RF Toolbox 2
User's Guide

MATLAB®

The MathWorks
Accelerating the pace of engineering and science

How to Contact The 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)

Contents

1 Getting Started

2 Selecting an RF Object

3 Modeling an RF Component

4 Exporting Verilog-A Models

5 RF Tool: An RF Analysis GUI

6 Objects — By Category

7 Objects — Alphabetical List

8 Methods — By Category

9 Methods — Alphabetical List

10 Functions — By Category

11 Functions — Alphabetical List

A AMP File Format

B Examples

Index

1

Getting Started

What Is RF Toolbox?

RF Toolbox extends MATLAB® with objects and functions for modeling RF circuits consisting of:

- Components such as RF filters, transmission lines, amplifiers, and mixers.
- Networks of interconnected components, such as cascaded, parallel, series, or hybrid networks.

Note To use RF Toolbox, you must have MATLAB installed.

You use objects from RF Toolbox to represent the components of your RF network. The toolbox provides several types of component representations using network parameters (S, Y, Z, ABCD, H, and T format) and physical properties.

You integrate the components to represent your RF network and analyze the network at specified frequencies.

You then perform one or more of the following tasks:

- Visualize network data using plots and Smith charts.
- Compute the transfer function and time-domain response of a circuit.
- Export a Verilog-A description of a component or network for use in a time-domain circuit simulator.

All RF Toolbox features are accessible and executable interactively from the MATLAB prompt or programmatically using M-code scripts.

RF Toolbox provides access to a subset of the command-line functionality through a graphical user interface, RF Tool. Using RF Tool, you can perform the following tasks:

- Design, analyze, and visualize RF components and networks.
- Export circuits to the MATLAB workspace, or to a file for use with RF Toolbox functions and other circuit objects.

A validated model of an RF circuit can provide an executable specification for verification in a system-level simulation.

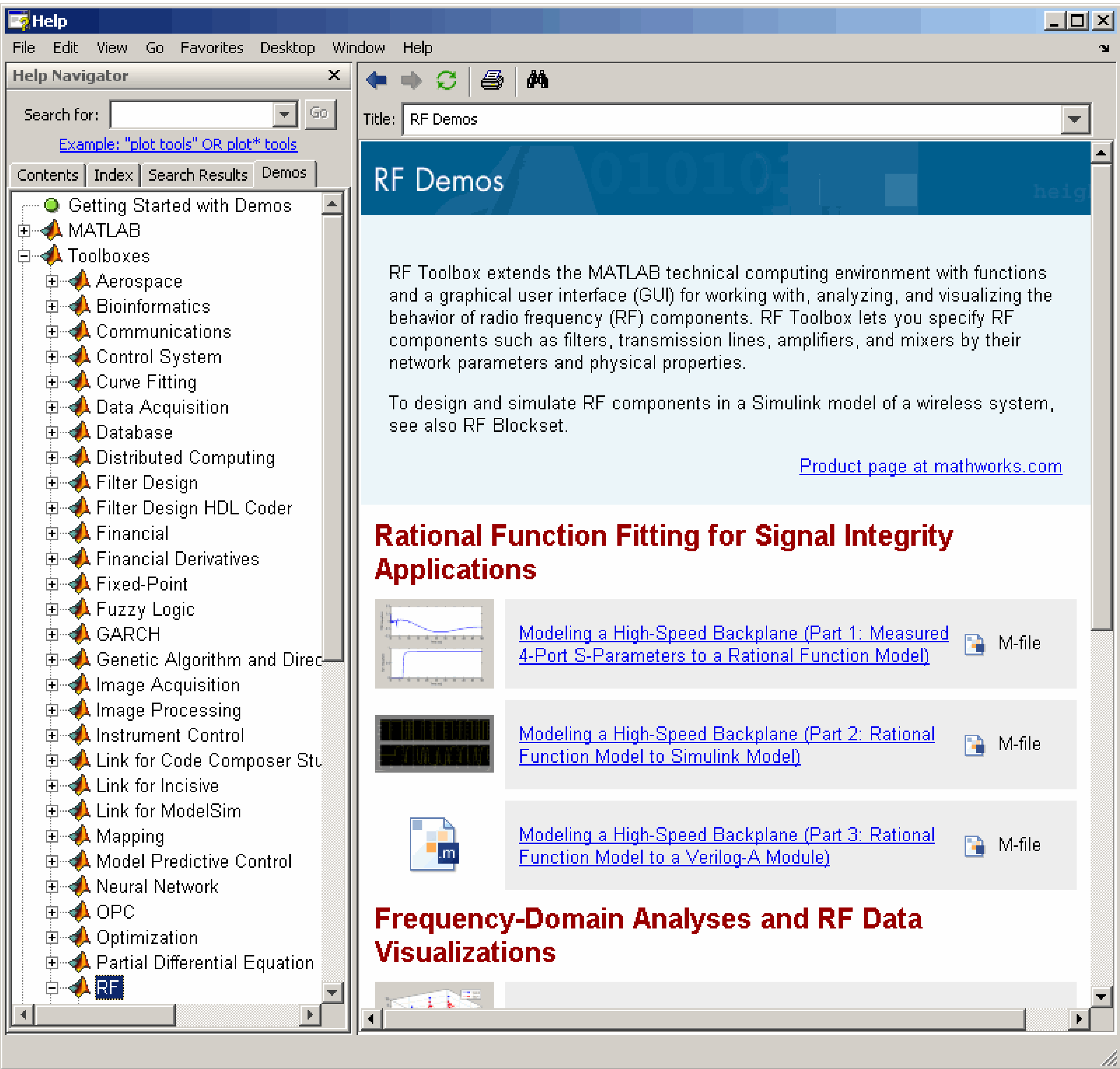
Related Products

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

Product	Description
Communications Blockset	Simulink blocks for time-domain simulation of modulation and demodulation of a wireless communications signal.
Communications Toolbox	MATLAB functions for signal modulation and demodulation.
Filter Design Toolbox	MATLAB functions for filtering the modulated communication signal.
RF Blockset	Time-domain simulation of RF components in Simulink.
Signal Processing Toolbox	MATLAB functions for filtering the modulated communication signal.
Signal Processing Blockset	Simulink blocks for time-domain simulation of for filtering the modulated communication signal.

Product Demos

You can find interactive RF Toolbox demos in the MATLAB Help browser.



To locate and open an RF Toolbox demo:

1 Type `demos` at the MATLAB prompt to open the Help browser to the **Demos** tab.

2 Select **Toolboxes > RF** in the **Demos** tab to see a list of demo categories.

3 Select a model, and click **Run in the Command Window** in the upper-right corner of the demo window to run the demo.

RF Objects

RF Toolbox 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 RF 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 2-9	`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 "Creating RF Objects" on page 3-2.

You use a different form of dot notation to specify object properties, as described in "Direct Property Referencing Using Dot Notation" on page 3-16. This is just one way to define component data. For more information on object properties, see "Specifying or Importing Component Data" on page 3-5.

You use object methods to perform frequency-domain analysis and visualize the results. For more information, see "Analyzing and Plotting RF Components" on page 3-20.

Note RF Toolbox also provides a graphical interface for creating and analyzing circuit objects. For more information, see Chapter 5, "RF Tool: An RF Analysis GUI".

RF Toolbox Workflow

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

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

   See Chapter 2, "Selecting an RF Object".

2. Create the selected objects.

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

3. Define component data by:

   - Specifying network parameters or physical properties (see "Setting 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 "Importing Property Values from Data Files" on page 3-8).
   - Where applicable, selecting operating condition values (see "Specifying Operating Conditions" on page 3-18).

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

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

5. Analyze the network in the frequency domain.

   See "Analyzing Networks in the Frequency Domain" on page 3-20.

6. Generate plots to gain insight into network behavior.

   The following plots and charts are available in RF Toolbox:

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

See “Visualizing Component and Network Data” on page 3-21.

7. Compute the network transfer function.

   See “Computing the Network Transfer Function” on page 3-30.

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

   See “Fitting a Model Object to Circuit Object Data” on page 3-30.

9. Plot the time-domain response.

   See “Computing and Plotting the Time-Domain Response” on page 3-31.

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

    See Chapter 4, “Exporting Verilog-A Models”.

Example — Modeling a Cascaded RF Network

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 "Example — Modeling an RF Network Using RF Tool" on page 5-30.

The network that you use in this example consists of an amplifier and two transmission lines. RF 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.

This example illustrates how to perform the following tasks:

- "Creating RF Components" on page 1-11
- "Specifying Component Data" on page 1-11
- "Validating RF Components" on page 1-12
- "Building and Simulating the Network" on page 1-14
- "Analyzing Simulation Results" on page 1-15

Creating 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;
```

Specifying 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`:

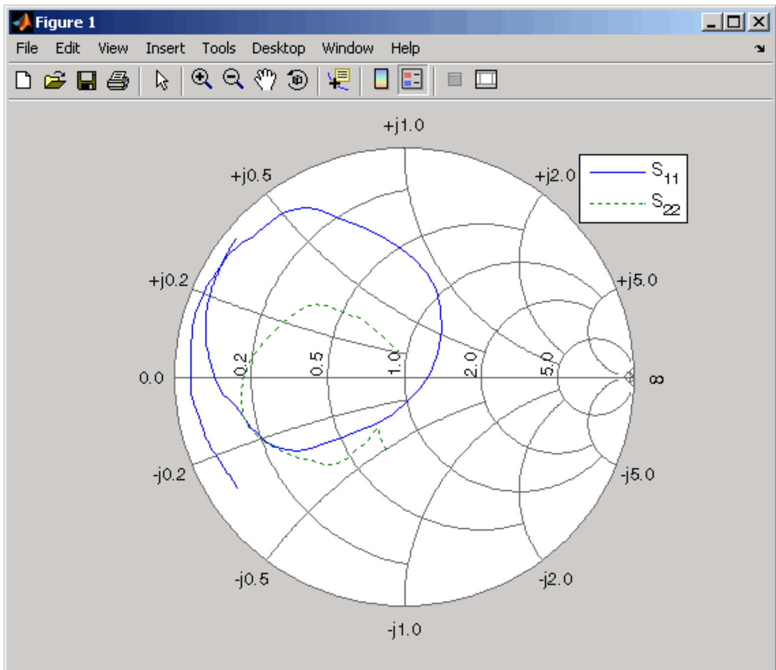
```
set(SecondCkt,'IntpType','cubic')
```

The `IntpType` property tells RF 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.

Validating 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.

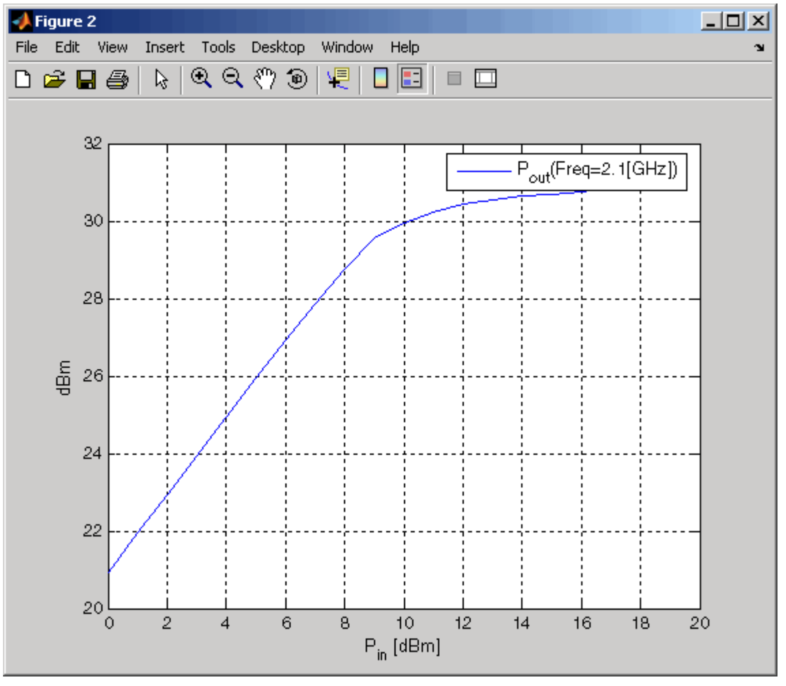
1 Type the following set of commands at the MATLAB prompt to use the `smith` command to plot the original S_{11} and S_{22} parameters of the amplifier (`SecondCkt`) on a Z Smith chart:



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:

```
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.

Building and Simulating 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});
```

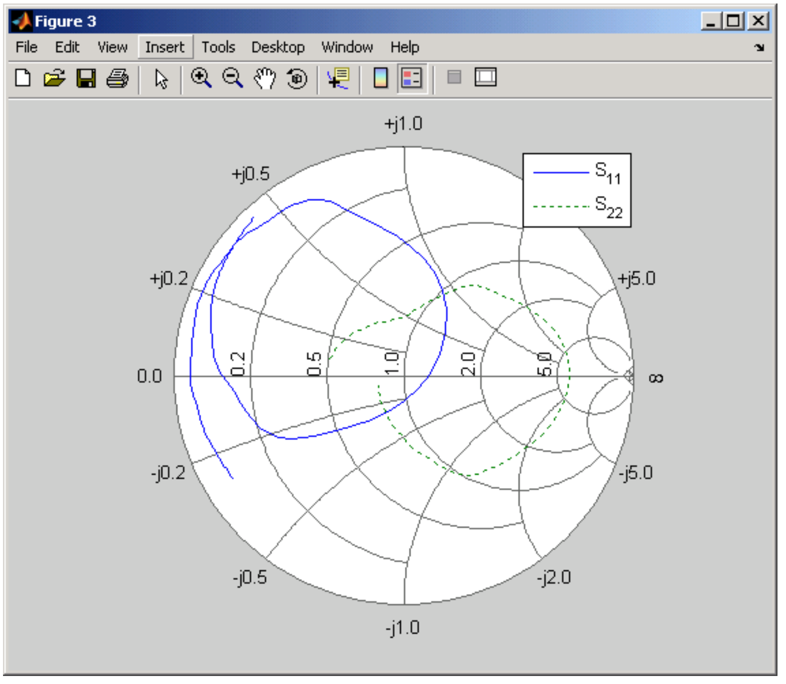
2 Type the following set of commands at the MATLAB prompt to define the range of frequencies over which to analyze the cascaded circuit, and then run the analysis:

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

Analyzing Simulation Results

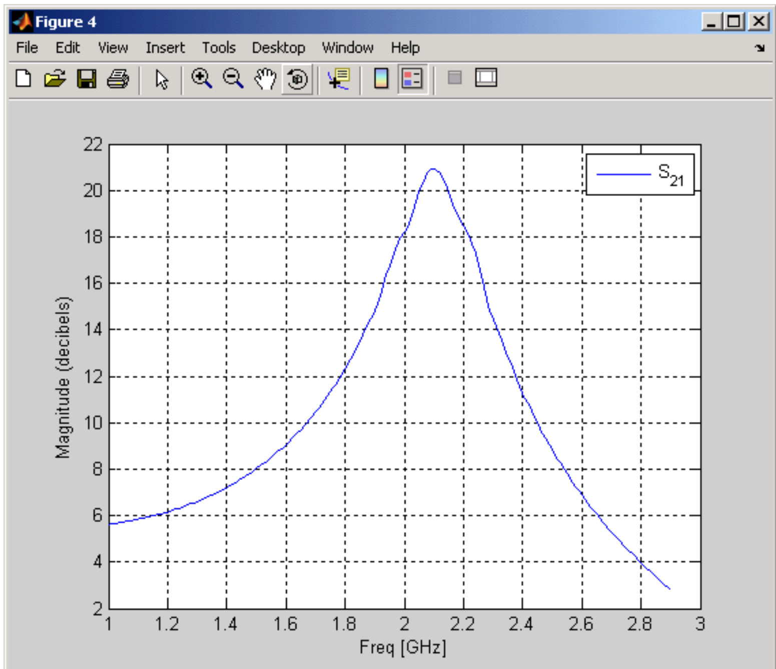
In this part of the example, you analyze the results of the simulation by plotting data for the circuit object that represents the cascaded amplifier network.

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:



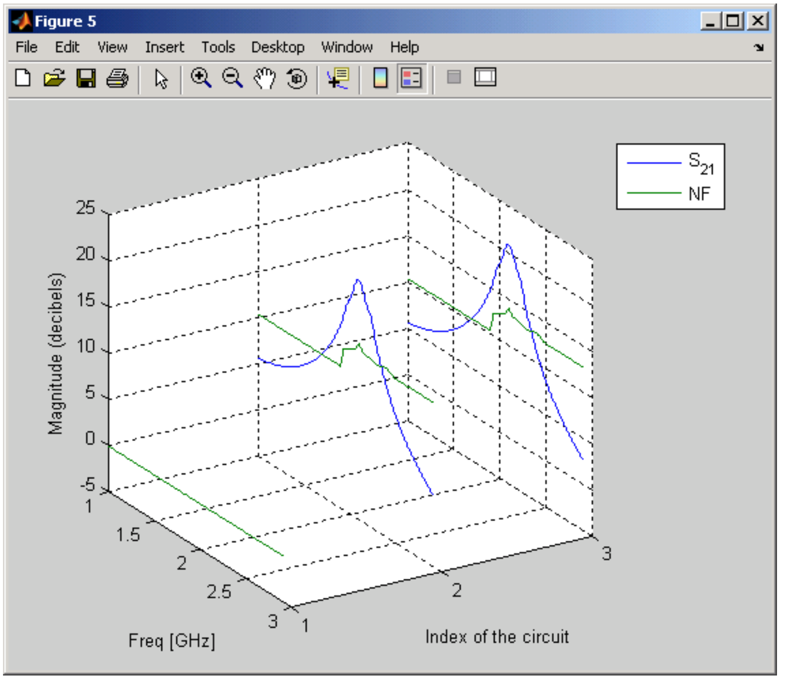
2 Type the following set of commands at the MATLAB prompt to use the `plot` command to plot the S_{21} parameter of the cascaded network, which represents the network gain, on an X-Y plane:

```
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:

```
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.

Example — Using a Rational Function Model to Analyze a Transmission Line

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.

This example illustrates how to perform the following tasks:

- "Building and Simulating the Transmission Line" on page 1-19
- "Computing the Transmission Line Transfer Function and Time-Domain Response" on page 1-19
- "Exporting a Verilog-A Model" on page 1-24

Building and Simulating 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);
```

Computing the Transmission Line Transfer Function and Time-Domain Response

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

- "Calculating the Transfer Function" on page 1-20
- "Fitting and Validating the Transfer Function Model" on page 1-20

- "Computing and Plotting the Time-Domain Response" on page 1-23

Calculating 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);
```

Fitting and Validating the Transfer Function Model

In this part of the example, you fit a rational function model to the transfer function. RF 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)
```

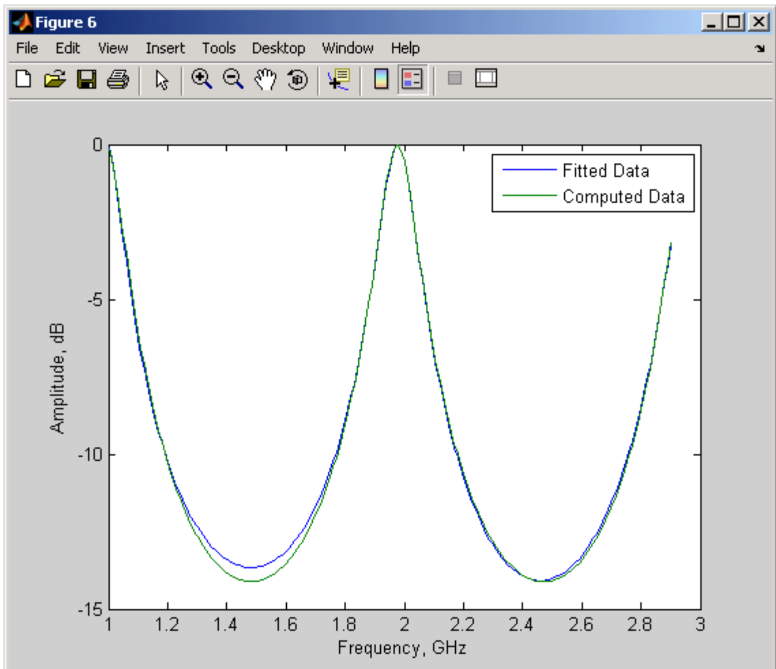
2 Type the following command at the MATLAB prompt to compute the frequency response of the fitted model data:

```
[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')
```

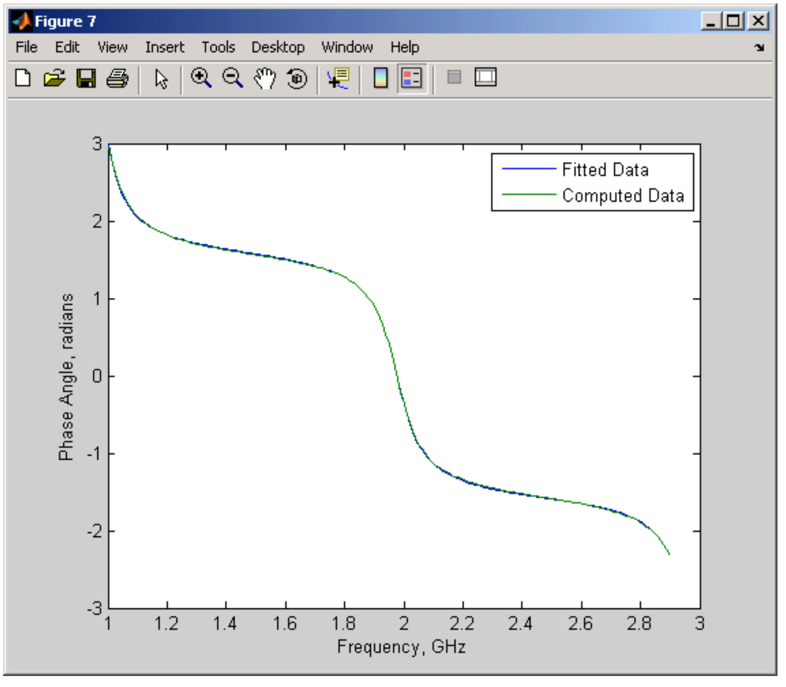
Note The amplitude of the model data is very close to the amplitude of the computed data. You can control the tradeoff between model accuracy and model complexity by specifying the optional tolerance argument, `tol`, to the `rationalfit` function, as described in "Representing a Circuit Object with a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



Computing and Plotting the Time-Domain Response

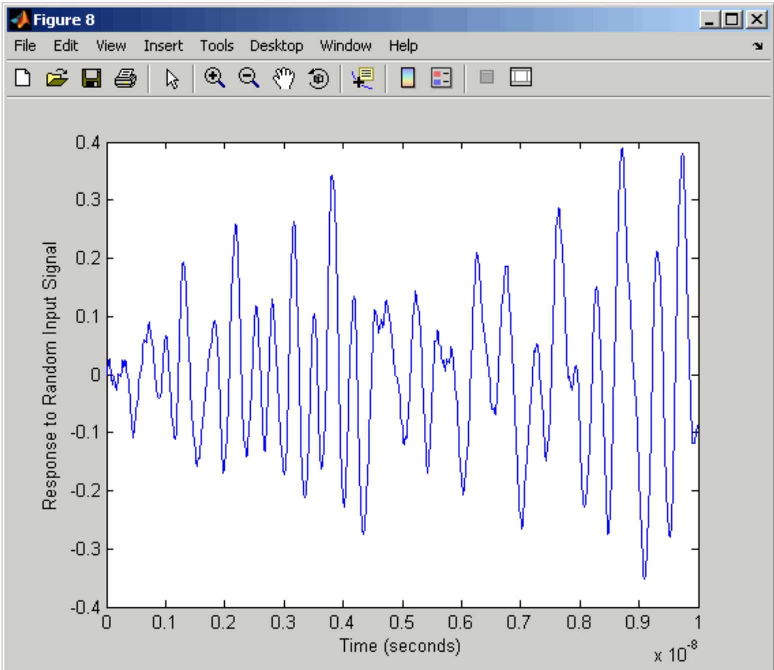
In this part of the example, you compute and plot the time-domain response of the transmission line.

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')
```



Exporting 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 Chapter 4, “Exporting Verilog-A Models”.

Selecting an RF Object

RF Data Objects

RF Toolbox 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.

This section contains the following topics:

Types of Data

RF 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

RF Toolbox 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

RF 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.

This section contains the following topics:

- "Components Versus Networks" on page 2-4
- "Available Components and Networks" on page 2-5
- "Circuit Object Methods" on page 2-7

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 "Example — Modeling 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
`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

Constructor	Description
`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
`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

Constructor	Description
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.
getz0	rfckt.txline, rfckt.rlcgline, rfckt.twowire, rfckt.parallelplate, rfckt.coaxial, rfdata.microstrip, rfckt.cpw	Get characteristic impedance of a transmission line.

Method	Types of Objects	Purpose
`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
`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

RF Toolbox 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

This section contains the following topics:

- "Available Model Objects" on page 2-9
- "Model Object Methods" on page 2-9

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.

3

Modeling an RF Component

Creating RF Objects

You create an RF object by performing one of the following tasks:

- "Constructing a New Object" on page 3-2
- "Copying an Existing Object" on page 3-4

Constructing 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:

- "Calling the Object Constructor" on page 3-2
- "Fitting a Rational Function to Passive Component Data" on page 3-3

Calling the Object Constructor

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

```
h = ObjectConstructorName
```

where `h` is the handle to the new object and `ObjectConstructorName` is the name of the object constructor. `ObjectConstructorName` is of the form `type.name`, where

- `type` is the object type (`rfdata`, `rfckt` or `rfmodel`).
- `name` is the object name.

For example, the `ObjectConstructorName` for an RLCG transmission line object is `rfckt.rlcgline` because the RLCG transmission line object is a circuit (`rfckt`) object named `rlcgline`.

For a list of the available object constructors for each type of object, see Chapter 2, "Selecting an RF Object".

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 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: 'None'
    Termination: 'None'
          Width: 6.0000e-004
         Height: 6.3500e-004
      Thickness: 5.0000e-006
       EpsilonR: 9.8000
      SigmaCond: Inf
    LossTangent: 0
```

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

Fitting 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 "Fitting a Model Object to Circuit Object Data" on page 3-30.

Copying 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 which has the same property values as the microstrip transmission line object with handle h.

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

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

Specifying or Importing Component Data

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 2-9 sections list the available types of objects and provide links to their reference pages.

This section contains the following topics:

Setting 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:

Specifying 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:

- “Constructing Components with Specified Properties” on page 3-6
- “Constructing Networks of Specified Components” on page 3-7

Constructing 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 RF 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: 'None'
        Termination: 'None'
        OuterRadius: 0.0026
        InnerRadius: 7.2500e-004
```

```
       MuR: 1
  EpsilonR: 2.3000
 SigmaCond: Inf
 SigmaDiel: 0
```

Constructing 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});
```

Changing 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 "Direct Property Referencing 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: {}
      SigmaCond: {}
      SigmaDiel: {}
```

Importing Property Values from Data Files

RF Toolbox 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

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

Object	Description	Supported Format(s)
`rfckt.amplifier`	Amplifier	Touchstone, AMP, P2D, S2D
`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 "Specifying Operating Conditions" on page 3-18.

Using 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)
```

Example — Setting 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;
```

Retrieving Property Values

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

This section contains the following topics:

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

Retrieving 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
```

Retrieving 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: 'None'
       Termination: 'None'
       OuterRadius: 0.0026
       InnerRadius: 7.2500e-004
               MuR: 1
          EpsilonR: 2.3000
         SigmaCond: Inf
         SigmaDiel: 0
```

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 RF Toolbox. The `AnalyzedResult` property value is calculated and set by the toolbox when you analyze the component at specified frequencies.

Direct Property Referencing 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 % gets property value
rfobj.PropertyName = PropertyValue % sets property value
```

These commands are respectively equivalent to

```
PropertyValue = get(rfobj,'PropertyName')
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 either of the commands

```
ckt.IntpType
```

```
ckt.in
```

produces

```
ans =
  Cubic
```

Specifying 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 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.

This section contains the following topics:

- "Setting Operating Conditions" on page 3-18
- "Displaying Available Operating Condition Values" on page 3-19

Setting Operating Conditions

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

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

where

- `obj` 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`.

Displaying 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`.

Analyzing and Plotting RF Components

RF Toolbox provides a variety of methods that act on objects. The methods let you perform operations such as

- "Analyzing Networks in the Frequency Domain" on page 3-20
- "Visualizing Component and Network Data" on page 3-21
- "Computing and Plotting Time-Domain Specifications" on page 3-30

For a complete listing of the available methods, listed by category, see Chapter 8, "Methods — By Category".

Analyzing Networks in the Frequency Domain

RF Toolbox 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, RF Toolbox performs a frequency-domain analysis when you use the `read` method to import the data.

When you analyze a circuit object, RF 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.

Visualizing Component and Network Data

RF 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

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

Plot Type	Methods	Plot Contents
Budget Plot (3-D)	`plot`	Parameters as a function of frequency for each component 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 RF 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 3-23
- “Budget” on page 3-23
- “Mixer Spur” on page 3-26
- “Polar Plots and Smith Charts” on page 3-29

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 RF 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

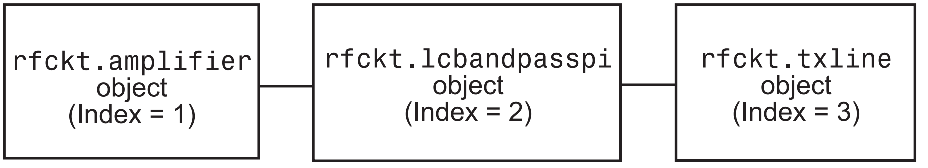
You use the budget plot to understand the individual contribution of each component to a plotted parameter value in a cascaded network with multiple components.

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})
```

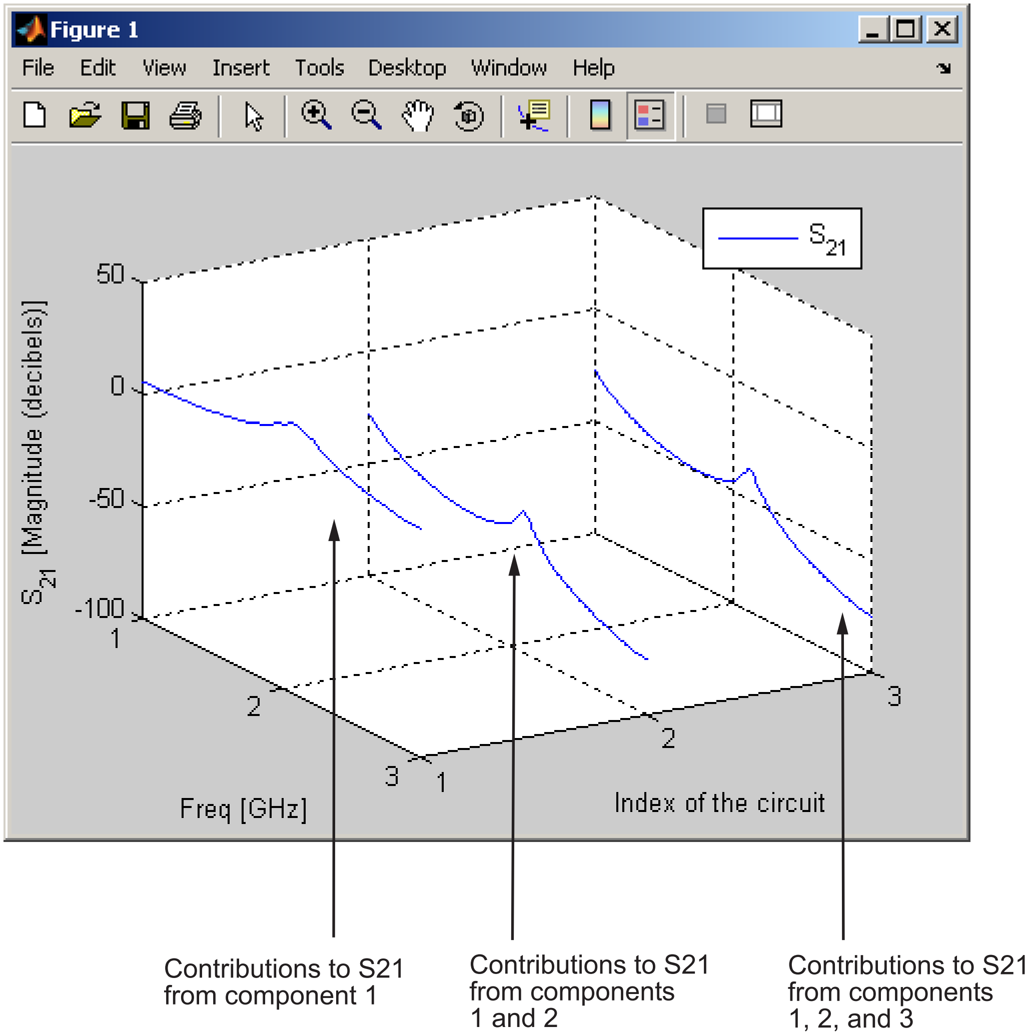
The following figure shows how the circuit index is assigned to each component in the cascade, based on its sequential position in the network.



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.



Example — Budget Plot

If you specify two or more parameters, RF 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.

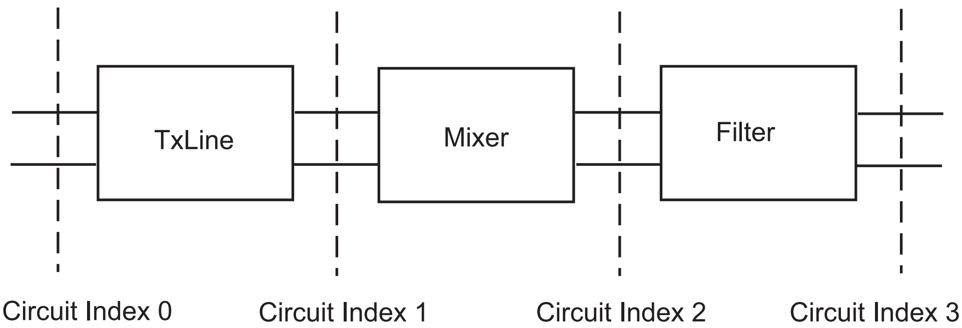
RF Toolbox calculates the output power from the mixer intermodulation table (IMT). These tables are described in detail in the Visualizing Mixer Spurs demo.

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.txline;
SecondCkt = read(rfckt.mixer, 'samplespur1.s2d');
ThirdCkt = rfckt.lcbandpasspi('L', [0.0144, 0.4395, 0.0144]*1.0e-7, ...
    'C', [0.3578, 0.0118, 0.3579]*1.0e-10);
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.

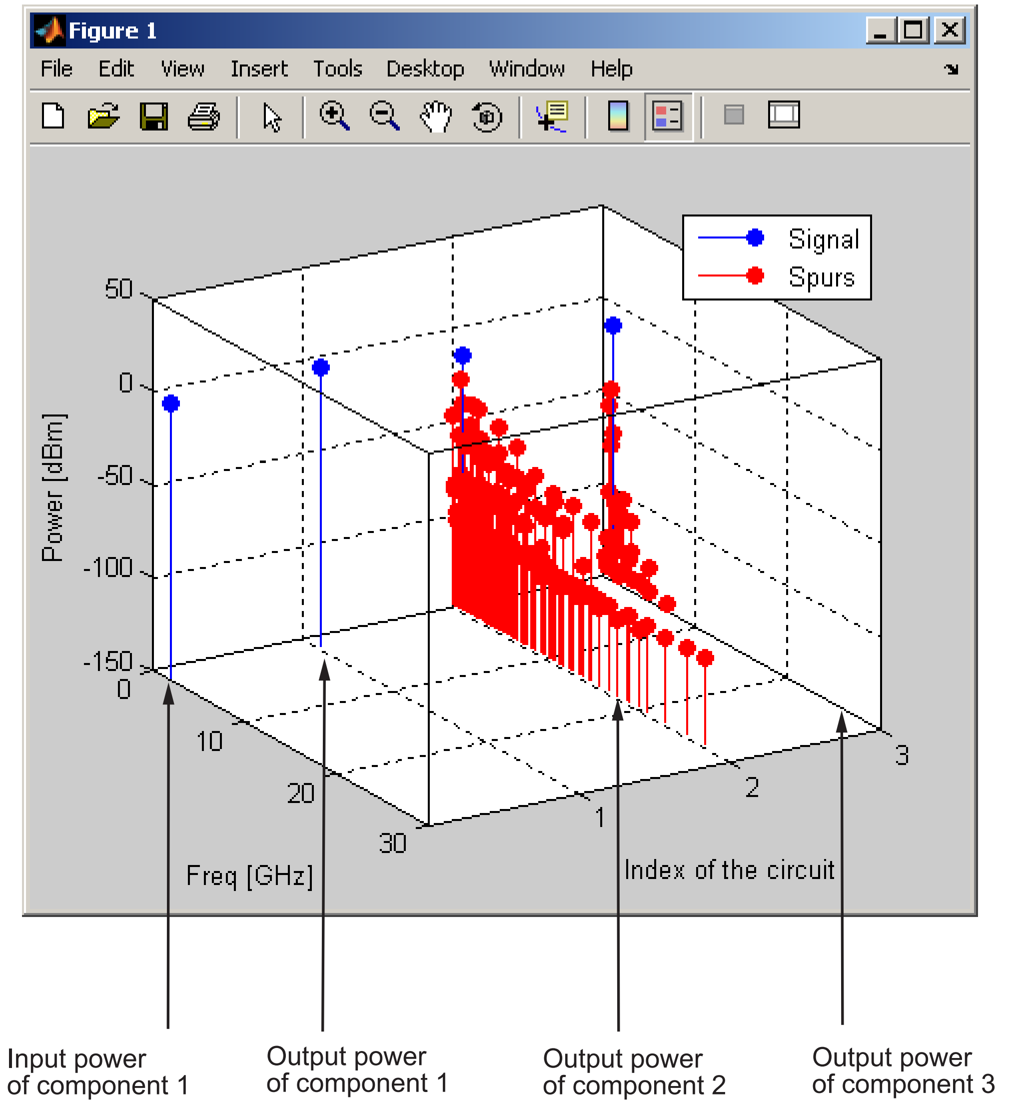


- Circuit index 0 corresponds to the cascade input.
- Circuit index 1 corresponds to the transmission line output.
- Circuit index 2 corresponds to the mixer output.
- Circuit index 3 corresponds to the filter output.

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.



Example — Mixer Spur Plot

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

Polar Plots and Smith Charts

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

The following table describes the Polar plot and Smith chart 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 parameter is plotted 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.

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

Computing and Plotting Time-Domain Specifications

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

This section contains the following topics:

Computing 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);
```

Fitting 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 RF 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 "Representing 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']);
```

Computing and Plotting the Time-Domain Response

You use the `timresp` 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.

Examples of Basic Operations with RF Objects

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

- "Reading and Analyzing RF Data from a Touchstone Data File" on page 3-33
- "De-Embedding S-Parameters" on page 3-35
- "Impedance Matching" on page 3-40

Reading and Analyzing 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.

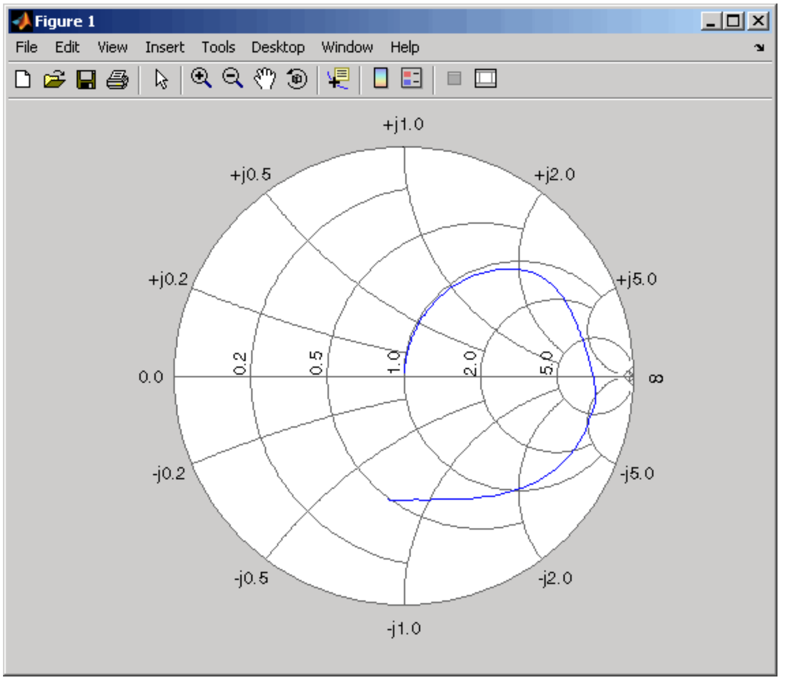
```
data = read(rfdata.data,'passive.s2p');
```

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:

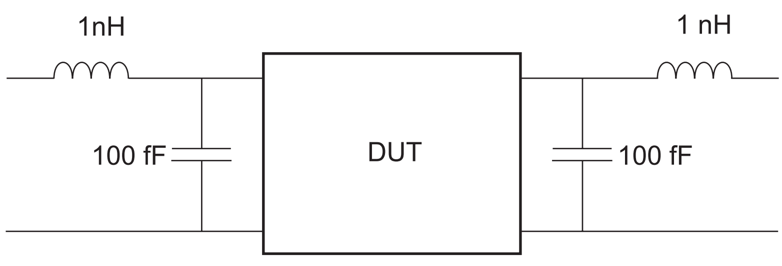
```
f =
  6.0000e+009

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-Embedding 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.

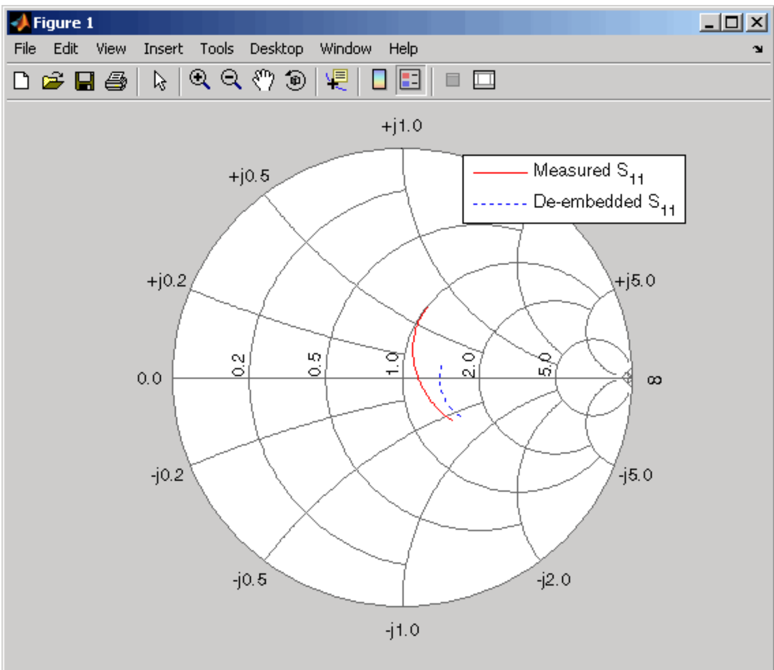
```
z0 = measured_data.Z0;

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);
```

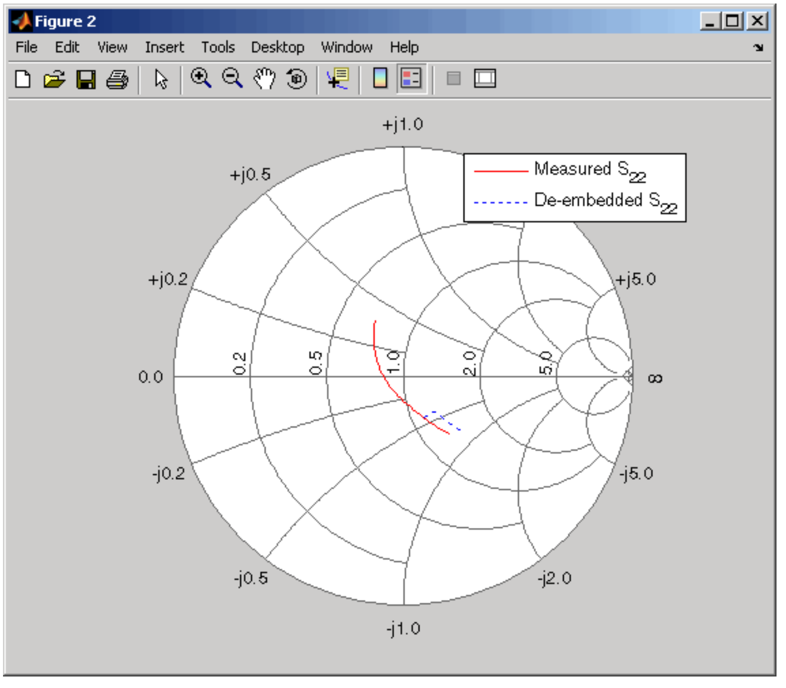
5 Plot the measured and de-embedded S_{11} parameters. Type the following set of commands at the MATLAB prompt to plot both the measured and the de-embedded S_{11} parameters on a Z Smith chart:

```
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;
```



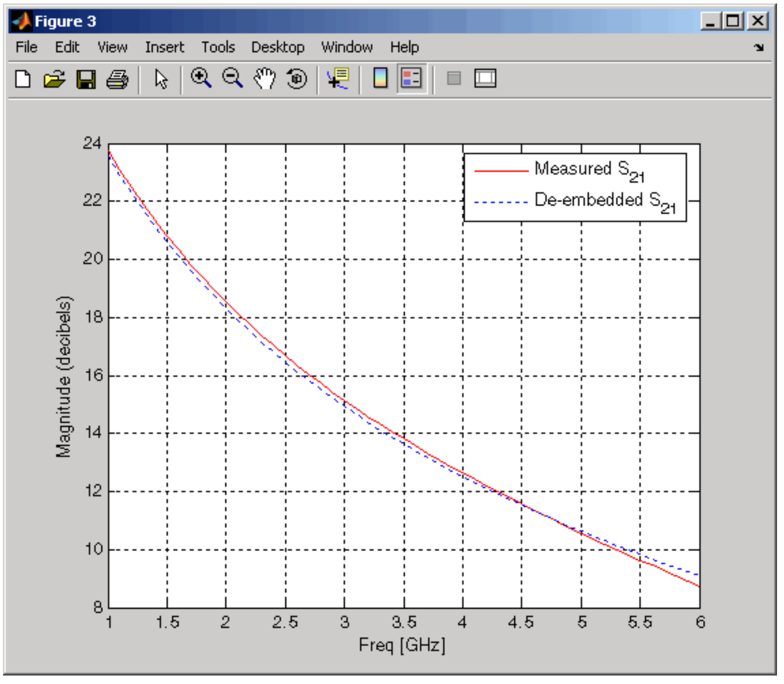
6 Plot the measured and de-embedded S_{22} parameters. Type the following set of commands at the MATLAB prompt to plot the measured and the de-embedded S_{22} parameters on a Z Smith chart:

```
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;
```



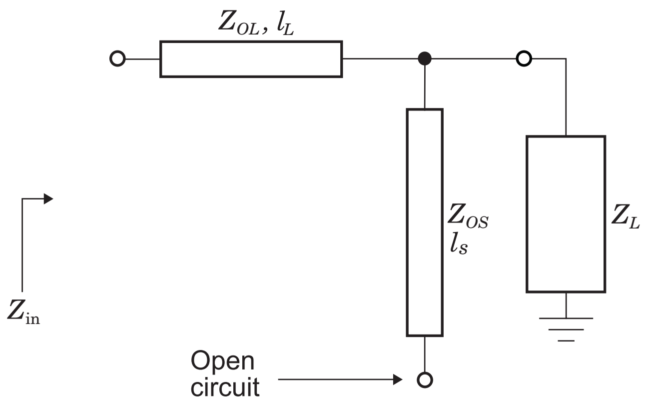
7 Plot the measured and de-embedded S_{21} parameters. Type the following set of commands at the MATLAB prompt to plot the measured and the de-embedded S_{21} parameters, in decibels, on an X-Y plane:

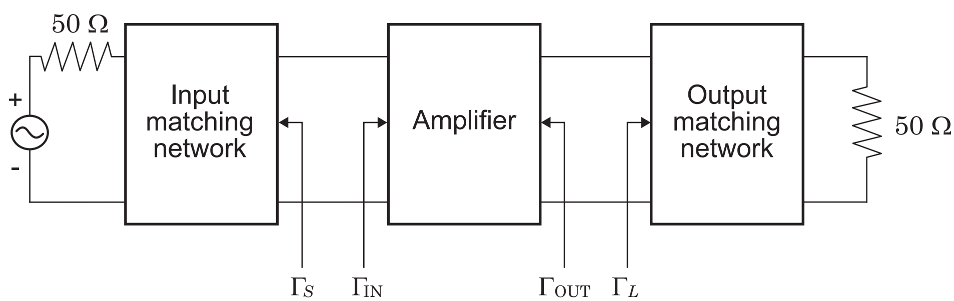
```
figure
hold off;
h = plot(measured_data,'S21', 'db');
set(h, 'Color', [1 0 0]);
hold on
i = plot(de_embedded_data,'S21','db');
set(i,'Color', [0 0 1],'LineStyle',':');
l = legend;
legend(l, {'Measured S_{21}', 'De-embedded S_{21}'});
legend show;
hold off;
```

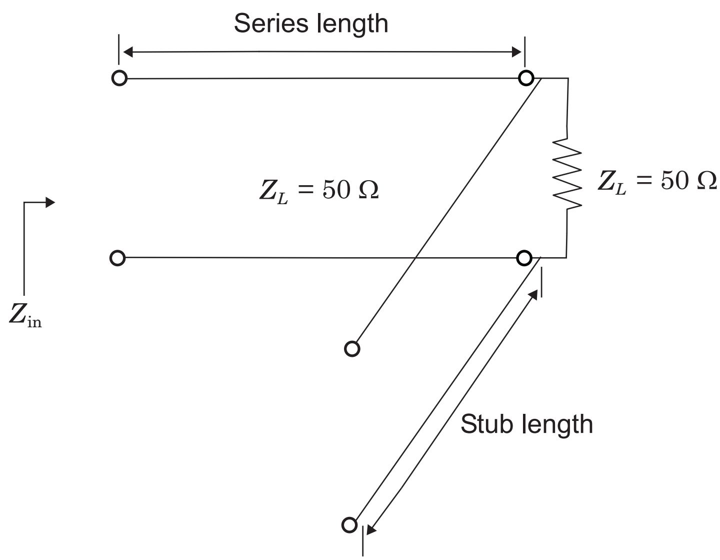


Impedance Matching

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







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

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

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

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

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

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

```
abs_delta = abs(delta)
```

The toolbox displays the following output:

```
K =

    1.0599

abs_delta =

    0.6776
```

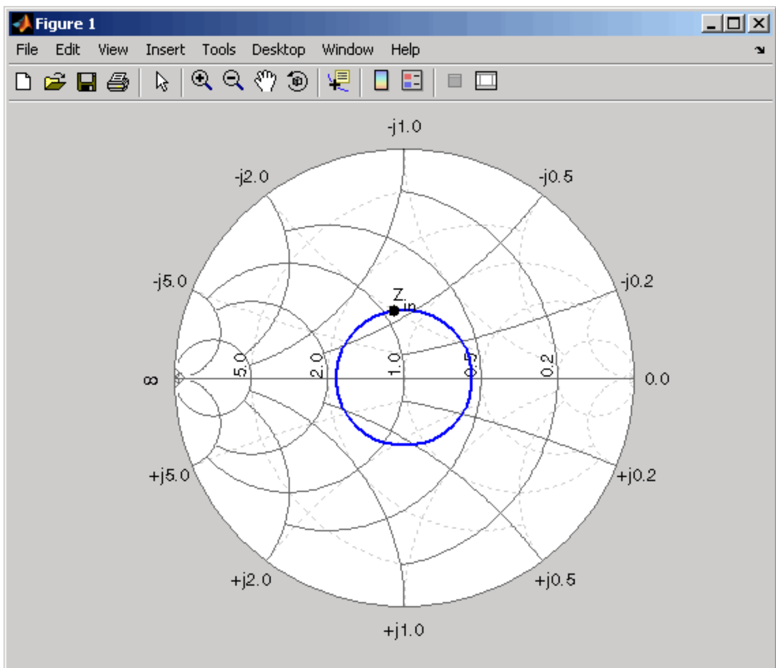
3 **Find the source and load reflection coefficients.** To design input and output matching networks, you must calculate the required source and load reflection coefficients that produce a simultaneous conjugate match. You can calculate the load reflection coefficient, `gammaL`, using the amplifier S-parameters.

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

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

desired input impedance point and the input SWR circle on a ZY Smith chart.

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

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



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

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

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

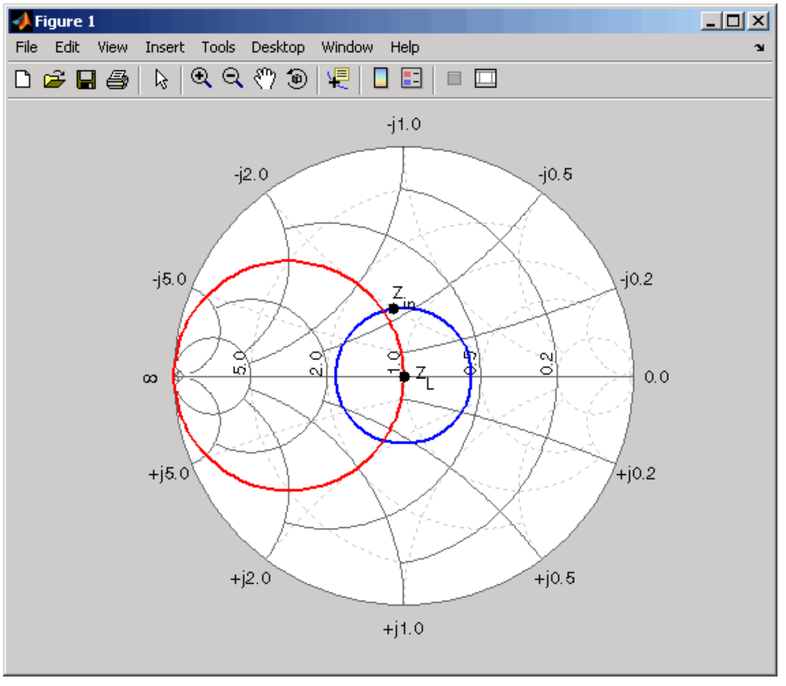
```
g  = real(yL); %g=1
d = -(g-1)/(g+1)+1; %d=1
C = -1+d/2; %C= 1/2
```

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

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

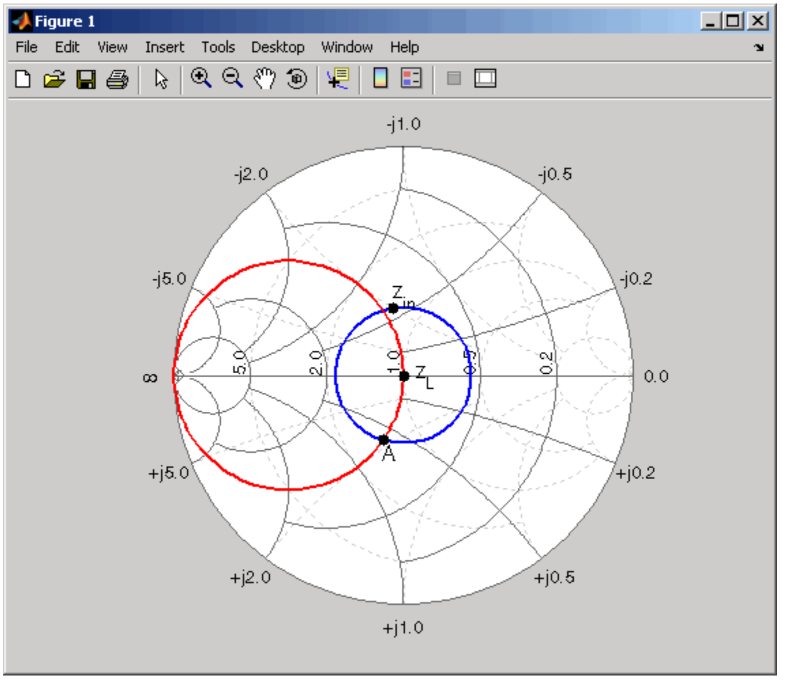
Finally, you plot and label the load impedance point along with the constant conductance circle associated with the load admittance on the Smith chart.

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



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

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



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

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

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

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

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

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

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

```
stubLengthA =
    0.0883

seriesLengthA =
    0.2147
```

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

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

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

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

The toolbox displays the following output:

```
Z0 =
   50.2561
```

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

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

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

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

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

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

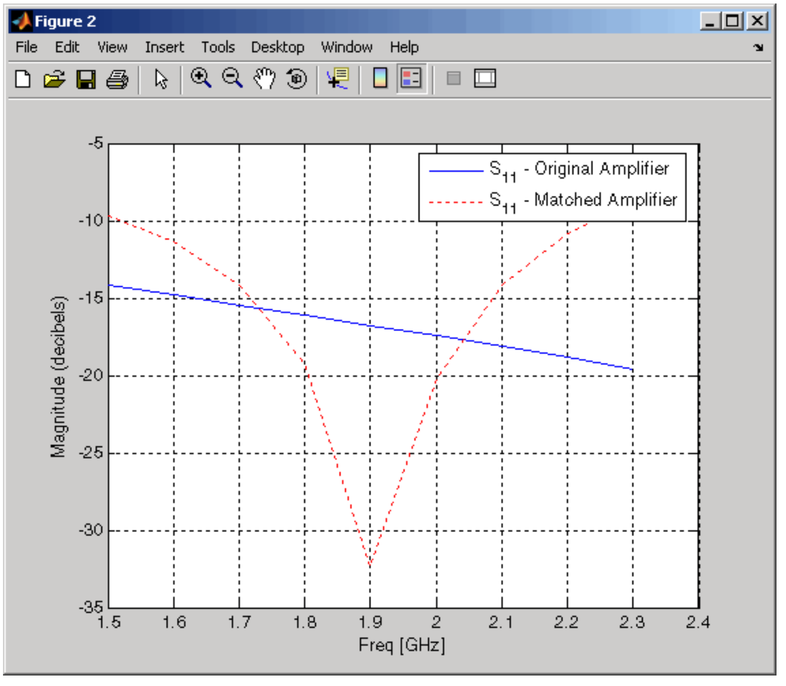
```
set(hstubOutput,'LineLength',phase_vel/1.9e9*stubLengthA,...
    'StubMode','shunt','Termination','open');
hstubInput = rfckt.microstrip(...
    'LineLength',phase_vel/2.1e9*stubLengthin,...
    'StubMode','shunt','Termination','open');
```

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

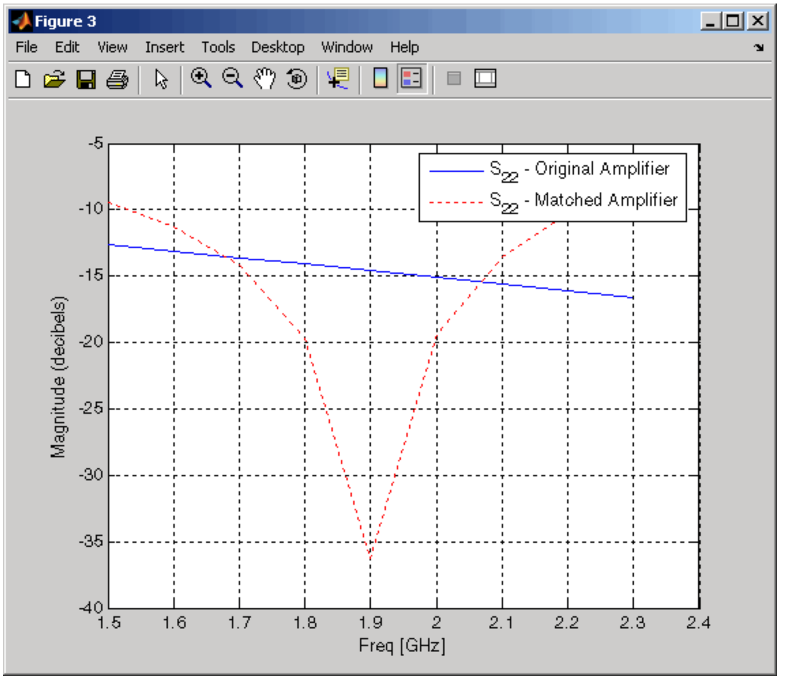
```
matched_amp = rfckt.cascade('Ckts',...
    {hstubInput,hseriesInput,amp,hseriesOutput,hstubOutput});
analyze(matched_amp,1.5e9:1e8:2.3e9);
analyze(amp,1.5e9:1e8:2.3e9);
```

To verify the simultaneous conjugate match at the input and output of the amplifier, plot S_{11} parameters and S_{22} parameters, in decibels, for both circuits:

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

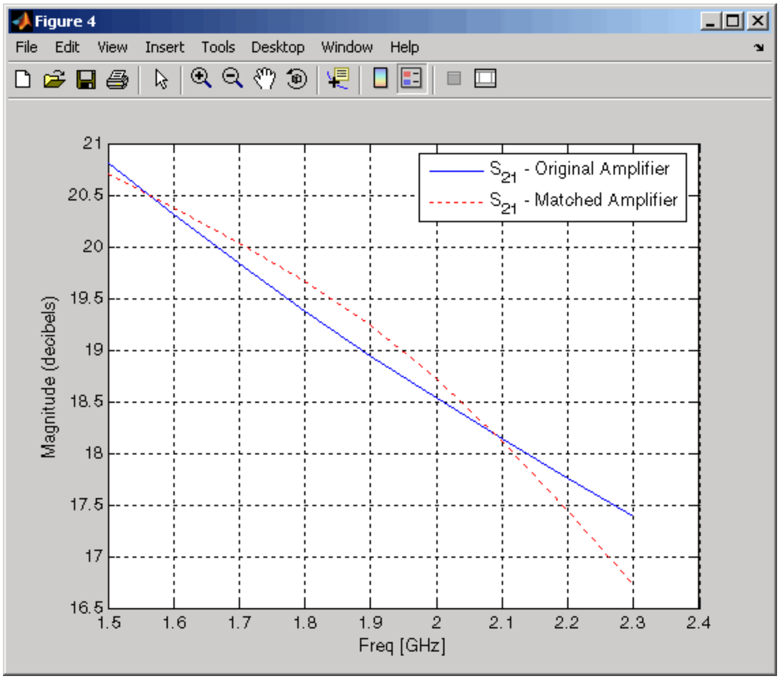


```
figure
hls(1) = plot(amp,'S22','dB');
hold on;
hls(2) = plot(matched_amp,'S22','dB');
set(hls(2),'Color',[1 0 0],'LineStyle',':');
legend(hls,'S_{22} - Original Amplifier',...
     'S_{22} - Matched Amplifier');
```



Finally, plot S_{21} parameters for both circuits:

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



You can compare the matched amplifier results with the expected transducer gain (in dB). From the S_{21} parameters plot, you can see that the gain of the matched amplifier at 1.9 GHz is between 19 dB and 19.5 dB. The expected gain is given by the following equation:

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

The toolbox displays the following output:

```
Gt =
   19.2407
```

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

Exporting Verilog-A Models

Modeling RF Objects Using Verilog-A

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 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 RF 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.

This section contains the following topics:

- "Behavioral Modeling Using Verilog-A" on page 4-2
- "Supported Verilog-A Models" on page 4-3

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, RF 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 lets you export a Verilog-A model of an `rfmodel` object. RF 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.

RF 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.
- N is the order of the denominator polynomial.
- n_k is the coefficient of the k^{th} power of s in the numerator.
- d_k is the coefficient of the k^{th} 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 RF 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. RF Toolbox converts the real poles of the rational function directly to a Laplace transform filter in numerator-denominator form.

How to Export a Verilog-A Model

To export a Verilog-A model of a component, you perform the following tasks:

- “Representing a Circuit Object with a Model Object” on page 4-5
- “Writing a Verilog-A Module” on page 4-7

An example of this export process appears in the RF Toolbox demo, “Modeling a High-Speed Backplane (Part 2: Rational Function Model to a Verilog-A Module”.

Representing 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.
- `data` is a vector that contains the data to fit.

- `freq` is a vector of frequency values that correspond to the data values.

For faster simulation, create a model object with the smallest number of poles required to accurately represent the component. Use the following arguments, which are described in detail in the `rationalfit` function reference page, to control the number of poles:

- `delayfactor` — controls the amount of delay used to fit the data. Specify a value that reflects the amount of delay in your 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 `delayfactor`, the `rationalfit` function represents the delay as an exponential phase shift. This phase shift allows the function to fit the data using fewer poles.
- `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.

The syntax is:

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

where `weight` is a vector that specifies the weighting of the fit at each frequency.

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.

Writing 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.

The following code illustrates how 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 a status of True if the operation is successful and False otherwise.

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

The writeva reference page describes the method arguments in detail.

RF Tool: An RF Analysis GUI

Introduction to 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.

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.

This section contains the following topics:

Opening 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 "Creating and Importing Circuits" 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 Sessions" on page 5-27.

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.

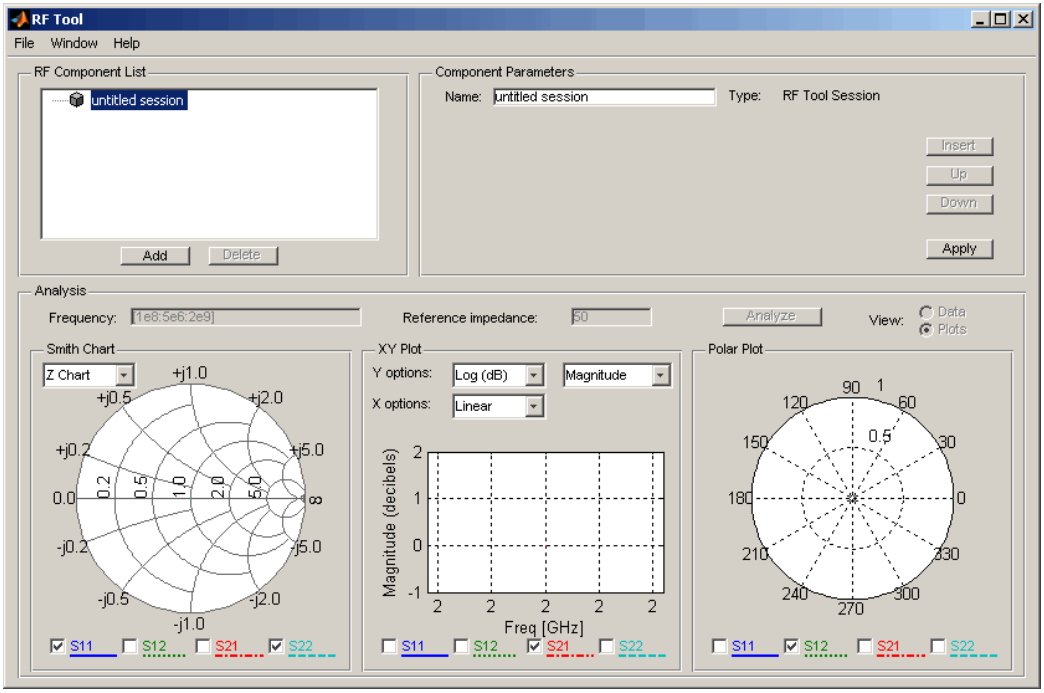
- **Component Parameters**

  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 “Creating and Importing Circuits” on page 5-6.

2. Specify component data.

   See “Modifying Component Data” on page 5-19.

3. Analyze the circuit.

   See “Analyzing Circuits” on page 5-20.

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

   See “Exporting RF Objects” on page 5-23.

Getting Help

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

Creating and Importing Circuits

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.

This section contains the following topics:

- "Creating RF Components" on page 5-6
- "Creating RF Networks" on page 5-10
- "Importing RF Objects" on page 5-15

Creating RF Components

This section contains the following topics:

- "Available RF Components" on page 5-6
- "How to Add an RF Component to a Session" on page 5-8

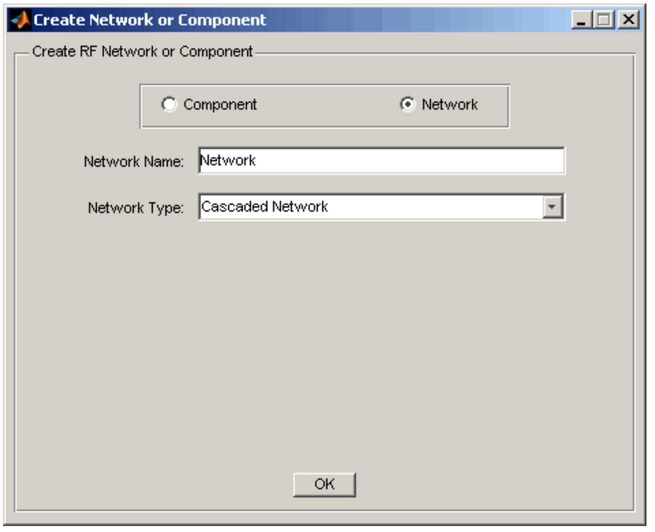
Available RF Components

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

RF Component	Corresponding RF Toolbox 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`

How to Add an RF Component to a Session

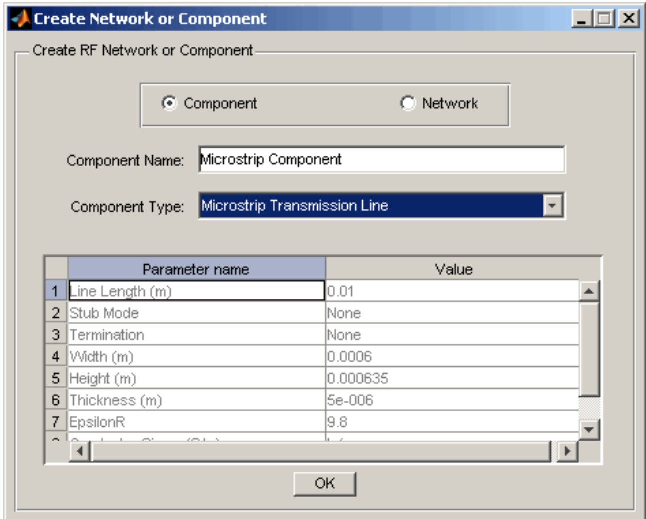
1 In the **RF Component List** pane, click **Add** to open the Create Network or Component dialog box.



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`.



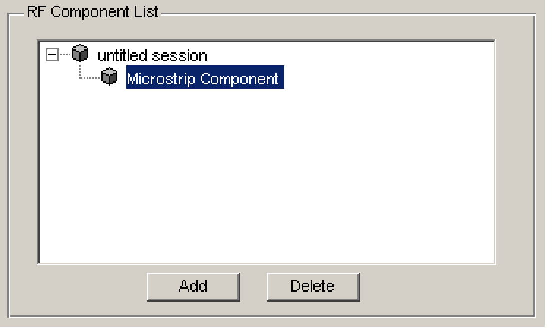
5 Adjust the parameter values as necessary.

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 "Modifying Component Data" on page 5-19.

---

6 Click **OK**.

RF Tool adds the component to your session.



Creating RF Networks

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 5-10
- "Adding an RF Network to a Session" on page 5-11
- "Populating an RF Network" on page 5-13
- "Reordering Circuits Within a Network" on page 5-14

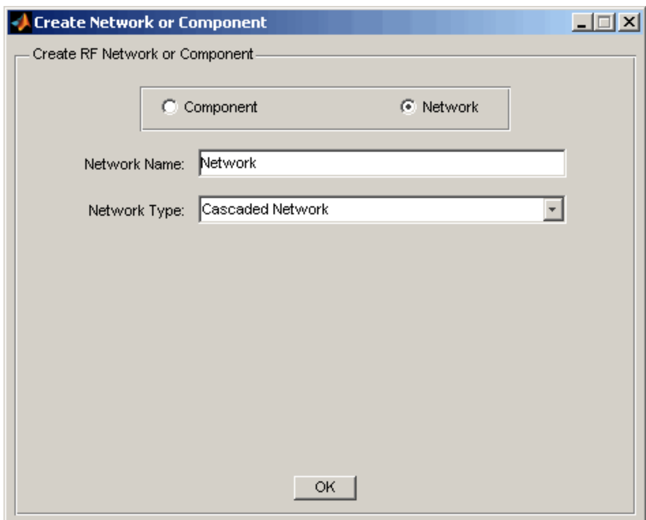
Available RF Networks

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`

Adding an RF Network to a Session

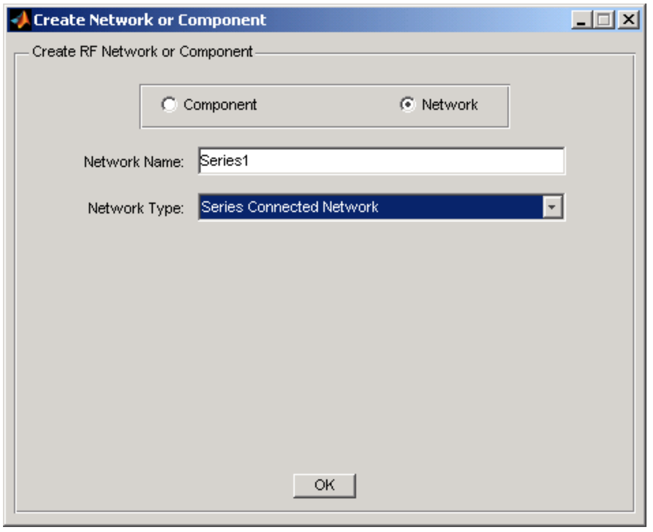
1 In the **RF Component List** pane, click **Add** to open the Create Network or Component dialog box.



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

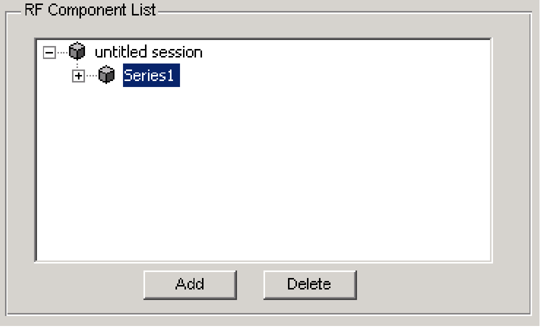
3 In the **Network Name** field, enter a name for the component. This name is used to identify the network in the **RF Component List** pane. For example, Series1.

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.

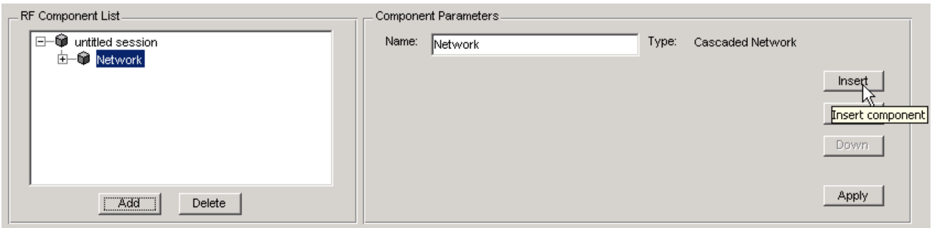


Populating an RF Network

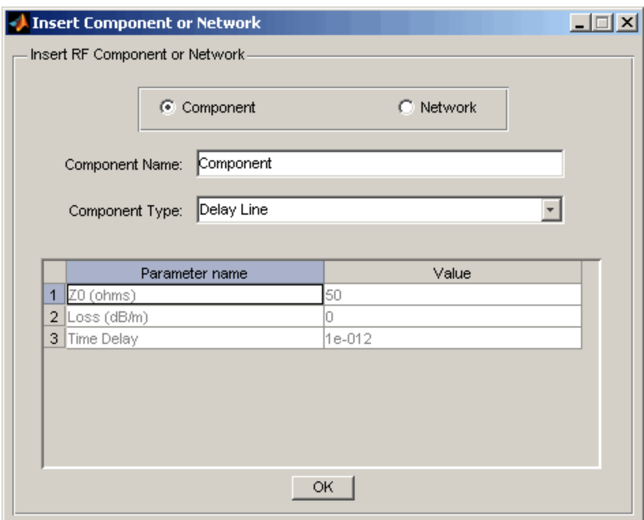
After you create a network using RF Tool, you must populate it with RF components and networks. You insert a component or network into a network in much the same way you add one to a session.

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**.



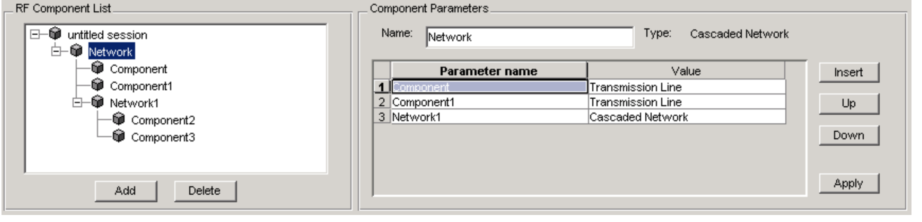
The Insert Component or Network dialog box appears.



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 "How to Add an RF Component to a Session" on page 5-8 or "Adding 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.



Reordering 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.

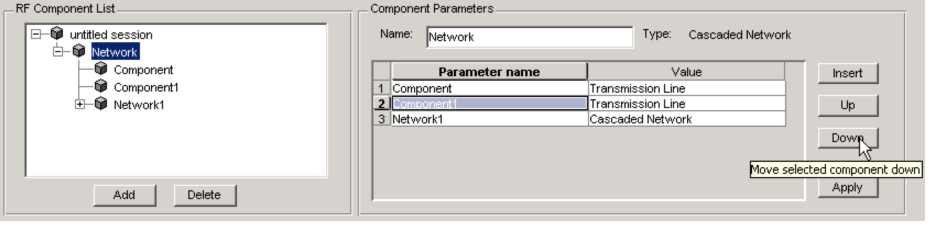
3 Click **Up** or **Down** until the circuit is where you want it.

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.



Importing RF Objects

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 “Exporting RF Objects” on page 5-23.

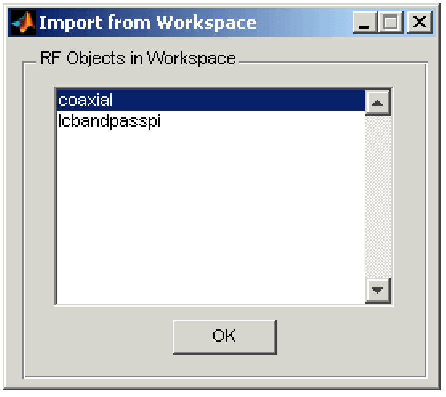
After you have imported an object, you can change its name and work with it as you would any other component or network.

This section contains the following topics:

Importing 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.

Importing from a File into a Session

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

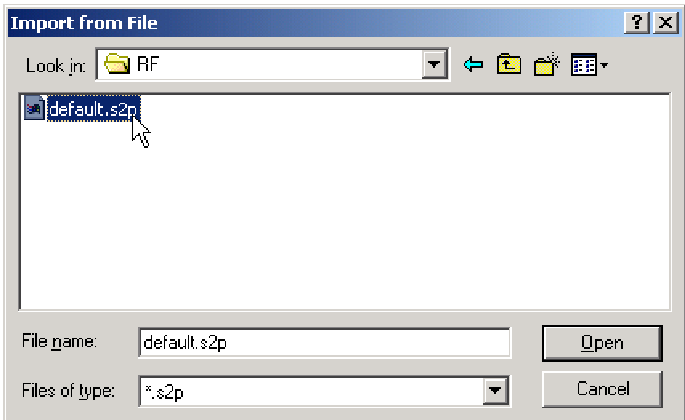
- S2P
- Y2P
- Z2P
- H2P

To import a component from one of these files:

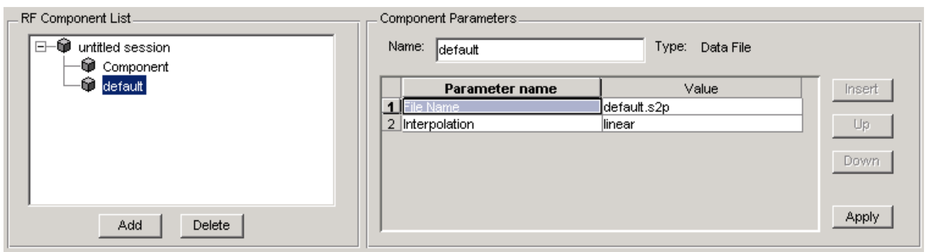
1 Select **Import From File** from the **File** menu. A file browser appears.

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.

Importing 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 "Populating 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.

"Example — Modeling an RF Network Using RF Tool" on page 5-30 shows this process.

Modifying Component Data

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.

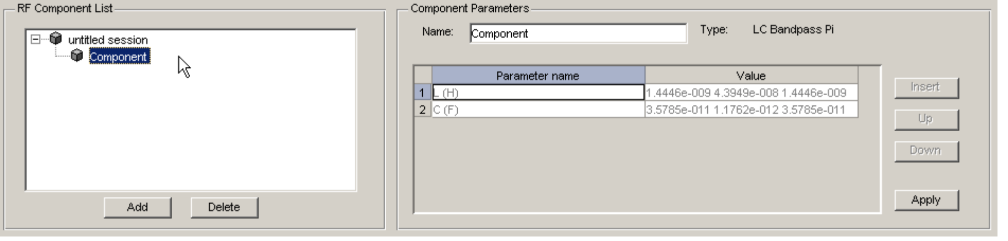
   Valid values for component parameters are listed on the corresponding RF Toolbox reference page. Use the links in "Available RF Components" on page 5-6 and "Available RF Networks" on page 5-10 to access these pages.

3. Click **Apply**.

Analyzing Circuits

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.

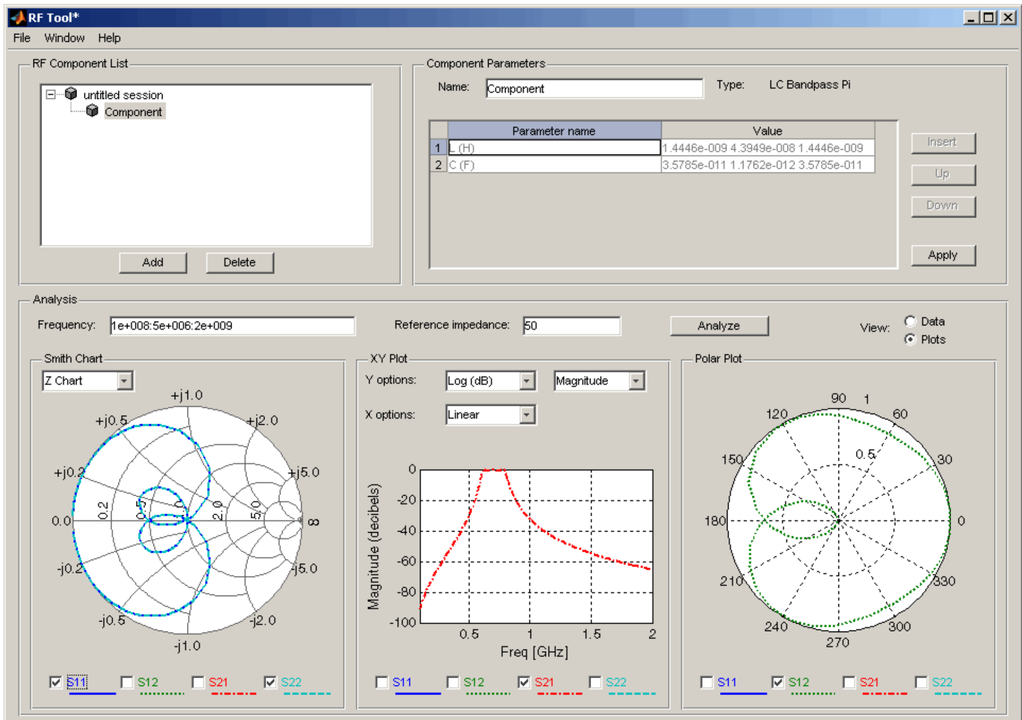
- Enter `50`, the reference impedance in ohms, in the **Reference impedance** field.

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

---

3 Click **Analyze**.

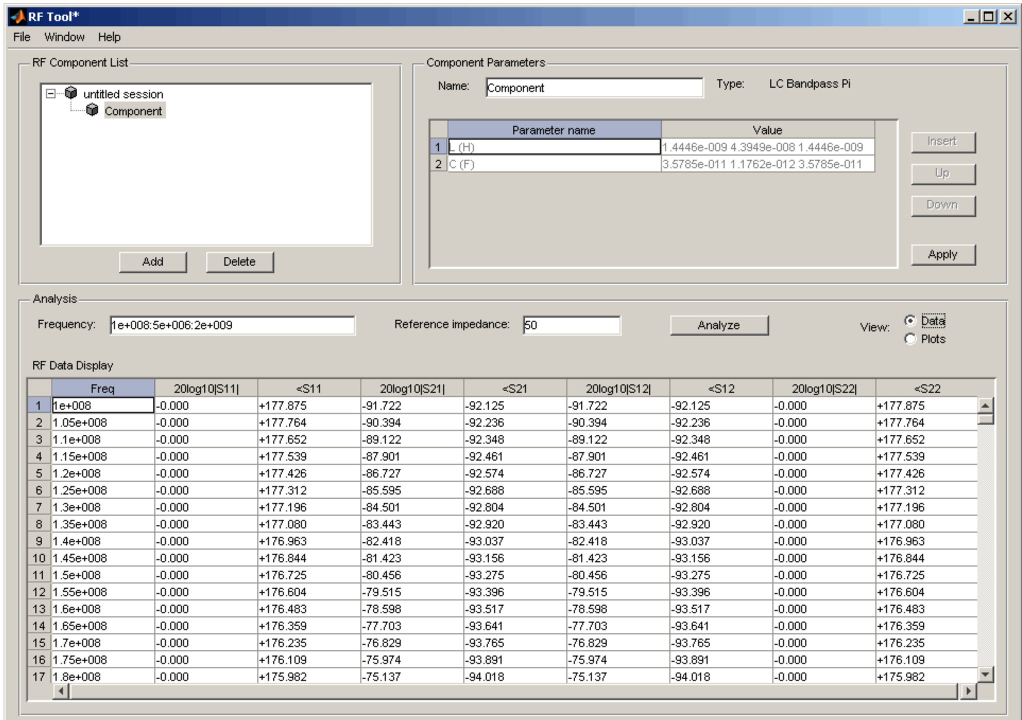
The **Analysis** pane displays Smith, XY, and polar plots of the analyzed circuit.



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.

Exporting RF Objects

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.

This section contains the following topics:

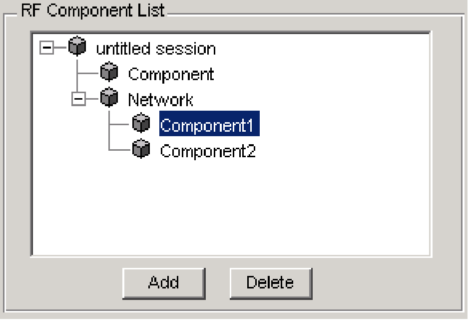
- "Exporting to the Workspace" on page 5-23
- "Exporting to a File" on page 5-24

Exporting 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.

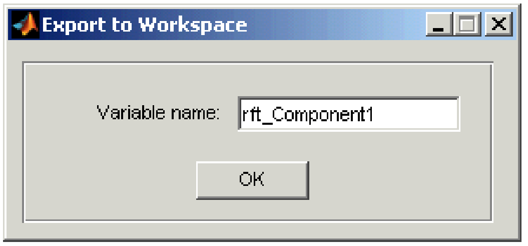
To export a component or network to the workspace:

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

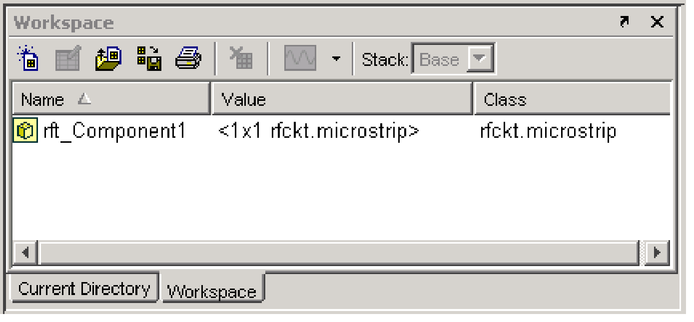


2 Select **Export to Workspace** from the **File** menu.

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.



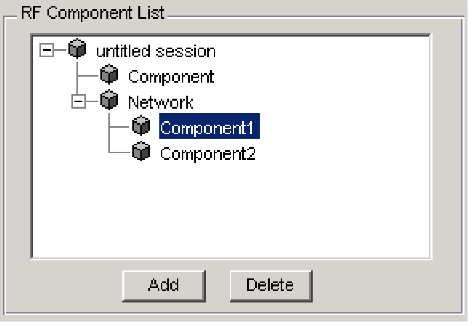
Exporting to a File

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

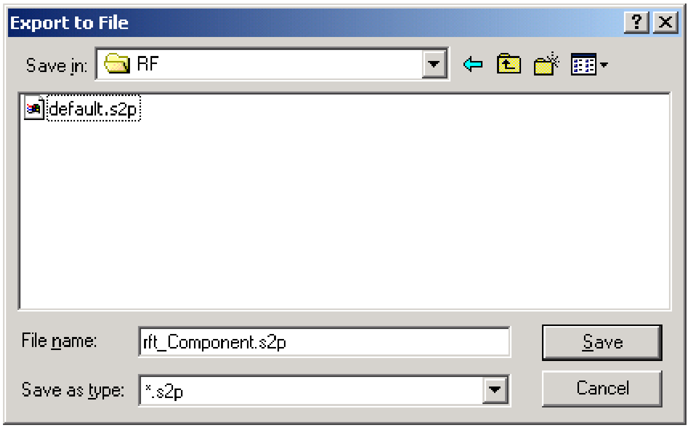
Note You must analyze a component or network in RF Tool before you can export it to a file. See "Analyzing Circuits" on page 5-20 for more information.

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 (_).

Managing Circuits and Sessions

This section contains the following topics:

- "Working with Circuits" on page 5-26
- "Working with Sessions" on page 5-27

Working with Circuits

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

- "Deleting Circuits" on page 5-26
- "Renaming Circuits" on page 5-27

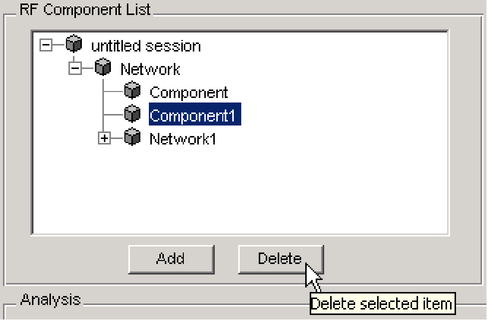
Deleting Circuits

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.



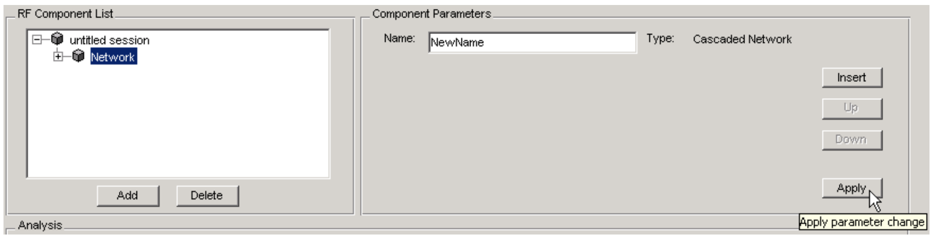
Renaming Circuits

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 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:

- "Naming or Renaming a Session" on page 5-27
- "Saving a Session" on page 5-28
- "Opening an Existing Session" on page 5-29
- "Starting a New Session" on page 5-29

Naming or Renaming a Session

To name or rename an RF 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.)

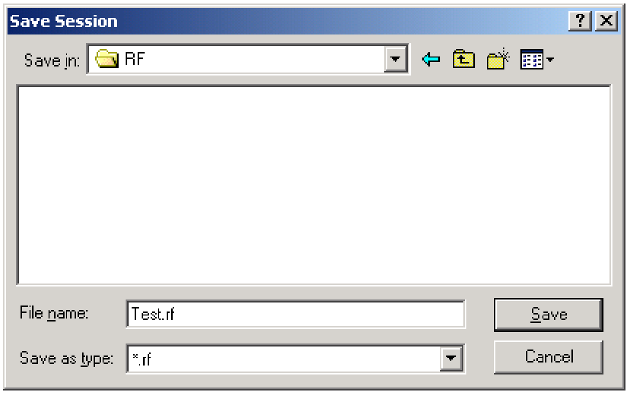
2 Type the desired name in the **Name** field of the **Component Parameters** pane.

3 Click **Apply**.

Saving a Session

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

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

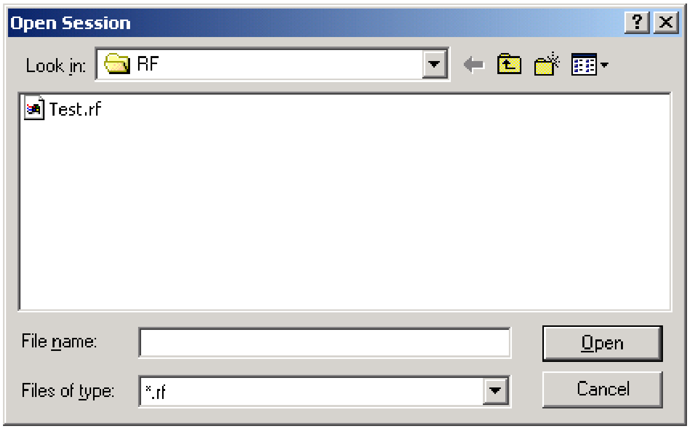


For example, to save your session as `Test.rf` in your current working directory, you would type `Test` in the **File name** field as shown above. RF Tool adds the `.rf` extension automatically to all RF Tool sessions you save.

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

Opening an Existing Session

You can load an existing session into 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.

Starting 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.

Example — Modeling an RF Network Using RF Tool

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

This example illustrates how to perform the following tasks:

- "Starting RF Tool" on page 5-30
- "Creating the Amplifier Network" on page 5-30
- "Populating the Amplifier Network" on page 5-33
- "Simulating the Amplifier Network" on page 5-37
- "Exporting the Network to the Workspace" on page 5-38

Starting RF Tool

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

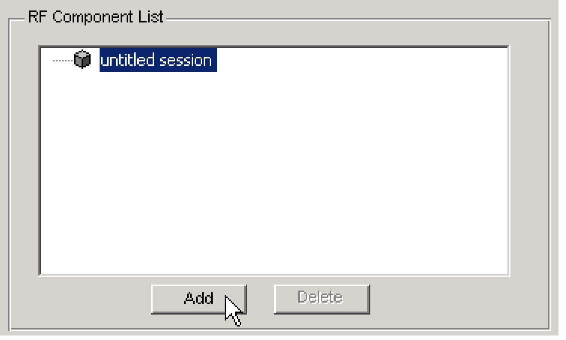
```
rftool
```

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

Creating 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**.



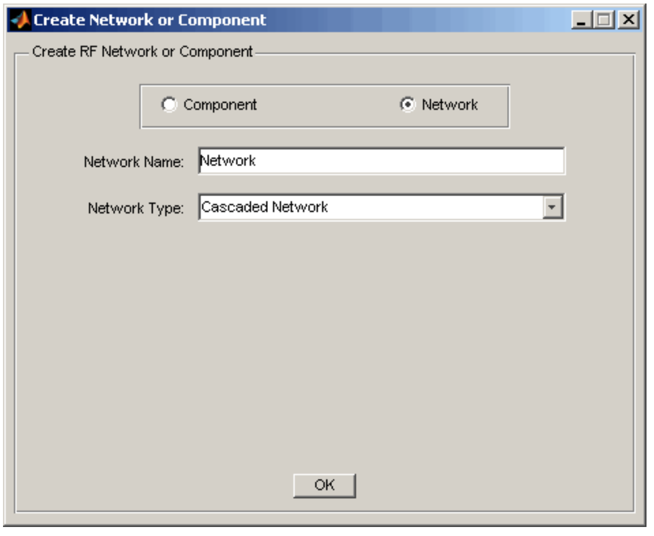
The Create Network or Component dialog box opens.

2 In the Create Network or Component dialog box:

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

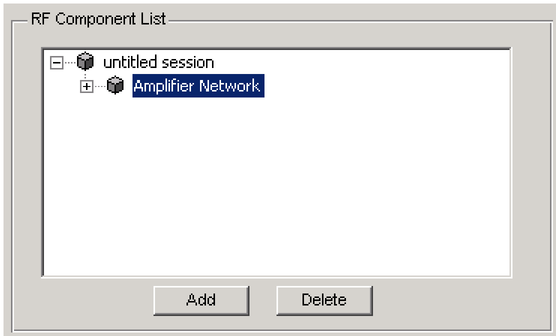
  This name is used to identify the network in the **RF Component List** pane.
- In the **Network Type** list, select `Cascaded Network`.

  A `Cascaded Network` means that when you add components to the network, RF Toolbox 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.



Populating the Amplifier Network

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

- "Transmission Line 1" on page 5-33
- "Amplifier" on page 5-34
- "Transmission Line 2" on page 5-36

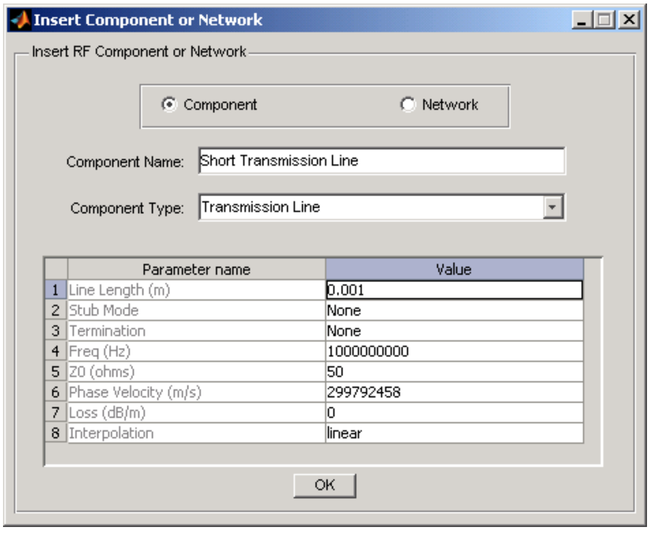
Transmission Line 1

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

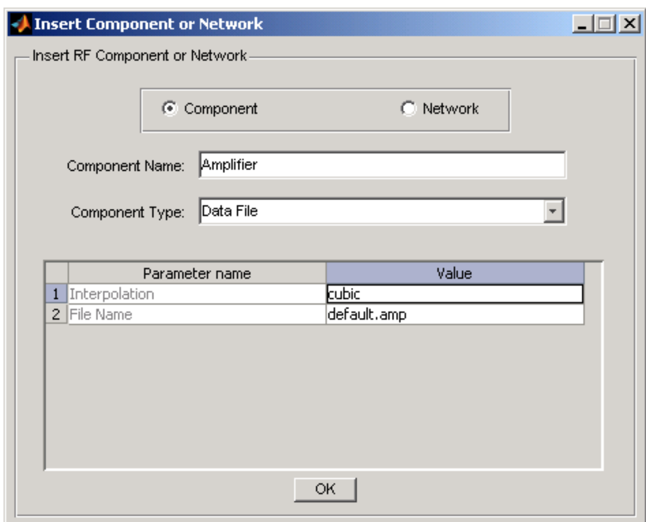
1 In the **Component Parameters** pane, click **Insert** to open the Insert Component or Network dialog box.

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 Toolbox 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.

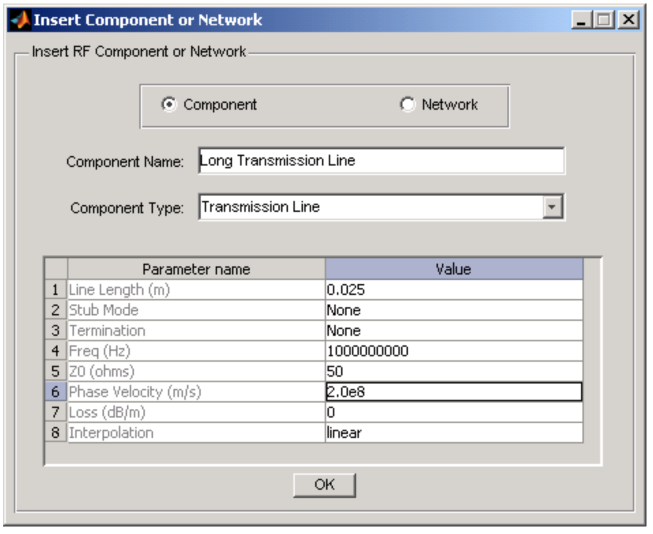
Transmission Line 2

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.

Simulating the Amplifier Network

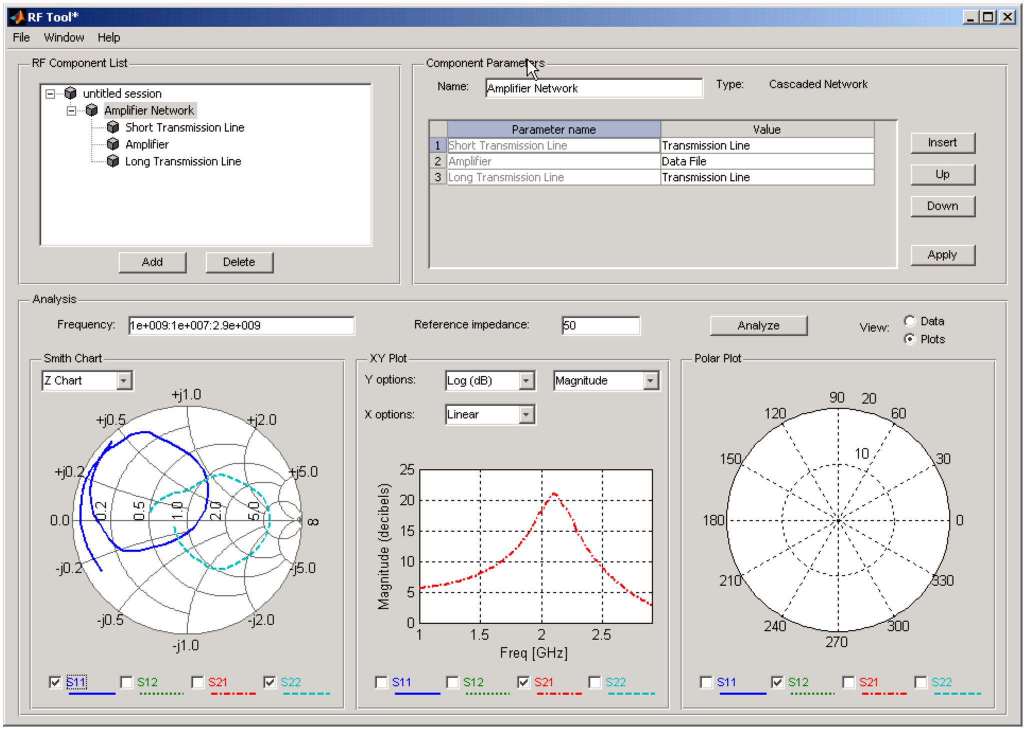
In this part of the example, you specify the range of frequencies over which to analyze the amplifier network and then run the analysis.

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 Smith, XY, and polar plots 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.

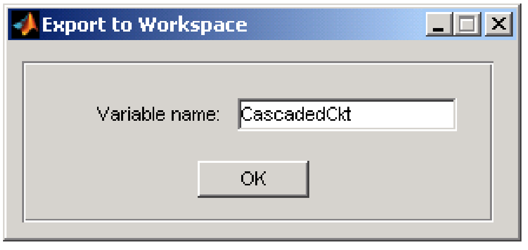
Exporting the Network to the Workspace

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**.

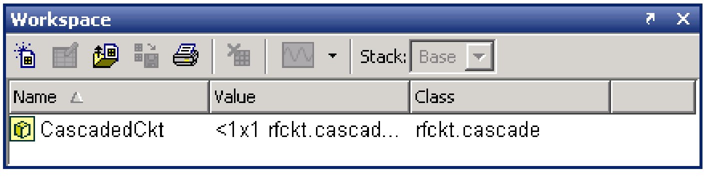
2 In the **Variable name** field, enter `CascadedCkt`.

   This name is the exported object's handle.



3 Click **OK**.

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



Objects — By Category

Circuit Objects

Components

Active Components

rfckt.amplifier	Model RF amplifier
rfckt.mixer	Model 2-port object representing RF mixer and its local oscillator

Ladder Filters

rfckt.lcbandpasspi	Model bandpass pi filter
rfckt.lcbandpasstee	Model bandpass tee filter
rfckt.lcbandstoppi	Model bandstop pi filter
rfckt.lcbandstoptee	Model bandstop tee filter
rfckt.lchighpasspi	Model highpass pi filter
rfckt.lchighpasstee	Model highpass tee filter
rfckt.lclowpasspi	Model lowpass pi filter
rfckt.lclowpasstee	Model lowpass tee filter

RLC Components

rfckt.seriesrlc	Model series RLC component
rfckt.shuntrlc	Model shunt RLC component

Transmission Lines

rfckt.coaxial	Model coaxial transmission line
rfckt.cpw	Model coplanar waveguide transmission line
rfckt.delay	Model delay line
rfckt.microstrip	Model microstrip transmission line
rfckt.parallelplate	Model parallel-plate transmission line
rfckt.rlcgline	Model RLCG transmission line
rfckt.twowire	Model two-wire transmission line
rfckt.txline	Model general transmission line

Black Box Elements

rfckt.datafile	Model component or network from file data
rfckt.passive	Model passive component or network

Networks

rfckt.cascade	Model cascaded network
rfckt.hybrid	Model hybrid connected network
rfckt.hybridg	Model inverse hybrid connected network
rfckt.parallel	Model parallel connected network
rfckt.series	Model series connected network

Data Objects

rfdata.data	Store result of circuit object analysis
rfdata.ip3	Store frequency-dependent, third-order intercept points
rfdata.mixerspur	Store data from intermodulation table
rfdata.network	Store frequency-dependent network parameters
rfdata.nf	Store frequency-dependent noise figure data for amplifiers or mixers
rfdata.noise	Store frequency-dependent spot noise data for amplifiers or mixers
rfdata.power	Store output power and phase information for amplifiers or mixers

Model Objects

rfmodel.rational	Rational function model

7

Objects — Alphabetical List

amplifier

Purpose

Model RF amplifier

Class Description

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

Constructor Summary

`rfckt.amplifier`	Construct `rfckt.amplifier` object

Property Summary

`AnalyzedResult`	Computed S-parameters, noise figure, and OIP3 values
`IntpType`	Interpolation method
`Name`	Object name
`NetworkData`	Network parameter information
`NoiseData`	Noise information
`NonlinearData`	Nonlinearity information
`nPort`	Number of ports

Method Summary

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

listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
read	Read RF data from file to new or existing circuit or data object
restore	Restore data to original frequencies
semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis
smith	Plot specified circuit object parameters on Smith chart
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]
```

See Also `rfckt.datafile`, `rfckt.mixer`, `rfckt.passive`, `rfdata.data`, `rfdata.ip3`, `rfdata.network`, `rfdata.nf`, `rfdata.noise`, `rfdata.power`

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

Properties

`AnalyzedResult`

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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, and OIP3 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:

  a Computing a scaling factor, which is equal to 3 divided by the IIP3 value, converted from decibels to linear units.

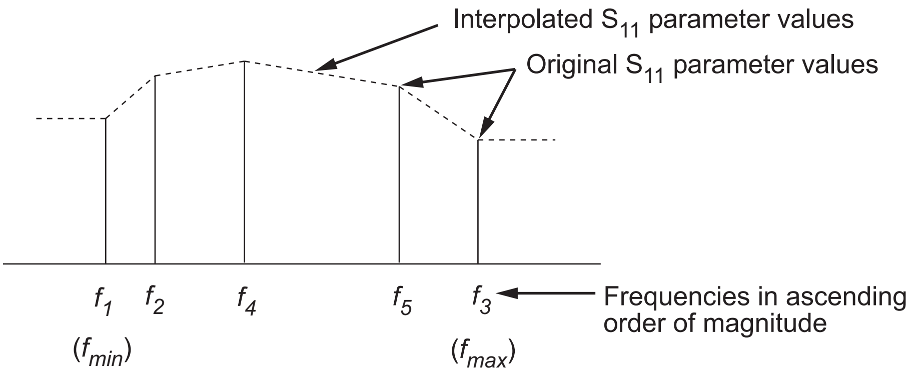
  b Applying the scaling factor.

  c Limiting the scaled input to a maximum value of 1 and applying an AM/AM conversion to the magnitude of the scaled signal according to the following function:

  $$F_{AM/AM}(u) = u - \frac{u}{3}$$

  where u is the magnitude of the scaled signal.
- 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]
            NF: [191x1 double]
            OIP3: [191x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
```

```
IntpType: 'Linear'
```

IntpType

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]
```

Name

Purpose

Object name

Values

`'Amplifier'`

Description

Read-only string that contains the name of the object.

Examples

```
amp = rfckt.amplifier;
amp.Name

ans =

              Amplifier
```

NetworkData

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
```

NoiseData

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]
```

NonlinearData

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]}
```

nPort

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
```

Constructor

rfckt.amplifier

Syntax

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

Description

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

`h =`
`rfckt.amplifier('Property1',value1,'Property2',value2,...)` returns a circuit object, `h`, based on the specified properties. Properties 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.

Purpose

Model cascaded network

Class Description

Use the `cascade` class to represent cascaded networks of RF objects that are characterized by the components that make up the network.

Constructor Summary

`rfckt.cascade`	Construct `rfckt.cascade` object

Property Summary

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

Method Summary

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

polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis
smith	Plot specified circuit object parameters on Smith chart
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}
```

See Also

`rfckt.hybrid`, `rfckt.hybridg`, `rfckt.parallel`, `rfckt.series`

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

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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 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 OIP3 of the nth element.

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

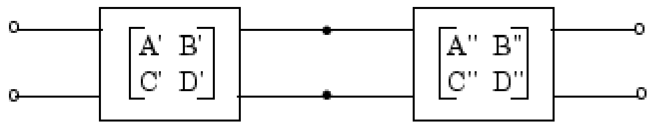
$$NF = NF_1 + \frac{NF_2 - 1}{G_1} + \frac{NF_3 - 1}{G_1 \cdot G_2} + \ldots + \frac{NF_N - 1}{G_1 \cdot G_2 \cdot \ldots \cdot G_{N-1}}$$

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

- 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`.

Examples

```
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

ans =

            Name: 'Data object'
            Freq: [101x1 double]
    S_Parameters: [2x2x101 double]
```

```
          NF: [101x1 double]
        OIP3: [101x1 double]
          Z0: 50
          ZS: 50
          ZL: 50
    IntpType: 'Linear'
```

Ckts

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]
```

Name

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
```

nPort

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
```

Constructor

rfckt.cascade

Syntax

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

Description

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

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

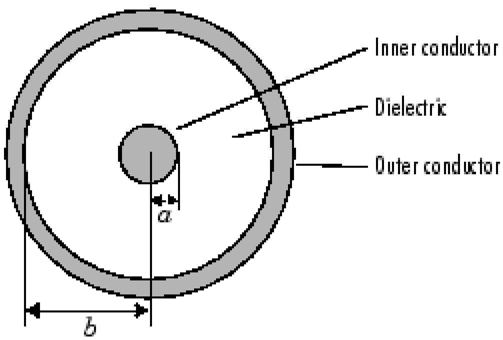
Purpose

Model coaxial transmission line

Class 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.



Constructor Summary

`rfckt.coaxial`	Construct `rfckt.coaxial` object

Property Summary

`AnalyzedResult`	Computed S-parameters, noise figure, and OIP3 values
`EpsilonR`	Relative permittivity of dielectric
`InnerRadius`	Inner conductor radius
`LineLength`	Transmission line length
`Loss`	Transmission line loss
`MuR`	Relative permeability of dielectric

`Name`	Object name
`nPort`	Number of ports
`OuterRadius`	Outer conductor radius
`PV`	Phase velocity
`SigmaCond`	Conductor conductivity
`SigmaDiel`	Dielectric conductivity
`StubMode`	Type of stub
`Termination`	Stub transmission line termination
`Z0`	Characteristic impedance

Method Summary

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
getz0	Characteristic impedance of transmission line object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides

polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis
smith	Plot specified circuit object parameters on Smith chart
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: 'None'
             Termination: 'None'
             OuterRadius: 0.0045
             InnerRadius: 7.2500e-004
             MuR: 1
             EpsilonR: 2.3000
             SigmaCond: Inf
             SigmaDiel: 0
```

See Also

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

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

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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 calculates the S-parameters for the specified frequencies, based on the physical length of the transmission line, D, and the complex propagation constant, k.

  $$S_{11} = 0$$

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

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

  $$S_{22} = 0$$

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

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

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

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

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

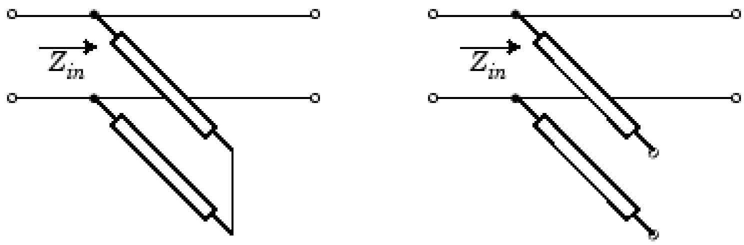
$$G = \frac{2\pi\sigma_{diel}}{\ln(b/a)}$$

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

In these equations, a is the radius of the inner conductor and b is the radius of the outer conductor. σ_{cond} is the conductivity in the conductor and σ_{diel} is the conductivity in the dielectric. μ is the relative permeability of the dielectric, ε is its permittivity as derived from the `EpsilonR` property, and skin depth δ is calculated as $1/\sqrt{\pi f\mu\sigma_{cond}}$.

- 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.

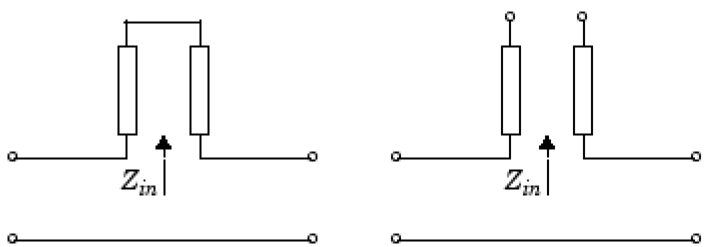
  When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown in the following figure.



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:

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

Examples

```
tx1 = rfckt.coaxial;
analyze(tx1,[1e9,2e9,3e9]);
tx1.AnalyzedResult

ans =

              Name: 'Data object'
              Freq: [3x1 double]
              S_Parameters: [2x2x3 double]
              NF: [3x1 double]
              OIP3: [3x1 double]
              Z0: 50
              ZS: 50
              ZL: 50
              IntpType: 'Linear'
```

EpsilonR

Purpose

Relative permittivity of dielectric

Values

Scalar

Description

The ratio of the permittivity of the dielectric to the permittivity in free space ε_0. The default is 2.3.

Examples

```
tx1=rfckt.coaxial;
tx1.EpsilonR=2.7;
```

InnerRadius

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;
```

LineLength

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;
```

Loss

Purpose

Transmission line loss

Values

Vector

Description

Read-only vector of line loss values, in decibels per meter, computed by the `analyze` method. The values correspond to the frequencies at which you analyze the transmission line. Line loss is the reduction in strength of the signal as it travels over the transmission line. This property is empty by default.

Examples

```
tx1 = rfckt.coaxial;
tx1.Loss

ans =

     []
```

MuR

Purpose

Relative permeability of dielectric

Values

Scalar

Description

The ratio of the permeability of the dielectric to the permeability in free space μ_0. The default is `1`.

Examples

```
tx1=rfckt.coaxial;
tx1.MuR=0.8;
```

Name

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
```

nPort

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
```

OuterRadius

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;
```

PV

Purpose

Phase velocity

Values

Vector

Description

Read-only vector of phase velocity values, in meters per second, computed by the `analyze` method. The values correspond to the frequencies at which you analyze the transmission line. The phase velocity is the propagation velocity of a uniform plane wave on the transmission line. This property is empty by default.

Examples

```
tx1 = rfckt.coaxial;
tx1.PV

ans =
```

`[]`

SigmaCond

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;
```

SigmaDiel

Purpose

Dielectric conductivity

Values

Scalar

Description

Conductivity, in Siemens per meter (S/m), of the dielectric. The default is `0`.

Examples

```
tx1=rfckt.coaxial;
tx1.SigmaDiel=0.002;
```

StubMode

Purpose

Type of stub

Values

`'None'` (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';
```

Termination

Purpose

Stub transmission line termination

Values

`'None'` (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 `'None'` when `StubMode` is `'None'`.

Examples

```
tx1 = rfckt.coaxial;
tx1.StubMode = 'Series';
tx1.Termination = 'Short';
```

Z0

Purpose

Characteristic impedance

Values

Vector

Description

Read-only vector of characteristic impedance values, in ohms, computed by the `analyze` method. The values correspond to the frequencies at which you analyze the transmission line. This property is empty by default.

Examples

```
tx1 = rfckt.coaxial;
tx1.Z0

ans =

     []
```

Constructor

rfckt.coaxial

Syntax

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

Description

`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.

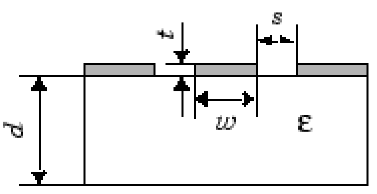
Purpose

Model coplanar waveguide transmission line

Class 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).



Constructor Summary

`rfckt.cpw`	Construct `rfckt.cpw` object

Property Summary

`AnalyzedResult`	Computed S-parameters, noise figure, and OIP3 values
`ConductorWidth`	Conductor width
`EpsilonR`	Relative permittivity of dielectric
`Height`	Dielectric thickness
`LineLength`	Transmission line length
`Loss`	Transmission line loss
`LossTangent`	Tangent of loss angle
`Name`	Object name
`nPort`	Number of ports
`PV`	Phase velocity

`SigmaCond`	Conductor conductivity
`SlotWidth`	Width of slot
`StubMode`	Type of stub
`Termination`	Stub transmission line termination
`Thickness`	Conductor thickness
`Z0`	Characteristic impedance

Method Summary

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
getz0	Characteristic impedance of transmission line object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates

semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis
smith	Plot specified circuit object parameters on Smith chart
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: 'None'
                    Termination: 'None'
                    ConductorWidth: 6.0000e-004
                    SlotWidth: 2.0000e-004
                    Height: 6.3500e-004
                    Thickness: 7.5000e-009
                    EpsilonR: 9.8000
                    SigmaCond: Inf
                    LossTangent: 0
```

See Also

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

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

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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 calculates the S-parameters for the specified frequencies, based on the physical length of the transmission line, D, and the complex propagation constant, k.

  $$S_{11} = 0$$
  $$S_{12} = e^{-kD}$$
  $$S_{21} = e^{-kD}$$
  $$S_{22} = 0$$

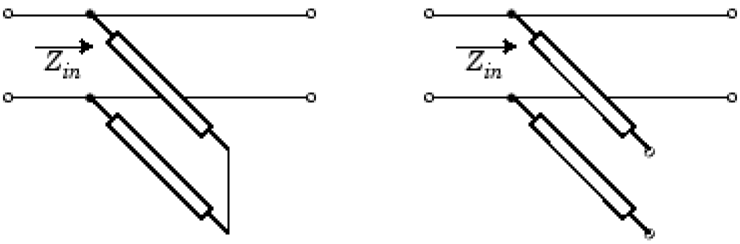
  $k = \alpha_a + i\beta$, where α_a is the attenuation coefficient and β is the wave number. The attenuation coefficient α_a is related to the loss, α, by

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

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

- 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.

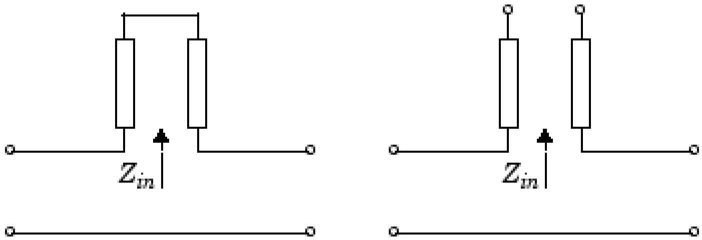
  When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown in the following figure.



  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:

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

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]
            OIP3: [3x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Linear'
```

ConductorWidth

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;
```

EpsilonR

Purpose

Relative permittivity of dielectric

Values

Scalar

Description

The ratio of the permittivity of the dielectric to the permittivity in free space ε_0. The default is `9.8`.

Examples

```
tx1=rfckt.cpw;
tx1.EpsilonR=2.7;
```

Height

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;
```

LineLength

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;
```

Loss

Purpose

Transmission line loss

Values

Vector

Description

Read-only vector of line loss values, in decibels per meter, computed by the `analyze` method. The values correspond to the frequencies at which you analyze the transmission line. Line loss is the reduction in strength of the signal as it travels over the transmission line. This property is empty by default.

Examples

```
tx1 = rfckt.cpw;
tx1.Loss

ans =

     []
```

LossTangent

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
```

Name

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
```

nPort

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
```

PV

Purpose

Phase velocity

Values

Vector

Description

Read-only vector of phase velocity values, in meters per second, computed by the `analyze` method. The values correspond to the frequencies at which you analyze the transmission line. The phase velocity is the propagation velocity of a uniform plane wave on the transmission line. This property is empty by default.

Examples

```
tx1 = rfckt.cpw;
tx1.PV

ans =

     []
```

SigmaCond

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;
```

SlotWidth

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;
```

StubMode

Purpose

Type of stub

Values

`'None'` (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';
```

Termination

Purpose

Stub transmission line termination

Values

'None' (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 'None' when StubMode is 'None'.

Examples

```
tx1 = rfckt.cpw;
tx1.StubMode = 'Series';
tx1.Termination = 'Short';
```

Thickness

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;
```

Z0

Purpose

Characteristic impedance

Values

Vector

Description

Read-only vector of characteristic impedance values, in ohms, computed by the `analyze` method. The values correspond to the frequencies at which you analyze the transmission line. This property is empty by default.

Examples

```
tx1 = rfckt.cpw;
tx1.Z0

ans =

     []
```

Constructor

rfckt.cpw

Syntax

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

Description

`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.

datafile

Purpose

Model component or network from file data

Class Description

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

Constructor Summary

`rfckt.datafile`	Construct `rfckt.datafile` object

Property Summary

`AnalyzedResult`	Computed S-parameters, noise figure, and OIP3 values
`File`	File containing circuit data
`IntpType`	Interpolation method
`Name`	Object name
`nPort`	Number of ports

Method Summary

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

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

See Also

rfckt.amplifier, rfckt.mixer, rfckt.passive

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

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

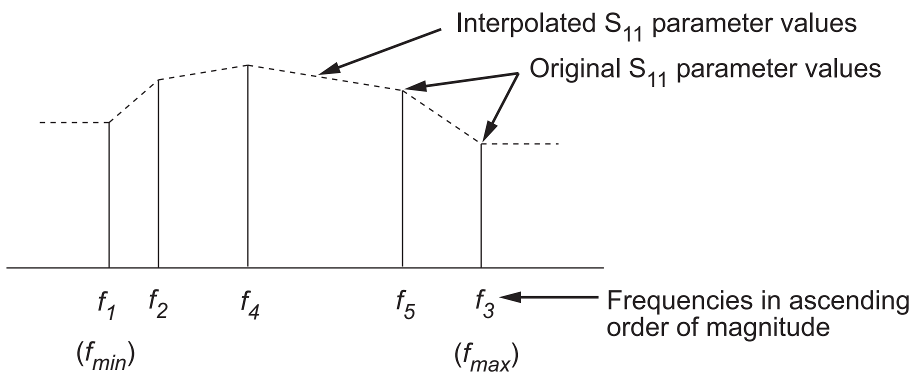
Values

rfdata.data object

Description

Handle to an rfdata.data object that contains the S-parameters, noise figure, and OIP3 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, and OIP3 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.

Examples

```
data = rfckt.datafile;
data.AnalyzedResult

ans =

            Name: 'Data object'
            Freq: [202x1 double]
            S_Parameters: [2x2x202 double]
            NF: [202x1 double]
            OIP3: [202x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
```

```
              IntpType: 'Linear'
```

File

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'
```

IntpType

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';
```

Name

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
```

nPort

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
```

Constructor

rfckt.datafile

Syntax

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

Description

`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.

delay

Purpose

Model delay line

Class Description

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

Constructor Summary

`rfckt.delay`	Construct `rfckt.delay` object

Property Summary

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

Method Summary

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
getz0	Characteristic impedance of transmission line object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale

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

See Also

`rfckt.rlcgline`, `rfckt.txline`

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

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

rfdata.data object

Description

Handle to an rfdata.data object that contains the S-parameters, noise figure, and OIP3 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.

$$S_{11} = 0$$

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

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

$$S_{22} = 0$$

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

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

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

$$\beta = 2\pi fD$$

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

Examples

```
del = rfckt.delay;
analyze(del,[1e9,2e9,3e9]);
del.AnalyzedResult

ans =

             Name: 'Data object'
             Freq: [3x1 double]
             S_Parameters: [2x2x3 double]
             NF: [3x1 double]
             OIP3: [3x1 double]
             Z0: 50
             ZS: 50
             ZL: 50
             IntpType: 'Linear'
```

Loss

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;
```

Name

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
```

nPort

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
```

TimeDelay

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;
```

Z0

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;
```

Constructor

rfckt.delay

Syntax

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

Description

`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.

Purpose

Model hybrid connected network

Class 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.

Constructor Summary

`rfckt.hybrid`	Construct `rfckt.hybrid` object

Property Summary

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

Method Summary

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane

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

See Also

`rfckt.cascade`, `rfckt.hybridg`, `rfckt.parallel`, `rfckt.series`

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

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

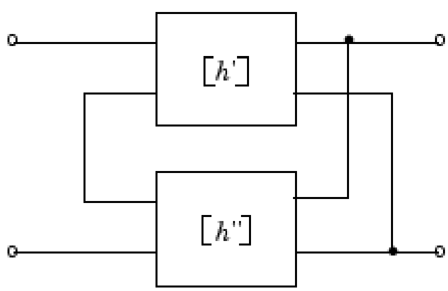
`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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 $[h'] = \begin{bmatrix} h_{11}' & h_{12}' \\ h_{21}' & h_{22}' \end{bmatrix}$ and $[h''] = \begin{bmatrix} h_{11}'' & h_{12}'' \\ h_{21}'' & h_{22}'' \end{bmatrix}$

- 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.

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]
              NF: [101x1 double]
            OIP3: [101x1 double]
              Z0: 50
              ZS: 50
              ZL: 50
        IntpType: 'Linear'
```

Ckts

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]
```

Name

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
```

nPort

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
```

Constructor

rfckt.hybrid

Syntax

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

Description

`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.

Purpose

Model inverse hybrid connected network

Class 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.

Constructor Summary

`rfckt.hybridg`	Construct `rfckt.hybridg` object

Property Summary

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

Method Summary

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane

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

See Also

`rfckt.cascade`, `rfckt.hybrid`, `rfckt.parallel`, `rfckt.series`

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

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

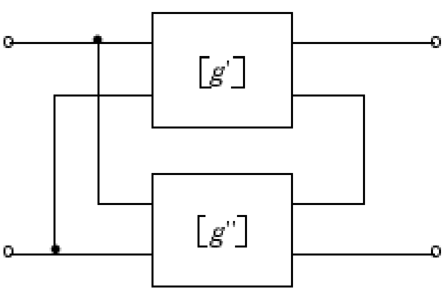
`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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 $[g'] = \begin{bmatrix} g_{11}' & g_{12}' \\ g_{21}' & g_{22}' \end{bmatrix}$ and $[g''] = \begin{bmatrix} g_{11}'' & g_{12}'' \\ g_{21}'' & g_{22}'' \end{bmatrix}$

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`.

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]
              NF: [101x1 double]
            OIP3: [101x1 double]
              Z0: 50
              ZS: 50
              ZL: 50
        IntpType: 'Linear'
```

Ckts

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]
```

Name

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
```

nPort

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
```

Constructor

rfckt.hybridg

Syntax

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

Description

`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.

lcbandpasspi

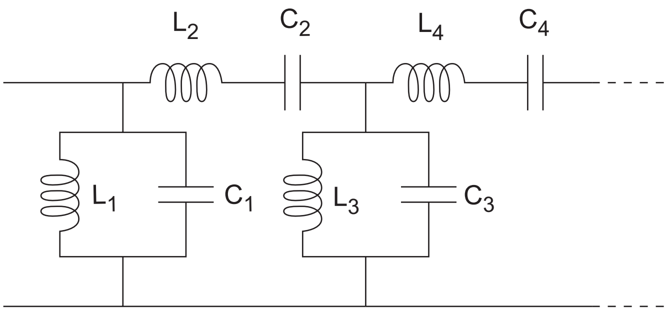
Purpose

Model bandpass pi filter

Class 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.

Constructor Summary

`rfckt.lcbandpasspi`	Construct `rfckt.lcbandpasspi` object

Property Summary

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

Method Summary

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis
smith	Plot specified circuit object parameters on Smith chart
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]
```

See Also

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

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.

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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]
            NF: [3x1 double]
            OIP3: [3x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Linear'
```

C

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;
```

L

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;
```

Name

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
```

nPort

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
```

Constructor

rfckt.lcbandpasspi

Syntax

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

Description

`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.

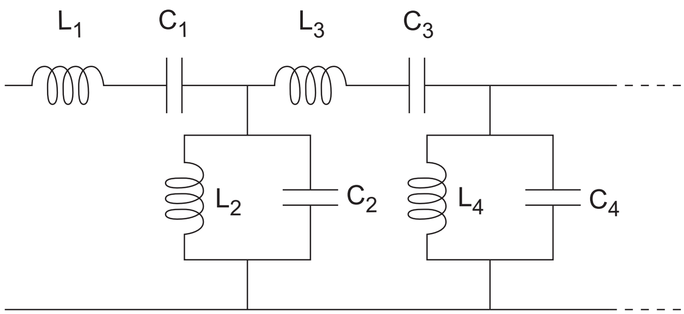
Purpose

Model bandpass tee filter

Class 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.

Constructor Summary

`rfckt.lcbandpasstee`	Construct `rfckt.lcbandpasstee` object

Property Summary

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

Method Summary

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis
smith	Plot specified circuit object parameters on Smith chart
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]
```

See Also

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

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.

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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]
            NF: [3x1 double]
            OIP3: [3x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Linear'
```

C

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;
```

L

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;
```

Name

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
```

nPort

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
```

Constructor

rfckt.lcbandpasstee

Syntax

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

Description

`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.

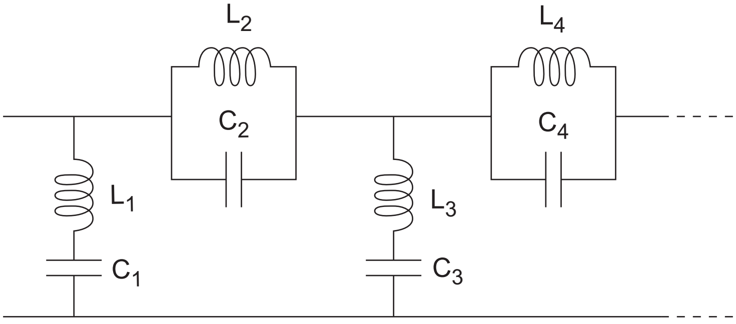
lcbandstoppi

Purpose

Model bandstop pi filter

Class 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.

Constructor Summary

`rfckt.lcbandstoppi`	Construct `rfckt.lcbandstoppi` object

Property Summary

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

Method Summary

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis
smith	Plot specified circuit object parameters on Smith chart
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]
```

See Also

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

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.

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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]
            NF: [3x1 double]
            OIP3: [3x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Linear'
```

C

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;
```

L

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;
```

Name

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
```

nPort

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
```

Constructor

rfckt.lcbandstoppi

Syntax

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

Description

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

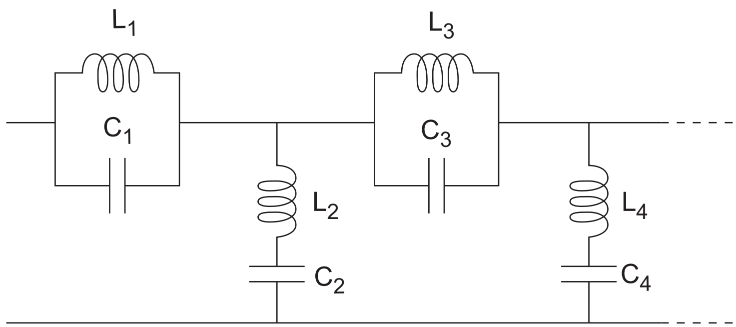
`h = rfckt.lcbandstoppi('Property1',value1,'Property2',value2,...)` returns an LC bandstop pi network object, `h`, based on the specified properties. Properties that you do not specify retain their default values.

Purpose

Model bandstop tee filter

Class Description

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

Constructor Summary

`rfckt.lcbandstoptee`	Construct `rfckt.lcbandstoptee` object

Property Summary

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

Method Summary

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis
smith	Plot specified circuit object parameters on Smith chart
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]
```

See Also

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

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.

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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]
            NF: [3x1 double]
            OIP3: [3x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Linear'
```

C

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;
```

L

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;
```

Name

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
```

nPort

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
```

Constructor

rfckt.lcbandstoptee

Syntax

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

Description

`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.

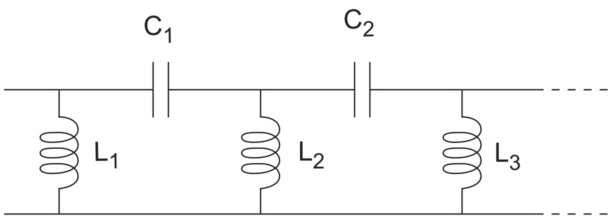
lchighpasspi

Purpose Model highpass pi filter

Class 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.

Constructor Summary

`rfckt.lchighpasspi`	Construct `rfckt.lchighpasspi` object

Property Summary

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

Method Summary

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis
smith	Plot specified circuit object parameters on Smith chart
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]
```

See Also

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

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.

Properties

`AnalyzedResult`

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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]
            NF: [3x1 double]
            OIP3: [3x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Linear'
```

C

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;
```

L

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;
```

Name

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
```

nPort

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
```

Constructor

rfckt.lchighpasspi

Syntax

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

Description
`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.

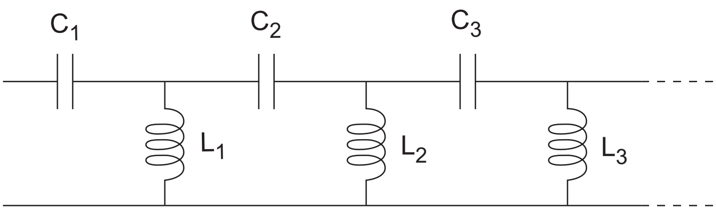
Purpose

Model highpass tee filter

Class 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.

Constructor Summary

`rfckt.lchighpasstee`	Construct `rfckt.lchighpasstee` object

Property Summary

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

Method Summary

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis
smith	Plot specified circuit object parameters on Smith chart
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]
```

See Also

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

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.

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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]
            NF: [3x1 double]
            OIP3: [3x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Linear'
```

C

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;
```

L

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;
```

Name

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
```

nPort

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
```

Constructor

rfckt.lchighpasstee

Syntax

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

Description

`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.

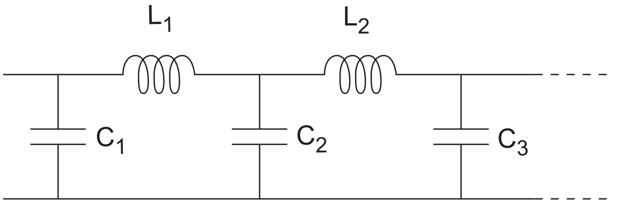
Purpose

Model lowpass pi filter

Class 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.

Constructor Summary

`rfckt.lclowpasspi`	Construct `rfckt.lclowpasspi` object

Property Summary

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

Method Summary

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis
smith	Plot specified circuit object parameters on Smith chart
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]
```

See Also

`rfckt.lcbandpasspi`, `rfckt.lcbandpasstee`, `rfckt.lcbandstoppi`, `rfckt.lcbandstoptee`, `rfckt.lchighpasspi`, `rfckt.lchighpasstee`, `rfckt.lclowpasstee`

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.

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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]
            NF: [3x1 double]
            OIP3: [3x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Linear'
```

C

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;
```

L

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;
```

Name

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
```

nPort

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
```

Constructor

rfckt.lclowpasspi

Syntax

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

Description

`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.

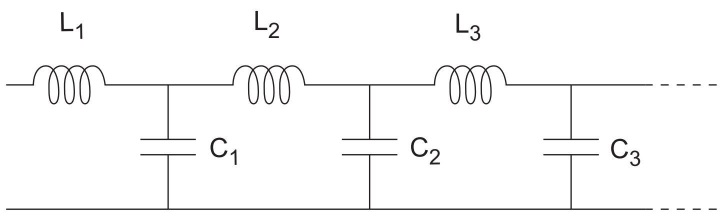
Purpose

Model lowpass tee filter

Class 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.

Constructor Summary

`rfckt.lclowpasstee`	Construct `rfckt.lclowpasstee` object

Property Summary

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

Method Summary

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis
smith	Plot specified circuit object parameters on Smith chart
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]
```

See Also

`rfckt.lcbandpasspi`, `rfckt.lcbandpasstee`, `rfckt.lcbandstoppi`, `rfckt.lcbandstoptee`, `rfckt.lchighpasspi`, `rfckt.lchighpasstee`, `rfckt.lclowpasspi`

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.

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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]
            NF: [3x1 double]
            OIP3: [3x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Linear'
```

C

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;
```

L

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;
```

Name

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

nPort

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
```

Constructor

rfckt.lclowpasstee

Syntax

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

Description

`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.

microstrip

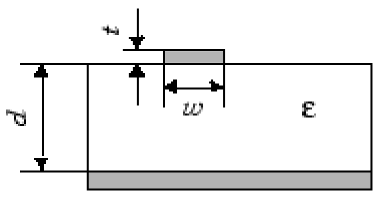
Purpose

Model microstrip transmission line

Class 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).



Constructor Summary

`rfckt.microstrip`	Construct `rfckt.microstrip` object

Property Summary

`AnalyzedResult`	Computed S-parameters, noise figure, and OIP3 values
`EpsilonR`	Relative permittivity of dielectric
`Height`	Dielectric thickness
`LineLength`	Microstrip line length
`Loss`	Transmission line loss
`LossTangent`	Tangent of loss angle
`Name`	Object name
`nPort`	Number of ports
`PV`	Phase velocity

`SigmaCond`	Conductor conductivity
`StubMode`	Type of stub
`Termination`	Stub transmission line termination
`Thickness`	Microstrip thickness
`Width`	Parallel-plate width
`Z0`	Characteristic impedance

Method Summary

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
getz0	Characteristic impedance of transmission line object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates

semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis
smith	Plot specified circuit object parameters on Smith chart
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: 'None'
            Termination: 'None'
            Width: 6.0000e-004
            Height: 6.3500e-004
            Thickness: 7.5000e-009
            EpsilonR: 9.8000
            SigmaCond: Inf
            LossTangent: 0
```

See Also

`rfckt.coaxial`, `rfckt.cpw`, `rfckt.parallelplate`, `rfckt.rlcgline`, `rfckt.twowire`, `rfckt.txline`

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

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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 calculates the S-parameters for the specified frequencies, based on the physical length of the transmission line, *D*, and the complex propagation constant, *k*.

  $$S_{11} = 0$$

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

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

  $$S_{22} = 0$$

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

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

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

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

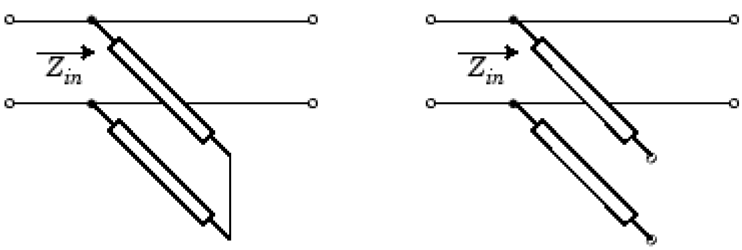
$$\beta = \frac{2\pi f}{V_p}$$

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

The phase velocity V_P is also known as the *wave propagation velocity*.

- 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.

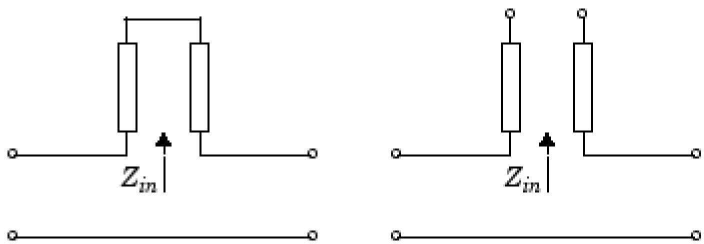
  When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown in the following figure.



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:

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

Examples

```
tx1 = rfckt.microstrip;
analyze(tx1,[1e9,2e9,3e9]);
tx1.AnalyzedResult

ans =

            Name: 'Data object'
            Freq: [3x1 double]
            S_Parameters: [2x2x3 double]
```

```
NF: [3x1 double]
OIP3: [3x1 double]
Z0: 50
ZS: 50
ZL: 50
IntpType: 'Linear'
```

EpsilonR

Purpose

Relative permittivity of dielectric

Values

Scalar

Description

The ratio of the permittivity of the dielectric to the permittivity in free space ε_0. The default is 9.8.

Examples

```
tx1=rfckt.microstrip;
tx1.EpsilonR=2.7;
```

Height

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;
```

LineLength

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;
```

Loss

Purpose

Transmission line loss

Values

Vector

Description

Read-only vector of line loss values, in decibels per meter, computed by the `analyze` method. The values correspond to the frequencies at which you analyze the transmission line. Line loss is the reduction in strength of the signal as it travels over the transmission line. This property is empty by default.

Examples

```
tx1 = rfckt.microstrip;
tx1.Loss

ans =

     []
```

LossTangent

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
```

Name

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
```

nPort

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
```

PV

Purpose

Phase velocity

Values

Vector

Description

Read-only vector of phase velocity values, in meters per second, computed by the `analyze` method. The values correspond to the frequencies at which you analyze the transmission line. The phase velocity is the propagation velocity of a uniform plane wave on the transmission line. This property is empty by default.

Examples

```
tx1 = rfckt.microstrip;
tx1.PV

ans =

     []
```

SigmaCond

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;
```

StubMode

Purpose

Type of stub

Values

'None' (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';
```

Termination

Purpose

Stub transmission line termination

Values

'None' (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 'None' when StubMode is 'None'.

Examples

```
tx1 = rfckt.microstrip;
tx1.StubMode = 'Series';
tx1.Termination = 'Short';
```

Thickness

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;
```

Width

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;
```

Z0

Purpose

Characteristic impedance

Values

Vector

Description

Read-only vector of characteristic impedance values, in ohms, computed by the `analyze` method. The values correspond to the frequencies at which you analyze the transmission line. This property is empty by default.

Examples

```
tx1 = rfckt.microstrip;
tx1.Z0

ans =

     []
```

Constructor

rfckt.microstrip

Syntax

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

Description

`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.

Purpose

Model 2-port object representing RF mixer and its local oscillator

Class 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.

Constructor Summary

`rfckt.mixer`	Construct `rfckt.mixer` object

Property Summary

`AnalyzedResult`	Computed S-parameters, noise figure, and OIP3 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

Method Summary

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

extract	Extract array of network parameters from data object
getop	Display operating conditions
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
read	Read RF data from file to new or existing circuit or data object
restore	Restore data to original frequencies
semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis
smith	Plot specified circuit object parameters on Smith chart
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
                MixerType: 'downconverter'
                FLO: 1.0000e+009
                FreqOffset: []
                PhaseNoiseLevel: []
```

See Also

`rfckt.amplifier`, `rfckt.datafile`, `rfckt.passive`, `rfdata.data`, `rfdata.ip3`, `rfdata.mixerspur`, `rfdata.network`, `rfdata.nf`, `rfdata.noise`, `rfdata.power`

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

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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, and OIP3 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:

  a Interpolating the specified phase noise amplitude to determine the phase noise values at the modeling frequencies.

  b 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:

  a Computing a scaling factor, which is equal to 3 divided by the IIP3 value, converted from decibels to linear units.

  b Applying the scaling factor.

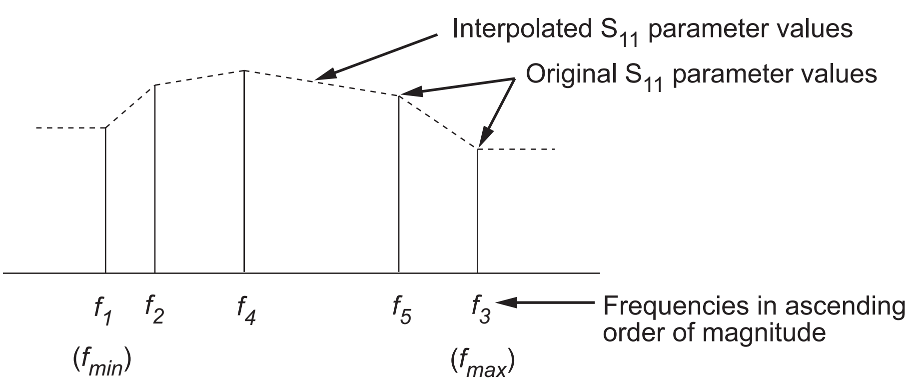
c Limiting the scaled input to a maximum value of 1 and applies an AM/AM conversion to the magnitude of the scaled signal according to the following function

$$F_{AM/AM}(u) = u - \frac{u}{3}$$

where u is the magnitude of the scaled signal.

- 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 Blockset 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]
            NF: [191x1 double]
            OIP3: [191x1 double]
            Z0: 50
```

```
ZS: 50
ZL: 50
IntpType: 'Linear'
```

FLO

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;
```

FreqOffset

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];
```

IntpType

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: []
```

MixerSpurData

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: []
```

MixerType

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';
```

Name

Purpose

Object name

Values

'Mixer'

Description

Read-only string that contains the name of the object.

Examples

```
mix1 = rfckt.mixer;
mix1.Name

ans =

           Mixer
```

NetworkData

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

```
mix1 = rfckt.mixer;
mix1.NetworkData

ans =

    Name: 'Network parameters'
    Type: 'S_PARAMETERS'
    Freq: [191x1 double]
    Data: [2x2x191 double]
    Z0: 50
```

NoiseData

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]
```

NonlinearData

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
```

nPort

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
```

PhaseNoiseLevel

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];
```

Constructor

rfckt.mixer

Syntax

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

Description

`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.

Purpose

Model parallel connected network

Class 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.

Constructor Summary

`rfckt.parallel`	Construct `rfckt.parallel` object

Property Summary

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

Method Summary

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane

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

See Also

`rfckt.cascade`, `rfckt.hybrid`, `rfckt.hybridg`, `rfckt.series`

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

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

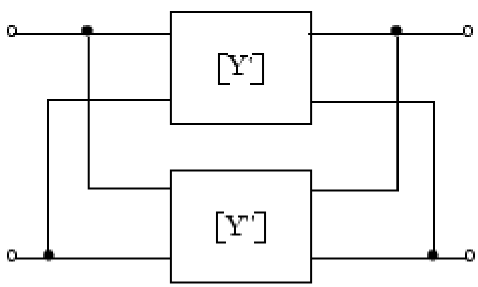
`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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 $[Y'] = \begin{bmatrix} Y_{11}' & Y_{12}' \\ Y_{21}' & Y_{22}' \end{bmatrix}$ and $[Y''] = \begin{bmatrix} Y_{11}'' & Y_{12}'' \\ Y_{21}'' & Y_{22}'' \end{bmatrix}$

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.

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]
              NF: [101x1 double]
            OIP3: [101x1 double]
              Z0: 50
              ZS: 50
              ZL: 50
        IntpType: 'Linear'
```

Ckts

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]
```

Name

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
```

nPort

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
```

Constructor

rfckt.parallel

Syntax

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

Description

`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.

parallelplate

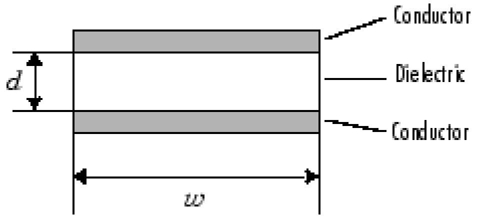
Purpose

Model parallel-plate transmission line

Class 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.



Constructor Summary

`rfckt.parallelplate`	Construct `rfckt.parallelplate` object

Property Summary

`AnalyzedResult`	Computed S-parameters, noise figure, and OIP3 values
`EpsilonR`	Relative permittivity of dielectric
`LineLength`	Parallel-plate line length
`Loss`	Transmission line loss
`MuR`	Relative permeability of dielectric
`Name`	Object name
`nPort`	Number of ports
`PV`	Phase velocity

`Separation`	Distance between plates
`SigmaCond`	Conductor conductivity
`SigmaDiel`	Dielectric conductivity
`StubMode`	Type of stub
`Termination`	Stub transmission line termination
`Width`	Transmission line width
`Z0`	Characteristic impedance

Method Summary

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
getz0	Characteristic impedance of transmission line object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates

semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis
smith	Plot specified circuit object parameters on Smith chart
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: 'None'
           Termination: 'None'
           Width: 0.0050
           Separation: 1.0000e-003
           MuR: 1
           EpsilonR: 2.3000
           SigmaCond: Inf
           SigmaDiel: 0
```

See Also

`rfckt.coaxial`, `rfckt.cpw`, `rfckt.microstrip`, `rfckt.rlcgline`, `rfckt.twowire`, `rfckt.txline`

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

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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 calculates the S-parameters for the specified frequencies, based on the physical length of the transmission line, *D*, and the complex propagation constant, *k*.

  $$S_{11} = 0$$

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

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

  $$S_{22} = 0$$

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

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

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

$$R = \frac{2}{w\sigma_{cond}\delta}$$

$$L = \mu\frac{d}{w}$$

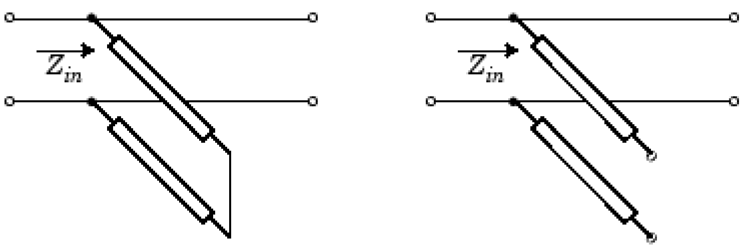
$$G = \sigma_{diel}\frac{w}{d}$$

$$C = \varepsilon\frac{w}{d}$$

In these equations, w is the plate width, and d is the plate separation. σ_{cond} is the conductivity in the conductor and σ_{diel} is the conductivity in the dielectric. μ is the relative permeability of the dielectric, ε is its permittivity as derived from the `EpsilonR` property, and skin depth δ is calculated as $1/\sqrt{\pi f\mu\sigma_{cond}}$.

- 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.

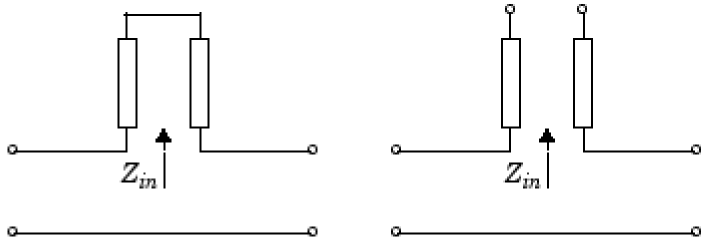
  When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown in the following figure.



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:

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

Examples

```
tx1 = rfckt.parallelplate;
analyze(tx1,[1e9,2e9,3e9]);
tx1.AnalyzedResult

ans =
```

```
Name: 'Data object'
Freq: [3x1 double]
S_Parameters: [2x2x3 double]
NF: [3x1 double]
OIP3: [3x1 double]
Z0: 50
ZS: 50
ZL: 50
IntpType: 'Linear'
```

EpsilonR

Purpose

Relative permittivity of dielectric

Values

Scalar

Description

The ratio of the permittivity of the dielectric to the permittivity in free space ε_0. The default is `2.3`.

Examples

```
tx1=rfckt.parallelplate;
tx1.EpsilonR=2.7;
```

LineLength

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;
```

Loss

Purpose

Transmission line loss

Values

Vector

Description

Read-only vector of line loss values, in decibels per meter, computed by the `analyze` method. The values correspond to the frequencies at which you analyze the transmission line. Line loss is the reduction in strength of the signal as it travels over the transmission line. This property is empty by default.

Examples

```
tx1 = rfckt.parallelplate;
tx1.Loss

ans =

     []
```

MuR

Purpose

Relative permeability of dielectric

Values

Scalar

Description

The ratio of the permeability of the dielectric to the permeability in free space μ_0. The default is 1.

Examples

```
tx1=rfckt.parallelplate;
tx1.MuR=0.8;
```

Name

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
```

nPort

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
```

PV

Purpose

Phase velocity

Values

Vector

Description

Read-only vector of phase velocity values, in meters per second, computed by the `analyze` method. The values correspond to the frequencies at which you analyze the transmission line. The phase velocity is the propagation velocity of a uniform plane wave on the transmission line. This property is empty by default.

Examples

```
tx1 = rfckt.parallelplate;
tx1.PV
```

```
ans =

     []
```

Separation

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;
```

SigmaCond

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;
```

SigmaDiel

Purpose

Dielectric conductivity

Values

Scalar

Description

Conductivity, in Siemens per meter (S/m), of the dielectric. The default is `0`.

Examples

```
tx1=rfckt.parallelplate;
tx1.SigmaDiel=0.002;
```

StubMode

Purpose

Type of stub

Values

`'None'` (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';
```

Termination

Purpose

Stub transmission line termination

Values

'None' (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 'None' when StubMode is 'None'.

Examples

```
tx1 = rfckt.parallelplate;
tx1.StubMode = 'Series';
tx1.Termination = 'Short';
```

Width

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;
```

Z0

Purpose

Characteristic impedance

Values

Vector

Description

Read-only vector of characteristic impedance values, in ohms, computed by the `analyze` method. The values correspond to the frequencies at which you analyze the transmission line. This property is empty by default.

Examples

```
tx1 = rfckt.parallelplate;
tx1.Z0

ans =

     []
```

Constructor

rfckt.parallelplate

Syntax

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

Description

`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.

Purpose

Model passive component or network

Class Description

Use the `passive` class to represent passive RF components and networks that are characterized by network parameter data.

Constructor Summary

`rfckt.passive`	Construct `rfckt.passive` object

Property Summary

`AnalyzedResult`	Computed S-parameters, noise figure, and OIP3 values
`IntpType`	Interpolation method
`Name`	Object name
`NetworkData`	Network parameter information
`nPort`	Number of ports

Method Summary

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

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

See Also

`rfckt.amplifier`, `rfckt.datafile`, `rfckt.mixer`, `rfdata.data`, `rfdata.network`

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

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

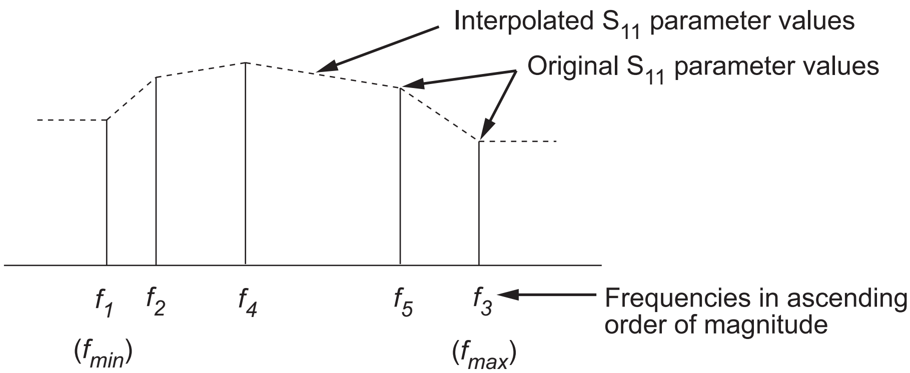
Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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, and OIP3 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.

Examples

```
pas = rfckt.passive;
pas.AnalyzedResult

ans =

           Name: 'Data object'
           Freq: [202x1 double]
```

```
S_Parameters: [2x2x202 double]
NF: [202x1 double]
OIP3: [202x1 double]
Z0: 50
ZS: 50
ZL: 50
IntpType: 'Linear'
```

IntpType

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]
```

Name

Purpose

Object name

Values

'Passive'

Description

Read-only string that contains the name of the object.

Examples

```
pas = rfckt.passive;
pas.Name

ans =

           Passive
```

NetworkData

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
```

nPort

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
```

Constructor

rfckt.passive

Syntax

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

Description

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

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

Use the `read` method to read the passive object 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.

Purpose

Model RLCG transmission line

Class Description

Use the `rlcgline` class to represent RLCG transmission lines that are characterized by line loss, line length, stub type, and termination.

Constructor Summary

`rfckt.rlcgline`	Construct `rfckt.rlcgline` object

Property Summary

`AnalyzedResult`	Computed S-parameters, noise figure, and OIP3 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

Method Summary

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

getz0	Characteristic impedance of transmission line object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis
smith	Plot specified circuit object parameters on Smith chart
write	Write RF data from circuit or data object to file

Examples

```
tx1=rfckt.rlcgline('R',10,'C',1e-12,'L',1e-9,'G',1e7')

tx1 =

             Name: 'RLCG Transmission Line'
             nPort: 2
```

```
AnalyzedResult: []
LineLength: 0.0100
StubMode: 'None'
Termination: 'None'
Freq: 1.0000e+009
R: 10
L: 1.0000e-009
C: 1.0000e-012
G: 10000000
IntpType: Linear'
```

See Also

`rfckt.coaxial`, `rfckt.cpw`, `rfckt.microstrip`, `rfckt.parallelplate`, `rfckt.twowire`, `rfckt.txline`

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

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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 calculates the S-parameters for the specified frequencies, based on the physical length of the transmission line, D, and the complex propagation constant, k.

  $$S_{11} = 0$$
  $$S_{12} = e^{-kD}$$
  $$S_{21} = e^{-kD}$$
  $$S_{22} = 0$$

  k is a vector whose elements correspond to the elements of the input vector `freq`. $k = \alpha_a + i\beta$, where α_a is the attenuation coefficient and β is the wave number. The attenuation coefficient α_a is related to the loss, α, by

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

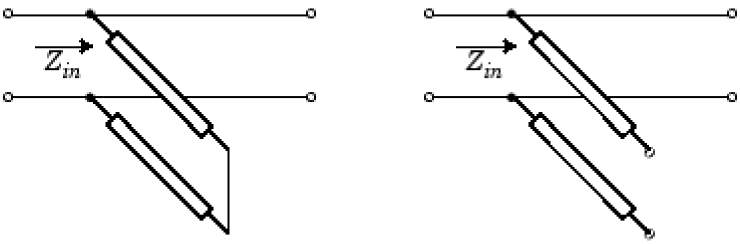
  and the wave number β is related to the phase velocity, V_P, by

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

  where f is the frequency range specified in the `analyze` input argument `freq`. The phase velocity V_P is derived from the `rfckt.rlcgline` 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.

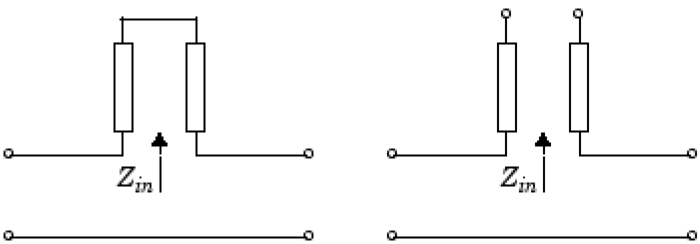
When you set the StubMode property to 'Shunt', the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown in the following figure.



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:

$A = 1$

$B = Z_{in}$

$C = 0$

$D = 1$

Examples

```
tx1 = rfckt.rlcgline;
analyze(tx1,[1e9,2e9,3e9]);
tx1.AnalyzedResult

ans =

             Name: 'Data object'
             Freq: [3x1 double]
             S_Parameters: [2x2x3 double]
             NF: [3x1 double]
             OIP3: [3x1 double]
             Z0: 50
             ZS: 50
             ZL: 50
             IntpType: 'Linear'
```

C

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;
```

Freq

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;
```

G

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]*1e6;
```

IntpType

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';
```

L

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;
```

LineLength

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;
```

Name

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
```

nPort

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
```

R

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];
```

StubMode

Purpose

Type of stub

Values

`'None'` (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';
```

Termination

Purpose

Stub transmission line termination

Values

`'None'` (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 `'None'` when `StubMode` is `'None'`.

Examples

```
tx1 = rfckt.rlcgline;
tx1.StubMode = 'Series';
tx1.Termination = 'Short';
```

Constructor

rfckt.rlcgline

Syntax

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

Description

`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.

Purpose

Model series connected network

Class 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.

Constructor Summary

`rfckt.series`	Construct `rfckt.series` object

Property Summary

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

Method Summary

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane

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

See Also

`rfckt.cascade`, `rfckt.hybrid`, `rfckt.hybridg`, `rfckt.parallel`

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

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

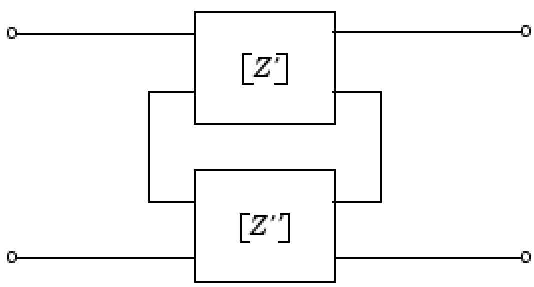
`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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 $[Z'] = \begin{bmatrix} Z_{11}' & Z_{12}' \\ Z_{21}' & Z_{22}' \end{bmatrix}$ and $[Z''] = \begin{bmatrix} Z_{11}'' & Z_{12}'' \\ Z_{21}'' & Z_{22}'' \end{bmatrix}$

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`.

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]
    NF: [101x1 double]
    OIP3: [101x1 double]
    Z0: 50
    ZS: 50
    ZL: 50
    IntpType: 'Linear'
```

Ckts

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]
```

Name

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
```

nPort

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
```

Constructor

rfckt.series

Syntax

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

Description

`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.

seriesrlc

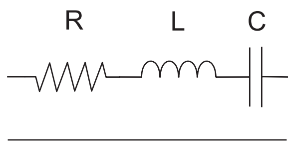
Purpose

Model series RLC component

Class 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.



Constructor Summary

`rfckt.seriesrlc`	Construct `rfckt.seriesrlc` object

Property Summary

`AnalyzedResult`	Computed S-parameters, noise figure, and OIP3 values
`C`	Capacitance value
`L`	Inductance value
`Name`	Object name
`nPort`	Number of ports
`R`	Resistance value

Method Summary

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

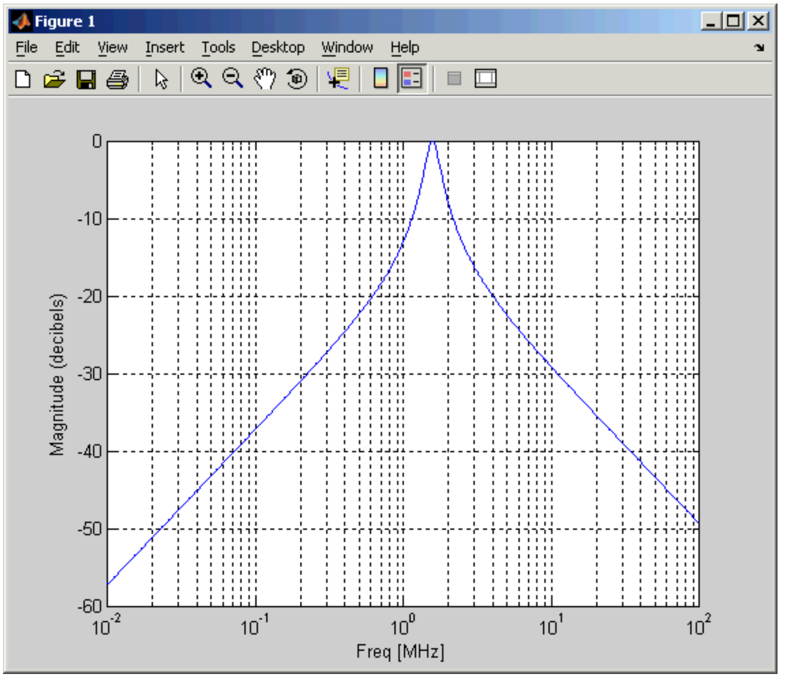
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis
smith	Plot specified circuit object parameters on Smith chart
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):

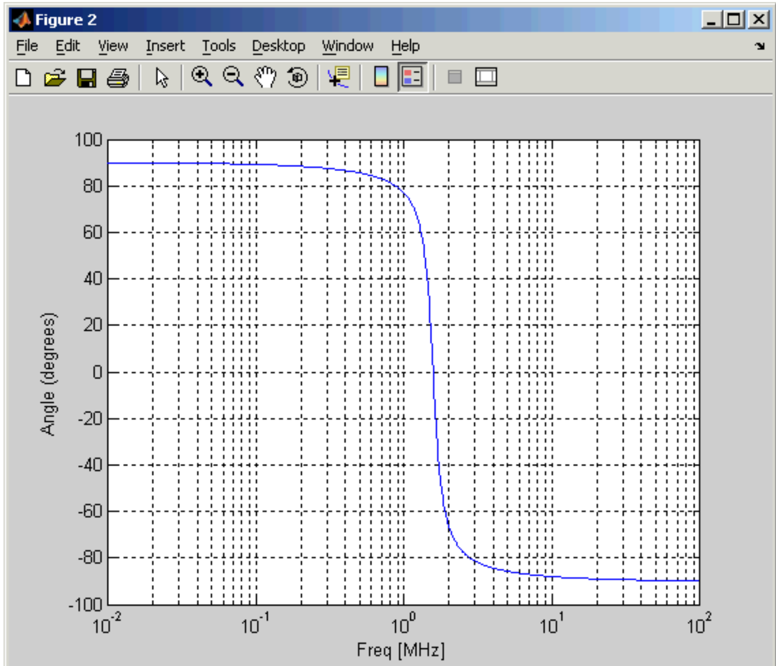
```
h = rfckt.seriesrlc('L',4.7e-5,'C',2.2e-10);
```

```
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')
```



See Also

`rfckt.shuntrlc`

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

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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 $\omega = 2\pi f$.

Examples

```
rlc1 = rfckt.seriesrlc;
analyze(rlc1,[1e9,2e9,3e9]);
rlc1.AnalyzedResult

ans =

           Name: 'Data object'
           Freq: [3x1 double]
           S_Parameters: [2x2x3 double]
           NF: [3x1 double]
           OIP3: [3x1 double]
           Z0: 50
           ZS: 50
           ZL: 50
           IntpType: 'Linear'
```

C

Purpose

Capacitance value

Values

Scalar

Description

Capacitance value in farads. The default is `Inf`.

Examples

```
rlc1=rfckt.seriesrlc;
rlc1.C = 1e-12;
```

L

Purpose

Inductance value

Values

Scalar

Description

Inductance value in henries. The default is `0`.

Examples

```
rlc1 = rfckt.seriesrlc;
rlc1.L = 1e-9;
```

Name

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
```

nPort

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
```

R

Purpose

Resistance value

Values

Scalar

Description

Resistance in ohms. The default is `0`.

Examples

```
rlc1 = rfckt.seriesrlc;
rlc1.R = 10;
```

Constructor

rfckt.seriesrlc

Syntax

```
h = rfckt.seriesrlc
h = rfckt.seriesrlc('R',Rvalue,'L',Lvalue,'C',Cvalue)
```

Description

`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.

shuntrlc

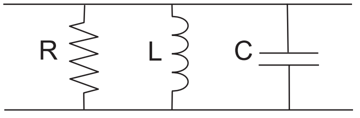
Purpose

Model shunt RLC component

Class 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.



Constructor Summary

`rfckt.shuntrlc`	Construct `rfckt.shuntrlc` object

Property Summary

`AnalyzedResult`	Computed S-parameters, noise figure, and OIP3 values
`C`	Capacitance value
`L`	Inductance value
`Name`	Object name
`nPort`	Number of ports
`R`	Resistance value

Method Summary

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

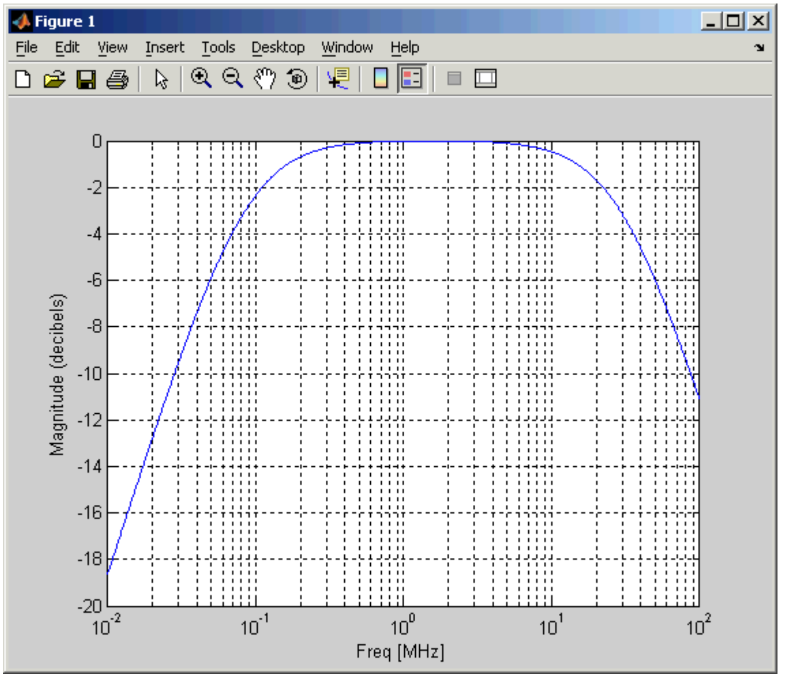
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis
smith	Plot specified circuit object parameters on Smith chart
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):

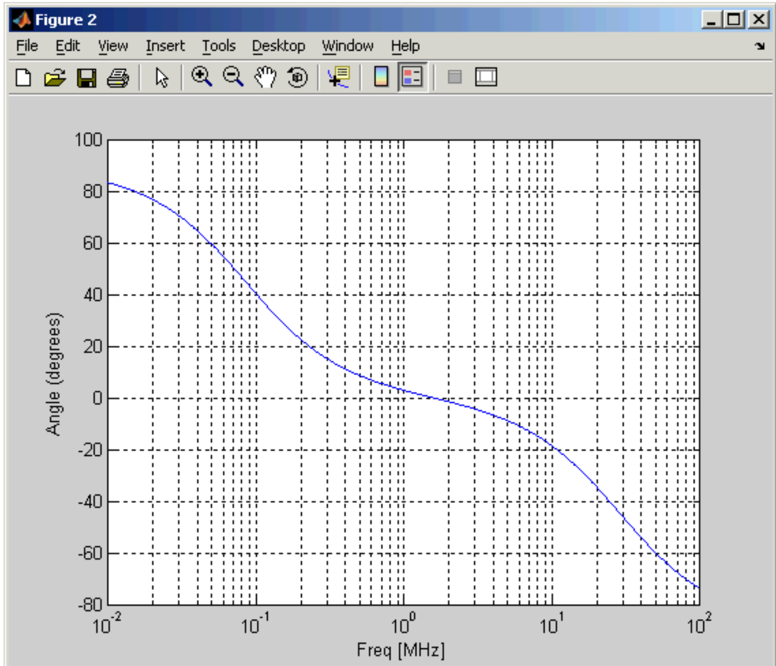
```
h = rfckt.shuntrlc('L',4.7e-5,'C',2.2e-10);
```

```
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')
```



See Also

`rfckt.seriesrlc`

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

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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 $\omega = 2\omega f$.

Examples

```
rlc1 = rfckt.shuntrlc;
analyze(rlc1,[1e9,2e9,3e9]);
rlc1.AnalyzedResult

ans =

            Name: 'Data object'
            Freq: [3x1 double]
            S_Parameters: [2x2x3 double]
            NF: [3x1 double]
            OIP3: [3x1 double]
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Linear'
```

C

Purpose
Capacitance value

Values
Scalar

Description
Capacitance value in farads. The default is `0`.

Examples

```
rlc1=rfckt.shuntrlc;
rlc1.C = 1e-12;
```

L

Purpose
Inductance value

Values
Scalar

Description
Inductance value in henries. The default is `Inf`.

Examples

```
rlc1 = rfckt.shuntrlc;
rlc1.L = 1e-9;
```

Name

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
```

nPort

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
```

R

Purpose

Resistance value

Values

Scalar

Description

Resistance in ohms. The default is `Inf`.

Examples

```
rlc1 = rfckt.shuntrlc;
rlc1.R = 10;
```

Constructor

rfckt.shuntrlc

Syntax

```
h = rfckt.shuntrlc
h = rfckt.shuntrlc('R',Rvalue,'L',Lvalue,'C',Cvalue)
```

Description

`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.

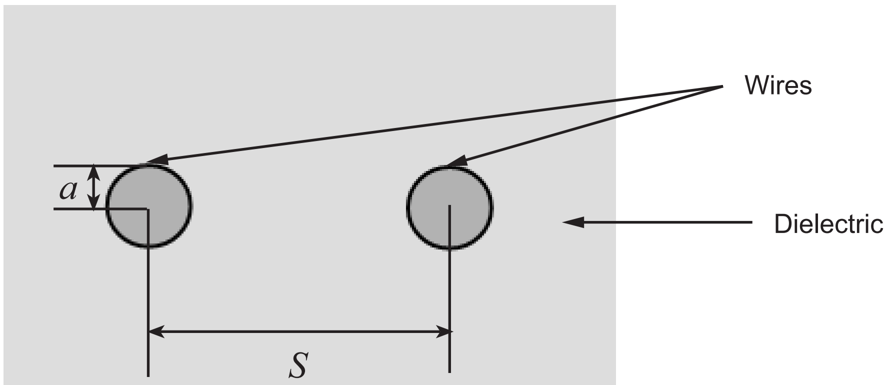
Purpose

Model two-wire transmission line

Class 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 assumes the relative permittivity and permeability are uniform.



Constructor Summary

`rfckt.twowire`	Construct `rfckt.twowire` object

Property Summary

`AnalyzedResult`	Computed S-parameters, noise figure, and OIP3 values
`EpsilonR`	Relative permittivity of dielectric
`LineLength`	Transmission line length
`Loss`	Transmission line loss

MuR	Relative permeability of dielectric
Name	Object name
nPort	Number of ports
PV	Phase velocity
Radius	Wire radius
Separation	Distance between wires
SigmaCond	Conductor conductivity
SigmaDiel	Dielectric conductivity
StubMode	Type of stub
Termination	Stub transmission line termination
Z0	Characteristic impedance

Method Summary

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
getz0	Characteristic impedance of transmission line object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane

plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis
smith	Plot specified circuit object parameters on Smith chart
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: 'None'
            Termination: 'None'
            Radius: 7.5000e-004
            Separation: 0.0016
            MuR: 1
            EpsilonR: 2.3000
            SigmaCond: Inf
            SigmaDiel: 0
```

See Also

`rfckt.coaxial`, `rfckt.cpw`, `rfckt.microstrip`, `rfckt.parallelplate`, `rfckt.rlcgline`, `rfckt.txline`

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

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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 calculates the S-parameters for the specified frequencies, based on the physical length of the transmission line, D, and the complex propagation constant, k.

$$S_{11} = 0$$

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

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

$$S_{22} = 0$$

k is a vector whose elements correspond to the elements of the input vector `freq`. $k = \alpha_a + i\beta$, where α_a is the attenuation coefficient and β is the wave number. The attenuation coefficient α_a is related to the loss, α, by

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

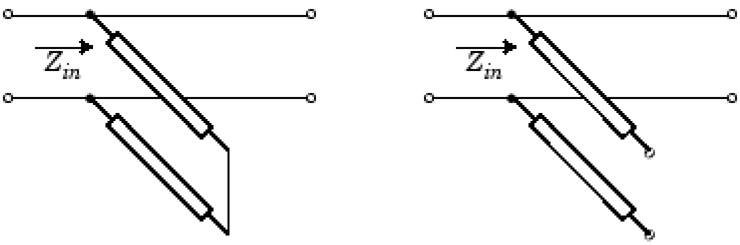
and the wave number β is related to the phase velocity, V_P, by

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

where f is the frequency range specified in the `analyze` input argument `freq`. The phase velocity V_P is derived from the `rfckt.txline` object properties. It is also known as the *wave propagation velocity*.

- 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:

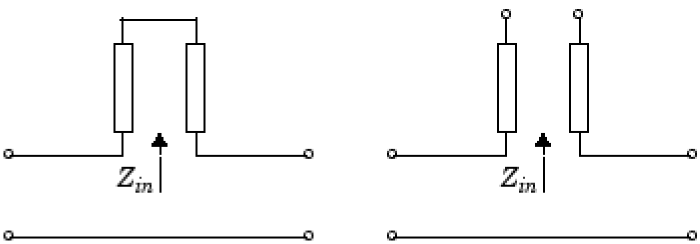
$A = 1$

$B = 0$

$C = 1/Z_{in}$

$D = 1$

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:

$A = 1$

$B = Z_{in}$

$C = 0$

$D = 1$

Examples

```
tx1 = rfckt.twowire;
analyze(tx1,[1e9,2e9,3e9]);
tx1.AnalyzedResult

ans =

             Name: 'Data object'
             Freq: [3x1 double]
             S_Parameters: [2x2x3 double]
             NF: [3x1 double]
             OIP3: [3x1 double]
             Z0: 50
             ZS: 50
             ZL: 50
             IntpType: 'Linear'
```

EpsilonR

Purpose

Relative permittivity of dielectric

Values

Scalar

Description

The ratio of the permittivity of the dielectric to the permittivity in free space ε_0. The default is `2.3`.

Examples

```
tx1=rfckt.twowire;
tx1.EpsilonR=2.7;
```

LineLength

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;
```

Loss

Purpose

Transmission line loss

Values

Vector

Description

Read-only vector of line loss values, in decibels per meter, computed by the `analyze` method. The values correspond to the frequencies at which you analyze the transmission line. Line loss is the reduction in strength of the signal as it travels over the transmission line. This property is empty by default.

Examples

```
tx1 = rfckt.twowire;
tx1.Loss

ans =
```

```
[]
```

MuR

Purpose

Relative permeability of dielectric

Values

Scalar

Description

The ratio of the permeability of the dielectric to the permeability in free space μ_0. The default is 1.

Examples

```
tx1=rfckt.twowire;
tx1.MuR=0.8;
```

Name

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
```

nPort

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
```

PV

Purpose

Phase velocity

Values

Vector

Description

Read-only vector of phase velocity values, in meters per second, computed by the `analyze` method. The values correspond to the frequencies at which you analyze the transmission line. Propagation velocity of a uniform plane wave on the transmission line. This property is empty by default.

Examples

```
tx1 = rfckt.twowire;
tx1.PV

ans =

     []
```

Radius

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;
```

Separation

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;
```

SigmaCond

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;
```

SigmaDiel

Purpose

Dielectric conductivity

Values

Scalar

Description

Conductivity, in Siemens per meter (S/m), of the dielectric. The default is `0`.

Examples

```
tx1=rfckt.twowire;
tx1.SigmaDiel=0.002;
```

`StubMode`

Purpose

Type of stub

Values

`'None'` (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';
```

`Termination`

Purpose

Stub transmission line termination

Values

`'None'` (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 `'None'` when `StubMode` is `'None'`.

Examples

```
tx1 = rfckt.twowire;
tx1.StubMode = 'Series';
tx1.Termination = 'Short';
```

Z0

Purpose

Characteristic impedance

Values

Vector

Description

Read-only vector of characteristic impedance values, in ohms, computed by the `analyze` method. The values correspond to the frequencies at which you analyze the transmission line. This property is empty by default.

Examples

```
tx1 = rfckt.twowire;
tx1.Z0

ans =

     []
```

Constructor

rfckt.twowire

Syntax

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

Description

`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.

Purpose

Model general transmission line

Class Description

Use the txline class to represent transmission lines that are characterized by line loss, line length, stub type, and termination.

Constructor Summary

rfckt.txline	Construct rfckt.txline object

Property Summary

AnalyzedResult	Computed S-parameters, noise figure, and OIP3 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

Method Summary

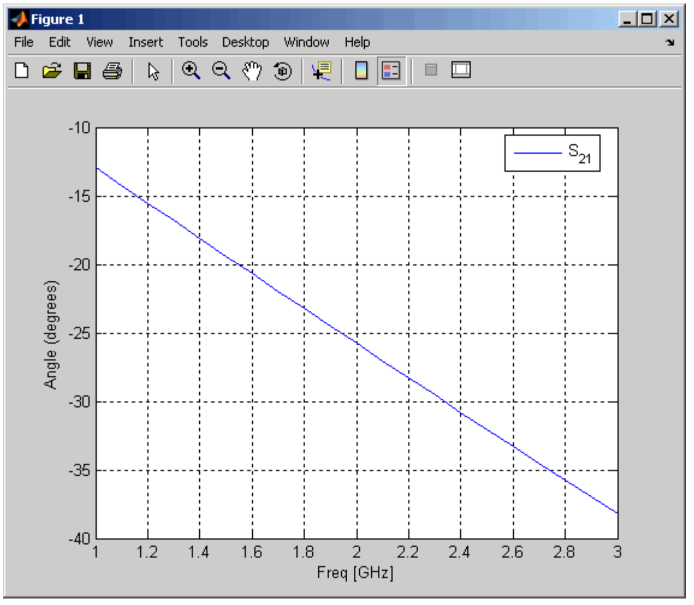
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object

getz0	Characteristic impedance of transmission line object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis
smith	Plot specified circuit object parameters on Smith chart
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: 'None'
              Termination: 'None'
              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
```



See Also

`rfckt.coaxial`, `rfckt.cpw`, `rfckt.microstrip`, `rfckt.parallelplate`, `rfckt.rlcgline`, `rfckt.twowire`

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

Properties

AnalyzedResult

Purpose

Computed S-parameters, noise figure, and OIP3 values

Values

`rfdata.data` object

Description

Handle to an `rfdata.data` object that contains the S-parameters, noise figure, and OIP3 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 calculates the S-parameters for the specified frequencies, based on the physical length of the transmission line, D, and the complex propagation constant, k.

  $$S_{11} = 0$$

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

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

  $$S_{22} = 0$$

  k is a vector whose elements correspond to the elements of the input vector `freq`. $k = \alpha_a + i\beta$, where α_a is the attenuation coefficient and β is the wave number. The attenuation coefficient α_a is related to the loss, α, by

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

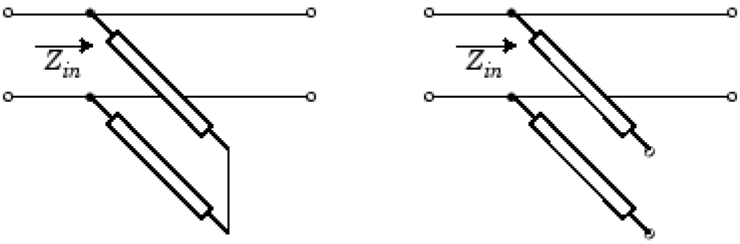
  and the wave number β is related to the phase velocity, V_P, by

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

  where f is the frequency range specified in the `analyze` input argument `freq`. The phase velocity V_P is derived from the `rfckt.txline` object properties. It is also known as the *wave propagation velocity*.

- 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.

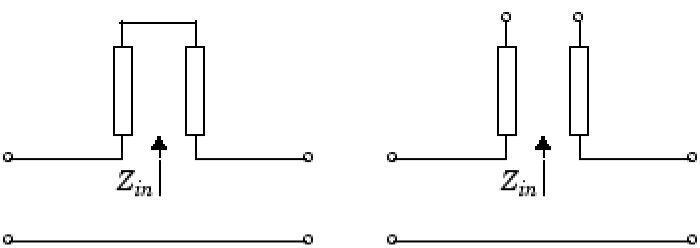
  When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown in the following figure.

  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:

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

Examples

```
tx1 = rfckt.txline;
analyze(tx1,[1e9,2e9,3e9]);
tx1.AnalyzedResult

ans =

                Name: 'Data object'
                Freq: [3x1 double]
                S_Parameters: [2x2x3 double]
                NF: [3x1 double]
                OIP3: [3x1 double]
                Z0: 50
                ZS: 50
                ZL: 50
                IntpType: 'Linear'
```

Freq

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;
```

IntpType

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';
```

LineLength

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;
```

Loss

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;
```

Name

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
```

nPort

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
```

PV

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;
```

StubMode

Purpose

Type of stub

Values

`'None'` (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';
```

Termination

Purpose

Stub transmission line termination

Values

'None' (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 'None' when StubMode is 'None'.

Examples

```
tx1 = rfckt.txline;
tx1.StubMode = 'Series';
tx1.Termination = 'Short';
```

Z0

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;
```

Constructor

rfckt.txline

Syntax

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

Description

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

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

Purpose

Store result of circuit object analysis

Class 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 stores the results in an `rfdata.data` object.

Constructor Summary

`rfdata.data`	Construct `rfdata.data` object

Property Summary

`Freq`	Frequency 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

Method Summary

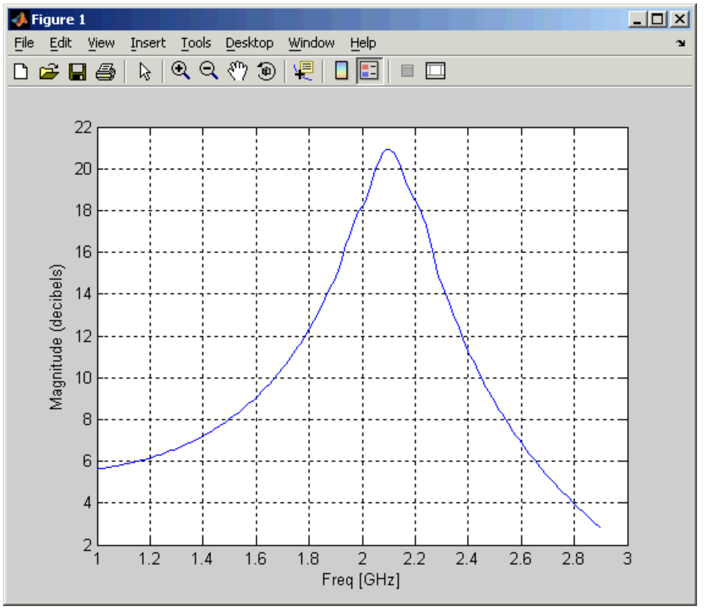
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
extract	Extract array of network parameters from data object
getop	Display operating conditions
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object
loglog	Plot specified circuit object parameters using log-log scale
plot	Plot specified circuit object parameters on X-Y plane
plotyy	Plot specified object parameters with y-axes on both left and right sides
polar	Plot specified circuit object parameters on polar coordinates
read	Read RF data from file to new or existing circuit or data object
restore	Restore data to original frequencies
semilogx	Plot specified circuit object parameters using log scale for X-axis
semilogy	Plot specified circuit object parameters using log scale for Y-axis

smith	Plot specified circuit object parameters on Smith chart
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

Properties

Freq

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;
```

IntpType

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: []
            NF: 0
            OIP3: Inf
            Z0: 50
            ZS: 50
            ZL: 50
            IntpType: 'Cubic'
```

Name

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
```

NF

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: []
            NF: 3
            OIP3: Inf
            Z0: 50
```

```
ZS: 50
ZL: 50
IntpType: 'Linear'
```

OIP3

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;
```

S_Parameters

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;
```

Z0

Purpose

Reference impedance

Values

Scalar

Description

Scalar reference impedance in ohms. The default is 50 ohms.

Examples

```
txdata = rfdata.data;
txdata.Z0 = 75;
```

ZL

Purpose

Load impedance

Values

Scalar

Description

Scalar load impedance in ohms. The default is 50 ohms.

Examples

```
txdata = rfdata.data;
txdata.ZL = 75;
```

ZS

Purpose

Source impedance

Values

Scalar

Description

Scalar source impedance in ohms. The default is 50 ohms.

Examples

```
txdata = rfdata.data;
txdata.ZS = 75;
```

Constructor

rfdata.data

Syntax

```
h = rfdata.data
h = rfdata.data('Property1',value1,'Property2',value2,...)
```

Description

`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.

Purpose

Store frequency-dependent, third-order intercept points

Class Description

Use the ip3 class to store third-order intercept point specifications for a circuit object.

Constructor Summary

rfdata.ip3	Construct rfdata.ip3 object

Property Summary

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

Properties

Data

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;
```

Freq

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;
```

Name

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
```

Type

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`

Constructor

rfdata.ip3

Syntax

```
h = rfdata.ip3
h = rfdata.ip3('Type',value1,'Freq',value2,'Data',value3)
```

Description

`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.

Purpose

Store data from intermodulation table

Class Description

Use the `mixerspur` class to store mixer spur power specifications for a circuit object.

Constructor Summary

`rfdata.mixerspur`	Construct `rfdata.mixerspur` object

Property Summary

`Data`	Mixer spur power values
`Name`	Object name
`PinRef`	Reference input power
`PLORef`	Reference local oscillator power

Examples

```
spurs=rfdata.mixerspur('Data',[2 5; 1 0],'PinRef',3,'PLORef',5)

spurs =

      Name: 'Intermodulation table'
    PLORef: 5
    PinRef: 3
      Data: [2x2 double]
```

See Also

`rfdata.data`, `rfdata.ip3`, `rfdata.network`, `rfdata.nf`, `rfdata.noise`, `rfdata.power`

Visualizing Mixer Spurs

Properties

Data

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; 1 0],'PinRef',3,'PLORef',5)
spurs.data

ans =

      2   5
      1   0
```

Name

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
```

PinRef

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; 1 0],'PinRef',3,'PLORef',5)
spurs.PinRef

ans =

      3
```

PLORef

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; 1 0],'PinRef',3,'PLORef',5)
spurs.PLORef

ans =

     5
```

Constructor

rfdata.mixerspur

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.

Purpose

Store frequency-dependent network parameters

Class Description

Use the `network` class to store frequency-dependent S-, Y-, or Z-parameters for a circuit object.

Constructor Summary

`rfdata.network`	Construct `rfdata.network` object

Property Summary

`Data`	Network parameter data
`Freq`	Frequency data
`Name`	Object name
`Type`	Type of network parameters.
`Z0`	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`

Properties

Data

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;
```

Freq

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;
```

Name

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
```

Type

Purpose

Type of network parameters.

Values

`'S'`, `'Y'`, `'Z'`, `'H'`, `'G'`, or `'T'`

Description

String that indicates whether the rfdata.network object .

Examples

```
netdata=rfdata.network;
netdata.Type='Y';
```

Z0

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;
```

Constructor

rfdata.network

Syntax

```
h = rfdata.network
h = rfdata.network('Type',value1,'Freq',value2, Data',value3,
    'Z0',value4)
```

Description

`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, 'Z0',value4)` returns a data object for the frequency-dependent network parameters, `h`, based on the specified properties.

Purpose Store frequency-dependent noise figure data for amplifiers or mixers

Class Description Use the `nf` class to store noise figure specifications for a circuit object.

Constructor Summary

`rfdata.nf`	Construct `rfdata.nf` object

Property Summary

`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`

Properties

`Data`

Purpose

Noise figure values

Values

Vector

Description

M-element vector of noise figure data, in watts, 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;
```

Freq

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;
```

Name

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
```

Constructor

rfdata.nf

Syntax

```
h = rfdata.nf
h = rfdata.nf('Freq',value1,'Data',value2)
```

Description

`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.

noise

Purpose

Store frequency-dependent spot noise data for amplifiers or mixers

Class Description

Use the `noise` class to store spot noise specifications for a circuit object.

Constructor Summary

`rfdata.noise`	Construct `rfdata.noise` object

Property Summary

`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`

Properties

FMIN

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;
```

Freq

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;
```

GAMMAOPT

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;
```

Name

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
```

RN

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;
```

Constructor

rfdata.noise

Syntax

```
h = rfdata.noise
h =  rfdata.noise('Freq',value1,'FMIN',value2,'GAMMAOPT',
   value3,'RN',value4)
```

Description

`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.

Purpose

Store output power and phase information for amplifiers or mixers

Class Description

Use the `power` class to store output power and phase specifications for a circuit object.

Constructor Summary

`rfdata.power`	Construct `rfdata.power` object

Property Summary

`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`

Properties

Freq

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;
```

Name

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
```

Phase

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;
```

Pin

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;
```

Pout

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;
```

Constructor

rfdata.power

Syntax

```
h = rfdata.power
h = rfdata.power(`property1',value1,'property2',value2,...)
```

Description

`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.

rational

Purpose

Rational function model

Class Description

Use the `rational` class to represent RF components using a rational function model of the form

$$F(s) = \left(\sum_{k=1}^{n} \frac{C_k}{s - A_k} + D\right) e^{-s*Delay}, \quad s = j2\pi * freq \tag{7-1}$$

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.

Constructor Summary

`rfmodel.rational`	Construct `rfmodel.rational` object

Property Summary

`A`	Poles of rational function
`C`	Residues of rational function
`D`	Frequency response offset
`Delay`	Frequency response time delay
`Name`	Object name

Method Summary

freqresp	Calculate frequency response of model object
timeresp	Calculate time response for model object
writeva	Write Verilog-A description of RF model object

Examples

```
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
```

Properties

A

Purpose

Poles of rational function

Values

Vector

Description

Complex vector containing poles of the rational function in radians per second. Its length, shown in Equation 7-1 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;
```

C

Purpose

Residues of rational function

Values

Vector

Description

Complex vector containing residues of the rational function in radians per second. Its length, shown in Equation 7-1 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;
```

D

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;
```

Delay

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;
```

Name

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
```

Constructor

rfmodel.rational

Syntax

```
h = rfmodel.rational
h = h = rfmodel.rational('Property1',value1,'Property2',
   value2,...)
```

Description

`h = rfmodel.rational` returns a rational function model object whose properties are set to their default values.

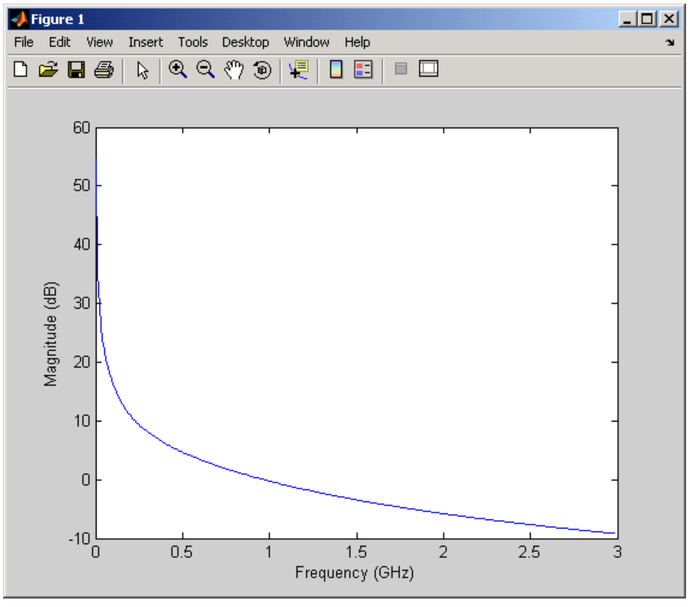
`h =`
`h = rfmodel.rational('Property1',value1,'Property2',value2,...)` returns a rational function model object, `h`, with the specified properties. Properties that you do not specify retain their default values.

Examples

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. Then, 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];           % Simulation frequencies
```

```
[resp,freq]=freqresp(rat,f);   % Compute frequency response
figure
 plot(freq/1e9,db(resp));      % Plot frequency response
 xlabel('Frequency (GHz)')
 ylabel('Magnitude (dB)')
```



Methods — By Category

Analysis (p. 8-2)	Calculate parameters of circuit objects, model objects, and networks
Plots and Charts (p. 8-2)	Display circuit object parameters
Parameters and Formats (p. 8-3)	List available parameters and formats for plots
Operating Conditions (p. 8-3)	Set or display operating condition information
Data I/O (p. 8-3)	Read or write data to or from circuit or data objects
Data Access and Restoration (p. 8-3)	Get object data in standard form or revert to original data

Analysis

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
freqresp	Calculate frequency response of model object
timeresp	Calculate time response for model object

Plots and Charts

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

Parameters and Formats

listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters for specified circuit object

Operating Conditions

getop	Display operating conditions
setop	Set operating conditions

Data I/O

read	Read RF data from file to new or existing circuit or data object
write	Write RF data from circuit or data object to file
writeva	Write Verilog-A description of RF model object

Data Access and Restoration

extract	Extract array of network parameters from data object
getz0	Characteristic impedance of transmission line object
restore	Restore data to original frequencies

9

Methods — Alphabetical List

analyze

Purpose

Analyze circuit object in frequency domain

Syntax

```
analyze(h,freq)
analyze(h,freq,zl,zs,zo)
analyze(h,freq,'condition1',value1,...,'conditionm',valuem)
```

Description

`analyze(h,freq)` calculates the circuit network parameters, noise figure values, and OIP3 values at the specified frequencies. `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. OIP3 is always ∞ for passive circuits.

`analyze(h,freq,zl,zs,zo)` calculates the circuit network parameters, noise figure, and OIP3 at the specified frequencies. The arguments `zl`, `zs`, and `zo` are optional. These arguments represent the circuit load, circuit source, and reference impedances of the S-parameters, respectively. The default value of all these arguments is 50 ohms.

Note When you specify impedance values, the `analyze` method changes the object's values to match your specification.

`analyze(h,freq,'condition1',value1,...,'conditionm',valuem)` calculates the network parameters, noise figure, and OIP3 at the specified frequency values and operating conditions for the object `h`. `condition1,value1,...`, `conditionm,valuem` are the condition/value pairs at which to analyze the object. This syntax is usually used 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 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`
- `rfckt.passive`
- `rfckt.amplifier`
- `rfckt.mixer`

By default, the `AnalyzedResult` property of `rfckt.datafile` objects contains the S-parameters and noise figure 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.

By default, the `AnalyzedResult` property of `rfckt.passive` objects contains the S-parameters and noise figure 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.

By default, the AnalyzedResult property of rfckt.amplifier objects contains the S-parameter, noise figure, and OIP3 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.

By default, the AnalyzedResult property of rfckt.mixer objects contains the S-parameter, noise figure, and OIP3 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 works for a particular object, see the AnalyzedResult property on the reference page for that object.

See Also

calculate	RF Toolbox
extract	RF Toolbox
getz0	RF Toolbox
listformat	RF Toolbox
listparam	RF Toolbox
loglog	RF Toolbox
plot	RF Toolbox
plotyy	RF Toolbox
polar	RF Toolbox
semilogx	RF Toolbox
semilogy	RF Toolbox
smith	RF Toolbox

`read`	RF Toolbox
`restore`	RF Toolbox
`write`	RF Toolbox

calculate

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 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`, `OIP3`, `VSWRIn`, `VSWROut`, `GAMMAIn`, `GAMMAOut`, `FMIN`, `GAMMAOPT`, `RN`	`Freq` (default), `Pin`
`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`	RF Toolbox
`extract`	RF Toolbox
`getz0`	RF Toolbox
`listformat`	RF Toolbox
`listparam`	RF Toolbox

`loglog`	RF Toolbox
`plot`	RF Toolbox
`plotyy`	RF Toolbox
`polar`	RF Toolbox
`semilogx`	RF Toolbox
`semilogy`	RF Toolbox
`smith`	RF Toolbox
`read`	RF Toolbox
`restore`	RF Toolbox
`write`	RF Toolbox

extract

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`	RF Toolbox
`calculate`	RF Toolbox
`getz0`	RF Toolbox
`listformat`	RF Toolbox
`listparam`	RF Toolbox
`loglog`	RF Toolbox
`plot`	RF Toolbox
`plotyy`	RF Toolbox
`polar`	RF Toolbox
`semilogx`	RF Toolbox
`semilogy`	RF Toolbox
`smith`	RF Toolbox
`read`	RF Toolbox

`restore`	RF Toolbox
`write`	RF Toolbox

freqresp

Purpose Calculate 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 positive vector of 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`	RF Toolbox
`rfmodel.rational`	RF Toolbox
`timeresp`	RF Toolbox
`writeva`	RF Toolbox

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.

See Also

`setop`	RF Toolbox

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`	RF Toolbox
`calculate`	RF Toolbox
`extract`	RF Toolbox
`listformat`	RF Toolbox
`listparam`	RF Toolbox
`loglog`	RF Toolbox
`plot`	RF Toolbox
`plotyy`	RF Toolbox
`polar`	RF Toolbox
`semilogx`	RF Toolbox
`semilogy`	RF Toolbox
`smith`	RF Toolbox
`read`	RF Toolbox
`restore`	RF Toolbox
`write`	RF Toolbox

impulse

Purpose

Calculate impulse response for model object

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`	RF Toolbox
`rationalfit`	RF Toolbox
`rfmodel.rational`	RF Toolbox
`writeva`	RF Toolbox

listformat

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)'`.

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`	RF Toolbox
`calculate`	RF Toolbox
`extract`	RF Toolbox
`getz0`	RF Toolbox
`listparam`	RF Toolbox
`loglog`	RF Toolbox
`plot`	RF Toolbox
`plotyy`	RF Toolbox
`polar`	RF Toolbox
`semilogx`	RF Toolbox
`semilogy`	RF Toolbox
`smith`	RF Toolbox
`read`	RF Toolbox
`restore`	RF Toolbox
`write`	RF Toolbox

listparam

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.

Examples

The following example 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)

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

See Also

`analyze`	RF Toolbox
`calculate`	RF Toolbox

`extract`	RF Toolbox
`getz0`	RF Toolbox
`listformat`	RF Toolbox
`loglog`	RF Toolbox
`plot`	RF Toolbox
`plotyy`	RF Toolbox
`polar`	RF Toolbox
`semilogx`	RF Toolbox
`semilogy`	RF Toolbox
`smith`	RF Toolbox
`read`	RF Toolbox
`restore`	RF Toolbox
`write`	RF Toolbox

loglog

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`, `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 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	RF Toolbox
calculate	RF Toolbox
extract	RF Toolbox
getz0	RF Toolbox
listformat	RF Toolbox
listparam	RF Toolbox
plot	RF Toolbox
plotyy	RF Toolbox
polar	RF Toolbox
read	RF Toolbox
restore	RF Toolbox
semilogx	RF Toolbox
semilogy	RF Toolbox

`smith`	RF Toolbox
`write`	RF Toolbox

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 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`, `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 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
VSWRIn, VSWROut	Magnitude (decibels) Magnitude (linear)
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 demo.

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.

See Also

`analyze`	RF Toolbox
`calculate`	RF Toolbox
`extract`	RF Toolbox
`getz0`	RF Toolbox
`listformat`	RF Toolbox
`listparam`	RF Toolbox
`loglog`	RF Toolbox
`plotyy`	RF Toolbox
`polar`	RF Toolbox
`read`	RF Toolbox
`restore`	RF Toolbox
`semilogx`	RF Toolbox
`semilogy`	RF Toolbox
`smith`	RF Toolbox
`write`	RF Toolbox

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 a predefined pair of formats for the left and right Y-axes. The formats define how RF Toolbox displays the data on the plot. `h` is the handle of a circuit (`rfckt`) or an `rfdata.data` object.

For some parameters, a related parameter is also plotted by default. The following table shows the predefined formats for the parameters for all circuit and data objects.

Specified Parameter/Related Parameter	Left Y-Axis Format	Right Y-Axis Format
S11, S12, S21, S22	Magnitude (decibels)	Angle (Degrees)
LS11, LS12, LS21, LS22	Magnitude (decibels)	Angle (Degrees)
NF	Magnitude (decibels)	Magnitude (linear)
OIP3	dBm	W

Specified Parameter/Related Parameter	Left Y-Axis Format	Right Y-Axis Format
POUT/Phase	dBm	Angle (Degrees)
Phase/POUT	Angle (Degrees)	dBm
AM/AM / AM/PM	Magnitude (decibels)	Angle (Degrees)
AM/PM / AM/AM	Angle (Degrees)	Magnitude (decibels)
GAMMAIn, GAMMAOut	Magnitude (decibels)	Magnitude (linear)
VSWRIn, VSWROut	Magnitude (decibels)	Magnitude (linear)
FMIN	Magnitude (decibels)	Magnitude (linear)
GAMMAOPT	Magnitude (decibels)	Angle (Degrees)
RN	None	None

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 and `ax(2)` is the right axes.`hlines1` is the `lineseries` object for the left Y-axis, and `hlines2` is the 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 2 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` using the predefined formats for the left and right Y-axes.

`[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 parametersparameter1`,...`, `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`, `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 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.

See Also

`analyze`	RF Toolbox
`calculate`	RF Toolbox
`extract`	RF Toolbox
`getz0`	RF Toolbox
`listformat`	RF Toolbox
`listparam`	RF Toolbox
`loglog`	RF Toolbox
`plot`	RF Toolbox
`polar`	RF Toolbox
`read`	RF Toolbox
`restore`	RF Toolbox
`semilogx`	RF Toolbox
`semilogy`	RF Toolbox
`smith`	RF Toolbox
`write`	RF Toolbox

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`	RF Toolbox
`calculate`	RF Toolbox
`extract`	RF Toolbox
`getz0`	RF Toolbox
`listformat`	RF Toolbox
`listparam`	RF Toolbox
`loglog`	RF Toolbox
`plot`	RF Toolbox
`plotyy`	RF Toolbox
`read`	RF Toolbox
`restore`	RF Toolbox
`semilogx`	RF Toolbox
`semilogy`	RF Toolbox
`smith`	RF Toolbox
`write`	RF Toolbox

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 more information on reading data from a file, see "Importing Property Values from Data Files" on page 3-8. For information about the `.amp` format, see Appendix A, "AMP File Format".

`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.

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	RF Toolbox
calculate	RF Toolbox
extract	RF Toolbox
getz0	RF Toolbox
listformat	RF Toolbox
listparam	RF Toolbox
loglog	RF Toolbox
plot	RF Toolbox
plotyy	RF Toolbox
polar	RF Toolbox
restore	RF Toolbox
semilogx	RF Toolbox
semilogy	RF Toolbox
smith	RF Toolbox
write	RF Toolbox

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`	RF Toolbox
`calculate`	RF Toolbox
`extract`	RF Toolbox
`getz0`	RF Toolbox
`listformat`	RF Toolbox
`listparam`	RF Toolbox
`loglog`	RF Toolbox
`plot`	RF Toolbox
`plotyy`	RF Toolbox
`polar`	RF Toolbox
`semilogx`	RF Toolbox
`semilogy`	RF Toolbox
`smith`	RF Toolbox
`read`	RF Toolbox
`write`	RF Toolbox

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, 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 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	RF Toolbox
calculate	RF Toolbox
extract	RF Toolbox
getz0	RF Toolbox
listformat	RF Toolbox
listparam	RF Toolbox
loglog	RF Toolbox
plot	RF Toolbox
plotyy	RF Toolbox
polar	RF Toolbox
read	RF Toolbox
restore	RF Toolbox
semilogy	RF Toolbox

`smith`	RF Toolbox
`write`	RF Toolbox

Purpose

Plot specified circuit object parameters using log scale for Y-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`, `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 `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`	RF Toolbox
`calculate`	RF Toolbox
`extract`	RF Toolbox
`getz0`	RF Toolbox
`listformat`	RF Toolbox
`listparam`	RF Toolbox
`loglog`	RF Toolbox
`plot`	RF Toolbox
`plotyy`	RF Toolbox
`polar`	RF Toolbox
`read`	RF Toolbox
`restore`	RF Toolbox
`semilogx`	RF Toolbox

`smith`	RF Toolbox
`write`	RF Toolbox

Purpose

Set operating conditions

Syntax

```
setop(h,'Condition1',value1,'Condition2',value2,...)
setop(h,'Condition1')
setop(h)
```

Description

`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.

Note RF Toolbox 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.

You can only change operating conditions for objects you import from a `.p2d` or `.s2d` file.

`setop(h,'Condition1')` lists the available values for the specified operating condition `'Condition1'`.

`setop(h)` lists the available values for all operating conditions of the object `h`.

When you set the operating conditions for a network that contains several objects, RF Toolbox 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 one of these types of networks, use a separate call to the `setop` method for each object.

See Also

`getop`	RF Toolbox

Purpose

Plot specified circuit object parameters on Smith chart

Syntax

`[lineseries,hsm] = smith(h,parameter1,...,parametern,type)`

Description

`[lineseries,hsm] = smith(h,parameter1,...,parametern,type)` plots the network parameters `parameter1,...`, `parametern` from the object `h` on a Smith chart. `h` is the handle of a circuit (`rfckt`) or data (`rfdata`) object. `type` is a string, `'z'` (default),`'y'`, or `'zy'`, specifying the type of Smith chart.

Type `listparam(h)` to get a list of valid parameters for a circuit object `h`.

Note For all circuit objects except those that contain data from a data file, you must use the `analyze` 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 `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 `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).

Property Name	Description	Units, Values
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.
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`	RF Toolbox
`calculate`	RF Toolbox
`getz0`	RF Toolbox
`listformat`	RF Toolbox
`listparam`	RF Toolbox
`loglog`	RF Toolbox
`plot`	RF Toolbox
`plotyy`	RF Toolbox
`polar`	RF Toolbox
`read`	RF Toolbox
`restore`	RF Toolbox
`semilogx`	RF Toolbox
`semilogy`	RF Toolbox
`write`	RF Toolbox

timeresp

Purpose

Calculate 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 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`	RF Toolbox
`rationalfit`	RF Toolbox
`rfmodel.rational`	RF Toolbox
`writeva`	RF Toolbox

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`. See Appendix A, "AMP File Format" for information about the `.amp` format. `write` returns `True` if the operation is successful and returns `False` otherwise.

`dataformat` specifies the format of the data to be written. It must be one of the case-insensitive strings in the following table.

Format	Description
'DB'	Data is given in (dB-magnitude, angle) pairs with angle in degrees.
'MA'	Data is given in (magnitude, angle) pairs with angle in degrees.
'RI'	Data is given in (real, imaginary) pairs (default).

`funit` specifies the frequency units of the data to be written. It must be `'GHz'`, `'MHz'`, `'KHz'`, or `'Hz'`. If you do not specify `funit`, its value is taken from the object `data`. All values are case-insensitive.

`printformat` is a string that specifies the precision of the network and noise parameters. See the Format String specification for `fprintf`.

`freqformat` is a string that specifies the precision of the frequency. See the Format String specification for `fprintf`.

Note Touchstone files, which have the `.snp`, `.ynp`, `.znp`, or `.hnp` extension, do not support noise figure, output third-order intercept point, source impedance, load impedance, or interpolation method data. Consequently, the `write` method does not write these property values to these files. AMP files do not support source impedance, load impedance, or interpolation method data. Consequently, the `write` method does not write these property values to these files.

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`	RF Toolbox
`calculate`	RF Toolbox
`extract`	RF Toolbox
`getz0`	RF Toolbox
`listformat`	RF Toolbox
`listparam`	RF Toolbox
`loglog`	RF Toolbox
`plot`	RF Toolbox
`plotyy`	RF Toolbox
`polar`	RF Toolbox
`semilogx`	RF Toolbox
`semilogy`	RF Toolbox
`smith`	RF Toolbox

`read`	RF Toolbox
`restore`	RF Toolbox

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`	RF Toolbox
`rationalfit`	RF Toolbox
`rfmodel.rational`	RF Toolbox
`timeresp`	RF Toolbox

Functions — By Category

Calculations

deembedsparams	De-embed S-parameters from cascaded network
gamma2z	Convert reflection coefficient to impedance
gammain	Calculate input reflection coefficient of 2-port network
gammaout	Calculate output reflection coefficient of 2-port network
rationalfit	Fit rational function to broadband data
stabilityk	Calculate stability factor *K* of 2-port network
stabilitymu	Calculate stability factor, mu, of 2-port network
vswr	Calculate VSWR at given reflection coefficient gamma

Data Visualization

smithchart	Plot complex vector on Smith chart

Utility Functions

copy	Copy circuit or data object
getdata	Data object containing analyzed result of specified circuit object

Network Parameter Conversion

`abcd2h`	Convert ABCD-parameters to hybrid h-parameters
`abcd2s`	Convert ABCD-parameters to S-parameters
`abcd2y`	Convert ABCD-parameters to Y-parameters
`abcd2z`	Convert ABCD-parameters to Z-parameters
`g2h`	Convert hybrid g-parameters to hybrid h-parameters
`h2abcd`	Convert hybrid h-parameters to ABCD-parameters
`h2g`	Convert hybrid h-parameters to hybrid g-parameters
`h2s`	Convert hybrid h-parameters to S-parameters
`h2y`	Convert hybrid h-parameters to Y-parameters
`h2z`	Convert hybrid h-parameters to Z-parameters
`s2abcd`	Convert S-parameters to ABCD-parameters
`s2h`	Convert S-parameters to hybrid h-parameters
`s2s`	Convert S-parameters to S-parameters with different impedance
`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 2-port, cross mode S-parameters (S_{dc})
s2sdd	Convert 4-port, single-ended S-parameters to 2-port, differential mode S-parameters (S_{dd})
s2t	Convert S-parameters to T-parameters
s2tf	Convert 2-port S-parameters to transfer function
s2y	Convert S-parameters to Y-parameters
s2z	Convert S-parameters to Z-parameters
t2s	Convert T-parameters to S-parameters
y2abcd	Convert Y-parameters to ABCD-parameters
y2h	Convert Y-parameters to hybrid h-parameters
y2s	Convert Y-parameters to S-parameters
y2z	Convert Y-parameters to Z-parameters
z2abcd	Convert Z-parameters to ABCD-parameters
z2h	Convert Z-parameters to hybrid h-parameters

`z2s`	Convert Z-parameters to S-parameters
`z2y`	Convert Z-parameters to Y-parameters

Graphical User Interface

`rftool`	Open RF Analysis Tool (RF Tool)

11

Functions — Alphabetical List

abcd2h

Purpose

Convert ABCD-parameters to hybrid h-parameters

Syntax

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

Description

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

See Also

`abcd2s`	RF Toolbox
`abcd2y`	RF Toolbox
`abcd2z`	RF Toolbox
`h2abcd`	RF Toolbox
`s2h`	RF Toolbox
`y2h`	RF Toolbox
`z2h`	RF Toolbox

Purpose

Convert ABCD-parameters to S-parameters

Syntax

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

Description

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

See Also

`abcd2h`	RF Toolbox
`abcd2y`	RF Toolbox
`abcd2z`	RF Toolbox
`s2abcd`	RF Toolbox
`s2h`	RF Toolbox
`y2h`	RF Toolbox
`z2h`	RF Toolbox

abcd2y

Purpose

Convert ABCD-parameters to Y-parameters

Syntax

```
y_params = abcd2y(abcd_params)
```

Description

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

See Also

`abcd2h`	RF Toolbox
`abcd2s`	RF Toolbox
`abcd2z`	RF Toolbox
`h2y`	RF Toolbox
`s2y`	RF Toolbox
`y2abcd`	RF Toolbox
`z2y`	RF Toolbox

Purpose

Convert ABCD-parameters to Z-parameters

Syntax

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

Description

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

See Also

`abcd2h`	RF Toolbox
`abcd2s`	RF Toolbox
`abcd2y`	RF Toolbox
`h2y`	RF Toolbox
`y2abcd`	RF Toolbox
`z2abcd`	RF Toolbox

Purpose

Copy circuit or data object

Syntax

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

Description

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

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

See Also

`analyze`	RF Toolbox

Purpose De-embed S-parameters from cascaded network

Syntax

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

Description

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

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

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

See Also

`rfckt.cascade`	RF Toolbox
`s2t`	RF Toolbox
`t2s`	RF Toolbox

g2h

Purpose

Convert hybrid g-parameters to hybrid h-parameters

Syntax

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

Description

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

See Also

h2g RF Toolbox

Purpose

Convert reflection coefficient to impedance

Syntax

```
z = gamma2z(gamma,z0)
z = gamma2z(gamma)
```

Description

`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 `z0`.

This conversion is shown in the following equation:

$$Z = Z_0 * \left(\frac{1+\Gamma}{1-\Gamma}\right)$$

`z = gamma2z(gamma)` converts the reflection coefficient `gamma` to the impedance `z` using a reference impedance of 50 ohms.

See Also

`gammain`	RF Toolbox
`gammaout`	RF Toolbox
`vswr`	RF Toolbox

gammain

Purpose Calculate input reflection coefficient of 2-port network

Syntax

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

Description

`result = gammain(s_params,z0,zl)` calculates the input reflection coefficient of a 2-port network as

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

where

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

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

See Also

`gamma2z`	RF Toolbox
`gammaout`	RF Toolbox
`vswr`	RF Toolbox

Purpose

Calculate output reflection coefficient of 2-port network

Syntax

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

Description

`result = gammaout(s_params,z0,zs)` calculates the output reflection coefficient of a 2-port network as

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

where

$$\mathrm{GammaS} = \frac{\mathrm{zs} - \mathrm{z0}}{\mathrm{zs} + \mathrm{z0}}$$

`s_params` 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. `zs` is the source impedance; its default is also 50 ohms. `result` is an `m`-element complex vector.

See Also

`gamma2z`	RF Toolbox
`gammain`	RF Toolbox
`vswr`	RF Toolbox

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 the circuit object `h` has not been analyzed, i.e., there is no analysis data, `getdata` displays an error message.

Note For all circuit objects except those of type `rfckt.amplifier`, `rfckt.datafile`, and `rfckt.mixer`, before calling `getdata`, you must use the `analyze` function to perform a frequency domain analysis for the circuit (`rfckt`) object. When you create an object of type `rfckt.amplifier`, `rfckt.datafile`, or `rfckt.mixer`, by reading data from a file, RF Toolbox automatically creates an `rfdata.data` object and stores data from the file as properties of the data object. You can use the `getdata` function, without first calling `analyze`, to retrieve the handle of this data object.

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.

See Also

`abcd2h`	RF Toolbox
`h2s`	RF Toolbox
`h2y`	RF Toolbox
`h2z`	RF Toolbox
`s2abcd`	RF Toolbox
`y2abcd`	RF Toolbox
`z2abcd`	RF Toolbox

h2g

Purpose

Convert hybrid h-parameters to hybrid g-parameters

Syntax

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

Description

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

See Also

`g2h`	RF Toolbox
`h2abcd`	RF Toolbox
`h2s`	RF Toolbox
`h2y`	RF Toolbox
`h2z`	RF Toolbox

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.

See Also

`abcd2s`	RF Toolbox
`h2abcd`	RF Toolbox
`h2y`	RF Toolbox
`h2z`	RF Toolbox
`y2s`	RF Toolbox
`z2s`	RF Toolbox

h2y

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.

See Also

`abcd2z`	RF Toolbox
`h2abcd`	RF Toolbox
`h2s`	RF Toolbox
`h2y`	RF Toolbox
`s2z`	RF Toolbox
`y2z`	RF Toolbox
`z2h`	RF Toolbox

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.

See Also

`abcd2z`	RF Toolbox
`h2abcd`	RF Toolbox
`h2s`	RF Toolbox
`h2y`	RF Toolbox
`s2z`	RF Toolbox
`y2z`	RF Toolbox
`z2h`	RF Toolbox

Purpose

Fit rational function to broadband data

Syntax

```
h = rationalfit(freq,data)
h = rationalfit(freq,data,tol,weight,delayfactor,diszero,
   npoles)
```

Description

`h = rationalfit(freq,data)` fits a rational function model of the form

$$F(s) = \left(\sum_{k=1}^{n} \frac{C_k}{s - A_k} + D \right) e^{-s*Delay}, \quad s = j2\pi * freq$$

to the complex vector of passive values in `data` over the corresponding frequency values in the positive vector `freq`. The function returns a handle to the rational function model object, `h`, whose properties, `A`, `C`, `D`, and `Delay`, are shown in the preceding equation.

`h = rationalfit(freq,data,tol,weight,delayfactor,diszero,npoles)` fits a rational function to the data using the optional arguments `tol`, `weight`, `delayfactor`, `diszero`, and `npoles` that control the data fitting.

`tol` is a scalar that specifies the relative error-fitting tolerance, in decibels. The relative error of the fit is computed as $\text{relerror} = \frac{\text{norm}(\text{abs}(data - fitdata))}{\text{norm}(\text{abs}(data))}$, where `fitdata` is a vector containing the dependent values of the fit data. The default tolerance is -10 dB. If the model does not fit the original data to within the specified tolerance, a warning message appears.

`weight` is a vector that specifies the weighting of the fit at each frequency. You can increase the weight at a particular frequency to improve the model fitting at that frequency. The length of `weight` must be equal to the length of `freq`. The `weight` vector is empty by default.

`delayfactor` is a scaling factor between 0 and 1 that controls the amount of delay used to fit the data. The `Delay` used to fit the model to the data is `delayfactor` times the ratio of the phase difference of the data across the specified frequencies to the difference between the maximum and minimum frequencies. The default value is 0. This value guarantees that no fitting accuracy is lost due to overestimating the delay. However, you may be able to fit the data accurately with a lower-order model (i.e., a model with fewer poles) by increasing `delayfactor`.

`diszero` is a Boolean value that specifies whether the constant term `D` in the preceding equation is zero or nonzero. A value of 1 indicates that `D` is zero. A value of 0 indicates that `D` is nonzero. The default value of 1 is appropriate for almost every set of data. However, if you are having trouble fitting the data after adjusting the other control arguments, you should change the value of `diszero`.

`npoles` is an even integer or a two-element vector `[M,N]` of even integers.

- If `npoles` is an integer, it specifies the number of poles (`k` in the previous equation) to use to fit the rational function to the data.
- If `npoles` is a vector, it specifies a range of values of the number of poles, `k`, to use in trying to fit the data. `rationalfit` first tries to fit the data using `M` poles. If the fit error using `M` poles is greater than `tol`, `rationalfit` increases the number of poles in the fit until the error is less than `tol` or the number of poles reaches `N`.

Specifying `npoles` can speed up the fitting process because `rationalfit` does not spend time trying to fit a model with an unreasonably small number of poles to the data. As a rule of thumb, you should specify a value of `npoles` greater than or equal to twice the number of peaks that can be readily observed by plotting the data in the frequency domain. The default is [4, MAX], where MAX is either one quarter the number of frequency samples or 256, whichever is smaller.

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. See the `freqresp` reference page or the examples in the next section for more information.

Examples

The following example shows how to fit a rational function model to data from the `passive.s2p` file and how to generate plots that compare the frequency response of the original data to that of the fit data.

```
orig_data = read(rfdata.data,'passive.s2p')
freq = orig_data.Freq;
data = orig_data.S_Parameters(1,1,:);
fit_data = rationalfit(freq,data)
[resp,freq] = freqresp(fit_data,freq);

plot(orig_data,'S11','dB');
hold on
plot(freq/1e9,db(resp));

figure

plot(orig_data,'S11','Angle (radians)');
hold on
plot(freq/1e9,unwrap(angle(resp)));
```

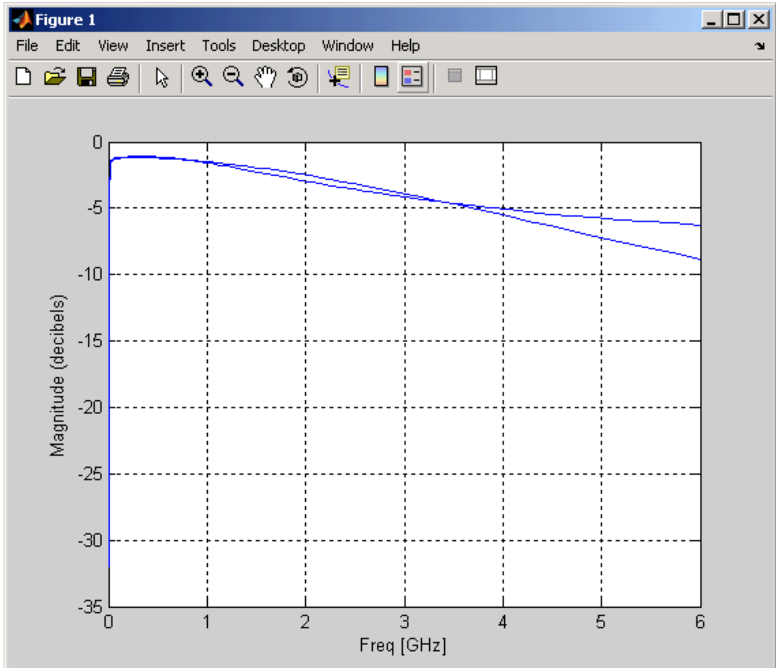
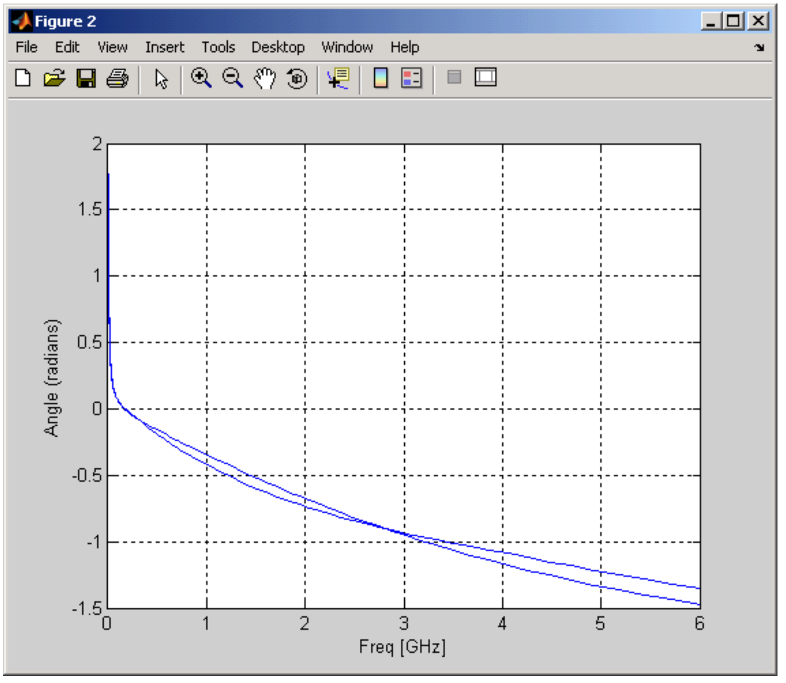


Figure 1
File Edit View Insert Tools Desktop Window Help
Magnitude (decibels)
Freq [GHz]
0
-5
-10
-15
-20
-25
-30
-35
0
1
2
3
4
5
6



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`	RF Toolbox
`rfmodel.rational`	RF Toolbox
`s2tf`	RF Toolbox

`timeresp`	RF Toolbox
`writeva`	RF Toolbox

Purpose

Open RF Analysis Tool (RF Tool)

Syntax

```
rftool
```

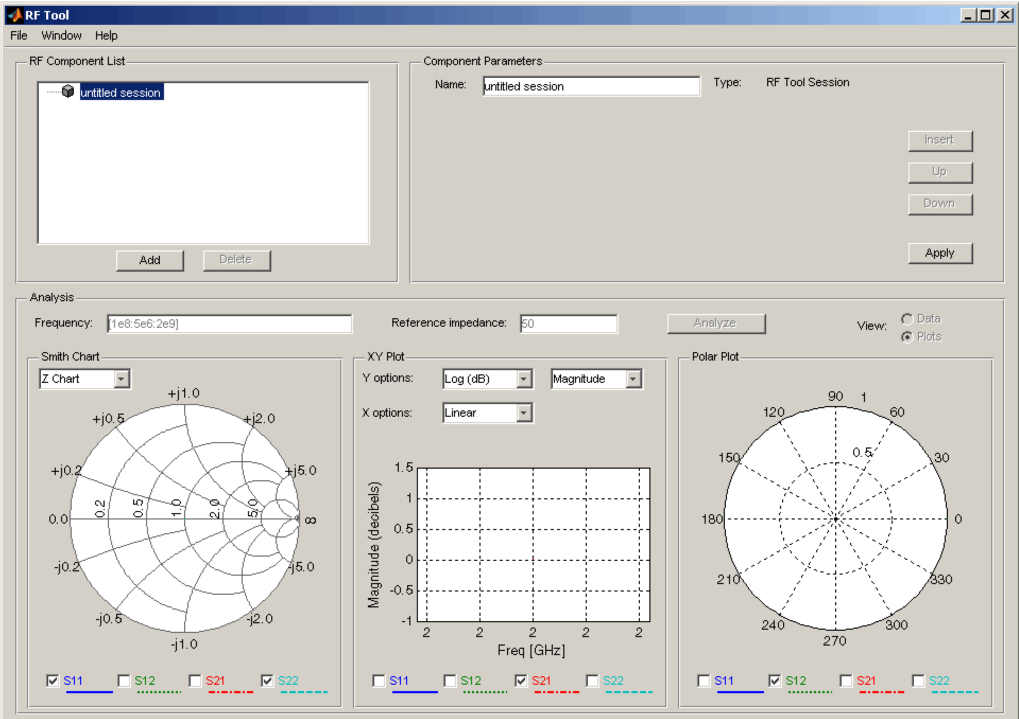
Description

`rftool` opens RF Tool. 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.

See Chapter 5, "RF Tool: An RF Analysis GUI" for more information.

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

s2abcd

Purpose

Convert S-parameters to ABCD-parameters

Syntax

```
abcd_params = s2abcd(s_params,z0)
```

Description

abcd_params = s2abcd(s_params,z0) converts the scattering parameters s_params into the ABCD-parameters abcd_params. The s_params input is a complex 2-by-2-by-m array, representing m 2-port S-parameters. z0 is the reference impedance; its default is 50 ohms. abcd_params is a complex 2-by-2-by-m array, representing m 2-port ABCD-parameters.

See Also

abcd2s	RF Toolbox
h2abcd	RF Toolbox
s2h	RF Toolbox
s2y	RF Toolbox
s2z	RF Toolbox
y2abcd	RF Toolbox
z2abcd	RF Toolbox

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.

See Also

`abcd2h`	RF Toolbox
`h2s`	RF Toolbox
`s2abcd`	RF Toolbox
`s2y`	RF Toolbox
`s2z`	RF Toolbox
`y2h`	RF Toolbox
`z2h`	RF Toolbox

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`.

See Also

`abcd2s`	RF Toolbox
`h2s`	RF Toolbox
`s2abcd`	RF Toolbox
`s2h`	RF Toolbox
`s2y`	RF Toolbox
`s2z`	RF Toolbox
`y2s`	RF Toolbox
`z2s`	RF Toolbox

Purpose

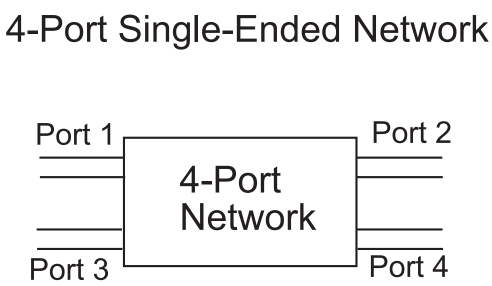
Convert 4-port, single-ended S-parameters to 2-port, common mode S-parameters (S_{cc})

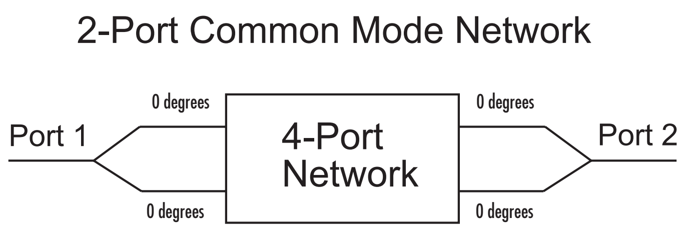
Syntax

```
scc_params = s2scc(s_params)
```

Description

`scc_params = s2scc(s_params)` converts the 4-port, single-ended S-parameters, `s_params`, to 2-port, common mode S-parameters, `scc_params`. `scc_params` is a complex 2-by-2-by-M array that represents M 2-port S-parameters. `s_params` is a complex 4-by-4-by-M array that represents M 4-port S-parameters.





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`	RF Toolbox
`s2sdc`	RF Toolbox
`s2sdd`	RF Toolbox

Purpose

Convert 4-port, single-ended S-parameters to 2-port, cross mode S-parameters (S_{cd})

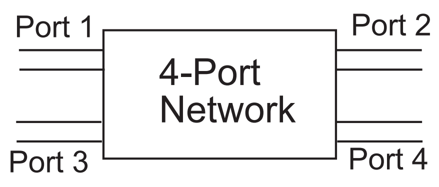
Syntax

```
scd_params = s2scd(s_params)
```

Description

`scd_params = s2scd(s_params)` converts the 4-port, single-ended S-parameters, `s_params`, to 2-port, cross mode S-parameters, `scd_params`. `scd_params` is a complex 2-by-2-by-M array that represents M 2-port cross mode S-parameters (S_{cd}). `s_params` is a complex 4-by-4-by-M array that represents M 4-port S-parameters.

4-Port Single-Ended Network



2-Port Cross Mode Network

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	RF Toolbox
s2sdc	RF Toolbox
s2sdd	RF Toolbox

Purpose

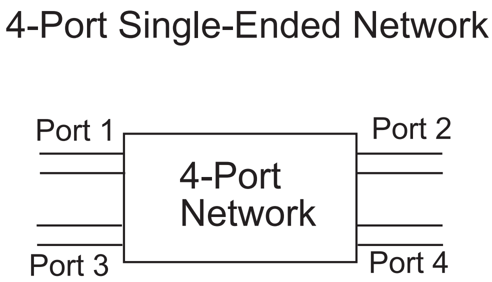
Convert 4-port, single-ended S-parameters to 2-port, cross mode S-parameters (S_{dc})

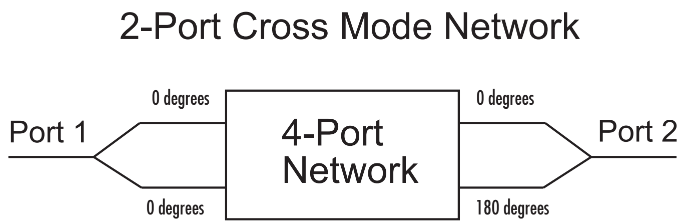
Syntax

```
sdc_params = s2sdc(s_params)
```

Description

`sdc_params = s2sdc(s_params)` converts the 4-port, single-ended S-parameters, `s_params`, to 2-port, cross mode S-parameters, `sdc_params`. `sdc_params` is a complex 2-by-2-by-M array that represents M 2-port cross mode S-parameters (S_{dc}). `s_params` is a complex 4-by-4-by-M array that represents M 4-port S-parameters.





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`	RF Toolbox
`s2scd`	RF Toolbox
`s2sdd`	RF Toolbox

Purpose

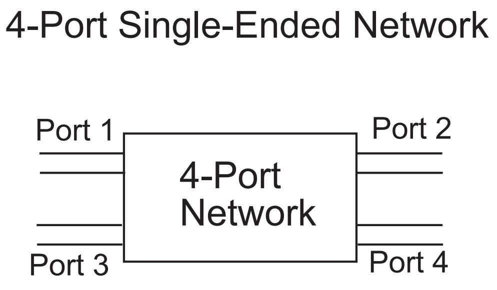
Convert 4-port, single-ended S-parameters to 2-port, differential mode S-parameters (S_{dd})

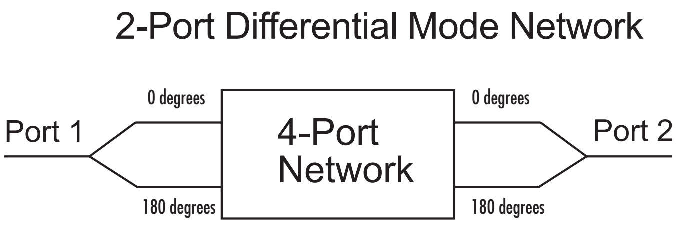
Syntax

```
sdd_params = s2sdd(s_params)
```

Description

`sdd_params = s2sdd(s_params)` converts the 4-port, single-ended S-parameters, `s_params`, to 2-port, differential mode S-parameters, `sdd_params`. `sdd_params` is a complex 2-by-2-by-M array that represents M 2-port differential mode S-parameters. `s_params` is a complex 4-by-4-by-M array that represents M 4-port S-parameters.





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`	RF Toolbox
`s2scd`	RF Toolbox
`s2sdc`	RF Toolbox

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.

See Also

`s2abcd`	RF Toolbox
`s2h`	RF Toolbox
`s2y`	RF Toolbox
`s2z`	RF Toolbox
`t2s`	RF Toolbox

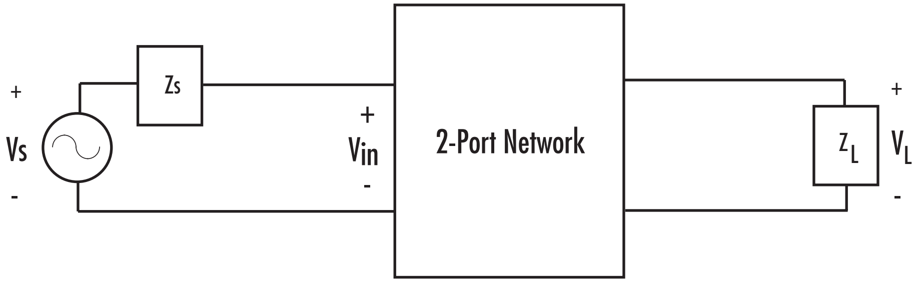
Purpose Convert 2-port S-parameters to transfer function

Syntax

```
tf = s2tf(s_params, z0, zs, zl,option)
```

Description

`tf = s2tf(s_params, z0, zs, zl,option)` converts the 2-port scattering parameters, `s_params`, into a transfer function that represents the normalized voltage gain of a 2-port network. The following figure shows the impedances and voltages that are used to define the gain.



The impedances shown in the figure are optional arguments to the `s2tf` function and are defined as follows:

- `z0` is the reference impedance of the S-parameters.
- `zs` is the source impedance.
- `zl` is the load impedance.

The default value of these impedances is 50 ohms.

The voltages in the figure are defined as follows:

- V_L is the output voltage over the load impedance.
- V_S is the source voltage.
- V_{in} is the input voltage when the input impedance of the 2-port network matches the source impedance. That is, $V_{in}=V_S/2$.

The definition of the transfer function is determined by the optional `option` argument.

`option` can be

- `1` — The transfer function is the gain from the input voltage to the output voltage:

  $$tf = \frac{V_L}{V_{in}} = \frac{S_{21} * (1 + \Gamma_l) * (1 - \Gamma_s)}{(1 - S_{22} * \Gamma_l)(1 - \Gamma_{in} * \Gamma_s)}$$

  where

  $$\Gamma_l = \frac{Z_l - Z_o}{Z_l + Z_o}$$

  $$\Gamma_s = \frac{Z_s - Z_o}{Z_s + Z_o}$$

  $$\Gamma_{in} = S_{11} + \left(S_{12} * S_{21} * \frac{\Gamma_l}{(1 - S_{22} * \Gamma_l)} \right)$$

- `2` — The transfer function is the gain from the source voltage to the output voltage:

  $$tf = \frac{V_L}{V_S} = \frac{S_{21} * (1 + \Gamma_l) * (1 - \Gamma_s)}{2 * (1 - S_{22} * \Gamma_l)(1 - \Gamma_{in} * \Gamma_s)}$$

The default value of `option` is `1`.

See Also

`rationalfit`	RF Toolbox
`s2scc`	RF Toolbox

`s2scd`	RF Toolbox
`s2sdc`	RF Toolbox
`s2sdd`	RF Toolbox

Purpose

Convert S-parameters to Y-parameters

Syntax

```
y_params = s2y(s_params,z0)
```

Description

`y_params = s2y(s_params,z0)` converts the scattering parameters `s_params` into the admittance parameters `y_params`. The `s_params` input is a complex `n`-by-`n`-by-`m` array, representing `m` n-port S-parameters. `z0` is the reference impedance; its default is 50 ohms. `y_params` is a complex `n`-by-`n`-by-`m` array, representing `m` n-port Y-parameters.

See Also

`abcd2y`	RF Toolbox
`h2y`	RF Toolbox
`s2abcd`	RF Toolbox
`s2h`	RF Toolbox
`s2z`	RF Toolbox
`y2s`	RF Toolbox
`z2y`	RF Toolbox

Purpose

Convert S-parameters to Z-parameters

Syntax

```
z_params = s2z(s_params,z0)
```

Description

`z_params = s2z(s_params,z0)` converts the scattering parameters `s_params` into the impedance parameters `z_params`. The `s_params` input is a complex `n`-by-`n`-by-`m` array, representing `m` n-port S-parameters. `z0` is the reference impedance; its default is 50 ohms. `z_params` is a complex `n`-by-`n`-by-`m` array, representing `m` n-port Z-parameters.

See Also

`abcd2z`	RF Toolbox
`h2z`	RF Toolbox
`s2abcd`	RF Toolbox
`s2h`	RF Toolbox
`s2y`	RF Toolbox
`y2z`	RF Toolbox
`z2s`	RF Toolbox

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 and returns

- `hsm`, which is the handle of the Smith chart object.

  Change the properties of the chart by changing the `smithchart` object properties described in the next section.

- `lineseries`, which is a column vector of handles to `lineseries` objects, one handle per plotted line.

  Change the properties of the plotted lines by changing the `lineseries properties`.

`hsm = smithchart` draws a blank Smith chart and returns the handle `hsm` of the Smith chart object.

Note To plot multiple sets of data on a Smith chart, use the following syntax:

```
[lineseries1,hsm] = smithchart(y)
hold on
lineseries2 = smithchart(z)
```

You can use change the properties of the lines, `lineseries1` and `lineseries2`, and of the properties of the chart, `hsm`.

Note To plot network parameters from a circuit (`rfckt`) or data (`rfdata`) object on a Smith chart, use the `smith` function.

Properties

smithchart creates the plot using default property values of a Smith chart. Use set(h,'PropertyName1',PropertyValue1,...) to change the property values. Use get(h) to get the property values.

This table lists all properties you can specify for smithchart objects along with units, valid values, and a descriptions of their use.

Property Name	Description	Units, Values
Color	Line color for a Z or Y Smith chart. For a ZY Smith chart, the Z line color.	ColorSpec. Default is [0.4 0.4 0.4] (dark 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.

Property Name	Description	Units, Values
`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`.
`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

`get`	RF Toolbox
`set`	RF Toolbox
`smith`	RF Toolbox

stabilityk

Purpose

Calculate stability factor K of 2-port network

Syntax

```
[k,b1,b2,delta] = stabilityk(s_params)
```

Description

`[k,b1,b2,delta] = stabilityk(s_params)` calculates and returns the stability factor `k`, as well as the conditions `b1`, `b2`, and `delta` for stability of a 2-port network. The input `s_params` is a complex 2-by-2-by-`m` array, representing `m` 2-port S-parameters.

$$K = 1 - |S_{11}|^2 - |S_{22}|^2 + |\Delta|^2 / (2|S_{12}S_{21}|)$$

$$B_1 = 1 + |S_{11}|^2 - |S_{22}|^2 - |\Delta|^2$$

$$B_2 = 1 - |S_{11}|^2 + |S_{22}|^2 - |\Delta|^2$$

where

- S_{11}, S_{12}, S_{21}, and S_{22} are vectors of S-parameters, taken from the input argument `s_params`.
- $\Delta = S_{11}S_{22} - S_{12}S_{21}$

References

Gonzalez, Guillermo, *Microwave Transistor Amplifiers: Analysis and Design*, 2nd edition, Prentice-Hall, pp. 217-228, 1997.

See Also

`stabilitymu` RF Toolbox

Purpose

Calculate stability factor, mu, 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.

$$\mu = (1-|S_{11}|^2)/(|S_{22}-S_{11}^*\Delta| + |S_{21}S_{12}|)$$

$$\mu' = (1-|S_{22}|^2)/(|S_{11}-S_{22}^*\Delta| + |S_{21}S_{12}|)$$

where

- S_{11}, S_{12}, S_{21}, and S_{22} are vectors of S-parameters, taken from the input argument `s_params`.
- $\Delta = S_{11}S_{22} - S_{12}S_{21}$
- S^* is the complex conjugate of the designated S-parameter.

μ defines the minimum distance between the center of the unit Smith chart and the unstable region in the load plane (the load is considered port 2).

μ' defines the minimum distance between the center of the unit Smith chart and the unstable region in the source plane (the source is considered port 1).

Having $\mu > 1$ (or $\mu' > 1$) is necessary and sufficient for the 2-port linear network, described by the S-parameters, to be unconditionally stable.

References

Edwards, Marion Lee, and Jeffrey H. Sinsky, "A New Criterion for Linear 2-Port Stability Using a Single Geometrically Derived Parameter," *IEEE Transactions on Microwave Theory and Techniques*, Vol. 40, No. 12, pp. 2303-2311, December 1992.

See Also

`stabilityk`	RF Toolbox

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.

See Also

`abcd2s`	RF Toolbox
`h2s`	RF Toolbox
`s2t`	RF Toolbox
`y2s`	RF Toolbox
`z2s`	RF Toolbox

Purpose

Calculate VSWR at given reflection coefficient gamma

Syntax

```
result = vswr(gamma)
```

Description

`result = vswr(gamma)` calculates the voltage standing-wave ratio (VSWR) at the given reflection coefficient gamma as

$$\text{VSWR} = \frac{1+|\Gamma|}{1-|\Gamma|}$$

where Γ is the given reflection coefficient `gamma`. The input `gamma` is a complex vector. `result` is a real vector of the same length as `gamma`.

See Also

`gamma2z`	RF Toolbox
`gammain`	RF Toolbox
`gammaout`	RF Toolbox

Purpose

Convert Y-parameters to ABCD-parameters

Syntax

```
abcd_params = y2abcd(y_params)
```

Description

abcd_params = y2abcd(y_params) converts the admittance parameters y_params into the ABCD-parameters abcd_params. The y_params input is a complex 2-by-2-by-m array, representing m 2-port Y-parameters. abcd_params is a complex 2-by-2-by-m array, representing m 2-port ABCD-parameters.

See Also

abcd2y	RF Toolbox
h2abcd	RF Toolbox
s2abcd	RF Toolbox
y2h	RF Toolbox
y2s	RF Toolbox
y2z	RF Toolbox
z2abcd	RF Toolbox

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.

See Also

`abcd2h`	RF Toolbox
`h2y`	RF Toolbox
`s2h`	RF Toolbox
`y2abcd`	RF Toolbox
`y2s`	RF Toolbox
`y2z`	RF Toolbox
`z2h`	RF Toolbox

Purpose

Convert Y-parameters to S-parameters

Syntax

```
s_params = y2s(y_params,z0)
```

Description

`s_params = y2s(y_params,z0)` converts the admittance parameters `y_params` into the scattering parameters `s_params`. The `y_params` input is a complex `n`-by-`n`-by-`m` array, representing `m` `n`-port Y-parameters. `z0` is the reference impedance; its default is 50 ohms. `s_params` is a complex `n`-by-`n`-by-`m` array, representing `m` `n`-port S-parameters.

See Also

`abcd2s`	RF Toolbox
`h2s`	RF Toolbox
`s2y`	RF Toolbox
`y2abcd`	RF Toolbox
`y2h`	RF Toolbox
`y2s`	RF Toolbox
`y2z`	RF Toolbox
`z2s`	RF Toolbox

Purpose

Convert Y-parameters to Z-parameters

Syntax

```
z_params = y2z(y_params)
```

Description

`z_params = y2z(y_params)` converts the admittance parameters `y_params` into the impedance parameters `z_params`. The `y_params` input is a complex `n`-by-`n`-by-`m` array, representing `m` `n`-port Y-parameters. `z_params` is a complex `n`-by-`n`-by-`m` array, representing `m` `n`-port Z-parameters.

See Also

`abcd2z`	RF Toolbox
`h2z`	RF Toolbox
`y2abcd`	RF Toolbox
`y2h`	RF Toolbox
`y2s`	RF Toolbox
`y2z`	RF Toolbox
`z2s`	RF Toolbox
`z2y`	RF Toolbox

Purpose

Convert Z-parameters to ABCD-parameters

Syntax

```
abcd_params = z2abcd(z_params)
```

Description

`abcd_params = z2abcd(z_params)` converts the impedance parameters `z_params` into the ABCD-parameters `abcd_params`. The `z_params` input is a complex 2-by-2-by-`m` array, representing `m` 2-port Z-parameters. `abcd_params` is a complex 2-by-2-by-`m` array, representing `m` 2-port ABCD-parameters.

See Also

`abcd2z`	RF Toolbox
`h2abcd`	RF Toolbox
`s2abcd`	RF Toolbox
`y2abcd`	RF Toolbox
`z2h`	RF Toolbox
`z2s`	RF Toolbox
`z2y`	RF Toolbox

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.

See Also

`abcd2h`	RF Toolbox
`h2z`	RF Toolbox
`s2h`	RF Toolbox
`y2h`	RF Toolbox
`z2abcd`	RF Toolbox
`z2s`	RF Toolbox
`z2y`	RF Toolbox

Purpose

Convert Z-parameters to S-parameters

Syntax

```
s_params = z2s(z_params,z0)
```

Description

`s_params = z2s(z_params,z0)` converts the impedance parameters `z_params` into the scattering parameters `s_params`. The `z_params` input is a complex `n`-by-`n`-by-`m` array, representing `m` `n`-port Z-parameters. `z0` is the reference impedance; its default is 50 ohms. `s_params` is a complex `n`-by-`n`-by-`m` array, representing `m` `n`-port S-parameters.

See Also

`abcd2s`	RF Toolbox
`h2s`	RF Toolbox
`s2z`	RF Toolbox
`y2s`	RF Toolbox
`z2abcd`	RF Toolbox
`z2h`	RF Toolbox
`z2y`	RF Toolbox

Purpose Convert Z-parameters to Y-parameters

Syntax

```
y_params = z2y(z_params)
```

Description

`y_params = z2y(z_params)` converts the impedance parameters `z_params` into the admittance parameters `y_params`. The `z_params` input is a complex `n`-by-`n`-by-`m` array, representing `m` n-port Z-parameters. `y_params` is a complex `n`-by-`n`-by-`m` array, representing `m` n-port Y-parameters.

See Also

`abcd2y`	RF Toolbox
`h2y`	RF Toolbox
`s2y`	RF Toolbox
`y2z`	RF Toolbox
`z2abcd`	RF Toolbox
`z2h`	RF Toolbox
`z2s`	RF Toolbox

AMP File Format

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 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-14.

Two AMP files, `samplepa1.amp` and `default.amp`, ship with RF Toolbox to show the AMP format. They describe a nonlinear 2-port amplifier with noise. See "Example — Modeling 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 "Data Sections" on page A-4. 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.

This section contains the following topics:

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-10 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 does not use the phase data directly. RF Blockset uses this data in conjunction with RF Toolbox to create the AM/PM conversion table for the General Amplifier and General Mixer blocks.

If all phases are zero, you can omit the third column. If all phases are zero or omitted, RF 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, RF 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-10 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

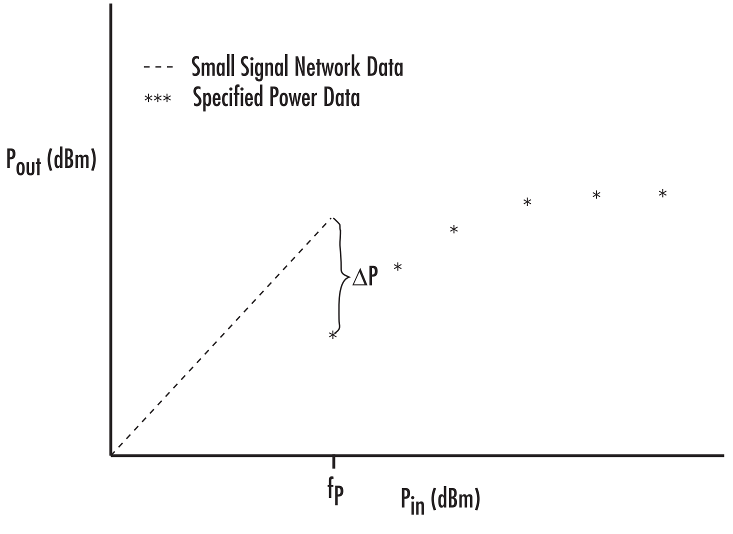
If an AMP file contains both network parameter data and power data, RF Toolbox checks the data for consistency.

RF 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:

$$\Delta P = S_{21}(f_P) - P_{out}(f_P) + P_{in}(f_P) \quad (dBm)$$

where f_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.

B

Examples

Use this list to find examples in the documentation.

Modeling a Cascaded RF Network

Modeling a Transmission Line

Working with Objects

Modeling an RF Network Using RF Tool

Index

D

E

F

G

H

I

L

O

P

R

S