RF Toolbox™

User's Guide

R2013a

MATLAB®

MathWorks®

How to Contact MathWorks

www.mathworks.com	Web
comp.soft-sys.matlab	Newsgroup
www.mathworks.com/contact_TS.html	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

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)
March 2006	Online only	Revised for Version 1.3 (Release 2006a)
September 2006	Online only	Revised for Version 2.0 (Release 2006b)
March 2007	Online only	Revised for Version 2.1 (Release 2007a)
September 2007	Online only	Revised for Version 2.2 (Release 2007b)
March 2008	Online only	Revised for Version 2.3 (Release 2008a)
October 2008	Online only	Revised for Version 2.4 (Release 2008b)
March 2009	Online only	Revised for Version 2.5 (Release 2009a)
September 2009	Online only	Revised for Version 2.6 (Release 2009b)
March 2010	Online only	Revised for Version 2.7 (Release 2010a)
September 2010	Online only	Revised for Version 2.8 (Release 2010b)
April 2011	Online only	Revised for Version 2.8.1 (Release 2011a)
September 2011	Online only	Revised for Version 2.9 (Release 2011b)
March 2012	Online only	Revised for Version 2.10 (Release 2012a)
September 2012	Online only	Revised for Version 2.11 (Release 2012b)
March 2013	Online only	Revised for Version 2.12 (Release 2013a)

Contents

1 Getting Started

2 RF Objects

3 Model an RF Component

4 Export Verilog-A Models

5 RF Tool: An RF Analysis GUI

6 Objects — Alphabetical List

7 Methods — Alphabetical List

8 Functions — Alphabetical List

A AMP File Format

Index

1

Getting Started

Product Description

Design, model, and analyze networks of RF components

RF Toolbox™ extends the MATLAB® product with functions and a graphical user interface (GUI) for designing, modeling, analyzing, and visualizing networks of radio frequency (RF) components. You can use RF Toolbox software for working on wireless communications, radar, and signal integrity projects.

Key Features

- Provides read and write access to industry-standard file formats for network parameters
- Defines RF filters, transmission lines, amplifiers, and mixers by their experimental or theoretical network parameters and physical properties
- Calculates network parameters for RF components in series, parallel, cascade, hybrid, and inverse hybrid configurations
- Builds models using the rational function fitting method
- Exports rational function models to Simulink® or as Verilog-A modules
- Calculates noise figures and third-order intercept points for cascaded components
- Converts among S, Y, Z, ABCD, h, g, and T network parameters
- Includes rectangular and polar plots and Smith® Charts for visualizing data

Related Products

Several MathWorks® products are especially relevant to the kinds of tasks you can perform with RF Toolbox software. The following table summarizes the related products and describes how they complement the features of the toolbox.

Product	Description
"Communications System Toolbox™"	Simulink blocks and MATLAB functions for time-domain simulation of modulation and demodulation of a wireless communications signal.
"DSP System Toolbox™"	Simulink blocks and MATLAB functions for time-domain simulation of for filtering the modulated communication signal.
"SimRF™"	Circuit-envelope and equivalent-baseband simulation of RF components in Simulink.
"Signal Processing Toolbox™"	MATLAB functions for filtering the modulated communication signal.

RF Objects

RF Toolbox software uses objects to represent RF components and networks. You create an object using the object's *constructor*. Every object has predefined fields called *properties*. The properties define the characteristics of the object. Each property associated with an object is assigned a value. Every object has a set of *methods*, which are operations that you can perform on the object. Methods are similar to functions except that they only act on an object.

The following table summarizes the types of objects that are available in the toolbox and describes the uses of each one. For more information on a particular type of object, including a list of the available objects and methods, follow the link in the table to the documentation for that object type.

Object Type	Name	Description
"RF Data Objects" on page 2-2	`rfdata`	Stores data for use by other RF objects or for plotting and network parameter conversion.
"RF Circuit Objects" on page 2-4	`rfckt`	Represents RF components and networks using network parameters and physical properties for frequency-domain simulation.
"RF Model Objects" on page 2-10	`rfmodel`	Represents RF components and networks mathematically for computing time-domain behavior and exporting models.

Each name in the preceding table is the prefix to the names of all object constructors of that type. The constructors use *dot notation* that consists of the object type, followed by a dot and then the component name. The component name is also called the *class*. For information on how to construct an RF object from the command line using dot notation, see "Create RF Objects" on page 3-2.

You use a different form of dot notation to specify object properties, as described in “Reference Properties Directly Using Dot Notation” on page 3-16. This is just one way to define component data. For more information on object properties, see “Specify or Import Component Data” on page 3-5.

You use object methods to perform frequency-domain analysis and visualize the results. For more information, see “Analyze and Plot RF Components” on page 3-25.

Note The toolbox also provides a graphical interface for creating and analyzing circuit objects. For more information, see “RF Tool” on page 5-2.

S-Parameter Notation

Define S-Parameters

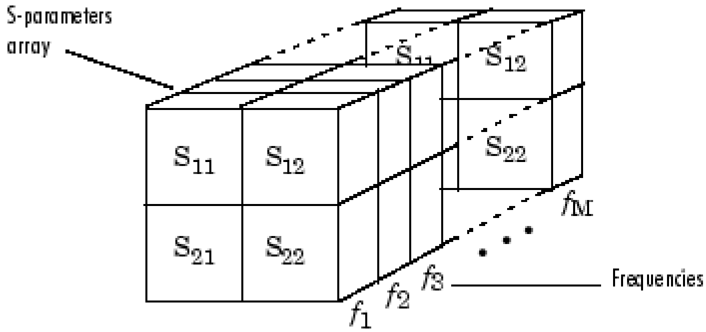
RF Toolbox software uses matrix notation to specify S-parameters. The indices of an S-parameter matrix correspond to the port numbers of the network that the data represent. For example, to define a matrix of 50-ohm, 2-port S-parameters, type:

```
s11 = 0.61*exp(j*165/180*pi);
s21 = 3.72*exp(j*59/180*pi);
s12 = 0.05*exp(j*42/180*pi);
s22 = 0.45*exp(j*(-48/180)*pi);
s_params = [s11 s12; s21 s22];
```

RF Toolbox functions that operate on `s_params` assume:

- `s_params(1,1)` corresponds to the reflection coefficient at port 1, S_{11}.
- `s_params(2,1)` corresponds to the transmission coefficient from port 1 to port 2, S_{21}.
- `s_params(1,2)` corresponds to the transmission coefficient from port 2 to port 1, S_{12}.
- `s_params(2,2)` corresponds to the reflection coefficient at port 2, S_{22}.

RF Toolbox software also supports three-dimensional arrays of S-parameters. The third dimension of an S-parameter array corresponds to S-parameter data at different frequencies. The following figure illustrates this convention.



Refer to S-Parameters Using Strings

RF Toolbox software uses strings to refer to S-parameters in plotting and calculation methods, such as `plot`. These strings have one of the following two forms:

- `'Snm'` — Use this syntax if *n* and *m* are both less than 10.
- `'Sn,m'` — Use this syntax if one or both are greater than 10. `'Sn,m'` is not a valid syntax when both *n* and *m* are less than 10.

The indices *n* and *m* are the port numbers for the S-parameters.

Most toolbox objects only analyze 2-port S-parameters. The following objects analyze S-parameters with more than two ports:

- `rfckt.passive`
- `rfckt.datafile`
- `rfdata.data`

You can get 2-port parameters from S-parameters with an arbitrary number of ports using one or more of the following steps:

- Extract 2-port S-parameters from N-port S-parameters.

  See "Extract M-Port S-Parameters from N-Port S-Parameters" on page 3-21.
- Convert single-ended 4-port parameters to differential 2-port parameters.

See "Convert Single-Ended S-Parameters to Mixed-Mode S-Parameters" on page 3-19.

RF Analysis

When you analyze an RF circuit using RF Toolbox software, your workflow might include the following tasks:

1. Select RF circuit objects to represent the components of your RF network.

   See "Create RF Objects" on page 3-2.

2. Define component data by:

   - Specifying network parameters or physical properties (see "Set Property Values" on page 3-5).
   - Importing data from an industry-standard Touchstone file, a MathWorks AMP file, an Agilent® P2D or S2D file, or the MATLAB workspace (see "Import Property Values from Data Files" on page 3-8).
   - Where applicable, selecting operating condition values (see "Specify Operating Conditions" on page 3-17).

3. Where applicable, perform network parameter conversions on imported file data.

   See "Process File Data for Analysis" on page 3-19.

4. Integrate components to form a cascade, hybrid, parallel, or series network.

   See "Construct Networks of Specified Components" on page 3-7.

5. Analyze the network in the frequency domain.

   See "Analyze Networks in the Frequency Domain" on page 3-25.

6. Generate plots to gain insight into network behavior.

   The following plots and charts are available in the toolbox:

   - Rectangular plots
   - Polar plots
   - Smith Charts
   - Budget plots (for cascaded S-parameters)

See “Visualize Component and Network Data” on page 3-26.

7. Compute the network transfer function.

   See “Compute the Network Transfer Function” on page 3-35.

8. Create an RF model object that describes the transfer function analytically.

   See “Fit a Model Object to Circuit Object Data” on page 3-36.

9. Plot the time-domain response.

   See “Compute and Plotting the Time-Domain Response” on page 3-37.

10. Export a Verilog-A description of the network.

    See “Export a Verilog-A Model” on page 4-5.

Model a Cascaded RF Network

In this section...

Overview

In this example, you use the RF Toolbox command-line interface to model the gain and noise figure of a cascaded network. You analyze the network in the frequency domain and plot the results.

Note To learn how to use RF Tool to perform these tasks, see "Model an RF Network Using RF Tool" on page 5-31.

The network that you use in this example consists of an amplifier and two transmission lines. The toolbox represents RF components and RF networks using RF circuit objects. You learn how to create and manipulate these objects to analyze the cascaded amplifier network.

Create RF Components

Type the following set of commands at the MATLAB prompt to create three circuit (`rfckt`) objects with the default property values. These circuit objects represent the two transmission lines and the amplifier:

```
FirstCkt = rfckt.txline;
SecondCkt = rfckt.amplifier;
ThirdCkt = rfckt.txline;
```

Specify Component Data

In this part of the example, you specify the following component properties:

- "Transmission Line Properties" on page 1-12
- "Amplifier Properties" on page 1-12

Transmission Line Properties

1. Type the following command at the MATLAB prompt to change the line length of the first transmission line, `FirstCkt`, to `0.001`:

   ```
   set(FirstCkt,'LineLength',0.001)
   ```

2. Type the following command at the MATLAB prompt to change the line length of the second transmission line, `ThirdCkt`, to `0.025` and to change the phase velocity to `2.0e8`:

   ```
   set(ThirdCkt,'LineLength',0.025,'PV',2.0e8)
   ```

Amplifier Properties

1. Type the following command at the MATLAB prompt to import network parameters, noise data, and power data from the `default.amp` file into the amplifier, `SecondCkt`:

   ```
   read(SecondCkt, 'default.amp');
   ```

2. Type the following command at the MATLAB prompt to change the interpolation method of the amplifier, `SecondCkt`, to `cubic`:

   ```
   set(SecondCkt,'IntpType','cubic')
   ```

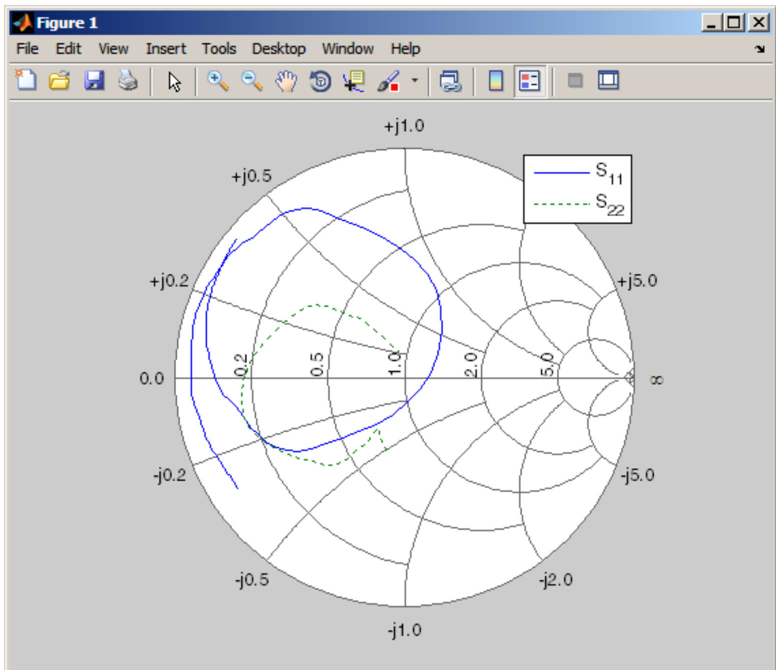
   The `IntpType` property tells the toolbox how to interpolate the network parameters, noise data, and power data when you analyze the amplifier at frequencies other than those specified in the file.

Validate RF Components

In this part of the example, you plot the network parameters and power data (output power versus input power) to validate the behavior of the amplifier.

1 Type the following set of commands at the MATLAB prompt to use the `smith` command to plot the original S_{11} and S_{22} parameters of the amplifier (`SecondCkt`) on a Z Smith Chart:

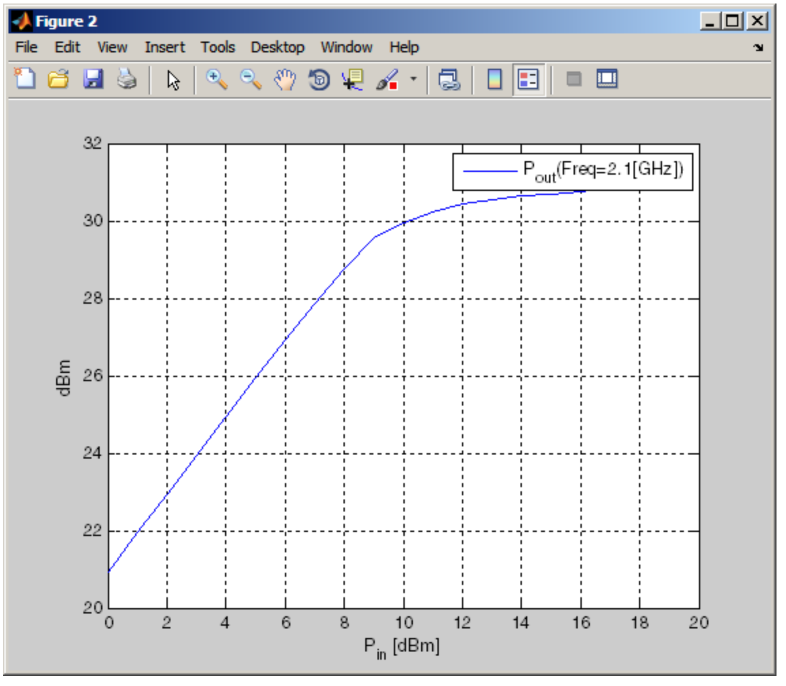
```
figure
lineseries1 = smith(SecondCkt,'S11','S22');
set(lineseries1(1), 'LineStyle','-', 'LineWidth', 1);
set(lineseries1(2), 'LineStyle',':', 'LineWidth', 1);
legend show
```



Note The plot shows the S-parameters over the frequency range for which network data is specified in the `default.amp` file — from 1 GHz to 2.9 GHz.

2 Type the following set of commands at the MATLAB prompt to use the RF Toolbox `plot` command to plot the amplifier (`SecondCkt`) output power (P_{out}) as a function of input power (P_{in}), both in decibels referenced to one milliwatt (dBm), on an X-Y plane plot:

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



Note The plot shows the power data at 2.1 GHz because this frequency is the one for which power data is specified in the `default.amp` file.

Build and Simulate the Network

In this part of the example, you create a circuit object to represent the cascaded amplifier and analyze the object in the frequency domain.

1 Type the following command at the MATLAB prompt to cascade the three circuit objects to form a new cascaded circuit object, `CascadedCkt`:

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

2 Type the following set of commands at the MATLAB prompt to define the range of frequencies over which to analyze the cascaded circuit, and then run the analysis:

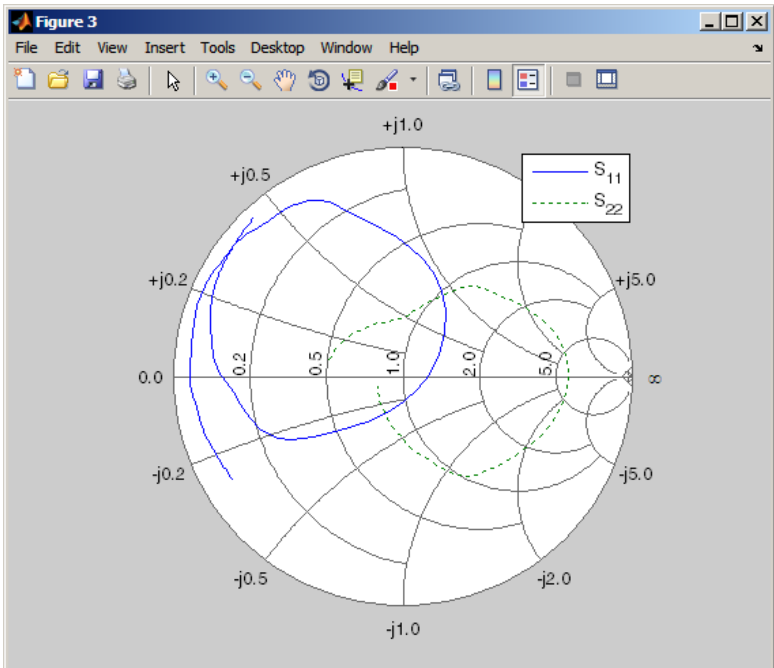
```
f = 1.0e9:1e7:2.9e9;
analyze(CascadedCkt,f);
```

Analyze Simulation Results

In this part of the example, you analyze the results of the simulation by plotting data for the circuit object that represents the cascaded amplifier network.

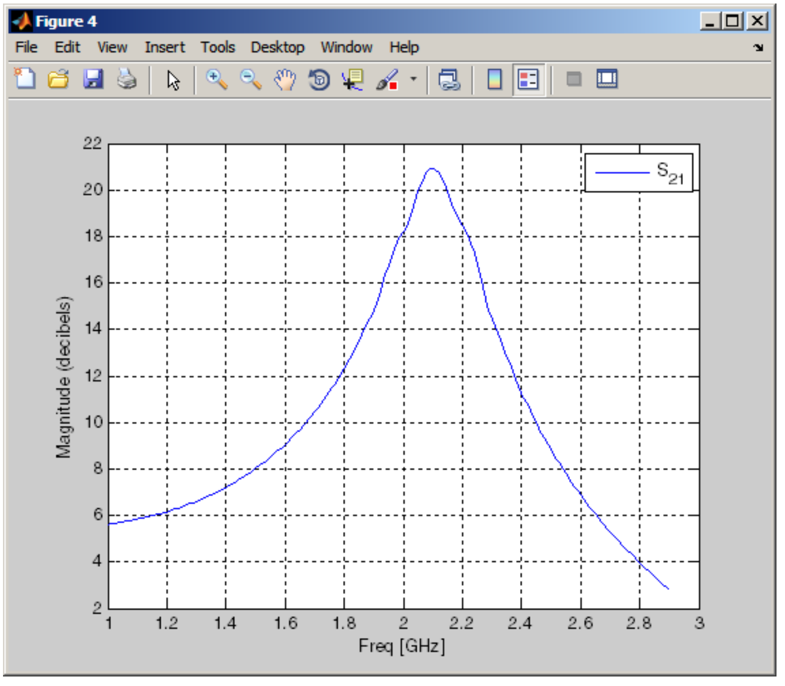
1 Type the following set of commands at the MATLAB prompt to use the `smith` command to plot the S_{11} and S_{22} parameters of the cascaded amplifier network on a Z Smith Chart:

```
figure
lineseries2 = smith(CascadedCkt,'S11','S22','z');
set(lineseries2(1),'LineStyle','-','LineWidth',1);
set(lineseries2(2),'LineStyle',':','LineWidth',1);
legend show
```



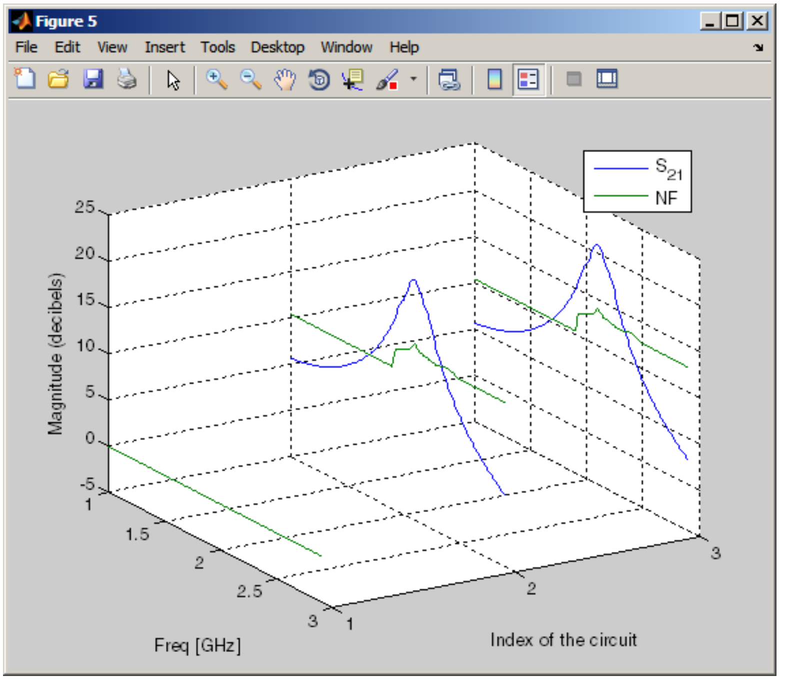
2 Type the following set of commands at the MATLAB prompt to use the `plot` command to plot the S_{21} parameter of the cascaded network, which represents the network gain, on an X-Y plane:

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



3 Type the following set of commands at the MATLAB prompt to use the `plot` command to create a budget plot of the S_{21} parameter and the noise figure of the amplifier network:

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



The budget plot shows parameters as a function of frequency by circuit index. Components are indexed based on their position in the network. In this example:

- Circuit index one corresponds to `FirstCkt`.
- Circuit index two corresponds to `SecondCkt`.
- Circuit index three corresponds to `ThirdCkt`.

The curve for each index represents the contributions of the RF components up to and including the component at that index.

Analyze a Transmission Line

In this section...

Overview

In this example, you use the RF Toolbox command-line interface to model the time-domain response of a parallel plate transmission line. You analyze the network in the frequency domain, compute and plot the time-domain response of the network, and export a Verilog-A model of the transmission line for use in system-level simulations.

Build and Simulate the Transmission Line

1 Type the following command at the MATLAB prompt to create a circuit (`rfckt`) object to represent the transmission line, which is 0.1 meters long and 0.05 meters wide:

```
tline = rfckt.parallelplate('LineLength',0.1,'Width',0.05);
```

2 Type the following set of commands at the MATLAB prompt to define the range of frequencies over which to analyze the transmission line and then run the analysis:

```
f = [1.0e9:1e7:2.9e9];
analyze(tline,f);
```

Compute the Transmission Line Transfer Function and Time-Domain Response

This part of the example illustrates how to perform the following tasks:

- "Calculate the Transfer Function" on page 1-20
- "Fit and Validate the Transfer Function Model" on page 1-20
- "Compute and Plot the Time-Domain Response" on page 1-23

Calculate the Transfer Function

1 Type the following command at the MATLAB prompt to extract the computed S-parameter values and the corresponding frequency values for the transmission line:

```
[S_Params, Freq] = extract(tline,'S_Parameters');
```

2 Type the following command at the MATLAB prompt to compute the transfer function from the frequency response data using the `s2tf` function:

```
TrFunc = s2tf(S_Params);
```

Fit and Validate the Transfer Function Model

In this part of the example, you fit a rational function model to the transfer function. The toolbox stores the fitting results in an `rfmodel` object. You use the RF Toolbox `freqresp` method to validate the fit of the rational function model.

1 Type the following command at the MATLAB prompt to fit a rational function to the computed data and store the result in an `rfmodel` object:

```
RationalFunc = rationalfit(Freq,TrFunc)
```

2 Type the following command at the MATLAB prompt to compute the frequency response of the fitted model data:

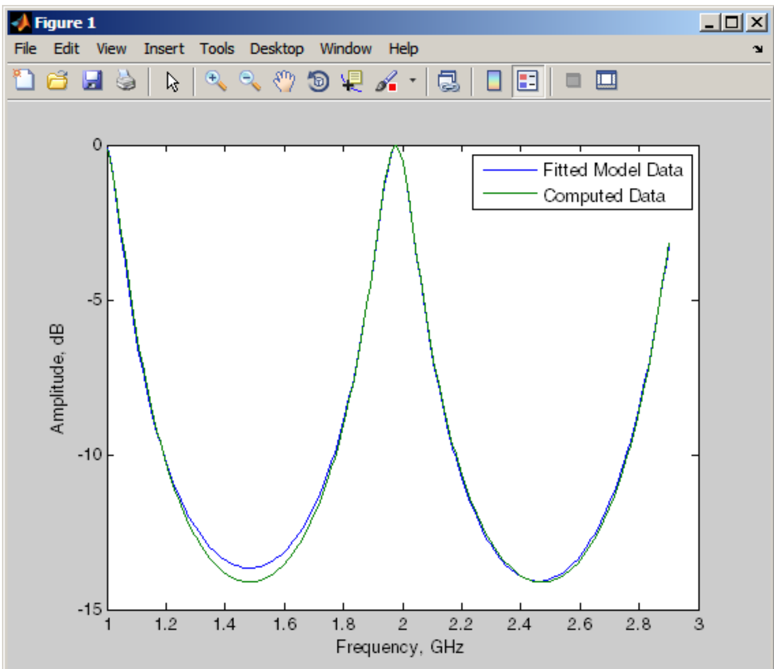
```
[fresp,freq]=freqresp(RationalFunc,Freq);
```

3 Type the following set of commands at the MATLAB prompt to plot the amplitude of the frequency response of the fitted model data and that of the computed data:

```
figure
plot(freq/1e9,db(fresp),freq/1e9,db(TrFunc));
```

```
xlabel('Frequency, GHz')
ylabel('Amplitude, dB')
legend('Fitted Model Data','Computed Data')
```

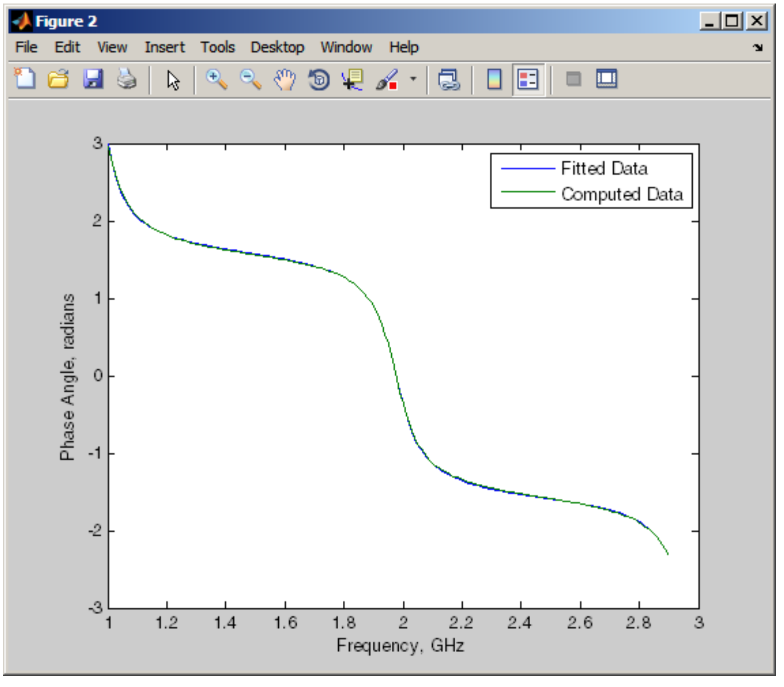
Note The amplitude of the model data is very close to the amplitude of the computed data. You can control the tradeoff between model accuracy and model complexity by specifying the optional tolerance argument, `tol`, to the `rationalfit` function, as described in "Represent a Circuit Object with a Model Object" on page 4-5.



4 Type the following set of commands at the MATLAB prompt to plot the phase angle of the frequency response of the fitted model data and that of the computed data:

```
figure
plot(freq/1e9,unwrap(angle(fresp)),...
     freq/1e9,unwrap(angle(TrFunc)));
xlabel('Frequency, GHz')
ylabel('Phase Angle, radians')
legend('Fitted Data','Computed Data')
```

Note The phase angle of the model data is very close to the phase angle of the computed data.



Compute and Plot the Time-Domain Response

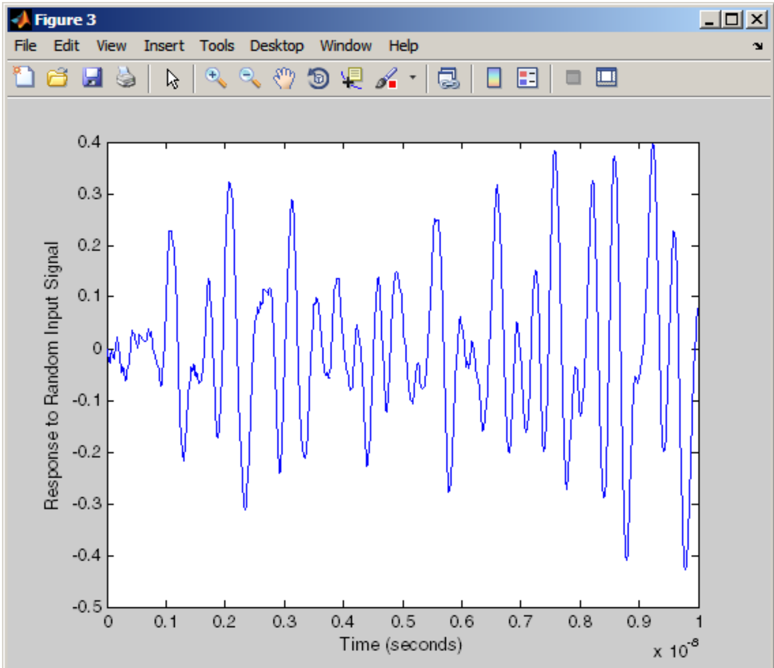
In this part of the example, you compute and plot the time-domain response of the transmission line.

1 Type the following set of commands at the MATLAB prompt to create a random input signal and compute the time response, `tresp`, of the fitted model data to the input signal:

```
SampleTime=1e-12;
NumberOfSamples=1e4;
OverSamplingFactor = 25;
InputTime = double((1:NumberOfSamples)')*SampleTime;
InputSignal = ...
     sign(randn(1, ceil(NumberOfSamples/OverSamplingFactor)));
InputSignal = repmat(InputSignal, [OverSamplingFactor, 1]);
InputSignal = InputSignal(:);

[tresp,t]=timeresp(RationalFunc,InputSignal,SampleTime);
```

2 Type the following set of commands at the MATLAB prompt to plot the time response of the fitted model data:

```
figure
plot(t,tresp);
xlabel('Time (seconds)')
ylabel('Response to Random Input Signal')
```



Export a Verilog-A Model

In this part of the example, you export a Verilog-A model of the transmission line. You can use this model in other simulation tools for detailed time-domain analysis and system simulations.

The following code illustrates how to use the `writeva` method to write a Verilog-A module for `RationalFunc` to the file `tline.va`. The module has one input, `tline_in`, and one output, `tline_out`. The method returns a `status` of `True`, if the operation is successful, and `False` if it is unsuccessful.

```
status = writeva(RationalFunc,'tline','tline_in','tline_out')
```

For more information on the `writeva` method and its arguments, see the `writeva` reference page. For more information on Verilog-A models, see “Export a Verilog-A Model” on page 4-5.

RF Objects

RF Data Objects

In this section...

Overview

RF Toolbox software uses data (`rfdata`) objects to store:

- Component data created from files or from information that you specify in the MATLAB workspace.
- Analyzed data from a frequency-domain simulation of a circuit object.

You can perform basic tasks, such as plotting and network parameter conversion, on the data stored in these objects. However, data objects are primarily used to store data for use by other RF objects.

Types of Data

The toolbox uses RF data objects to store one or more of the following types of data:

- Network parameters
- Spot noise
- Noise figure
- Third-order intercept point (IP3)
- Power out versus power in

Available Data Objects

The following table lists the available `rfdata` object constructors and describes the data the corresponding objects represent. For more information on a particular object, follow the link in the table to the reference page for that object.

Constructor	Description
`rfdata.data`	Data object containing network parameter data
`rfdata.ip3`	Data object containing IP3 information
`rfdata.mixerspur`	Data object containing mixer spur information from an intermodulation table
`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

Data Object Methods

The following table lists the methods of the data objects, the types of objects on which each can act, and the purpose of each method.

Method	Types of Objects	Purpose
`extract`	`rfdata.data`, `rfdata.network`	Extract specified network parameters from a circuit or data object and return the result in an array
`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.

RF Circuit Objects

In this section...

Overview of RF Circuit Objects

RF Toolbox software uses circuit (`rfckt`) objects to represent the following components:

- Circuit components such as amplifiers, transmission lines, and ladder filters
- RLC network components
- Networks of RF components

The toolbox represents each type of component and network with a different object. You use these objects to analyze components and networks in the frequency domain.

Components Versus Networks

You define component behavior using network parameters and physical properties.

To specify an individual RF component:

1. Construct a circuit object to represent the component.

2. Specify or import component data.

You define network behavior by specifying the components that make up the network. These components can be either individual components (such as amplifiers and transmission lines) or other networks.

To specify an RF network:

1 Build circuit objects to represent the network components.

2 Construct a circuit object to represent the network.

> **Note** This object defines how to connect the network components. However, the network is empty until you specify the components that it contains.

3 Specify, as the `Ckts` property of the object that represents the network, a list of components that make up the network.

These procedures are illustrated by example in "Model a Cascaded RF Network" on page 1-11.

Available Components and Networks

To create circuit objects that represent components, you use constructors whose names describe the components. To create circuit objects that represent networks, you use constructors whose names describe how the components are connected together.

The following table lists the available `rfckt` object constructors and describes the components or networks the corresponding objects represent. For more information on a particular object, follow the link in the table to the reference page for that object.

Constructor	Description
`rfckt.amplifier`	Amplifier, described by an `rfdata` object
`rfckt.cascade`	Cascaded network, described by the list of components and networks that comprise it
`rfckt.coaxial`	Coaxial transmission line, described by dimensions and electrical characteristics

Constructor	Description
`rfckt.cpw`	Coplanar waveguide transmission line, described by dimensions and electrical characteristics
`rfckt.datafile`	General circuit, described by a data file
`rfckt.delay`	Delay line, described by loss and delay
`rfckt.hybrid`	Hybrid connected network, described by the list of components and networks that comprise it
`rfckt.hybridg`	Inverse hybrid connected network, described by the list of components and networks that comprise it
`rfckt.lcbandpasspi`	LC bandpass pi network, described by LC values
`rfckt.lcbandpasstee`	LC bandpass tee network, described by LC values
`rfckt.lcbandstoppi`	LC bandstop pi network, described by LC values
`rfckt.lcbandstoptee`	LC bandstop tee network, described by LC values
`rfckt.lchighpasspi`	LC highpass pi network, described by LC values
`rfckt.lchighpasstee`	LC highpass tee network, described by LC values
`rfckt.lclowpasspi`	LC lowpass pi network, described by LC values
`rfckt.lclowpasstee`	LC lowpass tee network, described by LC values
`rfckt.microstrip`	Microstrip transmission line, described by dimensions and electrical characteristics
`rfckt.mixer`	Mixer, described by an `rfdata` object
`rfckt.parallel`	Parallel connected network, described by the list of components and networks that comprise it
`rfckt.parallelplate`	Parallel-plate transmission line, described by dimensions and electrical characteristics

Constructor	Description
`rfckt.passive`	Passive component, described by network parameters
`rfckt.rlcgline`	RLCG transmission line, described by RLCG values
`rfckt.series`	Series connected network, described by the list of components and networks that comprise it
`rfckt.seriesrlc`	Series RLC network, described by RLC values
`rfckt.shuntrlc`	Shunt RLC network, described by RLC values
`rfckt.twowire`	Two-wire transmission line, described by dimensions and electrical characteristics
`rfckt.txline`	General transmission line, described by dimensions and electrical characteristics

Circuit Object Methods

The following table lists the methods of the circuit objects, the types of objects on which each can act, and the purpose of each method.

Method	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.
`extract`	All circuit objects	Extract specified network parameters from a circuit or data object, and return the result in an array.
`getdata`	All circuit objects	Get data object containing analyzed result of a specified circuit object.

Method	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.
`loglog`	All circuit objects	Plot specified circuit object parameters using a log-log scale.
`plot`	All circuit objects	Plot the specified circuit object parameters on an X-Y plane.
`plotyy`	All circuit objects	Plot the specified object parameters with y-axes on both the left and right sides.
`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.
`semilogx`	All circuit objects	Plot the specified circuit object parameters using a log scale for the X-axis

Method	Types of Objects	Purpose
`semilogy`	All circuit objects	Plot the specified circuit object parameters using a log scale for the Y-axis
`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.

RF Model Objects

In this section...

Overview of RF Model Objects

RF Toolbox software uses model (`rfmodel`) objects to represent components and measured data mathematically for computing information such as time-domain response. Each type of model object uses a different mathematical model to represent the component.

RF model objects provide a high-level component representation for use after you perform detailed analysis using RF circuit objects. Use RF model objects to:

- Compute time-domain figures of merit for RF components
- Export Verilog-A models of RF components

Available Model Objects

The following table lists the available `rfmodel` object constructors and describes the model the corresponding objects use. For more information on a particular object, follow the link in the table to the reference page for that object.

Constructor	Description
`rfmodel.rational`	Rational function model

Model Object Methods

The following table lists the methods of the model objects, the types of objects on which each can act, and the purpose of each method.

Method	Types of Objects	Purpose
`freqresp`	All model objects	Compute the frequency response of a model object.
`timeresp`	All model objects	Compute the time response of a model object.
`writeva`	All model objects	Write data from a model object to a file.

RF Network Parameter Objects

In this section...

Overview of Network Parameter Objects

RF Toolbox software offers network parameter objects for:

- Importing network parameter data from a Touchstone file.
- Converting network parameters.
- Analyzing network parameter data.

Unlike circuit, model, and data objects, you can use existing RF Toolbox functions to operate directly on network parameter objects.

Available Network Parameter Objects

The following table lists the available network parameter objects and the functions that are used to construct them. For more information on a particular object, follow the link in the table to the reference page for that functions.

Network Parameter Object Type	Network Parameter Object Function
ABCD Parameter object	`abcdparameters`
Hybrid-g parameter object	`gparameters`
Hybrid parameter object	`hparameters`
S-parameter object	`sparameters`
Y-parameter object	`yparameters`
Z-parameter object	`zparameters`

Network Parameter Object Functions

The following table lists the functions that accept network parameter objects as inputs, the types of objects on which each can act, and the purpose of each function.

Function	Types of Objects	Purpose
`abcdparameters`	All network parameter objects	Convert any network parameters to ABCD parameters
`gparameters`	All network parameter objects	Convert any network parameters to hybrid-g parameters
`hparameters`	All network parameter objects	Convert any network parameters to hybrid parameters
`sparameters`	All network parameter objects	Convert any network parameters to S-parameters
`yparameters`	All network parameter objects	Convert any network parameters to Y-parameters
`zparameters`	All network parameter objects	Convert any network parameters to Z-parameters
`cascadesparams`	S-parameter objects	Cascade S-parameters
`deembedsparams`	S-parameter objects	De-embed S-parameters
`gammain`	S-parameter objects	Calculate input reflection coefficient
`gammaml`	S-parameter objects	Calculate load reflection coefficient
`gammams`	S-parameter objects	Calculate source reflection coefficient

Function	Types of Objects	Purpose
`gammaout`	S-parameter objects	Calculate output reflection coefficient
`ispassive`	S-parameter objects	Check S-parameter data passivity
`makepassive`	S-parameter objects	Make S-parameter data passive
`newref`	S-parameter objects	Change reference impedance
`powergain`	S-parameter objects	Calculate power gain
`rfplot`	S-parameter objects	Plot network parameters
`rfinterp1`	All network parameter objects	Interpolate network parameters at new frequencies
`rfparam`	All network parameter objects	Extract vector of network parameters
`s2tf`	S-parameter objects	Create transfer function from S-parameters
`stabilityk`	S-parameter objects	Calculate stability factor K of 2-port network
`stabilitymu`	S-parameter objects	Calculate stability factor μ of 2-port network
`smith`	All network parameter objects	Plot network parameter data on a Smith Chart

3

Model an RF Component

Create RF Objects

In this section...

Construct a New Object

You can create any `rfdata`, `rfckt` or `rfmodel` object by calling the object constructor. You can create an `rfmodel` object by fitting a rational function to passive component data.

This section contains the following topics:

Call the Object Constructor

To create a new RF object with default property values, you call the object constructor without any arguments:

```
h = objecttype.objectname
```

where:

- `h` is the handle to the new object.
- *`objecttype`* is the object type (`rfdata`, `rfckt`, or `rfmodel`).
- *`objectname`* is the object name.

For example, to create an RLCG transmission line object, type:

```
h = rfckt.rlcgline
```

because the RLCG transmission line object is a circuit (`rfckt`) object named `rlcgline`.

The following code illustrates how to call the object constructor to create a microstrip transmission line object with default property values. The output `t1` is the handle of the newly created transmission line object.

```
t1 = rfckt.microstrip
```

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

```
t1 =
           Name: 'Microstrip Transmission Line'
          nPort: 2
 AnalyzedResult: []
     LineLength: 0.0100
       StubMode: 'NotAStub'
    Termination: 'NotApplicable'
          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.

Fit a Rational Function to Passive Component Data

You can create a model object by fitting a rational function to passive component data. You use this approach to create a model object that represents one of the following using a rational function:

- A circuit object that you created and analyzed.
- Data that you imported from a file.

For more information, see "Fit a Model Object to Circuit Object Data" on page 3-36.

Copy an Existing Object

You can create a new object with the same property values as an existing object by using the `copy` function to copy the existing object. This function is useful if you have an object that is similar to one you want to create.

For example,

```
t2 = copy(t1);
```

creates a new object, `t2`, which has the same property values as the microstrip transmission line object, `t1`.

You can later change specific property values for this copy. For information on modifying object properties, see "Specify or Import Component Data" on page 3-5.

Note The syntax `t2 = t1` copies only the object handle and does not create a new object.

Specify or Import Component Data

In this section...

RF Object Properties

Object properties specify the behavior of an object. You can specify object properties, or you can import them from a data file. To learn about properties that are specific to a particular type of circuit, data, or model object, see the reference page for that type of object.

Note The "RF Circuit Objects" on page 2-4, "RF Data Objects" on page 2-2,"RF Model Objects" on page 2-10 sections list the available types of objects and provide links to their reference pages.

Set Property Values

You can specify object property values when you construct an object or you can modify the property values of an existing object.

This section contains the following topics:

Specify Property Values at Construction

To set a property when you construct an object, include a comma-separated list of one or more property/value pairs in the argument list of the object construction command. A property/value pair consists of the arguments `'PropertyName',PropertyValue`, where:

- `PropertyName` is a string specifying the property name. The name is case-insensitive. In addition, you need only type enough letters to uniquely identify the property name. For example, `'st'` is sufficient to refer to the `StubMode` property.

Note You must use single quotation marks around the property name.

- `PropertyValue` is the value to assign to the property.

Include as many property names in the argument list as there are properties you want to set. Any property values that 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 section contains examples of how to perform the following tasks:

- "Construct Components with Specified Properties" on page 3-6
- "Construct Networks of Specified Components" on page 3-7

Construct Components with Specified Properties. The following code creates a coaxial transmission line circuit object to represent a coaxial transmission line that is 0.05 meters long. Notice that the toolbox lists the available properties and their values.

```
t1 = rfckt.coaxial('LineLength',0.05)

t1 =

              Name: 'Coaxial Transmission Line'
             nPort: 2
    AnalyzedResult: []
        LineLength: 0.0500
          StubMode: 'NotAStub'
       Termination: 'NotApplicable'
       OuterRadius: 0.0026
```

```
        InnerRadius: 7.2500e-004
                MuR: 1
           EpsilonR: 2.3000
        LossTangent: 0
          SigmaCond: Inf
```

Construct Networks of Specified Components. To combine a set of RF components and existing networks to form an RF network, you create a network object with the `Ckts` property set to an array containing the handles of all the circuit objects in the network.

Suppose you have the following RF components:

```
t1 = rfckt.coaxial('LineLength',0.05);
a1 = rfckt.amplifier;
t2 = rfckt.coaxial('LineLength',0.1);
```

The following code creates a cascaded network of these components:

```
casc_network = rfckt.cascade('Ckts',{t1,a1,t2});
```

Change Property Values of an Existing Object

There are two ways to change the properties of an existing object:

- Using the `set` command
- Using structure-like assignments called dot notation

This section discusses the first option. For details on the second option, see "Reference Properties Directly Using Dot Notation" on page 3-16.

To modify the properties of an existing object, use the `set` command with one or more property/value pairs in the argument list. The general syntax of the command is

```
set(h,Property1',value1,'Property2',value2,...)
```

where

- `h` is the handle of the object.

- *'Property1',value1,'Property2',value2,...* is the list of property/value pairs.

For example, the following code creates a default coaxial transmission line object and changes it to a series stub with open termination.

```
t1 = rfckt.coaxial;
set(t1,'StubMode','series','Termination','open')
```

Note You can use the `set` command without specifying any property/value pairs to display a list of all properties you can set for a specific object. This example lists the properties you can set for the coaxial transmission line `t1`:

```
set(t1)

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

Import Property Values from Data Files

RF Toolbox software lets you import industry-standard data files, MathWorks AMP files, and Agilent P2D and S2D files into specific objects. This import capability lets you simulate the behavior of measured components.

You can import the following file formats:

- Industry-standard file formats — Touchstone SNP, YNP, ZNP, HNP, and GNP formats specify the network parameters and noise information for measured and simulated data.

For more information on Touchstone files, see `www.vhdl.org/pub/ibis/connector/touchstone_spec11.pdf`.

- Agilent P2D file format — Specifies amplifier and mixer large-signal, power-dependent network parameters, noise data, and intermodulation tables for several operating conditions, such as temperature and bias values.

  The P2D file format lets you import system-level verification models of amplifiers and mixers.

- Agilent S2D file format — Specifies amplifier and mixer network parameters with gain compression, power-dependent S_{21} parameters, noise data, and intermodulation tables for several operating conditions.

  The S2D file format lets you import system-level verification models of amplifiers and mixers.

- MathWorks amplifier (AMP) file format — Specifies amplifier network parameters, output power versus input power, noise data and third-order intercept point.

  For more information about `.amp` files, see Appendix A, "AMP File Format".

This section contains the following topics:

- "Objects Used to Import Data from a File" on page 3-9
- "How to Import Data Files" on page 3-10

Objects Used to Import Data from a File

One data object and three circuit objects accept data from a file. The following table lists the objects and any corresponding data format each supports.

Object	Description	Supported Format(s)
`rfdata.data`	Data object containing network parameter data, noise figure, and third-order intercept point	Touchstone, AMP, P2D, S2D
`rfckt.amplifier`	Amplifier	Touchstone, AMP, P2D, S2D

Object	Description	Supported Format(s)
`rfckt.mixer`	Mixer	Touchstone, AMP, P2D, S2D
`rfckt.passive`	Generic passive component	Touchstone

How to Import Data Files

To import file data into a circuit or data object at construction, use a `read` command of the form:

```
obj = read(obj_type,'filename');
```

where

- `obj` is the handle of the circuit or data object.
- *`obj_type`* is the type of object in which to store the data, from the list of objects that accept file data shown in "Objects Used to Import Data from a File" on page 3-9.
- *`filename`* is the name of the file that contains the data.

For example,

```
ckt_obj=read(rfckt.amplifier, 'default.amp');
```

imports data from the file `default.amp` into an `rfckt.amplifier` object.

You can also import file data into an existing circuit object. The following commands are equivalent to the previous command:

```
ckt_obj=rfckt.amplifier;
read(ckt_obj, 'default.amp');
```

Note When you import component data from a `.p2d` or `.s2d` file, properties are defined for several operating conditions. You must select an operating condition to specify the object behavior, as described in "Specify Operating Conditions" on page 3-17.

Use Data Objects to Specify Circuit Properties

To specify a circuit object property using a data object, use the `set` command with the name of the data object as the value in the property/value pair.

For example, suppose you have the following `rfckt.amplifier` and `rfdata.nf` objects:

```
amp = rfckt.amplifier
f = 2.0e9;
nf = 13.3244;
nfdata = rfdata.nf('Freq',f,'Data',nf)
```

The following command uses the `rfdata.nf` data object to specify the `rfckt.amplifier` `NoiseData` property:

```
set(amp,'NoiseData',nfdata)
```

Set Circuit Object Properties Using Data Objects

In this example, you create a circuit object. 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. These properties control the frequency response of the amplifier, which is stored in the `AnalyzedResult` property. By default, all amplifier 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.** Type the following set of commands at the MATLAB prompt to create an `rfdata.network` object that stores the 2-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.** Type the following set of commands at the MATLAB prompt 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.** Type the following command at the MATLAB prompt to create a `rfdata.ip3` object that contains an output third-order intercept point of 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.** Type the following set of commands at the MATLAB prompt 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;
```

Retrieve Property Values

You can retrieve one or more property values of an existing object using the `get` command.

This section contains the following topics:

- "Retrieve Specified Property Values" on page 3-14
- "Retrieve All Property Values" on page 3-15

Retrieve Specified Property Values

To retrieve specific property values for an object, use the `get` command with the following syntax:

```
PropertyValue = get(h,PropertyName)
```

where

- *`PropertyValue`* is the value assigned to the property.
- `h` is the handle of the object.
- *`PropertyName`* is a string specifying the property name.

For example, suppose you have the following coaxial transmission line:

```
h2 = rfckt.coaxial;
```

The following code retrieves the value of the inner radius and outer radius for the coaxial transmission line:

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

ir =
  7.2500e-004

or =
    0.0026
```

Retrieve All Property Values

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.

For example:

```
get(h2)
              Name: 'Coaxial Transmission Line'
             nPort: 2
    AnalyzedResult: []
        LineLength: 0.0100
          StubMode: 'NotAStub'
       Termination: 'NotApplicable'
       OuterRadius: 0.0026
       InnerRadius: 7.2500e-004
               MuR: 1
          EpsilonR: 2.3000
       LossTangent: 0
         SigmaCond: Inf
```

Note 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`, and `AnalyzedResult`. The `Name` and `nPort` properties are fixed by the toolbox. The `AnalyzedResult` property value is calculated and set by the toolbox when you analyze the component at specified frequencies.

Reference Properties Directly Using Dot Notation

An alternative way to query for or modify property values is by structure-like referencing. The field names for RF objects are the property names, so you can retrieve or modify property values with the structure-like syntax.

- `PropertyValue = rfobj.PropertyName` stores the value of the `PropertyName` property of the `rfobj` object in the `PropertyValue` variable. This command is equivalent to `PropertyValue = get(rfobj,'PropertyName')`.
- `rfobj.PropertyName = PropertyValue` sets the value of the `PropertyName` property to `PropertyValue` for the `rfobj` object. This command is equivalent to `set(rfobj,'PropertyName',PropertyValue)`.

For example, typing

```
ckt = rfckt.amplifier('IntpType','cubic');
ckt.IntpType
```

gives the value of the property `IntpType` for the circuit object `ckt`.

```
ans =
  Cubic
```

Similarly,

```
ckt.IntpType = 'linear';
```

resets the interpolation method to linear.

You do not need to type the entire field name or use uppercase characters. You only need to type the minimum number of characters sufficient to identify the property name uniquely. Thus entering the commands

```
ckt = rfckt.amplifier('IntpType','cubic');
ckt.in
```

also produces

```
ans =
  Cubic
```

Specify Operating Conditions

Available Operating Conditions

Agilent P2D and S2D files contain simulation results at one or more operating conditions. Operating conditions define the independent parameter settings that are used when creating the file data. The specified conditions differ from file to file.

When you import component data from a `.p2d` or `.s2d` file, the object contains property values for several operating conditions. The available conditions depend on the data in the file. By default, RF Toolbox software defines the object behavior using the property values that correspond to the operating conditions that appear first in the file. To use other property values, you must select a different operating condition.

Set Operating Conditions

To set the operating conditions of a circuit or data object, use a `setop` command of the form:

```
setop(,'Condition1',value1,...,'ConditionN',valueN,...)
```

where

- is the handle of the circuit or data object.
- *`Condition1`*,*`value1`*,...,*`ConditionN`*,*`valueN`* are the condition/value pairs that specify the operating condition.

For example,

```
setop(myp2d, 'BiasL', 2, 'BiasU', 6.3)
```

specifies an operating condition of `BiasL = 2` and `BiasU = 6.3` for *`myp2d`*.

Display Available Operating Condition Values

To display a list of available operating condition values for a circuit or data object, use the `setop` method.

```
setop(obj)
```

displays the available values for all operating conditions of the object `obj`.

```
setop(obj,'Condition1')
```

displays the available values for *`Condition1`*.

Process File Data for Analysis

In this section...

Convert Single-Ended S-Parameters to Mixed-Mode S-Parameters

After you import file data (as described in "Import Property Values from Data Files" on page 3-8), you can convert a matrix of single-ended S-parameter data to a matrix of mixed-mode S-parameters.

This section contains the following topics:

Functions for Converting S-Parameters

To convert between 4-port single-ended S-parameter data and 2-port differential-, common-, and cross-mode S-parameters, use one of these functions:

- `s2scc` — Convert 4-port, single-ended S-parameters to 2-port, common-mode S-parameters (S_{cc}).
- `s2scd` — Convert 4-port, single-ended S-parameters to 2-port, cross-mode S-parameters (S_{cd}).
- `s2sdc` — Convert 4-port, single-ended S-parameters to cross-mode S-parameters (S_{dc}).
- `s2sdd` — Convert 4-port, single-ended S-parameters to 2-port, differential-mode S-parameters (S_{dd}).

To perform the above conversions all at once, or to convert larger data sets, use one of these functions:

- `s2smm` — Convert 4N-port, single-ended S-parameters to 2N-port, mixed-mode S-parameters.
- `smm2s` — Convert 2N-port, mixed-mode S-parameters to 4N-port, single-ended S-parameters.

Conversion functions support a variety of port orderings. For more information on these functions, see the corresponding reference pages.

Convert S-Parameters

In this example, use the toolbox to import 4-port single-ended S-parameter data from a file, convert the data to 2-port differential S-parameter data, and create a new `rfckt` object to store the converted data for analysis.

At the MATLAB prompt:

1 Type this command to import data from the file `default.s4p`:

```
SingleEnded4Port = read(rfdata.data,'default.s4p');
```

2 Type this command to convert 4-port single-ended S-parameters to 2-port mixed-mode S-parameters:

```
DifferentialSParams = s2sdd(SingleEnded4Port.S_Parameters);
```

Note The S-parameters that you specify as input to the `s2sdd` function are the ones the toolbox stores in the `S_Parameters` property of the `rfdata.data` object.

---

3 Type this command to create an `rfckt.passive` object that stores the 2-port differential S-parameters for simulation:

```
DifferentialCkt = rfckt.passive('NetworkData', ...
    rfdata.network('Data', DifferentialSParams, 'Freq', ...
    SingleEnded4PortData.Freq));
```

Extract M-Port S-Parameters from N-Port S-Parameters

After you import file data (as described in "Import Property Values from Data Files" on page 3-8), you can extract a set of data with a smaller number of ports by terminating one or more ports with a specified impedance.

This section contains the following topics:

Extract S-Parameters

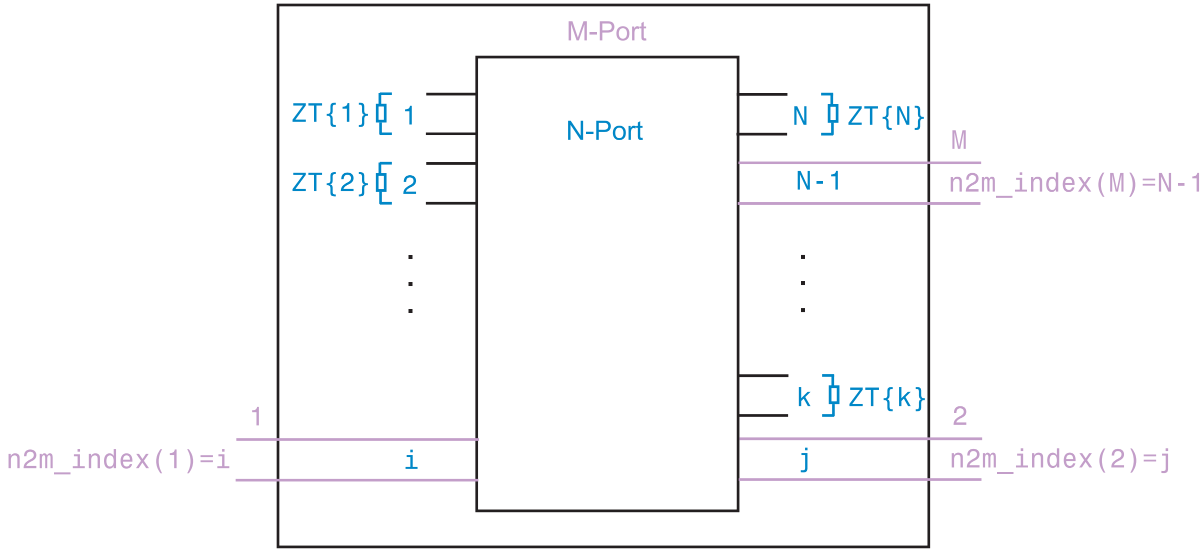
To extract M-port S-parameters from N-port S-parameters, use the `snp2smp` function with the following syntax:

```
s_params_mp = snp2smp(s_params_np, z0, n2m_index, zt)
```

where

- *`s_params_np`* is an array of *N*-port S-parameters with a reference impedance *`z0`*.
- `s_params_mp` is an array of *M*-port S-parameters.
- *`n2m_index`* is a vector of length *M* specifying how the ports of the *N*-port S-parameters map to the ports of the *M*-port S-parameters. *`n2m_index(i)`* is the index of the port from *`s_params_np`* that is converted to the *`i`*th port of `s_params_mp`.
- *`zt`* is the termination impedance of the ports.

The following figure illustrates how to specify the ports for the output data and the termination of the remaining ports.



For more details about the arguments to this function, see the `snp2smp` reference page.

Extract S-Parameters From Imported File Data

In this example, use the toolbox to import 16-port S-parameter data from a file, convert the data to 4-port S-parameter data by terminating the remaining ports, and create a new `rfckt` object to store the extracted data for analysis.

At the MATLAB prompt:

1 Type this command to import data from the file `default.s16p` into an `rfdata.data` object, `SingleEnded16PortData`:

```
SingleEnded16PortData = read(rfdata.data,'default.s16p');
```

2 Type this command to convert 16-port S-parameters to 4-port S-parameters by using ports 1, 16, 2, and 15 as the first, second, third, and fourth ports, and terminating the remaining 12 ports with an impedance of 50 ohms:

```
N2M_index = [1 16 2 15];
FourPortSParams = snp2smp(SingleEnded16PortData.S_Parameters, ...
```

```
SingleEnded16PortData.Z0, N2M_index, 50);
```

Note The S-parameters that you specify as input to the `snp2smp` function are the ones the toolbox stores in the `S_Parameters` property of the `rfdata.data` object.

---

3 Type this command to create an `rfckt.passive` object that stores the 4-port S-parameters for simulation:

```
FourPortChannel = rfckt.passive('NetworkData', ...
    rfdata.network('Data', FourPortSParams, 'Freq', ...
    SingleEnded16PortData.Freq));
```

Cascade N-Port S-Parameters

After you import file data (as described in "Import Property Values from Data Files" on page 3-8), you can cascade two or more networks of N-port S-parameters.

To cascade networks of N-port S-parameters, use the `cascadesparams` function with the following syntax:

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

where

- `s_params` is an array of cascaded S-parameters.
- *`s1_params,s2_params,...,sn_params`* are arrays of input S-parameters.
- *`nconn`* is a positive scalar or a vector of size `n-1` specifying how many connections to make between the ports of the input S-parameters. `cascadesparams` connects the last port(s) of one network to the first port(s) of the next network.

For more details about the arguments to this function, see the `cascadesparams` reference page.

Import and Cascade N-Port S-Parameters

In this example, use the toolbox to import 16-port and 4-port S-parameter file data and cascade the two S-parameter networks by connecting the last three ports of the 16-port network to the first three ports of the 4-port network. Then, create a new `rfckt` object to store the resulting network for analysis.

At the MATLAB prompt:

1 Type these commands to import data from the files `default.s16p` and `default.s4p`, and create the 16- and 4-port networks of S-parameters:

```
S_16Port = read(rfdata.data,'default.s16p');
S_4Port = read(rfdata.data,'default.s4p');
freq = [2e9 2.1e9];
analyze(S_16Port, freq);
analyze(S_4Port, freq);
sparams_16p = S_16Port.S_Parameters;
sparams_4p = S_4Port.S_Parameters;
```

2 Type this command to cascade 16-port S-parameters and 4-port S-parameters by connecting ports 14, 15, and 16 of the 16-port network to ports 1, 2, and 3 of the 4-port network:

```
sparams_cascaded = cascadesparams(sparams_16p, sparams_4p,3)
```

`cascadesparams` creates a 14-port network. Ports 1–13 are the first 13 ports of the 16-port network. Port 14 is the fourth port of the 4-port network.

3 Type this command to create an `rfckt.passive` object that stores the 14-port S-parameters for simulation:

```
Ckt14 = rfckt.passive('NetworkData', ...
    rfdata.network('Data', sparams_cascaded, 'Freq', ...
    freq));
```

For more examples of how to use this function, see the `cascadesparams` reference page.

Analyze and Plot RF Components

In this section...

Analyze Networks in the Frequency Domain

RF Toolbox software lets you analyze RF components and networks in the frequency domain. You use the `analyze` method to analyze a circuit object over a specified set of frequencies.

For example, to analyze a coaxial transmission line from 1 GHz to 2.9 GHz in increments of 10 MHz:

```
ckt = rfckt.coaxial;
f = [1.0e9:1e7:2.9e9];
analyze(ckt,f);
```

Note For all circuits objects except those that contain data from a file, you must perform a frequency-domain analysis with the `analyze` method before visualizing component and network data. For circuits that contain data from a file, the toolbox performs a frequency-domain analysis when you use the `read` method to import the data.

When you analyze a circuit object, the toolbox computes the circuit network parameters, noise figure values, and output third-order intercept point (OIP3) values at the specified frequencies and stores the result of the analysis in the object's `AnalyzedResult` property.

For more information, see the `analyze` reference page or the circuit object reference page.

Visualize Component and Network Data

The toolbox lets you validate the behavior of circuit objects that represent RF components and networks by plotting the following data:

- Large- and small-signal S-parameters
- Noise figure
- Output third-order intercept point
- Power data
- Phase noise
- Voltage standing-wave ratio
- Power gain
- Group delay
- Reflection coefficients
- Stability data
- Transfer function

The following table summarizes the available plots and charts, along with the methods you can use to create each one and a description of its contents.

Plot Type	Methods	Plot Contents

Rectangular Plot	`plot`
	`plotyy`
	`loglog`
	`semilogx`
	`semilogy`
		Parameters as a function of frequency or, where applicable, operating condition. The available parameters include:
		• S-parameters
		• Noise figure
		• Voltage standing-wave ratio (VSWR)
		• OIP3

Budget Plot (3-D)	`plot`	Parameters as a function of frequency for each component

Plot Type	Methods	Plot Contents
		in a cascade, where the curve for a given component represents the cumulative contribution of each RF component up to and including the parameter value of that component.
Mixer Spur Plot	`plot`	Mixer spur power as a function of frequency for an `rfckt.mixer` object or an `rfckt.cascade` object that contains a mixer.
Polar Plot	`polar`	Magnitude and phase of S-parameters as a function of frequency.
Smith Chart	`smith`	Real and imaginary parts of S-parameters as a function of frequency, used for analyzing the reflections caused by impedance mismatch.

For each plot you create, you choose a parameter to plot and, optionally, a format in which to plot that parameter. The plot format defines how the toolbox displays the data on the plot. The available formats vary with the data you select to plot. The data you can plot depends on the type of plot you create.

Note You can use the `listparam` method to list the parameters of a specified circuit object that are available for plotting. You can use the `listformat` method to list the available formats for a specified circuit object parameter.

The following topics describe the available plots:

- “Rectangular” on page 3-28
- “Budget” on page 3-28

- "Mixer Spur" on page 3-31
- "Polar Plots and Smith Charts" on page 3-34

Rectangular

You can plot any parameters that are relevant to your object on a rectangular plot. You can plot parameters as a function of frequency for any object. When you import object data from a `.p2d` or `.s2d` file, you can also plot parameters as a function of any operating condition from the file that has numeric values, such as bias. In addition, when you import object data from a `.p2d` file, you can plot large-signal S-parameters as a function of input power or as a function of frequency. These parameters are denoted `LS11`, `LS12`, `LS21`, and `LS22`.

The following table summarizes the methods that are available in the toolbox for creating rectangular plots and describes the uses of each one. For more information on a particular type of plot, follow the link in the table to the documentation for that method.

Method	Description
`plot`	Plot of one or more object parameters
`plotyy`	Plot of one or more object parameters with y-axes on both the left and right sides
`semilogx`	Plot of one or more object parameters using a log scale for the X-axis
`semilogy`	Plot of one or more object parameters using a log scale for the Y-axis
`loglog`	Plot of one or more object parameters using a log-log scale

Budget

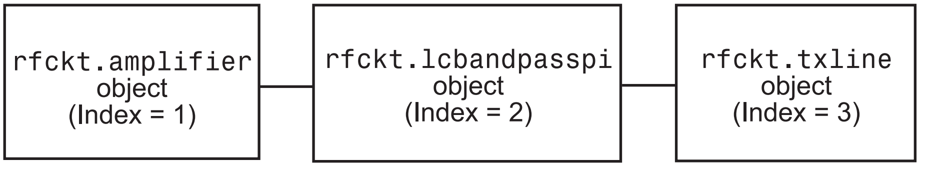
You use the link budget plot to understand the individual contribution of each component to a plotted parameter value in a cascaded network with multiple components.

The budget plot is a three-dimensional plot that shows one or more curves of parameter values as a function of frequency, ordered by the circuit index of the cascaded network.

Consider the following cascaded network:

```
casc = rfckt.cascade('Ckts',...
       {rfckt.amplifier,rfckt.lcbandpasspi,rfckt.txline})
```

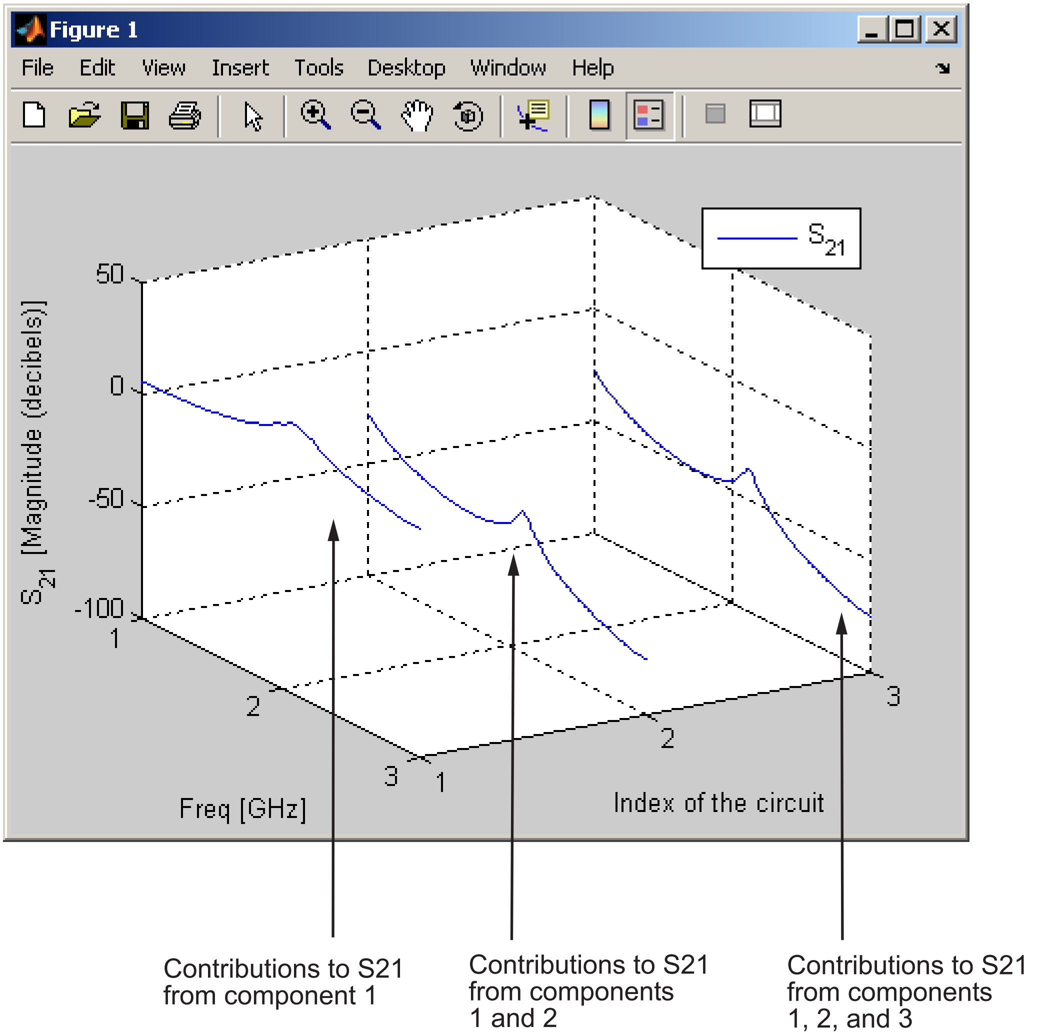
The following figure shows how the circuit index is assigned to each component in the cascade, based on its sequential position in the network.



You create a budget plot for this cascade using the `plot` method with the second argument set to `'budget'`, as shown in the following command:

```
plot(casc,'budget','s21')
```

A curve on the link budget plot for each circuit index represents the contributions to the parameter value of the RF components up to that index. The following figure shows the budget plot.



Budget Plot

If you specify two or more parameters, the toolbox puts the parameters in a single plot. You can only specify a single format for all the parameters.

Mixer Spur

You use the mixer spur plot to understand how mixer nonlinearities affect output power at the desired mixer output frequency and at the intermodulation products that occur at the following frequencies:

$$f_{out} = N * f_{in} + M * f_{LO}$$

where

- f_{in} is the input frequency.
- f_{LO} is the local oscillator frequency.
- N and M are integers.

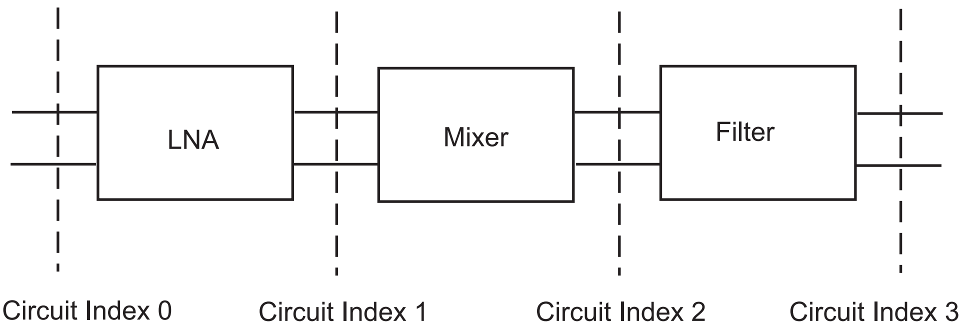
The toolbox calculates the output power from the mixer intermodulation table (IMT). These tables are described in detail in the Visualizing Mixer Spurs example.

The mixer spur plot shows power as a function of frequency for an `rfckt.mixer` object or an `rfckt.cascade` object that contains a mixer. By default, the plot is three-dimensional and shows a stem plot of power as a function of frequency, ordered by the circuit index of the object. You can create a two-dimensional stem plot of power as a function of frequency for a single circuit index by specifying the index in the mixer spur plot command.

Consider the following cascaded network:

```
FirstCkt = rfckt.amplifier('NetworkData', ...
    rfdata.network('Type', 'S', 'Freq', 2.1e9, ...
    'Data', [0,0;10,0]), 'NoiseData', 0, 'NonlinearData', inf);
SecondCkt = read(rfckt.mixer, 'samplespur1.s2d');
ThirdCkt = rfckt.lcbandpasstee('L', [97.21 3.66 97.21]*1e-9, ...
    'C', [1.63 43.25 1.63]*1.0e-12);
CascadedCkt = rfckt.cascade('Ckts', ...
    {FirstCkt, SecondCkt, ThirdCkt});
```

The following figure shows how the circuit index is assigned to the components in the cascade, based on its sequential position in the network.

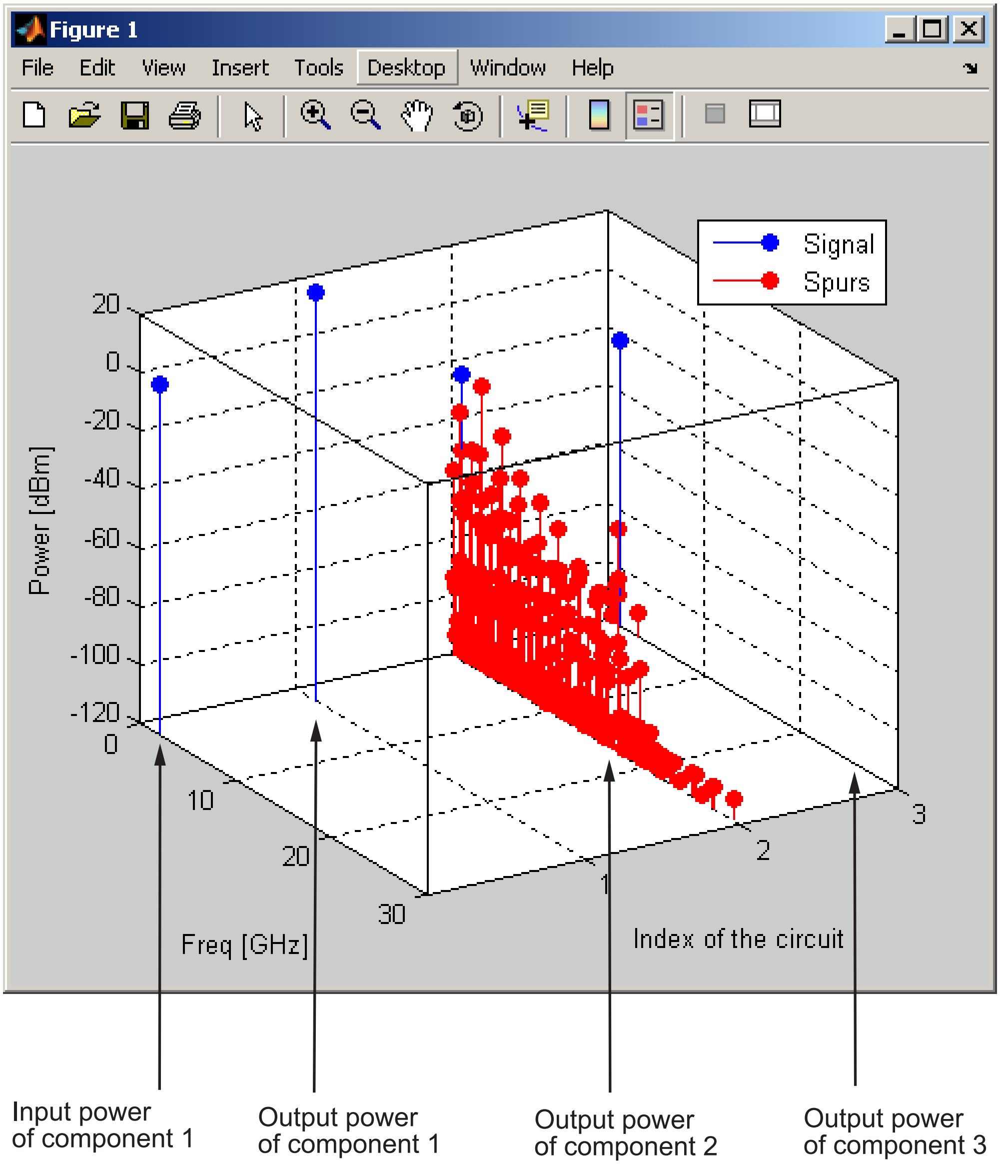


- Circuit index 0 corresponds to the cascade input.
- Circuit index 1 corresponds to the LNA output.
- Circuit index 2 corresponds to the mixer output.
- Circuit index 3 corresponds to the filter output.

You create a spur plot for this cascade using the `plot` method with the second argument set to `'mixerspur'`, as shown in the following command:

```
plot(CascadedCkt,'mixerspur')
```

Within the three dimensional plot, the stem plot for each circuit index represents the power at that circuit index. The following figure shows the mixer spur plot.



Mixer Spur Plot

For more information on mixer spur plots, see the `plot` reference page.

Polar Plots and Smith Charts

You can use the toolbox to generate Polar plots and Smith Charts. If you specify two or more parameters, the toolbox puts the parameters in a single plot.

The following table describes the Polar plot and Smith Chartt options, as well as the available parameters.

Note `LS11`, `LS12`, `LS21`, and `LS22` are large-signal S-parameters. You can plot these parameters as a function of input power or as a function of frequency.

Plot Type	Method	Parameter
`Polar plane`	`polar`	`S11`, `S12`, `S21`, `S22` `LS11`, `LS12`, `LS21`, `LS22` (Objects with data from a P2D file only)
`Z Smith chart`	`smith` with `type` argument set to `'z'`	`S11`, `S22` `LS11`, `LS22` (Objects with data from a P2D file only)
`Y Smith chart`	`smith` with `type` argument set to `'y'`	`S11`, `S22` `LS11`, `LS22` (Objects with data from a P2D file only)
`ZY Smith chart`	`smith` with `type` argument set to `'zy'`	`S11`, `S22` `LS11`, `LS22` (Objects with data from a P2D file only)

By default, the toolbox plots the parameter as a function of frequency. When you import block data from a `.p2d` or `.s2d` file, you can also plot parameters as a function of any operating condition from the file that has numeric values, such as bias.

Note The `circle` method lets you place circles on a Smith Chart to depict stability regions and display constant gain, noise figure, reflection and immittance circles. For more information about this method, see the `circle` reference page or the two-part RF Toolbox example about designing matching networks.

For more information on a particular type of plot, follow the link in the table to the documentation for that method.

Compute and Plot Time-Domain Specifications

The toolbox lets you compute and plot time-domain characteristics for RF components.

This section contains the following topics:

Compute the Network Transfer Function

You use the `s2tf` function to convert 2-port S-parameters to a transfer function. The function returns a vector of transfer function values that represent the normalized voltage gain of a 2-port network.

The following code illustrates how to read file data into a passive circuit object, extract the 2-port S-parameters from the object and compute the transfer function of the data at the frequencies for which the data is specified. `z0` is the reference impedance of the S-parameters, `zs` is the source impedance, and `zl` is the load impedance. See the `s2tf` reference page for more information on how these impedances are used to define the gain.

```
PassiveCkt = rfckt.passive('File','passive.s2p')
z0=50; zs=50; zl=50;
[SParams, Freq] = extract(PassiveCkt, 'S Parameters', z0);
TransFunc = s2tf(SParams, z0, zs, zl);
```

Fit a Model Object to Circuit Object Data

You use the `rationalfit` function to fit a rational function to the transfer function of a passive component. The `rationalfit` function returns an `rfmodel` object that represents the transfer function analytically.

The following code illustrates how to use the `rationalfit` function to create an `rfmodel.rational` object that contains a rational function model of the transfer function that you created in the previous example.

```
RationalFunc = rationalfit(Freq, TransFunc)
```

To find out how many poles the toolbox used to represent the data, look at the length of the `A` vector of the `RationalFunc` model object.

```
nPoles = length(RationalFunc.A)
```

Note The number of poles is important if you plan to use the RF model object to create a model for use in another simulator, because a large number of poles can increase simulation time. For information on how to represent a component accurately using a minimum number of poles, see "Represent a Circuit Object with a Model Object" on page 4-5.

See the `rationalfit` reference page for more information.

Use the `freqresp` method to compute the frequency response of the fitted data. To validate the model fit, plot the transfer function of the original data and the frequency response of the fitted data.

```
Resp = freqresp(RationalFunc, Freq);
plot(Freq, 20*log10(abs(TransFunc)), 'r', ...
    Freq, 20*log10(abs(Resp)), 'b--');
ylabel('Magnitude of H(s) (decibels)');
xlabel('Frequency (Hz)');
legend('Original', 'Fitting result');
title(['Rational fitting with ', int2str(nPoles), ' poles']);
```

Compute and Plotting the Time-Domain Response

You use the `timeresp` method to compute the time-domain response of the transfer function that `RationalFunc` represents.

The following code illustrates how to create a random input signal, compute the time-domain response of `RationalFunc` to the input signal, and plot the results.

```
SampleTime=1e-11;
NumberOfSamples=4750;
OverSamplingFactor = 25;
InputTime = double((1:NumberOfSamples)')*SampleTime;
InputSignal = ...
     sign(randn(1, ceil(NumberOfSamples/OverSamplingFactor)));
InputSignal = repmat(InputSignal, [OverSamplingFactor, 1]);
InputSignal = InputSignal(:);

[tresp,t]=timeresp(RationalFunc,InputSignal,SampleTime);
plot(t*1e9,tresp);
title('Fitting Time-Domain Response', 'fonts', 12);
ylabel('Response to Random Input Signal');
xlabel('Time (ns)');
```

For more information about computing the time response of a model object, see the `timeresp` reference page.

Export Component Data to a File

In this section...

Available Export Formats

RF Toolbox software lets you export data from any `rfckt` object or from an `rfdata.data` object to industry-standard data files and MathWorks AMP files. This export capability lets you store data for use in other simulations.

Note The toolbox also lets you export data from an `rfmodel` object to a Verilog-A file. For information on how to do this, see "Export a Verilog-A Model" on page 4-5.

You can export data to the following file formats:

- Industry-standard file formats — Touchstone SNP, YNP, ZNP, HNP, and GNP formats specify the network parameters and noise information for measured and simulated data.

  For more information about Touchstone files, see `www.vhdl.org/pub/ibis/connector/touchstone_spec11.pdf`.

- MathWorks amplifier (AMP) file format — Specifies amplifier network parameters, output power versus input power, noise data and third-order intercept point.

  For more information about `.amp` files, see Appendix A, "AMP File Format".

How to Export Object Data

To export data from a circuit or data object, use a `write` command of the form

```
status = write(obj,'filename');
```

where

- `status` is a return value that indicates whether the write operation was successful.
- *`obj`* is the handle of the circuit or `rfdata.data` object.
- *`filename`* is the name of the file that contains the data.

For example,

```
status = write(rfckt.amplifier, 'myamp.amp');
```

exports data from an `rfckt.amplifier` object to the file `myamp.amp`.

Export Object Data

In this example, use the toolbox to create a vector of S-parameter data, store it in an `rfdata.data` object, and export it to a Touchstone file.

At the MATLAB prompt:

1. Type the following to create a vector, `s_vec`, of S-parameter values at three frequency values:

   ```
   s_vec(:,:,1) = ...
       [-0.724725-0.481324i, -0.685727+1.782660i; ...
         0.000000+0.000000i, -0.074122-0.321568i];
   s_vec(:,:,2) = ...
       [-0.731774-0.471453i, -0.655990+1.798041i; ...
         0.001399+0.000463i, -0.076091-0.319025i];
   s_vec(:,:,3) = ...
       [-0.738760-0.461585i, -0.626185+1.813092i; ...
         0.002733+0.000887i, -0.077999-0.316488i];
   ```

2. Type the following to create an `rfdata.data` object called `txdata` with the default property values:

   ```
   txdata = rfdata.data;
   ```

3. Type the following to set the S-parameter values of `txdata` to the values you specified in `s_vec`:

```
txdata.S_Parameters = s_vec;
```

4 Type the following to set the frequency values of `txdata` to `[1e9 2e9 3e9]`:

```
txdata.Freq=1e9*[1 2 3];
```

5 Type the following to export the data in `txdata` to a Touchstone file called `test.s2p`:

```
write(txdata,'test')
```

Basic Operations with RF Objects

In this section...

Read and Analyze RF Data from a Touchstone Data File

In this example, you create an `rfdata.data` object by reading the S-parameters of a 2-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 from 315 kHz to 6 GHz. The `read` command creates an `rfdata.data` object, `data`, and stores data from the file in the object's properties.

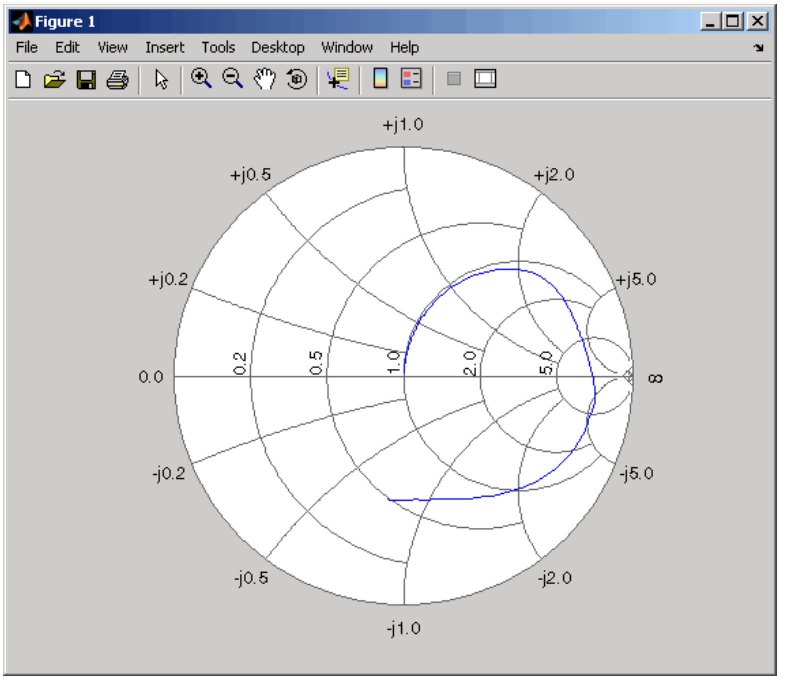
```
data = read(rfdata.data,'passive.s2p');
```

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 S_{11} parameters. Use the `smithchart` command to plot the 75-ohm S_{11} 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. Type the following set of commands at the MATLAB prompt 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:

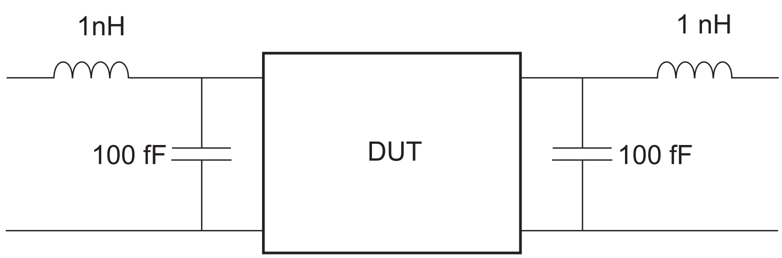
```
f =
  6.0000e+009

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.

De-Embed S-Parameters

The Touchstone data file `samplebjt2.s2p` contains S-parameter data collected from a bipolar transistor in a test fixture. The input of the fixture has a bond wire connected to a bond pad. The output of the fixture has a bond pad connected to a bond wire.

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.

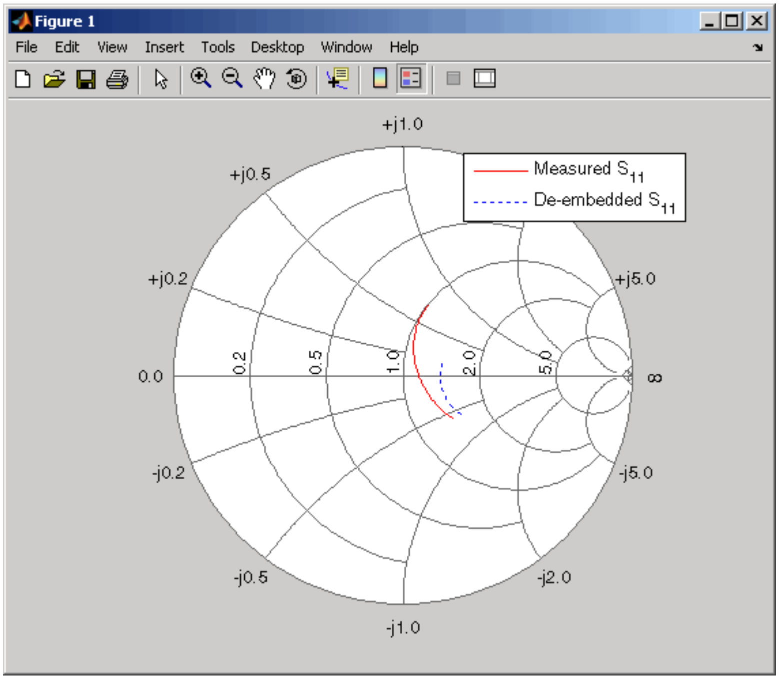
```
z0 = measured_data.Z0;

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);
```

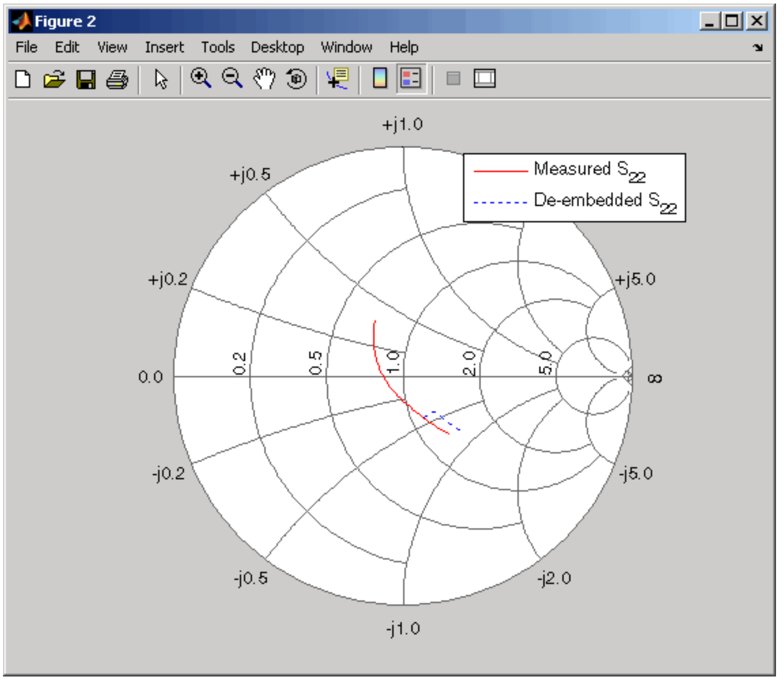
5 Plot the measured and de-embedded S_{11} parameters. Type the following set of commands at the MATLAB prompt to plot both the measured and the de-embedded S_{11} parameters on a Z Smith Chart:

```
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;
```



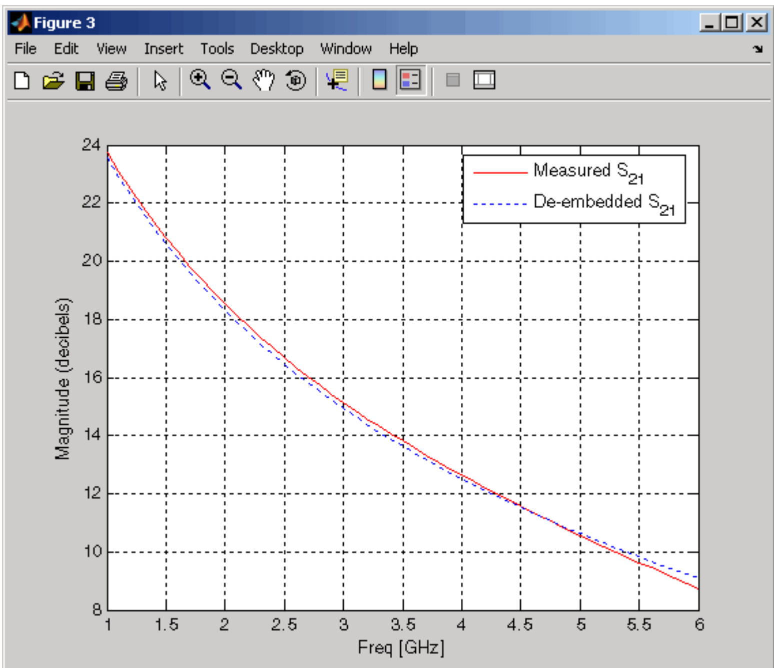
6 Plot the measured and de-embedded S_{22} parameters. Type the following set of commands at the MATLAB prompt to plot the measured and the de-embedded S_{22} parameters on a Z Smith Chart:

```
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 S_{21} parameters. Type the following set of commands at the MATLAB prompt to plot the measured and the de-embedded S_{21} 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;
```



4

Export Verilog-A Models

Model RF Objects Using Verilog-A

In this section...

Overview

Verilog-A is a language for modeling the high-level behavior of analog components and networks. Verilog-A describes components mathematically, for fast and accurate simulation.

RF Toolbox software lets you export a Verilog-A description of your circuit. You can create a Verilog-A model of any passive RF component or network and use it as a behavioral model for transient analysis in a third-party circuit simulator. This capability is useful in signal integrity engineering. For example, you can import the measured four-port S-parameters of a backplane into the toolbox, export a Verilog-A model of the backplane to a circuit simulator, and use the model to determine the performance of your driver and receiver circuitry when they are communicating across the backplane.

Behavioral Modeling Using Verilog-A

The Verilog-A language is a high-level language that uses modules to describe the structure and behavior of analog systems and their components. A *module* is a programming building block that forms an executable specification of the system.

Verilog-A uses modules to capture high-level analog behavior of components and systems. Modules describe circuit behavior in terms of

- Input and output nets characterized by predefined Verilog-A disciplines that describe the attributes of the nets.
- Equations and module parameters that define the relationship between the input and output nets mathematically.

When you create a Verilog-A model of your circuit, the toolbox writes a Verilog-A module that specifies circuit's input and output nets and the mathematical equations that describe how the circuit operates on the input to produce the output.

For more information on the Verilog-A language, see the Verilog-A Reference Manual.

Supported Verilog-A Models

RF Toolbox software lets you export a Verilog-A model of an `rfmodel` object. The toolbox provides one `rfmodel` object, `rfmodel.rational`, that you can use to represent any RF component or network for export to Verilog-A.

The `rfmodel.rational` object represents components as rational functions in pole-residue form, as described in the `rfmodel.rational` reference page. This representation can include complex poles and residues, which occur in complex-conjugate pairs.

The toolbox implements each `rfmodel.rational` object as a series of Laplace Transform S-domain filters in Verilog-A using the numerator-denominator form of the Laplace transform filter:

$$H(s) = \frac{\sum_{k=0}^{M} n_k s^k}{\sum_{k=0}^{N} d_k s^k}$$

where

- M is the order of the numerator polynomial.
- M is the order of the denominator polynomial.
- n_k is the coefficient of the kth power of s in the numerator.
- d_k is the coefficient of the kth power of s in the denominator.

The number of poles in the rational function is related to the number of Laplace transform filters in the Verilog-A module. However, there is not a one-to-one correspondence between the two. The difference arises because the toolbox combines each pair of complex-conjugate poles and the corresponding residues in the rational function to form a Laplace transform numerator and denominator with real coefficients. the toolbox converts the real poles of the rational function directly to a Laplace transform filter in numerator-denominator form.

Export a Verilog-A Model

In this section...

Represent a Circuit Object with a Model Object

Before you can write a Verilog-A model of an RF circuit object, you need to create an `rfmodel.rational` object to represent the component.

There are two ways to create an RF model object:

- You can fit a rational function model to the component data using the `rationalfit` function.
- You can use the `rfmodel.rational` constructor to specify the pole-residue representation of the component directly.

This section discusses using a rational function model. For more information on using the constructor, see the `rfmodel.rational` reference page.

When you use the `rationalfit` function to create an `rfmodel.rational` object that represents an RF component, the arguments you specify affect how quickly the resulting Verilog-A model runs in a circuit simulator.

You can use the `rationalfit` function with only the two required arguments. The syntax is:

```
model_obj = rationalfit(freq,data)
```

where

- `model_obj` is a handle to the rational function model object.
- *`freq`* is a vector of frequency values that correspond to the data values.
- *`data`* is a vector that contains the data to fit.

For faster simulation, create a model object with the smallest number of poles required to accurately represent the component. To control the number of poles, use the syntax:

```
model_obj = rationalfit(freq,data,tol,weight,delayfactor)
```

where

- *tol* — the relative error-fitting tolerance, in decibels. Specify the largest acceptable tolerance for your application. Using tighter tolerance values may force the `rationalfit` function to add more poles to the model to achieve a better fit.
- *weight* — a vector that specifies the weighting of the fit at each frequency.
- *delayfactor* — a value that controls the amount of delay used to fit the data. Delay introduces a phase shift in the frequency domain that may require a large number of poles to fit using a rational function model. When you specify the delay factor, the `rationalfit` function represents the delay as an exponential phase shift. This phase shift allows the function to fit the data using fewer poles.

These arguments are described in detail in the `rationalfit` function reference page.

Note You can also specify the number of poles directly using the `npoles` argument. The model accuracy is not guaranteed with approach, so you should not specify `npoles` when accuracy is critical. For more information on the `npoles` argument, see the `rationalfit` reference page.

If you plan to integrate the Verilog-A module into a large design for simulation using detailed models, such as transistor-level circuit models, the simulation time consumed by a Verilog-A module may have a trivial impact on the overall simulation time. In this case, there is no reason to take the time to optimize the rational function model of the component.

For more information on the `rationalfit` function arguments, see the `rationalfit` reference page.

Write a Verilog-A Module

You use the `writeva` method to create a Verilog-A module that describes the RF model object. This method writes the module to a specified file. Use the syntax:

```
status = writeva(model_obj,'obj1',{'inp','inn'},{'outp','outn'})
```

to write a Verilog-A module for the model object *`model_obj`* to the file `obj1.va`. The module has differential input nets, *`inp`* and *`inn`*, and differential output nets, *`outp`* and *`outn`*. The method returns `status`, a logical value of `true` if the operation is successful and `false` otherwise.

The `writeva` reference page describes the method arguments in detail.

An example of exporting a Verilog-A module appears in the RF Toolbox example, Modeling a High-Speed Backplane (Part 5: Rational Function Model to a Verilog-A Module).

5

RF Tool: An RF Analysis GUI

RF Tool

In this section...

What Is RF Tool?

RF Tool is a GUI that provides a visual interface for creating and analyzing RF components and networks. You can use RF Tool as a convenient alternative to the command-line RF circuit design and analysis objects and methods that come with RF Toolbox software.

RF Tool provides the ability to

- Create and import circuits.
- Set circuit parameters.
- Analyze circuits.
- Display circuit S-parameters in tabular form and on X-Y plots, polar plots, and Smith Charts.
- Export circuit data to the MATLAB workspace and to data files.

Open RF Tool

To open RF Tool, type the following at the MATLAB prompt:

```
rftool
```

For a description of the RF Tool GUI, see "RF Tool Window" on page 5-3. To learn how to create and import circuits, see "Create and Import Circuits with RF Tool" on page 5-6.

Note 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. For more information, see "Working with RF Tool Sessions" on page 5-28.

RF Tool Window

The RF Tool window consists of the following three panes:

- **RF Component List**

  Shows the components and networks in the session. The top-level node is the session.

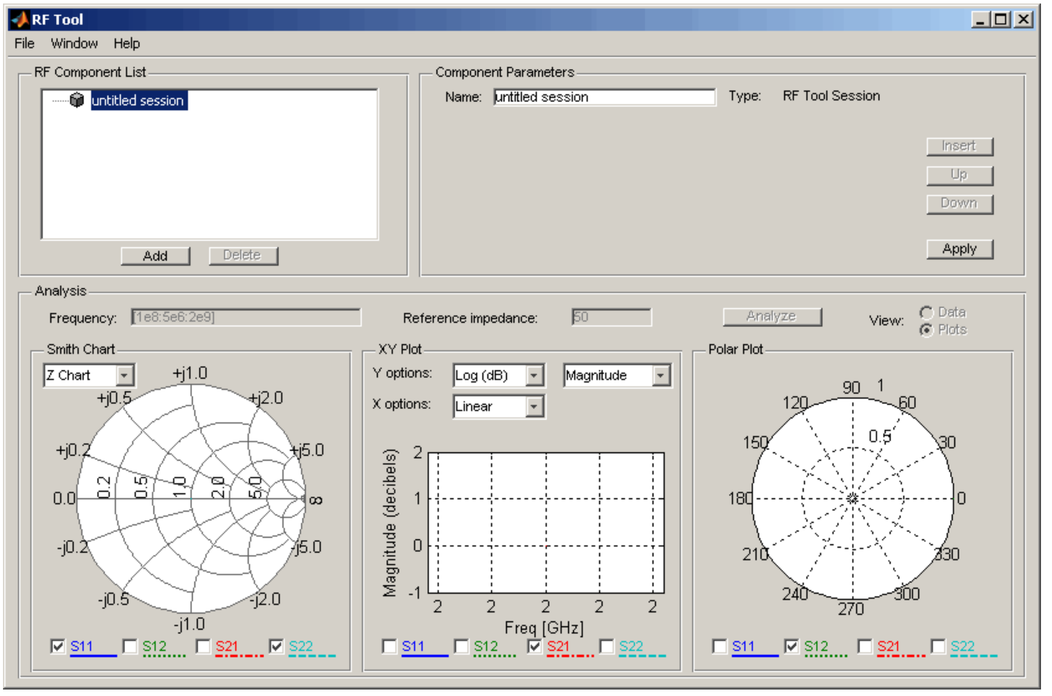
- **Component Parameters**

  Displays options and settings pertaining to the node you selected in the **RF Component List** pane.

- **Analysis**

  Displays options and settings pertaining to the circuit analysis and results display. After you analyze the circuit, this pane displays the analysis results and provides an interface for you to view the S-parameter data and modify the displayed plots.

The following figure shows the RF Tool window.



RF Tool Workflow

When you analyze a circuit using the RF Tool GUI, your workflow might include the following tasks:

1. Build the circuit by

   - Creating RF components and networks.
   - Importing components and networks from the MATLAB workspace or from a data file.

   See "Create and Import Circuits with RF Tool" on page 5-6.

2. Specify component data.

   See "Modify Component Data in RF Tool" on page 5-19.

3. Analyze the circuit.

   See "Analyze Circuits Using RF Tool" on page 5-20.

4. Export the circuit to the MATLAB workspace or to a file.

   See "Export RF Objects from RF Tool" on page 5-23.

Create and Import Circuits with RF Tool

In this section...

Circuits in RF Tool

In RF Tool, you can create circuits that include RF components and RF networks. Networks can contain both components and other networks.

Note In the circuit object command line interface, you create networks by building components and then connecting them together to form a network. In contrast, you build networks in RF Tool by creating a network and then populating it with components.

Create RF Components

This section contains the following topics:

- "Available RF Components" on page 5-7
- "Add an RF Component to a Session" on page 5-8

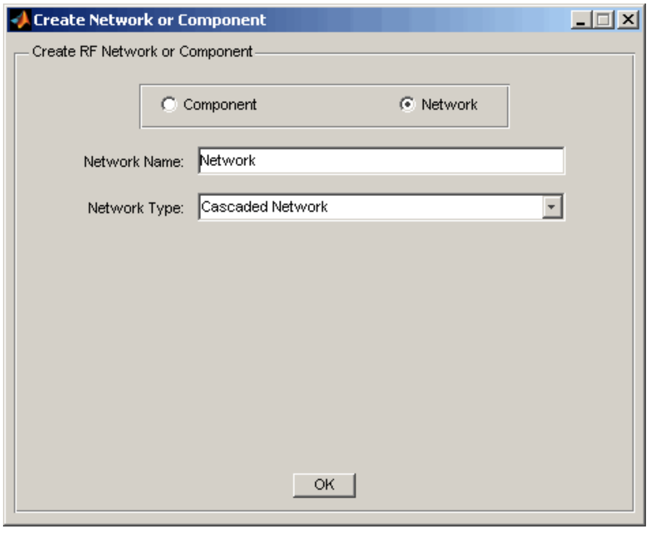
Available RF Components

The following table lists the RF components you can create using RF Tool and the corresponding RF Toolbox object.

RF Component	Corresponding RF Object
Data File	`rfckt.datafile`
Delay Line	`rfckt.delay`
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`
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.lchighpasspi`
LC Highpass Tee	`rfckt.lchighpasstee`
LC Lowpass Pi	`rfckt.lclowpasspi`
LC Lowpass Tee	`rfckt.lclowpasstee`

Add an RF Component to a Session

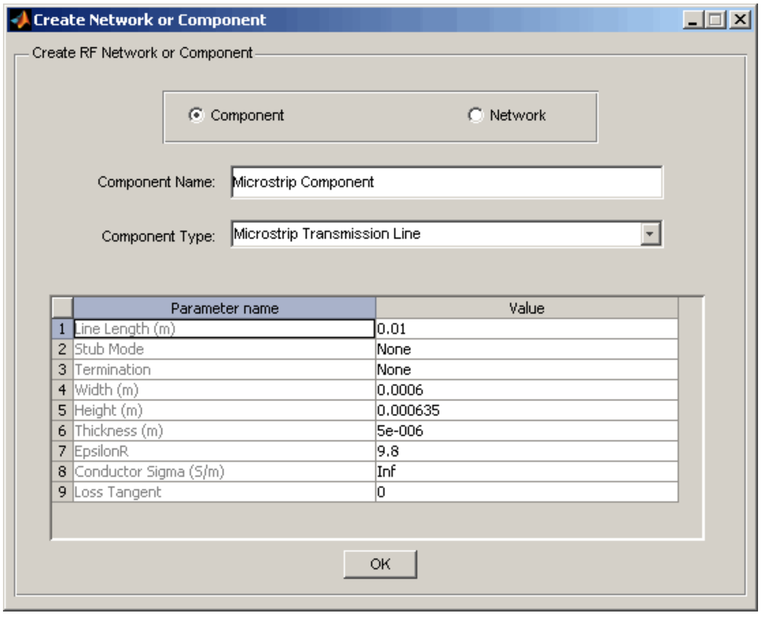
1 In the **RF Component List** pane, click **Add** to open the Create Network or Component dialog box.



2 In the Create Network or Component dialog box, select **Component**.

3 In the **Component Name** field, enter a name for the component. This name is used to identify the component in the **RF Component List** pane. For example, Microstrip Component.

4 From the **Component Type** menu, select the type of RF component you want to create. For example, `Microstrip Transmission Line`.



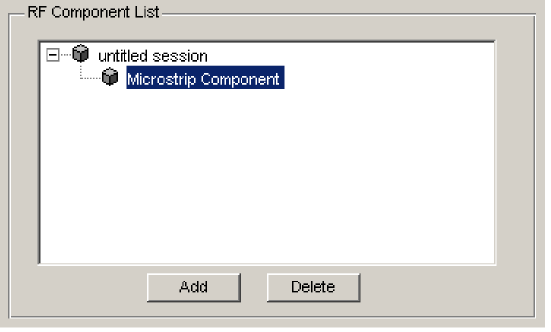
5 Adjust the parameter values as necessary.

Note You can accept the default values for some or all of the parameters and then change them later. For information on modifying the parameter values of an existing component, see "Modify Component Data in RF Tool" on page 5-19.

---

6 Click **OK**.

RF Tool adds the component to your session.



Create RF Networks with RF Tool

You create an RF network in RF Tool by adding a network to the session and then adding components to the network.

This section contains the following topics:

- "Available RF Networks" on page 5-11
- "Add an RF Network to a Session" on page 5-11
- "Populate an RF Network" on page 5-13
- "Reorder Circuits Within a Network" on page 5-14

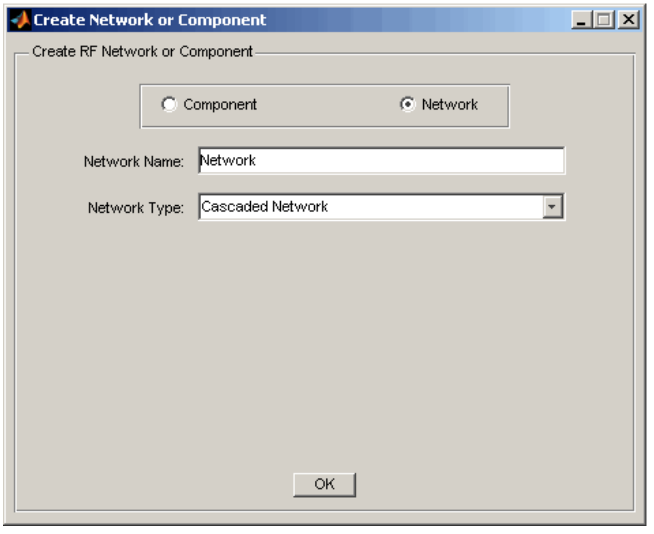
Available RF Networks

The following table lists the RF networks you can create using RF Tool.

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

Add an RF Network to a Session

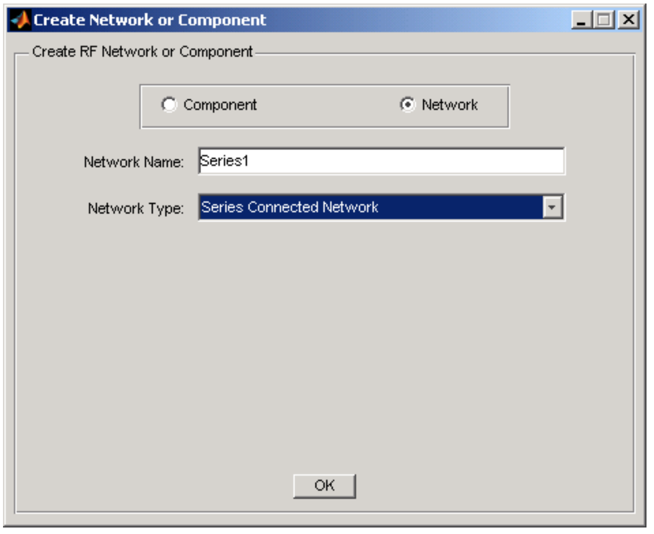
1 In the **RF Component List** pane, click **Add** to open the Create Network or Component dialog box.



2 In the Create Network or Component dialog box, select the **Network** option button.

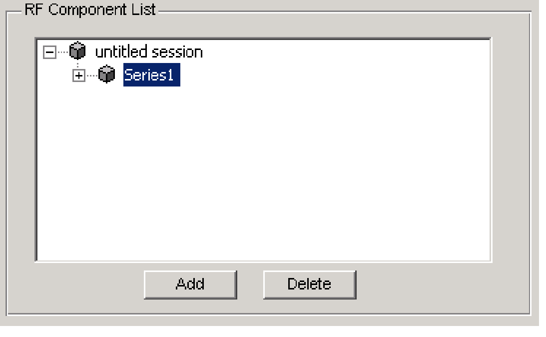
3 In the **Network Name** field, enter a name for the component. This name is used to identify the network in the **RF Component List** pane. For example, Series1.

4 From the **Network Type** menu, select the type of RF network you want to create. For example, `Series Connected Network`.



5 Click **OK**.

The RF Component List pane shows the new network.

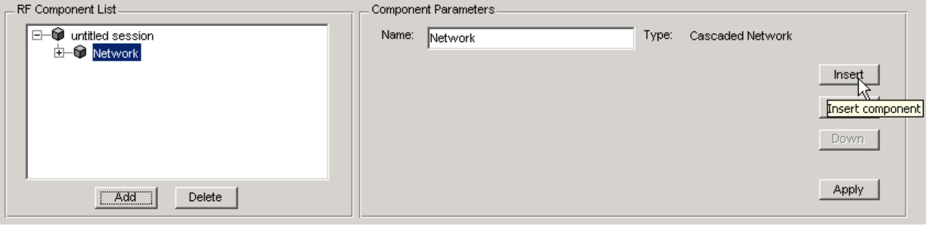


Populate an RF Network

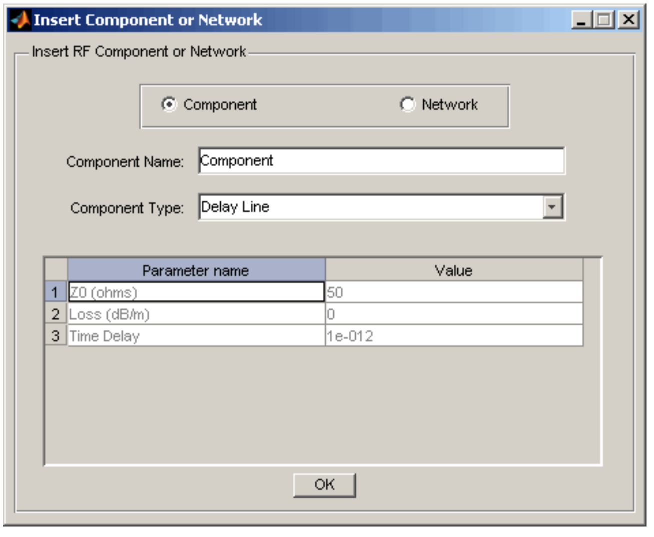
After you create a network using RF Tool, you must populate it with RF components and networks. You insert a component or network into a network in much the same way you add one to a session.

To populate an RF network:

1. In the **RF Component List** pane, select the network component you want to modify. Then, in the **Component Parameters** pane, click **Insert**.



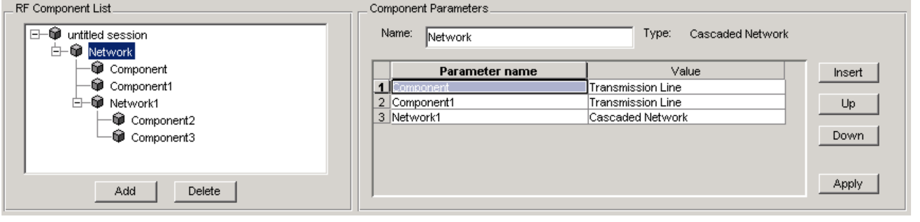
The Insert Component or Network dialog box appears.



2 Click **Component** or **Network** in the Insert Component or Network dialog box to add either a component or a network.

   Enter the component or network name, and select the appropriate type. If you are inserting a component, modify the parameter values as necessary. See “Add an RF Component to a Session” on page 5-8 or “Add an RF Network to a Session” on page 5-11 for details.

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



Reorder Circuits Within a Network

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

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

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

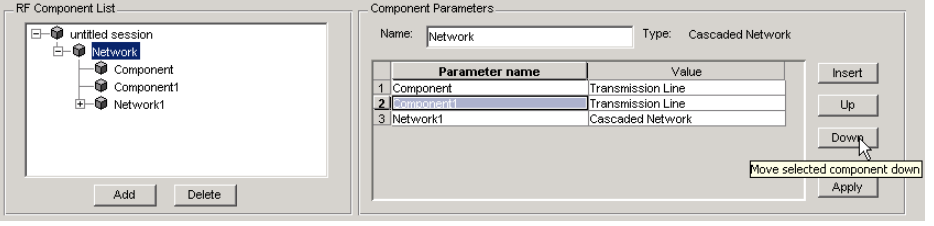
3 Click **Up** or **Down** until the circuit is where you want it.

To reverse the positions of `Component1` and `Network1` in the network shown in the following figure:

1 Select `Network` in the **RF Component List** pane.

2 Select `Component1` in the **Component Parameters** pane.

3 Click **Down** in the **Component Parameters** pane.



Import RF Objects into RF Tool

RF Tool lets you import RF objects from your workspace and from files to the top level of your session. You can import the following types of objects:

- Complex component and network objects that you created in your workspace using RF Toolbox objects.
- Components and networks you exported into your workspace from another RF Tool session.

  For information on exporting components and networks from an RF Tool session, see “Export RF Objects from RF Tool” on page 5-23.

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

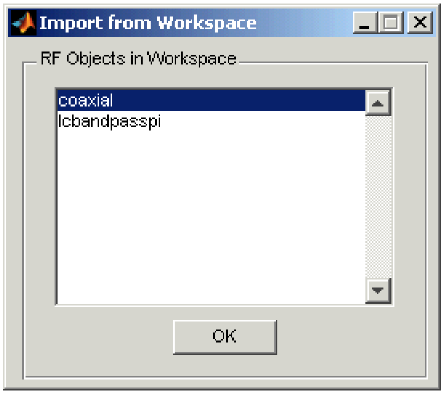
This section contains the following topics:

- “Import from the Workspace” on page 5-16
- “Import from a File into a Session” on page 5-16
- “Import from a File into a Network” on page 5-18

Import from the Workspace

To import RF circuit objects from the MATLAB workspace into your RF Tool session:

1. Select **Import From Workspace** from the **File** menu. The Import from Workspace dialog box appears. This dialog box 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.

Import from a File into a Session

You can import RF components from the following types of files into the top level of your session:

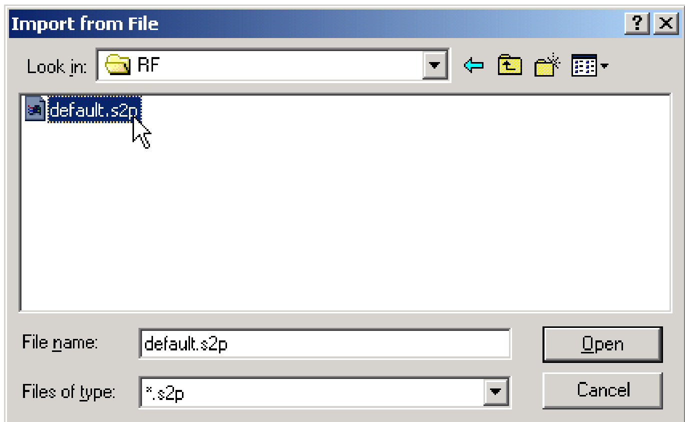
- S2P
- Y2P
- Z2P
- H2P

To import a component from one of these files:

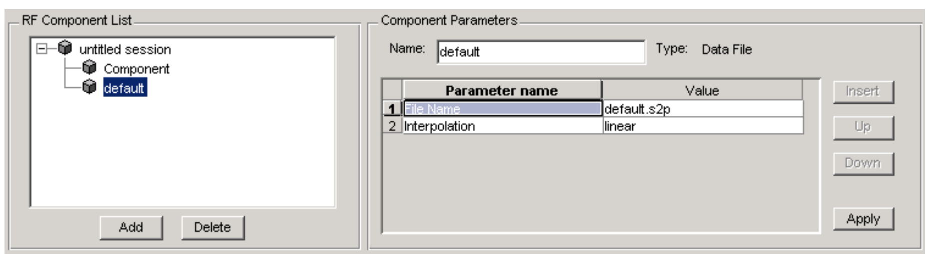
1 Select **Import From File** from the **File** menu. A file browser appears.

2 Select the file type you want to import.

3 Select the name of the file to import from the list of files in the browser.



4 Click **Open** to add the object to your session as a component.



The name of the component is the file name 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 file name, including the extension, 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.

Import from a File into a Network

You can import RF components from the following types of files into a network:

- S2P
- Y2P
- Z2P
- H2P

To import an RF component from a file into a network:

1. Insert a Data File component into the network.

   For more information on how add a component to a network, see "Populate an RF Network" on page 5-13.

2. Specify the name of the file from which to import the component in one of two ways:

   - Select the file name in the file name and type in the Import from File dialog box, and click **Open**.
   - Click **Cancel** to get out of the Import from File dialog box, and enter the file name in the **Value** field across from the **File Name** parameter in the Insert Component or Network dialog box.

"Model an RF Network Using RF Tool" on page 5-31 shows this process.

Modify Component Data in RF Tool

You can change the values of component parameters that you create and import. The component parameters in RF Tool correspond to the component properties that you specify in the command line.

To modify these values:

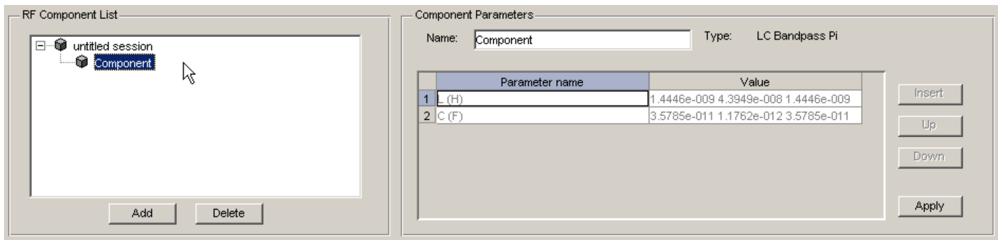
1. Select the component in the **RF Component List** pane.
2. In the **Component Parameters** pane, select the value you want to change, and enter the new value.

   Valid values for component parameters are listed on the corresponding RF Toolbox reference page. Use the links in "Available RF Components" on page 5-7 and "Available RF Networks" on page 5-11 to access these pages.
3. Click **Apply**.

Analyze Circuits Using RF Tool

After you add your circuits, you can analyze them with RF Tool:

1 Select the component or network you want to analyze in the **RF Component List** pane of RF Tool. For example, select the LC Bandpass Pi component, as shown in the following figure.



2 In the **Analysis** pane:

- Enter `[1e8:5e6:2e9]`, the analysis frequency range and step size in hertz, in the **Frequency** field.

  This value specifies an analysis from 0.1 GHz to 2 GHz in 5 MHz steps.

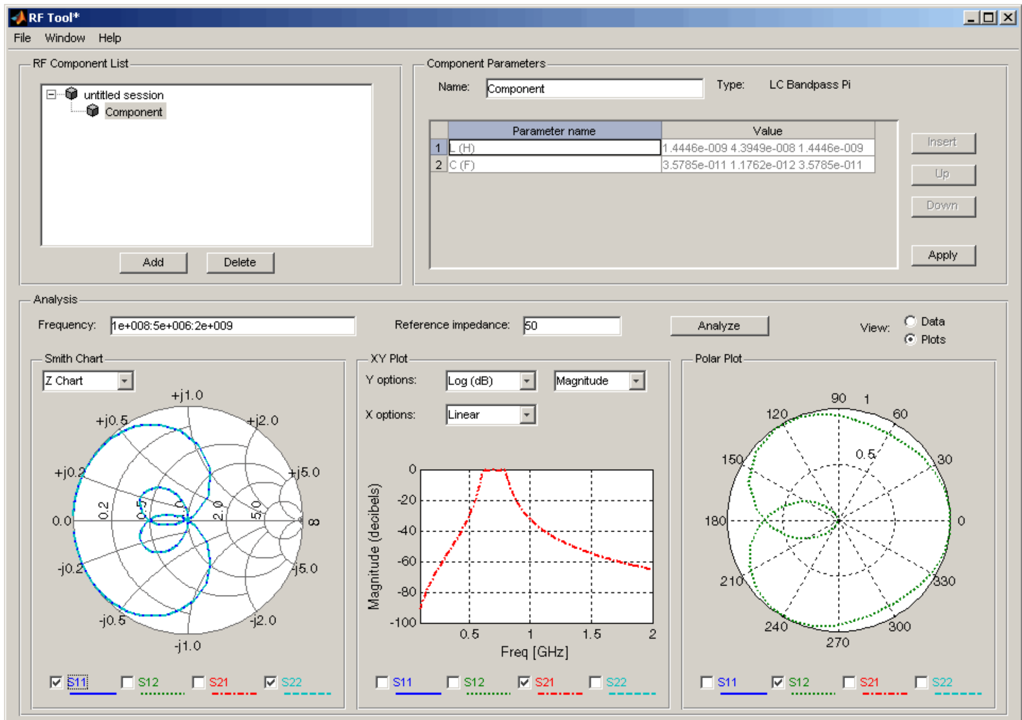
- Enter `50`, the reference impedance in ohms, in the **Reference impedance** field.

Note Alternately, you can specify the **Frequency** and **Reference impedance** values as MATLAB workspace variables or as valid MATLAB expressions.

---

3 Click **Analyze**.

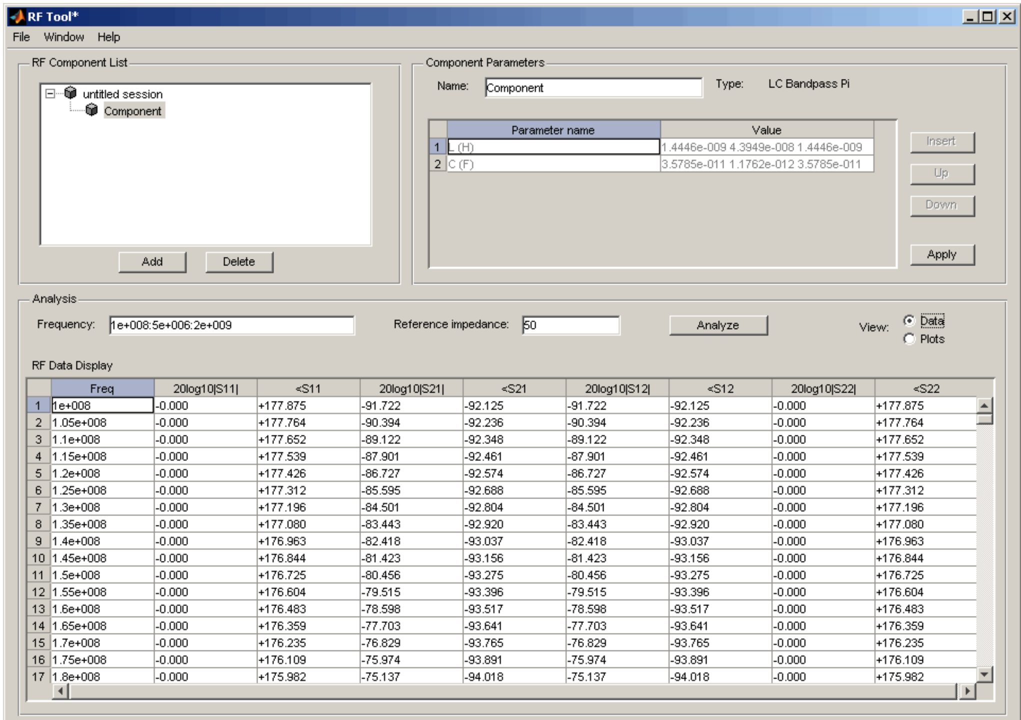
The **Analysis** pane displays a Smith Chart, an XY plot, and a polar plot of the analyzed circuit.



4 Select or deselect the S-parameter check boxes at the bottom of each plot to customize the parameters that the plot displays. Use the pull-downs at the top of each plot to customize the plot options.

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

5 Click **Data** in the upper-right corner of the **Analysis** pane to view the data in tabular form. The following figure shows the analysis data for the LC Bandpass Pi component at the frequencies and reference impedance shown in step 2.



Note The magnitude, in decibels, of S_{11} is listed in the 20log10[S11] column and the phase, in degrees, of S_{11} is listed in the <S11 column.

Export RF Objects from RF Tool

In this section...

Export Components and Networks

You can export RF components and networks that you create and refine in RF Tool to your MATLAB workspace or to files. You export circuits for the following reasons:

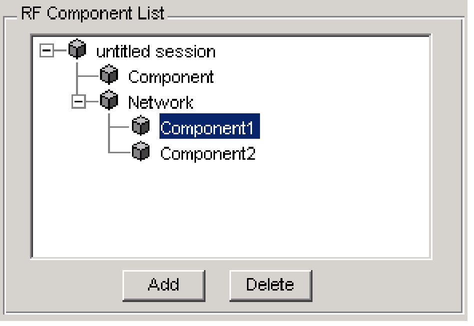
- To perform additional analysis using RF Toolbox functions that are not available in RF Tool.
- To incorporate them into larger RF systems.
- To import them into another session.

Export to the Workspace

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

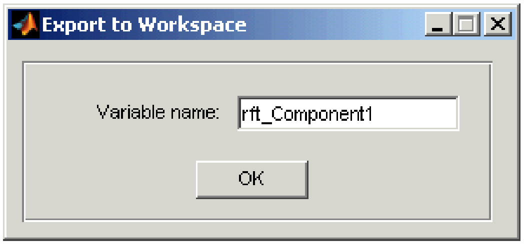
To export a component or network to the workspace:

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

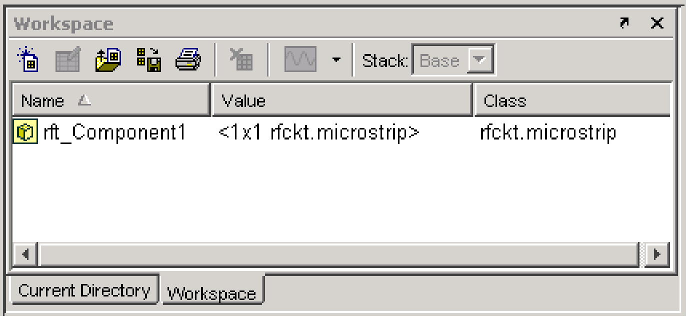


2 Select **Export to Workspace** from the **File** menu.

3 Enter a name for the exported object's handle in the **Variable name** field and click **OK**. The default name is the name of the component or network prefaced with the string `'rft_'`.



The component or network becomes accessible in the workspace via the specified object handle.



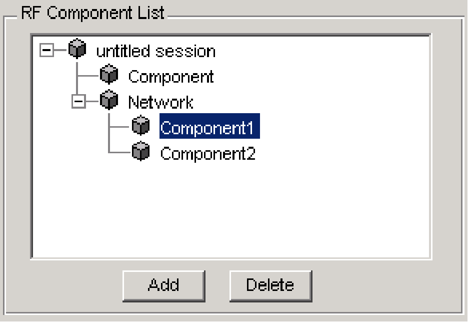
Export to a File

RF Tool lets you export components and networks to files in S2P format.

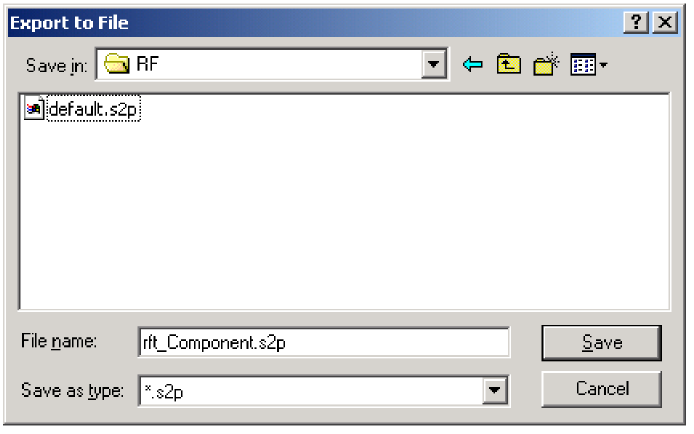
Note You must analyze a component or network in RF Tool before you can export it to a file. See “Analyze Circuits Using RF Tool” on page 5-20 for more information.

To export a component or network to a file:

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



2 Select **Export To File** from the **File** menu to open the file browser.



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

The default file name 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 (_).

Manage Circuits and Sessions in RF Tool

In this section...

Working with Circuits

In addition to building and specifying circuits, the RF Tool GUI allows you to perform the following tasks:

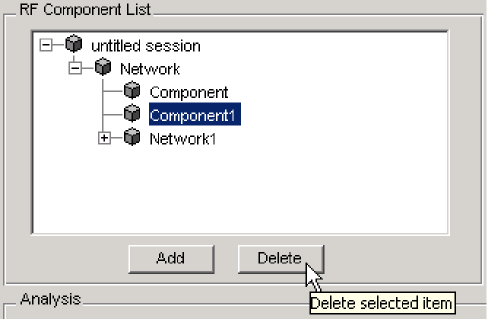
- "Delete a Circuit" on page 5-27
- "Rename a Circuit" on page 5-28

Delete a Circuit

To delete a circuit from your session:

1. Select the circuit in the **RF Component List** pane.
2. Click **Delete**.

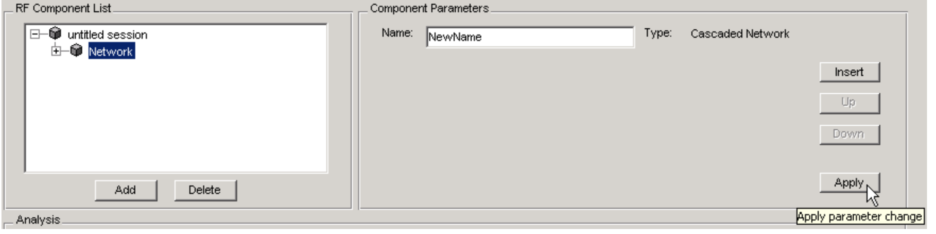
Note If the circuit you delete is a network, RF Tool deletes the network everything in the network.



Rename a Circuit

To rename a component or a network:

1. Select the component or network in the **RF Component List** pane.
2. Type the new name in the **Name** field of the **Component Parameters** pane.
3. Click **Apply**.



Working with RF Tool Sessions

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

This section contains the following topics:

- "Name or Rename a Session" on page 5-28
- "Save a Session" on page 5-29
- "Open a Session" on page 5-30
- "Start a New Session" on page 5-30

Name or Rename a Session

To name or rename an RF Tool session:

1. Select the session, or top-level node, in the **RF Component List** pane. (The session is selected by default when you open the RF Tool GUI.)

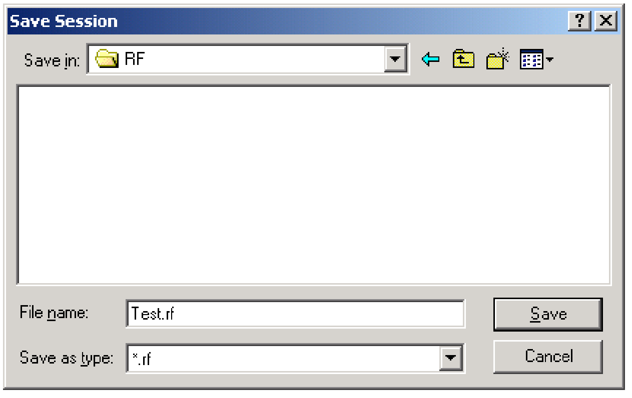
2 Type the desired name in the **Name** field of the **Component Parameters** pane.

3 Click **Apply**.

Save 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.

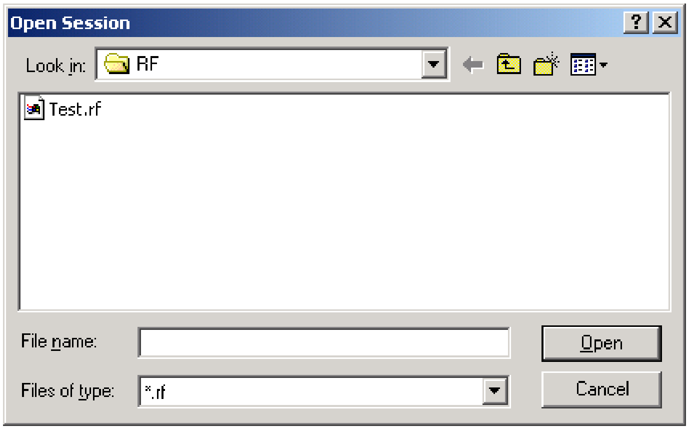


For example, to save your session as `Test.rf` in your current working directory, you would type `Test` in the **File name** field as shown above. RF Tool adds the `.rf` extension automatically to all RF Tool sessions you save.

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

Open a Session

You can load an existing session into 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, RF Tool prompts you to save your current session.

Start a New Session

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

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

Model an RF Network Using RF Tool

In this section...

Overview

In this example, you model the gain and noise figure of a cascaded network and then analyze the network using RF Tool.

The network used in this example consists of an amplifier and two transmission lines. Here, you learn how to create and analyze the network using RF Tool.

Start RF Tool

Type the following command at the MATLAB prompt to open the RF Tool window:

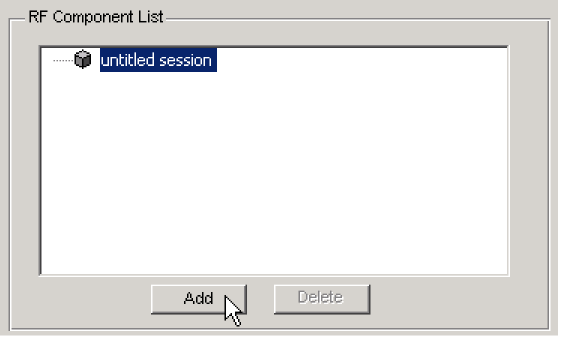
```
rftool
```

For more information about this GUI, see "RF Tool Window" on page 5-3.

Create the Amplifier Network

In this part of the example, you create a network to connect the amplifier components in cascade.

1 In the **RF Component List** pane, click **Add**.



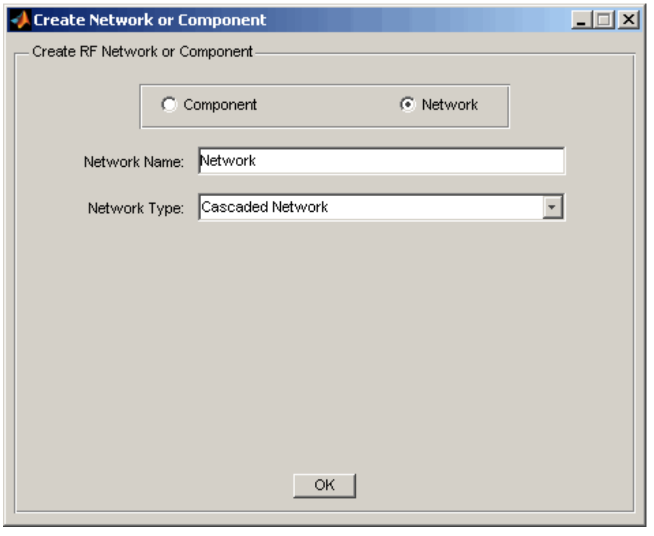
The Create Network or Component dialog box opens.

2 In the Create Network or Component dialog box:

- Select the **Network** option button.
- In the **Network Name** field, enter Amplifier Network.

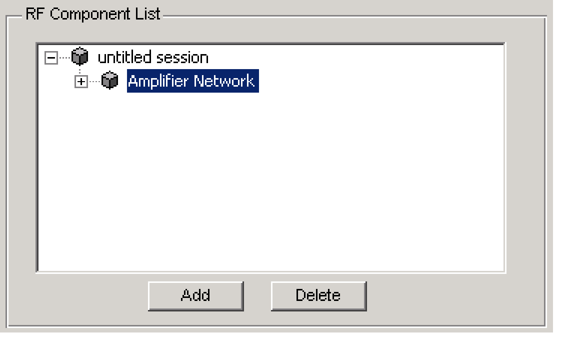
  This name is used to identify the network in the **RF Component List** pane.
- In the **Network Type** list, select `Cascaded Network`.

  A `Cascaded Network` means that when you add components to the network, RF Tool connects them in cascade.



3 Click **OK** to add the cascaded network to the session.

The network now appears in the **RF Component List** pane.



Populate the Amplifier Network

This part of the example shows how to add the following components to the network:

- "Transmission Line 1" on page 5-34
- "Amplifier" on page 5-35
- "Transmission Line 2" on page 5-37

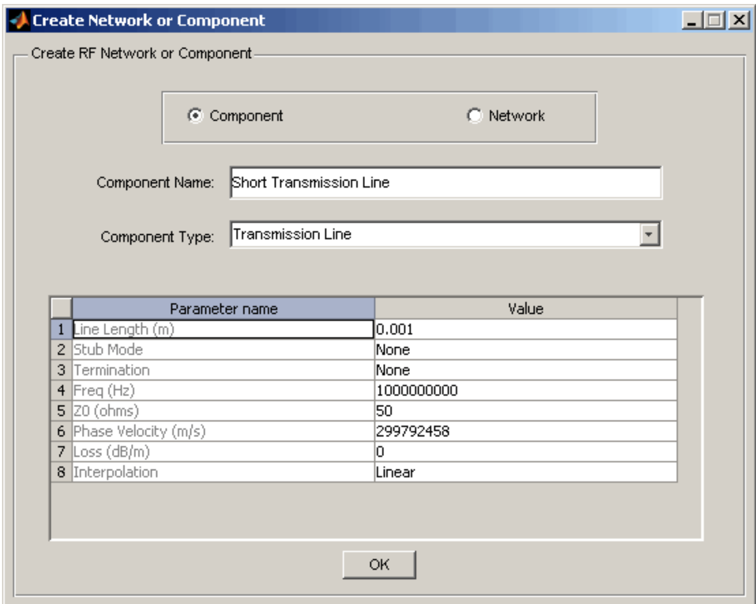
Transmission Line 1

1 In the **Component Parameters** pane, click **Insert** to open the Insert Component or Network dialog box.

2 In the Insert Component or Network dialog box:

- Select the **Component** option button.
- In the **Component Name** field, enter Short Transmission Line.

  This name is used to identify the component in the **RF Component List** pane.
- In the **Component Type** pull-down list, select `Transmission Line`.
- In the **Value** field across from the **Line Length (m)** parameter, enter `0.001`.



3 Click **OK** to add the transmission line to the network.

Amplifier

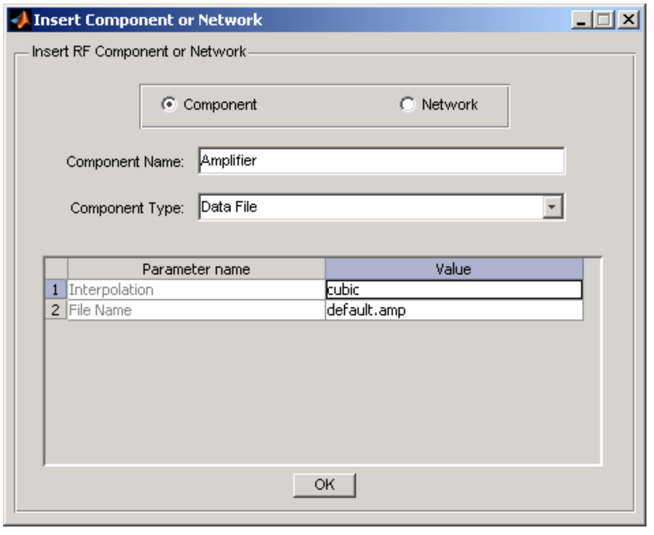
1 In the **Component Parameters** pane, click **Insert** to open the Insert Component or Network dialog box.

2 In the Insert Component or Network dialog box:

- Select the **Component** option button.
- In the **Component Name** field, enter Amplifier.

  This name is used to identify the component in the **RF Component List** pane.
- In the **Component Type** list, select `Data File`.
- In the Import from File dialog box that appears, click **Cancel** . You will specify the name of the file from which to import data in a later step.
- In the **Value** field across from the **Interpolation** parameter, enter `cubic`.

  This value tells RF Tool to use cubic interpolation to determine the behavior of the amplifier at frequency values that are not specified explicitly in the data file.
- In the **Value** field across from the **File Name** parameter, enter `default.amp`.



3 Click **OK** to add the amplifier to the network.

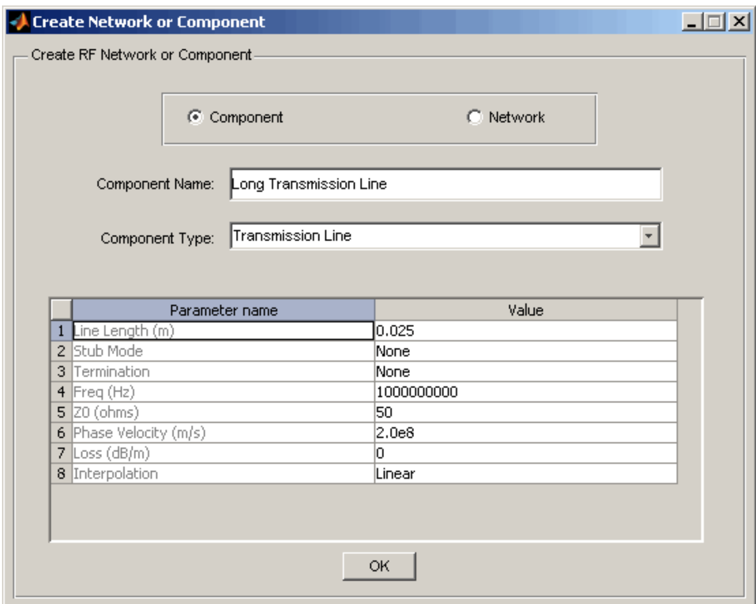
Transmission Line 2

1 In the **Component Parameters** pane, click **Insert** to open the Insert Component or Network dialog box.

2 In the Insert Component or Network dialog box, perform the following actions:

- Select the **Component** option button.
- In the **Component Name** field, enter Long Transmission Line.

  This name is used to identify the component in the **RF Component List** pane.
- In the **Component Type** list, select `Transmission Line`.
- In the **Value** field across from the **Line Length (m)** parameter, enter `0.025`.
- In the **Value** field across from the **Phase Velocity (m/s)** parameter, enter `2.0e8`.



3 Click **OK** to add the transmission line to the network.

Analyze the Amplifier Network

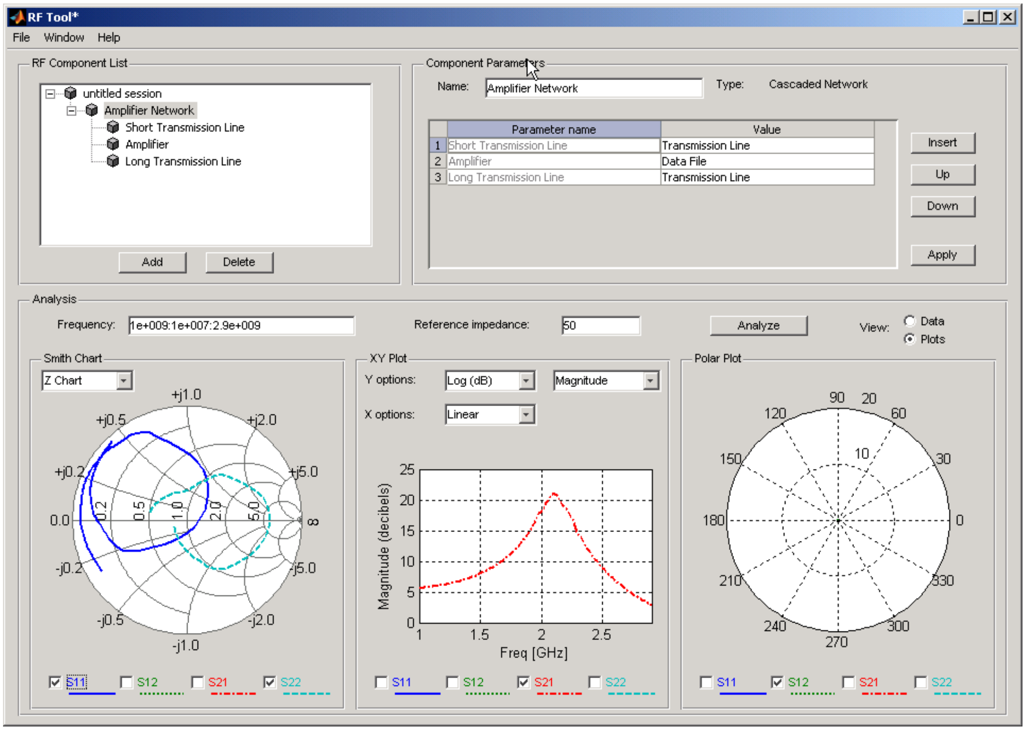
In this part of the example, you specify the range of frequencies over which to analyze the amplifier network and then run the analysis.

1 In the **Analysis** pane, change the **Frequency** entry to `[1.0e9:1e7:2.9e9]`.

   This value specifies an analysis from 1 GHz to 2.9 GHz by 10 MHz.

In the **Analysis** pane, click **Analyze** to simulate the network at the specified frequencies.

RF Tool displays a Smith Chart, an XY plot, and a polar plot of the analyzed circuit.



You can modify the plots by

- Selecting and deselecting the S-parameter check boxes at the bottom of each plot to customize the parameters that the plot displays.
- Using the pull-downs at the top of each plot to customize the plot options.

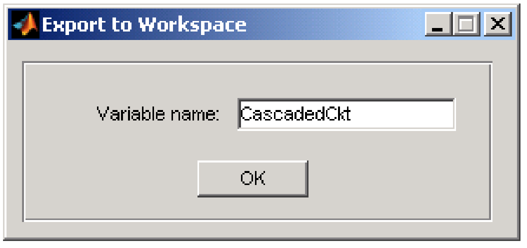
Export the Network to the Workspace

RF Tool lets you export components and networks to the workspace as circuit objects so you can use the RF Toolbox functions to perform additional analysis. This part of the example shows how to export the amplifier network to the workspace.

1 In the RF Tool window, select **File > Export to Workspace**.

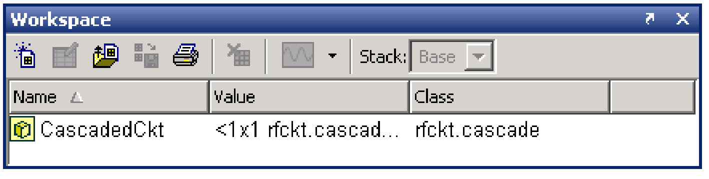
2 In the **Variable name** field, enter `CascadedCkt`.

   This name is the exported object's handle.



3 Click **OK**.

   RF Tool exports the amplifier network to an `rfckt.cascade` object, with the specified object handle, in the MATLAB workspace.



6

Objects — Alphabetical List

OpenIF

Purpose

Find open intermediate frequencies (IFs) in multiband transmitter or receiver architecture

Description

Use the `OpenIF` class to analyze the spurs and spur-free zones in a multiband transmitter or receiver. This information helps you determine intermediate frequencies (IFs) that do not produce interference in operating bands.

Construction

`hif = OpenIF` creates an intermediate-frequency (IF) planning object with properties set to their default values.

`hif = OpenIF(Name,Value)` creates an intermediate-frequency (IF) planning object with properties with additional options specified by one or more `Name,Value` pair arguments.

`hif = OpenIF(bandwidth)` creates an intermediate-frequency (IF) planning object with a specified IF bandwidth.

`hif = OpenIF(bandwidth,Name,Value)` creates an IF-planning object with a specified IF bandwidth and additional options specified by one or more `Name,Value` pair arguments.

Input Arguments

bandwidth

Specify the bandwidth of the IF signal. The bandwidth is a real positive scalar. The value you provide sets the `IFBW` property of your object. You can also set this property using an optional name-value pair argument.

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments, where `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside single quotes (`' '`). You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

IFLocation - IF location

`'MixerInput'` | `'MixerOutput'` (default)

Specify an up-conversion or down-conversion setup during object construction. The value you provide sets the `IFLocation` property of your object.

IFBW - IF bandwidth
nonnegative number

Specify the IF bandwidth during object construction. The value you provide sets the `IFBW` property of your object. You can also set this property using the optional `bandwidth` input argument.

SpurFloor - Spur floor
nonnegative number

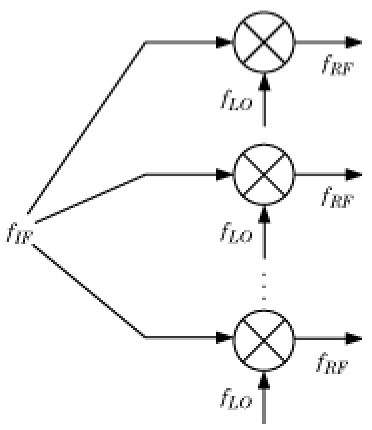
Specify the spur floor during object construction. The value you provide sets the `SpurFloor` property of your object.

Properties

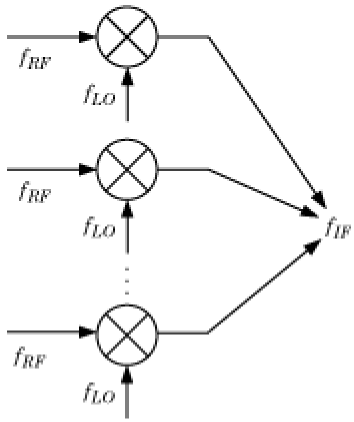
IFLocation

Specify an up-conversion or down-conversion setup.

- Setting `IFLocation` to `'MixerInput'` specifies an up-converting (transmitting) configuration, where one IF is mixed up to multiple RFs. The following figure shows this convention.



- Setting `IFLocation` to `'MixerOutput'` specifies a down-converting (receiving) configuration, where multiple RFs are mixed down to one IF. The following figure shows this convention.



The setting of `IFLocation` determines the available values for the `injection` argument of the `addMixer` method.

Default: `'MixerOutput'`

IFBW

Bandwidth of the IF signal in Hz. When you construct the object, the `bandwidth` argument specifies the value of this property.

Mixers

Vector of objects that holds mixer information. When you add mixers using the `addMixer` method, you also add an `OpenIFMixer` object to the `Mixers` vector of your original `OpenIF` object.

The following table lists the properties of each `OpenIFMixer` object.

Property	Description
`IMT`	Intermodulation table of the mixer.
`RFCF`	RF center frequency, in Hz.
`RFBW`	RF bandwidth, in Hz.
`MixingType`	Mixing (injection) type.
`SpurVector`	Vector of spur information.

The `IMT`, `RFCF`, `RFBW`, and `MixingType` properties are required inputs to the `addMixer` method.

NumMixers

Number of mixers. When you use the `addMixer` method, the number of mixers increases by one. The likelihood of finding an open IF decreases as you add mixers.

SpurFloor

Maximum difference in magnitude between a signal at 0 dBc and an intermodulation product that the `OpenIF` object considers a spur. The default value of this parameter is `99`, corresponding to a spur floor of –99 dBc.

Methods

addMixer	Add mixer to multiband transmitter or receiver for IF planning analysis
getSpurData	Find spurs in multiband transmitter or receiver frequency space
getSpurFreeZoneData	Find spur-free zones in multiband transmitter or receiver frequency space

report	Summarize mixer configurations and spur-free-zone information for a multiband transmitter or receiver
show	Produce a spur graph for a multiband transmitter or receiver

Copy Semantics

Handle. To learn how handle classes affect copy operations, see Copying Objects in the MATLAB documentation.

Examples

Spur-free zones of a multiband receiver

Set up an `OpenIF` object as a multiband receiver, add three mixers to it, and obtain information about its spur-free zones.

Define an `OpenIF` object. The first input is the bandwidth of the IF signal (50 MHz). The `'IFLocation','MixerOutput'` name-value pair specifies a downconverting configuration.

```
hif = OpenIF(50e6,'IFLocation','MixerOutput');
```

Define the first mixer with an intermodulation table and add it to the `OpenIF` object. Mixer 1 has an LO at 2.4 GHz, has a bandwidth of 100 MHz, and uses low-side injection.

```
IMT1 = [99 00 21 17 26; ...
        11 00 29 29 63; ...
        60 48 70 65 41; ...
        90 89 74 68 87; ...
        99 99 95 99 99];
addMixer(hif,IMT1,2.4e9,100e6,'low');
```

Mixer 2 has an LO at 3.7 GHz, has a bandwidth of 150 MHz, and uses low-side injection.

```
IMT2 = [99 00 09 12 15; ...
        20 00 26 31 48; ...
```

```
        55 70 51 70 53; ...
        85 90 60 70 94; ...
        96 95 94 93 92];
addMixer(hif,IMT2,3.7e9,150e6,'low');
```

Mixer 3 has an LO at 5 GHz, has a bandwidth of 200 MHz, and uses low-side injection.

```
IMT3 = [99 00 15 23 36; ...
        10 00 34 27 59; ...
        67 61 56 59 68; ...
        97 82 81 60 77; ...
        99 99 99 99 96];
addMixer(hif,IMT3,5e9,200e6,'low');
```

The multiband receiver is fully defined and ready for spur-free-zone analysis. Use the `report` method to analyze and display spur and spur-free zone information at the command line. The method also returns information about the mixers in the receiver.

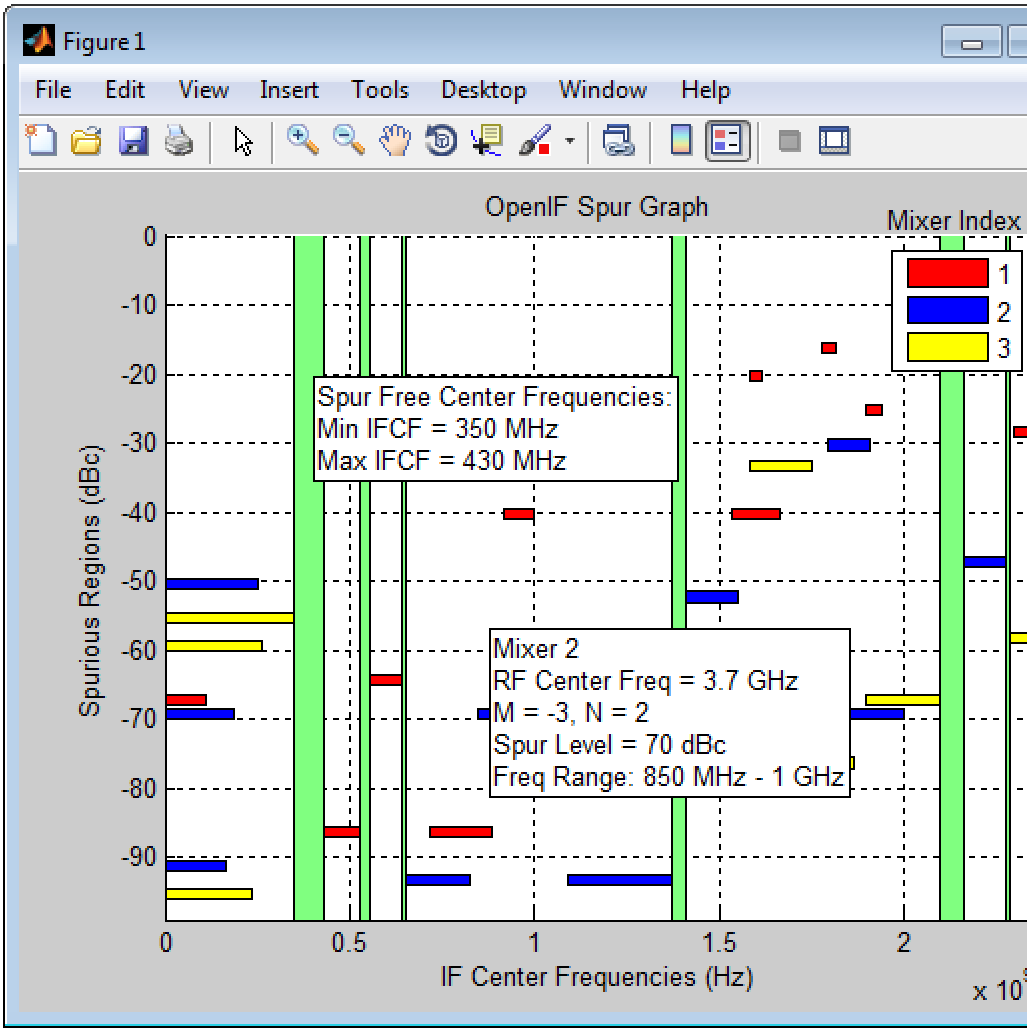
```
hif.report
```

```
     Intermediate Frequency (IF) Planner
     IF Location: MixerOutput

     -- MIXER 1 --
     RF Center Frequency: 2.4 GHz
     RF Bandwidth: 100 MHz
     IF Bandwidth: 50 MHz
     MixerType: low
     Intermodulation Table:   99   0  21  17  26
                              11   0  29  29  63
                              60  48  70  65  41
                              90  89  74  68  87
                              99  99  95  99  99
```

```
-- MIXER 2 --
RF Center Frequency: 3.7 GHz
RF Bandwidth: 150 MHz
IF Bandwidth: 50 MHz
MixerType: low
Intermodulation Table:   99   0   9  12  15
                         20   0  26  31  48
                         55  70  51  70  53
                         85  90  60  70  94
                         96  95  94  93  92

-- MIXER 3 --
RF Center Frequency: 5 GHz
RF Bandwidth: 200 MHz
IF Bandwidth: 50 MHz
MixerType: low
Intermodulation Table:   99   0  15  23  36
                         10   0  34  27  59
                         67  61  56  59  68
                         97  82  81  60  77
                         99  99  99  99  96

Spur-Free Zones:
 350.00 -  430.00 MHz
 530.00 -  556.25 MHz
 643.75 -  655.00 MHz
   1.38 -    1.41 GHz
   2.10 -    2.17 GHz
   2.28 -    2.29 GHz
```

Use the `show` method to analyze the receiver and produce an interactive spur graph. Generating a spur graph is a convenient way to summarize the results of the analysis graphically.

```
figure;
hif.show
```



References

Faria, Daniel, Lawrence Dunleavy, and Terje Svensen. “The Use of Intermodulation Tables for Mixer Simulations.” *Microwave Journal*. Vol. 45, No. 4, December 2002, p. 60.

rfckt.amplifier

Purpose

RF amplifier

Syntax

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

Description

Use the `amplifier` class to represent RF amplifiers that are characterized by network parameters, noise data, and nonlinearity data.

`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 that you do not specify retain their default values.

Use the `read` method to read the amplifier data from a data file in one of the following formats:

- Touchstone
- Agilent P2D
- Agilent S2D
- AMP

See Appendix A, "AMP File Format" for information about the `.amp` format.

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`IntpType`	Interpolation method
`Name`	Object name
`NetworkData`	Network parameter information

`NoiseData`	Noise information
`NonlinearData`	Nonlinearity information
`nPort`	Number of ports

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
extract	Extract array of network parameters from data object
extract	Extract array of network parameters from data object
getop	Display operating conditions
getop	Display operating conditions
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object

loglog	Plot specified circuit object parameters using log-log scale
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
read	Read RF data from file to new or existing circuit or data object
read	Read RF data from file to new or existing circuit or data object
restore	Restore data to original frequencies
restore	Restore data to original frequencies
semilogx	Plot specified circuit object parameters using log scale for *x*-axis

semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
amp = rfckt.amplifier('IntpType','cubic')

amp =

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

References

EIA/IBIS Open Forum, *Touchstone File Format Specification*, Rev. 1.1, 2002 (`http://www.vhdl.org/pub/ibis/connector/touchstone_spec11.pdf`).

See Also

rfckt.datafile | rfckt.mixer | rfckt.passive | rfdata.data | rfdata.ip3 | rfdata.network | rfdata.nf | rfdata.noise | rfdata.power

Purpose

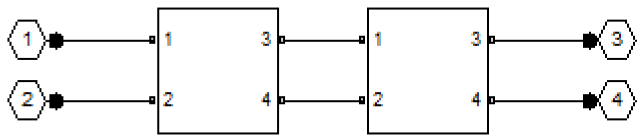
Cascaded network

Syntax

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

Description

Use the `cascade` class to represent cascaded networks of RF objects that are characterized by the components that make up the network. The following figure shows the configuration of a pair of cascaded networks.



`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. Properties you do not specify retain their default values.

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`Ckts`	Circuit objects in network
`Name`	Object name
`nPort`	Number of ports

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides

plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
amp = rfckt.amplifier('IntpType','cubic');
tx1 = rfckt.txline;
tx2 = rfckt.txline;
casc = rfckt.cascade('Ckts',{tx1,amp,tx2})
```

```
casc =

            Name: 'Cascaded Network'
            nPort: 2
            AnalyzedResult: []
            Ckts: {1x3 cell}
```

References

Ludwig, R. and P. Bretchko, *RF Circuit Design: Theory and Applications*, Prentice Hall, 2000.

See Also

`rfckt.hybrid` | `rfckt.hybridg` | `rfckt.parallel` | `rfckt.series`

Purpose

Coaxial transmission line

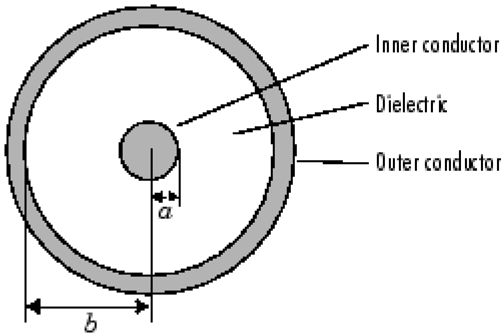
Syntax

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

Description

Use the `coaxial` class to represent coaxial transmission lines that are characterized by line dimensions, stub type, and termination.

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



`h = rfckt.coaxial` returns a coaxial transmission line object whose properties are set to their default values.

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

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`EpsilonR`	Relative permittivity of dielectric

`InnerRadius`	Inner conductor radius
`LineLength`	Transmission line length
`LossTangent`	Tangent of loss angle
`MuR`	Relative permeability of dielectric
`Name`	Object name
`nPort`	Number of ports
`OuterRadius`	Outer conductor radius
`SigmaCond`	Conductor conductivity
`StubMode`	Type of stub
`Termination`	Stub transmission line termination

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
getz0	Characteristic impedance of transmission line object
getz0	Characteristic impedance of transmission line object

listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for *x*-axis

semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
tx1=rfckt.coaxial('OuterRadius',0.0045)

tx1 =

           Name: 'Coaxial Transmission Line'
           nPort: 2
           AnalyzedResult: []
           LineLength: 0.0100
           StubMode: 'NotAStub'
           Termination: 'NotApplicable'
           OuterRadius: 0.0045
           InnerRadius: 7.2500e-004
           MuR: 1
           EpsilonR: 2.3000
           LossTangent: 0
```

```
SigmaCond: Inf
```

References

Pozar, David M. *Microwave Engineering*, John Wiley & Sons, Inc., 2005.

See Also

`rfckt.cpw` | `rfckt.microstrip` | `rfckt.parallelplate` | `rfckt.rlcgline` | `rfckt.twowire` | `rfckt.txline`

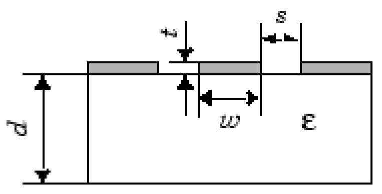
Purpose Coplanar waveguide transmission line

Syntax

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

Description Use the `cpw` class to represent coplanar waveguide transmission lines that are characterized by line dimensions, stub type, and termination.

A coplanar waveguide transmission line is shown in cross-section in the following figure. 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).



`h = rfckt.cpw` returns a coplanar waveguide transmission line object whose properties are set to their default values.

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

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`ConductorWidth`	Conductor width
`EpsilonR`	Relative permittivity of dielectric
`Height`	Dielectric thickness
`LineLength`	Transmission line length

`LossTangent`	Tangent of loss angle
`Name`	Object name
`nPort`	Number of ports
`SigmaCond`	Conductor conductivity
`SlotWidth`	Width of slot
`StubMode`	Type of stub
`Termination`	Stub transmission line termination
`Thickness`	Conductor thickness

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
getz0	Characteristic impedance of transmission line object
getz0	Characteristic impedance of transmission line object
listformat	List valid formats for specified circuit object parameter

listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides
plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for x-axis
semilogx	Plot specified circuit object parameters using log scale for x-axis

semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
tx1=rfckt.cpw('Thickness',0.0075e-6)

tx1 =

             Name: 'Coplanar Waveguide Transmission Line'
             nPort: 2
             AnalyzedResult: []
             LineLength: 0.0100
             StubMode: 'NotAStub'
             Termination: 'NotApplicable'
             ConductorWidth: 6.0000e-004
             SlotWidth: 2.0000e-004
             Height: 6.3500e-004
             Thickness: 7.5000e-009
             EpsilonR: 9.8000
             SigmaCond: Inf
             LossTangent: 0
```

References

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

See Also

`rfckt.coaxial` | `rfckt.microstrip` | `rfckt.parallelplate` | `rfckt.rlcgline` | `rfckt.twowire` | `rfckt.txline`

Purpose

Component or network from file data

Syntax

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

Description

Use the `datafile` class to represent RF components and networks that are characterized by measured or simulated data in a file.

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

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

Use the `read` method to read the data from a file in one of the following formats:

- Touchstone
- Agilent P2D
- Agilent S2D
- AMP

See Appendix A, "AMP File Format" for information about the `.amp` format.

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`File`	File containing circuit data
`IntpType`	Interpolation method

`Name`	Object name
`nPort`	Number of ports

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
extract	Extract array of network parameters from data object
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale

loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
read	Read RF data from file to new or existing circuit or data object
read	Read RF data from file to new or existing circuit or data object
restore	Restore data to original frequencies
restore	Restore data to original frequencies
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis

semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
data=rfckt.datafile('File','default.s2p')

data =

            Name: 'Data File'
            nPort: 2
            AnalyzedResult: [1x1 rfdata.data]
            IntpType: 'Linear'
            File: 'default.s2p'
```

References

EIA/IBIS Open Forum, *Touchstone File Format Specification*, Rev. 1.1, 2002 (`http://www.vhdl.org/pub/ibis/connector/touchstone_spec11.pdf`).

See Also

`rfckt.amplifier` | `rfckt.mixer` | `rfckt.passive` | http://www.vhdl.org/pub/ibis/connector/touchstone_spec11.pdf

Purpose

Delay line

Syntax

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

Description

Use the `delay` class to represent delay lines that are characterized by line loss and time delay.

`h = rfckt.delay` returns a delay line object whose properties are set to their default values.

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

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`Loss`	Delay line loss
`Name`	Object name
`nPort`	Number of ports
`TimeDelay`	Delay introduced by line
`Z0`	Characteristic impedance

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object

calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
getz0	Characteristic impedance of transmission line object
getz0	Characteristic impedance of transmission line object
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides

polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
del=rfckt.delay('TimeDelay',1e-11)

del =

          Name: 'Delay Line'
          nPort: 2
          AnalyzedResult: []
```

```
Z0: 50
Loss: 0
TimeDelay: 1.0000e-011
```

References

Ludwig, R. and P. Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

See Also

`rfckt.rlcgline` | `rfckt.txline`

Purpose

Hybrid connected network

Syntax

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

Description

Use the `hybrid` class to represent hybrid connected networks of linear RF objects that are characterized by the components that make up the network.

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

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

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`Ckts`	Circuit objects in network
`Name`	Object name
`nPort`	Number of ports

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object

circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates

semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
tx1 = rfckt.txline;
tx2 = rfckt.txline;
hyb = rfckt.hybrid('Ckts',{tx1,tx2})

hyb =

          Name: 'Hybrid Connected Network'
          nPort: 2
          AnalyzedResult: []
          Ckts: {1x2 cell}
```

References

Ludwig, R. and P. Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

See Also

`rfckt.cascade` | `rfckt.hybridg` | `rfckt.parallel` | `rfckt.series`

Purpose

Inverse hybrid connected network

Syntax

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

Description

Use the `hybridg` class to represent inverse hybrid connected networks of linear RF objects that are characterized by the components that make up the network.

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

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

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`Ckts`	Circuit objects in network
`Name`	Object name
`nPort`	Number of ports

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object

calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates

semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
tx1 = rfckt.txline;
tx2 = rfckt.txline;
invhyb = rfckt.hybridg('Ckts',{tx1,tx2})

invhyb =

          Name: 'Hybrid G Connected Network'
          nPort: 2
          AnalyzedResult: []
          Ckts: {1x2 cell}
```

References

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

See Also

`rfckt.cascade` | `rfckt.hybrid` | `rfckt.parallel` | `rfckt.series`

Purpose

Bandpass pi filter

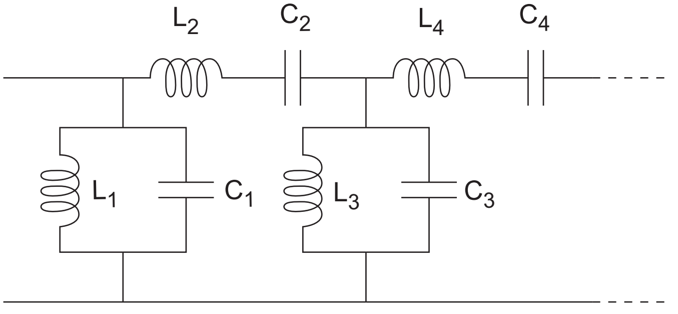
Syntax

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

Description

Use the `lcbandpasspi` class to represent a bandpass pi filter as a network of inductors and capacitors.

The LC bandpass pi network object is a 2-port network as shown in the following circuit diagram.



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

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

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

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`C`	Capacitance data
`L`	Inductance data
`Name`	Object name
`nPort`	Number of ports

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
extract	Extract array of network parameters from data object
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object

listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis

semilogy	Plot specified circuit object parameters using log scale for x-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
filter = rfckt.lcbandpasspi...
            ('C',[1e-12 4e-12],'L',[2e-9 2.5e-9])

filter =

              Name: 'LC Bandpass Pi'
              nPort: 2
              AnalyzedResult: []
              L: [2x1 double]
              C: [2x1 double]
```

References

Ludwig, R. and P. Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

Zverev, A.I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

`rfckt.lcbandpasstee` | `rfckt.lcbandstoppi` | `rfckt.lcbandstoptee` | `rfckt.lchighpasspi` | `rfckt.lchighpasstee` | `rfckt.lclowpasspi` | `rfckt.lclowpasstee`

Purpose

Bandpass tee filter

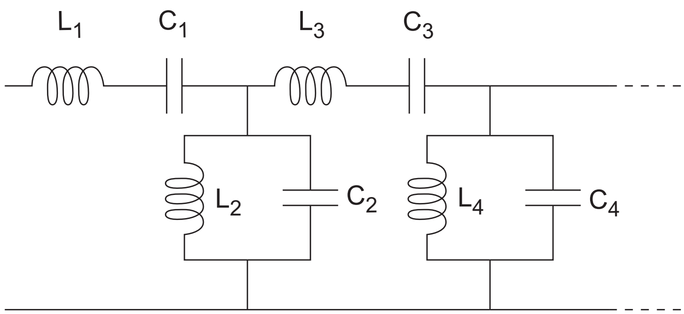
Syntax

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

Description

Use the `lcbandpasstee` class to represent a bandpass tee filter as a network of inductors and capacitors.

The LC bandpass tee network object is a 2-port network as shown in the following circuit diagram.



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

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

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

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`C`	Capacitance data
`L`	Inductance data
`Name`	Object name
`nPort`	Number of ports

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
extract	Extract array of network parameters from data object
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object

listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis

semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
filter = rfckt.lcbandpasstee...
            ('C',[1e-12 4e-12],'L',[2e-9 2.5e-9])

filter =

              Name: 'LC Bandpass Tee'
              nPort: 2
              AnalyzedResult: []
              L: [2x1 double]
              C: [2x1 double]
```

References

Ludwig, R. and P. Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

Zverev, A.I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

`rfckt.lcbandpasspi` | `rfckt.lcbandstoppi` | `rfckt.lcbandstoptee` | `rfckt.lchighpasspi` | `rfckt.lchighpasstee` | `rfckt.lclowpasspi` | `rfckt.lclowpasstee`

Purpose

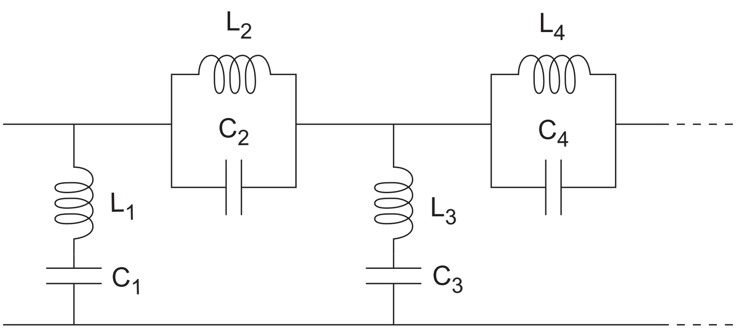
Bandstop pi filter

Syntax

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

Description

Use the `lcbandstoppi` class to represent a bandstop pi filter as a network of inductors and capacitors. The LC bandstop pi network object is a 2-port network as shown in the following circuit diagram.



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

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

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

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`C`	Capacitance data
`L`	Inductance data
`Name`	Object name
`nPort`	Number of ports

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
extract	Extract array of network parameters from data object
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object

listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis

semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
filter = rfckt.lcbandstoppi...
            ('C',[1e-12 4e-12],'L',[2e-9 2.5e-9])

filter =

              Name: 'LC Bandstop Pi'
              nPort: 2
              AnalyzedResult: []
              L: [2x1 double]
              C: [2x1 double]
```

References

Ludwig, R. and P. Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

Zverev, A.I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

`rfckt.lcbandpasspi` | `rfckt.lcbandpasstee` | `rfckt.lcbandstoptee` | `rfckt.lchighpasspi` | `rfckt.lchighpasstee` | `rfckt.lclowpasspi` | `rfckt.lclowpasstee`

Purpose

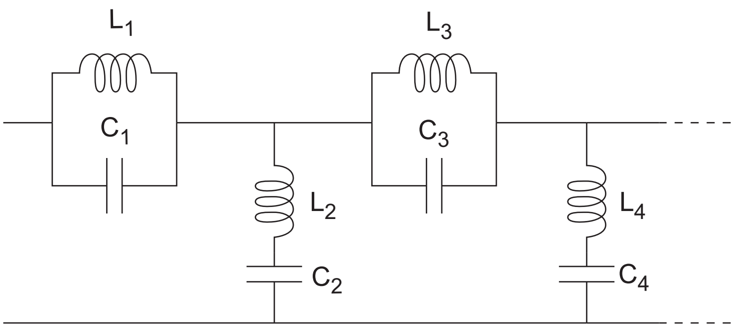
Bandstop tee filter

Syntax

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

Description

Use the `lcbandstoptee` class to represent a bandstop tee filter as a network of inductors and capacitor. The LC bandstop tee network object is a 2-port network as shown in the following circuit diagram.



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

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

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

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`C`	Capacitance data
`L`	Inductance data
`Name`	Object name
`nPort`	Number of ports

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
extract	Extract array of network parameters from data object
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object

listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis

semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
filter = rfckt.lcbandstoptee...
            ('C',[1e-12 4e-12],'L',[2e-9 2.5e-9])

filter =

              Name: 'LC Bandstop Tee'
              nPort: 2
              AnalyzedResult: []
              L: [2x1 double]
              C: [2x1 double]
```

References

Ludwig, R. and P. Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

Zverev, A.I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

`rfckt.lcbandpasspi` | `rfckt.lcbandpasstee` | `rfckt.lcbandstoppi` | `rfckt.lchighpasspi` | `rfckt.lchighpasstee` | `rfckt.lclowpasspi` | `rfckt.lclowpasstee`

Purpose

Highpass pi filter

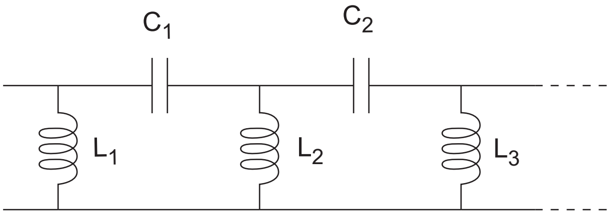
Syntax

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

Description

Use the `lchighpasspi` class to represent a highpass pi filter as a network of inductors and capacitors.

The LC highpass pi network object is a 2-port network as shown in the following circuit diagram.



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

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

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

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`C`	Capacitance data
`L`	Inductance data

`Name`	Object name
`nPort`	Number of ports

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
extract	Extract array of network parameters from data object
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale

loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart

smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
filter = rfckt.lchighpasspi...
            ('C',[1e-12 4e-12],'L',[2e-9 2.5e-9])

filter =

              Name: 'LC Highpass Pi'
              nPort: 2
              AnalyzedResult: []
              L: [2x1 double]
              C: [2x1 double]
```

References

Ludwig, R. and P. Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

Zverev, A.I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

`rfckt.lcbandpasspi` | `rfckt.lcbandpasstee` | `rfckt.lcbandstoppi` | `rfckt.lcbandstoptee` | `rfckt.lchighpasstee` | `rfckt.lclowpasspi` | `rfckt.lclowpasstee`

Purpose

Highpass tee filter

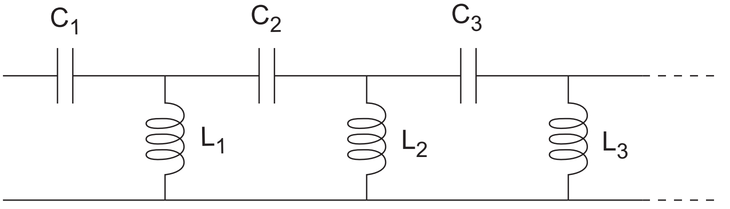
Syntax

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

Description

Use the `lchighpasstee` class to represent a highpass tee filter as a network of inductors and capacitors.

The LC highpass tee network object is a 2-port network as shown in the following circuit diagram.



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

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

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

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`C`	Capacitance data
`L`	Inductance data

`Name`	Object name
`nPort`	Number of ports

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
extract	Extract array of network parameters from data object
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale

loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart

smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
filter = rfckt.lchighpasstee...
               ('C',[1e-12 4e-12],'L',[2e-9 2.5e-9])

filter =

                 Name: 'LC Highpass Tee'
                 nPort: 2
                 AnalyzedResult: []
                 L: [2x1 double]
                 C: [2x1 double]
```

References

Ludwig, R. and P. Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

Zverev, A.I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

`rfckt.lcbandpasspi` | `rfckt.lcbandpasstee` | `rfckt.lcbandstoppi` | `rfckt.lcbandstoptee` | `rfckt.lchighpasspi` | `rfckt.lclowpasspi` | `rfckt.lclowpasstee`

Purpose

Lowpass pi filter

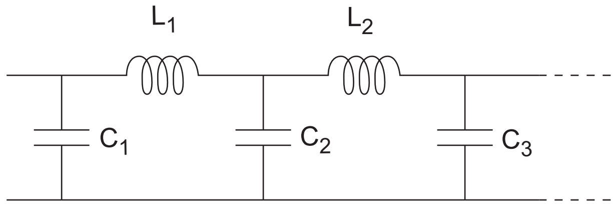
Syntax

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

Description

Use the `lclowpasspi` class to represent a lowpass pi filter as a network of inductors and capacitors.

The LC lowpass pi network object is a 2-port network as shown in the following circuit diagram.



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

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

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

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`C`	Capacitance data
`L`	Inductance data

`Name`	Object name
`nPort`	Number of ports

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
extract	Extract array of network parameters from data object
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale

loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart

smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
filter = rfckt.lclowpasspi...
           ('C',[1e-12 4e-12],'L',[2e-9 2.5e-9])

filter =

             Name: 'LC Lowpass Pi'
             nPort: 2
             AnalyzedResult: []
             L: [2x1 double]
             C: [2x1 double]
```

References

Ludwig, R. and P. Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

Zverev, A.I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

`rfckt.lcbandpasspi` | `rfckt.lcbandpasstee` | `rfckt.lcbandstoppi` | `rfckt.lcbandstoptee` | `rfckt.lchighpasspi` | `rfckt.lchighpasstee` | `rfckt.lclowpasstee`

Purpose

Lowpass tee filter

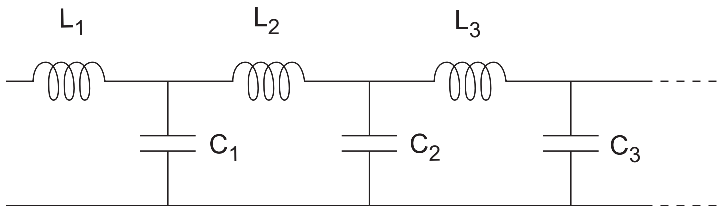
Syntax

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

Description

Use the `lclowpasstee` class to represent a lowpass tee filter as a network of inductors and capacitors.

The LC lowpass tee network object is a 2-port network as shown in the following circuit diagram.



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

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

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

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`C`	Capacitance data
`L`	Inductance data

`Name`	Object name
`nPort`	Number of ports

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
extract	Extract array of network parameters from data object
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale

loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart

smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
filter = rfckt.lclowpasstee...
            ('C',[1e-12 4e-12],'L',[2e-9 2.5e-9])

filter =

              Name: 'LC Lowpass Tee
              nPort: 2
              AnalyzedResult: []
              L: [2x1 double]
              C: [2x1 double]
```

References

Ludwig, R. and P. Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

Zverev, A.I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

`rfckt.lcbandpasspi` | `rfckt.lcbandpasstee` | `rfckt.lcbandstoppi` | `rfckt.lcbandstoptee` | `rfckt.lchighpasspi` | `rfckt.lchighpasstee` | `rfckt.lclowpasspi`

rfckt.microstrip

Purpose

Microstrip transmission line

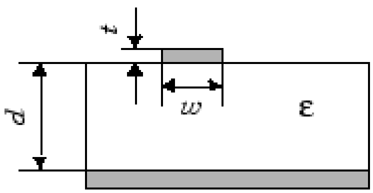
Syntax

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

Description

Use the `microstrip` class to represent microstrip transmission lines that are characterized by line dimensions and optional stub properties.

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



`h = rfckt.microstrip` returns a microstrip transmission line object whose properties are set to their default values.

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

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`EpsilonR`	Relative permittivity of dielectric
`Height`	Dielectric thickness
`LineLength`	Microstrip line length

`LossTangent`	Tangent of loss angle
`Name`	Object name
`nPort`	Number of ports
`SigmaCond`	Conductor conductivity
`StubMode`	Type of stub
`Termination`	Stub transmission line termination
`Thickness`	Microstrip thickness
`Width`	Parallel-plate width

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
getz0	Characteristic impedance of transmission line object
getz0	Characteristic impedance of transmission line object
listformat	List valid formats for specified circuit object parameter

listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides
plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for x-axis
semilogx	Plot specified circuit object parameters using log scale for x-axis

semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
tx1=rfckt.microstrip('Thickness',0.0075e-6)

tx1 =

            Name: 'Microstrip Transmission Line'
            nPort: 2
            AnalyzedResult: []
            LineLength: 0.0100
            StubMode: 'NotAStub'
            Termination: 'NotApplicable'
            Width: 6.0000e-004
            Height: 6.3500e-004
            Thickness: 7.5000e-009
            EpsilonR: 9.8000
            SigmaCond: Inf
            LossTangent: 0
```

References

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

See Also

`rfckt.coaxial` | `rfckt.cpw` | `rfckt.parallelplate` | `rfckt.rlcgline` | `rfckt.twowire` | `rfckt.txline`

Purpose

2-port representation of RF mixer and its local oscillator

Syntax

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

Description

Use the `mixer` class to represent RF mixers and their local oscillators that are characterized by network parameters, noise data, nonlinearity data, and local oscillator frequency.

`h = rfckt.mixer` returns a mixer object whose properties all have 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 that you do not specify retain their default values.

Use the `read` method to read the mixer data from a data file in one of the following formats:

- Touchstone
- Agilent P2D
- Agilent S2D
- AMP

See Appendix A, "AMP File Format" for information about the `.amp` format.

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`FLO`	Local oscillator frequency
`FreqOffset`	Frequency offset data
`IntpType`	Interpolation method

MixerSpurData	Data from mixer spur table
MixerType	Type of mixer
Name	Object name
NetworkData	Network parameter information
NoiseData	Noise information
NonlinearData	Nonlinearity information
nPort	Number of ports
PhaseNoiseLevel	Phase noise data

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
extract	Extract array of network parameters from data object
extract	Extract array of network parameters from data object
getop	Display operating conditions
getop	Display operating conditions
listformat	List valid formats for specified circuit object parameter

listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
read	Read RF data from file to new or existing circuit or data object
read	Read RF data from file to new or existing circuit or data object
restore	Restore data to original frequencies

restore	Restore data to original frequencies
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
mix1 = rfckt.mixer('IntpType','cubic')

mix1 =

            Name: 'Mixer'
            nPort: 2
            AnalyzedResult: [1x1 rfdata.data]
            IntpType: 'Cubic'
            NetworkData: [1x1 rfdata.network]
```

```
NoiseData: [1x1 rfdata.noise]
NonlinearData: Inf
MixerSpurData: []
MixerType: 'Downconverter'
FLO: 1.0000e+009
FreqOffset: []
PhaseNoiseLevel: []
```

References

EIA/IBIS Open Forum, *Touchstone File Format Specification*, Rev. 1.1, 2002 (`http://www.vhdl.org/pub/ibis/connector/touchstone_spec11.pdf`).

See Also

`rfckt.amplifier` | `rfckt.datafile` | `rfckt.passive` | `rfdata.data` | `rfdata.ip3` | `rfdata.mixerspur` | `rfdata.network` | `rfdata.nf` | `rfdata.noise` | `rfdata.power` | http://www.vhdl.org/pub/ibis/connector/touchstone_spec11.pdf

Purpose

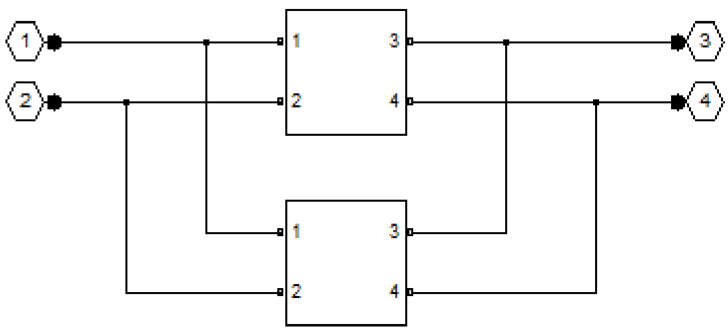
Parallel connected network

Syntax

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

Description

Use the `parallel` class to represent networks of linear RF objects connected in parallel that are characterized by the components that make up the network. The following figure shows a pair of networks in a parallel configuration.



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

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

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`Ckts`	Circuit objects in network
`Name`	Object name
`nPort`	Number of ports

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale

loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart

smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
tx1 = rfckt.txline;
tx2 = rfckt.txline;
plel = rfckt.parallel('Ckts',{tx1,tx2})

plel =

             Name: 'Parallel Connected Network'
             nPort: 2
             AnalyzedResult: []
             Ckts: {1x2 cell}
```

References

Ludwig, R. and P. Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

See Also

`rfckt.cascade` | `rfckt.hybrid` | `rfckt.hybridg` | `rfckt.series`

Purpose

Parallel-plate transmission line

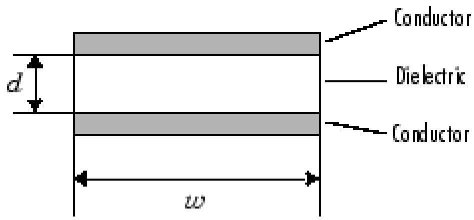
Syntax

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

Description

Use the `parallelplate` class to represent parallel-plate transmission lines that are characterized by line dimensions and optional stub properties.

A parallel-plate transmission line is shown in cross-section in the following figure. Its physical characteristics include the plate width w and the plate separation d.



`h = rfckt.parallelplate` returns a parallel-plate transmission line object whose properties are set to their default values.

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

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP, and group delay values
`EpsilonR`	Relative permittivity of dielectric
`LineLength`	Parallel-plate line length

LossTangent	Tangent of loss angle
MuR	Relative permeability of dielectric
Name	Object name
nPort	Number of ports
Separation	Distance between plates
SigmaCond	Conductor conductivity
StubMode	Type of stub
Termination	Stub transmission line termination
Width	Transmission line width

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
getz0	Characteristic impedance of transmission line object
getz0	Characteristic impedance of transmission line object
listformat	List valid formats for specified circuit object parameter

listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis

semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
tx1=rfckt.parallelplate('LineLength',0.045)

tx1 =

           Name: 'Parallel-Plate Transmission Line'
           nPort: 2
           AnalyzedResult: []
           LineLength: 0.0450
           StubMode: 'NotAStub'
           Termination: 'NotApplicable'
           Width: 0.0050
           Separation: 1.0000e-003
           MuR: 1
           EpsilonR: 2.3000
           LossTangent: 0
           SigmaCond: Inf
```

References

Pozar, David M. *Microwave Engineering*, John Wiley & Sons, Inc., 2005.

See Also

`rfckt.coaxial` | `rfckt.cpw` | `rfckt.microstrip` | `rfckt.rlcgline` | `rfckt.twowire` | `rfckt.txline`

rfckt.passive

Purpose Passive component or network

Syntax

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

Description Use the `passive` class to represent passive RF components and networks that are characterized by passive network parameter data.

`h = rfckt.passive` returns an passive-device object whose properties all have their default values.

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

Use the `read` method to read the passive object data from a Touchstone data file. When you read S-parameter data into an `rfckt.passive` object, the magnitude of your S_{21} data must be less than or equal to 1.

Due to random numerical error, data measured from a passive device is not necessarily passive. However, `rfckt.passive` objects can only contain passive data. To import data with active regions, use the `rfckt.amplifier` object, even if the original data represents a passive device.

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`IntpType`	Interpolation method
`Name`	Object name
`NetworkData`	Network parameter information
`nPort`	Number of ports

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
extract	Extract array of network parameters from data object
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane

plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
read	Read RF data from file to new or existing circuit or data object
read	Read RF data from file to new or existing circuit or data object
restore	Restore data to original frequencies
restore	Restore data to original frequencies
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis

semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
pas = rfckt.passive('IntpType','cubic')

pas =

           Name: 'Passive'
           nPort: 2
           AnalyzedResult: [1x1 rfdata.data]
           IntpType: 'Cubic'
           NetworkData: [1x1 rfdata.network]
```

References

EIA/IBIS Open Forum, *Touchstone File Format Specification*, Rev. 1.1, 2002 (`http://www.vhdl.org/pub/ibis/connector/touchstone_spec11.pdf`).

See Also

`rfckt.amplifier` | `rfckt.datafile` | `rfckt.mixer` | `rfdata.data` | `rfdata.network` | http://www.vhdl.org/pub/ibis/connector/touchstone_spec11.pdf

rfckt.rlcgline

Purpose

RLCG transmission line

Syntax

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

Description

Use the `rlcgline` class to represent RLCG transmission lines that are characterized by line loss, line length, stub type, and termination.

`h = rfckt.rlcgline` returns an RLCG transmission line object whose properties are set to their default values.

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

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`C`	Capacitance data
`Freq`	Frequency data
`G`	Conductance data
`IntpType`	Interpolation method
`L`	Inductance data
`LineLength`	Transmission line length
`Name`	Object name
`nPort`	Number of ports
`R`	Resistance data
`StubMode`	Type of stub
`Termination`	Stub transmission line termination

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
getz0	Characteristic impedance of transmission line object
getz0	Characteristic impedance of transmission line object
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane

plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart

write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
tx1=rfckt.rlcgline('R',0.002,'C',8.8542e-12,...
                   'L',1.2566e-6,'G',0.002')

tx1 =

           Name: 'RLCG Transmission Line'
           nPort: 2
           AnalyzedResult: []
           LineLength: 0.0100
           StubMode: 'NotAStub'
           Termination: 'NotApplicable'
           Freq: 1.0000e+009
           R: 0.0020
           L: 1.2566e-006
           C: 8.8542e-012
           G: 0.0020
           IntpType: 'Linear'
```

References

Ludwig, R. and P. Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

See Also

`rfckt.coaxial` | `rfckt.cpw` | `rfckt.microstrip` | `rfckt.parallelplate` | `rfckt.twowire` | `rfckt.txline`

Purpose

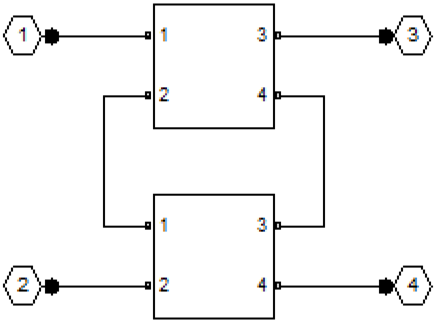
Series connected network

Syntax

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

Description

Use the `series` class to represent networks of linear RF objects connected in series that are characterized by the components that make up the network. The following figure shows a pair of networks in a series configuration.



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

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

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`Ckts`	Circuit objects in network

`Name`	Object name
`nPort`	Number of ports

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane

plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart

write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
tx1 = rfckt.txline;
tx2 = rfckt.txline;
ser = rfckt.series('Ckts',{tx1,tx2})

ser =

             Name: 'Series Connected Network'
             nPort: 2
             AnalyzedResult: []
             Ckts: {1x2 cell}
```

References

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

See Also

`rfckt.cascade` | `rfckt.hybrid` | `rfckt.hybridg` | `rfckt.parallel`

Purpose

Series RLC component

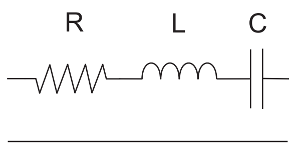
Syntax

```
h = rfckt.seriesrlc
h = rfckt.seriesrlc('R',Rvalue,'L',Lvalue,'C',Cvalue)
```

Description

Use the `seriesrlc` class to represent a component as a resistor, inductor, and capacitor connected in series.

The series RLC network object is a 2-port network as shown in the following circuit diagram.



`h = rfckt.seriesrlc` returns a series RLC network object whose properties all have their default values. The default object is equivalent to a pass-through 2-port network, i.e., the resistor, inductor, and capacitor are each replaced by a short circuit.

`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 that you do not specify retain their default values, allowing you to specify a network of a single resistor, inductor, or capacitor.

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`C`	Capacitance value
`L`	Inductance value
`Name`	Object name

nPort	Number of ports
R	Resistance value

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
extract	Extract array of network parameters from data object
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale

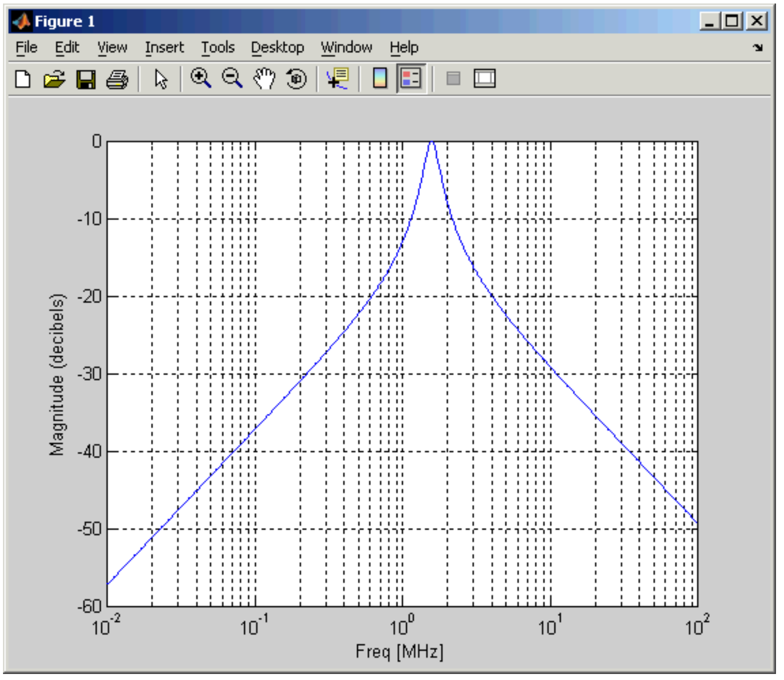
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart

smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

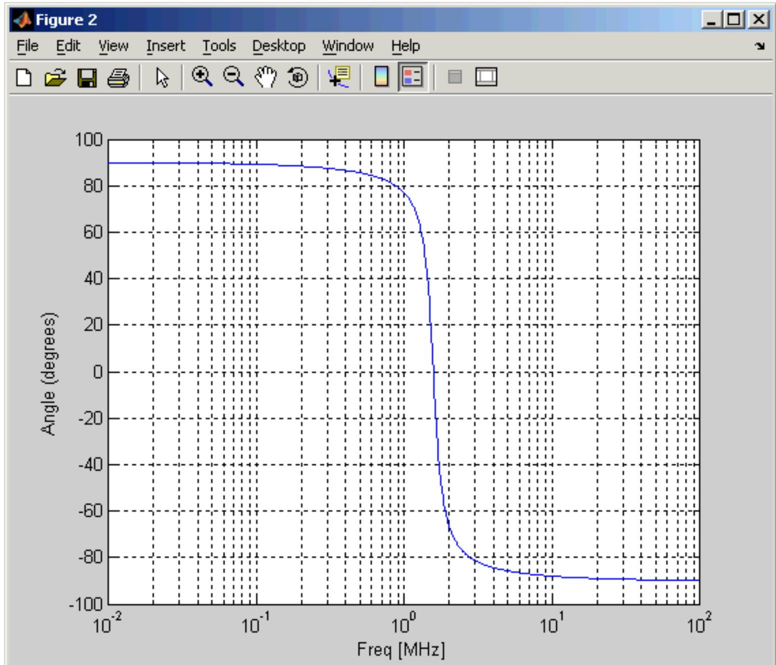
This example creates a series LC resonator and examines its frequency response. It first creates the circuit object and then uses the `analyze` method to calculate its frequency response. Finally, it plots the results — first, the magnitude in decibels (dB):

```
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

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

See Also

`rfckt.shuntrlc`

rfckt.shuntrlc

Purpose

Shunt RLC component

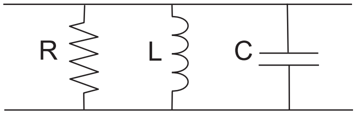
Syntax

```
h = rfckt.shuntrlc
h = rfckt.shuntrlc('R',Rvalue,'L',Lvalue,'C',Cvalue)
```

Description

Use the `shuntrlc` class to represent a component as a resistor, inductor, and capacitor connected in a shunt configuration.

The shunt RLC network object is a 2-port network as shown in the following circuit diagram.



`h = rfckt.shuntrlc` returns a shunt RLC network object whose properties all have their default values. The default object is equivalent to a pass-through 2-port network; i.e., the resistor, inductor, and capacitor are each replaced by a short circuit.

`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 that you do not specify retain their default values, allowing you to specify a network of a single resistor, inductor, or capacitor.

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`C`	Capacitance value
`L`	Inductance value
`Name`	Object name

`nPort`	Number of ports
`R`	Resistance value

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
extract	Extract array of network parameters from data object
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale

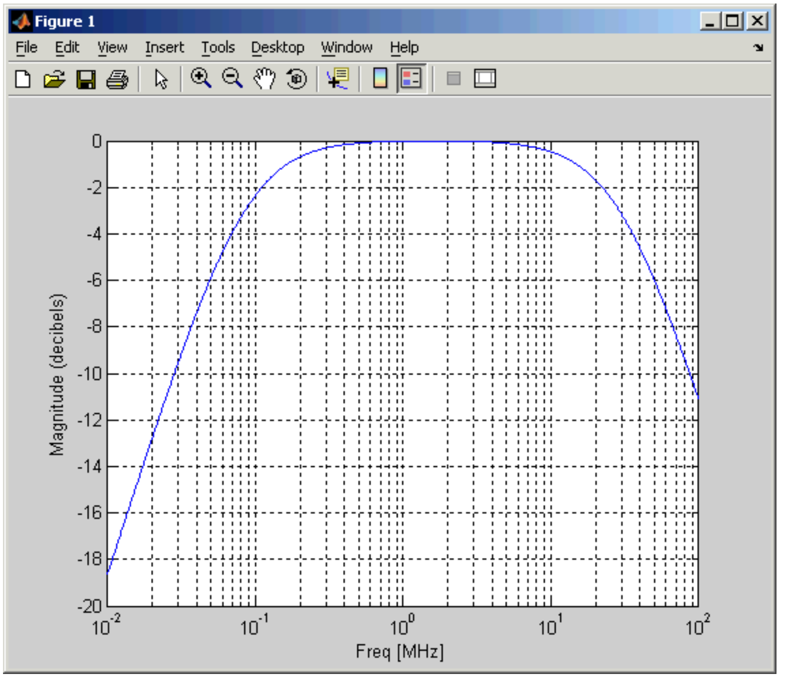
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart

smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

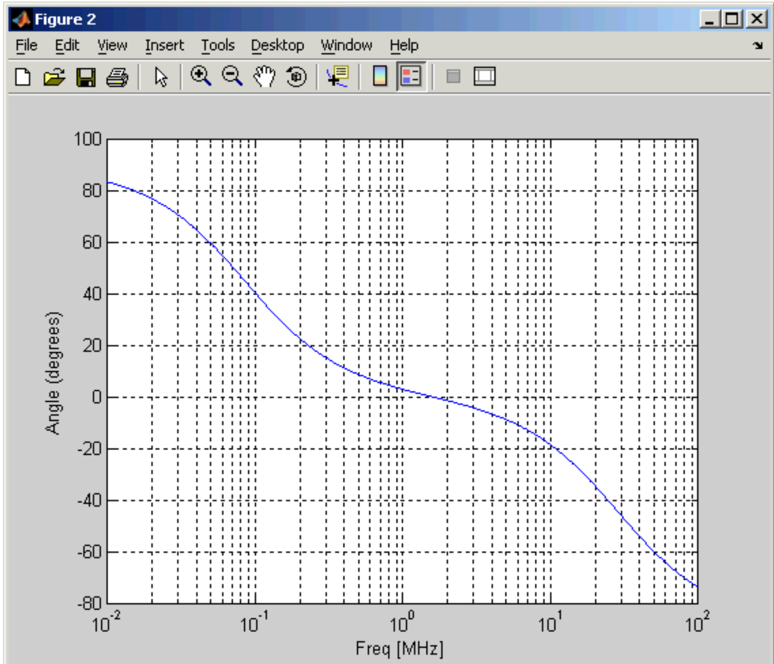
This example creates a shunt LC resonator and examines its frequency response. It first creates the circuit object and then uses the `analyze` method to calculate its frequency response. Finally, it plots the results — first, the magnitude in decibels (dB):

```
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

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

See Also

`rfckt.seriesrlc`

Purpose

Two-wire transmission line

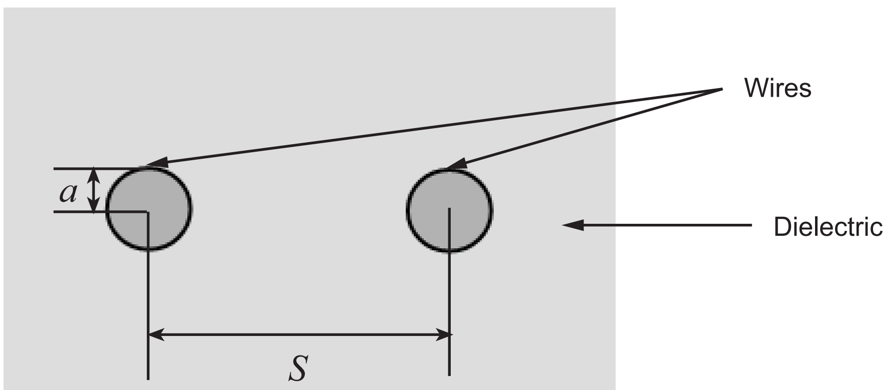
Syntax

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

Description

Use the `twowire` class to represent two-wire transmission lines that are characterized by line dimensions, stub type, and termination.

A two-wire transmission line is shown in cross-section in the following figure. Its physical characteristics include the radius of the wires a, the separation or physical distance between the wire centers S, and the relative permittivity and permeability of the wires. RF Toolbox software assumes the relative permittivity and permeability are uniform.



`h = rfckt.twowire` returns a two-wire transmission line object whose properties are set to their default values.

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

Construction

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`EpsilonR`	Relative permittivity of dielectric
`LineLength`	Transmission line length
`LossTangent`	Tangent of loss angle
`MuR`	Relative permeability of dielectric
`Name`	Object name
`nPort`	Number of ports
`Radius`	Wire radius
`Separation`	Distance between wires
`SigmaCond`	Conductor conductivity
`StubMode`	Type of stub
`Termination`	Stub transmission line termination

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart

circle	Draw circles on Smith chart
getz0	Characteristic impedance of transmission line object
getz0	Characteristic impedance of transmission line object
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates

polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

```
tx1=rfckt.twowire('Radius',7.5e-4)

tx1 =

           Name: 'Two-Wire Transmission Line'
           nPort: 2
           AnalyzedResult: []
           LineLength: 0.0100
           StubMode: 'NotAStub'
```

```
Termination: 'NotApplicable'
Radius: 7.5000e-004
Separation: 0.0016
MuR: 1
EpsilonR: 2.3000
LossTangent: 0
SigmaCond: Inf
```

References

Pozar, David M. *Microwave Engineering*, John Wiley & Sons, Inc., 2005.

See Also

`rfckt.coaxial` | `rfckt.cpw` | `rfckt.microstrip` | `rfckt.parallelplate` | `rfckt.rlcgline` | `rfckt.txline`

Purpose

General transmission line

Syntax

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

Description

Use the `txline` class to represent transmission lines that are characterized by line loss, line length, stub type, and termination.

`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 that you do not specify retain their default values.

Properties

`AnalyzedResult`	Computed S-parameters, noise figure, OIP3, and group delay values
`Freq`	Frequency data
`IntpType`	Interpolation method
`LineLength`	Transmission line length
`Loss`	Transmission line loss
`Name`	Object name
`nPort`	Number of ports
`PV`	Phase velocity
`StubMode`	Type of stub
`Termination`	Stub transmission line termination
`Z0`	Characteristic impedance

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
getz0	Characteristic impedance of transmission line object
getz0	Characteristic impedance of transmission line object
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane

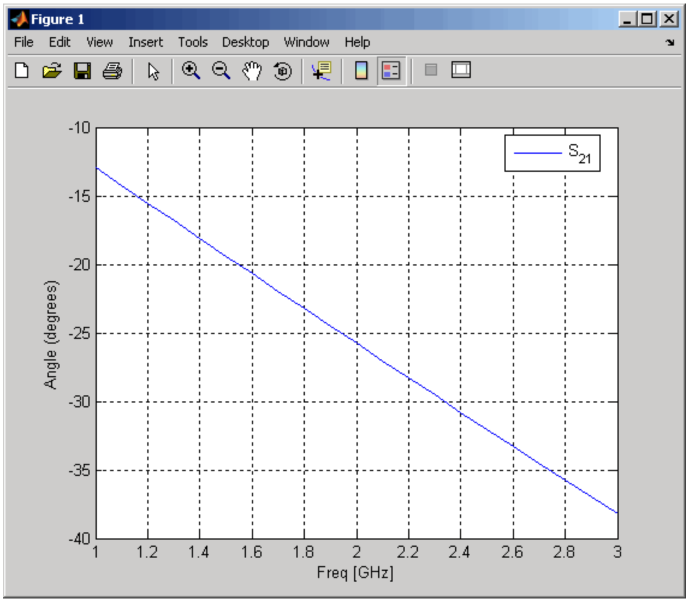
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart

write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

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 S_{21} network parameters, using the `'angle'` format, on the X-Y plane.

```
trl = rfckt.txline('Z0',75)

trl =

             Name: 'Transmission Line'
             nPort: 2
             AnalyzedResult: []
             LineLength: 0.0100
             StubMode: 'NotAStub'
             Termination: 'NotApplicable'
             Freq: 1.0000e+009
             Z0: 75
             PV: 299792458
             Loss: 0
             IntpType: 'Linear'

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



References

Ludwig, R. and P. Bretchko, *RF Circuit Design: Theory and Applications*, Prentice-Hall, 2000.

See Also

`rfckt.coaxial` | `rfckt.cpw` | `rfckt.microstrip` | `rfckt.parallelplate` | `rfckt.rlcgline` | `rfckt.twowire`

rfdata.data

Purpose

Store result of circuit object analysis

Syntax

```
h = rfdata.data
h = rfdata.data('Property1',value1,'Property2',value2,...)
```

Description

Use the `data` class to store S-parameters, noise figure in decibels, and frequency-dependent, third-order output (OIP3) intercept points.

There are three ways to create an `rfdata.data` object:

- You can construct it by specifying its properties from workspace data using the `rfdata.data` constructor.
- You can create it from file data using the `read` method.
- You can perform frequency domain analysis of a circuit object using the `analyze` method, and RF Toolbox software stores the results in an `rfdata.data` object.

`h = rfdata.data` returns a data object whose properties all have their default values.

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

Use the `read` method to read data from a file.

Properties

`Freq`	Frequency data
`GroupDelay`	Group delay data
`IntpType`	Interpolation method
`Name`	Object name
`NF`	Noise figure
`OIP3`	Output third-order intercept point

`S_Parameters`	S-parameter data
`Z0`	Reference impedance
`ZL`	Load impedance
`ZS`	Source impedance

Methods

analyze	Analyze circuit object in frequency domain
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith chart
circle	Draw circles on Smith chart
extract	Extract array of network parameters from data object
extract	Extract array of network parameters from data object
getop	Display operating conditions
getop	Display operating conditions
listformat	List valid formats for specified circuit object parameter
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object

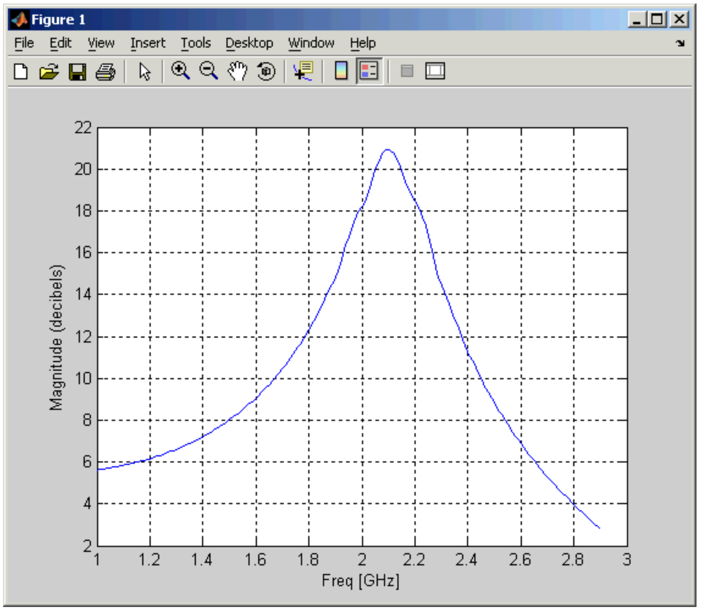
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
plotyy	Plot specified object parameters with *y*-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
polar	Plot specified circuit object parameters on polar coordinates
read	Read RF data from file to new or existing circuit or data object
read	Read RF data from file to new or existing circuit or data object
restore	Restore data to original frequencies
restore	Restore data to original frequencies

semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogx	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
semilogy	Plot specified circuit object parameters using log scale for *x*-axis
smith	Plot specified circuit object parameters on Smith chart
smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file
write	Write RF data from circuit or data object to file

Examples

Construct an RF data object from a `.s2p` data file.

```
file = 'default.s2p';
h = read(rfdata.data,file);  % Read file into data object.
figure
plot(h,'s21','db');   % Plot dB(S21) in XY plane.
```



See Also

rfdata.ip3 | rfdata.mixerspur | rfdata.network | rfdata.nf | rfdata.noise | rfdata.power

Purpose

Store frequency-dependent, third-order intercept points

Syntax

```
h = rfdata.ip3
h = rfdata.ip3('Type',value1,'Freq',value2,'Data',value3)
```

Description

Use the `ip3` class to store third-order intercept point specifications for a circuit object.

`h = rfdata.ip3` returns a data object for the frequency-dependent IP3, `h`, whose properties all have their default values.

`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

`Data`	Third-order intercept values
`Freq`	Frequency data
`Name`	Object name
`Type`	Power reference type

Examples

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

ip3data =

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

See Also

`rfdata.data` | `rfdata.mixerspur` | `rfdata.network` | `rfdata.nf` | `rfdata.noise` | `rfdata.power`

rfdata.mixerspur

Purpose

Store data from intermodulation table

Syntax

```
h = rfdata.mixerspur
h = rfdata.mixerspur('Data',value1,'PLORef',value2,'PinRef',
   'value3)
```

Description

Use the `mixerspur` class to store mixer spur power specifications for a circuit object.

`h = rfdata.mixerspur` returns a data object that defines an intermodulation table, `h`, whose properties all have their default values.

`h = rfdata.mixerspur('Data',value1,'PLORef',value2,'PinRef','value3)` returns a data object that defines an intermodulation table, `h`, based on the specified properties.

Properties

`Data`	Mixer spur power values
`Name`	Object name
`PinRef`	Reference input power
`PLORef`	Reference local oscillator power

Examples

```
spurs = rfdata.mixerspur...
        ('Data',[2 5 3; 1 0 99; 10 99 99],...
         'PinRef',3,'PLORef',5)

spurs =

      Name: 'Intermodulation table'
    PLORef: 5
    PinRef: 3
      Data: [3x3 double]
```

See Also

`rfdata.data` | `rfdata.ip3` | `rfdata.network` | `rfdata.nf` | `rfdata.noise` | `rfdata.power` | Visualizing Mixer Spurs

rfdata.network

Purpose

Store frequency-dependent network parameters

Syntax

```
h = rfdata.network
h = rfdata.network('Type',value1,'Freq',value2, Data',value3,
    'ZO',value4)
```

Description

Use the `network` class to store frequency-dependent S-, Y-, Z-, ABCD-, H-, G-, or T-parameters for a circuit object.

`h = rfdata.network` returns a data object for the frequency-dependent network parameters `h`, whose properties all have their default values.

`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

`Data`	Network parameter data
`Freq`	Frequency data
`Name`	Object name
`Type`	Type of network parameters.
`ZO`	Reference impedance

Examples

```
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];

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

See Also

`rfdata.data` | `rfdata.ip3` | `rfdata.mixerspur` | `rfdata.nf` | `rfdata.noise` | `rfdata.power`

rfdata.nf

Purpose

Store frequency-dependent noise figure data for amplifiers or mixers

Syntax

```
h = rfdata.nf
h = rfdata.nf('Freq',value1,'Data',value2)
```

Description

Use the `nf` class to store noise figure specifications for a circuit object.

`h = rfdata.nf` returns a data object for the frequency-dependent noise figure, `h`, whose properties all have their default values.

`h = rfdata.nf('Freq',value1,'Data',value2)` returns a data object for the frequency-dependent noise figure, `h`, based on the specified properties.

Properties

`Data`	Noise figure values
`Freq`	Frequency data
`Name`	Object name

Examples

```
f = 2.0e9;
nf = 13.3244;

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

See Also

`rfdata.data` | `rfdata.ip3` | `rfdata.mixerspur` | `rfdata.network` | `rfdata.noise` | `rfdata.power`

Purpose

Store frequency-dependent spot noise data for amplifiers or mixers

Syntax

```
h = rfdata.noise
h =  rfdata.noise('Freq',value1,'FMIN',value2,'GAMMAOPT',
   value3,'RN',value4)
```

Description

Use the `noise` class to store spot noise specifications for a circuit object.

`h = rfdata.noise` returns a data object for the frequency-dependent spot noise, `h`, whose properties all have their default values.

`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

`FMIN`	Minimum noise figure data
`Freq`	Frequency data
`GAMMAOPT`	Optimum source reflection coefficients
`Name`	Object name
`RN`	Equivalent normalized noise resistance data

Examples

```
f = [2.08 2.10]*1.0e9;
fmin = [12.08 13.40];
gopt = [0.2484-1.2102j 1.0999-0.9295j];
rn = [0.26 0.45];

noisedata = rfdata.noise('Freq',f,'FMIN',fmin,...
                         'GAMMAOPT',gopt,'RN',rn);
```

See Also

`rfdata.data` | `rfdata.mixerspur` | `rfdata.network` | `rfdata.nf` | `rfdata.power`

rfdata.power

Purpose

Store output power and phase information for amplifiers or mixers

Syntax

```
h = rfdata.power
h = rfdata.power(`property1',value1,'property2',value2,...)
```

Description

Use the `power` class to store output power and phase specifications for a circuit object.

`h = rfdata.power` returns a data object for the Pin/Pout power data, `h`, whose properties all have their default values.

`h = rfdata.power(`property1',value1,'property2',value2,...)` returns a data object for the Pin/Pout power data, `h`, based on the specified properties.

Properties

`Freq`	Frequency data
`Name`	Object name
`Phase`	Phase shift data
`Pin`	Input power data
`Pout`	Output power data

Examples

```
f = [2.08 2.10]*1.0e9;
phase = {[27.1 35.3],[15.4 19.3 21.1]};
pin = {[0.001 0.002],[0.001 0.005 0.01]};
pout = {[0.0025 0.0031],[0.0025 0.0028 0.0028]};
powerdata = rfdata.power;
powerdata.Freq = f;
powerdata.Phase = phase;
powerdata.Pin = pin;
powerdata.Pout = pout;
```

See Also

`rfdata.data` | `rfdata.ip3` | `rfdata.mixerspur` | `rfdata.network` | `rfdata.nf` | `rfdata.noise`

Purpose

Rational function model

Syntax

```
h = rfmodel.rational
h = rfmodel.rational('Property1',value1,'Property2',value2,
    ...)
```

Description

Use the `rational` class to represent RF components using a rational function model of the form

$$F(s) = \left(\sum_{k=1}^{n} \frac{C_k}{s - A_k} + D \right) e^{-s\tau}, \quad s = j2\pi f$$

There are two ways to construct an rfmodel.rational object:

- You can fit a rational function model to the component data using the `rationalfit` function.
- You can use the `rfmodel.rational` constructor to specify the pole-residue representation of the component directly.

`h = rfmodel.rational` returns a rational function model object whose properties are set to their default values.

`h = rfmodel.rational('Property1',value1,'Property2',value2,...)` returns a rational function model object, `h`, with the specified properties. Properties that you do not specify retain their default values.

Properties

`A`	Poles of rational function
`C`	Residues of rational function
`D`	Frequency response offset
`Delay`	Frequency response time delay
`Name`	Object name

Methods

freqresp	Frequency response of model object
freqresp	requency response of model object
ispassive	Check passivity of model object
ispassive	Check passivity of scalar model object
stepresp	response of model object
stepresp	Step-signal response of model object
timeresp	ime response for model object
timeresp	Time response for model object
writeva	Write Verilog-A description of RF model object
writeva	Write Verilog-A description of RF model object

Examples

Fit a rational function to data from an `rfdata.data` object:

```
orig_data=read(rfdata.data,'default.s2p');
freq=orig_data.Freq;
data=orig_data.S_Parameters(2,1,:);
fit_data=rationalfit(freq,data)

fit_data =

     Name: 'Rational Function'
     A: [2x1 double]
     C: [2x1 double]
     D: 0
     Delay: 0
```

Define, evaluate, and visualize a rational function in three steps:

- Construct a rational function model, `rat`, with poles at -4 Mrad/s, -3 Grad/s, and -5 Grad/s and residues of 600 Mrad/s, 2 Grad/s and 4 Grad/s.
- Perform frequency-domain analysis from 1.0 MHz to 3.0 GHz.
- Plot the resulting frequency response in decibels on the X-Y plane.

```
rat=rfmodel.rational...
     ('A',[-5e9,-3e9,-4e6],...
      'C',[6e8,2e9,4e9]);      % Create model
f = [1e6:1.0e7:3e9];
[resp,freq]=freqresp(rat,f);   % Compute frequency response
figure
 plot(freq/1e9,db(resp));      % Plot frequency response
 xlabel('Frequency (GHz)')
 ylabel('Magnitude (dB)')
```

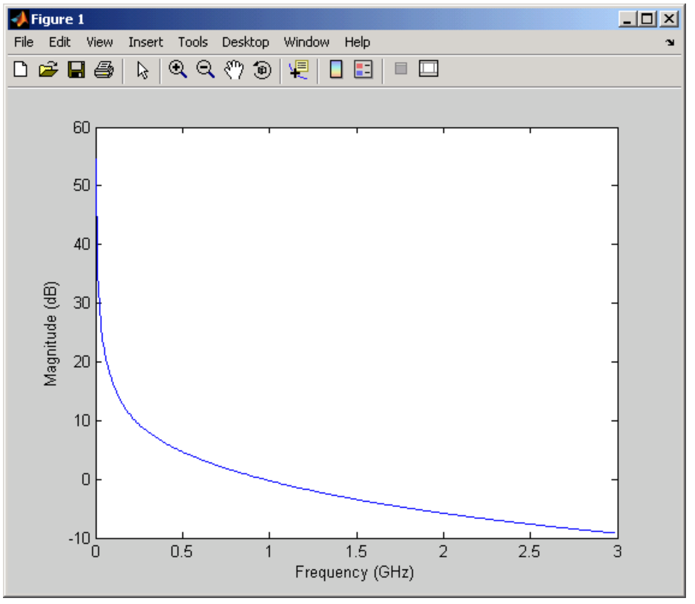


Figure 1
File Edit View Insert Tools Desktop Window Help
Magnitude (dB)
Frequency (GHz)
60
50
40
30
20
10
0
-10
0
0.5
1
1.5
2
2.5
3

Purpose Computed S-parameters, noise figure, OIP3, and group delay values

Values `rfdata.data` object

Description Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. The default is a 1-by-1 `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values that result from analyzing the values stored in the `default.amp` file at the frequencies stored in this file.

The `analyze` method computes the `AnalyzedResult` property using the data stored in the `rfckt.amplifier` object properties as follows:

- The `analyze` method uses the data stored in the `'NoiseData'` property of the `rfckt.amplifier` object to calculate the noise figure.
- The `analyze` method uses the data stored in the `'NonlinearData'` property of the `rfckt.amplifier` object to calculate OIP3.

  If power data exists in the `'NonlinearData'` property, the block extracts the AM/AM and AM/PM nonlinearities from the power data.

  If the `'NonlinearData'` property contains only IP3 data, the method computes and adds the nonlinearity by:

  1. Using the third-order input intercept point value in dBm to compute the factor, *f*, that scales the input signal before the amplifier object applies the nonlinearity:

     $$F_{AM/AM}(u) = u - \frac{u^3}{3}$$

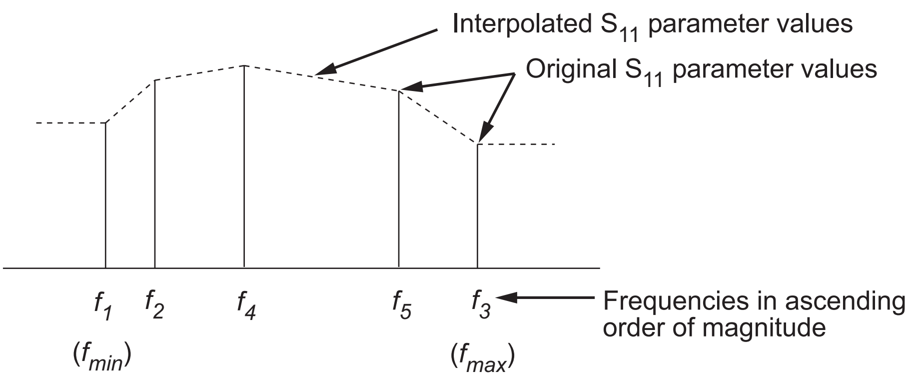
  2. Computing the scaled input signal by multiplying the amplifier input signal by *f*.
  3. Limiting the scaled input signal to a maximum value of 1.
  4. Applying an AM/AM conversion to the amplifier gain, according to the following cubic polynomial equation:

$$F_{AM/AM}(u) = u - \frac{u^3}{3}$$

where u is the magnitude of the scaled input signal, which is a unitless normalized input voltage.

- The `analyze` method uses the data stored in the `'NetworkData'` property of the `rfckt.amplifier` object to calculate the group delay values of the amplifier at the frequencies specified in `freq`, as described in the `analyze` reference page.
- The `analyze` method uses the data stored in the `'NetworkData'` property of the `rfckt.amplifier` object to calculate the S-parameter values of the amplifier at the frequencies specified in `freq`. If the `'NetworkData'` property contains network Y- or Z-parameters, the `analyze` method first converts the parameters to S-parameters. Using the interpolation method you specify with the `'IntpType'` property, the `analyze` method interpolates the S-parameter values to determine their values at the specified frequencies.

  Specifically, the `analyze` method 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 S_{11} parameters at five different frequencies.



For more information, see “One-Dimensional Interpolation” and the `interp1` reference page in the MATLAB documentation.

As shown in the preceding diagram, the `analyze` method 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, so you need to specify network parameter values over a range of frequencies that is wide enough to account for the amplifier behavior.

Examples

```
amp = rfckt.amplifier;
amp.AnalyzedResult

ans =

            Name: 'Data object'
            Freq: [191x1 double]
            S_Parameters: [2x2x191 double]
            GroupDelay: [191x1 double]
            NF: [191x1 double]
            OIP3: [191x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Linear'
```

rfckt.amplifier.IntpType property

Purpose

Interpolation method

Values

`'Linear'` (default), `'Spline'`, or `'Cubic'`

Description

The `analyze` method is flexible in that it does not require the frequencies of the specified S-parameters to match the requested analysis frequencies. If needed, `analyze` applies the interpolation and extrapolation method specified in the `IntpType` property to the specified data to create a new set of data at the requested analysis frequencies. The following table lists the available interpolation methods and describes each one.

Method	Description
`Linear` (default)	Linear interpolation
`Spline`	Cubic spline interpolation
`Cubic`	Piecewise cubic Hermite interpolation

Examples

```
amp = rfckt.amplifier;
amp.IntpType = 'cubic'

amp =

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

Purpose Object name

Values `'Amplifier'`

Description Read-only string that contains the name of the object.

Examples

```
amp = rfckt.amplifier;
amp.Name

ans =

          Amplifier
```

rfckt.amplifier.NetworkData property

Purpose

Network parameter information

Values

`rfdata.network` object

Description

An `rfdata.network` object that stores network parameter data. The default network parameter values are taken from the `'default.amp'` data file.

Examples

```
amp = rfckt.amplifier;
amp.NetworkData

ans =

    Name: 'Network parameters'
    Type: 'S_PARAMETERS'
    Freq: [191x1 double]
    Data: [2x2x191 double]
    Z0: 50
```

Purpose Noise information

Values Scalar noise figure in decibels, `rfdata.noise` object or `rfdata.nf` object

Description A scalar value or object that stores noise data. The default is an `rfdata.noise` object whose values are taken from the `'default.amp'` data file.

Examples

```
amp = rfckt.amplifier;
amp.NoiseData

ans =

        Name: 'Spot noise data'
        Freq: [9x1 double]
        FMIN: [9x1 double]
        GAMMAOPT: [9x1 double]
        RN: [9x1 double]
```

rfckt.amplifier.NonlinearData property

Purpose Nonlinearity information

Values Scalar OIP3 in decibels relative to one milliwatt, `rfdata.power` object or `rfdata.ip3` object

Description A scalar value or object that stores nonlinearity data. The default is an `rfdata.power` object whose values are taken from the `'default.amp'` data file.

Examples

```
amp = rfckt.amplifier;
amp.NonlinearData

ans =

      Name: 'Power data'
      Freq: 2.1000e+009
      Pin: {[20x1 double]}
      Pout: {[20x1 double]}
      Phase: {[20x1 double]}
```

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
amp = rfckt.amplifier;
amp.nPort

ans =

     2
```

rfckt.cascade.AnalyzedResult property

Purpose

Computed S-parameters, noise figure, OIP3, and group delay values

Values

`rfdata.data` object

Description

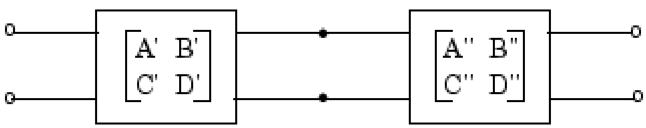
Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

The `analyze` method computes the `AnalyzedResult` property using the data stored in the `Ckts` property as follows:

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

  The following 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`.

- The `analyze` method calculates the noise figure for an N-element cascade. First, the method calculates noise correlation matrices C_A' and C_A'', corresponding to the first two matrices in the cascade, using the following equation:

$$C_A = 2kT\begin{bmatrix} R_n & \frac{NF_{\min}-1}{2} - R_n Y_{opt}{}^* \\ \frac{NF_{\min}-1}{2} - R_n Y_{opt} & R_n \left|Y_{opt}\right|^2 \end{bmatrix}$$

  where `k` is Boltzmann's constant, and `T` is the noise temperature in Kelvin.

  The method combines C_A' and C_A'' into a single correlation matrix C_A using the equation

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

  By applying this equation recursively, the method obtains a noise correlation matrix for the entire cascade. The method then calculates the noise factor, `F`, from the noise correlation matrix of as follows:

$$F = 1 + \frac{z^+ C_A z}{2kT\,\mathrm{Re}\{Z_S\}}$$

$$z = \begin{bmatrix} 1 \\ Z_S{}^* \end{bmatrix}$$

  In the two preceding equations, Z_S is the nominal impedance, which is 50 ohms, and z^+ is the Hermitian conjugation of z.

- The `analyze` method 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 OIP_3 of the n^{th} element.

- The `analyze` method uses the cascaded S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

Examples

Analyze a cascade of three circuit objects:

```
amp = rfckt.amplifier('IntpType','cubic');
tx1 = rfckt.txline;
tx2 = rfckt.txline;
casc = rfckt.cascade('Ckts',{tx1,amp,tx2});
analyze(casc,[1e9:1e7:2e9]);
casc.AnalyzedResult
```

References

Hillbrand, H. and P.H. Russer, "An Efficient Method for Computer Aided Noise Analysis of Linear Amplifier Networks," *IEEE Transactions on Circuits and Systems*, Vol. CAS-23, Number 4, pp. 235–238, 1976.

Purpose Circuit objects in network

Values Cell

Description Cell array containing handles to all circuit objects in the network, in order from source to load. All circuits must be 2-port. This property is empty by default.

Examples

```
amp = rfckt.amplifier('IntpType','cubic');
tx1 = rfckt.txline;
tx2 = rfckt.txline;
casc = rfckt.cascade;
casc.Ckts = {tx1,amp,tx2};
casc.Ckts

ans =

[1x1 rfckt.txline] [1x1 rfckt.amplifier] [1x1 rfckt.txline]
```

rfckt.cascade.Name property

Purpose Object name

Values `'Cascaded Network'`

Description Read-only string that contains the name of the object.

Examples

```
casc = rfckt.cascade;
casc.Name

ans =

          Cascaded Network
```

Purpose
Number of ports

Values
2

Description
A read-only integer that indicates the object has two ports.

Examples

```
casc = rfckt.cascade;
casc.nPort

ans =

     2
```

rfckt.coaxial.AnalyzedResult property

Purpose

Computed S-parameters, noise figure, OIP3, and group delay values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

The `analyze` method treats the transmission line as a 2-port linear network. It computes the `AnalyzedResult` property of a stub or as a stubless line using the data stored in the `rfckt.coaxial` object properties as follows:

- If you model the transmission line as a stubless line, the `analyze` method first calculates the ABCD-parameters at each frequency contained in the modeling frequencies vector. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

  The `analyze` method calculates the ABCD-parameters using the physical length of the transmission line, d, and the complex propagation constant, k, using the following equations:

$$A = \frac{e^{kd} + e^{-kd}}{2}$$

$$B = \frac{Z_0 * \left(e^{kd} - e^{-kd}\right)}{2}$$

$$C = \frac{e^{kd} - e^{-kd}}{2 * Z_0}$$

$$D = \frac{e^{kd} + e^{-kd}}{2}$$

  Z_0 and k are vectors whose elements correspond to the elements of f, the vector of frequencies specified in the `analyze` input argument `freq`. Both can be expressed in terms of the resistance (R), inductance

(L), conductance (G), and capacitance (C) per unit length (meters) as follows:

$$Z_0 = \sqrt{\frac{R + j2\pi fL}{G + j2\pi fC}}$$

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

where

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

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

$$G = \frac{2\pi\omega\varepsilon''}{\ln\left(\frac{b}{a}\right)}$$

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

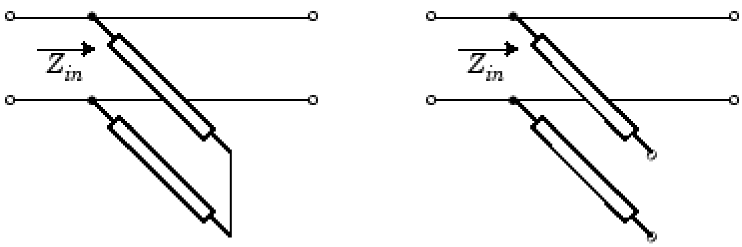
In these equations:

- a is the radius of the inner conductor.
- b is the radius of the outer conductor.
- σ_{cond} is the conductivity in the conductor.
- μ is the permeability of the dielectric.
- ε is the permittivity of the dielectric.
- ε'' is the imaginary part of ε, $\varepsilon'' = \varepsilon_0\varepsilon_r \tan\delta$, where:
  - ε_0 is the permittivity of free space.
  - ε_r is the `EpsilonR` property value.
  - $\tan\delta$ is the `LossTangent` property value.

- δ_{cond} is the skin depth of the conductor, which the method calculates as $1/\sqrt{\pi f \mu \sigma_{cond}}$.
- f is a vector of modeling frequencies determined by the Outport block.

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

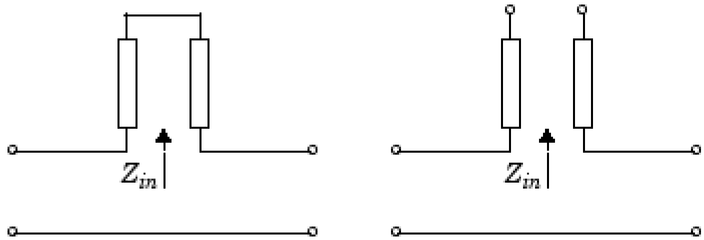
  When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown in the following figure.

  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 in the following figure.



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

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

The `analyze` method uses the S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

Examples

```
tx1 = rfckt.coaxial;
analyze(tx1,[1e9,2e9,3e9]);
tx1.AnalyzedResult

ans =

            Name: 'Data object'
            Freq: [3x1 double]
            S_Parameters: [2x2x3 double]
            GroupDelay: [3x1 double]
            NF: [3x1 double]
            OIP3: [3x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Linear'
```

rfckt.coaxial.EpsilonR property

Purpose

Relative permittivity of dielectric

Values

Scalar

Description

The ratio of the permittivity of the dielectric, ε, to the permittivity of free space, ε_0. The default value is `2.3`.

Examples

Change the relative permittivity of the dielectric:

```
tx1=rfckt.coaxial;
tx1.EpsilonR=2.7;
```

rfckt.coaxial.InnerRadius property

Purpose Inner conductor radius

Values Scalar

Description The radius of the inner conductor, in meters. The default is `7.25e-4`.

Examples

```
tx1=rfckt.coaxial;
tx1.InnerRadius=2.5e-4;
```

rfckt.coaxial.LineLength property

Purpose Transmission line length

Values Scalar

Description The physical length of the transmission line in meters. The default is 0.01.

Examples

```
tx1 = rfckt.coaxial;
tx1.LineLength = 0.001;
```

Purpose Tangent of loss angle

Values Scalar

Description The loss angle tangent of the dielectric. The default is `0`.

Examples

```
tx1=rfckt.coaxial;
tx1.LossTangent=0.002;
```

rfckt.coaxial.MuR property

Purpose

Relative permeability of dielectric

Values

Scalar

Description

The ratio of the permeability of the dielectric, μ, to the permeability in free space, μ_0. The default value is 1.

Examples

Change the relative permeability of the dielectric:

```
tx1=rfckt.coaxial;
tx1.MuR=0.8;
```

Purpose Object name

Values `'Coaxial Transmission Line'`

Description Read-only string that contains the name of the object.

Examples

```
tx1 = rfckt.coaxial;
tx1.Name

ans =

           Coaxial Transmission Line
```

rfckt.coaxial.OuterRadius property

Purpose Outer conductor radius

Values Scalar

Description The radius of the outer conductor, in meters. The default is `0.0026`.

Examples

```
tx1=rfckt.coaxial;
tx1.OuterRadius=0.0031;
```

Purpose Conductor conductivity

Values Scalar

Description Conductivity, in Siemens per meter (S/m), of the conductor. The default is `Inf`.

Examples

```
tx1=rfckt.coaxial;
tx1.SigmaCond=5.81e7;
```

rfckt.coaxial.StubMode property

Purpose Type of stub

Values `'NotAStub'` (default), `'Series'`, or `'Shunt'`

Description String that specifies what type of stub, if any, to include in the transmission line model.

Examples

```
tx1 = rfckt.coaxial;
tx1.StubMode = 'Series';
```

Purpose

Stub transmission line termination

Values

`'NotApplicable'` (default), `'Open'`, or `'Short'`.

Description

String that specifies what type of termination to use for `'Shunt'` and `'Series'` stub modes. `Termination` is ignored if the line has no stub. Use `'NotApplicable'` when `StubMode` is `'NotAStub'`.

Examples

```
tx1 = rfckt.coaxial;
tx1.StubMode = 'Series';
tx1.Termination = 'Short';
```

rfckt.coaxial.nPort property

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
tx1 = rfckt.coaxial;
tx1.nPort

ans =

     2
```

Purpose Computed S-parameters, noise figure, OIP3, and group delay values

Values `rfdata.data` object

Description Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

The `analyze` method treats the transmission line as a 2-port linear network. It computes the `AnalyzedResult` property of a stub or as a stubless line using the data stored in the `rfckt.cpw` object properties as follows:

- If you model the transmission line as a stubless line, the `analyze` method first calculates the ABCD-parameters at each frequency contained in the modeling frequencies vector. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

  The `analyze` method calculates the ABCD-parameters using the physical length of the transmission line, *d*, and the complex propagation constant, *k*, using the following equations:

$$A = \frac{e^{kd} + e^{-kd}}{2}$$

$$B = \frac{Z_0 * \left(e^{kd} - e^{-kd}\right)}{2}$$

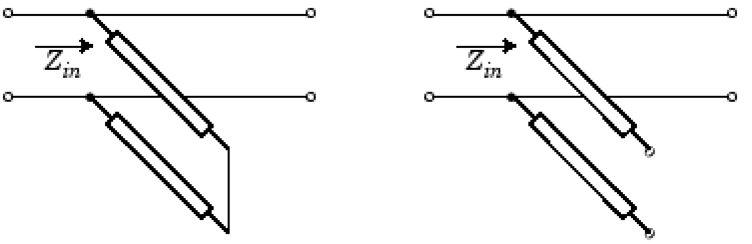
$$C = \frac{e^{kd} - e^{-kd}}{2 * Z_0}$$

$$D = \frac{e^{kd} + e^{-kd}}{2}$$

  Z_0 and k are vectors whose elements correspond to the elements of f, the vector of frequencies specified in the `analyze` input argument `freq`. Both can be expressed in terms of the specified conductor strip width, slot width, substrate height, conductor strip thickness, relative permittivity constant, conductivity and dielectric loss tangent of the transmission line, as described in [1].

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

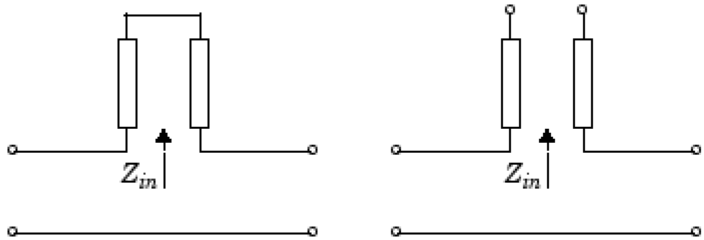
  When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown in the following figure.

  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 in the following figure.



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

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

The `analyze` method uses the S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

Examples

```
tx1 = rfckt.cpw;
analyze(tx1,[1e9,2e9,3e9]);
tx1.AnalyzedResult

ans =

              Name: 'Data object'
              Freq: [3x1 double]
              S_Parameters: [2x2x3 double]
              NF: [3x1 double]
              GroupDelay: [3x1 double]
              OIP3: [3x1 double]
              Z0: 50
              ZS: 50
              ZL: 50
              IntpType: 'Linear'
```

rfckt.cpw.ConductorWidth property

Purpose Conductor width

Values Scalar

Description Physical width, in meters, of the conductor. The default is `0.6e-4`.

Examples

```
tx1=rfckt.cpw;
tx1.ConductorWidth=0.001;
```

Purpose Relative permittivity of dielectric

Values Scalar

Description The ratio of the permittivity of the dielectric, ε, to the permittivity of free space, ε_0. The default value is `9.8`.

Examples Change the relative permittivity of the dielectric:

```
tx1=rfckt.cpw;
tx1.EpsilonR=2.7;
```

rfckt.cpw.Height property

Purpose Dielectric thickness

Values Scalar

Description Physical height, in meters, of the dielectric on which the conductor resides. The default is `0.635e-4`.

Examples

```
tx1=rfckt.cpw;
tx1.Height=0.001;
```

Purpose Transmission line length

Values Scalar

Description The physical length of the transmission line in meters. The default is `0.01`.

Examples

```
tx1 = rfckt.cpw;
tx1.LineLength = 0.001;
```

rfckt.cpw.LossTangent property

Purpose Tangent of loss angle

Values Scalar

Description The loss angle tangent of the dielectric. The default is `0`.

Examples

```
tx1 = rfckt.cpw;
tx1.LossTangent

ans =

     0
```

Purpose Object name

Values `'Coplanar Waveguide Transmission Line'`

Description Read-only string that contains the name of the object.

Examples

```
tx1 = rfckt.cpw;
tx1.Name

ans =

          Coplanar Waveguide Transmission Line
```

rfckt.cpw.SigmaCond property

Purpose

Conductor conductivity

Values

Scalar

Description

Conductivity, in Siemens per meter (S/m), of the conductor. The default is `Inf`.

Examples

```
tx1=rfckt.cpw;
tx1.SigmaCond=5.81e7;
```

Purpose Width of slot

Values Scalar

Description Physical width, in meters, of the slot. The default is `0.2e-4`.

Examples

```
tx1=rfckt.cpw;
tx1.SlotWidth=0.002;
```

rfckt.cpw.StubMode property

Purpose

Type of stub

Values

`'NotAStub'` (default), `'Series'`, or `'Shunt'`

Description

String that specifies what type of stub, if any, to include in the transmission line model.

Examples

```
tx1 = rfckt.cpw;
tx1.StubMode = 'Series';
```

Purpose Stub transmission line termination

Values `'NotApplicable'` (default), `'Open'`, or `'Short'`.

Description String that specifies what type of termination to use for `'Shunt'`and `'Series'` stub modes. `Termination` is ignored if the line has no stub. Use `'NotApplicable'` when `StubMode` is `'NotAStub'`.

Examples

```
tx1 = rfckt.cpw;
tx1.StubMode = 'Series';
tx1.Termination = 'Short';
```

rfckt.cpw.Thickness property

Purpose Conductor thickness

Values Scalar

Description Physical thickness, in meters, of the conductor. The default is `0.005e-6`.

Examples

```
tx1=rfckt.cpw;
tx1.Thickness=2e-5;
```

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
tx1 = rfckt.cpw;
tx1.nPort

ans =

     2
```

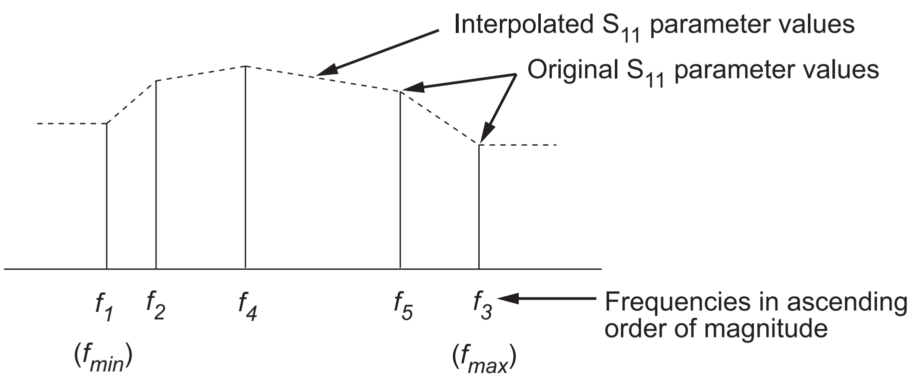
rfckt.datafile.AnalyzedResult property

Purpose Computed S-parameters, noise figure, OIP3, and group delay values

Values `rfdata.data` object

Description Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. The default is a 1-by-1 `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values that are the result of analyzing the values stored in the `passive.s2p` file at the frequencies stored in this file.

The `analyze` method computes the `AnalyzedResult` property using the data stored in the `File` object property. If the file you specify with this 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 S_{11} parameters at five different frequencies.



For more information, see "One-Dimensional Interpolation" and the `interp1` reference page in the MATLAB documentation.

As shown in the preceding diagram, the `analyze` method 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, so you need to specify network parameter values over a range of frequencies that is wide enough to account for the component behavior.

The `analyze` method uses the S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

Examples

```
data = rfckt.datafile;
data.AnalyzedResult

ans =

            Name: 'Data object'
            Freq: [202x1 double]
            S_Parameters: [2x2x202 double]
            GroupDelay: [202x1 double]
            NF: [202x1 double]
            OIP3: [202x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Linear'
```

rfckt.datafile.File property

Purpose File containing circuit data

Values String

Description The name of the `.snp`, `.ynp`, `.znp`, or `.hnp` file describing the circuit, where `n` is the number of ports. The default file name is `'passive.s2p'`.

Examples

```
data=rfckt.datafile;
data.File='default.s2p'

data =

           Name: 'Data File'
           nPort: 2
           AnalyzedResult: [1x1 rfdata.data]
           IntpType: 'Linear'
           File: 'default.s2p'
```

Purpose

Interpolation method

Values

`'Linear'` (default), `'Spline'`, or `'Cubic'`

Description

The `analyze` method is flexible in that it does not require the frequencies of the specified S-parameters to match the requested analysis frequencies. If needed, `analyze` applies the interpolation and extrapolation method specified in the `IntpType` property to the specified data to create a new set of data at the requested analysis frequencies. The following table lists the available interpolation methods and describes each one.

Method	Description
`Linear` (default)	Linear interpolation
`Spline`	Cubic spline interpolation
`Cubic`	Piecewise cubic Hermite interpolation

Examples

```
data = rfckt.datafile;
data.IntpType = 'cubic';
```

rfckt.datafile.Name property

Purpose Object name

Values `'Data object'`

Description Read-only string that contains the name of the object.

Examples

```
data = rfckt.datafile;
data.Name

ans =

           Data object
```

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
data = rfckt.datafile;
data.nPort

ans =

     2
```

rfckt.delay.AnalyzedResult property

Purpose

Computed S-parameters, noise figure, OIP3, and group delay values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

The `analyze` method treats the delay line, which can be lossy or lossless, as a 2-port linear network. It computes the `AnalyzedResult` property of the delay line using the data stored in the `rfckt.delay` object properties by calculating the S-parameters for the specified frequencies. This calculation is based on the values of the delay line's `loss`, α, and time delay, D.

$$\begin{cases} S_{11} = 0 \\ S_{12} = e^{-p} \\ S_{21} = e^{-p} \\ S_{22} = 0 \end{cases}$$

Above, $p = \alpha_a + i\beta$, 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

$$\alpha_a = -\ln\left(10^{\alpha/20}\right)$$

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

$$\beta = 2\pi fD$$

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

The `analyze` method uses the S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

Examples

Compute S-parameters, noise figure, OIP3, and group delay values:

```
del = rfckt.delay;
analyze(del,[1e9,2e9,3e9]);
del.AnalyzedResult
```

rfckt.delay.Loss property

Purpose Delay line loss

Values Scalar

Description Line loss value, in decibels. Line loss is the reduction in strength of the signal as it travels over the delay line and must be nonnegative. The default is `0`.

Examples

```
del = rfckt.delay;
del.Loss = 10;
```

Purpose Object name

Values `'Delay Line'`

Description Read-only string that contains the name of the object.

Examples

```
del = rfckt.delay;
del.Name

ans =

          Delay Line
```

rfckt.delay.TimeDelay property

Purpose Delay introduced by line

Values Scalar

Description The amount of time delay, in seconds. The default is `1.0000e-012`.

Examples

```
del = rfckt.delay;
del.TimeDelay = 1e-9;
```

Purpose Characteristic impedance

Values Scalar

Description The characteristic impedance, in ohms, of the delay line. The default is 50 ohms.

Examples

```
del = rfckt.delay;
del.Z0 = 75;
```

rfckt.delay.nPort property

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
del = rfckt.delay;
del.nPort

ans =

     2
```

Purpose

Computed S-parameters, noise figure, OIP3, and group delay values

Values

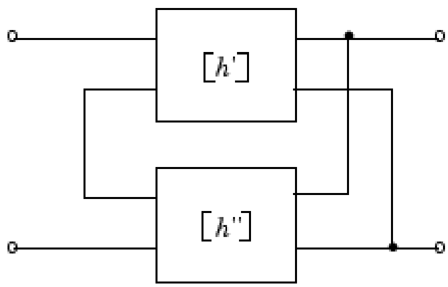
`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

The `analyze` method computes the S-parameters of the `AnalyzedResult` property using the data stored in the `Ckts` property as follows:

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



where

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

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

- The `analyze` method 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.

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

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

The `analyze` method uses the hybrid S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

Examples

```
tx1 = rfckt.txline;
tx2 = rfckt.txline;
hyb = rfckt.hybrid('Ckts',{tx1,tx2})
analyze(hyb,[1e9:1e7:2e9]);
hyb.AnalyzedResult

ans =

            Name: 'Data object'
            Freq: [101x1 double]
    S_Parameters: [2x2x101 double]
      GroupDelay: [101x1 double]
              NF: [101x1 double]
            OIP3: [101x1 double]
              Z0: 50
              ZS: 50
              ZL: 50
        IntpType: 'Linear'
```

Purpose Circuit objects in network

Values Cell

Description Cell array containing handles to all circuit objects in the network. All circuits must be 2-port and linear. This property is empty by default.

Examples

```
tx1 = rfckt.txline;
tx2 = rfckt.txline;
hyb = rfckt.hybrid;
hyb.Ckts = {tx1,tx2};
hyb.Ckts

ans =

    [1x1 rfckt.txline] [1x1 rfckt.txline]
```

rfckt.hybrid.Name property

Purpose Object name

Values `'Hybrid Connected Network'`

Description Read-only string that contains the name of the object.

Examples

```
hyb = rfckt.hybrid;
hyb.Name

ans =

          Hybrid Connected Network
```

Purpose

Number of ports

Values

2

Description

A read-only integer that indicates the object has two ports.

Examples

```
hyb = rfckt.hybrid;
hyb.nPort

ans =

     2
```

rfckt.hybridg.AnalyzedResult property

Purpose

Computed S-parameters, noise figure, OIP3, and group delay values

Values

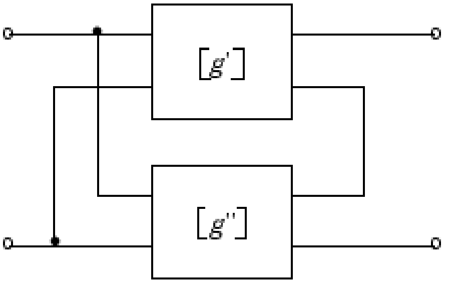
`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

The `analyze` method computes the S-parameters of the `AnalyzedResult` property using the data stored in the `Ckts` property as follows:

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



where

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

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

2 The analyze method 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.

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

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

The analyze method uses the inverse hybrid S-parameters to calculate the group delay values at the frequencies specified in the analyze input argument freq, as described in the analyze reference page.

Examples

```
tx1 = rfckt.txline;
tx2 = rfckt.txline;
invhyb = rfckt.hybridg('Ckts',{tx1,tx2})
analyze(invhyb,[1e9:1e7:2e9]);
invhyb.AnalyzedResult

ans =

            Name: 'Data object'
            Freq: [101x1 double]
    S_Parameters: [2x2x101 double]
      GroupDelay: [101x1 double]
              NF: [101x1 double]
            OIP3: [101x1 double]
              Z0: 50
              ZS: 50
              ZL: 50
        IntpType: 'Linear'
```

rfckt.hybridg.Ckts property

Purpose Circuit objects in network

Values Cell

Description Cell array containing handles to all circuit objects in the network. All circuits must be 2-port and linear. This property is empty by default.

Examples

```
tx1 = rfckt.txline;
tx2 = rfckt.txline;
invhyb = rfckt.hybridg;
invhyb.Ckts = {tx1,tx2};
invhyb.Ckts

ans =

    [1x1 rfckt.txline] [1x1 rfckt.txline]
```

Purpose Object name

Values `'Hybrid G Connected Network'`

Description Read-only string that contains the name of the object.

Examples

```
invhyb = rfckt.hybridg;
invhyb.Name

ans =

            Hybrid G Connected Network
```

rfckt.hybridg.nPort property

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
invhyb = rfckt.hybridg;
invhyb.nPort

ans =

     2
```

Purpose Computed S-parameters, noise figure, OIP3, and group delay values

Values `rfdata.data` object

Description Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

Examples

```
filter = rfckt.lcbandpasspi;
analyze(filter,[1e9,2e9,3e9]);
filter.AnalyzedResult

ans =

           Name: 'Data object'
           Freq: [3x1 double]
           S_Parameters: [2x2x3 double]
           GroupDelay: [3x1 double]
           NF: [3x1 double]
           OIP3: [3x1 double]
           Z0: 50
           ZS: 50
           ZL: 50
           IntpType: 'Linear'
```

rfckt.lcbandpasspi.C property

Purpose Capacitance data

Values Vector

Description Capacitance values in farads, in order from source to load, of all capacitors in the network. The length of the capacitance vector must be equal to the length of the vector you provide for `'L'`. All values must be strictly positive. The default is `[0.3579e-10, 0.0118e-10, 0.3579e-10]`.

Examples

```
filter=rfckt.lcbandpasspi;
filter.C = [10.1 4.5 14.2]*1e-12;
```

Purpose Inductance data

Values Vector

Description Inductance values in henries, in order from source to load, of all inductors in the network. The length of the inductance vector must be equal to the length of the vector you provide for `'C'`. All values must be strictly positive. The default is `[0.0144e-7, 0.4395e-7, 0.0144e-7]`.

Examples

```
filter = rfckt.lcbandpasspi;
filter.L = [3.1 5.9 16.3]*1e-9;
```

rfckt.lcbandpasspi.Name property

Purpose Object name

Values `'LC Bandpass Pi'`

Description Read-only string that contains the name of the object.

Examples

```
filter = rfckt.lcbandpasspi;
filter.Name

ans =

            LC Bandpass Pi
```

Purpose

Number of ports

Values

2

Description

A read-only integer that indicates the object has two ports.

Examples

```
filter = rfckt.lcbandpasspi;
filter.nPort

ans =

     2
```

rfckt.lcbandpasstee.AnalyzedResult property

Purpose

Computed S-parameters, noise figure, OIP3, and group delay values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

Examples

```
filter = rfckt.lcbandpasstee;
analyze(filter,[1e9,2e9,3e9]);
filter.AnalyzedResult

ans =

            Name: 'Data object'
            Freq: [3x1 double]
            S_Parameters: [2x2x3 double]
            GroupDelay: [3x1 double]
            NF: [3x1 double]
            OIP3: [3x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Linear'
```

Purpose Capacitance data

Values Vector

Description Capacitance values in farads, in order from source to load, of all capacitors in the network. The length of the capacitance vector must be equal to the length of the vector you provide for `'L'`. All values must be strictly positive. The default is `[0.0186e-10, 0.1716e-10, 0.0186e-10]`.

Examples

```
filter=rfckt.lcbandpasstee;
filter.C = [10.1 4.5 14.2]*1e-12;
```

rfckt.lcbandpasstee.L property

Purpose Inductance data

Values Vector

Description Inductance values in henries, in order from source to load, of all inductors in the network. The length of the inductance vector must be equal to the length of the vector you provide for `'C'`. All values must be strictly positive. The default is `[0.2781e-7, 0.0301e-7, 0.2781e-7]`.

Examples

```
filter = rfckt.lcbandpasstee;
filter.L = [3.1 5.9 16.3]*1e-9;
```

Purpose Object name

Values `'LC Bandpass Tee'`

Description Read-only string that contains the name of the object.

Examples

```
filter = rfckt.lcbandpasstee;
filter.Name

ans =

          LC Bandpass Tee
```

rfckt.lcbandpasstee.nPort property

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
filter = rfckt.lcbandpasstee;
filter.nPort

ans =

     2
```

Purpose

Computed S-parameters, noise figure, OIP3, and group delay values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

Examples

```
filter = rfckt.lcbandstoppi;
analyze(filter,[1e9,2e9,3e9]);
filter.AnalyzedResult

ans =

           Name: 'Data object'
           Freq: [3x1 double]
           S_Parameters: [2x2x3 double]
           GroupDelay: [3x1 double]
           NF: [3x1 double]
           OIP3: [3x1 double]
           Z0: 50
           ZS: 50
           ZL: 50
           IntpType: 'Linear'
```

rfckt.lcbandstoppi.C property

Purpose Capacitance data

Values Vector

Description Capacitance values in farads, in order from source to load, of all capacitors in the network. The length of the capacitance vector must be equal to the length of the vector you provide for `'L'`. All values must be strictly positive. The default is `[0.0184e-10, 0.2287e-10, 0.0184e-10]`.

Examples

```
filter=rfckt.lcbandstoppi;
filter.C = [10.1 4.5 14.2]*1e-12;
```

Purpose Inductance data

Values Vector

Description Inductance values in henries, in order from source to load, of all inductors in the network. The length of the inductance vector must be equal to the length of the vector you provide for `'C'`. All values must be strictly positive. The default is `[0.2809e-7, 0.0226e-7, 0.2809e-7]`.

Examples

```
filter = rfckt.lcbandstoppi;
filter.L = [3.1 5.9 16.3]*1e-9;
```

rfckt.lcbandstoppi.Name property

Purpose Object name

Values `'LC Bandstop Pi'`

Description Read-only string that contains the name of the object.

Examples

```
filter = rfckt.lcbandstoppi;
filter.Name

ans =

           LC Bandstop Pi
```

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
filter = rfckt.lcbandstoppi;
filter.nPort

ans =

       2
```

rfckt.lcbandstoptee.AnalyzedResult property

Purpose

Computed S-parameters, noise figure, OIP3, and group delay values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

Examples

```
filter = rfckt.lcbandstoptee;
analyze(filter,[1e9,2e9,3e9]);
filter.AnalyzedResult

ans =

            Name: 'Data object'
            Freq: [3x1 double]
            S_Parameters: [2x2x3 double]
            GroupDelay: [3x1 double]
            NF: [3x1 double]
            OIP3: [3x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Linear'
```

Purpose

Capacitance data

Values

Vector

Description

Capacitance values in farads, in order from source to load, of all capacitors in the network. The length of the capacitance vector must be equal to the length of the vector you provide for `'L'`. All values must be strictly positive. The default is `[0.1852e-10, 0.0105e-10, 0.1852e-10]`.

Examples

```
filter=rfckt.lcbandstoptee;
filter.C = [10.1 4.5 14.2]*1e-12;
```

rfckt.lcbandstoptee.L property

Purpose Inductance data

Values Vector

Description Inductance values in henries, in order from source to load, of all inductors in the network. The length of the inductance vector must be equal to the length of the vector you provide for `'C'`. All values must be strictly positive. The default is `[0.0279e-7, 0.4932e-7, 0.0279e-7]`.

Examples

```
filter = rfckt.lcbandstoptee;
filter.L = [3.1 5.9 16.3]*1e-9;
```

Purpose Object name

Values `'LC Bandstop Tee'`

Description Read-only string that contains the name of the object.

Examples

```
filter = rfckt.lcbandstoptee;
filter.Name

ans =

          LC Bandstop Tee
```

rfckt.lcbandstoptee.nPort property

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
filter = rfckt.lcbandstoptee;
filter.nPort

ans =

     2
```

Purpose Computed S-parameters, noise figure, OIP3, and group delay values

Values `rfdata.data` object

Description Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

Examples

```
filter = rfckt.lchighpasspi;
analyze(filter,[1e9,2e9,3e9]);
filter.AnalyzedResult

ans =

            Name: 'Data object'
            Freq: [3x1 double]
            S_Parameters: [2x2x3 double]
            GroupDelay: [3x1 double]
            NF: [3x1 double]
            OIP3: [3x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Linear'
```

rfckt.lchighpasspi.C property

Purpose Capacitance data

Values Vector

Description Capacitance values in farads, in order from source to load, of all capacitors in the network. The length of the capacitance vector must be equal to the length of the vector you provide for `'L'`. All values must be strictly positive. The default is `[0.1188e-5, 0.1188e-5]`.

Examples

```
filter=rfckt.lchighpasspi;
filter.C = [10.1 4.5 14.2]*1e-12;
```

Purpose Inductance data

Values Vector

Description Inductance values in henries, in order from source to load, of all inductors in the network. The length of the inductance vector must be equal to the length of the vector you provide for 'C'. All values must be strictly positive. The default is [2.2363e-9].

Examples

```
filter = rfckt.lchighpasspi;
filter.L = [3.1 5.9 16.3]*1e-9;
```

rfckt.lchighpasspi.Name property

Purpose Object name

Values `'LC Highpass Pi'`

Description Read-only string that contains the name of the object.

Examples

```
filter = rfckt.lchighpasspi;
filter.Name

ans =

            LC Highpass Pi
```

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
filter = rfckt.lchighpasspi;
filter.nPort

ans =

     2
```

rfckt.lchighpasstee.AnalyzedResult property

Purpose Computed S-parameters, noise figure, OIP3, and group delay values

Values `rfdata.data` object

Description Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

Examples

```
filter = rfckt.lchighpasstee;
analyze(filter,[1e9,2e9,3e9]);
filter.AnalyzedResult

ans =

            Name: 'Data object'
            Freq: [3x1 double]
            S_Parameters: [2x2x3 double]
            GroupDelay: [3x1 double]
            NF: [3x1 double]
            OIP3: [3x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Linear'
```

Purpose Capacitance data

Values Vector

Description Capacitances values in farads, in order from source to load, of all capacitors in the network. The length of the capacitance vector must be equal to the length of the vector you provide for `'L'`. All values must be strictly positive. The default is `[0.4752e-9, 0.4752e-9]`.

Examples

```
filter=rfckt.lchighpasstee;
filter.C = [10.1 4.5 14.2]*1e-12;
```

rfckt.lchighpasstee.L property

Purpose Inductance data

Values Vector

Description Inductance values in henries, in order from source to load, of all inductors in the network. The length of the inductance vector must be equal to the length of the vector you provide for `'C'`. All values must be strictly positive. The default is `[5.5907e-6]`.

Examples

```
filter = rfckt.lchighpasstee;
filter.L = [3.1 5.9 16.3]*1e-9;
```

Purpose Object name

Values `'LC Highpass Tee'`

Description Read-only string that contains the name of the object.

Examples

```
filter = rfckt.lchighpasstee;
filter.Name

ans =

          LC Highpass Tee
```

rfckt.lchighpasstee.nPort property

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
filter = rfckt.lchighpasstee;
filter.nPort

ans =

     2
```

Purpose Computed S-parameters, noise figure, OIP3, and group delay values

Values `rfdata.data` object

Description Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

Examples

```
filter = rfckt.lclowpasspi;
analyze(filter,[1e9,2e9,3e9]);
filter.AnalyzedResult

ans =

            Name: 'Data object'
            Freq: [3x1 double]
            S_Parameters: [2x2x3 double]
            GroupDelay: [3x1 double]
            NF: [3x1 double]
            OIP3: [3x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Linear'
```

rfckt.lclowpasspi.C property

Purpose

Capacitance data

Values

Vector

Description

Capacitance values in farads, in order from source to load, of all capacitors in the network. The length of the capacitance vector must be equal to or one greater than the length of the vector you provide for `'L'`. All values must be strictly positive. The default is `[0.5330e-8, 0.5330e-8]`.

Examples

```
filter=rfckt.lclowpasspi;
filter.C = [10.1 4.5 14.2]*1e-12;
```

Purpose

Inductance data

Values

Vector

Description

Inductance values in henries, in order from source to load, of all inductors in the network. The length of the inductance vector must be equal to or one less than the length of the vector you provide for `'C'`. All values must be strictly positive. The default is `[2.8318e-6]`.

Examples

```
filter = rfckt.lclowpasspi;
filter.L = [3.1 5.9 16.3]*1e-9;
```

rfckt.lclowpasspi.Name property

Purpose Object name

Values `'LC Lowpass Pi'`

Description Read-only string that contains the name of the object.

Examples

```
filter = rfckt.lclowpasspi;
filter.Name

ans =

          LC Lowpass Pi
```

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
filter = rfckt.lclowpasspi;
filter.nPort

ans =

     2
```

rfckt.lclowpasstee.AnalyzedResult property

Purpose

Computed S-parameters, noise figure, OIP3, and group delay values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

Examples

```
filter = rfckt.lclowpasstee;
analyze(filter,[1e9,2e9,3e9]);
filter.AnalyzedResult

ans =

          Name: 'Data object'
          Freq: [3x1 double]
          S_Parameters: [2x2x3 double]
          GroupDelay: [3x1 double]
          NF: [3x1 double]
          OIP3: [3x1 double]
          Z0: 50
          ZS: 50
          ZL: 50
          IntpType: 'Linear'
```

Purpose Capacitance data

Values Vector

Description Capacitance values in farads, in order from source to load, of all capacitors in the network. The length of the capacitance vector must be equal to or one less than the length of the vector you provide for `'L'`. All values must be strictly positive. The default is `[1.1327e-9]`.

Examples

```
filter=rfckt.lclowpasstee;
filter.C = [10.1 4.5 14.2]*1e-12;
```

rfckt.lclowpasstee.L property

Purpose Inductance data

Values Vector

Description Inductance values in henries, in order from source to load, of all inductors in the network. The length of the inductance vector must be equal to or one greater than the length of the vector you provide for `'C'`. All values must be strictly positive. The default is `[0.1332e-4, 0.1332e-4]`.

Examples

```
filter = rfckt.lclowpasstee;
filter.L = [3.1 5.9 16.3]*1e-9;
```

Purpose Object name

Values `'LC Lowpass Tee'`

Description Read-only string that contains the name of the object.

Examples

```
filter = rfckt.lclowpasstee;
filter.Name

ans =

          LC Lowpass Tee
```

rfckt.lclowpasstee.nPort property

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
filter = rfckt.lclowpasstee;
filter.nPort

ans =

      2
```

Purpose

Computed S-parameters, noise figure, OIP3, and group delay values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

The `analyze` method treats the microstrip line as a 2-port linear network and models the line as a transmission line with optional stubs. The `analyze` method computes the `AnalyzedResult` property of the transmission line using the data stored in the `rfckt.microstrip` object properties as follows:

- If you model the transmission line as a stubless line, the `analyze` method first calculates the ABCD-parameters at each frequency contained in the modeling frequencies vector. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

  The `analyze` method calculates the ABCD-parameters using the physical length of the transmission line, d, and the complex propagation constant, k, using the following equations:

$$A = \frac{e^{kd} + e^{-kd}}{2}$$

$$B = \frac{Z_0 * \left(e^{kd} - e^{-kd}\right)}{2}$$

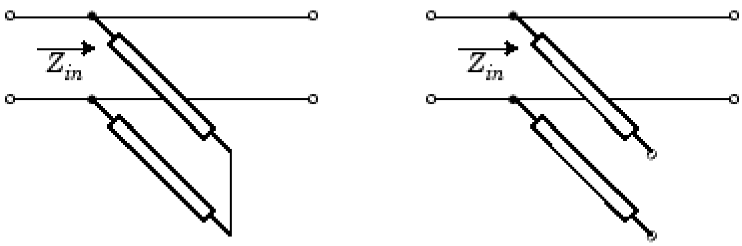
$$C = \frac{e^{kd} - e^{-kd}}{2 * Z_0}$$

$$D = \frac{e^{kd} + e^{-kd}}{2}$$

Z_0 and k are vectors whose elements correspond to the elements of f, the vector of frequencies specified in the `analyze` input argument `freq`. Both can be expressed in terms of the specified conductor strip width, substrate height, conductor strip thickness, relative permittivity constant, conductivity, and dielectric loss tangent of the microstrip line, as described in [1].

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

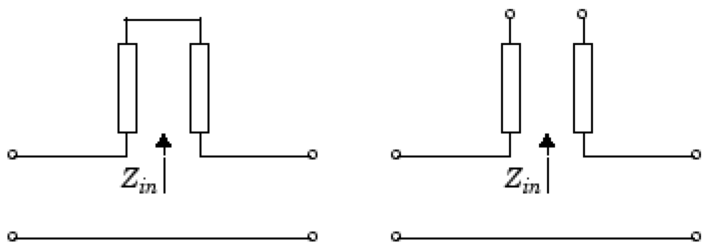
  When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown in the following figure.

  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 in the following figure.



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

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

The `analyze` method uses the S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

Examples

```
tx1 = rfckt.microstrip;
analyze(tx1,[1e9,2e9,3e9]);
tx1.AnalyzedResult

ans =

             Name: 'Data object'
             Freq: [3x1 double]
             S_Parameters: [2x2x3 double]
             GroupDelay: [3x1 double]
             NF: [3x1 double]
             OIP3: [3x1 double]
             Z0: 50
             ZS: 50
             ZL: 50
             IntpType: 'Linear'
```

rfckt.microstrip.EpsilonR property

Purpose

Relative permittivity of dielectric

Values

Scalar

Description

The ratio of the permittivity of the dielectric, ε, to the permittivity of free space, ε_0. The default value is `9.8`.

Examples

Change the relative permittivity of the dielectric:

```
tx1=rfckt.microstrip;
tx1.EpsilonR=2.7;
```

Purpose

Dielectric thickness

Values

Scalar

Description

Physical height, in meters, of the dielectric on which the microstrip resides. The default is `6.35e-4`.

Examples

```
tx1=rfckt.microstrip;
tx1.Height=0.001;
```

rfckt.microstrip.LineLength property

Purpose Microstrip line length

Values Scalar

Description The physical length of the transmission line in meters. The default is `0.01`.

Examples

```
tx1 = rfckt.microstrip;
tx1.LineLength = 0.001;
```

Purpose Tangent of loss angle

Values Scalar

Description The loss angle tangent of the dielectric. The default is `0`.

Examples

```
tx1 = rfckt.microstrip;
tx1.LossTangent

ans =

     0
```

rfckt.microstrip.Name property

Purpose Object name

Values `'Micrstrip Waveguide Transmission Line'`

Description Read-only string that contains the name of the object.

Examples

```
tx1 = rfckt.microstrip;
tx1.Name

ans =

           Microstrip Transmission Line
```

Purpose Conductor conductivity

Values Scalar

Description Conductivity, in Siemens per meter (S/m), of the conductor. The default is `Inf`.

Examples

```
tx1=rfckt.microstrip;
tx1.SigmaCond=5.81e7;
```

rfckt.microstrip.StubMode property

Purpose Type of stub

Values `'NotAStub'` (default), `'Series'`, or `'Shunt'`

Description String that specifies what type of stub, if any, to include in the transmission line model.

Examples

```
tx1 = rfckt.microstrip;
tx1.StubMode = 'Series';
```

Purpose

Stub transmission line termination

Values

`'NotApplicable'` (default), `'Open'`, or `'Short'`.

Description

String that specifies what type of termination to use for `'Shunt'`and `'Series'` stub modes. `Termination` is ignored if the line has no stub. Use `'NotApplicable` when `StubMode` is `'NotAStub'`.

Examples

```
tx1 = rfckt.microstrip;
tx1.StubMode = 'Series';
tx1.Termination = 'Short';
```

rfckt.microstrip.Thickness property

Purpose Microstrip thickness

Values Scalar

Description Physical thickness, in meters, of the microstrip. The default is `5.0e-6`.

Examples

```
tx1=rfckt.microstrip;
tx1.Thickness=2e-6;
```

Purpose Parallel-plate width

Values Scalar

Description Physical width, in meters, of the parallel-plate. The default is `6.0e-4`.

Examples

```
tx1=rfckt.microstrip;
tx1.Thickness=2e-4;
```

rfckt.microstrip.nPort property

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
tx1 = rfckt.microstrip;
tx1.nPort

ans =

     2
```

Purpose

Computed S-parameters, noise figure, OIP3, and group delay values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. The default is a 1-by-1 `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values that result from analyzing the values stored in the `default.amp` file at the frequencies stored in this file.

The `analyze` method computes the `AnalyzedResult` property using the data stored in the `rfckt.mixer` object properties as follows:

- The `analyze` method uses the data stored in the `'NoiseData'` property of the `rfckt.mixer` object to calculate the noise figure.
- The `analyze` method uses the data stored in the `'PhaseNoiseLevel'` property of the `rfckt.mixer` object to calculate phase noise. The `analyze` method first generates additive white Gaussian noise (AWGN) and filters the noise with a digital FIR filter. It then adds the resulting noise to the angle component of the input signal.

  The method computes the digital filter by:

  1. Interpolating the specified phase noise amplitude to determine the phase noise values at the modeling frequencies.
  2. Taking the IFFT of the resulting phase noise spectrum to get the coefficients of the FIR filter.
- The `analyze` method uses the data stored in the `'NonlinearData'` property of the `rfckt.mixer` object to calculate OIP3.

  If power data exists in the `'NonlinearData'` property, the block extracts the AM/AM and AM/PM nonlinearities from the power data.

  If the `'NonlinearData'` property contains only IP3 data, the method computes and adds the nonlinearity by:

1. Using the third-order input intercept point value in dBm to compute the factor, f, that scales the input signal before the mixer object applies the nonlinearity:

   $$F_{AM/AM}(u) = u - \frac{u^3}{3}$$

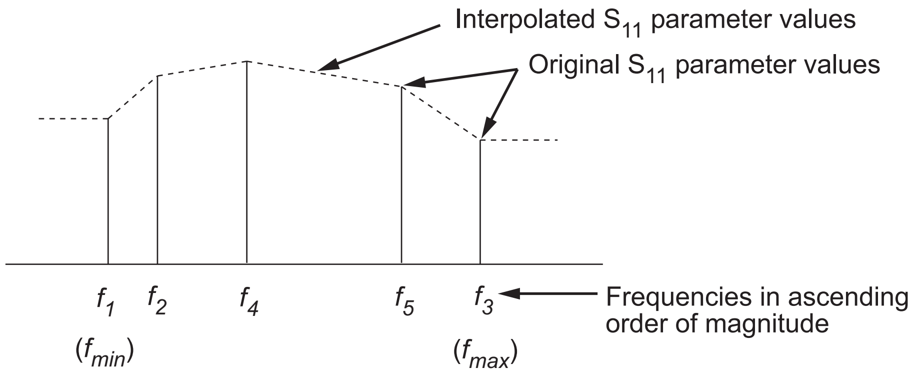
2. Computing the scaled input signal by multiplying the mixer input signal by f.
3. Limiting the scaled input signal to a maximum value of 1.
4. Applying an AM/AM conversion to the mixer gain, according to the following cubic polynomial equation:

   $$F_{AM/AM}(u) = u - \frac{u^3}{3}$$

   where u is the magnitude of the scaled input signal, which is a unitless normalized input voltage.

- The `analyze` method uses the data stored in the `'NetworkData'` property of the `rfckt.mixer` object to calculate the group delay values of the mixer at the frequencies specified in `freq`, as described in the `analyze` reference page.
- The `analyze` method uses the data stored in the `'NetworkData'` property of the `rfckt.mixer` object to calculate the S-parameter values of the mixer at the frequencies specified in `freq`. If the `'NetworkData'` property contains network Y- or Z-parameters, the `analyze` method first converts the parameters to S-parameters. Using the interpolation method you specify with the `'IntpType'` property, the `analyze` method interpolates the S-parameter values to determine their values at the specified frequencies.

  Specifically, the `analyze` method 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 S_{11} parameters at five different frequencies.



For more information, see "One-Dimensional Interpolation" and the `interp1` reference page in the MATLAB documentation.

As shown in the preceding diagram, the `analyze` method 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, so you need to specify network parameter values over a range of frequencies that is wide enough to account for the mixer behavior.

RF Toolbox software computes the reflected wave at the mixer input (b_1) and at the mixer output (b_2) from the interpolated S-parameters as

$$\begin{bmatrix} b_1(f_{in}) \\ b_2(f_{out}) \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} a_1(f_{in}) \\ a_2(f_{out}) \end{bmatrix}$$

where

- f_{in} and f_{out} are the mixer input and output frequencies, respectively.
- a^1 and a_2 are the incident waves at the mixer input and output, respectively.

The interpolated S_{21} parameter values describe the conversion gain as a function of frequency, referred to the mixer input frequency.

Examples

```
mix1 = rfckt.mixer;
mix1.AnalyzedResult

ans =

            Name: 'Data object'
            Freq: [191x1 double]
            S_Parameters: [2x2x191 double]
            GroupDelay: [191x1 double]
            NF: [191x1 double]
            OIP3: [191x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Linear'
```

Purpose Local oscillator frequency

Values Scalar

Description Frequency, in hertz, of the local oscillator. The default is `1.0e+9`.

If the `MixerType` property is set to `'Downconverter'`, the mixer output frequency is calculated as $f_{out} = f_{in} - f_{lo}$. If the `MixerType` property is set to `'Upconverter'`, the mixer output frequency is calculated as $f_{out} = f_{in} + f_{lo}$.

Examples

```
mix1 = rfckt.mixer;
mix1.FLO = 1.6e9;
```

rfckt.mixer.FreqOffset property

Purpose Frequency offset data

Values Vector

Description Vector specifying the frequency offset values, in hertz, that correspond to the phase noise level values specified by the `PhaseNoiseLevel` property. This property is empty by default.

Examples

```
mix1 = rfckt.mixer;
mix1.FreqOffset = [1.6e6, 2.1e6];
```

rfckt.mixer.IntpType property

Purpose

Interpolation method

Values

'Linear' (default), 'Spline', or 'Cubic'

Description

The analyze method is flexible in that it does not require the frequencies of the specified S-parameters to match the requested analysis frequencies. If needed, analyze applies the interpolation and extrapolation method specified in the IntpType property to the specified data to create a new set of data at the requested analysis frequencies. The following table lists the available interpolation methods and describes each one.

Method	Description
Linear (default)	Linear interpolation
Spline	Cubic spline interpolation
Cubic	Piecewise cubic Hermite interpolation

Examples

```
mix1 = rfckt.mixer;
mix1.IntpType = 'cubic'

mix1 =

            Name: 'Mixer'
            nPort: 2
            AnalyzedResult: [1x1 rfdata.data]
            IntpType: 'Cubic'
            NetworkData: [1x1 rfdata.network]
            NoiseData: [1x1 rfdata.noise]
            NonlinearData: Inf
            MixerType: 'downconverter'
            FLO: 1.0000e+009
            FreqOffset: []
            PhaseNoiseLevel: []
```

rfckt.mixer.MixerSpurData property

Purpose Data from mixer spur table

Values `rfdata.mixerspur` object

Description An `rfdata.mixerspur` object that stores data from an intermodulation table. This property is empty by default.

Examples

```
mix1 = rfckt.mixer;
mix1.MixerSpurData=rfdata.mixerspur('Data',[2 5; 1 0],...
                                    'PinRef',3,'PLORef',5)
mix1 =

               Name: 'Mixer'
              nPort: 2
     AnalyzedResult: [1x1 rfdata.data]
           IntpType: 'Linear'
        NetworkData: [1x1 rfdata.network]
          NoiseData: [1x1 rfdata.noise]
      NonlinearData: Inf
      MixerSpurData: [1x1 rfdata.mixerspur]
          MixerType: 'Downconverter'
                FLO: 1.0000e+009
         FreqOffset: []
    PhaseNoiseLevel: []
```

Purpose Type of mixer

Values 'Downconverter' (default) or 'Upconverter'

Description String specifying whether the mixer downconverting or upconverting.

Examples

```
mix1 = rfckt.mixer;
mix1.MixerType = 'Upconverter';
```

rfckt.mixer.Name property

Purpose Object name

Values `'Mixer'`

Description Read-only string that contains the name of the object.

Examples

```
mix1 = rfckt.mixer;
mix1.Name

ans =

          Mixer
```

Purpose Network parameter information

Values `rfdata.network` object

Description An `rfdata.network` object that stores network parameter data. The default network parameter values are taken from the `'default.s2p'` data file.

Examples

```
mix1 = rfckt.mixer;
mix1.NetworkData

ans =

    Name: 'Network parameters'
    Type: 'S_PARAMETERS'
    Freq: [191x1 double]
    Data: [2x2x191 double]
    Z0: 50
```

rfckt.mixer.NoiseData property

Purpose

Noise information

Values

Scalar noise figure in decibels, `rfdata.noise` object or `rfdata.nf` object

Description

A scalar value or object that stores noise data. The default is an `rfdata.noise` object whose values are taken from the `'default.s2p'` data file.

Examples

```
mix1 = rfckt.mixer;
mix1.NoiseData

ans =

        Name: 'Spot noise data'
        Freq: [9x1 double]
        FMIN: [9x1 double]
        GAMMAOPT: [9x1 double]
        RN: [9x1 double]
```

Purpose

Nonlinearity information

Values

Scalar OIP3 in decibels relative to one milliwatt, `rfdata.power` object or `rfdata.ip3` object

Description

A scalar value or object that stores nonlinearity data. The default is an `Inf`.

Examples

```
mix1 = rfckt.mixer;
mix1.NonlinearData

ans =

      Inf
```

rfckt.mixer.PhaseNoiseLevel property

Purpose Phase noise data

Values Vector

Description Vector specifying the phase noise levels, in dBc/Hz, that correspond to the frequency offset values specified by the `FreqOffset` property. This property is empty by default.

Examples

```
mix1 = rfckt.mixer;
mix1.PhaseNoiseLevel = [-75, -110];
```

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
mix1 = rfckt.mixer;
mix1.nPort

ans =

     2
```

rfckt.parallel.AnalyzedResult property

Purpose

Computed S-parameters, noise figure, OIP3, and group delay values

Values

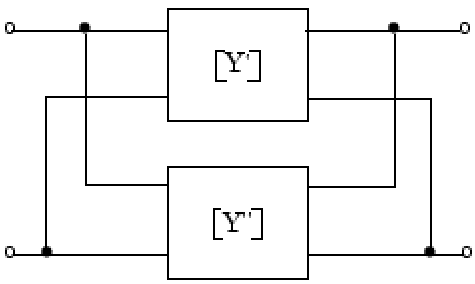
`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

The `analyze` method computes the S-parameters of the `AnalyzedResult` property using the data stored in the `Ckts` property as follows:

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



where

$$[Y'] = \begin{bmatrix} Y_{11}' & Y_{12}' \\ Y_{21}' & Y_{22}' \end{bmatrix}$$

$$[Y''] = \begin{bmatrix} Y_{11}'' & Y_{12}'' \\ Y_{21}'' & Y_{22}'' \end{bmatrix}$$

2 The `analyze` method 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.

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

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

The `analyze` method uses the parallel S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

Examples

```
tx1 = rfckt.txline;
tx2 = rfckt.txline;
plel = rfckt.parallel('Ckts',{tx1,tx2})
analyze(plel,[1e9:1e7:2e9]);
plel.AnalyzedResult

ans =

            Name: 'Data object'
            Freq: [101x1 double]
    S_Parameters: [2x2x101 double]
      GroupDelay: [101x1 double]
              NF: [101x1 double]
            OIP3: [101x1 double]
              Z0: 50
              ZS: 50
              ZL: 50
        IntpType: 'Linear'
```

rfckt.parallel.Ckts property

Purpose Circuit objects in network

Values Cell

Description Cell array containing handles to all circuit objects in the network. All circuits must be 2-port and linear. This property is empty by default.

Examples

```
tx1 = rfckt.txline;
tx2 = rfckt.txline;
plel = rfckt.parallel;
plel.Ckts = {tx1,tx2};
plel.Ckts

ans =

    [1x1 rfckt.txline] [1x1 rfckt.txline]
```

Purpose
Object name

Values
`'Parallel Connected Network'`

Description
Read-only string that contains the name of the object.

Examples

```
plel = rfckt.parallel;
plel.Name

ans =

          Parallel Connected Network
```

rfckt.parallel.nPort property

Purpose
Number of ports

Values
2

Description
A read-only integer that indicates the object has two ports.

Examples

```
plel = rfckt.parallel;
plel.nPort

ans =

     2
```

rfckt.parallelplate.AnalyzedResult property

Purpose

Computed S-parameters, noise figure, OIP_3, and group delay values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP_3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

The `analyze` method treats the parallel-plate line as a 2-port linear network and models the line as a transmission line with optional stubs. The `analyze` method computes the `AnalyzedResult` property of the line using the data stored in the `rfckt.parallelplate` object properties as follows:

- If you model the transmission line as a stubless line, the `analyze` method first calculates the ABCD-parameters at each frequency contained in the modeling frequencies vector. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

  The `analyze` method calculates the ABCD-parameters using the physical length of the transmission line, d, and the complex propagation constant, k, using the following equations:

  $$A = \frac{e^{kd} + e^{-kd}}{2}$$
  $$B = \frac{Z_0 * \left(e^{kd} - e^{-kd}\right)}{2}$$
  $$C = \frac{e^{kd} - e^{-kd}}{2 * Z_0}$$
  $$D = \frac{e^{kd} + e^{-kd}}{2}$$

  Z_0 and k are vectors whose elements correspond to the elements of f, the vector of frequencies specified in the `analyze` input argument `freq`. Both can be expressed in terms of the resistance (R), inductance (L), conductance (G), and capacitance (C) per unit length (meters) as follows:

$$Z_0 = \sqrt{\frac{R + j2\pi fL}{G + j2\pi fC}}$$

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

where

$$R = \frac{2}{w\sigma_{cond}\delta_{cond}}$$

$$L = \mu \frac{d}{w}$$

$$G = \omega\varepsilon'' \frac{w}{d}$$

$$C = \varepsilon \frac{w}{d}$$

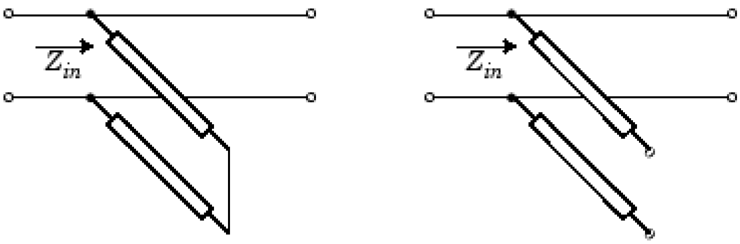
In these equations:

- w is the plate width.
- d is the plate separation.
- σ_{cond} is the conductivity in the conductor.
- μ is the permeability of the dielectric.
- ε is the permittivity of the dielectric.
- ε'' is the imaginary part of ε, $\varepsilon'' = \varepsilon_0\varepsilon_r \tan\delta$, where:
  - ε_0 is the permittivity of free space.
  - ε_r is the `EpsilonR` property value.
  - $\tan\delta$ is the `LossTangent` property value.

- δ_{cond} is the skin depth of the conductor, which the block calculates as $1/\sqrt{\pi f \mu \sigma_{cond}}$.
- f is a vector of modeling frequencies determined by the Outport block.

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

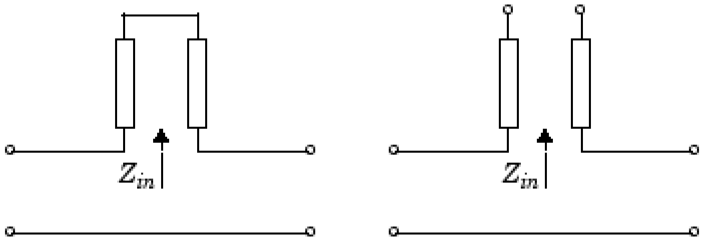
  When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown in the following figure.

  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 in the following figure.



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

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

The `analyze` method uses the S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

Examples

```
tx1 = rfckt.parallelplate;
analyze(tx1,[1e9,2e9,3e9]);
tx1.AnalyzedResult

ans =

              Name: 'Data object'
              Freq: [3x1 double]
              S_Parameters: [2x2x3 double]
              GroupDelay: [3x1 double]
              NF: [3x1 double]
              OIP3: [3x1 double]
              Z0: 50
              ZS: 50
              ZL: 50
              IntpType: 'Linear'
```

Purpose Relative permittivity of dielectric

Values Scalar

Description The ratio of the permittivity of the dielectric, ε, to the permittivity of free space, ε_0. The default value is `2.3`.

Examples

```
tx1=rfckt.parallelplate;
tx1.EpsilonR=2.7;
```

rfckt.parallelplate.LineLength property

Purpose

Parallel-plate line length

Values

Scalar

Description

The physical length of the parallel-plate transmission line in meters. The default is `0.01`.

Examples

```
tx1 = rfckt.parallelplate;
tx1.LineLength = 0.001;
```

Purpose Tangent of loss angle

Values Scalar

Description The loss angle tangent of the dielectric. The default is `0`.

Examples

```
tx1=rfckt.parallelplate;
tx1.LossTangent=0.002;
```

rfckt.parallelplate.MuR property

Purpose Relative permeability of dielectric

Values Scalar

Description The ratio of the permeability of the dielectric, μ, to the permeability of free space, μ_0. The default value is 1.

Examples Change the relative permeability of the dielectric:

```
tx1=rfckt.parallelplate;
tx1.MuR=0.8;
```

Purpose Object name

Values `'Parallel-Plate Transmission Line'`

Description Read-only string that contains the name of the object.

Examples

```
tx1 = rfckt.parallelplate;
tx1.Name

ans =

          Parallel-Plate Transmission Line
```

rfckt.parallelplate.Separation property

Purpose

Distance between plates

Values

Scalar

Description

Thickness, in meters, of the dielectric separating the plates. The default is `1.0e-3`..

Examples

```
tx1=rfckt.parallelplate;
tx1.Separation=0.8e-3;
```

Purpose Conductor conductivity

Values Scalar

Description Conductivity, in Siemens per meter (S/m), of the conductor. The default is `Inf`.

Examples

```
tx1=rfckt.parallelplate;
tx1.SigmaCond=5.81e7;
```

rfckt.parallelplate.StubMode property

Purpose Type of stub

Values `'NotAStub'` (default), `'Series'`, or `'Shunt'`

Description String that specifies what type of stub, if any, to include in the transmission line model.

Examples

```
tx1 = rfckt.parallelplate;
tx1.StubMode = 'Series';
```

Purpose

Stub transmission line termination

Values

'NotApplicable' (default), 'Open', or 'Short'.

Description

String that specifies what type of termination to use for 'Shunt'and 'Series' stub modes. Termination is ignored if the line has no stub. Use 'NotApplicable' when StubMode is 'NotAStub'.

Examples

```
tx1 = rfckt.parallelplate;
tx1.StubMode = 'Series';
tx1.Termination = 'Short';
```

rfckt.parallelplate.Width property

Purpose
Transmission line width

Values
Scalar

Description
Physical width, in meters, of the parallel-plate transmission line. The default is `.005`..

Examples

```
tx1=rfckt.parallelplate;
tx1.Width=0.001;
```

Purpose

Number of ports

Values

2

Description

A read-only integer that indicates the object has two ports.

Examples

```
tx1 = rfckt.parallelplate;
tx1.nPort

ans =

      2
```

rfckt.passive.AnalyzedResult property

Purpose

Computed S-parameters, noise figure, OIP3, and group delay values

Values

`rfdata.data` object

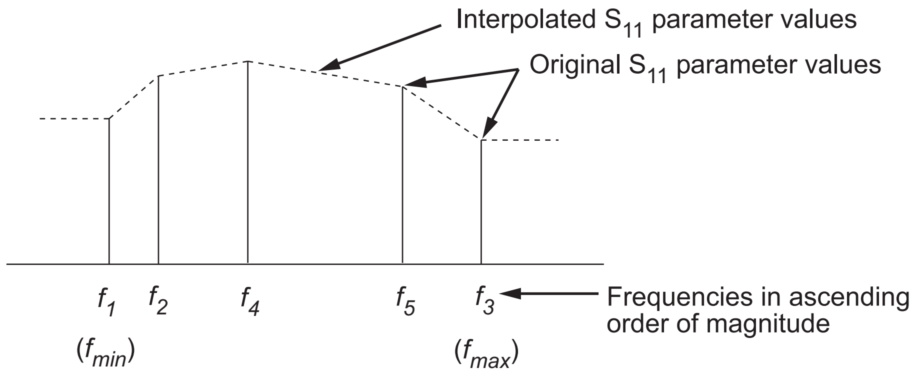
Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. The default is a 1-by-1 `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values that result from analyzing the values stored in the `passive.s2p` file at the frequencies stored in this file.

The `analyze` method computes the `AnalyzedResult` property as follows:

The `analyze` method uses the data stored in the `'NetworkData'` property of the `rfckt.passive` object to calculate the S-parameter values of the passive component at the frequencies specified in `freq`. If the `'NetworkData'` property contains network Y- or Z-parameters, the `analyze` method first converts the parameters to S-parameters. Using the interpolation method you specify with the `'IntpType'` property, the `analyze` method interpolates the S-parameter values to determine their values at the specified frequencies.

Specifically, the `analyze` method 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 S_{11} parameters at five different frequencies.



For more information, see "One-Dimensional Interpolation" and the `interp1` reference page in the MATLAB documentation.

As shown in the preceding diagram, the `analyze` method 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, so you need to specify network parameter values over a range of frequencies that is wide enough to account for the component behavior.

The `analyze` method uses the S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

Examples

```
pas = rfckt.passive;
pas.AnalyzedResult

ans =

          Name: 'Data object'
          Freq: [202x1 double]
          S_Parameters: [2x2x202 double]
          GroupDelay: [202x1 double]
          NF: [202x1 double]
          OIP3: [202x1 double]
```

```
ZO: 50
ZS: 50
ZL: 50
IntpType: 'Linear'
```

Purpose Interpolation method

Values `'Linear'` (default), `'Spline'`, or `'Cubic'`

Description The `analyze` method is flexible in that it does not require the frequencies of the specified S-parameters to match the requested analysis frequencies. If needed, `analyze` applies the interpolation and extrapolation method specified in the `IntpType` property to the specified data to create a new set of data at the requested analysis frequencies. The following table lists the available interpolation methods and describes each one.

Method	Description
`Linear` (default)	Linear interpolation
`Spline`	Cubic spline interpolation
`Cubic`	Piecewise cubic Hermite interpolation

Examples

```
pas = rfckt.passive;
pas.IntpType = 'cubic'

pas =

             Name: 'Passive'
             nPort: 2
             AnalyzedResult: [1x1 rfdata.data]
             IntpType: 'Cubic'
             NetworkData: [1x1 rfdata.network]
```

rfckt.passive.Name property

Purpose Object name

Values `'Passive'`

Description Read-only string that contains the name of the object.

Examples

```
pas = rfckt.passive;
pas.Name

ans =

           Passive
```

Purpose Network parameter information

Values `rfdata.network` object

Description An `rfdata.network` object that stores network parameter data. The default network parameter values are taken from the `'passive.s2p'` data file.

Examples

```
pas = rfckt.passive;
pas.NetworkData

ans =

    Name: 'Network parameters'
    Type: 'S_PARAMETERS'
    Freq: [202x1 double]
    Data: [2x2x202 double]
    Z0: 50
```

rfckt.passive.nPort property

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
pas = rfckt.passive;
pas.nPort

ans =

     2
```

Purpose

Computed S-parameters, noise figure, OIP3, and group delay values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

The `analyze` method treats the transmission line, which can be lossy or lossless, as a 2-port linear network. It uses the interpolation method you specify in the `IntpType` property to find the R, L, C, and G values at the frequencies you specify when you call `analyze`. Then, it calculates the characteristic impedance, Z0, phase velocity, PV, and loss using these interpolated values. It computes the `AnalyzedResult` property of a stub or as a stubless line using the data stored in the `rfckt.rlcgline` object properties as follows:

- If you model the transmission line as a stubless line, the `analyze` method first calculates the ABCD-parameters at each frequency contained in the modeling frequencies vector. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

  The `analyze` method calculates the ABCD-parameters using the physical length of the transmission line, d, and the complex propagation constant, k, using the following equations:

$$A = \frac{e^{kd} + e^{-kd}}{2}$$

$$B = \frac{Z_0 * \left(e^{kd} - e^{-kd}\right)}{2}$$

$$C = \frac{e^{kd} - e^{-kd}}{2 * Z_0}$$

$$D = \frac{e^{kd} + e^{-kd}}{2}$$

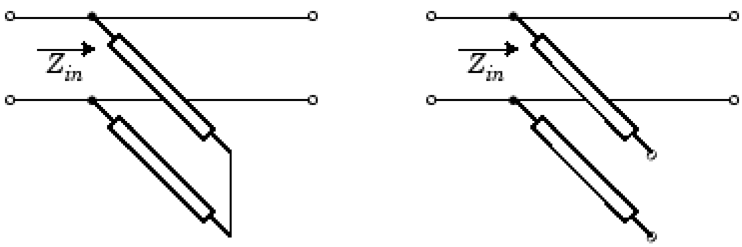
Z_0 and k are vectors whose elements correspond to the elements of f, the vector of frequencies specified in the `analyze` input argument `freq`. Both can be expressed in terms of the resistance (R), inductance (L), conductance (G), and capacitance (C) per unit length (meters) as follows:

$$Z_0 = \sqrt{\frac{R + j2\pi fL}{G + j2\pi fC}}$$

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

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

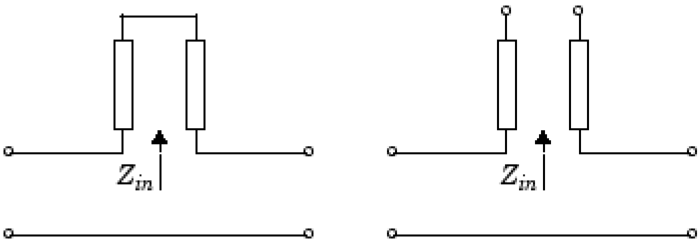
  When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown in the following figure.



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 in the following figure.



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

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

The `analyze` method uses the S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

Examples

```
tx1 = rfckt.rlcgline;
analyze(tx1,[1e9,2e9,3e9]);
tx1.AnalyzedResult

ans =

            Name: 'Data object'
            Freq: [3x1 double]
            S_Parameters: [2x2x3 double]
            GroupDelay: [3x1 double]
            F: [3x1 double]
            OIP3: [3x1 double]
            Z0: 50
```

```
ZS: 50
ZL: 50
IntpType: 'Linear'
```

Purpose Capacitance data

Values Vector

Description Capacitance values per length, in farads per meter, that correspond to the frequencies stored in the `Freq` property. All values must be nonnegative. The default is `0`.

Examples

```
tx1=rfckt.rlcgline;
tx1.C = [10.1 4.5 14.2]*1e-12;
```

rfckt.rlcgline.Freq property

Purpose Frequency data

Values Vector

Description M-element vector of frequency values in hertz for the RLCG values. The values must be positive, and the order of the frequencies must correspond to the order of the RLCG values. The default is `1e9`.

Examples

```
f = [2.08 2.10]*1.0e9;
tx1 = rfckt.rlcgline;
tx1.Freq = f;
```

Purpose

Conductance data

Values

Vector

Description

Conductances per length, in Siemens per meter, that correspond to the frequencies stored in the `Freq` property. All values must be nonnegative. The default is `0`.

Examples

```
tx1=rfckt.rlcgline;
tx1.G = [10.1 4.5 14.2]*1e-3;
```

rfckt.rlcgline.IntpType property

Purpose

Interpolation method

Values

`'Linear'` (default), `'Spline'`, or `'Cubic'`

Description

The `analyze` method is flexible in that it does not require the frequencies of the specified S-parameters to match the requested analysis frequencies. If needed, `analyze` applies the interpolation and extrapolation method specified in the `IntpType` property to the specified data to create a new set of data at the requested analysis frequencies. The following table lists the available interpolation methods and describes each one.

Method	Description
`Linear` (default)	Linear interpolation
`Spline`	Cubic spline interpolation
`Cubic`	Piecewise cubic Hermite interpolation

Examples

```
tx1 = rfckt.rlcgline;
tx1.IntpType = 'cubic';
```

Purpose Inductance data

Values Vector

Description Inductance values per length, in henries per meter, that correspond to the frequencies stored in the `Freq` property. All values must be nonnegative. The default is `0`.

Examples

```
filter = rfckt.rlcgline;
filter.L = [3.1 5.9 16.3]*1e-9;
```

rfckt.rlcgline.LineLength property

Purpose Transmission line length

Values Scalar

Description The physical length of the transmission line in meters. The default is 0.01.

Examples

```
tx1 = rfckt.rlcgline;
tx1.LineLength = 0.001;
```

Purpose Object name

Values `'RLCG Transmission Line'`

Description Read-only string that contains the name of the object.

Examples

```
tx1 = rfckt.rlcgline;
tx1.Name

ans =

            RLCG Transmission Line
```

rfckt.rlcgline.R property

Purpose Resistance data

Values Vector

Description Resistance per length, in ohms per meter, that correspond to the frequencies stored in the `Freq` property. All values must be nonnegative. The default is `0`.

Examples

```
filter = rfckt.rlcgline;
filter.R = [3.1 5.9 16.3]*1e-3;
```

Purpose

Type of stub

Values

`'NotAStub'` (default), `'Series'`, or `'Shunt'`

Description

String that specifies what type of stub, if any, to include in the transmission line model.

Examples

```
tx1 = rfckt.rlcgline;
tx1.StubMode = 'Series';
```

rfckt.rlcgline.Termination property

Purpose Stub transmission line termination

Values `'NotApplicable'` (default), `'Open'`, or `'Short'`.

Description String that specifies what type of termination to use for `'Shunt'` and `'Series'` stub modes. `Termination` is ignored if the line has no stub. Use `'NotApplicable'` when `StubMode` is `'NotAStub'`.

Examples

```
tx1 = rfckt.rlcgline;
tx1.StubMode = 'Series';
tx1.Termination = 'Short';
```

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
tx1 = rfckt.rlcgline;
tx1.nPort

ans =

      2
```

rfckt.series.AnalyzedResult property

Purpose

Computed S-parameters, noise figure, OIP3, and group delay values

Values

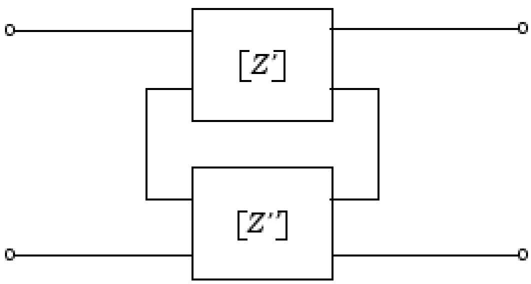
`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

The `analyze` method computes the S-parameters of the `AnalyzedResult` property using the data stored in the `Ckts` property as follows:

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

   where

   $$[Z'] = \begin{bmatrix} Z_{11}' & Z_{12}' \\ Z_{21}' & Z_{22}' \end{bmatrix}$$

   $$[Z''] = \begin{bmatrix} Z_{11}'' & Z_{12}'' \\ Z_{21}'' & Z_{22}'' \end{bmatrix}$$

2 The `analyze` method 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.

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

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

The `analyze` method uses the series S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

Examples

```
tx1 = rfckt.txline;
tx2 = rfckt.txline;
ser = rfckt.series('Ckts',{tx1,tx2})
analyze(ser,[1e9:1e7:2e9]);
ser.AnalyzedResult

ans =

    Name: 'Data object'
    Freq: [101x1 double]
    S_Parameters: [2x2x101 double]
    GroupDelay: [101x1 double]
    NF: [101x1 double]
    OIP3: [101x1 double]
    Z0: 50
    ZS: 50
    ZL: 50
    IntpType: 'Linear'
```

rfckt.series.Ckts property

Purpose Circuit objects in network

Values Cell

Description Cell array containing handles to all circuit objects in the network. All circuits must be 2-port and linear. This property is empty by default.

Examples

```
tx1 = rfckt.txline;
tx2 = rfckt.txline;
ser = rfckt.series;
ser.Ckts = {tx1,tx2};
ser.Ckts

ans =

    [1x1 rfckt.txline] [1x1 rfckt.txline]
```

Purpose Object name

Values `'Series Connected Network'`

Description Read-only string that contains the name of the object.

Examples

```
ser = rfckt.series;
ser.Name

ans =

          Series Connected Network
```

rfckt.series.nPort property

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
ser = rfckt.series;
ser.nPort

ans =

     2
```

Purpose

Computed S-parameters, noise figure, OIP3, and group delay values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

The `analyze` method computes the S-parameters of the `AnalyzedResult` property using the data stored in the `rfckt.seriesrlc` object properties by first calculating the ABCD-parameters for the circuit, and then converting the ABCD-parameters to S-parameters using the `abcd2s` function. For this circuit, A = 1, B = Z, C = 0, and D = 1, where

$$Z = \frac{-LC\omega^2 + jRC\omega + 1}{jC\omega}$$

and $\omega = 2\pi f$.

The `analyze` method uses the S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

Examples

```
rlc1 = rfckt.seriesrlc;
analyze(rlc1,[1e9,2e9,3e9]);
rlc1.AnalyzedResult

ans =

           Name: 'Data object'
           Freq: [3x1 double]
           S_Parameters: [2x2x3 double]
           GroupDelay: [3x1 double]
           NF: [3x1 double]
           OIP3: [3x1 double]
           Z0: 50
           ZS: 50
```

```
ZL: 50
IntpType: 'Linear'
```

Purpose Capacitance value

Values Scalar

Description Positive capacitance value in farads. The default is `Inf`.

Examples

```
rlc1=rfckt.seriesrlc;
rlc1.C = 1e-12;
```

rfckt.seriesrlc.L property

Purpose

Inductance value

Values

Scalar

Description

Positive inductance value in henries. The default is `0`.

Examples

```
rlc1 = rfckt.seriesrlc;
rlc1.L = 1e-9;
```

Purpose Object name

Values `'Series RLC'`

Description Read-only string that contains the name of the object.

Examples

```
rlc1 = rfckt.seriesrlc;
rlc1.Name

ans =

          Series RLC
```

rfckt.seriesrlc.R property

Purpose Resistance value

Values Scalar

Description Positive resistance in ohms. The default is `0`.

Examples

```
rlc1 = rfckt.seriesrlc;
rlc1.R = 10;
```

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
filter = rfckt.seriesrlc;
filter.nPort

ans =

     2
```

rfckt.shuntrlc.AnalyzedResult property

Purpose

Computed S-parameters, noise figure, OIP3, and group delay values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

The `analyze` method computes the S-parameters of the `AnalyzedResult` property using the data stored in the `rfckt.shuntrlc` object properties by first calculating the ABCD-parameters for the circuit, and then converting 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 $\omega = 2\pi f$.

The `analyze` method uses the S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

Examples

```
rlc1 = rfckt.shuntrlc;
analyze(rlc1,[1e9,2e9,3e9]);
rlc1.AnalyzedResult

ans =

            Name: 'Data object'
            Freq: [3x1 double]
            S_Parameters: [2x2x3 double]
            GroupDelay: [3x1 double]
            NF: [3x1 double]
            OIP3: [3x1 double]
            Z0: 50
```

```
ZS: 50
ZL: 50
IntpType: 'Linear'
```

rfckt.shuntrlc.C property

Purpose Capacitance value

Values Scalar

Description Positive capacitance value in farads. The default is `0`.

Examples

```
rlc1=rfckt.shuntrlc;
rlc1.C = 1e-12;
```

Purpose Inductance value

Values Scalar

Description Positive inductance value in henries. The default is `Inf`.

Examples

```
rlc1 = rfckt.shuntrlc;
rlc1.L = 1e-9;
```

rfckt.shuntrlc.Name property

Purpose Object name

Values `'Shunt RLC'`

Description Read-only string that contains the name of the object.

Examples

```
rlc1 = rfckt.shuntrlc;
rlc1.Name

ans =

           Shunt RLC
```

Purpose Resistance value

Values Scalar

Description Positive resistance in ohms. The default is `Inf`.

Examples

```
rlc1 = rfckt.shuntrlc;
rlc1.R = 10;
```

rfckt.shuntrlc.nPort property

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
filter = rfckt.shuntrlc;
filter.nPort

ans =

     2
```

Purpose

Computed S-parameters, noise figure, OIP3, and group delay values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

The `analyze` method treats the transmission line, which can be lossy or lossless, as a 2-port linear network. It computes the `AnalyzedResult` property of a stub or as a stubless line using the data stored in the `rfckt.twowire` object properties as follows:

- If you model the transmission line as a stubless line, the `analyze` method first calculates the ABCD-parameters at each frequency contained in the modeling frequencies vector. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

  The `analyze` method calculates the ABCD-parameters using the physical length of the transmission line, d, and the complex propagation constant, k, using the following equations:

$$A = \frac{e^{kd} + e^{-kd}}{2}$$

$$B = \frac{Z_0 * \left(e^{kd} - e^{-kd}\right)}{2}$$

$$C = \frac{e^{kd} - e^{-kd}}{2 * Z_0}$$

$$D = \frac{e^{kd} + e^{-kd}}{2}$$

  Z_0 and k are vectors whose elements correspond to the elements of f, the vector of frequencies specified in the `analyze` input argument `freq`. Both can be expressed in terms of the resistance (R), inductance

(L), conductance (G), and capacitance (C) per unit length (meters) as follows:

$$Z_0 = \sqrt{\frac{R + j2\pi fL}{G + j2\pi fC}}$$

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

where

$$R = \frac{1}{\pi a \sigma_{cond} \delta_{cond}}$$

$$L = \frac{\mu}{\pi} \operatorname{a} \cosh\left(\frac{D}{2a}\right)$$

$$G = \frac{\pi \omega \varepsilon''}{\operatorname{a} \cosh\left(\frac{D}{2a}\right)}$$

$$C = \frac{\pi \varepsilon}{\operatorname{a} \cosh\left(\frac{D}{2a}\right)}$$

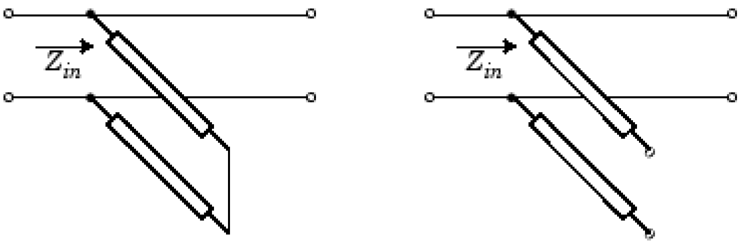
In these equations:

- w is the plate width.
- d is the plate separation.
- σ_{cond} is the conductivity in the conductor.
- μ is the permeability of the dielectric.
- ε is the permittivity of the dielectric.
- ε'' is the imaginary part of ε, $\varepsilon'' = \varepsilon_0 \varepsilon_r \tan \delta$, where:
  - ε_0 is the permittivity of free space.
  - ε_r is the `EpsilonR` property value.
  - $\tan \delta$ is the `LossTangent` property value.

- δ_{cond} is the skin depth of the conductor, which the block calculates as $1/\sqrt{\pi f \mu \sigma_{cond}}$.
- f is a vector of modeling frequencies determined by the Outport block.

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

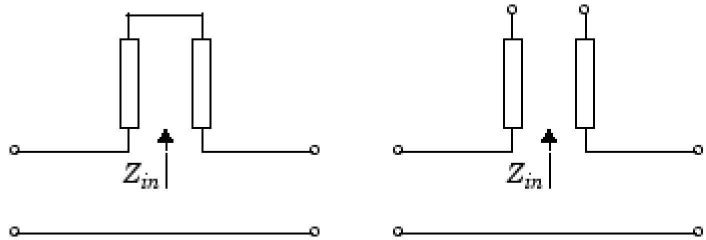
  When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown in the following figure.



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 in the following figure.



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

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

The `analyze` method uses the S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

Examples

```
tx1 = rfckt.twowire;
analyze(tx1,[1e9,2e9,3e9]);
tx1.AnalyzedResult

ans =

          Name: 'Data object'
          Freq: [3x1 double]
          S_Parameters: [2x2x3 double]
          GroupDelay: [3x1 double]
          NF: [3x1 double]
          OIP3: [3x1 double]
          Z0: 50
          ZS: 50
          ZL: 50
          IntpType: 'Linear'
```

Purpose Relative permittivity of dielectric

Values Scalar

Description The ratio of the permittivity of the dielectric, ε, to the permittivity of free space, ε_0. The default value is `2.3`.

Examples

```
tx1=rfckt.twowire;
tx1.EpsilonR=2.7;
```

rfckt.twowire.LineLength property

Purpose Transmission line length

Values Scalar

Description The physical length of the transmission line in meters. The default is 0.01.

Examples

```
tx1 = rfckt.twowire;
tx1.LineLength = 0.001;
```

Purpose Tangent of loss angle

Values Scalar

Description The loss angle tangent of the dielectric. The default is `0`.

Examples

```
tx1=rfckt.twowire;
tx1.LossTangent=0.002;
```

rfckt.twowire.MuR property

Purpose Relative permeability of dielectric

Values Scalar

Description The ratio of the permeability of the dielectric, μ, to the permeability of free space, μ_0. The default value is 1.

Examples Change the relative permeability of the dielectric:

```
tx1=rfckt.twowire;
tx1.MuR=0.8;
```

Purpose Object name

Values `'Two-Wire Transmission Line'`

Description Read-only string that contains the name of the object.

Examples

```
tx1 = rfckt.twowire;
tx1.Name

ans =

          Two-Wire Transmission Line
```

rfckt.twowire.Radius property

Purpose Wire radius

Values Scalar

Description The radius of the conducting wires, in meters. The default is `6.7e-4`.

Examples

```
tx1=rfckt.twowire;
tx1.Radius=0.0031;
```

Purpose

Distance between wires

Values

Scalar

Description

Distance, in meters, separating the wire centers. The default is `0.0016`.

Examples

```
tx1=rfckt.twowire;
tx1.Separation=0.8e-3;
```

rfckt.twowire.SigmaCond property

Purpose Conductor conductivity

Values Scalar

Description Conductivity, in Siemens per meter (S/m), of the conductor. The default is `Inf`.

Examples

```
tx1=rfckt.twowire;
tx1.SigmaCond=5.81e7;
```

Purpose Type of stub

Values `'NotAStub'` (default), `'Series'`, or `'Shunt'`

Description String that specifies what type of stub, if any, to include in the transmission line model.

Examples

```
tx1 = rfckt.twowire;
tx1.StubMode = 'Series';
```

rfckt.twowire.Termination property

Purpose Stub transmission line termination

Values `'NotApplicable'` (default), `'Open'`, or `'Short'`.

Description String that specifies what type of termination to use for `'Shunt'`and `'Series'` stub modes. `Termination` is ignored if the line has no stub. Use `'NotApplicable'` when `StubMode` is `'NotAStub'`.

Examples

```
tx1 = rfckt.twowire;
tx1.StubMode = 'Series';
tx1.Termination = 'Short';
```

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
tx1 = rfckt.twowire;
tx1.nPort

ans =

     2
```

rfckt.txline.AnalyzedResult property

Purpose Computed S-parameters, noise figure, OIP3, and group delay values

Values `rfdata.data` object

Description Handle to an `rfdata.data` object that contains the S-parameters, noise figure, OIP3, and group delay values computed over the specified frequency range using the `analyze` method. This property is empty by default.

The `analyze` method treats the transmission line, which can be lossy or lossless, as a 2-port linear network. It computes the `AnalyzedResult` property of a stub or as a stubless line using the data stored in the `rfckt.txline` object properties as follows:

- If you model the transmission line as a stubless line, the `analyze` method first calculates the ABCD-parameters at each frequency contained in the modeling frequencies vector. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

  The `analyze` method calculates the ABCD-parameters using the physical length of the transmission line, d, and the complex propagation constant, k, using the following equations:

  $$A = \frac{e^{kd} + e^{-kd}}{2}$$

  $$B = \frac{Z_0 * \left(e^{kd} - e^{-kd}\right)}{2}$$

  $$C = \frac{e^{kd} - e^{-kd}}{2 * Z_0}$$

  $$D = \frac{e^{kd} + e^{-kd}}{2}$$

  Z_0 is the specified characteristic impedance. k is a vector whose elements correspond to the elements of the input vector `freq`. The `analyze` method calculates k from the specified properties as

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

$$\alpha_a = -\ln\left(10^{\alpha/20}\right)$$

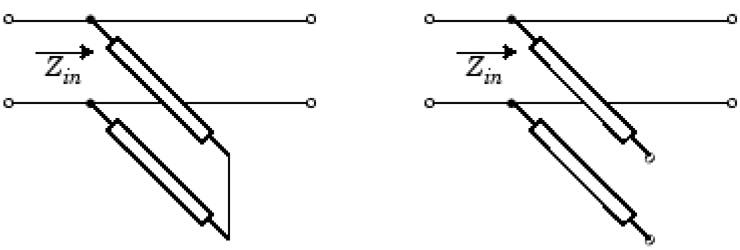
The wave number β is related to the specified phase velocity, V_p, by

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

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*.

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

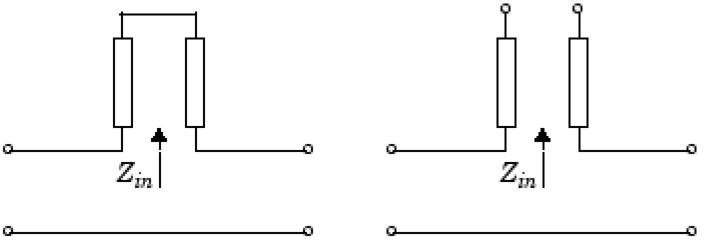
  When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown in the following figure.

  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 in the following figure.



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

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

The analyze method uses the S-parameters to calculate the group delay values at the frequencies specified in the analyze input argument freq, as described in the analyze reference page.

Examples

```
tx1 = rfckt.txline;
analyze(tx1,[1e9,2e9,3e9]);
tx1.AnalyzedResult

ans =
```

```
Name: 'Data object'
Freq: [3x1 double]
S_Parameters: [2x2x3 double]
GroupDelay: [3x1 double]
NF: [3x1 double]
OIP3: [3x1 double]
Z0: 50
ZS: 50
ZL: 50
IntpType: 'Linear'
```

rfckt.txline.Freq property

Purpose Frequency data

Values Vector

Description M-element vector of frequency values in hertz for the loss and phase velocity values in the `Loss` and `PV` properties. The values must be positive, and the order of the frequencies must correspond to the order of the loss and phase velocity values. This property is empty by default.

Examples

```
f = [2.08 2.10]*1.0e9;
tx1 = rfckt.txline;
tx1.Freq = f;
```

Purpose

Interpolation method

Values

`'Linear'` (default), `'Spline'`, or `'Cubic'`

Description

The `analyze` method is flexible in that it does not require the frequencies of the specified S-parameters to match the requested analysis frequencies. If needed, `analyze` applies the interpolation and extrapolation method specified in the `IntpType` property to the specified data to create a new set of data at the requested analysis frequencies. The following table lists the available interpolation methods and describes each one.

Method	Description
`Linear` (default)	Linear interpolation
`Spline`	Cubic spline interpolation
`Cubic`	Piecewise cubic Hermite interpolation

Examples

```
tx1 = rfckt.txline;
tx1.IntpType = 'cubic';
```

rfckt.txline.LineLength property

Purpose Transmission line length

Values Scalar

Description The physical length of the transmission line in meters. The default is 0.01.

Examples

```
tx1 = rfckt.txline;
tx1.LineLength = 0.001;
```

Purpose Transmission line loss

Values Vector

Description M-element vector of line loss values, in decibels per meter, 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, and must be nonnegative. The default is `0`.

Examples

```
tx1 = rfckt.txline;
tx1.Loss = [1.5 3.1]*1e-2;
```

rfckt.txline.Name property

Purpose Object name

Values `'Transmission Line'`

Description Read-only string that contains the name of the object.

Examples

```
tx1 = rfckt.txline;
tx1.Name

ans =

            Transmission Line
```

Purpose

Phase velocity

Values

Vector

Description

M-element vector of phase velocity values, in meters per second, that correspond to the frequencies stored in the `Freq` property. Propagation velocity of a uniform plane wave on the transmission line. The default is `299792458`.

Examples

```
tx1 = rfckt.txline;
tx1.PV = [1.5 3.1]*1e9;
```

rfckt.txline.StubMode property

Purpose Type of stub

Values `'NotAStub'` (default), `'Series'`, or `'Shunt'`

Description String that specifies what type of stub, if any, to include in the transmission line model.

Examples

```
tx1 = rfckt.txline;
tx1.StubMode = 'Series';
```

Purpose Stub transmission line termination

Values `'NotApplicable'` (default), `'Open'`, or `'Short'`.

Description String that specifies what type of termination to use for `'Shunt'`and `'Series'` stub modes. `Termination` is ignored if the line has no stub. Use `'NotApplicable'` when `StubMode` is `'NotAStub'`.

Examples

```
tx1 = rfckt.txline;
tx1.StubMode = 'Series';
tx1.Termination = 'Short';
```

rfckt.txline.Z0 property

Purpose

Characteristic impedance

Values

Vector

Description

Vector of characteristic impedance values, in ohms, that correspond to the frequencies stored in the `Freq` property. The default is `50` ohms.

Examples

```
tx1 = rfckt.txline;
tx1.Z0 = 75;
```

Purpose Number of ports

Values 2

Description A read-only integer that indicates the object has two ports.

Examples

```
tx1 = rfckt.txline;
tx1.nPort

ans =

     2
```

rfdata.data.Freq property

Purpose

Frequency data

Values

Vector

Description

M-element vector of frequency values in hertz for the S-parameters in the `S_Parameters` property. The values must be positive, and the order of the frequencies must correspond to the order of the S-parameters. This property is empty by default.

Examples

```
f = [2.08 2.10]*1.0e9;
txdata = rfdata.data;
txdata.Freq = f;
```

Purpose Group delay data

Values Vector

Description M-element vector of group delay values in seconds that the toolbox calculates at each frequency in the `Freq` property when you call the `analyze` method. This property is empty by default.

rfdata.data.IntpType property

Purpose

Interpolation method

Values

`'Linear'` (default), `'Spline'`, or `'Cubic'`

Description

The `analyze` method is flexible in that it does not require the frequencies of the specified S-parameters to match the requested analysis frequencies. If needed, `analyze` applies the interpolation and extrapolation method specified in the `IntpType` property to the specified data to create a new set of data at the requested analysis frequencies. The following table lists the available interpolation methods and describes each one.

Method	Description
`Linear` (default)	Linear interpolation
`Spline`	Cubic spline interpolation
`Cubic`	Piecewise cubic Hermite interpolation

Examples

```
txdata = rfdata.data;
txdata.IntpType = 'cubic'

txdata =

           Name: 'Data object'
           Freq: []
           S_Parameters: []
           GroupDelay: []
           NF: 0
           OIP3: Inf
           Z0: 50
           ZS: 50
           ZL: 50
           IntpType: 'Cubic'
```

Purpose Noise figure

Values Scalar

Description The amount of noise relative to a noise temperature of 290 degrees kelvin, in decibels. The default value of zero indicates a noiseless system.

Examples

```
txdata = rfdata.data;
txdata.NF=3

txdata =

            Name: 'Data object'
            Freq: []
            S_Parameters: []
            GroupDelay: []
            NF: 3
            OIP3: Inf
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Linear'
```

rfdata.data.Name property

Purpose Object name

Values `'Data object'`

Description Read-only string that contains the name of the object.

Examples

```
txdata = rfdata.data;
txdata.Name

ans =

      Data object
```

Purpose Output third-order intercept point

Values Scalar

Description Signal distortion in watts. This property represents the hypothetical output signal level at which the third-order tones would reach the same amplitude level as the desired input tones. The default is `Inf`.

Examples

```
txdata = rfdata.data;
txdata.OIP3 = 30;
```

rfdata.data.S_Parameters property

Purpose S-parameter data

Values

Description 2-by-2-by-M array of S-parameters of the circuit described by the `rfdata.data` object, where M is the number of frequencies at which the network parameters are specified. The values correspond to the frequencies stored in the `Freq` property. This property is empty by default.

Examples

```
s_vec(:,:,1) = ...
    [-0.724725-0.481324i, -0.685727+1.782660i; ...
      0.000000+0.000000i, -0.074122-0.321568i];
s_vec(:,:,2) = ...
    [-0.731774-0.471453i, -0.655990+1.798041i; ...
      0.001399+0.000463i, -0.076091-0.319025i];
s_vec(:,:,3) = ...
    [-0.738760-0.461585i, -0.626185+1.813092i; ...
      0.002733+0.000887i, -0.077999-0.316488i];
txdata = rfdata.data;
txdata.S_Parameters = s_vec;
```

Purpose Reference impedance

Values Scalar

Description Scalar reference impedance in ohms. The default is 50 ohms.

Examples

```
txdata = rfdata.data;
txdata.Z0 = 75;
```

rfdata.data.ZL property

Purpose Load impedance

Values Scalar

Description Scalar load impedance in ohms. The default is 50 ohms.

Examples

```
txdata = rfdata.data;
txdata.ZL = 75;
```

Purpose Source impedance

Values Scalar

Description Scalar source impedance in ohms. The default is 50 ohms.

Examples

```
txdata = rfdata.data;
txdata.ZS = 75;
```

rfdata.ip3.Data property

Purpose

Third-order intercept values

Values

Vector

Description

M-element vector of IP3 data, in watts, that corresponds to the frequencies stored in the `Freq` property. The default is `Inf`.

Examples

```
ip3_vec = [-5.2 7.1];
ip3data = rfdata.ip3;
ip3data.Data = ip3_vec;
```

Purpose Frequency data

Values Vector

Description M-element vector of frequency values in hertz for the IP3 data in the `Data` property. The values must be positive, and the order of the frequencies must correspond to the order of the IP3 values. This property is empty by default.

Examples

```
f = [2.08 2.10]*1.0e9;
ip3data = rfdata.ip3;
ip3data.Freq = f;
```

rfdata.ip3.Name property

Purpose Object name

Values `'3rd order intercept'`

Description Read-only string that contains the name of the object.

Examples

```
ip3data = rfdata.ip3;
ip3data.Name

ans =

      3rd order intercept
```

Purpose

Power reference type

Values

`'OIP3'` (default) or `'IIP3'`

Description

String that indicates whether the specified IP3 data is output or input IP3.

Examples

```
ip3data = rfdata.ip3;
ip3data.Type

ans =

      OIP3
```

rfdata.mixerspur.Data property

Purpose

Mixer spur power values

Values

Matrix

Description

Matrix of values, in decibels, by which the mixer spur power is less than the power at the fundamental output frequency. Values must be between 0 and 99. This property is empty by default.

Examples

```
spurs = rfdata.mixerspur...
         ('Data',[2 5 3; 1 0 99; 10 99 99],...
          'PinRef',3,'PLORef',5)
spurs.data

ans =

     2     5     3
     1     0    99
    10    99    99
```

Purpose Object name

Values `'Intermodulation table'`

Description Read-only string that contains the name of the object.

Examples

```
spurdata = rfdata.mixerspur;
spurdata.Name

ans =

      Intermodulation table
```

rfdata.mixerspur.PLORef property

Purpose Reference local oscillator power

Values Scalar

Description Scalar local oscillator power reference, in decibels relative to one milliwatt. The default is `0`.

Examples

```
spurs = rfdata.mixerspur...
          ('Data',[2 5 3; 1 0 99; 10 99 99],
           'PinRef',3,'PLORef',5)
spurs.PLORef

ans =

      5
```

Purpose Reference input power

Values Scalar

Description Scalar input power reference, in decibels relative to one milliwatt. The default is `0`.

Examples

```
spurs = rfdata.mixerspur...
         ('Data',[2 5 3; 1 0 99; 10 99 99],...
          'PinRef',3,'PLORef',5)
spurs.PinRef

ans =

      3
```

rfdata.network.Data property

Purpose Network parameter data

Values Array

Description 2-by-2-by-M array of network parameters, where M is the number of frequencies at which the network parameters are specified. The values correspond to the frequencies stored in the `Freq` property. This property is empty by default.

Examples

```
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;
netdata.Data=y;
```

Purpose

Frequency data

Values

Vector

Description

M-element vector of frequency values in hertz for the network parameters in the `Data` property. The values must be positive, and the order of the frequencies must correspond to the order of the network parameters. This property is empty by default.

Examples

```
f = [2.08 2.10]*1.0e9;
netdata = rfdata.network;
netdata.Freq=f;
```

rfdata.network.Name property

Purpose Object name

Values `'Network parameters'`

Description Read-only string that contains the name of the object.

Examples

```
netdata=rfdata.network;
netdata.Name

ans =

      Network parameters
```

Purpose Type of network parameters.

Values `'S'`, `'Y'`, `'Z'`, `'ABCD'`, `'H'`, `'G'`, or `'T'`

Description String that indicates whether the rfdata.network object .

Examples

```
netdata=rfdata.network;
netdata.Type='Y';
```

rfdata.network.Z0 property

Purpose Reference impedance

Values Scalar

Description Scalar reference impedance in ohms. This property is only available when the `Type` property is set to `'S'`. The default is 50 ohms.

Examples

```
netdata=rfdata.network;
netdata.z0=75;
```

Purpose Noise figure values

Values Vector

Description M-element vector of noise figure data, in dB, that corresponds to the frequencies stored in the `Freq` property. The default is `0`.

Examples

```
nf_vec = [1.2 3.1];
nfdata = rfdata.nf;
nfdata.Data = nf_vec;
```

rfdata.nf.Freq property

Purpose

Frequency data

Values

Vector

Description

M-element vector of frequency values in hertz for the noise figure data in the `Data` property. The values must be positive, and the order of the frequencies must correspond to the order of the noise figure values. This property is empty by default.

Examples

```
f = [2.08 2.10]*1.0e9;
nfdata = rfdata.nf;
nfdata.Freq = f;
```

Purpose Object name

Values `'Noise figure'`

Description Read-only string that contains the name of the object.

Examples

```
nfdata = rfdata.nf;
nfdata.Name

ans =

      Noise figure
```

rfdata.noise.FMIN property

Purpose

Minimum noise figure data

Values

Vector

Description

M-element vector of minimum noise figure values, in decibels, that correspond to the frequencies stored in the `Freq` property. The default is `1`.

Examples

```
fmin = [12.08 13.40];
noisedata = rfdata.noise;
noisedata.FMIN = fmin;
```

Purpose

Frequency data

Values

Vector

Description

M-element vector of frequency values in hertz for the spot noise data in the `FMIN`, `GAMMAOPT`, and `RN` properties. The values must be positive, and the order of the frequencies must correspond to the order of the spot noise values. This property is empty by default.

Examples

```
f = [2.08 2.10]*1.0e9;
noisedata = rfdata.noise;
noisedata.Freq = f;
```

rfdata.noise.GAMMAOPT property

Purpose Optimum source reflection coefficients

Values Vector

Description M-element vector of optimum source reflection coefficients that correspond to the frequencies stored in the `Freq` property. The default is `1`.

Examples

```
gopt = [0.2484-1.2102j 1.0999-0.9295j];
noisedata = rfdata.noise;
noisedata.GAMMAOPT = gopt;
```

Purpose Object name

Values `'Spot noise data'`

Description Read-only string that contains the name of the object.

Examples

```
noisedata = rfdata.noise;
noisedata.Name

ans =

      Spot noise data
```

rfdata.noise.RN property

Purpose

Equivalent normalized noise resistance data

Values

Vector

Description

M-element vector of equivalent normalized noise resistance values that correspond to the frequencies stored in the `Freq` property. The default is `1`.

Examples

```
rn = [0.26 0.45];
noisedata = rfdata.noise;
noisedata.RN = rn;
```

Purpose

Frequency data

Values

Vector

Description

M-element vector of frequency values in hertz for the power data in the `Phase`, `Pin`, and `Pout` properties. The values must be positive, and the order of the frequencies must correspond to the order of the phase and power values. This property is empty by default.

Examples

```
f = [2.08 2.10]*1.0e9;
powerdata = rfdata.power;
powerdata.Freq = f;
```

rfdata.power.Name property

Purpose Object name

Values `'Power data'`

Description Read-only string that contains the name of the object.

Examples

```
powerdata = rfdata.power;
powerdata.Name

ans =

      Power data
```

Purpose

Phase shift data

Values

Cell

Description

M-element cell of phase shift values, in degrees, where each element corresponds to a frequency stored in the `Freq` property. The values within each element correspond to the input power values stored in the `Pin` property. The default is `1`.

Examples

```
phase = {[27.1 35.3],[15.4 19.3 21.1]};
powerdata = rfdata.power;
powerdata.Phase = phase;
```

rfdata.power.Pin property

Purpose

Input power data

Values

Cell

Description

M-element cell of input power values, in watts, where each element corresponds to a frequency stored in the `Freq` property. 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. The default is `1`.

Examples

```
pin = {[0.001 0.002],[0.001 0.005 0.01]};
powerdata = rfdata.power;
powerdata.Pin = pin;
```

Purpose

Output power data

Values

Cell

Description

M-element cell of output power values, in watts, where each element corresponds to a frequency stored in the `Freq` property. The values within each element correspond to the input power values stored in the `Pin` property. The default is `1`.

Examples

```
pout = {[0.0025 0.0031],[0.0025 0.0028 0.0028]};
powerdata = rfdata.power;
powerdata.Pout = pout;
```

rfmodel.rational.A property

Purpose Poles of rational function

Values Vector

Description Complex vector containing poles of the rational function in radians per second. Its length, shown in as `n`, must be equal to the length of the vector you provide for `'C'`. `n` is the number of poles in the rational function model. This property is empty by default.

Examples

```
rat = rfmodel.rational;
rat.A = [-0.0532 + 1.3166i; -0.0532 - 1.3166i]*1e10;
```

Purpose

Residues of rational function

Values

Vector

Description

Complex vector containing residues of the rational function in radians per second. Its length, shown in as `n`, must be equal to the length of the vector you provide for `'A'`. `n` is the number of residues in the rational function model. This property is empty by default.

Examples

```
rat = rfmodel.rational;
rat.C = [4.4896 - 4.5025i; 4.4896 + 4.5025i]*1e9;
```

rfmodel.rational.D property

Purpose

Frequency response offset

Values

Scalar

Description

Scalar value specifying the constant offset in the frequency response of the rational function. The default is `0`.

Examples

```
rat = rfmodel.rational;
rat.D = 1e-3;
```

Purpose

Frequency response time delay

Values

Scalar

Description

Scalar value specifying the time delay, in seconds, in the frequency response of the rational function. The default is `0`.

Examples

```
rat = rfmodel.rational;
rat.Delay = 1e-9;
```

rfmodel.rational.Name property

Purpose Object name

Values `'Rational Function'`

Description Read-only string that contains the name of the object.

Examples

```
rat = rfmodel.rational;
rat.Name

ans =

            Rational Function
```

Methods — Alphabetical List

Purpose

Add mixer to multiband transmitter or receiver for IF planning analysis

Syntax

```
addMixer(hif,imt,rfbw,rfcf,injection,newIFBW)
```

Description

`addMixer(hif,imt,rfbw,rfcf,injection,newIFBW)` adds a mixer to a multiband transmitter or receiver object `hif` as part of an intermediate-frequency (IF) planning analysis workflow.

Input Arguments

hif

Specify the handle of the `OpenIF` object.

imt

Provide the intermodulation table (IMT) for the mixer as a matrix of real numbers. Each entry in the intermodulation table is a number between `0` and `99`. The matrix must be 2-by-2 or larger, and `imt(2,2)` must equal `0`.

rfcf

Specify the RF center frequency of the mixer in Hz.

rfbw

Specify the RF bandwidth of the mixer in Hz.

injection

Specify the mixer type. The available values for `injection` change depending on the setting of `hif.IFLocation`.

When `hif.IFLocation` is `'MixerOutput'`, the available values for `injection` are as follows.

- `'low'` — The LO frequency is less than the RF (low-side injection).
- `'high'` — The LO frequency is greater than the RF (high-side injection).

When `hif.IFLocation` is `'MixerInput'`, the available values for `injection` are as follows.

- `'sum'` — The RF signal is the sum of the LO frequency and the IF.
- `'diff'` — The RF signal is the difference between the LO frequency and the IF.

newIFBW

Specify the IF bandwidth in Hz after mixing. Use this argument if the IF bandwidth before mixing (defined by the `hif.IFBW` property) is different from the bandwidth after mixing.

Examples

- The `OpenIF` class reference page contains an example that shows how to find the spur-free zones of a multiband receiver with three mixers.
- The example Finding Free IF Bandwidths shows how to use information from a spur graph to design a multiband receiver with spur-free zones.

See Also

`getSpurData` | `getSpurFreeZoneData` | `report` | `show` | `OpenIF`

analyze

Purpose Analyze circuit object in frequency domain

Syntax

```
analyze(h,freq)
analyze(h,freq,zl,zs,zo,aperture)
analyze(h,freq,'condition1',value1,...,'conditionm',valuem)
```

Description

`analyze(h,freq)` calculates the following circuit data at the specified frequency values:

- Circuit network parameters
- Noise figure
- Output third-order intercept point
- Power data
- Phase noise
- Voltage standing-wave ratio
- Power gain
- Group delay
- Reflection coefficients
- Stability data
- Transfer function

`h` is the handle of the circuit object to be analyzed. `freq` is a vector of frequencies, specified in hertz, at which to analyze the circuit. OIP_3 is always infinite for passive circuits.

`analyze(h,freq,zl,zs,zo,aperture)` calculates the circuit data at the specified frequency values. The arguments `zl`, `zs`, `zo`, and `aperture` are optional. `zl`, `zs`, and `zo` represent the circuit load, circuit source, and reference impedances of the S-parameters, respectively. The default value of all these arguments is 50 ohms.

Note When you specify impedance values, the `analyze` method changes the object's values to match your specification.

The `aperture` argument determines the two frequency points that the `analyze` method uses to compute the group delay for each frequency in `freq`. `aperture` can be a positive scalar or a vector of the same length of as `freq`.

Note For `rfckt.datafile`, `rfckt.passive`, `rfckt.amplifier`, and `rfckt.mixer` objects that contain measured S-parameter data, the `analyze` method uses the two nearest measurement points to compute the group delay, regardless of the value of `aperture`.

Group delay τ_g at each frequency point f is the negative slope of the phase angle of S_{21} with respect to f:

$$\tau_g(f) = -\frac{\Delta\phi}{\Delta\omega} = -\frac{\arg\left(S_{21}(f_+)\right) - \arg\left(S_{21}(f_-)\right)}{2\pi\left(f_+ - f_-\right)}$$

where:

- f_+ is:
  - f(1 + `aperture`/2) for `aperture` < 1.
  - f + `aperture`/2 for `aperture` ≥ 1.

  If f is the maximum value of `freq`, then $f_+ = f$.
- f_- is:
  - f (1 – `aperture`/2) for `aperture` < 1.
  - f – `aperture`/2 for `aperture` ≥ 1.

  If f is the minimum value of `freq`, then $f_- = f$.

By default, `analyze` calculates the group delay in nanoseconds.

The value of `aperture` affects the accuracy of the computed group delay. If `aperture` is too large, the slope estimate may be not accurate. If `aperture` is too small, the computer numerical error may affect the accuracy of the group delay result.

`analyze(h,freq,'`*`condition1`*`',`*`value1`*`,...,'`*`conditionm`*`',`*`valuem`*`)` calculates the circuit data at the specified frequency values and operating conditions for the object `h`. The inputs `'`*`condition1`*`',`*`value1`*`,...,'`*`conditionm`*`',`*`valuem`* are the condition/value pairs at which to analyze the object. Use this syntax for `rfckt.amplifier`, `rfckt.mixer`, and `rfdata.data` objects where the condition/value pairs are operating conditions from a `.p2d` or `.s2d` file.

Note When you specify condition/value pairs, the `analyze` method changes the object's values to match your specification.

When you analyze a network that contains several objects, RF Toolbox software does not issue an error or warning if the specified conditions cannot be applied to all objects. For some networks, because there is no error or warning, you can call the `analyze` method once to apply the same set of operating conditions to any objects where operating conditions are applicable. However, you may want to analyze a network that contains one or more of the following:

- Several objects with different sets of operating conditions.
- Several objects with the same set of operating conditions that are configured differently.

To analyze such a network, you should use the `setop` method to configure the operating conditions of each individual object before analyzing the network.

Analysis of Circuit Objects

For most circuit objects, the `AnalyzedResult` property is empty until the `analyze` method is applied to the circuit object. However, the following four circuit objects are the exception to this rule:

- `rfckt.datafile` — By default, the `AnalyzedResult` property of `rfckt.datafile` objects contains the S-parameter, noise figure, and group delay values that are calculated over the network parameter frequencies in the `passive.s2p` data file. OIP3 is ∞ by default because the data in `passive.s2p` is passive.
- `rfckt.passive` — By default, the `AnalyzedResult` property of `rfckt.passive` objects contains the S-parameter, noise figure, and group delay 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. OIP3 is always ∞ for `rfckt.passive` objects because the data is passive.
- `rfckt.amplifier` — By default, the `AnalyzedResult` property of `rfckt.amplifier` objects contains the S-parameter, noise figure, OIP3, and group delay values that result from 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.
- `rfckt.mixer` — By default, the `AnalyzedResult` property of `rfckt.mixer` objects contains the S-parameter, noise figure, OIP3, and group delay values that result from 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.

For a detailed explanation of how the `analyze` method calculates the network parameters, noise figure values, and OIP3 values for a particular object, see the `AnalyzedResult` property on the reference page for that object.

References

`http://www.microwaves101.com/encyclopedia/groupdelaymeasurements.cfm`

See Also

`calculate` | `extract` | `getz0` | `listformat` | `listparam` | `loglog` | `plot` | `plotyy` | `polar` | `semilogx` | `semilogy` | `smith` | `read` | `restore` | `write`

Purpose

Calculate specified parameters for circuit object

Syntax

```
[data,params,freq] = calculate(h,'parameter1',...,'parametern',
   'format')
[ydata,params,xdata] = calculate(h,'parameter1',...,
   'parametern', 'format',xparameter,xformat, 'condition1',
   value1,...,'conditionm',valuem, 'freq',freq,'pin',pin)
```

Description

`[data,params,freq] = calculate(h,'parameter1',...,'parametern', 'format')` calculates the specified 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` is the list of parameters to be calculated. Use the `listparam` method to get a list of the valid parameters for a circuit object.

`format` is the format of the output `data`. The format determines if RF Toolbox software converts the parameter values to a new set of units, or operates on the components of complex parameter values.

For example:

- Specify `format` as `Real` to compute the real part of the selected parameter.
- Specify `format` as `'none'` to return the parameter values unchanged.

Use the `listformat` method to get a list of the valid formats for a particular parameter.

The output `params` is an n-element cell array containing the names, as strings, of the parameters in `data`. `freq` is a vector of frequencies at which the parameters are known.

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

`[ydata,params,xdata] = calculate(h,'parameter1',...,'parametern', 'format',xparameter,xformat, 'condition1',value1,...,'conditionm',valuem, 'freq',freq,'pin',pin)` calculates the specified parameters at the specified operating conditions for the object `h`.

`xparameter` is the independent parameter for which to calculate the specified parameters. Several `xparameter` values are available for all objects. When you import `rfckt.amplifier`, `rfckt.mixer`, or `rfdata.data` object specifications from a `.p2d` or `.s2d` file, you can also specify any operating conditions from the file that have numeric values, such as bias.

The following table shows the most commonly available parameters and the corresponding `xparameter` values. The default settings listed in the table are used if `xparameter` is not specified.

Parameter Name	`xparameter` **values**
`Pout`, `Phase`, `LS11`, `LS12`, `LS21`, `LS22`	`Pin` (default), `Freq`
`S11`, `S12`, `S21`, `S22`, `NF`, `IIP3`, `OIP3`, `GroupDelay`, `VSWRIn`, `VSWROut`, `GammaIn`, `GammaOut`, `FMIN`, `GammaOPT`, `RN`, `TF1`, `TF2`, `Gt`, `Ga`, `Gp`, `Gmag`, `Gmsg`, `GammaMS`, `GammaML`, `K`, `Delta`, `Mu`, `MuPrime`	`Freq`
`AM/AM`, `AM/PM`	`AM` (default, and only available value)

`xformat` is the format to use for the specified `xparameter`. No `xformat` specification is needed when `xparameter` is an operating condition.

The following table shows the `xformat` values that are available for the `xparameter` values listed in the preceding table, along with the default settings that are used if `xformat` is not specified.

`xparameter` values	`xformat` values
`Pin`	`dBm` (default), `mW`, `W`, `dBW`
`Freq`	`THz`, `GHz`, `MHz`, `KHz`, `Hz` By default, `xformat` is chosen to provide the best scaling for the given `xparameter` values.
`AM`	`Magnitude (decibels)` (default), `Magnitude (linear)`

`condition1,value1,...`, `conditionm,valuem` are the optional condition/value pairs at which to calculate the specified parameters. These pairs are usually operating conditions from a `.p2d` or `.s2d` file. For some parameters, you can specify a set of frequency or input power values at which to calculate the specified parameter.

For example:

- When you calculate large-signal S-parameters as a function of input power, you can specify frequency points of interest using condition/value pairs.
- When you calculate large-signal S-parameters as a function of frequency, you can specify input power levels of interest using condition/value pairs.
- When you calculate parameters as a function of an operating condition, you can specify both frequency and input power values using condition/value pairs.

`freq` is the optional frequency value at which to calculate the specified parameters.

`pin` is the optional input power value at which to calculate the specified parameters.

The method returns the following `n`-element cell arrays:

- `ydata` — The calculated values of the specified parameter.
- `params` — The names, as strings, of the parameters in `xdata` and `ydata`.
- `xdata` — The `xparameter` values at which the specified parameters are known.

Note For compatibility reasons, if `xdata` contains only one vector or if all `xdata` values are equal, then `xdata` is a numeric vector rather than a cell of a single vector.

If `h` has multiple operating conditions, such as from a `.p2d` or `.s2d` file, the `calculate` method operates as follows:

- If you do not specify any operating conditions as arguments to the `calculate` method, then the method returns the parameter values based on the currently selected operating condition.
- If one or more operating conditions are specified, the `calculate` method returns the parameter values based on those operating conditions.
- When an operating condition is used for the `xparameter` input argument, the `xdata` cell array returned by the `calculate` method contains the operating condition values in ascending order.

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 the S_{11} and S_{22} parameters in decibels.

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

```
data =
    [201x1 double]    [201x1 double]
params =
    'S_{11}'    'S_{22}'

freq = 1.0e+009 *

    1.0000
    1.0100
    1.0200
  ...
```

The `params` output is formatted so you can use it as a plot legend. The first few elements of `data{1}` look like

```
ans =

 1.0e+003 *

   -6.4661
   -0.3372
   -0.3432
   -0.3432
   -0.3432
  ...
```

See Also

`analyze` | `extract` | `getz0` | `listformat` | `listparam` | `loglog` | `plot` | `plotyy` | `polar` | `semilogx` | `semilogy` | `smith` | `read` | `restore` | `write`

Purpose

Draw circles on Smith chart

Syntax

```
[hlines, hsm] = circle(h, freq, type1, value1, ..., typen,
   valuen, hsm)
```

Description

`[hlines, hsm] = circle(h, freq, type1, value1, ..., typen, valuen, hsm)` draws the specified circles on a Smith chart and returns the following handles:

- `hlines` — A vector of handles to line objects, with one handle per circle specification.
- `hsm` — The handle to the Smith chart.

The arguments to `circle` are:

- `h` — The handle to an `rfckt` object.
- `freq` — A single frequency point of interest.
- `type1, value1, ..., typen, valuen` — The type/value pairs that specify the circles to plot.

The following table lists the supported circle `type` options and the definition of each option.

`type`	**Definition**
`'Ga'`	Constant available power gain circle
`'Gp'`	Constant operating power gain circle
`'Gt'`	Constant transducer power gain circle
`'Stab'`	Stability circle
`'NF'`	Constant noise figure circle
`'R'`	Constant resistance circle
`'X'`	Constant reactance circle
`'G'`	Constant conductance circle

type	Definition
'B'	Constant susceptance circle
'Gamma'	Constant reflection magnitude circle

The following table lists the available `value` options for the above types of circles and the definition of each value.

value	Definition
'Ga'	Scalar or vector of gains in dB
'Gp'	Scalar or vector of gains in dB
'Gt'	Scalar or vector of gains in dB
'Stab'	String `'in'` or `'source'` for input/source stability circle; string `'out'` or `'load'` for output/load stability circle
'NF'	Scalar or vector of noise figures in dB
'R'	Scalar or vector of normalized resistance
'X'	Scalar or vector of normalized reactance
'G'	Scalar or vector of normalized conductance
'B'	Scalar or vector of normalized susceptance
'Gamma'	Scalar or vector of non-negative reflection magnitude

`hsm` is an optional input argument that you can use to place circles on an existing Smith chart.

Examples

For an example of how to use the `circle` method, see the RF Toolbox example Designing Matching Networks (Part 1: Networks with an LNA and Lumped Elements).

See Also

`smith`

Purpose

Extract array of network parameters from data object

Syntax

```
[outmatrix, freq] = extract(h,outtype,z0)
```

Description

`[outmatrix, freq] = extract(h,outtype,z0)` extracts the network parameters of `outtype` from an `rfckt`, `rfdata.data` or `rfdata.network` object, `h`, and returns them in `outmatrix`. `freq` is a vector of frequencies that correspond to the network parameters.

`outtype` can be one of these case-insensitive strings `'ABCD_parameters'`, `'S_parameters'`, `'Y_parameters'`, `'Z_parameters'`, `'H_parameters'`, `'G_parameters'`, or `'T_parameters'`. `z0` is the reference impedance for the S-parameters. The default is 50 ohms.

See Also

`analyze` | `calculate` | `getz0` | `listformat` | `listparam` | `loglog` | `plot` | `plotyy` | `polar` | `semilogx` | `semilogy` | `smith` | `read` | `restore` | `write`

freqresp

Purpose

Frequency response of model object

Syntax

```
[resp,outfreq] = freqresp(h,infreq)
```

Description

`[resp,outfreq] = freqresp(h,infreq)` computes the frequency response, `resp`, of the `rfmodel` object, `h`, at the frequencies specified by `freq`.

The input `h` is the handle of a model object, and `infreq` is a vector of positive frequencies, in Hz, over which the frequency response is calculated.

The output argument `outfreq` is a vector that contains the same frequencies as the input frequency vector, in order of increasing frequency. The frequency response, `resp`, is a vector of frequency response values corresponding to these frequencies. It is computed using the analytical form of the rational function

$$resp = \left(\sum_{k=1}^{n} \frac{C_k}{s - A_k} + D \right) e^{-s*Delay}, \quad s = j2\pi * freq$$

where `A`, `C`, `D`, and `Delay` are properties of the `rfmodel` object, `h`.

Examples

The following example shows you how to compute the frequency response of the data stored in the file `default.s2p` by reading it into an `rfdata` object, fitting a rational function model to the data, and using the `freqresp` method to compute the frequency response of the model.

```
orig_data=read(rfdata.data,'default.s2p')
freq=orig_data.Freq;
data=orig_data.S_Parameters(2,1,:);
fit_data=rationalfit(freq,data)

[resp,freq]=freqresp(fit_data,freq);

plot(freq/1e9,db(resp));
figure
```

```
plot(freq/1e9,unwrap(angle(resp)));
```

See Also

`rationalfit` | `timeresp` | `writeva`

How To

- `rfmodel.rational`

getop

Purpose

Display operating conditions

Syntax

```
getop(h)
```

Description

`getop(h)` displays the selected operating conditions for the circuit or data object, `h`.

Information about operating conditions is available only when you import the object specifications from a `.p2d` or `.s2d` file.

Examples

Display the operating conditions of an object:

```
ckt1 = read(rfckt.amplifier, 'default.p2d');
getop(ckt1)
```

See Also

`setop`

Purpose

Find spurs in multiband transmitter or receiver frequency space

Syntax

```
spurs = getSpurData(hif)
```

Description

`spurs = getSpurData(hif)` returns a matrix of spur data for the network defined by the `OpenIF` object `hif`. Each spur is a range of frequencies. The two columns of `spurs` contain the endpoints of each spur. The first column contains the lower endpoints, and the second column contains the upper endpoints.

Input Arguments

hif

`OpenIF` object

Examples

- The `OpenIF` class reference page contains an example that shows how to find the spur-free zones of a multiband receiver with three mixers.
- The example Finding Free IF Bandwidths shows how to use information from a spur graph to design a multiband receiver with spur-free zones.

See Also

`addMixer` | `report` | `getSpurFreeZoneData` | `OpenIF`

getSpurFreeZoneData

Purpose

Find spur-free zones in multiband transmitter or receiver frequency space

Syntax

```
zones = getSpurFreeZoneData(hif)
```

Description

`zones = getSpurFreeZoneData(hif)` returns the spur-free zones for the network defined by the `OpenIF` object `hif`. Each zone is a range of IF center frequencies. An IF signal centered in this range does not generate interference in any transmission or reception bands. The two columns of `zones` contain the endpoints of each spur-free zone. The first column contains the lower endpoints, and the second column contains the upper endpoints.

Input Arguments

hif

`OpenIF` object

Alternatives

- The `report` method displays mixer configurations, intermodulation tables, and spur-free zone information at the command line.
- The `show` method generates an interactive spur graph that shows spurious regions and spur-free zones.

Examples

- The `OpenIF` class reference page contains an example that shows how to find the spur-free zones of a multiband receiver with three mixers.
- The example Finding Free IF Bandwidths shows how to use information from a spur graph to design a multiband receiver with spur-free zones.

See Also

`addMixer` | `getSpurData` | `report` | `show` | `OpenIF`

Purpose

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` | `extract` | `listformat` | `listparam` | `loglog` | `plot` | `plotyy` | `polar` | `semilogx` | `semilogy` | `smith` | `read` | `restore` | `write`

impulse

Purpose

Impulse response for model object

Note `impulse` may be removed in a future release. Use `timeresp` instead.

Syntax

```
[resp,t] = impulse(h,ts,n)
```

Description

`[resp,t] = impulse(h,ts,n)` computes the impulse response, `resp`, of the `rfmodel` object, `h`, over the time period specified by `ts` and `n`.

Note While you can compute the output response for a rational function model object by computing the impulse response of the object and then convolving that response with the input signal, this approach is not recommended. Instead, you should use the `timeresp` method to perform this computation because it generally gives a more accurate output signal for a given input signal.

The input `h` is the handle of a rational function model object. `ts` is a positive scalar value that specifies the sample time of the computed impulse response, and `n` is a positive integer that specifies the total number of samples in the response.

The vector of time samples of the impulse response, `t`, is computed from the inputs as `t = [0,ts,2*ts,...,(n-1)*ts]`. The impulse response, `resp`, is an `n`-element vector of impulse response values corresponding to these times. It is computed using the analytical form of the rational function

$$resp = \sum_{k=1}^{M} C_k e^{A_k(t-Delay)} u(t-Delay) + D\delta(t-Delay)$$

where

- A, C, D, and Delay are properties of the rfmodel object, h.
- M is the number of poles in the rational function model.

Examples

The following example shows you how to compute the impulse response of the data stored in the file default.s2p by fitting a rational function model to the data and using the impulse method to compute the impulse response of the model.

```
orig_data=read(rfdata.data,'default.s2p')
freq=orig_data.Freq;
data=orig_data.S_Parameters(2,1,:);
fit_data=rationalfit(freq,data)

[resp,t]=impulse(fit_data,1e-12,1e4);

plot(t,resp);
```

See Also

freqresp | rationalfit | writeva

How To

- rfmodel.rational

ispassive

Purpose

Check passivity of scalar model object

Syntax

```
result = ispassive(h)
```

Description

`result = ispassive(h)` checks the passivity of the `rfmodel` object, `h`, across all frequencies, and returns `result`, a logical value. If `h` is passive, then `result` is `1`. If `h` is not passive, then `result` is `0`.

Examples

Create a scalar model object and check the passivity of the object:

```
% Read a Touchstone data file
ckt = read(rfckt.passive, 'passive.s2p');
% Fit the transfer function into a rational function object
TF = s2tf(ckt.AnalyzedResult.S_Parameters);
TF_Object = rationalfit(ckt.AnalyzedResult.Freq, TF);
% Check the passivity of the rational function object
Is_Passive = ispassive(TF_Object)
```

See Also

`rfmodel.rational` | `rationalfit`

Purpose

List valid formats for 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)'`.

When you plot phase information as a function of frequency, RF Toolbox software unwraps the phase data using the MATLAB `unwrap` function. The resulting plot is only meaningful if the phase data varies smoothly as a function of frequency, as described in the `unwrap` reference page. If your data does not meet this requirement, you must obtain data on a finer frequency grid.

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

Note Before calling `listformat`, you must use the `analyze` method 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 | extract | getz0 | listparam | loglog | plot | plotyy | polar | semilogx | semilogy | smith | read | restore | write

Purpose

List valid parameters for 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` method to perform a frequency domain analysis for the circuit object.

Several parameters are available for all objects. When you import `rfckt.amplifier`, `rfckt.mixer`, or `rfdata.data` object specifications from a `.p2d` or `.s2d` file, the list of valid parameters also includes any operating conditions from the file that have numeric values, such as bias.

The following table describes the most commonly available parameters.

Parameter	Description
`S11`, `S12`, `S21`, `S22` `LS11`, `LS12`, `LS21`, `LS22` (Amplifier and mixer objects with multiple operating conditions only)	S-parameters
`GroupDelay`	Group delay
`GammaIn`, `GammaOut`	Input and output reflection coefficients
`VSWRIn`, `VSWROut`	Input and output voltage standing-wave ratio
`IIP3`, `OIP3` (Amplifier and mixer objects only)	Third-order intercept point
`NF`	Noise figure

Parameter	Description
TF1	Ratio of the load voltage to the output voltage of the source when the input port is conjugate matched
TF2	Ratio of load voltage to the source voltage
• Gt • Ga • Gp • Gmag • Gmsg	• Transducer power gain • Available power gain • Operating power gain • Maximum available power gain • Maximum stable gain
GammaMS, GammaML	Source and load reflection coefficients for simultaneous conjugate match
K, Mu, MuPrime	Stability factor
Delta	Stability condition

Examples

The following examples show you how to list the parameters for a transmission line object.

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

See Also

analyze | calculate | extract | getz0 | listformat | loglog | plot | plotyy | polar | semilogx | semilogy | smith | read | restore | write

Purpose

Plot specified circuit object parameters using log-log scale

Syntax

```
lineseries = loglog(h,parameter)
lineseries = loglog(h,parameter1,...,parametern)
lineseries = loglog(h,parameter1,...,parametern,format)
lineseries = loglog(h,'parameter1',...,'parametern', format,
   xparameter,xformat,'condition1',value1,..., 'conditionm',
   valuem,'freq',freq,'pin',pin)
```

Description

`lineseries = loglog(h,parameter)` plots the specified `parameter` in the default format using a log-log scale. `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 `loglog` method returns a column vector of handles to `lineseries` objects, one handle per line. This output is the same as the output returned by the MATLAB `loglog` method.

`lineseries = loglog(h,parameter1,...,parametern)` plots the parameters `parameter1`,..., `parametern` from the object `h` on an X-Y plane using logarithmic scales for both the *x*- and *y*- axes.

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

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

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 Use the MATLAB `loglog` function to create a log-log scale plot of parameters that are specified as vector data and are not part of a circuit (`rfckt`) object or data (`rfdata`) object.

`lineseries = loglog(h,'parameter1',...,'parametern', format,xparameter,xformat,'condition1',value1,..., 'conditionm',valuem,'freq',freq,'pin',pin)` plots the specified parameters at the specified operating conditions for the object `h`.

`xparameter` is the independent variable to use in plotting the specified parameters. Several `xparameter` values are available for all objects. When you import `rfckt.amplifier`, `rfckt.mixer`, or `rfdata.data` object specifications from a `.p2d` or `.s2d` file, you can also specify any operating conditions from the file that have numeric values, such as bias.

The following table shows the most commonly available parameters and the corresponding `xparameter` values. The default settings listed in the table are used if `xparameter` is not specified.

Parameter Name	`xparameter` values
`Pout`, `Phase`, `LS11`, `LS12`, `LS21`, `LS22`	`Pin` (default), `Freq`
`S11`, `S12`, `S21`, `S22`, `NF`, `IIP3`, `OIP3`, `GroupDelay`, `VSWRIn`, `VSWROut`, `GammaIn`, `GammaOut`, `FMIN`, `GammaOPT`, `RN`, `TF1`, `TF2`, `Gt`, `Ga`, `Gp`, `Gmag`, `Gmsg`, `GammaMS`, `GammaML`, `K`, `Delta`, `Mu`, `MuPrime`	`Freq`
`AM/AM`, `AM/PM`	`AM`

`xformat` is the format to use for the specified `xparameter`. No `xformat` specification is needed when `xparameter` is an operating condition.

The following table shows the `xformat` values that are available for the `xparameter` values listed in the preceding table, along with the default settings that are used if `xformat` is not specified.

`xparameter` **values**	`xformat` **values**
`Pin`	`dBm` (default), `mW`, `W`, `dBW`
`Freq`	`THz`, `GHz`, `MHz`, `KHz`, `Hz` By default, `xformat` is chosen to provide the best scaling for the given `xparameter` values.
`AM`	`Magnitude (decibels)` (default), `Magnitude (linear)`

`condition1,value1,...`, `conditionm,valuem` are the optional condition/value pairs at which to plot the specified parameters. These pairs are usually operating conditions from a `.p2d` or `.s2d` file. For some parameters, you can specify a set of frequency or input power values at which to plot the specified parameter.

For example:

- When plotting large-signal S-parameters as a function of input power, you can specify frequency points of interest using condition/value pairs.
- When plotting large-signal S-parameters as a function of frequency, you can specify input power levels of interest using condition/value pairs.
- When plotting parameters as a function of an operating condition, you can specify both frequency and input power values using condition/value pairs.

`freq` is the optional frequency value, in hertz, at which to plot the specified parameters.

`pin` is the optional input power value, in dBm, at which to plot the specified parameters.

If `h` has multiple operating conditions, such as from a `.p2d` or `.s2d` file, the `loglog` method operates as follows:

- If you do not specify any operating conditions as arguments to the `loglog` method, then the method plots the parameter values based on the currently selected operating condition.
- If you specify one or more operating conditions, the `loglog` method plots the parameter values based on those operating conditions.
- When you use an operating condition for the `xparameter` input argument, the method plots the parameters for all operating condition values.

See Also

`analyze` | `calculate` | `extract` | `getz0` | `listformat` | `listparam` | `plot` | `plotyy` | `polar` | `read` | `restore` | `semilogx` | `semilogy` | `smith` | `write`

Purpose

Plot specified circuit object parameters on X-Y plane

Syntax

```
lineseries = plot(h,parameter)
lineseries = plot(h,parameter1,...,parametern)
lineseries = plot(h,parameter1,...,parametern,format)
lineseries = plot(h,'parameter1',...,'parametern', format,
   xparameter,xformat, 'condition1',value1,..., 'conditionm',
   valuem,'freq',freq,'pin',pin)
lineseries = plot(h,'budget',...)
lineseries = plot(h,'mixerspur',k,pin,fin)
```

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. Use the `listparam` method to get a list of the valid parameters for a particular circuit object, `h`.

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

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

`lineseries = plot(h,parameter1,...,parametern,format)` plots the specified parameters `parameter1,...,parametern` in the specified `format`. The format determines if RF Toolbox software converts the parameter values to a new set of units, or operates on the components of complex parameter values. For example:

- Specify `format` as `Real` to plot the real part of the selected parameter.
- Specify `format` as `'none'` to plot the parameter values unchanged.

Use the `listformat` method to get a list of the valid formats for a particular parameter.

`lineseries = plot(h,'parameter1',...,'parametern',format,xparameter,xformat, 'condition1',value1,...,'conditionm',valuem,'freq',freq,'pin',pin)` plots the specified parameters at the specified operating conditions for the object `h`.

`xparameter` is the independent variable to use in plotting the specified parameters. Several `xparameter` values are available for all objects. When you import `rfckt.amplifier`, `rfckt.mixer`, or `rfdata.data` object specifications from a `.p2d` or `.s2d` file, you can also specify any operating conditions from the file that have numeric values, such as bias.

The following table shows the most commonly available parameters and the corresponding `xparameter` values. The default settings listed in the table are used if `xparameter` is not specified.

Parameter Name	`xparameter` values
`Pout`, `Phase`, `LS11`, `LS12`, `LS21`, `LS22`	`Pin` (default), `Freq`
`S11`, `S12`, `S21`, `S22`, `NF`, `IIP3`, `OIP3`, `GroupDelay`, `VSWRIn`, `VSWROut`, `GammaIn`, `GammaOut`, `FMIN`, `GammaOPT`, `RN`, `TF1`, `TF2`, `Gt`, `Ga`, `Gp`, `Gmag`, `Gmsg`, `GammaMS`, `GammaML`, `K`, `Delta`, `Mu`, `MuPrime`	`Freq`
`AM/AM`, `AM/PM`	`AM`

`xformat` is the format to use for the specified `xparameter`. No `xformat` specification is needed when `xparameter` is an operating condition.

The following table shows the `xformat` values that are available for the `xparameter` values listed in the preceding table, along with the default settings that are used if `xformat` is not specified.

xparameter values	xformat values
Pin	dBm (default), mW, W, dBW
Freq	THz, GHz, MHz, KHz, Hz By default, xformat is chosen to provide the best scaling for the given xparameter values.
AM	Magnitude (decibels) (default), Magnitude (linear)

condition1,value1,..., conditionm,valuem are the optional condition/value pairs at which to plot the specified parameters. These pairs are usually operating conditions from a .p2d or .s2d file. For some parameters, you can specify a set of frequency or input power values at which to plot the specified parameter.

For example:

- When plotting large-signal S-parameters as a function of input power, you can specify frequency points of interest using condition/value pairs.
- When plotting large-signal S-parameters as a function of frequency, you can specify input power levels of interest using condition/value pairs.
- When plotting parameters as a function of an operating condition, you can specify both frequency and input power values using condition/value pairs.

freq is the optional frequency value, in hertz, at which to plot the specified parameters.

pin is the optional input power value, in dBm, at which to plot the specified parameters.

If h has multiple operating conditions, such as from a .p2d or .s2d file, the plot method operates as follows:

- If you do not specify any operating conditions as arguments to the `plot` method, then the method plots the parameter values based on the currently selected operating condition.
- If you specify one or more operating conditions, the `plot` method plots the parameter values based on those operating conditions.
- When you use an operating condition for the `xparameter` input argument, the method plots the parameters for all operating condition values.

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

The following table summarizes the parameters and formats that are available for a budget plot.

Parameter	Format
S11, S12, S21, S22	Magnitude (decibels) Magnitude (linear) Angle (degrees) Real Imaginary
OIP3	dBm dBW W mW
NF	Magnitude (decibels) Magnitude (linear)

`lineseries = plot(h,'mixerspur',k,pin,fin)` plots spur power of an `rfckt.mixer` object or an `rfckt.cascade` object that contains one or more mixers.

`k` is the index of the circuit object for which to plot spur power. Its value can be an integer or `'all'`. The default is `'all'`. This value creates a budget plot of the spur power for `h`. Use `0` to plot the power at the input of `h`.

`pin` is the optional scalar input power value, in dBm, at which to plot the spur power. The default is `0` dBm. When you create a spur plot for an object, the previous input power value is used for subsequent plots until you specify a different value.

`fin` is the optional scalar input frequency value, in hertz, at which to plot the spur power. If `h` is an `rfckt.mixer` object, the default value of `fin` is the input frequency at which the magnitude of the S_{21} parameter of the mixer, in decibels, is highest. If `h` is an `rfckt.cascde` object, the default value of `fin` is the input frequency at which the magnitude of the S_{21} parameter of the first mixer in the cascade is highest. When you create a spur plot for an object, the previous input frequency value is used for subsequent plots until you specify a different value.

For more information on plotting mixer spur power, see the Visualizing Mixer Spurs example.

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

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 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.

Alternatives

The `rfplot` function creates magnitude-frequency plots for RF Toolbox S-parameter objects.

See Also

`analyze` | `calculate` | `extract` | `getz0` | `listformat` | `listparam` | `loglog` | `plotyy` | `polar` | `read` | `restore` | `rfplot` | `semilogx` | `semilogy` | `smith` | `write`

Purpose

Plot specified object parameters with *y*-axes on both left and right sides

Syntax

```
[ax,hlines1,hlines2] = plotyy(h,parameter)
[ax,hlines1,hlines2] = plotyy(h,parameter1,...,parametern)
[ax,hlines1,hlines2] = plotyy(h,parameter,format1,format2)
[ax,hlines1,hlines2] = plotyy(h, parameter1, ..., parametern,
   format1, format2)
[ax,hlines1,hlines2] = plotyy(h,parameter1_1,...,
   parameter1_n1, format1,parameter2_1,...,parameter2_n2,
   format2)
[ax,hlines1,hlines2] = plotyy(h,parameter1_1,...,parameter1_n1,
   format1,parameter2_1,...,parameter2_n2,format2,xparameter,
   xformat,'condition1',value1,...,'conditionm',valuem,
   'freq',freq,'pin',pin)
```

Description

`[ax,hlines1,hlines2] = plotyy(h,parameter)` plots the specified `parameter` using the predefined primary and secondary formats for the left and right *y*-axes, respectively. The formats define how RF Toolbox software displays the data on the plot. `h` is the handle of a circuit (`rfckt`) or an `rfdata.data` object.

- See "Determining Formats" on page 7-43 for a table that shows the predefined primary and secondary formats for the parameters for all circuit and data objects.
- 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 `plotyy` method returns the handles to the two axes created in `ax` and the handles to two `lineseries` objects in `hlines1` and `hlines2`.

- `ax(1)` is the left axes.
- `ax(2)` is the right axes.
- `hlines1` is the `lineseries` object for the left *y*-axis.
- `hlines2` is the `lineseries` object for the right *y*-axis.

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

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 Use the MATLAB `plotyy` function to plot parameters on two *y*-axes that are specified as vector data and are not part of a circuit (`rfckt`) object or data (`rfdata`) object.

`[ax,hlines1,hlines2] = plotyy(h,parameter1,...,parametern)` plots the parameters `parameter1,..., parametern`. `plotyy` determines the formats for the left and right *y*-axes based on the predefined primary and secondary formats for the specified parameters, as described in "Determining Formats" on page 7-43.

`[ax,hlines1,hlines2] = plotyy(h,parameter,format1,format2)` plots the specified `parameter` using `format1` for the left *y*-axis and `format2` for the right *y*-axis.

`[ax,hlines1,hlines2] = plotyy(h, parameter1, ..., parametern, format1, format2)` plots the parameters`parameter1,..., parametern` on an X-Y plane using `format1` for the left *y*-axis and `format2` for the right *y*-axis.

`[ax,hlines1,hlines2] = plotyy(h,parameter1_1,...,parameter1_n1, format1,parameter2_1,...,parameter2_n2,format2)` plots the following data:

- Parameters parameter1_1,..., parameter1_n1 using format1 for the left *y*-axis.
- Parameters parameter2_1,..., parameter2_n2 using format2 for the right *y*-axis.

[ax,hlines1,hlines2] = plotyy(h,parameter1_1,...,parameter1_n1, format1,parameter2_1,...,parameter2_n2,format2,xparameter, xformat,'condition1',value1,...,'conditionm',valuem, 'freq',freq,'pin',pin) plots the specified parameters at the specified operating conditions for the object h.

xparameter is the independent variable to use in plotting the specified parameters. Several xparameter values are available for all objects. When you import rfckt.amplifier, rfckt.mixer, or rfdata.data object specifications from a .p2d or .s2d file, you can also specify any operating conditions from the file that have numeric values, such as bias.

The following table shows the most commonly available parameters and the corresponding xparameter values. The default settings listed in the table are used if xparameter is not specified.

Parameter Name	xparameter values
Pout, Phase, LS11, LS12, LS21, LS22	Pin (default), Freq
S11, S12, S21, S22, NF, IIP3, OIP3, GroupDelay, VSWRIn, VSWROut, GammaIn, GammaOut, FMIN, GammaOPT, RN, TF1, TF2, Gt, Ga, Gp, Gmag, Gmsg, GammaMS, GammaML, K, Delta, Mu, MuPrime	Freq
AM/AM, AM/PM	AM

xformat is the format to use for the specified xparameter. No xformat specification is needed when xparameter is an operating condition.

The following table shows the xformat values that are available for the xparameter values listed in the preceding table, along with the default settings that are used if xformat is not specified.

xparameter **values**	xformat **values**
Pin	dBm (default), mW, W, dBW
Freq	THz, GHz, MHz, KHz, Hz By default, xformat is chosen to provide the best scaling for the given xparameter values.
AM	Magnitude (decibels) (default), Magnitude (linear)

condition1,value1,..., conditionm,valuem are the optional condition/value pairs at which to plot the specified parameters. These pairs are usually operating conditions from a .p2d or .s2d file. For some parameters, you can specify a set of frequency or input power values at which to plot the specified parameter.

For example:

- When plotting large-signal S-parameters as a function of input power, you can specify frequency points of interest using condition/value pairs.
- When plotting large-signal S-parameters as a function of frequency, you can specify input power levels of interest using condition/value pairs.
- When plotting parameters as a function of an operating condition, you can specify both frequency and input power values using condition/value pairs.

freq is the optional frequency value, in hertz, at which to plot the specified parameters.

pin is the optional input power value, in dBm, at which to plot the specified parameters.

If `h` has multiple operating conditions, such as from a `.p2d` or `.s2d` file, the `plotyy` method operates as follows:

- If you do not specify any operating conditions as arguments to the `plotyy` method, then the method plots the parameter values based on the currently selected operating condition.
- If you specify one or more operating conditions, the `plotyy` method plots the parameter values based on those operating conditions.
- When you use an operating condition for the `xparameter` input argument, the method plots the parameters for all operating condition values.

Determining Formats

When you call `plotyy` without specifying the plot formats for the left and right *y*-axes, `plotyy` determines the formats from the predefined primary and secondary formats for the one or more specified parameters.

This section contains the following topics:

- "Primary and Secondary Formats" on page 7-43
- "Determining Formats for One Parameter" on page 7-45
- "Determining Formats for Multiple Parameters" on page 7-45

Primary and Secondary Formats

The following table shows the primary and secondary formats for the parameters for all circuit and data objects. Use the `listparam` method to list the valid parameters for a particular object. Use the `listformat` method to list valid formats.

Parameter	Primary Format	Secondary Format
S11, S12, S21, S22	Magnitude (decibels)	Angle (Degrees)
LS11, LS12, LS21, LS22	Magnitude (decibels)	Angle (Degrees)

Parameter	Primary Format	Secondary Format
NF	Magnitude (decibels)	
OIP3	dBm	W
Pout	dBm	W
Phase	Angle (Degrees)	
AM/AM	Magnitude (decibels)	
AM/PM	Angle (Degrees)	
GammaIn, GammaOut	Magnitude (decibels)	Angle (Degrees)
Gt, Ga, Gp, Gmag, Gmsg	Magnitude (decibels)	
Delta	Magnitude (decibels)	Angle (Degrees)
TF1, TF2	Magnitude (decibels)	Angle (Degrees)
GammaMS, GammaML	Magnitude (decibels)	Angle (Degrees)
VSWRIn, VSWROut	Magnitude (decibels)	
GroupDelay	ns	
Fmin	Magnitude (decibels)	
GammaOPT	Magnitude (decibels)	Angle (Degrees)
K, Mu, MuPrime	None	
RN	None	
PhaseNoise	dBc/Hz	

Parameter	Primary Format	Secondary Format
NTemp	K	
NFactor	None	

Determining Formats for One Parameter

When you specify only one parameter for plotting, `plotyy` creates the plot as follows:

- The predefined primary format is the format for the left *y*-axis.
- The predefined secondary format is the format for the right *y*-axis.

If the specified parameter does not have the predefined secondary format, `plotyy` behaves the same way as `plot`, and does not add a second y-axis to the plot.

Determining Formats for Multiple Parameters

To plot multiple parameters on two *y*-axes, `plotyy` tries to find two formats from the predefined primary and secondary formats for the specified parameters. To be used in the plot, the formats must meet the following criteria:

- Each format must be a valid format for at least one parameter.
- Each parameter must be plotted at least on one *y*-axis.

If `plotyy` cannot meet this criteria it issues an error message.

The function uses the following algorithm to determine the two parameters:

1. Look up the primary and secondary formats for the specified parameters.

2. If one or more pairs of primary-secondary formats meets the preceding criteria for all parameters:

   - Select the pair that applies to the most parameters.
   - Use these formats to create the plot.

   Otherwise, proceed to the next step.

3 If no pairs of primary-secondary formats meet the criteria for all parameters, try to find one or more pairs of primary-primary formats that meets the criteria. If one or more pairs of primary-primary formats meets the preceding criteria for all parameters:

- Select the pair that applies to the most parameters.
- Use these formats to create the plot.

Otherwise, proceed to the next step.

4 If the preceding steps fail to produce a plot, try to find one format from the predefined primary formats. If a primary format is valid for all parameters, use this format to create the plot with the MATLAB `plot` function.

If this is not successful, issue an error message.

The following example shows how `plotyy` applies this criteria to create plots.

Example — Determining Formats for Multiple Parameters

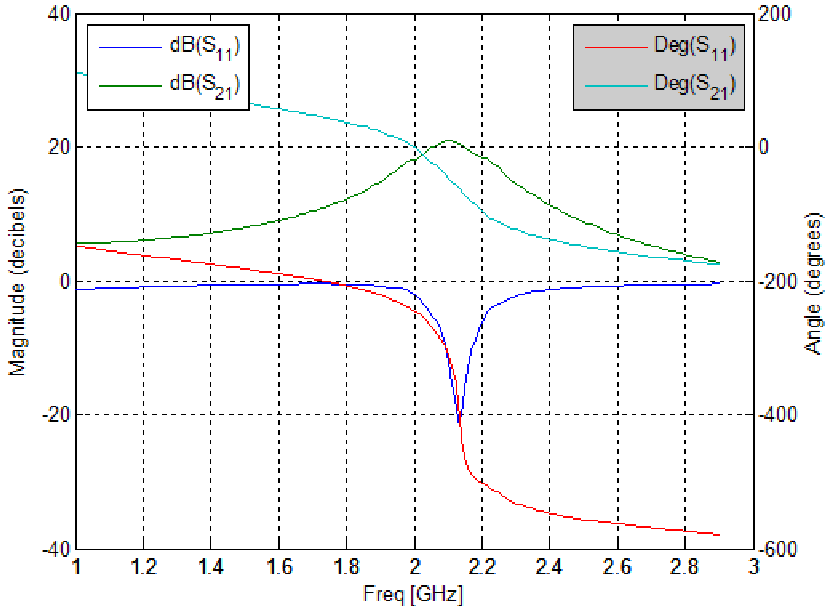
At the MATLAB prompt:

1 Type this command to create an `rfckt` object called `amp`:

```
amp = rfckt.amplifier;
```

2 Type this command to plot the `S11` and `S21` parameters of `amp` on two *y*-axes:

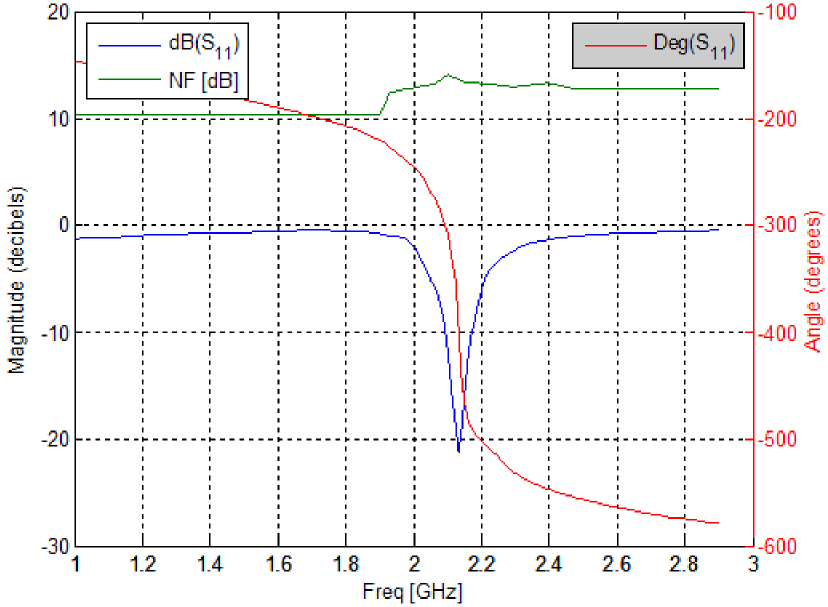
```
plotyy(amp, 'S11', 'S21')
```



The primary and secondary formats for both `S11` and `S21` are `Magnitude (decibels)` and `Angle (Degrees)`, respectively, so `plotyy` uses this primary-secondary format pair to create the plot.

3 Type this command to plot the `S11` and `NF` parameters of `amp` on two *y*-axes:

```
plotyy(amp, 'S11', 'NF')
```



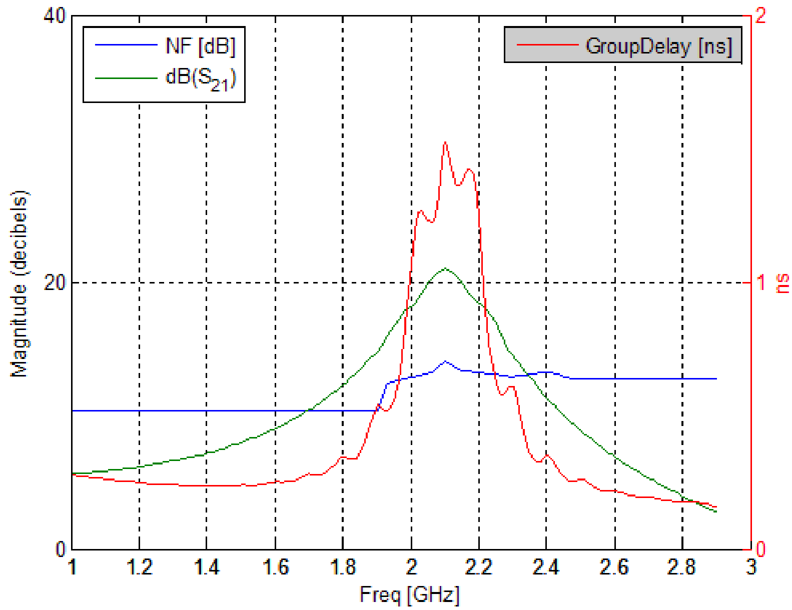
The primary and secondary formats for `S11` are `Magnitude (decibels)` and `Angle (Degrees)`, respectively.

- `Magnitude (decibels)` is a valid format for both `S11` and `NF`
- `Angle (Degrees)` is a valid format for `S11`.

These formats both meet the preceding criteria, so the function uses this primary-secondary format pair to create the plot.

4 Type this command to plot the `NF`, `S21` and `GroupDelay` parameters of `amp` on two *y*-axes:

```
plotyy(amp, 'NF', 'S21', 'GroupDelay')
```



The primary and secondary formats for `S21` are `Magnitude (decibels)` and `Angle (Degrees)`, respectively. Both `NF` and `GroupDelay` have only a primary format.

- `Magnitude (decibels)` is the primary format for `NF`.
- `ns` is the primary format for `GroupDelay`.

There is no primary-secondary format pair that meets the preceding criteria, so `plotyy` tries to find a pair of primary formats that meet the criteria. `plotyy` creates the plot using:

- `Magnitude (decibels)` for the left `y`-axis.

  This format is valid for both `NF` and `S21`.

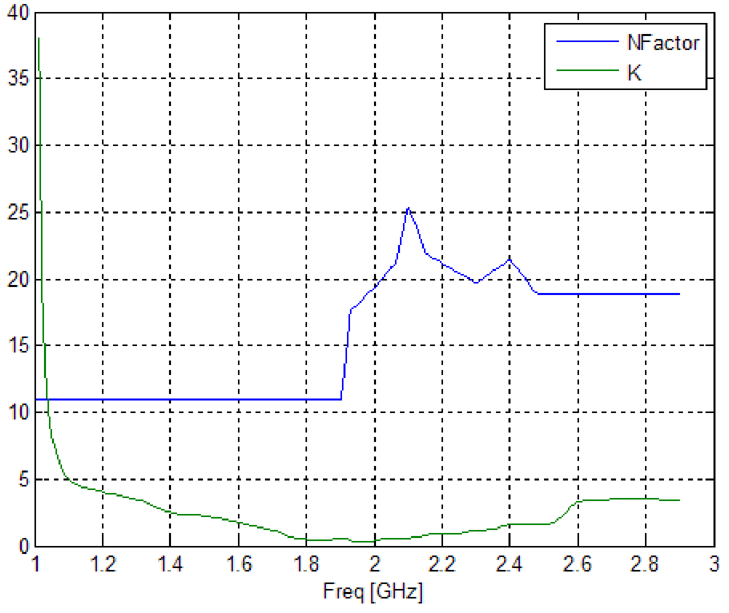
- `ns` for the right `y`-axis.

  This format is valid for `GroupDelay` .

These formats meet the criteria.

5 Type this command to plot the `NFactor` and `K` parameters of `amp` on two *y*-axes:

```
plotyy(amp, 'NFactor', 'K')
```



Both `NFactor` and `K` have only a primary format,`None`, so `plotyy` calls the `plot` command to create a plot with a single *y*-axis whose format is `None`.

6 Type this command to plot the `NTemp`, `S21` and `NFactor` parameters of `amp` on two *y*-axes:

```
plotyy(amp, 'NTemp', 'S21', 'NFactor')

??? Error using ==> rfdata.data.plotyyprocess at 97
No format specified for input parameters and cannot reconcile
default formats. Try reducing the number of parameters to plotyy
and explicitly specifying formats.
```

The primary and secondary formats for `S21` are `Magnitude (decibels)` and `Angle (Degrees)`, respectively. Both `NTemp` and `NFactor` have only a primary format.

- `Kelvin` is the primary format for `NTemp`.
- `None` is the primary format for `NFactor`.

These parameters have no formats in common, so no formats meet the criteria and `plotyy` issues an error message.

See Also

`analyze` | `calculate` | `extract` | `getz0` | `listformat` | `listparam` | `loglog` | `plot` | `polar` | `read` | `restore` | `semilogx` | `semilogy` | `smith` | `write`

Purpose Plot specified circuit object parameters on polar coordinates

Syntax

```
lineseries = polar(h,'parameter1',...,'parametern')
lineseries = polar(h,'parameter1',...,'parametern', xparameter,
   xformat,'condition1',value1,..., 'conditionm',valuem,
   'freq',freq,'pin',pin)
```

Description

`lineseries = polar(h,'parameter1',...,'parametern')` plots the parameters `parameter1,...`, `parametern` from the object `h` on polar coordinates. `h` is the handle of a circuit (`rfckt`) object.

`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.

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` method to perform a frequency domain analysis before calling `polar`.

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

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

`lineseries = polar(h,'parameter1',...,'parametern', xparameter,xformat,'condition1',value1,..., 'conditionm',valuem, 'freq',freq,'pin',pin)` plots the specified parameters at the specified operating conditions for the object `h`.

xparameter is the independent variable to use in plotting the specified parameters. Several xparameter values are available for all objects. When you import rfckt.amplifier, rfckt.mixer, or rfdata.data object specifications from a .p2d or .s2d file, you can also specify any operating conditions from the file that have numeric values, such as bias.

The following table shows the most commonly available parameters and the corresponding xparameter values. The default settings listed in the table are used if xparameter is not specified.

Parameter Name	xparameter values
Pout, Phase, LS11, LS12, LS21, LS22	Pin (default), Freq
S11, S12, S21, S22, NF, OIP3, VSWRIn, VSWROut, GammaIn, GammaOut, FMIN, GammaOPT, RN	Freq
AM/AM, AM/PM	AM

xformat is the format to use for the specified xparameter. No xformat specification is needed when xparameter is an operating condition.

The following table shows the xformat values that are available for the xparameter values listed in the preceding table, along with the default settings that are used if xformat is not specified.

xparameter values	xformat values
Pin	dBm (default), mW, W, dBW
Freq	THz, GHz, MHz, KHz, Hz By default, xformat is chosen to provide the best scaling for the given xparameter values.
AM	Magnitude (decibels) (default), Magnitude (linear)

`condition1,value1,..., conditionm,valuem` are the optional condition/value pairs at which to plot the specified parameters. These pairs are usually operating conditions from a `.p2d` or `.s2d` file. For some parameters, you can specify a set of frequency or input power values at which to plot the specified parameter.

For example:

- When plotting large-signal S-parameters as a function of input power, you can specify frequency points of interest using condition/value pairs.
- When plotting large-signal S-parameters as a function of frequency, you can specify input power levels of interest using condition/value pairs.
- When plotting parameters as a function of an operating condition, you can specify both frequency and input power values using condition/value pairs.

`freq` is the optional frequency value, in hertz, at which to plot the specified parameters.

`pin` is the optional input power value, in dBm, at which to plot the specified parameters.

If `h` has multiple operating conditions, such as from a `.p2d` or `.s2d` file, the `polar` method operates as follows:

- If you do not specify any operating conditions as arguments to the `polar` method, then the method plots the parameter values based on the currently selected operating condition.
- If you specify one or more operating conditions, the `polar` method plots the parameter values based on those operating conditions.
- When you use an operating condition for the `xparameter` input argument, the method plots the parameters for all operating condition values.

See Also

analyze | calculate | extract | getz0 | listformat | listparam | loglog | plot | plotyy | read | restore | semilogx | semilogy | smith | 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 file and then reads the data from that file into the circuit or data object, `h`. You can read data from an `.snp`, `.ynp`, `.znp`, `.hnp`, `.gnp`, or `.amp` file, where `n` is the number of ports. If `h` is an `rfckt.amplifier`, `rfckt.mixer`, or `rfdata.data` object, you can also read data from `.p2d` and `.s2d` files.

For an example of how to use RF Toolbox software to read data from a `.s2d` file, see Visualizing Mixer Spurs.

`h = read(h,filename)` updates `h` with data from the specified file. In this syntax, `h` can be a circuit or data object. `filename` is a string, representing the filename of a `.snp`, `.ynp`, `.znp`, `.hnp`, `.gnp`, or `.amp` file. If `h` is an `rfckt.amplifier`, `rfckt.mixer`, or `rfdata.data` object, `filename` can also represent a `.p2d` or `.s2d` file. For all files, 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`.

Examples

The following example shows you how to import data from the file `default.amp` into an `rfckt.amplifier` object.

```
ckt_obj=read(rfckt.amplifier, 'default.amp');
```

References

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

See Also

`analyze` | `calculate` | `extract` | `getz0` | `listformat` | `listparam` | `loglog` | `plot` | `plotyy` | `polar` | `restore` | `semilogx` | `semilogy` | `smith` | `write`

report

Purpose

Summarize mixer configurations and spur-free-zone information for a multiband transmitter or receiver

Syntax

```
report(hif)
```

Description

`report(hif)` displays a summary at the command line of the information contained in the `OpenIF` object `hif`. The summary contains:

- The IF location.
- The properties of each mixer, including the RF center frequencies, bandwidths, mixing types, and intermodulation tables.
- The spur-free zones.

Each spur-free zone is a range of IF center frequencies. An IF signal centered in this range does not generate interference in any transmission or reception bands.

Input Arguments

hif

`OpenIF` object

Alternatives

- The `getSpurFreeZoneData` returns the endpoints of the spur-free zones in a matrix.
- The `show` method generates an interactive spur graph that shows spurious regions and spur-free zones.

Examples

For an IF-planning example using an `OpenIF` object, see the “Examples” on page 6-6 section of the `OpenIF` class reference page.

See Also

`addMixer` | `getSpurData` | `getSpurFreeZoneData` | `show` | `OpenIF`

Purpose

Restore data to 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` | `extract` | `getz0` | `listformat` | `listparam` | `loglog` | `plot` | `plotyy` | `polar` | `semilogx` | `semilogy` | `smith` | `read` | `write`

semilogx

Purpose

Plot specified circuit object parameters using log scale for *x*-axis

Syntax

```
lineseries = semilogx(h,parameter)
lineseries = semilogx(h,parameter1,...,parametern)
lineseries = semilogx(h,parameter1,...,parametern,format)
lineseries = semilogx(h,'parameter1',...,'parametern', format,
   xparameter,xformat,'condition1',value1,..., 'conditionm',
   valuem, 'freq',freq,'pin',pin)
```

Description

`lineseries = semilogx(h,parameter)` plots the specified `parameter` in the default format using a logarithmic scale for the *x*-axis. `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 `semilogx` method returns a column vector of handles to `lineseries` objects, one handle per line. This output is the same as the output returned by the MATLAB `semilogx` function.

`lineseries = semilogx(h,parameter1,...,parametern)` plots the parameters `parameter1,...,` `parametern` from the object `h` on an X-Y plane using a logarithmic scale for the *x*-axis.

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

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

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 Use the MATLAB `semilogx` function to create a semi-log scale plot of network parameters that are specified as vector data and are not part of a circuit (`rfckt`) object or data (`rfdata`) object.

`lineseries = semilogx(h,'parameter1',...,'parametern', format,xparameter,xformat,'condition1',value1,..., 'conditionm',valuem, 'freq',freq,'pin',pin)` plots the specified parameters at the specified operating conditions for the object `h`.

`xparameter` is the independent variable to use in plotting the specified parameters. Several `xparameter` values are available for all objects. When you import `rfckt.amplifier`, `rfckt.mixer`, or `rfdata.data` object specifications from a `.p2d` or `.s2d` file, you can also specify any operating conditions from the file that have numeric values, such as bias.

The following table shows the most commonly available parameters and the corresponding `xparameter` values. The default settings listed in the table are used if `xparameter` is not specified.

Parameter Name	`xparameter` values
`Pout`, `Phase`, `LS11`, `LS12`, `LS21`, `LS22`	`Pin` (default), `Freq`
`S11`, `S12`, `S21`, `S22`, `NF`, `IIP3`, `OIP3`, `GroupDelay`, `VSWRIn`, `VSWROut`, `GammaIn`, `GammaOut`, `FMIN`, `GammaOPT`, `RN`, `TF1`, `TF2`, `Gt`, `Ga`, `Gp`, `Gmag`, `Gmsg`, `GammaMS`, `GammaML`, `K`, `Delta`, `Mu`, `MuPrime`	`Freq`
`AM/AM`, `AM/PM`	`AM`

`xformat` is the format to use for the specified `xparameter`. No `xformat` specification is needed when `xparameter` is an operating condition.

The following table shows the `xformat` values that are available for the `xparameter` values listed in the preceding table, along with the default settings that are used if `xformat` is not specified.

`xparameter` values	`xformat` values
`Pin`	`dBm` (default), `mW`, `W`, `dBW`
`Freq`	`THz`, `GHz`, `MHz`, `KHz`, `Hz` By default, `xformat` is chosen to provide the best scaling for the given `xparameter` values.
`AM`	`Magnitude (decibels)` (default), `Magnitude (linear)`

`condition1,value1,...`, `conditionm,valuem` are the optional condition/value pairs at which to plot the specified parameters. These pairs are usually operating conditions from a `.p2d` or `.s2d` file. For some parameters, you can specify a set of frequency or input power values at which to plot the specified parameter.

For example:

- When plotting large-signal S-parameters as a function of input power, you can specify frequency points of interest using condition/value pairs.
- When plotting large-signal S-parameters as a function of frequency, you can specify input power levels of interest using condition/value pairs.
- When plotting parameters as a function of an operating condition, you can specify both frequency and input power values using condition/value pairs.

`freq` is the optional frequency value, in hertz, at which to plot the specified parameters.

`pin` is the optional input power value, in dBm, at which to plot the specified parameters.

If `h` has multiple operating conditions, such as from a `.p2d` or `.s2d` file, the `semilogx` method operates as follows:

- If you do not specify any operating conditions as arguments to the `semilogx` method, then the method plots the parameter values based on the currently selected operating condition.
- If you specify one or more operating conditions, the `semilogx` method plots the parameter values based on those operating conditions.
- When you use an operating condition for the `xparameter` input argument, the method plots the parameters for all operating condition values.

See Also

`analyze` | `calculate` | `extract` | `getz0` | `listformat` | `listparam` | `loglog` | `plot` | `plotyy` | `polar` | `read` | `restore` | `semilogy` | `smith` | `write`

semilogy

Purpose

Plot specified circuit object parameters using log scale for x-axis

Syntax

```
lineseries = semilogy(h,parameter)
lineseries = semilogy(h,parameter1,...,parametern)
lineseries = semilogy(h,parameter1,...,parametern,format)
lineseries = semilogy(h,'parameter1',...,'parametern', format,
   xparameter,xformat,'condition1',value1,..., 'conditionm',
   valuem, 'freq',freq,'pin',pin)
```

Description

`lineseries = semilogy(h,parameter)` plots the specified `parameter` in the default format using a logarithmic scale for the y-axis. `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 `semilogy` method returns a column vector of handles to `lineseries` objects, one handle per line. This output is the same as the output returned by the MATLAB `semilogy` function.

`lineseries = semilogy(h,parameter1,...,parametern)` plots the parameters `parameter1,...`, `parametern` from the object `h` on an X-Y plane using a logarithmic scale for the y-axis.

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

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

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 Use the MATLAB `semilogy` function to create a semi-log scale plot of parameters that are specified as vector data and are not part of a circuit (`rfckt`) object or data (`rfdata`) object.

`lineseries = semilogy(h,'parameter1',...,'parametern', format,xparameter,xformat,'condition1',value1,..., 'conditionm',valuem, 'freq',freq,'pin',pin)` plots the specified parameters at the specified operating conditions for the object `h`.

`xparameter` is the independent variable to use in plotting the specified parameters. Several `xparameter` values are available for all objects. When you import `rfckt.amplifier`, `rfckt.mixer`, or `rfdata.data` object specifications from a `.p2d` or `.s2d` file, you can also specify any operating conditions from the file that have numeric values, such as bias.

The following table shows the most commonly available parameters and the corresponding `xparameter` values. The default settings listed in the table are used if `xparameter` is not specified.

Parameter Name	`xparameter` values
`Pout`, `Phase`, `LS11`, `LS12`, `LS21`, `LS22`	`Pin` (default), `Freq`
`S11`, `S12`, `S21`, `S22`, `NF`, `IIP3`, `OIP3`, `GroupDelay`, `VSWRIn`, `VSWROut`, `GammaIn`, `GammaOut`, `FMIN`, `GammaOPT`, `RN`, `TF1`, `TF2`, `Gt`, `Ga`, `Gp`, `Gmag`, `Gmsg`, `GammaMS`, `GammaML`, `K`, `Delta`, `Mu`, `MuPrime`	`Freq`
`AM/AM`, `AM/PM`	`AM`

`xformat` is the format to use for the specified `xparameter`. No `xformat` specification is needed when `xparameter` is an operating condition.

The following table shows the `xformat` values that are available for the `xparameter` values listed in the preceding table, along with the default settings that are used if `xformat` is not specified.

`xparameter` values	`xformat` values
`Pin`	`dBm` (default), `mW`, `W`, `dBW`
`Freq`	`THz`, `GHz`, `MHz`, `KHz`, `Hz` By default, `xformat` is chosen to provide the best scaling for the given `xparameter` values.
`AM`	`Magnitude (decibels)` (default), `Magnitude (linear)`

`condition1,value1,..., conditionm,valuem` are the optional condition/value pairs at which to plot the specified parameters. These pairs are usually operating conditions from a `.p2d` or `.s2d` file. For some parameters, you can specify a set of frequency or input power values at which to plot the specified parameter.

For example:

- When plotting large-signal S-parameters as a function of input power, you can specify frequency points of interest using condition/value pairs.
- When plotting large-signal S-parameters as a function of frequency, you can specify input power levels of interest using condition/value pairs.
- When plotting parameters as a function of an operating condition, you can specify both frequency and input power values using condition/value pairs.

`freq` is the optional frequency value, in hertz, at which to plot the specified parameters.

`pin` is the optional input power value, in dBm, at which to plot the specified parameters.

If `h` has multiple operating conditions, such as from a `.p2d` or `.s2d` file, the `semilogy` method operates as follows:

- If you do not specify any operating conditions as arguments to the `semilogy` method, then the method plots the parameter values based on the currently selected operating condition.
- If you specify one or more operating conditions, the `semilogy` method plots the parameter values based on those operating conditions.
- When you use an operating condition for the `xparameter` input argument, the method plots the parameters for all operating condition values.

See Also

`analyze` | `calculate` | `extract` | `getz0` | `listformat` | `listparam` | `loglog` | `plot` | `plotyy` | `polar` | `read` | `restore` | `semilogx` | `smith` | `write`

setop

Purpose

Set operating conditions

Syntax

```
setop(h)
setop(h,'Condition1')
setop(h,'Condition1',value1,'Condition2',value2,...)
```

Description

`setop(h)` lists the available values for all operating conditions of the object `h`. Operating conditions only apply to objects you import from a `.p2d` or `.s2d` file. To import these types of data into an object, use the `read` method. Operating conditions are not listed with other properties of an object.

`setop(h,'Condition1')` lists the available values for the specified operating condition `'Condition1'`.

`setop(h,'Condition1',value1,'Condition2',value2,...)` changes the operating conditions of the circuit or data object, `h`, to those specified by the condition/value pairs. Conditions you do not specify retain their original values. The method ignores any conditions that are not applicable to the specified object. Ignoring these conditions lets you apply the same set of operating conditions to an entire network where different conditions exist for different components.

When you set the operating conditions for a network that contains several objects, the software does not issue an error or warning if the specified conditions cannot be applied to all objects. For some networks, this lack of error or warning lets you call the `setop` method once to apply the same set of operating conditions to any objects where operating conditions are applicable. However, you may want to specify a network that contains one or more of the following:

- Several objects with different sets of operating conditions.
- Several objects with the same set of operating conditions that are configured differently.

To specify operating conditions one of these types of networks, use a separate call to the `setop` method for each object.

Examples

List the possible operating conditions of an object:

```
ckt1 = read(rfckt.amplifier, 'default.p2d');
setop(ckt1)
```

Analyze an object under specific operating conditions:

```
ckt1 = read(rfckt.amplifier, 'default.p2d');
freq = ckt1.AnalyzedResult.Freq;
setop(ckt1, 'Bias', '1.5');
result1 = analyze(ckt1, freq)
```

See Also

getop

show

Purpose

Produce a spur graph for a multiband transmitter or receiver

Syntax

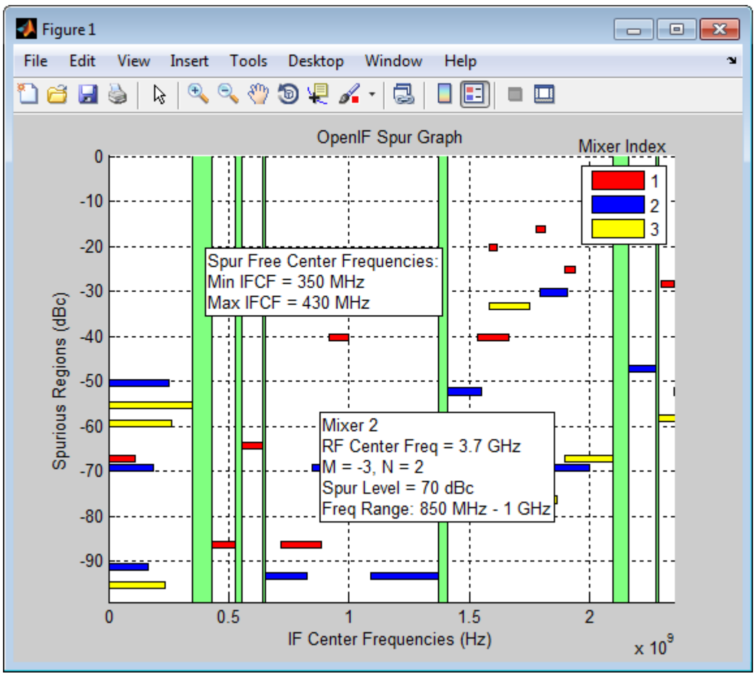
```
show(hif)
```

Description

`show(hif)` produces a spur graph of the `OpenIF` object `hif` The spur graph contains:

- Vertical green bands, representing spur-free zones.
- Horizontal colored bands, representing spurious regions.

The following figure shows a spur graph for the three-mixer multiband receiver example on the `OpenIF` class reference page.



Spur-free zones are ranges of possible IF center frequencies that are free from intermodulation distortion. Depending on the configuration of the mixers in `hif`, spur-free zones may not appear. Clicking on a spur-free zone produces a tooltip, which displays information about the spur-free zone:

- **Min IFCF** — The minimum possible IF center frequency f_{IF} for the corresponding spur-free zone.
- **Max IFCF** — The maximum possible IF center frequency f_{IF} for the corresponding spur-free zone.

Spurious regions contain intermodulation products from at least one mixer. The color of a spur on the spur graph indicates which mixer generates the spur, according to the legend on the spur graph. Clicking on a spurious region produces a tooltip, which displays information about the spur:

- **RF Center Freq** — the RF center frequency f_{RF} of the mixer that generates the spur
- **M**, **N** — the coefficients in the equation $|Mf_{RF} - N(f_{RF} \pm f_{IF})|$ (down-conversion) or the equation $|Mf_{IF} + N(f_{RF} \pm f_{IF})|$. The sign of '±' in these equations is determined by the injection type of the mixer. These coefficients refer to the particular mixing product that generates the spurious region.
- **Spur Level** — The difference in magnitude between a signal at 0 dBc and the spur. If you set `hif.SpurLevel` to a number greater than this value, then `hif` does not report the region as spurious.
- **Freq Range** — The frequency range of the spurious region. Choosing an IF center frequency in this range causes interference with the intermodulation product corresponding to the spur.

Input Arguments

hif

`OpenIF` object

Alternatives

- The `getSpurFreeZoneData` returns the endpoints of the spur-free zones in a matrix.
- The `report` method displays mixer configurations, intermodulation tables, and spur-free zone information at the command line.

Examples

- The `OpenIF` class reference page contains an example that shows how to find the spur-free zones of a multiband receiver with three mixers.
- The example Finding Free IF Bandwidths shows how to use information from a spur graph to design a multiband receiver with spur-free zones.

See Also

`addMixer` | `getSpurData` | `getSpurFreeZoneData` | `report` | `OpenIF`

Purpose

Plot specified circuit object parameters on Smith chart

Syntax

```
smith(hnet,i,j)
hsm = smith(hnet,i,j)
[lineseries,hsm] = smith(h,parameter1,...,parametern,type)
[lineseries,hsm] = smith(h,'parameter1',...,'parametern', type,
   xparameter,xformat,'condition1',value1,..., 'conditionm',
   valuem, 'freq',freq,'pin',pin)
```

Description

`smith(hnet,i,j)` plots the (*i*,*j*)th parameter of `hnet` on a Smith Chart. `hnet` is an RF Toolbox network parameter object. The inputs `i` and `j` are positive integers whose value is less than or equal to 2 for hybrid and hybrid-g parameter objects, or less than or equal to `hnet.NumPorts` for ABCD, S-, Y-, or Z-parameter objects.

`hsm = smith(hnet,i,j)` returns the handle of the figure used to create the plot, `hsm`.

`[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 that contains *n*-port network parameter data. `type` is a string that specifies the type of Smith chart:

- `'z'` (default)
- `'y'`
- `'zy'`

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` method to perform a frequency domain analysis before calling `smith`.

`[lineseries,hsm] = smith(h,'parameter1',...,'parametern', type,xparameter,xformat,'condition1',value1,..., 'conditionm',valuem, 'freq',freq,'pin',pin)` plots the specified parameters at the specified operating conditions for the object `h`.

`xparameter` is the independent variable to use in plotting the specified parameters. Several `xparameter` values are available for all objects. When you import 2-port `rfckt.amplifier`, `rfckt.mixer`, or `rfdata.data` object specifications from a `.p2d` or `.s2d` file, you can also specify any operating conditions from the file that have numeric values, such as bias.

The following table shows the most commonly available parameters and the corresponding `xparameter` values. The default settings listed in the table are used if `xparameter` is not specified.

Parameter Name	`xparameter` values
`Pout`, `Phase`, `LS11`, `LS12`, `LS21`, `LS22`	`Pin` (default), `Freq`
`S11`, `S12`, `S21`, `S22`, `NF`, `IIP3`, `OIP3`, `VSWRIn`, `VSWROut`, `GammaIn`, `GammaOut`, `FMIN`, `GammaOPT`, `RN`	`Freq`
`AM/AM`, `AM/PM`	`AM`

`xformat` is the format to use for the specified `xparameter`. No `xformat` specification is needed when `xparameter` is an operating condition.

The following table shows the `xformat` values that are available for the `xparameter` values listed in the preceding table, along with the default settings that are used if `xformat` is not specified.

`xparameter` values	`xformat` values
`Pin`	`dBm` (default), `mW`, `W`, `dBW`
`Freq`	`THz`, `GHz`, `MHz`, `KHz`, `Hz` By default, `xformat` is chosen to provide the best scaling for the given `xparameter` values.
`AM`	`Magnitude (decibels)` (default), `Magnitude (linear)`

`condition1,value1,...`, `conditionm,valuem` are the optional condition/value pairs at which to plot the specified parameters. These pairs are usually operating conditions from a `.p2d` or `.s2d` file. For some parameters, you can specify a set of frequency or input power values at which to plot the specified parameter.

For example:

- When plotting large-signal S-parameters as a function of input power, you can specify frequency points of interest using condition/value pairs.
- When plotting large-signal S-parameters as a function of frequency, you can specify input power levels of interest using condition/value pairs.
- When plotting parameters as a function of an operating condition, you can specify both frequency and input power values using condition/value pairs.

`freq` is the optional frequency value, in hertz, at which to plot the specified parameters.

`pin` is the optional input power value, in dBm, at which to plot the specified parameters.

If `h` has multiple operating conditions, such as from a `.p2d` or `.s2d` file, the `smith` method operates as follows:

- If you do not specify any operating conditions as arguments to the `smith` method, then the method plots the parameter values based on the currently selected operating condition.
- If you specify one or more operating conditions, the `smith` method plots the parameter values based on those operating conditions.
- When you use an operating condition for the `xparameter` input argument, the method plots the parameters for all operating condition values.

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` method returns `lineseries`, a column vector of handles to `lineseries` objects, one handle per plotted line. Use the MATLAB `lineseries properties` function to change the properties of these lines.

Changing Properties of the Smith Chart

The `smith` method 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 table 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 gray).
`LabelColor`	Color of the line labels.	`ColorSpec`. Default is `[0 0 0]` (black).
`LabelSize`	Size of the line labels.	`FontSize`. Default is `10`. See the `Annotation Textbox Properties` reference page for more information on specifying font size.
`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 gray).
`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`.

Property Name	Description	Units, Values
`Type`	Type of Smith chart.	`'z'` (default), `'y'`, or `'zy'`
`Value`	Two-row matrix. Row 1 specifies the values of the constant resistance and reactance lines that appear on the chart. For the constant resistance/reactance lines, each element in Row 2 specifies the value of the constant reactance/resistance line at which the corresponding line specified in Row 1 ends.	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` | `circle` | `getz0` | `listformat` | `listparam` | `loglog` | `plot` | `plotyy` | `polar` | `read` | `restore` | `semilogx` | `semilogy` | `write`

Purpose

Step-signal response of model object

Syntax

```
[yout,tout] = stepresp(h, ts, n, trise)
```

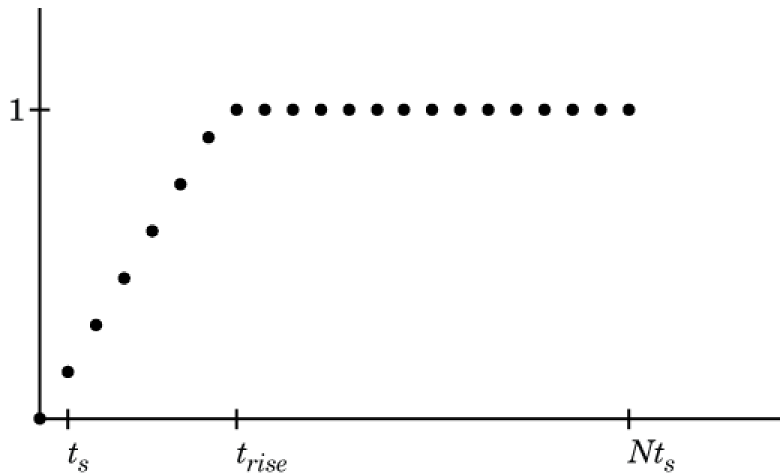
Description

`[yout,tout] = stepresp(h, ts, n, trise)` calculates the time-domain response of a rational function object, `h`, to a step signal, defined as:

$$\begin{cases} U(kt_s) = kt_s / t_{rise}, & 0 \le k < (t_{rise} / t_s) \\ U(kt_s) = 1, & (t_{rise} / t_s) \le k \le N \end{cases}$$

The variable t_s is the sample time, `ts`; N is the number of samples, `n`; and t_{rise} is the time, `trise`, that it takes for the step signal to reach its maximum value. The variable k is an integer between 0 and N, referring to the index of the samples.

The following figure illustrates the construction of this signal.



The output `yout` is the response of the step signal at time `tout`.

Examples

Calculate the step response of a rational function object:

```
% Read a .S2P data file
  h = read(rfckt.passive, 'passive.s2p')
% Get the S11 parameters
  S11 = h.AnalyzedResult.S_Parameters(1,1,:);
  Freq = h.AnalyzedResult.Freq;
% Fit S11 to a rational function object
  RationalModelS11 = rationalfit(Freq, S11);
% Define parameters for a step signal
  Ts = 1.0e-11; N = 10000; Trise = 1.0e-10;
% Calculate the step response for TDR and plot it
  [TDR, Time1] = stepresp(RationalModelS11, Ts, N, Trise);
  figure(1)
  plot(Time1*1e9, TDR);
  ylabel('TDR'); xlabel('Time (ns)');
% Calculate TDT and plot it
  S21 = h.AnalyzedResult.S_Parameters(2,1,:);
  RationalModelS21 = rationalfit(Freq, S21);
  [TDT, Time2] = stepresp(RationalModelS21, Ts, N, Trise);
  figure(2)
  plot(Time2*1e9, TDT);
  ylabel('TDT'); xlabel('Time (ns)');
```

See Also

`freqresp` | `rationalfit` | `timeresp`

How To

- `rfmodel.rational`

Purpose

Display specified RF object parameters in Variable Editor

Syntax

```
table(h,param1,format1,..., paramn,formatn)
table(h,'budget',param1,format1,...,paramn,formatn)
```

Description

`table(h,param1,format1,..., paramn,formatn)` displays the specified parameters *param1* through *paramn*, with units *format1* through *formatn*, in the Variable Editor. The input `h` is a function handle to an `rfckt` object.

The method creates a structure in the MATLAB workspace and constructs the name of the structure from the names of the object and parameters you provide. Specify parameters and formats in pairs. If you do not specify a format, the method uses the default format for that parameter.

To list valid parameters and parameter formats for `h`, use the `listparam` and `listformat` methods.

`table(h,'budget',param1,format1,...,paramn,formatn)` specified budget parameters of an `rfckt.cascade` object `h`.

Examples

Analyze an RF cascade and display the link budget in a table:

```
% Construct a cascaded RFCKT object
Cascaded_Ckt = rfckt.cascade('Ckts', ...
               {rfckt.txline('LineLength', .001), ...
               rfckt.amplifier, rfckt.txline( ...
               'LineLength', 0.025, 'PV', 2.0e8)})
% Do frequency domain analysis at the given frequency
freq = 2.1e9;
analyze(Cascaded_Ckt,freq);
% Plot the Budget S21 and NF
plot(Cascaded_Ckt, 'budget', 'S21', 'NF');
% Display the Budget S21 and NF in a table
table(Cascaded_Ckt, 'budget', 'S21', 'NF');
```

See Also

`openvar` | `plot`

Purpose

Time response for model object

Syntax

```
[y,t] = timeresp(h,u,ts)
```

Description

`[y,t] = timeresp(h,u,ts)` computes the output signal, `y`, that the `rfmodel` object, `h`, produces in response to the given input signal, `u`.

The input `h` is the handle of a model object. `ts` is a positive scalar value that specifies the sample time of the input signal.

The output `y` is the output signal. RF Toolbox software computes the value of the signal at the time samples in the vector `t` using the following equation.

$$Y(n) = sum(C.*X(n - Delay/ts)) + D*U(n - Delay/ts)$$

where

$$X(n+1) = F*X(n) + G*U(n)$$
$$X(1) = 0$$
$$F = \exp(A*ts)$$
$$G = (F-1)./A$$

and `A`, `C`, `D`, and `Delay` are properties of the `rfmodel.rational` object, `h`.

Examples

The following example shows you how to compute the time response of the data stored in the file `default.s2p` by fitting a rational function model to the data and using the `timeresp` method to compute the time response of the model.

```
% Define the input signal
SampleTime = 2e-11;
OverSamplingFactor = 25;
TotalSampleNumber = 2^12;
InputTime = double((1:TotalSampleNumber)')*SampleTime;
InputSignal = sign(randn(1, ...
              ceil(TotalSampleNumber/OverSamplingFactor)));
InputSignal = repmat(InputSignal, [OverSamplingFactor, 1]);
```

```
InputSignal = InputSignal(:);

% Create a rational function model
orig_data=read(rfdata.data,'default.s2p');
freq=orig_data.Freq;
data=orig_data.S_Parameters(2,1,:);
fit_data=rationalfit(freq,data);

% Compute the time response
[y,t]=timeresp(fit_data,InputSignal,SampleTime);
```

See Also

freqresp | rationalfit | writeva

How To

- rfmodel.rational

Purpose

Write RF data from circuit or data object to 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`. `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.

The `printformat` string that specifies the precision of the network and noise parameters. The default value is `%22.10f`. This value means the method writes the data using fixed-point notation with a precision of 10 digits. The minimum positive value the `write` method can express by default is 1e-10. For greater precision, specify a different `printformat`. See the Format String specification for `fprintf`.

The `freqformat` string that specifies the precision of the frequency. The default value is `%-22.10f`. See the Format String specification for `fprintf`.

Note The method only writes property values from `data` that the specified output file supports. For example, Touchstone files, which have the `.snp`, `.ynp`, `.znp`, or `.hnp` extension, do not support noise figure or output third-order intercept point. Consequently, the `write` method does not write these property values to these such files.

Examples

The following example shows you how to analyze the data stored in the file `default.s2p` for a different set of frequency values, and use the `write` method to store the results in a file called `test.s2p`.

```
orig_data=read(rfdata.data,'default.s2p')
freq=[1:.1:2]*1e9;
analyze(orig_data,freq);
write(orig_data,'test.s2p');
```

References

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

See Also

`analyze` | `calculate` | `extract` | `getz0` | `listformat` | `listparam` | `loglog` | `plot` | `plotyy` | `polar` | `semilogx` | `semilogy` | `smith` | `read` | `restore`

Purpose

Write Verilog-A description of RF model object

Syntax

```
status = writeva(h,filename,innets,outnets, ...
          printformat,discipline,filestoinclude)
```

Description

`status = writeva(h,filename,innets,outnets,printformat, discipline,filestoinclude)` writes a Verilog-A module that describes an `rfmodel` object `h` to the file specified by `filename`. The method implements the object in Verilog-A using Laplace Transform S-domain filters. It returns a `status` of `True`, if the operation is successful, and `False` if it is unsuccessful.

`h` is the handle to the `rfmodel.rational` object. Typically, the `rationalfit` function creates this object when you fit a rational function to a set of data.

`filename` is a string representing the name of the Verilog-A file to which to write the module. The `filename` can be specified with or without a path name and extension. The default extension, `.va`, is added automatically if `filename` does not end in this extension. The module name that is used in the file is the part of the `filename` that remains when the path name and extension are removed.

`innets` is a string or a cell of two strings that specifies the name of each of the module's input nets. The default is `'in'`.

`outnets` is a string or a cell of two strings that specifies the name of each of the module's output nets. The default is `'out'`.

`printformat` is a string that specifies the precision of the following Verilog-A module parameters using the C language conversion specifications:

- The numerator and denominator coefficients of the Verilog-A filter.
- The module's delay value and constant offset (or direct feedthrough), which are taken directly from the `rfmodel` object.

The default is `'%15.10e'`. For more information on how to specify `printformat`, see the Format String specification for `fprintf`.

`discipline` is a string that specifies the predefined Verilog-A discipline of the nets. The discipline defines attributes and characteristics associated with the nets. The default is `'electrical'`.

`filestoinclude` is a cell of strings that specifies a list of header files to include in the module using Verilog-A ``'`include'`` statements. By default, `filestoinclude` is set to ``'`include discipline.vams'``.

For more information on Verilog-A, see the Verilog-A Reference Manual.

See Also

`freqresp` | `rationalfit` | `timeresp`

How To

- `rfmodel.rational`

newref

Purpose

Change reference impedance of S-parameters

Syntax

```
hs2 = newref(hs,Z0)
```

Description

`hs2 = newref(hs,Z0)` creates an S-parameter object, `hs2`, by converting the S-parameters in `hs` to the specified reference impedance, `Z0`.

Input Arguments

hs - S-parameters
network parameter object

S-parameters, specified as an RF Toolbox network parameter object. To create this type of object, use the `sparameters` function.

Z0 - Reference impedance
real positive scalar

Characteristic impedance, in ohms, specified as a real positive scalar.

Output Arguments

hs2 - S-parameters
network parameter object

S-parameters with reference impedance `Z0`, returned as an RF Toolbox network parameter object.

Examples

Change reference impedance of S-parameters

Create an S-parameters object from data in the file, `default.s2p`.

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

Change the reference impedance to 40 ohms.

```
hs2 = newref(hs,40);
```

See Also

`sparameters`

Purpose

Interpolate network parameter data

Syntax

```
hnet2 = rfinterp1(hnet,freq)
```

Description

`hnet2 = rfinterp1(hnet,freq)` interpolates the network parameter data in `hnet` at the specified frequencies, `freq`, storing the results in `hnet2`.

Input Arguments

hnet - Original data
network parameter object

Data to interpolate, specified as an RF Toolbox network parameter object.

freq - Frequencies
vector of positive numbers

Frequencies of interpolation, specified as a vector of positive numbers ordered from smallest to largest.

Output Arguments

hnet2 - Interpolated data
network parameter object

Result of interpolation, returned as an RF Toolbox network parameter object of the same type as `hnet`.

Algorithms

The function uses the MATLAB function `interp1` to perform the interpolation operation. Overall performance is similar to the RF Toolbox `analyze` function. However, behaviors of the two functions differ when `freq` contains frequencies outside the range of the original data:

- `analyze` performs a zeroth-order extrapolation for out-of-range data points.
- `rfinterp1` inserts `NaN` values for out-of-range data points.

Examples

Interpolate S-parameter data

Read the data from the file `default.s2p` into an S-parameter object.

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

Interpolate the data at a specified set of frequencies.

```
freq = [1.2:0.2:2.8]*1e9;
hnet2 = rfinterp1(hnet,freq);
```

See Also

`analyze` | `interp1`

Purpose

Extract vector of network parameters

Syntax

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

Description

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

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

Input Arguments

abcdflag - ABCD-parameter index

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

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

habcd - 2-port ABCD parameters

ABCD parameter object

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

hnet - Network parameters

network parameter object

Network parameters, specified as an RF Toolbox network parameter object.

i - Row index

positive integer

Row index of data to extract, specified as a positive integer.

j - Column index

positive integer

Column index of data to extract, specified as a positive integer.

Output Arguments

n_ij - Network parameters (*i,j*)

vector

Network parameters (*i,j*), returned as a vector. The `i` and `j` input arguments determine which parameters the function returns.

Example: `S_21 = rfparam(hs,2,1)`

abcd_vector - *A*, *B*, *C*, or *D*-parameters

vector

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

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

Purpose

Plot S-parameter data

Syntax

```
rfplot(hs,i,j)
rfplot(hs)
rfplot(___,LineSpec)
hfig = rfplot(___)
```

Description

`rfplot(hs,i,j)` plots the magnitude of $S_{i,j}$, in decibels, versus frequency on the current axis.

`rfplot(hs)` plots all S-parameter vectors in `hs` (S_{11}, S_{12} ... S_{NN}) on the current axis.

`rfplot(___,LineSpec)` plots S-parameters using optional line types, symbols, and colors specified by `LineSpec`.

`hfig = rfplot(___)` plots the S-parameters and returns the handle to the figure, `hfig`.

Input Arguments

hs - S-parameters

network parameter object

S-parameters, specified as an RF Toolbox network parameter object. To create this type of object, use the `sparameters` function.

i - Row index

positive integer

Row index of data to plot, specified as a positive integer.

j - Column index

positive integer

Column index of data to plot, specified as a positive integer.

LineSpec - Line specification

string specifier

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

Example: `'-or'`

Output Arguments

hfig - Figure
figure handle

Figure containing the S-parameter plot, returned as a figure handle.

Examples

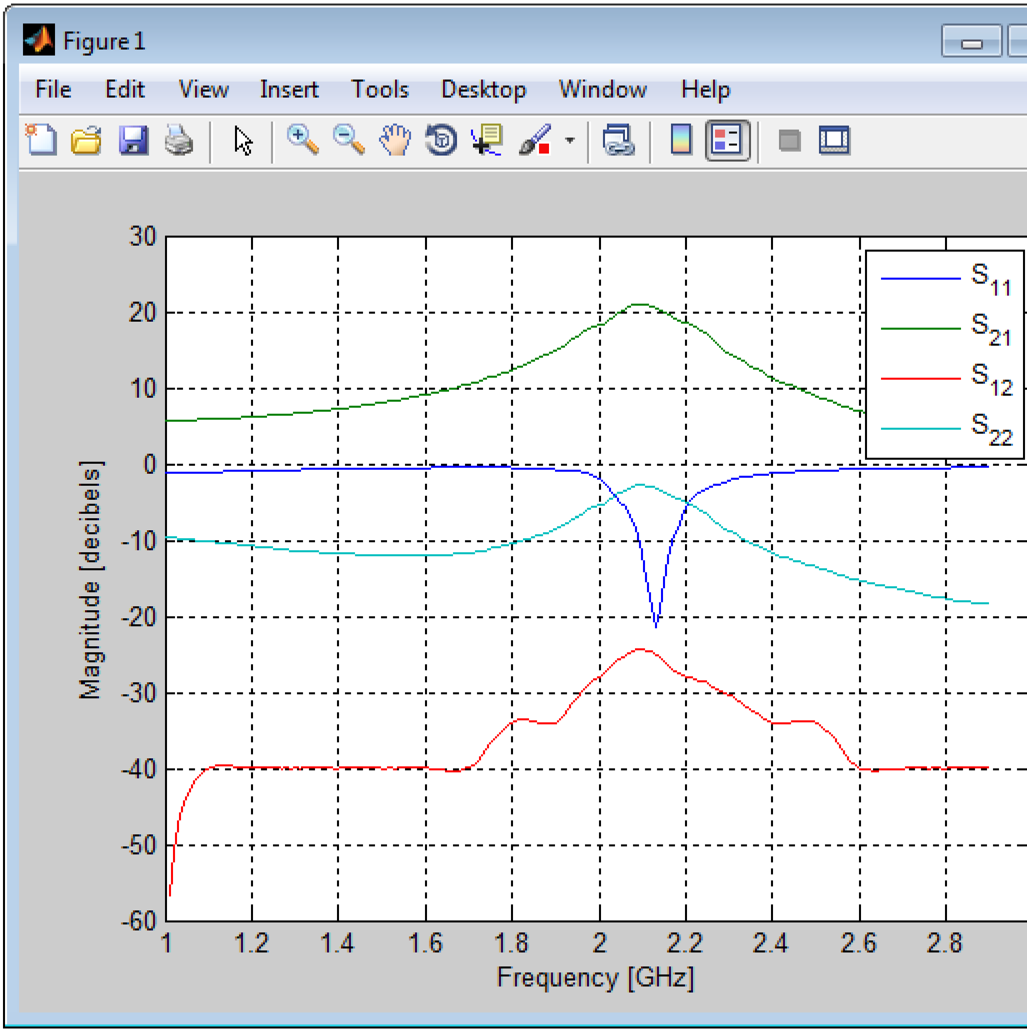
Plot S-parameter data

Create an S-parameter object from data in the file `default.s2p`.

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

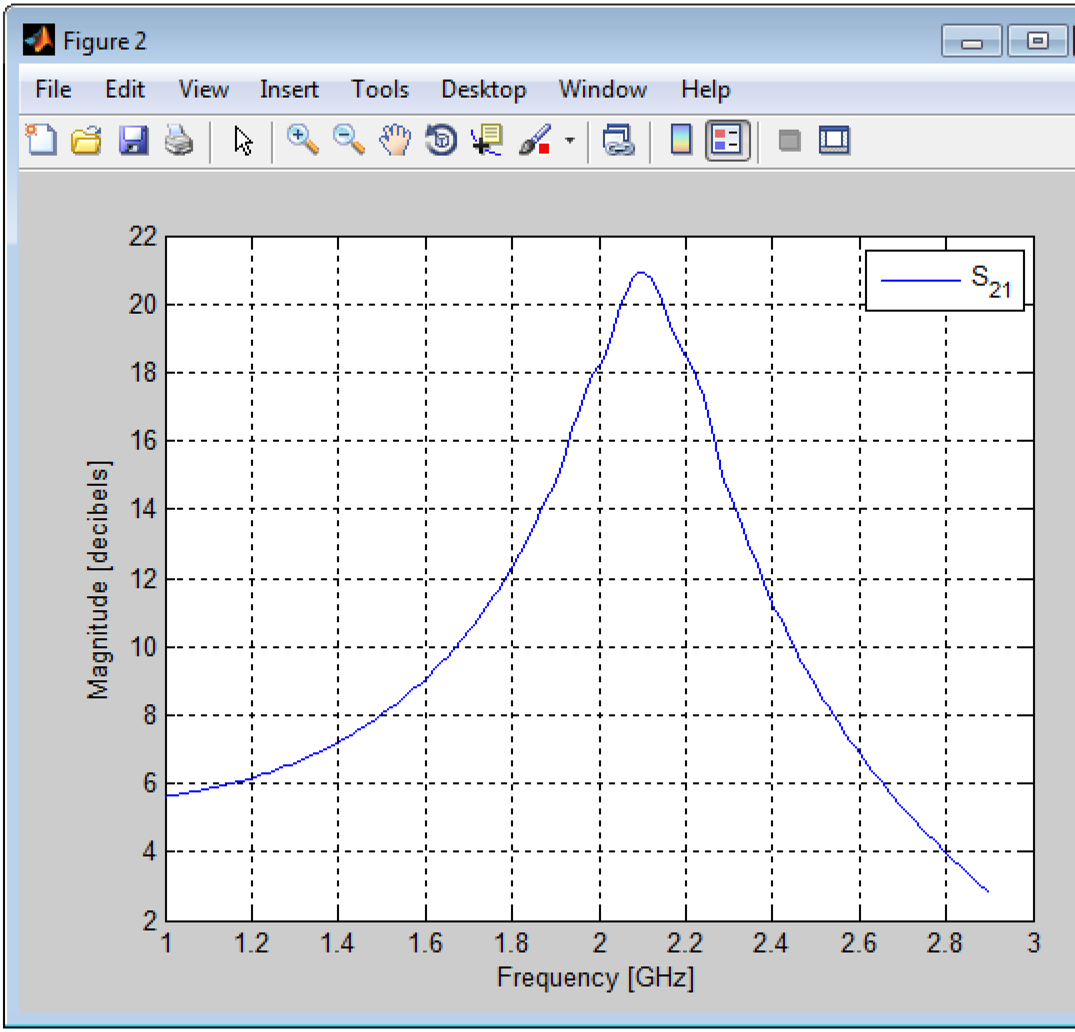
Plot all S-parameters. The data describes a 2-port network, and thus S_{11}, S_{12}, S_{21}, and S_{22} appear on the plot.

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



To plot S_{21} only, provide additional inputs to `rfplot`.

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



See Also

`smith`

Purpose

Create S-parameter object

Syntax

```
hs = sparameters(filename)

hs = sparameters(hnet)
hs = sparameters(data,freq)
hs = sparameters( ___ ,Z0)
```

Description

`hs = sparameters(filename)` creates an S-parameter object `hs` by importing data from the Touchstone file specified by `filename`.

`hs = sparameters(hnet)` creates an S-parameter object from the RF Toolbox network parameter object `hnet`.

`hs = sparameters(data,freq)` creates an S-parameter object from the S-parameter data, `data`, and frequencies, `freq`.

`hs = sparameters( ___ ,Z0)` creates an S-parameter object with a reference impedance of `Z0`.

Input Arguments

data - S-parameter data

array of complex data

S-parameter data, specified as an N-by-N-by-K array of complex numbers. The function uses this input argument to set the value of the `Parameters` property of `hs`.

filename - Touchstone data file that contains network parameter data

string

Touchstone data file, specified as a string. `filename` can be the name of a file on the MATLAB path or the full path to a file.

Example: `hs = sparameters('defaultbandpass.s2p');`

freq - S-parameter frequencies

vector of increasing positive scalars

S-parameter frequencies, specified as a vector of positive real numbers sorted from smallest to largest. The function uses this input argument to set the value of the `Frequencies` property of `hs`.

hnet - Network parameter data

network parameter object

Network parameter data, specified as a network parameter object. If `hnet` is an S-parameter object, then `hs` is a deep copy of `hnet`. Otherwise, the function performs a network parameter conversion to create `hs`. When converting network parameters, the same restrictions apply as those for RF Toolbox network parameter data conversion functions:

- ABCD parameter objects support 2*N*-port data.
- Hybrid-g parameter objects support 2-port data.
- H-parameter objects support 2-port data.
- S-parameter objects support *N*-port data.
- Y-parameter objects support *N*-port data.
- Z-parameter objects support *N*-port data.

Z0 - Reference impedance

positive real scalar | 50 (default)

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

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

Output Arguments

hs - S-parameter object
network parameter object

S-parameter data, returned as a network parameter object. `disp(hs)` returns the properties of the object:

- `NumPorts` — Number of ports, specified as an integer. The function calculates this value automatically when you create the object.
- `Frequencies` — S-parameter frequencies, specified as a K-by-1 vector of positive real numbers sorted from smallest to largest. The function sets this property from the `filename` or `freq` input arguments.
- `Parameters` — S-parameter data, specified as an N-by-N-by-K array of complex numbers. The function sets this property from the `filename` or `data` input arguments.
- `Impedance` — Reference impedance in ohms, specified as a positive real scalar. The function sets this property from the `filename` or `Z0` input arguments. If no reference impedance is provided, the function uses a default value of `50`.

See Also

`abcdparameters` | `gparameters` | `hparameters` | `yparameters` | `zparameters` | `rfparam` | `rfplot`

abcdparameters

Purpose

Create ABCD parameter object

Syntax

```
habcd = abcdparameters(filename)

habcd = abcdparameters(hnet)
habcd = abcdparameters(data,freq)
```

Description

`habcd = abcdparameters(filename)` creates an ABCD parameter object `habcd` by importing data from the Touchstone file specified by `filename`.

`habcd = abcdparameters(hnet)` creates an ABCD parameter object from the RF Toolbox network parameter object `hnet`.

`habcd = abcdparameters(data,freq)` creates an ABCD parameter object from the ABCD parameter data, `data`, and frequencies, `freq`.

Input Arguments

data - ABCD parameter data
array of complex data

ABCD parameter data, specified as a $2N$-by-$2N$-by-K array of complex numbers. The function uses this input argument to set the value of the `Parameters` property of `habcd`.

filename - Touchstone data file that contains network parameter data
string

Touchstone data file, specified as a string. `filename` can be the name of a file on the MATLAB path or the full path to a file.

Example: `habcd = abcdparameters('defaultbandpass.s2p');`

freq - ABCD parameter frequencies
vector of increasing positive scalars

ABCD parameter frequencies, specified as a vector of positive real numbers sorted from smallest to largest. The function uses this input argument to set the value of the `Frequencies` property of `habcd`.

hnet - Network parameter data

network parameter object

Network parameter data, specified as a network parameter object. If `hnet` is an ABCD parameter object, then `habcd` is a deep copy of `hnet`. Otherwise, the function performs a network parameter conversion to create `habcd`. When converting network parameters, the same restrictions apply as those for RF Toolbox network parameter data conversion functions:

- ABCD parameter objects support 2*N*-port data.
- Hybrid-g parameter objects support 2-port data.
- Hybrid parameter objects support 2-port data.
- S-parameter objects support *N*-port data.
- Y-parameter objects support *N*-port data.
- Z-parameter objects support *N*-port data.

Output Arguments

habcd - ABCD parameter object

network parameter object

ABCD parameter data, returned as a network parameter object. `disp(habcd)` returns the properties of the object:

- `NumPorts` — Number of ports, specified as an integer. The function calculates this value automatically when you create the object.
- `Frequencies` — ABCD parameter frequencies, specified as a *K*-by-1 vector of positive real numbers sorted from smallest to largest. The function sets this property from the `filename` or `freq` input arguments.

- `Parameters` — ABCD parameter data, specified as a $2N$-by-$2N$-by-K array of complex numbers. The function sets this property from the `filename` or `data` input arguments.

See Also

`gparameters` | `hparameters` | `sparameters` | `yparameters` | `zparameters` | `rfparam`

Purpose

Create hybrid-g parameter object

Syntax

```
hg = gparameters(filename)

hg = gparameters(hnet)
hg = gparameters(data,freq)
```

Description

`hg = gparameters(filename)` creates a hybrid-g parameter object `hg` by importing data from the Touchstone file specified by `filename`.

`hg = gparameters(hnet)` creates a hybrid-g parameter object from the RF Toolbox network parameter object `hnet`.

`hg = gparameters(data,freq)` creates a hybrid-g parameter object from the g-parameter data, `data`, and frequencies, `freq`.

Input Arguments

data - Hybrid-g parameter data

array of complex data

Hybrid-g parameter data, specified as a 2-by-2-by-K array of complex numbers. The function uses this input argument to set the value of the `Parameters` property of `hg`.

filename - Touchstone data file that contains network parameter data

string

Touchstone data file, specified as a string. `filename` can be the name of a file on the MATLAB path or the full path to a file.

Example: `hg = gparameters('defaultbandpass.s2p');`

freq - Hybrid-g parameter frequencies

vector of increasing positive scalars

Hybrid-g parameter frequencies, specified as a vector of positive real numbers sorted from smallest to largest. The function uses this input argument to set the value of the `Frequencies` property of `hg`.

hnet - Network parameter data

network parameter object

Network parameter data, specified as a network parameter object. If `hnet` is a hybrid-g parameter object, then `hg` is a deep copy of `hnet`. Otherwise, the function performs a network parameter conversion to create `hg`. When converting network parameters, the same restrictions apply as those for RF Toolbox network parameter data conversion functions:

- ABCD parameter objects support 2*N*-port data.
- Hybrid-g parameter objects support 2-port data.
- Hybrid parameter objects support 2-port data.
- S-parameter objects support *N*-port data.
- Y-parameter objects support *N*-port data.
- Z-parameter objects support *N*-port data.

Output Arguments

hg - Hybrid-g parameter object

network parameter object

Hybrid-g parameter data, returned as a network parameter object. `disp(hg)` returns the properties of the object:

- `Frequencies` — Hybrid-g parameter frequencies, specified as a *K*-by-1 vector of positive real numbers sorted from smallest to largest. The function sets this property from the `filename` or `freq` input arguments.
- `Parameters` — Hybrid-g parameter data, specified as an *N*-by-*N*-by-*K* array of complex numbers. The function sets this property from the `filename` or `data` input arguments.

See Also

`abcdparameters` | `hparameters` | `sparameters` | `yparameters` | `zparameters` | `rfparam`

hparameters

Purpose

Create hybrid parameter object

Syntax

```
hh = hparameters(filename)

hh = hparameters(hnet)
hh = hparameters(data,freq)
```

Description

`hh = hparameters(filename)` creates a hybrid parameter object `hh` by importing data from the Touchstone file specified by `filename`.

`hh = hparameters(hnet)` creates a hybrid parameter object from the RF Toolbox network parameter object `hnet`.

`hh = hparameters(data,freq)` creates a hybrid parameter object from the hybrid parameter data, `data`, and frequencies, `freq`.

Input Arguments

data - Hybrid parameter data

array of complex data

Hybrid parameter data, specified as 2-by-2-by-K array of complex numbers. The function uses this input argument to set the value of the `Parameters` property of `hh`.

filename - Touchstone data file that contains network parameter data

string

Touchstone data file, specified as a string. `filename` can be the name of a file on the MATLAB path or the full path to a file.

Example: `hh = hparameters('defaultbandpass.s2p');`

freq - Hybrid parameter frequencies

vector of increasing positive scalars

Hybrid parameter frequencies, specified as a vector of positive real numbers sorted from smallest to largest. The function uses this input argument to set the value of the `Frequencies` property of `hh`.

hnet - Network parameter data

network parameter object

Network parameter data, specified as a network parameter object. If `hnet` is a hybrid parameter object, then `hh` is a deep copy of `hnet`. Otherwise, the function performs a network parameter conversion to create `hh`. When converting network parameters, the same restrictions apply as those for RF Toolbox network parameter data conversion functions:

- ABCD parameter objects support $2N$-port data.
- Hybrid-g parameter objects support 2-port data.
- Hybrid parameter objects support 2-port data.
- S-parameter objects support N-port data.
- Y-parameter objects support N-port data.
- Z-parameter objects support N-port data.

Output Arguments

hh - Hybrid parameter object

network parameter object

Hybrid parameter data, returned as a network parameter object. `disp(hh)` returns the properties of the object:

- `Frequencies` — Hybrid parameter frequencies, specified as a K-by-1 vector of positive real numbers sorted from smallest to largest. The function sets this property from the `filename` or `freq` input arguments.
- `Parameters` — Hybrid parameter data, specified as a 2-by-2-by-K array of complex numbers. The function sets this property from the `filename` or `data` input arguments.

See Also

abcdparameters | gparameters | sparameters | yparameters | zparameters | rfparam

Purpose

Create Y-parameter object

Syntax

```
hy = yparameters(filename)

hy = yparameters(hnet)
hy = yparameters(data,freq)
```

Description

`hy = yparameters(filename)` creates a Y-parameter object `hy` by importing data from the Touchstone file specified by `filename`.

`hy = yparameters(hnet)` creates a Y-parameter object from the RF Toolbox network parameter object `hnet`.

`hy = yparameters(data,freq)` creates a Y-parameter object from the Y-parameter data, `data`, and frequencies, `freq`.

Input Arguments

data - Y-parameter data

array of complex data

Y-parameter data, specified as an N-by-N-by-K array of complex numbers. The function uses this input argument to set the value of the `Parameters` property of `hy`.

filename - Touchstone data file that contains network parameter data

string

Touchstone data file, specified as a string. `filename` can be the name of a file on the MATLAB path or the full path to a file.

Example: `hy = yparameters('defaultbandpass.s2p');`

freq - Y-parameter frequencies

vector of increasing positive scalars

Y-parameter frequencies, specified as a vector of positive real numbers sorted from smallest to largest. The function uses this input argument to set the value of the `Frequencies` property of `hy`.

hnet - Network parameter data

network parameter object

Network parameter data, specified as a network parameter object. If `hnet` is a Y-parameter object, then `hy` is a deep copy of `hnet`. Otherwise, the function performs a network parameter conversion to create `hy`. When converting network parameters, the same restrictions apply as those for RF Toolbox network parameter data conversion functions:

- ABCD parameter objects support $2N$-port data.
- Hybrid-g parameter objects support 2-port data.
- Hybrid parameter objects support 2-port data.
- S-parameter objects support N-port data.
- Y-parameter objects support N-port data.
- Z-parameter objects support N-port data.

Output Arguments

hy - Y-parameter object

network parameter object

Y-parameter data, returned as a network parameter object. `disp(hy)` returns the properties of the object:

- `NumPorts` — Number of ports, specified as an integer. The function calculates this value automatically when you create the object.
- `Frequencies` — Y-parameter frequencies, specified as a K-by-1 vector of positive real numbers sorted from smallest to largest. The function sets this property from the `filename` or `freq` input arguments.
- `Parameters` — Y-parameter data, specified as an N-by-N-by-K array of complex numbers. The function sets this property from the `filename` or `data` input arguments.

See Also

`abcdparameters` | `gparameters` | `hparameters` | `sparameters` | `zparameters` | `rfparam`

zparameters

Purpose

Create Z-parameter object

Syntax

```
hz = zparameters(filename)

hz = zparameters(hnet)
hz = zparameters(data,freq)
hz = zparameters( ___ ,Z0)
```

Description

`hz = zparameters(filename)` creates a Z-parameter object `hz` by importing data from the Touchstone file specified by `filename`.

`hz = zparameters(hnet)` creates a Z-parameter object from the RF Toolbox network parameter object `hnet`.

`hz = zparameters(data,freq)` creates a Z-parameter object from the Z-parameter data, `data`, and frequencies, `freq`.

`hz = zparameters( ___ ,Z0)` creates a Z-parameter object using the previously described syntax, with a reference impedance of `Z0`.

Input Arguments

data - Z-parameter data

array of complex data

Z-parameter data, specified as an N-by-N-by-K array of complex numbers. The function uses this input argument to set the value of the `Parameters` property of `hz`.

filename - Touchstone data file that contains network parameter data

string

Touchstone data file, specified as a string. `filename` can be the name of a file on the MATLAB path or the full path to a file.

Example: `hz = zparameters('defaultbandpass.s2p');`

freq - Z-parameter frequencies

vector of increasing positive scalars

Z-parameter frequencies, specified as a vector of positive real numbers sorted from smallest to largest. The function uses this input argument to set the value of the `Frequencies` property of `hz`.

hnet - Network parameter data

network parameter object

Network parameter data, specified as a network parameter object. If `hnet` is a Z-parameter object, then `hz` is a deep copy of `hnet`. Otherwise, the function performs a network parameter conversion to create `hz`. When converting network parameters, the same restrictions apply as those for RF Toolbox network parameter data conversion functions:

- ABCD parameter objects support $2N$-port data.
- Hybrid-g parameter objects support 2-port data.
- Hybrid parameter objects support 2-port data.
- S-parameter objects support N-port data.
- Y-parameter objects support N-port data.
- Z-parameter objects support N-port data.

Output Arguments

hz - Z-parameter object

network parameter object

Z-parameter data, returned as a network parameter object. `disp(hz)` returns the properties of the object:

- `NumPorts` — Number of ports, specified as an integer. The function calculates this value automatically when you create the object.
- `Frequencies` — Z-parameter frequencies, specified as a K-by-1 vector of positive real numbers sorted from smallest to largest. The function sets this property from the `filename` or `freq` input arguments.

- `Parameters` — Z-parameter data, specified as an N-by-N-by-K array of complex numbers. The function sets this property from the `filename` or `data` input arguments.

See Also

`abcdparameters` | `gparameters` | `hparameters` | `sparameters` | `yparameters` | `rfparam`

Functions — Alphabetical List

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 2-port ABCD-parameters. `h_params` is a complex 2-by-2-by-M array, representing M 2-port hybrid h-parameters.

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-12.

Examples

Convert ABCD-parameters to h-parameters:

```
%Define a matrix of ABCD-parameters.
A =      0.999884396265344 +  0.000129274757618717i;
B =      0.314079483671772 +      2.51935878310427i;
C = -6.56176712108866e-007 + 6.67455405306704e-006i;
D =      0.999806365547959 +  0.000247230611054075i;
abcd_params = [A,B; C,D];
%Convert to h-parameters
h_params = abcd2h(abcd_params);
```

See Also

`abcd2s` | `abcd2y` | `abcd2z` | `h2abcd` | `s2h` | `y2h` | `z2h`

Purpose

Convert ABCD-parameters to S-parameters

Syntax

```
s_params = abcd2s(abcd_params,z0)
```

Description

`s_params = abcd2s(abcd_params,z0)` converts the ABCD-parameters `abcd_params` into the scattering parameters `s_params`. The `abcd_params` input is a complex $2N$-by-$2N$-by-M array, representing M $2N$-port ABCD-parameters. `z0` is the reference impedance; its default is 50 ohms. The function assumes that the ABCD-parameter matrices have distinct A, B, C, and D submatrices:

$$\begin{bmatrix} [A] & [B] \\ [C] & [D] \end{bmatrix}$$

`s_params` is a complex $2N$-by-$2N$-by-M array, representing M $2N$-port S-parameters.

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-12.

Examples

Convert ABCD-parameters to S-parameters:

```
%Define a matrix of ABCD-parameters.
A =      0.999884396265344 +  0.000129274757618717i;
B =      0.314079483671772 +      2.51935878310427i;
C = -6.56176712108866e-007 + 6.67455405306704e-006i;
D =      0.999806365547959 +  0.000247230611054075i;
abcd_params = [A,B; C,D];
%Convert to S-parameters
s_params = abcd2s(abcd_params);
```

See Also

`abcd2h` | `abcd2y` | `abcd2z` | `s2abcd` | `s2h` | `y2h` | `z2h`

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 $2N$-by-$2N$-by-M array, representing M $2N$-port ABCD-parameters. The function assumes that the ABCD-parameter matrices have distinct A, B, C, and D submatrices:

$$\begin{bmatrix} [A] & [B] \\ [C] & [D] \end{bmatrix}$$

`y_params` is a complex $2N$-by-$2N$-by-M array, representing M $2N$-port Y-parameters.

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-12.

Examples

Convert ABCD-parameters to Y-parameters:

```
%Define a matrix of ABCD-parameters.
A =      0.999884396265344 +  0.000129274757618717i;
B =      0.314079483671772 +      2.51935878310427i;
C = -6.56176712108866e-007 + 6.67455405306704e-006i;
D =      0.999806365547959 +  0.000247230611054075i;
abcd_params = [A,B; C,D];
%Convert to Y-parameters
y_params = abcd2y(abcd_params);
```

See Also

`abcd2h | abcd2s | abcd2z | h2y | s2y | y2abcd | z2y`

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 $2N$-by-$2N$-by-M array, representing M $2N$-port ABCD-parameters. The function assumes that the ABCD-parameter matrices have distinct A, B, C, and D submatrices:

$$\begin{bmatrix} [A] & [B] \\ [C] & [D] \end{bmatrix}$$

`z_params` is a complex $2N$-by-$2N$-by-M array, representing M $2N$-port Z-parameters.

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-12.

Examples

Convert ABCD-parameters to Z-parameters:

```
%Define a matrix of ABCD-parameters.
A =      0.999884396265344 +  0.000129274757618717i;
B =      0.314079483671772 +      2.51935878310427i;
C = -6.56176712108866e-007 + 6.67455405306704e-006i;
D =      0.999806365547959 +  0.000247230611054075i;
abcd_params = [A,B; C,D];
%Convert to Z-parameters
z_params = abcd2z(abcd_params);
```

See Also

`abcd2h` | `abcd2s` | `abcd2y` | `h2y` | `y2abcd` | `z2abcd`

cascadesparams

Purpose

Combine S-parameters to form cascaded network

Syntax

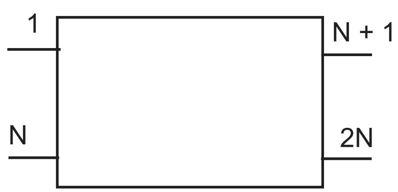
```
s_params = cascadesparams(s1_params,s2_params,...,sk_params)
hs = cascadesparams(hs1,hs2,...,hsk)
s_params = cascadesparams(s1_params,s2_params,...,sk_params,
   Kconn)
```

Description

`s_params = cascadesparams(s1_params,s2_params,...,sk_params)` cascades the scattering parameters of the K input networks described by the S-parameters `s1_params` through `sk_params`. The function stores the S-parameters of the cascade in `s_params`. Each of the input networks must be a $2N$-port network described by a $2N$-by-$2N$-by-M array of S-parameters. All networks must have the same reference impedance.

`hs = cascadesparams(hs1,hs2,...,hsk)` cascades K S-parameter objects to create the cascaded network `hs`. The function checks that the `Impedance` and `Frequencies` properties of each object are equal and that the `Parameters` property contains a $2N$-by-$2N$-by-M array of S-parameters.

`cascadesparams` assumes that you are using the port ordering given in the following illustration.



Based on this ordering, the function connects ports `N + 1` through `2N` of the first network to ports `1` through `N` of the second network. Therefore, when you use this syntax:

- Each network has an even number of ports
- Every network in the cascade has the same number of ports.

To use this function for S-parameters with different port arrangements, use the `snp2smp` function to reorder the port indices before cascading the networks.

`s_params = cascadesparams(s1_params,s2_params,...,sk_params,Kconn)` cascades the scattering parameters of the K input networks described by the S-parameters `s1_params` through `sk_params`. The function creates a cascaded network based on the number of cascade connections between networks, specified by `Kconn`. `Kconn` must be a positive scalar or vector of size $K - 1$.

- If `Kconn` is a scalar, `cascadesparams` makes the same number of connections between each pair of consecutive networks.
- If `Kconn` is a vector, the `i`th element of `Kconn` specifies the number of connections between the `i`th and the `i+1`th networks.

`cascadesparams` always connects the last `Kconn(i)` ports of the `i`th network and the first `Kconn(i)` ports of the `i+1`th network. The ports of the entire cascaded network represent the unconnected ports of each individual network, taken in order from the first network to the `n`th network.

Additionally, when you specify `Kconn`:

- Each network can have either an even or odd number of ports.
- Every network in the cascade can have a different number of ports.

Examples

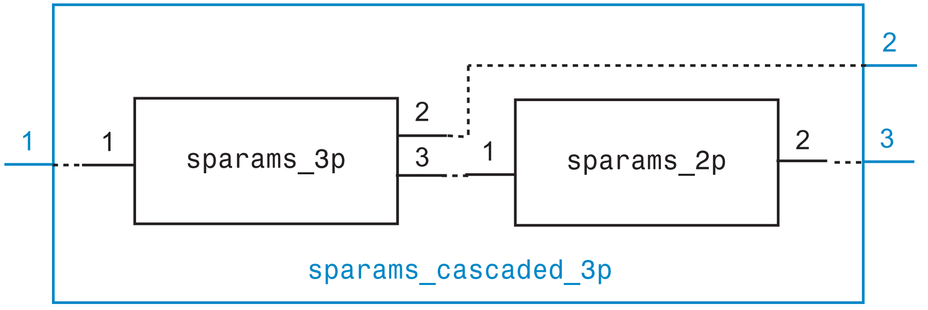
Assemble a 2-port cascaded network from two sets of 2-port S-parameters:

```
%Create two sets of 2-port S-parameters
ckt1 = read(rfckt.amplifier,'default.s2p');
ckt2 = read(rfckt.passive,'passive.s2p');
freq = [2e9 2.1e9];
analyze(ckt1,freq);
analyze(ckt2,freq);
sparams_2p_1 = ckt1.AnalyzedResult.S_Parameters;
sparams_2p_2 = ckt2.AnalyzedResult.S_Parameters;
```

```
%Cascade the S-parameters
sparams_cascaded_2p = ...
  cascadesparams(sparams_2p_1,sparams_2p_2)
```

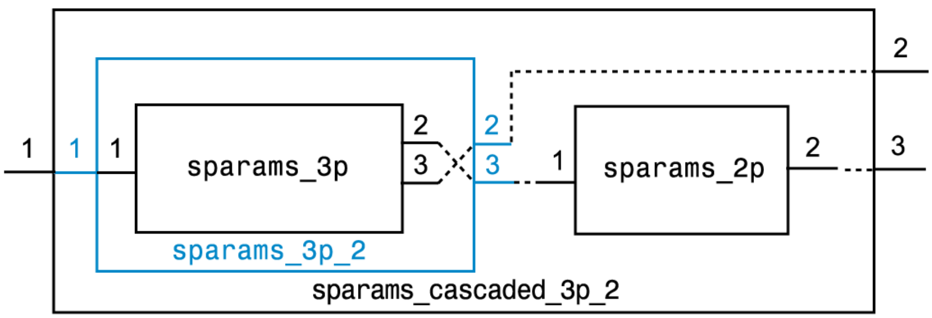
Assemble a 3-port cascaded network from a set of 3-port S-parameters and a set of 2-port S-parameters:

```
% Create one set of 3-port S-parameters
% and one set of 2-port S-parameters
ckt1 = read(rfckt.passive,'default.s3p');
ckt2 = read(rfckt.amplifier,'default.s2p');
freq = [2e9 2.1e9];
analyze(ckt1,freq);
analyze(ckt2,freq);
sparams_3p = ckt1.AnalyzedResult.S_Parameters;
sparams_2p = ckt2.AnalyzedResult.S_Parameters;
%Cascade the two sets by connecting one port between them
Kconn = 1
sparams_cascaded_3p = ...
  cascadesparams(sparams_3p,sparams_2p,Kconn)
```



Assemble a 3-port cascaded network from a set of 3-port S-parameters and a set of 2-port S-parameters, connecting the second port of the 3-port network to the first port of the 2-port network:

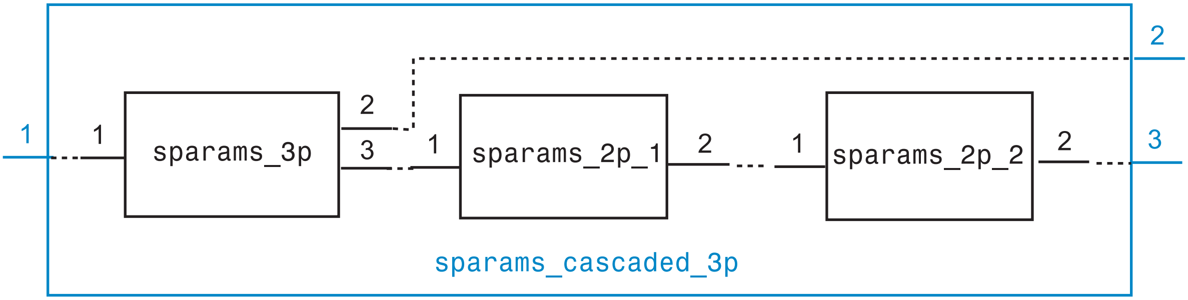
```
ckt1 = read(rfckt.passive,'default.s3p');
ckt2 = read(rfckt.amplifier,'default.s2p');
freq = [2e9 2.1e9];
analyze(ckt1,freq);
analyze(ckt2,freq);
sparams_3p = ckt1.AnalyzedResult.S_Parameters;
sparams_2p = ckt2.AnalyzedResult.S_Parameters;
%Reorder the second and third ports of the 3-port network
sparams_3p_2 = snp2smp(sparams_3p,50,[1 3 2])
%Cascade the two sets by connecting one port between them
Kconn = 1
sparams_cascaded_3p_2 = cascadesparams(sparams_3p_2,...
     sparams_2p,Kconn)
```



Assemble a 3-port cascaded network from a set of 3-port S-parameters and two sets of 2-port S-parameters:

```
ckt1 = read(rfckt.passive,'default.s3p');
ckt2 = read(rfckt.amplifier,'default.s2p');
ckt3 = read(rfckt.passive,'passive.s2p');
freq = [2e9 2.1e9];
analyze(ckt1,freq);
analyze(ckt2,freq);
analyze(ckt3,freq);
sparams_3p = ckt1.AnalyzedResult.S_Parameters;
```

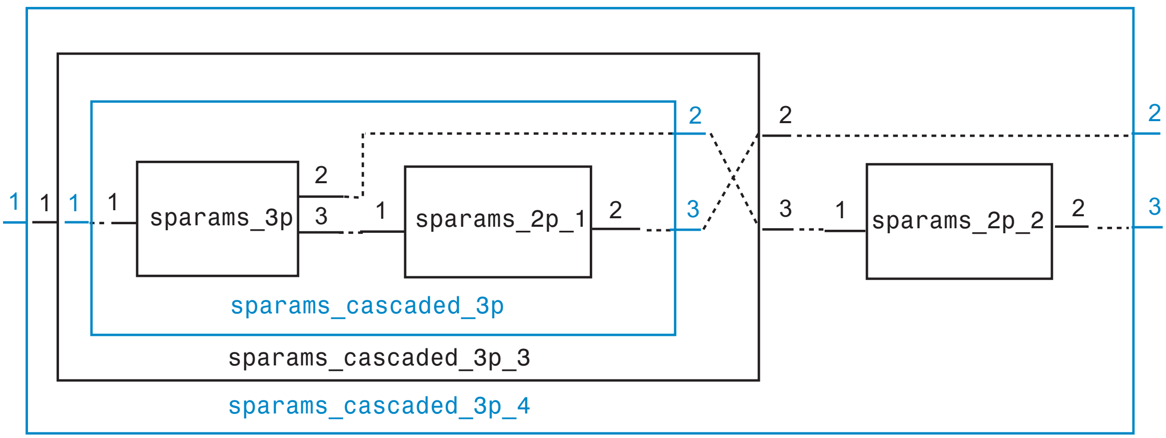
```
sparams_2p_1 = ckt2.AnalyzedResult.S_Parameters;
sparams_2p_2 = ckt3.AnalyzedResult.S_Parameters;
%Connect one port between each set of adjacent networks
Kconn = [1 1]
sparams_cascaded_3p_3 = cascadesparams(sparams_3p,...
     sparams_2p_1,sparams_2p_2,Kconn)
```



Assemble a 3-port cascaded network from a set of 3-port S-parameters and two sets of 2-port S-parameters, connecting the 3-port network to both 2-port networks:

```
ckt1 = read(rfckt.passive,'default.s3p');
ckt2 = read(rfckt.amplifier,'default.s2p');
ckt3 = read(rfckt.passive,'passive.s2p');
freq = [2e9 2.1e9];
analyze(ckt1,freq);
analyze(ckt2,freq);
analyze(ckt3,freq);
sparams_3p = ckt1.AnalyzedResult.S_Parameters;
sparams_2p_1 = ckt2.AnalyzedResult.S_Parameters;
sparams_2p_2 = ckt3.AnalyzedResult.S_Parameters;
%Cascade sparams_3p and sparams_2p_1
%by connecting one port between them
Kconn = 1
sparams_cascaded_3p = cascadesparams(...
  sparams_3p, ...
```

```
  sparams_2p_1, ...
  Kconn)
%Reorder the second and third ports of the 3-port network
sparams_cascaded_3p_3 = snp2smp(...
  sparams_cascaded_3p, ...
  50, ...
  [1 3 2])
%Cascade sparams_3p and sparams_2p_2
%by connecting one port between them
sparams_cascaded_3p_4 = cascadesparams(...
  sparams_cascaded_3p_3, ...
  sparams_2p_2, ...
  Kconn)
```



See Also

deembedsparams | rfckt.cascade | s2t | t2s

Purpose

Copy circuit or data object

Syntax

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

Description

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

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

Alternatives

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

See Also

`analyze`

Purpose

De-embed 2-port S-parameters

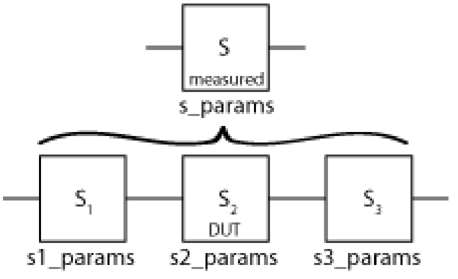
Syntax

```
s2_params = deembedsparams(s_params,s1_params,s3_params)
hs2 = deembedsparams(hs,hs1,hs3)
```

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`. `s2_params` is a 2-by-2-by-Q array containing the de-embedded S-parameters. This function is ideal for situations in which the S-parameters of a device under test (DUT) must be de-embedded from S-parameters obtained through measurement.

The function assumes the configuration of the cascade shown in the following illustration.



Each of the input networks is a 2-port network described by a 2-by-2-by-Q array of S-parameters. The function assumes that all networks in the cascade have the same reference impedance and are measured at the same frequencies..

`hs2 = deembedsparams(hs,hs1,hs3)` de-embeds `hs2` from the chain `hs`. In this syntax, inputs and outputs are RF Toolbox S-parameter objects. The function checks that the `Frequencies` and `Impedance` properties are the same for all three inputs.

See Also

`cascadesparams` | `rfckt.cascade`

g2h

Purpose

Convert hybrid g-parameters to hybrid h-parameters

Syntax

```
h_params = g2h(g_params)
```

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* 2-port g-parameters. `h_params` is a complex 2-by-2-by-*M* array, representing *M* 2-port h-parameters.

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-12.

Examples

Convert g-parameters to h-parameters:

```
%Define a matrix of g-parameters.
g_11 = -6.55389515512306e-007 + 6.67541048071651e-006i;
g_12 =     -0.999823389146385 -  0.000246785162909241i;
g_21 =       1.00011560038266 -  0.000129304649930592i;
g_22 =      0.314441556185771 +      2.51960941000598i;
g_params = [g_11,g_12; g_21,g_22];
%Convert to h-parameters
h_params = g2h(g_params);
```

See Also

`h2g`

Purpose

Convert reflection coefficient to impedance

Syntax

```
z = gamma2z(gamma)
z = gamma2z(gamma,z0)
```

Description

`z = gamma2z(gamma)` converts the reflection coefficient `gamma` to the impedance `z` using a reference impedance Z_0 of 50 ohms.

`z = gamma2z(gamma,z0)` converts the reflection coefficient `gamma` to the impedance `z` by:

- Computing the normalized impedance.
- Multiplying the normalized impedance by the reference impedance Z_0.

Algorithms

The following equation shows this conversion:

$$Z = Z_0 * \left(\frac{1+\Gamma}{1-\Gamma} \right)$$

Examples

Calculate impedance from given reference impedence and reflection coefficient values:

```
z0 = 50;
gamma = 1/3;
z = gamma2z(gamma,z0)
```

See Also

`gammain` | `gammaout` | `z2gamma`

gammain

Purpose

Input reflection coefficient of 2-port network

Syntax

```
coefficient = gammain(s_params,z0,zl)
coefficient = gammain(hs,zl)
```

Description

`coefficient = gammain(s_params,z0,zl)` calculates the input reflection coefficient of a 2-port network. `s_params` is a complex 2-by-2-by-M array, representing M 2-port S-parameters. `z0` is the reference impedance Z_0; its default value is 50 ohms. `zl` is the load impedance Z_l; its default value is also 50 ohms. `coefficient` is an M-element complex vector.

`coefficient = gammain(hs,zl)` calculates the input reflection coefficient of the 2-port network represented by the S-parameter object `hs`.

Algorithms

`gammain` uses the formula

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

where

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

Examples

Calculate the input reflection coefficients at each index of an S-parameter array:

```
ckt = read(rfckt.amplifier,'default.s2p');
s_params = ckt.NetworkData.Data;
z0 = ckt.NetworkData.Z0;
zl = 100;
coefficient = gammain(s_params,z0,zl);
```

See Also

`gamma2z` | `gammaml` | `gammams` | `gammaout` | `vswr`

Purpose

Load reflection coefficient of 2-port network

Syntax

```
coefficient = gammaml(s_params)
coefficient = gammaml(hs)
```

Description

`coefficient = gammaml(s_params)` calculates the load reflection coefficient of a 2-port network required for simultaneous conjugate match.

`s_params` is a complex 2-by-2-by-M array, representing M 2-port S-parameters, S_{ij}. `coefficient` is an M-element complex vector.

`coefficient = gammaml(hs)` calculates the load reflection coefficient of the 2-port network represented by the S-parameter object `hs`.

Algorithms

The function calculates `coefficient` using the equation

$$\Gamma_{ML} = \frac{B_2 \pm \sqrt{B_2{}^2 - 4\left|C_2{}^2\right|}}{2C_2}$$

where

$$B_2 = 1 - \left|S_{11}{}^2\right| + \left|S_{22}{}^2\right| - \left|\Delta^2\right|$$

$$C_2 = S_{22} - \Delta \cdot S_{11}^*$$

$$\Delta = S_{11}S_{22} - S_{12}S_{21}$$

Examples

Calculate the load reflection coefficient using network data from a file:

```
ckt = read(rfckt.amplifier,'default.s2p');
s_params = ckt.NetworkData.Data;
coefficient = gammaml(s_params);
```

See Also

`gammain` | `gammams` | `gammaout` | `stabilityk`

gammams

Purpose

Source reflection coefficient of 2-port network

Syntax

```
coefficient = gammams(s_params)
coefficient = gammams(hs)
```

Description

`coefficient = gammams(s_params)` calculates the source reflection coefficient of a 2-port network required for simultaneous conjugate match. `s_params` is a complex 2-by-2-by-M array, representing M 2-port S-parameters. `coefficient` is an M-element complex vector.

`coefficient = gammams(hs)` calculates the source reflection coefficient of the 2-port network represented by the S-parameter object `hs`.

Algorithms

The function calculates `coefficient` using the equation

$$\Gamma_{MS} = \frac{B_1 \pm \sqrt{B_1{}^2 - 4\left|C_1{}^2\right|}}{2C_1}$$

where

$$B_1 = 1 + \left|S_{11}{}^2\right| - \left|S_{22}{}^2\right| - \left|\Delta^2\right|$$

$$C_1 = S_{11} - \Delta \cdot S_{22}^*$$

$$\Delta = S_{11}S_{22} - S_{12}S_{21}$$

Examples

Calculate the source reflection coefficient using network data from a file:

```
ckt = read(rfckt.amplifier,'default.s2p');
s_params = ckt.NetworkData.Data;
coefficient = gammams(s_params);
```

See Also

`gammain` | `gammaml` | `gammaout` | `stabilityk`

Purpose

Output reflection coefficient of 2-port network

Syntax

```
coefficient = gammaout(s_params,z0,zs)
coefficient = gammaout(hs,zs)
```

Description

`coefficient = gammaout(s_params,z0,zs)` calculates the output reflection coefficient of a 2-port network.

`s_params` is a complex 2-by-2-by-M array, representing M 2-port S-parameters. `z0` is the reference impedance Z_0; its default is 50 ohms. `zs` is the source impedance Z_s; its default is also 50 ohms. `coefficient` is an M-element complex vector.

`coefficient = gammaout(hs,zs)` calculates the output reflection coefficient of the 2-port network represented by the S-parameter object `hs`.

Algorithms

The function calculates `coefficient` using the equation

$$\Gamma_{out} = S_{22} + \frac{S_{12}S_{21}\Gamma_S}{1 - S_{11}\Gamma_S}$$

where

$$\Gamma_S = \frac{Z_s - Z_0}{Z_s + Z_0}$$

Examples

Calculate the output reflection coefficient using network data from a file:

```
ckt = read(rfckt.amplifier,'default.s2p');
s_params = ckt.NetworkData.Data;
z0 = ckt.NetworkData.Z0;
zs = 100;
coefficient = gammaout(s_params,z0,zs);
```

See Also

`gamma2z` | `gammain` | `gammaml` | `gammams` | `vswr`

getdata

Purpose Data object containing analyzed result of 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 there is no analysis data, `getdata` displays an error message.

Note Before calling `getdata`, use the `analyze` function to perform a frequency domain analysis for the circuit (`rfckt`) object. Perform this action for all circuit objects except `rfckt.amplifier`, `rfckt.datafile`, and `rfckt.mixer`. When you create an `rfckt.amplifier`, `rfckt.datafile`, or `rfckt.mixer` object by reading data from a file, RF Toolbox software automatically creates an `rfdata.data` object. RF Toolbox 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 the `rfdata.data` object.

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* 2-port hybrid h-parameters. `abcd_params` is a complex 2-by-2-by-*M* array, representing *M* 2-port ABCD-parameters.

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-12.

Examples

Convert h-parameters to ABCD-parameters:

```
%Define a matrix of h-parameters.
h_11 =      0.314441556185771 +      2.51960941000598i;
h_12 =      0.999823389146385 -  0.000246785162909241i;
h_21 =     -1.000115600382660 -  0.000129304649930592i;
h_22 = -6.55389515512306e-007 + 6.67541048071651e-006i;
h_params = [h_11,h_12; h_21,h_22];
%Convert to ABCD-parameters
abcd_params = h2abcd(h_params);
```

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,z0)` 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 2-port h-parameters. `g_params` is a complex 2-by-2-by-M array, representing M 2-port g-parameters.

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-12.

Examples

Convert h-parameters to g-parameters:

```
%Define a matrix of h-parameters.
h_11 =      0.314441556185771 +      2.51960941000598i;
h_12 =      0.999823389146385 -  0.000246785162909241i;
h_21 =     -1.000115600382660 -  0.000129304649930592i;
h_22 = -6.55389515512306e-007 + 6.67541048071651e-006i;
h_params = [h_11,h_12; h_21,h_22];
%Convert to g-parameters
g_params = h2g(h_params);
```

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 `s_params`. The `h_params` input is a complex 2-by-2-by-M array, representing M 2-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 2-port S-parameters.

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-12.

Examples

Convert h-parameters to S-parameters:

```
%Define a matrix of h-parameters.
h_11 =      0.314441556185771 +      2.51960941000598i;
h_12 =      0.999823389146385 -  0.000246785162909241i;
h_21 =     -1.000115600382660 -  0.000129304649930592i;
h_22 = -6.55389515512306e-007 + 6.67541048071651e-006i;
h_params = [h_11,h_12; h_21,h_22];
%Convert to S-parameters
s_params = h2s(h_params);
```

See Also

`abcd2s` | `h2abcd` | `h2y` | `h2z` | `y2s` | `z2s`

Purpose

Convert hybrid h-parameters to Y-parameters

Syntax

```
y_params = h2y(h_params)
```

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* 2-port hybrid h-parameters. `y_params` is a complex 2-by-2-by-*M* array, representing *M* 2-port Y-parameters.

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-12.

Examples

Convert h-parameters to Y-parameters:

```
%Define a matrix of h-parameters.
h_11 =      0.314441556185771 +      2.51960941000598i;
h_12 =      0.999823389146385 -  0.000246785162909241i;
h_21 =     -1.000115600382660 -  0.000129304649930592i;
h_22 = -6.55389515512306e-007 + 6.67541048071651e-006i;
h_params = [h_11,h_12; h_21,h_22];
%Convert to Y-parameters
y_params = h2y(h_params);
```

See Also

`abcd2z` | `h2abcd` | `h2s` | `h2y` | `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* 2-port hybrid h-parameters. `z_params` is a complex 2-by-2-by-*M* array, representing *M* 2-port Z-parameters.

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see “RF Network Parameter Objects” on page 2-12.

Examples

Convert h-parameters to Z-parameters:

```
%Define a matrix of h-parameters.
h_11 =      0.314441556185771 +      2.51960941000598i;
h_12 =      0.999823389146385 -  0.000246785162909241i;
h_21 =     -1.000115600382660 -  0.000129304649930592i;
h_22 = -6.55389515512306e-007 + 6.67541048071651e-006i;
h_params = [h_11,h_12; h_21,h_22];
%Convert to Z-parameters
z_params = h2z(h_params);
```

See Also

`abcd2z` | `h2abcd` | `h2s` | `h2y` | `s2z` | `y2z` | `z2h`

ispassive

Purpose Check passivity of N-port S-parameters

Syntax

```
[flag,index_non_passive] = ispassive(s_params)
[flag,index_non_passive] = ispassive(hs)
```

Description

`[flag,index_non_passive] = ispassive(s_params)` checks the passivity of `sparams`, an array of N-port S-parameters. If all the S-parameters are passive, `ispassive` sets `flag` equal to `1` (`true`). Otherwise, `flag` is equal to `0` (`false`). `index_non_passive` is a vector of indices corresponding to the non-passive S-parameters in `sparams`. If `flag` is `true`, `index_non_passive` is empty.

`[flag,index_non_passive] = ispassive(hs)` checks the passivity of the network represented by the S-parameter object `hs`.

Examples

Check the passivity of an S-parameters array; separate non-passive S-parameters into a new array:

```
%Read a Touchstone data file
ckt = read(rfckt.passive,'passive.s2p')
%Check the passivity
data = ckt.AnalyzedResult
[result,index] = ispassive(data.S_Parameters);
%Get the non-passive S-parameters
if ~(result)
     AllNonPassiveSparams = data.S_Parameters(:,:,index);
     FirstNonPassiveSparams = AllNonPassiveSparams(:,:,1)
end
```

See Also

`rationalfit` | `rfmodel.rational.ispassive` | `s2tf` | `snp2smp`

Purpose

Enforce passivity of S-parameters

Syntax

```
sparams_passive = makepassive(sparams)
hsp = makepassive(hs)
```

Description

`sparams_passive = makepassive(sparams)` makes an array of S-parameters passive. Both `sparams` and `sparams_passive` are N-by-N-by-M arrays representing M N-port S-parameters.

The `makepassive` function enforces the following conditions on `sparams`:

$$S(-j\omega) = S^{*}(j\omega)$$
$$\|S(j\omega)\|_2 \le 1$$

The notation $\|S\|_2$ represents the 2-norm, or singular-value decomposition, of S.

`hsp = makepassive(hs)` creates a passive S-parameter object from the S-parameter object `hs`.

Input Arguments

hs

N-port S-parameters, specified as an RF Toolbox S-parameter object.

sparams

`sparams` can represent either an active network or a passive network. To check if `sparams` is passive, use the `ispassive` function.

Output Arguments

sparams_passive

The `makepassive` function uses a purely mathematical method to calculate `sparams_passive`. As a result, the array `sparams_passive` does not represent the same network as `sparams`. `sparams` and `sparams_passive` do not represent the same network unless `sparams` and `sparams_passive` are equal.

The more closely `sparams` represents a passive network, the better the approximation `sparams_passive` is to that network. Therefore, `makepassive` generates the most realistic results when `sparams` is active only due to small numerical errors.

Examples

Enforce passivity of the S-parameters that represent a passive network:

```
ckt = read(rfckt.passive,'passive.s2p');
sparams = ckt.NetworkData.Data;
Is_Passive = ispassive(sparams)
sparams_new = makepassive(sparams);
Is_Passive = ispassive(sparams_new)
ckt.NetworkData.Data = sparams_new;
```

See Also

`ispassive`

Purpose

Power gain of 2-port network

Syntax

```
g = powergain(s_params,z0,zs,zl,'Gt')
g = powergain(s_params,z0,zs,'Ga')
g = powergain(s_params,z0,zl,'Gp')
g = powergain(s_params,'Gmag')
g = powergain(s_params,'Gmsg')

g = powergain(hs,zs,zl,'Gt')
g = powergain(hs,zs,'Ga')
g = powergain(hs,zl,'Gp')
g = powergain(hs,'Gmag')
g = powergain(hs,'Gmsg')
```

Description

`g = powergain(s_params,z0,zs,zl,'Gt')` calculates the transducer power gain of the 2-port network `s_params`.

`g = powergain(s_params,z0,zs,'Ga')` calculates the available power gain of the 2-port network.

`g = powergain(s_params,z0,zl,'Gp')` calculates the operating power gain of the 2-port network.

`g = powergain(s_params,'Gmag')` calculates the maximum available power gain of the 2-port network.

`g = powergain(s_params,'Gmsg')` calculates the maximum stable gain of the 2-port network.

`g = powergain(hs,zs,zl,'Gt')` calculates the transducer power gain of the network represented by the S-parameter object `hs`.

`g = powergain(hs,zs,'Ga')` calculates the available power gain of the network represented by the S-parameter object `hs`.

`g = powergain(hs,zl,'Gp')` calculates the operating power gain of the network represented by the S-parameter object `hs`.

`g = powergain(hs,'Gmag')` calculates the maximum available power gain of the network represented by the S-parameter object `hs`.

`g = powergain(hs,'Gmsg')` calculates the maximum stable gain of the network represented by the S-parameter object `hs`.

Input Arguments

hs - 2-port S-parameters

S-parameter object

2-port S-parameters, specified as an RF Toolbox S-parameter object.

s_params - 2-port S-parameters

array of complex numbers

2-port S-parameters, specified as a complex 2-by-2-by-N array.

z0 - Reference impedance

positive scalar | 50 (default)

Reference impedance in ohms, specified as a positive scalar. If the first input argument is an S-parameter object `hs`, the function uses `hs.Impedance` for the reference impedance.

zl - Load impedance

positive scalar | 50 (default)

Load impedance in ohms, specified as a positive scalar.

zs - Source impedance

positive scalar | 50 (default)

Source impedance in ohms, specified as a positive scalar.

Output Arguments

g - Power gain
vector

Unitless power gain values, returned as a vector. To obtain power gain in decibels, use `10*log10(g)`.

If the specified type of power gain is undefined for one or more of the specified S-parameter values in `s_params`, the `powergain` function returns `NaN`. As a result, `g` is either `NaN` or a vector that contains one or more `NaN` entries.

Examples

Calculate power gains for a sample 2-port network:

```
s11 = 0.61*exp(j*165/180*pi);
s21 = 3.72*exp(j*59/180*pi);
s12 = 0.05*exp(j*42/180*pi);
s22 = 0.45*exp(j*(-48/180)*pi);
sparam = [s11 s12; s21 s22];
z0 = 50;
zs = 10 + j*20;
zl = 30 - j*40;
%Calculate the transducer power gain of the network
Gt = powergain(sparam,z0,zs,zl,'Gt')
%Calculate the available power gain of the network
Ga = powergain(sparam,z0,zs,'Ga')
%Calculate the operating power gain of the network
Gp = powergain(sparam,z0,zl,'Gp')
%Calculate the maximum available power gain of the network
Gmag = powergain(sparam,'Gmag')
%Calculate the maximum stable power gain of the network
Gmsg = powergain(sparam,'Gmsg')
```

See Also

`s2tf`

Purpose Approximate data using stable rational function

Syntax

```
h = rationalfit(freq,data)
h = rationalfit(freq,data,tol)
h = rationalfit(___,Name,Value)
```

Description

`h = rationalfit(freq,data)` fits a rational function model of the form

$$F(s) = \sum_{k=1}^{n} \frac{C_k}{s - A_k} + D, \quad s = j \cdot 2\pi f$$

to the complex vector `data` over the frequency values in the positive vector `freq`. The function returns a handle to the rational function model object, `h`, with properties `A`, `C`, `D`, and `Delay`.

`h = rationalfit(freq,data,tol)` fits a rational function model to complex data and constrains the error of the fit according to the optional input argument `tol`.

`h = rationalfit(___,Name,Value)` fits a rational function with additional options specified by one or more `Name,Value` pair arguments. These arguments offer finer control over the performance and accuracy of the fitting algorithm.

Examples

Stable rational-function approximation of S-parameter data

Fit a rational function model to S-parameter data, and compare the results by plotting the model against the data.

1

Read the S-parameter data into an RF data object.

```
orig_data = read(rfdata.data,'passive.s2p');
freq = orig_data.Freq;
data = orig_data.S_Parameters(1,1,:);
```

2

Fit a rational function to the data using `rationalfit`.

```
fit_data = rationalfit(freq,data);
```

3

Compute the frequency response of the rational function using `freqresp`.

```
[resp,freq] = freqresp(fit_data,freq);
```

4

Plot the magnitude of the original data against the rational function approximation. S_{11} data appears in blue, and the rational function appears in red. Scaling the frequency values by `1e9` converts them to units of GHz.

```
figure;
title('Rational fitting of S11 magnitude')
plot(orig_data,'S11','dB');
hold on;
plot(freq/1e9,db(resp),'r');
```

5

Plot the angle of the original data against the rational function approximation.

```
figure;
title('Rational fitting of S11 angle')
plot(orig_data,'S11','Angle (radians)');
hold on;
plot(freq/1e9,unwrap(angle(resp)),'r');
```

Tips

To see how well the model fits the original data, use the `freqresp` function to compute the frequency response of the model. Then, plot the original data and the frequency response of the rational function model. For more information, see the `freqresp` reference page or the examples in the next section.

Input Arguments

freq - Frequencies

vector of positive numbers

Frequencies over which the function fits a rational model, specified as a vector of length M.

data - Data to fit

vector of complex numbers | N-by-N-by-M array of complex numbers

Data to be approximated, specified as an N-by-N-by-M array of complex numbers. The function fits N^2 rational functions to the data along the M (frequency) dimension.

tol - Error tolerance

-40 (default) | scalar

Error tolerance ε, specified as a scalar in units of dB.. The error-fitting equation is

$$10^{\varepsilon/20} \geq \frac{\sqrt{\sum_{k=1}^{n} \left| F_0\{f_k\} - F(s) \right|^2}}{\sqrt{\sum_{k=1}^{n} \left| F_0\{f_k\} \right|^2}}$$

where

- ε is the specified value of `tol`.
- F_0 is the value of the original data (`data`) at the specified frequency f_k (`freq`).
- F is the value of the rational function at $s = j2\pi f$.

`rationalfit` computes the relative error as a vector containing the dependent values of the fit data. If the model does not fit the original data within the specified tolerance, a warning message appears.

Default: `-40`

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside single quotes (`' '`). You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

'DelayFactor' - Delay factor

scalar from 0 to 1 | 0 (default)

Scaling factor that controls the amount of delay to fit to the data, specified as the comma-separated pair consisting of `'DelayFactor'` and a scalar between 0 and 1 inclusive. The `Delay` parameter, τ, of the rational function object is equal to the specified value of `'DelayFactor'` times an estimate of the group delay of the data. If the original data has delay, increasing this value might allow `rationalfit` to fit the data with a lower-order model.

'IterationLimit' - Maximum number of iterations

integer | 12 (default)

Number of vector-fitting iterations performed during pole search, specified as the comma-separated pair consisting of `'IterationLimit'` and a positive integer. Increasing the limit extends the time that the algorithm takes to produce a fit, but may produce more accurate results.

'NPoles' - Number of poles

`[0 48]` (default) | nonnegative integer | vector of two nonnegative integers

Number of poles A_k of the rational function, specified as the comma-separated pair consisting of `'NPoles'` and an integer n or range of possible values of n.

To help `rationalfit` produce an accurate fit, choose a maximum value of `npoles` greater than or equal to twice the number of peaks on a plot of the data in the frequency domain.

After completing a rational fit, the function removes coefficient sets whose residues (C_k) are zero. Thus, when you specify a range for `npoles`, the number of poles of the fit may be less than `npoles(1)`.

'TendsToZero' - Asymptotic behavior of fit

`true` (default) | `false`

Asymptotic behavior of the rational function as frequency approaches infinity, specified as the comma-separated pair consisting of `'TendsToZero'` and a logical value. When this argument is `true`, the resulting rational function variable D is zero, and the function tends to zero. A value of `false` allows a nonzero value for `D`.

'Tolerance' - Error tolerance

-40 (default) | scalar

Error tolerance ε, specified as the comma-separated pair consisting of `'Tolerance'` and a scalar in units of dB. The error-fitting equation is

$$10^{\varepsilon/20} \geq \frac{\sqrt{\sum_{k=1}^{n} \left|F_0\{f_k\} - F(s)\right|^2}}{\sqrt{\sum_{k=1}^{n} \left|F_0\{f_k\}\right|^2}}$$

where

- ε is the specified tolerance.
- F_0 is the value of the original data (`data`) at the specified frequency f_k (`freq`).

- F is the value of the rational function at $s = j2\pi f$.

If the model does not fit the original data within the specified tolerance, the function throws a warning.

'WaitBar' - Graphical wait bar

`true` | `false` (default)

Logical value that toggles display of the graphical wait bar during fitting, specified as the comma-separated pair consisting of `'WaitBar'` and either `true` or `false`. The `true` setting shows the graphical wait bar, and the `false` setting hides it. If you expect `rationalfit` to take a long time, and you want to monitor its progress, set `'WaitBar'` to `true`.

'Weight' - Weighting of data

`ones(size(`*`freq`*`))` (default) | vector of positive numbers

Weighting of the data at each frequency, specified as the comma-separated pair consisting of `'Weight'` and a vector of positive numbers. Each entry in `weight` corresponds to a frequency in `freq`, so the length of `weight` must be equal to the length of `freq`. Increasing the weight at a particular frequency improves the model fitting at that frequency. Specifying a weight of `0` at a particular frequency causes `rationalfit` to ignore the corresponding data point.

Output Arguments

h - Rational model object

`rfmodel.rational` object

One or more rational models, returned as an N-by-N `rfmodel.rational` object. The number of dimensions in `data` determines the dimensionality of `h`.

References

B. Gustavsen and A. Semlyen, “Rational approximation of frequency domain responses by vector fitting,” IEEE Trans. Power Delivery, Vol. 14, No. 3, pp. 1052–1061, July 1999.

R. Zeng and J. Sinsky, “Modified Rational Function Modeling Technique for High Speed Circuits,” IEEE MTT-S Int. Microwave Symp. Dig., San Francisco, CA, June 11–16, 2006.

See Also

`freqresp` | `rfmodel.rational` | `s2tf` | `timeresp` | `writeva`

Purpose

Open RF Analysis Tool (RF Tool)

Syntax

```
rftool
```

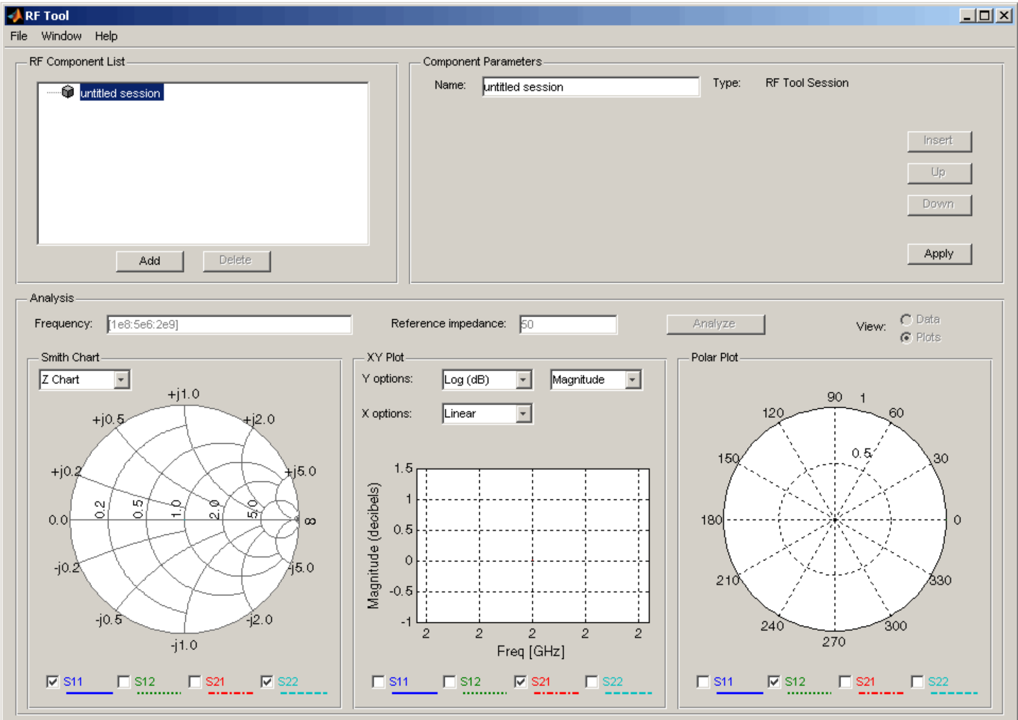
Description

`rftool` opens the RF Tool interface. 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 RF Tool sessions for later use.

For more information, see "RF Tool" on page 5-2.

The following figure shows the RF Tool in its default state.


RF Tool
File Window Help
RF Component List
untitled session
Add
Delete
Component Parameters
Name: untitled session
Type: RF Tool Session
Insert
Up
Down
Apply
Analysis
Frequency: [1e8:5e6:2e9]
Reference impedance: 50
Analyze
View: Data Plots
Smith Chart
Z Chart
XY Plot
Y options: Log (dB) Magnitude
X options: Linear
Magnitude (decibels)
Freq [GHz]
Polar Plot
S11 S12 S21 S22

Purpose

Convert RLGC transmission line parameters to S-parameters

Syntax

```
s_params = rlgc2s(R,L,G,C,length,freq,z0)
s_params = rlgc2s(R,L,G,C,length,freq)
```

Description

`s_params = rlgc2s(R,L,G,C,length,freq,z0)` transforms RLGC transmission line parameter data into S-parameters.

`s_params = rlgc2s(R,L,G,C,length,freq)` transforms RLGC transmission line parameter data into S-parameters with a reference impedance of 50 Ω.

Input Arguments

R

Specify an N-by-N-by-M array of distributed resistances, in units of Ω/m. The N-by-N matrices must be real symmetric, the diagonal terms must be nonnegative, and the off-diagonal terms must be nonnegative.

L

Specify an N-by-N-by-M array of distributed inductances, in units of H/m. The N-by-N matrices must be real symmetric, the diagonal terms must be positive, and the off-diagonal terms must be nonnegative.

G

Specify an N-by-N-by-M array of distributed conductances, in units of S/m. The N-by-N matrices must be real symmetric, the diagonal terms must be nonnegative, and the off-diagonal terms must be nonpositive.

C

Specify an N-by-N-by-M array of distributed capacitances, in units of F/m. The matrices must be real symmetric, the diagonal terms must be positive, and the off-diagonal terms must be nonpositive.

length

Specify the length of the transmission line in meters.

freq

Specify the vector of M frequencies over which the transmission line parameters are defined.

z0

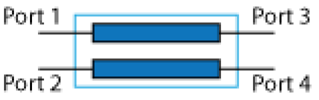
Specify the reference impedance of the resulting S-parameters in ohms.

Default: `50`

Output Arguments

s_params

The output is a $2N$-by-$2N$-by-M array of S-parameters. The following figure describes the port ordering convention of the output.



$$\left[\begin{array}{cc:cc} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} \\ \hdashline S_{31} & S_{32} & S_{33} & S_{34} \\ S_{41} & S_{42} & S_{43} & S_{44} \end{array}\right]$$

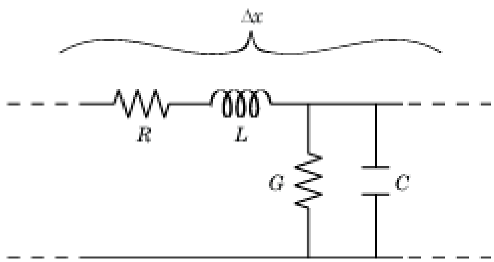
This port ordering convention assumes that:

- Each $2N$-by-$2N$ matrix consists of N input terminals and N output terminals.
- The first N ports (1 through N) of the S-parameter matrix are input ports.
- The last N ports (N + 1 through $2N$) are output ports.

To reorder ports after using this function, use the `snp2smp` function.

Definitions

The following figure illustrates the RLGC transmission line model.



The representation consists of:

- The distributed resistance, R, of the conductors, represented by a series resistor.
- The distributed inductance, L, represented by a series inductor.
- The distributed conductance, G,
- The distributed capacitance, C, between the two conductors, represented by a shunt capacitor.

RLGC component units are all per unit length Δx.

Examples

Convert RLGC transmission line parameters into S-parameters:

```
%Define transmission line variables
length = 1e-3;
freq = 1e9;
z0 = 50;
R = 50;
L = 1e-9;
G = .01;
C = 1e-12;
% Calculate S-parameters
s_params = rlgc2s(R,L,G,C,length,freq,z0)
```

References

Bhatti, A. Aziz. "A computer Based Method for Computing the N-Dimensional Generalized ABCD Parameter Matrices of N-Dimensional Systems with Distributed Parameters." *Southeastern*

Symposium on System Theory. SSST, 22nd Conference, 11–13 March 1990, pp. 590–593.

See Also s2rlgc

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 $2N$-by-$2N$-by-M array, representing M $2N$-port S-parameters. `z0` is the reference impedance; its default is 50 ohms. `abcd_params` is a complex $2N$-by-$2N$-by-M array, representing M 2-port ABCD-parameters. The output ABCD-parameter matrices have distinct A, B, C, and D submatrices:

$$\begin{bmatrix} [A] & [B] \\ [C] & [D] \end{bmatrix}$$

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-12.

Examples

Convert S-parameters to ABCD-parameters:

```
%Define a matrix of S-parameters
s_11 = 0.61*exp(j*165/180*pi);
s_21 = 3.72*exp(j*59/180*pi);
s_12 = 0.05*exp(j*42/180*pi);
s_22 = 0.45*exp(j*(-48/180)*pi);
s_params = [s_11 s_12; s_21 s_22];
z0 = 50;
%Convert to ABCD-parameters
abcd_params = s2abcd(s_params,z0)
```

See Also

`abcd2s` | `h2abcd` | `s2h` | `s2y` | `s2z` | `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* 2-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* 2-port hybrid h-parameters.

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-12.

Examples

Convert S-parameters to h-parameters:

```
%Define a matrix of S-parameters
s_11 = 0.61*exp(j*165/180*pi);
s_21 = 3.72*exp(j*59/180*pi);
s_12 = 0.05*exp(j*42/180*pi);
s_22 = 0.45*exp(j*(-48/180)*pi);
s_params = [s_11 s_12; s_21 s_22];
z0 = 50;
%Convert to h-parameters
h_params = s2h(s_params,z0)
```

See Also

`h2s`

Purpose

Convert S-parameters to RLGC transmission line parameters

Syntax

```
rlgc_params = s2rlgc(s_params,length,freq,z0)
rlgc_params = s2rlgc(s_params,length,freq)
```

Description

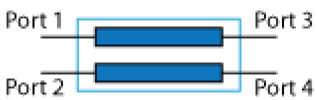
`rlgc_params = s2rlgc(s_params,length,freq,z0)` transforms multi-port S-parameter data into a frequency-domain representation of an RLGC transmission line.

`rlgc_params = s2rlgc(s_params,length,freq)` transforms multi-port S-parameter data into RLGC transmission line parameters using a reference impedance of 50 Ω.

Input Arguments

s_params

Specify a $2N$-by-$2N$-by-M array of S-parameters to transform into RLGC transmission line parameters. The following figure describes the port ordering convention assumed by the function.



$$\left[\begin{array}{cc:cc} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} \\ \hdashline S_{31} & S_{32} & S_{33} & S_{34} \\ S_{41} & S_{42} & S_{43} & S_{44} \end{array}\right]$$

The function assumes that:

- Each $2N$-by-$2N$ matrix consists of N input terminals and N output terminals.
- The first N ports (1 through N) of the S-parameter matrix are input ports.
- The last N ports ($N + 1$ through $2N$) are output ports.

To reorder ports before using this function, use the `snp2smp` function.

length

Specify the length of the transmission line in meters.

freq

Specify the vector of M frequencies over which the S-parameter array `s_params` is defined.

z0

Specify the reference impedance of the S-parameters in ohms.

> **Default:** `50`

Output Arguments

rlgc_params

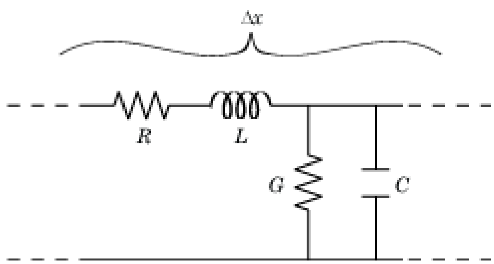
The output `rlgc_params` is structure whose fields are N-by-N-by-M arrays of transmission line parameters. Each of the M N-by-N matrices correspond to a frequency in the input vector `freq`.

- `rlgc_params.R` is an array of distributed resistances, in units of Ω/m. The matrices are real symmetric, the diagonal terms are nonnegative, and the off-diagonal terms are nonnegative.
- `rlgc_params.L` is an array of distributed inductances, in units of H/m. The matrices are real symmetric, the diagonal terms are positive, and the off-diagonal terms are nonnegative.
- `rlgc_params.G` is an array of distributed conductances, in units of S/m. The matrices are real symmetric, the diagonal terms are nonnegative, and the off-diagonal terms are nonpositive.
- `rlgc_params.C` is an array of distributed capacitances, in units of F/m. The matrices are real symmetric, the diagonal terms are positive, and the off-diagonal terms are nonpositive.
- `rlgc_params.Zc` is an array of complex characteristic line impedances, in ohms.
- `rlgc_params.alpha` is an array of real attenuation coefficients, in units of Np/m.

- `rlgc_params.beta` is an array of real phase constants, in units of rad/m.

Definitions

The following figure illustrates the RLGC transmission line model.



The representation consists of:

- The distributed resistance, R, of the conductors, represented by a series resistor.
- The distributed inductance, L, of the conductors, represented by a series inductor.
- The distributed conductance, G, between the two conductors, represented by a shunt resistor.
- The distributed capacitance, C, between the two conductors, represented by a shunt capacitor.

RLGC component units are all per unit length Δx.

Examples

Convert S-parameters to RLGC transmission line parameters:

```
s_11 = 0.000249791883190134 - 9.42320545953709e-005i;
s_12 = 0.999250283783862 -  0.000219770154524734i;
s_21 = 0.999250283783863 -  0.000219770154524756i;
s_22 = 0.000249791883190079 - 9.42320545953931e-005i;
s_params = [s_11,s_12; s_21,s_22];
length = 1e-3;
freq = 1e9;
z0 = 50;
```

```
rlgc_params = s2rlgc(s_params,length,freq,z0)
```

References

Degerstrom, M.J., B.K. Gilbert, and E.S. Daniel. “Accurate resistance, inductance, capacitance, and conductance (RLCG) from uniform transmission line measurements.” *Electrical Performance of Electronic Packaging,*. IEEE-EPEP, 18th Conference, 27–29 October 2008, pp. 77–80.

Sampath, M.K. “On addressing the practical issues in the extraction of RLGC parameters for lossy multiconductor transmission lines using S-parameter models.” *Electrical Performance of Electronic Packaging,*. IEEE-EPEP, 18th Conference, 27–29 October 2008, pp. 259–262.

See Also

`rlgc2s`

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 a default reference impedance of 50 ohms. Both `s_params` and `s_params_new` are complex N-by-N-by-M arrays, representing M N-port 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 reference impedance `z0_new`.

Alternatives

The `newref` function changes the reference impedance of S-parameters objects.

Examples

Convert S-parameters from one reference impedance to another reference impedance:

```
%Define a matrix of S-parameters
s_11 = 0.61*exp(j*165/180*pi);
s_21 = 3.72*exp(j*59/180*pi);
s_12 = 0.05*exp(j*42/180*pi);
s_22 = 0.45*exp(j*(-48/180)*pi);
s_params = [s_11 s_12; s_21 s_22];
z0 = 50;
z0_new = 40;
%Convert to new reference impedance
s_params_new = s2s(s_params,z0,z0_new)
```

See Also

`abcd2s` | `h2s` | `s2abcd` | `s2h` | `s2y` | `s2z` | `y2s` | `z2s`

Purpose

Convert single-ended S-parameters to common-mode S-parameters (S_{cc})

Syntax

```
scc_params = s2scc(s_params)
scc_params = s2scc(s_params,option)
```

Description

`scc_params = s2scc(s_params)` converts the 2*N*-port, single-ended S-parameters, `s_params`, to *N*-port, common-mode S-parameters, `scc_params`. `scc_params` is a complex *N*-by-*N*-by-*M* array that represents *M* *N*-port, common-mode S-parameters (S_{cc}).

`scc_params = s2scc(s_params,option)` converts S-parameters based on the optional `option` argument, which indicates the port-ordering convention of the S-parameters.

Input Arguments

s_params

`s_params` is a complex 4-by-4-by-*M* array that represents *M* 4-port S-parameters.

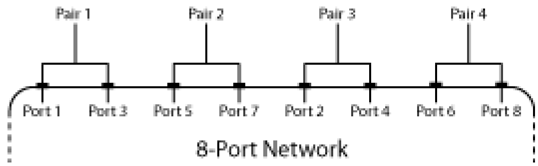
option

`1` (default) | `2` | `3`

`option` determines how the function orders the ports:

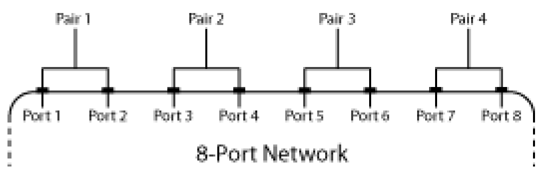
- `1` — `s2scc` pairs the odd-numbered ports together first, followed by the even-numbered ports. For example, in a single-ended, 8-port network:
  - Ports 1 and 3 become common-mode pair 1.
  - Ports 5 and 7 become common-mode pair 2.
  - Ports 2 and 4 become common-mode pair 3.
  - Ports 6 and 8 become common-mode pair 4.

  The following figure illustrates this convention for an 8-port device.



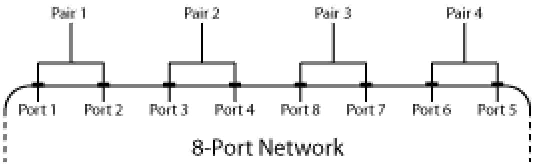
- `2` — `s2scc` pairs the input and output ports in ascending order. For example, in a single-ended, 8-port network:
  - Ports 1 and 2 become common-mode pair 1.
  - Ports 3 and 4 become common-mode pair 2.
  - Ports 5 and 6 become common-mode pair 3.
  - Ports 7 and 8 become common-mode pair 4.

  The following figure illustrates this convention for an 8-port device.



- `3` — `s2scc` pairs the input ports in ascending order and the output ports in descending order. For example, in a single-ended, 8-port network:
  - Ports 1 and 2 become common-mode pair 1.
  - Ports 3 and 4 become common-mode pair 2.
  - Ports 8 and 7 become common-mode pair 3.
  - Ports 6 and 5 become common-mode pair 4.

  The following figure illustrates this convention for an 8-port device.



Examples

Convert network data to common-mode S-parameters using the default port ordering:

```
ckt = read(rfckt.passive,'default.s4p');
s4p = ckt.NetworkData.Data;
s_cc = s2scc(s4p);
```

References

Fan, W., A. C. W. Lu, L. L. Wai, and B. K. Lok. "Mixed-Mode S-Parameter Characterization of Differential Structures." Electronic Packaging Technology Conference, pp. 533–537, 2003.

See Also

`s2scd` | `s2sdc` | `s2sdd` | `s2smm` | `smm2s`

Purpose

Convert 4-port, single-ended S-parameters to 2-port, cross-mode S-parameters (S_{cd})

Syntax

```
scd_params = s2scd(s_params)
scd_params = s2scd(s_params,option)
```

Description

`scd_params = s2scd(s_params)` converts the 2*N*-port, single-ended S-parameters, `s_params`, to *N*-port, cross-mode S-parameters, `scd_params`. `scd_params` is a complex *N*-by-*N*-by-*M* array that represents *M* *N*-port, cross-mode S-parameters (S_{cd}).

`scd_params = s2scd(s_params,option)` converts S-parameters based on the optional `option` argument, which indicates the port-ordering convention of the S-parameters.

Input Arguments

s_params

`s_params` is a complex 2*N*-by-2*N*-by-*M* array that represents *M* 2*N*-port S-parameters.

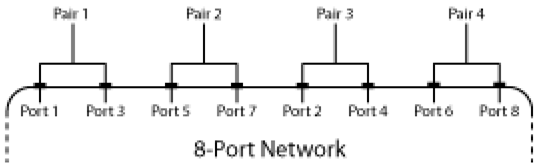
option

1 (default) | 2 | 3

`option` determines how the function orders the ports:

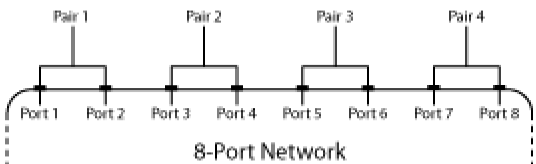
- `1` — `s2scd` pairs the odd-numbered ports together first, followed by the even-numbered ports. For example, in a single-ended, 8-port network:
  - Ports 1 and 3 become cross-mode pair 1.
  - Ports 5 and 7 become cross-mode pair 2.
  - Ports 2 and 4 become cross-mode pair 3.
  - Ports 6 and 8 become cross-mode pair 4.

  The following figure illustrates this convention for an 8-port device.



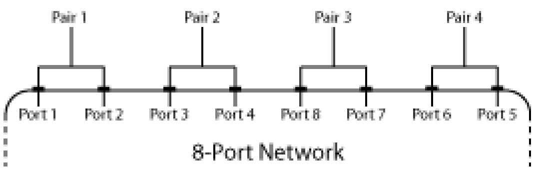
- `2` — `s2scd` pairs the input and output ports in ascending order. For example, in a single-ended, 8-port network:
  - Ports 1 and 2 become cross-mode pair 1.
  - Ports 3 and 4 become cross-mode pair 2.
  - Ports 5 and 6 become cross-mode pair 3.
  - Ports 7 and 8 become cross-mode pair 4.

  The following figure illustrates this convention for an 8-port device.



- `3` — `s2scd` pairs the input ports in ascending order and the output ports in descending order. For example, in a single-ended, 8-port network:
  - Ports 1 and 2 become cross-mode pair 1.
  - Ports 3 and 4 become cross-mode pair 2.
  - Ports 8 and 7 become cross-mode pair 3.
  - Ports 6 and 5 become cross-mode pair 4.

  The following figure illustrates this convention for an 8-port device.



Examples

Convert network data to cross-mode S-parameters using the default port ordering:

```
ckt = read(rfckt.passive,'default.s4p');
s4p = ckt.NetworkData.Data;
s_cd = s2scd(s4p);
```

References

Fan, W., A. C. W. Lu, L. L. Wai, and B. K. Lok. "Mixed-Mode S-Parameter Characterization of Differential Structures." Electronic Packaging Technology Conference, pp. 533–537, 2003.

See Also

s2scc | s2sdc | s2sdd | s2smm | smm2s

Purpose

Convert 4-port, single-ended S-parameters to 2-port, cross-mode S-parameters (S_{dc})

Syntax

```
sdc_params = s2sdc(s_params)
sdc_params = s2sdc(s_params,option)
```

Description

`sdc_params = s2sdc(s_params)` converts the 2*N*-port, single-ended S-parameters, `s_params`, to *N*-port, cross-mode S-parameters, `sdc_params`. `sdc_params` is a complex *N*-by-*N*-by-*M* array that represents *M* *N*-port, cross-mode S-parameters (S_{dc}).

`sdc_params = s2sdc(s_params,option)` converts S-parameters based on the optional `option` argument, which indicates the port-ordering convention of the S-parameters.

Input Arguments

s_params

`s_params` is a complex 2*N*-by-2*N*-by-*M* array that represents *M* 2*N*-port S-parameters.

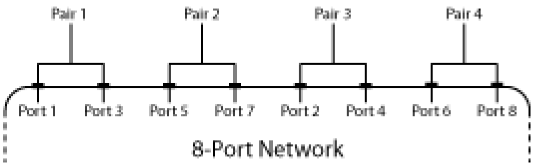
option

1 (default) | 2 | 3

`option` determines how the function orders the ports:

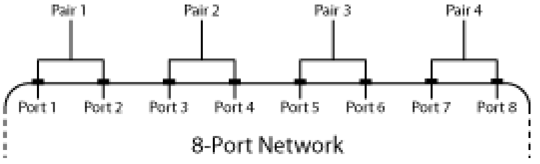
- `1` — `s2sdc` pairs the odd-numbered ports together first, followed by the even-numbered ports. For example, in a single-ended, 8-port network:
  - Ports 1 and 3 become cross-mode pair 1.
  - Ports 5 and 7 become cross-mode pair 2.
  - Ports 2 and 4 become cross-mode pair 3.
  - Ports 6 and 8 become cross-mode pair 4.

  The following figure illustrates this convention for an 8-port device.



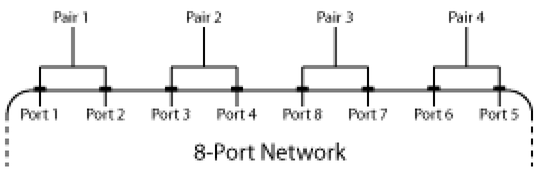
- `2` — `s2sdc` pairs the input and output ports in ascending order. For example, in a single-ended, 8-port network:
  - Ports 1 and 2 become cross-mode pair 1.
  - Ports 3 and 4 become cross-mode pair 2.
  - Ports 5 and 6 become cross-mode pair 3.
  - Ports 7 and 8 become cross-mode pair 4.

  The following figure illustrates this convention for an 8-port device.



- `3` — `s2sdc` pairs the input ports in ascending order and the output ports in descending order. For example, in a single-ended, 8-port network:
  - Ports 1 and 2 become cross-mode pair 1.
  - Ports 3 and 4 become cross-mode pair 2.
  - Ports 8 and 7 become cross-mode pair 3.
  - Ports 6 and 5 become cross-mode pair 4.

  The following figure illustrates this convention for an 8-port device.



Examples

Convert network data to cross-mode S-parameters using the default port ordering:

```
ckt = read(rfckt.passive,'default.s4p');
s4p = ckt.NetworkData.Data;
s_dc = s2sdc(s4p);
```

References

Fan, W., A. C. W. Lu, L. L. Wai, and B. K. Lok. "Mixed-Mode S-Parameter Characterization of Differential Structures." Electronic Packaging Technology Conference, pp. 533–537, 2003.

See Also

s2scc | s2scd | s2sdd | s2smm | smm2s

Purpose

Convert 4-port, single-ended S-parameters to 2-port, differential-mode S-parameters (S_{dd})

Syntax

```
sdd_params = s2sdd(s_params)
sdd_params = s2sdd(s_params,option)
```

Description

`sdd_params = s2sdd(s_params)` converts the 2*N*-port, single-ended S-parameters, `s_params`, to *N*-port, differential-mode S-parameters, `sdd_params`. `sdd_params` is a complex *N*-by-*N*-by-*M* array that represents *M* *N*-port, differential-mode S-parameters (S_{cd}).

`sdd_params = s2sdd(s_params,option)` converts S-parameters based on the optional `option` argument, which indicates the port-ordering convention of the S-parameters.

Input Arguments

s_params

`s_params` is a complex 2*N*-by-2*N*-by-*M* array that represents *M* 2*N*-port S-parameters.

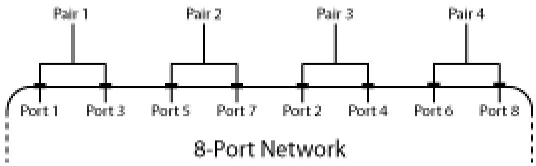
option

1 (default) | 2 | 3

`option` determines how the function orders the ports:

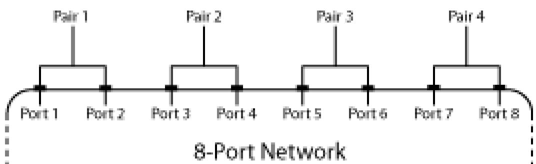
- `1` — `s2sdd` pairs the odd-numbered ports together first, followed by the even-numbered ports. For example, in a single-ended, 8-port network:
  - Ports 1 and 3 become differential-mode pair 1.
  - Ports 5 and 7 become differential-mode pair 2.
  - Ports 2 and 4 become differential-mode pair 3.
  - Ports 6 and 8 become differential-mode pair 4.

  The following figure illustrates this convention for an 8-port device.



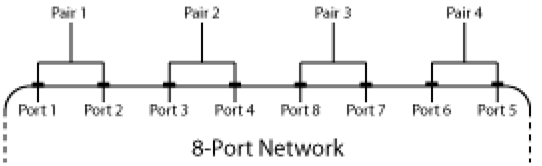
- `2` — `s2sdd` pairs the input and output ports in ascending order. For example, in a single-ended, 8-port network:
  - Ports 1 and 2 become differential-mode pair 1.
  - Ports 3 and 4 become differential-mode pair 2.
  - Ports 5 and 6 become differential-mode pair 3.
  - Ports 7 and 8 become differential-mode pair 4.

  The following figure illustrates this convention for an 8-port device.



- `3` — `s2sdd` pairs the input ports in ascending order and the output ports in descending order. For example, in a single-ended, 8-port network:
  - Ports 1 and 2 become differential-mode pair 1.
  - Ports 3 and 4 become differential-mode pair 2.
  - Ports 8 and 7 become differential-mode pair 3.
  - Ports 6 and 5 become differential-mode pair 4.

  The following figure illustrates this convention for an 8-port device.



Examples

Convert network data to differential-mode S-parameters using the default port ordering:

```
ckt = read(rfckt.passive,'default.s4p');
s4p = ckt.NetworkData.Data;
s_dd = s2sdd(s4p);
```

Read a data file, transform the data into differential-mode S-parameters, analyze the new data, and write the new data to a file:

```
% Create a circuit object from a data file
ckt = read(rfckt.passive,'default.s4p');
data = ckt.AnalyzedResult;
% Create a data object to store the
% differential S-parameters
diffSparams = rfdata.network;
diffSparams.Freq = data.Freq;
diffSparams.Data = s2sdd(data.S_Parameters);
diffSparams.Z0 = 2*data.Z0;
% Create a new circuit object with the data
% from the data object
diffCkt = rfckt.passive;
diffCkt.NetworkData = diffSparams;
% Analyze the new circuit object
frequencyRange = diffCkt.NetworkData.Freq;
ZL = 50;
ZS = 50;
Z0 = diffSparams.Z0;
analyze(diffCkt,frequencyRange,ZL,ZS,Z0);
diffData = diffCkt.AnalyzedResult;
% Write the differential S-parameters into a
```

```
% Touchstone data file
write(DiffCkt,'diffsparams.s2p');
```

References

Fan, W., A. C. W. Lu, L. L. Wai, and B. K. Lok. “Mixed-Mode S-Parameter Characterization of Differential Structures.” Electronic Packaging Technology Conference, pp. 533–537, 2003.

See Also

s2scc | s2scd | s2sdc | s2smm | smm2s

Purpose

Convert single-ended S-parameters to mixed-mode S-parameters

Syntax

```
[s_dd,s_dc,s_cd,s_cc] = s2smm(s_params_even)
s_mm = s2smm(s_params_odd,option)
```

Description

`[s_dd,s_dc,s_cd,s_cc] = s2smm(s_params_even)` converts the $2N$-port, single-ended S-parameters `s_params_even` into N-port, mixed-mode S-parameters. `s2smm` forms the mixed-mode ports by grouping the single-ended ports in pairs by port number (index), grouping odd-numbered ports first, followed by even-numbered ports.

`s_mm = s2smm(s_params_odd,option)` converts the S-parameter data according the port-numbering convention specified by `option`. You can also reorder the ports in `s_params` using the `snp2smp` function.

Input Arguments

s_params_even

`s_params_even` is a complex $2N$-by-$2N$-by-K array representing K single-ended, $2N$-port S-parameters. These parameters describe a device with an even number of ports.

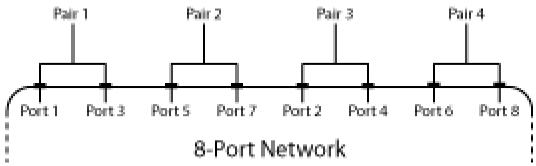
option

1 (default) | 2 | 3

`option` determines how the function orders the ports:

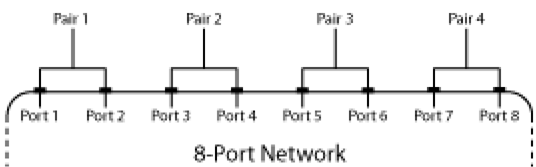
- 1 — `s2smm` pairs the odd-numbered ports together first, followed by the even-numbered ports. For example, in a single-ended, 8-port network:
  - Ports 1 and 3 become mixed-mode pair 1.
  - Ports 5 and 7 become mixed-mode pair 2.
  - Ports 2 and 4 become mixed-mode pair 3.
  - Ports 6 and 8 become mixed-mode pair 4.

  The following figure illustrates this convention for an 8-port device.



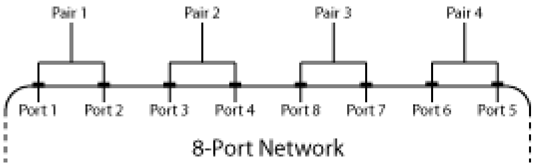
- `2` — `s2smm` pairs the input and output ports in ascending order. For example, in a single-ended, 8-port network:
  - Ports 1 and 2 become mixed-mode pair 1.
  - Ports 3 and 4 become mixed-mode pair 2.
  - Ports 5 and 6 become mixed-mode pair 3.
  - Ports 7 and 8 become mixed-mode pair 4.

  The following figure illustrates this convention for an 8-port device.



- `3` — `s2smm` pairs the input ports in ascending order and the output ports in descending order. For example, in a single-ended, 8-port network:
  - Ports 1 and 2 become mixed-mode pair 1.
  - Ports 3 and 4 become mixed-mode pair 2.
  - Ports 8 and 7 become mixed-mode pair 3.
  - Ports 6 and 5 become mixed-mode pair 4.

  The following figure illustrates this convention for an 8-port device.



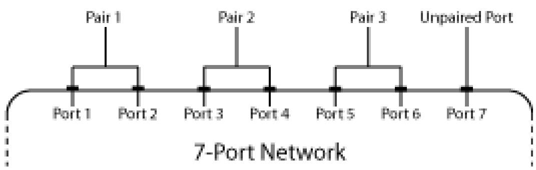
s_params_odd

`s_params_odd` is a complex $(2N + 1)$-by-$(2N + 1)$-by-K array representing K single-ended, (2N-port S-parameters. These parameters describe a device with an odd number of ports.

The port-ordering argument `option` is not available for $(2N + 1)$-by-$(2N + 1)$-by-K input arrays. In this case, the ports are always paired in ascending order, and the last port remains single-ended. For example, in a 7-port network:

- Ports 1 and 2 become mixed-mode pair 1.
- Ports 3 and 4 become mixed-mode pair 2.
- Ports 5 and 6 become mixed-mode pair 3.
- Ports 7 remains single ended.

The following figure illustrates this convention for a 7-port device.



Output Arguments

s_dd

`s_dd` is a complex N-by-N-by-K array containing K matrices of differential-mode, 2N-port S-parameters (S_{dd}).

s_dc

`s_dc` is a complex N-by-N-by-K array containing K matrices of cross-mode, N-port S-parameters (S_{dc}).

s_cd

`s_cd` is a complex N-by-N-by-K array containing K matrices of cross-mode, N-port S-parameters (S_{cd}).

s_cc

`s_cc` is a complex N-by-N-by-K array containing K matrices of common-mode, N-port S-parameters (S_{cc}).

s_mm

`s_mm` is a complex $(2N + 1)$-by-$(2N + 1)$-by-K array containing K matrices of mixed-mode S-parameters. The parameters are organized in the matrix as follows:

$$\begin{bmatrix} S_{dd,11} & \cdots & S_{dd,1N} & S_{dc,11} & \cdots & S_{dc,1N} & S_{ds,1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ S_{dd,N1} & \cdots & S_{dd,NN} & S_{dc,N1} & \cdots & S_{dc,NN} & S_{ds,N} \\ S_{cd,11} & \cdots & S_{cd,1N} & S_{cc,11} & \cdots & S_{cc,1N} & S_{cs,1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ S_{cd,N1} & \cdots & S_{cd,NN} & S_{cc,N1} & \cdots & S_{cc,NN} & S_{cs,N} \\ S_{sd,1} & \cdots & S_{sd,N} & S_{sc,1} & \cdots & S_{sc,N} & S_{ss} \end{bmatrix}$$

Examples

Convert 4-port S-parameters to 2-port, mixed-mode S-parameters:

```
ckt = read(rfckt.passive,'default.s4p');
s4p = ckt.NetworkData.Data;
[s_dd,s_dc,s_cd,s_cc] = s2smm(s4p);
```

References

Granberg, T., *Handbook of Digital Techniques for High-Speed Design.*Upper Saddle River, NJ: Prentice Hall, 2004.

See Also

`s2scc` | `s2scd` | `s2sdc` | `s2sdd` | `smm2s` | `snp2smp`

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 2-port S-parameters. `t_params` is a complex 2-by-2-by-M array, representing M 2-port T-parameters.

This function uses the following definition for T-parameters:

$$\begin{bmatrix} a_1 \\ b_1 \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \begin{bmatrix} b_2 \\ a_2 \end{bmatrix},$$

where:

- a_1 is the incident wave at the first port.
- b_1 is the reflected wave at the first port.
- a_2 is the incident wave at the second port.
- b_2 is the reflected wave at the second port.

Examples

Convert S-parameters to T-parameters:

```
%Define a matrix of S-parameters
s11 = 0.61*exp(j*165/180*pi);
s21 = 3.72*exp(j*59/180*pi);
s12 = 0.05*exp(j*42/180*pi);
s22 = 0.45*exp(j*(-48/180)*pi);
s_params = [s11 s12; s21 s22];
%Convert to T-parameters
t_params = s2t(s_params)
```

References

Gonzalez, Guillermo, *Microwave Transistor Amplifiers: Analysis and Design*, 2nd edition. Prentice-Hall, 1997, p. 25.

See Also

`s2abcd` | `s2h` | `s2y` | `s2z` | `t2s`

Purpose

Convert S-parameters of 2-port network to voltage or power-wave transfer function

Syntax

```
tf = s2tf(s_params)
tf = s2tf(s_params,z0,zs,zl)
tf = s2tf(s_params,z0,zs,zl,option)

tf = s2tf(hs)
tf = s2tf(hs,zs,zl)
tf = s2tf(hs,zs,zl,option)
```

Description

`tf = s2tf(s_params)` converts the scattering parameters, `s_params`, of a 2-port network into the voltage transfer function of the network.

`tf = s2tf(s_params,z0,zs,zl)` calculates the voltage transfer function using the reference impedance `z0`, source impedance `zs`, and load impedance `zl`.

`tf = s2tf(s_params,z0,zs,zl,option)` calculates the voltage or power-wave transfer function using the method specified by `option`.

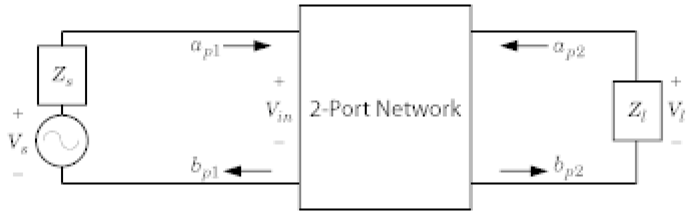
`tf = s2tf(hs)` converts the 2-port S-parameter object, `hs`, into the voltage transfer function of the network.

`tf = s2tf(hs,zs,zl)` calculates the voltage transfer function using the source impedance `zs`, and load impedance `zl`.

`tf = s2tf(hs,zs,zl,option)` calculates the voltage or power-wave transfer function using the method specified by `option`.

Algorithms

The following figure shows the setup for computing the transfer function, along with the impedences, voltages, and the power waves used to determine the gain.



The function uses the following voltages and power waves for calculations:

- V_l is the output voltage across the load impedance.
- V_s is the source voltage.
- V_{in} is the input voltage of the 2-port network.

- a_{p1} is the incident power wave, equal to $\frac{V_s}{2\sqrt{\mathrm{Re}(Z_s)}}$.
- b_{p2} is the transmitted power wave, equal to $\frac{\sqrt{\mathrm{Re}(Z_l)}}{Z_l} V_l$.

Input Arguments

hs

2-port S-parameters, specified as an RF Toolbox S-parameter object.

s_params

`s_params` is a complex 2-by-2-by-M array that represents M 2-port S-parameters.

z0

`z0` is the reference impedance, in ohms, of the S-parameters.

Default: `50`

zs

`zs` is the source impedance, in ohms, of the S-parameters.

Default: `50`

zl

`zl` is the load impedance, in ohms, of the S-parameters.

Default: `50`

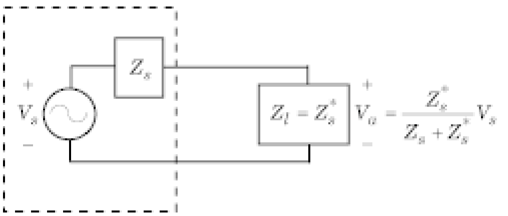
option

The optional `option` argument is an integer equal to `1`, `2`, or `3`. The value of `option` specifies the transfer function type:

- `1` — The transfer function is the gain from the incident voltage, V_a, to the output voltage for arbitrary source and load impedances:

  $$tf = \frac{V_l}{V_a}$$

  The following figure shows how to compute V_a from the source voltage V_s:

  IFor the S-parameters and impedance values, the transfer function is:

  $$tf = \frac{(Z_s + Z_s^*)}{Z_s^*} \frac{S_{21}(1+\Gamma_l)(1-\Gamma_s)}{2(1-S_{22}\Gamma_l)(1-\Gamma_{in}\Gamma_s)}$$

  where:

$$\Gamma_l = \frac{Z_l - Z_o}{Z_l + Z_o}$$

$$\Gamma_s = \frac{Z_s - Z_o}{Z_s + Z_o}$$

$$\Gamma_{in} = S_{11} + \left(S_{12}S_{21} \frac{\Gamma_l}{(1 - S_{22}\Gamma_l)} \right)$$

The following equation shows how the preceding transfer function is related to the transducer gain computed by the `powergain` function:

$$G_T = |tf|^2 \frac{\mathrm{Re}(Z_l)}{|Z_l|^2} \frac{|Z_s|^2}{\mathrm{Re}(Z_s)}$$

Notice that if Z_l and Z_S are real, $G_T = |tf|^2 \frac{Z_s}{Z_l}$.

- `2` — The transfer function is the gain from the source voltage to the output voltage for arbitrary source and load impedances:

$$tf = \frac{V_l}{V_s} = \frac{S_{21}(1 + \Gamma_l)(1 - \Gamma_s)}{2(1 - S_{22}\Gamma_l)(1 - \Gamma_{in}\Gamma_s)}$$

You can use this option to compute the transfer function $\frac{V_L}{V_{in}}$ by setting `zs` to `0`. This setting means that $\Gamma_s = -1$ and $V_{in} = V_s$.

- `3` — The transfer function is the power-wave gain from the incident power wave at the first port to the transmitted power wave at the second port:

$$tf = \frac{b_{p2}}{a_{p1}} = \frac{\sqrt{\mathrm{Re}(Z_l)\,\mathrm{Re}(Z_s)}}{Z_l} \frac{S_{21}(1 + \Gamma_l)(1 - \Gamma_s)}{(1 - S_{22}\Gamma_l)(1 - \Gamma_{in}\Gamma_s)}$$

Default: `1`

Output Arguments

tf

Examples

Calculate the voltage transfer function of an S-parameter array:

```
ckt = read(rfckt.passive,'passive.s2p');
sparams = ckt.NetworkData.Data;
tf = s2tf(sparams)
```

See Also

`powergain` | `rationalfit` | `s2scc` | `s2scd` | `s2sdc` | `s2sdd` | `snp2smp`

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.

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-12.

Examples

Convert S-parameters to Y-parameters:

```
%Define a matrix of S-parameters
s_11 = 0.61*exp(j*165/180*pi);
s_21 = 3.72*exp(j*59/180*pi);
s_12 = 0.05*exp(j*42/180*pi);
s_22 = 0.45*exp(j*(-48/180)*pi);
s_params = [s_11 s_12; s_21 s_22];
z0 = 50;
%Convert to Y-parameters
y_params = s2y(s_params,z0)
```

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.

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-12.

Examples

Convert S-parameters to Z-parameters:

```
%Define a matrix of S-parameters
s_11 = 0.61*exp(j*165/180*pi);
s_21 = 3.72*exp(j*59/180*pi);
s_12 = 0.05*exp(j*42/180*pi);
s_22 = 0.45*exp(j*(-48/180)*pi);
s_params = [s_11 s_12; s_21 s_22];
z0 = 50;
%Convert to Z-parameters
z_params = s2z(s_params,z0)
```

See Also

`abcd2z | h2z | s2abcd | s2h | s2y | y2z | z2s`

Purpose

Plot complex vector on Smith chart

Syntax

```
[lineseries,hsm] = smithchart(y)
hsm = smithchart
```

Description

`[lineseries,hsm] = smithchart(y)` plots the complex vector `y` on a Smith chart. `hsm` is the handle of the Smith chart object. `lineseries` is a column vector of handles to `lineseries` objects, one handle per plotted line.

To plot network parameters from a circuit (`rfckt`) or data (`rfdata`) object on a Smith chart, use the `smith` function.

`hsm = smithchart` draws a blank Smith chart and returns the handle, `hsm`, of the Smith chart object.

Output Arguments

lineseries

You change the properties of the plotted lines by changing the `lineseries properties`.

hsm

`hsm = 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 table lists all properties you can specify for `smithchart` objects along with units, valid values, and descriptions of their use.

Property Name	Description	Units and 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 gray).
`LabelColor`	Color of the line labels.	`ColorSpec`. Default is `[0 0 0]` (black).

Property Name	Description	Units and Values
`LabelSize`	Size of the line labels.	`FontSize`. Default is `10`. See the `Annotation Textbox Properties` reference page for more information on specifying font size.
`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 gray).
`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`.

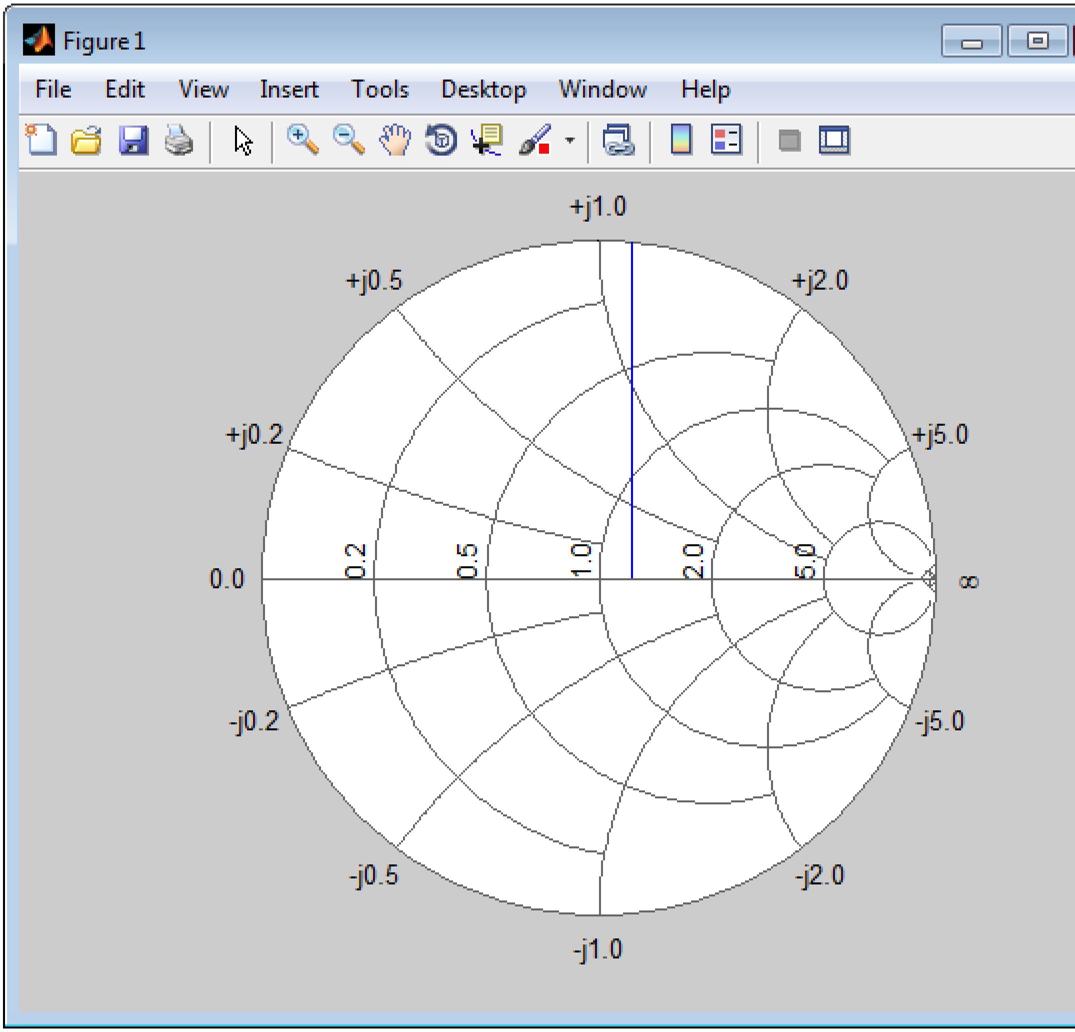
Property Name	Description	Units and Values
`Type`	Type of Smith chart.	`'z'` (default), `'y'`, or `'zy'`
`Value`	Two-row matrix. Row 1 specifies the values of the constant resistance and reactance lines in the chart. For the constant resistance and reactance lines, each element in Row 2 specifies the value at which the corresponding line in Row 1 ends.	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]`

Examples

Plot data on a Smith Chart

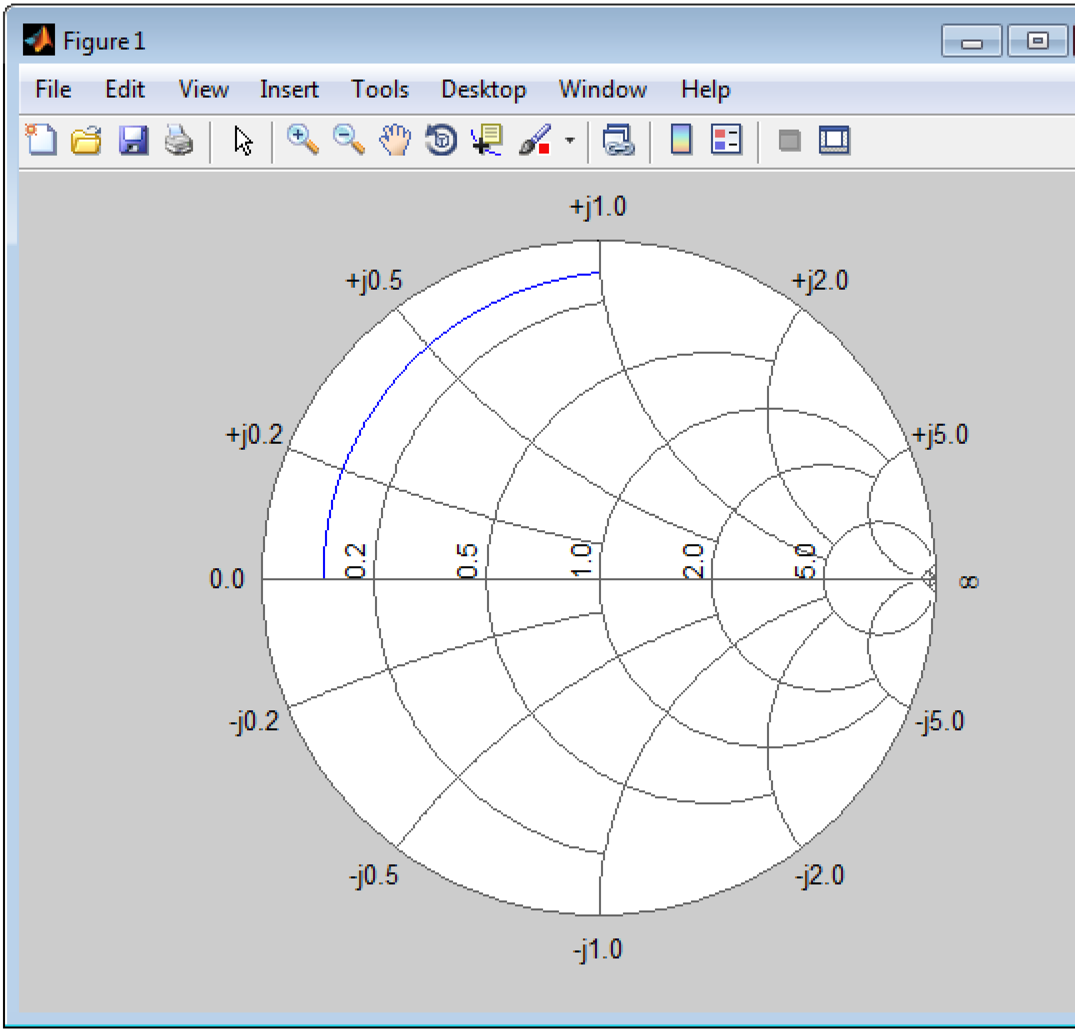
Define a vector of complex data.

```
smithdata = 0.1+j*[1:1000]*0.001;
smithchart(smithdata);
```



Using this vector as input to `smithchart` does not produce the expected result because the Smith Chart is drawn on the normal Cartesian coordinate system. However, you can use the `z2gamma` function to perform a bilinear transform on the vector first, converting the data to the Smith Chart coordinate system.

```
smithdata_trans = z2gamma(smithdata,1);
smithchart(smithdata_trans);
```



See Also

`get` | `set` | `smith`

Purpose

Convert mixed-mode 2N-port S-parameters to single-ended 4N-port S-parameters

Syntax

```
s_params = smm2s(s_dd,s_dc,s_cd,s_cc)
s_params = smm2s(s_dd,s_dc,s_cd,s_cc,option)
```

Description

`s_params = smm2s(s_dd,s_dc,s_cd,s_cc)` converts mixed-mode, N-port S-parameters into single-ended, $2N$-port S-parameters, `s_params`. `smm2s` maps the first half of the mixed-mode ports to the odd-numbered pairs of single-ended ports and maps the second half to the even-numbered pairs.

`s_params = smm2s(s_dd,s_dc,s_cd,s_cc,option)` converts the S-parameter data using the optional argument `option`. You can also reorder the ports in `s_params` using the `snp2smp` function.

Input Arguments

s_cc

`s_cc` is a complex N-by-N-by-K array containing K matrices of common-mode, N-port S-parameters (S_{cc}).

s_cd

`s_cd` is a complex N-by-N-by-K array containing K matrices of cross-mode, N-port S-parameters (S_{cd}).

s_dc

`s_dc` is a complex N-by-N-by-K array containing K matrices of cross-mode, N-port S-parameters (S_{dc}).

s_dd

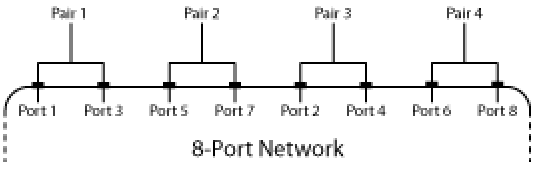
`s_dd` is a complex N-by-N-by-K array containing K matrices of differential-mode, N-port S-parameters (S_{dd}).

option

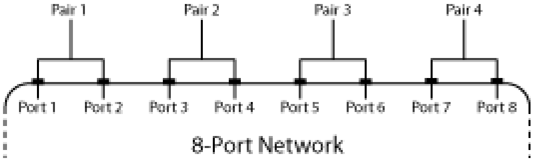
1 (default) | 2 | 3

`option` determines how the function orders the ports:

- `1` — `smm2s` maps the first half of the mixed-mode pairs to odd-numbered pairs of single-ended ports and maps the second half to even-numbered pairs. For example, in a mixed-mode, 4-port network:
  - Port 1 becomes single-ended ports 1 and 3.
  - Port 2 becomes single-ended ports 5 and 7.
  - Port 3 becomes single-ended ports 2 and 4.
  - Port 4 becomes single-ended ports 6 and 8.

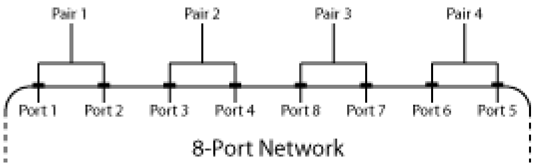


- `2` — `smm2s` maps the first half of the mixed-mode pairs to single-ended ports in ascending numerical order, followed by the second half, also in ascending order. For example, in a mixed-mode, 4-port network:
  - Port 1 becomes single-ended ports 1 and 2.
  - Port 2 becomes single-ended ports 3 and 4.
  - Port 3 becomes single-ended ports 5 and 6.
  - Port 4 becomes single-ended ports 7 and 8.



- `3` — `smm2s` maps the first half of the mixed-mode pairs to single-ended ports in ascending numerical order. The function maps the second half to pairs of ports in descending order. For example, in a mixed-mode, 4-port network:

- Port 1 becomes single-ended ports 1 and 2.
- Port 2 becomes single-ended ports 3 and 4.
- Port 3 becomes single-ended ports 8 and 7.
- Port 4 becomes single-ended ports 6 and 5.



Output Arguments

s_params

`s_params` is a complex $2N$-by-$2N$-by-K array representing K single-ended, $2N$-port S-parameters.

Examples

Convert between mixed-mode and single-ended S-parameters:

```
% Create mixed-mode S-parameters:
  ckt = read(rfckt.passive,'default.s4p');
  s4p = ckt.NetworkData.Data;
  [sdd,scd,sdc,scc] = s2smm(s4p);
% Convert them back to 4-port,single-ended S-parameters:
  s4p_converted_back = smm2s(sdd,scd,sdc,scc);
```

References

Granberg, T., *Handbook of Digital Techniques for High-Speed Design.*Upper Saddle River, NJ: Prentice Hall, 2004.

See Also

`s2scc` | `s2scd` | `s2sdc` | `s2sdd` | `s2smm` | `snp2smp`

snp2smp

Purpose

Convert single-ended N-port S-parameters to single-ended M-port S-parameters

Syntax

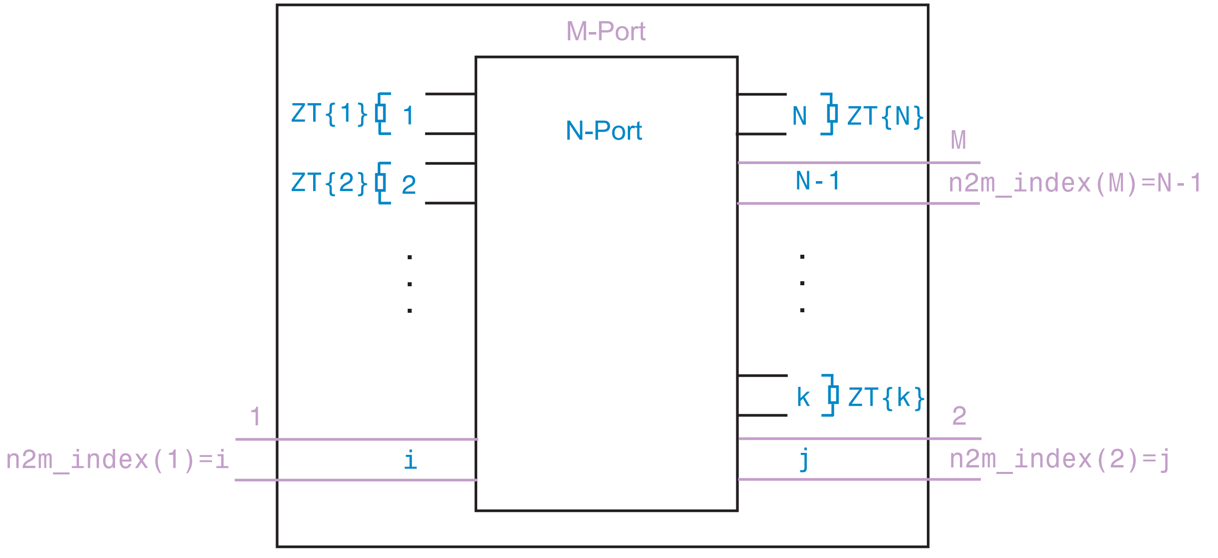
```
s_params_mp = snp2smp(s_params_np)
s_params_mp = snp2smp(s_params_np,Z0,n2m_index,ZT)
```

Description

`s_params_mp = snp2smp(s_params_np)` converts the single-ended N-port S-parameters, `s_params_np`, into the single-ended M-port S-parameters, `s_params_mp`. M must be less than or equal to N.

`s_params_mp = snp2smp(s_params_np,Z0,n2m_index,ZT)` converts the S-parameter data using the optional arguments `Z0`, `n2m_index`, and `ZT` that control the conversion.

The following figure illustrates how to use the optional input arguments to specify the ports for the output data and the termination of the remaining ports.



Input Arguments

s_params_np

s_params_np is a complex N-by-N-by-K array representing K N-port S-parameters.

Z0

Z0 is the reference impedance, in ohms, of s_params_np and s_params_mp.

Default: 50

n2m_index

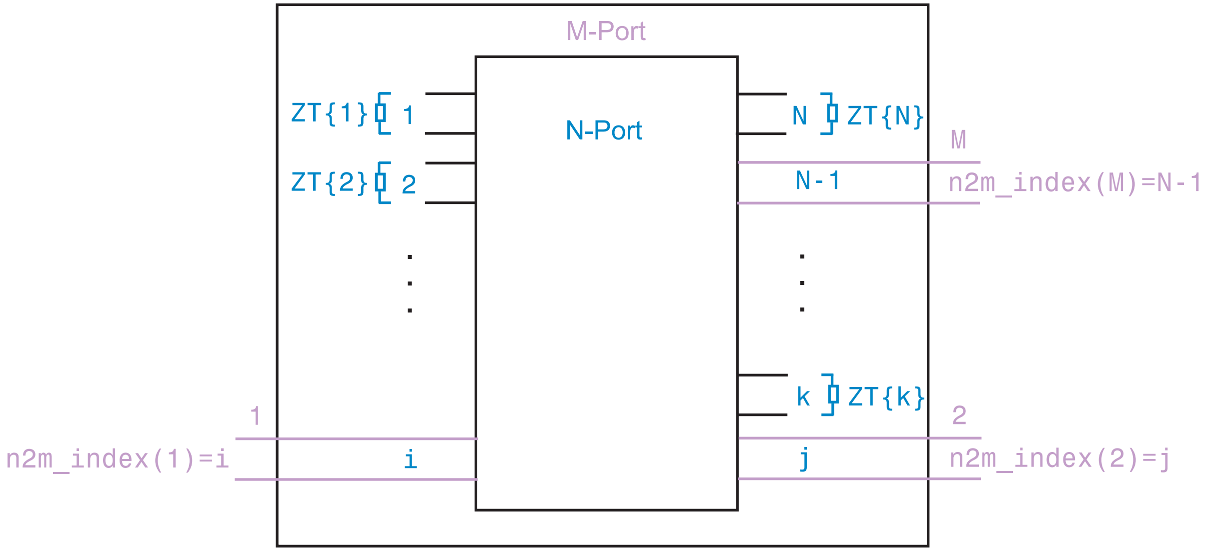
n2m_index is a vector of length M specifying how the ports of the N-port S-parameters map to the ports of the M-port S-parameters. n2m_index(i) is the index of the port from s_params_np that the function converts to the ith port of s_params_mp. For example, the setting [1, 2] means that M is 2, and the first two ports of the N-port S-parameters become the ports of the M-port parameters. The function terminates any additional ports with the impedances specified by ZT.

Default: [1, 2]

ZT

ZT is a scalar, vector, or cell array specifying the termination impedance of the ports. If M is less than N, snp2smp terminates the N–M ports not listed in n2m_index using the values in ZT. If ZT is a scalar, the function terminates all N–M ports not listed in n2m_index by the same impedance ZT. If ZT is a vector of length K, ZT[i] is the impedance that terminates all N–M ports of the ith frequency point not listed in n2m_index. If ZT is a cell array of length N, ZT{j} is the impedance that terminates the jth port of the N-port S-parameters. The function ignores impedances related to the ports listed in n2m_index. Each ZT{j} can be a scalar or a vector of length K.

The following figure illustrates how to use the optional input arguments to specify the ports for the output data and the termination of the remaining ports.



Default: `Z0`

Examples

Convert 3-port S-parameters to 3-port S-parameters with port indices swapped from `[1 2 3]` to `[2 3 1]`:

```
ckt = read(rfckt.passive,'default.s3p');
%default.s3p represents a real counterclockwise circulator
s3p = ckt.NetworkData.Data;
Z0 = ckt.NetworkData.Z0;
s3p_new = snp2smp(s3p,Z0,[2 3 1])
```

Convert 3-port S-parameters to 2-port S-parameters by terminating port 3 with an impedance of `Z0`:

```
ckt = read(rfckt.passive,'default.s3p');
```

```
s3p = ckt.NetworkData.Data;
Z0 = ckt.NetworkData.Z0;
s2p = snp2smp(s3p,Z0);
```

Convert 16-port S-parameters to 4-port S-parameters by using ports 1, 16, 2, and 15 as the first, second, third, and fourth ports; terminate the remaining 12 ports with an impedance of Z0:

```
ckt = read(rfckt.passive,'default.s16p');
s16p = ckt.NetworkData.Data;
Z0 = ckt.NetworkData.Z0;
s4p = snp2smp(s16p,Z0,[1 16 2 15],Z0)
```

Convert 16-port S-parameters to 4-port S-parameters by using ports 1, 16, 2, and 15 as the first, second, third, and fourth ports; terminate port 4 with an impedance of 100 ohms and terminate the remaining 11 ports with an impedance of 50 ohms:

```
ckt = read(rfckt.passive,'default.s16p');
s16p = ckt.NetworkData.Data;
Z0 = ckt.NetworkData.Z0;
ZT = {};
ZT(1:16) = {50};
ZT{4} = 100;
s4p = snp2smp(s16p,Z0,[1 16 2 15],ZT)
```

See Also

freqresp | rfmodel.rational | s2tf | timeresp | writeva

stabilityk

Purpose

Stability factor K of 2-port network

Syntax

```
[k,b1,b2,delta] = stabilityk(s_params)
[k,b1,b2,delta] = stabilityk(hs)
```

Description

`[k,b1,b2,delta] = stabilityk(s_params)` calculates and returns the stability factor, `k`, and the conditions `b1`, `b2`, and `delta` for the 2-port network. The input `s_params` is a complex 2-by-2-by-M array, representing M 2-port S-parameters.

`[k,b1,b2,delta] = stabilityk(hs)` calculates and returns the stability factor and stability conditions for the 2-port network represented by the S-parameter object `hs`.

Algorithms

Necessary and sufficient conditions for stability are `k>1` and `abs(delta)<1`. `stabilityk` calculates the outputs using the equations

$$K = \frac{1 - |S_{11}|^2 - |S_{22}|^2 + |\Delta|^2}{2|S_{12}S_{21}|}$$

$$B_1 = 1 + |S_{11}|^2 - |S_{22}|^2 - |\Delta|^2$$

$$B_2 = 1 - |S_{11}|^2 + |S_{22}|^2 - |\Delta|^2$$

where:

- S_{11}, S_{12}, S_{21}, and S_{22} are S-parameters from the input argument `s_params`.
- Δ is a vector whose members are the determinants of the M 2-port S-parameter matrices:

$$\Delta = S_{11}S_{22} - S_{12}S_{21}$$

The function performs these calculations element-wise for each of the M S-parameter matrices in `s_params`.

Examples

Examine the stability of network data from a file:

```
% Calculate stability factor and conditions
  ckt = read(rfckt.passive,'passive.s2p');
  s_params = ckt.NetworkData.Data;
  freq = ckt.NetworkData.Freq;
  [k b1 b2 delta] = stabilityk(s_params);
% Check stability criteria
  stability_index = (k>1)&(abs(delta)<1);
  is_stable = all(stability_index)
% List frequencies with unstable S-parameters
  freq_unstable = freq(~stability_index);
```

References

Gonzalez, Guillermo, *Microwave Transistor Amplifiers: Analysis and Design*, 2nd edition. Prentice-Hall, 1997, pp. 217–228.

See Also

`gammaml` | `gammams` | `stabilitymu`

stabilitymu

Purpose

Stability factor μ of 2-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 2-port network. The input `s_params` is a complex 2-by-2-by-M array, representing M 2-port S-parameters.

`[mu,muprime] = stabilitymu(hs)` calculates and returns the stability factors for the network represented by the S-parameter object `hs`.

The stability factor, μ, defines the minimum distance between the center of the unit Smith chart and the unstable region in the load plane. The function assumes that port 2 is the load.

The stability factor, μ', defines the minimum distance between the center of the unit Smith chart and the unstable region in the source plane. The function assumes that port 1 is the source.

Having $\mu > 1$ or $\mu' > 1$ is the necessary and sufficient condition for the 2-port linear network to be unconditionally stable, as described by the S-parameters.

Algorithms

`stabilitymu` calculates the stability factors using the equations

$$\mu = \frac{1-|S_{11}|^2}{\left|S_{22} - S_{11}^*\Delta\right| + |S_{21}S_{12}|}$$

$$\mu' = \frac{1-|S_{22}|^2}{\left|S_{11} - S_{22}^*\Delta\right| + |S_{21}S_{12}|}$$

where:

- S_{11}, S_{12}, S_{21}, and S_{22} are S-parameters, from the input argument `s_params`.
- Δ is a vector whose members are the determinants of the M 2-port S-parameter matrices:

$$\Delta = S_{11}S_{22} - S_{12}S_{21}$$

- S^* is the complex conjugate of the corresponding S-parameter.

The function performs these calculations element-wise for each of the M S-parameter matrices in `s_params`.

Examples

Examine the stability of network data from a file:

```
% Calculate stability factor and conditions
  ckt = read(rfckt.passive,'passive.s2p');
  s_params = ckt.NetworkData.Data;
  freq = ckt.NetworkData.Freq;
  [mu muprime] = stabilitymu(s_params);
% Check stability criteria
  stability_index = (mu>1)|(muprime>1);
  is_stable = all(stability_index)
% List frequencies with unstable S-parameters
  freq_unstable = freq(~stability_index);
```

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, pp. 2303-2311, December 1992.

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 2-port T-parameters. `s_params` is a complex 2-by-2-by-M array, representing M 2-port S-parameters.

This function defines the T-parameters as

$$\begin{bmatrix} a_1 \\ b_1 \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \begin{bmatrix} b_2 \\ a_2 \end{bmatrix},$$

where:

- a_1 is the incident wave at the first port.
- b_1 is the reflected wave at the first port.
- a_2 is the incident wave at the second port.
- b_2 is the reflected wave at the second port.

Examples

Convert T-parameters to S-parameters:

```
%Define a matrix of T-parameters
t11 =  0.138451095405929   - 0.230421317393041i;
t21 = -0.0451985986689165  + 0.157626245839348i;
t12 =  0.0353675449261375  + 0.115682026931012i;
t22 = -0.00194567217559662 - 0.0291212122613417i;
t_params = [t11 t12; t21 t22];
%Convert to S-parameters
s_params = t2s(s_params)
```

References

Gonzalez, Guillermo, *Microwave Transistor Amplifiers: Analysis and Design*, 2nd edition. Prentice-Hall, 1997, p. 25.

See Also

abcd2s | h2s | s2t | y2s | z2s

Purpose

VSWR at given reflection coefficient Γ

Syntax

```
ratio = vswr(gamma)
```

Description

`ratio = vswr(gamma)` calculates the voltage standing-wave ratio $VSWR$ at the given reflection coefficient Γ as

$$VSWR = \frac{1 + |\Gamma|}{1 - |\Gamma|}$$

The input `gamma` is a complex vector. The output `ratio` is a real vector of the same length as `gamma`.

Examples

Calculate the VSWR for a given reflection coefficient:

```
gamma = 1/3;
ratio = vswr(gamma)
```

See Also

`gamma2z` | `gammain` | `gammaout`

y2abcd

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 $2N$-by-$2N$-by-M array, representing M $2N$-port Y-parameters. `abcd_params` is a complex $2N$-by-$2N$-by-M array, representing M $2N$-port ABCD-parameters. The output ABCD-parameter matrices have distinct A, B, C, and D submatrices:

$$\begin{bmatrix} [A] & [B] \\ [C] & [D] \end{bmatrix}$$

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-12.

Examples

Convert Y-parameters to ABCD-parameters:

```
%Define a matrix of Y-parameters.
Y11 =  0.0488133074245012 - 0.390764155450191i;
Y12 = -0.0488588365420561 + 0.390719345880018i;
Y21 = -0.0487261119282660 + 0.390851884427087i;
Y22 =  0.0487710062903760 - 0.390800401433241i;
y_params = [Y11,Y12; Y21,Y22];
%Convert to ABCD-parameters
abcd_params = y2abcd(y_params);
```

See Also

`abcd2y` | `h2abcd` | `s2abcd` | `y2h` | `y2s` | `y2z` | `z2abcd`

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 2-port Y-parameters. `h_params` is a complex 2-by-2-by-M array, representing M 2-port hybrid h-parameters.

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-12.

Examples

Convert Y-parameters to h-parameters:

```
%Define a matrix of Y-parameters.
Y11 =  0.0488133074245012 - 0.390764155450191i;
Y12 = -0.0488588365420561 + 0.390719345880018i;
Y21 = -0.0487261119282660 + 0.390851884427087i;
Y22 =  0.0487710062903760 - 0.390800401433241i;
y_params = [Y11,Y12; Y21,Y22];
%Convert to h-parameters
h_params = y2h(y_params);
```

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. The default value of is `z0` 50 ohms. `s_params` is a complex N-by-N-by-*M* array, representing *M* N-port S-parameters.

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-12.

Examples

Convert Y-parameters to S-parameters:

```
%Define a matrix of Y-parameters.
Y11 =  0.0488133074245012 - 0.390764155450191i;
Y12 = -0.0488588365420561 + 0.390719345880018i;
Y21 = -0.0487261119282660 + 0.390851884427087i;
Y22 =  0.0487710062903760 - 0.390800401433241i;
y_params = [Y11,Y12; Y21,Y22];
%Convert to S-parameters
s_params = y2s(y_params);
```

See Also

`abcd2s | h2s | s2y | y2abcd | y2h | y2s | y2z | z2s`

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.

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-12.

Examples

Convert Y-parameters to Z-parameters:

```
%Define a matrix of Y-parameters.
Y11 =  0.0488133074245012 - 0.390764155450191i;
Y12 = -0.0488588365420561 + 0.390719345880018i;
Y21 = -0.0487261119282660 + 0.390851884427087i;
Y22 =  0.0487710062903760 - 0.390800401433241i;
y_params = [Y11,Y12; Y21,Y22];
%Convert to Z-parameters
z_params = y2z(y_params);
```

See Also

`abcd2z` | `h2z` | `y2abcd` | `y2h` | `y2s` | `y2z` | `z2s` | `z2y`

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 $2N$-by-$2N$-by-M array, representing M $2N$-port Z-parameters. `abcd_params` is a complex $2N$-by-$2N$-by-M array, representing M $2N$-port ABCD-parameters. The output ABCD-parameter matrices have distinct A, B, C, and D submatrices:

$$\begin{bmatrix} [A] & [B] \\ [C] & [D] \end{bmatrix}$$

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-12.

Examples

Convert Z-parameters to ABCD-parameters:

```
%Define a matrix of Z-parameters.
Z11 = -14567.2412789287 - 148373.315116592i
Z12 = -14588.1106171651 - 148388.583516562i
Z21 = -14528.0522132692 - 148350.705757767i
Z22 = -14548.5996561832 - 148363.457002006i
z_params = [Z11,Z12; Z21,Z22];
%Convert to ABCD-parameters
abcd_params = z2abcd(z_params);
```

See Also

`abcd2z` | `h2abcd` | `s2abcd` | `y2abcd` | `z2h` | `z2s` | `z2y`

z2gamma

Purpose

Convert impedance to reflection coefficient

Syntax

```
gamma = z2gamma(z)
gamma = z2gamma(z,z0)
```

Description

`gamma = z2gamma(z)` converts the impedance `z` to the reflection coefficient `gamma` using a reference impedance of 50 ohms.

`gamma = z2gamma(z,z0)` converts the impedance `z` to the reflection coefficient `gamma` using a reference impedance of `z0` ohms.

Algorithms

`z2gamma` calculates the coefficient using the equation

$$\Gamma = \frac{Z - Z_0}{Z + Z_0}$$

Examples

Convert an impedance of 100 ohms into a reflection coefficient, using a 50-ohm reference impedance:

```
z = 100;
gamma = z2gamma(z)
```

See Also

`gamma2z` | `gammain` | `gammaout`

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 2-port Z-parameters. `h_params` is a complex 2-by-2-by-M array, representing M 2-port hybrid h-parameters.

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-12.

Examples

Convert Z-parameters to H-parameters:

```
%Define a matrix of Z-parameters.
Z11 = -14567.2412789287 - 148373.315116592i
Z12 = -14588.1106171651 - 148388.583516562i
Z21 = -14528.0522132692 - 148350.705757767i
Z22 = -14548.5996561832 - 148363.457002006i
z_params = [Z11,Z12; Z21,Z22];
%Convert to h-parameters
h_params = z2h(z_params);
```

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.

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-12.

Examples

Convert Z-parameters to S-parameters:

```
%Define a matrix of Z-parameters.
Z11 = -14567.2412789287 - 148373.315116592i
Z12 = -14588.1106171651 - 148388.583516562i
Z21 = -14528.0522132692 - 148350.705757767i
Z22 = -14548.5996561832 - 148363.457002006i
z_params = [Z11,Z12; Z21,Z22];
%Convert to S-parameters
s_params = z2s(z_params);
```

See Also

`abcd2s` | `h2s` | `s2z` | `y2s` | `z2abcd` | `z2h` | `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.

Examples

Convert Z-parameters to Y-parameters:

```
%Define a matrix of Z-parameters.
Z11 = -14567.2412789287 - 148373.315116592i
Z12 = -14588.1106171651 - 148388.583516562i
Z21 = -14528.0522132692 - 148350.705757767i
Z22 = -14548.5996561832 - 148363.457002006i
z_params = [Z11,Z12; Z21,Z22];
%Convert to Y-parameters
y_params = z2y(z_params);
```

See Also

`abcd2y` | `h2y` | `s2y` | `y2z` | `z2abcd` | `z2h` | `z2s`

AMP File Format

AMP File Data Sections

In this section...

Overview

The AMP data file describes a single nonlinear device. Its format can contain the following types of data:

- S, Y, or Z network parameters
- Noise parameters
- Noise figure data
- Power data
- IP3 data

An AMP file must contain either power data 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.

Note If the file contains both network parameter data and power data, RF Toolbox software checks the data for consistency. If the amplifier gain computed from the network parameters is not consistent with the gain computed from the power data, a warning appears. For more information, see "Inconsistent Data Sections" on page A-13.

Two AMP files, `samplepa1.amp` and `default.amp`, ship with the toolbox to show the AMP format. They describe a nonlinear 2-port amplifier with noise. See "Model a Cascaded RF Network" on page 1-11 for an example that shows how to use an AMP file.

For information on specifying data in an AMP file, see "AMP File Data Sections" on page A-2. For information about adding comments to an AMP file, see "Denoting Comments" on page A-3.

Denoting 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.

Note In the data section descriptions, brackets (`[]`) indicate optional data or characters. All values are case insensitive.

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. Its value can be `S[PARAMETERS]`, `Y[PARAMETERS]`, or `Z[PARAMETERS]`. `Parameter` indicates the form of the data. Its value 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).
`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. Its value 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 1 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.

Allowable Power Data Units

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 the input power data. See Allowable Power Data Units on page A-9 for a complete list of valid values. 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 units of the frequency point must be specified explicitly using the abbreviations `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+009 Hz.

```
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. The data can appear in any order.
- The second column contains the corresponding output power magnitude.
- The third column contains the output phase shift in degrees.

Note RF Toolbox software does not use the phase data directly. SimRF blocks use this data in conjunction with RF Toolbox software to create the AM/PM conversion table for the Equivalent Baseband library General Amplifier and General Mixer blocks.

If all phases are zero, you can omit the third column. If all phases are zero or omitted, the toolbox assumes that the small signal phase from the network parameter section of the file (`180*angle(`S_{21}`(f))/pi`) is the phase for all power levels.

In contrast, if one or more phases in the power data section are nonzero, the toolbox interpolates and extrapolates the data to determine the phase at all power levels. The small signal phase (`180*angle(`S_{21}`(f))/pi`) from the network parameter section is ignored.

Inconsistency between the power data and network parameter sections of the file may cause incorrect results. To avoid this outcome, verify that the following criteria must is met:

- The lowest input power value for which power data exists falls in the small signal (linear) region.
- In the power table for each frequency point `f`, the power gain and phase at the lowest input power value are equal to `20*log10(abs(`S_{21}`(f)))` and `180*angle(`S_{21}`(f))/pi`, respectively, in the network parameter section.

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
```

Note The file can contain more than one section of power data, with each section corresponding to a different frequency value. When you analyze data from a file with multiple power data sections, power data is taken from the frequency point that is closest to the analysis frequency.

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. Valid values are `dBW`, `dBm`, `mW`, and `W`. The default is `W`. See Allowable Power Data Units on page A-9 for an explanation of the allowable `Units` values.

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. Valid values are `GHz`, `MHz`, and `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, then that same value is used for all frequencies. Enter the value in column 1 of the line following header line 2. You must include the second line of the header, but the application ignores it.

Inconsistent Data Sections

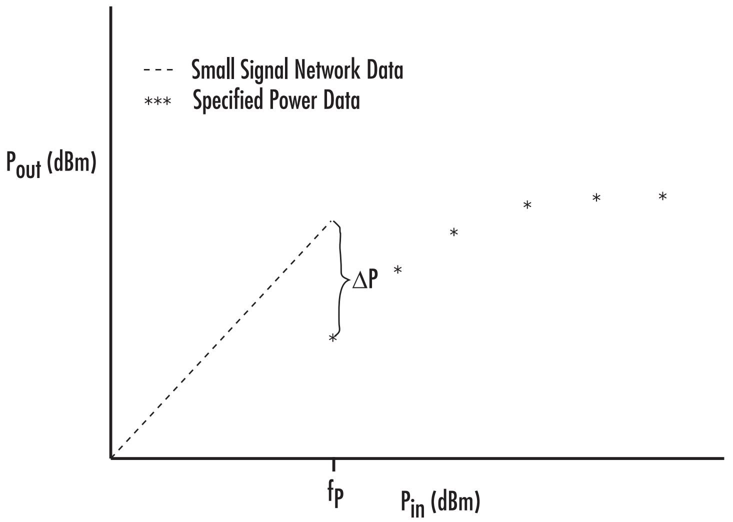
If an AMP file contains both network parameter data and power data, RF Toolbox software checks the data for consistency.

The toolbox compares the small-signal amplifier gain defined by the network parameters, S_{21}, and by the power data, $P_{out} - P_{in}$. The discrepancy between the two is computed in dBm using the following equation:

$$\Delta P = S_{21}(f_P) - P_{out}(f_P) + P_{in}(f_P)$$

wheref_P is the lowest frequency for which power data is specified.

The discrepancy is shown in the following graph.



If ΔP is more than 0.4 dB, a warning appears. Large discrepancies may indicate measurement errors that require resolution.

Index

A

B

C

P

R