RF Toolbox

For Use with MATLAB®

- Computation
- Visualization
- Programming

User's Guide

Version 1

The MathWorks

How to Contact The MathWorks:

www.mathworks.com	Web
comp.soft-sys.matlab	Newsgroup
support@mathworks.com	Technical support
suggest@mathworks.com	Product enhancement suggestions
bugs@mathworks.com	Bug reports
doc@mathworks.com	Documentation error reports
service@mathworks.com	Order status, license renewals, passcodes
info@mathworks.com	Sales, pricing, and general information
508-647-7000	Phone
508-647-7001	Fax
The MathWorks, Inc. 3 Apple Hill Drive Natick, MA 01760-2098	Mail

For contact information about worldwide offices, see the MathWorks Web site.

RF Toolbox User's Guide

Revision History

June 2004	Online only	New for Version 1.0 (Release 14)
August 2004	Online only	Revised for Version 1.0.1 (Release 14+)
March 2005	Online only	Revised for Version 1.1 (Release 14SP2)
September 2005	Online only	Revised for Version 1.2 (Release 14SP3)

Contents

1 Getting Started

2 Working with RF Objects

3 RF Tool: An RF Analysis GUI

4 Function Reference

A AMP File Format

Index

1

Getting Started

What Is the RF Toolbox?

The RF Toolbox enables you to create and combine RF (Radio Frequency) circuits for simulation in the frequency domain with support for both nonlinear and noise data. You can read, write, analyze, combine, and visualize RF network parameters.

RF technology is used to design and test RF circuits for cable television, wireless LAN, and other wireless applications such as broadcasting, radar, satellite communications, microwave relay, and mobile telephony.

Work Directly with Network Parameter Data

You can work directly with your own network parameter data or with data from files. Functions enable you to:

- Read and write RF data in Touchstone® SnP, YnP, ZnP, and HnP formats, as well as the MathWorks AMP format.
- Convert among S, Y, Z, h, T, and ABCD network parameters
- Plot your data on X-Y plane and polar plane plots, as well as Smith® charts
- Calculate cascaded S-parameters and de-embed S-parameters from a cascaded network
- Calculate input and output reflection coefficients, and voltage standing-wave ratio (VSWR) at the reflection coefficient

Model RF Networks

You can also assemble RF networks from circuit objects that model:

- Passive networks and general circuit elements using Touchstone `.snp`, `.ynp`, `.znp`, and `.hnp` files.
- Amplifiers and mixers using data from Touchstone format `.s2p`, `.y2p`, `.z2p`, and `.h2p` files as well as the MathWorks format .amp files.
- Transmission lines based on their geometries.
- LC ladder filters based on their electrical interactions.

From these components and previously created network objects, you can create cascaded, hybrid, parallel, and series networks.

Functions associated with these objects enable you to:

- Analyze network parameters at specified frequencies
- Calculate needed parameters
- Plot network parameters in X-Y plane, polar plane, and Smith® chart formats.
- Extract parameters from an object
- Perform a variety of utility functions such as copying an object and listing valid parameters for visualization.

You can move network data among Touchstone format files or MathWorks format `.amp` files, your workspace, and circuit or data objects – wherever you need it.

The RF Blockset, which accepts data generated by the RF Toolbox, provides time-domain simulation

Analyze Circuits Interactively

A graphical tool, RF Tool, enables you to design, analyze, and visualize RF components and networks interactively, then export the circuits to your workspace or to a file for use with RF Toolbox functions and other circuit objects.

Other Features

- "RF Circuits" on page 1-3
- "Data Visualization" on page 1-4
- "Data Format Support" on page 1-4
- "Required Products" on page 1-5
- "Demos" on page 1-5

RF Circuits

The RF Toolbox provides circuit objects that enable you to model:

- Passive networks
- Amplifiers and mixers

- Transmission lines: coaxial, coplanar waveguide, microstrip, parallel-plate, two-wire, and general transmission
- SeriesRLC and shuntRLC circuits
- LC ladder filters: LC bandpass pi, LC bandpass tee, LC bandstop pi, LC bandstop tee, LC highpass pi, LC highpass tee, LC lowpass pi, and LC lowpass tee
- Networks: cascade, hybrid, parallel, and series

You can also model general circuit elements from data files.

Data Visualization

The RF Toolbox enables you to plot the network parameters of the circuits you create.

You can generate an X-Y plane plot, polar plane plot, or Smith chart of one or more selected network parameters directly from your data. You can also generate these plots from circuit objects you create using the RF Toolbox. See the `rfckt` and `rfdata` reference pages for information.

Data Format Support

The RF Toolbox supports the Touchstone SnP, YnP, ZnP, and HnP data file formats. It also introduces the MathWorks AMP format for amplifier data.

For information about the Touchstone file formats, see `http://www.eda.org/pub/ibis/connector/touchstone_spec11.pdf`.

For information about the AMP file format see “AMP File Format” on page A-1.

RF Analysis GUI

RF Tool is an RF analysis GUI that provides a visual interface for creating and analyzing RF (radio frequency) components and networks. You can create RF circuits quickly with the GUI. You can also import and export circuits from the MATLAB® workspace and RF data files.

RF Tool also provides the ability to set circuit parameters, analyze circuits, view their resulting S-parameter data, and visualize the data using X-Y plane plots, polar plane plots, and Smith charts.

Required Products

The RF Toolbox requires MATLAB. It provides simulation in the frequency domain. The RF Blockset, which can accept data generated by the RF Toolbox, provides time-domain simulation.

Demos

Demos of the RF Toolbox capabilities are available on the Demos tab of the MATLAB Help browser. These demos show examples of

- RF data objects
- RF circuit objects
- De-embedding S-parameters
- Placing circles on a Smith chart
- Designing impedance matching networks

Help for Objects

Follow these instructions to get specific information about RF Toolbox objects and their methods. Note that methods are treated and referred to as functions in the rest of this user's guide. For help in using objects, see Chapter 2, "Working with RF Objects."

Lists of Objects and Methods

To get a list of available circuit objects and methods, type `help rfckt` or `doc rfckt` at the command line.

Similarly for data objects, type `help rfdata` or `doc rfdata`.

Method Descriptions

To get detailed descriptions of the methods for circuit (`rfckt`) and data (`rfdata`) objects:

- At the command line, type `doc` *`methodname`*. For example, type `doc analyze`. If more than one product has a method or function by that name, MATLAB returns a list from which you can choose.
- At the command line, type `help rfckt.`*`objecttype`*`.`*`methodname`* for circuit objects, or `help rfdata.data.`*`methodname`* for data objects. For example, type `help rfckt.amplifier.analyze`.

Object and Property Descriptions

To get detailed information about a specific object and its properties:

- At the command line, type `doc rfckt.`*`objecttype`* or `doc rfdata.data`. For example, type `doc rfckt.amplifier`, `doc rfckt.lcbandpasspi`, or `doc rfdata.data`.
- At the command line, type help rfckt.objecttype or help rfdata.data. For example, type `help rfckt.amplifier`, or `help rfdata.data`.

2

Working with RF Objects

Overview

The RF Toolbox uses circuit objects to create

- Circuit components such as amplifiers, transmission lines, and ladder filters
- RLC network components
- Networks of RF components. Networks can be cascaded, parallel, series, or hybrid. They can include both circuit and network components.

The RF Toolbox also uses data objects, created from files, to hold analyzed date for existing components or parameter data for general components.

This chapter explains concepts you need to know to work with these objects.

Creating an Object

You can create a new object by doing one of the following

- "Constructing a New Object" on page 2-3
- "Copying an Existing Object" on page 2-5

Constructing a New Object

Use the `rfckt` function to construct a new circuit object such as an amplifier or transmission line. Objects can be amplifiers, mixers, transmission lines, ladder filters, or networks.

Each type of object has a name. For example, an amplifier is an `rfckt.amplifier` object. A cascaded network is an `rfckt.cascade` object. The following table lists the types of objects you can create.

Type of Object	Description
`rfckt.amplifier`	Amplifier, described by a data file
`rfckt.cascade`	Cascaded network,
`rfckt.coaxial`	Coaxial transmission line
`rfckt.cpw`	Coplanar waveguide transmission line
`rfckt.datafile`	General circuit, described by a data file
`rfckt.hybrid`	Hybrid connected network
`rfckt.lcbandpasspi`	LC bandpass pi network
`rfckt.lcbandpasstee`	LC bandpass tee network
`rfckt.lcbandstoppi`	LC bandstop pi network
`rfckt.lcbandstoptee`	LC bandstop tee network
`rfckt.lchighpasspi`	LC highpass pi network
`rfckt.lchighpasstee`	LC highpass tee network

Type of Object	Description
`rfckt.lclowpasspi`	LC lowpass pi network
`rfckt.lclowpasstee`	LC lowpass tee network
`rfckt.microstrip`	Microstrip transmission line
`rfckt.mixer`	Mixer, described by a data file
`rfckt.parallel`	Parallel connected network
`rfckt.parallelplate`	Parallel-plate transmission line
`rfckt.series`	Series connected network
`rfckt.seriesrlc`	Series RLC network
`rfckt.shuntrlc`	Shunt RLC network
`rfckt.twowire`	Two-wire transmission line
`rfckt.txline`	General transmission line

Every type of object has predefined fields called properties. The properties define the characteristics of a particular object. You can specify object property values by either:

- Specifying the property values when you create the object
- Creating an object with default property values, and changing some or all of the property values later

Example. This example creates a microstrip transmission line object with default properties. The output `h` is the handle of the newly created transmission line object.

```
h = rfckt.microstrip
```

The RF Toolbox lists the properties of the transmission line you created along with the associated default property values.

```
h =
            Name: 'Microstrip Transmission Line'
           nPort: 2
```

```
   AnalyzedResult: []
               Z0: []
               PV: []
             Loss: []
       LineLength: 0.0100
         StubMode: 'None'
      Termination: 'None'
            Width: 6.0000e-004
           Height: 6.3500e-004
        Thickness: 5.0000e-006
         EpsilonR: 9.8000
        SigmaCond: Inf
      LossTangent: 0
```

The `rfckt.microstrip` reference page describes these properties in detail.

For examples of setting object properties, see "Properties and Property Values" on page 2-6.

Copying an Existing Object

If you already have an object with all or most property values set the way you want them, you can create a new one with the same property values by copying the first object. For example,

```
h2 = copy(h);
```

creates a new object which has the same property values as the microstrip transmission line object with handle `h`. You can later change specific property values for this copy.

Note The syntax `h2 = h` copies only the object handle and does not create a new object.

Properties and Property Values

All circuit (`rfckt`) and data (`rfdata`) objects have properties associated with them. The properties define the characteristics of a particular object.

Each property associated with an object is assigned a value. You can set the values of many properties or you can accept the default values. Some properties have read-only values.

To learn about properties that are specific to a specific type of circuit or data object, see the reference page for that type of object. For example, the `rfckt.amplifier` reference page describes the properties of amplifier objects.

Note The `rfckt` and `rfdata` reference pages list the available types of circuit and data objects and provide links to their reference pages.

• "Setting Property Values" on page 2-6
• "Retrieving Property Values" on page 2-8

Setting Property Values

You can set circuit and data object property values when you construct the object or at a later time using the `set` command.

• "Setting Property Values at Construction" on page 2-6
• "Setting Property Values for an Existing Object" on page 2-7

Setting Property Values at Construction

To set a property directly when you construct an object, include a property/value pair in the argument list of the object construction command. A property/value pair consists of:

- A string for the property name you want to set followed by a comma
- The associated property value.

Include as many property names in the argument list as there are properties you want to set directly. Any property values you do not set, retain their default values. The circuit and data object reference pages list the valid values as well as the default value for each property.

This example creates a coaxial transmission line circuit object. Note that RF Toolbox lists the available properties and their values.

```
h = rfckt.coaxial('LineLength',0.05)

h =
           Name: 'Coaxial Transmission Line'
          nPort: 2
 AnalyzedResult: []
             Z0: []
             PV: []
           Loss: []
     LineLength: 0.0500
       StubMode: 'None'
    Termination: 'None'
    OuterRadius: 1.0000e-003
    InnerRadius: 5.0000e-005
            MuR: 1
       EpsilonR: 1
      SigmaCond: Inf
      SigmaDiel: 0
```

Setting Property Values for an Existing Object

Once you construct an object, you can modify its property values using the `set` command. You can use the `set` command to both:

- Set specific property values
- Display a listing of all property values you can set

For example, this code creates a copy of the coaxial transmission line from the previous example then changes it to be a series stub with open termination.

```
h2 = copy(h);
set(h2,'StubMode','series','Termination','open')
```

Note When you set any object property values, the strings for property names and their values are case-insensitive. In addition, you only need to type the shortest uniquely identifying string for the property name. You could have written the previous function call as

```
set(h2,'st','series','t','open')
```

To display a list of all properties you can set for a specific object, use the `set` command without specifying any property/value pairs. This example list the properties you can set for the coaxial transmission line `h2`.

```
set(h2)

ans =
     LineLength: {}
       StubMode: {}
    Termination: {}
    OuterRadius: {}
    InnerRadius: {}
            MuR: {}
       EpsilonR: {}
      SigmaCond: {}
      SigmaDiel: {}
```

Retrieving Property Values

For an existing object, you can retrieve its property values using the `get` command. You can use the `get` command to both

- Retrieve specific property values for an object
- Display a list of properties associated with an object and their current values

For example, this code retrieves the value of the inner radius and outer radius for the coaxial transmission line in the previous example.

```
ir = get(h2,'InnerRadius')
or = get(h2,'OuterRadius')
```

```
ir =
  5.0000e-005

or =
  1.0000e-003
```

To display a list of properties associated with a specific object as well as their current values, use the `get` command without specifying a property name.

```
get(h2)
          Name: 'Coaxial Transmission Line'
         nPort: 2
AnalyzedResult: []
            Z0: []
            PV: []
          Loss: []
    LineLength: 0.0500
      StubMode: 'series'
   Termination: 'open'
   OuterRadius: 1.0000e-003
   InnerRadius: 5.0000e-005
           MuR: 1
      EpsilonR: 1
     SigmaCond: Inf
     SigmaDiel: 0
```

Note that this list includes read-only properties that do not appear when you type `set(h2)`. For a coaxial transmission line object, the read-only properties are `Name`, `nPort`, `AnalyzedResult`, `Z0`, `PV`, and `Loss`. The `Name` and `nPort` properties are fixed by the RF Toolbox. The remaining read-only property values are calculated and set by the toolbox when you analyze the component at specified frequencies.

Functions Acting on Objects

The RF Toolbox provides a variety of functions that act on circuit (`rfckt`) and data (`rfdata`) objects. The following table lists these functions and tells you the types of objects on which each can act. These functions are also referred to as methods.

"Examples" on page 2-12 illustrates the use of these functions.

Function	Types of Objects	Description
`analyze`	All circuit objects	Analyze a circuit object in the frequency domain.
`calculate`	All circuit objects	Calculate specified parameters for a circuit object.
`copy`	All circuit and data objects	Copy a circuit or data object.
`extract`	`rfdata.data`, `rfdata.network`	Extract the specified network parameters from a data object and return the result in a matrix.
`getdata`	All circuit objects	Create data object containing analyzed result of a specified circuit object.
`getz0`	`rfckt.txline`, `rfckt.rlcgline`, `rfckt.twowire`, `rfckt.parallelplate`, `rfckt.coaxial`, `rfdata.microstrip`, and `rfckt.cpwr`	Get characteristic impedance of a transmission line.
`listformat`	All circuit objects	List valid formats for a specified circuit object parameter.
`listparam`	All circuit objects	List valid parameters of a specified circuit object.

Function	Types of Objects	Description
`plot`	All circuit objects	Plot the specified circuit object parameters on an X-Y plane.
`polar`	All circuit objects	Plot the specified circuit object parameters on polar coordinates.
`read`	`rfckt.datafile`, `rfckt.passive`, `rfckt.amplifier`, `rfckt.mixer`,and `rfdata.data`	Read RF data from a file to a new or existing circuit or data object.
`restore`	`rfckt.datafile`, `rfckt.passive`, `rfckt.amplifier`, `rfckt.mixer`,and `rfdata.data`	Restore data to original frequencies of `NetworkData` for plotting.
`smith`	All circuit objects	Plot the specified circuit object parameters on a Smith chart.
`write`	All circuit objects and `rfdata.data`	Write RF data from a circuit or data object to a file.

Examples

These examples show you how to perform some basic operations with RF objects.

RF Circuit Objects

In this example, you create three circuit (`rfckt`) objects: two transmission lines and an amplifier. You visualize the amplifier data using RF Toolbox functions and retrieve frequency data that was read from a file into the amplifier `rfckt` object. Then you analyze the amplifier over a different frequency range and visualize the results.

Next, you cascade the three circuits to create a cascaded `rfckt` object. Then you analyze the cascaded network and visualize its S-parameters over the original frequency range of the amplifier. Finally, you plot the S11, S22, and S21 parameters and noise figure of the cascaded network.

1 **Create three rfckt objects and view their properties.** Create a default transmission line, an amplifier described by the data in the data file `'default.amp'`, and a second transmission line. Use the `get` command to view the properties of the first two `rfckt` objects. Use the `methods` command to view the methods of the third circuit object.

   By setting the interpolation method for the amplifier to `'cubic'`, you anticipate the interpolation you perform later in this example when you analyze the amplifier over a different frequency range.

```
% Create three circuit objects
FirstCkt = rfckt.txline;
SecondCkt = rfckt.amplifier('IntpType','cubic');
read(SecondCkt, 'default.amp');
ThirdCkt = rfckt.txline('LineLength',0.025,'PV',2.0e8);
```

```
% View their properties
PropertiesOfFirstCkt = get(FirstCkt)
PropertiesOfSecondCkt = get(SecondCkt)
MethodsOfThirdCkt = methods(ThirdCkt)
```

The toolbox displays the following output.

```
PropertiesOfFirstCkt =
           Name: 'Transmission Line'
          nPort: 2
 AnalyzedResult: []
     LineLength: 0.0100
       StubMode: 'None'
    Termination: 'None'
           Freq: 1.0000e+009
             Z0: 50
             PV: 299792458
           Loss: 0
       IntpType: 'linear'

PropertiesOfSecondCkt =
          Name: 'Amplifier'
         nPort: 2
AnalyzedResult: [1x1 rfdata.data]
      IntpType: 'cubic'
   NetworkData: [1x1 rfdata.network]
     NoiseData: [1x1 rfdata.noise]
 NonlinearData: [1x1 rfdata.power]

MethodsOfThirdCkt =
    'analyze'
    'calckl'
    'calculate'
    'calczin'
    'checkfrequency'
    'checkimpedance'
    'checkproperty'
    'checkreadonlyproperty'
    'convertfreq'
    'destroy'
```

```
'disp'
'getdata'
'getz0'
...
```

2 Change properties of rfckt objects. Use the `set` command to change the line length of the first transmission line (`FirstCkt`).

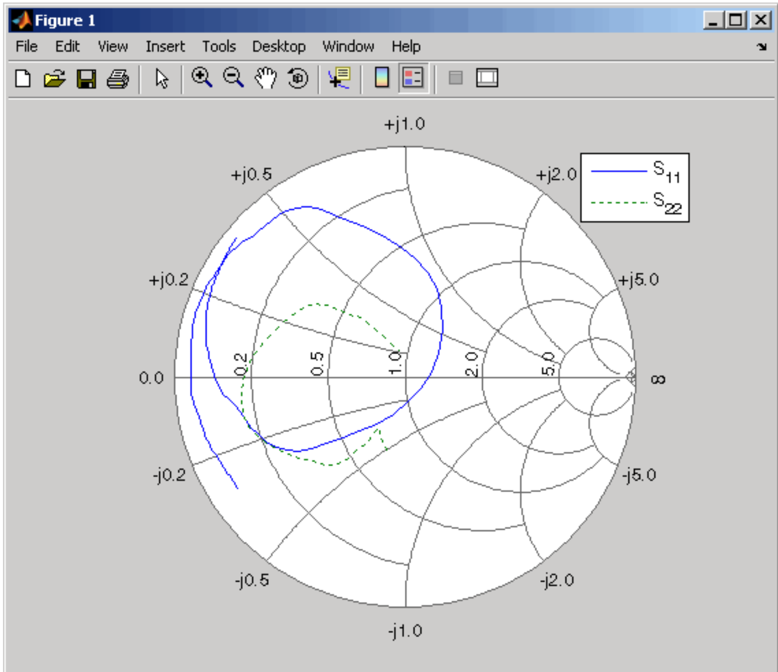
```
DefaultLength = get(FirstCkt,'LineLength')
set(FirstCkt,'LineLength',.001);
NewLength = get(FirstCkt,'LineLength')
```

The toolbox displays the following output.

```
DefaultLength =
    0.0100

NewLength =
  1.0000e-003
```

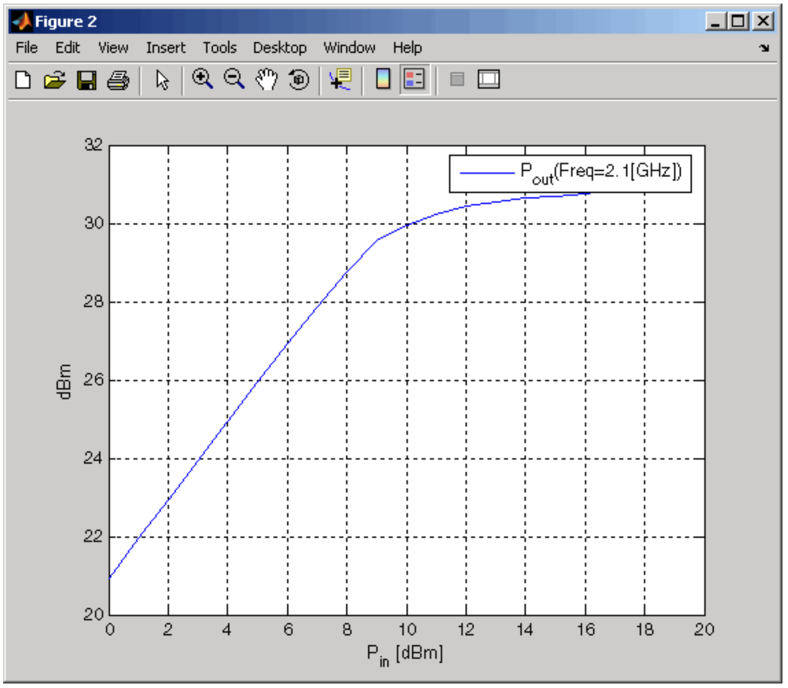
3 Plot the amplifier S11 and S22 parameters. Use the `smith` command to plot the original S11 and S22 parameters of the amplifier (`SecondCkt`) on a Z Smith chart. In step 1, you used the `read` command to take these parameters from the `default.amp` file. The amplifier's S-parameters range over the frequencies 1 GHz to 2.9 GHz.

```
lineseries1 = smith(SecondCkt,'S11','S22');
set(lineseries1(1), 'LineStyle','-', 'LineWidth', 1);
set(lineseries1(2), 'LineStyle',':', 'LineWidth', 1);
legend show
```



4 **Plot the amplifier Pin-Pout data.** Use the RF Toolbox `plot` command to plot the amplifier (`SecondCkt`) Pin-Pout data, in dBm, at 2.1 GHz on an X-Y plane.

```
figure
plot(SecondCkt,'Pout','dBm');
legend show
```



5 Get the original frequency data and the result of analyzing the amplifier over these frequencies. When the RF Toolbox reads data from a file into an amplifier object (`SecondCkt`), it also analyzes the amplifier over the frequencies saved in the file and stores the result in a property called `AnalyzedResult`. The following code gets the frequency values, which range from 1 GHz to 2.9 GHz, and the result of analyzing the amplifier at these frequencies.

```
f = SecondCkt.AnalyzedResult.Freq;
data = SecondCkt.AnalyzedResult
```

The toolbox displays the following output.

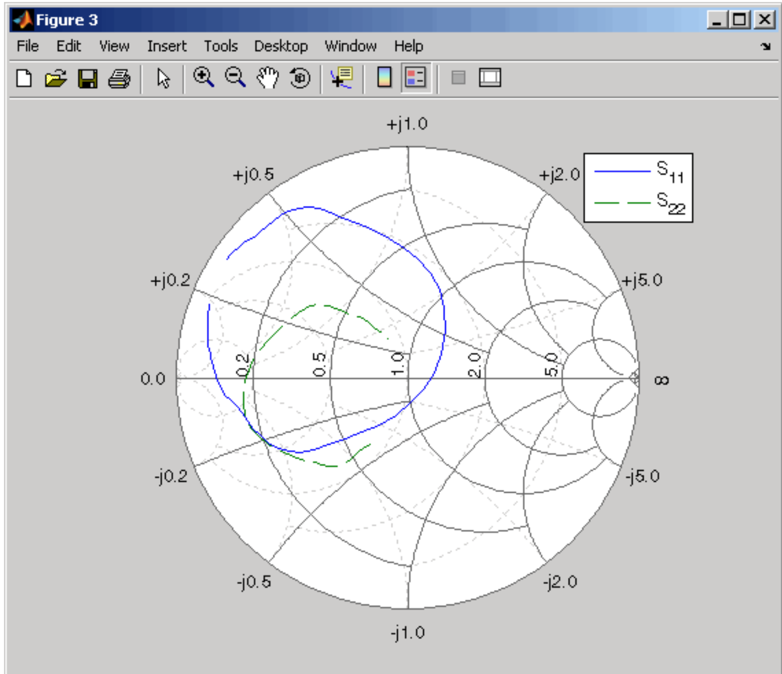
```
data =

            Name: 'Data object'
            Freq: [191x1 double]
    S_Parameters: [2x2x191 double]
              NF: [191x1 double]
```

```
        OIP3: [191x1 double]
          Z0: 50
          ZS: 50
          ZL: 50
    IntpType: 'linear'
```

You use these frequencies to analyze the cascaded circuit in a later step.

6 **Analyze the amplifier over a new frequency range and plot its new S11 and S22.** To visualize the S-parameters of a circuit over a different frequency range, you must first analyze the circuit over that frequency range.

```
analyze(SecondCkt,[1.85e9:1e7:2.55e9]);
figure
lineseries2 = smith(SecondCkt,'S11','S22','zy');
set(lineseries2(1),'LineStyle','-','LineWidth',1);
set(lineseries2(2),'LineStyle','--','LineWidth',1);
legend show
```

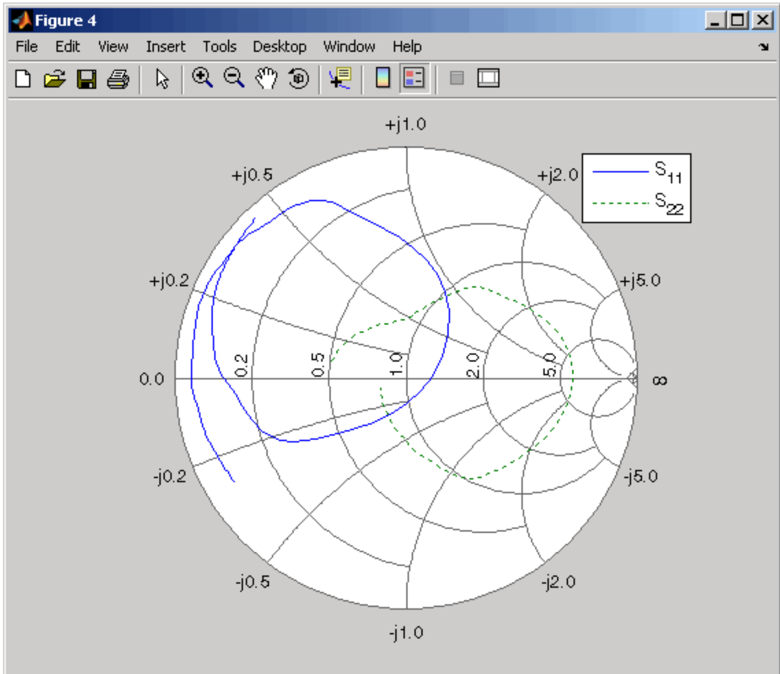


7 **Create and analyze a cascaded circuit object.** Cascade the three circuit objects to create a new cascaded circuit object, and then analyze it at the original amplifier frequencies

```
CascadedCkt = rfckt.cascade('Ckts',{FirstCkt,SecondCkt,...
                ThirdCkt});
analyze(CascadedCkt,f);
```

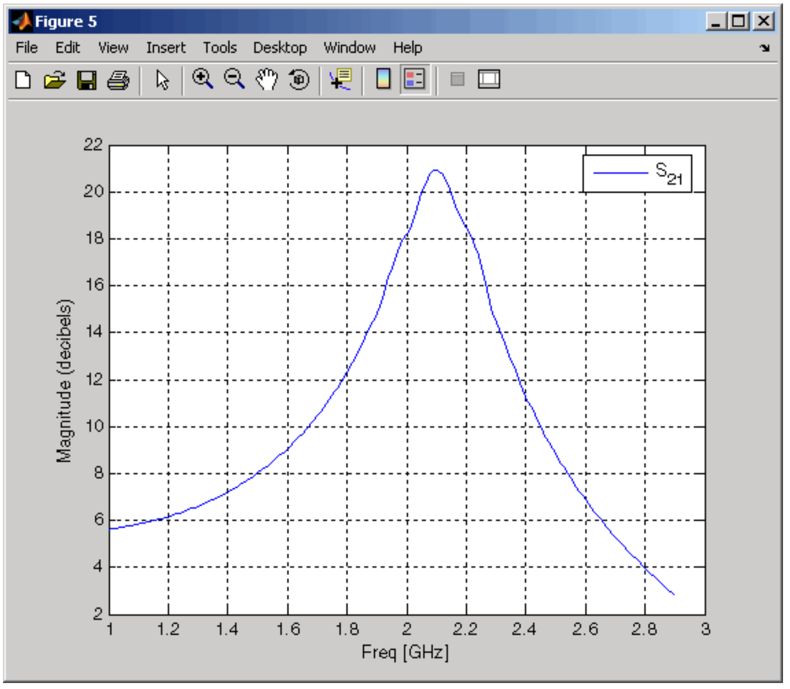
8 **Plot the S11 and S22 parameters of the cascaded circuit.** Use the `smith` command to plot the S11 and S22 parameters of the cascaded circuit (`CascadedCkt`) on a Z Smith chart.

```
figure
lineseries3 = smith(CascadedCkt,'S11','S22','z');
set(lineseries3(1),'LineStyle','-','LineWidth',1);
set(lineseries3(2),'LineStyle',':','LineWidth',1);
legend show
```



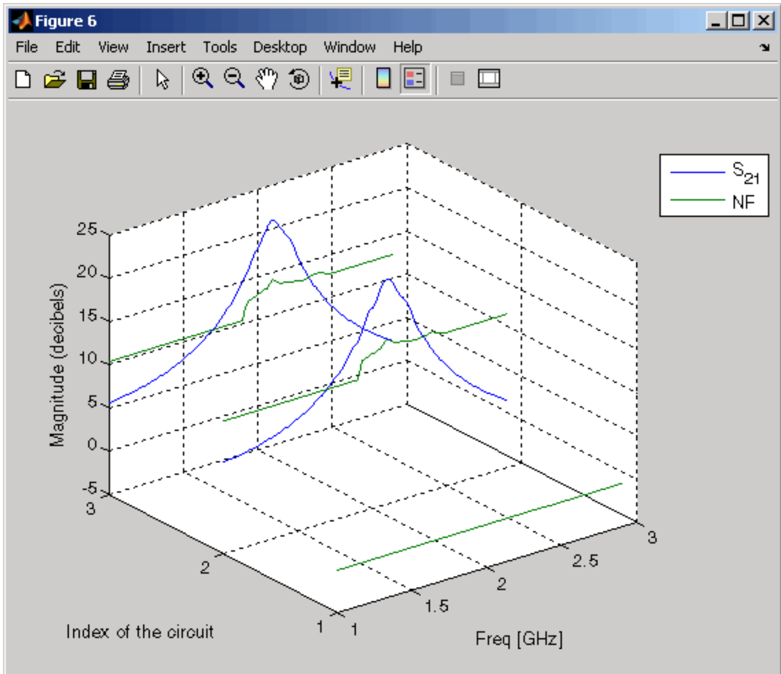
9 **Plot the S21 parameters of the cascaded circuit.** Use the RF Toolbox `plot` command to plot the S21 parameters of the cascaded circuit (`CascadedCkt`) on an X-Y plane.

```
figure
plot(CascadedCkt,'S21','dB');
legend show
```



10 Plot the budget S21 parameters and noise figure of the cascaded circuit. Use the RF Toolbox `plot` command to plot the budget S21 parameters and noise figure of the cascaded circuit (`CascadedCkt`) on an X-Y plane.

```
figure
plot(CascadedCkt,'budget', 'S21','NF');
legend show
```



RF Data Objects

These two examples demonstrate how to work with RF Toolbox data objects.

- "Read RF Data from a Touchstone Data File" on page 2-21
- "Setting Circuit Object Properties Using Data Objects" on page 2-24

Read RF Data from a Touchstone Data File

In this example, you create an `rfdata.data` object by reading the S-parameters of a two-port passive network stored in the Touchstone format data file, `passive.s2p`.

1 **Read S-parameter data from a data file.** Use the RF Toolbox `read` command to read the Touchstone data file, `passive.s2p`. This file contains 50 ohm S-parameters at frequencies ranging between 315 kHz and 6 GHz.

The `read` command creates an `rfdata.data` object, `data`, and stores data from the file in the object's properties.

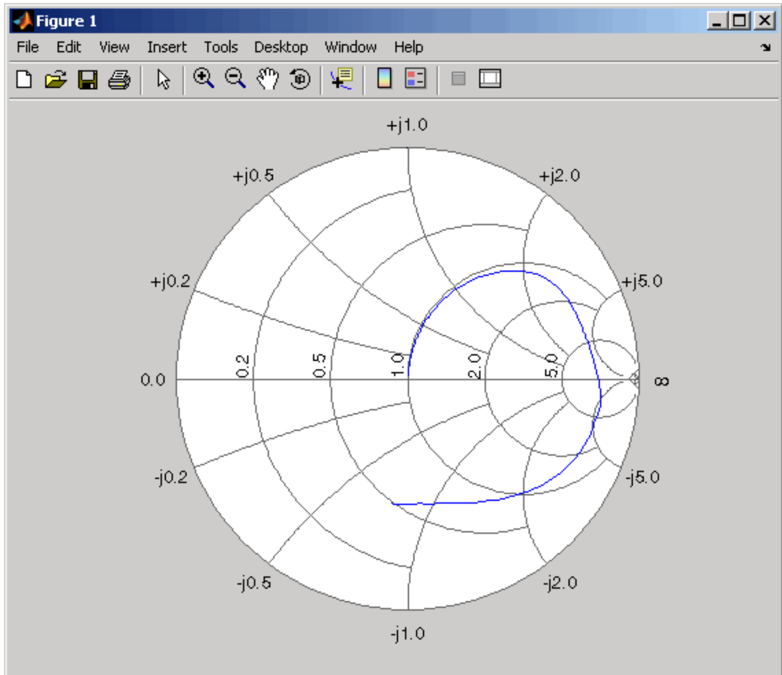
```
data = read(rfdata.data,'passive.s2p');
```

2 **Extract the network parameters from the data object.** Use the `extract` command to convert the 50 ohm S-parameters in the `rfdata.data` object, `data`, to 75 ohm S-parameters and save them in the variable `s_params`. You also use the command to extract the Y-parameters from the `rfdata.data` object and save them in the variable `y_params`.

```
freq = data.Freq;
s_params = extract(data,'S_PARAMETERS',75);
y_params = extract(data,'Y_PARAMETERS');
```

3 **Plot the S11 parameters.** Use the `smithchart` command to plot the 75 ohm S11 parameters on a Smith chart.

```
s11 = s_params(1,1,:);
smithchart(s11(:));
```



4 View the 75 ohm S-parameters and Y-parameters at 6 GHz. Use the following code to display the four 75 ohm S-parameter values and the four Y-parameter values at 6 GHz.

```
f = freq(end)
s = s_params(:,:,end)
y = y_params(:,:,end)
```

The toolbox displays the following output.

```
f =
  6.0000e+009

s =
  -0.0764 - 0.5401i   0.6087 - 0.3018i
   0.6094 - 0.3020i  -0.1211 - 0.5223i
```

```
y =
   0.0210 + 0.0252i  -0.0215 - 0.0184i
  -0.0215 - 0.0185i   0.0224 + 0.0266i
```

For more information, see the `rfdata.data`, `read`, and `extract` reference pages.

Setting Circuit Object Properties Using Data Objects

In this example, you create a circuit object, `rfckt.amplifier`. Then you create three data objects and use them to update the properties of the circuit object.

1 **Create an amplifier object.** This circuit object, `rfckt.amplifier`, has a network parameter, noise data, and nonlinear data properties. By default, these properties contain values from the `default.amp` file. The `NetworkData` property is an `rfdata.network` object that contains 50 ohm S-parameters. The `NoiseData` property is an `rfdata.noise` object that contains frequency-dependent spot noise data. The `NonlinearData` property is an `rfdata.power` object that contains output power and phase information.

   ```
   amp = rfckt.amplifier
   ```

   The toolbox displays the following output.

   ```
   amp =

                Name: 'Amplifier'
               nPort: 2
      AnalyzedResult: [1x1 rfdata.data]
            IntpType: 'linear'
         NetworkData: [1x1 rfdata.network]
           NoiseData: [1x1 rfdata.noise]
       NonlinearData: [1x1 rfdata.power]
   ```

2 **Create a data object that stores network data.** Use the following code to create an `rfdata.network` object that contains two-port Y-parameters at 2.08 GHz, 2.10 GHz, and 2.15 GHz. Later in this example, you use this data object to update the `NetworkData` property of the `rfckt.amplifier` object.

```
f = [2.08 2.10 2.15]*1.0e9;
y(:,:,1) = [-.0090-.0104i, .0013+.0018i; -.2947+.2961i
.0252+.0075i];
y(:,:,2) = [-.0086-.0047i, .0014+.0019i; -.3047+.3083i
.0251+.0086i];
y(:,:,3) = [-.0051+.0130i, .0017+.0020i; -.3335+.3861i
.0282+.0110i];

netdata =
rfdata.network('Type','Y_PARAMETERS','Freq',f,'Data',y)
```

The toolbox displays the following output.

```
netdata =

    Name: 'Network parameters'
    Type: 'Y_PARAMETERS'
    Freq: [3x1 double]
    Data: [2x2x3 double]
      Z0: 50
```

3 **Create a data object that stores noise figure values.** Use the following code to create a `rfdata.nf` object that contains noise figure values, in dB, at seven different frequencies. Later in this example, you use this data object to update the `NoiseData` property of the `rfckt.amplifier` object.

```
f = [1.93 2.06 2.08 2.10 2.15 2.30 2.40]*1.0e9;
nf = [12.4521 13.2466 13.6853 14.0612 13.4111 12.9499 13.3244];

nfdata = rfdata.nf('Freq',f,'Data',nf)
```

The toolbox displays the following output.

```
nfdata =

    Name: 'Noise figure'
    Freq: [7x1 double]
    Data: [7x1 double]
```

4 Create a data object that stores output third order intercept points. Use the following code to create a rfdata.ip3 object that contains output third-order intercept points, referenced to 8.45 watts, at 2.1 GHz. Later in this example, you use this data object to update the NonlinearData property of the rfckt.amplifier object.

```
ip3data = rfdata.ip3('Type','OIP3','Freq',2.1e9,'Data',8.45)
```

The toolbox displays the following output.

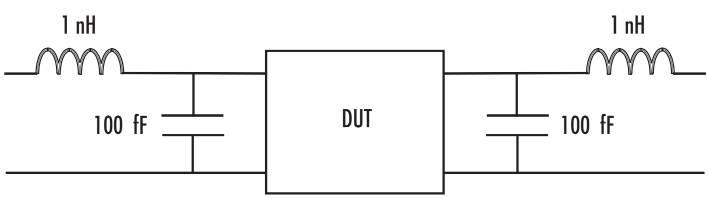
```
ip3data =

    Name: '3rd order intercept'
    Type: 'OIP3'
    Freq: 2.1000e+009
    Data: 8.4500
```

5 Update the properties of the amplifier object. Use the following code to update the NetworkData, NoiseData, and NonlinearData properties of the amplifier object with the data objects you created in the previous steps.

```
amp.NetworkData = netdata;
amp.NoiseData = nfdata;
amp.NonlinearData = ip3data;
```

De-embedding S-Parameters

The Touchstone data file samplebjt2.s2p contains S-parameter data collected from a bipolar transistor in a fixture with a bond wire connected to a bond pad on the input and a bond pad connected to a bond wire on the output. The configuration of the bipolar transistor, which is the device under test (DUT), and the fixture is shown in the following figure.



In this example, you remove the effects of the fixture and extract the S-parameters of the DUT.

1 Create RF objects. Create a data object for the measured S-parameters by reading the Touchstone data file samplebjt2.s2p. Then create two more circuit objects, one each for the input pad and output pad.

```
measured_data = read(rfdata.data,'samplebjt2.s2p');
input_pad = rfckt.cascade('Ckts',...
      {rfckt.seriesrlc('L',1e-9), ...
      rfckt.shuntrlc('C',100e-15)});    % L=1 nH, C=100 fF
output_pad = rfckt.cascade('Ckts',...
      {rfckt.shuntrlc('C',100e-15),...
      rfckt.seriesrlc('L',1e-9)});     % L=1 nH, C=100 fF
```

2 Analyze the input pad and output pad circuit objects. Analyze the circuit objects at the frequencies at which the S-parameters are measured.

```
freq = measured_data.Freq;
analyze(input_pad,freq);
analyze(output_pad,freq);
```

3 De-embed the S-parameters. Extract the S-parameters of the DUT from the measured S-parameters by removing the effects of the input and output pads.

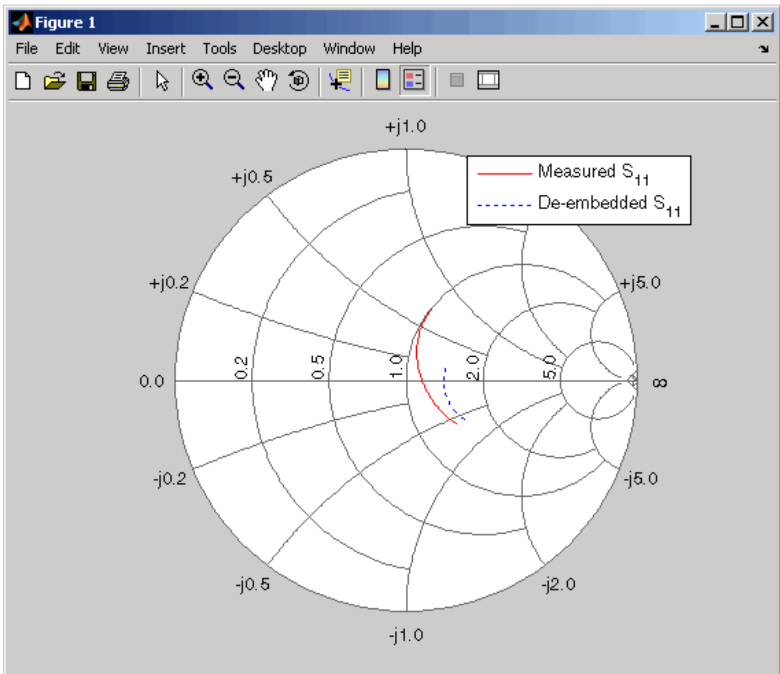
```
z0 = measured_data.Z0;

input_pad_sparams = extract(input_pad.AnalyzedResult,
'S_Parameters',z0);
output_pad_sparams = extract(output_pad.AnalyzedResult,
'S_Parameters',z0);

de_embedded_sparams =
deembedsparams(measured_data.S_Parameters,...
                      input_pad_sparams, output_pad_sparams);
```

4 Create a data object for the de-embedded S-parameters. In a later step, you use this data object to plot the de-embedded S-parameters.

```
de_embedded_data = rfdata.data('Z0',z0,...
                  'S_Parameters',de_embedded_sparams,...
                  'Freq',freq);
```

5 Plot the measured and de-embedded S11 parameters. Plot both the measured and the de-embedded S11 parameters on a Z Smith chart.

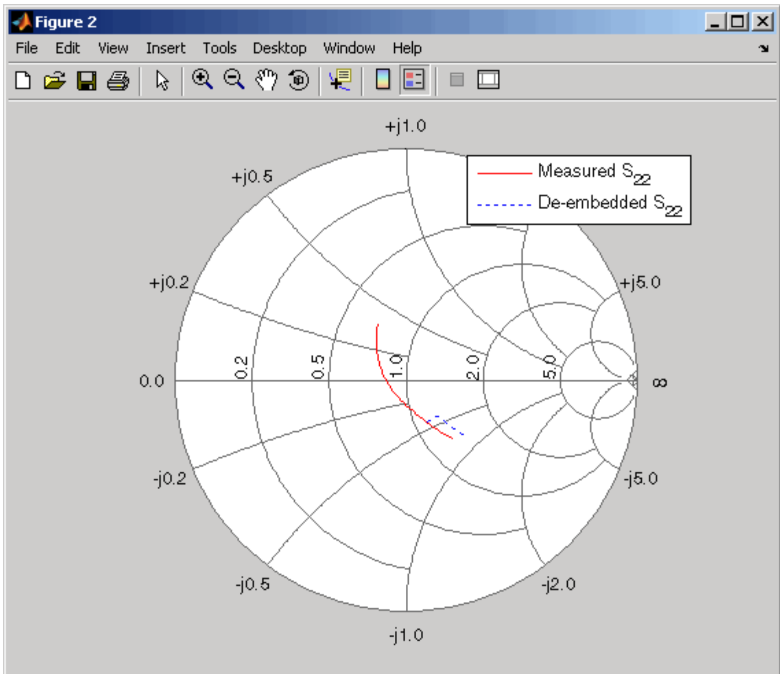
```
hold off;
h = smith(measured_data,'S11');
set(h, 'Color', [1 0 0]);
hold on
i = smith(de_embedded_data,'S11');
set(i,'Color', [0 0 1],'LineStyle',':');
l = legend;
legend(l, {'Measured S_{11}', 'De-embedded S_{11}'});
legend show;
```



6 Plot the measured and de-embedded S22 parameters. Plot the measured and the de-embedded S22 parameters on a Z Smith chart.

```
figure;
hold off;
```

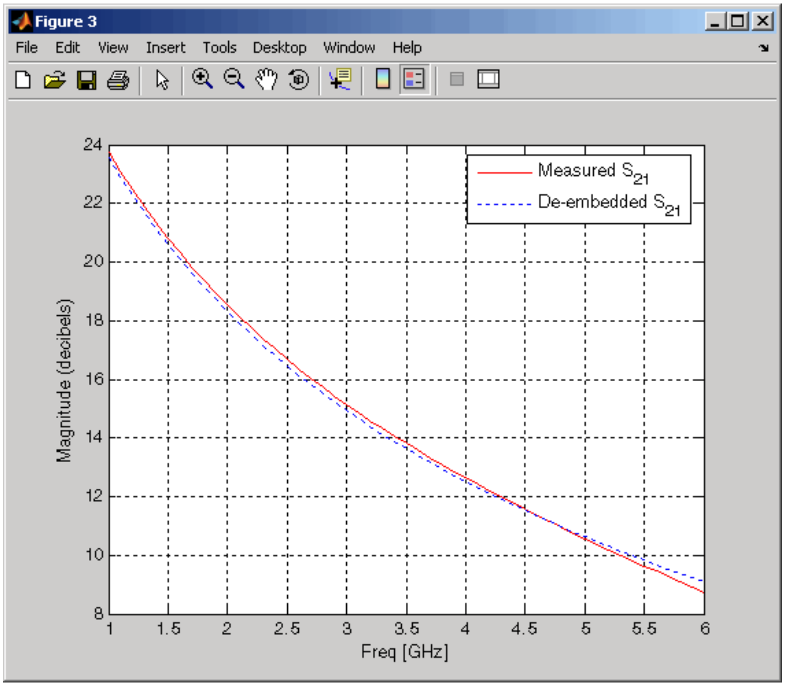
```
h = smith(measured_data,'S22');
set(h, 'Color', [1 0 0]);
hold on
i = smith(de_embedded_data,'S22');
set(i,'Color', [0 0 1],'LineStyle',':');
l = legend;
legend(l, {'Measured S_{22}', 'De-embedded S_{22}'});
legend show;
```



7 **Plot the measured and de-embedded S21 parameters.** Plot the measured and the de-embedded S21 parameters, in decibels, on an X-Y plane.

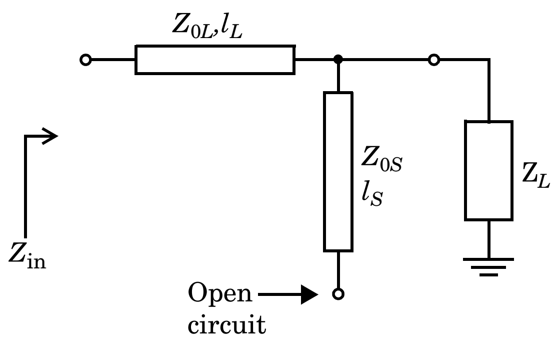
```
figure
hold off;
h = plot(measured_data,'S21', 'db');
set(h, 'Color', [1 0 0]);
hold on
```

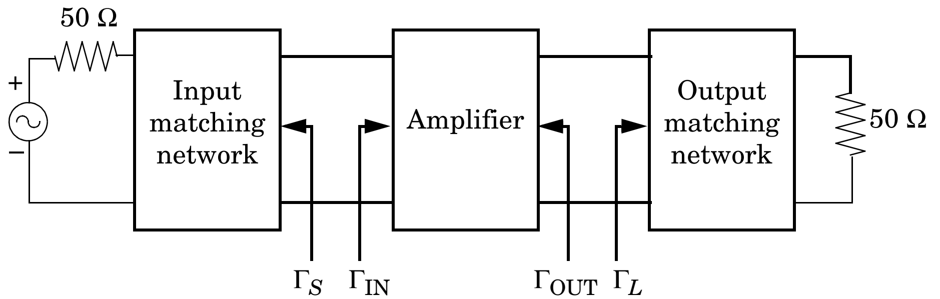
```
i = plot(de_embedded_data,'S21','db');
set(i,'Color', [0 0 1],'LineStyle',':');
l = legend;
legend(l, {'Measured S_{21}', 'De-embedded S_{21}'});
legend show;
hold off;
```

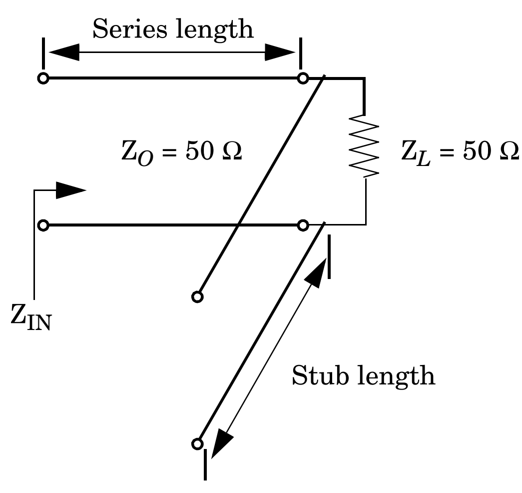


Impedance Matching

Input and output matching networks are an important part of amplifier design. In this example, you use a Smith chart to find the input and output matching networks that maximize the power delivered to a 50 ohm load. The single-stub network topology that consists of a series transmission line connected to a parallel combination of load and stub is shown in the following figure.


Z_{0L}, l_L
Z_{0S}
l_S
Z_L
Z_{in}
Open circuit


50 Ω
+
−
Input matching network
Amplifier
Output matching network
50 Ω
Γ_S
Γ_{IN}
Γ_{OUT}
Γ_L


Series length
Z_O = 50 Ω
Z_L = 50 Ω
Z_{IN}
Stub length

You begin by finding the required transmission line lengths for the single-stub matching networks. Then you cascade the matching networks with the amplifier and visualize the results.

1 **Create an amplifier object.** Create an amplifier object from the data in the file `samplebjt2.s2p`. Then analyze the amplifier at the center frequency of 1.9 GHz and get its S-parameters. For later convenience, you use the `deal` function to deal the various S-parameters into separate variables.

```
amp = rfckt.amplifier;
read(amp, 'samplebjt2.s2p');
analyze(amp, 1.9e9);
data = calculate(amp,'S11','S12','S21','S22','none');

[s11,s12,s21,s22] = deal(data{1},data{2},data{3},data{4});
```

2 **Check for amplifier stability.** For unconditional stability, K must be greater than 1 and the absolute value of delta must be less than 1. Use the following code to verify that the amplifier is stable.

```
delta = s11*s22-s12*s21;
K = (1-abs(s11)^2-abs(s22)^2+abs(delta)^2)/(2*abs(s12*s21))
abs_delta = abs(delta)
```

The toolbox displays the following output.

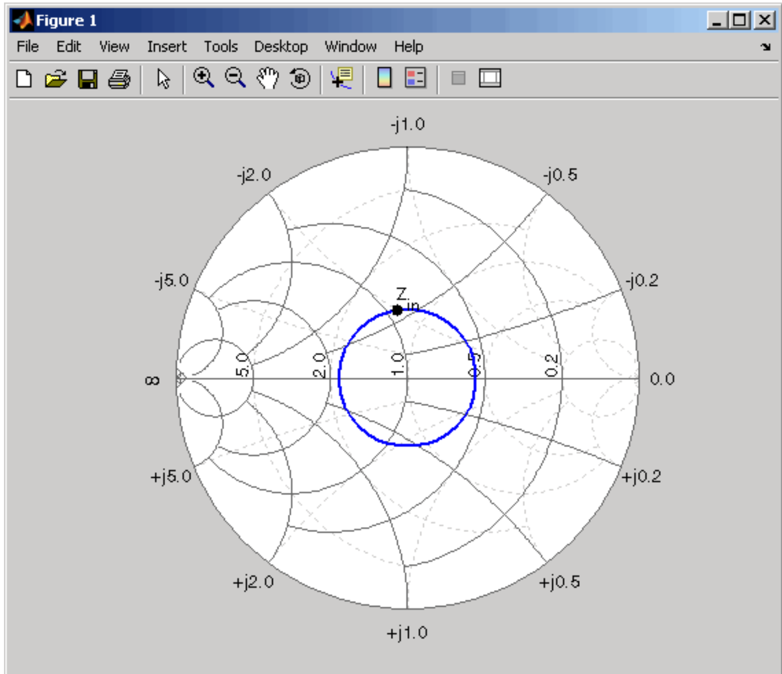
```
K =

    1.0599

abs_delta =

    0.6776
```

3 **Find the source and load reflection coefficients.** To design input and output matching networks, you must calculate the required source and load reflection coefficients that produce a simultaneous conjugate match. You can calculate the load reflection coefficient, `gammaL`, using the amplifier S-parameters.

```
B = 1+abs(s22)^2-abs(s11)^2-abs(delta)^2;
C = s22-delta*conj(s11);
gammaL = (B-sqrt(B^2-4*abs(C)^2))/2/C;
```

4 **Define the input standing wave ratio (SWR) circle associated with the load reflection coefficient.** The radius of this circle is given by the magnitude of the load reflection coefficient. You can use this radius (center is the origin) to calculate points on the SWR circle. Then you plot the desired input impedance point along with the input SWR circle on a ZY Smith chart.

```
theta = 0:pi/50:2*pi;
xin = abs(gammaL)*cos(theta);
yin = abs(gammaL)*sin(theta);

[hls, hs] = smithchart;
set(hs,'Type','yz');
hold on
plot(xin,yin,'-',real(gammaL),imag(gammaL),'k.',...
    'LineWidth',2,'MarkerSize',20);
text(-0.05, 0.35, 'z_{in}',...
    'FontSize',12,'FontUnits','normalized');
```



5 Draw the constant conductance circle. To find the required susceptance to move the 50 ohm load admittance to the SWR circle, you must define the constant conductance circle. To do this, you calculate the normalized load impedance and the corresponding 50 ohm load admittance for the transmission lines.

```
zL = 50/50; %zL = 1
yL = 1/zL; %yL = 1
```

Next you calculate the diameter and center of the circle using the conductance value.

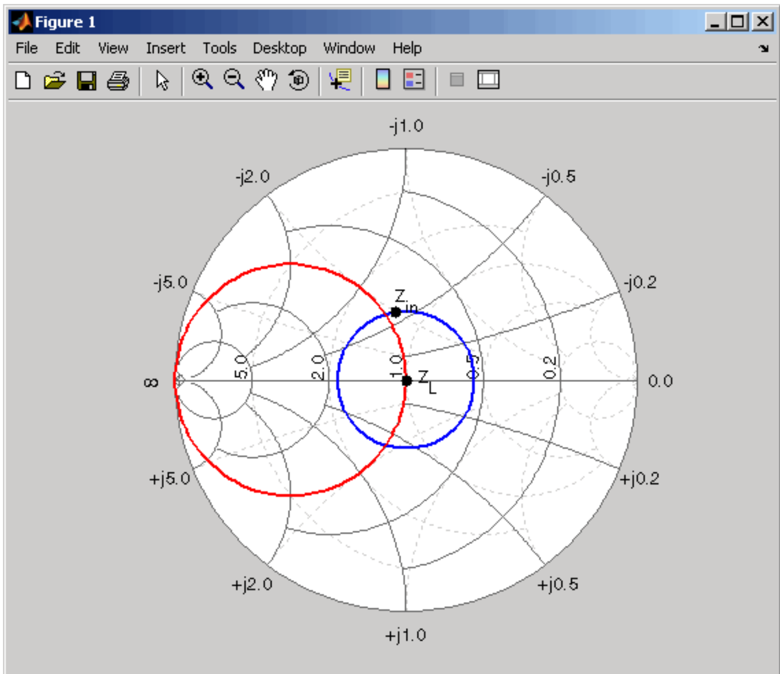
```
g  = real(yL); %g=1
d = -(g-1)/(g+1)+1; %d=1
C = -1+d/2; %C= 1/2
```

Then you use the radius and center of the constant conductance circle to calculate points on the circle.

```
xg = d/2*cos(theta)+C;
yg = d/2*sin(theta);
```

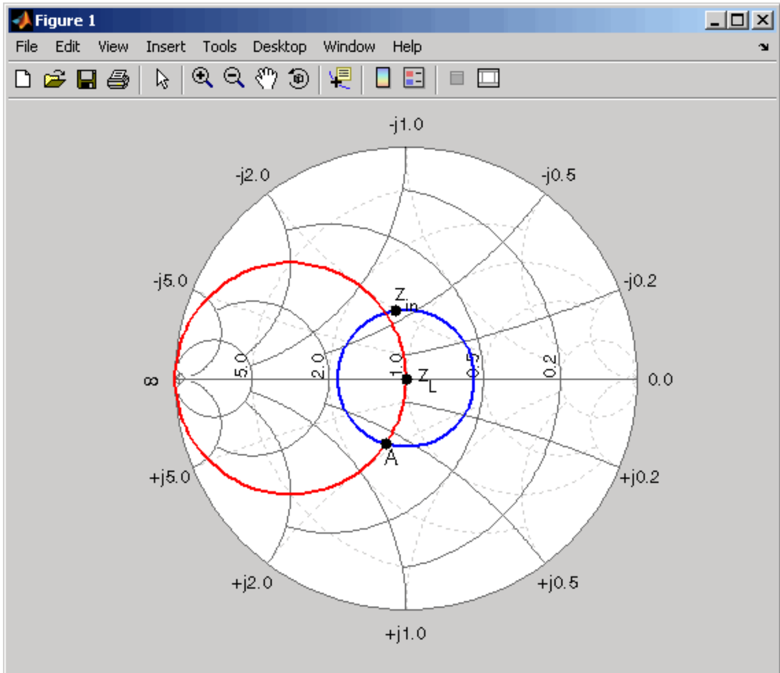
Finally, you plot and label the load impedance point along with the constant conductance circle associated with the load admittance on the Smith chart.

```
plot(xg, yg,'r',0,0,'k.','LineWidth',2,'MarkerSize',20);
text(0.05,0,'z_L','FontSize',12,'FontUnits','normalized');
```



6 **Find the intersection points.** Once you have drawn the input SWR and constant conductance circles, you can find the points of intersection that correspond to the two possible solutions. Since only one solution is necessary, choose the lower-half intersection point and designate this solution point A. Use the following code to plot and label this intersection point on the Smith chart using the reflection coefficient calculated from the admittance value.

```
yA = 1+0.62j;
gammaA = (1/yA-1)/(1/yA+1);
plot(real(gammaA),imag(gammaA),'k.','MarkerSize',20);
text(-0.09,-0.35,'A','FontSize',12,'FontUnits','normalized');
hold off
```



7 **Calculate the required lengths.** Based on the intersection point A, you can find the required lengths of the series transmission line and open-circuit stub. To accomplish this, first calculate the required susceptance value for the stub and its corresponding reflection coefficient.

```
jbSA = yA-yL;
gammaSA = (1/jbSA-1)/(1/jbSA+1);
```

Next you can find the stub length by calculating the angle of rotation from the `y = 0` (open-circuit) point to the calculated susceptance point.

```
ang = -angle(gammaSA)*180/pi;
stubLengthA = ang/360/2
```

Finally, you find the required length of the series transmission line based on the angle of rotation from point `A` to `Zin`.

```
seriesAngleA = 360-(angle(gammaL)-angle(gammaA))*180/pi;
seriesLengthA = seriesAngleA/360/2
```

The toolbox displays the following output, which represents the required lengths (in terms of wavelength) for the transmission lines based on the intersection point A.

```
stubLengthA =
    0.0883

seriesLengthA =
    0.2147
```

Using a similar approach, you can verify that the line lengths for the input matching network are

```
stubLengthin = 0.0763;
seriesLengthin = 0.2266;
```

8 **Verify the design.** Build the circuit using microstrip transmission lines, with a characteristic impedance of 50 ohms, for the matching networks. To accomplish this, analyze a microstrip object at 1.9 GHz.

```
hstubOutput = rfckt.microstrip;
analyze(hstubOutput,1.9e9);
Z0 = get(hstubOutput,'z0')
```

The toolbox displays the following output.

```
Z0 =
   50.2561
```

Because this characteristic impedance is close to the desired impedance, you can use it for the design.

To appropriately set the required transmission line lengths in meters, you must analyze the microstrip to get a phase velocity value, which is necessary to calculate the wavelength.

```
phase_vel = get(hstubOutput,'PV');
```

Set the appropriate transmission line lengths for the two series microstrip transmission lines necessary for the input and output matching networks.

```
hseriesOutput = rfckt.microstrip(...
    'LineLength',phase_vel/1.9e9*seriesLengthA);
hseriesInput = rfckt.microstrip(...
    'LineLength',phase_vel/1.9e9*seriesLengthin);
```

Similarly, set the transmission line lengths and the stub mode for the two stubs necessary for the input and output matching networks.

```
set(hstubOutput,'LineLength',phase_vel/1.9e9*stubLengthA,...
    'StubMode','shunt','Termination','open');
hstubInput = rfckt.microstrip(...
    'LineLength',phase_vel/2.1e9*stubLengthin,...
    'StubMode','shunt','Termination','open');
```

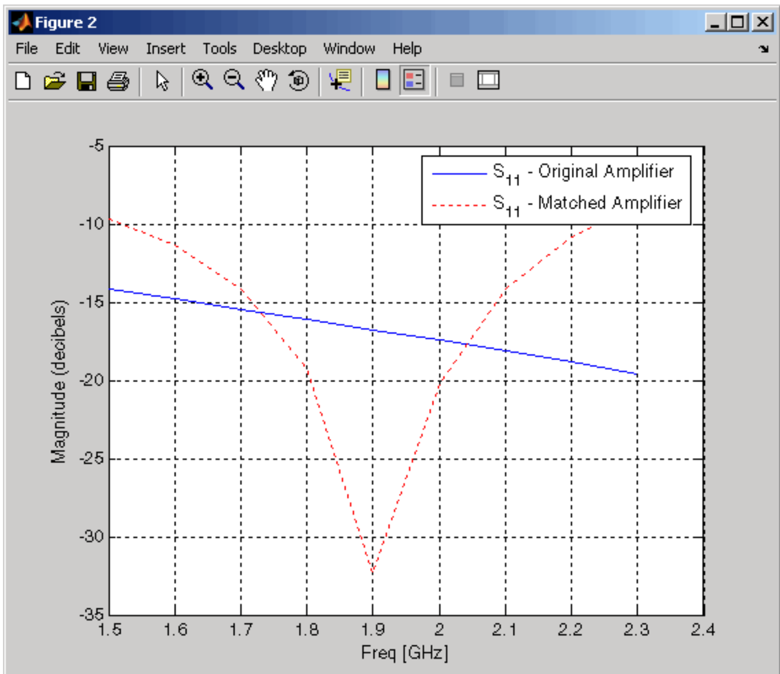
Then cascade the circuit elements and analyze the amplifier with and without the matching networks over the frequency range of 1.5 to 2.3 GHz to visualize and compare the results.

```
matched_amp = rfckt.cascade('Ckts',...
    {hstubInput,hseriesInput,amp,hseriesOutput,hstubOutput});
analyze(matched_amp,1.5e9:1e8:2.3e9);
analyze(amp,1.5e9:1e8:2.3e9);
```

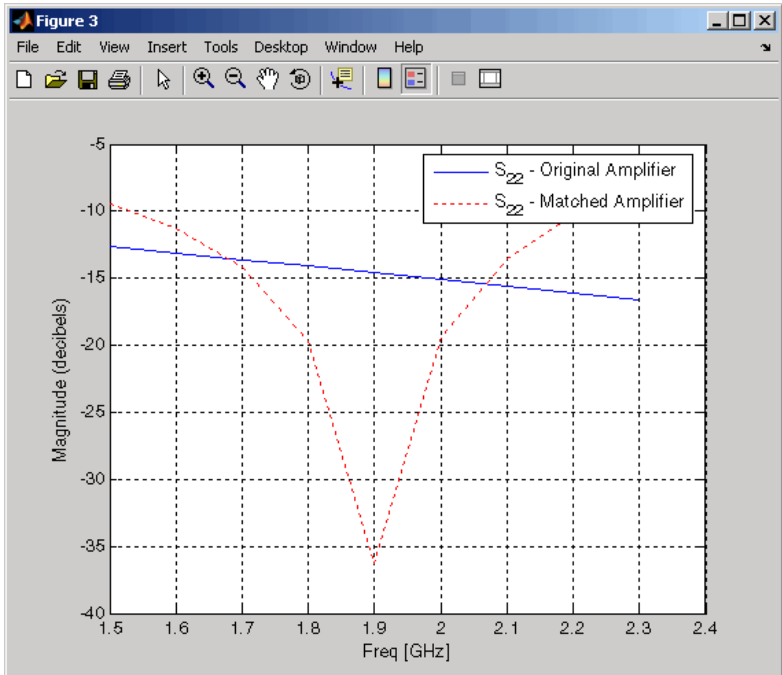
To verify the simultaneous conjugate match at the input and output of the amplifier, plot S11 parameters and S22 parameters, in decibels, for both circuits.

```
figure
```

```
hls = zeros(1,2);
hls(1) = plot(amp,'S11','dB');
hold on;
hls(2) = plot(matched_amp,'S11','dB');
set(hls(2),'Color',[1 0 0],'LineStyle',':');
legend(hls,'S_{11} - Original Amplifier',...
     'S_{11} - Matched Amplifier');
```

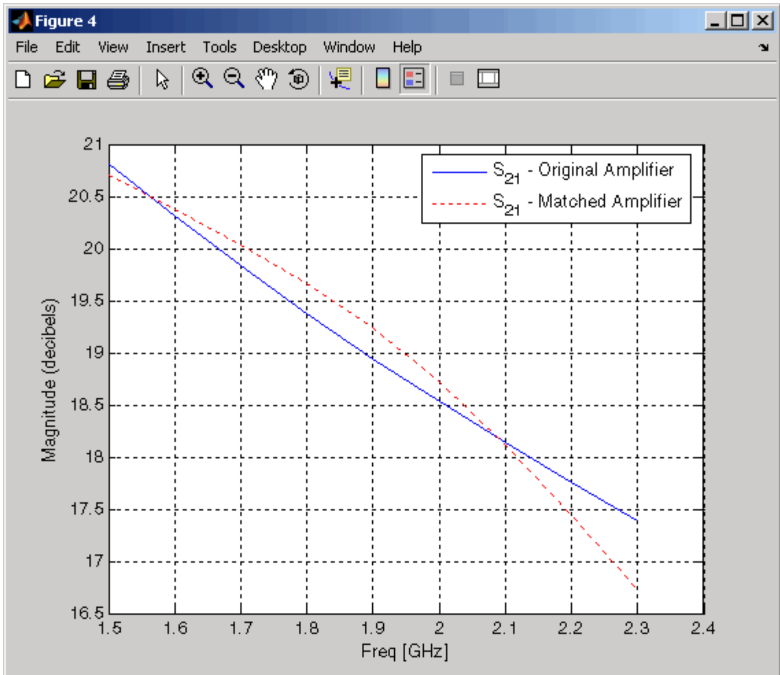



```
figure
hls(1) = plot(amp,'S22','dB');
hold on;
hls(2) = plot(matched_amp,'S22','dB');
set(hls(2),'Color',[1 0 0],'LineStyle',':');
legend(hls,'S_{22} - Original Amplifier',...
     'S_{22} - Matched Amplifier');
```



Finally, plot S21 parameters for both circuits.

```
figure
hls(1) = plot(amp,'S21','dB');
hold on;
hls(2) = plot(matched_amp,'S21','dB');
set(hls(2),'Color',[1 0 0],'LineStyle',':');
legend(hls,'S_{21} - Original Amplifier',...
     'S_{21} - Matched Amplifier');
hold off;
```



You can compare the matched amplifier results with the expected transducer gain (in dB). From the S21 parameters plot, you can see that the gain of the matched amplifier at 1.9 GHz is between 19 dB and 19.5 dB. The expected gain is given by the following equation:

```
Gt = 10*log10(abs(s21)/abs(s12)*(K-sqrt(K^2-1)))
```

The toolbox displays the following output.

```
Gt =
   19.2407
```

So, the matched amplifier's gain is very close to the expected transducer gain.

3

RF Tool: An RF Analysis GUI

Overview

The RF Tool is a GUI that provides a visual interface for creating and analyzing RF (radio frequency) components and networks. You can use the RF Tool as a convenient alternative to command line RF circuit design and analysis functions that come with the RF Toolbox.

RF Tool provides the ability to set circuit parameters, analyze the circuits, and display their S-parameters in both tabular and graphical form using X-Y plots, polar plots, and Smith charts. You can also import and export circuit data from the MATLAB workspace and RF data files.

RF Tool is available on supported UNIX and Windows platforms.

Sessions

The work you do with this tool is organized into sessions. Each session is a collection of independent RF circuits, which can be RF components or RF networks. You can save sessions and then load them for later use.

See “Working with Sessions” on page 3-22 for more information.

Available RF Circuits

The following tables lists the RF components and networks that you can create using RF Tool. These are the RF components:

RF Component	Corresponding RF Toolbox Function
Data File	`rfckt.datafile`
Coaxial Transmission Line	`rfckt.coaxial`
Coplanar Waveguide Transmission Line	`rfckt.cpw`
Microstrip Transmission Line	`rfckt.microstrip`
Parallel-Plate Transmission Line	`rfckt.parallelplate`
Transmission Line	`rfckt.txline`
Two-Wire Transmission Line	`rfckt.twowire`

RF Component	Corresponding RF Toolbox Function
Series RLC	`rfckt.seriesrlc`
Shunt RLC	`rfckt.shuntrlc`
LC Bandpass Pi	`rfckt.lcbandpasspi`
LC Bandpass Tee	`rfckt.lcbandpasstee`
LC Bandstop Pi	`rfckt.lcbandstoppi`
LC Bandstop Tee	`rfckt.lcbandstoptee`
LC Highpass Pi	`rfckt.lchighpasstee`
LC Highpass Tee	`rfckt.lchighpasstee`
LC Lowpass Pi	`rfckt.lclowpasspi`
LC Lowpass Tee	`rfckt.lclowpasstee`

The following table lists the available RF networks.

RF Network	Corresponding RF Toolbox Function
Cascaded Network	`rfckt.cascade`
Series Connected Network	`rfckt.series`
Parallel Connected Network	`rfckt.parallel`
Hybrid Connected Network	`rfckt.hybrid`

Getting Help

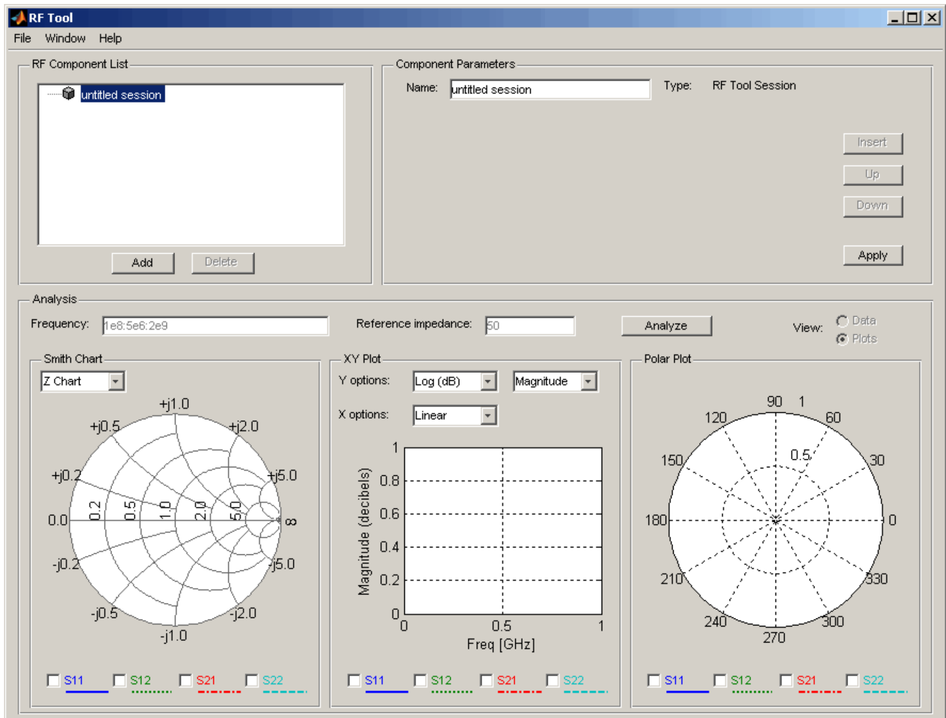
At any time, you can access the **Help** menu to see complete Help information on the RF Tool, RF Toolbox, and RF Demos.

Opening RF Tool

To open RF Tool, type

```
rftool
```

The RF Tool opens with a new untitled session.



To give the session a name:

1. Type the desired name in the **Name** field of the **Component Parameters** panel.
2. Click **Apply**.

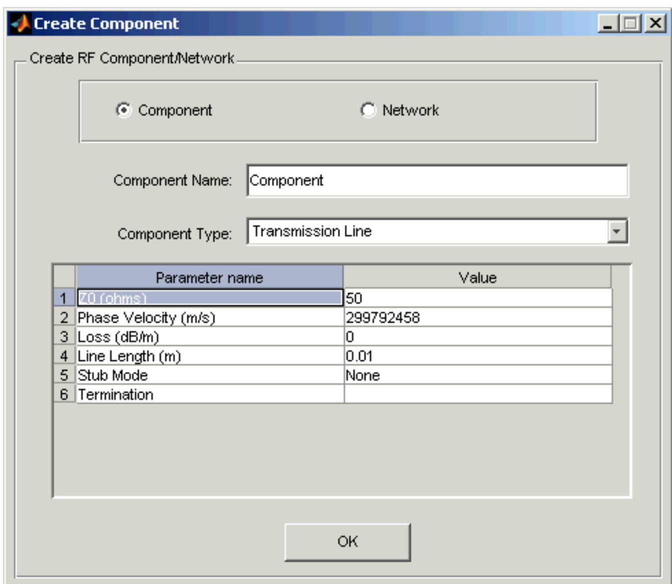
Creating Circuits

In the RF Tool, you can create circuits that include RF components and RF networks. Networks can contain both components and other networks. “Available RF Circuits” on page 3-2 lists the kinds of RF circuits you can create.

Topics in this section include:

Adding an RF Component to a Session

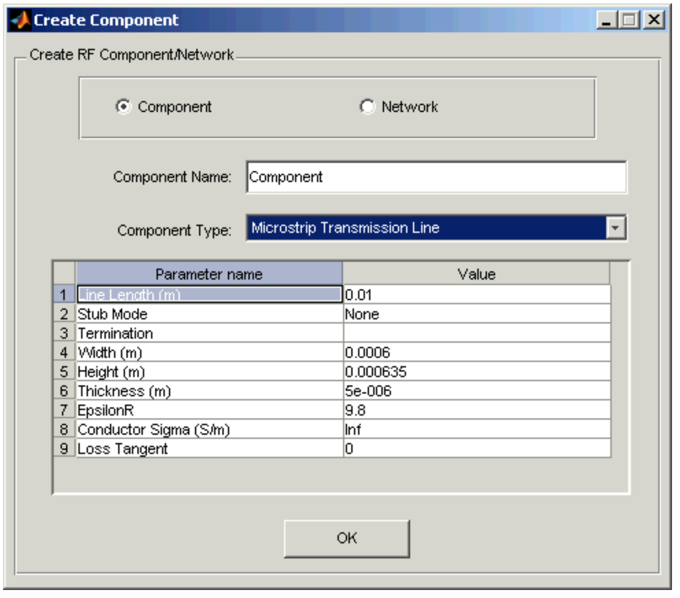
1 In the **RF Component List** of the RF Tool, click **Add** to open the **Create Component** dialog box.



2 In the **Create Component** dialog box, click the **Component** radio button if it is not already selected.

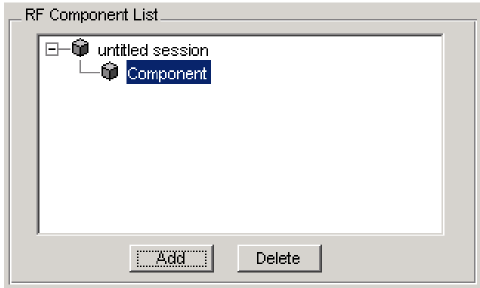
3 In the **Component Name** field, enter a name for the component.

4 From the **Component Type** pull-down menu, select the type of RF component you want to create. The RF Tool displays a list of that component's parameters in the **Create Component** dialog box. See "Available RF Circuits" on page 3-2 for a list of the available components.



5 Modify the parameters as necessary. See "Setting Component Parameters" on page 3-13 for more information.

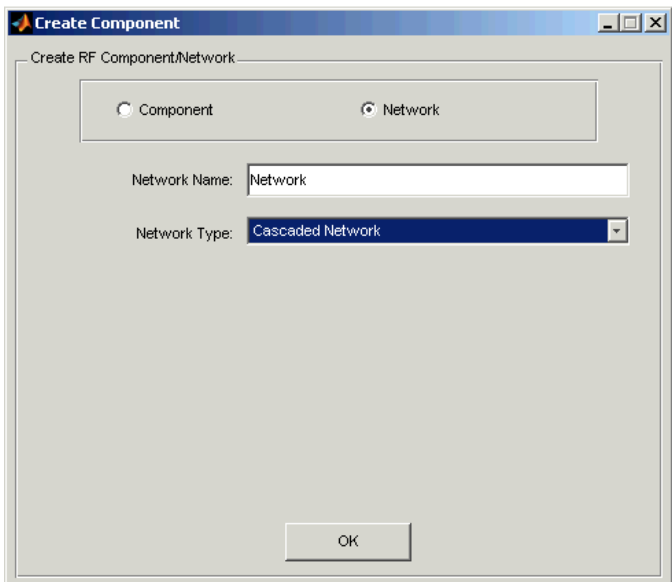
6 Click **OK**. The RF Tool adds the component to your session.



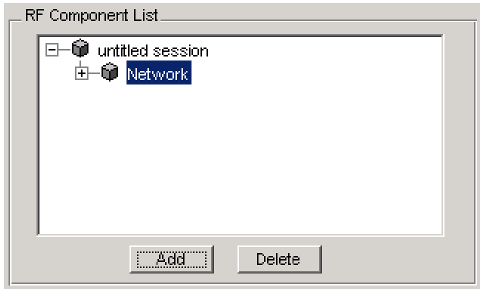
Adding an RF Network to a Session

To create a network, first add the network to your session, then populate the network by adding components and networks to it.

You add an RF network to a session in much the same way you add a component. However, to create a network, you must select the **Network** radio button. See "Adding an RF Component to a Session" on page 3-5 for details.



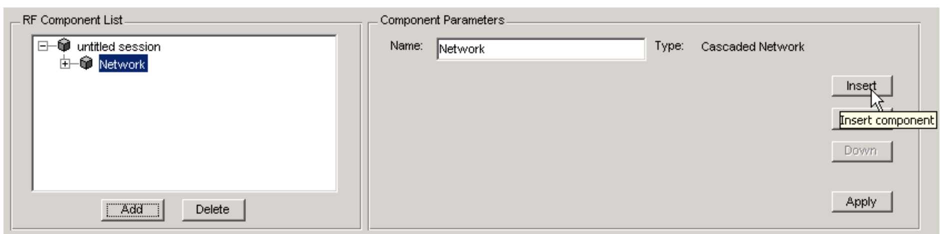
The RF Component List panel shows the new network.



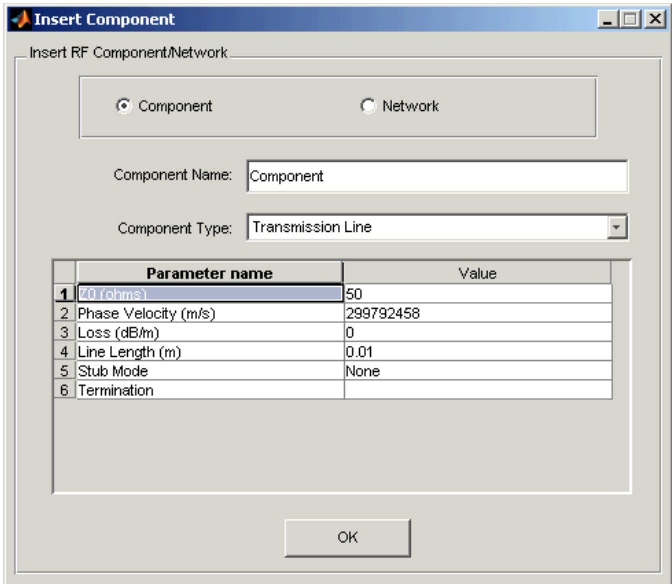
Populating an RF Network

After you create a network, you must populate it with RF components and networks.

1 In the **RF Component List** panel of RF Tool, select the network component you want to modify, and then click **Insert** in the **Component Parameters** panel.



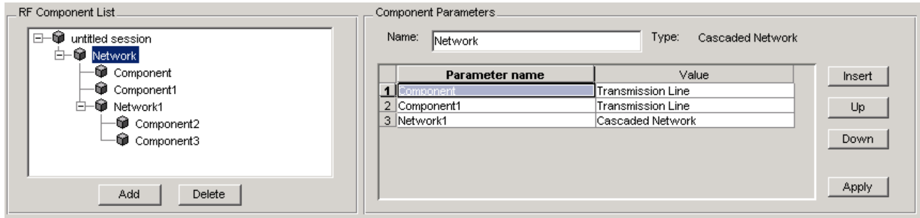
2 The **Insert Component** dialog box appears.



3 You insert a component or network in a network in much the same way you add one to a session.

In the **Insert Component** dialog box, start by selecting the **Component** or **Network** radio button as appropriate. Continue by giving the component or network a name, and selecting the appropriate type. If you are inserting a component, modify parameter values as necessary. See "Adding an RF Component to a Session" on page 3-5 or "Adding an RF Network to a Session" on page 3-7 for details.

As you insert components and networks into a network, they are reflected in the **RF Component List** and **Component Parameters** panels. This is an example of a cascaded network that contains two components and a network. The subnetwork, in turn, contains two components.



Reordering Circuits Within a Network

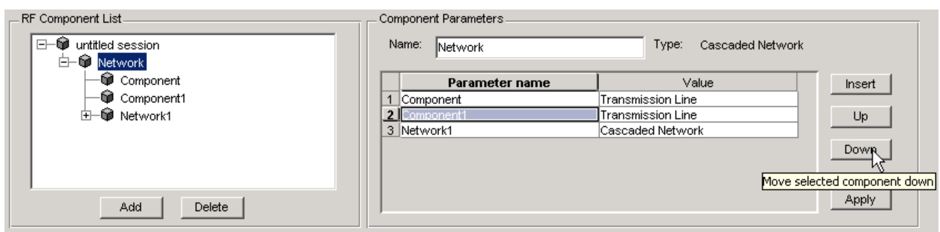
To change the order of the components and networks within a network:

1. In the **RF Component List** panel, select the network whose circuits you want to reorder.

2. In the **Component Parameters** panel, select the circuit whose position you want to change.

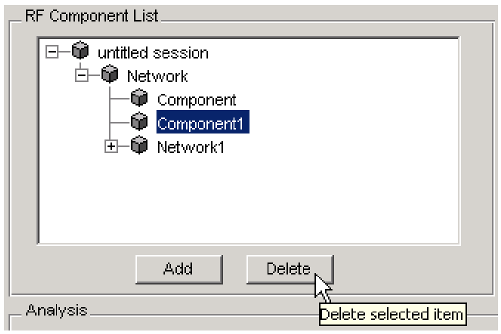
3. Click **Up** or **Down** until the circuit is where you want it.

In the example below, clicking **Down** after selecting `Network` in the **RF Component List** panel and selecting `Component1` in the **Component Parameters** panel, would reverse the positions of `Component1` and `Network1`.



Deleting Circuits

To delete a circuit from your session, select the circuit in the **RF Component List** panel and click **Delete**. If you select a network, the network and all its circuits are deleted.



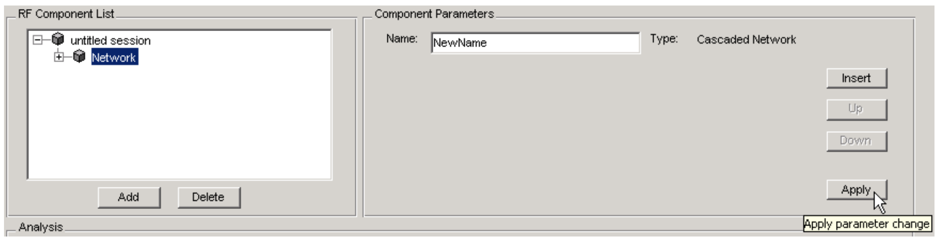
Renaming Circuits

To rename a component or a network:

1 Select the component or network in the **RF Component List** panel.

2 Type the new name in the **Name** field of the **Component Parameters** panel.

3 Click **Apply**.



To rename a session see "Renaming a Session" on page 3-23.

Setting Component Parameters

You can change the values of component parameters. To modify these values:

1. Select the component in the **RF Component List** panel.
2. In the **Component Parameters** panel, select the value you want to change, and enter the new value.

   Valid values for component parameters are listed on the corresponding RF Toolbox reference page. Use the links in "Available RF Circuits" on page 3-2 to access these pages. All values are case-insensitive.
3. Click **Apply**.

Analyzing Circuits

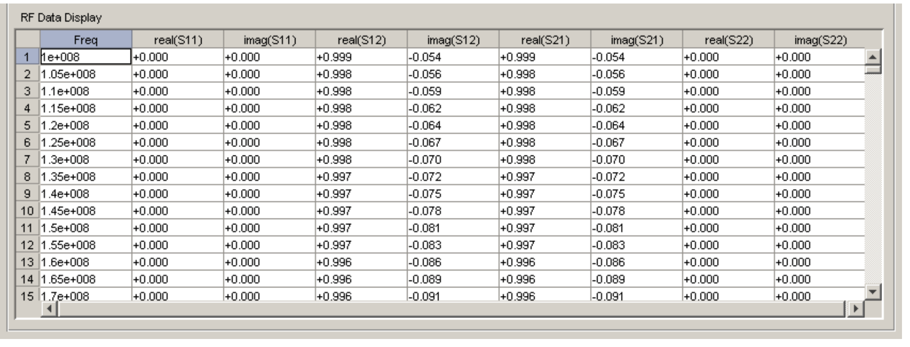
Once you have added your circuits, you can analyze them with the RF Tool:

1 Select the component or network you want to analyze in the **RF Component List** panel of the RF Tool.

2 In the **Analysis** panel, enter the analysis frequency range and step size in Hz in the **Frequency** field. Enter `50` in the **Reference impedance** field. You can specify these values as MATLAB workspace variables or as valid MATLAB expressions.

3 Click **Analyze**. This populates the data display panel with the component's S-parameter data as a function of the specified frequencies. To view the data, click on the **Plots** radio button.

The following figure shows the analysis data for a default Microstrip transmission line at the default frequencies and reference impedance shown in step 2.


RF Data Display

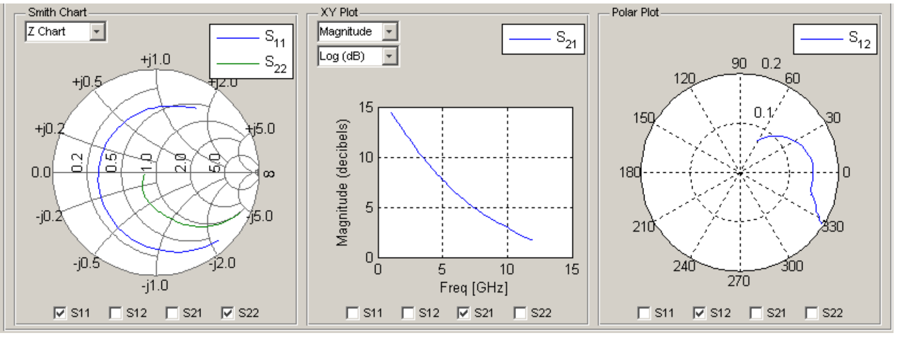
	Freq	real(S11)	imag(S11)	real(S12)	imag(S12)	real(S21)	imag(S21)	real(S22)	imag(S22)
1	1e+008	+0.000	+0.000	+0.999	-0.054	+0.999	-0.054	+0.000	+0.000
2	1.05e+008	+0.000	+0.000	+0.998	-0.056	+0.998	-0.056	+0.000	+0.000
3	1.1e+008	+0.000	+0.000	+0.998	-0.059	+0.998	-0.059	+0.000	+0.000
4	1.15e+008	+0.000	+0.000	+0.998	-0.062	+0.998	-0.062	+0.000	+0.000
5	1.2e+008	+0.000	+0.000	+0.998	-0.064	+0.998	-0.064	+0.000	+0.000
6	1.25e+008	+0.000	+0.000	+0.998	-0.067	+0.998	-0.067	+0.000	+0.000
7	1.3e+008	+0.000	+0.000	+0.998	-0.070	+0.998	-0.070	+0.000	+0.000
8	1.35e+008	+0.000	+0.000	+0.997	-0.072	+0.997	-0.072	+0.000	+0.000
9	1.4e+008	+0.000	+0.000	+0.997	-0.075	+0.997	-0.075	+0.000	+0.000
10	1.45e+008	+0.000	+0.000	+0.997	-0.078	+0.997	-0.078	+0.000	+0.000
11	1.5e+008	+0.000	+0.000	+0.997	-0.081	+0.997	-0.081	+0.000	+0.000
12	1.55e+008	+0.000	+0.000	+0.997	-0.083	+0.997	-0.083	+0.000	+0.000
13	1.6e+008	+0.000	+0.000	+0.996	-0.086	+0.996	-0.086	+0.000	+0.000
14	1.65e+008	+0.000	+0.000	+0.996	-0.089	+0.996	-0.089	+0.000	+0.000
15	1.7e+008	+0.000	+0.000	+0.996	-0.091	+0.996	-0.091	+0.000	+0.000

Plotting Circuit Data

After data is analyzed, setting the **View** radio button to **Plots** will display Smith, XY, and polar plots in the lower half of the RF Tool.

The plots will automatically update themselves as you change the check box and pull-down options on the GUI.

For example, loading the `samplebjt1.s2p` data file, clicking **Analyze**, and selecting **Plots** will display the following.



Importing RF Objects

RF Tool enables you to import RF objects from your workspace and from files to the top level of your session. You may want to import complex component and network objects that you created in your workspace using RF Toolbox functions. You may also want to import components and networks you exported into your workspace from another RF Tool session.

Once you have imported an object, you can change its name and work with it as you would any other component or network.

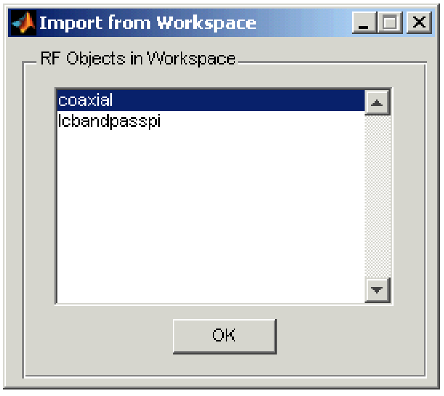
Topics in this section include:

- "Importing from the Workspace" on page 3-16
- "Importing From a File" on page 3-17

Importing from the Workspace

To import RF circuit objects from the MATLAB workspace into the top level of your session:

1 Select **Import From Workspace** from the **File** menu. The **Import from Workspace** dialog box appears. It lists the handles of all RF circuit (`rfckt`) objects in the workspace.



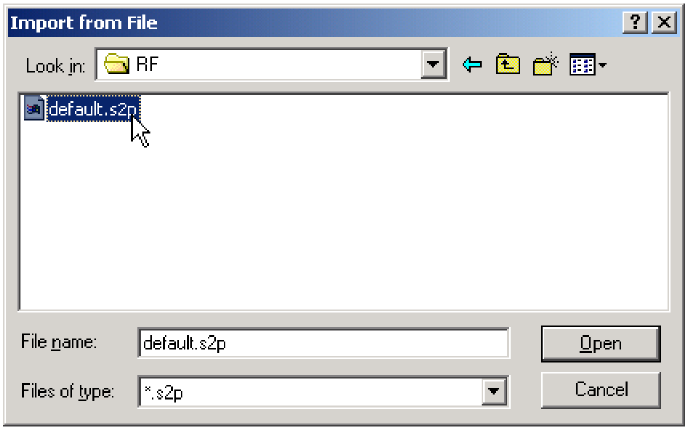
2 From the list of RF circuit objects, select the object you want to import, and click **OK**. The object is added to your session with the same name as the

object handle. If there is already a circuit by that name, RF Tool appends a numeral, starting with 1, to the new circuit name.

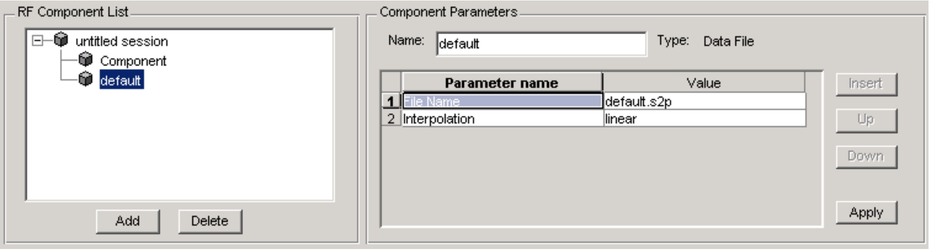
Importing From a File

You can import RF components from S2P, Y2P, Z2P, and H2P files. To import a component from one of these files:

1. Select **Import From File** from the **File** menu. A file browser appears.
2. Select the file type you want to import.
3. From the list of files in the browser, select the file to import.



4. Click **Open**. RF Tool adds the object to your session as a component.



The name of the component is the filename without the extension. If there is already a component by that name, RF Tool appends a numeral, starting with 1, to the new component name. The full filename appears as the value of the component's `File Name` parameter. If the file is not on the MATLAB path, the value of the `File Name` parameter also contains the file path.

Note To import an RF component from an S2P, Y2P, Z2P, or H2P file into a network, insert it in the network as a Data File component. See "Populating an RF Network" on page 3-8 for details.

Exporting RF Objects

RF Tool enables you to export RF components and networks that you have created and refined in RF Tool to your MATLAB workspace or to files. You might want to export circuits and then incorporate them into larger RF systems, or you may want to import them into another session.

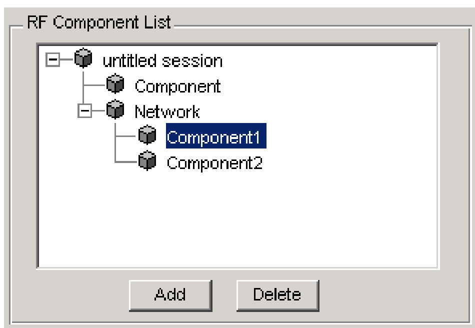
Topics in this section include:

- "Exporting to the Workspace" on page 3-19
- "Exporting to a File" on page 3-20

Exporting to the Workspace

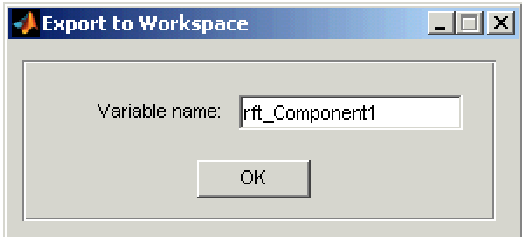
RF Tool enables you to export components and networks to the MATLAB workspace. Once in your workspace, you can use the resulting circuit (`rfckt`) object as you would any other RF circuit object.

1 Select the component or network to export in the **RF Component List** panel of the RF Tool.

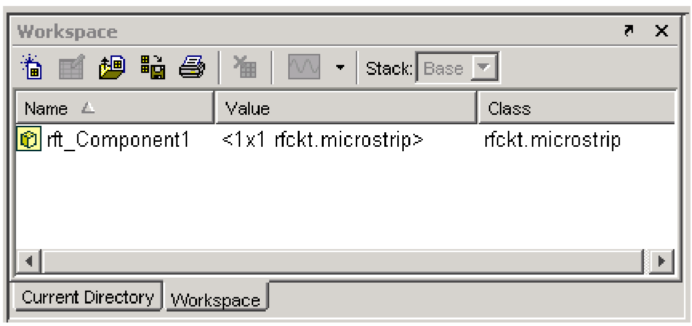


2 Select **Export to Workspace** from the **File** menu.

3 In the **Variable name** field, enter the name you want to give the exported object and click **OK**. The default name is the current name of the component or network prefaced with the string `'rft_'`.



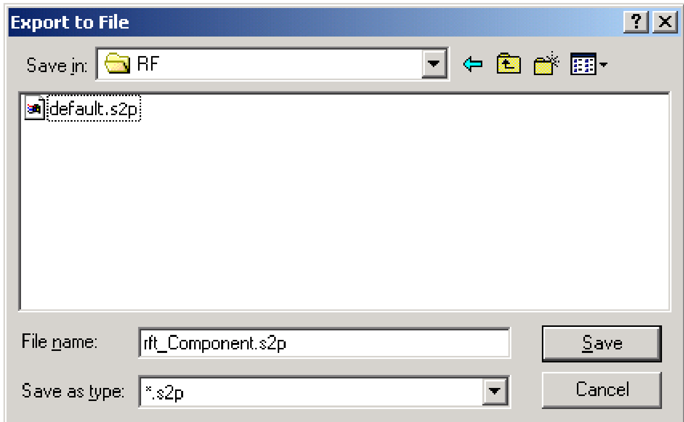
4 The component or network becomes accessible in the your workspace via its object handle `rft_Component1`.



Exporting to a File

RF Tool enables you to export components and networks to files in S2P format. Note that you must have analyzed a component or network in RF Tool before you can export it to a file. See "Analyzing Circuits" on page 3-14 for more information.

1 Select **Export To File** from the **File** menu. A file browser appears.



2 Browse to the appropriate directory. Enter the name you want to give the file and click **Save**.

The default filename is the current name of the component or network prefaced with the string `'rft_'`. RF Tool also converts any characters that are not alphanumeric to underscores (_).

Working with Sessions

The work you do with the RF Tool is organized into sessions. Each session is a collection of independent RF circuits, which can be RF components or RF networks.

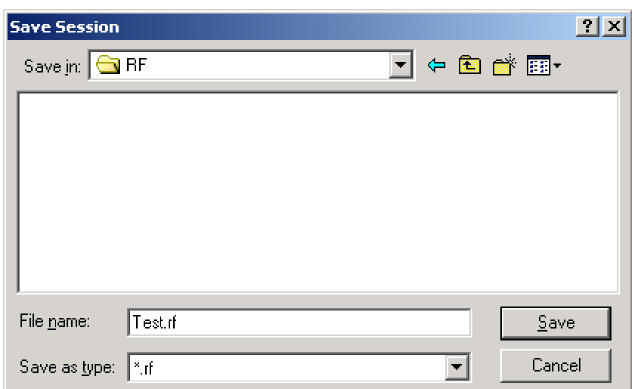
Topics in this section include:

- "Saving a Session" on page 3-22
- "Opening an Existing Session" on page 3-23
- "Renaming a Session" on page 3-23
- "Starting a New Session" on page 3-24

Saving a Session

To save your session, select **Save Session** or **Save Session As** from the **File** menu. The first time you save a session a browser opens, prompting you for a file name.

Note The default file name is the session name with any characters that are not alphanumeric converted to underscores (_). The name of the session itself is unchanged.

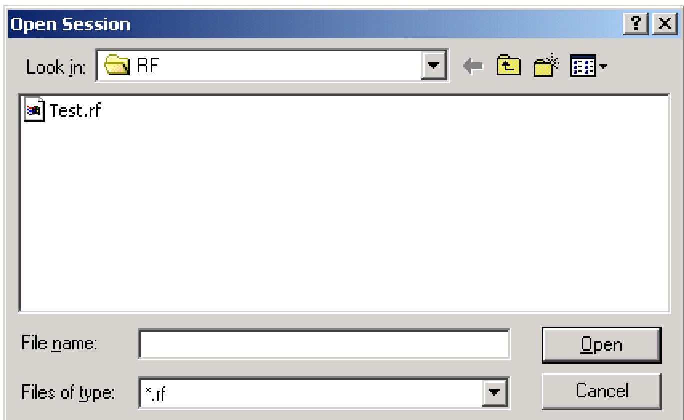


For example, to save your session as `Test.rf` in your current working directory, you would type `Test` in the **File name** field as shown above. RF Tool adds the `.rf` extension automatically to all RF Tool sessions you save.

If the name of your session is `gk's session`, the default file name is `gk_s_session.rf`.

Opening an Existing Session

You can load an existing session into the RF Tool by selecting **Open Session** from the **File** menu. A browser enables you to select from your previously saved sessions.



Before opening the requested session, RFTool prompts you to save your current session.

Renaming a Session

To rename a session:

1 Select the session in the **RF Component List** panel.

2 Type the desired name in the **Name** field of the **Component Parameters** panel.

3 Click **Apply**.

Starting a New Session

To start a new session, select **New Session** from the **File** menu. A new session opens in the RF Tool. All its values are set to their defaults.

Before starting a new session, RFTool prompts you to save your current session.

4

Function Reference

Functions — Categorical List

This section lists the RF Toolbox functions and objects according to their purpose.

Circuit Objects

`rfckt`	RF circuit object.
`rfckt.amplifier`	Amplifier, from a data file
`rfckt.cascade`	Cascaded network,
`rfckt.coaxial`	Coaxial transmission line
`rfckt.cpw`	Coplanar waveguide transmission line
`rfckt.datafile`	General circuit, from a data file
`rfckt.hybrid`	Hybrid connected network
`rfckt.lcbandpasspi`	LC bandpass pi network
`rfckt.lcbandpasstee`	LC bandpass tee network
`rfckt.lcbandstoppi`	LC bandstop pi network
`rfckt.lcbandstoptee`	LC bandstop tee network
`rfckt.lchighpasspi`	LC highpass pi network
`rfckt.lchighpasstee`	LC highpass tee network
`rfckt.lclowpasspi`	LC lowpass pi network
`rfckt.lclowpasstee`	LC lowpass tee network
`rfckt.microstrip`	Microstrip transmission line
`rfckt.mixer`	Mixer, from a data file
`rfckt.parallel`	Parallel connected network
`rfckt.parallelplate`	Parallel-plate transmission line
`rfckt.rlcgline`	Construct an RLCG transmission line object
`rfckt.series`	Series connected network
`rfckt.seriesrlc`	Series RLC network
`rfckt.shuntrlc`	Shunt RLC network
`rfckt.twowire`	Two-wire transmission line
`rfckt.txline`	General transmission line

Data Objects

`rfdata`	Data object.
`rfdata.data`	Network parameters data object.

Calculations

`analyze`	Calculate network parameters and noise figure for a circuit or data object at specified frequencies.
`calculate`	Calculate specified network parameters for a circuit or data object.
`cascadesparams`	Calculate cascaded S-parameters.
`deembedsparams`	De-embed S-parameters from a cascaded circuit.
`gammain`	Calculate GammaIn.
`gammaout`	Calculate GammaOut.
`vswr`	Calculate VSWR.

Data Visualization

`plot`	Plot network parameters from a circuit or data object on an X-Y plane.
`polar`	Plot network parameters from a circuit or data object on polar coordinates.
`smith`	Plot network parameters from a circuit or data object on a Smith chart.
`smithchart`	Plot a complex vector on a Smith chart.

Utility Functions

`copy`	Copy a circuit or data object.
`extract`	Extract specified network parameters from a data object and returns the result in a matrix.
`getdata`	Get data object containing analyzed data.

`listformat`	List valid formats for a specified network parameter for a specified circuit or data object.
`listparam`	List valid network parameters for a specified circuit or data object.

Data I/O

`read`	Read RF network parameters from a file to a new or existing data object.
`write`	Write RF data from a data object to a file.

Network Parameter Conversion

`abcd2h`	Convert ABCD-parameters to H-parameters.
`abcd2s`	Convert ABCD-parameters to S-parameters.
`abcd2y`	Convert ABCD-parameters to Y-parameters.
`abcd2z`	Convert ABCD-parameters to Z-parameters.
`h2abcd`	Convert H-parameters to ABCD-parameters.
`h2s`	Convert H-parameters to S-parameters.
`h2y`	Convert H-parameters to Y-parameters.
`h2z`	Convert H-parameters to Z-parameters.
`s2abcd`	Convert S-parameters to ABCD-parameters.
`s2h`	Convert S-parameters to H-parameters.
`s2s`	Convert S-parameters to S-parameters with different impedance.
`s2t`	Convert S-parameters to T-parameters.
`s2y`	Convert S-parameters to Y-parameters.
`s2z`	Convert S-parameters to Z-parameters.
`t2s`	Convert T-parameters to S-parameters.
`y2abcd`	Convert Y-parameters to ABCD-parameters.

`y2h`	Convert Y-parameters to H-parameters.
`y2s`	Convert Y-parameters to S-parameters.
`y2z`	Convert Y-parameters to Z-parameters.
`z2abcd`	Convert Z-parameters to ABCD-parameters.
`z2h`	Convert Z-parameters to H-parameters.
`z2s`	Convert Z-parameters to S-parameters.
`z2y`	Convert Z-parameters to Y-parameters.

Graphical User Interface

`rftool`	Visual interface for creating and analyzing RF circuits and networks.

Functions — Alphabetical List

This section contains function reference pages listed alphabetically.

abcd2h

Purpose Convert ABCD-parameters to hybrid h-parameters

Syntax

```
h_params = abcd2h(abcd_params)
```

Description `h_params = abcd2h(abcd_params)` converts the ABCD-parameters `abcd_params` into the hybrid parameters `h_params`. The `abcd_params` input is a complex 2-by-2-by-`m` array, representing `m` two-port ABCD-parameters. `h_params` is a complex 2-by-2-by-`m` array, representing `m` two-port hybrid h-parameters.

See Also `abcd2s`, `abcd2y`, `abcd2z`, `h2abcd`, `s2h`, `y2h`, `z2h`

Purpose Convert ABCD-parameters to S-parameters

Syntax

```
s_params = abcd2h(abcd_params,z0)
```

Description

s_params = abcd2h(abcd_params,z0) converts the ABCD-parameters abcd_params into the scattering parameters s_params. The abcd_params input is a complex 2-by-2-by-m array, representing m two-port ABCD-parameters. z0 is the reference impedance; its default is 50 ohms. s_params is a complex 2-by-2-by-m array, representing m two-port S-parameters.

See Also

abcd2h, abcd2y, abcd2z, s2abcd, s2h, y2h, z2h

abcd2y

Purpose Convert ABCD-parameters to Y-parameters

Syntax `y_params = abcd2y(abcd_params)`

Description `y_params = abcd2y(abcd_params)` converts the ABCD-parameters `abcd_params` into the admittance parameters `y_params`. The `abcd_params` input is a complex 2-by-2-by-`m` array, representing `m` two-port ABCD-parameters. `y_params` is a complex 2-by-2-by-`m` array, representing `m` two-port Y-parameters.

See Also `abcd2h`, `abcd2s`, `abcd2z`, `h2y`, `s2y`, `z2y`, `y2abcd`

Purpose Convert ABCD-parameters to Z-parameters

Syntax

```
z_params = abcd2z(abcd_params)
```

Description `z_params = abcd2z(abcd_params)` converts the ABCD-parameters `abcd_params` into the impedance parameters `z_params`. The `abcd_params` input is a complex 2-by-2-by-`m` array, representing `m` two-port ABCD-parameters. `z_params` is a complex 2-by-2-by-`m` array, representing `m` two-port Z-parameters.

See Also `abcd2h`, `abcd2s`, `abcd2y`, `h2y`, `s2z`, `y2z`, `z2abcd`

analyze

Purpose Analyze a circuit object in the frequency domain

Syntax

```
analyze(h,freq)
analyze(h,freq,zl,zs,zo)
```

Description `analyze(h,freq)` calculates the circuit network parameters and noise figure values at the specified frequencies. `h` is the handle of the circuit object to be analyzed. `freq` is a vector of frequencies, specified in Hz, at which the circuit is analyzed.

`analyze(h,freq,zl,zs,zo)` calculates the circuit network parameters and noise figure for the specified frequencies. The arguments `zl`, `zs`, and `zo` are optional. These arguments represent the circuit load, circuit source, and reference impedances of the S-parameters, respectively. The default value of all these arguments is 50 ohms.

Analysis of Circuit Objects

For most circuit objects, the `AnalyzedResult` property is empty until the `analyze` function is applied to the circuit object. However, these four circuit objects are the exception to this rule: `rfckt.datafile`, `rfckt.passive`, `rfckt.amplifier`, and `rfckt.mixer`.

By default, the `AnalyzedResult` property of `rfckt.datafile` objects contains the S-parameters, noise figure, and OIP3 values that are calculated over the network parameter frequencies in the `passive.s2p` data file.

By default, the `AnalyzedResult` property of `rfckt.passive` objects contains the S-parameters, noise figure, and OIP3 values that are the result of analyzing the values stored in the `passive.s2p` file at the frequencies stored in this file. These frequency values are also stored in the `NetworkData` property.

By default, the `AnalyzedResult` property of `rfckt.amplifier` objects contains the S-parameters, noise figure, and OIP3 values that are the result of analyzing the values stored in the `default.amp` file at the frequencies stored in this file. These frequency values are also stored in the `NetworkData` property.

By default, the `AnalyzedResult` property of `rfckt.mixer` objects contains the S-parameters, noise figure, and OIP3 values that are the result of analyzing the values stored in the `default.s2p` file at the frequencies stored in this file. These frequency values are also stored in the `NetworkData` property.

See Also `calculate`, `getz0`, `listformat`, `listparam`, `plot`, `polar`, `smith`, `read`, `restore`, `rfckt`, `rfdata`, `write`

calculate

Purpose Calculate specified parameters for a circuit object

Syntax

```
[data,params] = calculate(h,'parameter1',..., 'parametern',
   'format')
```

Description

`[data,params] = calculate(h,'parameter1',...,'parametern', 'format')` calculates the specified network parameters for the object `h` and returns them in the `n`-element cell array `data`. The input `h` is the handle of a circuit object. `parameter1,...,parametern` are the network parameters to be calculated. `format` is the format of the output `data`. Specify `format` as `'none'` to return the network parameters unchanged.

`params` is an n-element cell array containing the names, as strings, of the parameters in `data`.

Note Before calling `calculate`, you must use the `analyze` function to perform a frequency domain analysis for the circuit object.

For example, `[data,params] = calculate(h,'S11','S22','dB')` returns the S11 and S22 parameters in decibel format for the circuit object `h`.

Use the `listparam` and `listformat` functions to get lists of valid network parameters for a circuit object and the valid formats for a particular parameter.

Examples

Analyze a general transmission line, `trl`, with the default characteristic impedance of 50 ohms, phase velocity of 299792458 meters per second, and line length of 0.01 meters for frequencies of 1.0 GHz to 3.0 GHz. Then calculate S11 and S22 parameters in decibels.

```
trl = rfckt.txline;
f = [1e9:1.0e7:3e9];
analyze(trl,f);
[data,params] = calculate(trl,'S11','S22','dB')

data =
    [300x1 double]    [300x1 double]
```

```
params =
    'S_{11}'    'S_{22}'
```

The first few elements of `data{1}` look like

```
ans =

 -313.0712
 -312.5446
 -312.8039
 -312.8039
 -312.8039
 -312.8039
 -312.2928
 -312.8039
   ...
```

See Also

`analyze`, `getz0`, `listparam`, `listformat`, `plot`, `polar`, `smith`, `read`, `restore`, `rfckt`, `rfdata`, `write`

cascadesparams

Purpose Calculate the cascaded S-parameters

Syntax

```
s_params = cascadesparams(s1_params,s2_params,...,sn_params)
```

Description

`s_params = cascadesparams(s1_params,s2_params,...,sn_params)` calculates the scattering parameters, `s_params`, of the cascaded network.

Each of the input networks must be a two-port network described by a 2-by-2-by-`m` array of its S-parameters. All networks must have the same reference impedance.

`s_params` is a 2-by-2-by-`m` array containing the S-parameters of the resulting cascaded network.

See Also `t2s`, `s2t`, `deembedsparams`

Purpose Copy a circuit or data object

Syntax `h2 = copy(h)`

Description `h2 = copy(h)` returns a copy of the circuit or data object `h`.

Note The syntax `h2 = h` copies only the object handle and does not create a new object.

See Also `rfckt`, `rfdata`

deembedsparams

Purpose De-embed S-parameters from a cascaded network

Syntax

```
s2_params = deembedsparams(s_params,s1_params,s3_params)
```

Description

`s2_params = deembedsparams(s_params,s1_params,s3_params)` derives the `s2_params` from the cascaded S-parameters `s_params`, by removing the effects of `s1_params`, and `s3_params`.

Each of the input networks must be a two-port network described by a 2-by-2-by-`m` array of S-parameters. All networks must have the same reference impedance. `s_params` must contains the S-parameters of the cascaded network of `s1_params`, `s2_params`, and `s3_params`.

`s2_params` is a 2-by-2-by-`m` array. It contains the de-embedded S-parameters.

See Also `t2s`, `s2t`, `cascadesparams`

Purpose

Extract specified network parameters from a data object and return the result in an array

Syntax

```
outmatrix = extract(h,outtype)
```

Description

`outmatrix = extract(h,outtype)` extracts the network parameters of type `outtype` from an `rfdata.data` or `rfdata.network` object, `h`, and returns them in `outmatrix`.

`outtype` can be one of these case-insensitive strings `'ABCD_parameters'`, `'S_parameters'`, `'Y_parameters'`, `'Z_parameters'`, `'H_parameters'`, or `'T_parameters'`.

See Also

`analyze`, `calculate`, `getz0`, `listparam`, `listformat`, `plot`, `polar`, `smith`, `read`, `restore`, `rfckt`, `rfdata`, `write`

g2h

Purpose Convert hybrid g-parameters to hybrid h-parameters

Syntax

```
h_params = g2h(g_params,z0)
```

Description `h_params = g2h(g_params)` converts the hybrid g-parameters `g_params` into the hybrid h-parameters `h_params`. The `g_params` input is a complex 2-by-2-by-`m` array, representing `m` two-port g-parameters. `h_params` is a complex 2-by-2-by-`m` array, representing `m` two-port h-parameters.

See Also `h2g`

Purpose Calculates the input reflection coefficient of a two port network

Syntax

```
result = gammain(s_params,z0,zl)
```

Description

`result = gammain(s_params,z0,zl)` calculates the input reflection coefficient of a two port network as

$$\Gamma_{In} = S_{11} + \frac{(S_{12}*S_{21})*\Gamma_L}{1 - S_{22}*\Gamma_L}$$

where

$$\Gamma_L = \frac{Z_l - Z_0}{Z_l + Z_0}$$

`s_params` is a complex 2-by-2-by-`m` array, representing `m` two-port S-parameters. `z0` is the reference impedance Z_0; its default is 50 ohms. `zl` is the load impedance Z_l; its default is also 50 ohms. `result` is an `m`-element complex vector.

See Also `gammaout`

gammaout

Purpose Calculates the output reflection coefficient of a two port network

Syntax

```
result = gammaout(s_params,z0,zs)
```

Description `result = gammaout(s_params,z0,zs)` calculates the output reflection coefficient of a two port network as

$$\text{GammaOut} = S_{22} + \frac{(S_{12} * S_{21}) * \text{GammaS}}{1 - S_{11} * \text{GammaS}}$$

where

$$\text{GammaS} = \frac{\texttt{zs} - \texttt{z0}}{\texttt{zs} + \texttt{z0}}$$

`s_params` is a complex 2-by-2-by-`m` array, representing `m` two-port S-parameters. `z0` is the reference impedance; its default is 50 ohms. `zs` is the source impedance; its default is also 50 ohms. `result` is an `m`-element complex vector.

See Also `gammain`

Purpose Get data object containing analyzed result of a specified circuit object

Syntax

```
hd = getdata(h)
```

Description

`hd = getdata(h)` returns a handle `hd` to the `rfdata.data` object containing the analysis data, if any, for circuit (`rfckt`) object `h`. If the circuit object `h` has not been analyzed, i.e., there is no analysis data, `getdata` displays an error message.

Note For all circuit objects except those of type `rfckt.amplifier`, `rfckt.datafile`, and `rfckt.mixer`, before calling `getdata`, you must use the `analyze` function to perform a frequency domain analysis for the circuit (`rfckt`) object.

When you create an object of type `rfckt.amplifier`, `rfckt.datafile`, or `rfckt.mixer`, by reading data from a file, the RF Toolbox automatically creates an `rfdata.data` object and stores data from the file as properties of the data object. You can use the `getdata` function, without first calling `analyze`, to retrieve the handle of this data object.

See Also `rfckt`, `rfdata`

getz0

Purpose Get characteristic impedance of transmission line object

Syntax

```
z0 = getz0(h)
```

Description `z0 = getz0(h)` returns a scalar or vector, `z0`, that represents the characteristic impedance(s) of circuit object `h`. The object `h` can be `rfckt.txline`, `rfckt.rlcgline`, `rfckt.twowire`, `rfckt.parallelplate`, `rfckt.coaxial`, `rfckt.microstrip`, or `rfckt.cpw`.

See Also `analyze`, `calculate`, `listparam`, `listformat`, `plot`, `polar`, `smith`, `read`, `restore`, `rfckt`, `rfdata`, `write`

h2abcd

Purpose Convert hybrid h-parameters to ABCD-parameters

Syntax

```
abcd_params = h2abcd(h_params)
```

Description `abcd_params = h2abcd(h_params)` converts the hybrid parameters `h_params` into the ABCD-parameters `abcd_params`. The `h_params` input is a complex 2-by-2-by-`m` array, representing `m` two-port hybrid h-parameters. `abcd_params` is a complex 2-by-2-by-`m` array, representing `m` two-port ABCD-parameters.

See Also `abcd2h`, `h2s`, `h2y`, `h2z`, `s2abcd`, `y2abcd`, `z2abcd`

h2g

Purpose Convert hybrid h-parameters to hybrid g-parameters

Syntax

```
g_params = h2g(h_params,z0)
```

Description `g_params = h2g(h_params)` converts the hybrid parameters `h_params` into the hybrid g-parameters `g_params`. The `h_params` input is a complex 2-by-2-by-`m` array, representing `m` two-port h-parameters. `g_params` is a complex 2-by-2-by-`m` array, representing `m` two-port g-parameters.

See Also `g2h`, `h2abcd`, `h2s`, `h2y`, `h2z`

Purpose Convert hybrid h-parameters to S-parameters

Syntax

```
s_params = h2s(h_params,z0)
```

Description `s_params = h2s(h_params,z0)` converts the hybrid parameters `h_params` into the scattering parameters `abcd_params`. The `h_params` input is a complex 2-by-2-by-`m` array, representing `m` two-port hybrid h-parameters. `z0` is the reference impedance; its default is 50 ohms. `s_params` is a complex 2-by-2-by-`m` array, representing `m` two-port S-parameters.

See Also `abcd2s`, `h2abcd`, `h2y`, `h2z`, `s2h`, `y2s`, `z2s`

h2y

Purpose Convert hybrid h-parameters to Y-parameters

Syntax

```
y_params = h2y(h_params,z0)
```

Description `y_params = h2y(h_params)` converts the hybrid parameters `h_params` into the admittance parameters `y_params`. The `h_params` input is a complex 2-by-2-by-`m` array, representing `m` two-port hybrid h-parameters. `y_params` is a complex 2-by-2-by-`m` array, representing `m` two-port Y-parameters.

See Also `abcd2z`, `h2abcd`, `h2s`, `h2y`, `s2z`, `y2z`, `z2h`

Purpose Convert hybrid h-parameters to Z-parameters

Syntax

```
z_params = h2z(h_params)
```

Description `z_params = h2z(h_params)` converts the hybrid parameters `h_params` into the impedance parameters `z_params`. The `h_params` input is a complex 2-by-2-by-`m` array, representing `m` two-port hybrid h-parameters. `z_params` is a complex 2-by-2-by-`m` array, representing `m` two-port Z-parameters.

See Also `abcd2z`, `h2abcd`, `h2s`, `h2y`, `s2z`, `y2z`, `z2h`

listformat

Purpose List valid formats for a specified circuit object parameter

Syntax

```
list = listformat(h,'parameter')
```

Description

`list = listformat(h,'parameter')` lists the allowable formats for the specified network parameter. The first listed format is the default format for the specified parameter.

In these lists, `'Abs'` and `'Mag'` are the same as `'Magnitude (linear)'`, and `'Angle'` is the same as `'Angle (degrees)'`.

Use the `listparam` function to get the valid parameters of a circuit object.

Note Before calling `listformat`, you must use the `analyze` function to perform a frequency domain analysis for the circuit object.

Examples

```
trl = rfckt.txline;
f = [1e9:1.0e7:3e9];
analyze(trl,f);
list = listformat(trl,'S11')

list =
    'dB'
    'Magnitude (decibels)'
    'Abs'
    'Mag'
    'Magnitude (linear)'
    'Angle'
    'Angle (degrees)'
    'Angle (radians)'
    'Real'
    'Imag'
    'Imaginary'
```

See Also

`analyze`, `calculate`, `getz0`, `listparam`, `plot`, `polar`, `smith`, `read`, `restore`, `rfckt`, `rfdata`, `write`

Purpose

List valid parameters for a specified circuit object

Syntax

```
list = listparam(h)
```

Description

`list = listparam(h)` lists the valid parameters for the specified circuit object h.

Note Before calling `listparam`, you must use the `analyze` function to perform a frequency domain analysis for the circuit object.

Examples

```
trl = rfckt.txline;
f = [1e9:1.0e7:3e9];
analyze(trl,f);
list = listparam(trl)

list =
     'S11'
     'S12'
     'S21'
     'S22'
     'GAMMAIn'
     'GAMMAOut'
     'VSWRIn'
     'VSWROut'
     'OIP3'
     'NF'
```

See Also

`analyze`, `calculate`, `getz0`, `listformat`, `plot`, `polar`, `smith`, `read`, `restore`, `rfckt`, `rfdata`, `write`

plot

Purpose

Plot the specified circuit object parameters on an X-Y plane

Syntax

```
lineseries = plot(h,parameter)
lineseries = plot(h,parameter1,...,parametern)
lineseries = plot(h,parameter1,...,parametern,format)
lineseries = plot(h,'budget',...)
```

Description

`lineseries = plot(h,parameter)` plots the specified `parameter` on an X-Y plane in the default format. `h` is the handle of a circuit (`rfckt`) object.

Type `listparam(h)` to get a list of valid parameters for a circuit object, `h`. Type `listformat(h,parameter)` to see the legitimate formats for a specified `parameter`. The first listed format is the default for the specified parameter.

`The plot` function returns a column vector of handles to `lineseries` objects, one handle per line. This is the same as the output returned by the MATLAB `plot` function.

`lineseries = plot(h,parameter1,...,parametern)` plots the network parameters `parameter1,...`, `parametern` from the object `h` on an X-Y plane.

`lineseries = plot(h,parameter1,...,parametern,format)` plots the network parameters `parameter1,...`, `parametern` in the specified `format`. `format` is the format of the data to be plotted, e.g. `'Magnitude (decibels)'`.

`lineseries = plot(h,'budget',...)` plots budget data for the network parameters `parameter1,...`, `parametern` from the `rfckt.cascade` object `h`.

Use the Property Editor (`propertyeditor`) or the MATLAB `set` function to change `lineseries` properties. The reference pages for MATLAB functions such as `figure`, `axes`, and `text` also list available properties and provide links to more complete property descriptions.

Note For all circuit objects except those that contain data from a data file, you must perform a frequency domain analysis with the `analyze` function before calling `plot`.

Note Use the MATLAB plot function to plot network parameters that are specified as vector data and are not part of a circuit (rfckt) object or data (rfdata) object.

See Also

analyze, calculate, getz0, listparam, listformat, polar, smith, read, restore, rfckt, rfdata, write

polar

Purpose

Plot the specified circuit object parameters on polar coordinates

Syntax

```
lineseries = polar(h,parameter1,...,parametern,format)
```

Description

`lineseries = polar(h,parameter1,...,parametern,format)` plots the parameters `parameter1,...,` `parametern` from the object `h` on polar coordinates. `h` is the handle of a circuit (`rfckt`) object. `format` is the format of the data to be plotted, e.g. `'Magnitude (decibels)'`.

`polar` returns a column vector of handles to `lineseries` objects, one handle per line. This is the same as the output returned by the MATLAB `polar` function.

Use the Property Editor (`propertyeditor`) or the MATLAB `set` function to change the properties. The reference pages for MATLAB functions such as `figure`, `axes`, and `text` list available properties and provide links to more complete descriptions.

Type `listparam(h)` to get a list of valid parameters for a circuit object `h`. Type `listformat(h,parameter)` to see the legitimate formats for a specified `parameter`.

Note For all circuit objects except those that contain data from a data file, you must use the `analyze` function to perform a frequency domain analysis before calling `polar`.

Note Use the MATLAB `polar` function to plot parameters that are not part of a circuit (`rfckt`) object, but are specified as vector data.

See Also

`analyze`, `calculate`, `getz0`, `listparam`, `listformat`, `plot`, `smith`, `read`, `restore`, `rfckt`, `rfdata`, `write`

Purpose

Read RF data from file to new or existing circuit or data object

Syntax

```
h = read(h)
h = read(h,filename)
h = read(rfckt.datafile,filename)
h = read(rfckt.passive,filename)
h = read(rfckt.amplifier,filename)
h = read(rfckt.mixer,filename)
h = read(rfdata.data,filename)
```

Description

`h = read(h)` prompts you to select a `.snp`, `.ynp`, `.znp`, `.hnp`, or `.amp` file, where `n` is the number of ports. `read` then updates `h` with data from the file you select. Here, `h` can be a circuit or data object. See Appendix A, "AMP File Format" for information about the `.amp` format.

`h = read(h,filename)` updates `h` with data from the specified file. Here, `h` can be a circuit or data object. `filename` is a string, representing the filename of a `.snp`, `.ynp`, `.znp`, `.hnp`, or `.amp` file. The filename must include the file extension.

`h = read(rfckt.datafile,filename)` creates an `rfckt.datafile` object `h`, reads the RF data from the specified file, and stores it in `h`.

`h = read(rfckt.passive,filename)` creates an `rfckt.passive` object `h`, reads the RF data from the specified file, and stores it in `h`.

`h = read(rfckt.amplifier,filename)` creates an `rfckt.amplifier` object `h`, reads the RF data from the specified file, and stores it in `h`.

`h = read(rfckt.mixer,filename)` creates an `rfckt.mixer` object `h`, reads the RF data from the specified file, and stores it in `h`.

`h = read(rfdata.data,filename)` creates an `rfdata.data` object `h`, reads the RF data from the specified file, and stores it in `h`.

References

[1] EIA/IBIS Open Forum, "Touchstone File Format Specification," Rev. 1.1, 2002 (`http://www.eda.org/pub/ibis/connector/touchstone_spec11.pdf`).

See Also

analyze, calculate, getz0, listparam, listformat, plot, polar, smith, read, restore, rfckt, rfdata, write

Purpose Restore data to the original frequencies

Syntax

```
h = restore(h)
```

Description `h = restore(h)` restores data in `h` to the original frequencies of `NetworkData` for plotting. Here, `h` can be `rfckt.datafile`, `rfckt.passive`, `rfckt.amplifier`, or `rfckt.mixer`.

See Also `analyze`, `calculate`, `getz0`, `listparam`, `listformat`, `plot`, `polar`, `smith`, `read`, `rfckt`, `rfdata`, `write`

rfckt

Purpose Construct an RF circuit object

Syntax

```
h = rfckt.component('Property1',value1,...)
```

Description

`h = rfckt.component('Property1',value1,...)` returns a circuit object, `h`, of type *`component`*. See the individual `rfckt` component reference pages for information about a specific circuit object and its properties. See Chapter 2, "Working with RF Objects," for additional information.

Objects

The `component` for an `rfckt` object specifies the type of RF circuit object. The following table lists the available RF circuit objects.

rfckt.component	Description
`rfckt.amplifier`	Amplifier described by a data file
`rfckt.cascade`	Cascaded network
`rfckt.coaxial`	Coaxial transmission line
`rfckt.cpw`	Coplanar waveguide transmission line
`rfckt.datafile`	Circuit described by a data file
`rfckt.delay`	Delay line
`rfckt.hybrid`	Hybrid connected network
`rfckt.hybridg`	Inverse hybrid connected network
`rfckt.lcbandpasspi`	LC bandpass pi network
`rfckt.lcbandpasstee`	LC bandpass tee network
`rfckt.lcbandstoppi`	LC bandstop pi network
`rfckt.lcbandstoptee`	LC bandstop tee network
`rfckt.lchighpasspi`	LC highpass pi network
`rfckt.lchighpasstee`	LC highpass tee network
`rfckt.lclowpasspi`	LC lowpass pi network

rfckt.component	Description
rfckt.lclowpasstee	LC lowpass tee network
rfckt.microstrip	Microstrip transmission line
rfckt.mixer	Mixer described by a data file
rfckt.parallel	Parallel connected network
rfckt.parallelplate	Parallel-plate transmission line
rfckt.rlcgline	RLCG transmission line
rfckt.series	Series connected network
rfckt.seriesrlc	Series RLC network
rfckt.shuntrlc	Shunt RLC network
rfckt.twowire	Two-wire transmission line
rfckt.txline	General transmission line

Functions

The following table lists the functions that act on circuit objects and tells you the types of objects on which each can act. These functions are also referred to as methods.

Function	Types of Objects	Purpose
analyze	All circuit objects	Analyze a circuit object in the frequency domain.
calculate	All circuit objects	Calculate specified parameters for a circuit object
copy	All circuit objects	Copy a circuit or data object
getdata	All circuit objects	Get data object containing analyzed result of a specified circuit object

Function	Types of Objects	Purpose
`getz0`	`rfckt.txline`, `rfckt.rlcgline`, `rfckt.twowire`, `rfckt.parallelplate`, `rfckt.coaxial`, `rfdata.microstrip`, `rfckt.cpw`	Get characteristic impedance of a transmission line
`listformat`	All circuit objects	List valid formats for a specified circuit object parameter
`listparam`	All circuit objects	List valid parameters for a specified circuit object
`plot`	All circuit objects	Plot the specified circuit object parameters on an X-Y plane
`polar`	All circuit objects	Plot the specified circuit object parameters on polar coordinates
`read`	`rfckt.datafile`, `rfckt.passive`, `rfckt.amplifier`, `rfckt.mixer`	Read RF data from a file to a new or existing circuit object
`restore`	`rfckt.datafile`, `rfckt.passive`, `rfckt.amplifier`, `rfckt.mixer`	Restore data to original frequencies of `NetworkData` for plotting
`smith`	All circuit objects	Plot the specified circuit object parameters on a Smith chart
`write`	All circuit objects	Write RF data from a circuit object to a file

Properties

Properties vary for each type of component. See the individual component reference pages for information about properties.

Viewing Object Properties

You can use `get` to view an `rfckt` object's properties. To see a specific property of an object `h`, use

```
get(h,'PropertyName')
```

To see all properties for an object `h`, use

```
get(h)
```

Changing Object Properties

To see the properties of an object `h` whose values you can change use

```
set(h)
```

To change specific properties of object `h`, use

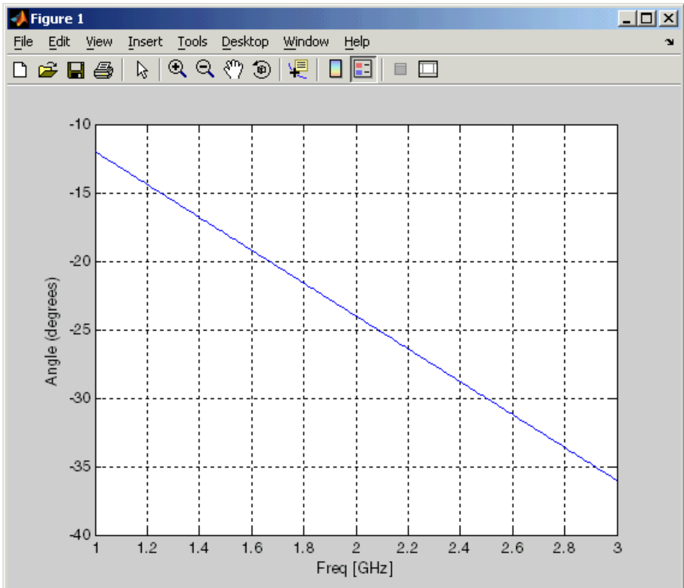
```
set(h,'PropertyName1',value1,'PropertyName2',value2,...)
```

Note that you must use single quotation marks around the property name.

Examples

Construct a general transmission line, `trl`, with the default characteristic impedance of 50 ohms, phase velocity of 299792458 meters per second, and line length of 0.01 meters. Then perform frequency domain analysis from 1.0 GHz to 3.0 GHz. Plot the resulting S21 network parameters, using the `'angle'` format, on the X-Y plane.

```
trl = rfckt.txline;
f = [1e9:1.0e7:3e9];      % Simulation frequencies
analyze(trl,f);           % Do frequency domain analysis
figure
plot(trl,'s21','angle');    % Plot magnitude of S21 in XY plane
```



You can also use other RF Toolbox functions such as `polar` and `smith` to visualize results.

See Also

`rfdata`

`analyze`, `calculate`, `copy`, `getdata`, `listformat`, `listparam`, `plot`, `polar`, `rfdata`, `smith`

Purpose Construct an amplifier object

Syntax

```
h = rfckt.amplifier
h = rfckt.amplifier('Property1',value1,'Property2',value2,...)
```

Description

`h = rfckt.amplifier` returns an amplifier circuit object whose properties all have their default values.

`h = rfckt.amplifier('Property1',value1,'Property2',value2,...)` returns a circuit object, `h`, based on the specified properties. Properties you do not specify retain their default values.

Use the `read` method to read the amplifier data from a Touchstone or AMP data file. See Appendix A, "AMP File Format" for information about the `.amp` format.

Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis

After you create the `rfckt.amplifier` circuit object, use the `analyze` function to calculate the S-parameters, output third-order intercept point, and noise figure at the specified frequencies. For `rfckt.amplifier` objects, `freq` must be nonnegative.

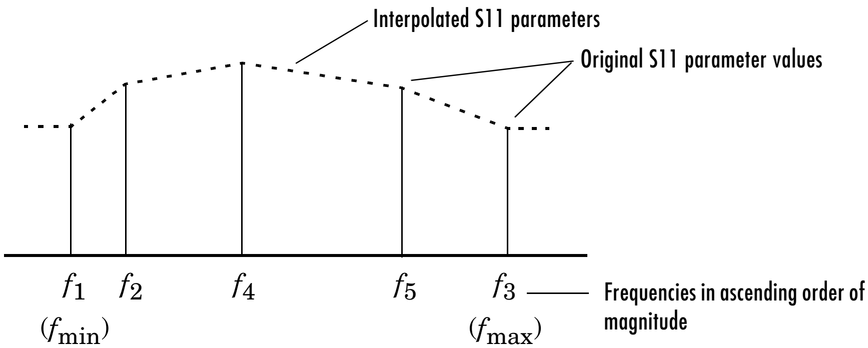
```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

If the `'NetworkData'` property of your `rfckt.amplifier` object contains network Y- or Z-parameters, the `analyze` function first converts the parameters to S-parameters. Using the interpolation method you specify with the `'IntpType'` property, the `analyze` function interpolates the S-parameter values to determine the S-parameters at the specified frequencies.

Specifically, the `analyze` function orders the S-parameters according to the ascending order of their frequencies, f_n. It then interpolates the S-parameters, using the MATLAB `interp1` function. For example, the curve in the following diagram illustrates the result of interpolating the S11 parameters at five different frequencies.



You can specify the interpolation method as `Cubic`, `Linear` (default), or `Spline`. For more information, see “One-Dimensional Interpolation” and the `interp1` reference page in the MATLAB documentation.

As shown in the diagram above, the `analyze` function uses the parameter values at f_{min}, the minimum input frequency, for all frequencies smaller than f_{min}. It uses the parameters values at f_{max}, the maximum input frequency, for all frequencies greater than f_{max}. In both cases, the results may not be accurate.

OIP3

The `analyze` function uses the data stored in the `'NonlinearData'` property of the `rfckt.amplifier` object to calculate OIP3.

Noise Figure

The `analyze` function uses the data stored in the `'NoiseData'` property of the `rfckt.amplifier` object to calculate the noise figure.

Properties

This table lists properties associated with `rfckt.amplifier` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the amplifier object	Handle. Default is `[1-by-1 rfdata.data]`.
`IntpType`	Interpolation method	`'Linear'` (default), `'Spline'`, or `'Cubic'`
`Name`	Object name (read only)	String. `'Amplifier'`
`NetworkData`	`rfdata.network` object	The default network parameters are taken from the `'default.amp'` data file.
`NoiseData`	Scalar noise figure in dB, `rfdata.noise` object or `rfdata.nf` object	The default noise data values are taken from the `'default.amp'` data file and stored in an `rfdata.noise` object.
`NonlinearData`	Scalar OIP3 in dBm, `rfdata.power` object or `rfdata.ip3` object	The default data values are taken from the `'default.amp'` data file and stored in an `rfdata.power` object.
`nPort`	Number of ports (read only)	Integer. The value is always 2.

References

[1] EIA/IBIS Open Forum, "Touchstone File Format Specification," Rev. 1.1, 2002 (`http://www.eda.org/pub/ibis/connector/touchstone_spec11.pdf`).

See Also

`analyze`, `calculate`, `listparam`, `listformat`, `plot`, `polar`, `read`, `restore`, `rfckt`, `rfckt.datafile`, `rfckt.mixer`, `rfckt.passive`, `rfdata`, `smith`, `write`

Purpose Construct cascaded network object

Syntax

```
h = rfckt.cascade
h = rfckt.cascade('Property1',value1,'Property2',value2,...)
```

Description `h = rfckt.cascade` returns a cascaded network object whose properties all have their default values.

`h = rfckt.cascade('Property1',value1,'Property2',value2,...)` returns a cascaded network object, `h`, based on the specified properties. Use the `'Ckts'` property to specify the `rfckt` objects to be cascaded. Properties you do not specify retain their default values.

Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis After you create the `cascade` network object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.cascade` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

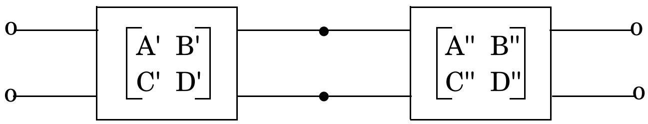
The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

The `analyze` function first calculates the ABCD parameters of the cascaded network. It starts by converting each component network's parameters to an ABCD parameters matrix. The figure shows a cascaded network consisting of two 2-port networks, each represented by its ABCD matrix.

The `analyze` function then calculates the ABCD parameter matrix for the cascaded network by calculating the product of the ABCD matrices of the individual networks.

The figure shows a cascaded network consisting of two 2-port networks, each represented by its ABCD-parameters.



The following equation illustrates calculations of the ABCD-parameters for two 2-port networks.

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} A' & B' \\ C' & D' \end{bmatrix} \begin{bmatrix} A'' & B'' \\ C'' & D'' \end{bmatrix}$$

Finally, `analyze` converts the ABCD parameters of the cascaded network to S-parameters at the frequencies specified in the `analyze` input argument `freq`.

OIP3

The `analyze` function calculates the output power at the third-order intercept point (OIP3) for an N-element cascade using the following equation

$$OIP_3 = \frac{1}{\frac{1}{OIP_{3,N}} + \frac{1}{(G_N \cdot OIP_{3,N-1})} + \ldots + \frac{1}{(G_N \cdot G_{N-1} \cdot \ldots \cdot G_2 \cdot OIP_{3,1})}}$$

where G_n is the gain of the nth element of the cascade and $OIP_{3,n}$ is the OPI3 of the nth element.

Noise Figure

The `analyze` function calculates the noise figure for an N-element cascade using the following equation

$$NF = NF_1 + \frac{NF_2 - 1}{G_1} + \frac{NF_3 - 1}{G_1 \cdot G_2} + \ldots + \frac{NF_N - 1}{G_1 \cdot G_2 \cdot \ldots \cdot G_{N-1}}$$

where G_n is the gain of the nth element of the cascade and NF_n is the noise figure of the nth element.

Properties

This table lists properties associated with `rfckt.cascade` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the cascaded network object	Handle. Default is `[]`.
`Ckts`	Cell array containing all circuit objects in the network, in order from source to load. All circuits must be 2-port	Handles to `rfckt` objects. Default is `{}`.
`Name`	Object name (read only)	String. `'Cascaded Network'`
`nPort`	Number of ports (read only)	Integer. The value is always `2`.

References

[1] Ludwig, Reinhold and Pavel Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

See Also

`analyze`, `calculate`, `listparam`, `listformat`, `plot`, `polar`, `rfckt`, `rfckt.hybrid`, `rfckt.hybridg`, `rfckt.parallel`, `rfckt.series`, `rfdata`, `smith`, `write`

rfckt.coaxial

Purpose Construct a coaxial transmission line object

Syntax

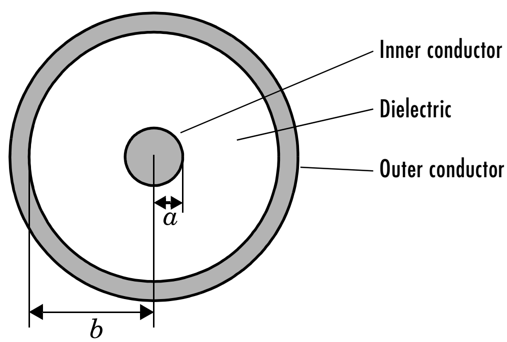
```
h = rfckt.coaxial('Property1',value1,'Property2',value2,...)
h = rfckt.coaxial
```

Description

`h = rfckt.coaxial('Property1',value1,'Property2',value2,...)` returns a coaxial transmission line object, `h`, with the specified properties. Properties you do not specify retain their default values.

`h = rfckt.coaxial` returns a coaxial transmission line object whose properties all have their default values.

A coaxial transmission line is shown here in cross-section. Its physical characteristics include the radius of the inner conductor of the coaxial transmission line a , and the radius of the outer conductor b .



Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis After you create the `coaxial` circuit object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.coaxial` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

A coaxial transmission line object enables you to model the transmission line as a stub or as a stubless line.

Stubless Transmission Line. If you model the transmission line as a stubless line, the `analyze` function calculates the S-parameters for the specified frequencies, based on the physical length of the transmission line, D, and the complex propagation constant, k.

$$S_{11} = 0$$

$$S_{12} = e^{-kD}$$

$$S_{21} = e^{-kD}$$

$$S_{22} = 0$$

k is a vector whose elements correspond to the elements of the input vector `freq`. k can be expressed in terms of the resistance (R), inductance (L), conductance (G), and capacitance (C) per unit length (meters) as

$$k = k_r + jk_i = \sqrt{(R + j2\pi fL)(G + j2\pi fC)}$$

where f is the frequency range specified in the `analyze` input argument `freq`, and

$$R = \frac{1}{2\pi\sigma_{\text{cond}}\delta}\left(\frac{1}{a} + \frac{1}{b}\right)$$

$$L = \frac{\mu}{2\pi}\ln(b/a)$$

$$G = \frac{2\pi\sigma_{\text{diel}}}{\ln(b/a)}$$

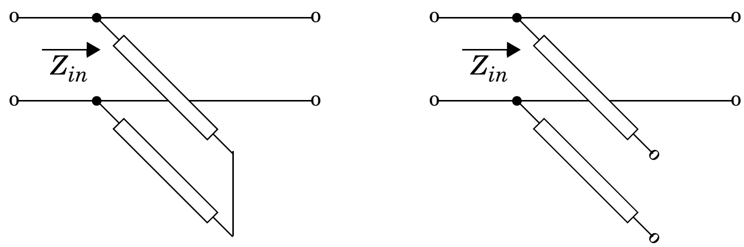
$$C = \frac{2\pi\varepsilon}{\ln(b/a)}$$

In these equations, σ_{cond} is the conductivity in the conductor and σ_{diel} is the conductivity in the dielectric. μ is the relative permeability of the dielectric,

ε is its permittivity as derived from the `EpsilonR` property, and skin depth δ is calculated as $1/\sqrt{\pi f \mu \sigma_{cond}}$.

Shunt and Series Stubs. If you model the transmission line as a shunt or series stub, the `analyze` function first calculates the ABCD-parameters at the specified frequencies. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

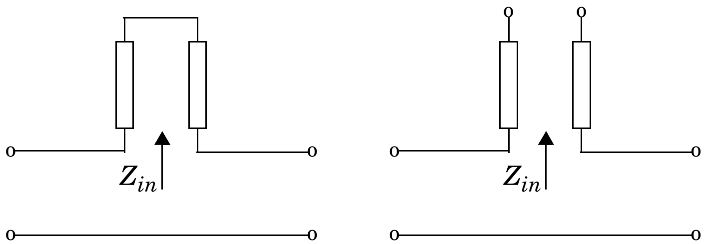
When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown here.



Z_{in} is the input impedance of the shunt circuit. The ABCD-parameters for the shunt stub are calculated as

$$A = 1$$
$$B = 0$$
$$C = 1/Z_{in}$$
$$D = 1$$

When you set the `StubMode` property to `'Series'`, the 2-port network consists of a series transmission line that you can terminate with either a short circuit or an open circuit as shown here.



Z_{in} is the input impedance of the series circuit. The ABCD-parameters for the series stub are calculated as

$$A = 1$$
$$B = Z_{in}$$
$$C = 0$$
$$D = 1$$

Properties

This table lists properties useful to `rfckt.coaxial` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the coaxial transmission line object	Handle. Default is `[]`.
`EpsilonR`	Relative permittivity of the dielectric expressed as the ratio of the permittivity of the dielectric to permittivity in free space ε_0	Default is `2.3`.
`Inner Radius`	Radius of the inner conductor	Meters. Default is `7.25e-4`.
`LineLength`	Physical length of the transmission line	Meters. Default is `0.01`.

Property	Description	Units, Values
Loss	Reduction in strength of the signal as it travels over the transmission line. Read-only; set by the analyze function.	Decibels per meter. Default is [].
MuR	Relative permeability of the dielectric expressed as the ratio of the permeability of the dielectric to permeability in free space μ_0	Default is 1.
Name	Object name (read only)	String. 'Coaxial Transmission Line'
nPort	Number of ports (read only)	Integer. The value is always 2.
Outer Radius	Radius of the outer conductor	Meters. Default is 0.0026.
PV	Phase velocity. Propagation velocity of a uniform plane wave on the transmission line. Read-only; set by the analyze function.	Meters per second. Default is [].
SigmaCond	Conductivity in the conductor	Siemens per meter (S/m). Default is Inf.
SigmaDiel	Conductivity in the dielectric	Siemens per meter (S/m). Default is 0.
StubMode	Type of stub.	String. 'None' (default), 'Series', or 'Shunt'

Property	Description	Units, Values
Termination	Stub termination for stub models Shunt and Series	String. 'None' (default), 'Open', or 'Short'. Use 'None' when StubMode is 'None'.
Z0	Characteristic impedance. Read-only; set by the analyze function.	Ohms. Default is [].

References

[1] Ludwig, Reinhold and Pavel Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

See Also

analyze, calculate, getz0, listparam, listformat, plot, polar, rfckt, rfckt.cpw, rfckt.microstrip, rfckt.parallelplate, rfckt.rlcgline, rfckt.twowire, rfckt.txline, rfdata, smith, write

rfckt.cpw

Purpose

Construct a coplanar waveguide transmission line object

Syntax

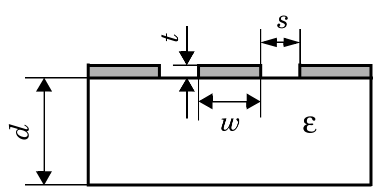
```
h = rfckt.cpw('Property1',value1,'Property2',value2,...)
h = rfckt.cpw
```

Description

`h = rfckt.cpw('Property1',value1,'Property2',value2,...)` returns a coplanar waveguide transmission line object, `h`, with the specified properties. Properties you do not specify retain their default values.

`h = rfckt.cpw` returns a coplanar waveguide transmission line object whose properties all have their default values.

A coplanar waveguide transmission line is shown here in cross-section. Its physical characteristics include the conductor width (w), the conductor thickness (t), the slot width (s), the substrate height (d), and the permittivity constant (ε).



Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis

After you create the `rfckt.cpw` circuit object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.cpw` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

A coplanar waveguide transmission line object enables you to model the transmission line as a stub or as a stubless line.

Stubless Transmission Line. If you model the transmission line as a stubless line, the `analyze` function calculates the S-parameters for the specified frequencies, based on the physical length of the transmission line, D, and the complex propagation constant, k.

$$S_{11} = 0$$

$$S_{12} = e^{-kD}$$

$$S_{21} = e^{-kD}$$

$$S_{22} = 0$$

$k = \alpha_a + i\beta$, where α_a is the attenuation coefficient and β is the wave number. The attenuation coefficient α_a is related to the loss, α, by

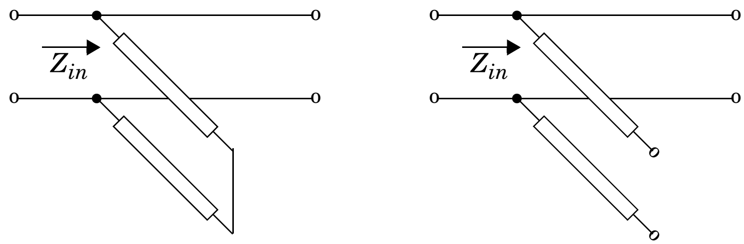
$$\alpha_a = -\ln 10^{-\frac{\alpha}{20}}$$

where α is the reduction in signal strength, in dB, per unit length. α combines both conductor loss and dielectric loss and is derived from the `rfckt.cpw` object properties.

The `analyze` function normalizes the S-parameters to 50 ohms. This is the default reference impedance of the `rfdata.data` object that the `analyze` function creates.

Shunt and Series Stubs. If you model the transmission line as a shunt or series stub, the `analyze` function first calculates the ABCD-parameters at the specified frequencies. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

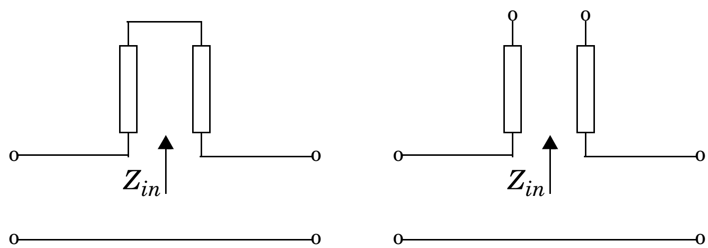
When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown here.



Z_{in} is the input impedance of the shunt circuit. The ABCD-parameters for the shunt stub are calculated as

$$A = 1$$
$$B = 0$$
$$C = 1/Z_{in}$$
$$D = 1$$

When you set the `StubMode` property to `'Series'`, the 2-port network consists of a series transmission line that you can terminate with either a short circuit or an open circuit as shown here.



Z_{in} is the input impedance of the series circuit. The ABCD-parameters for the series stub are calculated as

$$A = 1$$
$$B = Z_{in}$$
$$C = 0$$
$$D = 1$$

Properties

This table lists properties useful to `rfckt.cpw` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the coaxial transmission line object	Handle. Default is `[]`.
`ConductorWidth`	Physical width of the conductor.	Meters. Default is `0.6e-4`.
`EpsilonR`	Relative permittivity of the dielectric expressed as the ratio of the permittivity of the dielectric to permittivity in free space ε_0	Default is `9.8`.
`Height`	Thickness of the dielectric on which the conductor resides.	Meters. Default is `0.635e-4`.
`LineLength`	Physical length of the transmission line	Meters. Default is `0.01`.
`Loss`	Reduction in strength of the signal as it travels over the transmission line. Read-only; set by the `analyze` function.	Decibels per meter. Default is `[]`.
`LossTangent`	Loss angle tangent of the dielectric	Default is `0`.
`Name`	Object name (read only)	String. `'Coplanar Waveguide Transmission Line'`
`nPort`	Number of ports (read only)	Integer. The value is always `2`.

Property	Description	Units, Values
PV	Phase velocity. Propagation velocity of a uniform plane wave on the transmission line. Read-only; set by the analyze function.	Meters per second. Default is [].
SigmaCond	Conductivity in the conductor	Siemens per meter (S/m). Default is Inf.
SlotWidth	Physical width of the slot	Meters. Default is 0.2e-4.
StubMode	Type of stub	String. 'None' (default), 'Series', or 'Shunt'
Termination	Termination for stub modes 'Shunt' and 'Series'.	String. 'None' (default), 'Open', or 'Short'. Use 'None' when StubMode is 'None'.
Thickness	Physical thickness of the conductor.	Meters. Default is 0.005e-6.
Z0	Characteristic impedance. Read-only; set by the analyze function.	Ohms. Default is [].

References

[1] Gupta, K. C., Ramesh Garg, Inder Bahl, and Prakash Bhartia, *Microstrip Lines and Slotlines*, 2nd Edition, Artech House, Inc., Norwood, MA, 1996.

See Also

analyze, calculate, getz0, listparam, listformat, plot, polar, rfckt, rfckt.coaxial, rfckt.microstrip, rfckt.parallelplate, rfckt.rlcgline, rfckt.twowire, rfckt.txline, rfdata, smith, write

Purpose Construct a circuit object from a data file

```
h = rfckt.datafile('Property1',value1,'Property2',value2,...)
h = rfckt.datafile
```

Description `h = rfckt.datafile('Property1',value1,'Property2',value2,...)` returns a circuit object, `h`, based on the specified properties. Use the `'File'` property to specify a source `.snp`, `.ynp`, `.znp`, `.hnp`, or `.amp` file that describes an n-port circuit. Properties you do not specify retain their default values. See Appendix A, "AMP File Format" for information about the `.amp` format.

`h = rfckt.datafile` returns a circuit object whose properties all have their default values.

Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis After you create the `datafile` circuit object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.datafile` objects, `freq` must be nonnegative.

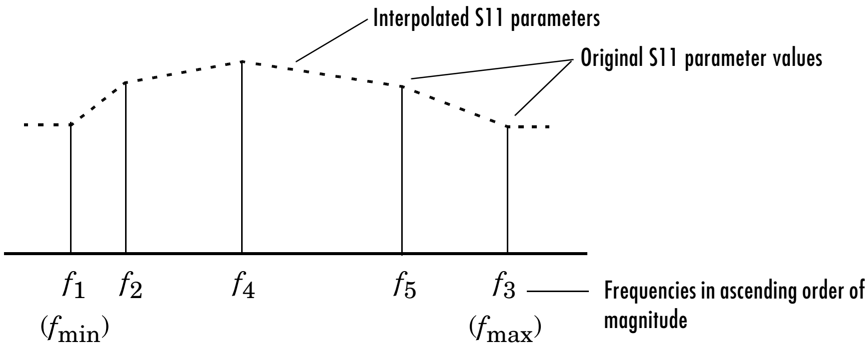
```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

If the file you specify with the `'File'` property contains network Y- or Z-parameters, `analyze` first converts these parameters, as they exist in the `rfckt.datafile` object, to S-parameters. Using the interpolation method you specify with the `'IntpType'` property, `analyze` interpolates the S-parameters to determine the S-parameters at the specified frequencies.

Specifically, `analyze` orders the S-parameters according to the ascending order of their frequencies, f_n. It then interpolates the S-parameters, using the MATLAB `interp1` function. For example, the curve in the following diagram illustrates the result of interpolating the S11 parameters at five different frequencies.



You can specify the interpolation method as `cubic`, `linear` (default), or `spline`. For more information, see "One-Dimensional Interpolation" and the `interp1` reference page in the MATLAB documentation.

As shown in the diagram above, `analyze` uses the parameter values at f_{min}, the minimum input frequency, for all frequencies smaller than f_{min}. It uses the parameters values at f_{max}, the maximum input frequency, for all frequencies greater than f_{max}. In both cases, the results may not be accurate.

Properties

This table lists properties useful to `rfckt.datafile` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the circuit object	Handle. Default is `[1x1 rfdata.data]`
`File`	`.snp`, `.ynp`, `.znp`, or `.hnp` file describing a circuit, where `n` is the number of ports	String. Default is `'passive.s2p'`.
`IntpType`	Interpolation method	`'linear'` (default), `'spline'`, or `'cubic'`

Property	Description	Units, Values
Name	Object name (read only)	String. 'Data File'
nPort	Number of ports.	Integer. Default is 2.

References

[1] EIA/IBIS Open Forum, "Touchstone File Format Specification," Rev. 1.1, 2002 (http://www.eda.org/pub/ibis/connector/touchstone_spec11.pdf).

See Also

analyze, calculate, listparam, listformat, plot, polar, read, restore, rfckt, rfckt.amplifier, rfckt.mixer, rfckt.passive, rfdata, smith, write

rfckt.delay

Purpose Construct a delay line object

Syntax

```
h = rfckt.delay('Property1',value1,'Property2',value2,...)
h = rfckt.delay
```

Description

`h = rfckt.delay('Property1',value1,'Property2',value2,...)` returns a delay line object, `h`, based on the specified properties. Properties you do not specify retain their default values.

`h = rfckt.delay` returns a delay line object whose properties all have their default values.

Note See the `rfckt` reference page for a list of functions that act on circuit (rfckt) objects.

Circuit Analysis

After you create the `delay` circuit object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.delay` objects, the elements of the vector `freq` must be strictly positive.

```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

The delay line object enables you to model time delay which can be lossy or lossless. It is treated as a two-port linear network.

The `analyze` function calculates the S-parameters for the specified frequencies, based on the values of the delay line's `loss`, α, and time delay, D.

$$S_{11} = 0$$

$$S_{12} = e^{-p}$$

$$S_{21} = e^{-p}$$

$$S_{22} = 0$$

where $p = \alpha_a + i\beta$, and α_a is the attenuation coefficient and β is the wave number. The attenuation coefficient α_a is related to the loss, α, by

$$\alpha_a = -\ln 10^{-\frac{\alpha}{20}}$$

and the wave number β is related to the time delay, D, by

$$\beta = 2\pi f D$$

where f is the frequency range specified in the `analyze` input argument `freq`.

Properties

This table lists properties useful to `rfckt.delay` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the delay line object	Handle. Default is `[]`.
`Loss`	Reduction in strength of the signal as it travels over the delay line	Decibels. Must be positive. Default is `0`.
`Name`	Object name (read only)	String. `'Delay'`
`nPort`	Number of ports (read only)	Integer. The value is always `2`.
`TimeDelay`	Time delay	Seconds. Default is 1.0000e-012
`Z0`	Characteristic impedance	Ohms. Default is 50.

References

[1] Ludwig, Reinhold and Pavel Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

See Also

`analyze`, `calculate`, `getz0`, `listparam`, `listformat`, `plot`, `polar`, `rfckt`, `rfckt.rlcgline`, `rfckt.txline`, `rfdata`, `smith`, `write`

Purpose Construct a hybrid connected network object

Syntax

```
h = rfckt.hybrid('Property1',value1,'Property2',value2,...)
h = rfckt.hybrid
```

Description `h = rfckt.hybrid('Property1',value1,'Property2',value2,...)` returns a hybrid connected network object, `h`, based on the specified properties. Use the `'Ckts'` property to specify the `rfckt` objects to be connected. Properties you do not specify retain their default values.

`h = rfckt.hybrid` returns a hybrid connected network object whose properties all have their default values.

Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

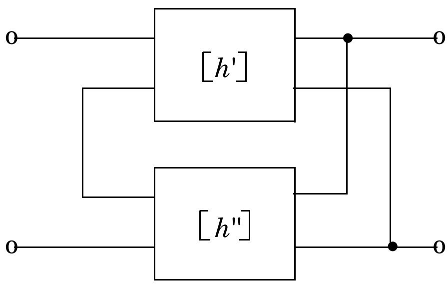
Circuit Analysis After you create the `hybrid` network object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.hybrid` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

The `analyze` function first calculates the h matrix of the hybrid network. It starts by converting each component network's parameters to an h matrix. The figure shows a hybrid connected network consisting of two 2-port networks, each represented by its h matrix.



where $\left[h'\right] = \begin{bmatrix} h_{11}' & h_{12}' \\ h_{21}' & h_{22}' \end{bmatrix}$ and $\left[h''\right] = \begin{bmatrix} h_{11}'' & h_{12}'' \\ h_{21}'' & h_{22}'' \end{bmatrix}$

The `analyze` function then calculates the h matrix for the hybrid network by calculating the sum of the h matrices of the individual networks. The following equation illustrates the calculations for two 2-port networks.

$$\left[h\right] = \begin{bmatrix} h_{11}' + h_{11}'' & h_{12}' + h_{12}'' \\ h_{21}' + h_{21}'' & h_{22}' + h_{22}'' \end{bmatrix}$$

Finally, `analyze` converts the h matrix of the hybrid network to S-parameters at the frequencies specified in the `analyze` input argument `freq`.

Properties

This table lists properties useful to `rfckt.hybrid` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the hybrid connected network object	Handle. Default is `[]`
`Ckts`	Cell array containing all circuit objects in the network, in order from source to load. All circuits must be 2-port.	Handles to `rfckt` objects. Default is `{}`.
`Name`	Object name (read only)	String. `'Hybrid Connected Network'`
`nPort`	Number of ports (read only)	Integer. The value is always `2`.

References

[1] Ludwig, Reinhold and Pavel Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

See Also

`analyze`, `calculate`, `listparam`, `listformat`, `plot`, `polar`, `rfckt`, `rfckt.cascade`, `rfckt.hybridg`, `rfckt.parallel`, `rfckt.series`, `rfdata`, `smith`, `write`

rfckt.hybridg

Purpose Construct an inverse hybrid connected network object

Syntax

```
h = rfckt.hybridg('Property1',value1,'Property2',value2,...)
h = rfckt.hybridg
```

Description `h = rfckt.hybridg('Property1',value1,'Property2',value2,...)` returns an inverse hybrid connected network object, `h`, based on the specified properties. Use the `'Ckts'` property to specify the `rfckt` objects to be connected. Properties you do not specify retain their default values.

`h = rfckt.hybridg` returns an inverse hybrid connected network object whose properties all have their default values.

Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

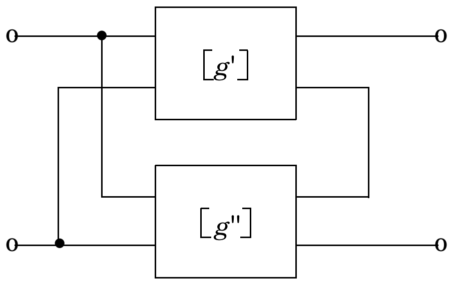
Circuit Analysis After you create the inverse `hybrid` network object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.hybridg` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

The `analyze` function first calculates the g matrix of the inverse hybrid network. It starts by converting each component network's parameters to a g matrix. The figure shows an inverse hybrid connected network consisting of two 2-port networks, each represented by its g matrix.



where $[g'] = \begin{bmatrix} g_{11}' & g_{12}' \\ g_{21}' & g_{22}' \end{bmatrix}$ and $[g''] = \begin{bmatrix} g_{11}'' & g_{12}'' \\ g_{21}'' & g_{22}'' \end{bmatrix}$

The `analyze` function then calculates the g matrix for the inverse hybrid network by calculating the sum of the g matrices of the individual networks. The following equation illustrates the calculations for two 2-port networks.

$$[g] = \begin{bmatrix} g_{11}' + g_{11}'' & g_{12}' + g_{12}'' \\ g_{21}' + g_{21}'' & g_{22}' + g_{22}'' \end{bmatrix}$$

Finally, `analyze` converts the g matrix of the inverse hybrid network to S-parameters at the frequencies specified in the `analyze` input argument `freq`.

Properties

This table lists properties useful to `rfckt.hybridg` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the inverse hybrid connected network object	Handle. Default is `[]`.
`Ckts`	Cell array containing all circuit objects in the network, in order from source to load. All circuits must be two-port.	Handles to `rfckt` objects. Default is `{}`.
`Name`	Object name (read only)	String. `'Hybrid G Connected Network'`
`nPort`	Number of ports (read only)	Integer. The value is always `2`.

References

Davis, Artice M., *Linear Circuit Analysis*, PWS Publishing Company, 1998.

See Also

`analyze`, `calculate`, `listparam`, `listformat`, `plot`, `polar`, `rfckt`, `rfckt.cascade`, `rfckt.hybrid`, `rfckt.parallel`, `rfckt.series`, `rfdata`, `smith`, `write`

Purpose Construct an LC bandpass pi network object

Syntax

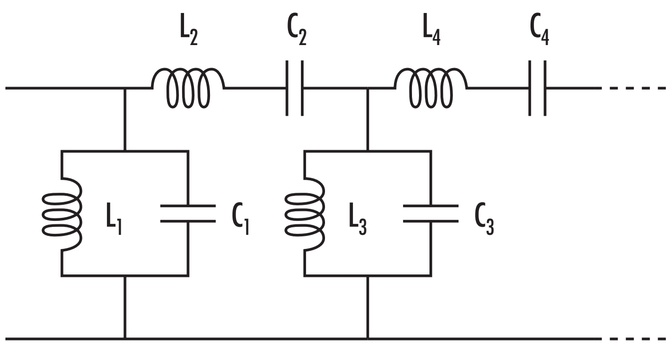
```
h = rfckt.lcbandpasspi('Property1',value1,'Property2',value2,...)
h = rfckt.lcbandpasspi
```

Description

`h = rfckt.lcbandpasspi('Property1',value1,'Property2',value2,...)` returns an LC bandpass pi network object, `h`, based on the specified properties. Properties you do not specify retain their default values.

`h = rfckt.lcbandpasspi` returns an LC bandpass pi network object whose properties all have their default values.

The LC bandpass pi network object is a two-port network as shown in the circuit diagram below.



Where [L_1, L_2, L_3, L_4, ...] is the value of the `'L'` property, and [C_1, C_2, C_3, C_4, ...] is the value of the `'C'` property.

Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis After you create the `lcbandpasspi` circuit object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.lcbandpasspi` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

For each inductor and capacitor pair in the network, the `analyze` function first calculates the ABCD-parameters for each frequency in the input vector, `freq`. For each series pair, A = 1, B = Z, C = 0, and D = 1, where Z is the impedance of the series pair. For each shunt pair, A = 1, B = 0, C = Y, and D = 1, where Y is the admittance of the shunt pair.

The `analyze` function cascades the ABCD-parameters for each series and shunt pair, then converts the cascaded parameters to S-parameters using the `abcd2s` function.

Properties

This table lists properties useful to `rfckt.lcbandpasspi` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the LC bandpass pi network object	Handle. Default is `[]`.
`C`	Vector containing the capacitances, in order from source to load, of all capacitors in the network. Its length must be equal to the length of the vector you provide for `'L'`. All values must be strictly positive.	Farads. Default is `[0.3579e-10, 0.0118e-10, 0.3579e-10]`.
`L`	Vector containing the inductances, in order from source to load, of all inductors in the network. The inductance vector must contain at least three elements. All values must be strictly positive.	Henrys. Default is `[0.0144e-7, 0.4395e-7, 0.0144e-7]`.

Property	Description	Units, Values
`Name`	Object name (read only)	String. `'LC Bandpass Pi'`
`nPort`	Number of ports (read only)	Integer. The value is always `2`.

References

[1] Ludwig, Reinhold and Pavel Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

[2] Zverev, Anatol I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

`analyze`, `calculate`, `listparam`, `listformat`, `plot`, `polar`, `rfckt`, `rfckt.lcbandpasstee`, `rfdata`, `smith`, `write`

rfckt.lcbandpasstee

Purpose Construct an LC bandpass tee network object

Syntax

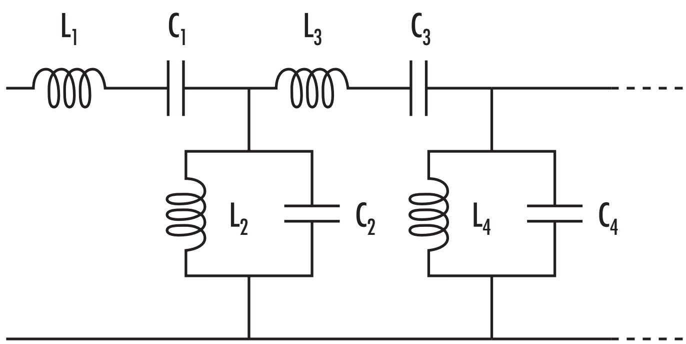
```
h = rfckt.lcbandpasstee('Property1',value1,'Property2',value2,...)
h = rfckt.lcbandpasstee
```

Description

`h = rfckt.lcbandpasstee('Property1',value1,'Property2',value2,...)` returns an LC bandpass tee network object, `h`, based on the specified properties. Properties you do not specify retain their default values.

`h = rfckt.lcbandpasstee` returns an LC bandpass tee network object whose properties all have their default values.

The LC bandpass tee network object is a two-port network as shown in the circuit diagram below.



Where [L_1, L_2, L_3, L_4, ...] is the value of the `'L'` property, and [C_1, C_2, C_3, C_4, ...] is the value of the `'C'` property.

Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis After you create the `lcbandpasstee` circuit object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.lcbandpasstee` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

For each inductor and capacitor pair in the network, the `analyze` function first calculates the ABCD-parameters for each frequency in the input vector, `freq`. For each series pair, A = 1, B = Z, C = 0, and D = 1, where Z is the impedance of the series pair. For each shunt pair, A = 1, B = 0, C = Y, and D = 1, where Y is the admittance of the shunt pair.

The `analyze` function cascades the ABCD-parameters for each series and shunt pair, then converts the cascaded parameters to S-parameters using the `abcd2s` function.

Properties

This table lists properties useful to `rfckt.lcbandpasstee` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the LC bandpass tee network object	Handle. Default is `[]`.
`C`	Vector containing the capacitances, in order from source to load, of all capacitors in the network. Its length must be equal to the length of the vector you provide for `'L'`. All values must be strictly positive.	Farads. Default is `[0.0186e-10, 0.1716e-10, 0.0186e-10]`.
`L`	Vector containing the inductances, in order from source to load, of all inductors in the network. The inductance vector must contain at least three elements. All values must be strictly positive.	Henrys. Default is `[0.2781e-7, 0.0301e-7, 0.2781e-7]`.

Property	Description	Units, Values
`Name`	Object name (read only)	String. `'LC Bandpass Tee'`
`nPort`	Number of ports (read only)	Integer. The value is always `2`.

References

[1] Ludwig, Reinhold and Pavel Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

[2] Zverev, Anatol I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

`analyze`, `calculate`, `listparam`, `listformat`, `plot`, `polar`, `rfckt`, `rfckt.lcbandpasspi`, `rfdata`, `smith`, `write`

Purpose Construct an LC bandstop pi network object

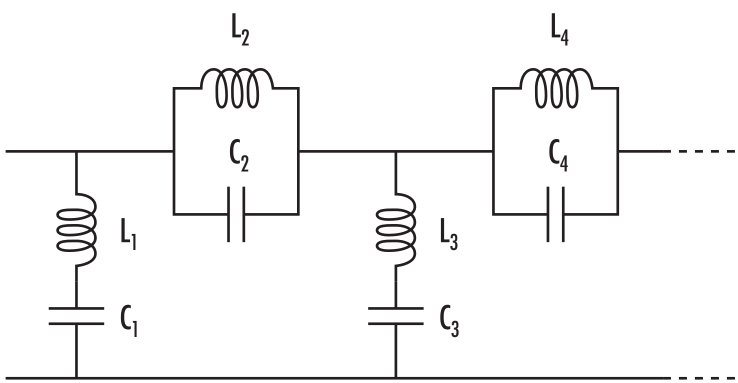
Syntax

```
h = rfckt.lcbandstoppi('Property1',value1,'Property2',value2,...)
h = rfckt.lcbandstoppi
```

Description `h = rfckt.lcbandstoppi('Property1',value1,'Property2',value2,...)` returns an LC bandstop pi network object, `h`, based on the specified properties. Properties you do not specify retain their default values.

`h = rfckt.lcbandstoppi` returns an LC bandstop pi network object whose properties all have their default values.

The LC bandstop pi network object is a two-port network as shown in the circuit diagram below.



Where [L_1, L_2, L_3, L_4, ...] is the value of the `'L'` property, and [C_1, C_2, C_3, C_4, ...] is the value of the `'C'` property.

Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis After you create the `lcbandstoppi` circuit object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.lcbandstoppi` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

The analyze function stores the results of the analysis in the AnalyzedResult property of the circuit object.

Network Parameters

For each inductor and capacitor pair in the network, the analyze function first calculates the ABCD-parameters for each frequency in the input vector, freq. For each series pair, A = 1, B = Z, C = 0, and D = 1, where Z is the impedance of the series pair. For each shunt pair, A = 1, B = 0, C = Y, and D = 1, where Y is the admittance of the shunt pair.

The analyze function cascades the ABCD-parameters for each series and shunt pair, then converts the cascaded parameters to S-parameters using the abcd2s function.

Properties

This table lists properties useful to rfckt.lcbandstoppi objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
AnalyzedResult	rfdata.data object that contains the result of applying the analyze function to the LC bandstop pi network object	Handle. Default is [].
C	Vector containing the capacitances, in order from source to load, of all capacitors in the network. Its length must be equal to the length of the vector you provide for 'L'. All values must be strictly positive.	Farads. Default is [0.0184e-10, 0.2287e-10, 0.0184e-10].
L	Vector containing the inductances, in order from source to load, of all inductors in the network. The inductance vector must contain at least three elements. All values must be strictly positive.	Henrys. Default is [0.2809e-7, 0.0226e-7, 0.2809e-7].

Property	Description	Units, Values
`Name`	Object name (read only)	String. `'LC Bandstop Pi'`
`nPort`	Number of ports (read only)	Integer. The value is always `2`.

References

[1] Ludwig, Reinhold and Pavel Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

[2] Zverev, Anatol I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

`analyze`, `calculate`, `listparam`, `listformat`, `plot`, `polar`, `rfckt`, `rfckt.lcbandstoptee`, `rfdata`, `smith`, `write`

rfckt.lcbandstoptee

Purpose

Construct an LC bandstop tee network object

Syntax

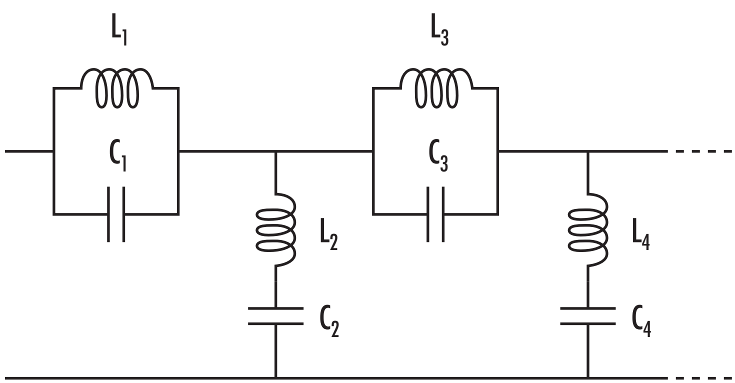
```
h = rfckt.lcbandstoptee('Property1',value1,'Property2',value2,...)
h = rfckt.lcbandstoptee
```

Description

`h = rfckt.lcbandstoptee('Property1',value1,'Property2',value2,...)` returns an LC bandstop tee network object, `h`, based on the specified properties. Properties you do not specify retain their default values.

`h = rfckt.lcbandstoptee` returns an LC bandstop tee network object whose properties all have their default values.

The LC bandstop tee network object is a two-port network as shown in the circuit diagram below.



Where [L_1, L_2, L_3, L_4, ...] is the value of the `'L'` property, and [C_1, C_2, C_3, C_4, ...] is the value of the `'C'` property.

Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis

After you create the `lcbandstoptee` circuit object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.lcbandstoptee` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

The analyze function stores the results of the analysis in the AnalyzedResult property of the circuit object.

Network Parameters

For each inductor and capacitor pair in the network, the analyze function first calculates the ABCD-parameters for each frequency in the input vector, freq. For each series pair, A = 1, B = Z, C = 0, and D = 1, where Z is the impedance of the series pair. For each shunt pair, A = 1, B = 0, C = Y, and D = 1, where Y is the admittance of the shunt pair.

The analyze function cascades the ABCD-parameters for each series and shunt pair, then converts the cascaded parameters to S-parameters using the abcd2s function.

Properties

This table lists properties useful to rfckt.lcbandstoptee objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
AnalyzedResult	rfdata.data object that contains the result of applying the analyze function to the LC bandstop tee network object	Handle. Default is [].
C	Vector containing the capacitances, in order from source to load, of all capacitors in the network. Its length must be equal to the length of the vector you provide for 'L'. All values must be strictly positive.	Farads. Default is [0.1852e-10, 0.0105e-10, 0.1852e-10].
L	Vector containing the inductances, in order from source to load, of all inductors in the network. The inductance vector must contain at least three elements. All values must be strictly positive.	Henrys. Default is [0.0279e-7, 0.4932e-7, 0.0279e-7].

Property	Description	Units, Values
`Name`	Object name (read only)	String. `'LC Bandstop Tee'`
`nPort`	Number of ports (read only)	Integer. The value is always `2`.

References

[1] Ludwig, Reinhold and Pavel Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

[2] Zverev, Anatol I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

`analyze`, `calculate`, `listparam`, `listformat`, `plot`, `polar`, `rfdata`, `rfckt`, `rfckt.lcbandstoppi`, `rfdata`, `smith`, `write`

Purpose Construct an LC highpass pi network object

Syntax

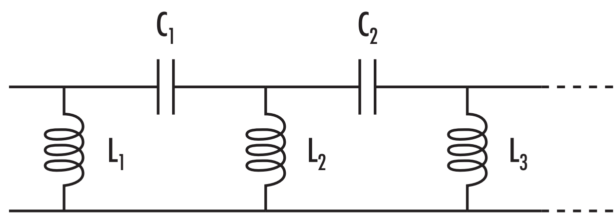
```
h = rfckt.lchighpasspi('Property1',value1,'Property2',value2,...)
h = rfckt.lchighpasspi
```

Description

`h = rfckt.lchighpasspi('Property1',value1,'Property2',value2,...)` returns an LC highpass pi network object, `h`, based on the specified properties. Properties you do not specify retain their default values.

`h = rfckt.lchighpasspi` returns an LC highpass pi network object whose properties all have their default values.

The LC highpass pi network object is a two-port network as shown in the circuit diagram below.



Where [L_1, L_2, L_3, ...] is the value of the `'L'` property, and [C_1, C_2, ...] is the value of the `'C'` property.

Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis

After you create the `lchighpasspi` circuit object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.lchighpasspi` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

For each inductor and capacitor in the network, the `analyze` function first calculates the ABCD-parameters for each frequency in the input vector, `freq`. For each series element, A = 1, B = Z, C = 0, and D = 1, where Z is the impedance of the series element. For each shunt element, A = 1, B = 0, C = Y, and D = 1, where Y is the admittance of the shunt element.

The `analyze` function cascades the ABCD-parameters for each circuit element, then converts the cascaded parameters to S-parameters using the `abcd2s` function.

Properties

This table lists properties useful to `rfckt.lchighpasspi` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the LC highpass pi network object	Handle. Default is `[]`.
`L`	Vector containing the inductances, in order from source to load, of all inductors in the network. The inductance vector must contain at least two elements. All values must be strictly positive.	Henrys. Default is `[2.2363e-9]`.
`C`	Vector containing the capacitances, in order from source to load, of all capacitors in the network. Its length must be equal to or one less than the length of the vector you provide for `'L'`. All values must be strictly positive.	Farads. Default is `[0.1188e-5, 0.1188e-5]`.

Property	Description	Units, Values
`Name`	Object name (read only)	String. `'LC Highpass Pi'`
`nPort`	Number of ports (read only)	Integer. The value is always `2`.

References

[1] Ludwig, Reinhold and Pavel Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

[2] Zverev, Anatol I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

`analyze`, `calculate`, `listparam`, `listformat`, `plot`, `polar`, `rfdata`, `rfckt`, `rfckt.lchighpasstee`, `rfdata`, `smith`, `write`

rfckt.lchighpasstee

Purpose

Construct an LC highpass tee network object

Syntax

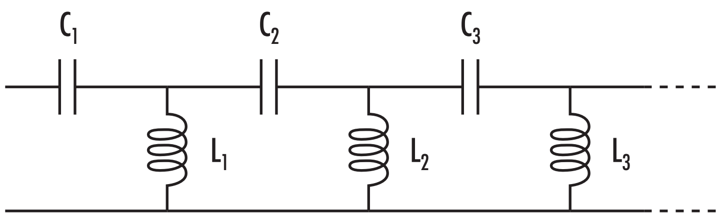
```
h = rfckt.lchighpasstee('Property1',value1,'Property2',value2,...)
h = rfckt.lchighpasstee
```

Description

`h = rfckt.lchighpasstee('Property1',value1,'Property2',value2,...)` returns an LC highpass tee network object, `h`, with the specified properties. Properties you do not specify retain their default values.

`h = rfckt.lchighpasstee` returns an LC highpass tee network object whose properties all have their default values.

The LC highpass tee network object is a two-port network as shown in the circuit diagram below.



Where [L_1, L_2, L_3, ...] is the value of the `'L'` property, and [C_1, C_2, C_3, ...] is the value of the `'C'` property.

Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis

After you create the `lchighpasstee` circuit object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.lchighpasstee` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

For each inductor and capacitor in the network, the `analyze` function first calculates the ABCD-parameters for each frequency in the input vector, `freq`. For each series element, A = 1, B = Z, C = 0, and D = 1, where Z is the impedance of the series element. For each shunt element, A = 1, B = 0, C = Y, and D = 1, where Y is the admittance of the shunt element.

The `analyze` function cascades the ABCD-parameters for each circuit element, then converts the cascaded parameters to S-parameters using the `abcd2s` function.

Properties

This table lists properties useful to `rfckt.lchighpasstee` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the LC highpass tee network object	Handle. Default is `[]`.
`C`	Vector containing the capacitances, in order from source to load, of all capacitors in the network. The capacitance vector must contain at least two elements. Its length must be equal to or one greater than the length of the vector you provide for `'L'`. All values must be strictly positive.	Farads. Default is `[0.4752e-9, 0.4752e-9]`.
`L`	Vector containing the inductances, in order from source to load, of all inductors in the network. All values must be strictly positive. The vector cannot be empty.	Henrys. Default is `[5.5907e-6]`.

Property	Description	Units, Values
`Name`	Object name (read only)	String. `'LC Highpass Tee'`
`nPort`	Number of ports (read only)	Integer. The value is always `2`.

References

[1] Ludwig, Reinhold and Pavel Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

[2] Zverev, Anatol I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

`analyze`, `calculate`, `listparam`, `listformat`, `plot`, `polar`, `rfckt`, `rfckt.lchighpasspi`, `rfdata`, `smith`, `write`

Purpose Construct an LC lowpass pi network object

Syntax

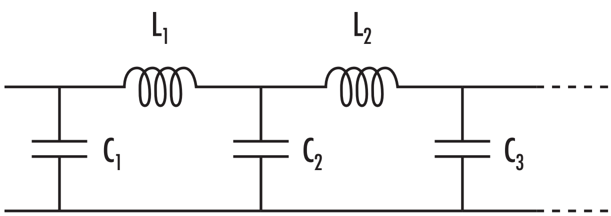
```
h = rfckt.lclowpasspi('Property1',value1,'Property2',value2,...)
h = rfckt.lclowpasspi
```

Description

`h = rfckt.lclowpasspi('Property1',value1,'Property2',value2,...)` returns an LC lowpass pi network object, `h`, based on the specified properties. Properties you do not specify retain their default values.

`h = rfckt.lclowpasspi` returns an LC lowpass pi network object whose properties all have their default values.

The LC lowpass pi network object is a two-port network as shown in the circuit diagram below.



Where [L_1, L_2, ...] is the value of the `'L'` property, and [C_1, C_2, C_3, ...] is the value of the `'C'` property.

Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis

After you create the `lclowpasspi` circuit object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.lclowpasspi` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

For each inductor and capacitor in the network, the `analyze` function first calculates the ABCD-parameters for each frequency in the input vector, `freq`. For each series element, A = 1, B = Z, C = 0, and D = 1, where Z is the impedance of the series element. For each shunt element, A = 1, B = 0, C = Y, and D = 1, where Y is the admittance of the shunt element.

The `analyze` function cascades the ABCD-parameters for each circuit element, then converts the cascaded parameters to S-parameters using the `abcd2s` function.

Properties

This table lists properties useful to `rfckt.lclowpasspi` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the LC lowpass pi network object	Handle. Default is `[]`.
`C`	Vector containing the capacitances, in order from source to load, of all capacitors in the network. The capacitance vector must contain at least two elements. Its length must be equal to or one greater than the length of the vector you provide for `'L'`. All values must be strictly positive.	Farads. Default is `[0.5330e-8, 0.5330e-8]`.
`L`	Vector containing the inductances, in order from source to load, of all inductors in the network. All values must be strictly positive. The vector cannot be empty.	Henrys. Default is `[2.8318e-6]`.

Property	Description	Units, Values
`Name`	Object name (read only)	String. `'LC Lowpass Pi'`
`nPort`	Number of ports (read only)	Integer. The value is always `2`.

References

[1] Ludwig, Reinhold and Pavel Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

[2] Zverev, Anatol I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

`analyze`, `calculate`, `listparam`, `listformat`, `plot`, `polar`, `rfckt`, `rfckt.lclowpasstee`, `rfdata`, `smith`, `write`

rfckt.lclowpasstee

Purpose Construct an LC lowpass tee filter object

Syntax

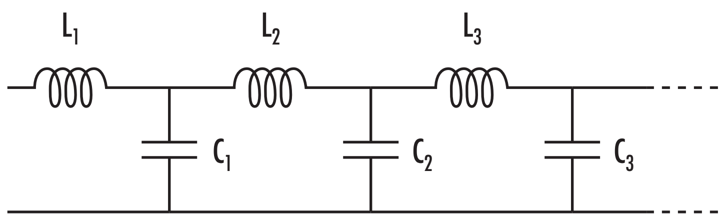
```
h = rfckt.lclowpasstee
h = rfckt.lclowpasstee('Property1',value1,'Property2',value2,...)
```

Description

`h = rfckt.lclowpasstee` returns an LC lowpass tee filter object whose properties all have their default values.

`h = rfckt.lclowpasstee('Property1',value1,'Property2',value2,...)` returns an LC lowpass tee filter object, `h`, based on the specified properties. Properties you do not specify retain their default values.

The LC lowpass tee network object is a two-port network as shown in the circuit diagram below.



Where [L_1, L_2, L_3, ...] is the value of the `'L'` property, and [C_1, C_2, C_3, ...] is the value of the `'C'` property.

Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis

After you create the `lclowpasstee` circuit object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.lclowpasstee` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

For each inductor and capacitor in the network, the `analyze` function first calculates the ABCD-parameters for each frequency in the input vector, `freq`. For each series element, A = 1, B = Z, C = 0, and D = 1, where Z is the impedance of the series element. For each shunt element, A = 1, B = 0, C = Y, and D = 1, where Y is the admittance of the shunt element.

The `analyze` function cascades the ABCD-parameters for each circuit element, then converts the cascaded parameters to S-parameters using the `abcd2s` function.

Properties

This table lists properties associated with `rfckt.lclowpasstee` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the LC lowpass tee network object	Handle. Default is `[]`.
`C`	Vector containing the capacitances, in order from source to load, of all capacitors in the network. Its length must be equal to or one less than the length of the vector you provide for `'L'`. All values must be strictly positive.	Farads. Default is `[1.1327e-9]`.
`L`	Vector containing the inductances, in order from source to load, of all inductors in the network. The inductance vector must contain at least two elements. All values must be strictly positive.	Henrys. Default is `[0.1332e-4, 0.1332e-4]`.

Property	Description	Units, Values
`Name`	Object name (read only)	String. `'LC Lowpass Tee'`
`nPort`	Number of ports (read only)	Integer. The value is always `2`.

References

[1] Ludwig, Reinhold and Pavel Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

[2] Zverev, Anatol I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

`analyze`, `calculate`, `listparam`, `listformat`, `plot`, `polar`, `rfckt`, `rfckt.lclowpasspi`, `rfdata`, `smith`, `write`

Purpose Construct a microstrip transmission line object

Syntax

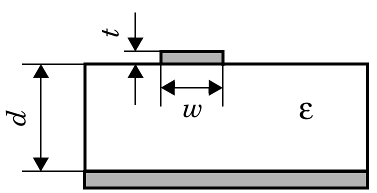
```
h = rfckt.microstrip('Property1',value1,'Property2',value2,...)
h = rfckt.microstrip
```

Description

`h = rfckt.microstrip('Property1',value1,'Property2',value2,...)` returns a microstrip transmission line object, `h`, with the specified properties. Properties you do not specify retain their default values.

`h = rfckt.microstrip` returns a microstrip transmission line object whose properties all have their default values.

A microstrip transmission line is shown here in cross-section. Its physical characteristics include the microstrip width (w), the microstrip thickness (t), the substrate height (d), and the relative permittivity constant (ε).



Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis

After you create the `microstrip` circuit object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.microstrip` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

A microstrip transmission line object enables you to model the transmission line as a stub or as a stubless line.

Stubless Transmission Line. If you model the transmission line as a stubless line, the `analyze` function calculates the S-parameters for the specified frequencies, based on the physical length of the transmission line, D, and the complex propagation constant, k.

$$S_{11} = 0$$

$$S_{12} = e^{-kD}$$

$$S_{21} = e^{-kD}$$

$$S_{22} = 0$$

$k = \alpha_a + i\beta$, where α_a is the attenuation coefficient and β is the wave number. The attenuation coefficient α_a is related to the loss, α, by

$$\alpha_a = -\ln 10^{-\frac{\alpha}{20}}$$

where α is the reduction in signal strength, in dB, per unit length. α combines both conductor loss and dielectric loss and is derived from the `rfckt.microstrip` object properties.

The wave number β is related to the phase velocity, V_P, by

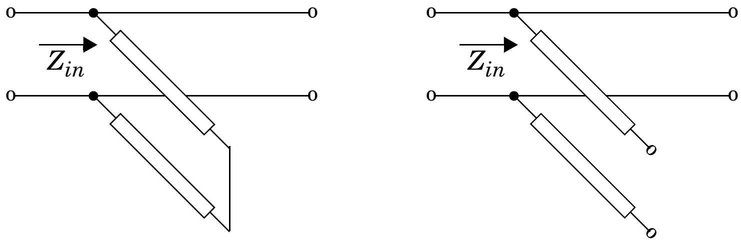
$$\beta = \frac{2\pi f}{V_p}$$

$V_P = c/\sqrt{\varepsilon_{\text{eff}}}$ where ε_{eff} is the frequency dependent effective dielectric constant. f is the frequency range specified in the `analyze` input argument `freq`. V_P and ε_{eff} are derived from the `rfckt.microstrip` object properties.

The phase velocity V_p is also known as the wave propagation velocity.

Shunt and Series Stubs. If you model the transmission line as a shunt or series stub, the `analyze` function first calculates the ABCD-parameters at the specified frequencies. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

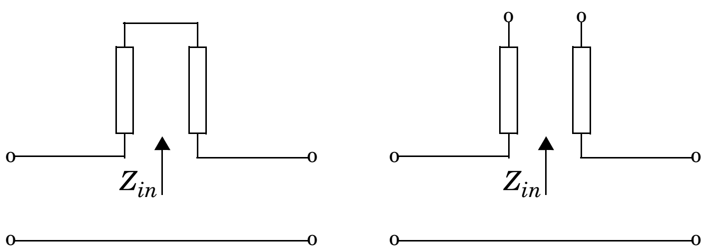
When you set the StubMode property to 'Shunt', the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown here.



Z_{in} is the input impedance of the shunt circuit. The ABCD-parameters for the shunt stub are calculated as

$$A = 1$$
$$B = 0$$
$$C = 1/Z_{in}$$
$$D = 1$$

When you set the StubMode property to 'Series', the 2-port network consists of a series transmission line that you can terminate with either a short circuit or an open circuit as shown here.



Z_{in} is the input impedance of the series circuit. The ABCD-parameters for the series stub are calculated as

$$A = 1$$
$$B = Z_{in}$$
$$C = 0$$
$$D = 1$$

Properties

This table lists properties useful to `rfckt.microstrip` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the microstrip transmission line object	Handle. Default is `[]`.
`EpsilonR`	Relative permittivity of the dielectric expressed as the ratio of the permittivity of the dielectric to permittivity in free space ε_0	Default is `9.8`.
`Height`	Thickness of the dielectric on which the microstrip resides	Meters. Default is `6.35e-4`.
`LineLength`	Physical length of the transmission line	Meters. Default is `0.01`.
`Loss`	Reduction in strength of the signal as it travels over the transmission line. Read-only; set by the `analyze` function.	Decibels per meter. Default is `[]`.
`LossTangent`	Loss angle tangent of the dielectric	Default is `0`.
`Name`	Object name (read only)	String. `'Microstrip Transmission Line'`
`nPort`	Number of ports (read only)	Integer. The value is always `2`.

Property	Description	Units, Values
PV	Phase velocity. Propagation velocity of a uniform plane wave on the transmission line. Read-only; set by the analyze function.	Meters per second. Default is [].
SigmaCond	Conductivity in the conductor	Siemens per meter (S/m). Default is Inf.
StubMode	Type of stub	String. 'None' (default), 'Series', or 'Shunt'
Termination	Termination for stub modes 'Shunt' and 'Series'.	String. 'None' (default), 'Open', or 'Short'. Use 'None' when StubMode is 'None'.
Thickness	Physical thickness of the microstrip	Meters. Default is 5.0e-6.
Width	Physical width of the parallel-plate	Meters. Default is 6.0e-4.
Z0	Characteristic impedance. Read-only; set by the analyze function.	Ohms. Default is [].

References

[1] Gupta, K.C., G. Ramesh, I. Bahl, and P. Bhartia, *Microstrip Lines and Slotlines*, Second Edition, Artech House, 1996. pp. 102-109.

See Also

analyze, calculate, getz0, listparam, listformat, plot, polar, rfckt, rfckt.coaxial, rfckt.cpw, rfckt.parallelplate, rfckt.rlcgline, rfckt.twowire, rfckt.txline, rfdata, smith, write

rfckt.mixer

Purpose Construct a two-port object that represents a mixer and its local oscillator

Syntax

```
h = rfckt.mixer
h = rfckt.mixer('Property1',value1,'Property2',value2,...)
```

Description `h = rfckt.mixer` returns a circuit object, `h`, whose properties are set to their default values.

`h = rfckt.mixer('Property1',value1,'Property2',value2,...)` returns a circuit object, `h`, that represents a mixer and its local oscillator (LO) with two ports (RF and IF). Properties you do not specify retain their default values.

Use the `read` method to read the mixer data from a Touchstone or AMP data file. See Appendix A, "AMP File Format" for information about the `.amp` format.

Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis After you create the `rfckt.mixer` circuit object, use the `analyze` function to calculate the S-parameters, output third-order intercept point, and noise figure at specified frequencies. For `rfckt.mixer` objects, `freq` must be nonnegative.

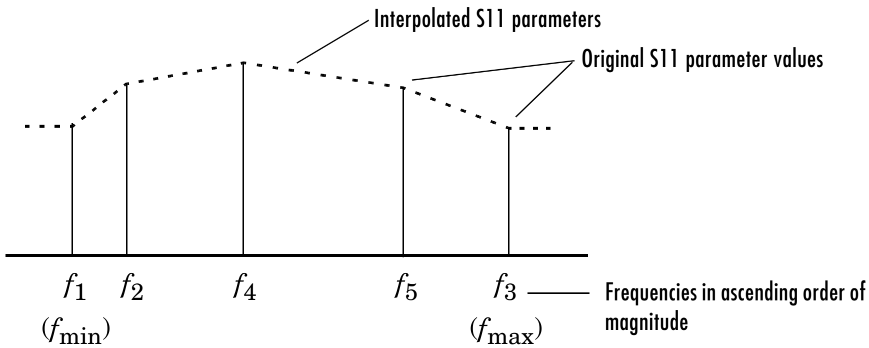
```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

If the `'NetworkData'` property of your `rfckt.mixer` object contains network Y- or Z-parameters, the `analyze` function first converts the parameters to S-parameters. Using the interpolation method you specify with the `'IntpType'` property, the `analyze` function interpolates the S-parameter values to determine the S-parameters at the specified frequencies.

Specifically, the `analyze` function orders the S-parameters according to the ascending order of their frequencies, f_n. It then interpolates the S-parameters, using the MATLAB `interp1` function. For example, the curve in the following diagram illustrates the result of interpolating the S11 parameters at five different frequencies.



You can specify the interpolation method as `cubic`, `linear` (default), or `spline`. For more information, see "One-Dimensional Interpolation" and the `interp1` reference page in the MATLAB documentation.

As shown in the diagram above, the `analyze` function uses the parameter values at f_{min}, the minimum input frequency, for all frequencies smaller than f_{min}. It uses the parameters values at f_{max}, the maximum input frequency, for all frequencies greater than f_{max}. In both cases, the results may not be accurate.

OIP3

The `analyze` function uses the data stored in the `'NonlinearData'` property of the `rfckt.mixer` object to calculate OIP3.

Noise Figure

The `analyze` function uses the data stored in the `'NoiseData'` property of the `rfckt.mixer` object to calculate the noise figure.

Properties

This table lists properties associated with `rfckt.mixer` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the mixer object	Handle. Default is`[1-by-1 rfdata.data]`.
`FLO`	Local oscillator frequency. For `MixerType = 'Downconverter'`, $f_{out} = f_{in} - f_{lo}$. For `MixerType = 'Upconverter'`, $f_{out} = f_{in} + f_{lo}$.	Hertz. Default is `1.0e+9`.
`FreqOffset`	Vector specifying the frequency offset for the phase noise level	Hertz. Default is `[]`.
`IntpType`	Interpolation method	String.`'Linear'` (default), `'Spline'`, or `'Cubic'`
`MixerType`	Type of mixer	String. `'Downconverter'` (default) or `'Upconverter'`
`Name`	Object name (read only)	String. `'Mixer'`
`NetworkData`	`rfdata.network` object	The default network parameters are taken from the `'default.amp'` data file.

Property	Description	Units, Values
NoiseData	Scalar noise figure in dB, rfdata.noise object, or rfdata.nf object	The default noise data values are taken from the 'default.s2p' data file and stored in an rfdata.noise object.
NonlinearData	Scalar OIP3 in dBm, rfdata.power object, or rfdata.ip3 object	The default is Inf.
nPort	Number of ports (read only)	Integer. The value is always 2.
PhaseNoiseLevel	Vector specifying the phase noise level	dBc/Hz. Default is [].

References

[1] EIA/IBIS Open Forum, “Touchstone File Format Specification,” Rev. 1.1, 2002 (http://www.eda.org/pub/ibis/connector/touchstone_spec11.pdf).

See Also

analyze, calculate, listparam, listformat, plot, polar, read, restore, rfckt, rfckt.amplifier, rfckt.datafile, rfckt.passive, rfdata, smith, write

rfckt.parallel

Purpose

Construct a parallel connected network object

Syntax

```
h = rfckt.parallel('Property1',value1,'Property2',value2,...)
h = rfckt.parallel
```

Description

`h = rfckt.parallel('Property1',value1,'Property2',value2,...)` returns a parallel connected network object, `h`, based on the specified properties. Use the `'Ckts'` property to specify the 2-port `rfckt` objects to be connected. Properties you do not specify retain their default values.

`h = rfckt.parallel` returns a parallel connected network object whose properties all have their default values.

Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis

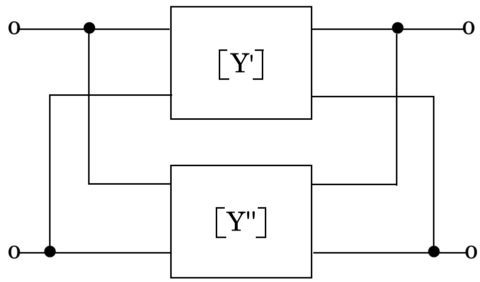
After you create the `parallel` network object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.parallel` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

The `analyze` function first calculates the admittance matrix of the parallel connected network. It starts by converting each component network's parameters to an admittance matrix. The figure shows a parallel connected network consisting of two 2-port networks, each represented by its admittance matrix.



where $\left[Y'\right] = \begin{bmatrix} Y_{11}' & Y_{12}' \\ Y_{21}' & Y_{22}' \end{bmatrix}$ and $\left[Y''\right] = \begin{bmatrix} Y_{11}'' & Y_{12}'' \\ Y_{21}'' & Y_{22}'' \end{bmatrix}$

The `analyze` function then calculates the admittance matrix for the parallel network by calculating the sum of the individual admittances. The following equation illustrates the calculations for two 2-port circuits.

$$\left[Y\right] = \left[Y'\right] + \left[Y''\right] = \begin{bmatrix} Y_{11}' + Y_{11}'' & Y_{12}' + Y_{12}'' \\ Y_{21}' + Y_{21}'' & Y_{22}' + Y_{22}'' \end{bmatrix}$$

Finally, `analyze` converts the admittance matrix of the parallel network to S-parameters at the frequencies specified in the `analyze` input argument `freq`.

rfckt.parallel

Properties

This table lists properties useful to `rfckt.parallel` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the parallel connected network object	Handle. Default is `[]`.
`Ckts`	Cell array containing all circuit objects in the network, in order from source to load. All circuits must be 2-port.	Handles to `rfckt` objects. Default is `{}`.
`Name`	Object name (read only)	String. `'Parallel Connected Network'`
`nPort`	Number of ports (read only)	Integer. The value is always `2`.

References

[1] Ludwig, Reinhold and Pavel Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

See Also

`analyze`, `calculate`, `listparam`, `listformat`, `plot`, `polar`, `rfckt`, `rfckt.cascade`, `rfckt.hybrid`, `rfckt.hybridg`, `rfckt.series`, `rfdata`, `smith`, `write`

Purpose

Construct a parallel-plate transmission line object

Syntax

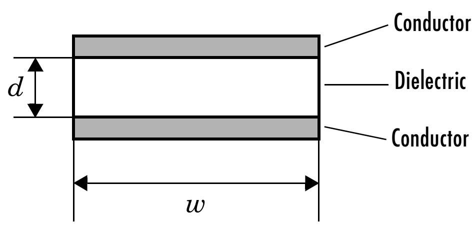
```
h = rfckt.parallelplate('Property1',value1,'Property2',value2,...)
h = rfckt.parallelplate
```

Description

`h = rfckt.parallelplate('Property1',value1,'Property2',value2,...)` returns a parallel-plate transmission line object, `h`, with the specified properties. Properties you do not specify retain their default values.

`h = rfckt.parallelplate` returns a parallel-plate transmission line object whose properties all have their default values.

A parallel-plate transmission line is shown here in cross-section. Its physical characteristics include the plate width w and the plate separation d.



Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis

After you create the `parallelplate` circuit object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.parallelplate` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

A parallel-plate transmission line object enables you to model the transmission line as a stub or as a stubless line.

Stubless Transmission Line. If you model the transmission line as a stubless line, the `analyze` function calculates the S-parameters for the specified frequencies, based on the physical length of the transmission line, D, and the complex propagation constant, k.

$$S_{11} = 0$$

$$S_{12} = e^{-kD}$$

$$S_{21} = e^{-kD}$$

$$S_{22} = 0$$

k is a vector whose elements correspond to the elements of the input vector `freq`. k can be expressed in terms of the resistance (R), inductance (L), conductance (G), and capacitance (C) per unit length (meters) as

$$k = k_r + jk_i = \sqrt{(R + j2\pi fL)(G + j2\pi fC)}$$

where f is the frequency range specified in the `analyze` input argument `freq`, and

$$R = \frac{2}{w\sigma_{\text{cond}}\delta}$$

$$L = \mu\frac{d}{w}$$

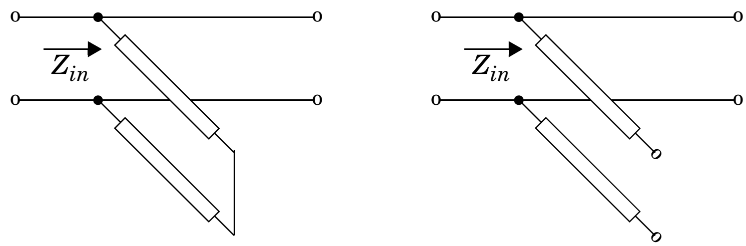
$$G = \sigma_{\text{diel}}\frac{w}{d}$$

$$C = \varepsilon\frac{w}{d}$$

In these equations, σ_{cond} is the conductivity in the conductor and σ_{diel} is the conductivity in the dielectric. μ is the relative permeability of the dielectric, ε is its permittivity as derived from the `EpsilonR` property, and skin depth δ is calculated as $1/\sqrt{\pi f\mu\sigma_{\text{cond}}}$.

Shunt and Series Stubs. If you model the transmission line as a shunt or series stub, the `analyze` function first calculates the ABCD-parameters at the specified frequencies. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

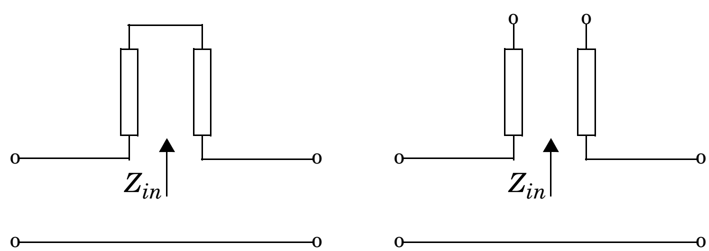
When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown here.



Z_{in} is the input impedance of the shunt circuit. The ABCD-parameters for the shunt stub are calculated as

$$A = 1$$
$$B = 0$$
$$C = 1/Z_{in}$$
$$D = 1$$

When you set the `StubMode` property to `'Series'`, the 2-port network consists of a series transmission line that you can terminate with either a short circuit or an open circuit as shown here.



Z_{in} is the input impedance of the series circuit. The ABCD-parameters for the series stub are calculated as

$$A = 1$$
$$B = Z_{in}$$
$$C = 0$$
$$D = 1$$

Properties

This table lists properties useful to `rfckt.parallelplate` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the parallel-plate transmission line object	Handle. Default is `[]`.
`EpsilonR`	Relative permittivity of the dielectric expressed as the ratio of the permittivity of the dielectric to permittivity in free space ε_0	Default is `2.3`.
`LineLength`	Physical length of the transmission line	Meters. Default is `0.01`.
`Loss`	Reduction in strength of the signal as it travels over the transmission line. Read-only; set by the `analyze` function.	Decibels per meter. Default is `[]`.
`MuR`	Relative permeability of the dielectric expressed as the ratio of the permeability of the dielectric to permeability in free space μ_0	Default is `1`.

Property	Description	Units, Values
Name	Object name (read only)	String. 'Parallel-Plate Transmission Line'
nPort	Number of ports (read only)	Integer. The value is always 2.
PV	Phase velocity. Propagation velocity of a uniform plane wave on the transmission line. Read-only; set by the analyze function.	Meters per second. Default is [].
Separation	Thickness of the dielectric separating the plates	Meters. Default is 1.0e-3.
SigmaCond	Conductivity in the conductor	Siemens per meter (S/m). Default is Inf.
SigmaDiel	Conductivity in the dielectric	Siemens per meter (S/m). Default is 0.
StubMode	Type of stub	String. 'None' (default), 'Series', or 'Shunt'
Termination	Termination for stub modes 'Shunt' and 'Series'.	String. 'None' (default), 'Open', or 'Short'. Use 'None' when StubMode is 'None'.
Width	Physical width of the parallel-plate transmission line	Meters. Default is .005.
Z0	Characteristic impedance. Read-only; set by the analyze function.	Ohms. Default is [].

References

[1] Ludwig, Reinhold and Pavel Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

See Also

`analyze`, `calculate`, `getz0`, `listparam`, `listformat`, `plot`, `polar`, `rfckt`, `rfckt.coaxial`, `rfckt.cpw`, `rfckt.microstrip`, `rfckt.rlcgline`, `rfckt.twowire`, `rfckt.txline`, `rfdata`, `smith`, `write`

Purpose

Construct a passive network object

Syntax

```
h = rfckt.passive('Property1',value1,'Property2',value2,...)
```

Description

`h = rfckt.passive('Property1',value1,'Property2',value2,...)` returns a passive circuit object, `h`, based on the specified properties. The properties include:

```
Name: 'Data File' (read only)
nPort: 2 (read only)
AnalyzedResult: Analyzed result (read only)
IntpType: 'Linear', 'Cubic' or 'Spline'
NetworkData: [1x1 rfdata.network]
```

`NetworkData` is an `rfdata.network` object. The default is the network parameters from `passive.s2p` data file.

Use the `read` method to read the passive network parameters from a Touchstone data file.

See Also

`analyze`, `calculate`, `listparam`, `listformat`, `plot`, `polar`, `read`, `restore`, `rfckt`, `rfckt.amplifier`, `rfckt.datafile`, `rfckt.mixer`, `rfdata`, `smith`, `write`

rfckt.rlcgline

Purpose Construct an RLCG transmission line object

Syntax

```
h = rfckt.rlcgline('Property1',value1,'Property2',value2, ...)
```

Description

`h = rfckt.rlcgline('Property1',value1,'Property2',value2, ...)` returns a RLCG transmission line object, `h`, based on the specified properties.

After you create the `rlcgline` circuit object, you can use the `analyze` function to calculate the network parameters and noise figure at the frequencies you pass into the `analyze` function. This function uses the interpolation method you specified in the `IntpType` property to find the R, L, C, and G values at these frequencies. Then, it calculates the characteristic impedance, Z0, phase velocity, PV, and loss using these interpolated values. For more information, see "Circuit Analysis" on page 4-140.

Properties

This table lists properties associated with `rfckt.rlcgline` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the RLCG transmission line object	Handle. Default is `[]`.
`C`	Vector of capacitance per length values that correspond to the frequencies stored in the `Freq` property	Farads/meter
`Freq`	Vector of positive frequency values	Hertz. Default is `[]`.
`G`	Vector of conductance per length values that correspond to the frequencies stored in the `Freq` property	Siemens/meter

Property	Description	Units, Values
IntpType	Interpolation method	'linear' (default), 'spline', or 'cubic'
L	Vector of inductance per length values that correspond to the frequencies stored in the Freq property	Henries/meter
LineLength	Scalar that represents the length of the transmission line	Meters. Default is 0.01.
Name	Object name (read only)	String. 'RLCG Transmission Line'
nPort	Number of ports (read only)	Integer. The value is always 2.
R	Vector of resistance per length values that correspond to the frequencies stored in the Freq property	Ohms/meter
StubMode	Type of stub	String. 'None' (default), 'Series', or 'Shunt'
Termination	Termination for stub modes 'Shunt' and 'Series'	String. 'None' (default), 'Open', or 'Short'. Use 'None' when StubMode is 'None'.

See Also

`analyze`, `calculate`, `getz0`, `listparam`, `listformat`, `plot`, `polar`, `rfckt`, `rfckt.coaxial`, `rfckt.cpw`, `rfckt.microstrip`, `rfckt.parallelplate`, `rfckt.twowire`, `rfckt.txline`, `rfdata`, `smith`, `write`

Purpose

Construct a series connected network object

Syntax

```
h = rfckt.series('Property1',value1,'Property2',value2,...)
h = rfckt.series
```

Description

`h = rfckt.series('Property1',value1,'Property2',value2,...)` returns a series connected network object, `h`, based on the specified properties. Use the `'Ckts'` property to specify the 2-port `rfckt` objects to be connected. Properties you do not specify retain their default values.

`h = rfckt.series` returns a series connected network object whose properties all have their default values.

Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis

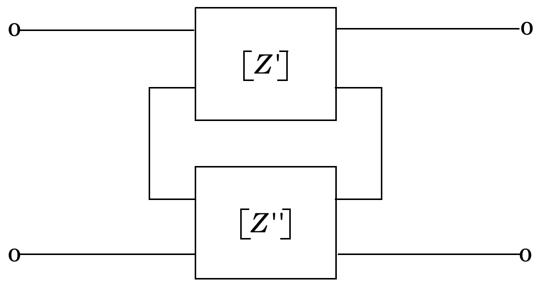
After you create the `series` network object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.series` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

The `analyze` function first calculates the impedance matrix of the series connected network. It starts by converting each component network's parameters to an impedance matrix. The figure shows a series connected network consisting of two 2-port networks, each represented by its impedance matrix.



where $[Z'] = \begin{bmatrix} Z_{11}' & Z_{12}' \\ Z_{21}' & Z_{22}' \end{bmatrix}$ and $[Z''] = \begin{bmatrix} Z_{11}'' & Z_{12}'' \\ Z_{21}'' & Z_{22}'' \end{bmatrix}$

The `analyze` function then calculates the impedance matrix for the series network by calculating the sum of the individual impedances. The following equation illustrates the calculations for two 2-port circuits.

$$[Z] = [Z'] + [Z''] = \begin{bmatrix} Z_{11}' + Z_{11}'' & Z_{12}' + Z_{12}'' \\ Z_{21}' + Z_{21}'' & Z_{22}' + Z_{22}'' \end{bmatrix}$$

Finally, `analyze` converts the impedance matrix of the series network to S-parameters at the frequencies specified in the analyze input argument `freq`.

Properties

This table lists properties useful to `rfckt.series` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the series connected network object	Handle. Default is [].
`Ckts`	Cell array containing all circuit objects in the network, in order from source to load. All circuits must be 2-port	Handles to `rfckt` objects. Default is `{}`.
`Name`	Object name (read only)	String. `'Series Connected Network'`
`nPort`	Number of ports (read only)	Integer. The value is always `2`.

References

[1] Ludwig, Reinhold and Pavel Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

See Also

`analyze`, `calculate`, `listparam`, `listformat`, `plot`, `polar`, `rfckt`, `rfckt.cascade`, `rfckt.hybrid`, `rfckt.hybridg`, `rfckt.parallel`, `rfdata`, `smith`, `write`

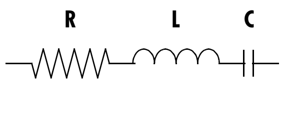
rfckt.seriesrlc

Purpose Construct a series RLC network object

Syntax

```
h = rfckt.seriesrlc('R',Rvalue,'L',Lvalue,'C',Cvalue)
h = rfckt.seriesrlc
```

Description The series RLC network object is a two-port network as shown in the circuit diagram below.



`h = rfckt.seriesrlc('R',Rvalue,'L',Lvalue,'C',Cvalue)` returns a series RLC network object, `h`, based on the specified resistance (R), inductance (L), and capacitance (C) values. Properties you do not specify retain their default values, allowing you to specify a network of a single resistor, inductor, or capacitor.

`h = rfckt.seriesrlc` returns a series RLC network object whose properties all have their default values. This is equivalent to a pass-through two port network, i.e., the resistor, inductor, and capacitor are each replaced by a short circuit.

Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis After you create the `seriesrlc` circuit object, use the `analyze` function to calculate the S-parameters and noise correlation matrix at specified frequencies. For `rfckt.seriesrlc` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

The `analyze` function first calculates the ABCD-parameters for the circuit, then converts the ABCD-parameters to S-parameters using the `abcd2s` function. For this circuit, A = 1, B = Z, C = 0, and D = 1, where

$$Z = \frac{-LC\omega^2 + jRC\omega + 1}{jC\omega}$$

where $\omega = 2\pi f$.

Properties

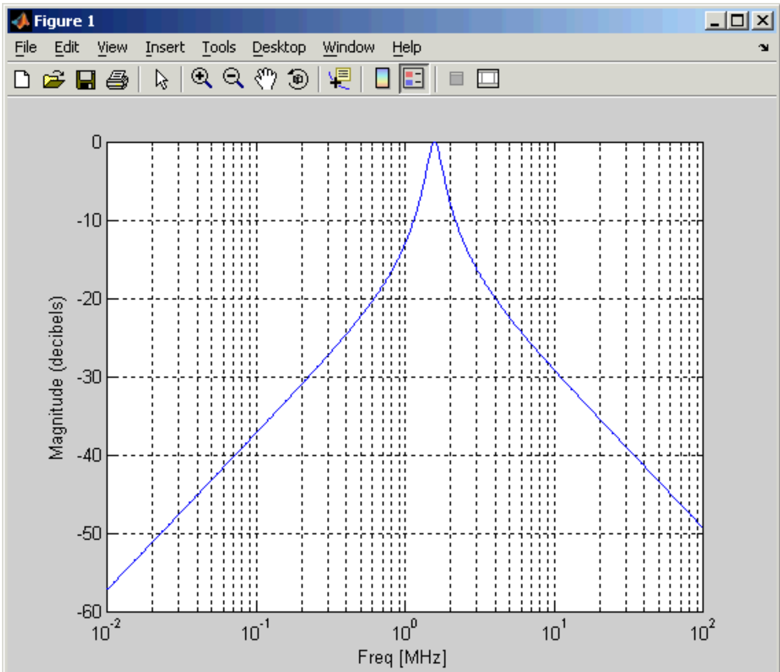
This table lists properties useful to `rfckt.seriesrlc` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the series RLC network object	Handle. Default is `[]`.
`C`	Scalar value for the capacitance	Farads. Default is `Inf`.
`L`	Scalar value for the inductance	Henries. Default is `0`.
`Name`	Object name (read only)	String, `'Series RLC'`.
`nPort`	Number of ports (read only)	Integer. The value is always `2`.
`R`	Scalar value for the resistance	Ohms. Default is `0`.

Examples

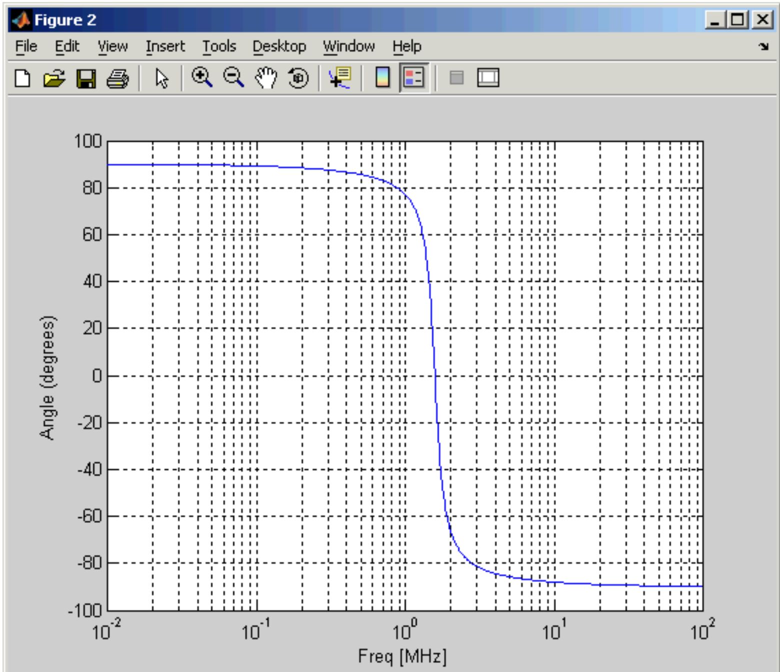
This example creates a series LC resonator and examines its frequency response. It first creates the circuit object then uses the `analyze` function to calculate its frequency response. Finally, it plots the results – first, the magnitude in decibels (dB).

```
h = rfckt.seriesrlc('L',4.7e-5,'C',2.2e-10);
analyze(h,logspace(4,8,1000));
plot(h,'s21','dB')
set(gca,'Xscale','log')
```



The example then plots the phase, in degrees

```
figure
plot(h,'s21','angle')
set(gca,'Xscale','log')
```



References

[1] Ludwig, Reinhold and Pavel Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

See Also

`analyze`, `calculate`, `listparam`, `listformat`, `plot`, `polar`, `rfckt`, `rfckt.shuntrlc`, `rfdata`, `smith`, `write`

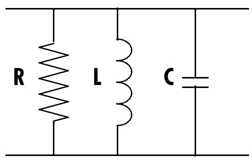
rfckt.shuntrlc

Purpose Construct a shunt RLC network object

Syntax

```
h = rfckt.shuntrlc('R',Rvalue,'L',Lvalue,'C',Cvalue)
h = rfckt.shuntrlc
```

Description The shunt RLC network object is a two-port network as shown in the circuit diagram below.



`h = rfckt.shuntrlc('R',Rvalue,'L',Lvalue,'C',Cvalue)` returns a shunt RLC network object, `h`, based on the specified resistance (R), inductance (L), and capacitance (C) values. Properties you do not specify retain their default values, allowing you to specify a network of a single resistor, inductor, or capacitor.

`h = rfckt.shuntrlc` returns a shunt RLC network object whose properties all have their default values. This is equivalent to a pass-through two port network, i.e., the resistor, inductor, and capacitor are each replaced by an open circuit.

Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis After you create the `shuntrlc` circuit object, use the `analyze` function to calculate the S-parameters and noise correlation matrix at specified frequencies. For `rfckt.shuntrlc` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

The `analyze` function first calculates the ABCD-parameters for the circuit, then converts the ABCD-parameters to S-parameters using the `abcd2s` function. For this circuit, A = 1, B = 0, C = Y, and D = 1, where

$$Y = \frac{-LC\omega^2 + j(L/R)\omega + 1}{jL\omega}$$

and $\omega = 2\omega f$.

Properties

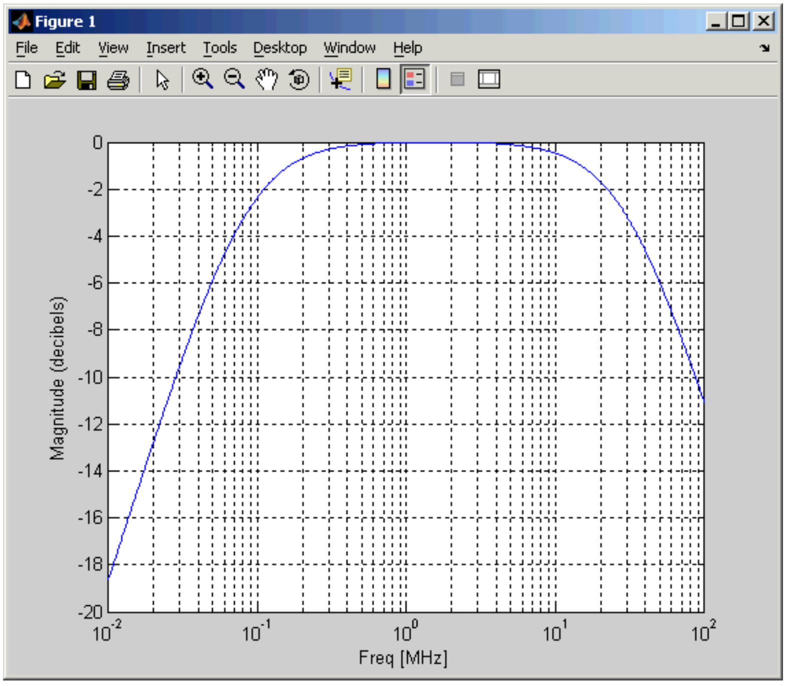
This table lists properties useful to `rfckt.shuntrlc` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the shunt RLC network object	Handle. Default is `[]`.
`C`	Scalar value for the capacitance	Farads. Default is `0`.
`L`	Scalar value for the inductance	Henries. Default is `Inf`.
`Name`	Object name (read only)	String. `'Shunt RLC'`.
`nPort`	Number of ports (read only)	Integer. The value is always `2`.
`R`	Scalar value for the resistance	Ohms. Default is `Inf`.

Examples

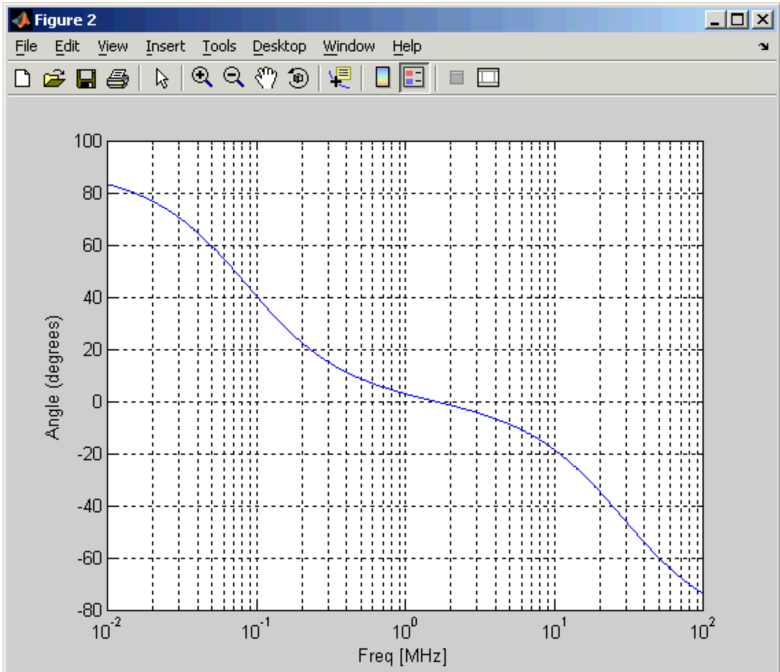
This example creates a shunt LC resonator and examines its frequency response. It first creates the circuit object then uses the `analyze` function to calculate its frequency response. Finally, it plots the results – first, the magnitude in decibels (dB).

```
h = rfckt.shuntrlc('L',4.7e-5,'C',2.2e-10);
analyze(h,logspace(4,8,1000));
plot(h,'s21','dB')
set(gca,'Xscale','log')
```



The example then plots the phase, in degrees

```
figure
plot(h,'s21','angle')
set(gca,'Xscale','log')
```



References

[1] Ludwig, Reinhold and Pavel Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

See Also

`analyze`, `calculate`, `listparam`, `listformat`, `plot`, `polar`, `rfckt`, `rfckt.seriesrlc`, `rfdata`, `smith`, `write`

rfckt.twowire

Purpose Construct a two-wire transmission line object

Syntax

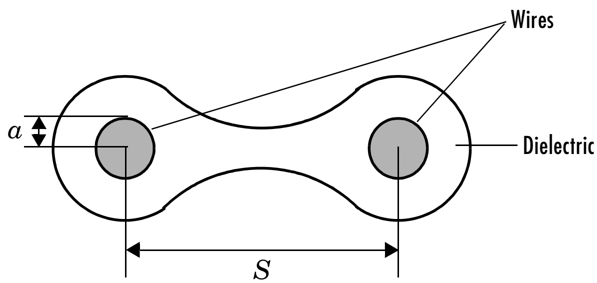
```
h = rfckt.twowire('Property1',value1,'Property2',value2,...)
h = rfckt.twowire
```

Description

`h = rfckt.twowire('Property1',value1,'Property2',value2,...)` returns a two-wire transmission line object, `h`, with the specified properties. Properties you do not specify retain their default values.

`h = rfckt.twowire` returns a two-wire transmission line object whose properties all have their default values.

A two-wire transmission line is shown here in cross-section. Its physical characteristics include the radius of the wires a , and the separation or physical distance between the wire centers S .



Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis

After you create the `twowire` circuit object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.twowire` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

A two-wire transmission line object enables you to model the transmission line as a stub or as a stubless line.

Stubless Transmission Line. If you model the transmission line as a stubless line, the `analyze` function calculates the S-parameters for the specified frequencies, based on the physical length of the transmission line, D , and the complex propagation constant, k .

$$S_{11} = 0$$

$$S_{12} = e^{-kD}$$

$$S_{21} = e^{-kD}$$

$$S_{22} = 0$$

k is a vector whose elements correspond to the elements of the input vector `freq`. k can be expressed in terms of the resistance (R), inductance (L), conductance (G), and capacitance (C) per unit length (meters) as

$$k = k_r + jk_i = \sqrt{(R + j2\pi fL)(G + j2\pi fC)}$$

where f is the frequency range specified in the `analyze` input argument `freq`, and

$$R = \frac{1}{\pi a \sigma_{\text{cond}} \delta}$$

$$L = \frac{\mu}{\pi} \operatorname{acosh}\left(\frac{D}{2a}\right)$$

$$G = \frac{\pi \sigma_{\text{diel}}}{\operatorname{acosh}(D/(2a))}$$

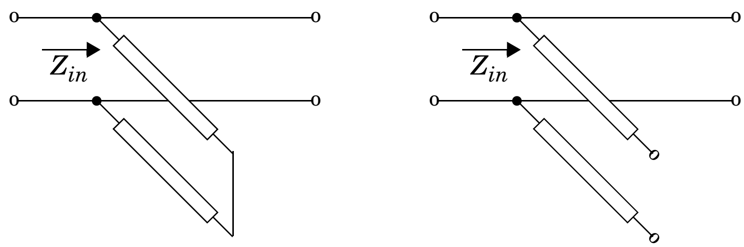
$$C = \frac{\pi \varepsilon}{\operatorname{acosh}(D/(2a))}$$

In these equations, σ_{cond} is the conductivity in the conductor and σ_{diel} is the conductivity in the dielectric. μ is the relative permeability of the dielectric,

ε is its permittivity as derived from the EpsilonR property, and skin depth δ is calculated as $1/\sqrt{\pi f \mu \sigma_{cond}}$.

Shunt and Series Stubs. If you model the transmission line as a shunt or series stub, the analyze function first calculates the ABCD-parameters at the specified frequencies. It then uses the abcd2s function to convert the ABCD-parameters to S-parameters.

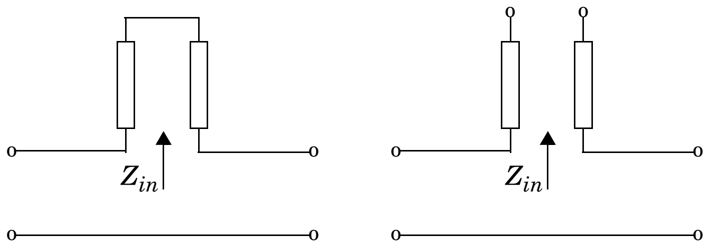
When you set the StubMode property to 'Shunt', the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown here.



Z_{in} is the input impedance of the shunt circuit. The ABCD-parameters for the shunt stub are calculated as

$$A = 1$$
$$B = 0$$
$$C = 1/Z_{in}$$
$$D = 1$$

When you set the StubMode property to 'Series', the 2-port network consists of a series transmission line that you can terminate with either a short circuit or an open circuit as shown here.



Z_{in} is the input impedance of the series circuit. The ABCD-parameters for the series stub are calculated as

$$A = 1$$
$$B = Z_{in}$$
$$C = 0$$
$$D = 1$$

Properties

This table lists properties useful to `rfckt.twowire` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the two-wire transmission line object	Handle. Default is `[]`.
`EpsilonR`	Relative permittivity of the dielectric expressed as the ratio of the permittivity of the dielectric to permittivity in free space ε_0	Default is `2.3`.
`LineLength`	Physical length of the transmission line	Meters. Default is `0.01`.

Property	Description	Units, Values
Loss	Reduction in strength of the signal as it travels over the transmission line. Read-only; set by the analyze function.	Decibels per meter. Default is [].
MuR	Relative permeability of the dielectric expressed as the ratio of the permeability of the dielectric to permeability in free space μ_0	Default is 1.
Name	Object name (read only)	String. 'Two-Wire Transmission Line'
nPort	Number of ports (read only)	Integer. The value is always 2.
PV	Phase velocity. Propagation velocity of a uniform plane wave on the transmission line. Read-only; set by the analyze function.	Meters per second. Default is [].
Radius	Radius of the conducting wires	Meters. Default is 6.7e-4.
Separation	Physical distance between the wires	Meters. Default is 0.0016.
SigmaCond	Conductivity in conductor	Siemens per meter (S/m). Default is Inf.
SigmaDiel	Conductivity in dielectric	Siemens per meter (S/m). Default is 0.

Property	Description	Units, Values
`StubMode`	Type of stub	String. `'None'` (default), `'Series'`, or `'Shunt'`
`Termination`	Termination for stub modes `'Shunt'` and `'Series'`.	String. `'None'` (default), `'Open'`, or `'Short'`. Use `'None'` when `StubMode` is `'None'`.
`Z0`	Characteristic impedance. Read-only; set by the `analyze` function.	Ohms. Default is `[]`.

References

[1] Ludwig, Reinhold and Pavel Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

See Also

`analyze`, `calculate`, `getz0`, `listparam`, `listformat`, `plot`, `polar`, `rfckt`, `rfckt.coaxial`, `rfckt.cpw`, `rfckt.microstrip`, `rfckt.parallelplate`, `rfckt.rlcgline`, `rfckt.txline`, `rfdata`, `smith`, `write`

rfckt.txline

Purpose Construct a transmission line object

Syntax

```
h = rfckt.txline
h = rfckt.txline('Property1',value1,'Property2',value2,...)
```

Description

`h = rfckt.txline` returns a transmission line object whose properties are set to their default values.

`h = rfckt.txline('Property1',value1,'Property2',value2,...)` returns a transmission line object, `h`, with the specified properties. Properties you do not specify retain their default values.

Note See the `rfckt` reference page for a list of functions that act on circuit (`rfckt`) objects.

Circuit Analysis

After you create the `txline` circuit object, use the `analyze` function to calculate the S-parameters and noise figure at specified frequencies. For `rfckt.txline` objects, `freq` must be strictly positive.

```
analyze(h,freq)
```

The `analyze` function stores the results of the analysis in the `AnalyzedResult` property of the circuit object.

Network Parameters

A general transmission line object enables you to model the transmission line as a stub or as a stubless line. The transmission line, which can be lossy or lossless, is treated as a 2-port linear network.

Stubless Transmission Line. If you model the transmission line as a stubless line, the `analyze` function calculates the S-parameters for the specified frequencies, based on the physical length of the transmission line, D, and the complex propagation constant, k.

$$S_{11} = 0$$

$$S_{12} = e^{-kD}$$

$$S_{21} = e^{-kD}$$

$$S_{22} = 0$$

k is a vector whose elements correspond to the elements of the input vector `freq`. $k = \alpha_a + i\beta$, where α_a is the attenuation coefficient and β is the wave number. The attenuation coefficient α_a is related to the loss, α , by

$$\alpha_a = -\ln 10^{-\frac{\alpha}{20}}$$

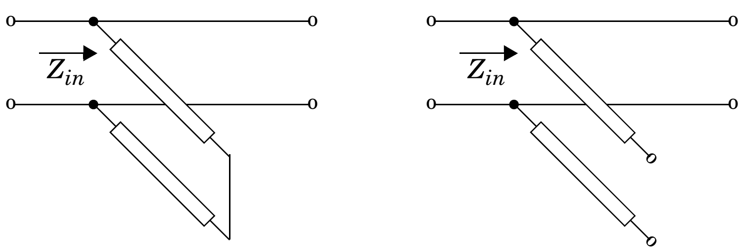
and the wave number β is related to the phase velocity, V_P , by

$$\beta = \frac{2\pi f}{V_p}$$

where f is the frequency range specified in the `analyze` input argument `freq`. The phase velocity V_p is derived from the `rfckt.txline` object properties. It is also known as the wave propagation velocity.

Shunt and Series Stubs. If you model the transmission line as a shunt or series stub, the `analyze` function first calculates the ABCD-parameters at the specified frequencies. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

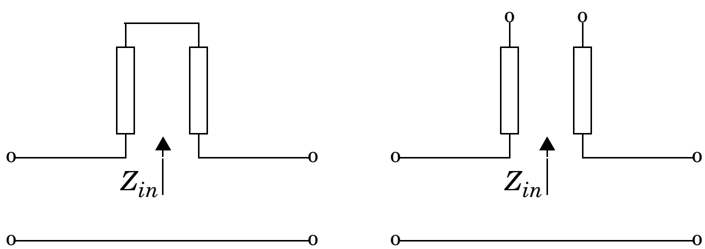
When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown here.



Z_{in} is the input impedance of the shunt circuit. The ABCD-parameters for the shunt stub are calculated as

$$A = 1$$
$$B = 0$$
$$C = 1/Z_{in}$$
$$D = 1$$

When you set the `StubMode` property to `'Series'`, the 2-port network consists of a series transmission line that you can terminate with either a short circuit or an open circuit as shown here.



Z_{in} is the input impedance of the series circuit. The ABCD-parameters for the series stub are calculated as

$$A = 1$$
$$B = Z_{in}$$
$$C = 0$$
$$D = 1$$

Properties

This table lists properties associated with `rfckt.txline` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`AnalyzedResult`	`rfdata.data` object that contains the result of applying the `analyze` function to the transmission line object	Handle. Default is `[]`.
`Freq`	Vector of positive frequencies at which the parameter values are known.	Hertz. Default is [].
`IntpType`	Interpolation method	`'linear'` (default), `'spline'`, or `'cubic'`
`LineLength`	Scalar that represents the physical length of the transmission line	Meters. Default is `0.01`.
`Loss`	Vector of line loss values that correspond to the frequencies stored in the `Freq` property. Line loss is the reduction in strength of the signal as it travels over the transmission line.	Decibels per meter. Must be positive. Default is `0`.
`Name`	Object name (read only)	String. `'Transmission Line'`
`nPort`	Number of ports (read only)	Integer. The value is always `2`.

Property	Description	Units, Values
PV	Vector of phase velocity values that correspond to the frequencies stored in the Freq property. Propagation velocity of a uniform plane wave on the transmission line	Meters per second. Default is 299792458.
StubMode	Type of stub	String. 'None' (default), 'Series', or 'Shunt'
Termination	Termination for 'Shunt'and 'Series' stub modes.	String. 'None' (default), 'Open', or 'Short'. Use 'None' when StubMode is 'None'.
Z0	Vector of characteristic impedance values that correspond to the frequencies stored in the Freq property	Ohms. Default is 50.

References

[1] Ludwig, Reinhold and Pavel Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

See Also

analyze, calculate, getz0, listparam, listformat, plot, polar, rfckt, rfckt.coaxial, rfckt.cpw, rfckt.microstrip, rfckt.parallelplate, rfckt.rlcgline, rfckt.twowire, rfdata, smith, write

Purpose Construct an RF data object

Description An `rfdata` object contains network parameter data. Only the `read` and `analyze` functions can create an `rfdata` object.

See the individual `rfdata` object reference pages for information about a specific data object and its properties. See Chapter 2, "Working with RF Objects," for additional information.

Objects The following table lists the available objects.

rfdata.type	Description
`rfdata.data`	Data object containing network parameter data
`rfdata.ip3`	Data object containing IP3 information
`rfdata.network`	Data object containing network parameter information
`rfdata.nf`	Data object containing noise figure information
`rfdata.noise`	Data object containing noise information
`rfdata.power`	Data object containing power and phase information

Functions The following table lists the functions that act on data objects and tells you the types of objects on which each can act. These functions are also referred to as methods.

Function	Types of Objects	Purpose
`copy`	All data objects	Copy a data object
`extract`	`rfdata.data`, `rfdata.network`	Extract the specified network parameters from a data object and return the result in a matrix

Function	Types of Objects	Purpose
`read`	`rfdata.data`	Read RF data parameters from a file to a new or existing data object.
`write`	`rfdata.data`	Write RF data from a data object to a file.

Properties

Properties vary for each type of object. See the individual object reference pages for information about properties.

Viewing Object Properties

You can use `get` to view an `rfdata` object's properties. To see a specific property, use

```
get(h,'PropertyName')
```

To see all properties for an object, use

```
get(h)
```

Changing Object Properties

To see the `rfdata` properties whose values you can change use

```
set(h)
```

To change specific properties, use

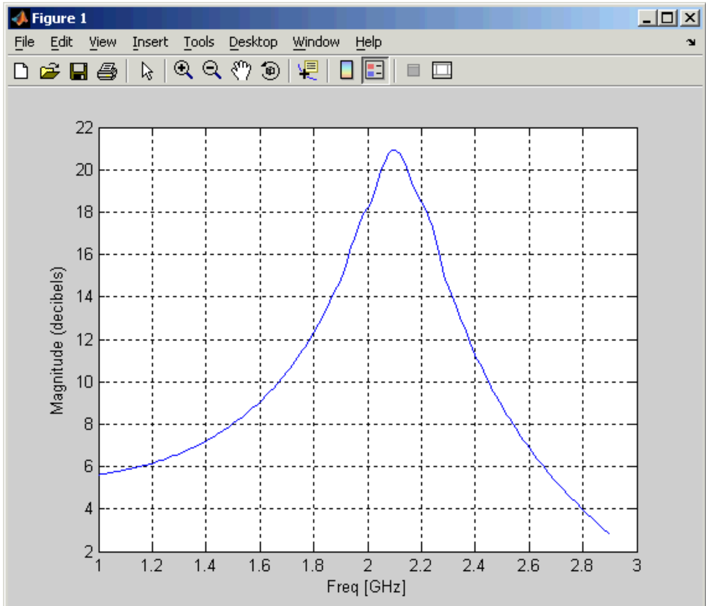
```
set(h,'PropertyName1',value1,'PropertyName2',value2,...)
```

Note that you must use single quotation marks around the property name.

Examples

Construct an RF data object from a `.s2p` data file.

```
file = 'default.s2p';
h = read(rfdata.data,file);  % Read file into rfdata.data object.
figure
plot(h,'s21','db');   % Plot dB(S21) in XY plane.
```



You can also use other RF Toolbox functions such as `polar` and `smith` to visualize results.

See Also

`analyze`, `calculate`, `copy`, `extract`, `listformat`, `listparam`, `plot`, `polar`, `read`, `rfckt`, `smith`, `write`

rfdata.data

Purpose

Store result of circuit object analysis

Description

An `rfdata.data` object contains the result of analyzing a circuit object. This result includes the S-parameters, noise figure in dB, and frequency-dependent third order output (OIP3) intercept points.

- `read` reads network parameters from a data file and writes those parameters to an `rfdata.data` object.
- `rfckt/analyze` stores the results of its analysis in an `rfdata.data` object.

Note See the `rfdata` reference page for a list of functions that act on `rfdata.data` objects.

Use `get` and `set` to view and change `rfdata.data` object properties. To see a specific property, use

```
get(h,'PropertyName')
```

To change specific properties, use

```
set(h,'PropertyName1',value1,'PropertyName2',value2,...)
```

Properties

This table lists properties useful to `rfdata.data` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`Freq`	Frequencies of the S-parameters as an M-element vector. The order of the frequencies must correspond to the order of `'S-parameters'`. All frequencies must be positive.	Hertz. Default is `[]`.
`IntpType`	Interpolation method	`'linear'` (default), `'spline'`, or `'cubic'`

Property	Description	Units, Values
Name	Object name (read only)	String. 'rfdata.data object'
NF	Noise figure. The amount of noise relative to a noise temperature of 290 degrees kelvin. 0 indicates a noiseless system	Decibels. Default is 0.
OIP3	Output third-order intercept point	Watts. Default is Inf.
S_Parameters	S-parameters of the circuit described by the rfdata.data object in a 2-by-2-by-M array. M is the number of S-parameters.	Default is [].
Z0	Reference impedance	Ohms. Default is 50.
ZL	Load impedance	Ohms. Default is 50.
ZS	Source impedance	Ohms. Default is 50.

See Also

extract, read, rfdata, rfdata.ip3, rfdata.network, rfdata.nf, rfdata.noise, rfdata.power, rfckt, write

rfdata.ip3

Purpose Store frequency-dependent, third-order intercept points for amplifiers or mixers

Syntax

```
h = rfdata.ip3('Type',value1,'Freq',value2,'Data',value3)
```

Description `h = rfdata.ip3('Type',value1,'Freq',value2,'Data',value3)` returns a data object for the frequency-dependent IP3, `h`, based on the specified properties.

Properties This table lists the properties associated with `rfdata.ip3` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`Data`	Vector of IP3 data that corresponds to the frequencies stored in the `Freq` property	Watts. Default is `[]`.
`Freq`	Vector of positive frequency values	Hertz. Default is `[]`.
`Name`	Object name (read only)	String. `'3rd order intercept'`
`Type`	Type of IP3	String. `'OIP3'` or `'IIP3'`

See Also `rfdata`, `rfdata.data`, `rfdata.network`, `rfdata.nf`, `rfdata.noise`, `rfdata.power`, `rfckt`

Purpose

Store frequency-dependent network parameters

Syntax

```
h =
  rfdata.network('Type',value1,'Freq',value2,'Data',value3,'ZO',
  value4)
```

Description

`h = rfdata.network('Type',value1,'Freq',value2,'Data',value3,'ZO',value4)` returns a data object for the frequency-dependent network parameters, `h`, based on the specified properties.

Properties

This table lists the properties associated with `rfdata.network` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`Data`	Matrix of network parameters that correspond to the frequencies stored in the `Freq` property	Default is `[]`.
`Freq`	Vector of positive frequency values	Hertz. Default is `[]`.
`Name`	Object name (read only)	String. `'Network parameters'`
`Type`	Type of network parameters	String. `'S'`, `'Y'`, `'Z'`, `'H'`, `'G'`, or `'T'`
`ZO`	Scalar reference impedance. This property is only available when the `Type` property is set to `'S'`.	Default is `[]`.

See Also

`rfckt`, `rfdata`, `rfdata.data`, `rfdata.ip3`, `rfdata.nf`, `rfdata.noise`, `rfdata.power`,

rfdata.nf

Purpose

Store frequency-dependent noise figure data for amplifiers or mixers

Syntax

```
h = rfdata.nf('Freq',value1,'Data',value2)
```

Description

`h = rfdata.nf('Freq',value1,'Data',value2)` returns a data object for the frequency-dependent noise figure, `h`, based on the specified properties.

Properties

This table lists the properties associated with `rfdata.nf` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`Data`	Vector of noise figure values that correspond to the frequencies stored in the `Freq` property	Decibels. Default is `[]`.
`Freq`	Vector of positive frequency values	Hertz. Default is `[]`.
`Name`	Object name (read only)	String. `'Noise figure'`

See Also

`rfckt`, `rfdata`, `rfdata.data`, `rfdata.ip3`, `rfdata.network`, `rfdata.noise`, `rfdata.power`

Purpose

Store frequency-dependent spot noise data for amplifiers or mixers

Syntax

```
h =
   rfdata.noise('Freq',value1,'FMIN',value2,'GAMMAOPT',value3,'RN',
     value4)
```

Description

`h = rfdata.noise('Freq',value1,'FMIN',value2,'GAMMAOPT',value3,'RN',value4)` returns a data object for the frequency-dependent spot noise, `h`, based on the specified properties.

Properties

This table lists the properties associated with `rfdata.noise` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`FMIN`	Vector of minimum noise figure values that correspond to the frequencies stored in the `Freq` property	Decibels. Default is `[]`.
`Freq`	Vector of positive frequency values.	Hertz. Default is `[]`.
`GAMMAOPT`	Vector of optimum source reflection coefficients that correspond to the frequencies stored in the `Freq` property	Default is `[]`.
`Name`	Object name (read only)	String. `'Spot noise data'`
`RN`	Vector of equivalent normalized noise resistance values that correspond to the frequencies stored in the `Freq` property	Default is `[]`.

See Also

`rfckt`, `rfdata`, `rfdata.data`, `rfdata.ip3`, `rfdata.network`, `rfdata.nf`, `rfdata.power`

rfdata.power

Purpose

Store output power and phase information for amplifiers or mixers

Syntax

```
h = rfdata.power(`property1',value1,'property2',value2,...)
```

Description

`h = rfdata.power(`property1',value1,'property2',value2,...)` returns a data object for the Pin/Pout power data, `h`, based on the specified properties.

Properties

This table lists the properties associated with `rfdata.power` objects along with units, valid values, and property descriptions.

Property	Description	Units, Values
`Freq`	Vector of positive frequency values	Hertz. Default is `[]`.
`Name`	Object name (read only)	String. `'Power data'`
`Phase`	Vector of phase shift values that correspond to the frequencies stored in the `Freq` property	Degrees. Default is `[]`.
`Pin`	Cell array of input power values. For example, `Pin = {[A]; [B]; [C]};` where `A`, `B`, and `C` are column vectors that contain the `Pin` values at the first three frequencies stored in the `Freq` property.	Watts. Default is `[]`.
`Pout`	Cell array of output power values	Watts. Default is `[]`.

See Also

`rfckt`, `rfdata`, `rfdata.data`, `rfdata.ip3`, `rfdata.network`, `rfdata.nf`, `rfdata.noise`

Purpose

Open the RF Analysis Tool (RFTool)

Syntax

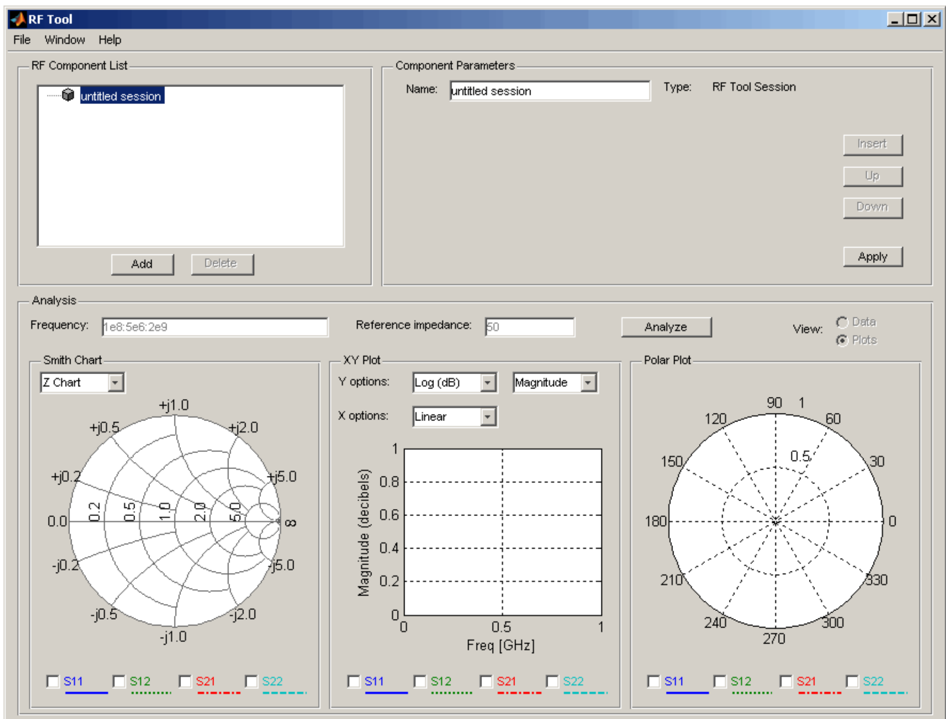
rftool

Description

`rftool` opens RFTool. Use this tool to:

- Create circuit components and set their parameters
- Analyze components over a specified frequency range and step size
- Plot the analysis results
- Import component objects to and export them from the MATLAB workspace
- Save RFTool sessions for later use

See Chapter 3, "RF Tool: An RF Analysis GUI" for more information



s2abcd

Purpose Convert S-parameters to ABCD-parameters

Syntax

```
abcd_params = s2abcd(s_params,z0)
```

Description

abcd_params = s2abcd(s_params,z0) converts the scattering parameters s_params into the ABCD parameters abcd_params. The s_params input is a complex 2-by-2-by-m array, representing m two-port S-parameters. z0 is the reference impedance; its default is 50 ohms. abcd_params is a complex 2-by-2-by-m array, representing m two-port ABCD-parameters.

See Also abcd2s, h2abcd, s2y, s2z, s2h, y2abcd, z2abcd

Purpose Convert S-parameters to hybrid h-parameters

Syntax

```
h_params = s2h(s_params,z0)
```

Description `h_params = s2h(s_params,z0)` converts the scattering parameters `s_params` into the hybrid parameters `h_params`. The `s_params` input is a complex 2-by-2-by-`m` array, representing `m` two-port S-parameters. `z0` is the reference impedance; its default is 50 ohms. `h_params` is a complex 2-by-2-by-`m` array, representing `m` two-port hybrid h-parameters.

See Also `abcd2h`, `h2s`, `s2abcd`, `s2y`, `s2z`, `y2h`, `z2h`

Purpose Convert S-parameters to S-parameters with different impedance

Syntax

```
s_params_new = s2s(s_params,z0)
s_params_new = s2s(s_params,z0,z0_new)
```

Description

`s_params_new = s2s(s_params,z0)` converts the scattering parameters `s_params` with reference impedance `z0` into the scattering parameters `s_params_new` with reference impedance 50 ohms. `s_params_new` is a complex `n`-by-`n`-by-`m` array, representing `m` `n`-port S-parameters. `s_params` is a complex `n`-by-`n`-by-`m` array, representing `m` `n`-port S-parameters. `z0` is the reference impedance of the input S-parameters.

`s_params_new = s2s(s_params,z0,z0_new)` converts the scattering parameters `s_params` with reference impedance `z0` into the scattering parameters `s_params_new` with the reference impedance `z0_new`.

See Also `abcd2s`, `h2s`, `s2abcd`, `s2h`, `s2y`, `s2z`, `y2s`, `z2s`

Purpose Convert 4-port S-parameters to 2-port common mode S-parameters (S_{cc})

Syntax

```
scc_params = s2scc(s_params)
```

Description `scc_params = s2scc(s_params)` converts the 4-port, single-ended S-parameters, `s_params`, to 2-port, common mode S-parameters, `scc_params`. `scc_params` is a complex 2-by-2-by-M array that represents M 2-port S-parameters. `s_params` is a complex 4-by-4-by-M array that represents M 4-port S-parameters.

Reference W. Fan, A. C. W. Lu, L. L. Wai and B. K. Lok. "Mixed-Mode S-Parameter Characterisation of Differential Structures." Electronic Packaging Technology Conference, pp. 533-537, 2003.

See Also `s2scd`, `s2sdc`, `s2sdd`

s2scd

Purpose Convert 4-port S-parameters to 2-port cross mode S-parameters (S_{cd})

Syntax

```
scd_params = s2scd(s_params)
```

Description `scd_params = s2scd(s_params)` converts the 4-port, single-ended S-parameters, `s_params`, to 2-port, cross mode S-parameters, `scd_params`. `scd_params` is a complex 2-by-2-by-M array that represents M 2-port cross mode S-parameters (S_{cd}). `s_params` is a complex 4-by-4-by-M array that represents M 4-port S-parameters.

Reference W. Fan, A. C. W. Lu, L. L. Wai and B. K. Lok. “Mixed-Mode S-Parameter Characterisation of Differential Structures.” Electronic Packaging Technology Conference, pp. 533-537, 2003.

See Also `s2scc`, `s2sdc`, `s2sdd`

Purpose Convert 4-port S-parameters to 2-port cross mode S-parameters (S_{dc})

Syntax

```
sdc_params = s2sdc(s_params)
```

Description `sdc_params = s2sdc(s_params)` converts the 4-port, single-ended S-parameters, `s_params`, to 2-port, cross mode S-parameters, `sdc_params`. `sdc_params` is a complex 2-by-2-by-M array that represents M 2-port cross mode S-parameters (S_{dc}). `s_params` is a complex 4-by-4-by-M array that represents M 4-port S-parameters.

Reference W. Fan, A. C. W. Lu, L. L. Wai and B. K. Lok. “Mixed-Mode S-Parameter Characterisation of Differential Structures.” Electronic Packaging Technology Conference, pp. 533-537, 2003.

See Also `s2scc`, `s2scd`, `s2sdd`

s2sdd

Purpose Convert 4-port S-parameters to 2-port differential mode S-parameters (S_{dd})

Syntax

```
sdd_params = s2sdd(s_params)
```

Description `sdd_params = s2sdd(s_params)` converts the 4-port, single-ended S-parameters, `s_params`, to 2-port, differential mode S-parameters, `sdd_params`. `sdd_params` is a complex 2-by-2-by-M array that represents M 2-port differential mode S-parameters. `s_params` is a complex 4-by-4-by-M array that represents M 4-port S-parameters.

Reference W. Fan, A. C. W. Lu, L. L. Wai and B. K. Lok. “Mixed-Mode S-Parameter Characterisation of Differential Structures.” Electronic Packaging Technology Conference, pp. 533-537, 2003.

See Also `s2scc`, `s2scd`, `s2sdc`

Purpose Convert S-parameters to T-parameters

Syntax

```
t_params = s2t(s_params)
```

Description `t_params = s2t(s_params)` converts the scattering parameters `s_params` into the chain scattering parameters `t_params`. The `s_params` input is a complex 2-by-2-by-`m` array, representing `m` two-port S-parameters. `t_params` is a complex 2-by-2-by-`m` array, representing `m` two-port T-parameters.

See Also `s2abcd`, `s2h`, `s2y`, `s2z`, `t2s`

s2y

Purpose

Convert S-parameters to Y-parameters

Syntax

```
y_params = s2y(s_params,z0)
```

Description

`y_params = s2y(s_params'z0)` converts the scattering parameters `s_params` into the admittance parameters `y_params`. The `s_params` input is a complex `n`-by-`n`-by-`m` array, representing `m` n-port S-parameters. `z0` is the reference impedance; its default is 50 ohms. `y_params` is a complex `n`-by-`n`-by-`m` array, representing `m` n-port Y-parameters.

See Also

`abcd2y`, `h2y`, `s2abcd`, `s2h`, `s2z`, `y2s`, `z2y`

Purpose Convert S-parameters to Z-parameters

Syntax

```
z_params = s2z(s_params,z0)
```

Description `z_params = s2z(s_params,z0)` converts the scattering parameters `s_params` into the impedance parameters `z_params`. The `s_params` input is a complex `n`-by-`n`-by-`m` array, representing `m` n-port S-parameters. `z0` is the reference impedance; its default is 50 ohms. `z_params` is a complex `n`-by-`n`-by-`m` array, representing `m` n-port Z-parameters.

See Also `abcd2z`, `h2z`, `s2abcd`, `s2h`, `s2y`, `y2z`, `z2s`

Purpose Plot specified circuit object parameters on a Smith chart

Syntax

```
[lineseries,hsm] = smith(h,parameter1,...,parametern,type)
```

Description

`[lineseries,hsm] = smith(h,parameter1,...,parametern,type)` plots the network parameters `parameter1,...,parametern` from the object `h` on a Smith chart. `h` is the handle of a circuit (`rfckt`) or data (`rfdata`) object. `type` is a string, `'z'` (default),`'y'`, or `'zy'`, specifying the type of Smith chart.

Type `listparam(h)` to get a list of valid parameters for a circuit object `h`.

Note For all circuit objects except those that contain data from a data file, you must use the `analyze` function to perform a frequency domain analysis before calling `smith`.

Note Use the `smithchart` function to plot network parameters that are not part of a circuit (`rfckt`) or data (`rfdata`) object, but are specified as vector data.

Changing Properties of the Plotted Lines

The `smith` function returns `lineseries`, a column vector of handles to `lineseries` objects, one handle per plotted line. Use the MATLAB `lineseries` properties to change the properties of these lines.

Changing Properties of the Smith Chart

The `smith` function returns the handle `hsm` of the Smith chart. Use the properties listed below to change the properties of the chart itself.

Properties

`smith` creates the plot using the default property values of a Smith chart. Use `set(hsm,'PropertyName1',PropertyValue1,...)` to change the property values of the chart. Use `get(hsm)` to get the property values.

This section lists all properties you can specify for a Smith chart object along with units, valid values, and a descriptions of their use.

Property Name	Description	Units, Values
`Color`	Line color for a Z or Y Smith chart. For a ZY Smith chart, the Z line color.	`ColorSpec`. Default is `[0.4 0.4 0.4]` (dark grey).
`LabelColor`	Color of the line labels.	`ColorSpec`. Default is `[0 0 0]` (black).
`LabelSize`	Size of the line labels.	`FontUnits`. Default is 10.
`LabelVisible`	Visibility of the line labels.	`'on'` (default) or `'off'`
`LineType`	Line spec for a Z or Y Smith chart. For a ZY Smith chart, the Z line spec.	`LineSpec`. Default is `'-'` (solid line).
`LineWidth`	Line width for a Z or Y Smith chart. For a ZY Smith chart, the Z line width.	Number of points. Default is `0.5`.
`SubColor`	The Y line color for a ZY Smith chart.	`ColorSpec`. Default is `[0.8 0.8 0.8]` (medium grey).
`SubLineType`	The Y line spec for a ZY Smith chart.	`LineSpec`. Default is `':'` (dotted line).
`SubLineWidth`	The Y line width for a ZY Smith chart.	Number of points. Default is `0.5`.
`Type`	Type of Smith chart	`'z'` (default), `'y'`, or `'zy'`
`Value`	Two-row matrix. Row 1 specifies the constant resistance lines. Row 2 specifies the constant reactance lines.	2-by-n matrix. Default is `[0.2000 0.5000 1.0000 2.0000 5.0000; 1.0000 2.0000 5.0000 5.0000 30.0000]`

See Also

analyze, calculate, getz0, listparam, listformat, plot, polar, read, restore, rfckt, rfdata, write

Purpose

Plot complex vector on a Smith chart

Syntax

```
[lineseries,hsm] = smithchart(y)
[lineseries,hsm] = smithchart
```

Description

`[lineseries,hsm] = smithchart(y)` plots the complex vector `y` on a Smith chart and returns the handle `h` of the Smith chart object. Change the Smith chart properties to customize the chart.

The `smithchart` function returns `lineseries`, a column vector of handles to `lineseries` objects, one handle per plotted line. Use the `lineseries` properties to change the properties of these lines.

The `smithchart` function also returns the handle `hsm` to the Smith chart. Use the properties listed below to change the properties of the chart itself.

`[lineseries,hsm] = smithchart` draws a blank Smith chart.

Note To plot network parameters from a circuit (`rfckt`) or data (`rfdata`) object on a Smith chart, use the `smith` function.

Properties

`smithchart` creates the plot using default property values of a Smith chart. Use `set(h,'PropertyName1',PropertyValue1,...)` to change the property values. Use `get(h)` to get the property values.

This section lists all properties you can specify for `smithchart` objects along with units, valid values, and a descriptions of their use.

Property Name	Description	Units, Values
`Color`	Line color for a Z or Y Smith chart. For a ZY Smith chart, the Z line color.	`ColorSpec`. Default is `[0.4 0.4 0.4]` (dark grey).
`LabelColor`	Color of the line labels.	`ColorSpec`. Default is `[0 0 0]` (black).

Property Name	Description	Units, Values
LabelSize	Size of the line labels.	FontUnits. Default is 10.
LabelVisible	Visibility of the line labels.	'on' (default) or 'off'
LineType	Line spec for a Z or Y Smith chart. For a ZY Smith chart, the Z line spec.	LineSpec. Default is '-' (solid line).
LineWidth	Line width for a Z or Y Smith chart. For a ZY Smith chart, the Z line width.	Number of points. Default is 0.5.
SubColor	The Y line color for a ZY Smith chart.	ColorSpec. Default is [0.8 0.8 0.8] (medium grey).
SubLineType	The Y line spec for a ZY Smith chart.	LineSpec. Default is ':' (dotted line).
SubLineWidth	The Y line width for a ZY Smith chart.	Number of points. Default is 0.5.
Type	Type of Smith chart	'z' (default), 'y', or 'zy'
Value	Two-row matrix. Row 1 specifies the constant resistance lines. Row 2 specifies the constant reactance lines.	2-by-n matrix. Default is [0.2000 0.5000 1.0000 2.0000 5.0000; 1.0000 2.0000 5.0000 5.0000 30.0000]

See Also

get, rfckt, rfdata, set, smith

Purpose

Calculate stability factor K of a two-port network

Syntax

```
[k,b1,b2,delta] = stabilityk(s_params)
```

Description

`[k,b1,b2,delta] = stabilityk(s_params)` calculates and returns the stability factor `k`, as well as the conditions `b1`, `b2`, and `delta` for stability of a two-port network. The input `s_params` is a complex 2-by-2-by-`m` array, representing `m` two-port S-parameters.

$$K = 1 - |S_{11}|^2 - |S_{22}|^2 + |\Delta|^2 / (2|S_{12}S_{21}|)$$

$$B_1 = 1 + |S_{11}|^2 - |S_{22}|^2 - |\Delta|^2$$

$$B_2 = 1 - |S_{11}|^2 + |S_{22}|^2 - |\Delta|^2$$

where

- S_{11}, S_{12}, S_{21}, and S_{22} are vectors of S-parameters, taken from the input argument `s_params`.
- $\Delta = S_{11}S_{22} - S_{12}S_{21}$

References

Gonzalez, Guillermo, *Microwave Transistor Amplifiers: Analysis and Design*, 2nd edition, Prentice Hall, 1997, pp. 217-228.

See Also

`stabilitymu`

stabilitymu

Purpose Calculate the stability factor μ of a two-port network

Syntax `[mu,muprime] = stabilitymu(s_params)`

Description `[mu,muprime] = stabilitymu(s_params)` calculates and returns the stability factors μ and μ', of a two-port network. The input `s_params` is a complex 2-by-2-by-`m` array, representing `m` two-port S-parameters.

$$\mu = (1 - |S_{11}|^2)/(|S_{22} - S_{11}^*\Delta| + |S_{21}S_{12}|)$$

$$\mu' = (1 - |S_{22}|^2)/(|S_{11} - S_{22}^*\Delta| + |S_{21}S_{12}|)$$

where

- S_{11}, S_{12}, S_{21}, and S_{22} are vectors of S-parameters, taken from the input argument `s_params`.
- $\Delta = S_{11}S_{22} - S_{12}S_{21}$
- S^* is the complex conjugate of the designated S-parameter.

μ defines the minimum distance between the center of the unit Smith chart and the unstable region in the load plane (the load is considered port 2).

μ' defines the minimum distance between the center of the unit Smith chart and the unstable region in the source plane (the source is considered port 1).

Having $\mu > 1$ (or $\mu' > 1$) is necessary and sufficient for the two-port linear network, described by the S-parameters, to be unconditionally stable.

References

Edwards, Marion Lee, and Jeffrey H. Sinsky, “A New Criterion for Linear 2-Port Stability Using a Single Geometrically Derived Parameter,” *IEEE Transactions on Microwave Theory and Techniques*, Vol. 40, No. 12, December 1992, pp. 2303-2311.

See Also `stabilityk`

Purpose Convert T-parameters to S-parameters

Syntax

```
s_params = t2s(t_params)
```

Description `s_params = t2s(t_params)` converts the chain scattering parameters `t_params` into the scattering parameters `s_params`. The `t_params` input is a complex 2-by-2-by-`m` array, representing `m` two-port T-parameters. `s_params` is a complex 2-by-2-by-`m` array, representing `m` two-port S-parameters.

See Also `abcd2s`, `h2s`, `s2t`, `y2s`, `z2s`

Purpose Calculates the VSWR at the given reflection coefficient gamma

Syntax

```
result = vswr(gamma)
```

Description

`result = vswr(gamma)` calculates the voltage standing-wave ratio (VSWR) at the given reflection coefficient gamma as

$$\mathrm{VSWR} = \frac{1 + |\Gamma|}{1 - |\Gamma|}$$

where Γ is the given reflection coefficient `gamma`. The input `gamma` is a complex vector. `result` is a real vector of the same length as `gamma`.

See Also `gammain`, `gammaout`

Purpose

Write RF data from a circuit or data object to a file

Syntax

```
status = write(data,filename,dataformat,funit,printformat,
   freqformat)
```

Description

`status = write(data,filename,dataformat,funit,printformat, freqformat)` writes information from `data` to the specified file. `data` is a circuit object or `rfdata.data` object that contains sufficient information to write the specified file. `filename` is a string representing the filename of a `.snp`, `.ynp`, `.znp`, `.hnp`, or `.amp` file, where `n` is the number of ports. The default `filename` extension is `.snp`. See Appendix A, "AMP File Format" for information about the `.amp` format. `write` returns `True` if the operation is successful and returns `False` otherwise.

`dataformat` specifies the format of the data to be written. It must be one of the case-insensitive strings in the following table.

Format	Description
`'DB'`	Data is given in (dB-magnitude, angle) pairs with angle in degrees.
`'MA'`	Data is given in (magnitude, angle) pairs with angle in degrees.
`'RI'`	Data is given in (real, imaginary) pairs (default).

`funit` specifies the frequency units of the data to be written. It must be `'GHz'`, `'MHz'`, `'KHz'`, or `'Hz'`. If you do not specify `funit`, its value is taken from the object `data`. All values are case insensitive.

`printformat` is a string that specifies the precision of the network and noise parameters. See the Precision specification for `fprintf`.

`freqformat` is a string that specifies the precision of the frequency. See the Precision specification for `fprintf`.

References

[1] EIA/IBIS Open Forum, "Touchstone File Format Specification," Rev. 1.1, 2002 (`http://www.eda.org/pub/ibis/connector/touchstone_spec11.pdf`).

See Also

analyze, calculate, getz0, listparam, listformat, plot, polar, smith, read, restore, rfckt, rfdata

Purpose Convert Y-parameters to ABCD-parameters

Syntax

```
abcd_params = y2abcd(y_params)
```

Description abcd_params = y2abcd(y_params) converts the admittance parameters y_params into the ABCD parameters abcd_params. The y_params input is a complex 2-by-2-by-m array, representing m two-port Y-parameters. abcd_params is a complex 2-by-2-by-m array, representing m two-port ABCD-parameters.

See Also abcd2y, h2abcd, s2abcd, y2h, y2s, y2z, z2abcd

y2h

Purpose Convert Y-parameters to hybrid h-parameters

Syntax

```
h_params = y2h(y_params)
```

Description `h_params = y2h(y_params)` converts the admittance parameters `y_params` into the hybrid parameters `h_params`. The `y_params` input is a complex 2-by-2-by-`m` array, representing `m` two-port Y-parameters. `h_params` is a complex 2-by-2-by-`m` array, representing `m` two-port hybrid h-parameters.

See Also `abcd2h`, `h2y`, `s2h`, `y2abcd`, `y2s`, `y2z`, `z2h`

Purpose Convert Y-parameters to S-parameters

Syntax

```
s_params = y2s(y_params,z0)
```

Description `s_params = y2s(y_params,z0)` converts the admittance parameters `y_params` into the scattering parameters `s_params`. The `y_params` input is a complex `n`-by-`n`-by-`m` array, representing `m` `n`-port Y-parameters. `z0` is the reference impedance; its default is 50 ohms. `s_params` is a complex `n`-by-`n`-by-`m` array, representing `m` `n`-port S-parameters.

See Also `abcd2s`, `h2s`, `s2y`, `y2abcd`, `y2h`, `y2s`, `y2z`, `z2s`

y2z

Purpose

Convert Y-parameters to Z-parameters

Syntax

```
z_params = y2z(y_params)
```

Description

`z_params = y2z(y_params)` converts the admittance parameters `y_params` into the impedance parameters `z_params`. The `y_params` input is a complex `n`-by-`n`-by-`m` array, representing `m` `n`-port Y-parameters. `z_params` is a complex `n`-by-`n`-by-`m` array, representing `m` `n`-port Z-parameters.

See Also

`abcd2z`, `h2z`, `y2abcd`, `y2h`, `y2s`, `y2z`, `z2s`, `z2y`

z2abcd

Purpose

Convert Z-parameters to ABCD-parameters

Syntax

```
abcd_params = z2abcd(z_params)
```

Description

`abcd_params = z2abcd(z_params)` converts the impedance parameters `z_params` into the ABCD parameters `abcd_params`. The `z_params` input is a complex 2-by-2-by-`m` array, representing `m` two-port Z-parameters. `abcd_params` is a complex 2-by-2-by-`m` array, representing `m` two-port ABCD-parameters.

See Also

`abcd2z`, `h2abcd`, `s2abcd`, `y2abcd`, `z2h`, `z2s`, `z2y`

z2h

Purpose Convert Z-parameters to hybrid h-parameters

Syntax

```
h_params = z2h(z_params)
```

Description h_params = z2h(z_params) converts the impedance parameters z_params into the hybrid parameters h_params. The z_params input is a complex 2-by-2-by-m array, representing m two-port Z-parameters. h_params is a complex 2-by-2-by-m array, representing m two-port hybrid h-parameters.

See Also abcd2h, h2z, s2h, y2h, z2abcd, z2s, z2y

Purpose Convert Z-parameters to S-parameters

Syntax

```
s_params = z2s(z_params,z0)
```

Description `s_params = z2s(z_params,z0)` converts the impedance parameters `z_params` into the scattering parameters `s_params`. The `z_params` input is a complex `n`-by-`n`-by-`m` array, representing `m` `n`-port Z-parameters. `z0` is the reference impedance; its default is 50 ohms. `s_params` is a complex `n`-by-`n`-by-`m` array, representing `m` `n`-port S-parameters.

See Also `abcd2s`, `h2s`, `s2z`, `y2s`, `z2abcd`, `z2h`, `z2y`

z2y

Purpose Convert Z-parameters to Y-parameters

Syntax

```
y_params = z2y(z_params)
```

Description `y_params = z2y(z_params)` converts the impedance parameters `z_params` into the admittance parameters `y_params`. The `z_params` input is a complex `n`-by-`n`-by-`m` array, representing `m` `n`-port Z-parameters. `y_params` is a complex `n`-by-`n`-by-`m` array, representing `m` `n`-port Y-parameters.

See Also `abcd2y`, `h2y`, `s2y`, `y2z`, `z2abcd`, `z2h`, `z2s`

AMP File Format

Overview

The AMP data file describes a single non-linear device. Its format can contain the following types of data. These topics describe the sections of the file that contain the data.

- "S, Y, or Z Network Parameters" on page A-4
- "Noise Parameters" on page A-6
- "Noise Figure Data" on page A-7
- "Power Data" on page A-9
- "IP3 Data" on page A-10

An AMP file must contain either power data and/or network parameter data to be valid. To accommodate analysis at more than one frequency, the file can contain more than one section of power data. Noise data, noise figure data, and IP3 data are optional.

Two sample AMP files, `samplepa1.amp` and `default.amp`, ship with the RF Toolbox. They describe a nonlinear 2-port amplifier with noise. "RF Circuit Objects" on page 2-12 is an example that uses `default.amp`.

See "Comments" on page A-3 for information about adding comments to an AMP file.

Comments

An asterisk (*) or an exclamation point (!) precedes a comment that appears on a separate line.

A semicolon (;) precedes a comment that appears following data on the same line.

Data Sections

Each kind of data resides in its own section. Each section consists of a two-line header followed by lines of numeric data. Numeric values can be in any valid MATLAB format.

A new header indicates the end of the previous section. The data sections can appear in any order in the file.

In the following descriptions, brackets (`[]`) indicate optional data or characters. All values are case-insensitive.

- "S, Y, or Z Network Parameters" on page A-4
- "Noise Parameters" on page A-6
- "Noise Figure Data" on page A-7
- "Power Data" on page A-9
- "IP3 Data" on page A-10

S, Y, or Z Network Parameters

Header Line 1

The first line of the header has the format

```
Keyword [Parameter] [R[REF][=]value]
```

`Keyword` indicates the type of network parameter. It can be `S[PARAMETERS]`, `Y[PARAMETERS]`, or `Z[PARAMETERS]`. `Parameter` indicates the form of the data. It can be `MA`, `DB`, or `RI`. The default for S-parameters is `MA`. The default for Y- and Z-parameters is `RI`. `R[REF][=]value` is the reference impedance. The default reference impedance is 50 ohms.

The following table explains the meaning of the allowable `Parameter` values.

Parameter	Description
`MA`	Data is given in (magnitude, angle) pairs with angle in degrees (default for S-parameters).

Parameter	Description
DB	Data is given in (dB-magnitude, angle) pairs with angle in degrees.
RI	Data is given in (real, imaginary) pairs (default for Y- and Z-parameters).

This example of a first line indicates that the section contains S-parameter data given in (real, imaginary) pairs, and that the reference impedance is 50 ohms.

```
S RI R 50
```

Header Line 2

The second line of the header has the format

```
Independent_variable Units
```

The data in a section is a function of the `Independent_variable`. Currently, for S-, Y-, and Z-parameters, the value of `Independent_variable` is always `F[REQ]`. `Units` indicates the default units of the frequency data. It can be `GHz`, `MHz`, or `KHz`. You must specify `Units`, but you can override this default on any given line of data.

This example of a second line indicates that the default units for frequency data is GHz.

```
FREQ GHZ
```

Data

The data that follows the header typically consists of nine columns.

The first column contains the frequency points where network parameters are measured. They can appear in any order. If the frequency is given in units other than those you specified as the default, you must follow the value with the appropriate units; there should be no intervening spaces. For example,

```
FREQ GHZ
1000MHZ  ...
2000MHZ  ...
3000MHZ  ...
```

Columns two though nine contain 2-port network parameters in the order N11, N21, N12, N22. Similar to the Touchstone format, each Nnn corresponds to two consecutive columns of data in the chosen form: MA, DB, or RI. The data can be in any valid MATLAB format.

This example is derived from the file `default.amp`. A comment line explains the column arrangement of the data where `re` indicates real and `im` indicates imaginary.

```
S RI R 50
FREQ GHZ
* FREQ     reS11      imS11      reS21      imS21      reS12      imS12      reS22      imS22
  1.00  -0.724725  -0.481324  -0.685727   1.782660   0.000000   0.000000  -0.074122  -0.321568
  1.01  -0.731774  -0.471453  -0.655990   1.798041   0.001399   0.000463  -0.076091  -0.319025
  1.02  -0.738760  -0.461585  -0.626185   1.813092   0.002733   0.000887  -0.077999  -0.316488
```

Noise Parameters

Header Line 1

The first line of the header has the format

```
Keyword
```

`Keyword` must be `NOI[SE]`.

Header Line 2

The second line of the header has the format

```
Variable Units
```

`Variable` must be `F[REQ]`. `Units` indicates the default units of the frequency data. It can be `GHz`, `MHz`, or `KHz`. You can override this default on any given line of data. This example of a second line indicates that frequency data is assumed to be in GHz, unless other units are specified.

```
FREQ GHz
```

Data

The data that follows the header must consist of five columns.

The first column contains the frequency points at which noise parameters were measured. The frequency points can appear in any order. If the frequency is given in units other than those you specified as the default, you must follow the value with the appropriate units; there should be no intervening spaces. For example,

```
NOI
FREQ GHZ
1000MHZ  ...
2000MHZ  ...
3        ...
4        ...
5        ...
```

Columns two through five contain, in order,

- Minimum noise figure in decibels
- Magnitude of the source reflection coefficient to realize minimum noise figure
- Phase in degrees of the source reflection coefficient
- Effective noise resistance normalized to the reference impedance of the network parameters

This example is taken from the file `default.amp`. A comment line explains the column arrangement of the data.

```
NOI RN
FREQ GHz
* Freq  Fmin(dB)  GammmaOpt(MA:Mag) GammmaOpt(MA:Ang) RN/Zo
  1.90  10.200000 1.234000         -78.400000         0.240000
  1.93  12.300000 1.235000         -68.600000         0.340000
  2.06  13.100000 1.254000         -56.700000         0.440000
  2.08  13.500000 1.534000         -52.800000         0.540000
  2.10  13.900000 1.263000         -44.400000         0.640000
```

Noise Figure Data

The AMP file format supports the use of frequency-dependent noise figure (NF) data.

Header Line 1

The first line of the header has the format

```
Keyword [Units]
```

For noise figure data, `Keyword` must be `NF`. The optional `Units` field indicates the default units of the NF data. It must be `dB`, i.e., data must be given in decibels.

This example of a first line indicates that the section contains NF data, which is assumed to be in decibels.

```
NF
```

Header Line 2

The second line of the header has the format

```
Variable Units
```

`Variable` must be `F[REQ]`. `Units` indicates the default units of the frequency data. It can be `GHz`, `MHz`, or `KHz`. This example of a second line indicates that frequency data is assumed to be in GHz.

```
FREQ GHz
```

Data

The data that follows the header typically consists of two columns.

The first column contains the frequency points at which the NF data are measured. Frequency points can appear in any order. For example,

```
NF
FREQ MHz
2090  ...
2180  ...
2270  ...
```

Column two contains the corresponding NF data in decibels.

This example is derived from the file `samplepa1.amp`.

```
NF dB
FREQ GHz
1.900   10.3963213
2.000   12.8797965
2.100   14.0611765
2.200   13.2556751
2.300   12.9498642
```

```
2.400   13.3244309
2.500   12.7545104
```

Note If your noise figure data consists of a single scalar value with no associated frequency, that same value is used for all frequencies. Enter the value in column one of the line following header line 2. You must include the second line of the header, but it is ignored.

Power Data

An AMP file describes power data as input power-dependent output power.

Header Line 1

The first line of the header has the format

```
Keyword [Units]
```

For power data, `Keyword` must be `POUT`, indicating that this section contains power data. Because output power is complex, `Units` indicates the default units of the magnitude of the output power data. It can be `dBW`, `dBm`, `mW`, or `W`. The default is `W`. You can override this default on any given line of data.

The following table explains the meaning of the allowable `Units` values.

Units	Description
dBW	Decibels referenced to one watt
dBm	Decibels referenced to one milliwatt
mW	Milliwatts
W	Watts

This example of a first line indicates that the section contains output power data whose magnitude is assumed to be in decibels referenced to one milliwatt, unless other units are specified.

```
POUT dBm
```

Header Line 2

The second line of the header has the format

```
Keyword [Units] FREQ[=]value
```

`Keyword` must be `PIN`. `Units` indicates the default units of input power data. It can be `dBW`, `dBm`, `mW`, or `W`. The default is `W`. You can override this default on any given line of data. `FREQ[=]value` is the frequency point at which the power is measured. The value must include the units as `GHz`, `MHz`, `kHz`, or `Hz`.

This example of a second line indicates that the section contains input power data that is assumed to be in decibels referenced to one milliwatt, unless other units are specified. It also indicates that the power data was measured at a frequency of 2.1E+009Hz.

```
PIN dBm FREQ=2.1E+009Hz
```

Data

The data that follows the header typically consists of three columns.

The first column contains input power data. It can appear in any order. The second column contains the corresponding output power magnitude. The third column contains the output phase shift in degrees. If all phases are zero, you can omit the third column.

If the power is given in units other than those you specified as the default, you must follow the value with the appropriate units; there should be no intervening spaces.

This example is derived from the file `default.amp`. A comment line explains the column arrangement of the data.

```
POUT dbm
PIN dBm FREQ = 2.10GHz
* Pin     Pout            Phase(degrees)
  0.0     19.28           0.0
  1.0     20.27           0.0
  2.0     21.26           0.0
```

IP3 Data

An AMP file can include frequency-dependent third order input (IIP3) or output (OIP3) intercept points.

Header Line 1

The first line of the header has the format

```
Keyword [Units]
```

For IP3 data, `Keyword` can be either `IIP3` or `OIP3`, indicating that this section contains input IP3 data or output IP3 data. `Units` indicates the default units of the IP3 data. It can be `dBW`, `dBm`, `mW`, or `W`. The default is `W`.

The following table explains the meaning of the allowable `Units` values.

Units	Description
`dBW`	Decibels referenced to one watt
`dBm`	Decibels referenced to one milliwatt
`mW`	Milliwatts
`W`	Watts

This example of a first line indicates that the section contains input IP3 data which is assumed to be in decibels referenced to one milliwatt.

```
IIP3 dBm
```

Header Line 2

The second line of the header has the format

```
Variable Units
```

`Variable` must be `FREQ`. `Units` indicates the default units of the frequency data. It can be `GHz`, `MHz`, or `KHz`. This example of a second line indicates that frequency data is assumed to be in GHz.

```
FREQ GHz
```

Data

The data that follows the header typically consists of two columns.

The first column contains the frequency points at which the IP3 parameters are measured. Frequency points can appear in any order.

```
OIP3
FREQ GHz
2.010  ...
2.020  ...
2.030  ...
```

Column two contains the corresponding IP3 data.

This example is derived from the file `samplepa1.amp`.

```
OIP3 dBm
FREQ GHz
2.100   38.8730377
```

Note If your IP3 data consists of a single scalar value with no associated frequency, that same value is used for all frequencies. Enter the value in column one of the line following header line 2. You must include the second line of the header, but it is ignored.

Index

D

E

F

M

N

O

P

R

Z