection` object to create a `RightSection`. Visualize the `RightSection` object.

```
RightSection = copy(LeftSection);
RightSection = rotateZ(RightSection,180);
RightSection = rotateX(RightSection,180);
figure;
show(RightSection);
```



Perform a Boolean add operation for the shapes `LeftSection` and `RightSection` to create a `combineSection` object. Use the `traceRectangular` object to create centerArm `Z2`. Perform a Boolean add operation for the shapes `combineSection`, `Z2`, and create a `filter` object. Visualize the `filter` object.

```
combineSection = LeftSection + RightSection;
Z2 = traceRectangular(Length = Z2_Length,Width = Z2_Width,...
    Center = [0 -Z12_Width/2+Z2_Width/2]);
filter = combineSection + Z2;
show(filter);
```



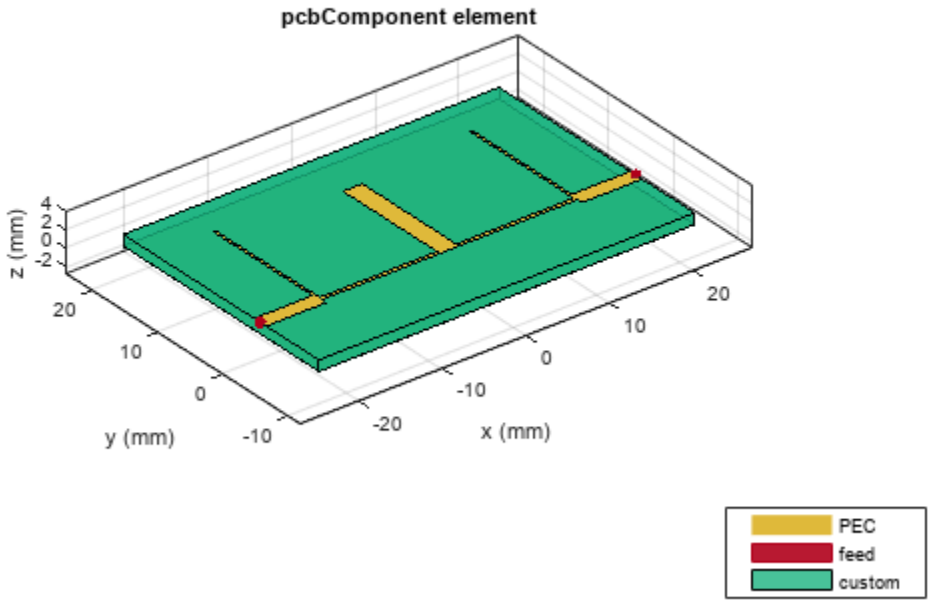
Define the substrate parameters and create a dielectric to use in the `pcbComponent` of the designed filter. Create a groundplane using the `traceRectangular` shape.

```
substrate = dielectric(EpsilonR = 6.15,LossTangent = 0.0027,...
    Name = "custom",Thickness = 1.27e-3);

ground = traceRectangular(Length = gndL,Width = gndW,...
    Center = [0,6e-3]);
```

Create PCB Filter Using `pcbComponent`

Use the `pcbComponent` to create a filter PCB. Assign the dielectric and ground plane to the `Layers` property of the `pcbComponent`. Assign the `FeedLocations` to the edge of the feed ports. Set the `BoardThickness` to 1.27 mm on the `pcbComponent` and visualize the filter.

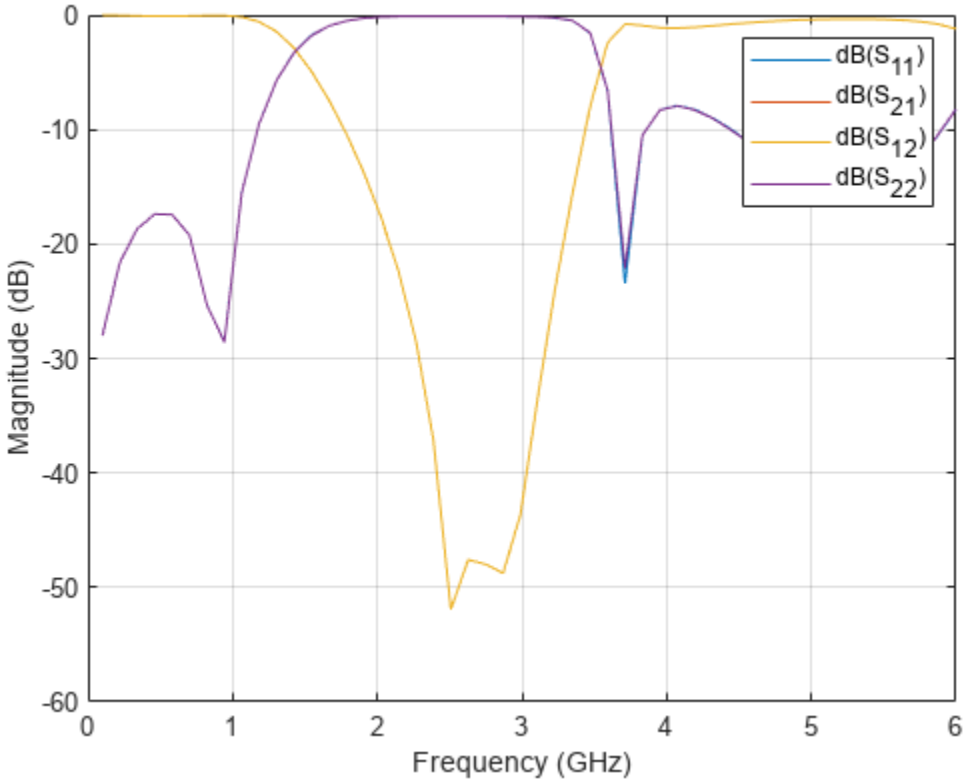
```
pcb = pcbComponent;
pcb.BoardShape = ground;
pcb.BoardThickness = 1.27e-3;
pcb.Layers ={filter,substrate,ground};
pcb.FeedDiameter = ZA_Width/2;
pcb.FeedLocations = [-gndL/2 0 1 3;gndL/2 0 1 3];
figure;
show(pcb);
```



Plot and Analyze the S-Parameters

Use the `sparameters` function to calculate the s-parameters for the band stop filter and plot it using the `rfplot` function.

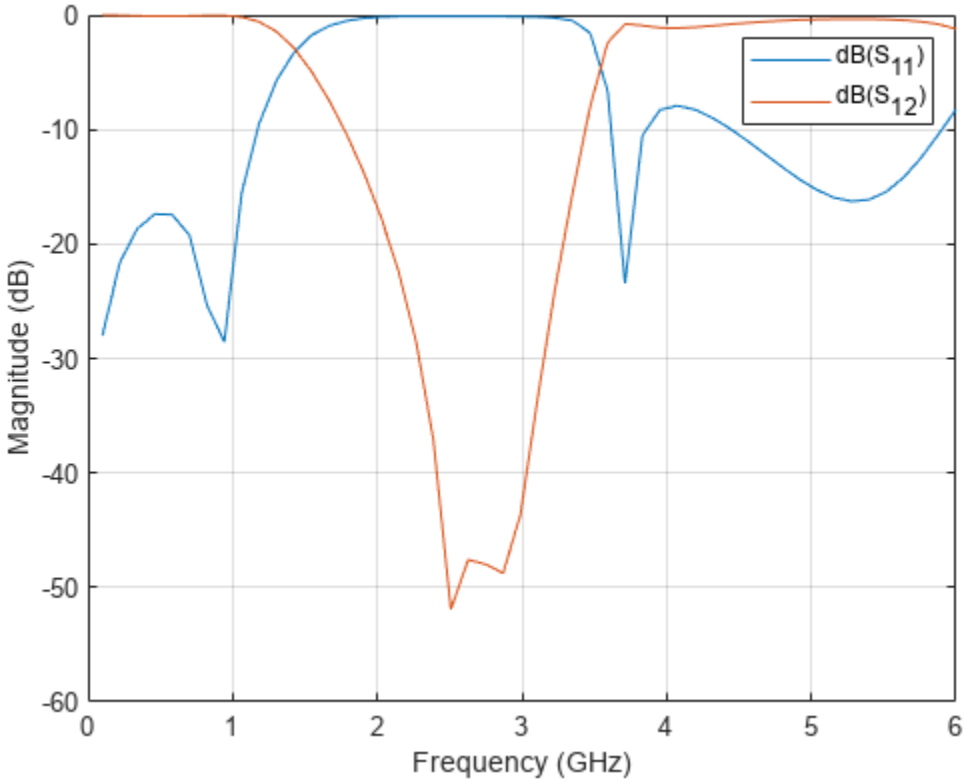
```
spar = sparameters(pcb,linspace(0.1e9,6e9,50));
figure;
rfplot(spar);
```



As there are four curves in the result, let us analyze the results.

Analyze the values of S_{12}, and S_{11} to understand the behavior of band stop filter.

```
figure
rfplot(spar,1,1:2)
```

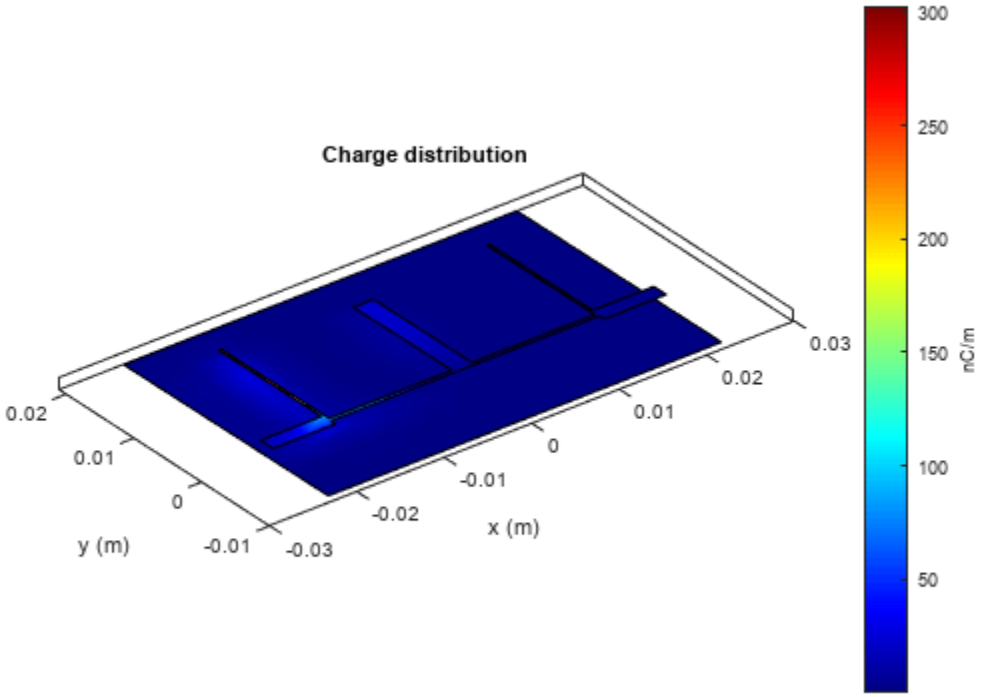


The result shows that the filter has center frequency f_0 = 2.5 GHz for the band-edge frequencies f_1 = 1.75 GHz and f_2 = 3.4 GHz. The S_{11} values are close to 0 dB and S_{12} values are less than -10 dB between frequencies f_1 = 1.4 GHz and f_2 = 3.4 GHz. The designed filter therefore has stopband response. The shift in band-edge frequencies f_1 and f_2 might be due to use of different full wave numerical solver used for EM simulation.

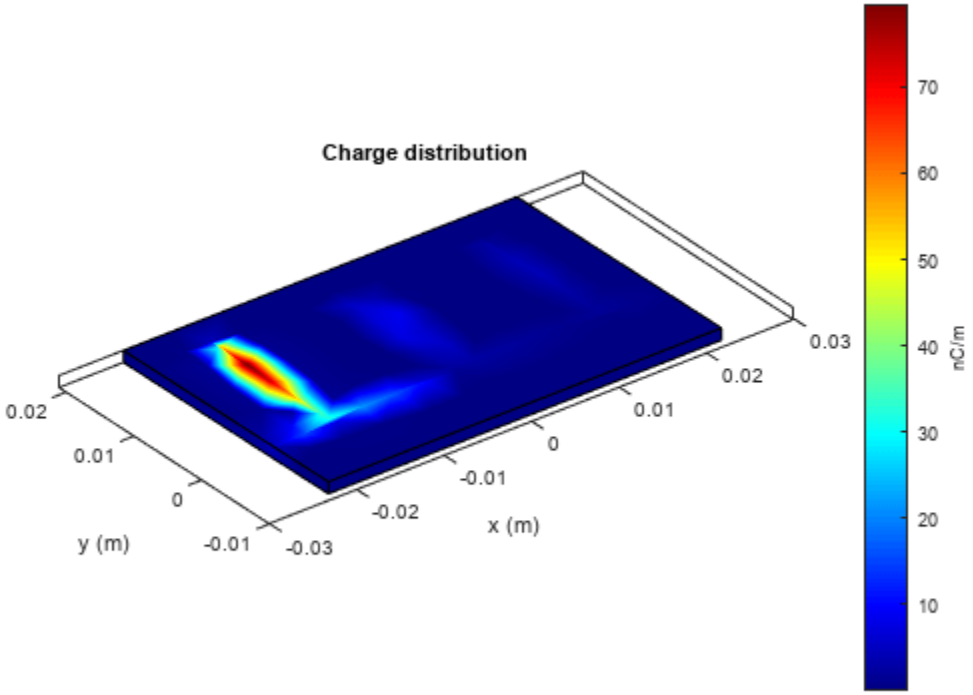
Visualize Charge and Current Distribution

Use the `charge` function to visualize the charge distribution on the metal surface and dielectric of band stop filter.

```
figure;
charge(pcb,2.4e9);
```

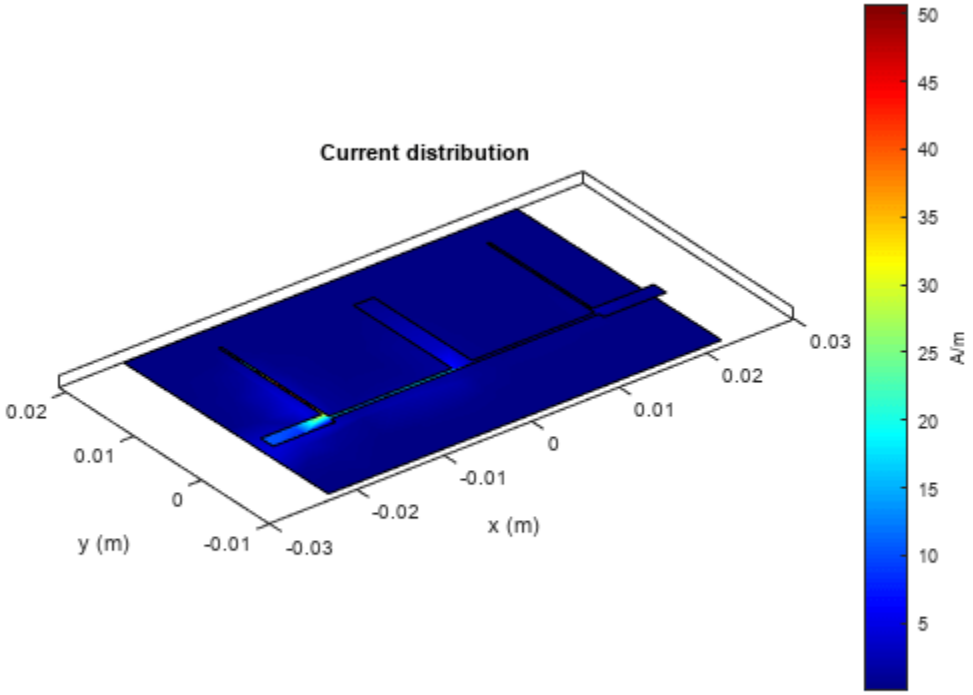



```
figure;
charge(pcb,2.4e9,'dielectric');
```

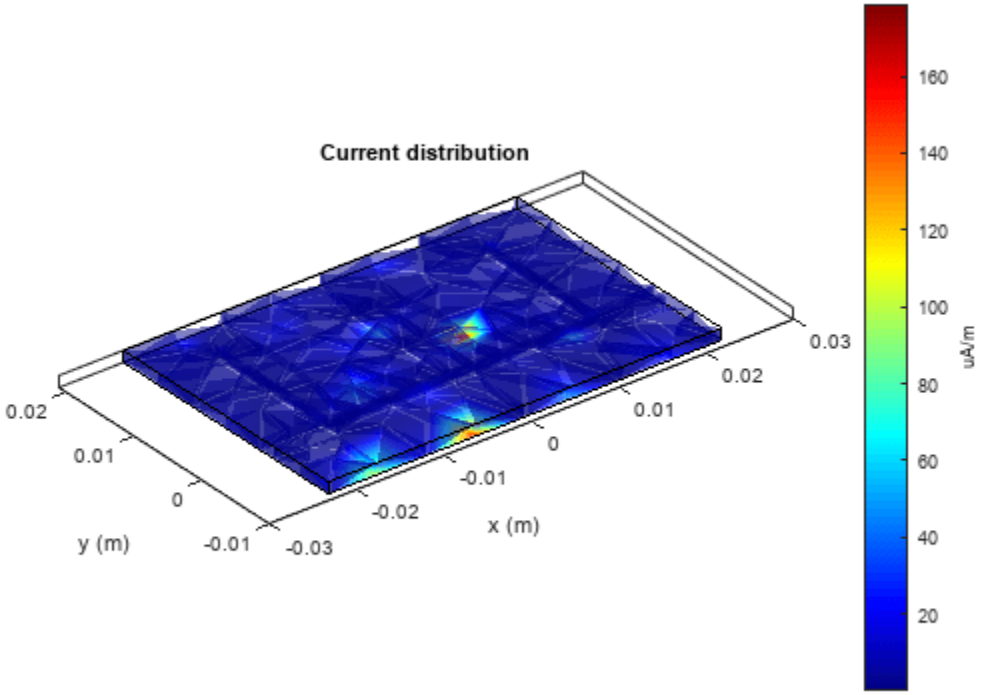


Use the `current` function to visualize the current distribution on the metal surface and the volume polarization currents on dielectric of band stop filter

```
figure;
current(pcb,2.4e9);
```




```
figure;
current(pcb,2.4e9,'dielectric');
```



References

[1] Jia-Sheng Hong "Microstrip Filters for RF/Microwave Applications", p. 184, John Wiley & Sons, 2nd Edition, 2011.

Design and Analyze HighPass Filter Using pcbComponent

This example shows how to design and analyze highpass filter using the pcbComponent object. A microstrip highpass filter is designed based on a three-pole (n = 3) Chebyshev highpass prototype with 0.1 dB passband ripple and cutoff frequency f_c = 1.5 GHz.

Design and Analyze HighPass Filter

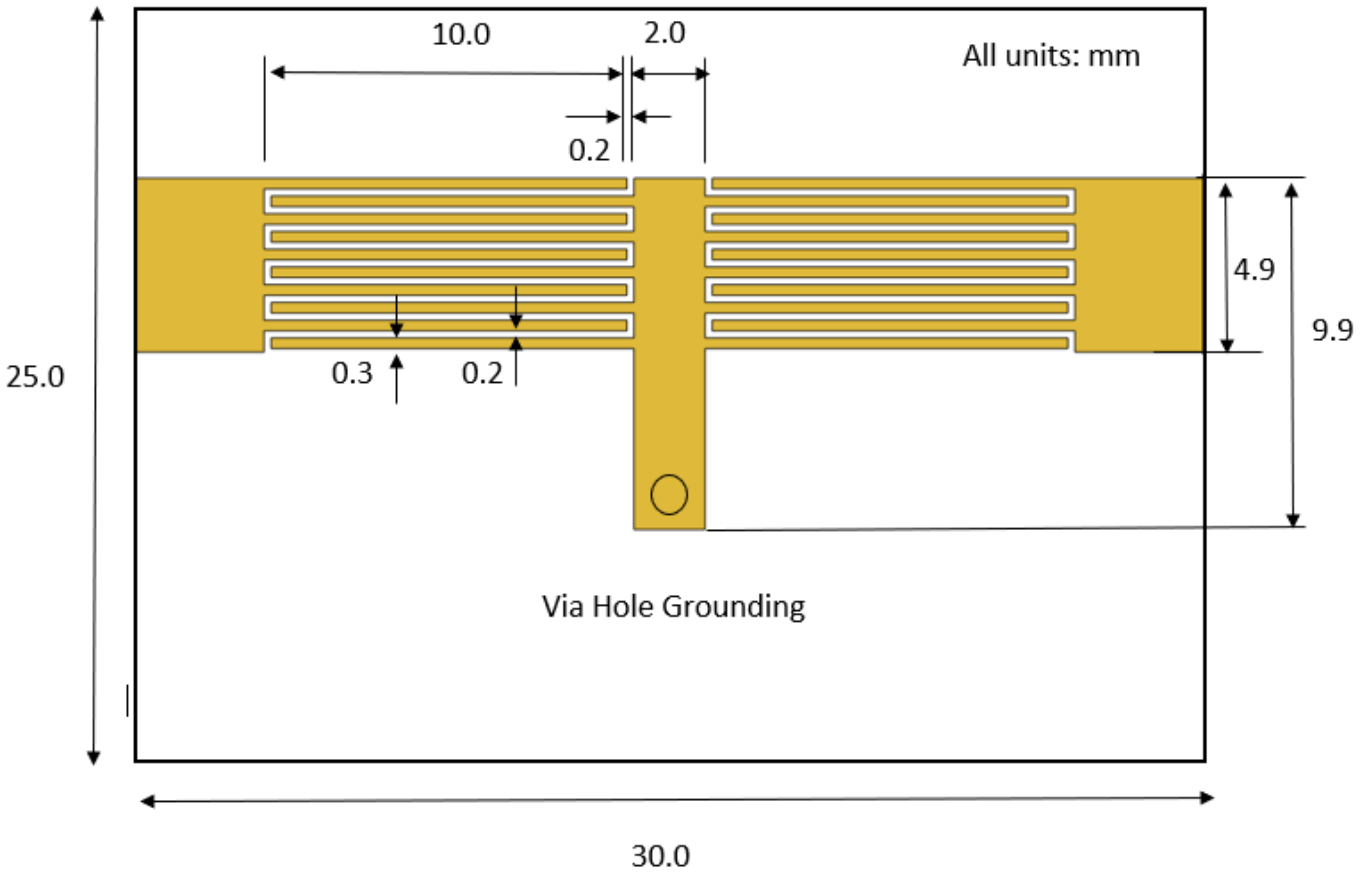
The element values of the corresponding low pass Chebyshev prototype are taken from Table 3.2 of reference [1]

- $g_0 = g_4 = 1.0$
- $g_1 = g_3 = 1.0316$
- $g_2 = 1.1474$

Using the design equations Eqs. (6.2) and (6.3) given in reference [1] for n = 3 and Z_0 = 50 ohm terminations, we can obtain quasi-lumped circuit elements values as

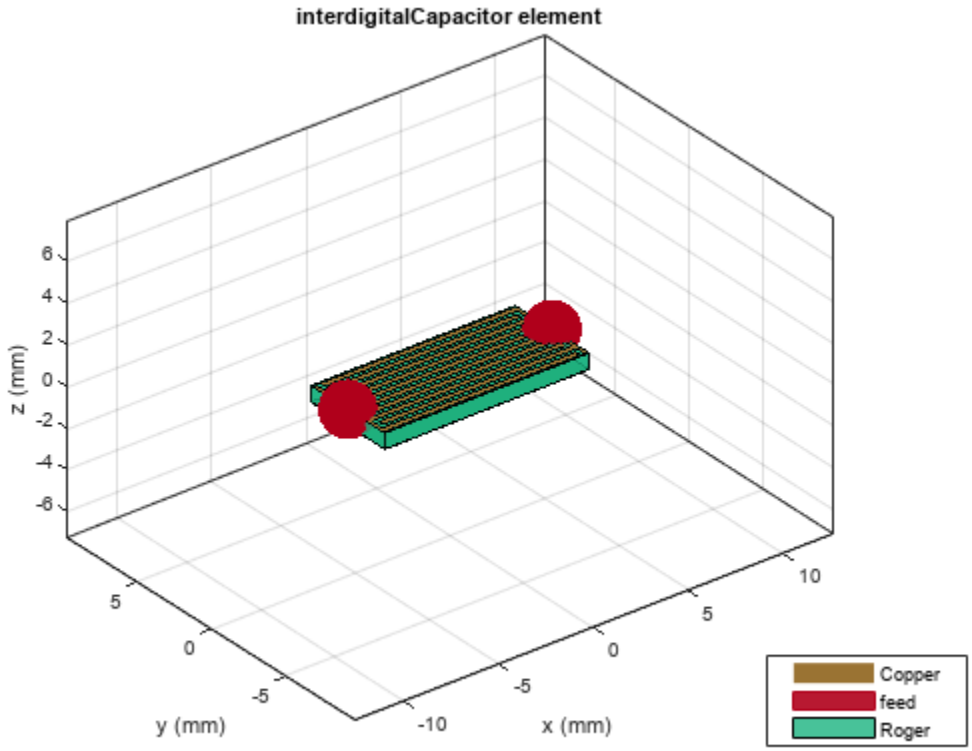
- $C_1 = C_3 = 2.0571$ pF
- $L_2 = 4.6236$ nH

A schematic diagram of such a highpass filter taken from fig 6.2 of reference [1] representing various feature dimensions is shown below. It is seen that the series capacitors C_1 and C_3 are realized by identical interdigital capacitors, while the shunt inductor L_2 is realized by a short-circuited stub. A commercial substrate (`RT/D 5880`) having a relative dielectric constant of 2.2 and thickness of 1.57 mm is chosen for this microstrip filter realization. The dimensions of the interdigital capacitors, such as the finger length, width, spacing between fingers, and number of the fingers are given in the fig 6.2 of reference [1]. The procedure to calculate the dimensions of interdigital capacitor and stub inductor is discussed in the section 6.1.1 of reference [1]. A closed-form design formulation for interdigital capacitor and alternatively full-wave EM simulations are used to extract desired values of lumped circult capacitance for different feature dimensions.



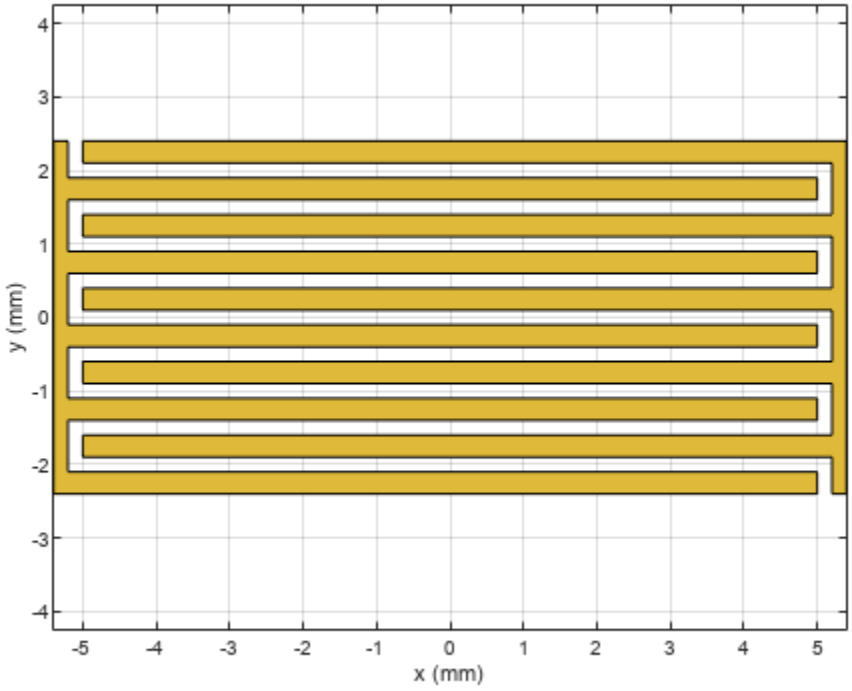
Use the `interdigitalCapacitor` object and change its properties as per given values in reference [1] to create interdigited fingers. Visualize created object using `show`

```
idc = interdigitalCapacitor;
idc.NumFingers = 10;
idc.FingerLength = 10e-3;
idc.FingerWidth = 0.3e-3;
idc.FingerSpacing = 0.2e-3;
idc.FingerEdgeGap = 0.2e-3;
idc.GroundPlaneWidth = 5e-3;
idc.TerminalStripWidth = 0.1e-3;
idc.PortLineLength = 0.1e-3;
idc.PortLineWidth = 4.8e-3;
figure;
show(idc);
```



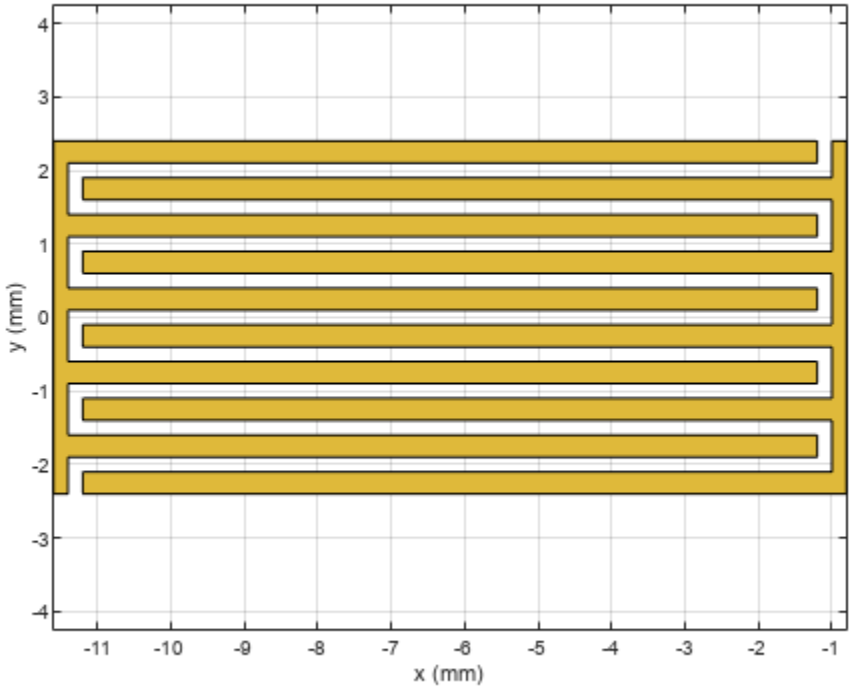
Use the `pcbComponent` on `idc` object to create a capacitor `cap1`. Visualize `cap1` using `show`

```
pcb = pcbComponent(idc);
cap1 = pcb.Layers{1};
figure;
show(cap1);
```



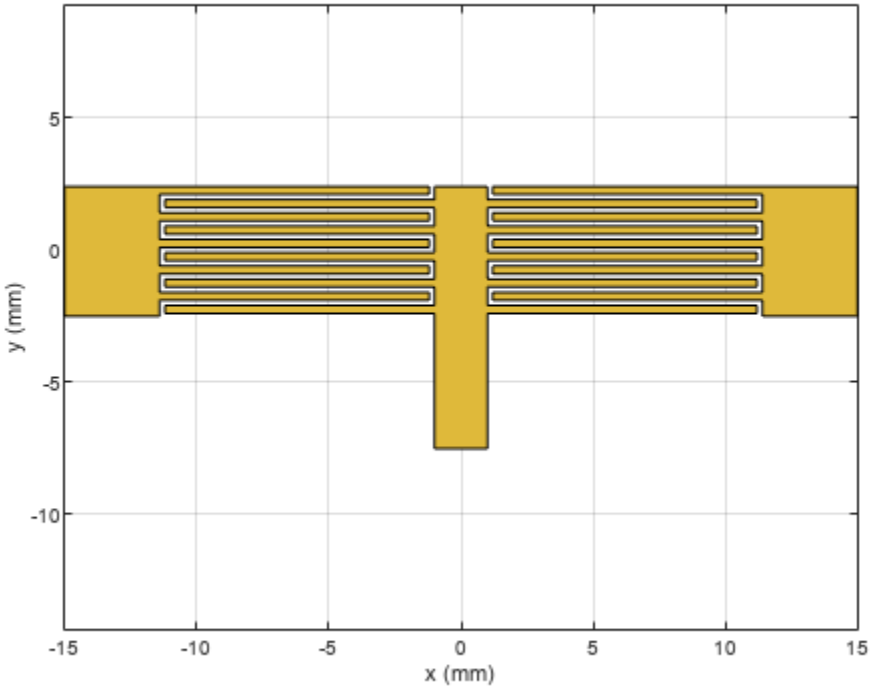
Use the `copy`, `rotateZ`, `rotateX` and `translate` operation methods on capacitor `cap1` object to create capacior `cap2`. Visualize `cap2` using `show`

```
cap2 = copy(cap1);
cap2 = rotateZ(cap2,180);
cap2 = rotateX(cap2,180);
cap2 = translate(cap2,[-6.2e-3 0 0]);
cap1 = translate(cap1,[6.2e-3 0 0]);
figure;
show(cap2);
```



Use the `traceRectangular` object to create both feeding ports `port1`, `port2` and short circuited stub `centerArm`. Perform a Boolean add operation for the shapes `port1`, `cap2`, `centerArm`, `cap1` and `port2` to create `filter`. Visualize `filter` using `show`

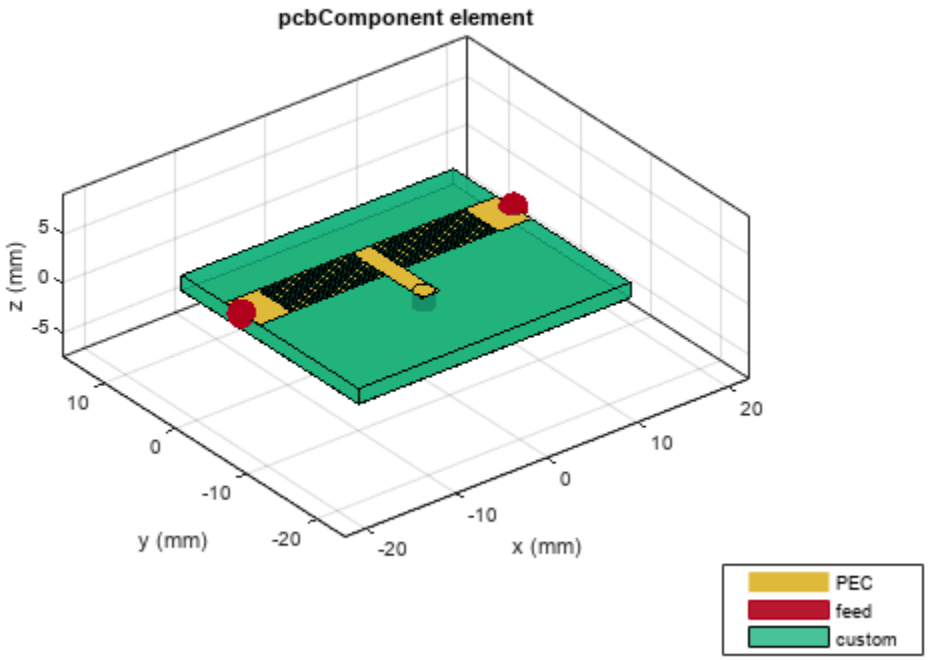
```
portW = 4.9e-3;
portL = 3.6e-3;
centerL = 2e-3;
centerW = 9.9e-3;
port1 = traceRectangular(Length = portL,Width = portW,Center = [-11.4e-3-portL/2 -0.05e-3]);
port2 = traceRectangular(Length = portL,Width = portW,Center = [11.4e-3+portL/2 -0.05e-3]);

centerArm = traceRectangular(Length = centerL,Width = centerW,Center = [0 -2.55e-3]);
filter = port1 + cap2 + centerArm + cap1 + port2;
figure;
show(filter);
```



Define the substrate parameters and create a dielectric to use in the `pcbComponent` of the designed filter. Create a groundplane using the `traceRectangular` shape.

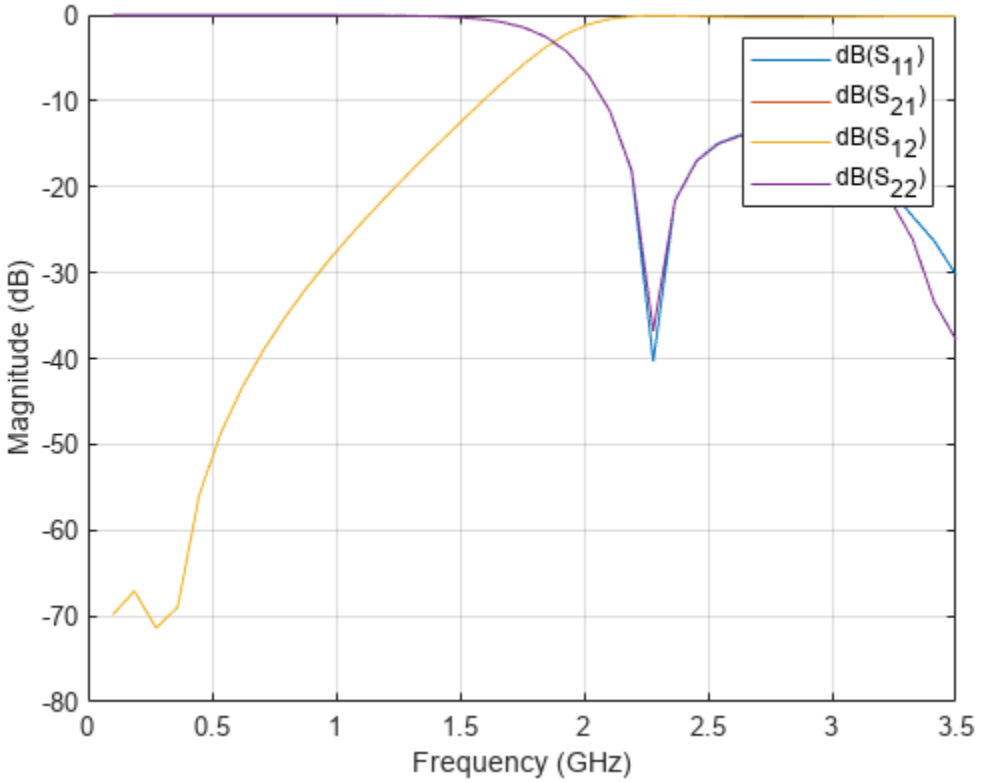
Use the `pcbComponent` to create a filter PCB. Assign the dielectric and ground plane to the `Layers` property on `pcbComponent`. Assign the `FeedLocations` to the edge of the feed ports. Assign `ViaLocations` at the edge of stub `centerArm`. Set the `BoardThickness` to 1.57 mm on the `pcbComponent` and visualize the filter. The below code performs these operations and creates the filter `pcb` object.

```
substrate = dielectric(EpsilonR = 2.2, LossTangent = 0.0009,...
    Name = "custom", Thickness = 1.57e-3);
gndL = 30e-3;
gndW = 25e-3;
ground = traceRectangular(Length = gndL, Width = gndW,...
    Center = [0,-4e-3]);

pcb = pcbComponent;
pcb.BoardShape = ground;
pcb.BoardThickness = 1.57e-3;
pcb.Layers ={filter,substrate,ground};
pcb.FeedDiameter = portW/2;
pcb.FeedLocations = [-gndL/2 0 1 3;gndL/2 0 1 3];
pcb.ViaDiameter = centerL;
pcb.ViaLocations = [0 -6.5e-3 1 3];
figure;
show(pcb);
```



Use the `sparameters` method to calculate the S-parameters for the highpass filter and plot it using the `rfplot` function.

```
spar = sparameters(pcb,linspace(0.1e9,3.5e9,40));
figure;
rfplot(spar);
```

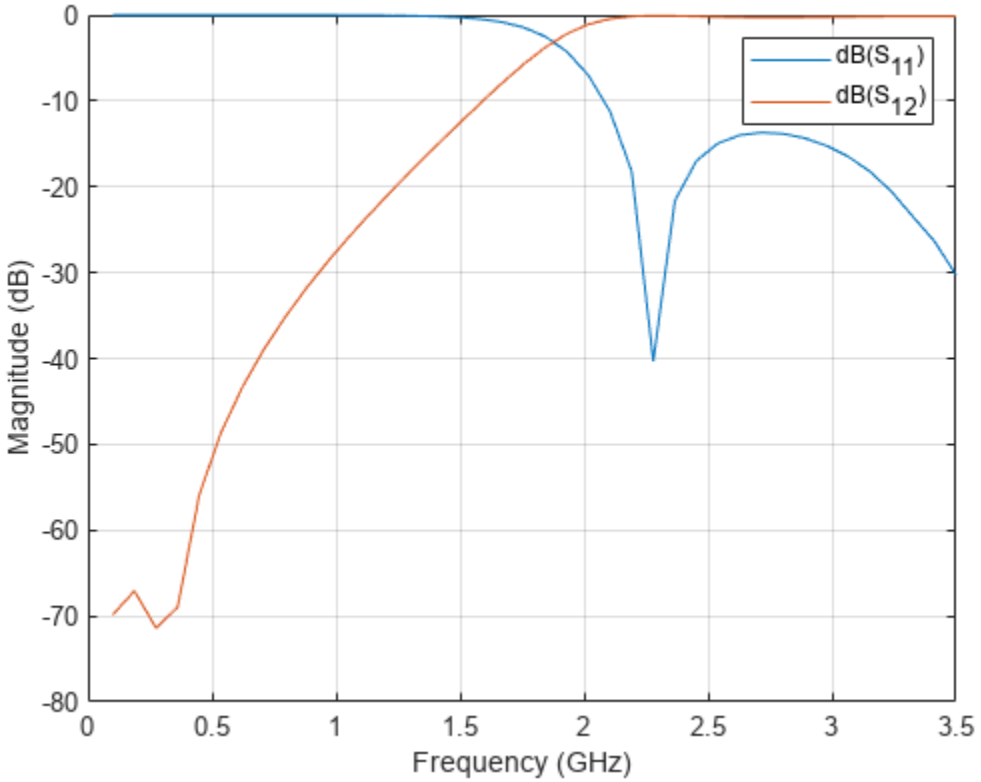


As there are four curves in the result, let us analyze the results.

Plot S-Parameters

Analyze the values of S_{11}, and S_{12} to understand the behavior of highpass filter.

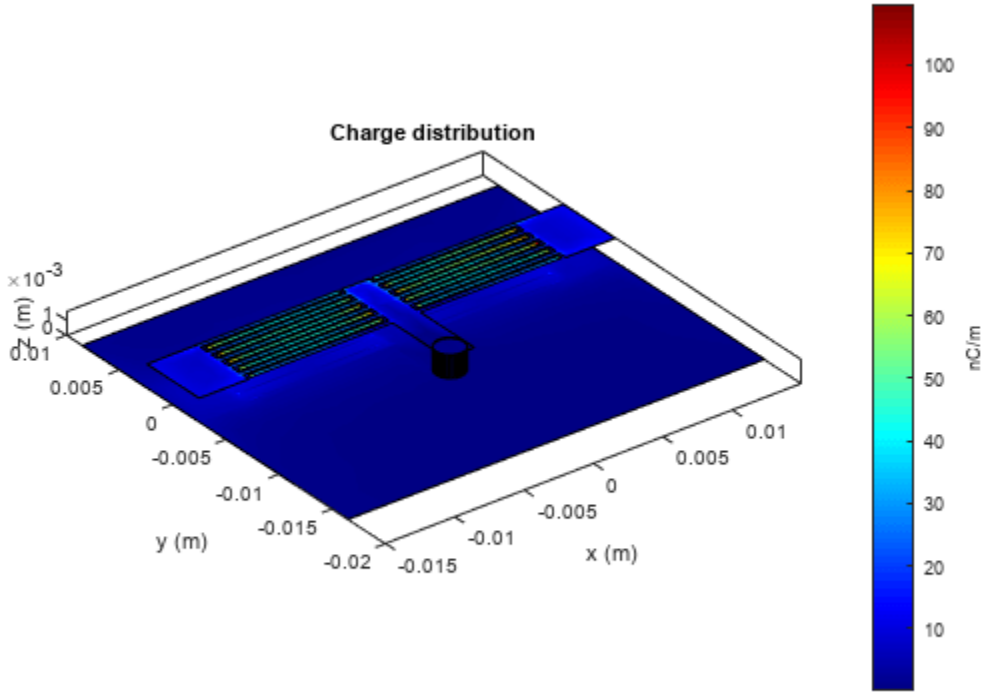
```
figure;
rfplot(spar,1,1);
hold on;
rfplot(spar,1,2);
hold on;
```



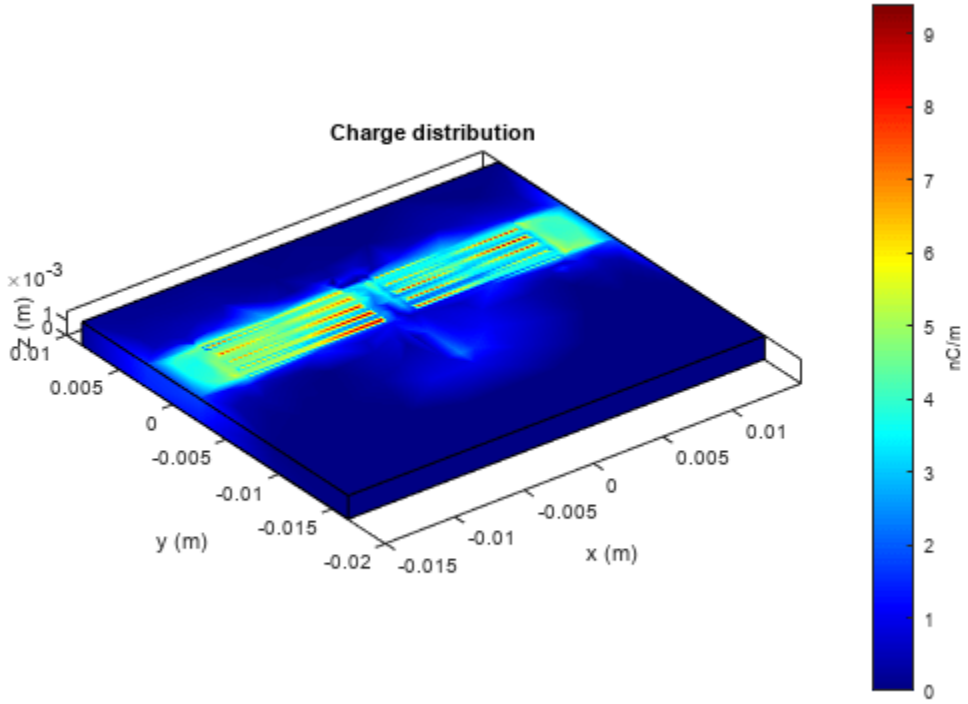
The result shows that the filter has -3 dB cut-off frequency f_c = 1.8 GHz . The S_{12} values are greater than -3 dB and S_{11} values are less than -10 dB above -3 dB cut-off frequency 1.8 GHz. The designed filter therefore has highpass response and em simulated results matches closely with reference. The 0.3 GHz shift in -3 dB cut-off frequency might be due to use of different numerical solver.

Use the `charge` method to visualize the charge distribution on the metal surface and dielectric of highpass filter

```
figure;
charge(pcb,3e9);
```

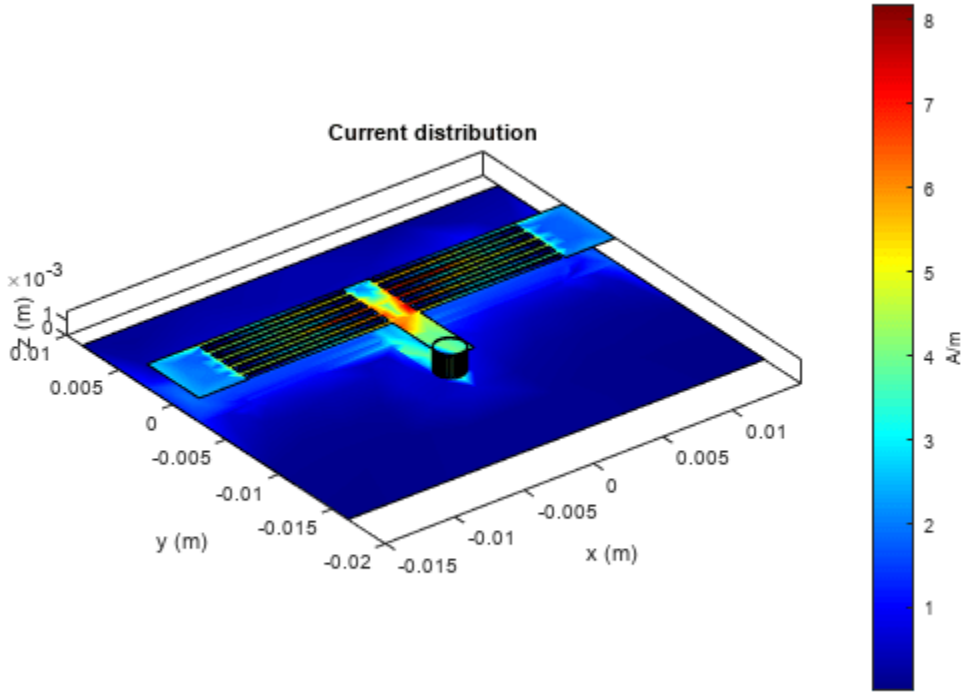



```
figure;
charge(pcb,3e9,'dielectric');
```

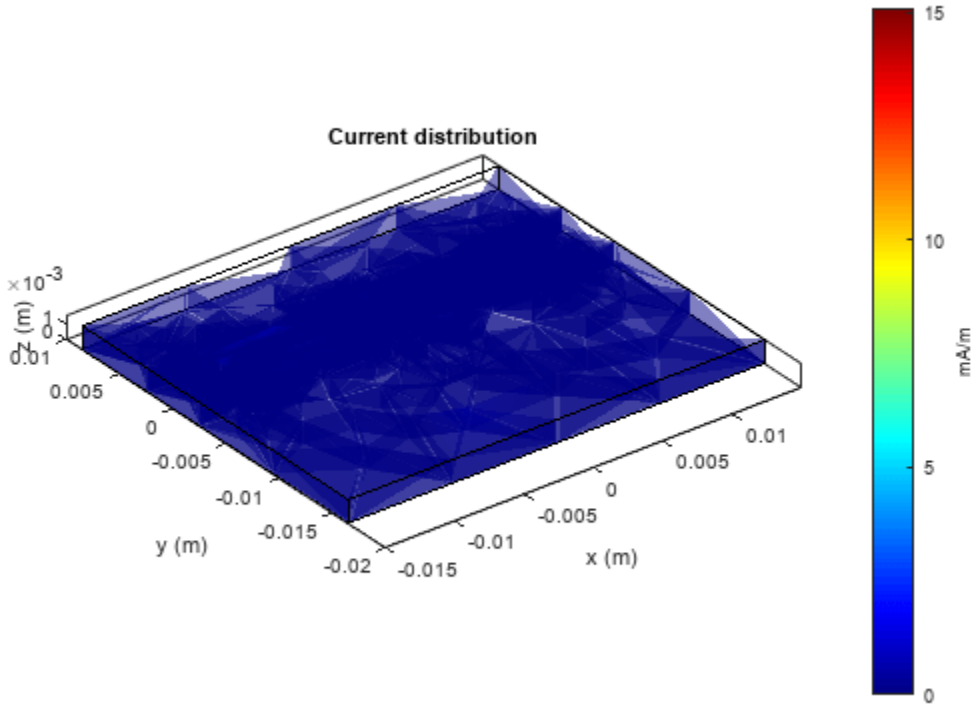


Use the `current` method to visualize the current distribution on the metal surface and volume polarization currents on dielectric of highpass filter

```
figure;
current(pcb,3e9);
```




```
figure;
current(pcb,3e9,'dielectric');
```



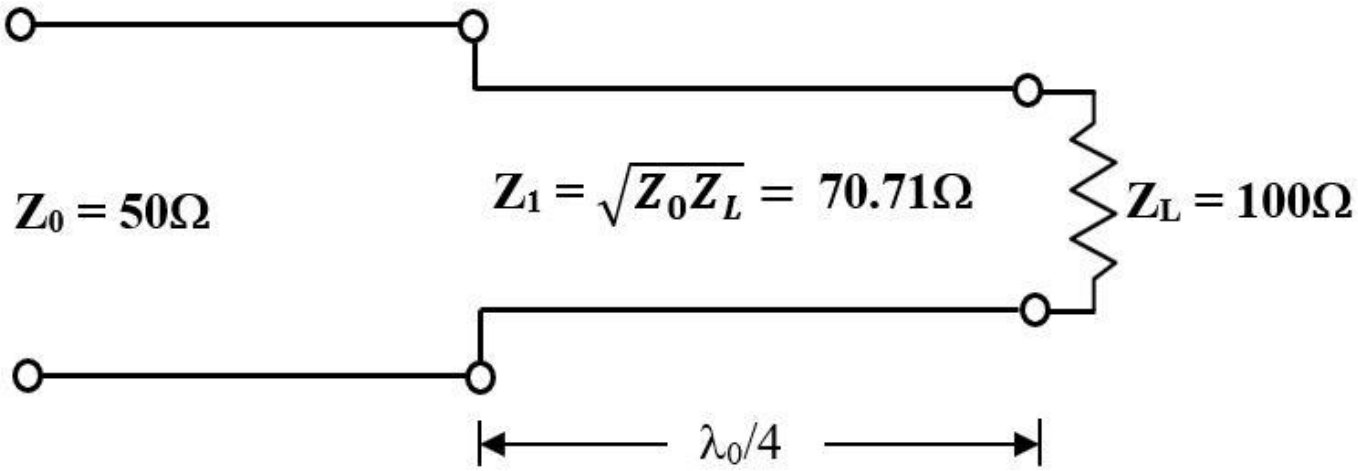
References

[1] Jia-Sheng Hong "Microstrip Filters for RF/Microwave Applications", p. 165, John Wiley & Sons, 2nd Edition, 2011.

Design of Quarter-Wave Transformer for Impedance Matching Applications

This example shows how to design the Quarter-wave transformer for impedance matching applications by using the `pcbComponent, microstripLine,` and `traceRectangular` object in the RF PCB Toolbox.

Quarter-wave transformer is a simple and useful circuit for matching the real impedance of a terminating load (Z_L) to the characteristic impedance of the feeding transmission line (Z_0) as depicted in the given figure. The characteristic impedance of the quarter-wave transformer is calculated as $Z_1 = \sqrt{(Z_0 Z_L)}$ [1]. This example is to design a single section quarter-wave transformer to match the100 Ω load to a 50 Ω transmission line at an operating frequency of 2 GHz. The calculated characteristic impedance of the quarter-wave transformer $Z_1$is 70 . 71 Ω.



Design of Single Section Quarter-Wave Transformer

Use the `design` function on the `microstripLine` object to create the 50 Ω input transmission line and70 . 71 Ω quarter-wave transformer's `Length` and `Width` dimension for the operating frequency of 2 GHz. The default substrate for `microstripLine` is Teflon with thickness of 1 . 6 mm.

```
freq = 2e9;
m50 = design(microstripLine,freq,Z0=50,LineLength=0.05); % input transmission line
m70 = design(microstripLine,freq,Z0=70.71,LineLength=0.25); % section 1
```

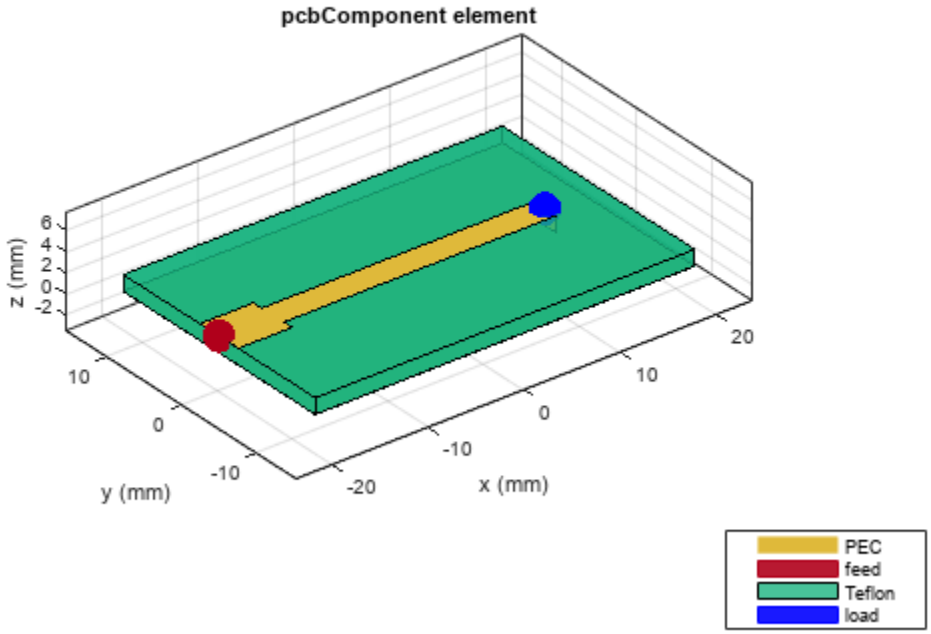
Use the `traceRectangular` object to create the groundplane, input transmission line, and quarter-wave transmission line shapes.

```
% ground plane dimension
gndL = 2*m50.Length+m70.Length;
gndW = 5*m50.Width;

ground = traceRectangular("Length",gndL,"Width",gndW);
% input transmission line
Z0 = traceRectangular("Length",m50.Length,"Width",m50.Width,...
    "Center",[-gndL/2+m50.Length/2 0]);
% First section Quarter-wave transformer
```

```
Z1 = traceRectangular("Length",m70.Length,"Width",m70.Width,"Center",...
    [-gndL/2+m50.Length+m70.Length/2 0]);

qline = Z0 + Z1;
```

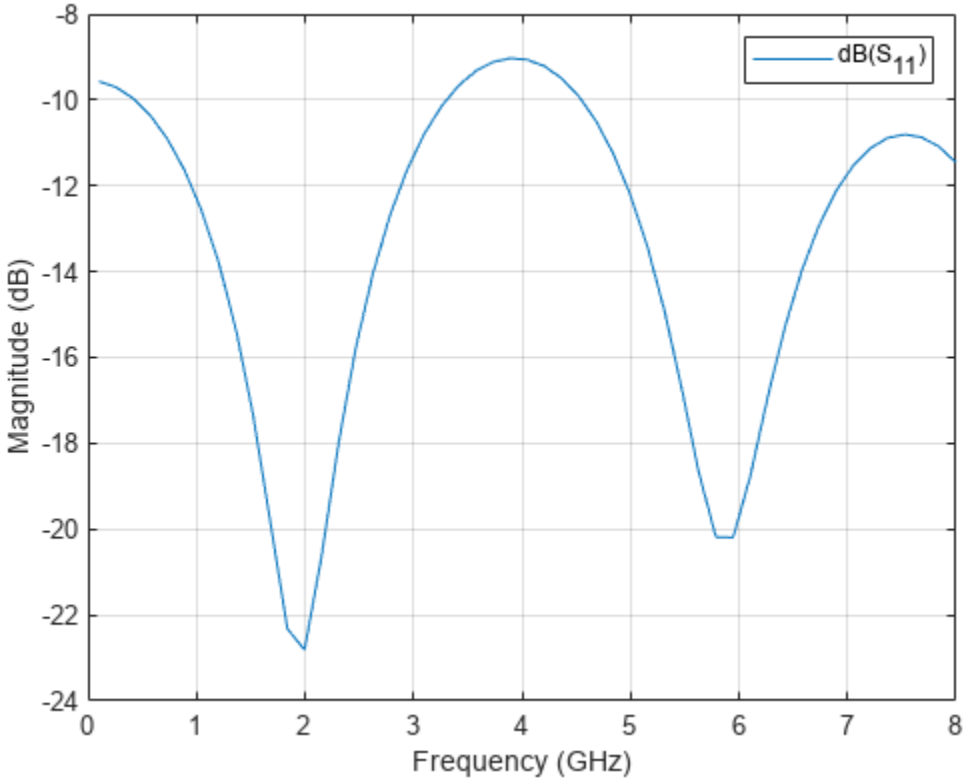
Use the `pcbComponent` to create the quarter-wave transformer and use the `lumpedElement` for the terminating load of 100 Ω and place it at the end of the quarter-wave transformer and use a via to connect the load to ground.

```
pcb =pcbComponent;
pcb.BoardShape = ground;
pcb.BoardThickness = m50.Height;
pcb.Layers = {qline,m50.Substrate,ground};
pcb.FeedDiameter = m50.Width/2;
pcb.FeedLocations = [-gndL/2 0 1 3];
pcb.ViaLocations = [-gndL/2+m50.Length+m70.Length,0,1,3];
pcb.ViaDiameter = m70.Width/2;
% Load
ZL = lumpedElement;
ZL.Impedance = 100;
ZL.Location = [-gndL/2+m50.Length+m70.Length,0,pcb.BoardThickness];
pcb.Load = ZL;
% show the single section quarter-wave transformer
figure;show(pcb)
```



Use the `sparameters` function to calculate the S parameters and plot it using the `rfplot` function.

```
sparams = sparameters(pcb,linspace(100e6,8e9,51));
figure; rfplot(sparams)
```



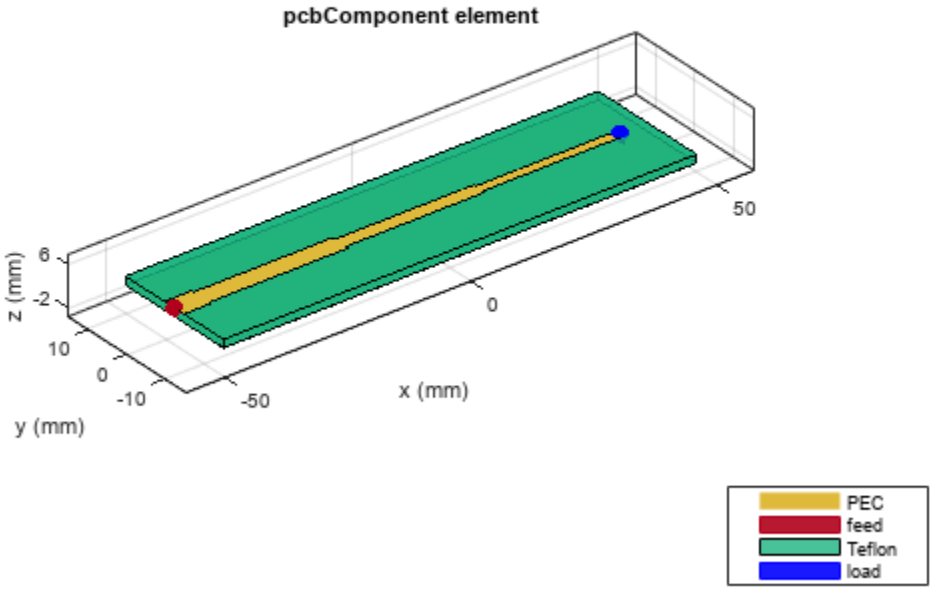
It is observed from the S_{11} values that the impedance is perfectly matched at the desired frequency of 2 GHz where the value of magnitude is less than -22 dB with the bandwith of 2800 MHz. Matching alsol occurs at frequencies where the Z_1has a length of $(2n+1)\lambda_0/4, n = 0, 1, 2, 3 \ldots$

Design of Multisection Quarter-Wave Transformer

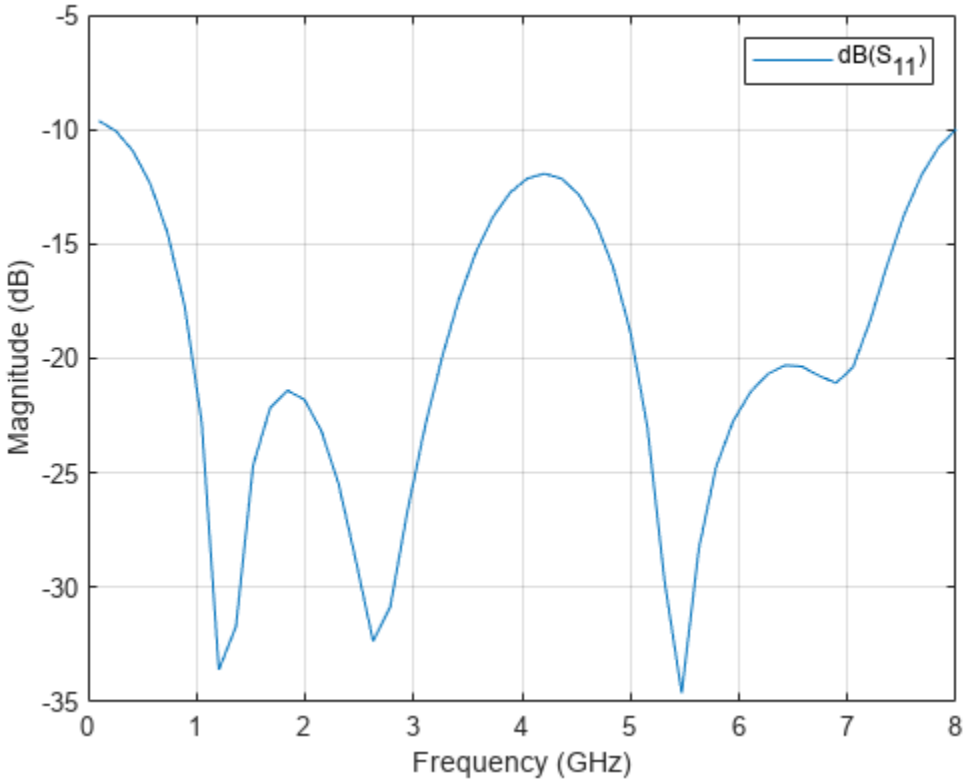
In general, the single section transformer may be sufficient for narrow band impedance match. This transformer can be extended to multisection in a methodical manner to yield optimum matching characteristics over a wider bandwidth [1]. The following example is for the design of three section Chebyshev matching transformer to match a $100\,\Omega$ load to a $50\,\Omega$ with ripple level = 0.05. Each section's characteristic impedances are computed using the formulas given in [1] and the values are $Z_1 = 57.5\Omega$, $Z_2 = 70.7\Omega$ and $Z_3 = 87\Omega$. The three section Chebyshev matching transformer is designed by using same steps described in the design of single section quarter-wave transformer.

```
m50 = design(microstripLine,freq,"Z0",50,"LineLength",0.05); % input transmission line
m57 = design(microstripLine,freq,"Z0",57.5,"LineLength",0.25); % section 1
m70 = design(microstripLine,freq,"Z0",70.7,"LineLength",0.25); % section 2
m87 = design(microstripLine,freq,"Z0",87,"LineLength",0.25); % section 3
% ground plane dimension
gndL = 2*m50.Length+m57.Length+m70.Length+m87.Length;
gndW = 5*m50.Width;

ground = traceRectangular("Length",gndL,"Width",gndW);
```

```
% inpt transmission line
Z0 = traceRectangular("Length",m50.Length,"Width",m50.Width,...
    "Center",[-gndL/2+m50.Length/2 0]);
% First section Quarter-wave transformer
Z1 = traceRectangular("Length",m57.Length,"Width",m57.Width,"Center",...
    [-gndL/2+m50.Length+m57.Length/2 0]);
% Second section Quarter-wave transformer
Z2 = traceRectangular("Length",m70.Length,"Width",m70.Width,"Center",...
    [-gndL/2+m50.Length+m57.Length+m70.Length/2 0]);
% Third section Quarter-wave transformer
Z3 = traceRectangular("Length",m87.Length,"Width",m87.Width,"Center",...
    [-gndL/2+m50.Length+m57.Length+m70.Length+m87.Length/2 0]);

qline = Z0+Z1+Z2+Z3;
% create pcbComponent
pcb =pcbComponent;
pcb.BoardShape = ground;
pcb.BoardThickness = m50.Height;
pcb.Layers ={qline,m50.Substrate,ground};
pcb.FeedDiameter = m50.Width/2;
pcb.FeedLocations = [-gndL/2 0 1 3];
pcb.ViaLocations = [-gndL/2+m50.Length+m57.Length+m70.Length+m87.Length,0,1,3];
pcb.ViaDiameter = m87.Width/2;
% Load
ZL = lumpedElement;
ZL.Impedance = 100;
ZL.Location = [-gndL/2+m50.Length+m57.Length+m70.Length+m87.Length,0,pcb.BoardThickness];
pcb.Load = ZL;
% show the three section quarter-wave transformer
figure;show(pcb)
```

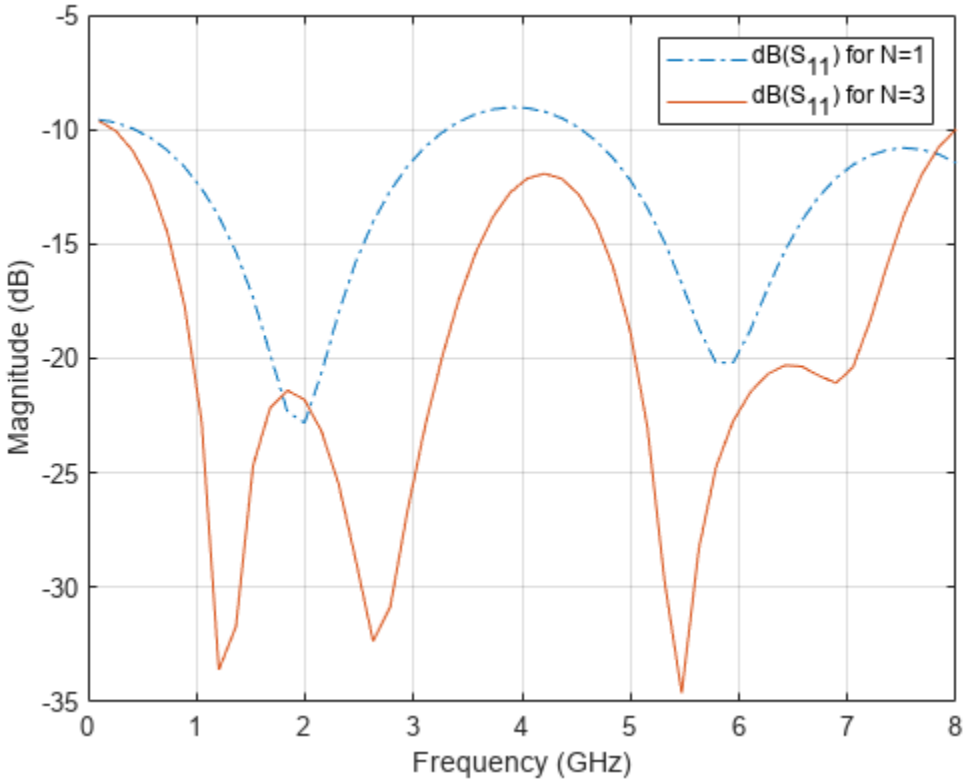



```
% Analysis
sparams3 = sparameters(pcb,linspace(100e6,8e9,51));
figure; rfplot(sparams3)
```



Comparison of S parameters

```
figure; rfplot(sparams,'-.')
hold on
rfplot(sparams3)
legend('dB(S_{11}) for N=1','dB(S_{11}) for N=3','Location','northeast')
```



From the comparison the s-parameters plot for single section and three section quarter-wave transformers designed at 2 GHz, it is observed that the bandwidth achieved for three section quarter-wave transformer is 7750 MHz with improved impedance matching characteristics.Therefore, it is evident that wider bandwidth is achieved for quarter-wave transformers with multiple sections.

References

[1] David M. Pozar, Microwave Engineering, pp. 246-261, 4th Edition, John Wiley & Sons, 2012.

Corporate Feed Divider Network for a Linear Patch Antenna array

This example shows how to integrate a corporate power divider with a microstrip patch antenna array. The corporate power divider is available as a catalog element in RF PCB Toolbox. The patch antenna array is built using the `patchMicrostripInsetfed` catalog element and `pcbStack` from Antenna Toolbox.

Create Variables

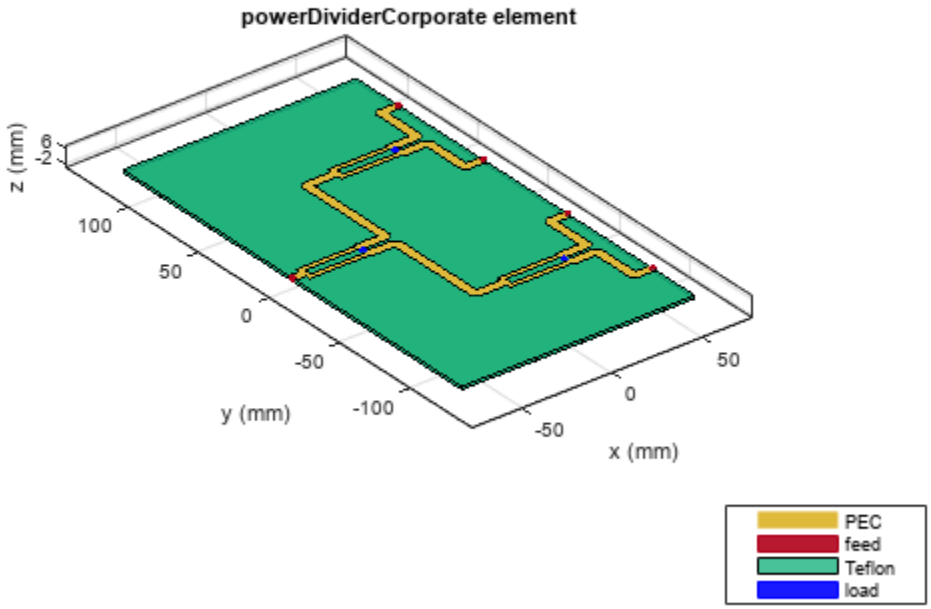
Create the common variables.

```
f = 5e9;
c = physconst('lightspeed');
```

Create Corporate Power Divider

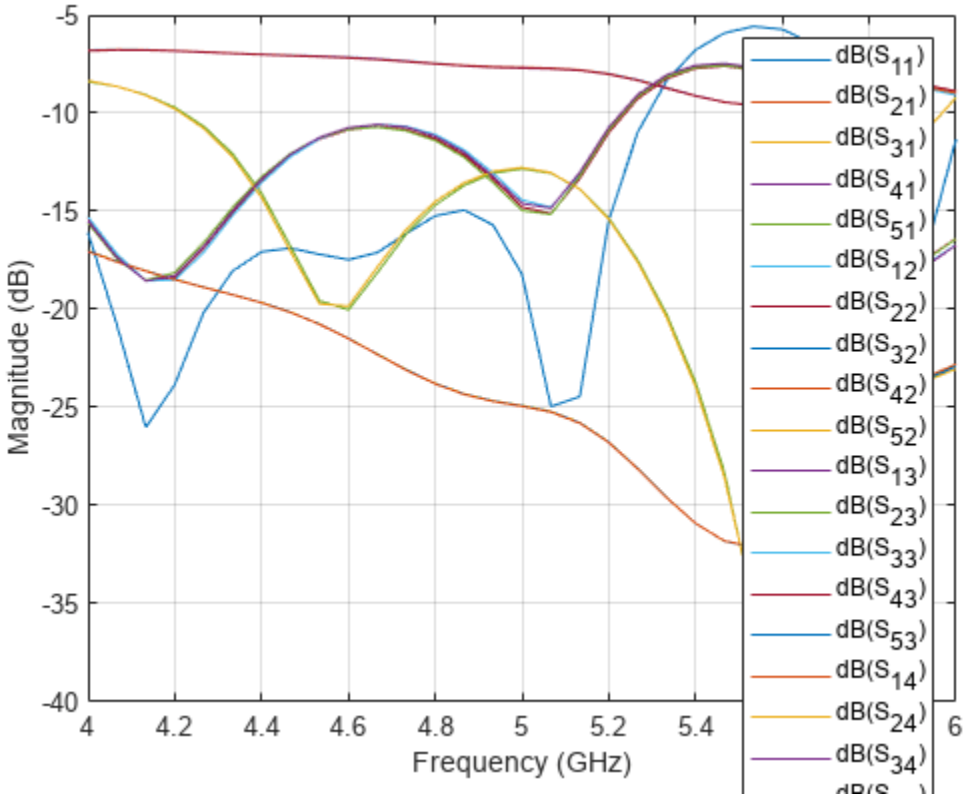
Use the `powerDividerCorporate` object to design the corporate power divider at 5 GHz and adjust the spacing to wavelength. Use the `show` function to visualize it.

```
pdc = powerDividerCorporate;
pdc = design(pdc,f);
pdc.SplitterElement.Substrate.LossTangent = 0;
pdc.PortSpacing = c/f;
pdc.PortLineLength = 10e-3;
pdc.GroundPlaneWidth = pdc.PortSpacing*4;
figure,show(pdc);
```



Use the `sparameters` function with `Behavioral` set to `true`.Plot s-parameters using the `rfplot` function. As the computation takes time, load the pre-computed mat file and plot the result using `rfplot`.

```
% out = sparameters(pdc,linspace(4e9,6e9,31),'Behavioral',true);
load pdc.mat
figure;
rfplot(sparPDC);
```



Create Inset Fed Patch Antenna

Use the `patchMicrostripInsetfed` object to design a microstrip inset fed patch that resonates at 5 GHz.

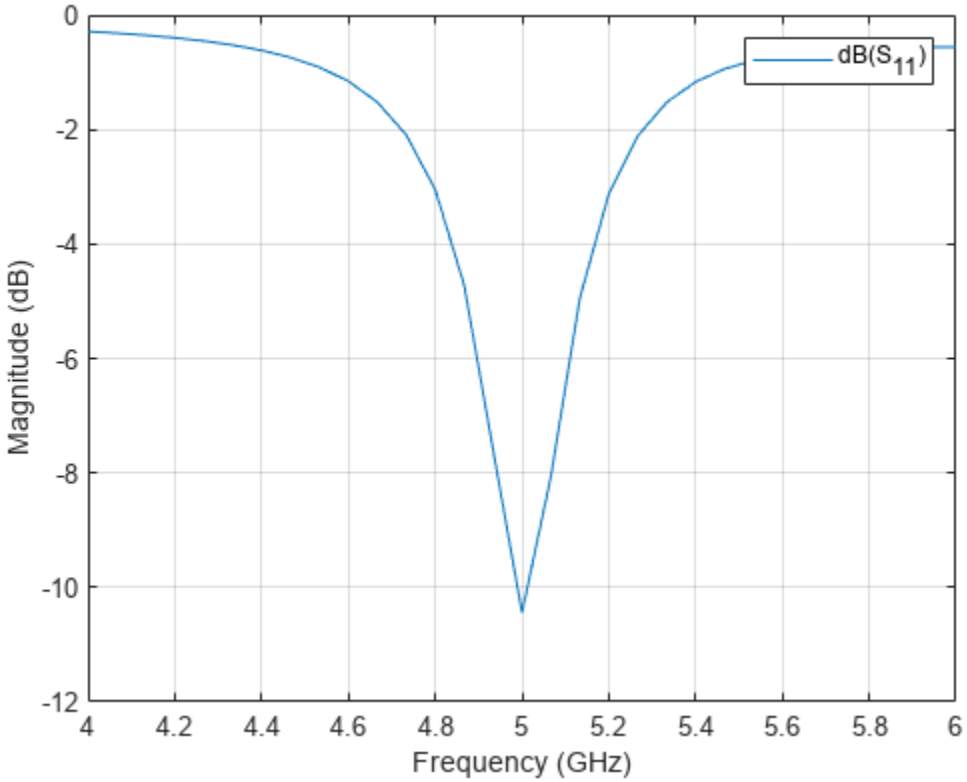
```
ant = patchMicrostripInsetfed;
ant.Substrate = dielectric('Teflon');
ant = design(ant,5e9);
ant.Height = 1.6e-3;
ant.Length = 20.5e-3;
ant.Width = 20.5e-3;
ant.StripLineWidth = pdc.PortLineWidth;
ant.NotchWidth = 7.1e-3;
ant.NotchLength = 5.9e-3;
```

Use the `sparameters` function to calculate the sparameters. As it takes more time to complete the simulation the pre-computed result is loaded.

```
% out = sparameters(ant,linspace(4e9,6e9,31));
```

Load the mat file and plot the sparameters using `rfplot` function.

```
load patch.mat
figure,rfplot(out);
```



Create pcbStack for Antenna

Use the `pcbStack` object to convert the patch antenna into a pcb stack so that the shape of the top layer can be extracted from the `Layers` property. Convert into a linear array of 4 elements. The patch is extracted from the `Layers` property of the `pcbStack` and the shape is copied and converted into a linear array.

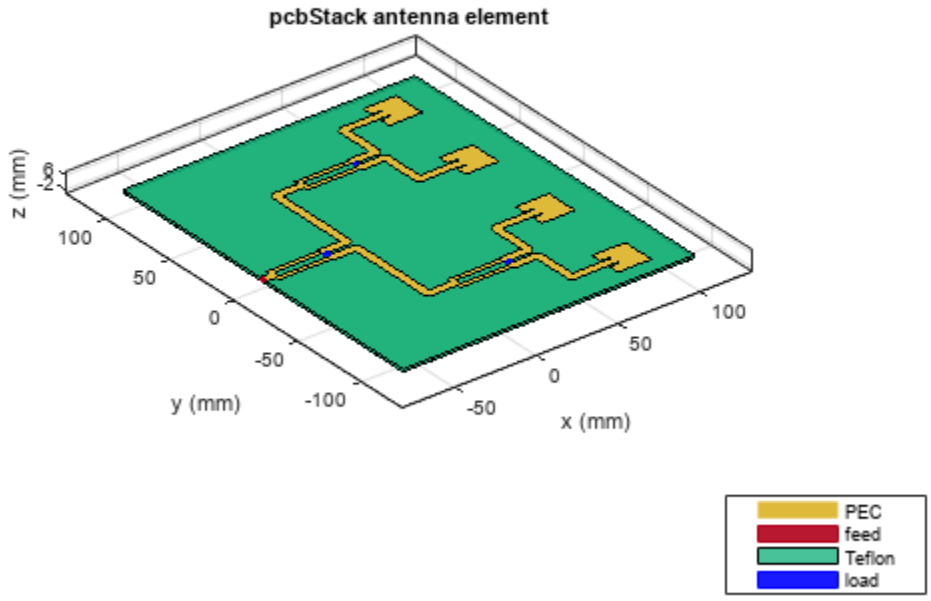
```
pcbant = pcbStack(ant);
TopLayer = pcbant.Layers{1};
TopLayer1 = copy(TopLayer);
for i = 2:4
    a = copy(TopLayer1);
    a = translate(a,[0,pdc.PortSpacing*(i-1),0]);
    TopLayer = TopLayer+a;
end
TopLayer = translate(TopLayer,[-ant.FeedLocation(1)+pdc.GroundPlaneLength/2,-pdc.PortSpacing*1.5
```

Create pcbComponent for Corporate Power Divider

Use the `pcbComponent` object and convert the corporate power divider into a PCB Component. Create a new `pcbStack` object to construct the cascade of power divider corporate and patch antennas. Assign all the properties of the pcbComponent to the pcbStack and then add the top layer of the corporate power divider with the patch antenna array and visualize it.

```
pcbcomp = pcbComponent(pdc);

pcbant1 = pcbStack;
```

```
pcbant1.BoardShape = pcbcomp.BoardShape;
pcbant1.BoardThickness =pcbcomp.BoardThickness;
pcbant1.Layers = pcbcomp.Layers;
pcbant1.FeedDiameter = pcbcomp.FeedDiameter;
pcbant1.FeedViaModel = pcbcomp.FeedViaModel;
pcbant1.FeedLocations = pcbcomp.FeedLocations;
pcbant1.Load = pcbcomp.Load;

a = pcbant1.Layers{1};
finalShape = a+TopLayer;
pcbant1.Layers{1} = finalShape;
feed = pcbant1.FeedLocations(1,:);
pcbant1.FeedLocations = feed;
gnd = pcbant1.Layers{3};
gnd.Length = gnd.Length+50e-3;
gnd.Width   = gnd.Width-20e-3;
gnd.Center(1) = 50e-3/2;
pcbant1.Layers{3} = gnd;
pcbant1.BoardShape = gnd;
figure,show(pcbant1);
```

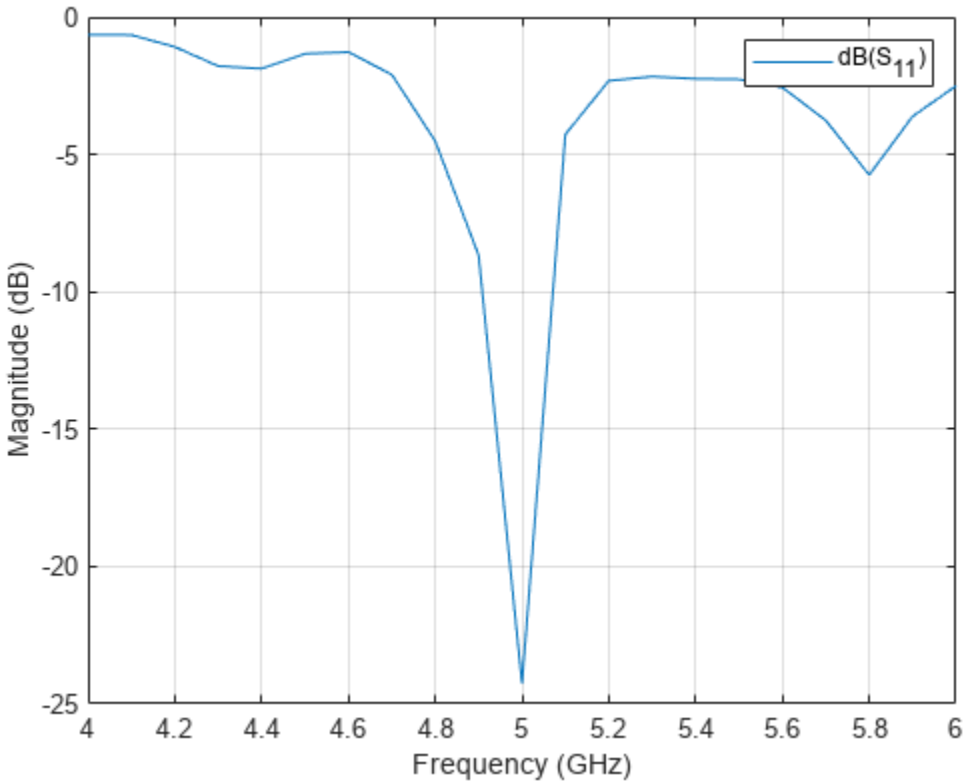


Use the `mesh` function to manually mesh the structure and use the `sparameters` function to compute the s-parameters. Generating a mesh and computing the result takes a lot of time as the structure is large compared to the wavelength. Hence the lines are commented out.

```
% figure,mesh(pcbant1,'MaxEdgeLength',6e-3);
% spar = sparameters(pcbant1,linspace(4e9,6e9,21));
% figure,pattern(pcbant1,4.9e9)
```

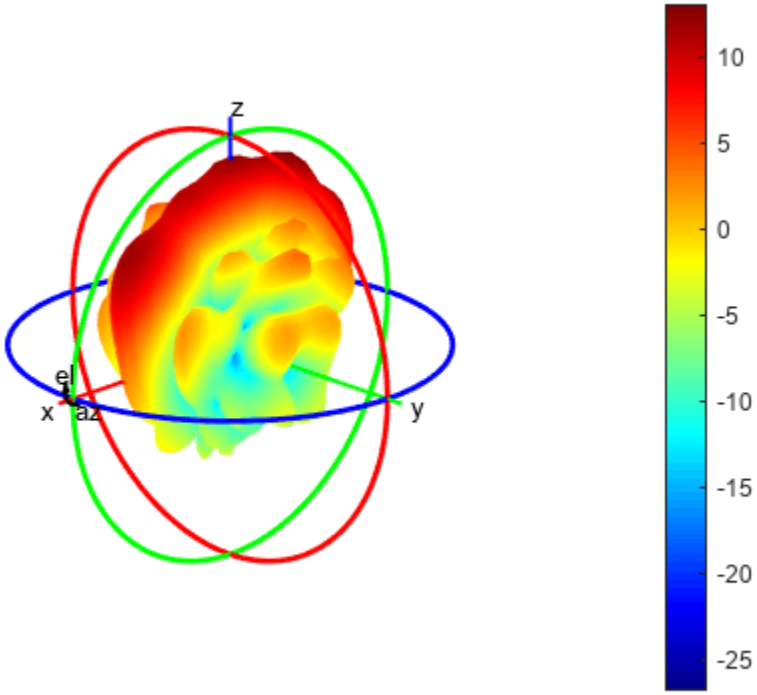
Load the precomputed result from `PDCpatch.mat` file and plot the results.

```
load PDCpatch.mat
figure;
rfplot(spar)
```



Use the `patternCustom` function to plot the 2-D or 3-D radiation pattern of an antenna magnitude, `magE` over the specified `phi` and `theta` angle vectors.

```
phi = az';
theta = (90-el);
MagE = pat';
figure;
patternCustom(MagE,theta,phi);
```


z
el
x
az
y
10
5
0
-5
-10
-15
-20
-25

Microstrip Composite Bandpass Filters for Ultra-Wideband (UWB) Wireless Communications

This example shows how to design and analyze a Low Pass Filter(LPF), High Pass Filter(HPF), and cascade responses of the LPF and HPF as well as two variations of a composite filter as shown in the paper[1].

The following steps are needed to create a filter:

1) Create the Geometry of the filter using RF PCB Shapes. Create a rectangle for the ground plane.

2) Use the `pcbComponent` and build the PCB stack. Set the `BoardShape` property of pcbComponent which will set the shape of the dielectric substrate.

3) Assign the filter shape, dielectric, and ground plane to the Layers property of pcbComponent.

4) Set the `FeedLocations` and `FeedDiameter`. If the design has Vias then set the `ViaLocations` and `ViaDiameter`.

5) Analyze the structure

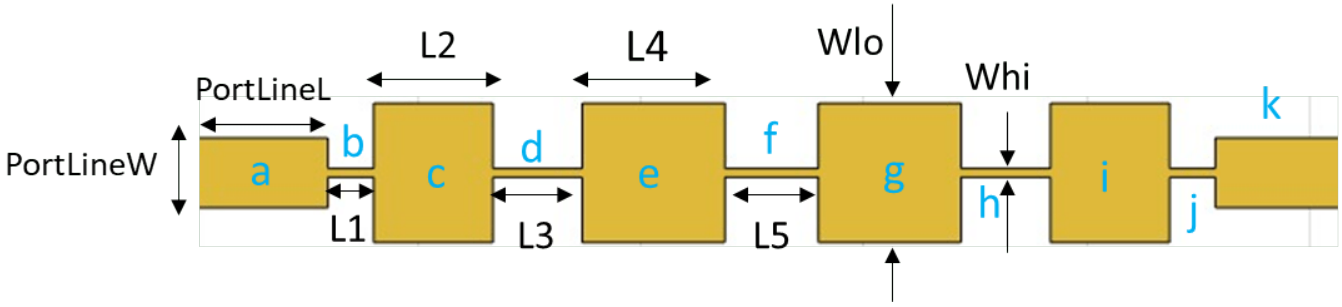
Create Variables

Define the variables required to create the geometry for all the filter designs

```
PortLineL = 3e-3;
PortLineW = 1.6e-3;
```

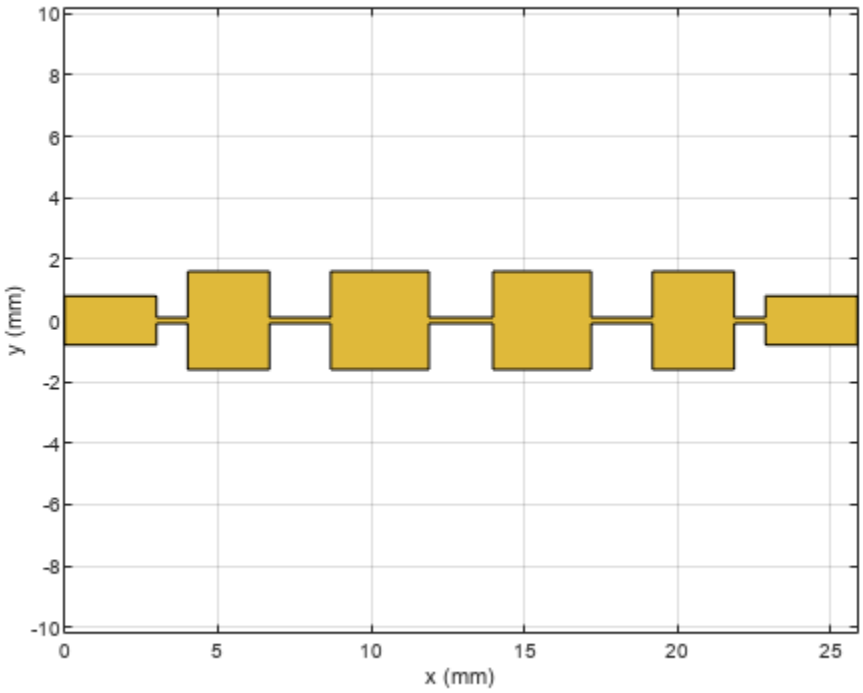
Low Pass Filter

Create the variables and assign the values given in the paper[1]. Use the `traceRectangular` shape to create all the rectangles as shown in the figure and join them to create the filter structure. Visualize the structure using the `show` function.




```
Whi = 0.2e-3;
Wlo = 3.2e-3;
L1  = 1.02e-3;
L2  = 2.67e-3;
L3  = 2e-3;
L4  = 3.2e-3;
L5  = 2.09e-3;

a = traceRectangular('Length',PortLineL,'Width',PortLineW,'Center',[PortLineL/2,0]);
```

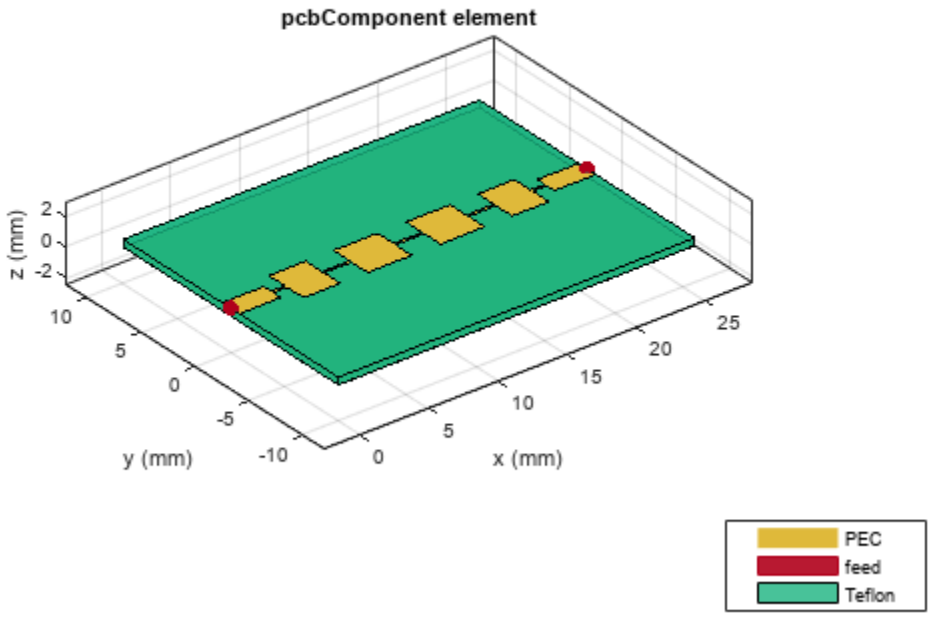
```
b = traceRectangular('Length',L1,'Width',Whi,'Center',[PortLineL+L1/2,0]);
c = traceRectangular('Length',L2,'Width',Wlo,'Center',[PortLineL+L1+L2/2,0]);
d = traceRectangular('Length',L3,'Width',Whi,'Center',[PortLineL+L1+L2+L3/2,0]);
e = traceRectangular('Length',L4,'Width',Wlo,'Center',[PortLineL+L1+L2+L3+L4/2,0]);
f = traceRectangular('Length',L5,'Width',Whi,'Center',[PortLineL+L1+L2+L3+L4+L5/2,0]);
g = traceRectangular('Length',L4,'Width',Wlo,'Center',[PortLineL+L1+L2+L3+L4+L5+L4/2,0]);
h = traceRectangular('Length',L3,'Width',Whi,'Center',[PortLineL+L1+L2+L3+L4+L5+L4+L3/2,0]);
i = traceRectangular('Length',L2,'Width',Wlo,'Center',[PortLineL+L1+L2+L3+L4+L5+L4+L3+L2/2,0]);
j = traceRectangular('Length',L1,'Width',Whi,'Center',[PortLineL+L1+L2+L3+L4+L5+L4+L3+L2+L1/2,0]
k = traceRectangular('Length',PortLineL,'Width',PortLineW,'Center',[PortLineL+L1+L2+L3+L4+L5+L4+
filt1 = a+b+c+d+e+f+g+h+i+j+k;
figure,show(filt1);
```



Create the PCB Stack of the filter using the `pcbComponent` object. Assign the filter shape, dielectric, and groundplane to the `Layers` property. Set the `BoardThickness` to 0.508 mm and assign the `BoardShape` to the ground plane. Set the `FeedDiameter` and `FeedLocations`. Visualize the PCB.

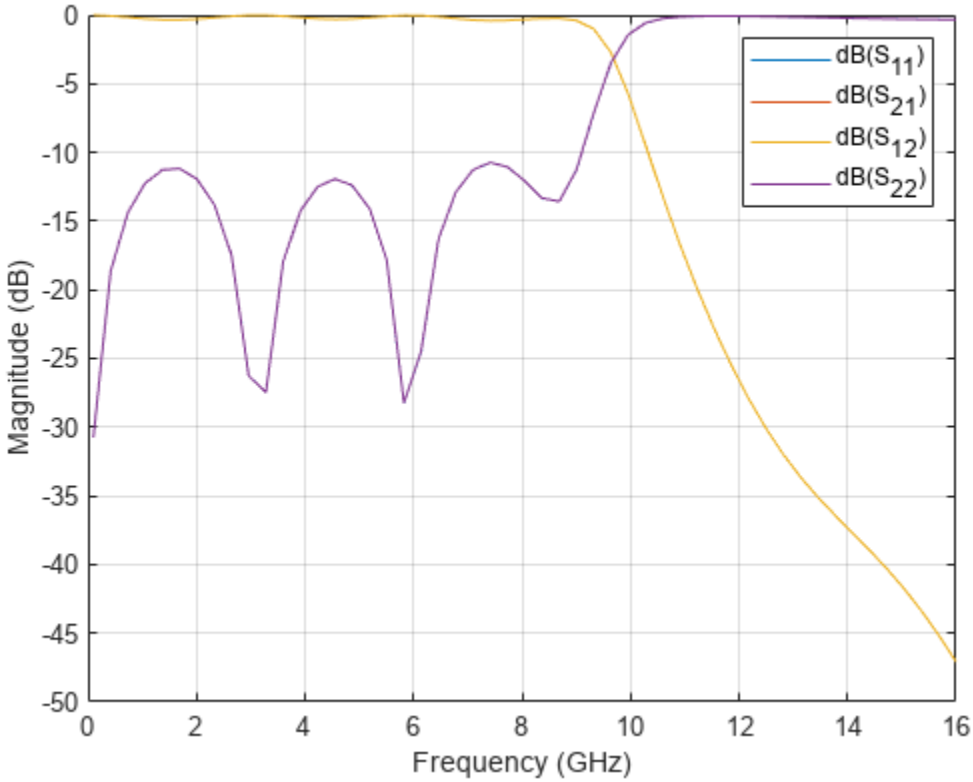
```
LPF                = pcbComponent;
d                  = dielectric('Teflon');
d.EpsilonR         = 2.2;
d.Thickness        = 0.508e-3;
GPL1               = PortLineL+L1+L2+L3+L4+L5+L4+L3+L2+L1+PortLineL;
GPW                = 20e-3;
gnd                = traceRectangular('Length',GPL1,'Width',GPW,'Center',[GPL1/2,0]);
LPF.BoardThickness = 0.508e-3;
LPF.Layers         = {filt1,d,gnd};
LPF.BoardShape     = gnd;
```

```
LPF.FeedDiameter   = PortLineW/2;
LPF.FeedLocations  = [0,0,1,3;GPL1,0,1,3];
figure,show(LPF);
```



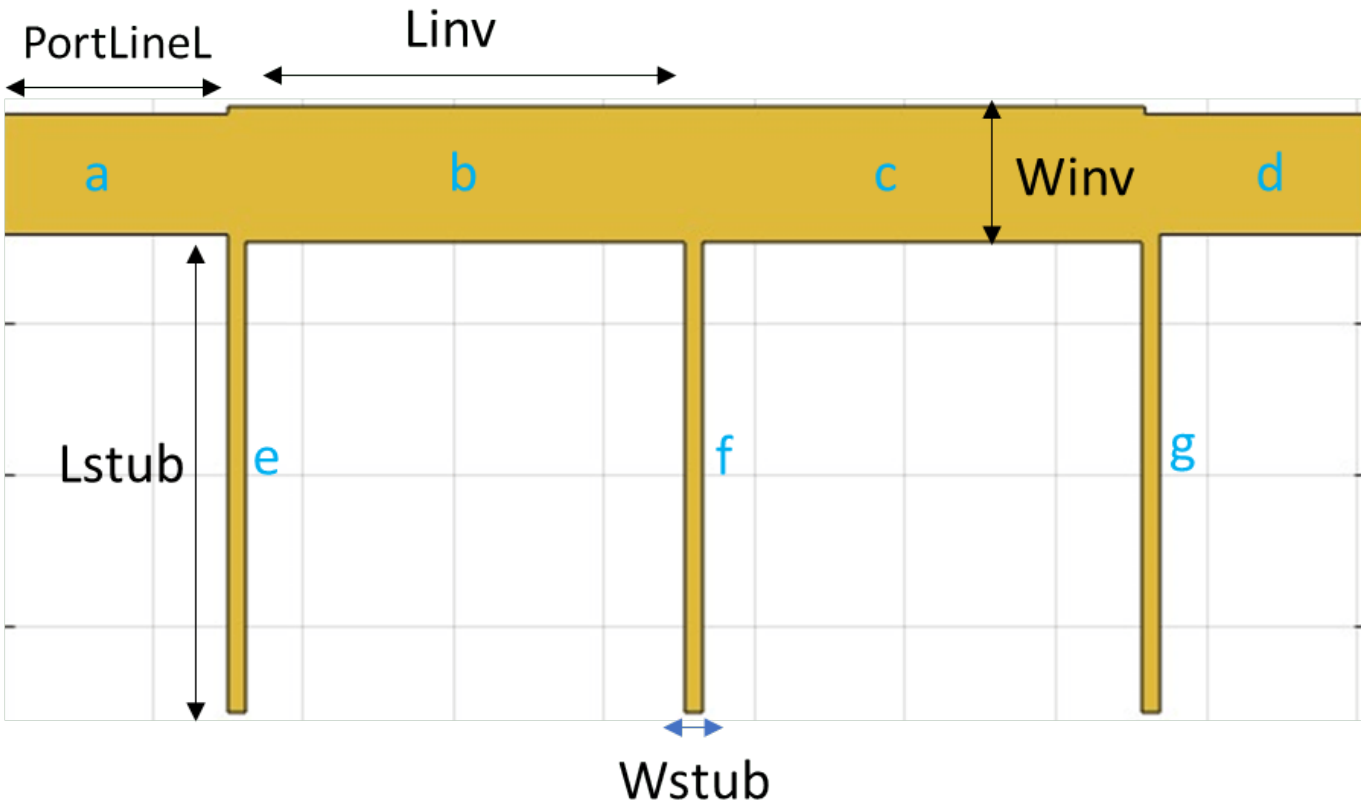
Use `sparameters` function to calculate the S-Parameters of the filter and plot them using the `rfplot` function.

```
spar1 = sparameters(LPF,linspace(0.1e9,16e9,51));
figure;
rfplot(spar1);
```

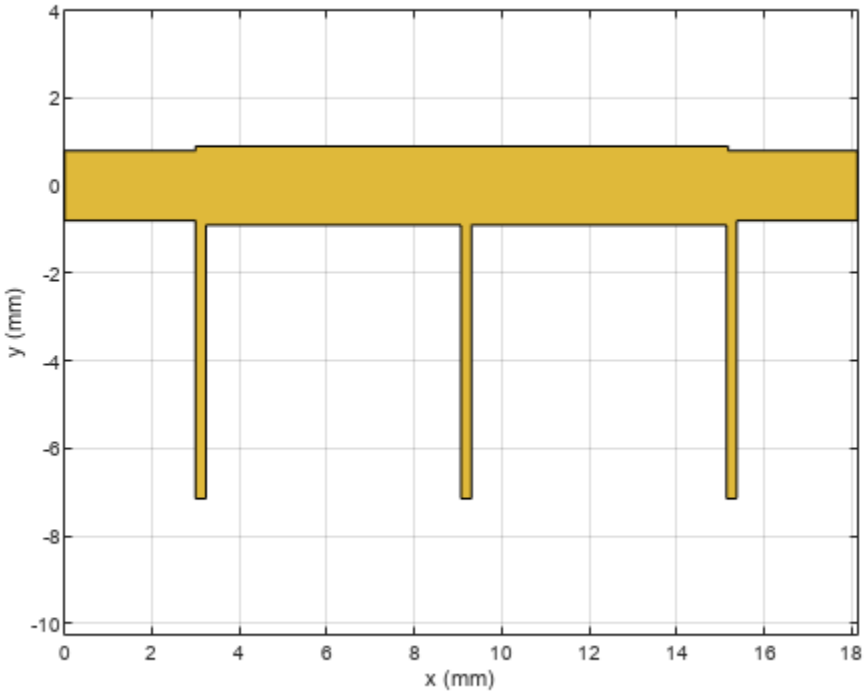


High Pass Filter

Create the variables and assign the values given in the paper[1]. Use the `traceRectangular` shape to create all the rectangles as shown in the figure and join them to create the filter structure. Visualize the structure using the `show` function.

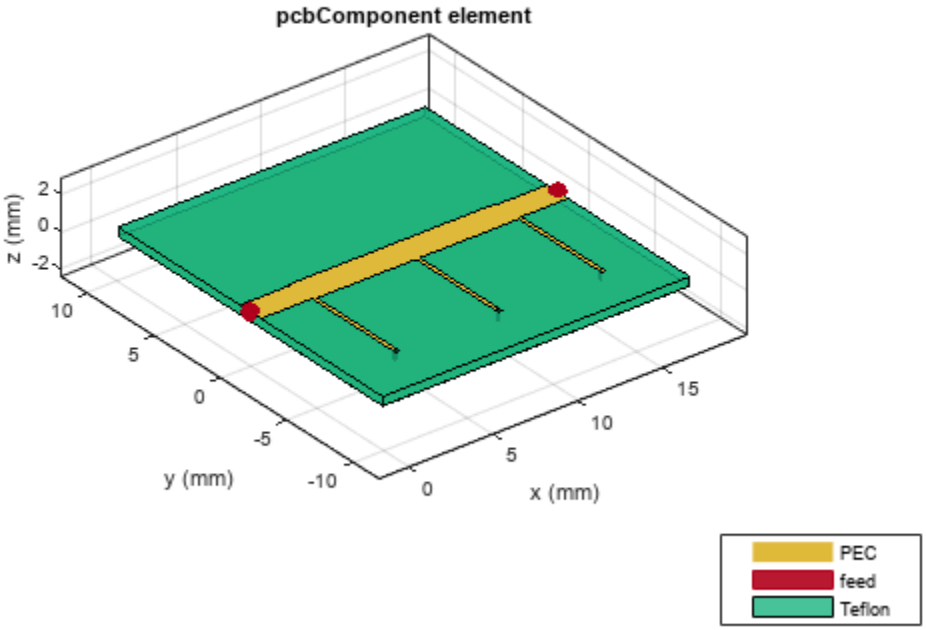



```
Lstub = 6.35e-3;
Wstub = 0.238e-3

Wstub = 2.3800e-04

Linv  = 6.07e-3;
Winv  = 1.8e-3;
a = traceRectangular('Length',PortLineL,'Width',PortLineW,'Center',[PortLineL/2,0]);
b = traceRectangular('Length',Linv,'Width',Winv,'Center',[PortLineL+Linv/2,0]);
c = traceRectangular('Length',Linv+0.04e-3,'Width',Winv,'Center',[PortLineL+Linv+Linv/2+0.04e-3/2
d = traceRectangular('Length',PortLineL+0.1e-3,'Width',PortLineW,'Center',[PortLineL+Linv+Linv+Po
e = traceRectangular('Length',Wstub,'Width',Lstub,'Center',[PortLineL+Wstub/2,-Lstub/2-PortLineW/
f = traceRectangular('Length',Wstub,'Width',Lstub,'Center',[PortLineL+Linv+Wstub/2,-Lstub/2-PortL
g = traceRectangular('Length',Wstub,'Width',Lstub,'Center',[PortLineL+Linv+Linv+Wstub/2,-Lstub/2-
filt2 = a+b+c+d+e+f+g;
figure;
show(filt2);
```



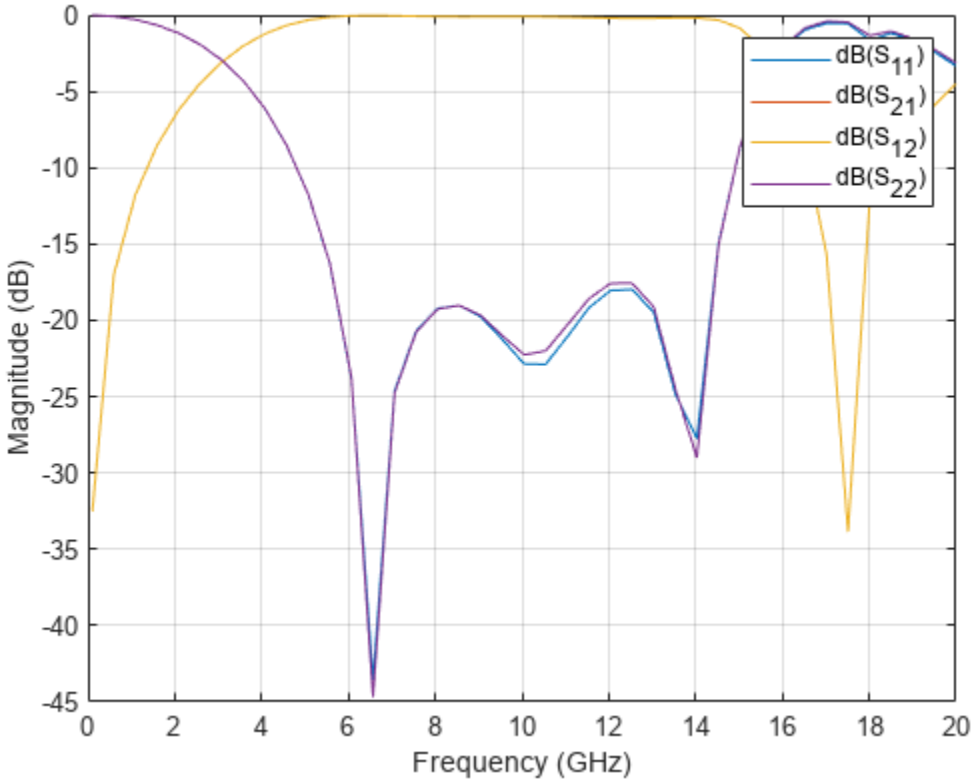
Create the PCB Stack of the filter using the `pcbComponent`. Assign the filter shape, dielectric, and groundplane to the `Layers` property. Set the `BoardThickness` to 0.508 mm and assign the `BoardShape` to the ground plane. Set the `FeedDiameter` and `FeedLocations`. Visualize the PCB.

```
HPF                = pcbComponent;
d                  = dielectric('Teflon');
d.EpsilonR         = 2.2;
d.Thickness        = 0.508e-3;
GPL2               = PortLineL+Linv+Linv+PortLineL;
GPW                = 20e-3;
gnd                = traceRectangular('Length',GPL2,'Width',GPW,'Center',[GPL2/2,0]);
HPF.BoardThickness = 0.508e-3;
HPF.Layers         = {filt2,d,gnd};
HPF.BoardShape     = gnd;
HPF.FeedDiameter   = PortLineW/2;
HPF.FeedLocations  = [0,0,1,3;GPL2,0,1,3];
HPF.ViaLocations   = [PortLineL+Wstub/2,-PortLineW/2-Lstub+0.2e-3,1,3;PortLineL+Linv+Wstub/2,-Po
HPF.ViaDiameter    = Wstub/2;
figure;
show(HPF);
```



Use `sparameters` function to calculate the S-Parameters of the filter and plot them using the `rfplot` function.

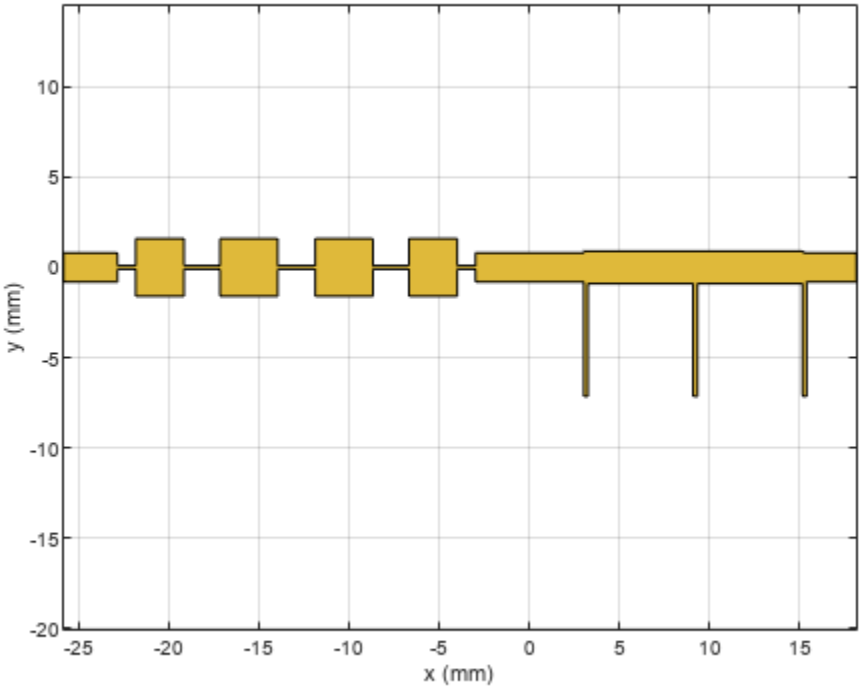
```
spar2 = sparameters(HPF,linspace(0.1e9,20e9,41));
figure;
rfplot(spar2);
```



Cascaded Filter Response

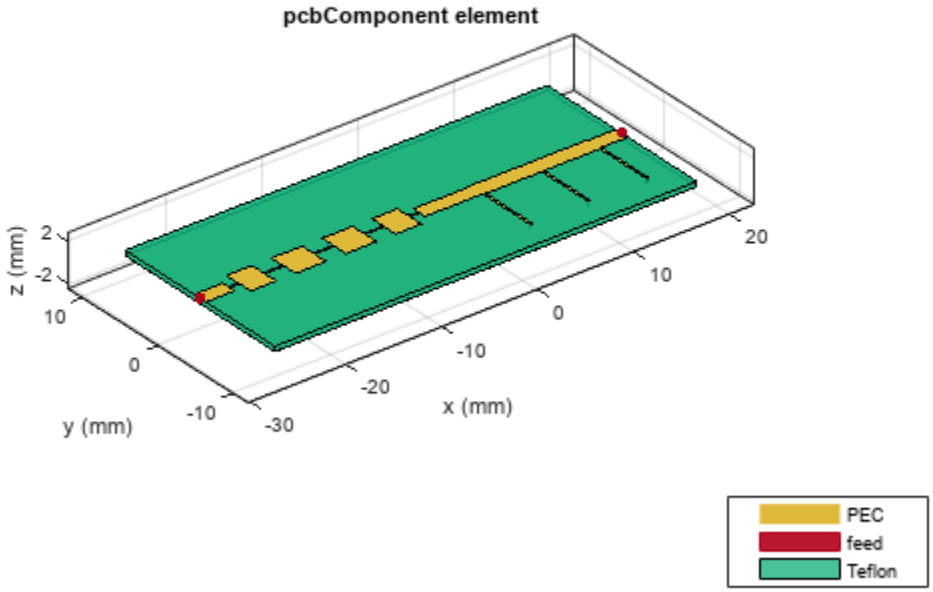
Cascade the LPF and HPF created above and create a ground plane so that it supports the cascaded shape.

```
filt12   = translate(copy(filt1),[-GPL1,0,0]);
filtcasc = filt12+filt2;
figure,show(filtcasc);
```



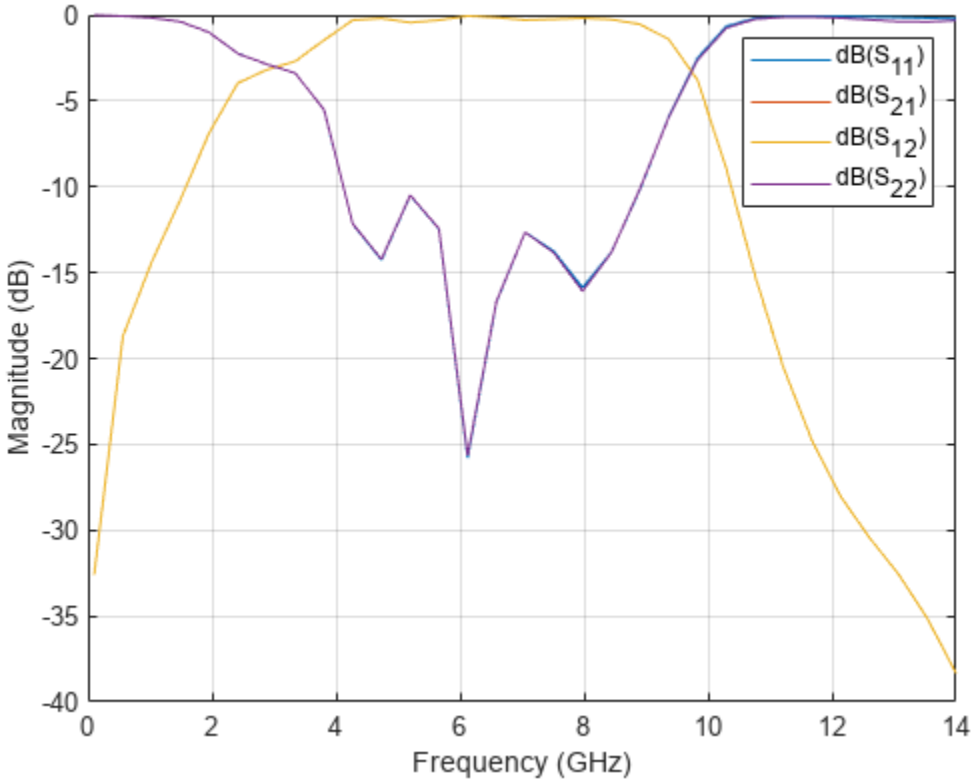
Create the PCB Stack of the filter using the `pcbComponent`. Create the groundplane using the `traceRectangular` shape. Assign the filter shape, dielectric, and groundplane to the `Layers` property. Set the `BoardThickness` to 0.508 mm and assign the `BoardShape` to the ground plane. Set the `FeedDiameter` and `FeedLocations`. Visualize the PCB.

```
casc                = pcbComponent;
d                   = dielectric('Teflon');
d.EpsilonR          = 2.2;
d.Thickness         = 0.508e-3;
GPL  = GPL1+GPL2;
GPW  = 20e-3;
gnd  = traceRectangular('Length',GPL,'Width',GPW,'Center',[(-GPL1+GPL2)/2,0]);
GPL3                = PortLineL+Linv+Linv+PortLineL;
casc.BoardThickness = 0.508e-3;
casc.Layers         = {filtcasc,d,gnd};
casc.BoardShape     = gnd;
casc.FeedDiameter   = PortLineW/2;
casc.FeedLocations  = [-GPL1,0,1,3;GPL3,0,1,3];
casc.ViaLocations   = [PortLineL+Wstub/2,-PortLineW/2-Lstub+0.2e-3,1,3;PortLineL+Linv+Wstub/2,-Po
casc.ViaDiameter    = Wstub/2;
figure;
show(casc);
```



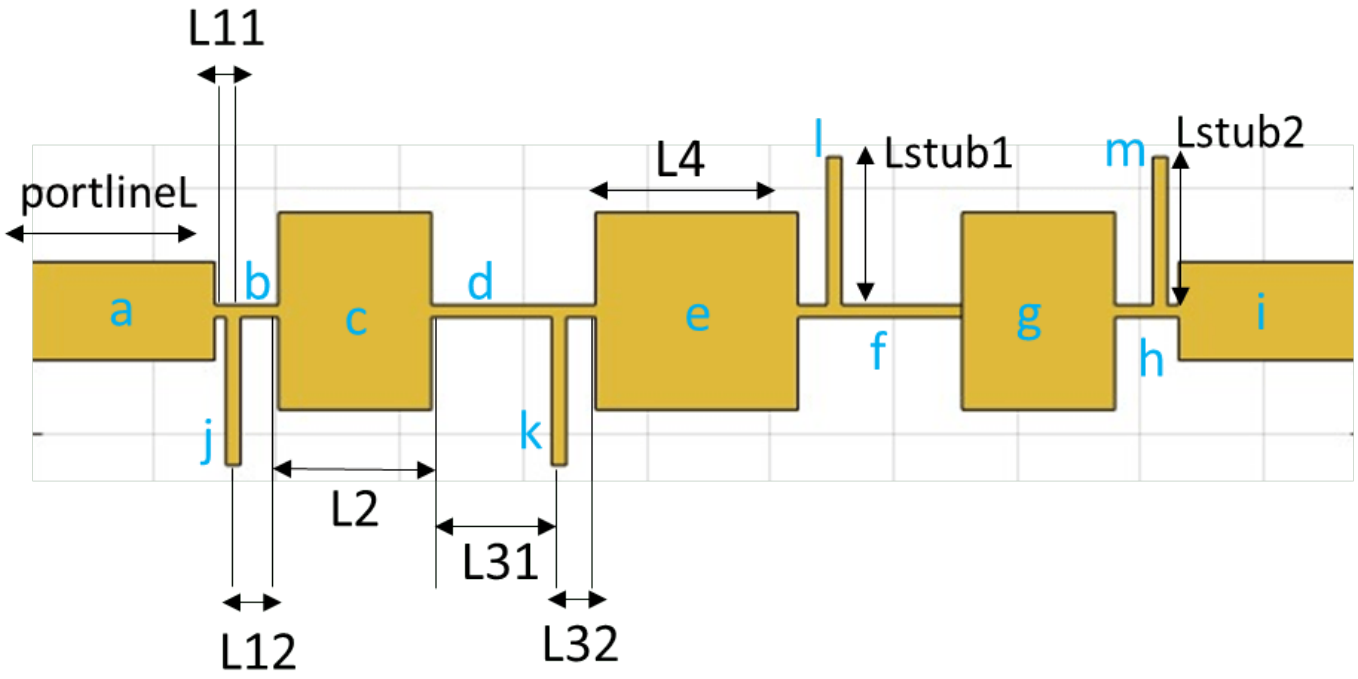
Use the `sparameters` function to calculate the S-Parameters and plot the parameters using the `rfplot` function

```
spar3 = sparameters(casc,linspace(0.1e9,14e9,31));
figure;
rfplot(spar3);
```

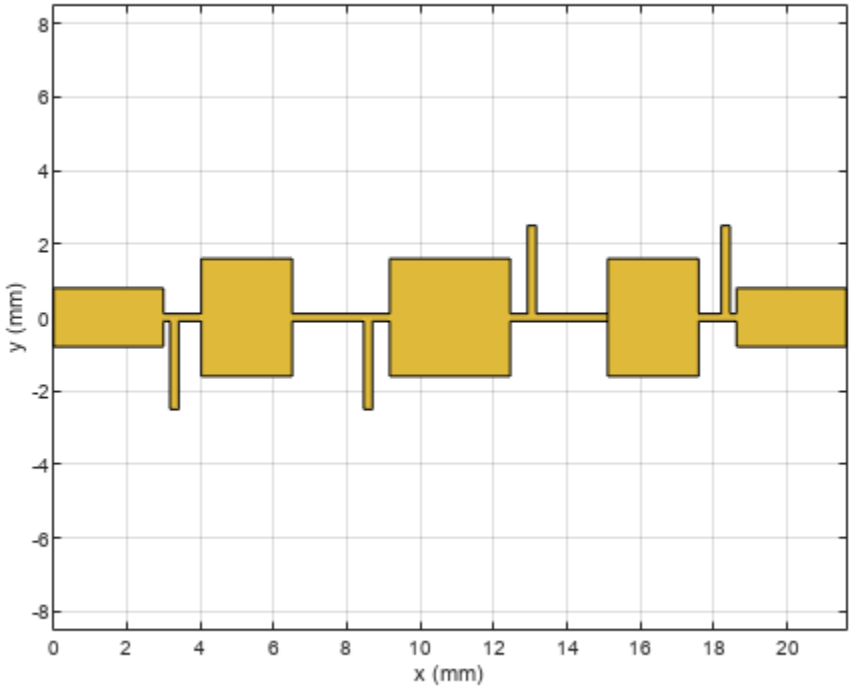


Composite Filter Type 1

Create the variables and assign the values given in the paper[1]. Use the `traceRectangular` shape to create all the rectangles as shown in the figure and join them to create the filter structure. Visualize the structure using the `show` function.

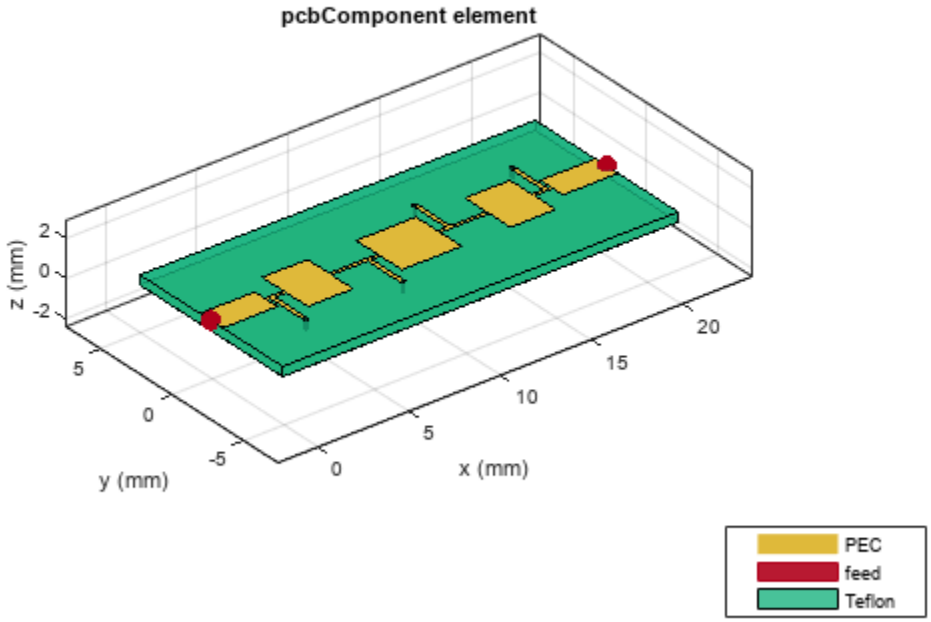



```
L11 = 0.3e-3;
L12 = 0.73e-3;
L2  = 2.48e-3;
L31 = 2.07e-3;
L32 = 0.59e-3;
L4  = 3.28e-3;
Lstub1 = 2.4e-3;
Lstub2 = 2.4e-3;
a = traceRectangular('Length',PortLineL,'Width',PortLineW,'Center',[PortLineL/2,0]);
b = traceRectangular('Length',L11+L12,'Width',Whi,'Center',[PortLineL+(L11+L12)/2,0]);
c = traceRectangular('Length',L2,'Width',Wlo,'Center',[PortLineL+L11+L12+L2/2,0]);
d = traceRectangular('Length',L31+L32,'Width',Whi,'Center',[PortLineL+L11+L12+L2+(L31+L32)/2,0])
e = traceRectangular('Length',L4,'Width',Wlo,'Center',[PortLineL+L11+L12+L2+(L31+L32)+L4/2,0]);
f = traceRectangular('Length',L31+L32,'Width',Whi,'Center',[PortLineL+L11+L12+L2+(L31+L32)+L4+(L
g = traceRectangular('Length',L2,'Width',Wlo,'Center',[PortLineL+L11+L12+L2+(L31+L32)+L4+(L31+L32
h = traceRectangular('Length',L11+L12,'Width',Whi,'Center',[PortLineL+L11+L12+L2+(L31+L32)+L4+(L
i = traceRectangular('Length',PortLineL,'Width',PortLineW,'Center',[PortLineL+L11+L12+L2+(L31+L32
j = traceRectangular('Length',Wstub,'Width',Lstub1,'Center',[PortLineL+L11,-Lstub1/2-Whi/2]);
k = traceRectangular('Length',Wstub,'Width',Lstub1,'Center',[PortLineL+L11+L12+L2+L31,-Lstub1/2-W
l = traceRectangular('Length',Wstub,'Width',Lstub1,'Center',[PortLineL+L11+L12+L2+L31+L32+L4+L32
m = traceRectangular('Length',Wstub,'Width',Lstub1,'Center',[PortLineL+L11+L12+L2+L31+L32+L4+L32-
compfiltShape1 = a+b+c+d+e+f+g+h+i+j+k+l+m;
figure;
show(compfiltShape1);
```



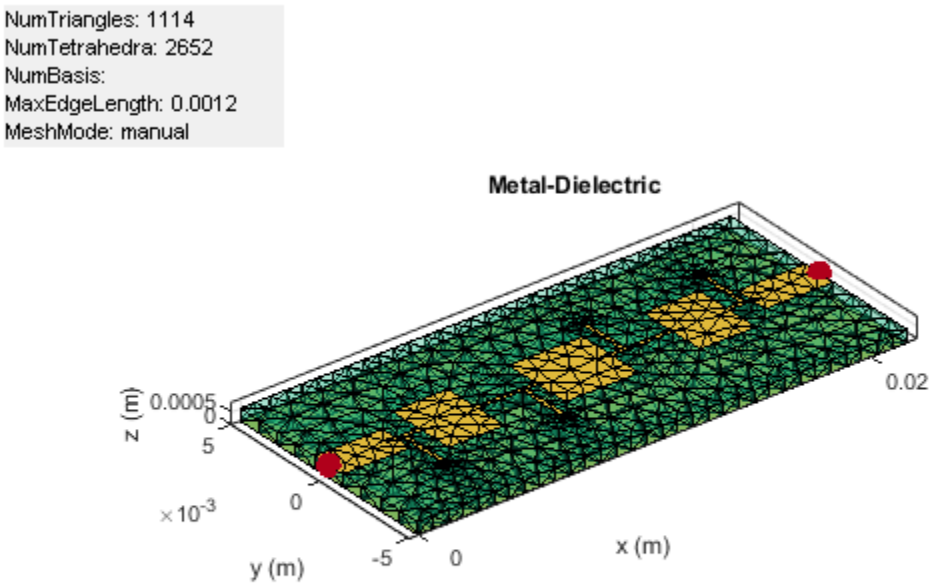
Create the PCB Stack of the filter using the `pcbComponent`. Assign the filter shape, dielectric, and groundplane to the `Layers` property. Set the `BoardThickness` to 0.508 mm and assign the `BoardShape` to the ground plane. Set the `FeedDiameter` and `FeedLocations`. Visualize the PCB.

```
compfilt1 = pcbComponent;
d    = dielectric('Teflon');
d.EpsilonR  = 2.2;
d.Thickness = 0.508e-3;
GPL2 = PortLineL+2*(L11+L12)+2*(L31+L32)+2*L2+L4+PortLineL;
GPW  = 20e-3;
gnd  = traceRectangular('Length',GPL2,'Width',GPW/2,'Center',[(GPL2)/2,0]);
compfilt1.BoardThickness = 0.508e-3;
compfilt1.Layers         = {compfiltShape1,d,gnd};
compfilt1.BoardShape     = gnd;
compfilt1.FeedDiameter   = PortLineW/2;
compfilt1.FeedLocations  = [0,0,1,3;GPL2,0,1,3];
compfilt1.ViaLocations   = [PortLineL+L11,-Whi/2-Lstub1+0.1e-3,1,3;PortLineL+L11+L12+L2+L31,-Whi/
compfilt1.ViaDiameter    = Wstub/2;
figure;
show(compfilt1);
```



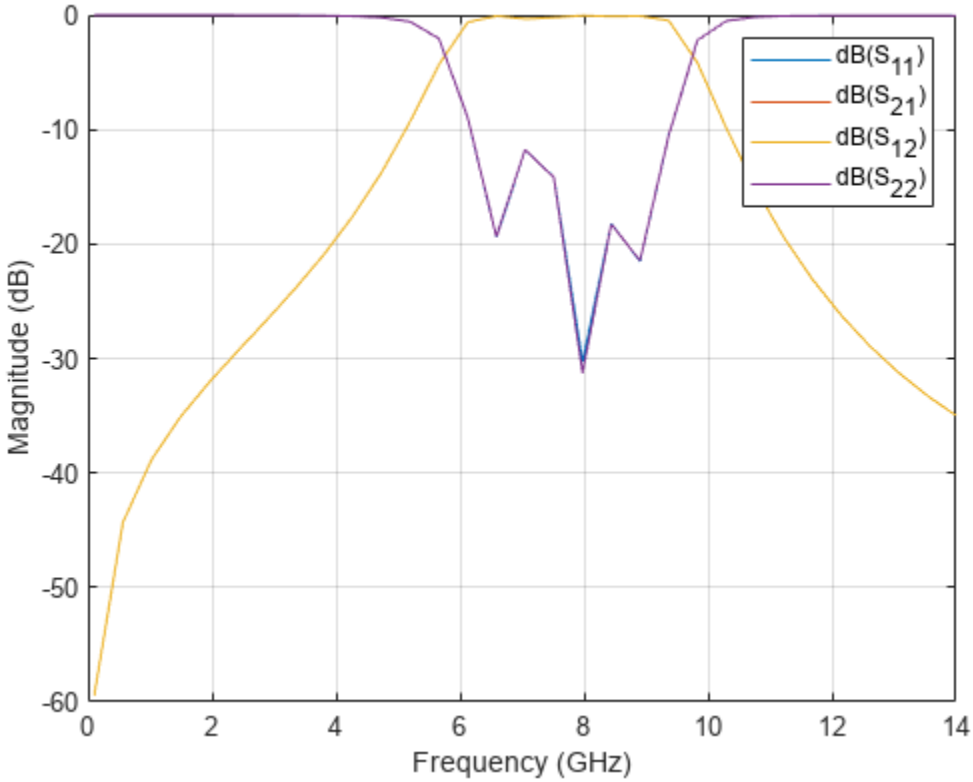
Use the `mesh` function to generate a manual mesh with a `MaxEdgeLength` as 1 mm.

```
figure;
mesh(compfilt1,'MaxEdgeLength',1.2e-3);
```



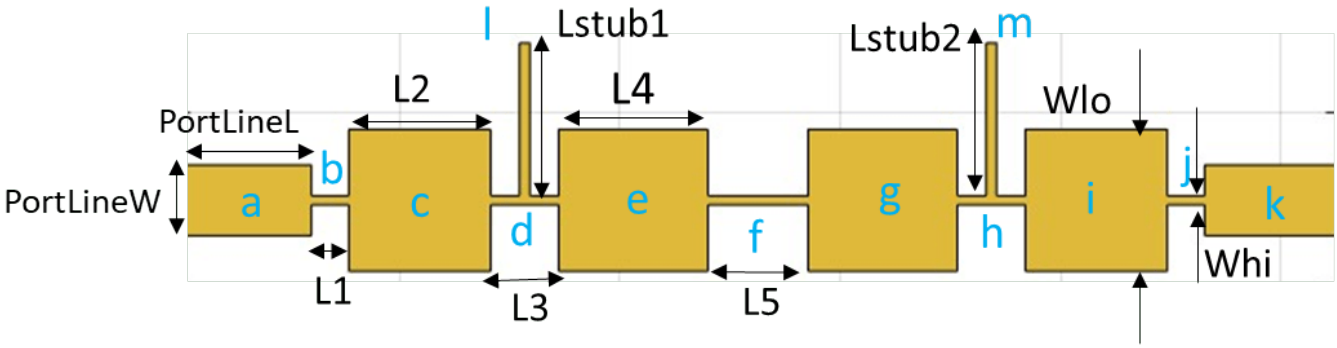
Use `sparameters` function to calculate the S-Parameters of the filter and plot them using the `rfplot` function.

```
spar4 = sparameters(compfilt1,linspace(0.1e9,14e9,31));
figure;
rfplot(spar4);
```



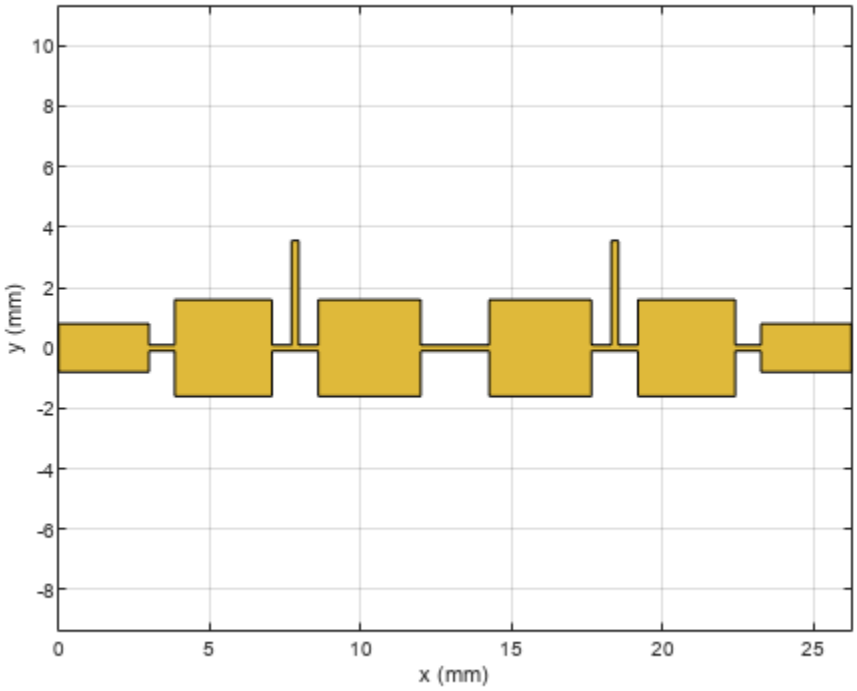
Composite Filter Type 2

Create the variables and assign the values given in the paper[1]. Use the `traceRectangular` shape to create all the rectangles as shown in the figure and join them to create the filter structure. Visualize the structure using the `show` function.




```
Whi = 0.2e-3;
Wlo = 3.2e-3;
L1  = 0.85e-3;
L2  = 3.22e-3;
L3  = 1.54e-3;
L4  = 3.39e-3;
L5  = 2.27e-3;
```

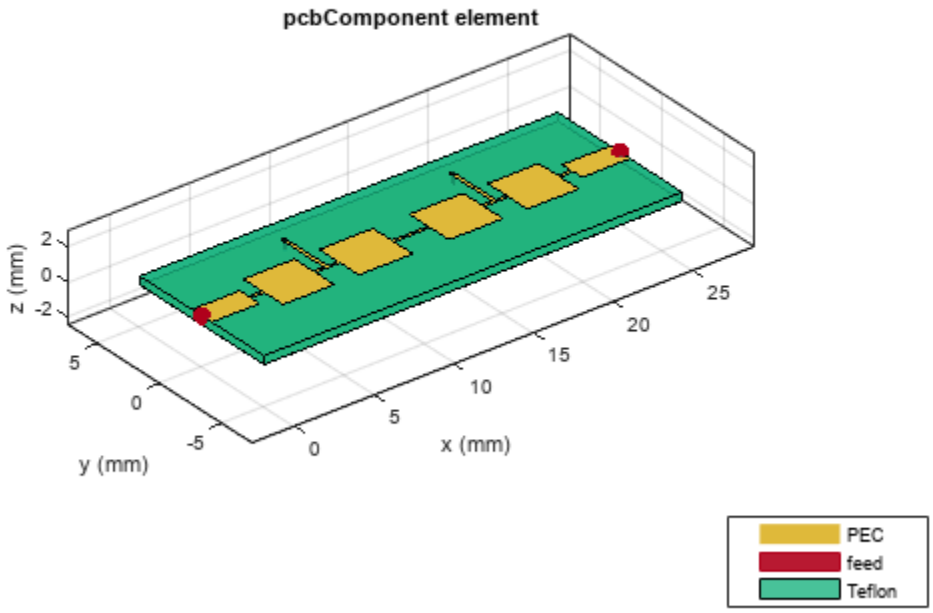
```
Lstub1 = 3.45e-3;
a  = traceRectangular('Length',PortLineL,'Width',PortLineW,'Center',[PortLineL/2,0]);
b  = traceRectangular('Length',L1,'Width',Whi,'Center',[PortLineL+L1/2,0]);
c  = traceRectangular('Length',L2,'Width',Wlo,'Center',[PortLineL+L1+L2/2,0]);
l = traceRectangular('Length',Wstub,'Width',Lstub1,'Center',[PortLineL+L1+L2+L3/2,Lstub1/2+Whi/2
d  = traceRectangular('Length',L3,'Width',Whi,'Center',[PortLineL+L1+L2+L3/2,0]);
e  = traceRectangular('Length',L4,'Width',Wlo,'Center',[PortLineL+L1+L2+L3+L4/2,0]);
f  = traceRectangular('Length',L5,'Width',Whi,'Center',[PortLineL+L1+L2+L3+L4+L5/2,0]);
g  = traceRectangular('Length',L4,'Width',Wlo,'Center',[PortLineL+L1+L2+L3+L4+L5+L4/2,0]);
m = traceRectangular('Length',Wstub,'Width',Lstub1,'Center',[PortLineL+L1+L2+L3+L4+L5+L4+L3/2,Ls
h  = traceRectangular('Length',L3,'Width',Whi,'Center',[PortLineL+L1+L2+L3+L4+L5+L4+L3/2,0]);
i  = traceRectangular('Length',L2,'Width',Wlo,'Center',[PortLineL+L1+L2+L3+L4+L5+L4+L3+L2/2,0]);
j  = traceRectangular('Length',L1,'Width',Whi,'Center',[PortLineL+L1+L2+L3+L4+L5+L4+L3+L2+L1/2,0
k  = traceRectangular('Length',PortLineL,'Width',PortLineW,'Center',[PortLineL+L1+L2+L3+L4+L5+L4-
compfiltShape2 = a+b+c+d+e+f+g+h+i+j+k+l+m;
figure;
show(compfiltShape2);
```



Create the PCB Stack of the filter using the `pcbComponent`. Assign the filter shape, dielectric, and groundplane to the `Layers` property. Set the `BoardThickness` to 0.508 mm and assign the `BoardShape` to the ground plane. Set the `FeedDiameter` and `FeedLocations`. Visualize the PCB.

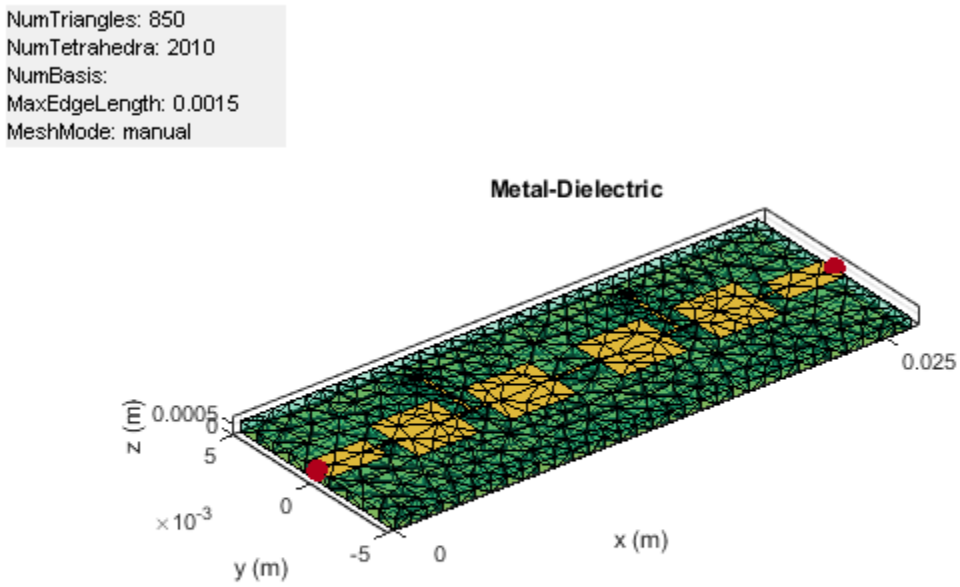
```
compfilt2 = pcbComponent;
d = dielectric('Teflon');
d.EpsilonR  = 2.2;
d.Thickness = 0.508e-3;
GPL1 = PortLineL+L1+L2+L3+L4+L5+L4+L3+L2+L1+PortLineL;
```

```
GPW  = 20e-3;
gnd  = traceRectangular('Length',GPL1,'Width',GPW/2,'Center',[GPL1/2,0]);
compfilt2.BoardThickness = 0.508e-3;
compfilt2.Layers         = {compfiltShape2,d,gnd};
compfilt2.BoardShape     = gnd;
compfilt2.FeedDiameter   = PortLineW/2;
compfilt2.FeedLocations  = [0,0,1,3;GPL1,0,1,3];
compfilt2.ViaLocations   = [PortLineL+L1+L2+L3/2,Lstub1+Whi/2-0.2e-3,1,3;PortLineL+L1+L2+L3+L4+L5
compfilt2.ViaDiameter    = Wstub/2;
figure;
show(compfilt2);
```



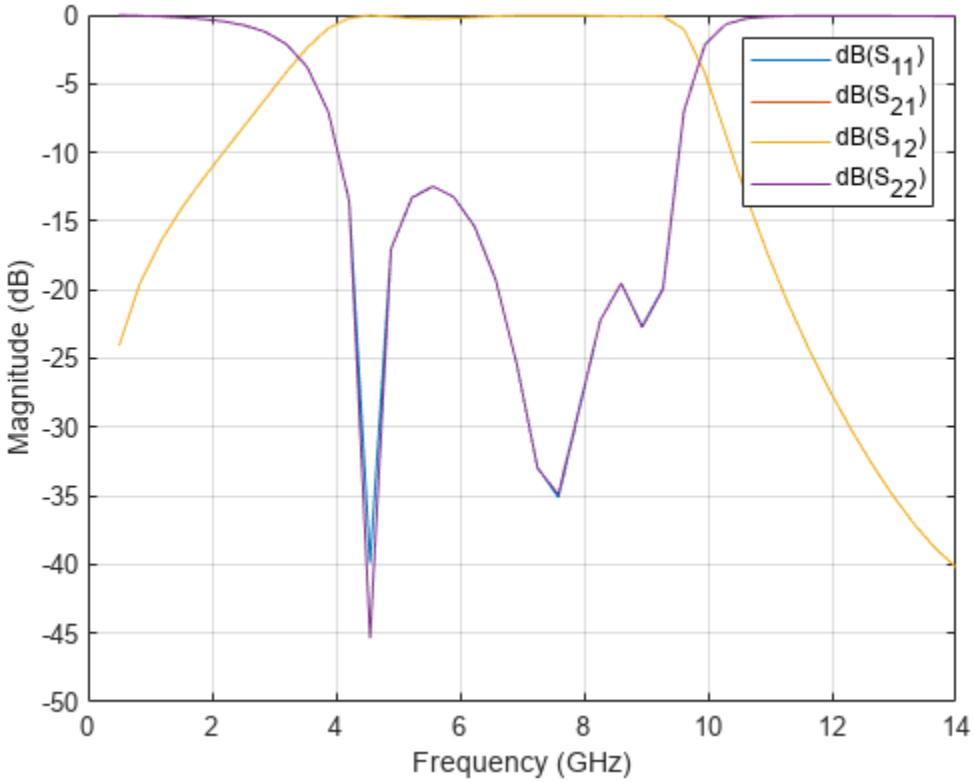
Use the `mesh` function to generate a manual mesh with a `MaxEdgeLength` as 1.5 mm.

```
figure;
mesh(compfilt2,'MaxEdgeLength',1.5e-3);
```



Use `sparameters` function to calculate the S-Parameters of the filter and plot them using the `rfplot` function.

```
spar5 = sparameters(compfilt2,linspace(0.5e9,14e9,41));
figure;
rfplot(spar5);
```



References

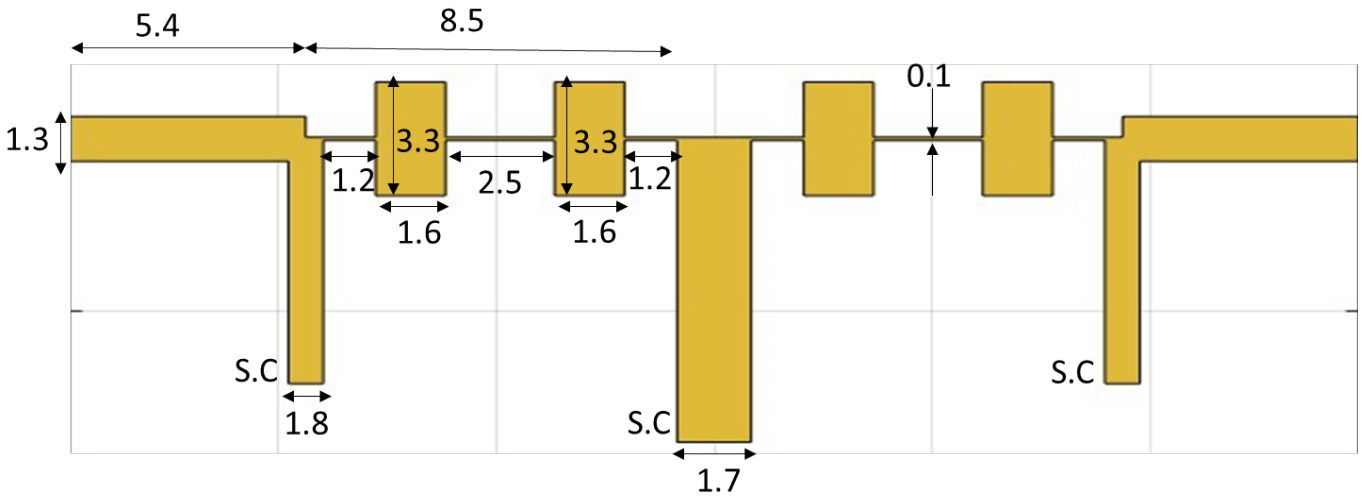
1) Ching-Luh Hsu, Fu-Chieh Hsu and Jen-Tsai Kuo, Microstrip Bandpass Filters for Ultra-Wideband (UWB) Wireless Communications.

UWB Bandpass Filter Using Open and Short-Circuited Stubs

This example shows how to create an open or short circuited stub filter using the `filterStub` catalog and analyze the filter using RF PCB Toolbox.

UWB 5-Pole Short Circuited Bandpass Filter

Schematic



Units are in mm

Create filter

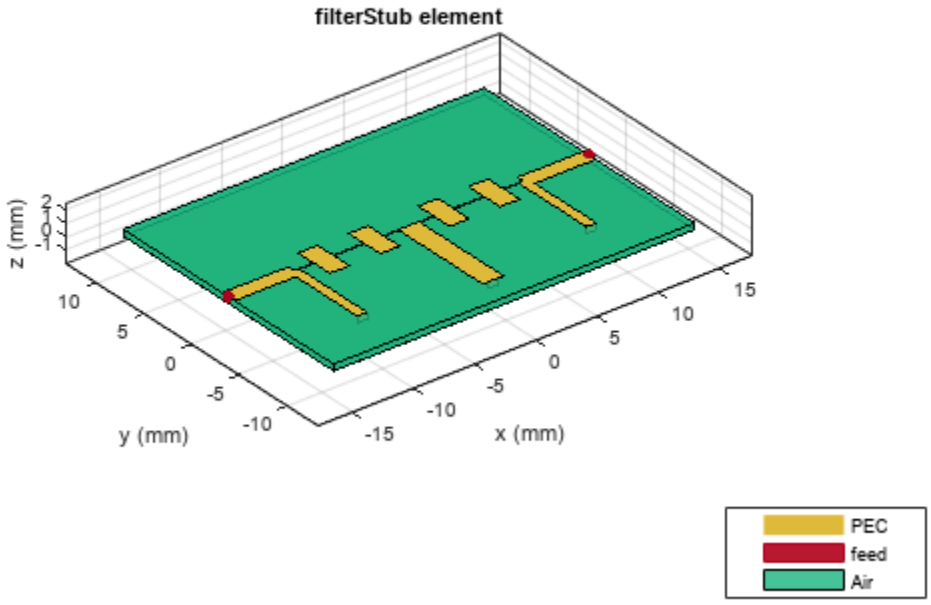
Use the `filterStub` object to create a stub filter. Define port lines and series line using the properties of `PortLineLength`, `PortLineWidth`, `SeriesLineLength`, and `SeriesLineWidth`. Define the stub length, stub width, and stub position using `StubLength`, `StubWidth`, and `StubOffsetX`. Set the `StubShort` and `StubDirection` according to the structure.

```
filter1 = filterStub;
filter1.PortLineLength = 5.4e-3;
filter1.PortLineWidth = 1.3e-3;
filter1.SeriesLineLength = 18.7e-3;
filter1.SeriesLineWidth = 0.1e-3;
filter1.StubLength = [7.1e-3 1.65e-3 1.65e-3 1.65e-3 1.65e-3 8.8e-3 1.65e-3 1.65e-3 1.65e-3 1.65e
filter1.StubWidth = [0.8e-3 1.6e-3 1.6e-3 1.6e-3 1.6e-3 1.7e-3 1.6e-3 1.6e-3 1.6e-3 1.6e-3 0.8e-3
filter1.StubOffsetX = [-9.35e-3 -6.95e-3 -6.95e-3 -2.85e-3 -2.85e-3 0e-3 2.85e-3 2.85e-3 6.95e-3
filter1.StubShort = [1 0 0 0 0 1 0 0 0 0 1];
filter1.StubDirection= [0 1 0 1 0 0 1 0 1 0 0];
filter1.GroundPlaneWidth= 0.0220;
filter1.Height = 0.508e-3;
substrate = dielectric("EpsilonR",3.05,"LossTangent",0.0001,"Thickness",0.508e-3);
filter1.Substrate = substrate;
```

Visualize stub filter

Use the `show` function to visualize the filter.

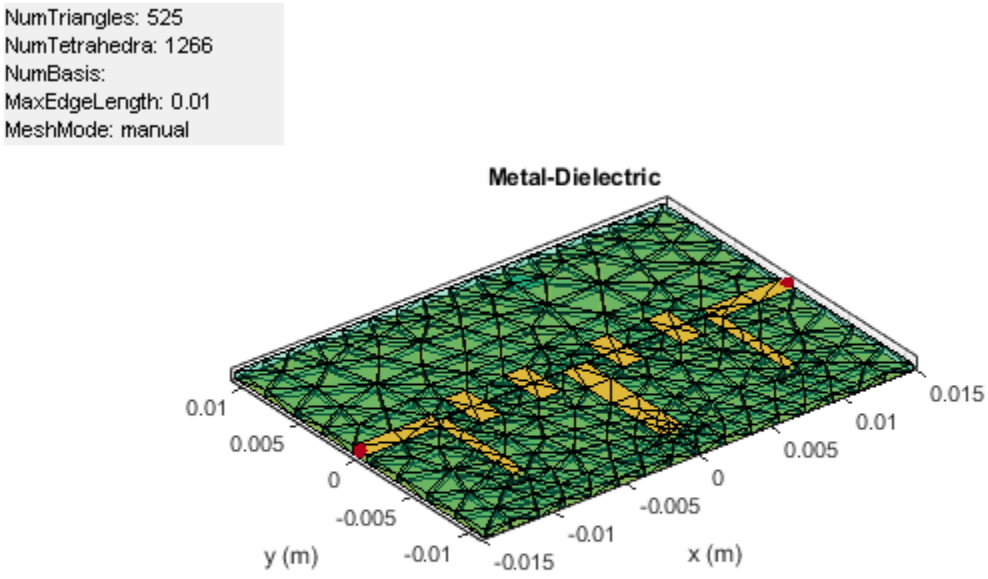
```
figure;show(filter1)
```



Analyze filter

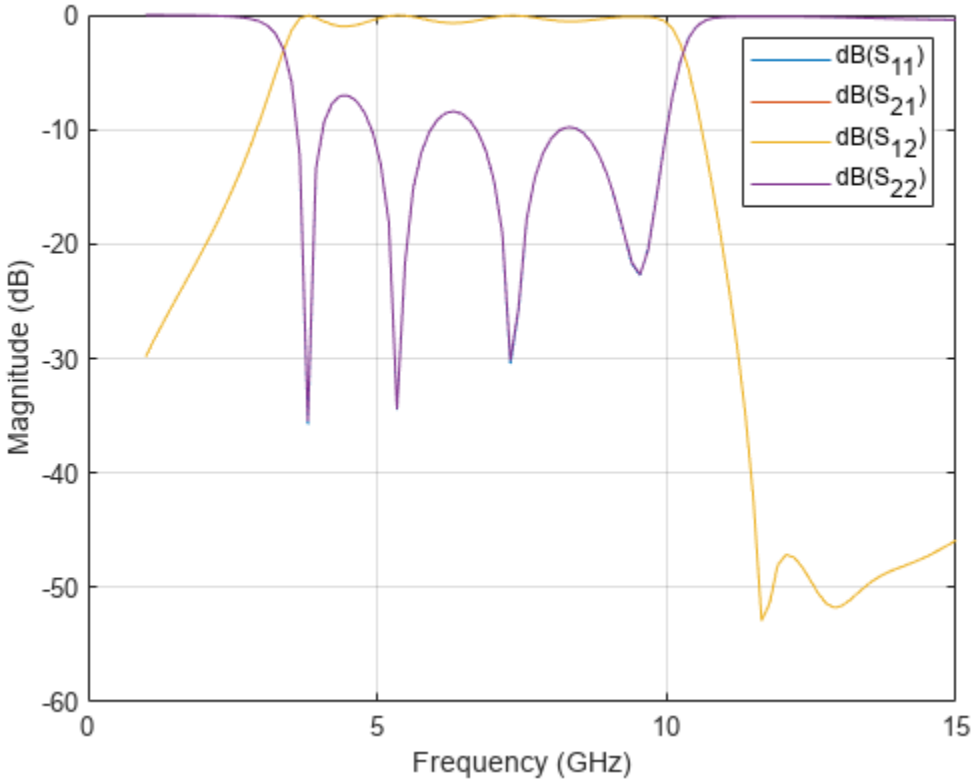
Use the `mesh` function and assign the `MaxEdgeLength` to 10 mm to ensure 25 triangles per wavelength.

```
figure;mesh(filter1,"MaxEdgeLength",0.010)
```



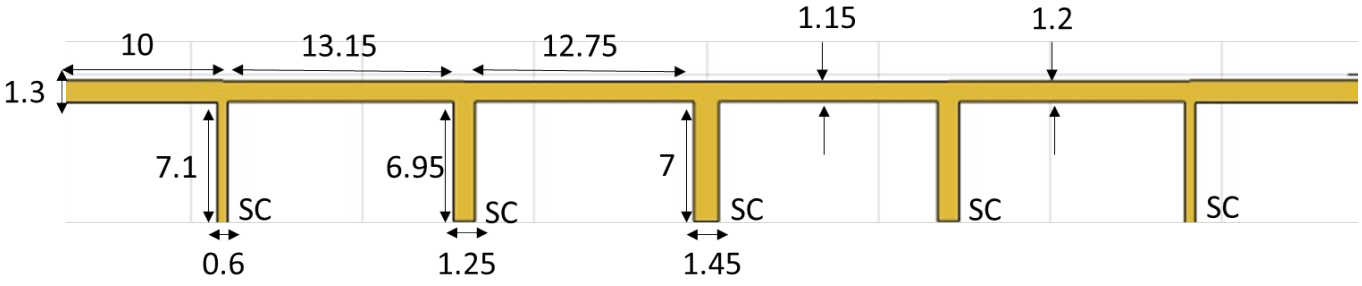
Use the sparameters function to calculate the S-Parameters and plot it using rfplot function.

```
s1=sparameters(filter1,linspace(1e9,15e9,101));
figure;rfplot(s1);
```



UWB 9-Pole Short Circuited Bandpass Filter

Schematic



Units are in mm

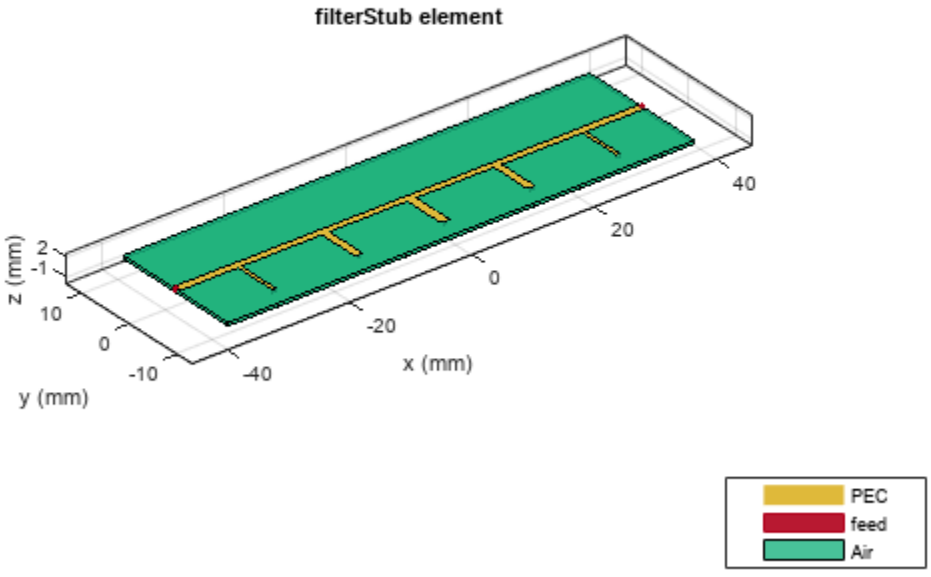
Create filter

Use the `filterStub` object to create a stub filter. Define port lines and series line using the properties of `PortLineLength`, `PortLineWidth`, `SeriesLineLength`, and `SeriesLineWidth`. Define the stub length, stub width, and stub position using `StubLength`, `StubWidth`, and `StubOffsetX`. Set the `StubShort` and `StubDirection` according to the structure.

```
filter2 = filterStub;
filter2.PortLineLength = 0.01;
filter2.PortLineWidth = 1.3e-3;
filter2.SeriesLineLength = [14.075e-3 14.1e-3 14.1e-3 14.075e-3];
filter2.SeriesLineWidth = [1.2e-3 1.15e-3 1.15e-3 1.2e-3];
filter2.StubLength = [7.75e-3 7.55e-3 7.575e-3 7.55e-3 7.75e-3];
filter2.StubWidth = [0.6e-3 1.25e-3 1.45e-3 1.25e-3 0.6e-3];
filter2.StubOffsetX = [-28.175e-3 -14.1e-3 0 14.1e-3 28.175e-3];
filter2.StubShort = ones(1,5);
filter2.StubDirection= zeros(1,5);
filter2.GroundPlaneWidth= 0.0220;
filter2.Height = 0.508e-3;
substrate = dielectric("EpsilonR",3.05,"LossTangent",0.0001,"Thickness",0.508e-3);
filter2.Substrate = substrate;
```

Visualize stub filter

```
figure;show(filter2)
```

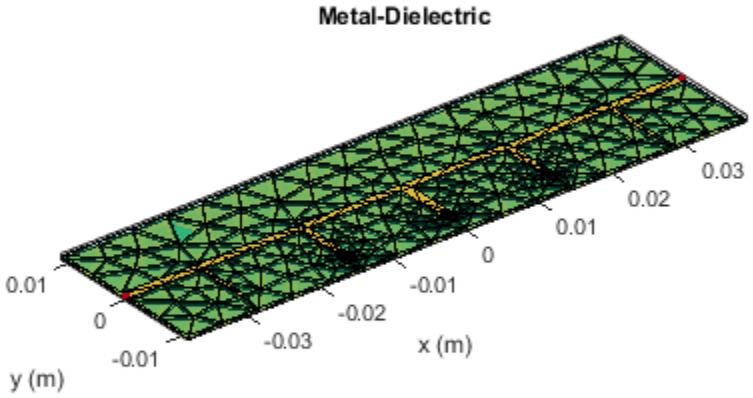


Analyze the filter

Use the `mesh` function and assign the `MaxEdgeLength` to 10 mm ensure 25 triangles per wavelength.

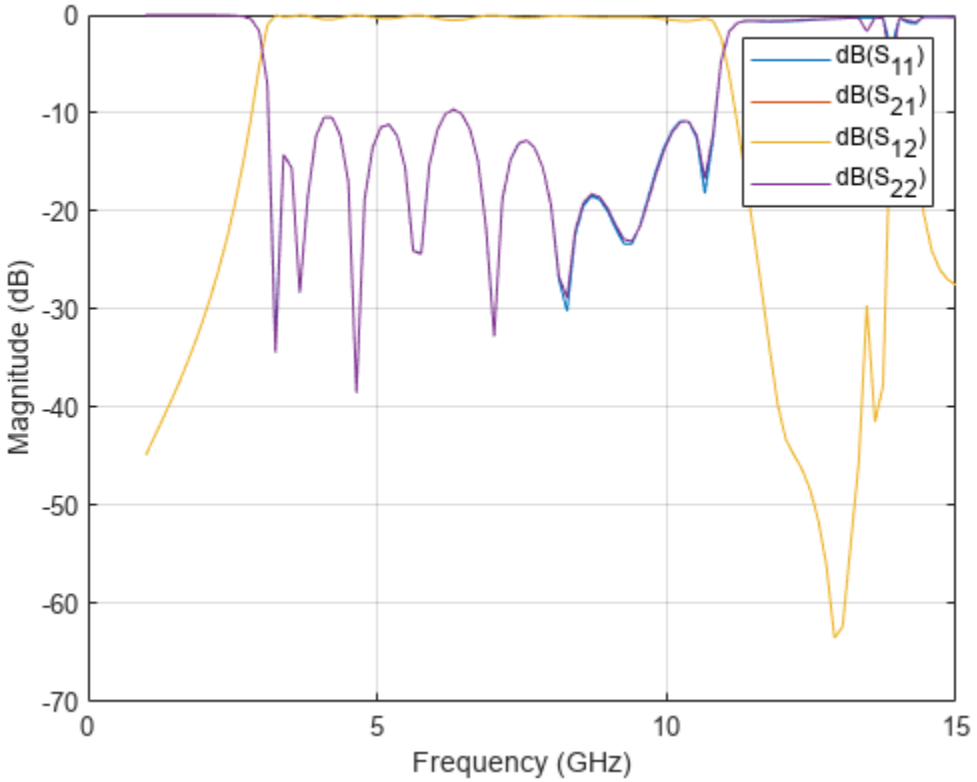
```
figure;mesh(filter2,"MaxEdgeLength",0.02)
```

```
NumTriangles: 591
NumTetrahedra: 1398
NumBasis:
MaxEdgeLength: 0.02
MeshMode: manual
```



Use the sparameters function to calculate the S-Parameters and plot it using rfplot function.

```
s2=sparameters(filter2,linspace(1e9,15e9,101));
figure;rfplot(s2);
```


0
-10
-20
-30
-40
-50
-60
-70
Magnitude (dB)
0
5
10
15
Frequency (GHz)
dB(S11)
dB(S21)
dB(S12)
dB(S22)

Comparison of Lumped and Distributed EM Models for Low Pass Filters

This example shows how to compare lumped and distributed electromagnetic (EM) model for low pass filters.

1. Use an `rffilter` object from the RF Toolbox to create a lumped element implementation for the low pass filter.
2. Use Richards and Kurodas identities to convert the lumped element filter into distributed element filter by calculating the line impedances.
3. Use the `microstripLine` design function to calculate the length and width of the transmission lines having different impedance
4. Use the `filterStub` catalog in RF PCB Toolbox is used to create and analyze this filter.

You will observe that the results from the RF Toolbox and RF PCB Toolbox are matching very well.

Define Frequencies

```
designFreq = 4e9;
freqRange = linspace(1e9,2*designFreq,51);
```

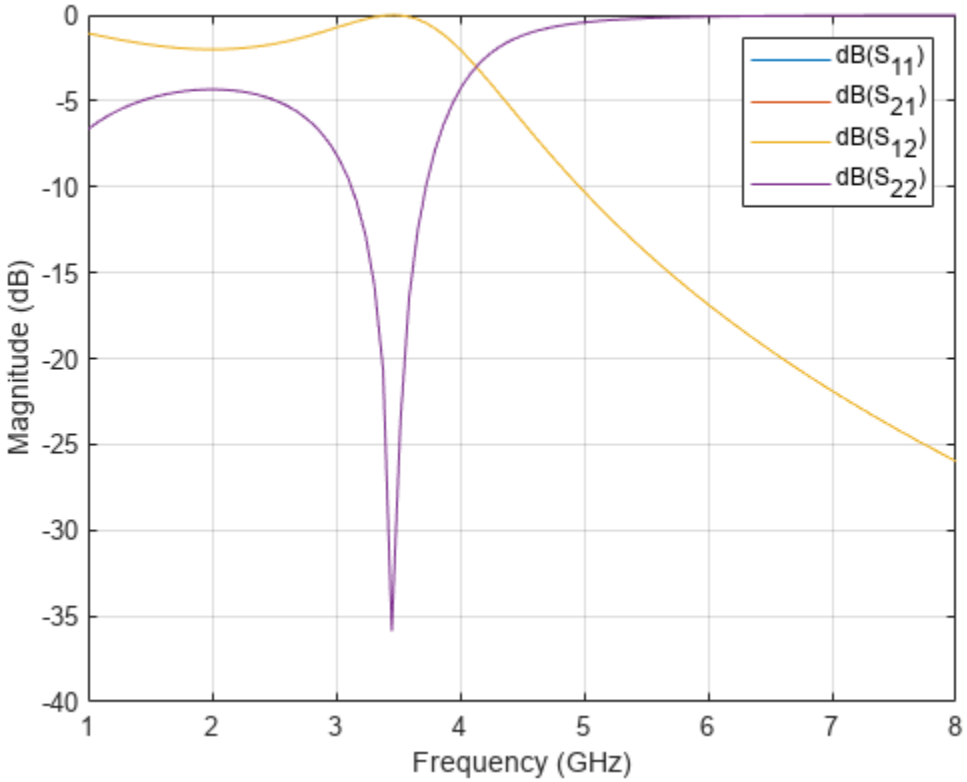
Create RF Filter

Use `rffilter` object to create an RF filter with a `filterOrder` 3 and with a 3 dB equal ripple passband response.

```
filt = rffilter('FilterOrder',3,'FilterType','Chebyshev');
filt.PassbandFrequency = designFreq;
filt.PassbandAttenuation = 2;
```

Use the sparameters function to compute the S-Parameters of the low pass filter and plot them using the rfplot function.

```
spar = sparameters(filt,linspace(1e9,8e9,101));
figure,rfplot(spar);
```



The `rffilter` computes the values of L and C for the required response. You can access these values from the `DesignData` property of the RF filter.

```
filt.DesignData
```

```
ans = struct with fields:
            FilterOrder: 3
              Inductors: [5.3927e-09 5.3927e-09]
             Capacitors: 6.6261e-13
               Topology: 'lclowpasstee'
      PassbandFrequency: 4.0000e+09
    PassbandAttenuation: 2
```

Transform the L and C values into normalized low pass filter element values. In order to construct the distributed form of this filter, you need to convert the lumped elements into distributed transmissions lines using Richards transformations and Kuroda identities. Using these g1, g2 and g3 values compute the line impedances for the series lines and the open stubs

```
L = filt.DesignData.Inductors;
C = filt.DesignData.Capacitors;

% Computation
% normalized low-pass element values
g1 = 2*pi*filt.PassbandFrequency*L(1)/filt.Zin;
g3 = g1;
g2 = 2*pi*filt.PassbandFrequency*C*filt.Zin;
```

```
% Richard's transformation and Kuroda's 2nd identity
n2 = 1+(1/g1);
Zshunt1 = n2*filt.Zin;
Zseries1 = n2*g1*filt.Zin;
Zshunt2 = (1/g2)*filt.Zin;
Zseries2 = n2*g3*filt.Zin; %#ok<*NASGU>
Zshunt3 = n2*filt.Zin;
```

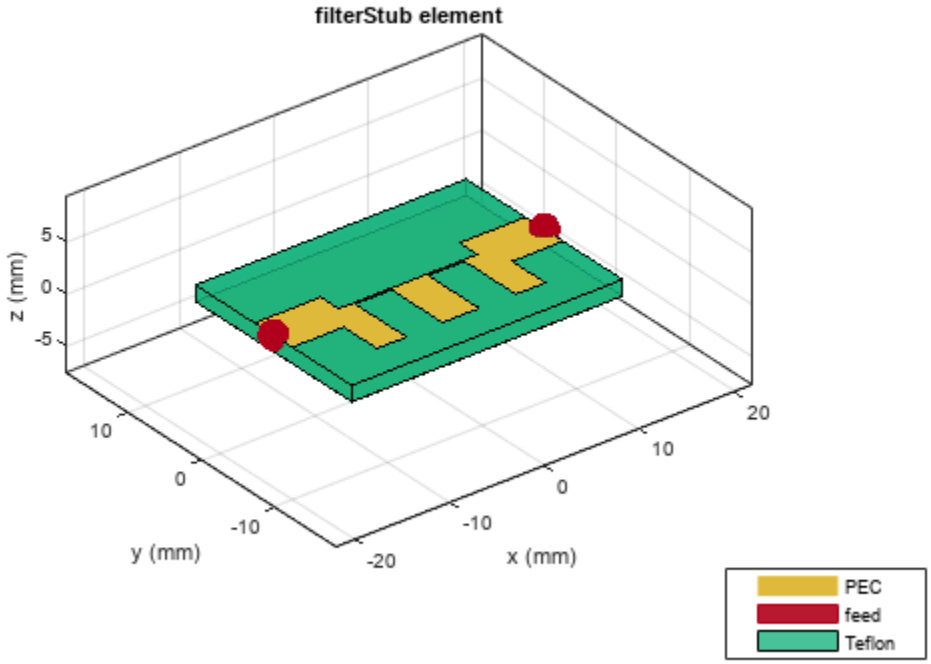
After calculating the impedances of transmission lines, use the design function from the `microstripLine` catalog to calculate the lengths and widths of the respective impedance.

```
obj = microstripLine;
portline = design(obj,designFreq,'Z0',50,'LineLength',1/8);
seriesLine1 = design(obj,designFreq,'Z0',Zseries1,'LineLength',1/8);
seriesLine2 = design(obj,designFreq,'Z0',Zseries2,'LineLength',1/8);
stub1 = design(obj,designFreq,'Z0',Zshunt1,'LineLength',1/8);
stub2 = design(obj,designFreq,'Z0',Zshunt2,'LineLength',1/8);
stub3 = design(obj,designFreq,'Z0',Zshunt3,'LineLength',1/8);
```

Comparison with EM Simulation

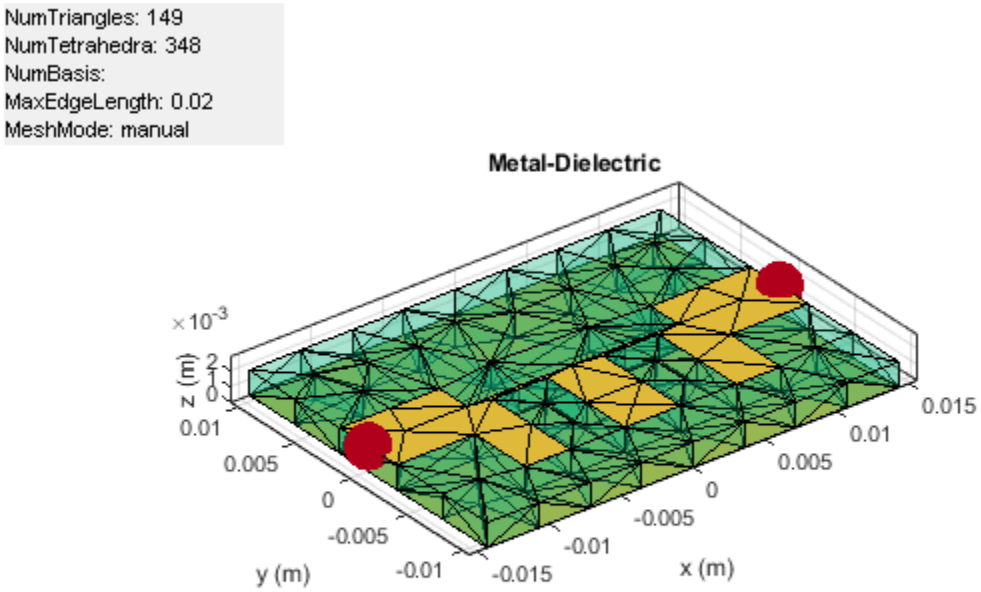
Use the `filterStub` object and create the filter using the dimensions from the RF filter.

```
filterEM = filterStub;
filterEM.PortLineLength = portline.Length;
filterEM.PortLineWidth = portline.Width;
filterEM.SeriesLineLength = [seriesLine1.Length seriesLine2.Length];
filterEM.SeriesLineWidth = [seriesLine1.Width seriesLine2.Width];
filterEM.StubLength = [stub1.Length stub2.Length stub3.Length];
filterEM.StubWidth = [stub1.Width stub2.Width stub3.Width];
filterEM.StubDirection = [0 0 0];
filterEM.StubShort = [0 0 0];
filterEM.StubOffsetX= [-seriesLine1.Length 0 seriesLine2.Length];
filterEM.GroundPlaneWidth = max(filterEM.StubLength)*3;
show(filterEM);
```



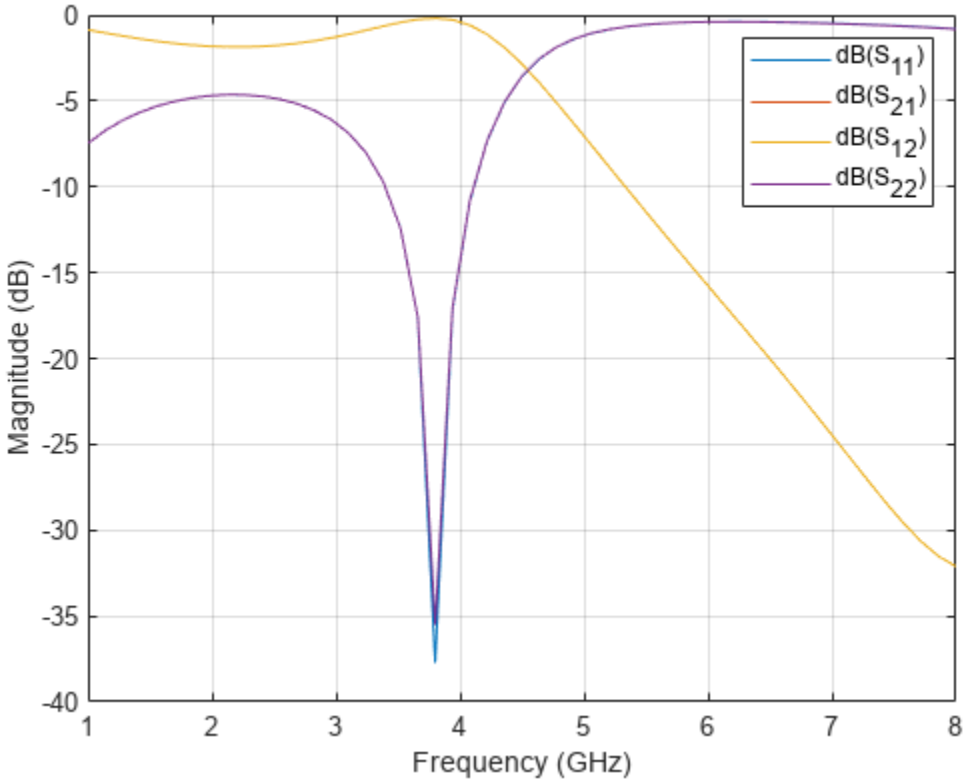
Use the `mesh` function and set the `MaxEdgeLength` to 5 mm to ensure 15 triangles per wavelength.

```
figure,mesh(filterEM,'MaxEdgeLength',0.02);
```



Use the `sparameters` function to calculate the S-parameters and plot it using the `rfplot` function.

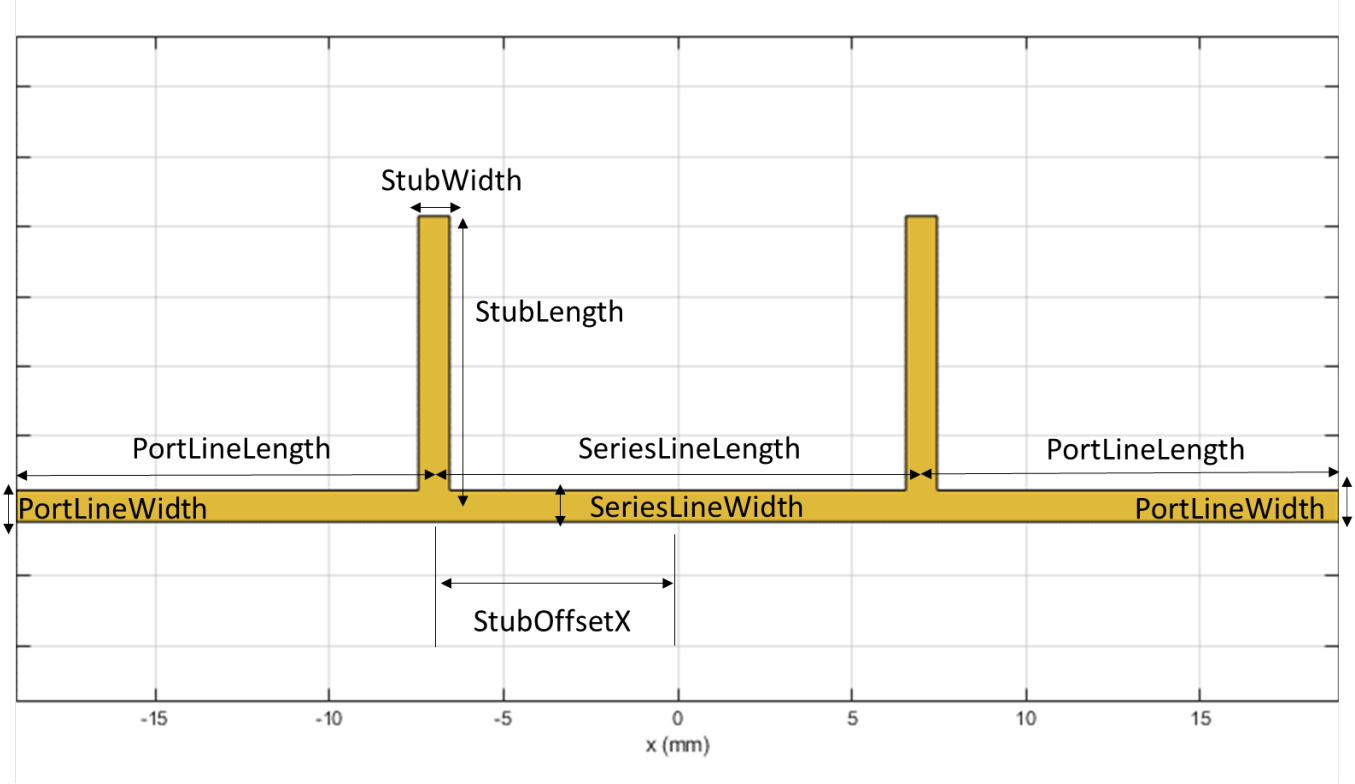
```
spar = sparameters(filterEM,linspace(1e9,8e9,51));
figure,rfplot(spar);
```


0
-5
-10
-15
-20
-25
-30
-35
-40
Magnitude (dB)
1
2
3
4
5
6
7
8
Frequency (GHz)
dB(S11)
dB(S21)
dB(S12)
dB(S22)

Variations of Open and Short Stub Filters

This example shows how to analyze the open and short stub filters using RF PCB Toolbox.

The stub filters find uses in many applications in RF and Microwave. You can place the stubs in multiple combinations to get the desired response from the filters. You can open-circuit or short-circuit the stubs and they can be oriented in any direction. The figure is a schematic of the filterStub with property definitions.



In this example, you use a `filterStub` object to generate the different variations of open and short stubs. The reference [1] lists out three types of open stub filters namely Type1, Type2 and Type3. These filters are listed below.

Filter with Single Open Stub - Type 1

Use `filterStub` object to create the stub filter and visualize its properties. The properties `PortLineLength` and `PortLineWidth` define the length and width of the port lines. The `SeriesLineWidth` and `SeriesLineLength` define the length and width of the series lines.

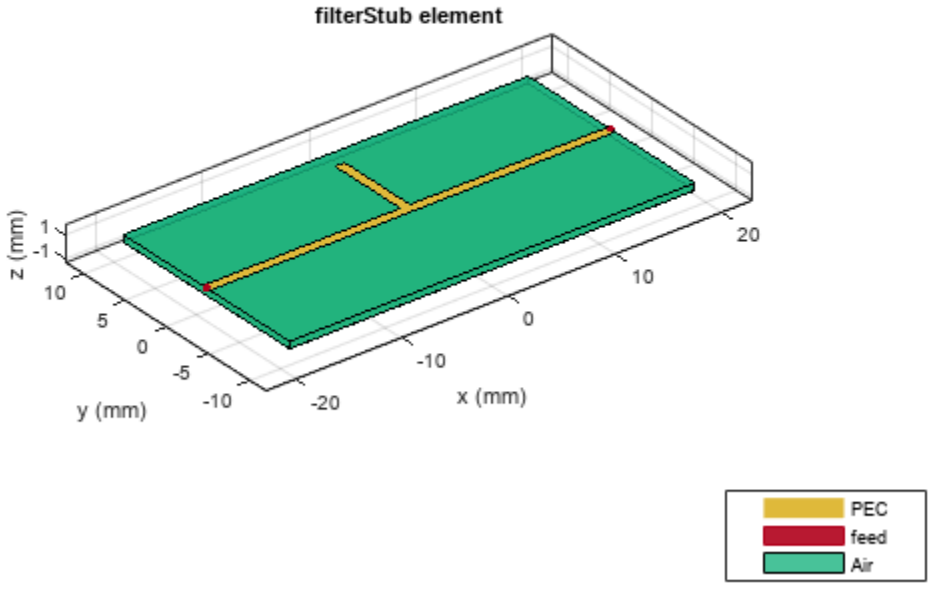
```
filt = filterStub
```

```
filt =
  filterStub with properties:

      PortLineLength: 0.0150
       PortLineWidth: 9.0000e-04
    SeriesLineLength: 0.0080
```

```
 SeriesLineWidth: 9.0000e-04
      StubLength: [0.0083 0.0083]
       StubWidth: [9.0000e-04 9.0000e-04]
     StubOffsetX: [-0.0090 0.0090]
   StubDirection: [1 0]
       StubShort: [0 0]
          Height: 0.0016
GroundPlaneWidth: 0.0200
       Substrate: [1x1 dielectric]
       Conductor: [1x1 metal]
```

The properties which control the stubs are `StubLength, StubWidth, StubOffsetX, StubDirection`, and `StubShort.` All these properties are vectors and can be used to create any number of stubs.

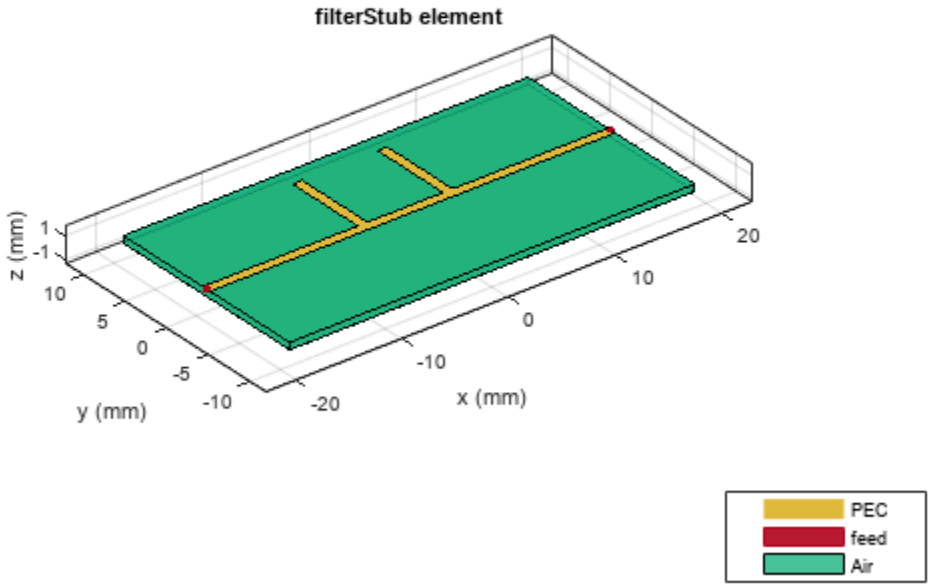
```
filt.StubLength = 0.0083;
filt.StubWidth = 0.0009;
filt.StubOffsetX = 0;
filt.StubDirection = 1;
filt.StubShort = 0;
d = dielectric('EpsilonR',6.15);
filt.Substrate = d;
filt.Height = 0.635e-3;
figure,show(filt);
```



Filter with two Open Stub - Type 2

In this variation, create two open circuited stubs by defining additional properties for the second stub.

```
filt.StubLength = [0.0083 0.0083];
filt.StubWidth = [0.0009 0.0009];
filt.StubOffsetX = [-0.004 0.004];
filt.StubDirection = [1 1];
filt.StubShort = [0 0];
d = dielectric('EpsilonR',6.15);
filt.Substrate = d;
filt.Height = 0.635e-3;
figure,show(filt);
```

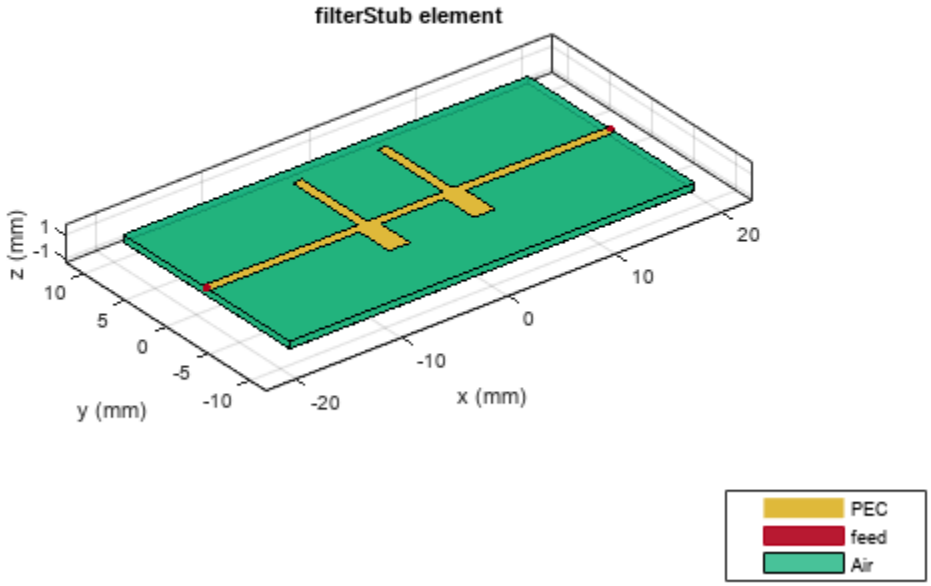


Filter with four Open Stub - Type 3

In this variation, create four open circuited stubs with different widths. The two stubs are placed on one side and other two on the opposite side. You can achieve this by changing the `StubDirection` property and assigning 1 for two stubs and 0 for the other two stubs.

```
filt.StubLength = [0.0083 0.0083 0.004 0.004];
filt.StubWidth = [0.0009 0.0009 0.002 0.002];
filt.StubOffsetX = [-0.004 0.004 -0.004 0.004];
filt.StubDirection = [1 1 0 0];
filt.StubShort = [0 0 0 0];
d = dielectric('EpsilonR',6.15);
```

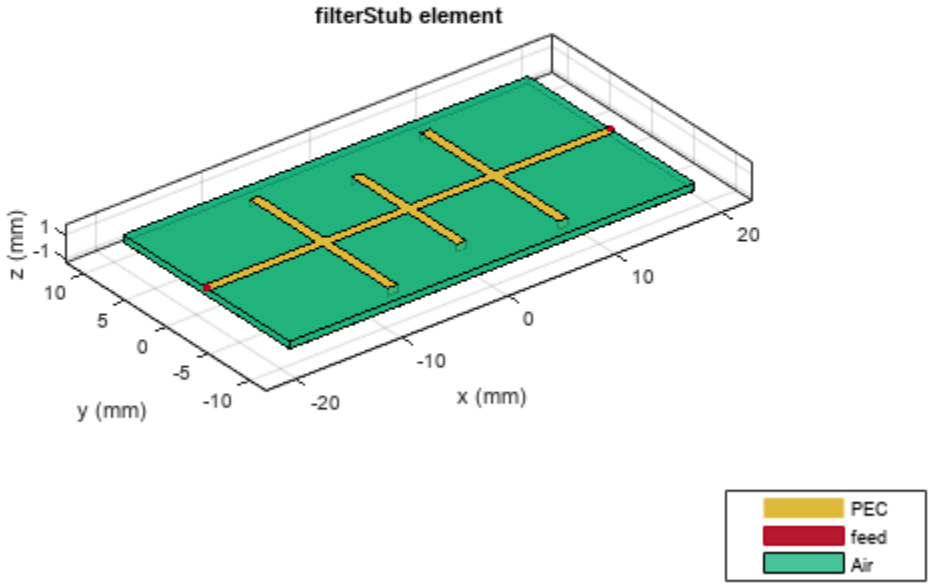
```
filt.Substrate = d;
filt.Height = 0.635e-3;
figure,show(filt);
```



Filter with Six Shorted Stubs

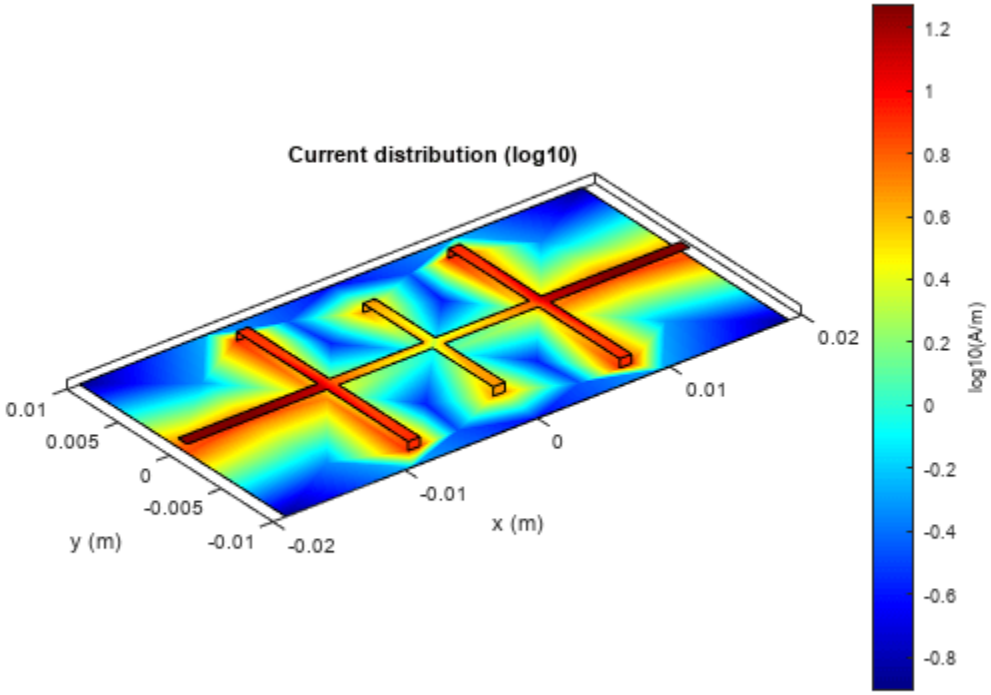
This variation of filter shows how to create a short-circuited stub. The `StubShort` property defines the stub as open or short. For this use case, you need to set this property as 1 for all the stubs.

```
filt.StubLength = [0.0083 0.0083 0.0063 0.0063 0.0083 0.0083];
filt.StubWidth = [0.0009 0.0009 0.0009 0.0009 0.0009 0.0009];
filt.StubOffsetX = [-0.008 0.008 0 0 -0.008 0.008];
filt.StubDirection = [1 1 1 0 0 0];
filt.StubShort = [1 1 1 1 1 1];
d = dielectric('EpsilonR',6.15);
filt.Substrate = d;
filt.Height = 0.635e-3;
figure,show(filt);
```



Use the `Current` method to plot the current distribution of the filter and excite both the ports using the `voltagePort` object. Set the `FeedVoltage` property as 1 to excite both the ports.

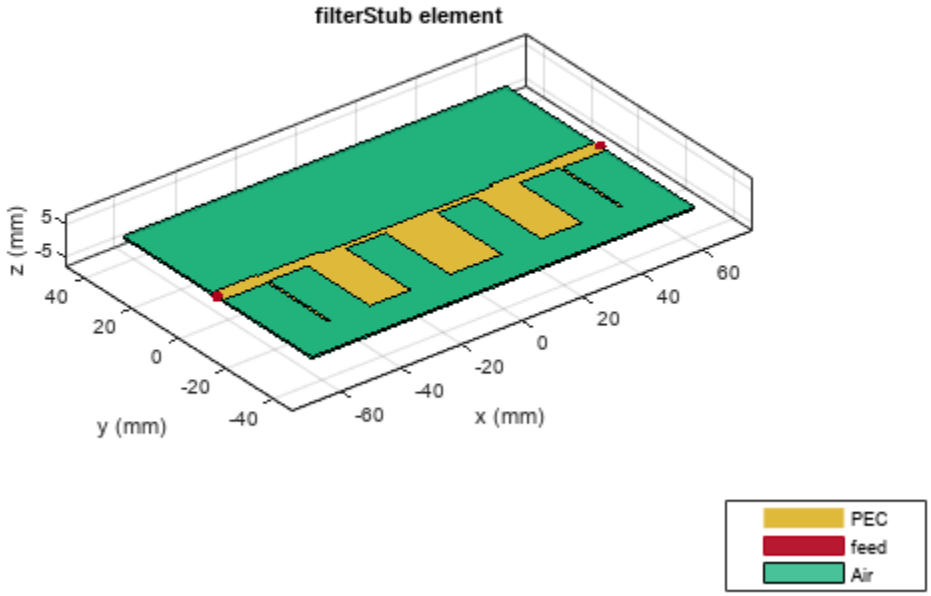
```
v = voltagePort(2);
v.FeedVoltage = [1 1];
figure;
current(filt,1e9,'scale','log10','Excitation',v);
```



Filter with Different Series Line Width

In this structure, use the `SeriesLineWidth` to control the width of the series line. The stubs are in the same direction but have different width.

```
filt.PortLineLength = 0.015;
filt.PortLineWidth  = 0.005;
filt.SeriesLineLength = [0.0200 0.0284 0.0284 0.0201];
filt.SeriesLineWidth = [0.0040 0.0021 0.0021 0.0040];
filt.StubLength = [0.0284 0.0284 0.0284 0.0284 0.0284];
filt.StubWidth = [7.9945e-04 0.0135 0.0172 0.0135 7.9945e-04];
filt.StubOffsetX =  [-0.0484 -0.0284 0 0.0284 0.0484];
filt.StubDirection = [0 0 0 0 0];
filt.StubShort = [0 0 0 0 0];
filt.GroundPlaneWidth = 80e-3;
figure,show(filt);
```



References

1 Santasri Koley, Lakhindar Murmu, and Biswajit Pal, A Pattern Reconfigurable Antenna for WLAN and WiMAX Systems

Microstrip Ultra-Wideband Bandpass Filter with Cascaded Broadband Bandpass and Bandstop Filters

This example shows how to create a compact ultra-wideband filter by connecting a broadband bandstop filter with a broadband bandpass filter.

In the recent years, wireless communication systems have increasing demands for broader bandwidth and research focuses on ultra-wideband filters. The frequency spectrum adopted for the ultra-wideband system in [1] is 3.1–10.6 GHz.

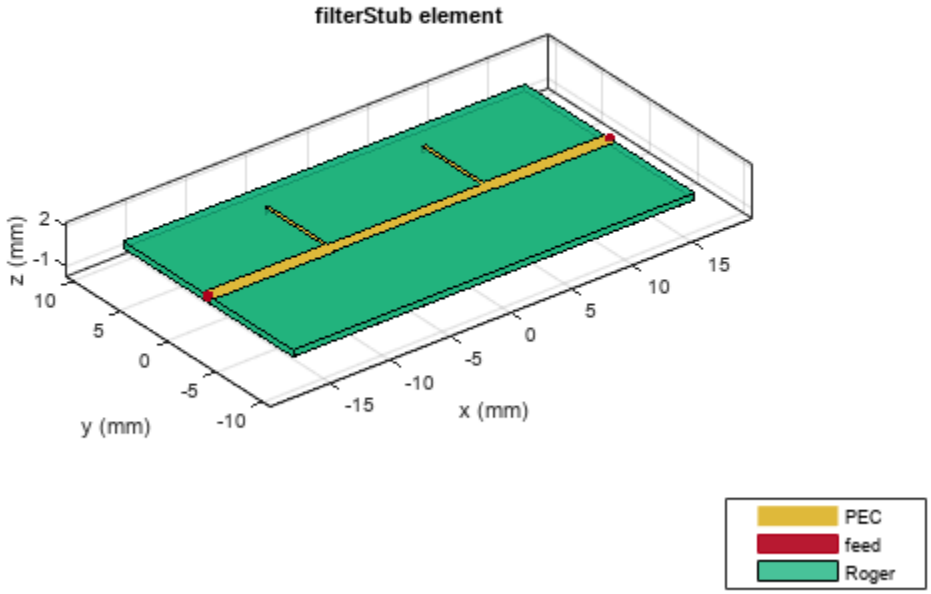
The proposed structure has a Rogers RO4003 substrate with dielectric constant of `3.38`, loss tangent of `0.0027`, and substrate thickness of 0.508 mm, respectively. The proposed filter exhibits a passband within 3.42–12 GHz and a return loss is greater than 15 dB. The filter also has a greater-than-15-dB insertion loss and the stopband is within 12.85–28.5 GHz.

Design of Broadband Bandpass filter

The ultra-wideband bandpass filter composes of a one-half wavelength transmission line shunted with two quarter-wavelength short stubs at the input and output port.

Create the bandpass filter using a `filterStub` object with the designed dimensions in [1] and visualize it.

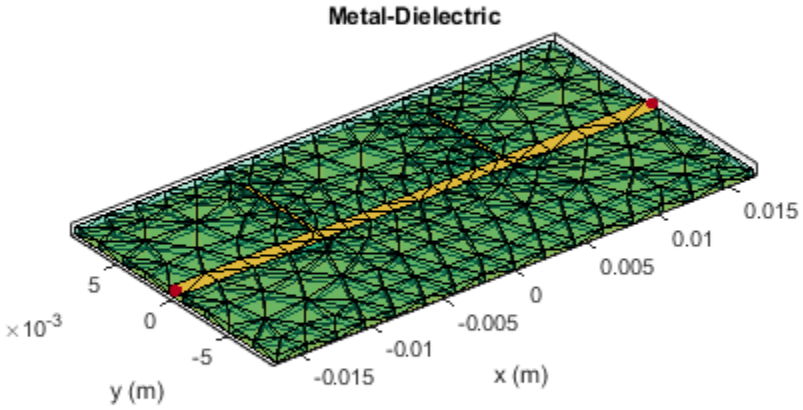
```
filterBpass = filterStub;
sub = dielectric('Name',{'Roger'},'EpsilonR',3.38,'LossTangent',0.0027,'Thickness',0.508e-3);
filterBpass.Substrate = sub;
filterBpass.Height = 0.508e-3;
filterBpass.StubLength = [6.75e-3,6.75e-3];
filterBpass.StubWidth = [0.3e-3,0.3e-3];
filterBpass.StubDirection = [1 1];
filterBpass.StubOffsetX = [-6.48e-3 6.48e-3];
filterBpass.StubShort = [1 1];
filterBpass.SeriesLineLength = 13.26e-3;
filterBpass.SeriesLineWidth = 0.96e-3;
filterBpass.PortLineLength = 10e-3;
filterBpass.PortLineWidth = 1.17e-3;
filterBpass.GroundPlaneWidth =  0.018;
figure; show(filterBpass);
```



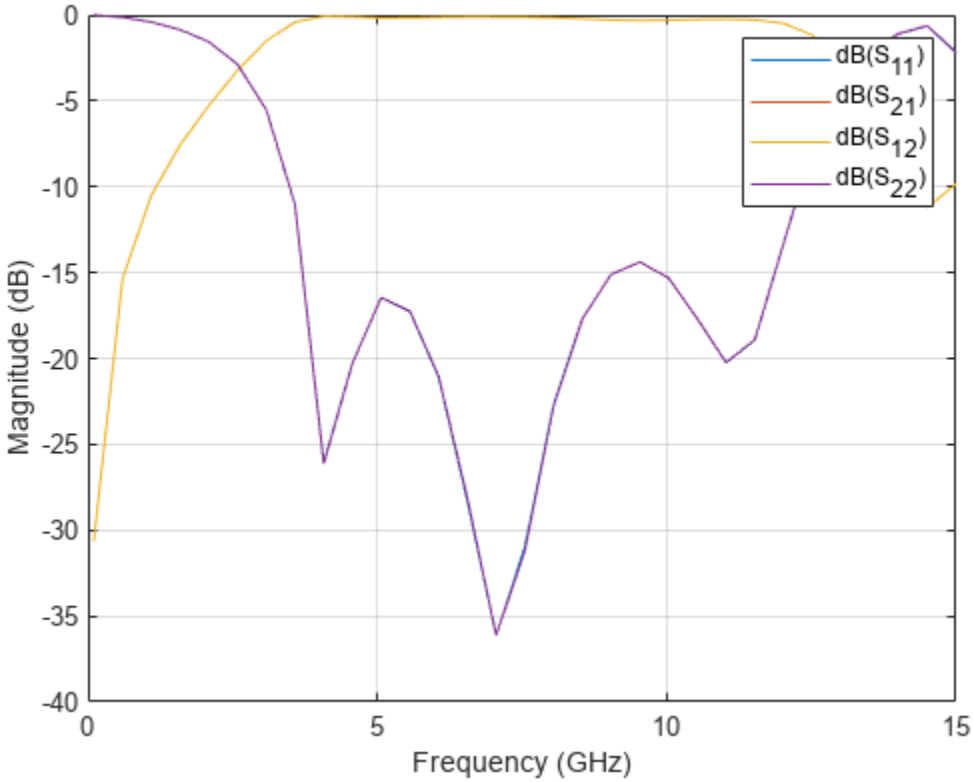
Analyze the s-parameters behavior of the filter at the UWB frequency range using manual meshing with MaxEdgeLength of 0.01.

```
figure; mesh(filterBpass,'MaxEdgeLength',0.01)
```

```
NumTriangles: 404
NumTetrahedra: 1032
NumBasis:
MaxEdgeLength: 0.01
MeshMode: manual
```




```
sparBpass = sparameters(filterBpass,linspace(0.1e9,15e9,31));
figure; rfplot(sparBpass)
```



Observe that the bandpass filter exhibits a passband behavior in the frequency range of 3.5 GHz to 12 GHz, with return loss greater than -10dB.

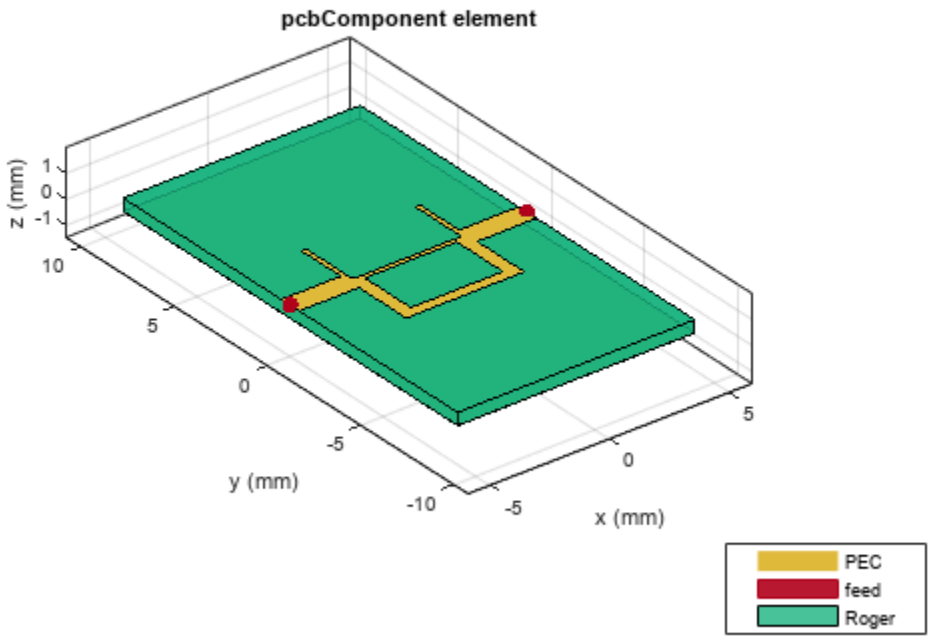
Design of Broadband Bandstop Filter

The structure of a broadband bandstop filter consists of one transmission line with a half-wavelength electric length in parallel with another transmission line with one-wavelength electric length.

Create the bandstop filter using `pcbComponent` object with the designed dimensions in [1], and visualize it.

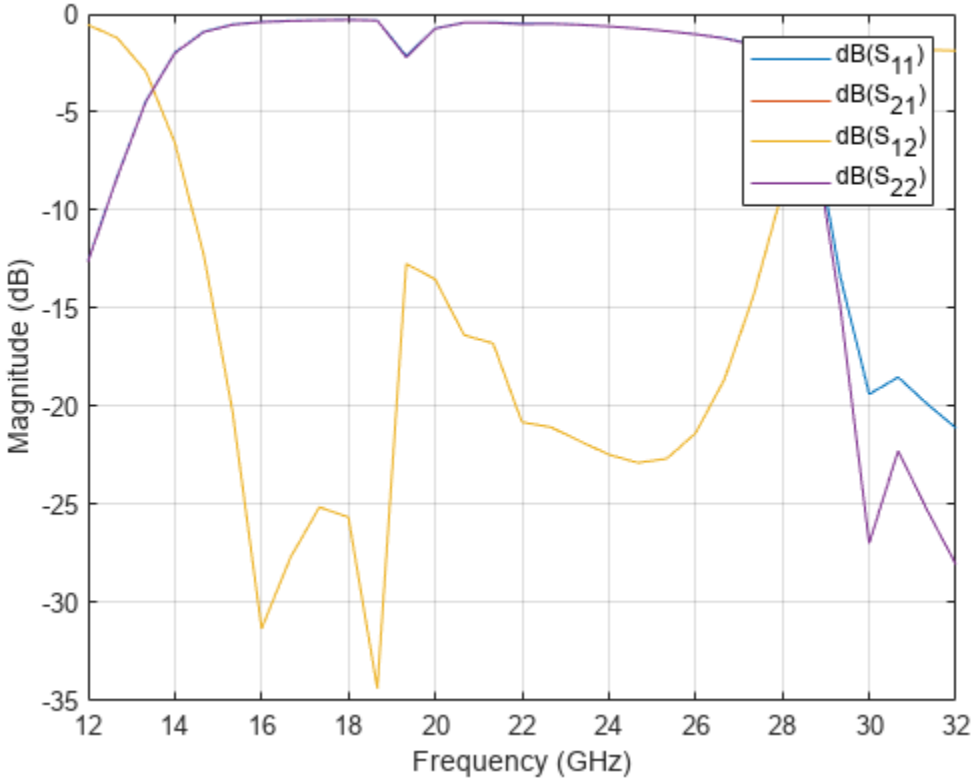
```
seriesLtop = 4.57e-3;
seriesWtop = 0.2e-3;
seriesWbtm = 0.45e-3;
shuntWtop = 0.2e-3;
shuntLtop = 2.48e-3;
shuntLbtm = 2.93e-3;
shuntWbtm = 0.55e-3;
PL = 2.5e-3;
PW = 0.97e-3;
gndL = (2*PL)+(2*shuntWtop)+seriesLtop;
gndW = 0.018;
h = 0.508e-3;
filterBstop = pcbComponent;
filterBstop.BoardThickness = h;
gnd = traceRectangular('Length',gndL,'Width',gndW);
filterBstop.BoardShape = gnd;
```

```
InPort =  traceRectangular('Length',PL,'Width',PW,'Center',[-gndL/2+PL/2,0]);
OutPort =  traceRectangular('Length',PL,'Width',PW,'Center',[gndL/2-PL/2,0]);
ShutLineTopIn =  traceRectangular('Length',shuntWtop,'Width',shuntLtop+seriesWtop/2,...
              'Center',[-gndL/2+PL+shuntWtop/2,shuntLtop/2+seriesWtop/2-seriesWtop/4]);
ShutLineTopOut =  traceRectangular('Length',shuntWtop,'Width',shuntLtop+seriesWtop/2,...
              'Center',[-(-gndL/2+PL+shuntWtop/2),shuntLtop/2+seriesWtop/2-seriesWtop/4]);
seriesLbtm = seriesLtop-(2*(shuntWbtm-shuntWtop));

SeriesLineTop =  traceRectangular('Length',seriesLtop+2*seriesWtop+PL/4,'Width',seriesWtop,...
              'Center',[0,0]);
ShutLineBtmIn =  traceRectangular('Length',shuntWbtm,'Width',shuntLbtm+seriesWtop/2,...
              'Center',[-gndL/2+PL+shuntWbtm/2,-shuntLbtm/2-seriesWtop/2+seriesWtop/4]);
ShutLineBtmOut =  traceRectangular('Length',shuntWbtm,'Width',shuntLbtm+seriesWtop/2,...
              'Center',[-(-gndL/2+PL+shuntWbtm/2),-shuntLbtm/2-seriesWtop/2+seriesWtop/4]);

SeriesLineBtm =  traceRectangular('Length',seriesLtop+seriesWbtm/2,'Width',seriesWbtm,...
              'Center',[0,-seriesWtop/2-shuntLbtm+seriesWbtm/2]);


Bstop = InPort+OutPort+ShutLineTopIn+ShutLineTopOut+ShutLineBtmIn+...
              ShutLineBtmOut+SeriesLineTop+SeriesLineBtm;

filterBstop.Layers ={Bstop,sub,gnd};
filterBstop.FeedLocations = [-gndL/2,0,1,3;gndL/2,0,1,3];
filterBstop.FeedDiameter = PW/2;
figure; show(filterBstop);
```



Analyze the s-parameters behavior of the filter in the frequency range of 12 to 32 GHz.

```
sparBstop = sparameters(filterBstop,linspace(12e9,32e9,31));
figure; rfplot(sparBstop)
```



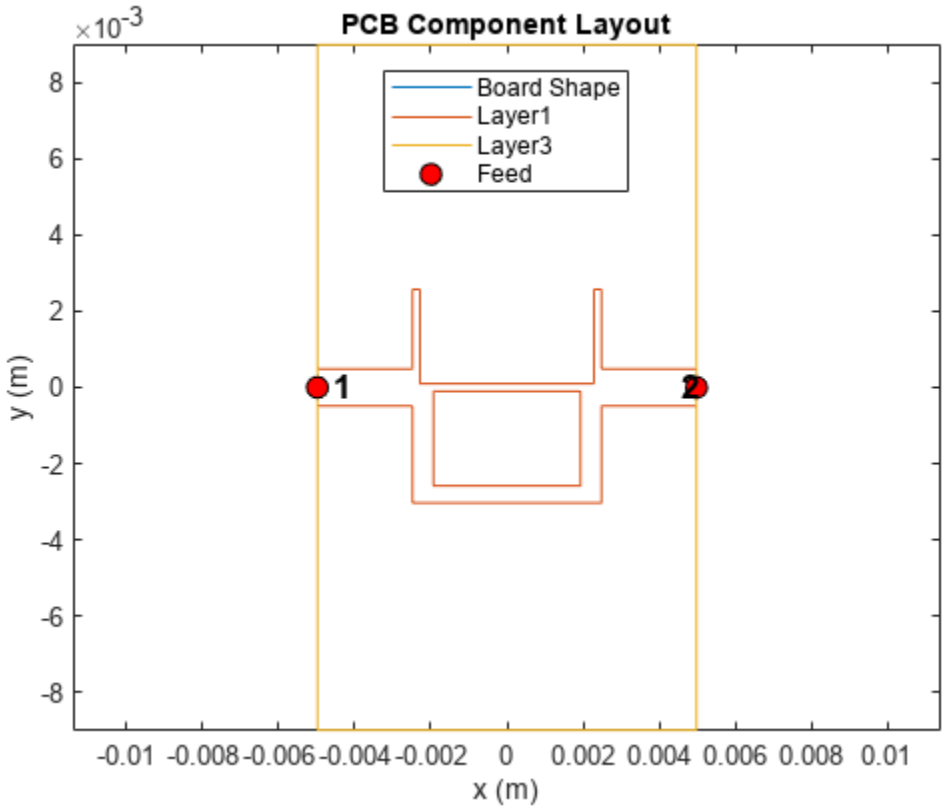
You can observe that the bandstop filter exhibits a bandstop behavior in the frequency range of 14 GHz to 28 Ghz with insertion loss greater than -10dB.

Design of Ultra-Wideband Filter

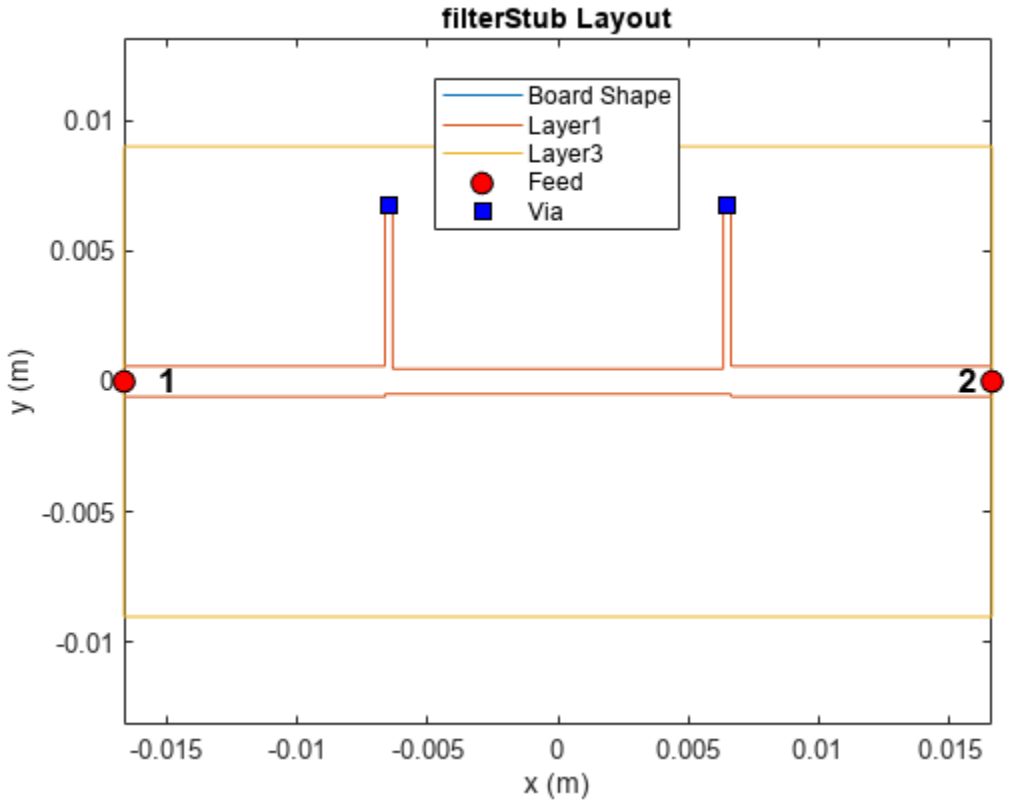
By cascading the bandpass filter and bandstop filter, you can create a new ultra-wideband filter to generate a wide passband and a broad stopband within the higher passband skirt simultaneously.

Cascade the broad bandpass and bandstop filter using `pcbCascade` object to create the ultra-wideband filter [1].

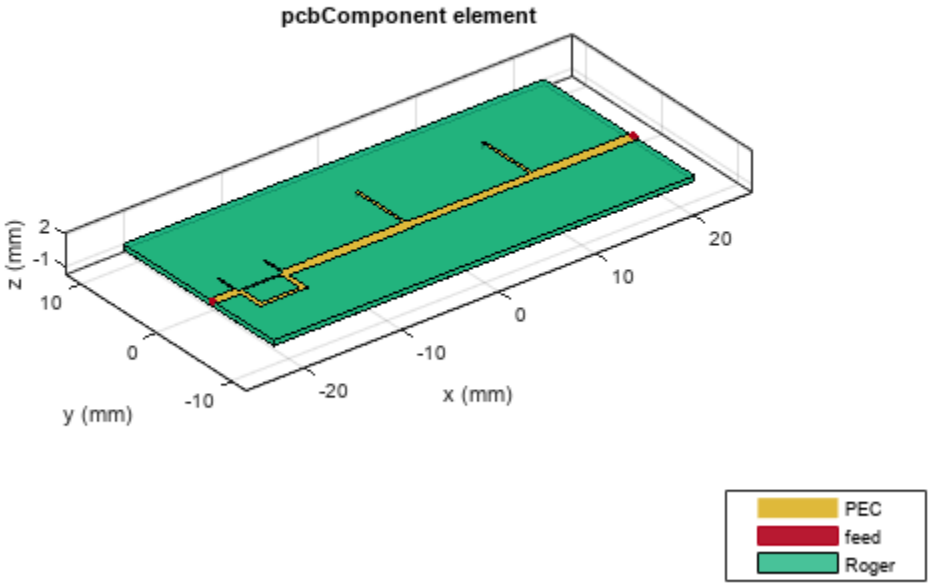
```
figure; layout(filterBstop);
```




```
figure; layout(filterBpass);
```



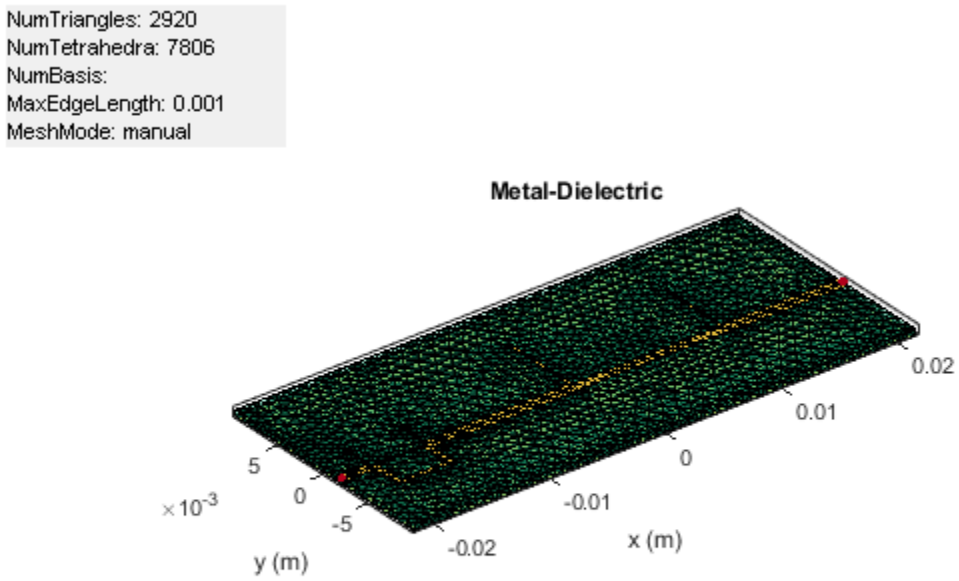

```
uwbFilter = pcbcascade(filterBstop,filterBpass,2,1);
gndL = uwbFilter.BoardShape.Length;
gndW = uwbFilter.BoardShape.Width + 10e-3;
gnd = traceRectangular('Length',gndL,'Width',gndW);
uwbFilter.BoardShape = gnd;
uwbFilter.Layers{3} = gnd;

viaYpt1 = uwbFilter.FeedLocations(1,2)+6.75e-3;
viaXpt1 = -(uwbFilter.BoardShape.Length)/2+(filterBstop.BoardShape.Length)+10e-3+0.3e-3/2;
viaXpt2 = -(uwbFilter.BoardShape.Length)/2+(filterBstop.BoardShape.Length)+10e-3+13.26e-3-0.3e-3-
uwbFilter.ViaLocations = [viaXpt1,viaYpt1,1,3;viaXpt2,viaYpt1,1,3];
uwbFilter.ViaDiameter = 0.3e-3/2;
uwbFilter.FeedDiameter = [0.97e-3/2,1.17e-3/2];
figure; show(uwbFilter);
```



Analyze the filter performance in frequency range of 0.1GHz to 32 Ghz.

Use the `mesh` function to manually change the mesh and set the MaxEdgeLength property to 0.001 m

```
figure; mesh(uwbFilter,'MaxEdgeLength',0.001);
```



Use the `memoryEstimate` function to calculate the memory required to solve the structure.

```
memEst = memoryEstimate(uwbFilter,32e9)
```

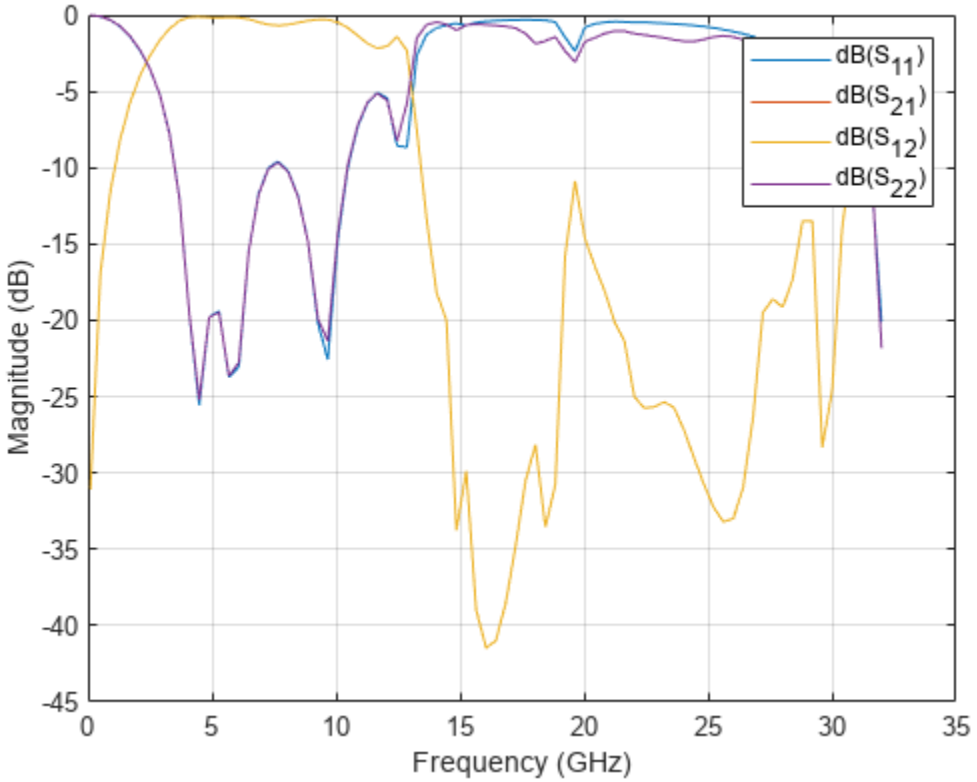
```
memEst =
'3.4 GB'
```

As the memory requirement is around 3.4 Ghz, you will need more time to run the analysis in this live script. On the machine with 64 GB of RAM and Intel(R) Xeon(R) W-2133 CPU clocked @ 3.60 GHz processor, it takes around 2511.21 seconds time to calculate the s-parameters for 51 points. Therefore, this example includes a MAT file containing the sparameters data to analyze the filter characteristics.

The code used to compute the results is available in the Return Loss for UWB Filter section.

Load the MAT file and plot the s-parameters to analyze the return loss and insertion loss using the `rfplot` function.

```
sparF =  load('sparam.mat');
figure; rfplot(sparF.spar);
```



Return Loss for UWB Filter

```
% spar = sparameters(uwbFilter,linspace(0.1e9,32e9,51));
% figure; rfplot(spar);
```

Conclusion

You can develop a ultra-wideband bandpass filter by cascading a broadband bandpass filter and a broadband bandstop filter. By carefully selecting impedances of the transmission lines, bandwidths of both broadband bandpass and bandstop filter are independently design. This filter exhibits the passband behavior in the frequency range 3.5 GHz to 12 Ghz and stopband behavior in the frequency range of 12 Ghz to 28 Ghz. The filter [1] is appropriate for implementation in ultra-wideband systems due to its simple structure and attractive performance.

Reference

Ching-Wen Tang, Ming-Guang Chen, 'A Microstrip Ultra-Wideband Bandpass Filter With Cascaded Broadband Bandpass and Bandstop Filters', IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES, VOL. 55, NO. 11, NOVEMBER 2007.

Computational Techniques

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

Method of Moments Solver for Metal and Dielectric Structures

In this section...
“MoM Formulation” on page 2-2 “Neighbor Region” on page 2-6 “Singularity Extraction” on page 2-7

Method of Moments computation technique for metal and dielectrics in a PCB.

A PCB 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 at the input port of the PCB component.
- I — Current vector that represents current on the PCB component surface.
- Z — Interaction matrix or impedance matrix that relates V to I. For calculating the interaction matrix, the effect of metal and dielectric parts iof the PCB antenna are taken separately.

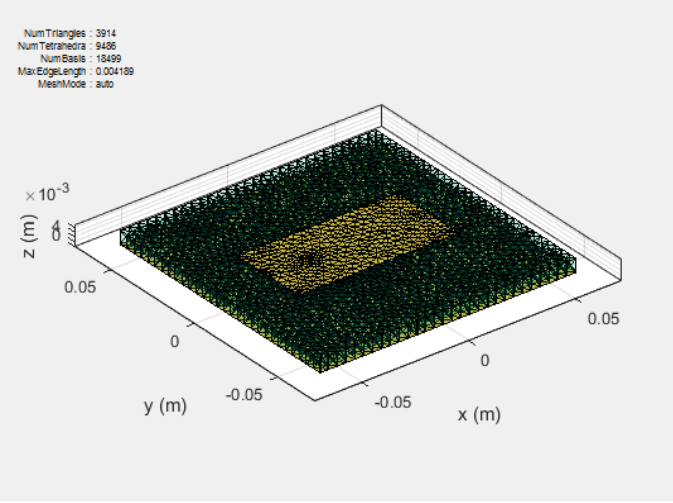
RF PCB Toolbox™ uses method of moments (MoM) to calculate the interaction matrix and solve system equations.

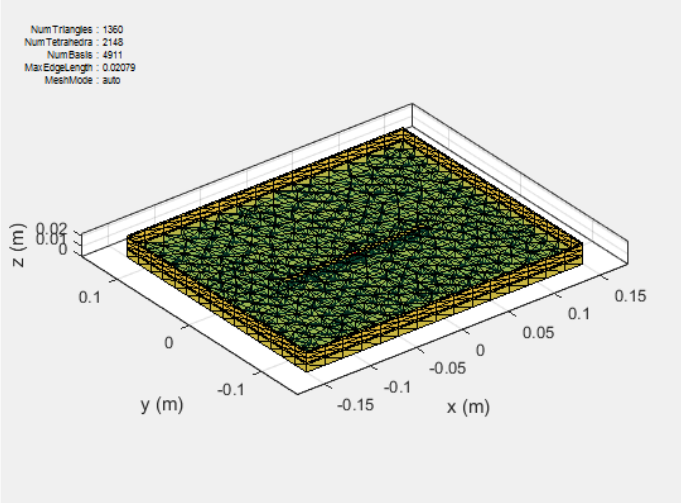
MoM Formulation

The MoM formulation is split into three parts.

Discretization of Metals and Dielectrics

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

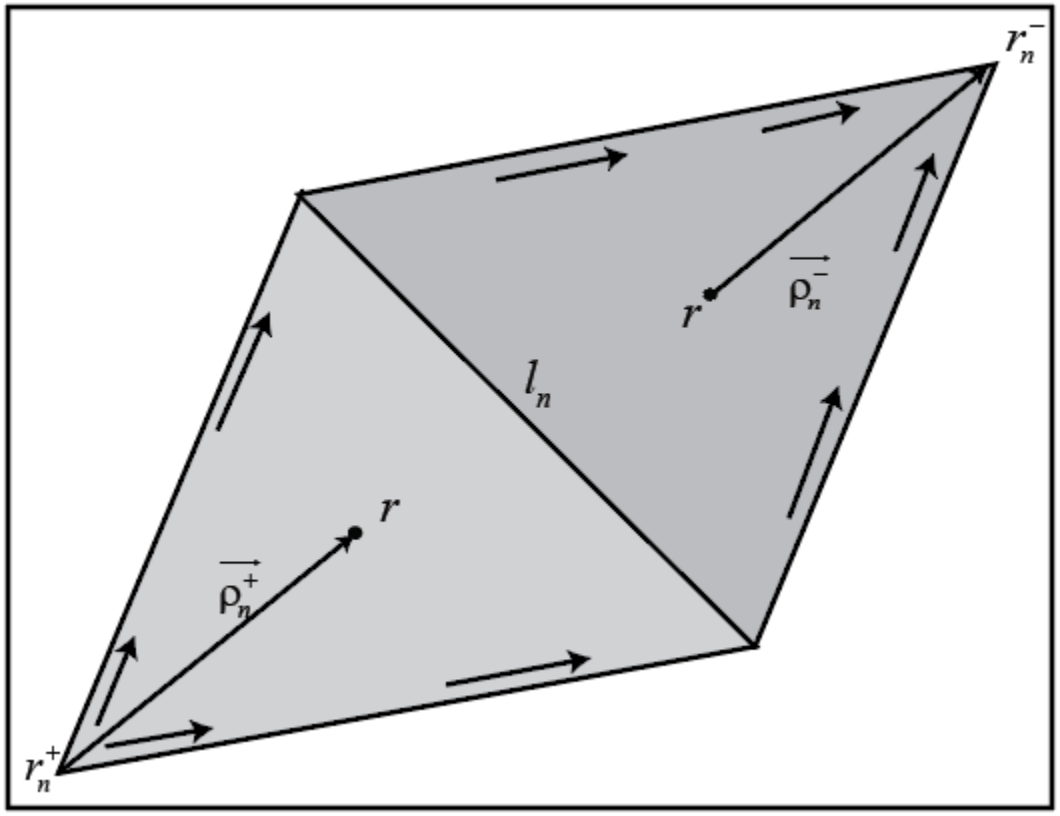




Basis Functions

Metals

To calculate the surface currents on the PCB, first define basis functions. RF PCB 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

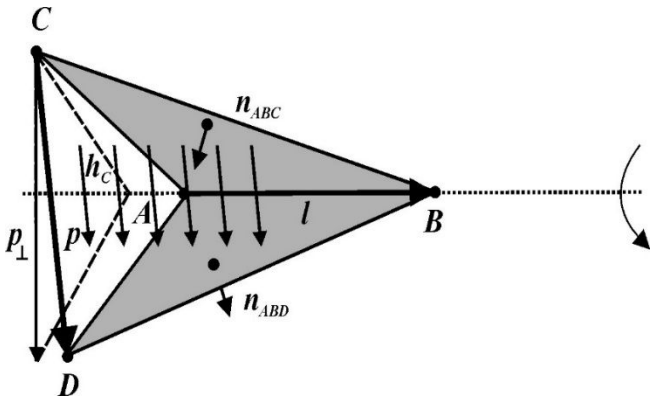
$$\nabla \cdot \vec{f}_n(\vec{r}) = \begin{cases} \frac{l_n}{A_n^+}, & \vec{r} \in t_n^+ \\ -\frac{l_n}{A_n^-}, & \vec{r} \in t_n^- \end{cases}$$

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.

Dielectrics

Basis functions are used to represent unknown quantities. In the case of a PCB component, the unknown quantities are the surface current on the metal structure and flux density due to dielectric volume. RF PCB Toolbox uses Rao-Wilton-Glisson (RWG) [2] basis functions.

For the dielectric volume, RF PCB 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

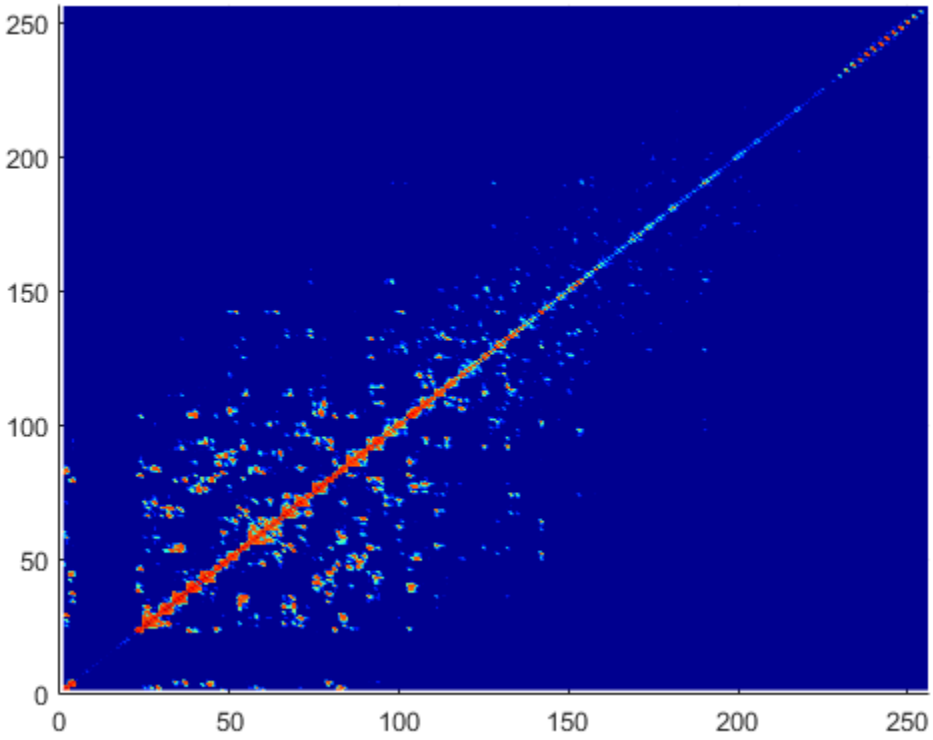
$$\vec{f} = c\vec{p}$$

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

Interaction Matrix

Metals

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 PCB surface. The final interaction matrix equations are:

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

where

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

To calculate the interaction matrix, excite the PCB by a voltage of 1 V at the input port. 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 PCB.

Dielectrics

The interaction matrix is a complex dense symmetric matrix. To fill out the interaction matrix, calculate the free-space Green's function between all the basis functions on the PCB 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.

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

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

$$\widehat{Z}_{mn}^{DD} = \sum_{p-1}^{P}\sum_{p'-1}^{P'}\frac{K_p}{\widehat{\varepsilon}_p}\int_{V_D}\vec{f}_{mp}(\vec{r})\cdot\vec{f}_{np'}(\vec{r})d\vec{r}$$

$$-\frac{\omega^2\mu_0}{4\pi}\sum_{p-1}^{P}\sum_{p'-1}^{P'}K_pK_{p'}\int_{V_D}\int_{V_{D'}}g(\vec{r},\vec{r'})\vec{f}_{mp}(\vec{r})\cdot\vec{f}_{np'}(\vec{r'})d\vec{r}d\vec{r'}$$

$$-\frac{1}{4\pi\varepsilon_0}\sum_{q-1}^{Q}\sum_{q'-1}^{Q'}\widehat{K}_q\widehat{K}_{q'}\int_{\Omega_q}\int_{\Omega_{q'}}g(\vec{r},\vec{r'})f_{\perp mq}(\vec{r})f_{\perp nq'}(\vec{r'})d_sd_{s'}\qquad m,n=1,....,N$$

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

$$Z_{mn}^{MD} = -\frac{\omega^2\mu_0}{4\pi}\sum_{p-1}^{2}\sum_{p'-1}^{P'}K_p\int_t\int_{V_{D'}}\vec{f}^{M}_n(\vec{r})\cdot\vec{f}_{mp'}(\vec{r'})g(\vec{r},\vec{r'})d\vec{r'}d_s$$

$$-\frac{1}{4\pi\varepsilon_0}\sum_{p-1}^{2}\sum_{q-1}^{Q}\widehat{K}_q\int_{t_D}\int_{\Omega_{q'}}(\nabla_s\cdot\vec{f}^{M}_n(\vec{r}))f_{\perp mq}(\vec{r'})g(\vec{r},\vec{r'})d_{\Omega'}d_s$$

$$m=1,....,N_D;n=1,...,N_M$$

$$Z_{mn}^{DM} = -\frac{j\omega\mu_0}{4\pi}\sum_{p-1}^{2}\sum_{p'-1}^{P'}K_{p'}\int_{V_D}\int_{S_{D'}}\vec{f}_{np}(\vec{r})\cdot\vec{f}^{M}_m(\vec{r'})g(\vec{r},\vec{r'})d_{s'}d\vec{r}$$

$$+\frac{1}{4\pi\varepsilon_0\omega}\sum_{p-1}^{2}\sum_{q-1}^{Q}\widehat{K}_q\int_{\Omega_q}\int_{S_{D'}}f_{\perp nq}(\vec{r})\cdot(\nabla_s\cdot\vec{f}^{M}_m(\vec{r'}))g(\vec{r},\vec{r'})d_{s'}d_\Omega$$

$$m=1,....,N_D;n=1,...,N_M$$

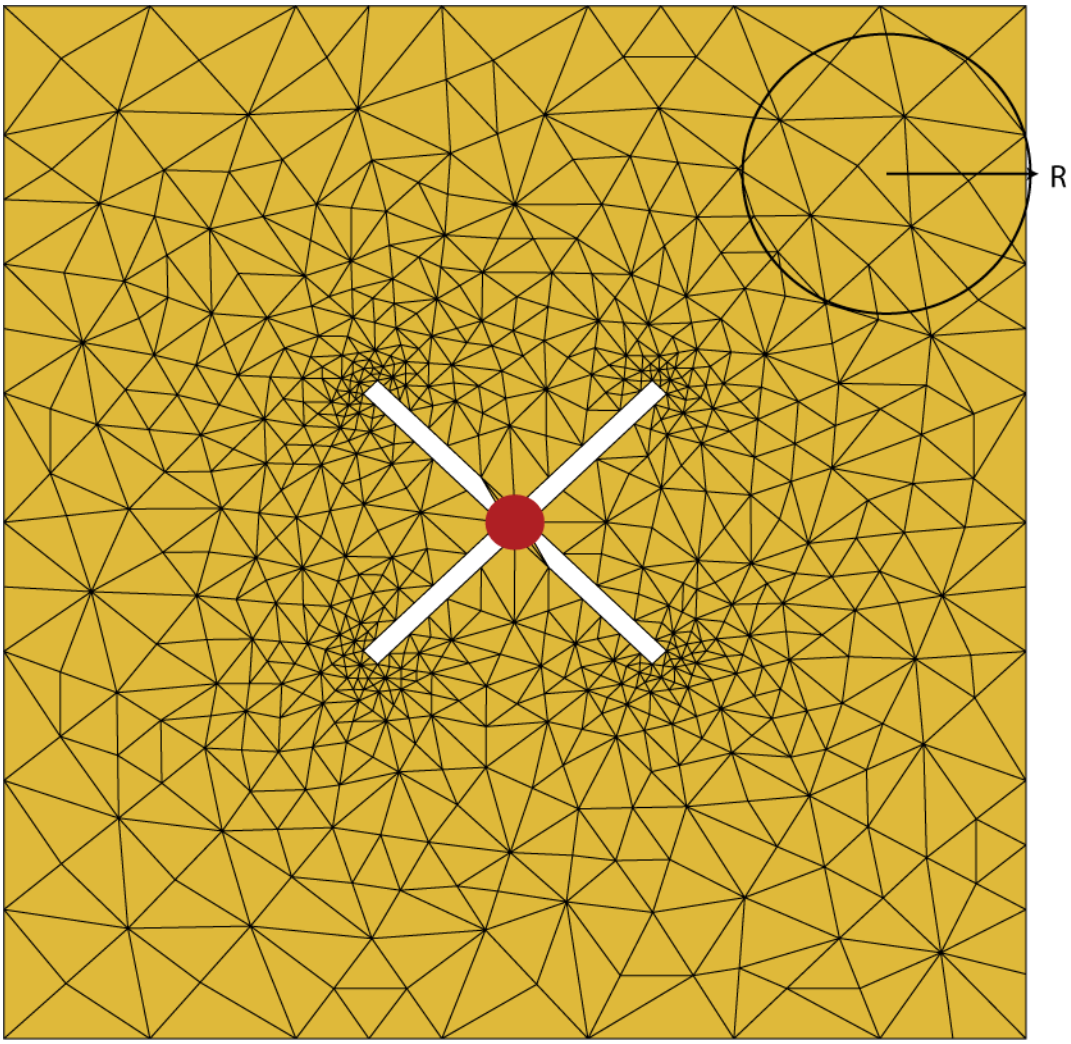
where

- $g(\vec{r},\vec{r'}) = \frac{\exp(-jkR)}{R}, R = \left|\vec{r}-\vec{r'}\right|$ 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

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

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. For a dielectric the *neighbor region* is based on the average size of the tetrahedron.

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.

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

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

The two integrals on the right-hand 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:

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

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

References

[1] Harringhton, R. F. *Field Computation by Moment Methods*. New York: Macmillan, 1968.

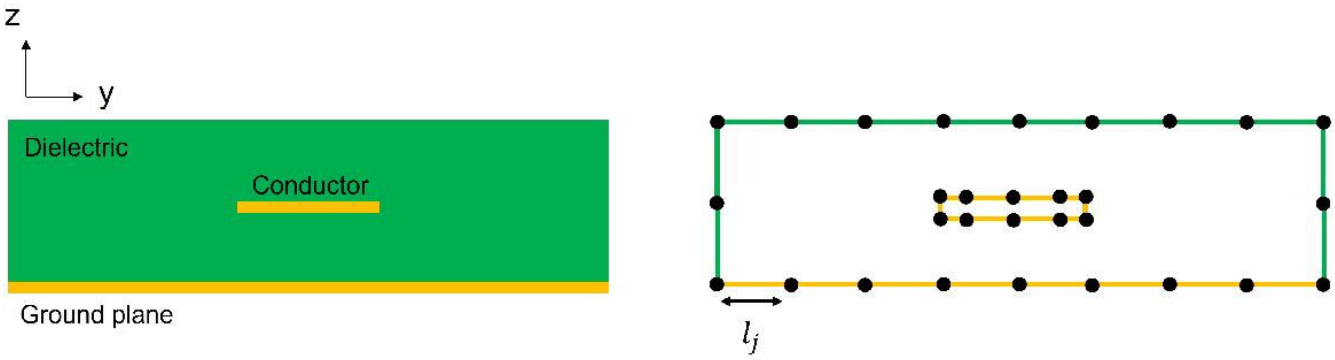
[2] Rao, S. M., D. R. Wilton, and A. W. Glisson. “Electromagnetic scattering by surfaces of arbitrary shape.” *IEEE. Trans. Antennas and Propagation*, Vol. AP-30, No. 3, May 1982, pp. 409–418.

[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.

2-D Field Solver

The 2-D field solver in RF PCB Toolbox™ allows you to model and analyze the cross sections of multiconductor transmission lines in a multilayered dielectric above the ground plane such as a microstrip line. For more detailed information, see [1]

The four primary transmission line parameters are the resistance R, the inductance L, the conductance G and the capacitance C. You can obtain these parameters by applying a total charge on the conductor-to-dielectric interfaces and a polarization charge on the dielectric-to-dielectric interfaces and solving the created electrostatic field created. The 2-D field solver uses pulse approximation for the total charge σ_T and discretizes the contours of the conductor-to-dielectric and dielectric-to-dielectric interfaces using straight-line segments. The densities of the segments are controlled by the distribution of nodes along the contours. This figure shows the discretization for a single-conductor transmission line in a single-layered dielectric above the ground plane.



At any point p in the YZ plane and above the ground plane, the potential ϕ is due to the combination of σ_T and the image of σ_T about the ground plane.

$$\phi(p) = \frac{1}{2\pi e_0} \sum_{j=1}^{J} \int \sigma_T(p') \left(\frac{|p - \widehat{p'}|}{|p - p'|} \right) dl'$$

where l_j is the contour of jth interface in the YZ plane, dl' is the differential element of length at p' on l_j, and $\widehat{p'}$ is the image of p' about the ground plane.

The electric field is then given by φ

$$E(p) = \frac{1}{2\pi e_0} \sum_{j=1}^{J} \int \sigma_T(p') \left(\frac{p - p'}{|p - p'|^2} - \frac{p - \widehat{p'}}{|p - \widehat{p'}|^2} \right) dl'$$

The normal component of a displacement field is continuous across the dielectric-to-dielectric interface. You can derive the second integral by substituting the interface conditions for the displacement fields with E_p. You can solve the set of integral equations for a total charge σ_T using the methods of moments solver and as described in [2]. You can analyze the skin effect and the conductor and dielectric losses.

References

.

[1] Djordjevis, Antonije R., Miodrag B. Bazdar, Tapan K. Sarkar and Roger F. Harrington, *Linpar for Windows: Matrix Parameters for Multiconductor Transmission Lines*. Artech House, 1999

[2] Harringhton, R. F. *Field Computation by Moment Methods*. New York, Macmillan, 1968

See Also