RF Toolbox™

User's Guide

MATLAB®

R2018a

MathWorks®

How to Contact MathWorks

Latest news:	www.mathworks.com
Sales and services:	www.mathworks.com/sales_and_services
User community:	www.mathworks.com/matlabcentral
Technical support:	www.mathworks.com/support/contact_us
Phone:	508-647-7000

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

RF Toolbox™ User's Guide

Revision History

June 2004	Online only	New for Version 1.0 (Release 14)
August 2004	Online only	Revised for Version 1.0.1 (Release 14+)
March 2005	Online only	Revised for Version 1.1 (Release 14SP2)
September 2005	Online only	Revised for Version 1.2 (Release 14SP3)
March 2006	Online only	Revised for Version 1.3 (Release 2006a)
September 2006	Online only	Revised for Version 2.0 (Release 2006b)
March 2007	Online only	Revised for Version 2.1 (Release 2007a)
September 2007	Online only	Revised for Version 2.2 (Release 2007b)
March 2008	Online only	Revised for Version 2.3 (Release 2008a)
October 2008	Online only	Revised for Version 2.4 (Release 2008b)
March 2009	Online only	Revised for Version 2.5 (Release 2009a)
September 2009	Online only	Revised for Version 2.6 (Release 2009b)
March 2010	Online only	Revised for Version 2.7 (Release 2010a)
September 2010	Online only	Revised for Version 2.8 (Release 2010b)
April 2011	Online only	Revised for Version 2.8.1 (Release 2011a)
September 2011	Online only	Revised for Version 2.9 (Release 2011b)
March 2012	Online only	Revised for Version 2.10 (Release 2012a)
September 2012	Online only	Revised for Version 2.11 (Release 2012b)
March 2013	Online only	Revised for Version 2.12 (Release 2013a)
September 2013	Online only	Revised for Version 2.13 (Release 2013b)
March 2014	Online only	Revised for Version 2.14 (Release 2014a)
October 2014	Online only	Revised for Version 2.15 (Release 2014b)
March 2015	Online only	Revised for Version 2.16 (Release 2015a)
September 2015	Online only	Revised for Version 2.17 (Release 2015b)
March 2016	Online only	Revised for Version 3.0 (Release 2016a)
September 2016	Online only	Revised for Version 3.1 (Release 2016b)
March 2017	Online only	Revised for Version 3.2 (Release 2017a)
September 2017	Online only	Revised for Version 3.3 (Release 2017b)
March 2018	Online only	Revised for Version 3.4 (Release 2018a)

Contents

1 Getting Started

2 RF Objects

3 Model an RF Component

4 Export Verilog-A Models

5 The RF Design and Analysis App

6 Objects — Alphabetical List

7 Methods — Alphabetical List

8 Functions — Alphabetical List

9 AMP File Format

Getting Started

RF Toolbox Product Description

Design, model, and analyze networks of RF components

RF Toolbox provides functions, objects, and apps for designing, modeling, analyzing, and visualizing networks of radio frequency (RF) components. You can use RF Toolbox for wireless communications, radar, and signal integrity projects.

With RF Toolbox you can build networks of RF components such as filters, transmission lines, amplifiers, and mixers. Components can be specified using measurement data, network parameters, or physical properties. You can calculate S-parameters, convert among S, Y, Z, ABCD, h, g, and T network parameters, and visualize RF data using rectangular and polar plots and Smith® Charts.

The RF Budget Analyzer app lets you analyze transmitters and receivers in terms of noise figure, gain, and IP3. You can generate RF Blockset™ testbenches and validate analytical results against circuit envelope simulations.

Using the rational function fitting method, you can build models of backplanes and interconnects, and export them as Simulink® blocks or as Verilog-A modules for SerDes design.

RF Toolbox provides functions to manipulate and automate RF measurement data analysis, including de-embedding, enforcing passivity, and computing group delay.

Key Features

- RF filters, transmission lines, amplifiers, and mixers specified by measurement data, network parameters, or physical properties
- S-parameter calculation for RF component networks
- RF Budget Analyzer app for calculating noise figure, gain, and IP3 of RF transceivers and for generating RF Blockset testbenches
- Rational function fitting method for building models and exporting them as Simulink blocks or Verilog-A modules
- De-embedding of N-port S-parameters measurement data
- Conversion among S, Y, Z, ABCD, h, g, and T network parameters
- RF data visualization using rectangular and polar plots and Smith Charts

Related Products

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

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

RF Objects

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

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

Object Type	Name	Description
"RF Data Objects" on page 2-2	`rfdata`	Stores data for use by other RF objects or for plotting and network parameter conversion.
"RF Circuit Objects" on page 2-4	`rfckt`	Represents RF components and networks using network parameters and physical properties for frequency-domain simulation.
"RF Model Objects" on page 2-9	`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 "Create RF Objects" on page 3-2.

You use a different form of dot notation to specify object properties, as described in "Reference Properties Directly Using Dot Notation" on page 3-14. This is just one way to define component data. For more information on object properties, see "Specify or Import Component Data" on page 3-5.

You use object methods to perform frequency-domain analysis and visualize the results. For more information, see “Analyze and Plot RF Components” on page 3-24.

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

S-Parameter Notation

In this section...

Define S-Parameters

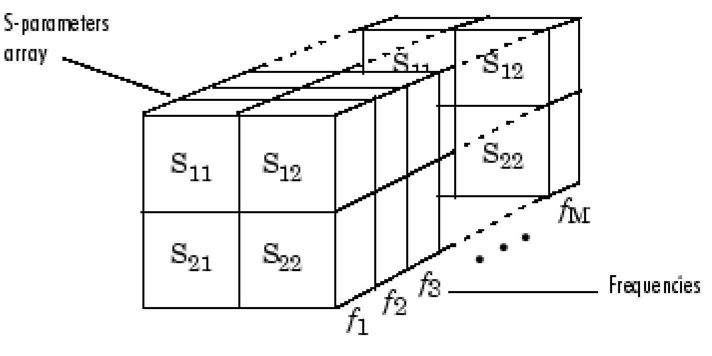
RF Toolbox software uses matrix notation to specify S-parameters. The indices of an S-parameter matrix correspond to the port numbers of the network that the data represent. For example, to define a matrix of 50-ohm, 2-port S-parameters, type:

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

RF Toolbox functions that operate on `s_params` assume:

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

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



Refer to S-Parameters Using Character Vector

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

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

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

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

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

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

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

  See “Extract M-Port S-Parameters from N-Port S-Parameters” on page 3-19.
- Convert single-ended 4-port parameters to differential 2-port parameters.

  See “Convert Single-Ended S-Parameters to Mixed-Mode S-Parameters” on page 3-18.

RF Analysis

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

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

   See "Create RF Objects" on page 3-2.
2. Define component data by:

   - Specifying network parameters or physical properties (see "Set Property Values" on page 3-5).
   - Importing data from an industry-standard Touchstone file, a MathWorks AMP file, an Agilent® P2D or S2D file, or the MATLAB workspace (see "Import Property Values from Data Files" on page 3-8).
   - Where applicable, selecting operating condition values (see "Specify Operating Conditions" on page 3-16).
3. Where applicable, perform network parameter conversions on imported file data.

   See "Process File Data for Analysis" on page 3-18.
4. Integrate components to form a cascade, hybrid, parallel, or series network.

   See "Construct Networks of Specified Components" on page 3-7.
5. Analyze the network in the frequency domain.

   See "Analyze Networks in the Frequency Domain" on page 3-24.
6. Generate plots to gain insight into network behavior.

   The following plots and charts are available in the toolbox:

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

   See "Visualize Component and Network Data" on page 3-24.
7. Compute the network transfer function.

See “Compute the Network Transfer Function” on page 3-34.

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

See “Fit a Model Object to Circuit Object Data” on page 3-34.

9 Plot the time-domain response.

See “Compute and Plot the Time-Domain Response” on page 3-35.

10 Export a Verilog-A description of the network.

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

Model a Cascaded RF Network

Overview

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

Note To learn how to use RF Design and Analysis App, to perform these tasks, see “Model an RF Network” on page 5-31.

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

Create RF Components

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

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

Specify Component Data

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

- "Transmission Line Properties" on page 1-11
- "Amplifier Properties" on page 1-11

Transmission Line Properties

1. Type the following command at the MATLAB prompt to change the line length of the first transmission line, `FirstCkt`, to `12`:

   ```
   FirstCkt.LineLength = 12;
   ```

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`:

   ```
   ThirdCkt.LineLength = 0.025;
   ThirdCkt.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`:

   ```
   SecondCkt.IntpType = 'cubic';
   ```

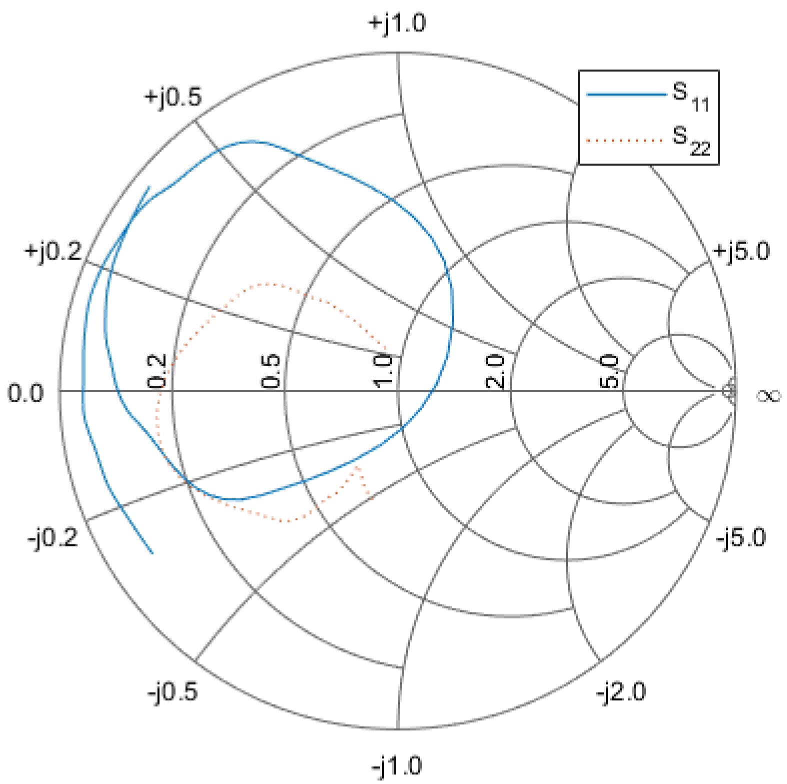
   The `IntpType` property tells the toolbox how to interpolate the network parameters, noise data, and power data when you analyze the amplifier at frequencies other than those specified in the file.

Validate RF Components

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

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

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

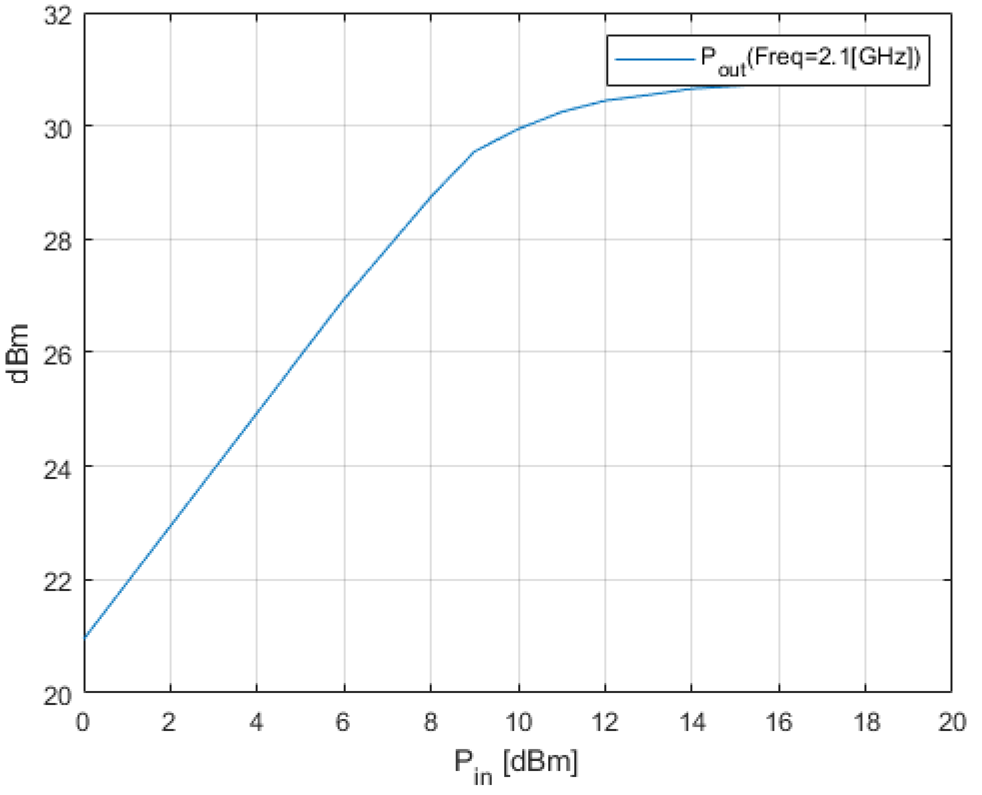



```
legend show
```

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

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

```
figure
plot(SecondCkt,'Pout','dBm')
```




```
legend show
```

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

Build and Simulate the Network

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

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

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

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

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

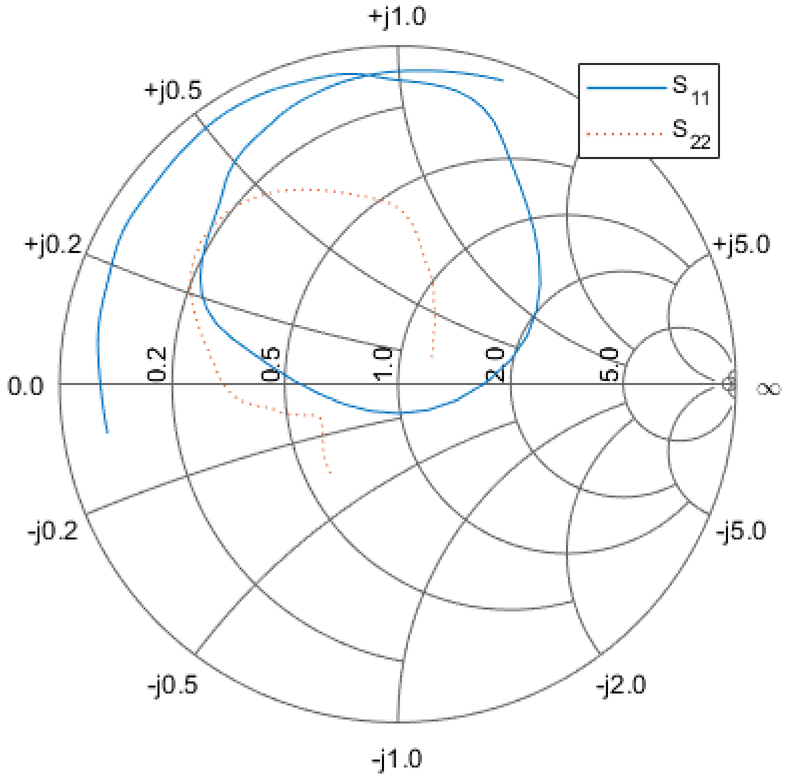
```
f = (1.0e9:1e7:2.9e9);
analyze(CascadedCkt,f);
```

Analyze Simulation Results

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

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

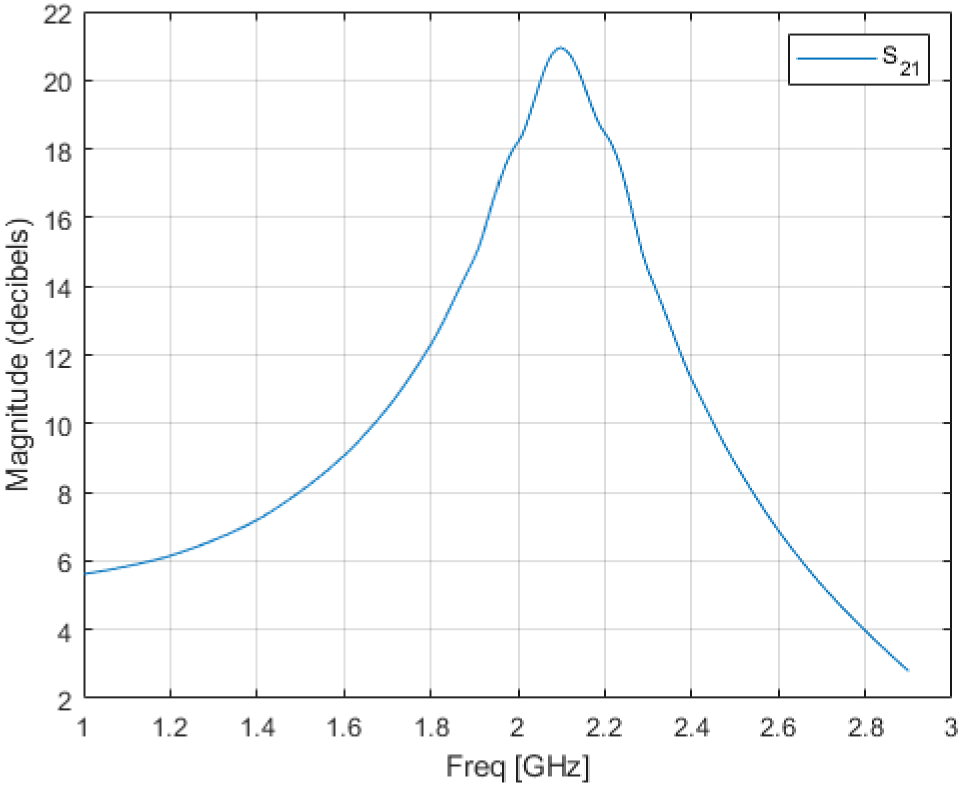
```
figure
lineseries2 = smith(CascadedCkt,'S11','S22','z');
lineseries2(1).LineStyle ='-';
lineseries2(1).LineWidth =1;
lineseries2(2).LineStyle = ':';
lineseries2(2).LineWidth = 1;
```




```
legend show
```

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

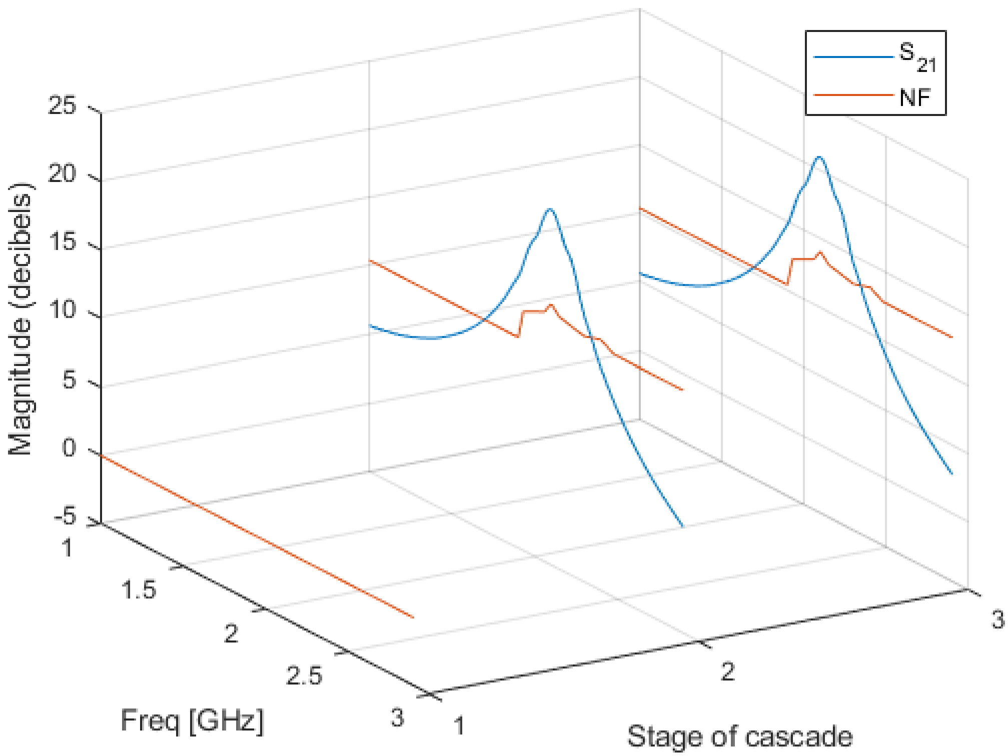
```
figure
plot(CascadedCkt,'S21','dB')
```




```
legend show
```

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

```
figure
plot(CascadedCkt,'budget', 'S21','NF')
```




```
legend show
```

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

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

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

Analyze a Transmission Line

In this section...

Overview

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

Build and Simulate the Transmission Line

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

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

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

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

Compute the Transmission Line Transfer Function and Time-Domain Response

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

- "Calculate the Transfer Function" on page 1-19
- "Fit and Validate the Transfer Function Model" on page 1-19

- “Compute and Plot the Time-Domain Response” on page 1-21

Calculate the Transfer Function

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

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

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

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

Fit and Validate the Transfer Function Model

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

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

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

```
RationalFunc =
   rfmodel.rational with properties:

        A: [7x1 double]
        C: [7x1 double]
        D: 0
    Delay: 0
     Name: 'Rational Function'
```

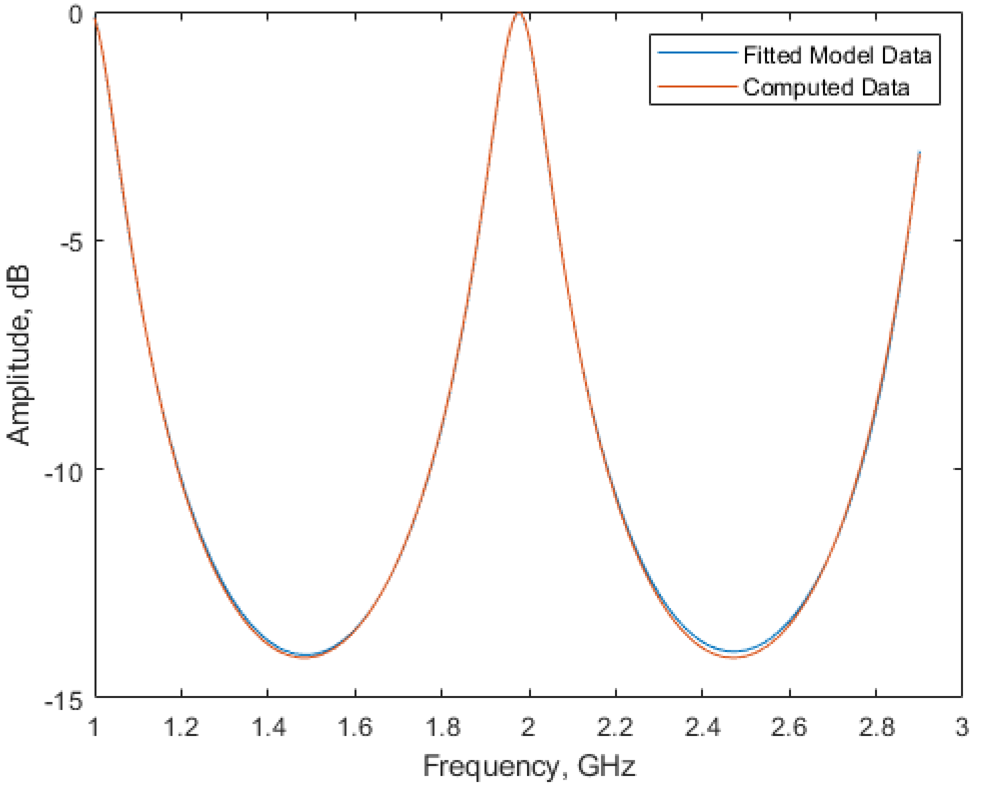
2. Type the following command at the MATLAB prompt to compute the frequency response of the fitted model data:

```
[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/1e9,20*log10(abs(fresp)),freq/1e9,20*log10(abs(TrFunc)))
xlabel('Frequency, GHz')
```

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

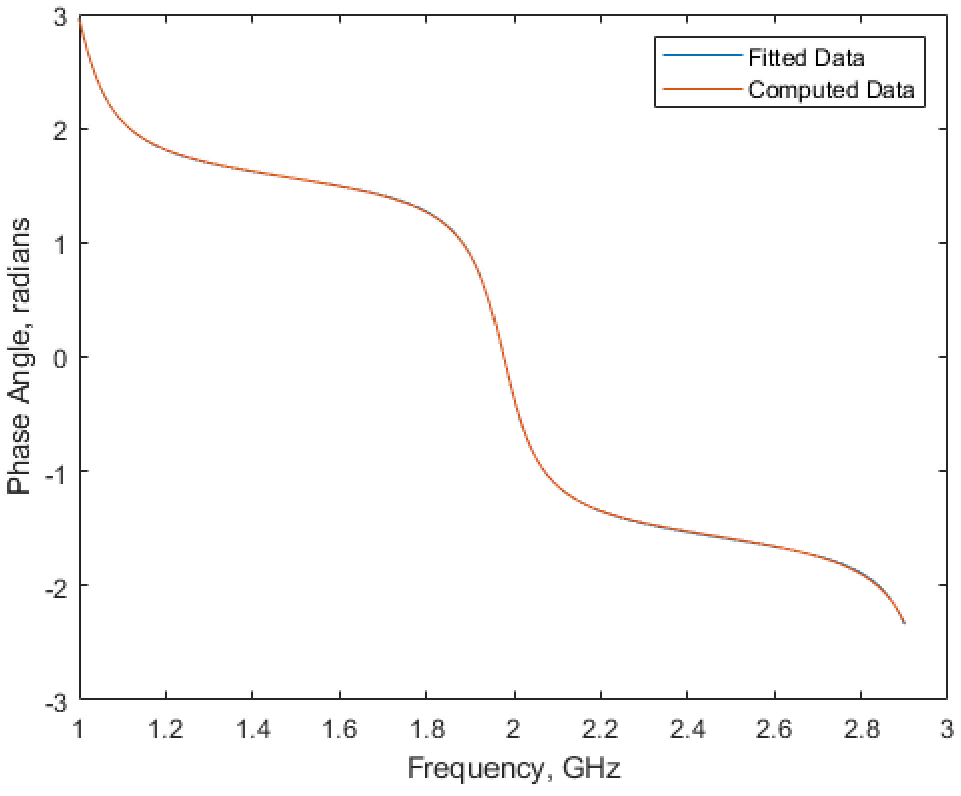


Note The amplitude of the model data is very close to the amplitude of the computed data. You can control the tradeoff between model accuracy and model complexity by specifying the optional tolerance argument, `tol`, to the `rationalfit` function, as described in “Represent a Circuit Object with a Model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

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

```
figure
plot(freq/1e9,unwrap(angle(fresp)),...
```

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



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

Compute and Plot the Time-Domain Response

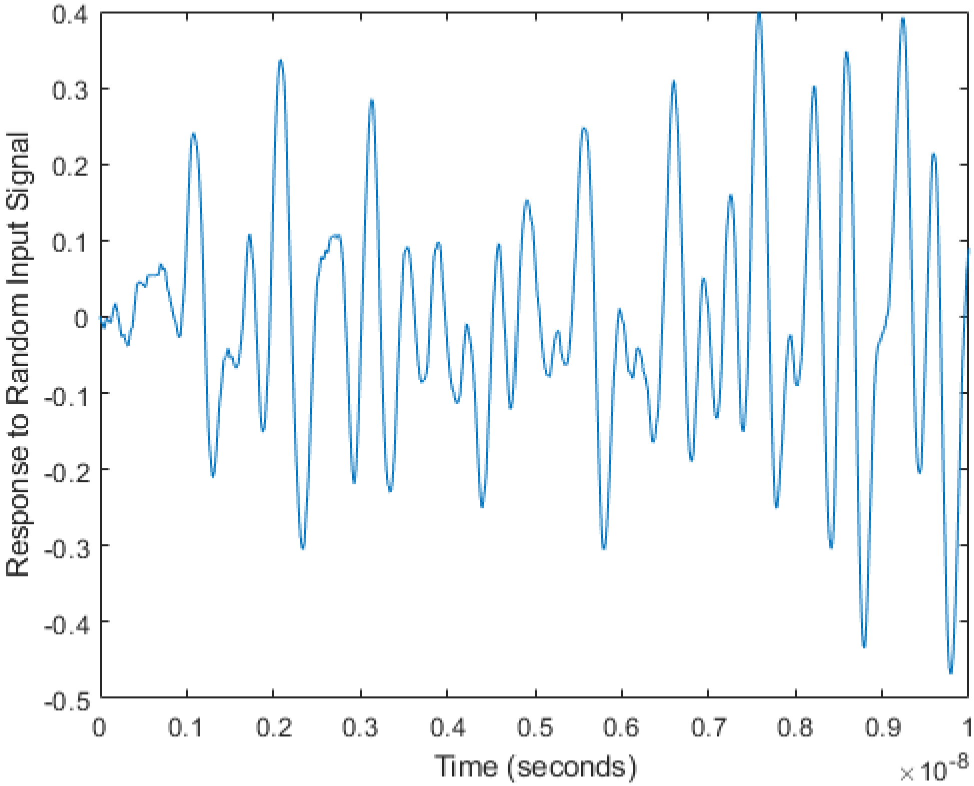
In this part of the example, you compute and plot the time-domain response of the transmission line.

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

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

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

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

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



Export a Verilog-A Model

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

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

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

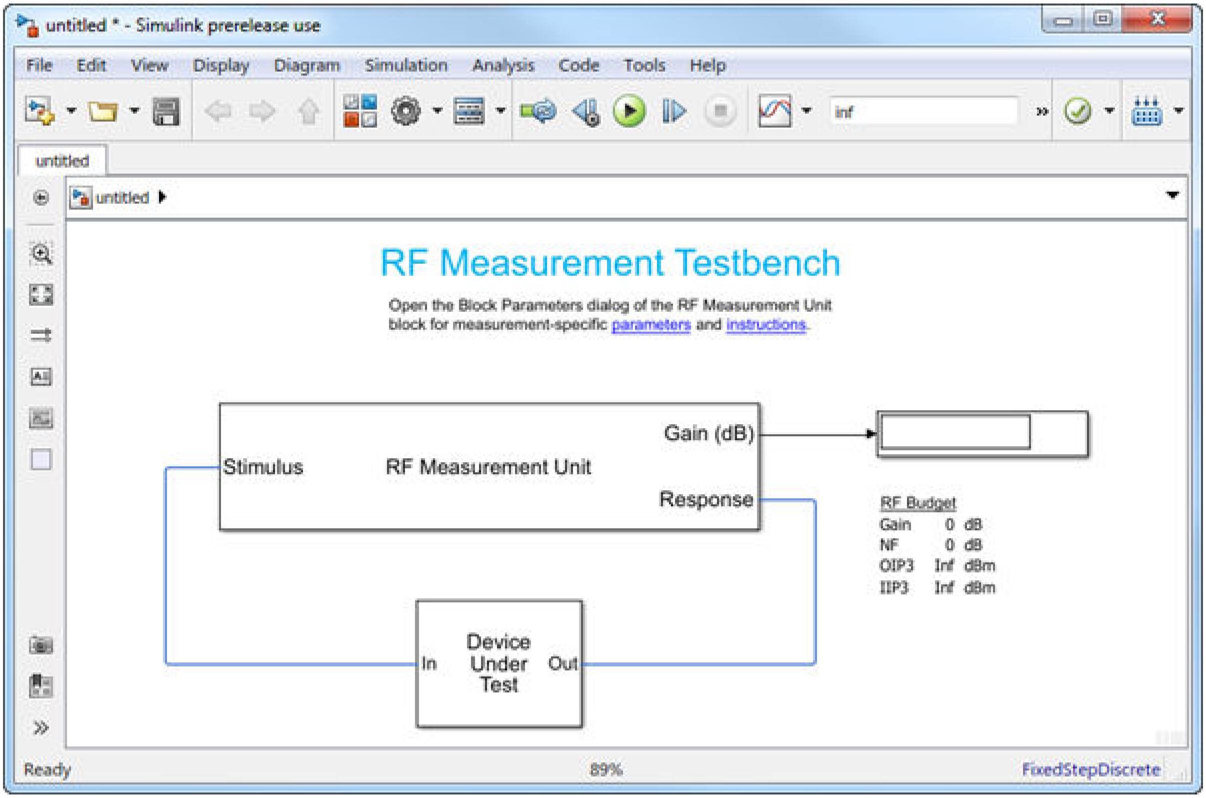
For more information on the `writeva` method and its arguments, see the `writeva` reference page. For more information on Verilog-A models, see "Export a Verilog-A Model" on page 4-4.

Using RF Measurement Testbench

In this section...

Introduction

Use the RF Measurement testbench to verify the cumulative gain, noise figure, and nonlinearity (IP3) values of an RF-to-RF system. To use the testbench, create a system in the **RF Budget Analyzer** app and click `Export` > `Export to Measurement Testbench`.

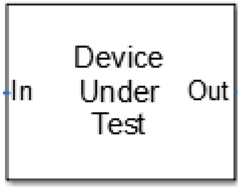


The testbench is made up of two subsystems:

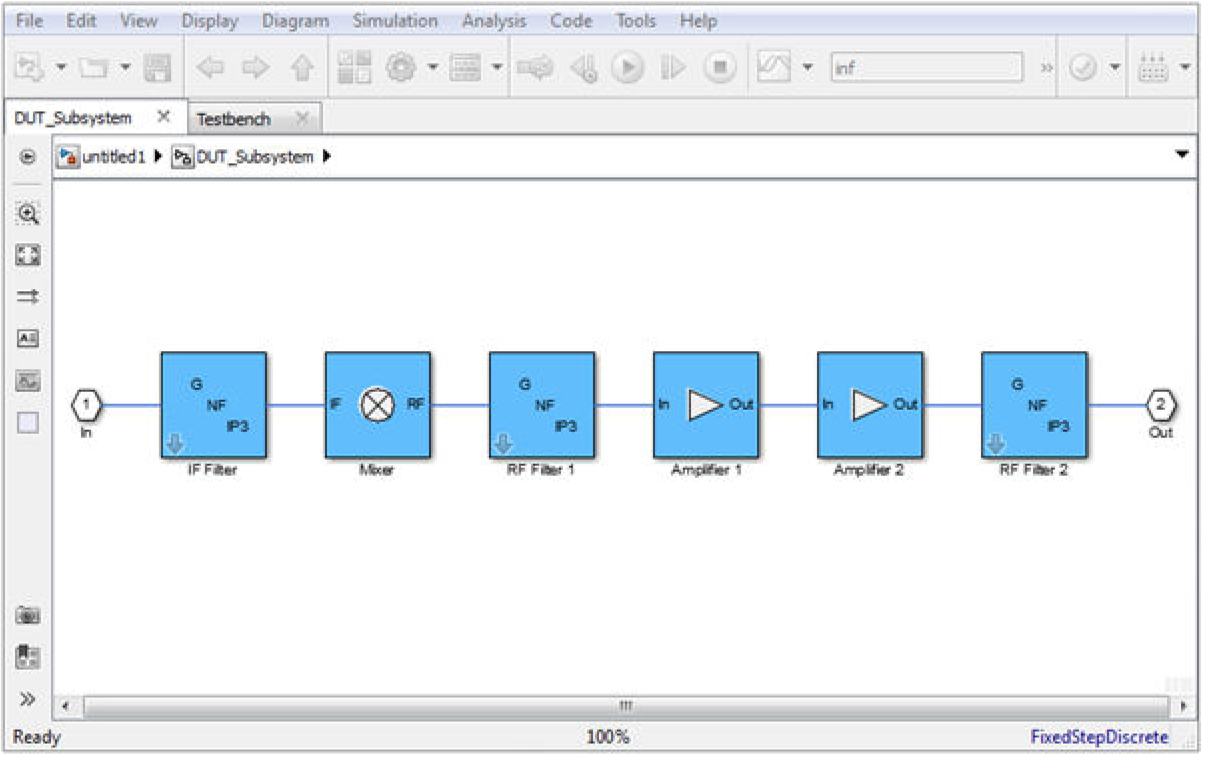
- `RF Measurement Unit`
- `Device Under Test`

The testbench display shows the verified output values of gain, NF (noise figure), and IP3 (third-order intercept).

Device Under Test Subsystem

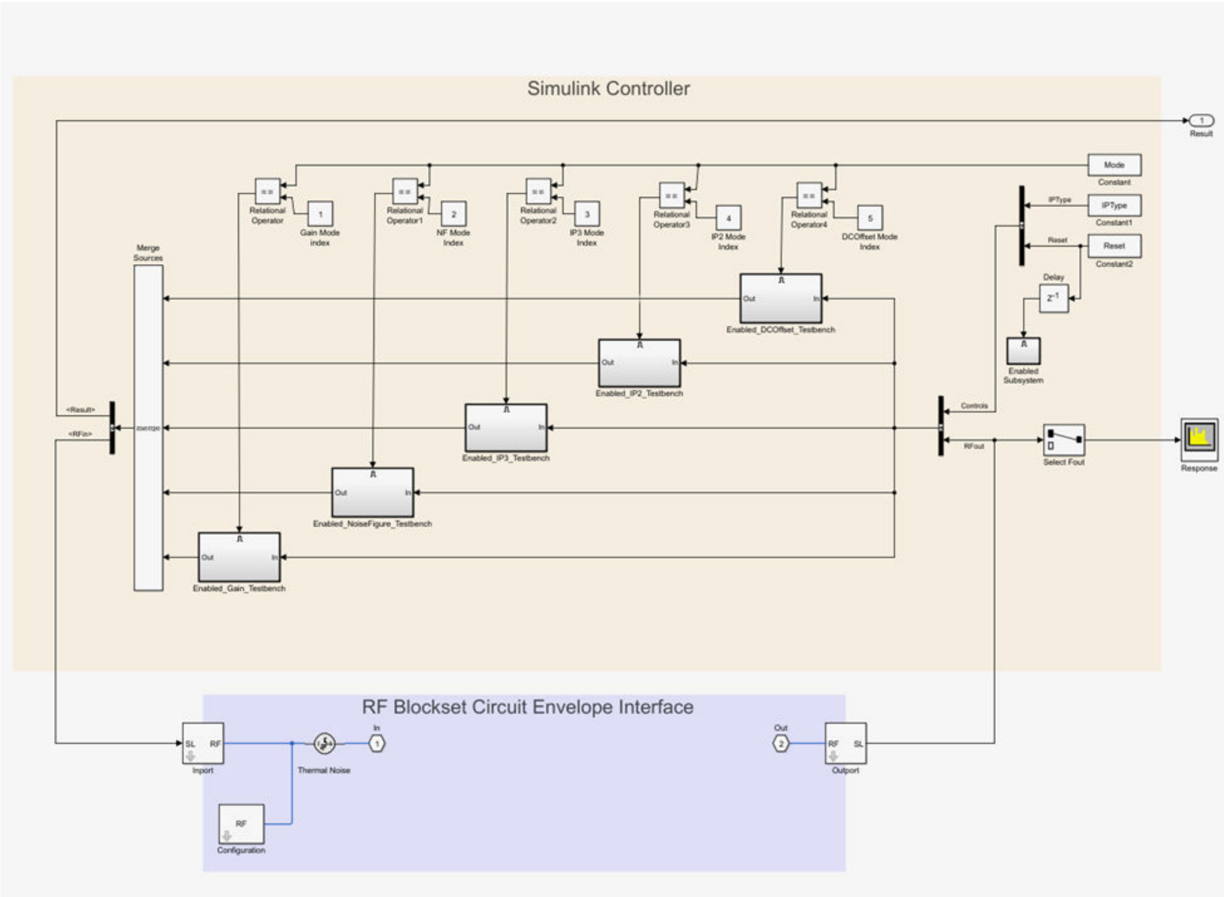


The Device Under Test subsystem contains the RF system exported from the app. To see the RF system, double-click the subsystem.

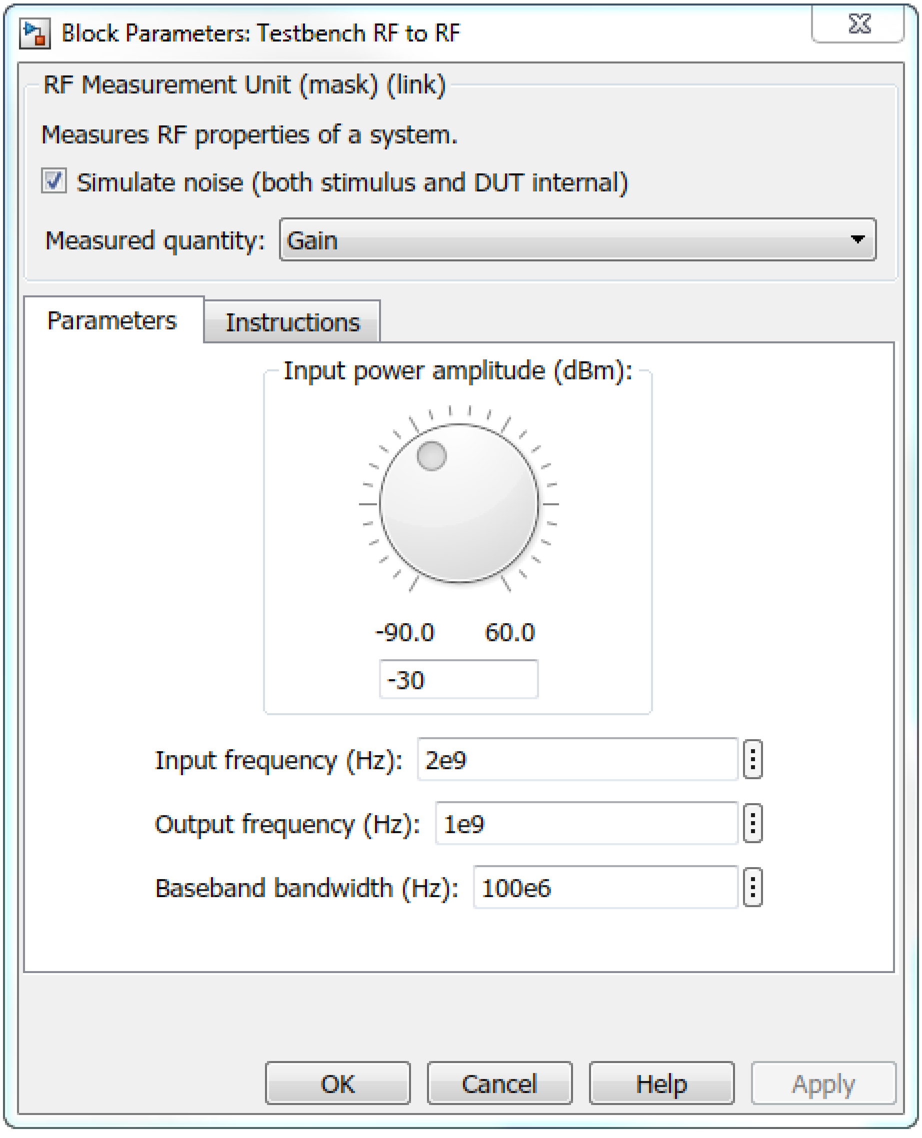


RF Measurement Unit

The RF Measurement Unit subsystem consists of a Simulink Controller and RF Blockset Circuit Envelope interface. The RF Blockset interface is used as input and output from the DUT.


Simulink Controller
Result
Mode
Constant
Relational Operator
Gain Mode index
Relational Operator1
NF Mode Index
Relational Operator2
IP3 Mode Index
Relational Operator3
IP2 Mode Index
Relational Operator4
DCOffset Mode Index
IPType
Constant1
Reset
Constant2
Delay
Enabled Subsystem
Merge Sources
Enabled_DCOffset_Testbench
Enabled_IP2_Testbench
Enabled_IP3_Testbench
Enabled_NoiseFigure_Testbench
Enabled_Gain_Testbench
Controls
RFout
Select Fout
Response
RF Blockset Circuit Envelope Interface
Inport
Thermal Noise
Configuration
Outport

RF Measurement Unit Parameters



- **Simulate noise (both stimulus and DUT internal)** — Select this check box to enable noise modeling in the stimulus signal entering the DUT and inside the DUT.

- **Measured quantity** — Choose the quantity you want to verify from:
  - `Gain` – Measure the transducer gain of the converter, assuming a load of 50 ohm. If you choose only `I` or only `Q` from **Response branch**, you see only half the value of the measured gain.
  - `NF` – Measure the noise figure value at the output of the converter.
  - `IP3` – Measure the output or input third-order intercept (IP3).
  - `IP2` – Measure the output or input second-order intercept (IP2).
  - `DC Offset` – Measure the DC level interference centered on the desired signal due to LO leakage mixing with input signal.

  The contents in the **Instructions** tab changes according to the **Measured quantity**.
- **IP Type** — Choose the type of intercept points (IP) to measure: `Output referred` or `Input referred`.

  By default, the testbench measures `Output referred`. This option is available when you set the **Measured quantity** to `IP2` or `IP3`.

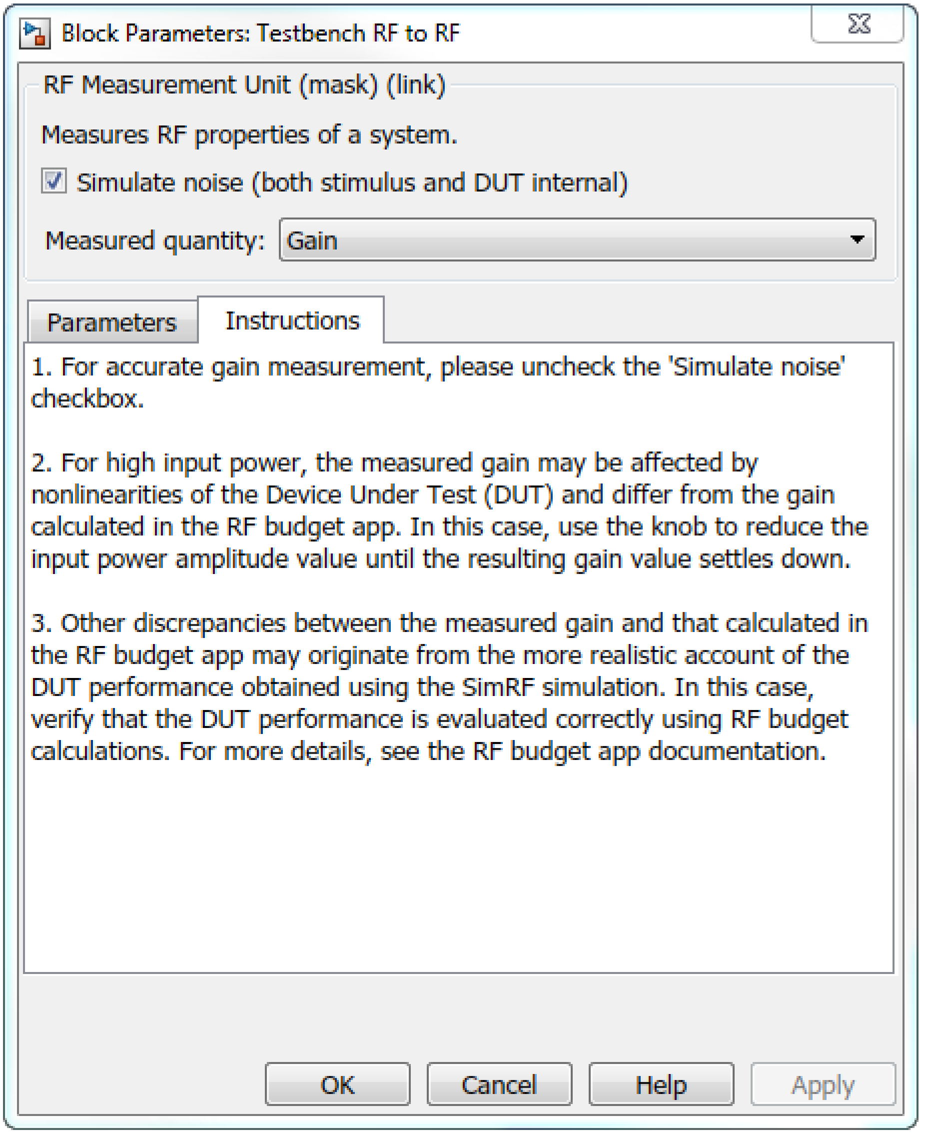
The two tabs are: **Parameters** and **Instructions**.

Parameters

- **Input power amplitude (dBm)** — Input power to the DUT. You can change the input power by manually specifying or by turning the knob. When measuring `DC Offset`, this input field is **Input RMS voltage (dBmV)**, because the Offset is measured in voltage units. The specified voltage represents the voltage falling on the input ports of the DUT.
- **Input frequency (Hz)** — Carrier frequency of the DUT.
- **Output frequency (Hz)** — Output frequency of the DUT.
- **Baseband bandwidth (Hz)** — Bandwidth of the input signal.
- **Ratio of test tone frequency to baseband bandwidth** — Position of the test tones used for IP3 measurements. By default, the value is `1/8`.

  This option is available when you set the **Measured quantity** to `IP2`, `IP3`, or `DC Offset`.

Instructions



Instructions for Gain Verification

- Clear **Simulate noise (both stimulus and DUT)** for accurate gain verification. Select the check box for account for noise.
- Change the **Input power amplitude (dBm)** or turn the knob to reduce the input power amplitude. For high input power, nonlinearities in the DUT can affect the gain measurements.

Instructions for NF Verification

- The testbench verifies the spot NF calculated. This calculation assumes a frequency-independent system within a given bandwidth. To simulate a frequency-independent system and calculate the correct NF value, reduce the baseband bandwidth until this condition is fulfilled. In common RF systems, the bandwidth should be reduced below 1 kHz for NF testing.
- Change **Input power amplitude (dBm)** or turn the knob to reduce or increase the input power amplitude. For high input power, nonlinearities in the DUT can affect the NF measurements. For low input power, the signal is too close or below the noise floor of the system. As a result, the NF fails to converge.

Instructions for OIP3 and IIP3 Verification

- Clear **Simulate noise (both stimulus and DUT)** for accurate OIP3 and IIP3 verification.
- Change **Input power amplitude (dBm)** or turn the knob to reduce the input power amplitude. For high input power, higher-order nonlinearities in the DUT can affect the OIP3 and IIP3 measurements.

For all measurement verifications using the testbench, you cannot correct result discrepancies using the **RF Budget Analyzer** app. The RF Blockset testbench provides true RF circuit simulation that incorporates RF phenomena including saturation and interaction between multiple tones and harmonics in nonlinear devices. These RF phenomena are not yet incorporated in **RF Budget Analyzer**, leading to some differences in the values between the testbench and the app.

Instructions for DC Offset Measurement

- Clear **Simulate noise (both stimulus and DUT)** for accurate DC offset measurement.
- Correct calculation of the DC offset assumes a frequency-independent system in the frequencies surrounding the test tones. Reduce the frequency separation between the

test tones or reduce the baseband bandwidth until this condition is fulfilled. In common RF systems, the bandwidth is reduced below `1 KHz` for DC offset testing.

- . Change **Input RMS voltage amplitude (dBmV)** or turn the knob to reduce the input RMS voltage amplitude. For high input RMS voltage, higher-order nonlinearities in the DUT can affect the DC offset measurements

For all measurement verifications using the testbench, you cannot correct result discrepancies using the **RF Budget Analyzer** app. The RF Blockset measurement testbench provides true RF circuit simulation that incorporates RF phenomena including saturation and interaction between multiple tones and harmonics in nonlinear devices. These RF phenomena are not yet incorporated in **RF Budget Analyzer**, leading to some differences in the values between the testbench and the app.

See Also

RF Budget Analyzer

RF Objects

RF Data Objects

In this section...

Overview

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

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

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

Types of Data

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

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

Available Data Objects

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

Constructor	Description
`rfdata.data`	Data object containing network parameter data
`rfdata.ip3`	Data object containing IP3 information
`rfdata.mixerspur`	Data object containing mixer spur information from an intermodulation table
`rfdata.network`	Data object containing network parameter information
`rfdata.nf`	Data object containing noise figure information
`rfdata.noise`	Data object containing noise information
`rfdata.power`	Data object containing power and phase information

Data Object Methods

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

Method	Types of Objects	Purpose
`extract`	`rfdata.data`, `rfdata.network`	Extract specified network parameters from a circuit or data object and return the result in an array
`read`	`rfdata.data`	Read RF data parameters from a file to a new or existing data object.
`write`	`rfdata.data`	Write RF data from a data object to a file.

RF Circuit Objects

In this section...

Overview of RF Circuit Objects

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

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

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

Components Versus Networks

You define component behavior using network parameters and physical properties.

To specify an individual RF component:

1. Construct a circuit object to represent the component.
2. Specify or import component data.

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

To specify an RF network:

1. Build circuit objects to represent the network components.
2. Construct a circuit object to represent the network.

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

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

These procedures are illustrated by example in “Model a Cascaded RF Network” on page 1-10.

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
`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

Constructor	Description
`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
`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.

Method	Types of Objects	Purpose
`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.
`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.

Method	Types of Objects	Purpose
`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

In this section...

Overview of RF Model Objects

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

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

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

Available Model Objects

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

Constructor	Description
`rfmodel.rational`	Rational function model

Model Object Methods

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

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

RF Network Parameter Objects

In this section...

Overview of Network Parameter Objects

RF Toolbox software offers network parameter objects for:

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

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

Available Network Parameter Objects

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

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

Network Parameter Object Functions

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

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

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

3

Model an RF Component

Create RF Objects

In this section...

Construct a New Object

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

This section contains the following topics:

Call the Object Constructor

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

```
h = objecttype.objectname
```

where:

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

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

```
h = rfckt.rlcgline
```

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

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

```
t1 = rfckt.microstrip
```

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

```
t1 =
           Name: 'Microstrip Transmission Line'
          nPort: 2
 AnalyzedResult: []
     LineLength: 0.0100
       StubMode: 'NotAStub'
    Termination: 'NotApplicable'
          Width: 6.0000e-004
         Height: 6.3500e-004
      Thickness: 5.0000e-006
       EpsilonR: 9.8000
      SigmaCond: Inf
    LossTangent: 0
```

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

Fit a Rational Function to Passive Component Data

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

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

For more information, see “Fit a Model Object to Circuit Object Data” on page 3-34.

Copy an Existing Object

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

For example,

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

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

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

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

Specify or Import Component Data

In this section...

RF Object Properties

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

Note The “RF Circuit Objects” on page 2-4, “RF Data Objects” on page 2-2,“RF Model Objects” on page 2-9 sections list the available types of objects and provide links to their reference pages.

Set Property Values

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

This section contains the following topics:

- “Specify Property Values at Construction” on page 3-5
- “Change Property Values of an Existing Object” on page 3-7

Specify Property Values at Construction

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

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

  Note You must use single quotation marks around the property name.
- *PropertyValue* is the value to assign to the property.

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

This section contains examples of how to perform the following tasks:

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

Construct Components with Specified Properties

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

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

t1 =

              Name: 'Coaxial Transmission Line'
             nPort: 2
    AnalyzedResult: []
        LineLength: 0.0500
          StubMode: 'NotAStub'
       Termination: 'NotApplicable'
       OuterRadius: 0.0026
       InnerRadius: 7.2500e-004
               MuR: 1
          EpsilonR: 2.3000
       LossTangent: 0
         SigmaCond: Inf
```

Construct Networks of Specified Components

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

Suppose you have the following RF components:

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

The following code creates a cascaded network of these components:

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

Change Property Values of an Existing Object

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

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

This section discusses the first option. For details on the second option, see "Reference Properties Directly Using Dot Notation" on page 3-14.

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

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

where

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

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

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

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

```
set(t1)

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

Import Property Values from Data Files

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

You can import the following file formats:

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

  For more information on Touchstone files, see `https://ibis.org/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 “AMP File Data Sections” on page 9-2.

This section contains the following topics:

- “Objects Used to Import Data from a File” on page 3-9
- “How to Import Data Files” on page 3-9

Objects Used to Import Data from a File

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

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

How to Import Data Files

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

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

where

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

For example,

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

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

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

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

Note When you import component data from a `.p2d` or `.s2d` file, properties are defined for several operating conditions. You must select an operating condition to specify the object behavior, as described in “Specify Operating Conditions” on page 3-16.

Use Data Objects to Specify Circuit Properties

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

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

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

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

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

Set Circuit Object Properties Using Data Objects

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

1 **Create an amplifier object.** This circuit object, `rfckt.amplifier`, has a network parameter, noise data, and nonlinear data properties. These properties control the frequency response of the amplifier, which is stored in the `AnalyzedResult` property. By default, all amplifier properties contain values from the `default.amp`

file. The `NetworkData` property is an `rfdata.network` object that contains 50-ohm S-parameters. The `NoiseData` property is an `rfdata.noise` object that contains frequency-dependent spot noise data. The `NonlinearData` property is an `rfdata.power` object that contains output power and phase information.

```
amp = rfckt.amplifier
```

The toolbox displays the following output:

```
amp =

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

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

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

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

The toolbox displays the following output:

```
netdata =

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

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

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

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

The toolbox displays the following output:

```
nfdata =

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

4 **Create a data object that stores output third-order intercept points.** Type the following command at the MATLAB prompt to create a `rfdata.ip3` object that contains an output third-order intercept point of 8.45 watts, at 2.1 GHz. Later in this example, you use this data object to update the `NonlinearData` property of the `rfckt.amplifier` object.

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

The toolbox displays the following output:

```
ip3data =

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

5 **Update the properties of the amplifier object.** Type the following set of commands at the MATLAB prompt to update the `NetworkData`, `NoiseData`, and `NonlinearData` properties of the amplifier object with the data objects you created in the previous steps:

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

Retrieve Property Values

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

This section contains the following topics:

- "Retrieve Specified Property Values" on page 3-13
- "Retrieve All Property Values" on page 3-13

Retrieve Specified Property Values

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

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

where

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

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

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

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

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

ir =
  7.2500e-004

or =
    0.0026
```

Retrieve All Property Values

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

For example:

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

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

Reference Properties Directly Using Dot Notation

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

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

For example, typing

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

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

```
ans =
        Cubic
```

Similarly,

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

resets the interpolation method to linear.

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

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

also produces

```
ans =
        Cubic
```

Specify Operating Conditions

In this section...

Available Operating Conditions

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

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

Set Operating Conditions

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

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

where

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

For example,

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

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

Display Available Operating Condition Values

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

```
setop(obj)
```

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

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

displays the available values for *`Condition1`*.

Process File Data for Analysis

In this section...
"Convert Single-Ended S-Parameters to Mixed-Mode S-Parameters" on page 3-18
"Extract M-Port S-Parameters from N-Port S-Parameters" on page 3-19
"Cascade N-Port S-Parameters" on page 3-21

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

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

This section contains the following topics:

- "Functions for Converting S-Parameters" on page 3-18
- "Convert S-Parameters" on page 3-19

Functions for Converting S-Parameters

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

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

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

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

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

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

Convert S-Parameters

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

At the MATLAB prompt:

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

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

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

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

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

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

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

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

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

This section contains the following topics:

- “Extract S-Parameters” on page 3-20

- "Extract S-Parameters From Imported File Data" on page 3-21

Extract S-Parameters

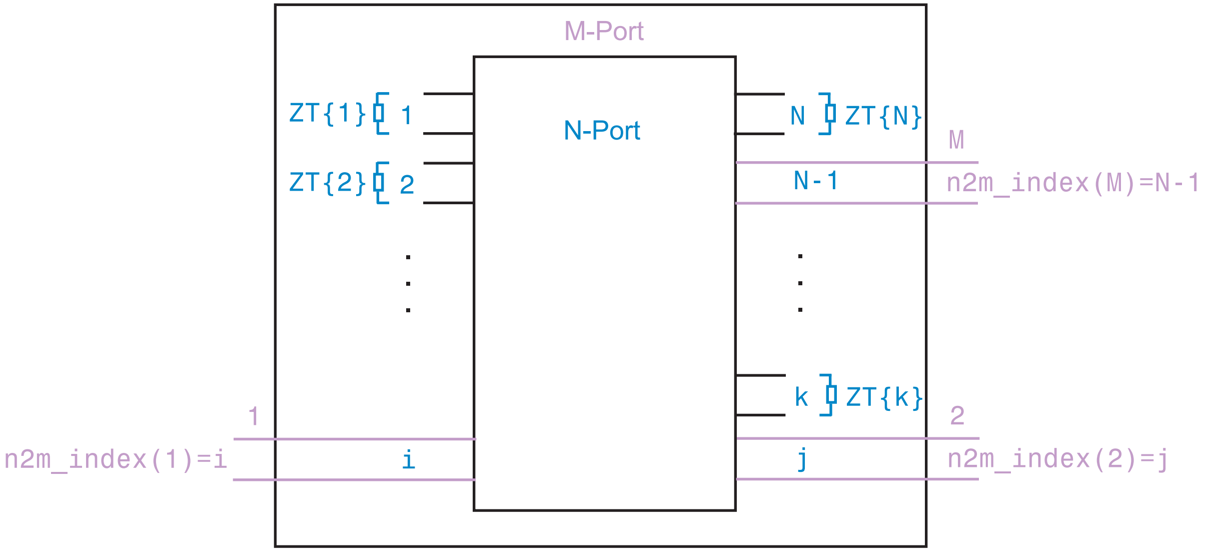
To extract M-port S-parameters from N-port S-parameters, use the `snp2smp` function with the following syntax:

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

where

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

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



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

Extract S-Parameters From Imported File Data

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

At the MATLAB prompt:

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

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

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

```
N2M_index = [1 16 2 15];
FourPortSParams = snp2smp(SingleEnded16PortData.S_Parameters, ...
    SingleEnded16PortData.Z0, N2M_index, 50);
```

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

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

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

Cascade N-Port S-Parameters

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

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

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

where

- `s_params` is an array of cascaded S-parameters.

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

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

Import and Cascade N-Port S-Parameters

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

At the MATLAB prompt:

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

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

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

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

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

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

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

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

Analyze and Plot RF Components

Analyze Networks in the Frequency Domain

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

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

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

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

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

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

Visualize Component and Network Data

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

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

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

Plot Type	Methods	Plot Contents

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

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

Plot Type	Methods	Plot Contents
Mixer Spur Plot	`plot`	Mixer spur power as a function of frequency for an `rfckt.mixer` object or an `rfckt.cascade` object that contains a mixer.
Polar Plot	`polar`	Magnitude and phase of S-parameters as a function of frequency.
Smith Chart	`smith`	Real and imaginary parts of S-parameters as a function of frequency, used for analyzing the reflections caused by impedance mismatch.

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

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

The following topics describe the available plots:

- “Rectangular” on page 3-26
- “Budget” on page 3-27
- “Mixer Spur” on page 3-30
- “Polar Plots and Smith Charts” on page 3-33

Rectangular

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

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

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

Budget

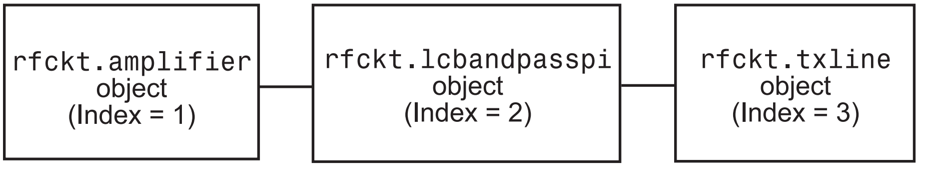
You use the link budget plot to understand the individual contribution of each component to a plotted parameter value in a cascaded network with multiple components.

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

Consider the following cascaded network:

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

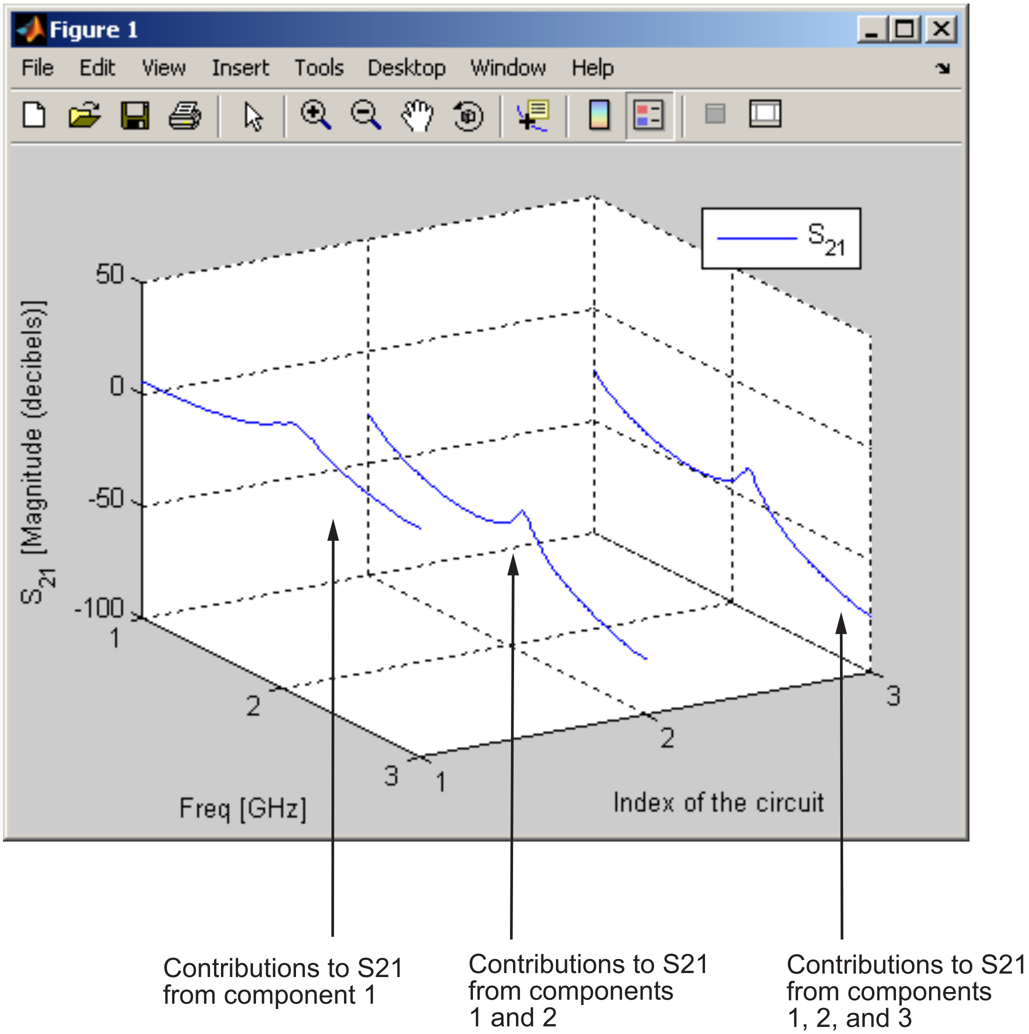
The following figure shows how the circuit index is assigned to each component in the cascade, based on its sequential position in the network.



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

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

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



Budget Plot

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

Mixer Spur

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

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

where

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

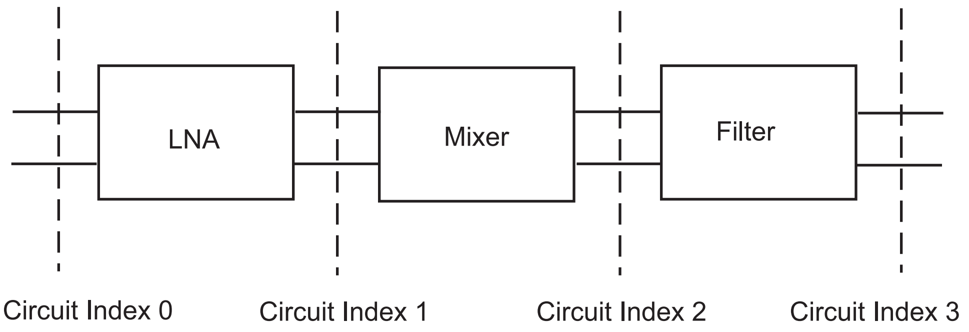
The toolbox calculates the output power from the mixer intermodulation table (IMT). These tables are described in detail in the Visualizing Mixer Spurs example.

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

Consider the following cascaded network:

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

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

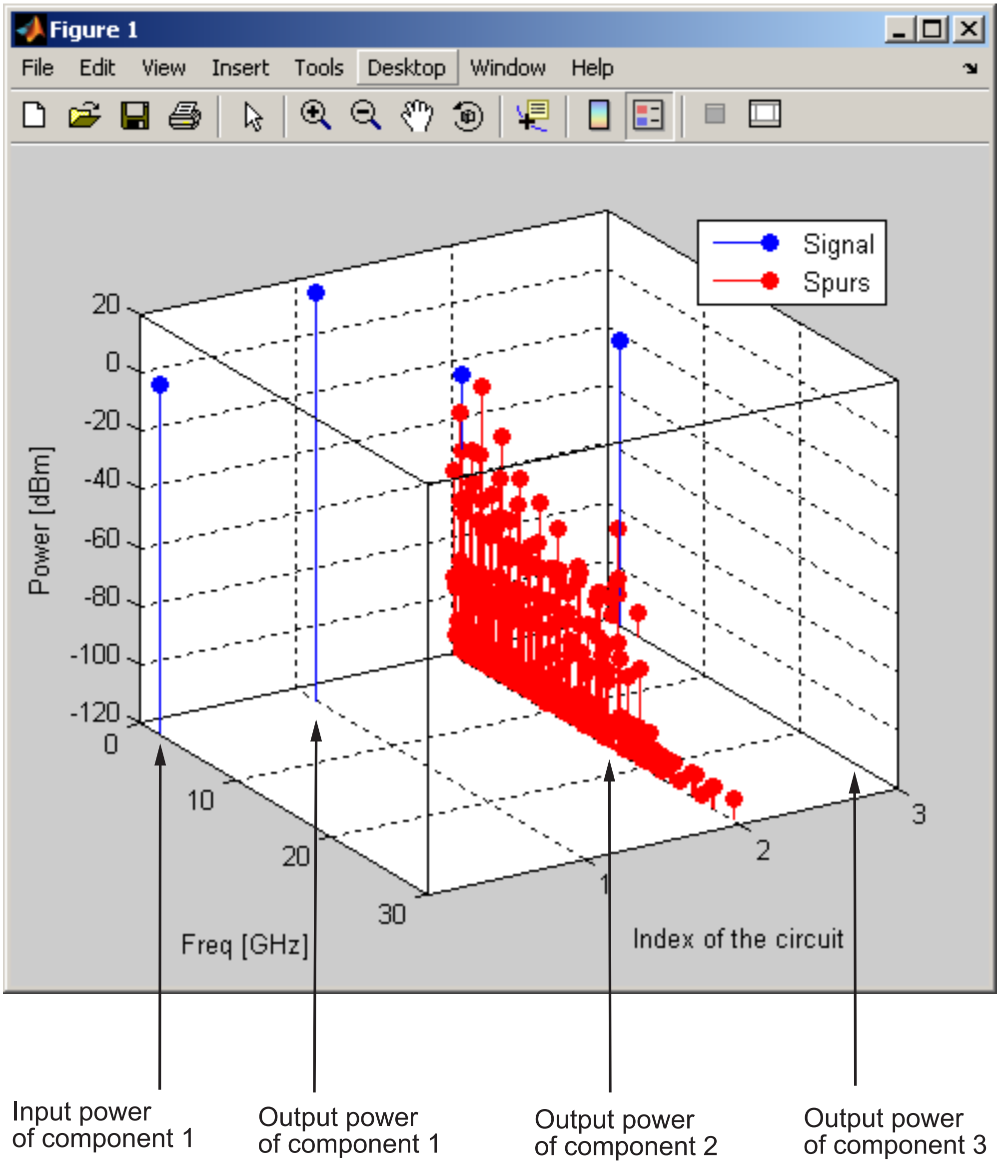


- Circuit index 0 corresponds to the cascade input.
- Circuit index 1 corresponds to the LNA output.
- Circuit index 2 corresponds to the mixer output.
- Circuit index 3 corresponds to the filter output.

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

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

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



Mixer Spur Plot

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

Polar Plots and Smith Charts

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

The following table describes the Polar plot and Smith 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 toolbox plots the parameter as a function of frequency. When you import block data from a .p2d or .s2d file, you can also plot parameters as a function of any operating condition from the file that has numeric values, such as bias.

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

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

Compute and Plot Time-Domain Specifications

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

This section contains the following topics:

Compute the Network Transfer Function

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

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

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

Fit a Model Object to Circuit Object Data

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

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

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

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

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

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

See the `rationalfit` reference page for more information.

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

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

Compute and Plot the Time-Domain Response

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

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

```
SampleTime=1e-11;
NumberOfSamples=4750;
OverSamplingFactor = 25;
```

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

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

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

Export Component Data to a File

In this section...

Available Export Formats

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

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

You can export data to the following file formats:

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

  For more information about Touchstone files, see `https://ibis.org/connector/touchstone_spec11.pdf`.
- MathWorks amplifier (AMP) file format — Specifies amplifier network parameters, output power versus input power, noise data and third-order intercept point.

  For more information about `.amp` files, see "AMP File Data Sections" on page 9-2.

How to Export Object Data

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

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

where

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

For example,

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

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

Export Object Data

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

At the MATLAB prompt:

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

   ```
   s_vec(:,:,1) = ...
       [-0.724725-0.481324i, -0.685727+1.782660i; ...
         0.000000+0.000000i, -0.074122-0.321568i];
   s_vec(:,:,2) = ...
       [-0.731774-0.471453i, -0.655990+1.798041i; ...
         0.001399+0.000463i, -0.076091-0.319025i];
   s_vec(:,:,3) = ...
       [-0.738760-0.461585i, -0.626185+1.813092i; ...
         0.002733+0.000887i, -0.077999-0.316488i];
   ```
2. Type the following to create an `rfdata.data` object called `txdata` with the default property values:

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

   ```
   txdata.S_Parameters = s_vec;
   ```
4. Type the following to set the frequency values of `txdata` to `[1e9 2e9 3e9]`:

   ```
   txdata.Freq=1e9*[1 2 3];
   ```
5. Type the following to export the data in `txdata` to a Touchstone file called `test.s2p`:

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

Basic Operations with RF Objects

In this section...

Read and Analyze RF Data from a Touchstone Data File

In this example, you create an `rfdata.data` object by reading the S-parameters of a 2-port passive network stored in the Touchstone format data file, `passive.s2p`.

1 **Read S-parameter data from a data file.** Use the RF Toolbox `read` command to read the Touchstone data file, `passive.s2p`. This file contains 50-ohm S-parameters at frequencies ranging from 315 kHz to 6 GHz. The `read` command creates an `rfdata.data` object, `data`, and stores data from the file in the object's properties.

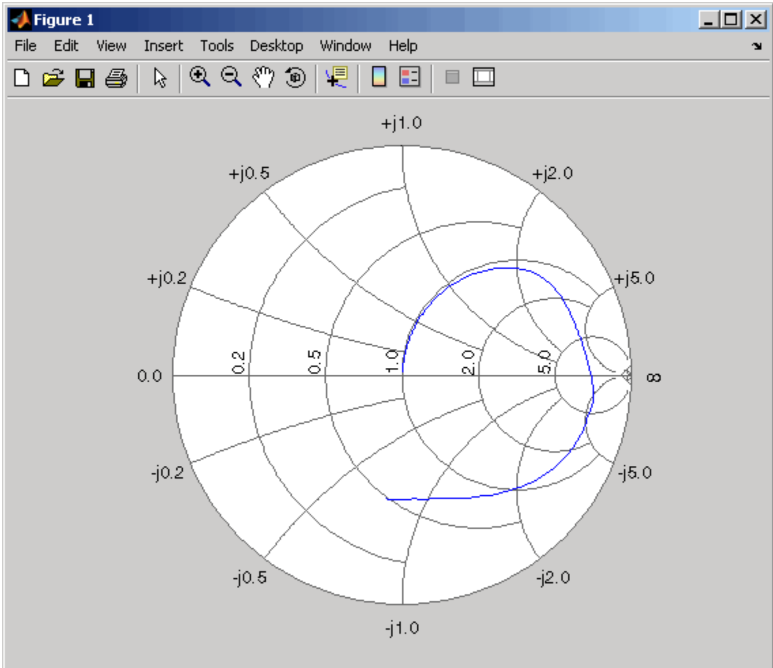
```
data = read(rfdata.data,'passive.s2p');
```

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

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

3 **Plot the S_{11} parameters.** Use the `smithchart` command to plot the 75-ohm S_{11} parameters on a Smith Chart:

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



4 **View the 75-ohm S-parameters and Y-parameters at 6 GHz.** Type the following set of commands at the MATLAB prompt to display the four 75-ohm S-parameter values and the four Y-parameter values at 6 GHz.

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

The toolbox displays the following output:

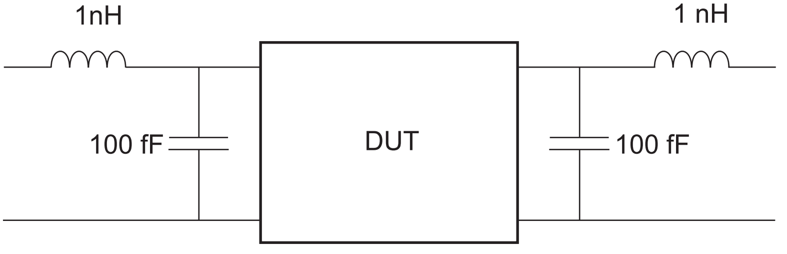
```
f =
  6.0000e+009

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

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

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

De-Embed S-Parameters

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

The configuration of the bipolar transistor, which is the device under test (DUT), and the fixture is shown in the following figure.



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

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

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

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

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

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

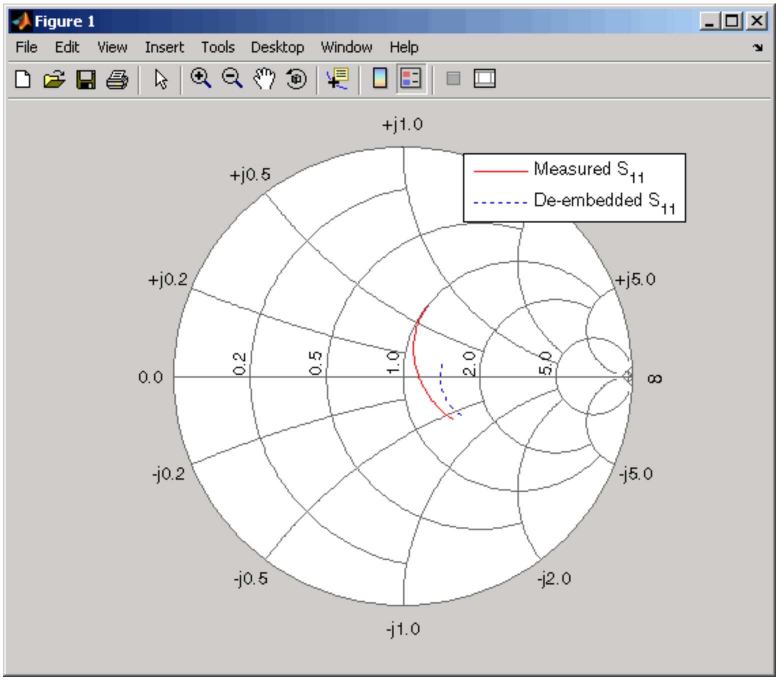
```
z0 = measured_data.Z0;

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

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

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

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

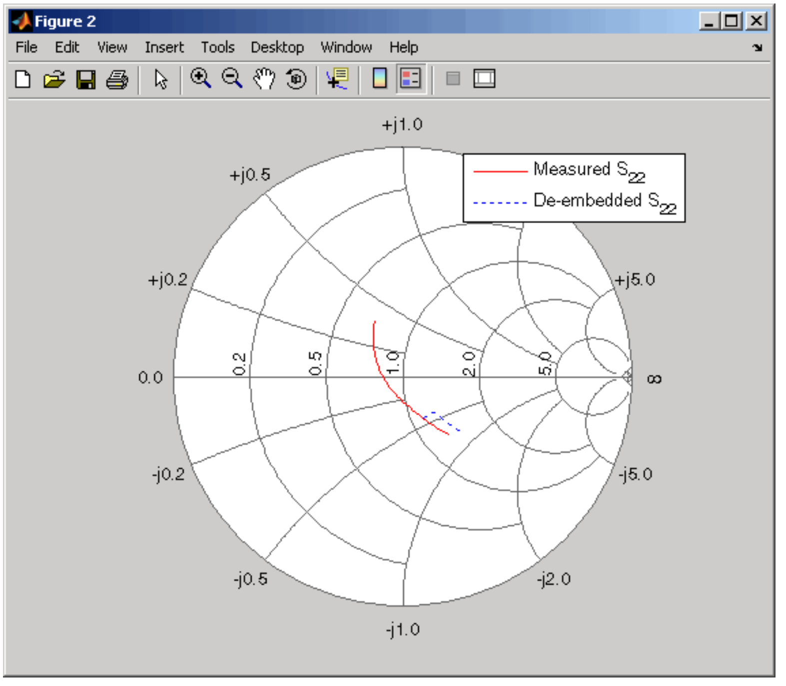
5 **Plot the measured and de-embedded S_{11} parameters.** Type the following set of commands at the MATLAB prompt to plot both the measured and the de-embedded S_{11} parameters on a Z Smith Chart:

```
hold off;
h = smith(measured_data,'S11');
set(h, 'Color', [1 0 0]);
hold on
i = smith(de_embedded_data,'S11');
set(i,'Color', [0 0 1],'LineStyle',':');
l = legend;
legend('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('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('Measured S_{21}', 'De-embedded S_{21}');
legend show;
hold off;
```

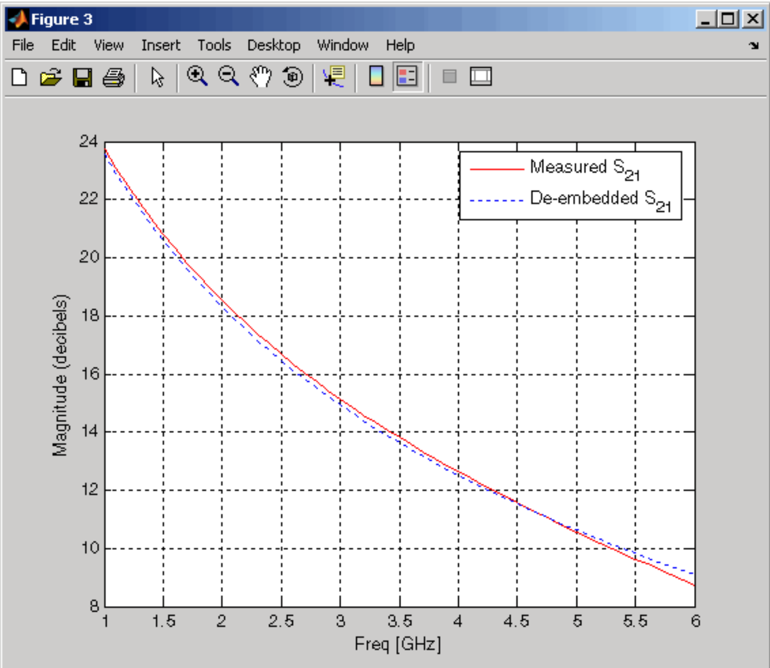


Figure 3
File Edit View Insert Tools Desktop Window Help
Measured S_{21}
De-embedded S_{21}
Magnitude (decibels)
Freq [GHz]
24
22
20
18
16
14
12
10
8
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6

4

Export Verilog-A Models

Model RF Objects Using Verilog-A

In this section...

Overview

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

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

Behavioral Modeling Using Verilog-A

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

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

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

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

Supported Verilog-A Models

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

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

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

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

where

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

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

Export a Verilog-A Model

In this section...

Represent a Circuit Object with a Model Object

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

There are two ways to create an RF model object:

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

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

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

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

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

where

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

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

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

where

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

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

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

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

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

Write a Verilog-A Module

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

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

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

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

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

The RF Design and Analysis App

The RF Design and Analysis App

In this section...

What Is the RF Design and Analysis App?

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

The RF Design and Analysis app provides the ability to

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

Open the RF Design and Analysis App

To open the app window, type the following at the MATLAB prompt:

```
rftool
```

For a description of the RF Design and Analysis user interface , see “The RF Design and Analysis Window” on page 5-3. To learn how to create and import circuits, see “Create and Import Circuits” on page 5-6.

Note The work you do with this app 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 the RF Design and Analysis App Sessions" on page 5-28.

The RF Design and Analysis Window

The app window consists of the following three panes:

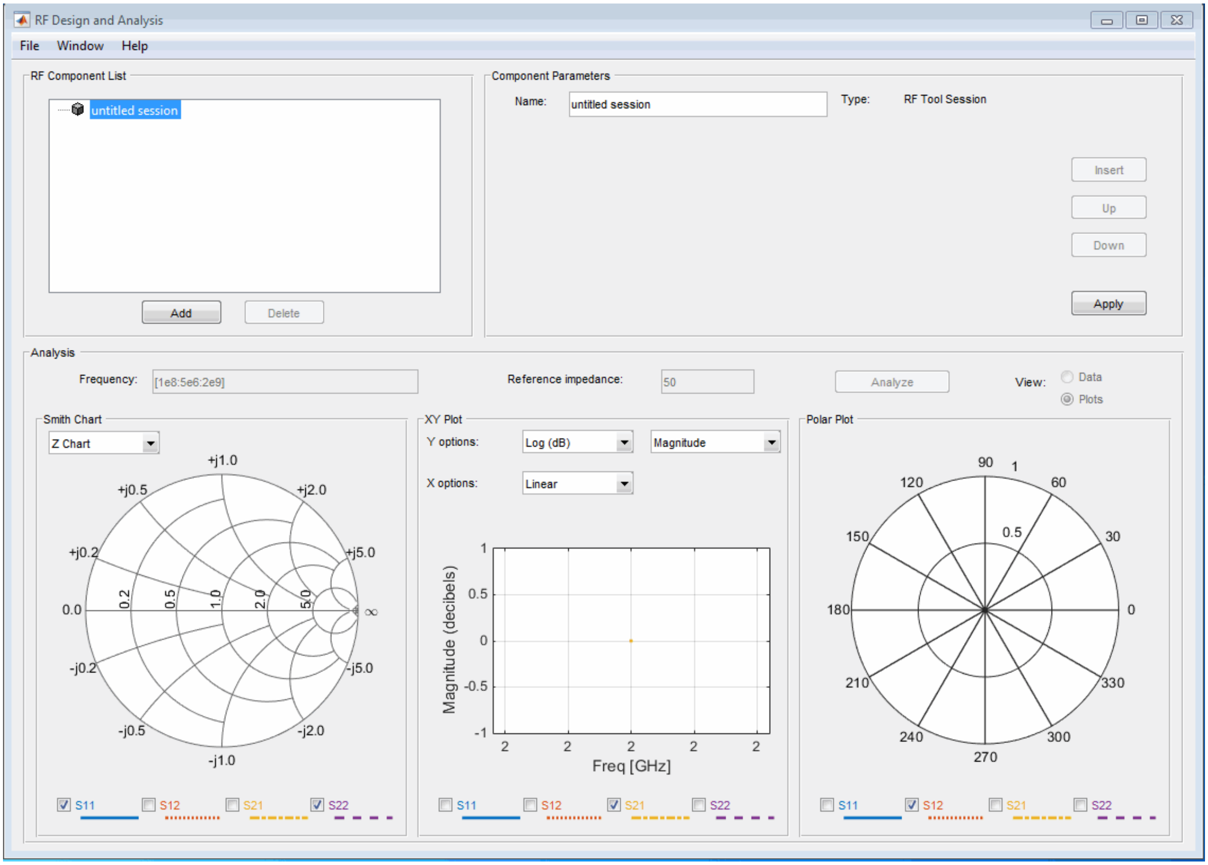
- **RF Component List**

  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 app window.



The RF Design and Analysis App Workflow

When you analyze a circuit using the app user interface your workflow might include the following tasks:

1. Build the circuit by

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

   See “Create and Import Circuits” on page 5-6.

2. Specify component data.

   See “Modify Component Data” on page 5-19.
3. Analyze the circuit.

   See “Analyze Circuits” on page 5-20.
4. Export the circuit to the MATLAB workspace or to a file.

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

Create and Import Circuits

In this section...

Circuits in the RF Design and Analysis App

In this app, 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 the app by creating a network and then populating it with components.

Create RF Components

This section contains the following topics:

Available RF Components

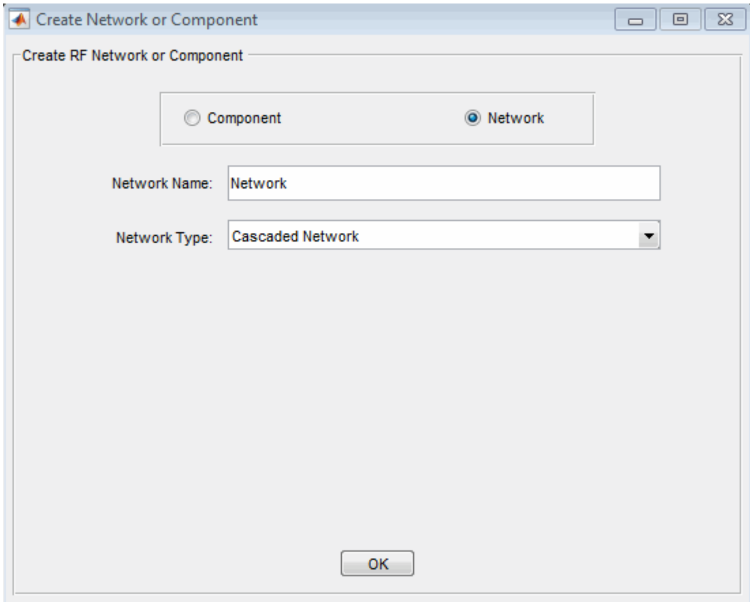
The following table lists the RF components you can create using the app and the corresponding RF Toolbox object.

RF Component	Corresponding RF Object
Data File	`rfckt.datafile`
Delay Line	`rfckt.delay`
Coaxial Transmission Line	`rfckt.coaxial`
Coplanar Waveguide Transmission Line	`rfckt.cpw`

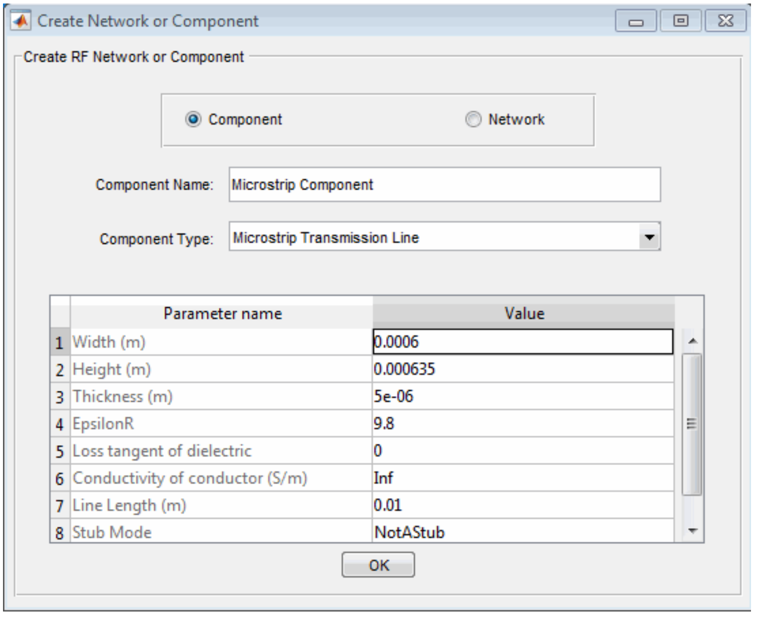
RF Component	Corresponding RF Object
Microstrip Transmission Line	`rfckt.mixer`
Parallel-Plate Transmission Line	`rfckt.parallelplate`
Transmission Line	`rfckt.txline`
Two-Wire Transmission Line	`rfckt.twowire`
Series RLC	`rfckt.seriesrlc`
Shunt RLC	`rfckt.shuntrlc`
LC Bandpass Pi	`rfckt.lcbandpasspi`
LC Bandpass Tee	`rfckt.lcbandpasstee`
LC Bandstop Pi	`rfckt.lcbandstoppi`
LC Bandstop Tee	`rfckt.lcbandstoptee`
LC Highpass Pi	`rfckt.lchighpasspi`
LC Highpass Tee	`rfckt.lchighpasstee`
LC Lowpass Pi	`rfckt.lclowpasspi`
LC Lowpass Tee	`rfckt.lclowpasstee`

Add an RF Component to a Session

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



2. In the Create Network or Component dialog box, select **Component**.
3. In the **Component Name** field, enter a name for the component. This name is used to identify the component in the **RF Component List** pane. For example, Microstrip Component.
4. From the **Component Type** menu, select the type of RF component you want to create. For example, `Microstrip Transmission Line`.



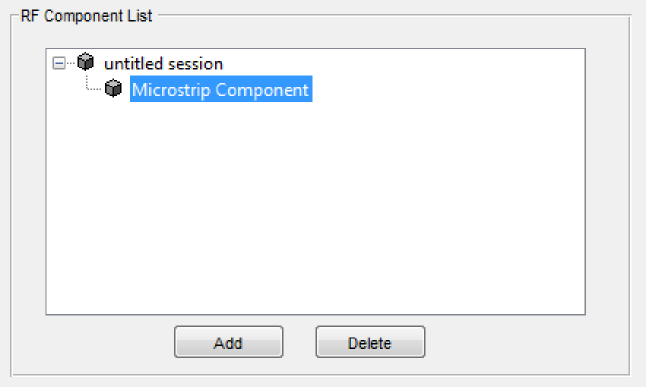
5 Adjust the parameter values as necessary.

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

---

6 Click **OK**.

The app adds the component to your session.



Create RF Networks

You create an RF network using the app 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
- "Add an RF Network to a Session" on page 5-11
- "Populate an RF Network" on page 5-13
- "Reorder Circuits Within a Network" on page 5-14

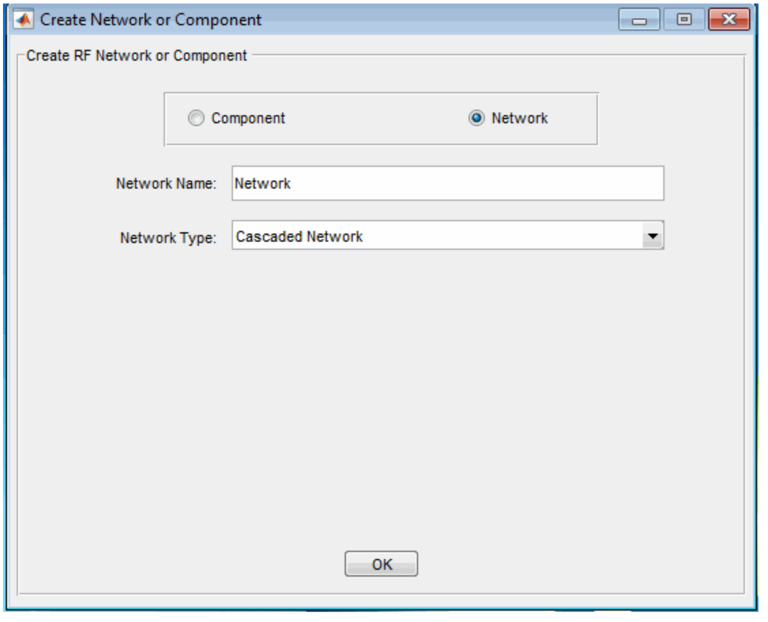
Available RF Networks

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

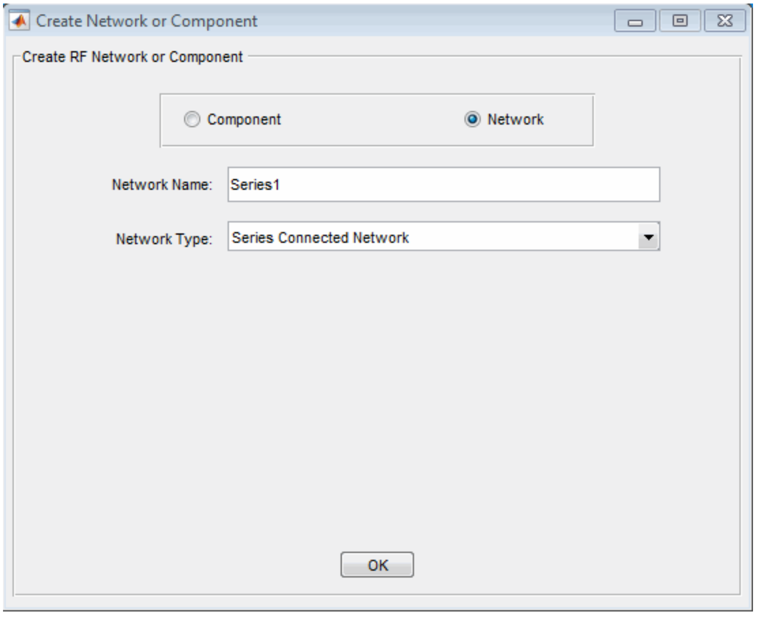
RF Network	Corresponding RF Toolbox Object
Cascaded Network	`rfckt.cascade`
Series Connected Network	`rfckt.series`
Parallel Connected Network	`rfckt.parallel`
Hybrid Connected Network	`rfckt.hybrid`
Inverse Hybrid Connected Network	`rfckt.hybridg`

Add an RF Network to a Session

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

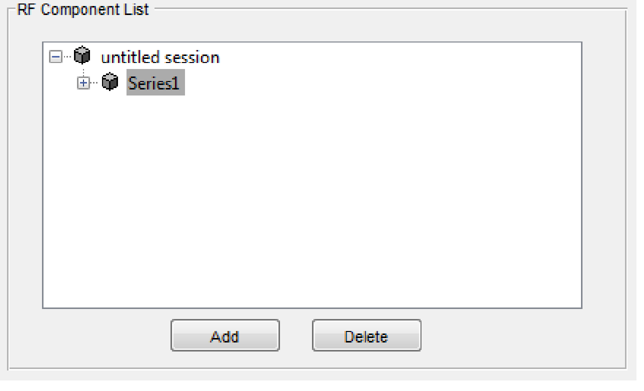


2. In the Create Network or Component dialog box, select the **Network** option button.
3. In the **Network Name** field, enter a name for the component. This name is used to identify the network in the **RF Component List** pane. For example, Series1.
4. From the **Network Type** menu, select the type of RF network you want to create. For example, `Series Connected Network`.



5 Click **OK**.

The RF Component List pane shows the new network.

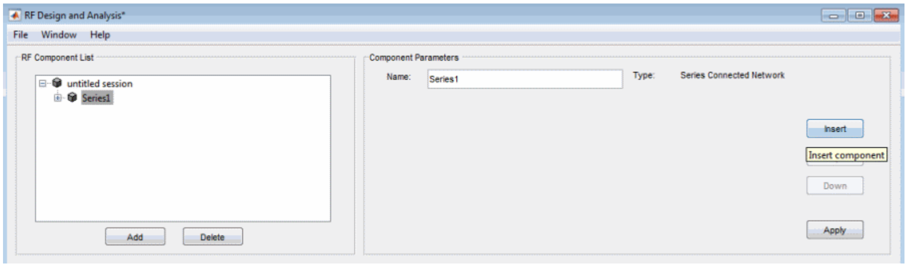


Populate an RF Network

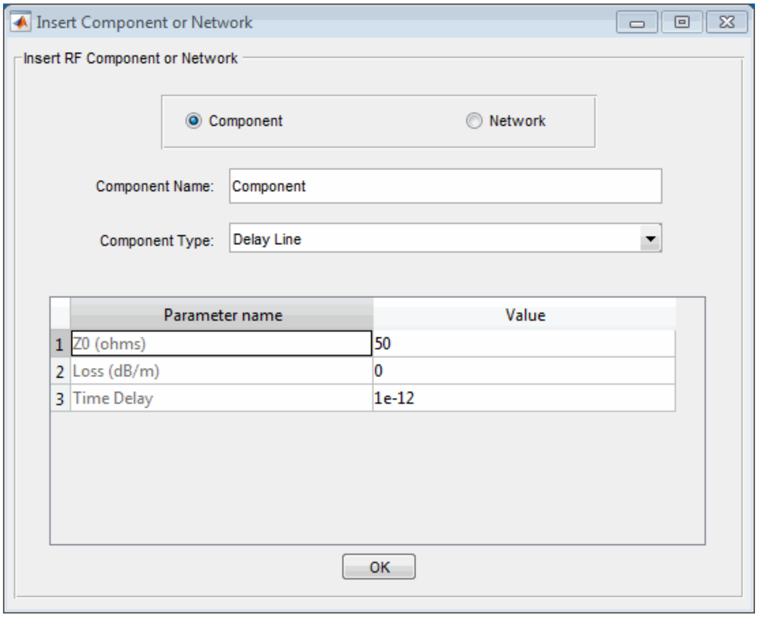
After you create a network using the app, you must populate it with RF components and networks. You insert a component or network into a network in much the same way you add one to a session.

To populate an RF network:

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



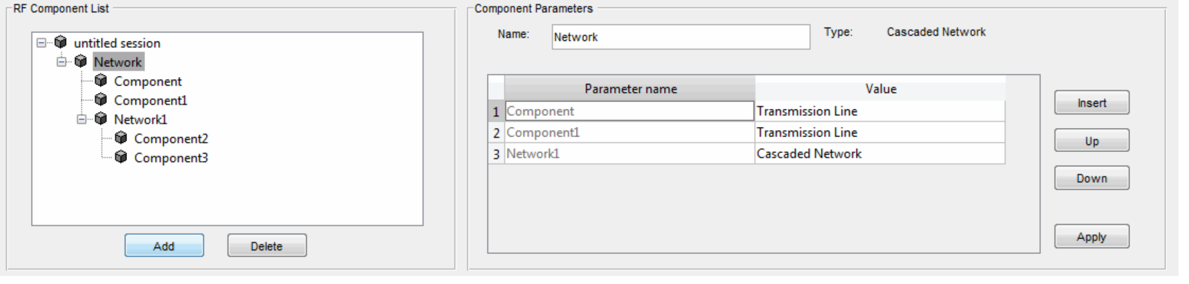
The Insert Component or Network dialog box appears.



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

   Enter the component or network name, and select the appropriate type. If you are inserting a component, modify the parameter values as necessary. See “Add an RF Component to a Session” on page 5-7 or “Add an RF Network to a Session” on page 5-11 for details.

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



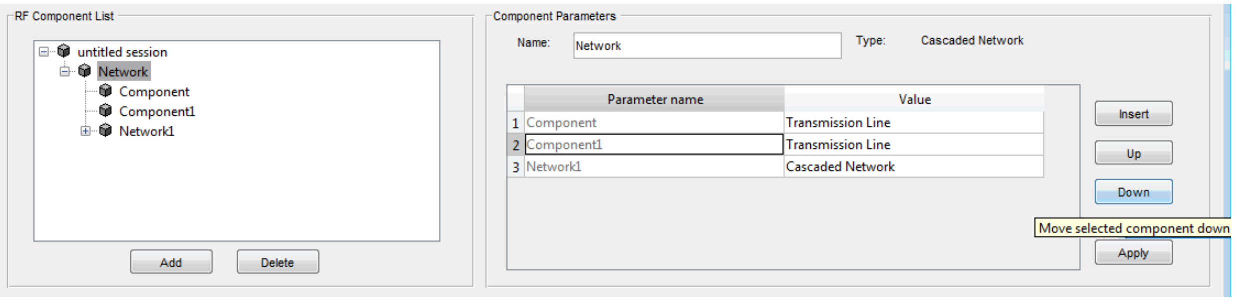
Reorder Circuits Within a Network

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

1. In the **RF Component List** pane, select the network whose circuits you want to reorder.
2. In the **Component Parameters** pane, select the circuit whose position you want to change.
3. Click **Up** or **Down** until the circuit is where you want it.

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

1. Select `Network` in the **RF Component List** pane.
2. Select `Component1` in the **Component Parameters** pane.
3. Click **Down** in the **Component Parameters** pane.



Import RF Objects into the RF Design and Analysis App

The RF Design and Analysis app 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 session.

  For information on exporting components and networks from another session, see "Export 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.

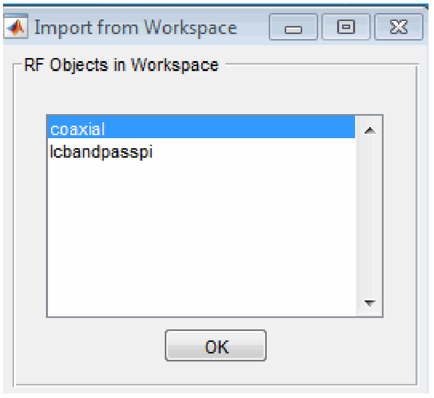
This section contains the following topics:

- "Import from the Workspace" on page 5-15
- "Import from a File into a Session" on page 5-16
- "Import from a File into a Network" on page 5-17

Import from the Workspace

To import RF circuit objects from the MATLAB workspace into your 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, the app appends a numeral, starting with 1, to the new circuit name.

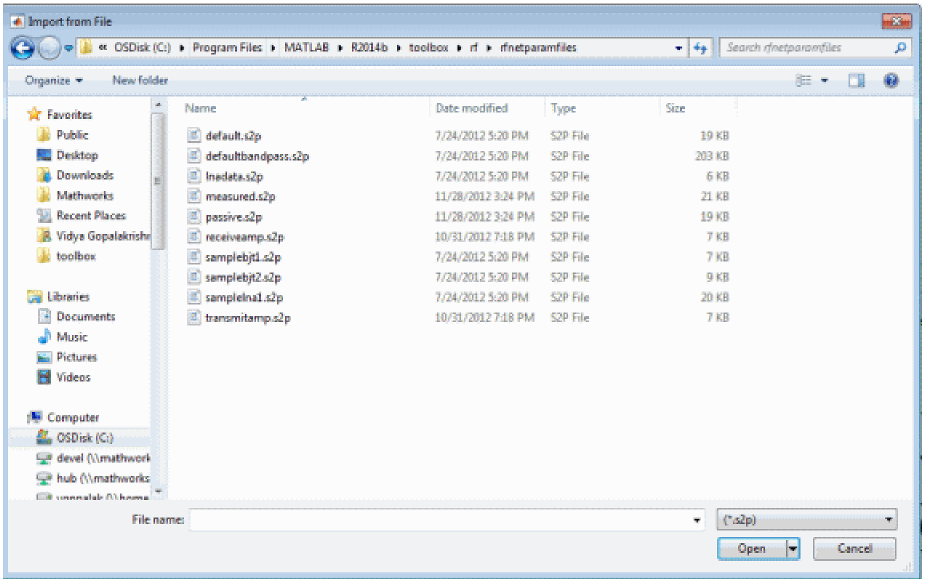
Import from a File into a Session

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

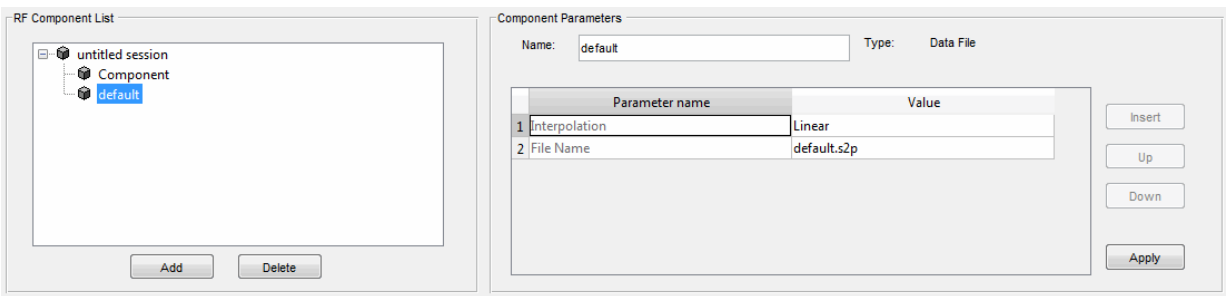
- S2P
- Y2P
- Z2P
- H2P

To import a component from one of these files:

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



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



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

Import from a File into a Network

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

- S2P
- Y2P
- Z2P
- H2P

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

1. Insert a Data File component into the network.

   For more information on how add a component to a network, see “Populate an RF Network” on page 5-13.
2. Specify the name of the file from which to import the component in one of two ways:

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

“Model an RF Network” on page 5-31 shows this process.

Modify Component Data

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

To modify these values:

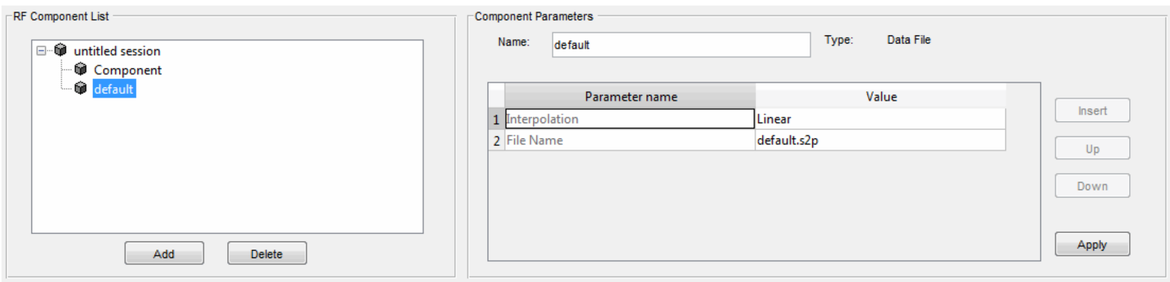
1. Select the component in the **RF Component List** pane.
2. In the **Component Parameters** pane, select the value you want to change, and enter the new value.

   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**.

Analyze Circuits

After you add your circuits, you can analyze them using the app:

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



2. In the **Analysis** pane:

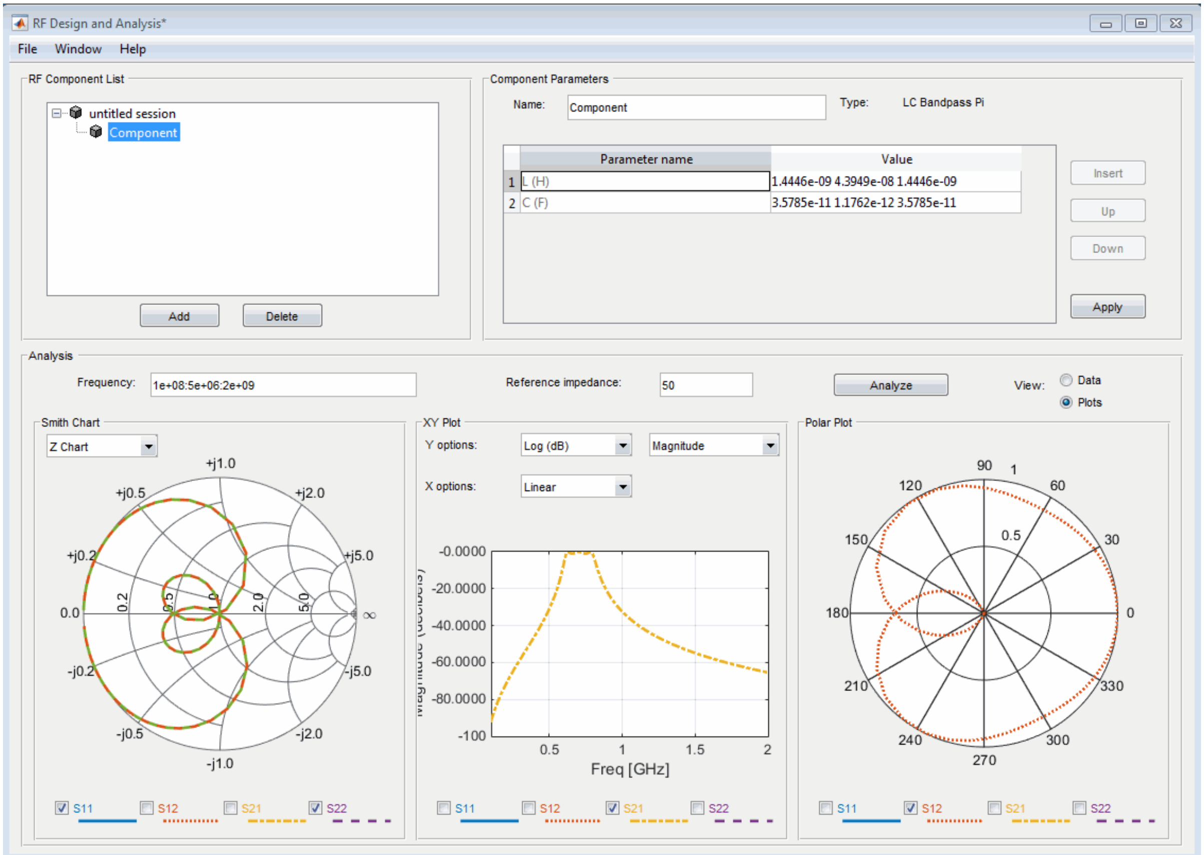
   - Enter `[1e8:5e6:2e9]`, the analysis frequency range and step size in hertz, in the **Frequency** field.

     This value specifies an analysis from 0.1 GHz to 2 GHz in 5 MHz steps.
   - Enter `50`, the reference impedance in ohms, in the **Reference impedance** field.

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

3. Click **Analyze**.

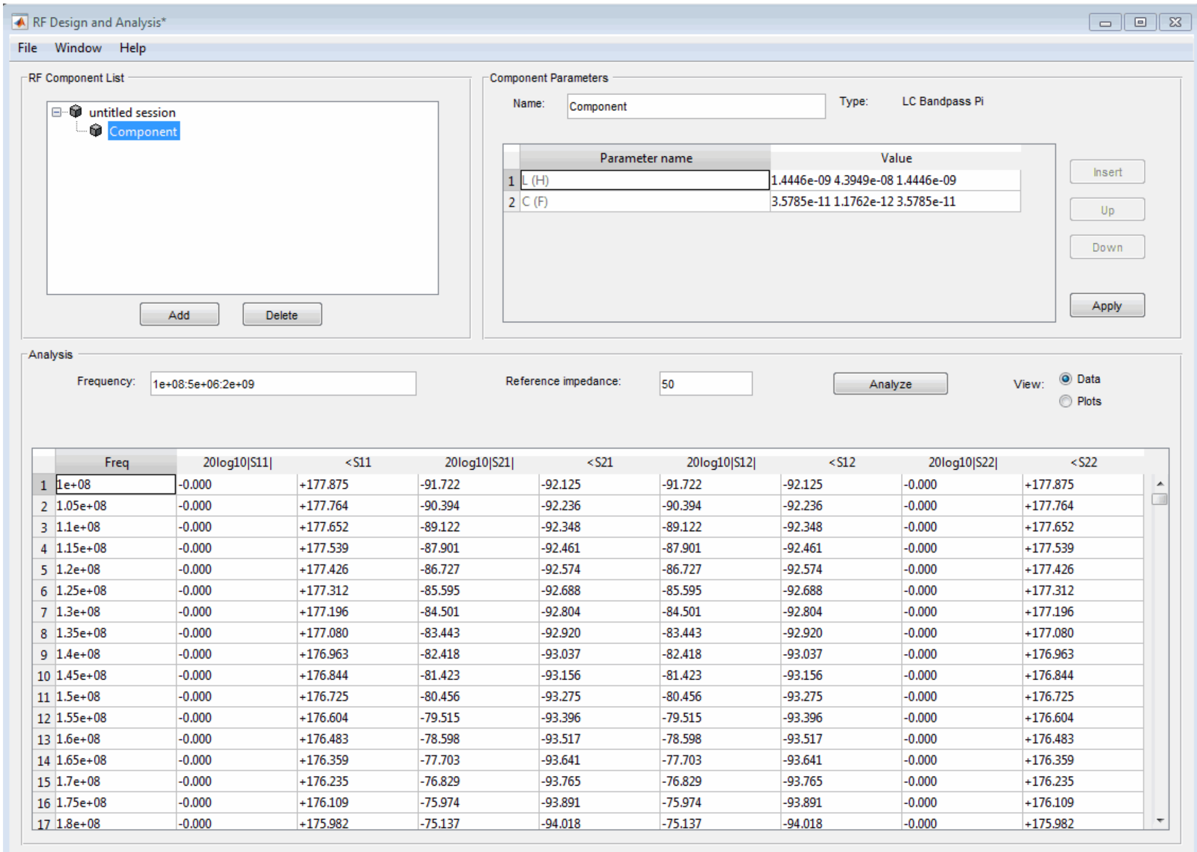
   The **Analysis** pane displays a Smith Chart, an XY plot, and a polar plot of the analyzed circuit.



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

The plots automatically update as you change the check box and drop-down list options on the user interface.

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



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

Export RF Objects

In this section...

Export Components and Networks

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

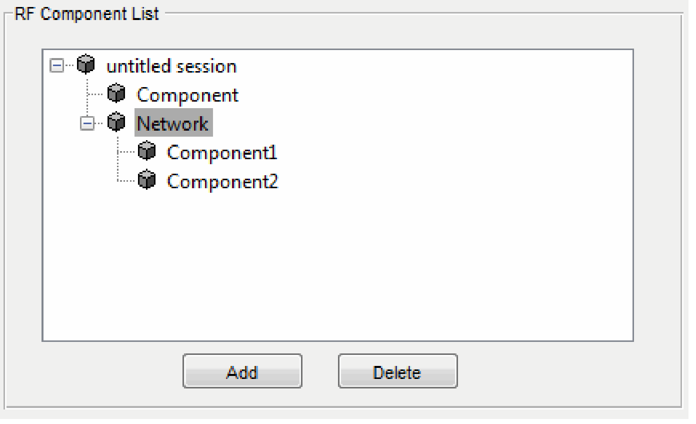
- To perform additional analysis using RF Toolbox functions that are not available in the app.
- To incorporate them into larger RF systems.
- To import them into another session.

Export to the Workspace

The RF Design and Analysis app 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.

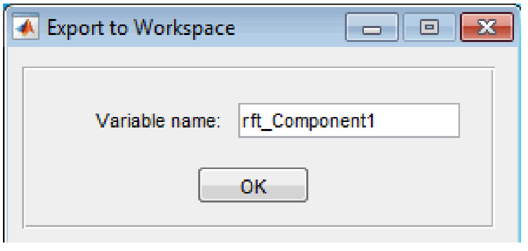
To export a component or network to the workspace:

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

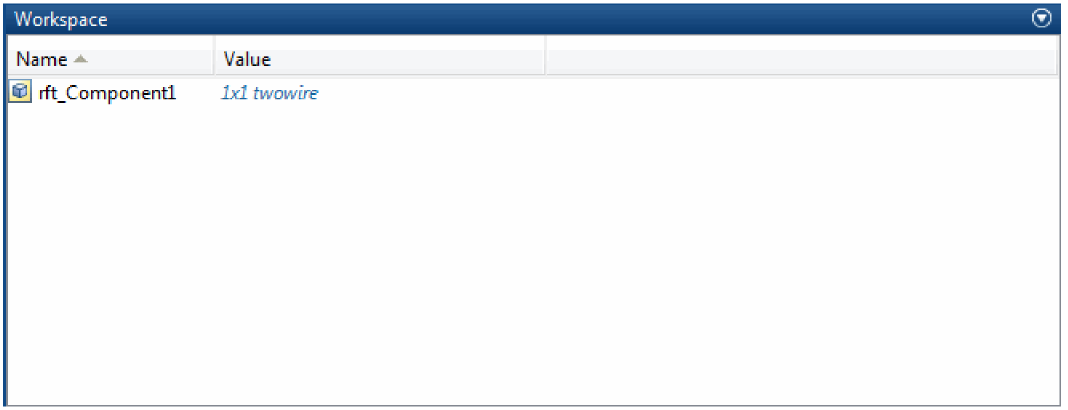


2 Select **Export to Workspace** from the **File** menu.

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 character vector `'rft_'`.



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



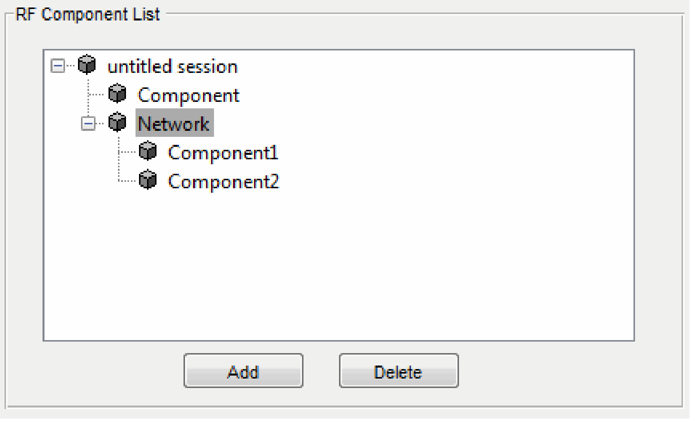
Export to a File

The RF Design and Analysis app lets you export components and networks to files in S2P format.

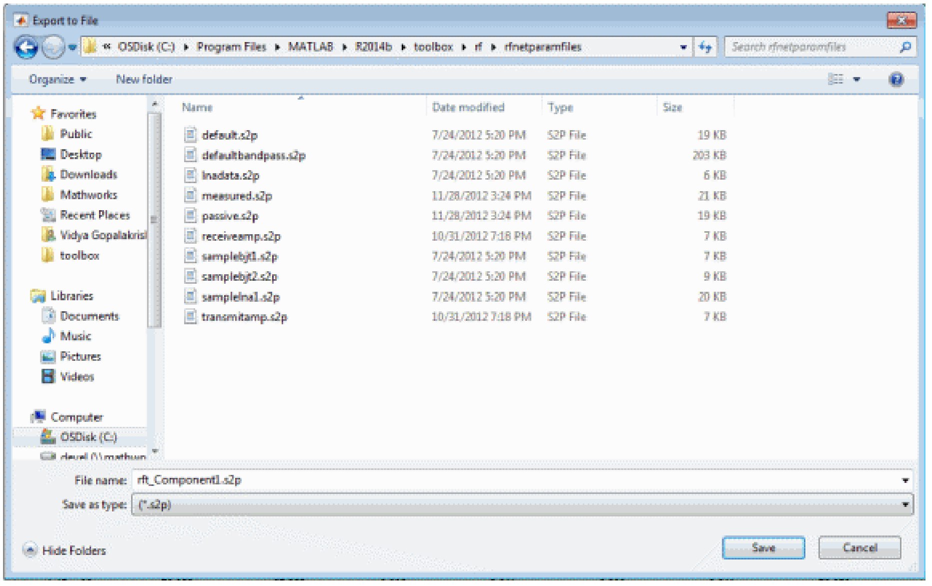
Note You must analyze a component or network in the RF Design and Analysis app before you can export it to a file. See "Analyze Circuits" on page 5-20 for more information.

To export a component or network to a file:

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



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 character vector `'rft_'`. The app also converts any characters that are not alphanumeric to underscores (_).

Manage Circuits and Sessions

In this section...

Working with Circuits

In addition to building and specifying circuits, the RF Design and Analysis app window allows you to perform the following tasks:

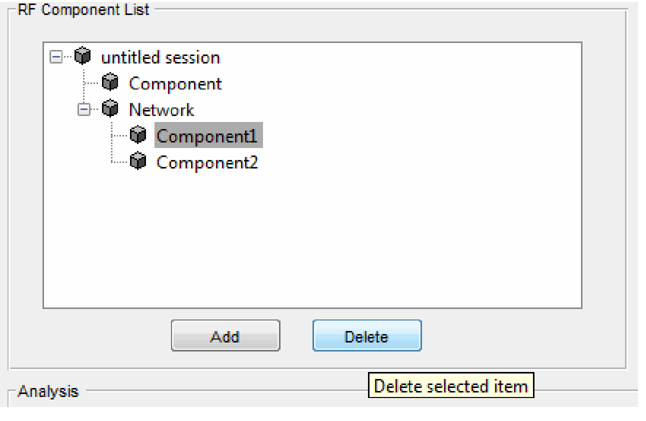
- "Delete a Circuit" on page 5-27
- "Rename a Circuit" on page 5-28

Delete a Circuit

To delete a circuit from your session:

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

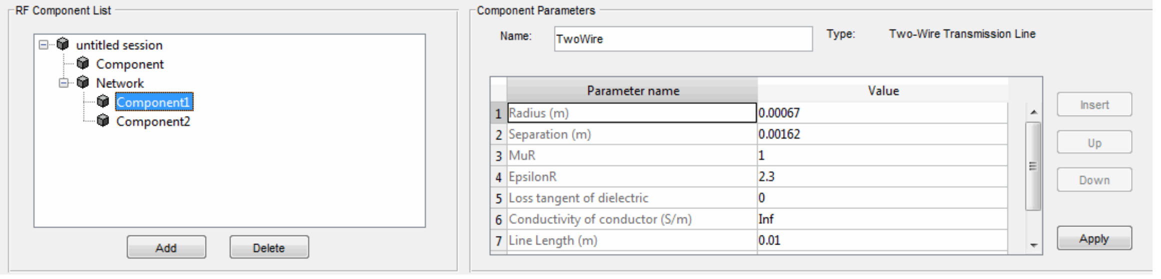
Note If the circuit you delete is a network, the app deletes the network and everything in the network.



Rename a Circuit

To rename a component or a network:

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



Working with the RF Design and Analysis App Sessions

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

This section contains the following topics:

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

Name or Rename a Session

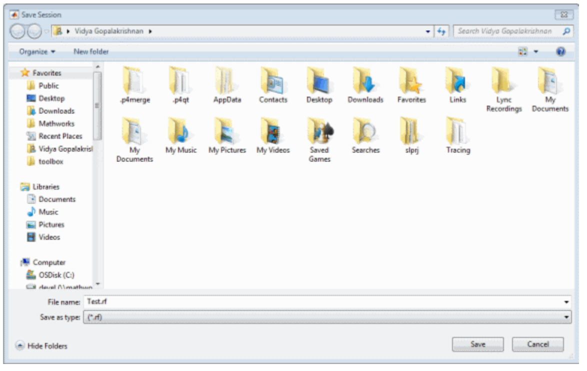
To name or rename a session:

1. Select the session, or top-level node, in the **RF Component List** pane. (The session is selected by default when you open the app user interface.
2. Type the desired name in the **Name** field of the **Component Parameters** pane.
3. Click **Apply**.

Save a Session

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

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

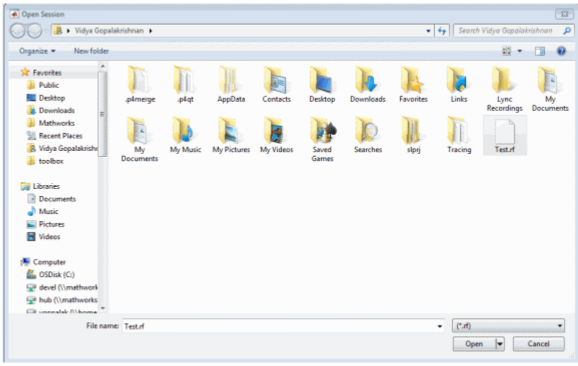


For example, to save your session as `Test.rf` in your current working directory, you would type `Test` in the **File name** field as shown above. The RF Design and Analysis app adds the `.rf` extension automatically to all the app sessions you save.

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

Open a Session

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



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

Start a New Session

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

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

Model an RF Network

In this section...

Overview

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

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 the RF Design and Analysis app.

Start the RF Design and Analysis App

Type the following command at the MATLAB prompt to open the app window:

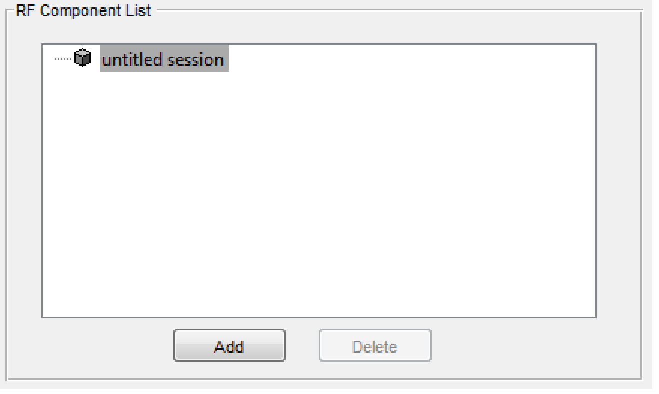
```
rftool
```

For more information about this user interface, see "The RF Design and Analysis Window" on page 5-3.

Create the Amplifier Network

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

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



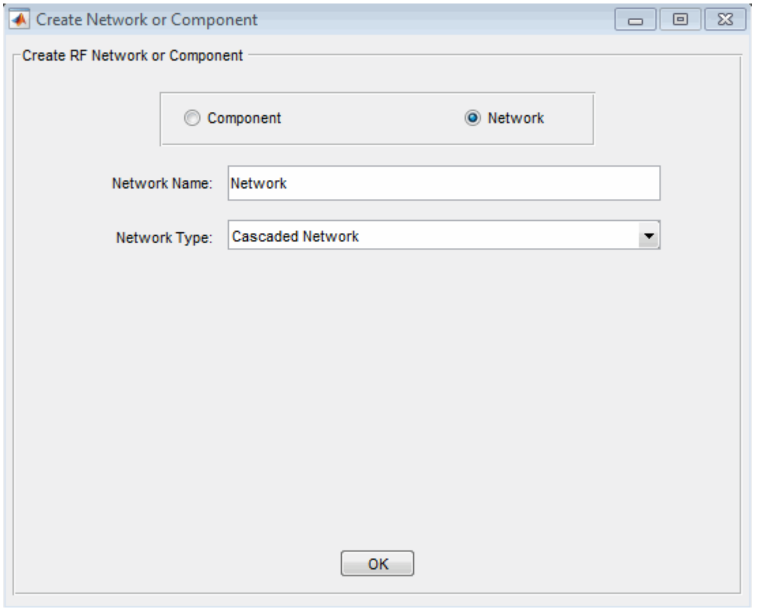
The Create Network or Component dialog box opens.

2 In the Create Network or Component dialog box:

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

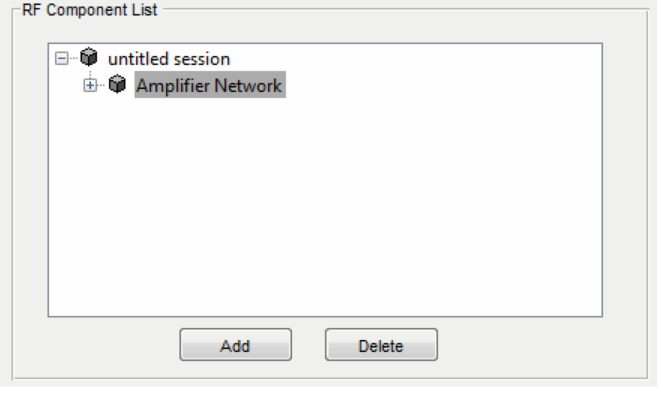
  This name is used to identify the network in the **RF Component List** pane.
- In the **Network Type** list, select `Cascaded Network`.

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



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

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



Populate the Amplifier Network

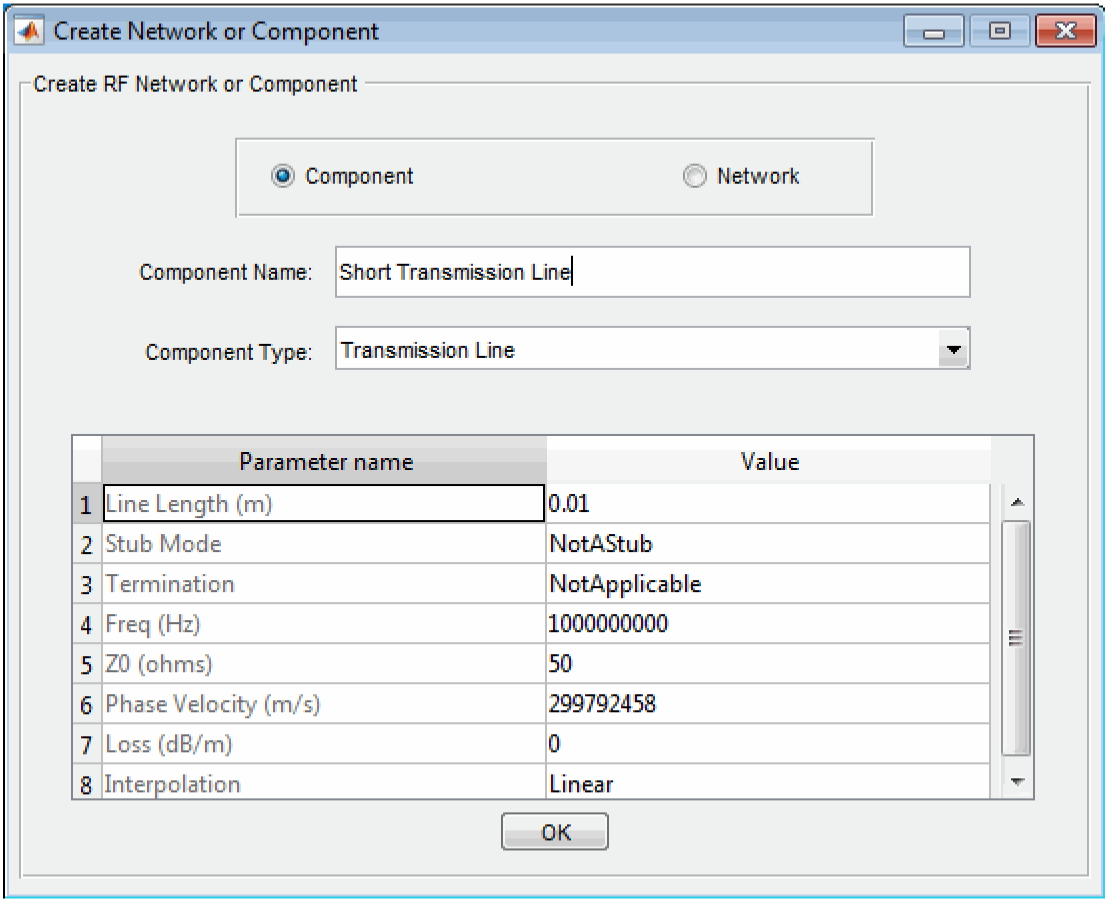
This part of the example shows how to add the following components to the network:

- "Transmission Line 1" on page 5-34
- "Amplifier" on page 5-35
- "Transmission Line 2" on page 5-36

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** drop-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.

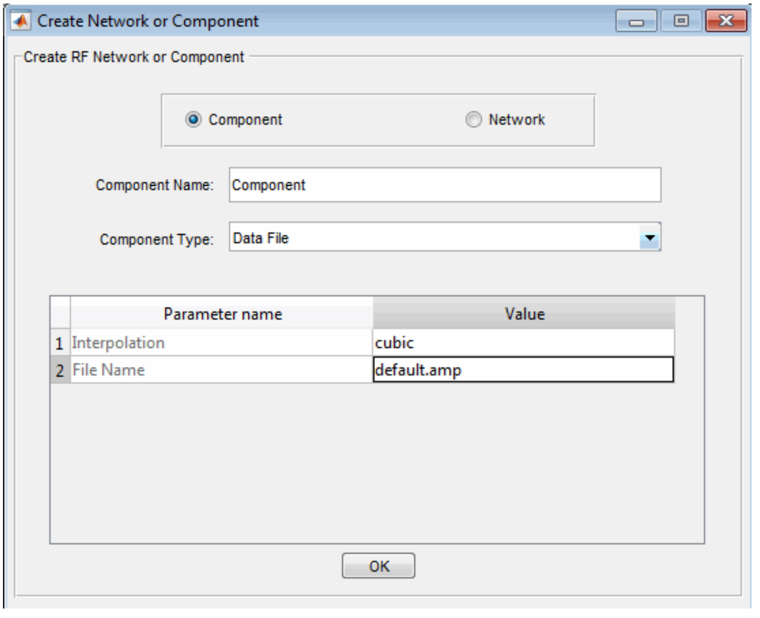
2 In the Insert Component or Network dialog box:

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



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

Transmission Line 2

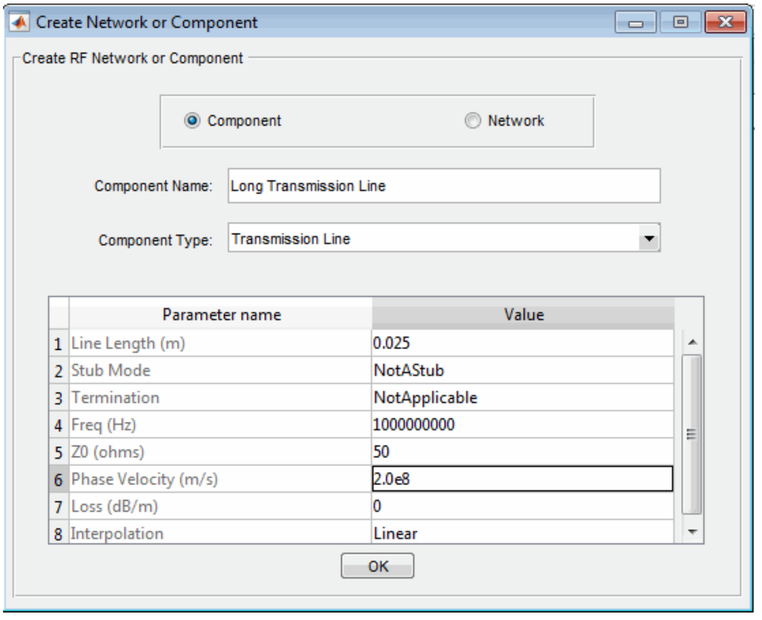
1 In the **Component Parameters** pane, click **Insert** to open the Insert Component or Network dialog box.

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

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

  This name is used to identify the component in the **RF Component List** pane.

- In the **Component Type** list, select `Transmission Line`.
- In the **Value** field across from the **Line Length (m)** parameter, enter `0.025`.
- In the **Value** field across from the **Phase Velocity (m/s)** parameter, enter `2.0e8`.



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

Analyze the Amplifier Network

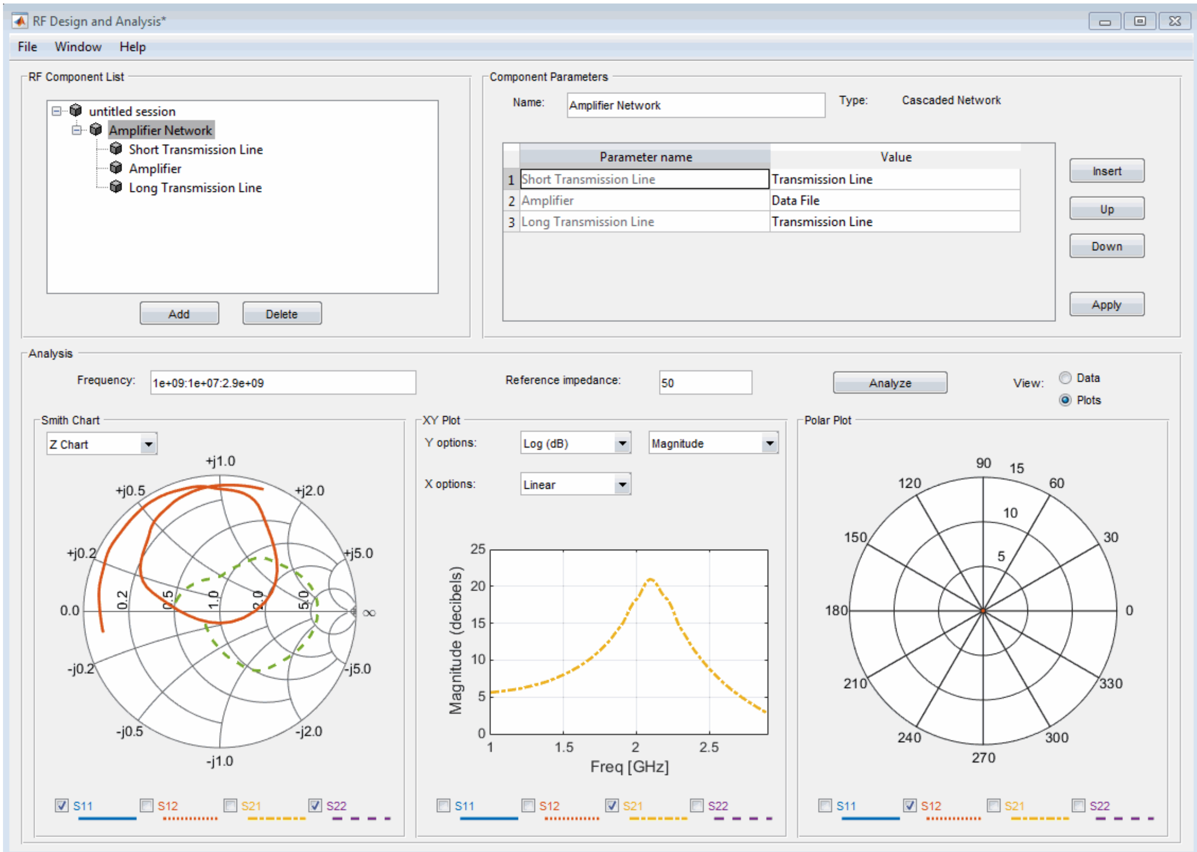
In this part of the example, you specify the range of frequencies over which to analyze the amplifier network and then run the analysis.

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

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

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

The RF Design and Analysis app displays a Smith Chart, an XY plot, and a polar plot of the analyzed circuit.



You can modify the plots by

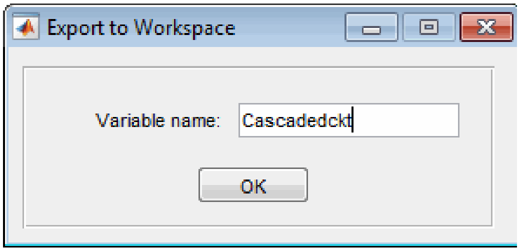
- Selecting and deselecting the S-parameter check boxes at the bottom of each plot to customize the parameters that the plot displays.
- Using the drop-down list at the top of each plot to customize the plot options.

Export the Network to the Workspace

The RF Design and Analysis app 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.

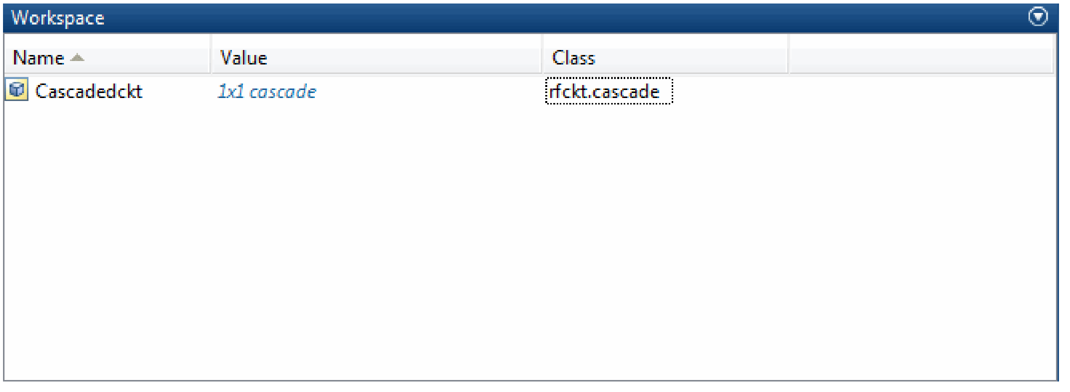
1. In the app window, select **File > Export to Workspace**.
2. In the **Variable name** field, enter `CascadedCkt`.

   This name is the exported object's handle.

3. Click **OK**.

   The RF Design and Analysis app exports the amplifier network to an `rfckt.cascade` object, with the specified object handle, in the MATLAB workspace.

6

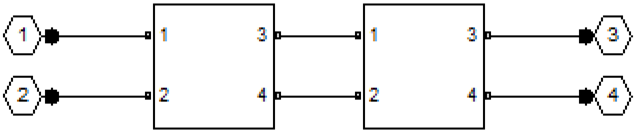
Objects — Alphabetical List

rfckt.cascade

Cascaded network

Description

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



Creation

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,...)`

sets properties using one or more name-value pairs. For example, `rfckt.cascade ('nport',2)`creates a 2-port RF cascade network. You can specify multiple name-value pairs. Enclose each property name in a quote.

Properties

`AnalyzedResult` — Computed S-parameters, noise figure, OIP3, and group delay values

`rfdata.data` object

This property is read-only.

Computed S-parameters, noise figure, OIP3, and group delay values. For more information refer, “Algorithms” on page 6-5.

Data Types: `function_handle`

`Ckts` — Circuit objects in network

cell array of object handles

Circuit objects in network. All circuits must be 2-port. By default, this property is empty.

Data Types: `char`

`Name` — Name of cascaded network

`1-by-N` character array

This property is read-only.

Name of cascaded network.

Data Types: `char`

`nport` — Number of ports of cascaded network

positive integer

This property is read-only.

Number of ports of cascaded network. The default value is 2.

Data Types: `double`

Object Functions

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

circle	Draw circles on Smith Chart
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

Create RF Circuit Cascade Network

Create a cascade network using `rfckt.cascade` with amplifier and transmission lines as elements.

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

```
casccircuit = 
   rfckt.cascade with properties:

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

- “Bandpass Filter Response Using RFCKT Objects”
- “MOS Interconnect and Crosstalk Using RFCKT Objects”

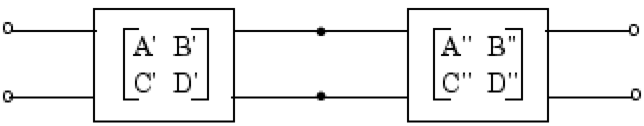
Algorithms

The `analyze` method computes the `AnalyzedResult` property using the data stored in the `Ckts` property as follows:

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

  The `analyze` method then calculates the ABCD-parameters matrix for the cascaded network by calculating the product of the ABCD matrices of the individual networks.

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

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

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

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

- The `analyze` method calculates the noise figure for an N-element cascade. First, the method calculates noise correlation matrices C_A' and C_A”, corresponding to the first two matrices in the cascade, using the following equation:

$$C_A = 2kT \begin{bmatrix} R_n & \dfrac{NF_{\min} - 1}{2} - R_n Y_{opt}{}^{*} \\ \dfrac{NF_{\min} - 1}{2} - R_n Y_{opt} & R_n \left| Y_{opt} \right|^2 \end{bmatrix}$$

  where `k` is Boltzmann's constant, and `T` is the noise temperature in Kelvin.

The method combines C_A' and C_A″ into a single correlation matrix C_A using the equation

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

By applying this equation recursively, the method obtains a noise correlation matrix for the entire cascade. The method then calculates the noise factor, `F`, from the noise correlation matrix of as follows:

$$F = 1 + \frac{z^+ C_A z}{2kT\,\mathrm{Re}\{Z_S\}}$$

$$z = \begin{bmatrix} 1 \\ Z_S{}^* \end{bmatrix}$$

In the two preceding equations, Z_S is the nominal impedance, which is 50 ohms, and z^+ is the Hermitian conjugation of z.

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

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

where G_n is the gain of the nth element of the cascade and $OIP_{3,N}$ is the OIP_3 of the n^{th} element.

- The `analyze` method uses the cascaded S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

References

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

See Also

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

Topics

"Bandpass Filter Response Using RFCKT Objects"
"MOS Interconnect and Crosstalk Using RFCKT Objects"

Introduced before R2006a

rfckt.coaxial

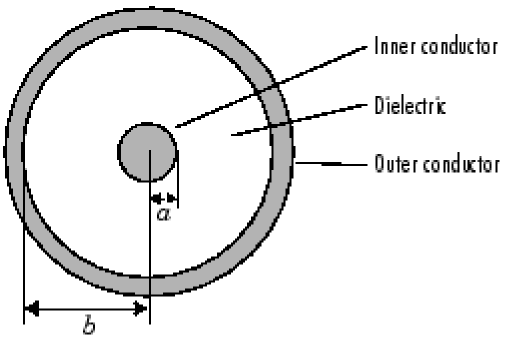
Coaxial transmission line

Description

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

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*.



Creation

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.

Properties

`AnalyzedResult` — Computed S-parameters, noise figure, OIP3, and group delay values
`rfdata.data` object

Computed S-parameters, noise figure, OIP3, and group delay values, specified as an rfdata.data object. This is a read-only property. For more information refer, .

Data Types: `function_handle`

`EpsilonR` — Relative permittivity of dielectric
scalar

Relative permittivity of dielectric, specified as a scalar. The relative permittivity is the ratio of permittivity of the dielectric, ε , to the permittivity in free space, ε_0 . The default value is`2.3`.

Data Types: `double`

`InnerRadius` — Inner conductor radius
scalar

Inner conductor radius, specified as a scalar in meters. The default value is `7.25e-4`.

Data Types: `double`

`LineLength` — Physical length of transmission line
scalar

Physical length of transmission line, specified as a scalar in meters. The default value is `0.01`.

Data Types: `double`

`LossTangent` — Tangent of loss angle of dielectric
scalar

Tangent of loss angle of dielectric, specified as a scalar. The default value is `0`.

Data Types: `double`

`MUR` — Relative permeability of dielectric
scalar

Relative permeability of dielectric, specified as a scalar. The ratio of permeability of dielectric, μ, to the permeability in free space, μ_0. The default value is `1`.

Data Types: `double`

`Name` — Object name
`'Coaxial Transmission Line'` (default) | `1-by-N` character array

Object name, specified as a `1-by-N` character array. This is a read-only property.

Data Types: `char`

`nport` — Number of ports
positive integer

Number of ports, specified as a positive integer.

Data Types: `double`

`OuterRadius` — Outer conductor radius
scalar in meters

Outer conductor radius, specified as a scalar in meters. The default value is `0.0026`.

Data Types: `double`

`SigmaCond` — Conductor conductivity
scalar

Conductor conductivity, specified as a scalar in Siemens per meter (S/m). The default value is `Inf`.

Data Types: `double`

StubMode — Type of stub
'NotaStub' | 'Series' | 'Shunt'

Type of stub, specified as one of the following values: `'NotaStub'`, `'Series'`, `'Shunt'`.

Data Types: `double`

Termination — Stub transmission line termination
'NotApplicable' (default) | 'Open' | 'Short'

Stub transmission line termination, specified as one of the following values: `'NotaStub'`, `'Series'`, `'Shunt'`.

Data Types: `double`

Object Functions

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

Create Coaxial Transmission Line

Create a coaxial transmission line with 0.0045 meters outer radius using rfckt.coaxial.

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

```
tx1 =
   rfckt.coaxial with properties:

       OuterRadius: 0.0045
       InnerRadius: 7.2500e-04
               MuR: 1
          EpsilonR: 2.3000
       LossTangent: 0
         SigmaCond: Inf
        LineLength: 0.0100
          StubMode: 'NotAStub'
       Termination: 'NotApplicable'
             nPort: 2
    AnalyzedResult: []
              Name: 'Coaxial Transmission Line'
```

Algorithms

The `analyze` method treats the transmission line as a 2-port linear network. It computes the `AnalyzedResult` property of a stub or as a stubless line using the data stored in the `rfckt.coaxial` object properties as follows:

- If you model the transmission line as a stubless line, the `analyze` method first calculates the ABCD-parameters at each frequency contained in the modeling frequencies vector. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

  The `analyze` method calculates the ABCD-parameters using the physical length of the transmission line, *d*, and the complex propagation constant, *k*, using the following equations:

$$A = \frac{e^{kd} + e^{-kd}}{2}$$
$$B = \frac{Z_0 * \left(e^{kd} - e^{-kd}\right)}{2}$$
$$C = \frac{e^{kd} - e^{-kd}}{2 * Z_0}$$
$$D = \frac{e^{kd} + e^{-kd}}{2}$$

Z_0 and k are vectors whose elements correspond to the elements of f, the vector of frequencies specified in the `analyze` input argument `freq`. Both can be expressed in terms of the resistance (R), inductance (L), conductance (G), and capacitance (C) per unit length (meters) as follows:

$$Z_0 = \sqrt{\frac{R + j2\pi fL}{G + j2\pi fC}}$$
$$k = k_r + jk_i = \sqrt{(R + j2\pi fL)(G + j2\pi FC)}$$

where

$$R = \frac{1}{2\pi\sigma_{cond}\delta_{cond}}\left(\frac{1}{a} + \frac{1}{b}\right)$$
$$L = \frac{\mu}{2\pi}\ln\left(\frac{b}{a}\right)$$
$$G = \frac{2\pi\omega\varepsilon''}{\ln\left(\frac{b}{a}\right)}$$
$$C = \frac{2\pi\varepsilon}{\ln\left(\frac{b}{a}\right)}$$

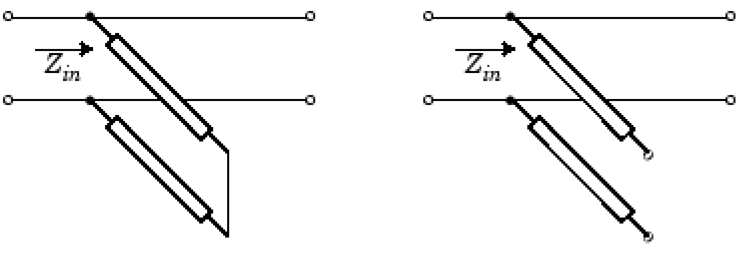
In these equations:

- a is the radius of the inner conductor.

  - b is the radius of the outer conductor.
  - σ_{cond} is the conductivity in the conductor.
  - μ is the permeability of the dielectric.
  - ε is the permittivity of the dielectric.
  - ε'' is the imaginary part of ε, $\varepsilon'' = \varepsilon_0\varepsilon_r\tan\delta$, where:
    - ε_0 is the permittivity of free space.
    - ε_r is the `EpsilonR` property value.
    - $\tan\delta$ is the `LossTangent` property value.
  - δ_{cond} is the skin depth of the conductor, which the method calculates as

    $1/\sqrt{\pi f \mu \sigma_{cond}}$.
  - f is a vector of modeling frequencies determined by the Outport block.
- If you model the transmission line as a shunt or series stub, the `analyze` method first calculates the ABCD-parameters at the specified frequencies. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

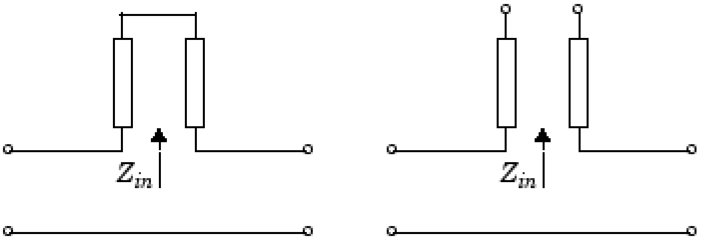
  When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown in the following figure.

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

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

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



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

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

References

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

See Also

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

Introduced before R2006a

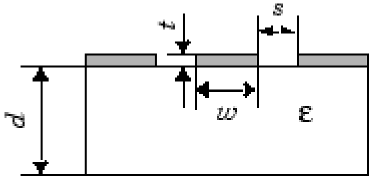
rfckt.cpw

Coplanar waveguide transmission line

Description

Use the `cpw` object 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*).



Creation

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,...)` sets properties using one or more name-value pairs. For example, `rfckt.cpw ('ConductorWidth',` 0.3) creates an RF coplanar waveguide transmission line with a width of 0.3 . You can specify multiple name-value pairs. Enclose each property name in a quote.

Properties

AnalyzedResult — Computed S-parameters, noise figure, OIP3, and group delay values
rfdata.data object

This property is read-only.

Computed S-parameters, noise figure, OIP3, and group delay values, specified as an rfdata.data object. For more information refer, “Algorithms” on page 6-20.

Data Types: function_handle

ConductorWidth — Physical width of conductor
scalar in meters

Physical width of conductor, specified as a scalar in meters. By default, the value is 0.6e-4.

Data Types: double

EpsilonR — Relative permittivity of dielectric
scalar

Relative permittivity of dielectric, specified as a scalar. The relative permittivity is the ratio of permittivity of the dielectric, ε , to the permittivity in free space, ε_0 . By default, the value is 9.8.

Data Types: double

Height — Dielectric thickness or physical height of conductor
scalar in meters

Dielectric thickness or physical height of the conductor, specified as a scalar in meters. The default value is 0.635e-4.

Data Types: double

`LineLength` — Physical length of transmission
scalar in meters

Physical length of transmission, specified as a scalar in meters. The default value is `0.01`.

Data Types: `double`

`LossTangent` — Loss angle tangent of dielectric
scalar

Loss angle tangent of dielectric, specified as a scalar. The default value is `0`.

Data Types: `double`

`Name` — Name of coplanar waveguide transmission line object
`1-by-N` character array

This property is read-only.

Name of coplanar waveguide transmission line object, specified as a `1-by-N` character array.

Data Types: `char`

`nport` — Number of ports
positive integer

Number of ports, specified as a positive integer.

Data Types: `double`

`SigmaCond` — Conductor conductivity
scalar in Siemens per meter

Conductor conductivity, specified as a scalar in Siemens per meter (S/m). The default value is `Inf`.

Data Types: `double`

`StubMode` — Type of stub
`'NotaStub'` | `'Series'` | `'Shunt'`

Type of stub, specified as one of the following values: `'NotaStub'`, `'Series'`, `'Shunt'`.

Data Types: `double`

`SlotWidth` — Physical width of slot
scalar in meters

Physical width of slot, specified as a scalar in meters. The default value is `0.2e-4`.

Data Types: `double`

`Termination` — Stub transmission line termination
`'NotApplicable'` (default) | `'Open'` | `'Short'`

Stub transmission line termination, specified as one of the following values: `'NotaStub'`, `'Series'`, `'Shunt'`.

Data Types: `double`

`Thickness` — Physical thickness of conductor
scalar in meters

Physical thickness of conductor, specified as a scalar in meters. The default value is `0.005e-6`.

Data Types: `double`

Object Functions

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

Create Coplanar Waveguide Transmission Line

Create a coplanar waveguide transmission line using rfckt.cpw.

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

```
tx =
   rfckt.cpw with properties:

    ConductorWidth: 6.0000e-04
         SlotWidth: 2.0000e-04
            Height: 6.3500e-04
         Thickness: 7.5000e-09
          EpsilonR: 9.8000
       LossTangent: 0
         SigmaCond: Inf
        LineLength: 0.0100
          StubMode: 'NotAStub'
       Termination: 'NotApplicable'
             nPort: 2
    AnalyzedResult: []
              Name: 'Coplanar Waveguide Transmission Line'
```

Algorithms

The `analyze` method treats the transmission line as a 2-port linear network. It computes the `AnalyzedResult` property of a stub or as a stub less line using the data stored in the `rfckt.cpw` object properties as follows:

- If you model the transmission line as a stub less line, the `analyze` method first calculates the ABCD-parameters at each frequency contained in the modeling frequencies vector. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

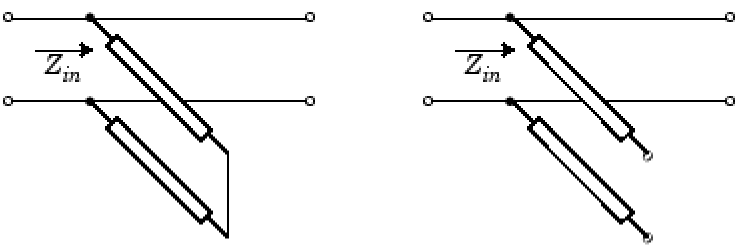
  The `analyze` method calculates the ABCD-parameters using the physical length of the transmission line, *d*, and the complex propagation constant, *k*, using the following equations:

$$A = \frac{e^{kd} + e^{-kd}}{2}$$
$$B = \frac{Z_0 * \left(e^{kd} - e^{-kd}\right)}{2}$$
$$C = \frac{e^{kd} - e^{-kd}}{2 * Z_0}$$
$$D = \frac{e^{kd} + e^{-kd}}{2}$$

Z_0 and k are vectors whose elements correspond to the elements of f, the vector of frequencies specified in the `analyze` input argument `freq`. Both can be expressed in terms of the specified conductor strip width, slot width, substrate height, conductor strip thickness, relative permittivity constant, conductivity and dielectric loss tangent of the transmission line, as described in [1].

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

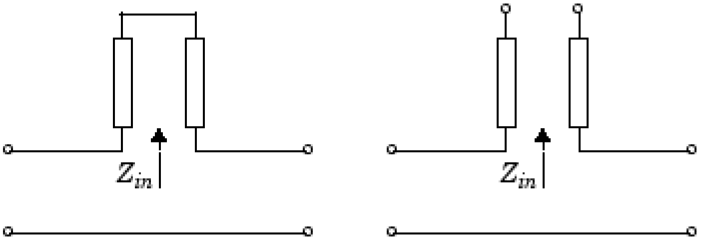
  When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown in the following figure.



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

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

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



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

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

The `analyze` method uses the S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

References

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

See Also

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

Introduced before R2006a

rfckt.datafile

Component or network from file data

Description

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

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

- Touchstone
- Agilent P2D
- Agilent S2D
- AMP

Creation

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,...)` sets properties using one or more name-value pairs. For example, `rfckt.amplfier ('IntType','Cubic')`creates an RF component or network that uses cubic interpolation. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`AnalyzedResult` — Computed S-parameters, noise figure, OIP3, and group delay values
`rfdata.data` object

This property is read-only.

Computed S-parameters, noise figure, OIP3, and group delay values, specified as a rfdata.data object. For more information refer, “Algorithms” on page 6-27.

Data Types: `function_handle`

`File` — File name containing circuit data
`1-by-1` character array

File name containing circuit data, specified as a `1-by-1` character array.

Data Types: `char`

`IntpType` — Interpolation method
`1-by-N` character array

Interpolation method, specified as a `1-by-N` character array of the following values:

Method	Description
`Linear` (default)	Linear interpolation
`Spline`	Cubic spline interpolation
`Cubic`	Piecewise cubic Hermite interpolation

Data Types: `char`

`Name` — Object name
`1-by-N` character array

This property is read-only.

Object name, specified as a `1-by-N` character array.

Data Types: `char`

`nport` — Number of ports
positive integer

This property is read-only.

Number of ports, specified as a positive integer. The default value is 2.

Data Types: `double`

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
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

Represent RF Components and Networks In Data File.

Represent RF components and networks that are characterized by measured or simulated data in a file using rfckt.datafile.

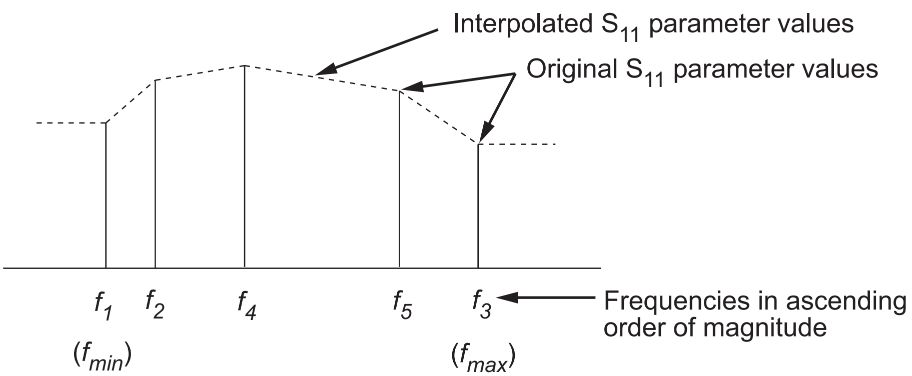
```
data=rfckt.datafile('File','default.s2p')
```

```
data =
   rfckt.datafile with properties:

          IntpType: 'Linear'
              File: 'default.s2p'
             nPort: 2
```

```
    AnalyzedResult: [1x1 rfdata.data]
              Name: 'Data File'
```

Algorithms

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.

References

[1] EIA/IBIS Open Forum, *Touchstone File Format Specification*, Rev. 1.1, 2002

See Also

`rfckt.amplifier` | `rfckt.mixer` | `rfckt.passive`

Introduced before R2006a

rfckt.delay

Delay line

Description

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

Creation

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,...)` sets properties using one or more name-value pairs. For example, `rfckt.delay ('Loss', 2)`creates an RF delay line with a line loss of 2 dB. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`AnalyzedResult` — Computed S-parameters, noise figure, OIP3, and group delay values
`rfdata.data` object

This property is read-only.

Computed S-parameters, noise figure, OIP3, and group delay values, specified as a rfdata.data object. For more information refer, “Algorithms” on page 6-32.

Data Types: `function_handle`

`'Loss'` — Line loss value

positive scalar in dB

Line loss value, specified as a positive scalar in dB. Line loss is the reduction in strength of the signal as it travels over the delay line . The default value is `0`.

Data Types: `double`

`Name` — Object name

`1-by-N` character array

This property is read-only.

Object name, specified as a `1-by-N` character array.

Data Types: `char`

`nport` — Number of ports

positive integer

This property is read-only.

Number of ports, specified as a positive integer. The default value is 2.

Data Types: `double`

`'TimeDelay'` — Amount of time delay

scalar in seconds

Amount of time delay introduced in the line, specified as a scalar in seconds. The default value is `1.0000e-012`.

Data Types: `double`

`'Z0'` — Characteristic impedance

scalar in ohms

Characteristic impedance of the delay line, specified as a scalar in ohms. The default value is `50`.

Data Types: `double`

Object Functions

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

Represent Delay Lines

Represent delay lines that are characterized by line loss and time delay using rfckt.delay.

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

```
del = 
   rfckt.delay with properties:

                Z0: 50.0000 + 0.0000i
              Loss: 0
         TimeDelay: 1.0000e-11
             nPort: 2
    AnalyzedResult: []
              Name: 'Delay Line'
```

Algorithms

The `analyze` method treats the delay line, which can be lossy or lossless, as a 2-port linear network. It computes the `AnalyzedResult` property of the delay line using the data stored in the `rfckt.delay` object properties by calculating the S-parameters for the specified frequencies. This calculation is based on the values of the delay line's `loss`, α, and time delay, D.

$$\begin{cases} S_{11} = 0 \\ S_{12} = e^{-p} \\ S_{21} = e^{-p} \\ S_{22} = 0 \end{cases}$$

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

$$\alpha_a = -\ln\left(10^{\alpha/20}\right)$$

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

$$\beta = 2\pi fD$$

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

References

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

See Also

`rfckt.rlcgline` | `rfckt.txline`

Introduced before R2006a

rfckt.hybrid

Hybrid connected network

Description

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

Creation

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,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`AnalyzedResult` — Computed S-parameters, noise figure, OIP3, and group delay values
`rfdata.data` object

This property is read-only.

Computed S-parameters, noise figure, OIP3, and group delay values, specified as an rfdata.data object. For more information refer, “Algorithms” on page 6-35.

Data Types: `function_handle`

`Ckts` — Circuit objects in network
cell array of object handles

Circuit objects in network, specified as a cell array of object handles. All circuits must be 2-port. By default, this property is empty.

Data Types: `char`

`Name` — Object name
`1-by-N` character array

This property is read-only.

Object name, specified as a `1-by-N` character array.

Data Types: `char`

`nport` — Number of ports
positive integer

This property is read-only.

Number of ports, specified as a positive integer. The default value is 2.

Data Types: `double`

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
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

Create Hybrid Connected Networks

Create hybrid connected networks of linear RF objects with two transmission line objects using rfckt.hybrid.

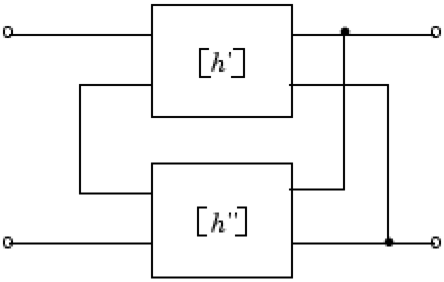
```
tx1 = rfckt.txline;
tx2 = rfckt.txline;
hyb = rfckt.hybrid('Ckts',{tx1,tx2})
```

```
hyb =
   rfckt.hybrid with properties:

              Ckts: {[1x1 rfckt.txline]  [1x1 rfckt.txline]}
             nPort: 2
    AnalyzedResult: []
              Name: 'Hybrid Connected Network'
```

Algorithms

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 parameters to an *h* matrix. The following figure shows a hybrid connected network consisting of two 2-port networks, each represented by its *h* matrix,



where

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

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

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

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

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

References

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

See Also

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

Introduced before R2006a

rfckt.hybridg

Inverse hybrid connected network

Description

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

Creation

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,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`AnalyzedResult` — Computed S-parameters, noise figure, OIP3, and group delay values
`rfdata.data` object

This property is read-only.

Computed S-parameters, noise figure, OIP3, and group delay values, specified as an rfdata.data object. For more information refer, “Algorithms” on page 6-40.

Data Types: `function_handle`

`Ckts` — Circuit objects in network
cell array of object handles

Circuit objects in network, specified as a cell array of object handles. All circuits must be 2-port. By default, this property is empty.

Data Types: `char`

`Name` — Object name
`1-by-N` character array

This property is read-only.

Object name, specified as a `1-by-N` character array.

Data Types: `char`

`nport` — Number of ports
positive integer

This property is read-only.

Number of ports, specified as a positive integer. The default value is 2.

Data Types: `double`

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
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

Create Inverse Hybrid Connected Networks

Create inverse hybrid connected networks of linear RF objects with two transmission line objects using rfckt.hybridg.

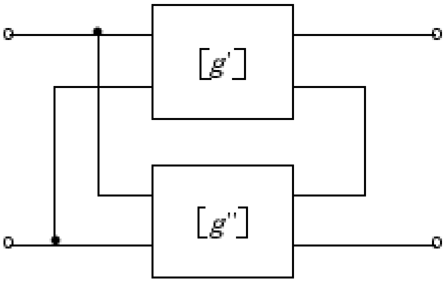
```
tx1 = rfckt.txline;
tx2 = rfckt.txline;
invhyb = rfckt.hybridg('Ckts',{tx1,tx2})
```

```
invhyb =
   rfckt.hybridg with properties:

              Ckts: {[1x1 rfckt.txline]  [1x1 rfckt.txline]}
             nPort: 2
    AnalyzedResult: []
              Name: 'Hybrid G Connected Network'
```

Algorithms

The `analyze` method computes the S-parameters of the `AnalyzedResult` property using the data stored in the `Ckts` property as follows:

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



where

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

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

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

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

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

References

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

See Also

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

Introduced before R2006a

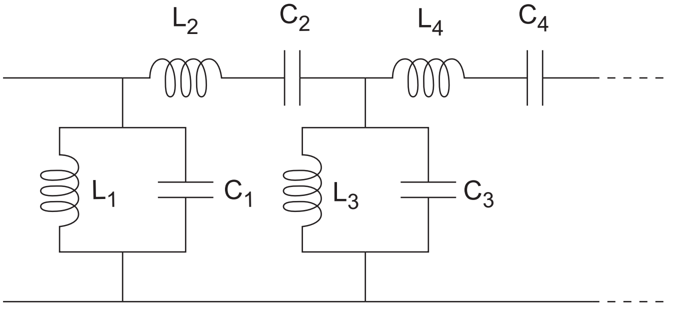
rfckt.lcbandpasspi

Bandpass pi filter

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.

Creation

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,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`'AnalyzedResult'` — Computed S-parameters, noise figure, OIP3, and group delay values
`rfdata.data` object

Computed S-parameters, noise figure, OIP3, and group delay values, specified as a comma-separated pair consisting of `'AnalyzedResult'` and rfdata.data object. The values are computed over the specified frequency range. By default, this property is empty.

Data Types: `function_handle`

`'C'` — Capacitance value
positive vector in farads

Capacitance value from source to load of all capacitors in the network, specified as a comma separated pair consisting of `'C'` and a positive vector in farads. The length of the capacitance vector must be equal to the length of the vector you provide for `'L'`. The default value is `[0.3579e-10, 0.0118e-10, 0.3579e-10]`.

Data Types: `double`

`'L'` — Inductance value
positive vector

Inductance value from source to load of all inductors in the network, specified as a comma separated pair consisting of `'L'` and a positive vector in henries. The length of the inductance vector must be equal to the length of the vector you provide for `'C'`. The default value is `[0.0144e-7, 0.4395e-7, 0.0144e-7]`.

Data Types: `double`

'Name' — Object name
'LC Bandpass Pi' (default) | 1-by-N character array

Object name, specified as a comma-separated pair consisting of 'Name' and 1-by-N character array. This is a read-only property.

Data Types: char

'nport' — Number of ports
positive integer

Number of ports, specified as a comma-separated pair consisting of 'nport' and a positive integer. This is a read-only property. The default value is 2.

Data Types: double

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
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

Create LC BandPass Pi Filter

Create an LC bandpass filter of capacitor values 1e-12 and 4e12 farads, inductor values 2e-9 and 2.5e-9 henries.

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

filter =
   rfckt.lcbandpasspi with properties:

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

References

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

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

See Also

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

Introduced before R2006a

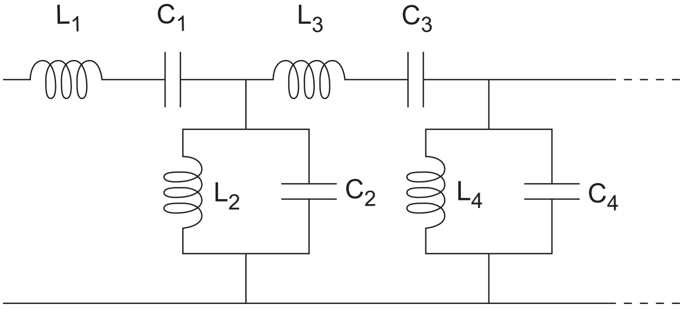
rfckt.lcbandpasstee

Bandpass tee filter

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.

Creation

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,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`'AnalyzedResult'` — Computed S-parameters, noise figure, OIP3, and group delay values

`rfdata.data` object

Computed S-parameters, noise figure, OIP3, and group delay values, specified as rfdata.data object. The values are computed over the specified frequency range. By default, this property is empty.

Data Types: `function_handle`

`'C'` — Capacitance value

positive vector in farads

Capacitance value from source to load of all capacitors in the network, specified as a positive vector in farads. The length of the capacitance vector must be equal to the length of the vector you provide for `'L'`. The default value is `[0.0186e-10, 0.1716e-10, 0.0186e-10]`.

Data Types: `double`

`'L'` — Inductance value

positive vector in henries

Inductance value from source to load of all inductors in the network, specified as a positive vector in henries. The length of the inductance vector must be equal to the length of the vector you provide for `'C'`. The default value is `[0.2781e-7, 0.0301e-7, 0.2781e-7]`.

Data Types: `double`

'Name' — Object name
'LC Bandpass Tee' (default) | 1-by-N character array

Object name, specified as a 1-by-N character array. This is a read-only property.

Data Types: char

'nport' — Number of ports
positive integer

Number of ports, specified as a positive integer. This is a read-only property. The default value is 2.

Data Types: double

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
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

LC Bandpass Tee Filter

Create a LC Bandpass Tee Filter using `rfckt.lcbandpasstee`.

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

```
filter =
   rfckt.lcbandpasstee with properties:

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

References

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

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

See Also

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

Introduced before R2006a

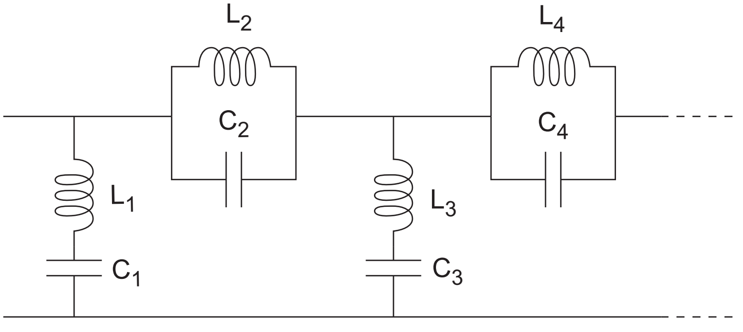
rfckt.lcbandstoppi

Bandstop pi filter

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.

Creation

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,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`'AnalyzedResult'` — Computed S-parameters, noise figure, OIP3, and group delay values

`rfdata.data` object

Computed S-parameters, noise figure, OIP3, and group delay values, specified as rfdata.data object. The values are computed over the specified frequency range. By default, this property is empty.

Data Types: `function_handle`

`'C'` — Capacitance value

positive vector in farads

Capacitance value from source to load of all capacitors in the network, specified as a positive vector in farads. The length of the capacitance vector must be equal to the length of the vector you provide for `'L'`. The default value is `[0.0184e-10, 0.2287e-10, 0.0184e-10]`.

Data Types: `double`

`'L'` — Inductance value

positive vector in henries

Inductance value from source to load of all inductors in the network, specified as a positive vector in henries. The length of the inductance vector must be equal to the length of the vector you provide for `'C'`. The default value is `[0.2809e-7, 0.0226e-7, 0.2809e-7]`.

Data Types: `double`

'Name' — Object name

'LC Bandstop Pi' (default) | 1-by-N character array

Object name, specified as a `1-by-N` character array. This is a read-only property.

Data Types: `char`

'nport' — Number of ports

positive integer

Number of ports, specified as a positive integer. This is a read-only property. The default value is 2.

Data Types: `double`

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
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

LC Bandstop Pi Filter

Create a LC Bandstop Pi Filter using `rfckt.lcbandstoppi`.

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

```
filter =
   rfckt.lcbandstoppi with properties:

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

References

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

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

See Also

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

Introduced before R2006a

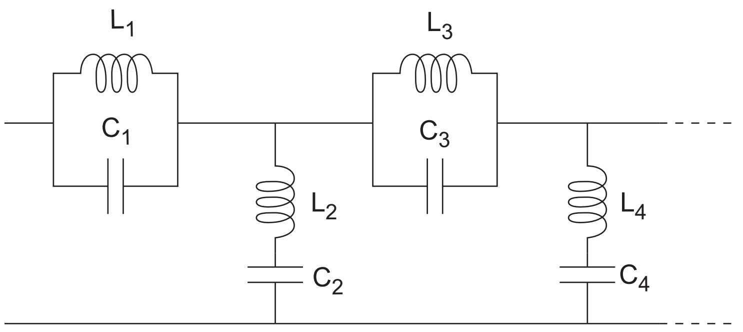
rfckt.lcbandstoptee

Bandstop tee filter

Description

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

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



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

Creation

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,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`'AnalyzedResult'` — Computed S-parameters, noise figure, OIP3, and group delay values

`rfdata.data` object

Computed S-parameters, noise figure, OIP3, and group delay values, specified as rfdata.data object. The values are computed over the specified frequency range. By default, this property is empty.

Data Types: `function_handle`

`'C'` — Capacitance value

positive vector in farads

Capacitance value from source to load of all capacitors in the network, specified as a positive vector in farads. The length of the capacitance vector must be equal to the length of the vector you provide for `'L'`. The default value is `[0.0186e-10, 0.1716e-10, 0.0186e-10]`.

Data Types: `double`

`'L'` — Inductance value

positive vector in henries

Inductance value from source to load of all inductors in the network, specified as a positive vector in henries. The length of the inductance vector must be equal to the length of the vector you provide for `'C'`. The default value is `[0.2781e-7, 0.0301e-7, 0.2781e-7]`.

Data Types: `double`

'Name' — Object name
'LC Bandstop Tee' (default) | 1-by-N character array

Object name, specified as a 1-by-N character array. This is a read-only property.

Data Types: char

'nport' — Number of ports
positive integer

Number of ports, specified as a positive integer. This is a read-only property. The default value is 2.

Data Types: double

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
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

LC Bandstop Tee Filter

Create a LC Bandstop Tee Filter using rfckt.lcbandstoptee.

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

```
filter =
   rfckt.lcbandstoptee with properties:

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

References

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

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

See Also

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

Introduced before R2006a

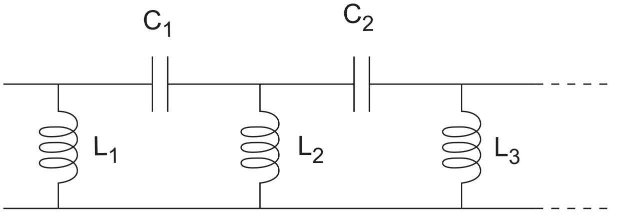
rfckt.lchighpasspi

Highpass pi filter

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.

Creation

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,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`'AnalyzedResult'` — Computed S-parameters, noise figure, OIP3, and group delay values
`rfdata.data` object

Computed S-parameters, noise figure, OIP3, and group delay values, specified as rfdata.data object. The values are computed over the specified frequency range. By default, this property is empty.

Data Types: `function_handle`

`'C'` — Capacitance value
positive vector in farads

Capacitance value from source to load of all capacitors in the network, specified as a positive vector in farads. The default value is `[0.1188e-5, 0.1188e-5]`.

Data Types: `double`

`'L'` — Inductance value
positive vector in henries

Inductance value from source to load of all inductors in the network, specified as a positive vector in henries. The default value is `[2.2363e-9]`.

Data Types: `double`

`'Name'` — Object name
`'LC Highpass Pi'` (default) | `1-by-N` character array

Object name, specified as a `1-by-N` character array. This is a read-only property.

Data Types: `char`

`'nport'` — Number of ports
positive integer

Number of ports, specified as a positive integer. This is a read-only property. The default value is 2.

Data Types: `double`

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
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

LC Highpass Pi Filter

Create a LC Highpass Pi Filter using `rfckt.lchighpasspi`.

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

```
filter = 
   rfckt.lchighpasspi with properties:

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

References

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

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

See Also

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

Introduced before R2006a

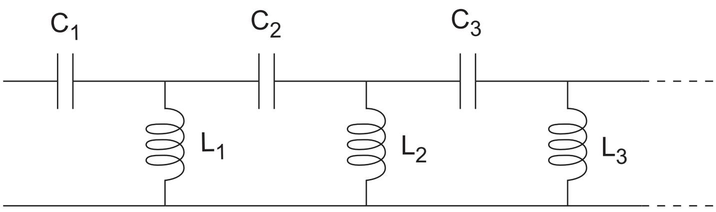
rfckt.lchighpasstee

Highpass tee filter

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.

Creation

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,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`'AnalyzedResult'` — Computed S-parameters, noise figure, OIP3, and group delay values
`rfdata.data` object

Computed S-parameters, noise figure, OIP3, and group delay values, specified as rfdata.data object. The values are computed over the specified frequency range. By default, this property is empty.

Data Types: `function_handle`

`'C'` — Capacitance value
positive vector in farads

Capacitance value from source to load of all capacitors in the network, specified as a positive vector in farads. The default value is `[[0.4752e-9, 0.4752e-9]`.

Data Types: `double`

`'L'` — Inductance value
positive vector in henries

Inductance value from source to load of all inductors in the network, specified as a positive vector in henries. The default value is `[5.5907e-6]`.

Data Types: `double`

`'Name'` — Object name
`'LC Highpass Tee'` (default) | `1-by-N` character array

Object name, specified as a `1-by-N` character array. This is a read-only property.

Data Types: `char`

`'nport'` — Number of ports
positive integer

Number of ports, specified as a positive integer. This is a read-only property. The default value is 2.

Data Types: double

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
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

LC Highpass Tee Filter

Create a LC Highpass Tee Filter using `rfckt.lchighpasstee`.

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

```
filter = 
   rfckt.lchighpasstee with properties:

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

References

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

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

See Also

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

Introduced before R2006a

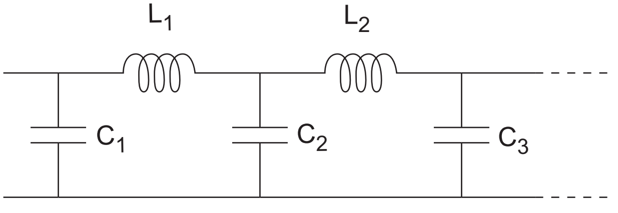
rfckt.lclowpasspi

Lowpass pi filter

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.

Creation

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,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`'AnalyzedResult'` — Computed S-parameters, noise figure, OIP3, and group delay values

`rfdata.data` object

Computed S-parameters, noise figure, OIP3, and group delay values, specified as rfdata.data object. The values are computed over the specified frequency range. By default, this property is empty.

Data Types: `function_handle`

`'C'` — Capacitance value

positive vector in farads

Capacitance value from source to load of all capacitors in the network, specified as a positive vector in farads. The default value is `[0.5330e-8, 0.5330e-8]`.

Data Types: `double`

`'L'` — Inductance value

positive vector in henries

Inductance value from source to load of all inductors in the network, specified as a positive vector in henries. The default value is `[2.8318e-6]`.

Data Types: `double`

`'Name'` — Object name

`'LC Lowpass Pi'` (default) | `1-by-N` character array

Object name, specified as a `1-by-N` character array. This is a read-only property.

Data Types: `char`

`'nport'` — Number of ports

positive integer

Number of ports, specified as a positive integer. This is a read-only property. By default, the value is 2.

Data Types: `double`

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
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

LC Lowpass Pi Filter

Create a LC lowpass pi Filter using `rfckt.lclowpasspi`.

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

```
filter =
   rfckt.lclowpasspi with properties:

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

References

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

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

See Also

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

Introduced before R2006a

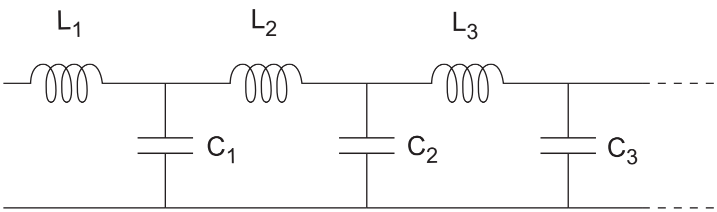
rfckt.lclowpasstee

Lowpass tee filter

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.

Creation

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,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`'AnalyzedResult'` — Computed S-parameters, noise figure, OIP3, and group delay values

`rfdata.data` object

Computed S-parameters, noise figure, OIP3, and group delay values, specified as rfdata.data object. The values are computed over the specified frequency range. By default, this property is empty.

Data Types: `function_handle`

`'C'` — Capacitance value

positive vector in farads

Capacitance value from source to load of all capacitors in the network, specified as a positive vector in farads. The default value is `[1.1327e-9]`.

Data Types: `double`

`'L'` — Inductance value

positive vector in henries

Inductance value from source to load of all inductors in the network, specified as a positive vector in henries. The default value is `[0.1332e-4, 0.1332e-4]`.

Data Types: `double`

`'Name'` — Object name

`'LC Lowpass Tee'` (default) | `1-by-N` character array

Object name, specified as a `1-by-N` character array. This is a read-only property.

Data Types: `char`

`'nport'` — Number of ports

positive integer

Number of ports, specified as a positive integer. This is a read-only property. The default value is 2.

Data Types: `double`

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
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

LC Lowpass Tee Filter

Create a LC lowpass tee Filter using `rfckt.lclowpasstee`.

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

```
filter = 
   rfckt.lclowpasstee with properties:

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

References

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

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

See Also

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

Introduced before R2006a

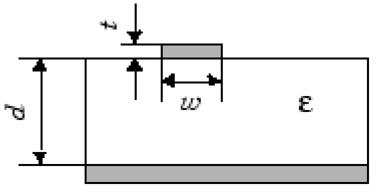
rfckt.microstrip

Microstrip transmission line

Description

Use the `microstrip` class to represent microstrip transmission lines 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*).



Creation

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,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`AnalyzedResult` — Computed S-parameters, noise figure, OIP3, and group delay values
`rfdata.data` object

Computed S-parameters, noise figure, OIP3, and group delay values, specified as rfdata.data object. `Analyzed Result` is a read-only property. For more information refer, “Algorithms” on page 6-78

Data Types: `function_handle`

`EpsilonR` — Relative permittivity of dielectric
scalar

Relative permittivity of dielectric, specified as a scalar. The relative permittivity is the ratio of permittivity of the dielectric, ε, to the permittivity in free space, ε_0. The default value is `9.8`.

Data Types: `double`

`Height` — Dielectric thickness or physical height of conductor
scalar

Dielectric thickness or physical height of the conductor, specified as a scalar in meters. The default value is `6.35e-4`.

Data Types: `double`

`LineLength` — Physical length of transmission
scalar

Physical length of transmission, specified as a scalar in meters. The default value is `0.01`.

Data Types: `double`

`LossTangent` — Loss angle tangent of dielectric
scalar

Loss angle tangent of dielectric, specified as a scalar. The default value is `0`.

Data Types: `double`

Name — Object name
'Microstrip Waveguide Transmission Line' (default) | 1-by-N character array

Object name, specified as an 1-by-N character array. Name is a read-only property.

Data Types: char

nport — Number of ports
positive integer

Number of ports, specified as a positive integer. nportt is a read-only property. The default value is 2.

Data Types: double

SigmaCond — Conductor conductivity
scalar

Conductor conductivity, specified as a scalar in Siemens per meter (S/m). The default value is Inf.

Data Types: double

StubMode — Type of stub
'NotaStub' (default) | 'Series' | 'Shunt'

Type of stub, specified as one of the following values: 'NotaStub', 'Series', 'Shunt'.

Data Types: double

Termination — Stub transmission line termination
'NotApplicable' (default) | 'Open' | 'Short'

Stub transmission line termination, specified as one of the following values: 'NotaStub', 'Series', 'Shunt'.

Data Types: double

Thickness — Physical thickness of microstrip
scalar

Physical thickness of microstrip, specified as a scalar in meters. The default value is 5.0e-6.

Data Types: double

`Width` — Physical width of parallel-plate
scalar

Physical width of parallel-plate, specified as a scalar in meters. The default value is `6.0e-4`.

Data Types: `double`

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
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

Microstrip Transmission Line

Create a microstrip transmission line using `rfckt.microstrip`.

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

```
tx1 = 
   rfckt.microstrip with properties:

            Width: 6.0000e-04
           Height: 6.3500e-04
        Thickness: 7.5000e-09
         EpsilonR: 9.8000
```

```
    LossTangent: 0
      SigmaCond: Inf
     LineLength: 0.0100
       StubMode: 'NotAStub'
    Termination: 'NotApplicable'
          nPort: 2
 AnalyzedResult: []
           Name: 'Microstrip Transmission Line'
```

Algorithms

The `analyze` method treats the microstrip line as a 2-port linear network and models the line as a transmission line with optional stubs. The `analyze` method computes the `AnalyzedResult` property of the transmission line using the data stored in the `rfckt.microstrip` object properties as follows:

- If you model the transmission line as a stubless line, the `analyze` method first calculates the ABCD-parameters at each frequency contained in the modeling frequencies vector. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

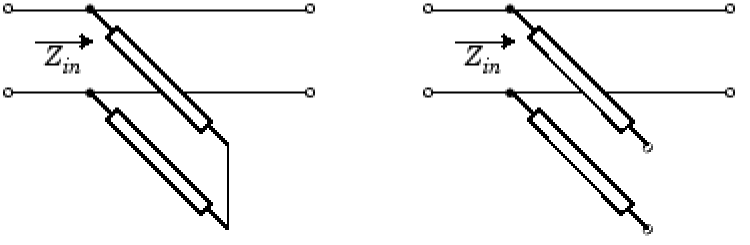
  The `analyze` method calculates the ABCD-parameters using the physical length of the transmission line, *d*, and the complex propagation constant, *k*, using the following equations:

$$A = \frac{e^{kd} + e^{-kd}}{2}$$
$$B = \frac{Z_0 * \left(e^{kd} - e^{-kd}\right)}{2}$$
$$C = \frac{e^{kd} - e^{-kd}}{2 * Z_0}$$
$$D = \frac{e^{kd} + e^{-kd}}{2}$$

  Z_0 and k are vectors whose elements correspond to the elements of f, the vector of frequencies specified in the `analyze` input argument `freq`. Both can be expressed in terms of the specified conductor strip width, substrate height, conductor strip thickness, relative permittivity constant, conductivity, and dielectric loss tangent of the microstrip line, as described in [1].

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

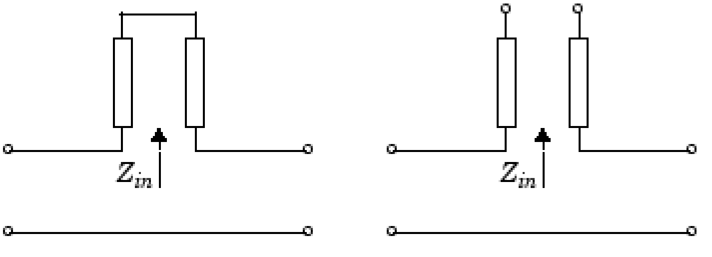
  When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit.

  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.

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

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

References

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

See Also

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

Introduced before R2006a

rfckt.mixer

2-port representation of RF mixer and its local oscillator

Description

Use the `mixer` class to represent RF mixers and their local oscillators characterized by network parameters, noise data, nonlinearity data, and local oscillator frequency.

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

Note If you set `NonLinearData` using `rfdata.ip3` or `rfdata.power`, then the property is converted from scalar OIP3 format to the format of `rfdata.ip3` or `rfdata.power`.

Creation

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,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`AnalyzedResult` — Computed S-parameters, noise figure, OIP3, and group delay values

`rfdata.data` object

Computed S-parameters, noise figure, OIP3, and group delay values, specified as rfdata.data object. `Analyzed Result` is a read-only property. For more information refer, "Algorithms" on page 6-85.

Data Types: `function_handle`

`FLO` — Local oscillator frequency

positive scalar

Local oscillator frequency, specified as a positive scalar in hertz. If the `MixerType` is set to `'DownConverter'`, the mixer output frequency is $f_{out} = f_{in} - f_{lo}$. If the `MixerType` is set to `'UpConverter'`, the mixer output frequency is $f_{out} = f_{in} + f_{lo}$.

Data Types: `double`

`FreqOffset` — Frequency offset data

positive vector

Frequency offset data, specified as a positive vector in hertz. The `'FreqOffset'` values correspond to phase noise level values specified by the `'PhaseNoiseLevel'` property. By default, this property is empty.

Data Types: `double`

`IntpType` — Interpolation method used in `rfckt.mixer`

`1-by-N` character array

Interpolation method used in `rfckt.mixer`, specified as a `1-by-N` character array of the following values:

Method	Description
`Linear` (default)	Linear interpolation
`Spline`	Cubic spline interpolation
`Cubic`	Piecewise cubic Hermite interpolation

Data Types: `char`

`MixerSpurData` — Data from mixer spur table
`rfdata.mixerspur` object

Data from mixer spur table, specified as an `rfdata.mixerspur` object.

Data Types: `function_handle`

`MixerType` — Type of mixer
`'DownConverter'` (default) | `'UpConverter'`

Type of mixer, specified as `'DownConverter'` or `'UpConverter'`.

Data Types: `char`

`Name` — Object name
`1-by-N` character array

Object name, specified as an `1-by-N` character array. `Name` is a read-only property.

Data Types: `char`

`NoiseData` — Noise information
scalar noise figure in decibels | `rfdata.noise` object | `rfdata.nf` object

Noise information, specified as one of the following:

- Scalar noise figure in dB
- `rfdata.noise` object
- `rfdata.nf` object

Data Types: `double` | `function_handle`

`NonlinearData` — Nonlinearity information
scalar OIP3 in dB | `rfdata.power` object | `rfdata.ip3` object

Noise information, specified as one of the following:

- Scalar OIP3 in dB
- `rfdata.power` object
- `rfdata.ip3` object

Data Types: `double` | `function_handle`

`nport` — Number of ports
positive integer

Number of ports, specified as a positive integer. `nportt` is a read-only property. The default value is `2`.

Data Types: `double`

`PhaseNoiseLevel` — Phase noise data
vector

Phase noise data, specified as a vector in dbc/Hz.

Data Types: `double`

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
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

RF Mixer

Create an RF mixer using `rfckt.mixer`.

```
rfmixer = rfckt.mixer('IntpType','cubic')
```

```
rfmixer = 
   rfckt.mixer with properties:

      MixerSpurData: []
          MixerType: 'Downconverter'
                FLO: 1.0000e+09
         FreqOffset: []
    PhaseNoiseLevel: []
          NoiseData: [1x1 rfdata.noise]
      NonlinearData: Inf
           IntpType: 'Cubic'
        NetworkData: [1x1 rfdata.network]
              nPort: 2
     AnalyzedResult: [1x1 rfdata.data]
               Name: 'Mixer'
```

Algorithms

The `analyze` method computes the `AnalyzedResult` property using the data stored in the `rfckt.amplifier` object properties as follows:

- The `analyze` method uses the data stored in the `'NoiseData'` property of the `rfckt.amplifier` object to calculate the noise figure.
- The `analyze` method uses the data stored in the `'NonlinearData'` property of the `rfckt.amplifier` object to calculate OIP3.

  If power data exists in the `'NonlinearData'` property, the block extracts the AM/AM and AM/PM nonlinearities from the power data.

  If the `'NonlinearData'` property contains only IP3 data, the method computes and adds the nonlinearity by:

1 Using the third-order input intercept point value in dBm to compute the factor, *f*, that scales the input signal before the amplifier object applies the nonlinearity:

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

2 Computing the scaled input signal by multiplying the amplifier input signal by *f*.

3 Limiting the scaled input signal to a maximum value of 1.

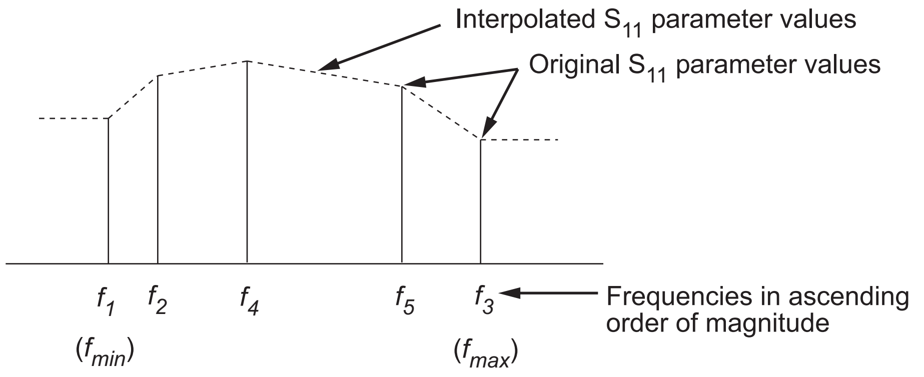
4 Applying an AM/AM conversion to the amplifier gain, according to the following cubic polynomial equation:

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

where *u* is the magnitude of the scaled input signal, which is a unitless normalized input voltage.

- The `analyze` method uses the data stored in the `'NetworkData'` property of the `rfckt.amplifier` object to calculate the group delay values of the amplifier at the frequencies specified in `freq`, as described in the `analyze` reference page.
- The `analyze` method uses the data stored in the `'NetworkData'` property of the `rfckt.amplifier` object to calculate the S-parameter values of the amplifier at the frequencies specified in `freq`. If the `'NetworkData'` property contains network Y- or Z-parameters, the `analyze` method first converts the parameters to S-parameters. Using the interpolation method you specify with the `'IntpType'` property, the `analyze` method interpolates the S-parameter values to determine their values at the specified frequencies.

  Specifically, the `analyze` method orders the S-parameters according to the ascending order of their frequencies, f_n. It then interpolates the S-parameters, using the MATLAB `interp1` function. For example, the curve in the following diagram illustrates the result of interpolating the S_{11} parameters at five different frequencies.



For more information, see “One-Dimensional Interpolation” and the `interp1` reference page in the MATLAB documentation.

As shown in the preceding diagram, the `analyze` method uses the parameter values at f_{min}, the minimum input frequency, for all frequencies smaller than f_{min}. It uses the parameters values at f_{max}, the maximum input frequency, for all frequencies greater than f_{max}. In both cases, the results may not be accurate, so you need to specify network parameter values over a range of frequencies that is wide enough to account for the amplifier behavior.

References

[1] EIA/IBIS Open Forum, *Touchstone File Format Specification*, Rev. 1.1, 2002

See Also

`rfckt.amplifier` | `rfckt.datafile` | `rfckt.passive` | `rfdata.mixerspur` | `rfdata.network` | `rfdata.nf` | `rfdata.noise` | `rfdata.power`

Introduced before R2006a

rfckt.passive

Passive component or network

Description

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

Use the `read` method to read the passive object data from a Touchstone data file. When you read S-parameter data into an `rfckt.passive` object, the magnitude of your S_{21} data must be less than or equal to 1.

Due to random numerical error, data measured from a passive device is not necessarily passive. However, `rfckt.passive` objects can only contain passive data. To import data with active regions, use the `rfckt.amplifier` object, even if the original data represents a passive device.

Creation

Syntax

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

Description

`h = rfckt.passive` returns an passive-device object whose properties all have their default values.

`h = rfckt.passive('Property1',value1,'Property2',value2,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`AnalyzedResult` — Computed S-parameters, noise figure, OIP3, and group delay values

`rfdata.data` object

Computed S-parameters, noise figure, OIP3, and group delay values, specified as rfdata.data object. `Analyzed Result` is a read-only property. For more information refer, “Algorithms” on page 6-91.

Data Types: `function_handle`

`IntpType` — Interpolation method used in `rfckt.passive`

`1-by-N` character array

Interpolation method used in `rfckt.passive`, specified as a `1-by-N` character array of the following values:

Method	Description
`Linear` (default)	Linear interpolation
`Spline`	Cubic spline interpolation
`Cubic`	Piecewise cubic Hermite interpolation

Data Types: `char`

`Name` — Object name

`1-by-N` character array

Object name, specified as an `1-by-N` character array. `Name` is a read-only property.

Data Types: `char`

`NetworkData` — Network parameter data

`rfdata.network` object

Network parameter data, specified as a `rfdata.network` object.

Data Types: `function_handle`

`nport` — Number of ports

positive integer

Number of ports, specified as a positive integer. `nportt` is a read-only property. The default value is `2`.

Data Types: `double`

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
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

Passive RF Components

Create passive RF components using `rfckt.passive`.

```
pas = rfckt.passive('IntpType','cubic')
```

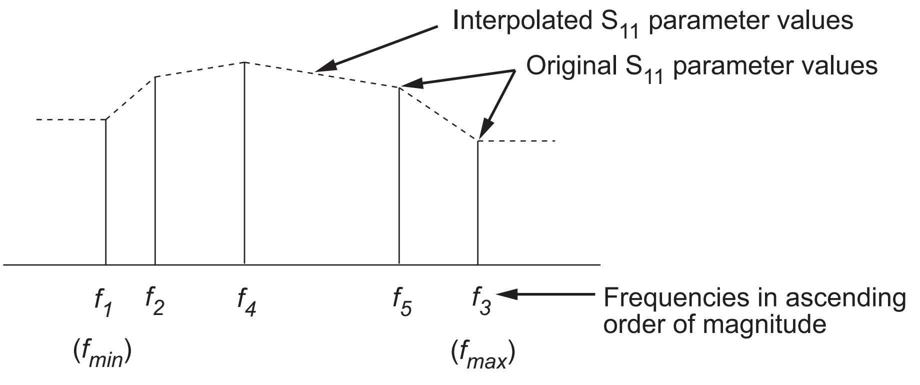
```
pas = 
   rfckt.passive with properties:

          IntpType: 'Cubic'
       NetworkData: [1x1 rfdata.network]
             nPort: 2
    AnalyzedResult: [1x1 rfdata.data]
              Name: 'Passive'
```

Algorithms

The `analyze` method computes the `AnalyzedResult` property as follows:

The `analyze` method uses the data stored in the `'NetworkData'` property of the `rfckt.passive` object to calculate the S-parameter values of the passive component at the frequencies specified in `freq`. If the `'NetworkData'` property contains network Y- or Z-parameters, the `analyze` method first converts the parameters to S-parameters. Using the interpolation method you specify with the `'IntpType'` property, the `analyze` method interpolates the S-parameter values to determine their values at the specified frequencies.

Specifically, the `analyze` method orders the S-parameters according to the ascending order of their frequencies, f_n. It then interpolates the S-parameters, using the MATLAB `interp1` function. For example, the curve in the following diagram illustrates the result of interpolating the S_{11} parameters at five different frequencies.



For more information, see “One-Dimensional Interpolation” and the `interp1` reference page in the MATLAB documentation.

As shown in the preceding diagram, the `analyze` method uses the parameter values at f_{min}, the minimum input frequency, for all frequencies smaller than f_{min}. It uses the parameters values at f_{max}, the maximum input frequency, for all frequencies greater than f_{max}. In both cases, the results may not be accurate, so you need to specify network parameter values over a range of frequencies that is wide enough to account for the component behavior.

The `analyze` method uses the S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

References

[1] EIA/IBIS Open Forum, *Touchstone File Format Specification*, Rev. 1.1, 2002

See Also

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

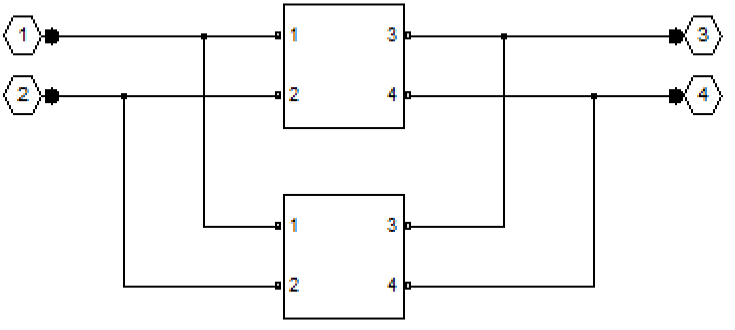
Introduced in R2009a

rfckt.parallel

Parallel connected network

Description

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



Creation

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,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`AnalyzedResult` — Computed S-parameters, noise figure, OIP3, and group delay values

`rfdata.data` object

Computed S-parameters, noise figure, OIP3, and group delay values, specified as rfdata.data object. `Analyzed Result` is a read-only property. For more information refer, "Algorithms" on page 6-95.

Data Types: `function_handle`

`Ckts` — Circuit objects in network

cell array of object handles

Circuit objects in network, specified as a cell array of object handles. All circuits must be 2-port. By default, this property is empty.

Data Types: `char`

`Name` — Object name

`1-by-N` character array

Object name, specified as an `1-by-N` character array. `Name` is a read-only property.

Data Types: `char`

`nport` — Number of ports

positive integer

Number of ports, specified as a positive integer. `nportt` is a read-only property. The default value is `2`.

Data Types: `double`

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
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

Network of RF Objects In Parallel

Create a network of transmission lines connected in parallel using `rfckt.parallel`.

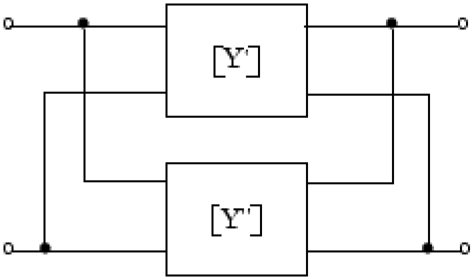
```
tx1 = rfckt.txline;
tx2 = rfckt.txline;
rfplel = rfckt.parallel('Ckts',{tx1,tx2})
```

```
rfplel = 
   rfckt.parallel with properties:

              Ckts: {[1x1 rfckt.txline]  [1x1 rfckt.txline]}
             nPort: 2
    AnalyzedResult: []
              Name: 'Parallel Connected Network'
```

Algorithms

The `analyze` method computes the S-parameters of the `AnalyzedResult` property using the data stored in the `Ckts` property as follows:

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



where

$$[Y'] = \begin{bmatrix} Y_{11}' & Y_{12}' \\ Y_{21}' & Y_{22}' \end{bmatrix}$$

$$[Y''] = \begin{bmatrix} Y_{11}'' & Y_{12}'' \\ Y_{21}'' & Y_{22}'' \end{bmatrix}$$

2 The `analyze` method then calculates the admittance matrix for the parallel network by calculating the sum of the individual admittances. The following equation illustrates the calculations for two 2-port circuits.

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

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

References

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

See Also

rfckt.cascade | rfckt.hybrid | rfckt.hybridg | rfckt.parallelplate | rfckt.series

Introduced before R2006a

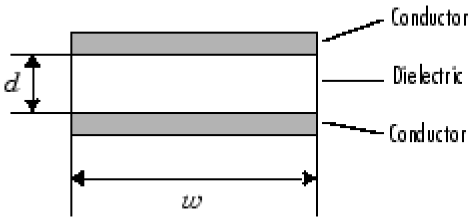
rfckt.parallelplate

Parallel-plate transmission line

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*.



Creation

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,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`AnalyzedResult` — Computed S-parameters, noise figure, OIP3, and group delay values
`rfdata.data` object

Computed S-parameters, noise figure, OIP3, and group delay values, specified as rfdata.data object. `Analyzed Result` is a read-only property. For more information refer, “Algorithms” on page 6-102.

Data Types: `function_handle`

`EpsilonR` — Relative permittivity of dielectric
scalar

Relative permittivity of dielectric, specified as a scalar. The relative permittivity is the ratio of permittivity of the dielectric, ε, to the permittivity in free space, ε_0. The default value is `2.3`.

Data Types: `double`

`LineLength` — Physical length of parallel-plate transmission line
scalar

Physical length of parallel-plate transmission line, specified as a scalar in meters. The default value is `0.01`.

Data Types: `double`

`LossTangent` — Tangent of loss angle of dielectric
scalar

Tangent of loss angle of dielectric, specified as a scalar. The default value is `0`.

Data Types: `double`

`MUR` — Relative permeability of dielectric
scalar

Relative permeability of dielectric, specified as a scalar. The ratio of permeability of dielectric, μ, to the permeability in free space, μ_0. The default value is `1`.

Data Types: `double`

Name — Object name
`1-by-N` character array

Object name, specified as an `1-by-N` character array. `Name` is a read-only property.

Data Types: `char`

nport — Number of ports
positive integer

Number of ports, specified as a positive integer. `nportt` is a read-only property. The default value is `2`.

Data Types: `double`

Separation — Thickness of dielectric
scalar

Thickness of the dielectric separating the plates, specified as a scalar in meters. The default value is `1.0e-3`.

Data Types: `double`

StubMode — Type of stub
`'NotaStub'` (default) | `'Series'` | `'Shunt'`

Type of stub, specified as one of the following values: `'NotaStub'`, `'Series'`, `'Shunt'`.

Data Types: `double`

Termination — Stub transmission line termination
`'NotApplicable'` (default) | `'Open'` | `'Short'`

Stub transmission line termination, specified as one of the following values: `'NotaStub'`, `'Series'`, `'Shunt'`.

Data Types: `double`

Width — Physical width of parallel-plate transmission line
scalar

Physical width of parallel-plate transmission line, specified as a scalar in meters. The default value is `6.0e-4`.

Data Types: `double`

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
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

Parallel Plate Transmission Line

Create a parallel plate transmission line using `rfckt.parallelplate`.

```
tx1=rfckt.parallelplate('LineLength',0.045)
```

```
tx1 =
   rfckt.parallelplate with properties:

             Width: 0.0050
        Separation: 1.0000e-03
               MuR: 1
          EpsilonR: 2.3000
       LossTangent: 0
         SigmaCond: Inf
        LineLength: 0.0450
          StubMode: 'NotAStub'
       Termination: 'NotApplicable'
             nPort: 2
    AnalyzedResult: []
              Name: 'Parallel-Plate Transmission Line'
```

Algorithms

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 stub less line, the `analyze` method first calculates the ABCD-parameters at each frequency contained in the modeling frequencies vector. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

  The `analyze` method calculates the ABCD-parameters using the physical length of the transmission line, *d*, and the complex propagation constant, *k*, using the following equations:

$$A = \frac{e^{kd} + e^{-kd}}{2}$$
$$B = \frac{Z_0 * \left(e^{kd} - e^{-kd}\right)}{2}$$
$$C = \frac{e^{kd} - e^{-kd}}{2 * Z_0}$$
$$D = \frac{e^{kd} + e^{-kd}}{2}$$

  Z_0 and k are vectors whose elements correspond to the elements of f, the vector of frequencies specified in the `analyze` input argument `freq`. Both can be expressed in terms of the resistance (R), inductance (L), conductance (G), and capacitance (C) per unit length (meters) as follows:

$$Z_0 = \sqrt{\frac{R + j2\pi fL}{G + j2\pi fC}}$$
$$k = k_r + jk_i = \sqrt{(R + j2\pi fL)(G + j2\pi FC)}$$

  where

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

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

$$G = \omega\varepsilon'' \frac{w}{d}$$

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

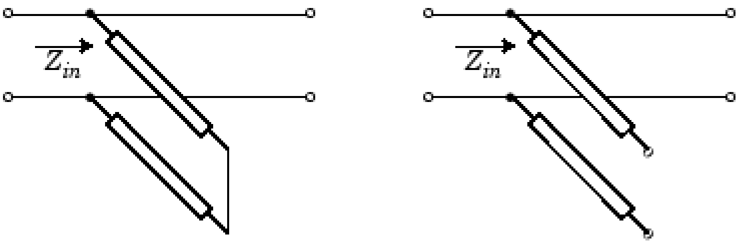
In these equations:

- w is the plate width.
- d is the plate separation.
- σ_{cond} is the conductivity in the conductor.
- μ is the permeability of the dielectric.
- ε is the permittivity of the dielectric.
- ε'' is the imaginary part of ε, $\varepsilon'' = \varepsilon_0\varepsilon_r \tan\delta$, where:
  - ε_0 is the permittivity of free space.
  - ε_r is the `EpsilonR` property value.
  - $\tan\delta$ is the `LossTangent` property value.
- δ_{cond} is the skin depth of the conductor, which the block calculates as

  $1/\sqrt{\pi f \mu\sigma_{cond}}$.
- f is a vector of modeling frequencies determined by the Outport block.

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

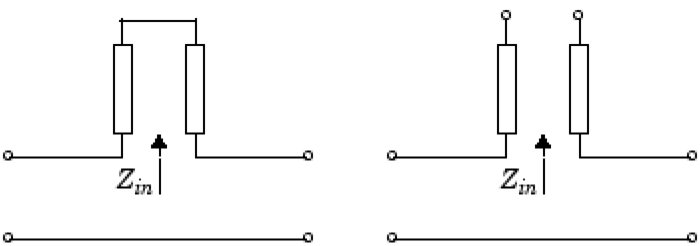
  When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown in the following figure.



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

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

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



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

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

References

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

See Also

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

Introduced in R2009a

rfckt.rlcgline

Passive component or network

Description

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

Creation

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,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`AnalyzedResult` — Computed S-parameters, noise figure, OIP3, and group delay values
`rfdata.data` object

Computed S-parameters, noise figure, OIP3, and group delay values, specified as rfdata.data object. `Analyzed Result` is a read-only property. For more information refer, “Algorithms” on page 6-110.

Data Types: `function_handle`

`C` — Capacitance values per length
vector

Capacitance values per length, specified as a vector in farads per meter. The capacitance values correspond to the frequency values in `'Freq'` property. All values must be positive. The default value is `0`.

Data Types: `double`

`Freq` — Frequency data
M-element vector

Frequency data for the RLCG values, specified as a *M*-element vector. The values must be positive and correspond to the order of the RLCG values. The default value is `1e9`.

Data Types: `double`

`G` — Conductance values per length
vector

Conductance values per length, specified as a vector in Siemens per meter. The conductance values correspond to the frequency values in `'Freq'` property. All values must be positive. The default value is `0`.

Data Types: `double`

`IntpType` — Interpolation method used in `rfckt.rlcgline`
`'Linear'` (default) | `'Spline'` | `'Cubic'`

Interpolation method used in `rfckt.rlcgline`, specified as one of the following values:

Method	Description
`Linear`	Linear interpolation
`Spline`	Cubic spline interpolation
`Cubic`	Piecewise cubic Hermite interpolation

Data Types: `char`

`L` — Inductance values per length
vector

Inductance values per length, specified as vector in henries per meter. The inductance values correspond to the frequency values in `'Freq'` property. All values must be positive. The default value is `0`.

Data Types: `double`

`LineLength` — Physical length of transmission line
scalar

Physical length of transmission line, specified as a scalar in meters. The default value is `0.01`.

Data Types: `double`

`Name` — Object name
`'RLCG Transmission Line'` (default) | `1-by-N` character array

Object name, specified as a `1-by-N` character array. `Name` is a read-only property.

Data Types: `char`

`nport` — Number of ports
positive integer

Number of ports, specified as a positive integer. `nport`t is a read-only property. The default value is `2`.

Data Types: `double`

`StubMode` — Type of stub
`'NotaStub'` (default) | `'Series'` | `'Shunt'`

Type of stub, specified as one of the following values: `'NotaStub'`, `'Series'`, `'Shunt'`.

Data Types: `double`

`Termination` — Stub transmission line termination
`'NotApplicable'` (default) | `'Open'` | `'Short'`

Stub transmission line termination, specified as one of the following values: `'NotaStub'`, `'Series'`, `'Shunt'`.

Data Types: `double`

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
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

RLCG Transmission Line

Create an RLCG transmission line using `rfckt.rlcgline`.

```
rlcgtx=rfckt.rlcgline('R',0.002,'C',8.8542e-12,'L',1.2566e-6,'G',0.002')
```

```
rlcgtx =
   rfckt.rlcgline with properties:

              Freq: 1.0000e+09
                 R: 0.0020
                 L: 1.2566e-06
                 C: 8.8542e-12
                 G: 0.0020
          IntpType: 'Linear'
        LineLength: 0.0100
          StubMode: 'NotAStub'
       Termination: 'NotApplicable'
             nPort: 2
    AnalyzedResult: []
              Name: 'RLCG Transmission Line'
```

Algorithms

The `analyze` method treats the transmission line, which can be lossy or lossless, as a 2-port linear network. It uses the interpolation method you specify in the `IntpType` property to find the R, L, C, and G values at the frequencies you specify when you call `analyze`. Then, it calculates the characteristic impedance, Z0, phase velocity, PV, and loss using these interpolated values. It computes the `AnalyzedResult` property of a stub or as a stubless line using the data stored in the `rfckt.rlcgline` object properties as follows:

- If you model the transmission line as a stubless line, the `analyze` method first calculates the ABCD-parameters at each frequency contained in the modeling frequencies vector. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

  The `analyze` method calculates the ABCD-parameters using the physical length of the transmission line, d, and the complex propagation constant, k, using the following equations:

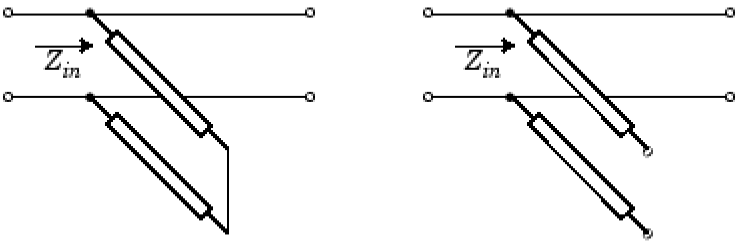
$$A = \frac{e^{kd} + e^{-kd}}{2}$$
$$B = \frac{Z_0 * \left(e^{kd} - e^{-kd}\right)}{2}$$
$$C = \frac{e^{kd} - e^{-kd}}{2 * Z_0}$$
$$D = \frac{e^{kd} + e^{-kd}}{2}$$

  Z_0 and k are vectors whose elements correspond to the elements of f, the vector of frequencies specified in the `analyze` input argument `freq`. Both can be expressed in terms of the resistance (R), inductance (L), conductance (G), and capacitance (C) per unit length (meters) as follows:

$$Z_0 = \sqrt{\frac{R + j2\pi fL}{G + j2\pi fC}}$$
$$k = k_r + jk_i = \sqrt{(R + j2\pi fL)(G + j2\pi FC)}$$

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

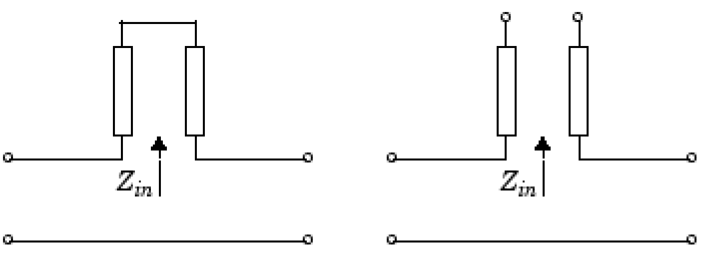
  When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown in the following figure.

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

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

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

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

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

The `analyze` method uses the S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

References

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

See Also

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

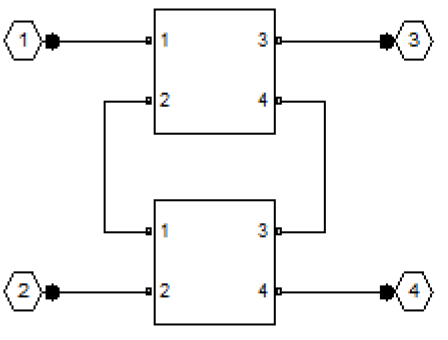
Introduced in R2009a

rfckt.series

Series connected network

Description

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



Creation

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,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`AnalyzedResult` — Computed S-parameters, noise figure, OIP3, and group delay values
`rfdata.data` object

Computed S-parameters, noise figure, OIP3, and group delay values, specified as rfdata.data object. `Analyzed Result` is a read-only property. For more information refer, “Algorithms” on page 6-115.

Data Types: `function_handle`

`Ckts` — Circuit objects in network
cell array of object handles

Circuit objects in network, specified as a cell array of object handles. All circuits must be 2-port. By default, this property is empty.

Data Types: `char`

`Name` — Object name
`'RLCG Transmission Line'` (default) | `1-by-N` character array

Object name, specified as a `1-by-N` character array. `Name` is a read-only property.

Data Types: `char`

`nport` — Number of ports
positive integer

Number of ports, specified as a positive integer. `nportt` is a read-only property. The default value is `2`.

Data Types: `double`

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
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

Series Connected RF Network Object

Create a series connected RF network object using `rfckt.series`

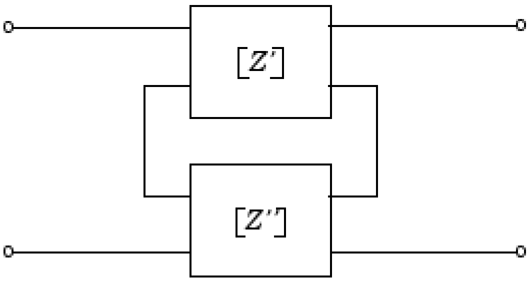
```
tx1 = rfckt.txline;
tx2 = rfckt.txline;
ser = rfckt.series('Ckts',{tx1,tx2})
```

```
ser = 
   rfckt.series with properties:

              Ckts: {[1x1 rfckt.txline]  [1x1 rfckt.txline]}
             nPort: 2
    AnalyzedResult: []
              Name: 'Series Connected Network'
```

Algorithms

The `analyze` method computes the S-parameters of the `AnalyzedResult` property using the data stored in the `Ckts` property as follows:

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



where

$$[Z'] = \begin{bmatrix} Z_{11}{}' & Z_{12}{}' \\ Z_{21}{}' & Z_{22}{}' \end{bmatrix}$$

$$[Z''] = \begin{bmatrix} Z_{11}{}'' & Z_{12}{}'' \\ Z_{21}{}'' & Z_{22}{}'' \end{bmatrix}$$

2 The `analyze` method then calculates the impedance matrix for the series network by calculating the sum of the individual impedances. The following equation illustrates the calculations for two 2-port circuits.

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

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

References

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

See Also

`rfckt.cascade` | `rfckt.hybrid` | `rfckt.hybridg` | `rfckt.parallel`

Introduced in R2009a

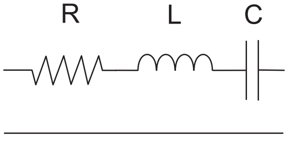
rfckt.seriesrlc

Series RLC component

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.



Creation

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)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`AnalyzedResult` — Computed S-parameters, noise figure, OIP3, and group delay values
`rfdata.data` object

Computed S-parameters, noise figure, OIP3, and group delay values, specified as rfdata.data object. `Analyzed Result` is a read-only property. For more information refer, "Algorithms" on page 6-122.

Data Types: `function_handle`

`R` — Resistance value
positive scalar

Resistance value, specified as a positive scalar in ohms. The default value is `0`.

Data Types: `double`

`C` — Capacitance value
positive scalar

Capacitance value, specified as a positive scalar in farads. The default value is `'Inf'`.

Data Types: `double`

`L` — Inductance value
positive scalar

Inductance value, specified as a positive scalar in henries. The default value is `0`.

Data Types: `double`

`Name` — Object name
`'RLCG Transmission Line'` (default) | `1-by-N` character array

Object name, specified as a `1-by-N` character array. `Name` is a read-only property.

Data Types: `char`

`nport` — Number of ports
positive integer

Number of ports, specified as a positive integer. `nportt` is a read-only property. The default value is `2`.

Data Types: `double`

Object Functions

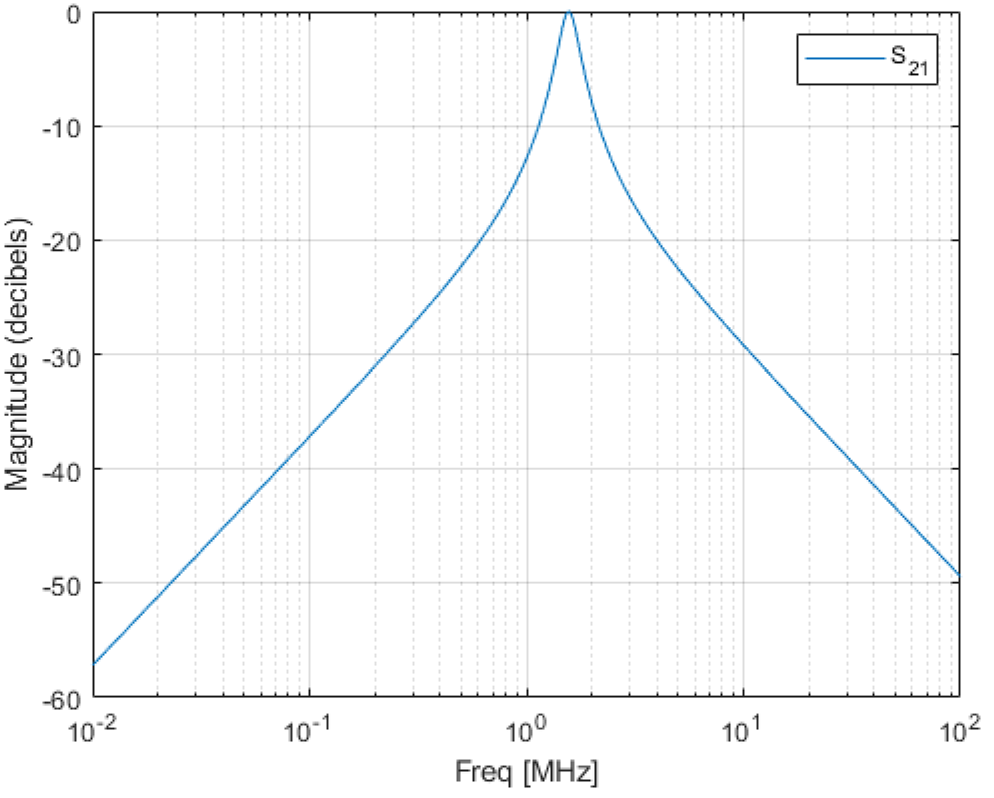
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters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

Frequency Response of an LC Resonator

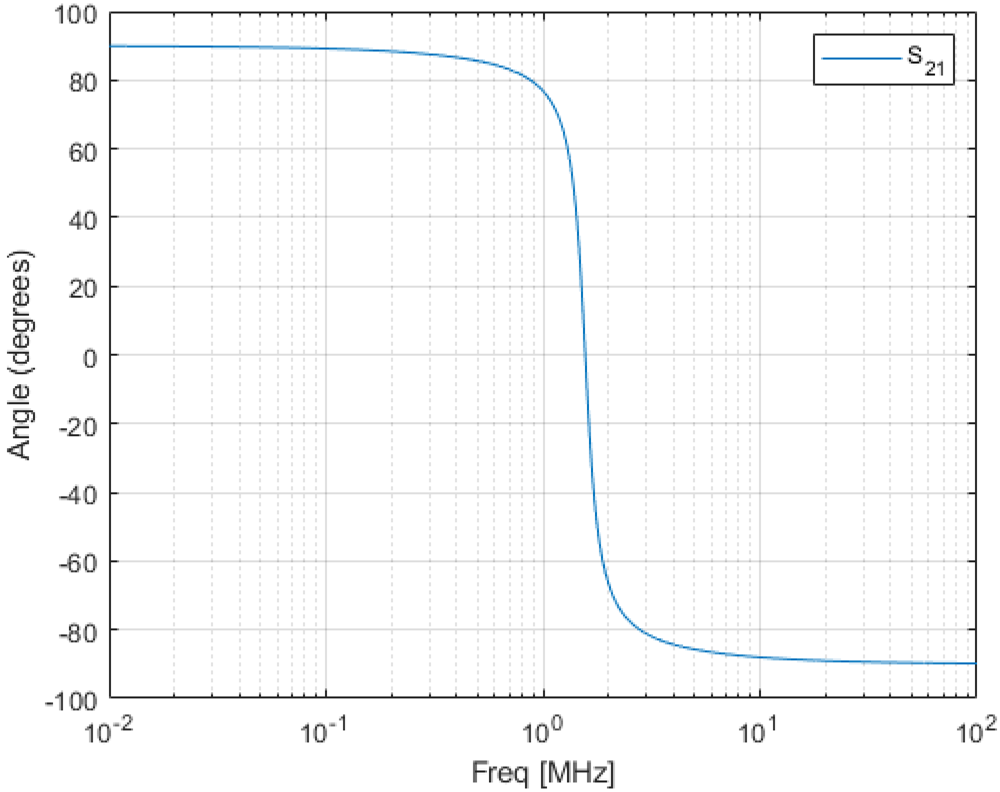
This example creates a series LC resonator and examines its frequency response. It first creates the circuit object and then uses the analyze method to calculate its frequency response. Finally, it plots the results - first, the magnitude in decibels (dB):

```
h = rfckt.seriesrlc('L',4.7e-5,'C',2.2e-10);
analyze(h,logspace(4,8,1000));
plot(h,'s21','dB')
ax = gca;
ax.XScale = 'log';
```



The example then plots the phase, in degrees:

```
figure
plot(h,'s21','angle')
ax = gca;
ax.XScale = 'log';
```



Algorithms

The `analyze` method computes the S-parameters of the `AnalyzedResult` property using the data stored in the `rfckt.seriesrlc` object properties by first calculating the ABCD-parameters for the circuit, and then converting the ABCD-parameters to S-parameters using the `abcd2s` function. For this circuit, A = 1, B = Z, C = 0, and D = 1, where

$$Z = \frac{-LC\omega^2 + jRC\omega + 1}{jC\omega}$$

and $\omega = 2\pi f$.

The `analyze` method uses the S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

References

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

See Also

`rfckt.shuntrlc`

Introduced in R2009a

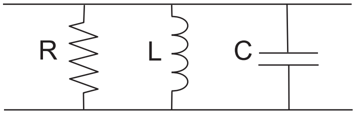
rfckt.shuntrlc

Shunt RLC component

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.



Creation

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)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

AnalyzedResult — Computed S-parameters, noise figure, OIP3, and group delay values
rfdata.data object

Computed S-parameters, noise figure, OIP3, and group delay values, specified as rfdata.data object. Analyzed Result is a read-only property. For more information refer, “Algorithms” on page 6-128.

Data Types: function_handle

R — Resistance value
positive scalar

Resistance value, specified as a positive scalar in ohms. The default value is 0.

Data Types: double

C — Capacitance value
positive scalar

Capacitance value, specified as a positive scalar in farads. The default value is 'Inf'.

Data Types: double

L — Inductance value
positive scalar

Inductance value, specified as a positive scalar in henries. The default value is 0.

Data Types: double

Name — Object name
'RLCG Transmission Line' (default) | 1-by-N character array

Object name, specified as a 1-by-N character array. Name is a read-only property.

Data Types: char

nport — Number of ports
positive integer

Number of ports, specified as a positive integer. nportt is a read-only property. The default value is 2.

Data Types: `double`

Object Functions

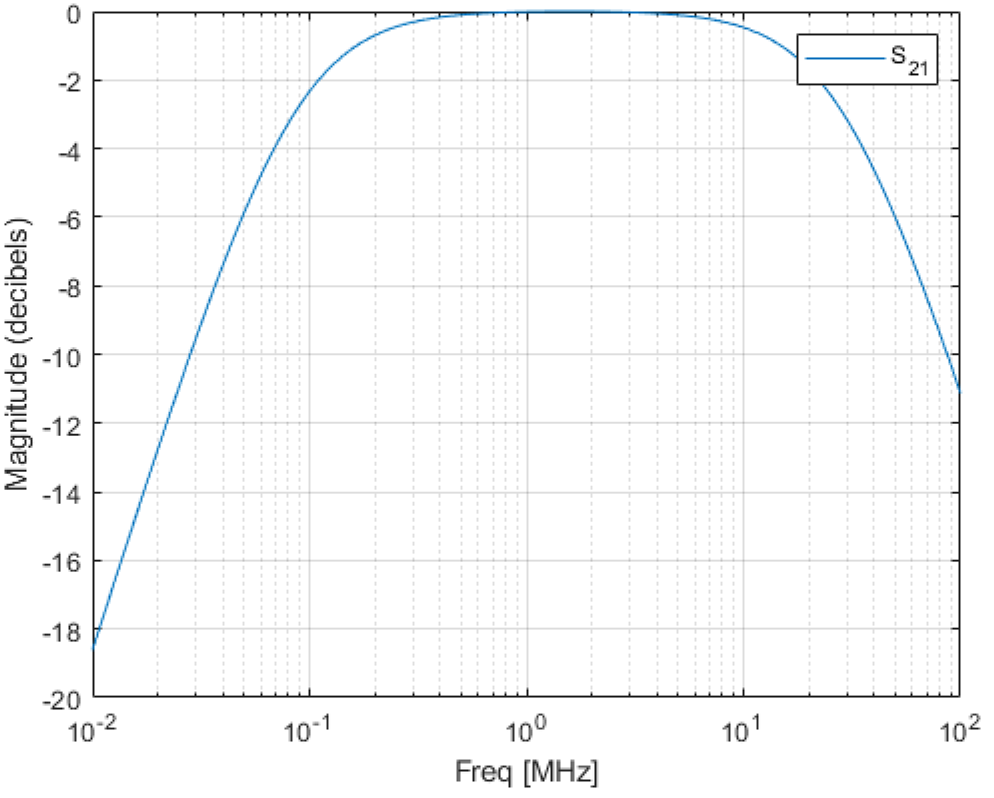
analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
extract	Extract array of network parameters from data object
listformat	List valid formats for specified circuit object parameter
listparam	List valid parameters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

Frequency Response of a Shunt LC Resonator

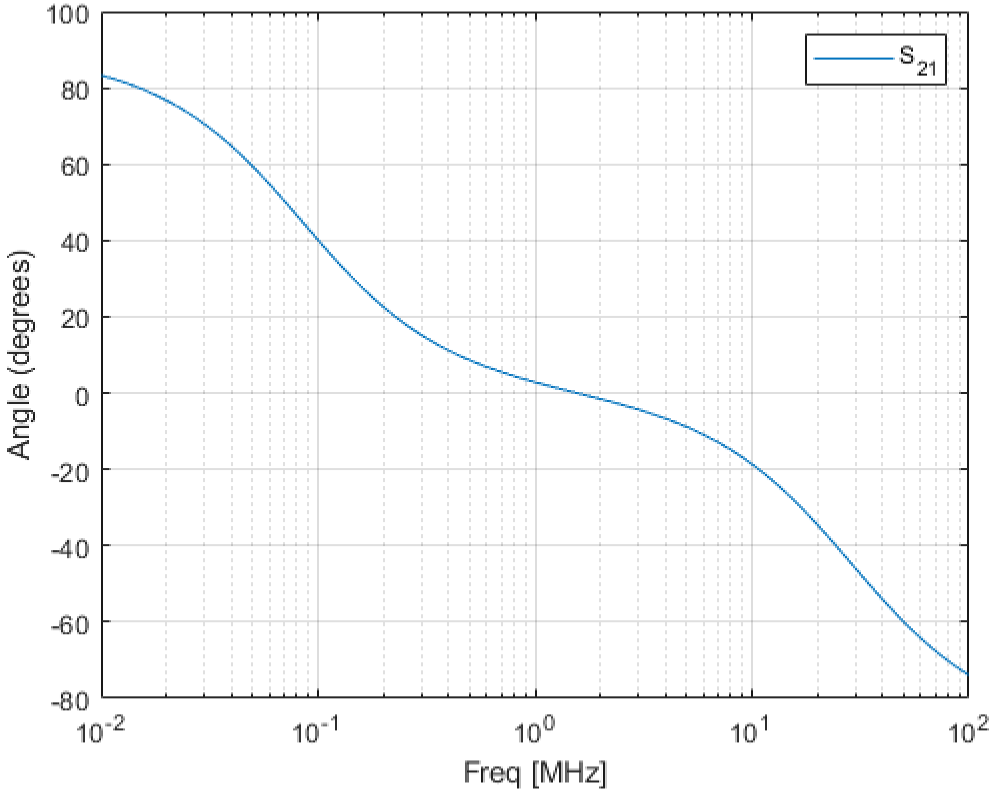
This example creates a shunt LC resonator and examines its frequency response. It first creates the circuit object and then uses the analyze method to calculate its frequency response. The plot is in decibels(dB).

```
h = rfckt.shuntrlc('L',4.7e-5,'C',2.2e-10);
analyze(h,logspace(4,8,1000));
plot(h,'s21','dB')
ax = gca;
ax.XScale = 'log';
```



The example then plots the phase, in degrees:

```
figure
plot(h,'s21','angle')
ax = gca;
ax.XScale = 'log';
```



Algorithms

The `analyze` method computes the S-parameters of the `AnalyzedResult` property using the data stored in the `rfckt.shuntrlc` object properties by first calculating the ABCD-parameters for the circuit, and then converting the ABCD-parameters to S-parameters using the `abcd2s` function. For this circuit, A = 1, B = 0, C = Y, and D = 1, where

$$Y = \frac{-LC\omega^2 + j(L\,/\,R)\omega + 1}{jL\omega}$$

and $\omega = 2\pi f$.

The `analyze` method uses the S-parameters to calculate the group delay values at the frequencies specified in the `analyze` input argument `freq`, as described in the `analyze` reference page.

References

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

See Also

`rfckt.seriesrlc`

Introduced in R2009a

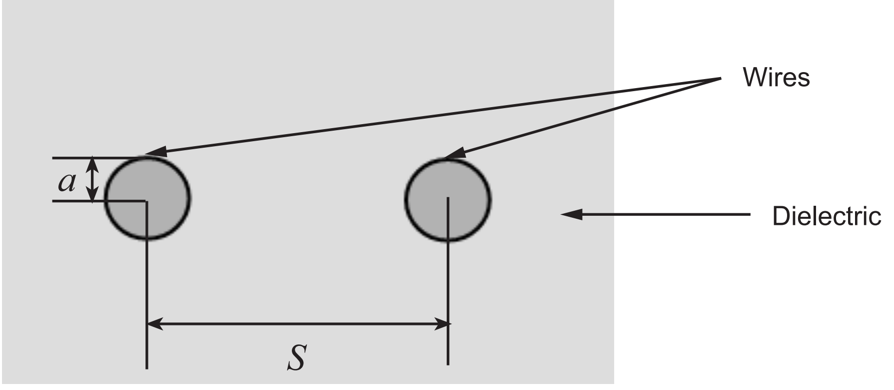
rfckt.twowire

Two-wire transmission line

Description

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

A two-wire transmission line is shown in cross-section in the following figure. Its physical characteristics include the radius of the wires *a*, the separation or physical distance between the wire centers *S*, and the relative permittivity and permeability of the wires. RF Toolbox software assumes the relative permittivity and permeability are uniform.



Creation

Syntax

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

Description

`h = rfckt.twowire` 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.twowire('Property1',value1,'Property2',value2,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`AnalyzedResult` — Computed S-parameters, noise figure, OIP3, and group delay values
`rfdata.data` object

Computed S-parameters, noise figure, OIP3, and group delay values, specified as a rfdata.data object. This is a read-only property. For more information refer, “Algorithms” on page 6-134.

Data Types: `function_handle`

`EpsilonR` — Relative permittivity of dielectric
scalar

Relative permittivity of dielectric, specified as a scalar. The relative permittivity is the ratio of permittivity of the dielectric, ε , to the permittivity in free space, ε_0 . The default value is `2.3`.

Data Types: `double`

`LineLength` — Physical length of transmission line
scalar

Physical length of transmission line, specified as a scalar in meters. The default value is `0.01`.

Data Types: `double`

`LossTangent` — Tangent of loss angle of dielectric
scalar

Tangent of loss angle of dielectric, specified as a scalar. The default value is `0`.

Data Types: `double`

`MUR` — Relative permeability of dielectric
scalar

Relative permeability of dielectric, specified as a scalar. The ratio of permeability of dielectric, μ, to the permeability in free space, μ_0. The default value is `1`.

Data Types: `double`

`Name` — Object name
`'Two-Wire Transmission Line'` (default) | `1-by-N` character array

Object name, specified as a `1-by-N` character array. This is a read-only property.

Data Types: `char`

`nport` — Number of ports
positive integer

Number of ports, specified as a positive integer. This is a read-only property. The default value is `2`.

Data Types: `double`

`Radius` — Conducting wire radius
scalar

Conducting wire radius, specified as a scalar in meters. The default value is `6.7e-4`.

Data Types: `double`

`SigmaCond` — Conductor conductivity
scalar in Siemens per meter

Conductor conductivity, specified as a scalar in Siemens per meter (S/m). The default value is `Inf`.

Data Types: `double`

`StubMode` — Type of stub
`'NotaStub'` (default) | `'Series'` | `'Shunt'`

Type of stub, specified as a one of the following values: `'NotaStub'`, `'Series'`, `'Shunt'`.

Data Types: `double`

`Termination` — Stub transmission line termination
`'NotApplicable'` (default) | `'Open'` | `'Short'`

Stub transmission line termination, specified as one of the following values: `'NotaStub'`, `'Series'`, `'Shunt'`.

Data Types: `double`

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
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

Two-Wire Transmission Line

Create a two-wire transmission line object using `rfckt.twowire`.

```
tx1=rfckt.twowire('Radius',7.5e-4)
```

```
tx1 = 
   rfckt.twowire with properties:
```

```
         Radius: 7.5000e-04
     Separation: 0.0016
            MuR: 1
       EpsilonR: 2.3000
    LossTangent: 0
      SigmaCond: Inf
     LineLength: 0.0100
       StubMode: 'NotAStub'
    Termination: 'NotApplicable'
          nPort: 2
 AnalyzedResult: []
           Name: 'Two-Wire Transmission Line'
```

Algorithms

- If you model the transmission line as a stubless line, the `analyze` method first calculates the ABCD-parameters at each frequency contained in the modeling frequencies vector. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

  The `analyze` method calculates the ABCD-parameters using the physical length of the transmission line, *d*, and the complex propagation constant, *k*, using the following equations:

  $$A = \frac{e^{kd} + e^{-kd}}{2}$$
  $$B = \frac{Z_0 * \left(e^{kd} - e^{-kd}\right)}{2}$$
  $$C = \frac{e^{kd} - e^{-kd}}{2 * Z_0}$$
  $$D = \frac{e^{kd} + e^{-kd}}{2}$$

  Z_0 and k are vectors whose elements correspond to the elements of f, the vector of frequencies specified in the `analyze` input argument `freq`. Both can be expressed in terms of the resistance (R), inductance (L), conductance (G), and capacitance (C) per unit length (meters) as follows:

$$Z_0 = \sqrt{\frac{R + j2\pi fL}{G + j2\pi fC}}$$

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

where

$$R = \frac{1}{\pi a \sigma_{cond} \delta_{cond}}$$

$$L = \frac{\mu}{\pi} \mathrm{a}\cosh\left(\frac{D}{2a}\right)$$

$$G = \frac{\pi \omega \varepsilon''}{\mathrm{a}\cosh\left(\frac{D}{2a}\right)}$$

$$C = \frac{\pi \varepsilon}{\mathrm{a}\cosh\left(\frac{D}{2a}\right)}$$

In these equations:

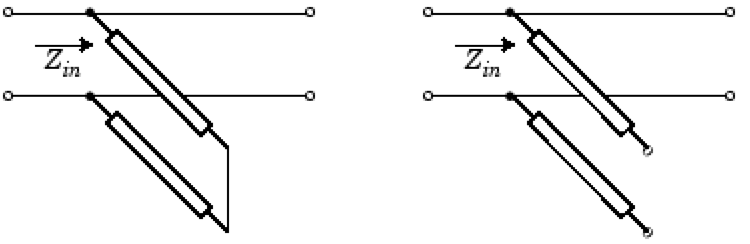
- w is the plate width.
- d is the plate separation.
- σ_{cond} is the conductivity in the conductor.
- μ is the permeability of the dielectric.
- ε is the permittivity of the dielectric.
- ε'' is the imaginary part of ε, $\varepsilon'' = \varepsilon_0 \varepsilon_r \tan \delta$, where:
  - ε_0 is the permittivity of free space.
  - ε_r is the `EpsilonR` property value.
  - $\tan \delta$ is the `LossTangent` property value.
- δ_{cond} is the skin depth of the conductor, which the block calculates as

  $1 / \sqrt{\pi f \mu \sigma_{cond}}$.

  - *f* is a vector of modeling frequencies determined by the Outport block.

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

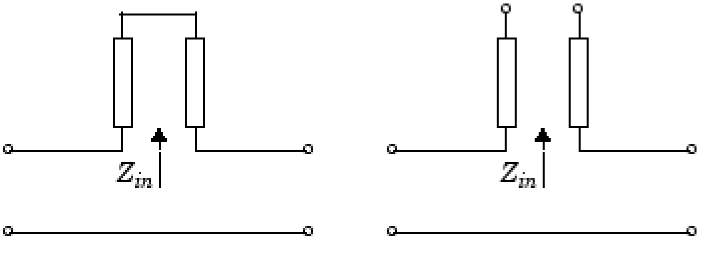
  When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown in the following figure.



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

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

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



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

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

References

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

See Also

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

Introduced in R2009a

rfckt.txline

General transmission line

Description

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

Creation

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,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`AnalyzedResult` — Computed S-parameters, noise figure, OIP3, and group delay values
`rfdata.data` object

Computed S-parameters, noise figure, OIP3, and group delay values, specified as rfdata.data object. This is a read-only property. For more information refer, “Algorithms” on page 6-142.

Data Types: `function_handle`

`Freq` — Frequency data
M-element vector

Frequency data for the RLCG values, specified as a *M*-element vector in Hz. The values must be positive and correspond to the order of loss and phase velocity values. By default, this property is empty.

Data Types: `double`

`IntpType` — Interpolation method used in `rfckt.rlcgline`
`'Linear'` (default) | `'Spline'` | `'Cubic'`

Interpolation method used in `rfckt.rlcgline`, specified as one of the following values:

Method	Description
`Linear`	Linear interpolation
`Spline`	Cubic spline interpolation
`Cubic`	Piecewise cubic Hermite interpolation

Data Types: `char`

`LineLength` — Physical length of transmission line
scalar

Physical length of transmission line, specified as a scalar in meters. The default value is `0.01`.

Data Types: `double`

`Loss` — Reduction in strength of signal
`0` (default) | nonnegative *M*-element vector

Reduction in strength of signal as it travels through the transmission line, specified as a nonnegative *M*-element vector in decibels per meter.

Data Types: `double`

`Name` — Object name
`'Transmission Line'` (default) | `1-by-N` character array

Object name, specified as a `1-by-N` character array. This is a read-only property.

Data Types: `char`

`nport` — Number of ports
positive integer

Number of ports, specified as a positive integer. This is a read-only property. The default value is `2`.

Data Types: `double`

`PV` — Phase velocity
M-element vector

Phase velocity or propagation velocity of a uniform plane wave on the transmission line specified as a *M*-element vector in meters/sec. The phase velocity values correspond to the frequency values. The default value is `299792458`.

Data Types: `double`

`StubMode` — Type of stub
`'NotaStub'` (default) | `'Series'` | `'Shunt'`

Type of stub, specified as one of the following values: `'NotaStub'`, `'Series'`, `'Shunt'`.

Data Types: `double`

`Termination` — Stub transmission line termination
`'NotApplicable'` (default) | `'Open'` | `'Short'`

Stub transmission line termination, specified as one of the following values: `'NotaStub'`, `'Series'`, `'Shunt'`.

Data Types: `double`

`Z0` — Characteristic impedance
vector in ohms

Characteristic impedance, specified as a vector in ohms. The default value is `50` ohms.

Data Types: `double`

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
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

Frequency Domain Analysis of a Transmission Line

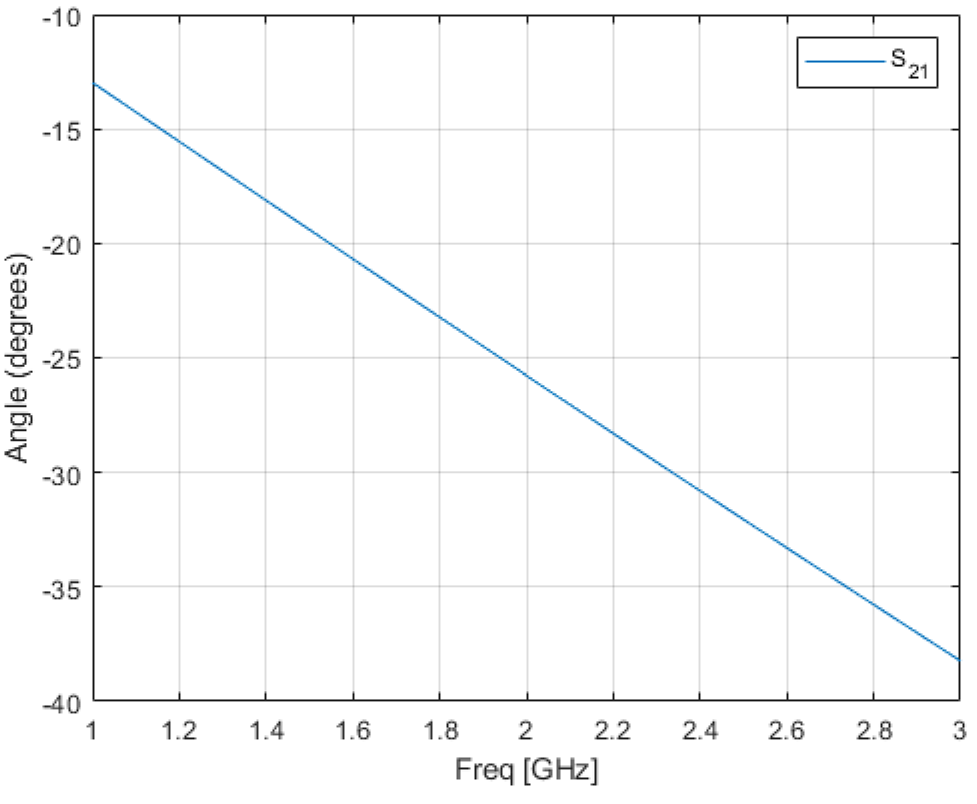
Transmission Line Properties

```
trl = rfckt.txline('Z0',75)
```

```
trl = 
   rfckt.txline with properties:

        LineLength: 0.0100
          StubMode: 'NotAStub'
       Termination: 'NotApplicable'
              Freq: 1.0000e+09
                Z0: 75
                PV: 299792458
              Loss: 0
          IntpType: 'Linear'
             nPort: 2
    AnalyzedResult: []
              Name: 'Transmission Line'
```

Plot

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



Algorithms

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 stub less line, the `analyze` method first calculates the ABCD-parameters at each frequency contained in the modeling frequencies vector. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

  The `analyze` method calculates the ABCD-parameters using the physical length of the transmission line, *d*, and the complex propagation constant, *k*, using the following equations:

  $$A = \frac{e^{kd} + e^{-kd}}{2}$$
  $$B = \frac{Z_0 * \left(e^{kd} - e^{-kd}\right)}{2}$$
  $$C = \frac{e^{kd} - e^{-kd}}{2 * Z_0}$$
  $$D = \frac{e^{kd} + e^{-kd}}{2}$$

  Z_0 is the specified characteristic impedance. *k* is a vector whose elements correspond to the elements of the input vector `freq`. The `analyze` method calculates *k* from the specified properties as $k = \alpha_a + i\beta$, where α_a is the attenuation coefficient and β is the wave number. The attenuation coefficient α_a is related to the specified loss, α, by

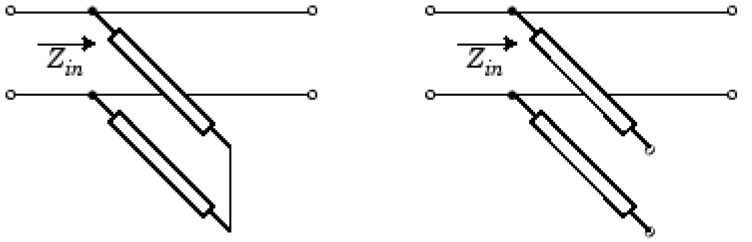
  $$\alpha_a = -\ln\left(10^{\alpha/20}\right)$$

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

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

  where *f* is the frequency range specified in the `analyze` input argument `freq`. The phase velocity V_p is derived from the `rfckt.txline` object properties. It is also known as the *wave propagation velocity*.
- If you model the transmission line as a shunt or series stub, the `analyze` method first calculates the ABCD-parameters at the specified frequencies. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

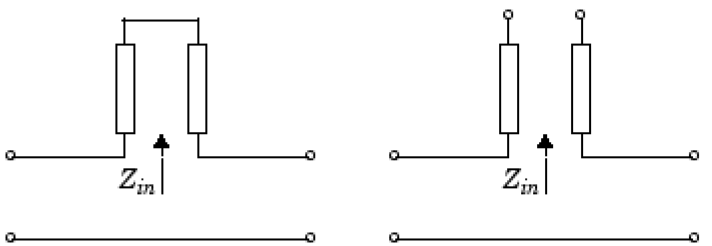
When you set the `StubMode` property to `'Shunt'`, the 2-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown in the following figure.



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

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

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



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

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

References

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

See Also

`rfckt.coaxial` | `rfckt.cpw` | `rfckt.microstrip` | `rfckt.rlcgline` | `rfckt.twowire`

Introduced in R2009a

rfdata.data

Store result of circuit object analysis

Description

Use the `data` class to store S-parameters, noise figure in decibels, and frequency-dependent, third-order output (OIP3) intercept points.

There are three ways to create an `rfdata.data` object:

- You can construct it by specifying its properties from workspace data using the `rfdata.data` constructor.
- You can create it from file data using the `read` method.
- You can perform frequency domain analysis of a circuit object using the `analyze` method, and RF Toolbox software stores the results in an `rfdata.data` object.

Creation

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,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

Freq — Frequency data for S-parameters
M-element vector

Frequency data for the S-parameters in the S-Parameters property, specified as a M-element vector in hertz. The values must be positive and correspond to the order of the S-parameters. By default, this property is empty.

Data Types: double

GroupDelayData — Group delay data
M-element vector

Group delay data calculated at each frequency, specified as a M-element vector in seconds. By default, this property is empty.

Data Types: double

IntpType — Interpolation method used in rfdata.data
1-by-N character array

Interpolation method used in rfdata.data, specified as a 1-by-N character array of the following values:

Method	Description
Linear (default)	Linear interpolation
Spline	Cubic spline interpolation
Cubic	Piecewise cubic Hermite interpolation

Data Types: char

Name — Object name
1-by-N character array

Object name, specified as a 1-by-N character array. This is a read-only property.

Data Types: char

NF — Noise figure
scalar

Noise figure, specified as a scalar in dB. 'NF' is the amount of noise relative to noise temperature of 290 degrees kelvin. The default value is zero indicating a noiseless system.

Data Types: function_handle

OIP' — Output third-order intercept
scalar

Output third-order intercept, specified as a scalar 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 value is Inf.

Data Types: double

S_Parameters — S-parameter data
2-by-2-by-M array

S-parameter data, specified as a 2-by-2-by-M array. M is the number of frequencies at which the network parameters are specified. By default, this property is empty.

Data Types: double

Z0 — Reference impedance
scalar

Reference impedance, specified as a scalar in ohms. The default value is 50 ohms.

Data Types: double

ZL — Load impedance
scalar

Load impedance, specified as a scalar in ohms. The default value is 50 ohms.

Data Types: double

ZS — Source impedance
scalar

Source impedance, specified as a scalar in ohms. The default value is 50 ohms.

Data Types: double

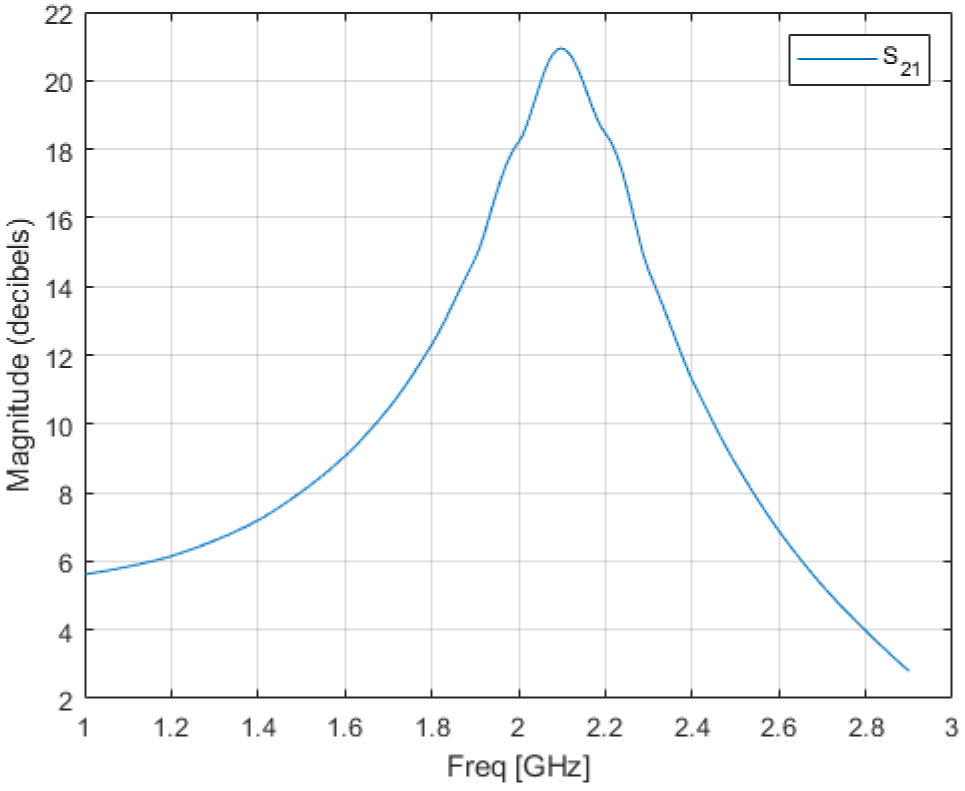
Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
extract	Extract array of network parameters from data object
getop	Display operating conditions
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
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

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.
```



- “RF Data Objects”

See Also

`rfdata.ip3` | `rfdata.mixerspur` | `rfdata.network` | `rfdata.nf` | `rfdata.noise` | `rfdata.power`

Topics

“RF Data Objects”

Introduced in R2009a

rfdata.ip3

Store frequency-dependent, third-order intercept points

Description

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

Note If you set `NonLinearData` using `rfdata.ip3` or `rfdata.power`, then the property is converted from scalar OIP3 format to the format of `rfdata.ip3` or `rfdata.power`.

Creation

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)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

Data — Third-order intercept values
M-element vector

Third-order intercept values, specified as a M-element vector in watts. The values correspond to the frequencies stored in the `'Freq'` property. The default value is `'Inf'`.

Data Types: `double`

`Freq` — Frequency data
M-element vector

Frequency data , specified as a M-element vector in hertz. The values must be positive and correspond to the order of the IP3 values. By default, this property is empty.

Data Types: `double`

`Name` — Object name
`1-by-N` character array

Object name, specified as a `1-by-N` character array. This is a read-only property.

Data Types: `char`

`Type` — IP3 data type
`'OIP3'` (default) | `'IIP3'`

IP3 data type, specified as a `'OIP3'` or `'IIP3'`.

Data Types: `double`

Examples

Store Third-Order Intercept Point Specifications

Create an object to store third-order intercept point specifications using `rfdata.ip3`.

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

```
ip3data =
   rfdata.ip3 with properties:

    Type: 'OIP3'
    Freq: 2.1000e+09
    Data: 8.4500
    Name: '3rd order intercept'
```

See Also

`rfdata.data` | `rfdata.mixerspur` | `rfdata.network` | `rfdata.nf` | `rfdata.noise` | `rfdata.power`

Introduced in R2009a

rfdata.mixerspur

Store data from intermodulation table

Description

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

Creation

Syntax

```
h = rfdata.mixerspur
h = rfdata.mixerspur('Data',value1,'PLORef',value2,'PinRef','value3)
```

Description

`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)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`Data` — Mixer spur power values
matrix

Mixer spur power values, specified as a matrix in decibels. The values are such that the mixer spur power is less than the power at the fundamental output frequency. Values must be between `0` and `99`. By default, this property is empty.

Data Types: `double`

Name — Object name
`1-by-N` character array

Object name, specified as a `1-by-N` character array. This property is a read-only.

Data Types: `char`

`PinRef` — Reference input power
scalar

Reference input power, specified as a scalar in decibels relative to 1 milliwatt. The default value is `0`.

Data Types: `double`

`PLORef` — Reference local oscillator power
scalar

Reference local oscillator power, specified as a scalar in decibels relative to 1 milliwatt. The default value is `0`.

Data Types: `double`

Examples

Store Mixer Spur Power Specifications

Create an object to store mixer spur power specifications using `rfdata.mixerspur`.

```
spurs = rfdata.mixerspur('Data',[2 5 3; 1 0 99; 10 99 99],...
          'PinRef',3,'PLORef',5)
```

```
spurs =
   rfdata.mixerspur with properties:

    PLORef: 5
    PinRef: 3
      Data: [3x3 double]
      Name: 'Intermodulation table'
```

See Also

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

Introduced in R2009a

rfdata.network

Store frequency-dependent network parameters

Description

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

Creation

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)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`Data` — Network parameter data
`2-by-2-by-`*M* array

Network parameter data, specified as a `2-by-2-by-M` array. *M* is the number of frequencies. The values correspond to the frequencies stored in the `'Freq'` property. By default, this property is empty.

Data Types: `double`

`Freq` — Frequency data

M-element vector

Frequency data , specified as a M-element vector in hertz. The values must be positive and correspond to the order of the IP3 values. By default, this property is empty.

Data Types: `double`

`Name` — Object name

`1-by-N` character array

Object name, specified as a `1-by-N` character array. This is a read-only property.

Data Types: `char`

`Type` — Type of network parameters

`'S'` | `'Y'` | `'Z'` | `'ABCD'` | `'H'` | `'G'` | `'T'`

Type of network parameters, specified as one of the following network parameters:

- `'S'`
- `'Y'`
- `'Z'`
- `'ABCD'`
- `'H'`
- `'G'`
- `'T'`

Data Types: `double`

`Z0` — Reference impedance

scalar

Reference impedance, specified as a scalar in ohms. This property is only available when the `'Type'` is set to `'S'`. The default value is `50` ohms.

Data Types: `double`

Examples

Store Frequency-Dependant RF Network Parameters.

Create an object to store frequency-dependant Y-parameters using `rfdata.network`.

```
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)
```

```
net =
   rfdata.network with properties:

    Type: 'Y_PARAMETERS'
    Freq: [3x1 double]
    Data: [2x2x3 double]
      Z0: 50.0000 + 0.0000i
    Name: 'Network parameters'
```

See Also

`rfdata.data` | `rfdata.ip3` | `rfdata.mixerspur` | `rfdata.nf` | `rfdata.noise` | `rfdata.power`

Introduced in R2009a

rfdata.nf

Store frequency-dependent noise figure data for amplifiers or mixers

Description

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

Creation

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)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

Data — Noise figure values
M-element vector

Noise figure values, specified as a *M*-element vector in dB. The values correspond to the frequencies stored in the `'Freq'` property. The default value is `0`.

Data Types: `double`

Freq — Frequency data
M-element vector

Frequency data , specified as a M-element vector in hertz. The values must be positive and correspond to the order of the noise figure values. By default, this property is empty.

Data Types: `double`

Name — Object name
`1-by-N` character array

Object name, specified as a `1-by-N` character array. This is a read-only property.

Data Types: `char`

Examples

Store Noise Figure Specifications of RF Circuit Object.

Create an object to store noise figure specifications using `rfdata.nf`.

```
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`

Introduced in R2009a

rfdata.noise

Store frequency-dependent spot noise data for amplifiers or mixers

Description

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

Creation

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)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`FMIN` — Minimum noise figure data
M-element vector

Noise figure values, specified as a *M*-element vector in dB. . The values correspond to the frequencies stored in the `'Freq'` property. By default, the value is `1`.

Data Types: `double`

Freq — Frequency data
M-element vector

Frequency data , specified as a M-element vector in hertz. The values must be positive and correspond to the spot noise data in 'FMIN', 'GAMMAOPT', and 'RN' properties. By default, this property is empty.

Data Types: double

GAMMAOPT — Optimum source reflection coefficients
M-element vector

Optimum source reflection coefficients , specified as a M-element vector. The values correspond to the frequencies stored in the 'Freq' property. The default value is 1.

Data Types: double

Name — Object name
1-by-N character array

Object name, specified as a 1-by-N character array. This is a read-only property.

Data Types: char

RN — Equivalent normalized noise resistance data
M-element vector

Equivalent normalized noise resistance data, specified as a M-element vector. The values correspond to the frequencies stored in the 'Freq' property. The default value is 1.

Data Types: double

Examples

Store Spot Noise Specifications of RF Circuit Object.

Create an object to store spot noise specifications using rfdata.noise.

```
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)

noisedata =
   rfdata.noise with properties:

        Freq: [2x1 double]
        Fmin: [2x1 double]
    GammaOPT: [2x1 double]
          RN: [2x1 double]
        Name: 'Spot noise data'
```

See Also

rfdata.data | rfdata.ip3 | rfdata.mixerspur | rfdata.network | rfdata.nf | rfdata.power

Introduced in R2009a

rfdata.power

Store output power and phase information for amplifiers or mixers

Description

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

Creation

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,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`Freq` — Frequency data
M-element vector

Frequency data , specified as a *M*-element vector in hertz. The values must be positive and correspond to the power data in `'Phase'`, `'Pin'`, and `'Pout'` properties. The order of frequencies is equal to the order of the phase and power values. By default, this property is empty.

Data Types: `double`

Name — Object name
`'Power data'` | `1-by-N` character array

Object name, specified as a `1-by-N` character array. This is a read-only property.

Data Types: `char`

Phase — Phase shift data
M-element cell

Phase shift data, specified as a *M*-element cell in degrees. . The values correspond to the frequencies stored in the `'Freq'` property. The values within each element correspond to the input power values stored in the `'Pin'` property. The default value is `1`.

Data Types: `double`

Pin — Input power data
M-element cell in watts

Input power data , specified as a *M*-element vector cell in watts. The values correspond to the frequencies stored in the `'Freq'` property. For example,

$$P_{in} = \{[A];[B];[C]\};$$

where `A`, `B`, and `C` are column vectors that contain the first three frequencies stored in the `'Freq'` property.

The default value is `1`.

Data Types: `double`

Pout — Output power data
M-element vector

Output power data, specified as a *M*-element vector in watts. The values correspond to the frequencies stored in the `'Freq'` property. The values within each element correspond to the input power values stored in the `'Pin'` property. The default value is `1`.

Data Types: `double`

Examples

Store Output Power and Phase Specifications of RF Circuit Object.

Create an object to store output power and phase specifications using `rfdata.power`.

```
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 =
   rfdata.power with properties:

     Freq: []
      Pin: {[1 10]}
     Pout: {[1 10]}
    Phase: {}
     Name: 'Power data'
```

```
powerdata.Freq = f;
powerdata.Phase = phase;
powerdata.Pin = pin;
powerdata.Pout = pout;
```

- `rfdata.data`
- `rfdata.ip3`
- `rfdata.mixerspur`
- `rfdata.network`
- `rfdata.nf`
- `rfdata.noise`

See Also

Topics
`rfdata.data`

```
rfdata.ip3
rfdata.mixerspur
rfdata.network
rfdata.nf
rfdata.noise
```

Introduced in R2009a

rfmodel.rational

Store output power and phase information for amplifiers or mixers

Description

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

$$F(s) = \left(\sum_{k=1}^{n} \frac{C_k}{s - A_k} + D \right) e^{-s\tau}, \quad s = j2\pi f$$

There are two ways to construct an rational function object:

- You can fit a rational function object 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.

Creation

Syntax

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

Description

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

`h = rfmodel.rational('Property1',value1,'Property2',value2,...)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote

Properties

`A` — Poles of rational function object
complex vector

Poles of rational function object, specified as a complex vector in radians/second. The property length is shown in:

$$F(s) = \left(\sum_{k=1}^{n} \frac{C_k}{s - A_k} + D \right) e^{-s\tau}, \quad s = j2\pi f$$

where, n must be equal to the length of the vector you provide for `'C'`. n is the number of poles in the rational function object. By default, this property is empty.

Data Types: `double`

`C` — Residues of rational function object
complex vector

Residues of the rational function object, specified as a complex vector in radians/second. The property length is shown in

$$F(s) = \left(\sum_{k=1}^{n} \frac{C_k}{s - A_k} + D \right) e^{-s\tau}, \quad s = j2\pi f$$

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 object. By default, this property is empty.

Data Types: `double`

`D` — Frequency response offset
scalar

Frequency response offset, specified as a scalar. The default value is `0`.

Data Types: `double`

`Delay` — Frequency response time delay
scalar

Frequency response time delay, specified as a scalar. The default value is `0`.

Data Types: double

Name — Object name
'Rational Function' (default) | 1-by-N character array

Object name, specified as a `1-by-N` character array. This is a read-only property.

Data Types: char

Object Functions

freqresp	Frequency response of rational function object
stepresp	Step-signal response of rational function object
ispassive	Check passivity of N-port S-parameters
timeresp	Time response for rational function object
writeva	Write Verilog-A description of rational function object

Examples

Fit a Rational Function to Data

Fit a rational function to data from an `rfdata.data` object.

```
S = sparameters('defaultbandpass.s2p');
freq = S.Frequencies;
data = rfparam(S,2,1);
fit = rationalfit(freq,data)
```

```
fit =
   rfmodel.rational with properties:

        A: [10x1 double]
        C: [10x1 double]
        D: 0
    Delay: 0
     Name: 'Rational Function'
```

Define, Evaluate and Visualize a Rational Function

Construct a rational function object, `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.

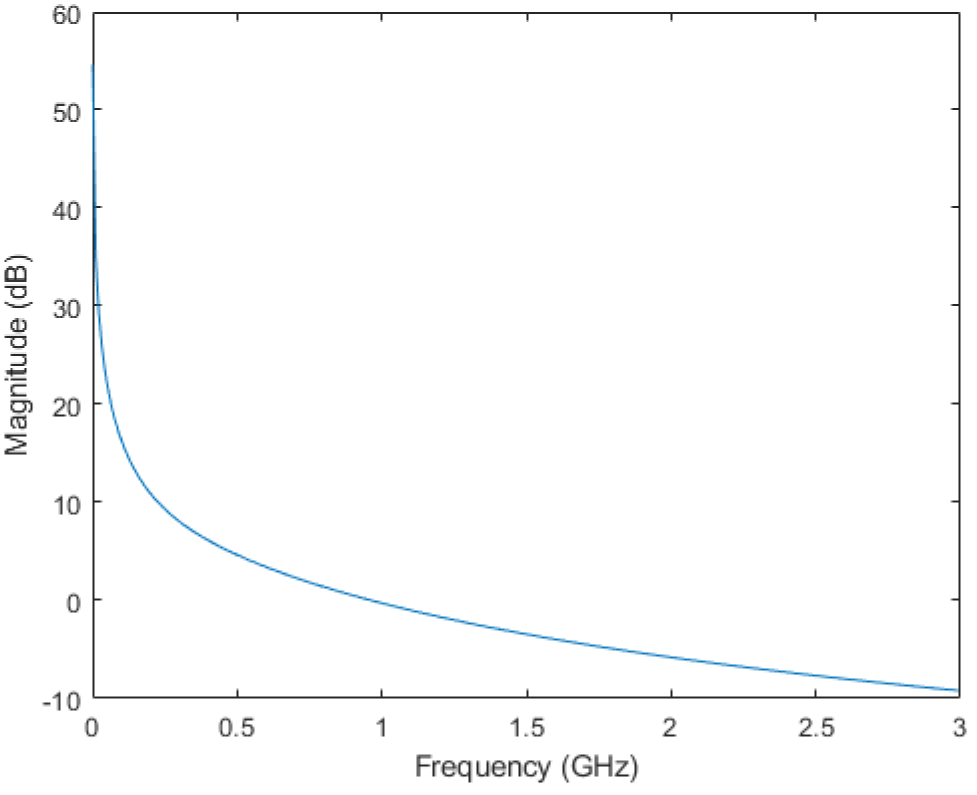
```
rat=rfmodel.rational('A',[-5e9,-3e9,-4e6],'C',[6e8,2e9,4e9]);
```

Perform frequency-domain analysis from 1.0 MHz to 3.0 GHz.

```
f = [1e6:1.0e7:3e9];
```

Plot the resulting frequency response in decibels on the X-Y plane.

```
[resp,freq]=freqresp(rat,f);
figure
plot(freq/1e9,20*log10(abs(resp)));
xlabel('Frequency (GHz)')
ylabel('Magnitude (dB)')
```



See Also

Introduced in R2009a

rfbudget

Create RF budget object and compute RF budget results

Description

Use the `rfbudget` object to create an rfbudget element to calculate RF budget results of a circuit. You can use any 2-port element in this circuit such as `amplifier`, `modulator`, or `nport`. Open the complete `rfbudget` circuit in an **RF Budget Analyzer** app. You can also export the completed circuit to RF Blockset.

Creation

Syntax

```
rfobj = rfbudget
rfobj = rfbudget(elements,inputfreq,inputpwr,bandwidth)
rfobj = rfbudget( ___ ,autoupdate)
rfobj = rfbudget(Name,Value)
```

Description

`rfobj = rfbudget` creates an rfbudget object, `rfobj`, with default empty property values.

`rfobj = rfbudget(elements,inputfreq,inputpwr,bandwidth)` creates an RF budget object from the input RF elements, and independently computes an RF budget analysis at the specified input frequencies, available input power, and signal bandwidth. The input arguments are stored in the `Elements`, `InputFrequency`, `AvailableInputPower`, and `SignalBandwidth` properties. The analysis results are stored in dependent properties. By default, if any of the input properties are changed, the object recomputes results.

`rfobj = rfbudget( ___ ,autoupdate)` sets the `'AUTOUPDATE'` property to `false`. Setting `AutoUpdate` to false turns off automatic budget recomputation as parameters are changed. You can use this syntax with any of the precious syntaxes.

`rfobj = rfbudget(Name,Value)` creates RF budget object with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

`'Elements'` — RF budget elements

RF toolbox object | array of RF toolbox objects

RF budget elements, specified as the comma-separated pair consisting of `'Elements'` and an RF toolbox object or array of RF toolbox objects. The possible elements are `amplifier`, `modulator`, generic `rfelement`, and `nport` objects. To specify a circuit consisting of multiple RF objects, specify the elements as a cell array.

Example: `a = amplifier;m =modulator;rfbudget('Elements',[a m])` calculates the RF budget analysis of the amplifier and modulator circuit.

`'InputFrequency'` — Input frequency of signal

nonnegative scalar or vector in Hz

Input frequency of signal, specified as the comma-separated pair consisting of `'InputFrequency'` and a nonnegative scalar or vector in Hz. If the input frequency is a vector, then the RF budget object calculates the analysis for each input frequency separately.

Example: `'InputFrequency',2e9`

`'AvailableInputPower'` — Power applied at input of cascade

scalar in dBm

Power applied at the input of the cascade, specified as the comma-separated pair consisting of `'AvailableInputPower'` and a scalar in dBm.

Example: `'AvailableInputPower',-30`

'SignalBandwidth' — Signal bandwidth at input of cascade
scalar in Hz

Signal bandwidth at the input of the cascade, specified as the comma-separated pair consisting of `'SignalBandwidth'` and a scalar in Hz.

Example: `'SignalBandwidth',10`

'AutoUpdate' — Automatically recompute rf budget analysis
`true` (default) | `false`

Option to automatically recompute the RF budget analysis by incorporating changes made to the existing circuit, specified as the comma-separated pair consisting of `'AutoUpdate'` and a boolean scalar.

Example: `'AutoUpdate',true`

'OutputFrequency' — Output frequencies
row vector in Hz

This is a read-only property.

Output frequencies, specified as the comma-separated pair consisting of `'OutputFrequency'` and a row vector in Hz.

'OutputPower' — Output power
row vector in dBm

This is a read-only property.

Output power, specified as the comma-separated pair consisting of `'OutputPower'` and a row vector in dBm.

'TransducerGain' — Transducer power gains
row vector in dB

This is a read-only property.

Transducer power gains, specified as the comma-separated pair consisting of `'TransducerGain'` and a row vector in dB.

'NF' — Noise figures
row vector in dB

This is a read-only property.

Noise figures, specified as the comma-separated pair consisting of `'NF'` and a row vector in dB.

`'OIP3'` — Output-referred third-order intercept
row vector in dBm

This is a read-only property.

Output-referred third-order intercept, specified as the comma-separated pair consisting of `'OIP3'` and a row vector in dBm.

`'IIP3'` — Input-referred third-order intercept
row vector in dBm

This is a read-only property.

Input-referred third-order intercept, specified as the comma-separated pair consisting of `'IIP3'` and a row vector in dBm.

`'SNR'` — Signal-to-noise ratio
row vector in dB

This is a read-only property.

Signal-to-noise ratio, specified as the comma-separated pair consisting of `'SNR'` and a row vector in dB.

Object Functions

show	Display RF budget object in RF Budget Analyzer app
computeBudget	Compute results of rfbudget object
exportScript	Export MATLAB code that generates RF budget object
exportRFBlockset	Create RF Blockset model from RF budget object
exportTestbench	Create measurement testbench from RF budget object
rfbudget.rfplot	Plot cumulative RF budget result versus cascade input frequency
smithplot	Plot of measurement data on Smith chart
polar	Plot specified circuit object parameters on polar coordinates

Examples

Default RF Budget

Open a default RF budget object.

```
obj = rfbudget
```

```
obj = 
  rfbudget with properties:

               Elements: []
         InputFrequency: [] Hz
    AvailableInputPower: [] dBm
        SignalBandwidth: [] Hz
             AutoUpdate: true
```

RF Budget Analysis of Series of RF Elements

Create an amplifier with a gain of 4 dB.

```
a = amplifier('Gain',4);
```

Create a modulator with an OIP3 of 13 dBm.

```
m = modulator('OIP3',13);
```

Create an nport using `passive.s2p`.

```
n = nport('passive.s2p');
```

Create an rf element with a gain of 10 dB.

```
r = rfelement('Gain',10);
```

Calculate the rf budget of a series of rf elements at an input frequency of 2.1 GHz, an available input power of -30 dB, and a bandwidth of 10 MHz.

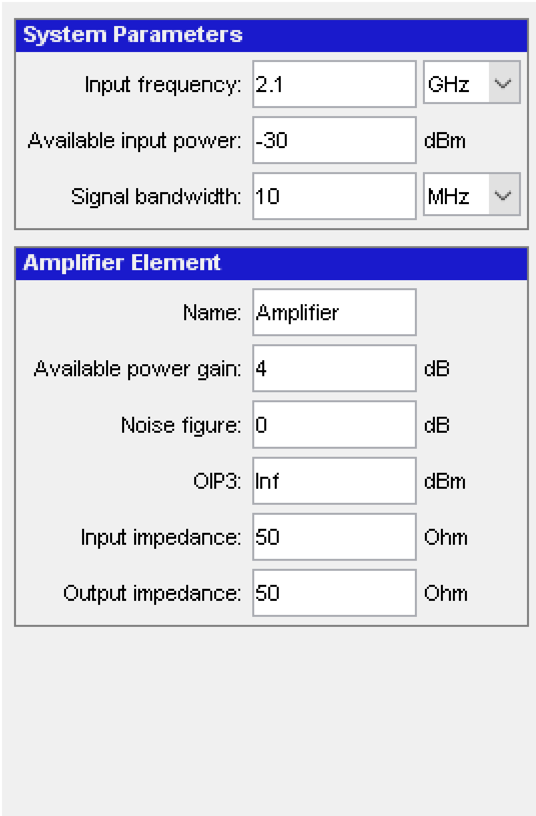
```
b = rfbudget([a m r n],2.1e9,-30,10e6)
```

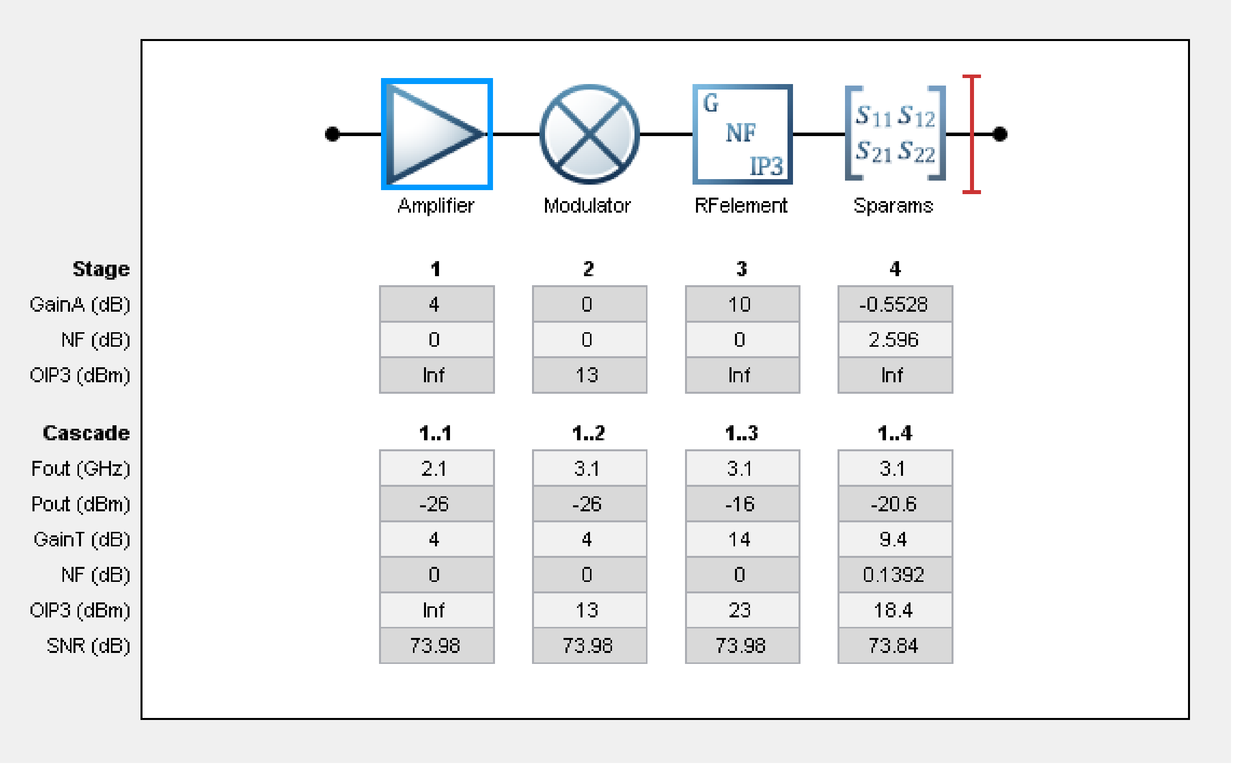
```
b = 
  rfbudget with properties:
```

```
              Elements: [1x4 rf.internal.rfbudget.Element]
        InputFrequency: 2.1 GHz
   AvailableInputPower: -30 dBm
       SignalBandwidth:  10 MHz
            AutoUpdate: true

   Analysis Results
       OutputFrequency: (GHz) [  2.1    3.1    3.1     3.1]
           OutputPower: (dBm) [  -26    -26    -16   -20.6]
        TransducerGain: (dB)  [    4      4     14     9.4]
                    NF: (dB)  [    0      0      0  0.1392]
                  OIP3: (dBm) [  Inf     13     23    18.4]
                  IIP3: (dBm) [  Inf      9      9       9]
                   SNR: (dB)  [73.98  73.98  73.98   73.84]
```

Show the analysis in the RF Budget Analyzer app.

```
show(b)
```


System Parameters
Input frequency: 2.1 GHz
Available input power: -30 dBm
Signal bandwidth: 10 MHz
Amplifier Element
Name: Amplifier
Available power gain: 4 dB
Noise figure: 0 dB
OIP3: Inf dBm
Input impedance: 50 Ohm
Output impedance: 50 Ohm



Plot Cumulative Output Power and Gain of RF System

Create an RF system.

Create an RF bandpassfilter using the Touchstone file `RFBudget_RF`.

```
f1 = nport('RFBudget_RF.s2p','RFBandpassFilter');
```

Create an amplifier with a gain of 11.53 dB, a noise figure (NF)of 1.53 dB, and an output third-order intercept (OIP3) of 35 dBm.

```
a1 = amplifier('Name','RFAmplifier','Gain',11.53,'NF',1.53,'OIP3',35);
```

Create a demodulator with a gain of 6 dB, a NF of 4 dB, and an OIP3 of 50 dBm.

```
d = modulator('Name','Demodulator','Gain',-6,'NF',4,'OIP3',50, ...
        'LO',2.03e9,'ConverterType','Down');
```

Create an IF bandpassfilter using the Touchstone file `RFBudget_IF`.

```
f2 = nport('RFBudget_IF.s2p','IFBandpassFilter');
```

Create an amplifier with a gain of 30 dB, a NF of 8 dB, and an OIP3 of 37 dBm.

```
a2 = amplifier('Name','IFAmplifier','Gain',30,'NF',8,'OIP3',37);
```

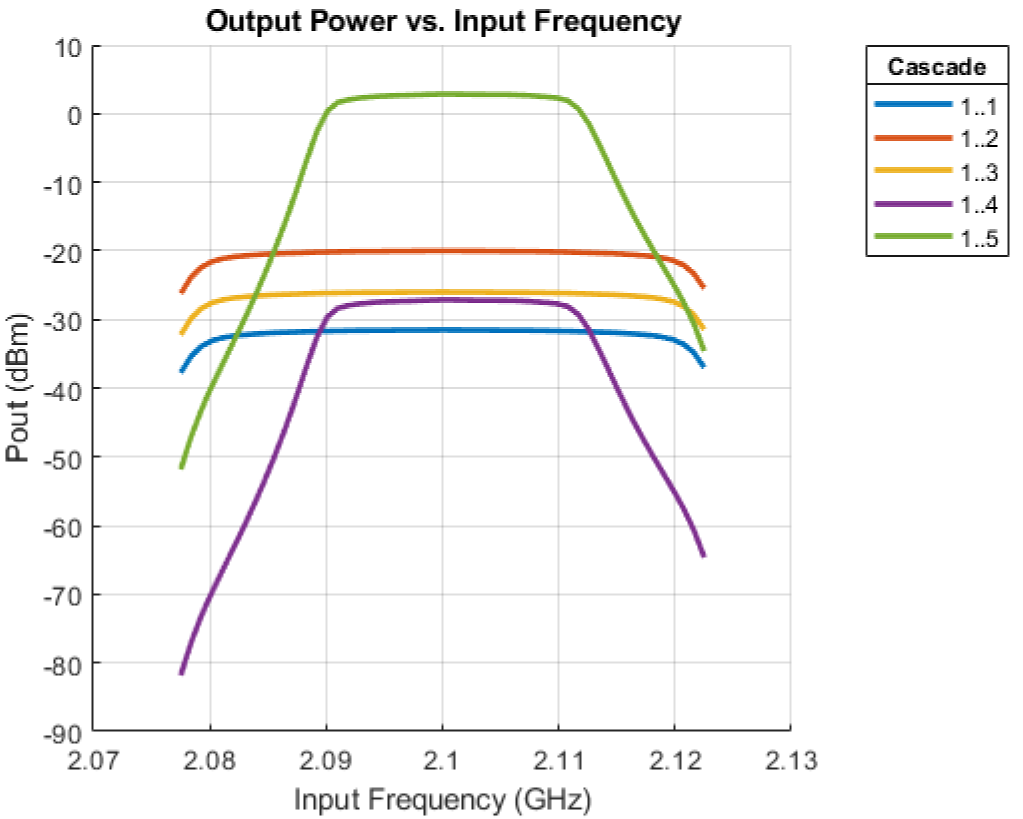
Calculate the RF budget of the system using an input frequency of 2.1 GHz, an input power of -30 dBm, and a bandwidth of 45 MHz.

```
b = rfbudget([f1 a1 d f2 a2],2.1e9,-30,45e6)
```

```
b = 
  rfbudget with properties:

               Elements: [1x5 rf.internal.rfbudget.Element]
         InputFrequency: 2.1 GHz
    AvailableInputPower: -30 dBm
        SignalBandwidth:  45 MHz
             AutoUpdate: true

   Analysis Results
        OutputFrequency: (GHz) [   2.1    2.1   0.07    0.07   0.07]
            OutputPower: (dBm) [-31.53    -20    -26  -27.15  2.847]
         TransducerGain: (dB)  [-1.534  9.996  3.996   2.847  32.85]
                     NF: (dB)  [ 1.533  3.064  3.377   3.611  7.036]
                   OIP3: (dBm) [   Inf     35  28.97   27.82  36.96]
                   IIP3: (dBm) [   Inf     25  24.97   24.97  4.116]
                    SNR: (dB)  [ 65.91  64.38  64.07   63.83  60.41]
```

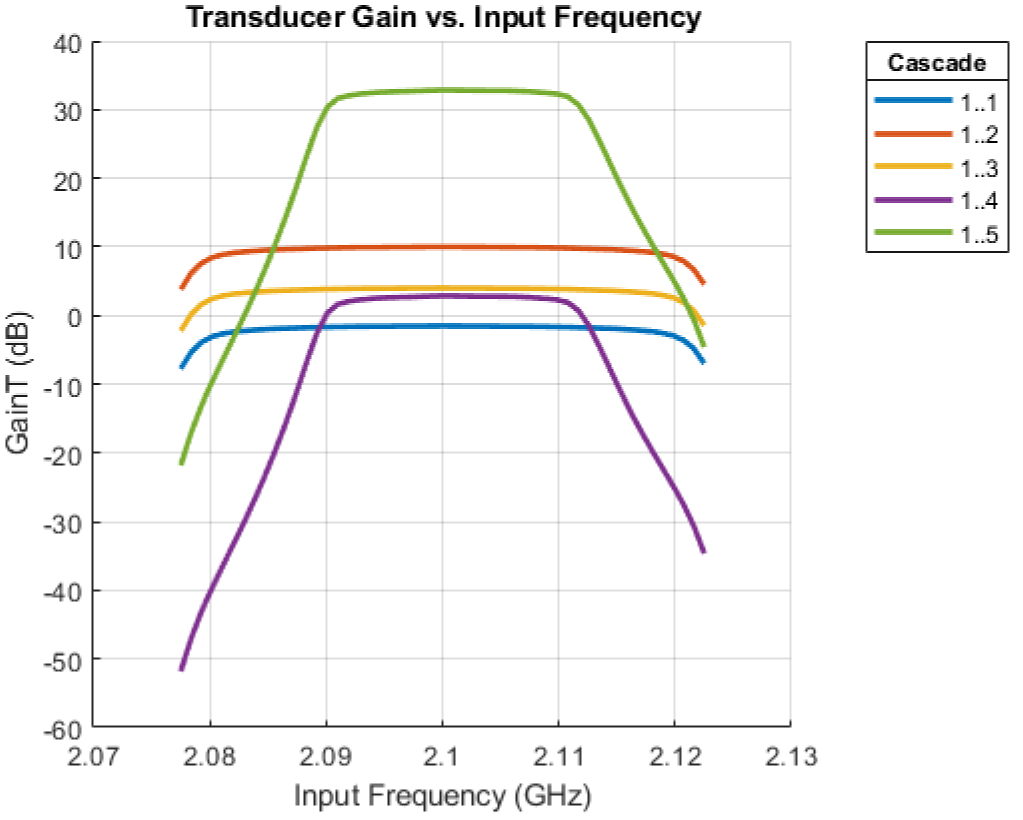
Plot the available output power.

```
rfplot(b,'Pout')
view(90,0)
```



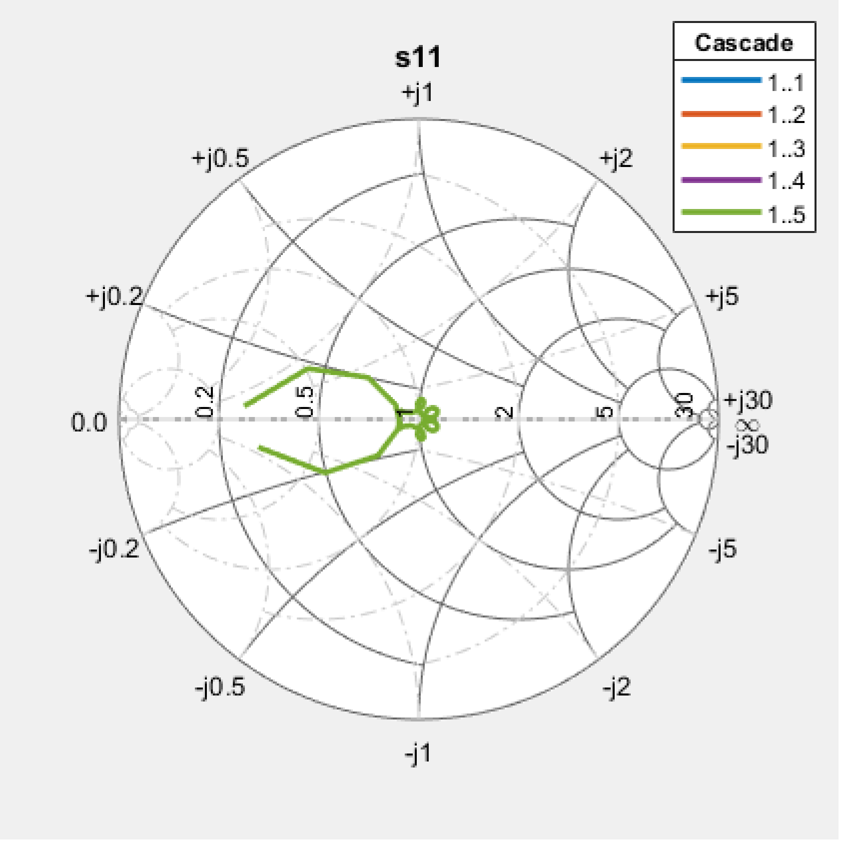
Plot the transducer gain.

```
rfplot(b,'GainT')
view(90,0)
```

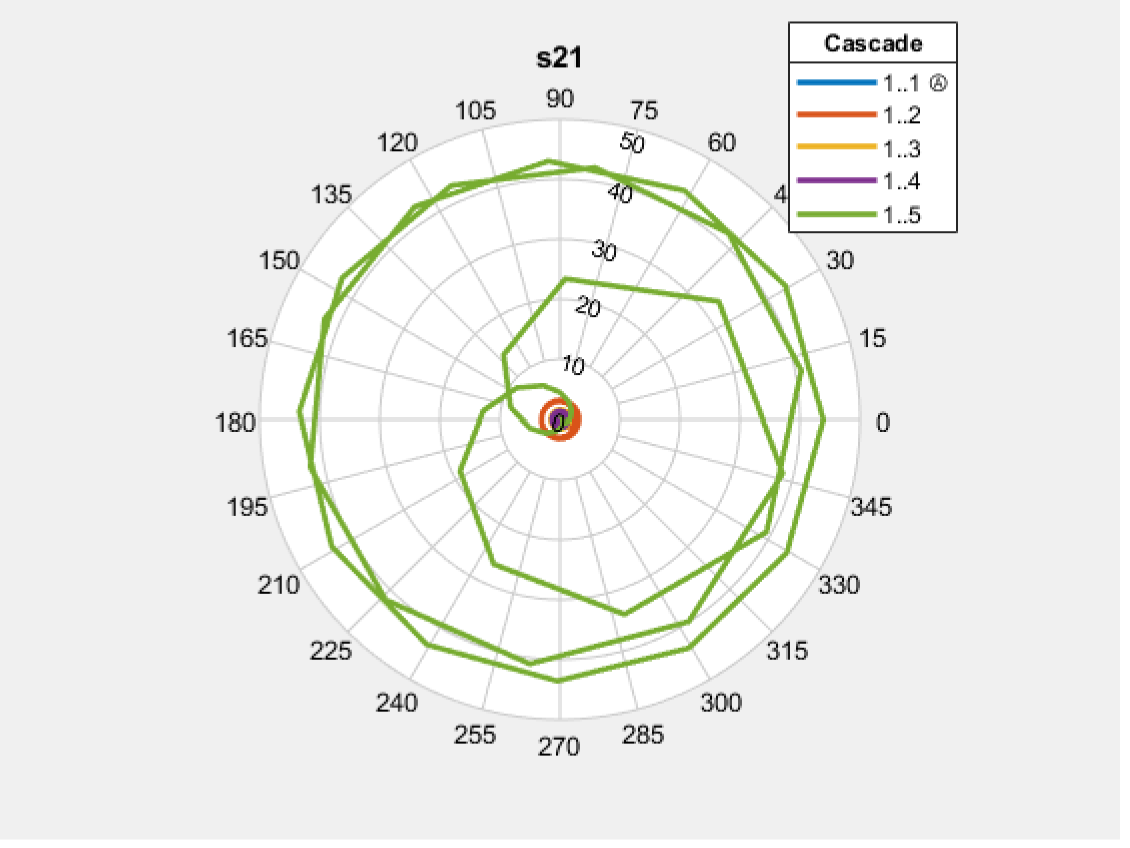


Plot sparameters of RF System on a Smith Chart and a Polar plot

```
s = smithplot(b,1,1,'GridType','ZY');
```


s11
Cascade
1..1
1..2
1..3
1..4
1..5
+j1
+j0.5
+j2
+j0.2
+j5
0.0
0.2
0.5
1
2
5
30
+j30
∞
-j30
-j0.2
-j5
-j0.5
-j2
-j1

```
p = polar(b,2,1);
```



- "Visualizing RF Budget Analysis Over Bandwidth"

See Also

`amplifier` | `modulator` | `nport`

Topics

"Visualizing RF Budget Analysis Over Bandwidth"

Introduced in R2017a

amplifier

Amplifier object

Description

Use the `amplifier` object to create an amplifier element. An amplifier is a 2-port RF circuit object. You can use this element in the `rfbudget` object and the `circuit` object.

Creation

Syntax

```
amp = amplifier
amp = amplifier(Name,Value)
```

Description

`amp = amplifier` creates an amplifier object with default property values.

`amp = amplifier(Name,Value)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote.

Properties

Name — Name of amplifier
`'Amplifier'` (default) | character vector

Name of amplifier, specified as a character vector. All names must be valid MATLAB variable names.

Example: `'Name','amp'`

Example: `amplifier.Name = 'amp'`

`Gain` — Available power gain

`0` (default) | real finite scalar

Available power gain, specified as a real finite scalar in dB.

Example: `'Gain',10`

Example: `amplifier.Gain = 10`

`NF` — Noise figure

`0` (default) | real finite nonnegative scalar

Noise figure, specified as a real finite nonnegative scalar dB.

Example: `'NF',-10`

Example: `amplifier.NF = -10`

`OIP3` — Output third-order intercept

`Inf` (default) | scalar

Output third-order intercept, specified as a scalar in dBm.

Example: `'OIP3',10`

Example: `amplifier.OIP3 = 10`

`Zin` — Input impedance

`50` (default) | positive real part finite scalar

Input impedance, specified as a positive real part finite scalar in ohms. You can also use a complex value with a positive real part.

Example: `'Zin',40`

Example: `amplifier.Zin = 40`

`Zout` — Output impedance

`50` (default) | positive real part finite scalar

Output impedance, specified as a scalar in ohms. You can also use a complex value with a positive real part.

Example: `'Zout',40`

Example: `amplifier.Zout = 40`

`NumPorts` — Number of ports
2 (default) | scalar integer

Number of ports, specified as a scalar integer.

Example: `'NumPorts',4`

Example: `amplifier.NumPorts = 4`

`Terminals` — Names of port terminals
`{'p1+' 'p1-'}` (default) | cell vector

Names of port terminals, specified as a cell vector. These names are always start with `p` and `n` for positive and negative ports.

Example: `'Terminals',{'p1+' 'p2+' 'p1-' 'p2-'}`

Example: `amplifier.Terminals = {'p1+' 'p2+' 'p1-' 'p2-'}`

Examples

Amplifier Element

Create an amplifier object named `'LNA'` and has a gain of 10 dB.

```
a = amplifier('Name','LNA','Gain',10)
```

```
a = 
  amplifier: Amplifier element

         Name: 'LNA'
         Gain: 10
           NF: 0
         OIP3: Inf
          Zin: 50
         Zout: 50
     NumPorts: 2
    Terminals: {'p1+'  'p2+'  'p1-'  'p2-'}
```

Amplifier Circuit

Create an amplifier object with a gain of 4 dB. Create another amplifier object that has an output third-order intercept (OIP3) 3 dBm.

```
amp1 = amplifier('Gain',4);
amp2 = amplifier('OIP3',13);
```

Build a 2-port circuit using the amplifiers.

```
c = circuit([amp1 amp2])
```

```
c =
  circuit: Circuit element

    ElementNames: {'Amplifier'  'Amplifier_1'}
           Nodes: [0 1 2 3]
            Name: 'unnamed'
        NumPorts: 2
       Terminals: {'p1+'  'p2+'  'p1-'  'p2-'}
```

RF Budget Analysis of Series of RF Elements

Create an amplifier with a gain of 4 dB.

```
a = amplifier('Gain',4);
```

Create a modulator with an OIP3 of 13 dBm.

```
m = modulator('OIP3',13);
```

Create an nport using `passive.s2p`.

```
n = nport('passive.s2p');
```

Create an rf element with a gain of 10 dB.

```
r = rfelement('Gain',10);
```

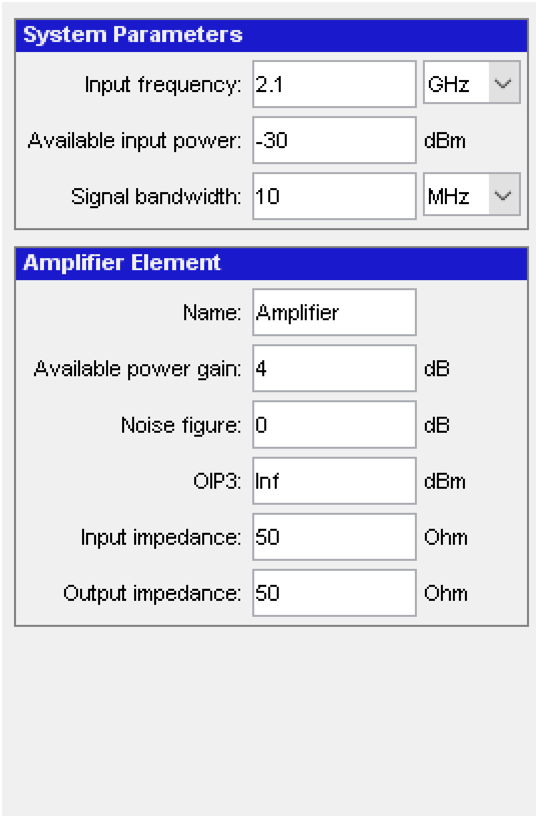
Calculate the rf budget of a series of rf elements at an input frequency of 2.1 GHz, an available input power of -30 dB, and a bandwidth of 10 MHz.

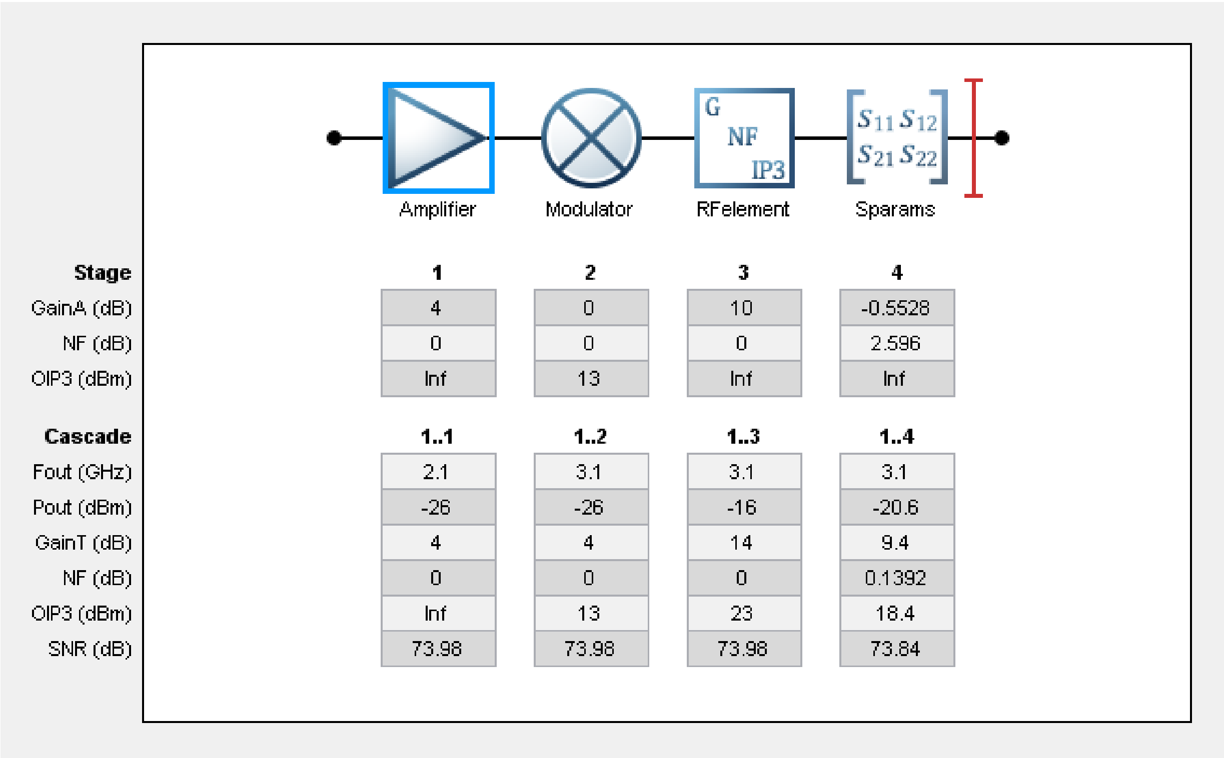
```
b = rfbudget([a m r n],2.1e9,-30,10e6)
```

```
b = 
  rfbudget with properties:

               Elements: [1x4 rf.internal.rfbudget.Element]
         InputFrequency: 2.1 GHz
    AvailableInputPower: -30 dBm
        SignalBandwidth:  10 MHz
             AutoUpdate: true

   Analysis Results
        OutputFrequency: (GHz) [  2.1    3.1    3.1     3.1]
            OutputPower: (dBm) [  -26    -26    -16   -20.6]
         TransducerGain: (dB)  [    4      4     14     9.4]
                     NF: (dB)  [    0      0      0  0.1392]
                   OIP3: (dBm) [  Inf     13     23    18.4]
                   IIP3: (dBm) [  Inf      9      9       9]
                    SNR: (dB)  [73.98  73.98  73.98   73.84]
```

Show the analysis in the RF Budget Analyzer app.

```
show(b)
```


System Parameters
Input frequency: 2.1 GHz
Available input power: -30 dBm
Signal bandwidth: 10 MHz
Amplifier Element
Name: Amplifier
Available power gain: 4 dB
Noise figure: 0 dB
OIP3: Inf dBm
Input impedance: 50 Ohm
Output impedance: 50 Ohm



See Also

`modulator` | `nport`

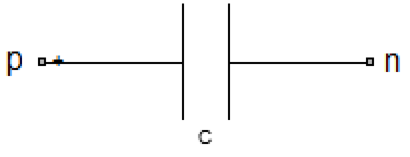
Introduced in R2017a

capacitor

Capacitor object

Description

Use the `capacitor` class to create a capacitor object that you can add to an existing circuit.



Creation

Syntax

```
cobj = capacitor(cvalue)
cobj = capacitor(cvalue,cname)
```

Description

`cobj = capacitor(cvalue)` creates a capacitor object, `cobj`, with a capacitance of `cvalue` and default name, `C`. `cvalue` must be a non-negative scalar.

`cobj = capacitor(cvalue,cname)` creates a capacitor object, `cobj`, with a capacitance of `cvalue` and name `cname`. `cname` must be a character vector.

Properties

`Capacitance` — Capacitance value
scalar

Capacitance value specified as a scalar in farads.

Example: `1e-12`

Example: `cobj.Capacitance = 1e-12`

`Name` — Name of capacitor object
C (default) | character vector

Name of capacitor object, specified as a character vector. Two elements in the same circuit cannot have the same name.

Example: `'cap'`

Example: `cobj.Name = 'cap'`

`Terminals` — Names of terminals of capacitor object
cell vector

Names of the terminals of capacitor object, specified as a cell vector. These names are always `p` and `n`.

Example: `{'p' 'n'}`

Example: `cobj.Terminals = {'p' 'n'}`

`ParentPath` — Full path of the circuit to which the capacitor object belongs
character vector

Full path of the circuit to which the capacitor object belongs, specified as character vector. This path appears only after the capacitor is added to the circuit.

Note "ParentPath" is only displayed after the capacitor has been added

into a circuit.

`ParentNodes` — Circuit nodes in the parent nodes connect to capacitor terminals
vector of integers.

Circuit nodes in the parent nodes connect to capacitor terminals, specified as a vector of integers. This property appears only after the capacitor is added to a circuit.

Example: `[1 2]`

Example: `lobj.ParentNodes = [1 2]`

Note "ParentNodes" are only displayed after the capacitor has been added

into a circuit.

Examples

Create Capacitor and Display Properties

Create a capacitor of capacitance 2 microfarad and display its properties.

```
hC1 = capacitor(2e-6);
disp(hC1)
```

```
  capacitor: Capacitor element

    Capacitance: 2.0000e-06
           Name: 'C'
      Terminals: {'p'  'n'}
```

Create and Extract S-parameters of a Capacitor

Create a capacitor and extract S-parameters of the capacitor.

```
hC = capacitor(2e-6,'C2uf');
hckt = circuit('example2');
add(hckt,[1 2],hC)
setports(hckt, [1 0],[2 0])
freq = linspace(1e3,2e3,100);
S = sparameters(hckt,freq);
disp(S)
```

```
  sparameters: S-parameters object
```

```
      NumPorts: 2
   Frequencies: [100x1 double]
    Parameters: [2x2x100 double]
     Impedance: 50

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

Add Capacitor to Circuit and Display Properties

Add capacitor to a circuit, display the parent path and parent nodes.

```
hC3 = capacitor(3e-6,'C3uf');
hckt3 = circuit('example3');
add(hckt3,[1 2],hC3)
setports(hckt3, [1 0],[2 0])
disp(hC3)
```

```
  capacitor: Capacitor element

    Capacitance: 3.0000e-06
           Name: 'C3uf'
      Terminals: {'p'  'n'}
    ParentNodes: [1 2]
     ParentPath: 'example3'
```

See Also

circuit | inductor | lcladder | resistor

Introduced in R2013b

rfckt.amplifier

RF Amplifier

Description

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

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

- Touchstone
- Agilent P2D
- Agilent S2D
- AMP

Note If you set `NonLinearData` using `rfdata.ip3` or `rfdata.power`, then the property is converted from scalar OIP3 format to the format of `rfdata.ip3` or `rfdata.power`.

Creation

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,...)` sets properties using one or more name-value pairs. For example, `rfckt.amplfier ('IntType','Cubic')`creates an RF amplifier circuit that uses cubic interpolation. You can specify multiple name-value pairs. Enclose each property name in a quote.

Properties

`AnalyzedResult` — Computed S-parameters, noise figure, OIP3, and group delay values
`rfdata.data` object

This property is read-only.

Computed S-parameters, noise figure, OIP3, and group delay values, specified as an `rfdata.data` object. For more information refer, .“Algorithms” on page 6-202.

Data Types: `function_handle`

`IntpType` — Interpolation method used in `rfckt.amplifier`
`'Linear'` (default) | `'Spline'` | `'Cubic'`

Interpolation method specified one of the following values:

Method	Description
`Linear`	Linear interpolation
`Spline`	Cubic spline interpolation
`Cubic`	Piecewise cubic Hermite interpolation

Data Types: `char`

`Name` — Name of amplifier object
`1-by-N` character array

This property is read-only.

Name of amplifier object.

Data Types: `char`

`NetworkData` — Network parameter data
`rfdata.network` object

Network parameter data.

Data Types: `function_handle`

`NoiseData` — Noise information
scalar noise figure in decibels | `rfdata.noise` object | `rfdata.nf` object

Noise information, specified as one of the following:

- Scalar noise figure in dB
- `rfdata.noise` object
- `rfdata.nf` object

Data Types: `double` | `function_handle`

`'NonlinearityData'` — Nonlinearity information
scalar OIP3 in dB | `rfdata.power` object | `rfdata.ip3` object

Noise information, specified as one of the following:

- Scalar OIP3 in dB
- `rfdata.power` object
- `rfdata.ip3` object

Data Types: `double` | `function_handle`

`nport` — Number of ports
positive integer

This property is read-only.

Number of ports. The default value is 2.

Data Types: `double`

Object Functions

analyze	Analyze circuit object in frequency domain
calculate	Calculate specified parameters for circuit object
circle	Draw circles on Smith Chart
extract	Extract array of network parameters from data object

Examples

Create RF Circuit Amplifier

Create an Amplifier using rfckt.amplifier object.

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

```
amp = 
   rfckt.amplifier with properties:

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

- “RF Circuit Objects”
- “RF Data Objects”

Algorithms

The `analyze` function computes the `AnalyzedResult` property using the data stored in the `rfckt.amplifier` object properties as follows:

- The `analyze` function uses the data stored in the `NoiseData` property of the `rfckt.amplifier` object to calculate the noise figure.
- The `analyze` function uses the data stored in the `NonlinearData` property of the `rfckt.amplifier` object to calculate OIP3.

  If power data exists in the `NonlinearData` property, the block extracts the AM/AM and AM/PM nonlinearities from the power data.

  If the `NonlinearData` property contains only IP3 data, the method computes and adds the nonlinearity by:

1 Using the third-order input intercept point value in dBm to compute the factor, *f*, that scales the input signal before the amplifier object applies the nonlinearity:

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

2 Computing the scaled input signal by multiplying the amplifier input signal by *f*.

3 Limiting the scaled input signal to a maximum value of 1.

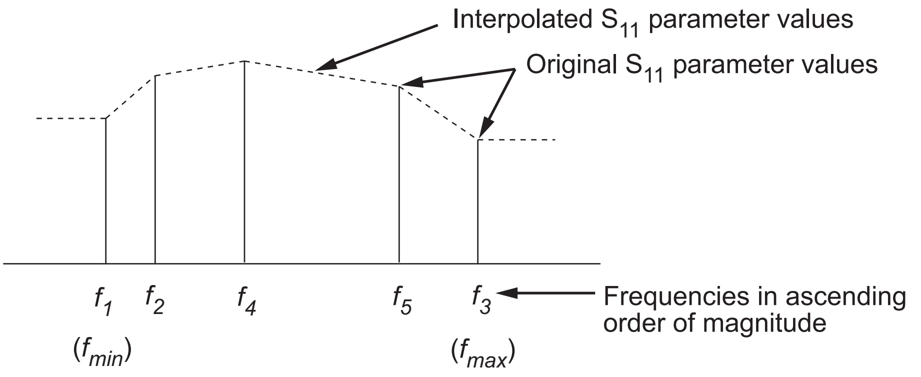
4 Applying an AM/AM conversion to the amplifier gain, according to the following cubic polynomial equation:

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

where u is the magnitude of the scaled input signal, which is a unitless normalized input voltage.

- The `analyze` function uses the data stored in the `'NetworkData'` property of the `rfckt.amplifier` object to calculate the group delay values of the amplifier at the frequencies specified in `freq`, as described in the `analyze` reference page.
- The `analyze` function 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-parameters or Z-parameters, the `analyze` function first converts the parameters to S-parameters. Using the interpolation method you specify with the `IntpType` property, the `analyze` method interpolates the S-parameter values to determine their values at the specified frequencies.

  Specifically, the `analyze` function orders the S-parameters according to the ascending order of their frequencies, f_n. It then interpolates the S-parameters, using the MATLAB `interp1` function. For example, the curve in the following diagram illustrates the result of interpolating the S_{11} parameters at five different frequencies.



For more information, see “One-Dimensional Interpolation” and the `interp1` reference page.

As shown in the preceding diagram, the `analyze` function uses the parameter values at f_{min}, the minimum input frequency, for all frequencies smaller than f_{min}. It uses the parameters values at f_{max}, the maximum input frequency, for all frequencies greater than f_{max}. In both cases, the results may not be accurate, so you need to specify network parameter values over a range of frequencies that is wide enough to account for the amplifier behavior.

References

[1] EIA/IBIS Open Forum. *Touchstone File Format Specification*, Rev. 1.1, 2002 (https://ibis.org/connector/touchstone_spec11.pdf).

See Also

`rfckt.datafile` | `rfckt.mixer` | `rfckt.passive` | `rfdata.data` | `rfdata.ip3` | `rfdata.nf` | `rfdata.noise`

Topics

“RF Circuit Objects”
“RF Data Objects”

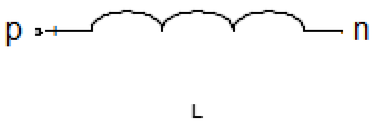
Introduced before R2006a

inductor

Inductor object

Description

Use `inductor` class to create an inductor object that you can add to an existing circuit.



Creation

Syntax

```
lobj = inductor(lvalue)
lobj = inductor(lvalue,cname)
```

Description

`lobj = inductor(lvalue)` creates a inductor object, `lobj`, with a inductance of `lvalue` and default name, `L`. `lvalue` must be a numeric positive scalar.

`lobj = inductor(lvalue,cname)` creates a inductor object, `lobj`, with a inductance of `lvalue` and name `lname`. `lname` must be a character vector.

Properties

`Inductance` — Inductance
scalar

Inductance, specified as a scalar in henries.

Example: `1e-9`

Example: `lobj.Inductance = 1e-9`

Name — Object name
L (default) | character vector

Name of inductor object, specified as a character vector. Two elements in the same circuit cannot have the same name.

Example: `'cap'`

Example: `lobj.Name = 'indcutor1'`

`Terminals` — Names of terminals of inductor object
cell vector

Names of the terminals of inductor object, specified as a cell vector These names are always `p` and `n`.

Example: `{'p' 'n'}`

Example: `lobj.Terminals = {'p' 'n'}`

`ParentPath` — Full path of the circuit
character vector

Full path of the circuit to which the inductor object belongs, specified as character vector. This path appears only after the inductor is added to the circuit.

Note "ParentPath" is only displayed after the capacitor has been added

into a circuit.

`ParentNodes` — Parent nodes connected to inductor terminals
vector of integers.

Parent nodes connected to inductor terminals, specified as a vector of integers. This property appears only after the inductor is added to a circuit.

Example: `[1 2]`

Example: `lobj.ParentNodes = [1 2]`

Note "ParentNodes" are only displayed after the capacitor has been added into a circuit.

Examples

Create and Display Inductor

Create an inductor of 3e-9 henry and display the properties.

```
hL1 = inductor(3e-9);
disp(hL1)
```

```
  inductor: Inductor element

    Inductance: 3.0000e-09
          Name: 'L'
     Terminals: {'p'  'n'}
```

Create and Extract S-parameters of Inductor

Create an inductor object and extract the s-parameters of this inductor.

```
hL = inductor(3e-9,'L3nh');
hckt = circuit('example2');
add(hckt,[1 2],hL)
setports (hckt, [1 0],[2 0])
freq = linspace (1e3,2e3,100);
S = sparameters(hckt,freq);
disp(S)
```

```
  sparameters: S-parameters object

       NumPorts: 2
    Frequencies: [100x1 double]
     Parameters: [2x2x100 double]
```

```
      Impedance: 50

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

Add Inductor to Circuit and Display Properties

Add an inductor to a circuit, display the parent path and parent nodes.

```
hL = inductor(3e-9,'L3n9');
hckt = circuit('example3');
add(hckt,[1 2],hL)
setports(hckt, [1 0],[2 0])
disp(hL)
```

```
  inductor: Inductor element

     Inductance: 3.0000e-09
           Name: 'L3n9'
      Terminals: {'p'  'n'}
    ParentNodes: [1 2]
     ParentPath: 'example3'
```

See Also

capacitor | circuit | resistor

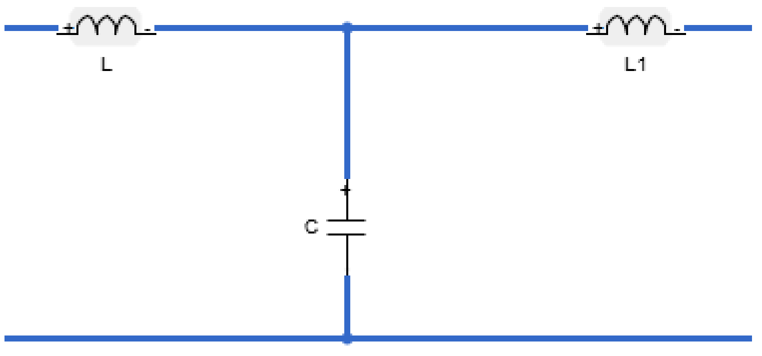
Introduced in R2013b

lcladder

LC ladder object

Description

`lcladder` class creates an LC ladder object that you can add to an existing circuit. Create filters and calculate s-parameters of filters using `lcladder` class. You can also add the `lcladder` object to an existing circuit.



Creation

Syntax

```
lcobj = lcladder(topology,inductances,capacitances)
lcobj = lcladder(topology,inductances,capacitances,lcname)
```

Description

`lcobj = lcladder(topology,inductances,capacitances)` creates an LC ladder object, `lcobj`, with a topology, `top`, inductor values, `l`, and capacitor values, `c`.

`lcobj = lcladder(topology,inductances,capacitances,lcname)` creates an LC ladder object, `lcobj`, with a name, `lcname`. `lcname` must be a character vector.

Properties

Topology — Topology type of the LC ladder
character vector

Topology type of the LC ladder, specified as a one of the following character vector:

- `'lowpasspi'`: Low-pass pi filter
- `'lowpasstee'`: Low-pass tee filter
- `'highpasspi'`: High-pass pi filter
- `'highpasstee'`: High-pass tee filter
- `'bandpasspi'`: Band-pass pi filter
- `'bandpasstee'`: Band-pass tee filter
- `'bandstoppi'`: Band-stop pi filter
- `'bandstoptee'`: Band-stop tee filter

Set the topology type in the `topology` argument of the syntax.

Example: `'lowpasspi'`

Inductances — Inductor values in LC ladder
numeric scalar or vector

Inductor values in LC ladder, specified as a numeric scalar or vector in henries. Set the inductor value in the `inductances` argument of the syntax.

Example: `3.18e-8`

Capacitances — Capacitor values in LC ladder
numeric scalar or vector

Capacitor values in LC ladder, specified as a numeric scalar or vector in farads. Set the capacitor value in the `c` argument of the syntax.

Example: `[6.37e-12 6.37e-12]`

`Name` — Name of LC ladder object
`'lcfilt'` (default) | character vector

Name of LC ladder object, specified as a character vector. Set the name of the LC ladder in `lcname` argument of the syntax.

Example: `'lcfilter'`

`NumPorts` — Number of ports in LC ladder object
scalar

Number of ports in LC ladder object. specified as a scalar. This value is always 2.

`Terminals` — Terminal names of LC ladder object
{'p1+' 'p2+' 'p1-' 'p2-'} | cell vector

Terminal names of LC ladder object, specified as the cell vector, `{'p1+' 'p2+' 'p1-' 'p2-'}`. An LC ladder object always has four terminals: two terminals associated with the first port (`'p1+'` and `'p1-'`) and two terminals associated with the second port (`'p2+'` and `'p2-'`).

Example: `{'p1+' 'p2+' 'p1-' 'p2-'}`

`ParentNodes` — Parent circuit nodes connected to LC ladder object terminals
vector of integers

Parent circuit nodes connected to LC ladder object terminals, specified as a vector of integers. This property appears only after the LC ladder object is added to a circuit.

Note "ParentNodes" are only displayed after the capacitor has been added

into a circuit.

`ParentPath` — Full path of the circuit to which the LC ladder object belongs
character vector

Full path of the circuit to which the LC ladder object belongs, specified as character vector. This path appears only after the inductor is added to the circuit.

Note "ParentPath" is only displayed after the capacitor has been added

into a circuit.

Examples

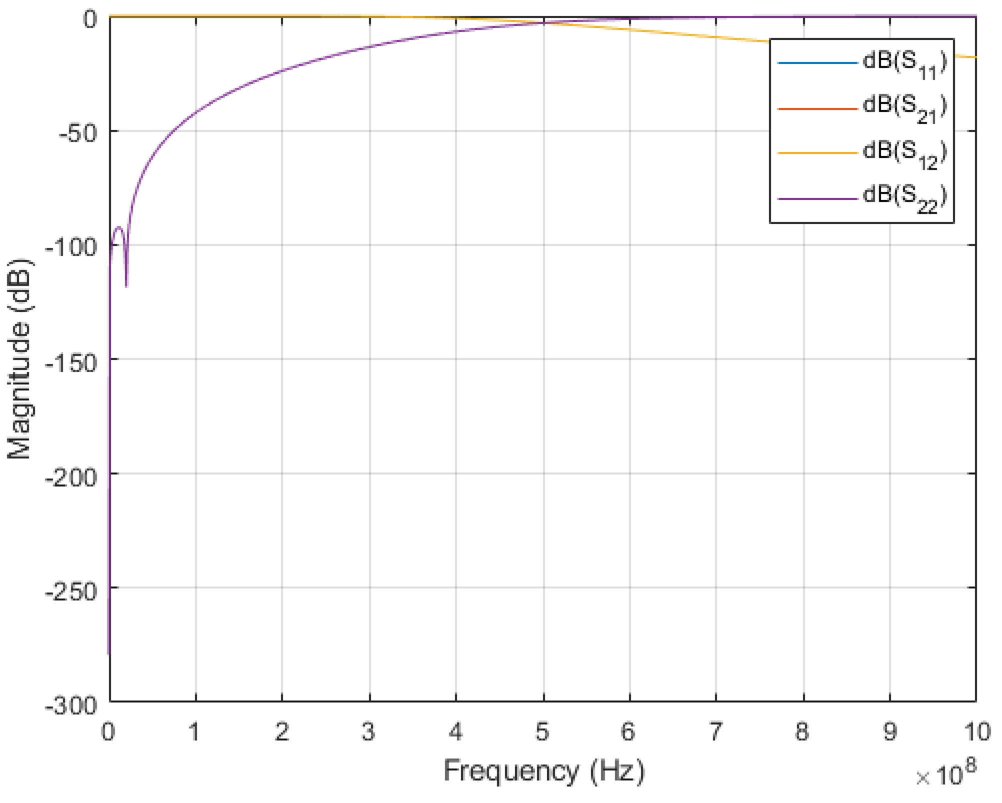
Create Low-Pass Pi LC Ladder Object and Plot S-Parameters

Create a low-pass pi lc ladder object with inductor value, 3.18e-8 and capacitor value, 6.37e12. Calculate and plot the s-parameters.

```
L = 3.18e-8;
C = [6.37e-12 6.37e-12];
lpp = lcladder('lowpasspi',L,C)
```

```
lpp =
  lcladder: LC Ladder element

        Topology: 'lowpasspi'
     Inductances: 3.1800e-08
    Capacitances: [6.3700e-12 6.3700e-12]
            Name: 'lcfilt'
        NumPorts: 2
       Terminals: {'p1+'  'p2+'  'p1-'  'p2-'}
```

```
freq = 0:1e6:1e9;
S = sparameters(lpp,freq);
rfplot(S)
```



See Also

`capacitor` | `circuit` | `inductor`

Introduced in R2013b

nport

Create linear n-port circuit element

Description

The `nport` class creates an n-port object that can be added into an RF Toolbox circuit. The n-port S-parameters define the n-port object.

Creation

Syntax

```
nport_obj = nport(filename)
nport_obj = nport(sparam_obj)
```

Description

`nport_obj = nport(filename)` creates an n-port object from the specified `filename`.

`nport_obj = nport(sparam_obj)`creates an n-port object from an S-parameters data object.

Properties

`NumPorts` — Number of ports
scalar

Number of ports, specified as a scalar.

Example: 2

`NetworkData` — S-parameter data
scalar

S-parameter data, specified as a scalar. The linear S-parameter data defines the n-port object.

Example: [1x1 sparameters]

`Name` — Name of n-port object
scalar handle

Name of n-port object, specified as a scalar handle.

Example: obj

`Ports` — Port names
cell vector

Port names, stored as a cell vector. This property is a read only.

Example: {'p1' 'p2'}

`Terminals` — Terminal names
cell vector

Terminal names, stored as a cell vector. There are two terminals per port. The positive terminal names are listed first ('p1+', 'p2+'...) followed by the negative terminal ('p1–', 'p2–'...). This property is read only.

`ParentNodes` — Parent circuit nodes connected to n-port object terminals
vector of integers.

Parent circuit nodes connected to n-port object terminals, stored as a vector of integers. `ParentNodes` is same length as `Terminals`. This property is read only and appears only after you add the n-port data.

`ParentPath` — Full path of circuit to which n-port object belongs
character vector

Full path of the circuit to which the n-port object belongs, stored as character vector. This property is read only and appear only after you add the n-port object is added to the circuit.

Examples

Create N-port Object

Create and display N-port data object.

```
npass = nport('passive.s2p')
```

```
npass = 
  nport: N-port element

    NetworkData: [1x1 sparameters]
           Name: 'Sparams'
       NumPorts: 2
      Terminals: {'p1+'  'p2+'  'p1-'  'p2-'}
```

Add N-Port Object to Circuit

Add a N-port object to a circuit. Display the object.

```
nobj = nport('passive.s2p');
ckt = circuit('example');
add(ckt,[1 2],nobj)
disp(nobj)
```

```
  nport: N-port element

    NetworkData: [1x1 sparameters]
           Name: 'Sparams'
       NumPorts: 2
      Terminals: {'p1+'  'p2+'  'p1-'  'p2-'}
    ParentNodes: [1 2 0 0]
     ParentPath: 'example'
```

See Also

capacitor | circuit | inductor | resistor | rfbudget

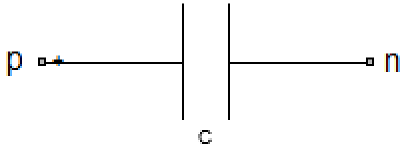
Introduced in R2013b

resistor

Resistor object

Description

Use the `capacitor` class to create a capacitor object that you can add to an existing circuit.



Creation

Syntax

```
robj = resistor(rvalue)
robj = resistor(rvalue,rname)
```

Description

`robj = resistor(rvalue)` with a resistance of `rvalue` and default name, `R`. `rvalue` must be a numeric non-negative scalar.

`robj = resistor(rvalue,rname)` creates a resistor object, `robj`, with a resistance of `rvalue` and name `rname`. `rname` must be a character vector.

Properties

`Resistance` — Resistance value
scalar

Resistance value specified as a scalar in ohms.

Example: `50`

Example: `robj.Resistance = 50`

`Name` — Name of resistor object
R (default) | character vector

Name of resistor object, specified as a character vector. Two elements in the same circuit cannot have the same name.

Example: `'resis'`

Example: `robj.Name = 'resis'`

`Terminals` — Names of terminals of capacitor object
cell vector

Names of the terminals of capacitor object, specified as a cell vector. These names are always `p` and `n`.

Example: `{'p' 'n'}`

Example: `robj.Terminals = {'p' 'n'}`

`ParentPath` — Full path of the circuit to which the capacitor object belongs
character vector

Full path of the circuit to which the capacitor object belongs, specified as character vector. This path appears only after the capacitor is added to the circuit.

Note "ParentPath" is only displayed after the capacitor has been added

into a circuit.

`ParentNodes` — Circuit nodes in the parent nodes connect to capacitor terminals
vector of integers.

Circuit nodes in the parent nodes connect to capacitor terminals, specified as a vector of integers. This property appears only after the capacitor is added to a circuit.

Example: `[1 2]`

Example: `robj.ParentNodes = [1 2]`

Note "ParentNodes" are only displayed after the capacitor has been added

into a circuit.

Examples

Create Resistor and Display Properties

Create a resistor of resistance 100 ohms and display its properties.

```
hR1 = resistor(100);
disp(hR1)
```

```
  resistor: Resistor element

    Resistance: 100
          Name: 'R'
     Terminals: {'p'  'n'}
```

Create and Extract S-parameters of Resistor

Create an resistor object and extract the s-parameters of this resistor.

```
hR = resistor(50,'R50');
hckt = circuit('example2');
add(hckt,[1 2],hR)
setports (hckt, [1 0],[2 0])
freq = linspace (1e3,2e3,100);
S = sparameters(hckt,freq);
disp(S)
```

```
  sparameters: S-parameters object
```

```
      NumPorts: 2
   Frequencies: [100x1 double]
    Parameters: [2x2x100 double]
     Impedance: 50

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

Add Resistor to Circuit and Display Properties

Add a resistor to a circuit, display the parent path and parent nodes.

```
hR = resistor(150,'R150');
hckt = circuit('resistorcircuit');
add(hckt,[1 2],hR)
setports(hckt, [1 0],[2 0])
disp(hR)
```

```
  resistor: Resistor element

    Resistance: 150
          Name: 'R150'
     Terminals: {'p'  'n'}
   ParentNodes: [1 2]
    ParentPath: 'resistorcircuit'
```

See Also

capacitor | circuit | inductor

Introduced in R2013b

rfchain

Create `rfchain` object

Description

Use the `rfchain` object to create an RF circuit cascade analysis object to calculate gain, noise figure, OIP3 (output third-order intercept), and IIP3 (input third order intercept).

Creation

Syntax

```
rfobj = rfchain()
obj = rfchain(g, nf, o3, 'Name', nm)
obj = rfchain(g, nf, 'IIP3', i3, 'Name', nm)
```

Description

`rfobj = rfchain()` creates an RF chain object `obj` having zero stages. To add stages to the RF chain, use `addstage` method.

`obj = rfchain(g, nf, o3, 'Name', nm)` creates an RF chain object `obj` having `N` stages. The gain `g`, noise figure `nf` and the OIP3 `o3` are vectors of length `N` . The name `nm` is a cell array of length `N` .

`obj = rfchain(g, nf, 'IIP3', i3, 'Name', nm)` creates an RF chain object having `N` stages, specifying an IIP3 for each stage, instead of an OIP3.

Properties

`Numstages` — Number of stages
scalar

Number of stages in an RF chain, specified as a scalar.

Data Types: `double`

`Name` — Name of each stage
character vector

Name of each stage of an RF chain, specified as a character vector. This will always be a name-value pair.

Data Types: `char`

`Gain` — Gain of each stage
vector

Gain, in dB, of each stage in an RF chain, specified as a vector.

Data Types: `double`

`NoiseFigure` — Noise figure of each stage
vector

Noise figure, in dB, of each stage in an RF chain, specified as a vector.

Data Types: `double`

`OIP3` — Output-referred third-order intercept
vector

Output-referred third-order intercept, in dB, of each stage in an RF chain, specified as a vector.

Data Types: `double`

`IIP3` — Input-referred third-order intercept
vector

Input-referred third-order intercept, in dB, of each stage in an RF chain, specified as a vector.

Data Types: `double`

Examples

Create RF Chain Object, Add Stages, and View Results

Create an RF chain object.

```
rfch = rfchain;
```

Add stage 1 and stage 2 with gain, noise figure, oip3.

```
addstage(rfch, 21, 15, 30, 'Name', 'amp1');
addstage(rfch, -5, 6, Inf, 'Name', 'filt1');
```

Add stage 3 and stage 4 with gain, noise figure, iip3.

```
addstage(rfch, 7, 5, 'IIP3', 10, 'Name', 'lna1');
addstage(rfch, 12, 14, 'IIP3', 20, 'Name', 'amp2');
```

Calculate the gain, noise figure, oip3, and iip3 of each stage.

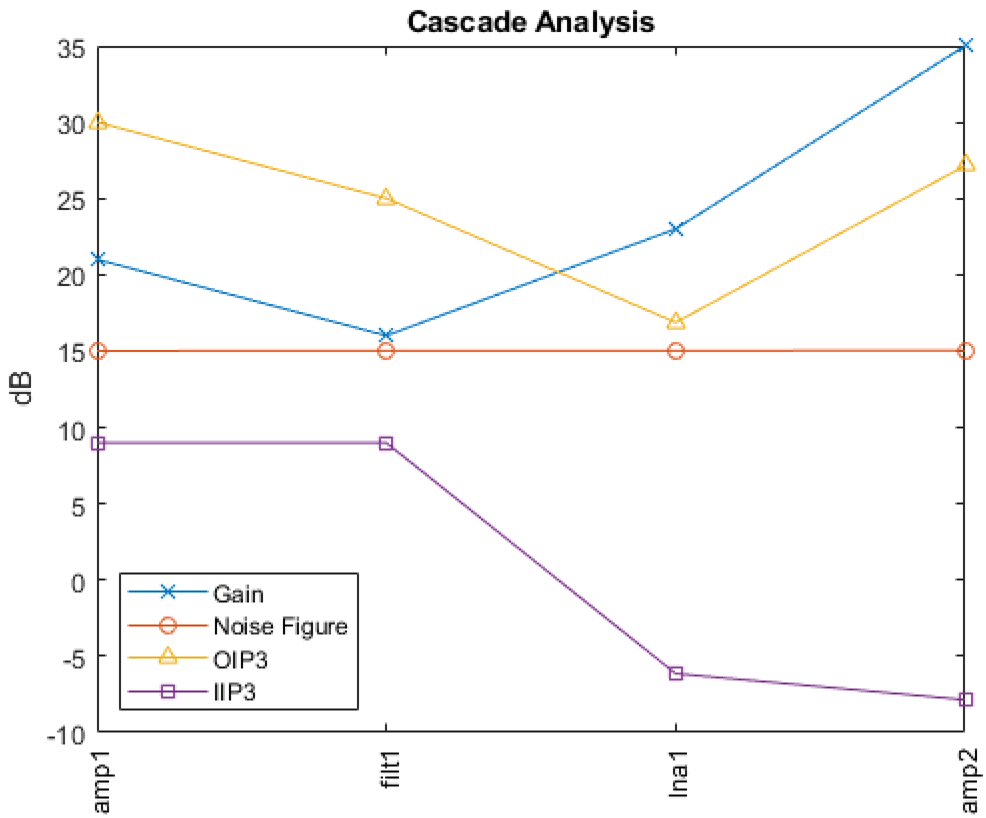
```
g = cumgain(rfch);
nf = cumnoisefig(rfch);
oip3val = cumoip3(rfch);
iip3val = cumiip3(rfch);
```

View the results on a table and plot it.

```
worksheet(rfch)
```

	amp1	filt1	lna1	amp2
Stage Gain	21	-5	7	12
Stage Noise Figure	15	6	5	14
Stage OIP3	30	Inf	17	32
Stage IIP3	9	Inf	10	20
Cascaded Gain	21	16	23	35
Cascaded Noise Figure	15	15.0033	15.0107	15.0272
Cascaded OIP3	30	25	16.8648	27.1451
Cascaded IIP3	9.0000	9.0000	-6.1352	-7.8549

```
figure
plot(rfch)
```



Create RF Chain Adding Stage-By-Stage Values

Assign three stage-by-stage values of gain, noise figure, OIP3 and stage names.

```
g = [11 -3 7];
nf = [25 3 5];
o3 = [30 Inf 10];
nm = {'amp1','filt1','lna1'};
```

Create an RF chain object.

```
rfch = rfchain(g,nf,o3,'Name',nm);
```

View results in a worksheet.

```
worksheet(rfch)
```

	amp1	filt1	lna1
Stage Gain	11	-3	7
Stage Noise Figure	25	3	5
Stage OIP3	30	Inf	10
Stage IIP3	19	Inf	3
Cascaded Gain	11	8	15
Cascaded Noise Figure	25	25.0011	25.0058
Cascaded OIP3	30	27	9.9827
Cascaded IIP3	19	19	-5.0173

See Also

Introduced in R2013b

modulator

Modulator object

Description

Use the `modulator` object to create an modulator element. A modulator is a 2-port RF circuit object. You can use this element in the `rfbudget` object and the `circuit` object.

Creation

Syntax

```
mod = modulator
mod = modulator(Name,Value)
```

Description

`mod = modulator` creates a modulator object, `mod`, with default property values.

`mod = modulator(Name,Value)` creates a modulator object with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values.

Properties

Name — Name of modulator
`'Modulator'` (default) | character vector

Name of modulator, specified as the comma-separated pair consisting of `'Name'` and a character vector. All names must be valid MATLAB variable names.

Example: 'Name','mod'

Gain — Available power gain
0 (default) | nonnegative scalar

Available power gain, specified as a nonnegative scalar in dB.

Example: 'Gain',10

NF — Noise figure
0 (default) | real finite nonnegative scalar

Noise figure, specified as a real finite nonnegative scalar in dB.

Example: 'NF',-10

OIP3 — Output third-order intercept
Inf (default) | scalar

Output third-order intercept, specified as a scalar in dBm

Example: 'OIP3',10

LO — Local oscillator frequency
1e9 (default) | real finite positive scalar

Local oscillator frequency, specified as a real finite positive scalar in Hz.

Example: 'LO',2e9

ConverterType — Type of modulator
'Up' (default) | 'Down'

Type of modulator, specified as 'Down' or 'Up'

Example: 'ConverterType','Up'

Zin — Input impedance
50 (default) | positive real part finite scalar

Input impedance, specified as a positive real part finite scalar in ohms. You can also use a complex value with a positive real part.

Example: 'Zin',40

`Zout` — Output impedance
`50` (default) | positive real part finite scalar

Output impedance, specified as a scalar in ohms. You can also use a complex value with a positive real part.

Example: `'Zout',40`

`NumPorts` — Number of ports
2 (default) | scalar integer

Number of ports, specified as a scalar integer.

Example: `'NumPorts',4`

`Terminals` — Names of port terminals
`{'p1+' 'p1-'}` (default) | cell vector

Names of port terminals, specified as a cell vector These names are always `p` and `n` for positive and negative nodes.

Example: `'Terminals',{'p1+' 'p2+' 'p1-' 'p2-'}`

Examples

Modulator Element

Create a downconverter modulator with a local oscillator (LO) frequency of 100 MHz.

```
m = modulator('ConverterType','Down','LO',100e6)
```

```
m =
  modulator: Modulator element

             Name: 'Modulator'
             Gain: 0
               NF: 0
             OIP3: Inf
               LO: 100000000
    ConverterType: 'Down'
              Zin: 50
             Zout: 50
```

```
        NumPorts: 2
       Terminals: {'p1+'  'p2+'  'p1-'  'p2-'}
```

Modulator Circuit

Create a modulator object with a gain of 4 dB and local oscillator (LO) frequency of 2.9 GHz. Create another modulator object that is an upconvertor and has an output third-order intercept (OIP3) of 13 dBm.

```
mod1 = modulator('Gain',4,'LO',2e9);
mod2 = modulator('OIP3',13,'ConverterType','Up');
```

Build a 2-port circuit using the modulators.

```
c = circuit([mod1 mod2])
```

```
c = 
  circuit: Circuit element

    ElementNames: {'Modulator'  'Modulator_1'}
           Nodes: [0 1 2 3]
            Name: 'unnamed'
        NumPorts: 2
       Terminals: {'p1+'  'p2+'  'p1-'  'p2-'}
```

RF Budget Analysis of Series of RF Elements

Create an amplifier with a gain of 4 dB.

```
a = amplifier('Gain',4);
```

Create a modulator with an OIP3 of 13 dBm.

```
m = modulator('OIP3',13);
```

Create an nport using `passive.s2p`.

```
n = nport('passive.s2p');
```

Create an rf element with a gain of 10 dB.

```
r = rfelement('Gain',10);
```

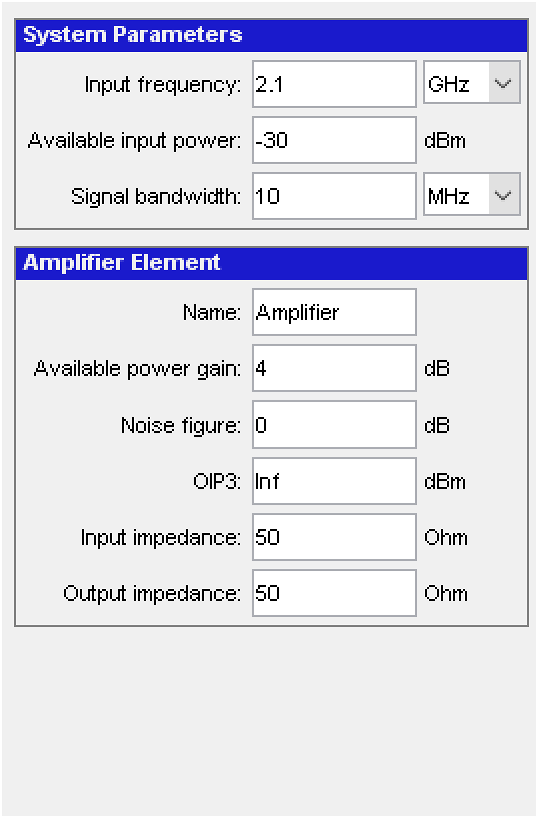
Calculate the rf budget of a series of rf elements at an input frequency of 2.1 GHz, an available input power of -30 dB, and a bandwidth of 10 MHz.

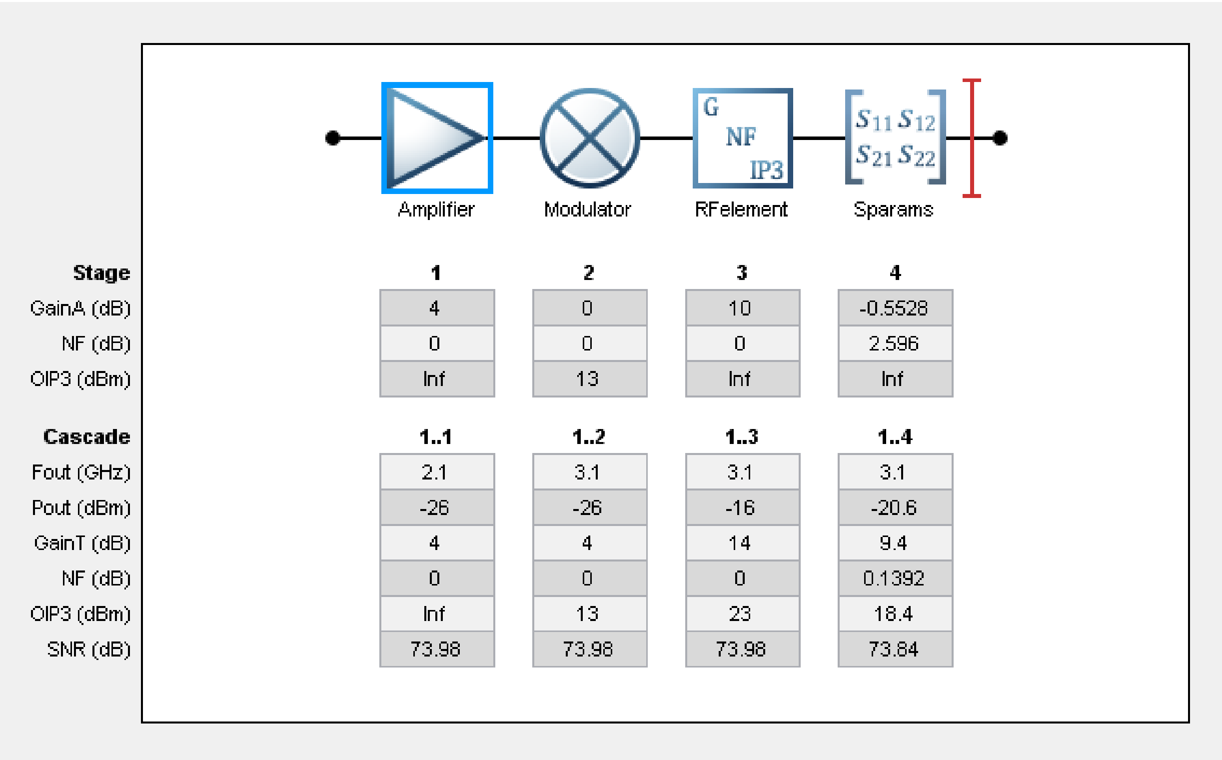
```
b = rfbudget([a m r n],2.1e9,-30,10e6)
```

```
b = 
  rfbudget with properties:

               Elements: [1x4 rf.internal.rfbudget.Element]
         InputFrequency: 2.1 GHz
    AvailableInputPower: -30 dBm
        SignalBandwidth:  10 MHz
             AutoUpdate: true

   Analysis Results
        OutputFrequency: (GHz) [  2.1    3.1    3.1     3.1]
            OutputPower: (dBm) [  -26    -26    -16   -20.6]
         TransducerGain: (dB)  [    4      4     14     9.4]
                     NF: (dB)  [    0      0      0  0.1392]
                   OIP3: (dBm) [  Inf     13     23    18.4]
                   IIP3: (dBm) [  Inf      9      9       9]
                    SNR: (dB)  [73.98  73.98  73.98   73.84]
```

Show the analysis in the RF Budget Analyzer app.

```
show(b)
```


System Parameters
Input frequency: 2.1 GHz
Available input power: -30 dBm
Signal bandwidth: 10 MHz
Amplifier Element
Name: Amplifier
Available power gain: 4 dB
Noise figure: 0 dB
OIP3: Inf dBm
Input impedance: 50 Ohm
Output impedance: 50 Ohm



See Also

`amplifier` | `nport` | `rfbudget`

Introduced in R2017a

circuit

Circuit object

Description

Use `circuit` object to build a circuit object which can contain elements like resistor, capacitor, and inductor.

Creation

Syntax

```
cktobj = circuit
cktobj = circuit(cktname)
cktobj = circuit([elem1,elem2,...])
cktobj = circuit([elem1,elem2,...],cktname)
cktobj = circuit(rfb)
cktobj = circuit(rfb,cktname)
```

Description

`cktobj = circuit` creates a circuit object `cktobj` with a default name.

`cktobj = circuit(cktname)` creates a circuit object `cktobj` with name of `cktname`.

`cktobj = circuit([elem1,elem2,...])` creates a circuit object `cktobj` by cascading the specified 2-port elements.

`cktobj = circuit([elem1,elem2,...],cktname)` creates a cascaded circuit object `cktobj` with the name, `cktname`.

`cktobj = circuit(rfb)` creates a circuit object `cktobj` by cascading the elements in the RF object, `rfb`.

`cktobj = circuit(rfb,cktname)` creates a circuit object `cktobj` by cascading the elements in the RF object, `rfb`, using name, `cktname`.

Input Arguments

`elem1,elem2...` — 2-port RF elements
character vector

2-port RF elements, specified as character vectors. The possible elements are `modulator`, `nport`, and `amplifier`

`rfb` — RF budget object
object handle

RF budget object, specified as an object handle.

Properties

`Name` — Object Name
unnamed (default) | character vector

Name of circuit, specified as a character vector. Default name is `unnamed`. Two circuit elements attached together or belonging to the same circuit cannot have the same name

`ElementNames` — Name of elements in the circuit
cell vector

Name of elements in the circuit, specified as a vector of cell vector. The possible elements here are resistor, capacitor, inductor, and circuit.

`Terminals` — Names of terminals in the circuit
cell vector

Names of terminals in the circuit, specified as a cell vector. Use `setterminals` or `setports` function to define the terminals. The terminals of the circuit are only displayed once it is defined.

`Ports` — Names of ports in a circuit
character vector

Names of ports in a circuit specified as a character vector. Use `setports` function to define the ports. The ports of the circuit are only displayed once it is defined.

`Nodes` — List of nodes defined in circuit
vector of integers

List of nodes defined in the circuit, specified as a vector of integers. These nodes are created when a new element is attached to the circuit.

`ParentPath` — Full path of parent circuit
character vector

Full path of parent circuit, specified as a character vector. This path appears only once the child circuit is added to the parent circuit.

`ParentNodes` — Nodes of parent circuit
vector of integers.

Nodes of parent circuit, specified as a vector of integers. This vector of integers is the same length as the `Terminals` property. This property appears only after the child circuit is added to the parent circuit.

`ParentNodes` — Circuit nodes in the parent nodes connect to capacitor terminals
vector of integers.

Circuit nodes in the parent nodes connect to capacitor terminals, specified as a vector of integers. This property appears only after the capacitor is added to a circuit.

Example: `[1 2]`

Example: `lobj.ParentNodes = [1 2]`

Note "ParentNodes" are only displayed after the capacitor has been added

into a circuit.

Examples

Create Circuit with Elements and Terminals

Create a circuit called new_circuit. Add a resistor and capacitor to the circuit. Set the terminals and display the results.

```
hckt = circuit('new_circuit1');
hC1= add(hckt,[1 2],capacitor(3e-9));
hR1 = add(hckt,[2 3],resistor(100));
setterminals (hckt,[1 3]);
disp(hckt)
```

```
  circuit: Circuit element

    ElementNames: {'C'  'R'}
           Nodes: [1 2 3]
            Name: 'new_circuit1'
       Terminals: {'t1'  't2'}
```

Create Circuit with Two Parallel Elements

Create a circuit called new_circuit. Add a capacitor and inductor parallel to the circuit.

```
hckt = circuit('new_circuit');
hC = add(hckt,[1 2],capacitor(1e-12));
hL = add(hckt,[1 2],inductor(1e-9));
disp(hckt)
```

```
  circuit: Circuit element

    ElementNames: {'C'  'L'}
           Nodes: [1 2]
            Name: 'new_circuit'
```

- “Bandpass Filter Response”
- “MOS Interconnect and Crosstalk”

See Also

amplifier | inductor | lcladder | modulator | nport | resistor | sparameters

Topics

"Bandpass Filter Response"
"MOS Interconnect and Crosstalk"

Introduced in R2013b

rfelement

Generic RF element object

Description

Use the `rfelement` object to create a generic RF element. An RF element is a 2-port RF circuit object. You can use this element in the `rfbudget` object and the `circuit` object.

Creation

Syntax

```
rfel = rfelement
rfel = rfelement(Name,Value)
```

Description

`rfel = rfelement` creates an RF element object with default property values.

`rfel = rfelement(Name,Value)` sets properties using one or more name-value pairs. You can specify multiple name-value pairs. Enclose each property name in a quote.

Properties

Name — Name given to identify RF element
`'RFelement'` (default) | character vector

Name given to identify rf element, specified as a character vector. All names must be valid MATLAB variable names.

Example: `'Name','rfel'`

Example: `rfel.Name = 'rfel'`

`Gain` — Available power gain
`0` (default) | scalar

Available power gain, specified as a scalar in dB.

Example: `'Gain',10`

Example: `rfel.Gain = 10`

`NF` — Noise figure
`0` (default) | real finite nonnegative scalar

Noise figure, specified as a real finite nonnegative scalar dB.

Example: `'NF',-10`

Example: `rfel.NF = -10`

`OIP3` — Output third-order intercept
`Inf` (default) | scalar

Output third-order intercept, specified as a scalar in dBm

Example: `'OIP3',10`

Example: `rfel.OIP3 = 10`

`Zin` — Input impedance
`50` (default) | positive real part finite scalar

Input impedance, specified as a positive real part finite scalar in Ohms. You can also use a complex value with a positive real part.

Example: `'Zin',40`

Example: `rfel.Zin = 40`

`Zout` — Output impedance
`50` (default) | positive real part finite scalar

Output impedance, specified as a scalar in Ohms. You can also use a complex value with a positive real part.

Example: `'Zout',40`

Example: `rfel.Zout = 40`

`NumPorts` — Number of ports
2 (default) | scalar integer

Number of ports, specified as a scalar integer.

Example: `'NumPorts',2`

Example: `rfel.NumPorts = 2`

`'Terminals'` — Names of port terminals
`{'p1+' 'p2+' 'p1-' 'p2-'}` (default) | cell vector

Names of port terminals, specified as a cell vector These names are always `p` and `n` for positive and negative nodes.

Example: `'Terminals',{'p1+' 'p2+' 'p1-' 'p2-'}`

Examples

RF Element

Create an rfelement object with a gain of 10 dB, noise figure of 3 dB, and OIP3 (output third-order intercept) of 2 dBm.

```
r = rfelement('Gain',10,'NF',3,'OIP3',2)
```

```
r =
  rfelement: RF element

         Name: 'RFelement'
         Gain: 10
           NF: 3
         OIP3: 2
          Zin: 50
         Zout: 50
     NumPorts: 2
    Terminals: {'p1+'  'p2+'  'p1-'  'p2-'}
```

RF Element Circuit

Create an rf element with a gain of 4 dB. Create another rf element with an output third-order intercept(OIP3) of 3 dBm.

```
rfel1 = rfelement('Gain',4);
rfel2 = rfelement('OIP3',13);
```

Build a 2-port circuit using the above defined rf elements.

```
c = circuit([rfel1 rfel2])
```

```
c = 
  circuit: Circuit element

    ElementNames: {'RFelement'  'RFelement_1'}
           Nodes: [0 1 2 3]
            Name: 'unnamed'
        NumPorts: 2
       Terminals: {'p1+'  'p2+'  'p1-'  'p2-'}
```

RF Budget Analysis of Series of RF Elements

Create an amplifier with a gain of 4 dB.

```
a = amplifier('Gain',4);
```

Create a modulator with an OIP3 of 13 dBm.

```
m = modulator('OIP3',13);
```

Create an nport using `passive.s2p`.

```
n = nport('passive.s2p');
```

Create an rf element with a gain of 10 dB.

```
r = rfelement('Gain',10);
```

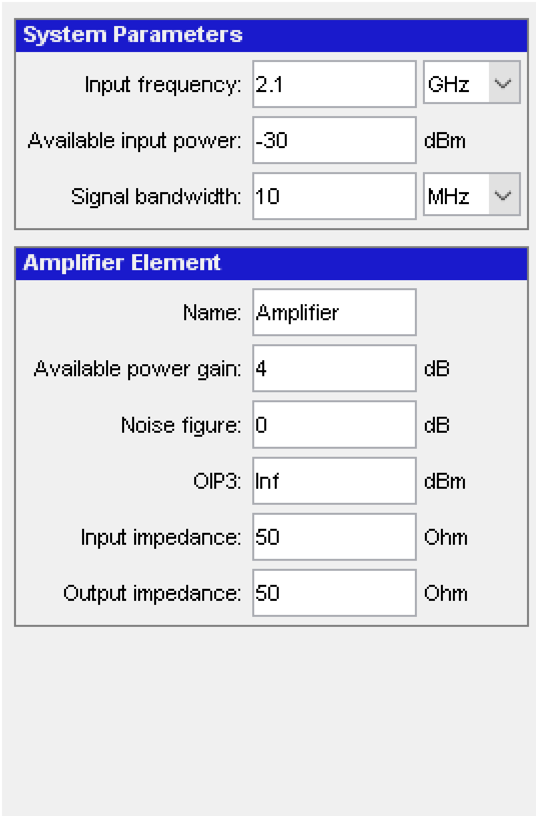
Calculate the rf budget of a series of rf elements at an input frequency of 2.1 GHz, an available input power of -30 dB, and a bandwidth of 10 MHz.

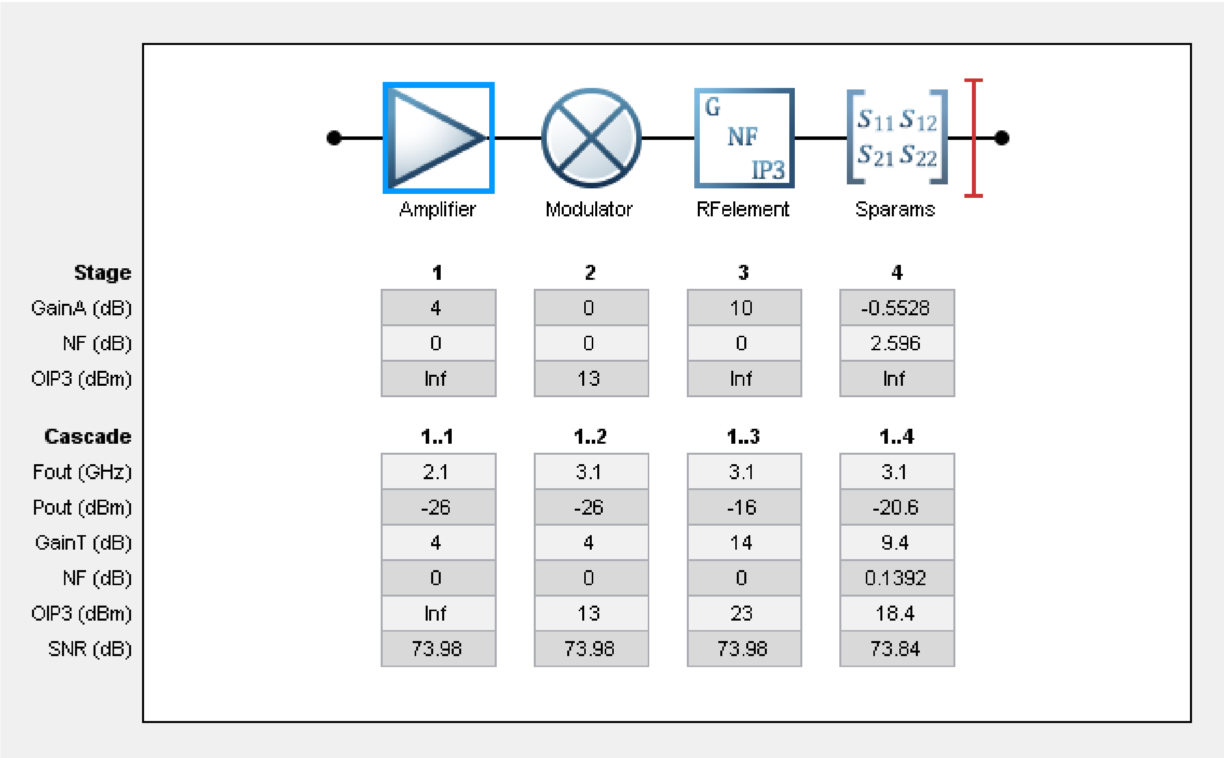
```
b = rfbudget([a m r n],2.1e9,-30,10e6)
```

```
b =
  rfbudget with properties:

               Elements: [1x4 rf.internal.rfbudget.Element]
         InputFrequency: 2.1 GHz
    AvailableInputPower: -30 dBm
        SignalBandwidth:  10 MHz
             AutoUpdate: true

   Analysis Results
        OutputFrequency: (GHz) [  2.1    3.1    3.1     3.1]
            OutputPower: (dBm) [  -26    -26    -16   -20.6]
         TransducerGain: (dB)  [    4      4     14     9.4]
                     NF: (dB)  [    0      0      0  0.1392]
                   OIP3: (dBm) [  Inf     13     23    18.4]
                   IIP3: (dBm) [  Inf      9      9       9]
                    SNR: (dB)  [73.98  73.98  73.98   73.84]
```

Show the analysis in the RF Budget Analyzer app.

```
show(b)
```


System Parameters
Input frequency:
2.1
GHz
Available input power:
-30
dBm
Signal bandwidth:
10
MHz
Amplifier Element
Name:
Amplifier
Available power gain:
4
dB
Noise figure:
0
dB
OIP3:
Inf
dBm
Input impedance:
50
Ohm
Output impedance:
50
Ohm



See Also

`amplifier` | `modulator` | `nport` | `rfbudget`

Introduced in R2017a

OpenIF

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

Description

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

Creation

Syntax

```
hif = OpenIF
hif = OpenIF(Name,Value)
hif = OpenIF(bandwidth)
hif = OpenIF(bandwidth,Name,Value)
```

Description

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

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

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

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

Input Arguments

`bandwidth` — Bandwidth of IF signal
real positive scalar

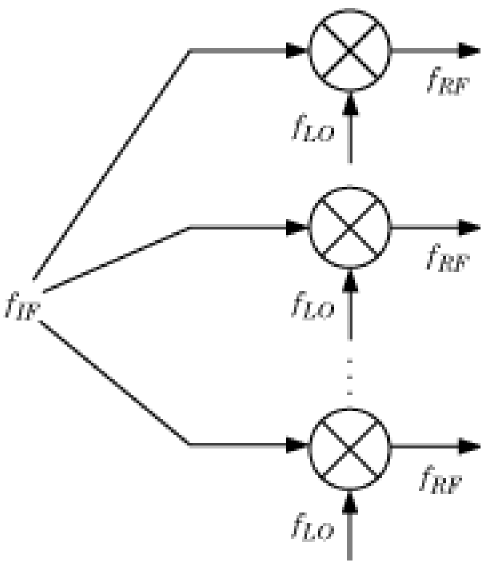
Bandwidth of IF signal, specified as a real positive scalar. The value you provide sets the `IFBW` property of your object.

Properties

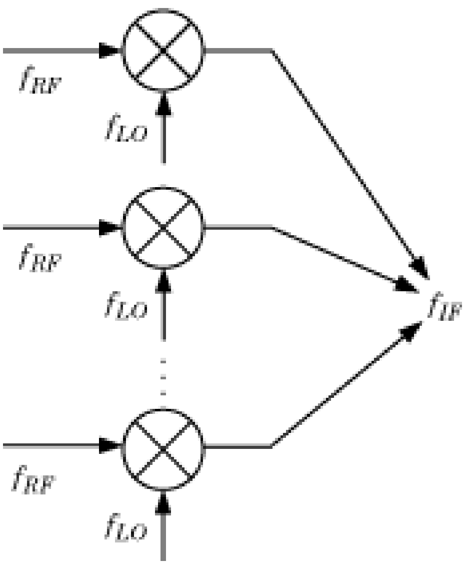
`IF Location` — Location of IF
`'MixerOutput'` (default) | `'MixerInput'`

Location of IF, specified as a `'MixerOutput'` or `'MixerInput'`.

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



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



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

Example: `'IFLocation','MixerInput'`

Example: `amplifier.IFLocation = 'MixerInput'`

`SpurFloor` — Maximum spur value

`99` (default) | scalar

Maximum difference in magnitude between a signal at 0 dBc and an intermodulation product that the `OpenIF` object considers a spur, specified as a scalar in dBc.

Example: `'SpurFloor',80`

Example: `amplifier.SpurFloor = 80`

`IFBW` — System wide IF bandwidth

`99` (default) | scalar

System wide IF bandwidth, specified as a scalar in hertz. You can also set this property using the optional `bandwidth` input argument.

Example: `'SpurFloor',80`

Example: `amplifier.SpurFloor = 80`

Examples

Spur-free zones of a multiband receiver

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

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

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

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

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

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

```
IMT2 = [99 00 09 12 15; ...
        20 00 26 31 48; ...
        55 70 51 70 53; ...
        85 90 60 70 94; ...
        96 95 94 93 92];
addMixer(hif,IMT2,3.7e9,150e6,'low');
```

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

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

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

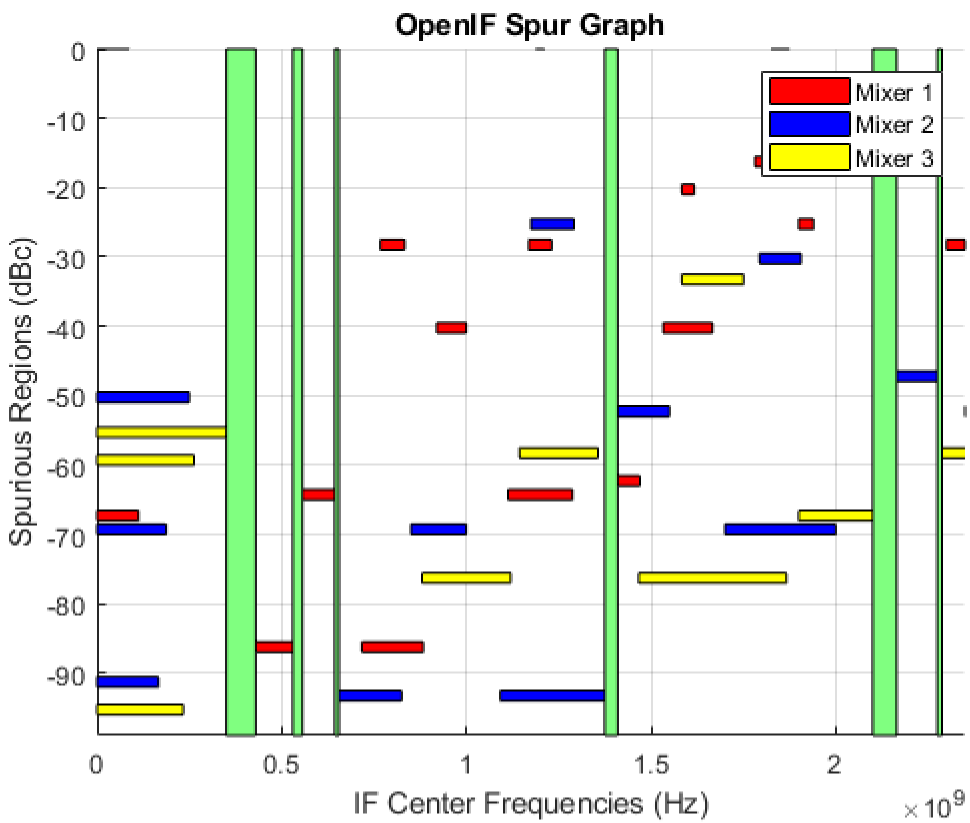
```
hif.report
```

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

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

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

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

     Spur-Free Zones:
      350.00 -  430.00 MHz
      530.00 -  556.25 MHz
```

```
643.75 -   655.00 MHz
  1.38 -     1.41 GHz
  2.10 -     2.17 GHz
  2.28 -     2.29 GHz
```

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

```
figure;
hif.show
```



See Also

Introduced before R2006a

Methods — Alphabetical List

addMixer

Add an additional mixer/RF specification

Syntax

```
addMixer(hif,newimt,newrfcf,newrfbw,newmixtype,newifbw)
```

Description

`addMixer(hif,newimt,newrfcf,newrfbw,newmixtype,newifbw)` adds a mixer to a multiband transmitter or receiver object `hif` as part of an intermediate-frequency (IF) planning analysis workflow.

Examples

Add Two Mixers to System

Set up the object

```
h = OpenIF('IFLocation','MixerOutput');
```

Add two mixers to the system

```
IMT1 = [99 0 21 17 26; 11 0 29 29 63; ...
        60 48 70 65 41; 90 89 74 68 87; 99 99 95 99 99];
addMixer(h,IMT1,2400e6,100e6,'low',50e6)

IMT2 = [99 0 9 12 15; 20 0 26 31 48; ...
```

```
        55 70 51 70 53; 85 90 60 70 94; 96 95 94 93 92];
addMixer(h,IMT2,3700e6,150e6,'high',50e6)
```

Input Arguments

`hif` — OpenIF object
object handle

OpenIF object, specified as an object handle,

`newimt` — Intermodulation table
matrix

Intermodulation table, specified as a matrix of size 2-by-2 or greater with each element unit in dB. Values in the matrix are intermodulation levels. Positive values represent greater attenuation.

Columns of the matrix represent integer multiples of the local oscillator (LO) of the mixer, where column one is `0*LO`, column 2 is `1*LO`, etc. Rows of the matrix represent multipliers for the input frequency to the mixer.

Example: `[99 0 21 17 26; 11 0 29 29 63; ... 60 48 70 65 41; 90 89 74 68 87; 99 99 95 99 99];`

Data Types: `double`

`newrfcf` — RF center frequency
scalar

RF center frequency, specified as a scalar in Hz.

Example: `2400e6`

Data Types: `double`

`newrfbw` — RF bandwidth
scalar

RF bandwidth, specified as a scalar in Hz.

Example: `100e6`

Data Types: `double`

`newifbw` — IF bandwidth
scalar

IF bandwidth, specified as a scalar in Hz.

Example: `50e6`

Data Types: `double`

`newmixtype` — Mixer type
`'sum'` | `'diff'` | `'low'` | `'high'`

Mixer type, specified as `'sum'`, `'diff'`, `'low'`, `'high'`. If the `IFLocation` property in `OpenIF` object is set to `'MixerInput'`, then the mixer type is `'sum'` or `'diff'`. If the `IFLocation` property in `OpenIF` object is set to `'MixerOutput'`, then the mixer type is `'low'` or `'high'`

Example: `50e6`

Data Types: `char`

See Also

Introduced in R2011b

analyze

Analyze circuit object in frequency domain

Syntax

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

Description

`analyze(h,freq)` calculates the following circuit data at the specified frequency values:

- Circuit network parameters
- Noise figure
- Output third-order intercept point
- Power data
- Phase noise
- Voltage standing-wave ratio
- Power gain
- Group delay
- Reflection coefficients
- Stability data
- Transfer function

`h` is the handle of the circuit object to be analyzed. `freq` is a vector of frequencies, specified in hertz, at which to analyze the circuit. OIP_3 is always infinite for passive circuits.

`analyze(h,freq,zl,zs,zo,aperture)` calculates the circuit data at the specified frequency values. The arguments `zl`, `zs`, `zo`, and `aperture` are optional. `zl`, `zs`, and `zo` represent the circuit load, circuit source, and reference impedances of the S-parameters, respectively. The default value of all these arguments is 50 ohms.

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

The `aperture` argument determines the two frequency points that the `analyze` method uses to compute the group delay for each frequency in `freq`. `aperture` can be a positive scalar or a vector of the same length of as `freq`.

Note For `rfckt.datafile`, `rfckt.passive`, `rfckt.amplifier`, and `rfckt.mixer` objects that contain measured S-parameter data, the `analyze` method uses the two nearest measurement points to compute the group delay, regardless of the value of `aperture`.

Group delay τ_g at each frequency point f is the negative slope of the phase angle of S_{21} with respect to f:

$$\tau_g(f) = -\frac{\Delta\phi}{\Delta\omega} = -\frac{\arg\left(S_{21}(f_+)\right) - \arg\left(S_{21}(f_-)\right)}{2\pi\left(f_+ - f_-\right)}$$

where:

- f_+ is:
  - f(1 + `aperture`/2) for `aperture` < 1.
  - f + `aperture`/2 for `aperture` ≥ 1.

  If f is the maximum value of `freq`, then $f_+ = f$.
- f_- is:
  - f (1 – `aperture`/2) for `aperture` < 1.
  - f – `aperture`/2 for `aperture` ≥ 1.

  If f is the minimum value of `freq`, then $f_- = f$.

By default, `analyze` calculates the group delay in nanoseconds.

The value of `aperture` affects the accuracy of the computed group delay. If `aperture` is too large, the slope estimate may be not accurate. If `aperture` is too small, the computer numerical error may affect the accuracy of the group delay result.

`analyze(h,freq,'condition1',value1,...,'conditionm',valuem)` calculates the circuit data at the specified frequency values and operating conditions for the object `h`. The inputs `'condition1',value1,...,'conditionm',valuem` are the condition/value pairs at which to analyze the object. Use this syntax for `rfckt.amplifier`, `rfckt.mixer`, and `rfdata.data` objects where the condition/value pairs are operating conditions from a `.p2d` or `.s2d` file.

Note When you specify condition/value pairs, the `analyze` method changes the object's values to match your specification.

When you analyze a network that contains several objects, RF Toolbox software does not issue an error or warning if the specified conditions cannot be applied to all objects. For some networks, because there is no error or warning, you can call the `analyze` method once to apply the same set of operating conditions to any objects where operating conditions are applicable. However, you may want to analyze a network that contains one or more of the following:

- Several objects with different sets of operating conditions.
- Several objects with the same set of operating conditions that are configured differently.

To analyze such a network, you should use the `setop` method to configure the operating conditions of each individual object before analyzing the network.

Analysis of Circuit Objects

For most circuit objects, the `AnalyzedResult` property is empty until the `analyze` method is applied to the circuit object. However, the following four circuit objects are the exception to this rule:

- `rfckt.datafile` — By default, the `AnalyzedResult` property of `rfckt.datafile` objects contains the S-parameter, noise figure, and group delay values that are calculated over the network parameter frequencies in the `passive.s2p` data file. OIP3 is ∞ by default because the data in `passive.s2p` is passive.
- `rfckt.passive` — By default, the `AnalyzedResult` property of `rfckt.passive` objects contains the S-parameter, noise figure, and group delay values that are the result of analyzing the values stored in the `passive.s2p` file at the frequencies stored in this file. These frequency values are also stored in the `NetworkData` property. OIP3 is always ∞ for `rfckt.passive` objects because the data is passive.

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

For a detailed explanation of how the `analyze` method calculates the network parameters, noise figure values, and OIP3 values for a particular object, see the `AnalyzedResult` property on the reference page for that object.

Examples

Analyze Network Object

Create and analyze a two-wire network object.

```
tx1=rfckt.twowire('Radius',7.5e-4);
analyze(tx1,1.9e9)
```

```
ans =
   rfckt.twowire with properties:

            Radius: 7.5000e-04
        Separation: 0.0016
               MuR: 1
          EpsilonR: 2.3000
       LossTangent: 0
         SigmaCond: Inf
        LineLength: 0.0100
          StubMode: 'NotAStub'
       Termination: 'NotApplicable'
             nPort: 2
    AnalyzedResult: [1x1 rfdata.data]
              Name: 'Two-Wire Transmission Line'
```

References

See Also

calculate | extract | getz0 | listformat | listparam | loglog | plot | plotyy | polar | read | restore | semilogx | semilogy | smith | write

Introduced before R2006a

calculate

Calculate specified parameters for circuit object

Syntax

```
[data,params,freq]=calculate(h,'parameter1',..., ...'parametern','fo
rmat')
[ydata,params,xdata]=calculate(h,'parameter1',...,
'parametern','format',xparameter,xformat,'condition1',value1,...,
'conditionm',valuem, 'freq',freq,'pin',pin)
```

Description

`[data,params,freq]=calculate(h,'parameter1',..., ...'parametern','format')` calculates the specified parameters for the object `h` and returns them in the `n`-element cell array `data`.

The input `h` is the handle of a circuit object.

`parameter1`, `...`, `parametern` is the list of parameters to be calculated. Use the `listparam` method to get a list of the valid parameters for a circuit object.

`format` is the format of the output `data`. The format determines if RF Toolbox software converts the parameter values to a new set of units, or operates on the components of complex parameter values.

For example:

- Specify `format` as `Real` to compute the real part of the selected parameter.
- Specify `format` as `'none'` to return the parameter values unchanged.

Use the `listformat` method to get a list of the valid formats for a particular parameter.

The output `params` is an n-element cell array containing the names of the parameters in `data`. `freq` is a vector of frequencies at which the parameters are known.

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

`[ydata,params,xdata]=calculate(h,'parameter1',..., 'parametern','format',xparameter,xformat,'condition1',value1,..., 'conditionm',valuem, 'freq',freq,'pin',pin)` calculates the specified parameters at the specified operating conditions for the object `h`.

`xparameter` is the independent parameter for which to calculate the specified parameters. Several `xparameter` values are available for all objects. When you import `rfckt.amplifier`, `rfckt.mixer`, or `rfdata.data` object specifications from a `.p2d` or `.s2d` file, you can also specify any operating conditions from the file that have numeric values, such as bias.

The following table shows the most commonly available parameters and the corresponding `xparameter` values. The default settings listed in the table are used if `xparameter` is not specified.

Parameter Name	xparameter values
Pout, Phase, LS11, LS12, LS21, LS22	Pin (default), Freq
S11, S12, S21, S22, NF, IIP3, OIP3, GroupDelay, VSWRIn, VSWROut, GammaIn, GammaOut, FMIN, GammaOPT, RN, TF1, TF2, Gt, Ga, Gp, Gmag, Gmsg, GammaMS, GammaML, K, Delta, Mu, MuPrime	Freq
AM/AM, AM/PM	AM (default, and only available value)

`xformat` is the format to use for the specified `xparameter`. No `xformat` specification is needed when `xparameter` is an operating condition.

The following table shows the `xformat` values that are available for the `xparameter` values listed in the preceding table, along with the default settings that are used if `xformat` is not specified.

xparameter values	xformat values
Pin	dBm (default), mW, W, dBW

xparameter values	xformat values
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 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

Calculate S-Parameters of Transmission Line

Analyze a general transmission line of impedance, 50 ohms, phase velocity of 299792458 m/s, and line length of 0.01 meters for frequencies 1.0 GHz to 3.0 GHz.

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

```
ans =
   rfckt.txline with properties:

        LineLength: 0.0100
          StubMode: 'NotAStub'
       Termination: 'NotApplicable'
              Freq: 1.0000e+09
                Z0: 50.0000 + 0.0000i
                PV: 299792458
              Loss: 0
          IntpType: 'Linear'
             nPort: 2
    AnalyzedResult: [1x1 rfdata.data]
              Name: 'Transmission Line'
```

Calculate the S11 and S22 parameters in dB.

```
[data,params,freq] = calculate(trl,'S11','S22','dB')
```

```
data = 1x2 cell array
    {201x1 double}    {201x1 double}


params = 1x2 cell array
    {'S_{11}'}    {'S_{22}'}


freq = 201×1
10⁹ ×

    1.0000
    1.0100
    1.0200
    1.0300
    1.0400
    1.0500
    1.0600
    1.0700
    1.0800
    1.0900
      ⋮
```

See Also

analyze | extract | getz0 | listformat | listparam | loglog | plot | plotyy | polar | read | restore | semilogx | semilogy | smith | write

Introduced before R2006a

extract

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 values: `'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.

Examples

Extract Network Parameters

Extract ABCD-parameters for an rfckt.amplifier object read from `default.s2p`.

```
amp = read(rfckt.amplifier,'default.s2p');
[outmatrix,freq] = extract(amp,'ABCD_parameters');
```

See Also

`analyze` | `calculate` | `getz0` | `listformat` | `listparam` | `loglog` | `plot` | `plotyy` | `polar` | `read` | `restore` | `semilogx` | `semilogy` | `smith` | `write`

Introduced before R2006a

freqresp

Frequency response of rational function object

Syntax

```
[resp,outfreq] = freqresp(h,infreq)
```

Description

`[resp,outfreq] = freqresp(h,infreq)` computes the frequency response, `resp`, of the rational function object, `h`, at the frequencies specified by `freq`.

The input `h` is the handle of a rational function object returned by `rationalfit`, and `infreq` is a vector of positive frequencies, in Hz, over which the frequency response is calculated.

The output argument `outfreq` is a vector that contains the same frequencies as the input frequency vector, in order of increasing frequency. The frequency response, `resp`, is a vector of frequency response values corresponding to these frequencies. It is computed using the analytical form of the rational function

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

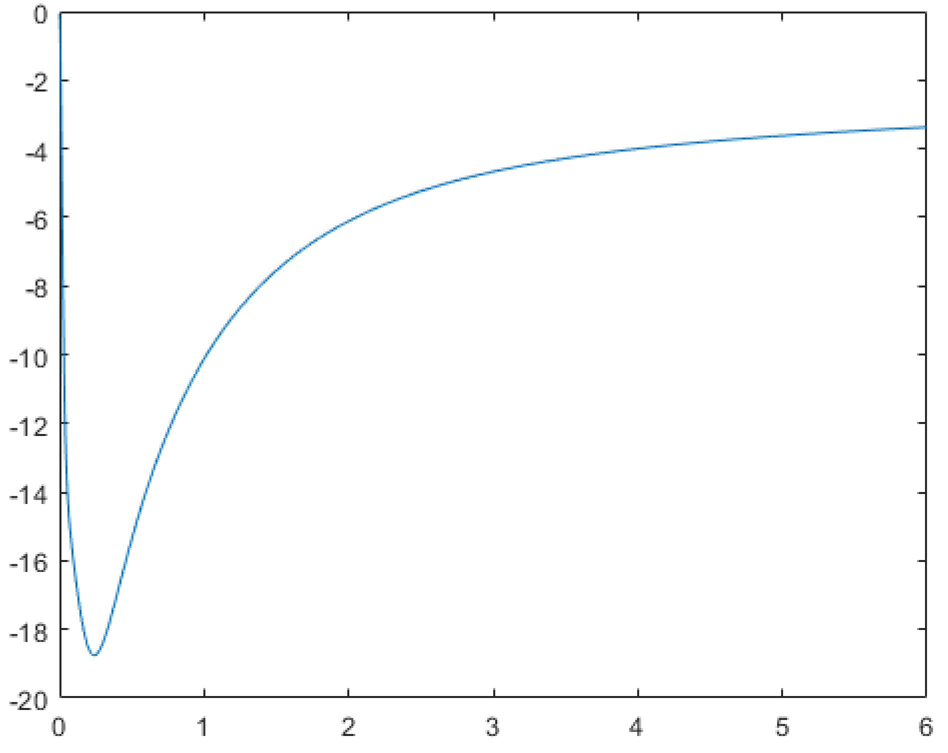
where `A`, `C`, `D`, and `Delay` are properties of the rational function object, `h`.

Examples

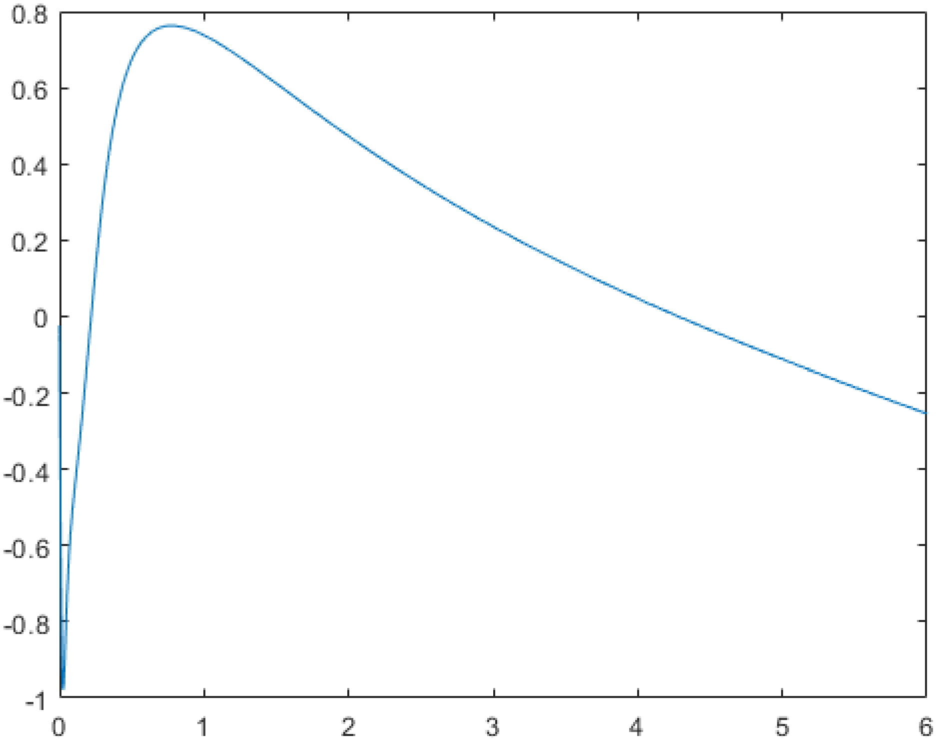
Frequency Response of Data Stored In File

Compute the frequency response of data stored in the file,|passive.s2p| by reading it into an `rfdata` object, fitting a rational function object to the data, and using the `freqresp` method to compute the frequency response of the object

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

orig_data =
   rfdata.data with properties:

            Freq: [202x1 double]
    S_Parameters: [2x2x202 double]
      GroupDelay: [202x1 double]
              NF: [202x1 double]
            OIP3: [202x1 double]
              Z0: 50.0000 + 0.0000i
              ZS: 50.0000 + 0.0000i
              ZL: 50.0000 + 0.0000i
        IntpType: 'Linear'
            Name: 'Data object'


freq=orig_data.Freq;
data=orig_data.S_Parameters(2,1,:);
fit_data=rationalfit(freq,data)

fit_data =
   rfmodel.rational with properties:

        A: [6x1 double]
        C: [6x1 double]
        D: 0
    Delay: 0
     Name: 'Rational Function'



[resp,freq]=freqresp(fit_data,freq);

plot(freq/1e9,20*log10(abs(resp)));
```




```
figure
plot(freq/1e9,unwrap(angle(resp)));
```



See Also

`rationalfit` | `rfmodel.rational` | `timeresp` | `writeva`

Introduced in R2006b

getop

Display operating conditions

Syntax

`getop(h)`

Description

`getop(h)` displays the selected operating conditions for the circuit or data object, `h`.

Information about operating conditions is available only when you import the object specifications from a `.p2d` or `.s2d` file.

Examples

Display Operating Conditions

Display the operating conditions of a circuit.

```
ckt1 = read(rfckt.amplifier, 'default.p2d');
getop(ckt1)
```

```
ans = 1x2 cell array
    {'Bias'}    {'1.5'}
```

See Also

`setop`

Introduced in R2007a

getz0

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`.

Examples

Get Z0 of Network Object

Create and analyze a two-wire network object.

```
tx1=rfckt.twowire('Radius',7.5e-4)
```

```
tx1 =
   rfckt.twowire with properties:

            Radius: 7.5000e-04
        Separation: 0.0016
               MuR: 1
          EpsilonR: 2.3000
       LossTangent: 0
         SigmaCond: Inf
        LineLength: 0.0100
          StubMode: 'NotAStub'
       Termination: 'NotApplicable'
             nPort: 2
```

```
    AnalyzedResult: []
              Name: 'Two-Wire Transmission Line'
```

```
analyze(tx1,1.9e9)
```

```
ans = 
   rfckt.twowire with properties:

            Radius: 7.5000e-04
        Separation: 0.0016
               MuR: 1
          EpsilonR: 2.3000
       LossTangent: 0
         SigmaCond: Inf
        LineLength: 0.0100
          StubMode: 'NotAStub'
       Termination: 'NotApplicable'
             nPort: 2
    AnalyzedResult: [1x1 rfdata.data]
              Name: 'Two-Wire Transmission Line'
```

Find the Z0 of the two-wire object.

```
z0 = getz0(tx1)
```

```
z0 = 31.4212
```

See Also

analyze | calculate | extract | listformat | listparam | loglog | plot | plotyy | polar | read | restore | semilogx | semilogy | smith | write

Introduced before R2006a

impulse

Impulse response for rational function object

Note `impulse` may be removed in a future release. Use `timeresp` instead.

Syntax

```
[resp,t] = impulse(h,ts,n)
```

Description

`[resp,t] = impulse(h,ts,n)` computes the impulse response, `resp`, of the rational function object, `h`, over the time period specified by `ts` and `n`.

Note While you can compute the output response for a rational function 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 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 rational function object, `h`.
- `M` is the number of poles in the rational function object.

Examples

Impulse Response of Data Stored In File

Compute the impulse response of the data stored in the file passive.s2p by

fitting a rational function object to the data and using the impulse method

to compute the impulse response of the object.

Section 1 Extract frequency and data from passive.s2p

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

```
orig_data =
   rfdata.data with properties:

            Freq: [202x1 double]
    S_Parameters: [2x2x202 double]
      GroupDelay: [202x1 double]
              NF: [202x1 double]
            OIP3: [202x1 double]
              Z0: 50.0000 + 0.0000i
              ZS: 50.0000 + 0.0000i
              ZL: 50.0000 + 0.0000i
        IntpType: 'Linear'
            Name: 'Data object'
```

```
freq=orig_data.Freq;
data=orig_data.S_Parameters(2,1,:);
```

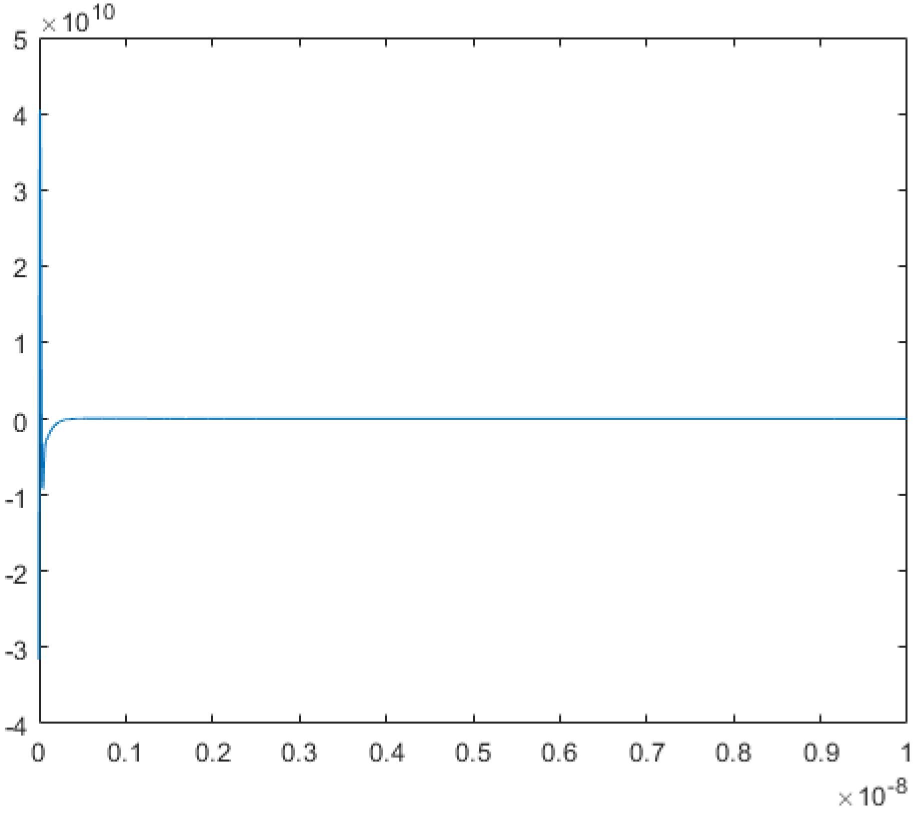
Section 2 Rational fit the data

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

```
fit_data =
   rfmodel.rational with properties:

        A: [6x1 double]
        C: [6x1 double]
        D: 0
    Delay: 0
     Name: 'Rational Function'
```

Section 3 Calculate the Impulse Response

```
[resp,t]=impulse(fit_data,1e-12,1e4);

plot(t,resp);
```



See Also

`freqresp` | `rationalfit` | `rfmodel.rational` | `writeva`

Introduced in R2006b

ispassive

Check passivity of scalar rational function object

Syntax

```
result = ispassive(h)
```

Description

`result = ispassive(h)` checks the passivity of the rational function object, `h`, across all frequencies, and returns `result`, a logical value. If `h` is passive, then `result` is `1`. If `h` is not passive, then `result` is `0`.

Examples

Check Passivity of Object

Create a scalar rational function object and check the passivity of the object. Read a Touchstone data file.

```
ckt = read(rfckt.passive, 'passive.s2p');
```

Fit the transfer function into a rational function object.

```
TF = s2tf(ckt.AnalyzedResult.S_Parameters);
TF_Object = rationalfit(ckt.AnalyzedResult.Freq, TF);
```

Check the passivity of the rational function object.

```
Is_Passive = ispassive(TF_Object)
```

```
Is_Passive = logical
   1
```

See Also

`rationalfit` | `rfmodel.rational`

Introduced in R2010a

listformat

List valid formats for specified circuit object parameter

Syntax

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

Description

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

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

When you plot phase information as a function of frequency, RF Toolbox software unwraps the phase data using the MATLAB `unwrap` function. The resulting plot is only meaningful if the phase data varies smoothly as a function of frequency, as described in the `unwrap` reference page. If your data does not meet this requirement, you must obtain data on a finer frequency grid.

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

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

Examples

List Format of Network Parameter

List the available formats of analysis of a transmission line.

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

```
ans = 11x1 cell array
    {'dB'                   }
    {'Magnitude (decibels)'}
    {'Abs'                  }
    {'Mag'                  }
    {'Magnitude (linear)'  }
    {'Angle'                }
    {'Angle (degrees)'     }
    {'Angle (radians)'     }
    {'Real'                 }
    {'Imag'                 }
    {'Imaginary'            }
```

See Also

analyze | calculate | extract | getz0 | listparam | loglog | plot | plotyy | polar | read | restore | semilogx | semilogy | smith | write

Introduced before R2006a

listparam

List valid parameters for specified circuit object

Syntax

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

Description

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

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

Several parameters are available for all objects. When you import `rfckt.amplifier`, `rfckt.mixer`, or `rfdata.data` object specifications from a `.p2d` or `.s2d` file, the list of valid parameters also includes any operating conditions from the file that have numeric values, such as bias.

The following table describes the most commonly available parameters.

Parameter	Description
`S11`, `S12`, `S21`, `S22` `LS11`, `LS12`, `LS21`, `LS22` (Amplifier and mixer objects with multiple operating conditions only)	S-parameters
`GroupDelay`	Group delay
`GammaIn`, `GammaOut`	Input and output reflection coefficients
`VSWRIn`, `VSWROut`	Input and output voltage standing-wave ratio

Parameter	Description
IIP3, OIP3 (Amplifier and mixer objects only)	Third-order intercept point
NF	Noise figure
TF1	Ratio of the load voltage to the output voltage of the source when the input port is conjugate matched
TF2	Ratio of load voltage to the source voltage
• Gt • Ga • Gp • Gmag • Gmsg	• Transducer power gain • Available power gain • Operating power gain • Maximum available power gain • Maximum stable gain
GammaMS, GammaML	Source and load reflection coefficients for simultaneous conjugate match
K, Mu, MuPrime	Stability factor
Delta	Stability condition

Examples

List Parameters of Network Object

List the available parameters of analysis of a transmission line.

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

```
ans = 28x1 cell array
    {'S11'       }
    {'S12'       }
    {'S21'       }
    {'S22'       }
    {'GroupDelay'}
```

```
{'GammaIn'   }
{'GammaOut'  }
{'VSWRIn'    }
{'VSWROut'   }
{'OIP3'      }
{'IIP3'      }
{'NF'        }
{'NFactor'   }
{'NTemp'     }
{'TF1'       }
{'TF2'       }
{'TF3'       }
{'Gt'        }
{'Ga'        }
{'Gp'        }
{'Gmag'      }
{'Gmsg'      }
{'GammaMS'   }
{'GammaML'   }
{'K'         }
{'Delta'     }
{'Mu'        }
{'MuPrime'   }
```

See Also

`analyze` | `calculate` | `extract` | `getz0` | `listformat` | `loglog` | `plot` | `plotyy` | `polar` | `read` | `restore` | `semilogx` | `semilogy` | `smith` | `write`

Introduced before R2006a

loglog

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 Chart Line. The reference pages for MATLAB functions such as `figure`, `axes`, and `text` also list available properties and provide links to more complete property descriptions.

Note Use the MATLAB `loglog` function to create a log-log scale plot of parameters that are specified as vector data and are not part of a circuit (`rfckt`) object or data (`rfdata`) object.

`lineseries=loglog(h,'parameter1',...,'parametern', format, xparameter,xformat,'condition1',value1,..., 'conditionm',valuem,'freq',freq,'pin',pin)` plots the specified parameters at the specified operating conditions for the object `h`.

`xparameter` is the independent variable to use in plotting the specified parameters. Several `xparameter` values are available for all objects. When you import `rfckt.amplifier`, `rfckt.mixer`, or `rfdata.data` object specifications from a `.p2d` or `.s2d` file, you can also specify any operating conditions from the file that have numeric values, such as bias.

The following table shows the most commonly available parameters and the corresponding `xparameter` values. The default settings listed in the table are used if `xparameter` is not specified.

Parameter Name	`xparameter` values
`Pout`, `Phase`, `LS11`, `LS12`, `LS21`, `LS22`	`Pin` (default), `Freq`
`S11`, `S12`, `S21`, `S22`, `NF`, `IIP3`, `OIP3`, `GroupDelay`, `VSWRIn`, `VSWROut`, `GammaIn`, `GammaOut`, `FMIN`, `GammaOPT`, `RN`, `TF1`, `TF2`, `Gt`, `Ga`, `Gp`, `Gmag`, `Gmsg`, `GammaMS`, `GammaML`, `K`, `Delta`, `Mu`, `MuPrime`	`Freq`
`AM/AM`, `AM/PM`	`AM`

`xformat` is the format to use for the specified `xparameter`. No `xformat` specification is needed when `xparameter` is an operating condition.

The following table shows the `xformat` values that are available for the `xparameter` values listed in the preceding table, along with the default settings that are used if `xformat` is not specified.

`xparameter` values	`xformat` values
`Pin`	`dBm` (default), `mW`, `W`, `dBW`
`Freq`	`THz`, `GHz`, `MHz`, `KHz`, `Hz` By default, `xformat` is chosen to provide the best scaling for the given `xparameter` values.
`AM`	`Magnitude (decibels)` (default), `Magnitude (linear)`

`condition1,value1`, `...`, `conditionm,valuem` are the optional condition/value pairs at which to plot the specified parameters. These pairs are usually operating conditions from a `.p2d` or `.s2d` file. For some parameters, you can specify a set of frequency or input power values at which to plot the specified parameter.

For example:

- When plotting large-signal S-parameters as a function of input power, you can specify frequency points of interest using condition/value pairs.
- When plotting large-signal S-parameters as a function of frequency, you can specify input power levels of interest using condition/value pairs.
- When plotting parameters as a function of an operating condition, you can specify both frequency and input power values using condition/value pairs.

`freq` is the optional frequency value, in hertz, at which to plot the specified parameters.

`pin` is the optional input power value, in dBm, at which to plot the specified parameters.

If `h` has multiple operating conditions, such as from a `.p2d` or `.s2d` file, the `loglog` method operates as follows:

- If you do not specify any operating conditions as arguments to the `loglog` method, then the method plots the parameter values based on the currently selected operating condition.
- If you specify one or more operating conditions, the `loglog` method plots the parameter values based on those operating conditions.
- When you use an operating condition for the `xparameter` input argument, the method plots the parameters for all operating condition values.

Examples

Plot Circuit Parameters Network Object Using Log-Log Scale

Create and analyze a two-wire network object.

```
tx1=rfckt.twowire('Radius',7.5e-4)
```

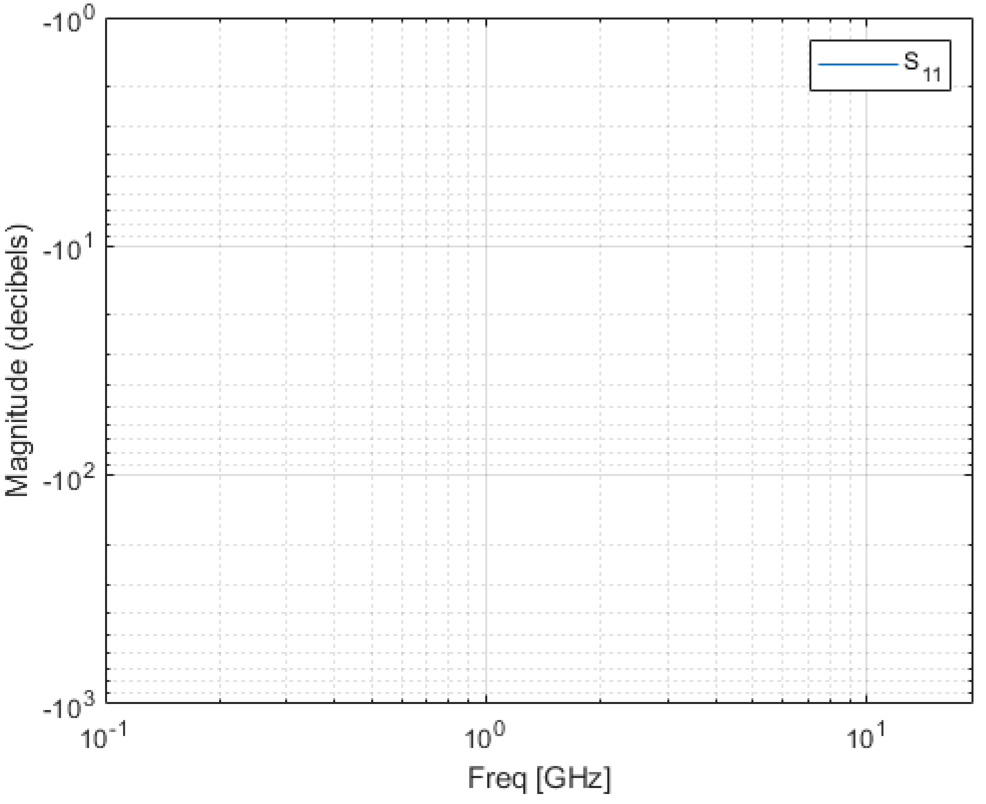
```
tx1 = 
   rfckt.twowire with properties:

             Radius: 7.5000e-04
         Separation: 0.0016
                MuR: 1
           EpsilonR: 2.3000
        LossTangent: 0
          SigmaCond: Inf
         LineLength: 0.0100
           StubMode: 'NotAStub'
        Termination: 'NotApplicable'
              nPort: 2
     AnalyzedResult: []
               Name: 'Two-Wire Transmission Line'
```

```
analyze(tx1,1.9e9)
```

```
ans = 
   rfckt.twowire with properties:

             Radius: 7.5000e-04
         Separation: 0.0016
                MuR: 1
           EpsilonR: 2.3000
        LossTangent: 0
          SigmaCond: Inf
         LineLength: 0.0100
           StubMode: 'NotAStub'
        Termination: 'NotApplicable'
              nPort: 2
     AnalyzedResult: [1x1 rfdata.data]
               Name: 'Two-Wire Transmission Line'
```

Plot S11 using the log-log scale.

```
linesereis = loglog(tx1,'S11')
```




```
linesereis = 
  Line (S_{11}) with properties:

              Color: [0 0.4470 0.7410]
          LineStyle: '-'
          LineWidth: 0.5000
             Marker: 'none'
         MarkerSize: 6
    MarkerFaceColor: 'none'
              XData: 1.9000
```

```
            YData: -11.5774
            ZData: [1x0 double]

  Show all properties
```

See Also

analyze | calculate | extract | getz0 | listformat | listparam | plot | plotyy | polar | read | restore | semilogx | semilogy | smith | write

Introduced in R2007a

plot

Plot specified circuit object parameters on X-Y plane

Syntax

```
lineseries = plot(h,parameter)
lineseries = plot(h,parameter1,...,parametern)
lineseries = plot(h,parameter1,...,parametern,format)
lineseries=plot(h,'parameter1',...,'parametern',
format ,xparameter,xformat,'condition1',value1,...,
'conditionm',valuem,'freq',freq,'pin',pin)
lineseries = plot(h,'budget',...)
lineseries = plot(h,'mixerspur',k,pin,fin)
```

Description

`lineseries = plot(h,parameter)` plots the specified `parameter` on an X-Y plane in the default format. `h` is the handle of a circuit (`rfckt`) object. Use the `listparam` method to get a list of the valid parameters for a particular circuit object, `h`.

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

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

`lineseries = plot(h,parameter1,...,parametern,format)` plots the specified parameters `parameter1,...`, `parametern` in the specified `format`. The format determines if RF Toolbox software converts the parameter values to a new set of units, or operates on the components of complex parameter values. For example:

- Specify `format` as `Real` to plot the real part of the selected parameter.
- Specify `format` as `'none'` to plot the parameter values unchanged.

Use the `listformat` method to get a list of the valid formats for a particular parameter.

`lineseries=plot(h,'parameter1',...,'parametern',`
`format ,xparameter,xformat,'condition1',value1,...,`
`'conditionm',valuem,'freq',freq,'pin',pin)` plots the specified parameters at the specified operating conditions for the object `h`.

`xparameter` is the independent variable to use in plotting the specified parameters. Several `xparameter` values are available for all objects. When you import `rfckt.amplifier`, `rfckt.mixer`, or `rfdata.data` object specifications from a `.p2d` or `.s2d` file, you can also specify any operating conditions from the file that have numeric values, such as bias.

The following table shows the most commonly available parameters and the corresponding `xparameter` values. The default settings listed in the table are used if `xparameter` is not specified.

Parameter Name	xparameter values
Pout, Phase, LS11, LS12, LS21, LS22	Pin (default), Freq
S_{11}, S_{12}, S_{21}, S_{22}, S_{ij},NF, IIP3, OIP3, GroupDelay, VSWRIn, VSWROut, GammaIn, GammaOut, FMIN, GammaOPT, RN, TF1, TF2, Gt, Ga, Gp, Gmag, Gmsg, GammaMS, GammaML, K, Delta, Mu, MuPrime	Freq
AM/AM, AM/PM	AM

`xformat` is the format to use for the specified `xparameter`. No `xformat` specification is needed when `xparameter` is an operating condition.

The following table shows the `xformat` values that are available for the `xparameter` values listed in the preceding table, along with the default settings that are used if `xformat` is not specified.

xparameter values	xformat values
Pin	dBm (default), mW, W, dBW
Freq	THz, GHz, MHz, KHz, Hz By default, `xformat` is chosen to provide the best scaling for the given `xparameter` values.

xparameter values	xformat 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
S_{11}, S_{12}, S_{21}, S_{22}, S_{ij}	Magnitude (decibels) Magnitude (linear) Angle (degrees) Real Imaginary
OIP3	dBm dBW W mW
NF	Magnitude (decibels) Magnitude (linear)

`lineseries = plot(h,'mixerspur',k,pin,fin)` plots spur power of an `rfckt.mixer` object or an `rfckt.cascade` object that contains one or more mixers.

`k` is the index of the circuit object for which to plot spur power. Its value can be an integer or `'all'`. The default is `'all'`. This value creates a budget plot of the spur power for `h`. Use `0` to plot the power at the input of `h`.

`pin` is the optional scalar input power value, in dBm, at which to plot the spur power. The default is `0` dBm. When you create a spur plot for an object, the previous input power value is used for subsequent plots until you specify a different value.

`fin` is the optional scalar input frequency value, in hertz, at which to plot the spur power. If `h` is an `rfckt.mixer` object, the default value of `fin` is the input frequency at which the magnitude of the S_{21} parameter of the mixer, in decibels, is highest. If `h` is an `rfckt.cascde` object, the default value of `fin` is the input frequency at which the magnitude of the S_{21} parameter of the first mixer in the cascade is highest. When you create a spur plot for an object, the previous input frequency value is used for subsequent plots until you specify a different value.

For more information on plotting mixer spur power, see the Visualizing Mixer Spurs example.

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

Use the Property Editor (`propertyeditor`) or the MATLAB `set` function to change Chart Line. 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.

Examples

Plot Circuit Parameters Network Object on X-Y plane

Create and analyze a two-wire network object.

```
tx1=rfckt.twowire('Radius',7.5e-4)
```

```
tx1 =
   rfckt.twowire with properties:

            Radius: 7.5000e-04
        Separation: 0.0016
               MuR: 1
          EpsilonR: 2.3000
       LossTangent: 0
         SigmaCond: Inf
        LineLength: 0.0100
          StubMode: 'NotAStub'
       Termination: 'NotApplicable'
             nPort: 2
    AnalyzedResult: []
              Name: 'Two-Wire Transmission Line'
```

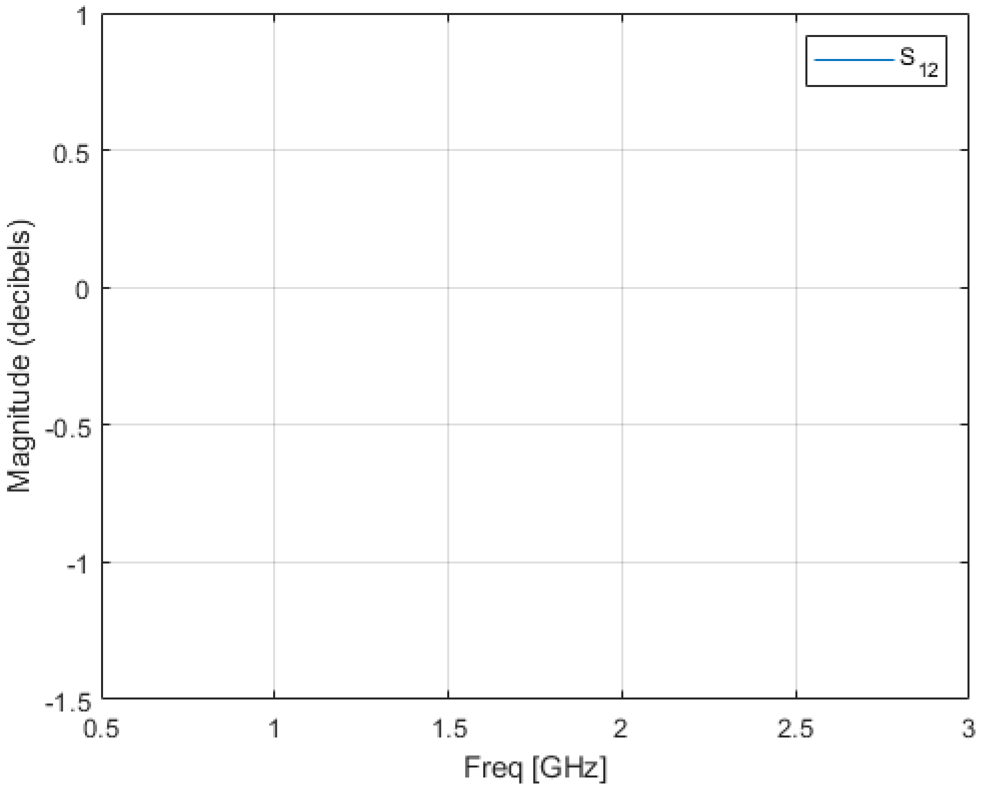
```
analyze(tx1,1.9e9)
```

```
ans =
   rfckt.twowire with properties:

            Radius: 7.5000e-04
        Separation: 0.0016
               MuR: 1
          EpsilonR: 2.3000
```

```
    LossTangent: 0
      SigmaCond: Inf
     LineLength: 0.0100
       StubMode: 'NotAStub'
    Termination: 'NotApplicable'
          nPort: 2
 AnalyzedResult: [1x1 rfdata.data]
           Name: 'Two-Wire Transmission Line'
```

Plot S12 on X-Y plane

```
linesereis = plot(tx1,'S12')
```



```
linesereis =
  Line (S_{12}) with properties:

              Color: [0 0.4470 0.7410]
          LineStyle: '-'
          LineWidth: 0.5000
             Marker: 'none'
         MarkerSize: 6
    MarkerFaceColor: 'none'
              XData: 1.9000
              YData: -0.3130
              ZData: [1x0 double]

  Show all properties
```

Alternatives

The `rfplot` function creates magnitude-frequency plots for RF Toolbox S-parameter objects.

See Also

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

Introduced before R2006a

plotyy

Plot specified object parameters with *y*-axes on both left and right sides

Syntax

```
[ax,hlines1,hlines2] = plotyy(h,parameter)
[ax,hlines1,hlines2] = plotyy(h,parameter1,...,parametern)
[ax,hlines1,hlines2] = plotyy(h,parameter,format1,format2)
[ax,hlines1,hlines2] = plotyy(h, parameter1, ..., parametern,
format1, format2)
[ax,hlines1,hlines2] = plotyy(h,parameter1_1,...,parameter1_n1,
format1,parameter2_1,...,parameter2_n2,format2)
[ax,hlines1,hlines2]=plotyy(h,parameter1_1,...,parameter1_n1,
format1,parameter2_1,...,parameter2_n2,format2,xparameter,
xformat,'condition1',value1,...,'conditionm',valuem,
'freq',freq,'pin',pin)
```

Description

`[ax,hlines1,hlines2] = plotyy(h,parameter)` plots the specified `parameter` using the predefined primary and secondary formats for the left and right *y*-axes, respectively. The formats define how RF Toolbox software displays the data on the plot. `h` is the handle of a circuit (`rfckt`) or an `rfdata.data` object.

- See “Determining Formats” on page 7-51 for a table that shows the predefined primary and secondary formats for the parameters for all circuit and data objects.
- Type `listparam(h)` to get a list of valid parameters for a circuit object, `h`. Type `listformat(h,parameter)` to see the legitimate formats for a specified `parameter`. The first listed format is the default for the specified parameter.

The `plotyy` method returns the handles to the two axes created in `ax` and the handles to two `lineseries` objects in `hlines1` and `hlines2`.

- `ax(1)` is the left axes.

- `ax(2)` is the right axes.
- `hlines1` is the `lineseries` object for the left *y*-axis.
- `hlines2` is the `lineseries` object for the right *y*-axis.

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

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

Note Use the MATLAB `plotyy` function to plot parameters on two *y*-axes that are specified as vector data and are not part of a circuit (`rfckt`) object or data (`rfdata`) object.

`[ax,hlines1,hlines2] = plotyy(h,parameter1,...,parametern)` plots the parameters `parameter1`, `...`, `parametern`. `plotyy` determines the formats for the left and right *y*-axes based on the predefined primary and secondary formats for the specified parameters, as described in "Determining Formats" on page 7-51.

`[ax,hlines1,hlines2] = plotyy(h,parameter,format1,format2)` plots the specified `parameter` using `format1` for the left *y*-axis and `format2` for the right *y*-axis.

`[ax,hlines1,hlines2] = plotyy(h, parameter1, ..., parametern, format1, format2)` plots the parameters `parameter1`, `...`, `parametern` on an X-Y plane using `format1` for the left *y*- axis and `format2` for the right *y*-axis.

`[ax,hlines1,hlines2] = plotyy(h,parameter1_1,...,parameter1_n1, format1,parameter2_1,...,parameter2_n2,format2)` plots the following data:

- Parameters `parameter1_1`, `...`, `parameter1_n1` using `format1` for the left *y*-axis.
- Parameters `parameter2_1`, `...`, `parameter2_n2` using `format2` for the right *y*-axis.

`[ax,hlines1,hlines2]=plotyy(h,parameter1_1,...,parameter1_n1, format1,parameter2_1,...,parameter2_n2,format2,xparameter, xformat,'condition1',value1,...,'conditionm',valuem, 'freq',freq,'pin',pin)` plots the specified parameters at the specified operating conditions for the object `h`.

`xparameter` is the independent variable to use in plotting the specified parameters. Several `xparameter` values are available for all objects. When you import `rfckt.amplifier`, `rfckt.mixer`, or `rfdata.data` object specifications from a `.p2d` or `.s2d` file, you can also specify any operating conditions from the file that have numeric values, such as bias.

The following table shows the most commonly available parameters and the corresponding `xparameter` values. The default settings listed in the table are used if `xparameter` is not specified.

Parameter Name	xparameter values
Pout, Phase, LS11, LS12, LS21, LS22	Pin (default), Freq
S11, S12, S21, S22, NF, IIP3, OIP3, GroupDelay, VSWRIn, VSWROut, GammaIn, GammaOut, FMIN, GammaOPT, RN, TF1, TF2, Gt, Ga, Gp, Gmag, Gmsg, GammaMS, GammaML, K, Delta, Mu, MuPrime	Freq
AM/AM, AM/PM	AM

`xformat` is the format to use for the specified `xparameter`. No `xformat` specification is needed when `xparameter` is an operating condition.

The following table shows the `xformat` values that are available for the `xparameter` values listed in the preceding table, along with the default settings that are used if `xformat` is not specified.

xparameter values	xformat values
Pin	dBm (default), mW, W, dBW
Freq	THz, GHz, MHz, KHz, Hz By default, `xformat` is chosen to provide the best scaling for the given `xparameter` values.
AM	Magnitude (decibels) (default), Magnitude (linear)

`condition1,value1`, `...`, `conditionm,valuem` are the optional condition/value pairs at which to plot the specified parameters. These pairs are usually operating conditions from a `.p2d` or `.s2d` file. For some parameters, you can specify a set of frequency or input power values at which to plot the specified parameter.

For example:

- When plotting large-signal S-parameters as a function of input power, you can specify frequency points of interest using condition/value pairs.
- When plotting large-signal S-parameters as a function of frequency, you can specify input power levels of interest using condition/value pairs.
- When plotting parameters as a function of an operating condition, you can specify both frequency and input power values using condition/value pairs.

`freq` is the optional frequency value, in hertz, at which to plot the specified parameters.

`pin` is the optional input power value, in dBm, at which to plot the specified parameters.

If `h` has multiple operating conditions, such as from a `.p2d` or `.s2d` file, the `plotyy` method operates as follows:

- If you do not specify any operating conditions as arguments to the `plotyy` method, then the method plots the parameter values based on the currently selected operating condition.
- If you specify one or more operating conditions, the `plotyy` method plots the parameter values based on those operating conditions.
- When you use an operating condition for the `xparameter` input argument, the method plots the parameters for all operating condition values.

Determining Formats

When you call `plotyy` without specifying the plot formats for the left and right *y*-axes, `plotyy` determines the formats from the predefined primary and secondary formats for the one or more specified parameters.

This section contains the following topics:

- “Primary and Secondary Formats” on page 7-52
- “Determining Formats for One Parameter” on page 7-53
- “Determining Formats for Multiple Parameters” on page 7-53

Primary and Secondary Formats

The following table shows the primary and secondary formats for the parameters for all circuit and data objects. Use the `listparam` method to list the valid parameters for a particular object. Use the `listformat` method to list valid formats.

Parameter	Primary Format	Secondary Format
S11, S12, S21, S22	Magnitude (decibels)	Angle (Degrees)
LS11, LS12, LS21, LS22	Magnitude (decibels)	Angle (Degrees)
NF	Magnitude (decibels)	
OIP3	dBm	W
Pout	dBm	W
Phase	Angle (Degrees)	
AM/AM	Magnitude (decibels)	
AM/PM	Angle (Degrees)	
GammaIn, GammaOut	Magnitude (decibels)	Angle (Degrees)
Gt, Ga, Gp, Gmag, Gmsg	Magnitude (decibels)	
Delta	Magnitude (decibels)	Angle (Degrees)
TF1, TF2	Magnitude (decibels)	Angle (Degrees)
GammaMS, GammaML	Magnitude (decibels)	Angle (Degrees)
VSWRIn, VSWROut	Magnitude (decibels)	
GroupDelay	ns	
Fmin	Magnitude (decibels)	
GammaOPT	Magnitude (decibels)	Angle (Degrees)
K, Mu, MuPrime	None	
RN	None	
PhaseNoise	dBc/Hz	
NTemp	K	
NFactor	None	

Determining Formats for One Parameter

When you specify only one parameter for plotting, `plotyy` creates the plot as follows:

- The predefined primary format is the format for the left *y*-axis.
- The predefined secondary format is the format for the right *y*-axis.

If the specified parameter does not have the predefined secondary format, `plotyy` behaves the same way as `plot`, and does not add a second y-axis to the plot.

Determining Formats for Multiple Parameters

To plot multiple parameters on two *y*-axes, `plotyy` tries to find two formats from the predefined primary and secondary formats for the specified parameters. To be used in the plot, the formats must meet the following criteria:

- Each format must be a valid format for at least one parameter.
- Each parameter must be plotted at least on one *y*-axis.

If `plotyy` cannot meet this criteria it issues an error message.

The function uses the following algorithm to determine the two parameters:

1. Look up the primary and secondary formats for the specified parameters.
2. If one or more pairs of primary-secondary formats meets the preceding criteria for all parameters:

   - Select the pair that applies to the most parameters.
   - Use these formats to create the plot.

   Otherwise, proceed to the next step.
3. If no pairs of primary-secondary formats meet the criteria for all parameters, try to find one or more pairs of primary-primary formats that meets the criteria. If one or more pairs of primary-primary formats meets the preceding criteria for all parameters:

   - Select the pair that applies to the most parameters.
   - Use these formats to create the plot.

   Otherwise, proceed to the next step.

4 If the preceding steps fail to produce a plot, try to find one format from the predefined primary formats. If a primary format is valid for all parameters, use this format to create the plot with the MATLAB `plot` function.

   If this is not successful, issue an error message.

The following example shows how `plotyy` applies this criteria to create plots.

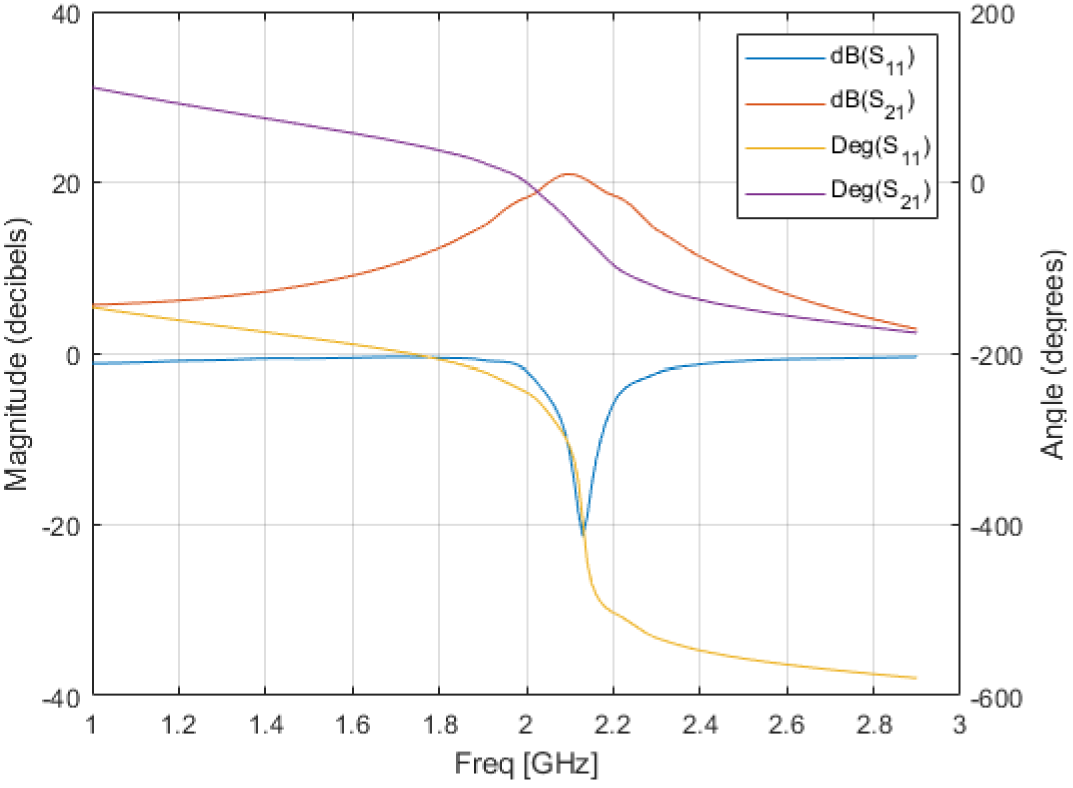
At the MATLAB prompt:

1 Type this command to create an `rfckt` object called `amp`:

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

2 Type this command to plot the `S11` and `S21` parameters of `amp` on two *y*-axis:

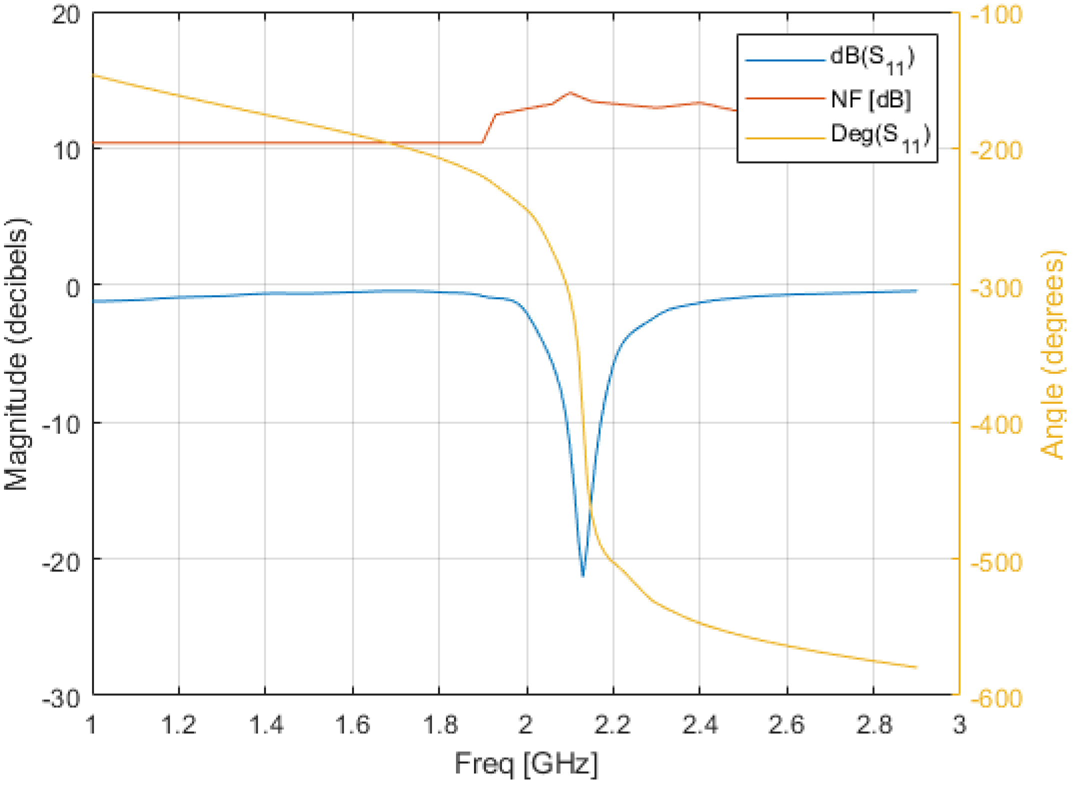
```
plotyy(amp, 'S11', 'S21')
```



The primary and secondary formats for both `S11` and `S21` are `Magnitude (decibels)` and `Angle (Degrees)`, respectively, so `plotyy` uses this primary-secondary format pair to create the plot

3 Type this command to plot the `S11` and `NF` parameters of `amp` on two *y*-axis:

```
plotyy(amp, 'S11', 'NF')
```



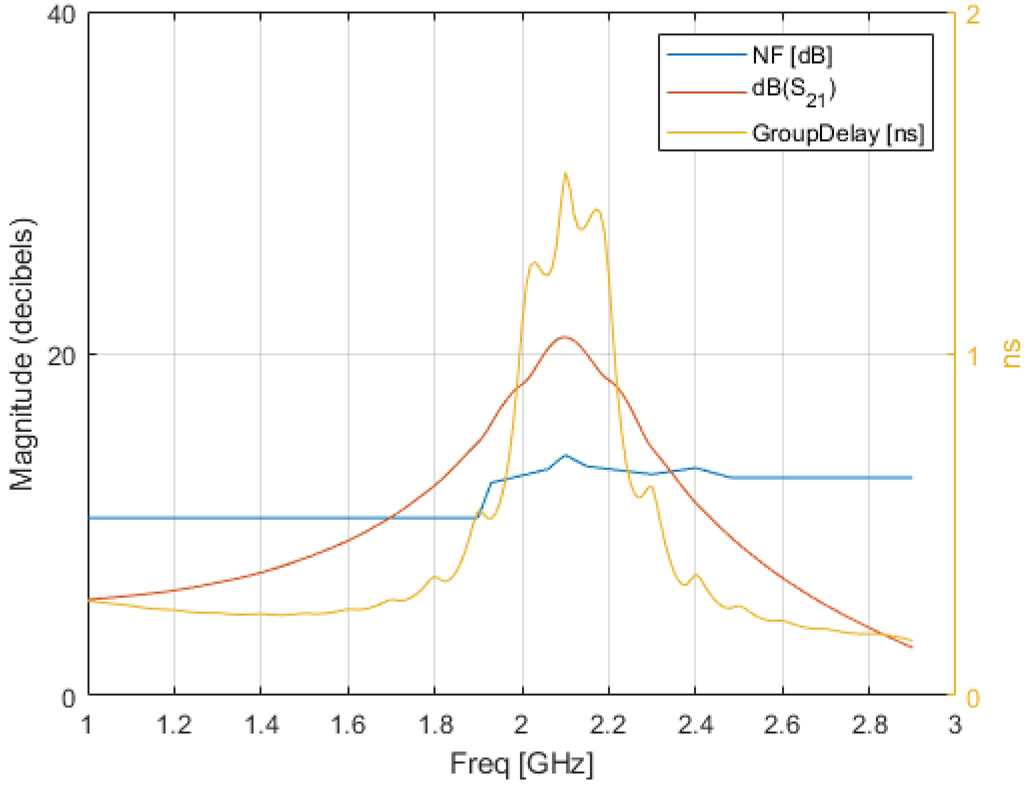
The primary and secondary formats for `S11` are `Magnitude (decibels)` and `Angle (Degrees)`, respectively.

- `Magnitude (decibels)` is a valid format for both `S11` and `NF`
- `Angle (Degrees)` is a valid format for `S11`.

These formats both meet the preceding criteria, so the function uses this primary-secondary format pair to create the plot.

4 Type this command to plot the `NF`, `S21` and `GroupDelay` parameters of `amp` on two *y*-axis:

```
plotyy(amp, 'NF', 'S21', 'GroupDelay')
```



The primary and secondary formats for `S21` are `Magnitude (decibels)` and `Angle (Degrees)`, respectively. Both `NF` and `GroupDelay` have only a primary format.

- `Magnitude (decibels)` is the primary format for `NF`.
- `ns` is the primary format for `GroupDelay`.

There is no primary-secondary format pair that meets the preceding criteria, so `plotyy` tries to find a pair of primary formats that meet the criteria. `plotyy` creates the plot using:

- `Magnitude (decibels)` for the left `y`-axis.

  This format is valid for both `NF` and `S21`.

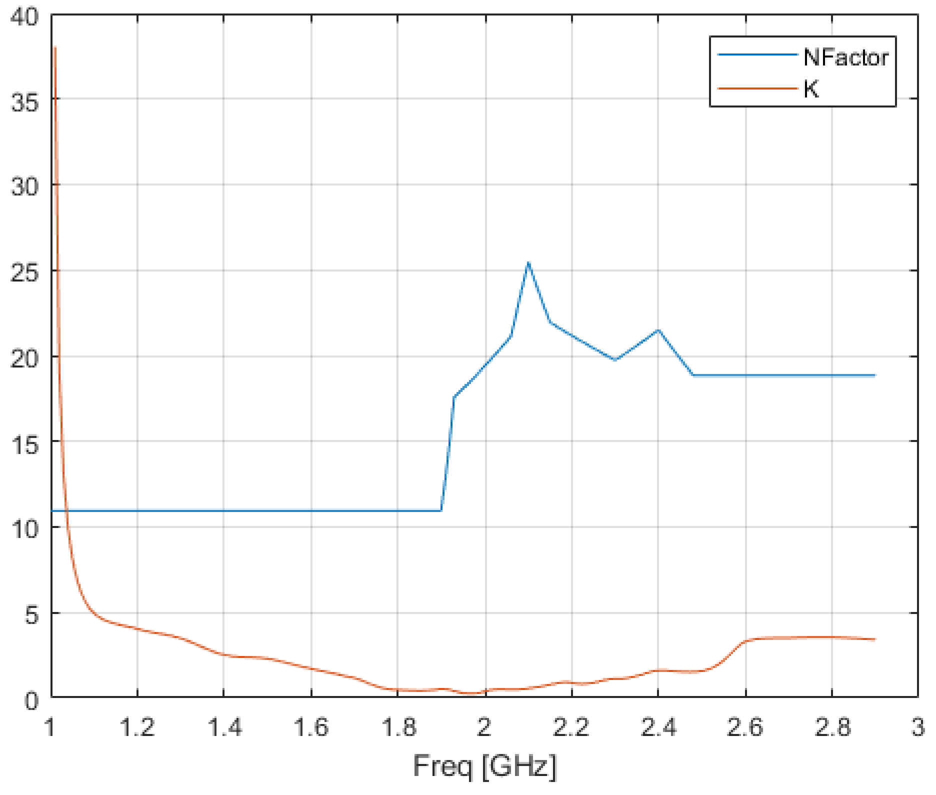
- `ns` for the right y-axis.

  This format is valid for `GroupDelay` .

These formats meet the criteria.

5 Type this command to plot the `NFactor` and `K` parameters of `amp` on two *y*-axis:

```
plotyy(amp, 'NFactor', 'K')
```



Both `NFactor` and `K` have only a primary format, `None`, so `plotyy` calls the `plot` command to create a plot with a single *y*-axis whose format is `None`.

6 Type this command to plot the `NTemp`, `S21` and `NFactor` parameters of `amp` on two *y*-axes:

```
plotyy(amp, 'NTemp', 'S21', 'NFactor')

??? Error using ==> rfdata.data.plotyyprocess at 97
No format specified for input parameters and cannot reconcile
default formats. Try reducing the number of parameters to plotyy
and explicitly specifying formats.
```

The primary and secondary formats for `S21` are `Magnitude (decibels)` and `Angle (Degrees)`, respectively. Both `NTemp` and `NFactor` have only a primary format.

- `Kelvin` is the primary format for `NTemp`.
- `None` is the primary format for `NFactor`.

These parameters have no formats in common, so no formats meet the criteria and `plotyy` issues an error message.

See Also

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

Introduced in R2007a

polar

Plot specified circuit object parameters on polar coordinates

Syntax

```
p = polar(rfbudgetobject,i,j)
lineseries = polar(h,'parameter1',...,'parametern')
lineseries=polar(h,'parameter1',...,'parametern',
xparameter,xformat,'condition1',value1,..., 'conditionm',valuem,
'freq',freq,'pin',pin)
```

Description

`p = polar(rfbudgetobject,i,j)` plots the ith and j th S-parameter on polar plot for rfbudget`rfbudget` object. `p` is a polar plot function object.

`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 Chart Line. 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.

Examples

Plot Circuit Parameters of Network Object on Polar Plot

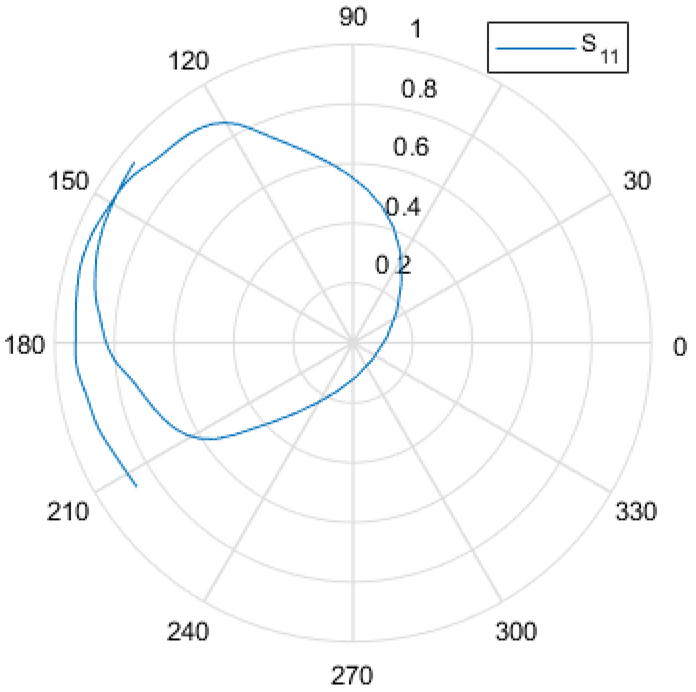
```
Create an amplifier object from |default.s2p|.
```

```
amp = read(rfckt.amplifier,'default.s2p');
```

Plot S11 on polar plot.

```
lineseries = polar(amp,'S11')
```

```
lineseries = 

  Line (S_{11}) with properties:

              Color: [0 0.4470 0.7410]
          LineStyle: '-'
          LineWidth: 0.5000
             Marker: 'none'
         MarkerSize: 6
    MarkerFaceColor: 'none'
              XData: [1x191 double]
              YData: [1x191 double]
              ZData: [1x0 double]

  Use GET to show all properties
```



See Also

`analyze` | `calculate` | `extract` | `getz0` | `listformat` | `listparam` | `loglog` | `plot` | `plotyy` | `read` | `restore` | `semilogx` | `semilogy` | `smith` | `write`

Introduced before R2006a

read

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

Syntax

```
h = read(h)

h = read(rfckt.datafile,filename)

h = read(rfckt.passive,filename)

h = read(rfckt.amplifier,filename)

h = read(rfckt.mixer,filename)

h = read(rfdata.data,filename)
```

Description

`h = read(h)` prompts you to select a file and then reads the data from that file into the circuit or data object, `h`. You can read data from an `.snp`, `.ynp`, `.znp`, `.hnp`, `.gnp`, or `.amp` file, where `n` is the number of ports. If `h` is an `rfckt.amplifier`, `rfckt.mixer`, or `rfdata.data` object, you can also read data from `.p2d` and `.s2d` files.

For an example of how to use RF Toolbox software to read data from a `.s2d` file, see Visualizing Mixer Spurs.

`h = read(h,filename)` updates `h` with data from the specified file. In this syntax, `h` can be a circuit or data object. `filename` is a character vector, representing the file name 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 file name must include the file extension.

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

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

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

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

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

Examples

Import Data

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

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

```
ckt_obj =
   rfckt.amplifier with properties:

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

References

EIA/IBIS Open Forum, “Touchstone File Format Specification,” Rev. 1.1, 2002 (`https://ibis.org/connector/touchstone_spec11.pdf`).

See Also

`analyze` | `calculate` | `extract` | `getz0` | `listformat` | `listparam` | `loglog` | `plot` | `plotyy` | `polar` | `restore` | `semilogx` | `semilogy` | `smith` | `write`

Introduced before R2006a

restore

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`.

Examples

Restore Data of Circuit Object

```
Create an amplifier object from |default.s2p| and restore data..

amp = read(rfckt.amplifier,'default.s2p');
restore(amp)


ans =

   rfckt.amplifier with properties:

         NoiseData: [1x1 rfdata.noise]
     NonlinearData: Inf
          IntpType: 'Linear'
       NetworkData: [1x1 rfdata.network]
             nPort: 2
    AnalyzedResult: [1x1 rfdata.data]
```

```
Name: 'Amplifier'
```

See Also

analyze | calculate | extract | getz0 | listformat | listparam | loglog | plot | plotyy | polar | read | semilogx | semilogy | smith | write

Introduced before R2006a

semilogx

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 Chart Line. The reference pages for MATLAB functions such as `figure`, `axes`, and `text` also list available properties and provide links to more complete property descriptions.

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

`lineseries=semilogx(h,'parameter1',...,'parametern', format,xparameter,xformat,'condition1',value1,..., 'conditionm',valuem, 'freq',freq,'pin',pin)` plots the specified parameters at the specified operating conditions for the object `h`.

`xparameter` is the independent variable to use in plotting the specified parameters. Several `xparameter` values are available for all objects. When you import `rfckt.amplifier`, `rfckt.mixer`, or `rfdata.data` object specifications from a `.p2d` or `.s2d` file, you can also specify any operating conditions from the file that have numeric values, such as bias.

The following table shows the most commonly available parameters and the corresponding `xparameter` values. The default settings listed in the table are used if `xparameter` is not specified.

Parameter Name	`xparameter` values
`Pout`, `Phase`, `LS11`, `LS12`, `LS21`, `LS22`	`Pin` (default), `Freq`
`S11`, `S12`, `S21`, `S22`, `NF`, `IIP3`, `OIP3`, `GroupDelay`, `VSWRIn`, `VSWROut`, `GammaIn`, `GammaOut`, `FMIN`, `GammaOPT`, `RN`, `TF1`, `TF2`, `Gt`, `Ga`, `Gp`, `Gmag`, `Gmsg`, `GammaMS`, `GammaML`, `K`, `Delta`, `Mu`, `MuPrime`	`Freq`
`AM/AM`, `AM/PM`	`AM`

`xformat` is the format to use for the specified `xparameter`. No `xformat` specification is needed when `xparameter` is an operating condition.

The following table shows the `xformat` values that are available for the `xparameter` values listed in the preceding table, along with the default settings that are used if `xformat` is not specified.

`xparameter` values	`xformat` values
`Pin`	`dBm` (default), `mW`, `W`, `dBW`
`Freq`	`THz`, `GHz`, `MHz`, `KHz`, `Hz` By default, `xformat` is chosen to provide the best scaling for the given `xparameter` values.
`AM`	`Magnitude (decibels)` (default), `Magnitude (linear)`

`condition1,value1`, `...`, `conditionm,valuem` are the optional condition/value pairs at which to plot the specified parameters. These pairs are usually operating conditions from a `.p2d` or `.s2d` file. For some parameters, you can specify a set of frequency or input power values at which to plot the specified parameter.

For example:

- When plotting large-signal S-parameters as a function of input power, you can specify frequency points of interest using condition/value pairs.
- When plotting large-signal S-parameters as a function of frequency, you can specify input power levels of interest using condition/value pairs.
- When plotting parameters as a function of an operating condition, you can specify both frequency and input power values using condition/value pairs.

`freq` is the optional frequency value, in hertz, at which to plot the specified parameters.

`pin` is the optional input power value, in dBm, at which to plot the specified parameters.

If `h` has multiple operating conditions, such as from a `.p2d` or `.s2d` file, the `semilogx` method operates as follows:

- If you do not specify any operating conditions as arguments to the `semilogx` method, then the method plots the parameter values based on the currently selected operating condition.
- If you specify one or more operating conditions, the `semilogx` method plots the parameter values based on those operating conditions.
- When you use an operating condition for the `xparameter` input argument, the method plots the parameters for all operating condition values.

Examples

Plot Parameters of Network Object Using Log Scale on X-Axis

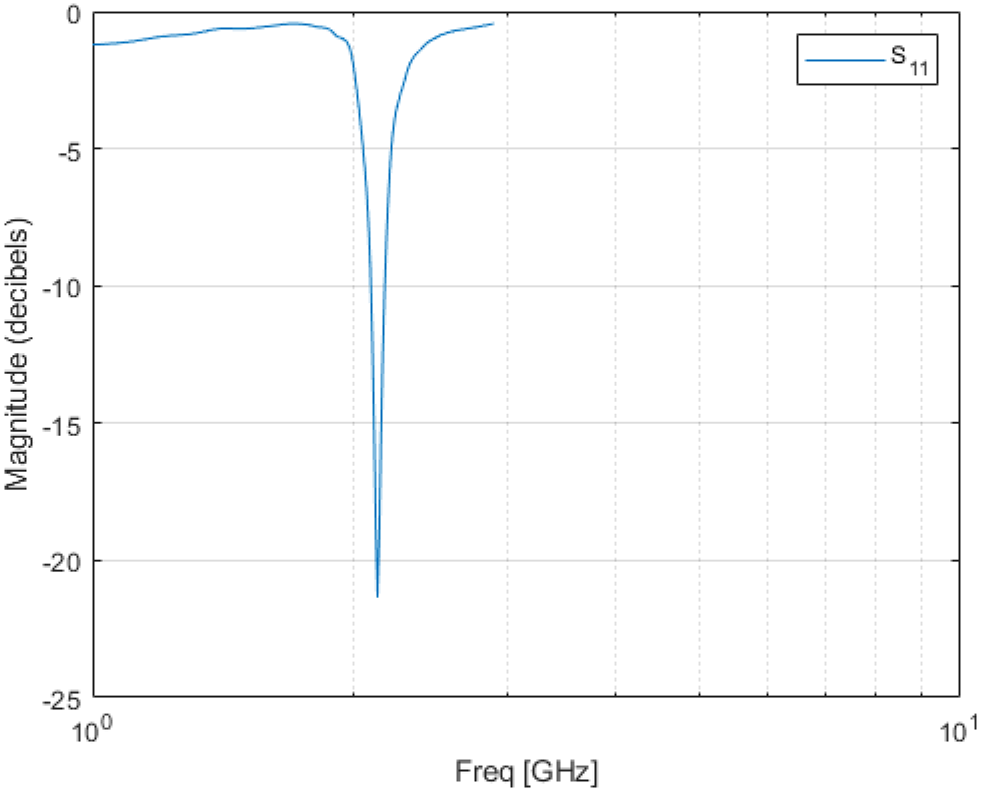
```
Create an amplifier object from |default.s2p|.
```

```
amp = read(rfckt.amplifier,'default.s2p');
```

Plot S11 using log scale on x-axis.

```
lineseries = semilogx(amp,'S11')
```

```
lineseries = 

  Line (S_{11}) with properties:

              Color: [0 0.4470 0.7410]
          LineStyle: '-'
          LineWidth: 0.5000
             Marker: 'none'
         MarkerSize: 6
    MarkerFaceColor: 'none'
              XData: [1x191 double]
              YData: [1x191 double]
              ZData: [1x0 double]

  Use GET to show all properties
```



See Also

analyze | calculate | extract | getz0 | listformat | listparam | loglog | plot | plotyy | polar | read | restore | semilogy | smith | write

Introduced in R2007a

semilogy

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 Chart Line . The reference pages for MATLAB functions such as `figure`, `axes`, and `text` also list available properties and provide links to more complete property descriptions.

Note Use the MATLAB `semilogy` function to create a semi-log scale plot of parameters that are specified as vector data and are not part of a circuit (`rfckt`) object or data (`rfdata`) object.

`lineseries=semilogy(h,'parameter1',...,'parametern', format,xparameter,xformat,'condition1',value1,..., 'conditionm',valuem, 'freq',freq,'pin',pin)` plots the specified parameters at the specified operating conditions for the object `h`.

`xparameter` is the independent variable to use in plotting the specified parameters. Several `xparameter` values are available for all objects. When you import `rfckt.amplifier`, `rfckt.mixer`, or `rfdata.data` object specifications from a `.p2d` or `.s2d` file, you can also specify any operating conditions from the file that have numeric values, such as bias.

The following table shows the most commonly available parameters and the corresponding `xparameter` values. The default settings listed in the table are used if `xparameter` is not specified.

Parameter Name	xparameter values
`Pout`, `Phase`, `LS11`, `LS12`, `LS21`, `LS22`	`Pin` (default), `Freq`
`S11`, `S12`, `S21`, `S22`, `NF`, `IIP3`, `OIP3`, `GroupDelay`, `VSWRIn`, `VSWROut`, `GammaIn`, `GammaOut`, `FMIN`, `GammaOPT`, `RN`, `TF1`, `TF2`, `Gt`, `Ga`, `Gp`, `Gmag`, `Gmsg`, `GammaMS`, `GammaML`, `K`, `Delta`, `Mu`, `MuPrime`	`Freq`
`AM/AM`, `AM/PM`	`AM`

`xformat` is the format to use for the specified `xparameter`. No `xformat` specification is needed when `xparameter` is an operating condition.

The following table shows the `xformat` values that are available for the `xparameter` values listed in the preceding table, along with the default settings that are used if `xformat` is not specified.

`xparameter` values	`xformat` values
`Pin`	`dBm` (default), `mW`, `W`, `dBW`
`Freq`	`THz`, `GHz`, `MHz`, `KHz`, `Hz` By default, `xformat` is chosen to provide the best scaling for the given `xparameter` values.
`AM`	`Magnitude (decibels)` (default), `Magnitude (linear)`

`condition1,value1`, `...`, `conditionm,valuem` are the optional condition/value pairs at which to plot the specified parameters. These pairs are usually operating conditions from a `.p2d` or `.s2d` file. For some parameters, you can specify a set of frequency or input power values at which to plot the specified parameter.

For example:

- When plotting large-signal S-parameters as a function of input power, you can specify frequency points of interest using condition/value pairs.
- When plotting large-signal S-parameters as a function of frequency, you can specify input power levels of interest using condition/value pairs.
- When plotting parameters as a function of an operating condition, you can specify both frequency and input power values using condition/value pairs.

`freq` is the optional frequency value, in hertz, at which to plot the specified parameters.

`pin` is the optional input power value, in dBm, at which to plot the specified parameters.

If `h` has multiple operating conditions, such as from a `.p2d` or `.s2d` file, the `semilogy` method operates as follows:

- If you do not specify any operating conditions as arguments to the `semilogy` method, then the method plots the parameter values based on the currently selected operating condition.
- If you specify one or more operating conditions, the `semilogy` method plots the parameter values based on those operating conditions.
- When you use an operating condition for the `xparameter` input argument, the method plots the parameters for all operating condition values.

Examples

Plot Parameters of Network Object Using Log Scale on Y-Axis

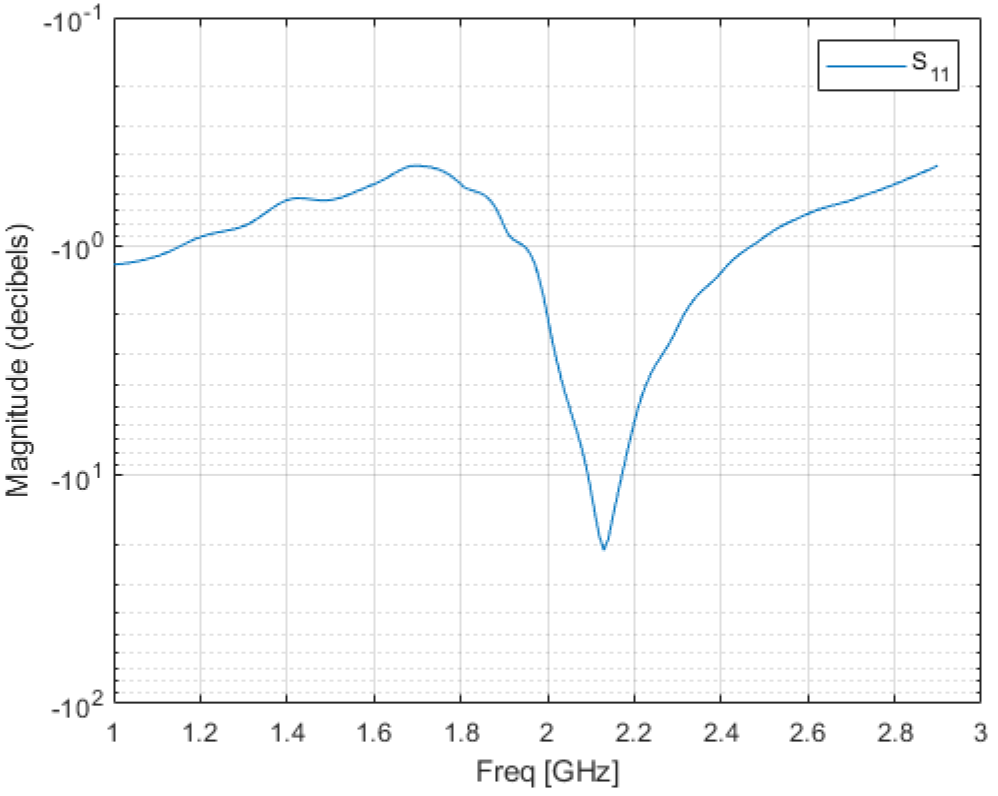
Create an amplifier object from |default.s2p|.

```
amp = read(rfckt.amplifier,'default.s2p');
```

Plot S11 using log scale on y-axis.

```
lineseries = semilogy(amp,'S11')
```

```
lineseries = 

  Line (S_{11}) with properties:

              Color: [0 0.4470 0.7410]
          LineStyle: '-'
          LineWidth: 0.5000
             Marker: 'none'
         MarkerSize: 6
    MarkerFaceColor: 'none'
              XData: [1x191 double]
              YData: [1x191 double]
              ZData: [1x0 double]

  Use GET to show all properties
```



See Also

`analyze` | `calculate` | `extract` | `getz0` | `listformat` | `listparam` | `loglog` | `plot` | `plotyy` | `polar` | `read` | `restore` | `semilogx` | `smith` | `write`

Introduced in R2007a

setop

Set operating conditions

Syntax

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

Description

`setop(h)` lists the available values for all operating conditions of the object `h`. Operating conditions only apply to objects you import from a `.p2d` or `.s2d` file. To import these types of data into an object, use the `read` method. Operating conditions are not listed with other properties of an object.

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

`setop(h,'Condition1',value1,'Condition2',value2,...)` changes the operating conditions of the circuit or data object, `h`, to those specified by the condition/value pairs. Conditions you do not specify retain their original values. The method ignores any conditions that are not applicable to the specified object. Ignoring these conditions lets you apply the same set of operating conditions to an entire network where different conditions exist for different components.

When you set the operating conditions for a network that contains several objects, the software does not issue an error or warning if the specified conditions cannot be applied to all objects. For some networks, this lack of error or warning lets you call the `setop` method once to apply the same set of operating conditions to any objects where operating conditions are applicable. However, you may want to specify a network that contains one or more of the following:

- Several objects with different sets of operating conditions.
- Several objects with the same set of operating conditions that are configured differently.

To specify operating conditions one of these types of networks, use a separate call to the `setop` method for each object.

Examples

List Operating Conditions of Network Object

List the operating conditions of `rfckt.amplifier` object.

```
ckt1 = read(rfckt.amplifier, 'default.p2d');
setop(ckt1)
```

```
Operating conditions set 1:
    'Bias'    '1.5'
```

Analyze Object Under Specific Operating Conditions

Analyze `rfckt.amplifier` under specific operating conditions set using the function `setop`.

```
ckt1 = read(rfckt.amplifier, 'default.p2d');
freq = ckt1.AnalyzedResult.Freq;
setop(ckt1, 'Bias', '1.5');
analyze(ckt1, freq)
```

```
ans =
   rfckt.amplifier with properties:

         NoiseData: [1x1 rfdata.noise]
     NonlinearData: [1x1 rfdata.p2d]
          IntpType: 'Linear'
       NetworkData: [1x1 rfdata.network]
             nPort: 2
    AnalyzedResult: [1x1 rfdata.data]
              Name: 'Amplifier'
```

See Also

`getop`

Introduced in R2007a

smith

Plot specified circuit object parameters on Smith chart

Syntax

```
smith(hnet,i,j)
hsm = smith(hnet,i,j)
[lineseries,hsm] = smith(h,parameter1,...,parametern,type)
[lineseries,hsm] = smith(h,'parameter1',...,'parametern',
type,xparameter,xformat,'condition1',value1,...,
'conditionm',valuem, 'freq',freq,'pin',pin)
```

Description

`smith(hnet,i,j)` plots the (*i*, *j*)th parameter of `hnet` on a Smith Chart. `hnet` is an RF Toolbox network parameter object. The inputs `i` and `j` are positive integers whose value is less than or equal to 2 for hybrid and hybrid-g parameter objects, or less than or equal to `hnet.NumPorts` for ABCD, S, Y, or Z-parameter objects.

`hsm = smith(hnet,i,j)` returns the line series handle used to create the plot, `hsm`.

`[lineseries,hsm] = smith(h,parameter1,...,parametern,type)` plots the network parameters `parameter1`, `...`, `parametern` from the object `h` on a Smith chart. `h` is the handle of a circuit (`rfckt`) or data (`rfdata`) object that contains *n*-port network parameter data. `type` is a text value that specifies the type of Smith chart:

- `'z'` (default)
- `'y'`
- `'zy'`

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

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

`[lineseries,hsm] = smith(h,'parameter1',...,'parametern', type,xparameter,xformat,'condition1',value1,..., 'conditionm',valuem, 'freq',freq,'pin',pin)` plots the specified parameters at the specified operating conditions for the object `h`.

`xparameter` is the independent variable to use in plotting the specified parameters. Several `xparameter` values are available for all objects. When you import 2-port `rfckt.amplifier`, `rfckt.mixer`, or `rfdata.data` object specifications from a `.p2d` or `.s2d` file, you can also specify any operating conditions from the file that have numeric values, such as bias.

The following table shows the most commonly available parameters and the corresponding `xparameter` values. The default settings listed in the table are used if `xparameter` is not specified.

Parameter Name	**`xparameter` values**
`Pout`, `Phase`, `LS11`, `LS12`, `LS21`, `LS22`	`Pin` (default), `Freq`
`S11`, `S12`, `S21`, `S22`, `NF`, `IIP3`, `OIP3`, `VSWRIn`, `VSWROut`, `GammaIn`, `GammaOut`, `FMIN`, `GammaOPT`, `RN`	`Freq`
`AM/AM`, `AM/PM`	`AM`

`xformat` is the format to use for the specified `xparameter`. No `xformat` specification is needed when `xparameter` is an operating condition.

The following table shows the `xformat` values that are available for the `xparameter` values listed in the preceding table, along with the default settings that are used if `xformat` is not specified.

`xparameter` values	**`xformat` values**
`Pin`	`dBm` (default), `mW`, `W`, `dBW`
`Freq`	`THz`, `GHz`, `MHz`, `KHz`, `Hz` By default, `xformat` is chosen to provide the best scaling for the given `xparameter` values.
`AM`	`Magnitude (decibels)` (default), `Magnitude (linear)`

`condition1,value1`, `...`, `conditionm,valuem` are the optional condition/value pairs at which to plot the specified parameters. These pairs are usually operating conditions from a `.p2d` or `.s2d` file. For some parameters, you can specify a set of frequency or input power values at which to plot the specified parameter.

For example:

- When plotting large-signal S-parameters as a function of input power, you can specify frequency points of interest using condition/value pairs.
- When plotting large-signal S-parameters as a function of frequency, you can specify input power levels of interest using condition/value pairs.
- When plotting parameters as a function of an operating condition, you can specify both frequency and input power values using condition/value pairs.

`freq` is the optional frequency value, in hertz, at which to plot the specified parameters.

`pin` is the optional input power value, in dBm, at which to plot the specified parameters.

If `h` has multiple operating conditions, such as from a `.p2d` or `.s2d` file, the `smith` method operates as follows:

- If you do not specify any operating conditions as arguments to the `smith` method, then the method plots the parameter values based on the currently selected operating condition.
- If you specify one or more operating conditions, the `smith` method plots the parameter values based on those operating conditions.
- When you use an operating condition for the `xparameter` input argument, the method plots the parameters for all operating condition values.

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

Changing Properties of the Plotted Lines

The `smith` method returns `lineseries`, a column vector of handles to `lineseries` objects, one handle per plotted line. Use the Chart Line 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`.
`LabelVisible`	Visibility of the line labels.	`'on'` (default) or `'off'`
`LineType`	Line spec for a Z or Y Smith chart. For a ZY Smith chart, the Z line spec.	`LineSpec`. Default is `'-'` (solid line).
`LineWidth`	Line width for a Z or Y Smith chart. For a ZY Smith chart, the Z line width.	Number of points. Default is `0.5`.
`SubColor`	The Y line color for a ZY Smith chart.	`ColorSpec`. Default is `[0.8 0.8 0.8]` (medium gray).
`SubLineType`	The Y line spec for a ZY Smith chart.	`LineSpec`. Default is `':'` (dotted line).
`SubLineWidth`	The Y line width for a ZY Smith chart.	Number of points. Default is `0.5`.

Property Name	Description	Units, Values
`Type`	Type of Smith chart.	`'z'` (default), `'y'`, or `'zy'`
`Value`	Two-row matrix. Row 1 specifies the values of the constant resistance and reactance lines that appear on the chart. For the constant resistance/reactance lines, each element in Row 2 specifies the value of the constant reactance/resistance line at which the corresponding line specified in Row 1 ends.	2-by-n matrix. Default is `[0.2000 0.5000 1.0000 2.0000 5.0000; 1.0000 2.0000 5.0000 5.0000 30.0000]`

Examples

Plot Parameters of Network Object on Smith Chart

Create an amplifier object from |default.s2p|.

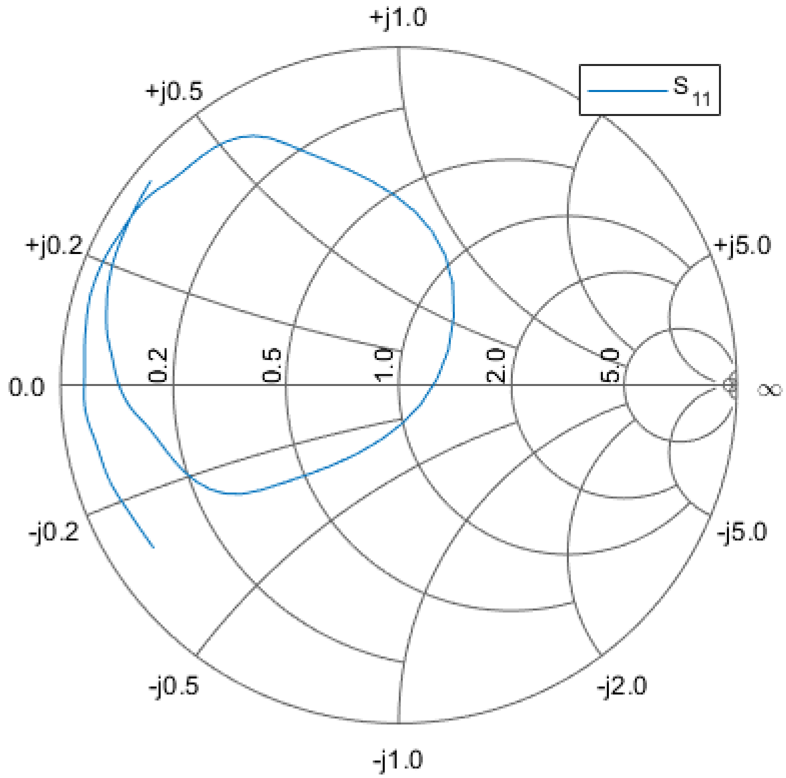
```
amp = read(rfckt.amplifier,'default.s2p');
```

Plot S11 on the smith chart.

```
smith(amp,'S11')
```

```
ans =

  Line (S_{11}) with properties:

              Color: [0 0.4470 0.7410]
          LineStyle: '-'
          LineWidth: 0.5000
             Marker: 'none'
         MarkerSize: 6
    MarkerFaceColor: 'none'
              XData: [1x191 double]
```

```
        YData: [1x191 double]
        ZData: [1x0 double]

  Use GET to show all properties
```



See Also

analyze | calculate | circle | getz0 | listformat | listparam | loglog | plot | plotyy | polar | read | restore | semilogx | semilogy | write

Introduced before R2006a

stepresp

Step-signal response of rational function object

Syntax

```
[yout,tout] = stepresp(h, ts, n, trise)
```

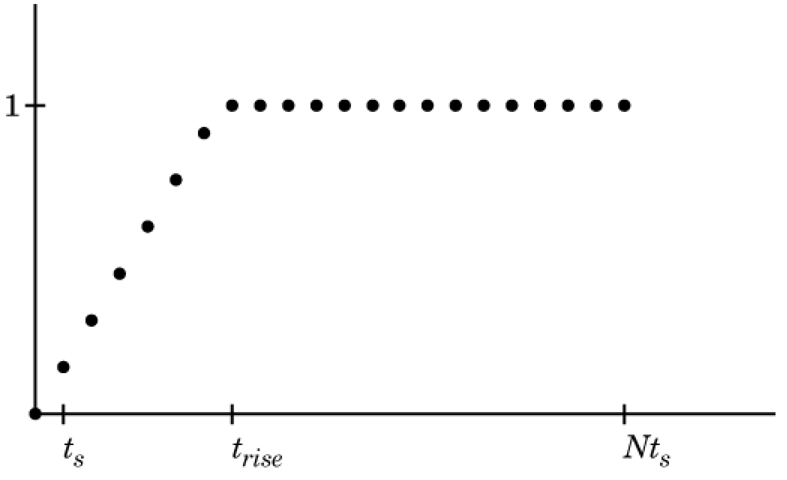
Description

`[yout,tout] = stepresp(h, ts, n, trise)` calculates the time-domain response of a rational function object, `h`, to a step signal, defined as:

$$\begin{cases} U(kt_s) = kt_s / t_{rise}, & 0 \le k < \left(t_{rise} / t_s\right) \\ U(kt_s) = 1, & \left(t_{rise} / t_s\right) \le k \le N \end{cases}$$

The input `h` is the handle of a rational function object returned by `rationalfit`. The variable t_s is the sample time, `ts`; N is the number of samples, `n`; and t_{rise} is the time, `trise`, that it takes for the step signal to reach its maximum value. The variable k is an integer between 0 and N, referring to the index of the samples.

The following figure illustrates the construction of this signal.



The output `yout` is the response of the step signal at time `tout`.

Examples

Calculate Step Response

Calculate the step response of a rational function object from the file `passive.s2p`. Read `passive.s2p`.

```
S = sparameters('passive.s2p');
freq = S.Frequencies;
```

Get S11 and convert to a TDR transfer function.

```
s11 = rfparam(S,1,1);
Vin = 1;
tdrfreqdata = Vin*(s11+1)/2;
```

Fit to a rational function object.

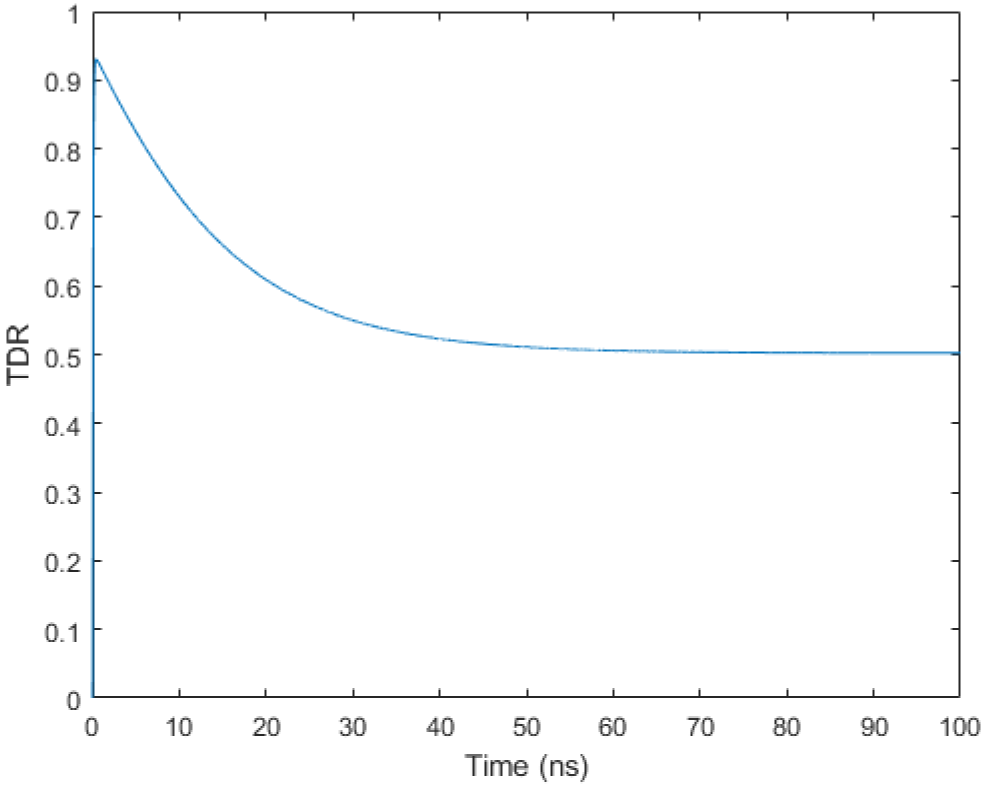
```
tdrfit = rationalfit(freq,tdrfreqdata);
```

Define parameters for a step signal. Define parameters for a step signal

```
Ts = 1.0e-11;
N = 10000;
Trise = 1.0e-10;
```

Calculate the step response for TDR and plot it

```
[tdr,t1] = stepresp(tdrfit,Ts,N,Trise);
figure
plot(t1*1e9,tdr)
ylabel('TDR')
xlabel('Time (ns)')
```



See Also

`freqresp` | `rationalfit` | `rfmodel.rational` | `timeresp`

Introduced in R2010a

table

Display specified RF object parameters in Variable Editor

Syntax

```
table(h,param1,format1,..., paramn,formatn)
table(h,'budget',param1,format1,...,paramn,formatn)
```

Description

`table(h,`*`param1`*`,`*`format1`*`,..., `*`paramn`*`,`*`formatn`*`)` displays the specified parameters *`param1`* through *`paramn`*, with units *`format1`* through *`formatn`*, in the Variable Editor. The input `h` is a function handle to an `rfckt` object.

The method creates a structure in the MATLAB workspace and constructs the name of the structure from the names of the object and parameters you provide. Specify parameters and formats in pairs. If you do not specify a format, the method uses the default format for that parameter.

To list valid parameters and parameter formats for `h`, use the `listparam` and `listformat` methods.

`table(h,'budget',`*`param1`*`,`*`format1`*`,...,`*`paramn`*`,`*`formatn`*`)` specified budget parameters of an `rfckt.cascade` object `h`.

Examples

Use Table to Display Link Budget of RF Cascade

Construct a cascaded RFCKT object.

```
Cascaded_Ckt = rfckt.cascade('Ckts', ...
               {rfckt.txline('LineLength', .001), ...
```

```
               rfckt.amplifier, rfckt.txline( ...
               'LineLength', 0.025, 'PV', 2.0e8)})
```

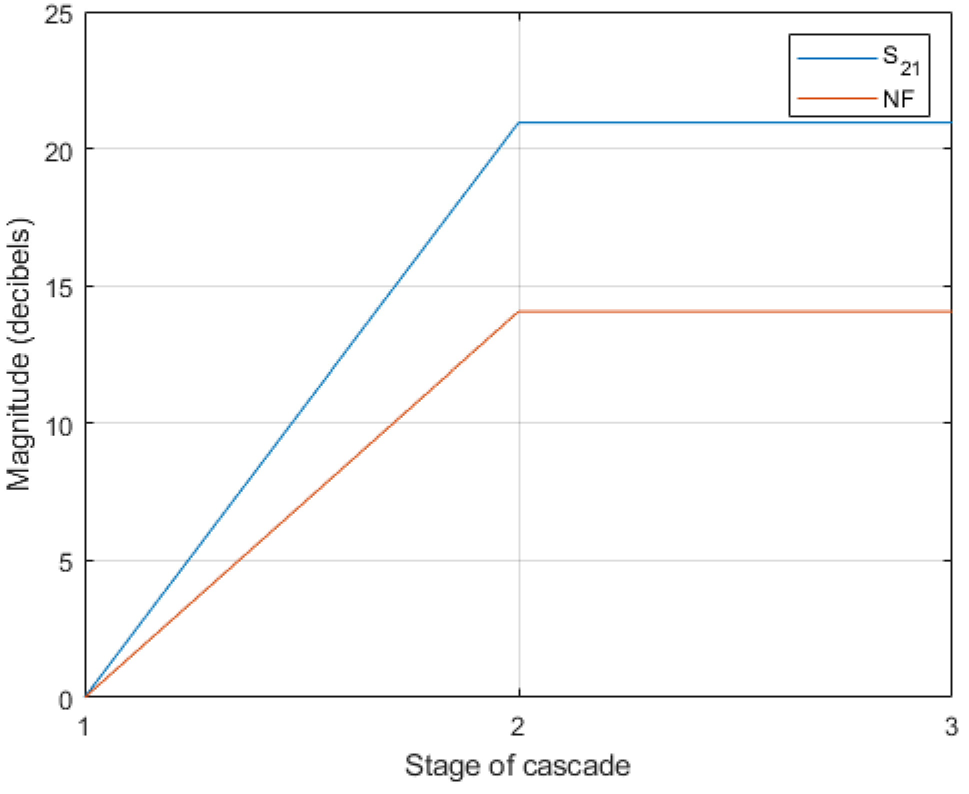
```
Cascaded_Ckt =
   rfckt.cascade with properties:

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

Analyze the RF cascade in frequency domain at 2.1 GHz.

```
freq = 2.1e9;
analyze(Cascaded_Ckt,freq);
```

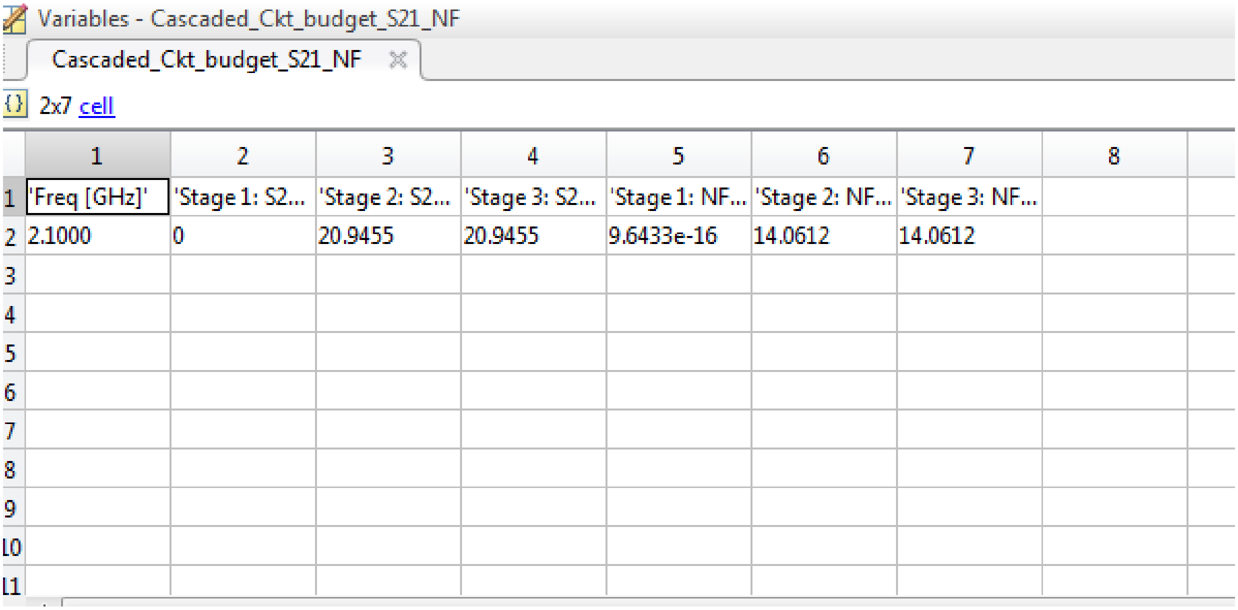
Plot the budget S21 and noise figure.

```
plot(Cascaded_Ckt,'budget','S21','NF');
```



Display the budget S21 and noise figure in a table. The table is displayed as a spreadsheet when the user runs the `table` command in the MATLAB® command line.

```
table(Cascaded_Ckt,'budget','S21','NF')
```



Variables - Cascaded_Ckt_budget_S21_NF

Cascaded_Ckt_budget_S21_NF

2x7 cell

	1	2	3	4	5	6	7	8
1	'Freq [GHz]'	'Stage 1: S2...	'Stage 2: S2...	'Stage 3: S2...	'Stage 1: NF...	'Stage 2: NF...	'Stage 3: NF...	
2	2.1000	0	20.9455	20.9455	9.6433e-16	14.0612	14.0612	
3								
4								
5								
6								
7								
8								
9								
10								
11								

See Also

`openvar` | `plot`

Introduced in R2010b

timeresp

Time response for rational function object

Syntax

```
[y,t] = timeresp(h,u,ts)
```

Description

`[y,t] = timeresp(h,u,ts)` computes the output signal, `y`, that the rational function object, `h`, produces in response to the given input signal, `u`.

The input `h` is the handle of a rational function object returned by `rationalfit`. `ts` is a positive scalar value that specifies the sample time of the input signal.

The output `y` is the output signal. RF Toolbox software computes the value of the signal at the time samples in the vector `t` using the following equation.

$$Y(n) = sum(C.*X(n - Delay/ts)) + D*U(n - Delay/ts)$$

where

$$X(n+1) = F*X(n) + G*U(n)$$
$$X(1) = 0$$
$$F = \exp(A*ts)$$
$$G = (F-1)./A$$

and `A`, `C`, `D`, and `Delay` are properties of the rational function 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 object to the data and using the `timeresp` method to compute the time response of the object.

```
% 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 object
orig_data=read(rfdata.data,'default.s2p');
freq=orig_data.Freq;
data=orig_data.S_Parameters(2,1,:);
fit_data=rationalfit(freq,data);

% Compute the time response
[y,t]=timeresp(fit_data,InputSignal,SampleTime);
```

See Also

freqresp | rationalfit | rfmodel.rational | writeva

Introduced in R2007a

write

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 character vector representing the file name of a `.snp`, `.ynp`, `.znp`, `.hnp`, or `.amp` file, where `n` is the number of ports. The default `filename` extension is `.snp`. `write` returns `True` if the operation is successful and returns `False` otherwise.

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

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

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

The `printformat` specifies the precision of the network and noise parameters. The default value is `%22.10f`. This value means the method writes the data using fixed-point notation with a precision of 10 digits. The minimum positive value the `write` method can express by default is `1e-10`. For greater precision, specify a different `printformat`. See the Format specification for `fprintf`.

The `freqformat` specifies the precision of the frequency. The default value is `%-22.10f`. See the Format specification for `fprintf`.

Note The method only writes property values from `data` that the specified output file supports. For example, Touchstone files, which have the `.snp`, `.ynp`, `.znp`, or `.hnp` extension, do not support noise figure or output third-order intercept point. Consequently, the `write` method does not write these property values to these such files.

Examples

Write Data to Touchstone File

Analyze the data stored in the file `default.s2p` for a set of frequency values. Use the `write` method to store the results in a file called `test.s2p`.

```
orig_data=read(rfdata.data,'default.s2p')
```

```
orig_data = 
   rfdata.data with properties:

            Freq: [191x1 double]
    S_Parameters: [2x2x191 double]
      GroupDelay: [191x1 double]
              NF: [191x1 double]
            OIP3: [191x1 double]
              Z0: 50.0000 + 0.0000i
              ZS: 50.0000 + 0.0000i
              ZL: 50.0000 + 0.0000i
        IntpType: 'Linear'
            Name: 'Data object'
```

```
freq = [1:.1:2]*1e9;
analyze(orig_data,freq);
write(orig_data,'test.s2p')
```

```
ans = logical
   1
```

References

EIA/IBIS Open Forum, “Touchstone File Format Specification,” Rev. 1.1, 2002 (`https://ibis.org/connector/touchstone_spec11.pdf`).

See Also

`analyze` | `calculate` | `extract` | `getz0` | `listformat` | `listparam` | `loglog` | `plot` | `plotyy` | `polar` | `read` | `restore` | `semilogx` | `semilogy` | `smith`

Introduced before R2006a

writeva

Write Verilog-A description of rational function object

Syntax

```
status = writeva(h,filename,innets,outnets, ...
          discipline,printformat,filestoinclude)
```

Description

`status = writeva(h,filename,innets,outnets,discipline,printformat, filestoinclude)` writes a Verilog-A module that describes a rational function 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 rational function object. Typically, the `rationalfit` function creates this object when you fit a rational function to a set of data.

`filename` is a character vector 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 character vector or a cell array of character vectors that specifies the name of each of the module's input nets. The default is `'in'`.

`outnets` is a character vector or a cell array of character vectors that specifies the name of each of the module's output nets. The default is `'out'`.

`printformat` is a character vector 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 rational function object.

The default is `'%15.10e'`. For more information on how to specify `printformat`, see the Format specification for `fprintf`.

`discipline` 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 array of character vectors 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'``.

Note `writeva` only accepts a single rational fit object. It does not work with an array/matrix of rational fit objects

---

For more information on Verilog-A, use the `Verilog-A Reference Manual`.

See Also

`freqresp` | `rationalfit` | `rfmodel.rational` | `timeresp`

Introduced in R2006b

newref

Change reference impedance of S-parameters

Syntax

```
hs2 = newref(hs,Z0)
```

Description

`hs2 = newref(hs,Z0)` creates an S-parameter object, `hs2`, by converting the S-parameters in `hs` to the specified reference impedance, `Z0`.

Examples

Change Reference Impedance of S-parameters

Create an S-parameters object from data in the file, `default.s2p`.

```
hs = sparameters('default.s2p');
```

Change the reference impedance to 40 ohms.

```
hs2 = newref(hs,40)
```

```
hs2 =
  sparameters: S-parameters object

       NumPorts: 2
    Frequencies: [191x1 double]
     Parameters: [2x2x191 double]
      Impedance: 40

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

Input Arguments

hs — S-parameters
network parameter object

S-parameters, specified as an RF Toolbox network parameter object. To create this type of object, use the `sparameters` function.

Z0 — Reference impedance
real positive scalar

Characteristic impedance, in ohms, specified as a real positive scalar.

Output Arguments

hs2 — S-parameters
network parameter object

S-parameters with reference impedance `Z0`, returned as an RF Toolbox network parameter object.

See Also

`sparameters`

Introduced in R2012b

rfinterp1

Interpolate network parameter data at new frequencies

Syntax

```
objnew = rfinterp1(objold,newfreq)
objnew = rfinterp1(objold,newfreq,'extrap')
```

Description

`objnew = rfinterp1(objold,newfreq)` interpolates the network parameter data in `objold` at the specified frequencies, `newfreq`, storing the results in `objnew`. `rfinterp1` uses the MATLAB function `interp1` to interpolate each individual `(i,j)` parameter of `objold` to the new frequencies.

If any value of the specified frequency is outside of the range specified by `objold.frequencies`, then `rfinterp1` inserts `NaNs` into `objnew` for those frequency values.

`objnew = rfinterp1(objold,newfreq,'extrap')` interpolates as above, but if any value of the specified frequency values are outside of the range of `objold.frequencies`, then `rfinterp1` will extrapolate flat using the nearest values in the frequency range.

Examples

Interpolate S-parameter data

Read the data from the file `default.s2p` into an S-parameter object.

```
hnet = sparameters('default.s2p');
```

Interpolate the data at a specified set of frequencies.

```
freq = [1.2:0.2:2.8]*1e9;
hnet2 = rfinterp1(hnet,freq)

hnet2 = 
  sparameters: S-parameters object

       NumPorts: 2
    Frequencies: [9x1 double]
     Parameters: [2x2x9 double]
      Impedance: 50

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

Input Arguments

`objold` — Original data
network parameter object

Data to interpolate, specified as an RF Toolbox network parameter object. `objold` must be a network parameter object of the following types: s-parameters, t-parameters, y-parameters, z-parameters, h-parameters, g-parameters, or abcd-parameters.

`newfreq` — Frequencies
vector of positive numbers

Frequencies of interpolation, specified as a vector of positive numbers ordered from smallest to largest.

Output Arguments

`objnew` — Interpolated data
network parameter object

Result of interpolation, returned as an RF Toolbox network parameter object of the same type as `objnew`.

Algorithms

The function uses the MATLAB function `interp1` to perform the interpolation operation. Overall performance is similar to the RF Toolbox `analyze` function. However, behaviors of the two functions differ when `freq` contains frequencies outside the range of the original data:

- `analyze` performs a zeroth-order extrapolation for out-of-range data points.
- `rfinterp1` inserts `NaN` values for out-of-range data points.

See Also

`analyze` | `interp1`

Introduced in R2012b

rfparam

Extract vector of network parameters

Syntax

```
n_ij = rfparam(hnet,i,j)
abcd_vector = rfparam(habcd,abcdflag)
```

Description

`n_ij = rfparam(hnet,i,j)` extracts the network parameter vector (*i*,*j*) from the network parameter object, `hnet`.

`abcd_vector = rfparam(habcd,abcdflag)` extracts the *A*, *B*, *C*, or *D* vector from ABCD-parameter object, `habcd`.

Examples

Create Data Vector From S-Parameter Object

Read in the file default.s2p into an sparameters object and get the S21 value.

```
S = sparameters('default.s2p');
s21 = rfparam(S,2,1)
```

```
s21 = 191×1 complex

  -0.6857 + 1.7827i
  -0.6560 + 1.7980i
  -0.6262 + 1.8131i
  -0.5963 + 1.8278i
  -0.5664 + 1.8422i
  -0.5363 + 1.8563i
  -0.5062 + 1.8700i
  -0.4760 + 1.8835i
```

```
-0.4457 + 1.8966i
-0.4152 + 1.9094i
      ⋮
```

Input Arguments

`abcdflag` — ABCD-parameter index
`'A'` | `'B'` | `'C'` | `'D'`

Flag that determines which ABCD parameters the function extracts, specified as `'A'`, `'B'`, `'C'`, or `'D'`.

`habcd` — 2-port ABCD parameters
ABCD parameter object

2-port ABCD parameters, specified as an RF Toolbox ABCD parameter object. When you specify `abcdflag`, you must also specify an ABCD parameter object.

`hnet` — Network parameters
network parameter object

Network parameters, specified as an RF Toolbox network parameter object.

`i` — Row index
positive integer

Row index of data to extract, specified as a positive integer.

`j` — Column index
positive integer

Column index of data to extract, specified as a positive integer.

Output Arguments

`n_ij` — Network parameters (*i, j*)
vector

Network parameters (*i, j*), returned as a vector. The `i` and `j` input arguments determine which parameters the function returns.

Example: `S_21 = rfparam(hs,2,1)`

`abcd_vector` — *A*, *B*, *C*, or *D*- parameters
vector

A, *B*, *C*, or *D*- parameters, returned as a vector. The `abcdflag` input argument determines which parameters the function returns. The function supports only 2-port ABCD parameters; thus, the output is always a vector.

Example: `a_vector = rfparam(habcd,'A');`

See Also

Introduced before R2006a

rfplot

Plot S-parameter data

Syntax

```
rfplot(s_obj)
rfplot(s_obj,i,j)
rfplot( ___ ,lineSpec)
rfplot( ___ ,plotflag)
hline = rfplot( ___ )
```

Description

`rfplot(s_obj)` plots the magnitude in dB versus frequency of all S-parameters (S_{11}, S_{12} ... S_{NN}) on the current axis. `s_obj` must be an s-parameter object.

`rfplot(s_obj,i,j)` plots the magnitude of $S_{i,j}$, in decibels, versus frequency on the current axis.

`rfplot( ___ ,lineSpec)` plots S-parameters using optional line types, symbols, and colors specified by `linespec`.

`rfplot( ___ ,plotflag)` allows to specify the type of plot by using the `plotflag`.

`hline = rfplot( ___ )` plots the S-parameters and returns the column vector of handles to the line objects, `hline`.

Examples

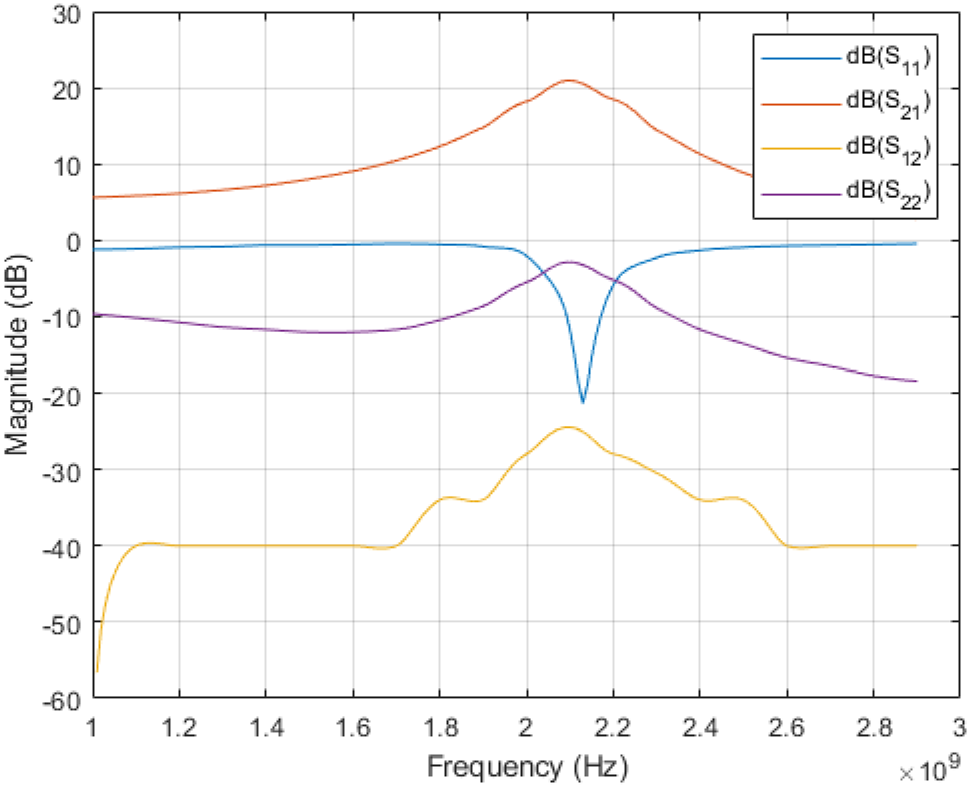
Plot S-Parameter Data Using `rfplot`

Use `sparameters` to create a set S-parameters.

```
hs = sparameters('default.s2p');
```

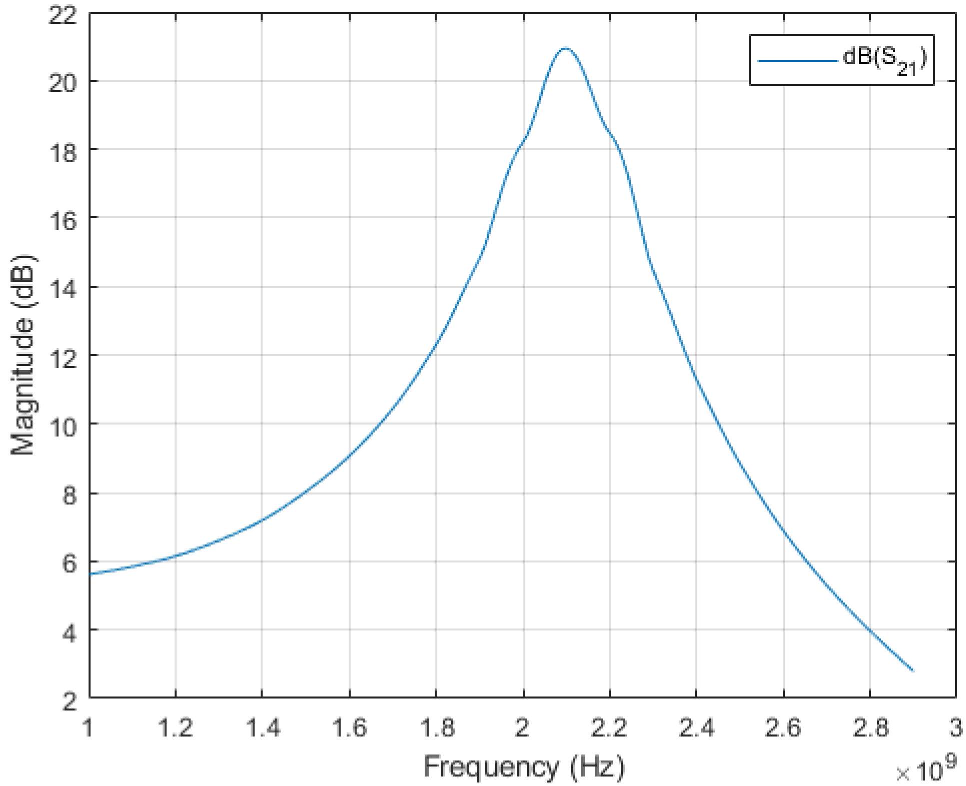
Plot all S-parameters.

```
rfplot(hs)
```



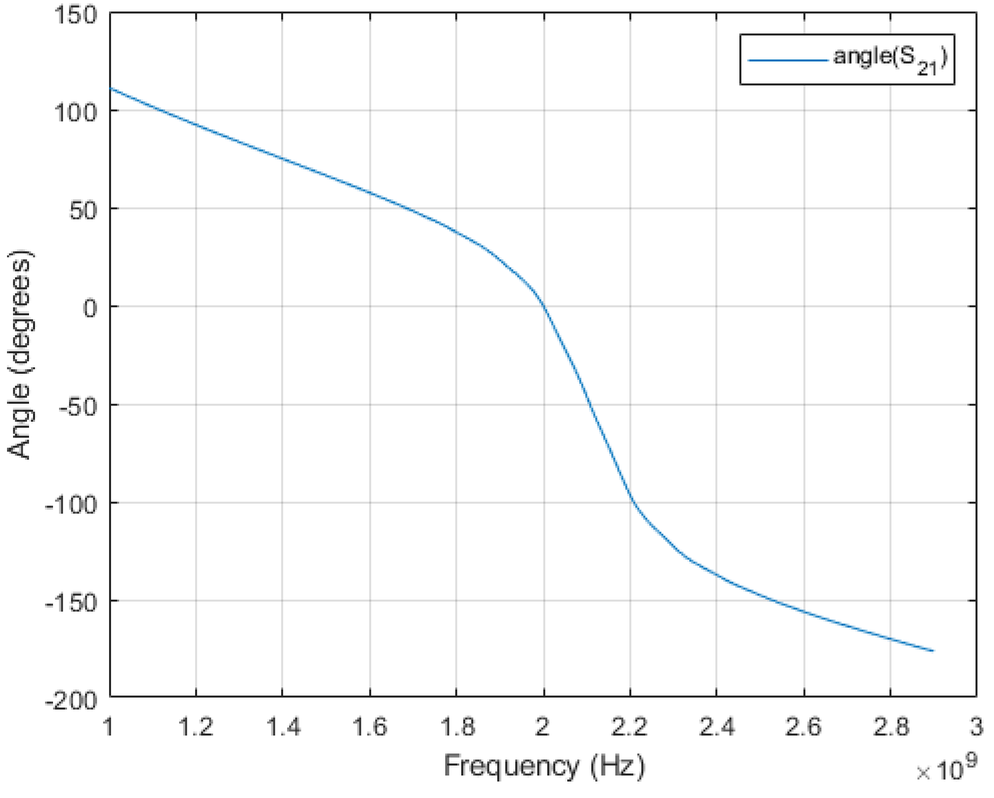
Plot S21.

```
rfplot(hs,2,1)
```



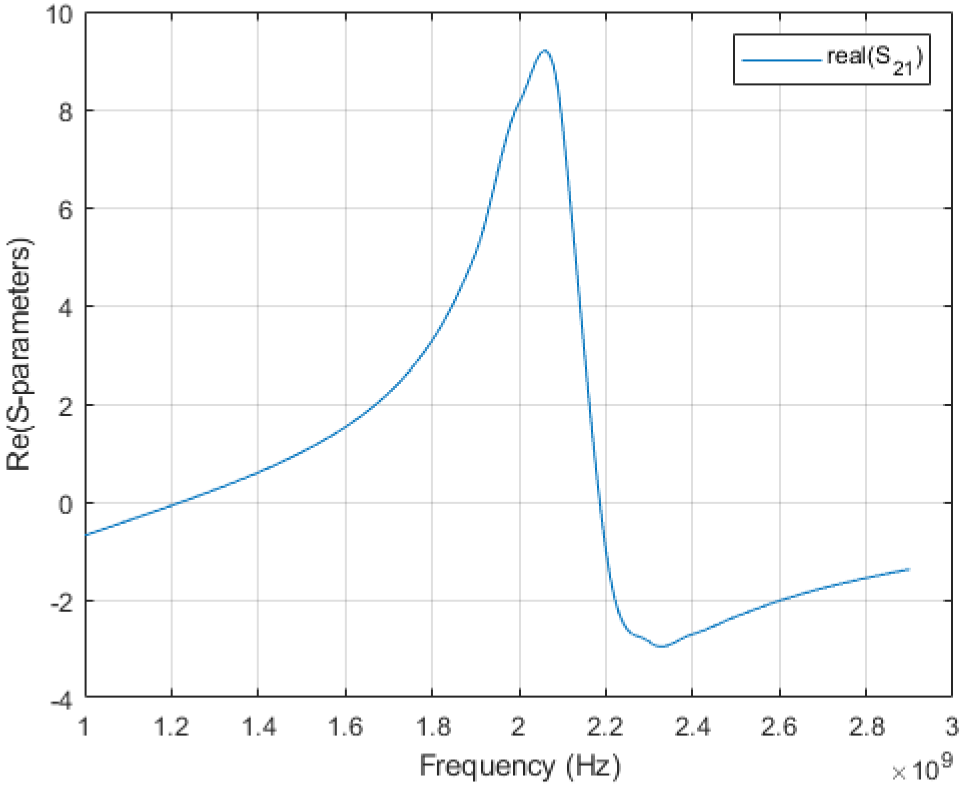
Plot the angle of S21 in degrees.

```
rfplot(hs,2,1,'angle')
```



Plot the real part of S21.

```
rfplot(hs,2,1,'real')
```



Input Arguments

`s_obj` — S-parameters
network parameter object

S-parameters, specified as an RF Toolbox network parameter object. To create this type of object, use the `sparameters` function.

`i` — Row index
positive integer

Row index of data to plot, specified as a positive integer.

j — Column index
positive integer

Column index of data to plot, specified as a positive integer.

lineSpec — Line specification
character array

Line specification, specified as a text input, that modifies the line types, symbols, and colors of the plot. The function takes text inputs in the same format as `plot` command. For more information on line specification values, see `linespec`.

Example: `'-or'`

plotflag — Plot types
`'db'` (default)

Plot types, specified as the following values: `'db'`, `'real'`, `'imag'`, `'abs'`, `'angle'`.

Example: `'angle'`

Output Arguments

hline — Line
line handle

Line containing the S-parameter plot, returned as a line handle.

See Also

`rfinterp1` | `rfparam` | `smith`

Introduced before R2006a

sparameters

S-parameter object

Syntax

```
sobj = sparameters(filename)

sobj = sparameters(data,freq)
sobj = sparameters(data,freq, Z0)

sobj = sparameters(circuitobj,freq)
sobj = sparameters(circuitobj,freq,Z0)

sobj = sparameters(netparamobj)
sobj = sparameters(netparamobj, Z0)

sobj = sparameters(rfdataobj)
sobj = sparameters(rfcktobj)
```

Description

`sobj = sparameters(filename)` creates an S-parameter object `sobj` by importing data from the Touchstone file specified by `filename`.

`sobj = sparameters(data,freq)` creates an S-parameter object from the S-parameter data, `data`, and frequencies, `freq`.

`sobj = sparameters(data,freq, Z0)` creates an S-parameter object from the S-parameter data, `data`, and frequencies, `freq`, with a given reference impedance `Z0`.

`sobj = sparameters(circuitobj,freq)` calculates the S-parameters of a `circuit` object with the default reference impedance.

`sobj = sparameters(circuitobj,freq,Z0)` calculates the S-parameters of a `circuit` object with a given reference impedance `Z0`.

`sobj = sparameters(netparamobj)` converts the network parameter object, `netparamobj`, to S-parameter object with the default reference impedance.

`sobj = sparameters(netparamobj, Z0)` converts the network parameter object, `netparamobj`, to S-parameter object with a given reference impedance, `Z0`.

`sobj = sparameters(rfdataobj)` extracts network data from `rfdataobj` and converts it into S-parameter object.

`sobj = sparameters(rfcktobj)` extracts network data from `rfcktobj` and converts it into S-parameter object.

Examples

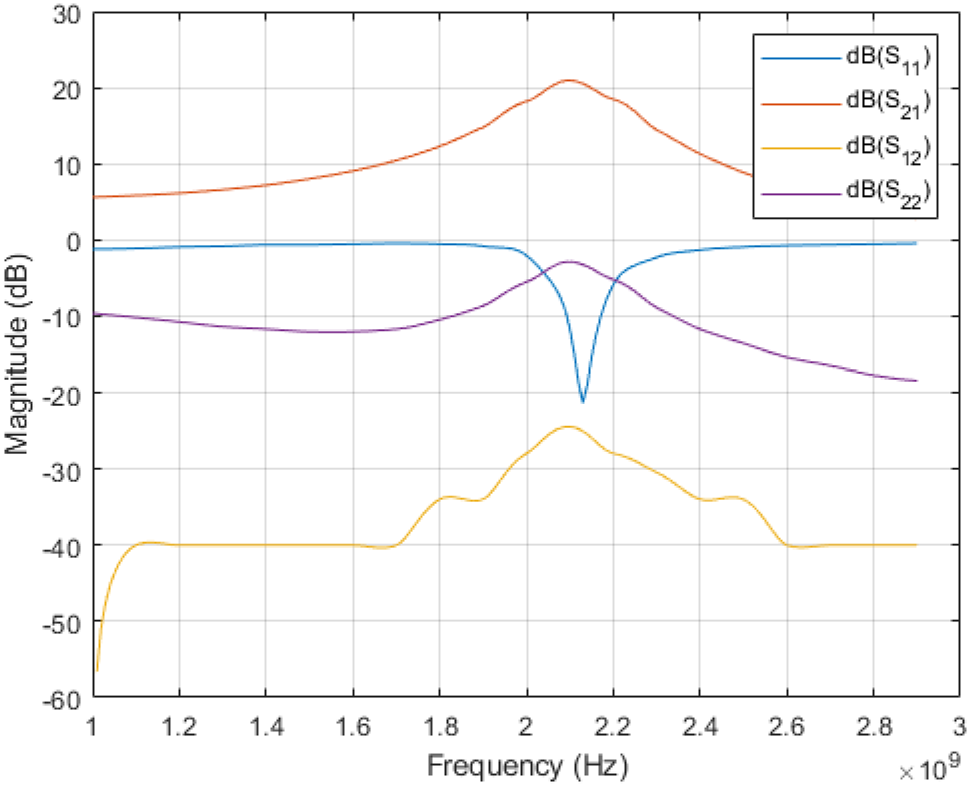
Extract and Plot the S-Parameters of File

Extract S-parameters from file default.s2p and plot it.

```
S = sparameters('default.s2p');
disp(S)
```

```
  sparameters: S-parameters object

       NumPorts: 2
    Frequencies: [191x1 double]
     Parameters: [2x2x191 double]
      Impedance: 50

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

```
rfplot(S)
```



Calculate the S-Parameters of Circuit Object

Create a resistor element R50 and add it to a circuit object `example2` . Calculate the S-parameters of `example2` .

```
hR1 = resistor(50,'R50');
hckt1 = circuit('example2');
add(hckt1,[1 2],hR1)
setports (hckt1, [1 0],[2 0])
freq = linspace (1e3,2e3,100);
S = sparameters(hckt1,freq,100);
disp(S)
```

```
  sparameters: S-parameters object

       NumPorts: 2
    Frequencies: [100x1 double]
     Parameters: [2x2x100 double]
      Impedance: 100

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

Convert Y-Parameters to S-Parameters

Extract Y-parameters from file default.s2p. Convert the resulting Y-parameters to S-parameters.

```
Y1 = yparameters('default.s2p');
S1 = sparameters(Y1,100);
disp(Y1)
disp(S1)
```

```
  yparameters: Y-parameters object

       NumPorts: 2
    Frequencies: [191x1 double]
     Parameters: [2x2x191 double]

  rfparam(obj,i,j) returns Y-parameter Yij

  sparameters: S-parameters object

       NumPorts: 2
    Frequencies: [191x1 double]
     Parameters: [2x2x191 double]
      Impedance: 100

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

Convert RF Data Object to S-parameters

```
file = 'default.s2p';
h = read(rfdata.data, file);
S = sparameters(h)
```

```
S = 
  sparameters: S-parameters object

       NumPorts: 2
    Frequencies: [191x1 double]
     Parameters: [2x2x191 double]
      Impedance: 50.0000 + 0.0000i

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

- “Bisect S-Parameters of Cascaded Probes”

Input Arguments

data — S-parameter data
array of complex numbers

S-parameter data, specified as an array of complex numbers, of size *N*-by-*N*-by-*K*.

circuitobj — Circuit object
circuit object

Circuit object. The function uses this input argument to calculate the S-parameters of the circuit object.

netparamobj — Network parameter object
network parameter object

Network parameter object. The network parameter objects are of the type: `sparameters`, `yparameters`, `zparameters`, `abcdparameters`, `gparameters`, `hparameters`, and `tparameters`.

Example: `S1 = sparameters(Y1,100)` . `Y1` is a parameter object. This example converts Y-parameters to S-parameters at 100 ohms.

filename — Touchstone data file
character vector

Touchstone data file, specified as a character vector, that contains network parameter data. `filename` can be the name of a file on the MATLAB path or the full path to a file.

Example: `sobj = sparameters('defaultbandpass.s2p');`

freq — S-parameter frequencies
vector of positive real numbers

S-parameter frequencies, specified as a vector of positive real numbers, sorted from smallest to largest.

Z0 — Reference impedance
`50` (default) | positive real scalar

Reference impedance in ohms, specified as a positive real scalar. You cannot specify `Z0` if you are importing data from a file. The argument Z0 is optional and is stored in the `Impedance` property.

rfdataobj — RF data object
RF data object.

RF data object. Specify `rfdataobj` as either `rfdata.data`, or `rfdata.network` object.

rfcktobj — RF Network object
RF network object

RF network object. Specify `rfcktobj` as any analyzed `rfckt` type object, such as `rfckt.amplifier`, `rkckt.cascade` object.

Output Arguments

sobj — S-parameter data
S-parameter object

S-parameter data, returned as an object. `disp(sobj)` returns the properties of the object:

- `NumPorts` — Number of ports, specified as an integer. The function calculates this value automatically when you create the object.

- `Frequencies` — S-parameter frequencies, specified as a *K*-by-1 vector of positive real numbers sorted from smallest to largest. The function sets this property from the `filename` or `freq` input arguments.
- `Parameters` — S-parameter data, specified as an *N*-by-*N*-by-*K* array of complex numbers. The function sets this property from the `filename` or `data` input arguments.
- `Impedance` — Reference impedance in ohms, specified as a positive real scalar. The function sets this property from the `filename` or `Z0` input arguments. If no reference impedance is provided, the function uses a default value of `50`.

See Also

`abcdparameters` | `circuit` | `gparameters` | `hparameters` | `rfparam` | `rfplot` | `smithplot` | `yparameters` | `zparameters`

Topics

"Bisect S-Parameters of Cascaded Probes"

Introduced in R2012a

abcdparameters

Create ABCD parameter object

Syntax

`habcd = abcdparameters(filename)`

`habcd = abcdparameters(hnet)`
`habcd = abcdparameters(data,freq)`

`habcd = abcdparameters(rftbxobj)`

Description

`habcd = abcdparameters(filename)` creates an ABCD parameter object `habcd` by importing data from the Touchstone file specified by `filename`. All data is stored in real/imag format.

`habcd = abcdparameters(hnet)` creates an ABCD parameter object from the RF Toolbox network parameter object `hnet`.

`habcd = abcdparameters(data,freq)` creates an ABCD parameter object from the ABCD parameter data, `data`, and frequencies, `freq`.

`habcd = abcdparameters(rftbxobj)` extracts network data from `rftbxobj` and converts it into ABCD-parameter data.

Examples

Read a File as ABCD-parameters and Extract A

Read the file `default.s2p` as abcd-parameters.

```
abcd = abcdparameters('default.s2p')
```

```
abcd =
  abcdparameters: ABCD-parameters object

        NumPorts: 2
     Frequencies: [191x1 double]
      Parameters: [2x2x191 double]

  rfparam(obj,specifier) returns specified ABCD-parameter 'A', 'B', 'C', or 'D'
```

Extract parameter A.

```
A = rfparam(abcd,'A')
```

```
A = 191×1 complex

  -0.1470 - 0.0698i
  -0.1421 - 0.0698i
  -0.1373 - 0.0696i
  -0.1325 - 0.0694i
  -0.1277 - 0.0691i
  -0.1231 - 0.0688i
  -0.1185 - 0.0683i
  -0.1140 - 0.0678i
  -0.1097 - 0.0672i
  -0.1054 - 0.0666i
      ⋮
```

Input Arguments

data — ABCD parameter data
array of complex numbers

ABCD parameter data, specified as an array of complex numbers, of size 2*N*-by-2*N*-by-*K*. The function uses this input argument to set the value of the `Parameters` property of `habcd`.

`filename` — Touchstone data file
character vector

Touchstone data file, specified as a character vector, that contains network parameter data. `filename` can be the name of a file on the MATLAB path or the full path to a file.

Example: `habcd = abcdparameters('defaultbandpass.s2p');`

`freq` — ABCD parameter frequencies

vector of positive numbers

ABCD parameter frequencies, specified as a vector of positive real numbers, sorted from smallest to largest. The function uses this input argument to set the value of the `Frequencies` property of `habcd`.

`hnet` — Network parameter data

scalar handle

Network parameter data, specified as a scalar handle. If `hnet` is an ABCD parameter object, then `habcd` is a deep copy of `hnet`. Otherwise, the function performs a network parameter conversion to create `habcd`. When converting network parameters, the same restrictions apply as those for RF Toolbox network parameter data conversion functions:

- ABCD parameter objects support 2*N*-port data.
- Hybrid-g parameter objects support 2-port data.
- Hybrid parameter objects support 2-port data.
- S-parameter objects support *N*-port data.
- Y-parameter objects support *N*-port data.
- Z-parameter objects support *N*-port data.

`rftbxobj` — network object

scalar handle

Network object, specified as a scalar handle. Specify `rftbxobj` as one of the following types: `rfdata.data`, `rfdata.network`, and any analyzed `rfckt` type.

Output Arguments

`habcd` — ABCD parameter data

scalar handle

ABCD parameter data, returned as a scalar handle. `disp(habcd)` returns the properties of the object:

- `NumPorts` — Number of ports, specified as an integer. The function calculates this value automatically when you create the object.
- `Frequencies` — ABCD parameter frequencies, specified as a K- by- 1 vector of positive real numbers sorted from smallest to largest. The function sets this property from the `filename` or `freq` input arguments.
- `Parameters` — ABCD parameter data, specified as a 2N-by-2N-by-K array of complex numbers. The function sets this property from the `filename` or `data` input arguments.

See Also

`gparameters` | `hparameters` | `rfparam` | `sparameters` | `yparameters` | `zparameters`

Introduced in R2012b

gparameters

Create hybrid-g parameter object

Syntax

```
hg = gparameters(filename)
```

```
hg = gparameters(hnet)
hg = gparameters(data,freq)
```

```
hg = gparameters(rftbxobj)
```

Description

`hg = gparameters(filename)` creates a hybrid-g parameter object `hg` by importing data from the Touchstone file specified by `filename`. All data is stored in real/imag format.

`hg = gparameters(hnet)` creates a hybrid-g parameter object from the RF Toolbox network parameter object `hnet`.

`hg = gparameters(data,freq)` creates a hybrid-g parameter object from the g-parameter data, `data`, and frequencies, `freq`.

`hg = gparameters(rftbxobj)` extracts network data from `rftbxobj` and converts it into G-parameter data.

Examples

Extract G11

Read the file `default.s2p` as g-parameters and extract G11.

```
g = gparameters('default.s2p')
```

```
g =
  gparameters: g-parameters object

       NumPorts: 2
    Frequencies: [191x1 double]
     Parameters: [2x2x191 double]

  rfparam(obj,i,j) returns g-parameter gij
```

```
g11 = rfparam(g,1,1);
```

Input Arguments

data — Hybrid-g parameter data
array of complex numbers

Hybrid-g parameter data, specified as an array of complex numbers, of size 2*N*-by-2*N*-by-*K*. The function uses this input argument to set the value of the `Parameters` property of `hg`.

filename — Touchstone data file
character vector

Touchstone data file, specified as a character vector, that contains network parameter data. `filename` can be the name of a file on the MATLAB path or the full path to a file.

Example: `hg = gparameters('defaultbandpass.s2p');`

freq — Hybrid-g parameter frequencies
vector of positive scalars

Hybrid-g parameter frequencies, specified as a vector of positive real numbers, sorted from smallest to largest. The function uses this input argument to set the value of the `Frequencies` property of `hg`.

hnet — Network parameter data
scalar handle

Network parameter data, specified as a scalar handle. If `hnet` is a hybrid-g parameter object, then `hg` is a deep copy of `hnet`. Otherwise, the function performs a network parameter conversion to create `hg`. When converting network parameters, the same restrictions apply as those for RF Toolbox network parameter data conversion functions:

- ABCD parameter objects support 2*N*-port data.
- Hybrid-g parameter objects support 2-port data.
- Hybrid parameter objects support 2-port data.
- S-parameter objects support *N*-port data.
- Y-parameter objects support *N*-port data.
- Z-parameter objects support *N*-port data.

`rftbxobj` — network object
scalar handle

Network object, specified as a scalar handle. Specify `rftbxobj` as one of the following types: `rfdata.data`, `rfdata.network`, and any analyzed `rfckt` type.

Output Arguments

hg — Hybrid-g parameter data
scalar handle

Hybrid-g parameter data, returned as a scalar handle. `disp(hg)` returns the properties of the object:

- `Frequencies` — Hybrid-g parameter frequencies, specified as a *K*-by-1 vector of positive real numbers sorted from smallest to largest. The function sets this property from the `filename` or `freq` input arguments.
- `Parameters` — Hybrid-g parameter data, specified as an *N*-by-*N*-by-*K* array of complex numbers. The function sets this property from the `filename` or `data` input arguments.

See Also

`abcdparameters` | `hparameters` | `rfparam` | `sparameters` | `yparameters` | `zparameters`

Introduced in R2012b

hparameters

Create hybrid parameter object

Syntax

```
hh = hparameters(filename)
```

```
hh = hparameters(hnet)
hh = hparameters(data,freq)
```

```
hh = hparameters(rftbxobj)
```

Description

`hh = hparameters(filename)` creates a hybrid parameter object `hh` by importing data from the Touchstone file specified by `filename`. All data is stored in real/imag format.

`hh = hparameters(hnet)` creates a hybrid parameter object from the RF Toolbox network parameter object `hnet`.

`hh = hparameters(data,freq)` creates a hybrid parameter object from the hybrid parameter data, `data`, and frequencies, `freq`.

`hh = hparameters(rftbxobj)` extracts network data from `rftbxobj` and converts it into H-parameter data.

Examples

Extract H11

Read the file `default.s2p` as h-parameters and extract H11.

```
h = hparameters('default.s2p')
```

```
h =
  hparameters: h-parameters object

       NumPorts: 2
    Frequencies: [191x1 double]
     Parameters: [2x2x191 double]

  rfparam(obj,i,j) returns h-parameter hij
```

```
h11 = rfparam(h,1,1);
```

Input Arguments

data — Hybrid parameter data
array of complex numbers

Hybrid parameter data, specified as array of complex numbers, of size 2-by-2-by-*K*. The function uses this input argument to set the value of the `Parameters` property of `hh`.

filename — Touchstone data file
character vector

Touchstone data file, specified as a character vector, that contains network parameter data. `filename` can be the name of a file on the MATLAB path or the full path to a file.

Example: `hh = hparameters('defaultbandpass.s2p');`

freq — Hybrid parameter frequencies
vector of positive numbers

Hybrid parameter frequencies, specified as a vector of positive real numbers, sorted from smallest to largest. The function uses this input argument to set the value of the `Frequencies` property of `hh`.

hnet — Network parameter data
scalar handle

Network parameter data, specified as a scalar handle. If `hnet` is a hybrid parameter object, then `hh` is a deep copy of `hnet`. Otherwise, the function performs a network parameter conversion to create `hh`. When converting network parameters, the same restrictions apply as those for RF Toolbox network parameter data conversion functions:

- ABCD parameter objects support 2*N*-port data.
- Hybrid-g parameter objects support 2-port data.
- Hybrid parameter objects support 2-port data.
- S-parameter objects support *N*-port data.
- Y-parameter objects support *N*-port data.
- Z-parameter objects support *N*-port data.

`rftbxobj` — network object
scalar handle

Network object, specified as scalar handle. Specify `rftbxobj` as one of the following types: `rfdata.data`, `rfdata.network`, and any analyzed `rfckt` type.

Output Arguments

hh — Hybrid parameter data
scalar handle

Hybrid parameter data, returned as a scalar handle. `disp(hh)` returns the properties of the object:

- `Frequencies` — Hybrid parameter frequencies, specified as a *K*-by-1 vector of positive real numbers sorted from smallest to largest. The function sets this property from the `filename` or `freq` input arguments.
- `Parameters` — Hybrid parameter data, specified as a 2-by-2-by-*K* array of complex numbers. The function sets this property from the `filename` or `data` input arguments.

See Also

`abcdparameters` | `gparameters` | `rfparam` | `sparameters` | `yparameters` | `zparameters`

Introduced in R2012b

yparameters

Create Y-parameter object

Syntax

```
hy = yparameters(filename)
```

```
hy = yparameters(hnet)
hy = yparameters(data,freq)
```

```
hy = yparameters(rftbxobj)
```

Description

`hy = yparameters(filename)` creates a Y-parameter object `hy` by importing data from the Touchstone file specified by `filename`. All data is stored in real/imag format.

`hy = yparameters(hnet)` creates a Y-parameter object from the RF Toolbox network parameter object `hnet`.

`hy = yparameters(data,freq)` creates a Y-parameter object from the Y-parameter data, `data`, and frequencies, `freq`.

`hy = yparameters(rftbxobj)` extracts network data from `rftbxobj` and converts it into y-parameter data.

Examples

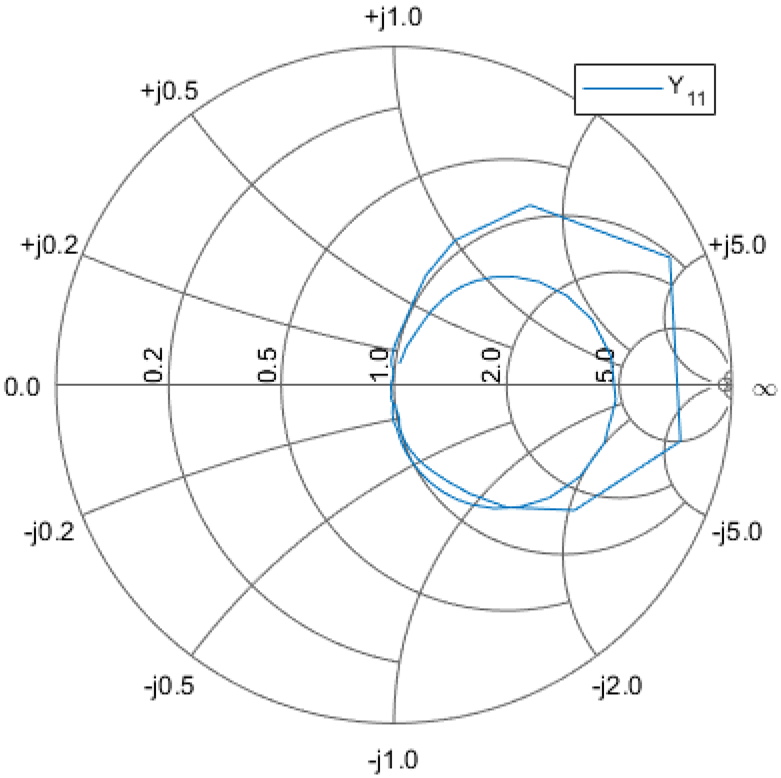
Plot Y-Parameters on Smith Chart

Extract y-parameters from `default.s2p` and plot on a smith chart.

```
Y = yparameters('default.s2p')
```

```
Y =
  yparameters: Y-parameters object

       NumPorts: 2
    Frequencies: [191x1 double]
     Parameters: [2x2x191 double]

  rfparam(obj,i,j) returns Y-parameter Yij
```

```
figure;
smith(Y,1,1)
```



Input Arguments

`data` — Y-parameter data
array of complex numbers

Y-parameter data, specified as an array of complex numbers, of size *N*-by-*N*-by-*K*. The function uses this input argument to set the value of the `Parameters` property of `hy`.

`filename` — Touchstone data file
character vector

Touchstone data file, specified as a character vector, that contains network parameter data. `filename` can be the name of a file on the MATLAB path or the full path to a file.

Example: `hy = yparameters('defaultbandpass.s2p');`

`freq` — Y-parameter frequencies
vector of positive numbers

Y-parameter frequencies, specified as a vector of positive real numbers, sorted from smallest to largest. The function uses this input argument to set the value of the `Frequencies` property of `hy`.

`hnet` — Network parameter data
scalar handle

Network parameter data, specified as a scalar handle. If `hnet` is a Y-parameter object, then `hy` is a deep copy of `hnet`. Otherwise, the function performs a network parameter conversion to create `hy`. When converting network parameters, the same restrictions apply as those for RF Toolbox network parameter data conversion functions:

- ABCD parameter objects support 2*N*-port data.
- Hybrid-g parameter objects support 2-port data.
- Hybrid parameter objects support 2-port data.
- S-parameter objects support *N*-port data.
- Y-parameter objects support *N*-port data.
- Z-parameter objects support *N*-port data.

`rftbxobj` — network object
scalar

Network object, specified as scalar handle. Specify `rftbxobj` as one of the following types: `rfdata.data`, `rfdata.network`, and any analyzed `rfckt` type.

Output Arguments

hy — Y-parameter data
scalar handle

Y-parameter data, returned as a scalar handle. `disp(hy)` returns the properties of the object:

- `NumPorts` — Number of ports, specified as an integer. The function calculates this value automatically when you create the object.
- `Frequencies` — Y-parameter frequencies, specified as a *K*-by-1 vector of positive real numbers sorted from smallest to largest. The function sets this property from the `filename` or `freq` input arguments.
- `Parameters` — Y-parameter data, specified as an *N*-by-*N*-by-*K* array of complex numbers. The function sets this property from the `filename` or `data` input arguments.

See Also

`abcdparameters` | `gparameters` | `hparameters` | `rfparam` | `sparameters` | `zparameters`

Introduced in R2012b

zparameters

Create Z-parameter object

Syntax

```
hz = zparameters(filename)
```

```
hz = zparameters(hnet)
hz = zparameters(data,freq)
```

```
hz = zparameters(rftbxobj)
```

Description

`hz = zparameters(filename)` creates a Z-parameter object `hz` by importing data from the Touchstone file specified by `filename`. All data is stored in real/imag format.

`hz = zparameters(hnet)` creates a Z-parameter object from the RF Toolbox network parameter object `hnet`.

`hz = zparameters(data,freq)` creates a Z-parameter object from the Z-parameter data, `data`, and frequencies, `freq`.

`hz = zparameters(rftbxobj)` extracts network data from `rftbxobj` and converts it into z-parameter data.

Examples

Extract and Plot Imaginary Part of Z11

Read the file `default.s2p` as z-parameters and extract Z11.

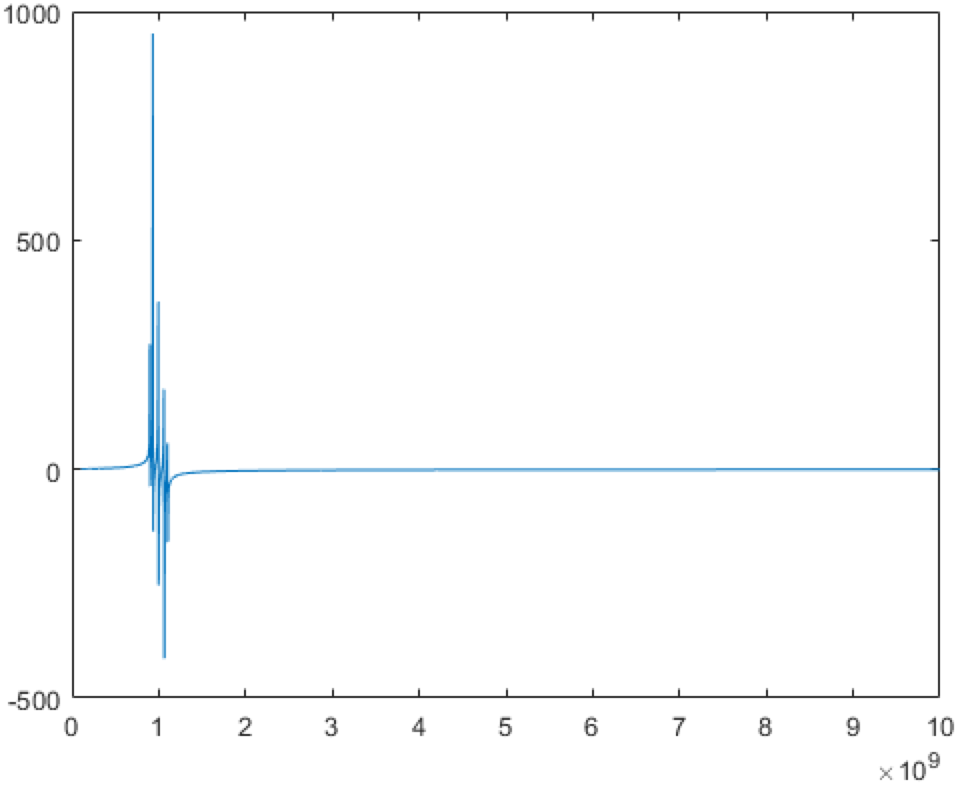
```
Z = zparameters('defaultbandpass.s2p')
```

```
Z = 
  zparameters: Z-parameters object

       NumPorts: 2
    Frequencies: [1000x1 double]
     Parameters: [2x2x1000 double]

  rfparam(obj,i,j) returns Z-parameter Zij
```

```
z11 = rfparam(Z,1,1);
```

Plot imaginary part of Z11.

```
 plot(Z.Frequencies, imag(z11))
```



Input Arguments

`data` — Z-parameter data
array of complex numbers

Z-parameter data, specified as an array of complex numbers, of size *N*-by-*N*-by-*K*. The function uses this input argument to set the value of the `Parameters` property of `hz`.

`filename` — Touchstone data file that contains network parameter data
character vector

Touchstone data file, specified as a character vector. `filename` can be the name of a file on the MATLAB path or the full path to a file.

Example: `hz = zparameters('defaultbandpass.s2p');`

`freq` — Z-parameter frequencies
vector of positive numbers

Z-parameter frequencies, specified as a vector of positive real numbers, sorted from smallest to largest. The function uses this input argument to set the value of the `Frequencies` property of `hz`.

`hnet` — Network parameter data
scalar handle

Network parameter data, specified as a scalar handle. If `hnet` is a Z-parameter object, then `hz` is a deep copy of `hnet`. Otherwise, the function performs a network parameter conversion to create `hz`. When converting network parameters, the same restrictions apply as those for RF Toolbox network parameter data conversion functions:

- ABCD parameter objects support 2*N*-port data.
- Hybrid-g parameter objects support 2-port data.
- Hybrid parameter objects support 2-port data.
- S-parameter objects support *N*-port data.
- Y-parameter objects support *N*-port data.
- Z-parameter objects support *N*-port data.

`rftbxobj` — network object
scalar

Network object, specified as scalar handle. Specify `rftbxobj` as one of the following types: `rfdata.data`, `rfdata.network`, and any analyzed `rfckt` type.

Output Arguments

`hz` — Z-parameter object
scalar handle

Z-parameter data, returned as a scalar handle. `disp(hz)` returns the properties of the object:

- `NumPorts` — Number of ports, specified as an integer. The function calculates this value automatically when you create the object.
- `Frequencies` — Z-parameter frequencies, specified as a *K*-by-1 vector of positive real numbers sorted from smallest to largest. The function sets this property from the `filename` or `freq` input arguments.
- `Parameters` — Z-parameter data, specified as an *N*-by-*N*-by-*K* array of complex numbers. The function sets this property from the `filename` or `data` input arguments.

See Also

`abcdparameters` | `gparameters` | `hparameters` | `rfparam` | `sparameters` | `yparameters`

Introduced in R2012b

tparameters

Create T-parameter object

Syntax

```
tobj = tparameters(filename)
tobj = tparameters(tobj_old,z0)

tobj = tparameters(rftbx_obj)
tobj = tparameters(hnet, z0)
tobj = tparameters(paramdata,freq,z0)
```

Description

`tobj = tparameters(filename)` creates a T-parameter object, `ht` by importing data from the Touchstone file specified by `filename`. All data is stored in real/imag format.

`tobj = tparameters(tobj_old,z0)` converts a T-parameter data in `tobj_old` to the new impedance z_0. z_0 is optional, and if not provided, `tparam_data` is copied instead of converted.

`tobj = tparameters(rftbx_obj)` extracts S-parameter network data from `rfdata.network` object, an `rfdata.data` object, or any analyzed network object, and then converts the data to T-parameter data.

`tobj = tparameters(hnet, z0)` converts the network parameter data in `hnet` into T-parameter data.

`tobj = tparameters(paramdata,freq,z0)` creates T-parameter object directly from the specified data, `paramdata` using specified frequency and impedance.

Examples

Convert File to T-Parameters

Read S-parameter data from a Touchstone file and convert the data to T-parameters

```
T1 = tparameters('passive.s2p');
disp(T1)
```

```
  tparameters: T-parameters object

       NumPorts: 2
    Frequencies: [202x1 double]
     Parameters: [2x2x202 double]
      Impedance: 50

  rfparam(obj,i,j) returns T-parameter Tij
```

Change Impedance of T-Parameters

Change the impedance of T-parameters to 100 ohms.

```
T1 = tparameters('passive.s2p');
disp(T1)
```

```
  tparameters: T-parameters object

       NumPorts: 2
    Frequencies: [202x1 double]
     Parameters: [2x2x202 double]
      Impedance: 50

  rfparam(obj,i,j) returns T-parameter Tij
```

```
T2 = tparameters(T1,100);
disp(T2)
```

```
  tparameters: T-parameters object

       NumPorts: 2
    Frequencies: [202x1 double]
     Parameters: [2x2x202 double]
      Impedance: 100

  rfparam(obj,i,j) returns T-parameter Tij
```

Input Arguments

`tobj_old` — T-parameter object
scalar handle

T-parameter object, specified as a scalar handle.

`paramdata` — Input T-parameter data
2-by-2-by-*K* array of complex numbers

Input T-parameter data, specified as 2-by-2-by-*K* array of complex numbers. The function uses this input argument to set the value of the `Parameters` property of `ht`.

`filename` — Touchstone data file
character vector

Touchstone data file, specified as a character vector. `filename` can be the name of a file on the MATLAB path or the full path to a file.

Example: `ht = tparameters('defaultbandpass.s2p');`

`freq` — T-parameter frequencies
vector of positive real numbers

T-parameter frequencies, specified as a vector of positive real numbers. The frequencies are sorted from smallest to largest. The function uses this input argument to set the value of the `Frequencies` property of `ht`.

`z0` — T-parameter impedance
50 (default) | scalar

T-parameter impedance, specified as a scalar. z_0 is optional and is stored in the `Impedance`.

`hnet` — Network parameter data
scalar handle

Network parameter data, specified as a scalar handle. If `hnet` is a T-parameter object, then `tobj` is a deep copy of `hnet`. Otherwise, the function performs a network parameter conversion to create `tobj`. Specify `hnet` as one of the following types: `sparameters`, `yparameters`, `gparameters`, `hparameters`, `zparameters`, or `abcdparameters`.

rftbx_obj — network object
scalar handle

Network object, specified as a scalar handle. You can specify rftbxobj as one of the following types: rfdata.data object, rfdata.network object, or as any analyzed rfckt type.

Output Arguments

tobj — T-parameter object
scalar handle

T-parameter data, returned as a scalar handle. disp(ht) returns the properties of the object:

- NumPorts — Number of ports, specified as an integer. The function calculates this value automatically when you create the object.
- Parameters — T-parameter data, specified as a 2-by-2-by-*K* array of complex numbers. The 2x2 T-parameter data is specified for each frequency in the "Frequencies" property. The function sets this property from the filename or paramdata input arguments.
- Impedance — Characteristic impedance used to measure the T-Parameters, specified as a numeric positive real scalar.
- Frequencies — T-parameter frequencies, specified as a *K*-by-1 vector of positive real numbers sorted from smallest to largest. The function sets this property from the filename or freq input arguments.

See Also

abcdparameters | gparameters | hparameters | rfparam | sparameters | yparameters | zparameters

Introduced in R2015a

add

Insert circuit element or circuit object into circuit

Syntax

```
add(cktobj,cktnodes,elem)
add(cktobj,cktnodes,elem,termorder)
```

```
elem = add(______)
```

Description

`add(cktobj,cktnodes,elem)` inserts a circuit element `elem` into a circuit object `cktobj`. The terminals of the `elem` are attached to the nodes specified in `cktnodes`. If `elem` is a Touchstone file name or s-parameters object, an `nport` object is created from input and added to `cktobj`.

`add(cktobj,cktnodes,elem,termorder)` the terminals specified in `termorder` are attached to circuit nodes specified in `cktnodes`.

`elem = add(______)` returns `elem` as an output.

Examples

Add Element to Circuit

Create a resistor, and add it to a circuit.

```
hR1 = resistor(50);
hckt1 = circuit('new_circuit1');
add(hckt1,[1 2],hR1)
disp(hR1)
```

```
  resistor: Resistor element
```

```
     Resistance: 50
           Name: 'R'
      Terminals: {'p'  'n'}
    ParentNodes: [1 2]
     ParentPath: 'new_circuit1'
```

```
disp(hckt1)
```

```
  circuit: Circuit element

    ElementNames: {'R'}
           Nodes: [1 2]
            Name: 'new_circuit1'
```

Add Element to Specific Nodes of Circuit

Create a capacitor.

```
hC2 = capacitor(1e-10)
```

```
hC2 =
  capacitor: Capacitor element

    Capacitance: 1.0000e-10
           Name: 'C'
      Terminals: {'p'  'n'}
```

```
disp(hC2)
```

```
  capacitor: Capacitor element

    Capacitance: 1.0000e-10
           Name: 'C'
      Terminals: {'p'  'n'}
```

Connect terminal **n** of the capacitor to node 3 and terminal **p** of the capacitor to node 4.

```
hckt2 = circuit('new_circuit2');
add(hckt2,[3 4],hC2,{'n' 'p'})
disp(hckt2)
```

```
  circuit: Circuit element
```

```
    ElementNames: {'C'}
           Nodes: [3 4]
            Name: 'new_circuit2'
```

Create and Insert Element in Circuit

Create a circuit.

```
hckt3 = circuit('new_circuit3')
```

```
hckt3 = 
  circuit: Circuit element

    ElementNames: {}
           Nodes: []
            Name: 'new_circuit3'
```

Insert an inductor into the circuit using the add function.

```
hL3 = add(hckt3,[100 200],inductor(1e-9));
disp(hckt3)
```

```
  circuit: Circuit element

    ElementNames: {'L'}
           Nodes: [100 200]
            Name: 'new_circuit3'
```

Add Two Circuits Together

Create circuit 1 and set the terminals using the **setterminals** functions.

```
hckt1 = circuit('circuit_new1');
add(hckt1,[1 2], resistor(100));
setterminals(hckt1, [1 2]);
disp(hckt1);
```

```
  circuit: Circuit element
```

```
    ElementNames: {'R'}
           Nodes: [1 2]
            Name: 'circuit_new1'
       Terminals: {'t1'  't2'}
```

Create circuit 2 and set the terminals.

```
hckt2 = circuit('circuit_new2');
add(hckt2, [3 4], capacitor(1.5e-9));
setterminals(hckt2, [3 4]);
disp(hckt2);
```

```
  circuit: Circuit element

    ElementNames: {'C'}
           Nodes: [3 4]
            Name: 'circuit_new2'
       Terminals: {'t1'  't2'}
```

Add the two circuits.

```
add(hckt1, [2 4], hckt2);
disp(hckt2)
```

```
  circuit: Circuit element

    ElementNames: {'C'}
           Nodes: [3 4]
            Name: 'circuit_new2'
       Terminals: {'t1'  't2'}
     ParentNodes: [2 4]
      ParentPath: 'circuit_new1'
```

```
disp(hckt1)
```

```
  circuit: Circuit element

    ElementNames: {'R'  'circuit_new2'}
           Nodes: [1 2 4]
            Name: 'circuit_new1'
       Terminals: {'t1'  't2'}
```

Input Arguments

`cktobj` — Circuit object
scalar handle object

Circuit object into which the circuit element is inserted, specified as scalar handle object. This circuit object can be a new circuit or a nport object, or an already existing circuit.

`cktnodes` — Circuit nodes
vector of integers

Circuit nodes of the circuit object, specified as vector of integers. The function uses this input argument to attach the new element to the circuit.

`elem` — Circuit elements
scalar handle objects

Circuit elements that are inserted into the circuit object, specified as scalar handle objects. The element can be a resistor, capacitor, inductor, Touchstone file name, s-parameter object, or an entire circuit.

`termorder` — Element terminals
scalar handle objects

Element terminals, which are the same scalar handle objects found in `Terminals` property of `elem`. These input arguments are specified as scalar handle objects. .

Output Arguments

`elem` — Circuit elements
scalar handle objects

Circuit elements, which are returned as scalar handle objects, after using the add function. The function uses any or all of the input arguments to create these circuit element.

See Also

`clone` | `setports` | `setterminals` | `sparameters`

Introduced in R2013b

setports

Set ports of circuit object

Syntax

```
setports(cktobj,nodepair_1,......,nodepair_n)
setports(cktobj,nodepair_1,......,nodepair_n,portnames)
```

Description

`setports(cktobj,nodepair_1,......,nodepair_n)` defines the node_pairs in an N-port `cktobj` as ports using `nodepair_1,......,nodepair_n`. This syntax then assigns the ports default names. It also defines the terminals of a `cktobj`, taking the terminal names from the port names. If any of the node pairs do not exist, `setports` creates it.

`setports(cktobj,nodepair_1,......,nodepair_n,portnames)` defines the node_pairs in an N-port `cktobj` as ports using `nodepair_1,......,nodepair_n`. After defining the ports, this syntax names them using `portnames`. The length of the `portnames` must equal to the number of node_pairs in the circuit.

Examples

Create 1-Port Circuit with Default Names

Create a 1-port circuit using setports.

```
hckt1 = circuit('new_circuit1');
add(hckt1,[1 2],resistor(50))
setports(hckt1,[1 2])
disp(hckt1)
```

```
  circuit: Circuit element
```

```
    ElementNames: {'R'}
           Nodes: [1 2]
            Name: 'new_circuit1'
        NumPorts: 1
       Terminals: {'p1+'  'p1-'}
```

Create 2-Port Circuit and Assign Port Names

Create a circuit and define two ports. Name the ports **in** and **out**.

```
hckt2 = circuit('example_circuit2');
add(hckt2,[2 3],resistor(50))
add(hckt2,[3 1],capacitor(1e-9))
setports(hckt2,[2 1],[3 1],{'in' 'out'})
disp(hckt2)
```

```
  circuit: Circuit element

    ElementNames: {'R'  'C'}
           Nodes: [1 2 3]
            Name: 'example_circuit2'
        NumPorts: 2
       Terminals: {'in+'  'out+'  'in-'  'out-'}
```

Input Arguments

`cktobj` — Circuit Object
scalar handle object

Circuit object for which the ports are defined, specified as scalar handle objects.

`nodepair_1,......,nodepair_n` — Node pairs
vector of integers

Node pairs of the circuit object, specified as vector of integers. The function uses this input argument to define the ports.

`portnames` — Port names
character vector

Names to name the ports defined for the circuit object, specified as character vector.

See Also

`add` | `clone` | `setterminals` | `sparameters`

Introduced in R2013b

setterminals

Set terminals of circuit object

Syntax

```
setterminals(cktobj,cktnodes)
setterminals(cktobj,cktnodes,termnames)
```

Description

`setterminals(cktobj,cktnodes)` defines the nodes in a `cktobj` as terminals using `cktnodes`. It then gives the terminals default names.

`setterminals(cktobj,cktnodes,termnames)` defines the nodes in a `cktobj` as terminals `cktnodes`. It then names the terminals using `termnames`. `cktnodes` and `termnames` must be same length.

Examples

Create a Circuit and Define Its Nodes as Terminals

Create a circuit names **new_circuit1**.

```
hckt1 = circuit('new_circuit1');
```

Add a resistor and capacitor to the circuit.

```
add(hckt1,[1 2],resistor(50));
add(hckt1,[2 3],capacitor(1e-9));
```

Set the terminals of the circuit.

```
setterminals(hckt1,[1 3])
disp(hckt1)
```

```
  circuit: Circuit element

    ElementNames: {'R'  'C'}
           Nodes: [1 2 3]
            Name: 'new_circuit1'
       Terminals: {'t1'  't2'}
```

Create a Circuit and Define Its Nodes as Terminals Using Names

Create a circuit and add three resistors to it.

```
hckt2 = circuit('example_circuit2');
add(hckt2,[1 2],resistor(50));
add(hckt2,[1 3],resistor(50));
add(hckt2,[1 4],resistor(50));
```

Set terminals of the circuit by using (a, b, c) as **termnames**.

```
setterminals(hckt2,[2 3 4],{'a' 'b' 'c'})
disp(hckt2)
```

```
  circuit: Circuit element

    ElementNames: {'R'  'R_1'  'R_2'}
           Nodes: [1 2 3 4]
            Name: 'example_circuit2'
       Terminals: {'a'  'b'  'c'}
```

Add Two Circuits Together

Create circuit 1 and set the terminals using the **setterminals** functions.

```
hckt1 = circuit('circuit_new1');
add(hckt1,[1 2], resistor(100));
setterminals(hckt1, [1 2]);
disp(hckt1);
```

```
  circuit: Circuit element

    ElementNames: {'R'}
```

```
           Nodes: [1 2]
            Name: 'circuit_new1'
       Terminals: {'t1'  't2'}
```

Create circuit 2 and set the terminals.

```
hckt2 = circuit('circuit_new2');
add(hckt2, [3 4], capacitor(1.5e-9));
setterminals(hckt2, [3 4]);
disp(hckt2);
```

```
  circuit: Circuit element

    ElementNames: {'C'}
           Nodes: [3 4]
            Name: 'circuit_new2'
       Terminals: {'t1'  't2'}
```

Add the two circuits.

```
add(hckt1, [2 4], hckt2);
disp(hckt2)
```

```
  circuit: Circuit element

    ElementNames: {'C'}
           Nodes: [3 4]
            Name: 'circuit_new2'
       Terminals: {'t1'  't2'}
     ParentNodes: [2 4]
      ParentPath: 'circuit_new1'
```

```
disp(hckt1)
```

```
  circuit: Circuit element

    ElementNames: {'R'  'circuit_new2'}
           Nodes: [1 2 4]
            Name: 'circuit_new1'
       Terminals: {'t1'  't2'}
```

Input Arguments

`cktobj` — Circuit object
scalar handle object

Circuit object for which the terminals are defined, specified as a scalar handle object.

`cktnodes` — Circuit nodes
vector of integers

Circuit nodes, used by the function to define the terminals of the circuit, specified as a vector of integers.

`termnames` — Names
character vector

Names, used to identify the terminals defined for the circuit object, specified as a character vector.

See Also

`add` | `clone` | `setports` | `sparameters`

Introduced in R2013b

clone

Create copy of existing circuit element or circuit object

Syntax

```
outelem = clone(inelem)
outckt = clone(inckt)
```

Description

`outelem = clone(inelem)` creates a circuit element, `outelem`, with identical properties as `inelem`. The clone does not copy information about the parent circuit such as `ParentNodes` and `ParentPath`.

`outckt = clone(inckt)` creates a circuit object, `outckt`, identical to `inckt`. Circuit elements in the `inckt` are cloned recursively and added to the same nodes in the `outckt`. The ports or terminals in the `outckt` are defined same as `inckt`.

Examples

Create an Element and Clone It

Create a resistor element.

```
hR1 = resistor(50);
disp (hR1)
```

```
  resistor: Resistor element

    Resistance: 50
          Name: 'R'
     Terminals: {'p'  'n'}
```

Clone resistor **hR1**.

```
hR2 = clone(hR1);
disp (hR2)
```

```
  resistor: Resistor element

    Resistance: 50
          Name: 'R'
     Terminals: {'p'  'n'}
```

Create an Circuit and Clone it

Create a circuit object. Add a resistor and capacitor to it.

```
hckt1 = circuit('circuit1');
hC1= add(hckt1,[1 2],capacitor(3e-9));
hR1 = add(hckt1,[2 3],resistor(100));
disp(hckt1)
```

```
  circuit: Circuit element

    ElementNames: {'C'  'R'}
           Nodes: [1 2 3]
            Name: 'circuit1'
```

Clone the circuit object.

```
hckt2 = clone(hckt1);
disp (hckt2)
```

```
  circuit: Circuit element

    ElementNames: {'C'  'R'}
           Nodes: [1 2 3]
            Name: 'circuit1'
```

Input Arguments

`inelem` — Circuit element
scalar handle object

Circuit element to be cloned, specified as scalar handle object. The circuit element can be a resistor, capacitor, or inductor.

`inckt` — Circuit object
scalar handle object

Circuit object to be cloned, specified as scalar handle object.

Output Arguments

`outelem` — Circuit element
scalar handle object

Cloned circuit element, returned as scalar handle object. The circuit element can be a resistor, capacitor, or inductor.

`outckt` — Circuit object
scalar handle object

Cloned circuit object, returned as scalar handle object.

See Also

`add` | `setports` | `setterminals` | `sparameters`

Introduced in R2013b

rfwrite

Write RF network data to Touchstone file

Syntax

```
rfwrite(data,freq,filename)
rfwrite(netobj,filename)
rfwrite(_____,Name,Value)
```

Description

`rfwrite(data,freq,filename)` creates a Touchstone data file, `filename`. `rfwrite` touchstone files output 16 digits.

`rfwrite(netobj,filename)` creates a Touchstone file from a network parameter object, `netobj`.

`rfwrite(_____,Name,Value)` creates a Touchstone file using the options in the name-value pair arguments following the `filename`.

Examples

Write a Touchstone File Using Data and Frequency Values

Write a new Touchstone file from file `default.s2p` using *data* as `S50.Parameters` and *freq* as `S50.Frequencies`. The output is stored in `defaultnew.s2p`.

```
S50 = sparameters('default.s2p');
data = S50.Parameters;
freq = S50.Frequencies;
rfwrite(data, freq, 'defaultnew.s2p')
```

Write a Touchstone File Using Network Object Parameters

Convert an existing Touchstone file `passive.s2p` to `S-parameters` with a new resistance value. Write a Touchstone file `passive100.s2p` using the new S-parameters.

```
S50 = sparameters('passive.s2p');
S100 = newref(S50,100);
rfwrite(S100, 'passive100.s2p');
```

Write a Touchstone File Using Name-Value Pair Arguments

Convert an existing Touchstone file `passive.s2p` to `S-parameters` with a new resistance value. Write a Touchstone file `passive150.s2p` in `MHz` using the new S-parameters.

```
S50 = sparameters('passive.s2p');
S150 = newref(S50,150);
rfwrite(S150, 'passive150.s2p','FrequencyUnit', 'MHz');
```

Write a Touchstone File Using Y-Parameters

Convert an existing Touchstone file `passive.s2p` to `Y-parameters`. Write a Touchstone file `passive.y2p` in `MHz` using the new Y-parameters.

```
Y50 = yparameters('passive.s2p');
rfwrite(Y50, 'passive.y2p','FrequencyUnit', 'MHz');
```

Input Arguments

data — Number of ports and frequencies
matrix

Number of ports and frequencies, specified as an `N-by-N-by-K` matrix, to create Touchstone file. `N` is the number of ports of data to be written. `K` is the number of frequencies.

Example: 2x2x20 complex double

Data Types: `double`

`freq` — Value of frequencies
numeric vector

Value of frequencies, specified as a numeric vector of length `K`, represents the value of frequencies in Hz.

Example: 202 x 1 double

Data Types: `double`

`filename` — Name of Touchstone file
character vector

Name of a Touchstone file, specified as a character vector.

Example: default.s2p

Data Types: `char`

`netobj` — Network parameter object
scalar

Network parameter object, specified as a scalar, to create Touchstone file. The `netobj` can be any one of the following types `s-parameters`, `y-parameters`, `z-parameters`, `h-parameters`, `g-parameters`, or `abcd-parameters`.

Example: 1x1 S-parameters

Data Types: `double`

Name-Value Pair Arguments

Optional comma-separated pairs of `Name,Value` arguments, where `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside single quotes (`''`).

Example: rfwrite(S150, 'passive150.s2p','FrequencyUnit', 'MHz')

`FrequencyUnit` — Scaling unit for frequency values
ghz (default) | mhz | khz | hz

Scaling unit for frequency value, specified as a comma-separated pair consisting of `'Frequency Unit'` and any one of the values shown in value summary.

Example: 3.150746640000000e-04

Data Types: `double`

`Parameter` — Network parameter type
S (default) | Y | Z | h | g

Network parameter type, specified as a comma-separated pair consisting of `'Parameter'` and any one of the values shown in value summary. This pair determines the parameter type the data has to be converted into in the Touchstone file.

Example: 0.0018 + 0.0122i 0.9981 - 0.0127i 0.9984 - 0.0131i 0.0017 + 0.0123i

Data Types: `double`

`Format` — File storage format
MA (Magnitude Angle) (default) | DB (Decibel) | RI (Real Imaginary)

File storage format, specified as a comma-separated pair consisting of `'Format'` any one of the values shown in value summary. This pair determines the format to store the Touchstone file.

Example: MA

`ReferenceResistance` — Resistance
50 (default) | positive scalar (Ohm)

Reference resistance, specified as a comma-separated pair consisting of `'ReferenceResistance'` and a positive scalar.

Example: 100

Data Types: `double`

See Also
`report` | `show` | `sparameters` | `write`

Topics
"Writing A Touchstone® File"

Introduced in R2014a

addstage

Add stage to RF chain object

Syntax

`addstage(obj,g,nf,oip3val,'Name',nm)`

`addstage(obj,g,nf,'IIP3',ip3val,'Name',nm)`

`addstage(___,Name,Value)`

Description

`addstage(obj,g,nf,oip3val,'Name',nm)` adds a stage to the RF chain object `obj`. This syntax also specifies the gain, noise figure, output-referred third-order intercept and name of the RF chain object `obj`. You must specify the stage name using name-value pair arguments.

`addstage(obj,g,nf,'IIP3',ip3val,'Name',nm)` adds a stage having input-referred third-order intercept of value `i3` to the RF chain object `obj`. You must specify the IIP3 value and stage name using name-value pair arguments.

`addstage(___,Name,Value)` adds a new stage having properties specified by one or more name-value pair arguments. Properties not specified are given their default values.

Input Arguments

`obj` — RF chain object
scalar handle

RF chain object, specified as a scalar handle.

`g` — Gain
0 (default) | scalar | vectors of same length

Gain of a stage, specified as a scalar or vectors of same length.

Example: 11

Data Types: `double`

`nf` — Noise figure
0 (default) | scalar | vectors of same length

Noise figure of a stage, specified as a scalar or vectors of same length.

Example: 25

Data Types: `double`

`oip3val` — Output-referred third-order intercept
inf (default) | scalar | vectors of same length

Output-referred third-order intercept of a stage, specified as a scalar or vectors of same length.

Example: 30

Data Types: `double`

Name-Value Pair Arguments

Optional comma-separated pairs of `Name,Value` pair arguments, where `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside single quotes (`''`).

Example: addstage(ch,2,'NoiseFigure',20, 'Name', 'lna1')

`Gain` — Gain
0 (default) | scalar | vectors of same length

Gain of a stage, specified as a comma-separated pair consisting of `'Gain'` and a scalar or vectors of same length.

Example: 10

Data Types: `double`

`NoiseFigure` — Noise figure
0 (default) | non-negative scalar | vectors of same length

Noise figure of a stage, specified as a comma-separated pair consisting of `'NoiseFigure'` and a non-negative scalar or vectors of same length.

Example: 30

Data Types: `double`

`OIP3` — Output-referred third-order intercept
inf (default) | scalar | vectors of same length

Output-referred third-order intercept of a stage, specified as a comma-separated pair consisting of `'OIP3'` and a scalar or vectors of same length.

Example: 30

Data Types: `double`

`IIP3` — Input-referred third-order intercept
inf (default) | scalar | vectors of same length

Input-referred third-order intercept of a stage, specified as a comma-separated pair consisting of `'Name'` and a scalar or vectors of same length.

Example: 30

Data Types: `double`

`Name` — Name of stage
character vector

Name of a character vector, specified as a comma-separated pair consisting of `'Name'` and a character vector.

Example: amp1

Data Types: `char`

Examples

Add Stages to RF Chain

Create an RF chain object and view it.

```
rfch = rfchain
```

```
rfch =
  rfchain with properties:

           Gain: []
    NoiseFigure: []
           OIP3: []
           IIP3: []
           Name: {}
      NumStages: 0

  Use worksheet or plot for cascade results
```

Add stage 1 with default name and IIP3.

```
addstage(rfch,11,25);
```

Add stage 2 with default noise figure.

```
addstage(rfch,-3,'IIP3', 10, 'Name','filt1');
```

View results on a worksheet.

```
worksheet(rfch)
```

	stage1	filt1
Stage Gain	11	-3
Stage Noise Figure	25	0
Stage OIP3	Inf	7
Stage IIP3	Inf	10
Cascaded Gain	11	8
Cascaded Noise Figure	25	25
Cascaded OIP3	Inf	7.0000
Cascaded IIP3	Inf	-1.0000

Add Stages to RF Chain Using Name-Value Pairs

Create an RF chain object and view it.

```
rfch = rfchain
```

```
rfch = 
  rfchain with properties:

             Gain: []
      NoiseFigure: []
             OIP3: []
             IIP3: []
             Name: {}
        NumStages: 0

  Use worksheet or plot for cascade results
```

Add stage 1 with OIP3.

```
addstage(rfch,'Gain', 10,'NoiseFigure', 20,'OIP3', 30, 'Name','amp1');
```

Add stage 2 with IIP3.

```
addstage(rfch,'Gain', 8,'NoiseFigure', 22, 'IIP3', 20,'Name','amp2');
```

View results on a worksheet.

```
worksheet(rfch)
```

	amp1	amp2
Stage Gain	10	8
Stage Noise Figure	20	22
Stage OIP3	30	28
Stage IIP3	20	20
Cascaded Gain	10	18
Cascaded Noise Figure	20	20.6352
Cascaded OIP3	30	27.5861
Cascaded IIP3	20	9.5861

See Also

`cumgain` | `cumiip3` | `cumnoisefig` | `cumoip3` | `plot` | `setstage` | `worksheet`

setstage

Update RF chain stage

Syntax

`setstage(obj,idx,g,nf,oip3val,'Name',nm)`

`setstage(obj,g,nf,'IIP3',ip3val,'Name',nm)`

`setstage(___,Name,Value)`

Description

`setstage(obj,idx,g,nf,oip3val,'Name',nm)` updates gain, noise figure, output-referred third-order intercept values of a stage. Use the index, `idx` of the RF chain object to specify the stage you want to update. At a time, you can change the name of only one stage.

`setstage(obj,g,nf,'IIP3',ip3val,'Name',nm)` updates the input-referred third-order intercept value of a stage.

`setstage(___,Name,Value)` updates the values of a stage using the name-value pair arguments.

Input Arguments

`obj` — RF chain object
scalar handle

RF chain object, specified as a scalar handle.

`idx` — Number of a stage
integer | vector of integers

Number of a stage, specified as an integer or vector of integers.

Example: 2

Data Types: `double`

g — Gain
0 (default) | scalar | vectors of same length

Gain of a stage, specified as a scalar or vectors of same length.

Example: -3

Data Types: `double`

`nf` — Noise figure
0 (default) | scalar | vectors of same length

Noise figure of a stage, specified as a scalar or vectors of same length.

Example: 20

Data Types: `double`

`oip3val` — Output-referred third-order intercept
inf (default) | scalar | vectors of same length

Output-referred third-order intercept of a stage, specified as a scalar or vectors of same length.

Example: 30

Data Types: `double`

Name-Value Pair Arguments

Optional comma-separated pairs of `Name,Value` pair arguments, where `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside single quotes (`''`).

Example: setstage(ch,2,'NoiseFigure',20)

`Gain` — Gain
0 (default) | scalar | vectors of same length

Gain of a stage, specified as a comma-separated pair consisting of `'Gain'` and a scalar or vectors of same length. This pair updates the gain of a stage specified by `idx`.

Example: 10

Data Types: `double`

`NoiseFigure` — Noise figure
0 (default) | scalar | vectors of same length

Noise figure of a stage, specified as a comma-separated pair consisting of `'NoiseFigure'` and a scalar or vectors of same length. This pair updates the noise figure of a stage specified by `idx`.

Example: 30

Data Types: `double`

`OIP3` — Output-referred third-order intercept
inf (default) | scalar | vectors of same length

Output-referred third-order intercept of a stage, specified as a comma-separated pair consisting of `'OIP3'` and a scalar or vectors of same length. This pair updates the output-referred third-order intercept of a stage specified by `idx`.

Example: 30

Data Types: `double`

`IIP3` — Input-referred third-order intercept
inf (default) | scalar | vectors of same length

Input-referred third-order intercept of a stage, specified as a comma-separated pair consisting of `'IIP3'` and a scalar or vectors of same length. This pair updates the input-referred third-order intercept of a stage specified by `idx`.

Example: 30

Data Types: `double`

`Name` — Name of stage
character vector

Name of a stage, specified as a comma-separated pair consisting of `'Name'` and a character vector. This pair updates the name of the stage specified by `idx`.

Example: amp1

Examples

Change Noise Figure Of RF Chain Stage

Create an RF chain object.

```
g = [11 -3];
nf = [25 3];
o3 = [30 Inf];
nm = {'amp1','filt1'};
rfch = rfchain(g,nf,o3,'Name',nm);
```

Change the noise figure of **filt1** to 20 dB.

```
setstage(rfch,2,'NoiseFigure',20)
```

View results on a worksheet.

```
worksheet(rfch)
```

	amp1	filt1
Stage Gain	11	-3
Stage Noise Figure	25	20
Stage OIP3	30	Inf
Stage IIP3	19	Inf
Cascaded Gain	11	8
Cascaded Noise Figure	25	25.1067
Cascaded OIP3	30	27
Cascaded IIP3	19	19

See Also

addstage | cumgain | cumiip3 | cumnoisefig | cumoip3 | plot | worksheet

cumgain

Cascaded gain of the RF chain object

Syntax

`g = cumgain(obj)`

Description

`g = cumgain(obj)` returns the cascaded gain for each stage of the RF chain object `obj`.

Input Arguments

`obj` — RF chain object
scalar handle

RF chain object, specified as a scalar handle.

Output Arguments

g — Cascaded gain
vectors

Cascaded gain of the RF chain object, returned as vectors. The vector length is equal to the number of stages in the RF chain object.

Examples

Calculate Cascaded Gain

Assign stage-by-stage values of gain, noise figure, OIP3 and stage names.

```
g = [11 -3 7];
nf = [25 3 5];
o3 = [30 Inf 10];
nm = {'amp1','filt1','lna1'};
```

Create an RF chain object.

```
rfch = rfchain(g,nf,o3,'Name',nm);

%Calculate cascaded gain.
gain = cumgain(rfch)
```

```
gain = 1×3

    11     8    15
```

See Also

addstage | cumiip3 | cumnoisefig | cumoip3 | plot | setstage | worksheet

cumnoisefig

Cascaded noise figure of the RF chain object

Syntax

```
nf = cumnoisefig(obj)
```

Description

`nf = cumnoisefig(obj)` returns the cascaded noise figure for each stage for RF chain object `obj`. The syntax first calculates the noise factor and then the noise figure. The formulae used are:

$$noisefactor(total) = noisefactor(1) + (noisefactor(2) - 1) / g1 + (noisefactor(3) - 1) / g1 + g2 + ...$$

$$noisefigure = 10 * \log 10(noisefactor)$$

Input Arguments

`obj` — RF chain object
scalar handle

RF chain object, specified as a scalar handle.

Output Arguments

`nf` — Cascaded noise figure
vectors

Cascaded noise figure for RF chain object, returned as vectors. The vector length is equal to the number of stages in the RF chain object.

Examples

Calculate Cascaded Noise Figure

Assign stage-by-stage values of gain, noise figure, OIP3 and stage names.

```
g = [11 -3 7];
nf = [25 3 5];
o3 = [30 Inf 10];
nm = {'amp1','filt1','lna1'};
```

Create an RF chain object.

```
rfch = rfchain(g,nf,o3,'Name',nm);
```

Calculate cascaded noise figure.

```
noisefig = cumnoisefig(rfch)
```

```
noisefig = 1×3

   25.0000   25.0011   25.0058
```

See Also

addstage | cumgain | cumiip3 | cumoip3 | plot | setstage | worksheet

cumoip3

Cascaded output-referred third-order intercept of the RF chain object

Syntax

```
oip3val = oip3(obj)
```

Description

`oip3val = oip3(obj)`returns the cascaded output-referred third-order intercept for each stage of the RF chain object `obj`. The oip3 is calculated using the formula:

$$oip3lin = iip3lin * gainlin$$

where all values are linear

Input Arguments

`obj` — RF chain object
scalar handle

RF chain object, specified as a scalar handle.

Output Arguments

`oip3val` — Cascaded output-referred third-order intercept
vectors

Cascaded output-referred third-order intercept for RF chain object, returned as vectors. The vector length is equal to the number of stages in the RF chain object.

Examples

Calculate Cascaded OIP3

Assign stage-by-stage values of gain, noise figure, OIP3 and stage names.

```
g = [11 -3 7];
nf = [25 3 5];
o3 = [30 Inf 10];
nm = {'amp1','filt1','lna1'};
```

Create an RF chain object.

```
rfch = rfchain(g,nf,o3,'Name',nm);
```

Calculate cascaded oip3 value.

```
oip3val = cumoip3(rfch)
```

```
oip3val = 1×3

   30.0000   27.0000    9.9827
```

See Also

addstage | cumgain | cumiip3 | cumnoisefig | plot | setstage | worksheet

cumiip3

Cascaded input-referred third-order intercept of the RF chain object

Syntax

```
ip3val = iip3(obj)
```

Description

`ip3val = iip3(obj)` returns the cascaded input-referred third-order intercept for each stage of the RF chain object `obj`. The input-referred third-order intercept is calculated using the formula:

$$1/iip3lin(total) = 1/iip3lin(1) + g1/iip3lin(2) + (g1*g2)/iiplin(3) + \ldots.$$
$$\text{where, } iip3lin = \text{iip3 (linear values)}$$

Input Arguments

`obj` — RF chain object
scalar handle

RF chain object, specified as a scalar handle.

Output Arguments

`ip3val` — Cascaded input-referred third-order intercept
vectors

Cascaded input-referred third-order intercept for RF chain object, returned as vectors. The vector length is equal to the number of stages in the RF chain object.

Examples

Calculate Cascaded IIP3

Assign stage-by-stage values of gain, noise figure, IIP3 and stage names.

```
g = [11 -3 7];
nf = [25 3 5];
i3 = [19 Inf 3];
nm = {'amp1','filt1','lna1'};
```

Create an RF chain object.

```
rfch = rfchain(g,nf,'IIP3',i3,'Name',nm);
```

Calculate cascaded iip3 value.

```
iip3val = cumiip3(rfch)
```

```
iip3val = 1×3

   19.0000   19.0000   -5.0173
```

See Also

addstage | cumgain | cumnoisefig | cumoip3 | plot | setstage | worksheet

plot

Plot RF chain cascaded analysis results.

Syntax

`plot(obj)`

`h = plot(obj)`

Description

`plot(obj)` displays a plot of the cascaded gain, noise figure, OIP3 and IIP3 values of the RF chain object `obj`.

`h = plot(obj)` returns a column vector of line series handles, where h contains one handle per plotted line.

Input Arguments

`obj` — RF chain object
scalar handle

RF chain object, specified as a scalar handle.

Output Arguments

h — line series handle
column vector

Line series handle, returned as a column vector, that contains one handle per plotted line.

Examples

Plot Results of RF Chain Object

Assign stage-by-stage values of gain, noise figure, OIP3 and stage names.

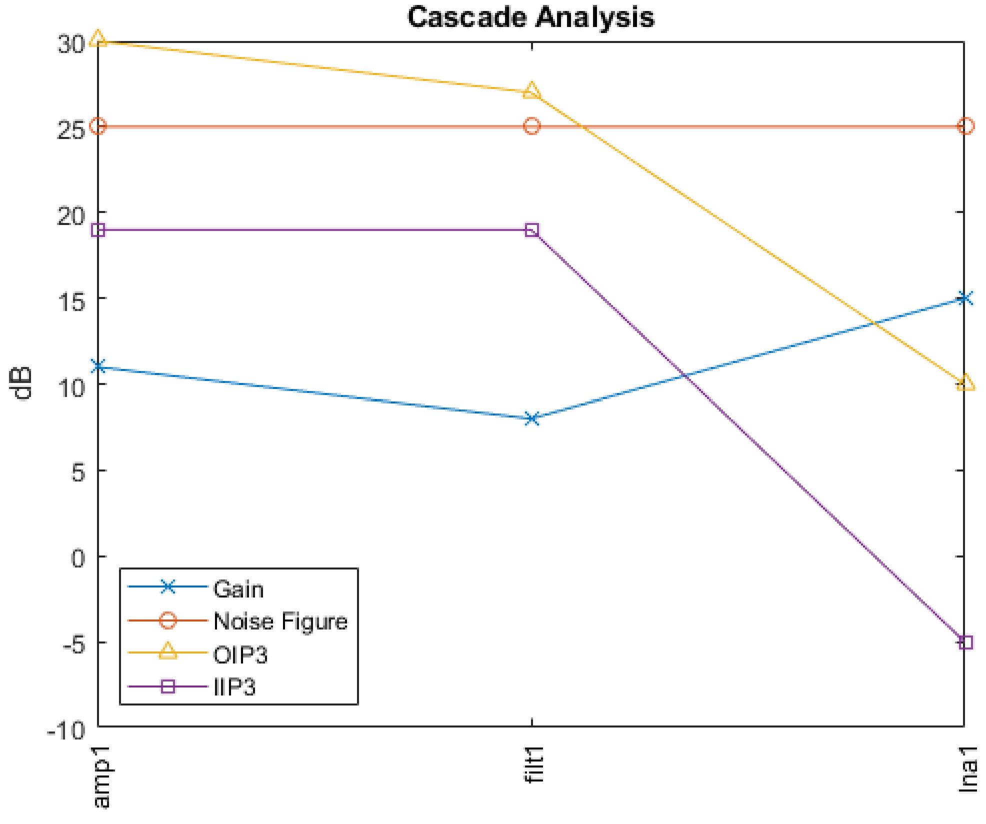
```
g = [11 -3 7];
nf = [25 3 5];
oip3 = [30 Inf 10];
nm = {'amp1','filt1','lna1'};
```

Create an RF chain object.

```
rfch = rfchain(g,nf,oip3,'Name',nm);
```

Plot the results.

```
plot(rfch)
```



See Also

`addstage` | `cumgain` | `cumiip3` | `cumnoisefig` | `cumoip3` | `setstage` | `worksheet`

worksheet

RF chain cascaded analysis table

Syntax

```
worksheet(obj)
```

```
fig = worksheet(obj)
```

Description

`worksheet(obj)` displays a table of values for the gain, noise figure, OIP3, and IIP3 of the RF chain object `obj`. The table contains both the original input values and the calculated cascade values.

`fig = worksheet(obj)` returns a figure handle of the table.

Input Arguments

`obj` — RF chain object
scalar handle

RF chain object, specified as a scalar handle.

Output Arguments

`fig` — figure handle
scalar handle object

Figure handle of the table, returned as a scalar handle object, that contains the properties of the RF chain object.

Examples

Create RF Chain Adding Stage-By-Stage Values

Assign three stage-by-stage values of gain, noise figure, OIP3 and stage names.

```
g = [11 -3 7];
nf = [25 3 5];
o3 = [30 Inf 10];
nm = {'amp1','filt1','lna1'};
```

Create an RF chain object.

```
rfch = rfchain(g,nf,o3,'Name',nm);
```

View results in a worksheet.

```
worksheet(rfch)
```

	amp1	filt1	lna1
Stage Gain	11	-3	7
Stage Noise Figure	25	3	5
Stage OIP3	30	Inf	10
Stage IIP3	19	Inf	3
Cascaded Gain	11	8	15
Cascaded Noise Figure	25	25.0011	25.0058
Cascaded OIP3	30	27	9.9827
Cascaded IIP3	19	19	-5.0173

See Also

addstage | cumgain | cumiip3 | cumnoisefig | cumoip3 | plot | setstage

groupdelay

Group delay of s-parameter object or RF Toolbox network object

Syntax

`gd = groupdelay(sparamobj)`

`gd = groupdelay(rfobj,freq)`

`gd = groupdelay(__,i,j)`

`gd = groupdelay(__,Name,Value)`

Description

`gd = groupdelay(sparamobj)` calculates the group delay of an S-parameter object at the frequencies specified in the S-parameter object file. `sparamobj` can be an s-parameters object or a nport object.

`gd = groupdelay(rfobj,freq)` calculates the group delay of an RF Toolbox network object, `rfobj`, at specified frequencies.

`gd = groupdelay(__,i,j)` calculates the group delay of a specific S_{ij}. If i, j are not specified, the group delay is calculated for S_{21} fro two-port objects and S_{11} for non-two-port objects.

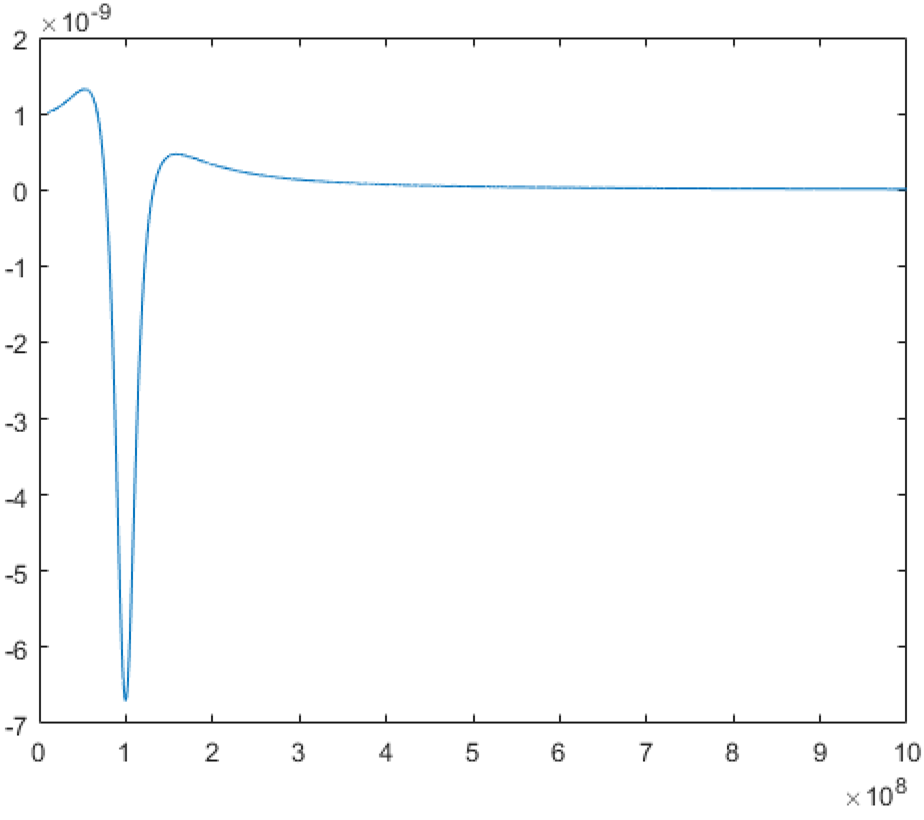
`gd = groupdelay(__,Name,Value)` calculates the group delay using additional options specified by one or more `Name,Value` pair arguments. You can use any of the arguments from previous syntaxes.

Examples

Group Delay of RLC Notch Filter

Calculate and plot the group delay of an RLC notch filter at a frequency range from 10 GHz through 1000 GHz frequency.

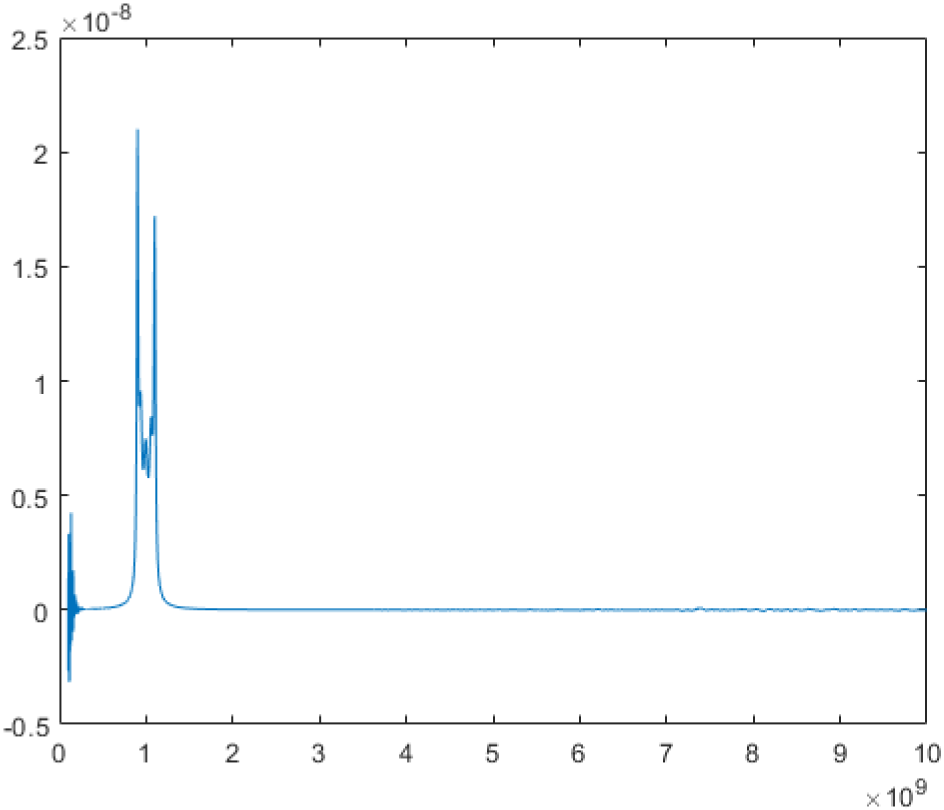
```
filt = circuit('notch');
add(filt,[1 2],resistor(200))
add(filt,[1 2],inductor(100e-9))
add(filt,[1 2],capacitor(25e-12))
setports(filt,[1 0],[2 0])
freq = 10e6:10e4:1000e6;
gd1 = groupdelay(filt,freq);
figure
plot(freq,gd1)
```



Group Delay of S-Parameter Data File.

Find and plot the group delay of the file 'defaultbandpass.s2p'.

```
S = sparameters('defaultbandpass.s2p');
freq = S.Frequencies;
gd2 = groupdelay(S,freq);
figure
plot(freq,gd2)
```



Input Arguments

`sparamobj` — S-parameter Touchstone data file
object handle

S-parameter Touchstone data file, specified as object handle. The function uses the data in this file to calculate the group delay.

Example: `'defaultbandpass.s2p'`

`rfobj` — RF network object
object

RF network object, specified as object, of the following types: s-parameters, nport, circuit, and lcladder.

Example: `lcladder`

`freq` — Frequencies
vector of positive real numbers for rf objects

Frequencies, specified as a vector of positive real numbers.

`i,j` — Port numbers of s-parameter object or rf object
scalar integers

Port numbers of s-parameter object or rf object, specified as a scalar integer.

Example: `S12`

Name-Value Pair Arguments

Example: `gd = groupdelay (filter, frequency, 'Aperture', 50)`

Optional comma-separated pairs of `Name,Value` arguments, where `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside single quotes (`''`).

`Aperture` — Width of two frequency points
freq*sqrt(eps) (default) | real, positive, numeric scalar or vector

Width of two frequency points, specified as the comma-separated pair consisting of `'Aperture'` and a real, positive, numerics scalar or vector.

Example: `'Aperture',50`

Data Types: `double`

`Impedance` — Impedance of S-parameters
real, positive, scalar

Impedance of S-parameters, specified as the comma-separated pair consisting of `'Impedance'` and a real positive numeric scalar. The default impedance values for different objects are:

- `50` — LC ladder and circuit objects
- `obj.impedance` — S-parameter objects

- `obj.networkdata.impedance` — N-port objects

Example: `50`

Data Types: `double`

Output Arguments

gd — Group delay
numeric scalar in seconds

Group delay, returned as a numeric scalar in seconds.

See Also

`lcladder` | `nport` | `sparameters`

Introduced in R2015b

computeBudget

Compute results of rfbudget object

Syntax

```
computeBudget(rfobj)
```

Description

`computeBudget(rfobj)` computes the result of an RF budget object. You can use this method only when the `AutoUpdate` property of the RF budget object is set to `false`.

Input Arguments

`rfobj` — RF budget analysis object
object handle

RF budget analysis object, specified as a object handle.

See Also

`exportRFBlockset` | `exportScript` | `exportTestbench` | `rfbudget` | `show`

Introduced in R2017a

exportScript

Export MATLAB code that generates RF budget object

Syntax

```
exportScript(rfobj)
```

Description

`exportScript(rfobj)` exports the MATLAB command-line code that generates an RF budget object. The script opens in an `Untitled*` window in the MATLAB editor.

Input Arguments

`rfobj` — RF budget analysis object
object handle

RF budget analysis object, specified as a object handle.

Examples

Export RF Budget Analysis to MATLAB Script

Create an RF budget object.

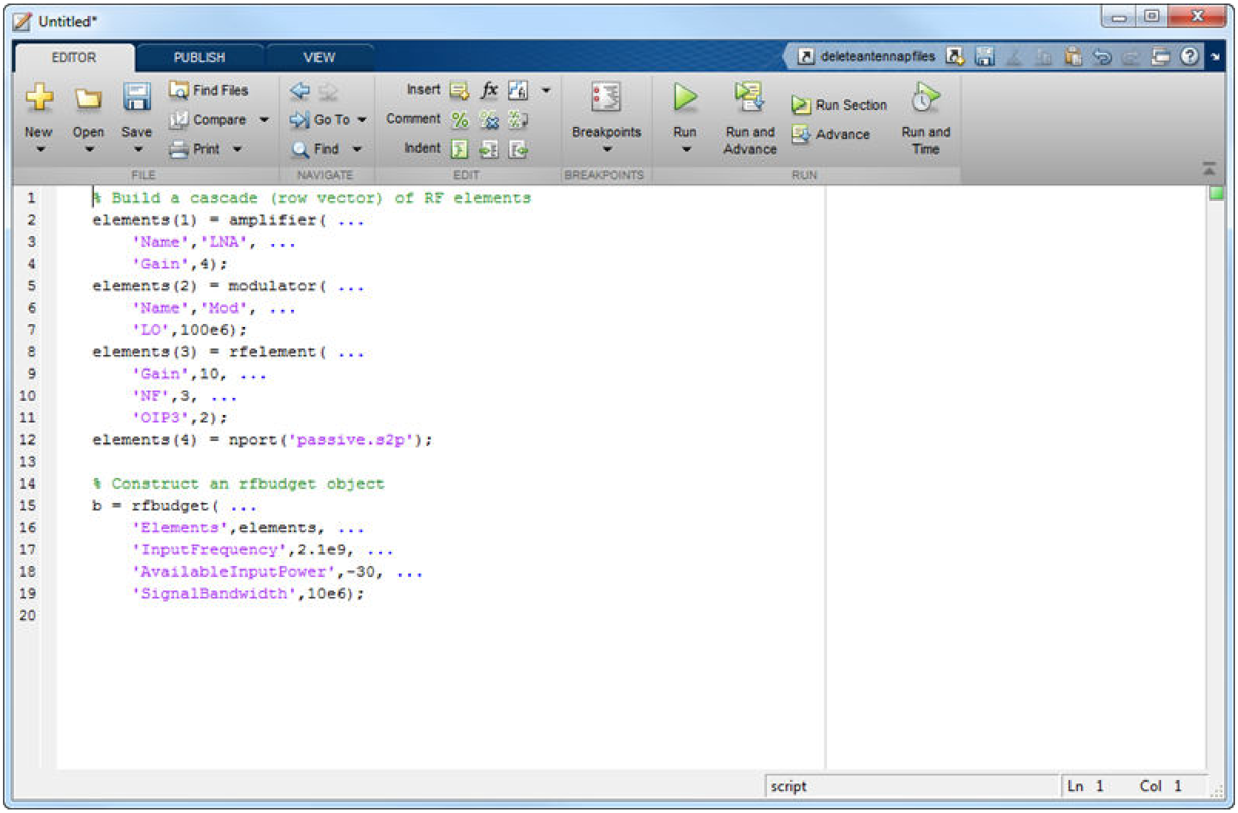
```
a = amplifier('Name','LNA','Gain',4);
m = modulator('ConverterType','Up','LO',100e6,'Name','Mod');
r = rfelement('Gain',10,'NF',3,'OIP3',2);
n = nport('passive.s2p');
```

Calculate the RF budget analysis.

```
b = rfbudget([a m r n],2.1e9,-30,10e6);
```

Export the analysis to a MATLAB script.

exportScript(b)



See Also

computeBudget | exportRFBlockset | exportTestbench | rfbudget | show

Introduced in R2017a

exportRFBlockset

Create RF Blockset model from RF budget object

Syntax

```
exportRFBlockset(rfobj)
sys = exportRFBlockset(rfobj)
```

Description

`exportRFBlockset(rfobj)` creates an RF Blockset model from the RF budget object, and opens the system.

`sys = exportRFBlockset(rfobj)` creates an RF Blockset model, and returns the system name.

Input Arguments

`rfobj` — RF budget analysis object
object handle

RF budget analysis object, specified as a object handle.

See Also

`computeBudget` | `exportScript` | `exportTestbench` | `rfbudget` | `show`

Introduced in R2017a

exportTestbench

Create measurement testbench from RF budget object

Syntax

```
exportTestbench(rfobj)
sys = exportTestbench(rfobj)
```

Description

`exportTestbench(rfobj)` creates an RF Blockset model from the RF budget object, and opens a measurement testbench system.

`sys = exportTestbench(rfobj)` creates an RF Blockset model, and returns the measurement testbench system.

Input Arguments

`rfobj` — RF budget analysis object
object handle

RF budget analysis object, specified as a object handle.

See Also

`computeBudget` | `exportRFBlockset` | `exportScript` | `rfbudget` | `show`

Introduced in R2017a

show

Display RF budget object in RF Budget Analyzer app

Syntax

```
show(rfobj)
```

Description

`show(rfobj)` opens an RF Budget Analyzer app to display a clone of the RF budget object.

Input Arguments

`obj` — RF budget analysis object
object handle

RF budget analysis object, specified as a object handle.

Examples

Display RF Budget Analysis in RF Budget Analyzer App

Create an RF budget object.

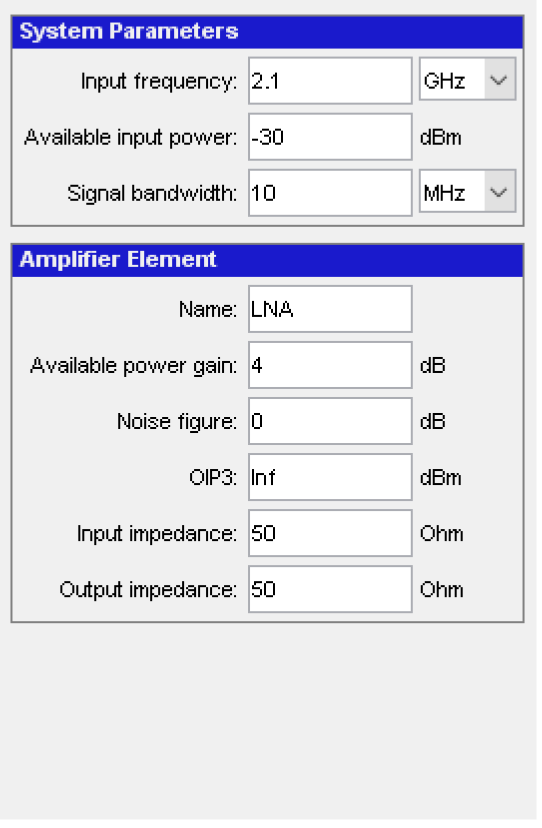
```
a = amplifier('Name','LNA','Gain',4);
m = modulator('ConverterType','Up','LO',100e6,'Name','Mod');
r = rfelement('Gain',10,'NF',3,'OIP3',2);
n = nport('passive.s2p');
```

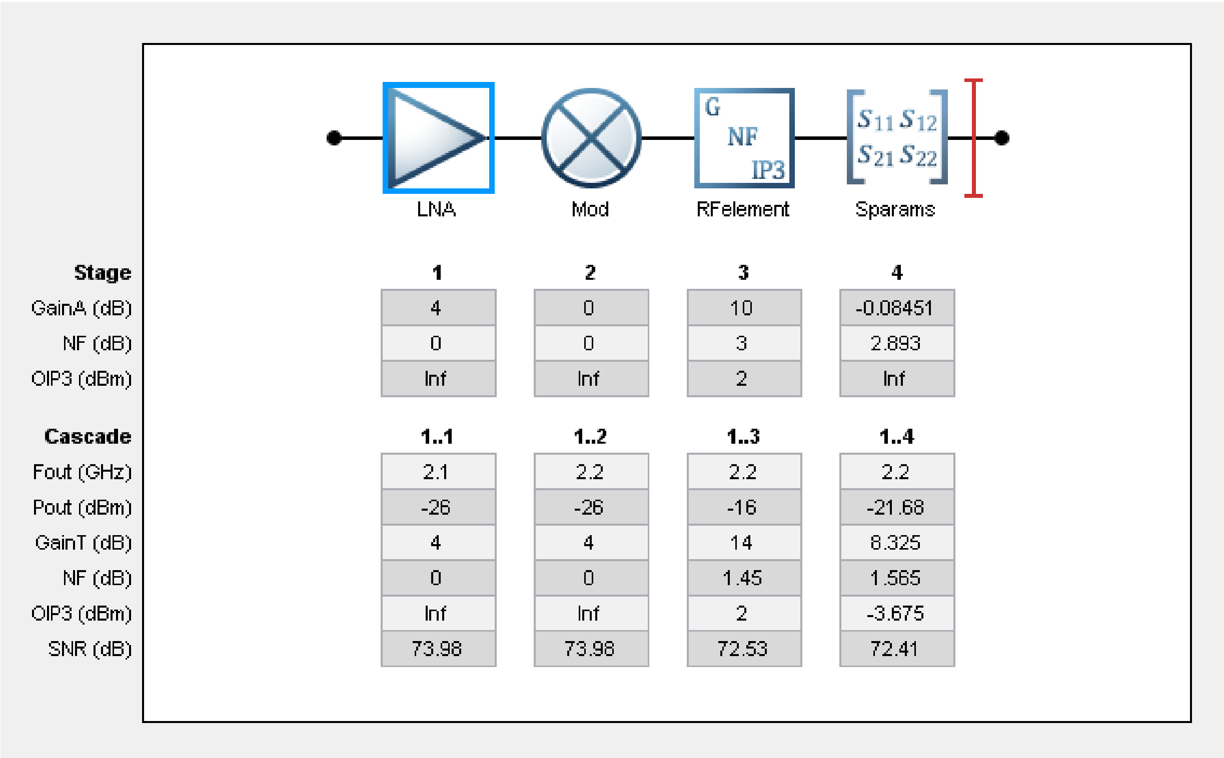
Calculate the RF budget analysis.

```
b = rfbudget([a m r n],2.1e9,-30,10e6);
```

Display the RF budget for exploration in the RF Budget Analyzer app.

```
show(b)
```





Stage	1	2	3	4
GainA (dB)	4	0	10	-0.08451
NF (dB)	0	0	3	2.893
OIP3 (dBm)	Inf	Inf	2	Inf
Cascade	**1..1**	**1..2**	**1..3**	**1..4**
Fout (GHz)	2.1	2.2	2.2	2.2
Pout (dBm)	-26	-26	-16	-21.68
GainT (dB)	4	4	14	8.325
NF (dB)	0	0	1.45	1.565
OIP3 (dBm)	Inf	Inf	2	-3.675
SNR (dB)	73.98	73.98	72.53	72.41

See Also

`computeBudget` | `exportRFBlockset` | `exportScript` | `exportTestbench` | `rfbudget`

Introduced in R2017a

rfplot

Plot cumulative RF budget result versus cascade input frequency

Syntax

```
rfplot(rfobj,str)
```

Description

`rfplot(rfobj,str)` plots the RF budget result specified by STR versus a range of input frequencies. The input frequencies are applied to the cascade of elements in the RF budget object, `rfobj`.

Cumulative (that is, terminated subcascade) results are automatically computed to show the variation of the RF budget result through the entire design.

Examples

Plot Cumulative Output Power and Gain of RF System

Create an RF system.

Create an RF bandpassfilter using the Touchstone file `RFBudget_RF`.

```
f1 = nport('RFBudget_RF.s2p','RFBandpassFilter');
```

Create an amplifier with a gain of 11.53 dB, a noise figure (NF)of 1.53 dB, and an output third-order intercept (OIP3) of 35 dBm.

```
a1 = amplifier('Name','RFAmplifier','Gain',11.53,'NF',1.53,'OIP3',35);
```

Create a demodulator with a gain of 6 dB, a NF of 4 dB, and an OIP3 of 50 dBm.

```
d = modulator('Name','Demodulator','Gain',-6,'NF',4,'OIP3',50, ...
        'LO',2.03e9,'ConverterType','Down');
```

Create an IF bandpassfilter using the Touchstone file RFBudget_IF.

```
f2 = nport('RFBudget_IF.s2p','IFBandpassFilter');
```

Create an amplifier with a gain of 30 dB, a NF of 8 dB, and an OIP3 of 37 dBm.

```
a2 = amplifier('Name','IFAmplifier','Gain',30,'NF',8,'OIP3',37);
```

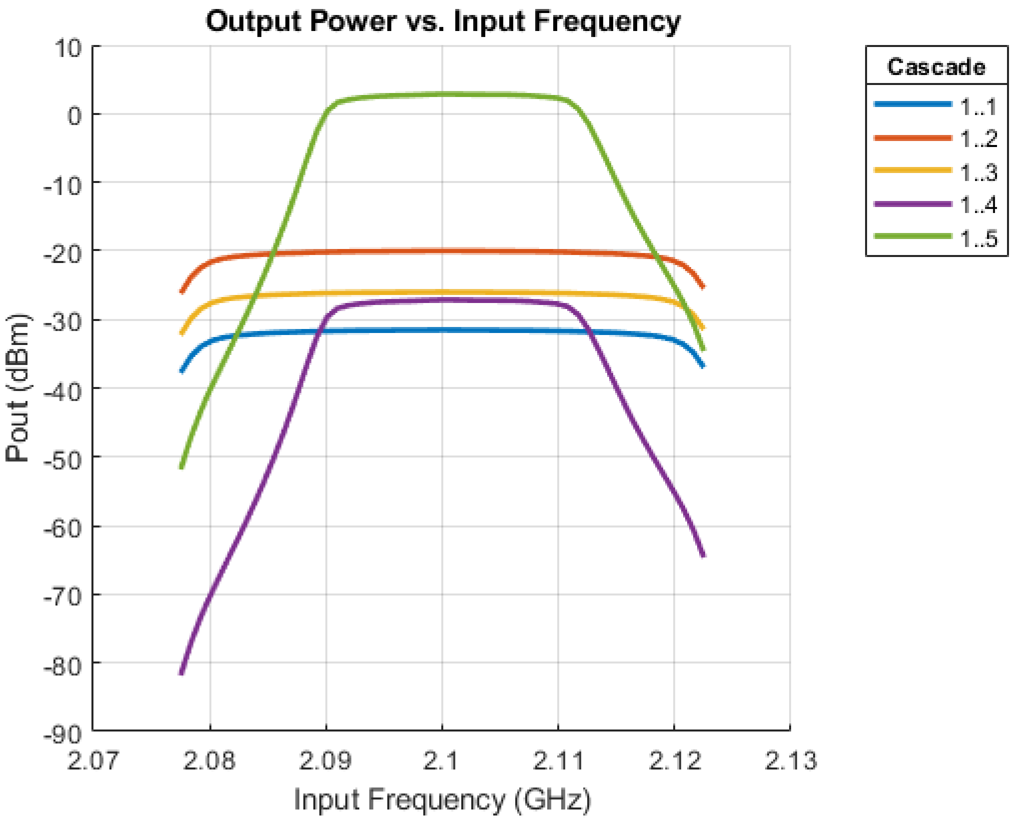
Calculate the RF budget of the system using an input frequency of 2.1 GHz, an input power of -30 dBm, and a bandwidth of 45 MHz.

```
b = rfbudget([f1 a1 d f2 a2],2.1e9,-30,45e6)
```

```
b = 
  rfbudget with properties:

               Elements: [1x5 rf.internal.rfbudget.Element]
         InputFrequency: 2.1 GHz
    AvailableInputPower: -30 dBm
        SignalBandwidth:  45 MHz
             AutoUpdate: true

   Analysis Results
        OutputFrequency: (GHz) [   2.1    2.1   0.07    0.07   0.07]
            OutputPower: (dBm) [-31.53    -20    -26  -27.15  2.847]
         TransducerGain: (dB)  [-1.534  9.996  3.996   2.847  32.85]
                     NF: (dB)  [ 1.533  3.064  3.377   3.611  7.036]
                   OIP3: (dBm) [   Inf     35  28.97   27.82  36.96]
                   IIP3: (dBm) [   Inf     25  24.97   24.97  4.116]
                    SNR: (dB)  [ 65.91  64.38  64.07   63.83  60.41]
```

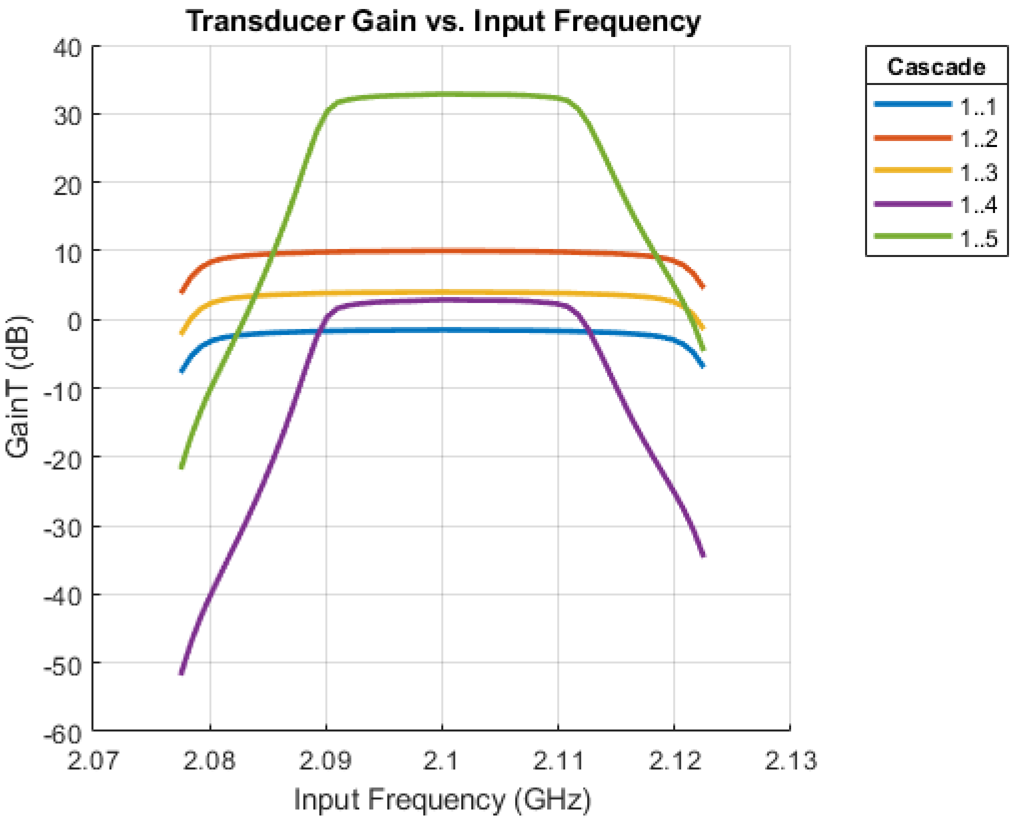
Plot the available output power.

```
rfplot(b,'Pout')
view(90,0)
```



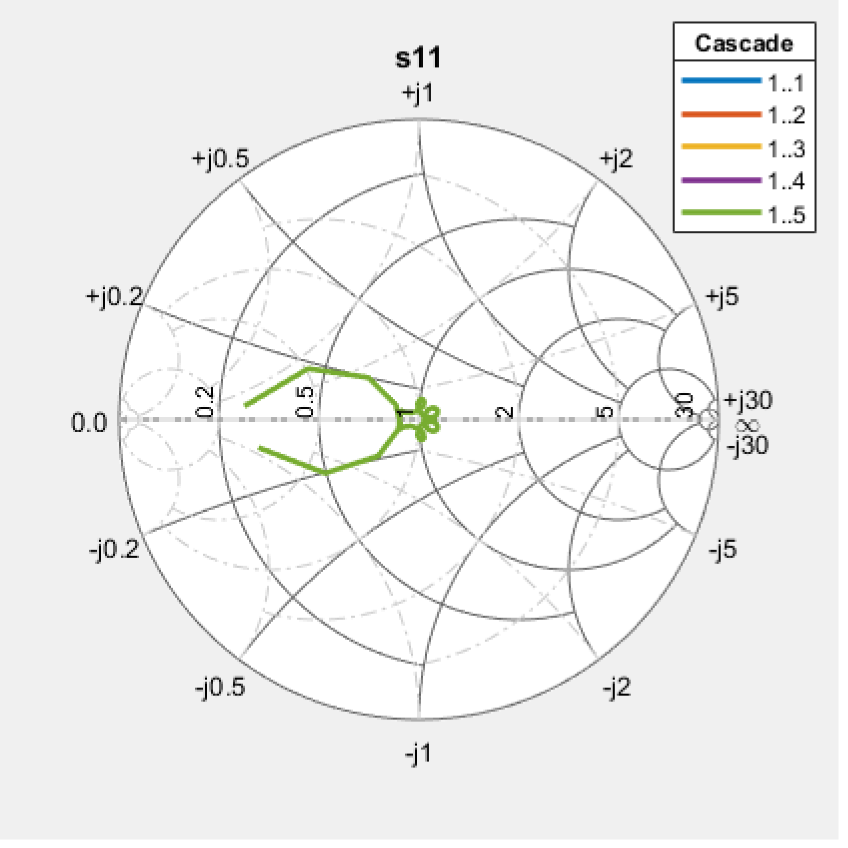
Plot the transducer gain.

```
rfplot(b,'GainT')
view(90,0)
```

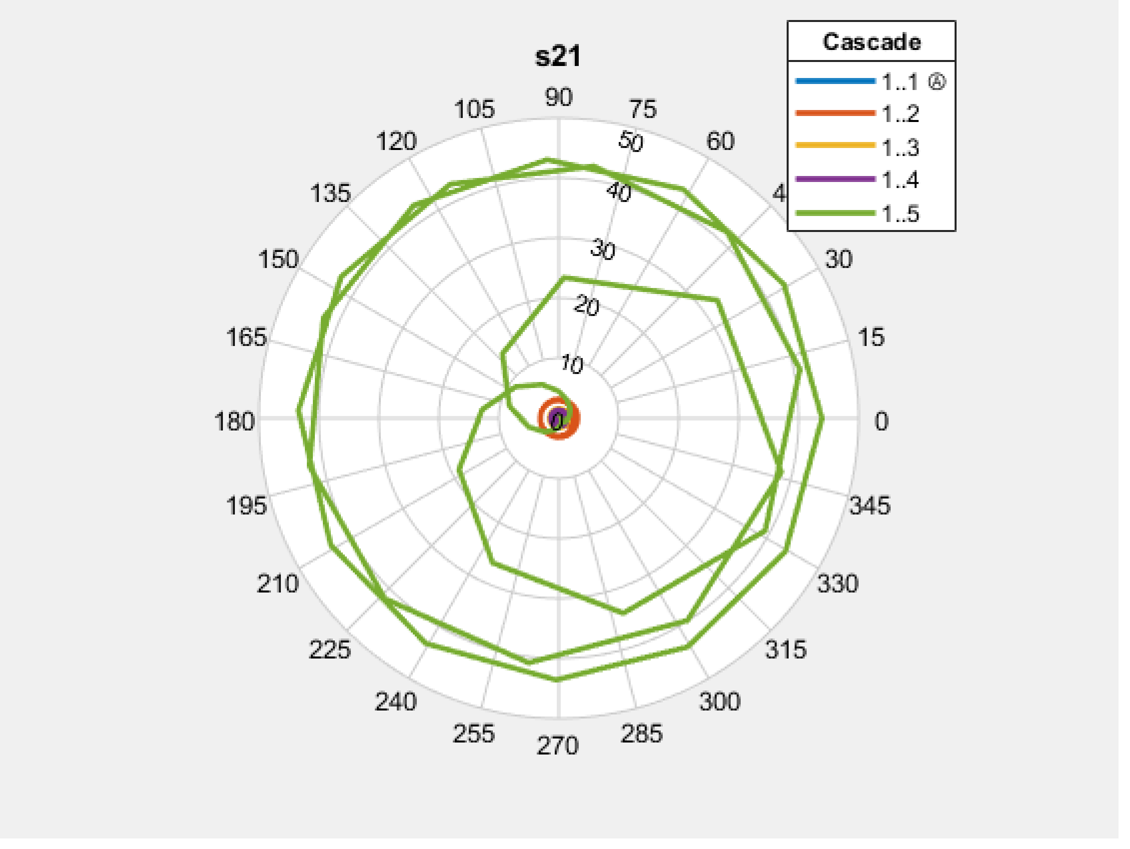


Plot sparameters of RF System on a Smith Chart and a Polar plot

```
s = smithplot(b,1,1,'GridType','ZY');
```


s11
Cascade
1..1
1..2
1..3
1..4
1..5
+j1
+j0.5
+j2
+j0.2
+j5
+j30
0.0
0.2
0.5
1
2
5
30
∞
-j30
-j0.2
-j5
-j0.5
-j2
-j1

```
p = polar(b,2,1);
```



Input Arguments

`rfobj` — Cumulative RF budget results
object (default)

Cumulative RF budget results, specified as an object.

Example: `rfplot(rfobj,'Pout')` where `rfobj` is created using `rfbudget` object.

`str` — STR values
`'Pout'` | `'GainT'` | `'NF'` | `'OIP3'` | `'IIP3` | `'SNR'`

STR values, specified as one of the following:

- `'Pout'` - Available output power (dBm)
- `'GainT'` - Transducer gain (dB)
- `'NF'` - Noise Figure (dB)
- `'OIP3'` - Output Third-Order Intercept (dBm)
- `'IIP3'` - Input Third-Order Intercept (dBm)
- `'SNR'` - Signal-to-Noise Ratio (dB)

Example: `rfplot(rfobj,'Pout')` where `'Pout'` is the available output power of an RF system obtained from the RF budget analysis.

See Also

`computeBudget` | `rfbudget` | `show`

Introduced in R2017b

getSpurFreeZoneData

Return frequency data related to the spur-free zones in multiband transmitter or receiver frequency space

Syntax

```
allfrequencyzones = getSpurFreeZoneData(hif)
```

Description

`allfrequencyzones = getSpurFreeZoneData(hif)` returns frequency data related to the spur-free zones in multiband transmitter or receiver frequency space. Each zone is a range of IF center frequencies. An IF centered in this range does not generate interference in any transmission or reception bands.

Examples

Spur-Free zones

Calculate spur-free zones.

```
h = OpenIF('IFLocation','MixerOutput');
```

Add two mixers to the system.

```
IMT1 = [99 0 21 17 26; 11 0 29 29 63; ...
       60 48 70 65 41; 90 89 74 68 87; 99 99 95 99 99];
addMixer(h,IMT1,2400e6,100e6,'low',50e6)
IMT2 = [99 0 9 12 15; 20 0 26 31 48; ...
       55 70 51 70 53; 85 90 60 70 94; 96 95 94 93 92];
addMixer(h,IMT2,3700e6,150e6,'high',50e6)
```

Check for spur-free zones.

```
sfzfreqs = getSpurFreeZoneData(h)
```

```
sfzfreqs =

   1.0e+09 *

    0.2500    0.4300
    0.5300    0.5563
    0.6438    0.7167
    1.0375    1.1125
    1.3417    1.4100
    1.4700    1.5333
    2.0750    2.3000
```

Input Arguments

`hif` — OpenIF object
object handle

OpenIF object, specified as an object handle.

Output Arguments

`allfrequencyzones` — Spur-free zones
K-by-2 matrix

Spur-free zones of a defined network, returned as a *K*-by-2 matrix. *K* is the number of spur free zones. The two columns in the matrix contain the start and stop frequencies of each spur-free zone. The first column contains the start frequencies and the second column contains the stop frequencies.

Alternative Functionality

- The `report` method displays mixer configurations, intermodulation tables, and spur-free zone information at the command line.
- The `show` method generates an interactive spur graph that shows spurious regions and spur-free zones.

See Also

Introduced in R2011b

getSpurData

Return frequency data related to the spurs in multiband transmitter or receiver frequency space

Syntax

```
allfrequencies = getSpurData(hif)
[allfrequencies,dBs,mixers,mns = getSpurData
```

Description

`allfrequencies = getSpurData(hif)`Return frequency data related to the spurs in multiband transmitter or receiver frequency space. Each spur is a range of frequencies.

`[allfrequencies,dBs,mixers,mns = getSpurData`returns relevant data for all spurs calculated by `OpenIF` object.

Examples

Spur Data

Setup the object.

```
h = OpenIF('IFLocation','MixerOutput');
```

Add two mixers to the system

```
IMT1 = [99 0 21 17 26; 11 0 29 29 63; ...
        60 48 70 65 41; 90 89 74 68 87; 99 99 95 99 99];
addMixer(h,IMT1,2400e6,100e6,'low',50e6)
IMT2 = [99 0 9 12 15; 20 0 26 31 48; ...
        55 70 51 70 53; 85 90 60 70 94; 96 95 94 93 92];
addMixer(h,IMT2,3700e6,150e6,'high',50e6)
```

Get spur free data.

```
[allspurs,dBs,mixers,mn] = getSpurData(h)

allspurs =

   1.0e+09 *

    1.9000    1.9400
    1.4100    1.4700
    0.9200    1.0000
    1.5333    1.6667
    0.4300    0.5300
    0.7167    0.8833
    1.7813    1.8188
    1.1687    1.2312
    2.3375    2.3500
    0.5563    0.6438
    1.1125    1.2875
         0    0.1125
    1.5833    1.6167
    0.7667    0.8333
    2.3000    2.3500
         0    0.1500
    1.1875    1.2125
         0    0.0375
    1.1875    1.2125
    1.5833    1.6167
    1.7813    1.8188
    1.9000    1.9400
         0    0.1667
         0    0.1875
    1.6250    2.0750
    0.8125    1.0375
         0    0.2500
    3.3750    3.6250
    1.1250    1.3417
    2.3333    2.6000
         0    0.0625
    1.7500    1.9500
    3.5625    3.6250


dBs =

    26
```

```
    63
    41
    41
    87
    87
    17
    29
    29
    65
    65
    68
    21
    29
    29
    70
     0
     0
     0
    21
    17
    26
    92
    70
    93
    93
    51
    60
    60
    94
     0
    70
    90


mixers =

     1
     1
     1
     1
     1
     1
     1
     1
```

```
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     2
     2
     2
     2
     2
     2
     2
     2
     2
     2
     2


mn =

    -4     0
    -4     1
    -4     2
    -4     2
    -4     3
    -4     3
    -3     0
    -3     1
    -3     1
    -3     2
    -3     2
    -3     3
    -2     0
    -2     1
    -2     1
```

```
-2     2
-1     0
-1     1
 1     0
 2     0
 3     0
 4     0
-4     4
-3     3
-3     4
-3     4
-2     2
-2     3
-2     3
-2     4
-1     1
-1     2
-1     3
```

Input Arguments

`hif` — OpenIF object

object handle

OpenIF object, specified as an object handle.

Output Arguments

`allfrequencies` — Start and stop frequencies of spur data

K-by-2 matrix

Start and stop frequencies of spur data, returned as a *K*-by-2 matrix or *K*-by-1 matrix. *K* is the number of spurs. The two columns in the matrix contain the start and stop frequencies of each spur-free zone. The first column contains the start frequencies and the second column contains the stop frequencies.

`dBs` — Decibel carpet value (dBc) of spur data

K-by-1 matrix

Decibel carpet value (dBc) value of spur data, returned as a *K*-by-1 matrix. Each K is a dBc value (relative to the output) of that spur.

`mixers` — Mixer that caused the spur
K-by-1 matrix

Mixer that caused the spur, returned as a *K*-by-1 matrix. Each K is the mixer that caused the spur.

`mns` — *M* and *N* values used to calculate the spur
K-by-2 matrix

M and *N* values used to calculate the spur, returned as a *K*-by-2 matrix.

See Also

Introduced in R2011b

report

Summarize IF planning results in command window

Syntax

```
report(hif)
```

Description

`report(hif)` returns the summary of IF planning results in command window. The summary contains:

- The IF location.
- The properties of each mixer, including RF center frequencies, bandwidths, mixing type, and intermodulation tables.

  The spur-free zones.

Examples

Values in OpenIF

Set up the object

```
h = OpenIF('IFLocation','MixerOutput');
```

Add two mixers to the system

```
IMT1 = [99 0 21 17 26; 11 0 29 29 63; ...
       60 48 70 65 41; 90 89 74 68 87; 99 99 95 99 99];
addMixer(h,IMT1,2400e6,100e6,'low',50e6)
IMT2 = [99 0 9 12 15; 20 0 26 31 48; ...
       55 70 51 70 53; 85 90 60 70 94; 96 95 94 93 92];
addMixer(h,IMT2,3700e6,150e6,'high',50e6)
```

Check for spur-free zones

```
report(h)
```

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

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

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

    Spur-Free Zones:
     250.00 -  430.00 MHz
     530.00 -  556.25 MHz
     643.75 -  716.67 MHz
       1.04 -    1.11 GHz
       1.34 -    1.41 GHz
       1.47 -    1.53 GHz
```

```
          2.08 -    2.30 GHz
```

Input Arguments

`hif` — OpenIF object
object handle

OpenIF object, specified as an object handle.

See Also

Introduced in R2011b

show

Graphical summary of all relevant spurs and spur-free zones

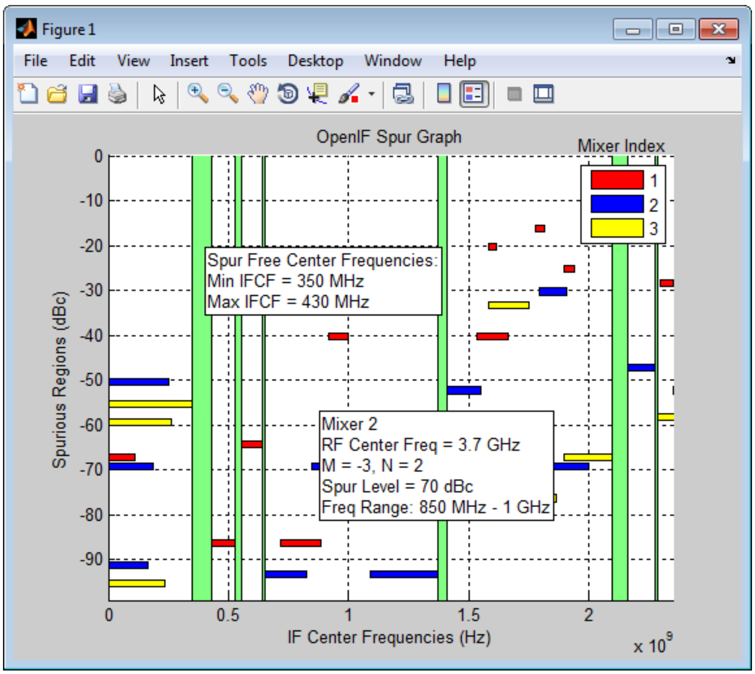
Syntax

`show(hif)`

Description

`show(hif)` produces a spur graph of the `OpenIF` object `hif`. The spur graph contains:

- Vertical green bands, representing spur-free zones.
- Horizontal colored bands, representing spurious regions.



Spur-free zones are ranges of possible IF center frequencies that are free from intermodulation distortion. Depending on the configuration of the mixers in `hif`, spur-free zones may not appear. Clicking a spur-free zone produces a tool tip, which displays information about the spur-free zone:

- **Min IFCF** — The minimum possible IF center frequency f_{IF} for the corresponding spur-free zone.
- **Max IFCF** — The maximum IF center frequency f_{IF} for the corresponding spur-free zone.

Spurious regions contain intermodulation products from at least one mixer. The color of a spur on the spur graph indicates which mixer generates the spur, according to the legend on the spur graph. Clicking a spurious region produces a tool tip, which displays information about the spur:

- **RF Center Freq** — The RF center frequency f_{RF} of the mixer that generates the spur
- **M**, **N** — The coefficients in the equation $|Mf_{RF} - N(f_{RF} \pm f_{IF})|$ (down-conversion) or the equation $|Mf_{IF} + N(f_{RF} \pm f_{IF})|$. Injection type of the receiver determines the sign in the equations. These coefficients refer to the particular mixing product that generates the spurious region.
- **Spur Level** — The difference in magnitude between a signal at 0 dBc and the spur. If you set `hif.SpurLevel` to a number greater than this value, then `hif` does not report the region as spurious.
- **Freq Range** — The frequency range of the spurious region. Choosing an IF center frequency in this range causes interference with the intermodulation product corresponding to the spur.

Examples

Show Spur-Free Zones

Set up the object

```
h = OpenIF('IFLocation','MixerOutput');
```

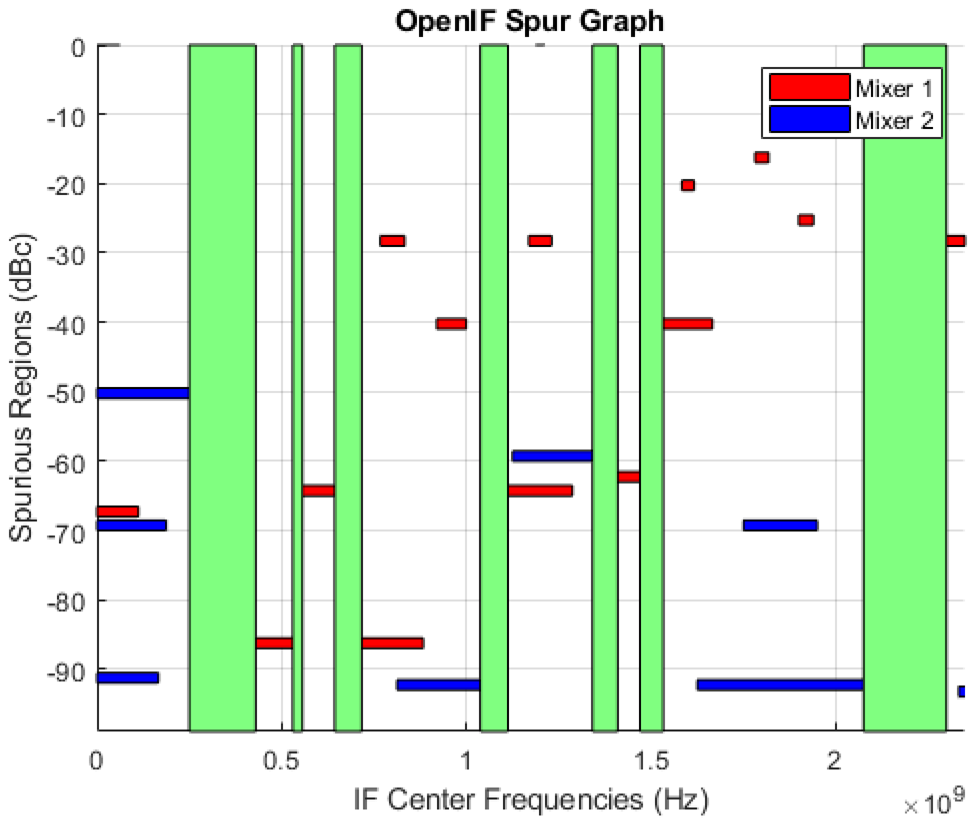
Add two mixers to the system

```
IMT1 = [99 0 21 17 26; 11 0 29 29 63; ...
       60 48 70 65 41; 90 89 74 68 87; 99 99 95 99 99];
addMixer(h,IMT1,2400e6,100e6,'low',50e6)
```

```
IMT2 = [99 0 9 12 15; 20 0 26 31 48; ...
       55 70 51 70 53; 85 90 60 70 94; 96 95 94 93 92];
addMixer(h,IMT2,3700e6,150e6,'high',50e6)
```

Check for spur-free zones

```
show(h)
```



Input Arguments

hif — OpenIF object
object handle

OpenIF object, specified as an object handle.

See Also

Introduced in R2011b

circle

Draw circles on Smith Chart

Syntax

```
[hsm1] = circle(rfcktobject,freq,
type1,value1,....,typen,valuen,hsm1)
[hlines,hsm] = circle(rfcktobject,freq,
type1,value1,....,typen,valuen,hsm)
```

Description

`[hsm1] = circle(rfcktobject,freq,
type1,value1,....,typen,valuen,hsm1)` draws the specified circles on a Smith chart created using the `smithplot` function. The syntax returns an existing smithplot handle.

`[hlines,hsm] = circle(rfcktobject,freq,
type1,value1,....,typen,valuen,hsm)` draws the specified circles on a Smith chart. This syntax returns vector handles of line objects and handles of the Smith chart.

Examples

Draw Circles on Smith Chart Created using `smithplot` Function

Create an amplifier object from `default.s2p`.

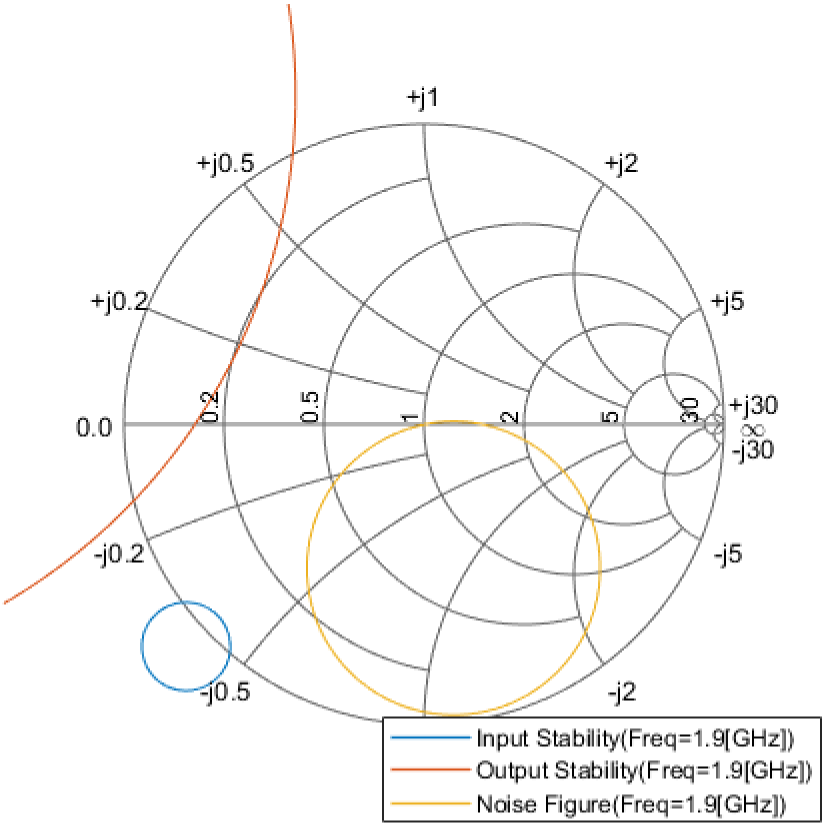
```
amp = read(rfckt.amplifier,'default.s2p');
```

Plot the noise figure of the amplifier 1.9 GHz using a Smith chart created using `smithplot` function

```
fc = 1.9e9;
h = smithplot
```

```
circle(amp,fc,'Stab','In','Stab','Out','NF',10.396,h);
legend('Location','SouthEast')
```

```
h = 

  smithplot with properties:

         Data: []
    Frequency: []

   Show <a href="matlab:if exist('h','var'),internal.polariCommon.getForDisplay('h',h,
```



- "Designing Matching Networks (Part 1: Networks with an LNA and Lumped Elements)"

Input Arguments

`rfcktobject` — RF Toolbox rfckt object
object handle

RF Toolbox rfckt object, specified as an object handle,

`freq` — Single frequency point of interest
scalar

Single frequency point of interest, specified as a scalar in Hz.

Data Types: `double`

`type1,value1,....,typen,valuen` — Type value pairs specifying circles to plots
character vector

Type value pairs specifying circles to plots, specified as a character vector.

The following table lists the supported circle type options:

type	Definition
`'Ga'`	Constant available power gain circle
`'Gp'`	Constant operating power gain circle
`'Stab'`	Stability circle
`'NF'`	Constant noise figure circle
`'R'`	Constant resistance circle
`'X'`	Constant reactance circle
`'G'`	Constant conductance circle
`'B'`	Constant susceptance circle
`'Gamma'`	Constant reflection magnitude circle

The following table lists the circle value options:

value	Definition
`'Ga'`	Scalar or vector of gains in dB
`'Gp'`	Scalar or vector of gains in dB
`'Stab'`	`'in'` or `'source'` for input/source stability circle; `'out'` or `'load'` for output/load stability circle
`'NF'`	Scalar or vector of noise figures in dB
`'R'`	Scalar or vector of normalized resistance
`'X'`	Scalar or vector of normalized reactance
`'G'`	Scalar or vector of normalized conductance
`'B'`	Scalar or vector of normalized susceptance

value	Definition
'Gamma'	Scalar or vector of non-negative reflection magnitude

Data Types: `char`

hsm1 — Existing Smith plot handle
object handle

Existing Smith chart handle created using `smithplot` function, specified as an object handle. You can obtain the object handle using `hsm1 = smithplot('gco')`.

hsm — Existing Smith chart handle
object handle

Existing Smith chart handle, specified as an object handle.

Output Arguments

hlines — Line objects for circle specifications
vector of line handle

Line objects for circle specifications, returned as a vector of line handles.

hsm — Smith chart
object handle

Smith chart, returned as an object handle.

hsm1 — Smith chart created using smithplot function
object handle

Smith chart created using `smithplot` function, returned as an object handle.

See Also

`smithplot`

Topics

"Designing Matching Networks (Part 1: Networks with an LNA and Lumped Elements)"

Introduced in R2007b

Functions — Alphabetical List

abcd2h

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.

Examples

ABCD-Parameters to H-Parameters

Convert ABCD-parameters to H-parameters. Define a matrix for ABCD-parameters.

```
A =      0.999884396265344 +  0.000129274757618717i;
B =      0.314079483671772 +      2.51935878310427i;
C = -6.56176712108866e-007 + 6.67455405306704e-006i;
D =      0.999806365547959 +  0.000247230611054075i;
abcd_params = [A,B; C,D];
```

Convert the result to H-parameters.

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

```
h_params = 2×2 complex

   0.3148 + 2.5198i   0.9999 + 0.0001i
  -1.0002 + 0.0002i  -0.0000 + 0.0000i
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-11.

See Also

`abcd2s` | `abcd2y` | `abcd2z` | `h2abcd` | `s2h` | `y2h` | `z2h`

Introduced before R2006a

abcd2s

Convert ABCD-parameters to S-parameters

Syntax

```
s_params = abcd2s(abcd_params,z0)
```

Description

`s_params = abcd2s(abcd_params,z0)` converts the ABCD-parameters `abcd_params` into the scattering parameters `s_params`. The `abcd_params` input is a complex $2N$-by-$2N$-by-M array, representing M $2N$-port ABCD-parameters. `z0` is the reference impedance; its default is 50 ohms. The function assumes that the ABCD-parameter matrices have distinct A, B, C, and D submatrices:

$$\begin{bmatrix} [A] & [B] \\ [C] & [D] \end{bmatrix}$$

`s_params` is a complex $2N$-by-$2N$-by-M array, representing M $2N$-port S-parameters.

Examples

Convert ABCD-Parameters to S-Parameters

Define a matrix of ABCD-parameters.

```
A =      0.999884396265344 +  0.000129274757618717i;
B =      0.314079483671772 +      2.51935878310427i;
C = -6.56176712108866e-007 + 6.67455405306704e-006i;
D =      0.999806365547959 +  0.000247230611054075i;
abcd_params = [A,B; C,D]
```

```
abcd_params = 2×2 complex
```

```
   0.9999 + 0.0001i   0.3141 + 2.5194i
  -0.0000 + 0.0000i   0.9998 + 0.0002i
```

Convert these ABCD parameters to S-parameters.

```
s_params = abcd2s(abcd_params)
```

```
s_params = 2×2 complex

   0.0038 + 0.0248i   0.9961 - 0.0250i
   0.9964 - 0.0254i   0.0037 + 0.0249i
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see “RF Network Parameter Objects” on page 2-11.

See Also

abcd2h | abcd2y | abcd2z | s2abcd | s2h | y2h | z2h

Introduced before R2006a

abcd2y

Convert ABCD-parameters to Y-parameters

Syntax

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

Description

`y_params = abcd2y(abcd_params)` converts the ABCD-parameters `abcd_params` into the admittance parameters `y_params`. The `abcd_params` input is a complex 2N-by-2N-by-M array, representing M 2N-port ABCD-parameters. The function assumes that the ABCD-parameter matrices have distinct A, B, C, and D submatrices:

$$\begin{bmatrix} [A] & [B] \\ [C] & [D] \end{bmatrix}$$

`y_params` is a complex 2N-by-2N-by-M array, representing M 2N-port Y-parameters.

Examples

Convert ABCD-Parameters to Y-Parameters

Define a matrix of ABCD parameters.

```
A =      0.999884396265344 +  0.000129274757618717i;
B =      0.314079483671772 +      2.51935878310427i;
C = -6.56176712108866e-007 + 6.67455405306704e-006i;
D =      0.999806365547959 +  0.000247230611054075i;
abcd_params = [A,B; C,D]
```

```
abcd_params = 2×2 complex

   0.9999 + 0.0001i   0.3141 + 2.5194i
```

```
  -0.0000 + 0.0000i   0.9998 + 0.0002i
```

Convert these ABCD-parameters to Y-parameters.

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

```
y_params = 2×2 complex

   0.0488 - 0.3908i  -0.0489 + 0.3907i
  -0.0487 + 0.3909i   0.0488 - 0.3908i
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see “RF Network Parameter Objects” on page 2-11.

See Also

abcd2h | abcd2s | abcd2z | h2y | s2y | y2abcd | z2y

Introduced before R2006a

abcd2z

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*N*-by-2*N*-by-*M* array, representing *M* 2*N*-port ABCD-parameters. The function assumes that the ABCD-parameter matrices have distinct *A*, *B*, *C*, and *D* submatrices:

$$\begin{bmatrix} [A] & [B] \\ [C] & [D] \end{bmatrix}$$

`z_params` is a complex 2*N*-by-2*N*-by-*M* array, representing *M* 2*N*-port Z-parameters.

Examples

ABCD-Parameters to Z-Parameters

Convert ABCD-parameters to Z-parameters. Define a matrix for ABCD-parameters.

```
A =      0.999884396265344 +  0.000129274757618717i;
B =      0.314079483671772 +      2.51935878310427i;
C = -6.56176712108866e-007 + 6.67455405306704e-006i;
D =      0.999806365547959 +  0.000247230611054075i;
abcd_params = [A,B; C,D];
```

Convert the result to Z-parameters.

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

```
z_params = 2×2 complex
10^5 ×

  -0.1457 - 1.4837i  -0.1453 - 1.4835i
  -0.1459 - 1.4839i  -0.1455 - 1.4836i
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see “RF Network Parameter Objects” on page 2-11.

See Also

abcd2h | abcd2s | abcd2y | h2y | y2abcd | z2abcd

Introduced before R2006a

cascadesparams

Combine S-parameters to form cascaded network

Syntax

```
s_params = cascadesparams(s1_params,s2_params,...,sk_params)
hs = cascadesparams(hs1,hs2,...,hsk)
s_params = cascadesparams(s1_params,s2_params,...,sk_params,Kconn)
```

Description

`s_params = cascadesparams(s1_params,s2_params,...,sk_params)` cascades the scattering parameters of the K input networks described by the S-parameters `s1_params` through `sk_params`. The function stores the S-parameters of the cascade in `s_params`. Each of the input networks must be a 2N-port network described by a 2N-by-2N-by-M array of S-parameters. All networks must have the same reference impedance.

`hs = cascadesparams(hs1,hs2,...,hsk)` cascades K S-parameter objects to create the cascaded network `hs`. The function checks that the `Impedance` and `Frequencies` properties of each object are equal and that the `Parameters` property contains a 2N-by-2N-by-M array of S-parameters.

`cascadesparams` assumes that you are using the port ordering given in the following illustration.

Based on this ordering, the function connects ports `N + 1` through `2N` of the first network to ports `1` through `N` of the second network. Therefore, when you use this syntax:

- Each network has an even number of ports
- Every network in the cascade has the same number of ports.

To use this function for S-parameters with different port arrangements, use the `snp2smp` function to reorder the port indices before cascading the networks.

`s_params = cascadesparams(s1_params,s2_params,...,sk_params,Kconn)` cascades the scattering parameters of the K input networks described by the S-parameters `s1_params` through `sk_params`. The function creates a cascaded network based on the number of cascade connections between networks, specified by `Kconn`. `Kconn` must be a positive scalar or vector of size $K - 1$.

- If `Kconn` is a scalar, `cascadesparams` makes the same number of connections between each pair of consecutive networks.
- If `Kconn` is a vector, the `i`th element of `Kconn` specifies the number of connections between the `i`th and the `i+1`th networks.

`cascadesparams` always connects the last `Kconn(i)` ports of the `i`th network and the first `Kconn(i)` ports of the `i+1`th network. The ports of the entire cascaded network represent the unconnected ports of each individual network, taken in order from the first network to the `n`th network.

Also, when you specify `Kconn`:

- Each network can have either an even or odd number of ports.
- Every network in the cascade can have a different number of ports.

Note The cascadesparams function uses ABCD parameters. Alternatively, one

could use sparameters and abcdparameters (or T-parameters) to cascade

sparameters together by hand (assuming identical frequencies)

Examples

Two-Port Cascaded Network

Assemble a 2-port cascaded network from two sets of 2-port S-parameters. Create two sets of 2-port S-parameters.

```
ckt1 = read(rfckt.amplifier,'default.s2p');
ckt2 = read(rfckt.passive,'passive.s2p');
freq = [2e9 2.1e9];
analyze(ckt1,freq);
analyze(ckt2,freq);
sparams_2p_1 = ckt1.AnalyzedResult.S_Parameters;
sparams_2p_2 = ckt2.AnalyzedResult.S_Parameters;
```

Cascade the S-parameters.

```
sparams_cascaded_2p = cascadesparams(sparams_2p_1,sparams_2p_2)
```

```
sparams_cascaded_2p =
sparams_cascaded_2p(:,:,1) =

  -0.4332 + 0.5779i   0.0081 - 0.0120i
   2.6434 + 1.2880i   0.5204 - 0.5918i


sparams_cascaded_2p(:,:,2) =

  -0.1271 + 0.3464i  -0.0004 - 0.0211i
   3.8700 - 0.6547i   0.4458 - 0.6250i
```

Three-Port Cascaded Network

Assemble a 3-port cascaded network from a set of 3-port S-parameters and a set of 2-port S-parameters.

Create one set of 3-port S-parameters and one set of 2-port S-parameters.

```
ckt1 = read(rfckt.passive,'default.s3p');
ckt2 = read(rfckt.amplifier,'default.s2p');
freq = [2e9 2.1e9];
analyze(ckt1,freq);
analyze(ckt2,freq);
```

```
sparams_3p = ckt1.AnalyzedResult.S_Parameters;
sparams_2p = ckt2.AnalyzedResult.S_Parameters;
```

Cascade the two sets by connecting one port between them.

```
Kconn = 1;
sparams_cascaded_3p = cascadesparams(sparams_3p,sparams_2p,Kconn)
```

```
sparams_cascaded_3p =
sparams_cascaded_3p(:,:,1) =

   0.1339 - 0.9561i   0.0325 + 0.2777i   0.0222 + 0.0092i
   0.3497 + 0.2449i   0.3130 - 0.9235i   0.0199 + 0.0255i
  -4.0617 + 5.0914i  -1.6296 + 4.7333i  -0.7133 - 0.7305i


sparams_cascaded_3p(:,:,2) =

  -0.3023 - 0.7303i   0.0635 + 0.4724i   0.0005 - 0.0220i
   0.1408 + 0.2705i  -0.1657 - 0.7749i   0.0198 - 0.0274i
   5.7709 + 2.2397i   4.1929 - 0.2165i  -0.5092 + 0.4251i
```

Three-Port Cascaded Network from S-Parameters

Assemble a 3-port cascaded network from a set of 3-port S-parameters and a set of 2-port S-parameters, connecting the second port of the 3-port network to the first port of the 2-port.

```
ckt1 = read(rfckt.passive,'default.s3p');
ckt2 = read(rfckt.amplifier,'default.s2p');
freq = [2e9 2.1e9];
analyze(ckt1,freq);
analyze(ckt2,freq);
sparams_3p = ckt1.AnalyzedResult.S_Parameters;
sparams_2p = ckt2.AnalyzedResult.S_Parameters;
```

Reorder the second and third ports of the 3-port network

```
sparams_3p_2 = snp2smp(sparams_3p,50,[1 3 2])
```

```
sparams_3p_2 =
sparams_3p_2(:,:,1) =
```

```
  -0.0073 - 0.8086i   0.1114 + 0.3027i  -0.0318 + 0.4208i
  -0.0285 + 0.4285i   0.0503 - 0.8080i   0.0898 + 0.3177i
   0.0869 + 0.3238i  -0.0701 + 0.4278i   0.1431 - 0.7986i


sparams_3p_2(:,:,2) =

  -0.2560 - 0.7399i   0.2124 + 0.2502i   0.0895 + 0.4536i
   0.1031 + 0.4867i  -0.2078 - 0.7553i   0.1989 + 0.2725i
   0.2079 + 0.2988i   0.0508 + 0.5019i  -0.1163 - 0.7761i
```

Cascade the two sets by connecting one port between them

```
Kconn = 1;
sparams_cascaded_3p_2 = cascadesparams(sparams_3p_2,...
     sparams_2p,Kconn)
```

```
sparams_cascaded_3p_2 =
sparams_cascaded_3p_2(:,:,1) =

   0.1391 - 0.9217i   0.3442 + 0.2475i   0.0180 + 0.0214i
   0.0487 + 0.3061i   0.2064 - 0.9111i   0.0190 + 0.0109i
  -1.7344 + 4.1655i  -4.2628 + 3.9827i  -0.6199 - 0.7368i


sparams_cascaded_3p_2(:,:,2) =

  -0.3058 - 0.7358i   0.1492 + 0.2216i   0.0164 - 0.0271i
   0.0714 + 0.5048i  -0.2584 - 0.7547i   0.0025 - 0.0230i
   4.6396 - 0.0736i   5.6709 + 3.0321i  -0.5803 + 0.4618i
```

Three-Port Cascade Network from Multiple S-Parameters

Assemble a 3-port cascaded network from a set of 3-port S-parameters and two sets of 2-port S-parameters.

```
ckt1 = read(rfckt.passive,'default.s3p');
ckt2 = read(rfckt.amplifier,'default.s2p');
ckt3 = read(rfckt.passive,'passive.s2p');
```

```
freq = [2e9 2.1e9];
analyze(ckt1,freq);
analyze(ckt2,freq);
analyze(ckt3,freq);
sparams_3p = ckt1.AnalyzedResult.S_Parameters;
sparams_2p_1 = ckt2.AnalyzedResult.S_Parameters;
sparams_2p_2 = ckt3.AnalyzedResult.S_Parameters;
```

Connect one port between each set of adjacent networks.

```
Kconn = [1 1];
sparams_cascaded_3p_3 = cascadesparams(sparams_3p,...
     sparams_2p_1,sparams_2p_2,Kconn)
```

```
sparams_cascaded_3p_3 =
sparams_cascaded_3p_3(:,:,1) =

   0.1144 - 0.8944i   0.0342 + 0.3273i   0.0046 + 0.0052i
   0.2861 + 0.3040i   0.2822 - 0.8643i   0.0020 + 0.0091i
  -1.6910 + 0.8202i  -1.0132 + 1.0296i   0.5275 - 0.6425i


sparams_cascaded_3p_3(:,:,2) =

  -0.2985 - 0.8130i   0.0429 + 0.4202i   0.0075 - 0.0062i
   0.2177 + 0.1692i  -0.1463 - 0.8590i   0.0149 - 0.0013i
   0.9210 + 2.5820i   1.2868 + 1.3420i   0.3627 - 0.5876i
```

Complex Three-Port Cascaded Network

Assemble a 3-port cascaded network from a set of 3-port S-parameters and two sets of 2-port S-parameters, connecting the 3-port network to both 2-port networks.

```
ckt1 = read(rfckt.passive,'default.s3p');
ckt2 = read(rfckt.amplifier,'default.s2p');
ckt3 = read(rfckt.passive,'passive.s2p');
freq = [2e9 2.1e9];
analyze(ckt1,freq);
analyze(ckt2,freq);
analyze(ckt3,freq);
sparams_3p = ckt1.AnalyzedResult.S_Parameters;
```

```
sparams_2p_1 = ckt2.AnalyzedResult.S_Parameters;
sparams_2p_2 = ckt3.AnalyzedResult.S_Parameters;
```

Cascade sparams_3p and sparams_2p_1 by connecting one port between them.

```
Kconn = 1;
sparams_cascaded_3p = cascadesparams(...
  sparams_3p, ...
  sparams_2p_1, ...
  Kconn)
```

```
sparams_cascaded_3p =
sparams_cascaded_3p(:,:,1) =

   0.1339 - 0.9561i   0.0325 + 0.2777i   0.0222 + 0.0092i
   0.3497 + 0.2449i   0.3130 - 0.9235i   0.0199 + 0.0255i
  -4.0617 + 5.0914i  -1.6296 + 4.7333i  -0.7133 - 0.7305i


sparams_cascaded_3p(:,:,2) =

  -0.3023 - 0.7303i   0.0635 + 0.4724i   0.0005 - 0.0220i
   0.1408 + 0.2705i  -0.1657 - 0.7749i   0.0198 - 0.0274i
   5.7709 + 2.2397i   4.1929 - 0.2165i  -0.5092 + 0.4251i
```

Reorder the second and third ports of the 3-port network.

```
sparams_cascaded_3p_3 = snp2smp(...
  sparams_cascaded_3p, ...
  50, ...
  [1 3 2])
```

```
sparams_cascaded_3p_3 =
sparams_cascaded_3p_3(:,:,1) =

   0.1339 - 0.9561i   0.0222 + 0.0092i   0.0325 + 0.2777i
  -4.0617 + 5.0914i  -0.7133 - 0.7305i  -1.6296 + 4.7333i
   0.3497 + 0.2449i   0.0199 + 0.0255i   0.3130 - 0.9235i


sparams_cascaded_3p_3(:,:,2) =

  -0.3023 - 0.7303i   0.0005 - 0.0220i   0.0635 + 0.4724i
   5.7709 + 2.2397i  -0.5092 + 0.4251i   4.1929 - 0.2165i
```

```
   0.1408 + 0.2705i   0.0198 - 0.0274i  -0.1657 - 0.7749i
```

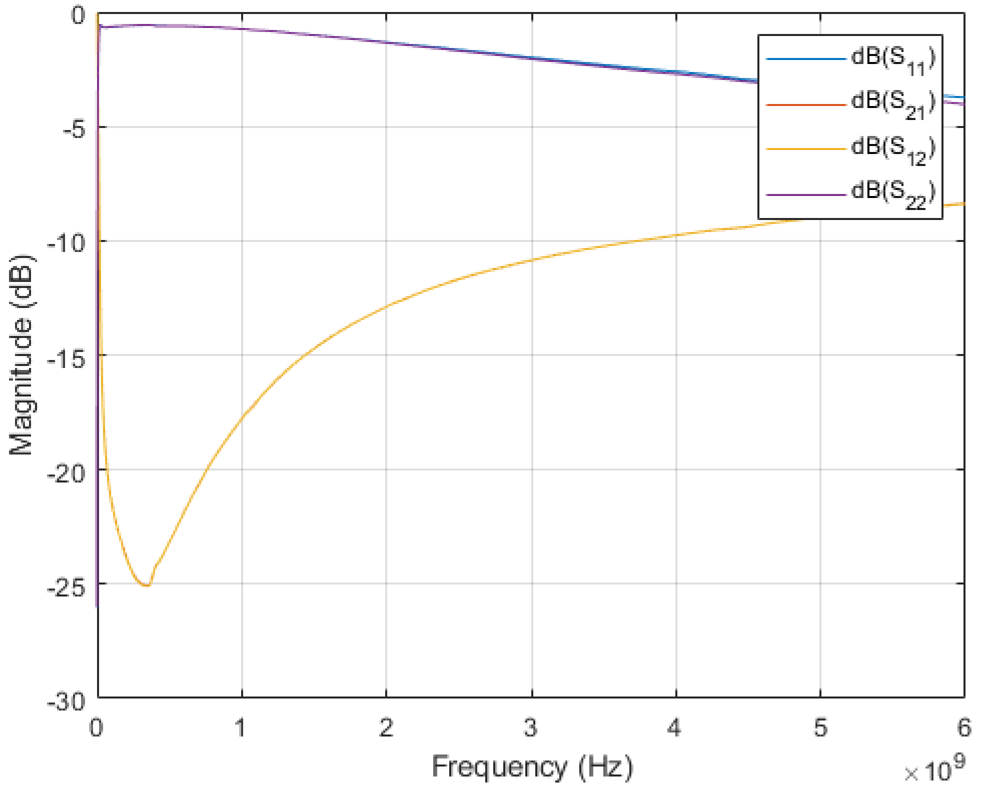
Cascade sparams_3p and sparams_2p_2 by connecting one port between them.

```
sparams_cascaded_3p_4 = cascadesparams(...
  sparams_cascaded_3p_3, ...
  sparams_2p_2, ...
  Kconn)
```

```
sparams_cascaded_3p_4 =
sparams_cascaded_3p_4(:,:,1) =

   0.1724 - 0.9106i   0.0240 + 0.0134i   0.0104 + 0.0971i
  -3.7923 + 6.1234i  -0.7168 - 0.6498i  -0.5855 + 1.6475i
   0.1214 + 0.0866i   0.0069 + 0.0090i   0.6289 - 0.6145i


sparams_cascaded_3p_4(:,:,2) =

  -0.3014 - 0.6620i   0.0072 - 0.0255i  -0.0162 + 0.1620i
   6.3709 + 2.2809i  -0.5349 + 0.3637i   1.4106 + 0.2587i
   0.0254 + 0.1011i   0.0087 - 0.0075i   0.5477 - 0.6253i
```

Using T-Parameters

```
S = sparameters('passive.s2p');
T = tparameters(S);
freq = S.Frequencies;
    for i = 1:length(freq)
        Tcasc(:,:,i) = T.Parameters(:,:,i)*T.Parameters(:,:,i);
    end
Tcasc = tparameters(Tcasc,freq);
Scasc = sparameters(Tcasc);
rfplot(Scasc)
```



See Also

deembedsparams | rfckt.cascade | s2t | snp2smp | t2s

Introduced before R2006a

copy

Copy circuit or data object

Syntax

`h2 = copy(h)`

Description

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

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

Alternatives

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

See Also

`analyze`

Topics

"Writing A Touchstone® File"

Introduced before R2006a

deembedsparams

De-embed 2N-port S-parameters

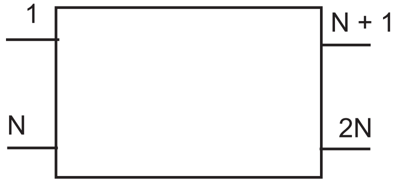
Syntax

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

`hs2 = deembedsparams(hs,hs1,hs3)`

Description

`s2_params = deembedsparams(s_params,s1_params,s3_params)` de-embeds `s2_params` from cascaded S-parameters `s_params`, by removing the effects of `s1_params` and `s3_params`. `deembedsparams` assumes that you are using the port ordering shown here:



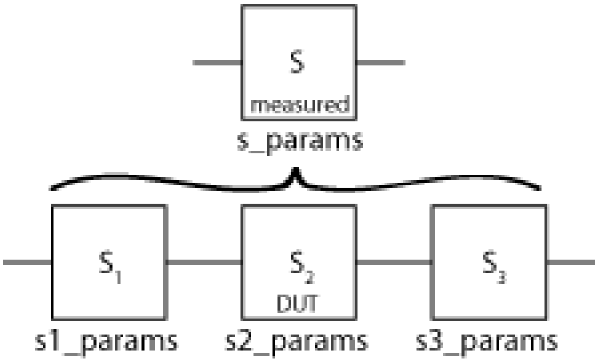
This function is ideal for situations in which the S-parameters of a `DUT` (device under test) must be de-embedded from S-parameters obtained through measurement.

`hs2 = deembedsparams(hs,hs1,hs3)` de-embeds S-parameter object, `hs2` from the chain `hs`.

Input Arguments

`s_params,s1_params,s3_params` — S-parameter data
numeric arrays

S-parameter data, specified as `2Nx2NxK` arrays of `K` 2N-port S-parameters. `s_params` is the measured S-parameter array of the cascaded network. `s1_params` represents the first network of the cascade, and `s3_params` represents the third network. The function assumes that all networks in the cascade have the same reference impedance and are measured at the same frequencies. The function assumes the configuration of the cascade shown here:



Data Types: `double`

`hs,hs1,hs3` — S-parameter objects
scalar handle objects

S-parameter objects, specified as 2N-port scalar handle objects, which can include numeric arrays of S-parameters. The function checks that the `Frequencies` and `Impedance` properties are the same for all three inputs.

Data Types: `function_handle`

Output Arguments

`s2_params` — S-parameter data
numeric arrays

S-parameter data, returned as `2Nx2NxK` arrays of `K` 2N-port s-parameters, containing de-embedded S-parameters of the DUT (device under test).

Data Types: `double`

`hs2` — S-parameter objects
scalar handle object

S-parameter objects, returned as 2N-port scalar handle objects, containing de-embedded S-parameter objects of DUT (device under test).

Data Types: function_handle

Examples

De-embed S-Parameters of a DUT from a Cascaded 2-port Network

Read measured S-parameters of the cascaded network from samplebjt2.s2p.

```
S_measuredBJT = sparameters('cascadedbackplanes.s4p');
freq = S_measuredBJT.Frequencies;
```

Calculate the S-parameters of the left fixture of the network.

```
leftpad = circuit('left');
add(leftpad,[1 2],inductor(1e-9));
add(leftpad,[2 3],capacitor(100e-15));
setports(leftpad,[1 0],[3 0],[2 0],[3 0]);
S_leftpad = sparameters(leftpad,freq)
```

```
S_leftpad = 
  sparameters: S-parameters object

       NumPorts: 4
    Frequencies: [1496x1 double]
     Parameters: [4x4x1496 double]
      Impedance: 50

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

Calculate the S-parameters of the right fixture of the network.

```
rightpad = circuit('right');
add(rightpad,[1 3],capacitor(100e-15));
add(rightpad,[1 2],inductor(1e-9));
setports(rightpad,[1 0],[3 0],[2 0],[3 0]);
S_rightpad = sparameters(rightpad,freq)
```

```
S_rightpad = 
  sparameters: S-parameters object
```

```
        NumPorts: 4
     Frequencies: [1496x1 double]
      Parameters: [4x4x1496 double]
       Impedance: 50

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

De-embed the S-parameters of the DUT. The output is stored in `S-DUT` in MATLAB® workspace.

```
S_DUT = deembedsparams(S_measuredBJT,S_leftpad,S_rightpad)
```

```
S_DUT = 
  sparameters: S-parameters object

        NumPorts: 4
     Frequencies: [1496x1 double]
      Parameters: [4x4x1496 double]
       Impedance: 50

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

De-embed S-Parameters of a DUT from a Cascaded 4-port Network

Read measured S-parameters of the cascaded network from `cascadedbackplanes.s4p`

```
S_measuredBJT = sparameters('cascadedbackplanes.s4p');
freq = S_measuredBJT.Frequencies;
```

Calculate the S-parameters of the left fixture of the network.

```
leftpad = circuit('left');
add(leftpad,[1 2],inductor(1e-9))
add(leftpad,[2 3],capacitor(100e-15))
setports(leftpad,[1 0],[3 0],[2 0],[3 0])
S_leftpad = sparameters(leftpad,freq)
```

```
S_leftpad = 
  sparameters: S-parameters object
```

```
        NumPorts: 4
     Frequencies: [1496x1 double]
      Parameters: [4x4x1496 double]
       Impedance: 50

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

Calculate the S-parameters of the right fixture of the network.

```
rightpad = circuit('right');
add(rightpad,[1 3],capacitor(100e-15))
add(rightpad,[1 2],inductor(1e-9))
setports(rightpad,[1 0],[3 0],[2 0],[3 0])
S_rightpad = sparameters(rightpad,freq)
```

```
S_rightpad =
  sparameters: S-parameters object

        NumPorts: 4
     Frequencies: [1496x1 double]
      Parameters: [4x4x1496 double]
       Impedance: 50

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

De-embed the S-parameters of the DUT. The output is stored in `S-DUT` in MATLAB® workspace.

```
S_DUT = deembedsparams(S_measuredBJT,S_leftpad,S_rightpad)
```

```
S_DUT =
  sparameters: S-parameters object

        NumPorts: 4
     Frequencies: [1496x1 double]
      Parameters: [4x4x1496 double]
       Impedance: 50

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

- “De-Embedding S-Parameters”

See Also

`cascadesparams` | `rfckt.cascade`

Topics

“De-Embedding S-Parameters”

Introduced before R2006a

g2h

Convert hybrid g-parameters to hybrid h-parameters

Syntax

```
h_params = g2h(g_params)
```

Description

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

Examples

G-Parameters to H-Parameters

Define a matrix of g-parameters.

```
g_11 = -6.55389515512306e-007 + 6.67541048071651e-006i;
g_12 =     -0.999823389146385 -  0.000246785162909241i;
g_21 =       1.00011560038266 -  0.000129304649930592i;
g_22 =      0.314441556185771 +      2.51960941000598i;
g_params = [g_11,g_12; g_21,g_22];
```

Convert to h-parameters

```
h_params = g2h(g_params);
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see “RF Network Parameter Objects” on page 2-11.

See Also

h2g

Introduced before R2006a

gamma2z

Convert reflection coefficient to impedance

Syntax

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

Description

`z = gamma2z(gamma)` converts the reflection coefficient `gamma` to the impedance `z` using a reference impedance Z_0 of 50 ohms.

`z = gamma2z(gamma,z0)` converts the reflection coefficient `gamma` to the impedance `z` by:

- Computing the normalized impedance.
- Multiplying the normalized impedance by the reference impedance Z_0.

Examples

Impedance Calculation

Calculate impedance from given reference impedance and reflection coefficient values

```
z0 = 50;
gamma = 1/3;
z = gamma2z(gamma,z0)
```

```
z = 100.0000
```

Algorithms

The following equation shows this conversion:

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

See Also

`gammain` | `gammaout` | `z2gamma`

Introduced in R2007a

gammain

Input reflection coefficient of 2-port network

Syntax

```
coefficient = gammain(s_params,z0,zl)
coefficient = gammain(hs,zl)
```

Description

`coefficient = gammain(s_params,z0,zl)` calculates the input reflection coefficient of a 2-port network. `s_params` is a complex 2-by-2-by-*M* array, representing *M* 2-port S-parameters. `z0` is the reference impedance Z_0; its default value is 50 ohms. `zl` is the load impedance Z_l; its default value is also 50 ohms. `coefficient` is an *M*-element complex vector.

`coefficient = gammain(hs,zl)` calculates the input reflection coefficient of the 2-port network represented by the S-parameter object `hs`.

Examples

Input Reflection Coefficient Calculation

Calculate the input reflection coefficients at each index of an S-parameter array.

```
ckt = read(rfckt.amplifier,'default.s2p');
s_params = ckt.NetworkData.Data;
z0 = ckt.NetworkData.Z0;
zl = 100;
coefficient = gammain(s_params,z0,zl)
```

```
coefficient = 191×1 complex

  -0.7247 - 0.4813i
```

```
  -0.7323 - 0.4707i
  -0.7397 - 0.4601i
  -0.7470 - 0.4495i
  -0.7542 - 0.4389i
  -0.7612 - 0.4284i
  -0.7682 - 0.4179i
  -0.7750 - 0.4075i
  -0.7817 - 0.3972i
  -0.7883 - 0.3870i
       ⋮
```

Algorithms

`gammain` uses the formula

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

where

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

See Also

`gamma2z` | `gammaml` | `gammams` | `gammaout` | `vswr`

Introduced before R2006a

gammaml

Load reflection coefficient of 2-port network

Syntax

```
coefficient = gammaml(s_params)
coefficient = gammaml(hs)
```

Description

`coefficient = gammaml(s_params)` calculates the load reflection coefficient of a 2-port network required for simultaneous conjugate match.

`s_params` is a complex 2-by-2-by-M array, representing M 2-port S-parameters, S_{ij}. `coefficient` is an M-element complex vector.

`coefficient = gammaml(hs)` calculates the load reflection coefficient of the 2-port network represented by the S-parameter object `hs`.

Examples

Load Reflection Coefficient Calculation

Calculate the load reflection coefficient using network data from a file

```
ckt = read(rfckt.amplifier,'default.s2p');
s_params = ckt.NetworkData.Data;
coefficient = gammaml(s_params);
```

Algorithms

The function calculates `coefficient` using the equation

$$\Gamma_{ML} = \frac{B_2 \pm \sqrt{B_2{}^2 - 4\left|C_2{}^2\right|}}{2C_2}$$

where

$$B_2 = 1 - \left|S_{11}{}^2\right| + \left|S_{22}{}^2\right| - \left|\Delta^2\right|$$
$$C_2 = S_{22} - \Delta \cdot S_{11}^*$$
$$\Delta = S_{11}S_{22} - S_{12}S_{21}$$

See Also

`gammain` | `gammams` | `gammaout` | `stabilityk`

Introduced in R2008a

gammams

Source reflection coefficient of 2-port network

Syntax

```
coefficient = gammams(s_params)
coefficient = gammams(hs)
```

Description

`coefficient = gammams(s_params)` calculates the source reflection coefficient of a 2-port network required for simultaneous conjugate match. `s_params` is a complex 2-by-2-by-*M* array, representing *M* 2-port S-parameters. `coefficient` is an *M*-element complex vector.

`coefficient = gammams(hs)` calculates the source reflection coefficient of the 2-port network represented by the S-parameter object `hs`.

Examples

Source Reflection Coefficient Calculation

Calculate the source reflection coefficient using network data from a file.

```
ckt = read(rfckt.amplifier,'default.s2p');
s_params = ckt.NetworkData.Data;
coefficient = gammams(s_params)
```

```
coefficient = 191×1 complex

  -0.7247 + 0.4813i
  -0.7324 + 0.4723i
  -0.7401 + 0.4632i
  -0.7478 + 0.4541i
```

```
-0.7554 + 0.4449i
-0.7630 + 0.4357i
-0.7704 + 0.4264i
-0.7778 + 0.4170i
-0.7850 + 0.4075i
-0.7921 + 0.3980i
     ⋮
```

Algorithms

The function calculates `coefficient` using the equation

$$\Gamma_{MS} = \frac{B_1 \pm \sqrt{B_1{}^2 - 4\left|C_1{}^2\right|}}{2C_1}$$

where

$$B_1 = 1 + \left|S_{11}{}^2\right| - \left|S_{22}{}^2\right| - \left|\Delta^2\right|$$

$$C_1 = S_{11} - \Delta \cdot S_{22}^*$$

$$\Delta = S_{11}S_{22} - S_{12}S_{21}$$

See Also

`gammain` | `gammaml` | `gammaout` | `stabilityk`

Introduced in R2008a

gammaout

Output reflection coefficient of 2-port network

Syntax

```
coefficient = gammaout(s_params,z0,zs)
coefficient = gammaout(hs,zs)
```

Description

`coefficient = gammaout(s_params,z0,zs)` calculates the output reflection coefficient of a 2-port network.

`s_params` is a complex 2-by-2-by-M array, representing M 2-port S-parameters. `z0` is the reference impedance Z_0; its default is 50 ohms. `zs` is the source impedance Z_s; its default is also 50 ohms. `coefficient` is an M-element complex vector.

`coefficient = gammaout(hs,zs)` calculates the output reflection coefficient of the 2-port network represented by the S-parameter object `hs`.

Examples

Output Reflection Coefficient Calculation

Calculate the output reflection coefficient using network data from a file.

```
ckt = read(rfckt.amplifier,'default.s2p');
s_params = ckt.NetworkData.Data;
z0 = ckt.NetworkData.Z0;
zs = 100;
coefficient = gammaout(s_params,z0,zs)
```

```
coefficient = 191×1 complex
```

```
  -0.0741 - 0.3216i
  -0.0765 - 0.3184i
  -0.0787 - 0.3152i
  -0.0809 - 0.3121i
  -0.0829 - 0.3090i
  -0.0848 - 0.3059i
  -0.0867 - 0.3029i
  -0.0884 - 0.3000i
  -0.0900 - 0.2971i
  -0.0915 - 0.2943i
       ⋮
```

Algorithms

The function calculates `coefficient` using the equation

$$\Gamma_{out} = S_{22} + \frac{S_{12}S_{21}\Gamma_S}{1 - S_{11}\Gamma_S}$$

where

$$\Gamma_S = \frac{Z_s - Z_0}{Z_s + Z_0}$$

See Also

`gamma2z` | `gammain` | `gammaml` | `gammams` | `vswr`

Introduced before R2006a

getdata

Data object containing analyzed result of specified circuit object

Syntax

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

Description

`hd = getdata(h)` returns a handle, `hd`, to the `rfdata.data` object containing the analysis data, if any, for circuit (`rfckt`) object `h`. If there is no analysis data, `getdata` displays an error message.

Note Before calling `getdata`, use the `analyze` function to perform a frequency domain analysis for the circuit (`rfckt`) object. Perform this action for all circuit objects except `rfckt.amplifier`, `rfckt.datafile`, and `rfckt.mixer`. When you create an `rfckt.amplifier`, `rfckt.datafile`, or `rfckt.mixer` object by reading data from a file, RF Toolbox software automatically creates an `rfdata.data` object. RF Toolbox stores data from the file as properties of the data object. You can use the `getdata` function, without first calling `analyze`, to retrieve the handle of the `rfdata.data` object.

Introduced before R2006a

h2abcd

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.

Examples

H-Parameters to ABCD-Parameters

Define a matrix of h-parameters.

```
h_11 =      0.314441556185771 +      2.51960941000598i;
h_12 =      0.999823389146385 -  0.000246785162909241i;
h_21 =     -1.000115600382660 -  0.0001293046499305921i;
h_22 = -6.55389515512306e-007 + 6.67541048071651e-006i;
h_params = [h_11,h_12; h_21,h_22];
```

Convert to ABCD-parameters.

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

```
abcd_params = 2×2 complex

   0.9998 - 0.0002i   0.3147 + 2.5193i
  -0.0000 + 0.0000i   0.9999 - 0.0001i
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see “RF Network Parameter Objects” on page 2-11.

See Also

abcd2h | h2s | h2y | h2z | s2abcd | y2abcd | z2abcd

Introduced before R2006a

h2g

Convert hybrid h-parameters to hybrid g-parameters

Syntax

```
g_params = h2g(h_params)
```

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.

Examples

H-Parameters to G-Parameters

Define a matrix of h-parameters.

```
h_11 =      0.314441556185771 +      2.51960941000598i;
h_12 =      0.999823389146385 -  0.000246785162909241i;
h_21 =     -1.000115600382660 -  0.0001293046499305921;
h_22 = -6.55389515512306e-007 + 6.67541048071651e-006i;
h_params = [h_11,h_12; h_21,h_22];
```

Convert to G-parameters.

```
g_params = h2g(h_params)
```

```
g_params = 2×2 complex

  -0.0000 + 0.0000i  -0.9999 + 0.0001i
   1.0002 + 0.0002i   0.3142 + 2.5198i
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see “RF Network Parameter Objects” on page 2-11.

See Also

g2h | h2abcd | h2s | h2y | h2z

Introduced before R2006a

h2s

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.

Examples

H-Parameters to S-Parameters

Define a matrix of h-parameters.

```
h_11 =      0.314441556185771 +      2.51960941000598i;
h_12 =      0.999823389146385 -  0.000246785162909241i;
h_21 =     -1.000115600382660 -  0.000129304649930592i;
h_22 = -6.55389515512306e-007 + 6.67541048071651e-006i;
h_params = [h_11,h_12; h_21,h_22];
```

Convert to S-parameters.

```
s_params = h2s(h_params)
```

```
s_params = 2×2 complex

   0.0037 + 0.0248i   0.9961 - 0.0254i
   0.9964 - 0.0250i   0.0038 + 0.0249i
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-11.

See Also

abcd2s | h2abcd | h2y | h2z | y2s | z2s

Introduced before R2006a

h2y

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.

Examples

H-Parameters to y-Parameters

Define a matrix of h-parameters.

```
h_11 =      0.314441556185771 +      2.51960941000598i;
h_12 =      0.999823389146385 -  0.000246785162909241i;
h_21 =     -1.000115600382660 -  0.000129304649930592i;
h_22 = -6.55389515512306e-007 + 6.67541048071651e-006i;
h_params = [h_11,h_12; h_21,h_22];
```

Convert to y-parameters.

```
y_params = h2y(h_params)
```

```
y_params = 2×2 complex

   0.0488 - 0.3908i  -0.0487 + 0.3907i
  -0.0488 + 0.3908i   0.0487 - 0.3908i
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see “RF Network Parameter Objects” on page 2-11.

See Also

abcd2z | h2abcd | h2s | h2y | h2y | s2z | y2z | z2h

Introduced before R2006a

h2z

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.

Examples

H-Parameters to Z-Parameters

Define a matrix of h-parameters.

```
h_11 =      0.314441556185771 +      2.51960941000598i;
h_12 =      0.999823389146385 -  0.000246785162909241i;
h_21 =     -1.000115600382660 -  0.000129304649930592i;
h_22 = -6.55389515512306e-007 + 6.67541048071651e-006i;
h_params = [h_11,h_12; h_21,h_22];
```

Convert to z-parameters.

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

```
z_params = 2×2 complex
10^5 ×

  -0.1458 - 1.4836i  -0.1460 - 1.4834i
```

```
  -0.1455 - 1.4839i  -0.1457 - 1.4837i
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see “RF Network Parameter Objects” on page 2-11.

See Also

abcd2z | h2abcd | h2s | h2y | s2z | y2z | z2h

Introduced before R2006a

ispassive

Check passivity of N-port S-parameters

Syntax

```
[result, idx_nonpassive]= ispassive(sparams)
[ ___ ]= ispassive(sparams,'Impedance',z0)
[ ___ ]= ispassive(fit_obj)
```

Description

`[result, idx_nonpassive]= ispassive(sparams)` checks the passivity of S-parameters object or data. If the S-parameters are passive at every frequency, then the result is `true`. Otherwise, the result is `false`. It also optionally returns `idx_non_passive`, the indices of the non-passive S-parameters.

`[ ___ ]= ispassive(sparams,'Impedance',z0)` checks the passivity of N-port S-parameters data, that is referenced to the impedance value in the name-value pair, `'Impedance',z0`. The impedance can be in general complex.

`[ ___ ]= ispassive(fit_obj)` checks the passivity of a scalar `rfmodel.rational` object. The `rfmodel.rational` object is the output of a rational fit function.

Input Arguments

`sparams` — S-parameters
scalar S-parameters object | complex *N*-by-*N*-by-*K* array

S-parameters, specified as one of the following:

- A scalar S-parameters object
- A complex *N*-by-*N*-by-*K* array for N-port S-parameters data.

`sparams_data` — S-parameter data referenced to z_0
N-by-*N*-by-*K* numeric matrix

S-parameter data referenced to z_0, specified as an N-by-N-by-K numeric matrix.

z0 — Reference impedance
50 (default) | complex scalar

Reference impedance, specified as a complex scalar or vector.

fit_obj — Output of rational fit function
scalar `rfmodel.rational` object

Output of rational fit function, specified as a scalar `rfmodel.rational` object.

Output Arguments

result — Passivity of S-parameter data
logical scalar

Passivity of s-parameter data, returned as a logical scalar of `0` or `1`. If all the S-parameters are passive, then `ispassive` sets `flag` equal to `1` (`true`). Otherwise, `flag` is equal to `0` (`false`). If `flag` is `true`, `idx_non_passive` is empty.

idx_nonpassive — Indices that correspond to the frequencies
vector of numeric integers

Indices that correspond to the frequencies where the S-parameter is not passive, returned as vector of numeric integers.

Examples

Check Passivity of S-parameter Data

Read a Touchstone data file.

```
S = sparameters('measured.s2p');
```

Check the passivity of the S-parameters.

```
[passivevar,idx] = ispassive(S);
passivevar
```

```
passivevar = logical
   0
```

Get the nonpassive S-parameters.

```
if ~passivevar
    nonpassivevals = S.Parameters(:,:,idx);
end
```

Passivity of N-port S-parameter Data

Convert `passive.s2p` Touchstone file to an nport object.

```
nobj = nport('passive.s2p');
```

Convert the n-port object, `nobj` to s-parameter object.

```
sobj = sparameters(nobj)
```

```
sobj = 
  sparameters: S-parameters object

       NumPorts: 2
    Frequencies: [202x1 double]
     Parameters: [2x2x202 double]
      Impedance: 50

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

Find the passivity of n-port sparameter data at impedance value, 60.

```
ispassive(sobj.Parameters,'Impedance',60)
```

```
ans = logical
   1
```

Passivity of Rationalfit Object

Converted `measured.s2p` to S-parameter object.

```
S = sparameters('measured.s2p');
```

Extract the S21 parameters and the frequencies of the s-parameters.

```
s21 = rfparam(S,2,1);
freq = S.Frequencies;
```

Rationalfit S21 data.

```
fit = rationalfit(freq,s21);
```

Check if the rationalfit of S21 data is passive.

```
ispass = ispassive(fit)
```

```
ispass = logical
   1
```

See Also

`rationalfit` | `rfmodel.rational.ispassive` | `s2tf` | `snp2smp`

Introduced in R2009b

makepassive

Make N-port S-parameters passive

Syntax

`sparams_passive = makepassive(sparams)`

Description

`sparams_passive = makepassive(sparams)` alters non-passive N-port S-parameters to make them passive. `makepassive` will error if the singular values at a frequency are too large. Reference impedance for S-parameters are assumed real and positive.

Input Arguments

`sparams` — S-parameters
scalar S-parameters object | complex *N*-by-*N*-by-*K* array

S-parameters specified as one of the following:

- A scalar S-parameters object
- A complex *N*-by-*N*-by-*K* array for N-port S-parameters data.

Output Arguments

`sparams_passive` — Passive S-parameters
S-parameter object

Passive S-parameters, returned as an s-parameter object.

Note The `makepassive` function uses a purely mathematical method to calculate `sparams_passive`. As a result, the array `sparams_passive` does not represent the same network as sparams, unless sparams and sparams_passive are equal. The more closely sparams represents a passive network, the better the approximation sparams_passive is to that network. Therefore, makepassive generates the most realistic results when sparams is active only due to small numerical errors.

Examples

Make S-Parameters Passive

Convert measured.s2p to S-parameter object.

```
S = sparameters('measured.s2p');
```

Check if the S-parameter object is passive.

```
ispassive(S)
```

```
ans = logical
   0
```

Make the S-parameters data passive using makepassive function.

```
S_new = makepassive(S);
```

Check if the new S-parameter object is passive.

```
ispassive(S_new)
```

```
ans = logical
   1
```

See Also

ispassive

Introduced in R2010a

powergain

Power gain of 2-port network

Syntax

```
g = powergain(s_params,z0,zs,zl,'Gt')
g = powergain(s_params,z0,zs,'Ga')
g = powergain(s_params,z0,zl,'Gp')
g = powergain(s_params,'Gmag')
g = powergain(s_params,'Gmsg')

g = powergain(hs,zs,zl,'Gt')
g = powergain(hs,zs,'Ga')
g = powergain(hs,zl,'Gp')
g = powergain(hs,'Gmag')
g = powergain(hs,'Gmsg')
```

Description

`g = powergain(s_params,z0,zs,zl,'Gt')` calculates the transducer power gain of the 2-port network `s_params`.

`g = powergain(s_params,z0,zs,'Ga')` calculates the available power gain of the 2-port network.

`g = powergain(s_params,z0,zl,'Gp')` calculates the operating power gain of the 2-port network.

`g = powergain(s_params,'Gmag')` calculates the maximum available power gain of the 2-port network.

`g = powergain(s_params,'Gmsg')` calculates the maximum stable gain of the 2-port network.

`g = powergain(hs,zs,zl,'Gt')` calculates the transducer power gain of the network represented by the S-parameter object `hs`.

`g = powergain(hs,zs,'Ga')` calculates the available power gain of the network represented by the S-parameter object `hs`.

`g = powergain(hs,zl,'Gp')` calculates the operating power gain of the network represented by the S-parameter object `hs`.

`g = powergain(hs,'Gmag')` calculates the maximum available power gain of the network represented by the S-parameter object `hs`.

`g = powergain(hs,'Gmsg')` calculates the maximum stable gain of the network represented by the S-parameter object `hs`.

Input Arguments

`hs` — 2-port S-parameters
S-parameter object

2-port S-parameters, specified as an RF Toolbox S-parameter object.

`s_params` — 2-port S-parameters
array of complex numbers

2-port S-parameters, specified as a complex 2-by-2-by-*N* array.

`z0` — Reference impedance
50 (default) | positive scalar

Reference impedance in ohms, specified as a positive scalar. If the first input argument is an S-parameter object `hs`, the function uses `hs.Impedance` for the reference impedance.

`zl` — Load impedance
50 (default) | positive scalar

Load impedance in ohms, specified as a positive scalar.

`zs` — Source impedance
50 (default) | positive scalar

Source impedance in ohms, specified as a positive scalar.

Output Arguments

g — Power gain
vector

Unitless power gain values, returned as a vector. To obtain power gain in decibels, use `10*log10(g)`.

If the specified type of power gain is undefined for one or more of the specified S-parameter values in `s_params`, the `powergain` function returns `NaN`. As a result, `g` is either `NaN` or a vector that contains one or more `NaN` entries.

Examples

Power Gain of Two-Port Network

Calculate power gains for a sample 2-port network.

```
s11 = 0.61*exp(j*165/180*pi);
s21 = 3.72*exp(j*59/180*pi);
s12 = 0.05*exp(j*42/180*pi);
s22 = 0.45*exp(j*(-48/180)*pi);
sparam = [s11 s12; s21 s22];
z0 = 50;
zs = 10 + j*20;
zl = 30 - j*40;
```

Calculate the transducer power gain of the network

```
Gt = powergain(sparam,z0,zs,zl,'Gt')
```

```
Gt = 4.7066
```

Calculate the available power gain of the network

```
Ga = powergain(sparam,z0,zs,'Ga')
```

```
Ga = 11.4361
```

Calculate the operating power gain of the network

```
Gp = powergain(sparam,z0,zl,'Gp')
```

```
Gp = 10.5098
```

Calculate the maximum available power gain of the network

```
Gmag = powergain(sparam,'Gmag')
```

```
Gmag = 41.5032
```

Calculate the maximum stable power gain of the network

```
Gmsg = powergain(sparam,'Gmsg')
```

```
Gmsg = 74.4000
```

See Also

`s2tf`

Introduced in R2007b

rationalfit

Approximate data using stable rational function object

Syntax

```
fit = rationalfit(freq,data)
fit = rationalfit(freq,data,tol)
fit = rationalfit(___,Name,Value)
[fit,errdb] = rationalfit(...)

fit = rationalfit(s_obj,i,j...)
```

Description

`fit = rationalfit(freq,data)` fits a rational function object of the form

$$F(s) = \sum_{k=1}^{n} \frac{C_k}{s - A_k} + D, \quad s = j * 2\pi f$$

to the complex vector `data` over the frequency values in the positive vector `freq`. The function returns a handle to the rational function object, `h`, with properties `A`, `C`, `D`, and `Delay`.

`fit = rationalfit(freq,data,tol)` fits a rational function object to complex data and constrains the error of the fit according to the optional input argument `tol`.

`fit = rationalfit(___,Name,Value)` fits a rational function object of the form

$$F(s) = (\sum_{k=1}^{n} \frac{C_k}{s - A_k} + D)e^{-s.Delay}, \quad s = j * 2\pi f$$

with additional options specified by one or more `Name,Value` pair arguments. These arguments offer finer control over the performance and accuracy of the fitting algorithm.

`[fit,errdb] = rationalfit(...)` fits a rational function object to complex data and also returns `ERRDB`, which is the achieved error.

`fit = rationalfit(s_obj,i,j...)` fits S_{ij} using `FREQ = s_obj.Frequencies` and `DATA = rfparam(s_obj,i,j)` for s-parameter object, `s_obj`.

Examples

Rational Function Approximation of S-parameter Data

Fit a rational function object to S-parameter data, and compare the results by plotting the object against the data.

Read the S-parameter data into an RF data object.

```
orig_data = read(rfdata.data,'passive.s2p');
freq = orig_data.Freq;
data = orig_data.S_Parameters(1,1,:);
```

Fit a rational function to the data using `rationalfit`.

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

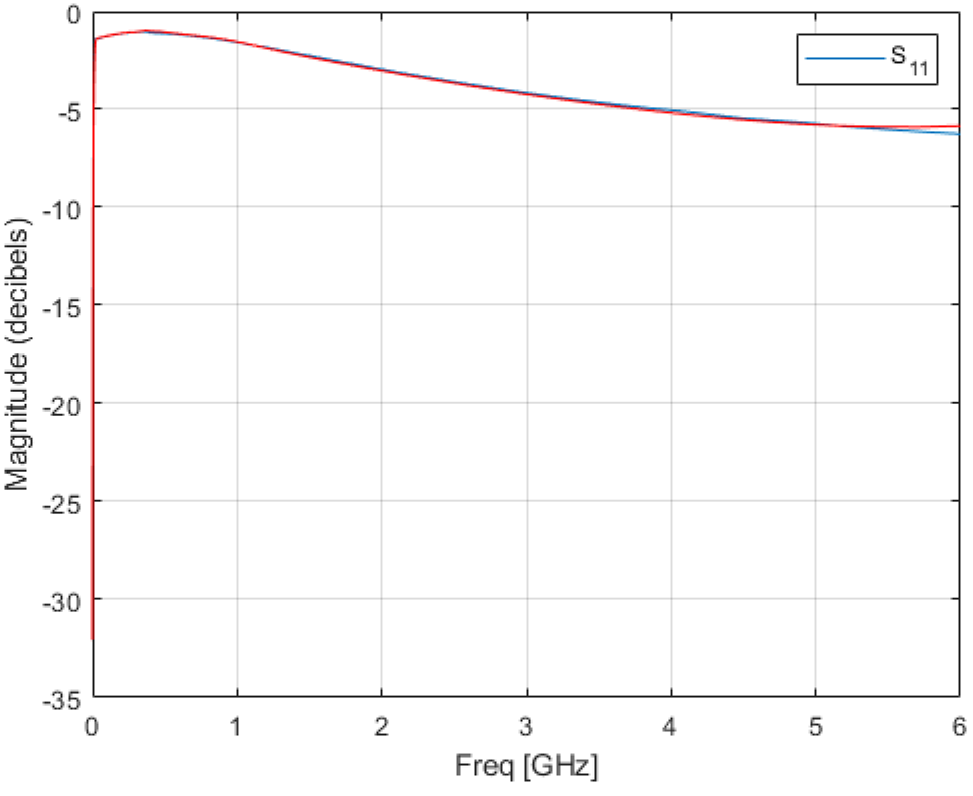
```
fit_data =
   rfmodel.rational with properties:

        A: [19x1 double]
        C: [19x1 double]
        D: 0
    Delay: 0
     Name: 'Rational Function'
```

Compute the frequency response of the rational function using `freqresp`.

```
[resp,freq] = freqresp(fit_data,freq);
```

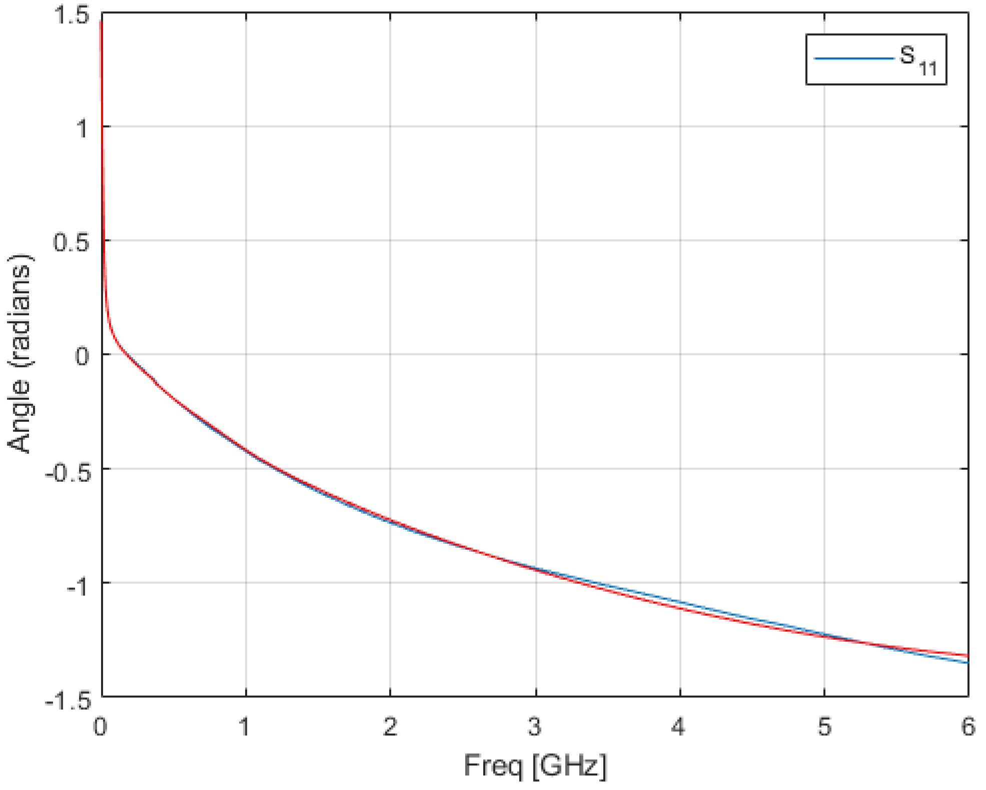
Plot the magnitude of the original data against the rational function approximation. S_{11} data appears in blue, and the rational function appears in red. Scaling the frequency values by `1e9` converts them to units of GHz.

```
figure
title('Rational fitting of S11 magnitude')
plot(orig_data,'S11','dB')
hold on
plot(freq/1e9,20*log10(abs(resp)),'r');
```



Plot the angle of the original data against the rational function approximation.

```
figure
title('Rational fitting of S11 angle')
plot(orig_data,'S11','Angle (radians)')
hold on
plot(freq/1e9,unwrap(angle(resp)),'r')
```



Rational Function Approximation of S-parameters

`rationalfit(freq,data)` also handles input 3D array of data (n x n x p), an input frequency array (p x 1), and returns a matrix (n x n) of rationalfit objects. Index into the matrix of rationalfit objects to access corresponding rationalfiit information.

Use `rationalfit` on multiple datasets defined in a matrix.

```
orig_data = sparameters('defaultbandpass.s2p');
data = orig_data.Parameters;
```

```
freq = orig_data.Frequencies;
fit_data = rationalfit(freq, data)
```

```
fit_data = 

  2x2 rfmodel.rational array with properties:

    A
    C
    D
    Delay
    Name
```

To access `rationalfit` data, use indexing on the rationalfit array. For example, to access the rational fit for the 1st element of the matrix, use:

```
S = fit_data(1, 1)
```

```
S = 

   rfmodel.rational with properties:

        A: [12x1 double]
        C: [12x1 double]
        D: 0
    Delay: 0
     Name: 'Rational Function'
```

Fit S-Parameter Object

Use rational fit to fit an S-parameter object from the file 'passive.s2p'.

```
S = sparameters('passive.s2p');
fit = rationalfit(S,1,1,'TendsToZero',false)
```

```
fit = 
   rfmodel.rational with properties:

        A: [5x1 double]
        C: [5x1 double]
```

```
    D: -0.4843
Delay: 0
 Name: 'Rational Function'
```

Input Arguments

freq — Frequencies

vector of positive numbers

Frequencies over which the function fits a rational object, specified as a vector of length M.

data — Data to fit

N-by-N-by-M array of complex numbers (default) | vector of complex numbers

Data to fit, specified as an N-by-N-by-M array of complex numbers. The function fits N^2 rational functions to the data along the M (frequency) dimension.

tol — Error tolerance

-40 (default) | scalar

Error tolerance ε, specified as a scalar in units of dB. The error-fitting equation is

$$10^{\varepsilon/20} \geq \frac{\sqrt{\sum_{k=0}^{n} \left| W_k F_0\{f_k\} - F(s) \right|^2}}{\sqrt{\sum_{k=0}^{n} \left| W_k F_0\{f_k\} \right|^2}}$$

where

- ε is the specified value of `tol`.
- F_0 is the value of the original data (`data`) at the specified frequency f_k (`freq`).
- F is the value of the rational function at $s = j2\pi f$.
- W is the weighting of the data.

rationalfit computes the relative error as a vector containing the dependent values of the fit data. If the object does not fit the original data within the specified tolerance, a warning message appears.

s_obj — S-parameter object
network parameter object

S-parameters, specified as an RF Toolbox network parameter object. To create this type of object, use the sparameters function.

i — Row index
positive integer

Row index of data to plot, specified as a positive integer.

j — Column index
positive integer

Column index of data to plot, specified as a positive integer.

Name-Value Pair Arguments

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

DelayFactor — Delay factor
0 (default) | scalar from 0 to 1

Scaling factor that controls the amount of delay to fit to the data, specified as the comma-separated pair consisting of 'DelayFactor' and a scalar between 0 and 1 inclusive. The Delay parameter, τ, of the rational function object is equal to the specified value of 'DelayFactor' times an estimate of the group delay of the data. If the original data has delay, increasing this value might allow rationalfit to fit the data with a lower-order object.

IterationLimit — Maximum number of rationalfit iterations
[4,12] (default) | vector of positive integers

Maximum number of rationalfit iterations, specified as a vector of positive integers. Provide a two-element vector to specify minimum and maximum [M1 M2]. Increasing the limit extends the time that the algorithm takes to produce a fit, but it might produce more accurate results.

`NPoles` — Number of poles

`[0 48]` (default) | nonnegative integer | vector of two nonnegative integers

Number of poles A_k of the rational function, specified as the comma-separated pair consisting of `'NPoles'` and an integer n or range of possible values of n.

To help `rationalfit` produce an accurate fit, choose a maximum value of `npoles` greater than or equal to twice the number of peaks on a plot of the data in the frequency domain.

After completing a rational fit, the function removes coefficient sets whose residues (C_k) are zero. Thus, when you specify a range for `npoles`, the number of poles of the fit may be less than `npoles(1)`.

`TendsToZero` — Asymptotic behavior of fit

`true` (default) | `false`

Asymptotic behavior of the rational function as frequency approaches infinity, specified as the comma-separated pair consisting of `'TendsToZero'` and a logical value. When this argument is `true`, the resulting rational function variable D is zero, and the function tends to zero. A value of `false` allows a nonzero value for `D`.

`Tolerance` — Error tolerance

-40 (default) | scalar

Error tolerance ε, specified as the comma-separated pair consisting of `'Tolerance'` and a scalar in units of dB. The error-fitting equation is

$$10^{\varepsilon/20} \geq \frac{\sqrt{\sum_{k=0}^{n} \left|W_k F_0\{f_k\} - F(s)\right|^2}}{\sqrt{\sum_{k=0}^{n} \left|W_k F_0\{f_k\}\right|^2}}$$

where

- ε is the specified tolerance.

- F_0 is the value of the original data (data) at the specified frequency f_k (freq).
- F is the value of the rational function at $s = j2\pi f$.
- W is the weighting of the data.

If the object does not fit the original data within the specified tolerance, the function throws a warning.

WaitBar — Graphical wait bar
false (default) | true

Logical value that toggles display of the graphical wait bar during fitting, specified as the comma-separated pair consisting of 'WaitBar' and either true or false. The true setting shows the graphical wait bar, and the false setting hides it. If you expect rationalfit to take a long time, and you want to monitor its progress, set 'WaitBar' to true.

Weight — Weighting of data
ones(size(*freq*)) (default) | vector of positive numbers

Weighting of the data at each frequency, specified as the comma-separated pair consisting of 'Weight' and a vector of positive numbers or an array same as that of the data. Each entry in weight corresponds to a frequency in freq, so the length of weight must be equal to the length of freq. Increasing the weight at a particular frequency improves the object fitting at that frequency. Specifying a weight of 0 at a particular frequency causes rationalfit to ignore the corresponding data point.

Output Arguments

fit — Rational function object
rfmodel.rational object

One or more rational function objects, returned as an N-by-N rfmodel.rational object. The number of dimensions in data determines the dimensionality of h.

errdb — Relative error
-40 (default) | double

Relative error achieved, returned as a double, in dB.

Tip

To see how well the object fits the original data, use the `freqresp` function to compute the frequency response of the object. Then, plot the original data and the frequency response of the rational function object. For more information, see the `freqresp` reference page or the examples in the next section.

References

Gustavsen.B 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.

Zeng.R and J. Sinsky, "Modified Rational Function Modeling Technique for High Speed Circuits," *IEEE MTT-S Int. Microwave Symp. Dig.*, San Francisco, CA, June 11–16, 2006.

See Also

`freqresp` | `rfmodel.rational` | `s2tf` | `timeresp` | `writeva`

Introduced in R2006b

rftool

Open RF Analysis Tool (RF Tool)

Syntax

```
rftool
```

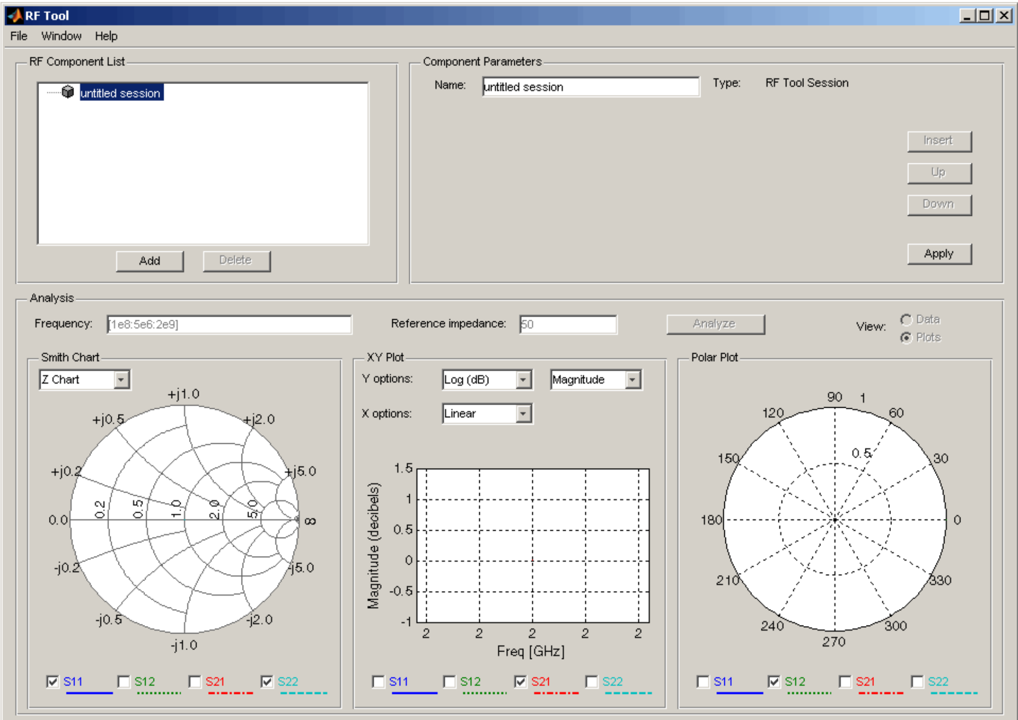
Description

`rftool` opens the RF Tool interface. Use this tool to:

- Create circuit components and set their parameters.
- Analyze components over a specified frequency range and step size.
- Plot the analysis results.
- Import component objects to and export them from the MATLAB workspace.
- Save RF Tool sessions for later use.

For more information, see “The RF Design and Analysis App” on page 5-2.

The following figure shows the RF Tool in its default state.



Introduced before R2006a

rlgc2s

Convert RLGC transmission line parameters to S-parameters

Syntax

```
s_params = rlgc2s(R,L,G,C,length,freq,z0)
s_params = rlgc2s(R,L,G,C,length,freq)
```

Description

`s_params = rlgc2s(R,L,G,C,length,freq,z0)` transforms RLGC transmission line parameter data into S-parameters.

`s_params = rlgc2s(R,L,G,C,length,freq)` transforms RLGC transmission line parameter data into S-parameters with a reference impedance of 50 Ω.

Input Arguments

R —

Specify an *N*-by-*N*-by-*M* array of distributed resistances, in units of Ω/m. The *N*-by-*N* matrices must be real symmetric, the diagonal terms must be nonnegative, and the off-diagonal terms must be nonnegative.

L —

Specify an *N*-by-*N*-by-*M* array of distributed inductances, in units of H/m. The *N*-by-*N* matrices must be real symmetric, the diagonal terms must be positive, and the off-diagonal terms must be nonnegative.

G —

Specify an N-by-N-by-M array of distributed conductances, in units of S/m. The N-by-N matrices must be real symmetric, the diagonal terms must be nonnegative, and the off-diagonal terms must be nonpositive.

`C` —

Specify an N-by-N-by-M array of distributed capacitances, in units of F/m. The matrices must be real symmetric, the diagonal terms must be positive, and the off-diagonal terms must be nonpositive.

`length` —

Specify the length of the transmission line in meters.

`freq` —

Specify the vector of M frequencies over which the transmission line parameters are defined.

`z0` — Reference Impedance

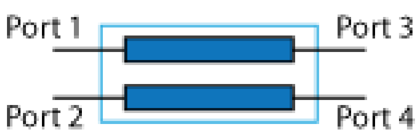
50 (default) | scalar

Reference impedance in ohms, specified as a scalar, of the resulting S-parameters.

Output Arguments

`s_params`

The output is a 2N-by-2N-by-M array of S-parameters. The following figure describes the port ordering convention of the output.



$$\left[\begin{array}{cc:cc} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} \\ \hdashline S_{31} & S_{32} & S_{33} & S_{34} \\ S_{41} & S_{42} & S_{43} & S_{44} \end{array}\right]$$

This port ordering convention assumes that:

- Each 2*N*-by-2*N* matrix consists of *N* input terminals and *N* output terminals.
- The first *N* ports (1 through *N*) of the S-parameter matrix are input ports.
- The last *N* ports (*N* + 1 through 2*N*) are output ports.

To reorder ports after using this function, use the `snp2smp` function.

Examples

Convert RLGC Transmission Line Parameters to S-Parameters

Define the variables for a transmission line.

```
length = 1e-3;
freq = 1e9;
z0 = 50;
R = 50;
L = 1e-9;
G = .01;
C = 1e-12;
```

Calculate the s-parameters.

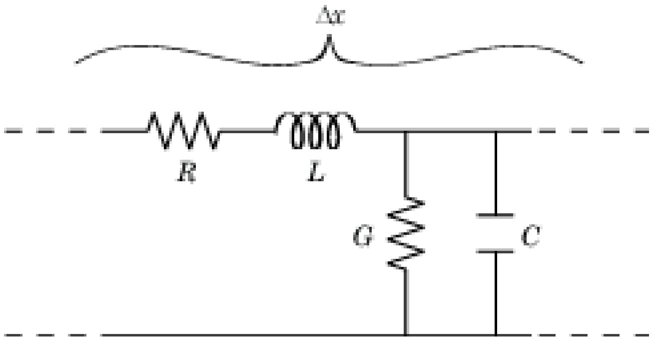
```
s_params = rlgc2s(R,L,G,C,length,freq,z0)
```

```
s_params = 2×2 complex

   0.0002 - 0.0001i   0.9993 - 0.0002i
   0.9993 - 0.0002i   0.0002 - 0.0001i
```

Definitions

RLCG Transmission Line Model

The following figure illustrates the RLGC transmission line model.



The representation consists of:

- The distributed resistance, *R*, of the conductors, represented by a series resistor.
- The distributed inductance, *L*, represented by a series inductor.
- The distributed conductance, *G*,
- The distributed capacitance, *C*, between the two conductors, represented by a shunt capacitor.

RLGC component units are all per unit length Δ*x*.

References

Bhatti, A. Aziz. “A computer Based Method for Computing the N-Dimensional Generalized ABCD Parameter Matrices of N-Dimensional Systems with Distributed Parameters.” *Southeastern Symposium on System Theory*. SSST, 22nd Conference, 11–13 March 1990, pp. 590–593.

See Also

`s2rlgc`

Introduced in R2011b

s2abcd

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*N*-by-2*N*-by-*M* array, representing *M* 2*N*-port S-parameters. `z0` is the reference impedance; its default is 50 ohms. `abcd_params` is a complex 2*N*-by-2*N*-by-*M* array, representing *M* 2-port ABCD-parameters. The output ABCD-parameters matrices have distinct *A*, *B*, *C*, and *D* submatrices:

$$\begin{bmatrix} [A] & [B] \\ [C] & [D] \end{bmatrix}$$

Examples

S-Parameters to ABCD-Parameters

Define a matrix of S-parameters.

```
s_11 = 0.61*exp(j*165/180*pi);
s_21 = 3.72*exp(j*59/180*pi);
s_12 = 0.05*exp(j*42/180*pi);
s_22 = 0.45*exp(j*(-48/180)*pi);
s_params = [s_11 s_12; s_21 s_22];
z0 = 50;
```

Convert to ABCD-parameters.

```
abcd_params = s2abcd(s_params,z0)
```

```
abcd_params = 2×2 complex

   0.0633 + 0.0069i   1.4958 - 3.9839i
   0.0022 - 0.0024i   0.0732 - 0.2664i
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see “RF Network Parameter Objects” on page 2-11.

See Also

abcd2s | h2abcd | s2h | s2y | s2z | y2abcd | z2abcd

Introduced before R2006a

s2h

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.

Examples

S-Parameters to H-Parameters

Define a matrix of S-parameters.

```
s_11 = 0.61*exp(j*165/180*pi);
s_21 = 3.72*exp(j*59/180*pi);
s_12 = 0.05*exp(j*42/180*pi);
s_22 = 0.45*exp(j*(-48/180)*pi);
s_params = [s_11 s_12; s_21 s_22];
z0 = 50;
```

Convert to H-parameters.

```
h_params = s2h(s_params,z0)
```

```
h_params = 2×2 complex

  15.3381 + 1.4019i   0.0260 + 0.0411i
```

```
  -0.9585 - 3.4902i   0.0106 + 0.0054i
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see “RF Network Parameter Objects” on page 2-11.

See Also

`h2s`

Introduced before R2006a

s2rlgc

Convert S-parameters to RLGC transmission line parameters

Syntax

```
rlgc_params = s2rlgc(s_params,length,freq,z0)
rlgc_params = s2rlgc(s_params,length,freq)
```

Description

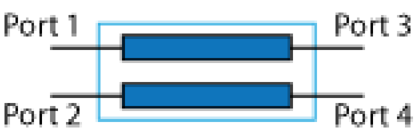
`rlgc_params = s2rlgc(s_params,length,freq,z0)` transforms multi-port S-parameter data into a frequency-domain representation of an RLGC transmission line.

`rlgc_params = s2rlgc(s_params,length,freq)` transforms multi-port S-parameter data into RLGC transmission line parameters using a reference impedance of 50 Ω.

Input Arguments

`s_params` —

Specify a 2*N*-by-2*N*-by-*M* array of S-parameters to transform into RLGC transmission line parameters. The following figure describes the port ordering convention assumed by the function.



$$\left[\begin{array}{cc:cc} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} \\ \hdashline S_{31} & S_{32} & S_{33} & S_{34} \\ S_{41} & S_{42} & S_{43} & S_{44} \end{array}\right]$$

The function assumes that:

- Each 2*N*-by-2*N* matrix consists of *N* input terminals and *N* output terminals.
- The first *N* ports (1 through *N*) of the S-parameter matrix are input ports.
- The last *N* ports (*N* + 1 through 2*N*) are output ports.

To reorder ports before using this function, use the `snp2smp` function.

`length` —

Specify the length of the transmission line in meters.

`freq` —

Specify the vector of *M* frequencies over which the S-parameter array `s_params` is defined.

`z0` — Reference Impedance
50 (default) | scalar

Reference impedance in ohms, specified as a scalar, of the resulting S-parameters.

Output Arguments

`rlgc_params`

The output `rlgc_params` is structure whose fields are *N*-by-*N*-by-*M* arrays of transmission line parameters. Each of the *M* *N*-by-*N* matrices correspond to a frequency in the input vector `freq`.

- `rlgc_params.R` is an array of distributed resistances, in units of Ω/m. The matrices are real symmetric, the diagonal terms are nonnegative, and the off-diagonal terms are nonnegative.
- `rlgc_params.L` is an array of distributed inductances, in units of H/m. The matrices are real symmetric, the diagonal terms are positive, and the off-diagonal terms are nonnegative.
- `rlgc_params.G` is an array of distributed conductances, in units of S/m. The matrices are real symmetric, the diagonal terms are nonnegative, and the off-diagonal terms are nonpositive.

- `rlgc_params.C` is an array of distributed capacitances, in units of F/m. The matrices are real symmetric, the diagonal terms are positive, and the off-diagonal terms are nonpositive.
- `rlgc_params.Zc` is an array of complex characteristic line impedances, in ohms.
- `rlgc_params.alpha` is an array of real attenuation coefficients, in units of Np/m.
- `rlgc_params.beta` is an array of real phase constants, in units of rad/m.

Examples

Convert S-Parameters to RLGC Parameters

Define the s-parameters.

```
s_11 = 0.000249791883190134 - 9.42320545953709e-005i;
s_12 = 0.999250283783862 -  0.000219770154524734i;
s_21 = 0.999250283783863 -  0.000219770154524756i;
s_22 = 0.000249791883190079 - 9.42320545953931e-005i;
s_params = [s_11,s_12; s_21,s_22];
```

Specify the length, frequency of operation, and impedance of the transmission line.

```
length = 1e-3;
freq = 1e9;
z0 = 50;
```

Convert from s-parameters to rlgc-parameters.

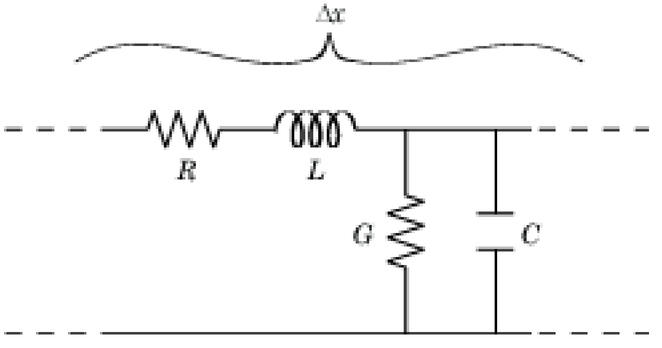
```
rlgc_params = s2rlgc(s_params,length,freq,z0)
```

```
rlgc_params = struct with fields:
        R: 50.0000
        L: 1.0000e-09
        G: 0.0100
        C: 1.0000e-12
    alpha: 0.7265
     beta: 0.2594
       Zc: 63.7761 -14.1268i
```

Definitions

RLCG Transmission Line Model

The following figure illustrates the RLGC transmission line model.



The representation consists of:

- The distributed resistance, *R*, of the conductors, represented by a series resistor.
- The distributed inductance, *L*, of the conductors, represented by a series inductor.
- The distributed conductance, *G*, between the two conductors, represented by a shunt resistor.
- The distributed capacitance, *C*, between the two conductors, represented by a shunt capacitor.

RLGC component units are all per unit length Δx.

References

[1] Degerstrom, M.J., Gilbert, B.K., and Daniel, E.S . "Accurate resistance, inductance, capacitance, and conductance (RLCG) from uniform transmission line measurements."*Electrical Performance of Electronic Packaging*. IEEE-EPEP, 18th Conference, 27–29 October 2008, pp. 77–80.

[2] Sampath, M.K. "On addressing the practical issues in the extraction of RLGC parameters for lossy multi-conductor transmission lines using S-parameter

models." *Electrical Performance of Electronic Packaging,*. IEEE-EPEP, 18th Conference, 27–29 October 2008, pp. 259–262.

[3] Eisenstadt, W. R., and Eo, Y. "S-parameter-based IC interconnect transmission line characterization," *IEEE Transactions on Components, Hybrids, and Manufacturing Technology*. Vol. 15, No. 4, August 1992, pp. 483–490.

See Also

`rlgc2s`

Introduced in R2011b

s2s

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`.

Examples

S-Parameters to S-Parameters with Different Impedance

Define a matrix of S-parameters.

```
s_11 = 0.61*exp(1i*165/180*pi);
s_21 = 3.72*exp(1i*59/180*pi);
s_12 = 0.05*exp(1i*42/180*pi);
s_22 = 0.45*exp(1i*(-48/180)*pi);
s_params = [s_11 s_12; s_21 s_22];
z0 = 50;
z0_new = 40;
```

Convert to S-parameters with different impedance.

```
s_params_new = s2s(s_params,z0,z0_new)
```

```
s_params_new = 2×2 complex

  -0.5039 + 0.1563i   0.0373 + 0.0349i
   1.8929 + 3.2940i   0.4150 - 0.3286i
```

Alternatives

The `newref` function changes the reference impedance of S-parameters objects.

See Also

`abcd2s` | `h2s` | `s2abcd` | `s2h` | `s2y` | `s2z` | `y2s` | `z2s`

Introduced before R2006a

s2scc

Convert single-ended S-parameters to common-mode S-parameters (S_{cc})

Syntax

```
scc_params = s2scc(s_params)
scc_params = s2scc(s_params,option)
```

Description

`scc_params = s2scc(s_params)` converts the 2*N*-port, single-ended S-parameters, `s_params`, to *N*-port, common-mode S-parameters, `scc_params`. `scc_params` is a complex *N*-by-*N*-by-*M* array that represents *M* *N*-port, common-mode S-parameters (S_{cc}).

`scc_params = s2scc(s_params,option)` converts S-parameters based on the optional `option` argument, which indicates the port-ordering convention of the S-parameters.

Input Arguments

`s_params` — S-parameters
array

S-parameters, specified as a complex 4-by-4-by-*M* array, that represents *M* 4-port S-parameters.

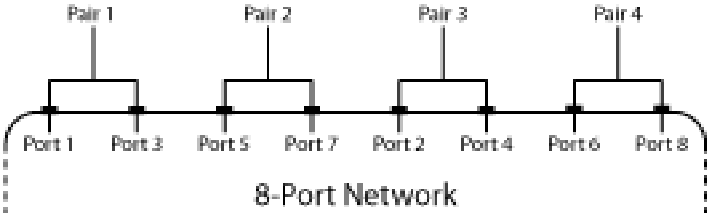
`option` — Port order
1 (default) | 2 | 3

Port order, specified as `1`, `2`, `3`, determines how the function orders the ports:

- `1` — `s2scc` pairs the odd-numbered ports together first, followed by the even-numbered ports. For example, in a single-ended, 8-port network:

- Ports 1 and 3 become common-mode pair 1.
- Ports 5 and 7 become common-mode pair 2.
- Ports 2 and 4 become common-mode pair 3.
- Ports 6 and 8 become common-mode pair 4.

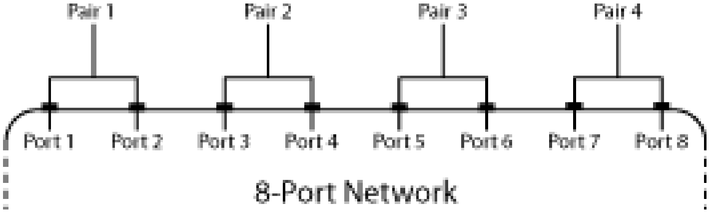
The following figure illustrates this convention for an 8-port device.



- `2` — `s2scc` pairs the input and output ports in ascending order. For example, in a single-ended, 8-port network:

  - Ports 1 and 2 become common-mode pair 1.
  - Ports 3 and 4 become common-mode pair 2.
  - Ports 5 and 6 become common-mode pair 3.
  - Ports 7 and 8 become common-mode pair 4.

  The following figure illustrates this convention for an 8-port device.

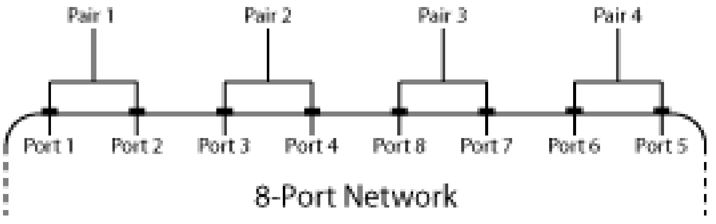


- `3` — `s2scc` pairs the input ports in ascending order and the output ports in descending order. For example, in a single-ended, 8-port network:

  - Ports 1 and 2 become common-mode pair 1.
  - Ports 3 and 4 become common-mode pair 2.
  - Ports 8 and 7 become common-mode pair 3.

- Ports 6 and 5 become common-mode pair 4.

The following figure illustrates this convention for an 8-port device.



Examples

Network Data to Common Mode S-Parameters

Convert network data to common-mode S-parameters using the default

```
%port ordering.
ckt = read(rfckt.passive,'default.s4p');
s4p = ckt.NetworkData.Data;
s_cc = s2scc(s4p)

s_cc =
s_cc(:,:,1) =

   0.1799 - 0.1839i  -0.5300 - 0.6771i
  -0.5314 - 0.6800i   0.1756 - 0.1910i


s_cc(:,:,2) =

   0.1045 - 0.2343i  -0.7609 - 0.3891i
  -0.7609 - 0.3898i   0.0961 - 0.2386i


s_cc(:,:,3) =

   0.0130 - 0.2415i  -0.8488 - 0.0337i
  -0.8486 - 0.0338i   0.0067 - 0.2421i
```

```
s_cc(:,:,4) =

  -0.0650 - 0.2066i  -0.7805 + 0.3262i
  -0.7804 + 0.3253i  -0.0693 - 0.2066i


s_cc(:,:,5) =

  -0.1130 - 0.1443i  -0.5679 + 0.6230i
  -0.5672 + 0.6235i  -0.1156 - 0.1464i


s_cc(:,:,6) =

  -0.1232 - 0.0760i  -0.2499 + 0.8020i
  -0.2518 + 0.8028i  -0.1258 - 0.0802i


s_cc(:,:,7) =

  -0.1018 - 0.0221i   0.1132 + 0.8285i
   0.1133 + 0.8298i  -0.1054 - 0.0290i


s_cc(:,:,8) =

  -0.0624 + 0.0026i   0.4531 + 0.6990i
   0.4531 + 0.6979i  -0.0687 - 0.0037i


s_cc(:,:,9) =

  -0.0268 - 0.0039i   0.7033 + 0.4372i
   0.7037 + 0.4363i  -0.0341 - 0.0089i


s_cc(:,:,10) =

  -0.0155 - 0.0331i   0.8141 + 0.0943i
   0.8139 + 0.0936i  -0.0215 - 0.0358i


s_cc(:,:,11) =
```

```
  -0.0318 - 0.0595i   0.7721 - 0.2566i
   0.7720 - 0.2571i  -0.0337 - 0.0626i


s_cc(:,:,12) =

  -0.0752 - 0.0707i   0.5821 - 0.5551i
   0.5834 - 0.5552i  -0.0737 - 0.0782i


s_cc(:,:,13) =

  -0.1271 - 0.0459i   0.2895 - 0.7399i
   0.2899 - 0.7407i  -0.1267 - 0.0609i


s_cc(:,:,14) =

  -0.1632 + 0.0134i  -0.0485 - 0.7809i
  -0.0485 - 0.7809i  -0.1681 - 0.0068i


s_cc(:,:,15) =

  -0.1620 + 0.0916i  -0.3650 - 0.6795i
  -0.3653 - 0.6794i  -0.1757 + 0.0713i


s_cc(:,:,16) =

  -0.1177 + 0.1635i  -0.6058 - 0.4611i
  -0.6061 - 0.4605i  -0.1383 + 0.1481i


s_cc(:,:,17) =

  -0.0412 + 0.2039i  -0.7345 - 0.1675i
  -0.7348 - 0.1679i  -0.0638 + 0.1964i


s_cc(:,:,18) =

   0.0431 + 0.2010i  -0.7329 + 0.1477i
```

```
  -0.7333 + 0.1482i   0.0238 + 0.1991i


s_cc(:,:,19) =

   0.1115 + 0.1560i  -0.6067 + 0.4354i
  -0.6069 + 0.4353i   0.0982 + 0.1586i


s_cc(:,:,20) =

   0.1406 + 0.0887i  -0.3711 + 0.6491i
  -0.3711 + 0.6486i   0.1341 + 0.0876i


s_cc(:,:,21) =

   0.1273 + 0.0268i  -0.0655 + 0.7460i
  -0.0656 + 0.7463i   0.1220 + 0.0191i


s_cc(:,:,22) =

   0.0853 - 0.0052i   0.2551 + 0.7042i
   0.2551 + 0.7046i   0.0764 - 0.0180i


s_cc(:,:,23) =

   0.0429 + 0.0017i   0.5294 + 0.5278i
   0.5290 + 0.5275i   0.0278 - 0.0107i


s_cc(:,:,24) =

   0.0252 + 0.0369i   0.6978 + 0.2508i
   0.6978 + 0.2509i   0.0075 + 0.0287i


s_cc(:,:,25) =

   0.0435 + 0.0754i   0.7300 - 0.0674i
   0.7308 - 0.0670i   0.0288 + 0.0709i
```

```
s_cc(:,:,26) =

   0.0904 + 0.0918i   0.6269 - 0.3635i
   0.6264 - 0.3630i   0.0813 + 0.0864i


s_cc(:,:,27) =

   0.1450 + 0.0739i   0.4133 - 0.5841i
   0.4131 - 0.5842i   0.1402 + 0.0605i


s_cc(:,:,28) =

   0.1839 + 0.0242i   0.1331 - 0.6951i
   0.1326 - 0.6951i   0.1761 - 0.0012i


s_cc(:,:,29) =

   0.1924 - 0.0413i  -0.1648 - 0.6822i
  -0.1646 - 0.6819i   0.1720 - 0.0769i


s_cc(:,:,30) =

   0.1682 - 0.1036i  -0.4299 - 0.5502i
  -0.4296 - 0.5499i   0.1279 - 0.1403i


s_cc(:,:,31) =

   0.1183 - 0.1469i  -0.6153 - 0.3216i
  -0.6152 - 0.3217i   0.0591 - 0.1706i


s_cc(:,:,32) =

   0.0566 - 0.1595i  -0.6911 - 0.0391i
  -0.6914 - 0.0392i  -0.0082 - 0.1610i


s_cc(:,:,33) =
```

```
   0.0037 - 0.1437i  -0.6452 + 0.2528i
  -0.6449 + 0.2525i  -0.0554 - 0.1260i


s_cc(:,:,34) =

  -0.0299 - 0.1102i  -0.4792 + 0.4989i
  -0.4789 + 0.4994i  -0.0742 - 0.0797i


s_cc(:,:,35) =

  -0.0391 - 0.0729i  -0.2242 + 0.6528i
  -0.2244 + 0.6523i  -0.0660 - 0.0416i


s_cc(:,:,36) =

  -0.0284 - 0.0462i   0.0708 + 0.6844i
   0.0714 + 0.6833i  -0.0437 - 0.0243i


s_cc(:,:,37) =

  -0.0093 - 0.0397i   0.3519 + 0.5864i
   0.3514 + 0.5858i  -0.0241 - 0.0301i


s_cc(:,:,38) =

   0.0020 - 0.0542i   0.5609 + 0.3785i
   0.5601 + 0.3783i  -0.0218 - 0.0513i


s_cc(:,:,39) =

  -0.0116 - 0.0757i   0.6556 + 0.1078i
   0.6552 + 0.1079i  -0.0459 - 0.0688i


s_cc(:,:,40) =

  -0.0341 - 0.0832i   0.6392 - 0.1674i
```

```
   0.6393 - 0.1670i  -0.0731 - 0.0655i


s_cc(:,:,41) =

  -0.0709 - 0.0866i   0.5092 - 0.4189i
   0.5089 - 0.4186i  -0.1064 - 0.0559i


s_cc(:,:,42) =

  -0.1154 - 0.0636i   0.2853 - 0.5879i
   0.2849 - 0.5877i  -0.1393 - 0.0243i


s_cc(:,:,43) =

  -0.1469 - 0.0145i   0.0139 - 0.6481i
   0.0142 - 0.6473i  -0.1570 + 0.0249i


s_cc(:,:,44) =

  -0.1504 + 0.0504i  -0.2534 - 0.5902i
  -0.2534 - 0.5893i  -0.1501 + 0.0820i


s_cc(:,:,45) =

  -0.1198 + 0.1106i  -0.4715 - 0.4289i
  -0.4705 - 0.4289i  -0.1165 + 0.1314i


s_cc(:,:,46) =

  -0.0637 + 0.1471i  -0.6021 - 0.1949i
  -0.6014 - 0.1947i  -0.0642 + 0.1584i


s_cc(:,:,47) =

  -0.0012 + 0.1505i  -0.6247 + 0.0697i
  -0.6244 + 0.0695i  -0.0106 + 0.1575i
```

```
s_cc(:,:,48) =

   0.0512 + 0.1220i  -0.5413 + 0.3188i
  -0.5405 + 0.3181i   0.0332 + 0.1369i


s_cc(:,:,49) =

   0.0709 + 0.0744i  -0.3604 + 0.5162i
  -0.3606 + 0.5157i   0.0568 + 0.0982i


s_cc(:,:,50) =

   0.0575 + 0.0358i  -0.1108 + 0.6199i
  -0.1110 + 0.6198i   0.0516 + 0.0622i


s_cc(:,:,51) =

   0.0271 + 0.0238i   0.1597 + 0.6075i
   0.1593 + 0.6072i   0.0287 + 0.0472i


s_cc(:,:,52) =

   0.0027 + 0.0416i   0.3987 + 0.4810i
   0.3983 + 0.4809i   0.0082 + 0.0583i


s_cc(:,:,53) =

   0.0028 + 0.0757i   0.5589 + 0.2664i
   0.5587 + 0.2666i   0.0083 + 0.0859i


s_cc(:,:,54) =

   0.0319 + 0.1049i   0.6126 + 0.0085i
   0.6123 + 0.0087i   0.0347 + 0.1104i


s_cc(:,:,55) =
```

```
   0.0798 + 0.1106i   0.5549 - 0.2437i
   0.5541 - 0.2432i   0.0791 + 0.1139i


s_cc(:,:,56) =

   0.1261 + 0.0857i   0.4021 - 0.4467i
   0.4014 - 0.4459i   0.1224 + 0.0892i


s_cc(:,:,57) =

   0.1537 + 0.0396i   0.1815 - 0.5690i
   0.1816 - 0.5682i   0.1491 + 0.0433i


s_cc(:,:,58) =

   0.1548 - 0.0140i  -0.0684 - 0.5912i
  -0.0685 - 0.5904i   0.1489 - 0.0121i


s_cc(:,:,59) =

   0.1310 - 0.0590i  -0.3067 - 0.5089i
  -0.3063 - 0.5083i   0.1206 - 0.0584i


s_cc(:,:,60) =

   0.0929 - 0.0840i  -0.4890 - 0.3345i
  -0.4884 - 0.3343i   0.0754 - 0.0812i


s_cc(:,:,61) =

   0.0543 - 0.0874i  -0.5816 - 0.1012i
  -0.5808 - 0.1009i   0.0311 - 0.0758i


s_cc(:,:,62) =

   0.0282 - 0.0744i  -0.5708 + 0.1496i
```

```
  -0.5699 + 0.1490i   0.0049 - 0.0536i


s_cc(:,:,63) =

   0.0186 - 0.0604i  -0.4544 + 0.3744i
  -0.4548 + 0.3740i  -0.0035 - 0.0282i


s_cc(:,:,64) =

   0.0171 - 0.0541i  -0.2545 + 0.5284i
  -0.2544 + 0.5277i   0.0044 - 0.0092i


s_cc(:,:,65) =

   0.0159 - 0.0569i  -0.0088 + 0.5827i
  -0.0090 + 0.5827i   0.0203 - 0.0052i


s_cc(:,:,66) =

   0.0084 - 0.0655i   0.2346 + 0.5282i
   0.2347 + 0.5286i   0.0320 - 0.0172i


s_cc(:,:,67) =

  -0.0090 - 0.0743i   0.4293 + 0.3769i
   0.4294 + 0.3764i   0.0293 - 0.0385i


s_cc(:,:,68) =

  -0.0384 - 0.0726i   0.5366 + 0.1635i
   0.5364 + 0.1631i   0.0070 - 0.0531i


s_cc(:,:,69) =

  -0.0590 - 0.0513i   0.5548 - 0.0649i
   0.5538 - 0.0649i  -0.0128 - 0.0482i
```

```
s_cc(:,:,70) =

  -0.0685 - 0.0341i   0.4796 - 0.2898i
   0.4797 - 0.2895i  -0.0281 - 0.0456i


s_cc(:,:,71) =

  -0.0774 - 0.0147i   0.3148 - 0.4631i
   0.3150 - 0.4622i  -0.0467 - 0.0363i


s_cc(:,:,72) =

  -0.0807 + 0.0071i   0.0927 - 0.5503i
   0.0930 - 0.5499i  -0.0616 - 0.0171i


s_cc(:,:,73) =

  -0.0773 + 0.0283i  -0.1448 - 0.5370i
  -0.1449 - 0.5364i  -0.0650 + 0.0077i


s_cc(:,:,74) =

  -0.0683 + 0.0460i  -0.3546 - 0.4245i
  -0.3539 - 0.4242i  -0.0555 + 0.0296i


s_cc(:,:,75) =

  -0.0567 + 0.0580i  -0.4954 - 0.2356i
  -0.4950 - 0.2357i  -0.0378 + 0.0402i


s_cc(:,:,76) =

  -0.0453 + 0.0663i  -0.5427 - 0.0094i
  -0.5424 - 0.0097i  -0.0239 + 0.0382i


s_cc(:,:,77) =
```

```
  -0.0335 + 0.0670i  -0.4972 + 0.2134i
  -0.4970 + 0.2131i  -0.0158 + 0.0313i


s_cc(:,:,78) =

  -0.0344 + 0.0642i  -0.3631 + 0.4018i
  -0.3628 + 0.4011i  -0.0192 + 0.0175i


s_cc(:,:,79) =

  -0.0436 + 0.0729i  -0.1599 + 0.5140i
  -0.1598 + 0.5140i  -0.0405 + 0.0117i


s_cc(:,:,80) =

  -0.0485 + 0.0971i   0.0700 + 0.5298i
   0.0696 + 0.5292i  -0.0687 + 0.0281i


s_cc(:,:,81) =

  -0.0372 + 0.1312i   0.2816 + 0.4474i
   0.2812 + 0.4473i  -0.0851 + 0.0677i


s_cc(:,:,82) =

  -0.0037 + 0.1623i   0.4365 + 0.2869i
   0.4367 + 0.2871i  -0.0750 + 0.1177i


s_cc(:,:,83) =

   0.0479 + 0.1755i   0.5104 + 0.0806i
   0.5101 + 0.0806i  -0.0357 + 0.1593i


s_cc(:,:,84) =

   0.1039 + 0.1614i   0.4939 - 0.1348i
```

```
   0.4937 - 0.1343i   0.0224 + 0.1756i


s_cc(:,:,85) =

   0.1481 + 0.1209i   0.3933 - 0.3222i
   0.3927 - 0.3220i   0.0806 + 0.1594i


s_cc(:,:,86) =

   0.1652 + 0.0648i   0.2271 - 0.4531i
   0.2271 - 0.4528i   0.1186 + 0.1185i


s_cc(:,:,87) =

   0.1518 + 0.0134i   0.0210 - 0.5078i
   0.0209 - 0.5078i   0.1307 + 0.0716i


s_cc(:,:,88) =

   0.1191 - 0.0170i  -0.1924 - 0.4715i
  -0.1923 - 0.4715i   0.1187 + 0.0318i


s_cc(:,:,89) =

   0.0840 - 0.0216i  -0.3721 - 0.3475i
  -0.3722 - 0.3474i   0.0936 + 0.0129i


s_cc(:,:,90) =

   0.0618 - 0.0068i  -0.4821 - 0.1587i
  -0.4820 - 0.1579i   0.0731 + 0.0171i


s_cc(:,:,91) =

   0.0603 + 0.0157i  -0.5001 + 0.0574i
  -0.5003 + 0.0577i   0.0745 + 0.0325i
```

```
s_cc(:,:,92) =

   0.0816 + 0.0304i  -0.4281 + 0.2594i
  -0.4279 + 0.2590i   0.0955 + 0.0363i


s_cc(:,:,93) =

   0.1133 + 0.0208i  -0.2779 + 0.4118i
  -0.2781 + 0.4113i   0.1213 + 0.0199i


s_cc(:,:,94) =

   0.1350 - 0.0134i  -0.0804 + 0.4853i
  -0.0804 + 0.4848i   0.1374 - 0.0158i


s_cc(:,:,95) =

   0.1337 - 0.0604i   0.1268 + 0.4690i
   0.1268 + 0.4690i   0.1323 - 0.0613i


s_cc(:,:,96) =

   0.1046 - 0.1036i   0.3053 + 0.3697i
   0.3058 + 0.3699i   0.1017 - 0.1024i


s_cc(:,:,97) =

   0.0538 - 0.1242i   0.4229 + 0.2106i
   0.4228 + 0.2108i   0.0510 - 0.1203i


s_cc(:,:,98) =

   0.0063 - 0.1109i   0.4711 + 0.0249i
   0.4708 + 0.0250i   0.0065 - 0.1052i


s_cc(:,:,99) =
```

```
  -0.0183 - 0.0861i   0.4445 - 0.1722i
   0.4444 - 0.1723i  -0.0156 - 0.0819i


s_cc(:,:,100) =

  -0.0302 - 0.0620i   0.3325 - 0.3443i
   0.3325 - 0.3443i  -0.0257 - 0.0591i


s_cc(:,:,101) =

  -0.0298 - 0.0399i   0.1572 - 0.4506i
   0.1572 - 0.4503i  -0.0228 - 0.0414i


s_cc(:,:,102) =

  -0.0185 - 0.0292i  -0.0456 - 0.4736i
  -0.0456 - 0.4736i  -0.0151 - 0.0375i


s_cc(:,:,103) =

  -0.0094 - 0.0349i  -0.2395 - 0.4089i
  -0.2395 - 0.4089i  -0.0131 - 0.0453i


s_cc(:,:,104) =

  -0.0138 - 0.0492i  -0.3855 - 0.2676i
  -0.3853 - 0.2679i  -0.0239 - 0.0595i


s_cc(:,:,105) =

  -0.0323 - 0.0588i  -0.4559 - 0.0828i
  -0.4559 - 0.0830i  -0.0513 - 0.0678i


s_cc(:,:,106) =

  -0.0561 - 0.0608i  -0.4468 + 0.1106i
```

```
  -0.4465 + 0.1107i  -0.0854 - 0.0590i


s_cc(:,:,107) =

  -0.0891 - 0.0549i  -0.3595 + 0.2852i
  -0.3598 + 0.2853i  -0.1165 - 0.0367i


s_cc(:,:,108) =

  -0.1262 - 0.0285i  -0.2064 + 0.4072i
  -0.2064 + 0.4072i  -0.1424 - 0.0003i


s_cc(:,:,109) =

  -0.1515 + 0.0206i  -0.0179 + 0.4515i
  -0.0176 + 0.4520i  -0.1543 + 0.0502i


s_cc(:,:,110) =

  -0.1516 + 0.0824i   0.1687 + 0.4141i
   0.1690 + 0.4140i  -0.1432 + 0.1071i


s_cc(:,:,111) =

  -0.1222 + 0.1407i   0.3211 + 0.3050i
   0.3211 + 0.3052i  -0.1069 + 0.1570i


s_cc(:,:,112) =

  -0.0695 + 0.1799i   0.4146 + 0.1456i
   0.4142 + 0.1460i  -0.0509 + 0.1867i


s_cc(:,:,113) =

  -0.0071 + 0.1913i   0.4366 - 0.0367i
   0.4366 - 0.0362i   0.0112 + 0.1880i
```

```
s_cc(:,:,114) =

   0.0496 + 0.1745i   0.3827 - 0.2124i
   0.3826 - 0.2121i   0.0629 + 0.1617i


s_cc(:,:,115) =

   0.0863 + 0.1375i   0.2605 - 0.3513i
   0.2607 - 0.3515i   0.0904 + 0.1184i


s_cc(:,:,116) =

   0.0958 + 0.0959i   0.0915 - 0.4269i
   0.0916 - 0.4273i   0.0877 + 0.0770i


s_cc(:,:,117) =

   0.0818 + 0.0691i  -0.0942 - 0.4270i
  -0.0943 - 0.4270i   0.0668 + 0.0578i


s_cc(:,:,118) =

   0.0651 + 0.0675i  -0.2653 - 0.3472i
  -0.2651 - 0.3466i   0.0481 + 0.0622i


s_cc(:,:,119) =

   0.0630 + 0.0817i  -0.3838 - 0.2007i
  -0.3837 - 0.2005i   0.0451 + 0.0821i


s_cc(:,:,120) =

   0.0799 + 0.0970i  -0.4272 - 0.0214i
  -0.4279 - 0.0217i   0.0657 + 0.1023i


s_cc(:,:,121) =
```

```
   0.1139 + 0.1013i  -0.3927 + 0.1557i
  -0.3934 + 0.1559i   0.1043 + 0.1061i


s_cc(:,:,122) =

   0.1559 + 0.0829i  -0.2902 + 0.3008i
  -0.2908 + 0.3008i   0.1475 + 0.0827i


s_cc(:,:,123) =

   0.1878 + 0.0386i  -0.1395 + 0.3892i
  -0.1394 + 0.3901i   0.1752 + 0.0335i


s_cc(:,:,124) =

   0.1945 - 0.0203i   0.0325 + 0.4084i
   0.0328 + 0.4094i   0.1742 - 0.0278i


s_cc(:,:,125) =

   0.1725 - 0.0763i   0.1950 + 0.3569i
   0.1954 + 0.3570i   0.1412 - 0.0823i


s_cc(:,:,126) =

   0.1283 - 0.1131i   0.3209 + 0.2452i
   0.3209 + 0.2450i   0.0850 - 0.1114i


s_cc(:,:,127) =

   0.0803 - 0.1219i   0.3922 + 0.0946i
   0.3925 + 0.0945i   0.0277 - 0.1046i


s_cc(:,:,128) =

   0.0472 - 0.1134i   0.4002 - 0.0752i
```

```
   0.4007 - 0.0751i  -0.0074 - 0.0741i


s_cc(:,:,129) =

   0.0268 - 0.1042i   0.3335 - 0.2385i
   0.3338 - 0.2390i  -0.0179 - 0.0396i


s_cc(:,:,130) =

   0.0125 - 0.0973i   0.1998 - 0.3568i
   0.2001 - 0.3575i  -0.0067 - 0.0117i


s_cc(:,:,131) =

   0.0014 - 0.0937i   0.0303 - 0.4033i
   0.0302 - 0.4036i   0.0191 - 0.0038i


s_cc(:,:,132) =

  -0.0086 - 0.0961i  -0.1386 - 0.3751i
  -0.1391 - 0.3753i   0.0412 - 0.0232i


s_cc(:,:,133) =

  -0.0278 - 0.1048i  -0.2790 - 0.2809i
  -0.2793 - 0.2808i   0.0403 - 0.0579i


s_cc(:,:,134) =

  -0.0598 - 0.1065i  -0.3663 - 0.1400i
  -0.3661 - 0.1397i   0.0133 - 0.0890i


s_cc(:,:,135) =

  -0.0958 - 0.0964i  -0.3900 + 0.0229i
  -0.3899 + 0.0230i  -0.0301 - 0.1024i
```

```
s_cc(:,:,136) =

  -0.1342 - 0.0738i  -0.3441 + 0.1819i
  -0.3438 + 0.1822i  -0.0801 - 0.0960i


s_cc(:,:,137) =

  -0.1688 - 0.0328i  -0.2352 + 0.3068i
  -0.2353 + 0.3068i  -0.1290 - 0.0664i


s_cc(:,:,138) =

  -0.1882 + 0.0244i  -0.0859 + 0.3734i
  -0.0856 + 0.3734i  -0.1644 - 0.0142i


s_cc(:,:,139) =

  -0.1836 + 0.0893i   0.0750 + 0.3718i
   0.0753 + 0.3720i  -0.1750 + 0.0518i


s_cc(:,:,140) =

  -0.1529 + 0.1490i   0.2195 + 0.3062i
   0.2196 + 0.3059i  -0.1559 + 0.1171i


s_cc(:,:,141) =

  -0.1017 + 0.1903i   0.3242 + 0.1880i
   0.3244 + 0.1877i  -0.1117 + 0.1659i


s_cc(:,:,142) =

  -0.0416 + 0.2051i   0.3727 + 0.0377i
   0.3724 + 0.0379i  -0.0552 + 0.1881i


s_cc(:,:,143) =
```

```
   0.0115 + 0.1937i   0.3546 - 0.1202i
   0.3544 - 0.1200i  -0.0033 + 0.1828i


s_cc(:,:,144) =

   0.0456 + 0.1652i   0.2717 - 0.2578i
   0.2717 - 0.2577i   0.0307 + 0.1589i


s_cc(:,:,145) =

   0.0548 + 0.1347i   0.1364 - 0.3477i
   0.1361 - 0.3474i   0.0393 + 0.1321i


s_cc(:,:,146) =

   0.0454 + 0.1181i  -0.0237 - 0.3684i
  -0.0236 - 0.3683i   0.0285 + 0.1229i


s_cc(:,:,147) =

   0.0306 + 0.1252i  -0.1737 - 0.3192i
  -0.1736 - 0.3192i   0.0216 + 0.1412i


s_cc(:,:,148) =

   0.0323 + 0.1561i  -0.2870 - 0.2137i
  -0.2869 - 0.2138i   0.0403 + 0.1724i


s_cc(:,:,149) =

   0.0664 + 0.1902i  -0.3429 - 0.0725i
  -0.3431 - 0.0726i   0.0861 + 0.1950i


s_cc(:,:,150) =

   0.1265 + 0.2025i  -0.3337 + 0.0735i
```

```
  -0.3337 + 0.0734i   0.1497 + 0.1931i


s_cc(:,:,151) =

   0.1937 + 0.1826i  -0.2689 + 0.1966i
  -0.2688 + 0.1967i   0.2155 + 0.1586i


s_cc(:,:,152) =

   0.2517 + 0.1312i  -0.1662 + 0.2820i
  -0.1661 + 0.2822i   0.2642 + 0.0927i


s_cc(:,:,153) =

   0.2839 + 0.0546i  -0.0412 + 0.3216i
  -0.0411 + 0.3219i   0.2806 + 0.0079i


s_cc(:,:,154) =

   0.2787 - 0.0297i   0.0891 + 0.3109i
   0.0892 + 0.3110i   0.2586 - 0.0765i


s_cc(:,:,155) =

   0.2378 - 0.1004i   0.2056 + 0.2503i
   0.2056 + 0.2508i   0.2030 - 0.1407i


s_cc(:,:,156) =

   0.1764 - 0.1404i   0.2910 + 0.1500i
   0.2913 + 0.1498i   0.1304 - 0.1694i


s_cc(:,:,157) =

   0.1164 - 0.1479i   0.3332 + 0.0192i
   0.3334 + 0.0191i   0.0657 - 0.1624i
```

```
s_cc(:,:,158) =

   0.0720 - 0.1351i   0.3167 - 0.1253i
   0.3167 - 0.1250i   0.0241 - 0.1361i


s_cc(:,:,159) =

   0.0467 - 0.1145i   0.2342 - 0.2516i
   0.2342 - 0.2519i   0.0069 - 0.1078i


s_cc(:,:,160) =

   0.0387 - 0.0989i   0.1018 - 0.3277i
   0.1015 - 0.3272i   0.0087 - 0.0921i


s_cc(:,:,161) =

   0.0385 - 0.0962i  -0.0481 - 0.3343i
  -0.0483 - 0.3345i   0.0139 - 0.0990i


s_cc(:,:,162) =

   0.0358 - 0.1075i  -0.1797 - 0.2774i
  -0.1798 - 0.2773i   0.0018 - 0.1249i


s_cc(:,:,163) =

   0.0202 - 0.1298i  -0.2735 - 0.1765i
  -0.2732 - 0.1762i  -0.0389 - 0.1486i


s_cc(:,:,164) =

  -0.0160 - 0.1515i  -0.3182 - 0.0486i
  -0.3180 - 0.0487i  -0.0976 - 0.1508i


s_cc(:,:,165) =
```

```
  -0.0719 - 0.1587i  -0.3074 + 0.0862i
  -0.3071 + 0.0861i  -0.1599 - 0.1266i


s_cc(:,:,166) =

  -0.1378 - 0.1377i  -0.2420 + 0.2030i
  -0.2418 + 0.2029i  -0.2137 - 0.0766i


s_cc(:,:,167) =

  -0.1948 - 0.0845i  -0.1358 + 0.2812i
  -0.1359 + 0.2808i  -0.2478 - 0.0063i


s_cc(:,:,168) =

  -0.2259 - 0.0075i  -0.0095 + 0.3088i
  -0.0095 + 0.3087i  -0.2542 + 0.0742i


s_cc(:,:,169) =

  -0.2218 + 0.0768i   0.1154 + 0.2843i
   0.1153 + 0.2840i  -0.2310 + 0.1517i


s_cc(:,:,170) =

  -0.1839 + 0.1499i   0.2200 + 0.2133i
   0.2197 + 0.2130i  -0.1825 + 0.2135i


s_cc(:,:,171) =

  -0.1224 + 0.1970i   0.2885 + 0.1056i
   0.2885 + 0.1055i  -0.1193 + 0.2505i


s_cc(:,:,172) =

  -0.0551 + 0.2107i   0.3081 - 0.0232i
```

```
   0.3081 - 0.0229i  -0.0557 + 0.2598i


s_cc(:,:,173) =

  -0.0004 + 0.1949i   0.2720 - 0.1504i
   0.2716 - 0.1500i  -0.0047 + 0.2480i


s_cc(:,:,174) =

   0.0283 + 0.1641i   0.1834 - 0.2523i
   0.1832 - 0.2520i   0.0257 + 0.2269i


s_cc(:,:,175) =

   0.0295 + 0.1387i   0.0568 - 0.3048i
   0.0571 - 0.3047i   0.0361 + 0.2119i


s_cc(:,:,176) =

   0.0156 + 0.1348i  -0.0777 - 0.2951i
  -0.0776 - 0.2948i   0.0391 + 0.2163i


s_cc(:,:,177) =

   0.0067 + 0.1574i  -0.1901 - 0.2295i
  -0.1895 - 0.2294i   0.0559 + 0.2396i


s_cc(:,:,178) =

   0.0230 + 0.1966i  -0.2610 - 0.1269i
  -0.2610 - 0.1270i   0.0981 + 0.2618i


s_cc(:,:,179) =

   0.0715 + 0.2285i  -0.2826 - 0.0093i
  -0.2830 - 0.0093i   0.1595 + 0.2653i
```

```
s_cc(:,:,180) =

   0.1408 + 0.2325i  -0.2567 + 0.1017i
  -0.2566 + 0.1018i   0.2261 + 0.2411i


s_cc(:,:,181) =

   0.2109 + 0.2015i  -0.1926 + 0.1908i
  -0.1925 + 0.1903i   0.2832 + 0.1901i


s_cc(:,:,182) =

   0.2627 + 0.1393i  -0.1028 + 0.2488i
  -0.1034 + 0.2484i   0.3185 + 0.1178i


s_cc(:,:,183) =

   0.2816 + 0.0600i   0.0018 + 0.2702i
   0.0014 + 0.2699i   0.3234 + 0.0362i


s_cc(:,:,184) =

   0.2635 - 0.0161i   0.1094 + 0.2504i
   0.1095 + 0.2512i   0.2964 - 0.0375i


s_cc(:,:,185) =

   0.2176 - 0.0692i   0.2046 + 0.1894i
   0.2047 + 0.1897i   0.2459 - 0.0886i


s_cc(:,:,186) =

   0.1639 - 0.0898i   0.2708 + 0.0907i
   0.2706 + 0.0906i   0.1903 - 0.1090i


s_cc(:,:,187) =
```

```
   0.1225 - 0.0850i   0.2897 - 0.0332i
   0.2900 - 0.0332i   0.1472 - 0.1058i


s_cc(:,:,188) =

   0.1026 - 0.0700i   0.2493 - 0.1573i
   0.2492 - 0.1570i   0.1254 - 0.0943i


s_cc(:,:,189) =

   0.1018 - 0.0623i   0.1554 - 0.2492i
   0.1554 - 0.2492i   0.1207 - 0.0909i


s_cc(:,:,190) =

   0.1066 - 0.0717i   0.0323 - 0.2868i
   0.0321 - 0.2869i   0.1200 - 0.1047i


s_cc(:,:,191) =

   0.1029 - 0.0945i  -0.0890 - 0.2672i
  -0.0887 - 0.2672i   0.1078 - 0.1341i


s_cc(:,:,192) =

   0.0861 - 0.1223i  -0.1859 - 0.2036i
  -0.1857 - 0.2040i   0.0719 - 0.1668i


s_cc(:,:,193) =

   0.0533 - 0.1496i  -0.2495 - 0.1107i
  -0.2494 - 0.1107i   0.0139 - 0.1833i


s_cc(:,:,194) =

   0.0032 - 0.1669i  -0.2726 - 0.0002i
```

```
  -0.2725 - 0.0003i  -0.0530 - 0.1738i


s_cc(:,:,195) =

  -0.0593 - 0.1637i  -0.2490 + 0.1115i
  -0.2489 + 0.1114i  -0.1156 - 0.1371i


s_cc(:,:,196) =

  -0.1219 - 0.1333i  -0.1809 + 0.2040i
  -0.1806 + 0.2038i  -0.1616 - 0.0771i


s_cc(:,:,197) =

  -0.1698 - 0.0777i  -0.0806 + 0.2601i
  -0.0804 + 0.2594i  -0.1818 - 0.0030i


s_cc(:,:,198) =

  -0.1929 - 0.0075i   0.0341 + 0.2701i
   0.0341 + 0.2699i  -0.1726 + 0.0716i


s_cc(:,:,199) =

  -0.1877 + 0.0642i   0.1434 + 0.2318i
   0.1430 + 0.2315i  -0.1378 + 0.1338i


s_cc(:,:,200) =

  -0.1579 + 0.1240i   0.2268 + 0.1512i
   0.2269 + 0.1509i  -0.0866 + 0.1737i


s_cc(:,:,201) =

  -0.1139 + 0.1631i   0.2701 + 0.0418i
   0.2696 + 0.0419i  -0.0325 + 0.1880i
```

```
s_cc(:,:,202) =

  -0.0688 + 0.1801i   0.2618 - 0.0763i
   0.2616 - 0.0761i   0.0117 + 0.1815i


s_cc(:,:,203) =

  -0.0328 + 0.1813i   0.2037 - 0.1798i
   0.2032 - 0.1791i   0.0383 + 0.1654i


s_cc(:,:,204) =

  -0.0106 + 0.1768i   0.1060 - 0.2472i
   0.1062 - 0.2470i   0.0474 + 0.1523i


s_cc(:,:,205) =

  -0.0004 + 0.1770i  -0.0100 - 0.2641i
  -0.0097 - 0.2637i   0.0464 + 0.1547i


s_cc(:,:,206) =

   0.0066 + 0.1895i  -0.1183 - 0.2283i
  -0.1182 - 0.2286i   0.0532 + 0.1773i


s_cc(:,:,207) =

   0.0227 + 0.2156i  -0.1968 - 0.1534i
  -0.1964 - 0.1536i   0.0847 + 0.2071i


s_cc(:,:,208) =

   0.0606 + 0.2445i  -0.2351 - 0.0577i
  -0.2348 - 0.0581i   0.1408 + 0.2226i


s_cc(:,:,209) =
```

```
   0.1197 + 0.2588i  -0.2304 + 0.0406i
  -0.2305 + 0.0398i   0.2088 + 0.2109i


s_cc(:,:,210) =

   0.1885 + 0.2474i  -0.1904 + 0.1241i
  -0.1903 + 0.1235i   0.2716 + 0.1693i


s_cc(:,:,211) =

   0.2526 + 0.2074i  -0.1258 + 0.1838i
  -0.1260 + 0.1831i   0.3167 + 0.1036i


s_cc(:,:,212) =

   0.2985 + 0.1432i  -0.0466 + 0.2164i
  -0.0469 + 0.2163i   0.3335 + 0.0240i


s_cc(:,:,213) =

   0.3157 + 0.0664i   0.0401 + 0.2192i
   0.0395 + 0.2196i   0.3185 - 0.0559i


s_cc(:,:,214) =

   0.3015 - 0.0070i   0.1246 + 0.1899i
   0.1243 + 0.1900i   0.2749 - 0.1202i


s_cc(:,:,215) =

   0.2639 - 0.0614i   0.1946 + 0.1275i
   0.1948 + 0.1275i   0.2152 - 0.1572i


s_cc(:,:,216) =

   0.2188 - 0.0911i   0.2369 + 0.0368i
```

```
   0.2372 + 0.0367i   0.1572 - 0.1641i


s_cc(:,:,217) =

   0.1803 - 0.0999i   0.2352 - 0.0699i
   0.2354 - 0.0700i   0.1169 - 0.1511i


s_cc(:,:,218) =

   0.1564 - 0.0996i   0.1829 - 0.1672i
   0.1828 - 0.1671i   0.0997 - 0.1357i


s_cc(:,:,219) =

   0.1452 - 0.1037i   0.0907 - 0.2287i
   0.0905 - 0.2289i   0.0970 - 0.1348i


s_cc(:,:,220) =

   0.1355 - 0.1193i  -0.0166 - 0.2404i
  -0.0164 - 0.2406i   0.0912 - 0.1537i


s_cc(:,:,221) =

   0.1176 - 0.1423i  -0.1124 - 0.2058i
  -0.1126 - 0.2060i   0.0659 - 0.1838i


s_cc(:,:,222) =

   0.0879 - 0.1668i  -0.1823 - 0.1393i
  -0.1829 - 0.1395i   0.0167 - 0.2066i


s_cc(:,:,223) =

   0.0436 - 0.1872i  -0.2200 - 0.0530i
  -0.2204 - 0.0534i  -0.0484 - 0.2086i
```

```
s_cc(:,:,224) =

  -0.0153 - 0.1934i  -0.2209 + 0.0396i
  -0.2209 + 0.0401i  -0.1157 - 0.1831i


s_cc(:,:,225) =

  -0.0809 - 0.1763i  -0.1849 + 0.1248i
  -0.1849 + 0.1250i  -0.1729 - 0.1320i


s_cc(:,:,226) =

  -0.1399 - 0.1323i  -0.1186 + 0.1886i
  -0.1182 + 0.1883i  -0.2091 - 0.0617i


s_cc(:,:,227) =

  -0.1786 - 0.0674i  -0.0318 + 0.2200i
  -0.0318 + 0.2200i  -0.2186 + 0.0178i


s_cc(:,:,228) =

  -0.1899 + 0.0059i   0.0604 + 0.2143i
   0.0605 + 0.2144i  -0.1996 + 0.0934i


s_cc(:,:,229) =

  -0.1747 + 0.0746i   0.1433 + 0.1720i
   0.1436 + 0.1718i  -0.1576 + 0.1540i


s_cc(:,:,230) =

  -0.1388 + 0.1280i   0.2031 + 0.0979i
   0.2032 + 0.0979i  -0.1034 + 0.1926i


s_cc(:,:,231) =
```

```
  -0.0931 + 0.1604i   0.2267 + 0.0037i
   0.2263 + 0.0038i  -0.0482 + 0.2084i


s_cc(:,:,232) =

  -0.0494 + 0.1724i   0.2075 - 0.0921i
   0.2073 - 0.0922i  -0.0011 + 0.2080i


s_cc(:,:,233) =

  -0.0158 + 0.1714i   0.1492 - 0.1708i
   0.1493 - 0.1709i   0.0336 + 0.1986i


s_cc(:,:,234) =

   0.0047 + 0.1662i   0.0624 - 0.2174i
   0.0627 - 0.2172i   0.0555 + 0.1888i


s_cc(:,:,235) =

   0.0137 + 0.1668i  -0.0367 - 0.2209i
  -0.0365 - 0.2206i   0.0704 + 0.1879i


s_cc(:,:,236) =

   0.0208 + 0.1807i  -0.1244 - 0.1801i
  -0.1241 - 0.1802i   0.0905 + 0.1980i


s_cc(:,:,237) =

   0.0407 + 0.2068i  -0.1834 - 0.1088i
  -0.1834 - 0.1089i   0.1265 + 0.2106i


s_cc(:,:,238) =

   0.0810 + 0.2303i  -0.2057 - 0.0230i
```

```
  -0.2060 - 0.0231i   0.1791 + 0.2106i


s_cc(:,:,239) =

   0.1373 + 0.2376i  -0.1916 + 0.0599i
  -0.1917 + 0.0600i   0.2372 + 0.1874i


s_cc(:,:,240) =

   0.1988 + 0.2218i  -0.1487 + 0.1268i
  -0.1489 + 0.1266i   0.2883 + 0.1410i


s_cc(:,:,241) =

   0.2536 + 0.1810i  -0.0879 + 0.1708i
  -0.0878 + 0.1712i   0.3222 + 0.0766i


s_cc(:,:,242) =

   0.2894 + 0.1199i  -0.0162 + 0.1913i
  -0.0163 + 0.1916i   0.3319 + 0.0017i


s_cc(:,:,243) =

   0.2974 + 0.0512i   0.0594 + 0.1858i
   0.0593 + 0.1859i   0.3148 - 0.0711i


s_cc(:,:,244) =

   0.2787 - 0.0106i   0.1308 + 0.1510i
   0.1308 + 0.1511i   0.2750 - 0.1295i


s_cc(:,:,245) =

   0.2410 - 0.0532i   0.1861 + 0.0871i
   0.1862 + 0.0875i   0.2228 - 0.1664i
```

```
s_cc(:,:,246) =

   0.1989 - 0.0715i   0.2113 + 0.0013i
   0.2115 + 0.0015i   0.1714 - 0.1812i


s_cc(:,:,247) =

   0.1672 - 0.0700i   0.1950 - 0.0914i
   0.1953 - 0.0911i   0.1310 - 0.1821i


s_cc(:,:,248) =

   0.1537 - 0.0620i   0.1371 - 0.1683i
   0.1376 - 0.1688i   0.1018 - 0.1808i


s_cc(:,:,249) =

   0.1543 - 0.0622i   0.0505 - 0.2102i
   0.0508 - 0.2106i   0.0770 - 0.1857i


s_cc(:,:,250) =

   0.1568 - 0.0774i  -0.0437 - 0.2087i
  -0.0439 - 0.2091i   0.0456 - 0.1957i


s_cc(:,:,251) =

   0.1493 - 0.1033i  -0.1251 - 0.1686i
  -0.1252 - 0.1687i   0.0022 - 0.2017i


s_cc(:,:,252) =

   0.1277 - 0.1315i  -0.1807 - 0.1011i
  -0.1812 - 0.1010i  -0.0500 - 0.1937i


s_cc(:,:,253) =
```

```
   0.0915 - 0.1542i  -0.2050 - 0.0182i
  -0.2051 - 0.0182i  -0.1022 - 0.1669i


s_cc(:,:,254) =

   0.0429 - 0.1638i  -0.1943 + 0.0673i
  -0.1943 + 0.0676i  -0.1458 - 0.1227i


s_cc(:,:,255) =

  -0.0108 - 0.1537i  -0.1493 + 0.1418i
  -0.1491 + 0.1420i  -0.1736 - 0.0657i


s_cc(:,:,256) =

  -0.0588 - 0.1224i  -0.0770 + 0.1912i
  -0.0768 + 0.1912i  -0.1818 - 0.0036i


s_cc(:,:,257) =

  -0.0912 - 0.0746i   0.0101 + 0.2066i
   0.0102 + 0.2063i  -0.1709 + 0.0556i


s_cc(:,:,258) =

  -0.1017 - 0.0207i   0.0964 + 0.1832i
   0.0964 + 0.1830i  -0.1449 + 0.1052i


s_cc(:,:,259) =

  -0.0920 + 0.0269i   0.1648 + 0.1245i
   0.1646 + 0.1243i  -0.1106 + 0.1411i


s_cc(:,:,260) =

  -0.0712 + 0.0605i   0.2005 + 0.0418i
```

```
   0.2004 + 0.0418i  -0.0751 + 0.1648i


s_cc(:,:,261) =

  -0.0501 + 0.0812i   0.1963 - 0.0465i
   0.1963 - 0.0466i  -0.0420 + 0.1809i


s_cc(:,:,262) =

  -0.0331 + 0.0949i   0.1564 - 0.1223i
   0.1564 - 0.1222i  -0.0101 + 0.1932i


s_cc(:,:,263) =

  -0.0203 + 0.1062i   0.0906 - 0.1743i
   0.0903 - 0.1741i   0.0215 + 0.2017i


s_cc(:,:,264) =

  -0.0128 + 0.1193i   0.0089 - 0.1946i
   0.0087 - 0.1944i   0.0522 + 0.2081i


s_cc(:,:,265) =

  -0.0057 + 0.1409i  -0.0737 - 0.1774i
  -0.0735 - 0.1771i   0.0846 + 0.2152i


s_cc(:,:,266) =

   0.0111 + 0.1714i  -0.1387 - 0.1264i
  -0.1384 - 0.1265i   0.1250 + 0.2200i


s_cc(:,:,267) =

   0.0446 + 0.2021i  -0.1737 - 0.0556i
  -0.1734 - 0.0558i   0.1745 + 0.2151i
```

```
s_cc(:,:,268) =

   0.0944 + 0.2222i  -0.1756 + 0.0196i
  -0.1755 + 0.0194i   0.2264 + 0.1932i


s_cc(:,:,269) =

   0.1536 + 0.2236i  -0.1485 + 0.0854i
  -0.1486 + 0.0851i   0.2723 + 0.1538i


s_cc(:,:,270) =

   0.2132 + 0.2026i  -0.1009 + 0.1330i
  -0.1012 + 0.1330i   0.3067 + 0.1011i


s_cc(:,:,271) =

   0.2630 + 0.1608i  -0.0422 + 0.1596i
  -0.0423 + 0.1597i   0.3240 + 0.0385i


s_cc(:,:,272) =

   0.2931 + 0.1039i   0.0221 + 0.1644i
   0.0221 + 0.1644i   0.3210 - 0.0274i


s_cc(:,:,273) =

   0.2991 + 0.0449i   0.0861 + 0.1452i
   0.0863 + 0.1454i   0.2971 - 0.0865i


s_cc(:,:,274) =

   0.2837 - 0.0050i   0.1408 + 0.1007i
   0.1408 + 0.1007i   0.2572 - 0.1292i


s_cc(:,:,275) =
```

```
   0.2586 - 0.0368i   0.1741 + 0.0338i
   0.1743 + 0.0338i   0.2142 - 0.1529i


s_cc(:,:,276) =

   0.2359 - 0.0522i   0.1751 - 0.0448i
   0.1756 - 0.0447i   0.1781 - 0.1617i


s_cc(:,:,277) =

   0.2240 - 0.0598i   0.1394 - 0.1174i
   0.1396 - 0.1177i   0.1526 - 0.1666i


s_cc(:,:,278) =

   0.2225 - 0.0716i   0.0739 - 0.1656i
   0.0739 - 0.1660i   0.1320 - 0.1772i


s_cc(:,:,279) =

   0.2218 - 0.0950i  -0.0049 - 0.1788i
  -0.0051 - 0.1789i   0.1052 - 0.1939i


s_cc(:,:,280) =

   0.2113 - 0.1274i  -0.0788 - 0.1564i
  -0.0790 - 0.1564i   0.0663 - 0.2090i


s_cc(:,:,281) =

   0.1856 - 0.1594i  -0.1340 - 0.1070i
  -0.1341 - 0.1070i   0.0167 - 0.2117i


s_cc(:,:,282) =

   0.1476 - 0.1833i  -0.1634 - 0.0412i
```

```
  -0.1634 - 0.0411i  -0.0347 - 0.1970i


s_cc(:,:,283) =

   0.1025 - 0.1932i  -0.1630 + 0.0295i
  -0.1632 + 0.0295i  -0.0795 - 0.1674i


s_cc(:,:,284) =

   0.0556 - 0.1894i  -0.1341 + 0.0923i
  -0.1342 + 0.0925i  -0.1145 - 0.1279i


s_cc(:,:,285) =

   0.0113 - 0.1709i  -0.0836 + 0.1367i
  -0.0835 + 0.1370i  -0.1386 - 0.0805i


s_cc(:,:,286) =

  -0.0233 - 0.1374i  -0.0209 + 0.1575i
  -0.0206 + 0.1578i  -0.1493 - 0.0281i


s_cc(:,:,287) =

  -0.0402 - 0.0953i   0.0455 + 0.1516i
   0.0459 + 0.1516i  -0.1440 + 0.0232i


s_cc(:,:,288) =

  -0.0384 - 0.0580i   0.1040 + 0.1180i
   0.1041 + 0.1178i  -0.1272 + 0.0674i


s_cc(:,:,289) =

  -0.0279 - 0.0337i   0.1414 + 0.0625i
   0.1415 + 0.0624i  -0.1041 + 0.1041i
```

```
s_cc(:,:,290) =

  -0.0201 - 0.0197i   0.1509 - 0.0010i
   0.1508 - 0.0010i  -0.0772 + 0.1358i


s_cc(:,:,291) =

  -0.0178 - 0.0072i   0.1354 - 0.0594i
   0.1352 - 0.0595i  -0.0440 + 0.1635i


s_cc(:,:,292) =

  -0.0163 + 0.0073i   0.1010 - 0.1070i
   0.1010 - 0.1071i  -0.0038 + 0.1827i


s_cc(:,:,293) =

  -0.0144 + 0.0233i   0.0503 - 0.1402i
   0.0503 - 0.1401i   0.0379 + 0.1905i


s_cc(:,:,294) =

  -0.0121 + 0.0418i  -0.0128 - 0.1506i
  -0.0127 - 0.1506i   0.0774 + 0.1899i


s_cc(:,:,295) =

  -0.0072 + 0.0651i  -0.0764 - 0.1313i
  -0.0762 - 0.1314i   0.1141 + 0.1836i


s_cc(:,:,296) =

   0.0060 + 0.0931i  -0.1245 - 0.0850i
  -0.1244 - 0.0849i   0.1485 + 0.1724i


s_cc(:,:,297) =
```

```
   0.0325 + 0.1207i  -0.1459 - 0.0232i
  -0.1460 - 0.0232i   0.1812 + 0.1558i


s_cc(:,:,298) =

   0.0716 + 0.1387i  -0.1385 + 0.0391i
  -0.1385 + 0.0391i   0.2106 + 0.1352i


s_cc(:,:,299) =

   0.1169 + 0.1413i  -0.1071 + 0.0902i
  -0.1071 + 0.0904i   0.2386 + 0.1087i


s_cc(:,:,300) =

   0.1607 + 0.1268i  -0.0597 + 0.1228i
  -0.0598 + 0.1227i   0.2631 + 0.0744i


s_cc(:,:,301) =

   0.1948 + 0.0990i  -0.0054 + 0.1333i
  -0.0055 + 0.1334i   0.2798 + 0.0324i


s_cc(:,:,302) =

   0.2154 + 0.0641i   0.0469 + 0.1223i
   0.0468 + 0.1224i   0.2845 - 0.0142i


s_cc(:,:,303) =

   0.2237 + 0.0293i   0.0902 + 0.0931i
   0.0902 + 0.0933i   0.2763 - 0.0599i


s_cc(:,:,304) =

   0.2203 - 0.0017i   0.1197 + 0.0502i
```

```
   0.1198 + 0.0503i   0.2577 - 0.1002i


s_cc(:,:,305) =

   0.2101 - 0.0221i   0.1306 - 0.0027i
   0.1307 - 0.0025i   0.2326 - 0.1322i


s_cc(:,:,306) =

   0.2042 - 0.0305i   0.1180 - 0.0564i
   0.1181 - 0.0565i   0.2044 - 0.1567i


s_cc(:,:,307) =

   0.2101 - 0.0391i   0.0833 - 0.0996i
   0.0833 - 0.0996i   0.1756 - 0.1771i


s_cc(:,:,308) =

   0.2211 - 0.0591i   0.0348 - 0.1233i
   0.0349 - 0.1234i   0.1435 - 0.1939i


s_cc(:,:,309) =

   0.2251 - 0.0905i  -0.0177 - 0.1254i
  -0.0177 - 0.1255i   0.1081 - 0.2053i


s_cc(:,:,310) =

   0.2166 - 0.1263i  -0.0656 - 0.1069i
  -0.0659 - 0.1072i   0.0701 - 0.2097i


s_cc(:,:,311) =

   0.1953 - 0.1595i  -0.1027 - 0.0719i
  -0.1028 - 0.0720i   0.0312 - 0.2062i
```

```
s_cc(:,:,312) =

   0.1638 - 0.1849i  -0.1237 - 0.0253i
  -0.1240 - 0.0253i  -0.0074 - 0.1943i


s_cc(:,:,313) =

   0.1259 - 0.1994i  -0.1256 + 0.0277i
  -0.1255 + 0.0277i  -0.0430 - 0.1732i


s_cc(:,:,314) =

   0.0881 - 0.2013i  -0.1039 + 0.0792i
  -0.1041 + 0.0793i  -0.0722 - 0.1440i


s_cc(:,:,315) =

   0.0541 - 0.1944i  -0.0601 + 0.1172i
  -0.0600 + 0.1173i  -0.0929 - 0.1103i


s_cc(:,:,316) =

   0.0256 - 0.1825i  -0.0031 + 0.1305i
  -0.0031 + 0.1306i  -0.1068 - 0.0753i


s_cc(:,:,317) =

   0.0009 - 0.1681i   0.0521 + 0.1164i
   0.0521 + 0.1167i  -0.1156 - 0.0395i


s_cc(:,:,318) =

  -0.0224 - 0.1504i   0.0932 + 0.0816i
   0.0934 + 0.0816i  -0.1195 - 0.0007i


s_cc(:,:,319) =
```

```
  -0.0424 - 0.1273i   0.1148 + 0.0356i
   0.1148 + 0.0355i  -0.1149 + 0.0419i


s_cc(:,:,320) =

  -0.0566 - 0.1004i   0.1169 - 0.0125i
   0.1169 - 0.0126i  -0.0976 + 0.0849i


s_cc(:,:,321) =

  -0.0639 - 0.0722i   0.1017 - 0.0571i
   0.1015 - 0.0571i  -0.0669 + 0.1217i


s_cc(:,:,322) =

  -0.0663 - 0.0453i   0.0702 - 0.0930i
   0.0703 - 0.0929i  -0.0270 + 0.1461i


s_cc(:,:,323) =

  -0.0653 - 0.0179i   0.0265 - 0.1133i
   0.0266 - 0.1133i   0.0158 + 0.1565i


s_cc(:,:,324) =

  -0.0598 + 0.0084i  -0.0225 - 0.1144i
  -0.0222 - 0.1144i   0.0550 + 0.1537i


s_cc(:,:,325) =

  -0.0520 + 0.0359i  -0.0678 - 0.0940i
  -0.0678 - 0.0942i   0.0862 + 0.1427i


s_cc(:,:,326) =

  -0.0370 + 0.0668i  -0.0995 - 0.0568i
```

```
  -0.0995 - 0.0567i   0.1082 + 0.1300i


s_cc(:,:,327) =

  -0.0096 + 0.0968i  -0.1119 - 0.0106i
  -0.1118 - 0.0106i   0.1258 + 0.1212i


s_cc(:,:,328) =

   0.0306 + 0.1172i  -0.1044 + 0.0347i
  -0.1045 + 0.0347i   0.1459 + 0.1166i


s_cc(:,:,329) =

   0.0771 + 0.1216i  -0.0805 + 0.0720i
  -0.0807 + 0.0721i   0.1729 + 0.1094i


s_cc(:,:,330) =

   0.1217 + 0.1100i  -0.0454 + 0.0966i
  -0.0455 + 0.0965i   0.2045 + 0.0927i


s_cc(:,:,331) =

   0.1592 + 0.0842i  -0.0047 + 0.1060i
  -0.0047 + 0.1061i   0.2332 + 0.0627i


s_cc(:,:,332) =

   0.1839 + 0.0486i   0.0368 + 0.1003i
   0.0368 + 0.1004i   0.2520 + 0.0226i


s_cc(:,:,333) =

   0.1929 + 0.0094i   0.0745 + 0.0794i
   0.0744 + 0.0795i   0.2570 - 0.0225i
```

```
s_cc(:,:,334) =

   0.1874 - 0.0241i   0.1028 + 0.0442i
   0.1029 + 0.0442i   0.2469 - 0.0665i


s_cc(:,:,335) =

   0.1737 - 0.0474i   0.1152 - 0.0023i
   0.1153 - 0.0023i   0.2252 - 0.1031i


s_cc(:,:,336) =

   0.1611 - 0.0587i   0.1058 - 0.0522i
   0.1059 - 0.0523i   0.1966 - 0.1304i


s_cc(:,:,337) =

   0.1557 - 0.0652i   0.0741 - 0.0943i
   0.0741 - 0.0943i   0.1642 - 0.1488i


s_cc(:,:,338) =

   0.1572 - 0.0756i   0.0258 - 0.1176i
   0.0258 - 0.1176i   0.1300 - 0.1586i


s_cc(:,:,339) =

   0.1579 - 0.0947i  -0.0275 - 0.1167i
  -0.0274 - 0.1168i   0.0954 - 0.1593i


s_cc(:,:,340) =

   0.1504 - 0.1192i  -0.0743 - 0.0929i
  -0.0743 - 0.0929i   0.0641 - 0.1499i


s_cc(:,:,341) =
```

```
   0.1321 - 0.1431i  -0.1056 - 0.0520i
  -0.1057 - 0.0520i   0.0392 - 0.1341i


s_cc(:,:,342) =

   0.1054 - 0.1599i  -0.1167 - 0.0024i
  -0.1166 - 0.0025i   0.0230 - 0.1152i


s_cc(:,:,343) =

   0.0745 - 0.1676i  -0.1058 + 0.0468i
  -0.1059 + 0.0469i   0.0129 - 0.0989i


s_cc(:,:,344) =

   0.0443 - 0.1659i  -0.0750 + 0.0870i
  -0.0749 + 0.0869i   0.0037 - 0.0849i


s_cc(:,:,345) =

   0.0179 - 0.1576i  -0.0297 + 0.1094i
  -0.0296 + 0.1094i  -0.0068 - 0.0701i


s_cc(:,:,346) =

  -0.0043 - 0.1462i   0.0205 + 0.1089i
   0.0206 + 0.1089i  -0.0176 - 0.0512i


s_cc(:,:,347) =

  -0.0246 - 0.1331i   0.0635 + 0.0861i
   0.0635 + 0.0861i  -0.0260 - 0.0261i


s_cc(:,:,348) =

  -0.0449 - 0.1172i   0.0895 + 0.0486i
```

```
   0.0895 + 0.0485i  -0.0273 + 0.0062i


s_cc(:,:,349) =

  -0.0640 - 0.0947i   0.0960 + 0.0073i
   0.0960 + 0.0073i  -0.0157 + 0.0425i


s_cc(:,:,350) =

  -0.0771 - 0.0650i   0.0865 - 0.0292i
   0.0865 - 0.0292i   0.0124 + 0.0749i


s_cc(:,:,351) =

  -0.0793 - 0.0318i   0.0663 - 0.0571i
   0.0663 - 0.0569i   0.0539 + 0.0938i


s_cc(:,:,352) =

  -0.0709 - 0.0023i   0.0399 - 0.0754i
   0.0398 - 0.0753i   0.0998 + 0.0928i


s_cc(:,:,353) =

  -0.0579 + 0.0194i   0.0095 - 0.0841i
   0.0096 - 0.0842i   0.1386 + 0.0726i


s_cc(:,:,354) =

  -0.0468 + 0.0363i  -0.0226 - 0.0825i
  -0.0226 - 0.0823i   0.1605 + 0.0409i


s_cc(:,:,355) =

  -0.0373 + 0.0549i  -0.0531 - 0.0688i
  -0.0531 - 0.0687i   0.1642 + 0.0107i
```

```
s_cc(:,:,356) =

  -0.0228 + 0.0776i  -0.0772 - 0.0440i
  -0.0772 - 0.0440i   0.1570 - 0.0081i


s_cc(:,:,357) =

   0.0015 + 0.0996i  -0.0910 - 0.0102i
  -0.0909 - 0.0103i   0.1508 - 0.0134i


s_cc(:,:,358) =

   0.0360 + 0.1133i  -0.0903 + 0.0284i
  -0.0903 + 0.0282i   0.1541 - 0.0122i


s_cc(:,:,359) =

   0.0745 + 0.1127i  -0.0726 + 0.0654i
  -0.0727 + 0.0652i   0.1681 - 0.0158i


s_cc(:,:,360) =

   0.1077 + 0.0978i  -0.0389 + 0.0927i
  -0.0391 + 0.0925i   0.1849 - 0.0309i


s_cc(:,:,361) =

   0.1299 + 0.0746i   0.0051 + 0.1023i
   0.0049 + 0.1024i   0.1961 - 0.0560i


s_cc(:,:,362) =

   0.1395 + 0.0508i   0.0498 + 0.0909i
   0.0497 + 0.0910i   0.1974 - 0.0870i


s_cc(:,:,363) =
```

```
   0.1408 + 0.0326i   0.0848 + 0.0601i
   0.0850 + 0.0603i   0.1872 - 0.1187i


s_cc(:,:,364) =

   0.1396 + 0.0223i   0.1024 + 0.0167i
   0.1025 + 0.0168i   0.1668 - 0.1456i


s_cc(:,:,365) =

   0.1424 + 0.0180i   0.0982 - 0.0298i
   0.0982 - 0.0297i   0.1386 - 0.1652i


s_cc(:,:,366) =

   0.1526 + 0.0139i   0.0735 - 0.0685i
   0.0737 - 0.0686i   0.1055 - 0.1754i


s_cc(:,:,367) =

   0.1684 + 0.0028i   0.0354 - 0.0907i
   0.0355 - 0.0909i   0.0701 - 0.1751i


s_cc(:,:,368) =

   0.1828 - 0.0182i  -0.0062 - 0.0936i
  -0.0062 - 0.0938i   0.0372 - 0.1621i


s_cc(:,:,369) =

   0.1903 - 0.0472i  -0.0428 - 0.0796i
  -0.0430 - 0.0796i   0.0129 - 0.1372i


s_cc(:,:,370) =

   0.1868 - 0.0797i  -0.0690 - 0.0533i
```

```
  -0.0692 - 0.0532i   0.0028 - 0.1077i


s_cc(:,:,371) =

   0.1718 - 0.1098i  -0.0820 - 0.0204i
  -0.0821 - 0.0203i   0.0072 - 0.0818i


s_cc(:,:,372) =

   0.1483 - 0.1331i  -0.0813 + 0.0135i
  -0.0813 + 0.0137i   0.0200 - 0.0675i


s_cc(:,:,373) =

   0.1202 - 0.1479i  -0.0680 + 0.0438i
  -0.0678 + 0.0439i   0.0311 - 0.0657i


s_cc(:,:,374) =

   0.0901 - 0.1550i  -0.0441 + 0.0664i
  -0.0440 + 0.0663i   0.0328 - 0.0701i


s_cc(:,:,375) =

   0.0586 - 0.1549i  -0.0133 + 0.0771i
  -0.0132 + 0.0770i   0.0239 - 0.0719i


s_cc(:,:,376) =

   0.0270 - 0.1459i   0.0183 + 0.0741i
   0.0184 + 0.0741i   0.0094 - 0.0638i


s_cc(:,:,377) =

  -0.0021 - 0.1267i   0.0449 + 0.0593i
   0.0449 + 0.0593i  -0.0023 - 0.0425i
```

```
s_cc(:,:,378) =

  -0.0237 - 0.0979i   0.0626 + 0.0368i
   0.0627 + 0.0367i  -0.0022 - 0.0121i


s_cc(:,:,379) =

  -0.0327 - 0.0629i   0.0710 + 0.0104i
   0.0709 + 0.0103i   0.0146 + 0.0189i


s_cc(:,:,380) =

  -0.0278 - 0.0301i   0.0696 - 0.0168i
   0.0695 - 0.0168i   0.0464 + 0.0397i


s_cc(:,:,381) =

  -0.0136 - 0.0053i   0.0588 - 0.0420i
   0.0588 - 0.0420i   0.0846 + 0.0428i


s_cc(:,:,382) =

   0.0029 + 0.0099i   0.0396 - 0.0624i
   0.0396 - 0.0625i   0.1181 + 0.0285i


s_cc(:,:,383) =

   0.0176 + 0.0200i   0.0130 - 0.0755i
   0.0129 - 0.0755i   0.1385 + 0.0034i


s_cc(:,:,384) =

   0.0311 + 0.0287i  -0.0190 - 0.0776i
  -0.0189 - 0.0776i   0.1440 - 0.0222i


s_cc(:,:,385) =
```

```
   0.0459 + 0.0369i  -0.0517 - 0.0658i
  -0.0516 - 0.0658i   0.1400 - 0.0405i


s_cc(:,:,386) =

   0.0637 + 0.0434i  -0.0778 - 0.0387i
  -0.0777 - 0.0388i   0.1350 - 0.0493i


s_cc(:,:,387) =

   0.0849 + 0.0464i  -0.0894 - 0.0008i
  -0.0894 - 0.0009i   0.1350 - 0.0528i


s_cc(:,:,388) =

   0.1086 + 0.0432i  -0.0819 + 0.0394i
  -0.0819 + 0.0393i   0.1415 - 0.0575i


s_cc(:,:,389) =

   0.1316 + 0.0322i  -0.0563 + 0.0719i
  -0.0563 + 0.0718i   0.1515 - 0.0684i


s_cc(:,:,390) =

   0.1502 + 0.0139i  -0.0189 + 0.0889i
  -0.0191 + 0.0888i   0.1602 - 0.0870i


s_cc(:,:,391) =

   0.1609 - 0.0090i   0.0215 + 0.0875i
   0.0214 + 0.0875i   0.1616 - 0.1119i


s_cc(:,:,392) =

   0.1631 - 0.0317i   0.0564 + 0.0687i
```

```
   0.0563 + 0.0688i   0.1539 - 0.1390i


s_cc(:,:,393) =

   0.1589 - 0.0504i   0.0792 + 0.0371i
   0.0793 + 0.0371i   0.1348 - 0.1632i


s_cc(:,:,394) =

   0.1523 - 0.0636i   0.0859 - 0.0004i
   0.0858 - 0.0003i   0.1076 - 0.1795i


s_cc(:,:,395) =

   0.1468 - 0.0730i   0.0757 - 0.0358i
   0.0757 - 0.0357i   0.0755 - 0.1857i


s_cc(:,:,396) =

   0.1425 - 0.0820i   0.0526 - 0.0623i
   0.0528 - 0.0622i   0.0428 - 0.1791i


s_cc(:,:,397) =

   0.1374 - 0.0917i   0.0224 - 0.0765i
   0.0224 - 0.0766i   0.0156 - 0.1607i


s_cc(:,:,398) =

   0.1297 - 0.1004i  -0.0101 - 0.0782i
  -0.0101 - 0.0782i  -0.0012 - 0.1336i


s_cc(:,:,399) =

   0.1210 - 0.1054i  -0.0409 - 0.0674i
  -0.0409 - 0.0674i  -0.0039 - 0.1045i
```

```
s_cc(:,:,400) =

   0.1152 - 0.1073i  -0.0654 - 0.0447i
  -0.0655 - 0.0445i   0.0062 - 0.0809i


s_cc(:,:,401) =

   0.1138 - 0.1109i  -0.0782 - 0.0126i
  -0.0783 - 0.0125i   0.0233 - 0.0687i


s_cc(:,:,402) =

   0.1136 - 0.1199i  -0.0754 + 0.0218i
  -0.0753 + 0.0220i   0.0400 - 0.0683i


s_cc(:,:,403) =

   0.1091 - 0.1339i  -0.0580 + 0.0510i
  -0.0579 + 0.0511i   0.0505 - 0.0767i


s_cc(:,:,404) =

   0.0969 - 0.1495i  -0.0304 + 0.0692i
  -0.0303 + 0.0692i   0.0518 - 0.0878i


s_cc(:,:,405) =

   0.0773 - 0.1613i   0.0012 + 0.0738i
   0.0013 + 0.0737i   0.0449 - 0.0952i


s_cc(:,:,406) =

   0.0528 - 0.1651i   0.0308 + 0.0650i
   0.0307 + 0.0650i   0.0345 - 0.0951i


s_cc(:,:,407) =
```

```
   0.0286 - 0.1594i   0.0529 + 0.0456i
   0.0530 + 0.0456i   0.0267 - 0.0864i


s_cc(:,:,408) =

   0.0091 - 0.1458i   0.0650 + 0.0202i
   0.0650 + 0.0201i   0.0271 - 0.0727i


s_cc(:,:,409) =

  -0.0023 - 0.1281i   0.0663 - 0.0067i
   0.0662 - 0.0067i   0.0380 - 0.0604i


s_cc(:,:,410) =

  -0.0058 - 0.1111i   0.0581 - 0.0312i
   0.0581 - 0.0312i   0.0565 - 0.0575i


s_cc(:,:,411) =

  -0.0043 - 0.0987i   0.0421 - 0.0514i
   0.0421 - 0.0513i   0.0745 - 0.0668i


s_cc(:,:,412) =

  -0.0020 - 0.0914i   0.0193 - 0.0651i
   0.0193 - 0.0649i   0.0836 - 0.0858i


s_cc(:,:,413) =

  -0.0020 - 0.0873i  -0.0085 - 0.0695i
  -0.0085 - 0.0696i   0.0803 - 0.1067i


s_cc(:,:,414) =

  -0.0050 - 0.0838i  -0.0382 - 0.0618i
```

```
  -0.0381 - 0.0618i   0.0666 - 0.1223i


s_cc(:,:,415) =

  -0.0109 - 0.0791i  -0.0629 - 0.0406i
  -0.0630 - 0.0406i   0.0476 - 0.1289i


s_cc(:,:,416) =

  -0.0185 - 0.0711i  -0.0758 - 0.0089i
  -0.0758 - 0.0090i   0.0281 - 0.1262i


s_cc(:,:,417) =

  -0.0255 - 0.0582i  -0.0726 + 0.0257i
  -0.0725 + 0.0256i   0.0128 - 0.1153i


s_cc(:,:,418) =

  -0.0280 - 0.0412i  -0.0538 + 0.0551i
  -0.0538 + 0.0550i   0.0052 - 0.0999i


s_cc(:,:,419) =

  -0.0245 - 0.0240i  -0.0242 + 0.0728i
  -0.0241 + 0.0728i   0.0064 - 0.0854i


s_cc(:,:,420) =

  -0.0166 - 0.0098i   0.0101 + 0.0758i
   0.0101 + 0.0759i   0.0134 - 0.0769i


s_cc(:,:,421) =

  -0.0074 + 0.0006i   0.0425 + 0.0634i
   0.0426 + 0.0634i   0.0207 - 0.0762i
```

```
s_cc(:,:,422) =

   0.0008 + 0.0094i   0.0658 + 0.0373i
   0.0659 + 0.0372i   0.0234 - 0.0810i


s_cc(:,:,423) =

   0.0094 + 0.0196i   0.0744 + 0.0035i
   0.0743 + 0.0035i   0.0188 - 0.0865i


s_cc(:,:,424) =

   0.0212 + 0.0304i   0.0661 - 0.0293i
   0.0662 - 0.0293i   0.0078 - 0.0867i


s_cc(:,:,425) =

   0.0369 + 0.0399i   0.0447 - 0.0539i
   0.0448 - 0.0540i  -0.0046 - 0.0782i


s_cc(:,:,426) =

   0.0560 + 0.0464i   0.0158 - 0.0657i
   0.0158 - 0.0656i  -0.0124 - 0.0601i


s_cc(:,:,427) =

   0.0779 + 0.0498i  -0.0139 - 0.0634i
  -0.0140 - 0.0633i  -0.0092 - 0.0356i


s_cc(:,:,428) =

   0.1031 + 0.0486i  -0.0385 - 0.0488i
  -0.0384 - 0.0488i   0.0074 - 0.0126i


s_cc(:,:,429) =
```

```
   0.1301 + 0.0399i  -0.0533 - 0.0262i
  -0.0533 - 0.0262i   0.0351 + 0.0015i


s_cc(:,:,430) =

   0.1546 + 0.0228i  -0.0565 - 0.0006i
  -0.0565 - 0.0006i   0.0675 + 0.0016i


s_cc(:,:,431) =

   0.1737 + 0.0009i  -0.0489 + 0.0224i
  -0.0489 + 0.0224i   0.0966 - 0.0120i


s_cc(:,:,432) =

   0.1890 - 0.0242i  -0.0330 + 0.0388i
  -0.0328 + 0.0389i   0.1164 - 0.0356i


s_cc(:,:,433) =

   0.2012 - 0.0540i  -0.0129 + 0.0464i
  -0.0129 + 0.0463i   0.1240 - 0.0630i


s_cc(:,:,434) =

   0.2071 - 0.0900i   0.0071 + 0.0450i
   0.0072 + 0.0451i   0.1203 - 0.0883i


s_cc(:,:,435) =

   0.2021 - 0.1315i   0.0237 + 0.0361i
   0.0238 + 0.0360i   0.1088 - 0.1068i


s_cc(:,:,436) =

   0.1814 - 0.1725i   0.0346 + 0.0216i
```

```
   0.0345 + 0.0217i   0.0946 - 0.1161i


s_cc(:,:,437) =

   0.1466 - 0.2054i   0.0381 + 0.0051i
   0.0380 + 0.0050i   0.0838 - 0.1178i


s_cc(:,:,438) =

   0.1030 - 0.2247i   0.0343 - 0.0103i
   0.0341 - 0.0102i   0.0803 - 0.1168i


s_cc(:,:,439) =

   0.0570 - 0.2296i   0.0252 - 0.0214i
   0.0251 - 0.0214i   0.0838 - 0.1197i


s_cc(:,:,440) =

   0.0139 - 0.2216i   0.0137 - 0.0270i
   0.0137 - 0.0269i   0.0886 - 0.1314i


s_cc(:,:,441) =

  -0.0235 - 0.2043i   0.0027 - 0.0278i
   0.0029 - 0.0278i   0.0885 - 0.1515i


s_cc(:,:,442) =

  -0.0556 - 0.1815i  -0.0068 - 0.0254i
  -0.0068 - 0.0254i   0.0774 - 0.1756i


s_cc(:,:,443) =

  -0.0847 - 0.1530i  -0.0148 - 0.0204i
  -0.0148 - 0.0203i   0.0546 - 0.1965i
```

```
s_cc(:,:,444) =

  -0.1099 - 0.1152i  -0.0202 - 0.0130i
  -0.0202 - 0.0130i   0.0221 - 0.2079i


s_cc(:,:,445) =

  -0.1252 - 0.0667i  -0.0223 - 0.0046i
  -0.0223 - 0.0047i  -0.0141 - 0.2055i


s_cc(:,:,446) =

  -0.1244 - 0.0115i  -0.0214 + 0.0030i
  -0.0214 + 0.0031i  -0.0469 - 0.1886i


s_cc(:,:,447) =

  -0.1041 + 0.0426i  -0.0187 + 0.0094i
  -0.0187 + 0.0093i  -0.0691 - 0.1611i


s_cc(:,:,448) =

  -0.0671 + 0.0887i  -0.0148 + 0.0148i
  -0.0148 + 0.0148i  -0.0776 - 0.1296i


s_cc(:,:,449) =

  -0.0172 + 0.1211i  -0.0094 + 0.0193i
  -0.0094 + 0.0192i  -0.0733 - 0.1022i


s_cc(:,:,450) =

   0.0404 + 0.1363i  -0.0022 + 0.0223i
  -0.0022 + 0.0223i  -0.0623 - 0.0839i


s_cc(:,:,451) =
```

```
   0.0987 + 0.1334i   0.0065 + 0.0226i
   0.0065 + 0.0225i  -0.0517 - 0.0752i


s_cc(:,:,452) =

   0.1517 + 0.1139i   0.0154 + 0.0191i
   0.0154 + 0.0191i  -0.0467 - 0.0708i


s_cc(:,:,453) =

   0.1951 + 0.0815i   0.0226 + 0.0118i
   0.0226 + 0.0119i  -0.0472 - 0.0647i


s_cc(:,:,454) =

   0.2254 + 0.0412i   0.0263 + 0.0016i
   0.0262 + 0.0015i  -0.0486 - 0.0531i


s_cc(:,:,455) =

   0.2430 - 0.0018i   0.0249 - 0.0097i
   0.0249 - 0.0098i  -0.0458 - 0.0370i


s_cc(:,:,456) =

   0.2507 - 0.0434i   0.0184 - 0.0193i
   0.0185 - 0.0194i  -0.0361 - 0.0201i


s_cc(:,:,457) =

   0.2519 - 0.0828i   0.0087 - 0.0249i
   0.0086 - 0.0249i  -0.0193 - 0.0063i


s_cc(:,:,458) =

   0.2471 - 0.1215i  -0.0023 - 0.0255i
```

```
  -0.0023 - 0.0255i   0.0023 + 0.0013i


s_cc(:,:,459) =

   0.2362 - 0.1609i  -0.0117 - 0.0216i
  -0.0118 - 0.0215i   0.0264 + 0.0006i


s_cc(:,:,460) =

   0.2163 - 0.1986i  -0.0182 - 0.0149i
  -0.0182 - 0.0149i   0.0492 - 0.0091i


s_cc(:,:,461) =

   0.1868 - 0.2326i  -0.0216 - 0.0071i
  -0.0216 - 0.0071i   0.0669 - 0.0273i


s_cc(:,:,462) =

   0.1494 - 0.2583i  -0.0224 + 0.0007i
  -0.0223 + 0.0007i   0.0754 - 0.0508i


s_cc(:,:,463) =

   0.1069 - 0.2744i  -0.0210 + 0.0082i
  -0.0210 + 0.0083i   0.0730 - 0.0744i


s_cc(:,:,464) =

   0.0619 - 0.2797i  -0.0174 + 0.0155i
  -0.0174 + 0.0154i   0.0619 - 0.0923i


s_cc(:,:,465) =

   0.0164 - 0.2744i  -0.0114 + 0.0216i
  -0.0114 + 0.0217i   0.0468 - 0.1011i
```

```
s_cc(:,:,466) =

  -0.0266 - 0.2576i  -0.0026 + 0.0258i
  -0.0026 + 0.0257i   0.0334 - 0.1013i


s_cc(:,:,467) =

  -0.0640 - 0.2293i   0.0081 + 0.0262i
   0.0081 + 0.0262i   0.0263 - 0.0964i


s_cc(:,:,468) =

  -0.0917 - 0.1919i   0.0189 + 0.0217i
   0.0189 + 0.0218i   0.0261 - 0.0924i


s_cc(:,:,469) =

  -0.1066 - 0.1491i   0.0273 + 0.0126i
   0.0274 + 0.0127i   0.0294 - 0.0936i


s_cc(:,:,470) =

  -0.1083 - 0.1059i   0.0313 + 0.0003i
   0.0312 + 0.0003i   0.0308 - 0.1009i


s_cc(:,:,471) =

  -0.0987 - 0.0672i   0.0296 - 0.0133i
   0.0296 - 0.0132i   0.0259 - 0.1110i


s_cc(:,:,472) =

  -0.0817 - 0.0352i   0.0221 - 0.0250i
   0.0222 - 0.0250i   0.0140 - 0.1189i


s_cc(:,:,473) =
```

```
  -0.0604 - 0.0096i   0.0100 - 0.0329i
   0.0100 - 0.0329i  -0.0022 - 0.1208i


s_cc(:,:,474) =

  -0.0360 + 0.0107i  -0.0049 - 0.0352i
  -0.0050 - 0.0352i  -0.0187 - 0.1148i


s_cc(:,:,475) =

  -0.0087 + 0.0254i  -0.0201 - 0.0308i
  -0.0200 - 0.0307i  -0.0318 - 0.1027i


s_cc(:,:,476) =

   0.0206 + 0.0337i  -0.0323 - 0.0196i
  -0.0323 - 0.0196i  -0.0391 - 0.0872i


s_cc(:,:,477) =

   0.0502 + 0.0358i  -0.0384 - 0.0033i
  -0.0384 - 0.0034i  -0.0408 - 0.0712i


s_cc(:,:,478) =

   0.0792 + 0.0321i  -0.0361 + 0.0142i
  -0.0362 + 0.0142i  -0.0377 - 0.0573i


s_cc(:,:,479) =

   0.1071 + 0.0230i  -0.0258 + 0.0285i
  -0.0258 + 0.0285i  -0.0318 - 0.0462i


s_cc(:,:,480) =

   0.1338 + 0.0078i  -0.0105 + 0.0363i
```

```
  -0.0105 + 0.0363i  -0.0246 - 0.0382i


s_cc(:,:,481) =

   0.1581 - 0.0141i   0.0059 + 0.0365i
   0.0059 + 0.0365i  -0.0173 - 0.0330i


s_cc(:,:,482) =

   0.1768 - 0.0434i   0.0202 + 0.0298i
   0.0204 + 0.0297i  -0.0108 - 0.0291i


s_cc(:,:,483) =

   0.1867 - 0.0779i   0.0300 + 0.0182i
   0.0300 + 0.0181i  -0.0053 - 0.0255i


s_cc(:,:,484) =

   0.1860 - 0.1142i   0.0339 + 0.0041i
   0.0338 + 0.0040i   0.0006 - 0.0210i


s_cc(:,:,485) =

   0.1744 - 0.1481i   0.0317 - 0.0095i
   0.0317 - 0.0096i   0.0083 - 0.0158i


s_cc(:,:,486) =

   0.1546 - 0.1750i   0.0248 - 0.0206i
   0.0247 - 0.0208i   0.0188 - 0.0111i


s_cc(:,:,487) =

   0.1303 - 0.1939i   0.0146 - 0.0280i
   0.0144 - 0.0281i   0.0323 - 0.0084i
```

```
s_cc(:,:,488) =

   0.1058 - 0.2049i   0.0028 - 0.0310i
   0.0028 - 0.0310i   0.0480 - 0.0097i


s_cc(:,:,489) =

   0.0812 - 0.2104i  -0.0090 - 0.0298i
  -0.0091 - 0.0298i   0.0635 - 0.0156i


s_cc(:,:,490) =

   0.0569 - 0.2110i  -0.0198 - 0.0246i
  -0.0200 - 0.0245i   0.0774 - 0.0263i


s_cc(:,:,491) =

   0.0337 - 0.2057i  -0.0285 - 0.0154i
  -0.0284 - 0.0154i   0.0872 - 0.0403i


s_cc(:,:,492) =

   0.0132 - 0.1939i  -0.0332 - 0.0030i
  -0.0332 - 0.0030i   0.0923 - 0.0560i


s_cc(:,:,493) =

  -0.0019 - 0.1774i  -0.0327 + 0.0110i
  -0.0325 + 0.0110i   0.0927 - 0.0713i


s_cc(:,:,494) =

  -0.0103 - 0.1584i  -0.0261 + 0.0243i
  -0.0260 + 0.0242i   0.0894 - 0.0846i


s_cc(:,:,495) =
```

```
  -0.0116 - 0.1392i  -0.0138 + 0.0339i
  -0.0138 + 0.0338i   0.0839 - 0.0956i


s_cc(:,:,496) =

  -0.0069 - 0.1226i   0.0020 + 0.0373i
   0.0020 + 0.0374i   0.0776 - 0.1046i


s_cc(:,:,497) =

   0.0030 - 0.1105i   0.0181 + 0.0332i
   0.0180 + 0.0333i   0.0705 - 0.1120i


s_cc(:,:,498) =

   0.0149 - 0.1044i   0.0308 + 0.0225i
   0.0308 + 0.0224i   0.0628 - 0.1184i


s_cc(:,:,499) =

   0.0264 - 0.1042i   0.0374 + 0.0071i
   0.0373 + 0.0071i   0.0553 - 0.1236i


s_cc(:,:,500) =

   0.0353 - 0.1087i   0.0367 - 0.0095i
   0.0365 - 0.0094i   0.0476 - 0.1281i


s_cc(:,:,501) =

   0.0406 - 0.1161i   0.0290 - 0.0237i
   0.0290 - 0.0236i   0.0395 - 0.1330i


s_cc(:,:,502) =

   0.0413 - 0.1246i   0.0165 - 0.0333i
```

```
   0.0165 - 0.0332i   0.0298 - 0.1376i


s_cc(:,:,503) =

   0.0381 - 0.1322i   0.0013 - 0.0369i
   0.0014 - 0.0368i   0.0181 - 0.1405i


s_cc(:,:,504) =

   0.0317 - 0.1370i  -0.0136 - 0.0340i
  -0.0136 - 0.0341i   0.0047 - 0.1406i


s_cc(:,:,505) =

   0.0241 - 0.1388i  -0.0264 - 0.0256i
  -0.0264 - 0.0257i  -0.0093 - 0.1366i


s_cc(:,:,506) =

   0.0167 - 0.1373i  -0.0349 - 0.0128i
  -0.0349 - 0.0128i  -0.0222 - 0.1275i


s_cc(:,:,507) =

   0.0100 - 0.1330i  -0.0375 + 0.0026i
  -0.0375 + 0.0026i  -0.0321 - 0.1145i


s_cc(:,:,508) =

   0.0055 - 0.1261i  -0.0335 + 0.0183i
  -0.0335 + 0.0183i  -0.0373 - 0.0984i


s_cc(:,:,509) =

   0.0040 - 0.1172i  -0.0229 + 0.0312i
  -0.0229 + 0.0313i  -0.0375 - 0.0814i
```

```
s_cc(:,:,510) =

   0.0063 - 0.1082i  -0.0075 + 0.0384i
  -0.0075 + 0.0383i  -0.0331 - 0.0652i


s_cc(:,:,511) =

   0.0122 - 0.1006i   0.0098 + 0.0381i
   0.0097 + 0.0381i  -0.0248 - 0.0510i


s_cc(:,:,512) =

   0.0201 - 0.0950i   0.0253 + 0.0302i
   0.0253 + 0.0302i  -0.0133 - 0.0390i


s_cc(:,:,513) =

   0.0299 - 0.0916i   0.0358 + 0.0163i
   0.0359 + 0.0163i   0.0004 - 0.0312i


s_cc(:,:,514) =

   0.0412 - 0.0909i   0.0392 - 0.0009i
   0.0391 - 0.0009i   0.0151 - 0.0271i


s_cc(:,:,515) =

   0.0532 - 0.0935i   0.0346 - 0.0176i
   0.0345 - 0.0177i   0.0293 - 0.0265i


s_cc(:,:,516) =

   0.0653 - 0.0994i   0.0231 - 0.0305i
   0.0230 - 0.0305i   0.0427 - 0.0288i


s_cc(:,:,517) =
```

```
   0.0767 - 0.1091i   0.0074 - 0.0365i
   0.0073 - 0.0365i   0.0553 - 0.0331i


s_cc(:,:,518) =

   0.0862 - 0.1228i  -0.0087 - 0.0353i
  -0.0088 - 0.0353i   0.0674 - 0.0389i


s_cc(:,:,519) =

   0.0925 - 0.1403i  -0.0223 - 0.0274i
  -0.0221 - 0.0274i   0.0795 - 0.0473i


s_cc(:,:,520) =

   0.0941 - 0.1602i  -0.0307 - 0.0149i
  -0.0307 - 0.0149i   0.0910 - 0.0585i


s_cc(:,:,521) =

   0.0908 - 0.1808i  -0.0330 - 0.0008i
  -0.0330 - 0.0007i   0.1012 - 0.0728i


s_cc(:,:,522) =

   0.0828 - 0.2022i  -0.0294 + 0.0125i
  -0.0292 + 0.0125i   0.1089 - 0.0901i


s_cc(:,:,523) =

   0.0693 - 0.2223i  -0.0209 + 0.0227i
  -0.0209 + 0.0226i   0.1124 - 0.1092i


s_cc(:,:,524) =

   0.0508 - 0.2405i  -0.0095 + 0.0281i
```

```
  -0.0095 + 0.0280i   0.1114 - 0.1292i


s_cc(:,:,525) =

   0.0260 - 0.2549i   0.0024 + 0.0286i
   0.0025 + 0.0284i   0.1061 - 0.1475i


s_cc(:,:,526) =

  -0.0038 - 0.2630i   0.0130 + 0.0245i
   0.0130 + 0.0244i   0.0976 - 0.1632i


s_cc(:,:,527) =

  -0.0358 - 0.2620i   0.0207 + 0.0169i
   0.0207 + 0.0168i   0.0873 - 0.1754i


s_cc(:,:,528) =

  -0.0669 - 0.2519i   0.0249 + 0.0075i
   0.0249 + 0.0075i   0.0773 - 0.1855i


s_cc(:,:,529) =

  -0.0939 - 0.2332i   0.0254 - 0.0024i
   0.0254 - 0.0023i   0.0680 - 0.1957i


s_cc(:,:,530) =

  -0.1142 - 0.2080i   0.0224 - 0.0117i
   0.0223 - 0.0117i   0.0566 - 0.2070i


s_cc(:,:,531) =

  -0.1265 - 0.1786i   0.0163 - 0.0192i
   0.0163 - 0.0191i   0.0418 - 0.2185i
```

```
s_cc(:,:,532) =

  -0.1297 - 0.1484i   0.0078 - 0.0240i
   0.0079 - 0.0239i   0.0222 - 0.2278i


s_cc(:,:,533) =

  -0.1249 - 0.1200i  -0.0019 - 0.0255i
  -0.0018 - 0.0254i  -0.0013 - 0.2323i


s_cc(:,:,534) =

  -0.1133 - 0.0959i  -0.0119 - 0.0231i
  -0.0118 - 0.0232i  -0.0265 - 0.2309i


s_cc(:,:,535) =

  -0.0975 - 0.0775i  -0.0203 - 0.0170i
  -0.0204 - 0.0171i  -0.0506 - 0.2231i


s_cc(:,:,536) =

  -0.0799 - 0.0649i  -0.0260 - 0.0077i
  -0.0260 - 0.0077i  -0.0715 - 0.2094i


s_cc(:,:,537) =

  -0.0624 - 0.0568i  -0.0276 + 0.0036i
  -0.0276 + 0.0036i  -0.0878 - 0.1914i


s_cc(:,:,538) =

  -0.0457 - 0.0521i  -0.0242 + 0.0149i
  -0.0242 + 0.0149i  -0.0979 - 0.1714i


s_cc(:,:,539) =
```

```
  -0.0288 - 0.0493i  -0.0161 + 0.0242i
  -0.0161 + 0.0242i  -0.1024 - 0.1515i


s_cc(:,:,540) =

  -0.0112 - 0.0495i  -0.0045 + 0.0292i
  -0.0045 + 0.0292i  -0.1020 - 0.1331i


s_cc(:,:,541) =

   0.0070 - 0.0534i   0.0085 + 0.0287i
   0.0085 + 0.0286i  -0.0986 - 0.1171i


s_cc(:,:,542) =

   0.0244 - 0.0621i   0.0202 + 0.0225i
   0.0201 + 0.0224i  -0.0930 - 0.1036i


s_cc(:,:,543) =

   0.0395 - 0.0750i   0.0278 + 0.0116i
   0.0279 + 0.0116i  -0.0869 - 0.0916i


s_cc(:,:,544) =

   0.0513 - 0.0919i   0.0299 - 0.0016i
   0.0299 - 0.0017i  -0.0796 - 0.0803i


s_cc(:,:,545) =

   0.0591 - 0.1112i   0.0258 - 0.0142i
   0.0258 - 0.0144i  -0.0704 - 0.0695i


s_cc(:,:,546) =

   0.0617 - 0.1322i   0.0168 - 0.0234i
```

```
   0.0168 - 0.0234i  -0.0586 - 0.0596i


s_cc(:,:,547) =

   0.0597 - 0.1535i   0.0051 - 0.0276i
   0.0051 - 0.0275i  -0.0437 - 0.0524i


s_cc(:,:,548) =

   0.0526 - 0.1738i  -0.0066 - 0.0261i
  -0.0066 - 0.0261i  -0.0267 - 0.0493i


s_cc(:,:,549) =

   0.0414 - 0.1920i  -0.0160 - 0.0203i
  -0.0160 - 0.0204i  -0.0088 - 0.0514i


s_cc(:,:,550) =

   0.0262 - 0.2075i  -0.0219 - 0.0119i
  -0.0218 - 0.0118i   0.0081 - 0.0590i


s_cc(:,:,551) =

   0.0080 - 0.2193i  -0.0239 - 0.0023i
  -0.0239 - 0.0023i   0.0221 - 0.0719i


s_cc(:,:,552) =

  -0.0120 - 0.2272i  -0.0225 + 0.0067i
  -0.0224 + 0.0068i   0.0312 - 0.0883i


s_cc(:,:,553) =

  -0.0330 - 0.2309i  -0.0180 + 0.0145i
  -0.0180 + 0.0144i   0.0346 - 0.1067i
```

```
s_cc(:,:,554) =

  -0.0550 - 0.2302i  -0.0112 + 0.0202i
  -0.0112 + 0.0200i   0.0325 - 0.1241i


s_cc(:,:,555) =

  -0.0772 - 0.2250i  -0.0028 + 0.0229i
  -0.0028 + 0.0228i   0.0262 - 0.1390i


s_cc(:,:,556) =

  -0.0985 - 0.2145i   0.0064 + 0.0224i
   0.0064 + 0.0224i   0.0180 - 0.1508i


s_cc(:,:,557) =

  -0.1173 - 0.1987i   0.0147 + 0.0185i
   0.0146 + 0.0183i   0.0093 - 0.1601i


s_cc(:,:,558) =

  -0.1316 - 0.1781i   0.0211 + 0.0113i
   0.0210 + 0.0113i   0.0007 - 0.1684i


s_cc(:,:,559) =

  -0.1399 - 0.1543i   0.0242 + 0.0020i
   0.0243 + 0.0020i  -0.0091 - 0.1770i


s_cc(:,:,560) =

  -0.1416 - 0.1295i   0.0232 - 0.0081i
   0.0233 - 0.0081i  -0.0220 - 0.1854i


s_cc(:,:,561) =
```

```
  -0.1365 - 0.1059i   0.0182 - 0.0170i
   0.0183 - 0.0170i  -0.0386 - 0.1917i


s_cc(:,:,562) =

  -0.1255 - 0.0853i   0.0098 - 0.0232i
   0.0099 - 0.0232i  -0.0584 - 0.1937i


s_cc(:,:,563) =

  -0.1097 - 0.0694i  -0.0005 - 0.0255i
  -0.0003 - 0.0255i  -0.0799 - 0.1899i


s_cc(:,:,564) =

  -0.0913 - 0.0596i  -0.0108 - 0.0234i
  -0.0107 - 0.0234i  -0.1003 - 0.1793i


s_cc(:,:,565) =

  -0.0720 - 0.0558i  -0.0198 - 0.0171i
  -0.0198 - 0.0172i  -0.1175 - 0.1631i


s_cc(:,:,566) =

  -0.0539 - 0.0572i  -0.0254 - 0.0076i
  -0.0255 - 0.0075i  -0.1293 - 0.1427i


s_cc(:,:,567) =

  -0.0383 - 0.0635i  -0.0266 + 0.0038i
  -0.0267 + 0.0039i  -0.1342 - 0.1198i


s_cc(:,:,568) =

  -0.0262 - 0.0723i  -0.0226 + 0.0148i
```

```
  -0.0225 + 0.0149i  -0.1328 - 0.0975i


s_cc(:,:,569) =

  -0.0170 - 0.0823i  -0.0141 + 0.0231i
  -0.0141 + 0.0230i  -0.1257 - 0.0777i


s_cc(:,:,570) =

  -0.0099 - 0.0927i  -0.0030 + 0.0267i
  -0.0028 + 0.0267i  -0.1146 - 0.0617i


s_cc(:,:,571) =

  -0.0037 - 0.1041i   0.0085 + 0.0251i
   0.0086 + 0.0249i  -0.1009 - 0.0498i


s_cc(:,:,572) =

   0.0009 - 0.1173i   0.0176 + 0.0188i
   0.0177 + 0.0186i  -0.0859 - 0.0416i


s_cc(:,:,573) =

   0.0032 - 0.1328i   0.0231 + 0.0098i
   0.0231 + 0.0097i  -0.0710 - 0.0366i


s_cc(:,:,574) =

   0.0013 - 0.1502i   0.0246 - 0.0000i
   0.0246 - 0.0001i  -0.0563 - 0.0334i


s_cc(:,:,575) =

  -0.0061 - 0.1675i   0.0223 - 0.0096i
   0.0222 - 0.0096i  -0.0410 - 0.0320i
```

```
s_cc(:,:,576) =

  -0.0191 - 0.1828i   0.0168 - 0.0172i
   0.0167 - 0.0173i  -0.0242 - 0.0327i


s_cc(:,:,577) =

  -0.0365 - 0.1941i   0.0088 - 0.0224i
   0.0087 - 0.0223i  -0.0061 - 0.0371i


s_cc(:,:,578) =

  -0.0566 - 0.1998i  -0.0007 - 0.0242i
  -0.0007 - 0.0242i   0.0120 - 0.0468i


s_cc(:,:,579) =

  -0.0774 - 0.1999i  -0.0101 - 0.0221i
  -0.0101 - 0.0221i   0.0277 - 0.0622i


s_cc(:,:,580) =

  -0.0972 - 0.1949i  -0.0183 - 0.0166i
  -0.0182 - 0.0165i   0.0391 - 0.0825i


s_cc(:,:,581) =

  -0.1150 - 0.1854i  -0.0236 - 0.0082i
  -0.0236 - 0.0081i   0.0444 - 0.1061i


s_cc(:,:,582) =

  -0.1297 - 0.1725i  -0.0254 + 0.0019i
  -0.0255 + 0.0019i   0.0429 - 0.1308i


s_cc(:,:,583) =
```

```
  -0.1410 - 0.1571i  -0.0229 + 0.0123i
  -0.0230 + 0.0123i   0.0352 - 0.1545i


s_cc(:,:,584) =

  -0.1481 - 0.1401i  -0.0163 + 0.0211i
  -0.0164 + 0.0211i   0.0220 - 0.1753i


s_cc(:,:,585) =

  -0.1515 - 0.1228i  -0.0064 + 0.0264i
  -0.0064 + 0.0264i   0.0049 - 0.1923i


s_cc(:,:,586) =

  -0.1511 - 0.1058i   0.0053 + 0.0272i
   0.0052 + 0.0272i  -0.0146 - 0.2051i


s_cc(:,:,587) =

  -0.1476 - 0.0895i   0.0164 + 0.0230i
   0.0164 + 0.0230i  -0.0358 - 0.2140i


s_cc(:,:,588) =

  -0.1405 - 0.0746i   0.0251 + 0.0142i
   0.0249 + 0.0141i  -0.0572 - 0.2190i


s_cc(:,:,589) =

  -0.1305 - 0.0614i   0.0288 + 0.0025i
   0.0290 + 0.0024i  -0.0793 - 0.2209i


s_cc(:,:,590) =

  -0.1179 - 0.0507i   0.0276 - 0.0101i
```

```
   0.0276 - 0.0099i  -0.1014 - 0.2190i


s_cc(:,:,591) =

  -0.1030 - 0.0429i   0.0210 - 0.0210i
   0.0211 - 0.0210i  -0.1240 - 0.2124i


s_cc(:,:,592) =

  -0.0869 - 0.0392i   0.0103 - 0.0280i
   0.0104 - 0.0281i  -0.1452 - 0.2013i


s_cc(:,:,593) =

  -0.0700 - 0.0388i  -0.0025 - 0.0300i
  -0.0025 - 0.0301i  -0.1643 - 0.1857i


s_cc(:,:,594) =

  -0.0534 - 0.0425i  -0.0149 - 0.0264i
  -0.0150 - 0.0265i  -0.1802 - 0.1658i


s_cc(:,:,595) =

  -0.0384 - 0.0504i  -0.0248 - 0.0177i
  -0.0249 - 0.0177i  -0.1920 - 0.1426i


s_cc(:,:,596) =

  -0.0259 - 0.0621i  -0.0301 - 0.0056i
  -0.0302 - 0.0057i  -0.1991 - 0.1178i


s_cc(:,:,597) =

  -0.0172 - 0.0765i  -0.0297 + 0.0076i
  -0.0298 + 0.0077i  -0.2008 - 0.0923i
```

```
s_cc(:,:,598) =

  -0.0134 - 0.0920i  -0.0238 + 0.0195i
  -0.0237 + 0.0196i  -0.1982 - 0.0668i


s_cc(:,:,599) =

  -0.0137 - 0.1071i  -0.0134 + 0.0276i
  -0.0133 + 0.0277i  -0.1909 - 0.0429i


s_cc(:,:,600) =

  -0.0169 - 0.1203i  -0.0004 + 0.0306i
  -0.0005 + 0.0306i  -0.1791 - 0.0213i


s_cc(:,:,601) =

  -0.0222 - 0.1321i   0.0124 + 0.0280i
   0.0125 + 0.0279i  -0.1637 - 0.0024i


s_cc(:,:,602) =

  -0.0285 - 0.1424i   0.0228 + 0.0200i
   0.0229 + 0.0199i  -0.1443 + 0.0122i


s_cc(:,:,603) =

  -0.0364 - 0.1522i   0.0290 + 0.0086i
   0.0290 + 0.0085i  -0.1228 + 0.0222i


s_cc(:,:,604) =

  -0.0457 - 0.1606i   0.0297 - 0.0041i
   0.0297 - 0.0042i  -0.0996 + 0.0275i


s_cc(:,:,605) =
```

```
  -0.0566 - 0.1680i   0.0252 - 0.0160i
   0.0252 - 0.0161i  -0.0758 + 0.0279i


s_cc(:,:,606) =

  -0.0688 - 0.1735i   0.0163 - 0.0248i
   0.0162 - 0.0249i  -0.0522 + 0.0233i


s_cc(:,:,607) =

  -0.0825 - 0.1771i   0.0046 - 0.0292i
   0.0045 - 0.0292i  -0.0299 + 0.0140i


s_cc(:,:,608) =

  -0.0971 - 0.1774i  -0.0077 - 0.0285i
  -0.0078 - 0.0284i  -0.0099 + 0.0002i


s_cc(:,:,609) =

  -0.1120 - 0.1747i  -0.0187 - 0.0228i
  -0.0187 - 0.0228i   0.0066 - 0.0172i


s_cc(:,:,610) =

  -0.1261 - 0.1684i  -0.0265 - 0.0133i
  -0.0263 - 0.0132i   0.0190 - 0.0373i


s_cc(:,:,611) =

  -0.1383 - 0.1583i  -0.0296 - 0.0013i
  -0.0296 - 0.0012i   0.0267 - 0.0594i


s_cc(:,:,612) =

  -0.1471 - 0.1452i  -0.0278 + 0.0111i
```

```
  -0.0278 + 0.0112i   0.0290 - 0.0826i


s_cc(:,:,613) =

  -0.1517 - 0.1306i  -0.0209 + 0.0218i
  -0.0209 + 0.0218i   0.0259 - 0.1052i


s_cc(:,:,614) =

  -0.1520 - 0.1168i  -0.0100 + 0.0289i
  -0.0101 + 0.0288i   0.0173 - 0.1264i


s_cc(:,:,615) =

  -0.1489 - 0.1046i   0.0030 + 0.0308i
   0.0029 + 0.0308i   0.0038 - 0.1448i


s_cc(:,:,616) =

  -0.1439 - 0.0953i   0.0158 + 0.0268i
   0.0157 + 0.0270i  -0.0134 - 0.1596i


s_cc(:,:,617) =

  -0.1382 - 0.0888i   0.0260 + 0.0178i
   0.0259 + 0.0179i  -0.0335 - 0.1698i


s_cc(:,:,618) =

  -0.1333 - 0.0841i   0.0313 + 0.0053i
   0.0313 + 0.0052i  -0.0548 - 0.1753i


s_cc(:,:,619) =

  -0.1288 - 0.0804i   0.0307 - 0.0087i
   0.0308 - 0.0087i  -0.0766 - 0.1761i
```

```
s_cc(:,:,620) =

  -0.1244 - 0.0770i   0.0243 - 0.0211i
```

References

Fan, W., A. C. W. Lu, L. L. Wai, and B. K. Lok. “Mixed-Mode S-Parameter Characterization of Differential Structures.” Electronic Packaging Technology Conference, pp. 533–537, 2003.

See Also

`s2scd` | `s2sdc` | `s2sdd` | `s2smm` | `smm2s`

Introduced in R2006a

s2scd

Convert 4-port, single-ended S-parameters to 2-port, cross-mode S-parameters (S_{cd})

Syntax

```
scd_params = s2scd(s_params)
scd_params = s2scd(s_params,option)
```

Description

`scd_params = s2scd(s_params)` converts the 2*N*-port, single-ended S-parameters, `s_params`, to *N*-port, cross-mode S-parameters, `scd_params`. `scd_params` is a complex *N*-by-*N*-by-*M* array that represents *M* *N*-port, cross-mode S-parameters (S_{cd}).

`scd_params = s2scd(s_params,option)` converts S-parameters based on the optional `option` argument, which indicates the port-ordering convention of the S-parameters.

Input Arguments

`s_params` — S-parameters
array

S-parameters, specified as a complex 4-by-4-by-*M* array, that represents *M* 4-port S-parameters.

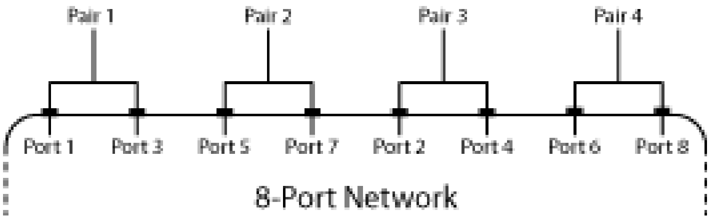
`option` — Port order
1 (default) | 2 | 3

Port order, specified as `1`, `2`, `3`, determines how the function orders the ports:

- `1` — `s2scd` pairs the odd-numbered ports together first, followed by the even-numbered ports. For example, in a single-ended, 8-port network:

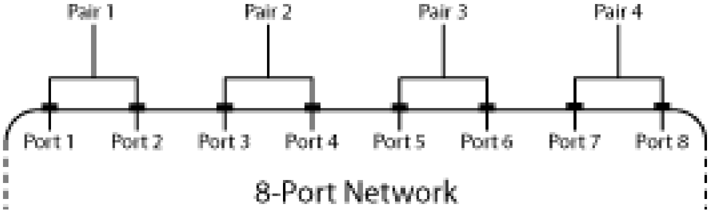
- Ports 1 and 3 become cross-mode pair 1.
- Ports 5 and 7 become cross-mode pair 2.
- Ports 2 and 4 become cross-mode pair 3.
- Ports 6 and 8 become cross-mode pair 4.

The following figure illustrates this convention for an 8-port device.



- `2` — `s2scd` pairs the input and output ports in ascending order. For example, in a single-ended, 8-port network:
  - Ports 1 and 2 become cross-mode pair 1.
  - Ports 3 and 4 become cross-mode pair 2.
  - Ports 5 and 6 become cross-mode pair 3.
  - Ports 7 and 8 become cross-mode pair 4.

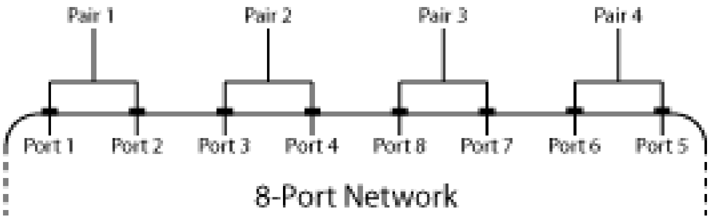
  The following figure illustrates this convention for an 8-port device.



- `3` — `s2scd` pairs the input ports in ascending order and the output ports in descending order. For example, in a single-ended, 8-port network:
  - Ports 1 and 2 become cross-mode pair 1.
  - Ports 3 and 4 become cross-mode pair 2.
  - Ports 8 and 7 become cross-mode pair 3.

- Ports 6 and 5 become cross-mode pair 4.

The following figure illustrates this convention for an 8-port device.



Examples

Network Data to Cross-Mode S-Parameters

Convert network data to cross-mode S-parameters using the default port ordering.

```
ckt = read(rfckt.passive,'default.s4p');
s4p = ckt.NetworkData.Data;
s_cd = s2scd(s4p)
```

```
s_cd =
s_cd(:,:,1) =

   0.0015 - 0.0029i  -0.0005 + 0.0014i
   0.0003 - 0.0009i   0.0019 - 0.0027i


s_cd(:,:,2) =

   0.0030 - 0.0019i   0.0006 - 0.0008i
   0.0003 - 0.0006i   0.0011 - 0.0042i


s_cd(:,:,3) =

   0.0007 - 0.0039i   0.0005 - 0.0001i
  -0.0003 - 0.0009i  -0.0010 - 0.0043i
```

```
s_cd(:,:,4) =

  -0.0003 - 0.0052i  -0.0004 - 0.0023i
  -0.0003 - 0.0006i  -0.0028 - 0.0030i


s_cd(:,:,5) =

  -0.0020 - 0.0045i  -0.0011 - 0.0007i
  -0.0003 + 0.0002i  -0.0038 - 0.0014i


s_cd(:,:,6) =

  -0.0018 - 0.0034i  -0.0013 - 0.0013i
  -0.0003 - 0.0007i  -0.0027 + 0.0005i


s_cd(:,:,7) =

  -0.0021 - 0.0019i  -0.0017 + 0.0002i
  -0.0010 + 0.0002i  -0.0014 + 0.0012i


s_cd(:,:,8) =

  -0.0011 - 0.0015i  -0.0013 + 0.0003i
  -0.0008 + 0.0010i  -0.0002 + 0.0004i


s_cd(:,:,9) =

  -0.0000 - 0.0024i  -0.0004 + 0.0003i
  -0.0007 + 0.0013i   0.0006 - 0.0011i


s_cd(:,:,10) =

   0.0005 - 0.0037i  -0.0003 + 0.0008i
  -0.0005 + 0.0014i  -0.0011 - 0.0024i


s_cd(:,:,11) =
```

```
    0.0002 - 0.0045i  -0.0001 + 0.0018i
    0.0003 + 0.0017i  -0.0025 - 0.0023i


s_cd(:,:,12) =

  -0.0010 - 0.0055i   0.0014 + 0.0012i
   0.0007 + 0.0009i  -0.0047 - 0.0005i


s_cd(:,:,13) =

  -0.0025 - 0.0064i   0.0012 + 0.0009i
   0.0013 + 0.0001i  -0.0050 + 0.0022i


s_cd(:,:,14) =

  -0.0046 - 0.0062i   0.0034 - 0.0004i
   0.0013 + 0.0001i  -0.0033 + 0.0052i


s_cd(:,:,15) =

  -0.0073 - 0.0053i   0.0025 - 0.0010i
   0.0007 - 0.0007i   0.0001 + 0.0064i


s_cd(:,:,16) =

  -0.0087 - 0.0036i   0.0025 - 0.0023i
  -0.0002 - 0.0007i   0.0039 + 0.0056i


s_cd(:,:,17) =

  -0.0098 - 0.0009i   0.0003 - 0.0031i
   0.0005 - 0.0017i   0.0066 + 0.0026i


s_cd(:,:,18) =

  -0.0102 + 0.0018i  -0.0009 - 0.0033i
   0.0004 - 0.0018i   0.0069 - 0.0016i
```

```
s_cd(:,:,19) =

  -0.0091 + 0.0042i  -0.0022 - 0.0021i
  -0.0009 - 0.0009i   0.0053 - 0.0053i


s_cd(:,:,20) =

  -0.0077 + 0.0063i  -0.0031 - 0.0002i
  -0.0006 - 0.0005i   0.0015 - 0.0078i


s_cd(:,:,21) =

  -0.0058 + 0.0076i  -0.0025 + 0.0007i
  -0.0014 + 0.0001i  -0.0027 - 0.0074i


s_cd(:,:,22) =

  -0.0037 + 0.0082i  -0.0015 + 0.0016i
  -0.0014 - 0.0001i  -0.0058 - 0.0047i


s_cd(:,:,23) =

  -0.0014 + 0.0081i  -0.0008 + 0.0015i
  -0.0011 + 0.0010i  -0.0067 - 0.0011i


s_cd(:,:,24) =

   0.0002 + 0.0073i  -0.0004 + 0.0018i
  -0.0008 + 0.0012i  -0.0053 + 0.0020i


s_cd(:,:,25) =

   0.0013 + 0.0062i   0.0007 + 0.0019i
   0.0004 + 0.0016i  -0.0023 + 0.0032i
```

```
s_cd(:,:,26) =

   0.0016 + 0.0056i   0.0017 + 0.0016i
   0.0009 + 0.0015i   0.0007 + 0.0022i


s_cd(:,:,27) =

   0.0020 + 0.0057i   0.0023 + 0.0019i
   0.0013 + 0.0008i   0.0022 - 0.0005i


s_cd(:,:,28) =

   0.0028 + 0.0059i   0.0032 + 0.0008i
   0.0015 + 0.0001i   0.0016 - 0.0034i


s_cd(:,:,29) =

   0.0050 + 0.0061i   0.0035 - 0.0012i
   0.0011 - 0.0003i  -0.0007 - 0.0053i


s_cd(:,:,30) =

   0.0076 + 0.0047i   0.0034 - 0.0022i
   0.0009 - 0.0004i  -0.0038 - 0.0058i


s_cd(:,:,31) =

   0.0096 + 0.0023i   0.0017 - 0.0033i
   0.0012 - 0.0003i  -0.0069 - 0.0043i


s_cd(:,:,32) =

   0.0099 - 0.0006i  -0.0006 - 0.0041i
   0.0014 - 0.0016i  -0.0085 - 0.0011i


s_cd(:,:,33) =
```

```
   0.0101 - 0.0035i  -0.0023 - 0.0031i
   0.0006 - 0.0023i  -0.0086 + 0.0022i


s_cd(:,:,34) =

   0.0087 - 0.0063i  -0.0035 - 0.0020i
  -0.0005 - 0.0020i  -0.0064 + 0.0056i


s_cd(:,:,35) =

   0.0066 - 0.0075i  -0.0040 - 0.0000i
  -0.0018 - 0.0017i  -0.0026 + 0.0069i


s_cd(:,:,36) =

   0.0047 - 0.0080i  -0.0029 + 0.0010i
  -0.0026 - 0.0003i   0.0011 + 0.0053i


s_cd(:,:,37) =

   0.0038 - 0.0077i  -0.0021 + 0.0023i
  -0.0020 + 0.0008i   0.0029 + 0.0015i


s_cd(:,:,38) =

   0.0036 - 0.0084i  -0.0011 + 0.0023i
  -0.0014 + 0.0011i   0.0016 - 0.0028i


s_cd(:,:,39) =

   0.0022 - 0.0095i  -0.0012 + 0.0022i
  -0.0014 + 0.0010i  -0.0032 - 0.0048i


s_cd(:,:,40) =

   0.0015 - 0.0093i   0.0002 + 0.0045i
  -0.0009 + 0.0027i  -0.0068 - 0.0024i
```

```
s_cd(:,:,41) =

   0.0003 - 0.0109i   0.0031 + 0.0034i
   0.0008 + 0.0024i  -0.0086 + 0.0005i


s_cd(:,:,42) =

  -0.0028 - 0.0119i   0.0040 + 0.0015i
   0.0011 + 0.0017i  -0.0086 + 0.0045i


s_cd(:,:,43) =

  -0.0064 - 0.0112i   0.0041 - 0.0003i
   0.0014 + 0.0013i  -0.0060 + 0.0080i


s_cd(:,:,44) =

  -0.0098 - 0.0093i   0.0037 - 0.0026i
   0.0022 + 0.0003i  -0.0017 + 0.0099i


s_cd(:,:,45) =

  -0.0126 - 0.0058i   0.0022 - 0.0042i
   0.0022 - 0.0003i   0.0034 + 0.0097i


s_cd(:,:,46) =

  -0.0139 - 0.0010i   0.0002 - 0.0041i
   0.0026 - 0.0014i   0.0083 + 0.0064i


s_cd(:,:,47) =

  -0.0133 + 0.0044i  -0.0017 - 0.0037i
   0.0023 - 0.0027i   0.0107 + 0.0010i
```

```
s_cd(:,:,48) =

  -0.0101 + 0.0095i  -0.0033 - 0.0031i
   0.0005 - 0.0041i   0.0107 - 0.0039i


s_cd(:,:,49) =

  -0.0051 + 0.0128i  -0.0044 - 0.0015i
  -0.0017 - 0.0037i   0.0096 - 0.0093i


s_cd(:,:,50) =

   0.0014 + 0.0143i  -0.0048 + 0.0010i
  -0.0029 - 0.0028i   0.0059 - 0.0144i


s_cd(:,:,51) =

   0.0089 + 0.0132i  -0.0030 + 0.0034i
  -0.0042 - 0.0015i  -0.0005 - 0.0177i


s_cd(:,:,52) =

   0.0157 + 0.0086i  -0.0011 + 0.0047i
  -0.0044 + 0.0008i  -0.0080 - 0.0177i


s_cd(:,:,53) =

   0.0197 + 0.0013i   0.0009 + 0.0037i
  -0.0037 + 0.0029i  -0.0152 - 0.0143i


s_cd(:,:,54) =

   0.0205 - 0.0067i   0.0021 + 0.0024i
  -0.0022 + 0.0042i  -0.0203 - 0.0076i


s_cd(:,:,55) =
```

```
   0.0180 - 0.0139i   0.0023 + 0.0010i
   0.0001 + 0.0043i  -0.0213 + 0.0010i


s_cd(:,:,56) =

   0.0132 - 0.0193i   0.0013 + 0.0004i
   0.0025 + 0.0039i  -0.0180 + 0.0090i


s_cd(:,:,57) =

   0.0077 - 0.0218i   0.0012 + 0.0007i
   0.0040 + 0.0020i  -0.0106 + 0.0142i


s_cd(:,:,58) =

   0.0027 - 0.0222i   0.0017 + 0.0006i
   0.0047 + 0.0004i  -0.0015 + 0.0137i


s_cd(:,:,59) =

  -0.0008 - 0.0213i   0.0025 - 0.0000i
   0.0038 - 0.0017i   0.0051 + 0.0080i


s_cd(:,:,60) =

  -0.0030 - 0.0212i   0.0028 - 0.0016i
   0.0023 - 0.0029i   0.0066 - 0.0005i


s_cd(:,:,61) =

  -0.0058 - 0.0214i   0.0019 - 0.0029i
   0.0012 - 0.0031i   0.0028 - 0.0075i


s_cd(:,:,62) =

  -0.0095 - 0.0211i   0.0004 - 0.0036i
  -0.0001 - 0.0037i  -0.0039 - 0.0111i
```

```
s_cd(:,:,63) =

  -0.0132 - 0.0193i  -0.0018 - 0.0036i
  -0.0018 - 0.0037i  -0.0117 - 0.0097i


s_cd(:,:,64) =

  -0.0162 - 0.0161i  -0.0031 - 0.0028i
  -0.0033 - 0.0015i  -0.0161 - 0.0043i


s_cd(:,:,65) =

  -0.0171 - 0.0120i  -0.0036 - 0.0010i
  -0.0037 - 0.0002i  -0.0164 + 0.0016i


s_cd(:,:,66) =

  -0.0157 - 0.0088i  -0.0040 + 0.0002i
  -0.0028 + 0.0012i  -0.0135 + 0.0051i


s_cd(:,:,67) =

  -0.0125 - 0.0079i  -0.0042 + 0.0013i
  -0.0022 + 0.0018i  -0.0109 + 0.0048i


s_cd(:,:,68) =

  -0.0108 - 0.0098i  -0.0045 + 0.0035i
  -0.0019 + 0.0020i  -0.0122 + 0.0029i


s_cd(:,:,69) =

  -0.0100 - 0.0121i  -0.0025 + 0.0067i
  -0.0010 + 0.0037i  -0.0159 + 0.0043i
```

```
s_cd(:,:,70) =

  -0.0108 - 0.0174i   0.0023 + 0.0076i
   0.0012 + 0.0036i  -0.0197 + 0.0078i


s_cd(:,:,71) =

  -0.0173 - 0.0228i   0.0058 + 0.0050i
   0.0021 + 0.0014i  -0.0226 + 0.0150i


s_cd(:,:,72) =

  -0.0275 - 0.0241i   0.0076 + 0.0002i
   0.0023 + 0.0007i  -0.0217 + 0.0257i


s_cd(:,:,73) =

  -0.0391 - 0.0193i   0.0060 - 0.0032i
   0.0017 + 0.0006i  -0.0139 + 0.0369i


s_cd(:,:,74) =

  -0.0486 - 0.0083i   0.0028 - 0.0050i
   0.0017 + 0.0012i   0.0007 + 0.0431i


s_cd(:,:,75) =

  -0.0523 + 0.0072i   0.0001 - 0.0050i
   0.0036 + 0.0005i   0.0179 + 0.0402i


s_cd(:,:,76) =

  -0.0479 + 0.0233i  -0.0011 - 0.0037i
   0.0045 - 0.0020i   0.0306 + 0.0286i


s_cd(:,:,77) =
```

```
  -0.0370 + 0.0346i  -0.0014 - 0.0035i
   0.0027 - 0.0051i   0.0358 + 0.0133i


s_cd(:,:,78) =

  -0.0247 + 0.0398i  -0.0030 - 0.0020i
  -0.0002 - 0.0060i   0.0335 - 0.0024i


s_cd(:,:,79) =

  -0.0129 + 0.0402i  -0.0037 - 0.0013i
  -0.0036 - 0.0050i   0.0239 - 0.0146i


s_cd(:,:,80) =

  -0.0035 + 0.0369i  -0.0036 - 0.0001i
  -0.0054 - 0.0026i   0.0105 - 0.0190i


s_cd(:,:,81) =

   0.0025 + 0.0322i  -0.0031 + 0.0015i
  -0.0057 + 0.0005i  -0.0016 - 0.0158i


s_cd(:,:,82) =

   0.0066 + 0.0276i  -0.0029 + 0.0026i
  -0.0048 + 0.0027i  -0.0084 - 0.0075i


s_cd(:,:,83) =

   0.0090 + 0.0234i  -0.0023 + 0.0036i
  -0.0026 + 0.0043i  -0.0086 + 0.0015i


s_cd(:,:,84) =

   0.0111 + 0.0198i  -0.0006 + 0.0048i
  -0.0010 + 0.0047i  -0.0038 + 0.0073i
```

```
s_cd(:,:,85) =

   0.0133 + 0.0164i   0.0012 + 0.0055i
   0.0008 + 0.0049i   0.0026 + 0.0082i


s_cd(:,:,86) =

   0.0150 + 0.0117i   0.0041 + 0.0045i
   0.0030 + 0.0044i   0.0067 + 0.0052i


s_cd(:,:,87) =

   0.0151 + 0.0068i   0.0058 + 0.0023i
   0.0044 + 0.0026i   0.0082 + 0.0010i


s_cd(:,:,88) =

   0.0141 + 0.0026i   0.0068 - 0.0011i
   0.0050 + 0.0005i   0.0068 - 0.0032i


s_cd(:,:,89) =

   0.0127 - 0.0009i   0.0049 - 0.0043i
   0.0047 - 0.0017i   0.0030 - 0.0055i


s_cd(:,:,90) =

   0.0103 - 0.0041i   0.0027 - 0.0058i
   0.0031 - 0.0037i  -0.0011 - 0.0052i


s_cd(:,:,91) =

   0.0073 - 0.0062i  -0.0004 - 0.0064i
   0.0016 - 0.0048i  -0.0040 - 0.0027i
```

```
s_cd(:,:,92) =

   0.0046 - 0.0065i  -0.0031 - 0.0056i
  -0.0009 - 0.0048i  -0.0049 + 0.0002i


s_cd(:,:,93) =

   0.0029 - 0.0065i  -0.0048 - 0.0032i
  -0.0023 - 0.0039i  -0.0044 + 0.0027i


s_cd(:,:,94) =

   0.0017 - 0.0067i  -0.0052 - 0.0008i
  -0.0037 - 0.0022i  -0.0027 + 0.0041i


s_cd(:,:,95) =

   0.0003 - 0.0073i  -0.0045 + 0.0009i
  -0.0037 - 0.0007i  -0.0011 + 0.0034i


s_cd(:,:,96) =

  -0.0013 - 0.0083i  -0.0042 + 0.0019i
  -0.0037 - 0.0000i  -0.0010 + 0.0014i


s_cd(:,:,97) =

  -0.0042 - 0.0084i  -0.0042 + 0.0034i
  -0.0044 + 0.0011i  -0.0037 - 0.0004i


s_cd(:,:,98) =

  -0.0067 - 0.0069i  -0.0031 + 0.0063i
  -0.0041 + 0.0041i  -0.0071 + 0.0010i


s_cd(:,:,99) =
```

```
  -0.0075 - 0.0062i   0.0011 + 0.0074i
  -0.0012 + 0.0060i  -0.0088 + 0.0034i


s_cd(:,:,100) =

  -0.0095 - 0.0070i   0.0048 + 0.0064i
   0.0012 + 0.0057i  -0.0101 + 0.0062i


s_cd(:,:,101) =

  -0.0133 - 0.0064i   0.0072 + 0.0031i
   0.0039 + 0.0050i  -0.0104 + 0.0100i


s_cd(:,:,102) =

  -0.0166 - 0.0039i   0.0081 - 0.0006i
   0.0055 + 0.0027i  -0.0087 + 0.0141i


s_cd(:,:,103) =

  -0.0195 - 0.0005i   0.0059 - 0.0048i
   0.0061 - 0.0003i  -0.0052 + 0.0178i


s_cd(:,:,104) =

  -0.0213 + 0.0050i   0.0027 - 0.0062i
   0.0045 - 0.0024i   0.0002 + 0.0193i


s_cd(:,:,105) =

  -0.0200 + 0.0114i   0.0003 - 0.0058i
   0.0035 - 0.0043i   0.0051 + 0.0177i


s_cd(:,:,106) =

  -0.0150 + 0.0161i  -0.0012 - 0.0054i
   0.0017 - 0.0058i   0.0085 + 0.0149i
```

```
s_cd(:,:,107) =

  -0.0093 + 0.0169i  -0.0038 - 0.0047i
  -0.0014 - 0.0056i   0.0101 + 0.0109i


s_cd(:,:,108) =

  -0.0058 + 0.0150i  -0.0055 - 0.0028i
  -0.0037 - 0.0041i   0.0087 + 0.0069i


s_cd(:,:,109) =

  -0.0048 + 0.0134i  -0.0058 - 0.0003i
  -0.0044 - 0.0021i   0.0048 + 0.0057i


s_cd(:,:,110) =

  -0.0052 + 0.0137i  -0.0056 + 0.0019i
  -0.0045 - 0.0004i   0.0015 + 0.0078i


s_cd(:,:,111) =

  -0.0049 + 0.0167i  -0.0046 + 0.0038i
  -0.0044 + 0.0011i   0.0007 + 0.0126i


s_cd(:,:,112) =

  -0.0020 + 0.0210i  -0.0025 + 0.0055i
  -0.0039 + 0.0018i   0.0032 + 0.0172i


s_cd(:,:,113) =

   0.0045 + 0.0245i  -0.0004 + 0.0063i
  -0.0042 + 0.0037i   0.0091 + 0.0204i
```

```
s_cd(:,:,114) =

   0.0134 + 0.0250i   0.0021 + 0.0064i
  -0.0020 + 0.0055i   0.0173 + 0.0201i


s_cd(:,:,115) =

   0.0230 + 0.0210i   0.0051 + 0.0051i
   0.0009 + 0.0068i   0.0253 + 0.0143i


s_cd(:,:,116) =

   0.0305 + 0.0121i   0.0079 + 0.0025i
   0.0046 + 0.0063i   0.0292 + 0.0040i


s_cd(:,:,117) =

   0.0331 + 0.0004i   0.0089 - 0.0019i
   0.0079 + 0.0026i   0.0276 - 0.0072i


s_cd(:,:,118) =

   0.0309 - 0.0111i   0.0067 - 0.0067i
   0.0078 - 0.0019i   0.0213 - 0.0168i


s_cd(:,:,119) =

   0.0249 - 0.0210i   0.0020 - 0.0089i
   0.0050 - 0.0052i   0.0109 - 0.0228i


s_cd(:,:,120) =

   0.0152 - 0.0283i  -0.0024 - 0.0081i
   0.0021 - 0.0062i  -0.0012 - 0.0234i


s_cd(:,:,121) =
```

```
    0.0039 - 0.0309i  -0.0060 - 0.0053i
   -0.0006 - 0.0061i  -0.0120 - 0.0185i


s_cd(:,:,122) =

   -0.0064 - 0.0297i  -0.0074 - 0.0014i
   -0.0030 - 0.0045i  -0.0189 - 0.0097i


s_cd(:,:,123) =

   -0.0151 - 0.0258i  -0.0057 + 0.0023i
   -0.0038 - 0.0030i  -0.0213 + 0.0005i


s_cd(:,:,124) =

   -0.0211 - 0.0197i  -0.0033 + 0.0038i
   -0.0041 - 0.0011i  -0.0187 + 0.0097i


s_cd(:,:,125) =

   -0.0244 - 0.0134i  -0.0010 + 0.0036i
   -0.0041 - 0.0001i  -0.0133 + 0.0153i


s_cd(:,:,126) =

   -0.0257 - 0.0078i  -0.0008 + 0.0024i
   -0.0044 + 0.0007i  -0.0080 + 0.0173i


s_cd(:,:,127) =

   -0.0256 - 0.0023i  -0.0011 + 0.0030i
   -0.0044 + 0.0028i  -0.0048 + 0.0175i


s_cd(:,:,128) =

   -0.0229 + 0.0015i   0.0005 + 0.0041i
   -0.0029 + 0.0049i  -0.0029 + 0.0175i
```

```
s_cd(:,:,129) =

  -0.0208 + 0.0016i   0.0027 + 0.0041i
  -0.0007 + 0.0060i  -0.0030 + 0.0177i


s_cd(:,:,130) =

  -0.0224 + 0.0002i   0.0050 + 0.0021i
   0.0017 + 0.0058i  -0.0040 + 0.0208i


s_cd(:,:,131) =

  -0.0275 + 0.0015i   0.0054 - 0.0005i
   0.0041 + 0.0056i  -0.0026 + 0.0275i


s_cd(:,:,132) =

  -0.0330 + 0.0068i   0.0048 - 0.0036i
   0.0072 + 0.0031i   0.0038 + 0.0339i


s_cd(:,:,133) =

  -0.0375 + 0.0155i   0.0017 - 0.0055i
   0.0084 - 0.0006i   0.0140 + 0.0373i


s_cd(:,:,134) =

  -0.0391 + 0.0286i  -0.0012 - 0.0054i
   0.0071 - 0.0046i   0.0266 + 0.0358i


s_cd(:,:,135) =

  -0.0343 + 0.0433i  -0.0035 - 0.0033i
   0.0039 - 0.0079i   0.0387 + 0.0285i
```

```
s_cd(:,:,136) =

  -0.0229 + 0.0559i  -0.0046 - 0.0004i
  -0.0003 - 0.0087i   0.0475 + 0.0148i


s_cd(:,:,137) =

  -0.0072 + 0.0629i  -0.0033 + 0.0022i
  -0.0048 - 0.0074i   0.0489 - 0.0029i


s_cd(:,:,138) =

   0.0091 + 0.0636i  -0.0004 + 0.0028i
  -0.0080 - 0.0041i   0.0410 - 0.0196i


s_cd(:,:,139) =

   0.0240 + 0.0590i   0.0017 + 0.0011i
  -0.0088 + 0.0004i   0.0260 - 0.0300i


s_cd(:,:,140) =

   0.0354 + 0.0506i   0.0013 - 0.0012i
  -0.0068 + 0.0044i   0.0082 - 0.0309i


s_cd(:,:,141) =

   0.0428 + 0.0403i  -0.0004 - 0.0025i
  -0.0036 + 0.0065i  -0.0063 - 0.0226i


s_cd(:,:,142) =

   0.0468 + 0.0303i  -0.0025 - 0.0019i
   0.0001 + 0.0062i  -0.0130 - 0.0094i


s_cd(:,:,143) =
```

```
   0.0491 + 0.0207i  -0.0044 + 0.0005i
   0.0018 + 0.0047i  -0.0116 + 0.0039i


s_cd(:,:,144) =

   0.0500 + 0.0115i  -0.0037 + 0.0037i
   0.0031 + 0.0033i  -0.0039 + 0.0127i


s_cd(:,:,145) =

   0.0493 + 0.0022i  -0.0007 + 0.0056i
   0.0035 + 0.0019i   0.0060 + 0.0148i


s_cd(:,:,146) =

   0.0466 - 0.0070i   0.0028 + 0.0054i
   0.0039 + 0.0008i   0.0130 + 0.0109i


s_cd(:,:,147) =

   0.0406 - 0.0153i   0.0058 + 0.0029i
   0.0043 - 0.0011i   0.0154 + 0.0056i


s_cd(:,:,148) =

   0.0325 - 0.0192i   0.0061 - 0.0009i
   0.0028 - 0.0031i   0.0144 + 0.0010i


s_cd(:,:,149) =

   0.0258 - 0.0192i   0.0043 - 0.0033i
   0.0012 - 0.0033i   0.0112 - 0.0010i


s_cd(:,:,150) =

   0.0222 - 0.0175i   0.0022 - 0.0039i
   0.0003 - 0.0025i   0.0082 - 0.0000i
```

```
s_cd(:,:,151) =

   0.0208 - 0.0159i   0.0014 - 0.0041i
   0.0002 - 0.0022i   0.0080 + 0.0028i


s_cd(:,:,152) =

   0.0214 - 0.0155i   0.0000 - 0.0047i
  -0.0002 - 0.0032i   0.0110 + 0.0045i


s_cd(:,:,153) =

   0.0225 - 0.0177i  -0.0016 - 0.0050i
  -0.0012 - 0.0034i   0.0149 + 0.0031i


s_cd(:,:,154) =

   0.0219 - 0.0218i  -0.0038 - 0.0052i
  -0.0033 - 0.0035i   0.0179 - 0.0017i


s_cd(:,:,155) =

   0.0180 - 0.0266i  -0.0072 - 0.0031i
  -0.0056 - 0.0019i   0.0173 - 0.0086i


s_cd(:,:,156) =

   0.0117 - 0.0297i  -0.0092 + 0.0012i
  -0.0065 + 0.0012i   0.0123 - 0.0146i


s_cd(:,:,157) =

   0.0046 - 0.0303i  -0.0075 + 0.0062i
  -0.0054 + 0.0054i   0.0046 - 0.0174i
```

```
s_cd(:,:,158) =

  -0.0027 - 0.0291i  -0.0027 + 0.0095i
  -0.0019 + 0.0073i  -0.0036 - 0.0160i


s_cd(:,:,159) =

  -0.0102 - 0.0258i   0.0027 + 0.0091i
   0.0022 + 0.0071i  -0.0108 - 0.0107i


s_cd(:,:,160) =

  -0.0170 - 0.0197i   0.0065 + 0.0061i
   0.0051 + 0.0054i  -0.0142 - 0.0020i


s_cd(:,:,161) =

  -0.0213 - 0.0109i   0.0079 + 0.0027i
   0.0068 + 0.0024i  -0.0125 + 0.0066i


s_cd(:,:,162) =

  -0.0212 - 0.0007i   0.0085 - 0.0007i
   0.0075 - 0.0008i  -0.0075 + 0.0121i


s_cd(:,:,163) =

  -0.0169 + 0.0075i   0.0074 - 0.0045i
   0.0061 - 0.0046i  -0.0023 + 0.0145i


s_cd(:,:,164) =

  -0.0109 + 0.0129i   0.0044 - 0.0070i
   0.0024 - 0.0069i   0.0024 + 0.0155i


s_cd(:,:,165) =
```

```
  -0.0040 + 0.0153i   0.0002 - 0.0082i
  -0.0012 - 0.0065i   0.0071 + 0.0149i


s_cd(:,:,166) =

   0.0023 + 0.0154i  -0.0031 - 0.0068i
  -0.0040 - 0.0047i   0.0107 + 0.0123i


s_cd(:,:,167) =

   0.0073 + 0.0136i  -0.0052 - 0.0043i
  -0.0049 - 0.0021i   0.0121 + 0.0087i


s_cd(:,:,168) =

   0.0106 + 0.0115i  -0.0060 - 0.0020i
  -0.0043 + 0.0000i   0.0110 + 0.0054i


s_cd(:,:,169) =

   0.0131 + 0.0100i  -0.0057 - 0.0001i
  -0.0037 + 0.0010i   0.0082 + 0.0047i


s_cd(:,:,170) =

   0.0150 + 0.0092i  -0.0054 + 0.0019i
  -0.0034 + 0.0019i   0.0057 + 0.0064i


s_cd(:,:,171) =

   0.0177 + 0.0092i  -0.0050 + 0.0038i
  -0.0026 + 0.0031i   0.0049 + 0.0104i


s_cd(:,:,172) =

   0.0228 + 0.0091i  -0.0037 + 0.0060i
  -0.0017 + 0.0041i   0.0072 + 0.0151i
```

```
s_cd(:,:,173) =

   0.0294 + 0.0062i  -0.0009 + 0.0080i
   0.0002 + 0.0051i   0.0135 + 0.0189i


s_cd(:,:,174) =

   0.0362 - 0.0003i   0.0029 + 0.0084i
   0.0029 + 0.0050i   0.0223 + 0.0187i


s_cd(:,:,175) =

   0.0403 - 0.0118i   0.0073 + 0.0064i
   0.0056 + 0.0030i   0.0310 + 0.0133i


s_cd(:,:,176) =

   0.0386 - 0.0263i   0.0104 + 0.0017i
   0.0067 - 0.0004i   0.0359 + 0.0032i


s_cd(:,:,177) =

   0.0290 - 0.0397i   0.0103 - 0.0040i
   0.0054 - 0.0047i   0.0355 - 0.0078i


s_cd(:,:,178) =

   0.0139 - 0.0470i   0.0062 - 0.0096i
   0.0017 - 0.0068i   0.0302 - 0.0176i


s_cd(:,:,179) =

  -0.0019 - 0.0469i  -0.0003 - 0.0110i
  -0.0029 - 0.0062i   0.0214 - 0.0234i
```

```
s_cd(:,:,180) =

  -0.0150 - 0.0412i  -0.0062 - 0.0080i
  -0.0051 - 0.0023i   0.0115 - 0.0245i


s_cd(:,:,181) =

  -0.0237 - 0.0328i  -0.0082 - 0.0029i
  -0.0048 + 0.0010i   0.0028 - 0.0214i


s_cd(:,:,182) =

  -0.0289 - 0.0241i  -0.0069 + 0.0021i
  -0.0026 + 0.0032i  -0.0030 - 0.0154i


s_cd(:,:,183) =

  -0.0320 - 0.0160i  -0.0037 + 0.0044i
   0.0003 + 0.0036i  -0.0054 - 0.0081i


s_cd(:,:,184) =

  -0.0340 - 0.0075i  -0.0005 + 0.0038i
   0.0028 + 0.0019i  -0.0049 - 0.0012i


s_cd(:,:,185) =

  -0.0343 + 0.0019i   0.0008 + 0.0019i
   0.0029 - 0.0010i  -0.0014 + 0.0048i


s_cd(:,:,186) =

  -0.0315 + 0.0121i   0.0005 + 0.0004i
   0.0013 - 0.0027i   0.0046 + 0.0086i


s_cd(:,:,187) =
```

```
  -0.0250 + 0.0214i  -0.0010 + 0.0001i
  -0.0013 - 0.0031i   0.0124 + 0.0084i


s_cd(:,:,188) =

  -0.0155 + 0.0275i  -0.0017 + 0.0014i
  -0.0032 - 0.0023i   0.0187 + 0.0036i


s_cd(:,:,189) =

  -0.0052 + 0.0295i  -0.0015 + 0.0032i
  -0.0047 + 0.0005i   0.0213 - 0.0040i


s_cd(:,:,190) =

   0.0035 + 0.0284i   0.0003 + 0.0049i
  -0.0041 + 0.0039i   0.0195 - 0.0112i


s_cd(:,:,191) =

   0.0101 + 0.0265i   0.0039 + 0.0057i
  -0.0010 + 0.0065i   0.0151 - 0.0166i


s_cd(:,:,192) =

   0.0166 + 0.0241i   0.0078 + 0.0028i
   0.0035 + 0.0058i   0.0089 - 0.0197i


s_cd(:,:,193) =

   0.0223 + 0.0196i   0.0087 - 0.0024i
   0.0060 + 0.0025i   0.0018 - 0.0196i


s_cd(:,:,194) =

   0.0257 + 0.0138i   0.0057 - 0.0074i
   0.0059 - 0.0014i  -0.0038 - 0.0164i
```

```
s_cd(:,:,195) =

   0.0264 + 0.0079i   0.0010 - 0.0092i
   0.0042 - 0.0040i  -0.0068 - 0.0115i


s_cd(:,:,196) =

   0.0247 + 0.0035i  -0.0040 - 0.0080i
   0.0019 - 0.0049i  -0.0072 - 0.0074i


s_cd(:,:,197) =

   0.0223 + 0.0013i  -0.0071 - 0.0046i
  -0.0007 - 0.0051i  -0.0069 - 0.0045i


s_cd(:,:,198) =

   0.0204 + 0.0015i  -0.0076 - 0.0002i
  -0.0026 - 0.0042i  -0.0069 - 0.0019i


s_cd(:,:,199) =

   0.0205 + 0.0027i  -0.0066 + 0.0025i
  -0.0039 - 0.0027i  -0.0071 + 0.0013i


s_cd(:,:,200) =

   0.0218 + 0.0039i  -0.0046 + 0.0046i
  -0.0047 - 0.0007i  -0.0062 + 0.0051i


s_cd(:,:,201) =

   0.0250 + 0.0046i  -0.0020 + 0.0056i
  -0.0046 + 0.0011i  -0.0036 + 0.0096i
```

```
s_cd(:,:,202) =

   0.0295 + 0.0039i   0.0001 + 0.0060i
  -0.0037 + 0.0033i   0.0012 + 0.0130i


s_cd(:,:,203) =

   0.0350 + 0.0013i   0.0022 + 0.0056i
  -0.0024 + 0.0050i   0.0078 + 0.0135i


s_cd(:,:,204) =

   0.0405 - 0.0039i   0.0046 + 0.0038i
   0.0006 + 0.0054i   0.0146 + 0.0102i


s_cd(:,:,205) =

   0.0446 - 0.0126i   0.0058 + 0.0022i
   0.0033 + 0.0048i   0.0182 + 0.0035i


s_cd(:,:,206) =

   0.0448 - 0.0233i   0.0064 - 0.0010i
   0.0052 + 0.0021i   0.0170 - 0.0041i


s_cd(:,:,207) =

   0.0400 - 0.0332i   0.0051 - 0.0044i
   0.0054 - 0.0012i   0.0120 - 0.0090i


s_cd(:,:,208) =

   0.0324 - 0.0393i   0.0017 - 0.0063i
   0.0033 - 0.0031i   0.0058 - 0.0099i


s_cd(:,:,209) =
```

```
   0.0246 - 0.0421i  -0.0014 - 0.0051i
   0.0008 - 0.0034i   0.0001 - 0.0072i


s_cd(:,:,210) =

   0.0181 - 0.0422i  -0.0030 - 0.0031i
  -0.0006 - 0.0029i  -0.0027 - 0.0019i


s_cd(:,:,211) =

   0.0129 - 0.0419i  -0.0028 - 0.0014i
  -0.0013 - 0.0020i  -0.0014 + 0.0040i


s_cd(:,:,212) =

   0.0082 - 0.0413i  -0.0023 - 0.0004i
  -0.0013 - 0.0017i   0.0027 + 0.0078i


s_cd(:,:,213) =

   0.0033 - 0.0407i  -0.0021 - 0.0008i
  -0.0013 - 0.0012i   0.0083 + 0.0084i


s_cd(:,:,214) =

  -0.0024 - 0.0390i  -0.0026 - 0.0007i
  -0.0017 - 0.0009i   0.0132 + 0.0064i


s_cd(:,:,215) =

  -0.0080 - 0.0353i  -0.0039 + 0.0003i
  -0.0021 - 0.0003i   0.0161 + 0.0020i


s_cd(:,:,216) =

  -0.0122 - 0.0299i  -0.0042 + 0.0023i
  -0.0021 + 0.0006i   0.0164 - 0.0025i
```

```
s_cd(:,:,217) =

  -0.0141 - 0.0240i  -0.0023 + 0.0047i
  -0.0013 + 0.0015i   0.0145 - 0.0065i


s_cd(:,:,218) =

  -0.0143 - 0.0183i   0.0001 + 0.0050i
  -0.0010 + 0.0019i   0.0110 - 0.0085i


s_cd(:,:,219) =

  -0.0133 - 0.0136i   0.0019 + 0.0046i
  -0.0006 + 0.0020i   0.0078 - 0.0083i


s_cd(:,:,220) =

  -0.0117 - 0.0095i   0.0037 + 0.0040i
   0.0001 + 0.0028i   0.0056 - 0.0072i


s_cd(:,:,221) =

  -0.0094 - 0.0051i   0.0058 + 0.0023i
   0.0017 + 0.0031i   0.0044 - 0.0062i


s_cd(:,:,222) =

  -0.0050 - 0.0015i   0.0068 - 0.0008i
   0.0030 + 0.0015i   0.0034 - 0.0051i


s_cd(:,:,223) =

   0.0010 + 0.0004i   0.0061 - 0.0042i
   0.0034 - 0.0003i   0.0023 - 0.0042i
```

```
s_cd(:,:,224) =

   0.0070 - 0.0003i   0.0032 - 0.0071i
   0.0027 - 0.0020i   0.0021 - 0.0029i


s_cd(:,:,225) =

   0.0122 - 0.0030i  -0.0005 - 0.0075i
   0.0013 - 0.0027i   0.0018 - 0.0022i


s_cd(:,:,226) =

   0.0156 - 0.0062i  -0.0040 - 0.0063i
   0.0000 - 0.0027i   0.0016 - 0.0015i


s_cd(:,:,227) =

   0.0182 - 0.0098i  -0.0061 - 0.0038i
  -0.0011 - 0.0020i   0.0015 - 0.0009i


s_cd(:,:,228) =

   0.0202 - 0.0134i  -0.0071 - 0.0006i
  -0.0011 - 0.0014i   0.0009 - 0.0005i


s_cd(:,:,229) =

   0.0213 - 0.0174i  -0.0063 + 0.0027i
  -0.0016 - 0.0008i   0.0000 + 0.0003i


s_cd(:,:,230) =

   0.0216 - 0.0211i  -0.0044 + 0.0049i
  -0.0015 - 0.0005i  -0.0012 + 0.0019i


s_cd(:,:,231) =
```

```
   0.0218 - 0.0244i  -0.0019 + 0.0059i
  -0.0017 + 0.0000i  -0.0023 + 0.0052i


s_cd(:,:,232) =

   0.0218 - 0.0282i   0.0006 + 0.0061i
  -0.0017 + 0.0004i  -0.0017 + 0.0097i


s_cd(:,:,233) =

   0.0214 - 0.0328i   0.0024 + 0.0057i
  -0.0016 + 0.0012i   0.0019 + 0.0147i


s_cd(:,:,234) =

   0.0200 - 0.0384i   0.0052 + 0.0047i
  -0.0009 + 0.0021i   0.0081 + 0.0171i


s_cd(:,:,235) =

   0.0163 - 0.0450i   0.0070 + 0.0019i
   0.0004 + 0.0023i   0.0147 + 0.0164i


s_cd(:,:,236) =

   0.0088 - 0.0508i   0.0075 - 0.0021i
   0.0012 + 0.0016i   0.0202 + 0.0128i


s_cd(:,:,237) =

  -0.0010 - 0.0531i   0.0055 - 0.0057i
   0.0014 + 0.0007i   0.0231 + 0.0070i


s_cd(:,:,238) =

  -0.0107 - 0.0514i   0.0019 - 0.0078i
   0.0012 + 0.0005i   0.0216 + 0.0011i
```

```
s_cd(:,:,239) =

  -0.0187 - 0.0465i  -0.0023 - 0.0072i
   0.0011 + 0.0003i   0.0176 - 0.0019i


s_cd(:,:,240) =

  -0.0236 - 0.0403i  -0.0052 - 0.0048i
   0.0017 - 0.0002i   0.0138 - 0.0014i


s_cd(:,:,241) =

  -0.0257 - 0.0342i  -0.0063 - 0.0017i
   0.0017 - 0.0008i   0.0119 + 0.0008i


s_cd(:,:,242) =

  -0.0260 - 0.0291i  -0.0056 + 0.0017i
   0.0014 - 0.0013i   0.0115 + 0.0032i


s_cd(:,:,243) =

  -0.0254 - 0.0256i  -0.0037 + 0.0035i
   0.0009 - 0.0017i   0.0135 + 0.0051i


s_cd(:,:,244) =

  -0.0260 - 0.0224i  -0.0017 + 0.0043i
   0.0004 - 0.0022i   0.0161 + 0.0063i


s_cd(:,:,245) =

  -0.0254 - 0.0187i   0.0001 + 0.0043i
  -0.0002 - 0.0024i   0.0191 + 0.0052i
```

```
s_cd(:,:,246) =

  -0.0243 - 0.0147i   0.0015 + 0.0038i
  -0.0016 - 0.0026i   0.0211 + 0.0033i


s_cd(:,:,247) =

  -0.0218 - 0.0116i   0.0026 + 0.0027i
  -0.0030 - 0.0019i   0.0225 + 0.0011i


s_cd(:,:,248) =

  -0.0186 - 0.0098i   0.0032 + 0.0010i
  -0.0036 - 0.0002i   0.0226 - 0.0009i


s_cd(:,:,249) =

  -0.0159 - 0.0096i   0.0028 - 0.0003i
  -0.0037 + 0.0016i   0.0229 - 0.0023i


s_cd(:,:,250) =

  -0.0147 - 0.0102i   0.0022 - 0.0014i
  -0.0024 + 0.0035i   0.0234 - 0.0034i


s_cd(:,:,251) =

  -0.0148 - 0.0100i   0.0014 - 0.0016i
  -0.0002 + 0.0038i   0.0242 - 0.0050i


s_cd(:,:,252) =

  -0.0151 - 0.0082i   0.0010 - 0.0018i
   0.0016 + 0.0031i   0.0246 - 0.0068i


s_cd(:,:,253) =
```

```
  -0.0142 - 0.0054i   0.0002 - 0.0023i
   0.0022 + 0.0019i   0.0249 - 0.0089i


s_cd(:,:,254) =

  -0.0116 - 0.0027i  -0.0009 - 0.0022i
   0.0023 + 0.0009i   0.0242 - 0.0113i


s_cd(:,:,255) =

  -0.0073 - 0.0011i  -0.0017 - 0.0015i
   0.0022 + 0.0004i   0.0232 - 0.0137i


s_cd(:,:,256) =

  -0.0022 - 0.0017i  -0.0021 - 0.0005i
   0.0023 + 0.0003i   0.0215 - 0.0156i


s_cd(:,:,257) =

   0.0024 - 0.0047i  -0.0018 + 0.0006i
   0.0027 - 0.0004i   0.0196 - 0.0175i


s_cd(:,:,258) =

   0.0058 - 0.0099i  -0.0009 + 0.0011i
   0.0033 - 0.0016i   0.0166 - 0.0191i


s_cd(:,:,259) =

   0.0066 - 0.0163i  -0.0001 + 0.0007i
   0.0031 - 0.0033i   0.0129 - 0.0199i


s_cd(:,:,260) =

   0.0043 - 0.0227i   0.0000 - 0.0001i
   0.0015 - 0.0054i   0.0081 - 0.0193i
```

```
s_cd(:,:,261) =

  -0.0004 - 0.0272i  -0.0007 - 0.0006i
  -0.0015 - 0.0064i   0.0038 - 0.0163i


s_cd(:,:,262) =

  -0.0060 - 0.0291i  -0.0015 + 0.0000i
  -0.0050 - 0.0053i   0.0013 - 0.0112i


s_cd(:,:,263) =

  -0.0111 - 0.0289i  -0.0018 + 0.0014i
  -0.0072 - 0.0025i   0.0018 - 0.0051i


s_cd(:,:,264) =

  -0.0152 - 0.0273i  -0.0010 + 0.0027i
  -0.0082 + 0.0013i   0.0051 - 0.0002i


s_cd(:,:,265) =

  -0.0189 - 0.0249i   0.0007 + 0.0034i
  -0.0071 + 0.0052i   0.0104 + 0.0029i


s_cd(:,:,266) =

  -0.0217 - 0.0217i   0.0028 + 0.0028i
  -0.0040 + 0.0085i   0.0168 + 0.0028i


s_cd(:,:,267) =

  -0.0237 - 0.0177i   0.0040 + 0.0011i
   0.0006 + 0.0100i   0.0227 + 0.0002i
```

```
s_cd(:,:,268) =

  -0.0248 - 0.0126i   0.0041 - 0.0012i
   0.0057 + 0.0085i   0.0263 - 0.0050i


s_cd(:,:,269) =

  -0.0237 - 0.0071i   0.0032 - 0.0030i
   0.0092 + 0.0050i   0.0271 - 0.0101i


s_cd(:,:,270) =

  -0.0203 - 0.0020i   0.0015 - 0.0042i
   0.0104 - 0.0002i   0.0265 - 0.0145i


s_cd(:,:,271) =

  -0.0154 + 0.0012i  -0.0005 - 0.0049i
   0.0091 - 0.0050i   0.0249 - 0.0180i


s_cd(:,:,272) =

  -0.0100 + 0.0019i  -0.0028 - 0.0041i
   0.0056 - 0.0085i   0.0235 - 0.0211i


s_cd(:,:,273) =

  -0.0054 + 0.0004i  -0.0048 - 0.0023i
   0.0010 - 0.0100i   0.0208 - 0.0232i


s_cd(:,:,274) =

  -0.0017 - 0.0024i  -0.0056 + 0.0002i
  -0.0036 - 0.0090i   0.0184 - 0.0259i


s_cd(:,:,275) =
```

```
  -0.0004 - 0.0057i  -0.0048 + 0.0032i
  -0.0071 - 0.0062i   0.0150 - 0.0274i


s_cd(:,:,276) =

  -0.0003 - 0.0089i  -0.0024 + 0.0051i
  -0.0085 - 0.0022i   0.0106 - 0.0281i


s_cd(:,:,277) =

  -0.0018 - 0.0110i   0.0008 + 0.0055i
  -0.0082 + 0.0016i   0.0065 - 0.0275i


s_cd(:,:,278) =

  -0.0036 - 0.0115i   0.0032 + 0.0040i
  -0.0065 + 0.0045i   0.0031 - 0.0257i


s_cd(:,:,279) =

  -0.0041 - 0.0107i   0.0044 + 0.0018i
  -0.0043 + 0.0064i   0.0004 - 0.0234i


s_cd(:,:,280) =

  -0.0039 - 0.0100i   0.0044 - 0.0007i
  -0.0013 + 0.0077i  -0.0015 - 0.0209i


s_cd(:,:,281) =

  -0.0026 - 0.0095i   0.0032 - 0.0022i
   0.0021 + 0.0075i  -0.0029 - 0.0183i


s_cd(:,:,282) =

  -0.0007 - 0.0095i   0.0018 - 0.0032i
   0.0050 + 0.0057i  -0.0034 - 0.0150i
```

```
s_cd(:,:,283) =

   0.0017 - 0.0099i   0.0001 - 0.0032i
   0.0065 + 0.0029i  -0.0026 - 0.0115i


s_cd(:,:,284) =

   0.0045 - 0.0119i  -0.0014 - 0.0026i
   0.0066 + 0.0001i  -0.0005 - 0.0092i


s_cd(:,:,285) =

   0.0069 - 0.0150i  -0.0022 - 0.0013i
   0.0060 - 0.0021i   0.0018 - 0.0080i


s_cd(:,:,286) =

   0.0083 - 0.0193i  -0.0021 - 0.0000i
   0.0047 - 0.0041i   0.0038 - 0.0079i


s_cd(:,:,287) =

   0.0086 - 0.0241i  -0.0013 + 0.0008i
   0.0032 - 0.0055i   0.0055 - 0.0083i


s_cd(:,:,288) =

   0.0070 - 0.0294i  -0.0007 + 0.0007i
   0.0012 - 0.0064i   0.0063 - 0.0089i


s_cd(:,:,289) =

   0.0034 - 0.0344i  -0.0004 + 0.0002i
  -0.0015 - 0.0067i   0.0062 - 0.0091i
```

```
s_cd(:,:,290) =

  -0.0025 - 0.0375i  -0.0011 + 0.0000i
  -0.0043 - 0.0061i   0.0063 - 0.0084i


s_cd(:,:,291) =

  -0.0086 - 0.0371i  -0.0015 + 0.0010i
  -0.0068 - 0.0038i   0.0071 - 0.0069i


s_cd(:,:,292) =

  -0.0123 - 0.0343i  -0.0009 + 0.0021i
  -0.0080 - 0.0005i   0.0093 - 0.0055i


s_cd(:,:,293) =

  -0.0128 - 0.0324i   0.0006 + 0.0026i
  -0.0076 + 0.0029i   0.0132 - 0.0048i


s_cd(:,:,294) =

  -0.0129 - 0.0331i   0.0019 + 0.0019i
  -0.0057 + 0.0060i   0.0174 - 0.0069i


s_cd(:,:,295) =

  -0.0144 - 0.0357i   0.0029 + 0.0004i
  -0.0024 + 0.0081i   0.0203 - 0.0114i


s_cd(:,:,296) =

  -0.0188 - 0.0385i   0.0025 - 0.0013i
   0.0015 + 0.0083i   0.0207 - 0.0173i


s_cd(:,:,297) =
```

```
  -0.0254 - 0.0391i   0.0013 - 0.0022i
   0.0049 + 0.0065i   0.0188 - 0.0227i


s_cd(:,:,298) =

  -0.0321 - 0.0368i  -0.0002 - 0.0023i
   0.0068 + 0.0032i   0.0136 - 0.0266i


s_cd(:,:,299) =

  -0.0374 - 0.0328i  -0.0015 - 0.0015i
   0.0067 + 0.0004i   0.0076 - 0.0268i


s_cd(:,:,300) =

  -0.0414 - 0.0273i  -0.0018 - 0.0000i
   0.0056 - 0.0016i   0.0036 - 0.0248i


s_cd(:,:,301) =

  -0.0441 - 0.0211i  -0.0012 + 0.0012i
   0.0049 - 0.0027i   0.0018 - 0.0210i


s_cd(:,:,302) =

  -0.0453 - 0.0150i   0.0000 + 0.0014i
   0.0042 - 0.0042i   0.0024 - 0.0178i


s_cd(:,:,303) =

  -0.0453 - 0.0086i   0.0011 + 0.0009i
   0.0031 - 0.0058i   0.0047 - 0.0161i


s_cd(:,:,304) =

  -0.0442 - 0.0019i   0.0016 - 0.0001i
   0.0011 - 0.0076i   0.0073 - 0.0164i
```

```
s_cd(:,:,305) =

  -0.0417 + 0.0049i   0.0014 - 0.0015i
  -0.0023 - 0.0084i   0.0098 - 0.0184i


s_cd(:,:,306) =

  -0.0363 + 0.0114i   0.0000 - 0.0024i
  -0.0063 - 0.0076i   0.0117 - 0.0213i


s_cd(:,:,307) =

  -0.0288 + 0.0159i  -0.0017 - 0.0024i
  -0.0102 - 0.0043i   0.0112 - 0.0263i


s_cd(:,:,308) =

  -0.0206 + 0.0173i  -0.0032 - 0.0009i
  -0.0118 + 0.0011i   0.0081 - 0.0313i


s_cd(:,:,309) =

  -0.0135 + 0.0151i  -0.0032 + 0.0012i
  -0.0103 + 0.0071i   0.0021 - 0.0346i


s_cd(:,:,310) =

  -0.0089 + 0.0117i  -0.0020 + 0.0029i
  -0.0057 + 0.0112i  -0.0050 - 0.0347i


s_cd(:,:,311) =

  -0.0060 + 0.0085i  -0.0002 + 0.0035i
   0.0002 + 0.0125i  -0.0117 - 0.0317i
```

```
s_cd(:,:,312) =

  -0.0042 + 0.0054i   0.0014 + 0.0029i
   0.0057 + 0.0111i  -0.0160 - 0.0264i


s_cd(:,:,313) =

  -0.0030 + 0.0028i   0.0024 + 0.0019i
   0.0095 + 0.0075i  -0.0175 - 0.0206i


s_cd(:,:,314) =

  -0.0024 + 0.0004i   0.0028 + 0.0009i
   0.0116 + 0.0029i  -0.0172 - 0.0159i


s_cd(:,:,315) =

  -0.0016 - 0.0016i   0.0031 - 0.0000i
   0.0122 - 0.0018i  -0.0161 - 0.0128i


s_cd(:,:,316) =

  -0.0010 - 0.0045i   0.0033 - 0.0012i
   0.0105 - 0.0069i  -0.0154 - 0.0110i


s_cd(:,:,317) =

  -0.0015 - 0.0078i   0.0029 - 0.0029i
   0.0070 - 0.0111i  -0.0160 - 0.0095i


s_cd(:,:,318) =

  -0.0031 - 0.0107i   0.0013 - 0.0044i
   0.0016 - 0.0134i  -0.0175 - 0.0072i


s_cd(:,:,319) =
```

```
  -0.0061 - 0.0131i  -0.0011 - 0.0049i
  -0.0048 - 0.0132i  -0.0188 - 0.0027i


s_cd(:,:,320) =

  -0.0095 - 0.0140i  -0.0036 - 0.0039i
  -0.0105 - 0.0098i  -0.0182 + 0.0028i


s_cd(:,:,321) =

  -0.0129 - 0.0134i  -0.0053 - 0.0017i
  -0.0141 - 0.0042i  -0.0154 + 0.0087i


s_cd(:,:,322) =

  -0.0158 - 0.0122i  -0.0053 + 0.0011i
  -0.0143 + 0.0025i  -0.0103 + 0.0137i


s_cd(:,:,323) =

  -0.0181 - 0.0100i  -0.0041 + 0.0032i
  -0.0117 + 0.0086i  -0.0036 + 0.0166i


s_cd(:,:,324) =

  -0.0198 - 0.0075i  -0.0024 + 0.0045i
  -0.0065 + 0.0128i   0.0041 + 0.0173i


s_cd(:,:,325) =

  -0.0210 - 0.0039i  -0.0006 + 0.0050i
  -0.0001 + 0.0143i   0.0115 + 0.0152i


s_cd(:,:,326) =

  -0.0207 + 0.0006i   0.0017 + 0.0051i
   0.0063 + 0.0125i   0.0176 + 0.0110i
```

```
s_cd(:,:,327) =

  -0.0179 + 0.0048i   0.0039 + 0.0040i
   0.0107 + 0.0080i   0.0213 + 0.0051i


s_cd(:,:,328) =

  -0.0137 + 0.0077i   0.0057 + 0.0020i
   0.0123 + 0.0024i   0.0227 - 0.0001i


s_cd(:,:,329) =

  -0.0090 + 0.0085i   0.0060 - 0.0010i
   0.0115 - 0.0025i   0.0226 - 0.0046i


s_cd(:,:,330) =

  -0.0048 + 0.0078i   0.0049 - 0.0035i
   0.0092 - 0.0063i   0.0221 - 0.0077i


s_cd(:,:,331) =

  -0.0014 + 0.0060i   0.0026 - 0.0055i
   0.0062 - 0.0091i   0.0214 - 0.0101i


s_cd(:,:,332) =

   0.0012 + 0.0038i  -0.0002 - 0.0061i
   0.0023 - 0.0107i   0.0219 - 0.0124i


s_cd(:,:,333) =

   0.0033 + 0.0017i  -0.0028 - 0.0051i
  -0.0019 - 0.0110i   0.0221 - 0.0149i
```

```
s_cd(:,:,334) =

   0.0040 - 0.0000i  -0.0047 - 0.0030i
  -0.0061 - 0.0096i   0.0215 - 0.0179i


s_cd(:,:,335) =

   0.0053 - 0.0018i  -0.0052 - 0.0006i
  -0.0095 - 0.0066i   0.0207 - 0.0214i


s_cd(:,:,336) =

   0.0065 - 0.0030i  -0.0046 + 0.0015i
  -0.0117 - 0.0023i   0.0195 - 0.0248i


s_cd(:,:,337) =

   0.0082 - 0.0043i  -0.0032 + 0.0031i
  -0.0122 + 0.0026i   0.0173 - 0.0289i


s_cd(:,:,338) =

   0.0107 - 0.0064i  -0.0018 + 0.0036i
  -0.0104 + 0.0078i   0.0131 - 0.0332i


s_cd(:,:,339) =

   0.0133 - 0.0101i  -0.0005 + 0.0040i
  -0.0063 + 0.0121i   0.0073 - 0.0363i


s_cd(:,:,340) =

   0.0147 - 0.0152i   0.0009 + 0.0039i
  -0.0002 + 0.0140i   0.0003 - 0.0366i


s_cd(:,:,341) =
```

```
   0.0145 - 0.0208i   0.0022 + 0.0034i
   0.0064 + 0.0127i  -0.0065 - 0.0346i


s_cd(:,:,342) =

   0.0125 - 0.0266i   0.0036 + 0.0023i
   0.0112 + 0.0082i  -0.0108 - 0.0299i


s_cd(:,:,343) =

   0.0090 - 0.0314i   0.0043 + 0.0008i
   0.0131 + 0.0023i  -0.0124 - 0.0251i


s_cd(:,:,344) =

   0.0048 - 0.0349i   0.0042 - 0.0009i
   0.0125 - 0.0034i  -0.0120 - 0.0222i


s_cd(:,:,345) =

   0.0010 - 0.0378i   0.0036 - 0.0024i
   0.0097 - 0.0075i  -0.0116 - 0.0208i


s_cd(:,:,346) =

  -0.0026 - 0.0408i   0.0026 - 0.0035i
   0.0060 - 0.0105i  -0.0122 - 0.0203i


s_cd(:,:,347) =

  -0.0070 - 0.0445i   0.0012 - 0.0046i
   0.0017 - 0.0122i  -0.0142 - 0.0197i


s_cd(:,:,348) =

  -0.0132 - 0.0484i  -0.0006 - 0.0052i
  -0.0034 - 0.0121i  -0.0177 - 0.0181i
```

```
s_cd(:,:,349) =

  -0.0213 - 0.0509i  -0.0030 - 0.0049i
  -0.0085 - 0.0099i  -0.0214 - 0.0137i


s_cd(:,:,350) =

  -0.0310 - 0.0513i  -0.0052 - 0.0031i
  -0.0121 - 0.0055i  -0.0237 - 0.0069i


s_cd(:,:,351) =

  -0.0410 - 0.0490i  -0.0063 - 0.0006i
  -0.0134 + 0.0004i  -0.0227 + 0.0011i


s_cd(:,:,352) =

  -0.0503 - 0.0439i  -0.0059 + 0.0022i
  -0.0119 + 0.0060i  -0.0186 + 0.0088i


s_cd(:,:,353) =

  -0.0586 - 0.0365i  -0.0046 + 0.0046i
  -0.0080 + 0.0105i  -0.0116 + 0.0143i


s_cd(:,:,354) =

  -0.0648 - 0.0271i  -0.0024 + 0.0063i
  -0.0026 + 0.0127i  -0.0030 + 0.0164i


s_cd(:,:,355) =

  -0.0690 - 0.0158i   0.0007 + 0.0074i
   0.0029 + 0.0122i   0.0048 + 0.0153i
```

```
s_cd(:,:,356) =

  -0.0693 - 0.0033i   0.0043 + 0.0064i
   0.0077 + 0.0095i   0.0114 + 0.0121i


s_cd(:,:,357) =

  -0.0660 + 0.0087i   0.0070 + 0.0036i
   0.0104 + 0.0052i   0.0156 + 0.0075i


s_cd(:,:,358) =

  -0.0595 + 0.0183i   0.0077 - 0.0002i
   0.0109 + 0.0008i   0.0181 + 0.0029i


s_cd(:,:,359) =

  -0.0518 + 0.0246i   0.0065 - 0.0037i
   0.0100 - 0.0031i   0.0198 - 0.0013i


s_cd(:,:,360) =

  -0.0442 + 0.0287i   0.0038 - 0.0058i
   0.0080 - 0.0061i   0.0206 - 0.0056i


s_cd(:,:,361) =

  -0.0377 + 0.0308i   0.0008 - 0.0065i
   0.0055 - 0.0086i   0.0209 - 0.0094i


s_cd(:,:,362) =

  -0.0320 + 0.0324i  -0.0018 - 0.0059i
   0.0022 - 0.0102i   0.0209 - 0.0125i


s_cd(:,:,363) =
```

```
  -0.0265 + 0.0340i  -0.0036 - 0.0044i
  -0.0016 - 0.0108i   0.0209 - 0.0155i


s_cd(:,:,364) =

  -0.0207 + 0.0358i  -0.0046 - 0.0027i
  -0.0057 - 0.0100i   0.0212 - 0.0189i


s_cd(:,:,365) =

  -0.0142 + 0.0367i  -0.0051 - 0.0009i
  -0.0099 - 0.0077i   0.0209 - 0.0232i


s_cd(:,:,366) =

  -0.0069 + 0.0372i  -0.0050 + 0.0006i
  -0.0132 - 0.0034i   0.0194 - 0.0280i


s_cd(:,:,367) =

   0.0017 + 0.0358i  -0.0049 + 0.0024i
  -0.0146 + 0.0028i   0.0160 - 0.0331i


s_cd(:,:,368) =

   0.0106 + 0.0313i  -0.0040 + 0.0043i
  -0.0126 + 0.0097i   0.0102 - 0.0372i


s_cd(:,:,369) =

   0.0178 + 0.0236i  -0.0023 + 0.0060i
  -0.0071 + 0.0153i   0.0026 - 0.0386i


s_cd(:,:,370) =

   0.0212 + 0.0135i   0.0005 + 0.0070i
   0.0011 + 0.0173i  -0.0055 - 0.0369i
```

```
s_cd(:,:,371) =

   0.0211 + 0.0040i   0.0037 + 0.0065i
   0.0091 + 0.0150i  -0.0115 - 0.0320i


s_cd(:,:,372) =

   0.0187 - 0.0042i   0.0065 + 0.0046i
   0.0148 + 0.0090i  -0.0142 - 0.0262i


s_cd(:,:,373) =

   0.0146 - 0.0107i   0.0081 + 0.0012i
   0.0167 + 0.0015i  -0.0141 - 0.0219i


s_cd(:,:,374) =

   0.0098 - 0.0158i   0.0080 - 0.0027i
   0.0155 - 0.0056i  -0.0129 - 0.0200i


s_cd(:,:,375) =

   0.0042 - 0.0189i   0.0062 - 0.0059i
   0.0117 - 0.0114i  -0.0134 - 0.0198i


s_cd(:,:,376) =

  -0.0008 - 0.0211i   0.0030 - 0.0082i
   0.0059 - 0.0153i  -0.0158 - 0.0199i


s_cd(:,:,377) =

  -0.0060 - 0.0225i  -0.0009 - 0.0088i
  -0.0011 - 0.0165i  -0.0196 - 0.0185i
```

```
s_cd(:,:,378) =

  -0.0123 - 0.0229i  -0.0047 - 0.0077i
  -0.0083 - 0.0146i  -0.0237 - 0.0145i


s_cd(:,:,379) =

  -0.0184 - 0.0212i  -0.0078 - 0.0049i
  -0.0140 - 0.0095i  -0.0260 - 0.0079i


s_cd(:,:,380) =

  -0.0234 - 0.0179i  -0.0090 - 0.0008i
  -0.0169 - 0.0022i  -0.0253 - 0.0001i


s_cd(:,:,381) =

  -0.0272 - 0.0135i  -0.0085 + 0.0031i
  -0.0158 + 0.0056i  -0.0213 + 0.0071i


s_cd(:,:,382) =

  -0.0298 - 0.0086i  -0.0065 + 0.0065i
  -0.0113 + 0.0118i  -0.0151 + 0.0121i


s_cd(:,:,383) =

  -0.0314 - 0.0028i  -0.0029 + 0.0088i
  -0.0046 + 0.0150i  -0.0081 + 0.0143i


s_cd(:,:,384) =

  -0.0310 + 0.0037i   0.0014 + 0.0092i
   0.0024 + 0.0147i  -0.0023 + 0.0141i


s_cd(:,:,385) =
```

```
   -0.0277 + 0.0099i   0.0054 + 0.0074i
    0.0079 + 0.0115i   0.0021 + 0.0131i


s_cd(:,:,386) =

   -0.0225 + 0.0138i   0.0079 + 0.0041i
    0.0110 + 0.0068i   0.0055 + 0.0122i


s_cd(:,:,387) =

   -0.0169 + 0.0153i   0.0085 + 0.0002i
    0.0119 + 0.0019i   0.0079 + 0.0115i


s_cd(:,:,388) =

   -0.0123 + 0.0149i   0.0075 - 0.0032i
    0.0112 - 0.0024i   0.0110 + 0.0116i


s_cd(:,:,389) =

   -0.0087 + 0.0136i   0.0052 - 0.0058i
    0.0093 - 0.0061i   0.0147 + 0.0117i


s_cd(:,:,390) =

   -0.0060 + 0.0118i   0.0022 - 0.0070i
    0.0064 - 0.0089i   0.0190 + 0.0113i


s_cd(:,:,391) =

   -0.0046 + 0.0097i  -0.0008 - 0.0071i
    0.0028 - 0.0106i   0.0246 + 0.0089i


s_cd(:,:,392) =

   -0.0039 + 0.0085i  -0.0033 - 0.0059i
   -0.0013 - 0.0112i   0.0295 + 0.0044i
```

```
s_cd(:,:,393) =

  -0.0037 + 0.0083i  -0.0051 - 0.0040i
  -0.0055 - 0.0102i   0.0336 - 0.0015i


s_cd(:,:,394) =

  -0.0023 + 0.0093i  -0.0059 - 0.0017i
  -0.0094 - 0.0077i   0.0357 - 0.0088i


s_cd(:,:,395) =

   0.0013 + 0.0093i  -0.0062 + 0.0004i
  -0.0123 - 0.0032i   0.0360 - 0.0164i


s_cd(:,:,396) =

   0.0053 + 0.0077i  -0.0056 + 0.0027i
  -0.0131 + 0.0022i   0.0339 - 0.0244i


s_cd(:,:,397) =

   0.0088 + 0.0043i  -0.0044 + 0.0047i
  -0.0110 + 0.0079i   0.0295 - 0.0315i


s_cd(:,:,398) =

   0.0104 - 0.0001i  -0.0021 + 0.0063i
  -0.0064 + 0.0119i   0.0232 - 0.0363i


s_cd(:,:,399) =

   0.0107 - 0.0047i   0.0007 + 0.0068i
  -0.0006 + 0.0133i   0.0162 - 0.0385i
```

```
s_cd(:,:,400) =

   0.0096 - 0.0087i   0.0036 + 0.0058i
   0.0049 + 0.0120i   0.0100 - 0.0378i


s_cd(:,:,401) =

   0.0082 - 0.0117i   0.0054 + 0.0038i
   0.0092 + 0.0089i   0.0053 - 0.0356i


s_cd(:,:,402) =

   0.0062 - 0.0149i   0.0063 + 0.0014i
   0.0118 + 0.0045i   0.0026 - 0.0331i


s_cd(:,:,403) =

   0.0041 - 0.0171i   0.0064 - 0.0011i
   0.0126 - 0.0003i   0.0010 - 0.0308i


s_cd(:,:,404) =

   0.0015 - 0.0192i   0.0057 - 0.0035i
   0.0117 - 0.0052i  -0.0006 - 0.0297i


s_cd(:,:,405) =

  -0.0013 - 0.0215i   0.0041 - 0.0056i
   0.0091 - 0.0096i  -0.0027 - 0.0287i


s_cd(:,:,406) =

  -0.0050 - 0.0230i   0.0016 - 0.0071i
   0.0046 - 0.0130i  -0.0058 - 0.0270i


s_cd(:,:,407) =
```

```
  -0.0092 - 0.0234i  -0.0015 - 0.0075i
  -0.0014 - 0.0143i  -0.0086 - 0.0243i


s_cd(:,:,408) =

  -0.0132 - 0.0232i  -0.0048 - 0.0064i
  -0.0078 - 0.0127i  -0.0110 - 0.0198i


s_cd(:,:,409) =

  -0.0173 - 0.0223i  -0.0074 - 0.0040i
  -0.0130 - 0.0079i  -0.0115 - 0.0143i


s_cd(:,:,410) =

  -0.0222 - 0.0204i  -0.0086 - 0.0004i
  -0.0154 - 0.0010i  -0.0098 - 0.0090i


s_cd(:,:,411) =

  -0.0265 - 0.0170i  -0.0081 + 0.0035i
  -0.0140 + 0.0060i  -0.0056 - 0.0048i


s_cd(:,:,412) =

  -0.0298 - 0.0121i  -0.0057 + 0.0069i
  -0.0092 + 0.0115i  -0.0003 - 0.0029i


s_cd(:,:,413) =

  -0.0314 - 0.0069i  -0.0019 + 0.0088i
  -0.0030 + 0.0136i   0.0050 - 0.0035i


s_cd(:,:,414) =

  -0.0315 - 0.0011i   0.0023 + 0.0085i
   0.0030 + 0.0125i   0.0094 - 0.0056i
```

```
s_cd(:,:,415) =

  -0.0301 + 0.0043i   0.0055 + 0.0064i
   0.0072 + 0.0095i   0.0122 - 0.0088i


s_cd(:,:,416) =

  -0.0277 + 0.0087i   0.0072 + 0.0034i
   0.0095 + 0.0057i   0.0138 - 0.0121i


s_cd(:,:,417) =

  -0.0245 + 0.0121i   0.0078 + 0.0001i
   0.0103 + 0.0018i   0.0145 - 0.0149i


s_cd(:,:,418) =

  -0.0211 + 0.0151i   0.0069 - 0.0030i
   0.0100 - 0.0019i   0.0150 - 0.0172i


s_cd(:,:,419) =

  -0.0172 + 0.0169i   0.0050 - 0.0052i
   0.0086 - 0.0052i   0.0155 - 0.0190i


s_cd(:,:,420) =

  -0.0134 + 0.0178i   0.0026 - 0.0066i
   0.0062 - 0.0079i   0.0163 - 0.0207i


s_cd(:,:,421) =

  -0.0103 + 0.0182i  -0.0001 - 0.0070i
   0.0031 - 0.0097i   0.0172 - 0.0232i
```

```
s_cd(:,:,422) =

  -0.0077 + 0.0186i  -0.0030 - 0.0063i
  -0.0008 - 0.0105i   0.0177 - 0.0263i


s_cd(:,:,423) =

  -0.0049 + 0.0197i  -0.0048 - 0.0047i
  -0.0048 - 0.0098i   0.0171 - 0.0299i


s_cd(:,:,424) =

  -0.0018 + 0.0211i  -0.0063 - 0.0026i
  -0.0086 - 0.0075i   0.0153 - 0.0338i


s_cd(:,:,425) =

   0.0028 + 0.0224i  -0.0070 - 0.0001i
  -0.0114 - 0.0035i   0.0125 - 0.0374i


s_cd(:,:,426) =

   0.0083 + 0.0229i  -0.0067 + 0.0024i
  -0.0121 + 0.0016i   0.0083 - 0.0404i


s_cd(:,:,427) =

   0.0145 + 0.0219i  -0.0055 + 0.0050i
  -0.0106 + 0.0065i   0.0030 - 0.0418i


s_cd(:,:,428) =

   0.0211 + 0.0193i  -0.0034 + 0.0071i
  -0.0071 + 0.0104i  -0.0027 - 0.0414i


s_cd(:,:,429) =
```

```
   0.0273 + 0.0148i  -0.0004 + 0.0084i
  -0.0024 + 0.0125i  -0.0076 - 0.0391i


s_cd(:,:,430) =

   0.0322 + 0.0076i   0.0032 + 0.0084i
   0.0028 + 0.0125i  -0.0107 - 0.0356i


s_cd(:,:,431) =

   0.0345 + 0.0000i   0.0070 + 0.0067i
   0.0077 + 0.0106i  -0.0121 - 0.0320i


s_cd(:,:,432) =

   0.0348 - 0.0066i   0.0100 + 0.0031i
   0.0116 + 0.0070i  -0.0124 - 0.0290i


s_cd(:,:,433) =

   0.0350 - 0.0121i   0.0107 - 0.0020i
   0.0139 + 0.0018i  -0.0118 - 0.0267i


s_cd(:,:,434) =

   0.0355 - 0.0176i   0.0086 - 0.0069i
   0.0140 - 0.0044i  -0.0108 - 0.0253i


s_cd(:,:,435) =

   0.0362 - 0.0244i   0.0043 - 0.0098i
   0.0112 - 0.0103i  -0.0096 - 0.0245i


s_cd(:,:,436) =

   0.0363 - 0.0331i  -0.0004 - 0.0103i
   0.0059 - 0.0145i  -0.0090 - 0.0246i
```

```
s_cd(:,:,437) =

   0.0336 - 0.0433i  -0.0044 - 0.0087i
  -0.0010 - 0.0158i  -0.0088 - 0.0250i


s_cd(:,:,438) =

   0.0273 - 0.0538i  -0.0074 - 0.0061i
  -0.0078 - 0.0139i  -0.0091 - 0.0250i


s_cd(:,:,439) =

   0.0164 - 0.0628i  -0.0091 - 0.0027i
  -0.0130 - 0.0092i  -0.0094 - 0.0247i


s_cd(:,:,440) =

   0.0031 - 0.0680i  -0.0094 + 0.0011i
  -0.0156 - 0.0028i  -0.0097 - 0.0244i


s_cd(:,:,441) =

  -0.0113 - 0.0685i  -0.0082 + 0.0049i
  -0.0151 + 0.0039i  -0.0099 - 0.0242i


s_cd(:,:,442) =

  -0.0247 - 0.0657i  -0.0055 + 0.0079i
  -0.0120 + 0.0096i  -0.0103 - 0.0241i


s_cd(:,:,443) =

  -0.0371 - 0.0597i  -0.0016 + 0.0096i
  -0.0073 + 0.0134i  -0.0111 - 0.0241i
```

```
s_cd(:,:,444) =

  -0.0482 - 0.0508i   0.0025 + 0.0095i
  -0.0016 + 0.0152i  -0.0122 - 0.0239i


s_cd(:,:,445) =

  -0.0567 - 0.0387i   0.0064 + 0.0076i
   0.0045 + 0.0150i  -0.0135 - 0.0230i


s_cd(:,:,446) =

  -0.0610 - 0.0250i   0.0092 + 0.0041i
   0.0104 + 0.0123i  -0.0146 - 0.0215i


s_cd(:,:,447) =

  -0.0622 - 0.0111i   0.0101 - 0.0003i
   0.0151 + 0.0073i  -0.0146 - 0.0196i


s_cd(:,:,448) =

  -0.0597 + 0.0026i   0.0088 - 0.0047i
   0.0173 + 0.0005i  -0.0139 - 0.0184i


s_cd(:,:,449) =

  -0.0539 + 0.0153i   0.0058 - 0.0080i
   0.0165 - 0.0071i  -0.0127 - 0.0180i


s_cd(:,:,450) =

  -0.0448 + 0.0255i   0.0017 - 0.0094i
   0.0122 - 0.0140i  -0.0124 - 0.0185i


s_cd(:,:,451) =
```

```
   -0.0343 + 0.0323i  -0.0023 - 0.0089i
    0.0050 - 0.0184i  -0.0128 - 0.0195i


s_cd(:,:,452) =

   -0.0230 + 0.0366i  -0.0055 - 0.0069i
   -0.0036 - 0.0191i  -0.0141 - 0.0197i


s_cd(:,:,453) =

   -0.0116 + 0.0387i  -0.0074 - 0.0041i
   -0.0117 - 0.0156i  -0.0156 - 0.0190i


s_cd(:,:,454) =

    0.0004 + 0.0380i  -0.0081 - 0.0011i
   -0.0172 - 0.0088i  -0.0165 - 0.0173i


s_cd(:,:,455) =

    0.0118 + 0.0351i  -0.0081 + 0.0021i
   -0.0193 - 0.0006i  -0.0164 - 0.0157i


s_cd(:,:,456) =

    0.0217 + 0.0294i  -0.0067 + 0.0050i
   -0.0177 + 0.0075i  -0.0160 - 0.0144i


s_cd(:,:,457) =

    0.0302 + 0.0226i  -0.0044 + 0.0072i
   -0.0129 + 0.0142i  -0.0156 - 0.0136i


s_cd(:,:,458) =

    0.0378 + 0.0140i  -0.0015 + 0.0084i
   -0.0059 + 0.0184i  -0.0156 - 0.0130i
```

```
s_cd(:,:,459) =

   0.0442 + 0.0034i   0.0018 + 0.0086i
   0.0025 + 0.0192i  -0.0158 - 0.0118i


s_cd(:,:,460) =

   0.0482 - 0.0090i   0.0051 + 0.0077i
   0.0104 + 0.0165i  -0.0155 - 0.0100i


s_cd(:,:,461) =

   0.0485 - 0.0225i   0.0082 + 0.0053i
   0.0164 + 0.0107i  -0.0145 - 0.0081i


s_cd(:,:,462) =

   0.0449 - 0.0361i   0.0101 + 0.0016i
   0.0194 + 0.0027i  -0.0127 - 0.0069i


s_cd(:,:,463) =

   0.0384 - 0.0484i   0.0103 - 0.0029i
   0.0188 - 0.0059i  -0.0105 - 0.0066i


s_cd(:,:,464) =

   0.0286 - 0.0587i   0.0085 - 0.0072i
   0.0146 - 0.0134i  -0.0090 - 0.0074i


s_cd(:,:,465) =

   0.0164 - 0.0663i   0.0047 - 0.0106i
   0.0075 - 0.0184i  -0.0088 - 0.0084i
```

```
s_cd(:,:,466) =

   0.0028 - 0.0703i  -0.0003 - 0.0119i
  -0.0011 - 0.0200i  -0.0094 - 0.0090i


s_cd(:,:,467) =

  -0.0112 - 0.0710i  -0.0058 - 0.0107i
  -0.0097 - 0.0175i  -0.0105 - 0.0082i


s_cd(:,:,468) =

  -0.0245 - 0.0687i  -0.0100 - 0.0073i
  -0.0163 - 0.0114i  -0.0104 - 0.0067i


s_cd(:,:,469) =

  -0.0369 - 0.0631i  -0.0123 - 0.0022i
  -0.0195 - 0.0031i  -0.0093 - 0.0053i


s_cd(:,:,470) =

  -0.0474 - 0.0548i  -0.0122 + 0.0036i
  -0.0184 + 0.0056i  -0.0073 - 0.0048i


s_cd(:,:,471) =

  -0.0556 - 0.0443i  -0.0094 + 0.0085i
  -0.0139 + 0.0125i  -0.0058 - 0.0057i


s_cd(:,:,472) =

  -0.0606 - 0.0327i  -0.0047 + 0.0119i
  -0.0072 + 0.0168i  -0.0048 - 0.0071i


s_cd(:,:,473) =
```

```
  -0.0633 - 0.0202i   0.0008 + 0.0124i
   0.0005 + 0.0176i  -0.0049 - 0.0084i


s_cd(:,:,474) =

  -0.0623 - 0.0073i   0.0060 + 0.0109i
   0.0074 + 0.0154i  -0.0054 - 0.0091i


s_cd(:,:,475) =

  -0.0584 + 0.0043i   0.0100 + 0.0072i
   0.0126 + 0.0108i  -0.0058 - 0.0092i


s_cd(:,:,476) =

  -0.0515 + 0.0140i   0.0119 + 0.0022i
   0.0155 + 0.0048i  -0.0057 - 0.0088i


s_cd(:,:,477) =

  -0.0436 + 0.0214i   0.0116 - 0.0028i
   0.0160 - 0.0015i  -0.0051 - 0.0088i


s_cd(:,:,478) =

  -0.0345 + 0.0269i   0.0092 - 0.0071i
   0.0140 - 0.0076i  -0.0044 - 0.0092i


s_cd(:,:,479) =

  -0.0250 + 0.0301i   0.0056 - 0.0102i
   0.0100 - 0.0125i  -0.0039 - 0.0098i


s_cd(:,:,480) =

  -0.0158 + 0.0312i   0.0012 - 0.0117i
   0.0043 - 0.0158i  -0.0040 - 0.0100i
```

```
s_cd(:,:,481) =

  -0.0066 + 0.0308i  -0.0037 - 0.0111i
  -0.0026 - 0.0162i  -0.0038 - 0.0099i


s_cd(:,:,482) =

   0.0024 + 0.0283i  -0.0079 - 0.0088i
  -0.0092 - 0.0139i  -0.0034 - 0.0097i


s_cd(:,:,483) =

   0.0104 + 0.0241i  -0.0109 - 0.0049i
  -0.0142 - 0.0090i  -0.0028 - 0.0098i


s_cd(:,:,484) =

   0.0171 + 0.0180i  -0.0120 - 0.0000i
  -0.0167 - 0.0023i  -0.0025 - 0.0105i


s_cd(:,:,485) =

   0.0223 + 0.0110i  -0.0112 + 0.0050i
  -0.0165 + 0.0050i  -0.0028 - 0.0114i


s_cd(:,:,486) =

   0.0254 + 0.0034i  -0.0083 + 0.0092i
  -0.0129 + 0.0115i  -0.0039 - 0.0122i


s_cd(:,:,487) =

   0.0271 - 0.0047i  -0.0039 + 0.0121i
  -0.0069 + 0.0159i  -0.0054 - 0.0126i
```

```
s_cd(:,:,488) =

   0.0280 - 0.0126i   0.0016 + 0.0130i
   0.0005 + 0.0174i  -0.0075 - 0.0120i


s_cd(:,:,489) =

   0.0270 - 0.0208i   0.0069 + 0.0112i
   0.0078 + 0.0156i  -0.0092 - 0.0102i


s_cd(:,:,490) =

   0.0244 - 0.0296i   0.0112 + 0.0074i
   0.0139 + 0.0109i  -0.0103 - 0.0075i


s_cd(:,:,491) =

   0.0193 - 0.0376i   0.0136 + 0.0020i
   0.0172 + 0.0039i  -0.0097 - 0.0043i


s_cd(:,:,492) =

   0.0119 - 0.0444i   0.0133 - 0.0040i
   0.0171 - 0.0039i  -0.0076 - 0.0014i


s_cd(:,:,493) =

   0.0028 - 0.0484i   0.0105 - 0.0096i
   0.0137 - 0.0109i  -0.0046 + 0.0006i


s_cd(:,:,494) =

  -0.0067 - 0.0494i   0.0054 - 0.0133i
   0.0074 - 0.0156i  -0.0007 + 0.0012i


s_cd(:,:,495) =
```

```
  -0.0156 - 0.0473i  -0.0011 - 0.0144i
  -0.0003 - 0.0169i   0.0034 + 0.0004i


s_cd(:,:,496) =

  -0.0225 - 0.0431i  -0.0073 - 0.0125i
  -0.0072 - 0.0147i   0.0066 - 0.0013i


s_cd(:,:,497) =

  -0.0272 - 0.0378i  -0.0119 - 0.0081i
  -0.0123 - 0.0098i   0.0092 - 0.0036i


s_cd(:,:,498) =

  -0.0296 - 0.0324i  -0.0142 - 0.0022i
  -0.0146 - 0.0036i   0.0114 - 0.0063i


s_cd(:,:,499) =

  -0.0306 - 0.0278i  -0.0138 + 0.0041i
  -0.0143 + 0.0025i   0.0128 - 0.0089i


s_cd(:,:,500) =

  -0.0309 - 0.0240i  -0.0107 + 0.0096i
  -0.0117 + 0.0077i   0.0137 - 0.0117i


s_cd(:,:,501) =

  -0.0308 - 0.0203i  -0.0054 + 0.0133i
  -0.0074 + 0.0114i   0.0140 - 0.0143i


s_cd(:,:,502) =

  -0.0307 - 0.0172i   0.0008 + 0.0142i
  -0.0022 + 0.0130i   0.0138 - 0.0169i
```

```
s_cd(:,:,503) =

  -0.0301 - 0.0142i   0.0069 + 0.0124i
   0.0031 + 0.0127i   0.0135 - 0.0192i


s_cd(:,:,504) =

  -0.0292 - 0.0112i   0.0114 + 0.0081i
   0.0075 + 0.0102i   0.0130 - 0.0212i


s_cd(:,:,505) =

  -0.0280 - 0.0085i   0.0136 + 0.0023i
   0.0109 + 0.0063i   0.0124 - 0.0240i


s_cd(:,:,506) =

  -0.0265 - 0.0060i   0.0132 - 0.0037i
   0.0123 + 0.0015i   0.0114 - 0.0267i


s_cd(:,:,507) =

  -0.0242 - 0.0040i   0.0100 - 0.0088i
   0.0118 - 0.0034i   0.0096 - 0.0301i


s_cd(:,:,508) =

  -0.0222 - 0.0024i   0.0052 - 0.0120i
   0.0094 - 0.0077i   0.0069 - 0.0331i


s_cd(:,:,509) =

  -0.0199 - 0.0010i  -0.0003 - 0.0128i
   0.0054 - 0.0107i   0.0032 - 0.0356i
```

```
s_cd(:,:,510) =

  -0.0173 - 0.0000i  -0.0054 - 0.0113i
   0.0007 - 0.0119i  -0.0011 - 0.0370i


s_cd(:,:,511) =

  -0.0146 + 0.0005i  -0.0094 - 0.0079i
  -0.0041 - 0.0113i  -0.0052 - 0.0370i


s_cd(:,:,512) =

  -0.0120 + 0.0002i  -0.0118 - 0.0034i
  -0.0082 - 0.0085i  -0.0088 - 0.0362i


s_cd(:,:,513) =

  -0.0096 + 0.0001i  -0.0120 + 0.0018i
  -0.0106 - 0.0045i  -0.0114 - 0.0345i


s_cd(:,:,514) =

  -0.0072 - 0.0002i  -0.0099 + 0.0067i
  -0.0114 + 0.0001i  -0.0132 - 0.0331i


s_cd(:,:,515) =

  -0.0047 - 0.0006i  -0.0060 + 0.0101i
  -0.0101 + 0.0044i  -0.0144 - 0.0316i


s_cd(:,:,516) =

  -0.0018 - 0.0011i  -0.0012 + 0.0114i
  -0.0077 + 0.0079i  -0.0154 - 0.0304i


s_cd(:,:,517) =
```

```
   0.0021 - 0.0023i   0.0038 + 0.0107i
  -0.0041 + 0.0101i  -0.0167 - 0.0293i


s_cd(:,:,518) =

   0.0063 - 0.0049i   0.0075 + 0.0078i
   0.0001 + 0.0108i  -0.0181 - 0.0281i


s_cd(:,:,519) =

   0.0102 - 0.0092i   0.0097 + 0.0038i
   0.0042 + 0.0101i  -0.0193 - 0.0262i


s_cd(:,:,520) =

   0.0134 - 0.0149i   0.0100 - 0.0003i
   0.0077 + 0.0077i  -0.0202 - 0.0239i


s_cd(:,:,521) =

   0.0147 - 0.0216i   0.0088 - 0.0040i
   0.0101 + 0.0043i  -0.0207 - 0.0217i


s_cd(:,:,522) =

   0.0144 - 0.0294i   0.0064 - 0.0068i
   0.0112 + 0.0002i  -0.0212 - 0.0196i


s_cd(:,:,523) =

   0.0123 - 0.0361i   0.0033 - 0.0086i
   0.0106 - 0.0041i  -0.0209 - 0.0172i


s_cd(:,:,524) =

   0.0089 - 0.0425i  -0.0002 - 0.0092i
   0.0085 - 0.0080i  -0.0207 - 0.0154i
```

```
s_cd(:,:,525) =

   0.0043 - 0.0485i  -0.0037 - 0.0084i
   0.0051 - 0.0110i  -0.0200 - 0.0134i


s_cd(:,:,526) =

  -0.0017 - 0.0541i  -0.0067 - 0.0065i
   0.0003 - 0.0127i  -0.0192 - 0.0115i


s_cd(:,:,527) =

  -0.0090 - 0.0584i  -0.0090 - 0.0036i
  -0.0049 - 0.0124i  -0.0178 - 0.0099i


s_cd(:,:,528) =

  -0.0168 - 0.0612i  -0.0099 + 0.0002i
  -0.0101 - 0.0097i  -0.0156 - 0.0084i


s_cd(:,:,529) =

  -0.0258 - 0.0627i  -0.0091 + 0.0044i
  -0.0138 - 0.0047i  -0.0130 - 0.0079i


s_cd(:,:,530) =

  -0.0354 - 0.0622i  -0.0068 + 0.0078i
  -0.0149 + 0.0016i  -0.0100 - 0.0083i


s_cd(:,:,531) =

  -0.0448 - 0.0594i  -0.0030 + 0.0102i
  -0.0131 + 0.0080i  -0.0073 - 0.0102i
```

```
s_cd(:,:,532) =

  -0.0537 - 0.0547i   0.0016 + 0.0107i
  -0.0086 + 0.0129i  -0.0058 - 0.0126i


s_cd(:,:,533) =

  -0.0616 - 0.0478i   0.0061 + 0.0093i
  -0.0023 + 0.0156i  -0.0051 - 0.0154i


s_cd(:,:,534) =

  -0.0674 - 0.0394i   0.0095 + 0.0059i
   0.0044 + 0.0153i  -0.0057 - 0.0181i


s_cd(:,:,535) =

  -0.0711 - 0.0297i   0.0112 + 0.0013i
   0.0104 + 0.0122i  -0.0070 - 0.0203i


s_cd(:,:,536) =

  -0.0723 - 0.0198i   0.0108 - 0.0036i
   0.0147 + 0.0068i  -0.0090 - 0.0220i


s_cd(:,:,537) =

  -0.0712 - 0.0104i   0.0083 - 0.0078i
   0.0163 + 0.0001i  -0.0114 - 0.0229i


s_cd(:,:,538) =

  -0.0682 - 0.0019i   0.0042 - 0.0106i
   0.0151 - 0.0068i  -0.0140 - 0.0230i


s_cd(:,:,539) =
```

```
  -0.0640 + 0.0053i  -0.0007 - 0.0115i
   0.0110 - 0.0125i  -0.0168 - 0.0225i


s_cd(:,:,540) =

  -0.0585 + 0.0113i  -0.0056 - 0.0101i
   0.0048 - 0.0161i  -0.0189 - 0.0211i


s_cd(:,:,541) =

  -0.0526 + 0.0162i  -0.0093 - 0.0066i
  -0.0023 - 0.0168i  -0.0208 - 0.0194i


s_cd(:,:,542) =

  -0.0458 + 0.0200i  -0.0112 - 0.0020i
  -0.0092 - 0.0143i  -0.0221 - 0.0174i


s_cd(:,:,543) =

  -0.0391 + 0.0223i  -0.0108 + 0.0029i
  -0.0144 - 0.0093i  -0.0230 - 0.0152i


s_cd(:,:,544) =

  -0.0316 + 0.0237i  -0.0083 + 0.0071i
  -0.0171 - 0.0025i  -0.0235 - 0.0131i


s_cd(:,:,545) =

  -0.0240 + 0.0236i  -0.0045 + 0.0096i
  -0.0167 + 0.0048i  -0.0236 - 0.0111i


s_cd(:,:,546) =

  -0.0164 + 0.0217i  -0.0001 + 0.0104i
  -0.0133 + 0.0112i  -0.0234 - 0.0088i
```

```
s_cd(:,:,547) =

  -0.0095 + 0.0183i   0.0040 + 0.0094i
  -0.0076 + 0.0158i  -0.0230 - 0.0064i


s_cd(:,:,548) =

  -0.0034 + 0.0133i   0.0072 + 0.0068i
  -0.0004 + 0.0177i  -0.0219 - 0.0038i


s_cd(:,:,549) =

   0.0017 + 0.0067i   0.0091 + 0.0035i
   0.0070 + 0.0166i  -0.0200 - 0.0011i


s_cd(:,:,550) =

   0.0047 - 0.0016i   0.0097 - 0.0003i
   0.0136 + 0.0124i  -0.0172 + 0.0012i


s_cd(:,:,551) =

   0.0055 - 0.0100i   0.0089 - 0.0041i
   0.0177 + 0.0057i  -0.0137 + 0.0028i


s_cd(:,:,552) =

   0.0043 - 0.0187i   0.0066 - 0.0074i
   0.0189 - 0.0022i  -0.0095 + 0.0033i


s_cd(:,:,553) =

   0.0005 - 0.0273i   0.0033 - 0.0096i
   0.0164 - 0.0101i  -0.0055 + 0.0027i
```

```
s_cd(:,:,554) =

  -0.0054 - 0.0341i  -0.0009 - 0.0103i
   0.0109 - 0.0162i  -0.0018 + 0.0007i


s_cd(:,:,555) =

  -0.0125 - 0.0395i  -0.0051 - 0.0092i
   0.0030 - 0.0196i   0.0011 - 0.0019i


s_cd(:,:,556) =

  -0.0211 - 0.0427i  -0.0086 - 0.0064i
  -0.0056 - 0.0192i   0.0031 - 0.0047i


s_cd(:,:,557) =

  -0.0294 - 0.0439i  -0.0106 - 0.0022i
  -0.0135 - 0.0149i   0.0045 - 0.0079i


s_cd(:,:,558) =

  -0.0378 - 0.0434i  -0.0106 + 0.0025i
  -0.0186 - 0.0079i   0.0057 - 0.0111i


s_cd(:,:,559) =

  -0.0457 - 0.0409i  -0.0087 + 0.0067i
  -0.0203 + 0.0009i   0.0067 - 0.0150i


s_cd(:,:,560) =

  -0.0534 - 0.0368i  -0.0052 + 0.0097i
  -0.0180 + 0.0092i   0.0068 - 0.0196i


s_cd(:,:,561) =
```

```
   -0.0599 - 0.0308i  -0.0005 + 0.0110i
   -0.0123 + 0.0160i   0.0055 - 0.0247i


s_cd(:,:,562) =

  -0.0651 - 0.0234i   0.0042 + 0.0103i
  -0.0044 + 0.0197i   0.0028 - 0.0296i


s_cd(:,:,563) =

  -0.0686 - 0.0148i   0.0082 + 0.0076i
   0.0043 + 0.0197i  -0.0016 - 0.0335i


s_cd(:,:,564) =

  -0.0703 - 0.0054i   0.0107 + 0.0036i
   0.0121 + 0.0161i  -0.0069 - 0.0360i


s_cd(:,:,565) =

  -0.0692 + 0.0046i   0.0113 - 0.0014i
   0.0178 + 0.0096i  -0.0124 - 0.0369i


s_cd(:,:,566) =

  -0.0658 + 0.0137i   0.0096 - 0.0060i
   0.0202 + 0.0011i  -0.0179 - 0.0362i


s_cd(:,:,567) =

  -0.0603 + 0.0215i   0.0061 - 0.0096i
   0.0188 - 0.0074i  -0.0227 - 0.0340i


s_cd(:,:,568) =

  -0.0534 + 0.0275i   0.0013 - 0.0112i
   0.0138 - 0.0147i  -0.0271 - 0.0312i
```

```
s_cd(:,:,569) =

  -0.0462 + 0.0313i  -0.0035 - 0.0106i
   0.0063 - 0.0189i  -0.0305 - 0.0275i


s_cd(:,:,570) =

  -0.0392 + 0.0337i  -0.0075 - 0.0081i
  -0.0020 - 0.0198i  -0.0331 - 0.0232i


s_cd(:,:,571) =

  -0.0320 + 0.0345i  -0.0100 - 0.0042i
  -0.0101 - 0.0173i  -0.0347 - 0.0189i


s_cd(:,:,572) =

  -0.0254 + 0.0343i  -0.0108 + 0.0004i
  -0.0162 - 0.0115i  -0.0354 - 0.0143i


s_cd(:,:,573) =

  -0.0189 + 0.0337i  -0.0096 + 0.0047i
  -0.0197 - 0.0039i  -0.0352 - 0.0102i


s_cd(:,:,574) =

  -0.0122 + 0.0316i  -0.0068 + 0.0081i
  -0.0197 + 0.0047i  -0.0346 - 0.0059i


s_cd(:,:,575) =

  -0.0056 + 0.0280i  -0.0030 + 0.0102i
  -0.0161 + 0.0126i  -0.0337 - 0.0020i
```

```
s_cd(:,:,576) =

   0.0002 + 0.0229i   0.0013 + 0.0105i
  -0.0093 + 0.0184i  -0.0320 + 0.0022i


s_cd(:,:,577) =

   0.0050 + 0.0160i   0.0054 + 0.0090i
  -0.0007 + 0.0208i  -0.0293 + 0.0063i


s_cd(:,:,578) =

   0.0081 + 0.0084i   0.0085 + 0.0060i
   0.0083 + 0.0190i  -0.0251 + 0.0100i


s_cd(:,:,579) =

   0.0093 - 0.0002i   0.0102 + 0.0021i
   0.0156 + 0.0136i  -0.0199 + 0.0125i


s_cd(:,:,580) =

   0.0084 - 0.0088i   0.0101 - 0.0023i
   0.0200 + 0.0057i  -0.0137 + 0.0135i


s_cd(:,:,581) =

   0.0052 - 0.0170i   0.0084 - 0.0061i
   0.0205 - 0.0032i  -0.0075 + 0.0130i


s_cd(:,:,582) =

  -0.0003 - 0.0242i   0.0053 - 0.0091i
   0.0171 - 0.0116i  -0.0018 + 0.0105i


s_cd(:,:,583) =
```

```
  -0.0075 - 0.0296i   0.0012 - 0.0105i
   0.0105 - 0.0178i   0.0031 + 0.0070i


s_cd(:,:,584) =

  -0.0157 - 0.0324i  -0.0034 - 0.0101i
   0.0021 - 0.0204i   0.0068 + 0.0026i


s_cd(:,:,585) =

  -0.0243 - 0.0329i  -0.0075 - 0.0078i
  -0.0067 - 0.0192i   0.0097 - 0.0024i


s_cd(:,:,586) =

  -0.0323 - 0.0308i  -0.0101 - 0.0039i
  -0.0141 - 0.0146i   0.0108 - 0.0076i


s_cd(:,:,587) =

  -0.0390 - 0.0272i  -0.0107 + 0.0006i
  -0.0190 - 0.0072i   0.0112 - 0.0127i


s_cd(:,:,588) =

  -0.0445 - 0.0226i  -0.0094 + 0.0050i
  -0.0200 + 0.0015i   0.0108 - 0.0181i


s_cd(:,:,589) =

  -0.0489 - 0.0171i  -0.0064 + 0.0085i
  -0.0173 + 0.0098i   0.0090 - 0.0232i


s_cd(:,:,590) =

  -0.0522 - 0.0109i  -0.0024 + 0.0104i
  -0.0115 + 0.0159i   0.0063 - 0.0286i
```

```
s_cd(:,:,591) =

  -0.0545 - 0.0041i   0.0023 + 0.0105i
  -0.0037 + 0.0192i   0.0020 - 0.0333i


s_cd(:,:,592) =

  -0.0553 + 0.0032i   0.0064 + 0.0087i
   0.0046 + 0.0189i  -0.0030 - 0.0367i


s_cd(:,:,593) =

  -0.0543 + 0.0110i   0.0094 + 0.0051i
   0.0120 + 0.0150i  -0.0089 - 0.0390i


s_cd(:,:,594) =

  -0.0514 + 0.0190i   0.0109 + 0.0006i
   0.0170 + 0.0084i  -0.0148 - 0.0401i


s_cd(:,:,595) =

  -0.0461 + 0.0262i   0.0101 - 0.0041i
   0.0188 + 0.0005i  -0.0205 - 0.0397i


s_cd(:,:,596) =

  -0.0389 + 0.0317i   0.0073 - 0.0079i
   0.0170 - 0.0074i  -0.0259 - 0.0387i


s_cd(:,:,597) =

  -0.0301 + 0.0347i   0.0035 - 0.0101i
   0.0121 - 0.0136i  -0.0312 - 0.0368i
```

```
s_cd(:,:,598) =

  -0.0208 + 0.0348i  -0.0012 - 0.0106i
   0.0053 - 0.0171i  -0.0360 - 0.0342i


s_cd(:,:,599) =

  -0.0124 + 0.0328i  -0.0056 - 0.0093i
  -0.0021 - 0.0175i  -0.0408 - 0.0304i


s_cd(:,:,600) =

  -0.0057 + 0.0285i  -0.0089 - 0.0062i
  -0.0089 - 0.0151i  -0.0451 - 0.0260i


s_cd(:,:,601) =

  -0.0004 + 0.0231i  -0.0105 - 0.0017i
  -0.0141 - 0.0100i  -0.0493 - 0.0198i


s_cd(:,:,602) =

   0.0024 + 0.0172i  -0.0103 + 0.0026i
  -0.0170 - 0.0035i  -0.0518 - 0.0126i


s_cd(:,:,603) =

   0.0041 + 0.0118i  -0.0083 + 0.0066i
  -0.0170 + 0.0038i  -0.0529 - 0.0047i


s_cd(:,:,604) =

   0.0048 + 0.0066i  -0.0047 + 0.0096i
  -0.0141 + 0.0104i  -0.0522 + 0.0040i


s_cd(:,:,605) =
```

```
   0.0048 + 0.0017i  -0.0005 + 0.0106i
  -0.0086 + 0.0155i  -0.0495 + 0.0125i


s_cd(:,:,606) =

   0.0041 - 0.0031i   0.0038 + 0.0098i
  -0.0012 + 0.0178i  -0.0444 + 0.0203i


s_cd(:,:,607) =

   0.0026 - 0.0080i   0.0075 + 0.0074i
   0.0064 + 0.0167i  -0.0374 + 0.0269i


s_cd(:,:,608) =

   0.0004 - 0.0127i   0.0099 + 0.0036i
   0.0130 + 0.0124i  -0.0291 + 0.0312i


s_cd(:,:,609) =

  -0.0035 - 0.0175i   0.0106 - 0.0008i
   0.0170 + 0.0056i  -0.0200 + 0.0334i


s_cd(:,:,610) =

  -0.0086 - 0.0214i   0.0093 - 0.0052i
   0.0177 - 0.0020i  -0.0104 + 0.0331i


s_cd(:,:,611) =

  -0.0153 - 0.0238i   0.0064 - 0.0086i
   0.0151 - 0.0092i  -0.0014 + 0.0309i


s_cd(:,:,612) =

  -0.0222 - 0.0237i   0.0022 - 0.0105i
   0.0096 - 0.0146i   0.0066 + 0.0267i
```

```
s_cd(:,:,613) =

  -0.0287 - 0.0218i  -0.0024 - 0.0105i
   0.0024 - 0.0172i   0.0135 + 0.0211i


s_cd(:,:,614) =

  -0.0340 - 0.0179i  -0.0066 - 0.0084i
  -0.0049 - 0.0162i   0.0194 + 0.0143i


s_cd(:,:,615) =

  -0.0375 - 0.0129i  -0.0095 - 0.0048i
  -0.0110 - 0.0125i   0.0239 + 0.0065i


s_cd(:,:,616) =

  -0.0387 - 0.0075i  -0.0107 - 0.0004i
  -0.0149 - 0.0067i   0.0268 - 0.0021i


s_cd(:,:,617) =

  -0.0385 - 0.0027i  -0.0099 + 0.0041i
  -0.0161 + 0.0000i   0.0279 - 0.0115i


s_cd(:,:,618) =

  -0.0373 + 0.0011i  -0.0072 + 0.0079i
  -0.0145 + 0.0065i   0.0270 - 0.0207i


s_cd(:,:,619) =

  -0.0355 + 0.0042i  -0.0031 + 0.0103i
  -0.0106 + 0.0118i   0.0239 - 0.0297i
```

```
s_cd(:,:,620) =

  -0.0332 + 0.0066i   0.0015 + 0.0107i
```

References

Fan, W., A. C. W. Lu, L. L. Wai, and B. K. Lok. “Mixed-Mode S-Parameter Characterization of Differential Structures.” Electronic Packaging Technology Conference, pp. 533–537, 2003.

See Also

s2scc | s2sdc | s2sdd | s2smm | smm2s

Introduced in R2006a

s2sdc

Convert 4-port, single-ended S-parameters to 2-port, cross-mode S-parameters (S_{dc})

Syntax

```
sdc_params = s2sdc(s_params)
sdc_params = s2sdc(s_params,option)
```

Description

`sdc_params = s2sdc(s_params)` converts the 2*N*-port, single-ended S-parameters, `s_params`, to *N*-port, cross-mode S-parameters, `sdc_params`. `sdc_params` is a complex *N*-by-*N*-by-*M* array that represents *M* *N*-port, cross-mode S-parameters (S_{dc}).

`sdc_params = s2sdc(s_params,option)` converts S-parameters based on the optional `option` argument, which indicates the port-ordering convention of the S-parameters.

Input Arguments

`s_params` — S-parameters
array

S-parameters, specified as a complex 4-by-4-by-*M* array, that represents *M* 4-port S-parameters.

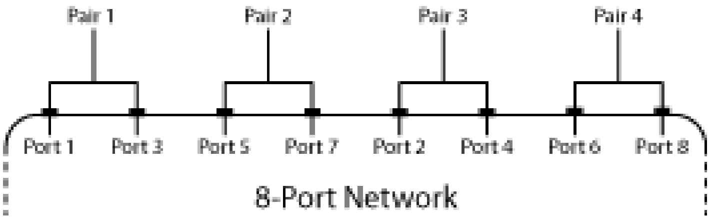
`option` — Port order
1 (default) | 2 | 3

Port order, specified as `1`, `2`, `3`, determines how the function orders the ports:

- `1` — `s2sdc` pairs the odd-numbered ports together first, followed by the even-numbered ports. For example, in a single-ended, 8-port network:

- Ports 1 and 3 become cross-mode pair 1.
- Ports 5 and 7 become cross-mode pair 2.
- Ports 2 and 4 become cross-mode pair 3.
- Ports 6 and 8 become cross-mode pair 4.

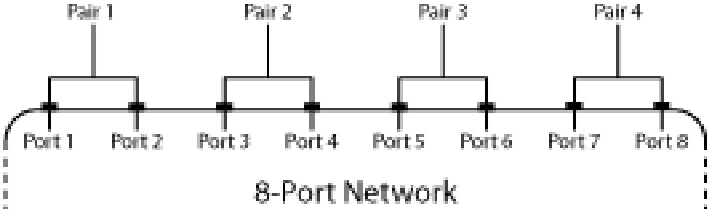
The following figure illustrates this convention for an 8-port device.



- `2` — `s2sdc` pairs the input and output ports in ascending order. For example, in a single-ended, 8-port network:

  - Ports 1 and 2 become cross-mode pair 1.
  - Ports 3 and 4 become cross-mode pair 2.
  - Ports 5 and 6 become cross-mode pair 3.
  - Ports 7 and 8 become cross-mode pair 4.

  The following figure illustrates this convention for an 8-port device.

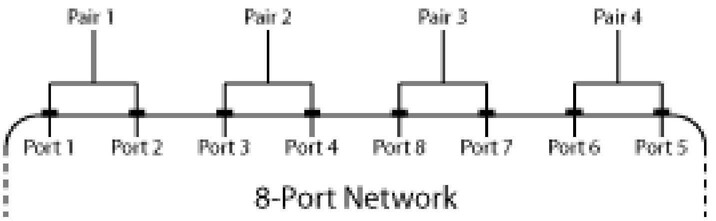


- `3` — `s2sdc` pairs the input ports in ascending order and the output ports in descending order. For example, in a single-ended, 8-port network:

  - Ports 1 and 2 become cross-mode pair 1.
  - Ports 3 and 4 become cross-mode pair 2.
  - Ports 8 and 7 become cross-mode pair 3.

- Ports 6 and 5 become cross-mode pair 4.

The following figure illustrates this convention for an 8-port device.



Examples

4-port Single-Ended S-Parameters to 2-port Cross-Mode S-Parameters

Convert network data to cross-mode S-parameters using the default port ordering

```
ckt = read(rfckt.passive,'default.s4p');
s4p = ckt.NetworkData.Data;
s_dc = s2sdc(s4p)
```

```
s_dc =
s_dc(:,:,1) =

   0.0024 - 0.0035i  -0.0005 + 0.0019i
   0.0007 - 0.0012i   0.0023 - 0.0027i


s_dc(:,:,2) =

   0.0020 - 0.0033i   0.0015 - 0.0012i
   0.0005 - 0.0008i   0.0006 - 0.0044i


s_dc(:,:,3) =

   0.0018 - 0.0040i   0.0006 - 0.0005i
  -0.0006 - 0.0011i  -0.0011 - 0.0046i
```

```
s_dc(:,:,4) =

   0.0004 - 0.0049i  -0.0000 - 0.0013i
  -0.0005 - 0.0011i  -0.0026 - 0.0030i


s_dc(:,:,5) =

  -0.0010 - 0.0047i  -0.0003 - 0.0005i
  -0.0010 - 0.0000i  -0.0034 - 0.0014i


s_dc(:,:,6) =

  -0.0018 - 0.0029i  -0.0009 - 0.0014i
  -0.0009 - 0.0007i  -0.0033 + 0.0005i


s_dc(:,:,7) =

  -0.0017 - 0.0018i  -0.0014 + 0.0002i
  -0.0014 + 0.0001i  -0.0014 + 0.0012i


s_dc(:,:,8) =

  -0.0008 - 0.0014i  -0.0012 + 0.0002i
  -0.0010 + 0.0013i  -0.0003 + 0.0005i


s_dc(:,:,9) =

   0.0000 - 0.0024i  -0.0000 + 0.0004i
  -0.0010 + 0.0015i   0.0006 - 0.0010i


s_dc(:,:,10) =

   0.0002 - 0.0035i   0.0001 + 0.0004i
  -0.0008 + 0.0017i  -0.0010 - 0.0025i


s_dc(:,:,11) =
```

```
   0.0000 - 0.0046i   0.0001 + 0.0014i
  -0.0000 + 0.0022i  -0.0028 - 0.0021i


s_dc(:,:,12) =

  -0.0010 - 0.0056i   0.0012 + 0.0003i
   0.0006 + 0.0018i  -0.0045 - 0.0005i


s_dc(:,:,13) =

  -0.0026 - 0.0063i   0.0005 - 0.0002i
   0.0019 + 0.0012i  -0.0049 + 0.0025i


s_dc(:,:,14) =

  -0.0050 - 0.0057i   0.0023 - 0.0007i
   0.0027 + 0.0007i  -0.0031 + 0.0051i


s_dc(:,:,15) =

  -0.0071 - 0.0050i   0.0010 - 0.0008i
   0.0025 - 0.0009i   0.0003 + 0.0068i


s_dc(:,:,16) =

  -0.0088 - 0.0035i   0.0011 - 0.0009i
   0.0012 - 0.0020i   0.0041 + 0.0056i


s_dc(:,:,17) =

  -0.0098 - 0.0009i  -0.0001 - 0.0011i
   0.0009 - 0.0037i   0.0068 + 0.0024i


s_dc(:,:,18) =

  -0.0099 + 0.0020i  -0.0001 - 0.0016i
  -0.0005 - 0.0035i   0.0069 - 0.0020i
```

```
s_dc(:,:,19) =

  -0.0089 + 0.0040i  -0.0006 - 0.0004i
  -0.0023 - 0.0021i   0.0047 - 0.0054i


s_dc(:,:,20) =

  -0.0078 + 0.0061i  -0.0011 - 0.0003i
  -0.0027 - 0.0005i   0.0014 - 0.0076i


s_dc(:,:,21) =

  -0.0060 + 0.0077i  -0.0009 - 0.0002i
  -0.0029 + 0.0012i  -0.0027 - 0.0073i


s_dc(:,:,22) =

  -0.0038 + 0.0084i  -0.0008 + 0.0001i
  -0.0020 + 0.0013i  -0.0059 - 0.0048i


s_dc(:,:,23) =

  -0.0013 + 0.0083i  -0.0009 + 0.0002i
  -0.0010 + 0.0023i  -0.0068 - 0.0012i


s_dc(:,:,24) =

   0.0004 + 0.0074i  -0.0007 + 0.0008i
  -0.0005 + 0.0022i  -0.0053 + 0.0020i


s_dc(:,:,25) =

   0.0015 + 0.0063i   0.0004 + 0.0012i
   0.0007 + 0.0022i  -0.0025 + 0.0032i
```

```
s_dc(:,:,26) =

   0.0015 + 0.0055i   0.0014 + 0.0007i
   0.0012 + 0.0023i   0.0003 + 0.0022i


s_dc(:,:,27) =

   0.0018 + 0.0056i   0.0015 + 0.0010i
   0.0022 + 0.0018i   0.0021 - 0.0001i


s_dc(:,:,28) =

   0.0033 + 0.0058i   0.0015 + 0.0003i
   0.0033 + 0.0009i   0.0017 - 0.0029i


s_dc(:,:,29) =

   0.0050 + 0.0055i   0.0011 - 0.0007i
   0.0035 - 0.0007i  -0.0004 - 0.0055i


s_dc(:,:,30) =

   0.0071 + 0.0048i   0.0012 - 0.0002i
   0.0028 - 0.0024i  -0.0037 - 0.0061i


s_dc(:,:,31) =

   0.0096 + 0.0025i   0.0012 - 0.0003i
   0.0017 - 0.0033i  -0.0072 - 0.0046i


s_dc(:,:,32) =

   0.0101 - 0.0009i   0.0010 - 0.0015i
  -0.0002 - 0.0043i  -0.0088 - 0.0008i


s_dc(:,:,33) =
```

```
   0.0098 - 0.0037i   0.0004 - 0.0019i
  -0.0019 - 0.0035i  -0.0084 + 0.0026i


s_dc(:,:,34) =

   0.0085 - 0.0062i  -0.0008 - 0.0023i
  -0.0032 - 0.0017i  -0.0062 + 0.0056i


s_dc(:,:,35) =

   0.0066 - 0.0076i  -0.0021 - 0.0014i
  -0.0036 - 0.0003i  -0.0024 + 0.0069i


s_dc(:,:,36) =

   0.0049 - 0.0080i  -0.0022 - 0.0007i
  -0.0033 + 0.0015i   0.0012 + 0.0051i


s_dc(:,:,37) =

   0.0039 - 0.0078i  -0.0018 + 0.0007i
  -0.0021 + 0.0023i   0.0029 + 0.0014i


s_dc(:,:,38) =

   0.0037 - 0.0086i  -0.0013 + 0.0010i
  -0.0013 + 0.0026i   0.0014 - 0.0026i


s_dc(:,:,39) =

   0.0021 - 0.0097i  -0.0017 + 0.0006i
  -0.0009 + 0.0025i  -0.0032 - 0.0047i


s_dc(:,:,40) =

   0.0011 - 0.0092i  -0.0008 + 0.0030i
   0.0001 + 0.0044i  -0.0068 - 0.0024i
```

```
s_dc(:,:,41) =

   0.0002 - 0.0109i   0.0010 + 0.0021i
   0.0029 + 0.0038i  -0.0085 + 0.0005i


s_dc(:,:,42) =

  -0.0028 - 0.0121i   0.0011 + 0.0012i
   0.0041 + 0.0019i  -0.0087 + 0.0044i


s_dc(:,:,43) =

  -0.0068 - 0.0115i   0.0012 + 0.0014i
   0.0045 - 0.0002i  -0.0061 + 0.0082i


s_dc(:,:,44) =

  -0.0101 - 0.0089i   0.0019 + 0.0005i
   0.0040 - 0.0027i  -0.0016 + 0.0101i


s_dc(:,:,45) =

  -0.0124 - 0.0053i   0.0022 - 0.0002i
   0.0022 - 0.0042i   0.0034 + 0.0095i


s_dc(:,:,46) =

  -0.0137 - 0.0011i   0.0023 - 0.0007i
   0.0004 - 0.0048i   0.0082 + 0.0064i


s_dc(:,:,47) =

  -0.0134 + 0.0044i   0.0020 - 0.0021i
  -0.0014 - 0.0042i   0.0106 + 0.0010i
```

```
s_dc(:,:,48) =

  -0.0100 + 0.0095i   0.0006 - 0.0034i
  -0.0032 - 0.0038i   0.0106 - 0.0039i


s_dc(:,:,49) =

  -0.0049 + 0.0128i  -0.0010 - 0.0036i
  -0.0051 - 0.0015i   0.0095 - 0.0092i


s_dc(:,:,50) =

   0.0015 + 0.0144i  -0.0028 - 0.0027i
  -0.0049 + 0.0011i   0.0058 - 0.0143i


s_dc(:,:,51) =

   0.0089 + 0.0132i  -0.0036 - 0.0013i
  -0.0035 + 0.0031i  -0.0004 - 0.0175i


s_dc(:,:,52) =

   0.0156 + 0.0086i  -0.0044 + 0.0012i
  -0.0010 + 0.0041i  -0.0079 - 0.0177i


s_dc(:,:,53) =

   0.0196 + 0.0016i  -0.0036 + 0.0028i
   0.0008 + 0.0037i  -0.0151 - 0.0145i


s_dc(:,:,54) =

   0.0208 - 0.0064i  -0.0019 + 0.0044i
   0.0017 + 0.0022i  -0.0203 - 0.0077i


s_dc(:,:,55) =
```

```
   0.0184 - 0.0140i   0.0005 + 0.0046i
   0.0018 + 0.0007i  -0.0215 + 0.0010i


s_dc(:,:,56) =

   0.0136 - 0.0195i   0.0021 + 0.0040i
   0.0017 + 0.0004i  -0.0181 + 0.0092i


s_dc(:,:,57) =

   0.0077 - 0.0220i   0.0039 + 0.0027i
   0.0015 + 0.0001i  -0.0104 + 0.0140i


s_dc(:,:,58) =

   0.0027 - 0.0222i   0.0044 + 0.0004i
   0.0019 + 0.0006i  -0.0017 + 0.0135i


s_dc(:,:,59) =

  -0.0006 - 0.0217i   0.0038 - 0.0018i
   0.0023 + 0.0000i   0.0049 + 0.0080i


s_dc(:,:,60) =

  -0.0031 - 0.0214i   0.0024 - 0.0030i
   0.0027 - 0.0014i   0.0065 - 0.0004i


s_dc(:,:,61) =

  -0.0059 - 0.0216i   0.0010 - 0.0034i
   0.0022 - 0.0027i   0.0027 - 0.0073i


s_dc(:,:,62) =

  -0.0096 - 0.0210i  -0.0004 - 0.0033i
   0.0006 - 0.0040i  -0.0040 - 0.0110i
```

```
s_dc(:,:,63) =

  -0.0132 - 0.0193i  -0.0021 - 0.0029i
  -0.0016 - 0.0043i  -0.0118 - 0.0097i


s_dc(:,:,64) =

  -0.0162 - 0.0163i  -0.0031 - 0.0021i
  -0.0034 - 0.0022i  -0.0162 - 0.0042i


s_dc(:,:,65) =

  -0.0172 - 0.0124i  -0.0033 - 0.0002i
  -0.0041 - 0.0010i  -0.0165 + 0.0018i


s_dc(:,:,66) =

  -0.0160 - 0.0090i  -0.0029 + 0.0010i
  -0.0039 + 0.0005i  -0.0133 + 0.0051i


s_dc(:,:,67) =

  -0.0130 - 0.0079i  -0.0023 + 0.0015i
  -0.0041 + 0.0017i  -0.0108 + 0.0047i


s_dc(:,:,68) =

  -0.0110 - 0.0097i  -0.0023 + 0.0020i
  -0.0042 + 0.0034i  -0.0124 + 0.0027i


s_dc(:,:,69) =

  -0.0102 - 0.0123i  -0.0014 + 0.0037i
  -0.0023 + 0.0067i  -0.0160 + 0.0044i
```

```
s_dc(:,:,70) =

  -0.0114 - 0.0176i   0.0013 + 0.0036i
   0.0023 + 0.0076i  -0.0194 + 0.0080i


s_dc(:,:,71) =

  -0.0174 - 0.0225i   0.0021 + 0.0022i
   0.0060 + 0.0042i  -0.0225 + 0.0149i


s_dc(:,:,72) =

  -0.0276 - 0.0238i   0.0022 + 0.0003i
   0.0076 + 0.0006i  -0.0217 + 0.0256i


s_dc(:,:,73) =

  -0.0392 - 0.0193i   0.0014 + 0.0006i
   0.0063 - 0.0033i  -0.0138 + 0.0369i


s_dc(:,:,74) =

  -0.0488 - 0.0082i   0.0017 + 0.0011i
   0.0029 - 0.0051i   0.0008 + 0.0431i


s_dc(:,:,75) =

  -0.0523 + 0.0074i   0.0034 + 0.0003i
   0.0003 - 0.0048i   0.0178 + 0.0402i


s_dc(:,:,76) =

  -0.0480 + 0.0235i   0.0042 - 0.0022i
  -0.0008 - 0.0036i   0.0306 + 0.0286i


s_dc(:,:,77) =
```

```
  -0.0372 + 0.0348i   0.0031 - 0.0049i
  -0.0018 - 0.0037i   0.0358 + 0.0133i


s_dc(:,:,78) =

  -0.0248 + 0.0400i  -0.0004 - 0.0053i
  -0.0029 - 0.0027i   0.0336 - 0.0025i


s_dc(:,:,79) =

  -0.0130 + 0.0405i  -0.0035 - 0.0050i
  -0.0037 - 0.0013i   0.0238 - 0.0145i


s_dc(:,:,80) =

  -0.0035 + 0.0372i  -0.0052 - 0.0026i
  -0.0038 - 0.0001i   0.0104 - 0.0190i


s_dc(:,:,81) =

   0.0028 + 0.0325i  -0.0051 + 0.0006i
  -0.0038 + 0.0015i  -0.0016 - 0.0159i


s_dc(:,:,82) =

   0.0067 + 0.0277i  -0.0045 + 0.0029i
  -0.0032 + 0.0025i  -0.0084 - 0.0075i


s_dc(:,:,83) =

   0.0094 + 0.0236i  -0.0028 + 0.0042i
  -0.0020 + 0.0037i  -0.0085 + 0.0015i


s_dc(:,:,84) =

   0.0115 + 0.0200i  -0.0007 + 0.0048i
  -0.0010 + 0.0046i  -0.0037 + 0.0072i
```

```
s_dc(:,:,85) =

   0.0136 + 0.0160i   0.0009 + 0.0049i
   0.0013 + 0.0055i   0.0024 + 0.0081i


s_dc(:,:,86) =

   0.0153 + 0.0114i   0.0029 + 0.0040i
   0.0041 + 0.0048i   0.0067 + 0.0053i


s_dc(:,:,87) =

   0.0152 + 0.0067i   0.0042 + 0.0025i
   0.0061 + 0.0022i   0.0084 + 0.0011i


s_dc(:,:,88) =

   0.0142 + 0.0024i   0.0053 + 0.0004i
   0.0065 - 0.0009i   0.0069 - 0.0033i


s_dc(:,:,89) =

   0.0126 - 0.0011i   0.0043 - 0.0022i
   0.0052 - 0.0040i   0.0030 - 0.0055i


s_dc(:,:,90) =

   0.0104 - 0.0042i   0.0033 - 0.0035i
   0.0024 - 0.0059i  -0.0011 - 0.0052i


s_dc(:,:,91) =

   0.0073 - 0.0063i   0.0014 - 0.0047i
  -0.0001 - 0.0065i  -0.0040 - 0.0026i
```

```
s_dc(:,:,92) =

   0.0046 - 0.0067i  -0.0007 - 0.0050i
  -0.0033 - 0.0054i  -0.0048 + 0.0003i


s_dc(:,:,93) =

   0.0031 - 0.0067i  -0.0026 - 0.0037i
  -0.0046 - 0.0033i  -0.0044 + 0.0028i


s_dc(:,:,94) =

   0.0016 - 0.0071i  -0.0036 - 0.0022i
  -0.0054 - 0.0009i  -0.0028 + 0.0038i


s_dc(:,:,95) =

   0.0002 - 0.0078i  -0.0034 - 0.0008i
  -0.0048 + 0.0010i  -0.0014 + 0.0035i


s_dc(:,:,96) =

  -0.0019 - 0.0083i  -0.0037 - 0.0000i
  -0.0042 + 0.0020i  -0.0013 + 0.0015i


s_dc(:,:,97) =

  -0.0048 - 0.0087i  -0.0043 + 0.0011i
  -0.0042 + 0.0035i  -0.0038 - 0.0001i


s_dc(:,:,98) =

  -0.0072 - 0.0070i  -0.0044 + 0.0040i
  -0.0029 + 0.0066i  -0.0073 + 0.0013i


s_dc(:,:,99) =
```

```
  -0.0078 - 0.0062i  -0.0014 + 0.0056i
   0.0012 + 0.0080i  -0.0088 + 0.0035i


s_dc(:,:,100) =

  -0.0098 - 0.0067i   0.0012 + 0.0058i
   0.0047 + 0.0063i  -0.0102 + 0.0061i


s_dc(:,:,101) =

  -0.0134 - 0.0060i   0.0037 + 0.0045i
   0.0075 + 0.0035i  -0.0103 + 0.0102i


s_dc(:,:,102) =

  -0.0167 - 0.0039i   0.0058 + 0.0026i
   0.0078 - 0.0007i  -0.0088 + 0.0142i


s_dc(:,:,103) =

  -0.0199 - 0.0005i   0.0057 - 0.0007i
   0.0063 - 0.0044i  -0.0051 + 0.0178i


s_dc(:,:,104) =

  -0.0218 + 0.0051i   0.0048 - 0.0027i
   0.0024 - 0.0060i   0.0002 + 0.0194i


s_dc(:,:,105) =

  -0.0203 + 0.0119i   0.0035 - 0.0043i
   0.0004 - 0.0059i   0.0054 + 0.0177i


s_dc(:,:,106) =

  -0.0150 + 0.0164i   0.0016 - 0.0055i
  -0.0009 - 0.0058i   0.0086 + 0.0148i
```

```
s_dc(:,:,107) =

  -0.0093 + 0.0169i  -0.0016 - 0.0054i
  -0.0037 - 0.0050i   0.0100 + 0.0108i


s_dc(:,:,108) =

  -0.0060 + 0.0155i  -0.0035 - 0.0040i
  -0.0057 - 0.0030i   0.0086 + 0.0071i


s_dc(:,:,109) =

  -0.0048 + 0.0140i  -0.0042 - 0.0020i
  -0.0061 - 0.0004i   0.0052 + 0.0058i


s_dc(:,:,110) =

  -0.0048 + 0.0142i  -0.0044 - 0.0005i
  -0.0057 + 0.0020i   0.0018 + 0.0076i


s_dc(:,:,111) =

  -0.0046 + 0.0170i  -0.0043 + 0.0007i
  -0.0047 + 0.0042i   0.0006 + 0.0120i


s_dc(:,:,112) =

  -0.0016 + 0.0212i  -0.0037 + 0.0021i
  -0.0026 + 0.0053i   0.0030 + 0.0172i


s_dc(:,:,113) =

   0.0049 + 0.0246i  -0.0036 + 0.0038i
  -0.0009 + 0.0062i   0.0090 + 0.0206i
```

```
s_dc(:,:,114) =

   0.0138 + 0.0249i  -0.0022 + 0.0057i
   0.0023 + 0.0063i   0.0174 + 0.0200i


s_dc(:,:,115) =

   0.0234 + 0.0209i   0.0011 + 0.0065i
   0.0052 + 0.0054i   0.0252 + 0.0139i


s_dc(:,:,116) =

   0.0310 + 0.0117i   0.0048 + 0.0057i
   0.0078 + 0.0029i   0.0290 + 0.0037i


s_dc(:,:,117) =

   0.0333 + 0.0001i   0.0076 + 0.0025i
   0.0091 - 0.0019i   0.0274 - 0.0071i


s_dc(:,:,118) =

   0.0309 - 0.0111i   0.0076 - 0.0019i
   0.0067 - 0.0067i   0.0211 - 0.0166i


s_dc(:,:,119) =

   0.0250 - 0.0211i   0.0052 - 0.0051i
   0.0018 - 0.0091i   0.0109 - 0.0227i


s_dc(:,:,120) =

   0.0156 - 0.0285i   0.0024 - 0.0062i
  -0.0026 - 0.0080i  -0.0011 - 0.0234i


s_dc(:,:,121) =
```

```
   0.0040 - 0.0314i  -0.0009 - 0.0060i
  -0.0059 - 0.0054i  -0.0120 - 0.0186i


s_dc(:,:,122) =

  -0.0066 - 0.0300i  -0.0031 - 0.0046i
  -0.0073 - 0.0012i  -0.0190 - 0.0096i


s_dc(:,:,123) =

  -0.0150 - 0.0259i  -0.0038 - 0.0026i
  -0.0058 + 0.0021i  -0.0213 + 0.0004i


s_dc(:,:,124) =

  -0.0210 - 0.0204i  -0.0043 - 0.0011i
  -0.0033 + 0.0039i  -0.0189 + 0.0096i


s_dc(:,:,125) =

  -0.0248 - 0.0142i  -0.0039 + 0.0002i
  -0.0012 + 0.0035i  -0.0135 + 0.0156i


s_dc(:,:,126) =

  -0.0264 - 0.0079i  -0.0043 + 0.0006i
  -0.0008 + 0.0025i  -0.0080 + 0.0175i


s_dc(:,:,127) =

  -0.0258 - 0.0025i  -0.0046 + 0.0026i
  -0.0009 + 0.0031i  -0.0047 + 0.0177i


s_dc(:,:,128) =

  -0.0235 + 0.0013i  -0.0029 + 0.0047i
   0.0005 + 0.0043i  -0.0028 + 0.0175i
```

```
s_dc(:,:,129) =

  -0.0214 + 0.0015i  -0.0008 + 0.0056i
   0.0029 + 0.0044i  -0.0030 + 0.0178i


s_dc(:,:,130) =

  -0.0229 + 0.0005i   0.0017 + 0.0056i
   0.0051 + 0.0024i  -0.0040 + 0.0212i


s_dc(:,:,131) =

  -0.0279 + 0.0016i   0.0040 + 0.0052i
   0.0056 - 0.0004i  -0.0026 + 0.0275i


s_dc(:,:,132) =

  -0.0334 + 0.0070i   0.0070 + 0.0029i
   0.0050 - 0.0035i   0.0038 + 0.0340i


s_dc(:,:,133) =

  -0.0379 + 0.0158i   0.0079 - 0.0007i
   0.0021 - 0.0055i   0.0142 + 0.0373i


s_dc(:,:,134) =

  -0.0393 + 0.0289i   0.0071 - 0.0051i
  -0.0013 - 0.0050i   0.0265 + 0.0358i


s_dc(:,:,135) =

  -0.0346 + 0.0435i   0.0040 - 0.0078i
  -0.0035 - 0.0034i   0.0386 + 0.0285i
```

```
s_dc(:,:,136) =

  -0.0232 + 0.0564i  -0.0006 - 0.0088i
  -0.0044 - 0.0003i   0.0476 + 0.0149i


s_dc(:,:,137) =

  -0.0075 + 0.0636i  -0.0050 - 0.0075i
  -0.0032 + 0.0023i   0.0490 - 0.0029i


s_dc(:,:,138) =

   0.0094 + 0.0643i  -0.0077 - 0.0040i
  -0.0006 + 0.0027i   0.0409 - 0.0196i


s_dc(:,:,139) =

   0.0242 + 0.0594i  -0.0084 + 0.0002i
   0.0013 + 0.0014i   0.0257 - 0.0298i


s_dc(:,:,140) =

   0.0353 + 0.0510i  -0.0069 + 0.0042i
   0.0015 - 0.0009i   0.0081 - 0.0304i


s_dc(:,:,141) =

   0.0431 + 0.0411i  -0.0034 + 0.0063i
  -0.0005 - 0.0024i  -0.0059 - 0.0225i


s_dc(:,:,142) =

   0.0475 + 0.0305i   0.0004 + 0.0062i
  -0.0027 - 0.0019i  -0.0129 - 0.0096i


s_dc(:,:,143) =
```

```
   0.0496 + 0.0209i   0.0022 + 0.0049i
  -0.0046 + 0.0003i  -0.0117 + 0.0036i


s_dc(:,:,144) =

   0.0502 + 0.0116i   0.0032 + 0.0033i
  -0.0037 + 0.0037i  -0.0041 + 0.0127i


s_dc(:,:,145) =

   0.0498 + 0.0022i   0.0041 + 0.0016i
  -0.0011 + 0.0057i   0.0059 + 0.0148i


s_dc(:,:,146) =

   0.0468 - 0.0071i   0.0043 + 0.0002i
   0.0026 + 0.0060i   0.0130 + 0.0109i


s_dc(:,:,147) =

   0.0409 - 0.0153i   0.0043 - 0.0016i
   0.0058 + 0.0032i   0.0153 + 0.0055i


s_dc(:,:,148) =

   0.0328 - 0.0194i   0.0029 - 0.0032i
   0.0061 - 0.0008i   0.0144 + 0.0010i


s_dc(:,:,149) =

   0.0262 - 0.0195i   0.0010 - 0.0034i
   0.0044 - 0.0031i   0.0112 - 0.0011i


s_dc(:,:,150) =

   0.0225 - 0.0178i  -0.0001 - 0.0026i
   0.0025 - 0.0038i   0.0082 + 0.0000i
```

```
s_dc(:,:,151) =

   0.0210 - 0.0163i   0.0001 - 0.0023i
   0.0014 - 0.0040i   0.0080 + 0.0028i


s_dc(:,:,152) =

   0.0218 - 0.0160i  -0.0004 - 0.0030i
   0.0001 - 0.0048i   0.0109 + 0.0044i


s_dc(:,:,153) =

   0.0224 - 0.0182i  -0.0013 - 0.0033i
  -0.0013 - 0.0052i   0.0148 + 0.0031i


s_dc(:,:,154) =

   0.0216 - 0.0224i  -0.0029 - 0.0036i
  -0.0042 - 0.0051i   0.0180 - 0.0016i


s_dc(:,:,155) =

   0.0178 - 0.0273i  -0.0056 - 0.0018i
  -0.0073 - 0.0030i   0.0171 - 0.0084i


s_dc(:,:,156) =

   0.0112 - 0.0304i  -0.0069 + 0.0014i
  -0.0088 + 0.0010i   0.0123 - 0.0147i


s_dc(:,:,157) =

   0.0038 - 0.0305i  -0.0055 + 0.0052i
  -0.0074 + 0.0065i   0.0048 - 0.0174i
```

```
s_dc(:,:,158) =

  -0.0033 - 0.0292i  -0.0019 + 0.0073i
  -0.0027 + 0.0095i  -0.0038 - 0.0162i


s_dc(:,:,159) =

  -0.0106 - 0.0258i   0.0023 + 0.0070i
   0.0026 + 0.0093i  -0.0110 - 0.0106i


s_dc(:,:,160) =

  -0.0174 - 0.0199i   0.0054 + 0.0049i
   0.0062 + 0.0065i  -0.0145 - 0.0018i


s_dc(:,:,161) =

  -0.0217 - 0.0109i   0.0069 + 0.0021i
   0.0077 + 0.0028i  -0.0127 + 0.0071i


s_dc(:,:,162) =

  -0.0217 - 0.0008i   0.0075 - 0.0013i
   0.0084 - 0.0003i  -0.0073 + 0.0126i


s_dc(:,:,163) =

  -0.0177 + 0.0076i   0.0059 - 0.0051i
   0.0076 - 0.0042i  -0.0019 + 0.0149i


s_dc(:,:,164) =

  -0.0113 + 0.0130i   0.0024 - 0.0066i
   0.0043 - 0.0073i   0.0029 + 0.0154i


s_dc(:,:,165) =
```

```
  -0.0046 + 0.0155i  -0.0014 - 0.0066i
   0.0004 - 0.0080i   0.0073 + 0.0145i


s_dc(:,:,166) =

   0.0018 + 0.0158i  -0.0037 - 0.0046i
  -0.0034 - 0.0069i   0.0107 + 0.0120i


s_dc(:,:,167) =

   0.0068 + 0.0144i  -0.0046 - 0.0021i
  -0.0055 - 0.0042i   0.0121 + 0.0085i


s_dc(:,:,168) =

   0.0106 + 0.0124i  -0.0045 - 0.0002i
  -0.0056 - 0.0017i   0.0108 + 0.0057i


s_dc(:,:,169) =

   0.0131 + 0.0107i  -0.0036 + 0.0010i
  -0.0056 - 0.0001i   0.0080 + 0.0047i


s_dc(:,:,170) =

   0.0153 + 0.0100i  -0.0030 + 0.0021i
  -0.0058 + 0.0018i   0.0057 + 0.0065i


s_dc(:,:,171) =

   0.0183 + 0.0099i  -0.0026 + 0.0030i
  -0.0051 + 0.0039i   0.0050 + 0.0104i


s_dc(:,:,172) =

   0.0232 + 0.0092i  -0.0016 + 0.0041i
  -0.0036 + 0.0059i   0.0073 + 0.0152i
```

```
s_dc(:,:,173) =

   0.0296 + 0.0064i   0.0004 + 0.0051i
  -0.0010 + 0.0078i   0.0136 + 0.0190i


s_dc(:,:,174) =

   0.0365 - 0.0002i   0.0028 + 0.0049i
   0.0031 + 0.0085i   0.0224 + 0.0187i


s_dc(:,:,175) =

   0.0406 - 0.0116i   0.0054 + 0.0029i
   0.0074 + 0.0064i   0.0311 + 0.0133i


s_dc(:,:,176) =

   0.0390 - 0.0261i   0.0067 - 0.0008i
   0.0104 + 0.0020i   0.0360 + 0.0031i


s_dc(:,:,177) =

   0.0295 - 0.0397i   0.0052 - 0.0044i
   0.0104 - 0.0045i   0.0357 - 0.0080i


s_dc(:,:,178) =

   0.0142 - 0.0469i   0.0014 - 0.0066i
   0.0063 - 0.0097i   0.0304 - 0.0177i


s_dc(:,:,179) =

  -0.0014 - 0.0469i  -0.0027 - 0.0056i
  -0.0006 - 0.0114i   0.0214 - 0.0236i
```

```
s_dc(:,:,180) =

  -0.0144 - 0.0415i  -0.0053 - 0.0022i
  -0.0061 - 0.0080i   0.0116 - 0.0247i


s_dc(:,:,181) =

  -0.0232 - 0.0335i  -0.0047 + 0.0012i
  -0.0084 - 0.0030i   0.0028 - 0.0216i


s_dc(:,:,182) =

  -0.0286 - 0.0248i  -0.0021 + 0.0036i
  -0.0074 + 0.0017i  -0.0032 - 0.0155i


s_dc(:,:,183) =

  -0.0318 - 0.0166i   0.0005 + 0.0037i
  -0.0041 + 0.0043i  -0.0058 - 0.0082i


s_dc(:,:,184) =

  -0.0340 - 0.0084i   0.0027 + 0.0017i
  -0.0005 + 0.0040i  -0.0052 - 0.0007i


s_dc(:,:,185) =

  -0.0348 + 0.0013i   0.0029 - 0.0009i
   0.0008 + 0.0019i  -0.0011 + 0.0053i


s_dc(:,:,186) =

  -0.0321 + 0.0118i   0.0016 - 0.0029i
   0.0002 + 0.0007i   0.0053 + 0.0085i


s_dc(:,:,187) =
```

```
  -0.0254 + 0.0212i  -0.0011 - 0.0034i
  -0.0009 + 0.0005i   0.0128 + 0.0080i


s_dc(:,:,188) =

  -0.0159 + 0.0272i  -0.0033 - 0.0021i
  -0.0015 + 0.0012i   0.0185 + 0.0032i


s_dc(:,:,189) =

  -0.0057 + 0.0294i  -0.0047 + 0.0006i
  -0.0016 + 0.0032i   0.0209 - 0.0042i


s_dc(:,:,190) =

   0.0030 + 0.0282i  -0.0040 + 0.0036i
   0.0000 + 0.0052i   0.0191 - 0.0111i


s_dc(:,:,191) =

   0.0097 + 0.0264i  -0.0009 + 0.0064i
   0.0038 + 0.0057i   0.0148 - 0.0166i


s_dc(:,:,192) =

   0.0158 + 0.0239i   0.0035 + 0.0056i
   0.0077 + 0.0030i   0.0085 - 0.0197i


s_dc(:,:,193) =

   0.0217 + 0.0198i   0.0060 + 0.0025i
   0.0088 - 0.0025i   0.0017 - 0.0196i


s_dc(:,:,194) =

   0.0250 + 0.0140i   0.0059 - 0.0013i
   0.0057 - 0.0074i  -0.0039 - 0.0163i
```

```
s_dc(:,:,195) =

   0.0257 + 0.0082i   0.0043 - 0.0036i
   0.0009 - 0.0095i  -0.0068 - 0.0115i


s_dc(:,:,196) =

   0.0241 + 0.0041i   0.0018 - 0.0051i
  -0.0040 - 0.0079i  -0.0075 - 0.0072i


s_dc(:,:,197) =

   0.0220 + 0.0023i  -0.0006 - 0.0050i
  -0.0071 - 0.0046i  -0.0072 - 0.0040i


s_dc(:,:,198) =

   0.0205 + 0.0024i  -0.0025 - 0.0037i
  -0.0076 - 0.0006i  -0.0065 - 0.0015i


s_dc(:,:,199) =

   0.0204 + 0.0034i  -0.0041 - 0.0028i
  -0.0065 + 0.0026i  -0.0067 + 0.0012i


s_dc(:,:,200) =

   0.0220 + 0.0049i  -0.0048 - 0.0009i
  -0.0045 + 0.0046i  -0.0060 + 0.0050i


s_dc(:,:,201) =

   0.0254 + 0.0052i  -0.0045 + 0.0012i
  -0.0021 + 0.0054i  -0.0037 + 0.0094i
```

```
s_dc(:,:,202) =

   0.0299 + 0.0042i  -0.0037 + 0.0032i
   0.0001 + 0.0060i   0.0011 + 0.0128i


s_dc(:,:,203) =

   0.0352 + 0.0018i  -0.0023 + 0.0049i
   0.0020 + 0.0056i   0.0079 + 0.0135i


s_dc(:,:,204) =

   0.0409 - 0.0036i   0.0005 + 0.0051i
   0.0046 + 0.0041i   0.0145 + 0.0101i


s_dc(:,:,205) =

   0.0452 - 0.0124i   0.0033 + 0.0049i
   0.0061 + 0.0020i   0.0181 + 0.0032i


s_dc(:,:,206) =

   0.0451 - 0.0233i   0.0052 + 0.0022i
   0.0065 - 0.0012i   0.0166 - 0.0040i


s_dc(:,:,207) =

   0.0404 - 0.0329i   0.0051 - 0.0012i
   0.0052 - 0.0044i   0.0117 - 0.0086i


s_dc(:,:,208) =

   0.0330 - 0.0392i   0.0030 - 0.0035i
   0.0018 - 0.0059i   0.0060 - 0.0095i


s_dc(:,:,209) =
```

```
   0.0255 - 0.0423i   0.0009 - 0.0034i
  -0.0016 - 0.0052i   0.0004 - 0.0070i


s_dc(:,:,210) =

   0.0186 - 0.0429i  -0.0006 - 0.0027i
  -0.0030 - 0.0031i  -0.0023 - 0.0019i


s_dc(:,:,211) =

   0.0131 - 0.0424i  -0.0011 - 0.0019i
  -0.0031 - 0.0013i  -0.0011 + 0.0037i


s_dc(:,:,212) =

   0.0083 - 0.0418i  -0.0012 - 0.0013i
  -0.0024 - 0.0007i   0.0027 + 0.0074i


s_dc(:,:,213) =

   0.0033 - 0.0414i  -0.0016 - 0.0012i
  -0.0020 - 0.0006i   0.0079 + 0.0082i


s_dc(:,:,214) =

  -0.0029 - 0.0397i  -0.0016 - 0.0009i
  -0.0027 - 0.0007i   0.0129 + 0.0065i


s_dc(:,:,215) =

  -0.0084 - 0.0358i  -0.0023 - 0.0004i
  -0.0038 + 0.0003i   0.0159 + 0.0023i


s_dc(:,:,216) =

  -0.0124 - 0.0303i  -0.0024 + 0.0006i
  -0.0039 + 0.0024i   0.0165 - 0.0024i
```

```
s_dc(:,:,217) =

  -0.0144 - 0.0242i  -0.0015 + 0.0016i
  -0.0020 + 0.0047i   0.0145 - 0.0064i


s_dc(:,:,218) =

  -0.0147 - 0.0185i  -0.0010 + 0.0015i
   0.0001 + 0.0053i   0.0110 - 0.0086i


s_dc(:,:,219) =

  -0.0137 - 0.0138i  -0.0009 + 0.0020i
   0.0023 + 0.0045i   0.0076 - 0.0084i


s_dc(:,:,220) =

  -0.0121 - 0.0098i  -0.0001 + 0.0030i
   0.0037 + 0.0037i   0.0054 - 0.0070i


s_dc(:,:,221) =

  -0.0100 - 0.0054i   0.0018 + 0.0029i
   0.0055 + 0.0025i   0.0045 - 0.0058i


s_dc(:,:,222) =

  -0.0055 - 0.0013i   0.0031 + 0.0014i
   0.0066 - 0.0007i   0.0038 - 0.0051i


s_dc(:,:,223) =

   0.0005 + 0.0003i   0.0034 - 0.0004i
   0.0061 - 0.0042i   0.0026 - 0.0045i
```

```
s_dc(:,:,224) =

   0.0063 - 0.0001i   0.0024 - 0.0021i
   0.0034 - 0.0070i   0.0021 - 0.0031i


s_dc(:,:,225) =

   0.0112 - 0.0021i   0.0012 - 0.0024i
  -0.0005 - 0.0077i   0.0020 - 0.0022i


s_dc(:,:,226) =

   0.0155 - 0.0053i  -0.0000 - 0.0025i
  -0.0041 - 0.0065i   0.0017 - 0.0015i


s_dc(:,:,227) =

   0.0182 - 0.0093i  -0.0007 - 0.0021i
  -0.0066 - 0.0036i   0.0014 - 0.0011i


s_dc(:,:,228) =

   0.0202 - 0.0129i  -0.0013 - 0.0015i
  -0.0069 - 0.0004i   0.0009 - 0.0006i


s_dc(:,:,229) =

   0.0212 - 0.0166i  -0.0015 - 0.0008i
  -0.0064 + 0.0027i  -0.0001 + 0.0003i


s_dc(:,:,230) =

   0.0219 - 0.0204i  -0.0015 - 0.0002i
  -0.0042 + 0.0047i  -0.0014 + 0.0021i


s_dc(:,:,231) =
```

```
   0.0221 - 0.0241i  -0.0015 - 0.0000i
  -0.0020 + 0.0059i  -0.0024 + 0.0053i


s_dc(:,:,232) =

   0.0220 - 0.0279i  -0.0015 + 0.0005i
   0.0004 + 0.0059i  -0.0015 + 0.0100i


s_dc(:,:,233) =

   0.0217 - 0.0325i  -0.0018 + 0.0012i
   0.0026 + 0.0057i   0.0022 + 0.0149i


s_dc(:,:,234) =

   0.0202 - 0.0380i  -0.0007 + 0.0021i
   0.0051 + 0.0046i   0.0084 + 0.0171i


s_dc(:,:,235) =

   0.0165 - 0.0447i   0.0001 + 0.0021i
   0.0074 + 0.0019i   0.0151 + 0.0161i


s_dc(:,:,236) =

   0.0092 - 0.0503i   0.0007 + 0.0014i
   0.0078 - 0.0020i   0.0201 + 0.0123i


s_dc(:,:,237) =

  -0.0004 - 0.0528i   0.0009 + 0.0010i
   0.0059 - 0.0059i   0.0225 + 0.0067i


s_dc(:,:,238) =

  -0.0100 - 0.0516i   0.0013 + 0.0008i
   0.0019 - 0.0080i   0.0214 + 0.0013i
```

```
s_dc(:,:,239) =

  -0.0183 - 0.0468i   0.0014 + 0.0005i
  -0.0027 - 0.0074i   0.0178 - 0.0016i


s_dc(:,:,240) =

  -0.0232 - 0.0404i   0.0019 - 0.0000i
  -0.0054 - 0.0049i   0.0140 - 0.0016i


s_dc(:,:,241) =

  -0.0252 - 0.0346i   0.0018 - 0.0010i
  -0.0063 - 0.0014i   0.0119 + 0.0006i


s_dc(:,:,242) =

  -0.0258 - 0.0298i   0.0013 - 0.0012i
  -0.0056 + 0.0016i   0.0115 + 0.0033i


s_dc(:,:,243) =

  -0.0255 - 0.0262i   0.0011 - 0.0016i
  -0.0038 + 0.0034i   0.0134 + 0.0053i


s_dc(:,:,244) =

  -0.0261 - 0.0225i   0.0004 - 0.0020i
  -0.0019 + 0.0041i   0.0162 + 0.0063i


s_dc(:,:,245) =

  -0.0256 - 0.0188i  -0.0002 - 0.0023i
   0.0002 + 0.0043i   0.0191 + 0.0053i
```

```
s_dc(:,:,246) =

  -0.0244 - 0.0149i  -0.0016 - 0.0024i
   0.0014 + 0.0034i   0.0211 + 0.0033i


s_dc(:,:,247) =

  -0.0218 - 0.0117i  -0.0029 - 0.0016i
   0.0026 + 0.0022i   0.0226 + 0.0011i


s_dc(:,:,248) =

  -0.0186 - 0.0100i  -0.0036 - 0.0003i
   0.0033 + 0.0011i   0.0227 - 0.0010i


s_dc(:,:,249) =

  -0.0160 - 0.0099i  -0.0037 + 0.0018i
   0.0030 - 0.0006i   0.0229 - 0.0024i


s_dc(:,:,250) =

  -0.0148 - 0.0106i  -0.0023 + 0.0034i
   0.0021 - 0.0013i   0.0233 - 0.0036i


s_dc(:,:,251) =

  -0.0151 - 0.0105i  -0.0002 + 0.0039i
   0.0015 - 0.0017i   0.0240 - 0.0049i


s_dc(:,:,252) =

  -0.0157 - 0.0085i   0.0017 + 0.0033i
   0.0010 - 0.0020i   0.0244 - 0.0067i


s_dc(:,:,253) =
```

```
  -0.0149 - 0.0054i   0.0023 + 0.0018i
   0.0000 - 0.0024i   0.0246 - 0.0085i


s_dc(:,:,254) =

  -0.0121 - 0.0026i   0.0022 + 0.0009i
  -0.0009 - 0.0021i   0.0243 - 0.0107i


s_dc(:,:,255) =

  -0.0076 - 0.0008i   0.0019 + 0.0004i
  -0.0016 - 0.0014i   0.0237 - 0.0132i


s_dc(:,:,256) =

  -0.0025 - 0.0013i   0.0022 + 0.0003i
  -0.0020 - 0.0003i   0.0220 - 0.0158i


s_dc(:,:,257) =

   0.0023 - 0.0045i   0.0027 - 0.0002i
  -0.0017 + 0.0004i   0.0196 - 0.0180i


s_dc(:,:,258) =

   0.0057 - 0.0094i   0.0035 - 0.0014i
  -0.0009 + 0.0010i   0.0164 - 0.0194i


s_dc(:,:,259) =

   0.0065 - 0.0158i   0.0033 - 0.0034i
  -0.0000 + 0.0007i   0.0126 - 0.0200i


s_dc(:,:,260) =

   0.0044 - 0.0222i   0.0015 - 0.0056i
   0.0001 - 0.0001i   0.0079 - 0.0192i
```

```
s_dc(:,:,261) =

  -0.0003 - 0.0268i  -0.0014 - 0.0065i
  -0.0006 - 0.0005i   0.0038 - 0.0163i


s_dc(:,:,262) =

  -0.0060 - 0.0287i  -0.0048 - 0.0054i
  -0.0018 + 0.0000i   0.0014 - 0.0112i


s_dc(:,:,263) =

  -0.0111 - 0.0283i  -0.0073 - 0.0026i
  -0.0017 + 0.0015i   0.0017 - 0.0051i


s_dc(:,:,264) =

  -0.0152 - 0.0267i  -0.0082 + 0.0012i
  -0.0010 + 0.0026i   0.0050 - 0.0002i


s_dc(:,:,265) =

  -0.0186 - 0.0242i  -0.0073 + 0.0052i
   0.0008 + 0.0034i   0.0105 + 0.0029i


s_dc(:,:,266) =

  -0.0212 - 0.0210i  -0.0042 + 0.0087i
   0.0027 + 0.0027i   0.0167 + 0.0028i


s_dc(:,:,267) =

  -0.0232 - 0.0172i   0.0006 + 0.0100i
   0.0040 + 0.0010i   0.0226 + 0.0001i
```

```
s_dc(:,:,268) =

  -0.0239 - 0.0124i   0.0056 + 0.0087i
   0.0041 - 0.0012i   0.0261 - 0.0050i


s_dc(:,:,269) =

  -0.0229 - 0.0073i   0.0091 + 0.0051i
   0.0031 - 0.0030i   0.0271 - 0.0101i


s_dc(:,:,270) =

  -0.0198 - 0.0023i   0.0104 - 0.0000i
   0.0016 - 0.0044i   0.0266 - 0.0145i


s_dc(:,:,271) =

  -0.0150 + 0.0008i   0.0092 - 0.0051i
  -0.0007 - 0.0049i   0.0250 - 0.0181i


s_dc(:,:,272) =

  -0.0100 + 0.0014i   0.0057 - 0.0085i
  -0.0030 - 0.0043i   0.0235 - 0.0212i


s_dc(:,:,273) =

  -0.0054 + 0.0002i   0.0010 - 0.0100i
  -0.0048 - 0.0024i   0.0207 - 0.0233i


s_dc(:,:,274) =

  -0.0018 - 0.0025i  -0.0036 - 0.0092i
  -0.0057 + 0.0003i   0.0184 - 0.0259i


s_dc(:,:,275) =
```

```
  -0.0003 - 0.0059i  -0.0071 - 0.0062i
  -0.0049 + 0.0033i   0.0149 - 0.0274i


s_dc(:,:,276) =

  -0.0003 - 0.0090i  -0.0086 - 0.0022i
  -0.0023 + 0.0053i   0.0106 - 0.0282i


s_dc(:,:,277) =

  -0.0015 - 0.0112i  -0.0081 + 0.0015i
   0.0008 + 0.0057i   0.0065 - 0.0275i


s_dc(:,:,278) =

  -0.0033 - 0.0118i  -0.0066 + 0.0043i
   0.0034 + 0.0042i   0.0030 - 0.0256i


s_dc(:,:,279) =

  -0.0041 - 0.0111i  -0.0044 + 0.0064i
   0.0044 + 0.0018i   0.0004 - 0.0233i


s_dc(:,:,280) =

  -0.0042 - 0.0105i  -0.0014 + 0.0075i
   0.0043 - 0.0005i  -0.0014 - 0.0208i


s_dc(:,:,281) =

  -0.0031 - 0.0098i   0.0018 + 0.0075i
   0.0034 - 0.0022i  -0.0029 - 0.0182i


s_dc(:,:,282) =

  -0.0011 - 0.0095i   0.0048 + 0.0058i
   0.0018 - 0.0034i  -0.0033 - 0.0150i
```

```
s_dc(:,:,283) =

   0.0014 - 0.0101i   0.0066 + 0.0031i
  -0.0000 - 0.0036i  -0.0025 - 0.0117i


s_dc(:,:,284) =

   0.0041 - 0.0118i   0.0068 + 0.0002i
  -0.0016 - 0.0027i  -0.0007 - 0.0093i


s_dc(:,:,285) =

   0.0066 - 0.0149i   0.0060 - 0.0020i
  -0.0023 - 0.0013i   0.0018 - 0.0080i


s_dc(:,:,286) =

   0.0082 - 0.0189i   0.0048 - 0.0038i
  -0.0023 - 0.0001i   0.0037 - 0.0077i


s_dc(:,:,287) =

   0.0085 - 0.0240i   0.0033 - 0.0053i
  -0.0015 + 0.0008i   0.0056 - 0.0082i


s_dc(:,:,288) =

   0.0069 - 0.0295i   0.0013 - 0.0064i
  -0.0006 + 0.0009i   0.0066 - 0.0091i


s_dc(:,:,289) =

   0.0032 - 0.0345i  -0.0014 - 0.0068i
  -0.0005 + 0.0003i   0.0064 - 0.0094i
```

```
s_dc(:,:,290) =

  -0.0028 - 0.0375i  -0.0041 - 0.0062i
  -0.0010 + 0.0001i   0.0062 - 0.0086i


s_dc(:,:,291) =

  -0.0093 - 0.0371i  -0.0068 - 0.0039i
  -0.0015 + 0.0010i   0.0068 - 0.0068i


s_dc(:,:,292) =

  -0.0131 - 0.0337i  -0.0080 - 0.0007i
  -0.0008 + 0.0021i   0.0094 - 0.0051i


s_dc(:,:,293) =

  -0.0130 - 0.0316i  -0.0077 + 0.0030i
   0.0006 + 0.0025i   0.0132 - 0.0046i


s_dc(:,:,294) =

  -0.0129 - 0.0328i  -0.0058 + 0.0061i
   0.0020 + 0.0018i   0.0174 - 0.0069i


s_dc(:,:,295) =

  -0.0152 - 0.0353i  -0.0025 + 0.0081i
   0.0027 + 0.0005i   0.0203 - 0.0113i


s_dc(:,:,296) =

  -0.0190 - 0.0371i   0.0016 + 0.0083i
   0.0024 - 0.0012i   0.0209 - 0.0172i


s_dc(:,:,297) =
```

```
  -0.0245 - 0.0380i   0.0051 + 0.0065i
   0.0012 - 0.0022i   0.0189 - 0.0227i


s_dc(:,:,298) =

  -0.0312 - 0.0367i   0.0068 + 0.0033i
  -0.0003 - 0.0023i   0.0137 - 0.0267i


s_dc(:,:,299) =

  -0.0373 - 0.0327i   0.0066 + 0.0003i
  -0.0015 - 0.0013i   0.0076 - 0.0270i


s_dc(:,:,300) =

  -0.0413 - 0.0267i   0.0057 - 0.0015i
  -0.0017 + 0.0000i   0.0034 - 0.0249i


s_dc(:,:,301) =

  -0.0436 - 0.0205i   0.0050 - 0.0027i
  -0.0011 + 0.0012i   0.0017 - 0.0210i


s_dc(:,:,302) =

  -0.0446 - 0.0146i   0.0043 - 0.0041i
   0.0001 + 0.0013i   0.0023 - 0.0177i


s_dc(:,:,303) =

  -0.0448 - 0.0083i   0.0031 - 0.0059i
   0.0012 + 0.0009i   0.0047 - 0.0159i


s_dc(:,:,304) =

  -0.0437 - 0.0018i   0.0009 - 0.0074i
   0.0016 - 0.0002i   0.0075 - 0.0162i
```

```
s_dc(:,:,305) =

  -0.0411 + 0.0053i  -0.0022 - 0.0083i
   0.0013 - 0.0016i   0.0101 - 0.0183i


s_dc(:,:,306) =

  -0.0356 + 0.0117i  -0.0064 - 0.0075i
   0.0002 - 0.0025i   0.0119 - 0.0216i


s_dc(:,:,307) =

  -0.0277 + 0.0158i  -0.0101 - 0.0042i
  -0.0017 - 0.0023i   0.0111 - 0.0268i


s_dc(:,:,308) =

  -0.0197 + 0.0164i  -0.0118 + 0.0012i
  -0.0031 - 0.0009i   0.0078 - 0.0313i


s_dc(:,:,309) =

  -0.0134 + 0.0143i  -0.0101 + 0.0071i
  -0.0033 + 0.0013i   0.0020 - 0.0344i


s_dc(:,:,310) =

  -0.0088 + 0.0112i  -0.0058 + 0.0111i
  -0.0019 + 0.0029i  -0.0051 - 0.0347i


s_dc(:,:,311) =

  -0.0058 + 0.0076i  -0.0001 + 0.0125i
  -0.0000 + 0.0033i  -0.0116 - 0.0317i
```

```
s_dc(:,:,312) =

  -0.0045 + 0.0045i   0.0055 + 0.0109i
   0.0013 + 0.0029i  -0.0162 - 0.0265i


s_dc(:,:,313) =

  -0.0035 + 0.0024i   0.0095 + 0.0073i
   0.0022 + 0.0019i  -0.0178 - 0.0206i


s_dc(:,:,314) =

  -0.0027 + 0.0003i   0.0115 + 0.0030i
   0.0028 + 0.0009i  -0.0175 - 0.0158i


s_dc(:,:,315) =

  -0.0015 - 0.0018i   0.0120 - 0.0020i
   0.0032 + 0.0000i  -0.0164 - 0.0126i


s_dc(:,:,316) =

  -0.0012 - 0.0051i   0.0105 - 0.0068i
   0.0033 - 0.0013i  -0.0156 - 0.0106i


s_dc(:,:,317) =

  -0.0020 - 0.0081i   0.0070 - 0.0109i
   0.0028 - 0.0029i  -0.0160 - 0.0090i


s_dc(:,:,318) =

  -0.0037 - 0.0108i   0.0016 - 0.0135i
   0.0013 - 0.0044i  -0.0169 - 0.0068i


s_dc(:,:,319) =
```

```
  -0.0065 - 0.0131i  -0.0046 - 0.0131i
  -0.0012 - 0.0050i  -0.0181 - 0.0031i


s_dc(:,:,320) =

  -0.0101 - 0.0141i  -0.0103 - 0.0099i
  -0.0037 - 0.0040i  -0.0183 + 0.0022i


s_dc(:,:,321) =

  -0.0137 - 0.0134i  -0.0138 - 0.0042i
  -0.0054 - 0.0016i  -0.0159 + 0.0084i


s_dc(:,:,322) =

  -0.0166 - 0.0116i  -0.0142 + 0.0024i
  -0.0053 + 0.0010i  -0.0107 + 0.0138i


s_dc(:,:,323) =

  -0.0186 - 0.0093i  -0.0116 + 0.0085i
  -0.0041 + 0.0033i  -0.0036 + 0.0170i


s_dc(:,:,324) =

  -0.0200 - 0.0066i  -0.0066 + 0.0127i
  -0.0024 + 0.0045i   0.0043 + 0.0175i


s_dc(:,:,325) =

  -0.0211 - 0.0030i  -0.0003 + 0.0141i
  -0.0004 + 0.0050i   0.0118 + 0.0152i


s_dc(:,:,326) =

  -0.0206 + 0.0014i   0.0061 + 0.0126i
   0.0018 + 0.0050i   0.0177 + 0.0111i
```

```
s_dc(:,:,327) =

  -0.0179 + 0.0058i   0.0106 + 0.0081i
   0.0040 + 0.0040i   0.0215 + 0.0052i


s_dc(:,:,328) =

  -0.0133 + 0.0086i   0.0123 + 0.0026i
   0.0055 + 0.0018i   0.0230 - 0.0002i


s_dc(:,:,329) =

  -0.0082 + 0.0091i   0.0115 - 0.0023i
   0.0061 - 0.0010i   0.0228 - 0.0046i


s_dc(:,:,330) =

  -0.0041 + 0.0080i   0.0093 - 0.0061i
   0.0049 - 0.0037i   0.0221 - 0.0080i


s_dc(:,:,331) =

  -0.0009 + 0.0062i   0.0063 - 0.0090i
   0.0026 - 0.0054i   0.0215 - 0.0103i


s_dc(:,:,332) =

   0.0017 + 0.0042i   0.0025 - 0.0108i
  -0.0003 - 0.0060i   0.0217 - 0.0126i


s_dc(:,:,333) =

   0.0038 + 0.0022i  -0.0019 - 0.0109i
  -0.0029 - 0.0051i   0.0218 - 0.0149i
```

```
s_dc(:,:,334) =

   0.0046 + 0.0004i  -0.0061 - 0.0096i
  -0.0047 - 0.0030i   0.0213 - 0.0176i


s_dc(:,:,335) =

   0.0061 - 0.0013i  -0.0095 - 0.0067i
  -0.0053 - 0.0006i   0.0208 - 0.0211i


s_dc(:,:,336) =

   0.0077 - 0.0028i  -0.0118 - 0.0024i
  -0.0045 + 0.0015i   0.0198 - 0.0250i


s_dc(:,:,337) =

   0.0094 - 0.0046i  -0.0122 + 0.0027i
  -0.0033 + 0.0031i   0.0171 - 0.0294i


s_dc(:,:,338) =

   0.0116 - 0.0070i  -0.0104 + 0.0078i
  -0.0018 + 0.0037i   0.0128 - 0.0334i


s_dc(:,:,339) =

   0.0138 - 0.0107i  -0.0063 + 0.0121i
  -0.0005 + 0.0039i   0.0071 - 0.0361i


s_dc(:,:,340) =

   0.0154 - 0.0156i  -0.0002 + 0.0140i
   0.0009 + 0.0039i   0.0004 - 0.0363i


s_dc(:,:,341) =
```

```
   0.0149 - 0.0217i   0.0063 + 0.0126i
   0.0023 + 0.0034i  -0.0062 - 0.0345i


s_dc(:,:,342) =

   0.0127 - 0.0274i   0.0113 + 0.0082i
   0.0035 + 0.0023i  -0.0107 - 0.0301i


s_dc(:,:,343) =

   0.0089 - 0.0321i   0.0133 + 0.0023i
   0.0042 + 0.0008i  -0.0125 - 0.0256i


s_dc(:,:,344) =

   0.0049 - 0.0355i   0.0123 - 0.0032i
   0.0043 - 0.0010i  -0.0123 - 0.0222i


s_dc(:,:,345) =

   0.0010 - 0.0384i   0.0098 - 0.0076i
   0.0036 - 0.0023i  -0.0118 - 0.0207i


s_dc(:,:,346) =

  -0.0028 - 0.0415i   0.0061 - 0.0105i
   0.0027 - 0.0035i  -0.0121 - 0.0200i


s_dc(:,:,347) =

  -0.0075 - 0.0450i   0.0016 - 0.0121i
   0.0013 - 0.0045i  -0.0142 - 0.0197i


s_dc(:,:,348) =

  -0.0138 - 0.0489i  -0.0034 - 0.0120i
  -0.0005 - 0.0053i  -0.0177 - 0.0181i
```

```
s_dc(:,:,349) =

  -0.0221 - 0.0513i  -0.0083 - 0.0099i
  -0.0031 - 0.0049i  -0.0215 - 0.0138i


s_dc(:,:,350) =

  -0.0320 - 0.0515i  -0.0120 - 0.0055i
  -0.0052 - 0.0032i  -0.0236 - 0.0068i


s_dc(:,:,351) =

  -0.0420 - 0.0488i  -0.0134 + 0.0002i
  -0.0062 - 0.0006i  -0.0227 + 0.0012i


s_dc(:,:,352) =

  -0.0512 - 0.0435i  -0.0119 + 0.0060i
  -0.0060 + 0.0022i  -0.0185 + 0.0089i


s_dc(:,:,353) =

  -0.0593 - 0.0362i  -0.0081 + 0.0104i
  -0.0047 + 0.0046i  -0.0114 + 0.0143i


s_dc(:,:,354) =

  -0.0659 - 0.0265i  -0.0027 + 0.0126i
  -0.0025 + 0.0065i  -0.0031 + 0.0164i


s_dc(:,:,355) =

  -0.0697 - 0.0145i   0.0030 + 0.0124i
   0.0006 + 0.0073i   0.0049 + 0.0154i
```

```
s_dc(:,:,356) =

  -0.0697 - 0.0015i   0.0077 + 0.0095i
   0.0042 + 0.0065i   0.0114 + 0.0122i


s_dc(:,:,357) =

  -0.0654 + 0.0099i   0.0103 + 0.0052i
   0.0071 + 0.0037i   0.0157 + 0.0078i


s_dc(:,:,358) =

  -0.0589 + 0.0190i   0.0109 + 0.0008i
   0.0079 - 0.0002i   0.0186 + 0.0030i


s_dc(:,:,359) =

  -0.0515 + 0.0253i   0.0099 - 0.0031i
   0.0066 - 0.0038i   0.0202 - 0.0015i


s_dc(:,:,360) =

  -0.0440 + 0.0297i   0.0080 - 0.0062i
   0.0038 - 0.0059i   0.0205 - 0.0060i


s_dc(:,:,361) =

  -0.0370 + 0.0319i   0.0055 - 0.0086i
   0.0008 - 0.0065i   0.0207 - 0.0095i


s_dc(:,:,362) =

  -0.0310 + 0.0335i   0.0022 - 0.0101i
  -0.0017 - 0.0059i   0.0208 - 0.0125i


s_dc(:,:,363) =
```

```
  -0.0255 + 0.0349i  -0.0016 - 0.0108i
  -0.0036 - 0.0045i   0.0211 - 0.0152i


s_dc(:,:,364) =

  -0.0197 + 0.0362i  -0.0057 - 0.0101i
  -0.0048 - 0.0028i   0.0214 - 0.0188i


s_dc(:,:,365) =

  -0.0133 + 0.0375i  -0.0098 - 0.0077i
  -0.0051 - 0.0009i   0.0211 - 0.0232i


s_dc(:,:,366) =

  -0.0051 + 0.0377i  -0.0131 - 0.0035i
  -0.0051 + 0.0008i   0.0196 - 0.0282i


s_dc(:,:,367) =

   0.0040 + 0.0357i  -0.0146 + 0.0027i
  -0.0048 + 0.0025i   0.0162 - 0.0331i


s_dc(:,:,368) =

   0.0124 + 0.0300i  -0.0126 + 0.0097i
  -0.0040 + 0.0043i   0.0105 - 0.0375i


s_dc(:,:,369) =

   0.0183 + 0.0219i  -0.0070 + 0.0153i
  -0.0021 + 0.0059i   0.0024 - 0.0389i


s_dc(:,:,370) =

   0.0219 + 0.0127i   0.0010 + 0.0172i
   0.0005 + 0.0069i  -0.0056 - 0.0371i
```

```
s_dc(:,:,371) =

   0.0220 + 0.0029i   0.0090 + 0.0149i
   0.0037 + 0.0065i  -0.0116 - 0.0321i


s_dc(:,:,372) =

   0.0189 - 0.0059i   0.0144 + 0.0091i
   0.0066 + 0.0044i  -0.0142 - 0.0263i


s_dc(:,:,373) =

   0.0142 - 0.0119i   0.0167 + 0.0016i
   0.0081 + 0.0011i  -0.0141 - 0.0217i


s_dc(:,:,374) =

   0.0092 - 0.0163i   0.0155 - 0.0055i
   0.0079 - 0.0027i  -0.0132 - 0.0200i


s_dc(:,:,375) =

   0.0045 - 0.0196i   0.0117 - 0.0112i
   0.0060 - 0.0060i  -0.0135 - 0.0197i


s_dc(:,:,376) =

  -0.0009 - 0.0222i   0.0060 - 0.0152i
   0.0031 - 0.0082i  -0.0157 - 0.0199i


s_dc(:,:,377) =

  -0.0070 - 0.0235i  -0.0010 - 0.0165i
  -0.0010 - 0.0088i  -0.0196 - 0.0184i
```

```
s_dc(:,:,378) =

  -0.0132 - 0.0232i  -0.0081 - 0.0147i
  -0.0048 - 0.0076i  -0.0237 - 0.0144i


s_dc(:,:,379) =

  -0.0189 - 0.0212i  -0.0141 - 0.0096i
  -0.0078 - 0.0047i  -0.0261 - 0.0078i


s_dc(:,:,380) =

  -0.0238 - 0.0183i  -0.0167 - 0.0022i
  -0.0091 - 0.0010i  -0.0253 - 0.0000i


s_dc(:,:,381) =

  -0.0280 - 0.0140i  -0.0157 + 0.0054i
  -0.0085 + 0.0031i  -0.0212 + 0.0071i


s_dc(:,:,382) =

  -0.0311 - 0.0086i  -0.0114 + 0.0116i
  -0.0064 + 0.0065i  -0.0149 + 0.0120i


s_dc(:,:,383) =

  -0.0325 - 0.0021i  -0.0048 + 0.0151i
  -0.0030 + 0.0088i  -0.0080 + 0.0140i


s_dc(:,:,384) =

  -0.0314 + 0.0047i   0.0023 + 0.0147i
   0.0014 + 0.0093i  -0.0025 + 0.0138i


s_dc(:,:,385) =
```

```
  -0.0279 + 0.0104i   0.0078 + 0.0114i
   0.0054 + 0.0075i   0.0019 + 0.0128i


s_dc(:,:,386) =

  -0.0230 + 0.0142i   0.0109 + 0.0068i
   0.0079 + 0.0041i   0.0049 + 0.0121i


s_dc(:,:,387) =

  -0.0175 + 0.0163i   0.0120 + 0.0021i
   0.0085 + 0.0001i   0.0076 + 0.0121i


s_dc(:,:,388) =

  -0.0123 + 0.0163i   0.0113 - 0.0024i
   0.0074 - 0.0033i   0.0110 + 0.0123i


s_dc(:,:,389) =

  -0.0078 + 0.0145i   0.0093 - 0.0060i
   0.0051 - 0.0058i   0.0152 + 0.0121i


s_dc(:,:,390) =

  -0.0055 + 0.0120i   0.0064 - 0.0089i
   0.0022 - 0.0070i   0.0196 + 0.0113i


s_dc(:,:,391) =

  -0.0046 + 0.0103i   0.0028 - 0.0108i
  -0.0008 - 0.0070i   0.0247 + 0.0085i


s_dc(:,:,392) =

  -0.0037 + 0.0098i  -0.0012 - 0.0113i
  -0.0033 - 0.0059i   0.0293 + 0.0041i
```

```
s_dc(:,:,393) =

  -0.0026 + 0.0096i  -0.0056 - 0.0103i
  -0.0051 - 0.0040i   0.0333 - 0.0013i


s_dc(:,:,394) =

  -0.0010 + 0.0100i  -0.0094 - 0.0077i
  -0.0060 - 0.0018i   0.0358 - 0.0085i


s_dc(:,:,395) =

   0.0026 + 0.0096i  -0.0122 - 0.0032i
  -0.0061 + 0.0005i   0.0361 - 0.0162i


s_dc(:,:,396) =

   0.0065 + 0.0081i  -0.0131 + 0.0023i
  -0.0055 + 0.0026i   0.0341 - 0.0244i


s_dc(:,:,397) =

   0.0104 + 0.0043i  -0.0110 + 0.0079i
  -0.0042 + 0.0046i   0.0296 - 0.0316i


s_dc(:,:,398) =

   0.0122 - 0.0007i  -0.0064 + 0.0120i
  -0.0022 + 0.0062i   0.0233 - 0.0365i


s_dc(:,:,399) =

   0.0120 - 0.0054i  -0.0006 + 0.0133i
   0.0007 + 0.0067i   0.0162 - 0.0385i
```

```
s_dc(:,:,400) =

   0.0108 - 0.0094i   0.0048 + 0.0121i
   0.0035 + 0.0058i   0.0097 - 0.0377i


s_dc(:,:,401) =

   0.0092 - 0.0129i   0.0091 + 0.0089i
   0.0053 + 0.0038i   0.0054 - 0.0355i


s_dc(:,:,402) =

   0.0074 - 0.0161i   0.0117 + 0.0047i
   0.0063 + 0.0013i   0.0027 - 0.0328i


s_dc(:,:,403) =

   0.0050 - 0.0189i   0.0127 - 0.0002i
   0.0064 - 0.0011i   0.0011 - 0.0310i


s_dc(:,:,404) =

   0.0017 - 0.0213i   0.0117 - 0.0052i
   0.0057 - 0.0036i  -0.0002 - 0.0298i


s_dc(:,:,405) =

  -0.0017 - 0.0229i   0.0091 - 0.0096i
   0.0040 - 0.0055i  -0.0025 - 0.0289i


s_dc(:,:,406) =

  -0.0053 - 0.0241i   0.0044 - 0.0130i
   0.0016 - 0.0071i  -0.0055 - 0.0274i


s_dc(:,:,407) =
```

```
  -0.0093 - 0.0247i  -0.0015 - 0.0142i
  -0.0015 - 0.0075i  -0.0087 - 0.0248i


s_dc(:,:,408) =

  -0.0141 - 0.0248i  -0.0079 - 0.0126i
  -0.0047 - 0.0064i  -0.0114 - 0.0204i


s_dc(:,:,409) =

  -0.0189 - 0.0234i  -0.0129 - 0.0080i
  -0.0074 - 0.0040i  -0.0124 - 0.0144i


s_dc(:,:,410) =

  -0.0231 - 0.0207i  -0.0153 - 0.0012i
  -0.0087 - 0.0004i  -0.0103 - 0.0083i


s_dc(:,:,411) =

  -0.0271 - 0.0174i  -0.0140 + 0.0061i
  -0.0081 + 0.0035i  -0.0056 - 0.0039i


s_dc(:,:,412) =

  -0.0309 - 0.0130i  -0.0092 + 0.0115i
  -0.0057 + 0.0070i   0.0004 - 0.0025i


s_dc(:,:,413) =

  -0.0333 - 0.0070i  -0.0029 + 0.0135i
  -0.0019 + 0.0088i   0.0058 - 0.0038i


s_dc(:,:,414) =

  -0.0333 - 0.0005i   0.0030 + 0.0124i
   0.0023 + 0.0084i   0.0093 - 0.0064i
```

```
s_dc(:,:,415) =

  -0.0315 + 0.0051i   0.0071 + 0.0094i
   0.0055 + 0.0064i   0.0116 - 0.0092i


s_dc(:,:,416) =

  -0.0287 + 0.0097i   0.0095 + 0.0057i
   0.0073 + 0.0033i   0.0132 - 0.0120i


s_dc(:,:,417) =

  -0.0255 + 0.0134i   0.0103 + 0.0019i
   0.0077 + 0.0001i   0.0143 - 0.0144i


s_dc(:,:,418) =

  -0.0217 + 0.0165i   0.0099 - 0.0018i
   0.0068 - 0.0029i   0.0153 - 0.0167i


s_dc(:,:,419) =

  -0.0177 + 0.0185i   0.0086 - 0.0051i
   0.0050 - 0.0052i   0.0159 - 0.0190i


s_dc(:,:,420) =

  -0.0136 + 0.0197i   0.0062 - 0.0078i
   0.0025 - 0.0067i   0.0165 - 0.0212i


s_dc(:,:,421) =

  -0.0098 + 0.0201i   0.0032 - 0.0097i
  -0.0003 - 0.0070i   0.0170 - 0.0234i
```

```
s_dc(:,:,422) =

  -0.0067 + 0.0204i  -0.0007 - 0.0105i
  -0.0029 - 0.0062i   0.0174 - 0.0264i


s_dc(:,:,423) =

  -0.0039 + 0.0212i  -0.0047 - 0.0098i
  -0.0049 - 0.0046i   0.0171 - 0.0297i


s_dc(:,:,424) =

  -0.0002 + 0.0225i  -0.0086 - 0.0075i
  -0.0063 - 0.0025i   0.0156 - 0.0336i


s_dc(:,:,425) =

   0.0044 + 0.0236i  -0.0113 - 0.0035i
  -0.0069 - 0.0002i   0.0128 - 0.0374i


s_dc(:,:,426) =

   0.0105 + 0.0236i  -0.0121 + 0.0015i
  -0.0067 + 0.0024i   0.0083 - 0.0408i


s_dc(:,:,427) =

   0.0169 + 0.0221i  -0.0106 + 0.0064i
  -0.0054 + 0.0050i   0.0029 - 0.0420i


s_dc(:,:,428) =

   0.0237 + 0.0186i  -0.0071 + 0.0102i
  -0.0034 + 0.0069i  -0.0027 - 0.0415i


s_dc(:,:,429) =
```

```
   0.0295 + 0.0131i  -0.0025 + 0.0125i
  -0.0004 + 0.0083i  -0.0076 - 0.0391i


s_dc(:,:,430) =

   0.0337 + 0.0065i   0.0027 + 0.0125i
   0.0032 + 0.0084i  -0.0110 - 0.0358i


s_dc(:,:,431) =

   0.0362 - 0.0014i   0.0075 + 0.0107i
   0.0069 + 0.0067i  -0.0127 - 0.0320i


s_dc(:,:,432) =

   0.0365 - 0.0088i   0.0116 + 0.0071i
   0.0098 + 0.0031i  -0.0126 - 0.0287i


s_dc(:,:,433) =

   0.0356 - 0.0152i   0.0140 + 0.0018i
   0.0107 - 0.0020i  -0.0116 - 0.0261i


s_dc(:,:,434) =

   0.0352 - 0.0201i   0.0141 - 0.0042i
   0.0085 - 0.0068i  -0.0102 - 0.0250i


s_dc(:,:,435) =

   0.0361 - 0.0265i   0.0112 - 0.0102i
   0.0043 - 0.0098i  -0.0092 - 0.0249i


s_dc(:,:,436) =

   0.0358 - 0.0354i   0.0060 - 0.0145i
  -0.0004 - 0.0101i  -0.0086 - 0.0253i
```

```
s_dc(:,:,437) =

   0.0323 - 0.0459i  -0.0009 - 0.0159i
  -0.0043 - 0.0087i  -0.0090 - 0.0254i


s_dc(:,:,438) =

   0.0247 - 0.0558i  -0.0078 - 0.0140i
  -0.0072 - 0.0061i  -0.0095 - 0.0251i


s_dc(:,:,439) =

   0.0143 - 0.0637i  -0.0130 - 0.0093i
  -0.0090 - 0.0028i  -0.0098 - 0.0245i


s_dc(:,:,440) =

   0.0011 - 0.0681i  -0.0156 - 0.0028i
  -0.0093 + 0.0011i  -0.0096 - 0.0240i


s_dc(:,:,441) =

  -0.0130 - 0.0691i  -0.0151 + 0.0038i
  -0.0082 + 0.0047i  -0.0095 - 0.0241i


s_dc(:,:,442) =

  -0.0270 - 0.0654i  -0.0121 + 0.0095i
  -0.0055 + 0.0079i  -0.0099 - 0.0245i


s_dc(:,:,443) =

  -0.0397 - 0.0584i  -0.0074 + 0.0133i
  -0.0018 + 0.0095i  -0.0113 - 0.0247i
```

```
s_dc(:,:,444) =

  -0.0499 - 0.0491i  -0.0016 + 0.0152i
   0.0025 + 0.0095i  -0.0127 - 0.0240i


s_dc(:,:,445) =

  -0.0578 - 0.0371i   0.0045 + 0.0150i
   0.0064 + 0.0076i  -0.0138 - 0.0226i


s_dc(:,:,446) =

  -0.0625 - 0.0231i   0.0104 + 0.0124i
   0.0092 + 0.0041i  -0.0145 - 0.0210i


s_dc(:,:,447) =

  -0.0628 - 0.0080i   0.0150 + 0.0073i
   0.0102 - 0.0003i  -0.0146 - 0.0197i


s_dc(:,:,448) =

  -0.0590 + 0.0057i   0.0174 + 0.0005i
   0.0089 - 0.0047i  -0.0141 - 0.0185i


s_dc(:,:,449) =

  -0.0520 + 0.0170i   0.0165 - 0.0071i
   0.0058 - 0.0080i  -0.0132 - 0.0178i


s_dc(:,:,450) =

  -0.0435 + 0.0265i   0.0122 - 0.0141i
   0.0017 - 0.0094i  -0.0124 - 0.0182i


s_dc(:,:,451) =
```

```
  -0.0327 + 0.0339i   0.0049 - 0.0184i
  -0.0022 - 0.0089i  -0.0125 - 0.0191i


s_dc(:,:,452) =

  -0.0208 + 0.0382i  -0.0037 - 0.0190i
  -0.0054 - 0.0070i  -0.0137 - 0.0194i


s_dc(:,:,453) =

  -0.0084 + 0.0392i  -0.0117 - 0.0156i
  -0.0074 - 0.0042i  -0.0152 - 0.0190i


s_dc(:,:,454) =

   0.0030 + 0.0374i  -0.0172 - 0.0088i
  -0.0083 - 0.0011i  -0.0163 - 0.0176i


s_dc(:,:,455) =

   0.0134 + 0.0341i  -0.0193 - 0.0006i
  -0.0080 + 0.0022i  -0.0163 - 0.0159i


s_dc(:,:,456) =

   0.0233 + 0.0285i  -0.0177 + 0.0075i
  -0.0066 + 0.0049i  -0.0157 - 0.0147i


s_dc(:,:,457) =

   0.0322 + 0.0217i  -0.0128 + 0.0142i
  -0.0044 + 0.0072i  -0.0156 - 0.0143i


s_dc(:,:,458) =

   0.0395 + 0.0121i  -0.0058 + 0.0184i
  -0.0016 + 0.0084i  -0.0157 - 0.0136i
```

```
s_dc(:,:,459) =

   0.0451 + 0.0015i   0.0025 + 0.0191i
   0.0018 + 0.0087i  -0.0161 - 0.0121i


s_dc(:,:,460) =

   0.0488 - 0.0109i   0.0103 + 0.0164i
   0.0051 + 0.0078i  -0.0159 - 0.0100i


s_dc(:,:,461) =

   0.0483 - 0.0246i   0.0164 + 0.0106i
   0.0081 + 0.0052i  -0.0146 - 0.0079i


s_dc(:,:,462) =

   0.0448 - 0.0382i   0.0195 + 0.0027i
   0.0100 + 0.0016i  -0.0123 - 0.0068i


s_dc(:,:,463) =

   0.0376 - 0.0504i   0.0189 - 0.0058i
   0.0102 - 0.0028i  -0.0104 - 0.0071i


s_dc(:,:,464) =

   0.0272 - 0.0599i   0.0147 - 0.0134i
   0.0084 - 0.0072i  -0.0092 - 0.0079i


s_dc(:,:,465) =

   0.0150 - 0.0671i   0.0076 - 0.0185i
   0.0047 - 0.0105i  -0.0093 - 0.0086i
```

```
s_dc(:,:,466) =

   0.0018 - 0.0711i  -0.0012 - 0.0200i
  -0.0004 - 0.0118i  -0.0100 - 0.0086i


s_dc(:,:,467) =

  -0.0124 - 0.0716i  -0.0097 - 0.0175i
  -0.0056 - 0.0108i  -0.0105 - 0.0078i


s_dc(:,:,468) =

  -0.0259 - 0.0690i  -0.0163 - 0.0114i
  -0.0100 - 0.0072i  -0.0101 - 0.0065i


s_dc(:,:,469) =

  -0.0379 - 0.0630i  -0.0193 - 0.0031i
  -0.0124 - 0.0022i  -0.0090 - 0.0054i


s_dc(:,:,470) =

  -0.0484 - 0.0547i  -0.0184 + 0.0055i
  -0.0122 + 0.0034i  -0.0075 - 0.0053i


s_dc(:,:,471) =

  -0.0566 - 0.0443i  -0.0138 + 0.0125i
  -0.0095 + 0.0085i  -0.0062 - 0.0059i


s_dc(:,:,472) =

  -0.0619 - 0.0321i  -0.0071 + 0.0167i
  -0.0049 + 0.0118i  -0.0054 - 0.0068i


s_dc(:,:,473) =
```

```
  -0.0639 - 0.0192i   0.0005 + 0.0175i
   0.0008 + 0.0127i  -0.0052 - 0.0076i


s_dc(:,:,474) =

  -0.0629 - 0.0065i   0.0073 + 0.0154i
   0.0061 + 0.0110i  -0.0051 - 0.0085i


s_dc(:,:,475) =

  -0.0587 + 0.0051i   0.0125 + 0.0108i
   0.0101 + 0.0071i  -0.0050 - 0.0089i


s_dc(:,:,476) =

  -0.0517 + 0.0150i   0.0154 + 0.0049i
   0.0119 + 0.0023i  -0.0050 - 0.0092i


s_dc(:,:,477) =

  -0.0433 + 0.0227i   0.0158 - 0.0015i
   0.0116 - 0.0029i  -0.0049 - 0.0095i


s_dc(:,:,478) =

  -0.0340 + 0.0277i   0.0139 - 0.0074i
   0.0093 - 0.0072i  -0.0047 - 0.0096i


s_dc(:,:,479) =

  -0.0244 + 0.0307i   0.0100 - 0.0125i
   0.0056 - 0.0103i  -0.0044 - 0.0097i


s_dc(:,:,480) =

  -0.0150 + 0.0316i   0.0043 - 0.0156i
   0.0010 - 0.0117i  -0.0043 - 0.0096i
```

```
s_dc(:,:,481) =

  -0.0058 + 0.0310i  -0.0024 - 0.0161i
  -0.0039 - 0.0111i  -0.0041 - 0.0095i


s_dc(:,:,482) =

   0.0031 + 0.0284i  -0.0090 - 0.0139i
  -0.0081 - 0.0088i  -0.0034 - 0.0093i


s_dc(:,:,483) =

   0.0113 + 0.0241i  -0.0140 - 0.0091i
  -0.0109 - 0.0049i  -0.0026 - 0.0094i


s_dc(:,:,484) =

   0.0179 + 0.0178i  -0.0168 - 0.0025i
  -0.0121 - 0.0001i  -0.0020 - 0.0105i


s_dc(:,:,485) =

   0.0231 + 0.0106i  -0.0164 + 0.0048i
  -0.0112 + 0.0049i  -0.0023 - 0.0117i


s_dc(:,:,486) =

   0.0260 + 0.0030i  -0.0129 + 0.0113i
  -0.0085 + 0.0092i  -0.0035 - 0.0128i


s_dc(:,:,487) =

   0.0275 - 0.0051i  -0.0070 + 0.0159i
  -0.0039 + 0.0122i  -0.0056 - 0.0130i
```

```
s_dc(:,:,488) =

   0.0285 - 0.0130i   0.0004 + 0.0175i
   0.0016 + 0.0128i  -0.0077 - 0.0120i


s_dc(:,:,489) =

   0.0275 - 0.0213i   0.0078 + 0.0158i
   0.0068 + 0.0113i  -0.0092 - 0.0100i


s_dc(:,:,490) =

   0.0249 - 0.0301i   0.0138 + 0.0109i
   0.0113 + 0.0074i  -0.0101 - 0.0074i


s_dc(:,:,491) =

   0.0198 - 0.0384i   0.0173 + 0.0039i
   0.0136 + 0.0019i  -0.0094 - 0.0045i


s_dc(:,:,492) =

   0.0122 - 0.0453i   0.0171 - 0.0038i
   0.0132 - 0.0041i  -0.0074 - 0.0018i


s_dc(:,:,493) =

   0.0029 - 0.0493i   0.0137 - 0.0110i
   0.0104 - 0.0095i  -0.0047 + 0.0001i


s_dc(:,:,494) =

  -0.0066 - 0.0502i   0.0074 - 0.0156i
   0.0054 - 0.0133i  -0.0011 + 0.0010i


s_dc(:,:,495) =
```

```
  -0.0155 - 0.0480i  -0.0002 - 0.0168i
  -0.0011 - 0.0144i   0.0028 + 0.0008i


s_dc(:,:,496) =

  -0.0225 - 0.0441i  -0.0071 - 0.0147i
  -0.0072 - 0.0125i   0.0065 - 0.0009i


s_dc(:,:,497) =

  -0.0273 - 0.0388i  -0.0123 - 0.0098i
  -0.0119 - 0.0082i   0.0095 - 0.0033i


s_dc(:,:,498) =

  -0.0300 - 0.0336i  -0.0147 - 0.0036i
  -0.0143 - 0.0022i   0.0119 - 0.0064i


s_dc(:,:,499) =

  -0.0313 - 0.0290i  -0.0142 + 0.0025i
  -0.0138 + 0.0042i   0.0130 - 0.0096i


s_dc(:,:,500) =

  -0.0316 - 0.0247i  -0.0117 + 0.0076i
  -0.0105 + 0.0096i   0.0134 - 0.0121i


s_dc(:,:,501) =

  -0.0319 - 0.0213i  -0.0074 + 0.0114i
  -0.0055 + 0.0133i   0.0136 - 0.0145i


s_dc(:,:,502) =

  -0.0319 - 0.0180i  -0.0023 + 0.0131i
   0.0008 + 0.0143i   0.0137 - 0.0167i
```

```
s_dc(:,:,503) =

  -0.0316 - 0.0146i   0.0030 + 0.0126i
   0.0069 + 0.0124i   0.0133 - 0.0190i


s_dc(:,:,504) =

  -0.0308 - 0.0113i   0.0076 + 0.0103i
   0.0114 + 0.0082i   0.0129 - 0.0211i


s_dc(:,:,505) =

  -0.0293 - 0.0080i   0.0109 + 0.0063i
   0.0137 + 0.0025i   0.0123 - 0.0238i


s_dc(:,:,506) =

  -0.0274 - 0.0053i   0.0123 + 0.0014i
   0.0131 - 0.0037i   0.0115 - 0.0267i


s_dc(:,:,507) =

  -0.0253 - 0.0031i   0.0118 - 0.0034i
   0.0101 - 0.0087i   0.0097 - 0.0300i


s_dc(:,:,508) =

  -0.0230 - 0.0010i   0.0093 - 0.0077i
   0.0052 - 0.0120i   0.0070 - 0.0332i


s_dc(:,:,509) =

  -0.0201 + 0.0004i   0.0056 - 0.0108i
  -0.0004 - 0.0128i   0.0032 - 0.0355i
```

```
s_dc(:,:,510) =

  -0.0171 + 0.0013i   0.0009 - 0.0119i
  -0.0054 - 0.0113i  -0.0012 - 0.0370i


s_dc(:,:,511) =

  -0.0141 + 0.0016i  -0.0041 - 0.0112i
  -0.0094 - 0.0079i  -0.0052 - 0.0369i


s_dc(:,:,512) =

  -0.0112 + 0.0013i  -0.0081 - 0.0086i
  -0.0116 - 0.0033i  -0.0086 - 0.0361i


s_dc(:,:,513) =

  -0.0086 + 0.0009i  -0.0106 - 0.0045i
  -0.0120 + 0.0019i  -0.0112 - 0.0347i


s_dc(:,:,514) =

  -0.0063 + 0.0004i  -0.0113 + 0.0001i
  -0.0099 + 0.0067i  -0.0132 - 0.0332i


s_dc(:,:,515) =

  -0.0035 + 0.0000i  -0.0102 + 0.0044i
  -0.0061 + 0.0102i  -0.0146 - 0.0320i


s_dc(:,:,516) =

  -0.0000 - 0.0011i  -0.0077 + 0.0078i
  -0.0010 + 0.0115i  -0.0159 - 0.0304i


s_dc(:,:,517) =
```

```
   0.0037 - 0.0029i  -0.0041 + 0.0101i
   0.0037 + 0.0106i  -0.0168 - 0.0290i


s_dc(:,:,518) =

   0.0078 - 0.0060i   0.0001 + 0.0108i
   0.0075 + 0.0078i  -0.0180 - 0.0276i


s_dc(:,:,519) =

   0.0113 - 0.0105i   0.0041 + 0.0100i
   0.0097 + 0.0038i  -0.0188 - 0.0261i


s_dc(:,:,520) =

   0.0138 - 0.0164i   0.0077 + 0.0077i
   0.0099 - 0.0003i  -0.0199 - 0.0241i


s_dc(:,:,521) =

   0.0151 - 0.0229i   0.0101 + 0.0044i
   0.0087 - 0.0040i  -0.0207 - 0.0221i


s_dc(:,:,522) =

   0.0146 - 0.0307i   0.0111 + 0.0002i
   0.0064 - 0.0068i  -0.0212 - 0.0197i


s_dc(:,:,523) =

   0.0125 - 0.0379i   0.0106 - 0.0040i
   0.0033 - 0.0086i  -0.0211 - 0.0174i


s_dc(:,:,524) =

   0.0085 - 0.0446i   0.0086 - 0.0079i
  -0.0002 - 0.0090i  -0.0208 - 0.0154i
```

```
s_dc(:,:,525) =

   0.0032 - 0.0505i   0.0051 - 0.0110i
  -0.0036 - 0.0085i  -0.0202 - 0.0134i


s_dc(:,:,526) =

  -0.0029 - 0.0554i   0.0004 - 0.0127i
  -0.0067 - 0.0067i  -0.0194 - 0.0115i


s_dc(:,:,527) =

  -0.0102 - 0.0594i  -0.0049 - 0.0125i
  -0.0088 - 0.0036i  -0.0179 - 0.0099i


s_dc(:,:,528) =

  -0.0185 - 0.0621i  -0.0101 - 0.0097i
  -0.0097 + 0.0002i  -0.0156 - 0.0083i


s_dc(:,:,529) =

  -0.0276 - 0.0628i  -0.0138 - 0.0048i
  -0.0092 + 0.0042i  -0.0130 - 0.0078i


s_dc(:,:,530) =

  -0.0367 - 0.0619i  -0.0149 + 0.0016i
  -0.0067 + 0.0079i  -0.0099 - 0.0083i


s_dc(:,:,531) =

  -0.0460 - 0.0594i  -0.0131 + 0.0079i
  -0.0030 + 0.0103i  -0.0072 - 0.0102i
```

```
s_dc(:,:,532) =

  -0.0549 - 0.0543i  -0.0087 + 0.0130i
   0.0017 + 0.0108i  -0.0057 - 0.0126i


s_dc(:,:,533) =

  -0.0626 - 0.0473i  -0.0024 + 0.0156i
   0.0060 + 0.0092i  -0.0051 - 0.0154i


s_dc(:,:,534) =

  -0.0686 - 0.0386i   0.0044 + 0.0152i
   0.0095 + 0.0060i  -0.0058 - 0.0181i


s_dc(:,:,535) =

  -0.0719 - 0.0287i   0.0104 + 0.0122i
   0.0112 + 0.0015i  -0.0071 - 0.0203i


s_dc(:,:,536) =

  -0.0729 - 0.0187i   0.0146 + 0.0068i
   0.0108 - 0.0035i  -0.0089 - 0.0220i


s_dc(:,:,537) =

  -0.0712 - 0.0091i   0.0163 + 0.0002i
   0.0084 - 0.0078i  -0.0115 - 0.0229i


s_dc(:,:,538) =

  -0.0679 - 0.0009i   0.0151 - 0.0067i
   0.0043 - 0.0107i  -0.0141 - 0.0232i


s_dc(:,:,539) =
```

```
  -0.0636 + 0.0062i   0.0111 - 0.0125i
  -0.0007 - 0.0115i  -0.0167 - 0.0225i


s_dc(:,:,540) =

  -0.0581 + 0.0120i   0.0049 - 0.0162i
  -0.0056 - 0.0101i  -0.0190 - 0.0211i


s_dc(:,:,541) =

  -0.0518 + 0.0169i  -0.0024 - 0.0169i
  -0.0093 - 0.0066i  -0.0208 - 0.0194i


s_dc(:,:,542) =

  -0.0452 + 0.0205i  -0.0092 - 0.0144i
  -0.0112 - 0.0020i  -0.0222 - 0.0173i


s_dc(:,:,543) =

  -0.0380 + 0.0226i  -0.0144 - 0.0093i
  -0.0108 + 0.0030i  -0.0229 - 0.0154i


s_dc(:,:,544) =

  -0.0304 + 0.0235i  -0.0171 - 0.0026i
  -0.0083 + 0.0070i  -0.0234 - 0.0132i


s_dc(:,:,545) =

  -0.0231 + 0.0229i  -0.0167 + 0.0048i
  -0.0044 + 0.0097i  -0.0236 - 0.0113i


s_dc(:,:,546) =

  -0.0159 + 0.0212i  -0.0133 + 0.0112i
  -0.0001 + 0.0104i  -0.0235 - 0.0090i
```

```
s_dc(:,:,547) =

  -0.0089 + 0.0175i  -0.0077 + 0.0158i
   0.0040 + 0.0093i  -0.0232 - 0.0065i


s_dc(:,:,548) =

  -0.0029 + 0.0122i  -0.0005 + 0.0177i
   0.0072 + 0.0069i  -0.0220 - 0.0036i


s_dc(:,:,549) =

   0.0016 + 0.0054i   0.0069 + 0.0166i
   0.0092 + 0.0035i  -0.0201 - 0.0009i


s_dc(:,:,550) =

   0.0046 - 0.0026i   0.0134 + 0.0125i
   0.0097 - 0.0004i  -0.0171 + 0.0013i


s_dc(:,:,551) =

   0.0055 - 0.0112i   0.0177 + 0.0059i
   0.0089 - 0.0042i  -0.0136 + 0.0029i


s_dc(:,:,552) =

   0.0039 - 0.0200i   0.0189 - 0.0022i
   0.0067 - 0.0074i  -0.0095 + 0.0033i


s_dc(:,:,553) =

  -0.0004 - 0.0282i   0.0165 - 0.0100i
   0.0031 - 0.0096i  -0.0054 + 0.0026i
```

```
s_dc(:,:,554) =

  -0.0064 - 0.0351i   0.0108 - 0.0163i
  -0.0009 - 0.0104i  -0.0019 + 0.0008i


s_dc(:,:,555) =

  -0.0140 - 0.0401i   0.0029 - 0.0196i
  -0.0050 - 0.0091i   0.0011 - 0.0019i


s_dc(:,:,556) =

  -0.0221 - 0.0431i  -0.0057 - 0.0192i
  -0.0085 - 0.0062i   0.0029 - 0.0048i


s_dc(:,:,557) =

  -0.0308 - 0.0439i  -0.0136 - 0.0150i
  -0.0107 - 0.0023i   0.0044 - 0.0079i


s_dc(:,:,558) =

  -0.0392 - 0.0432i  -0.0187 - 0.0079i
  -0.0107 + 0.0024i   0.0056 - 0.0112i


s_dc(:,:,559) =

  -0.0468 - 0.0403i  -0.0202 + 0.0009i
  -0.0088 + 0.0066i   0.0066 - 0.0150i


s_dc(:,:,560) =

  -0.0541 - 0.0362i  -0.0180 + 0.0094i
  -0.0052 + 0.0098i   0.0067 - 0.0195i


s_dc(:,:,561) =
```

```
  -0.0605 - 0.0303i  -0.0123 + 0.0160i
  -0.0007 + 0.0111i   0.0056 - 0.0247i


s_dc(:,:,562) =

  -0.0656 - 0.0229i  -0.0044 + 0.0196i
   0.0041 + 0.0104i   0.0027 - 0.0296i


s_dc(:,:,563) =

  -0.0691 - 0.0142i   0.0042 + 0.0197i
   0.0081 + 0.0077i  -0.0016 - 0.0335i


s_dc(:,:,564) =

  -0.0705 - 0.0046i   0.0121 + 0.0161i
   0.0106 + 0.0036i  -0.0069 - 0.0359i


s_dc(:,:,565) =

  -0.0692 + 0.0052i   0.0178 + 0.0096i
   0.0112 - 0.0012i  -0.0125 - 0.0367i


s_dc(:,:,566) =

  -0.0656 + 0.0142i   0.0201 + 0.0012i
   0.0097 - 0.0061i  -0.0179 - 0.0363i


s_dc(:,:,567) =

  -0.0601 + 0.0219i   0.0186 - 0.0074i
   0.0062 - 0.0096i  -0.0227 - 0.0340i


s_dc(:,:,568) =

  -0.0531 + 0.0279i   0.0138 - 0.0146i
   0.0013 - 0.0113i  -0.0271 - 0.0313i
```

```
s_dc(:,:,569) =

  -0.0456 + 0.0319i   0.0064 - 0.0190i
  -0.0035 - 0.0107i  -0.0306 - 0.0275i


s_dc(:,:,570) =

  -0.0386 + 0.0336i  -0.0019 - 0.0198i
  -0.0077 - 0.0081i  -0.0331 - 0.0233i


s_dc(:,:,571) =

  -0.0316 + 0.0344i  -0.0098 - 0.0173i
  -0.0102 - 0.0042i  -0.0348 - 0.0189i


s_dc(:,:,572) =

  -0.0250 + 0.0342i  -0.0162 - 0.0115i
  -0.0109 + 0.0004i  -0.0354 - 0.0142i


s_dc(:,:,573) =

  -0.0184 + 0.0333i  -0.0196 - 0.0039i
  -0.0096 + 0.0047i  -0.0352 - 0.0100i


s_dc(:,:,574) =

  -0.0119 + 0.0309i  -0.0195 + 0.0046i
  -0.0069 + 0.0080i  -0.0347 - 0.0060i


s_dc(:,:,575) =

  -0.0054 + 0.0273i  -0.0161 + 0.0125i
  -0.0029 + 0.0102i  -0.0337 - 0.0020i
```

```
s_dc(:,:,576) =

  -0.0000 + 0.0220i  -0.0094 + 0.0183i
   0.0015 + 0.0105i  -0.0319 + 0.0023i


s_dc(:,:,577) =

   0.0047 + 0.0153i  -0.0007 + 0.0206i
   0.0054 + 0.0089i  -0.0292 + 0.0064i


s_dc(:,:,578) =

   0.0077 + 0.0077i   0.0082 + 0.0191i
   0.0085 + 0.0059i  -0.0251 + 0.0099i


s_dc(:,:,579) =

   0.0087 - 0.0008i   0.0156 + 0.0139i
   0.0101 + 0.0020i  -0.0198 + 0.0124i


s_dc(:,:,580) =

   0.0078 - 0.0094i   0.0200 + 0.0060i
   0.0101 - 0.0023i  -0.0139 + 0.0135i


s_dc(:,:,581) =

   0.0045 - 0.0175i   0.0206 - 0.0032i
   0.0085 - 0.0061i  -0.0076 + 0.0129i


s_dc(:,:,582) =

  -0.0010 - 0.0246i   0.0171 - 0.0117i
   0.0052 - 0.0092i  -0.0018 + 0.0106i


s_dc(:,:,583) =
```

```
   -0.0085 - 0.0298i   0.0105 - 0.0177i
    0.0011 - 0.0105i   0.0030 + 0.0070i


s_dc(:,:,584) =

   -0.0167 - 0.0324i   0.0021 - 0.0204i
   -0.0034 - 0.0101i   0.0066 + 0.0026i


s_dc(:,:,585) =

   -0.0253 - 0.0326i  -0.0068 - 0.0193i
   -0.0075 - 0.0078i   0.0096 - 0.0024i


s_dc(:,:,586) =

   -0.0328 - 0.0306i  -0.0142 - 0.0145i
   -0.0099 - 0.0040i   0.0108 - 0.0074i


s_dc(:,:,587) =

   -0.0396 - 0.0269i  -0.0189 - 0.0071i
   -0.0107 + 0.0005i   0.0112 - 0.0126i


s_dc(:,:,588) =

   -0.0450 - 0.0221i  -0.0200 + 0.0016i
   -0.0096 + 0.0050i   0.0108 - 0.0181i


s_dc(:,:,589) =

   -0.0493 - 0.0166i  -0.0174 + 0.0097i
   -0.0066 + 0.0084i   0.0091 - 0.0231i


s_dc(:,:,590) =

   -0.0525 - 0.0105i  -0.0116 + 0.0160i
   -0.0024 + 0.0104i   0.0064 - 0.0286i
```

```
s_dc(:,:,591) =

  -0.0546 - 0.0037i  -0.0037 + 0.0191i
   0.0023 + 0.0106i   0.0020 - 0.0333i


s_dc(:,:,592) =

  -0.0555 + 0.0036i   0.0047 + 0.0187i
   0.0066 + 0.0087i  -0.0031 - 0.0367i


s_dc(:,:,593) =

  -0.0545 + 0.0114i   0.0120 + 0.0149i
   0.0095 + 0.0051i  -0.0089 - 0.0389i


s_dc(:,:,594) =

  -0.0514 + 0.0192i   0.0170 + 0.0084i
   0.0108 + 0.0006i  -0.0148 - 0.0401i


s_dc(:,:,595) =

  -0.0461 + 0.0263i   0.0187 + 0.0005i
   0.0101 - 0.0040i  -0.0206 - 0.0398i


s_dc(:,:,596) =

  -0.0387 + 0.0316i   0.0169 - 0.0072i
   0.0074 - 0.0078i  -0.0258 - 0.0386i


s_dc(:,:,597) =

  -0.0300 + 0.0348i   0.0122 - 0.0134i
   0.0034 - 0.0102i  -0.0312 - 0.0368i
```

```
s_dc(:,:,598) =

  -0.0207 + 0.0348i   0.0055 - 0.0170i
  -0.0012 - 0.0108i  -0.0359 - 0.0342i


s_dc(:,:,599) =

  -0.0124 + 0.0325i  -0.0020 - 0.0175i
  -0.0055 - 0.0093i  -0.0408 - 0.0305i


s_dc(:,:,600) =

  -0.0055 + 0.0283i  -0.0089 - 0.0150i
  -0.0088 - 0.0061i  -0.0452 - 0.0261i


s_dc(:,:,601) =

  -0.0005 + 0.0228i  -0.0140 - 0.0102i
  -0.0105 - 0.0018i  -0.0494 - 0.0199i


s_dc(:,:,602) =

   0.0024 + 0.0168i  -0.0170 - 0.0034i
  -0.0104 + 0.0027i  -0.0519 - 0.0126i


s_dc(:,:,603) =

   0.0039 + 0.0114i  -0.0170 + 0.0037i
  -0.0083 + 0.0068i  -0.0531 - 0.0047i


s_dc(:,:,604) =

   0.0045 + 0.0060i  -0.0140 + 0.0105i
  -0.0048 + 0.0095i  -0.0522 + 0.0040i


s_dc(:,:,605) =
```

```
   0.0043 + 0.0012i  -0.0085 + 0.0155i
  -0.0004 + 0.0107i  -0.0494 + 0.0124i


s_dc(:,:,606) =

   0.0036 - 0.0036i  -0.0012 + 0.0177i
   0.0039 + 0.0099i  -0.0446 + 0.0202i


s_dc(:,:,607) =

   0.0021 - 0.0084i   0.0064 + 0.0168i
   0.0076 + 0.0074i  -0.0376 + 0.0268i


s_dc(:,:,608) =

  -0.0005 - 0.0131i   0.0128 + 0.0125i
   0.0100 + 0.0035i  -0.0292 + 0.0312i


s_dc(:,:,609) =

  -0.0042 - 0.0178i   0.0170 + 0.0057i
   0.0105 - 0.0008i  -0.0200 + 0.0334i


s_dc(:,:,610) =

  -0.0094 - 0.0213i   0.0177 - 0.0020i
   0.0093 - 0.0051i  -0.0106 + 0.0332i


s_dc(:,:,611) =

  -0.0160 - 0.0237i   0.0150 - 0.0093i
   0.0063 - 0.0085i  -0.0016 + 0.0311i


s_dc(:,:,612) =

  -0.0228 - 0.0238i   0.0095 - 0.0147i
   0.0022 - 0.0104i   0.0066 + 0.0269i
```

```
s_dc(:,:,613) =

  -0.0293 - 0.0215i   0.0023 - 0.0171i
  -0.0023 - 0.0104i   0.0137 + 0.0212i


s_dc(:,:,614) =

  -0.0348 - 0.0176i  -0.0049 - 0.0163i
  -0.0065 - 0.0084i   0.0196 + 0.0143i


s_dc(:,:,615) =

  -0.0382 - 0.0125i  -0.0111 - 0.0125i
  -0.0095 - 0.0049i   0.0238 + 0.0065i


s_dc(:,:,616) =

  -0.0394 - 0.0070i  -0.0150 - 0.0067i
  -0.0106 - 0.0004i   0.0267 - 0.0020i


s_dc(:,:,617) =

  -0.0390 - 0.0021i  -0.0161 + 0.0000i
  -0.0099 + 0.0040i   0.0278 - 0.0114i


s_dc(:,:,618) =

  -0.0378 + 0.0016i  -0.0145 + 0.0065i
  -0.0071 + 0.0079i   0.0269 - 0.0206i


s_dc(:,:,619) =

  -0.0358 + 0.0047i  -0.0104 + 0.0118i
  -0.0032 + 0.0103i   0.0240 - 0.0296i
```

```
s_dc(:,:,620) =

  -0.0337 + 0.0071i  -0.0047 + 0.0150i
```

References

Fan, W., A. C. W. Lu, L. L. Wai, and B. K. Lok. “Mixed-Mode S-Parameter Characterization of Differential Structures.” Electronic Packaging Technology Conference, pp. 533–537, 2003.

See Also

s2scc | s2scd | s2sdd | s2smm | smm2s

Introduced in R2006a

s2sdd

Convert 4-port, single-ended S-parameters to 2-port, differential-mode S-parameters (S_{dd})

Syntax

```
sdd_params = s2sdd(s_params)
sdd_params = s2sdd(s_params,option)
```

Description

`sdd_params = s2sdd(s_params)` converts the 2*N*-port, single-ended S-parameters, `s_params`, to *N*-port, differential-mode S-parameters, `sdd_params`. `sdd_params` is a complex *N*-by-*N*-by-*M* array that represents *M* *N*-port, differential-mode S-parameters (S_{cd}).

`sdd_params = s2sdd(s_params,option)` converts S-parameters based on the optional `option` argument, which indicates the port-ordering convention of the S-parameters.

Input Arguments

`s_params` — S-parameters
array

S-parameters, specified as a complex 4-by-4-by-*M* array, that represents *M* 4-port S-parameters.

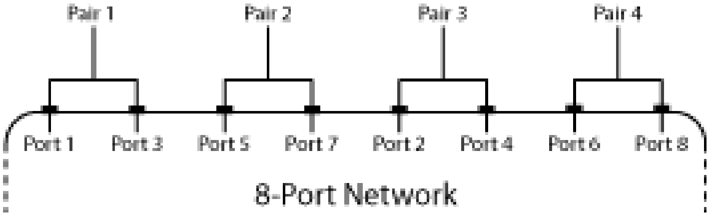
`option` — Port order
1 (default) | 2 | 3

Port order, specified as `1`, `2`, `3`, determines how the function orders the ports:

- `1` — `s2sdd` pairs the odd-numbered ports together first, followed by the even-numbered ports. For example, in a single-ended, 8-port network:

- Ports 1 and 3 become differential-mode pair 1.
- Ports 5 and 7 become differential-mode pair 2.
- Ports 2 and 4 become differential-mode pair 3.
- Ports 6 and 8 become differential-mode pair 4.

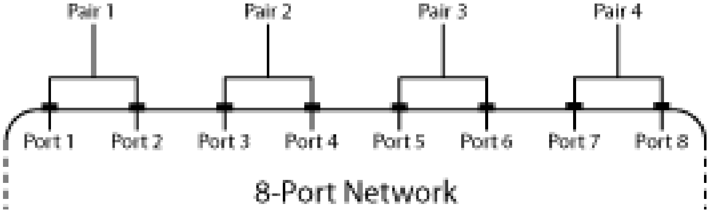
The following figure illustrates this convention for an 8-port device.



- `2` — `s2sdd` pairs the input and output ports in ascending order. For example, in a single-ended, 8-port network:

  - Ports 1 and 2 become differential-mode pair 1.
  - Ports 3 and 4 become differential-mode pair 2.
  - Ports 5 and 6 become differential-mode pair 3.
  - Ports 7 and 8 become differential-mode pair 4.

  The following figure illustrates this convention for an 8-port device.

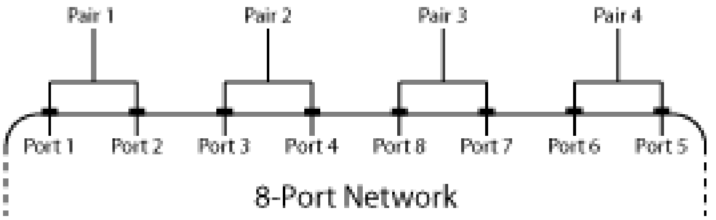


- `3` — `s2sdd` pairs the input ports in ascending order and the output ports in descending order. For example, in a single-ended, 8-port network:

  - Ports 1 and 2 become differential-mode pair 1.
  - Ports 3 and 4 become differential-mode pair 2.
  - Ports 8 and 7 become differential-mode pair 3.

- Ports 6 and 5 become differential-mode pair 4.

The following figure illustrates this convention for an 8-port device.



Examples

Analyze Differential Mode S-Parameters

Create a circuit object from a data file.

```
ckt = read(rfckt.passive,'default.s4p');
data = ckt.AnalyzedResult;
```

Create a data object to store the differential S-parameters.

```
diffSparams = rfdata.network;
diffSparams.Freq = data.Freq;
diffSparams.Data = s2sdd(data.S_Parameters);
diffSparams.Z0 = 2*data.Z0;
```

Create a new circuit object with the data from the data object.

```
diffCkt = rfckt.passive;
diffCkt.NetworkData = diffSparams;
```

Analyze the new circuit object.

```
frequencyRange = diffCkt.NetworkData.Freq;
ZL = 50;
ZS = 50;
Z0 = diffSparams.Z0;
analyze(diffCkt,frequencyRange,ZL,ZS,Z0);
diffData = diffCkt.AnalyzedResult;
```

Write the differential S-parameters into a Touchstone data file.

```
write(diffCkt,'diffsparams.s2p')
```

```
ans = logical
   1
```

Single-ended S-Parameters to Differential-Mode S-Parameters

Convert network data to differential-mode S-parameters using the default port ordering.

```
ckt = read(rfckt.passive,'default.s4p');
s4p = ckt.NetworkData.Data;
s_dd = s2sdd(s4p);
s_dd2 = s_dd(1:5)
```

```
s_dd2 = 1×5 complex

  -0.0124 - 0.0433i  -0.5428 - 0.6900i  -0.5434 - 0.6872i  -0.0192 - 0.0504i  -0.0138 -
```

References

Fan, W., A. C. W. Lu, L. L. Wai, and B. K. Lok. "Mixed-Mode S-Parameter Characterization of Differential Structures." Electronic Packaging Technology Conference, pp. 533–537, 2003.

See Also

s2scc | s2scd | s2sdc | s2smm | smm2s

Introduced in R2006a

s2smm

Convert single-ended S-parameters to mixed-mode S-parameters

Syntax

```
[s_dd,s_dc,s_cd,s_cc] = s2smm(s_params_even,rfflag)
s_mm = s2smm(s_params_odd)
```

Description

`[s_dd,s_dc,s_cd,s_cc] = s2smm(s_params_even,rfflag)` converts single-ended S-parameters to mixed-mode form.

`s_mm = s2smm(s_params_odd)` converts single-ended odd S-parameters matrix to mixed-mode matrix. To create a mixed-mode matrix from `s_params_odd`, the single-ended input ports are paired sequentially (port1 with port 2, port 3 with port 4, etc.), and the last port is left single ended.

Input Arguments

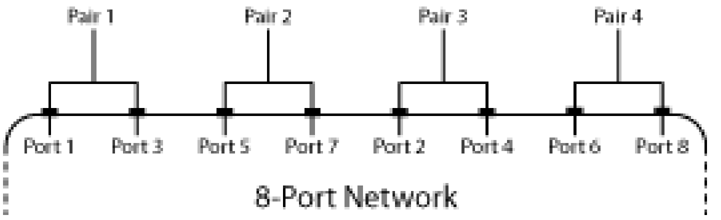
`s_params_even` — S-parameters
2 *N* by 2 *N* by *K* array

S-parameters, specified as a complex 2 *N* by 2 *N* by K array of *K* 2 *N*-port S-Parameters. These parameters describe a device with an even number of ports.

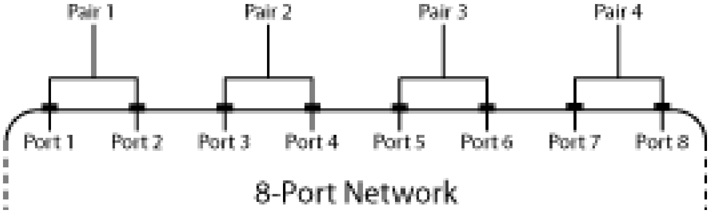
`rfflag` — Port order
1 (default) | 2 | 3

Port order, specified as `1`, `2`, `3`, determines how the function orders the ports:

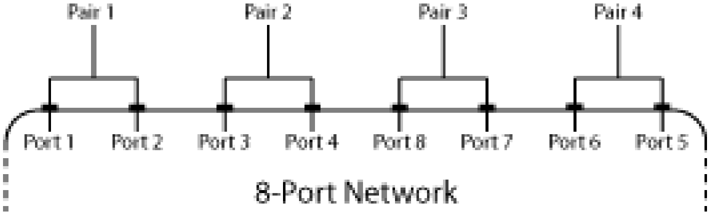
- `rfflag = 1` — `s2smm` Odd-numbered ports are followed by even-numbered ports: 1,3,5,.....,2*N*-4,2*N*-2,2*N*.



  - Ports 1 and 3 become mixed-mode pair 1.
  - Ports 5 and 7 become mixed-mode pair 2.
  - Ports 2 and 4 become mixed-mode pair 3.
  - Ports 6 and 8 become mixed-mode pair 4.
- `rfflag = 2` — Ports are paired in ascending or descending order: (1,2),.....,(2*N*-1,2*N*)



  - Ports 1 and 2 become mixed-mode pair 1.
  - Ports 3 and 4 become mixed-mode pair 2.
  - Ports 5 and 6 become mixed-mode pair 3.
  - Ports 7 and 8 become mixed-mode pair 4.
- `rfflag = 3` — Half of the ports are in ascending order and half of the ports are in descending order: 1,2,.....,*N*,2*N*,2*N*-1,....,*N*+1.



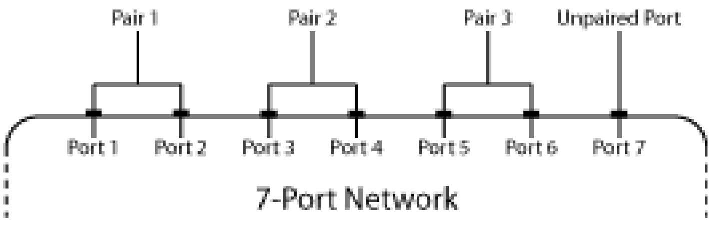
  - Ports 1 and 2 become mixed-mode pair 1.

- Ports 3 and 4 become mixed-mode pair 2.
- Ports 8 and 7 become mixed-mode pair 3.
- Ports 6 and 5 become mixed-mode pair 4.

s_params_odd — S-parameters
array

S-parameters, specified as a complex (2 *N*+1) by (2 *N*+1) by K array of *K* (2 *N*+1) port single-ended S-Parameters matrix. These parameters describe a device with an odd number of ports.

The port-ordering argument `option` is not available for (2*N* + 1)-by-(2*N* + 1)-by-*K* input arrays. In this case, the ports are always paired in ascending order, and the last port remains single-ended. For example, in a 7-port network:



- Ports 1 and 2 become mixed-mode pair 1.
- Ports 3 and 4 become mixed-mode pair 2.
- Ports 5 and 6 become mixed-mode pair 3.
- Ports 7 remains single ended.

Output Arguments

s_dd — Mixed-mode S-parameters
complex *N*-by-*N*-by-*K* array

Mixed-mode S-parameters, returned as complex *N*-by-*N*-by-*K* array, containing *K* matrices of differential-mode, 2*N*-port S-parameters (S_{dd}).

s_dc — Mixed-mode S-parameters
complex *N*-by-*N*-by-*K* array

Mixed-mode S-parameters, returned as complex *N*-by-*N*-by-*K* array, containing *K* matrices of differential-mode, 2*N*-port S-parameters (S_{dc}).

s_cd — Mixed-mode S-parameters
complex *N*-by-*N*-by-*K* array

Mixed-mode S-parameters, returned as complex *N*-by-*N*-by-*K* array, containing *K* matrices of differential-mode, 2*N*-port S-parameters (S_{cd}).

s_cc — Mixed-mode S-parameters
complex *N*-by-*N*-by-*K* array

Mixed-mode S-parameters, returned as complex *N*-by-*N*-by-*K* array, containing *K* matrices of differential-mode, 2*N*-port S-parameters (S_{cc}).

s_mm — Mixed-mode S-parameters
complex *N*-by-*N*-by-*K* array

Mixed-mode S-parameters, returned as complex *N*-by-*N*-by-*K* array, containing *K* matrices of differential-mode, 2*N*-port S-parameters (S_{mm}).

$$\begin{bmatrix} S_{dd,11} & \cdots & S_{dd,1N} & S_{dc,11} & \cdots & S_{dc,1N} & S_{ds,1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ S_{dd,N1} & \cdots & S_{dd,NN} & S_{dc,N1} & \cdots & S_{dc,NN} & S_{ds,N} \\ S_{cd,11} & \cdots & S_{cd,1N} & S_{cc,11} & \cdots & S_{cc,1N} & S_{cs,1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ S_{cd,N1} & \cdots & S_{cd,NN} & S_{cc,N1} & \cdots & S_{cc,NN} & S_{cs,N} \\ S_{sd,1} & \cdots & S_{sd,N} & S_{sc,1} & \cdots & S_{sc,N} & S_{ss} \end{bmatrix}$$

Examples

4-Port S-Parameters to 2-Port Mixed-Mode S-Parameters

Convert 4-port S-parameters to 2-port, mixed-mode S-parameters.

```
ckt = read(rfckt.passive,'default.s4p');
s4p = ckt.NetworkData.Data;
```

```
[s_dd,s_dc,s_cd,s_cc] = s2smm(s4p);
s_dd1=s_dd(1:5)
```

```
s_dd1 = 1×5 complex

  -0.0124 - 0.0433i  -0.5428 - 0.6900i  -0.5434 - 0.6872i  -0.0192 - 0.0504i  -0.0138 -
```

```
s_dc1=s_dc(1:5)
```

```
s_dc1 = 1×5 complex

   0.0024 - 0.0035i   0.0007 - 0.0012i  -0.0005 + 0.0019i   0.0023 - 0.0027i   0.0020 -
```

```
s_cc1=s_cc(1:5)
```

```
s_cc1 = 1×5 complex

   0.1799 - 0.1839i  -0.5314 - 0.6800i  -0.5300 - 0.6771i   0.1756 - 0.1910i   0.1045 -
```

```
s_cd=s_cd(1:5)
```

```
s_cd = 1×5 complex

   0.0015 - 0.0029i   0.0003 - 0.0009i  -0.0005 + 0.0014i   0.0019 - 0.0027i   0.0030 -
```

References

Granberg, T., *Handbook of Digital Techniques for High-Speed Design*.Upper Saddle River, NJ: Prentice Hall, 2004.

See Also

`s2scc` | `s2scd` | `s2sdc` | `s2sdd` | `smm2s` | `snp2smp`

Introduced in R2009a

s2t

Convert S-parameters to T-parameters

Syntax

```
t_params = s2t(s_params)
```

Description

`t_params = s2t(s_params)` converts the scattering parameters `s_params` into the chain scattering parameters `t_params`. The `s_params` input is a complex 2-by-2-by-*M* array, representing *M* 2-port S-parameters. `t_params` is a complex 2-by-2-by-*M* array, representing *M* 2-port T-parameters.

This function uses the following definition for T-parameters:

$$\begin{bmatrix} a_1 \\ b_1 \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \begin{bmatrix} b_2 \\ a_2 \end{bmatrix},$$

where:

- a_1 is the incident wave at the first port.
- b_1 is the reflected wave at the first port.
- a_2 is the incident wave at the second port.
- b_2 is the reflected wave at the second port.

Examples

S-Parameters to T-Parameters

Convert S-parameters to T-parameters. Define a matrix of S-parameters.

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

Convert to T-parameters

```
t_params = s2t(s_params)
```

```
t_params = 2×2 complex

   0.1385 - 0.2304i   0.0354 + 0.1157i
  -0.0452 + 0.1576i  -0.0019 - 0.0291i
```

References

Gonzalez, Guillermo, *Microwave Transistor Amplifiers: Analysis and Design*, 2nd edition. Prentice-Hall, 1997, p. 25.

See Also

s2abcd | s2h | s2y | s2z | t2s

Introduced before R2006a

s2tf

Convert S-parameters of 2-port network to voltage or power-wave transfer function

Syntax

```
tf = s2tf(s_params)
tf = s2tf(s_params,z0,zs,zl)
tf = s2tf(s_params,z0,zs,zl,option)

tf = s2tf(hs)
tf = s2tf(hs,zs,zl)
tf = s2tf(hs,zs,zl,option)
```

Description

`tf = s2tf(s_params)` converts the scattering parameters, `s_params`, of a 2-port network into the voltage transfer function of the network.

`tf = s2tf(s_params,z0,zs,zl)` calculates the voltage transfer function using the reference impedance `z0`, source impedance `zs`, and load impedance `zl`.

`tf = s2tf(s_params,z0,zs,zl,option)` calculates the voltage or power-wave transfer function using the method specified by `option`.

`tf = s2tf(hs)` converts the 2-port S-parameter object, `hs`, into the voltage transfer function of the network.

`tf = s2tf(hs,zs,zl)` calculates the voltage transfer function using the source impedance `zs`, and load impedance `zl`.

`tf = s2tf(hs,zs,zl,option)` calculates the voltage or power-wave transfer function using the method specified by `option`.

Input Arguments

`hs` — 2-port s-parameters
`s-parameter` object

2-port S-parameters, specified as an RF Toolbox S-parameter object.

`s_params` — Scattering parameters
2x2xM array (default)

Scattering parameters, specified as a complex 2-by-2-by-*M* array.

`z0` — Reference impedance
50 ohms (default)

Reference impedance of S-parameters, specified in ohms.

`zs` — Source impedance
50 ohms (default)

Source impedance of S-parameters, specified in ohms.

`zl` — Load impedance
50 ohms (default)

Load impedance of S-parameters, specified in ohms.

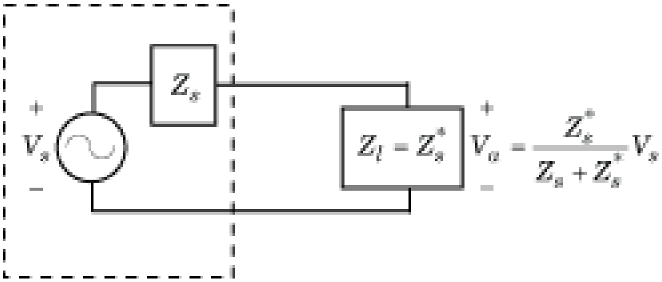
`option` — Transfer function type
1 (default) | integer

Transfer function type, specified as an integer equal to `1`, `2`, or `3`.

- `1` — The transfer function is the gain from the incident voltage, V_a, to the output voltage for arbitrary source and load impedances:

  $$tf = \frac{V_l}{V_a}$$

  The following figure shows how to compute V_a from the source voltage V_s:



For the S-parameters and impedance values, the transfer function is:

$$tf = \frac{(Z_s + Z_s^*)}{Z_s^*} \frac{S_{21}(1+\Gamma_l)(1-\Gamma_s)}{2(1-S_{22}\Gamma_l)(1-\Gamma_{in}\Gamma_s)}$$

where:

$$\Gamma_l = \frac{Z_l - Z_o}{Z_l + Z_o}$$

$$\Gamma_s = \frac{Z_s - Z_o}{Z_s + Z_o}$$

$$\Gamma_{in} = S_{11} + \left(S_{12}S_{21} \frac{\Gamma_l}{(1 - S_{22}\Gamma_l)} \right)$$

The following equation shows how the preceding transfer function is related to the transducer gain computed by the `powergain` function:

$$G_T = |tf|^2 \frac{\mathrm{Re}(Z_l)}{|Z_l|^2} \frac{|Z_s|^2}{\mathrm{Re}(Z_s)}$$

Notice that if Z_l and Z_S are real, $G_T = |tf|^2 \frac{Z_s}{Z_l}$.

- 2 — The transfer function is the gain from the source voltage to the output voltage for arbitrary source and load impedances:

$$tf = \frac{V_l}{V_s} = \frac{S_{21}(1+\Gamma_l)(1-\Gamma_s)}{2(1-S_{22}\Gamma_l)(1-\Gamma_{in}\Gamma_s)}$$

You can use this option to compute the transfer function $\frac{V_L}{V_{in}}$ by setting `zs` to `0`. This setting means that Γ_s = –1 and $V_{in} = V_s$.

- `3` — The transfer function is the power-wave gain from the incident power wave at the first port to the transmitted power wave at the second port:

$$tf = \frac{b_{p2}}{a_{p1}} = \frac{\sqrt{\mathrm{Re}(Z_l)\,\mathrm{Re}(Z_s)}}{Z_l}\frac{S_{21}(1+\Gamma_l)(1-\Gamma_s)}{(1-S_{22}\Gamma_l)(1-\Gamma_{in}\Gamma_s)}$$

Output Arguments

`tf` — Voltage transfer function

vector of doubles

Voltage transfer function, returned as a vector of doubles.

Examples

S-Parameters to Voltage or Power Transfer Function

Calculate the voltage transfer function of an S-parameter array.

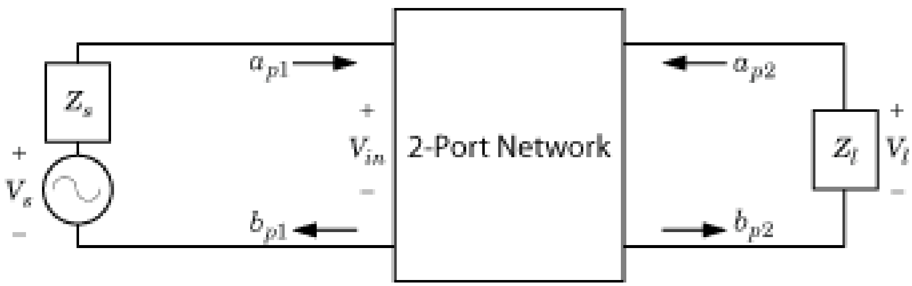
```
ckt = read(rfckt.passive,'passive.s2p');
sparams = ckt.NetworkData.Data;
tf = s2tf(sparams)
```

```
tf = 202×1 complex

   0.9964 - 0.0254i
   0.9960 - 0.0266i
   0.9956 - 0.0284i
   0.9961 - 0.0290i
```

```
0.9960 - 0.0301i
0.9953 - 0.0317i
0.9953 - 0.0334i
0.9952 - 0.0349i
0.9949 - 0.0367i
0.9946 - 0.0380i
    ⋮
```

Algorithms

The following figure shows the setup for computing the transfer function, along with the impedances, voltages, and the power waves used to determine the gain.



The function uses the following voltages and power waves for calculations:

- V_l is the output voltage across the load impedance.
- V_s is the source voltage.
- V_{in} is the input voltage of the 2-port network.
- a_{p1} is the incident power wave, equal to $\dfrac{V_s}{2\sqrt{\mathrm{Re}(Z_s)}}$.
- b_{p2} is the transmitted power wave, equal to $\dfrac{\sqrt{\mathrm{Re}(Z_l)}}{Z_l}V_l$.

See Also

`powergain` | `rationalfit` | `s2scc` | `s2scd` | `s2sdc` | `s2sdd` | `snp2smp`

Introduced in R2006b

s2y

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.

Examples

Convert S-Parameters to Y-Parameters

Define the s-parameters and impedance.

```
s_11 = 0.61*exp(1i*165/180*pi);
s_21 = 3.72*exp(1i*59/180*pi);
s_12 = 0.05*exp(1i*42/180*pi);
s_22 = 0.45*exp(1i*(-48/180)*pi);
s_params = [s_11 s_12; s_21 s_22];
z0 = 50;
```

Convert the s-parameters to y-parameters.

```
y_params = s2y(s_params,z0)
```

```
y_params = 2×2 complex

   0.0647 - 0.0059i  -0.0019 - 0.0025i
```

```
  -0.0826 - 0.2200i   0.0037 + 0.0145i
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see “RF Network Parameter Objects” on page 2-11.

See Also

abcd2y | h2y | s2abcd | s2h | s2z | y2s | z2y

Introduced before R2006a

s2z

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.

Examples

S-Parameters to Z-Parameters

Convert S-parameters to Z-parameters. Define a matrix of S-parameters.

```
s_11 = 0.61*exp(j*165/180*pi);
s_21 = 3.72*exp(j*59/180*pi);
s_12 = 0.05*exp(j*42/180*pi);
s_22 = 0.45*exp(j*(-48/180)*pi);
s_params = [s_11 s_12; s_21 s_22];
z0 = 50;
```

Convert to Z-parameters.

```
z_params = s2z(s_params,z0)
```

```
z_params = 2×2 complex
10^2 ×
```

```
0.1141 + 0.1567i   0.0352 + 0.0209i
2.0461 + 2.2524i   0.7498 - 0.3803i
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see “RF Network Parameter Objects” on page 2-11.

See Also

`abcd2z` | `h2z` | `s2abcd` | `s2h` | `s2y` | `y2z` | `z2s`

Introduced before R2006a

smithchart

Plot complex vector of a reflection coefficient on Smith chart

Syntax

`[lineseries,hsm] = smithchart(gamma)`

`hsm = smithchart`

Description

`[lineseries,hsm] = smithchart(gamma)` plots the complex vector of a reflection coefficient `gamma` on a Smith chart. `hsm` is the handle of the Smith chart object. `lineseries` is a column vector of handles to `lineseries` objects, one handle per plotted line.

To plot network parameters from a circuit (`rfckt`) or data (`rfdata`) object on a Smith chart, use the `smith` function.

`hsm = smithchart` draws a blank Smith chart and returns the handle, `hsm`, of the Smith chart object.

Input arguments

gamma — reflection coefficient
complex vector

Reflection coefficient, specified as a complex vector.

Data Types: `double`

Output Arguments

`lineseries` — line properties handle
column vector

Line properties handle, returned as a column vector, changes the properties of the plotted lines by changing the Chart Line. There is one handle per plotted line.

`hsm` — Smith chart handle
scalar handle

Smith chart handle, specified as a scalar handle.

This table lists all properties you can specify for `smithchart` objects along with units, valid values, and descriptions of their use.

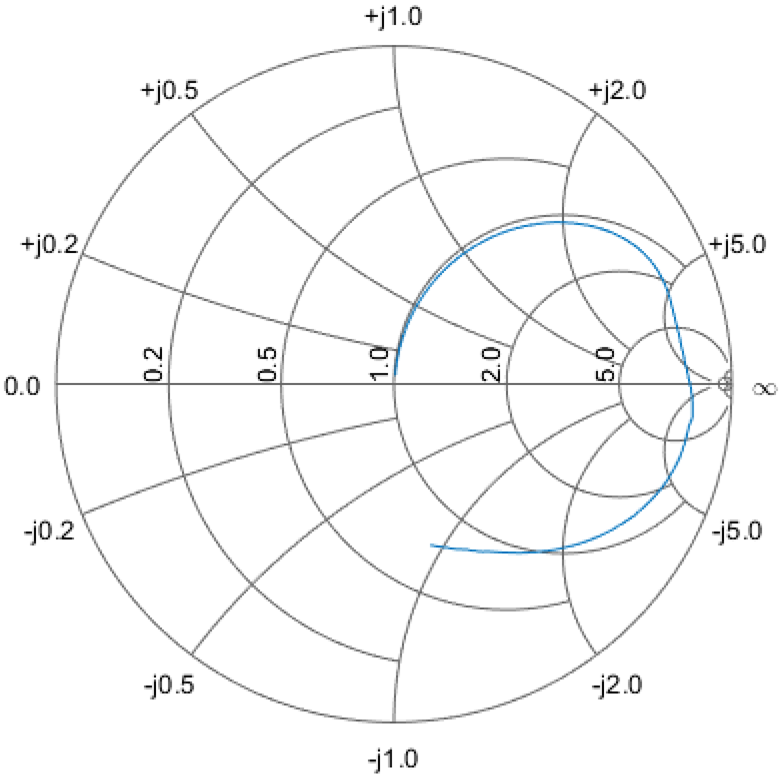
Property Name	Description	Units and Values
`Color`	Line color for a Z or Y Smith chart. For a ZY Smith chart, the Z line color.	`ColorSpec`. Default is `[0.4 0.4 0.4]` (dark gray).
`LabelColor`	Color of the line labels.	`ColorSpec`. Default is `[0 0 0]` (black).
`LabelSize`	Size of the line labels.	`FontSize`. Default is `10`.
`LabelVisible`	Visibility of the line labels.	`'on'` (default) or `'off'`
`LineType`	Line spec for a Z or Y Smith chart. For a ZY Smith chart, the Z line spec.	`LineSpec`. Default is `'-'` (solid line).
`LineWidth`	Line width for a Z or Y Smith chart. For a ZY Smith chart, the Z line width.	Number of points. Default is `0.5`.
`SubColor`	The Y line color for a ZY Smith chart.	`ColorSpec`. Default is `[0.8 0.8 0.8]` (medium gray).
`SubLineType`	The Y line spec for a ZY Smith chart.	`LineSpec`. Default is `':'` (dotted line).

Property Name	Description	Units and Values
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 in the chart. For the constant resistance and reactance lines, each element in Row 2 specifies the value at which the corresponding line in Row 1 ends.	2-by-n matrix. Default is [0.2000 0.5000 1.0000 2.0000 5.0000; 1.0000 2.0000 5.0000 5.0000 30.0000]

Examples

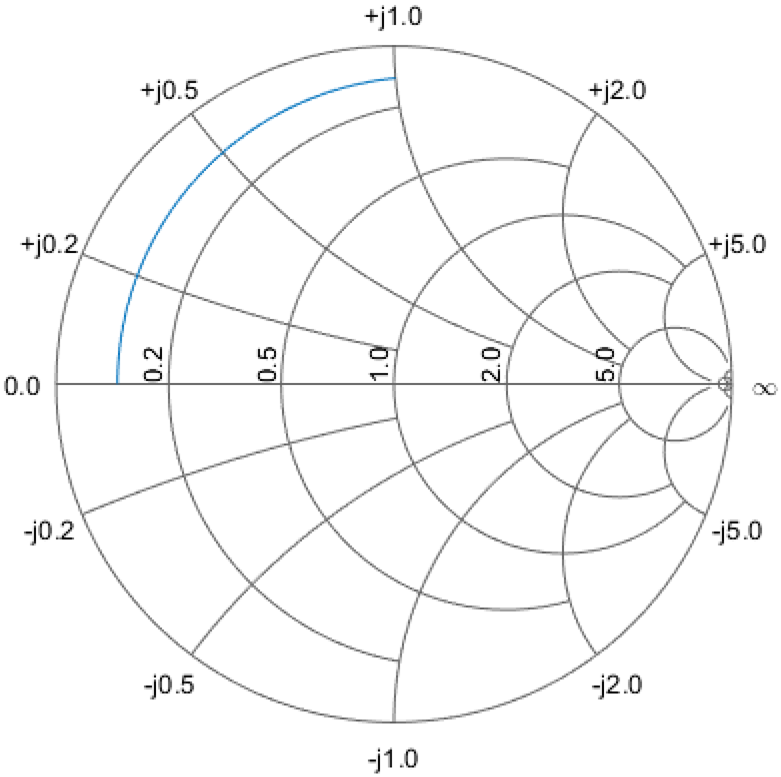
Plot Reflection Data On a Smith Chart

```
S = sparameters('passive.s2p');
    s11 = rfparam(S,1,1);
    figure
    smithchart(s11)
```



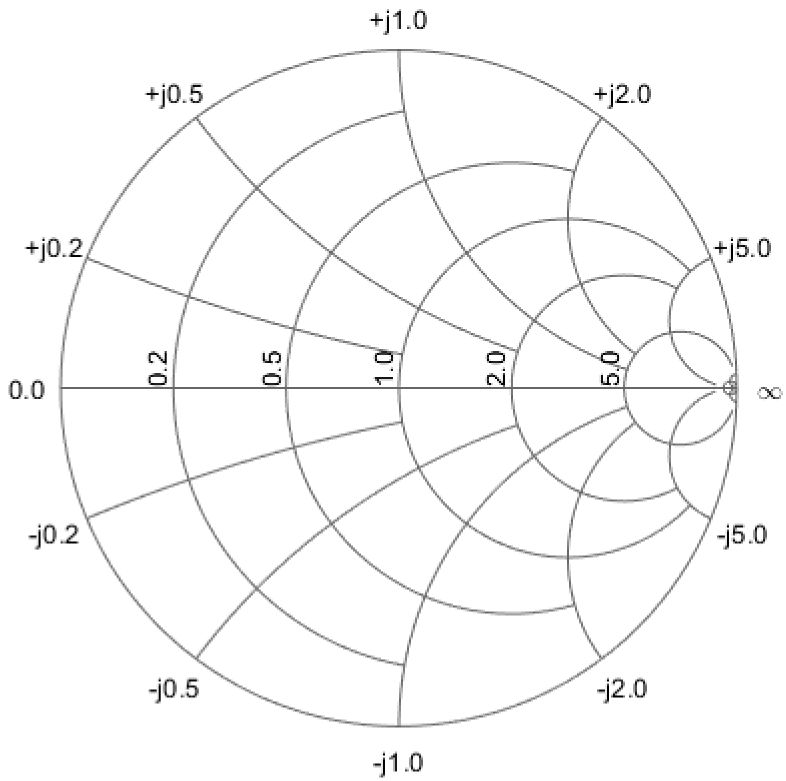
Plot Impedance Data On a Smith Chart

```
z = 0.1*50 + 1j*(0:0.1:50);
gamma = z2gamma(z);
figure
smithchart(gamma)
```



Draw a Blank Smithchart

```
hsm = smithchart
```




```
hsm =
   rfchart.smith with properties:

            Type: 'Z'
          Values: [2x5 double]
           Color: [0.4000 0.4000 0.4000]
       LineWidth: 0.5000
        LineType: '-'
        SubColor: [0.8000 0.8000 0.8000]
    SubLineWidth: 0.5000
     SubLineType: ':'
    LabelVisible: 'On'
       LabelSize: 10
      LabelColor: [0 0 0]
```

```
Name: 'Smith chart'
```

See Also

get | set | smith | smithplot

Introduced before R2006a

smm2s

Convert mixed-mode 2N-port S-parameters to single-ended 4N-port S-parameters

Syntax

```
s_params = smm2s(s_dd,s_dc,s_cd,s_cc)
s_params = smm2s(s_dd,s_dc,s_cd,s_cc,option)
```

Description

`s_params = smm2s(s_dd,s_dc,s_cd,s_cc)` converts mixed-mode, *N*-port S-parameters into single-ended, 2*N*-port S-parameters, `s_params`. `smm2s` maps the first half of the mixed-mode ports to the odd-numbered pairs of single-ended ports and maps the second half to the even-numbered pairs.

`s_params = smm2s(s_dd,s_dc,s_cd,s_cc,option)` converts the S-parameter data using the optional argument `option`. You can also reorder the ports in `s_params` using the `snp2smp` function.

Input Arguments

s_cc — S-parameters

array

S-parameters, specified as a complex *N*-by-*N*-by-*K* array containing *K* matrices of common-mode, *N*-port S-parameters (S_{cc}).

s_cd — S-parameters

array

S-parameters, specified as a complex *N*-by-*N*-by-*K* array containing *K* matrices of cross-mode, *N*-port S-parameters (S_{cd}).

s_dc — S-parameters

array

S-parameters, specified as a complex *N*-by-*N*-by-*K* array containing *K* matrices of cross-mode, *N*-port S-parameters (S_{dc}).

`s_dd` — S-parameters
array

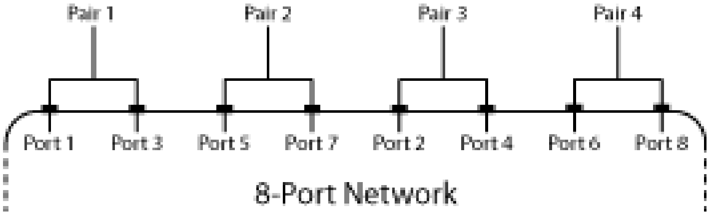
S-parameters, specified as a complex *N*-by-*N*-by-*K* array containing *K* matrices of differential-mode, *N*-port S-parameters (S_{dd}).

`option` — Port order
1 (default) | 2 | 3

Port order, specified as `1`, `2`, `3` determines how the function orders the ports:

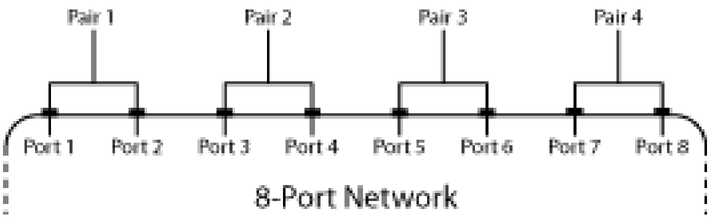
- `1` — `smm2s` maps the first half of the mixed-mode pairs to odd-numbered pairs of single-ended ports and maps the second half to even-numbered pairs. For example, in a mixed-mode, 4-port network:

  - Port 1 becomes single-ended ports 1 and 3.
  - Port 2 becomes single-ended ports 5 and 7.
  - Port 3 becomes single-ended ports 2 and 4.
  - Port 4 becomes single-ended ports 6 and 8.

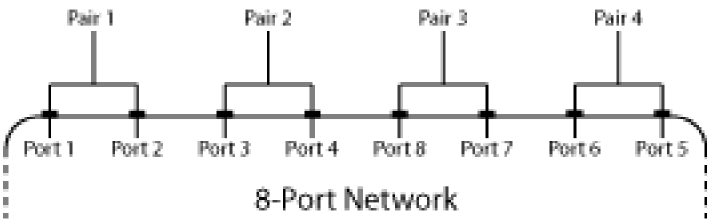


- `2` — `smm2s` maps the first half of the mixed-mode pairs to single-ended ports in ascending numerical order, followed by the second half, also in ascending order. For example, in a mixed-mode, 4-port network:

  - Port 1 becomes single-ended ports 1 and 2.
  - Port 2 becomes single-ended ports 3 and 4.
  - Port 3 becomes single-ended ports 5 and 6.
  - Port 4 becomes single-ended ports 7 and 8.



- `3` — `smm2s` maps the first half of the mixed-mode pairs to single-ended ports in ascending numerical order. The function maps the second half to pairs of ports in descending order. For example, in a mixed-mode, 4-port network:
  - Port 1 becomes single-ended ports 1 and 2.
  - Port 2 becomes single-ended ports 3 and 4.
  - Port 3 becomes single-ended ports 8 and 7.
  - Port 4 becomes single-ended ports 6 and 5.



Output Arguments

`s_params` — S-parameters
array

S-parameters, returned as a complex 2*N*-by-2*N*-by-*K* array representing *K* single-ended, 2*N*-port S-parameters.

Examples

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

Convert between mixed-mode and single-ended S-parameters. Create mixed-mode S-parameters:

```
ckt = read(rfckt.passive,'default.s4p');
s4p = ckt.NetworkData.Data;
[sdd,scd,sdc,scc] = s2smm(s4p);
```

Convert them back to 4-port,single-ended S-parameters.

```
s4p_converted_back = smm2s(sdd,scd,sdc,scc)
```

```
s4p_converted_back =
s4p_converted_back(:,:,1) =

   0.0857 - 0.1168i  -0.5372 - 0.6804i   0.0966 - 0.0706i   0.0067 + 0.0053i
  -0.5366 - 0.6860i   0.0803 - 0.1234i   0.0059 + 0.0048i   0.0977 - 0.0703i
   0.0957 - 0.0700i   0.0067 + 0.0048i   0.0818 - 0.1104i  -0.5362 - 0.6838i
   0.0055 + 0.0051i   0.0972 - 0.0703i  -0.5376 - 0.6840i   0.0761 - 0.1180i


s4p_converted_back(:,:,2) =

   0.0479 - 0.1334i  -0.7665 - 0.3900i   0.0586 - 0.1042i   0.0071 - 0.0003i
  -0.7674 - 0.3903i   0.0365 - 0.1395i   0.0070 - 0.0004i   0.0602 - 0.1034i
   0.0597 - 0.1028i   0.0062 + 0.0001i   0.0428 - 0.1282i  -0.7686 - 0.3880i
   0.0068 - 0.0001i   0.0607 - 0.1033i  -0.7682 - 0.3889i   0.0348 - 0.1310i


s4p_converted_back(:,:,3) =

   0.0031 - 0.1361i  -0.8526 - 0.0298i   0.0118 - 0.1094i   0.0044 - 0.0045i
  -0.8535 - 0.0309i  -0.0084 - 0.1364i   0.0043 - 0.0041i   0.0140 - 0.1103i
   0.0107 - 0.1093i   0.0043 - 0.0040i   0.0005 - 0.1282i  -0.8536 - 0.0292i
   0.0047 - 0.0039i   0.0141 - 0.1100i  -0.8526 - 0.0288i  -0.0063 - 0.1275i


s4p_converted_back(:,:,4) =

  -0.0362 - 0.1206i  -0.7807 + 0.3291i  -0.0284 - 0.0909i   0.0001 - 0.0043i
  -0.7805 + 0.3295i  -0.0459 - 0.1168i  -0.0004 - 0.0054i  -0.0261 - 0.0929i
  -0.0291 - 0.0912i  -0.0003 - 0.0052i  -0.0363 - 0.1105i  -0.7802 + 0.3327i
  -0.0001 - 0.0049i  -0.0263 - 0.0928i  -0.7798 + 0.3313i  -0.0404 - 0.1107i
```

```
s4p_converted_back(:,:,5) =

  -0.0649 - 0.0912i  -0.5652 + 0.6246i  -0.0491 - 0.0579i  -0.0030 - 0.0022i
  -0.5640 + 0.6257i  -0.0717 - 0.0865i  -0.0041 - 0.0022i  -0.0473 - 0.0614i
  -0.0501 - 0.0576i  -0.0038 - 0.0024i  -0.0619 - 0.0819i  -0.5638 + 0.6259i
  -0.0035 - 0.0020i  -0.0477 - 0.0614i  -0.5628 + 0.6255i  -0.0646 - 0.0836i


s4p_converted_back(:,:,6) =

  -0.0760 - 0.0541i  -0.2470 + 0.7983i  -0.0490 - 0.0247i  -0.0037 + 0.0024i
  -0.2483 + 0.7999i  -0.0810 - 0.0502i  -0.0045 + 0.0023i  -0.0481 - 0.0295i
  -0.0489 - 0.0253i  -0.0041 + 0.0025i  -0.0724 - 0.0479i  -0.2448 + 0.8009i
  -0.0038 + 0.0023i  -0.0475 - 0.0295i  -0.2471 + 0.8013i  -0.0749 - 0.0513i


s4p_converted_back(:,:,7) =

  -0.0714 - 0.0200i   0.1122 + 0.8232i  -0.0321 - 0.0040i  -0.0004 + 0.0055i
   0.1127 + 0.8241i  -0.0737 - 0.0185i  -0.0009 + 0.0059i  -0.0331 - 0.0093i
  -0.0326 - 0.0040i  -0.0008 + 0.0055i  -0.0676 - 0.0163i   0.1154 + 0.8228i
  -0.0005 + 0.0059i  -0.0331 - 0.0093i   0.1151 + 0.8238i  -0.0708 - 0.0209i


s4p_converted_back(:,:,8) =

  -0.0516 + 0.0022i   0.4469 + 0.6936i  -0.0116 - 0.0010i   0.0049 + 0.0056i
   0.4472 + 0.6934i  -0.0540 + 0.0025i   0.0049 + 0.0058i  -0.0150 - 0.0057i
  -0.0119 - 0.0012i   0.0049 + 0.0057i  -0.0497 + 0.0051i   0.4494 + 0.6931i
   0.0050 + 0.0055i  -0.0150 - 0.0058i   0.4490 + 0.6911i  -0.0534 + 0.0016i


s4p_converted_back(:,:,9) =

  -0.0277 + 0.0060i   0.6935 + 0.4364i   0.0010 - 0.0123i   0.0097 + 0.0012i
   0.6933 + 0.4368i  -0.0296 + 0.0057i   0.0094 + 0.0011i  -0.0040 - 0.0156i
   0.0009 - 0.0123i   0.0094 + 0.0011i  -0.0277 + 0.0109i   0.6940 + 0.4357i
   0.0096 + 0.0009i  -0.0040 - 0.0157i   0.6951 + 0.4340i  -0.0307 + 0.0077i


s4p_converted_back(:,:,10) =

  -0.0136 - 0.0068i   0.8055 + 0.1016i  -0.0017 - 0.0298i   0.0087 - 0.0069i
```

```
   0.8046 + 0.1017i  -0.0150 - 0.0075i   0.0085 - 0.0065i  -0.0075 - 0.0308i
  -0.0014 - 0.0300i   0.0083 - 0.0065i  -0.0143 + 0.0004i   0.8057 + 0.1004i
   0.0089 - 0.0068i  -0.0076 - 0.0307i   0.8059 + 0.0987i  -0.0129 - 0.0026i


s4p_converted_back(:,:,11) =

  -0.0148 - 0.0237i   0.7676 - 0.2439i  -0.0170 - 0.0404i   0.0047 - 0.0114i
   0.7675 - 0.2439i  -0.0141 - 0.0259i   0.0044 - 0.0109i  -0.0224 - 0.0387i
  -0.0168 - 0.0403i   0.0045 - 0.0109i  -0.0151 - 0.0146i   0.7675 - 0.2471i
   0.0047 - 0.0114i  -0.0221 - 0.0389i   0.7673 - 0.2479i  -0.0088 - 0.0216i


s4p_converted_back(:,:,12) =

  -0.0356 - 0.0360i   0.5868 - 0.5408i  -0.0407 - 0.0403i  -0.0034 - 0.0141i
   0.5872 - 0.5403i  -0.0338 - 0.0416i  -0.0031 - 0.0130i  -0.0443 - 0.0371i
  -0.0406 - 0.0402i  -0.0033 - 0.0131i  -0.0336 - 0.0249i   0.5842 - 0.5423i
  -0.0031 - 0.0139i  -0.0446 - 0.0371i   0.5859 - 0.5431i  -0.0246 - 0.0406i


s4p_converted_back(:,:,13) =

  -0.0662 - 0.0284i   0.3018 - 0.7298i  -0.0635 - 0.0239i  -0.0118 - 0.0103i
   0.3028 - 0.7304i  -0.0657 - 0.0383i  -0.0110 - 0.0092i  -0.0659 - 0.0201i
  -0.0634 - 0.0239i  -0.0110 - 0.0091i  -0.0610 - 0.0157i   0.3000 - 0.7306i
  -0.0115 - 0.0102i  -0.0660 - 0.0204i   0.2996 - 0.7317i  -0.0558 - 0.0430i


s4p_converted_back(:,:,14) =

  -0.0917 + 0.0025i  -0.0307 - 0.7801i  -0.0765 + 0.0051i  -0.0155 - 0.0015i
  -0.0315 - 0.7792i  -0.0944 - 0.0105i  -0.0144 - 0.0011i  -0.0769 + 0.0088i
  -0.0761 + 0.0046i  -0.0144 - 0.0012i  -0.0821 + 0.0145i  -0.0364 - 0.7790i
  -0.0158 - 0.0017i  -0.0770 + 0.0089i  -0.0354 - 0.7799i  -0.0879 - 0.0208i


s4p_converted_back(:,:,15) =

  -0.0963 + 0.0478i  -0.3504 - 0.6877i  -0.0728 + 0.0388i  -0.0137 + 0.0074i
  -0.3510 - 0.6874i  -0.1031 + 0.0358i  -0.0119 + 0.0070i  -0.0723 + 0.0423i
  -0.0730 + 0.0385i  -0.0121 + 0.0072i  -0.0819 + 0.0582i  -0.3539 - 0.6859i
  -0.0136 + 0.0073i  -0.0725 + 0.0419i  -0.3542 - 0.6857i  -0.1035 + 0.0226i
```

```
s4p_converted_back(:,:,16) =

  -0.0732 + 0.0920i  -0.5976 - 0.4743i  -0.0533 + 0.0679i  -0.0070 + 0.0123i
  -0.5993 - 0.4736i  -0.0826 + 0.0835i  -0.0056 + 0.0111i  -0.0516 + 0.0702i
  -0.0532 + 0.0678i  -0.0056 + 0.0109i  -0.0557 + 0.0992i  -0.6012 - 0.4711i
  -0.0070 + 0.0124i  -0.0518 + 0.0701i  -0.6003 - 0.4709i  -0.0906 + 0.0723i


s4p_converted_back(:,:,17) =

  -0.0290 + 0.1190i  -0.7348 - 0.1811i  -0.0220 + 0.0840i   0.0002 + 0.0125i
  -0.7346 - 0.1822i  -0.0369 + 0.1137i   0.0006 + 0.0105i  -0.0201 + 0.0850i
  -0.0221 + 0.0839i   0.0006 + 0.0106i  -0.0094 + 0.1208i  -0.7350 - 0.1769i
   0.0002 + 0.0125i  -0.0203 + 0.0852i  -0.7359 - 0.1767i  -0.0503 + 0.1088i


s4p_converted_back(:,:,18) =

   0.0215 + 0.1194i  -0.7381 + 0.1372i   0.0116 + 0.0836i   0.0051 + 0.0088i
  -0.7380 + 0.1376i   0.0178 + 0.1141i   0.0043 + 0.0072i   0.0129 + 0.0830i
   0.0114 + 0.0834i   0.0044 + 0.0071i   0.0416 + 0.1156i  -0.7372 + 0.1422i
   0.0052 + 0.0089i   0.0129 + 0.0834i  -0.7379 + 0.1428i   0.0039 + 0.1177i


s4p_converted_back(:,:,19) =

   0.0632 + 0.0932i  -0.6125 + 0.4297i   0.0394 + 0.0669i   0.0052 + 0.0053i
  -0.6129 + 0.4291i   0.0635 + 0.0859i   0.0037 + 0.0041i   0.0394 + 0.0673i
   0.0392 + 0.0671i   0.0036 + 0.0036i   0.0812 + 0.0849i  -0.6097 + 0.4322i
   0.0050 + 0.0052i   0.0400 + 0.0675i  -0.6097 + 0.4322i   0.0535 + 0.0966i


s4p_converted_back(:,:,20) =

   0.0810 + 0.0534i  -0.3771 + 0.6442i   0.0518 + 0.0414i   0.0049 + 0.0046i
  -0.3766 + 0.6435i   0.0832 + 0.0385i   0.0027 + 0.0047i   0.0524 + 0.0416i
   0.0519 + 0.0415i   0.0029 + 0.0047i   0.0966 + 0.0411i  -0.3729 + 0.6447i
   0.0049 + 0.0047i   0.0525 + 0.0414i  -0.3733 + 0.6444i   0.0802 + 0.0538i


s4p_converted_back(:,:,21) =

   0.0744 + 0.0170i  -0.0731 + 0.7403i   0.0469 + 0.0174i   0.0067 + 0.0055i
```

```
  -0.0737 + 0.7411i   0.0716 - 0.0056i   0.0052 + 0.0065i   0.0476 + 0.0174i
   0.0471 + 0.0174i   0.0050 + 0.0064i   0.0862 + 0.0017i  -0.0696 + 0.7397i
   0.0067 + 0.0054i   0.0476 + 0.0173i  -0.0694 + 0.7398i   0.0771 + 0.0091i


s4p_converted_back(:,:,22) =

   0.0516 - 0.0028i   0.2431 + 0.7003i   0.0300 + 0.0060i   0.0112 + 0.0040i
   0.2426 + 0.7005i   0.0401 - 0.0287i   0.0106 + 0.0054i   0.0305 + 0.0060i
   0.0300 + 0.0058i   0.0105 + 0.0055i   0.0591 - 0.0194i   0.2454 + 0.6986i
   0.0112 + 0.0040i   0.0305 + 0.0061i   0.2459 + 0.6993i   0.0518 - 0.0193i


s4p_converted_back(:,:,23) =

   0.0292 - 0.0024i   0.5134 + 0.5301i   0.0124 + 0.0124i   0.0151 - 0.0020i
   0.5128 + 0.5306i   0.0080 - 0.0246i   0.0151 - 0.0009i   0.0130 + 0.0127i
   0.0123 + 0.0122i   0.0152 - 0.0008i   0.0319 - 0.0188i   0.5151 + 0.5283i
   0.0151 - 0.0021i   0.0131 + 0.0127i   0.5149 + 0.5273i   0.0215 - 0.0222i


s4p_converted_back(:,:,24) =

   0.0182 + 0.0119i   0.6831 + 0.2629i   0.0074 + 0.0324i   0.0140 - 0.0113i
   0.6830 + 0.2633i  -0.0066 - 0.0021i   0.0143 - 0.0103i   0.0088 + 0.0328i
   0.0072 + 0.0322i   0.0143 - 0.0103i   0.0176 - 0.0027i   0.6842 + 0.2602i
   0.0140 - 0.0113i   0.0088 + 0.0328i   0.6842 + 0.2600i   0.0040 - 0.0061i


s4p_converted_back(:,:,25) =

   0.0236 + 0.0276i   0.7237 - 0.0476i   0.0214 + 0.0541i   0.0067 - 0.0186i
   0.7246 - 0.0469i   0.0024 + 0.0205i   0.0069 - 0.0179i   0.0239 + 0.0536i
   0.0212 + 0.0540i   0.0070 - 0.0179i   0.0209 + 0.0151i   0.7227 - 0.0508i
   0.0066 - 0.0185i   0.0241 + 0.0537i   0.7235 - 0.0507i   0.0071 + 0.0141i


s4p_converted_back(:,:,26) =

   0.0402 + 0.0343i   0.6325 - 0.3429i   0.0516 + 0.0630i  -0.0041 - 0.0199i
   0.6313 - 0.3416i   0.0270 + 0.0275i  -0.0037 - 0.0191i   0.0546 + 0.0611i
   0.0518 + 0.0630i  -0.0038 - 0.0190i   0.0371 + 0.0232i   0.6294 - 0.3451i
  -0.0040 - 0.0199i   0.0550 + 0.0610i   0.6292 - 0.3454i   0.0260 + 0.0231i
```

```
s4p_converted_back(:,:,27) =

   0.0599 + 0.0286i   0.4280 - 0.5680i   0.0870 + 0.0510i  -0.0132 - 0.0152i
   0.4277 - 0.5682i   0.0529 + 0.0131i  -0.0124 - 0.0141i   0.0894 + 0.0474i
   0.0871 + 0.0510i  -0.0124 - 0.0142i   0.0561 + 0.0173i   0.4242 - 0.5709i
  -0.0133 - 0.0152i   0.0895 + 0.0469i   0.4242 - 0.5708i   0.0486 + 0.0137i


s4p_converted_back(:,:,28) =

   0.0748 + 0.0116i   0.1527 - 0.6880i   0.1124 + 0.0185i  -0.0181 - 0.0069i
   0.1522 - 0.6881i   0.0648 - 0.0177i  -0.0164 - 0.0062i   0.1131 + 0.0137i
   0.1120 + 0.0186i  -0.0164 - 0.0064i   0.0687 - 0.0002i   0.1480 - 0.6890i
  -0.0181 - 0.0069i   0.1130 + 0.0131i   0.1474 - 0.6891i   0.0614 - 0.0114i


s4p_converted_back(:,:,29) =

   0.0808 - 0.0112i  -0.1454 - 0.6846i   0.1167 - 0.0246i  -0.0183 + 0.0017i
  -0.1452 - 0.6838i   0.0560 - 0.0522i  -0.0160 + 0.0011i   0.1155 - 0.0302i
   0.1167 - 0.0239i  -0.0160 + 0.0012i   0.0707 - 0.0228i  -0.1500 - 0.6828i
  -0.0183 + 0.0016i   0.1153 - 0.0300i  -0.1498 - 0.6828i   0.0571 - 0.0414i


s4p_converted_back(:,:,30) =

   0.0771 - 0.0354i  -0.4133 - 0.5588i   0.0983 - 0.0634i  -0.0154 + 0.0084i
  -0.4133 - 0.5588i   0.0294 - 0.0779i  -0.0134 + 0.0064i   0.0948 - 0.0685i
   0.0987 - 0.0634i  -0.0133 + 0.0064i   0.0624 - 0.0449i  -0.4179 - 0.5564i
  -0.0154 + 0.0084i   0.0947 - 0.0682i  -0.4170 - 0.5560i   0.0369 - 0.0660i


s4p_converted_back(:,:,31) =

   0.0641 - 0.0578i  -0.6035 - 0.3350i   0.0639 - 0.0866i  -0.0106 + 0.0131i
  -0.6034 - 0.3351i  -0.0065 - 0.0862i  -0.0101 + 0.0102i   0.0584 - 0.0891i
   0.0639 - 0.0868i  -0.0101 + 0.0101i   0.0449 - 0.0626i  -0.6064 - 0.3314i
  -0.0106 + 0.0131i   0.0587 - 0.0887i  -0.6063 - 0.3316i   0.0076 - 0.0773i


s4p_converted_back(:,:,32) =

   0.0415 - 0.0734i  -0.6848 - 0.0555i   0.0253 - 0.0870i  -0.0052 + 0.0149i
```

```
  -0.6848 - 0.0558i  -0.0389 - 0.0755i  -0.0068 + 0.0123i   0.0219 - 0.0863i
   0.0250 - 0.0867i  -0.0068 + 0.0123i   0.0214 - 0.0720i  -0.6853 - 0.0499i
  -0.0052 + 0.0150i   0.0222 - 0.0866i  -0.6860 - 0.0499i  -0.0216 - 0.0735i


s4p_converted_back(:,:,33) =

   0.0154 - 0.0791i  -0.6442 + 0.2343i  -0.0018 - 0.0683i  -0.0006 + 0.0166i
  -0.6436 + 0.2336i  -0.0615 - 0.0550i  -0.0032 + 0.0154i  -0.0023 - 0.0684i
  -0.0016 - 0.0681i  -0.0033 + 0.0154i  -0.0045 - 0.0719i  -0.6424 + 0.2393i
  -0.0006 + 0.0167i  -0.0025 - 0.0687i  -0.6423 + 0.2394i  -0.0444 - 0.0599i


s4p_converted_back(:,:,34) =

  -0.0118 - 0.0740i  -0.4858 + 0.4792i  -0.0096 - 0.0424i   0.0057 + 0.0174i
  -0.4850 + 0.4799i  -0.0717 - 0.0291i   0.0030 + 0.0178i  -0.0088 - 0.0450i
  -0.0094 - 0.0425i   0.0030 + 0.0178i  -0.0289 - 0.0615i  -0.4814 + 0.4835i
   0.0057 + 0.0175i  -0.0090 - 0.0450i  -0.4814 + 0.4836i  -0.0590 - 0.0403i


s4p_converted_back(:,:,35) =

  -0.0332 - 0.0572i  -0.2395 + 0.6356i   0.0006 - 0.0233i   0.0132 + 0.0158i
  -0.2392 + 0.6348i  -0.0694 - 0.0052i   0.0112 + 0.0171i   0.0009 - 0.0295i
   0.0007 - 0.0232i   0.0113 + 0.0171i  -0.0463 - 0.0421i  -0.2334 + 0.6370i
   0.0131 + 0.0157i   0.0008 - 0.0295i  -0.2339 + 0.6368i  -0.0644 - 0.0190i


s4p_converted_back(:,:,36) =

  -0.0433 - 0.0342i   0.0479 + 0.6736i   0.0197 - 0.0200i   0.0207 + 0.0102i
   0.0482 + 0.6728i  -0.0590 + 0.0108i   0.0199 + 0.0119i   0.0165 - 0.0301i
   0.0196 - 0.0201i   0.0200 + 0.0119i  -0.0528 - 0.0182i   0.0530 + 0.6732i
   0.0206 + 0.0102i   0.0164 - 0.0298i   0.0540 + 0.6716i  -0.0612 + 0.0005i


s4p_converted_back(:,:,37) =

  -0.0402 - 0.0128i   0.3240 + 0.5872i   0.0348 - 0.0347i   0.0261 - 0.0000i
   0.3234 + 0.5866i  -0.0464 + 0.0173i   0.0259 + 0.0015i   0.0251 - 0.0461i
   0.0347 - 0.0347i   0.0258 + 0.0016i  -0.0479 + 0.0028i   0.3279 + 0.5841i
   0.0260 + 0.0000i   0.0251 - 0.0460i   0.3275 + 0.5835i  -0.0521 + 0.0144i
```

```
s4p_converted_back(:,:,38) =

  -0.0289 - 0.0023i   0.5345 + 0.3935i   0.0345 - 0.0605i   0.0252 - 0.0141i
   0.5336 + 0.3935i  -0.0376 + 0.0151i   0.0252 - 0.0126i   0.0171 - 0.0690i
   0.0345 - 0.0603i   0.0253 - 0.0127i  -0.0362 + 0.0147i   0.5369 + 0.3902i
   0.0251 - 0.0141i   0.0173 - 0.0691i   0.5363 + 0.3898i  -0.0405 + 0.0205i


s4p_converted_back(:,:,39) =

  -0.0221 - 0.0024i   0.6400 + 0.1340i   0.0126 - 0.0830i   0.0139 - 0.0256i
   0.6399 + 0.1345i  -0.0389 + 0.0106i   0.0144 - 0.0240i  -0.0101 - 0.0841i
   0.0128 - 0.0829i   0.0143 - 0.0240i  -0.0265 + 0.0168i   0.6429 + 0.1311i
   0.0139 - 0.0255i  -0.0101 - 0.0842i   0.6422 + 0.1310i  -0.0326 + 0.0201i


s4p_converted_back(:,:,40) =

  -0.0164 - 0.0045i   0.6355 - 0.1371i  -0.0166 - 0.0879i   0.0028 - 0.0273i
   0.6355 - 0.1370i  -0.0396 + 0.0117i   0.0039 - 0.0256i  -0.0403 - 0.0796i
  -0.0162 - 0.0880i   0.0038 - 0.0258i  -0.0191 + 0.0140i   0.6361 - 0.1447i
   0.0029 - 0.0274i  -0.0403 - 0.0796i   0.6363 - 0.1441i  -0.0260 + 0.0166i


s4p_converted_back(:,:,41) =

  -0.0225 - 0.0161i   0.5202 - 0.3888i  -0.0482 - 0.0814i  -0.0100 - 0.0280i
   0.5197 - 0.3881i  -0.0464 + 0.0081i  -0.0079 - 0.0267i  -0.0685 - 0.0635i
  -0.0482 - 0.0814i  -0.0079 - 0.0267i  -0.0230 + 0.0057i   0.5160 - 0.3942i
  -0.0100 - 0.0281i  -0.0686 - 0.0635i   0.5160 - 0.3942i  -0.0293 + 0.0071i


s4p_converted_back(:,:,42) =

  -0.0433 - 0.0177i   0.3090 - 0.5661i  -0.0749 - 0.0580i  -0.0225 - 0.0206i
   0.3086 - 0.5655i  -0.0608 + 0.0134i  -0.0195 - 0.0204i  -0.0873 - 0.0333i
  -0.0749 - 0.0578i  -0.0197 - 0.0203i  -0.0378 + 0.0063i   0.3038 - 0.5689i
  -0.0225 - 0.0205i  -0.0871 - 0.0332i   0.3034 - 0.5691i  -0.0435 + 0.0045i


s4p_converted_back(:,:,43) =

  -0.0667 - 0.0012i   0.0446 - 0.6389i  -0.0870 - 0.0247i  -0.0295 - 0.0078i
```

```
   0.0451 - 0.6381i  -0.0738 + 0.0307i  -0.0265 - 0.0094i  -0.0893 + 0.0024i
  -0.0866 - 0.0244i  -0.0266 - 0.0094i  -0.0535 + 0.0214i   0.0393 - 0.6400i
  -0.0295 - 0.0078i  -0.0892 + 0.0022i   0.0392 - 0.6393i  -0.0617 + 0.0144i


s4p_converted_back(:,:,44) =

  -0.0786 + 0.0321i  -0.2223 - 0.5959i  -0.0819 + 0.0093i  -0.0292 + 0.0063i
  -0.2221 - 0.5952i  -0.0770 + 0.0586i  -0.0273 + 0.0033i  -0.0747 + 0.0336i
  -0.0816 + 0.0089i  -0.0274 + 0.0032i  -0.0587 + 0.0503i  -0.2279 - 0.5938i
  -0.0291 + 0.0062i  -0.0748 + 0.0334i  -0.2283 - 0.5929i  -0.0736 + 0.0385i


s4p_converted_back(:,:,45) =

  -0.0693 + 0.0712i  -0.4465 - 0.4468i  -0.0629 + 0.0342i  -0.0228 + 0.0177i
  -0.4456 - 0.4467i  -0.0635 + 0.0890i  -0.0227 + 0.0137i  -0.0496 + 0.0519i
  -0.0632 + 0.0337i  -0.0228 + 0.0137i  -0.0443 + 0.0823i  -0.4509 - 0.4424i
  -0.0227 + 0.0175i  -0.0496 + 0.0521i  -0.4500 - 0.4422i  -0.0703 + 0.0698i


s4p_converted_back(:,:,46) =

  -0.0389 + 0.1013i  -0.5866 - 0.2202i  -0.0385 + 0.0447i  -0.0131 + 0.0246i
  -0.5857 - 0.2207i  -0.0332 + 0.1098i  -0.0153 + 0.0211i  -0.0228 + 0.0550i
  -0.0388 + 0.0448i  -0.0152 + 0.0212i  -0.0113 + 0.1034i  -0.5892 - 0.2154i
  -0.0131 + 0.0245i  -0.0228 + 0.0550i  -0.5888 - 0.2145i  -0.0497 + 0.0970i


s4p_converted_back(:,:,47) =

   0.0029 + 0.1118i  -0.6210 + 0.0409i  -0.0175 + 0.0430i  -0.0018 + 0.0266i
  -0.6203 + 0.0403i   0.0040 + 0.1125i  -0.0055 + 0.0250i  -0.0039 + 0.0460i
  -0.0173 + 0.0431i  -0.0055 + 0.0251i   0.0296 + 0.1030i  -0.6213 + 0.0468i
  -0.0018 + 0.0265i  -0.0039 + 0.0460i  -0.6212 + 0.0471i  -0.0173 + 0.1105i


s4p_converted_back(:,:,48) =

   0.0441 + 0.0986i  -0.5477 + 0.2909i  -0.0029 + 0.0328i   0.0069 + 0.0244i
  -0.5468 + 0.2896i   0.0389 + 0.0987i   0.0031 + 0.0247i   0.0049 + 0.0343i
  -0.0029 + 0.0328i   0.0031 + 0.0247i   0.0642 + 0.0797i  -0.5450 + 0.2975i
   0.0068 + 0.0244i   0.0050 + 0.0344i  -0.5441 + 0.2976i   0.0176 + 0.1065i
```

```
s4p_converted_back(:,:,49) =

   0.0672 + 0.0679i  -0.3769 + 0.4907i  -0.0013 + 0.0193i   0.0155 + 0.0219i
  -0.3777 + 0.4901i   0.0619 + 0.0665i   0.0120 + 0.0240i   0.0044 + 0.0225i
  -0.0014 + 0.0193i   0.0120 + 0.0240i   0.0772 + 0.0423i  -0.3714 + 0.4957i
   0.0154 + 0.0219i   0.0044 + 0.0225i  -0.3708 + 0.4953i   0.0428 + 0.0850i


s4p_converted_back(:,:,50) =

   0.0692 + 0.0373i  -0.1383 + 0.6016i  -0.0102 + 0.0130i   0.0247 + 0.0156i
  -0.1385 + 0.6016i   0.0620 + 0.0295i   0.0226 + 0.0194i  -0.0046 + 0.0184i
  -0.0103 + 0.0129i   0.0226 + 0.0193i   0.0663 + 0.0085i  -0.1307 + 0.6033i
   0.0246 + 0.0154i  -0.0045 + 0.0183i  -0.1306 + 0.6033i   0.0503 + 0.0582i


s4p_converted_back(:,:,51) =

   0.0581 + 0.0185i   0.1251 + 0.6023i  -0.0221 + 0.0186i   0.0310 + 0.0038i
   0.1242 + 0.6018i   0.0422 + 0.0037i   0.0316 + 0.0085i  -0.0139 + 0.0259i
  -0.0221 + 0.0186i   0.0316 + 0.0086i   0.0402 - 0.0080i   0.1317 + 0.6002i
   0.0309 + 0.0039i  -0.0140 + 0.0258i   0.1319 + 0.6001i   0.0431 + 0.0389i


s4p_converted_back(:,:,52) =

   0.0461 + 0.0142i   0.3632 + 0.4930i  -0.0278 + 0.0359i   0.0311 - 0.0108i
   0.3629 + 0.4924i   0.0155 - 0.0026i   0.0344 - 0.0074i  -0.0153 + 0.0432i
  -0.0277 + 0.0360i   0.0344 - 0.0074i   0.0148 - 0.0029i   0.3687 + 0.4871i
   0.0311 - 0.0107i  -0.0154 + 0.0432i   0.3683 + 0.4875i   0.0315 + 0.0329i


s4p_converted_back(:,:,53) =

   0.0428 + 0.0187i   0.5317 + 0.2937i  -0.0203 + 0.0586i   0.0236 - 0.0245i
   0.5315 + 0.2940i  -0.0039 + 0.0089i   0.0280 - 0.0237i  -0.0030 + 0.0626i
  -0.0203 + 0.0584i   0.0281 - 0.0236i   0.0034 + 0.0158i   0.5344 + 0.2872i
   0.0235 - 0.0245i  -0.0031 + 0.0627i   0.5344 + 0.2874i   0.0264 + 0.0377i


s4p_converted_back(:,:,54) =

   0.0507 + 0.0216i   0.6008 + 0.0455i   0.0021 + 0.0769i   0.0099 - 0.0326i
```

```
   0.6001 + 0.0455i  -0.0081 + 0.0282i   0.0139 - 0.0346i   0.0226 + 0.0745i
   0.0018 + 0.0766i   0.0139 - 0.0346i   0.0094 + 0.0347i   0.6006 + 0.0388i
   0.0099 - 0.0325i   0.0226 + 0.0746i   0.6006 + 0.0391i   0.0324 + 0.0435i


s4p_converted_back(:,:,55) =

   0.0642 + 0.0147i   0.5611 - 0.2061i   0.0341 + 0.0819i  -0.0056 - 0.0330i
   0.5597 - 0.2059i   0.0031 + 0.0436i  -0.0038 - 0.0366i   0.0545 + 0.0713i
   0.0337 + 0.0820i  -0.0038 - 0.0366i   0.0277 + 0.0426i   0.5582 - 0.2117i
  -0.0055 - 0.0330i   0.0546 + 0.0713i   0.5578 - 0.2109i   0.0459 + 0.0416i


s4p_converted_back(:,:,56) =

   0.0736 - 0.0038i   0.4223 - 0.4162i   0.0660 + 0.0701i  -0.0182 - 0.0265i
   0.4221 - 0.4155i   0.0219 + 0.0470i  -0.0189 - 0.0301i   0.0823 + 0.0514i
   0.0657 + 0.0702i  -0.0189 - 0.0302i   0.0468 + 0.0350i   0.4189 - 0.4205i
  -0.0181 - 0.0266i   0.0824 + 0.0513i   0.4178 - 0.4197i   0.0581 + 0.0288i


s4p_converted_back(:,:,57) =

   0.0722 - 0.0272i   0.2118 - 0.5495i   0.0891 + 0.0448i  -0.0263 - 0.0169i
   0.2120 - 0.5494i   0.0408 + 0.0376i  -0.0290 - 0.0188i   0.0979 + 0.0198i
   0.0892 + 0.0450i  -0.0290 - 0.0189i   0.0568 + 0.0166i   0.2067 - 0.5528i
  -0.0264 - 0.0168i   0.0978 + 0.0199i   0.2065 - 0.5514i   0.0617 + 0.0093i


s4p_converted_back(:,:,58) =

   0.0599 - 0.0488i  -0.0336 - 0.5852i   0.0976 + 0.0127i  -0.0305 - 0.0056i
  -0.0335 - 0.5845i   0.0514 + 0.0168i  -0.0331 - 0.0054i   0.0958 - 0.0154i
   0.0976 + 0.0127i  -0.0332 - 0.0055i   0.0545 - 0.0045i  -0.0396 - 0.5862i
  -0.0304 - 0.0056i   0.0960 - 0.0152i  -0.0401 - 0.5854i   0.0546 - 0.0104i


s4p_converted_back(:,:,59) =

   0.0401 - 0.0632i  -0.2719 - 0.5170i   0.0903 - 0.0174i  -0.0310 + 0.0063i
  -0.2717 - 0.5163i   0.0485 - 0.0075i  -0.0323 + 0.0080i   0.0770 - 0.0428i
   0.0900 - 0.0171i  -0.0322 + 0.0080i   0.0416 - 0.0202i  -0.2782 - 0.5151i
  -0.0308 + 0.0062i   0.0772 - 0.0429i  -0.2778 - 0.5146i   0.0385 - 0.0235i
```

```
s4p_converted_back(:,:,60) =

   0.0181 - 0.0690i  -0.4597 - 0.3560i   0.0717 - 0.0364i  -0.0268 + 0.0185i
  -0.4592 - 0.3556i   0.0329 - 0.0274i  -0.0265 + 0.0200i   0.0490 - 0.0542i
   0.0718 - 0.0361i  -0.0265 + 0.0200i   0.0243 - 0.0265i  -0.4649 - 0.3514i
  -0.0269 + 0.0184i   0.0492 - 0.0544i  -0.4642 - 0.3513i   0.0197 - 0.0265i


s4p_converted_back(:,:,61) =

  -0.0027 - 0.0681i  -0.5633 - 0.1328i   0.0511 - 0.0408i  -0.0174 + 0.0281i
  -0.5621 - 0.1322i   0.0096 - 0.0362i  -0.0165 + 0.0286i   0.0242 - 0.0469i
   0.0512 - 0.0406i  -0.0165 + 0.0286i   0.0090 - 0.0252i  -0.5661 - 0.1265i
  -0.0175 + 0.0282i   0.0243 - 0.0471i  -0.5655 - 0.1263i   0.0042 - 0.0214i


s4p_converted_back(:,:,62) =

  -0.0203 - 0.0623i  -0.5656 + 0.1129i   0.0389 - 0.0331i  -0.0056 + 0.0335i
  -0.5644 + 0.1119i  -0.0133 - 0.0353i  -0.0049 + 0.0331i   0.0142 - 0.0293i
   0.0390 - 0.0331i  -0.0048 + 0.0331i  -0.0012 - 0.0202i  -0.5655 + 0.1198i
  -0.0055 + 0.0334i   0.0142 - 0.0294i  -0.5649 + 0.1197i  -0.0053 - 0.0132i


s4p_converted_back(:,:,63) =

  -0.0350 - 0.0556i  -0.4648 + 0.3370i   0.0403 - 0.0241i   0.0083 + 0.0345i
  -0.4649 + 0.3357i  -0.0352 - 0.0252i   0.0086 + 0.0339i   0.0200 - 0.0128i
   0.0403 - 0.0241i   0.0086 + 0.0338i  -0.0085 - 0.0170i  -0.4608 + 0.3435i
   0.0083 + 0.0345i   0.0200 - 0.0128i  -0.4615 + 0.3438i  -0.0117 - 0.0057i


s4p_converted_back(:,:,64) =

  -0.0506 - 0.0472i  -0.2800 + 0.4967i   0.0515 - 0.0233i   0.0224 + 0.0296i
  -0.2801 + 0.4966i  -0.0500 - 0.0063i   0.0223 + 0.0288i   0.0381 - 0.0071i
   0.0514 - 0.0231i   0.0224 + 0.0289i  -0.0182 - 0.0147i  -0.2738 + 0.5016i
   0.0224 + 0.0295i   0.0382 - 0.0072i  -0.2734 + 0.5004i  -0.0176 + 0.0022i


s4p_converted_back(:,:,65) =

  -0.0657 - 0.0333i  -0.0462 + 0.5637i   0.0644 - 0.0360i   0.0342 + 0.0188i
```

```
  -0.0469 + 0.5636i  -0.0548 + 0.0158i   0.0338 + 0.0180i   0.0587 - 0.0192i
   0.0645 - 0.0356i   0.0338 + 0.0180i  -0.0314 - 0.0089i  -0.0393 + 0.5649i
   0.0342 + 0.0188i   0.0587 - 0.0193i  -0.0390 + 0.5649i  -0.0219 + 0.0124i


s4p_converted_back(:,:,66) =

  -0.0762 - 0.0136i   0.1906 + 0.5263i   0.0686 - 0.0609i   0.0412 + 0.0029i
   0.1906 + 0.5270i  -0.0497 + 0.0354i   0.0402 + 0.0021i   0.0684 - 0.0475i
   0.0689 - 0.0607i   0.0401 + 0.0021i  -0.0445 + 0.0042i   0.1975 + 0.5252i
   0.0413 + 0.0029i   0.0682 - 0.0475i   0.1972 + 0.5252i  -0.0229 + 0.0252i


s4p_converted_back(:,:,67) =

  -0.0785 + 0.0084i   0.3867 + 0.3949i   0.0565 - 0.0906i   0.0403 - 0.0165i
   0.3868 + 0.3948i  -0.0391 + 0.0478i   0.0385 - 0.0167i   0.0576 - 0.0816i
   0.0569 - 0.0906i   0.0384 - 0.0167i  -0.0530 + 0.0242i   0.3932 + 0.3921i
   0.0404 - 0.0166i   0.0576 - 0.0815i   0.3931 + 0.3912i  -0.0174 + 0.0383i


s4p_converted_back(:,:,68) =

  -0.0755 + 0.0296i   0.5062 + 0.1990i   0.0261 - 0.1119i   0.0281 - 0.0335i
   0.5063 + 0.1986i  -0.0303 + 0.0553i   0.0259 - 0.0320i   0.0250 - 0.1056i
   0.0263 - 0.1120i   0.0260 - 0.0320i  -0.0537 + 0.0492i   0.5130 + 0.1935i
   0.0282 - 0.0335i   0.0251 - 0.1054i   0.5125 + 0.1932i  -0.0058 + 0.0496i


s4p_converted_back(:,:,69) =

  -0.0600 + 0.0506i   0.5404 - 0.0206i  -0.0092 - 0.1142i   0.0131 - 0.0406i
   0.5397 - 0.0207i  -0.0160 + 0.0630i   0.0118 - 0.0375i  -0.0128 - 0.1068i
  -0.0089 - 0.1139i   0.0119 - 0.0376i  -0.0399 + 0.0750i   0.5442 - 0.0310i
   0.0131 - 0.0406i  -0.0127 - 0.1069i   0.5430 - 0.0311i   0.0159 + 0.0543i


s4p_converted_back(:,:,70) =

  -0.0386 + 0.0533i   0.4850 - 0.2405i  -0.0414 - 0.1050i  -0.0042 - 0.0457i
   0.4852 - 0.2402i  -0.0010 + 0.0570i  -0.0032 - 0.0417i  -0.0465 - 0.0946i
  -0.0408 - 0.1048i  -0.0031 - 0.0418i  -0.0163 + 0.0882i   0.4815 - 0.2516i
  -0.0042 - 0.0457i  -0.0468 - 0.0948i   0.4816 - 0.2514i   0.0381 + 0.0412i
```

```
s4p_converted_back(:,:,71) =

  -0.0275 + 0.0455i   0.3423 - 0.4194i  -0.0673 - 0.0827i  -0.0253 - 0.0415i
   0.3424 - 0.4193i   0.0030 + 0.0465i  -0.0214 - 0.0387i  -0.0722 - 0.0679i
  -0.0672 - 0.0830i  -0.0216 - 0.0387i   0.0072 + 0.0908i   0.3343 - 0.4266i
  -0.0253 - 0.0415i  -0.0723 - 0.0678i   0.3344 - 0.4250i   0.0481 + 0.0165i


s4p_converted_back(:,:,72) =

  -0.0280 + 0.0368i   0.1380 - 0.5235i  -0.0803 - 0.0535i  -0.0431 - 0.0265i
   0.1383 - 0.5228i  -0.0020 + 0.0414i  -0.0376 - 0.0266i  -0.0814 - 0.0330i
  -0.0803 - 0.0538i  -0.0377 - 0.0266i   0.0271 + 0.0847i   0.1282 - 0.5241i
  -0.0430 - 0.0265i  -0.0813 - 0.0329i   0.1284 - 0.5240i   0.0414 - 0.0099i


s4p_converted_back(:,:,73) =

  -0.0371 + 0.0348i  -0.0921 - 0.5322i  -0.0794 - 0.0257i  -0.0512 - 0.0042i
  -0.0920 - 0.5316i  -0.0068 + 0.0461i  -0.0467 - 0.0080i  -0.0720 - 0.0014i
  -0.0793 - 0.0258i  -0.0467 - 0.0079i   0.0412 + 0.0734i  -0.0995 - 0.5296i
  -0.0512 - 0.0042i  -0.0721 - 0.0014i  -0.1000 - 0.5289i   0.0209 - 0.0277i


s4p_converted_back(:,:,74) =

  -0.0479 + 0.0437i  -0.3048 - 0.4429i  -0.0692 - 0.0058i  -0.0480 + 0.0195i
  -0.3042 - 0.4426i  -0.0039 + 0.0559i  -0.0468 + 0.0133i  -0.0508 + 0.0168i
  -0.0690 - 0.0060i  -0.0470 + 0.0133i   0.0495 + 0.0601i  -0.3094 - 0.4390i
  -0.0479 + 0.0195i  -0.0509 + 0.0168i  -0.3088 - 0.4386i  -0.0054 - 0.0303i


s4p_converted_back(:,:,75) =

  -0.0524 + 0.0614i  -0.4582 - 0.2741i  -0.0566 + 0.0041i  -0.0337 + 0.0388i
  -0.4576 - 0.2740i   0.0088 + 0.0623i  -0.0371 + 0.0335i  -0.0289 + 0.0182i
  -0.0566 + 0.0038i  -0.0371 + 0.0335i   0.0522 + 0.0468i  -0.4617 - 0.2694i
  -0.0338 + 0.0388i  -0.0288 + 0.0182i  -0.4615 - 0.2697i  -0.0268 - 0.0182i


s4p_converted_back(:,:,76) =

  -0.0456 + 0.0825i  -0.5257 - 0.0596i  -0.0476 + 0.0073i  -0.0128 + 0.0480i
```

```
  -0.5251 - 0.0597i   0.0242 + 0.0588i  -0.0181 + 0.0464i  -0.0175 + 0.0080i
  -0.0476 + 0.0071i  -0.0181 + 0.0466i   0.0502 + 0.0357i  -0.5288 - 0.0538i
  -0.0128 + 0.0480i  -0.0175 + 0.0080i  -0.5287 - 0.0541i  -0.0371 + 0.0016i


s4p_converted_back(:,:,77) =

  -0.0278 + 0.0956i  -0.5008 + 0.1608i  -0.0430 + 0.0062i   0.0067 + 0.0478i
  -0.5010 + 0.1603i   0.0376 + 0.0472i   0.0022 + 0.0491i  -0.0176 - 0.0025i
  -0.0428 + 0.0060i   0.0022 + 0.0492i   0.0464 + 0.0262i  -0.5026 + 0.1691i
   0.0067 + 0.0477i  -0.0176 - 0.0025i  -0.5019 + 0.1691i  -0.0340 + 0.0206i


s4p_converted_back(:,:,78) =

  -0.0120 + 0.0985i  -0.3889 + 0.3542i  -0.0472 + 0.0057i   0.0254 + 0.0422i
  -0.3884 + 0.3529i   0.0421 + 0.0244i   0.0227 + 0.0455i  -0.0276 - 0.0095i
  -0.0471 + 0.0055i   0.0228 + 0.0456i   0.0375 + 0.0187i  -0.3854 + 0.3615i
   0.0255 + 0.0422i  -0.0277 - 0.0094i  -0.3853 + 0.3616i  -0.0251 + 0.0294i


s4p_converted_back(:,:,79) =

  -0.0008 + 0.0992i  -0.2054 + 0.4803i  -0.0558 + 0.0141i   0.0420 + 0.0286i
  -0.2052 + 0.4804i   0.0278 + 0.0019i   0.0417 + 0.0323i  -0.0444 - 0.0048i
  -0.0557 + 0.0139i   0.0418 + 0.0324i   0.0251 + 0.0185i  -0.1982 + 0.4866i
   0.0419 + 0.0287i  -0.0444 - 0.0048i  -0.1979 + 0.4867i  -0.0200 + 0.0310i


s4p_converted_back(:,:,80) =

   0.0083 + 0.1018i   0.0133 + 0.5192i  -0.0603 + 0.0325i   0.0515 + 0.0080i
   0.0128 + 0.5186i  -0.0007 - 0.0052i   0.0531 + 0.0104i  -0.0576 + 0.0143i
  -0.0603 + 0.0322i   0.0531 + 0.0104i   0.0153 + 0.0277i   0.0221 + 0.5219i
   0.0515 + 0.0080i  -0.0575 + 0.0143i   0.0220 + 0.5213i  -0.0215 + 0.0328i


s4p_converted_back(:,:,81) =

   0.0199 + 0.1076i   0.2261 + 0.4636i  -0.0543 + 0.0561i   0.0505 - 0.0157i
   0.2251 + 0.4635i  -0.0290 + 0.0091i   0.0524 - 0.0147i  -0.0576 + 0.0427i
  -0.0545 + 0.0559i   0.0524 - 0.0148i   0.0146 + 0.0429i   0.2343 + 0.4616i
   0.0504 - 0.0157i  -0.0577 + 0.0427i   0.2345 + 0.4614i  -0.0258 + 0.0408i
```

```
s4p_converted_back(:,:,82) =

   0.0390 + 0.1129i   0.3936 + 0.3260i  -0.0360 + 0.0771i   0.0383 - 0.0363i
   0.3936 + 0.3260i  -0.0426 + 0.0401i   0.0399 - 0.0365i  -0.0408 + 0.0701i
  -0.0361 + 0.0771i   0.0399 - 0.0366i   0.0257 + 0.0576i   0.4011 + 0.3206i
   0.0384 - 0.0362i  -0.0408 + 0.0701i   0.4016 + 0.3209i  -0.0258 + 0.0551i


s4p_converted_back(:,:,83) =

   0.0654 + 0.1100i   0.4891 + 0.1336i  -0.0081 + 0.0891i   0.0185 - 0.0487i
   0.4891 + 0.1336i  -0.0339 + 0.0741i   0.0191 - 0.0493i  -0.0104 + 0.0867i
  -0.0086 + 0.0889i   0.0190 - 0.0493i   0.0470 + 0.0630i   0.4942 + 0.1257i
   0.0185 - 0.0487i  -0.0104 + 0.0867i   0.4936 + 0.1256i  -0.0167 + 0.0711i


s4p_converted_back(:,:,84) =

   0.0930 + 0.0934i   0.4973 - 0.0790i   0.0223 + 0.0881i  -0.0040 - 0.0510i
   0.4967 - 0.0786i  -0.0064 + 0.0965i  -0.0040 - 0.0510i   0.0250 + 0.0863i
   0.0219 + 0.0879i  -0.0040 - 0.0510i   0.0705 + 0.0535i   0.4986 - 0.0886i
  -0.0040 - 0.0509i   0.0249 + 0.0864i   0.4987 - 0.0880i   0.0011 + 0.0820i


s4p_converted_back(:,:,85) =

   0.1126 + 0.0625i   0.4184 - 0.2739i   0.0490 + 0.0745i  -0.0242 - 0.0434i
   0.4178 - 0.2737i   0.0282 + 0.0986i  -0.0238 - 0.0428i   0.0547 + 0.0689i
   0.0487 + 0.0749i  -0.0239 - 0.0428i   0.0858 + 0.0301i   0.4163 - 0.2843i
  -0.0243 - 0.0434i   0.0549 + 0.0690i   0.4157 - 0.2841i   0.0233 + 0.0823i


s4p_converted_back(:,:,86) =

   0.1149 + 0.0238i   0.2684 - 0.4198i   0.0656 + 0.0524i  -0.0384 - 0.0293i
   0.2684 - 0.4192i   0.0550 + 0.0822i  -0.0372 - 0.0288i   0.0703 + 0.0417i
   0.0653 + 0.0528i  -0.0372 - 0.0288i   0.0846 + 0.0007i   0.2614 - 0.4283i
  -0.0383 - 0.0293i   0.0704 + 0.0416i   0.2612 - 0.4284i   0.0416 + 0.0716i


s4p_converted_back(:,:,87) =

   0.0977 - 0.0097i   0.0714 - 0.4926i   0.0693 + 0.0298i  -0.0462 - 0.0126i
```

```
   0.0716 - 0.4926i   0.0683 + 0.0578i  -0.0446 - 0.0130i   0.0708 + 0.0150i
   0.0692 + 0.0298i  -0.0446 - 0.0129i   0.0674 - 0.0232i   0.0614 - 0.4974i
  -0.0463 - 0.0126i   0.0707 + 0.0148i   0.0611 - 0.4974i   0.0517 + 0.0556i


s4p_converted_back(:,:,88) =

   0.0692 - 0.0279i  -0.1384 - 0.4775i   0.0641 + 0.0133i  -0.0487 + 0.0064i
  -0.1388 - 0.4774i   0.0666 + 0.0330i  -0.0471 + 0.0049i   0.0590 - 0.0045i
   0.0640 + 0.0134i  -0.0472 + 0.0049i   0.0409 - 0.0328i  -0.1505 - 0.4768i
  -0.0486 + 0.0064i   0.0589 - 0.0043i  -0.1502 - 0.4769i   0.0529 + 0.0395i


s4p_converted_back(:,:,89) =

   0.0407 - 0.0289i  -0.3245 - 0.3759i   0.0559 + 0.0062i  -0.0433 + 0.0262i
  -0.3242 - 0.3754i   0.0544 + 0.0177i  -0.0427 + 0.0240i   0.0421 - 0.0103i
   0.0560 + 0.0063i  -0.0428 + 0.0241i   0.0154 - 0.0269i  -0.3337 - 0.3693i
  -0.0432 + 0.0262i   0.0422 - 0.0102i  -0.3342 - 0.3696i   0.0484 + 0.0287i


s4p_converted_back(:,:,90) =

   0.0207 - 0.0180i  -0.4495 - 0.2056i   0.0516 + 0.0071i  -0.0292 + 0.0434i
  -0.4498 - 0.2050i   0.0418 + 0.0142i  -0.0298 + 0.0411i   0.0302 - 0.0023i
   0.0515 + 0.0071i  -0.0298 + 0.0410i  -0.0000 - 0.0097i  -0.4556 - 0.1962i
  -0.0291 + 0.0434i   0.0303 - 0.0023i  -0.4553 - 0.1954i   0.0440 + 0.0247i


s4p_converted_back(:,:,91) =

   0.0124 - 0.0027i  -0.4909 + 0.0004i   0.0552 + 0.0121i  -0.0078 + 0.0523i
  -0.4909 + 0.0007i   0.0377 + 0.0173i  -0.0095 + 0.0505i   0.0328 + 0.0126i
   0.0551 + 0.0122i  -0.0095 + 0.0506i  -0.0022 + 0.0097i  -0.4919 + 0.0115i
  -0.0078 + 0.0522i   0.0329 + 0.0125i  -0.4925 + 0.0120i   0.0457 + 0.0226i


s4p_converted_back(:,:,92) =

   0.0159 + 0.0096i  -0.4437 + 0.2033i   0.0703 + 0.0141i   0.0149 + 0.0510i
  -0.4437 + 0.2033i   0.0404 + 0.0156i   0.0126 + 0.0503i   0.0503 + 0.0210i
   0.0702 + 0.0144i   0.0126 + 0.0505i   0.0068 + 0.0227i  -0.4400 + 0.2139i
   0.0149 + 0.0509i   0.0503 + 0.0209i  -0.4396 + 0.2135i   0.0501 + 0.0151i
```

```
s4p_converted_back(:,:,93) =

   0.0244 + 0.0110i  -0.3158 + 0.3677i   0.0920 + 0.0030i   0.0353 + 0.0404i
  -0.3157 + 0.3671i   0.0418 + 0.0075i   0.0330 + 0.0409i   0.0751 + 0.0152i
   0.0918 + 0.0033i   0.0332 + 0.0408i   0.0184 + 0.0242i  -0.3085 + 0.3746i
   0.0353 + 0.0403i   0.0751 + 0.0151i  -0.3089 + 0.3743i   0.0506 + 0.0020i


s4p_converted_back(:,:,94) =

   0.0274 + 0.0034i  -0.1331 + 0.4615i   0.1092 - 0.0239i   0.0491 + 0.0216i
  -0.1331 + 0.4609i   0.0373 - 0.0040i   0.0474 + 0.0230i   0.0973 - 0.0079i
   0.1093 - 0.0236i   0.0475 + 0.0230i   0.0241 + 0.0173i  -0.1243 + 0.4645i
   0.0491 + 0.0217i   0.0974 - 0.0077i  -0.1241 + 0.4640i   0.0428 - 0.0119i


s4p_converted_back(:,:,95) =

   0.0211 - 0.0060i   0.0698 + 0.4691i   0.1128 - 0.0622i   0.0536 - 0.0009i
   0.0695 + 0.4691i   0.0246 - 0.0132i   0.0525 + 0.0008i   0.1063 - 0.0447i
   0.1129 - 0.0617i   0.0525 + 0.0009i   0.0205 + 0.0092i   0.0777 + 0.4689i
   0.0536 - 0.0009i   0.1066 - 0.0448i   0.0780 + 0.4689i   0.0271 - 0.0200i


s4p_converted_back(:,:,96) =

   0.0062 - 0.0094i   0.2535 + 0.3931i   0.0965 - 0.1025i   0.0481 - 0.0234i
   0.2541 + 0.3934i   0.0060 - 0.0145i   0.0476 - 0.0215i   0.0944 - 0.0864i
   0.0970 - 0.1025i   0.0476 - 0.0215i   0.0094 + 0.0072i   0.2614 + 0.3911i
   0.0481 - 0.0234i   0.0947 - 0.0865i   0.2619 + 0.3914i   0.0084 - 0.0174i


s4p_converted_back(:,:,97) =

  -0.0113 - 0.0008i   0.3857 + 0.2521i   0.0602 - 0.1321i   0.0329 - 0.0404i
   0.3855 + 0.2524i  -0.0130 - 0.0030i   0.0330 - 0.0381i   0.0602 - 0.1174i
   0.0609 - 0.1318i   0.0330 - 0.0381i  -0.0023 + 0.0163i   0.3943 + 0.2477i
   0.0328 - 0.0405i   0.0604 - 0.1177i   0.3942 + 0.2478i  -0.0056 - 0.0025i


s4p_converted_back(:,:,98) =

  -0.0179 + 0.0214i   0.4517 + 0.0764i   0.0170 - 0.1392i   0.0150 - 0.0475i
```

```
   0.4517 + 0.0766i  -0.0188 + 0.0216i   0.0162 - 0.0450i   0.0181 - 0.1255i
   0.0174 - 0.1392i   0.0162 - 0.0452i  -0.0040 + 0.0353i   0.4592 + 0.0662i
   0.0150 - 0.0475i   0.0182 - 0.1257i   0.4587 + 0.0659i  -0.0045 + 0.0193i


s4p_converted_back(:,:,99) =

  -0.0063 + 0.0387i   0.4452 - 0.1147i  -0.0198 - 0.1309i  -0.0021 - 0.0519i
   0.4453 - 0.1143i  -0.0057 + 0.0392i   0.0003 - 0.0499i  -0.0186 - 0.1175i
  -0.0195 - 0.1310i   0.0003 - 0.0501i   0.0089 + 0.0510i   0.4455 - 0.1277i
  -0.0021 - 0.0520i  -0.0186 - 0.1177i   0.4453 - 0.1284i   0.0118 + 0.0323i


s4p_converted_back(:,:,100) =

   0.0092 + 0.0432i   0.3582 - 0.2881i  -0.0492 - 0.1118i  -0.0245 - 0.0503i
   0.3581 - 0.2882i   0.0115 + 0.0435i  -0.0210 - 0.0498i  -0.0474 - 0.0965i
  -0.0488 - 0.1122i  -0.0210 - 0.0498i   0.0284 + 0.0568i   0.3522 - 0.3004i
  -0.0245 - 0.0503i  -0.0473 - 0.0964i   0.3523 - 0.3003i   0.0318 + 0.0311i


s4p_converted_back(:,:,101) =

   0.0225 + 0.0394i   0.2069 - 0.4094i  -0.0658 - 0.0852i  -0.0459 - 0.0367i
   0.2072 - 0.4086i   0.0274 + 0.0367i  -0.0424 - 0.0382i  -0.0605 - 0.0678i
  -0.0656 - 0.0857i  -0.0424 - 0.0381i   0.0492 + 0.0518i   0.1959 - 0.4171i
  -0.0460 - 0.0366i  -0.0606 - 0.0680i   0.1957 - 0.4171i   0.0480 + 0.0164i


s4p_converted_back(:,:,102) =

   0.0325 + 0.0280i   0.0197 - 0.4572i  -0.0677 - 0.0611i  -0.0594 - 0.0138i
   0.0193 - 0.4572i   0.0347 + 0.0200i  -0.0571 - 0.0172i  -0.0586 - 0.0433i
  -0.0676 - 0.0612i  -0.0572 - 0.0170i   0.0658 + 0.0358i   0.0057 - 0.4592i
  -0.0595 - 0.0138i  -0.0585 - 0.0433i   0.0060 - 0.4592i   0.0522 - 0.0083i


s4p_converted_back(:,:,103) =

   0.0329 + 0.0115i  -0.1723 - 0.4239i  -0.0622 - 0.0469i  -0.0615 + 0.0144i
  -0.1719 - 0.4235i   0.0296 + 0.0018i  -0.0613 + 0.0102i  -0.0479 - 0.0293i
  -0.0618 - 0.0469i  -0.0612 + 0.0102i   0.0723 + 0.0124i  -0.1839 - 0.4184i
  -0.0615 + 0.0143i  -0.0479 - 0.0293i  -0.1842 - 0.4189i   0.0399 - 0.0338i
```

```
s4p_converted_back(:,:,104) =

   0.0227 - 0.0016i  -0.3315 - 0.3117i  -0.0583 - 0.0425i  -0.0492 + 0.0413i
  -0.3315 - 0.3116i   0.0143 - 0.0117i  -0.0513 + 0.0377i  -0.0379 - 0.0284i
  -0.0577 - 0.0426i  -0.0513 + 0.0379i   0.0657 - 0.0116i  -0.3389 - 0.3028i
  -0.0492 + 0.0413i  -0.0379 - 0.0284i  -0.3385 - 0.3033i   0.0138 - 0.0504i


s4p_converted_back(:,:,105) =

   0.0083 - 0.0054i  -0.4273 - 0.1454i  -0.0609 - 0.0415i  -0.0251 + 0.0583i
  -0.4272 - 0.1456i  -0.0083 - 0.0153i  -0.0283 + 0.0568i  -0.0377 - 0.0348i
  -0.0607 - 0.0420i  -0.0283 + 0.0568i   0.0487 - 0.0287i  -0.4311 - 0.1353i
  -0.0252 + 0.0583i  -0.0379 - 0.0348i  -0.4311 - 0.1355i  -0.0188 - 0.0507i


s4p_converted_back(:,:,106) =

  -0.0022 - 0.0042i  -0.4467 + 0.0429i  -0.0689 - 0.0402i   0.0015 + 0.0622i
  -0.4462 + 0.0426i  -0.0291 - 0.0053i  -0.0012 + 0.0623i  -0.0476 - 0.0388i
  -0.0690 - 0.0405i  -0.0013 + 0.0623i   0.0278 - 0.0367i  -0.4471 + 0.0538i
   0.0014 + 0.0623i  -0.0478 - 0.0388i  -0.4470 + 0.0543i  -0.0462 - 0.0350i


s4p_converted_back(:,:,107) =

  -0.0139 - 0.0031i  -0.3880 + 0.2234i  -0.0844 - 0.0348i   0.0269 + 0.0565i
  -0.3881 + 0.2232i  -0.0419 + 0.0103i   0.0246 + 0.0571i  -0.0646 - 0.0362i
  -0.0844 - 0.0349i   0.0247 + 0.0571i   0.0046 - 0.0369i  -0.3826 + 0.2334i
   0.0269 + 0.0565i  -0.0644 - 0.0362i  -0.3830 + 0.2338i  -0.0620 - 0.0113i


s4p_converted_back(:,:,108) =

  -0.0293 + 0.0038i  -0.2589 + 0.3627i  -0.1029 - 0.0168i   0.0490 + 0.0404i
  -0.2591 + 0.3626i  -0.0495 + 0.0281i   0.0471 + 0.0416i  -0.0844 - 0.0213i
  -0.1027 - 0.0173i   0.0471 + 0.0416i  -0.0176 - 0.0267i  -0.2499 + 0.3696i
   0.0490 + 0.0404i  -0.0842 - 0.0215i  -0.2497 + 0.3697i  -0.0668 + 0.0141i


s4p_converted_back(:,:,109) =

  -0.0426 + 0.0204i  -0.0843 + 0.4340i  -0.1137 + 0.0142i   0.0622 + 0.0155i
```

```
  -0.0842 + 0.4345i  -0.0519 + 0.0487i   0.0606 + 0.0171i  -0.0972 + 0.0074i
  -0.1136 + 0.0136i   0.0606 + 0.0172i  -0.0330 - 0.0070i  -0.0743 + 0.4363i
   0.0622 + 0.0155i  -0.0976 + 0.0073i  -0.0737 + 0.4369i  -0.0619 + 0.0372i


s4p_converted_back(:,:,110) =

  -0.0472 + 0.0450i   0.1016 + 0.4266i  -0.1092 + 0.0516i   0.0627 - 0.0129i
   0.1018 + 0.4266i  -0.0465 + 0.0709i   0.0615 - 0.0106i  -0.0950 + 0.0438i
  -0.1096 + 0.0511i   0.0615 - 0.0105i  -0.0373 + 0.0170i   0.1116 + 0.4251i
   0.0627 - 0.0130i  -0.0953 + 0.0441i   0.1120 + 0.4250i  -0.0498 + 0.0554i


s4p_converted_back(:,:,111) =

  -0.0389 + 0.0718i   0.2662 + 0.3440i  -0.0879 + 0.0859i   0.0506 - 0.0383i
   0.2662 + 0.3447i  -0.0316 + 0.0913i   0.0502 - 0.0352i  -0.0747 + 0.0777i
  -0.0882 + 0.0856i   0.0503 - 0.0353i  -0.0294 + 0.0381i   0.2752 + 0.3395i
   0.0505 - 0.0383i  -0.0746 + 0.0783i   0.2754 + 0.3394i  -0.0330 + 0.0667i


s4p_converted_back(:,:,112) =

  -0.0175 + 0.0927i   0.3819 + 0.2030i  -0.0536 + 0.1084i   0.0290 - 0.0553i
   0.3813 + 0.2031i  -0.0079 + 0.1044i   0.0304 - 0.0517i  -0.0401 + 0.0995i
  -0.0540 + 0.1082i   0.0303 - 0.0519i  -0.0139 + 0.0505i   0.3881 + 0.1954i
   0.0290 - 0.0552i  -0.0398 + 0.0995i   0.3878 + 0.1959i  -0.0141 + 0.0700i


s4p_converted_back(:,:,113) =

   0.0120 + 0.1012i   0.4296 + 0.0282i  -0.0142 + 0.1147i   0.0034 - 0.0611i
   0.4291 + 0.0287i   0.0204 + 0.1056i   0.0067 - 0.0586i  -0.0002 + 0.1030i
  -0.0146 + 0.1146i   0.0066 - 0.0587i   0.0026 + 0.0521i   0.4336 + 0.0181i
   0.0034 - 0.0612i  -0.0001 + 0.1028i   0.4341 + 0.0188i   0.0023 + 0.0646i


s4p_converted_back(:,:,114) =

   0.0413 + 0.0943i   0.4019 - 0.1502i   0.0220 + 0.1051i  -0.0213 - 0.0565i
   0.4020 - 0.1499i   0.0459 + 0.0929i  -0.0171 - 0.0558i   0.0344 + 0.0888i
   0.0217 + 0.1052i  -0.0171 - 0.0559i   0.0141 + 0.0444i   0.4020 - 0.1623i
  -0.0214 - 0.0566i   0.0343 + 0.0889i   0.4017 - 0.1618i   0.0112 + 0.0528i
```

```
s4p_converted_back(:,:,115) =

   0.0616 + 0.0746i   0.3037 - 0.3017i   0.0481 + 0.0838i  -0.0420 - 0.0431i
   0.3039 - 0.3015i   0.0601 + 0.0689i  -0.0379 - 0.0446i   0.0555 + 0.0634i
   0.0477 + 0.0839i  -0.0380 - 0.0445i   0.0151 + 0.0327i   0.2975 - 0.3133i
  -0.0422 - 0.0432i   0.0556 + 0.0638i   0.2977 - 0.3138i   0.0096 + 0.0407i


s4p_converted_back(:,:,116) =

   0.0668 + 0.0496i   0.1521 - 0.3985i   0.0600 + 0.0580i  -0.0559 - 0.0226i
   0.1521 - 0.3985i   0.0568 + 0.0436i  -0.0527 - 0.0259i   0.0598 + 0.0371i
   0.0595 + 0.0584i  -0.0528 - 0.0258i   0.0053 + 0.0257i   0.1395 - 0.4068i
  -0.0560 - 0.0225i   0.0601 + 0.0373i   0.1397 - 0.4077i  -0.0014 + 0.0359i


s4p_converted_back(:,:,117) =

   0.0565 + 0.0316i  -0.0257 - 0.4258i   0.0586 + 0.0376i  -0.0609 + 0.0013i
  -0.0256 - 0.4258i   0.0413 + 0.0303i  -0.0596 - 0.0032i   0.0529 + 0.0204i
   0.0584 + 0.0379i  -0.0596 - 0.0031i  -0.0099 + 0.0311i  -0.0422 - 0.4265i
  -0.0609 + 0.0013i   0.0531 + 0.0203i  -0.0425 - 0.4264i  -0.0136 + 0.0446i


s4p_converted_back(:,:,118) =

   0.0430 + 0.0284i  -0.2008 - 0.3758i   0.0530 + 0.0280i  -0.0569 + 0.0268i
  -0.2006 - 0.3754i   0.0250 + 0.0306i  -0.0578 + 0.0220i   0.0442 + 0.0149i
   0.0530 + 0.0281i  -0.0578 + 0.0220i  -0.0188 + 0.0506i  -0.2150 - 0.3673i
  -0.0567 + 0.0268i   0.0444 + 0.0148i  -0.2151 - 0.3667i  -0.0173 + 0.0640i


s4p_converted_back(:,:,119) =

   0.0368 + 0.0343i  -0.3377 - 0.2558i   0.0511 + 0.0263i  -0.0410 + 0.0500i
  -0.3377 - 0.2557i   0.0154 + 0.0410i  -0.0442 + 0.0461i   0.0406 + 0.0184i
   0.0510 + 0.0264i  -0.0442 + 0.0462i  -0.0130 + 0.0764i  -0.3449 - 0.2419i
  -0.0409 + 0.0500i   0.0406 + 0.0183i  -0.3446 - 0.2414i  -0.0064 + 0.0866i


s4p_converted_back(:,:,120) =

   0.0397 + 0.0410i  -0.4097 - 0.0912i   0.0558 + 0.0275i  -0.0151 + 0.0636i
```

```
  -0.4106 - 0.0916i   0.0176 + 0.0543i  -0.0199 + 0.0618i   0.0470 + 0.0246i
   0.0555 + 0.0277i  -0.0199 + 0.0617i   0.0089 + 0.0978i  -0.4097 - 0.0769i
  -0.0151 + 0.0636i   0.0469 + 0.0246i  -0.4101 - 0.0774i   0.0199 + 0.1011i


s4p_converted_back(:,:,121) =

   0.0497 + 0.0431i  -0.4079 + 0.0861i   0.0682 + 0.0269i   0.0142 + 0.0636i
  -0.4083 + 0.0863i   0.0295 + 0.0614i   0.0090 + 0.0643i   0.0628 + 0.0261i
   0.0681 + 0.0274i   0.0091 + 0.0643i   0.0419 + 0.1054i  -0.4009 + 0.0974i
   0.0143 + 0.0636i   0.0628 + 0.0262i  -0.4018 + 0.0977i   0.0535 + 0.0985i


s4p_converted_back(:,:,122) =

   0.0623 + 0.0347i  -0.3331 + 0.2459i   0.0871 + 0.0181i   0.0397 + 0.0503i
  -0.3336 + 0.2460i   0.0444 + 0.0567i   0.0355 + 0.0536i   0.0841 + 0.0164i
   0.0872 + 0.0185i   0.0355 + 0.0536i   0.0753 + 0.0945i  -0.3226 + 0.2519i
   0.0398 + 0.0504i   0.0842 + 0.0163i  -0.3234 + 0.2516i   0.0823 + 0.0761i


s4p_converted_back(:,:,123) =

   0.0677 + 0.0165i  -0.1994 + 0.3585i   0.1051 - 0.0038i   0.0561 + 0.0281i
  -0.1994 + 0.3590i   0.0519 + 0.0413i   0.0541 + 0.0331i   0.1021 - 0.0073i
   0.1050 - 0.0037i   0.0542 + 0.0330i   0.0978 + 0.0682i  -0.1900 + 0.3588i
   0.0562 + 0.0280i   0.1021 - 0.0073i  -0.1898 + 0.3599i   0.0944 + 0.0403i


s4p_converted_back(:,:,124) =

   0.0602 - 0.0032i  -0.0328 + 0.4051i   0.1132 - 0.0374i   0.0610 + 0.0022i
  -0.0324 + 0.4061i   0.0469 + 0.0232i   0.0620 + 0.0072i   0.1084 - 0.0413i
   0.1132 - 0.0368i   0.0619 + 0.0071i   0.1023 + 0.0369i  -0.0252 + 0.4024i
   0.0611 + 0.0022i   0.1086 - 0.0413i  -0.0250 + 0.4033i   0.0845 + 0.0038i


s4p_converted_back(:,:,125) =

   0.0422 - 0.0152i   0.1359 + 0.3791i   0.1055 - 0.0752i   0.0551 - 0.0220i
   0.1362 + 0.3790i   0.0296 + 0.0112i   0.0580 - 0.0184i   0.0981 - 0.0780i
   0.1059 - 0.0744i   0.0581 - 0.0185i   0.0913 + 0.0123i   0.1408 + 0.3753i
   0.0552 - 0.0221i   0.0983 - 0.0782i   0.1415 + 0.3756i   0.0564 - 0.0197i
```

```
s4p_converted_back(:,:,126) =

   0.0211 - 0.0136i   0.2756 + 0.2867i   0.0808 - 0.1074i   0.0410 - 0.0408i
   0.2756 + 0.2866i   0.0059 + 0.0135i   0.0445 - 0.0391i   0.0711 - 0.1074i
   0.0815 - 0.1073i   0.0445 - 0.0390i   0.0733 + 0.0021i   0.2806 + 0.2837i
   0.0409 - 0.0409i   0.0711 - 0.1076i   0.2809 + 0.2834i   0.0220 - 0.0212i


s4p_converted_back(:,:,127) =

   0.0088 + 0.0013i   0.3652 + 0.1491i   0.0458 - 0.1257i   0.0225 - 0.0518i
   0.3657 + 0.1491i  -0.0117 + 0.0335i   0.0259 - 0.0515i   0.0348 - 0.1204i
   0.0460 - 0.1255i   0.0259 - 0.0514i   0.0601 + 0.0062i   0.3709 + 0.1434i
   0.0225 - 0.0519i   0.0347 - 0.1206i   0.3710 + 0.1432i  -0.0023 - 0.0017i


s4p_converted_back(:,:,128) =

   0.0137 + 0.0167i   0.3942 - 0.0129i   0.0100 - 0.1287i   0.0031 - 0.0576i
   0.3947 - 0.0126i  -0.0109 + 0.0607i   0.0065 - 0.0582i   0.0007 - 0.1174i
   0.0106 - 0.1285i   0.0065 - 0.0581i   0.0601 + 0.0139i   0.3966 - 0.0217i
   0.0031 - 0.0577i   0.0006 - 0.1174i   0.3971 - 0.0218i  -0.0052 + 0.0258i


s4p_converted_back(:,:,129) =

   0.0272 + 0.0183i   0.3532 - 0.1747i  -0.0218 - 0.1209i  -0.0204 - 0.0582i
   0.3536 - 0.1749i   0.0052 + 0.0812i  -0.0169 - 0.0597i  -0.0261 - 0.1029i
  -0.0212 - 0.1209i  -0.0170 - 0.0597i   0.0695 + 0.0152i   0.3513 - 0.1844i
  -0.0204 - 0.0582i  -0.0261 - 0.1030i   0.3513 - 0.1853i   0.0112 + 0.0456i


s4p_converted_back(:,:,130) =

   0.0361 + 0.0070i   0.2476 - 0.3042i  -0.0465 - 0.1038i  -0.0461 - 0.0471i
   0.2479 - 0.3045i   0.0303 + 0.0910i  -0.0428 - 0.0505i  -0.0410 - 0.0816i
  -0.0459 - 0.1041i  -0.0428 - 0.0506i   0.0814 + 0.0063i   0.2409 - 0.3118i
  -0.0462 - 0.0472i  -0.0410 - 0.0819i   0.2411 - 0.3127i   0.0384 + 0.0490i


s4p_converted_back(:,:,131) =

   0.0340 - 0.0097i   0.0991 - 0.3757i  -0.0605 - 0.0824i  -0.0649 - 0.0224i
```

```
   0.0992 - 0.3757i   0.0589 + 0.0863i  -0.0633 - 0.0282i  -0.0424 - 0.0626i
  -0.0601 - 0.0825i  -0.0634 - 0.0281i   0.0895 - 0.0128i   0.0897 - 0.3804i
  -0.0648 - 0.0223i  -0.0424 - 0.0626i   0.0895 - 0.3809i   0.0641 + 0.0312i


s4p_converted_back(:,:,132) =

   0.0214 - 0.0253i  -0.0627 - 0.3812i  -0.0634 - 0.0638i  -0.0688 + 0.0090i
  -0.0630 - 0.3812i   0.0819 + 0.0647i  -0.0711 + 0.0024i  -0.0368 - 0.0539i
  -0.0630 - 0.0640i  -0.0711 + 0.0025i   0.0878 - 0.0391i  -0.0745 - 0.3805i
  -0.0689 + 0.0091i  -0.0369 - 0.0540i  -0.0752 - 0.3808i   0.0742 - 0.0032i


s4p_converted_back(:,:,133) =

  -0.0031 - 0.0352i  -0.2131 - 0.3202i  -0.0627 - 0.0538i  -0.0580 + 0.0386i
  -0.2130 - 0.3202i   0.0886 + 0.0345i  -0.0642 + 0.0339i  -0.0342 - 0.0551i
  -0.0623 - 0.0541i  -0.0642 + 0.0338i   0.0723 - 0.0665i  -0.2227 - 0.3141i
  -0.0579 + 0.0387i  -0.0343 - 0.0551i  -0.2235 - 0.3141i   0.0604 - 0.0401i


s4p_converted_back(:,:,134) =

  -0.0341 - 0.0289i  -0.3243 - 0.2043i  -0.0650 - 0.0487i  -0.0349 + 0.0592i
  -0.3242 - 0.2036i   0.0795 + 0.0070i  -0.0432 + 0.0589i  -0.0398 - 0.0602i
  -0.0647 - 0.0490i  -0.0432 + 0.0589i   0.0443 - 0.0864i  -0.3302 - 0.1938i
  -0.0348 + 0.0594i  -0.0397 - 0.0602i  -0.3300 - 0.1940i   0.0265 - 0.0646i


s4p_converted_back(:,:,135) =

  -0.0589 - 0.0076i  -0.3792 - 0.0523i  -0.0715 - 0.0453i  -0.0068 + 0.0674i
  -0.3792 - 0.0522i   0.0613 - 0.0115i  -0.0141 + 0.0718i  -0.0528 - 0.0624i
  -0.0712 - 0.0454i  -0.0142 + 0.0719i   0.0100 - 0.0944i  -0.3797 - 0.0412i
  -0.0067 + 0.0674i  -0.0528 - 0.0624i  -0.3797 - 0.0410i  -0.0160 - 0.0685i


s4p_converted_back(:,:,136) =

  -0.0731 + 0.0218i  -0.3664 + 0.1093i  -0.0844 - 0.0392i   0.0217 + 0.0639i
  -0.3659 + 0.1097i   0.0389 - 0.0224i   0.0177 + 0.0722i  -0.0715 - 0.0587i
  -0.0840 - 0.0396i   0.0178 + 0.0722i  -0.0270 - 0.0905i  -0.3613 + 0.1185i
   0.0217 + 0.0638i  -0.0715 - 0.0588i  -0.3612 + 0.1187i  -0.0562 - 0.0521i
```

```
s4p_converted_back(:,:,137) =

  -0.0761 + 0.0544i  -0.2865 + 0.2506i  -0.1002 - 0.0236i   0.0463 + 0.0487i
  -0.2864 + 0.2507i   0.0124 - 0.0249i   0.0479 + 0.0584i  -0.0923 - 0.0444i
  -0.0999 - 0.0243i   0.0480 + 0.0584i  -0.0614 - 0.0722i  -0.2782 + 0.2559i
   0.0463 + 0.0487i  -0.0925 - 0.0443i  -0.2783 + 0.2558i  -0.0855 - 0.0191i


s4p_converted_back(:,:,138) =

  -0.0681 + 0.0861i  -0.1556 + 0.3445i  -0.1106 + 0.0027i   0.0620 + 0.0248i
  -0.1556 + 0.3445i  -0.0153 - 0.0160i   0.0693 + 0.0317i  -0.1082 - 0.0178i
  -0.1109 + 0.0019i   0.0693 + 0.0317i  -0.0867 - 0.0419i  -0.1474 + 0.3457i
   0.0620 + 0.0248i  -0.1081 - 0.0178i  -0.1470 + 0.3459i  -0.0973 + 0.0232i


s4p_converted_back(:,:,139) =

  -0.0502 + 0.1136i   0.0005 + 0.3750i  -0.1093 + 0.0352i   0.0660 - 0.0029i
   0.0004 + 0.3754i  -0.0382 + 0.0052i   0.0762 - 0.0020i  -0.1111 + 0.0168i
  -0.1094 + 0.0348i   0.0761 - 0.0020i  -0.0984 - 0.0049i   0.0073 + 0.3736i
   0.0661 - 0.0031i  -0.1108 + 0.0165i   0.0079 + 0.3736i  -0.0899 + 0.0650i


s4p_converted_back(:,:,140) =

  -0.0242 + 0.1338i   0.1539 + 0.3396i  -0.0934 + 0.0661i   0.0587 - 0.0292i
   0.1541 + 0.3395i  -0.0500 + 0.0358i   0.0669 - 0.0346i  -0.0977 + 0.0508i
  -0.0934 + 0.0657i   0.0668 - 0.0346i  -0.0949 + 0.0323i   0.1595 + 0.3365i
   0.0586 - 0.0293i  -0.0977 + 0.0504i   0.1595 + 0.3360i  -0.0663 + 0.0971i


s4p_converted_back(:,:,141) =

   0.0072 + 0.1437i   0.2788 + 0.2442i  -0.0658 + 0.0877i   0.0420 - 0.0500i
   0.2788 + 0.2443i  -0.0466 + 0.0683i   0.0450 - 0.0589i  -0.0710 + 0.0750i
  -0.0662 + 0.0870i   0.0450 - 0.0588i  -0.0787 + 0.0623i   0.2826 + 0.2405i
   0.0419 - 0.0500i  -0.0714 + 0.0750i   0.2830 + 0.2401i  -0.0344 + 0.1135i


s4p_converted_back(:,:,142) =

   0.0395 + 0.1410i   0.3540 + 0.1069i  -0.0336 + 0.0947i   0.0190 - 0.0630i
```

```
   0.3535 + 0.1071i  -0.0295 + 0.0941i   0.0162 - 0.0711i  -0.0386 + 0.0844i
  -0.0343 + 0.0944i   0.0162 - 0.0711i  -0.0549 + 0.0802i   0.3562 + 0.1027i
   0.0189 - 0.0631i  -0.0388 + 0.0845i   0.3561 + 0.1029i  -0.0036 + 0.1132i


s4p_converted_back(:,:,143) =

   0.0667 + 0.1260i   0.3645 - 0.0486i  -0.0056 + 0.0886i  -0.0077 - 0.0666i
   0.3640 - 0.0487i  -0.0053 + 0.1075i  -0.0142 - 0.0710i  -0.0096 + 0.0788i
  -0.0061 + 0.0884i  -0.0143 - 0.0710i  -0.0319 + 0.0844i   0.3667 - 0.0541i
  -0.0078 - 0.0665i  -0.0095 + 0.0791i   0.3668 - 0.0538i   0.0179 + 0.1000i


s4p_converted_back(:,:,144) =

   0.0830 + 0.1024i   0.3099 - 0.1947i   0.0128 + 0.0744i  -0.0350 - 0.0597i
   0.3099 - 0.1948i   0.0174 + 0.1076i  -0.0419 - 0.0593i   0.0092 + 0.0641i
   0.0125 + 0.0743i  -0.0419 - 0.0593i  -0.0172 + 0.0793i   0.3104 - 0.2018i
  -0.0350 - 0.0597i   0.0093 + 0.0640i   0.3105 - 0.2017i   0.0255 + 0.0821i


s4p_converted_back(:,:,145) =

   0.0845 + 0.0773i   0.1995 - 0.3049i   0.0200 + 0.0596i  -0.0590 - 0.0412i
   0.1987 - 0.3044i   0.0303 + 0.0990i  -0.0637 - 0.0374i   0.0149 + 0.0479i
   0.0196 + 0.0596i  -0.0638 - 0.0373i  -0.0145 + 0.0729i   0.1961 - 0.3121i
  -0.0591 - 0.0411i   0.0150 + 0.0479i   0.1964 - 0.3119i   0.0184 + 0.0694i


s4p_converted_back(:,:,146) =

   0.0726 + 0.0602i   0.0540 - 0.3565i   0.0196 + 0.0508i  -0.0734 - 0.0117i
   0.0538 - 0.3557i   0.0306 + 0.0935i  -0.0749 - 0.0066i   0.0109 + 0.0403i
   0.0194 + 0.0509i  -0.0749 - 0.0065i  -0.0208 + 0.0743i   0.0468 - 0.3622i
  -0.0735 - 0.0117i   0.0109 + 0.0403i   0.0474 - 0.3625i   0.0047 + 0.0717i


s4p_converted_back(:,:,147) =

   0.0540 + 0.0587i  -0.0970 - 0.3430i   0.0175 + 0.0512i  -0.0724 + 0.0222i
  -0.0969 - 0.3426i   0.0288 + 0.1015i  -0.0709 + 0.0266i   0.0081 + 0.0451i
   0.0173 + 0.0512i  -0.0709 + 0.0267i  -0.0275 + 0.0893i  -0.1071 - 0.3443i
  -0.0725 + 0.0223i   0.0082 + 0.0453i  -0.1069 - 0.3448i  -0.0019 + 0.0904i
```

```
s4p_converted_back(:,:,148) =

   0.0427 + 0.0770i  -0.2281 - 0.2696i   0.0224 + 0.0598i  -0.0560 + 0.0528i
  -0.2281 - 0.2696i   0.0385 + 0.1179i  -0.0528 + 0.0550i   0.0162 + 0.0555i
   0.0221 + 0.0600i  -0.0527 + 0.0550i  -0.0227 + 0.1156i  -0.2371 - 0.2656i
  -0.0560 + 0.0527i   0.0161 + 0.0556i  -0.2370 - 0.2657i   0.0097 + 0.1159i


s4p_converted_back(:,:,149) =

   0.0530 + 0.1030i  -0.3153 - 0.1489i   0.0396 + 0.0677i  -0.0265 + 0.0730i
  -0.3154 - 0.1488i   0.0625 + 0.1320i  -0.0233 + 0.0731i   0.0348 + 0.0619i
   0.0392 + 0.0679i  -0.0232 + 0.0730i   0.0010 + 0.1417i  -0.3206 - 0.1422i
  -0.0265 + 0.0730i   0.0348 + 0.0620i  -0.3210 - 0.1424i   0.0401 + 0.1341i


s4p_converted_back(:,:,150) =

   0.0842 + 0.1194i  -0.3435 - 0.0055i   0.0648 + 0.0653i   0.0097 + 0.0764i
  -0.3431 - 0.0057i   0.0983 + 0.1346i   0.0120 + 0.0753i   0.0596 + 0.0585i
   0.0645 + 0.0657i   0.0120 + 0.0752i   0.0395 + 0.1546i  -0.3455 + 0.0010i
   0.0097 + 0.0766i   0.0596 + 0.0584i  -0.3459 + 0.0006i   0.0819 + 0.1346i


s4p_converted_back(:,:,151) =

   0.1248 + 0.1158i  -0.3107 + 0.1322i   0.0899 + 0.0505i   0.0419 + 0.0622i
  -0.3107 + 0.1323i   0.1393 + 0.1187i   0.0432 + 0.0603i   0.0842 + 0.0427i
   0.0897 + 0.0509i   0.0432 + 0.0604i   0.0829 + 0.1480i  -0.3122 + 0.1385i
   0.0420 + 0.0621i   0.0842 + 0.0427i  -0.3122 + 0.1386i   0.1233 + 0.1131i


s4p_converted_back(:,:,152) =

   0.1636 + 0.0903i  -0.2288 + 0.2431i   0.1099 + 0.0249i   0.0622 + 0.0359i
  -0.2286 + 0.2432i   0.1730 + 0.0814i   0.0626 + 0.0342i   0.1021 + 0.0157i
   0.1096 + 0.0254i   0.0627 + 0.0342i   0.1203 + 0.1218i  -0.2284 + 0.2508i
   0.0623 + 0.0358i   0.1022 + 0.0158i  -0.2286 + 0.2512i   0.1512 + 0.0725i


s4p_converted_back(:,:,153) =

   0.1873 + 0.0456i  -0.1109 + 0.3124i   0.1190 - 0.0092i   0.0685 + 0.0059i
```

```
  -0.1106 + 0.3125i   0.1872 + 0.0291i   0.0683 + 0.0042i   0.1082 - 0.0181i
   0.1191 - 0.0087i   0.0682 + 0.0042i   0.1424 + 0.0815i  -0.1080 + 0.3207i
   0.0684 + 0.0060i   0.1083 - 0.0180i  -0.1081 + 0.3211i   0.1576 + 0.0229i


s4p_converted_back(:,:,154) =

   0.1867 - 0.0065i   0.0243 + 0.3291i   0.1137 - 0.0456i   0.0619 - 0.0218i
   0.0241 + 0.3291i   0.1762 - 0.0257i   0.0610 - 0.0233i   0.1004 - 0.0524i
   0.1139 - 0.0450i   0.0610 - 0.0233i   0.1432 + 0.0377i   0.0309 + 0.3378i
   0.0619 - 0.0217i   0.1003 - 0.0525i   0.0315 + 0.3378i   0.1403 - 0.0224i


s4p_converted_back(:,:,155) =

   0.1613 - 0.0513i   0.1541 + 0.2909i   0.0943 - 0.0763i   0.0459 - 0.0424i
   0.1541 + 0.2914i   0.1411 - 0.0686i   0.0442 - 0.0436i   0.0791 - 0.0805i
   0.0945 - 0.0757i   0.0443 - 0.0438i   0.1254 + 0.0026i   0.1668 + 0.2958i
   0.0460 - 0.0425i   0.0793 - 0.0807i   0.1669 + 0.2962i   0.1066 - 0.0516i


s4p_converted_back(:,:,156) =

   0.1213 - 0.0758i   0.2587 + 0.2055i   0.0662 - 0.0951i   0.0254 - 0.0542i
   0.2593 + 0.2053i   0.0931 - 0.0882i   0.0231 - 0.0544i   0.0496 - 0.0959i
   0.0668 - 0.0944i   0.0231 - 0.0544i   0.0984 - 0.0156i   0.2747 + 0.2030i
   0.0254 - 0.0543i   0.0496 - 0.0958i   0.2747 + 0.2030i   0.0685 - 0.0589i


s4p_converted_back(:,:,157) =

   0.0825 - 0.0781i   0.3230 + 0.0829i   0.0377 - 0.1003i   0.0047 - 0.0585i
   0.3233 + 0.0830i   0.0494 - 0.0831i   0.0027 - 0.0574i   0.0211 - 0.0967i
   0.0384 - 0.1000i   0.0027 - 0.0574i   0.0741 - 0.0173i   0.3360 + 0.0715i
   0.0047 - 0.0585i   0.0210 - 0.0966i   0.3361 + 0.0711i   0.0400 - 0.0483i


s4p_converted_back(:,:,158) =

   0.0544 - 0.0681i   0.3316 - 0.0598i   0.0143 - 0.0961i  -0.0167 - 0.0583i
   0.3316 - 0.0595i   0.0205 - 0.0642i  -0.0176 - 0.0561i  -0.0002 - 0.0881i
   0.0149 - 0.0960i  -0.0176 - 0.0561i   0.0604 - 0.0099i   0.3361 - 0.0766i
  -0.0167 - 0.0582i  -0.0001 - 0.0879i   0.3362 - 0.0763i   0.0279 - 0.0320i
```

```
s4p_converted_back(:,:,159) =

   0.0376 - 0.0537i   0.2765 - 0.1945i  -0.0015 - 0.0866i  -0.0400 - 0.0502i
   0.2765 - 0.1946i   0.0077 - 0.0428i  -0.0397 - 0.0481i  -0.0119 - 0.0756i
  -0.0010 - 0.0866i  -0.0397 - 0.0481i   0.0584 - 0.0020i   0.2715 - 0.2105i
  -0.0401 - 0.0502i  -0.0117 - 0.0757i   0.2717 - 0.2110i   0.0296 - 0.0215i


s4p_converted_back(:,:,160) =

   0.0302 - 0.0414i   0.1680 - 0.2924i  -0.0089 - 0.0774i  -0.0608 - 0.0304i
   0.1674 - 0.2915i   0.0080 - 0.0281i  -0.0597 - 0.0292i  -0.0138 - 0.0658i
  -0.0085 - 0.0772i  -0.0597 - 0.0292i   0.0646 - 0.0018i   0.1561 - 0.3034i
  -0.0607 - 0.0304i  -0.0136 - 0.0660i   0.1561 - 0.3034i   0.0368 - 0.0243i


s4p_converted_back(:,:,161) =

   0.0278 - 0.0343i   0.0292 - 0.3318i  -0.0111 - 0.0729i  -0.0705 - 0.0004i
   0.0290 - 0.3318i   0.0123 - 0.0271i  -0.0696 + 0.0002i  -0.0111 - 0.0648i
  -0.0106 - 0.0728i  -0.0695 + 0.0002i   0.0709 - 0.0125i   0.0145 - 0.3366i
  -0.0705 - 0.0003i  -0.0109 - 0.0652i   0.0145 - 0.3370i   0.0375 - 0.0408i


s4p_converted_back(:,:,162) =

   0.0268 - 0.0332i  -0.1085 - 0.3100i  -0.0127 - 0.0751i  -0.0637 + 0.0314i
  -0.1086 - 0.3096i   0.0076 - 0.0385i  -0.0627 + 0.0319i  -0.0131 - 0.0738i
  -0.0122 - 0.0751i  -0.0627 + 0.0319i   0.0698 - 0.0317i  -0.1245 - 0.3081i
  -0.0637 + 0.0314i  -0.0133 - 0.0743i  -0.1245 - 0.3085i   0.0223 - 0.0632i


s4p_converted_back(:,:,163) =

   0.0223 - 0.0387i  -0.2242 - 0.2370i  -0.0198 - 0.0836i  -0.0434 + 0.0554i
  -0.2237 - 0.2364i  -0.0141 - 0.0491i  -0.0419 + 0.0560i  -0.0267 - 0.0846i
  -0.0190 - 0.0837i  -0.0419 + 0.0560i   0.0569 - 0.0537i  -0.2376 - 0.2274i
  -0.0434 + 0.0555i  -0.0270 - 0.0850i  -0.2374 - 0.2275i  -0.0100 - 0.0785i


s4p_converted_back(:,:,164) =

   0.0093 - 0.0458i  -0.3001 - 0.1231i  -0.0366 - 0.0926i  -0.0157 + 0.0679i
```

```
  -0.2999 - 0.1235i  -0.0455 - 0.0466i  -0.0138 + 0.0675i  -0.0492 - 0.0888i
  -0.0361 - 0.0928i  -0.0138 + 0.0676i   0.0315 - 0.0717i  -0.3069 - 0.1095i
  -0.0157 + 0.0680i  -0.0497 - 0.0887i  -0.3065 - 0.1093i  -0.0508 - 0.0775i


s4p_converted_back(:,:,165) =

  -0.0128 - 0.0486i  -0.3230 + 0.0114i  -0.0637 - 0.0946i   0.0142 + 0.0682i
  -0.3226 + 0.0115i  -0.0765 - 0.0295i   0.0159 + 0.0666i  -0.0760 - 0.0825i
  -0.0631 - 0.0948i   0.0158 + 0.0666i  -0.0043 - 0.0794i  -0.3218 + 0.0262i
   0.0143 + 0.0682i  -0.0762 - 0.0821i  -0.3218 + 0.0260i  -0.0910 - 0.0590i


s4p_converted_back(:,:,166) =

  -0.0407 - 0.0399i  -0.2874 + 0.1429i  -0.0953 - 0.0820i   0.0418 + 0.0555i
  -0.2875 + 0.1427i  -0.1013 - 0.0007i   0.0424 + 0.0533i  -0.1018 - 0.0640i
  -0.0948 - 0.0824i   0.0424 + 0.0532i  -0.0448 - 0.0710i  -0.2806 + 0.1543i
   0.0418 + 0.0555i  -0.1018 - 0.0637i  -0.2802 + 0.1543i  -0.1227 - 0.0249i


s4p_converted_back(:,:,167) =

  -0.0668 - 0.0170i  -0.2014 + 0.2471i  -0.1212 - 0.0531i   0.0609 + 0.0320i
  -0.2017 + 0.2469i  -0.1159 + 0.0363i   0.0604 + 0.0298i  -0.1197 - 0.0340i
  -0.1207 - 0.0539i   0.0604 + 0.0298i  -0.0809 - 0.0451i  -0.1915 + 0.2534i
   0.0610 + 0.0319i  -0.1198 - 0.0339i  -0.1913 + 0.2531i  -0.1401 + 0.0190i


s4p_converted_back(:,:,168) =

  -0.0829 + 0.0186i  -0.0814 + 0.3056i  -0.1323 - 0.0137i   0.0675 + 0.0031i
  -0.0812 + 0.3056i  -0.1187 + 0.0772i   0.0660 + 0.0013i  -0.1247 + 0.0027i
  -0.1324 - 0.0145i   0.0660 + 0.0013i  -0.1041 - 0.0053i  -0.0710 + 0.3078i
   0.0673 + 0.0030i  -0.1246 + 0.0024i  -0.0712 + 0.3074i  -0.1405 + 0.0661i


s4p_converted_back(:,:,169) =

  -0.0829 + 0.0607i   0.0507 + 0.3099i  -0.1257 + 0.0267i   0.0611 - 0.0246i
   0.0506 + 0.3096i  -0.1088 + 0.1180i   0.0591 - 0.0257i  -0.1142 + 0.0384i
  -0.1257 + 0.0260i   0.0591 - 0.0257i  -0.1092 + 0.0401i   0.0600 + 0.3090i
   0.0610 - 0.0246i  -0.1141 + 0.0384i   0.0599 + 0.3087i  -0.1250 + 0.1086i
```

```
s4p_converted_back(:,:,170) =

  -0.0651 + 0.1009i   0.1721 + 0.2614i  -0.1035 + 0.0590i   0.0448 - 0.0459i
   0.1715 + 0.2609i  -0.0861 + 0.1542i   0.0423 - 0.0461i  -0.0907 + 0.0659i
  -0.1038 + 0.0582i   0.0424 - 0.0462i  -0.0953 + 0.0818i   0.1806 + 0.2574i
   0.0448 - 0.0460i  -0.0907 + 0.0658i   0.1807 + 0.2573i  -0.0975 + 0.1412i


s4p_converted_back(:,:,171) =

  -0.0313 + 0.1301i   0.2632 + 0.1673i  -0.0727 + 0.0768i   0.0227 - 0.0588i
   0.2632 + 0.1674i  -0.0541 + 0.1809i   0.0202 - 0.0580i  -0.0602 + 0.0800i
  -0.0733 + 0.0761i   0.0203 - 0.0579i  -0.0674 + 0.1110i   0.2708 + 0.1605i
   0.0227 - 0.0588i  -0.0604 + 0.0800i   0.2708 + 0.1604i  -0.0640 + 0.1601i


s4p_converted_back(:,:,172) =

   0.0112 + 0.1406i   0.3081 + 0.0436i  -0.0430 + 0.0792i  -0.0015 - 0.0627i
   0.3081 + 0.0439i  -0.0172 + 0.1951i  -0.0036 - 0.0609i  -0.0311 + 0.0799i
  -0.0434 + 0.0791i  -0.0036 - 0.0608i  -0.0348 + 0.1223i   0.3134 + 0.0335i
  -0.0016 - 0.0627i  -0.0312 + 0.0797i   0.3133 + 0.0339i  -0.0318 + 0.1648i


s4p_converted_back(:,:,173) =

   0.0504 + 0.1302i   0.2983 - 0.0873i  -0.0212 + 0.0710i  -0.0258 - 0.0580i
   0.2977 - 0.0869i   0.0190 + 0.1975i  -0.0271 - 0.0553i  -0.0101 + 0.0695i
  -0.0214 + 0.0709i  -0.0271 - 0.0551i  -0.0086 + 0.1176i   0.2987 - 0.1003i
  -0.0259 - 0.0580i  -0.0102 + 0.0695i   0.2985 - 0.0998i  -0.0081 + 0.1596i


s4p_converted_back(:,:,174) =

   0.0752 + 0.1044i   0.2342 - 0.2027i  -0.0105 + 0.0595i  -0.0480 - 0.0447i
   0.2341 - 0.2023i   0.0484 + 0.1896i  -0.0478 - 0.0412i  -0.0002 + 0.0560i
  -0.0108 + 0.0594i  -0.0479 - 0.0412i   0.0026 + 0.1049i   0.2285 - 0.2160i
  -0.0480 - 0.0447i  -0.0003 + 0.0560i   0.2281 - 0.2159i   0.0036 + 0.1522i


s4p_converted_back(:,:,175) =

   0.0793 + 0.0744i   0.1272 - 0.2801i  -0.0092 + 0.0526i  -0.0649 - 0.0218i
```

```
   0.1276 - 0.2799i   0.0675 + 0.1782i  -0.0631 - 0.0184i  -0.0003 + 0.0470i
  -0.0094 + 0.0525i  -0.0630 - 0.0183i  -0.0016 + 0.0978i   0.1144 - 0.2894i
  -0.0649 - 0.0218i  -0.0003 + 0.0470i   0.1146 - 0.2893i   0.0054 + 0.1516i


s4p_converted_back(:,:,176) =

   0.0649 + 0.0540i  -0.0005 - 0.3048i  -0.0103 + 0.0547i  -0.0705 + 0.0088i
  -0.0005 - 0.3041i   0.0785 + 0.1710i  -0.0668 + 0.0114i  -0.0034 + 0.0484i
  -0.0107 + 0.0546i  -0.0668 + 0.0114i  -0.0127 + 0.1064i  -0.0177 - 0.3057i
  -0.0705 + 0.0089i  -0.0035 + 0.0485i  -0.0175 - 0.3057i   0.0066 + 0.1647i


s4p_converted_back(:,:,177) =
```

References

Granberg, T., *Handbook of Digital Techniques for High-Speed Design*.Upper Saddle River, NJ: Prentice Hall, 2004.

See Also

`s2scc` | `s2scd` | `s2sdc` | `s2sdd` | `s2smm` | `snp2smp`

Introduced in R2009a

snp2smp

Convert and reorder single-ended N-port S-parameters to single-ended M-port S-parameters

Syntax

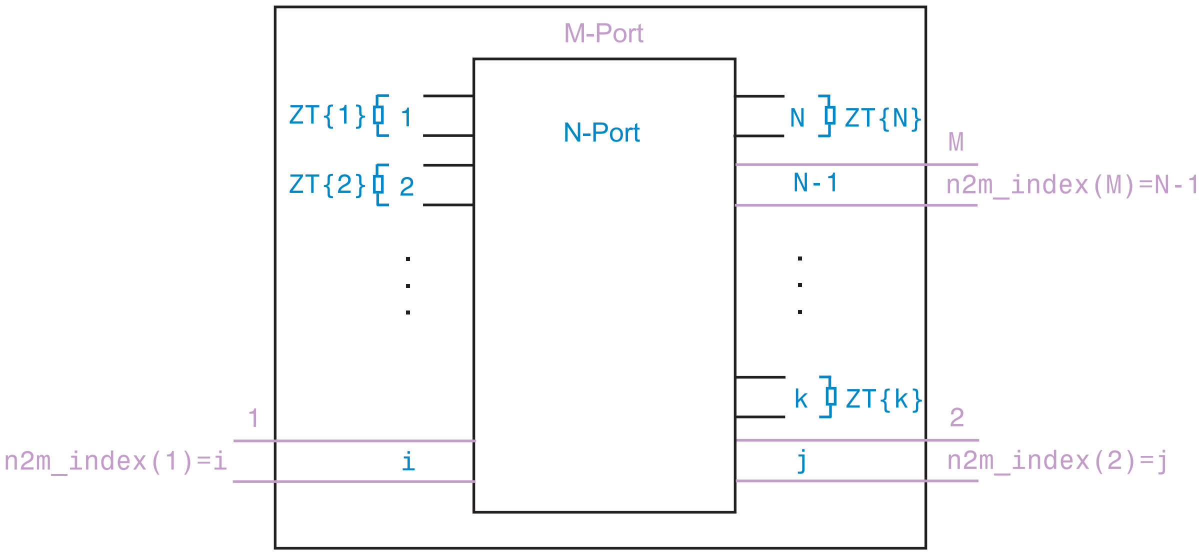
```
s_params_mp = snp2smp(s_params_np)
s_params_mp = snp2smp(s_params_np,Z0,n2m_index,ZT)
s_params_mp = snp2smp(s_obj,n2m_index,ZT)
```

Description

`s_params_mp = snp2smp(s_params_np)` convert and reorder the single-ended N-port S-parameters, `s_params_np`, into the single-ended M-port S-parameters, `s_params_mp`. *M* must be less than or equal to *N*.

`s_params_mp = snp2smp(s_params_np,Z0,n2m_index,ZT)` convert and reorder the S-parameter data using the optional arguments `Z0`, `n2m_index`, and `ZT` that control the conversion.

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



`s_params_mp = snp2smp(s_obj,n2m_index,ZT)` convert and reorder a S-parameters object, `s_obj`, into the single-ended M-port S-parameters, `s_params_mp`. *M* must be less than or equal to *N*.

Input Arguments

`s_params_np` — S-parameters
N-by-N-by-*K* array

S-parameters, specified as a N-by-N-by-*K* array representing *K* N-port S-parameters.

`s_obj` — S-parameter object
scalar handle objects

S-parameter object, specified as N-port scalar handle objects, which include numeric arrays of S-parameters.

`Z0` — Reference Impedance
50 (default) | scalar

Reference impedance in ohms, specified as a scalar, of the resulting S-parameters.

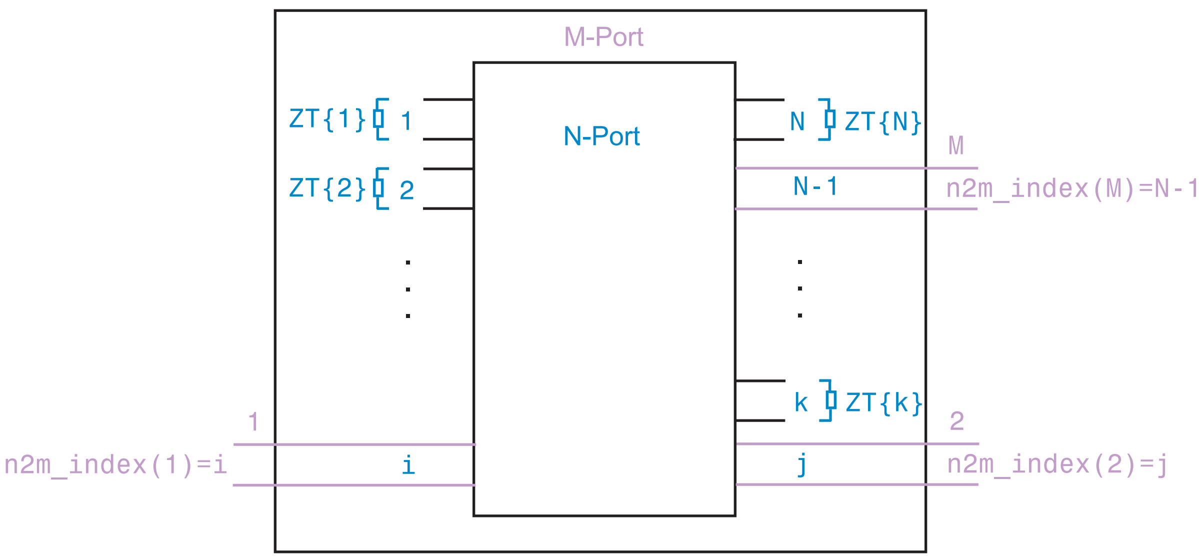
n2m_index — Mapping index
[1, 2] (default)

n2m_index is a vector of length *M* specifying how the ports of the N-port S-parameters map to the ports of the M-port S-parameters. n2m_index(i) is the index of the port from s_params_np that the function converts to the ith port of s_params_mp. For example, the setting [1, 2] means that *M* is 2, and the first two ports of the N-port S-parameters become the ports of the M-port parameters. The function terminates any additional ports with the impedances specified by ZT.

ZT —

ZT is a scalar, vector, or cell array specifying the termination impedance of the ports. If *M* is less than *N*, snp2smp terminates the *N*–*M* ports not listed in n2m_index using the values in ZT. If ZT is a scalar, the function terminates all *N*–*M* ports not listed in n2m_index by the same impedance ZT. If ZT is a vector of length *K*, ZT[i] is the impedance that terminates all *N*–*M* ports of the ith frequency point not listed in n2m_index. If ZT is a cell array of length *N*, ZT{j} is the impedance that terminates the jth port of the N-port S-parameters. The function ignores impedances related to the ports listed in n2m_index. Each ZT{j} can be a scalar or a vector of length *K*.

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



Examples

Swap Ports of S-Parameters

Convert 3-port S-parameters to 3-port S-parameters with port indices swapped from `[1 2 3]` to `[2 3 1]`.

```
ckt = read(rfckt.passive,'default.s3p');
```

Default.s3p represents a real counterclockwise circulator.

```
s3p = ckt.NetworkData.Data;
Z0 = ckt.NetworkData.Z0;
s3p_new = snp2smp(s3p,Z0,[2 3 1])
```

```
s3p_new =
s3p_new(:,:,1) =

   0.1431 - 0.7986i  -0.0701 + 0.4278i   0.0869 + 0.3238i
   0.0898 + 0.3177i   0.0503 - 0.8080i  -0.0285 + 0.4285i
  -0.0318 + 0.4208i   0.1114 + 0.3027i  -0.0073 - 0.8086i


s3p_new(:,:,2) =

   0.1175 - 0.8008i  -0.0610 + 0.4258i   0.0946 + 0.3169i
   0.1065 + 0.2991i   0.0243 - 0.8072i  -0.0202 + 0.4259i
  -0.0079 + 0.4073i   0.1249 + 0.2829i  -0.0330 - 0.8060i


s3p_new(:,:,3) =

   0.0913 - 0.8017i  -0.0467 + 0.4315i   0.1081 + 0.3133i
   0.1137 + 0.3057i  -0.0022 - 0.8051i  -0.0054 + 0.4295i
  -0.0050 + 0.4223i   0.1330 + 0.2889i  -0.0587 - 0.8021i


s3p_new(:,:,4) =

   0.0650 - 0.8016i  -0.0313 + 0.4378i   0.1224 + 0.3095i
   0.1199 + 0.3158i  -0.0287 - 0.8020i   0.0106 + 0.4335i
  -0.0049 + 0.4415i   0.1406 + 0.2985i  -0.0842 - 0.7972i
```

```
s3p_new(:,:,5) =

   0.0389 - 0.8007i  -0.0154 + 0.4435i   0.1366 + 0.3051i
   0.1262 + 0.3259i  -0.0549 - 0.7980i   0.0270 + 0.4369i
  -0.0047 + 0.4607i   0.1483 + 0.3080i  -0.1094 - 0.7916i


s3p_new(:,:,6) =

   0.0129 - 0.7989i   0.0008 + 0.4487i   0.1507 + 0.3000i
   0.1326 + 0.3359i  -0.0810 - 0.7931i   0.0436 + 0.4397i
  -0.0045 + 0.4799i   0.1563 + 0.3174i  -0.1344 - 0.7851i


s3p_new(:,:,7) =

  -0.0130 - 0.7962i   0.0173 + 0.4532i   0.1646 + 0.2943i
   0.1392 + 0.3459i  -0.1067 - 0.7874i   0.0605 + 0.4419i
  -0.0043 + 0.4990i   0.1644 + 0.3266i  -0.1591 - 0.7779i


s3p_new(:,:,8) =

  -0.0392 - 0.7925i   0.0251 + 0.4656i   0.1751 + 0.2959i
   0.1565 + 0.3286i  -0.1325 - 0.7806i   0.0705 + 0.4534i
   0.0207 + 0.4894i   0.1788 + 0.3082i  -0.1839 - 0.7695i


s3p_new(:,:,9) =

  -0.0652 - 0.7878i   0.0333 + 0.4778i   0.1859 + 0.2972i
   0.1723 + 0.3105i  -0.1580 - 0.7730i   0.0809 + 0.4648i
   0.0447 + 0.4786i   0.1916 + 0.2892i  -0.2083 - 0.7604i


s3p_new(:,:,10) =

  -0.0909 - 0.7824i   0.0419 + 0.4899i   0.1968 + 0.2982i
   0.1864 + 0.2918i  -0.1831 - 0.7645i   0.0918 + 0.4759i
   0.0677 + 0.4667i   0.2028 + 0.2699i  -0.2323 - 0.7505i


s3p_new(:,:,11) =
```

```
  -0.1163 - 0.7761i   0.0508 + 0.5019i   0.2079 + 0.2988i
   0.1989 + 0.2725i  -0.2078 - 0.7553i   0.1031 + 0.4867i
   0.0895 + 0.4536i   0.2124 + 0.2502i  -0.2560 - 0.7399i


s3p_new(:,:,12) =

  -0.1415 - 0.7689i   0.0601 + 0.5119i   0.2184 + 0.2979i
   0.2098 + 0.2565i  -0.2321 - 0.7451i   0.1143 + 0.4957i
   0.1084 + 0.4444i   0.2208 + 0.2339i  -0.2791 - 0.7284i


s3p_new(:,:,13) =

  -0.1670 - 0.7602i   0.0688 + 0.5097i   0.2239 + 0.2889i
   0.2197 + 0.2654i  -0.2566 - 0.7335i   0.1223 + 0.4933i
   0.1135 + 0.4681i   0.2320 + 0.2419i  -0.3019 - 0.7156i


s3p_new(:,:,14) =

  -0.1920 - 0.7507i   0.0775 + 0.5073i   0.2290 + 0.2799i
   0.2298 + 0.2743i  -0.2804 - 0.7212i   0.1303 + 0.4908i
   0.1186 + 0.4919i   0.2433 + 0.2498i  -0.3241 - 0.7021i


s3p_new(:,:,15) =

  -0.2166 - 0.7405i   0.0862 + 0.5047i   0.2338 + 0.2708i
   0.2400 + 0.2830i  -0.3038 - 0.7081i   0.1382 + 0.4882i
   0.1236 + 0.5158i   0.2547 + 0.2576i  -0.3458 - 0.6879i


s3p_new(:,:,16) =

  -0.2407 - 0.7294i   0.0947 + 0.5021i   0.2384 + 0.2616i
   0.2503 + 0.2917i  -0.3266 - 0.6942i   0.1460 + 0.4855i
   0.1285 + 0.5396i   0.2662 + 0.2651i  -0.3669 - 0.6731i


s3p_new(:,:,17) =

  -0.2644 - 0.7173i   0.1051 + 0.5000i   0.2440 + 0.2520i
```

```
   0.2577 + 0.2924i  -0.3489 - 0.6794i   0.1558 + 0.4829i
   0.1339 + 0.5517i   0.2740 + 0.2655i  -0.3875 - 0.6574i


s3p_new(:,:,18) =

  -0.2880 - 0.7039i   0.1213 + 0.5001i   0.2533 + 0.2412i
   0.2558 + 0.2702i  -0.3712 - 0.6631i   0.1717 + 0.4809i
   0.1405 + 0.5285i   0.2695 + 0.2451i  -0.4079 - 0.6403i


s3p_new(:,:,19) =

  -0.3110 - 0.6898i   0.1376 + 0.4996i   0.2623 + 0.2300i
   0.2526 + 0.2485i  -0.3927 - 0.6462i   0.1876 + 0.4785i
   0.1462 + 0.5053i   0.2639 + 0.2253i  -0.4276 - 0.6226i


s3p_new(:,:,20) =

  -0.3333 - 0.6749i   0.1540 + 0.4986i   0.2707 + 0.2185i
   0.2482 + 0.2274i  -0.4135 - 0.6285i   0.2035 + 0.4755i
   0.1509 + 0.4821i   0.2574 + 0.2061i  -0.4465 - 0.6043i


s3p_new(:,:,21) =

  -0.3550 - 0.6593i   0.1704 + 0.4970i   0.2786 + 0.2066i
   0.2426 + 0.2069i  -0.4335 - 0.6102i   0.2193 + 0.4719i
   0.1547 + 0.4590i   0.2500 + 0.1876i  -0.4647 - 0.5854i


s3p_new(:,:,22) =

  -0.3760 - 0.6425i   0.1869 + 0.4962i   0.2863 + 0.1949i
   0.2476 + 0.1933i  -0.4526 - 0.5908i   0.2355 + 0.4691i
   0.1688 + 0.4531i   0.2537 + 0.1743i  -0.4819 - 0.5655i


s3p_new(:,:,23) =

  -0.3962 - 0.6241i   0.2037 + 0.4968i   0.2941 + 0.1836i
   0.2719 + 0.1887i  -0.4705 - 0.5701i   0.2523 + 0.4677i
   0.2030 + 0.4744i   0.2772 + 0.1676i  -0.4981 - 0.5442i
```

```
s3p_new(:,:,24) =

  -0.4155 - 0.6051i   0.2206 + 0.4968i   0.3015 + 0.1720i
   0.2965 + 0.1819i  -0.4875 - 0.5488i   0.2692 + 0.4658i
   0.2397 + 0.4935i   0.3008 + 0.1587i  -0.5134 - 0.5225i


s3p_new(:,:,25) =

  -0.4340 - 0.5856i   0.2377 + 0.4963i   0.3084 + 0.1601i
   0.3211 + 0.1730i  -0.5036 - 0.5271i   0.2863 + 0.4633i
   0.2786 + 0.5100i   0.3243 + 0.1475i  -0.5278 - 0.5005i


s3p_new(:,:,26) =

  -0.4516 - 0.5655i   0.2550 + 0.4952i   0.3148 + 0.1479i
   0.3456 + 0.1618i  -0.5188 - 0.5049i   0.3034 + 0.4602i
   0.3196 + 0.5239i   0.3474 + 0.1342i  -0.5413 - 0.4780i


s3p_new(:,:,27) =

  -0.4686 - 0.5442i   0.2739 + 0.4950i   0.3216 + 0.1352i
   0.3549 + 0.1487i  -0.5332 - 0.4816i   0.3223 + 0.4576i
   0.3417 + 0.5241i   0.3556 + 0.1209i  -0.5539 - 0.4547i


s3p_new(:,:,28) =

  -0.4849 - 0.5217i   0.2946 + 0.4954i   0.3289 + 0.1220i
   0.3489 + 0.1356i  -0.5467 - 0.4574i   0.3433 + 0.4553i
   0.3434 + 0.5127i   0.3490 + 0.1096i  -0.5655 - 0.4306i


s3p_new(:,:,29) =

  -0.5001 - 0.4988i   0.3157 + 0.4950i   0.3356 + 0.1083i
   0.3424 + 0.1229i  -0.5591 - 0.4328i   0.3644 + 0.4521i
   0.3448 + 0.5014i   0.3420 + 0.0987i  -0.5762 - 0.4063i
```

```
s3p_new(:,:,30) =

  -0.5143 - 0.4755i   0.3370 + 0.4938i   0.3417 + 0.0943i
   0.3356 + 0.1107i  -0.5704 - 0.4080i   0.3857 + 0.4480i
   0.3461 + 0.4901i   0.3347 + 0.0882i  -0.5858 - 0.3817i


s3p_new(:,:,31) =

  -0.5275 - 0.4518i   0.3585 + 0.4917i   0.3472 + 0.0800i
   0.3283 + 0.0989i  -0.5807 - 0.3829i   0.4071 + 0.4430i
   0.3470 + 0.4789i   0.3271 + 0.0780i  -0.5944 - 0.3570i


s3p_new(:,:,32) =

  -0.5393 - 0.4270i   0.3734 + 0.4900i   0.3492 + 0.0682i
   0.3312 + 0.0831i  -0.5895 - 0.3569i   0.4225 + 0.4399i
   0.3675 + 0.4728i   0.3291 + 0.0630i  -0.6015 - 0.3314i


s3p_new(:,:,33) =

  -0.5499 - 0.4015i   0.3845 + 0.4886i   0.3490 + 0.0582i
   0.3394 + 0.0643i  -0.5968 - 0.3305i   0.4341 + 0.4381i
   0.4000 + 0.4682i   0.3360 + 0.0443i  -0.6071 - 0.3052i


s3p_new(:,:,34) =

  -0.5593 - 0.3758i   0.3955 + 0.4870i   0.3485 + 0.0482i
   0.3466 + 0.0447i  -0.6030 - 0.3039i   0.4458 + 0.4361i
   0.4328 + 0.4616i   0.3419 + 0.0250i  -0.6116 - 0.2791i


s3p_new(:,:,35) =

  -0.5676 - 0.3500i   0.4066 + 0.4852i   0.3477 + 0.0383i
   0.3526 + 0.0245i  -0.6080 - 0.2774i   0.4576 + 0.4338i
   0.4659 + 0.4530i   0.3467 + 0.0051i  -0.6150 - 0.2531i


s3p_new(:,:,36) =
```

```
  -0.5747 - 0.3242i   0.4178 + 0.4831i   0.3467 + 0.0284i
   0.3575 + 0.0038i  -0.6119 - 0.2510i   0.4694 + 0.4312i
   0.4991 + 0.4422i   0.3503 - 0.0152i  -0.6173 - 0.2273i


s3p_new(:,:,37) =

  -0.5797 - 0.2973i   0.4348 + 0.4748i   0.3446 + 0.0139i
   0.3545 - 0.0071i  -0.6137 - 0.2238i   0.4856 + 0.4211i
   0.5115 + 0.4381i   0.3474 - 0.0252i  -0.6176 - 0.2006i


s3p_new(:,:,38) =

  -0.5831 - 0.2702i   0.4535 + 0.4639i   0.3416 - 0.0019i
   0.3491 - 0.0143i  -0.6139 - 0.1965i   0.5031 + 0.4080i
   0.5169 + 0.4366i   0.3425 - 0.0315i  -0.6164 - 0.1739i


s3p_new(:,:,39) =

  -0.5854 - 0.2434i   0.4719 + 0.4522i   0.3379 - 0.0174i
   0.3435 - 0.0213i  -0.6130 - 0.1696i   0.5203 + 0.3942i
   0.5224 + 0.4351i   0.3375 - 0.0376i  -0.6142 - 0.1477i


s3p_new(:,:,40) =

  -0.5864 - 0.2168i   0.4901 + 0.4398i   0.3334 - 0.0326i
   0.3378 - 0.0281i  -0.6109 - 0.1430i   0.5370 + 0.3797i
   0.5278 + 0.4334i   0.3323 - 0.0435i  -0.6108 - 0.1218i


s3p_new(:,:,41) =

  -0.5863 - 0.1905i   0.5080 + 0.4267i   0.3283 - 0.0474i
   0.3319 - 0.0347i  -0.6077 - 0.1169i   0.5534 + 0.3645i
   0.5333 + 0.4318i   0.3271 - 0.0493i  -0.6063 - 0.0965i


s3p_new(:,:,42) =

  -0.5840 - 0.1634i   0.5315 + 0.4125i   0.3236 - 0.0642i
   0.3233 - 0.0554i  -0.6024 - 0.0904i   0.5751 + 0.3470i
```

```
   0.5559 + 0.4056i   0.3179 - 0.0687i  -0.5998 - 0.0712i


s3p_new(:,:,43) =

  -0.5803 - 0.1367i   0.5554 + 0.3970i   0.3181 - 0.0809i
   0.3127 - 0.0770i  -0.5957 - 0.0643i   0.5971 + 0.3281i
   0.5794 + 0.3746i   0.3068 - 0.0891i  -0.5921 - 0.0465i


s3p_new(:,:,44) =

  -0.5755 - 0.1106i   0.5790 + 0.3803i   0.3117 - 0.0973i
   0.3008 - 0.0975i  -0.5880 - 0.0390i   0.6185 + 0.3080i
   0.6013 + 0.3423i   0.2945 - 0.1083i  -0.5834 - 0.0225i


s3p_new(:,:,45) =

  -0.5695 - 0.0851i   0.6021 + 0.3623i   0.3045 - 0.1132i
   0.2876 - 0.1166i  -0.5792 - 0.0144i   0.6392 + 0.2866i
   0.6213 + 0.3086i   0.2810 - 0.1262i  -0.5737 + 0.0007i


s3p_new(:,:,46) =

  -0.5624 - 0.0603i   0.6247 + 0.3431i   0.2965 - 0.1286i
   0.2732 - 0.1343i  -0.5694 + 0.0093i   0.6592 + 0.2641i
   0.6394 + 0.2738i   0.2665 - 0.1427i  -0.5632 + 0.0231i


s3p_new(:,:,47) =

  -0.5530 - 0.0353i   0.6474 + 0.3267i   0.2880 - 0.1423i
   0.2686 - 0.1466i  -0.5573 + 0.0327i   0.6801 + 0.2442i
   0.6675 + 0.2650i   0.2623 - 0.1546i  -0.5499 + 0.0451i


s3p_new(:,:,48) =

  -0.5426 - 0.0112i   0.6697 + 0.3090i   0.2789 - 0.1556i
   0.2634 - 0.1586i  -0.5442 + 0.0550i   0.7006 + 0.2231i
   0.6956 + 0.2549i   0.2576 - 0.1663i  -0.5358 + 0.0659i
```

```
s3p_new(:,:,49) =

  -0.5311 + 0.0120i   0.6916 + 0.2902i   0.2691 - 0.1682i
   0.2577 - 0.1705i  -0.5302 + 0.0763i   0.7203 + 0.2009i
   0.7236 + 0.2435i   0.2523 - 0.1779i  -0.5209 + 0.0857i


s3p_new(:,:,50) =

  -0.5186 + 0.0342i   0.7129 + 0.2702i   0.2588 - 0.1803i
   0.2515 - 0.1822i  -0.5155 + 0.0965i   0.7394 + 0.1775i
   0.7515 + 0.2307i   0.2465 - 0.1893i  -0.5054 + 0.1045i


s3p_new(:,:,51) =

  -0.5051 + 0.0552i   0.7333 + 0.2497i   0.2480 - 0.1915i
   0.2435 - 0.1932i  -0.4999 + 0.1156i   0.7574 + 0.1537i
   0.7768 + 0.2151i   0.2391 - 0.2001i  -0.4890 + 0.1220i


s3p_new(:,:,52) =

  -0.4898 + 0.0747i   0.7505 + 0.2325i   0.2365 - 0.1996i
   0.2266 - 0.2004i  -0.4825 + 0.1329i   0.7735 + 0.1336i
   0.7850 + 0.1872i   0.2231 - 0.2067i  -0.4710 + 0.1379i


s3p_new(:,:,53) =

  -0.4737 + 0.0929i   0.7673 + 0.2145i   0.2248 - 0.2071i
   0.2097 - 0.2064i  -0.4646 + 0.1491i   0.7891 + 0.1128i
   0.7922 + 0.1591i   0.2071 - 0.2123i  -0.4525 + 0.1526i


s3p_new(:,:,54) =

  -0.4571 + 0.1100i   0.7836 + 0.1957i   0.2128 - 0.2140i
   0.1928 - 0.2111i  -0.4463 + 0.1640i   0.8041 + 0.0912i
   0.7984 + 0.1306i   0.1911 - 0.2168i  -0.4336 + 0.1660i


s3p_new(:,:,55) =
```

```
  -0.4398 + 0.1258i   0.7995 + 0.1762i   0.2008 - 0.2202i
   0.1761 - 0.2147i  -0.4275 + 0.1776i   0.8185 + 0.0688i
   0.8036 + 0.1020i   0.1752 - 0.2202i  -0.4144 + 0.1781i


s3p_new(:,:,56) =

  -0.4218 + 0.1400i   0.8134 + 0.1531i   0.1872 - 0.2257i
   0.1599 - 0.2176i  -0.4082 + 0.1896i   0.8300 + 0.0431i
   0.8101 + 0.0737i   0.1599 - 0.2231i  -0.3947 + 0.1888i


s3p_new(:,:,57) =

  -0.4029 + 0.1522i   0.8218 + 0.1210i   0.1698 - 0.2299i
   0.1450 - 0.2210i  -0.3883 + 0.1995i   0.8339 + 0.0097i
   0.8228 + 0.0465i   0.1459 - 0.2267i  -0.3742 + 0.1973i


s3p_new(:,:,58) =

  -0.3836 + 0.1630i   0.8288 + 0.0885i   0.1525 - 0.2329i
   0.1303 - 0.2236i  -0.3683 + 0.2081i   0.8364 - 0.0240i
   0.8345 + 0.0186i   0.1320 - 0.2294i  -0.3537 + 0.2045i


s3p_new(:,:,59) =

  -0.3642 + 0.1725i   0.8346 + 0.0556i   0.1353 - 0.2347i
   0.1157 - 0.2251i  -0.3482 + 0.2154i   0.8375 - 0.0579i
   0.8453 - 0.0101i   0.1181 - 0.2314i  -0.3332 + 0.2104i


s3p_new(:,:,60) =

  -0.3447 + 0.1808i   0.8391 + 0.0224i   0.1185 - 0.2353i
   0.1013 - 0.2258i  -0.3282 + 0.2215i   0.8373 - 0.0919i
   0.8551 - 0.0396i   0.1045 - 0.2325i  -0.3128 + 0.2150i


s3p_new(:,:,61) =

  -0.3251 + 0.1873i   0.8441 - 0.0132i   0.1016 - 0.2355i
```

```
   0.0856 - 0.2250i  -0.3083 + 0.2259i   0.8369 - 0.1285i
   0.8602 - 0.0730i   0.0895 - 0.2322i  -0.2928 + 0.2181i


s3p_new(:,:,62) =

  -0.3055 + 0.1919i   0.8503 - 0.0528i   0.0841 - 0.2354i
   0.0677 - 0.2218i  -0.2887 + 0.2284i   0.8368 - 0.1698i
   0.8576 - 0.1124i   0.0725 - 0.2296i  -0.2733 + 0.2196i


s3p_new(:,:,63) =

  -0.2860 + 0.1953i   0.8546 - 0.0931i   0.0670 - 0.2341i
   0.0507 - 0.2172i  -0.2694 + 0.2298i   0.8347 - 0.2114i
   0.8532 - 0.1516i   0.0562 - 0.2257i  -0.2540 + 0.2199i


s3p_new(:,:,64) =

  -0.2668 + 0.1975i   0.8570 - 0.1339i   0.0505 - 0.2316i
   0.0345 - 0.2114i  -0.2504 + 0.2300i   0.8304 - 0.2531i
   0.8470 - 0.1907i   0.0406 - 0.2208i  -0.2352 + 0.2191i


s3p_new(:,:,65) =

  -0.2478 + 0.1986i   0.8575 - 0.1750i   0.0346 - 0.2279i
   0.0195 - 0.2045i  -0.2317 + 0.2292i   0.8241 - 0.2949i
   0.8389 - 0.2295i   0.0259 - 0.2149i  -0.2167 + 0.2172i


s3p_new(:,:,66) =

  -0.2296 + 0.1984i   0.8581 - 0.2130i   0.0201 - 0.2232i
   0.0053 - 0.1977i  -0.2141 + 0.2272i   0.8188 - 0.3337i
   0.8346 - 0.2689i   0.0122 - 0.2091i  -0.1995 + 0.2138i


s3p_new(:,:,67) =

  -0.2122 + 0.1972i   0.8595 - 0.2479i   0.0071 - 0.2176i
  -0.0080 - 0.1911i  -0.1975 + 0.2241i   0.8150 - 0.3699i
   0.8339 - 0.3095i  -0.0007 - 0.2037i  -0.1834 + 0.2091i
```

```
s3p_new(:,:,68) =

  -0.1952 + 0.1950i   0.8595 - 0.2831i  -0.0052 - 0.2113i
  -0.0204 - 0.1837i  -0.1814 + 0.2201i   0.8095 - 0.4062i
   0.8312 - 0.3505i  -0.0128 - 0.1974i  -0.1678 + 0.2035i


s3p_new(:,:,69) =

  -0.1786 + 0.1918i   0.8580 - 0.3187i  -0.0167 - 0.2042i
  -0.0318 - 0.1756i  -0.1658 + 0.2154i   0.8025 - 0.4427i
   0.8265 - 0.3919i  -0.0242 - 0.1905i  -0.1527 + 0.1973i


s3p_new(:,:,70) =

  -0.1626 + 0.1877i   0.8550 - 0.3545i  -0.0275 - 0.1966i
  -0.0422 - 0.1667i  -0.1507 + 0.2098i   0.7938 - 0.4792i
   0.8197 - 0.4335i  -0.0348 - 0.1829i  -0.1382 + 0.1903i


s3p_new(:,:,71) =

  -0.1486 + 0.1828i   0.8456 - 0.3898i  -0.0377 - 0.1872i
  -0.0509 - 0.1562i  -0.1377 + 0.2036i   0.7787 - 0.5137i
   0.8090 - 0.4685i  -0.0436 - 0.1738i  -0.1259 + 0.1824i


s3p_new(:,:,72) =

  -0.1360 + 0.1773i   0.8318 - 0.4243i  -0.0471 - 0.1766i
  -0.0579 - 0.1448i  -0.1262 + 0.1968i   0.7592 - 0.5465i
   0.7958 - 0.4993i  -0.0509 - 0.1637i  -0.1150 + 0.1741i


s3p_new(:,:,73) =

  -0.1238 + 0.1712i   0.8166 - 0.4583i  -0.0553 - 0.1655i
  -0.0636 - 0.1332i  -0.1150 + 0.1896i   0.7384 - 0.5785i
   0.7814 - 0.5297i  -0.0572 - 0.1534i  -0.1045 + 0.1653i
```

```
s3p_new(:,:,74) =

  -0.1120 + 0.1646i   0.8001 - 0.4918i  -0.0624 - 0.1541i
  -0.0681 - 0.1214i  -0.1043 + 0.1820i   0.7162 - 0.6098i
   0.7659 - 0.5597i  -0.0626 - 0.1429i  -0.0945 + 0.1563i


s3p_new(:,:,75) =

  -0.1008 + 0.1574i   0.7821 - 0.5247i  -0.0682 - 0.1426i
  -0.0714 - 0.1097i  -0.0941 + 0.1739i   0.6928 - 0.6402i
   0.7492 - 0.5893i  -0.0669 - 0.1323i  -0.0848 + 0.1469i


s3p_new(:,:,76) =

  -0.0922 + 0.1499i   0.7621 - 0.5577i  -0.0737 - 0.1307i
  -0.0751 - 0.0965i  -0.0865 + 0.1658i   0.6670 - 0.6703i
   0.7234 - 0.6202i  -0.0713 - 0.1206i  -0.0784 + 0.1373i


s3p_new(:,:,77) =

  -0.0847 + 0.1423i   0.7404 - 0.5901i  -0.0782 - 0.1187i
  -0.0775 - 0.0830i  -0.0798 + 0.1576i   0.6397 - 0.6995i
   0.6935 - 0.6504i  -0.0747 - 0.1086i  -0.0729 + 0.1278i


s3p_new(:,:,78) =

  -0.0774 + 0.1344i   0.7174 - 0.6217i  -0.0814 - 0.1068i
  -0.0782 - 0.0699i  -0.0734 + 0.1493i   0.6111 - 0.7277i
   0.6624 - 0.6791i  -0.0767 - 0.0967i  -0.0675 + 0.1183i


s3p_new(:,:,79) =

  -0.0704 + 0.1264i   0.6930 - 0.6524i  -0.0833 - 0.0950i
  -0.0772 - 0.0575i  -0.0672 + 0.1409i   0.5812 - 0.7547i
   0.6302 - 0.7064i  -0.0776 - 0.0851i  -0.0620 + 0.1088i


s3p_new(:,:,80) =
```

```
  -0.0635 + 0.1182i   0.6673 - 0.6821i  -0.0840 - 0.0835i
  -0.0747 - 0.0457i  -0.0611 + 0.1324i   0.5502 - 0.7804i
   0.5968 - 0.7322i  -0.0772 - 0.0738i  -0.0566 + 0.0992i


s3p_new(:,:,81) =

  -0.0594 + 0.1104i   0.6386 - 0.7109i  -0.0847 - 0.0717i
  -0.0735 - 0.0338i  -0.0580 + 0.1244i   0.5160 - 0.8044i
   0.5683 - 0.7589i  -0.0770 - 0.0634i  -0.0553 + 0.0901i


s3p_new(:,:,82) =

  -0.0557 + 0.1027i   0.6085 - 0.7385i  -0.0842 - 0.0603i
  -0.0709 - 0.0226i  -0.0553 + 0.1167i   0.4806 - 0.8269i
   0.5395 - 0.7850i  -0.0758 - 0.0535i  -0.0541 + 0.0812i


s3p_new(:,:,83) =

  -0.0519 + 0.0951i   0.5773 - 0.7648i  -0.0825 - 0.0495i
  -0.0667 - 0.0126i  -0.0524 + 0.1089i   0.4442 - 0.8478i
   0.5095 - 0.8099i  -0.0736 - 0.0441i  -0.0523 + 0.0726i


s3p_new(:,:,84) =

  -0.0480 + 0.0875i   0.5448 - 0.7897i  -0.0796 - 0.0394i
  -0.0613 - 0.0040i  -0.0494 + 0.1012i   0.4068 - 0.8671i
   0.4783 - 0.8337i  -0.0705 - 0.0353i  -0.0500 + 0.0642i


s3p_new(:,:,85) =

  -0.0441 + 0.0799i   0.5113 - 0.8133i  -0.0756 - 0.0301i
  -0.0549 + 0.0032i  -0.0464 + 0.0935i   0.3686 - 0.8847i
   0.4460 - 0.8563i  -0.0665 - 0.0273i  -0.0471 + 0.0560i


s3p_new(:,:,86) =

  -0.0429 + 0.0734i   0.4798 - 0.8354i  -0.0724 - 0.0201i
  -0.0498 + 0.0134i  -0.0460 + 0.0871i   0.3331 - 0.9019i
```

```
   0.4108 - 0.8745i  -0.0629 - 0.0186i  -0.0493 + 0.0489i


s3p_new(:,:,87) =

  -0.0415 + 0.0671i   0.4474 - 0.8562i  -0.0680 - 0.0112i
  -0.0429 + 0.0218i  -0.0453 + 0.0807i   0.2968 - 0.9176i
   0.3748 - 0.8913i  -0.0583 - 0.0108i  -0.0506 + 0.0419i


s3p_new(:,:,88) =

  -0.0397 + 0.0610i   0.4141 - 0.8759i  -0.0626 - 0.0034i
  -0.0349 + 0.0279i  -0.0443 + 0.0744i   0.2597 - 0.9319i
   0.3382 - 0.9066i  -0.0528 - 0.0043i  -0.0510 + 0.0351i


s3p_new(:,:,89) =

  -0.0377 + 0.0549i   0.3799 - 0.8942i  -0.0564 + 0.0030i
  -0.0263 + 0.0319i  -0.0430 + 0.0682i   0.2220 - 0.9447i
   0.3009 - 0.9205i  -0.0466 + 0.0010i  -0.0507 + 0.0286i


s3p_new(:,:,90) =

  -0.0359 + 0.0491i   0.3448 - 0.9106i  -0.0498 + 0.0086i
  -0.0180 + 0.0341i  -0.0418 + 0.0623i   0.1836 - 0.9553i
   0.2632 - 0.9325i  -0.0402 + 0.0056i  -0.0502 + 0.0226i


s3p_new(:,:,91) =

  -0.0368 + 0.0443i   0.3080 - 0.9222i  -0.0437 + 0.0162i
  -0.0112 + 0.0390i  -0.0431 + 0.0576i   0.1446 - 0.9599i
   0.2262 - 0.9414i  -0.0344 + 0.0125i  -0.0529 + 0.0183i


s3p_new(:,:,92) =

  -0.0373 + 0.0396i   0.2708 - 0.9324i  -0.0366 + 0.0219i
  -0.0031 + 0.0424i  -0.0441 + 0.0530i   0.1055 - 0.9628i
   0.1889 - 0.9488i  -0.0277 + 0.0172i  -0.0553 + 0.0137i
```

```
s3p_new(:,:,93) =

  -0.0374 + 0.0350i   0.2333 - 0.9410i  -0.0291 + 0.0256i
   0.0058 + 0.0442i  -0.0447 + 0.0484i   0.0664 - 0.9642i
   0.1513 - 0.9547i  -0.0206 + 0.0198i  -0.0573 + 0.0088i


s3p_new(:,:,94) =

  -0.0370 + 0.0305i   0.1956 - 0.9481i  -0.0214 + 0.0274i
   0.0154 + 0.0439i  -0.0449 + 0.0439i   0.0274 - 0.9640i
   0.1136 - 0.9591i  -0.0138 + 0.0203i  -0.0588 + 0.0037i


s3p_new(:,:,95) =

  -0.0369 + 0.0264i   0.1576 - 0.9542i  -0.0145 + 0.0286i
   0.0245 + 0.0425i  -0.0454 + 0.0397i  -0.0118 - 0.9628i
   0.0750 - 0.9625i  -0.0080 + 0.0204i  -0.0604 - 0.0010i


s3p_new(:,:,96) =

  -0.0384 + 0.0226i   0.1191 - 0.9601i  -0.0081 + 0.0318i
   0.0320 + 0.0424i  -0.0473 + 0.0363i  -0.0517 - 0.9617i
   0.0340 - 0.9654i  -0.0025 + 0.0232i  -0.0631 - 0.0037i


s3p_new(:,:,97) =

  -0.0395 + 0.0187i   0.0804 - 0.9644i  -0.0009 + 0.0336i
   0.0399 + 0.0409i  -0.0489 + 0.0327i  -0.0916 - 0.9589i
  -0.0071 - 0.9666i   0.0038 + 0.0244i  -0.0658 - 0.0066i


s3p_new(:,:,98) =

  -0.0403 + 0.0149i   0.0416 - 0.9672i   0.0067 + 0.0337i
   0.0479 + 0.0382i  -0.0503 + 0.0291i  -0.1313 - 0.9544i
  -0.0483 - 0.9660i   0.0106 + 0.0238i  -0.0683 - 0.0098i


s3p_new(:,:,99) =
```

```
  -0.0407 + 0.0110i   0.0026 - 0.9684i   0.0142 + 0.0321i
   0.0557 + 0.0340i  -0.0515 + 0.0255i  -0.1709 - 0.9484i
  -0.0894 - 0.9636i   0.0174 + 0.0212i  -0.0707 - 0.0132i


s3p_new(:,:,100) =

  -0.0413 + 0.0077i  -0.0362 - 0.9679i   0.0209 + 0.0303i
   0.0627 + 0.0294i  -0.0529 + 0.0223i  -0.2098 - 0.9405i
  -0.1284 - 0.9592i   0.0230 + 0.0184i  -0.0732 - 0.0160i


s3p_new(:,:,101) =

  -0.0427 + 0.0051i  -0.0747 - 0.9657i   0.0269 + 0.0296i
   0.0686 + 0.0254i  -0.0550 + 0.0201i  -0.2480 - 0.9310i
  -0.1637 - 0.9528i   0.0279 + 0.0172i  -0.0763 - 0.0176i


s3p_new(:,:,102) =

  -0.0440 + 0.0024i  -0.1130 - 0.9620i   0.0330 + 0.0278i
   0.0743 + 0.0206i  -0.0570 + 0.0176i  -0.2857 - 0.9199i
  -0.1986 - 0.9450i   0.0327 + 0.0151i  -0.0793 - 0.0193i


s3p_new(:,:,103) =

  -0.0451 - 0.0006i  -0.1512 - 0.9568i   0.0391 + 0.0249i
   0.0796 + 0.0150i  -0.0589 + 0.0150i  -0.3229 - 0.9072i
  -0.2333 - 0.9360i   0.0374 + 0.0121i  -0.0823 - 0.0211i


s3p_new(:,:,104) =

  -0.0460 - 0.0036i  -0.1892 - 0.9500i   0.0449 + 0.0209i
   0.0845 + 0.0087i  -0.0607 + 0.0123i  -0.3595 - 0.8931i
  -0.2675 - 0.9258i   0.0418 + 0.0082i  -0.0853 - 0.0230i


s3p_new(:,:,105) =

  -0.0473 - 0.0060i  -0.2262 - 0.9417i   0.0500 + 0.0169i
```

```
   0.0884 + 0.0020i  -0.0627 + 0.0103i  -0.3950 - 0.8778i
  -0.3043 - 0.9144i   0.0455 + 0.0042i  -0.0882 - 0.0243i


s3p_new(:,:,106) =

  -0.0490 - 0.0078i  -0.2623 - 0.9320i   0.0543 + 0.0133i
   0.0915 - 0.0048i  -0.0651 + 0.0091i  -0.4293 - 0.8615i
  -0.3437 - 0.9017i   0.0486 + 0.0005i  -0.0909 - 0.0251i


s3p_new(:,:,107) =

  -0.0507 - 0.0097i  -0.2980 - 0.9208i   0.0583 + 0.0092i
   0.0940 - 0.0121i  -0.0675 + 0.0078i  -0.4629 - 0.8439i
  -0.3825 - 0.8872i   0.0514 - 0.0037i  -0.0936 - 0.0259i


s3p_new(:,:,108) =

  -0.0523 - 0.0117i  -0.3332 - 0.9083i   0.0621 + 0.0044i
   0.0959 - 0.0197i  -0.0698 + 0.0064i  -0.4959 - 0.8249i
  -0.4208 - 0.8710i   0.0539 - 0.0083i  -0.0964 - 0.0267i


s3p_new(:,:,109) =

  -0.0539 - 0.0138i  -0.3679 - 0.8945i   0.0654 - 0.0009i
   0.0971 - 0.0277i  -0.0721 + 0.0049i  -0.5280 - 0.8047i
  -0.4583 - 0.8533i   0.0559 - 0.0134i  -0.0991 - 0.0275i


s3p_new(:,:,110) =

  -0.0558 - 0.0152i  -0.4031 - 0.8773i   0.0677 - 0.0062i
   0.0972 - 0.0350i  -0.0745 + 0.0043i  -0.5592 - 0.7803i
  -0.4892 - 0.8328i   0.0568 - 0.0178i  -0.1017 - 0.0276i


s3p_new(:,:,111) =

  -0.0578 - 0.0161i  -0.4381 - 0.8575i   0.0693 - 0.0114i
   0.0964 - 0.0418i  -0.0770 + 0.0044i  -0.5891 - 0.7531i
  -0.5157 - 0.8107i   0.0572 - 0.0218i  -0.1042 - 0.0273i
```

```
s3p_new(:,:,112) =

  -0.0599 - 0.0171i  -0.4721 - 0.8364i   0.0706 - 0.0168i
   0.0952 - 0.0485i  -0.0795 + 0.0045i  -0.6178 - 0.7247i
  -0.5412 - 0.7878i   0.0572 - 0.0259i  -0.1068 - 0.0270i


s3p_new(:,:,113) =

  -0.0620 - 0.0181i  -0.5051 - 0.8139i   0.0714 - 0.0225i
   0.0934 - 0.0553i  -0.0820 + 0.0046i  -0.6451 - 0.6952i
  -0.5657 - 0.7642i   0.0570 - 0.0300i  -0.1093 - 0.0266i


s3p_new(:,:,114) =

  -0.0640 - 0.0192i  -0.5371 - 0.7901i   0.0717 - 0.0284i
   0.0912 - 0.0621i  -0.0845 + 0.0046i  -0.6710 - 0.6648i
  -0.5892 - 0.7397i   0.0565 - 0.0343i  -0.1118 - 0.0262i


s3p_new(:,:,115) =

  -0.0662 - 0.0194i  -0.5687 - 0.7687i   0.0714 - 0.0337i
   0.0877 - 0.0691i  -0.0869 + 0.0056i  -0.6988 - 0.6370i
  -0.6233 - 0.7138i   0.0552 - 0.0387i  -0.1139 - 0.0252i


s3p_new(:,:,116) =

  -0.0684 - 0.0193i  -0.5998 - 0.7471i   0.0706 - 0.0388i
   0.0833 - 0.0761i  -0.0892 + 0.0068i  -0.7267 - 0.6091i
  -0.6602 - 0.6859i   0.0535 - 0.0431i  -0.1159 - 0.0240i


s3p_new(:,:,117) =

  -0.0706 - 0.0191i  -0.6301 - 0.7243i   0.0695 - 0.0440i
   0.0783 - 0.0828i  -0.0916 + 0.0081i  -0.7534 - 0.5801i
  -0.6958 - 0.6560i   0.0515 - 0.0475i  -0.1179 - 0.0228i
```

```
s3p_new(:,:,118) =

  -0.0729 - 0.0189i  -0.6595 - 0.7003i   0.0680 - 0.0492i
   0.0728 - 0.0892i  -0.0939 + 0.0095i  -0.7790 - 0.5499i
  -0.7300 - 0.6243i   0.0491 - 0.0518i  -0.1199 - 0.0215i


s3p_new(:,:,119) =

  -0.0751 - 0.0186i  -0.6879 - 0.6751i   0.0662 - 0.0543i
   0.0668 - 0.0952i  -0.0962 + 0.0109i  -0.8034 - 0.5185i
  -0.7629 - 0.5908i   0.0463 - 0.0560i  -0.1219 - 0.0202i


s3p_new(:,:,120) =

  -0.0769 - 0.0173i  -0.7129 - 0.6468i   0.0635 - 0.0584i
   0.0610 - 0.0991i  -0.0978 + 0.0132i  -0.8226 - 0.4858i
  -0.7818 - 0.5639i   0.0434 - 0.0582i  -0.1229 - 0.0181i


s3p_new(:,:,121) =

  -0.0786 - 0.0158i  -0.7363 - 0.6173i   0.0605 - 0.0623i
   0.0550 - 0.1024i  -0.0993 + 0.0156i  -0.8398 - 0.4523i
  -0.7980 - 0.5377i   0.0405 - 0.0601i  -0.1238 - 0.0158i


s3p_new(:,:,122) =

  -0.0803 - 0.0143i  -0.7585 - 0.5870i   0.0573 - 0.0660i
   0.0489 - 0.1053i  -0.1007 + 0.0182i  -0.8557 - 0.4183i
  -0.8134 - 0.5110i   0.0374 - 0.0618i  -0.1247 - 0.0135i


s3p_new(:,:,123) =

  -0.0820 - 0.0127i  -0.7794 - 0.5559i   0.0539 - 0.0695i
   0.0425 - 0.1079i  -0.1021 + 0.0208i  -0.8703 - 0.3837i
  -0.8278 - 0.4838i   0.0343 - 0.0634i  -0.1255 - 0.0112i


s3p_new(:,:,124) =
```

```
  -0.0836 - 0.0109i  -0.7991 - 0.5241i   0.0502 - 0.0728i
   0.0361 - 0.1101i  -0.1034 + 0.0235i  -0.8834 - 0.3486i
  -0.8414 - 0.4562i   0.0311 - 0.0648i  -0.1262 - 0.0089i


s3p_new(:,:,125) =

  -0.0845 - 0.0084i  -0.8194 - 0.4905i   0.0456 - 0.0751i
   0.0277 - 0.1108i  -0.1033 + 0.0270i  -0.8966 - 0.3108i
  -0.8599 - 0.4163i   0.0268 - 0.0653i  -0.1260 - 0.0061i


s3p_new(:,:,126) =

  -0.0852 - 0.0058i  -0.8382 - 0.4561i   0.0409 - 0.0771i
   0.0194 - 0.1109i  -0.1030 + 0.0305i  -0.9082 - 0.2726i
  -0.8766 - 0.3757i   0.0225 - 0.0656i  -0.1258 - 0.0033i


s3p_new(:,:,127) =

  -0.0859 - 0.0031i  -0.8557 - 0.4209i   0.0362 - 0.0788i
   0.0113 - 0.1104i  -0.1026 + 0.0340i  -0.9182 - 0.2339i
  -0.8913 - 0.3344i   0.0182 - 0.0656i  -0.1255 - 0.0006i


s3p_new(:,:,128) =

  -0.0865 - 0.0004i  -0.8717 - 0.3851i   0.0314 - 0.0802i
   0.0033 - 0.1093i  -0.1022 + 0.0375i  -0.9265 - 0.1949i
  -0.9041 - 0.2926i   0.0141 - 0.0653i  -0.1252 + 0.0021i


s3p_new(:,:,129) =

  -0.0868 + 0.0024i  -0.8862 - 0.3486i   0.0265 - 0.0812i
  -0.0045 - 0.1076i  -0.1013 + 0.0410i  -0.9332 - 0.1553i
  -0.9151 - 0.2509i   0.0101 - 0.0647i  -0.1246 + 0.0048i


s3p_new(:,:,130) =

  -0.0857 + 0.0055i  -0.8995 - 0.3104i   0.0214 - 0.0807i
  -0.0121 - 0.1047i  -0.0988 + 0.0443i  -0.9382 - 0.1138i
```

```
  -0.9251 - 0.2131i   0.0065 - 0.0629i  -0.1229 + 0.0070i


s3p_new(:,:,131) =

  -0.0845 + 0.0085i  -0.9113 - 0.2717i   0.0165 - 0.0800i
  -0.0193 - 0.1012i  -0.0962 + 0.0474i  -0.9413 - 0.0721i
  -0.9336 - 0.1749i   0.0032 - 0.0610i  -0.1211 + 0.0092i


s3p_new(:,:,132) =

  -0.0831 + 0.0114i  -0.9213 - 0.2325i   0.0117 - 0.0789i
  -0.0260 - 0.0973i  -0.0935 + 0.0504i  -0.9426 - 0.0304i
  -0.9405 - 0.1364i   0.0000 - 0.0589i  -0.1193 + 0.0113i


s3p_new(:,:,133) =

  -0.0817 + 0.0143i  -0.9297 - 0.1930i   0.0071 - 0.0776i
  -0.0323 - 0.0929i  -0.0907 + 0.0533i  -0.9421 + 0.0113i
  -0.9459 - 0.0976i  -0.0029 - 0.0566i  -0.1175 + 0.0133i


s3p_new(:,:,134) =

  -0.0797 + 0.0172i  -0.9369 - 0.1532i   0.0026 - 0.0759i
  -0.0384 - 0.0877i  -0.0875 + 0.0563i  -0.9404 + 0.0531i
  -0.9500 - 0.0576i  -0.0055 - 0.0540i  -0.1152 + 0.0151i


s3p_new(:,:,135) =

  -0.0761 + 0.0203i  -0.9442 - 0.1129i  -0.0019 - 0.0737i
  -0.0444 - 0.0809i  -0.0835 + 0.0597i  -0.9389 + 0.0956i
  -0.9535 - 0.0141i  -0.0078 - 0.0505i  -0.1119 + 0.0162i


s3p_new(:,:,136) =

  -0.0725 + 0.0232i  -0.9497 - 0.0722i  -0.0061 - 0.0712i
  -0.0496 - 0.0738i  -0.0793 + 0.0629i  -0.9354 + 0.1380i
  -0.9551 + 0.0297i  -0.0097 - 0.0469i  -0.1086 + 0.0173i
```

```
s3p_new(:,:,137) =

  -0.0688 + 0.0257i  -0.9535 - 0.0313i  -0.0099 - 0.0685i
  -0.0539 - 0.0663i  -0.0750 + 0.0658i  -0.9301 + 0.1802i
  -0.9546 + 0.0735i  -0.0113 - 0.0432i  -0.1053 + 0.0183i


s3p_new(:,:,138) =

  -0.0650 + 0.0279i  -0.9555 + 0.0098i  -0.0135 - 0.0656i
  -0.0572 - 0.0587i  -0.0706 + 0.0685i  -0.9228 + 0.2223i
  -0.9521 + 0.1174i  -0.0125 - 0.0395i  -0.1020 + 0.0191i


s3p_new(:,:,139) =

  -0.0607 + 0.0300i  -0.9550 + 0.0512i  -0.0160 - 0.0623i
  -0.0600 - 0.0505i  -0.0660 + 0.0701i  -0.9132 + 0.2638i
  -0.9471 + 0.1597i  -0.0128 - 0.0356i  -0.0983 + 0.0192i


s3p_new(:,:,140) =

  -0.0556 + 0.0318i  -0.9514 + 0.0928i  -0.0168 - 0.0586i
  -0.0621 - 0.0413i  -0.0610 + 0.0701i  -0.9009 + 0.3044i
  -0.9392 + 0.1993i  -0.0120 - 0.0313i  -0.0939 + 0.0180i


s3p_new(:,:,141) =

  -0.0506 + 0.0332i  -0.9459 + 0.1342i  -0.0174 - 0.0549i
  -0.0630 - 0.0322i  -0.0561 + 0.0699i  -0.8868 + 0.3444i
  -0.9298 + 0.2386i  -0.0110 - 0.0271i  -0.0895 + 0.0169i


s3p_new(:,:,142) =

  -0.0456 + 0.0340i  -0.9387 + 0.1752i  -0.0179 - 0.0512i
  -0.0627 - 0.0236i  -0.0514 + 0.0694i  -0.8710 + 0.3838i
  -0.9187 + 0.2775i  -0.0098 - 0.0229i  -0.0851 + 0.0158i


s3p_new(:,:,143) =
```

```
  -0.0407 + 0.0344i  -0.9297 + 0.2159i  -0.0181 - 0.0476i
  -0.0612 - 0.0154i  -0.0467 + 0.0686i  -0.8533 + 0.4225i
  -0.9059 + 0.3158i  -0.0085 - 0.0188i  -0.0807 + 0.0147i


s3p_new(:,:,144) =

  -0.0346 + 0.0347i  -0.9177 + 0.2572i  -0.0197 - 0.0425i
  -0.0593 - 0.0058i  -0.0409 + 0.0689i  -0.8319 + 0.4612i
  -0.8898 + 0.3564i  -0.0041 - 0.0173i  -0.0765 + 0.0122i


s3p_new(:,:,145) =

  -0.0274 + 0.0343i  -0.9026 + 0.2987i  -0.0220 - 0.0361i
  -0.0562 + 0.0048i  -0.0337 + 0.0700i  -0.8067 + 0.4996i
  -0.8700 + 0.3991i   0.0028 - 0.0163i  -0.0725 + 0.0085i


s3p_new(:,:,146) =

  -0.0207 + 0.0329i  -0.8857 + 0.3394i  -0.0230 - 0.0297i
  -0.0512 + 0.0140i  -0.0267 + 0.0703i  -0.7796 + 0.5366i
  -0.8482 + 0.4406i   0.0082 - 0.0128i  -0.0682 + 0.0051i


s3p_new(:,:,147) =

  -0.0147 + 0.0304i  -0.8669 + 0.3792i  -0.0229 - 0.0236i
  -0.0449 + 0.0217i  -0.0199 + 0.0700i  -0.7509 + 0.5721i
  -0.8243 + 0.4810i   0.0114 - 0.0078i  -0.0639 + 0.0021i


s3p_new(:,:,148) =

  -0.0096 + 0.0270i  -0.8464 + 0.4180i  -0.0218 - 0.0180i
  -0.0376 + 0.0276i  -0.0133 + 0.0690i  -0.7206 + 0.6062i
  -0.7986 + 0.5201i   0.0122 - 0.0025i  -0.0594 - 0.0006i


s3p_new(:,:,149) =

  -0.0017 + 0.0258i  -0.8263 + 0.4557i  -0.0154 - 0.0208i
```

```
  -0.0310 + 0.0347i  -0.0043 + 0.0645i  -0.6904 + 0.6390i
  -0.7721 + 0.5582i   0.0160 - 0.0006i  -0.0568 - 0.0055i


s3p_new(:,:,150) =

   0.0073 + 0.0234i  -0.8059 + 0.4923i  -0.0044 - 0.0246i
  -0.0235 + 0.0422i   0.0045 + 0.0569i  -0.6598 + 0.6707i
  -0.7446 + 0.5953i   0.0225 - 0.0006i  -0.0548 - 0.0117i


s3p_new(:,:,151) =

   0.0144 + 0.0180i  -0.7839 + 0.5281i   0.0068 - 0.0230i
  -0.0141 + 0.0480i   0.0110 + 0.0483i  -0.6276 + 0.7008i
  -0.7153 + 0.6310i   0.0290 - 0.0003i  -0.0522 - 0.0175i


s3p_new(:,:,152) =

   0.0187 + 0.0109i  -0.7603 + 0.5629i   0.0156 - 0.0169i
  -0.0033 + 0.0517i   0.0151 + 0.0391i  -0.5941 + 0.7295i
  -0.6843 + 0.6653i   0.0355 + 0.0001i  -0.0490 - 0.0228i


s3p_new(:,:,153) =

   0.0200 + 0.0031i  -0.7352 + 0.5966i   0.0206 - 0.0078i
   0.0085 + 0.0529i   0.0169 + 0.0299i  -0.5593 + 0.7565i
  -0.6516 + 0.6980i   0.0419 + 0.0007i  -0.0452 - 0.0277i


s3p_new(:,:,154) =

   0.0245 - 0.0013i  -0.7045 + 0.6325i   0.0248 - 0.0069i
   0.0195 + 0.0557i   0.0196 + 0.0237i  -0.5195 + 0.7853i
  -0.6154 + 0.7287i   0.0491 - 0.0010i  -0.0450 - 0.0340i


s3p_new(:,:,155) =

   0.0301 - 0.0064i  -0.6705 + 0.6679i   0.0296 - 0.0092i
   0.0314 + 0.0577i   0.0219 + 0.0183i  -0.4768 + 0.8131i
  -0.5770 + 0.7573i   0.0563 - 0.0043i  -0.0453 - 0.0412i
```

```
s3p_new(:,:,156) =

   0.0346 - 0.0132i  -0.6348 + 0.7015i   0.0343 - 0.0118i
   0.0443 + 0.0572i   0.0230 + 0.0129i  -0.4326 + 0.8386i
  -0.5373 + 0.7839i   0.0633 - 0.0085i  -0.0446 - 0.0488i


s3p_new(:,:,157) =

   0.0375 - 0.0215i  -0.5974 + 0.7332i   0.0388 - 0.0147i
   0.0578 + 0.0539i   0.0229 + 0.0077i  -0.3870 + 0.8617i
  -0.4964 + 0.8085i   0.0699 - 0.0134i  -0.0429 - 0.0565i


s3p_new(:,:,158) =

   0.0386 - 0.0310i  -0.5584 + 0.7629i   0.0432 - 0.0179i
   0.0713 + 0.0476i   0.0217 + 0.0030i  -0.3402 + 0.8822i
  -0.4545 + 0.8309i   0.0762 - 0.0191i  -0.0401 - 0.0643i


s3p_new(:,:,159) =

   0.0412 - 0.0385i  -0.5170 + 0.7892i   0.0476 - 0.0217i
   0.0839 + 0.0423i   0.0234 - 0.0010i  -0.2899 + 0.8974i
  -0.4090 + 0.8514i   0.0822 - 0.0263i  -0.0421 - 0.0717i


s3p_new(:,:,160) =

   0.0434 - 0.0462i  -0.4743 + 0.8131i   0.0519 - 0.0258i
   0.0962 + 0.0352i   0.0248 - 0.0056i  -0.2386 + 0.9093i
  -0.3624 + 0.8696i   0.0875 - 0.0346i  -0.0447 - 0.0793i


s3p_new(:,:,161) =

   0.0447 - 0.0543i  -0.4306 + 0.8347i   0.0559 - 0.0303i
   0.1079 + 0.0258i   0.0252 - 0.0108i  -0.1869 + 0.9183i
  -0.3150 + 0.8851i   0.0921 - 0.0437i  -0.0470 - 0.0869i
```

```
s3p_new(:,:,162) =

   0.0452 - 0.0628i  -0.3858 + 0.8540i   0.0596 - 0.0350i
   0.1186 + 0.0142i   0.0245 - 0.0162i  -0.1349 + 0.9243i
  -0.2669 + 0.8981i   0.0958 - 0.0535i  -0.0490 - 0.0947i


s3p_new(:,:,163) =

   0.0447 - 0.0715i  -0.3402 + 0.8709i   0.0631 - 0.0401i
   0.1280 + 0.0004i   0.0226 - 0.0218i  -0.0829 + 0.9274i
  -0.2184 + 0.9085i   0.0986 - 0.0640i  -0.0508 - 0.1025i


s3p_new(:,:,164) =

   0.0452 - 0.0791i  -0.2916 + 0.8850i   0.0688 - 0.0491i
   0.1361 - 0.0130i   0.0275 - 0.0282i  -0.0305 + 0.9245i
  -0.1647 + 0.9169i   0.1000 - 0.0757i  -0.0546 - 0.1096i


s3p_new(:,:,165) =

   0.0450 - 0.0868i  -0.2424 + 0.8965i   0.0738 - 0.0588i
   0.1427 - 0.0280i   0.0320 - 0.0350i   0.0212 + 0.9185i
  -0.1108 + 0.9222i   0.1002 - 0.0878i  -0.0584 - 0.1167i


s3p_new(:,:,166) =

   0.0443 - 0.0947i  -0.1928 + 0.9052i   0.0778 - 0.0692i
   0.1476 - 0.0443i   0.0361 - 0.0421i   0.0722 + 0.9098i
  -0.0568 + 0.9243i   0.0991 - 0.1004i  -0.0622 - 0.1238i


s3p_new(:,:,167) =

   0.0430 - 0.1026i  -0.1429 + 0.9112i   0.0810 - 0.0801i
   0.1506 - 0.0619i   0.0398 - 0.0494i   0.1223 + 0.8982i
  -0.0030 + 0.9233i   0.0967 - 0.1132i  -0.0661 - 0.1309i


s3p_new(:,:,168) =
```

```
   0.0415 - 0.1106i  -0.0934 + 0.9146i   0.0828 - 0.0917i
   0.1516 - 0.0803i   0.0430 - 0.0573i   0.1709 + 0.8843i
   0.0503 + 0.9191i   0.0929 - 0.1262i  -0.0701 - 0.1380i


s3p_new(:,:,169) =

   0.0415 - 0.1187i  -0.0464 + 0.9161i   0.0807 - 0.1049i
   0.1513 - 0.0984i   0.0450 - 0.0677i   0.2166 + 0.8713i
   0.1014 + 0.9120i   0.0877 - 0.1394i  -0.0754 - 0.1455i


s3p_new(:,:,170) =

   0.0411 - 0.1269i   0.0006 + 0.9151i   0.0771 - 0.1183i
   0.1489 - 0.1171i   0.0459 - 0.0785i   0.2613 + 0.8559i
   0.1518 + 0.9020i   0.0811 - 0.1524i  -0.0809 - 0.1530i


s3p_new(:,:,171) =

   0.0404 - 0.1351i   0.0473 + 0.9118i   0.0719 - 0.1317i
   0.1443 - 0.1361i   0.0457 - 0.0896i   0.3050 + 0.8382i
   0.2015 + 0.8893i   0.0729 - 0.1652i  -0.0864 - 0.1603i


s3p_new(:,:,172) =

   0.0393 - 0.1433i   0.0937 + 0.9060i   0.0650 - 0.1450i
   0.1375 - 0.1551i   0.0443 - 0.1008i   0.3476 + 0.8183i
   0.2502 + 0.8739i   0.0634 - 0.1776i  -0.0921 - 0.1676i


s3p_new(:,:,173) =

   0.0380 - 0.1522i   0.1423 + 0.8962i   0.0557 - 0.1575i
   0.1280 - 0.1739i   0.0423 - 0.1126i   0.3911 + 0.7942i
   0.2983 + 0.8546i   0.0519 - 0.1891i  -0.0984 - 0.1752i


s3p_new(:,:,174) =

   0.0366 - 0.1629i   0.1977 + 0.8780i   0.0427 - 0.1675i
   0.1151 - 0.1917i   0.0407 - 0.1253i   0.4393 + 0.7613i
```

```
   0.3464 + 0.8293i   0.0377 - 0.1988i  -0.1067 - 0.1837i


s3p_new(:,:,175) =

   0.0347 - 0.1736i   0.2514 + 0.8564i   0.0283 - 0.1765i
   0.1000 - 0.2085i   0.0380 - 0.1381i   0.4849 + 0.7255i
   0.3925 + 0.8012i   0.0223 - 0.2075i  -0.1153 - 0.1920i


s3p_new(:,:,176) =

   0.0324 - 0.1844i   0.3032 + 0.8315i   0.0126 - 0.1842i
   0.0826 - 0.2242i   0.0342 - 0.1510i   0.5276 + 0.6870i
   0.4365 + 0.7707i   0.0057 - 0.2149i  -0.1241 - 0.2000i


s3p_new(:,:,177) =

   0.0295 - 0.1951i   0.3528 + 0.8035i  -0.0042 - 0.1905i
   0.0632 - 0.2383i   0.0293 - 0.1637i   0.5675 + 0.6461i
   0.4783 + 0.7378i  -0.0119 - 0.2209i  -0.1332 - 0.2079i


s3p_new(:,:,178) =

   0.0255 - 0.2069i   0.3997 + 0.7738i  -0.0217 - 0.1948i
   0.0407 - 0.2497i   0.0230 - 0.1773i   0.6039 + 0.6034i
   0.5182 + 0.6989i  -0.0313 - 0.2240i  -0.1437 - 0.2155i


s3p_new(:,:,179) =

   0.0197 - 0.2204i   0.4438 + 0.7432i  -0.0395 - 0.1968i
   0.0146 - 0.2570i   0.0149 - 0.1925i   0.6366 + 0.5594i
   0.5557 + 0.6517i  -0.0527 - 0.2225i  -0.1566 - 0.2227i


s3p_new(:,:,180) =

   0.0130 - 0.2337i   0.4857 + 0.7101i  -0.0578 - 0.1972i
  -0.0126 - 0.2615i   0.0052 - 0.2072i   0.6662 + 0.5138i
   0.5891 + 0.6028i  -0.0741 - 0.2190i  -0.1698 - 0.2295i
```

```
s3p_new(:,:,181) =

   0.0054 - 0.2468i   0.5252 + 0.6747i  -0.0764 - 0.1958i
  -0.0406 - 0.2632i  -0.0059 - 0.2214i   0.6925 + 0.4667i
   0.6185 + 0.5523i  -0.0952 - 0.2134i  -0.1835 - 0.2356i


s3p_new(:,:,182) =

  -0.0030 - 0.2597i   0.5622 + 0.6371i  -0.0952 - 0.1927i
  -0.0690 - 0.2617i  -0.0185 - 0.2350i   0.7156 + 0.4183i
   0.6438 + 0.5006i  -0.1160 - 0.2057i  -0.1976 - 0.2413i


s3p_new(:,:,183) =

  -0.0146 - 0.2718i   0.5994 + 0.5949i  -0.1152 - 0.1853i
  -0.0984 - 0.2569i  -0.0340 - 0.2474i   0.7373 + 0.3658i
   0.6731 + 0.4494i  -0.1377 - 0.1953i  -0.2134 - 0.2446i


s3p_new(:,:,184) =

  -0.0295 - 0.2827i   0.6361 + 0.5478i  -0.1355 - 0.1733i
  -0.1284 - 0.2484i  -0.0525 - 0.2583i   0.7567 + 0.3089i
   0.7064 + 0.3972i  -0.1600 - 0.1815i  -0.2309 - 0.2454i


s3p_new(:,:,185) =

  -0.0456 - 0.2928i   0.6691 + 0.4985i  -0.1546 - 0.1591i
  -0.1577 - 0.2363i  -0.0725 - 0.2678i   0.7719 + 0.2512i
   0.7357 + 0.3425i  -0.1810 - 0.1652i  -0.2487 - 0.2452i


s3p_new(:,:,186) =

  -0.0627 - 0.3020i   0.6983 + 0.4473i  -0.1722 - 0.1428i
  -0.1859 - 0.2207i  -0.0939 - 0.2757i   0.7827 + 0.1929i
   0.7607 + 0.2856i  -0.2003 - 0.1465i  -0.2668 - 0.2440i


s3p_new(:,:,187) =
```

```
  -0.0809 - 0.3101i   0.7237 + 0.3944i  -0.1880 - 0.1245i
  -0.2125 - 0.2017i  -0.1166 - 0.2819i   0.7891 + 0.1344i
   0.7813 + 0.2269i  -0.2176 - 0.1255i  -0.2852 - 0.2417i


s3p_new(:,:,188) =

  -0.0985 - 0.3108i   0.7474 + 0.3411i  -0.1996 - 0.0990i
  -0.2359 - 0.1703i  -0.1398 - 0.2778i   0.7947 + 0.0751i
   0.7926 + 0.1666i  -0.2298 - 0.0946i  -0.2988 - 0.2333i


s3p_new(:,:,189) =

  -0.1152 - 0.3067i   0.7686 + 0.2868i  -0.2061 - 0.0696i
  -0.2531 - 0.1309i  -0.1621 - 0.2667i   0.7979 + 0.0150i
   0.7963 + 0.1060i  -0.2352 - 0.0583i  -0.3091 - 0.2212i


s3p_new(:,:,190) =

  -0.1317 - 0.3017i   0.7859 + 0.2313i  -0.2084 - 0.0402i
  -0.2640 - 0.0903i  -0.1836 - 0.2537i   0.7966 - 0.0451i
   0.7955 + 0.0459i  -0.2350 - 0.0225i  -0.3189 - 0.2087i


s3p_new(:,:,191) =

  -0.1480 - 0.2958i   0.7991 + 0.1747i  -0.2066 - 0.0115i
  -0.2686 - 0.0493i  -0.2041 - 0.2391i   0.7908 - 0.1050i
   0.7901 - 0.0136i  -0.2296 + 0.0119i  -0.3282 - 0.1958i


s3p_new(:,:,192) =

  -0.1641 - 0.2891i   0.8083 + 0.1174i  -0.2010 + 0.0159i
  -0.2669 - 0.0090i  -0.2234 - 0.2228i   0.7805 - 0.1642i
   0.7803 - 0.0720i  -0.2193 + 0.0443i  -0.3369 - 0.1824i


s3p_new(:,:,193) =

  -0.1406 - 0.3114i   0.7965 + 0.0013i  -0.1734 - 0.0669i
```

```
  -0.2324 + 0.1028i  -0.1976 - 0.2538i   0.7256 - 0.2725i
   0.7327 - 0.1932i  -0.1684 + 0.1413i  -0.3307 - 0.2152i


s3p_new(:,:,194) =

  -0.0989 - 0.3394i   0.7595 - 0.1263i  -0.0886 - 0.1412i
  -0.1356 + 0.1967i  -0.1486 - 0.2941i   0.6405 - 0.3809i
   0.6519 - 0.3181i  -0.0550 + 0.2095i  -0.3135 - 0.2632i


s3p_new(:,:,195) =

  -0.0521 - 0.3616i   0.7029 - 0.2417i   0.0120 - 0.1470i
  -0.0118 + 0.2233i  -0.0914 - 0.3248i   0.5407 - 0.4683i
   0.5516 - 0.4195i   0.0750 + 0.1997i  -0.2894 - 0.3101i


s3p_new(:,:,196) =

  -0.0010 - 0.3771i   0.6297 - 0.3421i   0.0849 - 0.0962i
   0.0995 + 0.1831i  -0.0278 - 0.3441i   0.4308 - 0.5330i
   0.4378 - 0.4947i   0.1740 + 0.1178i  -0.2584 - 0.3548i


s3p_new(:,:,197) =

   0.0534 - 0.3852i   0.5434 - 0.4253i   0.1068 - 0.0223i
   0.1674 + 0.0963i   0.0399 - 0.3509i   0.3157 - 0.5744i
   0.3167 - 0.5426i   0.2068 - 0.0043i  -0.2207 - 0.3964i


s3p_new(:,:,198) =

  -0.0017 - 0.4508i   0.4888 - 0.3831i   0.1067 - 0.0724i
   0.2049 - 0.0174i  -0.0243 - 0.4191i   0.2857 - 0.5145i
   0.3127 - 0.4565i   0.1846 - 0.1142i  -0.3023 - 0.4136i


s3p_new(:,:,199) =

  -0.0959 - 0.5110i   0.4373 - 0.3268i   0.0653 - 0.1397i
   0.1699 - 0.1431i  -0.1374 - 0.4755i   0.2643 - 0.4436i
   0.3082 - 0.3577i   0.1090 - 0.2016i  -0.4096 - 0.4064i
```

```
s3p_new(:,:,200) =

  -0.2114 - 0.5497i   0.3835 - 0.2731i  -0.0179 - 0.1787i
   0.0604 - 0.2309i  -0.2755 - 0.4990i   0.2389 - 0.3745i
   0.2864 - 0.2663i  -0.0062 - 0.2412i  -0.5210 - 0.3747i


s3p_new(:,:,201) =

  -0.3433 - 0.5613i   0.3275 - 0.2220i  -0.1219 - 0.1647i
  -0.0897 - 0.2389i  -0.4291 - 0.4817i   0.2095 - 0.3074i
   0.2490 - 0.1847i  -0.1318 - 0.2164i  -0.6314 - 0.3170i
```

3-Port to 2-Port S-Parameters

Convert 3-port S-parameters to 2-port S-parameters by terminating port 3 with an impedance of `Z0`.

```
ckt = read(rfckt.passive,'default.s3p');
s3p = ckt.NetworkData.Data;
Z0 = ckt.NetworkData.Z0;
s2p = snp2smp(s3p,Z0)
```

```
s2p = 
s2p(:,:,1) =

  -0.0073 - 0.8086i  -0.0318 + 0.4208i
   0.0869 + 0.3238i   0.1431 - 0.7986i


s2p(:,:,2) =

  -0.0330 - 0.8060i  -0.0079 + 0.4073i
   0.0946 + 0.3169i   0.1175 - 0.8008i


s2p(:,:,3) =

  -0.0587 - 0.8021i  -0.0050 + 0.4223i
```

```
   0.1081 + 0.3133i   0.0913 - 0.8017i


s2p(:,:,4) =

  -0.0842 - 0.7972i  -0.0049 + 0.4415i
   0.1224 + 0.3095i   0.0650 - 0.8016i


s2p(:,:,5) =

  -0.1094 - 0.7916i  -0.0047 + 0.4607i
   0.1366 + 0.3051i   0.0389 - 0.8007i


s2p(:,:,6) =

  -0.1344 - 0.7851i  -0.0045 + 0.4799i
   0.1507 + 0.3000i   0.0129 - 0.7989i


s2p(:,:,7) =

  -0.1591 - 0.7779i  -0.0043 + 0.4990i
   0.1646 + 0.2943i  -0.0130 - 0.7962i


s2p(:,:,8) =

  -0.1839 - 0.7695i   0.0207 + 0.4894i
   0.1751 + 0.2959i  -0.0392 - 0.7925i


s2p(:,:,9) =

  -0.2083 - 0.7604i   0.0447 + 0.4786i
   0.1859 + 0.2972i  -0.0652 - 0.7878i


s2p(:,:,10) =

  -0.2323 - 0.7505i   0.0677 + 0.4667i
   0.1968 + 0.2982i  -0.0909 - 0.7824i
```

```
s2p(:,:,11) =

  -0.2560 - 0.7399i   0.0895 + 0.4536i
   0.2079 + 0.2988i  -0.1163 - 0.7761i


s2p(:,:,12) =

  -0.2791 - 0.7284i   0.1084 + 0.4444i
   0.2184 + 0.2979i  -0.1415 - 0.7689i


s2p(:,:,13) =

  -0.3019 - 0.7156i   0.1135 + 0.4681i
   0.2239 + 0.2889i  -0.1670 - 0.7602i


s2p(:,:,14) =

  -0.3241 - 0.7021i   0.1186 + 0.4919i
   0.2290 + 0.2799i  -0.1920 - 0.7507i


s2p(:,:,15) =

  -0.3458 - 0.6879i   0.1236 + 0.5158i
   0.2338 + 0.2708i  -0.2166 - 0.7405i


s2p(:,:,16) =

  -0.3669 - 0.6731i   0.1285 + 0.5396i
   0.2384 + 0.2616i  -0.2407 - 0.7294i


s2p(:,:,17) =

  -0.3875 - 0.6574i   0.1339 + 0.5517i
   0.2440 + 0.2520i  -0.2644 - 0.7173i


s2p(:,:,18) =
```

```
  -0.4079 - 0.6403i   0.1405 + 0.5285i
   0.2533 + 0.2412i  -0.2880 - 0.7039i


s2p(:,:,19) =

  -0.4276 - 0.6226i   0.1462 + 0.5053i
   0.2623 + 0.2300i  -0.3110 - 0.6898i


s2p(:,:,20) =

  -0.4465 - 0.6043i   0.1509 + 0.4821i
   0.2707 + 0.2185i  -0.3333 - 0.6749i


s2p(:,:,21) =

  -0.4647 - 0.5854i   0.1547 + 0.4590i
   0.2786 + 0.2066i  -0.3550 - 0.6593i


s2p(:,:,22) =

  -0.4819 - 0.5655i   0.1688 + 0.4531i
   0.2863 + 0.1949i  -0.3760 - 0.6425i


s2p(:,:,23) =

  -0.4981 - 0.5442i   0.2030 + 0.4744i
   0.2941 + 0.1836i  -0.3962 - 0.6241i


s2p(:,:,24) =

  -0.5134 - 0.5225i   0.2397 + 0.4935i
   0.3015 + 0.1720i  -0.4155 - 0.6051i


s2p(:,:,25) =

  -0.5278 - 0.5005i   0.2786 + 0.5100i
```

```
   0.3084 + 0.1601i  -0.4340 - 0.5856i


s2p(:,:,26) =

  -0.5413 - 0.4780i   0.3196 + 0.5239i
   0.3148 + 0.1479i  -0.4516 - 0.5655i


s2p(:,:,27) =

  -0.5539 - 0.4547i   0.3417 + 0.5241i
   0.3216 + 0.1352i  -0.4686 - 0.5442i


s2p(:,:,28) =

  -0.5655 - 0.4306i   0.3434 + 0.5127i
   0.3289 + 0.1220i  -0.4849 - 0.5217i


s2p(:,:,29) =

  -0.5762 - 0.4063i   0.3448 + 0.5014i
   0.3356 + 0.1083i  -0.5001 - 0.4988i


s2p(:,:,30) =

  -0.5858 - 0.3817i   0.3461 + 0.4901i
   0.3417 + 0.0943i  -0.5143 - 0.4755i


s2p(:,:,31) =

  -0.5944 - 0.3570i   0.3470 + 0.4789i
   0.3472 + 0.0800i  -0.5275 - 0.4518i


s2p(:,:,32) =

  -0.6015 - 0.3314i   0.3675 + 0.4728i
   0.3492 + 0.0682i  -0.5393 - 0.4270i
```

```
s2p(:,:,33) =

  -0.6071 - 0.3052i   0.4000 + 0.4682i
   0.3490 + 0.0582i  -0.5499 - 0.4015i


s2p(:,:,34) =

  -0.6116 - 0.2791i   0.4328 + 0.4616i
   0.3485 + 0.0482i  -0.5593 - 0.3758i


s2p(:,:,35) =

  -0.6150 - 0.2531i   0.4659 + 0.4530i
   0.3477 + 0.0383i  -0.5676 - 0.3500i


s2p(:,:,36) =

  -0.6173 - 0.2273i   0.4991 + 0.4422i
   0.3467 + 0.0284i  -0.5747 - 0.3242i


s2p(:,:,37) =

  -0.6176 - 0.2006i   0.5115 + 0.4381i
   0.3446 + 0.0139i  -0.5797 - 0.2973i


s2p(:,:,38) =

  -0.6164 - 0.1739i   0.5169 + 0.4366i
   0.3416 - 0.0019i  -0.5831 - 0.2702i


s2p(:,:,39) =

  -0.6142 - 0.1477i   0.5224 + 0.4351i
   0.3379 - 0.0174i  -0.5854 - 0.2434i


s2p(:,:,40) =
```

```
  -0.6108 - 0.1218i   0.5278 + 0.4334i
   0.3334 - 0.0326i  -0.5864 - 0.2168i


s2p(:,:,41) =

  -0.6063 - 0.0965i   0.5333 + 0.4318i
   0.3283 - 0.0474i  -0.5863 - 0.1905i


s2p(:,:,42) =

  -0.5998 - 0.0712i   0.5559 + 0.4056i
   0.3236 - 0.0642i  -0.5840 - 0.1634i


s2p(:,:,43) =

  -0.5921 - 0.0465i   0.5794 + 0.3746i
   0.3181 - 0.0809i  -0.5803 - 0.1367i


s2p(:,:,44) =

  -0.5834 - 0.0225i   0.6013 + 0.3423i
   0.3117 - 0.0973i  -0.5755 - 0.1106i


s2p(:,:,45) =

  -0.5737 + 0.0007i   0.6213 + 0.3086i
   0.3045 - 0.1132i  -0.5695 - 0.0851i


s2p(:,:,46) =

  -0.5632 + 0.0231i   0.6394 + 0.2738i
   0.2965 - 0.1286i  -0.5624 - 0.0603i


s2p(:,:,47) =

  -0.5499 + 0.0451i   0.6675 + 0.2650i
```

```
   0.2880 - 0.1423i  -0.5530 - 0.0353i


s2p(:,:,48) =

  -0.5358 + 0.0659i   0.6956 + 0.2549i
   0.2789 - 0.1556i  -0.5426 - 0.0112i


s2p(:,:,49) =

  -0.5209 + 0.0857i   0.7236 + 0.2435i
   0.2691 - 0.1682i  -0.5311 + 0.0120i


s2p(:,:,50) =

  -0.5054 + 0.1045i   0.7515 + 0.2307i
   0.2588 - 0.1803i  -0.5186 + 0.0342i


s2p(:,:,51) =

  -0.4890 + 0.1220i   0.7768 + 0.2151i
   0.2480 - 0.1915i  -0.5051 + 0.0552i


s2p(:,:,52) =

  -0.4710 + 0.1379i   0.7850 + 0.1872i
   0.2365 - 0.1996i  -0.4898 + 0.0747i


s2p(:,:,53) =

  -0.4525 + 0.1526i   0.7922 + 0.1591i
   0.2248 - 0.2071i  -0.4737 + 0.0929i


s2p(:,:,54) =

  -0.4336 + 0.1660i   0.7984 + 0.1306i
   0.2128 - 0.2140i  -0.4571 + 0.1100i
```

```
s2p(:,:,55) =

  -0.4144 + 0.1781i   0.8036 + 0.1020i
   0.2008 - 0.2202i  -0.4398 + 0.1258i


s2p(:,:,56) =

  -0.3947 + 0.1888i   0.8101 + 0.0737i
   0.1872 - 0.2257i  -0.4218 + 0.1400i


s2p(:,:,57) =

  -0.3742 + 0.1973i   0.8228 + 0.0465i
   0.1698 - 0.2299i  -0.4029 + 0.1522i


s2p(:,:,58) =

  -0.3537 + 0.2045i   0.8345 + 0.0186i
   0.1525 - 0.2329i  -0.3836 + 0.1630i


s2p(:,:,59) =

  -0.3332 + 0.2104i   0.8453 - 0.0101i
   0.1353 - 0.2347i  -0.3642 + 0.1725i


s2p(:,:,60) =

  -0.3128 + 0.2150i   0.8551 - 0.0396i
   0.1185 - 0.2353i  -0.3447 + 0.1808i


s2p(:,:,61) =

  -0.2928 + 0.2181i   0.8602 - 0.0730i
   0.1016 - 0.2355i  -0.3251 + 0.1873i


s2p(:,:,62) =
```

```
  -0.2733 + 0.2196i   0.8576 - 0.1124i
   0.0841 - 0.2354i  -0.3055 + 0.1919i


s2p(:,:,63) =

  -0.2540 + 0.2199i   0.8532 - 0.1516i
   0.0670 - 0.2341i  -0.2860 + 0.1953i


s2p(:,:,64) =

  -0.2352 + 0.2191i   0.8470 - 0.1907i
   0.0505 - 0.2316i  -0.2668 + 0.1975i


s2p(:,:,65) =

  -0.2167 + 0.2172i   0.8389 - 0.2295i
   0.0346 - 0.2279i  -0.2478 + 0.1986i


s2p(:,:,66) =

  -0.1995 + 0.2138i   0.8346 - 0.2689i
   0.0201 - 0.2232i  -0.2296 + 0.1984i


s2p(:,:,67) =

  -0.1834 + 0.2091i   0.8339 - 0.3095i
   0.0071 - 0.2176i  -0.2122 + 0.1972i


s2p(:,:,68) =

  -0.1678 + 0.2035i   0.8312 - 0.3505i
  -0.0052 - 0.2113i  -0.1952 + 0.1950i


s2p(:,:,69) =

  -0.1527 + 0.1973i   0.8265 - 0.3919i
```

```
  -0.0167 - 0.2042i  -0.1786 + 0.1918i


s2p(:,:,70) =

  -0.1382 + 0.1903i   0.8197 - 0.4335i
  -0.0275 - 0.1966i  -0.1626 + 0.1877i


s2p(:,:,71) =

  -0.1259 + 0.1824i   0.8090 - 0.4685i
  -0.0377 - 0.1872i  -0.1486 + 0.1828i


s2p(:,:,72) =

  -0.1150 + 0.1741i   0.7958 - 0.4993i
  -0.0471 - 0.1766i  -0.1360 + 0.1773i


s2p(:,:,73) =

  -0.1045 + 0.1653i   0.7814 - 0.5297i
  -0.0553 - 0.1655i  -0.1238 + 0.1712i


s2p(:,:,74) =

  -0.0945 + 0.1563i   0.7659 - 0.5597i
  -0.0624 - 0.1541i  -0.1120 + 0.1646i


s2p(:,:,75) =

  -0.0848 + 0.1469i   0.7492 - 0.5893i
  -0.0682 - 0.1426i  -0.1008 + 0.1574i


s2p(:,:,76) =

  -0.0784 + 0.1373i   0.7234 - 0.6202i
  -0.0737 - 0.1307i  -0.0922 + 0.1499i
```

```
s2p(:,:,77) =

  -0.0729 + 0.1278i   0.6935 - 0.6504i
  -0.0782 - 0.1187i  -0.0847 + 0.1423i


s2p(:,:,78) =

  -0.0675 + 0.1183i   0.6624 - 0.6791i
  -0.0814 - 0.1068i  -0.0774 + 0.1344i


s2p(:,:,79) =

  -0.0620 + 0.1088i   0.6302 - 0.7064i
  -0.0833 - 0.0950i  -0.0704 + 0.1264i


s2p(:,:,80) =

  -0.0566 + 0.0992i   0.5968 - 0.7322i
  -0.0840 - 0.0835i  -0.0635 + 0.1182i


s2p(:,:,81) =

  -0.0553 + 0.0901i   0.5683 - 0.7589i
  -0.0847 - 0.0717i  -0.0594 + 0.1104i


s2p(:,:,82) =

  -0.0541 + 0.0812i   0.5395 - 0.7850i
  -0.0842 - 0.0603i  -0.0557 + 0.1027i


s2p(:,:,83) =

  -0.0523 + 0.0726i   0.5095 - 0.8099i
  -0.0825 - 0.0495i  -0.0519 + 0.0951i


s2p(:,:,84) =
```

```
  -0.0500 + 0.0642i   0.4783 - 0.8337i
  -0.0796 - 0.0394i  -0.0480 + 0.0875i


s2p(:,:,85) =

  -0.0471 + 0.0560i   0.4460 - 0.8563i
  -0.0756 - 0.0301i  -0.0441 + 0.0799i


s2p(:,:,86) =

  -0.0493 + 0.0489i   0.4108 - 0.8745i
  -0.0724 - 0.0201i  -0.0429 + 0.0734i


s2p(:,:,87) =

  -0.0506 + 0.0419i   0.3748 - 0.8913i
  -0.0680 - 0.0112i  -0.0415 + 0.0671i


s2p(:,:,88) =

  -0.0510 + 0.0351i   0.3382 - 0.9066i
  -0.0626 - 0.0034i  -0.0397 + 0.0610i


s2p(:,:,89) =

  -0.0507 + 0.0286i   0.3009 - 0.9205i
  -0.0564 + 0.0030i  -0.0377 + 0.0549i


s2p(:,:,90) =

  -0.0502 + 0.0226i   0.2632 - 0.9325i
  -0.0498 + 0.0086i  -0.0359 + 0.0491i


s2p(:,:,91) =

  -0.0529 + 0.0183i   0.2262 - 0.9414i
```

```
  -0.0437 + 0.0162i  -0.0368 + 0.0443i


s2p(:,:,92) =

  -0.0553 + 0.0137i   0.1889 - 0.9488i
  -0.0366 + 0.0219i  -0.0373 + 0.0396i


s2p(:,:,93) =

  -0.0573 + 0.0088i   0.1513 - 0.9547i
  -0.0291 + 0.0256i  -0.0374 + 0.0350i


s2p(:,:,94) =

  -0.0588 + 0.0037i   0.1136 - 0.9591i
  -0.0214 + 0.0274i  -0.0370 + 0.0305i


s2p(:,:,95) =

  -0.0604 - 0.0010i   0.0750 - 0.9625i
  -0.0145 + 0.0286i  -0.0369 + 0.0264i


s2p(:,:,96) =

  -0.0631 - 0.0037i   0.0340 - 0.9654i
  -0.0081 + 0.0318i  -0.0384 + 0.0226i


s2p(:,:,97) =

  -0.0658 - 0.0066i  -0.0071 - 0.9666i
  -0.0009 + 0.0336i  -0.0395 + 0.0187i


s2p(:,:,98) =

  -0.0683 - 0.0098i  -0.0483 - 0.9660i
   0.0067 + 0.0337i  -0.0403 + 0.0149i
```

```
s2p(:,:,99) =

  -0.0707 - 0.0132i  -0.0894 - 0.9636i
   0.0142 + 0.0321i  -0.0407 + 0.0110i


s2p(:,:,100) =

  -0.0732 - 0.0160i  -0.1284 - 0.9592i
   0.0209 + 0.0303i  -0.0413 + 0.0077i


s2p(:,:,101) =

  -0.0763 - 0.0176i  -0.1637 - 0.9528i
   0.0269 + 0.0296i  -0.0427 + 0.0051i


s2p(:,:,102) =

  -0.0793 - 0.0193i  -0.1986 - 0.9450i
   0.0330 + 0.0278i  -0.0440 + 0.0024i


s2p(:,:,103) =

  -0.0823 - 0.0211i  -0.2333 - 0.9360i
   0.0391 + 0.0249i  -0.0451 - 0.0006i


s2p(:,:,104) =

  -0.0853 - 0.0230i  -0.2675 - 0.9258i
   0.0449 + 0.0209i  -0.0460 - 0.0036i


s2p(:,:,105) =

  -0.0882 - 0.0243i  -0.3043 - 0.9144i
   0.0500 + 0.0169i  -0.0473 - 0.0060i


s2p(:,:,106) =
```

```
  -0.0909 - 0.0251i  -0.3437 - 0.9017i
   0.0543 + 0.0133i  -0.0490 - 0.0078i


s2p(:,:,107) =

  -0.0936 - 0.0259i  -0.3825 - 0.8872i
   0.0583 + 0.0092i  -0.0507 - 0.0097i


s2p(:,:,108) =

  -0.0964 - 0.0267i  -0.4208 - 0.8710i
   0.0621 + 0.0044i  -0.0523 - 0.0117i


s2p(:,:,109) =

  -0.0991 - 0.0275i  -0.4583 - 0.8533i
   0.0654 - 0.0009i  -0.0539 - 0.0138i


s2p(:,:,110) =

  -0.1017 - 0.0276i  -0.4892 - 0.8328i
   0.0677 - 0.0062i  -0.0558 - 0.0152i


s2p(:,:,111) =

  -0.1042 - 0.0273i  -0.5157 - 0.8107i
   0.0693 - 0.0114i  -0.0578 - 0.0161i


s2p(:,:,112) =

  -0.1068 - 0.0270i  -0.5412 - 0.7878i
   0.0706 - 0.0168i  -0.0599 - 0.0171i


s2p(:,:,113) =

  -0.1093 - 0.0266i  -0.5657 - 0.7642i
```

```
   0.0714 - 0.0225i  -0.0620 - 0.0181i


s2p(:,:,114) =

  -0.1118 - 0.0262i  -0.5892 - 0.7397i
   0.0717 - 0.0284i  -0.0640 - 0.0192i


s2p(:,:,115) =

  -0.1139 - 0.0252i  -0.6233 - 0.7138i
   0.0714 - 0.0337i  -0.0662 - 0.0194i


s2p(:,:,116) =

  -0.1159 - 0.0240i  -0.6602 - 0.6859i
   0.0706 - 0.0388i  -0.0684 - 0.0193i


s2p(:,:,117) =

  -0.1179 - 0.0228i  -0.6958 - 0.6560i
   0.0695 - 0.0440i  -0.0706 - 0.0191i


s2p(:,:,118) =

  -0.1199 - 0.0215i  -0.7300 - 0.6243i
   0.0680 - 0.0492i  -0.0729 - 0.0189i


s2p(:,:,119) =

  -0.1219 - 0.0202i  -0.7629 - 0.5908i
   0.0662 - 0.0543i  -0.0751 - 0.0186i


s2p(:,:,120) =

  -0.1229 - 0.0181i  -0.7818 - 0.5639i
   0.0635 - 0.0584i  -0.0769 - 0.0173i
```

```
s2p(:,:,121) =

  -0.1238 - 0.0158i  -0.7980 - 0.5377i
   0.0605 - 0.0623i  -0.0786 - 0.0158i


s2p(:,:,122) =

  -0.1247 - 0.0135i  -0.8134 - 0.5110i
   0.0573 - 0.0660i  -0.0803 - 0.0143i


s2p(:,:,123) =

  -0.1255 - 0.0112i  -0.8278 - 0.4838i
   0.0539 - 0.0695i  -0.0820 - 0.0127i


s2p(:,:,124) =

  -0.1262 - 0.0089i  -0.8414 - 0.4562i
   0.0502 - 0.0728i  -0.0836 - 0.0109i


s2p(:,:,125) =

  -0.1260 - 0.0061i  -0.8599 - 0.4163i
   0.0456 - 0.0751i  -0.0845 - 0.0084i


s2p(:,:,126) =

  -0.1258 - 0.0033i  -0.8766 - 0.3757i
   0.0409 - 0.0771i  -0.0852 - 0.0058i


s2p(:,:,127) =

  -0.1255 - 0.0006i  -0.8913 - 0.3344i
   0.0362 - 0.0788i  -0.0859 - 0.0031i


s2p(:,:,128) =
```

```
  -0.1252 + 0.0021i  -0.9041 - 0.2926i
   0.0314 - 0.0802i  -0.0865 - 0.0004i


s2p(:,:,129) =

  -0.1246 + 0.0048i  -0.9151 - 0.2509i
   0.0265 - 0.0812i  -0.0868 + 0.0024i


s2p(:,:,130) =

  -0.1229 + 0.0070i  -0.9251 - 0.2131i
   0.0214 - 0.0807i  -0.0857 + 0.0055i


s2p(:,:,131) =

  -0.1211 + 0.0092i  -0.9336 - 0.1749i
   0.0165 - 0.0800i  -0.0845 + 0.0085i


s2p(:,:,132) =

  -0.1193 + 0.0113i  -0.9405 - 0.1364i
   0.0117 - 0.0789i  -0.0831 + 0.0114i


s2p(:,:,133) =

  -0.1175 + 0.0133i  -0.9459 - 0.0976i
   0.0071 - 0.0776i  -0.0817 + 0.0143i


s2p(:,:,134) =

  -0.1152 + 0.0151i  -0.9500 - 0.0576i
   0.0026 - 0.0759i  -0.0797 + 0.0172i


s2p(:,:,135) =

  -0.1119 + 0.0162i  -0.9535 - 0.0141i
```

```
  -0.0019 - 0.0737i  -0.0761 + 0.0203i


s2p(:,:,136) =

  -0.1086 + 0.0173i  -0.9551 + 0.0297i
  -0.0061 - 0.0712i  -0.0725 + 0.0232i


s2p(:,:,137) =

  -0.1053 + 0.0183i  -0.9546 + 0.0735i
  -0.0099 - 0.0685i  -0.0688 + 0.0257i


s2p(:,:,138) =

  -0.1020 + 0.0191i  -0.9521 + 0.1174i
  -0.0135 - 0.0656i  -0.0650 + 0.0279i


s2p(:,:,139) =

  -0.0983 + 0.0192i  -0.9471 + 0.1597i
  -0.0160 - 0.0623i  -0.0607 + 0.0300i


s2p(:,:,140) =

  -0.0939 + 0.0180i  -0.9392 + 0.1993i
  -0.0168 - 0.0586i  -0.0556 + 0.0318i


s2p(:,:,141) =

  -0.0895 + 0.0169i  -0.9298 + 0.2386i
  -0.0174 - 0.0549i  -0.0506 + 0.0332i


s2p(:,:,142) =

  -0.0851 + 0.0158i  -0.9187 + 0.2775i
  -0.0179 - 0.0512i  -0.0456 + 0.0340i
```

```
s2p(:,:,143) =

  -0.0807 + 0.0147i  -0.9059 + 0.3158i
  -0.0181 - 0.0476i  -0.0407 + 0.0344i


s2p(:,:,144) =

  -0.0765 + 0.0122i  -0.8898 + 0.3564i
  -0.0197 - 0.0425i  -0.0346 + 0.0347i


s2p(:,:,145) =

  -0.0725 + 0.0085i  -0.8700 + 0.3991i
  -0.0220 - 0.0361i  -0.0274 + 0.0343i


s2p(:,:,146) =

  -0.0682 + 0.0051i  -0.8482 + 0.4406i
  -0.0230 - 0.0297i  -0.0207 + 0.0329i


s2p(:,:,147) =

  -0.0639 + 0.0021i  -0.8243 + 0.4810i
  -0.0229 - 0.0236i  -0.0147 + 0.0304i


s2p(:,:,148) =

  -0.0594 - 0.0006i  -0.7986 + 0.5201i
  -0.0218 - 0.0180i  -0.0096 + 0.0270i


s2p(:,:,149) =

  -0.0568 - 0.0055i  -0.7721 + 0.5582i
  -0.0154 - 0.0208i  -0.0017 + 0.0258i


s2p(:,:,150) =
```

```
  -0.0548 - 0.0117i  -0.7446 + 0.5953i
  -0.0044 - 0.0246i   0.0073 + 0.0234i


s2p(:,:,151) =

  -0.0522 - 0.0175i  -0.7153 + 0.6310i
   0.0068 - 0.0230i   0.0144 + 0.0180i


s2p(:,:,152) =

  -0.0490 - 0.0228i  -0.6843 + 0.6653i
   0.0156 - 0.0169i   0.0187 + 0.0109i


s2p(:,:,153) =

  -0.0452 - 0.0277i  -0.6516 + 0.6980i
   0.0206 - 0.0078i   0.0200 + 0.0031i


s2p(:,:,154) =

  -0.0450 - 0.0340i  -0.6154 + 0.7287i
   0.0248 - 0.0069i   0.0245 - 0.0013i


s2p(:,:,155) =

  -0.0453 - 0.0412i  -0.5770 + 0.7573i
   0.0296 - 0.0092i   0.0301 - 0.0064i


s2p(:,:,156) =

  -0.0446 - 0.0488i  -0.5373 + 0.7839i
   0.0343 - 0.0118i   0.0346 - 0.0132i


s2p(:,:,157) =

  -0.0429 - 0.0565i  -0.4964 + 0.8085i
```

```
   0.0388 - 0.0147i   0.0375 - 0.0215i


s2p(:,:,158) =

  -0.0401 - 0.0643i  -0.4545 + 0.8309i
   0.0432 - 0.0179i   0.0386 - 0.0310i


s2p(:,:,159) =

  -0.0421 - 0.0717i  -0.4090 + 0.8514i
   0.0476 - 0.0217i   0.0412 - 0.0385i


s2p(:,:,160) =

  -0.0447 - 0.0793i  -0.3624 + 0.8696i
   0.0519 - 0.0258i   0.0434 - 0.0462i


s2p(:,:,161) =

  -0.0470 - 0.0869i  -0.3150 + 0.8851i
   0.0559 - 0.0303i   0.0447 - 0.0543i


s2p(:,:,162) =

  -0.0490 - 0.0947i  -0.2669 + 0.8981i
   0.0596 - 0.0350i   0.0452 - 0.0628i


s2p(:,:,163) =

  -0.0508 - 0.1025i  -0.2184 + 0.9085i
   0.0631 - 0.0401i   0.0447 - 0.0715i


s2p(:,:,164) =

  -0.0546 - 0.1096i  -0.1647 + 0.9169i
   0.0688 - 0.0491i   0.0452 - 0.0791i
```

```
s2p(:,:,165) =

  -0.0584 - 0.1167i  -0.1108 + 0.9222i
   0.0738 - 0.0588i   0.0450 - 0.0868i


s2p(:,:,166) =

  -0.0622 - 0.1238i  -0.0568 + 0.9243i
   0.0778 - 0.0692i   0.0443 - 0.0947i


s2p(:,:,167) =

  -0.0661 - 0.1309i  -0.0030 + 0.9233i
   0.0810 - 0.0801i   0.0430 - 0.1026i


s2p(:,:,168) =

  -0.0701 - 0.1380i   0.0503 + 0.9191i
   0.0828 - 0.0917i   0.0415 - 0.1106i


s2p(:,:,169) =

  -0.0754 - 0.1455i   0.1014 + 0.9120i
   0.0807 - 0.1049i   0.0415 - 0.1187i


s2p(:,:,170) =

  -0.0809 - 0.1530i   0.1518 + 0.9020i
   0.0771 - 0.1183i   0.0411 - 0.1269i


s2p(:,:,171) =

  -0.0864 - 0.1603i   0.2015 + 0.8893i
   0.0719 - 0.1317i   0.0404 - 0.1351i


s2p(:,:,172) =
```

```
  -0.0921 - 0.1676i   0.2502 + 0.8739i
   0.0650 - 0.1450i   0.0393 - 0.1433i


s2p(:,:,173) =

  -0.0984 - 0.1752i   0.2983 + 0.8546i
   0.0557 - 0.1575i   0.0380 - 0.1522i


s2p(:,:,174) =

  -0.1067 - 0.1837i   0.3464 + 0.8293i
   0.0427 - 0.1675i   0.0366 - 0.1629i


s2p(:,:,175) =

  -0.1153 - 0.1920i   0.3925 + 0.8012i
   0.0283 - 0.1765i   0.0347 - 0.1736i


s2p(:,:,176) =

  -0.1241 - 0.2000i   0.4365 + 0.7707i
   0.0126 - 0.1842i   0.0324 - 0.1844i


s2p(:,:,177) =

  -0.1332 - 0.2079i   0.4783 + 0.7378i
  -0.0042 - 0.1905i   0.0295 - 0.1951i


s2p(:,:,178) =

  -0.1437 - 0.2155i   0.5182 + 0.6989i
  -0.0217 - 0.1948i   0.0255 - 0.2069i


s2p(:,:,179) =

  -0.1566 - 0.2227i   0.5557 + 0.6517i
```

```
  -0.0395 - 0.1968i   0.0197 - 0.2204i


s2p(:,:,180) =

  -0.1698 - 0.2295i   0.5891 + 0.6028i
  -0.0578 - 0.1972i   0.0130 - 0.2337i


s2p(:,:,181) =

  -0.1835 - 0.2356i   0.6185 + 0.5523i
  -0.0764 - 0.1958i   0.0054 - 0.2468i


s2p(:,:,182) =

  -0.1976 - 0.2413i   0.6438 + 0.5006i
  -0.0952 - 0.1927i  -0.0030 - 0.2597i


s2p(:,:,183) =

  -0.2134 - 0.2446i   0.6731 + 0.4494i
  -0.1152 - 0.1853i  -0.0146 - 0.2718i


s2p(:,:,184) =

  -0.2309 - 0.2454i   0.7064 + 0.3972i
  -0.1355 - 0.1733i  -0.0295 - 0.2827i


s2p(:,:,185) =

  -0.2487 - 0.2452i   0.7357 + 0.3425i
  -0.1546 - 0.1591i  -0.0456 - 0.2928i


s2p(:,:,186) =

  -0.2668 - 0.2440i   0.7607 + 0.2856i
  -0.1722 - 0.1428i  -0.0627 - 0.3020i
```

```
s2p(:,:,187) =

  -0.2852 - 0.2417i   0.7813 + 0.2269i
  -0.1880 - 0.1245i  -0.0809 - 0.3101i


s2p(:,:,188) =

  -0.2988 - 0.2333i   0.7926 + 0.1666i
  -0.1996 - 0.0990i  -0.0985 - 0.3108i


s2p(:,:,189) =

  -0.3091 - 0.2212i   0.7963 + 0.1060i
  -0.2061 - 0.0696i  -0.1152 - 0.3067i


s2p(:,:,190) =

  -0.3189 - 0.2087i   0.7955 + 0.0459i
  -0.2084 - 0.0402i  -0.1317 - 0.3017i


s2p(:,:,191) =

  -0.3282 - 0.1958i   0.7901 - 0.0136i
  -0.2066 - 0.0115i  -0.1480 - 0.2958i


s2p(:,:,192) =

  -0.3369 - 0.1824i   0.7803 - 0.0720i
  -0.2010 + 0.0159i  -0.1641 - 0.2891i


s2p(:,:,193) =

  -0.3307 - 0.2152i   0.7327 - 0.1932i
  -0.1734 - 0.0669i  -0.1406 - 0.3114i


s2p(:,:,194) =
```

```
  -0.3135 - 0.2632i   0.6519 - 0.3181i
  -0.0886 - 0.1412i  -0.0989 - 0.3394i


s2p(:,:,195) =

  -0.2894 - 0.3101i   0.5516 - 0.4195i
   0.0120 - 0.1470i  -0.0521 - 0.3616i


s2p(:,:,196) =

  -0.2584 - 0.3548i   0.4378 - 0.4947i
   0.0849 - 0.0962i  -0.0010 - 0.3771i


s2p(:,:,197) =

  -0.2207 - 0.3964i   0.3167 - 0.5426i
   0.1068 - 0.0223i   0.0534 - 0.3852i


s2p(:,:,198) =

  -0.3023 - 0.4136i   0.3127 - 0.4565i
   0.1067 - 0.0724i  -0.0017 - 0.4508i


s2p(:,:,199) =

  -0.4096 - 0.4064i   0.3082 - 0.3577i
   0.0653 - 0.1397i  -0.0959 - 0.5110i


s2p(:,:,200) =

  -0.5210 - 0.3747i   0.2864 - 0.2663i
  -0.0179 - 0.1787i  -0.2114 - 0.5497i


s2p(:,:,201) =

  -0.6314 - 0.3170i   0.2490 - 0.1847i
```

```
  -0.1219 - 0.1647i  -0.3433 - 0.5613i
```

16-Port S-Parameters to 4-Port S-Parameters

Convert 16-port S-parameters to 4-port S-parameters by using ports 1, 16, 2, and 15 as the first, second, third, and fourth ports. Terminate the remaining 12 ports with an impedance of Z0.

```
ckt = read(rfckt.passive,'default.s16p');
s16p = ckt.NetworkData.Data;
Z0 = ckt.NetworkData.Z0;
s4p = snp2smp(s16p,Z0,[1 16 2 15],Z0)
```

```
s4p =
s4p(:,:,1) =

   0.0857 - 0.1168i  -0.5372 - 0.6804i   0.0966 - 0.0706i   0.0067 + 0.0053i
  -0.5366 - 0.6860i   0.0803 - 0.1234i   0.0059 + 0.0048i   0.0977 - 0.0703i
   0.0957 - 0.0700i   0.0067 + 0.0048i   0.0818 - 0.1104i  -0.5362 - 0.6838i
   0.0055 + 0.0051i   0.0972 - 0.0703i  -0.5376 - 0.6840i   0.0761 - 0.1180i


s4p(:,:,2) =

   0.0479 - 0.1334i  -0.7665 - 0.3900i   0.0586 - 0.1042i   0.0071 - 0.0003i
  -0.7674 - 0.3903i   0.0365 - 0.1395i   0.0070 - 0.0004i   0.0602 - 0.1034i
   0.0597 - 0.1028i   0.0062 + 0.0001i   0.0428 - 0.1282i  -0.7686 - 0.3880i
   0.0068 - 0.0001i   0.0607 - 0.1033i  -0.7682 - 0.3889i   0.0348 - 0.1310i


s4p(:,:,3) =

   0.0031 - 0.1361i  -0.8526 - 0.0298i   0.0118 - 0.1094i   0.0044 - 0.0045i
  -0.8535 - 0.0309i  -0.0084 - 0.1364i   0.0043 - 0.0041i   0.0140 - 0.1103i
   0.0107 - 0.1093i   0.0043 - 0.0040i   0.0005 - 0.1282i  -0.8536 - 0.0292i
   0.0047 - 0.0039i   0.0141 - 0.1100i  -0.8526 - 0.0288i  -0.0063 - 0.1275i


s4p(:,:,4) =

  -0.0362 - 0.1206i  -0.7807 + 0.3291i  -0.0284 - 0.0909i   0.0001 - 0.0043i
```

```
  -0.7805 + 0.3295i  -0.0459 - 0.1168i  -0.0004 - 0.0054i  -0.0261 - 0.0929i
  -0.0291 - 0.0912i  -0.0003 - 0.0052i  -0.0363 - 0.1105i  -0.7802 + 0.3327i
  -0.0001 - 0.0049i  -0.0263 - 0.0928i  -0.7798 + 0.3313i  -0.0404 - 0.1107i


s4p(:,:,5) =

  -0.0649 - 0.0912i  -0.5652 + 0.6246i  -0.0491 - 0.0579i  -0.0030 - 0.0022i
  -0.5640 + 0.6257i  -0.0717 - 0.0865i  -0.0041 - 0.0022i  -0.0473 - 0.0614i
  -0.0501 - 0.0576i  -0.0038 - 0.0024i  -0.0619 - 0.0819i  -0.5638 + 0.6259i
  -0.0035 - 0.0020i  -0.0477 - 0.0614i  -0.5628 + 0.6255i  -0.0646 - 0.0836i


s4p(:,:,6) =

  -0.0760 - 0.0541i  -0.2470 + 0.7983i  -0.0490 - 0.0247i  -0.0037 + 0.0024i
  -0.2483 + 0.7999i  -0.0810 - 0.0502i  -0.0045 + 0.0023i  -0.0481 - 0.0295i
  -0.0489 - 0.0253i  -0.0041 + 0.0025i  -0.0724 - 0.0479i  -0.2448 + 0.8009i
  -0.0038 + 0.0023i  -0.0475 - 0.0295i  -0.2471 + 0.8013i  -0.0749 - 0.0513i


s4p(:,:,7) =

  -0.0714 - 0.0200i   0.1122 + 0.8232i  -0.0321 - 0.0040i  -0.0004 + 0.0055i
   0.1127 + 0.8241i  -0.0737 - 0.0185i  -0.0009 + 0.0059i  -0.0331 - 0.0093i
  -0.0326 - 0.0040i  -0.0008 + 0.0055i  -0.0676 - 0.0163i   0.1154 + 0.8228i
  -0.0005 + 0.0059i  -0.0331 - 0.0093i   0.1151 + 0.8238i  -0.0708 - 0.0209i


s4p(:,:,8) =

  -0.0516 + 0.0022i   0.4469 + 0.6936i  -0.0116 - 0.0010i   0.0049 + 0.0056i
   0.4472 + 0.6934i  -0.0540 + 0.0025i   0.0049 + 0.0058i  -0.0150 - 0.0057i
  -0.0119 - 0.0012i   0.0049 + 0.0057i  -0.0497 + 0.0051i   0.4494 + 0.6931i
   0.0050 + 0.0055i  -0.0150 - 0.0058i   0.4490 + 0.6911i  -0.0534 + 0.0016i


s4p(:,:,9) =

  -0.0277 + 0.0060i   0.6935 + 0.4364i   0.0010 - 0.0123i   0.0097 + 0.0012i
   0.6933 + 0.4368i  -0.0296 + 0.0057i   0.0094 + 0.0011i  -0.0040 - 0.0156i
   0.0009 - 0.0123i   0.0094 + 0.0011i  -0.0277 + 0.0109i   0.6940 + 0.4357i
   0.0096 + 0.0009i  -0.0040 - 0.0157i   0.6951 + 0.4340i  -0.0307 + 0.0077i
```

```
s4p(:,:,10) =

  -0.0136 - 0.0068i   0.8055 + 0.1016i  -0.0017 - 0.0298i   0.0087 - 0.0069i
   0.8046 + 0.1017i  -0.0150 - 0.0075i   0.0085 - 0.0065i  -0.0075 - 0.0308i
  -0.0014 - 0.0300i   0.0083 - 0.0065i  -0.0143 + 0.0004i   0.8057 + 0.1004i
   0.0089 - 0.0068i  -0.0076 - 0.0307i   0.8059 + 0.0987i  -0.0129 - 0.0026i


s4p(:,:,11) =

  -0.0148 - 0.0237i   0.7676 - 0.2439i  -0.0170 - 0.0404i   0.0047 - 0.0114i
   0.7675 - 0.2439i  -0.0141 - 0.0259i   0.0044 - 0.0109i  -0.0224 - 0.0387i
  -0.0168 - 0.0403i   0.0045 - 0.0109i  -0.0151 - 0.0146i   0.7675 - 0.2471i
   0.0047 - 0.0114i  -0.0221 - 0.0389i   0.7673 - 0.2479i  -0.0088 - 0.0216i


s4p(:,:,12) =

  -0.0356 - 0.0360i   0.5868 - 0.5408i  -0.0407 - 0.0403i  -0.0034 - 0.0141i
   0.5872 - 0.5403i  -0.0338 - 0.0416i  -0.0031 - 0.0130i  -0.0443 - 0.0371i
  -0.0406 - 0.0402i  -0.0033 - 0.0131i  -0.0336 - 0.0249i   0.5842 - 0.5423i
  -0.0031 - 0.0139i  -0.0446 - 0.0371i   0.5859 - 0.5431i  -0.0246 - 0.0406i


s4p(:,:,13) =

  -0.0662 - 0.0284i   0.3018 - 0.7298i  -0.0635 - 0.0239i  -0.0118 - 0.0103i
   0.3028 - 0.7304i  -0.0657 - 0.0383i  -0.0110 - 0.0092i  -0.0659 - 0.0201i
  -0.0634 - 0.0239i  -0.0110 - 0.0091i  -0.0610 - 0.0157i   0.3000 - 0.7306i
  -0.0115 - 0.0102i  -0.0660 - 0.0204i   0.2996 - 0.7317i  -0.0558 - 0.0430i


s4p(:,:,14) =

  -0.0917 + 0.0025i  -0.0307 - 0.7801i  -0.0765 + 0.0051i  -0.0155 - 0.0015i
  -0.0315 - 0.7792i  -0.0944 - 0.0105i  -0.0144 - 0.0011i  -0.0769 + 0.0088i
  -0.0761 + 0.0046i  -0.0144 - 0.0012i  -0.0821 + 0.0145i  -0.0364 - 0.7790i
  -0.0158 - 0.0017i  -0.0770 + 0.0089i  -0.0354 - 0.7799i  -0.0879 - 0.0208i


s4p(:,:,15) =

  -0.0963 + 0.0478i  -0.3504 - 0.6877i  -0.0728 + 0.0388i  -0.0137 + 0.0074i
```

```
   -0.3510 - 0.6874i  -0.1031 + 0.0358i  -0.0119 + 0.0070i  -0.0723 + 0.0423i
   -0.0730 + 0.0385i  -0.0121 + 0.0072i  -0.0819 + 0.0582i  -0.3539 - 0.6859i
   -0.0136 + 0.0073i  -0.0725 + 0.0419i  -0.3542 - 0.6857i  -0.1035 + 0.0226i


s4p(:,:,16) =

   -0.0732 + 0.0920i  -0.5976 - 0.4743i  -0.0533 + 0.0679i  -0.0070 + 0.0123i
   -0.5993 - 0.4736i  -0.0826 + 0.0835i  -0.0056 + 0.0111i  -0.0516 + 0.0702i
   -0.0532 + 0.0678i  -0.0056 + 0.0109i  -0.0557 + 0.0992i  -0.6012 - 0.4711i
   -0.0070 + 0.0124i  -0.0518 + 0.0701i  -0.6003 - 0.4709i  -0.0906 + 0.0723i


s4p(:,:,17) =

   -0.0290 + 0.1190i  -0.7348 - 0.1811i  -0.0220 + 0.0840i   0.0002 + 0.0125i
   -0.7346 - 0.1822i  -0.0369 + 0.1137i   0.0006 + 0.0105i  -0.0201 + 0.0850i
   -0.0221 + 0.0839i   0.0006 + 0.0106i  -0.0094 + 0.1208i  -0.7350 - 0.1769i
    0.0002 + 0.0125i  -0.0203 + 0.0852i  -0.7359 - 0.1767i  -0.0503 + 0.1088i


s4p(:,:,18) =

    0.0215 + 0.1194i  -0.7381 + 0.1372i   0.0116 + 0.0836i   0.0051 + 0.0088i
   -0.7380 + 0.1376i   0.0178 + 0.1141i   0.0043 + 0.0072i   0.0129 + 0.0830i
    0.0114 + 0.0834i   0.0044 + 0.0071i   0.0416 + 0.1156i  -0.7372 + 0.1422i
    0.0052 + 0.0089i   0.0129 + 0.0834i  -0.7379 + 0.1428i   0.0039 + 0.1177i


s4p(:,:,19) =

    0.0632 + 0.0932i  -0.6125 + 0.4297i   0.0394 + 0.0669i   0.0052 + 0.0053i
   -0.6129 + 0.4291i   0.0635 + 0.0859i   0.0037 + 0.0041i   0.0394 + 0.0673i
    0.0392 + 0.0671i   0.0036 + 0.0036i   0.0812 + 0.0849i  -0.6097 + 0.4322i
    0.0050 + 0.0052i   0.0400 + 0.0675i  -0.6097 + 0.4322i   0.0535 + 0.0966i


s4p(:,:,20) =

    0.0810 + 0.0534i  -0.3771 + 0.6442i   0.0518 + 0.0414i   0.0049 + 0.0046i
   -0.3766 + 0.6435i   0.0832 + 0.0385i   0.0027 + 0.0047i   0.0524 + 0.0416i
    0.0519 + 0.0415i   0.0029 + 0.0047i   0.0966 + 0.0411i  -0.3729 + 0.6447i
    0.0049 + 0.0047i   0.0525 + 0.0414i  -0.3733 + 0.6444i   0.0802 + 0.0538i
```

```
s4p(:,:,21) =

   0.0744 + 0.0170i  -0.0731 + 0.7403i   0.0469 + 0.0174i   0.0067 + 0.0055i
  -0.0737 + 0.7411i   0.0716 - 0.0056i   0.0052 + 0.0065i   0.0476 + 0.0174i
   0.0471 + 0.0174i   0.0050 + 0.0064i   0.0862 + 0.0017i  -0.0696 + 0.7397i
   0.0067 + 0.0054i   0.0476 + 0.0173i  -0.0694 + 0.7398i   0.0771 + 0.0091i


s4p(:,:,22) =

   0.0516 - 0.0028i   0.2431 + 0.7003i   0.0300 + 0.0060i   0.0112 + 0.0040i
   0.2426 + 0.7005i   0.0401 - 0.0287i   0.0106 + 0.0054i   0.0305 + 0.0060i
   0.0300 + 0.0058i   0.0105 + 0.0055i   0.0591 - 0.0194i   0.2454 + 0.6986i
   0.0112 + 0.0040i   0.0305 + 0.0061i   0.2459 + 0.6993i   0.0518 - 0.0193i


s4p(:,:,23) =

   0.0292 - 0.0024i   0.5134 + 0.5301i   0.0124 + 0.0124i   0.0151 - 0.0020i
   0.5128 + 0.5306i   0.0080 - 0.0246i   0.0151 - 0.0009i   0.0130 + 0.0127i
   0.0123 + 0.0122i   0.0152 - 0.0008i   0.0319 - 0.0188i   0.5151 + 0.5283i
   0.0151 - 0.0021i   0.0131 + 0.0127i   0.5149 + 0.5273i   0.0215 - 0.0222i


s4p(:,:,24) =

   0.0182 + 0.0119i   0.6831 + 0.2629i   0.0074 + 0.0324i   0.0140 - 0.0113i
   0.6830 + 0.2633i  -0.0066 - 0.0021i   0.0143 - 0.0103i   0.0088 + 0.0328i
   0.0072 + 0.0322i   0.0143 - 0.0103i   0.0176 - 0.0027i   0.6842 + 0.2602i
   0.0140 - 0.0113i   0.0088 + 0.0328i   0.6842 + 0.2600i   0.0040 - 0.0061i


s4p(:,:,25) =

   0.0236 + 0.0276i   0.7237 - 0.0476i   0.0214 + 0.0541i   0.0067 - 0.0186i
   0.7246 - 0.0469i   0.0024 + 0.0205i   0.0069 - 0.0179i   0.0239 + 0.0536i
   0.0212 + 0.0540i   0.0070 - 0.0179i   0.0209 + 0.0151i   0.7227 - 0.0508i
   0.0066 - 0.0185i   0.0241 + 0.0537i   0.7235 - 0.0507i   0.0071 + 0.0141i


s4p(:,:,26) =

   0.0402 + 0.0343i   0.6325 - 0.3429i   0.0516 + 0.0630i  -0.0041 - 0.0199i
```

```
   0.6313 - 0.3416i   0.0270 + 0.0275i  -0.0037 - 0.0191i   0.0546 + 0.0611i
   0.0518 + 0.0630i  -0.0038 - 0.0190i   0.0371 + 0.0232i   0.6294 - 0.3451i
  -0.0040 - 0.0199i   0.0550 + 0.0610i   0.6292 - 0.3454i   0.0260 + 0.0231i


s4p(:,:,27) =

   0.0599 + 0.0286i   0.4280 - 0.5680i   0.0870 + 0.0510i  -0.0132 - 0.0152i
   0.4277 - 0.5682i   0.0529 + 0.0131i  -0.0124 - 0.0141i   0.0894 + 0.0474i
   0.0871 + 0.0510i  -0.0124 - 0.0142i   0.0561 + 0.0173i   0.4242 - 0.5709i
  -0.0133 - 0.0152i   0.0895 + 0.0469i   0.4242 - 0.5708i   0.0486 + 0.0137i


s4p(:,:,28) =

   0.0748 + 0.0116i   0.1527 - 0.6880i   0.1124 + 0.0185i  -0.0181 - 0.0069i
   0.1522 - 0.6881i   0.0648 - 0.0177i  -0.0164 - 0.0062i   0.1131 + 0.0137i
   0.1120 + 0.0186i  -0.0164 - 0.0064i   0.0687 - 0.0002i   0.1480 - 0.6890i
  -0.0181 - 0.0069i   0.1130 + 0.0131i   0.1474 - 0.6891i   0.0614 - 0.0114i


s4p(:,:,29) =

   0.0808 - 0.0112i  -0.1454 - 0.6846i   0.1167 - 0.0246i  -0.0183 + 0.0017i
  -0.1452 - 0.6838i   0.0560 - 0.0522i  -0.0160 + 0.0011i   0.1155 - 0.0302i
   0.1167 - 0.0239i  -0.0160 + 0.0012i   0.0707 - 0.0228i  -0.1500 - 0.6828i
  -0.0183 + 0.0016i   0.1153 - 0.0300i  -0.1498 - 0.6828i   0.0571 - 0.0414i


s4p(:,:,30) =

   0.0771 - 0.0354i  -0.4133 - 0.5588i   0.0983 - 0.0634i  -0.0154 + 0.0084i
  -0.4133 - 0.5588i   0.0294 - 0.0779i  -0.0134 + 0.0064i   0.0948 - 0.0685i
   0.0987 - 0.0634i  -0.0133 + 0.0064i   0.0624 - 0.0449i  -0.4179 - 0.5564i
  -0.0154 + 0.0084i   0.0947 - 0.0682i  -0.4170 - 0.5560i   0.0369 - 0.0660i


s4p(:,:,31) =

   0.0641 - 0.0578i  -0.6035 - 0.3350i   0.0639 - 0.0866i  -0.0106 + 0.0131i
  -0.6034 - 0.3351i  -0.0065 - 0.0862i  -0.0101 + 0.0102i   0.0584 - 0.0891i
   0.0639 - 0.0868i  -0.0101 + 0.0101i   0.0449 - 0.0626i  -0.6064 - 0.3314i
  -0.0106 + 0.0131i   0.0587 - 0.0887i  -0.6063 - 0.3316i   0.0076 - 0.0773i
```

```
s4p(:,:,32) =

   0.0415 - 0.0734i  -0.6848 - 0.0555i   0.0253 - 0.0870i  -0.0052 + 0.0149i
  -0.6848 - 0.0558i  -0.0389 - 0.0755i  -0.0068 + 0.0123i   0.0219 - 0.0863i
   0.0250 - 0.0867i  -0.0068 + 0.0123i   0.0214 - 0.0720i  -0.6853 - 0.0499i
  -0.0052 + 0.0150i   0.0222 - 0.0866i  -0.6860 - 0.0499i  -0.0216 - 0.0735i


s4p(:,:,33) =

   0.0154 - 0.0791i  -0.6442 + 0.2343i  -0.0018 - 0.0683i  -0.0006 + 0.0166i
  -0.6436 + 0.2336i  -0.0615 - 0.0550i  -0.0032 + 0.0154i  -0.0023 - 0.0684i
  -0.0016 - 0.0681i  -0.0033 + 0.0154i  -0.0045 - 0.0719i  -0.6424 + 0.2393i
  -0.0006 + 0.0167i  -0.0025 - 0.0687i  -0.6423 + 0.2394i  -0.0444 - 0.0599i


s4p(:,:,34) =

  -0.0118 - 0.0740i  -0.4858 + 0.4792i  -0.0096 - 0.0424i   0.0057 + 0.0174i
  -0.4850 + 0.4799i  -0.0717 - 0.0291i   0.0030 + 0.0178i  -0.0088 - 0.0450i
  -0.0094 - 0.0425i   0.0030 + 0.0178i  -0.0289 - 0.0615i  -0.4814 + 0.4835i
   0.0057 + 0.0175i  -0.0090 - 0.0450i  -0.4814 + 0.4836i  -0.0590 - 0.0403i


s4p(:,:,35) =

  -0.0332 - 0.0572i  -0.2395 + 0.6356i   0.0006 - 0.0233i   0.0132 + 0.0158i
  -0.2392 + 0.6348i  -0.0694 - 0.0052i   0.0112 + 0.0171i   0.0009 - 0.0295i
   0.0007 - 0.0232i   0.0113 + 0.0171i  -0.0463 - 0.0421i  -0.2334 + 0.6370i
   0.0131 + 0.0157i   0.0008 - 0.0295i  -0.2339 + 0.6368i  -0.0644 - 0.0190i


s4p(:,:,36) =

  -0.0433 - 0.0342i   0.0479 + 0.6736i   0.0197 - 0.0200i   0.0207 + 0.0102i
   0.0482 + 0.6728i  -0.0590 + 0.0108i   0.0199 + 0.0119i   0.0165 - 0.0301i
   0.0196 - 0.0201i   0.0200 + 0.0119i  -0.0528 - 0.0182i   0.0530 + 0.6732i
   0.0206 + 0.0102i   0.0164 - 0.0298i   0.0540 + 0.6716i  -0.0612 + 0.0005i


s4p(:,:,37) =

  -0.0402 - 0.0128i   0.3240 + 0.5872i   0.0348 - 0.0347i   0.0261 - 0.0000i
```

```
    0.3234 + 0.5866i  -0.0464 + 0.0173i   0.0259 + 0.0015i   0.0251 - 0.0461i
    0.0347 - 0.0347i   0.0258 + 0.0016i  -0.0479 + 0.0028i   0.3279 + 0.5841i
    0.0260 + 0.0000i   0.0251 - 0.0460i   0.3275 + 0.5835i  -0.0521 + 0.0144i


s4p(:,:,38) =

   -0.0289 - 0.0023i   0.5345 + 0.3935i   0.0345 - 0.0605i   0.0252 - 0.0141i
    0.5336 + 0.3935i  -0.0376 + 0.0151i   0.0252 - 0.0126i   0.0171 - 0.0690i
    0.0345 - 0.0603i   0.0253 - 0.0127i  -0.0362 + 0.0147i   0.5369 + 0.3902i
    0.0251 - 0.0141i   0.0173 - 0.0691i   0.5363 + 0.3898i  -0.0405 + 0.0205i


s4p(:,:,39) =

   -0.0221 - 0.0024i   0.6400 + 0.1340i   0.0126 - 0.0830i   0.0139 - 0.0256i
    0.6399 + 0.1345i  -0.0389 + 0.0106i   0.0144 - 0.0240i  -0.0101 - 0.0841i
    0.0128 - 0.0829i   0.0143 - 0.0240i  -0.0265 + 0.0168i   0.6429 + 0.1311i
    0.0139 - 0.0255i  -0.0101 - 0.0842i   0.6422 + 0.1310i  -0.0326 + 0.0201i


s4p(:,:,40) =

   -0.0164 - 0.0045i   0.6355 - 0.1371i  -0.0166 - 0.0879i   0.0028 - 0.0273i
    0.6355 - 0.1370i  -0.0396 + 0.0117i   0.0039 - 0.0256i  -0.0403 - 0.0796i
   -0.0162 - 0.0880i   0.0038 - 0.0258i  -0.0191 + 0.0140i   0.6361 - 0.1447i
    0.0029 - 0.0274i  -0.0403 - 0.0796i   0.6363 - 0.1441i  -0.0260 + 0.0166i


s4p(:,:,41) =

   -0.0225 - 0.0161i   0.5202 - 0.3888i  -0.0482 - 0.0814i  -0.0100 - 0.0280i
    0.5197 - 0.3881i  -0.0464 + 0.0081i  -0.0079 - 0.0267i  -0.0685 - 0.0635i
   -0.0482 - 0.0814i  -0.0079 - 0.0267i  -0.0230 + 0.0057i   0.5160 - 0.3942i
   -0.0100 - 0.0281i  -0.0686 - 0.0635i   0.5160 - 0.3942i  -0.0293 + 0.0071i


s4p(:,:,42) =

   -0.0433 - 0.0177i   0.3090 - 0.5661i  -0.0749 - 0.0580i  -0.0225 - 0.0206i
    0.3086 - 0.5655i  -0.0608 + 0.0134i  -0.0195 - 0.0204i  -0.0873 - 0.0333i
   -0.0749 - 0.0578i  -0.0197 - 0.0203i  -0.0378 + 0.0063i   0.3038 - 0.5689i
   -0.0225 - 0.0205i  -0.0871 - 0.0332i   0.3034 - 0.5691i  -0.0435 + 0.0045i
```

```
s4p(:,:,43) =

  -0.0667 - 0.0012i   0.0446 - 0.6389i  -0.0870 - 0.0247i  -0.0295 - 0.0078i
   0.0451 - 0.6381i  -0.0738 + 0.0307i  -0.0265 - 0.0094i  -0.0893 + 0.0024i
  -0.0866 - 0.0244i  -0.0266 - 0.0094i  -0.0535 + 0.0214i   0.0393 - 0.6400i
  -0.0295 - 0.0078i  -0.0892 + 0.0022i   0.0392 - 0.6393i  -0.0617 + 0.0144i


s4p(:,:,44) =

  -0.0786 + 0.0321i  -0.2223 - 0.5959i  -0.0819 + 0.0093i  -0.0292 + 0.0063i
  -0.2221 - 0.5952i  -0.0770 + 0.0586i  -0.0273 + 0.0033i  -0.0747 + 0.0336i
  -0.0816 + 0.0089i  -0.0274 + 0.0032i  -0.0587 + 0.0503i  -0.2279 - 0.5938i
  -0.0291 + 0.0062i  -0.0748 + 0.0334i  -0.2283 - 0.5929i  -0.0736 + 0.0385i


s4p(:,:,45) =

  -0.0693 + 0.0712i  -0.4465 - 0.4468i  -0.0629 + 0.0342i  -0.0228 + 0.0177i
  -0.4456 - 0.4467i  -0.0635 + 0.0890i  -0.0227 + 0.0137i  -0.0496 + 0.0519i
  -0.0632 + 0.0337i  -0.0228 + 0.0137i  -0.0443 + 0.0823i  -0.4509 - 0.4424i
  -0.0227 + 0.0175i  -0.0496 + 0.0521i  -0.4500 - 0.4422i  -0.0703 + 0.0698i


s4p(:,:,46) =

  -0.0389 + 0.1013i  -0.5866 - 0.2202i  -0.0385 + 0.0447i  -0.0131 + 0.0246i
  -0.5857 - 0.2207i  -0.0332 + 0.1098i  -0.0153 + 0.0211i  -0.0228 + 0.0550i
  -0.0388 + 0.0448i  -0.0152 + 0.0212i  -0.0113 + 0.1034i  -0.5892 - 0.2154i
  -0.0131 + 0.0245i  -0.0228 + 0.0550i  -0.5888 - 0.2145i  -0.0497 + 0.0970i


s4p(:,:,47) =

   0.0029 + 0.1118i  -0.6210 + 0.0409i  -0.0175 + 0.0430i  -0.0018 + 0.0266i
  -0.6203 + 0.0403i   0.0040 + 0.1125i  -0.0055 + 0.0250i  -0.0039 + 0.0460i
  -0.0173 + 0.0431i  -0.0055 + 0.0251i   0.0296 + 0.1030i  -0.6213 + 0.0468i
  -0.0018 + 0.0265i  -0.0039 + 0.0460i  -0.6212 + 0.0471i  -0.0173 + 0.1105i


s4p(:,:,48) =

   0.0441 + 0.0986i  -0.5477 + 0.2909i  -0.0029 + 0.0328i   0.0069 + 0.0244i
```

```
  -0.5468 + 0.2896i   0.0389 + 0.0987i   0.0031 + 0.0247i   0.0049 + 0.0343i
  -0.0029 + 0.0328i   0.0031 + 0.0247i   0.0642 + 0.0797i  -0.5450 + 0.2975i
   0.0068 + 0.0244i   0.0050 + 0.0344i  -0.5441 + 0.2976i   0.0176 + 0.1065i


s4p(:,:,49) =

   0.0672 + 0.0679i  -0.3769 + 0.4907i  -0.0013 + 0.0193i   0.0155 + 0.0219i
  -0.3777 + 0.4901i   0.0619 + 0.0665i   0.0120 + 0.0240i   0.0044 + 0.0225i
  -0.0014 + 0.0193i   0.0120 + 0.0240i   0.0772 + 0.0423i  -0.3714 + 0.4957i
   0.0154 + 0.0219i   0.0044 + 0.0225i  -0.3708 + 0.4953i   0.0428 + 0.0850i


s4p(:,:,50) =

   0.0692 + 0.0373i  -0.1383 + 0.6016i  -0.0102 + 0.0130i   0.0247 + 0.0156i
  -0.1385 + 0.6016i   0.0620 + 0.0295i   0.0226 + 0.0194i  -0.0046 + 0.0184i
  -0.0103 + 0.0129i   0.0226 + 0.0193i   0.0663 + 0.0085i  -0.1307 + 0.6033i
   0.0246 + 0.0154i  -0.0045 + 0.0183i  -0.1306 + 0.6033i   0.0503 + 0.0582i


s4p(:,:,51) =

   0.0581 + 0.0185i   0.1251 + 0.6023i  -0.0221 + 0.0186i   0.0310 + 0.0038i
   0.1242 + 0.6018i   0.0422 + 0.0037i   0.0316 + 0.0085i  -0.0139 + 0.0259i
  -0.0221 + 0.0186i   0.0316 + 0.0086i   0.0402 - 0.0080i   0.1317 + 0.6002i
   0.0309 + 0.0039i  -0.0140 + 0.0258i   0.1319 + 0.6001i   0.0431 + 0.0389i


s4p(:,:,52) =

   0.0461 + 0.0142i   0.3632 + 0.4930i  -0.0278 + 0.0359i   0.0311 - 0.0108i
   0.3629 + 0.4924i   0.0155 - 0.0026i   0.0344 - 0.0074i  -0.0153 + 0.0432i
  -0.0277 + 0.0360i   0.0344 - 0.0074i   0.0148 - 0.0029i   0.3687 + 0.4871i
   0.0311 - 0.0107i  -0.0154 + 0.0432i   0.3683 + 0.4875i   0.0315 + 0.0329i


s4p(:,:,53) =

   0.0428 + 0.0187i   0.5317 + 0.2937i  -0.0203 + 0.0586i   0.0236 - 0.0245i
   0.5315 + 0.2940i  -0.0039 + 0.0089i   0.0280 - 0.0237i  -0.0030 + 0.0626i
  -0.0203 + 0.0584i   0.0281 - 0.0236i   0.0034 + 0.0158i   0.5344 + 0.2872i
   0.0235 - 0.0245i  -0.0031 + 0.0627i   0.5344 + 0.2874i   0.0264 + 0.0377i
```

```
s4p(:,:,54) =

   0.0507 + 0.0216i   0.6008 + 0.0455i   0.0021 + 0.0769i   0.0099 - 0.0326i
   0.6001 + 0.0455i  -0.0081 + 0.0282i   0.0139 - 0.0346i   0.0226 + 0.0745i
   0.0018 + 0.0766i   0.0139 - 0.0346i   0.0094 + 0.0347i   0.6006 + 0.0388i
   0.0099 - 0.0325i   0.0226 + 0.0746i   0.6006 + 0.0391i   0.0324 + 0.0435i


s4p(:,:,55) =

   0.0642 + 0.0147i   0.5611 - 0.2061i   0.0341 + 0.0819i  -0.0056 - 0.0330i
   0.5597 - 0.2059i   0.0031 + 0.0436i  -0.0038 - 0.0366i   0.0545 + 0.0713i
   0.0337 + 0.0820i  -0.0038 - 0.0366i   0.0277 + 0.0426i   0.5582 - 0.2117i
  -0.0055 - 0.0330i   0.0546 + 0.0713i   0.5578 - 0.2109i   0.0459 + 0.0416i


s4p(:,:,56) =

   0.0736 - 0.0038i   0.4223 - 0.4162i   0.0660 + 0.0701i  -0.0182 - 0.0265i
   0.4221 - 0.4155i   0.0219 + 0.0470i  -0.0189 - 0.0301i   0.0823 + 0.0514i
   0.0657 + 0.0702i  -0.0189 - 0.0302i   0.0468 + 0.0350i   0.4189 - 0.4205i
  -0.0181 - 0.0266i   0.0824 + 0.0513i   0.4178 - 0.4197i   0.0581 + 0.0288i


s4p(:,:,57) =

   0.0722 - 0.0272i   0.2118 - 0.5495i   0.0891 + 0.0448i  -0.0263 - 0.0169i
   0.2120 - 0.5494i   0.0408 + 0.0376i  -0.0290 - 0.0188i   0.0979 + 0.0198i
   0.0892 + 0.0450i  -0.0290 - 0.0189i   0.0568 + 0.0166i   0.2067 - 0.5528i
  -0.0264 - 0.0168i   0.0978 + 0.0199i   0.2065 - 0.5514i   0.0617 + 0.0093i


s4p(:,:,58) =

   0.0599 - 0.0488i  -0.0336 - 0.5852i   0.0976 + 0.0127i  -0.0305 - 0.0056i
  -0.0335 - 0.5845i   0.0514 + 0.0168i  -0.0331 - 0.0054i   0.0958 - 0.0154i
   0.0976 + 0.0127i  -0.0332 - 0.0055i   0.0545 - 0.0045i  -0.0396 - 0.5862i
  -0.0304 - 0.0056i   0.0960 - 0.0152i  -0.0401 - 0.5854i   0.0546 - 0.0104i


s4p(:,:,59) =

   0.0401 - 0.0632i  -0.2719 - 0.5170i   0.0903 - 0.0174i  -0.0310 + 0.0063i
```

```
  -0.2717 - 0.5163i   0.0485 - 0.0075i  -0.0323 + 0.0080i   0.0770 - 0.0428i
   0.0900 - 0.0171i  -0.0322 + 0.0080i   0.0416 - 0.0202i  -0.2782 - 0.5151i
  -0.0308 + 0.0062i   0.0772 - 0.0429i  -0.2778 - 0.5146i   0.0385 - 0.0235i


s4p(:,:,60) =

   0.0181 - 0.0690i  -0.4597 - 0.3560i   0.0717 - 0.0364i  -0.0268 + 0.0185i
  -0.4592 - 0.3556i   0.0329 - 0.0274i  -0.0265 + 0.0200i   0.0490 - 0.0542i
   0.0718 - 0.0361i  -0.0265 + 0.0200i   0.0243 - 0.0265i  -0.4649 - 0.3514i
  -0.0269 + 0.0184i   0.0492 - 0.0544i  -0.4642 - 0.3513i   0.0197 - 0.0265i


s4p(:,:,61) =

  -0.0027 - 0.0681i  -0.5633 - 0.1328i   0.0511 - 0.0408i  -0.0174 + 0.0281i
  -0.5621 - 0.1322i   0.0096 - 0.0362i  -0.0165 + 0.0286i   0.0242 - 0.0469i
   0.0512 - 0.0406i  -0.0165 + 0.0286i   0.0090 - 0.0252i  -0.5661 - 0.1265i
  -0.0175 + 0.0282i   0.0243 - 0.0471i  -0.5655 - 0.1263i   0.0042 - 0.0214i


s4p(:,:,62) =

  -0.0203 - 0.0623i  -0.5656 + 0.1129i   0.0389 - 0.0331i  -0.0056 + 0.0335i
  -0.5644 + 0.1119i  -0.0133 - 0.0353i  -0.0049 + 0.0331i   0.0142 - 0.0293i
   0.0390 - 0.0331i  -0.0048 + 0.0331i  -0.0012 - 0.0202i  -0.5655 + 0.1198i
  -0.0055 + 0.0334i   0.0142 - 0.0294i  -0.5649 + 0.1197i  -0.0053 - 0.0132i


s4p(:,:,63) =

  -0.0350 - 0.0556i  -0.4648 + 0.3370i   0.0403 - 0.0241i   0.0083 + 0.0345i
  -0.4649 + 0.3357i  -0.0352 - 0.0252i   0.0086 + 0.0339i   0.0200 - 0.0128i
   0.0403 - 0.0241i   0.0086 + 0.0338i  -0.0085 - 0.0170i  -0.4608 + 0.3435i
   0.0083 + 0.0345i   0.0200 - 0.0128i  -0.4615 + 0.3438i  -0.0117 - 0.0057i


s4p(:,:,64) =

  -0.0506 - 0.0472i  -0.2800 + 0.4967i   0.0515 - 0.0233i   0.0224 + 0.0296i
  -0.2801 + 0.4966i  -0.0500 - 0.0063i   0.0223 + 0.0288i   0.0381 - 0.0071i
   0.0514 - 0.0231i   0.0224 + 0.0289i  -0.0182 - 0.0147i  -0.2738 + 0.5016i
   0.0224 + 0.0295i   0.0382 - 0.0072i  -0.2734 + 0.5004i  -0.0176 + 0.0022i
```

```
s4p(:,:,65) =

  -0.0657 - 0.0333i  -0.0462 + 0.5637i   0.0644 - 0.0360i   0.0342 + 0.0188i
  -0.0469 + 0.5636i  -0.0548 + 0.0158i   0.0338 + 0.0180i   0.0587 - 0.0192i
   0.0645 - 0.0356i   0.0338 + 0.0180i  -0.0314 - 0.0089i  -0.0393 + 0.5649i
   0.0342 + 0.0188i   0.0587 - 0.0193i  -0.0390 + 0.5649i  -0.0219 + 0.0124i


s4p(:,:,66) =

  -0.0762 - 0.0136i   0.1906 + 0.5263i   0.0686 - 0.0609i   0.0412 + 0.0029i
   0.1906 + 0.5270i  -0.0497 + 0.0354i   0.0402 + 0.0021i   0.0684 - 0.0475i
   0.0689 - 0.0607i   0.0401 + 0.0021i  -0.0445 + 0.0042i   0.1975 + 0.5252i
   0.0413 + 0.0029i   0.0682 - 0.0475i   0.1972 + 0.5252i  -0.0229 + 0.0252i


s4p(:,:,67) =

  -0.0785 + 0.0084i   0.3867 + 0.3949i   0.0565 - 0.0906i   0.0403 - 0.0165i
   0.3868 + 0.3948i  -0.0391 + 0.0478i   0.0385 - 0.0167i   0.0576 - 0.0816i
   0.0569 - 0.0906i   0.0384 - 0.0167i  -0.0530 + 0.0242i   0.3932 + 0.3921i
   0.0404 - 0.0166i   0.0576 - 0.0815i   0.3931 + 0.3912i  -0.0174 + 0.0383i


s4p(:,:,68) =

  -0.0755 + 0.0296i   0.5062 + 0.1990i   0.0261 - 0.1119i   0.0281 - 0.0335i
   0.5063 + 0.1986i  -0.0303 + 0.0553i   0.0259 - 0.0320i   0.0250 - 0.1056i
   0.0263 - 0.1120i   0.0260 - 0.0320i  -0.0537 + 0.0492i   0.5130 + 0.1935i
   0.0282 - 0.0335i   0.0251 - 0.1054i   0.5125 + 0.1932i  -0.0058 + 0.0496i


s4p(:,:,69) =

  -0.0600 + 0.0506i   0.5404 - 0.0206i  -0.0092 - 0.1142i   0.0131 - 0.0406i
   0.5397 - 0.0207i  -0.0160 + 0.0630i   0.0118 - 0.0375i  -0.0128 - 0.1068i
  -0.0089 - 0.1139i   0.0119 - 0.0376i  -0.0399 + 0.0750i   0.5442 - 0.0310i
   0.0131 - 0.0406i  -0.0127 - 0.1069i   0.5430 - 0.0311i   0.0159 + 0.0543i


s4p(:,:,70) =

  -0.0386 + 0.0533i   0.4850 - 0.2405i  -0.0414 - 0.1050i  -0.0042 - 0.0457i
```

```
   0.4852 - 0.2402i  -0.0010 + 0.0570i  -0.0032 - 0.0417i  -0.0465 - 0.0946i
  -0.0408 - 0.1048i  -0.0031 - 0.0418i  -0.0163 + 0.0882i   0.4815 - 0.2516i
  -0.0042 - 0.0457i  -0.0468 - 0.0948i   0.4816 - 0.2514i   0.0381 + 0.0412i


s4p(:,:,71) =

  -0.0275 + 0.0455i   0.3423 - 0.4194i  -0.0673 - 0.0827i  -0.0253 - 0.0415i
   0.3424 - 0.4193i   0.0030 + 0.0465i  -0.0214 - 0.0387i  -0.0722 - 0.0679i
  -0.0672 - 0.0830i  -0.0216 - 0.0387i   0.0072 + 0.0908i   0.3343 - 0.4266i
  -0.0253 - 0.0415i  -0.0723 - 0.0678i   0.3344 - 0.4250i   0.0481 + 0.0165i


s4p(:,:,72) =

  -0.0280 + 0.0368i   0.1380 - 0.5235i  -0.0803 - 0.0535i  -0.0431 - 0.0265i
   0.1383 - 0.5228i  -0.0020 + 0.0414i  -0.0376 - 0.0266i  -0.0814 - 0.0330i
  -0.0803 - 0.0538i  -0.0377 - 0.0266i   0.0271 + 0.0847i   0.1282 - 0.5241i
  -0.0430 - 0.0265i  -0.0813 - 0.0329i   0.1284 - 0.5240i   0.0414 - 0.0099i


s4p(:,:,73) =

  -0.0371 + 0.0348i  -0.0921 - 0.5322i  -0.0794 - 0.0257i  -0.0512 - 0.0042i
  -0.0920 - 0.5316i  -0.0068 + 0.0461i  -0.0467 - 0.0080i  -0.0720 - 0.0014i
  -0.0793 - 0.0258i  -0.0467 - 0.0079i   0.0412 + 0.0734i  -0.0995 - 0.5296i
  -0.0512 - 0.0042i  -0.0721 - 0.0014i  -0.1000 - 0.5289i   0.0209 - 0.0277i


s4p(:,:,74) =

  -0.0479 + 0.0437i  -0.3048 - 0.4429i  -0.0692 - 0.0058i  -0.0480 + 0.0195i
  -0.3042 - 0.4426i  -0.0039 + 0.0559i  -0.0468 + 0.0133i  -0.0508 + 0.0168i
  -0.0690 - 0.0060i  -0.0470 + 0.0133i   0.0495 + 0.0601i  -0.3094 - 0.4390i
  -0.0479 + 0.0195i  -0.0509 + 0.0168i  -0.3088 - 0.4386i  -0.0054 - 0.0303i


s4p(:,:,75) =

  -0.0524 + 0.0614i  -0.4582 - 0.2741i  -0.0566 + 0.0041i  -0.0337 + 0.0388i
  -0.4576 - 0.2740i   0.0088 + 0.0623i  -0.0371 + 0.0335i  -0.0289 + 0.0182i
  -0.0566 + 0.0038i  -0.0371 + 0.0335i   0.0522 + 0.0468i  -0.4617 - 0.2694i
  -0.0338 + 0.0388i  -0.0288 + 0.0182i  -0.4615 - 0.2697i  -0.0268 - 0.0182i
```

```
s4p(:,:,76) =

  -0.0456 + 0.0825i  -0.5257 - 0.0596i  -0.0476 + 0.0073i  -0.0128 + 0.0480i
  -0.5251 - 0.0597i   0.0242 + 0.0588i  -0.0181 + 0.0464i  -0.0175 + 0.0080i
  -0.0476 + 0.0071i  -0.0181 + 0.0466i   0.0502 + 0.0357i  -0.5288 - 0.0538i
  -0.0128 + 0.0480i  -0.0175 + 0.0080i  -0.5287 - 0.0541i  -0.0371 + 0.0016i


s4p(:,:,77) =

  -0.0278 + 0.0956i  -0.5008 + 0.1608i  -0.0430 + 0.0062i   0.0067 + 0.0478i
  -0.5010 + 0.1603i   0.0376 + 0.0472i   0.0022 + 0.0491i  -0.0176 - 0.0025i
  -0.0428 + 0.0060i   0.0022 + 0.0492i   0.0464 + 0.0262i  -0.5026 + 0.1691i
   0.0067 + 0.0477i  -0.0176 - 0.0025i  -0.5019 + 0.1691i  -0.0340 + 0.0206i


s4p(:,:,78) =

  -0.0120 + 0.0985i  -0.3889 + 0.3542i  -0.0472 + 0.0057i   0.0254 + 0.0422i
  -0.3884 + 0.3529i   0.0421 + 0.0244i   0.0227 + 0.0455i  -0.0276 - 0.0095i
  -0.0471 + 0.0055i   0.0228 + 0.0456i   0.0375 + 0.0187i  -0.3854 + 0.3615i
   0.0255 + 0.0422i  -0.0277 - 0.0094i  -0.3853 + 0.3616i  -0.0251 + 0.0294i


s4p(:,:,79) =

  -0.0008 + 0.0992i  -0.2054 + 0.4803i  -0.0558 + 0.0141i   0.0420 + 0.0286i
  -0.2052 + 0.4804i   0.0278 + 0.0019i   0.0417 + 0.0323i  -0.0444 - 0.0048i
  -0.0557 + 0.0139i   0.0418 + 0.0324i   0.0251 + 0.0185i  -0.1982 + 0.4866i
   0.0419 + 0.0287i  -0.0444 - 0.0048i  -0.1979 + 0.4867i  -0.0200 + 0.0310i


s4p(:,:,80) =

   0.0083 + 0.1018i   0.0133 + 0.5192i  -0.0603 + 0.0325i   0.0515 + 0.0080i
   0.0128 + 0.5186i  -0.0007 - 0.0052i   0.0531 + 0.0104i  -0.0576 + 0.0143i
  -0.0603 + 0.0322i   0.0531 + 0.0104i   0.0153 + 0.0277i   0.0221 + 0.5219i
   0.0515 + 0.0080i  -0.0575 + 0.0143i   0.0220 + 0.5213i  -0.0215 + 0.0328i


s4p(:,:,81) =

   0.0199 + 0.1076i   0.2261 + 0.4636i  -0.0543 + 0.0561i   0.0505 - 0.0157i
```

```
   0.2251 + 0.4635i  -0.0290 + 0.0091i   0.0524 - 0.0147i  -0.0576 + 0.0427i
  -0.0545 + 0.0559i   0.0524 - 0.0148i   0.0146 + 0.0429i   0.2343 + 0.4616i
   0.0504 - 0.0157i  -0.0577 + 0.0427i   0.2345 + 0.4614i  -0.0258 + 0.0408i


s4p(:,:,82) =

   0.0390 + 0.1129i   0.3936 + 0.3260i  -0.0360 + 0.0771i   0.0383 - 0.0363i
   0.3936 + 0.3260i  -0.0426 + 0.0401i   0.0399 - 0.0365i  -0.0408 + 0.0701i
  -0.0361 + 0.0771i   0.0399 - 0.0366i   0.0257 + 0.0576i   0.4011 + 0.3206i
   0.0384 - 0.0362i  -0.0408 + 0.0701i   0.4016 + 0.3209i  -0.0258 + 0.0551i


s4p(:,:,83) =

   0.0654 + 0.1100i   0.4891 + 0.1336i  -0.0081 + 0.0891i   0.0185 - 0.0487i
   0.4891 + 0.1336i  -0.0339 + 0.0741i   0.0191 - 0.0493i  -0.0104 + 0.0867i
  -0.0086 + 0.0889i   0.0190 - 0.0493i   0.0470 + 0.0630i   0.4942 + 0.1257i
   0.0185 - 0.0487i  -0.0104 + 0.0867i   0.4936 + 0.1256i  -0.0167 + 0.0711i


s4p(:,:,84) =

   0.0930 + 0.0934i   0.4973 - 0.0790i   0.0223 + 0.0881i  -0.0040 - 0.0510i
   0.4967 - 0.0786i  -0.0064 + 0.0965i  -0.0040 - 0.0510i   0.0250 + 0.0863i
   0.0219 + 0.0879i  -0.0040 - 0.0510i   0.0705 + 0.0535i   0.4986 - 0.0886i
  -0.0040 - 0.0509i   0.0249 + 0.0864i   0.4987 - 0.0880i   0.0011 + 0.0820i


s4p(:,:,85) =

   0.1126 + 0.0625i   0.4184 - 0.2739i   0.0490 + 0.0745i  -0.0242 - 0.0434i
   0.4178 - 0.2737i   0.0282 + 0.0986i  -0.0238 - 0.0428i   0.0547 + 0.0689i
   0.0487 + 0.0749i  -0.0239 - 0.0428i   0.0858 + 0.0301i   0.4163 - 0.2843i
  -0.0243 - 0.0434i   0.0549 + 0.0690i   0.4157 - 0.2841i   0.0233 + 0.0823i


s4p(:,:,86) =

   0.1149 + 0.0238i   0.2684 - 0.4198i   0.0656 + 0.0524i  -0.0384 - 0.0293i
   0.2684 - 0.4192i   0.0550 + 0.0822i  -0.0372 - 0.0288i   0.0703 + 0.0417i
   0.0653 + 0.0528i  -0.0372 - 0.0288i   0.0846 + 0.0007i   0.2614 - 0.4283i
  -0.0383 - 0.0293i   0.0704 + 0.0416i   0.2612 - 0.4284i   0.0416 + 0.0716i
```

```
s4p(:,:,87) =

   0.0977 - 0.0097i   0.0714 - 0.4926i   0.0693 + 0.0298i  -0.0462 - 0.0126i
   0.0716 - 0.4926i   0.0683 + 0.0578i  -0.0446 - 0.0130i   0.0708 + 0.0150i
   0.0692 + 0.0298i  -0.0446 - 0.0129i   0.0674 - 0.0232i   0.0614 - 0.4974i
  -0.0463 - 0.0126i   0.0707 + 0.0148i   0.0611 - 0.4974i   0.0517 + 0.0556i


s4p(:,:,88) =

   0.0692 - 0.0279i  -0.1384 - 0.4775i   0.0641 + 0.0133i  -0.0487 + 0.0064i
  -0.1388 - 0.4774i   0.0666 + 0.0330i  -0.0471 + 0.0049i   0.0590 - 0.0045i
   0.0640 + 0.0134i  -0.0472 + 0.0049i   0.0409 - 0.0328i  -0.1505 - 0.4768i
  -0.0486 + 0.0064i   0.0589 - 0.0043i  -0.1502 - 0.4769i   0.0529 + 0.0395i


s4p(:,:,89) =

   0.0407 - 0.0289i  -0.3245 - 0.3759i   0.0559 + 0.0062i  -0.0433 + 0.0262i
  -0.3242 - 0.3754i   0.0544 + 0.0177i  -0.0427 + 0.0240i   0.0421 - 0.0103i
   0.0560 + 0.0063i  -0.0428 + 0.0241i   0.0154 - 0.0269i  -0.3337 - 0.3693i
  -0.0432 + 0.0262i   0.0422 - 0.0102i  -0.3342 - 0.3696i   0.0484 + 0.0287i


s4p(:,:,90) =

   0.0207 - 0.0180i  -0.4495 - 0.2056i   0.0516 + 0.0071i  -0.0292 + 0.0434i
  -0.4498 - 0.2050i   0.0418 + 0.0142i  -0.0298 + 0.0411i   0.0302 - 0.0023i
   0.0515 + 0.0071i  -0.0298 + 0.0410i  -0.0000 - 0.0097i  -0.4556 - 0.1962i
  -0.0291 + 0.0434i   0.0303 - 0.0023i  -0.4553 - 0.1954i   0.0440 + 0.0247i


s4p(:,:,91) =

   0.0124 - 0.0027i  -0.4909 + 0.0004i   0.0552 + 0.0121i  -0.0078 + 0.0523i
  -0.4909 + 0.0007i   0.0377 + 0.0173i  -0.0095 + 0.0505i   0.0328 + 0.0126i
   0.0551 + 0.0122i  -0.0095 + 0.0506i  -0.0022 + 0.0097i  -0.4919 + 0.0115i
  -0.0078 + 0.0522i   0.0329 + 0.0125i  -0.4925 + 0.0120i   0.0457 + 0.0226i


s4p(:,:,92) =

   0.0159 + 0.0096i  -0.4437 + 0.2033i   0.0703 + 0.0141i   0.0149 + 0.0510i
```

```
 -0.4437 + 0.2033i   0.0404 + 0.0156i   0.0126 + 0.0503i   0.0503 + 0.0210i
  0.0702 + 0.0144i   0.0126 + 0.0505i   0.0068 + 0.0227i  -0.4400 + 0.2139i
  0.0149 + 0.0509i   0.0503 + 0.0209i  -0.4396 + 0.2135i   0.0501 + 0.0151i


s4p(:,:,93) =

  0.0244 + 0.0110i  -0.3158 + 0.3677i   0.0920 + 0.0030i   0.0353 + 0.0404i
 -0.3157 + 0.3671i   0.0418 + 0.0075i   0.0330 + 0.0409i   0.0751 + 0.0152i
  0.0918 + 0.0033i   0.0332 + 0.0408i   0.0184 + 0.0242i  -0.3085 + 0.3746i
  0.0353 + 0.0403i   0.0751 + 0.0151i  -0.3089 + 0.3743i   0.0506 + 0.0020i


s4p(:,:,94) =

  0.0274 + 0.0034i  -0.1331 + 0.4615i   0.1092 - 0.0239i   0.0491 + 0.0216i
 -0.1331 + 0.4609i   0.0373 - 0.0040i   0.0474 + 0.0230i   0.0973 - 0.0079i
  0.1093 - 0.0236i   0.0475 + 0.0230i   0.0241 + 0.0173i  -0.1243 + 0.4645i
  0.0491 + 0.0217i   0.0974 - 0.0077i  -0.1241 + 0.4640i   0.0428 - 0.0119i


s4p(:,:,95) =

  0.0211 - 0.0060i   0.0698 + 0.4691i   0.1128 - 0.0622i   0.0536 - 0.0009i
  0.0695 + 0.4691i   0.0246 - 0.0132i   0.0525 + 0.0008i   0.1063 - 0.0447i
  0.1129 - 0.0617i   0.0525 + 0.0009i   0.0205 + 0.0092i   0.0777 + 0.4689i
  0.0536 - 0.0009i   0.1066 - 0.0448i   0.0780 + 0.4689i   0.0271 - 0.0200i


s4p(:,:,96) =

  0.0062 - 0.0094i   0.2535 + 0.3931i   0.0965 - 0.1025i   0.0481 - 0.0234i
  0.2541 + 0.3934i   0.0060 - 0.0145i   0.0476 - 0.0215i   0.0944 - 0.0864i
  0.0970 - 0.1025i   0.0476 - 0.0215i   0.0094 + 0.0072i   0.2614 + 0.3911i
  0.0481 - 0.0234i   0.0947 - 0.0865i   0.2619 + 0.3914i   0.0084 - 0.0174i


s4p(:,:,97) =

 -0.0113 - 0.0008i   0.3857 + 0.2521i   0.0602 - 0.1321i   0.0329 - 0.0404i
  0.3855 + 0.2524i  -0.0130 - 0.0030i   0.0330 - 0.0381i   0.0602 - 0.1174i
  0.0609 - 0.1318i   0.0330 - 0.0381i  -0.0023 + 0.0163i   0.3943 + 0.2477i
  0.0328 - 0.0405i   0.0604 - 0.1177i   0.3942 + 0.2478i  -0.0056 - 0.0025i
```

```
s4p(:,:,98) =

  -0.0179 + 0.0214i   0.4517 + 0.0764i   0.0170 - 0.1392i   0.0150 - 0.0475i
   0.4517 + 0.0766i  -0.0188 + 0.0216i   0.0162 - 0.0450i   0.0181 - 0.1255i
   0.0174 - 0.1392i   0.0162 - 0.0452i  -0.0040 + 0.0353i   0.4592 + 0.0662i
   0.0150 - 0.0475i   0.0182 - 0.1257i   0.4587 + 0.0659i  -0.0045 + 0.0193i


s4p(:,:,99) =

  -0.0063 + 0.0387i   0.4452 - 0.1147i  -0.0198 - 0.1309i  -0.0021 - 0.0519i
   0.4453 - 0.1143i  -0.0057 + 0.0392i   0.0003 - 0.0499i  -0.0186 - 0.1175i
  -0.0195 - 0.1310i   0.0003 - 0.0501i   0.0089 + 0.0510i   0.4455 - 0.1277i
  -0.0021 - 0.0520i  -0.0186 - 0.1177i   0.4453 - 0.1284i   0.0118 + 0.0323i


s4p(:,:,100) =

   0.0092 + 0.0432i   0.3582 - 0.2881i  -0.0492 - 0.1118i  -0.0245 - 0.0503i
   0.3581 - 0.2882i   0.0115 + 0.0435i  -0.0210 - 0.0498i  -0.0474 - 0.0965i
  -0.0488 - 0.1122i  -0.0210 - 0.0498i   0.0284 + 0.0568i   0.3522 - 0.3004i
  -0.0245 - 0.0503i  -0.0473 - 0.0964i   0.3523 - 0.3003i   0.0318 + 0.0311i


s4p(:,:,101) =

   0.0225 + 0.0394i   0.2069 - 0.4094i  -0.0658 - 0.0852i  -0.0459 - 0.0367i
   0.2072 - 0.4086i   0.0274 + 0.0367i  -0.0424 - 0.0382i  -0.0605 - 0.0678i
  -0.0656 - 0.0857i  -0.0424 - 0.0381i   0.0492 + 0.0518i   0.1959 - 0.4171i
  -0.0460 - 0.0366i  -0.0606 - 0.0680i   0.1957 - 0.4171i   0.0480 + 0.0164i


s4p(:,:,102) =

   0.0325 + 0.0280i   0.0197 - 0.4572i  -0.0677 - 0.0611i  -0.0594 - 0.0138i
   0.0193 - 0.4572i   0.0347 + 0.0200i  -0.0571 - 0.0172i  -0.0586 - 0.0433i
  -0.0676 - 0.0612i  -0.0572 - 0.0170i   0.0658 + 0.0358i   0.0057 - 0.4592i
  -0.0595 - 0.0138i  -0.0585 - 0.0433i   0.0060 - 0.4592i   0.0522 - 0.0083i


s4p(:,:,103) =

   0.0329 + 0.0115i  -0.1723 - 0.4239i  -0.0622 - 0.0469i  -0.0615 + 0.0144i
```

```
  -0.1719 - 0.4235i   0.0296 + 0.0018i  -0.0613 + 0.0102i  -0.0479 - 0.0293i
  -0.0618 - 0.0469i  -0.0612 + 0.0102i   0.0723 + 0.0124i  -0.1839 - 0.4184i
  -0.0615 + 0.0143i  -0.0479 - 0.0293i  -0.1842 - 0.4189i   0.0399 - 0.0338i


s4p(:,:,104) =

   0.0227 - 0.0016i  -0.3315 - 0.3117i  -0.0583 - 0.0425i  -0.0492 + 0.0413i
  -0.3315 - 0.3116i   0.0143 - 0.0117i  -0.0513 + 0.0377i  -0.0379 - 0.0284i
  -0.0577 - 0.0426i  -0.0513 + 0.0379i   0.0657 - 0.0116i  -0.3389 - 0.3028i
  -0.0492 + 0.0413i  -0.0379 - 0.0284i  -0.3385 - 0.3033i   0.0138 - 0.0504i


s4p(:,:,105) =

   0.0083 - 0.0054i  -0.4273 - 0.1454i  -0.0609 - 0.0415i  -0.0251 + 0.0583i
  -0.4272 - 0.1456i  -0.0083 - 0.0153i  -0.0283 + 0.0568i  -0.0377 - 0.0348i
  -0.0607 - 0.0420i  -0.0283 + 0.0568i   0.0487 - 0.0287i  -0.4311 - 0.1353i
  -0.0252 + 0.0583i  -0.0379 - 0.0348i  -0.4311 - 0.1355i  -0.0188 - 0.0507i


s4p(:,:,106) =

  -0.0022 - 0.0042i  -0.4467 + 0.0429i  -0.0689 - 0.0402i   0.0015 + 0.0622i
  -0.4462 + 0.0426i  -0.0291 - 0.0053i  -0.0012 + 0.0623i  -0.0476 - 0.0388i
  -0.0690 - 0.0405i  -0.0013 + 0.0623i   0.0278 - 0.0367i  -0.4471 + 0.0538i
   0.0014 + 0.0623i  -0.0478 - 0.0388i  -0.4470 + 0.0543i  -0.0462 - 0.0350i


s4p(:,:,107) =

  -0.0139 - 0.0031i  -0.3880 + 0.2234i  -0.0844 - 0.0348i   0.0269 + 0.0565i
  -0.3881 + 0.2232i  -0.0419 + 0.0103i   0.0246 + 0.0571i  -0.0646 - 0.0362i
  -0.0844 - 0.0349i   0.0247 + 0.0571i   0.0046 - 0.0369i  -0.3826 + 0.2334i
   0.0269 + 0.0565i  -0.0644 - 0.0362i  -0.3830 + 0.2338i  -0.0620 - 0.0113i


s4p(:,:,108) =

  -0.0293 + 0.0038i  -0.2589 + 0.3627i  -0.1029 - 0.0168i   0.0490 + 0.0404i
  -0.2591 + 0.3626i  -0.0495 + 0.0281i   0.0471 + 0.0416i  -0.0844 - 0.0213i
  -0.1027 - 0.0173i   0.0471 + 0.0416i  -0.0176 - 0.0267i  -0.2499 + 0.3696i
   0.0490 + 0.0404i  -0.0842 - 0.0215i  -0.2497 + 0.3697i  -0.0668 + 0.0141i
```

```
s4p(:,:,109) =

  -0.0426 + 0.0204i  -0.0843 + 0.4340i  -0.1137 + 0.0142i   0.0622 + 0.0155i
  -0.0842 + 0.4345i  -0.0519 + 0.0487i   0.0606 + 0.0171i  -0.0972 + 0.0074i
  -0.1136 + 0.0136i   0.0606 + 0.0172i  -0.0330 - 0.0070i  -0.0743 + 0.4363i
   0.0622 + 0.0155i  -0.0976 + 0.0073i  -0.0737 + 0.4369i  -0.0619 + 0.0372i


s4p(:,:,110) =

  -0.0472 + 0.0450i   0.1016 + 0.4266i  -0.1092 + 0.0516i   0.0627 - 0.0129i
   0.1018 + 0.4266i  -0.0465 + 0.0709i   0.0615 - 0.0106i  -0.0950 + 0.0438i
  -0.1096 + 0.0511i   0.0615 - 0.0105i  -0.0373 + 0.0170i   0.1116 + 0.4251i
   0.0627 - 0.0130i  -0.0953 + 0.0441i   0.1120 + 0.4250i  -0.0498 + 0.0554i


s4p(:,:,111) =

  -0.0389 + 0.0718i   0.2662 + 0.3440i  -0.0879 + 0.0859i   0.0506 - 0.0383i
   0.2662 + 0.3447i  -0.0316 + 0.0913i   0.0502 - 0.0352i  -0.0747 + 0.0777i
  -0.0882 + 0.0856i   0.0503 - 0.0353i  -0.0294 + 0.0381i   0.2752 + 0.3395i
   0.0505 - 0.0383i  -0.0746 + 0.0783i   0.2754 + 0.3394i  -0.0330 + 0.0667i


s4p(:,:,112) =

  -0.0175 + 0.0927i   0.3819 + 0.2030i  -0.0536 + 0.1084i   0.0290 - 0.0553i
   0.3813 + 0.2031i  -0.0079 + 0.1044i   0.0304 - 0.0517i  -0.0401 + 0.0995i
  -0.0540 + 0.1082i   0.0303 - 0.0519i  -0.0139 + 0.0505i   0.3881 + 0.1954i
   0.0290 - 0.0552i  -0.0398 + 0.0995i   0.3878 + 0.1959i  -0.0141 + 0.0700i


s4p(:,:,113) =

   0.0120 + 0.1012i   0.4296 + 0.0282i  -0.0142 + 0.1147i   0.0034 - 0.0611i
   0.4291 + 0.0287i   0.0204 + 0.1056i   0.0067 - 0.0586i  -0.0002 + 0.1030i
  -0.0146 + 0.1146i   0.0066 - 0.0587i   0.0026 + 0.0521i   0.4336 + 0.0181i
   0.0034 - 0.0612i  -0.0001 + 0.1028i   0.4341 + 0.0188i   0.0023 + 0.0646i


s4p(:,:,114) =

   0.0413 + 0.0943i   0.4019 - 0.1502i   0.0220 + 0.1051i  -0.0213 - 0.0565i
```

```
   0.4020 - 0.1499i   0.0459 + 0.0929i  -0.0171 - 0.0558i   0.0344 + 0.0888i
   0.0217 + 0.1052i  -0.0171 - 0.0559i   0.0141 + 0.0444i   0.4020 - 0.1623i
  -0.0214 - 0.0566i   0.0343 + 0.0889i   0.4017 - 0.1618i   0.0112 + 0.0528i


s4p(:,:,115) =

   0.0616 + 0.0746i   0.3037 - 0.3017i   0.0481 + 0.0838i  -0.0420 - 0.0431i
   0.3039 - 0.3015i   0.0601 + 0.0689i  -0.0379 - 0.0446i   0.0555 + 0.0634i
   0.0477 + 0.0839i  -0.0380 - 0.0445i   0.0151 + 0.0327i   0.2975 - 0.3133i
  -0.0422 - 0.0432i   0.0556 + 0.0638i   0.2977 - 0.3138i   0.0096 + 0.0407i


s4p(:,:,116) =

   0.0668 + 0.0496i   0.1521 - 0.3985i   0.0600 + 0.0580i  -0.0559 - 0.0226i
   0.1521 - 0.3985i   0.0568 + 0.0436i  -0.0527 - 0.0259i   0.0598 + 0.0371i
   0.0595 + 0.0584i  -0.0528 - 0.0258i   0.0053 + 0.0257i   0.1395 - 0.4068i
  -0.0560 - 0.0225i   0.0601 + 0.0373i   0.1397 - 0.4077i  -0.0014 + 0.0359i


s4p(:,:,117) =

   0.0565 + 0.0316i  -0.0257 - 0.4258i   0.0586 + 0.0376i  -0.0609 + 0.0013i
  -0.0256 - 0.4258i   0.0413 + 0.0303i  -0.0596 - 0.0032i   0.0529 + 0.0204i
   0.0584 + 0.0379i  -0.0596 - 0.0031i  -0.0099 + 0.0311i  -0.0422 - 0.4265i
  -0.0609 + 0.0013i   0.0531 + 0.0203i  -0.0425 - 0.4264i  -0.0136 + 0.0446i


s4p(:,:,118) =

   0.0430 + 0.0284i  -0.2008 - 0.3758i   0.0530 + 0.0280i  -0.0569 + 0.0268i
  -0.2006 - 0.3754i   0.0250 + 0.0306i  -0.0578 + 0.0220i   0.0442 + 0.0149i
   0.0530 + 0.0281i  -0.0578 + 0.0220i  -0.0188 + 0.0506i  -0.2150 - 0.3673i
  -0.0567 + 0.0268i   0.0444 + 0.0148i  -0.2151 - 0.3667i  -0.0173 + 0.0640i


s4p(:,:,119) =

   0.0368 + 0.0343i  -0.3377 - 0.2558i   0.0511 + 0.0263i  -0.0410 + 0.0500i
  -0.3377 - 0.2557i   0.0154 + 0.0410i  -0.0442 + 0.0461i   0.0406 + 0.0184i
   0.0510 + 0.0264i  -0.0442 + 0.0462i  -0.0130 + 0.0764i  -0.3449 - 0.2419i
  -0.0409 + 0.0500i   0.0406 + 0.0183i  -0.3446 - 0.2414i  -0.0064 + 0.0866i
```

```
s4p(:,:,120) =

   0.0397 + 0.0410i  -0.4097 - 0.0912i   0.0558 + 0.0275i  -0.0151 + 0.0636i
  -0.4106 - 0.0916i   0.0176 + 0.0543i  -0.0199 + 0.0618i   0.0470 + 0.0246i
   0.0555 + 0.0277i  -0.0199 + 0.0617i   0.0089 + 0.0978i  -0.4097 - 0.0769i
  -0.0151 + 0.0636i   0.0469 + 0.0246i  -0.4101 - 0.0774i   0.0199 + 0.1011i


s4p(:,:,121) =

   0.0497 + 0.0431i  -0.4079 + 0.0861i   0.0682 + 0.0269i   0.0142 + 0.0636i
  -0.4083 + 0.0863i   0.0295 + 0.0614i   0.0090 + 0.0643i   0.0628 + 0.0261i
   0.0681 + 0.0274i   0.0091 + 0.0643i   0.0419 + 0.1054i  -0.4009 + 0.0974i
   0.0143 + 0.0636i   0.0628 + 0.0262i  -0.4018 + 0.0977i   0.0535 + 0.0985i


s4p(:,:,122) =

   0.0623 + 0.0347i  -0.3331 + 0.2459i   0.0871 + 0.0181i   0.0397 + 0.0503i
  -0.3336 + 0.2460i   0.0444 + 0.0567i   0.0355 + 0.0536i   0.0841 + 0.0164i
   0.0872 + 0.0185i   0.0355 + 0.0536i   0.0753 + 0.0945i  -0.3226 + 0.2519i
   0.0398 + 0.0504i   0.0842 + 0.0163i  -0.3234 + 0.2516i   0.0823 + 0.0761i


s4p(:,:,123) =

   0.0677 + 0.0165i  -0.1994 + 0.3585i   0.1051 - 0.0038i   0.0561 + 0.0281i
  -0.1994 + 0.3590i   0.0519 + 0.0413i   0.0541 + 0.0331i   0.1021 - 0.0073i
   0.1050 - 0.0037i   0.0542 + 0.0330i   0.0978 + 0.0682i  -0.1900 + 0.3588i
   0.0562 + 0.0280i   0.1021 - 0.0073i  -0.1898 + 0.3599i   0.0944 + 0.0403i


s4p(:,:,124) =

   0.0602 - 0.0032i  -0.0328 + 0.4051i   0.1132 - 0.0374i   0.0610 + 0.0022i
  -0.0324 + 0.4061i   0.0469 + 0.0232i   0.0620 + 0.0072i   0.1084 - 0.0413i
   0.1132 - 0.0368i   0.0619 + 0.0071i   0.1023 + 0.0369i  -0.0252 + 0.4024i
   0.0611 + 0.0022i   0.1086 - 0.0413i  -0.0250 + 0.4033i   0.0845 + 0.0038i


s4p(:,:,125) =

   0.0422 - 0.0152i   0.1359 + 0.3791i   0.1055 - 0.0752i   0.0551 - 0.0220i
```

```
   0.1362 + 0.3790i   0.0296 + 0.0112i   0.0580 - 0.0184i   0.0981 - 0.0780i
   0.1059 - 0.0744i   0.0581 - 0.0185i   0.0913 + 0.0123i   0.1408 + 0.3753i
   0.0552 - 0.0221i   0.0983 - 0.0782i   0.1415 + 0.3756i   0.0564 - 0.0197i


s4p(:,:,126) =

   0.0211 - 0.0136i   0.2756 + 0.2867i   0.0808 - 0.1074i   0.0410 - 0.0408i
   0.2756 + 0.2866i   0.0059 + 0.0135i   0.0445 - 0.0391i   0.0711 - 0.1074i
   0.0815 - 0.1073i   0.0445 - 0.0390i   0.0733 + 0.0021i   0.2806 + 0.2837i
   0.0409 - 0.0409i   0.0711 - 0.1076i   0.2809 + 0.2834i   0.0220 - 0.0212i


s4p(:,:,127) =

   0.0088 + 0.0013i   0.3652 + 0.1491i   0.0458 - 0.1257i   0.0225 - 0.0518i
   0.3657 + 0.1491i  -0.0117 + 0.0335i   0.0259 - 0.0515i   0.0348 - 0.1204i
   0.0460 - 0.1255i   0.0259 - 0.0514i   0.0601 + 0.0062i   0.3709 + 0.1434i
   0.0225 - 0.0519i   0.0347 - 0.1206i   0.3710 + 0.1432i  -0.0023 - 0.0017i


s4p(:,:,128) =

   0.0137 + 0.0167i   0.3942 - 0.0129i   0.0100 - 0.1287i   0.0031 - 0.0576i
   0.3947 - 0.0126i  -0.0109 + 0.0607i   0.0065 - 0.0582i   0.0007 - 0.1174i
   0.0106 - 0.1285i   0.0065 - 0.0581i   0.0601 + 0.0139i   0.3966 - 0.0217i
   0.0031 - 0.0577i   0.0006 - 0.1174i   0.3971 - 0.0218i  -0.0052 + 0.0258i


s4p(:,:,129) =

   0.0272 + 0.0183i   0.3532 - 0.1747i  -0.0218 - 0.1209i  -0.0204 - 0.0582i
   0.3536 - 0.1749i   0.0052 + 0.0812i  -0.0169 - 0.0597i  -0.0261 - 0.1029i
  -0.0212 - 0.1209i  -0.0170 - 0.0597i   0.0695 + 0.0152i   0.3513 - 0.1844i
  -0.0204 - 0.0582i  -0.0261 - 0.1030i   0.3513 - 0.1853i   0.0112 + 0.0456i


s4p(:,:,130) =

   0.0361 + 0.0070i   0.2476 - 0.3042i  -0.0465 - 0.1038i  -0.0461 - 0.0471i
   0.2479 - 0.3045i   0.0303 + 0.0910i  -0.0428 - 0.0505i  -0.0410 - 0.0816i
  -0.0459 - 0.1041i  -0.0428 - 0.0506i   0.0814 + 0.0063i   0.2409 - 0.3118i
  -0.0462 - 0.0472i  -0.0410 - 0.0819i   0.2411 - 0.3127i   0.0384 + 0.0490i
```

```
s4p(:,:,131) =

   0.0340 - 0.0097i   0.0991 - 0.3757i  -0.0605 - 0.0824i  -0.0649 - 0.0224i
   0.0992 - 0.3757i   0.0589 + 0.0863i  -0.0633 - 0.0282i  -0.0424 - 0.0626i
  -0.0601 - 0.0825i  -0.0634 - 0.0281i   0.0895 - 0.0128i   0.0897 - 0.3804i
  -0.0648 - 0.0223i  -0.0424 - 0.0626i   0.0895 - 0.3809i   0.0641 + 0.0312i


s4p(:,:,132) =

   0.0214 - 0.0253i  -0.0627 - 0.3812i  -0.0634 - 0.0638i  -0.0688 + 0.0090i
  -0.0630 - 0.3812i   0.0819 + 0.0647i  -0.0711 + 0.0024i  -0.0368 - 0.0539i
  -0.0630 - 0.0640i  -0.0711 + 0.0025i   0.0878 - 0.0391i  -0.0745 - 0.3805i
  -0.0689 + 0.0091i  -0.0369 - 0.0540i  -0.0752 - 0.3808i   0.0742 - 0.0032i


s4p(:,:,133) =

  -0.0031 - 0.0352i  -0.2131 - 0.3202i  -0.0627 - 0.0538i  -0.0580 + 0.0386i
  -0.2130 - 0.3202i   0.0886 + 0.0345i  -0.0642 + 0.0339i  -0.0342 - 0.0551i
  -0.0623 - 0.0541i  -0.0642 + 0.0338i   0.0723 - 0.0665i  -0.2227 - 0.3141i
  -0.0579 + 0.0387i  -0.0343 - 0.0551i  -0.2235 - 0.3141i   0.0604 - 0.0401i


s4p(:,:,134) =

  -0.0341 - 0.0289i  -0.3243 - 0.2043i  -0.0650 - 0.0487i  -0.0349 + 0.0592i
  -0.3242 - 0.2036i   0.0795 + 0.0070i  -0.0432 + 0.0589i  -0.0398 - 0.0602i
  -0.0647 - 0.0490i  -0.0432 + 0.0589i   0.0443 - 0.0864i  -0.3302 - 0.1938i
  -0.0348 + 0.0594i  -0.0397 - 0.0602i  -0.3300 - 0.1940i   0.0265 - 0.0646i


s4p(:,:,135) =

  -0.0589 - 0.0076i  -0.3792 - 0.0523i  -0.0715 - 0.0453i  -0.0068 + 0.0674i
  -0.3792 - 0.0522i   0.0613 - 0.0115i  -0.0141 + 0.0718i  -0.0528 - 0.0624i
  -0.0712 - 0.0454i  -0.0142 + 0.0719i   0.0100 - 0.0944i  -0.3797 - 0.0412i
  -0.0067 + 0.0674i  -0.0528 - 0.0624i  -0.3797 - 0.0410i  -0.0160 - 0.0685i


s4p(:,:,136) =

  -0.0731 + 0.0218i  -0.3664 + 0.1093i  -0.0844 - 0.0392i   0.0217 + 0.0639i
```

```
  -0.3659 + 0.1097i   0.0389 - 0.0224i   0.0177 + 0.0722i  -0.0715 - 0.0587i
  -0.0840 - 0.0396i   0.0178 + 0.0722i  -0.0270 - 0.0905i  -0.3613 + 0.1185i
   0.0217 + 0.0638i  -0.0715 - 0.0588i  -0.3612 + 0.1187i  -0.0562 - 0.0521i


s4p(:,:,137) =

  -0.0761 + 0.0544i  -0.2865 + 0.2506i  -0.1002 - 0.0236i   0.0463 + 0.0487i
  -0.2864 + 0.2507i   0.0124 - 0.0249i   0.0479 + 0.0584i  -0.0923 - 0.0444i
  -0.0999 - 0.0243i   0.0480 + 0.0584i  -0.0614 - 0.0722i  -0.2782 + 0.2559i
   0.0463 + 0.0487i  -0.0925 - 0.0443i  -0.2783 + 0.2558i  -0.0855 - 0.0191i


s4p(:,:,138) =

  -0.0681 + 0.0861i  -0.1556 + 0.3445i  -0.1106 + 0.0027i   0.0620 + 0.0248i
  -0.1556 + 0.3445i  -0.0153 - 0.0160i   0.0693 + 0.0317i  -0.1082 - 0.0178i
  -0.1109 + 0.0019i   0.0693 + 0.0317i  -0.0867 - 0.0419i  -0.1474 + 0.3457i
   0.0620 + 0.0248i  -0.1081 - 0.0178i  -0.1470 + 0.3459i  -0.0973 + 0.0232i


s4p(:,:,139) =

  -0.0502 + 0.1136i   0.0005 + 0.3750i  -0.1093 + 0.0352i   0.0660 - 0.0029i
   0.0004 + 0.3754i  -0.0382 + 0.0052i   0.0762 - 0.0020i  -0.1111 + 0.0168i
  -0.1094 + 0.0348i   0.0761 - 0.0020i  -0.0984 - 0.0049i   0.0073 + 0.3736i
   0.0661 - 0.0031i  -0.1108 + 0.0165i   0.0079 + 0.3736i  -0.0899 + 0.0650i


s4p(:,:,140) =

  -0.0242 + 0.1338i   0.1539 + 0.3396i  -0.0934 + 0.0661i   0.0587 - 0.0292i
   0.1541 + 0.3395i  -0.0500 + 0.0358i   0.0669 - 0.0346i  -0.0977 + 0.0508i
  -0.0934 + 0.0657i   0.0668 - 0.0346i  -0.0949 + 0.0323i   0.1595 + 0.3365i
   0.0586 - 0.0293i  -0.0977 + 0.0504i   0.1595 + 0.3360i  -0.0663 + 0.0971i


s4p(:,:,141) =

   0.0072 + 0.1437i   0.2788 + 0.2442i  -0.0658 + 0.0877i   0.0420 - 0.0500i
   0.2788 + 0.2443i  -0.0466 + 0.0683i   0.0450 - 0.0589i  -0.0710 + 0.0750i
  -0.0662 + 0.0870i   0.0450 - 0.0588i  -0.0787 + 0.0623i   0.2826 + 0.2405i
   0.0419 - 0.0500i  -0.0714 + 0.0750i   0.2830 + 0.2401i  -0.0344 + 0.1135i
```

```
s4p(:,:,142) =

   0.0395 + 0.1410i   0.3540 + 0.1069i  -0.0336 + 0.0947i   0.0190 - 0.0630i
   0.3535 + 0.1071i  -0.0295 + 0.0941i   0.0162 - 0.0711i  -0.0386 + 0.0844i
  -0.0343 + 0.0944i   0.0162 - 0.0711i  -0.0549 + 0.0802i   0.3562 + 0.1027i
   0.0189 - 0.0631i  -0.0388 + 0.0845i   0.3561 + 0.1029i  -0.0036 + 0.1132i


s4p(:,:,143) =

   0.0667 + 0.1260i   0.3645 - 0.0486i  -0.0056 + 0.0886i  -0.0077 - 0.0666i
   0.3640 - 0.0487i  -0.0053 + 0.1075i  -0.0142 - 0.0710i  -0.0096 + 0.0788i
  -0.0061 + 0.0884i  -0.0143 - 0.0710i  -0.0319 + 0.0844i   0.3667 - 0.0541i
  -0.0078 - 0.0665i  -0.0095 + 0.0791i   0.3668 - 0.0538i   0.0179 + 0.1000i


s4p(:,:,144) =

   0.0830 + 0.1024i   0.3099 - 0.1947i   0.0128 + 0.0744i  -0.0350 - 0.0597i
   0.3099 - 0.1948i   0.0174 + 0.1076i  -0.0419 - 0.0593i   0.0092 + 0.0641i
   0.0125 + 0.0743i  -0.0419 - 0.0593i  -0.0172 + 0.0793i   0.3104 - 0.2018i
  -0.0350 - 0.0597i   0.0093 + 0.0640i   0.3105 - 0.2017i   0.0255 + 0.0821i


s4p(:,:,145) =

   0.0845 + 0.0773i   0.1995 - 0.3049i   0.0200 + 0.0596i  -0.0590 - 0.0412i
   0.1987 - 0.3044i   0.0303 + 0.0990i  -0.0637 - 0.0374i   0.0149 + 0.0479i
   0.0196 + 0.0596i  -0.0638 - 0.0373i  -0.0145 + 0.0729i   0.1961 - 0.3121i
  -0.0591 - 0.0411i   0.0150 + 0.0479i   0.1964 - 0.3119i   0.0184 + 0.0694i


s4p(:,:,146) =

   0.0726 + 0.0602i   0.0540 - 0.3565i   0.0196 + 0.0508i  -0.0734 - 0.0117i
   0.0538 - 0.3557i   0.0306 + 0.0935i  -0.0749 - 0.0066i   0.0109 + 0.0403i
   0.0194 + 0.0509i  -0.0749 - 0.0065i  -0.0208 + 0.0743i   0.0468 - 0.3622i
  -0.0735 - 0.0117i   0.0109 + 0.0403i   0.0474 - 0.3625i   0.0047 + 0.0717i


s4p(:,:,147) =

   0.0540 + 0.0587i  -0.0970 - 0.3430i   0.0175 + 0.0512i  -0.0724 + 0.0222i
```

```
  -0.0969 - 0.3426i   0.0288 + 0.1015i  -0.0709 + 0.0266i   0.0081 + 0.0451i
   0.0173 + 0.0512i  -0.0709 + 0.0267i  -0.0275 + 0.0893i  -0.1071 - 0.3443i
  -0.0725 + 0.0223i   0.0082 + 0.0453i  -0.1069 - 0.3448i  -0.0019 + 0.0904i


s4p(:,:,148) =

   0.0427 + 0.0770i  -0.2281 - 0.2696i   0.0224 + 0.0598i  -0.0560 + 0.0528i
  -0.2281 - 0.2696i   0.0385 + 0.1179i  -0.0528 + 0.0550i   0.0162 + 0.0555i
   0.0221 + 0.0600i  -0.0527 + 0.0550i  -0.0227 + 0.1156i  -0.2371 - 0.2656i
  -0.0560 + 0.0527i   0.0161 + 0.0556i  -0.2370 - 0.2657i   0.0097 + 0.1159i


s4p(:,:,149) =

   0.0530 + 0.1030i  -0.3153 - 0.1489i   0.0396 + 0.0677i  -0.0265 + 0.0730i
  -0.3154 - 0.1488i   0.0625 + 0.1320i  -0.0233 + 0.0731i   0.0348 + 0.0619i
   0.0392 + 0.0679i  -0.0232 + 0.0730i   0.0010 + 0.1417i  -0.3206 - 0.1422i
  -0.0265 + 0.0730i   0.0348 + 0.0620i  -0.3210 - 0.1424i   0.0401 + 0.1341i


s4p(:,:,150) =

   0.0842 + 0.1194i  -0.3435 - 0.0055i   0.0648 + 0.0653i   0.0097 + 0.0764i
  -0.3431 - 0.0057i   0.0983 + 0.1346i   0.0120 + 0.0753i   0.0596 + 0.0585i
   0.0645 + 0.0657i   0.0120 + 0.0752i   0.0395 + 0.1546i  -0.3455 + 0.0010i
   0.0097 + 0.0766i   0.0596 + 0.0584i  -0.3459 + 0.0006i   0.0819 + 0.1346i


s4p(:,:,151) =

   0.1248 + 0.1158i  -0.3107 + 0.1322i   0.0899 + 0.0505i   0.0419 + 0.0622i
  -0.3107 + 0.1323i   0.1393 + 0.1187i   0.0432 + 0.0603i   0.0842 + 0.0427i
   0.0897 + 0.0509i   0.0432 + 0.0604i   0.0829 + 0.1480i  -0.3122 + 0.1385i
   0.0420 + 0.0621i   0.0842 + 0.0427i  -0.3122 + 0.1386i   0.1233 + 0.1131i


s4p(:,:,152) =

   0.1636 + 0.0903i  -0.2288 + 0.2431i   0.1099 + 0.0249i   0.0622 + 0.0359i
  -0.2286 + 0.2432i   0.1730 + 0.0814i   0.0626 + 0.0342i   0.1021 + 0.0157i
   0.1096 + 0.0254i   0.0627 + 0.0342i   0.1203 + 0.1218i  -0.2284 + 0.2508i
   0.0623 + 0.0358i   0.1022 + 0.0158i  -0.2286 + 0.2512i   0.1512 + 0.0725i
```

```
s4p(:,:,153) =

   0.1873 + 0.0456i  -0.1109 + 0.3124i   0.1190 - 0.0092i   0.0685 + 0.0059i
  -0.1106 + 0.3125i   0.1872 + 0.0291i   0.0683 + 0.0042i   0.1082 - 0.0181i
   0.1191 - 0.0087i   0.0682 + 0.0042i   0.1424 + 0.0815i  -0.1080 + 0.3207i
   0.0684 + 0.0060i   0.1083 - 0.0180i  -0.1081 + 0.3211i   0.1576 + 0.0229i


s4p(:,:,154) =

   0.1867 - 0.0065i   0.0243 + 0.3291i   0.1137 - 0.0456i   0.0619 - 0.0218i
   0.0241 + 0.3291i   0.1762 - 0.0257i   0.0610 - 0.0233i   0.1004 - 0.0524i
   0.1139 - 0.0450i   0.0610 - 0.0233i   0.1432 + 0.0377i   0.0309 + 0.3378i
   0.0619 - 0.0217i   0.1003 - 0.0525i   0.0315 + 0.3378i   0.1403 - 0.0224i


s4p(:,:,155) =

   0.1613 - 0.0513i   0.1541 + 0.2909i   0.0943 - 0.0763i   0.0459 - 0.0424i
   0.1541 + 0.2914i   0.1411 - 0.0686i   0.0442 - 0.0436i   0.0791 - 0.0805i
   0.0945 - 0.0757i   0.0443 - 0.0438i   0.1254 + 0.0026i   0.1668 + 0.2958i
   0.0460 - 0.0425i   0.0793 - 0.0807i   0.1669 + 0.2962i   0.1066 - 0.0516i


s4p(:,:,156) =

   0.1213 - 0.0758i   0.2587 + 0.2055i   0.0662 - 0.0951i   0.0254 - 0.0542i
   0.2593 + 0.2053i   0.0931 - 0.0882i   0.0231 - 0.0544i   0.0496 - 0.0959i
   0.0668 - 0.0944i   0.0231 - 0.0544i   0.0984 - 0.0156i   0.2747 + 0.2030i
   0.0254 - 0.0543i   0.0496 - 0.0958i   0.2747 + 0.2030i   0.0685 - 0.0589i


s4p(:,:,157) =

   0.0825 - 0.0781i   0.3230 + 0.0829i   0.0377 - 0.1003i   0.0047 - 0.0585i
   0.3233 + 0.0830i   0.0494 - 0.0831i   0.0027 - 0.0574i   0.0211 - 0.0967i
   0.0384 - 0.1000i   0.0027 - 0.0574i   0.0741 - 0.0173i   0.3360 + 0.0715i
   0.0047 - 0.0585i   0.0210 - 0.0966i   0.3361 + 0.0711i   0.0400 - 0.0483i


s4p(:,:,158) =

   0.0544 - 0.0681i   0.3316 - 0.0598i   0.0143 - 0.0961i  -0.0167 - 0.0583i
```

```
   0.3316 - 0.0595i   0.0205 - 0.0642i  -0.0176 - 0.0561i  -0.0002 - 0.0881i
   0.0149 - 0.0960i  -0.0176 - 0.0561i   0.0604 - 0.0099i   0.3361 - 0.0766i
  -0.0167 - 0.0582i  -0.0001 - 0.0879i   0.3362 - 0.0763i   0.0279 - 0.0320i


s4p(:,:,159) =

   0.0376 - 0.0537i   0.2765 - 0.1945i  -0.0015 - 0.0866i  -0.0400 - 0.0502i
   0.2765 - 0.1946i   0.0077 - 0.0428i  -0.0397 - 0.0481i  -0.0119 - 0.0756i
  -0.0010 - 0.0866i  -0.0397 - 0.0481i   0.0584 - 0.0020i   0.2715 - 0.2105i
  -0.0401 - 0.0502i  -0.0117 - 0.0757i   0.2717 - 0.2110i   0.0296 - 0.0215i


s4p(:,:,160) =

   0.0302 - 0.0414i   0.1680 - 0.2924i  -0.0089 - 0.0774i  -0.0608 - 0.0304i
   0.1674 - 0.2915i   0.0080 - 0.0281i  -0.0597 - 0.0292i  -0.0138 - 0.0658i
  -0.0085 - 0.0772i  -0.0597 - 0.0292i   0.0646 - 0.0018i   0.1561 - 0.3034i
  -0.0607 - 0.0304i  -0.0136 - 0.0660i   0.1561 - 0.3034i   0.0368 - 0.0243i


s4p(:,:,161) =

   0.0278 - 0.0343i   0.0292 - 0.3318i  -0.0111 - 0.0729i  -0.0705 - 0.0004i
   0.0290 - 0.3318i   0.0123 - 0.0271i  -0.0696 + 0.0002i  -0.0111 - 0.0648i
  -0.0106 - 0.0728i  -0.0695 + 0.0002i   0.0709 - 0.0125i   0.0145 - 0.3366i
  -0.0705 - 0.0003i  -0.0109 - 0.0652i   0.0145 - 0.3370i   0.0375 - 0.0408i


s4p(:,:,162) =

   0.0268 - 0.0332i  -0.1085 - 0.3100i  -0.0127 - 0.0751i  -0.0637 + 0.0314i
  -0.1086 - 0.3096i   0.0076 - 0.0385i  -0.0627 + 0.0319i  -0.0131 - 0.0738i
  -0.0122 - 0.0751i  -0.0627 + 0.0319i   0.0698 - 0.0317i  -0.1245 - 0.3081i
  -0.0637 + 0.0314i  -0.0133 - 0.0743i  -0.1245 - 0.3085i   0.0223 - 0.0632i


s4p(:,:,163) =

   0.0223 - 0.0387i  -0.2242 - 0.2370i  -0.0198 - 0.0836i  -0.0434 + 0.0554i
  -0.2237 - 0.2364i  -0.0141 - 0.0491i  -0.0419 + 0.0560i  -0.0267 - 0.0846i
  -0.0190 - 0.0837i  -0.0419 + 0.0560i   0.0569 - 0.0537i  -0.2376 - 0.2274i
  -0.0434 + 0.0555i  -0.0270 - 0.0850i  -0.2374 - 0.2275i  -0.0100 - 0.0785i
```

```
s4p(:,:,164) =

   0.0093 - 0.0458i  -0.3001 - 0.1231i  -0.0366 - 0.0926i  -0.0157 + 0.0679i
  -0.2999 - 0.1235i  -0.0455 - 0.0466i  -0.0138 + 0.0675i  -0.0492 - 0.0888i
  -0.0361 - 0.0928i  -0.0138 + 0.0676i   0.0315 - 0.0717i  -0.3069 - 0.1095i
  -0.0157 + 0.0680i  -0.0497 - 0.0887i  -0.3065 - 0.1093i  -0.0508 - 0.0775i


s4p(:,:,165) =

  -0.0128 - 0.0486i  -0.3230 + 0.0114i  -0.0637 - 0.0946i   0.0142 + 0.0682i
  -0.3226 + 0.0115i  -0.0765 - 0.0295i   0.0159 + 0.0666i  -0.0760 - 0.0825i
  -0.0631 - 0.0948i   0.0158 + 0.0666i  -0.0043 - 0.0794i  -0.3218 + 0.0262i
   0.0143 + 0.0682i  -0.0762 - 0.0821i  -0.3218 + 0.0260i  -0.0910 - 0.0590i


s4p(:,:,166) =

  -0.0407 - 0.0399i  -0.2874 + 0.1429i  -0.0953 - 0.0820i   0.0418 + 0.0555i
  -0.2875 + 0.1427i  -0.1013 - 0.0007i   0.0424 + 0.0533i  -0.1018 - 0.0640i
  -0.0948 - 0.0824i   0.0424 + 0.0532i  -0.0448 - 0.0710i  -0.2806 + 0.1543i
   0.0418 + 0.0555i  -0.1018 - 0.0637i  -0.2802 + 0.1543i  -0.1227 - 0.0249i


s4p(:,:,167) =

  -0.0668 - 0.0170i  -0.2014 + 0.2471i  -0.1212 - 0.0531i   0.0609 + 0.0320i
  -0.2017 + 0.2469i  -0.1159 + 0.0363i   0.0604 + 0.0298i  -0.1197 - 0.0340i
  -0.1207 - 0.0539i   0.0604 + 0.0298i  -0.0809 - 0.0451i  -0.1915 + 0.2534i
   0.0610 + 0.0319i  -0.1198 - 0.0339i  -0.1913 + 0.2531i  -0.1401 + 0.0190i


s4p(:,:,168) =

  -0.0829 + 0.0186i  -0.0814 + 0.3056i  -0.1323 - 0.0137i   0.0675 + 0.0031i
  -0.0812 + 0.3056i  -0.1187 + 0.0772i   0.0660 + 0.0013i  -0.1247 + 0.0027i
  -0.1324 - 0.0145i   0.0660 + 0.0013i  -0.1041 - 0.0053i  -0.0710 + 0.3078i
   0.0673 + 0.0030i  -0.1246 + 0.0024i  -0.0712 + 0.3074i  -0.1405 + 0.0661i


s4p(:,:,169) =

  -0.0829 + 0.0607i   0.0507 + 0.3099i  -0.1257 + 0.0267i   0.0611 - 0.0246i
```

```
   0.0506 + 0.3096i  -0.1088 + 0.1180i   0.0591 - 0.0257i  -0.1142 + 0.0384i
  -0.1257 + 0.0260i   0.0591 - 0.0257i  -0.1092 + 0.0401i   0.0600 + 0.3090i
   0.0610 - 0.0246i  -0.1141 + 0.0384i   0.0599 + 0.3087i  -0.1250 + 0.1086i


s4p(:,:,170) =

  -0.0651 + 0.1009i   0.1721 + 0.2614i  -0.1035 + 0.0590i   0.0448 - 0.0459i
   0.1715 + 0.2609i  -0.0861 + 0.1542i   0.0423 - 0.0461i  -0.0907 + 0.0659i
  -0.1038 + 0.0582i   0.0424 - 0.0462i  -0.0953 + 0.0818i   0.1806 + 0.2574i
   0.0448 - 0.0460i  -0.0907 + 0.0658i   0.1807 + 0.2573i  -0.0975 + 0.1412i


s4p(:,:,171) =

  -0.0313 + 0.1301i   0.2632 + 0.1673i  -0.0727 + 0.0768i   0.0227 - 0.0588i
   0.2632 + 0.1674i  -0.0541 + 0.1809i   0.0202 - 0.0580i  -0.0602 + 0.0800i
  -0.0733 + 0.0761i   0.0203 - 0.0579i  -0.0674 + 0.1110i   0.2708 + 0.1605i
   0.0227 - 0.0588i  -0.0604 + 0.0800i   0.2708 + 0.1604i  -0.0640 + 0.1601i


s4p(:,:,172) =

   0.0112 + 0.1406i   0.3081 + 0.0436i  -0.0430 + 0.0792i  -0.0015 - 0.0627i
   0.3081 + 0.0439i  -0.0172 + 0.1951i  -0.0036 - 0.0609i  -0.0311 + 0.0799i
  -0.0434 + 0.0791i  -0.0036 - 0.0608i  -0.0348 + 0.1223i   0.3134 + 0.0335i
  -0.0016 - 0.0627i  -0.0312 + 0.0797i   0.3133 + 0.0339i  -0.0318 + 0.1648i


s4p(:,:,173) =

   0.0504 + 0.1302i   0.2983 - 0.0873i  -0.0212 + 0.0710i  -0.0258 - 0.0580i
   0.2977 - 0.0869i   0.0190 + 0.1975i  -0.0271 - 0.0553i  -0.0101 + 0.0695i
  -0.0214 + 0.0709i  -0.0271 - 0.0551i  -0.0086 + 0.1176i   0.2987 - 0.1003i
  -0.0259 - 0.0580i  -0.0102 + 0.0695i   0.2985 - 0.0998i  -0.0081 + 0.1596i


s4p(:,:,174) =

   0.0752 + 0.1044i   0.2342 - 0.2027i  -0.0105 + 0.0595i  -0.0480 - 0.0447i
   0.2341 - 0.2023i   0.0484 + 0.1896i  -0.0478 - 0.0412i  -0.0002 + 0.0560i
  -0.0108 + 0.0594i  -0.0479 - 0.0412i   0.0026 + 0.1049i   0.2285 - 0.2160i
  -0.0480 - 0.0447i  -0.0003 + 0.0560i   0.2281 - 0.2159i   0.0036 + 0.1522i
```

```
s4p(:,:,175) =

   0.0793 + 0.0744i   0.1272 - 0.2801i  -0.0092 + 0.0526i  -0.0649 - 0.0218i
   0.1276 - 0.2799i   0.0675 + 0.1782i  -0.0631 - 0.0184i  -0.0003 + 0.0470i
  -0.0094 + 0.0525i  -0.0630 - 0.0183i  -0.0016 + 0.0978i   0.1144 - 0.2894i
  -0.0649 - 0.0218i  -0.0003 + 0.0470i   0.1146 - 0.2893i   0.0054 + 0.1516i


s4p(:,:,176) =

   0.0649 + 0.0540i  -0.0005 - 0.3048i  -0.0103 + 0.0547i  -0.0705 + 0.0088i
  -0.0005 - 0.3041i   0.0785 + 0.1710i  -0.0668 + 0.0114i  -0.0034 + 0.0484i
  -0.0107 + 0.0546i  -0.0668 + 0.0114i  -0.0127 + 0.1064i  -0.0177 - 0.3057i
  -0.0705 + 0.0089i  -0.0035 + 0.0485i  -0.0175 - 0.3057i   0.0066 + 0.1647i


s4p(:,:,177) =

   0.0424 + 0.0530i  -0.1246 - 0.2740i  -0.0062 + 0.0647i  -0.0603 + 0.0401i
  -0.1239 - 0.2743i   0.0908 + 0.1731i  -0.0552 + 0.0404i   0.0008 + 0.0585i
  -0.0066 + 0.0647i  -0.0552 + 0.0404i  -0.0161 + 0.1324i  -0.1401 - 0.2656i
  -0.0603 + 0.0402i   0.0006 + 0.0587i  -0.1397 - 0.2650i   0.0197 + 0.1889i


s4p(:,:,178) =

   0.0284 + 0.0728i  -0.2236 - 0.1968i   0.0089 + 0.0768i  -0.0360 + 0.0633i
  -0.2234 - 0.1970i   0.1121 + 0.1760i  -0.0313 + 0.0603i   0.0165 + 0.0680i
   0.0085 + 0.0768i  -0.0312 + 0.0602i   0.0003 + 0.1667i  -0.2313 - 0.1806i
  -0.0360 + 0.0633i   0.0163 + 0.0681i  -0.2314 - 0.1805i   0.0514 + 0.2114i


s4p(:,:,179) =

   0.0356 + 0.1003i  -0.2817 - 0.0869i   0.0345 + 0.0813i  -0.0037 + 0.0720i
  -0.2822 - 0.0875i   0.1411 + 0.1728i  -0.0013 + 0.0668i   0.0399 + 0.0689i
   0.0340 + 0.0813i  -0.0012 + 0.0667i   0.0389 + 0.1941i  -0.2787 - 0.0703i
  -0.0037 + 0.0720i   0.0399 + 0.0691i  -0.2788 - 0.0700i   0.0982 + 0.2198i


s4p(:,:,180) =

   0.0632 + 0.1186i  -0.2903 + 0.0351i   0.0632 + 0.0724i   0.0283 + 0.0644i
```

```
  -0.2900 + 0.0350i   0.1731 + 0.1581i   0.0273 + 0.0587i   0.0645 + 0.0584i
   0.0626 + 0.0727i   0.0274 + 0.0587i   0.0926 + 0.2013i  -0.2788 + 0.0453i
   0.0283 + 0.0644i   0.0644 + 0.0586i  -0.2788 + 0.0453i   0.1501 + 0.2072i


s4p(:,:,181) =

   0.1005 + 0.1167i  -0.2488 + 0.1485i   0.0872 + 0.0512i   0.0515 + 0.0436i
  -0.2488 + 0.1478i   0.2015 + 0.1307i   0.0480 + 0.0395i   0.0845 + 0.0378i
   0.0867 + 0.0520i   0.0479 + 0.0395i   0.1474 + 0.1830i  -0.2359 + 0.1501i
   0.0516 + 0.0435i   0.0845 + 0.0380i  -0.2357 + 0.1498i   0.1959 + 0.1737i


s4p(:,:,182) =

   0.1325 + 0.0926i  -0.1667 + 0.2357i   0.1017 + 0.0219i   0.0618 + 0.0167i
  -0.1678 + 0.2350i   0.2193 + 0.0923i   0.0569 + 0.0151i   0.0960 + 0.0099i
   0.1014 + 0.0227i   0.0570 + 0.0152i   0.1900 + 0.1415i  -0.1576 + 0.2301i
   0.0617 + 0.0166i   0.0962 + 0.0101i  -0.1577 + 0.2301i   0.2256 + 0.1232i


s4p(:,:,183) =

   0.1460 + 0.0539i  -0.0573 + 0.2830i   0.1038 - 0.0105i   0.0596 - 0.0092i
  -0.0580 + 0.2825i   0.2214 + 0.0489i   0.0553 - 0.0084i   0.0962 - 0.0209i
   0.1036 - 0.0099i   0.0554 - 0.0084i   0.2099 + 0.0865i  -0.0541 + 0.2750i
   0.0597 - 0.0090i   0.0966 - 0.0208i  -0.0541 + 0.2746i   0.2326 + 0.0652i


s4p(:,:,184) =

   0.1357 + 0.0157i   0.0632 + 0.2817i   0.0938 - 0.0402i   0.0489 - 0.0296i
   0.0632 + 0.2828i   0.2063 + 0.0107i   0.0458 - 0.0275i   0.0849 - 0.0489i
   0.0938 - 0.0393i   0.0458 - 0.0275i   0.2036 + 0.0317i   0.0609 + 0.2762i
   0.0491 - 0.0297i   0.0852 - 0.0494i   0.0609 + 0.2768i   0.2164 + 0.0126i


s4p(:,:,185) =

   0.1077 - 0.0063i   0.1739 + 0.2322i   0.0751 - 0.0617i   0.0336 - 0.0437i
```

16-Port S-Parameters to 4-Port S-Parameters Using Two Impedances

Convert 16-port S-parameters to 4-port S-parameters by using ports `1`, `16`, `2`, and `15` as the first, second, third, and fourth ports terminate port `4` with an impedance of `100` ohms and terminate the remaining `11` ports with an impedance of `50` ohms.

```
ckt = read(rfckt.passive,'default.s16p');
s16p = ckt.NetworkData.Data;
Z0 = ckt.NetworkData.Z0;
ZT = {};
ZT(1:16) = {50};
ZT{4} = 100;
s4p = snp2smp(s16p,Z0,[1 16 2 15],ZT)
```

```
s4p =
s4p(:,:,1) =

   0.0857 - 0.1168i  -0.5372 - 0.6804i   0.0966 - 0.0706i   0.0067 + 0.0053i
  -0.5366 - 0.6860i   0.0803 - 0.1234i   0.0059 + 0.0048i   0.0977 - 0.0703i
   0.0957 - 0.0700i   0.0067 + 0.0048i   0.0818 - 0.1104i  -0.5362 - 0.6838i
   0.0055 + 0.0051i   0.0972 - 0.0703i  -0.5376 - 0.6840i   0.0761 - 0.1180i


s4p(:,:,2) =

   0.0479 - 0.1334i  -0.7665 - 0.3900i   0.0586 - 0.1042i   0.0071 - 0.0003i
  -0.7674 - 0.3903i   0.0365 - 0.1395i   0.0070 - 0.0004i   0.0602 - 0.1034i
   0.0597 - 0.1028i   0.0062 + 0.0001i   0.0428 - 0.1282i  -0.7686 - 0.3880i
   0.0068 - 0.0001i   0.0607 - 0.1033i  -0.7682 - 0.3889i   0.0348 - 0.1310i


s4p(:,:,3) =

   0.0031 - 0.1361i  -0.8526 - 0.0298i   0.0118 - 0.1094i   0.0044 - 0.0045i
  -0.8535 - 0.0309i  -0.0084 - 0.1364i   0.0043 - 0.0041i   0.0140 - 0.1103i
   0.0107 - 0.1093i   0.0043 - 0.0040i   0.0005 - 0.1282i  -0.8536 - 0.0292i
   0.0047 - 0.0039i   0.0141 - 0.1100i  -0.8526 - 0.0288i  -0.0063 - 0.1275i


s4p(:,:,4) =

  -0.0362 - 0.1206i  -0.7807 + 0.3291i  -0.0284 - 0.0909i   0.0001 - 0.0043i
  -0.7805 + 0.3295i  -0.0459 - 0.1168i  -0.0004 - 0.0054i  -0.0261 - 0.0929i
  -0.0291 - 0.0912i  -0.0003 - 0.0052i  -0.0363 - 0.1105i  -0.7802 + 0.3327i
  -0.0001 - 0.0049i  -0.0263 - 0.0928i  -0.7798 + 0.3313i  -0.0404 - 0.1107i
```

```
s4p(:,:,5) =

  -0.0649 - 0.0912i  -0.5652 + 0.6246i  -0.0491 - 0.0579i  -0.0030 - 0.0022i
  -0.5640 + 0.6257i  -0.0717 - 0.0864i  -0.0041 - 0.0022i  -0.0473 - 0.0613i
  -0.0501 - 0.0576i  -0.0038 - 0.0024i  -0.0619 - 0.0819i  -0.5638 + 0.6259i
  -0.0035 - 0.0020i  -0.0477 - 0.0614i  -0.5628 + 0.6255i  -0.0646 - 0.0836i


s4p(:,:,6) =

  -0.0760 - 0.0541i  -0.2470 + 0.7983i  -0.0490 - 0.0247i  -0.0037 + 0.0024i
  -0.2483 + 0.7999i  -0.0810 - 0.0502i  -0.0045 + 0.0023i  -0.0481 - 0.0295i
  -0.0489 - 0.0253i  -0.0041 + 0.0025i  -0.0724 - 0.0479i  -0.2448 + 0.8009i
  -0.0038 + 0.0023i  -0.0475 - 0.0295i  -0.2471 + 0.8013i  -0.0749 - 0.0513i


s4p(:,:,7) =

  -0.0714 - 0.0200i   0.1122 + 0.8232i  -0.0321 - 0.0040i  -0.0004 + 0.0055i
   0.1127 + 0.8241i  -0.0736 - 0.0185i  -0.0009 + 0.0058i  -0.0331 - 0.0093i
  -0.0326 - 0.0040i  -0.0008 + 0.0055i  -0.0676 - 0.0163i   0.1154 + 0.8228i
  -0.0004 + 0.0059i  -0.0331 - 0.0093i   0.1152 + 0.8238i  -0.0708 - 0.0209i


s4p(:,:,8) =

  -0.0516 + 0.0022i   0.4469 + 0.6936i  -0.0116 - 0.0010i   0.0049 + 0.0056i
   0.4472 + 0.6934i  -0.0539 + 0.0025i   0.0049 + 0.0058i  -0.0150 - 0.0058i
  -0.0119 - 0.0012i   0.0049 + 0.0057i  -0.0497 + 0.0051i   0.4494 + 0.6931i
   0.0050 + 0.0055i  -0.0150 - 0.0058i   0.4490 + 0.6911i  -0.0534 + 0.0016i


s4p(:,:,9) =

  -0.0277 + 0.0060i   0.6935 + 0.4364i   0.0010 - 0.0123i   0.0097 + 0.0012i
   0.6933 + 0.4368i  -0.0296 + 0.0057i   0.0094 + 0.0011i  -0.0040 - 0.0156i
   0.0009 - 0.0123i   0.0093 + 0.0011i  -0.0277 + 0.0109i   0.6940 + 0.4357i
   0.0096 + 0.0009i  -0.0040 - 0.0157i   0.6951 + 0.4340i  -0.0307 + 0.0077i


s4p(:,:,10) =
```

```
  -0.0136 - 0.0068i   0.8055 + 0.1016i  -0.0017 - 0.0298i   0.0087 - 0.0069i
   0.8045 + 0.1017i  -0.0150 - 0.0075i   0.0085 - 0.0065i  -0.0075 - 0.0308i
  -0.0014 - 0.0300i   0.0083 - 0.0065i  -0.0143 + 0.0004i   0.8057 + 0.1004i
   0.0089 - 0.0068i  -0.0076 - 0.0307i   0.8059 + 0.0987i  -0.0129 - 0.0026i


s4p(:,:,11) =

  -0.0148 - 0.0237i   0.7676 - 0.2439i  -0.0170 - 0.0404i   0.0047 - 0.0113i
   0.7675 - 0.2439i  -0.0141 - 0.0259i   0.0044 - 0.0109i  -0.0224 - 0.0387i
  -0.0168 - 0.0403i   0.0045 - 0.0109i  -0.0151 - 0.0146i   0.7675 - 0.2471i
   0.0047 - 0.0114i  -0.0221 - 0.0389i   0.7673 - 0.2479i  -0.0088 - 0.0216i


s4p(:,:,12) =

  -0.0356 - 0.0360i   0.5868 - 0.5408i  -0.0407 - 0.0403i  -0.0034 - 0.0140i
   0.5872 - 0.5403i  -0.0338 - 0.0416i  -0.0031 - 0.0130i  -0.0443 - 0.0371i
  -0.0406 - 0.0402i  -0.0033 - 0.0131i  -0.0336 - 0.0249i   0.5842 - 0.5423i
  -0.0031 - 0.0139i  -0.0446 - 0.0371i   0.5859 - 0.5431i  -0.0246 - 0.0406i


s4p(:,:,13) =

  -0.0662 - 0.0284i   0.3018 - 0.7298i  -0.0635 - 0.0239i  -0.0117 - 0.0102i
   0.3028 - 0.7304i  -0.0657 - 0.0383i  -0.0110 - 0.0092i  -0.0659 - 0.0201i
  -0.0634 - 0.0239i  -0.0110 - 0.0091i  -0.0610 - 0.0157i   0.3000 - 0.7306i
  -0.0115 - 0.0102i  -0.0660 - 0.0204i   0.2996 - 0.7317i  -0.0558 - 0.0430i


s4p(:,:,14) =

  -0.0917 + 0.0025i  -0.0307 - 0.7801i  -0.0765 + 0.0051i  -0.0155 - 0.0015i
  -0.0315 - 0.7792i  -0.0944 - 0.0105i  -0.0144 - 0.0011i  -0.0768 + 0.0088i
  -0.0761 + 0.0046i  -0.0144 - 0.0012i  -0.0821 + 0.0145i  -0.0364 - 0.7790i
  -0.0157 - 0.0017i  -0.0770 + 0.0089i  -0.0354 - 0.7799i  -0.0879 - 0.0208i


s4p(:,:,15) =

  -0.0963 + 0.0478i  -0.3504 - 0.6877i  -0.0728 + 0.0388i  -0.0137 + 0.0074i
  -0.3510 - 0.6874i  -0.1031 + 0.0357i  -0.0119 + 0.0070i  -0.0723 + 0.0423i
  -0.0730 + 0.0385i  -0.0121 + 0.0072i  -0.0819 + 0.0582i  -0.3539 - 0.6859i
  -0.0136 + 0.0072i  -0.0725 + 0.0419i  -0.3542 - 0.6857i  -0.1035 + 0.0225i
```

```
s4p(:,:,16) =

  -0.0732 + 0.0920i  -0.5976 - 0.4743i  -0.0533 + 0.0679i  -0.0070 + 0.0123i
  -0.5993 - 0.4736i  -0.0826 + 0.0835i  -0.0056 + 0.0111i  -0.0516 + 0.0702i
  -0.0532 + 0.0678i  -0.0056 + 0.0109i  -0.0557 + 0.0992i  -0.6012 - 0.4711i
  -0.0070 + 0.0124i  -0.0518 + 0.0701i  -0.6003 - 0.4709i  -0.0906 + 0.0723i


s4p(:,:,17) =

  -0.0290 + 0.1190i  -0.7349 - 0.1811i  -0.0220 + 0.0840i   0.0002 + 0.0125i
  -0.7346 - 0.1822i  -0.0369 + 0.1137i   0.0006 + 0.0105i  -0.0201 + 0.0850i
  -0.0221 + 0.0839i   0.0006 + 0.0106i  -0.0094 + 0.1208i  -0.7350 - 0.1769i
   0.0002 + 0.0126i  -0.0203 + 0.0852i  -0.7359 - 0.1767i  -0.0503 + 0.1087i


s4p(:,:,18) =

   0.0215 + 0.1194i  -0.7381 + 0.1372i   0.0116 + 0.0836i   0.0051 + 0.0088i
  -0.7380 + 0.1376i   0.0178 + 0.1141i   0.0043 + 0.0072i   0.0129 + 0.0830i
   0.0114 + 0.0834i   0.0044 + 0.0071i   0.0416 + 0.1156i  -0.7372 + 0.1422i
   0.0052 + 0.0089i   0.0129 + 0.0834i  -0.7379 + 0.1428i   0.0039 + 0.1177i


s4p(:,:,19) =

   0.0632 + 0.0931i  -0.6125 + 0.4297i   0.0394 + 0.0669i   0.0053 + 0.0053i
  -0.6129 + 0.4291i   0.0635 + 0.0859i   0.0037 + 0.0041i   0.0394 + 0.0673i
   0.0392 + 0.0671i   0.0036 + 0.0036i   0.0812 + 0.0849i  -0.6097 + 0.4322i
   0.0050 + 0.0053i   0.0400 + 0.0675i  -0.6097 + 0.4322i   0.0535 + 0.0966i


s4p(:,:,20) =

   0.0810 + 0.0534i  -0.3771 + 0.6442i   0.0518 + 0.0414i   0.0049 + 0.0046i
  -0.3766 + 0.6435i   0.0832 + 0.0385i   0.0027 + 0.0047i   0.0524 + 0.0416i
   0.0519 + 0.0415i   0.0029 + 0.0047i   0.0966 + 0.0411i  -0.3729 + 0.6447i
   0.0049 + 0.0047i   0.0525 + 0.0414i  -0.3733 + 0.6444i   0.0802 + 0.0539i


s4p(:,:,21) =
```

```
   0.0744 + 0.0170i  -0.0731 + 0.7403i   0.0469 + 0.0175i   0.0067 + 0.0055i
  -0.0738 + 0.7410i   0.0717 - 0.0056i   0.0052 + 0.0065i   0.0477 + 0.0174i
   0.0471 + 0.0174i   0.0050 + 0.0064i   0.0862 + 0.0017i  -0.0697 + 0.7397i
   0.0067 + 0.0054i   0.0477 + 0.0173i  -0.0694 + 0.7398i   0.0771 + 0.0091i


s4p(:,:,22) =

   0.0516 - 0.0028i   0.2431 + 0.7003i   0.0300 + 0.0060i   0.0112 + 0.0040i
   0.2425 + 0.7005i   0.0401 - 0.0287i   0.0106 + 0.0054i   0.0305 + 0.0059i
   0.0300 + 0.0058i   0.0105 + 0.0056i   0.0591 - 0.0194i   0.2454 + 0.6986i
   0.0111 + 0.0040i   0.0305 + 0.0060i   0.2459 + 0.6993i   0.0518 - 0.0193i


s4p(:,:,23) =

   0.0292 - 0.0024i   0.5133 + 0.5301i   0.0124 + 0.0124i   0.0151 - 0.0020i
   0.5128 + 0.5306i   0.0080 - 0.0247i   0.0151 - 0.0009i   0.0130 + 0.0126i
   0.0123 + 0.0122i   0.0152 - 0.0008i   0.0319 - 0.0188i   0.5151 + 0.5284i
   0.0151 - 0.0021i   0.0131 + 0.0127i   0.5149 + 0.5274i   0.0215 - 0.0223i


s4p(:,:,24) =

   0.0182 + 0.0119i   0.6832 + 0.2629i   0.0074 + 0.0324i   0.0140 - 0.0113i
   0.6830 + 0.2634i  -0.0066 - 0.0022i   0.0143 - 0.0103i   0.0088 + 0.0328i
   0.0072 + 0.0322i   0.0143 - 0.0103i   0.0176 - 0.0028i   0.6842 + 0.2602i
   0.0140 - 0.0113i   0.0088 + 0.0328i   0.6843 + 0.2600i   0.0040 - 0.0061i


s4p(:,:,25) =

   0.0236 + 0.0275i   0.7237 - 0.0476i   0.0213 + 0.0541i   0.0067 - 0.0186i
   0.7246 - 0.0469i   0.0023 + 0.0205i   0.0069 - 0.0179i   0.0239 + 0.0536i
   0.0211 + 0.0540i   0.0070 - 0.0179i   0.0209 + 0.0151i   0.7227 - 0.0508i
   0.0066 - 0.0185i   0.0240 + 0.0537i   0.7235 - 0.0507i   0.0071 + 0.0141i


s4p(:,:,26) =

   0.0402 + 0.0343i   0.6325 - 0.3430i   0.0516 + 0.0630i  -0.0041 - 0.0199i
   0.6313 - 0.3416i   0.0270 + 0.0276i  -0.0037 - 0.0191i   0.0546 + 0.0611i
   0.0518 + 0.0630i  -0.0038 - 0.0191i   0.0371 + 0.0232i   0.6294 - 0.3452i
  -0.0040 - 0.0199i   0.0549 + 0.0611i   0.6292 - 0.3455i   0.0260 + 0.0232i
```

```
s4p(:,:,27) =

   0.0599 + 0.0287i   0.4280 - 0.5680i   0.0870 + 0.0510i  -0.0133 - 0.0152i
   0.4277 - 0.5683i   0.0529 + 0.0131i  -0.0124 - 0.0141i   0.0894 + 0.0474i
   0.0871 + 0.0510i  -0.0124 - 0.0143i   0.0561 + 0.0173i   0.4242 - 0.5709i
  -0.0133 - 0.0152i   0.0895 + 0.0470i   0.4242 - 0.5709i   0.0486 + 0.0137i


s4p(:,:,28) =

   0.0748 + 0.0116i   0.1527 - 0.6880i   0.1124 + 0.0185i  -0.0181 - 0.0069i
   0.1522 - 0.6881i   0.0648 - 0.0177i  -0.0164 - 0.0062i   0.1131 + 0.0137i
   0.1120 + 0.0186i  -0.0164 - 0.0064i   0.0687 - 0.0002i   0.1480 - 0.6890i
  -0.0182 - 0.0069i   0.1130 + 0.0131i   0.1474 - 0.6891i   0.0614 - 0.0114i


s4p(:,:,29) =

   0.0808 - 0.0112i  -0.1454 - 0.6845i   0.1167 - 0.0246i  -0.0183 + 0.0017i
  -0.1452 - 0.6838i   0.0561 - 0.0522i  -0.0160 + 0.0012i   0.1155 - 0.0302i
   0.1167 - 0.0239i  -0.0160 + 0.0012i   0.0707 - 0.0228i  -0.1500 - 0.6828i
  -0.0183 + 0.0016i   0.1153 - 0.0300i  -0.1498 - 0.6828i   0.0571 - 0.0415i


s4p(:,:,30) =

   0.0771 - 0.0355i  -0.4133 - 0.5588i   0.0983 - 0.0634i  -0.0154 + 0.0085i
  -0.4133 - 0.5587i   0.0294 - 0.0780i  -0.0134 + 0.0064i   0.0948 - 0.0686i
   0.0987 - 0.0635i  -0.0133 + 0.0064i   0.0624 - 0.0449i  -0.4179 - 0.5564i
  -0.0154 + 0.0085i   0.0947 - 0.0683i  -0.4170 - 0.5560i   0.0369 - 0.0660i


s4p(:,:,31) =

   0.0640 - 0.0578i  -0.6035 - 0.3350i   0.0639 - 0.0867i  -0.0105 + 0.0131i
  -0.6034 - 0.3351i  -0.0065 - 0.0862i  -0.0101 + 0.0102i   0.0584 - 0.0891i
   0.0639 - 0.0868i  -0.0101 + 0.0101i   0.0449 - 0.0626i  -0.6064 - 0.3314i
  -0.0106 + 0.0131i   0.0587 - 0.0888i  -0.6063 - 0.3316i   0.0076 - 0.0773i


s4p(:,:,32) =
```

```
   0.0415 - 0.0734i  -0.6848 - 0.0555i   0.0253 - 0.0870i  -0.0052 + 0.0149i
  -0.6848 - 0.0558i  -0.0389 - 0.0755i  -0.0068 + 0.0123i   0.0219 - 0.0863i
   0.0250 - 0.0867i  -0.0068 + 0.0124i   0.0214 - 0.0720i  -0.6852 - 0.0499i
  -0.0052 + 0.0150i   0.0222 - 0.0866i  -0.6860 - 0.0499i  -0.0216 - 0.0736i


s4p(:,:,33) =

   0.0154 - 0.0791i  -0.6442 + 0.2342i  -0.0018 - 0.0683i  -0.0006 + 0.0166i
  -0.6436 + 0.2336i  -0.0615 - 0.0550i  -0.0032 + 0.0154i  -0.0024 - 0.0683i
  -0.0016 - 0.0681i  -0.0033 + 0.0155i  -0.0045 - 0.0719i  -0.6424 + 0.2393i
  -0.0006 + 0.0166i  -0.0025 - 0.0687i  -0.6423 + 0.2394i  -0.0445 - 0.0598i


s4p(:,:,34) =

  -0.0117 - 0.0740i  -0.4858 + 0.4792i  -0.0096 - 0.0424i   0.0057 + 0.0174i
  -0.4851 + 0.4799i  -0.0717 - 0.0290i   0.0030 + 0.0178i  -0.0088 - 0.0449i
  -0.0094 - 0.0425i   0.0030 + 0.0178i  -0.0289 - 0.0615i  -0.4814 + 0.4835i
   0.0056 + 0.0175i  -0.0090 - 0.0450i  -0.4814 + 0.4836i  -0.0590 - 0.0402i


s4p(:,:,35) =

  -0.0332 - 0.0572i  -0.2395 + 0.6356i   0.0006 - 0.0233i   0.0132 + 0.0158i
  -0.2392 + 0.6349i  -0.0693 - 0.0052i   0.0112 + 0.0171i   0.0010 - 0.0295i
   0.0007 - 0.0232i   0.0113 + 0.0171i  -0.0463 - 0.0421i  -0.2333 + 0.6370i
   0.0131 + 0.0157i   0.0008 - 0.0294i  -0.2339 + 0.6368i  -0.0644 - 0.0189i


s4p(:,:,36) =

  -0.0433 - 0.0342i   0.0480 + 0.6736i   0.0197 - 0.0200i   0.0207 + 0.0102i
   0.0482 + 0.6728i  -0.0589 + 0.0108i   0.0199 + 0.0119i   0.0166 - 0.0301i
   0.0196 - 0.0201i   0.0200 + 0.0119i  -0.0528 - 0.0182i   0.0530 + 0.6732i
   0.0206 + 0.0102i   0.0165 - 0.0299i   0.0540 + 0.6716i  -0.0612 + 0.0004i


s4p(:,:,37) =

  -0.0402 - 0.0128i   0.3241 + 0.5872i   0.0349 - 0.0347i   0.0261 - 0.0000i
   0.3235 + 0.5867i  -0.0463 + 0.0173i   0.0259 + 0.0015i   0.0252 - 0.0461i
   0.0347 - 0.0347i   0.0258 + 0.0015i  -0.0479 + 0.0028i   0.3279 + 0.5841i
   0.0260 + 0.0000i   0.0251 - 0.0460i   0.3275 + 0.5835i  -0.0521 + 0.0144i
```

```
s4p(:,:,38) =

  -0.0289 - 0.0023i   0.5345 + 0.3935i   0.0345 - 0.0605i   0.0252 - 0.0141i
   0.5336 + 0.3934i  -0.0376 + 0.0150i   0.0252 - 0.0126i   0.0170 - 0.0690i
   0.0345 - 0.0603i   0.0253 - 0.0127i  -0.0362 + 0.0147i   0.5369 + 0.3902i
   0.0252 - 0.0141i   0.0173 - 0.0692i   0.5363 + 0.3898i  -0.0405 + 0.0204i


s4p(:,:,39) =

  -0.0221 - 0.0025i   0.6400 + 0.1339i   0.0126 - 0.0830i   0.0139 - 0.0256i
   0.6399 + 0.1344i  -0.0390 + 0.0105i   0.0143 - 0.0240i  -0.0102 - 0.0841i
   0.0128 - 0.0829i   0.0143 - 0.0240i  -0.0265 + 0.0168i   0.6429 + 0.1311i
   0.0139 - 0.0256i  -0.0102 - 0.0842i   0.6421 + 0.1309i  -0.0326 + 0.0201i


s4p(:,:,40) =

  -0.0164 - 0.0045i   0.6355 - 0.1371i  -0.0166 - 0.0879i   0.0028 - 0.0273i
   0.6355 - 0.1370i  -0.0396 + 0.0118i   0.0039 - 0.0256i  -0.0403 - 0.0796i
  -0.0162 - 0.0880i   0.0038 - 0.0258i  -0.0191 + 0.0140i   0.6361 - 0.1446i
   0.0029 - 0.0274i  -0.0404 - 0.0795i   0.6362 - 0.1440i  -0.0261 + 0.0167i


s4p(:,:,41) =

  -0.0225 - 0.0161i   0.5201 - 0.3887i  -0.0483 - 0.0814i  -0.0100 - 0.0280i
   0.5197 - 0.3881i  -0.0464 + 0.0082i  -0.0079 - 0.0266i  -0.0685 - 0.0634i
  -0.0482 - 0.0814i  -0.0079 - 0.0267i  -0.0230 + 0.0057i   0.5160 - 0.3942i
  -0.0100 - 0.0280i  -0.0686 - 0.0634i   0.5160 - 0.3942i  -0.0293 + 0.0072i


s4p(:,:,42) =

  -0.0433 - 0.0176i   0.3090 - 0.5660i  -0.0749 - 0.0580i  -0.0225 - 0.0206i
   0.3086 - 0.5654i  -0.0607 + 0.0134i  -0.0195 - 0.0203i  -0.0872 - 0.0332i
  -0.0749 - 0.0578i  -0.0197 - 0.0203i  -0.0378 + 0.0063i   0.3039 - 0.5688i
  -0.0225 - 0.0205i  -0.0871 - 0.0331i   0.3034 - 0.5691i  -0.0434 + 0.0045i


s4p(:,:,43) =
```

```
  -0.0667 - 0.0012i   0.0447 - 0.6389i  -0.0870 - 0.0247i  -0.0295 - 0.0078i
   0.0452 - 0.6381i  -0.0737 + 0.0307i  -0.0265 - 0.0094i  -0.0892 + 0.0024i
  -0.0866 - 0.0244i  -0.0266 - 0.0094i  -0.0535 + 0.0214i   0.0393 - 0.6400i
  -0.0295 - 0.0078i  -0.0891 + 0.0022i   0.0393 - 0.6393i  -0.0616 + 0.0144i


s4p(:,:,44) =

  -0.0786 + 0.0321i  -0.2223 - 0.5960i  -0.0819 + 0.0093i  -0.0292 + 0.0062i
  -0.2221 - 0.5953i  -0.0769 + 0.0585i  -0.0273 + 0.0032i  -0.0747 + 0.0335i
  -0.0816 + 0.0089i  -0.0273 + 0.0032i  -0.0587 + 0.0504i  -0.2279 - 0.5938i
  -0.0291 + 0.0062i  -0.0748 + 0.0333i  -0.2282 - 0.5929i  -0.0736 + 0.0385i


s4p(:,:,45) =

  -0.0693 + 0.0711i  -0.4465 - 0.4469i  -0.0629 + 0.0341i  -0.0228 + 0.0176i
  -0.4456 - 0.4468i  -0.0635 + 0.0889i  -0.0227 + 0.0137i  -0.0496 + 0.0518i
  -0.0632 + 0.0337i  -0.0227 + 0.0137i  -0.0442 + 0.0823i  -0.4509 - 0.4424i
  -0.0227 + 0.0175i  -0.0496 + 0.0520i  -0.4500 - 0.4423i  -0.0703 + 0.0697i


s4p(:,:,46) =

  -0.0389 + 0.1013i  -0.5867 - 0.2202i  -0.0385 + 0.0447i  -0.0131 + 0.0245i
  -0.5857 - 0.2207i  -0.0332 + 0.1097i  -0.0153 + 0.0211i  -0.0229 + 0.0550i
  -0.0388 + 0.0448i  -0.0152 + 0.0212i  -0.0113 + 0.1034i  -0.5892 - 0.2154i
  -0.0131 + 0.0245i  -0.0228 + 0.0550i  -0.5888 - 0.2145i  -0.0498 + 0.0970i


s4p(:,:,47) =

   0.0029 + 0.1118i  -0.6210 + 0.0410i  -0.0175 + 0.0430i  -0.0018 + 0.0267i
  -0.6203 + 0.0403i   0.0039 + 0.1125i  -0.0055 + 0.0250i  -0.0040 + 0.0460i
  -0.0173 + 0.0431i  -0.0055 + 0.0250i   0.0296 + 0.1030i  -0.6213 + 0.0468i
  -0.0019 + 0.0266i  -0.0040 + 0.0460i  -0.6212 + 0.0471i  -0.0174 + 0.1105i


s4p(:,:,48) =

   0.0441 + 0.0986i  -0.5477 + 0.2910i  -0.0029 + 0.0328i   0.0069 + 0.0245i
  -0.5467 + 0.2897i   0.0388 + 0.0987i   0.0031 + 0.0247i   0.0049 + 0.0344i
  -0.0029 + 0.0328i   0.0031 + 0.0247i   0.0642 + 0.0797i  -0.5450 + 0.2975i
   0.0068 + 0.0244i   0.0049 + 0.0344i  -0.5441 + 0.2976i   0.0175 + 0.1065i
```

```
s4p(:,:,49) =

   0.0672 + 0.0678i  -0.3768 + 0.4907i  -0.0013 + 0.0193i   0.0155 + 0.0220i
  -0.3776 + 0.4901i   0.0620 + 0.0666i   0.0120 + 0.0240i   0.0044 + 0.0226i
  -0.0014 + 0.0193i   0.0120 + 0.0240i   0.0772 + 0.0423i  -0.3714 + 0.4957i
   0.0154 + 0.0219i   0.0044 + 0.0226i  -0.3708 + 0.4953i   0.0429 + 0.0851i


s4p(:,:,50) =

   0.0692 + 0.0372i  -0.1382 + 0.6016i  -0.0102 + 0.0130i   0.0247 + 0.0155i
  -0.1384 + 0.6015i   0.0621 + 0.0296i   0.0225 + 0.0193i  -0.0045 + 0.0184i
  -0.0103 + 0.0129i   0.0226 + 0.0193i   0.0663 + 0.0085i  -0.1307 + 0.6033i
   0.0246 + 0.0154i  -0.0044 + 0.0184i  -0.1306 + 0.6033i   0.0504 + 0.0582i


s4p(:,:,51) =

   0.0581 + 0.0185i   0.1251 + 0.6023i  -0.0221 + 0.0186i   0.0309 + 0.0038i
   0.1242 + 0.6017i   0.0423 + 0.0036i   0.0316 + 0.0085i  -0.0138 + 0.0259i
  -0.0221 + 0.0186i   0.0316 + 0.0086i   0.0402 - 0.0080i   0.1317 + 0.6002i
   0.0309 + 0.0038i  -0.0139 + 0.0258i   0.1319 + 0.6002i   0.0432 + 0.0388i


s4p(:,:,52) =

   0.0461 + 0.0142i   0.3631 + 0.4930i  -0.0278 + 0.0360i   0.0311 - 0.0108i
   0.3628 + 0.4924i   0.0156 - 0.0027i   0.0344 - 0.0073i  -0.0153 + 0.0431i
  -0.0277 + 0.0360i   0.0344 - 0.0073i   0.0148 - 0.0029i   0.3687 + 0.4871i
   0.0310 - 0.0107i  -0.0154 + 0.0431i   0.3683 + 0.4875i   0.0315 + 0.0328i


s4p(:,:,53) =

   0.0428 + 0.0187i   0.5317 + 0.2938i  -0.0203 + 0.0586i   0.0236 - 0.0244i
   0.5315 + 0.2941i  -0.0040 + 0.0088i   0.0280 - 0.0236i  -0.0031 + 0.0625i
  -0.0202 + 0.0584i   0.0281 - 0.0236i   0.0034 + 0.0158i   0.5345 + 0.2873i
   0.0235 - 0.0244i  -0.0032 + 0.0626i   0.5344 + 0.2874i   0.0264 + 0.0376i


s4p(:,:,54) =
```

```
   0.0507 + 0.0216i   0.6009 + 0.0456i   0.0021 + 0.0769i   0.0100 - 0.0325i
   0.6002 + 0.0456i  -0.0083 + 0.0283i   0.0139 - 0.0345i   0.0225 + 0.0746i
   0.0018 + 0.0766i   0.0139 - 0.0346i   0.0094 + 0.0347i   0.6007 + 0.0388i
   0.0100 - 0.0325i   0.0225 + 0.0747i   0.6006 + 0.0391i   0.0323 + 0.0435i


s4p(:,:,55) =

   0.0641 + 0.0147i   0.5612 - 0.2061i   0.0341 + 0.0819i  -0.0056 - 0.0330i
   0.5598 - 0.2059i   0.0030 + 0.0438i  -0.0037 - 0.0366i   0.0544 + 0.0714i
   0.0336 + 0.0820i  -0.0038 - 0.0366i   0.0277 + 0.0426i   0.5583 - 0.2117i
  -0.0055 - 0.0330i   0.0546 + 0.0714i   0.5579 - 0.2109i   0.0459 + 0.0417i


s4p(:,:,56) =

   0.0736 - 0.0038i   0.4224 - 0.4162i   0.0660 + 0.0701i  -0.0182 - 0.0266i
   0.4221 - 0.4155i   0.0220 + 0.0471i  -0.0189 - 0.0301i   0.0824 + 0.0515i
   0.0657 + 0.0702i  -0.0189 - 0.0302i   0.0468 + 0.0350i   0.4190 - 0.4206i
  -0.0181 - 0.0267i   0.0825 + 0.0514i   0.4178 - 0.4198i   0.0582 + 0.0289i


s4p(:,:,57) =

   0.0722 - 0.0271i   0.2117 - 0.5495i   0.0891 + 0.0448i  -0.0264 - 0.0170i
   0.2120 - 0.5494i   0.0409 + 0.0376i  -0.0290 - 0.0189i   0.0980 + 0.0198i
   0.0891 + 0.0451i  -0.0290 - 0.0189i   0.0568 + 0.0166i   0.2067 - 0.5529i
  -0.0265 - 0.0169i   0.0979 + 0.0200i   0.2065 - 0.5515i   0.0618 + 0.0093i


s4p(:,:,58) =

   0.0600 - 0.0487i  -0.0337 - 0.5852i   0.0976 + 0.0127i  -0.0306 - 0.0056i
  -0.0336 - 0.5845i   0.0515 + 0.0168i  -0.0332 - 0.0054i   0.0959 - 0.0154i
   0.0976 + 0.0127i  -0.0332 - 0.0055i   0.0545 - 0.0044i  -0.0397 - 0.5862i
  -0.0305 - 0.0056i   0.0961 - 0.0152i  -0.0401 - 0.5855i   0.0547 - 0.0105i


s4p(:,:,59) =

   0.0402 - 0.0633i  -0.2720 - 0.5169i   0.0903 - 0.0174i  -0.0310 + 0.0063i
  -0.2717 - 0.5163i   0.0486 - 0.0076i  -0.0323 + 0.0080i   0.0770 - 0.0429i
   0.0900 - 0.0171i  -0.0323 + 0.0080i   0.0416 - 0.0202i  -0.2783 - 0.5151i
  -0.0308 + 0.0063i   0.0772 - 0.0430i  -0.2778 - 0.5146i   0.0385 - 0.0236i
```

```
s4p(:,:,60) =

   0.0181 - 0.0691i  -0.4597 - 0.3559i   0.0717 - 0.0364i  -0.0268 + 0.0185i
  -0.4592 - 0.3555i   0.0328 - 0.0275i  -0.0265 + 0.0200i   0.0490 - 0.0543i
   0.0718 - 0.0361i  -0.0265 + 0.0200i   0.0243 - 0.0265i  -0.4649 - 0.3514i
  -0.0268 + 0.0184i   0.0491 - 0.0545i  -0.4642 - 0.3513i   0.0197 - 0.0266i


s4p(:,:,61) =

  -0.0028 - 0.0681i  -0.5633 - 0.1327i   0.0511 - 0.0408i  -0.0173 + 0.0281i
  -0.5620 - 0.1322i   0.0095 - 0.0363i  -0.0165 + 0.0286i   0.0241 - 0.0470i
   0.0512 - 0.0407i  -0.0165 + 0.0287i   0.0090 - 0.0252i  -0.5661 - 0.1265i
  -0.0174 + 0.0282i   0.0242 - 0.0471i  -0.5655 - 0.1263i   0.0041 - 0.0215i


s4p(:,:,62) =

  -0.0203 - 0.0623i  -0.5656 + 0.1128i   0.0389 - 0.0331i  -0.0056 + 0.0334i
  -0.5644 + 0.1119i  -0.0134 - 0.0352i  -0.0048 + 0.0331i   0.0140 - 0.0292i
   0.0390 - 0.0331i  -0.0048 + 0.0332i  -0.0012 - 0.0202i  -0.5655 + 0.1198i
  -0.0055 + 0.0333i   0.0141 - 0.0293i  -0.5649 + 0.1197i  -0.0054 - 0.0132i


s4p(:,:,63) =

  -0.0349 - 0.0556i  -0.4648 + 0.3369i   0.0403 - 0.0241i   0.0083 + 0.0344i
  -0.4649 + 0.3357i  -0.0352 - 0.0250i   0.0086 + 0.0339i   0.0199 - 0.0126i
   0.0403 - 0.0241i   0.0086 + 0.0338i  -0.0085 - 0.0170i  -0.4608 + 0.3435i
   0.0083 + 0.0344i   0.0200 - 0.0126i  -0.4615 + 0.3438i  -0.0118 - 0.0056i


s4p(:,:,64) =

  -0.0506 - 0.0472i  -0.2800 + 0.4967i   0.0515 - 0.0233i   0.0224 + 0.0295i
  -0.2801 + 0.4966i  -0.0499 - 0.0061i   0.0223 + 0.0288i   0.0382 - 0.0070i
   0.0514 - 0.0231i   0.0224 + 0.0289i  -0.0182 - 0.0147i  -0.2738 + 0.5016i
   0.0223 + 0.0295i   0.0383 - 0.0071i  -0.2733 + 0.5004i  -0.0175 + 0.0023i


s4p(:,:,65) =
```

```
  -0.0658 - 0.0334i  -0.0463 + 0.5638i   0.0644 - 0.0360i   0.0342 + 0.0189i
  -0.0469 + 0.5637i  -0.0546 + 0.0158i   0.0338 + 0.0180i   0.0588 - 0.0192i
   0.0645 - 0.0356i   0.0338 + 0.0180i  -0.0314 - 0.0089i  -0.0393 + 0.5649i
   0.0342 + 0.0189i   0.0589 - 0.0193i  -0.0390 + 0.5649i  -0.0218 + 0.0124i


s4p(:,:,66) =

  -0.0762 - 0.0136i   0.1906 + 0.5264i   0.0686 - 0.0609i   0.0412 + 0.0030i
   0.1907 + 0.5271i  -0.0496 + 0.0352i   0.0402 + 0.0021i   0.0685 - 0.0476i
   0.0689 - 0.0607i   0.0401 + 0.0021i  -0.0445 + 0.0042i   0.1975 + 0.5251i
   0.0413 + 0.0029i   0.0683 - 0.0476i   0.1973 + 0.5252i  -0.0228 + 0.0250i


s4p(:,:,67) =

  -0.0785 + 0.0084i   0.3868 + 0.3949i   0.0565 - 0.0906i   0.0403 - 0.0165i
   0.3869 + 0.3948i  -0.0392 + 0.0476i   0.0385 - 0.0167i   0.0576 - 0.0818i
   0.0570 - 0.0906i   0.0384 - 0.0167i  -0.0530 + 0.0242i   0.3932 + 0.3921i
   0.0405 - 0.0166i   0.0575 - 0.0816i   0.3931 + 0.3912i  -0.0175 + 0.0382i


s4p(:,:,68) =

  -0.0754 + 0.0296i   0.5062 + 0.1989i   0.0261 - 0.1120i   0.0282 - 0.0336i
   0.5064 + 0.1985i  -0.0305 + 0.0552i   0.0259 - 0.0320i   0.0248 - 0.1057i
   0.0263 - 0.1121i   0.0260 - 0.0321i  -0.0537 + 0.0492i   0.5130 + 0.1935i
   0.0282 - 0.0335i   0.0250 - 0.1055i   0.5124 + 0.1931i  -0.0059 + 0.0496i


s4p(:,:,69) =

  -0.0600 + 0.0506i   0.5403 - 0.0207i  -0.0092 - 0.1142i   0.0130 - 0.0406i
   0.5397 - 0.0208i  -0.0162 + 0.0631i   0.0117 - 0.0375i  -0.0129 - 0.1067i
  -0.0089 - 0.1140i   0.0118 - 0.0376i  -0.0399 + 0.0750i   0.5442 - 0.0310i
   0.0130 - 0.0406i  -0.0129 - 0.1068i   0.5429 - 0.0311i   0.0158 + 0.0544i


s4p(:,:,70) =

  -0.0386 + 0.0532i   0.4850 - 0.2404i  -0.0414 - 0.1050i  -0.0043 - 0.0457i
   0.4851 - 0.2402i  -0.0010 + 0.0572i  -0.0032 - 0.0416i  -0.0466 - 0.0945i
  -0.0408 - 0.1048i  -0.0032 - 0.0417i  -0.0163 + 0.0882i   0.4814 - 0.2516i
  -0.0043 - 0.0457i  -0.0468 - 0.0946i   0.4815 - 0.2514i   0.0381 + 0.0414i
```

```
s4p(:,:,71) =

  -0.0275 + 0.0455i   0.3422 - 0.4194i  -0.0674 - 0.0827i  -0.0254 - 0.0414i
   0.3424 - 0.4193i   0.0031 + 0.0466i  -0.0214 - 0.0386i  -0.0721 - 0.0677i
  -0.0673 - 0.0830i  -0.0216 - 0.0386i   0.0072 + 0.0908i   0.3343 - 0.4265i
  -0.0254 - 0.0414i  -0.0722 - 0.0676i   0.3344 - 0.4249i   0.0481 + 0.0166i


s4p(:,:,72) =

  -0.0280 + 0.0368i   0.1380 - 0.5234i  -0.0803 - 0.0534i  -0.0430 - 0.0264i
   0.1384 - 0.5227i  -0.0018 + 0.0415i  -0.0376 - 0.0265i  -0.0812 - 0.0329i
  -0.0803 - 0.0537i  -0.0377 - 0.0266i   0.0271 + 0.0847i   0.1282 - 0.5240i
  -0.0429 - 0.0264i  -0.0812 - 0.0328i   0.1284 - 0.5240i   0.0415 - 0.0099i


s4p(:,:,73) =

  -0.0371 + 0.0348i  -0.0920 - 0.5322i  -0.0794 - 0.0257i  -0.0511 - 0.0042i
  -0.0919 - 0.5316i  -0.0066 + 0.0460i  -0.0466 - 0.0080i  -0.0719 - 0.0015i
  -0.0793 - 0.0258i  -0.0466 - 0.0079i   0.0412 + 0.0734i  -0.0995 - 0.5296i
  -0.0511 - 0.0042i  -0.0720 - 0.0015i  -0.1000 - 0.5289i   0.0210 - 0.0278i


s4p(:,:,74) =

  -0.0478 + 0.0436i  -0.3048 - 0.4430i  -0.0691 - 0.0058i  -0.0480 + 0.0194i
  -0.3041 - 0.4426i  -0.0039 + 0.0557i  -0.0467 + 0.0133i  -0.0508 + 0.0166i
  -0.0690 - 0.0060i  -0.0469 + 0.0133i   0.0495 + 0.0601i  -0.3093 - 0.4390i
  -0.0479 + 0.0195i  -0.0508 + 0.0166i  -0.3087 - 0.4386i  -0.0054 - 0.0304i


s4p(:,:,75) =

  -0.0524 + 0.0613i  -0.4583 - 0.2742i  -0.0566 + 0.0040i  -0.0338 + 0.0387i
  -0.4576 - 0.2741i   0.0087 + 0.0621i  -0.0371 + 0.0334i  -0.0290 + 0.0181i
  -0.0565 + 0.0038i  -0.0371 + 0.0335i   0.0522 + 0.0467i  -0.4617 - 0.2694i
  -0.0338 + 0.0387i  -0.0289 + 0.0180i  -0.4615 - 0.2698i  -0.0269 - 0.0183i


s4p(:,:,76) =
```

```
  -0.0457 + 0.0825i  -0.5258 - 0.0596i  -0.0476 + 0.0073i  -0.0129 + 0.0480i
  -0.5252 - 0.0597i   0.0240 + 0.0587i  -0.0181 + 0.0463i  -0.0177 + 0.0080i
  -0.0476 + 0.0071i  -0.0181 + 0.0465i   0.0503 + 0.0357i  -0.5288 - 0.0538i
  -0.0129 + 0.0480i  -0.0177 + 0.0080i  -0.5288 - 0.0541i  -0.0373 + 0.0016i


s4p(:,:,77) =

  -0.0278 + 0.0956i  -0.5009 + 0.1608i  -0.0430 + 0.0062i   0.0067 + 0.0478i
  -0.5011 + 0.1604i   0.0374 + 0.0473i   0.0022 + 0.0491i  -0.0178 - 0.0024i
  -0.0428 + 0.0060i   0.0022 + 0.0491i   0.0464 + 0.0262i  -0.5026 + 0.1691i
   0.0066 + 0.0478i  -0.0177 - 0.0024i  -0.5020 + 0.1691i  -0.0342 + 0.0207i


s4p(:,:,78) =

  -0.0120 + 0.0985i  -0.3888 + 0.3543i  -0.0472 + 0.0057i   0.0254 + 0.0423i
  -0.3884 + 0.3530i   0.0421 + 0.0247i   0.0227 + 0.0455i  -0.0276 - 0.0092i
  -0.0471 + 0.0055i   0.0227 + 0.0456i   0.0375 + 0.0187i  -0.3855 + 0.3615i
   0.0255 + 0.0423i  -0.0277 - 0.0092i  -0.3853 + 0.3616i  -0.0250 + 0.0296i


s4p(:,:,79) =

  -0.0008 + 0.0992i  -0.2053 + 0.4803i  -0.0558 + 0.0142i   0.0421 + 0.0286i
  -0.2051 + 0.4804i   0.0280 + 0.0020i   0.0417 + 0.0323i  -0.0442 - 0.0047i
  -0.0557 + 0.0139i   0.0417 + 0.0324i   0.0251 + 0.0185i  -0.1982 + 0.4866i
   0.0420 + 0.0287i  -0.0441 - 0.0047i  -0.1980 + 0.4867i  -0.0198 + 0.0311i


s4p(:,:,80) =

   0.0083 + 0.1017i   0.0133 + 0.5191i  -0.0603 + 0.0325i   0.0515 + 0.0079i
   0.0128 + 0.5186i  -0.0004 - 0.0054i   0.0530 + 0.0105i  -0.0574 + 0.0142i
  -0.0603 + 0.0322i   0.0530 + 0.0104i   0.0153 + 0.0277i   0.0221 + 0.5219i
   0.0515 + 0.0079i  -0.0573 + 0.0142i   0.0220 + 0.5213i  -0.0213 + 0.0327i


s4p(:,:,81) =

   0.0198 + 0.1076i   0.2260 + 0.4636i  -0.0543 + 0.0561i   0.0504 - 0.0158i
   0.2250 + 0.4634i  -0.0290 + 0.0088i   0.0524 - 0.0146i  -0.0576 + 0.0424i
  -0.0545 + 0.0559i   0.0524 - 0.0147i   0.0146 + 0.0429i   0.2342 + 0.4616i
   0.0504 - 0.0157i  -0.0576 + 0.0425i   0.2345 + 0.4614i  -0.0258 + 0.0406i
```

```
s4p(:,:,82) =

   0.0389 + 0.1129i   0.3935 + 0.3260i  -0.0360 + 0.0771i   0.0382 - 0.0363i
   0.3935 + 0.3261i  -0.0427 + 0.0399i   0.0399 - 0.0364i  -0.0409 + 0.0699i
  -0.0361 + 0.0771i   0.0399 - 0.0365i   0.0257 + 0.0575i   0.4011 + 0.3206i
   0.0383 - 0.0362i  -0.0410 + 0.0699i   0.4016 + 0.3209i  -0.0259 + 0.0550i


s4p(:,:,83) =

   0.0654 + 0.1100i   0.4891 + 0.1337i  -0.0081 + 0.0890i   0.0184 - 0.0487i
   0.4890 + 0.1337i  -0.0341 + 0.0741i   0.0191 - 0.0493i  -0.0106 + 0.0867i
  -0.0085 + 0.0889i   0.0191 - 0.0493i   0.0470 + 0.0630i   0.4942 + 0.1258i
   0.0185 - 0.0486i  -0.0106 + 0.0867i   0.4936 + 0.1257i  -0.0169 + 0.0711i


s4p(:,:,84) =

   0.0931 + 0.0933i   0.4973 - 0.0789i   0.0223 + 0.0881i  -0.0040 - 0.0509i
   0.4968 - 0.0786i  -0.0065 + 0.0967i  -0.0039 - 0.0510i   0.0249 + 0.0864i
   0.0219 + 0.0878i  -0.0039 - 0.0510i   0.0705 + 0.0535i   0.4986 - 0.0886i
  -0.0040 - 0.0508i   0.0248 + 0.0866i   0.4988 - 0.0880i   0.0010 + 0.0822i


s4p(:,:,85) =

   0.1126 + 0.0625i   0.4185 - 0.2739i   0.0490 + 0.0744i  -0.0241 - 0.0435i
   0.4179 - 0.2737i   0.0283 + 0.0988i  -0.0238 - 0.0428i   0.0548 + 0.0691i
   0.0487 + 0.0748i  -0.0238 - 0.0429i   0.0857 + 0.0301i   0.4163 - 0.2843i
  -0.0242 - 0.0434i   0.0550 + 0.0692i   0.4157 - 0.2842i   0.0233 + 0.0824i


s4p(:,:,86) =

   0.1149 + 0.0238i   0.2685 - 0.4199i   0.0655 + 0.0524i  -0.0383 - 0.0294i
   0.2684 - 0.4193i   0.0551 + 0.0823i  -0.0372 - 0.0289i   0.0705 + 0.0418i
   0.0652 + 0.0528i  -0.0372 - 0.0288i   0.0846 + 0.0007i   0.2614 - 0.4284i
  -0.0383 - 0.0294i   0.0705 + 0.0417i   0.2612 - 0.4285i   0.0417 + 0.0717i


s4p(:,:,87) =
```

```
   0.0977 - 0.0096i   0.0714 - 0.4927i   0.0693 + 0.0298i  -0.0463 - 0.0127i
   0.0715 - 0.4926i   0.0685 + 0.0577i  -0.0447 - 0.0130i   0.0710 + 0.0149i
   0.0692 + 0.0299i  -0.0447 - 0.0129i   0.0674 - 0.0232i   0.0613 - 0.4975i
  -0.0464 - 0.0127i   0.0708 + 0.0148i   0.0610 - 0.4975i   0.0519 + 0.0556i


s4p(:,:,88) =

   0.0693 - 0.0278i  -0.1385 - 0.4775i   0.0641 + 0.0133i  -0.0488 + 0.0064i
  -0.1389 - 0.4774i   0.0667 + 0.0328i  -0.0472 + 0.0049i   0.0591 - 0.0046i
   0.0640 + 0.0135i  -0.0472 + 0.0049i   0.0409 - 0.0328i  -0.1506 - 0.4768i
  -0.0487 + 0.0064i   0.0590 - 0.0044i  -0.1503 - 0.4769i   0.0530 + 0.0393i


s4p(:,:,89) =

   0.0407 - 0.0289i  -0.3245 - 0.3758i   0.0559 + 0.0062i  -0.0434 + 0.0263i
  -0.3243 - 0.3753i   0.0544 + 0.0175i  -0.0427 + 0.0240i   0.0421 - 0.0105i
   0.0560 + 0.0063i  -0.0429 + 0.0242i   0.0154 - 0.0269i  -0.3337 - 0.3693i
  -0.0433 + 0.0263i   0.0421 - 0.0104i  -0.3342 - 0.3696i   0.0484 + 0.0285i


s4p(:,:,90) =

   0.0207 - 0.0181i  -0.4495 - 0.2055i   0.0516 + 0.0071i  -0.0292 + 0.0435i
  -0.4498 - 0.2049i   0.0416 + 0.0141i  -0.0298 + 0.0411i   0.0300 - 0.0024i
   0.0515 + 0.0071i  -0.0298 + 0.0411i  -0.0000 - 0.0098i  -0.4556 - 0.1962i
  -0.0291 + 0.0435i   0.0301 - 0.0024i  -0.4553 - 0.1953i   0.0439 + 0.0246i


s4p(:,:,91) =

   0.0124 - 0.0028i  -0.4908 + 0.0004i   0.0552 + 0.0121i  -0.0077 + 0.0523i
  -0.4908 + 0.0007i   0.0374 + 0.0174i  -0.0094 + 0.0506i   0.0326 + 0.0126i
   0.0551 + 0.0122i  -0.0095 + 0.0506i  -0.0022 + 0.0097i  -0.4919 + 0.0115i
  -0.0077 + 0.0522i   0.0327 + 0.0126i  -0.4924 + 0.0120i   0.0455 + 0.0227i


s4p(:,:,92) =

   0.0159 + 0.0096i  -0.4437 + 0.2032i   0.0703 + 0.0141i   0.0150 + 0.0509i
  -0.4437 + 0.2032i   0.0403 + 0.0158i   0.0126 + 0.0503i   0.0502 + 0.0212i
   0.0702 + 0.0144i   0.0126 + 0.0505i   0.0068 + 0.0227i  -0.4399 + 0.2139i
   0.0150 + 0.0509i   0.0502 + 0.0211i  -0.4395 + 0.2135i   0.0500 + 0.0153i
```

```
s4p(:,:,93) =

   0.0244 + 0.0111i  -0.3159 + 0.3676i   0.0920 + 0.0030i   0.0353 + 0.0403i
  -0.3157 + 0.3670i   0.0419 + 0.0077i   0.0331 + 0.0409i   0.0753 + 0.0154i
   0.0918 + 0.0033i   0.0332 + 0.0408i   0.0184 + 0.0242i  -0.3085 + 0.3746i
   0.0353 + 0.0402i   0.0752 + 0.0153i  -0.3088 + 0.3743i   0.0507 + 0.0022i


s4p(:,:,94) =

   0.0274 + 0.0034i  -0.1332 + 0.4615i   0.1092 - 0.0239i   0.0490 + 0.0216i
  -0.1332 + 0.4609i   0.0375 - 0.0040i   0.0474 + 0.0230i   0.0975 - 0.0079i
   0.1093 - 0.0236i   0.0475 + 0.0229i   0.0241 + 0.0173i  -0.1243 + 0.4645i
   0.0490 + 0.0217i   0.0977 - 0.0077i  -0.1241 + 0.4640i   0.0430 - 0.0119i


s4p(:,:,95) =

   0.0211 - 0.0060i   0.0698 + 0.4692i   0.1128 - 0.0622i   0.0535 - 0.0008i
   0.0694 + 0.4692i   0.0248 - 0.0133i   0.0526 + 0.0008i   0.1065 - 0.0449i
   0.1129 - 0.0617i   0.0526 + 0.0008i   0.0205 + 0.0092i   0.0777 + 0.4689i
   0.0536 - 0.0008i   0.1068 - 0.0449i   0.0781 + 0.4689i   0.0273 - 0.0202i


s4p(:,:,96) =

   0.0062 - 0.0094i   0.2536 + 0.3931i   0.0965 - 0.1025i   0.0482 - 0.0234i
   0.2541 + 0.3934i   0.0060 - 0.0148i   0.0476 - 0.0215i   0.0944 - 0.0866i
   0.0970 - 0.1024i   0.0477 - 0.0215i   0.0094 + 0.0072i   0.2614 + 0.3911i
   0.0482 - 0.0234i   0.0947 - 0.0867i   0.2619 + 0.3914i   0.0084 - 0.0175i


s4p(:,:,97) =

  -0.0113 - 0.0007i   0.3858 + 0.2521i   0.0603 - 0.1321i   0.0330 - 0.0404i
   0.3856 + 0.2523i  -0.0132 - 0.0031i   0.0330 - 0.0382i   0.0601 - 0.1176i
   0.0609 - 0.1318i   0.0330 - 0.0381i  -0.0022 + 0.0163i   0.3942 + 0.2476i
   0.0329 - 0.0405i   0.0602 - 0.1178i   0.3942 + 0.2477i  -0.0057 - 0.0026i


s4p(:,:,98) =
```

```
  -0.0179 + 0.0214i   0.4517 + 0.0763i   0.0170 - 0.1393i   0.0150 - 0.0476i
   0.4517 + 0.0765i  -0.0190 + 0.0217i   0.0162 - 0.0451i   0.0179 - 0.1254i
   0.0175 - 0.1392i   0.0162 - 0.0452i  -0.0040 + 0.0353i   0.4592 + 0.0661i
   0.0151 - 0.0476i   0.0180 - 0.1257i   0.4587 + 0.0659i  -0.0046 + 0.0193i


s4p(:,:,99) =

  -0.0063 + 0.0386i   0.4451 - 0.1148i  -0.0198 - 0.1310i  -0.0022 - 0.0520i
   0.4452 - 0.1144i  -0.0058 + 0.0394i   0.0003 - 0.0499i  -0.0187 - 0.1173i
  -0.0195 - 0.1310i   0.0003 - 0.0500i   0.0089 + 0.0510i   0.4454 - 0.1277i
  -0.0022 - 0.0520i  -0.0187 - 0.1175i   0.4452 - 0.1284i   0.0117 + 0.0324i


s4p(:,:,100) =

   0.0092 + 0.0431i   0.3581 - 0.2881i  -0.0492 - 0.1119i  -0.0246 - 0.0503i
   0.3580 - 0.2882i   0.0116 + 0.0438i  -0.0210 - 0.0497i  -0.0473 - 0.0963i
  -0.0489 - 0.1122i  -0.0210 - 0.0497i   0.0284 + 0.0568i   0.3522 - 0.3003i
  -0.0246 - 0.0503i  -0.0472 - 0.0962i   0.3523 - 0.3002i   0.0319 + 0.0313i


s4p(:,:,101) =

   0.0225 + 0.0393i   0.2068 - 0.4093i  -0.0658 - 0.0852i  -0.0460 - 0.0366i
   0.2071 - 0.4085i   0.0276 + 0.0368i  -0.0424 - 0.0381i  -0.0603 - 0.0677i
  -0.0657 - 0.0856i  -0.0424 - 0.0380i   0.0492 + 0.0518i   0.1959 - 0.4170i
  -0.0460 - 0.0365i  -0.0604 - 0.0679i   0.1958 - 0.4171i   0.0482 + 0.0165i


s4p(:,:,102) =

   0.0325 + 0.0281i   0.0197 - 0.4571i  -0.0677 - 0.0611i  -0.0594 - 0.0137i
   0.0194 - 0.4571i   0.0349 + 0.0199i  -0.0570 - 0.0171i  -0.0584 - 0.0434i
  -0.0677 - 0.0611i  -0.0571 - 0.0170i   0.0658 + 0.0358i   0.0058 - 0.4591i
  -0.0594 - 0.0137i  -0.0583 - 0.0434i   0.0061 - 0.4591i   0.0524 - 0.0084i


s4p(:,:,103) =

   0.0330 + 0.0115i  -0.1722 - 0.4239i  -0.0622 - 0.0468i  -0.0614 + 0.0143i
  -0.1718 - 0.4235i   0.0297 + 0.0016i  -0.0612 + 0.0102i  -0.0478 - 0.0295i
  -0.0618 - 0.0469i  -0.0612 + 0.0102i   0.0723 + 0.0124i  -0.1839 - 0.4185i
  -0.0614 + 0.0143i  -0.0478 - 0.0295i  -0.1842 - 0.4189i   0.0400 - 0.0340i
```

```
s4p(:,:,104) =

   0.0227 - 0.0016i  -0.3314 - 0.3118i  -0.0582 - 0.0425i  -0.0492 + 0.0412i
  -0.3315 - 0.3117i   0.0141 - 0.0120i  -0.0513 + 0.0377i  -0.0381 - 0.0286i
  -0.0577 - 0.0426i  -0.0513 + 0.0378i   0.0658 - 0.0116i  -0.3389 - 0.3028i
  -0.0492 + 0.0412i  -0.0380 - 0.0287i  -0.3384 - 0.3033i   0.0137 - 0.0506i


s4p(:,:,105) =

   0.0084 - 0.0055i  -0.4274 - 0.1455i  -0.0609 - 0.0415i  -0.0252 + 0.0582i
  -0.4273 - 0.1458i  -0.0086 - 0.0153i  -0.0283 + 0.0567i  -0.0379 - 0.0349i
  -0.0606 - 0.0421i  -0.0283 + 0.0567i   0.0487 - 0.0287i  -0.4311 - 0.1354i
  -0.0252 + 0.0582i  -0.0381 - 0.0349i  -0.4311 - 0.1355i  -0.0190 - 0.0507i


s4p(:,:,106) =

  -0.0023 - 0.0042i  -0.4468 + 0.0429i  -0.0689 - 0.0403i   0.0013 + 0.0622i
  -0.4463 + 0.0426i  -0.0294 - 0.0051i  -0.0013 + 0.0623i  -0.0478 - 0.0386i
  -0.0690 - 0.0405i  -0.0013 + 0.0623i   0.0278 - 0.0367i  -0.4471 + 0.0538i
   0.0013 + 0.0623i  -0.0480 - 0.0386i  -0.4471 + 0.0543i  -0.0464 - 0.0348i


s4p(:,:,107) =

  -0.0140 - 0.0031i  -0.3881 + 0.2235i  -0.0844 - 0.0348i   0.0269 + 0.0566i
  -0.3882 + 0.2233i  -0.0419 + 0.0107i   0.0245 + 0.0571i  -0.0646 - 0.0360i
  -0.0844 - 0.0349i   0.0246 + 0.0571i   0.0046 - 0.0369i  -0.3827 + 0.2334i
   0.0269 + 0.0566i  -0.0644 - 0.0359i  -0.3830 + 0.2338i  -0.0620 - 0.0111i


s4p(:,:,108) =

  -0.0293 + 0.0038i  -0.2588 + 0.3629i  -0.1029 - 0.0168i   0.0491 + 0.0405i
  -0.2590 + 0.3627i  -0.0492 + 0.0283i   0.0471 + 0.0416i  -0.0841 - 0.0211i
  -0.1027 - 0.0172i   0.0471 + 0.0416i  -0.0176 - 0.0267i  -0.2499 + 0.3696i
   0.0491 + 0.0405i  -0.0840 - 0.0213i  -0.2497 + 0.3698i  -0.0666 + 0.0142i


s4p(:,:,109) =
```

```
  -0.0426 + 0.0204i  -0.0842 + 0.4340i  -0.1137 + 0.0142i   0.0623 + 0.0155i
  -0.0841 + 0.4345i  -0.0516 + 0.0486i   0.0606 + 0.0172i  -0.0970 + 0.0073i
  -0.1136 + 0.0136i   0.0606 + 0.0172i  -0.0330 - 0.0070i  -0.0743 + 0.4363i
   0.0623 + 0.0155i  -0.0973 + 0.0072i  -0.0737 + 0.4369i  -0.0617 + 0.0371i


s4p(:,:,110) =

  -0.0472 + 0.0450i   0.1016 + 0.4265i  -0.1092 + 0.0516i   0.0627 - 0.0130i
   0.1018 + 0.4265i  -0.0464 + 0.0706i   0.0615 - 0.0106i  -0.0949 + 0.0436i
  -0.1096 + 0.0511i   0.0615 - 0.0105i  -0.0373 + 0.0170i   0.1116 + 0.4252i
   0.0628 - 0.0130i  -0.0952 + 0.0438i   0.1120 + 0.4250i  -0.0497 + 0.0552i


s4p(:,:,111) =

  -0.0390 + 0.0718i   0.2662 + 0.3440i  -0.0879 + 0.0859i   0.0505 - 0.0384i
   0.2661 + 0.3446i  -0.0318 + 0.0910i   0.0502 - 0.0352i  -0.0748 + 0.0775i
  -0.0882 + 0.0856i   0.0503 - 0.0352i  -0.0294 + 0.0381i   0.2752 + 0.3395i
   0.0504 - 0.0384i  -0.0747 + 0.0781i   0.2753 + 0.3394i  -0.0331 + 0.0665i


s4p(:,:,112) =

  -0.0175 + 0.0927i   0.3818 + 0.2030i  -0.0536 + 0.1084i   0.0289 - 0.0553i
   0.3812 + 0.2031i  -0.0082 + 0.1044i   0.0304 - 0.0517i  -0.0403 + 0.0995i
  -0.0540 + 0.1082i   0.0303 - 0.0518i  -0.0139 + 0.0505i   0.3881 + 0.1954i
   0.0289 - 0.0552i  -0.0400 + 0.0995i   0.3878 + 0.1960i  -0.0143 + 0.0699i


s4p(:,:,113) =

   0.0120 + 0.1012i   0.4296 + 0.0283i  -0.0141 + 0.1147i   0.0033 - 0.0611i
   0.4291 + 0.0288i   0.0202 + 0.1057i   0.0067 - 0.0586i  -0.0003 + 0.1032i
  -0.0146 + 0.1145i   0.0066 - 0.0586i   0.0026 + 0.0520i   0.4337 + 0.0182i
   0.0034 - 0.0611i  -0.0003 + 0.1029i   0.4342 + 0.0188i   0.0021 + 0.0647i


s4p(:,:,114) =

   0.0414 + 0.0942i   0.4020 - 0.1501i   0.0220 + 0.1051i  -0.0213 - 0.0564i
   0.4021 - 0.1498i   0.0459 + 0.0932i  -0.0170 - 0.0559i   0.0344 + 0.0891i
   0.0217 + 0.1052i  -0.0170 - 0.0559i   0.0141 + 0.0443i   0.4020 - 0.1623i
  -0.0213 - 0.0566i   0.0343 + 0.0891i   0.4017 - 0.1618i   0.0112 + 0.0530i
```

```
s4p(:,:,115) =

   0.0616 + 0.0745i   0.3038 - 0.3017i   0.0481 + 0.0837i  -0.0420 - 0.0431i
   0.3040 - 0.3015i   0.0603 + 0.0691i  -0.0379 - 0.0447i   0.0556 + 0.0636i
   0.0477 + 0.0839i  -0.0380 - 0.0446i   0.0151 + 0.0327i   0.2975 - 0.3134i
  -0.0421 - 0.0432i   0.0557 + 0.0640i   0.2977 - 0.3138i   0.0097 + 0.0409i


s4p(:,:,116) =

   0.0667 + 0.0495i   0.1521 - 0.3987i   0.0600 + 0.0580i  -0.0558 - 0.0227i
   0.1521 - 0.3987i   0.0570 + 0.0436i  -0.0528 - 0.0259i   0.0601 + 0.0371i
   0.0594 + 0.0584i  -0.0528 - 0.0259i   0.0053 + 0.0257i   0.1395 - 0.4069i
  -0.0559 - 0.0226i   0.0603 + 0.0373i   0.1397 - 0.4078i  -0.0012 + 0.0360i


s4p(:,:,117) =

   0.0564 + 0.0317i  -0.0258 - 0.4259i   0.0585 + 0.0376i  -0.0610 + 0.0013i
  -0.0257 - 0.4259i   0.0414 + 0.0302i  -0.0597 - 0.0032i   0.0531 + 0.0202i
   0.0584 + 0.0380i  -0.0596 - 0.0031i  -0.0099 + 0.0311i  -0.0423 - 0.4265i
  -0.0610 + 0.0013i   0.0532 + 0.0201i  -0.0426 - 0.4265i  -0.0135 + 0.0444i


s4p(:,:,118) =

   0.0430 + 0.0284i  -0.2009 - 0.3758i   0.0531 + 0.0280i  -0.0570 + 0.0268i
  -0.2007 - 0.3753i   0.0250 + 0.0303i  -0.0579 + 0.0220i   0.0442 + 0.0147i
   0.0530 + 0.0281i  -0.0579 + 0.0220i  -0.0188 + 0.0506i  -0.2151 - 0.3673i
  -0.0568 + 0.0268i   0.0443 + 0.0146i  -0.2151 - 0.3667i  -0.0173 + 0.0638i


s4p(:,:,119) =

   0.0369 + 0.0343i  -0.3377 - 0.2557i   0.0511 + 0.0263i  -0.0410 + 0.0501i
  -0.3378 - 0.2556i   0.0152 + 0.0408i  -0.0442 + 0.0462i   0.0404 + 0.0182i
   0.0510 + 0.0264i  -0.0442 + 0.0463i  -0.0129 + 0.0764i  -0.3449 - 0.2418i
  -0.0410 + 0.0501i   0.0404 + 0.0181i  -0.3446 - 0.2414i  -0.0066 + 0.0864i


s4p(:,:,120) =
```

```
   0.0397 + 0.0410i  -0.4097 - 0.0911i   0.0558 + 0.0275i  -0.0151 + 0.0637i
  -0.4106 - 0.0914i   0.0173 + 0.0544i  -0.0199 + 0.0619i   0.0468 + 0.0246i
   0.0555 + 0.0277i  -0.0199 + 0.0618i   0.0089 + 0.0978i  -0.4097 - 0.0768i
  -0.0151 + 0.0637i   0.0467 + 0.0247i  -0.4101 - 0.0773i   0.0197 + 0.1012i


s4p(:,:,121) =

   0.0497 + 0.0430i  -0.4077 + 0.0861i   0.0682 + 0.0269i   0.0143 + 0.0636i
  -0.4082 + 0.0862i   0.0294 + 0.0617i   0.0091 + 0.0643i   0.0627 + 0.0263i
   0.0681 + 0.0274i   0.0091 + 0.0643i   0.0419 + 0.1054i  -0.4009 + 0.0974i
   0.0144 + 0.0636i   0.0627 + 0.0264i  -0.4018 + 0.0977i   0.0534 + 0.0987i


s4p(:,:,122) =

   0.0622 + 0.0347i  -0.3330 + 0.2458i   0.0870 + 0.0181i   0.0398 + 0.0502i
  -0.3336 + 0.2458i   0.0445 + 0.0570i   0.0355 + 0.0536i   0.0842 + 0.0166i
   0.0872 + 0.0185i   0.0356 + 0.0536i   0.0753 + 0.0944i  -0.3225 + 0.2519i
   0.0399 + 0.0503i   0.0844 + 0.0165i  -0.3233 + 0.2516i   0.0824 + 0.0763i


s4p(:,:,123) =

   0.0677 + 0.0166i  -0.1995 + 0.3584i   0.1051 - 0.0038i   0.0560 + 0.0280i
  -0.1994 + 0.3589i   0.0521 + 0.0413i   0.0542 + 0.0331i   0.1023 - 0.0073i
   0.1050 - 0.0037i   0.0542 + 0.0330i   0.0978 + 0.0682i  -0.1900 + 0.3588i
   0.0561 + 0.0280i   0.1023 - 0.0073i  -0.1898 + 0.3599i   0.0946 + 0.0404i


s4p(:,:,124) =

   0.0603 - 0.0032i  -0.0329 + 0.4051i   0.1132 - 0.0374i   0.0609 + 0.0022i
  -0.0325 + 0.4061i   0.0471 + 0.0230i   0.0620 + 0.0072i   0.1086 - 0.0415i
   0.1132 - 0.0368i   0.0619 + 0.0070i   0.1023 + 0.0369i  -0.0252 + 0.4024i
   0.0610 + 0.0022i   0.1087 - 0.0414i  -0.0250 + 0.4033i   0.0847 + 0.0037i


s4p(:,:,125) =

   0.0422 - 0.0153i   0.1359 + 0.3792i   0.1055 - 0.0752i   0.0551 - 0.0219i
   0.1361 + 0.3791i   0.0296 + 0.0110i   0.0580 - 0.0185i   0.0981 - 0.0782i
   0.1059 - 0.0744i   0.0581 - 0.0185i   0.0913 + 0.0123i   0.1408 + 0.3753i
   0.0552 - 0.0220i   0.0983 - 0.0784i   0.1415 + 0.3756i   0.0564 - 0.0199i
```

```
s4p(:,:,126) =

   0.0211 - 0.0136i   0.2757 + 0.2867i   0.0808 - 0.1074i   0.0411 - 0.0408i
   0.2757 + 0.2867i   0.0058 + 0.0134i   0.0445 - 0.0391i   0.0709 - 0.1076i
   0.0815 - 0.1073i   0.0445 - 0.0391i   0.0733 + 0.0021i   0.2806 + 0.2836i
   0.0409 - 0.0408i   0.0709 - 0.1078i   0.2809 + 0.2834i   0.0219 - 0.0214i


s4p(:,:,127) =

   0.0087 + 0.0013i   0.3653 + 0.1491i   0.0458 - 0.1257i   0.0225 - 0.0519i
   0.3657 + 0.1491i  -0.0119 + 0.0335i   0.0259 - 0.0516i   0.0346 - 0.1204i
   0.0460 - 0.1255i   0.0259 - 0.0515i   0.0602 + 0.0062i   0.3709 + 0.1434i
   0.0225 - 0.0519i   0.0345 - 0.1206i   0.3709 + 0.1432i  -0.0024 - 0.0017i


s4p(:,:,128) =

   0.0137 + 0.0167i   0.3943 - 0.0130i   0.0100 - 0.1287i   0.0031 - 0.0577i
   0.3947 - 0.0127i  -0.0110 + 0.0608i   0.0064 - 0.0582i   0.0006 - 0.1172i
   0.0106 - 0.1285i   0.0064 - 0.0581i   0.0601 + 0.0139i   0.3966 - 0.0218i
   0.0031 - 0.0577i   0.0005 - 0.1172i   0.3970 - 0.0218i  -0.0053 + 0.0259i


s4p(:,:,129) =

   0.0273 + 0.0183i   0.3531 - 0.1748i  -0.0218 - 0.1210i  -0.0205 - 0.0582i
   0.3535 - 0.1750i   0.0052 + 0.0814i  -0.0170 - 0.0597i  -0.0260 - 0.1027i
  -0.0212 - 0.1209i  -0.0170 - 0.0597i   0.0695 + 0.0152i   0.3512 - 0.1844i
  -0.0205 - 0.0582i  -0.0261 - 0.1028i   0.3513 - 0.1853i   0.0112 + 0.0458i


s4p(:,:,130) =

   0.0361 + 0.0070i   0.2475 - 0.3041i  -0.0465 - 0.1039i  -0.0462 - 0.0471i
   0.2478 - 0.3045i   0.0305 + 0.0912i  -0.0428 - 0.0505i  -0.0408 - 0.0814i
  -0.0459 - 0.1041i  -0.0428 - 0.0506i   0.0814 + 0.0063i   0.2408 - 0.3117i
  -0.0463 - 0.0471i  -0.0409 - 0.0818i   0.2411 - 0.3127i   0.0385 + 0.0491i


s4p(:,:,131) =
```

```
   0.0340 - 0.0097i   0.0991 - 0.3756i  -0.0605 - 0.0824i  -0.0649 - 0.0223i
   0.0991 - 0.3756i   0.0591 + 0.0863i  -0.0633 - 0.0281i  -0.0422 - 0.0626i
  -0.0602 - 0.0825i  -0.0634 - 0.0280i   0.0894 - 0.0128i   0.0898 - 0.3803i
  -0.0648 - 0.0222i  -0.0422 - 0.0626i   0.0895 - 0.3809i   0.0643 + 0.0312i


s4p(:,:,132) =

   0.0214 - 0.0253i  -0.0626 - 0.3812i  -0.0634 - 0.0638i  -0.0688 + 0.0090i
  -0.0630 - 0.3811i   0.0820 + 0.0645i  -0.0710 + 0.0024i  -0.0367 - 0.0540i
  -0.0630 - 0.0640i  -0.0710 + 0.0025i   0.0878 - 0.0391i  -0.0744 - 0.3804i
  -0.0688 + 0.0091i  -0.0367 - 0.0542i  -0.0751 - 0.3808i   0.0743 - 0.0034i


s4p(:,:,133) =

  -0.0031 - 0.0352i  -0.2129 - 0.3202i  -0.0627 - 0.0538i  -0.0579 + 0.0386i
  -0.2129 - 0.3202i   0.0885 + 0.0342i  -0.0641 + 0.0339i  -0.0342 - 0.0554i
  -0.0623 - 0.0541i  -0.0641 + 0.0338i   0.0724 - 0.0665i  -0.2227 - 0.3141i
  -0.0578 + 0.0387i  -0.0343 - 0.0553i  -0.2234 - 0.3141i   0.0603 - 0.0403i


s4p(:,:,134) =

  -0.0340 - 0.0289i  -0.3243 - 0.2044i  -0.0649 - 0.0487i  -0.0348 + 0.0591i
  -0.3241 - 0.2038i   0.0792 + 0.0068i  -0.0431 + 0.0588i  -0.0400 - 0.0602i
  -0.0647 - 0.0490i  -0.0432 + 0.0588i   0.0443 - 0.0864i  -0.3301 - 0.1939i
  -0.0347 + 0.0593i  -0.0399 - 0.0603i  -0.3300 - 0.1941i   0.0263 - 0.0647i


s4p(:,:,135) =

  -0.0588 - 0.0077i  -0.3793 - 0.0524i  -0.0715 - 0.0453i  -0.0069 + 0.0673i
  -0.3793 - 0.0523i   0.0611 - 0.0113i  -0.0141 + 0.0718i  -0.0530 - 0.0623i
  -0.0712 - 0.0454i  -0.0143 + 0.0718i   0.0100 - 0.0944i  -0.3797 - 0.0413i
  -0.0068 + 0.0673i  -0.0530 - 0.0623i  -0.3798 - 0.0410i  -0.0162 - 0.0684i


s4p(:,:,136) =

  -0.0731 + 0.0217i  -0.3666 + 0.1093i  -0.0843 - 0.0392i   0.0216 + 0.0639i
  -0.3660 + 0.1097i   0.0388 - 0.0221i   0.0176 + 0.0722i  -0.0715 - 0.0584i
  -0.0840 - 0.0397i   0.0177 + 0.0722i  -0.0270 - 0.0905i  -0.3613 + 0.1185i
   0.0216 + 0.0638i  -0.0715 - 0.0586i  -0.3612 + 0.1187i  -0.0562 - 0.0519i
```

```
s4p(:,:,137) =

  -0.0761 + 0.0544i  -0.2866 + 0.2507i  -0.1002 - 0.0236i   0.0463 + 0.0488i
  -0.2865 + 0.2508i   0.0126 - 0.0247i   0.0479 + 0.0584i  -0.0922 - 0.0442i
  -0.0999 - 0.0243i   0.0480 + 0.0584i  -0.0614 - 0.0722i  -0.2783 + 0.2559i
   0.0463 + 0.0488i  -0.0923 - 0.0442i  -0.2784 + 0.2558i  -0.0854 - 0.0190i


s4p(:,:,138) =

  -0.0682 + 0.0861i  -0.1555 + 0.3446i  -0.1106 + 0.0027i   0.0620 + 0.0249i
  -0.1555 + 0.3446i  -0.0150 - 0.0160i   0.0693 + 0.0317i  -0.1080 - 0.0178i
  -0.1109 + 0.0019i   0.0693 + 0.0317i  -0.0867 - 0.0419i  -0.1474 + 0.3457i
   0.0620 + 0.0249i  -0.1078 - 0.0178i  -0.1470 + 0.3459i  -0.0971 + 0.0232i


s4p(:,:,139) =

  -0.0502 + 0.1136i   0.0006 + 0.3750i  -0.1093 + 0.0352i   0.0661 - 0.0029i
   0.0005 + 0.3754i  -0.0380 + 0.0050i   0.0762 - 0.0020i  -0.1109 + 0.0166i
  -0.1094 + 0.0348i   0.0761 - 0.0020i  -0.0984 - 0.0049i   0.0073 + 0.3736i
   0.0662 - 0.0031i  -0.1107 + 0.0164i   0.0079 + 0.3736i  -0.0898 + 0.0649i


s4p(:,:,140) =

  -0.0241 + 0.1338i   0.1539 + 0.3395i  -0.0934 + 0.0661i   0.0587 - 0.0293i
   0.1542 + 0.3394i  -0.0501 + 0.0355i   0.0669 - 0.0345i  -0.0978 + 0.0506i
  -0.0934 + 0.0657i   0.0668 - 0.0346i  -0.0949 + 0.0323i   0.1595 + 0.3365i
   0.0587 - 0.0293i  -0.0977 + 0.0502i   0.1595 + 0.3361i  -0.0663 + 0.0970i


s4p(:,:,141) =

   0.0072 + 0.1437i   0.2788 + 0.2442i  -0.0658 + 0.0877i   0.0420 - 0.0501i
   0.2788 + 0.2442i  -0.0468 + 0.0682i   0.0450 - 0.0589i  -0.0711 + 0.0749i
  -0.0662 + 0.0870i   0.0450 - 0.0588i  -0.0787 + 0.0623i   0.2826 + 0.2405i
   0.0419 - 0.0501i  -0.0716 + 0.0748i   0.2830 + 0.2401i  -0.0345 + 0.1134i


s4p(:,:,142) =
```

```
   0.0395 + 0.1410i   0.3540 + 0.1069i  -0.0336 + 0.0947i   0.0189 - 0.0630i
   0.3535 + 0.1071i  -0.0297 + 0.0942i   0.0162 - 0.0711i  -0.0388 + 0.0844i
  -0.0343 + 0.0944i   0.0162 - 0.0711i  -0.0549 + 0.0802i   0.3562 + 0.1027i
   0.0189 - 0.0631i  -0.0390 + 0.0846i   0.3561 + 0.1029i  -0.0037 + 0.1132i


s4p(:,:,143) =

   0.0667 + 0.1260i   0.3644 - 0.0486i  -0.0056 + 0.0886i  -0.0077 - 0.0666i
   0.3640 - 0.0486i  -0.0054 + 0.1077i  -0.0141 - 0.0710i  -0.0098 + 0.0790i
  -0.0061 + 0.0884i  -0.0142 - 0.0710i  -0.0319 + 0.0844i   0.3667 - 0.0541i
  -0.0078 - 0.0665i  -0.0096 + 0.0793i   0.3668 - 0.0538i   0.0178 + 0.1002i


s4p(:,:,144) =

   0.0830 + 0.1024i   0.3100 - 0.1947i   0.0128 + 0.0744i  -0.0349 - 0.0596i
   0.3099 - 0.1948i   0.0175 + 0.1078i  -0.0419 - 0.0593i   0.0093 + 0.0642i
   0.0125 + 0.0743i  -0.0419 - 0.0593i  -0.0172 + 0.0793i   0.3105 - 0.2018i
  -0.0350 - 0.0596i   0.0094 + 0.0642i   0.3105 - 0.2018i   0.0255 + 0.0823i


s4p(:,:,145) =

   0.0846 + 0.0773i   0.1995 - 0.3049i   0.0200 + 0.0596i  -0.0589 - 0.0412i
   0.1988 - 0.3044i   0.0305 + 0.0991i  -0.0637 - 0.0374i   0.0151 + 0.0480i
   0.0196 + 0.0596i  -0.0638 - 0.0373i  -0.0146 + 0.0729i   0.1961 - 0.3121i
  -0.0590 - 0.0412i   0.0152 + 0.0480i   0.1964 - 0.3119i   0.0185 + 0.0695i


s4p(:,:,146) =

   0.0726 + 0.0601i   0.0540 - 0.3566i   0.0196 + 0.0508i  -0.0734 - 0.0118i
   0.0538 - 0.3558i   0.0308 + 0.0934i  -0.0749 - 0.0067i   0.0111 + 0.0402i
   0.0193 + 0.0509i  -0.0749 - 0.0066i  -0.0208 + 0.0743i   0.0468 - 0.3622i
  -0.0735 - 0.0118i   0.0111 + 0.0403i   0.0474 - 0.3626i   0.0048 + 0.0716i


s4p(:,:,147) =

   0.0539 + 0.0587i  -0.0971 - 0.3431i   0.0175 + 0.0513i  -0.0725 + 0.0222i
  -0.0969 - 0.3427i   0.0289 + 0.1013i  -0.0710 + 0.0266i   0.0082 + 0.0449i
   0.0173 + 0.0513i  -0.0710 + 0.0267i  -0.0275 + 0.0893i  -0.1072 - 0.3443i
  -0.0726 + 0.0223i   0.0082 + 0.0451i  -0.1070 - 0.3448i  -0.0018 + 0.0903i
```

```
s4p(:,:,148) =

   0.0427 + 0.0770i  -0.2282 - 0.2696i   0.0224 + 0.0598i  -0.0560 + 0.0528i
  -0.2282 - 0.2696i   0.0384 + 0.1177i  -0.0528 + 0.0550i   0.0161 + 0.0553i
   0.0221 + 0.0600i  -0.0528 + 0.0551i  -0.0226 + 0.1156i  -0.2372 - 0.2656i
  -0.0561 + 0.0527i   0.0160 + 0.0554i  -0.2371 - 0.2656i   0.0096 + 0.1157i


s4p(:,:,149) =

   0.0531 + 0.1030i  -0.3154 - 0.1488i   0.0397 + 0.0677i  -0.0265 + 0.0731i
  -0.3154 - 0.1487i   0.0622 + 0.1320i  -0.0233 + 0.0732i   0.0346 + 0.0619i
   0.0393 + 0.0679i  -0.0232 + 0.0731i   0.0010 + 0.1417i  -0.3206 - 0.1421i
  -0.0265 + 0.0731i   0.0346 + 0.0619i  -0.3210 - 0.1424i   0.0399 + 0.1341i


s4p(:,:,150) =

   0.0842 + 0.1193i  -0.3434 - 0.0054i   0.0649 + 0.0653i   0.0097 + 0.0765i
  -0.3430 - 0.0056i   0.0982 + 0.1348i   0.0120 + 0.0753i   0.0594 + 0.0586i
   0.0645 + 0.0656i   0.0120 + 0.0752i   0.0395 + 0.1546i  -0.3455 + 0.0010i
   0.0098 + 0.0766i   0.0595 + 0.0586i  -0.3459 + 0.0007i   0.0818 + 0.1348i


s4p(:,:,151) =

   0.1248 + 0.1157i  -0.3106 + 0.1322i   0.0899 + 0.0505i   0.0420 + 0.0621i
  -0.3106 + 0.1323i   0.1394 + 0.1189i   0.0433 + 0.0603i   0.0842 + 0.0428i
   0.0897 + 0.0509i   0.0432 + 0.0604i   0.0829 + 0.1480i  -0.3122 + 0.1385i
   0.0421 + 0.0621i   0.0842 + 0.0429i  -0.3122 + 0.1386i   0.1233 + 0.1133i


s4p(:,:,152) =

   0.1635 + 0.0903i  -0.2287 + 0.2430i   0.1099 + 0.0249i   0.0623 + 0.0358i
  -0.2286 + 0.2431i   0.1732 + 0.0815i   0.0627 + 0.0342i   0.1022 + 0.0158i
   0.1095 + 0.0254i   0.0627 + 0.0342i   0.1203 + 0.1218i  -0.2284 + 0.2508i
   0.0624 + 0.0357i   0.1023 + 0.0159i  -0.2286 + 0.2512i   0.1513 + 0.0726i


s4p(:,:,153) =
```

```
   0.1873 + 0.0456i  -0.1110 + 0.3123i   0.1190 - 0.0092i   0.0684 + 0.0059i
  -0.1107 + 0.3124i   0.1874 + 0.0291i   0.0683 + 0.0042i   0.1083 - 0.0181i
   0.1191 - 0.0087i   0.0682 + 0.0042i   0.1424 + 0.0815i  -0.1080 + 0.3207i
   0.0683 + 0.0059i   0.1085 - 0.0181i  -0.1081 + 0.3211i   0.1577 + 0.0229i


s4p(:,:,154) =

   0.1868 - 0.0065i   0.0242 + 0.3291i   0.1136 - 0.0456i   0.0618 - 0.0218i
   0.0240 + 0.3291i   0.1763 - 0.0259i   0.0609 - 0.0233i   0.1005 - 0.0525i
   0.1139 - 0.0450i   0.0610 - 0.0234i   0.1432 + 0.0377i   0.0309 + 0.3378i
   0.0618 - 0.0217i   0.1004 - 0.0527i   0.0315 + 0.3377i   0.1404 - 0.0225i


s4p(:,:,155) =

   0.1613 - 0.0513i   0.1540 + 0.2910i   0.0943 - 0.0763i   0.0459 - 0.0424i
   0.1540 + 0.2914i   0.1410 - 0.0688i   0.0442 - 0.0436i   0.0790 - 0.0807i
   0.0945 - 0.0756i   0.0443 - 0.0438i   0.1254 + 0.0026i   0.1668 + 0.2958i
   0.0459 - 0.0424i   0.0792 - 0.0809i   0.1669 + 0.2962i   0.1066 - 0.0517i


s4p(:,:,156) =

   0.1214 - 0.0758i   0.2587 + 0.2056i   0.0662 - 0.0951i   0.0254 - 0.0541i
   0.2594 + 0.2054i   0.0930 - 0.0882i   0.0231 - 0.0545i   0.0495 - 0.0960i
   0.0668 - 0.0944i   0.0231 - 0.0544i   0.0984 - 0.0156i   0.2747 + 0.2030i
   0.0254 - 0.0542i   0.0494 - 0.0959i   0.2747 + 0.2030i   0.0684 - 0.0590i


s4p(:,:,157) =

   0.0825 - 0.0781i   0.3230 + 0.0829i   0.0377 - 0.1003i   0.0048 - 0.0585i
   0.3234 + 0.0831i   0.0492 - 0.0831i   0.0026 - 0.0574i   0.0210 - 0.0966i
   0.0384 - 0.1000i   0.0027 - 0.0574i   0.0741 - 0.0173i   0.3360 + 0.0715i
   0.0048 - 0.0585i   0.0208 - 0.0966i   0.3361 + 0.0711i   0.0398 - 0.0483i


s4p(:,:,158) =

   0.0544 - 0.0681i   0.3316 - 0.0598i   0.0143 - 0.0961i  -0.0167 - 0.0583i
   0.3316 - 0.0596i   0.0204 - 0.0640i  -0.0176 - 0.0561i  -0.0003 - 0.0879i
   0.0149 - 0.0960i  -0.0176 - 0.0561i   0.0604 - 0.0099i   0.3361 - 0.0766i
  -0.0167 - 0.0582i  -0.0002 - 0.0877i   0.3362 - 0.0763i   0.0279 - 0.0319i
```

```
s4p(:,:,159) =

   0.0376 - 0.0537i   0.2765 - 0.1945i  -0.0015 - 0.0866i  -0.0401 - 0.0502i
   0.2764 - 0.1946i   0.0078 - 0.0425i  -0.0397 - 0.0481i  -0.0119 - 0.0754i
  -0.0010 - 0.0866i  -0.0397 - 0.0481i   0.0584 - 0.0020i   0.2715 - 0.2105i
  -0.0401 - 0.0503i  -0.0117 - 0.0755i   0.2716 - 0.2110i   0.0297 - 0.0213i


s4p(:,:,160) =

   0.0302 - 0.0414i   0.1680 - 0.2924i  -0.0089 - 0.0774i  -0.0609 - 0.0304i
   0.1673 - 0.2915i   0.0083 - 0.0280i  -0.0597 - 0.0292i  -0.0137 - 0.0657i
  -0.0085 - 0.0772i  -0.0597 - 0.0292i   0.0646 - 0.0018i   0.1561 - 0.3034i
  -0.0608 - 0.0304i  -0.0134 - 0.0659i   0.1561 - 0.3034i   0.0369 - 0.0242i


s4p(:,:,161) =

   0.0278 - 0.0343i   0.0292 - 0.3317i  -0.0111 - 0.0729i  -0.0705 - 0.0003i
   0.0290 - 0.3317i   0.0125 - 0.0272i  -0.0695 + 0.0002i  -0.0109 - 0.0649i
  -0.0107 - 0.0729i  -0.0695 + 0.0002i   0.0708 - 0.0125i   0.0145 - 0.3365i
  -0.0706 - 0.0002i  -0.0107 - 0.0654i   0.0145 - 0.3369i   0.0377 - 0.0409i


s4p(:,:,162) =

   0.0268 - 0.0332i  -0.1084 - 0.3100i  -0.0128 - 0.0751i  -0.0637 + 0.0314i
  -0.1086 - 0.3095i   0.0076 - 0.0387i  -0.0627 + 0.0319i  -0.0130 - 0.0741i
  -0.0123 - 0.0751i  -0.0627 + 0.0319i   0.0698 - 0.0316i  -0.1245 - 0.3081i
  -0.0637 + 0.0315i  -0.0133 - 0.0745i  -0.1244 - 0.3085i   0.0223 - 0.0634i


s4p(:,:,163) =

   0.0223 - 0.0386i  -0.2241 - 0.2370i  -0.0198 - 0.0836i  -0.0433 + 0.0554i
  -0.2236 - 0.2364i  -0.0143 - 0.0493i  -0.0419 + 0.0559i  -0.0268 - 0.0848i
  -0.0190 - 0.0836i  -0.0418 + 0.0559i   0.0569 - 0.0537i  -0.2375 - 0.2274i
  -0.0433 + 0.0555i  -0.0272 - 0.0852i  -0.2374 - 0.2276i  -0.0101 - 0.0786i


s4p(:,:,164) =
```

```
   0.0093 - 0.0458i  -0.3000 - 0.1232i  -0.0366 - 0.0926i  -0.0157 + 0.0679i
  -0.2999 - 0.1236i  -0.0457 - 0.0466i  -0.0138 + 0.0675i  -0.0494 - 0.0888i
  -0.0361 - 0.0927i  -0.0138 + 0.0675i   0.0315 - 0.0717i  -0.3069 - 0.1095i
  -0.0157 + 0.0679i  -0.0499 - 0.0886i  -0.3066 - 0.1094i  -0.0509 - 0.0775i


s4p(:,:,165) =

  -0.0128 - 0.0486i  -0.3231 + 0.0113i  -0.0636 - 0.0946i   0.0142 + 0.0681i
  -0.3227 + 0.0114i  -0.0766 - 0.0293i   0.0158 + 0.0666i  -0.0761 - 0.0824i
  -0.0630 - 0.0949i   0.0157 + 0.0666i  -0.0043 - 0.0794i  -0.3218 + 0.0261i
   0.0143 + 0.0681i  -0.0763 - 0.0820i  -0.3218 + 0.0259i  -0.0910 - 0.0588i


s4p(:,:,166) =

  -0.0407 - 0.0399i  -0.2875 + 0.1429i  -0.0953 - 0.0820i   0.0417 + 0.0555i
  -0.2876 + 0.1427i  -0.1012 - 0.0005i   0.0423 + 0.0533i  -0.1017 - 0.0638i
  -0.0948 - 0.0824i   0.0423 + 0.0532i  -0.0448 - 0.0710i  -0.2807 + 0.1543i
   0.0417 + 0.0556i  -0.1017 - 0.0636i  -0.2802 + 0.1543i  -0.1226 - 0.0247i


s4p(:,:,167) =

  -0.0668 - 0.0170i  -0.2014 + 0.2472i  -0.1212 - 0.0531i   0.0609 + 0.0320i
  -0.2017 + 0.2470i  -0.1158 + 0.0363i   0.0603 + 0.0298i  -0.1196 - 0.0339i
  -0.1207 - 0.0539i   0.0603 + 0.0298i  -0.0809 - 0.0451i  -0.1916 + 0.2535i
   0.0609 + 0.0319i  -0.1196 - 0.0338i  -0.1914 + 0.2532i  -0.1400 + 0.0191i


s4p(:,:,168) =

  -0.0829 + 0.0186i  -0.0814 + 0.3056i  -0.1324 - 0.0137i   0.0675 + 0.0031i
  -0.0811 + 0.3057i  -0.1185 + 0.0771i   0.0660 + 0.0013i  -0.1246 + 0.0026i
  -0.1324 - 0.0145i   0.0660 + 0.0013i  -0.1041 - 0.0053i  -0.0710 + 0.3078i
   0.0673 + 0.0031i  -0.1244 + 0.0024i  -0.0712 + 0.3074i  -0.1404 + 0.0660i


s4p(:,:,169) =

  -0.0829 + 0.0608i   0.0508 + 0.3099i  -0.1257 + 0.0267i   0.0612 - 0.0246i
   0.0507 + 0.3096i  -0.1087 + 0.1179i   0.0591 - 0.0257i  -0.1142 + 0.0382i
  -0.1257 + 0.0260i   0.0591 - 0.0257i  -0.1092 + 0.0401i   0.0600 + 0.3090i
   0.0610 - 0.0246i  -0.1140 + 0.0383i   0.0599 + 0.3087i  -0.1249 + 0.1085i
```

```
s4p(:,:,170) =

  -0.0651 + 0.1009i   0.1722 + 0.2613i  -0.1035 + 0.0590i   0.0448 - 0.0460i
   0.1716 + 0.2609i  -0.0862 + 0.1540i   0.0423 - 0.0461i  -0.0907 + 0.0658i
  -0.1038 + 0.0582i   0.0424 - 0.0461i  -0.0953 + 0.0818i   0.1806 + 0.2574i
   0.0448 - 0.0461i  -0.0908 + 0.0657i   0.1808 + 0.2573i  -0.0975 + 0.1411i


s4p(:,:,171) =

  -0.0313 + 0.1301i   0.2632 + 0.1673i  -0.0727 + 0.0768i   0.0227 - 0.0588i
   0.2631 + 0.1673i  -0.0543 + 0.1808i   0.0202 - 0.0579i  -0.0604 + 0.0799i
  -0.0733 + 0.0761i   0.0203 - 0.0579i  -0.0674 + 0.1110i   0.2708 + 0.1605i
   0.0227 - 0.0588i  -0.0605 + 0.0799i   0.2708 + 0.1604i  -0.0641 + 0.1600i


s4p(:,:,172) =

   0.0112 + 0.1406i   0.3080 + 0.0436i  -0.0430 + 0.0792i  -0.0016 - 0.0627i
   0.3080 + 0.0439i  -0.0174 + 0.1952i  -0.0036 - 0.0609i  -0.0313 + 0.0799i
  -0.0434 + 0.0791i  -0.0036 - 0.0608i  -0.0348 + 0.1223i   0.3134 + 0.0335i
  -0.0017 - 0.0627i  -0.0313 + 0.0798i   0.3133 + 0.0339i  -0.0319 + 0.1649i


s4p(:,:,173) =

   0.0504 + 0.1302i   0.2983 - 0.0873i  -0.0212 + 0.0710i  -0.0259 - 0.0580i
   0.2976 - 0.0869i   0.0189 + 0.1976i  -0.0271 - 0.0553i  -0.0101 + 0.0697i
  -0.0214 + 0.0709i  -0.0271 - 0.0551i  -0.0086 + 0.1176i   0.2988 - 0.1003i
  -0.0259 - 0.0580i  -0.0103 + 0.0696i   0.2986 - 0.0998i  -0.0082 + 0.1597i


s4p(:,:,174) =

   0.0752 + 0.1044i   0.2342 - 0.2026i  -0.0105 + 0.0595i  -0.0480 - 0.0447i
   0.2341 - 0.2022i   0.0485 + 0.1898i  -0.0478 - 0.0413i  -0.0002 + 0.0561i
  -0.0108 + 0.0594i  -0.0479 - 0.0412i   0.0026 + 0.1049i   0.2285 - 0.2160i
  -0.0480 - 0.0447i  -0.0002 + 0.0561i   0.2281 - 0.2159i   0.0037 + 0.1523i


s4p(:,:,175) =
```

```
   0.0793 + 0.0744i   0.1272 - 0.2801i  -0.0092 + 0.0526i  -0.0648 - 0.0218i
   0.1276 - 0.2799i   0.0677 + 0.1782i  -0.0631 - 0.0185i  -0.0001 + 0.0470i
  -0.0094 + 0.0525i  -0.0630 - 0.0184i  -0.0016 + 0.0978i   0.1144 - 0.2894i
  -0.0649 - 0.0218i  -0.0002 + 0.0471i   0.1146 - 0.2893i   0.0056 + 0.1516i


s4p(:,:,176) =

   0.0649 + 0.0539i  -0.0004 - 0.3048i  -0.0103 + 0.0547i  -0.0705 + 0.0088i
  -0.0004 - 0.3041i   0.0787 + 0.1708i  -0.0668 + 0.0113i  -0.0032 + 0.0483i
  -0.0107 + 0.0546i  -0.0668 + 0.0114i  -0.0127 + 0.1064i  -0.0177 - 0.3057i
  -0.0705 + 0.0089i  -0.0033 + 0.0483i  -0.0175 - 0.3057i   0.0067 + 0.1646i


s4p(:,:,177) =

   0.0424 + 0.0530i  -0.1246 - 0.2740i  -0.0062 + 0.0647i  -0.0603 + 0.0401i
  -0.1239 - 0.2743i   0.0908 + 0.1728i  -0.0553 + 0.0404i   0.0008 + 0.0584i
  -0.0066 + 0.0647i  -0.0552 + 0.0405i  -0.0161 + 0.1324i  -0.1402 - 0.2656i
  -0.0603 + 0.0402i   0.0006 + 0.0585i  -0.1397 - 0.2650i   0.0196 + 0.1887i


s4p(:,:,178) =

   0.0284 + 0.0728i  -0.2237 - 0.1968i   0.0089 + 0.0769i  -0.0360 + 0.0633i
  -0.2234 - 0.1970i   0.1119 + 0.1759i  -0.0314 + 0.0604i   0.0163 + 0.0679i
   0.0085 + 0.0768i  -0.0312 + 0.0603i   0.0003 + 0.1667i  -0.2313 - 0.1806i
  -0.0360 + 0.0633i   0.0161 + 0.0681i  -0.2314 - 0.1804i   0.0513 + 0.2113i


s4p(:,:,179) =

   0.0356 + 0.1003i  -0.2817 - 0.0868i   0.0345 + 0.0813i  -0.0037 + 0.0721i
  -0.2822 - 0.0874i   0.1409 + 0.1729i  -0.0013 + 0.0669i   0.0397 + 0.0689i
   0.0340 + 0.0813i  -0.0012 + 0.0667i   0.0389 + 0.1941i  -0.2787 - 0.0703i
  -0.0037 + 0.0721i   0.0397 + 0.0692i  -0.2788 - 0.0699i   0.0981 + 0.2198i


s4p(:,:,180) =

   0.0632 + 0.1186i  -0.2903 + 0.0351i   0.0633 + 0.0724i   0.0283 + 0.0645i
  -0.2899 + 0.0351i   0.1730 + 0.1582i   0.0274 + 0.0588i   0.0644 + 0.0585i
   0.0626 + 0.0727i   0.0275 + 0.0587i   0.0926 + 0.2013i  -0.2788 + 0.0453i
   0.0284 + 0.0645i   0.0644 + 0.0587i  -0.2788 + 0.0453i   0.1500 + 0.2073i
```

```
s4p(:,:,181) =

   0.1005 + 0.1167i  -0.2487 + 0.1485i   0.0872 + 0.0512i   0.0515 + 0.0436i
  -0.2488 + 0.1478i   0.2015 + 0.1308i   0.0480 + 0.0395i   0.0845 + 0.0379i
   0.0867 + 0.0520i   0.0480 + 0.0395i   0.1474 + 0.1830i  -0.2358 + 0.1501i
   0.0516 + 0.0435i   0.0846 + 0.0382i  -0.2357 + 0.1498i   0.1960 + 0.1738i


s4p(:,:,182) =

   0.1324 + 0.0926i  -0.1667 + 0.2357i   0.1016 + 0.0219i   0.0618 + 0.0166i
  -0.1677 + 0.2349i   0.2195 + 0.0924i   0.0569 + 0.0151i   0.0961 + 0.0100i
   0.1014 + 0.0227i   0.0570 + 0.0151i   0.1900 + 0.1415i  -0.1576 + 0.2301i
   0.0617 + 0.0166i   0.0963 + 0.0101i  -0.1577 + 0.2301i   0.2257 + 0.1233i


s4p(:,:,183) =

   0.1460 + 0.0539i  -0.0574 + 0.2829i   0.1038 - 0.0105i   0.0595 - 0.0092i
  -0.0580 + 0.2825i   0.2215 + 0.0489i   0.0553 - 0.0084i   0.0963 - 0.0209i
   0.1036 - 0.0099i   0.0554 - 0.0084i   0.2099 + 0.0865i  -0.0541 + 0.2749i
   0.0596 - 0.0090i   0.0968 - 0.0209i  -0.0541 + 0.2746i   0.2327 + 0.0652i


s4p(:,:,184) =

   0.1357 + 0.0158i   0.0631 + 0.2817i   0.0938 - 0.0402i   0.0489 - 0.0297i
   0.0632 + 0.2827i   0.2063 + 0.0105i   0.0458 - 0.0275i   0.0849 - 0.0490i
   0.0938 - 0.0393i   0.0458 - 0.0275i   0.2036 + 0.0317i   0.0609 + 0.2762i
   0.0491 - 0.0297i   0.0853 - 0.0495i   0.0609 + 0.2768i   0.2165 + 0.0125i


s4p(:,:,185) =

   0.1077 - 0.0063i   0.1738 + 0.2322i   0.0751 - 0.0616i   0.0335 - 0.0436i
```

See Also

freqresp | rfmodel.rational | s2tf | timeresp | writeva

Introduced in R2007b

stabilityk

Stability factor K of 2-port network

Syntax

```
[k,b1,b2,delta] = stabilityk(s_params)
[k,b1,b2,delta] = stabilityk(hs)
```

Description

`[k,b1,b2,delta] = stabilityk(s_params)` calculates and returns the stability factor, `k`, and the conditions `b1`, `b2`, and `delta` for the 2-port network. The input `s_params` is a complex 2-by-2-by-M array, representing M 2-port S-parameters.

`[k,b1,b2,delta] = stabilityk(hs)` calculates and returns the stability factor and stability conditions for the 2-port network represented by the S-parameter object `hs`.

Examples

Stability of Network

Examine the stability of network data from a file. Calculate stability factor and conditions

```
ckt = read(rfckt.passive,'passive.s2p');
s_params = ckt.NetworkData.Data;
freq = ckt.NetworkData.Freq;
[k,b1,b2,delta] = stabilityk(s_params);
```

Check stability criteria

```
stability_index = (k>1)&(abs(delta)<1);
is_stable = all(stability_index)
```

```
is_stable = logical
   1
```

List frequencies with unstable S-parameters

```
freq_unstable = freq(~stability_index)
```

```
freq_unstable =

  0x1 empty double column vector
```

Algorithms

Necessary and sufficient conditions for stability are `k>1` and `abs(delta)<1`. `stabilityk` calculates the outputs using the equations

$$K = \frac{1 - |S_{11}|^2 - |S_{22}|^2 + |\Delta|^2}{2|S_{12}S_{21}|}$$

$$B_1 = 1 + |S_{11}|^2 - |S_{22}|^2 - |\Delta|^2$$

$$B_2 = 1 - |S_{11}|^2 + |S_{22}|^2 - |\Delta|^2$$

where:

- S_{11}, S_{12}, S_{21}, and S_{22} are S-parameters from the input argument `s_params`.
- Δ is a vector whose members are the determinants of the M 2-port S-parameter matrices:

  $$\Delta = S_{11}S_{22} - S_{12}S_{21}$$

The function performs these calculations element-wise for each of the M S-parameter matrices in `s_params`.

References

Gonzalez, Guillermo, *Microwave Transistor Amplifiers: Analysis and Design*, 2nd edition. Prentice-Hall, 1997, pp. 217–228.

See Also

`gammaml` | `gammams` | `stabilitymu`

Introduced before R2006a

stabilitymu

Stability factor μ of 2-port network

Syntax

```
[mu,muprime] = stabilitymu(s_params)
```

Description

`[mu,muprime] = stabilitymu(s_params)` calculates and returns the stability factors μ and μ' of a 2-port network. The input `s_params` is a complex 2-by-2-by-M array, representing M 2-port S-parameters.

`[mu,muprime] = stabilitymu(hs)` calculates and returns the stability factors for the network represented by the S-parameter object `hs`.

The stability factor, μ, defines the minimum distance between the center of the unit Smith chart and the unstable region in the load plane. The function assumes that port 2 is the load.

The stability factor, μ', defines the minimum distance between the center of the unit Smith chart and the unstable region in the source plane. The function assumes that port 1 is the source.

Having $\mu > 1$ or $\mu' > 1$ is the necessary and sufficient condition for the 2-port linear network to be unconditionally stable, as described by the S-parameters.

Examples

Stability Factor of Two-Port Network

Examine the stability of network data from a file.Calculate stability factor and conditions

```
ckt = read(rfckt.passive,'passive.s2p');
s_params = ckt.NetworkData.Data;
```

```
freq = ckt.NetworkData.Freq;
[mu,muprime] = stabilitymu(s_params);
```

Check stability criteria

```
stability_index = (mu>1)|(muprime>1);
is_stable = all(stability_index)
```

```
is_stable = logical
   1
```

List frequencies with unstable S-parameters

```
freq_unstable = freq(~stability_index);
```

Algorithms

`stabilitymu` calculates the stability factors using the equations

$$\mu = \frac{1-|S_{11}|^2}{|S_{22}-S_{11}^{*}\Delta| + |S_{21}S_{12}|}$$

$$\mu' = \frac{1-|S_{22}|^2}{|S_{11}-S_{22}^{*}\Delta| + |S_{21}S_{12}|}$$

where:

- S_{11}, S_{12}, S_{21}, and S_{22} are S-parameters, from the input argument `s_params`.
- Δ is a vector whose members are the determinants of the M 2-port S-parameter matrices:

  $$\Delta = S_{11}S_{22} - S_{12}S_{21}$$

- S^* is the complex conjugate of the corresponding S-parameter.

The function performs these calculations element-wise for each of the M S-parameter matrices in `s_params`.

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`

Introduced before R2006a

t2s

Convert T-parameters to S-parameters

Syntax

```
s_params = t2s(t_params)
```

Description

`s_params = t2s(t_params)` converts the chain scattering parameters `t_params` into the scattering parameters `s_params`. The `t_params` input is a complex 2-by-2-by-*M* array, representing *M* 2-port T-parameters. `s_params` is a complex 2-by-2-by-*M* array, representing *M* 2-port S-parameters.

This function defines the T-parameters as

$$\begin{bmatrix} a_1 \\ b_1 \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \begin{bmatrix} b_2 \\ a_2 \end{bmatrix},$$

where:

- a_1 is the incident wave at the first port.
- b_1 is the reflected wave at the first port.
- a_2 is the incident wave at the second port.
- b_2 is the reflected wave at the second port.

Examples

T-Parameters to S-Parameters

Define a matrix of T-parameters

```
t11 =  0.138451095405929   - 0.230421317393041i;
t21 = -0.0451985986689165  + 0.157626245839348i;
t12 =  0.0353675449261375  + 0.115682026931012i;
t22 = -0.00194567217559662 - 0.0291212122613417i;
t_params = [t11 t12; t21 t22];
```

Convert to S-parameters

```
s_params = t2s(t_params)
```

```
s_params = 2×2 complex

  -0.5892 + 0.1579i   0.0372 + 0.0335i
   1.9159 + 3.1887i   0.3011 - 0.3344i
```

References

Gonzalez, Guillermo, *Microwave Transistor Amplifiers: Analysis and Design*, 2nd edition. Prentice-Hall, 1997, p. 25.

See Also

abcd2s | h2s | s2t | y2s | z2s

Introduced before R2006a

vswr

VSWR at given reflection coefficient Γ

Syntax

```
ratio = vswr(gamma)
```

Description

`ratio = vswr(gamma)` calculates the voltage standing-wave ratio *VSWR* at the given reflection coefficientΓ as

$$VSWR = \frac{1+|\Gamma|}{1-|\Gamma|}$$

The input `gamma` is a complex vector. The output `ratio` is a real vector of the same length as `gamma`.

Examples

VSWR from Reflection Coefficient

Calculate the VSWR for a given reflection coefficient.

```
gamma = 1/3;
ratio = vswr(gamma)
```

```
ratio = 2.0000
```

See Also

`gamma2z` | `gammain` | `gammaout`

Introduced before R2006a

y2abcd

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*N*-by-2*N*-by-*M* array, representing *M* 2*N*-port Y-parameters. `abcd_params` is a complex 2*N*-by-2*N*-by-*M* array, representing *M* 2*N*-port ABCD-parameters. The output ABCD-parameters matrices have distinct *A*, *B*, *C*, and *D* submatrices:

$$\begin{bmatrix} [A] & [B] \\ [C] & [D] \end{bmatrix}$$

Examples

Y-Parameters to ABCD-Parameters

Define a matrix of Y-parameters.

```
Y11 =  0.0488133074245012 - 0.390764155450191i;
Y12 = -0.0488588365420561 + 0.390719345880018i;
Y21 = -0.0487261119282660 + 0.390851884427087i;
Y22 =  0.0487710062903760 - 0.390800401433241i;
y_params = [Y11,Y12; Y21,Y22];
```

Convert to ABCD-parameters

```
abcd_params = y2abcd(y_params)
```

```
abcd_params = 2×2 complex

   0.9999 + 0.0001i   0.3141 + 2.5194i
  -0.0000 + 0.0000i   0.9998 + 0.0002i
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see “RF Network Parameter Objects” on page 2-11.

See Also

abcd2y | h2abcd | s2abcd | y2h | y2s | y2z | z2abcd

Introduced before R2006a

y2h

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.

Examples

Y-Parameters to H-Parameters

Define a matrix of Y-parameters.

```
Y11 =  0.0488133074245012 - 0.390764155450191i;
Y12 = -0.0488588365420561 + 0.390719345880018i;
Y21 = -0.0487261119282660 + 0.390851884427087i;
Y22 =  0.0487710062903760 - 0.390800401433241i;
y_params = [Y11,Y12; Y21,Y22];
```

Convert to h-parameters

```
h_params = y2h(y_params)
```

```
h_params = 2×2 complex

   0.3148 + 2.5198i   0.9999 + 0.0001i
  -1.0002 + 0.0002i  -0.0000 + 0.0000i
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see “RF Network Parameter Objects” on page 2-11.

See Also

abcd2h | h2y | s2h | y2abcd | y2s | y2z | z2h

Introduced before R2006a

y2s

Convert Y-parameters to S-parameters

Syntax

```
s_params = y2s(y_params,z0)
```

Description

`s_params = y2s(y_params,z0)` converts the admittance parameters `y_params` into the scattering parameters `s_params`. The `y_params` input is a complex N-by-N-by-*M* array, representing *M* N-port Y-parameters. `z0` is the reference impedance. The default value of is `z0` 50 ohms. `s_params` is a complex N-by-N-by-*M* array, representing *M* N-port S-parameters.

Examples

Y-Parameters to S-Parameters

Define a matrix of Y-parameters.

```
Y11 =  0.0488133074245012 - 0.390764155450191i;
Y12 = -0.0488588365420561 + 0.390719345880018i;
Y21 = -0.0487261119282660 + 0.390851884427087i;
Y22 =  0.0487710062903760 - 0.390800401433241i;
y_params = [Y11,Y12; Y21,Y22];
```

Convert to s-parameters

```
s_params = y2s(y_params)
```

```
s_params = 2×2 complex

   0.0038 + 0.0248i   0.9961 - 0.0250i
```

```
   0.9964 - 0.0254i   0.0037 + 0.0249i
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see "RF Network Parameter Objects" on page 2-11.

See Also

`abcd2s` | `h2s` | `s2y` | `y2abcd` | `y2h` | `y2s` | `y2z` | `z2s`

Introduced before R2006a

y2z

Convert Y-parameters to Z-parameters

Syntax

```
z_params = y2z(y_params)
```

Description

`z_params = y2z(y_params)` converts the `y_params` into the`z_params`.

Input Arguments

`y_params` — Admittance parameters
N-by-*N*-by-*M* complex array

Admittance parameters, specified as a *N*-by-*N*-by-*M* complex array representing *M* N-port Y-parameters.

Output Arguments

`z_params` — Impedance parameters
N-by-*N*-by-*M* complex array

Impedance parameters, specified as a *N*-by-*N*-by-*M* complex array representing *M* N-port Z-parameters.

Examples

Convert Y-parameters to Z-parameters

Define a matrix of Y-parameters.

```
Y11 =  0.0488133074245012 - 0.390764155450191i;
Y12 = -0.0488588365420561 + 0.390719345880018i;
Y21 = -0.0487261119282660 + 0.390851884427087i;
Y22 =  0.0487710062903760 - 0.390800401433241i;
y_params = [Y11,Y12; Y21,Y22];
```

Convert to Z-parameters.

```
z_params = y2z(y_params)
```

```
z_params = 2×2 complex
10^5 ×

  -0.1457 - 1.4837i  -0.1453 - 1.4835i
  -0.1459 - 1.4839i  -0.1455 - 1.4836i
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see “RF Network Parameter Objects” on page 2-11.

See Also

abcd2z | h2z | y2abcd | y2h | y2s | y2z | z2s | z2y

Introduced before R2006a

z2abcd

Convert Z-parameters to ABCD-parameters

Syntax

```
abcd_params = z2abcd(z_params)
```

Description

`abcd_params = z2abcd(z_params)` converts the impedance parameters `z_params` into the ABCD-parameters `abcd_params`. The `z_params` input is a complex $2N$-by-$2N$-by-M array, representing M $2N$-port Z-parameters. `abcd_params` is a complex $2N$-by-$2N$-by-M array, representing M $2N$-port ABCD-parameters. The output ABCD-parameters matrices have distinct A, B, C, and D submatrices:

$$\begin{bmatrix} [A] & [B] \\ [C] & [D] \end{bmatrix}$$

Examples

Z-Parameters to ABCD-Parameters

Define a matrix of Z-parameters.

```
Z11 = -14567.2412789287 - 148373.315116592i;
Z12 = -14588.1106171651 - 148388.583516562i;
Z21 = -14528.0522132692 - 148350.705757767i;
Z22 = -14548.5996561832 - 148363.457002006i;
z_params = [Z11,Z12; Z21,Z22];
```

Convert to abcd-parameters

```
abcd_params = z2abcd(z_params)
```

```
abcd_params = 2×2 complex

   1.0002 - 0.0002i   0.3151 + 2.5200i
  -0.0000 + 0.0000i   1.0001 - 0.0001i
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see “RF Network Parameter Objects” on page 2-11.

See Also

abcd2z | h2abcd | s2abcd | y2abcd | z2h | z2s | z2y

Introduced before R2006a

z2gamma

Convert impedance to reflection coefficient

Syntax

```
gamma = z2gamma(z)
gamma = z2gamma(z,z0)
```

Description

`gamma = z2gamma(z)` converts the impedance `z` to the reflection coefficient `gamma` using a reference impedance of 50 ohms.

`gamma = z2gamma(z,z0)` converts the impedance `z` to the reflection coefficient `gamma` using a reference impedance of `z0` ohms.

Examples

Impedance to Reflection Coefficient

Convert an impedance of 100 ohms into a reflection coefficient, using a 50-ohm reference impedance

```
z = 100;
gamma = z2gamma(z)
```

```
gamma = 0.3333
```

Algorithms

`z2gamma` calculates the coefficient using the equation

$$\Gamma = \frac{Z - Z_0}{Z + Z_0}$$

See Also

gamma2z | gammain | gammaout

Introduced in R2008a

z2h

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.

Examples

Convert Z-Parameters to H-Parameters

Define a matrix of z-parameters.

```
Z11 = -14567.2412789287 - 148373.315116592i;
Z12 = -14588.1106171651 - 148388.583516562i;
Z21 = -14528.0522132692 - 148350.705757767i;
Z22 = -14548.5996561832 - 148363.457002006i;
z_params = [Z11,Z12; Z21,Z22];
```

Convert the z-parameters to h-parameters.

```
h_params = z2h(z_params)
```

```
h_params = 2×2 complex

   0.3148 + 2.5198i   1.0002 - 0.0002i
  -0.9999 - 0.0001i  -0.0000 + 0.0000i
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see “RF Network Parameter Objects” on page 2-11.

See Also

abcd2h | h2z | s2h | y2h | z2abcd | z2s | z2y

Introduced before R2006a

z2s

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.

Examples

Z-Parameters to S-Parameters

Define a matrix of Z-parameters.

```
Z11 = -14567.2412789287 - 148373.315116592i;
Z12 = -14588.1106171651 - 148388.583516562i;
Z21 = -14528.0522132692 - 148350.705757767i;
Z22 = -14548.5996561832 - 148363.457002006i;
z_params = [Z11,Z12; Z21,Z22];
```

Convert to s-parameters

```
s_params = z2s(z_params)
```

```
s_params = 2×2 complex

   0.0038 + 0.0248i   0.9964 - 0.0254i
```

```
   0.9961 - 0.0250i   0.0037 + 0.0249i
```

Alternatives

You can also use network parameter objects to perform network parameter conversions. For more information, see “RF Network Parameter Objects” on page 2-11.

See Also

abcd2s | h2s | s2z | y2s | z2abcd | z2h | z2y

Introduced before R2006a

z2y

Convert Z-parameters to Y-parameters

Syntax

```
y_params = z2y(z_params)
```

Description

`y_params = z2y(z_params)` converts `z_params` into `y_params`.

Input Arguments

`z_params` — Impedance parameters
N-by-*N*-by-*M* complex array

Impedance parameters, specified as a *N*-by-*N*-by-*M* complex array representing *M* N-port Z-parameters.

Output Arguments

`y_params` — Admittance parameters
N-by-*N*-by-*M* complex array

Admittance parameters, returned as a *N*-by-*N*-by-*M* complex array representing *M* N-port Y-parameters.

Examples

Convert Z-parameters to Y-parameters

Define a matrix of Z-parameters.

```
Z11 = -14567.2412789287 - 148373.315116592i;
Z12 = -14588.1106171651 - 148388.583516562i;
Z21 = -14528.0522132692 - 148350.705757767i;
Z22 = -14548.5996561832 - 148363.457002006i;
z_params = [Z11,Z12; Z21,Z22];
```

Convert to Y-parameters.

```
y_params = z2y(z_params);
```

See Also

abcd2y | h2y | s2y | y2z | z2abcd | z2h | z2s

Introduced before R2006a

add

Add additional data to existing Smith chart

Syntax

```
add(plot,data)
add(plot,frequency,data)
```

Description

`add(plot,data)` adds data to an existing Smith chart.

`add(plot,frequency,data)` adds data to an existing Smith chart based on multiple data sets containing frequencies corresponding to columns of data matrix.

Examples

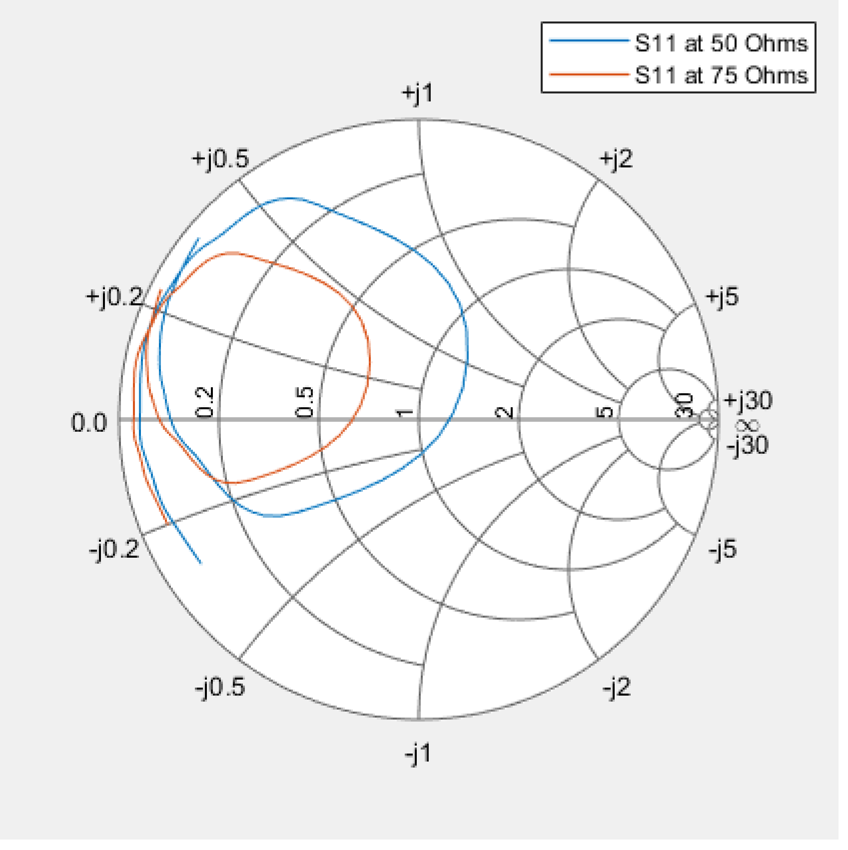
Add S-Parameter Data to Existing Smith Plot

Read S-parameter data.

```
amp = read(rfckt.amplifier,'default.s2p');
Sa = sparameters(amp);
figure
smithplot(Sa,[1,1])
```

Plot S-parameter object with new impedance of Z0 = 75 Ohms.

```
Sa = sparameters(Sa,75);
S11 = rfparam(Sa,1,1);
Freq = Sa.Frequencies;
s = smithplot('gco');
add(s, Freq, S11);
s.LegendLabels = {'S11 at 50 Ohms', 'S11 at 75 Ohms'};
```



Input Arguments

plot — Smith chart
function handle

Smith chart handle, specified as a function handle. If the handle of the Smith chart is not retained during creation, it is obtained by using the command `p = smithplot('gco')`.

Data Types: `double`

data — Input data
complex vector | complex matrix

Input data, specified as a complex vector or complex matrix.

For a matrix *D*, the columns of *D* are independent data sets. For *N*-by-*D* arrays, dimensions 2 and greater are independent data sets.

Data Types: `double`
Complex Number Support: Yes

`frequency` — Frequency data
real vector

Frequency data, specified as a real vector.

Data Types: `double`

See Also

`replace` | `smithplot`

Introduced in R2017b

replace

Remove current data and add new data to Smith chart

Syntax

```
replace(plot,data)
replace(plot,frequency,data)
```

Description

`replace(plot,data)` removes all current data and adds new data to the Smith chart.

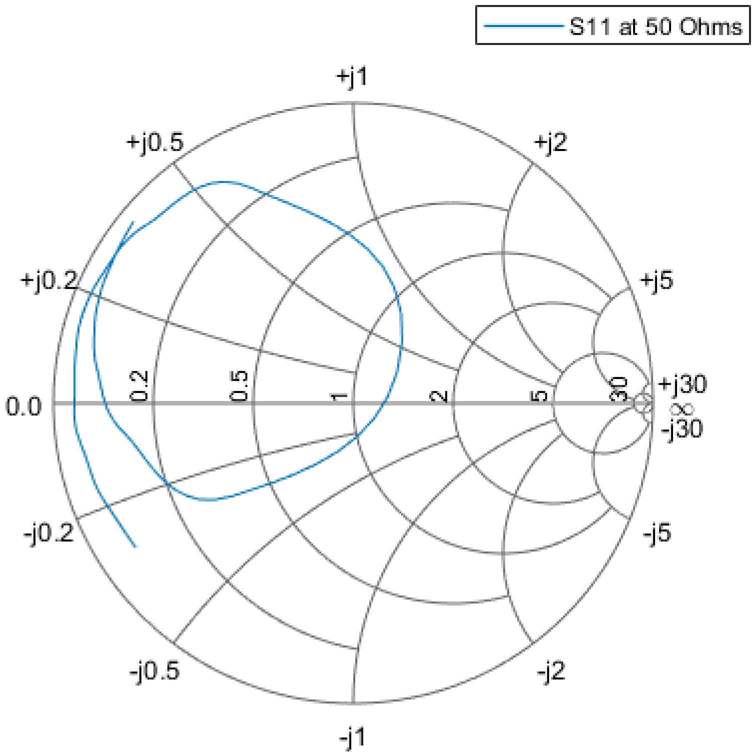
`replace(plot,frequency,data)` removes all current data and adds new data to the Smith chart based on multiple data sets containing frequencies corresponding to columns of the data matrix.

Examples

Replace S-Parameter Data on an existing Smith Plot

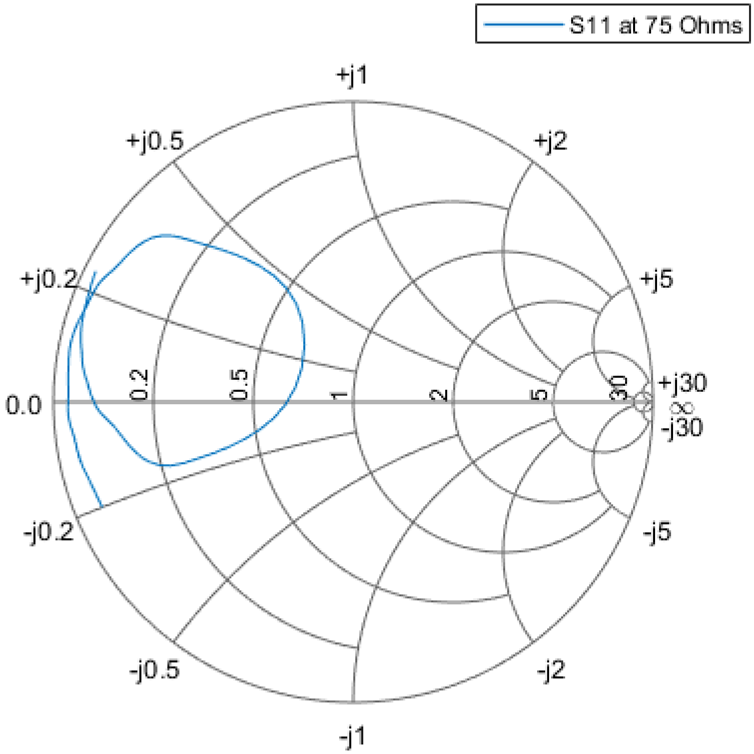
Read S-parameter data.

```
amp = read(rfckt.amplifier,'default.s2p');
Sa = sparameters(amp);
smithplot(Sa,[1,1],'LegendLabels','S11 at 50 Ohms');
```



Plot S-parameter object with a new impedance of Z0 = 75 Ohms.

```
Sa = sparameters(Sa,75);
S11 = rfparam(Sa,1,1);
Freq = Sa.Frequencies;
s = smithplot('gco');
replace(s, Freq, S11);
s.LegendLabels = 'S11 at 75 Ohms';
```



Input Arguments

plot — Smith plot
function handle

Smith chart handle, specified as a function handle. If the handle of the Smith chart is not retained during creation, it is obtained by using the command `p = smithplot('gco')`.

Data Types: `double`

data — Input data
complex vector | complex matrix

Input data, specified as a complex vector or complex matrix.

For a matrix *D*, the columns of *D* are independent datasets. For *N*-by-*D* arrays, dimensions 2 and greater are independent datasets.

Data Types: `double`
Complex Number Support: Yes

`frequency` — Frequency data
real vector

Frequency data, specified as a real vector.

Data Types: `double`

See Also

`add` | `smithplot`

Introduced in R2017b

smithplot

Plot of measurement data on Smith chart

Syntax

```
smithplot(data)
smithplot(frequency,data)
smithplot(ax,___)
smithplot(hnet)
smithplot(hnet,i,j)
smithplot(hnet,[i1,j1;i2,j2;....,in,jn])
s = smithplot(___)
s = smithplot('gco')
smithplot(___,Name,Value)
```

Description

`smithplot(data)` creates a Smith chart based on input data values.

Note The Smith chart is commonly used to display the relationship between a reflection coefficient, typically given as S11 or S22, and a normalized impedance

`smithplot(frequency,data)` creates a Smith chart based on frequency and data values.

`smithplot(ax,___)` creates a Smith chart with a user defined axes handle, `ax`, instead of the current axes handle. Axes handles are not supported for network parameter objects.

`smithplot(hnet)` plots all the network parameter objects in `hnet`.

`smithplot(hnet,i,j)` plots the *(i,j)*th parameter of `hnet`. `hnet` can be a network parameter, an rfckt, an rfdata, an nport, or an rfbudget object.

`smithplot(hnet,[i1,j1;i2,j2;....,in,jn])` plots multiple parameters (i_1,j_1,i_2,j_2) of `hnet`. `hnet` can be a network parameter, an rfckt, an rfdata, an nport, or an rfbudget object.

Note For rfbudget objects, smith plot is restricted to reflection coefficients.

`s = smithplot( ___ )` returns a Smith chart function handle so you can customize the plot and add measurements.

`s = smithplot('gco')` returns a Smith chart function handle of the current plot. This syntax is useful when the function handle, `p` was not returned or retained.

`smithplot( ___ ,Name,Value)` creates a Smith chart, with additional properties specified by one or more name-value pair arguments. `Name` is the property name and `Value` is the corresponding property value. You can specify several name-value pair arguments in any order as `Name1`, `Value1`, `...`, `NameN`, `ValueN`. Properties not specified retain their default values. To list all the property `Name,Value` pairs, use `details(p)`. You can use the properties to extract any data from the Smith chart. For example, `p = smithplot(data,'GridType','Z')` displays the impedance data grid on the Smith chart. You can also use the `smithplot` interactive menu to change the line and marker styles.

For a list of properties, see SmithPlot Properties.

Examples

Smithplot of S-parameters from a nport circuit element

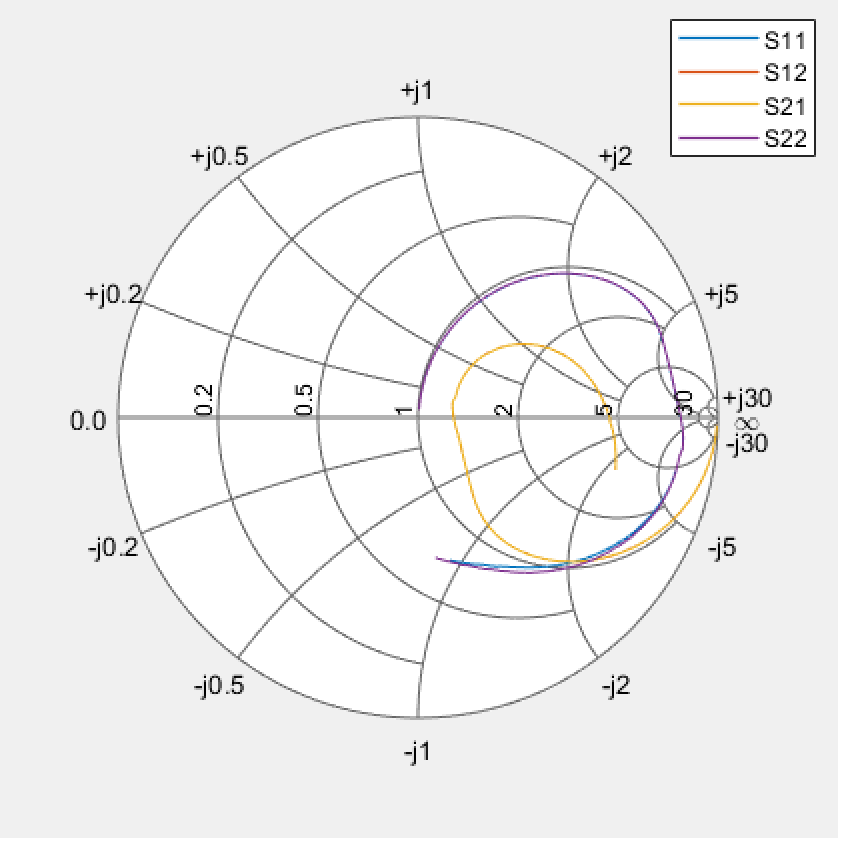
Plot the Smith plot of s-parameters data file, `passive.s2p`.

```
data = nport('passive.s2p')
```

```
data =
  nport: N-port element

    NetworkData: [1x1 sparameters]
           Name: 'Sparams'
       NumPorts: 2
```

```
    Terminals: {'p1+'  'p2+'  'p1-'  'p2-'}
```

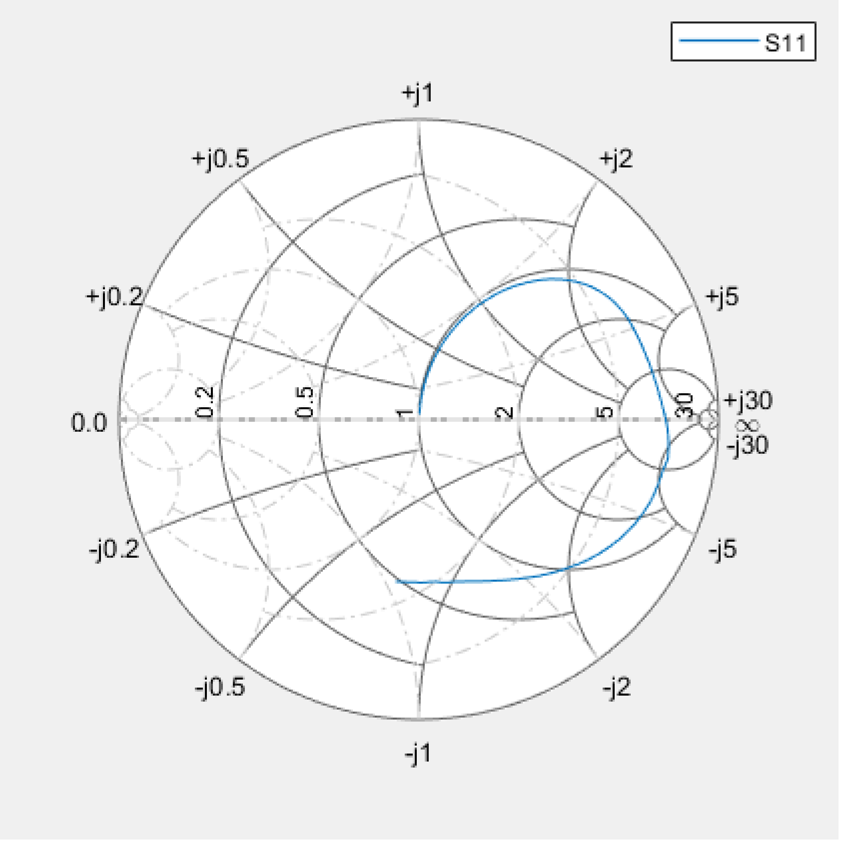
```
smithplot(data);
```



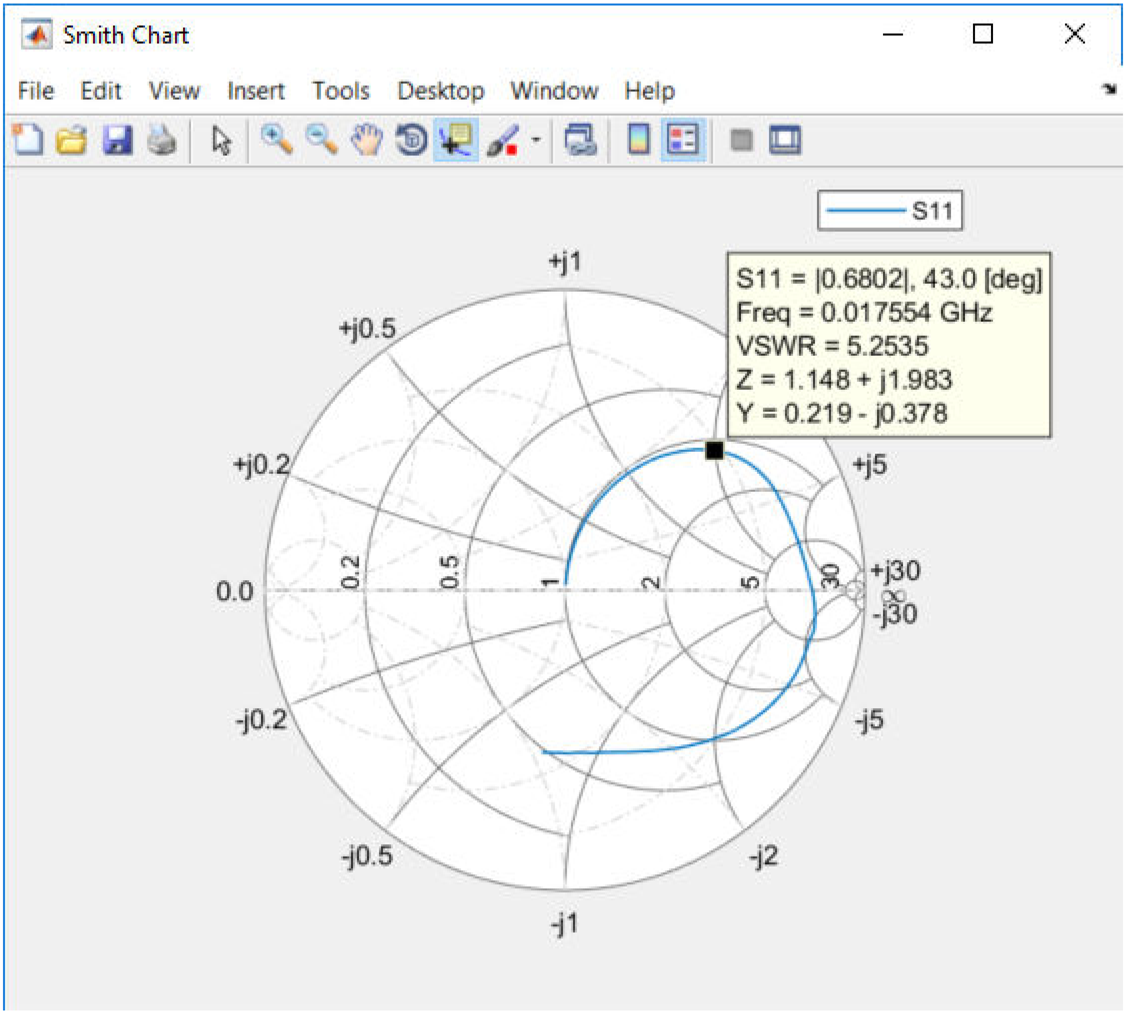
Smith Plot of (i,j)th Parameter of S-Parameter Data

Plot the Smith plot of S11 of s-parameter data file using an impedance of 75 ohms.

```
data = sparameters('passive.s2p' );
s = sparameters(data,75);
p = smithplot(s,1,1, 'GridType','ZY');
```



Use the data cursor icon in the toolbar to insert a cursor on your smith plot chart. You now know the S11, VSWR, Impedance, and frequency values at that cursor. For admittance value, change the Grid Type.

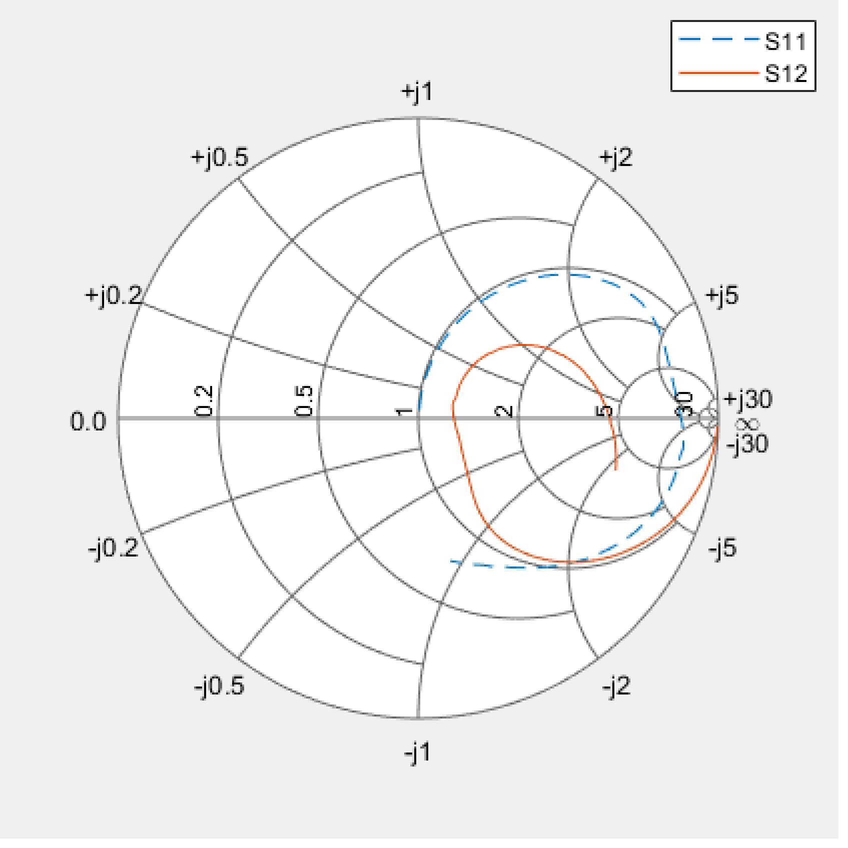


Smith Plot of rfckt Object

Plot the Smith chart of an rfckt.amplifier object.

```
S = read(rfckt.amplifier,'passive.s2p');
ports = [1,1;1,2];
```

```
s = smithplot(S,ports);
s.LineStyle = {'--', '-'};
```



Smith Plot Interactive Menu

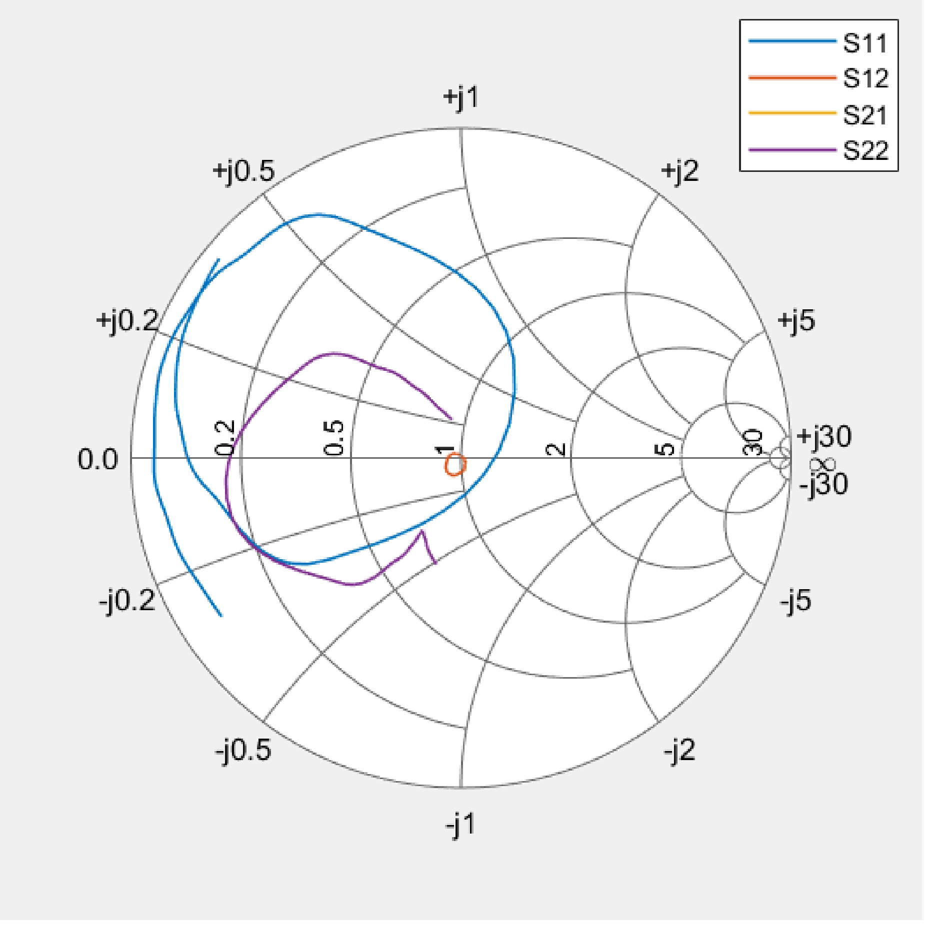
Use the Smith plot interactive menu for changing line and marker styles.

Plot the Smith plot of s-parameters data file, `default.s2p`.

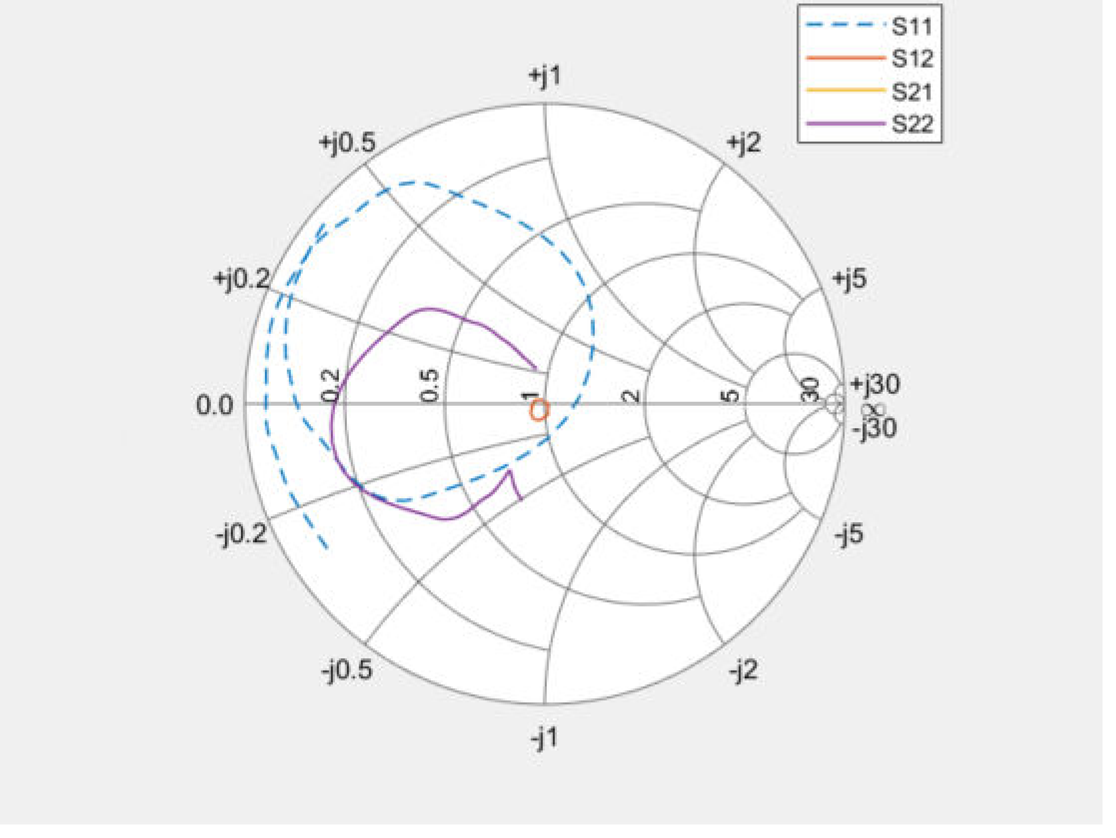
```
data = nport('default.s2p')
```

```
data = 
  nport: N-port element

    NetworkData: [1×1 sparameters]
           Name: 'Sparams'
       NumPorts: 2
      Terminals: {'p1+'  'p2+'  'p1-'  'p2-'}
```

```
smithplot(data)
```



Right click on the S11 line to reveal interactive menu, `DATASET 1.` Use LineStyle to change the style of S11 line on the Smith plot.



Input Arguments

data — Input data
complex vector | complex matrix

Input data, specified as a complex vector or complex matrix.

For a matrix *D*, the columns of *D* are independent data sets. For *N*-by-*D* arrays, dimensions 2 and greater are independent data sets.

Data Types: `double`
Complex Number Support: Yes

`frequency` — Frequency data
real vector

Frequency data, specified as a real vector.

Data Types: `double`

`hnet` — Input objects
RF Toolbox network parameter object | rfckt object | rfdata object | nport object | rfbudget object

Input objects, specified as one of the following:

- RF Toolbox network parameter object
- rfckt object
- rfdata object
- nport object
- rfbudget object.

Data Types: `double`

Output Arguments

`s` — Smith chart function handle
object

Smith chart function handle, returned as an object to customize the plot and add measurements using MATLAB commands.

See Also
`add` | `circle` | `replace`

Introduced in R2017b

9

AMP File Format

AMP File Data Sections

In this section...

Overview

The AMP data file describes a single nonlinear device. Its format can contain the following types of data:

- S, Y, or Z network parameters
- Noise parameters
- Noise figure data
- Power data
- IP3 data

An AMP file must contain either power data or network parameter data to be valid. To accommodate analysis at more than one frequency, the file can contain more than one section of power data. Noise data, noise figure data, and IP3 data are optional.

Note If the file contains both network parameter data and power data, RF Toolbox software checks the data for consistency. If the amplifier gain computed from the network parameters is not consistent with the gain computed from the power data, a warning appears.

Two AMP files, `samplepa1.amp` and `default.amp`, ship with the toolbox to show the AMP format. They describe a nonlinear 2-port amplifier with noise. See “Model a Cascaded RF Network” on page 1-10 for an example that shows how to use an AMP file.

Denoting Comments

An asterisk (*) or an exclamation point (`!`) precedes a comment that appears on a separate line.

A semicolon (`;`) precedes a comment that appears following data on the same line.

Data Sections

Each kind of data resides in its own section. Each section consists of a two-line header followed by lines of numeric data. Numeric values can be in any valid MATLAB format.

A new header indicates the end of the previous section. The data sections can appear in any order in the file.

Note In the data section descriptions, brackets (`[]`) indicate optional data or characters. All values are case insensitive.

S, Y, or Z Network Parameters

Header Line 1

The first line of the header has the format

```
Keyword [Parameter] [R[REF][=]value]
```

`Keyword` indicates the type of network parameter. Its value can be `S[PARAMETERS]`, `Y[PARAMETERS]`, or `Z[PARAMETERS]`. `Parameter` indicates the form of the data. Its value can be `MA`, `DB`, or `RI`. The default for S-parameters is `MA`. The default for Y- and Z-parameters is `RI`. `R[REF][=]value` is the reference impedance. The default reference impedance is 50 ohms.

The following table explains the meaning of the allowable `Parameter` values.

Parameter	Description
MA	Data is given in (magnitude, angle) pairs with angle in degrees (default for S-parameters).
DB	Data is given in (dB-magnitude, angle) pairs with angle in degrees.
RI	Data is given in (real, imaginary) pairs (default for Y- and Z-parameters).

This example of a first line indicates that the section contains S-parameter data given in (real, imaginary) pairs, and that the reference impedance is 50 ohms.

```
S RI R 50
```

Header Line 2

The second line of the header has the format

```
Independent_variable Units
```

The data in a section is a function of the `Independent_variable`. Currently, for S-, Y-, and Z-parameters, the value of Independent_variable is always `F[REQ]`. `Units` indicates the default units of the frequency data. It can be `GHz`, `MHz`, or `KHz`. You must specify `Units`, but you can override this default on any given line of data.

This example of a second line indicates that the default units for frequency data is GHz.

```
FREQ GHZ
```

Data

The data that follows the header typically consists of nine columns.

The first column contains the frequency points where network parameters are measured. They can appear in any order. If the frequency is given in units other than those you specified as the default, you must follow the value with the appropriate units; there should be no intervening spaces. For example,

```
FREQ GHZ
1000MHZ  ...
2000MHZ  ...
3000MHZ  ...
```

Columns two though nine contain 2-port network parameters in the order N11, N21, N12, N22. Similar to the Touchstone format, each Nnn corresponds to two consecutive columns of data in the chosen form: MA, DB, or RI. The data can be in any valid MATLAB format.

This example is derived from the file `default.amp`. A comment line explains the column arrangement of the data where `re` indicates real and `im` indicates imaginary.

```
S RI R 50
FREQ GHZ
* FREQ     reS11      imS11      reS21      imS21    reS12     imS12      reS22      imS22
  1.00  -0.724725  -0.481324  -0.685727  1.782660  0.000000  0.000000  -0.074122  -0.321568
  1.01  -0.731774  -0.471453  -0.655990  1.798041  0.001399  0.000463  -0.076091  -0.319025
  1.02  -0.738760  -0.461585  -0.626185  1.813092  0.002733  0.000887  -0.077999  -0.316488
```

Noise Parameters

Header Line 1

The first line of the header has the format

```
Keyword
```

`Keyword` must be `NOI[SE]`.

Header Line 2

The second line of the header has the format

```
Variable Units
```

`Variable` must be `F[REQ]`. `Units` indicates the default units of the frequency data. It can be `GHz`, `MHz`, or `KHz`. You can override this default on any given line of data. This example of a second line indicates that frequency data is assumed to be in GHz, unless other units are specified.

```
FREQ GHz
```

Data

The data that follows the header must consist of five columns.

The first column contains the frequency points at which noise parameters were measured. The frequency points can appear in any order. If the frequency is given in units other than those you specified as the default, you must follow the value with the appropriate units; there should be no intervening spaces. For example,

```
NOI
FREQ GHZ
1000MHZ  ...
2000MHZ  ...
3        ...
4        ...
5        ...
```

Columns two through five contain, in order,

- Minimum noise figure in decibels
- Magnitude of the source reflection coefficient to realize minimum noise figure
- Phase in degrees of the source reflection coefficient
- Effective noise resistance normalized to the reference impedance of the network parameters

This example is taken from the file `default.amp`. A comment line explains the column arrangement of the data.

```
NOI RN
FREQ GHz
* Freq  Fmin(dB)  GammmaOpt(MA:Mag) GammmaOpt(MA:Ang) RN/Zo
  1.90  10.200000 1.234000         -78.400000         0.240000
  1.93  12.300000 1.235000         -68.600000         0.340000
  2.06  13.100000 1.254000         -56.700000         0.440000
  2.08  13.500000 1.534000         -52.800000         0.540000
  2.10  13.900000 1.263000         -44.400000         0.640000
```

Noise Figure Data

The AMP file format supports the use of frequency-dependent noise figure (NF) data.

Header Line 1

The first line of the header has the format

```
Keyword [Units]
```

For noise figure data, `Keyword` must be `NF`. The optional `Units` field indicates the default units of the NF data. Its value must be `dB`, i.e., data must be given in decibels.

This example of a first line indicates that the section contains NF data, which is assumed to be in decibels.

```
NF
```

Header Line 2

The second line of the header has the format

```
Variable Units
```

`Variable` must be `F[REQ]`. `Units` indicates the default units of the frequency data. It can be `GHz`, `MHz`, or `KHz`. This example of a second line indicates that frequency data is assumed to be in GHz.

```
FREQ GHz
```

Data

The data that follows the header typically consists of two columns.

The first column contains the frequency points at which the NF data are measured. Frequency points can appear in any order. For example,

```
NF
FREQ MHz
2090  ...
2180  ...
2270  ...
```

Column two contains the corresponding NF data in decibels.

This example is derived from the file `samplepa1.amp`.

```
NF dB
FREQ GHz
1.900   10.3963213
2.000   12.8797965
2.100   14.0611765
2.200   13.2556751
2.300   12.9498642
2.400   13.3244309
2.500   12.7545104
```

Note If your noise figure data consists of a single scalar value with no associated frequency, that same value is used for all frequencies. Enter the value in column 1 of the line following header line 2. You must include the second line of the header, but it is ignored.

Power Data

An AMP file describes power data as input power-dependent output power.

Header Line 1

The first line of the header has the format

```
Keyword [Units]
```

For power data, `Keyword` must be `POUT`, indicating that this section contains power data. Because output power is complex, `Units` indicates the default units of the magnitude of the output power data. It can be `dBW`, `dBm`, `mW`, or `W`. The default is `W`. You can override this default on any given line of data.

The following table explains the meaning of the allowable `Units` values.

Allowable Power Data Units

Units	Description
dBW	Decibels referenced to one watt
dBm	Decibels referenced to one milliwatt
mW	Milliwatts
W	Watts

This example of a first line indicates that the section contains output power data whose magnitude is assumed to be in decibels referenced to one milliwatt, unless other units are specified.

```
POUT dBm
```

Header Line 2

The second line of the header has the format

```
Keyword [Units] FREQ[=]value
```

`Keyword` must be `PIN`. `Units` indicates the default units of the input power data. The default is `W`. You can override this default on any given line of data. `FREQ[=]value` is the frequency point at which the power is measured. The units of the frequency point must be specified explicitly using the abbreviations `GHz`, `MHz`, `kHz`, or `Hz`.

This example of a second line indicates that the section contains input power data that is assumed to be in decibels referenced to one milliwatt, unless other units are specified. It also indicates that the power data was measured at a frequency of 2.1E+009 Hz.

```
PIN dBm FREQ=2.1E+009Hz
```

Data

The data that follows the header typically consists of three columns:

- The first column contains input power data. The data can appear in any order.
- The second column contains the corresponding output power magnitude.
- The third column contains the output phase shift in degrees.

Note RF Toolbox software does not use the phase data directly. RF Blockset blocks use this data in conjunction with RF Toolbox software to create the AM/PM conversion table for the Equivalent Baseband library General Amplifier and General Mixer blocks.

If all phases are zero, you can omit the third column. If all phases are zero or omitted, the toolbox assumes that the small signal phase from the network parameter section of the file ($180*angle(S_{21}(f))/pi$) is the phase for all power levels.

In contrast, if one or more phases in the power data section are nonzero, the toolbox interpolates and extrapolates the data to determine the phase at all power levels. The small signal phase ($180*angle(S_{21}(f))/pi$) from the network parameter section is ignored.

Inconsistency between the power data and network parameter sections of the file may cause incorrect results. To avoid this outcome, verify that the following criteria must is met:

- The lowest input power value for which power data exists falls in the small signal (linear) region.
- In the power table for each frequency point `f`, the power gain and phase at the lowest input power value are equal to $20*log10(abs(S_{21}(f)))$ and $180*angle(S_{21}(f))/pi$, respectively, in the network parameter section.

If the power is given in units other than those you specified as the default, you must follow the value with the appropriate units. There should be no intervening spaces.

This example is derived from the file `default.amp`. A comment line explains the column arrangement of the data.

```
POUT dbm
PIN dBm FREQ = 2.10GHz
* Pin     Pout            Phase(degrees)
  0.0     19.28           0.0
  1.0     20.27           0.0
  2.0     21.26           0.0
```

Note The file can contain more than one section of power data, with each section corresponding to a different frequency value. When you analyze data from a file with multiple power data sections, power data is taken from the frequency point that is closest to the analysis frequency.

IP3 Data

An AMP file can include frequency-dependent, third-order input (IIP3) or output (OIP3) intercept points.

Header Line 1

The first line of the header has the format

```
Keyword [Units]
```

For IP3 data, `Keyword` can be either `IIP3` or `OIP3`, indicating that this section contains input IP3 data or output IP3 data. `Units` indicates the default units of the IP3 data. Valid values are `dBW`, `dBm`, `mW`, and `W`. The default is `W`.

This example of a first line indicates that the section contains input IP3 data which is assumed to be in decibels referenced to one milliwatt.

```
IIP3 dBm
```

Header Line 2

The second line of the header has the format

```
Variable Units
```

`Variable` must be `FREQ`. `Units` indicates the default units of the frequency data. Valid values are `GHz`, `MHz`, and `KHz`. This example of a second line indicates that frequency data is assumed to be in GHz.

```
FREQ GHz
```

Data

The data that follows the header typically consists of two columns.

The first column contains the frequency points at which the IP3 parameters are measured. Frequency points can appear in any order.

```
OIP3
FREQ GHz
2.010  ...
2.020  ...
2.030  ...
```

Column two contains the corresponding IP3 data.

This example is derived from the file `samplepa1.amp`.

```
OIP3 dBm
FREQ GHz
2.100   38.8730377
```

Note If your IP3 data consists of a single scalar value with no associated frequency, then that same value is used for all frequencies. Enter the value in column 1 of the line following header line 2. You must include the second line of the header, but the application ignores it.

Inconsistent Data Sections

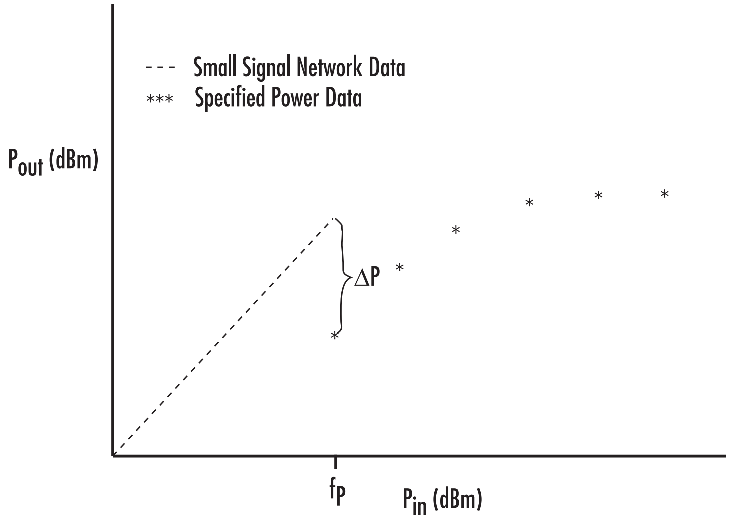
If an AMP file contains both network parameter data and power data, RF Toolbox software checks the data for consistency.

The toolbox compares the small-signal amplifier gain defined by the network parameters, S_{21}, and by the power data, $P_{out} - P_{in}$. The discrepancy between the two is computed in dBm using the following equation:

$$\Delta P = S_{21}(f_P) - P_{out}(f_P) + P_{in}(f_P)$$

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.

10

RF Online

App Reference

RF Budget Analyzer

Analyze gain, noise figure, and IP3 of cascaded RF elements and export to RF Blockset

Description

The **RF Budget Analyzer** app analyzes the gain, noise figure, and nonlinearity of a proposed RF system architecture.

Using this app, you can:

- Build a cascade of RF elements.
- Calculate the per-stage and cascade output power, gain, noise figure, SNR, and IP3 (third-order intercept) of the system.
- Plot rfbudget results across bandwidth and from stage to stage.
- Plot S-parameters of RF System on a Smith Chart and a Polar plot.
- Export per-stage and cascade values to the MATLAB workspace.
- Export the system design to RF Blockset for simulation.
- Export the system design to the RF Blockset Testbench as a DUT (device under test) subsystem and verify the results using simulation.

Available Blocks

The app toolstrip contains these blocks for creating an RF system:

- Amplifier
- Modulator
- S-parameters
- Generic

Available Templates

The app tool strip contains these templates for transmitter and receiver systems:

- Receiver template

- Transmitter template

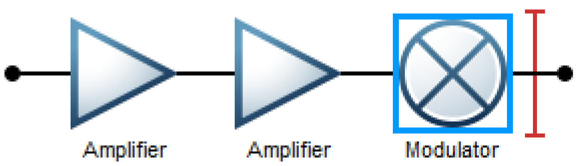
Open the RF Budget Analyzer App

- MATLAB Tool strip: On the **Apps** tab, under **Signal Processing and Communications**, click the app icon.
- MATLAB command prompt: Enter `rfBudgetAnalyzer`.

Examples

RF Budget Analyzer Design Canvas

The **RF Budget Analyzer** display canvas consists of two parts:



Stage	1	2	3
GainA (dB)	20	-10	0
NF (dB)	0	0	0
OIP3 (dBm)	Inf	Inf	Inf

Cascade	1	1..2	1..3
Fout (GHz)	2.1	2.1	4.2
Pout (dBm)	-10	-20	-20
GainT (dB)	20	10	10
NF (dB)	0	0	0
OIP3 (dBm)	Inf	Inf	Inf
SNR (dB)	73.98	73.98	73.98

For more information on the different types of gain, see [1].

1. **Stage: Individual Parameters of Each Element**

   - **GainA (dB)** — Available power gain
   - **NF (dB)** — Noise figure
   - **OIP3 (dBm)** — Output third-order intercept
2. **Cascade: Cumulative Parameters of Each Element**

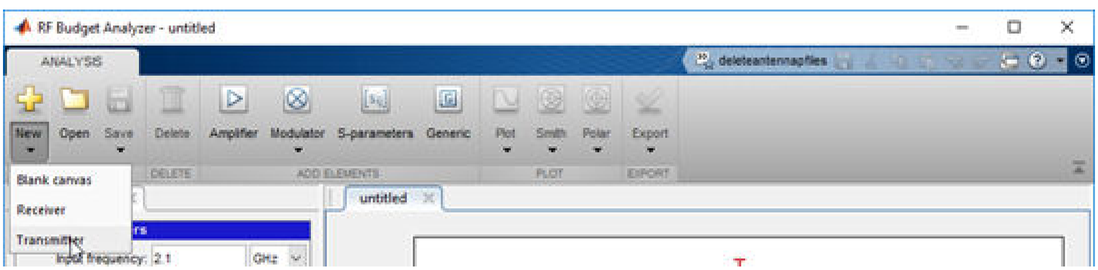
   - **Fout (GHz)** — Output frequency
   - **Pout (dBm)** — Output power
   - **GainT (dB)** — Transducer power gain
   - **NF (dB)** — Noise figure
   - **OIP3 (dBm)** — Output third-order intercept
   - **SNR (dB)** — Signal-to-noise ratio

RF Transmitter System Analysis

Design and analyze an RF transmitter using the **RF Budget Analyzer** app.

1. Open the app.

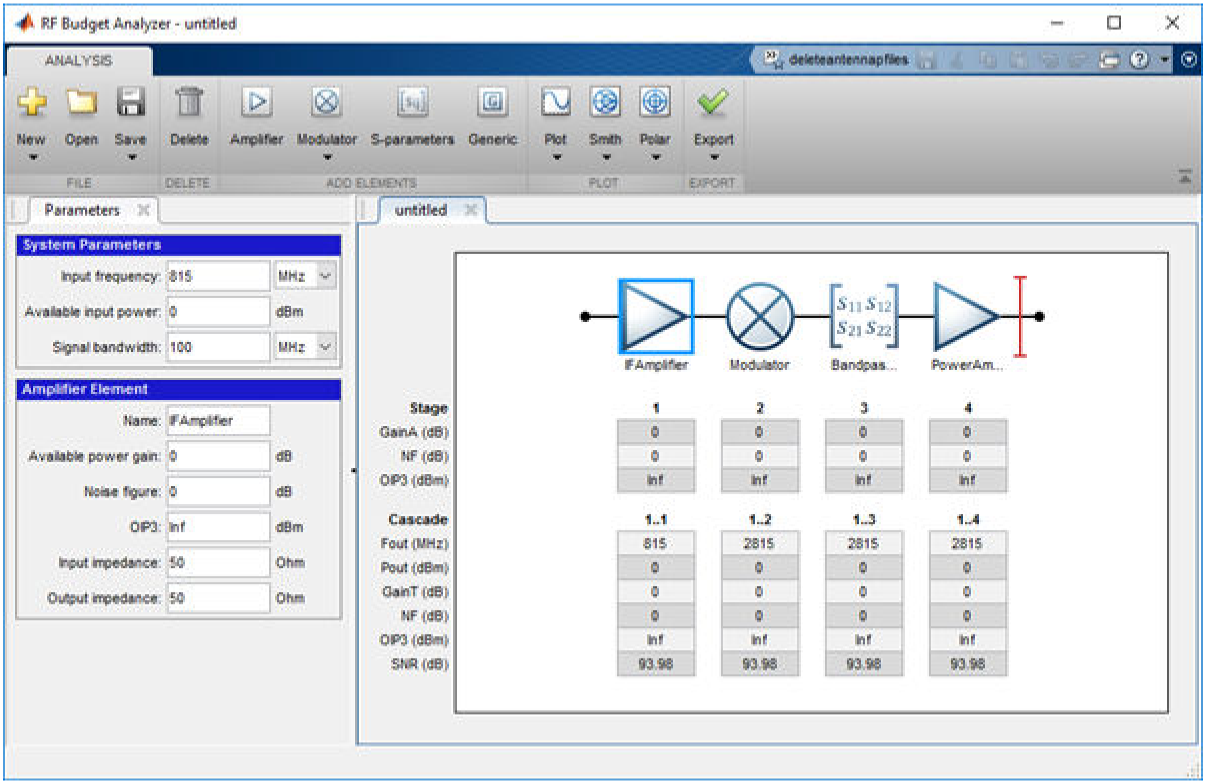
   ```
   rfBudgetAnalyzer
   ```
2. Use the transmitter template to create a basic transmitter.

3. In **System Parameters**, specify the requirements for the RF transmitter:

   - **Input frequency** — 815 MHz
   - **Available input power** — `0` dBm

- **Signal bandwidth** — `100` MHz



4 Click the IF Amplifier in the design canvas. Delete it using the  tool strip button.

5 Add a Generic block in place of the IF Amplifier using the  tool strip button. In **Element Parameters**, specify:

- **Name** — `IFFilter`
- **Available power gain** — `-3.6` dB

6 Click the Modulator block. In **Element Parameters**, specify:

- **Name** — `Mixer`
- **Available power gain** — `-6.5` dB

- **OIP3** — `11.5` dBm
- **LO frequency** — `4.97 GHz`
- **Converter type** — `Up`

7 Delete the S-Parameters block named BandpassFilter. Add a Generic block. In **Element Parameters**, specify:

- **Name** — `RFFilter1`
- **Available power gain** — `-1.4` dB

8 In the Power Amplifier block **Element Parameters**, specify:

- **Name** — `PowerAmplifier1`
- **Available power gain** — `20` dB
- **OIP3** — `43` dBm

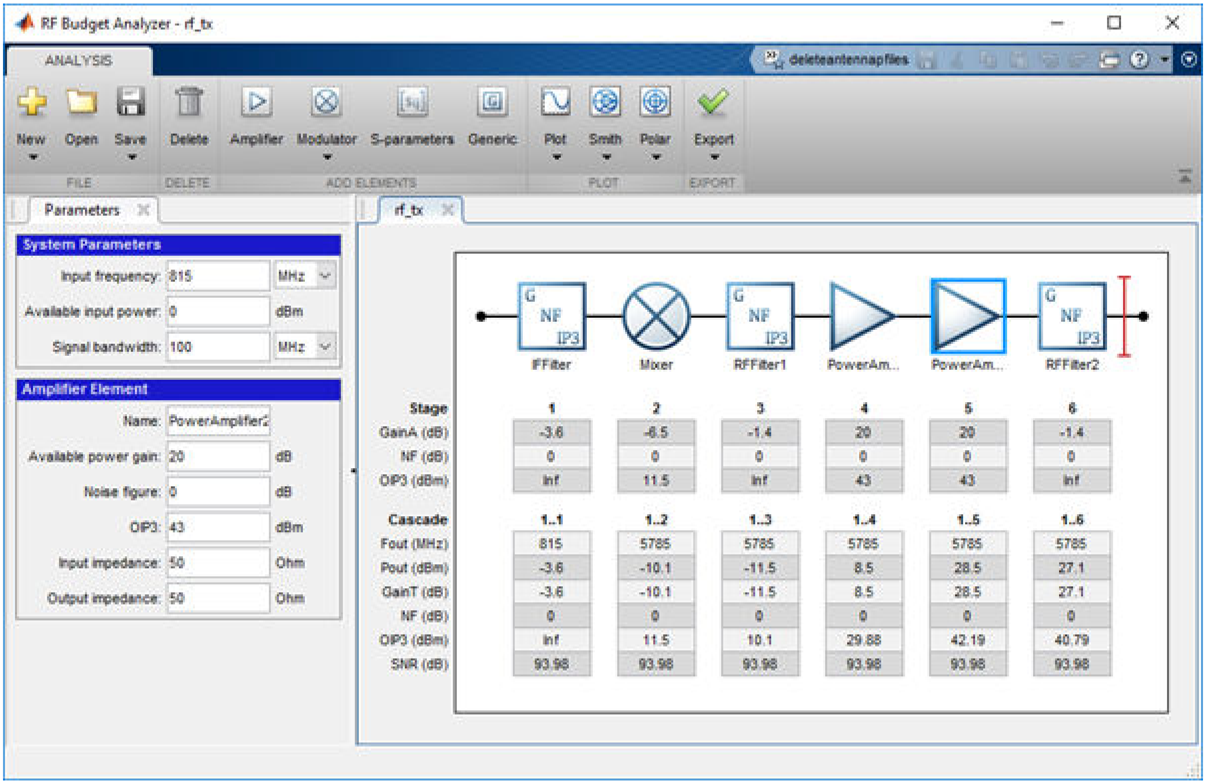
9 Add another Amplifier block using the  tool strip button. In **Element Parameters**, specify:

- **Name** — `PowerAmplifier2`
- **Available power gain** — `20` dB
- **OIP3** — `43` dBm

10 Add another Generic block. In **Element Parameters**, specify:

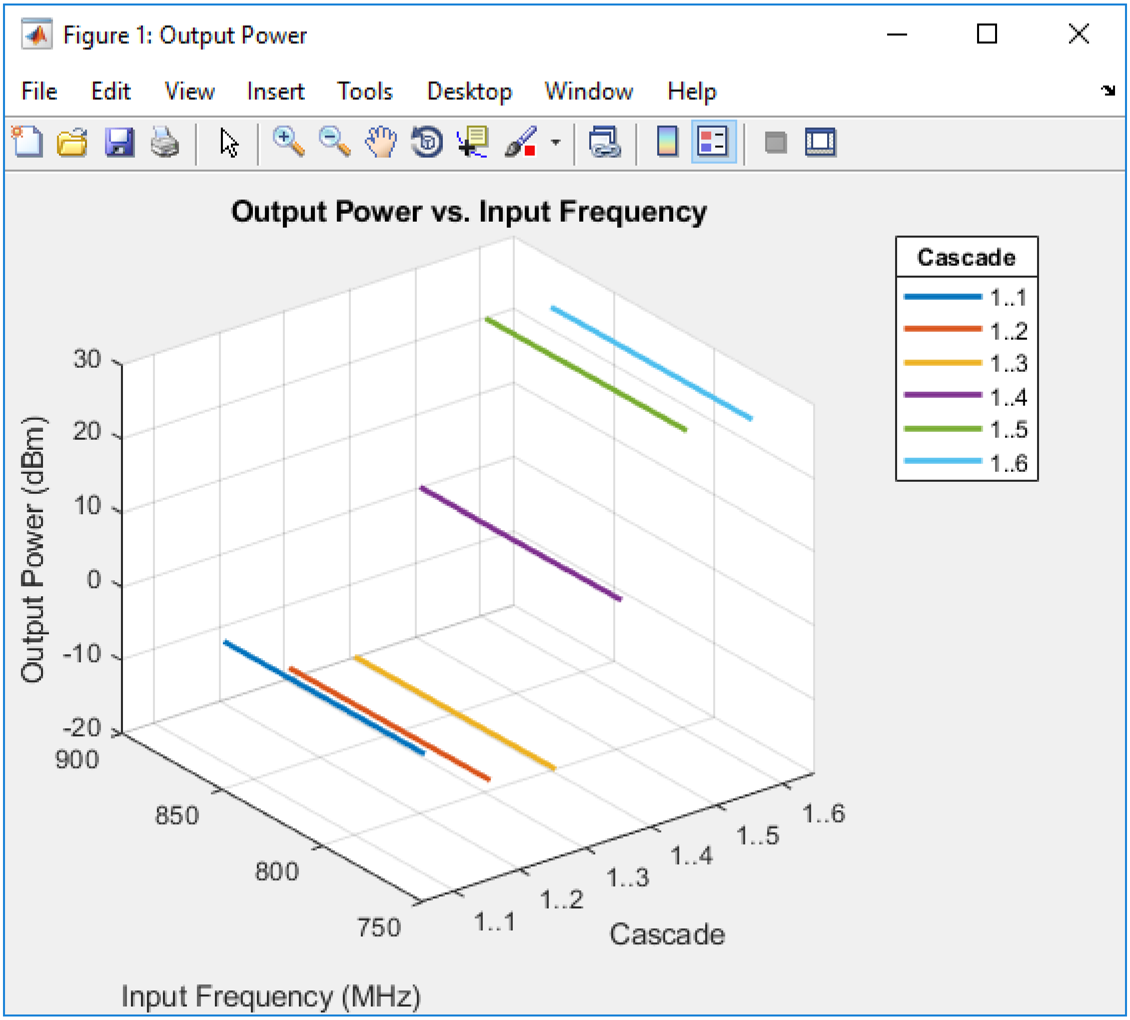
- **Name** — `RFFilter2`
- **Available power gain** — `-1.4` dB

11 Save the system. The app saves the system in a MAT file.



12

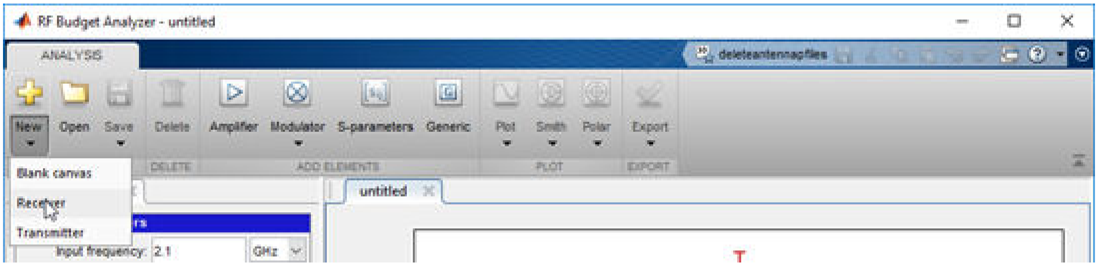
Plot the Output Power of the Transmitter analysis using the  button.



RF Receiver System Analysis

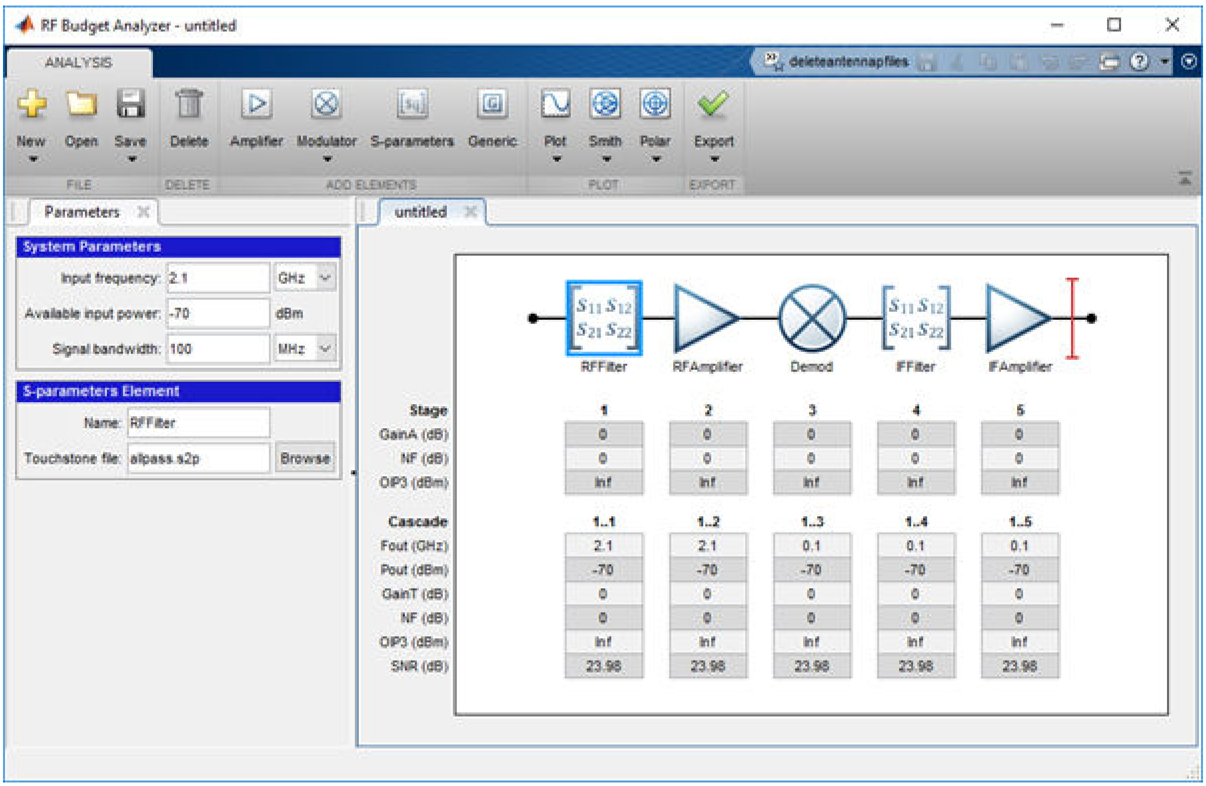
Design and analyze an RF receiver using the **RF Budget Analyzer** app.

1. Open the app.
2. Use the receiver template to create a basic receiver.



3 In **System Parameters**, specify the requirements for the RF receiver:

- **Input frequency** — `5.745` MHz
- **Available input power** — `-65` dBm
- **Signal bandwidth** — `100` MHz



4 Click RF Filter in the design region. This block is an S-parameters block. It accepts a Touchstone File in the `.s2p` format.

- **Name**: `BandpassFilter`
- **S2P file**: Choose an S2P file by clicking the **Browse**.

5 Click the RF Amplifier block. In **Element Parameters**, specify:

- **Name** — `LNA1`
- **Available power gain** — 12 dB
- **OIP3** — `20` dBm

6 Add another Amplifier block using the  tool strip button. In **Element Parameters**, specify:

- **Name** — `LNA2`
- **Available power gain** — 12 dB
- **OIP3** — `20` dBm

7 Add a Generic block. In **Element Parameters**, specify the block requirements:

- **Name** — `IRFilter`
- **Available power gain** — `-4.05` dB

8 Click the Demod block **Element Parameters**, specify:

- **Name** — `Mixer`
- **Available power gain** — `-6.5` dB
- **OIP3** — `11.5` dBm
- **LO frequency** — `4.93 GHz`
- **Converter type** — `Down`

9 Delete the S-parameters block. Add a Generic block in its place. In **Element Parameters**, specify:

- **Name** — `CSFilter`
- **Available power gain** — `-9.55` dB

10 Click the IF Amplifier block. In the **Element Parameters**, specify:

- **Name** — `PowerAmp1`
- **Available power gain** — 16 dB

- **OIP3** — `26` dBm

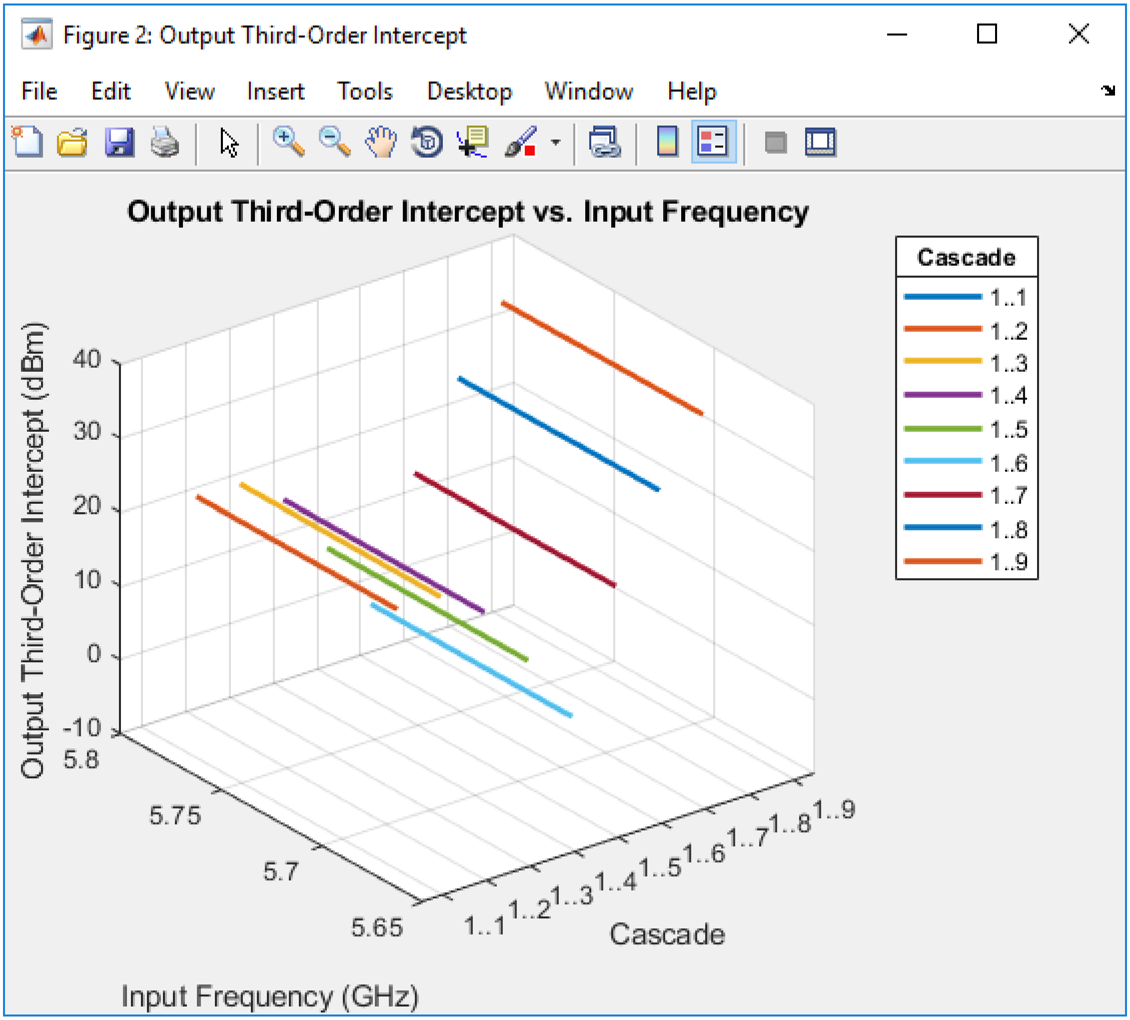
11 Add two more Amplifier blocks. For each block, **Element Parameters** specify:

- **Name** — `PowerAmp2` | `PowerAmp3` respectively.
- **Available power gain** — `16` dB | `20` dB
- **OIP3** — `26` dBm | `33` dBm

12 Save the system. The app saves the system in a MAT file.

13 Plot the Output OIP3 of the Receiver .using the 

button.



- Superheterodyne Receiver Using RF Budget Analyzer App

Parameters

System Parameters

`Input frequency` — Carrier frequency
2.1 `GHz` (default) | scalar

Carrier frequency of the RF system, specified as a scalar in: `Hz`, `kHz`, `MHz`, or `GHz`.

Note RF Budget Analyzer accepts 0 Hz as input frequency of a system.

`Available input power` — Available Input power
-30 (default) | scalar in dBm

Available input power to the RF system, specified as a scalar in dBm.

`Signal bandwidth` — Bandwidth of input signal
10 `MHz` (default) | scalar

Bandwidth of input signal, specified as a scalar in: `Hz`, `kHz`, `MHz`, or `GHz`.

Element Parameters

`Name` — Name of element
character vector

Name of the element added to the RF System, specified as a character vector.

`Touchstone file` — Touchstone data file
`allpass.s2p` (default) | character vector

Touchstone data file, specified as a character vector, containing network parameter data. You can use only `.s2p` Touchstone files.

`Available power gain` — Available power gain
0 (default) | scalar

Available power gain added of the element, specified as a scalar.

`Noise Figure` — Degradation of signal-to-noise ratio
0 (default) | scalar in dB

Degradation of signal-to-noise ratio by the element, specified as a scalar in dB.

OIP3 — Output third-order intercept
inf (default) | scalar in dBm

Output third-order intercept of the element, specified as a scalar in dBm.

Input Impedance — Input impedance
50 (default) | scalar in Ohm

Input impedance of the element, specified as a scalar in Ohm.

Output Impedance — Output impedance
50 (default) | scalar in Ohm

Output impedance of the element, specified as a scalar in Ohm.

LO frequency — Local oscillator frequency of modulator
2.1 GHz (default) | scalar

Local oscillator frequency of Modulator element, specified as a scalar. Frequency units are the following: Hz, kHz, MHz, or GHz. This option is available when you choose the Modulator tool strip button.

Note RF Budget Analyzer do not accept 0 Hz as input frequency for down conversion.

Converter Type — Conversion type of modulator
Up (default) | Down

Conversion type of Modulator element, specified as Up or Down. This option is available when you choose the Modulator tool strip button.

Programmatic Use

`rfBudgetAnalyzer` opens the RF Budget Analyzer app to analyze the per-stage and total gain, noise figure, and nonlinearity (IP3) of an RF system.

`rfBudgetAnalyzer(rfsystem)` opens an RF system saved using the RF Budget Analyzer app. `rfsystem` is a MAT file.

References

[1] M. Pozar, David. “Microwave Amplifier Design.” *Microwave Engineering*. Hoboken, NJ: John Wiley & Sons, Inc. 4th Edition. 2012, p. 559

See Also

Topics

Superheterodyne Receiver Using RF Budget Analyzer App
“Using RF Measurement Testbench” on page 1-25

Introduced in R2016a

12

Properties

SmithPlot Properties

Control appearance and behavior of Smith chart

Description

Smith chart properties control the appearance and behavior of the Smith plot object. By changing property values, you can modify certain aspects of the Smith chart. To change the default properties use: .

```
s = smithplot(____,Name,Value)
```

To view all the properties of the Smith plot object use:

```
details(s)
```

Properties

Display

`ClipData` — Clip data to outer circle
`1` (default) | `0`

Clip data to outer circle, specified as `0` or `1`.

Data Types: `logical`

`ColorOrder` — Colors to use for multiline plots
seven predefined colors (default) | three-column matrix of RGB triplets

Colors to use for multi-line plots, specified as three-column matrix of RGB triplets. Each row of the matrix defines one color in the color order.

Data Types: `double`

`ColorOrderIndex` — Next color to use in color order
`1` (default) | positive integer

Next color to use in color order, specified as a positive integer. New plots added to the axes use colors based on the current value of the color order index.

Data Types: `double`

FontName — Font name

`'Helvetica'` (default) | character vector

Font name, specified as a character vector.

Data Types: `char`

FontSize — Font size

`10` (default) | positive integer

Font size, specified as a positive integer.

Data Types: `double`

FontSizeMode — Changes the font size based on window size

`'auto'` (default) | 'manual'

Font size mode, specified as `'auto'`. Changes the font size based on window size.

Data Types: `char`

GridBackgroundColor — Background grid line color

`'w'` (default) | character vector of color names | `'none'`

Background gird line color, specified as an RGB triplet, or as a character vector of color names, or `'none'`. Using `'none'` turns off the grid completely.

An RGB triplet is a three-element row vector whose elements specify the intensities of the red, green, and blue components of the color. The intensities must be in the range `[0,1]`; for example, `[0.4 0.6 0.7]`. Alternatively, you can specify some common colors by name. This table lists the long and short color name options and the equivalent RGB triplet values.

Option	Description	Equivalent RGB Triplet
`'red'` or `'r'`	Red	`[1 0 0]`
`'green'` or `'g'`	Green	`[0 1 0]`
`'blue'` or `'b'`	Blue	`[0 0 1]`

Option	Description	Equivalent RGB Triplet
'yellow' or 'y'	Yellow	[1 1 0]
'magenta' or 'm'	Magenta	[1 0 1]
'cyan' or 'c'	Cyan	[0 1 1]
'white' or 'w'	White	[1 1 1]
'black' or 'k'	Black	[0 0 0]

Data Types: char | double

GridForegroundColor — Foreground grid line color
[0.4000 0.4000 0.4000] (default) | 'none' | character vector of color names

Foreground grid line color, specified as RGB triplet, or as a character vector of color names, or 'none'.

An RGB triplet is a three-element row vector whose elements specify the intensities of the red, green, and blue components of the color. The intensities must be in the range [0,1]; for example, [0.4 0.6 0.7]. Alternatively, you can specify some common colors by name. This table lists the long and short color name options and the equivalent RGB triplet values.

Option	Description	Equivalent RGB Triplet
'red' or 'r'	Red	[1 0 0]
'green' or 'g'	Green	[0 1 0]
'blue' or 'b'	Blue	[0 0 1]
'yellow' or 'y'	Yellow	[1 1 0]
'magenta' or 'm'	Magenta	[1 0 1]
'cyan' or 'c'	Cyan	[0 1 1]
'white' or 'w'	White	[1 1 1]
'black' or 'k'	Black	[0 0 0]

Data Types: double | char

GridLineStyle — Grid line style
'-' (default) | '--' | ':' | '-.' | 'none'

Grid line style, specified as one of the following:

Line Style	Description	Resulting Line
'-'	Solid line	———————
'--'	Dashed line	- - - - - - - - -
':'	Dotted line	....
'-.'	Dash-dotted line	-.-.-.
'none'	No line	No line

Data Types: `char`

`GridLineWidth` — Grid line width

`'0.5000'` (default) | positive scalar

Grid line width, specified as positive scalar.

Data Types: `double`

`GridOverData` — Draw grid over data plots

`0` (default) | `1`

Draw grid over data plots, specified as `0` or `1`.

Data Types: `logical`

`GridSubForegroundColor` — Sub-foreground grid lines color

`[0.8000 0.8000 0.8000]` (default) | `'none'` | character vector of color names

Sub foreground grid lines color, specified as an RGB triplet, character vector of color names, or `'none'`.

An RGB triplet is a three-element row vector whose elements specify the intensities of the red, green, and blue components of the color. The intensities must be in the range `[0,1]`; for example, `[0.4 0.6 0.7]`. Alternatively, you can specify some common colors by name. This table lists the long and short color name options and the equivalent RGB triplet values.

Option	Description	Equivalent RGB Triplet
`'red'` or `'r'`	Red	`[1 0 0]`
`'green'` or `'g'`	Green	`[0 1 0]`
`'blue'` or `'b'`	Blue	`[0 0 1]`

Option	Description	Equivalent RGB Triplet
'yellow' or 'y'	Yellow	[1 1 0]
'magenta' or 'm'	Magenta	[1 0 1]
'cyan' or 'c'	Cyan	[0 1 1]
'white' or 'w'	White	[1 1 1]
'black' or 'k'	Black	[0 0 0]

Data Types: char | double

GridSubLineStyle — Subgrid line style
'- .' (default) | '--' | ':' | '-' | 'none'

Subgrids line style, specified as one of the following:

Line Style	Description	Resulting Line
'-'	Solid line	-------
'--'	Dashed line	- - - - - - - - -
':'	Dotted line	....
'-.'	Dash-dotted line	-.-.-.
'none'	No line	No line

Data Types: char

GridSubLineWidth — Subgrid line width
'0.5000' (default) | positive scalar

Subgrid line width, specified as positive scalar.

Data Types: double

GridType — Grid type
'Z' (default) | 'Y' | 'ZY' | 'YZ'

Grid type, specified as 'Z', 'Y', 'ZY', 'YZ'. Grid type specifies if the plot is an admittance plot, impedance plot, or both.

Data Types: char

`GridValue` — Defines constant resistance circles and constant reactance arcs

`[30.0 5.0 2.0 1.0 0.5 0.2; Inf 30.0 5.0 5.0 2.0 1.0]` (default)

Two-row matrix. Row 1 specifies the values of the constant resistance circles and constant reactance arcs in the chart. Row 2 specifies the value at which the corresponding arcs and circles defined in Row 1 end.

Data Types: `double`

`GridVisible` — Show grid on Smith chart

`'1'` (default) | `'0'`

Show grid on Smith chart, specified as `'1'` or `'0'`.

Data Types: `logical`

`NextPlot` — Directive on how to add next plot

`'replace'` (default) | `'new'` | `'add'`

Directive on how to add next plot, specified as a comma-separated pair consisting of `'NextPlot'` and one of the values in the table:

Property Value	Effect
`'new'`	Creates a figure and uses it as the current figure.
`'add'`	Adds new graphics objects without clearing or resetting the current figure.
`'replace'`	Removes all axes objects and resets figure properties to their defaults before adding new graphics objects.

`Parent` — Figure parent

root object

Figure parent, returned as a root object.

`TitleBottom` — Title to display below Smith chart

character vector

Title to display below the Smith chart, specified as a character vector.

Data Types: `char`

`TitleBottomFontSizeMultiplier` — Bottom title font scale factor

`0.9000` (default) | numeric value greater than zero

Bottom title font scale factor, specified as a numeric value greater than zero.

Data Types: `double`

`TitleBottomFontWeight` — Bottom title font thickness

`'normal'` (default) | `'bold'`

Bottom title font thickness, specified as `'bold'` or `'normal`.

Data Types: `char`

`TitleBottomOffset` — Offset between bottom title and arc ticks

`0.1500` (default) | scalar

Offset between bottom title and angle ticks,specified as a scalar. The value must be in the range [`-0.5`,`0.5`].

Data Types: `double`

`TitleBottomTextInterpreter` — Interpretation of bottom title characters

`'none'` (default) | `'tex'` | `'latex'`

Interpretation of bottom title characters, specified one of the following:

- `'tex'` — Interpret using a subset of TeX markup
- `'latex'` — Interpret using LaTeX markup
- `'none'` — Display literal characters

TeX Markup

By default, MATLAB supports a subset of TeX markup. Use TeX markup to add superscripts and subscripts, modify the text type and color, and include special characters in the text.

This table lists the supported modifiers when the `TickLabelInterpreter` property is set to `'tex'`, which is the default value. Modifiers remain in effect until the end of the text, except for superscripts and subscripts which only modify the next character or the text within the curly braces `{}`.

Modifier	Description	Example
`^{ }`	Superscript	`'text^{superscript}'`
`_{ }`	Subscript	`'text_{subscript}'`
`\bf`	Bold font	`'\bf text'`
`\it`	Italic font	`'\it text'`
`\sl`	Oblique font (rarely available)	`'\sl text'`
`\rm`	Normal font	`'\rm text'`
`\fontname{specifier}`	Set `specifier` as the name of a font family to change the font style. You can use this markup with other modifiers.	`'\fontname{Courier} text'`
`\fontsize{specifier}`	Set `specifier` as a scalar numeric value to change the font size.	`'\fontsize{15} text'`
`\color{specifier}`	Set `specifer` as one of these colors: `red`, `green`, `yellow`, `magenta`, `blue`, `black`, `white`, `gray`, `darkGreen`, `orange`, or `lightBlue`.	`'\color{magenta} text'`
`\color[rgb]{specifier}`	Set `specifier` as a three-element RGB triplet to change the font color.	`'\color[rgb]{0,0.5,0.5} text'`

LaTeX Markup

To use LaTeX markup, set the `TickLabelInterpreter` property to `'latex'`. The displayed text uses the default LaTeX font style. To change the font style, use LaTeX markup within the text.

The maximum size of the text that you can use with the LaTeX interpreter is 1200 characters. For multiline text, the maximum size reduces by about 10 characters per line.

For more information about the LaTeX system, see The LaTeX Project website at http://www.latex-project.org/.

Data Types: `char`

`TitleTop` — Title to display above the Smith chart
character vector

Title to display above the Smith chart, specified as a character vector.

Data Types: `char`

`TitleTopFontSizeMultiplier` — Top title font scale factor
`1.1000` (default) | numeric value greater than zero

Top title font scale factor, specified as a numeric value greater than zero.

Data Types: `double`

`TitleTopFontWeight` — Top title font thickness
`'bold'` (default) | `'normal'`

Top title font thickness, specified as `'bold'` or `'normal`.

Data Types: `char`

`TitleTopOffset` — Offset between top title and arc ticks
`0.1500` (default) | scalar

Offset between top title and angle ticks, specified as a scalar. The value must be in the range [`-0.5`,`0.5`].

Data Types: `double`

`TitleTopTextInterpreter` — Interpretor of top title characters
`'none'` (default) | `'tex'` | `'latex'`

Interpretation of top title characters, specified one of the following:

- `'tex'` — Interpret using a subset of TeX markup
- `'latex'` — Interpret using LaTeX markup
- `'none'` — Display literal characters

TeX Markup

By default, MATLAB supports a subset of TeX markup. Use TeX markup to add superscripts and subscripts, modify the text type and color, and include special characters in the text.

This table lists the supported modifiers when the `TickLabelInterpreter` property is set to `'tex'`, which is the default value. Modifiers remain in effect until the end of the text, except for superscripts and subscripts which only modify the next character or the text within the curly braces `{}`.

Modifier	Description	Example
`^{ }`	Superscript	`'text^{superscript}'`
`_{ }`	Subscript	`'text_{subscript}'`
`\bf`	Bold font	`'\bf text'`
`\it`	Italic font	`'\it text'`
`\sl`	Oblique font (rarely available)	`'\sl text'`
`\rm`	Normal font	`'\rm text'`
`\fontname{specifier}`	Set `specifier` as the name of a font family to change the font style. You can use this markup with other modifiers.	`'\fontname{Courier} text'`
`\fontsize{specifier}`	Set `specifier` as a scalar numeric value to change the font size.	`'\fontsize{15} text'`
`\color{specifier}`	Set `specifer` as one of these colors: `red`, `green`, `yellow`, `magenta`, `blue`, `black`, `white`, `gray`, `darkGreen`, `orange`, or `lightBlue`.	`'\color{magenta} text'`
`\color[rgb]{specifier}`	Set `specifier` as a three-element RGB triplet to change the font color.	`'\color[rgb]{0,0.5,0.5} text'`

LaTeX Markup

To use LaTeX markup, set the `TickLabelInterpreter` property to `'latex'`. The displayed text uses the default LaTeX font style. To change the font style, use LaTeX markup within the text.

The maximum size of the text that you can use with the LaTeX interpreter is 1200 characters. For multi-line text, the maximum size reduces by about 10 characters per line.

For more information about the LaTeX system, see The LaTeX Project website at http://www.latex-project.org/.

Data Types: `char`

Datasets

`EdgeColor` — Data line color

`'k'` (default) | RGB triplet vector

Data line color, specified as a character vector of color names or as an RGB triplet vector.

An RGB triplet is a three-element row vector whose elements specify the intensities of the red, green, and blue components of the color. The intensities must be in the range `[0,1]`; for example, `[0.4 0.6 0.7]`. Alternatively, you can specify some common colors by name. This table lists the long and short color name options and the equivalent RGB triplet values.

Option	Description	Equivalent RGB Triplet
`'red'` or `'r'`	Red	`[1 0 0]`
`'green'` or `'g'`	Green	`[0 1 0]`
`'blue'` or `'b'`	Blue	`[0 0 1]`
`'yellow'` or `'y'`	Yellow	`[1 1 0]`
`'magenta'` or `'m'`	Magenta	`[1 0 1]`
`'cyan'` or `'c'`	Cyan	`[0 1 1]`
`'white'` or `'w'`	White	`[1 1 1]`
`'black'` or `'k'`	Black	`[0 0 0]`

Data Types: `double` | `char`

`LegendLabels` — Data tables for legend annotation
character vector | cell array of character vectors

Data tables for legend annotation, specified as a character vector or as a cell array of character vectors.

Data Types: `char`

`LegendVisible` — Show legend label
`1` (default) | `0`

Show legend label, specified as `0` or `1`.

Data Types: `logical`

`LineStyle` — Plot line style
`'-'` (default) | `'--'` | `':'` | `'-.'` | `'none'`

Plot line style, specified as one of the symbols in the table:

Symbol	Line Style	Resulting Line
`'-'`	Solid line	
`'--'`	Dashed line	
`':'`	Dotted line	
`'-.'`	Dash-dotted line	
`'none'`	No line	No line

`LineWidth` — Plot line width
`1` (default) | positive scalar

Plot line width, specified as a positive scalar.

`Marker` — Marker symbol
`'none'` (default) | character vector of symbols

Marker symbol, specified as `'none'` or one of the symbols in this table. By default, a line does not have markers. Add markers at selected points along the line by specifying a marker.

Value	Description
'o'	Circle
'+'	Plus sign
'*'	Asterisk
'.'	Point
'x'	Cross
'square' or 's'	Square
'diamond' or 'd'	Diamond
'^'	Upward-pointing triangle
'v'	Downward-pointing triangle
'>'	Right-pointing triangle
'<'	Left-pointing triangle
'pentagram' or 'p'	Five-pointed star (pentagram)
'hexagram' or 'h'	Six-pointed star (hexagram)
'none'	No markers

MarkerSize — Marker size
6 (default) | positive value

Marker size, specified as a positive value in points.

Data Types: double

Arcs

ArcFontSizeMultiplier — Arc tick font scale factor
1 (default) | numeric value greater than zero

Arc tick font scale factor, specified as a numeric value greater than zero.

Data Types: double

ArcTickLabelColor — Arc tick labels
'k' (default) | RGB triplet vector

Arc tick labels, specified as a character vector of color names or as an RGB triplet vector.

An RGB triplet is a three-element row vector whose elements specify the intensities of the red, green, and blue components of the color. The intensities must be in the range `[0,1]`; for example, `[0.4 0.6 0.7]`. Alternatively, you can specify some common colors by name. This table lists the long and short color name options and the equivalent RGB triplet values.

Option	Description	Equivalent RGB Triplet
`'red'` or `'r'`	Red	`[1 0 0]`
`'green'` or `'g'`	Green	`[0 1 0]`
`'blue'` or `'b'`	Blue	`[0 0 1]`
`'yellow'` or `'y'`	Yellow	`[1 1 0]`
`'magenta'` or `'m'`	Magenta	`[1 0 1]`
`'cyan'` or `'c'`	Cyan	`[0 1 1]`
`'white'` or `'w'`	White	`[1 1 1]`
`'black'` or `'k'`	Black	`[0 0 0]`

Data Types: `char` | `double`

`ArcTickLabelVisible` — Show arc tick labels
`1` (default) | `0`

Show arc tick labels, specified as `0` or `1`.

Data Types: `logical`

Circles

`CircleFontSizeMultiplier` — Circle tick font scale factor
`0.9000` (default) | numeric value greater than zero

Circle tick font scale factor, specified as a numeric value greater than zero.

Data Types: `double`

`CircleTickLabelColor` — Circle tick label color
`'k'` (default) | RGB triplet vector

Circle tick labels color, specified as a character vector of color names or as an RGB triplet vector.

An RGB triplet is a three-element row vector whose elements specify the intensities of the red, green, and blue components of the color. The intensities must be in the range `[0,1]`; for example, `[0.4 0.6 0.7]`. Alternatively, you can specify some common colors by name. This table lists the long and short color name options and the equivalent RGB triplet values.

Option	Description	Equivalent RGB Triplet
'red' or 'r'	Red	[1 0 0]
'green' or 'g'	Green	[0 1 0]
'blue' or 'b'	Blue	[0 0 1]
'yellow' or 'y'	Yellow	[1 1 0]
'magenta' or 'm'	Magenta	[1 0 1]
'cyan' or 'c'	Cyan	[0 1 1]
'white' or 'w'	White	[1 1 1]
'black' or 'k'	Black	[0 0 0]

Data Types: `double` | `char`

CircleTickLabelVisible — Show circle tick labels
`1` (default) | `0`

Show arc tick labels, specified as `0` or `1`.

Data Types: `logical`

See Also

`smithplot`