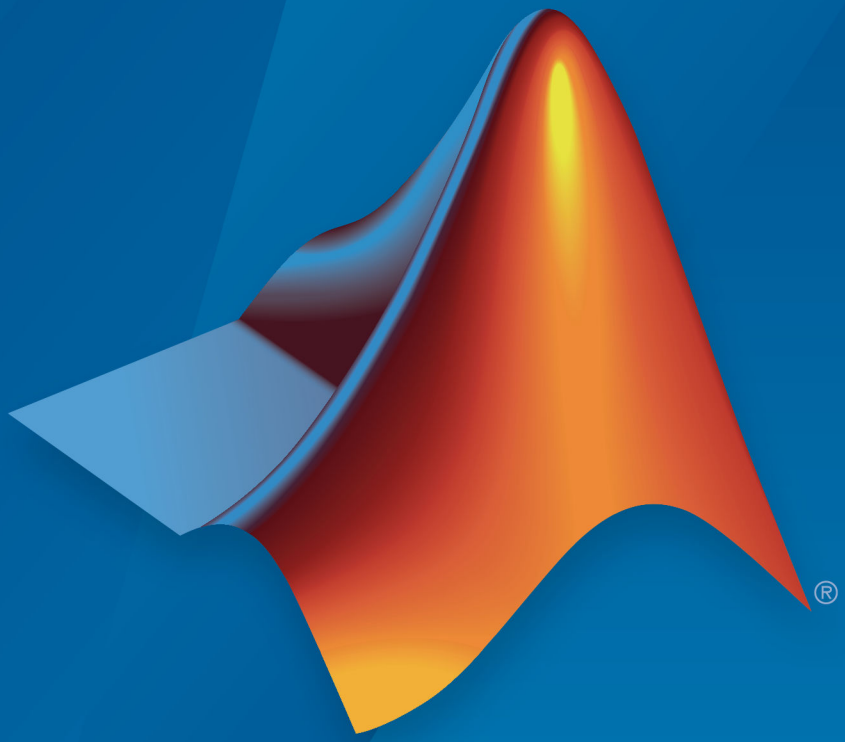
RF Blockset™

Reference



MATLAB® & SIMULINK®

R2019b

MathWorks®

How to Contact MathWorks

Latest news: `www.mathworks.com`

Sales and services: `www.mathworks.com/sales_and_services`

User community: `www.mathworks.com/matlabcentral`

Technical support: `www.mathworks.com/support/contact_us`

Phone: 508-647-7000

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

RF Blockset™ Reference

Revision History

September 2010	Online only	New for Version 3.0 (Release 2010b)
April 2011	Online only	Revised for Version 3.0.2 (Release 2011a)
September 2011	Online only	Revised for Version 3.1 (Release 2011b)
March 2012	Online only	Revised for Version 3.2 (Release 2012a)
September 2012	Online only	Revised for Version 3.3 (Release 2012b)
March 2013	Online only	Revised for Version 4.0 (Release 2013a)
September 2013	Online only	Revised for Version 4.1 (Release 2013b)
March 2014	Online only	Revised for Version 4.2 (Release 2014a)
October 2014	Online only	Revised for Version 4.3 (Release 2014b)
March 2015	Online only	Revised for Version 4.4 (Release 2015a)
September 2015	Online only	Revised for Version 4.5 (Release 2015b)
March 2016	Online only	Revised for Version 5.0 (Release 2016a)
September 2016	Online only	Revised for Version 5.1 (Release 2016b)
March 2017	Online only	Revised for Version 6.0 (Release 2017a)
September 2017	Online only	Revised for Version 6.1 (Release 2017b)
March 2018	Online only	Revised for Version 7.0 (Release 2018a)
September 2018	Online only	Revised for Version 7.1 (Release 2018b)
March 2019	Online only	Revised for Version 7.2 (Release 2019a)
September 2019	Online only	Revised for Version 7.3 (Release 2019b)

Contents

1 Circuit Envelope Blocks — Alphabetical List

2 Equivalent Baseband Blocks — Alphabetical List

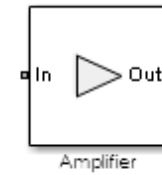
3 Functions — Alphabetical List

1

Circuit Envelope Blocks — Alphabetical List

Amplifier

Model amplifier in RF systems
Library: RF Blockset / Circuit Envelope / Elements



Description

Use the Amplifier block to model a linear or nonlinear amplifier, with or without noise. Defining the amplifier gain using a data source also defines input data visualization and modeling. Use the **Main** tab parameters to specify amplifier gain and noise using data sheet values, standard `s2p` files, S-parameters, or circuit envelope polynomial coefficients.

The amplifier is implemented as a polynomial, voltage-controlled voltage source (VCVS) except when the amplifier gain is obtained from a `Data source`. The VCVS includes nonlinearities that are described using parameters listed in the **Nonlinearity** tab. To model linear amplification, the amplifier implements the relation $V_{out} = a_1 * V_{in}$ between the input and output voltages. The input voltage is $V_i(t) = A_i(t)e^{j\omega t}$, and the output voltage is $V_o(t) = A_o(t)e^{j\omega t}$ at each carrier $w = 2\pi f$ in the RF Blockset environment.

In case the amplifier gain is obtained from a data source, amplifier implementation is based on S-parameter data.

Nonlinear amplification is modeled as a polynomial (with the saturation power computed automatically). It also produces additional intermodulation frequencies.

Parameters

Main

`Source of amplifier gain` — Source parameter of the amplifier gain

`Available power gain` (default) | `Open circuit voltage gain` | `Data source` | `Polynomial coefficients`

Source parameter of the amplifier gain, specified as one of the following:

- `Available power gain` — **Available power gain** parameter is used to calculate the linear voltage gain term of the polynomial VCVS, a_1. This calculation assumes a matched load termination for the amplifier.
- `Open circuit voltage gain` — **Open circuit voltage gain** parameter is used as the linear voltage gain term of the polynomial VCVS, a_1.
- `Data source` —

  When using the data source option, S_{11} and S_{22}, are used as the input and output impedances. The data sources are specified using either `Data file` or `Network-parameters` or `Rational model`, depending on the value of `Data source`.
- `Polynomial coefficients` — The block implements a nonlinear voltage gain according to the specified polynomial coefficients

`Available power gain` — Available power gain

0 dB (default) | scalar

Available power gain of amplifier, specified as a scalar in dB. Specify the units from the corresponding drop-down list.

Dependencies

To enable this parameter, choose `Available power gain` in the **Source of amplifier gain** tab.

`Open circuit voltage gain` — Open circuit voltage gain

0 dB (default) | scalar

Open circuit voltage of amplifier, specified as a scalar in dB. Specify the units from the corresponding drop-down list.

Dependencies

To enable this parameter, choose `Open circuit voltage gain` in the **Source of amplifier gain** tab.

`Data source` — Data source

`Data File` (default) | `Network-parameters` | `Rational Model`

Data source, specified as one of the following:

- `Data file` — Name of a Touchstone file with the extension `.s2p`.
- `Network-parameters` — Provide **Network parameter** data such as `S-parameters`, `Y-parameters`, and `Z-parameters` with corresponding **Frequency** and **Reference impedance (ohms)** for the amplifier.
- `Rational model` — Provide values for **Residues**, **Poles**, and **Direct feedthrough** parameters which correspond to the equation for a rational model

  $$F(s) = \left(\sum_{k=1}^{n} \frac{C_k}{s - A_k} + D \right),\ s = j2\pi f$$

  In this rational model equation, each C_k is the residue of the pole A_k. If C_k is complex, a corresponding complex conjugate pole and residue must also be enumerated. This object has the properties `C`, `A`, and `D`. You can use these properties to specify the **Residues**, **Poles**, and **Direct feedthrough** parameters.

When the amplifier is nonlinear, the nonlinearity applies only to the S21 term of the scattering parameters representing the 2-port element. In this case, S21 is frequency-independent with its constant value being either the maximal value of S21, or the S21 value at an Operation frequency specified by the user. The other scattering parameters, S11, S12, and S22 remain the same as in the linear case.

Dependencies

To enable this parameter, select `Data source` in **Source of amplifier gain** tab.

`Polynomial coefficients` — Polynomial coefficients

`[0 1]` (default) | vector

Order of polynomial, specified as a vector.

The order of the polynomial must be less than or equal to 9. The coefficients are ordered in ascending powers. If a vector has 10 coefficients, `[`a_0`,`a_1`,`a_2`, ... `a_9`]`, the polynomial it represents is:

$V_{out} = a_0 + a_1V_{in} + a_2{V_{in}}^2 + ... + a_9{V_{in}}^9$
where a_1 represents the linear gain term, and higher-order terms are modeled according to [1].

For example, the vector `[`a_0`,`a_1`,`a_2`,`a_3`]` specifies the relation $V_{out} = a_0 + a_1V_1 + a_2{V_1}^2 + a_3{V_1}^3$. Trailing zeroes are omitted. So, `[`a_0`,`a_1`,`a_2`]` defines the same polynomial as `[`a_0`,`a_1`,`a_2`, 0]`. The default value of this parameter is [0,1], corresponding to the linear relation $V_{out} = V_{in}$.

Dependencies

To enable this parameter, select `Polynomial coefficients` in **Source of amplifier gain** tab.

`Network parameter type` — Network parameter type
`S-parameters` (default) | `Y-parameters` | `Z-parameters`

Network parameter type, specified as `S-parameters`, `Y-parameters`, or `Z-parameters`.

Dependencies

To enable this parameter, first select `Data source` in **Source of amplifier gain** tab. Then, select `Network-parameters` in the **Data source** tab.

`Input impedance (Ohm)` — Input impedance
`50` (default) | scalar

Input impedance of amplifier, specified as a scalar.

Dependencies

To enable this parameter, select `Available power gain`, `Open circuit voltage gain`, or `Polynomial coefficients` in **Source of amplifier gain** tab.

`Output impedance (Ohm)` — Output impedance
`50` (default) | scalar

Output impedance of amplifier, specified as a scalar.

Dependencies

To enable this parameter, select `Available power gain`, `Open circuit voltage gain`, or `Polynomial coefficients` in **Source of amplifier gain** tab.

`Data file` — Name of network parameter data file

`simrfV2_unitygain.s2p` (default) | character vector

Name of network parameter data file, specified as a character vector.

Dependencies

To enable this parameter, first select `Data source` in **Source of amplifier gain** tab. Then, select `Data file` in **Data source**.

`Frequency (dB)` — Frequency of network parameters

`1e9 Hz` (default) | scalar | `Hz` | `kHz` | `MHz` | `GHz`

Frequency of network parameters, specified as a scalar in Hz.

Dependencies

To enable this parameter, first select `Data source` in **Source of amplifier gain** tab. Then, select `Network-parameters` in **Data source**.

`Reference Impedance(Ohm)` — Reference impedance of network parameters

`50` (default) | scalar

Reference impedance of network parameters, specified as a scalar.

Dependencies

To enable this parameter, first select `Data source` in **Source of amplifier gain** tab. Then, select `Network-parameters` in **Data source**.

`Residues` — Residues in order of rational model

`0` (default) | vector

Residues in order of rational model, specified as a vector.

Dependencies

To enable this parameter, first select `Data source` in **Source of amplifier gain** tab. Then, select `Rational model` in **Data source**.

`Poles` — Residues in order of rational model

`0` (default) | vector

Poles in order of rational model, specified as a vector.

Dependencies

To enable this parameter, first select `Data source` in **Source of amplifier gain** tab. Then, select `Rational model` in **Data source**.

`Direct feedthrough` — Direct feedthrough

`{0 0:1 0}` (default) | array of vectors

Direct feedthrough, specified as an array vector.

Dependencies

To enable this parameter, first select `Data source` in **Source of amplifier gain** tab. Then, select `Rational model` in **Data source**.

`Specify operation frequency` — Specify operation frequency

`on` (default) | `off`

Select this option to specify operation frequency.

By default, this option is not selected.

Dependencies

To enable this parameter, first you should specify nonlinear `Polynomial coefficients` in **Source of amplifier gain**. Then select `Piece-wise linear` or`Colored` in **Noise distribution** in the Noise pane.

`Operation frequency` — Operation frequency

`0` (default) | scalar | vector

Operation frequency, specified as a scalar or vector in Hz.

Dependencies

To enable this parameter, first you should select **Specify operation frequency**.

`Ground and hide negative terminals` — Ground RF circuit terminals

`on` (default) | `off`

Select this option to ground and hide the negative terminals. Clear this parameter to expose the negative terminals. By exposing these terminals, you can connect them to other parts of your model.

By default, this option is selected.

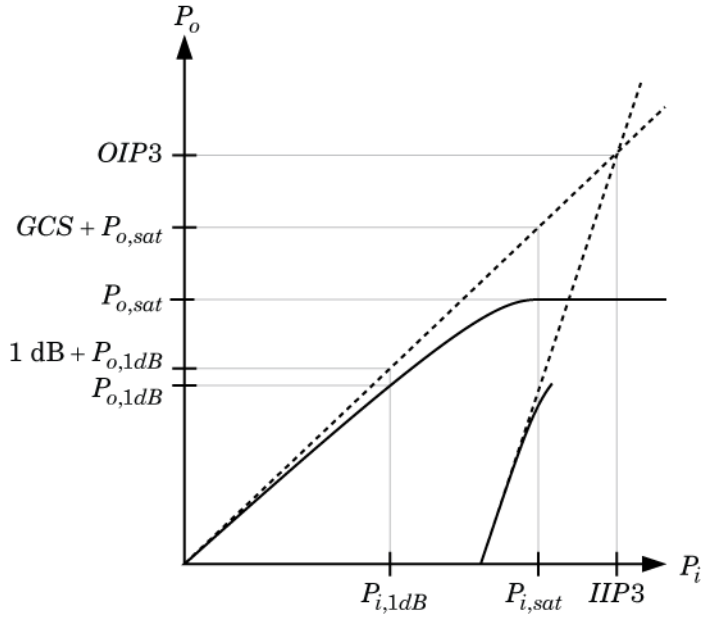
Nonlinearity

`Nonlinear polynomial type` — Type of nonlinearity
`Even and odd order` (default) | `Odd order`

Type of nonlinearity, specified as `Even and odd order` or `Odd order`.

- When you select `Even and odd order`, the amplifier can produce second- and third-order intermodulation frequencies in addition to a linear term.
- When you select `Odd order`, the amplifier generates only odd order intermodulation frequencies.

  The linear gain determines the linear a_1 term. The block calculates the remaining terms from the specified parameters. These parameters are **IP3**, **1-dB gain compression power**, **Output saturation power**, and **Gain compression at saturation**. The number of constraints you specify determines the order of the model. The figure shows the graphical definition of the nonlinear amplifier parameters.



`Intercept points convention` — Intercept points convention
`Output` (default) | `Input`

Intercept points convention, specified a `Input`-referred, or `Output`-referred convention. Use this specification for the intercept points, 1-dB gain compression power, and saturation power.

`IP2` — Second-order intercept point
`infdBm` (default) | scalar | `W` | `mW` | `dBW` | `dBm`

Second-order intercept point, specified as a scalar.

Dependencies

To set this parameter, select `Even and odd order` in **Nonlinear polynomial type**.

`IP3` — Third-order intercept point
`infdBm` (default) | scalar | `W` | `mW` | `dBW` | `dBm`

Third-order intercept point, specified as a scalar.

`1-dB gain compression power` — 1-dB gain compression power
`inf dBm` (default) | scalar | `W` | `mW` | `dBW` | `dBm`

1-dB gain compression power, specified as a scalar.

Dependencies

To set this parameter, select `Odd order` in **Nonlinear polynomial type**.

`Output saturation power` — Output saturation power
`inf dBm` (default) | scalar | `W` | `mW` | `dBW` | `dBm`

Output saturation power, specified as scalar. The block uses this value to calculate the voltage saturation point used in the nonlinear model. In this case, the first derivative of the polynomial is zero, and the second derivative is negative.

Dependencies

To set this parameter, select `Odd order` in **Nonlinear polynomial type**.

`Gain compression at saturation` — Gain compression at saturation
`inf dBm` (default) | scalar | `W` | `mW` | `dBW` | `dBm`

Gain compression at saturation, specified as scalar.

When **Nonlinear polynomial type** is `Odd order`, specify the gain compression at saturation.

Dependencies

To set this parameter, first select `Odd order` in **Nonlinear polynomial type**. Then, change the default value of **Output saturation power**

`Specify operation frequency` — Specify operation frequency
`on` (default) | `off`

Select this option to specify operation frequency.

By default, this option is not selected.

Dependencies

To enable this parameter, the data source must be nonlinear or the noise should be colored.

`Operation frequency` — Operation frequency

`0` (default) | scalar | vector

Operation frequency, specified as a scalar or vector in Hz.

Dependencies

To enable this parameter, first you should select **Specify operation frequency**.

Noise

`Simulate noise` — Simulate thermal noise

`on` (default) | `off`

Select this parameter, to simulate noise as specified in block parameters or on file.

If the noise is specified in an `.s2p` file, then it is used for simulation.

`Noise type` — Noise type

`Noise figure` (default) | `Spot noise data`

Noise type, specified as `Noise figure` or `Spot noise data`.

`Noise distribution` — Noise distribution

`White` (default) | `Piece-wise linear` | `Colored`

Noise distribution, specified as:

- `White`, spectral density is a single non-negative value. The power value of the noise depends on the bandwidth of the carrier and the bandwidth depends on the time step. This is an uncorrelated noise source.
- `Piece-wise linear`, spectral density is a vector of values $[p_i]$. For each carrier, the noise source behaves like a white uncorrelated noise. The power of the noise source is carrier-dependent.
- `Colored`, depends on both carrier and bandwidth. This is a correlated noise source.

`Noise figure (dB)` — Noise figure

0 (default) | scalar

Noise figure, specified as a scalar in decibels.

`Frequencies` — Frequency data

`0Hz` (default) | scalar | vector

Frequency data, specified as a scalar or vector in hertz.

Dependencies

To set this parameter, first select `Piece-wise linear` or `Colored` in **Noise distribution**.

`Minimum noise figure (dB)` — Minimum noise figure

`0` (default) | scalar | vector

Minimum noise figure, specified as a scalar or vector in decibels.

Dependencies

To set this parameter, first select `Spot noise data` in **Noise type**.

`Optim reflection coefficient` — Optim reflection coefficient

`0` (default) | scalar | vector

Optim reflection coefficient, specified as a scalar or a vector.

Dependencies

To set this parameter, first select `Spot noise data` in **Noise type**.

`Equivalent normalized noise resistance` — Equivalent normalized noise resistance

`0` (default) | scalar | vector

Equivalent normalized noise resistance, specified as a scalar or vector.

Dependencies

To set this parameter, first select `Spot noise data` in **Noise type**.

`Automatically estimate impulse response duration` — Automatically estimate impulse response duration

`on` (default) | `off`

Select this parameter to automatically calculate impulse response for frequency dependent noises. Clear this parameter to manually specify the impulse response duration using **Impulse response duration**. You cannot specify impulse response when amplifier is nonlinear, as in this case noise is simulated as white-noise.

Dependencies

To set this parameter, first select `Colored` in **Noise distribution**.

`Impulse response duration` — Impulse response duration

`1e-10s` (default) | scalar

Impulse response duration used to simulate frequency dependant noise, specified as a scalar in seconds. You cannot specify impulse response if the amplifier is nonlinear.

Dependencies

To set this parameter, first clear **Automatically estimate impulse response duration**.

Modeling

`Modeling options` — Model S-parameters

`Time-domain (rationalfit)` (default) | `Frequency-domain`

Model S-parameters, specified as:

- Time-domain (rationalfit) technique creates an analytical rational model that approximates the whole range of the data. When modeling using `Time domain`, the **Plot** in `Visualization` tab plots the data defined in `Data Source` and the values in the `rationalfit` function.
- Frequency-domain computes the baseband impulse response for each carrier frequency independently. This technique is based on convolution. There is an option to specify the duration of the impulse response. For more information, see “Compare Time and Frequency Domain Simulation Options for S-parameters”.
- For the Amplifier and S-parameters blocks, the default value is `Time domain (rationalfit)`. For the Transmission Line block, the default value is `Frequency domain`.

Dependencies

To set this parameter, first select `Data source` in **Source of amplifier gain**. This selection activates the **Modeling** Tab which contains **Modeling options**

`Fitting options` — Rationalfit fitting options

`Fit individually` (default) | `Share poles by column` | `Share all poles`

Rationalfit fitting options, specified as `Fit individually`, `Share poles by column`, or `Share all poles`.

Rational fitting results shows values of **Number of independent fits**, **Number of required poles**, and **Relative error achieved (dB)**.

Dependencies

To set this parameter, select `Time domain (rationalfit)` in **Modeling options**.

`Relative error desired (dB)` — Relative error acceptable for the rational fit

`-40` (default) | scalar

Relative error acceptable for the rational fit, specified as a scalar.

Dependencies

To set this parameter, select `Time domain (rationalfit)` in **Modeling options**.

`Automatically estimate impulse response duration` — Automatically calculate impulse response

`on` | `off`

Select this parameter to automatically calculate impulse response. Clear this parameter to manually specify the impulse response duration using **Impulse response duration**.

Dependencies

To set this parameter, select `Frequency domain` in **Modeling options**.

`Impulse response duration` — Impulse response duration

`1e-10` (default) | scalar

Impulse response duration, specified as a scalar.

Dependencies

To set this parameter, first select `Frequency domain` in **Modeling options**. Then, clear `Automatically estimate impulse response duration`.

`Use only S-parameter magnitude with appropriate delay` — Use only S-parameter magnitude with appropriate delay

`off` (default) | `on`

Select this parameter to s-parameter phase and delay the impulse response by half its length. This parameter is applicable only for S-parameter data modeled in time domain. You can use this to shape spectral content with filter effects by specifying only magnitude.

Note This parameter introduces an artificial delay to the system.

Visualization

`Source of frequency data` — Frequency data source

`Extracted from data source` (default) | `User-defined`

Frequency data source, specified as:

When **Source of frequency data** is `Extracted from data source`, the **Data source** must be set to `Data file`. Verify that the specified **Data file** contains frequency data.

When **Source of frequency data** is `User-specified`, specify a vector of frequencies in the **Frequency data** parameter. Also, specify units from the corresponding drop-down list.

Dependencies

To set this parameter, first select `Data source` in **Source of amplifier gain**. This selection activates the **Visualization** Tab which contains **Source of frequency data**

`Frequency data` — Frequency data range

`[1e9:1e6:3e9]` (default) | vector | `Hz` | `kHz` | `MHz` | `GHz`

Frequency data range, specified as a vector

`Plot type` — Type of data plot

`X-Y plane` (default) | `Polar plane` | `Z Smith chart` | `Y Smith chart` | `ZY Smith chart`

Type of data plot that you want to produce with your data specified as one of the following:

- `X-Y plane` — Generate a Cartesian plot of your data versus frequency. To create linear, semilog, or log-log plots, set the **Y-axis scale** and **X-axis scale** accordingly.
- `Polar plane` — Generate a polar plot of your data. The block plots only the range of data corresponding to the specified frequencies.
- `Z smith chart`, `Y smith chart`, and `ZY smith chart` — Generate a Smith® chart. The block plots only the range of data corresponding to the specified frequencies.

`Parameter 1` — Type of S-Parameters to plot
`S11` (default) | `S12` | `S21` | `S22` | `NF`

Type of S-Parameters to plot, specified as `S11`, `S12`, `S21`, or `S22`. When noise is spectral `NF` plotting is possible.

`Parameter 2` — Type of S-Parameters to plot
`None` (default) | `S11` | `S12` | `S21` | `S22` | `NF`

Type of S-Parameters to plot, specified as `S11`, `S12`, `S21`, or `S22`. When noise is spectral `NF` plotting is possible.

`Format1` — Plot format
`Magnitude (decibels)` (default) | `Angle(degrees)` | `Real` | `Imaginary`

Plot format, specified as `Magnitude (decibels)`, `Angle(degrees)`, `Real`, or `Imaginary`.

`Format2` — Plot format
`Magnitude (decibels)` (default) | `Angle(degrees)` | `Real` | `Imaginary`

Plot format, specified as `Magnitude (decibels)`, `Angle(degrees)`, `Real`, or `Imaginary`.

`Y-axis scale` — Y-axis scale
`Linear` (default) | `Logarithmic`

Y-axis scale, specified as `Linear` or `Logarithmic`.

`X-axis scale` — X-axis scale
`Linear` (default) | `Logarithmic`

X-axis scale, specified as `Linear` or `Logarithmic`.

`Plot` — Plot specified data
button

Plot specified data using plot button.

More About

Noise Figure Data

Noise figure represents only a subset of the noise information (spot noise data) needed to fully describe the noise behavior of a two-port device. When only noise figure is specified, RF Blockset amplifier defines the spot noise parameters in the following manner:

$$NF_{\min} = NF\ (F_{\min} = 10^{NF/10}$$
$$R_n = Z_0 \frac{F_{\min} - 1}{4}, \quad Z_0 \in \mathrm{R}$$
$$Y_{opt} = \frac{1}{Z_0}$$

Amplifier exhibits specified noise figure when source impedance is matched to the reference impedance ($Z = Z_0, Z_0 \in \mathrm{R}$).

Spot Noise Data

Noise in RF Blockset amplifiers are represented as two correlated noise sources at the input port of a noiseless two-port:

The noise sources variance and correlation are governed by an ABCD-correlation matrix:

that is determined by measurable quantities:

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

- NF_{min} - Minimum noise figure
- R_n - Equivalent noise resistance

- Y_{opt} - Optimal source admittance
- k - Boltzman's constant
- T - Noise temperature in Kelvin

.

The above quantities are specified in the amplifier from the noise data section in the `.s2p` file or directly as masked parameters in the noise pane. In both cases:

- NF_{min} is specified in decibels
- R_n is specified as equivalent normalized resistance, R_N ($R_n = Z_0R_N$).
- Y_{opt} is specified in terms of optimal reflection coefficient, Γ_{opt} ($Y_{opt} = Y_0(1-\Gamma_{opt})/(1+\Gamma_{opt})$).

In the above, $Z_0 = 1/Y_0$ is the reference impedance that is real. If the **Source of amplifier gain** is `Data source`, the reference impedance is specified in the `.s2p` file or in the amplifier mask. Other wise the reference impedance is 50 ohms.

The noise factor, F, of the amplifier is affected by the noisy source impedance, Z_s, and is determined from the ABCD-correlation matrix:

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

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

The noise figure, NF, is obtained from the noise factor using, `NF = 10log(F)`.

References

[1] Gonzalez, Guillermo. “Microwave Transistor Amplifiers: Analysis and Design”, Englewood Cliffs, N.J.: Prentice-Hall, 1984.

[2] Grob, Siegfried and Juergen Lindner. “Polynomial Model Derivation of Nonlinear Amplifiers, *Department of Information Technology*, University of Ulm, Germany.

[3] Kundert, Ken. “Accurate and Rapid Measurement of IP_2 and IP_3”, *The Designers Guide Community*, Version 1b, May 22, 2002. http://www.designers-guide.org/analysis/intercept-point.pdf.

[4] Pozar, David M. “Microwave Engineering”, Hoboken NJ: John Wiley & Sons, 2005.

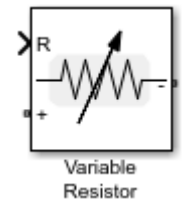
See Also

Mixer | Power Amplifier | S-Parameters

Introduced in R2010b

Variable Resistor

Model variable resistor
Library: RF Blockset / Circuit Envelope / Elements



Description

The Variable Resistor block controls the output of RF Blockset feedback circuits using Simulink® controlled resistance in ohms. The minimum value of the resistance (*Rmin*) is an RF Blockset defined constant independent of the Simulink control signal. The block has two electrical terminals. One terminal is for the Simulink control signal and one terminal is for the RF Blockset signal.

Parameters

`Simulate noise` — Simulate thermal noise
`on` (default) | `off`

Select this parameter, to simulate thermal noise in the variable resistor. Then, in the Configuration block dialog box, also select the **Simulate noise** check box. By default, both **Simulate noise** check boxes are selected.

This parameter inserts a current noise source with the single-sided power density of 4 kT/R A^2/Hz, where:

- k is the Boltzmann constant
- T is the value of the **Temperature** parameter, in degrees Kelvin. (Also located in the Configuration block.)

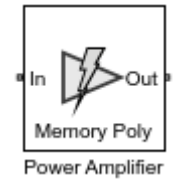
See Also

Resistor | Variable Capacitor | Variable Inductor

Introduced in R2017b

Power Amplifier

Model power amplifier with memory
Library: RF Blockset / Circuit Envelope / Elements



Description

The Power Amplifier block models two-port power amplifiers. A memory polynomial expression derived from the Volterra series models the nonlinear relationship between input and output signals. This power amplifier includes memory effects because the output response depends on the current input signal and the input signal at previous times. These power amplifiers are useful when transmitting wideband or narrowband signals.

Parameters

`Model` — Model type
`Memory polynomial` (default) | `Generalized Hammerstein` | `Cross-Term Memory` | `Cross-Term Hammerstein`

Model type, specified as `Memory polynomial`, `Generalized Hammerstein`, `Cross-Term Memory`, or `Cross-Term Hammerstein`. The following table summarizes the characteristics of the different models:

Model	Characterization Data	Type of Coefficients	In-Band Spectral Regrowth	Out-of-Band Harmonic Generation
`Memory polynomial` (default)	Bandpass (I,Q)	Complex	Yes	No

Model	Characterization Data	Type of Coefficients	In-Band Spectral Regrowth	Out-of-Band Harmonic Generation
`Generalized Hammerstein`	True passband	Real	Yes	Yes
`Cross-Term Memory`	Bandpass (I,Q)	Complex	Yes	No
`Cross-Term Hammerstein`	True passband	Real	Yes	Yes

- `Memory polynomial` – This narrowband memory polynomial implementation (equation (19) of [1]) operates on the envelope of the input signal, does not generate new frequency components, and captures in-band spectral regrowth. Use this model to create a narrowband amplifier operating at high frequency.

  The output signal, at any instant of time, is the sum of all the elements of the following complex matrix of dimensions Memory Depth (mem)×Voltage Order (deg):

  $$\begin{bmatrix} C_{11}V_0 & C_{12}V_0|V_0| & \cdots & C_{1,\deg}V_0|V_0|^{\deg-1} \\ C_{21}V_1 & C_{22}V_1|V_1| & \cdots & C_{2,\deg}V_1|V_1|^{\deg-1} \\ \vdots & \vdots & \ddots & \vdots \\ C_{\mathrm{mem},1}V_{\mathrm{mem}-1} & C_{\mathrm{mem},2}V_{\mathrm{mem}-1}|V_{\mathrm{mem}-1}| & \cdots & C_{\mathrm{mem},\deg}V_{\mathrm{mem}-1}|V_{\mathrm{mem}-1}|^{\deg-1} \end{bmatrix}.$$

  In the matrix, the number of rows equals the number of memory terms, and the number of columns equals the degree of the nonlinearity. The signal subscript represents amount of delay.

- `Generalized Hammerstein` – This wideband memory polynomial implementation (equation (18) of [1]) operates on the envelope of the input signal, generates frequency components that are integral multiples of carrier frequencies, and captures in-band spectral regrowth. Increasing the degree of the nonlinearity increases the number of out-of-band frequencies generated. Use this model to create wideband amplifiers operating at low frequency.

  The output signal, at any instant of time, is the sum of all the elements of the following real matrix of dimensions Memory Depth (mem)×Voltage Order (deg):

$$\begin{bmatrix} C_{11}V_0 & C_{12}V_0^2 & \cdots & C_{1,\deg}V_0^{\deg} \\ C_{21}V_1 & C_{22}V_1^2 & \cdots & C_{2,\deg}V_1^{\deg} \\ \vdots & \vdots & \ddots & \vdots \\ C_{\mathrm{mem},1}V_{\mathrm{mem}-1} & C_{\mathrm{mem},2}V_{\mathrm{mem}-1}^2 & \cdots & C_{\mathrm{mem},\deg}V_{\mathrm{mem}-1}^{\deg} \end{bmatrix}.$$

In the matrix, the number of rows equals the number of memory terms, and the number of columns equals the degree of the nonlinearity. The signal subscript represents amount of delay.

- `Cross-Term Memory` – This narrowband memory polynomial implementation (equation (23) of [1]) operates on the envelope of the input signal, does not generate new frequency components, and captures in-band spectral regrowth. Use this model to create a narrowband amplifier operating at high frequency. The model includes leading and lagging memory terms and provides a generalized implementation of the memory polynomial model.

  The output signal, at any instant of time, is the sum of all the elements of a matrix specified by the element-by-element product
  $\mathbf{C}$.* $\mathbf{M}_{\mathrm{CTM}}$,
  where $\mathbf{C}$ is a complex coefficient matrix of dimensions
  Memory Depth (mem) × {Memory Depth (mem) · (Voltage Order (deg) − 1) + 1} and

$$\mathbf{M}_{\mathrm{CTM}} = \begin{bmatrix} V_0 \\ V_1 \\ \vdots \\ V_{\mathrm{mem}-1} \end{bmatrix} \begin{bmatrix} 1 & |V_0| & |V_1| & \cdots & |V_{\mathrm{mem}-1}| & |V_0|^2 & \cdots & |V_{\mathrm{mem}-1}|^2 & \cdots & |V_0|^{\deg-1} \end{bmatrix}$$

$$= \begin{bmatrix} V_0 & V_0|V_0| & V_0|V_1| & \cdots & V_0|V_{\mathrm{mem}-1}| & V_0|V_0|^2 & \cdots & V_0|V_{\mathrm{mem}-1}|^2 & \cdots \\ V_1 & V_1|V_0| & V_1|V_1| & \cdots & V_1|V_{\mathrm{mem}-1}| & V_1|V_0|^2 & \cdots & V_1|V_{\mathrm{mem}-1}|^2 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots \\ V_{\mathrm{mem}-1} & V_{\mathrm{mem}-1}|V_0| & V_{\mathrm{mem}-1}|V_1| & \cdots & V_{\mathrm{mem}-1}|V_{\mathrm{mem}-1}| & V_{\mathrm{mem}-1}|V_0|^2 & \cdots & V_{\mathrm{mem}-1}|V_{\mathrm{mem}-1}|^2 & \cdots & V_{\mathrm{m} \end{bmatrix}$$

  In the matrix, the number of rows equals the number of memory terms, and the number of columns is proportional to the degree of the nonlinearity and the number of memory terms. The signal subscript represents amount of delay. The additional columns that do not appear in the `Memory polynomial` model represent the cross terms.

- `Cross-Term Hammerstein` – This wideband memory polynomial implementation operates on the envelope of the input signal, generates frequency components that are integral multiples of carrier frequencies, and captures in-band spectral regrowth. Increasing the order of the nonlinearity increases the number of out-of-band frequencies generated. Use this model to create wideband amplifiers operating at low frequency.

  The output signal, at any instant of time, is the sum of all the elements of a matrix specified by the element-by-element product
  $\mathbf{C} .* \mathbf{M}_{\mathrm{CTH}}$,
  where $\mathbf{C}$ is a complex coefficient matrix of dimensions
  Memory Depth (mem) × {Memory Depth (mem) · (Voltage Order (deg) − 1) + 1} and

$$\mathbf{M}_{\mathrm{CTH}} = \begin{bmatrix} V_0 \\ V_1 \\ \vdots \\ V_{\mathrm{mem}-1} \end{bmatrix} \begin{bmatrix} 1 & V_0 & V_1 & \cdots & V_{\mathrm{mem}-1} & V_0^2 & \cdots & V_{\mathrm{mem}-1}^2 & \cdots & V_0^{\mathrm{deg}-1} & \cdots & V_{\mathrm{mem}-1}^{\mathrm{deg}-1} \end{bmatrix}$$

$$= \begin{bmatrix} V_0 & V_0^2 & V_0V_1 & \cdots & V_0V_{\mathrm{mem}-1} & V_0^3 & \cdots & V_0V_{\mathrm{mem}-1}^2 & \cdots & V_0^{\mathrm{deg}} & \cdots & V_0 \\ V_1 & V_1V_0 & V_1^2 & \cdots & V_1V_{\mathrm{mem}-1} & V_1V_0^2 & \cdots & V_1V_{\mathrm{mem}-1}^2 & \cdots & V_1V_0^{\mathrm{deg}-1} & \cdots & V_1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \\ V_{\mathrm{mem}-1} & V_{\mathrm{mem}-1}V_0 & V_{\mathrm{mem}-1}V_1 & \cdots & V_{\mathrm{mem}-1}^2 & V_{\mathrm{mem}-1}V_0^2 & \cdots & V_{\mathrm{mem}-1}^3 & \cdots & V_{\mathrm{mem}-1}V_0^{\mathrm{deg}-1} & \cdots & V_{\mathrm{m} \end{bmatrix}$$

  In the matrix, the number of rows equals the number of memory terms, and the number of columns is proportional to the degree of the nonlinearity and the number of memory terms. The signal subscript represents amount of delay. The additional columns that do not appear in the `Generalized Hammerstein` model represent the cross terms.

`Coefficient Matrix` — Coefficient matrix

1 (default) | complex matrix | real matrix

Coefficient matrix, specified as a complex matrix for the `Memory polynomial` and `Cross-Term Memory` models and as a real matrix for the `Generalized Hammerstein` and `Cross-Term Hammerstein` models.

- For the `Memory polynomial` and `Cross-Term Memory` models, you can identify the complex coefficient matrix based on the measured complex (I,Q) output-vs.-input

amplifier characteristic. As an example, see the helper function in “Coefficient Matrix Computation” on page 1-27.

- For the `Generalized Hammerstein` and `Cross-Term Hammerstein` models, you can identify the real coefficient matrix based on the measured real passband output-vs.-input amplifier characteristic.

The size of the matrix depends on the number of delays and the degree of the system nonlinearity.

- For the `Memory polynomial` and `Generalized Hammerstein` models, the matrix has dimensions Memory Depth (mem)×Voltage Order (deg).
- For the `Cross-Term Memory` and `Cross-Term Hammerstein` models, the matrix has dimensions
  Memory Depth (mem) × {Memory Depth (mem) · (Voltage Order (deg) − 1) + 1}.

`Coefficient Sample Time (s)` — Sample interval of input-output data
`1e-6` (default) | real positive scalar

Sample interval of input-output data used to identify the coefficient matrix, specified as a real positive scalar.

The accuracy of the model can be affected if the coefficient sample time differs from the simulation step size specified in the Configuration block. For best results, use a coefficient sample time at least as large as the simulation step size.

`Rin (ohm)` — Input resistance
`50` (default) | real positive scalar

Input resistance, specified as a real positive scalar.

`Rout (ohm)` — Output resistance
`50` (default) | real positive scalar

Output resistance, specified as a real positive scalar.

`Ground and hide negative terminals` — Ground RF circuit terminals
on (default) | off

Select this parameter to ground and hide the negative terminals. Clear the parameter to expose the negative terminals. By exposing these terminals, you can connect them to other parts of your model.

Algorithms

Coefficient Matrix Computation

To compute coefficient matrices, the block solves an overdetermined linear system of equations. Consider the `Memory polynomial` model for the case where the memory length is 2 and the system nonlinearity is of third degree. The matrix that describes the system is

$$\begin{bmatrix} C_{11}V_0 & C_{12}V_0|V_0| & C_{13}V_0|V_0|^2 \\ C_{21}V_1 & C_{22}V_1|V_1| & C_{23}V_1|V_1|^2 \end{bmatrix},$$

and the sum of its elements is equivalent to the inner product

$$\begin{bmatrix} V_0 & V_1 & V_0|V_0| & V_1|V_1| & V_0|V_0|^2 & V_1|V_1|^2 \end{bmatrix} \begin{bmatrix} C_{11} \\ C_{21} \\ C_{12} \\ C_{22} \\ C_{13} \\ C_{23} \end{bmatrix}.$$

If the input to the amplifier is the five-sample signal [$x(1)$ $x(2)$ $x(3)$ $x(4)$ $x(5)$] and the corresponding output is [$y(1)$ $y(2)$ $y(3)$ $y(4)$ $y(5)$], then the solution to

$$\begin{bmatrix} x(2) & x(1) & x(2)|x(2)| & x(1)|x(1)| & x(2)|x(2)|^2 & x(1)|x(1)|^2 \\ x(3) & x(2) & x(3)|x(3)| & x(2)|x(2)| & x(3)|x(3)|^2 & x(2)|x(2)|^2 \\ x(4) & x(3) & x(4)|x(4)| & x(3)|x(3)| & x(4)|x(4)|^2 & x(3)|x(3)|^2 \\ x(5) & x(4) & x(5)|x(5)| & x(4)|x(4)| & x(5)|x(5)|^2 & x(4)|x(4)|^2 \end{bmatrix} \begin{bmatrix} C_{11} \\ C_{21} \\ C_{12} \\ C_{22} \\ C_{13} \\ C_{23} \end{bmatrix} = \begin{bmatrix} y(2) \\ y(3) \\ y(4) \\ y(5) \end{bmatrix},$$

which can be found using the MATLAB® backslash operator, provides an estimate of the coefficient matrix.

The treatment of the `Cross-Term Memory` model is similar. The matrix that describes the system is

$$\begin{bmatrix} C_{11}V_0 & C_{12}V_0|V_0| & C_{13}V_0|V_1| & C_{14}V_0|V_0|^2 & C_{15}V_0|V_1|^2 \\ C_{21}V_1 & C_{22}V_1|V_0| & C_{23}V_1|V_1| & C_{24}V_1|V_0|^2 & C_{25}V_1|V_1|^2 \end{bmatrix},$$

and the sum of its elements is equivalent to the inner product

$$\begin{bmatrix} V_0 & V_1 & V_0|V_0| & V_1|V_0| & V_0|V_1| & V_1|V_1| & V_0|V_0|^2 & V_1|V_0|^2 & V_0|V_1|^2 & V_1|V_1|^2 \end{bmatrix} \begin{bmatrix} C_{11} \\ C_{21} \\ C_{12} \\ C_{22} \\ C_{13} \\ C_{23} \\ C_{14} \\ C_{24} \\ C_{15} \\ C_{25} \end{bmatrix}.$$

If the input to the amplifier is the five-sample signal [$x(1)$ $x(2)$ $x(3)$ $x(4)$ $x(5)$] and the corresponding output is [$y(1)$ $y(2)$ $y(3)$ $y(4)$ $y(5)$], then the solution to

$$\begin{bmatrix} x(2) & x(1) & x(2)|x(2)| & x(1)|x(2)| & x(2)|x(1)| & x(1)|x(1)| & x(2)|x(2)|^2 & x(1)|x(2)|^2 & x(2)|x(1)|^2 & x(1)|x(1)|^2 \\ x(3) & x(2) & x(3)|x(3)| & x(2)|x(3)| & x(3)|x(2)| & x(2)|x(2)| & x(3)|x(3)|^2 & x(2)|x(3)|^2 & x(3)|x(2)|^2 & x(2)|x(2)|^2 \\ x(4) & x(3) & x(4)|x(4)| & x(3)|x(4)| & x(4)|x(3)| & x(3)|x(3)| & x(4)|x(4)|^2 & x(3)|x(4)|^2 & x(4)|x(3)|^2 & x(3)|x(3)|^2 \\ x(5) & x(4) & x(5)|x(5)| & x(4)|x(5)| & x(5)|x(4)| & x(4)|x(4)| & x(5)|x(5)|^2 & x(4)|x(5)|^2 & x(5)|x(4)|^2 & x(4)|x(4)|^2 \end{bmatrix}$$

$$\begin{bmatrix} C_{11} \\ C_{21} \\ C_{12} \\ C_{22} \\ C_{13} \\ C_{23} \\ C_{14} \\ C_{24} \\ C_{15} \\ C_{25} \end{bmatrix} = \begin{bmatrix} y(2) \\ y(3) \\ y(4) \\ y(5) \end{bmatrix}$$

provides an estimate of the coefficient matrix.

Use this helper function to compute coefficient matrices for the `Memory polynomial` and `Cross-Term Memory` models. The inputs to the function are the given input and output signals, the memory length, the degree of nonlinearity, and the absence or presence of cross terms.

```
function a_coef = fit_memory_poly_model(x,y,memLen,degLen,modType)
% FIT_MEMORY_POLY_MODEL
%   Procedure to compute a coefficient matrix given input and output
%   signals, memory length, nonlinearity degree, and model type.
%
%   Copyright 2017 MathWorks, Inc.

x = x(:);
y = y(:);
xLen = length(x);

switch modType
  case 'memPoly'  % Memory polynomial
    xrow = reshape((memLen:-1:1)' + (0:xLen:xLen*(degLen-1)),1,[]);
    xVec = (0:xLen-memLen)' + xrow;
    xPow = x.*(abs(x).^(0:degLen-1));
    xVec = xPow(xVec);
  case 'ctMemPoly'  % Cross-term memory polynomial
    absPow = (abs(x).^(1:degLen-1));
    partTop1 = reshape((memLen:-1:1)'+(0:xLen:xLen*(degLen-2)),1,[]);
    topPlane = reshape(                                                  ...
        [ones(xLen-memLen+1,1),absPow((0:xLen-memLen)' + partTop1)].',  ...
        1,memLen*(degLen-1)+1,xLen-memLen+1);
    sidePlane = reshape(x((0:xLen-memLen)' + (memLen:-1:1)).',          ...
        memLen,1,xLen-memLen+1);
    cube = sidePlane.*topPlane;
    xVec = reshape(cube,memLen*(memLen*(degLen-1)+1),xLen-memLen+1).';
end

coef = xVec\y(memLen:xLen);
a_coef = reshape(coef,memLen,numel(coef)/memLen);
```

References

[1] Morgan, Dennis R., Zhengxiang Ma, Jaehyeong Kim, Michael G. Zierdt, and John Pastalan. "A Generalized Memory Polynomial Model for Digital Predistortion of

Power Amplifiers." *IEEE® Transactions on Signal Processing*. Vol. 54, No. 10, October 2006, pp. 3852–3860.

[2] Gan, Li, and Emad Abd-Elrady. "Digital Predistortion of Memory Polynomial Systems using Direct and Indirect Learning Architectures." In *Proceedings of the Eleventh IASTED International Conference on Signal and Image Processing (SIP)* (F. Cruz-Roldán and N. B. Smith, eds.), No. 654-802. Calgary, AB: ACTA Press, 2009.

See Also

Amplifier

Topics

“Power Amplifier Characterization with DPD for Reduced Signal Distortion”

Introduced in R2017b

Attenuator

Model attenuator for RF circuit
Library: RF Blockset / Circuit Envelope / Elements

Description

The Attenuator block to attenuates the signal power by a given factor known as Insertion Loss in dB. Commonly, the block matches the impedance of the RF circuit at the input and output ports. You can use attenuators to dampen the power of the incoming signal to protect RF circuits.

Parameters

`Attenuation (dB)` — Level of insertion loss or attenuation
`3` (default) | scalar

Level of insertion loss or attenuation to apply to the signal, specified as a scalar in dB.

`Input impedance (Ohm)` — Input impedance
`50` (default) | scalar

Input impedance of the attenuator, specified as a scalar in ohms.

`Output impedance (Ohm)` — Output impedance
`50` (default) | scalar

Output impedance of the attenuator, specified as a scalar in ohms.

`Simulate noise` — Simulate thermal noise
on (default) | off

Select this parameter to simulate thermal noise in the attenuator. You must select **Simulate noise** in the Configuration block.

This parameter inserts a current noise source with the single-sided power density of 4 kT/R A^2/Hz, where:

- T is the value of the **Temperature** parameter in the Configuration block. Units are in degrees Kelvin.
- k is the Boltzmann constant.

`Ground and hide negative terminals` — Ground RF circuit terminals
on (default) | off

Select this parameter to ground and hide the negative terminals. To expose the negative terminals, clear this parameter. By exposing these terminals, you can connect them to other parts of your model.

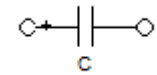
See Also

Variable Attenuator

Introduced in R2016b

Capacitor

Model capacitor for circuit envelope analysis
Library: RF Blockset / Circuit Envelope / Elements



Description

The Capacitor block models a capacitor in circuit envelope environment.

Parameters

`Capacitance` — Capacitance value

`1e-12 F` (default) | real number

Capacitance value, specified as a real number. Specify the units of the capacitance from the corresponding drop-down menu.

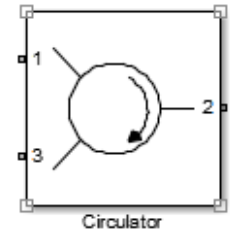
If you set this parameter value between `0` and `1e-18 F`, the block uses a value equal to `1e-18 F` during simulation. By default, the value is `1e-12 F`.

See Also

Inductor | Resistor

Circulator

Model ideal frequency-independent circulators with S-parameters
Library: RF Blockset / Circuit Envelope / Junctions



Description

Use the Circulator block to model ideal passive three-port circulators in a circuit envelope environment. Circulators are used to control signal direction and flow inside an RF circuit. The Circulator block can circulate the signal in clockwise or counterclockwise direction depending on the parameter selection.

Parameters

`Select component` — Circulator direction
`Circulator clockwise` (default) | `Circulator counter clockwise` | `Tee H-plane (S11=0)` | `Reciprocal phase shifter`

Circulator direction, specified as:

- `Circulator clockwise`

  The default option is `Circulator clockwise`. The s-parameter matrix for `Circulator clockwise` is:

  $$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

  `Circulator counter clockwise`

The s-parameter matrix for `Circulator counter clockwise` is:

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

- `Tee H-plane (S11=0)`

  The s-parameter matrix for `Tee H-plane (S11=0)` is:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- `Reciprocal phase shifter`

  The s-parameter matrix for `Reciprocal phase shifter` is:

$$\begin{pmatrix} 0 & phase_{12} & 0 \\ phase_{12} & 0 & 0 \\ 0 & 0 & phase_{33} \end{pmatrix} \left(\frac{-j}{\sqrt{2}} \right)$$

`Reference impedances (Ohm)` — Reference impedance of circulator
`50` (default) | positive scalar | three-tuple vector

Reference impedance of circulator, specified as a scalar or three-tuple vector.

`Phase shift, ports 1 and 2 (rad)` — Angle to shift phase of signal in port 1 and port 2
`0` (default) | positive scalar

Angle to shift the phase of the signal in port 1 and port 2, specified as a positive scalar in radians.

`Phase shift, ports 3 (rad)` — Angle to shift phase of signal in port 3
`0` (default) | positive scalar

Angle to shift the phase of the signal in port 3, specified as a positive scalar in radians.

`Ground and hide negative terminals` — Ground RF circuit terminals
`on` (default) | `off`

Select this parameter to ground and hide the negative terminals. To expose the negative terminals, clear this parameter. By exposing these terminals, you can connect them to other parts of your model.

By default, this option is selected.

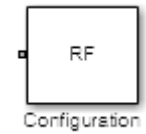
See Also

Coupler | Divider

Introduced in R2014a

Configuration

Define system simulation settings
Library: RF Blockset / Circuit Envelope / Utilities



Description

Use the Configuration block to set the model conditions for a circuit envelope simulation. The block parameter defines RF and solver attributes. The RF attributes include properties such as simulation frequencies, harmonic order, envelope bandwidth, and thermal noise. The solver attributes include transient analysis types, tolerances, and small signal approximation.

The small signal transient simulation performs a full non-linear harmonic balance steady state solution to determine an operation point for a subsequent linear transient analysis. This option allows you to capture the right spectral behavior of a small signal affected by large constant (over carrier) signals.

Connect one Configuration block to each topologically distinct RF Blockset subsystem. Each Configuration block defines the parameters of the connected RF Blockset subsystem. To see an example of the Configuration block in a model, enter `RFNoiseExample` in the MATLAB Command Window.

For an introduction to RF simulation, see “Simulate High Frequency Components”.

Parameters

Main

`Automatically select fundamental tones and harmonic order` — Automatically select fundamental tones and harmonic order
`on` (default) | `off`

Select this parameter to choose **Fundamental tones** and **Harmonic order** parameters automatically when you update the model. Automatic selection does not always return the smallest possible set of simulation frequencies. This approach uses conservative number of simulation frequencies to capture the non -linear behavior of the system.

To set the **Fundamental tones** and **Harmonic order**, clear this parameter. A smaller set of simulation frequencies decreases simulation time and decreases memory requirements. However, a decrease in simulation frequencies can reduce accuracy.

`Fundamental tones` — Fundamental tones of set of simulation frequency

vector of positive integers in Hz

Fundamental tones of a set of simulation frequencies, specified as a vector of positive integers in Hz.

Dependencies

To enable this parameter, clear **Automatically select fundamental tones and harmonic order**.

`Harmonic order` — Harmonic order for each fundamental tone

vector of positive integers

Harmonic order for each fundamental tone, specified as a vector of positive integer. You can also specify a scalar and this value is applied to each **Fundamental tones**.

Dependencies

To enable this parameter, clear **Automatically select fundamental tones and harmonic order**.

`Total simulation frequencies: Computed at simulation time` — Displays number for simulation frequencies

button

The block determines the simulation frequencies based on the fundamental tones and their respective harmonic order. The solver computes a solution to the network at each simulation frequency and the computation time scales according to the total number of simulation frequencies.

Combinations of fundamental tones determine the set of simulation frequencies: [$m*f1$ + $n*f2$ + ...]. In this case, the fundamental tones are represented by [f_s1,$f2$,...], and the

integers m and n are integers bounded by the corresponding **Harmonic order**, $|m| \leqq h1$, $|n| \leqq h2$, etc. Only positive frequencies are considered.

Click **View** to open the dialog box containing additional information about the simulation frequencies in your system. The `Configuration` block displays the number of simulation frequencies for a nonlinear model. For linear models, the actual number of frequencies are automatically optimized during simulation.

By clicking a listed simulation frequency, you can see which linear or multiple combinations of fundamental tones represent that frequency. From the dialog box, you can also plot the simulation frequencies on a number line.

Consider a single fundamental tone $f1$ = 2 GHz and corresponding harmonic order $h1$ = 3. The set of simulation frequencies are: `[0, f1, 2f1, 3f1] = [0GHz, 2 GHz, 4 GHz, 6GHz]`.

Consider a circuit with two fundamental tones `[f1 = 2 GHz, f2 = 50 MHz]` and corresponding harmonic orders `h1 = h2 = 1`. This setup results in five simulation frequencies with values: `[0, f2, f1-f2, f1, f1+f2]`.

Consider a circuit with two fundamental tones `[f1 = 2 GHz, f2 = 3GHz]` and corresponding harmonic orders `h1 = 1`, and `h2 = 3`. This setup results in 11 simulation frequencies with values: `[0, f2, f1-f2, f1, f1+f2, -f1+2f2, 2f2, -f1+3f2, f1+2f2, 3f2, f1+3f2]`.

The set of simulation frequencies must include all carrier frequencies specified in the RF Blockset subsystem such as the carrier frequencies inside Inport, Outport, and source blocks.

Dependencies

To enable this parameter, select **Automatically select fundamental tones and harmonic order**. If you clear **Automatically select fundamental tones and harmonic order**, the option becomes, **Total simulation frequencies: N/A: Fundamental tones undefined**.

`Step size` — Time step for fixed step solver configuration

`1e-6` (default) | scalar in seconds

Time step for fixed step solver configuration, specified as a scalar in seconds. The inverse of the time step determines the simulation bandwidth of the signal envelope centered around each simulation frequency.

The time step of a circuit envelope simulation should be commensurate to relative signal bandwidth and not to the absolute value of the carrier frequency.

The default (1e-6s) is sufficient for modelling envelope signals with bandwidths of up to 1/*h*, or 1MHz. Simulation accuracy is reduced when simulating close to the maximum bandwidth. Reduce the step size to model signals with a larger bandwidth, or improve accuracy.

The simulation speed is inversely proportional to the simulation step size. A smaller simulation step size corresponds to a wider envelope bandwidth and to a slower simulation.

When the white noise is simulated, the noise bandwidth for each simulation frequency is equal to 1/*h*.

`Envelope bandwidth` — Maximum simulated envelope bandwidth
`1 MHz` (default) | scalar in Hz

Maximum simulated envelope bandwidth, returned as a scalar in Hz. Configuration block automatically calculates this value using the **Step size** parameter. The formula used is: $bandwidth = \frac{1}{(step\ size)}$.

`Simulate noise` — Globally enable or disable noise modeling
`on` (default) | `off`

Select this parameter to globally enable noise modeling in RF Blockset circuits. When this check box is selected:

- Amplifier and Mixer blocks use the value of their respective **Noise figure (dB)** parameters.
- Amplifier and Mixer blocks simulate with thermal noise at the temperature specified by the **Temperature** parameter.
- Resistor blocks model thermal noise using the **Temperature** parameters.
- Noise blocks model a specified noise power as a voltage or current source.

To disable noise modeling globally, clear this parameter.

`Use default random number generator` — Default pseudorandom noise stream for RF Blockset sources
`on` (default) | `off`

Select this parameter to retain the default pseudo random noise stream for RF Blockset sources. Clear this option to specify an independent pseudo random number stream for the RF Blockset topological subsystem and determine the seed of the noise stream.

Dependencies

To expose this parameter, select **Simulate noise**.

`Noise seed` — Seed of the independent pseudo random number stream

`0` (default) | scalar positive integer

Seed of the independent pseudo random number stream, specified as a scalar positive integer.

Dependencies

To expose this parameter, clear **Use default random number generator**.

`Temperature` — Global noise temperature

`290.0K` | scalar integer in kelvin

Global noise temperature, specified as a scalar integer in kelvin.

`Samples per frame` — Number of samples in each channel of input signal to `Inport` block

`1` | positive scalar integer

Number of samples in each channel of input signal to Inport block, specified as a positive scalar integer. A channel corresponds to an input frequency in the `Inport` block.

`Normalize Carrier Power` — Normalize power of carrier signal

`on` (default) | `off`

Select this option to normalize the carrier power such that the average power of the signal is:

$$I^2 + Q^2$$

In this case, the equation gives the corresponding passband signal at ω:

$$s_k(t) = I(t)\sqrt{2}\cos(2\pi f_k t) - Q(t)\sqrt{2}\sin(2\pi f_k t)$$

where:

- $I(t)$ is the in-phase part of the carrier signal.
- $Q(t)$ is the quadrature part of the carrier signal.
- f_k are the carrier frequencies.

Clear this option so the average power of the carrier signal is:

$$\frac{I^2 + Q^2}{2}$$

In this case, the corresponding passband signal at ω represented by the equation

$$s_k(t) = I(t)\cos(2\pi f_k t) - Q(t)\sin(2\pi f_k t)$$

0 carrier frequency is a special case. Its passband representation is always I and average power I^2

`Enable input interpolation filter` — Automatic interpolation of lower rate baseband signal to higher rate RF signal
on (default) | off

Select this option to enable input interpolation filter to up sample the input signal rates to fit sample rate of the RF solver. You can now directly use baseband communication signals using a lower sample rate in a wider band circuit. This filter introduces a delay in the RF signal. **Filter delay (in samples)** shows the delay introduced after you simulate the model.

Note When you enable this filter,

- The RF-to-baseband sample rate ratio must be 2, 4, 6, or 8.
- The RF Blockset model can have only one Inport block.

Advanced

`Transient analysis` — Fixed-step solver of RF Blockset environment
`Auto` (default) | `NDF2` | `Trapezoidal Rule` | `Backward Euler`

Fixed-step solver of RF Blockset environment, specified as one of the following:

- `Auto`: Set this parameter to `Auto`, when you are not sure which solver to use.
- `NDF2`: Set this parameter to `NDF2` to balance narrowband and wideband accuracy. This solver is suitable for situations where the frequency content of the signals in the system is unknown relative to the Nyquist rate.
- `Trapezoidal Rule`: Set this parameter to `Trapezoidal Rule` for narrowband simulations. Frequency warping and the lack of damping effects make this method inappropriate for most wideband simulations.
- `Backward Euler`: Set this parameter to `Backward Euler` to simulate the largest class of systems and signals. Damping effects make this solver suitable for wideband simulation, but overall accuracy is low.

The RF Blockset solver is an extension of the Simscape™ local solver. For more information on the Simscape local solver, see the Solver Configuration block reference page.

`Approximate transient as small signal` — Choose small subset of frequencies for transient small signal analysis
off (default) | on

Select this option to choose a small subset of frequencies for transient small signal analysis.

`Use all steady-state simulation frequencies for small signal analysis` — Choose all steady-state simulation frequencies
on (default) | off

Select this option to choose all steady-state simulation frequencies. Clear this option to specify the frequencies for small signal transient simulation.

Dependencies

To expose this parameter, check **Approximate transient as small signal**.

`Small signal frequencies` — Frequencies for small signal transient simulation
scalar | vector

Frequencies for small signal transient simulation, specified as a scalar or vector. The frequencies specified are contained in the entire set of simulation frequencies determined from **Fundamental tones** and **Harmonic order** in the **Main** tab.

The default values in this box and the corresponding units are not constants. The values depend on the state of the configuration dialog box when **Use all steady-state simulation frequencies for small signal analysis** is first cleared.

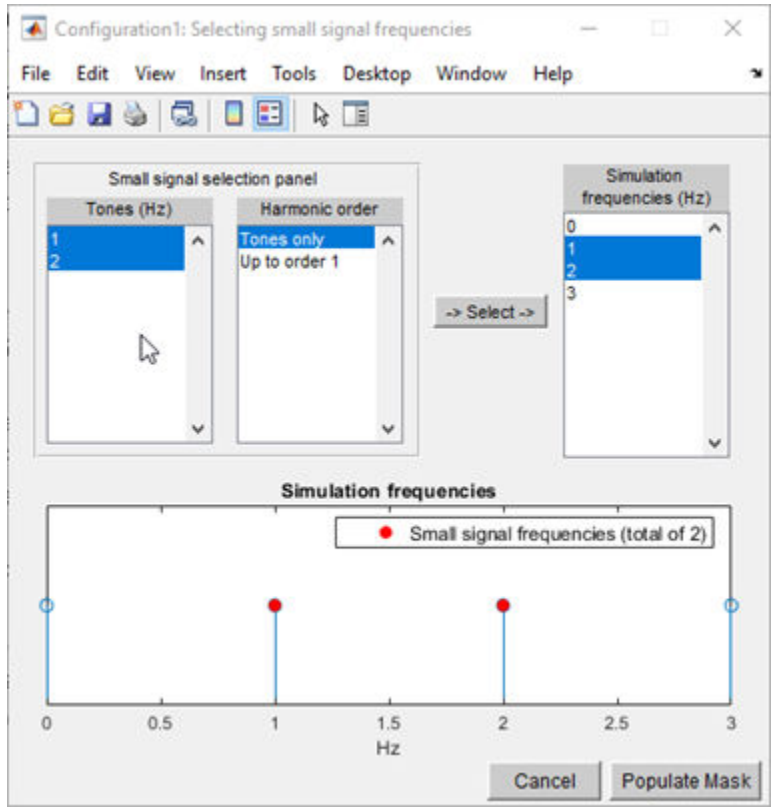
Dependencies

To expose this parameter, clear **Use all steady-state simulation frequencies for small signal analysis**.

`Populate Frequencies` — Tool to choose small signal transient frequencies

button

Tool to choose small signal transient frequencies to populate **Small signal frequencies**. The selected frequencies are a subset of the simulation frequencies determined from **Fundamental tones** and **Harmonic order** inputs in the **Main** tab. The entire set of simulation frequencies are given in the combo box on the right-hand side of the dialog box and the selected frequencies are highlighted. You can select by directly choosing the frequencies in the selection box, or by choosing the desired tones and harmonic order in the **Small signal selection panel** and pressing **Select**. The **Tones(Hz)** and **Harmonic order** values in the combo boxes are also populated using **Fundamental tones** and **Harmonic order** inputs in the **Main** tab.



Dependencies

To expose this parameter, clear **Use all steady-state simulation frequencies for small signal analysis**.

`Relative tolerance` — Relative newton tolerance for system variables

`1e-3` (default) | real positive finite scalar

Relative newton tolerance for system variables, specified as a real positive finite scalar.

`Absolute tolerance` — Absolute newton tolerance for system variables

`1e-6` (default) | real positive finite scalar

Absolute newton tolerance for system variables, specified as a real positive finite scalar.

`Maximum iterations` — Number iterations required for convergence
`10` (default) | real positive integer scalar

Number iterations required for convergence, specified as a real positive integer scalar.

`Error estimation` — Check for error of convergence in system variables
`2-norm over all variables` (default) | `Each variable separately`

Check for error of convergence in system variables, specified as:

- `2-norm over all variables`: Use this option to calculate the 2-norm of all the state variables and then check the error in convergence of state variables.
- `Each variable separately`: Use this option to check the error in convergence of each variable separately.

`Restore Default Settings` — Restore newton solver to default values
button

Restore newton solver to default values, specified as a button.

More About

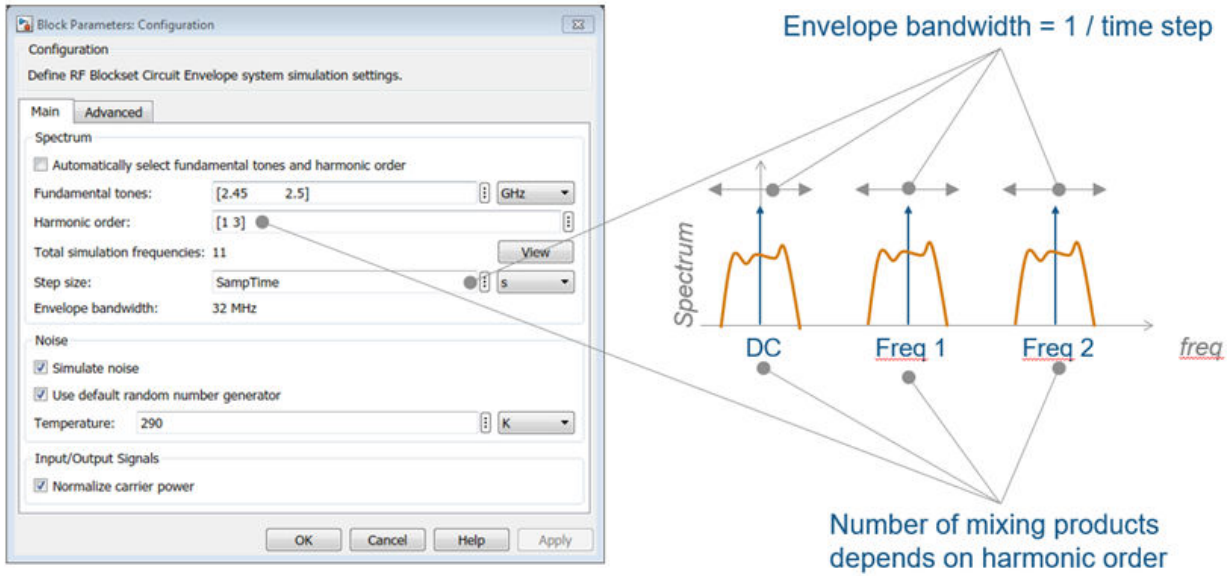
Simulation Setup and Complexity

The key parameters in setting up a Circuit Envelope simulation are the fundamental tones, the harmonic order, and the step size. To speed up simulation, you can trade off the simulation step size and the total number of simulation frequencies.

For example, suppose that you have two large inputs signals each with 100 MHz bandwidth, centered around 10 GHz, and 10.1 GHz respectively. You can simulate the two signals using two separate fundamental tones [10 10.1] GHz. Each tone has a harmonic order of 3 (for a total of 13 simulation frequencies), and a simulation step size equal to 1/200MHz = 5 ns.

You could also set up the RF subsystem so that both of the signals are within the same simulation bandwidth centered around 10.05 GHz. In this case, you set the harmonic order equal to 3 (for a total of 4 simulation frequencies), and a simulation step size equal to 1/400MHz = 2.5 ns. The latter configuration is faster as the number of simulation frequencies is smaller by a factor 3, and the simulation step size is only smaller by a factor 2.

When setting up a circuit envelope simulation, avoid overlapping envelopes. The thermal noise generated by passive components are accounted for separately in each subband thus allowing for overlap of separate envelopes.



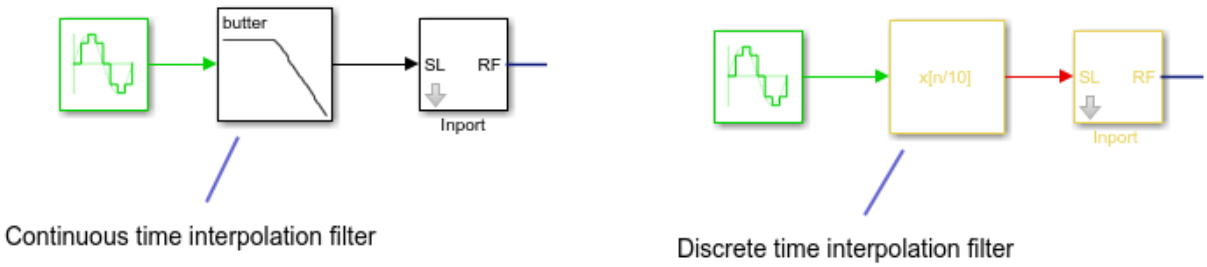
Criteria for Determining Simulation Step Size

The simulation step size must be small enough to capture the signal bandwidth and in-band spectral regrowth.

For example, your complex input Simulink signal has a sample frequency equal to 10 MHz. The minimum time step required to simulate this signal is 1/20 MHz = 50 ns. You can use an oversampling factor from 4 through 8, corresponding to a simulation time step between 25 ns and 12.5 ns. This captures the spectral regrowth caused by non-linear effects.

By default, the configuration block allows automatic interpolation of lower rate baseband signal to higher rate RF signal. If you disable this property, it is recommended that you use the same step size as the input Simulink signals. The input port resamples the input signal with the step size specified in the Configuration block. Using the same step size avoids undesired aliasing effects. It is best to resample the Simulink signals before

importing them in RF Blockset using either analog (continuous time) or digital (discrete time) interpolation filters.



Relative Tolerance and Absolute Tolerance

Circuit envelope solver in the RF Blockset is a solving a set of nonlinear equations from a set of system variables. These system variables are derived from the circuit topology and simulation frequencies. Relative tolerance and absolute tolerance are used to keep the error in convergence of the system variables to minimum. The number of iterations used at each time step dramatically affects the speed of the solutions and the tradeoff between accuracy and speed. The tradeoff is governed by the stopping criterion for the iterations. This stopping criterion is based on 3 sub criteria:

- Variable error convergence:

  $$|\Delta X| < \mathrm{Re}lTol \cdot \max_{t}(|X|) + AbsTol_X$$

  where:

  - X- System variables
  - t- maximum iterations.
- Residue error convergence:

  $$|F(X)| < \mathrm{Re}lTol \cdot \max_{t,n}(|F_n(X)|) + AbsTol_F$$

  where:

  - $F_n(X)$- represents a part of $F(X)$ coming from the nth branch.
- Maximum number of iterations.

Stop the calculations if the first two sub criteria are filled or the last sub criterion is filled. If only one of the sub-criteria is filled, error out that the ' non-linear solver failed'.

See Also

Inport | Outport

Introduced in R2013a

Continuous Wave

Model constant envelope source
Library: RF Blockset / Circuit Envelope / Sources

Description

The Continuous Wave block models a constant modulation on a carrier in the RF Blockset circuit envelope simulation environment. For an introduction to RF simulation, see the example, “Simulate High Frequency Components”.

The block implements the relation

$$v(t) = v_0 e^{j\omega_k t}$$

or

$$i(t) = i_0 e^{j\omega_k t}$$

at the carrier ω_k, depending on the source type.

Parameters

`Source type` — Wave type
`Ideal voltage` (default) | `Ideal current` | `Power`

- `Ideal Voltage` — The block simulates a constant voltage envelope v_0 at the specified **Carrier frequencies**. The envelope has real and imaginary parts specified by the **Constant in-phase value** and **Constant quadrature value** parameters.
- `Ideal Current` — The block simulates a constant current envelope i_0 at the specified **Carrier frequencies**. The envelope has real and imaginary parts determined by the **Constant in-phase value** and **Constant quadrature value** parameters.

- `Power` — The block simulates the constant voltage envelope

  $$v_0 = 2\sqrt{P_0 \mathrm{Re}(Z_s)} e^{j\frac{\pi}{180}\phi}$$

  where:

  - P_0 is the value of the **Available power** parameter
  - Z_s is the value of the **Source impedance (ohms)** parameter.
  - ϕ is the value of the **Angle (degrees)** parameter.

`Constant in-phase value` — In-phase modulations of carrier frequencies

`0 V` (default) | vector of real numbers | `V` | `mV` | `kV`

In-phase modulations for each of the **Carrier frequencies**, specified as a vector of real numbers. Specify the units from the corresponding drop-down list.

Dependencies

To enable this parameter, on the **Source type** tab, choose `Ideal voltage` or `Ideal current`.

`Constant quadrature value` — Quadrature modulations of carrier frequencies

`0 V` (default) | vector of real numbers | `V` | `mV` | `kV`

Quadrature modulations for each of the RF circuit carrier frequencies, specified as a vector of real numbers. Specify the units from the corresponding drop-down list.

Dependencies

To enable this parameter, on the **Source type** tab, choose `Ideal voltage` or `Ideal current`.

`Source impedance(Ohm)` — Input impedance of source

`50` (default) | positive real number | complex number

Input impedance of source, specified as a positive real number or a complex number. The complex number must contain real and imaginary parts greater than or equal to `1e-18` ohms.

Dependencies

To enable this parameter, on the **Source type** tab, choose `Power`.

`Available power` — Input impedance of source

`0 W` (default) | vector of real numbers | `W` | `mW` | `dBW` | `dBm`

Available power at the specified **Carrier frequencies**, specified as a vector of real numbers. Specify the units from the drop-down list. The default value is `0 W`.

Dependencies

To enable this parameter, on the **Source type** tab, choose `Power`. The default value is `0 W`

`Angle(degrees)` — Phase angle of power waves

`0` (default) | vector of real numbers

Phase angle of power waves at the specified **Carrier frequencies**, specified as a vector of real numbers. The default value is `0` degrees.

Dependencies

To enable this parameter, on the **Source type** tab, choose `Power`.

`Carrier frequencies` — Carrier frequencies

`0` (default) | vector of real positive numbers | `Hz` | `kHz` | `MHz` | `GHz`

Carrier frequencies, specified as a vector of real positive numbers. The elements in the carrier frequencies are combinations of fundamental tones and corresponding harmonics in the Configuration block. The default value is `0 Hz`.

`Add phase noise` — Add phase noise

`off` (default) | `on`

Select this parameter, to add phase noise to your system with continuous wave source.

`Phase noise frequency offset (Hz)` — Phase noise frequency offset

`0` (default) | scalar | vector | matrix

Phase noise frequency offset, specified as a scalar or vector or matrix with each element units in hertz.

If you specify a matrix, each column should correspond to a non-DC carrier frequency of the CW source. The frequency offset values must be bounded by the envelope bandwidth of the simulation. For more information see, Configuration.

Dependencies

To enable this parameter, select **Add phase noise**.

`Phase noise level (dBc/Hz)` — Phase noise level

`0` (default) | scalar | vector | matrix

Phase noise level, specified as a scalar or vector or matrix with element unit in decibel per hertz.

If you specify a matrix, each column should correspond to a non-DC carrier frequency of the CW source. The frequency offset values must be bounded by the envelope bandwidth of the simulation. For more information see, Configuration.

Dependencies

To enable this parameter, select **Add phase noise**.

`Automatically estimate impulse response duration` — Automatically estimate impulse response duration

`on` (default) | `off`

Select this parameter to automatically calculate impulse response for phase noise. Clear this parameter to manually specify the impulse response duration using **Impulse response duration**.

`Impulse response duration` — Impulse response duration

`1e-10s` (default) | scalar

Impulse response duration used to simulate phase noise, specified as a scalar in seconds. You cannot specify impulse response if the amplifier is nonlinear.

Note The phase noise profile resolution in frequency is limited by the duration of the impulse response used to simulate it. Increase this duration to improve the accuracy of the phase noise profile. A warning message appears if the phase noise frequency offset resolution is too high for a given impulse response duration, specifying the minimum duration suitable for the required resolution

---

Dependencies

To set this parameter, first clear **Automatically estimate impulse response duration**.

`Ground and hide negative terminals` — Ground RF circuit terminals
`on` (default) | `off`

Select this option to internally ground and hide the negative terminals. To expose the negative terminals, clear the option. By exposing these terminals, you can connect them to other parts of your model.

By default, this option is selected.

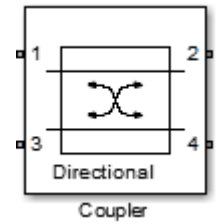
See Also

Inport | Noise | Sinusoid

Introduced in R2010b

Coupler

Model ideal frequency-independent couplers with S-parameters
Library: RF Blockset / Circuit Envelope / Junctions



Description

The Coupler block models four port directional couplers in a circuit envelope environment as an ideal S-parameter model. The four ports of the coupler are `Input port` (Port 1), `Through port` (Port 2), `Isolated port` (Port 3), `Coupled port` (Port 4).

Directional couplers are used to sample forward and reflected waves propagating along a transmission line. Directional couplers find uses in many RF design applications such as line power sensors and transmitter automatic level controls.

Hybrid couplers are used to split or combine signals with specific phase relations.

Parameters

`Select component` — Coupler type
`Directional coupler` (default) | `Coupler symmetrical` | `Coupler antisymmetrical` | `Hybrid quadrature (90 deg)` | `Hybrid rat-race` | `Magic tee`

Coupler type, specified as:

- `Directional coupler`

  The default option is `Directional coupler`. The s-parameter matrix for `Directional coupler` is:

$$\begin{bmatrix} r_l & i_l & i_s & c \\ i_l & r_l & c & i_s \\ i_s & c & r_l & i_l \\ c & i_s & i_l & r_l \end{bmatrix}$$

where:

- $r_l = 10^{(-\text{ReturnLoss}/20)}$
- $i_l = j10^{(-\text{InsertionLoss}/20)}$
- $i_s = j10^{(-(\text{Coupling}+\text{Directivity})/20)}$
- $c = 10^{(-\text{Coupling}/20)}$

Use this option to model coupler parameters from data sheets.

- `Coupler symmetrical`

  The s-parameter matrix for `Coupler symmetrical` is:

  $$\begin{bmatrix} 0 & \alpha & 0 & j\beta \\ \alpha & 0 & j\beta & 0 \\ 0 & j\beta & 0 & \alpha \\ j\beta & 0 & \alpha & 0 \end{bmatrix}$$

  where:

  - $|\alpha| \le 1$ = Power transmission coefficient
  - $\beta = \text{sqrt}(1 - \alpha^*\alpha)$
- `Coupler antisymmetrical`

  The s-parameter matrix for `Coupler antisymmetrical` is:

  $$\begin{bmatrix} 0 & \alpha & 0 & \beta \\ \alpha & 0 & -\beta & 0 \\ 0 & -\beta & 0 & \alpha \\ \beta & 0 & \alpha & 0 \end{bmatrix}$$

  where:

- $|\alpha| \le 1$ = Power transmission coefficient.
- β = sqrt (1– $\alpha^*\alpha$)

- `Hybrid quadrature (90deg)`

  The s-parameter matrix for `Hybrid quadrature(90deg)` is:

$$\begin{bmatrix} 0 & -j/\sqrt{2} & 0 & -1/\sqrt{2} \\ -j/\sqrt{2} & 0 & -1/\sqrt{2} & 0 \\ 0 & -1/\sqrt{2} & 0 & -j/\sqrt{2} \\ -1/\sqrt{2} & 0 & -j/\sqrt{2} & 0 \end{bmatrix}$$

- `Hybrid rat-race`

  The s-parameter matrix for `Hybrid rat-race` is:

$$\begin{bmatrix} 0 & -j/\sqrt{2} & 0 & -j/\sqrt{2} \\ -j/\sqrt{2} & 0 & j/\sqrt{2} & 0 \\ 0 & j/\sqrt{2} & 0 & -j/\sqrt{2} \\ -j/\sqrt{2} & 0 & -j/\sqrt{2} & 0 \end{bmatrix}$$

- `Magic tee`

  The s-parameter matrix for the `Magic tee` is:

$$\frac{\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \\ 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}}{\sqrt{2}}$$

`Coupling (dB)` — Fraction of input signal power coupled to output port
0 (default) | nonnegative and real scalar

Fraction of input signal power coupled to output port of the `Directional coupler`, specified as a nonnegative and real scalar. The default value is `0` dB.

Dependencies

To enable this parameter, select `Directional coupler` in **Select component** tab.

`Directivity (dB)` — Ratio of power at coupled port to power at isolated port

`inf` (default) | nonnegative and real scalar

Ratio of power at coupled port to power at isolated port of the `Directional coupler`, specified as a nonnegative and real scalar. The default value is `inf`.

Dependencies

To enable this parameter, select `Directional coupler` in **Select component** tab.

`Insertion loss (dB)` — Loss of signal power between input and output ports

`inf` (default) | nonnegative and real scalar

Loss of signal power between input and output ports of the `Directional coupler`, specified as a nonnegative and real scalar. The default value is `inf`.

Dependencies

To enable this parameter, select `Directional coupler` in **Select component** tab.

`Return loss (dB)` — Loss of signal power due to impedance mismatch

`inf` (default) | nonnegative and real scalar

Loss of signal power due to impedance mismatch of the `Directional coupler`, specified as a nonnegative, and real scalar. The default value is `inf`.

Dependencies

To enable this parameter, select `Directional coupler` in **Select component** tab.

`Power transmission coefficient` — Transmitted signal power

0 (default) | real scalar

Transmitted signal power of the `Directional coupler`, specified as a real scalar. The default value is 0.

Dependencies

To enable this parameter, select `Coupler symmetrical` or `Coupler antisymmetrical` in **Select component** tab.

`Reference impedances (Ohm)` — Reference impedance of coupler
`50` (default) | positive scalar | three-tuple

Reference impedance of coupler, specified as a scalar or three-tuple. The default value is `50 Ohms`.

`Ground and hide negative terminals` — Ground RF circuit terminals
`on` (default) | `off`

Select this parameter to ground and hide the negative terminals. To expose the negative terminals, clear this parameter. By exposing these terminals, you can connect them to other parts of your model.

By default, this option is selected.

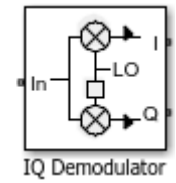
See Also

Circulator | Divider

Introduced in R2014a

IQ Demodulator

Convert RF signal to baseband signal
Library: RF Blockset / Circuit Envelope / Systems



Description

The `IQ Demodulator` converts an RF signal to baseband signal. `I` stands for the in-phase component of the signal and `Q` stands for the quadrature phase component of the signal. You can use the IQ Demodulator to design direct conversion receivers.

Parameters

Main

`Source of conversion gain` — Source parameter of conversion gain
`Available power gain` (default) | `Open circuit voltage gain` | `Polynomial coefficients`

Source parameter of conversion gain, specified as one of the following:

- `Available power gain` — Relates the ratio of the power of a single sideband (SSB) of the output `I` branch to the input power. If there is no gain mismatch, the gain at the `Q` branch matches the gain at the `I` branch.
- `Open circuit voltage gain` — Value of the open circuit voltage gain parameter as the linear voltage gain term of the polynomial voltage controlled voltage-source (VCVS).
- `Polynomial coefficients` — Implements a nonlinear voltage gain according to the polynomial you specify.

`Available power gain` — Ratio of power of SSB at output `I` branch to input power

`0` dB (default) | scalar in dB or a unitless ratio

Ratio of power of SSB at output `I` branch to input power, specified as a scalar in dB or a unitless ratio. For a unitless ratio, select `None`.

Dependencies

To enable this parameter, set **Source of conversion gain** to `Available power gain`.

`Open circuit voltage gain` — Open circuit voltage gain

`0` dB (default) | scalar

Open circuit voltage gain, specified as a scalar in dB.

Dependencies

To enable this parameter, set **Source of conversion gain** to `Open circuit voltage gain`.

`Polynomial coefficients` — Coefficients of polynomial specifying voltage gain

`[0 1]` (default) | vector

Polynomial coefficients, specified as a vector.

The order of the polynomial must be less than or equal to 9. The coefficients must be ordered in ascending powers. If a vector has 10 coefficients, $[a_0, a_1, a_2, \ldots a_9]$, the polynomial it represents is:

$V_{out} = a_0 + a_1 V_{in} + a_2 V_{in}^2 + \ldots + a_9 V_{in}^9$
a_1 represents the linear gain term, and higher-order terms are modeled according to [2].

For example, the vector $[a_0, a_1, a_2, a_3]$ specifies the relation $V_{out} = a_0 + a_1 V_1 + a_2 V_1^2 + a_3 V_1^3$. Trailing zeros are omitted. So $[a_0, a_1, a_2]$ defines the same polynomial as $[a_0, a_1, a_2, 0]$.

By default, the value is `[0,1]`, corresponding to the linear relation $V_{out} = V_{in}$.

Dependencies

To enable this parameter, set **Source of conversion gain** to `Polynomial coefficients`.

`Local oscillator frequency` — Local oscillator (LO) frequency
`0 Hz` (default) | scalar

Local oscillator (LO) frequency, specified as a scalar in `Hz`, `kHz`, `MHz`, or `GHz`.

`Input impedance (Ohm)` — Input impedance of IQ demodulator
`50` (default) | scalar

Input impedance of IQ demodulator, specified as a scalar in Ohms.

`Output impedance (Ohm)` — Output impedance of IQ demodulator
`50` (default) | scalar

Output impedance of IQ demodulator, specified as a scalar in Ohms.

`Add Image Reject filter` — Image reject (IR) filter parameters
`off` (default) | `on`

Select to add the **IR filter** parameter tab. Clear to remove the tab.

`Add Channel Select filters` — Channel select (CS) filter parameters
`off` (default) | `on`

Select to add the **CS filter** parameter tab. Clear to remove the tab.

`Ground and hide negative terminals` — Ground and hide circuit terminals
`on` (default) | `off`

Select to internally ground and hide the negative terminals. Clear to expose the negative terminals. When the terminals are exposed, you can connect them to other parts of your model.

`Edit System` — Break IQ demodulator block links and replace internal variables by appropriate values
button

Use this button to break IQ modulator links to the library. The internal variables are replaced by their values which are estimated using IQ modulator parameters. The IQ Modulator becomes a simple subsystem masked only to keep the icon.

Use **Edit System** to edit the internal variables without expanding the subsystem. Use **Expand System** to expand the subsystem in the Simulink canvas and to edit the subsystem.

Impairments

`I/Q gain mismatch` — Gain difference between `I` and `Q` branches
`0 dB` (default) | scalar

Gain difference between `I` and `Q` branches, specified as a scalar in dB. Gain mismatch is assumed to be forward-going, that is, the mismatch does not affect leakage from LO to RF.

If the gain mismatch is specified, the value (*Available power gain* + *I*/*Q gain mismatch*) relates the ratio of power of the single-sideband (SSB) at output the `Q` branch to the input power.

`I/Q phase mismatch` — Phase difference between `I` and `Q` branches
`0 degrees` (default) | scalar in degrees or radians

Phase difference between `I` and `Q` branches, specified as a scalar in degrees or radians. The phase mismatch affects the LO to input RF leakage.

`LO to RF isolation` — Ratio of magnitude between LO voltage to leaked RF voltage
`inf dB` (default) | scalar

Ratio of magnitude between LO voltage to leaked RF voltage, specified as a scalar in dB. Phase accumulation in the path from LO input to the internal `I` and `Q` mixers (after phase shift and phase mismatch) and then to the RF is assumed to be zero.

`Noise figure (dB)` — Signal-to-noise ratio (SNR) between outputs and input
`0` (default) | scalar

Single-sideband noise figure of mixer, specified as a scalar.

To model noise in circuit envelope model with a Noise, Amplifier, or Mixer, IQ Demodulator block, you must select the **Simulate noise** check box in the Configuration block dialog box.

The following table summarizes the two competing definitions for specifying SSB noise, where the image frequency (IM) is defined as $\omega_{IM} = \omega_{LO} + (\omega_{LO} - \omega_{RF})$.

Noise Convention	Signal at RF Frequency	Signal at IM Frequency	IQ Demodulator Block Supports This Model?
Single-sideband noise (SSB)	$S + N$, signal with noise	N, noise only	Yes
IEEE definition of single-sideband noise (SSB_{IEEE})	$S + N$, signal with noise	No signal	No; you can create an equivalent model using an ideal filter created from an S-parameters block.

`Add phase noise` — Add phase noise
`off` (default) | `on`

Select this parameter to add phase noise to your IQ demodulator system.

`Phase noise frequency offset (Hz)` — Phase noise frequency offset
`1` (default) | scalar | vector | matrix

Phase noise frequency offset, specified as a scalar, vector, or matrix with each element unit in Hz.

If you specify a matrix, each column corresponds to a non-DC carrier frequency of the CW source. The frequency offset values bind the envelope bandwidth of the simulation. For more information, see Configuration.

Dependencies

To enable this parameter, select **Add phase noise**.

`Phase noise level (dBc/Hz)` — Phase noise level
`-Inf` (default) | scalar | vector | matrix

Phase noise level, specified as a scalar, vector, or matrix with element unit in decibel per dBc/Hz.

If you specify a matrix, each column corresponds to a non-DC carrier frequency of the CW source. The frequency offset values bind the envelope bandwidth of the simulation. For more information, see Configuration.

Dependencies

To enable this parameter, select **Add phase noise**.

`Automatically estimate impulse response duration` — Automatically estimate impulse response duration
`on` (default) | `off`

Select to automatically estimate impulse response for phase noise. Clear to specify the impulse response duration using **Impulse response duration**.

`Impulse response duration` — Impulse response duration
`1e-10s` (default) | scalar

Impulse response duration used to simulate phase noise, specified as a scalar in s, ms, us, or ns.

Note The phase noise profile resolution in frequency is limited by the duration of the impulse response used to simulate it. Increase this duration to improve the accuracy of the phase noise profile. A warning message appears if the phase noise frequency offset resolution is too high for a given impulse response duration. This message also specifies the minimum duration suitable for the required resolution.

Dependencies

To set this parameter, clear **Automatically estimate impulse response duration**.

Nonlinearity

Selecting `Polynomial coefficients` for **Source of conversion gain** in the **Main** tab removes the **Nonlinearity** parameters.

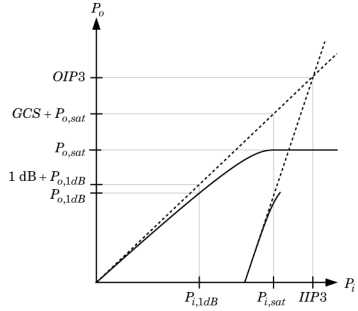
`Nonlinear polynomial type` — Polynomial nonlinearity
`Even and odd order` (default) | `Odd order`

Polynomial nonlinearity, specified as one of the following:

- `Even and odd order`: The IQ Demodulator can produce second-order and third-order intermodulation frequencies, in addition to a linear term.

- `Odd order`: The IQ Demodulator generates only "odd- order" intermodulation frequencies.

  The linear gain determines the linear a_1 term. The block calculates the remaining terms from the values specified in **IP3**, **1-dB gain compression power**, **Output saturation power**, and **Gain compression at saturation**. The number of constraints you specify determines the order of the model. The figure shows the graphical definition of the nonlinear IQ demodulator parameters.



`Intercept points convention` — Intercept points convention

`Input` (default) | `Output`

Intercept points convention, specified as `Input` (input-referred) or `Output` (output-referred). Use this specification for the intercept points **IP2**, **IP3**, the **1-dB gain compression power**, and the **Output saturation power**.

`IP2` — Second-order intercept point

`inf dBm` (default) | scalar

Second-order intercept point, specified as a scalar in dBm, W, mW, or dBW. The default value `inf dBm` corresponds to an unspecified point.

Dependencies

To enable this parameter, set **Nonlinear polynomial type** to `Even and odd order`.

`IP3` — Third-order intercept point

`inf dBm` (default) | scalar

Third-order intercept point, specified as a scalar in dBm, W, mW, or dBW. The default value `inf dBm` corresponds to an unspecified point.

Dependencies

To enable this parameter, set **Nonlinear polynomial type** to `Even and odd order`.

`1-dB gain compression power` — 1-dB gain compression power
`infdBm` (default) | scalar

1-dB gain compression power, specified as a scalar in dBm, W, mW, or dBW. The 1-dB gain compression point must be less than the output saturation power.

Dependencies

To enable this parameter, set `Odd order` in **Nonlinear polynomial type** tab.

`Output saturation power` — Output saturation power
`infdBm` (default) | scalar

Output saturation power, specified as a scalar. The block uses this value to calculate the voltage saturation point used in the nonlinear model. In this case, the first derivative of the polynomial is zero, and the second derivative is negative.

Dependencies

To enable this parameter, set `Odd order` in **Nonlinear polynomial type** tab.

`Gain compression at saturation` — Gain compression at saturation
`infdBm` (default) | scalar

Gain compression at saturation, specified as a scalar.

Dependencies

To enable this parameter, first select `Odd order` in **Nonlinear polynomial type** tab. Then change the default value of **Output saturation power**.

IR Filter

Select **Add Image Reject filter** in the **Main** tab to see the **IR Filter** parameters tab.

`Design method` — Simulation type
`Ideal` (default) | `Butterworth` | `Chebyshev`

Simulation type. Simulates an ideal, Butterworth, or Chebyshev filter of the type specified in **Filter type** and the model specified in **Implementation**.

`Filter type` — Filter type
`Lowpass` (default) | `Highpass` | `Bandpass` | `Bandstop`

Filter. Simulates a lowpass, highpass, bandpass, or bandstop filter type of the design specified in **Design method**

`Implementation` — Implementation
`LC Tee` | `LC Pi` | `Transfer function` | `Constant per carrier` | `Frequency Domain`

Implementation, specified as one of the following:

- `LC Tee`: Model an analog filter with an LC lumped Tee structure when the **Design method** is Butterworth or Chebyshev.
- `LC Pi`: Model an analog filter with an LC lumped Pi structure when the **Design method** is Butterworth or Chebyshev.
- `Transfer Function`: Model an analog filter using two-port S-parameters when the **Design method** is Butterworth or Chebyshev.
- `Constant per carrier`: Model a filter with either full transmission or full reflection set as constant throughout the entire envelope band around each carrier. The **Design method** is specified as ideal.
- `Filter Domain`: Model a filter using convolution with an impulse response. The **Design method** is specified as ideal. The impulse response is computed independently for each carrier frequency to capture the ideal filtering response. When a transition between full transmission and full reflection of the ideal filter occurs within the envelope band around a carrier, the frequency-domain implementation captures this transition correctly up to a frequency resolution specified in **Impulse response duration**.

  Note Due to causality, a delay of half the impulse response duration is included for both reflected and transmitted signals. This delay impairs the filter performance when the Source and Load resistances differ from the values specified in filter parameters.

By default, the **Implementation** is `Constant per carrier` for an ideal filter and `LC Tee` for Butterworth or Chebyshev.

`Passband edge frequency` — Passband edge frequency
`2 GHz` (default) | scalar

Passband edge frequency, specified as a scalar in Hz, kHz, MHz, or GHz.

Dependencies

To enable this parameter, set **Design method** to `Ideal` and **Filter type** to `Lowpass` or `Highpass`.

`Implement using filter order` — Implement using filter order

`on` (default) | `off`

Select this parameter to implement the filter order manually.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Filter order` — Filter order

3 (default) | scalar

Filter order, specified as a scalar. For a **Filter type** of `Lowpass` or `Highpass`, the filter order is the number of lumped storage elements. For a **Filter type** of `Bandpass` of `Bandstop`, the number of lumped storage elements is twice the filter order.

Note For even order Chebyshev filters, the resistance ratio $\frac{R_{load}}{R_{source}} > R_{ratio}$ for Tee network implementation and $\frac{R_{load}}{R_{source}} < \frac{1}{R_{ratio}}$ for Pi network implementation.

$$R_{ratio} = \frac{\sqrt{1+\varepsilon^2}+\varepsilon}{\sqrt{1+\varepsilon^2}-\varepsilon}$$

where:

- $\varepsilon = \sqrt{10^{(0.1R_p)}-1}$
- R_p is the passband ripple in dB.

Dependencies

To enable this parameter, select **Implement using filter order**.

`Passband frequency` — Passband frequency for lowpass and highpass filters

scalar

Passband frequency for lowpass and highpass filters, specified as a scalar in Hz, kHz, MHz, or GHz. The default value is `1 GHz` for `Lowpass` filters and `2 GHz` for `Highpass` filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Lowpass` or `Highpass`.

`Passband frequencies` — Passband frequencies for bandpass filters

`[2 3] GHz` (default) | 2-tuple vector

Passband frequencies for bandpass filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for bandstop filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandpass`.

`Passband attenuation (dB)` — Passband attenuation

`10*log10(2)` (default) | scalar

Passband attenuation, specified as a scalar in dB. For bandpass filters, this value is applied equally to both edges of the passband.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Stopband frequencies` — Stopband frequencies for bandstop filters

`[2.1 2.9] GHz` (default) | 2-tuple vector

Stopband frequencies for bandstop filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for bandpass filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandstop`.

`Stopband edge frequencies` — Stopband edge frequencies for ideal bandstop filters
`[2.1 2.9] GHz` (default) | 2-tuple vector

Stopband edge frequencies for bandstop filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for ideal bandpass filters.

Dependencies

To enable this parameter, set **Design method** to `Ideal` and **Filter type** to `Bandstop`.

`Stopband attenuation (dB)` — Stopband attenuation
`40` (default) | scalar

Stopband attenuation, specified as a scalar in dB. For bandstop filters, this value is applied equally to both edges of the stopband.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandstop`.

`Source impedance (Ohm)` — Input source resistance
`50` (default) | scalar

Input source resistance, specified as a scalar in Ohms.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Load impedance (Ohm)` — Output load resistance
`50` (default) | scalar

Output load resistance, specified as a scalar in Ohms.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Automatically estimate impulse response duration` — Automatically estimate impulse response duration
`on` (default) | `off`

Select to automatically estimate impulse response for phase noise. Clear to manually specify the impulse response duration using **Impulse response duration**.

Dependencies

To enable this parameter, set **Design method** to `Ideal` and **Implementation** to `Frequency domain`.

`Impulse response duration` — Impulse response duration

`1e-10s` (default) | scalar

Impulse response duration used to simulate phase noise, specified as a scalar in s, ms, us, or ns. You cannot specify impulse response if the amplifier is nonlinear.

Note The phase noise profile resolution in frequency is limited by the duration of the impulse response used to simulate it. Increase this duration to improve the accuracy of the phase noise profile. A warning message appears if the phase noise frequency offset resolution is too high for a given impulse response duration. This message also specifies the minimum duration suitable for the required resolution

---

Dependencies

To enable this parameter, clear **Automatically estimate impulse response duration**.

`Export` — Save filter design to a file

button

Use this button to save filter design to a file. Valid file types are `.mat` and `.txt`.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

CS Filter

Select **Add Channel Select filters** in the **Main** tab to see the **CS Filter** parameters.

`Design method` — Simulation type

`Ideal` (default) | `Butterworth` | `Chebyshev`

Simulation type. Simulates an ideal, Butterworth, or Chebyshev filter of the type specified in **Filter type** and the model specified in **Implementation**.

`Filter type` — Filter type
`Lowpass` (default) | `Highpass` | `Bandpass` | `Bandstop`

Filter. Simulates a lowpass, highpass, bandpass, or bandstop filter type of the design specified in **Design method**.

`Implementation` — Implementation
`LC Tee` | `LC Pi` | `Transfer function` | `Constant per carrier` | `Frequency Domain`

Implementation, specified as one of the following:

- `LC Tee`: Model an analog filter with an LC lumped Tee structure when the **Design method** is Butterworth or Chebyshev.
- `LC Pi`: Model an analog filter with an LC lumped Pi structure when the **Design method** is Butterworth or Chebyshev.
- `Transfer Function`: Model an analog filter using two-port S-parameters when the **Design method** is Butterworth or Chebyshev.
- `Constant per carrier`: Model a filter with either full transmission or full reflection set as constant throughout the entire envelope band around each carrier. The **Design method** is specified as ideal.
- `Filter Domain`: Model a filter using convolution with an impulse response. The **Design method** is specified as ideal. The impulse response is computed independently for each carrier frequency to capture the ideal filtering response. When a transition between full transmission and full reflection of the ideal filter occurs within the envelope band around a carrier, the frequency-domain implementation captures this transition correctly up to a frequency resolution specified in **Impulse response duration**.

  Note Due to causality, a delay of half the impulse response duration is included for both reflected and transmitted signals. This delay impairs the filter performance when the Source and Load resistances differ from the values specified in filter parameters.

By default, the **Implementation** is `Constant per carrier` for an ideal filter and `LC Tee` for Butterworth or Chebyshev.

`Passband edge frequency` — Passband edge frequency

`2 GHz` (default) | scalar

Passband edge frequency, specified as a scalar in Hz, kHz, MHz, or GHz.

Dependencies

To enable this parameter, set **Design method** to `Ideal`.

`Implement using filter order` — Implement using filter order

`on` (default) | `off`

Select this parameter to implement the filter order manually.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Filter order` — Filter order

3 (default) | scalar

Filter order, specified as a scalar. This order is the number of lumped storage elements in `lowpass` or `highpass`. In `bandpass` or `bandstop`, the number of lumped storage elements are twice the value.

Note For even order Chebyshev filters, the resistance ratio $\frac{R_{\text{load}}}{R_{\text{source}}} > R_{\text{ratio}}$ for Tee network implementation and $\frac{R_{\text{load}}}{R_{\text{source}}} < \frac{1}{R_{\text{ratio}}}$ for Pi network implementation.

$$R_{\text{ratio}} = \frac{\sqrt{1+\varepsilon^2}+\varepsilon}{\sqrt{1+\varepsilon^2}-\varepsilon}$$

where:

- $\varepsilon = \sqrt{10^{(0.1R_{\text{p}})}-1}$
- R_{p} is the passband ripple in dB.

Dependencies

To enable this parameter, select **Implement using filter order**.

`Passband frequency` — Passband frequency for lowpass and highpass filters

scalar

Passband frequency for lowpass and highpass filters, specified as a scalar in Hz, kHz, MHz, or GHz. By default, the passband frequency is `1 GHz` for `Lowpass` filters and `2 GHz` for `Highpass` filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Lowpass` or `Highpass`.

`Passband frequencies` — Passband frequencies for bandpass filters

`[2 3] GHz` (default) | 2-tuple vector

Passband frequencies for bandpass filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for bandstop filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandpass`.

`Passband attenuation (dB)` — Passband attenuation

`10*log10(2)` (default) | scalar

Passband attenuation, specified as a scalar in dB. For bandpass filters, this value is applied equally to both edges of the passband.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Stopband frequencies` — Stopband frequencies for bandstop filters

`[2.1 2.9] GHz` (default) | 2-tuple vector

Stopband frequencies for bandstop filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for bandpass filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandstop`.

`Stopband edge frequencies` — Stopband edge frequencies for ideal bandstop filters

`[2.1 2.9] GHz` (default) | 2-tuple vector

Stopband edge frequencies for bandstop filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for ideal bandpass filters.

Dependencies

To enable this parameter, set **Design method** to `Ideal` and **Filter type** to `Bandstop`.

`Stopband attenuation (dB)` — Stopband attenuation

`40` (default) | scalar

Stopband attenuation, specified as a scalar in dB. For bandstop filters, this value is applied equally to both edges of the stopband.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandstop`.

`Source impedance (Ohm)` — Input source resistance

`50` (default) | scalar

Input source resistance, specified as a scalar in Ohms.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Load impedance (Ohm)` — Output load resistance

`50` (default) | scalar

Output load resistance, specified as a scalar in Ohms.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Automatically estimate impulse response duration` — Automatically estimate impulse response duration
`on` (default) | `off`

Select to automatically estimate impulse response for phase noise. Clear to specify the impulse response duration using **Impulse response duration**.

Dependencies

set **Design method** to `Ideal` and **Implementation** to `Frequency domain`.

`Impulse response duration` — Impulse response duration
`1e-10s` (default) | scalar

Impulse response duration used to simulate phase noise, specified as a scalar in seconds. You cannot specify impulse response if the amplifier is nonlinear.

Note The phase noise profile resolution in frequency is limited by the duration of the impulse response used to simulate it. Increase this duration to improve the accuracy of the phase noise profile. A warning message appears if the phase noise frequency offset resolution is too high for a given impulse response duration. This message also specifies the minimum duration suitable for the required resolution

---

Dependencies

To enable this parameter, clear **Automatically estimate impulse response duration**.

`Export` — Save filter design to a file
button

Use this button to save filter design to a file. Valid file types are `.mat` and `.txt`.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

References

[1] Razavi, Behzad. *RF Microelectronics*. Upper Saddle River, NJ: Prentice Hall, 2011.

[2] Grob, Siegfried and Lindner, Jurgen, “Polynomial Model Derivation of Nonlinear Amplifiers”, *Department of Information Technology*, University of Ulm, Germany.

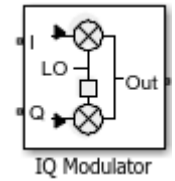
See Also

IQ Modulator | Mixer

Introduced in R2017a

IQ Modulator

Convert baseband signal to RF signal
Library: RF Blockset / Circuit Envelope / Systems



Description

The `IQ Modulator` converts a baseband signal to RF signal and models an IQ modulator with impairments. `I` stands for the in-phase component of the signal and `Q` stands for the quadrature phase component of the signal. You can use the IQ Modulator to design direct conversion transmitters.

Parameters

Main

`Source of conversion gain` — Source parameter of conversion gain
`Available power gain` (default) | `Open circuit voltage gain` | `Polynomial coefficients`

Source parameter of conversion gain, specified as one of the following:

- `Available power gain` — Relates the ratio of the power of a single sideband (SSB) of the output to the input power at the `I` branch. This assumes no gain mismatch and that the input at the Q branch is $Q_{in} = -j.I_{in}$
- `Open circuit voltage gain` — Value of the open circuit voltage gain parameter as the linear voltage gain term of the polynomial voltage-controlled voltage source (VCVS).
- `Polynomial coefficients` — Implements a nonlinear voltage gain according to the polynomial you specify.

`Available power gain` — Ratio of power of SSB at output to input power at `I`

`0 dB` (default) | scalar in dB or a unitless ratio

Ratio of the power of SSB at the output to input power at `I` branch, specified as a scalar in dB or a unitless ratio. For a unitless ratio, select `None`.

Dependencies

To enable this parameter, set **Source of conversion gain** to `Available power gain`.

`Open circuit voltage gain` — Open circuit voltage gain

`0 dB` (default) | scalar in dB or a unitless ratio

Open circuit voltage of IQ modulator, specified as a scalar in dB or a unitless ratio. For a unitless ratio, select `None`.

Dependencies

To enable this parameter, set **Source of conversion gain** to `Open circuit voltage gain`.

`Polynomial coefficients` — Coefficients of polynomial specifying voltage gain

`[0 1]` (default) | vector

Polynomial coefficients, specified as a vector.

The order of the polynomial must be less than or equal to 9. The coefficients must be ordered in ascending powers. If a vector has 10 coefficients, $[a_0, a_1, a_2, \ldots a_9]$, the polynomial it represents is:

$V_{out} = a_0 + a_1 V_{in} + a_2 V_{in}^2 + \ldots + a_9 V_{in}^9$
a_1 represents the linear gain term, and higher-order terms are modeled according to [2].

For example, the vector $[a_0, a_1, a_2, a_3]$ specifies the relation
$V_{out} = a_0 + a_1 V_1 + a_2 V_1^2 + a_3 V_1^3$. Trailing zeros are omitted. So $[a_0, a_1, a_2]$ defines the same polynomial as $[a_0, a_1, a_2, 0]$.

By default, the value is [0,1], corresponding to the linear relation $V_{out} = V_{in}$.

Dependencies

To enable this parameter, set **Source of conversion gain** to `Polynomial coefficients`.

`Local oscillator frequency` — Local oscillator (LO) frequency
`0 Hz` (default) | scalar

Local oscillator (LO) frequency, specified as a scalar in Hz, kHz, MHz, or GHz.

`Input impedance (Ohm)` — Input impedance of IQ modulator
`50` (default) | scalar

Input impedance of IQ modulator, specified as a scalar in Ohms.

`Output impedance (Ohm)` — Output impedance of IQ modulator
`50` (default) | scalar

Output impedance of IQ modulator, specified as a scalar in Ohms.

`Add Image Reject filters` — Image reject (IR) filter parameters
`off` (default) | `on`

Select to add the **IR filter** parameter tab. Clear to remove the tab.

`Add Channel Select filter` — Channel select (CS) filter parameters
`off` (default) | `on`

Select to add the **CS filter** parameter tab. Clear to remove the tab.

`Ground and hide negative terminals` — Ground and hide terminals
`on` (default) | `off`

Select to internally ground and hide the negative terminals. Clear to expose the negative terminals. When the terminals are exposed, you can connect them to other parts of your model.

`Edit System` — Break IQ modulator block links and replace internal variables by appropriate values
button

Use this button to break IQ modulator links to the library. The internal variables are replaced by their values which are estimated using IQ modulator parameters. The IQ Modulator becomes a simple subsystem masked only to keep the icon.

Use **Edit System** to edit the internal variables without expanding the subsystem. Use **Expand System** to expand the subsystem in the Simulink canvas and to edit the subsystem.

Impairments

`I/Q gain mismatch` — Gain difference between `I` and `Q` branches
`0 dB` (default) | scalar

Gain difference between `I` and `Q` branches, specified as a scalar in dB, or a unitless ratio. Gain mismatch is assumed to be forward-going, that is, the mismatch does not affect leakage from LO to RF.

If the gain mismatch is specified, the value (*Available power gain* + *I*/*Q gain mismatch*) relates the ratio of power of the single-sideband (SSB) at the `Q` input branch to the output power.

`I/Q phase mismatch` — Phase difference between `I` and `Q` branches
`0 degrees` (default) | scalar in degrees or radians

Phase difference between `I` and `Q` branches, specified as a scalar in degrees or radians. This mismatch affects the LO to input RF leakage.

`LO to RF isolation` — Ratio of magnitude between LO voltage to leaked RF voltage
`inf dB` (default) | scalar

Ratio of magnitude between LO voltage to leaked RF voltage, specified as a scalar in dB, or a unitless ratio. For a unitless ratio, select `None`.

`Noise floor (dBm/Hz)` — Single-sided noise power spectral distribution
`-inf` (default) | scalar in dBm/Hz

Single-sided noise power spectral distribution, specified as a scalar in dBm/Hz. This block assumes -174dBm/Hz noise input at both `I` and `Q` branches.

`Add phase noise` — Add phase noise
`off` (default) | `on`

Select this parameter to add phase noise to your IQ modulator system.

`Phase noise frequency offset (Hz)` — Phase noise frequency offset
`1` (default) | scalar | vector | matrix

Phase noise frequency offset, specified as a scalar, vector, or matrix with each element unit in Hz.

If you specify a matrix, each column corresponds to a non-DC carrier frequency of the CW source. The frequency offset values bind the envelope bandwidth of the simulation. For more information, see Configuration.

Dependencies

To enable this parameter, select **Add phase noise**.

`Phase noise level (dBc/Hz)` — Phase noise level

`-Inf` (default) | scalar | vector | matrix

Phase noise level, specified as a scalar, vector, or matrix with element unit in dBc/Hz.

If you specify a matrix, each column corresponds to a non-DC carrier frequency of the CW source. The frequency offset values bind the envelope bandwidth of the simulation. For more information, see Configuration.

Dependencies

To enable this parameter, select **Add phase noise**.

`Automatically estimate impulse response duration` — Automatically estimate impulse response duration

`on` (default) | `off`

Select to automatically estimate impulse response for phase noise. Clear to specify the impulse response duration using **Impulse response duration**.

`Impulse response duration` — Impulse response duration

`1e-10 s` (default) | scalar

Impulse response duration used to simulate phase noise, specified as a scalar in s, ms, us, or ns.

Note The phase noise profile resolution in frequency is limited by the duration of the impulse response used to simulate it. Increase this duration to improve the accuracy of the phase noise profile. A warning message appears if the phase noise frequency offset resolution is too high for a given impulse response duration. This message also specifies the minimum duration suitable for the required resolution.

Dependencies

To set this parameter, clear **Automatically estimate impulse response duration**.

Nonlinearity

Selecting `Polynomial coefficients` for **Source of conversion gain** in the **Main** tab removes the **Nonlinearity** parameters.

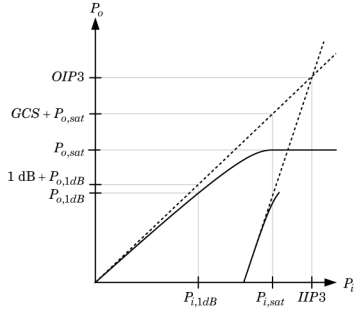
`Nonlinear polynomial type` — Polynomial nonlinearity

`Even and odd order` (default) | `Odd order`

Polynomial nonlinearity, specified as one of the following:

- `Even and odd order`: The IQ Modulator can produce second-order and third-order intermodulation frequencies, in addition to a linear term.
- `Odd order`: The IQ Modulator generates only "odd-order" intermodulation frequencies.

  The linear gain determines the linear a_1 term. The block calculates the remaining terms from the values specified in **IP3**, **1-dB gain compression power**, **Output saturation power**, and **Gain compression at saturation**. The number of constraints you specify determines the order of the model. The figure shows the graphical definition of the nonlinear IQ modulator parameters.



`Intercept points convention` — Intercept points convention

`Output` (default) | `Input`

Intercept points convention, specified as `Input` (input-referred) or `Output` (output-referred). Use this specification for the intercept points **IP2**, **IP3**, the **1-dB gain compression power**, and the **Output saturation power**.

`IP2` — Second-order intercept point

`inf dBm` (default) | scalar

Second-order intercept point, specified as a scalar in dBm, W, mW, or dBW. The default value `inf dBm` corresponds to an unspecified point.

Dependencies

To enable this parameter, set **Nonlinear polynomial type** to `Even and odd order`.

`IP3` — Third-order intercept point

`inf dBm` (default) | scalar

Third-order intercept point, specified as a scalar in dBm, W, mW, or dBW. The default value `inf dBm` corresponds to an unspecified point.

Dependencies

To enable this parameter, set **Nonlinear polynomial type** to `Even and odd order`.

`1-dB gain compression power` — 1-dB gain compression power

`inf dBm` (default) | scalar

1-dB gain compression power, specified as a scalar in dBm, W, mW, or dBW. The 1-dB gain compression point must be less than the output saturation power.

Dependencies

To enable this parameter, set `Odd order` in **Nonlinear polynomial type** tab.

`Output saturation power` — Output saturation power

`inf dBm` (default) | scalar

Output saturation power, specified as a scalar. The block uses this value to calculate the voltage saturation point used in the nonlinear model. In this case, the first derivative of the polynomial is zero, and the second derivative is negative.

Dependencies

To enable this parameter, set `Odd order` in **Nonlinear polynomial type** tab.

`Gain compression at saturation` — Gain compression at saturation

`inf dBm` (default) | scalar

Gain compression at saturation, specified as a scalar.

Dependencies

To enable this parameter, first select `Odd order` in **Nonlinear polynomial type** tab. Then change the default value of **Output saturation power** .

IR Filter

Select **Add Image Reject filters** in the **Main** tab to see the **IR Filter** parameters tab.

`Design method` — Simulation type
`Ideal` (default) | `Butterworth` | `Chebyshev`

Simulation type. Simulates an ideal, Butterworth, or Chebyshev filter of the type specified in **Filter type** and the model specified in **Implementation**.

`Filter type` — Filter type
`Lowpass` (default) | `Highpass` | `Bandpass` | `Bandstop`

Filter. Simulates a lowpass, highpass, bandpass, or bandstop filter type of the design specified in **Design method**

`Implementation` — Implementation
`LC Tee` | `LC Pi` | `Transfer function` | `Constant per carrier` | `Frequency Domain`

Implementation, specified as one of the following:

- `LC Tee`: Model an analog filter with an LC lumped Tee structure when the **Design method** is Butterworth or Chebyshev.
- `LC Pi`: Model an analog filter with an LC lumped Pi structure when the **Design method** is Butterworth or Chebyshev.
- `Transfer Function`: Model an analog filter using two-port S-parameters when the **Design method** is Butterworth or Chebyshev.
- `Constant per carrier`: Model a filter with either full transmission or full reflection set as constant throughout the entire envelope band around each carrier. The **Design method** is specified as ideal.
- `Filter Domain`: Model a filter using convolution with an impulse response. The **Design method** is specified as ideal. The impulse response is computed independently for each carrier frequency to capture the ideal filtering response. When a transition between full transmission and full reflection of the ideal filter occurs

within the envelope band around a carrier, the frequency-domain implementation captures this transition correctly up to a frequency resolution specified in **Impulse response duration**.

Note Due to causality, a delay of half the impulse response duration is included for both reflected and transmitted signals. This delay impairs the filter performance when the Source and Load resistances differ from the values specified in filter parameters.

By default, the **Implementation** is `Constant per carrier` for an ideal filter and `LC Tee` for Butterworth or Chebyshev.

`Passband edge frequency` — Passband edge frequency

`2 GHz` (default) | scalar

Passband edge frequency, specified as a scalar in Hz, kHz, MHZ, or GHz.

Dependencies

To enable this parameter, set **Design method** to `Ideal` and **Filter type** to `Lowpass` or `Highpass`.

`Implement using filter order` — Implement using filter order

`on` (default) | `off`

Select this parameter to implement the filter order manually.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Filter order` — Filter order

3 (default) | scalar

Filter order, specified as a scalar. For a **Filter type** of `Lowpass` or `Highpass`, the filter order is the number of lumped storage elements. For a **Filter type** of `Bandpass` of `Bandstop`, the number of lumped storage elements is twice the filter order.

Note For even order Chebyshev filters, the resistance ratio $\frac{R_{load}}{R_{source}} > R_{ratio}$ for Tee network implementation and $\frac{R_{load}}{R_{source}} < \frac{1}{R_{ratio}}$ for Pi network implementation.

$$R_{\text{ratio}} = \frac{\sqrt{1+\varepsilon^2}+\varepsilon}{\sqrt{1+\varepsilon^2}-\varepsilon}$$

where:

- $\varepsilon = \sqrt{10^{(0.1R_p)}-1}$
- R_p is the passband ripple in dB.

Dependencies

To enable this parameter, select **Implement using filter order**.

`Passband frequency` — Passband frequency for lowpass and highpass filters

scalar

Passband frequency for lowpass and highpass filters, specified as a scalar in Hz, kHz, MHz, or GHz. The default value is `1 GHz` for `Lowpass` filters and `2 GHz` for `Highpass` filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Lowpass` or `Highpass`.

`Passband frequencies` — Passband frequencies for bandpass filters

`[2 3] GHz` (default) | 2-tuple vector

Passband frequencies for bandpass filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for bandstop filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandpass`.

`Passband attenuation (dB)` — Passband attenuation

`10*log10(2)` (default) | scalar

Passband attenuation, specified as a scalar in dB. For bandpass filters, this value is applied equally to both edges of the passband.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Stopband frequencies` — Stopband frequencies for bandstop filters

`[2.1 2.9] GHz` (default) | 2-tuple vector

Stopband frequencies for bandstop filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for bandpass filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandstop`.

`Stopband edge frequencies` — Stopband edge frequencies for ideal bandstop filters

`[2.1 2.9] GHz` (default) | 2-tuple vector

Stopband edge frequencies for bandstop filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for ideal bandpass filters.

Dependencies

To enable this parameter, set **Design method** to `Ideal` and **Filter type** to `Bandstop`.

`Stopband attenuation (dB)` — Stopband attenuation

`40` (default) | scalar

Stopband attenuation, specified as a scalar in dB. For bandstop filters, this value is applied equally to both edges of the stopband.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandstop`.

`Source impedance (Ohm)` — Input source resistance

`50` (default) | scalar

Input source resistance, specified as a scalar in Ohms.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Load impedance (Ohm)` — Output load resistance

`50` (default) | scalar

Output load resistance, specified as a scalar in Ohms.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Automatically estimate impulse response duration` — Automatically estimate impulse response duration

`on` (default) | `off`

Select to automatically estimate impulse response for phase noise. Clear to manually specify the impulse response duration using **Impulse response duration**.

Dependencies

To enable this parameter, set **Design method** to `Ideal` and **Implementation** to `Frequency domain`.

`Impulse response duration` — Impulse response duration

`1e-10 s` (default) | scalar

Impulse response duration used to simulate phase noise, specified as a scalar in s, ms, us, or ns. You cannot specify impulse response if the amplifier is nonlinear.

Note The phase noise profile resolution in frequency is limited by the duration of the impulse response used to simulate it. Increase this duration to improve the accuracy of the phase noise profile. A warning message appears if the phase noise frequency offset resolution is too high for a given impulse response duration. This message also specifies the minimum duration suitable for the required resolution

---

Dependencies

To enable this parameter, clear **Automatically estimate impulse response duration**.

`Export` — Save filter design to a file

button

Use this button to save filter design to a file. Valid file types are `.mat` and `.txt`.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

CS Filter

Select **Add Channel Select filter** in the **Main** tab to see the **CS Filter** parameters.

`Design method` — Simulation type
`Ideal` (default) | `Butterworth` | `Chebyshev`

Simulation type. Simulates an ideal, Butterworth, or Chebyshev filter of the type specified in **Filter type** and the model specified in **Implementation**.

`Filter type` — Filter type
`Lowpass` (default) | `Highpass` | `Bandpass` | `Bandstop`

Filter. Simulates a lowpass, highpass, bandpass, or bandstop filter type of the design specified in **Design method**.

`Implementation` — Implementation
`LC Tee` | `LC Pi` | `Transfer function` | `Constant per carrier` | `Frequency Domain`

Implementation, specified as one of the following:

- `LC Tee`: Model an analog filter with an LC lumped Tee structure when the **Design method** is Butterworth or Chebyshev.
- `LC Pi`: Model an analog filter with an LC lumped Pi structure when the **Design method** is Butterworth or Chebyshev.
- `Transfer Function`: Model an analog filter using two-port S-parameters when the **Design method** is Butterworth or Chebyshev.
- `Constant per carrier`: Model a filter with either full transmission or full reflection set as constant throughout the entire envelope band around each carrier. The **Design method** is specified as ideal.
- `Filter Domain`: Model a filter using convolution with an impulse response. The **Design method** is specified as ideal. The impulse response is computed independently for each carrier frequency to capture the ideal filtering response. When a transition between full transmission and full reflection of the ideal filter occurs within the envelope band around a carrier, the frequency-domain implementation

captures this transition correctly up to a frequency resolution specified in **Impulse response duration**.

Note Due to causality, a delay of half the impulse response duration is included for both reflected and transmitted signals. This delay impairs the filter performance when the Source and Load resistances differ from the values specified in filter parameters.

By default, the **Implementation** is `Constant per carrier` for an ideal filter and `LC Tee` for Butterworth or Chebyshev.

`Passband edge frequency` — Passband edge frequency

`2 GHz` (default) | scalar

Passband edge frequency, specified as a scalar in Hz, kHz, MHz, or GHz.

Dependencies

To enable this parameter, set **Design method** to `Ideal`.

`Implement using filter order` — Implement using filter order

`on` (default) | `off`

Select this parameter to implement the filter order manually.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Filter order` — Filter order

3 (default) | scalar

Filter order, specified as a scalar. This order is the number of lumped storage elements in `lowpass` or `highpass`. In `bandpass` or `bandstop`, the number of lumped storage elements are twice the value.

Note For even order Chebyshev filters, the resistance ratio $\frac{R_{\text{load}}}{R_{\text{source}}} > R_{\text{ratio}}$ for Tee network implementation and $\frac{R_{\text{load}}}{R_{\text{source}}} < \frac{1}{R_{\text{ratio}}}$ for Pi network implementation.

$$R_{\text{ratio}} = \frac{\sqrt{1+\varepsilon^2}+\varepsilon}{\sqrt{1+\varepsilon^2}-\varepsilon}$$

where:

- $\varepsilon = \sqrt{10^{(0.1R_p)} - 1}$
- R_p is the passband ripple in dB.

Dependencies

To enable this parameter, select **Implement using filter order**.

`Passband frequency` — Passband frequency for lowpass and highpass filters

scalar

Passband frequency for lowpass and highpass filters, specified as a scalar in Hz, kHz, MHz, or GHz. By default, the passband frequency is `1 GHz` for `Lowpass` filters and `2 GHz` for `Highpass` filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Lowpass` or `Highpass`.

`Passband frequencies` — Passband frequencies for bandpass filters

`[2 3] GHz` (default) | 2-tuple vector

Passband frequencies for bandpass filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for bandstop filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandpass`.

`Passband attenuation (dB)` — Passband attenuation

`10*log10(2)` (default) | scalar

Passband attenuation, specified as a scalar in dB. For bandpass filters, this value is applied equally to both edges of the passband.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Stopband frequencies` — Stopband frequencies for bandstop filters

`[2.1 2.9] GHz` (default) | 2-tuple vector

Stopband frequencies for bandstop filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for bandpass filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandstop`.

`Stopband edge frequencies` — Stopband edge frequencies for ideal bandstop filters

`[2.1 2.9] GHz` (default) | 2-tuple vector

Stopband edge frequencies for bandstop filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for ideal bandpass filters.

Dependencies

To enable this parameter, set **Design method** to `Ideal` and **Filter type** to `Bandstop`.

`Stopband attenuation (dB)` — Stopband attenuation

`40` (default) | scalar

Stopband attenuation, specified as a scalar in dB. For bandstop filters, this value is applied equally to both edges of the stopband.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandstop`.

`Source impedance (Ohm)` — Input source resistance

`50` (default) | scalar

Input source resistance, specified as a scalar in Ohms.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Load impedance (Ohm)` — Output load resistance

`50` (default) | scalar

Output load resistance, specified as a scalar in Ohms.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Automatically estimate impulse response duration` — Automatically estimate impulse response duration

`on` (default) | `off`

Select to automatically estimate impulse response for phase noise. Clear to specify the impulse response duration using **Impulse response duration**.

Dependencies

To enable this parameter, set **Design method** to `Ideal` and **Implementation** to `Frequency domain`.

`Impulse response duration` — Impulse response duration

`1e-10s` (default) | scalar

Impulse response duration used to simulate phase noise, specified as a scalar in s, ms, us, or ns. You cannot specify impulse response if the amplifier is nonlinear.

Note The phase noise profile resolution in frequency is limited by the duration of the impulse response used to simulate it. Increase this duration to improve the accuracy of the phase noise profile. A warning message appears if the phase noise frequency offset resolution is too high for a given impulse response duration. This message also specifies the minimum duration suitable for the required resolution

---

Dependencies

To set this parameter, clear **Automatically estimate impulse response duration**.

`Export` — Save filter design to a file

button

Use this button to save filter design to a file. Valid file types are `.mat` and `.txt`.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

References

[1] Razavi, Behzad. *RF Microelectronics*. Upper Saddle River, NJ: Prentice Hall, 2011.

[2] Grob, Siegfried and Lindner, Jurgen, “Polynomial Model Derivation of Nonlinear Amplifiers”, *Department of Information Technology*, University of Ulm, Germany.

See Also

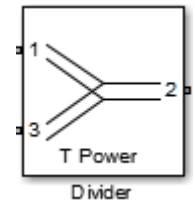
IQ Demodulator | Mixer | Modulator

Introduced in R2017a

Divider

Model ideal frequency-independent dividers (combiners) with S-parameters

Library: RF Blockset / Circuit Envelope / Junctions



Description

The Divider block models power dividers (combiners) in a circuit envelope environment as an ideal s-parameter model.

Parameters

`Select component` — Divider type

`T power divider` (default) | `Resistive power divider` | `Wlkinson power divider` | `Tee H-plane (S33=0)` | `Tee E-plane`

Divider type, specified as:

- `T power divider`

  The s-parameter matrix for `T-power divider` is :

  $$\begin{bmatrix} s_{11} & s_{21} & s_{31} \\ s_{21} & s_{22} & s_{32} \\ s_{31} & s_{32} & s_{33} \end{bmatrix}$$

  where:

  - $s_{11} = (z_{23} - z_1) / (z_{23} + z_1)$
  - $s_{22} = (z_{13} - z_2) / (z_{13} + z_2)$

- $s_{33} = (z_{12} - z_3) / (z_{12} + z_3)$
- $s_{21} = (1 + s_{11}) * \mathrm{sqrt}(z_1 / z_2)$
- $s_{31} = (1 + s_{11}) * \mathrm{sqrt}(z_1 / z_3)$
- $s_{32} = (1 + s_{22}) * \mathrm{sqrt}(z_2 / z_3)$
- $z_{12} = z_1 * z_2 / (z_1 + z_2)$
- $z_{13} = z_1 * z_3 / (z_1 + z_3)$
- $z_{23} = z_2 * z_3 / (z_2 + z_3)$
- $z_1 = Z0(1)$, $z_2 = Z0(2)$, $z_3 = Z0(3)$
- Reference Impedances: $Z0 = [z_1, z_2, z_3]$

- `Resistive power divider`

  The s-parameter matrix for `Resistive power divider` is :

  $$\begin{bmatrix} 0 & 1/2 & 1/2 \\ 1/2 & 0 & 1/2 \\ 1/2 & 1/2 & 0 \end{bmatrix}$$

  When you select this option, the following parameter is available:

- `Wilkinson power divider`

  The s-parameter matrix for `Wilkinson power divider` is:

  $$\begin{bmatrix} 0 & -j/\sqrt{2} & -j/\sqrt{2} \\ -j/\sqrt{2} & 0 & 0 \\ -j/\sqrt{2} & 0 & 0 \end{bmatrix}$$

  Note For DC carrier (0 Hz) frequency, Wilkinson power divider is a zero matrix.

- `Tee H-plane (S33=0)`

  The `Tee H-plane (S33=0)` is symmetrical, lossless, and reciprocal. The s-parameter matrix is:

$$\frac{\begin{pmatrix} -1 & 1 & \sqrt{2} \\ 1 & -1 & \sqrt{2} \\ \sqrt{2} & \sqrt{2} & 0 \end{pmatrix}}{2}$$

- `Tee E-plane`

  The `Tee E-plane` is symmetrical, lossless, and reciprocal. The s-parameter matrix is:

$$\frac{\begin{pmatrix} 1 & 1 & \sqrt{2} \\ 1 & 1 & -\sqrt{2} \\ \sqrt{2} & -\sqrt{2} & 0 \end{pmatrix}}{2}$$

`Reference impedances (Ohm)` — Reference impedance of divider
`50` (default) | positive scalar | three-tuple vector

Reference impedance of divider, specified as a positive scalar or three-tuple vector.

`Ground and hide negative terminals` — Ground RF circuit terminals
`on` (default) | `off`

Select this parameter to ground and hide the negative terminals. To expose the negative terminals, clear the parameter. By exposing these terminals, you can connect them to other parts of your model.

By default, this option is selected.

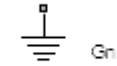
See Also

Circulator | Coupler

Introduced in R2014a

Ground

Simulate connection to electrical ground
Library: RF Blockset / Circuit Envelope / Elements



Description

The Ground block represents an electrical ground in a RF Blockset circuit envelope simulation environment. Connect at least one Ground to the RF Blockset environment; otherwise, models with RF Blockset blocks do not run.

Parameters

The Ground block has no parameters.

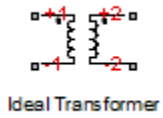
All models in the RF Blockset documentation contain a Ground block. See Model an RF Mixer for an introduction to circuit envelope simulation.

See Also

Introduced in R2016b

Ideal Transformer

Model ideal transformer



Library

Elements

Description

The Ideal Transformer block models a constant IV relationship within the RF Blockset circuit envelope simulation environment. For an introduction to RF simulation, see the example, “Simulate High Frequency Components”.

Parameters

Winding ratio

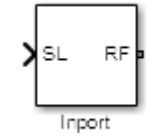
Specify the winding ratio.

See Also

Mutual Inductor | Three Winding Transformer

Inport

Convert Simulink input signal to RF Blockset signal
Library: RF Blockset / Circuit Envelope / Utilities



Description

The Inport block imports Simulink signals into the RF Blockset circuit envelope simulation environment. For an introduction to RF simulation, see the example, “Simulate High Frequency Components”.

Complex-valued input signals $I_k(t) + j \cdot Q_k(t)$ are the modulations at the frequencies $\{f_k\}$ specified in the **Carrier frequencies** parameter of the block.

The input port converts the complex simulink input signals into an Rf signal suitable for multicarrier simulation:

$$out = \mathrm{Re}\,al(\sum_{k=1}^{N}(I_k(t) + j \bullet Q_k(t)) \bullet e^{j2\pi f_k t})$$

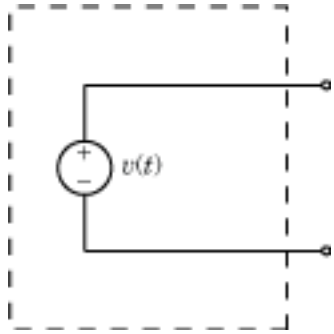
The **Source type** parameter specifies the Simulink signal as either current, or voltage, or power source.

Parameters

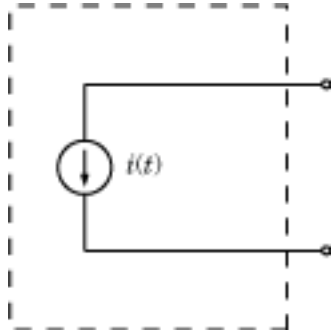
`Source type` — Inport block interpretation of Simulink signal
`Ideal voltage` (default) | `Ideal current` | `Power`

Inport block interpretation of Simulink signal, specified as:

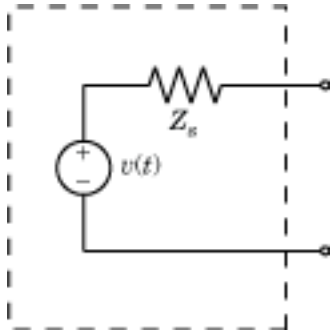
- `Ideal voltage` — The block outputs Simulink signals as voltage signals $v(t)$ in the RF Blockset environment. When you choose an ideal voltage input port you need to manually add a series of source impedance to match the blocks connected to the input port. The following figure illustrates the internal configuration of the block.



- `Ideal current` — The block outputs Simulink signals as current signals $i(t)$ in the RF Blockset environment. When you use an ideal current input port, you manually add a parallel source impedance to match the blocks connected to the input port. The following figure illustrates the internal configuration of the block.



- `Power` — The block interprets the Simulink signals, $P_v(t)$, as available power and internally uses a voltage source, and series impedance. You use this option to import a 50 ohm environment or different reference impedance signals created using Communications Toolbox™. When you select this option, the input port automatically inserts a source impedance in your circuit as shown in the following figure.



The voltage $v(t)$ is a scaling of the Simulink signal $v_{SL}(t)$:

$$v(t) = 2\sqrt{\mathrm{Re}(Z_s)}v_{SL}(t)$$

In the preceding equation, Z_s is the value of the **Source impedance (ohms)** parameter.

The generator delivers real power to the load Z_l:

$$P_{load} = |v(t)|^2 \frac{\mathrm{Re}(Z_s)}{|Z_s + Z_l|^2}$$

When $Z_l = Z_s^*$, this generator delivers the available power $|v_{SL}(t)|^2$.

`Source impedance (Ohm)` — Source impedance for available power match

`50` (default) | vector of positive integers in ohms.

Dependencies

To enable this parameter, select `Power` in **Source type**.

`Carrier frequencies` — Carrier frequencies

`0 Hz` (default) | vector of positive integers in Hz

Carrier frequencies, specified as a vector of positive integers in Hz. In carrier frequencies, the elements are a combination of fundamental tones and corresponding harmonics in the Configuration block.

`Ground and hide negative terminals` — Ground RF circuit terminals

`on` (default) | `off`

Select this parameter to ground and hide the negative terminals. Clear the parameter to expose the negative terminals. By exposing these terminals, you can connect them to other parts of your model.

By default, this option is selected.

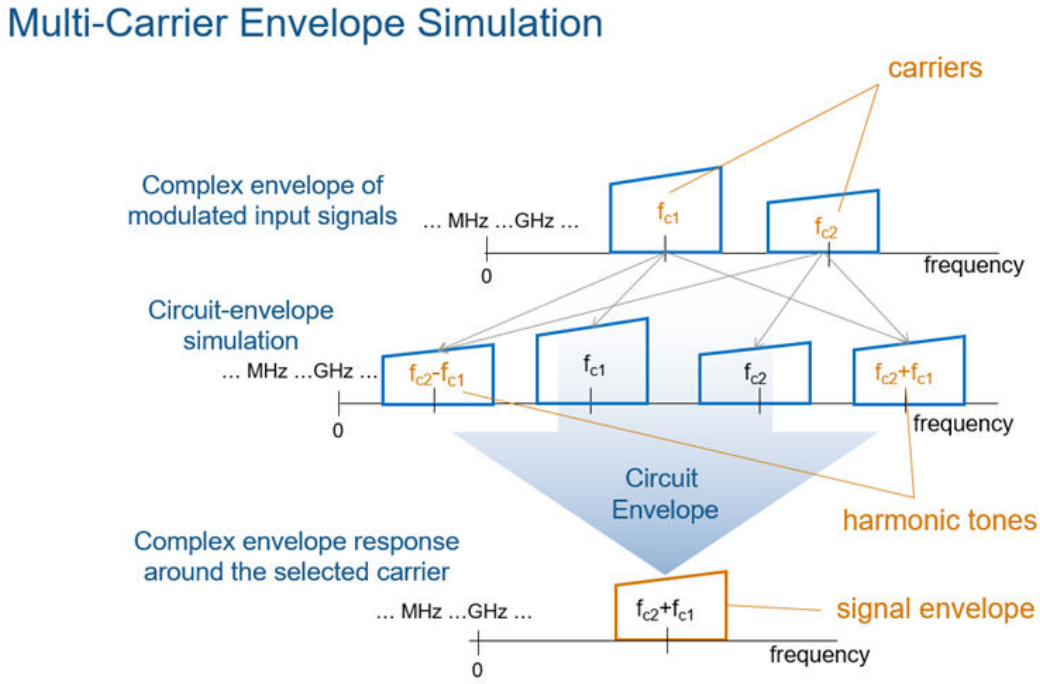
More About

Multi-Carrier Envelope Simulation

Using the Inport block you can specify the complex envelopes of your input signals and import them as RF signals for multi-carrier simulation.

The Configuration block automatically determines the fundamental tones specified in the input ports and proposes a suitable harmonic order to capture the non-linearity of the system. You can also manually specify the harmonic order for each fundamental tone in the simulation.

In the input port, you can specify as many carrier frequencies as you want. It is recommended that you trade off the simulation bandwidth (inversely proportional to the simulation time step) and the total number of simulation frequencies.



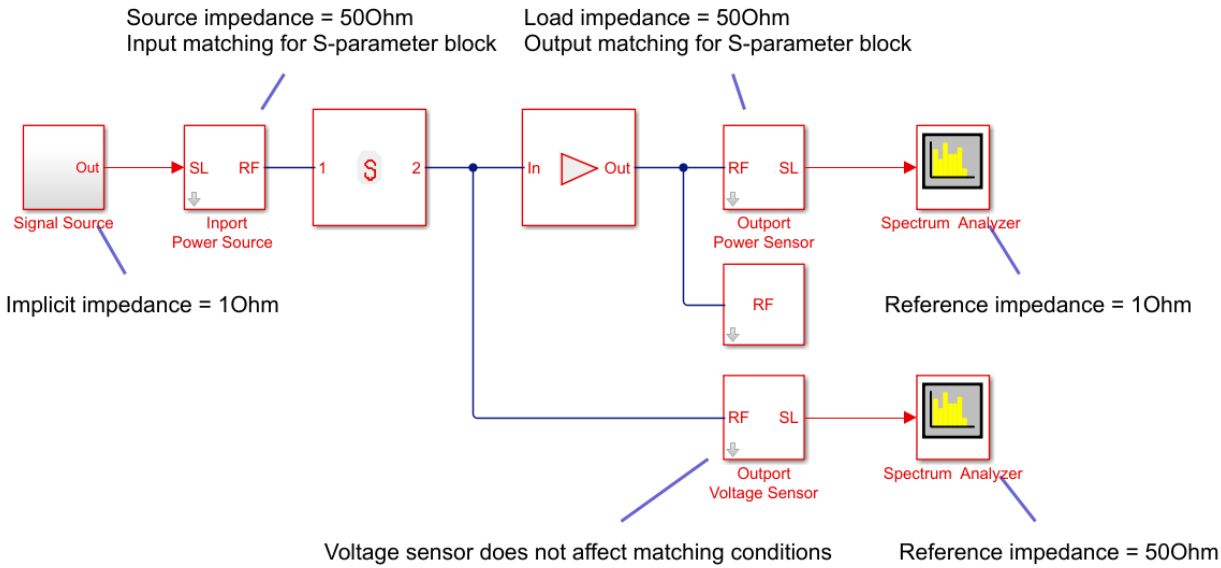
Algorithms

How to Use Inport Block

Physical Interpretation of Simulink Signals

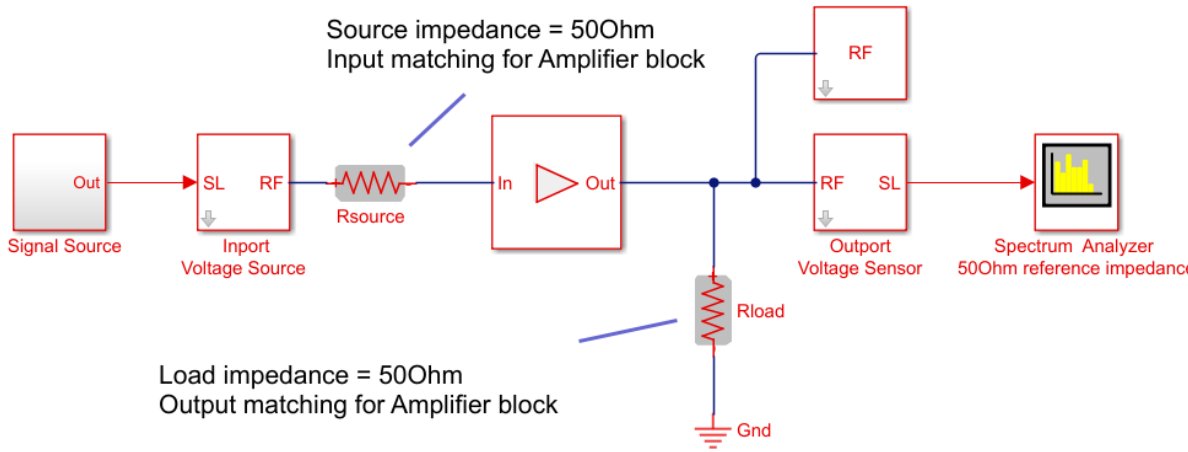
The Inport block allows you to specify the complex envelopes of your input signals and import them as RF signals for multi-carrier simulation.

The power option automatically insert a source or load impedance in your network, and normalizes the signal power with respect to the specified impedance. You do not need to manually insert source and load terminations, and your signals are automatically scaled between RF Blockset and the Simulink environment that assumes an implicit 1 Ohm reference impedance.



When using voltage sources and sensors, manually add source and load terminations, otherwise there might be an undesired impedance mismatch in your network. When you measure the power of a voltage signal, make sure that you use a 50Ohm reference impedance.

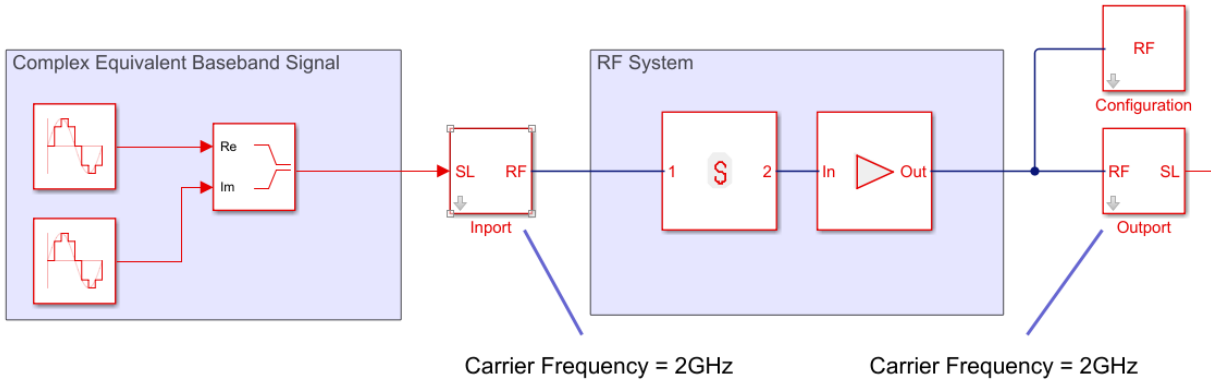
If you use an ideal voltage source and add a source impedance, in perfectly matched conditions, the actual voltage applied to the first block of the RF chain is half of the value of the input Simulink signal. The source impedance and the input impedance of the first block of the RF chain form a voltage divider network.



Implicit Carrier for Complex Equivalent Baseband Signal

Input signal is a digital communication complex equivalent baseband signal (I,Q). You assume an implicit carrier for the system that is equal to the carrier frequency, F_c. You want to model RF effects such as amplifier nonlinearity and S-parameter filters using RF Blockset:

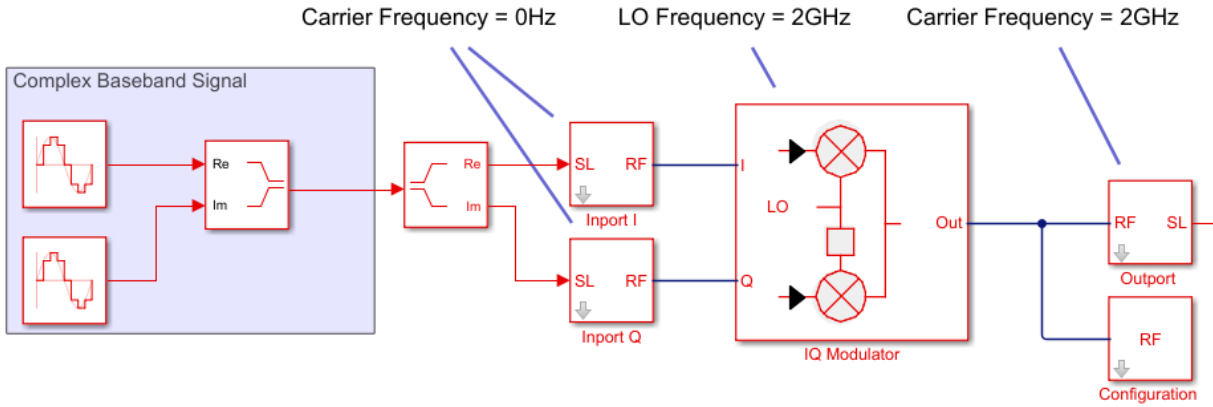
- Enter F_c in the **Carrier Frequencies** parameter.
- The simulation step size in the configuration block is the same as the sample time of the Simulink input signal, and it is not related to the carrier frequency.
- If the RF chain does not include any modulator or demodulator, use an Outport block at the end of the chain. You can use the `Outport` block to probe the complex equivalent signal centered on F_c.



No Carrier for Baseband Signal

Input signal is a digital communication complex input baseband signal (I,Q). You assume that no carrier is associated with the input signal. You want to upconvert the signal to F_c and model RF effects such as amplifier nonlinearity and S-parameter filters:

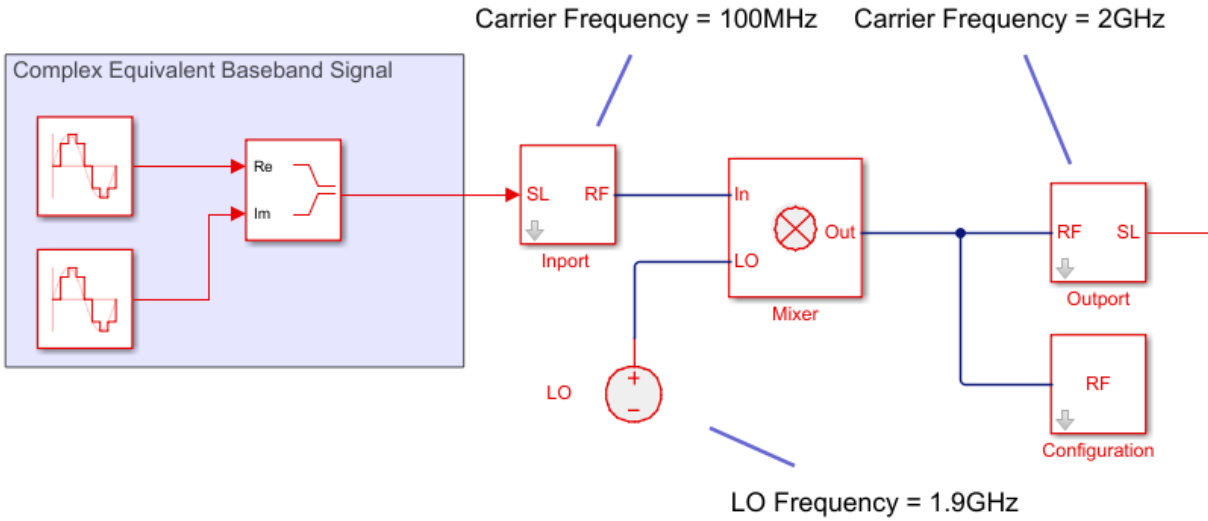
- Use to two Inport blocks for the I and Q components of the input signal. Set the **Carrier Frequencies** parameter of each Inport block to `0`
- To upconvert the signal, use an IQ Modulator block. Set the **Local oscillator frequency** to F_c.
- The simulation step size in the configuration block is the same as the sample time of the Simulink input signal, and it is not related to the Local Oscillator frequency.
- Use an Outport block at the end of the chain. and probe the signal at F_c.



Upconvert Signal to IF and Then RF

Input signal is a digital communication complex equivalent baseband signal (I,Q). You want to first upconvert the signal to intermediate frequency (IF), then to RF, and model RF imperfections:

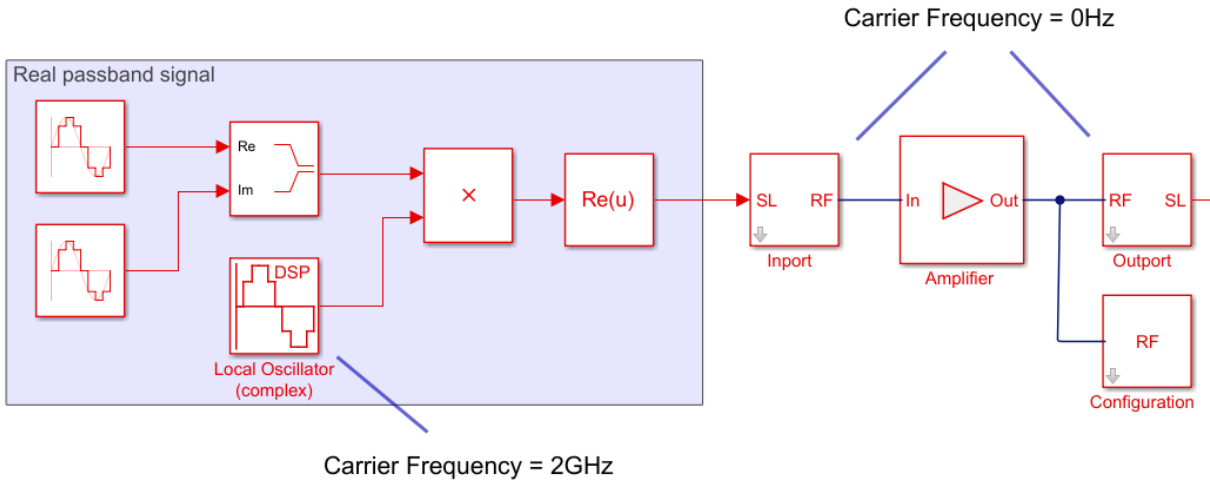
- Set the **Carrier Frequencies** parameter of each Inport block to `IF`.
- Use the Mixer block. Set the LO carrier frequency to `RF-IF`.
- Use an Outport block at the end of the chain. and probe the signal at RF.



Real Passband Signal

Input signal is a digital or analog real passband signal that is explicitly modulated to high frequency in Simulink domain:

- Set the **Carrier Frequencies** parameter of each Inport block to `0` and simulate RF effects.
- The simulation step size in the configuration block is the same as the sample time of the Simulink input signal, and it is proportional to the RF frequency.
- However there is no speed benefit in using RF Blockset for real-passband simulation. This option is not recommended



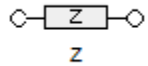
See Also

Configuration | Outport

Introduced in R2010b

Z (Impedance)

Model complex impedance



Library

Elements

Description

The Impedance block implements the relation, $v_k(t) = Z(f_k)*i_k(t)$, for each simulation frequency, f_k, where:

- $Z(f_k)$ represents complex-valued impedance at a specified simulation frequency.
- $v_k(t)$ represents the voltage across the terminals of the element at time t.
- $i_k(t)$ represents the current through the element at time t.

Circuit envelope current and voltage signals comprise in-phase, I_k, and quadrature, Q_k, components at each frequency, f_k.

Frequency-dependent impedance typically cannot be realized as a physical network such as an RLC chain. You can, however, use the Impedance block to model nonphysical behavior, such as frequency-independent negative capacitance or negative inductance. You can also use this block to specify resonant frequency offsets in filter networks.

Parameters

Impedance type

Choose `Frequency independent` to apply the same impedance for all frequencies. Choose `Frequency dependent` to apply the impedance as a piecewise linear function.

Complex impedance (ohm)

When **Impedance type** is set to `Frequency independent`, impedance is a scalar complex number that is applied to all simulation frequencies. When **Impedance type** is set to `Frequency dependent`, impedance is a vector of complex numbers, [Z_1, Z_2, Z_3, ...]. This vector is linearly interpolated for all simulation frequencies.

In both cases, for zero simulation frequency, the imaginary part of the impedance is ignored. Also, the real part is forced to be positive to produce a stable simulation.

Frequency

When **Impedance type** is set to `Frequency dependent`, specify a vector of nonnegative frequencies, [f_1, f_2, f_3, ...]. The pairs, f_i, Z_i, define a piecewise linear function, $Z(f)$, that is linearly interpolated for the simulation frequencies. For values outside the range, there is constant extrapolation. The default value of this parameter is `0` Hz.

Examples

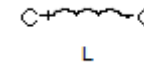
The example, “Frequency Response of an RF Transmit/Receive Duplex Filter”, simulates an analog RF filter comprising Capacitor, Inductor, Resistor, and Impedance blocks.

See Also

Capacitor | Inductor | Resistor

Inductor

Model inductor for circuit envelope analysis
Library: RF Blockset / Circuit Envelope / Elements



Description

The Inductor block models an inductor in circuit envelope environment.

Parameters

`Inductance` — Inductance value
`1e-9 H` (default) | real number

Inductance value, specified as a real number. Specify the units of the capacitance from the corresponding drop-down menu.

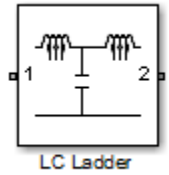
If you set this parameter to a value between `0` and `1e-18 H`, the block uses a value equal to `1e-18 H` during simulation. By default, the value is `1e-9 H`.

See Also

Capacitor | Resistor

LC Ladder

Model LC ladder networks



Library

Elements

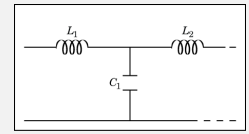
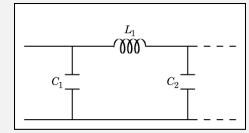
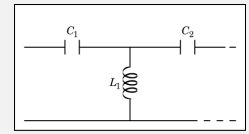
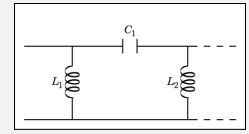
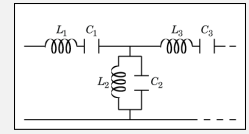
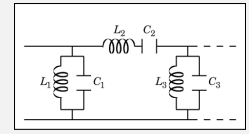
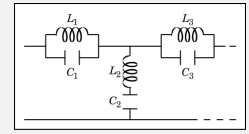
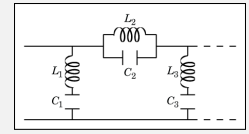
Description

The LC Ladder block models common two-port LC lossless networks.

Parameters

Specify ladder topology from the following options:

Ladder Topology

LC Lowpass Tee	
LC Lowpass Pi	
LC Highpass Tee	
LC Highpass Pi	
LC Bandpass Tee	
LC Bandpass Pi	
LC Bandstop Tee	
LC Bandstop Pi	

Inductance

Specify a vector of inductance values. The vector index of a value corresponds to the inductance index in the corresponding selected ladder topology. All values must be greater than zero.

Capacitance

Specify a vector of capacitance values. The vector index of a value corresponds to the capacitance index in the corresponding selected ladder topology. All values must be greater than zero.

The relationship between vector lengths of Capacitance and Inductance values must correspond to the Ladder topology selected.

Ground and hide negative terminals

Select this option to internally ground and hide the negative terminals. Clear this to expose the negative terminals. By exposing these terminals, you can connect them to other parts of your model.

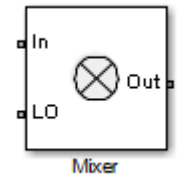
By default, this option is selected.

Examples

- The section, “Filter Mixing Products”, shows the use of a LC Ladder.

Mixer

Model mixer in RF systems
Library: RF Blockset / Circuit Envelope / Elements



Description

The Mixer block performs signal frequency translation and nonlinear amplification.

For a given RF input signal, $V_{RF} = A_{RF}\cos(\omega_{RF}t)$ and an LO input signal $V_{LO} = Ac_{LO}\cos(\omega_{LO}t)$, the mixer multiplies the signals at the input ports:

$$V_{in}V_{LO} = A_{in}\cos(\omega_{in}t)A_{LO}\cos(\omega_{LO}t)$$

$$= \frac{A_{in}A_{LO}}{2}\cos[(\omega_{in} + \omega_{LO})t] + \frac{A_{in}A_{LO}}{2}\cos[(\omega_{in} - \omega_{LO})t]$$

This mixing converts the frequency of RF signal to $\omega_{RF} + \omega_{LO}$ and $\omega_{RF} - \omega_{LO}$. For the mixer to perform this operation correctly, you must include the frequencies $\omega_{RF} + \omega_{LO}$ or $\omega_{RF} - \omega_{LO}$ in the simulation frequencies the Configuration block calculates.

The Power gain specification for this block relates the power of a single-sideband (SSB) to the input.

After mixing the RF and LO signals, the mixer block performs amplification. To model linear amplification, the mixer scales the signals by the coefficient a_1. A Voltage Controlled Voltage Source (VCVS), specified with a polynomial, implements nonlinear amplification. The polynomial includes saturation automatically and produces additional intermodulation frequencies.

Parameters

Main

`Source of conversion gain` — Source parameter of conversion gain

`Available power gain` (default) | `Open circuit voltage gain` | `Polynomial coefficients`

Source parameter of conversion gain, specified as one of the following:

- `Available power gain` — The block uses the value of the Available power gain parameter to calculate the linear voltage gain term of the polynomial VCVS, a_1. This calculation assumes a matched load termination for the mixer.
- `Open circuit voltage gain` — The block uses the value of the Open circuit voltage gain parameter as the linear voltage gain term of the polynomial VCVS, a_1.
- `Polynomial coefficients` — The block implements a nonlinear voltage gain according to the polynomial you specify. The order of the polynomial must be less than or equal to 9 and the coefficients are ordered in ascending powers. If a vector a has 10 coefficients, [a_0, a_1, a_2, ..., a_9], the polynomial it represents is $V_{out} = a_0 + a_1 V_{in} + a_2 V_{in}^2 + \cdots + a_9 V_{in}^9$. In this case, a_1 represents the linear gain term, and the modeling of higher-order terms is done according to [1].

  For example, the vector [a_0, a_1, a_2, a_3] specifies the relation $V_{out} = a_0 + a_1 V_{in} + a_2 V_{in}^2 + \cdots + a_3 V_{in}^3$.

  Trailing zeroes are omitted: if $a_3 = 0$, [a_0, a_1, a_2] defines the same polynomial as [a_0, a_1, a_2, 0]. The default value of this parameter is [0 1], corresponding to the linear relation $V_o = V_i$.

`Available power gain` — Linear gain of mixer

0 dB (default) | scalar

Linear gain of mixer, specified as a scalar in dB. Specify the units from the corresponding drop-down list.

Dependencies

To enable this parameter, select `Available power gain` in **Source of conversion gain** tab.

`Open circuit voltage gain` — Open circuit voltage gain

0 dB (default) | scalar

Open circuit voltage of mixer, specified as a scalar in dB. Specify the units from the corresponding drop-down list.

Dependencies

To enable this parameter, select `Open circuit voltage gain` in **Source of conversion gain** tab.

`Polynomial coefficients` — Order of polynomial

`[0 1]` (default) | vector

Order of polynomial, specified as a vector.

The order of the polynomial must be less than or equal to 9. The coefficients are ordered in ascending powers. If a vector has 10 coefficients, `[`a_0`,`a_1`,`a_2`, ... `a_9`]`, the polynomial it represents is:

$V_{out} = a_0 + a_1V_{in} + a_2{V_{in}}^2 + \ldots + a_9{V_{in}}^9$
where a_1 represents the linear gain term, and higher-order terms are modeled according to [1].

For example, the vector `[`a_0`,`a_1`,`a_2`,`a_{32}`]` specifies the relation $V_o = a_0 + a_1V_1 + a_2{V_1}^2 + a_3{V_1}^3$. Trailing zeroes are omitted. If $a_3 = 0$, then `[`a_0`,`a_1`,`a_2`]` defines the same polynomial as [a_0,a_1,a_2,0]. The default value of this parameter is [0,1], corresponding to the linear relation $V_o = V_i$.

Dependencies

To enable this parameter, select `Polynomial coefficients` in **Source of conversion gain** tab.

`Input impedance (Ohm)` — Input impedance of mixer

`50` (default) | scalar

Input impedance of mixer, specified as a scalar.

`Output impedance (Ohm)` — Output impedance of mixer

`50` (default) | scalar

Output impedance of mixer, specified as a scalar.

`LO impedance (Ohm)` — Impedance at LO port of mixer

`inf` (default) | scalar

Output impedance of mixer, specified as a scalar.

`Noise figure (dB)` — Single-sideband IEEE noise figure of mixer
0 dB (default) | scalar

Single-sideband noise figure of mixer, specified as a scalar according to the IEEE definition.

To model noise in circuit envelope model with a Noise, Amplifier, or Mixer block, you must select the **Simulate noise** check box in the Configuration block dialog box.

The IEEE SSB definition assumes that the noise in the image bandwidth at the input of the mixer is perfectly rejected, while the mixer internally generates noise in both the image bandwidth and the signal bandwidth. As a result, the noise at the output of the mixer is the sum of two contributions:

$$N_{\text{out}} = N_{\text{in}}G_{\text{mix}} + 2N_{\text{mixer}}G_{\text{mix}},$$

where:

- N_{out} is the noise at the output of the mixer.
- N_{in} is the noise at the input of the mixer (assuming that the noise in the image bandwidth is perfectly rejected).
- N_{mixer} is the noise internally generated by the mixer in both the signal and the image bandwidths.
- G_{mix} is the mixer gain.

As a result, the noise factor according to the IEEE SSB definition can be expressed as

$$F_{\text{SSB-IEEE}} = 1 + 2N_{\text{mixer}}/N_{\text{in}},$$

which is related to other commonly used definitions through

$$F_{\text{SSB}} = 2 + 2N_{\text{mixer}}/N_{\text{in}} = 1 + F_{\text{SSB-IEEE}},$$

$$F_{\text{DSB}} = 1 + N_{\text{mixer}}/N_{\text{in}} = \frac{1}{2}(1 + F_{\text{SSB-IEEE}}).$$

You can apply the IEEE SSB definition directly to describe mixer stages when using the Friis formulas for link budget analysis. Using the other definitions requires changing the Friis formulas. Both the Mixer block in RF Blockset and the **RF Budget Analyzer** app in RF Toolbox™ use the IEEE definition.

The analytic relationships between the three definitions allow you to simulate the noise level at the output of the mixer. For example,

$$F_{\mathrm{SSB}} = 4\ \mathrm{dB} \Rightarrow F_{\mathrm{SSB\text{-}IEEE}} = 10\log_{10}(10^{F_{\mathrm{SSB}}/10} - 1) = 1.79\ \mathrm{dB},$$

$$F_{\mathrm{DSB}} = 4\ \mathrm{dB} \Rightarrow F_{\mathrm{SSB\text{-}IEEE}} = 10\log_{10}(2 \times 10^{F_{\mathrm{DSB}}/10} - 1) = 6.04\ \mathrm{dB}.$$

If you simulate an RF Blockset mixer without including an ideal image rejection filter, then the noise at the output of the mixer is larger than that predicted by the noise figure, because the noise in the image bandwidth is effectively folded onto the output signal.

For this reason, when generating models, both the Modulator and Demodulator blocks insert an ideal image-rejection filter automatically. (You can remove the filtering within the block mask.)

The Noise Figure Testbench block measures the SSB noise figure and enables you to verify that the simulated noise figure has the expected value.

- If you add an ideal image rejection filter to your model, then the effective noise figure is consistent with the analytic value.
- If you remove the image-rejection filter, or if you use a filter with partial rejection, then the measured noise figure is larger than the analytic value.

`Ground and hide negative terminals` — Ground RF circuit terminals
`on` (default) | `off`

Select this parameter to internally ground and hide the negative terminals. To expose the negative terminals, clear this parameter. By exposing these terminals, you can connect them to other parts of your model.

By default, this option is selected.

Nonlinearity

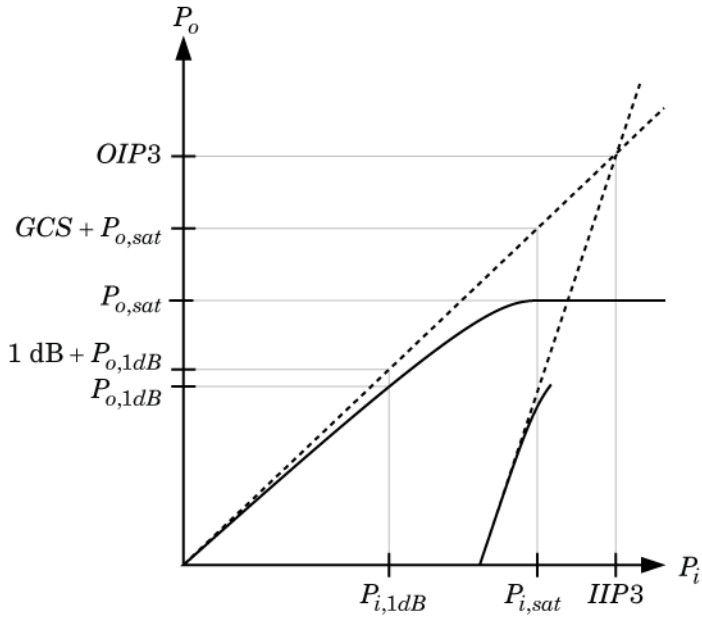
`Nonlinear polynomial type` — Polynomial nonlinearity
`Even and odd order` (default) | `Odd order`

Polynomial nonlinearity, specified as one of the following:

- `Even and odd order`: When you select `Even and odd order`, the amplifier can produce second- and third-order intermodulation frequencies in addition to a linear term.

- `Odd order`: When you select `Odd order`, the amplifier generates only odd order intermodulation frequencies.

  The linear gain determines the linear a_1 term. The block calculates the remaining terms from the specified parameters. These parameters are **IP3**, **1-dB gain compression power**, **Output saturation power**, and **Gain compression at saturation**. The number of constraints you specify determines the order of the model. The figure shows the graphical definition of the nonlinear amplifier parameters.



`Intercept points convention` — Intercept points convention
`Output` (default) | `Input`

Intercept points convention, specified as `Input`-referred or `Output`-referred. Use this specification for the intercept points, 1-dB gain compression power, and saturation power.

`IP2` — Second-order intercept point
`infdBm` (default) | scalar

Second-order intercept point. specified as a scalar. The default value `inf dBm` corresponds to an unspecified point.

Dependencies

To enable this parameter, select `Even and odd order` in **Nonlinear polynomial type** tab.

`IP3` — Third-order intercept point

`inf dBm` (default) | scalar

Third-order intercept point, specified as a scalar. The default value `inf dBm` corresponds to an unspecified point.

`1-dB gain compression power` — 1-dB gain compression power

`infdBm` (default) | scalar

1-dB gain compression power, specified as a scalar. The 1-dB gain compression point must be less than the output saturation power.

Dependencies

To enable this parameter, select `Odd order` in **Nonlinear polynomial type** tab.

`Output saturation power` — Output saturation power

`infdBm` (default) | scalar

Output saturation power, specified as a scalar. The block uses this value to calculate the voltage saturation point used in the nonlinear model. In this case, the first derivative of the polynomial is zero, and the second derivative is negative.

Dependencies

To enable this parameter, select `Odd order` in **Nonlinear polynomial type** tab.

`Gain compression at saturation` — Gain compression at saturation

`infdBm` (default) | scalar

Gain compression at saturation, specified as a scalar.

Dependencies

To enable this parameter, select `Odd order` in **Nonlinear polynomial type** tab and set **Output saturation power** .

References

[1] Grob, Siegfried, and Jürgen Lindner. “Polynomial Model Derivation of Nonlinear Amplifiers.” *Department of Information Technology*, University of Ulm, Germany.

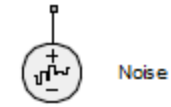
See Also

Amplifier | S-Parameters

Introduced in R2010b

Noise

Model noise using current or voltage noise source in RF systems
Library: RF Blockset / Circuit Envelope / Sources



Description

Use the Noise block to model noise as a ideal current or voltage source for blocks. When you use a Noise to simulate noise in an RF model, also select the **Simulate noise** check box in the Configuration block. Otherwise, the model simulates without noise. The Noise block does not depend on the Temperature parameter in the Configuration.

Parameters

`Source type` — Noise type
`Ideal voltage` (default) | `Ideal current`

Noise type, specified as `Ideal voltage` or `Ideal current`.

`Noise distribution` — Noise distribution type
`White` (default) | `Piece-wise linear` | `Colored`

Noise distribution type, specified as `White`, `Piece-wise linear`, or `Colored`.

`Noise power spectral density` — Single-sided noise power spectral distribution (PSD)
`0` (default)

Single-sided noise power spectral distribution (PSD), specified as:

- `White`, spectral density is a single non-negative value. The power value of the noise depends on the bandwidth of the carrier and the bandwidth depends on the time step. This is an uncorrelated noise source.

- `Piece-wise linear`, spectral density is a vector of values [p_i]. For each carrier, the noise source behaves like a white uncorrelated noise. The power of the noise source is carrier-dependent.
- `Colored`, depends on both carrier and bandwidth. This is a correlated noise source.

When **Source type** is set to:

- `Ideal voltage`, spectral density units are V^2/Hz.
- `Ideal Current`, spectral density units are A^2/Hz.

`Frequencies` — Frequencies for piece-wise linear noise distribution
`0 Hz` (default) | vector of non-negative frequencies | `Hz` | `kHz` | `MHz` | `GHz`

Frequencies for piece-wise linear noise distribution, specified as vector of non-negative frequencies.

`Ground and hide negative terminals` — Ground RF circuit terminals
`on` (default) | `off`

Select this option to internally ground and hide the negative terminals. To expose the negative terminals, clear the option. By exposing these terminals, you can connect them to other parts of your model.

By default, this option is selected.

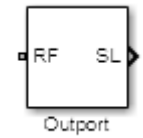
See Also

Continuous Wave | Sinusoid

Introduced in R2010b

Outport

Convert RF Blockset signal to Simulink output signals
Library: RF Blockset / Circuit Envelope / Utilities



Description

The Outport block outputs carrier modulation signals in the RF Blockset circuit envelope simulation environment as Simulink signal. For an introduction to RF simulation, see the example, “Simulate High Frequency Components”.

The output port senses current and voltage complex envelope or real passband signals. Complex baseband signals consist of in-phase (I_k) and quadrature (Q_k) components centered around each specified center frequency f_k.

The **Sensor type** parameter determines which signal the block measures, and the **Output** parameter defines the format of the Simulink signal.

Note Outport block `Real passband` output does not support frame-based processing (supported by the Configuration blocks). This block errors if the **Samples per frame** is more than 1 in the configuration block.

Parameters

`Sensor type` — Type of signal measured by sensor

`Ideal voltage` (default) | `Ideal current` | `Power`

Type of signal measured by sensor, specified as:

- `Ideal voltage` — The block outputs the modulations of the voltage signal at the specified **Carrier frequencies** in the format specified by the **Output** parameter. This

is the recommended option to sense a signal without adding a loading impedance and changing matching conditions.

- `Ideal current` — The block outputs the modulations of the current signal at the specified **Carrier frequencies** in the format specified by the **Output** parameter.
- `Power` — The block outputs the modulations of the voltage signal at the specified **Carrier frequencies** and scaled with respect to the specified load impedance. This is the recommended option to sense a signal generated in RF Blockset 50 Ohm environment or a different reference impedance. When you use the power option , the output port automatically inserts a load impedance in your circuit.

  $$\frac{\sqrt{\mathrm{Re}(Z_l)}}{Z_l}v(t)$$

  where Z_l is the value of the **Load impedance (ohms)** parameter.

`Load impedance (Ohm)` — Load impedance of RF circuit

`inf` (default) | vector of positive integers in ohms

Load impedance of RF circuit used to measure signal power, specified as a vector of positive integers in ohms. When you use the power option , the output port automatically inserts a load impedance in your circuit. When you use multiple Outport blocks as power sources at the same node in a given circuit, the resulting load is the parallel combination of the specified load impedances.

Dependencies

To enable this parameter, select `Power` in **Sensor type**.

`Output` — Format of output signals

`Complex Baseband` (default) | `In-phase and Quadrature Baseband` | `Magnitude and Angle Baseband` | `Real Passband`

Format of output signals, specified as one of the following:

- `Complex Baseband` — The block outputs a vector of complex-valued signals $I_k(t) + j \cdot Q_k(t)$ at the port labeled **SL**. The kth element of the vector is the kth frequency specified by the **Carrier frequencies** parameter.
- `In-phase and Quadrature Baseband` — The block outputs two vectors of real-valued signals $I_k(t)$ and $Q_k(t)$ at the **I** port and **Q** port, respectively. The signal at the **I** port contains the in-phase components, and the signal at the **Q** port contains the quadrature components. The kth element of the vector is the kth frequency specified

by the **Carrier frequencies** parameter. The quadrature component of a signal with carrier frequency equal to 0 Hz is zero.

- `Magnitude and Angle Baseband` — The block outputs two real-valued vectors, whose elements are the magnitude and phase angle of the modulation. The **Mag** port outputs $|I_k(t) + j \cdot Q_k(t)|$ and the **Ang** port outputs $\text{Arg}[I_k(t) + j \cdot Q_k(t)]$. The kth element of the vector is the kth frequency specified by the **Carrier frequencies** parameter.
- `Real Passband` — The block outputs real passband signals by combining envelope and carrier signals for all frequencies listed under **Carrier frequencies**. When using the `Real Passband` option, the solver takes time steps small enough to resolve the carrier. Thus, simulation speed improvements from envelope simulation may be limited.

$$out = \text{Re}\,al(\sum_{k=1}^{N}(I_k(t) + j \bullet Q_k(t)) \bullet e^{j2\pi f_k t})$$

where t is the value of **Load impedance (ohms)** parameter.

`Automatically compute output step size` — Determine optimal time step to resolve highest listed carrier frequency
`on` (default) | `off`

Select this parameter to allow RF Blockset to determine the optimal time step to resolve the highest listed carrier frequency. Clear the parameter selection to enter a value for step size.

`Step size` — Time step
`1e-6 s` (default) | positive integer in seconds

Time step, specified as a positive integer in seconds. The step size should be small enough to resolve the fastest carrier signal. The size helps to avoid passband output undersampling and aliasing effects.

Set the time step value to `-1` to inherit the time step specified from `Step size` in Configuration block.

`Carrier frequencies` — Carrier frequencies
`0 Hz` (default) | vector of positive integers in Hz

Carrier frequencies, specified as a vector of positive integers in Hz. In carrier frequencies, the elements are a combination of fundamental tones and corresponding harmonics in the Configuration block.

`Ground and hide negative terminals` — Ground RF circuit terminals
`on` (default) | `off`

Select this parameter to ground and hide the negative terminals. Clear the parameter to expose the negative terminals. By exposing these terminals, you can connect them to other parts of your model.

By default, this option is selected.

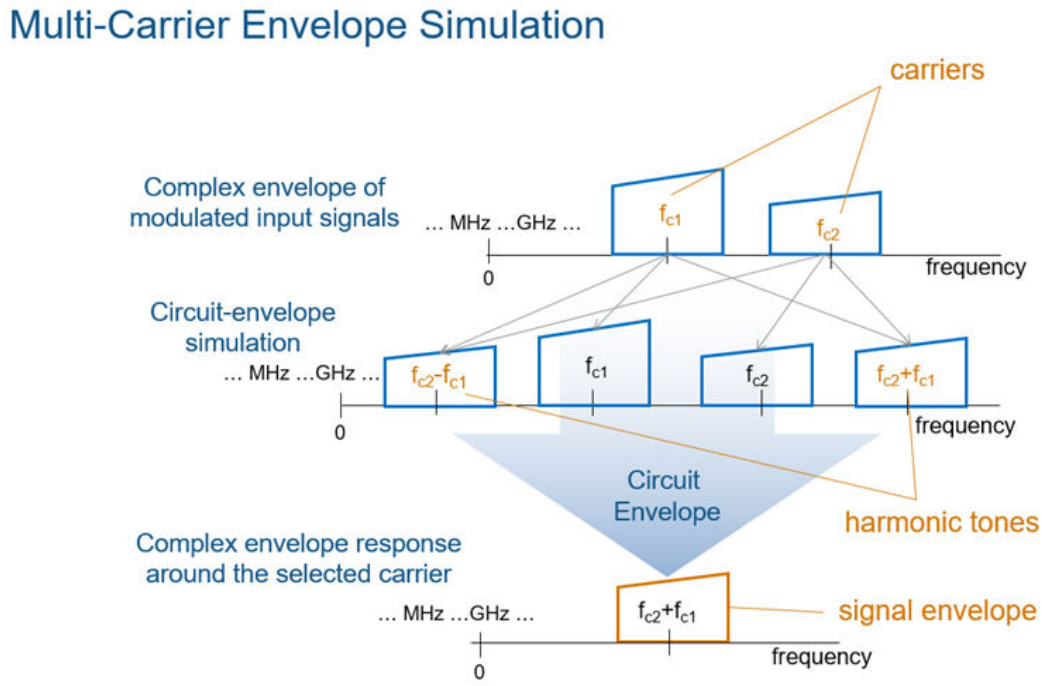
More About

Multi-Carrier Envelope Simulation

Using the Inport block you can specify the complex envelopes of your input signals and import them as RF signals for multi-carrier simulation.

The Configuration block automatically determines the fundamental tones specified in the input ports and proposes a suitable harmonic order to capture the non-linearity of the system. You can also manually specify the harmonic order for each fundamental tone in the simulation.

In the input port, you can specify as many carrier frequencies as you want. It is recommended that you trade off the simulation bandwidth (inversely proportional to the simulation time step) and the total number of simulation frequencies.



Real Passband Formula

Normalized carrier power option in the Configuration block defines the passband formula:

- When this option is selected, RF Blockset interprets complex envelope I+jQ signal for the k^{th} carrier as,

  $$s_k(t) = I(t)\sqrt{2}\cos(2\pi f_k t) - Q(t)\sqrt{2}\sin(2\pi f_k t)$$

- When this option is not selected, the signal on the k^{th}

  $$s_k(t) = I(t)\cos(2\pi f_k t) - Q(t)\sin(2\pi f_k t)$$

- In both cases, the signal for zero-frequency (DC) carrier is $x(t) = I(t)$. The final output signal is computed as $s(t) = \text{sum}(s_k)$

Formula for Time Step

The formula for the time step selected is:

$\min(h, ((1/2\pi f) \div 8))$

- f is the largest listed carrier frequency.
- h is the time step listed in Configuration block.

Algorithms

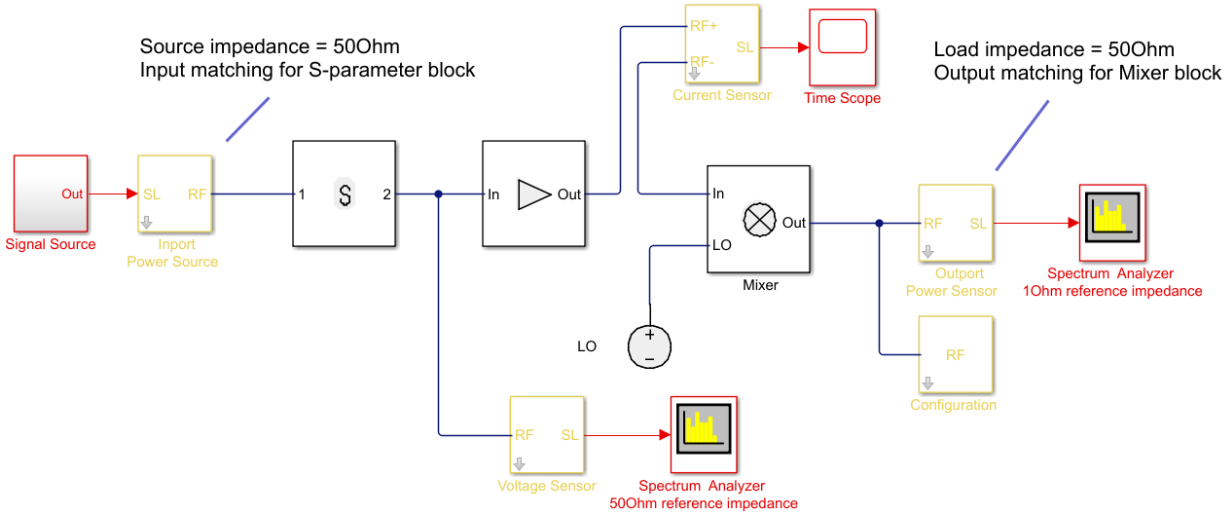
How to Use Outport Block

Sensing Signals at Different Nodes of RF Chain

Consider an RF chain composed by several stages. You want to sense and inspect the signal behavior at different intermediate nodes.

Use the power input port and power sensor port at the input and output of your chain, as gateways between Simulink (reference 1Ohm) and the RF domain.

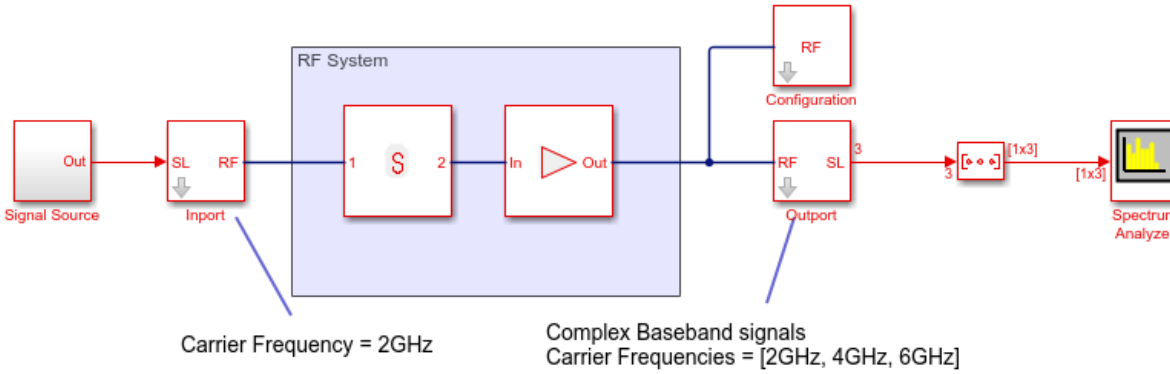
Use the voltage or current output ports connected to the intermediate nodes and branches that you want to inspect. To measure the power of a voltage signal, change the default spectrum analyzer reference impedance to 50Ohm.



Sense Complex Equivalent Baseband Signal for Digital Signal Processing

Output signal is a combination of (digital communication) complex equivalent baseband signals (I,Q). For each envelope, assume an implicit center frequency for the signal that is equal to the carrier frequency, F_c.

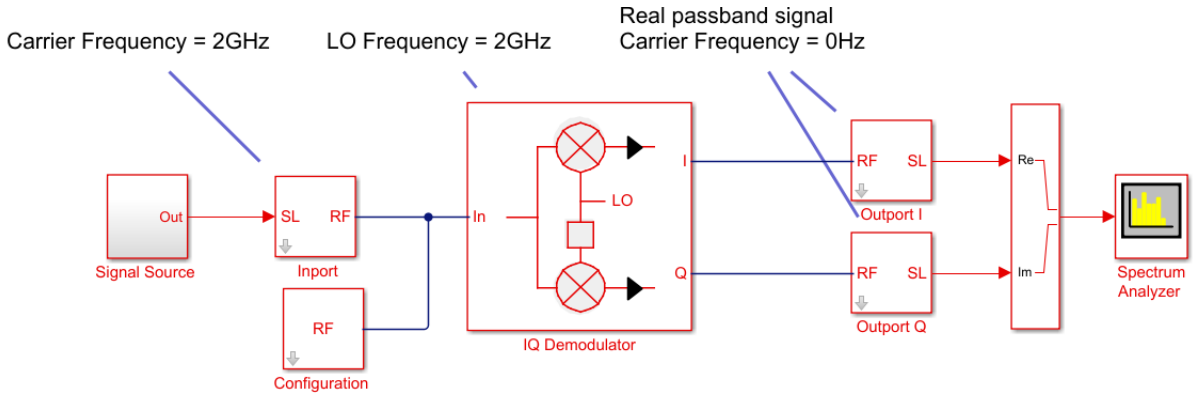
- Enter the array F_c in the **Carrier Frequencies** parameter corresponding to the center frequency of the envelopes that you want to sense. The output signals is an array of complex (I,Q) envelopes.
- The output port of sensor type power provides a termination (by default 50Ohm) at the end of the chain.
- The simulation step size in the configurationConfiguration block is the same as the sample time of the Simulink signal, and it is not related to the carrier frequency.



Sense Real Passband Signal

Output signal is a (digital communication) complex baseband signal (I,Q) resulted from a direct conversion receiver. Assume that no carrier is associated with the output signal.

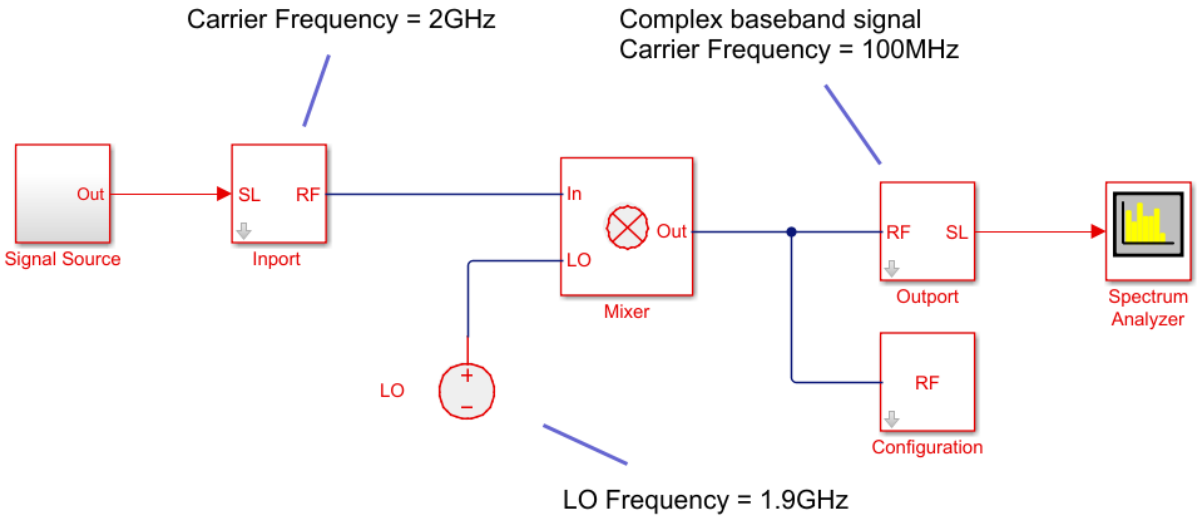
- Use two Outport blocks for the I and Q components of the signal. Set the **Carrier Frequencies** parameter of each Outport block to `0`. Use the real passband option to sense a real signal rather than a complex signal with quadrature component equal to `0`.
- The output ports of sensor type power provide a termination (by default 50Ohm) at the end of the chain.
- To down-convert the signal, use an IQ Demodulator block. Set the **Local oscillator frequency** to F_c equal to the center frequency of the input signal (direct conversion).
- The simulation step size in the configuration block is the same as the sample time of the Simulink signal, and it is not related to the Local Oscillator frequency.



Down Convert Signal to IF and Sense Complex Equivalent Baseband Signal

Output signal is a (digital communication) complex equivalent baseband signal (I,Q) down-converted to intermediate frequency (IF). Apply digital signal processing techniques for processing the complex equivalent baseband information of the signal.

- Set the **Carrier frequencies** parameter of the Outport block to `IF`. Use the complex baseband option.
- Use the Mixer block for down-conversion. Set the LO carrier frequency to `LO = RF-IF`. The output port will behave as an ideal filter and select only the down-converted signal. The envelope at `RF+LO = 2RF-IF` frequency is simulated but it is not sensed by the output port.
- The simulation step size in the configuration block is the same as the sample time of the Simulink input signal, and it is not related to the IF, RF or LO frequencies.



Down Convert Signal to IF and Sense Real Passband Signal

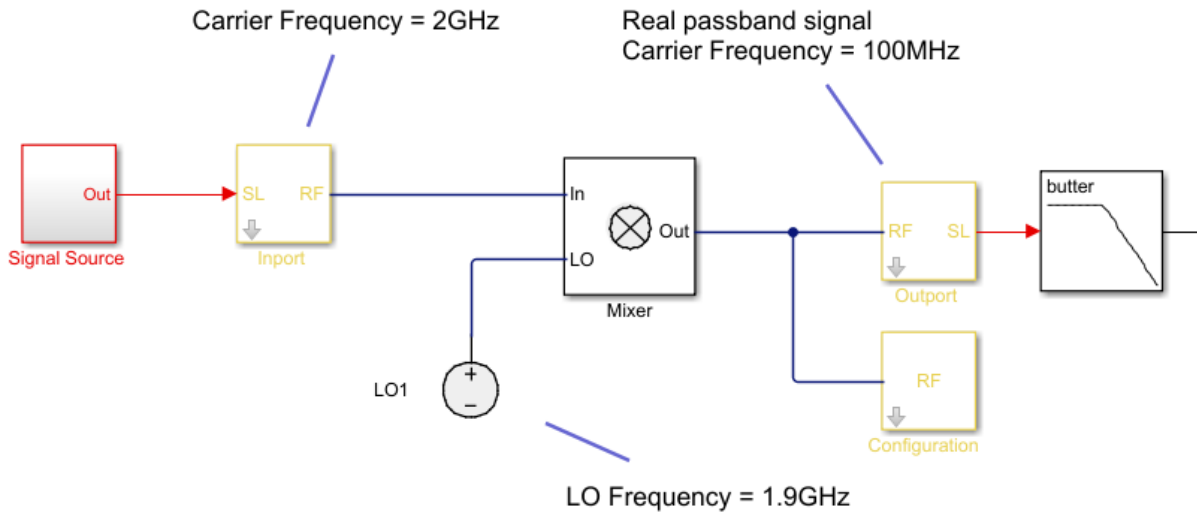
Output signal is a (digital communication) signal (I,Q) down-converted to intermediate frequency (IF). Apply analog signal processing techniques to the output signal.

- Set the **Carrier frequencies** parameter of the Output block to `IF`. Use the real passband option.

  If the intermediate frequency is within the simulation bandwidth defined in the configuration block, Use the same step size in the output without the need for resampling the signal.

  If the intermediate frequency is not within the simulation bandwidth defined in the configuration block, you need to resample the signal (as described in the real-passband formula) to correctly resample the carrier.
- Use the Mixer block. Set the LO carrier frequency to `LO = RF-IF`. The output port will behave as an ideal filter, and select only the down-converted signal. The envelope at `RF+LO = 2RF-IF` frequency is simulated but it is not sensed by the output port.
- The simulation step size in the configuration block is the same as the sample time of the Simulink input signal. All the blocks in the RF Blockset network connected to the configuration block are executed with the same step size. The time step of the outport,

when the real passband option is selected, might differ from the time step of the input Simulink signal and of the time step set in the configuration block.



See Also

Configuration | Inport

Introduced in R2010b

Phase Shift

Model phase shift in RF systems

Library

Elements

Description

The Phase Shift block models an ideal phase shift in the circuit envelope environment.

Parameters

Phase-shift

The default value of this parameter is `90 deg`. The phase shift is applied to all nonzero simulation frequencies. For zero (DC) frequency, the shift is always zero.

Ground and hide negative terminals

Select this option to internally ground and hide the negative terminals. Clear this to expose the negative terminals. By exposing these terminals, you can connect them to other parts of your model.

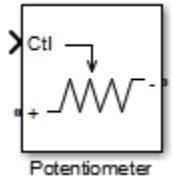
By default, this option is selected.

Examples

The example, “Measuring Image Rejection Ratio in Receivers”, shows how to use Phase Shift blocks to model an LO phase offset in a receiver architecture.

Potentiometer

Simulink controlled potentiometer



Library

Junctions

Description

The Potentiometer block models a variable resistor. The Potentiometer block uses Simulink signal as input control signal to vary the resistance between the positive and the negative terminals.

The equations for the potentiometer voltage–current relationships when the control signal varies between zero and one are:

Linear: $V_{pot} = I_{pot} . (R_{off} . Ctl + R_{on} . (1 - Ctl))$

Logarithmic: $V_{pot} = I_{pot} . (C_1 . \exp(C_2 . Ctl) + C_3)$

Antilog: $V_{pot} = I_{pot} . (C_1 . \log(1 + C_2 . Ctl) + C_3 . Ctl + R_{on})$

- Linear, Logarithmic, Antilog — Electrical characteristic of the potentiometer.
- `Ctl` — Simulink signal
- R_{on} — Minimum resistance of the potentiometer
- R_{off} — Maximum resistance of the potentiometer

Parameters

Minimum resistance

Minimum input resistance of the potentiometer, specified as a positive scalar. The default value is `10` ohms.

Maximum resistance

Maximum input resistance of the potentiometer, specified as a positive scalar. The default value is `50` ohms. This value must be greater than the minimum resistance.

Potentiometer type

Curve type of the potentiometer: `Linear`, `Logarithmic`, or `Antilog`. The default is `Linear`.

Percentage of resistance at half wiper

Percentage of resistance at half wiper for logarithmic and antilog potentiometer types, specified as a positive scalar. The valid values depend on the potentiometer type:

- `Logarithmic` — Greater than 0 ohms and less than 50 ohms. The default is 20.
- `Antilog` — Greater than 50 ohms and less than 100 ohms. The default is 80.

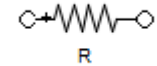
See Also

SPDT | SPST | Switch

Introduced in R2015b

Resistor

Model resistor for circuit envelope analysis
Library: RF Blockset / Circuit Envelope / Elements



Description

The Resistor block models a resistor within the RF Blockset circuit envelope simulation environment. For an introduction to RF simulation, see the example, “Simulate High Frequency Components”.

The block implements the relation

$$v(t) = Ri(t)$$

where:

- R represents the resistance, as a function of temperature.
- $i(t)$ represents the current through the capacitor at time t.
- $v(t)$ represents the voltage across the terminals of the capacitor at time t.

RF Blockset current and voltage signals consist of in-phase (I_k) and quadrature (Q_k) components at each frequency f_k specified in the Configuration block:

$$i(t) = \sum_{\{f_k\}} \left(i_{I_k}(t) + j \cdot i_{Q_k}(t)\right) e^{j(2\pi f_k)t}$$

$$v(t) = \sum_{\{f_k\}} \left(v_{I_k}(t) + j \cdot v_{Q_k}(t)\right) e^{j(2\pi f_k)t}$$

Parameters

`Resistance` — Resistance value

`50 Ohm` (default) | real number in ohms

Resistance value , specified as a real number in ohms greater than zero. Specify the units of the resistance from the corresponding drop-down menu.

`Simulate noise` — Simulate thermal noise
`on` (default) | `off`

Select this parameter, to simulate thermal noise in a resistor. Then, in the Configuration block dialog box, also select the **Simulate noise** check box. By default, both **Simulate noise** check boxes are selected.

This parameter inserts a current noise source with the single-sided power density of 4 kT/R A^2/Hz, where:

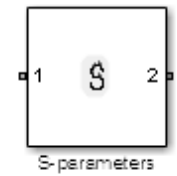
- k is the Boltzmann constant
- T is the value of the **Temperature** parameter, in degrees Kelvin. (Also located in the Configuration block.)

See Also

Capacitor | Inductor

S-Parameters

Model S-parameter network
Library: RF Blockset / Circuit Envelope / Elements



Description

The S-parameters block models a network defined by S-parameters in the RF Blockset circuit envelope simulation environment. The device can have up to eight ports. For an introduction to RF simulation, see the example, “Simulate High Frequency Components”.

The block models S-parameter data in the RF Blockset environment by fitting a rational function to the specified data. For more information about rational fitting of S-parameters, see the RF Toolbox `rationalfit` function.

Parameters

Main

`Data source` — Data source
`Data file` (default) | `Network-parameters` | `Rational model`

Data source for S-parameters behavior, specified as one of the following:

- `Data file` — Name of a Touchstone file with the extension `.s2p`. The block ignores noise and nonlinearity data in imported files.
- `Network-parameters` — Provide **Network parameter** data such as `S-parameters`, `Y-parameters`, and `Z-parameters` with corresponding **Frequency** and **Reference impedance (ohms)** for the s-parameters.
- `Rational model` — Provide values for **Residues**, **Poles**, and **Direct feedthrough** parameters which correspond to the equation for a rational model

$$F(s) = \left(\sum_{k=1}^{n} \frac{C_k}{s - A_k} + D \right),\ s = j2\pi f$$

.In this rational model equation, each C_k is the residue of the pole A_k. If C_k is complex, a corresponding complex conjugate pole and residue must also be enumerated. This object has the properties `C`, `A`, and `D`. You can use these properties to specify the **Residues**, **Poles**, and **Direct feedthrough** parameters.

`Data file` — Name of network parameter data file

`simrfV2_unitygain.s2p` (default) | character vector

Name of network parameter data file, specified as a character vector.

Dependencies

To enable this parameter, select `Data file` in **Data source** tab.

`Network parameter type` — Network parameter type

`S-parameters` (default) | `Y-parameters` | `Z-parameters`

Network parameter type, specified as `S-parameters`, `Y-parameters`, or `Z-parameters`.

Dependencies

To enable this parameter, select `Network-parameters` in **Data source** tab.

`Network-parameters` — Network parameter values

`[0 0;1 0]` (default) | multidimensional array

Network parameter values specified as a multidimensional array. The third dimension of the S-parameter array must be the same length as the vector of frequencies specified by the **Frequency** parameter. The default values are different for S-parameters, Y-parameters, and Z-parameters respectively.

Dependencies

To enable this parameter, select `Network-parameters` in **Data source** tab.

`Frequency (dB)` — Frequency of network parameters

`1e9 Hz` (default) | scalar | vector | `Hz` | `kHz` | `MHz` | `GHz`

Frequency of network parameters, specified as a scalar or a vector in Hz.

Dependencies

To enable this parameter, select `Network-parameters` in **Data source**.

`Reference Impedance(Ohm)` — Reference impedance of network parameters

`50` (default) | scalar

Reference impedance of network parameters, specified as a scalar.

Dependencies

To enable this parameter, select `Network-parameters` in **Data source** tab.

`Residues` — Residues in order of rational model

`0` (default) | vector

Residues in order of rational model, specified as a vector.

Dependencies

To enable this parameter, select `Rational model` in **Data source** tab.

`Poles` — Residues in order of rational model

`0` (default) | vector

Poles in order of rational model, specified as a vector.

Dependencies

To enable this parameter, select `Rational model` in **Data source** tab.

`Direct feedthrough` — Direct feedthrough

`{0 0:1 0}` (default) | array of vectors

Direct feedthrough, specified as an array vector.

Dependencies

To enable this parameter, select `Rational model` in **Data source** tab. .

`Simulate noise` — Generate thermal noise waves

`off` (default) | `on`

Choose this parameter to generate thermal noise waves [1]. Clear this parameter to stop simulating noise. For more information see, “Generate Thermal Noise” on page 1-153.

`Ground and hide negative terminals` — Ground RF circuit terminals

`on` (default) | `off`

Select this parameter to ground and hide the negative terminals. Clear this parameter to expose the negative terminals. By exposing these terminals, you can connect them to other parts of your model.

By default, this option is selected.

Modeling

`Modeling options` — Model S-parameters

`Time-domain (rationalfit)` (default) | `Frequency-domain`

Model S-parameters, specified as:

- Time-domain (rationalfit) technique creates an analytical rational model that approximates the whole range of the data. When modeling using `Time domain`, the **Plot** in `Visualization` tab plots the data defined in `Data Source` and the values in the `rationalfit` function.
- Frequency-domain computes the baseband impulse response for each carrier frequency independently. This technique is based on convolution. There is an option to specify the duration of the impulse response. For more information, see “Compare Time and Frequency Domain Simulation Options for S-parameters”.
- For the Amplifier and S-parameters blocks, the default value is `Time domain (rationalfit)`. For the Transmission Line block, the default value is `Frequency domain`.

Dependencies

To set this parameter, first select `Data file` or `Network-parameters` in **Data source**. This selection activates the **Visualization** Tab which contains **Source of frequency data**

`Fitting options` — Rationalfit fitting options

`Fit individually` (default) | `Share poles by column` | `Share all poles`

Rationalfit fitting options, specified as `Fit individually`, `Share poles by column`, or `Share all poles`.

Rational fitting results shows values of **Number of independent fits**, **Number of required poles**, and **Relative error achieved (dB)**.

Dependencies

To set this parameter, select `Time domain (rationalfit)` in **Modeling options**.

`Relative error desired (dB)` — Relative error acceptable for the rational fit
`-40` (default) | scalar

Relative error acceptable for the rational fit, specified as a scalar.

Dependencies

To set this parameter, select `Time domain (rationalfit)` in **Modeling options**.

`Automatically estimate impulse response duration` — Automatically calculate impulse response
`on` | `off`

Select this parameter to automatically calculate impulse response. Clear this parameter to manually specify the impulse response duration using **Impulse response duration**.

Dependencies

To set this parameter, select `Frequency domain` in **Modeling options**.

`Impulse response duration` — Impulse response duration
`1e-10` (default) | scalar

Impulse response duration, specified as a scalar.

Dependencies

To set this parameter, first select `Frequency domain` in **Modeling options**. Then, clear `Automatically estimate impulse response duration`.

`Use only S-parameter magnitude with appropriate delay` — Use only S-parameter magnitude with appropriate delay
`off` (default) | `on`

Select this parameter to ignore the s-parameter phase and delay the impulse response by half its length. This parameter is applicable only for S-parameter data modeled in time domain. You can use this to shape spectral content with filter effects by specifying only magnitude.

Note This parameter introduces an artificial delay to the system.

Visualization

`Source of frequency data` — Frequency data source
`Extracted from data source` (default) | `User-defined`

Frequency data source, specified as:

When **Source of frequency data** is `Extracted from data source`, the **Data source** must be set to `Data file`. Verify that the specified **Data file** contains frequency data.

When **Source of frequency data** is `User-specified`, specify a vector of frequencies in the **Frequency data** parameter. Also, specify units from the corresponding drop-down list.

`Frequency data` — Frequency data range
`[1e9:1e6:3e9]` (default) | vector | `Hz` | `kHz` | `MHz` | `GHz`

Frequency data range, specified as a vector

`Plot type` — Type of data plot
`X-Y plane` (default) | `Polar plane` | `Z Smith chart` | `Y Smith chart` | `ZY Smith chart`

Type of data plot that you want to produce with your data specified as one of the following:

- `X-Y plane` — Generate a Cartesian plot of your data versus frequency. To create linear, semilog, or log-log plots, set the **Y-axis scale** and **X-axis scale** accordingly.
- `Polar plane` — Generate a polar plot of your data. The block plots only the range of data corresponding to the specified frequencies.
- `Z smith chart`, `Y smith chart`, and `ZY smith chart` — Generate a Smith chart. The block plots only the range of data corresponding to the specified frequencies.

`Parameter 1` — Type of S-Parameters to plot
`S11` (default) | `SNN`

Type of S-Parameters to plot, specified as `SNN`, where *N* is the number of ports in the s-parameters block.

`Parameter 2` — Type of S-Parameters to plot
`None` (default) | `SNN`

Type of S-Parameters to plot, specified as `SNN`, where *N* is the number of ports in the s-parameters block.

`Format1` — Plot format
`Magnitude (decibels)` (default) | `Angle(degrees)` | `Real` | `Imaginary`

Plot format, specified as `Magnitude (decibels)`, `Angle(degrees)`, `Real`, or `Imaginary`.

`Format2` — Plot format
`Magnitude (decibels)` (default) | `Angle(degrees)` | `Real` | `Imaginary`

Plot format, specified as `Magnitude (decibels)`, `Angle(degrees)`, `Real`, or `Imaginary`.

`Y-axis scale` — Y-axis scale
`Linear` (default) | `Logarithmic`

Y-axis scale, specified as `Linear` or `Logarithmic`.

`X-axis scale` — X-axis scale
`Linear` (default) | `Logarithmic`

X-axis scale, specified as `Linear` or `Logarithmic`.

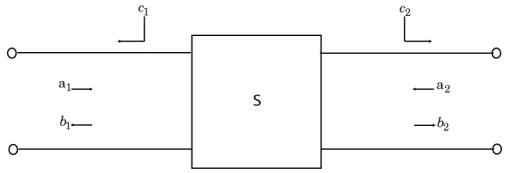
`Plot` — Plot specified data
button

Plot specified data using plot button.

More About

Generate Thermal Noise

You can only generate thermal noise if the given S-parameters multiport components are passive.



To include the noise waves, the block augments the S-parameters equation:

$$b = Sa + c$$

- a and b — Customary wave vectors
- c — Noise wave vector

The noise wave vector, noise correlation matrix, and S-parameters have these relationships:

$$C_s = K * T(I - SS^{\dagger}) = \overline{cc^{\dagger}}$$

where:

- K — Boltzmann's constant
- T — System temperature
- c — Noise wave vector

References

[1] Wedge, Scott & Rutledge, David. " Wave Techniques for Noise Modeling and Measurement" *IEEE Transactions on Microwave Theory and Techniques*. Vol. 40, Number 11, pp. 2004–2012, Nov. 1992.

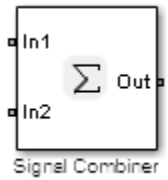
See Also

Amplifier | Mixer | Transmission Line

Introduced in R2010b

Signal Combiner

Compute sum of RF signals



Library

Elements

Description

Use the Signal Combiner block to sum signals across each carrier frequency in the RF Blockset circuit envelope simulation environment. For an introduction to RF simulation, see the example, “Simulate High Frequency Components”

Parameters

Ground and hide negative terminals

Select this option to internally ground and hide the negative terminals. Clear this to expose the negative terminals. By exposing these terminals, you can connect them to other parts of your model.

By default, this option is selected.

Examples

- The example, “Measuring Image Rejection Ratio in Receivers” shows how to use a signal combiner to perform image rejection.

- The example, “Carrier to Interference Performance of Weaver Receiver” uses a Signal Combiner block as part of a realization of the Weaver receiver architecture.

See Also

Sinusoid

Model DC offset and sinusoidal modulation
Library: RF Blockset / Circuit Envelope / Sources

Description

The Sinusoid block implements a voltage or current source that provides a DC offset and sine wave modulation. This block can be used with each listed block carrier in the circuit envelope environment.

The block implements the following voltage (or current) relationships for the in-phase (u_i), and quadrature (u_q), components of the k^{th} listed block carrier:

$$u_{k,i}(t) = D_i + A_i \sin(w_k(t - \tau))$$
$$u_{k,q}(t) = D_q + A_q \sin(w_k(t - \tau))$$

where:

- D_i and D_q are DC offsets.
- A_i and A_q are in-phase and quadrature amplitudes.
- τ is the time delay.
- ω_k is the specified modulation frequency at a given carrier frequency f_k.
- t is the time.

Note Sinusoid block does not support frame-based processing (supported by the Configuration blocks) as it uses SL Sine wave block. This block errors if the **Samples per frame** is more than 1 in the configuration block.

Parameters

`Source type` — Wave type
`Ideal voltage` (default) | `Ideal current`

Wave type, specified as:

- `Ideal Voltage` — The block simulates a voltage envelope $v(t)$ at the specified **Carrier frequencies**.
- `Ideal Current` — The block simulates a current envelope $i(t)$ at the specified **Carrier frequencies**.

`Offset in-phase` — In-phase offset
`0 V` (default) | `0 A` | vector of real or complex numbers | `V` | `mV` | `kV` | `A` | `mA` | `uA` | `kA`

In-phase offset for each of the RF circuit carrier frequencies, specified as a vector of real or complex numbers. Specify the units from the corresponding drop-down list. The units are in volts for `Ideal Voltage` and amperes for `Ideal Current`.

`Offset quadrature` — Quadrature offset
`0 V` (default) | `0 A` | vector of real or complex numbers | `V` | `mV` | `kV` | `A` | `mA` | `uA` | `kA`

Quadrature offset for each of the RF circuit carrier frequencies, specified as a vector of real or complex numbers. Specify the units from the corresponding drop-down list. The units are in volts for `Ideal Voltage` and amperes for `Ideal Current`.

`Sinusoidal amplitude in-phase` — In-phase amplitude
`0` (default) | vector of real or complex numbers | `V` | `mV` | `kV` | `A` | `mA` | `uA` | `kA`

In-phase amplitude, specified as a vector of real number or a complex number. Specify the units from the corresponding drop-down list. The units are in volts for `Ideal Voltage` and amperes for `Ideal Current`.

`Sinusoidal amplitude quadrature` — Quadrature amplitude
`0` (default) | vector of real or complex numbers | `V` | `mV` | `kV` | `A` | `mA` | `uA` | `kA`

Quadrature amplitude, specified as a vector of real number or a complex number. Specify the units from the corresponding drop-down list. The units are in volts for `Ideal Voltage` and amperes for `Ideal Current`.

`Sinusoidal modulation frequency` — Modulation frequency
`0` (default) | vector of real or complex numbers | `Hz` | `kHz` | `MHz` | `GHz`

Modulation frequency at each of the RF circuit carrier frequencies, specified as a vector of real number or a complex number. Specify the units from the corresponding drop-down list. Th

`Time delay` — Time delay of modulation
`0` (default) | vector of real | `s` | `ms` | `us` | `ns`

Time delay of modulation, specified as a vector of positive numbers. Specify the units from the corresponding drop-down list.

`Carrier frequencies` — Carrier frequencies
`0` (default) | vector of real positive numbers | `Hz` | `kHz` | `MHz` | `GHz`

Carrier frequencies, specified as a vector of real positive numbers. The elements in the carrier frequencies are combinations of fundamental tones and corresponding harmonics in the Configuration block. The default value is `0 Hz`.

`Ground and hide negative terminals` — Ground RF circuit terminals
`on` (default) | `off`

Select this parameter to ground and hide the negative terminals. To expose the negative terminals, clear the parameter. By exposing these terminals, you can connect them to other parts of your model.

By default, this option is selected.

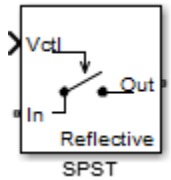
See Also

Continuous Wave | Noise

Introduced in R2010b

SPST

Single pole single throw switch



Library

Junctions

Description

The SPST block models a single pole single throw switch. The input Simulink signal, V_{ctl}, controls when the signal at the input port is transferred to the output port.

For resistance characterization:

- If the control voltage is less than the threshold voltage, the block attenuates the signal using the resistance value specified in **Off resistance**. The block places the resistor between the positive input and output terminals.
- If the control voltage is greater than or equal to the threshold voltage, the block attenuates the signal using the resistance value specified in **On resistance**.
- When the switch loading type is absorptive, and the control voltage is less than the threshold voltage, the block places internal shunt resistors at the input and output ports.

For insertion loss characterization:

- If the control voltage is less than the threshold voltage, calculate the off resistance value using the **Isolation (dB)** value.
- If the control voltage is greater than or equal to the threshold voltage, calculate the on resistance value using the **Insertion loss (dB)** value.

- When the switch loading type is absorptive, and the control voltage is less than the threshold voltage, the block places internal shunt resistors at the input and output ports.

The shunt resistor values provide matching terminations at the input and output ports.

The voltage–current relationship at the terminals depends on the relationship between control voltage, V_{ctl}, and the threshold voltage, $V_{threshold}$.

- If $V_{ctl} < V_{threshold}$

  $$I_{in} = V_{inout} \cdot G_{off} + V_{in} \cdot G_{Z01}$$
  $$I_{out} = -V_{inout} \cdot G_{off} + V_{out} \cdot G_{Z02}$$

- If $V_{ctl} \geq V_{threshold}$

  $$I_{in} = V_{inout} \cdot G_{off}$$
  $$I_{out} = -V_{inout} \cdot G_{on}$$

- G_{on}, G_{off} — On and off path conductances
- G_{Z01}, G_{Z02} — Shunt port conductances

Parameters

Threshold voltage

Threshold voltage of the switch, specified as a positive scalar. The default value is `0` volts.

Characterization

Characterization of the SPST switch: `Resistance` or `Insertion loss`. The default value is `Resistance`.

On resistance

On resistance value of the switch, specified as a positive scalar. The default value is `10` ohms. **On resistance** is available when you set **Characterization** to `Resistance`.

Off resistance

Off resistance value of the switch, specified as a positive scalar. The default value is `1e6` ohms. **Off resistance** is available when you set **Characterization** to `Resistance`.

Insertion loss (dB)

Insertion loss value of the switch, specified as a positive scalar. The default value is `1` dB. **Insertion loss (dB)** is available when you set **Characterization** to `Insertion loss`.

Isolation (dB)

Isolation value of the switch, specified as a positive scalar. The default value is `70` dB. **Isolation (dB)** is available when you set **Characterization** to `Insertion loss`

Loading type

Loading type of the `SPST` switch: `Absorptive` or `Reflective`. The default is `Reflective`.

Port terminations

Port termination that matches the impedance conditions of the `SPST` switch. The default value is `50` ohms.

Note **Port terminations** is available when you set the `SPST` switch **Characterization** to `Insertion loss` or when you set **Loading type** to `Absorptive`.

Ground and hide reference terminals

Select this option to internally ground and hide the reference terminals. Clear the option to expose the reference terminals. By exposing these terminals, you can connect them to other parts of your model.

By default, this option is selected.

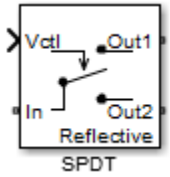
See Also

Potentiometer | SPDT | Switch

Introduced in R2015b

SPDT

Single pole double throw switch



Library

Junctions

Description

The SPDT block models a single pole double throw switch. The input Simulink signal, V_{ctl}, controls the transfer of the signal at the input port, In, to one of the two output ports.

If the control voltage is less than the threshold voltage, the block transfers the input signal to output port 1. If control voltage is greater than or equal to the threshold voltage, the block transfers the input signal to output port 2. Two different resistors regulate the conduction path for transfer of input signal from input port to either output port.

The voltage–current relationship at the terminals depends on the relationship between control voltage, V_{ctl}, and the threshold voltage, $V_{threshold}$.

- If $V_{ctl} < V_{threshold}$:

  $$I_{in} = V_{in1} \cdot G_{on} + V_{in2} \cdot G_{off}$$
  $$I_1 = -V_{in1} \cdot G_{on}$$
  $$I_2 = -V_{in2} \cdot G_{off} + V_2 \cdot G_{Z02}$$

- If $V_{ctl} \geq V_{threshold}$:

  $$I_{in} = V_{inout} \cdot G_{off}$$
  $$I_{out} = -V_{inout} \cdot G_{on}$$

- I_{in}, I_1 and I_2 — Currents into the input terminal and two output terminals
- V_{in1}, V_{in2} — Voltages between the input terminal and the output 1 and output 2 terminals, respectively
- V_1, V_2 — Voltages at output port 1 and output port 2, respectively
- G_{on}, G_{off} — On and off path conductances
- G_{Z01}, G_{Z02} — Shunt port conductances

Parameters

Threshold voltage

Threshold voltage of the switch, specified as a positive scalar. The default value is `0` volts.

Characterization

Characterization of the SPDT switch: `Resistance` or `Insertion loss`. The default value is `Resistance`.

On resistance

On resistance value of the switch, specified as a positive scalar. The default value is `10` ohms. **On resistance** is available when you set **Characterization** to `Resistance`.

Off resistance

Off resistance value of the switch, specified as a positive scalar. The default value is `1e6` ohms. **Off resistance** is available when you set **Characterization** to `Resistance`.

Insertion loss (dB)

Insertion loss value of the switch, specified as a positive scalar. The default value is `1` dB. **Insertion loss (dB)** is available when you set **Characterization** to `Insertion loss`.

Isolation (dB)

Isolation value of the switch, specified as a positive scalar. The default value is `70` dB. **Isolation (dB)** is available when you set **Characterization** to `Insertion loss`

Loading type

Loading type of the SPDT switch: `Absorptive` or `Reflective`. The default is `Reflective`.

Port terminations

Port termination that matches the impedance conditions of the `SPDT` switch. The default value is `50` ohms.

Note **Port terminations** is available when you set the `SPDT` switch **Characterization** to `Insertion loss` or when you set **Loading type** to `Absorptive`.

Ground and hide reference terminals

Select this option to internally ground and hide the reference terminals. Clear the option to expose the reference terminals. By exposing these terminals, you can connect them to other parts of your model.

By default, this option is selected.

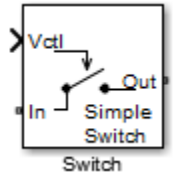
See Also

Potentiometer | SPST | Switch

Introduced in R2015b

Switch

Simulink controlled two–terminal switch



Library

Junctions

Description

The switch block models a two terminal switch. The input Simulink signal, V_{ctl} , controls the transfer of the RF Blockset signal from the `In` terminal to the `Out` terminal.

If the control voltage is less than the threshold voltage, the block attenuates the signal using the resistance value specified in **Off resistance** between the input and output terminals. If the control voltage is greater than the threshold voltage, the block passes the signal using the resistance value specified in **On resistance** between the input and output terminals.

The voltage–current relationship for the switch, or controlled resistor:

- If $V_{ctl} < V_{thres}$:

  $$I_{res} . R_{off} = V_{res}$$

- If $V_{ctl} \geq V_{thres}$:

  $$I_{res} . R_{on} = V_{res}$$

R_{off} is the off resistance and R_{on} is the on resistance of the switch.

If R_{on} is less than R_{off} , and V_{ctl} is greater or equal to V_{thres}, the switch is on.

If R_{on} is greater than R_{off}, and V_{ctl} is greater or equal to V_{thres}, the switch is off.

Parameters

On resistance

On resistance value of the switch, specified as a positive scalar. The default value is `10` ohms.

Off resistance

Off resistance value of the switch, specified as a positive scalar. The default value is `1e6` ohms.

Threshold voltage

Threshold voltage of the switch. The default value is `0` volts.

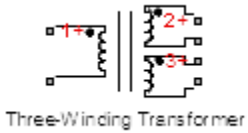
See Also

Potentiometer | SPDT | SPST

Introduced in R2015b

Three-Winding Transformer

Model three coupled inductors for circuit envelope analysis



Library

Elements

Description

The Three-Winding Transformer block models three coupled inductors within the RF Blockset circuit envelope simulation environment. For an introduction to RF simulation, see the example, “Simulate High Frequency Components”.

The block implements the relations

$$v_1(t) = L_1\frac{d}{dt}[i_1(t)] + M_{12}\frac{d}{dt}[i_2(t)] + M_{13}\frac{d}{dt}[i_3(t)]$$

$$v_2(t) = M_{12}\frac{d}{dt}[i_1(t)] + L_2\frac{d}{dt}[i_2(t)] + M_{23}\frac{d}{dt}[i_3(t)]$$

$$v_3(t) = M_{13}\frac{d}{dt}[i_1(t)] + M_{23}\frac{d}{dt}[i_2(t)] + L_3\frac{d}{dt}[i_3(t)]$$

$$M_{pq} = K_{pq}\sqrt{L_pL_q}$$

where:

- L_1, L_2, and L_3 represent inductances.
- M_{pq} represents the mutual inductance between the pth and qth inductors, with coefficient of coupling K_{pq}.

- $v_1(t)$, $v_2(t)$, and $v_3(t)$ represent the voltage across the terminals of the inductors at time t.
- $i_1(t)$, $i_2(t)$, and $i_3(t)$ represent the current through the inductors at time t. The block uses standard dot notation to indicate the direction of positive current flow relative to a positive voltage.

RF Blockset current and voltage signals consist of in-phase (I_k) and quadrature (Q_k) components at each frequency f_k specified in the Configuration block:

$$i(t) = \sum_{\{f_k\}} \left(i_{I_k}(t) + j \cdot i_{Q_k}(t)\right) e^{j(2\pi f_k)t}$$

$$v(t) = \sum_{\{f_k\}} \left(v_{I_k}(t) + j \cdot v_{Q_k}(t)\right) e^{j(2\pi f_k)t}$$

Parameters

Inductance L1

Specify the inductance of the first inductor, L_1, as a scalar value greater than or equal to `0`. Specify the units of the inductance from the corresponding drop-down list. The default value of this parameter is `1e-6 H`.

Inductance L2

Specify the inductance of the second inductor, L_2, as a scalar value greater than or equal to `0`. Specify the units of the inductance from the corresponding drop-down list. The default value of this parameter is `1e-6 H`.

Inductance L3

Specify the inductance of the third inductor, L_3, as a scalar value greater than or equal to `0`. Specify the units of the inductance from the corresponding drop-down list. The default value of this parameter is `1e-6 H`.

Coefficient of coupling K12

Specify the coefficient of coupling for the mutual inductance of the first and second inductors, K_{12}, as a scalar value between `0` and `1`, inclusive. The default value of this parameter is `0.9`.

Coefficient of coupling K13

Specify the coefficient of coupling for the mutual inductance of the first and third inductors, K_{13}, as a scalar value between `0` and `1`, inclusive. The default value of this parameter is `0.9`.

Coefficient of coupling K23

Specify the coefficient of coupling for the mutual inductance of the second and third inductors, K_{23}, as a scalar value between `0` and `1`, inclusive. The default value of this parameter is `0.9`.

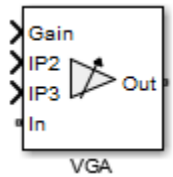
Note The minimum nonzero inductance value that the RF Blockset environment recognizes is `1e-18 H`. During simulation, the block uses a value of `1e-18 H` for any inductance and mutual inductance values specified between `0` and `1e-18 H`.

See Also

Inductor | Mutual Inductor

VGA

Variable gain amplifier



Library

Elements

Description

The VGA block models a RF Blockset variable gain amplifier. The Simulink signal controls the non-linear gain. The VGA output voltage is a function of the input voltage:

$$V_{out} = g * V_{in} + c_2 * V_{in}^2 + c_3 * V_{in}^2$$

The table shows the formulas for the coefficients, g, c_2, and c_3

Coefficient Formulae

	Input Referred	Output Referred
g	$g = \frac{2\sqrt{G * Z_{out} * Z_{in}}}{\lvert Z_{in} \rvert}$	
c_2	$C_2 = \frac{g}{\sqrt{2 * IP2 * Z_{in}}} volts^{-1}$	$C_2 = \frac{g^2}{\sqrt{8 * IP2 * Z_{out}}} volts^{-1}$
c_3	$C_3 = \frac{4}{3} \cdot \frac{g}{2 * IP3 * Z_{in}} volts^{-2}$	$C_2 = \frac{4}{3} \cdot \frac{g^3}{8 * IP3 * Z_{out}} volts^{-2}$

The three Simulink input ports are `Gain`, `IP2` (second-order intercept point), and `IP3` (third-order intercept point). `In` is the RF Blockset voltage input and `OUT` is RF Blockset voltage output.

Parameters

Input impedance (Ohm)

Input impedance of the amplifier, specified as a scalar. The default value is `50` ohms.

Output impedance (Ohm)

Output impedance of the amplifier, specified as a scalar. The default value is `50` ohms.

Intercept points convention

Input-referred or output-referred convention for IP2 or IP3 intercept points, specified as `Input` or `Output`.

The default is `Output`.

Ground and hide negative terminals

Select this option to internally ground and hide the negative terminals. Clear the option to expose the negative terminals. By exposing these terminals, you can connect them to other parts of your model.

By default, this option is selected.

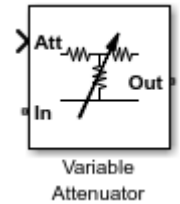
See Also

Amplifier

Introduced in R2015b

Variable Attenuator

Model variable attenuator
Library: RF Blockset / Circuit Envelope / Elements



Description

The Variable Attenuator block attenuates the signal power by a given factor known as Insertion Loss in dB. Using the Variable Attenuator block, you can vary the attenuation of the input Simulink signal during simulation. Commonly, the block matches the impedance of the RF circuit at the input and output ports. You can use attenuators to dampen the power of the incoming signal to protect RF circuits.

Parameters

`Minimum attenuation (dB)` — Lowest value for insertion loss or attenuation
`0.001` (default) | scalar

Lowest value of insertion loss or attenuation to apply to the signal, specified as a scalar in dB.

`Maximum attenuation (dB)` — Highest value of insertion loss or attenuation
`1000` (default) | scalar

Highest value of insertion loss or attenuation to apply to the signal, specified as a scalar in dB.

`Input impedance (Ohm)` — Input impedance
`50` (default) | scalar

Input impedance of the attenuator, specified as a scalar in ohms.

`Output impedance (Ohm)` — Output impedance
`50` (default) | scalar

Output impedance of the attenuator, specified as a scalar in ohms.

`Simulate noise` — Simulate thermal noise
on (default) | off

Select this parameter to simulate thermal noise in the attenuator. You must select **Simulate noise** in the Configuration block.

This parameter inserts a current noise source with the single-sided power density of 4 kT/R A^2/Hz, where:

- T is the value of the **Temperature** parameter in the Configuration block. Units are in degrees Kelvin.
- k is the Boltzmann constant.

`Ground and hide negative terminals` — Ground RF circuit terminals
on (default) | off

Select this parameter to ground and hide the negative terminals. To expose the negative terminals, clear this parameter. By exposing these terminals, you can connect them to other parts of your model.

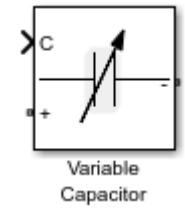
See Also

Attenuator | Variable Inductor | Variable Capacitor | Variable Phase Shift

Introduced in R2016b

Variable Capacitor

Model variable capacitor
Library: RF Blockset / Circuit Envelope / Elements



Description

The Variable Capacitor block controls the output of RF Blockset feedback circuits using Simulink controlled capacitance in farads. The minimum value of the capacitance (*Cmin*) is a RF Blockset defined constant independent of the Simulink control signal. The block has two electrical terminals. One terminal is for the Simulink control signal and one terminal is for the RF Blockset signal.

Parameters

The Variable Capacitor block has no parameters.

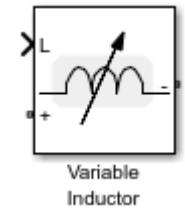
See Also

Capacitor | Variable Inductor

Introduced in R2016b

Variable Inductor

Model variable inductor
Library: RF Blockset / Circuit Envelope / Elements



Description

The Variable Inductor block controls the output of RF Blockset feedback circuits using Simulink controlled inductance in henries. The minimum value of the inductance (*Lmin*) is a RF Blockset defined constant independent of the Simulink control signal. The block has two electrical terminals. One terminal is for the Simulink control signal and one terminal is for the RF Blockset signal.

Parameters

The Variable Inductor block has no parameters.

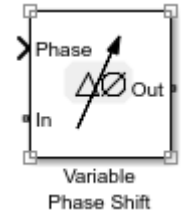
See Also

Inductor | Variable Capacitor

Introduced in R2016b

Variable Phase Shift

Model variable phase device
Library: RF Blockset / Circuit Envelope / Elements



Description

The Variable Phase Shift block controls the phase of the output signal of a RF Blockset circuit. The block uses a Simulink signal to control the phase of the circuit. One input terminal is for the Simulink control signal and one input terminal is for the RF Blockset signal.

Phase shifter are commonly used in phased array antenna systems. In these systems, electronically controlled phase shifters steer the antenna beam in space. Phase shifters are also used in test and measurement systems.

Parameters

`Phase shift unit` — Angle units for controlling phase of circuit
`rad` (default) | `deg`

Angle units for controlling the phase of the signal, specified as `rad` for radians or `deg` for degrees.

`Reference impedances (Ohm)` — Reference impedance for phase shifter
`50` (default) | vector of positive scalars

Reference impedance for phase shifter, specified as a vector of positive scalars.

`Ground and hide negative terminals` — Ground RF circuit terminals
on (default) | off

Select this parameter to ground and hide the negative terminals. To expose the negative terminals, clear this parameter. By exposing these terminals, you can connect them to other parts of your model.

See Also

PhaseShift | Variable Inductor | Variable Attenuator | Variable Capacitor

Introduced in R2016b

Spectrum Analyzer

Display frequency spectrum of time-domain signals
Library: RF Blockset / Circuit Envelope / Utilities

Description

Note The Spectrum Analyzer block in the RF Blockset product contains a subset of functionality of the DSP System Toolbox™ block with the same name. This page describes the block configuration and functionality available with a RF Blockset license. If you also have a DSP System Toolbox license, then the Spectrum Analyzer block in the RF Blockset > Utilities library is identical to the block in the DSP System Toolbox > Sinks library. For more information, see Spectrum Analyzer in the DSP System Toolbox documentation.

The Spectrum Analyzer block accepts input signals with discrete sample times and displays frequency spectra of these signals.

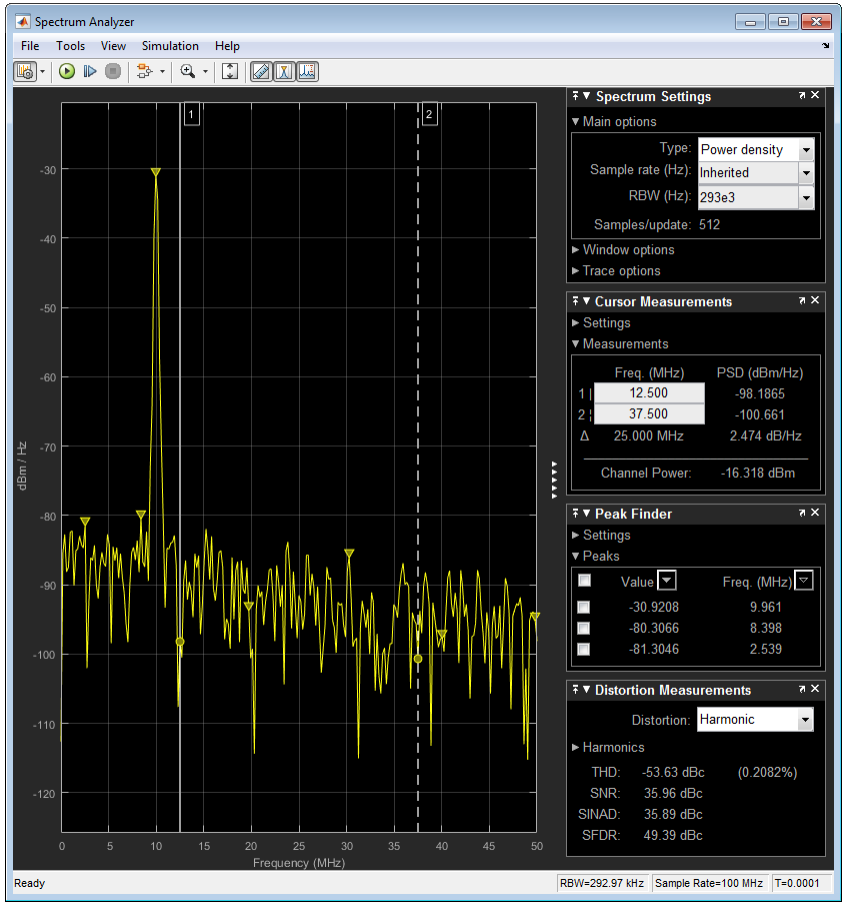
To use a Spectrum Analyzer block, instead of a regular scope, in a Simscape model:

1. Add a Spectrum Analyzer block to your block diagram.
2. If your model uses a variable-step solver, also add a Rate Transition block and connect it to the input of the Spectrum Analyzer, setting the **Output port sample time** to the sample time you wish the Spectrum Analyzer to use.

   If your model uses a local solver, then it produces output physical signals with discrete sample times and you do not need to add a Rate Transition block. However, if you need to down-sample from the solver fixed step size, you can also use a Rate Transition block. For more information on using local solvers, see “Making Optimal Solver Choices for Physical Simulation” (Simscape).
3. Use a PS-Simulink Converter block to connect the output physical signal of interest to the input of the Spectrum Analyzer block (or to the input of the Rate Transition

block, if using one). For more information, see “Connecting Simscape Diagrams to Simulink Sources and Scopes” (Simscape). You can also use additional signal processing blocks between the PS-Simulink Converter and the Spectrum Analyzer to enhance signal quality.

4 Run the simulation. The Spectrum Analyzer, referred to here as the scope, opens and displays the frequency spectrum of the signal.



Limitations

This reference page describes the Spectrum Analyzer block available with Simscape or RF Blockset. If you have DSP System Toolbox, more parameters and measurements are available. For information about the full Spectrum Analyzer, see Spectrum Analyzer.

Ports

Input

`Port_1` — Signals to visualize

scalar | vector | matrix | array

Connect the signals you want to visualize. You can have up to 96 input ports. Input signals can have these characteristics:

- **Signal Domain** — Frequency or time signals
- **Type** — Discrete (sample-based and frame-based).
- **Data type** — Any data type that Simulink supports. See “Data Types Supported by Simulink” (Simulink).
- **Dimension** — One dimensional (vector), two dimensional (matrix), or multidimensional (array). Input must have fixed number of channels. See “Signal Dimensions” (Simulink) and “Determine Output Signal Dimensions” (Simulink).

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `uint8` | `uint16` | `uint32` | `fixed point`

Parameters

This section lists the parameters available in the Spectrum Analyzer when you do not have DSP System Toolbox. For the full parameter list, see Spectrum Analyzer.

Spectrum Settings

The **Spectrum Settings** pane appears at the right side of the Spectrum Analyzer window. This pane controls how the spectrum is calculated. To show the Spectrum Settings, in the

Spectrum Analyzer menu, select **View > Spectrum Settings** or use the  button in the toolbar.

Main

Type — Type of spectrum to display

`Power` (default) | `Power density` | `RMS`

`Power` — Spectrum Analyzer shows the power spectrum.

`Power density` — Spectrum Analyzer shows the power spectral density. The power spectral density is the magnitude of the spectrum normalized to a bandwidth of 1 hertz.

`RMS` — Spectrum Analyzer shows the root mean squared spectrum.

Tunable: Yes

Programmatic Use

See `SpectrumType`.

Sample rate — Sample rate of the input signal in hertz

`Inherited` (default) | positive scalar

Select `Inherited` to use the same sample rate as the input signal. To specify a sample rate, delete `Inherited` and enter a sample rate value.

Programmatic Use

See SampleRate.

RBW (Hz) — Resolution bandwidth

`Auto` (default) | positive scalar

The resolution bandwidth in hertz. This parameter defines the smallest positive frequency that can be resolved. By default, this parameter is set to `Auto`. In this case, the Spectrum Analyzer determines the appropriate value to ensure that there are 1024 RBW intervals over the specified frequency span.

If you set this parameter to a numeric value, the value must allow at least two RBW intervals over the specified frequency span. In other words, the ratio of the overall frequency span to RBW must be greater than two:

$$\frac{span}{RBW} > 2$$

Tunable: Yes

Programmatic Use

See RBW.

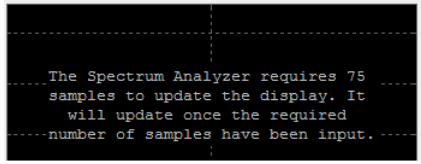
`Samples/update` — Required number of input samples

positive scalar

This property is read-only.

The number of input samples required to compute one spectral update. You cannot modify this parameter; it is shown in the spectrum analyzer for informational purposes only. This parameter is directly related to **RBW (Hz)/Window length/Number of frequency bands**. For more details, see "Algorithms" (DSP System Toolbox).

If the input does not have enough samples to achieve the resolution bandwidth that you specify, Spectrum Analyzer produces a message on the display.



Window

`Overlap (%)` — Segment overlap percentage

0 (default) | scalar between 0 and 100

This parameter defines the amount of overlap between the previous and current buffered data segments. The overlap creates a window segment that is used to compute a spectral estimate. The value must be greater than or equal to zero and less than 100.

Tunable: Yes

Programmatic Use

See `OverlapPercent`.

`Window` — Windowing method

`Hann` (default) | `Rectangular`

The windowing method to apply to the spectrum. Windowing is used to control the effect of sidelobes in spectral estimation. The window you specify affects the window length required to achieve a resolution bandwidth and the required number of samples per update. For more information about windowing, see “Windows” (Signal Processing Toolbox).

Tunable: Yes

Programmatic Use

See Window.

`NENBW` — Normalized effective noise bandwidth

scalar

This property is read-only.

The normalized effective noise bandwidth of the window. You cannot modify this parameter; it is shown for informational purposes only. This parameter is a measure of the noise performance of the window. The value is the width of a rectangular filter that accumulates the same noise power with the same peak power gain.

The rectangular window has the smallest NENBW, with a value of 1. All other windows have a larger NENBW value. For example, the Hann window has an NENBW value of approximately 1.5.

Trace

`Units` — Spectrum units

`dBm` (default)

This property is read-only.

The units of the spectrum. To change units, you must have the DSP System Toolbox.

Tunable: Yes

Programmatic Use

See SpectrumUnits.

`Averaging method` — Smoothing method

`Running` (default) | `Exponential`

Specify the smoothing method as:

- `Running` — Running average of the last *n* samples. Use the `Averages` property to specify *n*.
- `Exponential` — Weighted average of samples. Use the `Forgetting factor` property to specify the weighted forgetting factor.

For more information about the averaging methods, see “Averaging Method” on page 1-203.

Programmatic Use

See `AveragingMethod`.

`Averages` — Number of spectral averages

`1` (default) | positive integer

Specify the number of spectral averages as a positive integer. The spectrum analyzer computes the current power spectrum estimate by computing a running average of the last *N* power spectrum estimates. This parameter defines the number of spectral averages, *N*.

Dependency

This parameter applies only when **Averaging method** is `Running`.

Programmatic Use

See `SpectralAverages`.

`Forgetting factor` — Weighting forgetting factor

`0.9` (default) | scalar in the range (0,1]

Specify the exponential weighting as a scalar value greater than 0 and less than or equal to 1.

Dependency

This parameter applies only when the **Averaging method** is `Exponential`.

Programmatic Use

See `ForgettingFactor`.

`Reference load` — Reference load

`1` (default) | positive real scalar

The reference load in ohms that the Spectrum Analyzer uses as a reference to compute power values.

Dependency

To use this parameter, set “Input domain” (DSP System Toolbox) to `Time`.

Programmatic Use

See `ReferenceLoad`.

`Scale` — Scale of frequency axis

`Linear` (default) | `Logarithmic`

Choose a linear or logarithm scale for the frequency axis. When the frequency span contains negative frequency values, you cannot choose the logarithmic option.

Programmatic Use

See `FrequencyScale`.

`Offset` — Constant frequency offset

`0` (default) | scalar

The constant frequency offset to apply to the entire spectrum, or a vector of frequencies to apply to each spectrum for multiple inputs. The offset parameter is added to the values on the Frequency axis in the Spectrum Analyzer window. This parameter is not used in any spectral computations. You must take the parameter into consideration when you set the **Span (Hz)** and **CF (Hz)** parameters to ensure that the frequency span is within the “Nyquist frequency interval” (DSP System Toolbox).

Dependency

To use this parameter, set “Input domain” (DSP System Toolbox) to `Time`.

Programmatic Use

See `FrequencyOffset`.

`Two-sided spectrum` — Enable two-sided spectrum view

off (default) | on

Select this check box to enable a two-sided spectrum view. In this view, both negative and positive frequencies are shown. If you clear this check box, Spectrum Analyzer shows a one-sided spectrum with only positive frequencies. Spectrum Analyzer requires that this parameter is selected when the input signal is complex-valued.

Programmatic Use

See `PlotAsTwoSidedSpectrum`.

Configuration Properties

The **Configuration Properties** dialog box controls visual aspects of the Spectrum Analyzer. To open the Configuration Properties, in the Spectrum Analyzer menu, select **View** > **Configuration Properties** or select the  button in the toolbar dropdown.

`Title` — Display title

character vector | string

Specify the display title. Enter `%<SignalLabel>` to use the signal labels in the Simulink model as the axes titles.

Tunable: Yes

Programmatic Use

See `Title`.

`Show legend` — Display signal legend

off (default) | on

Show signal legend. The names listed in the legend are the signal names from the model. For signals with multiple channels, a channel index is appended after the signal name. Continuous signals have straight lines before their names and discrete signals have step-shaped lines.

From the legend, you can control which signals are visible. This control is equivalent to changing the visibility in the **Style** parameters. In the scope legend, click a signal name to hide the signal in the scope. To show the signal, click the signal name again. To show only

one signal, right-click the signal name, which hides all other signals. To show all signals, press **ESC**.

Note The legend only shows the first 20 signals. Any additional signals cannot be viewed or controlled from the legend.

Dependency

To enable this parameter, set "View" (DSP System Toolbox) to `Spectrum` or `Spectrum and spectrogram`.

Programmatic Use

See `ShowLegend`.

`Show grid` — Show internal grid lines

on (default) | off

Show internal grid lines on the Spectrum Analyzer

Programmatic Use

See `ShowGrid`.

`Y-limits (minimum)` — Y-axis minimum

`-80` (default) | scalar

Specify the minimum value of the *y*-axis.

Programmatic Use

See `YLimits`.

`Y-limits (maximum)` — Y-axis maximum

`20` (default) | scalar

Specify the maximum value of the *y*-axis.

Programmatic Use

See `YLimits`.

`Y-label` — Y-axis label
character vector | string

To display signal units, add `(%<SignalUnits>)` to the label. At the beginning of a simulation, Simulink replaces `(%SignalUnits)` with the units associated with the signals. For example, if you have a signal for velocity with units of m/s enter

```
Velocity (%<SignalUnits>)
```

Programmatic Use

See `YLabel`.

Style

The **Style** dialog box controls how to Spectrum Analyzer appears. To open the Style properties, in the Spectrum Analyzer menu, select **View** > **Style** or select the  button in the toolbar dropdown.

`Figure color` — Window background
gray (default) | color picker

Specify the color that you want to apply to the background of the scope figure.

`Plot type` — Plot type
`Line` (default) | `Stem`

Specify whether to display a `Line` or `Stem` plot.

Programmatic Use

See `PlotType`.

`Axes colors` — Axes background color
black (default) | color picker

Specify the color that you want to apply to the background of the axes.

`Properties for line` — Channel for visual property settings
channel names

Specify the channel for which you want to modify the visibility, line properties, and marker properties.

`Visible` — Channel visibility
on (default) | off

Specify whether the selected channel is visible. If you clear this check box, the line disappears. You can also change signal visibility using the scope legend.

`Line` — Line style
line, 0.5, yellow (default)

Specify the line style, line width, and line color for the selected channel.

`Marker` — Data point markers
`none` (default)

Specify marks for the selected channel to show at its data points. This parameter is similar to the 'Marker' property for plots. You can choose any of the marker symbols from the dropdown.

Axes Scaling

The **Axes Scaling** dialog box controls the axes limits of the Spectrum Analyzer. To open the Axes Scaling properties, in the Spectrum Analyzer menu, select **Tools > Axes Scaling > Axes Scaling Properties**.

`Axes scaling` — Automatic axes scaling
`Auto` (default) | `Manual` | `After N Updates`

Specify when the scope automatically scales the y-axis. By default, this parameter is set to `Auto`, and the scope does not shrink the y-axis limits when scaling the axes. You can select one of the following options:

- `Auto` — The scope scales the axes as needed, both during and after simulation. Selecting this option shows the **Do not allow Y-axis limits to shrink**.
- `Manual` — When you select this option, the scope does not automatically scale the axes. You can manually scale the axes in any of the following ways:
  - Select **Tools > Scaling Properties**.
  - Press one of the **Scale Axis Limits** toolbar buttons.
  - When the scope figure is the active window, press **Ctrl+A**.
- `After N Updates` — Selecting this option causes the scope to scale the axes after a specified number of updates. This option is useful, and most efficient, when your

frequency signal values quickly reach steady-state after a short period. Selecting this option shows the **Number of updates** edit box where you can modify the number of updates to wait before scaling.

Tunable: Yes

Programmatic Use

See AxesScaling.

`Do not allow Y-axis limits to shrink` — Axes scaling limits

on (default) | off

When you select this parameter, the y-axis is allowed to grow during axes scaling operations. If you clear this check box, the y-axis limits can shrink during axes scaling operations.

Dependency

This parameter appears only when you select `Auto` for the **Axis scaling** parameter. When you set the **Axes scaling** parameter to `Manual` or `After N Updates`, the *y*-axis limits can shrink.

`Number of updates` — Number of updates before scaling

`10` (default) | positive number

The number of updates after which the axes scale, specified as a positive integer. If the spectrogram is displayed, this parameter specifies the number of updates after which the color axes scales.

Tunable: Yes

Dependency

This parameter appears only when you set “Axes scaling/Color scaling” (DSP System Toolbox) to `After N Updates`.

Programmatic Use

See `AxesScalingNumUpdates`.

`Scale limits at stop` — Scale axes at stop

off (default) | on

Select this check box to scale the axes when the simulation stops. If the spectrogram is displayed, select this check box to scale the color when the simulation stops. The *y*-axis is always scaled. The *x*-axis limits are only scaled if you also select the **Scale X-axis limits** check box.

`Data range (%)` — Percent of axes
100 (default) | number in the range [1,100]

Set the percentage of the axis that the scope uses to display the data when scaling the axes. If the spectrogram is displayed, set the percentage of the power values range within the colormap. Valid values are from 1 through 100. For example, if you set this parameter to `100`, the scope scales the axis limits such that your data uses the entire axis range. If you then set this parameter to `30`, the scope increases the *y*-axis or color range such that your data uses only 30% of the axis range.

Tunable: Yes

`Align` — Alignment along axes
`Center` (default) | `Bottom` | `Top` | `Left` | `Right`

Specify where the scope aligns your data along the axis when it scales the axes. If the spectrogram is displayed, specify where the scope aligns your data along the axis when it scales the color. If you are using CCDF Measurements (DSP System Toolbox), the x axis is also configurable.

Tunable: Yes

Algorithms

Welch's Method

When you set the **Method** property to `Welch`, the following algorithms apply. The Spectrum Analyzer uses the `RBW` or the `Window Length` setting in the **Spectrum Settings** pane to determine the data window length. Then, it partitions the input signal into a number of windowed data segments. Finally, Spectrum Analyzer uses the modified periodogram method to compute spectral updates, averaging the windowed periodograms for each segment.

Spectrum Analyzer requires that a minimum number of samples to compute a spectral estimate. This number of input samples required to compute one spectral update is shown

as **Samples/update** in the **Main options** pane. This value is directly related to resolution bandwidth, *RBW*, by the following equation, or to the window length, by the equation shown in step 2.

$$N_{samples} = \frac{\left(1 - \frac{O_p}{100}\right) \times NENBW \times F_s}{RBW}$$

The normalized effective noise bandwidth, *NENBW*, is a factor that depends on the windowing method. Spectrum Analyzer shows the value of *NENBW* in the **Window Options** pane of the **Spectrum Settings** pane. Overlap percentage, O_p, is the value of the **Overlap %** parameter in the **Window Options** pane of the **Spectrum Settings** pane. F_s is the sample rate of the input signal. Spectrum Analyzer shows sample rate in the **Main Options** pane of the **Spectrum Settings** pane.

1. When in **RBW (Hz)** mode, the window length required to compute one spectral update, N_{window}, is directly related to the resolution bandwidth and normalized effective noise bandwidth:

   $$N_{window} = \frac{NENBW \times F_s}{RBW}$$

   When in **Window Length** mode, the window length is used as specified.

2. The number of input samples required to compute one spectral update, $N_{samples}$, is directly related to the window length and the amount of overlap by the following equation.

   $$N_{samples} = \left(1 - \frac{O_p}{100}\right) N_{window}$$

   When you increase the overlap percentage, fewer new input samples are needed to compute a new spectral update. For example, if the window length is 100, then the number of input samples required to compute one spectral update is given as shown in the following table.

O_p	$N_{samples}$
0%	100
50%	50
80%	20

3 The normalized effective noise bandwidth, *NENBW*, is a window parameter determined by the window length, N_{window}, and the type of window used. If $w(n)$ denotes the vector of N_{window} window coefficients, then *NENBW* is given by the following equation.

$$NENBW = N_{window} \times \frac{\sum_{n=1}^{N_{window}} w^2(n)}{\left[\sum_{n=1}^{N_{window}} w(n)\right]^2}$$

4 When in **RBW (Hz)** mode, you can set the resolution bandwidth using the value of the **RBW (Hz)** parameter on the **Main options** pane of the **Spectrum Settings** pane. You must specify a value to ensure that there are at least two RBW intervals over the specified frequency span. The ratio of the overall span to RBW must be greater than two:

$$\frac{span}{RBW} > 2$$

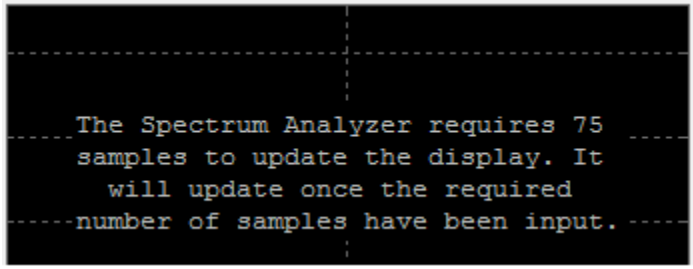
By default, the **RBW (Hz)** parameter on the **Main options** pane is set to `Auto`. In this case, the Spectrum Analyzer determines the appropriate value to ensure that there are 1024 RBW intervals over the specified frequency span. When you set **RBW (Hz)** to `Auto`, *RBW* is calculated as:

$$RBW_{auto} = \frac{span}{1024}$$

5 When in **Window Length** mode, you specify N_{window} and the resulting *RBW* is:

$$\frac{NENBW \times F_s}{N_{window}}$$

Sometimes, the number of input samples provided are not sufficient to achieve the resolution bandwidth that you specify. When this situation occurs, Spectrum Analyzer displays a message:



Spectrum Analyzer removes this message and displays a spectral estimate when enough data has been input.

Note The number of FFT points (N_{fft}) is independent of the window length (N_{window}). You can set them to different values if N_{fft} is greater than or equal to N_{window}.

Filter Bank

When you set the **Method** property to `Filter Bank`, the following algorithms apply. The Spectrum Analyzer uses the **RBW (Hz)** or the **Number of frequency band** property in the **Spectrum Settings** pane to determine the input frame length.

Spectrum Analyzer requires a minimum number of samples to compute a spectral estimate. This number of input samples required to compute one spectral update is shown as **Samples/update** in the **Main options** pane. This value is directly related to resolution bandwidth, *RBW*, by the following equation.

$$N_{samples} = \frac{F_s}{RBW}$$

F_s is the sample rate of the input signal. Spectrum Analyzer shows sample rate in the **Main Options** pane of the **Spectrum Settings** pane.

1. When in **RBW (Hz)** mode, you can set the resolution bandwidth using the value of the **RBW (Hz)** parameter on the **Main options** pane of the **Spectrum Settings** pane. You must specify a value to ensure that there are at least two RBW intervals over the specified frequency span. The ratio of the overall span to RBW must be greater than two:

$$\frac{span}{RBW} > 2$$

By default, the **RBW** parameter on the **Main options** pane is set to `Auto`. In this case, the Spectrum Analyzer determines the appropriate value to ensure that there are 1024 RBW intervals over the specified frequency span. Thus, when you set **RBW** to `Auto`, it is calculated by the following equation.$RBW_{auto} = \frac{span}{1024}$

2 When in **Number of frequency bands** mode, you specify the input frame size. When the number of frequency bands is `Auto`, the resulting RBW is:

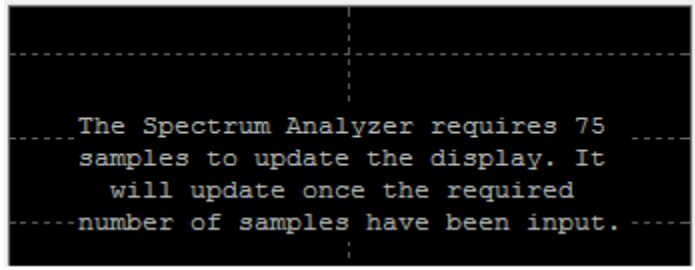
$$RBW = \frac{F_s}{\text{Input Frame Size}}$$

When the number of frequency bands is manually specified, the resulting RBW is:

$$RBW = \frac{F_s}{FFTLength}$$

For more information about the filter bank algorithm, see “Polyphase Implementation” (DSP System Toolbox).

Sometimes, the number of input samples provided are not sufficient to achieve the resolution bandwidth that you specify. When this situation occurs, Spectrum Analyzer displays a message:



Spectrum Analyzer removes this message and displays a spectral estimate when enough data has been input.

Nyquist frequency interval

When the PlotAsTwoSidedSpectrum property is set to `true`, the interval is $\left[-\frac{SampleRate}{2}, \frac{SampleRate}{2}\right] + FrequencyOffset$ hertz.

When the `PlotAsTwoSidedSpectrum` property is set to `false`, the interval is $\left[0, \frac{SampleRate}{2}\right] + FrequencyOffset$ hertz.

Periodogram and Spectrogram

Spectrum Analyzer calculates and plots the power spectrum, power spectrum density, and RMS computed by the modified Periodogram estimator. For more information about the Periodogram method, see `periodogram`.

Power Spectral Density — The power spectral density (PSD) is given by the following equation.

$$\mathrm{PSD}(f) = \frac{1}{P}\sum_{p=1}^{P}\frac{\left|\sum_{n=1}^{N_{FFT}} x^{p}[n]e^{-j2\pi f(n-1)T}\right|^{2}}{F_s \times \sum_{n=1}^{N_{window}} w^{2}[n]}$$

In this equation, $x[n]$ is the discrete input signal. On every input signal frame, Spectrum Analyzer generates as many overlapping windows as possible, with each window denoted as $x^{(p)}[n]$, and computes their periodograms. Spectrum Analyzer displays a running average of the P most current periodograms.

Power Spectrum — The power spectrum is the product of the power spectral density and the resolution bandwidth, as given by the following equation.

$$P_{spectrum}(f) = \mathrm{PSD}(f) \times RBW = \mathrm{PSD}(f) \times \frac{F_s \times NENBW}{N_{window}}$$

$$= \frac{1}{P} \sum_{p=1}^{P} \frac{\left| \sum_{n=1}^{N_{FFT}} x^p[n] e^{-j2\pi f(n-1)T} \right|^2}{\left[\sum_{n=1}^{N_{window}} w[n] \right]^2}$$

Frequency Vector

When set to `Auto`, the frequency vector for frequency-domain input is calculated by the software.

When the PlotAsTwoSidedSpectrum property is set to true, the frequency vector is:

$$\left[-\frac{SampleRate}{2}, \frac{SampleRate}{2} \right]$$

When the PlotAsTwoSidedSpectrum property is set to false, the frequency vector is:

$$\left[0, \frac{SampleRate}{2} \right]$$

Occupied BW

The *Occupied BW* is calculated as follows.

1. Calculate the total power in the measured frequency range.
2. Determine the lower frequency value. Starting at the lowest frequency in the range and moving upward, the power distributed in each frequency is summed until this result is

   $$\frac{100 - OccupiedBW\%}{2}$$

   of the total power.

3. Determine the upper frequency value. Starting at the highest frequency in the range and moving downward, the power distributed in each frequency is summed until the result reaches

$$\frac{100 - OccupiedBW\%}{2}$$

of the total power.

4. The bandwidth between the lower and upper power frequency values is the occupied bandwidth.

5. The frequency halfway between the lower and upper frequency values is the center frequency.

Distortion Measurements

The *Distortion Measurements* are computed as follows.

1. Spectral content is estimated by finding peaks in the spectrum. When the algorithm detects a peak, it records the width of the peak and clears all monotonically decreasing values. That is, the algorithm treats all these values as if they belong to the peak. Using this method, all spectral content centered at DC (0 Hz) is removed from the spectrum and the amount of bandwidth cleared (W_0) is recorded.

2. The fundamental power (P_1) is determined from the remaining maximum value of the displayed spectrum. A local estimate (Fe_1) of the fundamental frequency is made by computing the central moment of the power near the peak. The bandwidth of the fundamental power content (W_1) is recorded. Then, the power from the fundamental is removed as in step 1.

3. The power and width of the higher-order harmonics (P_2, W_2, P_3, W_3, etc.) are determined in succession by examining the frequencies closest to the appropriate multiple of the local estimate (Fe_1). Any spectral content that decreases monotonically about the harmonic frequency is removed from the spectrum first before proceeding to the next harmonic.

4. Once the DC, fundamental, and harmonic content is removed from the spectrum, the power of the remaining spectrum is examined for its sum ($P_{remaining}$), peak value ($P_{maxspur}$), and median value ($P_{estnoise}$).

5. The sum of all the removed bandwidth is computed as $W_{sum} = W_0 + W_1 + W_2 + ... + W_n$.

The sum of powers of the second and higher-order harmonics are computed as $P_{harmonic} = P_2 + P_3 + P_4 + ... + P_n$.

6 The sum of the noise power is estimated as:

$$P_{noise} = (P_{remaining} \cdot dF + P_{est.noise} \cdot W_{sum})/RBW$$

Where dF is the absolute difference between frequency bins, and RBW is the resolution bandwidth of the window.

7 The metrics for SNR, THD, SINAD, and SFDR are then computed from the estimates.

$$THD = 10 \cdot \log_{10}\left(\frac{P_{harmonic}}{P_1}\right)$$

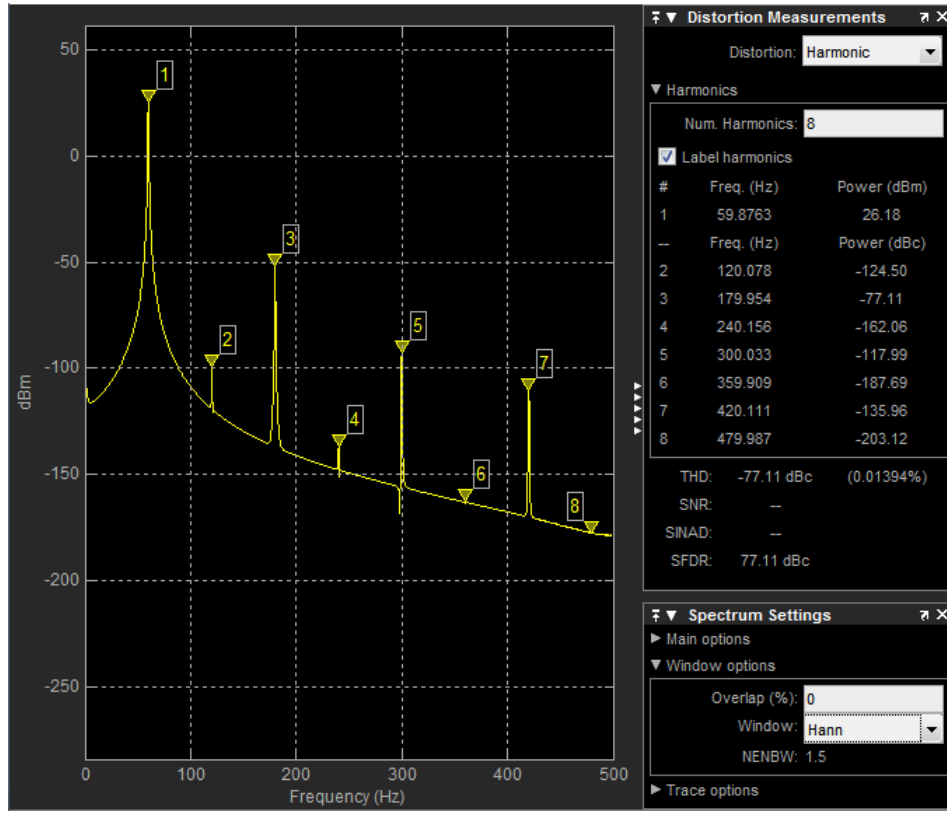
$$SINAD = 10 \cdot \log_{10}\left(\frac{P_1}{P_{harmonic} + P_{noise}}\right)$$

$$SNR = 10 \cdot \log_{10}\left(\frac{P_1}{P_{noise}}\right)$$

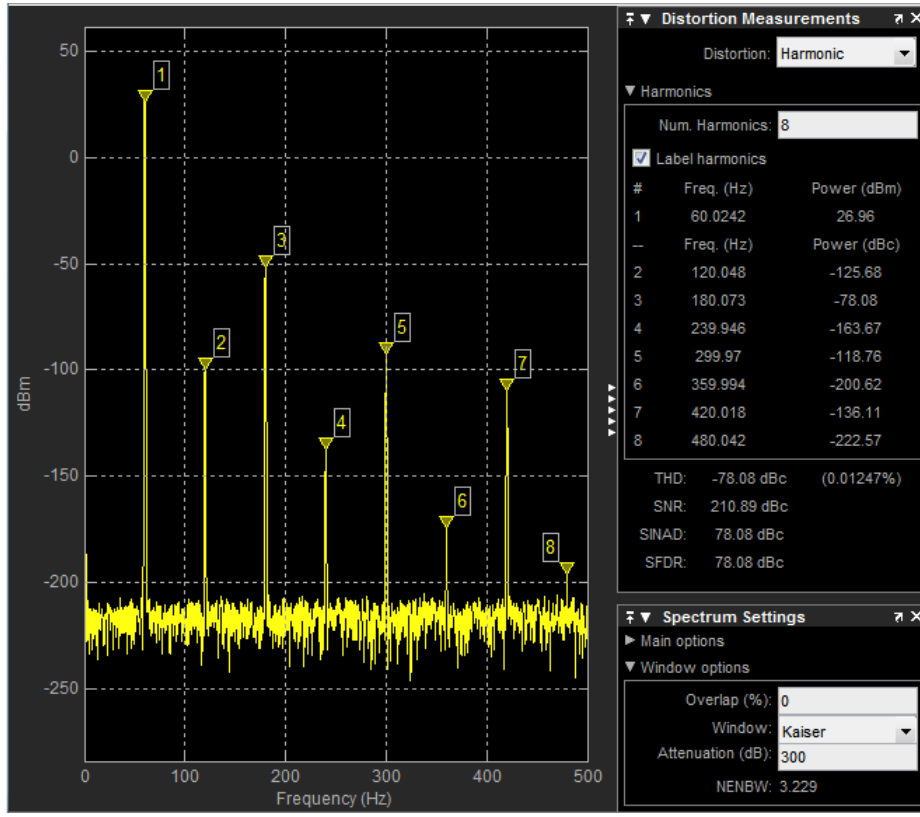
$$SFDR = 10 \cdot \log_{10}\left(\frac{P_1}{\max\left(P_{maxspur}, \max(P_2, P_3, \ldots, P_n)\right)}\right)$$

Harmonic Measurements

1 The harmonic distortion measurements use the spectrum trace shown in the display as the input to the measurements. The default Hann window setting of the Spectrum Analyzer may exhibit leakage that can completely mask the noise floor of the measured signal.



The harmonic measurements attempt to correct for leakage by ignoring all frequency content that decreases monotonically away from the maximum of harmonic peaks. If the window leakage covers more than 70% of the frequency bandwidth in your spectrum, you may see a blank reading (–) reported for **SNR** and **SINAD**. If your application can tolerate the increased equivalent noise bandwidth (ENBW), consider using a Kaiser window with a high attenuation (up to 330 dB) to minimize spectral leakage.



2 The DC component is ignored.

3 After windowing, the width of each harmonic component masks the noise power in the neighborhood of the fundamental frequency and harmonics. To estimate the noise power in each region, Spectrum Analyzer computes the median noise level in the nonharmonic areas of the spectrum. It then extrapolates that value into each region.

4 N^{th} order intermodulation products occur at $A*F1 + B*F2$,

where $F1$ and $F2$ are the sinusoid input frequencies and $|A| + |B| = N$. A and B are integer values.

5 For intermodulation measurements, the third-order intercept (TOI) point is computed as follows, where P is power in decibels of the measured power referenced to 1 milliwatt (dBm):

- $TOI_{lower} = P_{F1} + (P_{F2} - P_{(2F1-F2)})/2$
- $TOI_{upper} = P_{F2} + (P_{F1} - P_{(2F2-F1)})/2$
- $TOI = + (TOI_{lower} + TOI_{upper})/2$

Averaging Method

The moving averaging is done by using one of two methods:

- `Running` — For each frame of input, average the last N scaled Z vectors that are computed by the algorithm. The variable N is the value you specify for the number of spectral averages. If the algorithm does not have enough Z vectors, the algorithm uses zeros to fill the empty elements.
- `Exponential` — The moving average using the exponential weighting method is computed over the current Z vector and the previous Z vector. For details on the computation, see “Exponential Weighting Method” (DSP System Toolbox) in the `dsp.MovingAverage` object.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

This block can be used for simulation visibility in systems that generate code, but is not included in the generated code.

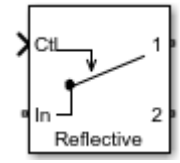
See Also

Spectrum Analyzer | `SpectrumAnalyzerConfiguration` | `dsp.SpectrumAnalyzer`

Introduced in R2017b

SPnT

Single pole multiple throw switch
Library: RF Blockset / Circuit Envelope / Junctions



Description

The SPnT block models a single pole multiple throw switch. This block supports both reflective and absorptive switches. You can specify two to eight output ports for the switch. The control signal is a Simulink signal connected to the `Ctl` port of the block. The control signal specifies which output port transmits the input signal. When the input to the `Ctl` port is between one and the number of output ports, the `Ctl` signal value is rounded to the nearest integer. The output port that corresponds to the integer value transmits the data. If the rounded `Ctl` signal value does not correspond to any of the output ports, all the output ports are turned off.

Parameters

Main

`Insertion loss (dB)` — Insertion loss of switch
`1` (default) | scalar

Insertion loss value of switch, specified as a finite scalar in decibels. Specify an insertion loss value greater than zero but less than the isolation value of the switch. For the equation, see “Insertion Loss” on page 1-206.

`Isolation (dB)` — Isolation value of switch
`70` (default) | scalar

Isolation value of switch, specified as a scalar in decibels. Specify an isolation value greater than the insertion loss value of the switch. For the equation, see “Isolation” on page 1-206.

`Loading type` — Switch type
`Absorptive` (default) | `Reflective`

Switch type, specified as `Absorptive` or `Reflective`.

`Number of outputs` — Number of outputs for switch
2 (default) | scalar

Number of outputs for the switch, specified as a scalar. The number of outputs must be greater than or equal to two or less than or equal to eight.

`Port terminations (Ohm)` — Port terminations
`50` (default) | scalar

Port terminations, specified as a scalar in ohms.

`Ground and hide reference terminals` — Ground RF circuit terminals
on (default) | off

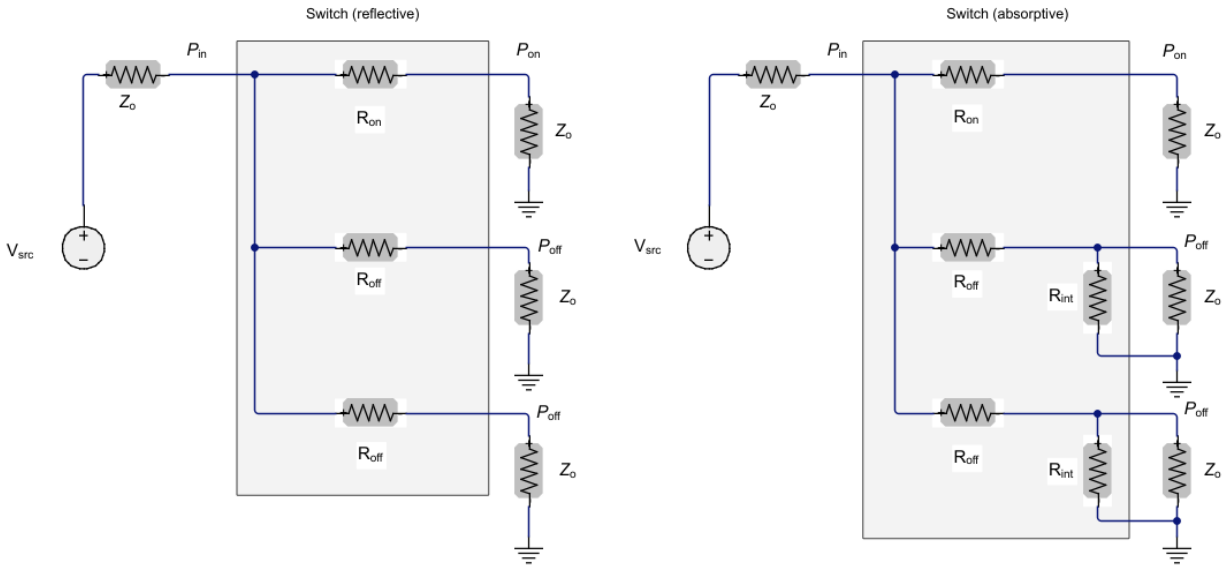
Select this parameter to internally ground and hide the reference terminals. To expose the reference terminals, clear this parameter. By exposing this terminal, you can connect it to other parts of your model.

More About

Reflective and Absorptive SPnT Switches

If there is no internal resistor connected in parallel to the load when the output terminal is turned off, any incident signal at the output side is reflected back to its source. This type of switch is known as *reflective switch*.

If there is a matching internal resistor connected in parallel to the load when the output terminal is turned off, any incident signal at the output side is absorbed by the switch. This type of switch is known as *absorptive switch*.



Insertion Loss

Insertion loss is given by the equation:

$$I_{\text{loss}}(\text{dB}) = -10\log_{10}\left(\frac{P_{\text{on}}}{P_{\text{in}}}\right)$$

Isolation

Isolation is given by the equation:

$$I_{\text{iso}}(\text{dB}) = -10\log_{10}\left(\frac{P_{\text{off}}}{P_{\text{in}}}\right)$$

References

[1] Razavi, Behzad. *RF Microelectronics*. Upper Saddle River, NJ: Prentice Hall, 2011.

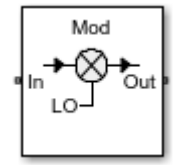
See Also

SPDT | SPST | Switch

Introduced in R2018a

Modulator

Model RF to RF modulator
Library: RF Blockset / Circuit Envelope / Systems



Description

The Modulator block models an RF to RF modulator.

Parameters

Main

`Source of conversion gain` — Source parameter of conversion gain
`Available power gain` (default) | `Open circuit voltage gain` | `Polynomial coefficients`

Source parameter of conversion gain, specified as one of the following:

- `Available power gain` — Relates the ratio of power of a single sideband (SSB) at the output to the input power. This calculation assumes a matched load and source termination.
- `Open circuit voltage gain` — Value of the open circuit voltage gain parameter as the linear voltage gain term of the polynomial voltage-controlled voltage source (VCVS).
- `Polynomial coefficients` — Implements a nonlinear voltage gain according to the polynomial you specify.

`Available power gain` — Ratio of power of SSB at the output to input power
`0` dB (default) | scalar in dB or a unitless ratio

Ratio of power of SSB at output to input power, specified as a scalar in dB or a unitless ratio. For a unitless ratio, select **None**.

Dependencies

To enable this parameter, set **Source of conversion gain** to `Available power gain`.

`Open circuit voltage gain` — Open circuit voltage gain

`0 dB` (default) | scalar

Open circuit voltage gain, specified as a scalar in dB.

Dependencies

To enable this parameter, set **Source of conversion gain** to `Open circuit voltage gain`.

`Polynomial coefficients` — Coefficients of polynomial specifying voltage gain

`[0 1]` (default) | vector

Polynomial coefficients, specified as a vector.

The order of the polynomial must be less than or equal to 9. The coefficients must be ordered in ascending powers. If a vector has 10 coefficients, $[a_0, a_1, a_2, \ldots a_9]$, the polynomial it represents is:

$V_{out} = a_0 + a_1 V_{in} + a_2 {V_{in}}^2 + \ldots + a_9 {V_{in}}^9$
a_1 represents the linear gain term, and higher-order terms are modeled according to [2].

For example, the vector $[a_0, a_1, a_2, a_3]$ specifies the relation $V_{out} = a_0 + a_1 V_{in} + a_2 {V_{in}}^2 + a_3 {V_{in}}^3$. Trailing zeros are omitted. So, $[a_0, a_1, a_2]$ defines the same polynomial as $[a_0, a_1, a_2, 0]$.

The default value is `[0,1]`, corresponding to the linear relation $V_{out} = V_{in}$.

Dependencies

To enable this parameter, set **Source of conversion gain** to `Polynomial coefficients`.

`Local oscillator frequency` — Local oscillator (LO) frequency

`0 Hz` (default) | scalar

Local oscillator (LO) frequency, specified as a scalar in Hz, kHz, MHz, or GHz.

`Input impedance (Ohm)` — Input impedance of modulator
`50` (default) | scalar

Input impedance of modulator, specified as a scalar in Ohms.

`Output impedance (Ohm)` — Output impedance of modulator
`50` (default) | scalar

Output impedance of modulator, specified as a scalar in Ohms.

`Add Image Reject filter` — Image reject (IR) filter parameters
`off` (default) | `on`

Select to add the **IR filter** parameter tab. Clear to remove the tab.

`Add Channel Select filter` — Channel select (CS) filter parameters
`off` (default) | `on`

Select to add the **CS filter** parameter tab. Clear to remove the tab.

`Ground and hide negative terminals` — Ground and hide negative terminals
`on` (default) | `off`

Select to internally ground and hide the negative terminals. Clear to expose the negative terminals. When the terminals are exposed, you can connect them to other parts of your model.

`Edit System` — Break modulator block links and replace internal variables by appropriate values
button

Use this button to break modulator links to the library. The internal variables are replaced by their values which are estimated using modulator parameters. The Modulator becomes a simple subsystem masked only to keep the icon.

Use **Edit System** to edit the internal variables without expanding the subsystem. Use **Expand System** to expand the subsystem in Simulink canvas and to edit the subsystem.

Impairments

`LO to Out isolation` — Ratio of magnitude of LO voltage to leaked voltage at output port (RF)

`inf` dB (default) | scalar

Ratio of magnitude of LO voltage to leaked voltage at output port (RF), specified as a scalar in dB, or a unitless ratio. For a unitless ratio, select **None**.

`Noise figure (dB)` — Signal-to-noise ratio (SNR) between outputs and input

`0` (default) | scalar

Single-sideband noise figure of mixer, specified as a scalar in dB.

To model noise in a circuit envelope model with a Modulator block, you must select the **Simulate noise** check box in the Configuration block dialog box.

`Add phase noise` — Add phase noise

`off` (default) | `on`

Select this parameter to add phase noise to your modulator system.

`Phase noise frequency offset (Hz)` — Phase noise frequency offset

`1` (default) | scalar | vector | matrix

Phase noise frequency offset, specified as a scalar, vector, or matrix with each element unit in Hz.

If you specify a matrix, each column corresponds to a non-DC carrier frequency of the CW source. The frequency offset values bind the envelope bandwidth of the simulation. For more information, see Configuration.

Dependencies

To enable this parameter, select **Add phase noise**.

`Phase noise level (dBc/Hz)` — Phase noise level

`-Inf` (default) | scalar | vector | matrix

Phase noise level, specified as a scalar, vector, or matrix with each element in dBc/Hz.

If you specify a matrix, each column corresponds to a non-DC carrier frequency of the CW source. The frequency offset values bind the envelope bandwidth of the simulation. For more information, see Configuration.

Dependencies

To enable this parameter, select **Add phase noise**.

`Automatically estimate impulse response duration` — Automatically estimate impulse response duration
`on` (default) | `off`

Select to automatically estimate impulse response for phase noise. Clear to specify the impulse response duration using **Impulse response duration**.

`Impulse response duration` — Impulse response duration
`1e-10 s` (default) | scalar

Impulse response duration used to simulate phase noise, specified as a scalar in s, ms, us, or ns.

Note The phase noise profile resolution in frequency is limited by the duration of the impulse response used to simulate it. Increase this duration to improve the accuracy of the phase noise profile. A warning message appears if the phase noise frequency offset resolution is too high for a given impulse response duration. This message also specifies the minimum duration suitable for the required resolution

---

Dependencies

To set this parameter, first clear **Automatically estimate impulse response duration**.

Nonlinearity

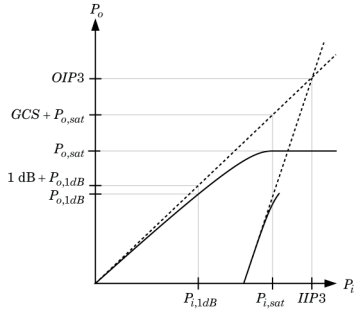
Selecting `Polynomial coefficients` for **Source of conversion gain** in the **Main** tab removes the **Nonlinearity** parameters.

`Nonlinear polynomial type` — Polynomial nonlinearity
`Even and odd order` (default) | `Odd order`

Polynomial nonlinearity, specified as one of the following:

- `Even and odd order`: The Modulator can produce second-order and third-order intermodulation frequencies, in addition to a linear term.
- `Odd order`: The Modulator generates only "odd-order" intermodulation frequencies.

  The linear gain determines the linear a_1 term. The block calculates the remaining terms from the values specified in **IP3**, **1-dB gain compression power**, **Output saturation power**, and **Gain compression at saturation**. The number of constraints you specify determines the order of the model. The figure shows the graphical definition of the nonlinear Modulator parameters.



`Intercept points convention` — Intercept points convention

`Output` (default) | `Input`

Intercept points convention, specified as `Input` (input referred) or `Output` (output referred). Use this specification for the intercept points **IP2**, **IP3**, the **1-dB gain compression power**, and the **Output saturation power**.

`IP2` — Second-order intercept point

`inf dBm` (default) | scalar

Second-order intercept point, specified as a scalar in dBm, W, mW, or dBW. The default value, `inf dBm`, corresponds to an unspecified point.

Dependencies

To enable this parameter, set **Nonlinear polynomial type** to `Even and odd order`.

`IP3` — Third-order intercept point

`inf dBm` (default) | scalar

Third-order intercept point, specified as a scalar in dBm, W, mW, or dBW. The default value, `inf dBm`, corresponds to an unspecified point.

`1-dB gain compression power` — 1-dB gain compression power
`inf dBM` (default) | scalar

1-dB gain compression power, specified as a scalar in dBm, W, mW, or dBW.

Dependencies

To set this parameter, select `Odd order` in **Nonlinear polynomial type**.

`1-dB gain compression power` — 1-dB gain compression power
`inf dBM` (default) | scalar

1-dB gain compression power, specified as a scalar in dBm, W, mW, or dBW.

Dependencies

To set this parameter, select `Odd order` in **Nonlinear polynomial type**.

`Gain compression at saturation` — Gain compression at saturation
`inf dBm` (default) | scalar

Gain compression at saturation, specified as scalar in dBm, W, mW, or dBW.

When **Nonlinear polynomial type** is `Odd order`, specify the gain compression at saturation.

Dependencies

To set this parameter, first select `Odd order` in **Nonlinear polynomial type**. Then, change the default value of **Output saturation power**

IR Filter

Select **Add Image Reject filter** in the **Main** tab to see the **IR Filter** parameters tab.

`Design method` — Simulation type
`Ideal` (default) | `Butterworth` | `Chebyshev`

Simulation type. Simulates an ideal, Butterworth, or Chebyshev filter of the type specified in **Filter type** and the model specified in **Implementation**.

`Filter type` — Filter type
`Lowpass` (default) | `Highpass` | `Bandpass` | `Bandstop`

Filter. Simulates a lowpass, highpass, bandpass, or bandstop filter type of the design specified in **Design method**.

`Implementation` — Implementation

`LC Tee` | `LC Pi` | `Transfer function` | `Constant per carrier` | `Frequency Domain`

Implementation, specified as one of the following:

- `LC Tee`: Model an analog filter with an LC lumped Tee structure when the **Design method** is Butterworth or Chebyshev.
- `LC Pi`: Model an analog filter with an LC lumped Pi structure when the **Design method** is Butterworth or Chebyshev.
- `Transfer Function`: Model an analog filter using two-port S-parameters when the **Design method** is Butterworth or Chebyshev.
- `Constant per carrier`: Model a filter with either full transmission or full reflection set as constant throughout the entire envelope band around each carrier. The **Design method** is specified as ideal.
- `Frequency domain`: Model a filter using convolution with an impulse response. The **Design method** is specified as ideal. The impulse response is computed independently for each carrier frequency to capture the ideal filtering response. When a transition between full transmission and full reflection of the ideal filter occurs within the envelope band around a carrier, the frequency-domain implementation captures this transition correctly up to a frequency resolution specified in **Impulse response duration**.

  Note Due to causality, a delay of half the impulse response duration is included for both reflected and transmitted signals. This delay impairs the filter performance when the Source and Load resistances differ from the values specified in filter parameters.

By default, the **Implementation** is `Constant per carrier` for an ideal filter and `LC Tee` for Butterworth or Chebyshev.

`Passband edge frequency` — Passband edge frequency

`2 GHz` (default) | scalar

Passband edge frequency, specified as a scalar in Hz, kHz, MHz, or GHz.

Dependencies

To enable this parameter, set **Design method** to `Ideal` and **Filter type** to `Lowpass` or `Highpass`.

`Implement using filter order` — Implement using filter order

`on` (default) | `off`

Select this parameter to implement the filter order manually.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Filter order` — Filter order

3 (default) | scalar

Filter order, specified as a scalar. For a **Filter type** of `Lowpass` or `Highpass`, the filter order is the number of lumped storage elements. For a **Filter type** of `Bandpass` of `Bandstop`, the number of lumped storage elements is twice the filter order.

Note For even order Chebyshev filters, the resistance ratio $\frac{R_{load}}{R_{source}} > R_{ratio}$ for Tee network implementation and $\frac{R_{load}}{R_{source}} < \frac{1}{R_{ratio}}$ for Pi network implementation.

$$R_{ratio} = \frac{\sqrt{1+\varepsilon^2}+\varepsilon}{\sqrt{1+\varepsilon^2}-\varepsilon}$$

where:

- $\varepsilon = \sqrt{10^{(0.1R_p)}-1}$
- R_p is the passband ripple in dB.

Dependencies

To enable this parameter, select **Implement using filter order** and set **Design method** to `Butterworth` or `Chebyshev`.

`Passband frequency` — Passband frequency for lowpass and highpass filters

scalar

Passband frequency for lowpass and highpass filters, specified as a scalar in Hz, kHz, MHz, or GHz. The default value is `1 GHz` for `Lowpass` filters and `2 GHz` for `Highpass` filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Lowpass` or `Highpass`.

`Passband frequencies` — Passband frequencies for bandpass filters

`[2 3] GHz` (default) | 2-tuple vector

Passband frequencies for bandpass filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for bandstop filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandpass`.

`Passband attenuation (dB)` — Passband attenuation

`10*log10(2)` (default) | scalar

Passband attenuation, specified as a scalar in dB. For bandpass filters, this value is applied equally to both edges of the passband.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Stopband frequencies` — Stopband frequencies for bandstop filters

`[2.1 2.9]GHz` (default) | 2-tuple vector

Stopband frequencies for bandstop filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for bandpass filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandstop`.

`Stopband edge frequencies` — Stopband edge frequencies for ideal bandstop filters
`[2.1 2.9]GHz` (default) | 2-tuple vector

Stopband edge frequencies for bandstop filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for ideal bandpass filters.

Dependencies

To enable this parameter, set **Design method** to `Ideal` and **Filter type** to `Bandstop`.

`Stopband attenuation (dB)` — Stopband attenuation
`40` (default) | scalar

Stopband attenuation, specified as a scalar in dB. For bandstop filters, this value is applied equally to both edges of the stopband.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandstop`.

`Source impedance (Ohm)` — Input source resistance
`50` (default) | scalar

Input source resistance, specified as a scalar in Ohms.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Load impedance (Ohm)` — Output load resistance
`50` (default) | scalar

Output load resistance, specified as a scalar in Ohms.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Automatically estimate impulse response duration` — Automatically estimate impulse response duration
`on` (default) | `off`

Select to automatically estimate impulse response for phase noise. Clear to manually specify the impulse response duration using **Impulse response duration**.

Dependencies

To enable this parameter, set **Design method** to `Ideal` and **Implementation** to `Frequency domain`.

`Impulse response duration` — Impulse response duration

`1e-10 s` (default) | scalar

Impulse response duration used to simulate phase noise, specified as a scalar in s, ms, us or ns. You cannot specify impulse response if the amplifier is nonlinear.

Note The phase noise profile resolution in frequency is limited by the duration of the impulse response used to simulate it. Increase this duration to improve the accuracy of the phase noise profile. A warning message appears if the phase noise frequency offset resolution is too high for a given impulse response duration. The message also specifies the minimum duration suitable for the required resolution

---

Dependencies

To enable this parameter, clear **Automatically estimate impulse response duration**.

`Export` — Save filter design to a file

button

Use this button to save filter design to a file. Valid file types are `.mat` and `.txt`.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

CS Filter

Select **Add Channel Select filter** in the **Main** tab to see the **CS Filter** parameters.

`Design method` — Simulation type

`Ideal` (default) | `Butterworth` | `Chebyshev`

Simulation type. Simulates an ideal, Butterworth, or Chebyshev filter of the type specified in **Filter type** and the model specified in **Implementation**.

`Filter type` — Filter type
`Lowpass` (default) | `Highpass` | `Bandpass` | `Bandstop`

Filter. Simulates a lowpass, highpass, bandpass, or bandstop filter type of the design specified in **Design method**.

`Implementation` — Implementation
`LC Tee` | `LC Pi` | `Transfer function` | `Constant per carrier` | `Frequency Domain`

Implementation, specified as one of the following:

- `LC Tee`: Model an analog filter with an LC lumped Tee structure when the **Design method** is Butterworth or Chebyshev.
- `LC Pi`: Model an analog filter with an LC lumped Pi structure when the **Design method** is Butterworth or Chebyshev.
- `Transfer Function`: Model an analog filter using two-port S-parameters when the **Design method** is Butterworth or Chebyshev.
- `Constant per carrier`: Model a filter with either full transmission or full reflection set as constant throughout the entire envelope band around each carrier.The **Design method** is specified as ideal.
- `Frequency domain`: Model a filter using convolution with an impulse response. The **Design method** is specified as ideal. The impulse response is computed independently for each carrier frequency to capture the ideal filtering response. When a transition between full transmission and full reflection of the ideal filter occurs within the envelope band around a carrier, the frequency-domain implementation captures this transition correctly up to a frequency resolution specified in **Impulse response duration**.

  Note Due to causality, a delay of half the impulse response duration is included for both reflected and transmitted signals. This delay impairs the filter performance when the Source and Load resistances differ from the values specified in filter parameters.

By default, the **Implementation** is `Constant per carrier` for an ideal filter and `LC Tee` for Butterworth or Chebyshev.

`Passband edge frequency` — Passband edge frequency

`2 GHz` (default) | scalar

Passband edge frequency, specified as a scalar in Hz, kHz, MHz, or GHz.

Dependencies

To enable this parameter, set **Design method** to `Ideal`.

`Implement using filter order` — Implement using filter order

`on` (default) | `off`

Select this parameter to implement the filter order manually.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Filter order` — Filter order

3 (default) | scalar

Filter order, specified as a scalar. This order is the number of lumped storage elements in `lowpass` or `highpass`. In `bandpass` or `bandstop`, the number of lumped storage elements are twice the value.

Note For even order Chebyshev filters, the resistance ratio $\frac{R_{\text{load}}}{R_{\text{source}}} > R_{\text{ratio}}$ for Tee network implementation and $\frac{R_{\text{load}}}{R_{\text{source}}} < \frac{1}{R_{\text{ratio}}}$ for Pi network implementation.

$$R_{\text{ratio}} = \frac{\sqrt{1+\varepsilon^2}+\varepsilon}{\sqrt{1+\varepsilon^2}-\varepsilon}$$

where:

- $\varepsilon = \sqrt{10^{(0.1R_{\text{p}})}-1}$
- R_{p} is the passband ripple in dB.

Dependencies

To enable this parameter, select **Implement using filter order**.

`Passband frequency` — Passband frequency for lowpass and highpass filters

scalar

Passband frequency for lowpass and highpass filters, specified as a scalar in Hz, kHz, MHz, or GHz. By default, the passband frequency is `1 GHz` for `Lowpass` filters and `2 GHz` for `Highpass` filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Lowpass` or `Highpass`.

`Passband frequencies` — Passband frequencies for bandpass filters

`[2 3] GHz` (default) | 2-tuple vector

Passband frequencies for bandpass filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for bandstop filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandpass`.

`Passband attenuation (dB)` — Passband attenuation

`10*log10(2)` (default) | scalar

Passband attenuation, specified as a scalar in dB. For bandpass filters, this value is applied equally to both edges of the passband.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Stopband frequencies` — Stopband frequencies for bandstop filters

`[2.1 2.9] GHz` (default) | 2-tuple vector

Stopband frequencies for bandstop filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for bandpass filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandstop`.

`Stopband edge frequencies` — Stopband edge frequencies for ideal bandstop filters

`[2.1 2.9] GHz` (default) | 2-tuple vector

Stopband edge frequencies for bandstop filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for ideal bandpass filters.

Dependencies

To enable this parameter, set **Design method** to `Ideal` and **Filter type** to `Bandstop`.

`Stopband attenuation (dB)` — Stopband attenuation

`40` (default) | scalar

Stopband attenuation, specified as a scalar in dB. For bandstop filters, this value is applied equally to both edges of the stopband.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandstop`.

`Source impedance (Ohm)` — Input source resistance

`50` (default) | scalar

Input source resistance, specified as a scalar in Ohms.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Load impedance (Ohm)` — Output load resistance

`50` (default) | scalar

Output load resistance, specified as a scalar in Ohms.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Automatically estimate impulse response duration` — Automatically estimate impulse response duration
`on` (default) | `off`

Select to automatically estimate impulse response for phase noise. Clear to specify the impulse response duration using **Impulse response duration**.

Dependencies

To enable this parameter, set **Design method** to `Ideal` and **Implementation** to `Frequency domain`.

`Impulse response duration` — Impulse response duration
`1e-10s` (default) | scalar

Impulse response duration used to simulate phase noise, specified as a scalar in s, ms, us or ns. You cannot specify impulse response if the amplifier is nonlinear.

Note The phase noise profile resolution in frequency is limited by the duration of the impulse response used to simulate it. Increase this duration to improve the accuracy of the phase noise profile. A warning message appears if the phase noise frequency offset resolution is too high for a given impulse response duration. This message also specifies the minimum duration suitable for the required resolution

---

Dependencies

To set this parameter, first clear **Automatically estimate impulse response duration**.

`Export` — Save filter design to a file
button

Use this button to save filter design to a file. Valid file types are `.mat` and `.txt`.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

References

[1] Razavi, Behzad. *RF Microelectronics*. Upper Saddle River, NJ: Prentice Hall, 2011.

[2] Grob, Siegfried, and Lindner, Jurgen. "Polynomial Model Derivation of Nonlinear Amplifiers." *Department of Information Technology*, University of Ulm, Germany.

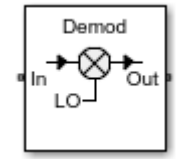
See Also

Demodulator | IQ Modulator | Mixer

Introduced in R2018a

Demodulator

Model RF to RF demodulator
Library: RF Blockset / Circuit Envelope / Systems



Description

The Demodulator block models an RF to RF demodulator.

Parameters

Main

`Source of conversion gain` — Source parameter of conversion gain
`Available power gain` (default) | `Open circuit voltage gain` | `Polynomial coefficients`

Source parameter of conversion gain, specified as one of the following:

- `Available power gain` — Relates the ratio of power of a single sideband (SSB) at the output to the input power. This calculation assumes a matched load and source termination.
- `Open circuit voltage gain` — Value of the open circuit voltage gain parameter as the linear voltage gain term of the polynomial voltage-controlled voltage source (VCVS).
- `Polynomial coefficients` — Implements a nonlinear voltage gain according to the polynomial you specify.

`Available power gain` — Ratio of power of SSB at the output to input power
`0` dB (default) | scalar in dB or a unitless ratio

Ratio of power of SSB at output to input power, specified as a scalar in dB or a unitless ratio. For a unitless ratio, select **None**.

Dependencies

To enable this parameter, set **Source of conversion gain** to `Available power gain`.

`Open circuit voltage gain` — Open circuit voltage gain

`0 dB` (default) | scalar

Open circuit voltage gain, specified as a scalar in dB.

Dependencies

To enable this parameter, set **Source of conversion gain** to `Open circuit voltage gain`.

`Polynomial coefficients` — Coefficients of polynomial specifying voltage gain

`[0 1]` (default) | vector

Polynomial coefficients, specified as a vector.

The order of the polynomial must be less than or equal to 9. The coefficients must be ordered in ascending powers. If a vector has 10 coefficients, $[a_0, a_1, a_2, \ldots a_9]$, the polynomial it represents is:

$V_{out} = a_0 + a_1 V_{in} + a_2 V_{in}^2 + \ldots + a_9 V_{in}^9$
a_1 represents the linear gain term, and higher-order terms are modeled according to [2].

For example, the vector $[a_0, a_1, a_2, a_3]$ specifies the relation $V_{out} = a_0 + a_1 V_1 + a_2 V_1^2 + a_3 V_1^3$. Trailing zeros are omitted. So, $[a_0, a_1, a_2]$ defines the same polynomial as $[a_0, a_1, a_2, 0]$.

The default value is `[0,1]`, corresponding to the linear relation $V_{out} = V_{in}$.

Dependencies

To enable this parameter, set **Source of conversion gain** to `Polynomial coefficients`.

`Local oscillator frequency` — Local oscillator (LO) frequency

`0 Hz` (default) | scalar

Local oscillator (LO) frequency, specified as a scalar in Hz, kHz, MHz, or GHz.

`Input impedance (Ohm)` — Input impedance of demodulator
`50` (default) | scalar

Input impedance of demodulator, specified as a scalar in Ohms.

`Output impedance (Ohm)` — Output impedance of demodulator
`50` (default) | scalar

Output impedance of demodulator, specified as a scalar in Ohms.

`Add Image Reject filter` — Image reject (IR) filter parameters
`off` (default) | `on`

Select to add the **IR filter** parameter tab. Clear to remove the tab.

`Add Channel Select filters` — Channel select (CS) filter parameters
`off` (default) | `on`

Select to add the **CS filter** parameter tab. Clear to remove the tab.

`Ground and hide negative terminals` — Ground and hide negative circuit terminals
`on` (default) | `off`

Select to internally ground and hide the negative terminals. Clear to expose the negative terminals. When the terminals are exposed, you can connect them to other parts of your model.

`Edit System` — Break demodulator block links and replace internal variables by appropriate values
button

Use this button to break demodulator links to the library. The internal variables are replaced by their values which are estimated using demodulator parameters. The Demodulator becomes a simple subsystem masked only to keep the icon.

Use **Edit System** to edit the internal variables without expanding the subsystem. Use **Expand System** to expand the subsystem in Simulink canvas and to edit the subsystem.

Impairments

`LO to In isolation` — Ratio of magnitude between LO voltage to leaked voltage at input port (RF)

`inf` dB (default) | scalar

Ratio of magnitude of LO voltage to leaked voltage at input port (RF), specified as a scalar in dB.

`Noise figure (dB)` — Signal-to-noise ratio (SNR) between outputs and input

`0` (default) | scalar

Single-sideband noise figure of mixer, specified as a scalar in dB.

To model noise in a circuit envelope model with a Demodulator block, you must select the **Simulate noise** check box in the Configuration block dialog box.

`Add phase noise` — Add phase noise

`off` (default) | `on`

Select this parameter to add phase noise to your demodulator system.

`Phase noise frequency offset (Hz)` — Phase noise frequency offset

`1` (default) | scalar | vector | matrix

Phase noise frequency offset, specified as a scalar, vector, or matrix with each element unit in Hz.

If you specify a matrix, each column corresponds to a non-DC carrier frequency of the CW source. The frequency offset values bind the envelope bandwidth of the simulation. For more information, see Configuration.

Dependencies

To enable this parameter, select **Add phase noise**.

`Phase noise level (dBc/Hz)` — Phase noise level

`-Inf` (default) | scalar | vector | matrix

Phase noise level, specified as a scalar, vector, or matrix with element in dBc/Hz.

If you specify a matrix, each column corresponds to a non-DC carrier frequency of the CW source. The frequency offset values bind the envelope bandwidth of the simulation. For more information, see Configuration.

Dependencies

To enable this parameter, select **Add phase noise**.

`Automatically estimate impulse response duration` — Automatically estimate impulse response duration
`on` (default) | `off`

Select to automatically estimate impulse response for phase noise. Clear to specify the impulse response duration using **Impulse response duration**.

`Impulse response duration` — Impulse response duration
`1e-10 s` (default) | scalar

Impulse response duration used to simulate phase noise, specified as a scalar in s, ms, us, or ns.

Note The phase noise profile resolution in frequency is limited by the duration of the impulse response used to simulate it. Increase this duration to improve the accuracy of the phase noise profile. A warning message appears if the phase noise frequency offset resolution is too high for a given impulse response duration. This message also specifies the minimum duration suitable for the required resolution

---

Dependencies

To set this parameter, first clear **Automatically estimate impulse response duration**.

Nonlinearity

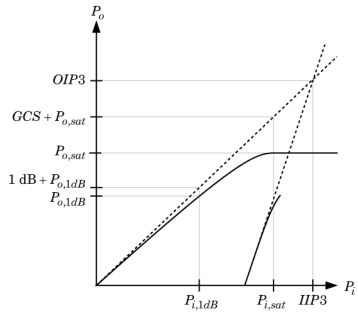
Selecting `Polynomial coefficients` for **Source of conversion gain** in the **Main** tab removes the **Nonlinearity** parameters.

`Nonlinear polynomial type` — Polynomial nonlinearity
`Even and odd order` (default) | `Odd order`

Polynomial nonlinearity, specified as one of the following:

- `Even and odd order`: The Demodulator can produce second-order and third-order intermodulation frequencies, in addition to a linear term.
- `Odd order`: The Demodulator generates only "odd-order" intermodulation frequencies.

  The linear gain determines the linear a_1 term. The block calculates the remaining terms from the values specified in **IP3**, **1-dB gain compression power**, **Output saturation power**, and **Gain compression at saturation**. The number of constraints you specify determines the order of the model. The figure shows the graphical definition of the nonlinear Demodulator parameters.



`Intercept points convention` — Intercept points convention

`Input` (default) | `Output`

Intercept points convention, specified as `Input` (input referred) or `Output` (output referred). Use this specification for the intercept points **IP2**, **IP3**, the **1-dB gain compression power**, and the **Output saturation power**.

IP2 — Second-order intercept point

`inf dBm` (default) | scalar

Second-order intercept point, specified as a scalar in dBm, W, mW, or dBW. The default value, `inf dBm`, corresponds to an unspecified point.

Dependencies

To enable this parameter, set **Nonlinear polynomial type** to `Even and odd order`.

IP3 — Third-order intercept point

`inf dBm` (default) | scalar

Third-order intercept point, specified as a scalar in dBm, W, mW, or dBW. The default value `inf dBm` corresponds to an unspecified point.

`1-dB gain compression power` — 1-dB gain compression power
`inf dBM` (default) | scalar

1-dB gain compression power, specified as a scalar in dBm, W, mW, or dBW.

Dependencies

To set this parameter, select `Odd order` in **Nonlinear polynomial type**.

`1-dB gain compression power` — 1-dB gain compression power
`inf dBM` (default) | scalar

1-dB gain compression power, specified as a scalar in dBm, W, mW, or dBW.

Dependencies

To set this parameter, select `Odd order` in **Nonlinear polynomial type**.

`Gain compression at saturation` — Gain compression at saturation
`inf dBm` (default) | scalar

Gain compression at saturation, specified as scalar in dBm, W, mW, or dBW.

When **Nonlinear polynomial type** is `Odd order`, specify the gain compression at saturation.

Dependencies

To set this parameter, first select `Odd order` in **Nonlinear polynomial type**. Then, change the default value of **Output saturation power**

IR Filter

Select **Add Image Reject filter** in the **Main** tab to see the **IR Filter** parameters tab.

`Design method` — Simulation type
`Ideal` (default) | `Butterworth` | `Chebyshev`

Simulation type. Simulates an ideal, Butterworth, or Chebyshev filter of the type specified in **Filter type** and the model specified in **Implementation**.

`Filter type` — Filter type
`Lowpass` (default) | `Highpass` | `Bandpass` | `Bandstop`

Filter. Simulates a lowpass, highpass, bandpass, or bandstop filter type of the design specified in **Design method**.

`Implementation` — Implementation
`LC Tee` | `LC Pi` | `Transfer function` | `Constant per carrier` | `Frequency Domain`

Implementation, specified as one of the following:

- `LC Tee`: Model an analog filter with an LC lumped Tee structure when the **Design method** is Butterworth or Chebyshev.
- `LC Pi`: Model an analog filter with an LC lumped Pi structure when the **Design method** is Butterworth or Chebyshev.
- `Transfer Function`: Model an analog filter using two-port S-parameters when the **Design method** is Butterworth or Chebyshev.
- `Constant per carrier`: Model a filter with either full transmission or full reflection set as constant throughout the entire envelope band around each carrier. The **Design method** is specified as ideal.
- `Frequency Domain`: Model a ideal filter using convolution with an impulse response. The **Design method** is specified as ideal. The impulse response is computed independently for each carrier frequency to capture the ideal filtering response. When a transition between full transmission and full reflection of the ideal filter occurs within the envelope band around a carrier, the frequency-domain implementation captures this transition correctly up to a frequency resolution specified in **Impulse response duration**.

  Note Due to causality, a delay of half the impulse response duration is included for both reflected and transmitted signals. This delay impairs the filter performance when the Source and Load resistances differ from the values specified in filter parameters.

By default, the **Implementation** is `Constant per carrier` for an ideal filter and `LC Tee` for Butterworth or Chebyshev.

`Passband edge frequency` — Passband edge frequency
`2 GHz` (default) | scalar

Passband edge frequency, specified as a scalar in Hz, kHz, MHz, or GHz.

Dependencies

To enable this parameter, set **Design method** to `Ideal` and **Filter type** to `Lowpass` or `Highpass`.

`Implement using filter order` — Implement using filter order

`on` (default) | `off`

Select this parameter to implement the filter order manually.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Filter order` — Filter order

3 (default) | scalar

Filter order, specified as a scalar. For a **Filter type** of `Lowpass` or `Highpass`, the filter order is the number of lumped storage elements. For a **Filter type** of `Bandpass` of `Bandstop`, the number of lumped storage elements is twice the filter order.

Note For even order Chebyshev filters, the resistance ratio $\frac{R_{load}}{R_{source}} > R_{ratio}$ for Tee network implementation and $\frac{R_{load}}{R_{source}} < \frac{1}{R_{ratio}}$ for Pi network implementation.

$$R_{ratio} = \frac{\sqrt{1+\varepsilon^2}+\varepsilon}{\sqrt{1+\varepsilon^2}-\varepsilon}$$

where:

- $\varepsilon = \sqrt{10^{(0.1R_p)}-1}$
- R_p is the passband ripple in dB.

Dependencies

To enable this parameter, select **Implement using filter order** and set **Design method** to `Butterworth` or `Chebyshev`.

`Passband frequency` — Passband frequency for lowpass and highpass filters

scalar

Passband frequency for lowpass and highpass filters, specified as a scalar in Hz, kHz, MHz, or GHz. The default passband frequency is `1 GHz` for `Lowpass` filters and `2 GHz` for `Highpass` filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Lowpass` or `Highpass`.

`Passband frequencies` — Passband frequencies for bandpass filters

`[2 3] GHz` (default) | 2-tuple vector

Passband frequencies for bandpass filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for bandstop filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandpass`.

`Passband attenuation (dB)` — Passband attenuation

`10*log10(2)` (default) | scalar

Passband attenuation, specified as a scalar in dB. For bandpass filters, this value is applied equally to both edges of the passband.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Stopband frequencies` — Stopband frequencies for bandstop filters

`[2.1 2.9]GHz` (default) | 2-tuple vector

Stopband frequencies for bandstop filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for bandpass filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandstop`.

`Stopband edge frequencies` — Stopband edge frequencies for ideal bandstop filters

`[2.1 2.9] GHz` (default) | 2-tuple vector

Stopband edge frequencies for bandstop filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for ideal bandpass filters.

Dependencies

To enable this parameter, set **Design method** to `Ideal` and **Filter type** to `Bandstop`.

`Stopband attenuation (dB)` — Stopband attenuation

`40` (default) | scalar

Stopband attenuation, specified as a scalar in dB. For bandstop filters, this value is applied equally to both edges of the stopband.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandstop`.

`Source impedance (Ohm)` — Input source resistance

`50` (default) | scalar

Input source resistance, specified as a scalar in Ohms.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Load impedance (Ohm)` — Output load resistance

`50` (default) | scalar

Output load resistance, specified as a scalar in Ohms.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Automatically estimate impulse response duration` — Automatically estimate impulse response duration

`on` (default) | `off`

Select to automatically estimate impulse response for phase noise. Clear to manually specify the impulse response duration using **Impulse response duration**.

Dependencies

To enable this parameter, set **Design method** to `Ideal` and **Implementation** to `Frequency domain`.

`Impulse response duration` — Impulse response duration

`1e-10s` (default) | scalar

Impulse response duration used to simulate phase noise, specified as a scalar in s, ms, us, or ns. You cannot specify impulse response if the amplifier is nonlinear.

Note The phase noise profile resolution in frequency is limited by the duration of the impulse response used to simulate it. Increase this duration to improve the accuracy of the phase noise profile. A warning message appears if the phase noise frequency offset resolution is too high for a given impulse response duration. The message also specifies the minimum duration suitable for the required resolution

---

Dependencies

To enable this parameter, clear **Automatically estimate impulse response duration**.

`Export` — Save filter design to a file

button

Use this button to save filter design to a file. Valid file types are `.mat` and `.txt`.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

CS Filter

Select **Add Channel Select filters** in the **Main** tab to see the **CS Filter** parameters.

`Design method` — Simulation type

`Ideal` (default) | `Butterworth` | `Chebyshev`

Simulation type. Simulates an ideal, Butterworth, or Chebyshev filter of the type specified in **Filter type** and the model specified in **Implementation**.

`Filter type` — Filter type
`Lowpass` (default) | `Highpass` | `Bandpass` | `Bandstop`

Filter. Simulates a lowpass, highpass, bandpass, or bandstop filter type of the design specified in **Design method**.

`Implementation` — Implementation
`LC Tee` | `LC Pi` | `Transfer function` | `Constant per carrier` | `Frequency Domain`

Implementation, specified as one of the following:

- `LC Tee`: Model an analog filter with an LC lumped Tee structure when the **Design method** is Butterworth or Chebyshev.
- `LC Pi`: Model an analog filter with an LC lumped Pi structure when the **Design method** is Butterworth or Chebyshev.
- `Transfer Function`: Model an analog filter using two-port S-parameters when the **Design method** is Butterworth or Chebyshev.
- `Constant per carrier`: Model a filter with either full transmission or full reflection set as constant throughout the entire envelope band around each carrier.The **Design method** is specified as ideal.
- `Frequency Domain`: Model a filter using convolution with an impulse response. The **Design method** is specified as ideal. The impulse response is computed independently for each carrier frequency to capture the ideal filtering response. When a transition between full transmission and full reflection of the ideal filter occurs within the envelope band around a carrier, the frequency-domain implementation captures this transition correctly up to a frequency resolution specified in **Impulse response duration**.

  Note Due to causality, a delay of half the impulse response duration is included for both reflected and transmitted signals. This delay impairs the filter performance when the Source and Load resistances differ from the values specified in filter parameters.

By default, the **Implementation** is `Constant per carrier` for an ideal filter and `LC Tee` for Butterworth or Chebyshev.

`Passband edge frequency` — Passband edge frequency

`2 GHz` (default) | scalar

Passband edge frequency, specified as a scalar in Hz, kHz, MHz, or GHz.

Dependencies

To enable this parameter, set **Design method** to `Ideal`.

`Implement using filter order` — Implement using filter order

`on` (default) | `off`

Select this parameter to implement the filter order manually.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Filter order` — Filter order

3 (default) | scalar

Filter order, specified as a scalar. This order is the number of lumped storage elements in `lowpass` or `highpass`. In `bandpass` or `bandstop`, the number of lumped storage elements are twice the value.

Note For even order Chebyshev filters, the resistance ratio $\frac{R_{load}}{R_{source}} > R_{ratio}$ for Tee network implementation and $\frac{R_{load}}{R_{source}} < \frac{1}{R_{ratio}}$ for Pi network implementation.

$$R_{ratio} = \frac{\sqrt{1+\varepsilon^2}+\varepsilon}{\sqrt{1+\varepsilon^2}-\varepsilon}$$

where:

- $\varepsilon = \sqrt{10^{(0.1R_p)}-1}$
- R_p is the passband ripple in dB.

Dependencies

To enable this parameter, select **Implement using filter order**.

`Passband frequency` — Passband frequency for lowpass and highpass filters

scalar

Passband frequency for lowpass and highpass filters, specified as a scalar in Hz, kHz, MHz, or GHz. By default, the passband frequency is `1 GHz` for `Lowpass` filters and `2 GHz` for `Highpass` filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Lowpass` or `Highpass`.

`Passband frequencies` — Passband frequencies for bandpass filters

`[2 3] GHz` (default) | 2-tuple vector

Passband frequencies for bandpass filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for bandstop filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandpass`.

`Passband attenuation (dB)` — Passband attenuation

`10*log10(2)` (default) | scalar

Passband attenuation, specified as a scalar in dB. For bandpass filters, this value is applied equally to both edges of the passband.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Stopband frequencies` — Stopband frequencies for bandstop filters

`[2.1 2.9] GHz` (default) | 2-tuple vector

Stopband frequencies for bandstop filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for bandpass filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandstop`.

`Stopband edge frequencies` — Stopband edge frequencies for ideal bandstop filters

`[2.1 2.9] GHz` (default) | 2-tuple vector

Stopband edge frequencies for bandstop filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for ideal bandpass filters.

Dependencies

To enable this parameter, set **Design method** to `Ideal` and **Filter type** to `Bandstop`.

`Stopband attenuation (dB)` — Stopband attenuation

`40` (default) | scalar

Stopband attenuation, specified as a scalar in dB. For bandstop filters, this value is applied equally to both edges of the stopband.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandstop`.

`Source impedance (Ohm)` — Input source resistance

`50` (default) | scalar

Input source resistance, specified as a scalar in 0hms.

`Load impedance (Ohm)` — Output load resistance

`50` (default) | scalar

Output load resistance, specified as a scalar in Ohms.

`Automatically estimate impulse response duration` — Automatically estimate impulse response duration

`on` (default) | `off`

Select to automatically estimate impulse response for phase noise. Clear to specify the impulse response duration using **Impulse response duration**.

Dependencies

To enable this parameter, set **Design method** to `Ideal` and **Implementation** to `Frequency domain`.

`Impulse response duration` — Impulse response duration

`1e-10s` (default) | scalar

Impulse response duration used to simulate phase noise, specified as a scalar in s, ms, us, or ns. You cannot specify impulse response if the amplifier is nonlinear.

Note The phase noise profile resolution in frequency is limited by the duration of the impulse response used to simulate it. Increase this duration to improve the accuracy of the phase noise profile. A warning message appears if the phase noise frequency offset resolution is too high for a given impulse response duration. The message also specifies the minimum duration suitable for the required resolution

Dependencies

To set this parameter, first clear **Automatically estimate impulse response duration**.

`Export` — Save filter design to a file

button

Use this button to save filter design to a file. Valid file types are `.mat` and `.txt`.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

References

[1] Razavi, Behzad. *RF Microelectronics*. Upper Saddle River, NJ: Prentice Hall, 2011.

[2] Grob, Siegfried, and Lindner, Jurgen. “Polynomial Model Derivation of Nonlinear Amplifiers.” *Department of Information Technology*, University of Ulm, Germany.

See Also

IQ Demodulator | Mixer | Modulator

Introduced in R2018a

IIP2 Testbench

Measure input second intercept point of system
Library: RF Blockset / Circuit Envelope / Testbenches

Description

Use the `IIP2 Testbench` to measure the input second intercept point (IIP2) of an RF device under test (DUT).

Parameters

Parameters

`Use Internal Configuration block` — Use testbench internal configuration block
on (default) | on

Select to use testbench internal configuration block. Clear this parameter to specify your own configuration block.

Note When using your own configuration block, parameters such as step size, fundamental tones, harmonic order, and simulate noise may affect the measured results.

`Simulate noise (both stimulus and DUT internal)` — Enable noise modeling in stimulus signal
on (default) | off

Select to enable noise modeling in the stimulus signal entering the DUT and inside the DUT.

Dependencies

To enable this parameter, select **Use internal Configuration block**.

`Input power amplitude (dBm)` — Input power to DUT
`-30` (default) | scalar

Input power to DUT, specified as a scalar in dBm. You can change the input power by entering the value in the text box or selecting a value using the knob. The specified input power represents the power available at the input ports of the DUT. The valid values are between -90 dBm and 60 dBm

`Input frequency (Hz)` — Carrier frequency of DUT
`2.1e9` (default) | scalar

Carrier frequency of the DUT, specified as a scalar in Hz. Input frequency must be greater than baseband bandwidth.

`Output frequency (Hz)` — Output frequency of DUT
`2.1e9` (default) | scalar

Output frequency of DUT, specified as a scalar in Hz. Output frequency must be greater than baseband bandwidth.

`Baseband bandwidth (Hz)` — Baseband bandwidth of input signal
`10e6` (default) | scalar

Baseband bandwidth of input signal, specified as a scalar in Hz. The value must be greater than zero.

`Ratio of test tone frequency to baseband bandwidth` — Position of the test tones
`1/8` (default) | scalar

Position of the test tones, specified as a scalar.

`Source resistance (Ohm)` — Source resistance to measure DUT
`50` (default) | positive finite scalar

Source resistance to measure DUT, specified as a positive finite scalar in ohms.

`Load resistance (Ohm)` — Load resistance to measure DUT
`50` (default) | positive finite scalar

Load resistance to measure DUT, specified as a positive real scalar in ohms.

`Show Response spectrum` — View response spectrum using a spectrum scope during simulation
on (default) | off

Select to view response spectrum using a spectrum scope during simulation.

Note To view the response spectrum using a Spectrum Analyzer, you need a DSP System Toolbox license.

`Ground and hide negative terminals` — Internally ground and hide RF circuit terminals
on (default) | off

Select to internally ground and hide the negative terminals. Clear to expose the negative terminals. By exposing these terminals, you can connect them to other parts of your model.

References

[1] Razavi, Behzad. *RF Microelectronics*. Upper Saddle River, NJ: Prentice Hall, 2011.

[2] Grob, Siegfried and, Jurgen Lindner. “Polynomial Model Derivation of Nonlinear Amplifiers”. *Department of Information Technology*, University of Ulm, Germany.

[3] Kundert, Ken. “Accurate and Rapid Measurements of IP2 and IP3”, *Designer's Guide Consulting,Inc.*.

See Also

IIP3 Testbench | Noise Figure Testbench | OIP2 Testbench | OIP3 Testbench | Transducer Gain Testbench

Introduced in R2018a

IIP3 Testbench

Measure input third intercept point of system
Library: RF Blockset / Circuit Envelope / Testbenches

Description

Use the `IIP3 Testbench` to measure the input third intercept point (IIP3) of an RF device under test (DUT).

Parameters

Parameters

`Use Internal Configuration block` — Use testbench internal configuration block
on (default) | off

Select to use testbench internal configuration block. Clear this parameter to specify your own configuration block.

Note When using your own configuration block, parameters such as step size, fundamental tones, harmonic order, and simulate noise may affect the measured results.

`Simulate noise (both stimulus and DUT internal)` — Enable noise modeling in stimulus signal
on (default) | off

Select to enable noise modeling in the stimulus signal entering the DUT and inside the DUT.

Dependencies

To enable this parameter, select **Use internal Configuration block**.

`Input power amplitude (dBm)` — Input power to DUT
`-30` (default) | scalar

Input power to DUT, specified as a scalar in dBm. You can change the input power by entering the value in the text box or selecting a value using the knob. The specified input power represents the power available at the input ports of the DUT. The valid values are between -90 dBm and 60 dBm

`Input frequency (Hz)` — Carrier frequency of DUT
`2.1e9` (default) | scalar

Carrier frequency of the DUT, specified as a scalar in Hz. Input frequency must be greater than baseband bandwidth.

`Output frequency (Hz)` — Output frequency of DUT
`2.1e9` (default) | scalar

Output frequency of DUT, specified as a scalar in Hz. Output frequency must be greater than baseband bandwidth.

`Baseband bandwidth (Hz)` — Baseband bandwidth of input signal
`10e6` (default) | scalar

Baseband bandwidth of input signal, specified as a scalar in Hz. The value must be greater than zero.

`Ratio of test tone frequency to baseband bandwidth` — Position of the test tones
`1/8` (default) | scalar

Position of the test tones, specified as a scalar.

`Source resistance (Ohm)` — Source resistance to measure DUT
`50` (default) | positive finite scalar

Source resistance to measure DUT, specified as a positive finite scalar in ohms.

`Load resistance (Ohm)` — Load resistance to measure DUT
`50` (default) | positive finite scalar

Load resistance to measure DUT, specified as a positive finite scalar in ohms.

`Show Response spectrum` — View response spectrum using a spectrum scope during simulation
on (default) | off

Select to view response spectrum using a spectrum scope during simulation.

Note To view the response spectrum using a Spectrum Analyzer, you need a DSP System Toolbox license.

`Ground and hide negative terminals` — Internally ground and hide RF circuit terminals
on (default) | off

Select to internally ground and hide the negative terminals. Clear to expose the negative terminals. By exposing these terminals, you can connect them to other parts of your model.

References

[1] Razavi, Behzad. *RF Microelectronics*. Upper Saddle River, NJ: Prentice Hall, 2011.

[2] Grob, Siegfried and, Jurgen Lindner. “Polynomial Model Derivation of Nonlinear Amplifiers”. *Department of Information Technology*, University of Ulm, Germany.

[3] Kundert, Ken. “Accurate and Rapid Measurements of IP2 and IP3”, *Designer's Guide Consulting,Inc.*.

See Also

IIP2 Testbench | Noise Figure Testbench | OIP2 Testbench | OIP3 Testbench | Transducer Gain Testbench

Introduced in R2018a

Noise Figure Testbench

Measures noise figure of system
Library: RF Blockset / Circuit Envelope / Testbenches

Description

Use the `Noise Figure Testbench` to measure the noise figure (NF) of an RF device under test (DUT).

Parameters

Parameters

`Use Internal Configuration block` — Use testbench internal configuration block
on (default) | off

Select to use testbench internal configuration block. Clear this parameter to specify your own configuration block.

Note When using your own configuration block, parameters such as step size, fundamental tones, harmonic order, and simulate noise may affect the measured results.

`Simulate noise (both stimulus and DUT internal)` — Enable noise modeling in stimulus signal
on (default) | off

Select to enable noise modeling in the stimulus signal entering the DUT and inside the DUT. In the NF testbench, this parameter is permanently disabled because stimulus noise is required for meaningful noise calculation.

Dependencies

To enable this parameter, select **Use internal Configuration block**.

`Input power amplitude (dBm)` — Input power to DUT
`-30` (default) | scalar

Input power to DUT, specified as a scalar in dBm. You can change the input power by entering the value in the text box or selecting a value using the knob. The specified input power represents the power available at the input ports of the DUT. The valid values are between -90 dBm and 60 dBm

`Input frequency (Hz)` — Carrier frequency of DUT
`2.1e9` (default) | scalar

Carrier frequency of the DUT, specified as a scalar in Hz. Input frequency must greater than baseband bandwidth.

`Output frequency (Hz)` — Output frequency of DUT
`2.1e9` (default) | scalar

Output frequency of DUT, specified as a scalar in Hz. Output frequency must greater than baseband bandwidth.

`Baseband bandwidth (Hz)` — Baseband bandwidth of input signal
`10e6` (default) | scalar

Baseband bandwidth of input signal, specified as a scalar in Hz. The value must be greater than zero.

`Source impedance (Ohm)` — Source impedance to measure DUT
`50` (default) | complex finite scalar

Source impedance to measure DUT, specified as a complex finite scalar in ohms. The real part of the impedance must be positive.

`Clear noise history` — Clear noise history
button (default)

Use this button to clear noise history used for internal noise measurement. When a DUT has an initial settling time duration, it is recommended to clear noise history after this duration. This improves the accuracy and convergence of noise measurement.

`Show Response spectrum` — View response spectrum using a spectrum scope during simulation
on (default) | off

Select to view response spectrum using a spectrum scope during simulation.

`Ground and hide negative terminals` — Internally ground and hide RF circuit terminals
on (default) | off

Select to internally ground and hide the negative terminals. Clear to expose the negative terminals. By exposing these terminals, you can connect them to other parts of your model.

References

[1] Razavi, Behzad. *RF Microelectronics*. Upper Saddle River, NJ: Prentice Hall, 2011.

[2] Grob, Siegfried and, Jurgen Lindner. “Polynomial Model Derivation of Nonlinear Amplifiers”. *Department of Information Technology*, University of Ulm, Germany.

See Also

IIP2 Testbench | IIP3 Testbench | OIP2 Testbench | OIP3 Testbench | Transducer Gain Testbench

Introduced in R2018a

OIP2 Testbench

Measure output second intercept point of system
Library: RF Blockset / Circuit Envelope / Testbenches

Description

Use the `OIP2 Testbench` block to measure the output second intercept point (OIP2) of an RF device under test (DUT).

Parameters

Parameters

`Use Internal Configuration block` — Use testbench internal configuration block
on (default) | off

Select to use testbench internal configuration block. Clear this parameter to specify your own configuration block.

Note When using your own configuration block, parameters such as step size, fundamental tones, harmonic order, and simulate noise may affect the measured results.

`Simulate noise (both stimulus and DUT internal)` — Enable noise modeling in stimulus signal
on (default) | off

Select to enable noise modeling in the stimulus signal entering the DUT and inside the DUT.

Dependencies

To enable this parameter, select **Use internal Configuration block**.

`Input power amplitude (dBm)` — Input power to DUT
`-30` (default) | scalar

Input power to DUT, specified as a scalar in dBm. You can change the input power by entering the value in the text box or selecting a value using the knob. The specified input power represents the power available at the input ports of the DUT. The valid values are between -90 dBm and 60 dBm

`Input frequency (Hz)` — Carrier frequency of DUT
`2.1e9` (default) | scalar

Carrier frequency of the DUT, specified as a scalar in Hz. Input frequency must be greater than baseband bandwidth.

`Output frequency (Hz)` — Output frequency of DUT
`2.1e9` (default) | scalar

Output frequency of DUT, specified as a scalar in Hz. Output frequency must be greater than baseband bandwidth.

`Baseband bandwidth (Hz)` — Baseband bandwidth of input signal
`10e6` (default) | scalar

Baseband bandwidth of input signal, specified as a scalar in Hz. The value must be greater than zero.

`Ratio of test tone frequency to baseband bandwidth` — Position of the test tones
`1/8` (default) | scalar

Position of the test tones, specified as a scalar.

`Source resistance (Ohm)` — Source resistance to measure DUT
`50` (default) | positive finite scalar

Source resistance to measure DUT, specified as a positive finite scalar in ohms.

`Load resistance (Ohm)` — Load resistance to measure DUT
`50` (default) | positive finite scalar

Load resistance to measure DUT, specified as a positive finite scalar in ohms.

`Show Response spectrum` — View response spectrum using a spectrum scope during simulation
on (default) | off

Select to view response spectrum using a spectrum scope during simulation.

Note To view the response spectrum using a Spectrum Analyzer, you need a DSP System Toolbox license.

`Ground and hide negative terminals` — Internally ground and hide RF circuit terminals
on (default) | off

Select to internally ground and hide the negative terminals. Clear to expose the negative terminals. By exposing these terminals, you can connect them to other parts of your model.

References

[1] Razavi, Behzad. *RF Microelectronics*. Upper Saddle River, NJ: Prentice Hall, 2011.

[2] Grob, Siegfried and, Jurgen Lindner. “Polynomial Model Derivation of Nonlinear Amplifiers”. *Department of Information Technology*, University of Ulm, Germany.

[3] Kundert, Ken. “Accurate and Rapid Measurements of IP2 and IP3”, *Designer's Guide Consulting,Inc.*.

See Also

IIP2 Testbench | IIP3 Testbench | Noise Figure Testbench | OIP3 Testbench | Transducer Gain Testbench

Introduced in R2018a

OIP3 Testbench

Measure output third intercept point of system
Library: RF Blockset / Circuit Envelope / Testbenches

Description

Use the `OIP3 Testbench` to measure the output third intercept point (OIP3) of an RF device under test (DUT).

Parameters

Parameters

`Use Internal Configuration block` — Use testbench internal configuration block
on (default) | off

Select to use testbench internal configuration block. Clear this parameter to specify your own configuration block.

Note When using your own configuration block, parameters such as step size, fundamental tones, harmonic order, and simulate noise may affect the measured results.

`Simulate noise (both stimulus and DUT internal)` — Enable noise modeling in stimulus signal
`on` (default) | `off`

Select to enable noise modeling in the stimulus signal entering the DUT and inside the DUT.

Dependencies

To enable this parameter, select **Use internal Configuration block**.

`Input power amplitude (dBm)` — Input power to DUT
`-30` (default) | scalar

Input power to DUT, specified as a scalar in dBm. You can change the input power by entering the value in the text box or selecting a value using the knob. The specified input power represents the power available at the input ports of the DUT. The valid values are between -90 dBm and 60 dBm

`Input frequency (Hz)` — Carrier frequency of DUT
`2.1e9` (default) | scalar

Carrier frequency of the DUT, specified as a scalar in Hz. Input frequency must be greater than baseband bandwidth.

`Output frequency (Hz)` — Output frequency of DUT
`2.1e9` (default) | scalar

Output frequency of DUT, specified as a scalar in Hz. Output frequency must be greater than baseband bandwidth.

`Baseband bandwidth (Hz)` — Baseband bandwidth of input signal
`10e6` (default) | scalar

Baseband bandwidth of input signal, specified as a scalar in Hz. The value must be greater than zero.

`Ratio of test tone frequency to baseband bandwidth` — Position of the test tones
`1/8` (default) | scalar

Position of the test tones, specified as a scalar.

`Source resistance (Ohm)` — Source resistance to measure DUT
`50` (default) | positive finite scalar

Source resistance to measure DUT, specified as a positive finite scalar in ohms.

`Load resistance (Ohm)` — Load resistance to measure DUT
`50` (default) | positive finite scalar

Load resistance to measure DUT, specified as a positive finite scalar in ohms.

`Show Response spectrum` — View response spectrum using a spectrum scope during simulation
on (default) | off

Select to view response spectrum using a spectrum scope during simulation.

Note To view the response spectrum using a Spectrum Analyzer, you need a DSP System Toolbox license.

`Ground and hide negative terminals` — Internally ground and hide RF circuit terminals
on (default) | off

Select to internally ground and hide the negative terminals. Clear to expose the negative terminals. By exposing these terminals, you can connect them to other parts of your model.

References

[1] Razavi, Behzad. *RF Microelectronics*. Upper Saddle River, NJ: Prentice Hall, 2011.

[2] Grob, Siegfried and, Jurgen Lindner. “Polynomial Model Derivation of Nonlinear Amplifiers”. *Department of Information Technology*, University of Ulm, Germany.

[3] Kundert, Ken. “Accurate and Rapid Measurements of IP2 and IP3”, *Designer's Guide Consulting,Inc.*.

See Also

IIP2 Testbench | IIP3 Testbench | Noise Figure Testbench | OIP2 Testbench | Transducer Gain Testbench

Introduced in R2018a

Transducer Gain Testbench

Measures transducer gain of system
Library: RF Blockset / Circuit Envelope / Testbenches

Description

Use the `Transducer Gain Testbench` to measure the transducer gain (G_T) of an RF device under test (DUT).

Parameters

Parameters

`Use Internal Configuration block` — Use testbench internal configuration block

on (default) | off

Select to use testbench internal configuration block. Clear this parameter to specify your own configuration block.

Note When using your own configuration block, parameters such as step size, fundamental tones, harmonic order, and simulate noise may affect the measured results.

`Simulate noise (both stimulus and DUT internal)` — Enable noise modeling in stimulus signal

on (default) | off

Select to enable noise modeling in the stimulus signal entering the DUT and inside the DUT.

Dependencies

To enable this parameter, select **Use internal Configuration block**.

`Input power amplitude (dBm)` — Input power to DUT
`-30` (default) | scalar

Input power to DUT, specified as a scalar in dBm. You can change the input power by entering the value in the text box or selecting a value using the knob. The specified input power represents the power available at the input ports of the DUT. The valid values are between -90 dBm and 60 dBm

`Input frequency (Hz)` — Carrier frequency of DUT
`2.1e9` (default) | scalar

Carrier frequency of the DUT, specified as a scalar in Hz. Input frequency must be greater than baseband bandwidth.

`Output frequency (Hz)` — Output frequency of DUT
`2.1e9` (default) | scalar

Output frequency of DUT, specified as a scalar in Hz. Output frequency must be greater than baseband bandwidth.

`Baseband bandwidth (Hz)` — Baseband bandwidth of input signal
`10e6` (default) | scalar

Baseband bandwidth of input signal, specified as a scalar in Hz. The value must be greater than zero.

`Source resistance (Ohm)` — Source resistance to measure DUT
`50` (default) | positive finite scalar

Source resistance to measure DUT, specified as a positive finite scalar in ohms.

`Load resistance (Ohm)` — Load resistance to measure DUT
`50` (default) | positive finite scalar

Load resistance to measure DUT, specified as a positive finite scalar in ohms.

`Show Response spectrum` — View response spectrum using a spectrum scope during simulation
on (default) | off

Select to view response spectrum using a spectrum scope during simulation.

Note To view the response spectrum using a Spectrum Analyzer, you need a DSP System Toolbox license.

`Ground and hide negative terminals` — Internally ground and hide RF circuit terminals
on (default) | off

Select to internally ground and hide the negative terminals. Clear to expose the negative terminals. By exposing these terminals, you can connect them to other parts of your model.

References

[1] Razavi, Behzad. *RF Microelectronics*. Upper Saddle River, NJ: Prentice Hall, 2011.

[2] Grob, Siegfried and, Jurgen Lindner. “Polynomial Model Derivation of Nonlinear Amplifiers”. *Department of Information Technology*, University of Ulm, Germany.

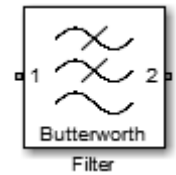
See Also

IIP2 Testbench | IIP3 Testbench | Noise Figure Testbench | OIP2 Testbench | OIP3 Testbench

Introduced in R2018a

Filter

Model RF Filter
Library: RF Blockset / Circuit Envelope / Elements



Description

The Filter block models RF filters of three designs:

- Butterworth: Butterworth filters have a magnitude response that is maximally flat in the passband and monotonic overall. This smoothness comes at the price of decreased roll-off steepness.
- Chebyshev: Chebyshev Type I filters have ripples of equal magnitude in the passband and monotonic in the stopband.
- Inverse Chebyshev: Chebyshev Type II filters have ripples of equal magnitude in the stopband and monotonic in the passband.
- Ideal: Ideal filters perfectly allow frequencies in the passband and completely reject frequencies in the stopband.

Parameters

Main

`Design method` — Simulation type
`Butterworth` (default) | `Chebyshev` | `Inverse Chebyshev` | `Ideal`

Simulation type, specified as one of the following:

- `Ideal`

Simulates an ideal filter of the type specified in **Filter type** and the model specified in **Implementation**.

- `Butterworth`

  Simulates a Butterworth filter of the type specified in **Filter type** and the model specified in **Implementation**.

- `Chebyshev`

  Simulates a Chebyshev filter of the type specified in **Filter type** and the model specified in **Implementation**.

- `Inverse Chebyshev`

  Simulates a inverse Chebyshev filter of the type specified in **Filter type** and the `Transfer function` model specified in **Implementation**.

`Filter type` — Filter type

`Lowpass` (default) | `Highpass` | `Bandpass` | `Bandstop`

Filter type, specified as one of the following:

- `Lowpass`: Simulates a lowpass filter type of the design specified in **Design method**.
- `Highpass`: Simulates a highpass filter type of the design specified in **Design method**.
- `Bandpass`: Simulates a bandpass filter type of the design specified in **Design method**.
- `Bandstop`: Simulates a bandstop filter type of the design specified in **Design method**.

`Implementation` — Implementation

`LC Tee` | `LC Pi` | `Transfer function` | `Constant per carrier` | `Frequency Domain`

Implementation, specified as one of the following:

- `LC Tee`: Model an analog filter with an LC lumped Tee structure when the **Design method** is Butterworth or Chebyshev.
- `LC Pi`: Model an analog filter with an LC lumped Pi structure when the **Design method** is Butterworth or Chebyshev.
- `Transfer Function`: Model an analog filter using two-port S-parameters when the **Design method** is Butterworth or Chebyshev.

- `Constant per carrier`: Model a filter with either full transmission or full reflection set as constant throughout the entire envelope band around each carrier.The **Design method** is specified as ideal.
- `Frequency Domain`: Model a filter using convolution with an impulse response. The **Design method** is specified as ideal. The impulse response is computed independently for each carrier frequency to capture the ideal filtering response. When a transition between full transmission and full reflection of the ideal filter occurs within the envelope band around a carrier, the frequency-domain implementation captures this transition correctly up to a frequency resolution specified in **Impulse response duration**.

By default, the **Implementation** is `Constant per carrier` for an ideal filter and `LC Tee` for Butterworth or Chebyshev.

Note Due to causality, a delay of half the impulse response duration is included for both reflected and transmitted signals. This delay will impair the filter performance when the source and load resistances differ from the values specified as filter parameters.

`Passband edge frequency` — Passband edge frequency

`1 GHz` (default) | scalar

Passband edge frequency, specified as a scalar in Hz, kHz, MHz, or GHz.

Dependencies

To enable this parameter, set **Design method** to `Ideal`.

`Implement using filter order` — Implement using filter order

`off` (default) | `on`

Select this parameter to implement the filter order manually.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Filter order` — Filter order

3 (default) | scalar

Filter order, specified as a scalar. This order is the number of lumped storage elements in `lowpass` or `highpass`. In `bandpass` or `bandstop`, the number of lumped storage elements are twice the value.

Note For even order Chebyshev filters, the resistance ratio $\frac{R_{load}}{R_{source}} > R_{ratio}$ for Tee network implementation and $\frac{R_{load}}{R_{source}} < \frac{1}{R_{ratio}}$ for Pi network implementation.

$$R_{ratio} = \frac{\sqrt{1+\varepsilon^2}+\varepsilon}{\sqrt{1+\varepsilon^2}-\varepsilon}$$

where:

- $\varepsilon = \sqrt{10^{(0.1R_p)}-1}$
- R_p is the passband ripple in dB.

Dependencies

To enable this parameter, select **Implement using filter order**.

`Passband frequency` — Passband frequency for lowpass and highpass filters

scalar

Passband frequency for lowpass and highpass filters, specified as a scalar in Hz, kHz, MHz, or GHz. The default value is `1 GHz` for `Lowpass` filters and `2 GHz` for `Highpass` filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Lowpass` or `Highpass`.

`Passband frequencies` — Passband frequencies for bandpass filters

`[2 3] GHz` (default) | 2-tuple vector

Passband frequencies for bandpass filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for bandstop filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandpass`.

`Passband attenuation (dB)` — Passband attenuation

`10*log10(2)` (default) | scalar

Passband attenuation, specified as a scalar dB. For bandpass filters, this value is applied equally to both edges of the passband.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Stopband frequencies` — Stopband frequencies for bandstop filters

`[2.1 2.9] GHz` (default) | 2-tuple vector

Stopband frequencies for bandstop filters, specified as a 2-tuple vector in Hz, kHz, MHz, or GHz. This option is not available for bandpass filters.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandstop`.

`Stopband attenuation (dB)` — Stopband attenuation

`40` (default) | scalar

Stopband attenuation, specified as a scalar dB. For bandstop filters, this value is applied equally to both edges of the stopband.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev` and **Filter type** to `Bandstop`.

`Source impedance (Ohm)` — Input source resistance

`50` (default) | scalar

Input source resistance, specified as a scalar in ohms.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Load impedance (Ohm)` — Output load resistance
`50` (default) | scalar

Output load resistance, specified as a scalar in ohms.

Dependencies

To enable this parameter, set **Design method** to `Butterworth` or `Chebyshev`.

`Ground and hide negative terminals` — Ground RF circuit terminals
`on` (default) | `off`

Select to internally ground and hide the negative terminals. Clear to expose the negative terminals. When the terminals are exposed, you can connect them to other parts of your model.

`Export` — Save filter design to a file
button (default)

Use this button to save filter design to a file. Valid file types are `.mat` and `.txt`.

Visualization

`Parameter 1` — Type of plots on y-axis
`Voltage transfer` (default) | `Phase delay` | `Group delay`

Type of plots, specified as `Voltage transfer`, `Phase delay`, or `Group delay`.

`Parameter 2` — Type of plots
`None` (default) | `Voltage transfer` | `Phase delay` | `Group delay`

Type of plots, specified as `None`, `Voltage transfer`, `Phase delay`, or `Group delay`.

`Format 1` — Scaling of y-axis
`Magnitude (decibels)` (default) | `Magnitude (linear)` | `Angle (degrees)` | `Real` | `Imaginary`

Scaling of y-axis, specified as,

- `Magnitude(decibels)`, `Magnitude(linear)` or `Angle(degrees)`, `Real`, or `Imaginary` for `Voltage transfer` parameters.
- `Magnitude(decibels)` or `Magnitude(linear)` for `Phase delay` or `Group delay` parameters.

`Format 2` — Scaling of y-axis
`Magnitude (decibels)` (default) | `Magnitude (linear)` | `Angle (degrees)` | `Real` | `Imaginary`

Scaling of y-axis, specified as,

- `Magnitude(decibels)`, `Magnitude(linear)` or `Angle(degrees)`, `Real`, or `Imaginary` for `Voltage transfer` parameters.
- `Magnitude(decibels)` or `Magnitude(linear)` for `Phase delay` or `Group delay` parameters.

`Frequency points` — Frequency points to plot on x-axis
`logspace(0,10,101) Hz` (default) | vector

Frequency points to plot on x-axis, specified as a vector with each element units in Hz, kHz, MHz, or GHz.

`X-axis scale` — X-axis scale
`Linear` (default) | `Logarithmic`

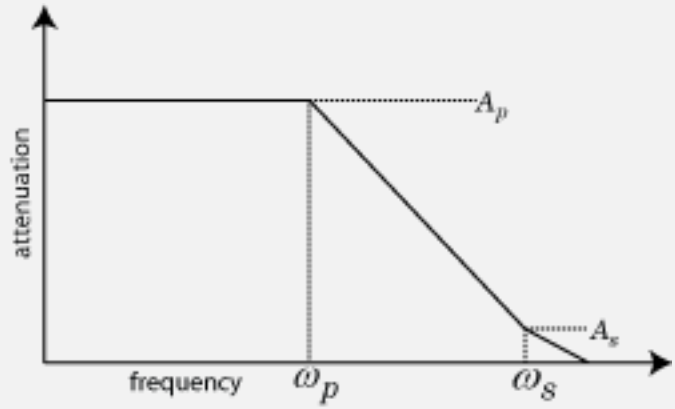
X-axis scale, specified as `Linear` or `Logarithmic`.

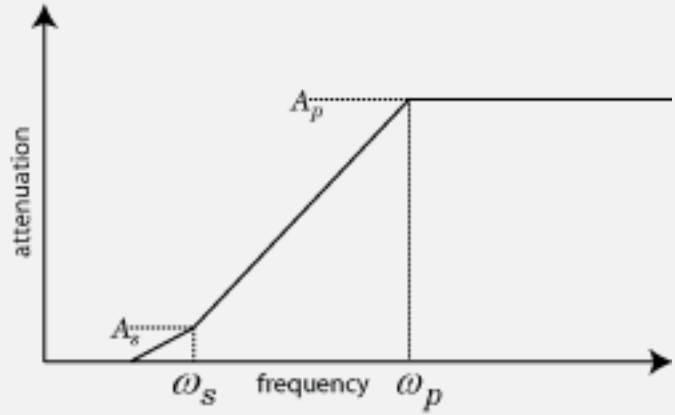
`Y-axis scale` — Y-axis scale
`Linear` (default) | `Logarithmic`

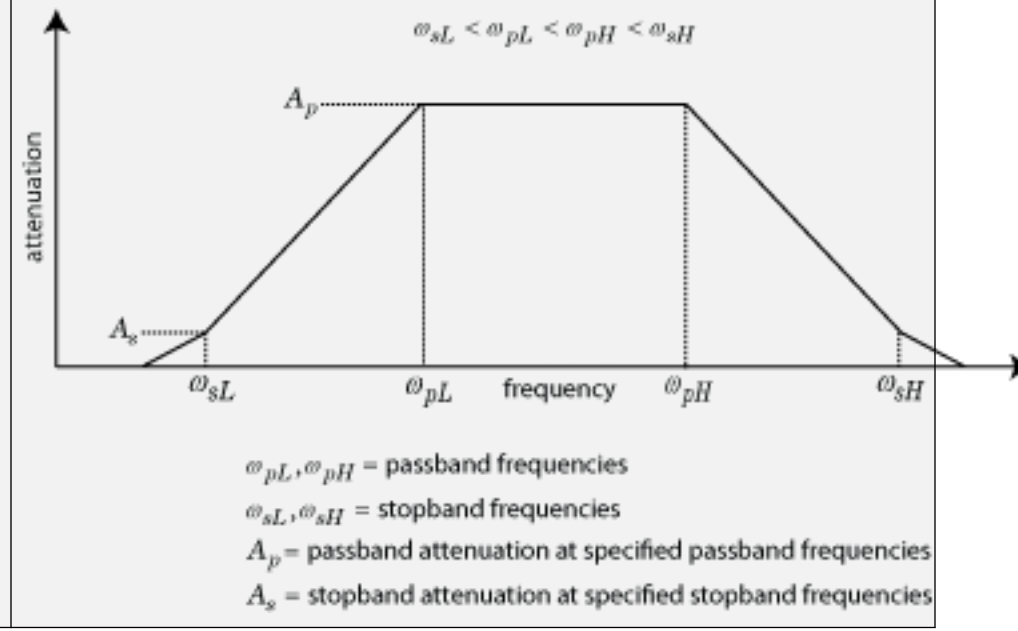
Y-axis scale, specified as `Linear` or `Logarithmic`.

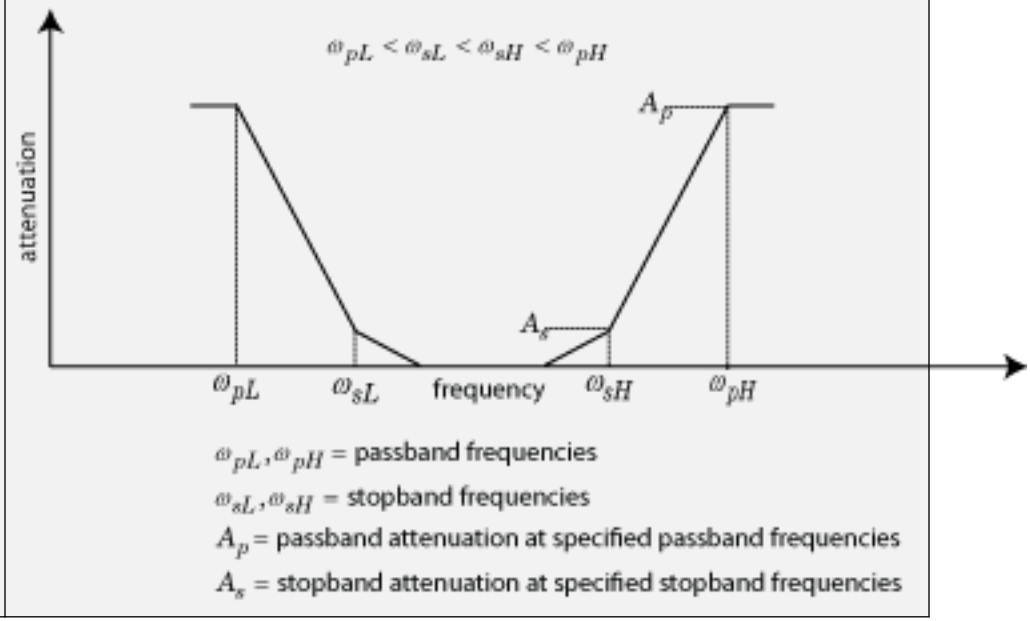
More About

Frequency Responses

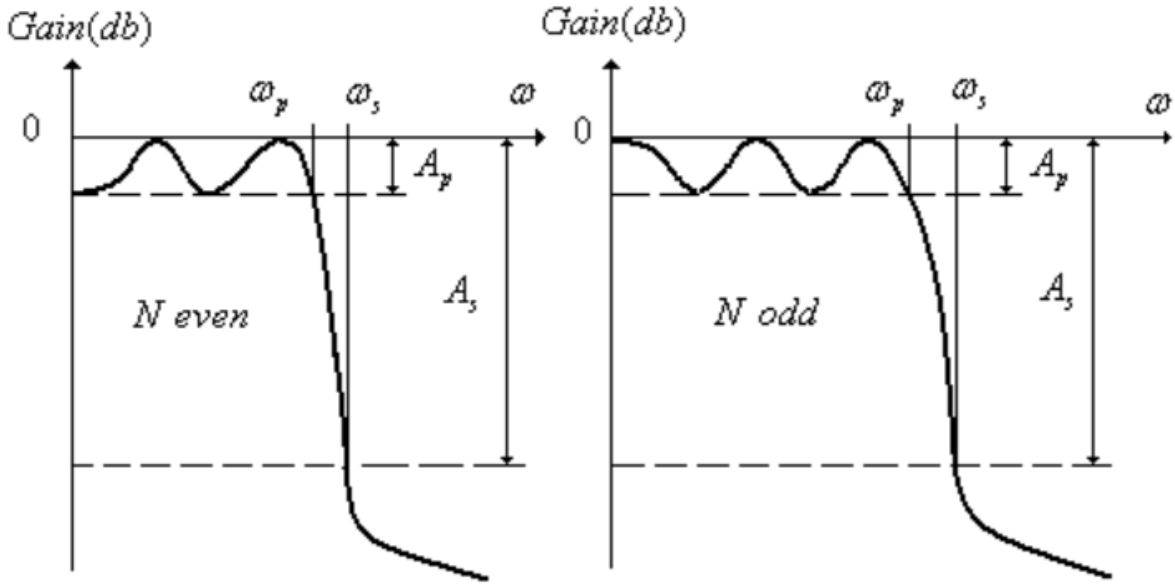
Filter Type	Frequency Response
Lowpass	 ω_p = passband frequency ω_s = stopband frequency A_p = passband attenuation @ ω_p A_s = stopband attenuation @ ω_s

Filter Type	Frequency Response
Highpass	 ω_p = passband frequency ω_s = stopband frequency A_p = passband attenuation @ ω_p A_s = stopband attenuation @ ω_s

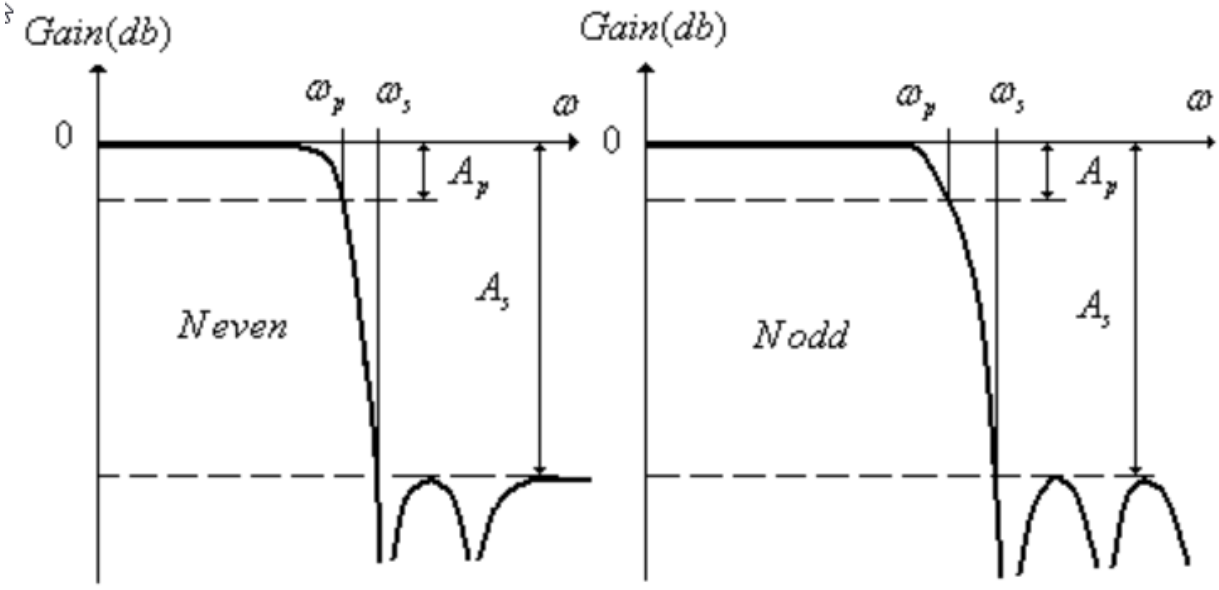
Filter Type	Frequency Response
Bandpass	

Filter Type	Frequency Response
Bandstop	

Frequency response of typical lowpass even and odd order Chebyshev filters:



Frequency response of typical inverse low pass even and odd order Chebyshev filters:



Parameters To Define Filter and Design Tips

This table shows all the parameters required to design each filter correctly:

	Low-pass	High-pass	Band-pass	Band-stop
Butterworth	Order, f_p, A_p	Order, f_p, A_p	Order, f_p, A_p	Order, f_p, A_p
	f_p, f_s, A_p, A_s	f_p, f_s, A_p, A_s	f_p, f_s, A_p, A_s	f_p, f_s, A_p, A_s
Chebyshev	Order, f_p, R_p	Order, f_p, R_p	Order, f_p, R_p	Order, f_p, R_p
	f_p, f_s, R_p, A_s	f_p, f_s, R_p, A_s	f_p, f_s, R_p, A_s	f_p, f_s, R_p, A_s
Chebyshev Inverse	Order, f_p, A_p, A_s	Order, f_p, A_p, A_s	Order, f_p, A_p, A_s	Order, f_s, A_s
	f_p, f_s, A_s, A_p	f_p, f_s, A_s, A_p	f_p, f_s, A_s, A_p	f_p, f_s, A_s, A_p
	Order, f_s, A_s	Order, f_s, A_s	Order, f_s, A_s	
Legend	passband frequency - f_p, passband attenuation/passband ripple – A_p/R_p Note: Passband ripple is parsed in as passband attenuation.		stopband frequency - f_s, stopband attenuation/ stopband ripple - A_s/R_s Note: Stopband ripple is parsed in as stopband attenuation.	

Some additional design tips:

	Low-pass	High-pass	Band-pass	Band-stop
Butterworth	Order, f_p $(3dB)$, Auxiliary (Polynomial of Numerator21)		Order, f_p, Auxiliary (Polynomial of Numerator21, Wx)	Order, f_s, Auxiliary (Wx)
Chebyshev	Order, f_p, Auxiliary (Polynomial of Numerator21)			Order, f_s, Auxiliary (Quartics of Numerator21, Wx)
Chebyshev Inverse	Order, f_s, Auxiliary (Wx)			Order, f_s

Additional Design Tips

	Low-pass	High-pass	Band-pass	Band-stop
Butterworth	Order, f_p $(3dB)$, Auxiliary (Polynomial of Numerator21)		Order, f_p, Auxiliary (Polynomial of Numerator21, Wx)	Order, f_s, Auxiliary (Wx)
Chebyshev	Order, f_p, Auxiliary (Polynomial of Numerator21)			Order, f_s, Auxiliary (Quartics of Numerator21, Wx)
Chebyshev Inverse	Order, f_s, Auxiliary (Wx)			Order, f_s

References

[1] Kendall Su, *Analog Filters, Second Edition*.

[2] Louis Weinberg, *Network Analysis and Synthesis*, Huntington, New York: Robert E. Krieger Publishing Company, 1975.

[3] Larry D. Paarmann, *Design and Analysis of Analog Filters, A Signal Processing Perspective with MATLAB Examples*, Kluwer Academic Publishers, 2001.

[4] Michael G. Ellis, SR., *Electronic Filter Analysis and Synthesis*, Norwood, MA: Artech House, 1994.

[5] Anatol I. Zverev, *Handbook of Filter Synthesis*, Hoboken, NJ: John Wiley & Sons, 2005.

See Also

Configuration | Inport | Outport

Introduced in R2016b

S-Parameter Testbench

Measure S-parameters of system
Library: RF Blockset / Circuit Envelope / Testbenches

Description

Use the S-Parameter Testbench block to measure the S-parameter data of a general RF system. The S-parameter testbench sequentially injects a stimulus signal into each port and measures the response at all ports to obtain the scattering matrix of the RF system. While the stimulus should be small signal for meaningful measurement, the testbench allows for large steady state external signals.

Parameters

Main

`Use Internal Configuration block` — Use testbench internal configuration block
on (default) | off

Select to use testbench internal configuration block. Clear this parameter to specify your own configuration block. Clearing this check box removes the **Approximate transient as small signal** option in the **Advanced** tab.

Note When using your own configuration block, parameters such as step size, fundamental tones, harmonic order, and simulate noise may affect the measured results.

`Input power amplitude (dBm)` — Input power to DUT
`-30` (default) | real-valued scalar

Input power to device under test (DUT), specified as a real valued scalar in dBm. You can change the input power by entering the value in the text box or selecting a value using

the knob. The specified input power represents the power available at the input ports of the DUT. The valid values are from `-90` through `60` dBm.

This parameter is disabled while simulation is running and when the **Approximate transient as small signal** option is checked in the **Advanced** tab.

Data Types: `double`

`Number of ports` — Number of measured ports
2 (default) | scalar integer

Number of measured ports, specified as a scalar integer limited to the range 1:128. Once you change the number of ports, new ports appear on the block.

Data Types: `double`

`Measure all S-parameters` — Measure entire S-parameters matrix
on (default) | off

Select to measure entire S-parameters matrix. This measurement is done by sequentially exciting each port and measuring all the ports for the output. When you select this box, **Save measurement result to .s2p** button appears below this parameter. In this case, the output signal S-parameters are of dimension *N*-by-*N*-by-*F*. *N* is the number of ports and *F* is the number of frequencies.

Clear this parameter to manually specify the S-parameters elements. In this case, the output signal S-parameters are of dimension *M*-by-*F*. *M* is the total number of S-parameter elements and *F* is the number of frequencies.

`Export measurement results to sNp` — Save S-parameters data to Touchstone file
button

Click to save the measured S-parameters data to a Touchstone file. This button opens a standard dialog box for browsing and choosing a file. The only file type suggested is `.s`*`N`*`p`, where *N* is the number of measured ports. This button is disabled before the first simulation takes place and while simulations are running or initializing. It captures the results of the previous simulations. The *N* value in `.sNp` corresponds to the number of ports previously measured.

Dependencies

To enable this parameter, select **Measure all S-parameters**.

`S-Parameter elements` — S-parameter elements to measure

`[1 1]` (default) | two-column matrix

S-parameter elements to measure, specified as a two-column matrix. Each row represents an S-parameter element. The first column represents the incident wave port and the second column represents the scattered wave port. For example, `[[2 1];[1 1]]` indicates a two element measurement: S21 and S11. You can choose any elements from the matrix in any order, but the elements must be unique.

Dependencies

To enable this parameter, clear **Measure all S-parameters**.

Data Types: `double`

`Workspace variable name` — Variable name to store the measurement data in MATLAB workspace

`'SparamObjOut'` (default) | character vector

Variable name to store the measurement data in MATLAB workspace, specified as a character vector. The data is stored as an RF Toolbox `sparameters` object.

Data Types: `char`

`Input frequency (Hz)` — Carrier frequency of DUT

`2.1e9` (default) | scalar

Carrier frequency of the DUT, specified as a scalar in Hz. By default, output frequency is equal to the input frequency because S-parameters are measured to quantify linear systems.

For small-signal measurements over large constant external signals, you can specify an output frequency in the **Advanced** tab when **Adjust for steady-state external signals** is selected.

Data Types: `double`

`Baseband bandwidth (Hz)` — Baseband bandwidth of input signal

`10e6` (default) | positive finite scalar

Baseband bandwidth of input signal, specified as a positive finite scalar in Hz. The measured frequencies reside on this band around the input carrier frequency. This band is by default narrower than the solver envelope bandwidth to keep simulation artifacts

outside of the measured results. The ratio of the two bands can be controlled using **Ratio of Envelope to Baseband bandwidths** in the **Advanced** tab.

Data Types: `double`

`Reference Impedance (Ohm)` — Impedance for S-parameter measurement
`50` (default) | positive finite scalar

Impedance for S-parameter measurement, specified as a positive finite scalar in ohms. All ports are measured using the same reference impedance.

Data Types: `double`

`Show S-Parameter spectrum` — View measured values of specified S-parameters over the specified frequency bandwidth
off (default) | on

Select to view measured values of specified S-parameters over the specified frequency bandwidth using a spectrum analyzer. You can view two types of curves in the plot: `Magnitude` or `Real & Imag`. The `Magnitude` plot shows the s-parameter data magnitudes is in dBm. `Real & Imag` shows both the real and imaginary parts of the S-parameters. The real and imaginary plot of the S-parameters has no units.

Note To view the S-parameters spectrum using a Spectrum Analyzer, you need a DSP System Toolbox license.

`Ground and hide negative terminals` — Internally ground and hide RF circuit terminals
on (default) | off

Select to internally ground and hide the negative terminals. Clear to expose the negative terminals. By exposing these terminals, you can connect them to other parts of your model.

Advanced

`FFT Length` — Number of FFT bins
128 (default) | scalar integer power of 2

Number of FFT bins used for measurements, specified as a scalar integer power of 2. The value must be an integer power of 2. This value controls the spectral resolution of measurements over the specified bandwidths.

`Measurement Time` — Total time duration
`12.8` µs (default) | scalar integer

This parameter is read-only.

Total time duration in which each port output is measured to get the spectral result, specified as a scalar integer. The value is the ratio of **FFT Length** over the **Baseband bandwidth** specified in the **Main** tab. It lets you know that any system response beyond this value is not included in the spectral result. This bound is an outcome of the spectral resolution limitation.

`Ratio of wait time to measurement time` — Intermission between sequential measurements
`0` (default) | scalar integer

Intermission between sequential measurements where the port excitation varies, specified as a scalar integer number of **Measurement time** instances. Similar to real measurements, the DUT cannot be reset to clear its internal states. By switching between measurements too fast, outputs that extend beyond the **Measurement time** can be collected in the next measurement. This value helps to avoid the contamination of the next measurement.

`Ratio of Envelope to Baseband bandwidths` — Ratio of internal solver envelope bandwidth to measured bandwidth
8 (default) | scalar integer power of 2

Ratio between internal solver envelope bandwidth and measured bandwidth, specified as a scalar integer power of 2.

`Adjust for steady state external signals` — Adjust for large steady state signals
off (default) | on

Select to adjust for large steady-state external signals and measure only the linear effect of the DUT for small signal stimulus injected by the testbench. This means that the DUT can contain elements that mix with large signals that are constant-in-time over the envelope at each carrier. When you select this parameter, you can specify an output frequency different from the input frequency.

Clear this if there are no signals injected to the DUT external to the testbench.

Note The DUT should not contain any external signals that are time varying over the envelope at each carrier.

`Output frequency (Hz)` — Output carrier frequency

`2.1e9` (default) | positive finite scalar

Output carrier frequency, specified as a positive finite scalar.

Dependencies

To enable this parameter, select **Adjust for steady state external signals**.

`Approximate transient as small signal` — Choose small subset of frequencies for transient small signal analysis

off (default) | on

Select this option to choose a small subset of frequencies for transient small signal analysis. Use this parameter to accelerate the measurement of S-parameters of the large nonlinear systems around a given operation point specified by large external steady-state signals.

`Use all steady-state simulation frequencies for small signal analysis` — Use all frequencies automatically chosen for full-harmonic balance nonlinear solution

on (default) | off

Select this option to use all frequencies automatically chosen for a full-harmonic balance nonlinear solution. Clear to specify the frequencies that carry the small-signal transient.

Dependencies

To enable this parameter, check **Approximate transient as small signal**.

`Small signal frequencies (Hz)` — Frequencies that carry small signal transient

`2.1e9` (default) | scalar | vector

Frequencies that carry the small signal transient, specified as a real scalar integer or vector. The specified frequencies should be contained in the entire set of simulation frequencies. If some frequencies are not contained, a warning message appears. If all frequencies are not contained, an error message appears.

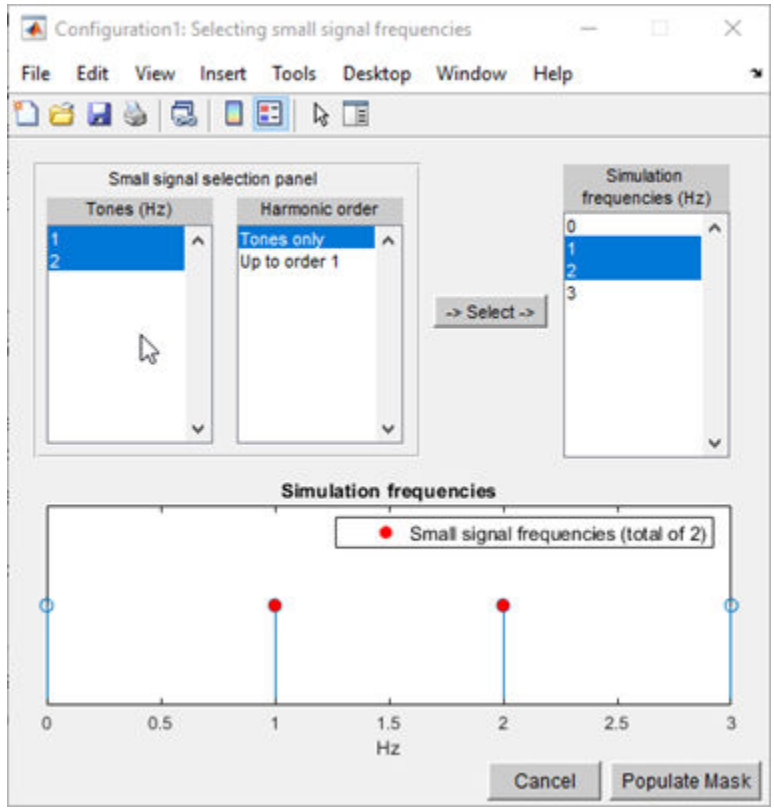
Dependencies

To enable this parameter, clear **Use all steady-state simulation frequencies for small signal analysis**.

`Populate Frequencies` — Tool to choose small signal transient frequencies

button

Open the tool to choose small signal transient frequencies to populate **Small signal frequencies**. The selected frequencies are a subset of the simulation frequencies determined from **Fundamental tones** and **Harmonic order** used in simulation. The entire set of simulation frequencies are given in the combo box on the right side of the dialog box, and the selected frequencies are highlighted. You can select by directly choosing the frequencies in the selection box, or by choosing the desired tones and harmonic order in the **Small signal selection panel** and clicking **Select**. The **Tones (Hz)** and **Harmonic order** values in the combo boxes are also populated using **Fundamental tones** and **Harmonic order** used in simulation.



Dependencies

To expose this parameter, clear **Use all steady-state simulation frequencies for small signal analysis**.

References

[1] Razavi, Behzad. *RF Microelectronics*. Upper Saddle River, NJ: Prentice Hall, 2011.

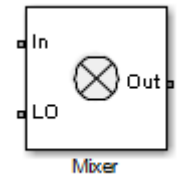
See Also

IIP2 Testbench | IIP3 Testbench | Noise Figure Testbench | OIP3 Testbench | Transducer Gain Testbench

Introduced in R2019b

IMT Mixer

Model mixer using intermodulation table (IMT)
Library: RF Blockset / Circuit Envelope / Elements



Description

Use the IMT Mixer to perform frequency translation defined in an intermodulation table (see [1], [2], [3], and [4]) for a single tone carrier mixed with a local oscillator (LO) signal. The block includes nonlinear amplification, device and phase noise, and mixer spur visualization. For a single tone carrier F_{car} nonlinearly modulated with an LO signal of frequency F_{LO}, the mixer output intermodulation products occur at frequencies:

$$F_{\text{out}}(M,N) = |M \times F_{\text{car}} \pm N \times F_{\text{LO}}|$$

where:

- F_{car} – input RF signal carrier frequency
- F_{LO} – local oscillator frequency
- M and N are nonnegative integers (0,1,..., order of nonlinearity)

For a downconverter, the desired output tone is $|F_{\text{car}} - F_{\text{LO}}|$, and for an upconverter it is $|F_{\text{car}} + F_{\text{LO}}|$. All other combinations of M and N represent the spurious intermodulation products.

Parameters

Main

`Carrier frequency (Hz)` — Carrier frequency

`1e9` (default) | scalar

Carrier frequency, specified as a scalar in hertz. When multiple carriers exist on the input connection, this carrier frequency is selected as the RF signal input. Distance between adjacent carriers must be greater than $(1 - 1e^{-8})F_c$.

Data Types: `double`

`Local oscillator frequency (Hz)` — Local oscillator (LO) frequency

`0.9e9` (default) | scalar

Local oscillator (LO) frequency, specified as a scalar in hertz.

`Reference input power (dBm)` — Reference input power

`-10` (default) | scalar

Reference input power, specified as a scalar in dBm. The expression for the normalized input signal for the specified reference input power is:

$$10^{\left(\frac{\left(-P_{\mathrm{rf}} + 20\log10\left(\frac{1}{\sqrt{R_{\mathrm{in}}}}\right) + 30\right)}{20}\right)}$$

`Nominal output power (dbm)` — Nominal LO power

`-20` (default) | scalar

Nominal output power, specified as a scalar in dBm. The expression for the normalized output signal for the specified reference input power is:

$$2 \bullet 10^{\left(\frac{\left(P_{\mathrm{if}} - 20\log10\left(\frac{1}{\sqrt{R_{\mathrm{out}}}}\right) - 30\right)}{20}\right)}$$

`Use data file` — Specify data file to use

`off` (default) | `on`

Select this parameter to specify the data file you want to use to extract the spur table. Clear to specify your own spur values. The data file may contain any combination of IMT table and colored spot noise in s2d or p2d format. See [4].

`Data file` — Data file
`samplespur1.s2d` (default) |

Data file, specified as IMT data and colored spot noise.

Dependencies

To set this parameter, first select **Use data file**.

`Input impedance (Ohm)` — Input impedance of mixer
`50` (default) | real scalar

Input impedance of mixer, specified as a real scalar.

`Output impedance (Ohm)` — Output impedance of mixer
`50` (default) | real scalar

Output impedance of mixer, specified as a real scalar.

`Ground and hide negative terminals` — Ground RF circuit terminals
`on` (default) | `off`

Select this parameter to internally ground and hide the negative terminals. To expose the negative terminals, clear this parameter. If the terminals are exposed, the input signal is not referenced to the ground.

IMT

`IMT table` — IMT spur visualization
`[99 99 99; 99 0 99; 99 99 99]` (default) | square matrix

IMT spur visualization, specified as a square matrix.

`Output signal power (dBm)` — Signal power of desired output tone
`0` (default) | scalar

Signal power of the desired output tone when plotting intermodulation products, specified as a scalar.

`Mixer type` — Mixer type
`Upconverter` (default) | `Downconverter`

Mixer type, specified as `Upconverter` or `Downconverter`.

`Plot` — Visualize IMT table values using specified signal power and mixer type
button

Visualize IMT table values using specified signal power and mixer type.

Noise

`Simulate noise` — Simulate device or phase noise
`on` (default) | `off`

Select this parameter to simulate noise as specified in block parameters or in file.

If the noise is specified in an `.s2p` file, then it is used for simulation.

`Noise type` — Noise type
`Noise figure` (default) | `Spot noise data`

Noise type, specified as `Noise figure` or `Spot noise data`.

Dependencies

To set this parameter, first select **Simulate noise**.

`Noise distribution` — Noise distribution
`White` (default) | `Piece-wise linear` | `Colored`

Noise distribution, specified as:

- `White` – Spectral density is a single nonnegative value. The power value of the noise depends on the bandwidth of the carrier, and the bandwidth depends on the time step. This is an uncorrelated noise source.
- `Piece-wise linear` – Spectral density is a vector of values $[p_i]$. For each carrier, the noise source behaves like a white uncorrelated noise. The power of the noise source is carrier dependent.
- `Colored` – Depends on both carrier and bandwidth. This is a correlated noise source.

Dependencies

To set this parameter, first select **Simulate noise**.

`Noise figure (dB)` — Noise figure

`0` (default) | scalar

Noise figure, specified as a scalar in decibels.

Dependencies

To set this parameter, first select **Simulate noise**.

`Frequencies (Hz)` — Frequency data

`0` (default) | scalar | vector

Frequency data, specified as a scalar for white noise or vector for piece-wise linear or colored noise in hertz.

Dependencies

To set this parameter, first select **Select noise** then select `Piece-wise linear` or `Colored` in **Noise distribution**.

`Minimum noise figure (dB)` — Minimum noise figure

`0` (default) | scalar | vector

Minimum noise figure, specified as a scalar or vector in decibels.

Dependencies

To set this parameter, first select `Spot noise data` in **Noise type**.

Dependencies

To set this parameter, first select **Select noise** then select `Spot noise data` in **Noise Type**.

`Optimal reflection coefficient` — Optimal reflection coefficient

`0` (default) | scalar | vector

Optimal reflection coefficient, specified as a scalar or a vector.

Dependencies

To set this parameter, first select **Select noise** then select `Spot noise data` in **Noise Type**.

`Equivalent normalized noise resistance` — Equivalent normalized noise resistance

`0` (default) | scalar | vector

Equivalent normalized noise resistance, specified as a scalar or vector.

Dependencies

To set this parameter, first select **Select noise** then select `Spot noise data` in **Noise Type**.

`Add LO phase noise` — Add phase noise

`off` (default) | `on`

Select this parameter to add phase noise to your system with a continuous wave source.

Dependencies

To set this parameter, select **Simulate noise**.

`Phase noise frequency offset (Hz)` — Phase noise frequency offset

`1` (default) | scalar | vector

Phase noise frequency offset with respect to LO signal, specified as a scalar or vector with each element unit in hertz.

The frequency offset values must be bounded by the envelope bandwidth of the simulation. For more information see Configuration.

Dependencies

To enable this parameter, first select **Simulate noise** then select **Add phase noise**.

`Phase noise level (dBc/Hz)` — Phase noise level

`-Inf` (default) | scalar | vector | matrix

Phase noise level, specified as a scalar or vector or matrix with elements in decibel per hertz.

If you specify a matrix, each column should correspond to a non-DC carrier frequency of the CW source. The frequency offset values must be bounded by the envelope bandwidth of the simulation. For more information see Configuration.

Dependencies

To enable this parameter, first select **Simulate noise** then select **Add phase noise**.

`Automatically estimate impulse response duration` — Automatically estimate impulse response duration
`on` (default) | `off`

Select this parameter to automatically calculate impulse response for frequency-dependent noises. Clear this parameter to manually specify the impulse response duration using **Impulse response duration**.

Dependencies

To enable this parameter, first select **Simulate noise** then select **Add phase noise**.

`Impulse response duration` — Impulse response duration
`1e-10s` (default) | scalar

Impulse response duration used to simulate frequency-dependent noise, specified as a scalar in seconds. The time should be an integer multiple of the step size in the configuration block, $T_{duration} = NT_{step}$.

Dependencies

To set this parameter, first clear **Automatically estimate impulse response duration**.

References

[1] https://www.mathworks.com/help/rf/examples/visualizing-mixer-spurs.html

[2] https://www.microwavejournal.com/articales/3430-the-use-of-intermodulation-tables-for-mixer-simulations

[3] https://www.electronics-notes.com/articles/radio/rf-mixer/rf-mising-basics.php

[4] https://www.mathworks.com/help/rf/examples/rf-data-objects.html

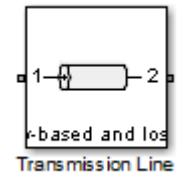
See Also

Amplifier | S-Parameters

Introduced in R2019b

Transmission Line

Model transmission line
Library: RF Blockset / Circuit Envelope / Elements



Description

Use the Transmission Line block to model delayed-based, lumped, and distributed transmission lines. Mask dialog box options change automatically to accommodate model type selection.

Parameters

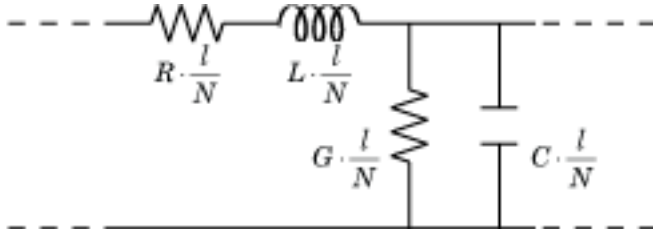
Main

`Model type` — Model type of transmission line

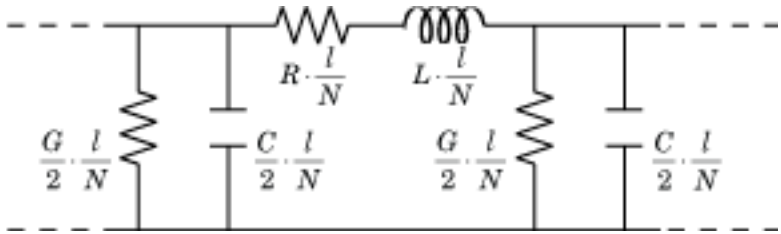
`Delay-based and lossless` (default) | `Delay-based and lossy` | `Lumped parameter L-section` | `Lumped parameter Pi-section` | `Coaxial` | `Coplanar waveguide` | `Microstrip` | `Two-wire` | `Parallel-plate` | `Equation-based` | `RLCG`

Model type of the transmission line, specified as one of the following:

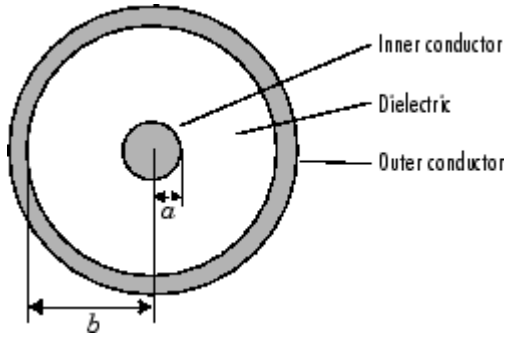
- `Delay-based and lossless` – transmission line is delay-based but no loss.
- `Delay-based and lossy` – transmission line is delay-based and there is loss.
- `Lumped parameter L-section` – transmission line as a number of RLGC L-sections.



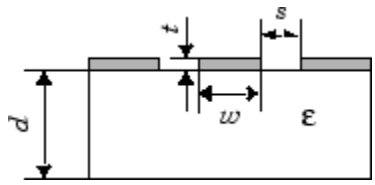
- `Lumped parameter Pi-section` – transmission line as a number of RLGC pi-sections.



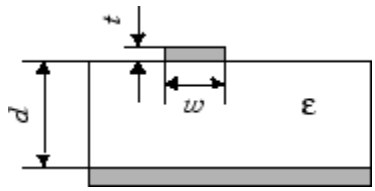
- `Coaxial` – transmission line as a coaxial transmission line. A coaxial transmission line is shown in cross-section in the following figure. Its physical characteristics include the radius of the inner conductor, *a*, and the radius of the outer conductor *b*.



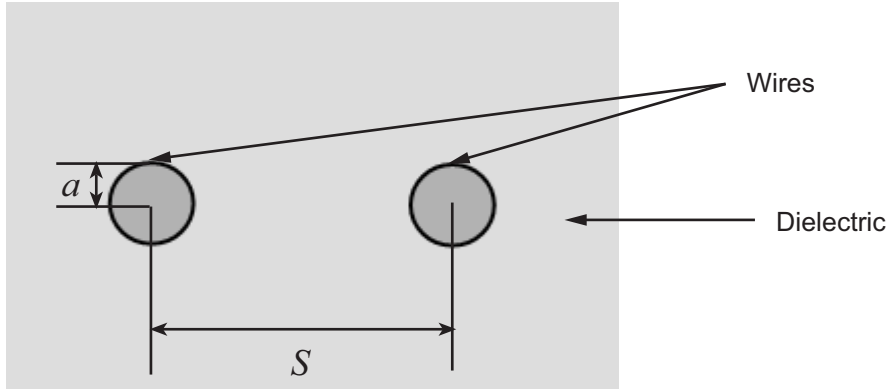
- `Coplanar waveguide` – transmission line as a coplanar waveguide. A coplanar waveguide transmission line is shown in cross-section in the following figure. Its physical characteristics include the conductor width, *w*, the conductor thickness, *t*, the slot width, *s*, the substrate height, *d*, and the relative permittivity constant, *ε*.



- `Microstrip` – transmission line as a microstrip transmission line. 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*.

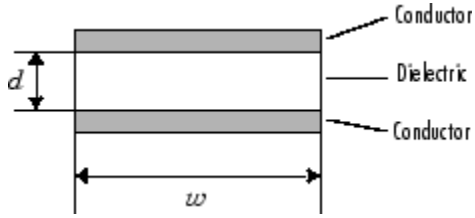


- `Two-wire` – transmission line as two-wire transmission line. A two-wire transmission line is shown in cross-section in the following figure. Its physical characteristics include the radius of the wires, *a*, the separation or physical distance between the wire centers, *S*, and the relative permittivity and permeability of the wires “References” on page 1-311. RF Blockset Equivalent Baseband software assumes the relative permittivity and permeability are uniform.



- `Parallel plate` –

transmission line as a parallel-plate transmission line. 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*. “References” on page 1-311.

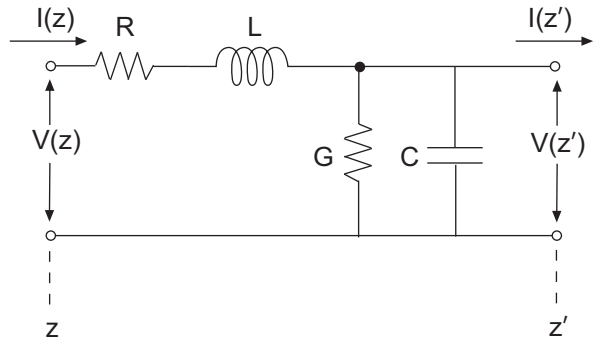


- `Equation based` –

  transmission line as an equation-based transmission line. The transmission line, which can be lossy or lossless, is treated as a two-port linear network.

- `RLCG` –

  transmission line as an RLCG transmission line. This line is described in the block dialog box in terms of its frequency-dependent resistance, inductance, capacitance, and conductance. The transmission line, which can be lossy or lossless, is treated as a two-port linear network.



`Transmission delay` — Delay in transmission line

`4.7e-9 s` (default) | real scalar

Delay in the transmission line, specified as a real scalar in `s`, milliseconds, microseconds, or nanoseconds.

To enable this parameter, choose one of the following:

- `Delay-based and lossless` in **Model type**.
- `Delay-based and lossy` in **Model type**.

`Characteristic impedance` — Impedance of transmission line

`50 Ohm` (default) | real scalar

Impedance of the transmission line, specified as a real scalar in `Ohm`, `kOhm`, `MOhm`, or `GOhm`.

Dependencies

To enable this parameter, choose one of the following:

- `Delay-based and lossless`, `Delay-based and lossy`, or `Equation-based` in **Model type**.
- `Lumped parameter L-section` or `Lumped parameter Pi-section` in **Model type** and `By characterisitc impedance and capacitance` in **Parameterization**.

`Resistance per unit length` — Resistance per unit length of transmission line

`0.3 Ohm/m` (default) | positive scalar

Resistance per unit length of the transmission line, specified as a positive scalar in `Ohm/m`, `kOhm/m`, `MOhm/m`, or `GOhm/m`.

Dependencies

To enable this parameter, choose one of the following:

- `Delay-based and lossy` or `RLCG` in **Model type**.
- `Lumped parameter L-section` or `Lumped parameter Pi-section` in **Model type** and `By characterisitc impedance and capacitance` in **Parameterization**.

`Line length` — Physical length of transmission line

`1 cm` (default) | positive scalar

Physical length of the transmission line or *l*, specified as a positive scalar in `m`, `cm`, `mm`, `um`, `in`, or `ft`.

Dependencies

To enable this parameter, choose one of the following:

- `Delay-based and lossy`, `Coaxial`, `Coplanar waveguide`, `Microstrip`, or `Two-wire`, `Parallel-plate`, `Equation-based`, or `RLCG` in **Model type**.
- `Lumped parameter L-section` or `Lumped parameter Pi-section` in **Model type** and `By characterisitc impedance and capacitance` or `By inductance and capacitance` in **Parameterization**.

`Number of segments` — Number of segments in transmission line

`10` (default) | positive scalar

Number of segments in the transmission line, specified as a positive scalar.

Dependencies

To enable this parameter, choose one of the following:

- `Delay-based and lossy` in **Model type**.
- `Lumped parameter L-section` or `Lumped parameter Pi-section` in **Model type** and `By characterisitc impedance and capacitance` or `By inductance and capacitance` in **Parameterization**.

`Parameterization` — Type of parameters to model segments in transmission line

`By characterisitc impedance and capacitance` (default) | `By inductance and capacitance`

Type of parameters to model segments in transmission line, specified as `By characterisitc impedance and capacitance` or `By inductance and capacitance`.

Dependencies

To enable this parameter, select `Lumped parameter L-section` or `Lumped parameter Pi-section` in **Model type**.

`Capacitance per unit length` — Capacitance per unit length of transmission line

`94e-12 F/m` (default) | positive scalar

Capacitance per unit length of the transmission line, specified as a positive scalar in `F/m`, `mF/m`, `uF/m`, `nF/m`, or `pF/m`.

Dependencies

To enable this parameter, choose `Lumped parameter L-section`, `Lumped parameter Pi-section`, or `RLCG` in **Model type**.

`Conductance per unit length` — Conductance per unit length of transmission line

`5e-6 S/m` (default) | positive scalar

Conductance per unit length of the transmission line, specified as a positive scalar in `S/m`, `mS/m`, `uS/m`, or `nS/m`.

Dependencies

To enable this parameter, choose `Lumped parameter L-section`, `Lumped parameter Pi-section`, or `RLCG` in **Model type**.

`Inductance per unit length` — Inductance per unit length of transmission line

`235e-9 H/m` (default) | positive scalar

Inductance per unit length of the transmission line, specified as a positive scalar in `H/m`, `mH/m`, `uH/m`, or `nH/m`.

Dependencies

To enable this parameter, choose one of the following:

- `Lumped parameter L-section`, or `Lumped parameter Pi-section` in **Model type** and `By inductance and capacitance` in **Parameterization**.
- `RLCG` in **Model type**

`Outer radius` — Outer radius of coaxial transmission line

`2.57 mm` (default) | positive scalar

Outer radius of coaxial transmission line, specified as a positive scalar in `m`, `cm`, `mm`, `um`, `in`, or `ft`.

Dependencies

To enable this parameter, choose `Coaxial` in **Model type**.

`Inner radius` — Inner radius of coaxial transmission line

`2.57 mm` (default) | positive scalar

Inner radius of coaxial transmission line, specified as a positive scalar in `m`, `cm`, `mm`, `um`, `in`, or `ft`.

Dependencies

To enable this parameter, choose `Coaxial` in **Model type**.

`Relative permeability constant` — Relative permeability of dielectric

`1` (default) | scalar

Relative permeability of the dielectric, specified as a scalar.

Dependencies

To enable this parameter, choose `Coaxial`, `Two-wire`, or `Parallel-plate` in **Model type**.

`Relative permittivity constant` — Relative permittivity of dielectric

`2.2` (default) | scalar

Relative permittivity of the dielectric, specified as a scalar.

Dependencies

To enable this parameter, choose `Coaxial`, `Coplanar waveguide`, `Microstrip`, `Two-wire`, or `Parallel-plate` in **Model type**.

`Loss Tangent of dielectric` — Loss angle tangent of dielectric

`0` (default) | scalar

Loss angle tangent of the dielectric, specified as a scalar.

Dependencies

To enable this parameter, choose `Coaxial`, `Coplanar waveguide`, `Microstrip`, `Two-wire`, or `Parallel-plate` in **Model type**.

`Conductivity of conductor` — Conductivity of conductor

`inf` (default) | scalar

Conductivity of conductor, specified as a scalar in `S/m`, `mS/m`, `uS/m`, or `nS/m`.

Dependencies

To enable this parameter, choose `Coaxial`, `Coplanar waveguide`, `Microstrip`, `Two-wire` or `Parallel-plate` in **Model type**.

`Stub mode` — Type of stub

`Not a stub` (default) | `Shunt` | `Series`

Type of stub, specified as `Not a stub`, `Shunt`, or `Series`. See "Parameter Calculations for Transmission Line with Stub" on page 1-309.

Dependencies

To enable this parameter, choose `Coaxial`, `Coplanar waveguide`, `Microstrip` `Two-wire`, `Parallel-plate`, `Equation-based`, or `RLCG` in **Model type**.

`Termination of stub` — Type of termination for stub

`Open` (default) | `Short`

Type of termination for stub, specified as `Open` or `Short`.

Dependencies

To enable this parameter, choose `Series` or `Shunt` in **Stub mode**.

`Conductor width` — Physical width of conductor

`0.6 mm` (default) | positive scalar

Physical width of the conductor, specified as a positive scalar in `m`, `cm`, `mm`, `um`, `in`, or `ft`.

Dependencies

To enable this parameter, choose `Coplanar waveguide` in **Model type**.

`Slot width` — Physical width of slot

`0.2 mm` (default) | positive scalar

Physical width of the slot, specified as a positive scalar in `m`, `cm`, `mm`, `um`, `in`, or `ft`.

Dependencies

To enable this parameter, choose `Coplanar waveguide` in **Model type**.

`Substrate height` — Thickness of dielectric on which conductor resides

`0.635 mm` (default) | positive scalar

Thickness of the dielectric on which the conductor resides, specified as a positive scalar in `m`, `cm`, `mm`, `um`, `in`, or `ft`.

Dependencies

To enable this parameter, choose `Coplanar waveguide` or `Microstrip` in **Model type**.

`Strip thickness` — Physical thickness of conductor

`5 um` (default) | positive scalar

Physical thickness of the conductor, specified as a positive scalar in `m`, `cm`, `mm`, `um`, `in`, or `ft`.

Dependencies

To enable this parameter, choose `Coplanar waveguide` or `Microstrip` in **Model type**.

`Strip Width` — Width of microstrip transmission line

`0.6 mm` (default) | positive scalar

Width of microstrip transmission line, specified as a positive scalar in `m`, `cm`, `mm`, `um`, `in`, or `ft`.

Dependencies

To enable this parameter, choose `Microstrip` in **Model type**.

`Wire radius` — Radius of conducting wires of two-wire transmission line

`0.67 mm` (default) | positive scalar

Radius of the conducting wires of the two-wire transmission line, specified as a positive scalar in `m`, `cm`, `mm`, `um`, `in`, or `ft`.

Dependencies

To enable this parameter, choose `Two-wire` in **Model type**.

`Wire separation` — Physical distance between conducting wires of two-wire transmission line

`1.62 mm` (default) | positive scalar

Physical distance between the conducting wires of the two-wire transmission line, specified as a positive scalar in `m`, `cm`, `mm`, `um`, `in`, or `ft`.

Dependencies

To enable this parameter, choose `Two-wire` in **Model type**.

`Plate width` — Width of parallel-plate transmission line

`5 mm` (default) | positive scalar

Width of the parallel-plate transmission line, specified as a positive scalar in `m`, `cm`, `mm`, `um`, `in`, or `ft`.

Dependencies

To enable this parameter, choose `Parallel-plate` in **Model type**.

`Plate separation` — Thickness of dielectric separating plates

`1 mm` (default) | positive scalar

Thickness of the dielectric separating the plates, specified as a positive scalar in `m`, `cm`, `mm`, `um`, `in`, or `ft`.

Dependencies

To enable this parameter, choose `Parallel-plate` in **Model type**.

`Phase velocity (m/s)` — Propagation velocity of a uniform plane wave on transmission line

`299792458` (default) | positive scalar

Propagation velocity of a uniform plane wave on the transmission line, specified as a positive scalar in meters per second

Dependencies

To enable this parameter, choose `Equation-based` in **Model type**.

`Loss (dB/m)` — Reduction in strength of signal as it travels over transmission line

`0` (default) | positive scalar

Reduction in strength of the signal as it travels over the transmission line, specified as a positive scalar in meters per second

Dependencies

To enable this parameter, choose `Equation-based` in **Model type**.

`Frequency` — Modeling frequencies

`1e9` (default) | positive scalar

Modeling frequencies, specified as a positive scalar or vector in `Hz`, `kHz`, `MHz`, or `GHz`.

Dependencies

To enable this parameter, choose `Equation-based` or `RLCG` in **Model type**.

`Interpolation method` — Interpolation method used to calculate values at the modeling frequencies

`Linear` (default) | `Spline` | `Cubic`

Interpolation method used to calculate the values at the modeling frequencies, specified as `Linear`, `Spline`, or `Cubic`.

Dependencies

To enable this parameter, choose `Equation-based` or `RLCG` in **Model type**.

`Ground and hide negative terminals` — Ground RF circuit terminals

`on` (default) | `off`

Select this parameter to internally ground and hide the negative terminals. To expose the negative terminals, clear this parameter. By exposing these terminals, you can connect them to other parts of your model.

By default, this option is selected.

Note Modeling options tab is activated for all transmission line options except `Delay-based and lossless`, `Delay-based and lossy`, `Lumped parameter L-section`, and `Lumped parameter pi-section`.

Modeling

`Modeling Options` — Options to model S-parameters

`Frequency domain` (default) | `Time domain (rationalfit)`

Options to model S-parameters, specified as:

- `Frequency domain` – Computes the baseband impulse response for each carrier frequency independently. This technique is based on convolution. There is an option to

specify the duration of the impulse response. For more information, see “Compare Time and Frequency Domain Simulation Options for S-parameters”.

- `Time domain (rationalfit)` – Computes the analytical rational model that approximates the whole range of the data.

For the Amplifier and S-parameters blocks, the default value is `Time domain (rationalfit)`. For the Transmission Line block, the default value is `Frequency domain`.

`Automatically estimate impulse response duration` — Calculate impulse response duration automatically
`off` (default) | `on`

Select **Automatically estimate impulse response duration** to calculate impulse response duration automatically. Clear the selection to specify impulse response duration.

Dependencies

To enable this parameter, choose `Frequency domain` in **Modeling options**.

`Impulse response duration` — Manually specify impulse response duration
`0 s` (default) | positive scalar

Manually specify impulse response duration, specified as a positive scalar in `s`, `ms`, `us`, or `ns`.

Dependencies

To enable this parameter, clear **Automatically estimate impulse response duration**.

`Fitting options` — Fitting options for rationalfit
`Share all poles` (default) | `Share poles by columns` | `Fit individually`

Fitting options for rationalfit, specified as `Share all poles`, `Share poles by columns`, or `Fit individually`.

For the Amplifier block, the default value is `Fit individually`. For the S-parameters block and Transmission Line block, the default value is `Share all poles`.

Dependencies

To enable this parameter, choose `Time domain (rationalfit)` in **Modeling options**.

`Relative error desired (dB)` — Relative error acceptable in rationalfit output
`-40` (default) | real scalar

Relative error acceptable in rationalfit output, specified as a real scalar in decibels.

Dependencies

To enable this parameter, choose `Time domain (rationalfit)` in **Modeling options**.

`Rational fitting results` — Values of rationalfit calculations
read-only (default)

Shows values of **Number of independent fits**, **Number of required poles**, and **Relative error achieved (dB)**.

When modeling using `Time domain`, the **Plot** in `Visualization` tab plots the data defined in `Data Source` and the values in the `rationalfit` function.

Dependencies

To enable this parameter, choose `Time domain (rationalfit)` in **Modeling options**.

More About

Equations for ABCD Parameter Calculations

The following auxiliary equations are used for ABCD-parameter calculations.

$$Z_0 = \sqrt{\frac{R + j\omega L}{G + j\omega C}}$$

$$k = k_r + jk_i = \sqrt{(R + j\omega L)(G + j\omega C)}$$

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 **Relative permittivity constant** parameter value.
  - $\tan\delta$ is the **Loss tangent of dielectric** parameter 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 internal modeling frequencies.
- Z_0 is the specified characteristic impedance.
- k is a vector whose elements correspond to the elements of the input vector, `freq`. The block calculates k from the specified parameters 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

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

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

Parameter Calculations for Distributed Transmission Line

When modeling distributed transmission lines, the block first calculates ABCD-parameters at a set of internal frequencies. The ABCD-parameters are converted S-parameters for simulation.

The block calculates the ABCD-parameters from the physical length of the transmission line, d, and the complex propagation constant, k, using the following set of equations:

$$A = \frac{e^{kd} + e^{-kd}}{2}$$

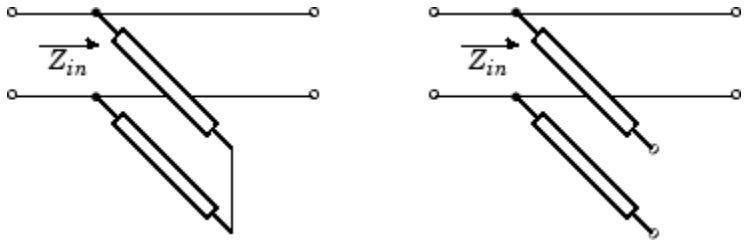
$$B = \frac{Z_0 * \left(e^{kd} - e^{-kd}\right)}{2}$$

$$C = \frac{e^{kd} - e^{-kd}}{2 * Z_0}$$

$$D = \frac{e^{kd} + e^{-kd}}{2}$$

Parameter Calculations for Transmission Line with Stub

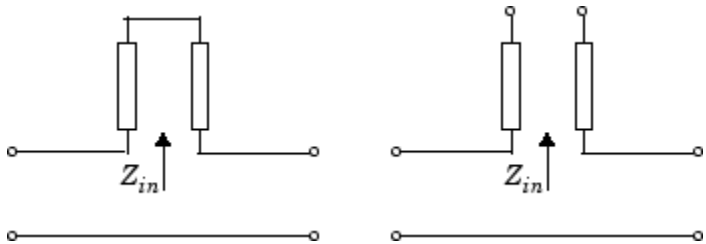
When you set the **Stub mode** parameter in the mask dialog box to `Shunt`, the two-port network consists of a transmission line in series with a stub. You can terminate the stub with a short circuit or an open circuit as shown in the following figure.



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 **Stub mode** parameter in the mask dialog box to `Series`, the two-port network comprises a series transmission line. You can terminate this line with either a short circuit or an open circuit as shown here.



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

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

Tips

- In general, blocks that model delay effects rely on signal history. You can minimize numerical error that occur due to a lack of signal history at the start of a simulation. To do so, in the Configuration Parameters dialog box Solver pane you can specify an **Initial step size**. For models with delay-based Transmission Line blocks, use an initial step size that is less than the value of the **Delay** parameter.

References

[1] Sussman-Fort, S. E., and J. C. Hantgan. “SPICE Implementation of Lossy Transmission Line and Schottky Diode Models.” *IEEE Transactions on Microwave Theory and Techniques*.Vol. 36, No.1, January 1988.

[2] Pozar, David M. *Microwave Engineering*. Hoboken, NJ: John Wiley & Sons, Inc., 2005.

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

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

[5] True, Kenneth M. “Data Transmission Lines and Their Characteristics.” *National Semiconductor Application Note 806*, April 1992.

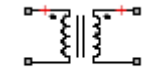
See Also

Amplifier | S-Parameters

Introduced in R2012a

Mutual Inductor

Model two coupled inductors for circuit envelope analysis
Library: RF Blockset / Circuit Envelope / Elements



Description

The Mutual Inductor block models an inductor within the RF Blockset circuit envelope simulation environment. For an introduction to RF simulation, see the example, “Simulate High Frequency Components”.

The block implements the relations

$$v_1(t) = L_1\frac{d}{dt}[i_1(t)] + M\frac{d}{dt}[i_2(t)]$$

$$v_2(t) = M\frac{d}{dt}[i_1(t)] + L_2\frac{d}{dt}[i_2(t)]$$

$$M = K\sqrt{L_1L_2}$$

where:

- L_1 and L_2 represent inductances.
- M represents a mutual inductance with coefficient of coupling K.
- $v_1(t)$ and $v_2(t)$ represent the voltage across the terminals of the inductors at time t.
- $i_1(t)$ and $i_2(t)$ represent the current through the inductors at time t. The block uses standard dot notation to indicate the direction of positive current flow relative to a positive voltage.

RF Blockset current and voltage signals consist of in-phase (I_k) and quadrature (Q_k) components at each frequency f_k specified in the Configuration block:

$$i(t) = \sum_{\{f_k\}} \left(i_{I_k}(t) + j \cdot i_{Q_k}(t)\right) e^{j(2\pi f_k)t}$$

$$v(t) = \sum_{\{f_k\}} \left(v_{I_k}(t) + j \cdot v_{Q_k}(t)\right) e^{j(2\pi f_k)t}$$

Parameters

`Inductance L1` — Inductance of first inductor
`1e-6` `H` (default) | | positive scalar

Inductance of the first inductor, specified as a positive scalar in `H`, `nH`, `µH`, `mH`.

`Inductance L2` — Inductance of second inductor
`1e-6` `H` (default) | | positive scalar

Inductance of the second inductor, specified as a positive scalar in `H`, `nH`, `µH`, `mH`.

`Coefficient of coupling` — Coefficient of coupling for mutual inductance
`0.9` (default) | | scalar

Coefficient of coupling for the mutual inductance of the two inductors, specified as a scalar value from 0 through 1.

See Also

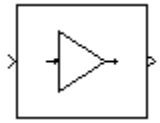
Inductor | Three-Winding Transformer

Introduced in R2012a

Equivalent Baseband Blocks — Alphabetical List

Amplifier (Idealized Baseband)

Complex baseband model of amplifier with noise



Library

Mathematical

Description

The Amplifier block generates a complex baseband model of an amplifier with thermal noise. It provides six methods for modeling nonlinearity and three ways to specify noise.

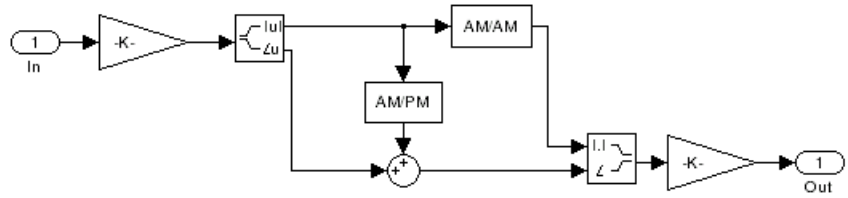
Note This block assumes a nominal impedance of 1 ohm.

Modeling Nonlinearity

Use the **Method** parameter in the block dialog box to specify the method for modeling amplifier nonlinearity. The options for the **Method** parameter are

- `Linear`
- `Cubic polynomial`
- `Hyperbolic tangent`
- `Saleh model`
- `Ghorbani model`
- `Rapp model`

The linear method is implemented by a Gain block. The other nonlinear methods are implemented by subsystems underneath the block's mask. Each subsystem has the same basic structure, as shown in the following figure.



Application of Nonlinearity

All five subsystems for the nonlinear **Method** options apply a memoryless nonlinearity to the complex baseband input signal. Each one

1. Multiplies the signal by a gain factor.
2. Splits the complex signal into its magnitude and angle components.
3. Applies an AM/AM conversion to the magnitude of the signal, according to the selected nonlinearity method, to produce the magnitude of the output signal.
4. Applies an AM/PM conversion to the phase of the signal, according to the selected nonlinearity method, and adds the result to the angle of the signal to produce the angle of the output signal.
5. Combines the new magnitude and angle components into a complex signal and multiplies the result by a gain factor, which is controlled by the **Linear gain** parameter.

AM/AM and AM/PM Conversions

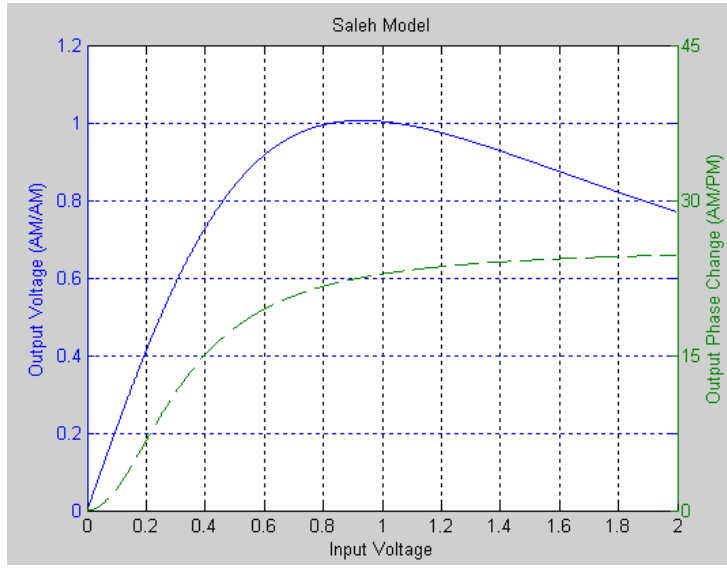
The subsystems for the nonlinear methods implement the AM/AM and AM/PM conversions differently, according to the nonlinearity method you specify. To see exactly how the Amplifier block implements the conversions for a specific method, you can view the AM/AM and AM/PM subsystems that implement these conversions as follows:

1. Right-click the Amplifier block.
2. Select **Look under mask** in the pop-up menu. This displays the block's configuration underneath the mask. The block contains five subsystems corresponding to the five nonlinearity methods.

3. Double-click the subsystem for the method in which you are interested. A subsystem displays similar to the one shown in the preceding figure.
4. Double-click one of the subsystems labeled AM/AM or AM/PM to view how the block implements the conversions.

The following figure shows, for the Saleh method, plots of

- Output voltage against input voltage for the AM/AM conversion
- Output phase against input voltage for the AM/PM conversion



Model Parameters and Characteristics of Nonlinearity Modeling Methods

The following sections discuss how the parameters specific to the following nonlinear amplifier models affect the AM/AM and AM/PM characteristics of the Amplifier block:

- “Cubic Polynomial Model” on page 2-5
- “Hyperbolic Tangent Model” on page 2-6
- “Saleh Model” on page 2-6

- “Ghorbani Model” on page 2-7
- “Rapp Model” on page 2-8

Note The Amplifier block also enables you to model a linear amplifier.

Cubic Polynomial Model

When you select `Cubic polynomial` for the nonlinearity modeling **Method** parameter, the Amplifier block models the AM/AM nonlinearity by:

1. Using the third-order input intercept point **IIP3 (dBm)** parameter to compute the factor, *f*, that scales the input signal before the Amplifier block applies the nonlinearity:

   $$f = \sqrt{\frac{3}{IIP3\,(Watts)}} = \sqrt{\frac{3}{10^{(\textbf{IIP3 (dBm)} - 30)/10}}}$$

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 Amplifier block uses the **AM/PM conversion (degrees per dB)** parameter, which specifies the linear phase change, to add the AM/PM nonlinearity within the power limits specified by the **Lower input power limit for AM/PM conversion (dBm)** parameter and the **Upper input power limit for AM/PM conversion (dBm)** parameter. Outside those limits, the phase change is constant at the values corresponding to the lower and upper input power limits, which are zero and

(AM/PM conversion) · (upper input power limit − lower input power limit),

respectively.

The **Linear gain (dB)** parameter scales the output signal.

Hyperbolic Tangent Model

When you select `Hyperbolic tangent` for the nonlinearity modeling **Method** parameter, the Amplifier block computes and adds the AM/AM nonlinearity by:

1. Using the third-order input intercept point **IIP3 (dBm)** parameter to compute the factor, *f*, that scales the input signal before the Amplifier block applies the nonlinearity:

   $$f = \sqrt{\frac{3}{IIP3\,(Watts)}} = \sqrt{\frac{3}{10^{(\textbf{IIP3 (dBm)} - 30)/10}}}$$

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:

   $$F_{AM/AM}(u) = \tanh u$$

   where *u* is the magnitude of the scaled input signal, which is a unitless normalized input voltage.

The Amplifier block uses the **AM/PM conversion (degrees per dB)** parameter, which specifies the linear phase change, to add the AM/PM nonlinearity within the power limits specified by the **Lower input power limit for AM/PM conversion (dBm)** parameter and the **Upper input power limit for AM/PM conversion (dBm)** parameter. Outside those limits, the phase change is constant at the values corresponding to the lower and upper input power limits, which are zero and

(AM/PM conversion) · (upper input power limit − lower input power limit),

respectively.

The **Linear gain (dB)** parameter scales the output signal.

Saleh Model

When you select `Saleh model` for the nonlinearity modeling **Method** parameter, the **Input scaling (dB)** parameter scales the input signal before the nonlinearity is applied. The block multiplies the input signal by the parameter value, converted from decibels to linear units. If you set the parameter to be the inverse of the input signal amplitude, the scaled signal has amplitude normalized to 1.

The AM/AM parameters, alpha and beta, are used to compute the amplitude gain for an input signal using the following function

$$F_{AM/AM}(u) = \frac{\alpha u}{1 + \beta u^2}$$

where u is the magnitude of the scaled signal.

The AM/PM parameters, alpha and beta, are used to compute the phase change for an input signal using the following function

$$F_{AM/PM}(u) = \frac{\alpha u^2}{1 + \beta u^2}$$

where u is the magnitude of the input signal. Note that the AM/AM and AM/PM parameters, although similarly named alpha and beta, are distinct.

The **Output scaling (dB)** parameter scales the output signal similarly.

Ghorbani Model

When you select `Ghorbani model` for the nonlinearity modeling **Method** parameter, the **Input scaling (dB)** parameter scales the input signal before the nonlinearity is applied. The block multiplies the input signal by the parameter value, converted from decibels to linear units. If you set the parameter to be the inverse of the input signal amplitude, the scaled signal has amplitude normalized to 1.

The AM/AM parameters, $[x_1\ x_2\ x_3\ x_4]$, are used to compute the amplitude gain for an input signal using the following function

$$F_{AM/AM}(u) = \frac{x_1 u^{x_2}}{1 + x_3 u^{x_2}} + x_4 u$$

where u is the magnitude of the scaled signal.

The AM/PM parameters, $[y_1\ y_2\ y_3\ y_4]$, are used to compute the phase change for an input signal using the following function

$$F_{AM/PM}(u) = \frac{y_1 u^{y_2}}{1 + y_3 u^{y_2}} + y_4 u$$

where u is the magnitude of the scaled signal.

The **Output scaling (dB)** parameter scales the output signal similarly.

Rapp Model

When you select `Rapp model` for the nonlinearity modeling **Method** parameter, the **Smoothness factor** and **Output saturation level** parameters are used to compute the amplitude gain for an input signal by the following function

$$F_{AM/AM}(u) = \frac{u}{\left(1 + \left(\frac{u}{O_{sat}}\right)^{2S}\right)^{\frac{1}{2S}}}$$

where u is the magnitude of the scaled signal, S is the **Smoothness factor** and O_{sat} is the **Output saturation level**.

The Rapp model does not apply a phase change to the input signal.

The **Output saturation level** parameter limits the output signal level. The **Smoothness factor** parameter controls the transition for the amplitude gain as the input amplitude approaches saturation. The smaller the smoothness factor, the smoother the curve.

Thermal Noise Simulation

You can specify the amount of thermal noise in three ways, according to the **Specification method** parameter you select.

- `Noise temperature` — Specifies the noise in kelvin.
- `Noise factor` — Specifies the noise by the following equation:

  $$\text{Noise factor} = 1 + \frac{\text{Noise temperature}}{290}$$

- `Noise figure` — Specifies the noise in decibels relative to a noise temperature of 290 kelvin. In terms of noise factor,

  $$\text{Noise figure} = 10\log(\text{Noise factor})$$

Note Some RF blocks require the sample time to perform baseband modeling calculations. To ensure the accuracy of these calculations, the Input Port block, as well

as the mathematical RF blocks, compare the input sample time to the sample time you provide in the mask. If they do not match, or if the input sample time is missing because the blocks are not connected, an error message appears.

Parameters

The parameters displayed in the dialog box vary for different methods of modeling nonlinearity. Only some of these parameters are visible in the dialog box at any one time.

You can change tunable parameters while the model is running.

Method

Method used to model the nonlinearity. The choices are `Linear`, `Cubic polynomial`, `Hyperbolic tangent`, `Saleh model`, `Ghorbani model`, `Rapp model`. Tunable.

Linear gain (dB)

Scalar specifying the linear gain for the output function. This field becomes visible if you select `Linear`, `Cubic polynomial`, `Hyperbolic tangent`, or `Rapp model` as the **Method** parameter. Tunable.

IIP3 (dBm)

Input power intercept point as a scalar value. This field becomes visible if you select `Cubic polynomial` or `Hyperbolic tangent` as the **Method** parameter. For both of these methods, the nominal impedance is 1 ohm. Tunable.

AM/PM conversion (degrees per dB)

Scalar specifying the AM/PM conversion in degrees per decibel. This field becomes visible if you select `Cubic polynomial` or `Hyperbolic tangent` as the **Method** parameter. Tunable.

Lower input power limit for AM/PM conversion (dBm)

Scalar specifying the minimum input power for which AM/PM conversion scales linearly with input power value. Below this value, the phase shift resulting from AM/PM conversion is zero. This field becomes visible if you select `Cubic polynomial` or `Hyperbolic tangent` as the **Method** parameter. Tunable.

Upper input power limit for AM/PM conversion (dBm)

Scalar specifying the maximum input power for which AM/PM conversion scales linearly with input power value. Above this value, the phase shift resulting from AM/PM conversion is constant. The value of this maximum shift is given by:

(AM/PM conversion) · (upper input power limit − lower input power limit),

This field becomes visible if you select `Cubic polynomial` or `Hyperbolic tangent` as the **Method** parameter. Tunable.

Input scaling (dB)

Number that scales the input signal level. This field becomes visible if you select `Saleh model` or `Ghorbani model` as the **Method** parameter. Tunable.

Output scaling (dB)

Number that scales the output signal level. This field becomes visible if you select `Saleh model` or `Ghorbani model` as the **Method** parameter. Tunable.

AM/AM parameters [alpha beta]

Vector specifying the AM/AM parameters. This field becomes visible if you select `Saleh model` as the **Method** parameter. Tunable.

AM/PM parameters [alpha beta]

Vector specifying the AM/PM parameters. This field becomes visible if you select `Saleh model` as the **Method** parameter. Tunable.

AM/AM parameters [x1 x2 x3 x4]

Vector specifying the AM/AM parameters. This field becomes visible if you select `Ghorbani model` as the **Method** parameter. Tunable.

AM/PM parameters [y1 y2 y3 y4]

Vector specifying the AM/PM parameters. This field becomes visible if you select `Ghorbani model` as the **Method** parameter. Tunable.

Smoothness factor

Scalar specifying the smoothness factor. This field becomes visible if you select `Rapp model` as the **Method** parameter. Tunable.

Output saturation level

Scalar specifying the output saturation level. This field becomes visible if you select `Rapp model` as the **Method** parameter. Tunable.

Specification method

The method by which you specify the amount of noise. The choices are `Noise temperature`, `Noise figure`, and `Noise factor`. Tunable.

Noise temperature (K)

Scalar specifying the amount of noise. This field becomes visible if you select `Noise temperature` as the **Specification method** parameter. Tunable.

Noise figure (dB)

Scalar specifying the amount of noise relative to a noise temperature of 290 kelvin. A `Noise figure` setting of 0 decibels indicates a noiseless system. This field becomes visible if you select `Noise figure` as the **Specification method** parameter. Tunable.

Noise factor

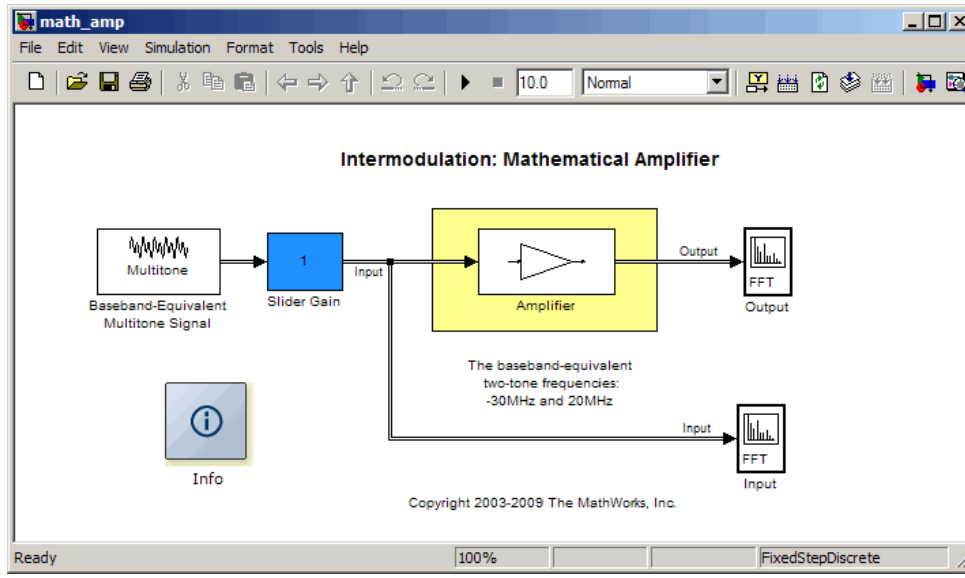
Scalar specifying the amount of noise relative to a noise temperature of 290 kelvin. This field becomes visible if you select `Noise factor` as the **Specification method** parameter. Tunable.

Initial seed

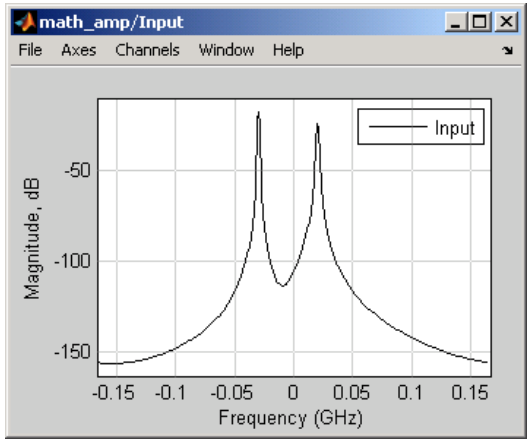
Nonnegative integer specifying the initial seed for the random number generator the block uses to generate noise.

Examples

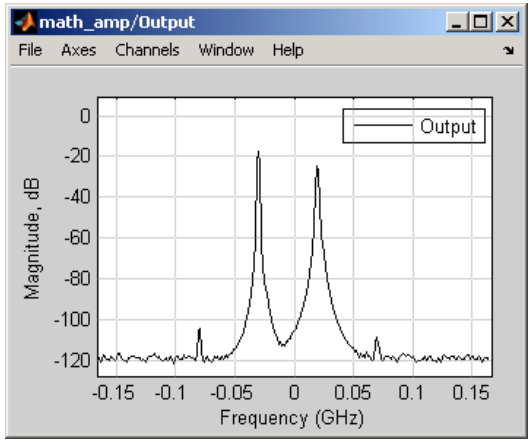
You can see the effect of the Amplifier block in the example Intermodulation: Mathematical Amplifier.



This example uses a baseband-equivalent multitone signal as input to the Amplifier block. A Simulink Slider Gain block enables you to vary the gain from 1 to 10. The following figure shows the input signal with gain set to the default 1.



The next figure shows the same signal after it passes through the Amplifier block, with the **Method** parameter set to `Hyperbolic tangent`. The example uses the default Amplifier block **IIP3 (dBm)** value of 30. It uses no AM/PM conversion. The example specifies thermal noise as `Noise figure`, for which it uses the default 3.01 dB.



References

[1] Ghorbani, A. and M. Sheikhan, “The Effect of Solid State Power Amplifiers (SSPAs) Nonlinearities on MPSK and M-QAM Signal Transmission,” *Sixth Int'l Conference on Digital Processing of Signals in Comm.*, 1991, pp. 193-197.

[2] Rapp, C., “Effects of HPA-Nonlinearity on a 4-DPSK/OFDM-Signal for a Digital Sound Broadcasting System,” in *Proceedings of the Second European Conference on Satellite Communications*, Liege, Belgium, Oct. 22-24, 1991, pp. 179-184.

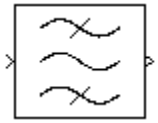
[3] Saleh, A.A.M., “Frequency-independent and frequency-dependent nonlinear models of TWT amplifiers,” *IEEE Trans. Communications*, vol. COM-29, pp.1715-1720, November 1981.

See Also

Bandpass RF Filter, Bandstop RF Filter, Highpass RF Filter, Lowpass RF Filter, Mixer

Bandpass RF Filter

Standard bandpass RF filters in baseband-equivalent complex form



Library

Mathematical

Note To use this block, you must install DSP System Toolbox software. For more information, see the RF Blockset release notes.

Description

The Bandpass RF Filter block lets you design standard analog bandpass filters, implemented in baseband-equivalent complex form. The following table describes the available design methods.

Design Method	Description
`Butterworth`	The magnitude response of a Butterworth filter is maximally flat in the passband and monotonic overall.
`Chebyshev I`	The magnitude response of a Chebyshev I filter is equiripple in the passband and monotonic in the stopband.
`Chebyshev II`	The magnitude response of a Chebyshev II filter is monotonic in the passband and equiripple in the stopband.
`Elliptic`	The magnitude response of an elliptic filter is equiripple in both the passband and the stopband.
`Bessel`	The delay of a Bessel filter is maximally flat in the passband.

The block input must be a discrete-time complex signal.

Note This block assumes a nominal impedance of 1 ohm.

Select the design of the filter from the **Design method** list in the dialog box. For each design method, the block enables you to specify the filter design parameters shown in the following table.

Design Method	Filter Design Parameters
`Butterworth`	Order, lower passband edge frequency, upper passband edge frequency
`Chebyshev I`	Order, lower passband edge frequency, upper passband edge frequency, passband ripple
`Chebyshev II`	Order, lower stopband edge frequency, upper stopband edge frequency, stopband attenuation
`Elliptic`	Order, lower passband edge frequency, upper passband edge frequency, passband ripple, stopband attenuation
`Bessel`	Order, lower passband edge frequency, upper passband edge frequency

The Bandpass RF Filter block designs the filters using the Signal Processing Toolbox™ filter design functions `buttap`, `cheb1ap`, `cheb2ap`, `ellipap`, and `besselap`.

Note Some RF blocks require the sample time to perform baseband modeling calculations. To ensure the accuracy of these calculations, the Input Port block, as well as the mathematical RF blocks, compare the input sample time to the sample time you provide in the mask. If they do not match, or if the input sample time is missing because the blocks are not connected, an error message appears.

Parameters

The parameters displayed in the dialog box vary for different design methods. Only some of these parameters are visible in the dialog box at any one time.

You can change tunable parameters while the model is running.

Design method

Filter design method. The design method can be `Butterworth`, `Chebyshev I`, `Chebyshev II`, `Elliptic`, or `Bessel`. Tunable.

Filter order

Order of the lowpass analog prototype filter that forms the basis for the bandpass filter design. The order of the final filter is twice this value.

Lower passband edge frequency (Hz)

Lower passband edge frequency for Butterworth, Chebyshev I, elliptic, and Bessel designs. Tunable.

Upper passband edge frequency (Hz)

Upper passband edge frequency for Butterworth, Chebyshev I, elliptic, and Bessel designs. Tunable.

Lower stopband edge frequency (Hz)

Lower stopband edge frequency for Chebyshev II designs. Tunable.

Upper stopband edge frequency (Hz)

Upper stopband edge frequency for Chebyshev II designs. Tunable.

Passband ripple in dB

Passband ripple for Chebyshev I and elliptic designs. Tunable.

Stopband attenuation in dB

Stopband attenuation for Chebyshev II and elliptic designs. Tunable.

Finite impulse response filter length

Desired length of the baseband-equivalent impulse response for the filter.

Center frequency (Hz)

Center of the modeling frequencies.

Sample time (s)

Time interval between consecutive samples of the input signal.

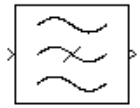
See Also

Amplifier, Bandstop RF Filter, Highpass RF Filter, Lowpass RF Filter, Mixer

`buttap`, `cheb1ap`, `cheb2ap`, `ellipap`, `besselap` (Signal Processing Toolbox)

Bandstop RF Filter

Standard bandstop RF filters in baseband-equivalent complex form



Library

Mathematical

Note To use this block, you must install DSP System Toolbox software. For more information, see the RF Blockset release notes.

Description

The Bandstop RF Filter block lets you design standard analog bandstop filters, implemented in baseband-equivalent complex form. The following table describes the available design methods.

Design Method	Description
`Butterworth`	The magnitude response of a Butterworth filter is maximally flat in the passband and monotonic overall.
`Chebyshev I`	The magnitude response of a Chebyshev I filter is equiripple in the passband and monotonic in the stopband.
`Chebyshev II`	The magnitude response of a Chebyshev II filter is monotonic in the passband and equiripple in the stopband.
`Elliptic`	The magnitude response of an elliptic filter is equiripple in both the passband and the stopband.

Design Method	Description
`Bessel`	The delay of a Bessel filter is maximally flat in the passband.

The block input must be a discrete-time complex signal.

Note This block assumes a nominal impedance of 1 ohm.

Select the design of the filter from the **Design method** list in the dialog box. For each design method, the block enables you to specify the filter design parameters shown in the following table.

Design Method	Filter Design Parameters
`Butterworth`	Order, lower passband edge frequency, upper passband edge frequency
`Chebyshev I`	Order, lower passband edge frequency, upper passband edge frequency, passband ripple
`Chebyshev II`	Order, lower stopband edge frequency, upper stopband edge frequency, stopband attenuation
`Elliptic`	Order, lower passband edge frequency, upper passband edge frequency, passband ripple, stopband attenuation
`Bessel`	Order, lower passband edge frequency, upper passband edge frequency

The Bandstop RF Filter block designs the filters using the Signal Processing Toolbox filter design functions `buttap`, `cheb1ap`, `cheb2ap`, `ellipap`, and `besselap`.

Note Some RF blocks require the sample time to perform baseband modeling calculations. To ensure the accuracy of these calculations, the Input Port block, as well as the mathematical RF blocks, compare the input sample time to the sample time you provide in the mask. If they do not match, or if the input sample time is missing because the blocks are not connected, an error message appears.

Parameters

The parameters displayed in the dialog box vary for different design methods. Only some of these parameters are visible in the dialog box at any one time.

You can change tunable parameters while the model is running.

Design method

Filter design method. The design method can be `Butterworth`, `Chebyshev I`, `Chebyshev II`, `Elliptic`, or `Bessel`. Tunable.

Filter order

Order of the lowpass analog prototype filter that forms the basis for the bandstop filter design. The order of the final filter is twice this value.

Lower passband edge frequency (Hz)

Lower passband edge frequency for Butterworth, Chebyshev I, elliptic, and Bessel designs. Tunable.

Upper passband edge frequency (Hz)

Upper passband edge frequency for Butterworth, Chebyshev I, elliptic, and Bessel designs. Tunable.

Lower stopband edge frequency (Hz)

Lower stopband edge frequency for Chebyshev II designs. Tunable.

Upper stopband edge frequency (Hz)

Upper stopband edge frequency for Chebyshev II designs. Tunable.

Passband ripple in dB

Passband ripple for Chebyshev I and elliptic designs. Tunable.

Stopband attenuation in dB

Stopband attenuation for Chebyshev II and elliptic designs. Tunable.

Finite impulse response filter length

Desired length of the baseband-equivalent impulse response for the filter.

Center frequency (Hz)

Center of the modeling frequencies.

Sample time (s)

Time interval between consecutive samples of the input signal.

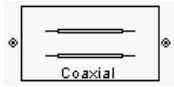
See Also

Amplifier, Bandpass RF Filter, Highpass RF Filter, Lowpass RF Filter, Mixer

`buttap`, `cheb1ap`, `cheb2ap`, `ellipap`, `besselap` (Signal Processing Toolbox)

Coaxial Transmission Line

Model coaxial transmission line

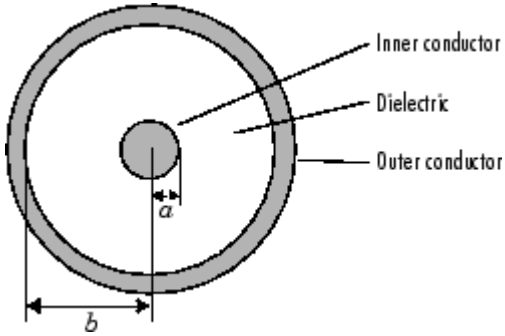


Library

Transmission Lines sublibrary of the Physical library

Description

The Coaxial Transmission Line block models the coaxial transmission line described in the block dialog box in terms of its frequency-dependent S-parameters. A coplanar waveguide transmission line is shown in cross-section in the following figure. Its physical characteristics include the radius of the inner conductor *a* and the radius of the outer conductor *b*.



The block lets you model the transmission line as a stub or as a stubless line.

Stubless Transmission Line

If you model a coaxial transmission line as a stubless line, the Coaxial Transmission Line block 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 block 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, a vector of modeling frequencies, determined by the Output Port block. Both can be expressed in terms of the resistance (R), inductance (L), conductance (G), and capacitance (C) per unit length (meters) as follows:

$$Z_0 = \sqrt{\frac{R + j\omega L}{G + j\omega C}}$$

$$k = k_r + jk_i = \sqrt{(R + j\omega L)(G + j\omega C)}$$

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:

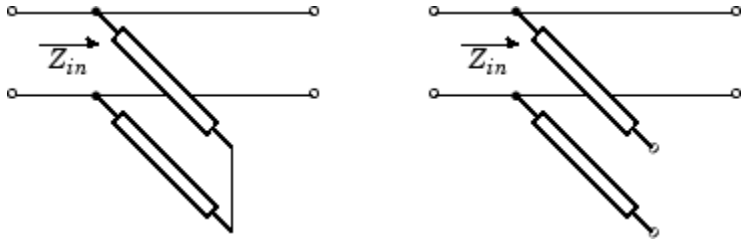
- a is the radius of the inner conductor.
- b is the radius of the outer conductor.
- σ_{cond} is the conductivity in the conductor.
- μ is the permeability of the dielectric.$\mu = \mu_0 \mu_r$where:
  - μ_0 is the permeability in free space.
  - μ_r is the **Relative permeability constant** parameter value.
- The is a complex dielectric constant given by $\varepsilon = \varepsilon' - j\varepsilon'' = \varepsilon' (1 - jtan\delta)$
- ε' is the real part of complex dielectric constant ε, $\varepsilon' = \varepsilon_0\varepsilon_r$. ε'' is the imaginary part of complex dielectric constant ε, $\varepsilon'' = \varepsilon_0\varepsilon_r$tan δ where :
  - ε_0 is the permittivity of free space.
  - ε_r is the **Relative permittivity constant** parameter value.
  - tan δ is the **Loss tangent of dielectric** parameter value.
- δ_{cond} is the skin depth of the conductor, which the block calculates as $1/\sqrt{\pi f \mu \sigma_{cond}}$.

Shunt and Series Stubs

If you model the transmission line as a shunt or series stub, the Coaxial Transmission Line block 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.

Shunt ABCD-Parameters

When you set the **Stub mode** parameter in the mask dialog box to `Shunt`, the two-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown here.

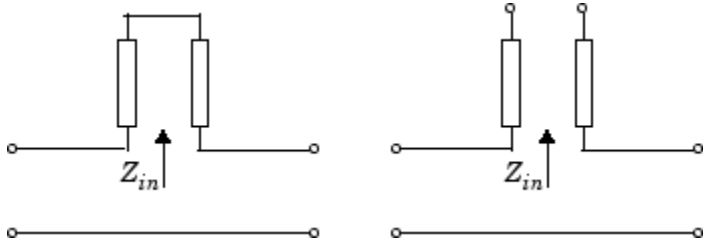


Z_{in} is the input impedance of the shunt circuit. The ABCD-parameters for the shunt stub are calculated as

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

Series ABCD-Parameters

When you set the **Stub mode** parameter in the mask dialog box to `Series`, the two-port network consists of a series transmission line that you can terminate with either a short circuit or an open circuit as shown here.



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

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

Parameters

Main Tab

Outer radius (m)

Radius of the outer conductor of the coaxial transmission line.

Inner radius (m)

Radius of the inner conductor of the coaxial transmission line.

Relative permeability constant

Relative permeability of the dielectric expressed as the ratio of the permeability of the dielectric to permeability in free space μ_0.

Relative permittivity constant

Relative permittivity of the dielectric expressed as the ratio of the permittivity of the dielectric to permittivity in free space ε_0.

Loss tangent of dielectric

Loss angle tangent of the dielectric.

Conductivity of conductor (S/m)

Conductivity of the conductor in siemens per meter.

Transmission line length (m)

Physical length of the transmission line.

Stub mode

Type of stub. Choices are `Not a stub`, `Shunt`, or `Series`.

Termination of stub

Stub termination for stub modes `Shunt` and `Series`. Choices are `Open` or `Short`. This parameter becomes visible only when **Stub mode** is set to `Shunt` or `Series`.

Visualization Tab

For information about plotting, see “Create Plots”.

References

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

See Also

Coplanar Waveguide Transmission Line, General Passive Network, Transmission Line, Microstrip Transmission Line, Parallel-Plate Transmission Line, Two-Wire Transmission Line

Connection Port

Connection port for RF subsystem

Library

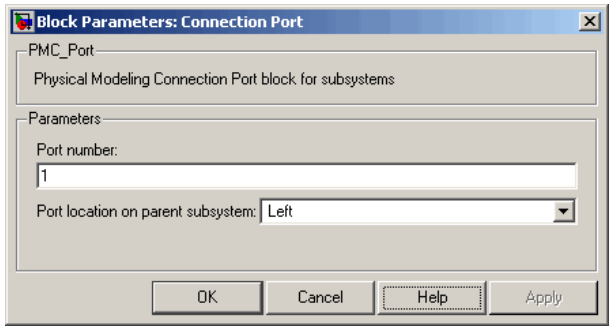
Input/Output Ports sublibrary of the Physical library

Description

The Connection Port block, placed inside a subsystem composed of RF Blockset Equivalent Baseband blocks, creates an open round physical modeling connector port ○ on the boundary of the subsystem. When it is connected to a connection line, the port becomes solid ●.

You connect individual blocks and subsystems made of RF Blockset Equivalent Baseband blocks to one another with connection lines instead of normal Simulink signal lines. These blocks and subsystems are anchored at the open, round physical modeling connector ports ○. Subsystems constructed out of RF Blockset Equivalent Baseband blocks automatically have such open round physical modeling connector ports. You can add additional connector ports by adding Connection Port blocks to your subsystem.

Dialog Box



Port number

This field labels the subsystem connector port created by this block. Multiple connector ports on the boundary of a single subsystem require different numbers as labels. The default value for the first port is `1`.

Port location on parent subsystem

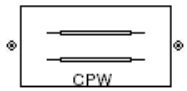
Use this parameter to choose on which side of the parent subsystem boundary the Port is placed. The choices are `Left` or `Right`. The default choice is `Left`.

See Also

Creating Subsystems (Simulink)

Coplanar Waveguide Transmission Line

Model coplanar waveguide transmission line

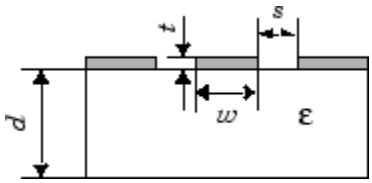


Library

Transmission Lines sublibrary of the Physical library

Description

The Coplanar Waveguide Transmission Line block models the coplanar waveguide transmission line described in the block dialog box in terms of its frequency-dependent S-parameters. 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 relative permittivity constant (ε).



The block lets you model the transmission line as a stub or as a stubless line.

Stubless Transmission Line

If you model a coplanar waveguide transmission line as a stubless line, the Coplanar Waveguide Transmission Line block 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 block calculates the ABCD-parameters using the physical length of the transmission line, d, and the complex propagation constant, k, using the following equations:

$$A = \frac{e^{kd} + e^{-kd}}{2}$$

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

$$C = \frac{e^{kd} - e^{-kd}}{2 * Z_0}$$

$$D = \frac{e^{kd} + e^{-kd}}{2}$$

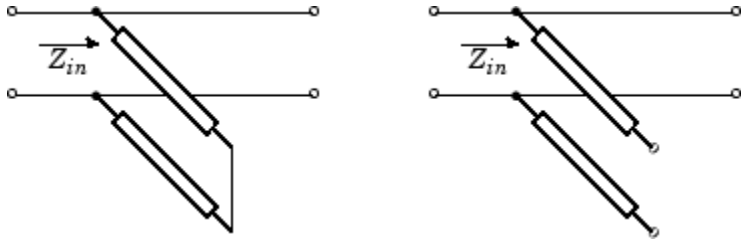
Z_0 and k are vectors whose elements correspond to the elements of f, a vector of modeling frequencies. Both can be expressed in terms of the specified conductor strip width, slot width, substrate height, conductor strip thickness, relative permittivity constant, conductivity and dielectric loss tangent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

Shunt and Series Stubs

If you model the transmission line as a shunt or series stub, the Coplanar Waveguide Transmission Line block first calculates the ABCD-parameters at each frequency contained in the vector of modeling frequencies. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

Shunt ABCD-Parameters

When you set the **Stub mode** parameter in the mask dialog box to `Shunt`, the two-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown here.

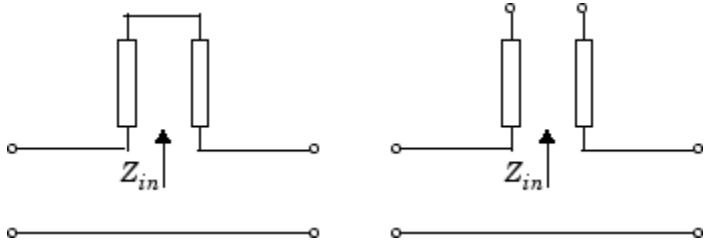


Z_{in} is the input impedance of the shunt circuit. The ABCD-parameters for the shunt stub are calculated as

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

Series ABCD-Parameters

When you set the **Stub mode** parameter in the mask dialog box to `Series`, the two-port network consists of a series transmission line that you can terminate with either a short circuit or an open circuit as shown here.



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

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

Parameters

Main Tab

Conductor width (m)

Physical width of the conductor.

Slot width (m)

Physical width of the slot.

Substrate height (m)

Thickness of the dielectric on which the conductor resides.

Strip thickness (m)

Physical thickness of the conductor.

Relative permittivity constant

Relative permittivity of the dielectric expressed as the ratio of the permittivity of the dielectric to permittivity in free space ε_0.

Conductivity of conductor (S/m)

Conductivity of the conductor in siemens per meter.

Loss tangent of dielectric

Loss angle tangent of the dielectric.

Transmission line length (m)

Physical length of the transmission line.

Stub mode

Type of stub. Choices are `Not a stub`, `Shunt`, or `Series`.

Termination of stub

Stub termination for stub modes `Shunt` and `Series`. Choices are `Open` or `Short`. This parameter becomes visible only when **Stub mode** is set to `Shunt` or `Series`.

Visualization Tab

For information about plotting, see “Create Plots”.

References

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

See Also

Coaxial Transmission Line, General Passive Network, Transmission Line, Microstrip Transmission Line, Parallel-Plate Transmission Line, Two-Wire Transmission Line

General Amplifier

Model nonlinear amplifier described by `rfdata` object or file data

Library

Amplifiers sublibrary of the Physical library

Description

The General Amplifier block models the nonlinear amplifier described by a data source. The data source consists of either an RF Toolbox data (`rfdata.data`) object or data from a file.

Network Parameters

If network parameter data and corresponding frequencies exist as S-parameters in the data source, the General Amplifier block interpolates the S-parameters to determine their values at the modeling frequencies. If the network parameters are Y- or Z-parameters, the block first converts them to S-parameters. For more information, see “Map Network Parameters to Modeling Frequencies”.

Nonlinearity

If power data exists in the data source, the block extracts the AMAM/AMPM nonlinearities from the power data.

If the data source contains no power data, then you can introduce nonlinearities into your model by specifying parameters in the **Nonlinearity Data** tab of the General Amplifier block dialog box. Depending on which of these parameters you specify, the block computes up to four of the coefficients c_1, c_3, c_5, and c_7 of the polynomial

$$F_{AM/AM}(s) = c_1 s + c_3|s|^2 s + c_5|s|^4 s + c_7|s|^6 s$$

that determines the AM/AM conversion for the input signal s. The block automatically calculates c_1, the linear gain term. If you do not specify additional nonlinearity data, the block operates as a linear amplifier. If you do, the block calculates one or more of the remaining coefficients as the solution to a system of linear equations, determined by the following method.

1 The block checks whether you have specified a value other than `Inf` for:

- The third-order intercept point ($OIP3$ or $IIP3$).
- The output power at the 1-dB compression point ($P_{1dB,\,out}$).
- The output power at saturation ($P_{sat,\,out}$).

In addition, if you have specified $P_{sat,\,out}$, the block uses the value for the gain compression at saturation (GC_{sat}). Otherwise, GC_{sat} is not used. You define each of these parameters in the block dialog box, on the **Nonlinearity Data** tab.

2 The block calculates a corresponding input or output value for the parameters that you have specified. In units of dB and dBm,

$$\begin{aligned} P_{sat,\,out} + GC_{sat} &= P_{sat,\,in} + G_{lin} \\ P_{1dB,\,out} + 1 &= P_{1dB,\,in} + G_{lin} \\ OIP3 &= IIP3 + G_{lin} \end{aligned}$$

where G_{lin} is c_1 in units of dB.

3 The block formulates the coefficients c_3, c_5, and c_7, where applicable, as the solutions to a system of one, two, or three linear equations. The number of equations is equal to the number of parameters that you provide. For example, if you specify all three parameters, the block formulates the coefficients according to the following equations:

$$\begin{aligned} \sqrt{P_{sat,\,out}} &= c_1\sqrt{P_{sat,\,in}} + c_3\left(\sqrt{P_{sat,\,in}}\right)^3 + c_5\left(\sqrt{P_{sat,\,in}}\right)^5 + c_7\left(\sqrt{P_{sat,\,in}}\right)^7 \\ \sqrt{P_{1dB,\,out}} &= c_1\sqrt{P_{1dB,\,in}} + c_3\left(\sqrt{P_{1dB,\,in}}\right)^3 + c_5\left(\sqrt{P_{1dB,\,in}}\right)^5 + c_7\left(\sqrt{P_{1dB,\,in}}\right)^7 \\ 0 &= \frac{c_1}{IIP3} + c_3 \end{aligned}$$

The first two equations are the evaluation of the polynomial $F_{AM/AM}(s)$ at the points $\left(\sqrt{P_{sat,\,in}}, \sqrt{P_{sat,\,out}}\right)$ and $\left(\sqrt{P_{1dB,\,in}}, \sqrt{P_{1dB,\,out}}\right)$, expressed in linear units (such as W or

mW) and normalized to a 1-Ω impedance. The third equation is the definition of the third-order intercept point.

The calculation omits higher-order terms according to the available degrees of freedom of the system. If you specify only two of the three parameters, the block does not use the equation involving the parameter you did not specify, and eliminates any c_7 terms from the remaining equations. Similarly, if you provide only one of the parameters, the block uses only the solution to the equation involving that parameter and omits any c_5 or c_7 terms.

If you provide vectors of nonlinearity and frequency data, the block calculates the polynomial coefficients using values for the parameters interpolated at the center frequency.

Active Noise

You can specify active block noise in one of the following ways:

- Spot noise data in the data source.
- Spot noise data in the block dialog box.
- Spot noise data object in the block dialog box.
- Noise figure, noise factor, or noise temperature value in the block dialog box.
- Frequency-dependent noise figure data (`rfdata.nf`) object in the block dialog box.

The latter four options are only available if noise data does not exist in the data source.

If you specify block noise as spot noise data, the block uses the data to calculate noise figure. The block first interpolates the noise data for the modeling frequencies, using the specified **Interpolation method**. It then calculates the noise figure using the resulting values.

Operating Conditions

Agilent® P2D and S2D files define block parameters for several operating conditions. Operating conditions are the independent parameter settings that are used when creating the file data. By default, RF Blockset Equivalent Baseband software defines the block behavior using the parameter 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 in the General Amplifier block dialog box.

Data Consistency

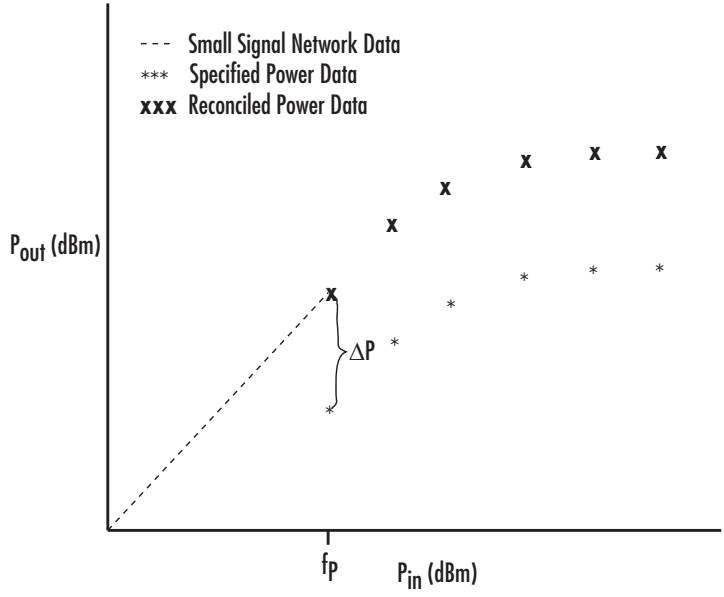
If the data source is a MathWorks AMP file or an Agilent S2D file that contains both network parameter data and power data, the blockset checks the data for consistency and reconciles it as necessary.

The blockset 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:

$$\Delta P = S_{21}(f_P) - P_{out}(f_P) + P_{in}(f_P)\ \ (dBm)$$

where f_P is the lowest frequency for which power data is specified.

If ΔP is more than 0.4 dB, a warning appears, and the blockset adds ΔP to the output power values at each specified input power value to resolve the discrepancy for simulation. The following graph shows this discrepancy.



Parameters

Main Tab

Data source

Determines the source of the data that describes the amplifier behavior. The data source must contain network parameters and may also include noise data, nonlinearity data, or both. The value can be `Data file` or `RFDATA object`.

Data file

If **Data source** is set to `Data file`, use this field to specify the name of the file that contains the amplifier data. The file name must include the extension. If the file is not in your MATLAB path, specify the full path to the file or click the **Browse** button to find the file.

RFDATA object

If **Data source** is set to `RFDATA object`, use this field to specify an RF Toolbox data (`rfdata.data`) object that describes an amplifier. You can specify the object as:

- The handle of a data object previously created using RF Toolbox software.
- An RF Toolbox command such as `rfdata.data('Freq',1e9,'S_Parameters',[0 0; 0.5 0])`, which creates a data object.
- A MATLAB expression that generates such an object.

Interpolation method

The method used to interpolate the network parameters. The following table lists the available methods describes each one.

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

Noise Data Tab

Noise type

Type of noise data. The value can be `Noise figure`, `Spot noise data`, `Noise factor`, or `Noise temperature`. This parameter is disabled if the data source contains noise data.

Noise figure (dB)

Scalar ratio or vector of ratios, in decibels, of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Noise figure`.

Minimum noise figure (dB)

Minimum scalar ratio or vector of minimum ratios of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Spot noise data`.

Optimal reflection coefficient

Optimal amplifier source impedance. This parameter is enabled if **Noise type** is set to `Spot noise data`. The value can be a scalar or vector.

Equivalent normalized resistance

Resistance or vector of resistances normalized to the resistance value or values used to take the noise measurement. This parameter is enabled if **Noise type** is set to `Spot noise data`.

Noise factor

Scalar ratio or vector of ratios of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Noise factor`.

Noise temperature (K)

Equivalent temperature or vector of temperatures that produce the same amount of noise power as the amplifier. This parameter is enabled if **Noise type** is set to `Noise temperature`.

Frequency (Hz)

Scalar value or vector corresponding to the domain of frequencies over which you are specifying the noise data. If you provide a scalar value for your noise data, the block ignores the **Frequency (Hz)** parameter and uses the noise data for all frequencies. If you provide a vector of values for your noise data, it must be the same size as the vector of frequencies. The block uses the **Interpolation method** specified in the **Main** tab to interpolate noise data.

Nonlinearity Data Tab

IP3 type

Type of third-order intercept point. The value can be `IIP3` (input intercept point) or `OIP3` (output intercept point). This parameter is disabled if the data source contains power data or IP3 data.

IP3 (dBm)

Value of third-order intercept point. This parameter is disabled if the data source contains power data or IP3 data. Use the default value, `Inf`, if you do not know the IP3 value. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

1 dB gain compression power (dBm)

Output power value ($P_{1dB,\,out}$) at which gain has decreased by 1 dB. This parameter is disabled if the data source contains power data or 1-dB compression point data. Use the default value, `Inf`, if you do not know the 1-dB compression point. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Output saturation power (dBm)

Output power value ($P_{sat,\,out}$) that the amplifier produces when fully saturated. This parameter is disabled if the data source contains output saturation power data. Use the default value, `Inf`, if you do not know the saturation power. If you specify this parameter, you must also specify the **Gain compression at saturation (dB)**. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Gain compression at saturation (dB)

Decrease in gain (GC_{sat}) when the power is fully saturated. The block ignores this parameter if you do not specify the **Output saturation power (dBm)**. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Frequency (Hz)

Scalar or vector value of frequency points corresponding to the third-order intercept and power data. This parameter is disabled if the data source contains power data or IP3 data. If you use a scalar value, the **IP3 (dBm)**, **1 dB gain compression power (dBm)**, and **Output saturation power (dBm)** parameters must all be scalars. If you use a vector value, one or more of the **IP3 (dBm)**, **1 dB gain compression power (dBm)**, and **Output saturation power (dBm)** parameters must also be a vector.

Visualization Tab

For information about plotting, see “Create Plots”. Use `rftool` or the RF Toolbox plotting functions to plot other data.

Operating Conditions Tab

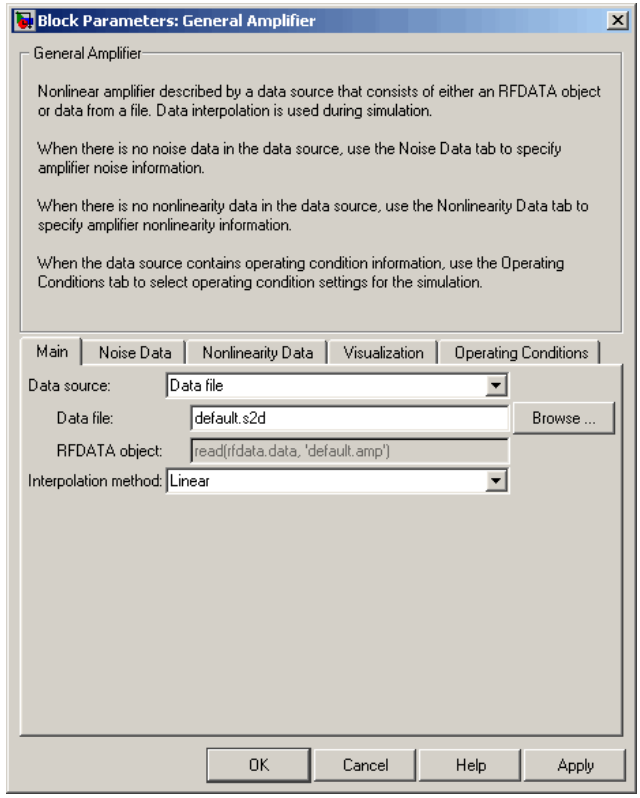
If the data source contains data at multiple operating conditions, the **Operating Conditions** tab contains two columns. The **Conditions** column shows the available conditions, and the **Values** column contains a drop-down list of the available values for the corresponding condition. Use the drop-down lists to specify the operating condition values to use in simulation.

Examples

Creating a General Amplifier Block from File Data

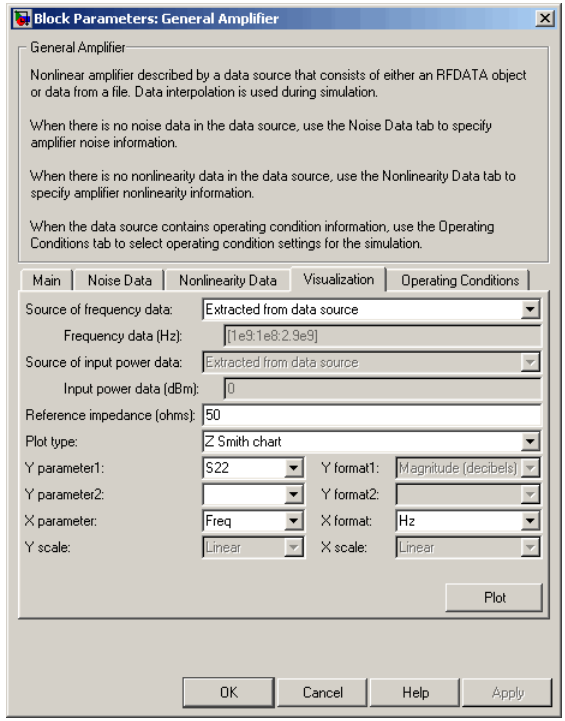
This example uses the default data source, which is the nonlinear amplifier in the file `default.s2d`. The file contains S-parameters for frequencies from 1.0 to 2.9 GHz at intervals of 0.01 GHz, power data at frequency 2.1 GHz, and active noise parameters. By default, the General Amplifier block uses linear interpolation to model the network described in the object.

1. On the **Main** tab, accept the default settings.

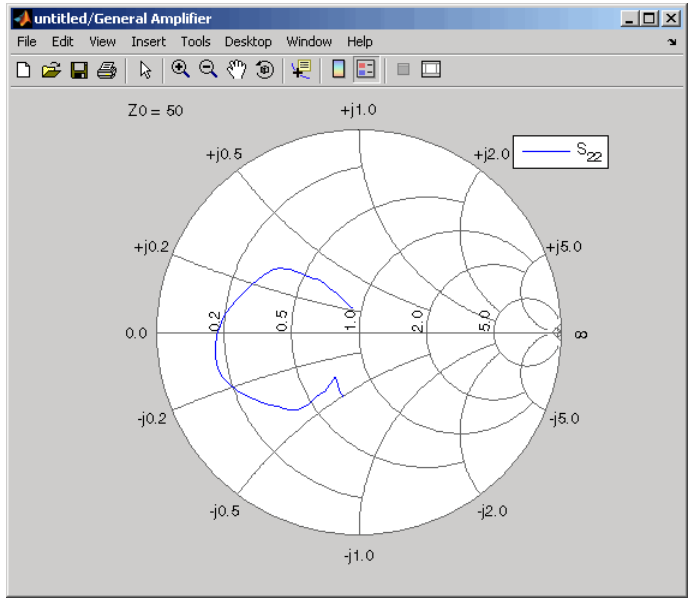


2 On the **Visualization** tab, set the parameters as follows:

- In the **Plot type** list, select `Z Smith chart`.
- In the **Y parameter1** list, select `S22`.



Click **Plot**. This action creates Z Smith chart of the S_{22} parameters using the frequencies taken from the data source.



For more about using an Agilent `.s2d` file in a Simulink model, see Effect of Nonlinear Amplifier on QPSK Modulation.

See Also

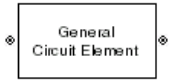
Output Port, S-Parameters Amplifier, Y-Parameters Amplifier, Z-Parameters Amplifier

`rfdata.data` (RF Toolbox)

`interp1` (MATLAB)

General Circuit Element

Model two-port network described by `rfckt` object



Library

Black Box Elements sublibrary of the Physical library

Description

The General Circuit Element block models the two-port network described by an RF Toolbox circuit (`rfckt`) object.

The block uses the `rfckt/analyze` method to calculate the network parameters at the modeling frequencies.

Parameters

Main Tab

RFCKT object

An RF Toolbox circuit (`rfckt`) object. You can specify the object as (1) the handle of a circuit object previously created using RF Toolbox software, (2) an RF Toolbox command such as `rfckt.txline`, `rfckt.coaxial`, or `rfckt.cascade` that creates a default circuit object of the specified type, or (3) a MATLAB expression that generates such an object. See “RF Circuit Objects” (RF Toolbox) for more information about circuit objects.

Visualization Tab

For information about plotting, see “Create Plots”.

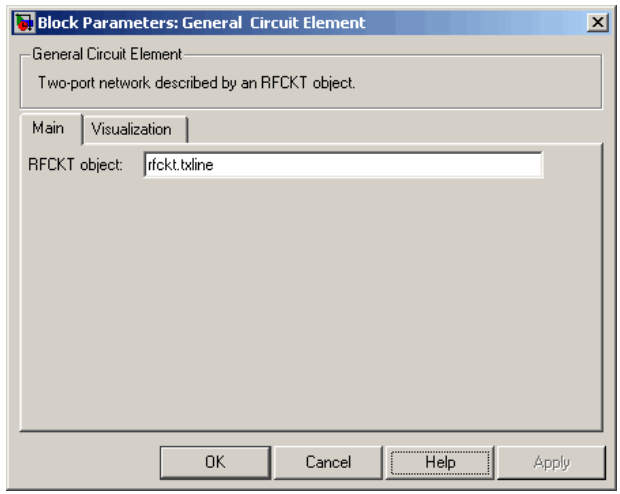
Examples

Creating a General Circuit Element from an RF Toolbox Object

This example uses the `rfckt.txline` object, which describes a transmission line.

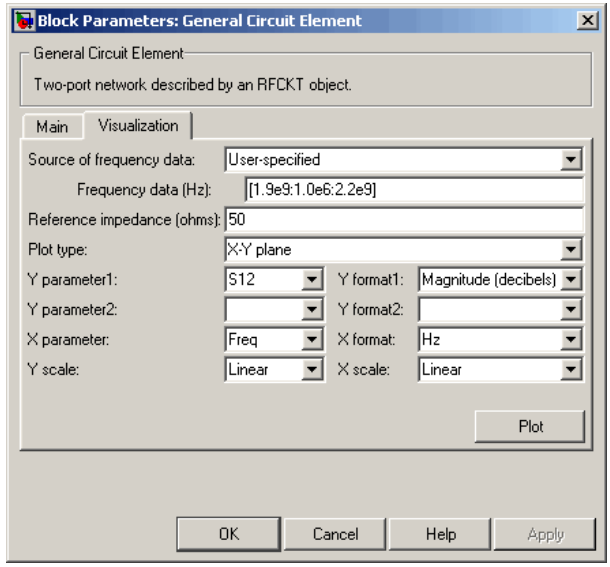
1 On the **Main** tab, set the **RFCKT object** parameter to `rfckt.txline`.

Click **Apply**. This action applies the specified settings.

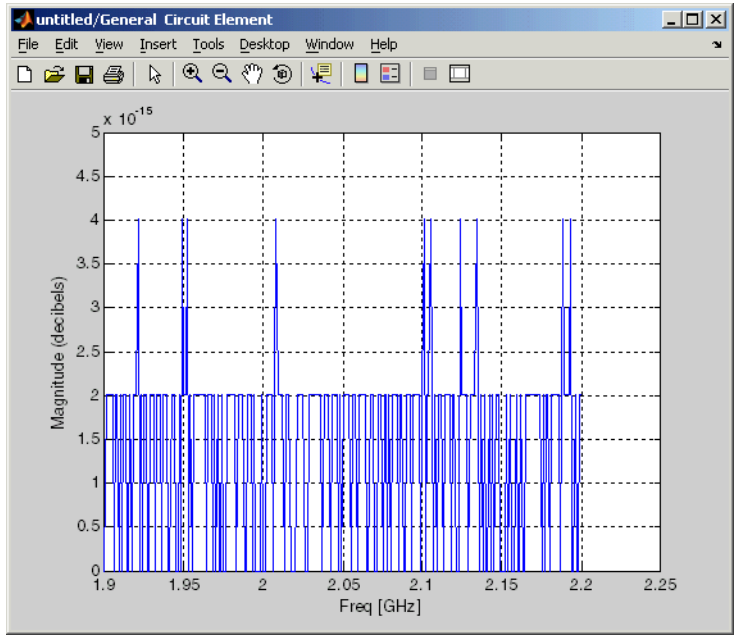


2 Set the General Circuit Element block parameters on the **Visualization** tab as follows:

- In the **Y parameter1** list, select `S12`.



Click **Plot**. This action creates an X-Y Plane plot of the S_{12} parameters in the frequency range 1.9 to 2.2 GHz.



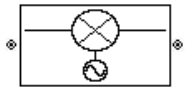
See Also

General Passive Network, S-Parameters Passive Network, Y-Parameters Passive Network, Z-Parameters Passive Network

`interp1` (MATLAB)

General Mixer

Model mixer and local oscillator described by `rfdata` object



Library

Mixer sublibrary of the Physical library

Description

The General Mixer block models the mixer described by an RF Toolbox data (`rfdata.data`) object.

Network Parameters

The network parameter values all refer to the mixer input frequency. If network parameter data and corresponding frequencies exist as S-parameters in the `rfdata.data` object, the General Mixer block interpolates the S-parameters to determine their values at the modeling frequencies. If the block contains network Y- or Z-parameters, the block first converts them to S-parameters. See “Map Network Parameters to Modeling Frequencies” for more details.

RF Blockset Equivalent Baseband software computes the reflected wave at the mixer input (b_1) and at the mixer output (b_2) from the interpolated S-parameters as

$$\begin{bmatrix} b_1(f_{in}) \\ b_2(f_{out}) \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} a_1(f_{in}) \\ a_2(f_{out}) \end{bmatrix}$$

where

- f_{in} and f_{out} are the mixer input and output frequencies, respectively.
- a_1 and a_2 are the incident waves at the mixer input and output, respectively.

The interpolated S_{21} parameter values describe the conversion gain as a function of frequency, referred to the mixer input frequency.

Active Noise

You can specify active block noise in one of the following ways:

- Spot noise data in the data source.
- Spot noise data in the block dialog box.
- Spot noise data (`rfdata.noise`) object in the block dialog box.
- Noise figure, noise factor, or noise temperature value in the block dialog box.
- Frequency-dependent noise figure data (`rfdata.nf`) object in the block dialog box.

The latter four options are only available if noise data does not exist in the data source.

If you specify block noise as spot noise data, the block uses the data to calculate noise figure. The block first interpolates the noise data for the modeling frequencies, using the specified **Interpolation method**. It then calculates the noise figure using the resulting values.

Phase Noise

The General Mixer block applies phase noise to a complex baseband signal. The block first generates additive white Gaussian noise (AWGN) and filters the noise with a digital FIR filter. It then adds the resulting noise to the angle component of the input signal.

The blockset computes the digital filter by:

1. Interpolating the specified phase noise level to determine the phase noise values at the modeling frequencies.
2. Taking the IFFT of the resulting phase noise spectrum to get the coefficients of the FIR filter.

Note If you specify phase noise as a scalar value, the blockset assumes that the phase noise is constant at all modeling frequencies and does not have a *1/f* slope. This assumption differs from that made by the Mathematical Mixer block.

Nonlinearity

If power data exists in the data source, the block extracts the AMAM/AMPM nonlinearities from it.

If the data source contains no power data, then you can introduce nonlinearities into your model by specifying parameters in the **Nonlinearity Data** tab of the General Mixer block dialog box. Depending on which of these parameters you specify, the block computes up to four of the coefficients c_1, c_3, c_5, and c_7 of the polynomial

$$F_{AM/AM}(s) = c_1 s + c_3|s|^2 s + c_5|s|^4 s + c_7|s|^6 s$$

that determines the AM/AM conversion for the input signal s. The block automatically calculates c_1, the linear gain term. If you do not specify additional nonlinearity data, the block operates as a mixer with a linear gain. If you do, the block calculates one or more of the remaining coefficients as the solution to a system of linear equations, determined by the following method.

1. The block checks whether you have specified a value other than `Inf` for:

   - The third-order intercept point ($OIP3$ or $IIP3$).
   - The output power at the 1-dB compression point ($P_{1dB,\,out}$).
   - The output power at saturation ($P_{sat,\,out}$).

   In addition, if you have specified $P_{sat,\,out}$, the block uses the value for the gain compression at saturation (GC_{sat}). Otherwise, GC_{sat} is not used. You define each of these parameters in the block dialog box, on the **Nonlinearity Data** tab.

2. The block calculates a corresponding input or output value for the parameters you have specified. In units of dB and dBm,

   $$P_{sat,\,out} + GC_{sat} = P_{sat,\,in} + G_{lin}$$
   $$P_{1dB,\,out} + 1 = P_{1dB,\,in} + G_{lin}$$
   $$OIP3 = IIP3 + G_{lin}$$

   where G_{lin} is c_1 in units of dB.

3. The block formulates the coefficients c_3, c_5, and c_7, where applicable, as the solutions to a system of one, two, or three linear equations. The number of equations used is equal to the number of parameters you provide. For example, if you specify all three

parameters, the block formulates the coefficients according to the following equations:

$$\sqrt{P_{sat,out}} = c_1\sqrt{P_{sat,in}} + c_3\left(\sqrt{P_{sat,in}}\right)^3 + c_5\left(\sqrt{P_{sat,in}}\right)^5 + c_7\left(\sqrt{P_{sat,in}}\right)^7$$

$$\sqrt{P_{1dB,out}} = c_1\sqrt{P_{1dB,in}} + c_3\left(\sqrt{P_{1dB,in}}\right)^3 + c_5\left(\sqrt{P_{1dB,in}}\right)^5 + c_7\left(\sqrt{P_{1dB,in}}\right)^7$$

$$0 = \frac{c_1}{IIP3} + c_3$$

The first two equations are the evaluation of the polynomial $F_{AM/AM}(s)$ at the points $(\sqrt{P_{sat,in}}, \sqrt{P_{sat,out}})$ and $(\sqrt{P_{1dB,in}}, \sqrt{P_{1dB,out}})$, expressed in linear units (such as W or mW) and normalized to a 1-Ω impedance. The third equation is the definition of the third-order intercept point.

The calculation omits higher-order terms according to the available degrees of freedom of the system. If you specify only two of the three parameters, the block does not use the equation involving the parameter you did not specify, and eliminates any c_7 terms from the remaining equations. Similarly, if you provide only one of the parameters, the block uses only the solution to the equation involving that parameter and omits any c_5 or c_7 terms.

If you provide vectors of nonlinearity and frequency data, the block calculates the polynomial coefficients using values for the parameters interpolated at the center frequency.

Operating Conditions

Agilent P2D and S2D files define block parameters for several operating conditions. Operating conditions are the independent parameter settings that are used when creating the file data. By default, the blockset defines the block behavior using the parameter 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 in the General Mixer block dialog box.

Parameters

Main Tab

Data source

Determines the source of the data that describes the mixer behavior. The data source must contain network parameters and may also include noise data, nonlinearity data, or both. The value can be `Data file` or `RFDATA object`.

Data file

If **Data source** is set to `Data file`, use this field to specify the name of the file that contains the mixer data. The file name must include the extension. If the file is not in your MATLAB path, specify the full path to the file or click the **Browse** button to find the file.

Note If the data file contains an intermodulation table, the General Mixer block ignores the table. Use RF Toolbox software to ensure the cascade has no significant spurs in the frequency band of interest before running a simulation.

RFDATA object

If **Data source** is set to `RFDATA object`, use this field to specify an RF Toolbox data (`rfdata.data`) object that describes a mixer. You can specify the object as one of the following:

- The handle of a data object previously created using RF Toolbox software.
- An RF Toolbox command such as `rfdata.data('Freq',1e9,'S_Parameters',[0 0; 0.5 0])`, which creates a data object.
- A MATLAB expression that generates a data object.

For more information about data objects, see the `rfdata.data` reference page in the RF Toolbox documentation.

Interpolation method

The method used to interpolate the network parameters. The following table lists the available methods describes each one.

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

Mixer Type

Type of mixer. Choices are `Downconverter` (default) and `Upconverter`.

LO frequency (Hz)

Local oscillator frequency. If you choose `Downconverter`, the blockset computes the mixer output frequency, f_{out}, from the mixer input frequency, f_{in}, and the local oscillator frequency, f_{lo}, as $f_{out} = f_{in} - f_{lo}$. If you choose `Upconverter`, $f_{out} = f_{in} + f_{lo}$.

Note For a downconverting mixer, the local oscillator frequency must satisfy the condition $f_{in} - f_{lo} \geq 1/(2t_s)$, where t_s is the sample time specified in the Input Port block. Otherwise, an error appears.

Noise Data Tab

Phase noise frequency offset (Hz)

Vector specifying the frequency offset.

Phase noise level (dBc/Hz)

Vector specifying the phase noise level.

Noise type

Type of noise data. The value can be `Noise figure`, `Spot noise data`, `Noise factor`, or `Noise temperature`. This parameter is disabled if the data source contains noise data.

Noise figure (dB)

Scalar ratio or vector of ratios, in decibels, of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Noise figure`.

Minimum noise figure (dB)

Minimum scalar ratio or vector of minimum ratios of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Spot noise data`.

Optimal reflection coefficient

Optimal mixer source impedance. This parameter is enabled if **Noise type** is set to `Spot noise data`. The value can be a scalar or vector.

Equivalent normalized resistance

Resistance or vector of resistances normalized to the resistance value or values used to take the noise measurement. This parameter is enabled if **Noise type** is set to `Spot noise data`.

Noise factor

Scalar ratio or vector of ratios of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Noise factor`.

Noise temperature (K)

Equivalent temperature or vector of temperatures that produce the same amount of noise power as the mixer. This parameter is enabled if **Noise type** is set to `Noise temperature`.

Frequency (Hz)

Scalar value or vector corresponding to the domain of frequencies over which you are specifying the noise data. If you provide a scalar value for your noise data, the block ignores the **Frequency (Hz)** parameter and uses the noise data for all frequencies. If you provide a vector of values for your noise data, it must be the same size as the vector of frequencies. The block uses the **Interpolation method** specified in the **Main** tab to interpolate noise data.

Nonlinearity Data Tab

IP3 type

Type of third-order intercept point. The value can be `IIP3` (input intercept point) or `OIP3` (output intercept point). This parameter is disabled if the data source contains power data or IP3 data.

IP3 (dBm)

Value of third-order intercept point. This parameter is disabled if the data source contains power data or IP3 data. Use the default value, `Inf`, if you do not know the IP3 value. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

1 dB gain compression power (dBm)

Output power value ($P_{1dB,\,out}$) at which gain has decreased by 1 dB. This parameter is disabled if the data source contains power data or 1-dB compression point data. Use the default value, `Inf`, if you do not know the 1 dB compression point. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Output saturation power (dBm)

Output power value ($P_{sat,\,out}$) that the mixer produces when fully saturated. This parameter is disabled if the data source contains output saturation power data. Use the default value, `Inf`, if you do not know the saturation power. If you specify this parameter, you must also specify the **Gain compression at saturation (dB)**. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Gain compression at saturation (dB)

Decrease in gain (GC_{sat}) when the power is fully saturated. The block ignores this parameter if you do not specify the **Output saturation power (dBm)**. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Frequency (Hz)

Scalar or vector value of frequency points corresponding to the third-order intercept and power data. This parameter is disabled if the data source contains power data or IP3 data. If you use a scalar value, the **IP3 (dBm)**, **1 dB gain compression power (dBm)**, and **Output saturation power (dBm)** parameters must all be scalars. If you use a vector value, one or more of the **IP3 (dBm)**, **1 dB gain compression power (dBm)**, and **Output saturation power (dBm)** parameters must also be a vector.

Visualization Tab

For information about plotting, see “Create Plots”. Use `rftool` or the RF Toolbox plotting functions to plot other data.

Operating Conditions Tab

If the data source contains data at multiple operating conditions, the **Operating Conditions** tab contains two columns. The **Conditions** column shows the available conditions, and the **Values** column contains a drop-down list of the available values for

the corresponding condition. Use the drop-down lists to specify the operating condition values to use in simulation.

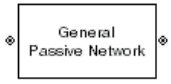
See Also

Output Port, S-Parameters Mixer, Y-Parameters Mixer, Z-Parameters Mixer

`rfdata.data` (RF Toolbox)

General Passive Network

Model two-port passive network described by `rfdata` object



Library

Black Box Elements sublibrary of the Physical library

Description

The General Passive Network block models the two-port passive network described by an RF Toolbox data (`rfdata.data`) object.

If network parameter data and their corresponding frequencies exist as S-parameters in the `rfdata.data` object, the General Passive Network block interpolates the S-parameters to determine their values at the modeling frequencies. If the block contains network Y- or Z-parameters, the block first converts them to S-parameters. See “Map Network Parameters to Modeling Frequencies” for more details.

Parameters

Main Tab

Data source

Determines the source of the data that describes the passive device behavior. The data source must contain network parameters and may also include noise data, nonlinearity data, or both. The value can be `Data file` or `RFDATA object`.

RFDATA object

If **Data source** is set to `RFDATA object`, use this field to specify an RF Toolbox data (`rfdata.data`) object. You can specify the object as (1) the handle of a data object

previously created using RF Toolbox software, (2) an RF Toolbox command such as `rfdata.data('Freq',1e9,'S_Parameters',[0 0; 0.5 0])`, which creates a data object, or (3) a MATLAB expression that generates such an object.

Data file

If **Data source** is set to `Data file`, use this field to specify the name of the file that contains the amplifier data. The file name must include the extension. If the file is not in your MATLAB path, specify the full path to the file or click the **Browse** button to find the file.

Interpolation method

The method used to interpolate the network parameters. The following table lists the available methods describes each one.

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

Visualization Tab

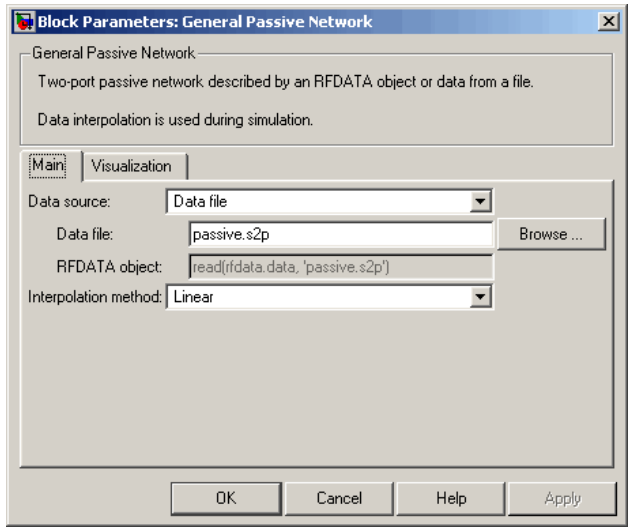
For information about plotting, see “Create Plots”.

Examples

Creating a General Passive Network Block from File Data

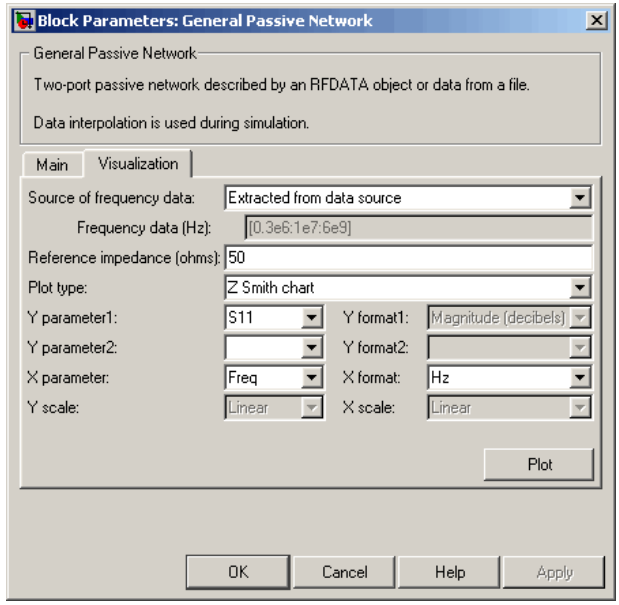
This example creates a two-port passive network from the data in the file `passive.s2p`. The file contains S-parameters for frequencies from about 0.315 MHz to 6.0 GHz. The General Passive Network block uses linear interpolation to model the network described in the object.

1 On the **Main** tab, accept the default settings.

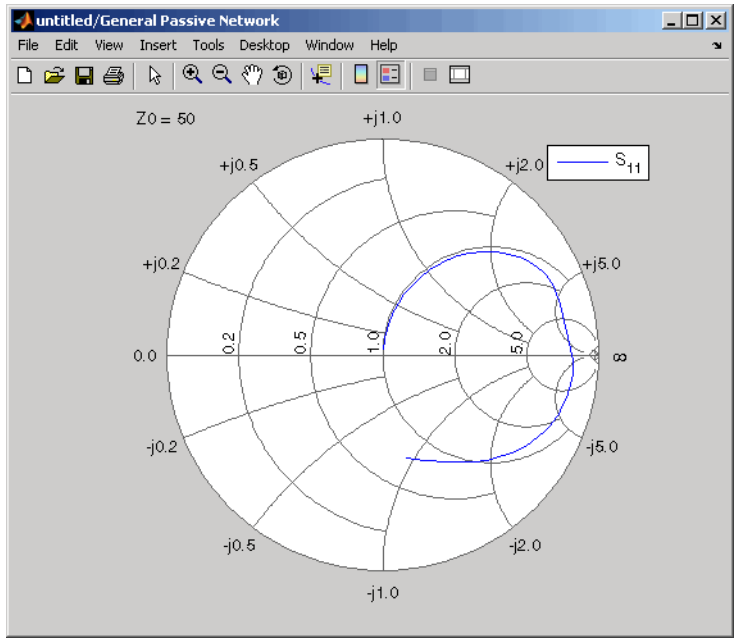


2 On the **Visualization** tab, set the parameters as follows:

- In the **Plot type** list, select `Z Smith chart`.



Click **Plot**. This action creates a Z Smith chart of the S_{11} parameters, using the frequencies taken from the **RFDATA object** parameter on the **Main** tab.



See Also

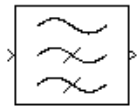
General Circuit Element, Output Port, S-Parameters Passive Network, Y-Parameters Passive Network, Z-Parameters Passive Network

`rfdata.data` (RF Toolbox)

`interp1` (MATLAB)

Highpass RF Filter

Standard highpass RF filters in baseband-equivalent complex form



Library

Mathematical

Note To use this block, you must install DSP System Toolbox software. For more information, see the RF Blockset release notes.

Description

The Highpass RF Filter block lets you design standard analog highpass filters, implemented in baseband-equivalent complex form. The following table describes the available design methods.

Design Method	Description
`Butterworth`	The magnitude response of a Butterworth filter is maximally flat in the passband and monotonic overall.
`Chebyshev I`	The magnitude response of a Chebyshev I filter is equiripple in the passband and monotonic in the stopband.
`Chebyshev II`	The magnitude response of a Chebyshev II filter is monotonic in the passband and equiripple in the stopband.
`Elliptic`	The magnitude response of an elliptic filter is equiripple in both the passband and the stopband.

Design Method	Description
`Bessel`	The delay of a Bessel filter is maximally flat in the passband.

The block input must be a discrete-time complex signal.

Note This block assumes a nominal impedance of 1 ohm.

Select the design of the filter from the **Design method** list in the dialog box. For each design method, the block lets you specify the filter design parameters shown in the following table.

Design Method	Filter Design Parameters
`Butterworth`	Order, passband edge frequency
`Chebyshev I`	Order, passband edge frequency, passband ripple
`Chebyshev II`	Order, stopband edge frequency, stopband attenuation
`Elliptic`	Order, passband edge frequency, passband ripple, stopband attenuation
`Bessel`	Order, passband edge frequency

The Highpass RF Filter block designs the filters using the Signal Processing Toolbox filter design functions `buttap`, `cheb1ap`, `cheb2ap`, `ellipap`, and `besselap`.

Note Some RF blocks require the sample time to perform baseband modeling calculations. To ensure the accuracy of these calculations, the Input Port block, as well as the mathematical RF blocks, compare the input sample time to the sample time you provide in the mask. If they do not match, or if the input sample time is missing because the blocks are not connected, an error message appears.

Parameters

The parameters displayed in the dialog box vary for different design methods. Only some of these parameters are visible in the dialog box at any one time.

You can change tunable parameters while the model is running.

Design method

Filter design method. The design method can be `Butterworth`, `Chebyshev I`, `Chebyshev II`, `Elliptic`, or `Bessel`. Tunable.

Filter order

Order of the filter.

Passband edge frequency (Hz)

Passband edge frequency for Butterworth, Chebyshev I, elliptic, and Bessel designs. Tunable.

Stopband edge frequency (Hz)

Stopband edge frequency for Chebyshev II designs. Tunable.

Passband ripple in dB

Passband ripple for Chebyshev I and elliptic designs. Tunable.

Stopband attenuation in dB

Stopband attenuation for Chebyshev II and elliptic designs. Tunable.

Finite impulse response filter length

Desired length of the baseband-equivalent impulse response for the filter.

Center frequency (Hz)

Center of the modeling frequencies.

Sample time

Time interval between consecutive samples of the input signal.

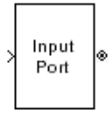
See Also

Amplifier, Bandpass RF Filter, Bandstop RF Filter, Lowpass RF Filter, Mixer

`buttap`, `cheb1ap`, `cheb2ap`, `ellipap`, `besselap` (Signal Processing Toolbox)

Input Port

Connection block from Simulink environment to RF physical blocks



Library

Input/Output Ports sublibrary of the Physical library

Description

The Input Port block is a connecting port from the Simulink, or mathematical, part of the model to an RF physical part of the model. The Input Port block lets you provide the parameter data needed to calculate the modeling frequencies and the baseband-equivalent impulse response for the physical subsystem. It also lets you specify information about how to interpret the incoming Simulink signal.

For more information about how the Input Port block converts the mathematical Simulink signals to physical modeling environment signals, see “Convert to and from Simulink Signals”.

Note Some RF blocks use the sample time to perform baseband modeling calculations. To ensure the accuracy of these calculations, the Input Port block, as well as the mathematical RF blocks, compare the input sample time to the sample time you provide in the Input Port mask. If they do not match, or if the input sample time is missing because the blocks are not connected, an error message appears.

Parameters

Treat input Simulink signal as

Select one of the following options for interpreting the input Simulink signal:

- `Incident power wave` — Interpret the input signal as the incident power wave of the RF system described in the physical model to which it connects. When you select this option, the output signal of the RF system is the transmitted power wave. This is the most common RF modeling interpretation.
- `Source voltage` — Interpret the input signal as the source voltage of the RF system described in the physical model to which it connects. As a result, the baseband-equivalent model includes the loss through the source impedance. When you select this option, the output signal of the RF system is the load voltage.

For more information about these options, see “Convert to and from Simulink Signals”.

Source impedance (ohms)

Source impedance of the RF network described in the physical model to which it connects.

Finite impulse response filter length

Desired length of the baseband-equivalent impulse response for the physical model. The longer the FIR filter in the time domain, the finer the frequency resolution in the frequency domain. The frequency resolution is approximately equal to 1/ (**Finite impulse response filter length*****Sample time (s)**). For a graphical representation of this parameter, see “Baseband-Equivalent Modeling”.

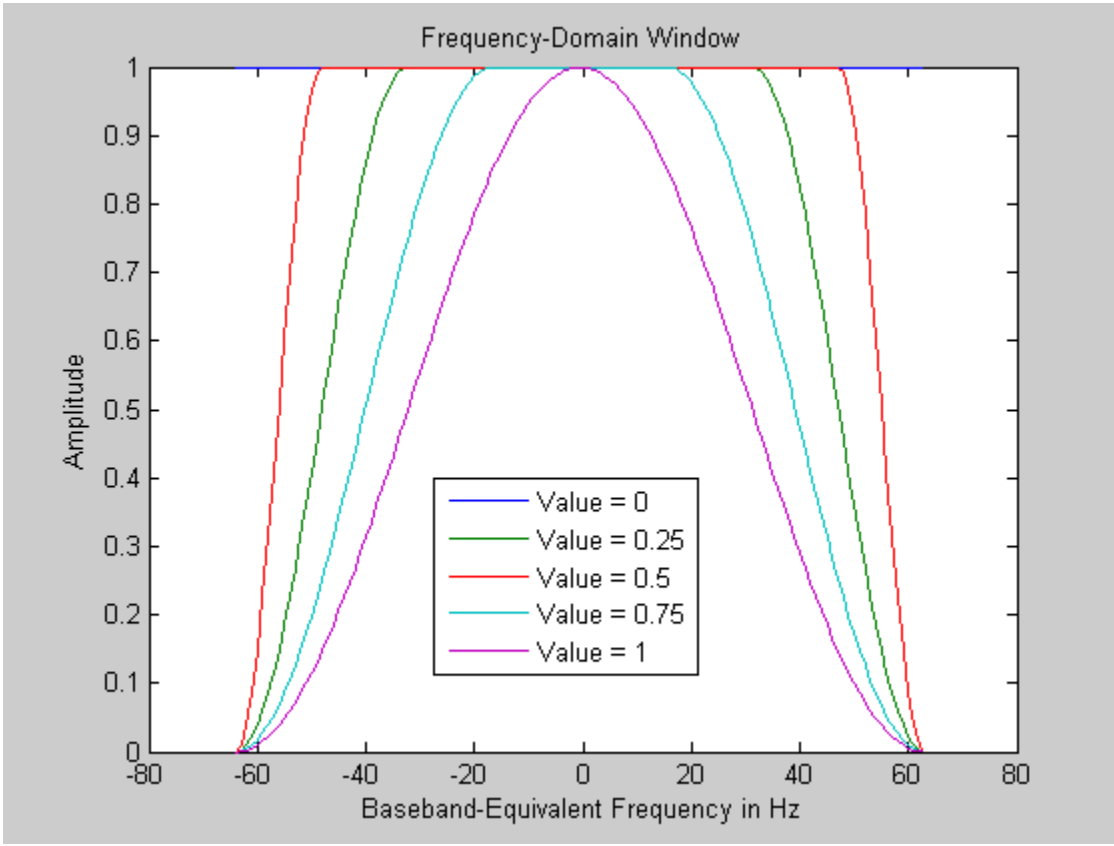
Note The equivalent-baseband simulation algorithm uses the next power of 2 greater than the specified filter length in its calculations and then truncates the impulse response to the specified length. As a result, you get different results when you set the **Finite impulse response filter length** parameter to a number that is not a power of 2. For more information, see “Calculate the Baseband-Equivalent Impulse Response”.

Fractional bandwidth of guard bands

Fraction of modeling bandwidth over which to taper the edges of the transfer function of the system when creating the baseband-equivalent model. This parameter defines the ratio of the bandwidth of sections that are tapered using a Tukey, or cosine-tapered, window to the bandwidth of the constant, or untapered, sections.

A value less than or equal to 0 tells the Input Port block to use a rectangular (`rectwin`) window. A value greater than or equal to 1 tells the Input Port block to use a hann window.

The blockset uses the Signal Processing Toolbox `tukeywin` function to generate the window. The following figure shows the resulting frequency-domain window for several values of the **Fractional bandwidth of guard bands** parameter.



See “Create Complex Baseband-Equivalent Model” for information about how the Input Port block applies this window to reduce the Gibbs phenomenon (also known as ringing), and other artifacts in the baseband-equivalent model of the system.

Modeling delay (samples)

Number of time samples by which to delay the impulse response of the baseband-equivalent model to ensure that the baseband-equivalent model has a causal response.

See “Create Complex Baseband-Equivalent Model” for information on how the Input Port block applies this delay to ensure a causal response.

Center frequency (Hz)

Center of the modeling frequencies. See the Output Port block reference page for information about calculating the modeling frequencies.

Sample time (s)

Time interval between consecutive samples of the input signal.

Note The Input Port block does not automatically inherit a sample time from its input signal. The specified **Sample time (s)** value must match the sample time of the input signal.

Add noise

If you select this parameter, noise data in the RF physical blocks that are bracketed by the Input Port block and Output Port block is taken into consideration. If you do not select this parameter, noise data is ignored.

Initial seed

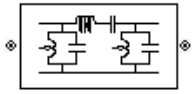
Nonnegative integer specifying the initial seed for the random number generator the block uses to generate noise. This parameter becomes visible if you select the **Add noise** parameter. If you specify the initial seed parameter with a variable, the initial seed changes each successive time you run a model.

See Also

Output Port

LC Bandpass Pi

Model LC bandpass pi network



Library

Ladder Filters sublibrary of the Physical library

Description

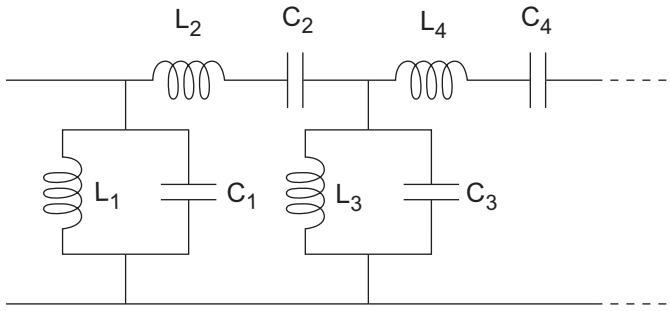
The LC Bandpass Pi block models the LC bandpass pi network described in the block dialog box, in terms of its frequency-dependent S-parameters.

For each inductor and capacitor pair in the network, the block first calculates the ABCD-parameters at each frequency contained in the vector of modeling frequencies. For each series pair, A = 1, B = Z, C = 0, and D = 1, where Z is the impedance of the series pair. For each shunt pair, A = 1, B = 0, C = Y, and D = 1, where Y is the admittance of the shunt pair.

The LC Bandpass Pi block then cascades the ABCD-parameters for each series and shunt pair at each of the modeling frequencies, and converts the cascaded parameters to S-parameters using the RF Toolbox `abcd2s` function.

See the Output Port block for information about determining the modeling frequencies.

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



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

Parameters

Main Tab

Inductance (H)

Vector containing the inductances, in order from source to load, of all inductors in the network. The inductance vector must contain at least three elements. All values must be strictly positive.

Capacitance (F)

Vector containing the capacitances, in order from source to load, of all capacitors in the network. Its length must be equal to the length of the vector you provide in the **Inductance** parameter. All values must be strictly positive.

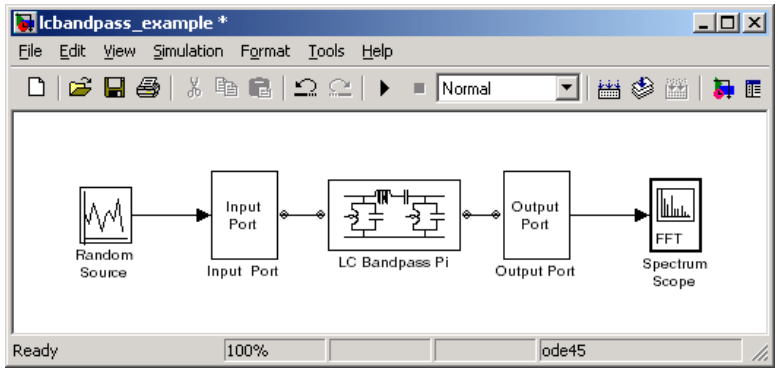
Visualization Tab

For information about plotting, see “Create Plots”.

Examples

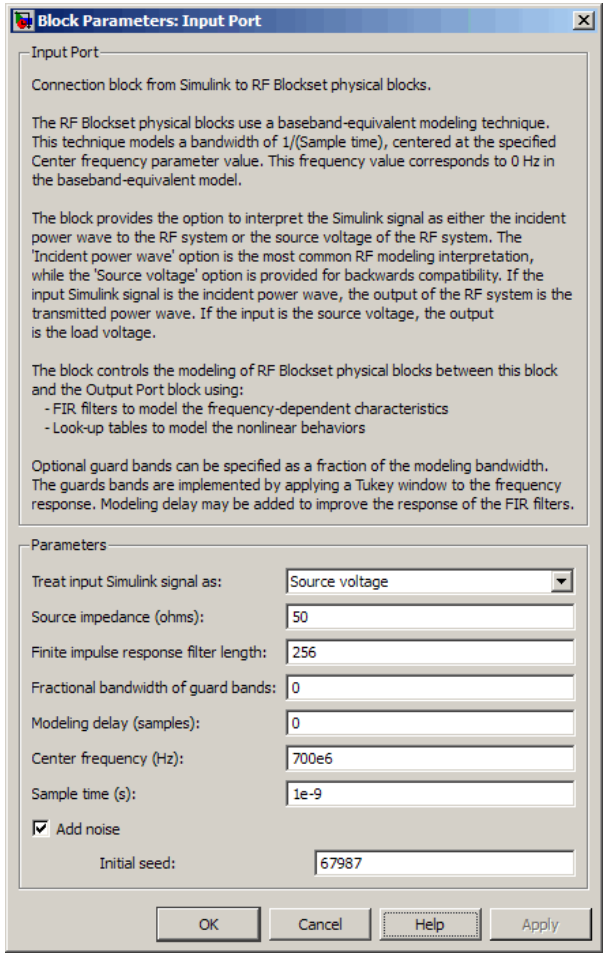
Using a Ladder Filter Block to Filter Gaussian Noise

This example provides complex random noise in Gaussian form as input to an LC Bandpass Pi block. A DSP System Toolbox Spectrum Scope block plots the filtered output.

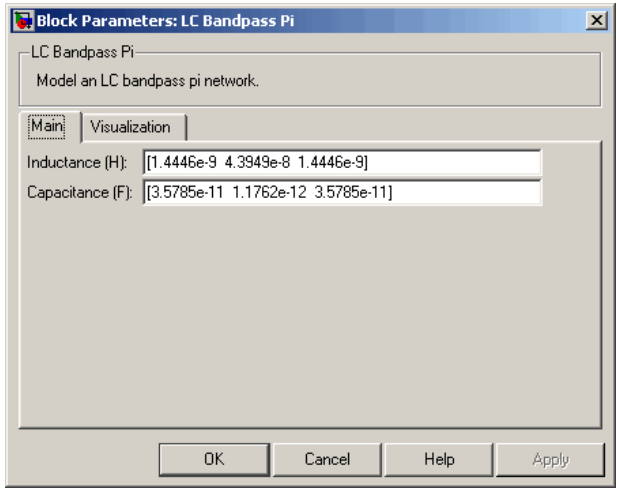


The DSP System Toolbox Random Source block produces frame-based output at 512 samples per frame. Its **Sample time** parameter is set to 1.0e-9. This sample time must match the sample time for the physical part of the model, which you provide in the Input Port block diagram.

The Input Port block specifies **Finite impulse response filter length** as 256, **Center frequency** as 700.0e6 Hz, **Sample time** as 1.0e-9, and **Source impedance** as 50 ohms.

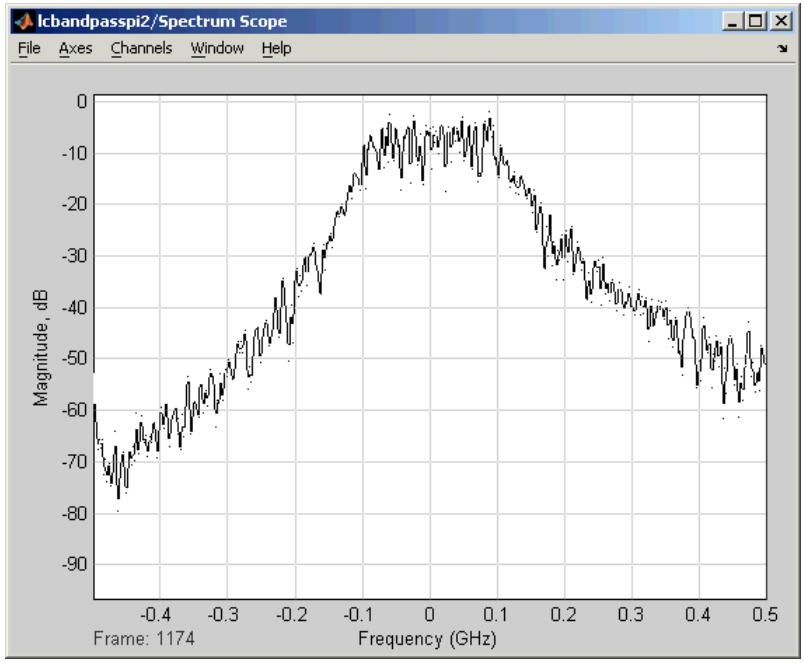


The LC Bandpass Pi block provides the inductances for three inductors, in order from source to load, [1.4446e-9, 4.3949e-8, 1.4446e-9]. Similarly, it provides the capacitances for three capacitors [3.5785e-11, 1.1762e-12, 3.5785e-11].



The following plot shows a sample of the baseband-equivalent RF signal generated by this LC Bandpass Pi block. Zero (0) on the frequency axis corresponds to the center frequency specified in the Input Port block. The bandwidth of the frequency spectrum is 1/sample time. You specify the **Sample time** parameter in the Input Port block.

The Axis Properties of the Spectrum Scope block have been adjusted to show the frequencies above and below the carrier. The **Minimum Y-limit** parameter is -90, and **Maximum Y-limit** is 0.



References

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

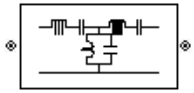
[2] Zverev, Anatol I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

General Passive Network, LC Bandpass Tee, LC Bandstop Pi, LC Bandstop Tee, LC Highpass Pi, LC Highpass Tee, LC Lowpass Pi, LC Lowpass Tee, Series C, Series L, Series R, Series RLC, Shunt C, Shunt L, Shunt R, Shunt RLC

LC Bandpass Tee

Model LC bandpass tee network



Library

Ladder Filters sublibrary of the Physical library

Description

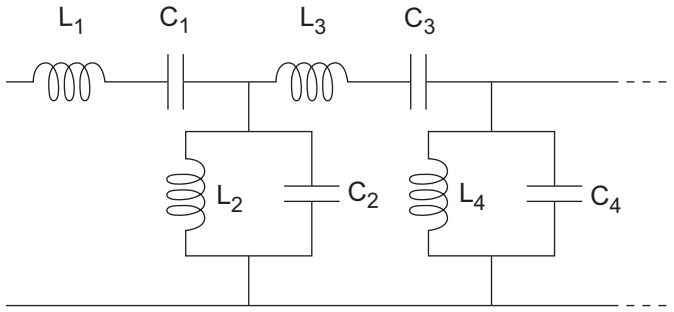
The LC Bandpass Tee block models the LC bandpass tee network described in the block dialog box, in terms of its frequency-dependent S-parameters.

For each inductor and capacitor pair in the network, the block first calculates the ABCD-parameters at each frequency contained in the vector of modeling frequencies. For each series pair, A = 1, B = Z, C = 0, and D = 1, where Z is the impedance of the series pair. For each shunt pair, A = 1, B = 0, C = Y, and D = 1, where Y is the admittance of the shunt pair.

The LC Bandpass Tee block then cascades the ABCD-parameters for each series and shunt pair at each of the modeling frequencies, and converts the cascaded parameters to S-parameters using the RF Toolbox `abcd2s` function.

See the Output Port block reference page for information about determining the modeling frequencies.

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



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

Parameters

Main Tab

Inductance (H)

Vector containing the inductances, in order from source to load, of all inductors in the network. The inductance vector must contain at least three elements. All values must be strictly positive.

Capacitance (F)

Vector containing the capacitances, in order from source to load, of all capacitors in the network. Its length must be equal to the length of the vector you provide in the **Inductance** parameter. All values must be strictly positive.

Visualization Tab

For information about plotting, see “Create Plots”.

Examples

See the LC Bandpass Pi block for an example of an LC filter.

References

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

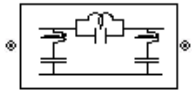
[2] Zverev, Anatol I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

General Passive Network, LC Bandpass Pi, LC Bandstop Pi, LC Bandstop Tee, LC Highpass Pi, LC Highpass Tee, LC Lowpass Pi, LC Lowpass Tee, Series C, Series L, Series R, Series RLC, Shunt C, Shunt L, Shunt R, Shunt RLC

LC Bandstop Pi

Model LC bandstop pi network



Library

Ladder Filters sublibrary of the Physical library

Description

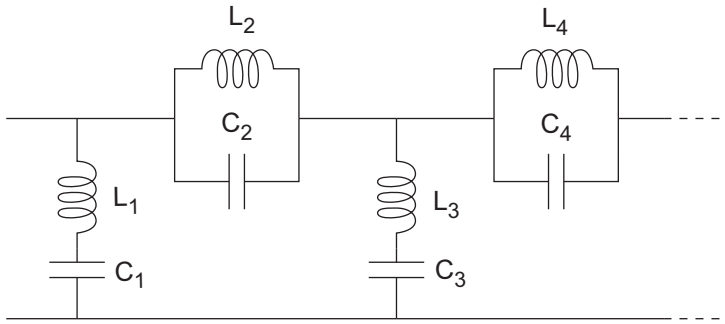
The LC Bandstop Pi block models the LC bandstop pi network described in the block dialog box, in terms of its frequency-dependent S-parameters.

For each inductor and capacitor pair in the network, the block first calculates the ABCD-parameters at each frequency contained in the vector of modeling frequencies. For each series pair, A = 1, B = *Z*, C = 0, and D = 1, where *Z* is the impedance of the series pair. For each shunt pair, A = 1, B = 0, C = *Y*, and D = 1, where *Y* is the admittance of the shunt pair.

The LC Bandstop Pi block then cascades the ABCD-parameters for each series and shunt pair at each of the modeling frequencies, and converts the cascaded parameters to S-parameters using the RF Toolbox `abcd2s` function.

See the Output Port block for information about determining the modeling frequencies.

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



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

Parameters

Main Tab

Inductance (H)

Vector containing the inductances, in order from source to load, of all inductors in the network. The inductance vector must contain at least three elements. All values must be strictly positive.

Capacitance (F)

Vector containing the capacitances, in order from source to load, of all capacitors in the network. Its length must be equal to the length of the vector you provide in the **Inductance** parameter. All values must be strictly positive.

Visualization Tab

For information about plotting, see “Create Plots”.

Examples

See the LC Bandpass Pi block for an example of an LC filter.

References

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

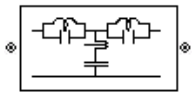
[2] Zverev, Anatol I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

General Passive Network, LC Bandpass Pi, LC Bandpass Tee, LC Bandstop Tee, LC Highpass Pi, LC Highpass Tee, LC Lowpass Pi, LC Lowpass Tee, Series C, Series L, Series R, Series RLC, Shunt C, Shunt L, Shunt R, Shunt RLC

LC Bandstop Tee

Model LC bandstop tee network



Library

Ladder Filters sublibrary of the Physical library

Description

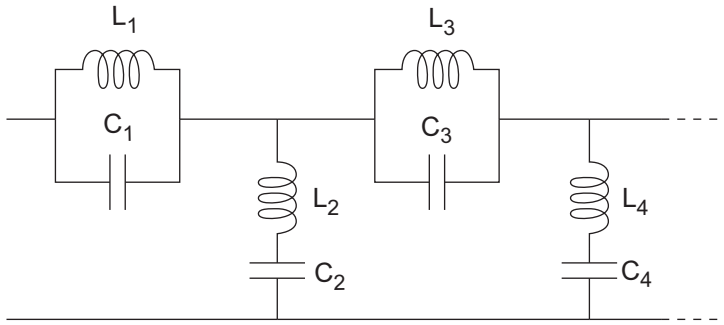
The LC Bandstop Tee block models the LC bandstop tee network described in the block dialog box, in terms of its frequency-dependent S-parameters.

For each inductor and capacitor pair in the network, the block first calculates the ABCD-parameters at each frequency contained in the vector of modeling frequencies. For each series pair, A = 1, B = Z, C = 0, and D = 1, where Z is the impedance of the series pair. For each shunt pair, A = 1, B = 0, C = Y, and D = 1, where Y is the admittance of the shunt pair.

The LC Bandstop Tee block then cascades the ABCD-parameters for each series and shunt pair at each of the modeling frequencies, and converts the cascaded parameters to S-parameters using the RF Toolbox `abcd2s` function.

See the Output Port block for information about determining the modeling frequencies.

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



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

Parameters

Main Tab

Inductance (H)

Vector containing the inductances, in order from source to load, of all inductors in the network. The inductance vector must contain at least three elements. All values must be strictly positive.

Capacitance (F)

Vector containing the capacitances, in order from source to load, of all capacitors in the network. Its length must be equal to the length of the vector you provide in the **Inductance** parameter. All values must be strictly positive.

Visualization Tab

For information about plotting, see “Create Plots”.

Examples

See the LC Bandpass Pi block for an example of an LC filter.

References

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

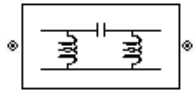
[2] Zverev, Anatol I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

General Passive Network, LC Bandpass Pi, LC Bandpass Tee, LC Bandstop Pi, LC Highpass Pi, LC Highpass Tee, LC Lowpass Pi, LC Lowpass Tee, Series C, Series L, Series R, Series RLC, Shunt C, Shunt L, Shunt R, Shunt RLC

LC Highpass Pi

Model LC highpass pi network



Library

Ladder Filters sublibrary of the Physical library

Description

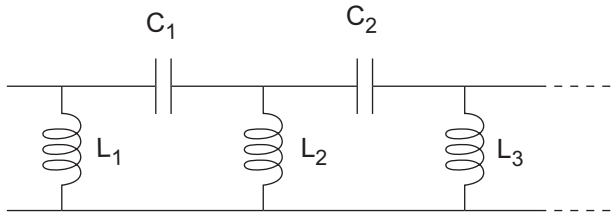
The LC Highpass Pi block models the LC highpass pi network described in the block dialog box, in terms of its frequency-dependent S-parameters.

For each inductor and capacitor in the network, the block first calculates the ABCD-parameters at each frequency contained in the vector of modeling frequencies. For each series circuit, A = 1, B = Z, C = 0, and D = 1, where Z is the impedance of the series circuit. For each shunt, A = 1, B = 0, C = Y, and D = 1, where Y is the admittance of the shunt circuit.

The LC Highpass Pi block then cascades the ABCD-parameters for each circuit element at each of the modeling frequencies, and converts the cascaded parameters to S-parameters using the RF Toolbox `abcd2s` function.

See the Output Port block reference page for information about determining the modeling frequencies.

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



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

Parameters

Main Tab

Inductance (H)

Vector containing the inductances, in order from source to load, of all inductors in the network. The inductance vector must contain at least two elements. All values must be strictly positive.

Capacitance (F)

Vector containing the capacitances, in order from source to load, of all capacitors in the network. Its length must be equal to or one less than the length of the vector you provide in the **Inductance** parameter. All values must be strictly positive.

Visualization Tab

For information about plotting, see “Create Plots”.

Examples

See the LC Bandpass Pi block for an example of an LC filter.

References

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

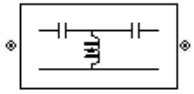
[2] Zverev, Anatol I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

General Passive Network, LC Bandpass Pi, LC Bandpass Tee, LC Bandstop Pi, LC Bandstop Tee, LC Highpass Tee, LC Lowpass Pi, LC Lowpass Tee, Series C, Series L, Series R, Series RLC, Shunt C, Shunt L, Shunt R, Shunt RLC

LC Highpass Tee

Model LC highpass tee network



Library

Ladder Filters sublibrary of the Physical library

Description

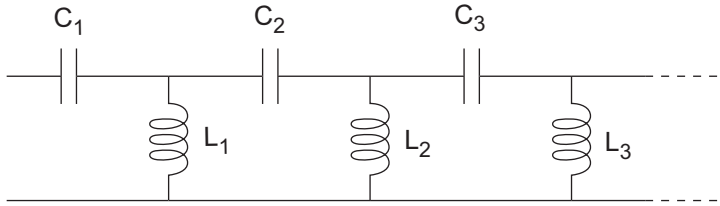
The LC Highpass Tee block models the LC highpass tee network described in the block dialog box, in terms of its frequency-dependent S-parameters.

For each inductor and capacitor in the network, the block first calculates the ABCD-parameters at each frequency contained in the vector of modeling frequencies. For each series circuit, A = 1, B = Z, C = 0, and D = 1, where Z is the impedance of the series circuit. For each shunt, A = 1, B = 0, C = Y, and D = 1, where Y is the admittance of the shunt circuit.

The LC Highpass Tee block then cascades the ABCD-parameters for each circuit element at each of the modeling frequencies, and converts the cascaded parameters to S-parameters using the RF Toolbox `abcd2s` function.

See the Output Port block reference page for information about determining the modeling frequencies.

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



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

Parameters

Main Tab

Inductance (H)

Vector containing the inductances, in order from source to load, of all inductors in the network. All values must be strictly positive. The vector cannot be empty.

Capacitance (F)

Vector containing the capacitances, in order from source to load, of all capacitors in the network. The capacitance vector must contain at least two elements. Its length must be equal to or one greater than the length of the vector you provide in the **Inductance** parameter. All values must be strictly positive.

Visualization Tab

For information about plotting, see “Create Plots”.

Examples

See the LC Bandpass Pi block for an example of an LC filter.

References

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

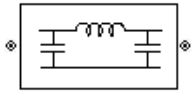
[2] Zverev, Anatol I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

General Passive Network, LC Bandpass Pi, LC Bandpass Tee, LC Bandstop Pi, LC Bandstop Tee, LC Highpass Pi, LC Lowpass Pi, LC Lowpass Tee, Series C, Series L, Series R, Series RLC, Shunt C, Shunt L, Shunt R, Shunt RLC

LC Lowpass Pi

Model LC lowpass pi network



Library

Ladder Filters sublibrary of the Physical library

Description

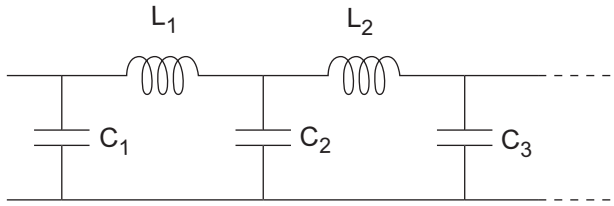
The LC Lowpass Pi block models the LC lowpass pi network described in the block dialog box, in terms of its frequency-dependent S-parameters.

For each inductor and capacitor in the network, the block first calculates the ABCD-parameters at each frequency contained in the vector of modeling frequencies. For each series circuit, A = 1, B = *Z*, C = 0, and D = 1, where *Z* is the impedance of the series circuit. For each shunt, A = 1, B = 0, C = *Y*, and D = 1, where *Y* is the admittance of the shunt circuit.

The LC Lowpass Pi block then cascades the ABCD-parameters for each circuit element at each of the modeling frequencies, and converts the cascaded parameters to S-parameters using the RF Toolbox `abcd2s` function.

See the Output Port block reference page for information about determining the modeling frequencies.

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



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

Parameters

Main Tab

Inductance (H)

Vector containing the inductances, in order from source to load, of all inductors in the network. All values must be strictly positive. The vector cannot be empty.

Capacitance (F)

Vector containing the capacitances, in order from source to load, of all capacitors in the network. The capacitance vector must contain at least two elements. Its length must be equal to or one greater than the length of the vector you provide in the **Inductance** parameter. All values must be strictly positive.

Visualization Tab

For information about plotting, see “Create Plots”.

Examples

See the LC Bandpass Pi block for an example of an LC filter.

References

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

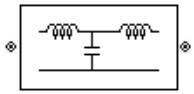
[2] Zverev, Anatol I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

General Passive Network, LC Bandpass Pi, LC Bandpass Tee, LC Bandstop Pi, LC Bandstop Tee, LC Highpass Pi, LC Highpass Tee, LC Lowpass Tee, Series C, Series L, Series R, Series RLC, Shunt C, Shunt L, Shunt R, Shunt RLC

LC Lowpass Tee

Model LC lowpass tee network



Library

Ladders Filters sublibrary of the Physical library

Description

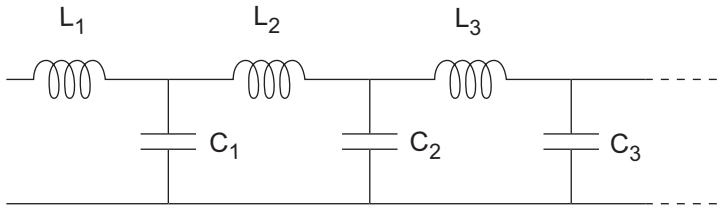
The LC Lowpass Tee block models the LC lowpass tee network described in the block dialog box in terms of its frequency-dependent S-parameters.

For each inductor and capacitor in the network, the block first calculates the ABCD-parameters at each frequency contained in the vector of modeling frequencies. For each series circuit, A = 1, B = Z, C = 0, and D = 1, where Z is the impedance of the series circuit. For each shunt, A = 1, B = 0, C = Y, and D = 1, where Y is the admittance of the shunt circuit.

The LC Lowpass Tee block then cascades the ABCD-parameters for each circuit element at each of the modeling frequencies, and converts the cascaded parameters to S-parameters using the RF Toolbox `abcd2s` function.

See the Output Port block reference page for information about determining the modeling frequencies.

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



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

Parameters

Main Tab

Inductance (H)

Vector containing the inductances, in order from source to load, of all inductors in the network. The inductance vector must contain at least two elements. All values must be strictly positive.

Capacitance (F)

Vector containing the capacitances, in order from source to load, of all capacitors in the network. Its length must be equal to or one less than the length of the vector you provide in the **Inductance** parameter. All values must be strictly positive.

Visualization Tab

For information about plotting, see “Create Plots”.

Examples

See the LC Bandpass Pi block for an example of an LC filter.

References

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

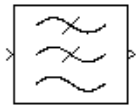
[2] Zverev, Anatol I., *Handbook of Filter Synthesis*, John Wiley & Sons, 1967.

See Also

General Passive Network, LC Bandpass Pi, LC Bandpass Tee, LC Bandstop Pi, LC Bandstop Tee, LC Highpass Pi, LC Highpass Tee, LC Lowpass Pi, Series C, Series L, Series R, Series RLC, Shunt C, Shunt L, Shunt R, Shunt RLC

Lowpass RF Filter

Standard lowpass RF filters in baseband-equivalent complex form



Library

Mathematical

Note To use this block, you must install DSP System Toolbox software. For more information, see the RF Blockset release notes.

Description

The Lowpass RF Filter block lets you design standard analog lowpass filters, implemented in baseband-equivalent complex form. The following table describes the available design methods.

Design Method	Description
`Butterworth`	The magnitude response of a Butterworth filter is maximally flat in the passband and monotonic overall.
`Chebyshev I`	The magnitude response of a Chebyshev I filter is equiripple in the passband and monotonic in the stopband.
`Chebyshev II`	The magnitude response of a Chebyshev II filter is monotonic in the passband and equiripple in the stopband.
`Elliptic`	The magnitude response of an elliptic filter is equiripple in both the passband and the stopband.
`Bessel`	The delay of a Bessel filter is maximally flat in the passband.

The block input must be a discrete-time complex signal.

Note This block assumes a nominal impedance of 1 ohm.

Select the design of the filter from the **Design method** list in the dialog box. For each design method, the block enables you to specify the filter design parameters shown in the following table.

Design Method	Filter Design Parameters
`Butterworth`	Order, passband edge frequency
`Chebyshev I`	Order, passband edge frequency, passband ripple
`Chebyshev II`	Order, stopband edge frequency, stopband attenuation
`Elliptic`	Order, passband edge frequency, passband ripple, stopband attenuation
`Bessel`	Order, passband edge frequency

The Lowpass RF Filter block designs the filters using the Signal Processing Toolbox filter design functions `buttap`, `cheb1ap`, `cheb2ap`, `ellipap`, and `besselap`.

Note Some RF blocks require the sample time to perform baseband modeling calculations. To ensure the accuracy of these calculations, the Input Port block, as well as the mathematical RF blocks, compare the input sample time to the sample time you provide in the mask. If they do not match, or if the input sample time is missing because the blocks are not connected, an error message appears.

Parameters

The parameters displayed in the dialog box vary for different design methods. Only some of these parameters are visible in the dialog box at any one time.

Parameters that are tunable can be changed while the model is running.

Design method

Filter design method. The design method can be `Butterworth`, `Chebyshev I`, `Chebyshev II`, `Elliptic`, or `Bessel`. Tunable.

Filter order

Order of the filter.

Passband edge frequency (Hz)

Passband edge frequency for Butterworth, Chebyshev I, elliptic, and Bessel designs. Tunable.

Stopband edge frequency (Hz)

Stopband edge frequency for Chebyshev II designs. Tunable.

Passband ripple in dB

Passband ripple for Chebyshev I and elliptic designs. Tunable.

Stopband attenuation in dB

Stopband attenuation for Chebyshev II and elliptic designs. Tunable.

Finite impulse response filter length

Desired length of the baseband-equivalent impulse response for the filter.

Center frequency (Hz)

Center of the modeling frequencies.

Sample time (s)

Time interval between consecutive samples of the input signal.

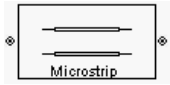
See Also

Amplifier, Bandpass RF Filter, Bandstop RF Filter, Highpass RF Filter, Mixer

`buttap`, `cheb1ap`, `cheb2ap`, `ellipap`, `besselap` (Signal Processing Toolbox)

Microstrip Transmission Line

Model microstrip transmission line

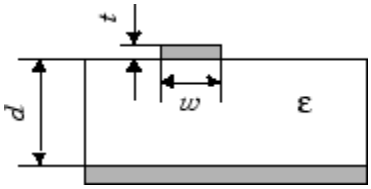


Library

Transmission Lines sublibrary of the Physical library

Description

The Microstrip Transmission Line block models the microstrip transmission line described in the block dialog in terms of its frequency-dependent S-parameters. 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).



The block lets you model the transmission line as a stub or as a stubless line.

Stubless Transmission Line

If you model a microstrip transmission line as a stubless line, the Microstrip Transmission Line block 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 block calculates the ABCD-parameters using the physical length of the transmission line, d, and the complex propagation constant, k, using the following equations:

$$A = \frac{e^{kd} + e^{-kd}}{2}$$

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

$$C = \frac{e^{kd} - e^{-kd}}{2 * Z_0}$$

$$D = \frac{e^{kd} + e^{-kd}}{2}$$

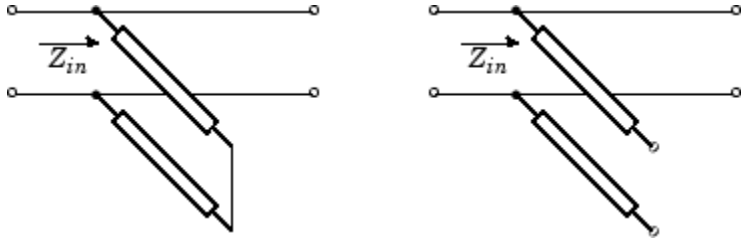
Z_0 and k are vectors whose elements correspond to the elements of f, a vector of modeling frequencies. Both can be expressed in terms of the specified conductor strip width, substrate height, conductor strip thickness, relative permittivity constant, conductivity, and dielectric loss tangent of the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

Shunt and Series Stubs

If you model the transmission line as a shunt or series stub, the Microstrip Transmission Line block first calculates the ABCD-parameters at each frequency contained in the vector of modeling frequencies. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

Shunt ABCD-Parameters

When you set the **Stub mode** parameter in the mask dialog box to `Shunt`, the two-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown here.

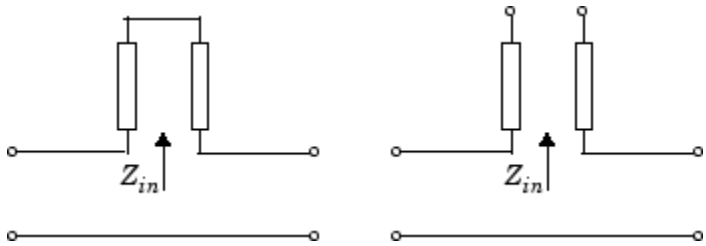


Z_{in} is the input impedance of the shunt circuit. The ABCD-parameters for the shunt stub are calculated as

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

Series ABCD-Parameters

When you set the **Stub mode** parameter in the mask dialog box to `Series`, the two-port network consists of a series transmission line that you can terminate with either a short circuit or an open circuit as shown here.



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

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

Parameters

Main Tab

Strip width (m)

Width of the microstrip transmission line.

Substrate height (m)

Thickness of the dielectric on which the microstrip resides.

Strip thickness (m)

Physical thickness of the microstrip.

Relative permittivity constant

Relative permittivity of the dielectric expressed as the ratio of the permittivity of the dielectric to permittivity in free space ε_0.

Loss tangent in dielectric

Loss angle tangent of the dielectric.

Conductivity in conductor (S/m)

Conductivity of the conductor in siemens per meter.

Transmission line length (m)

Physical length of the transmission line.

Stub mode

Type of stub. Choices are `Not a stub`, `Shunt`, or `Series`.

Termination of stub

Stub termination for stub modes `Shunt` and `Series`. Choices are `Open` or `Short`. This parameter becomes visible only when **Stub mode** is set to `Shunt` or `Series`.

Visualization Tab

For information about plotting, see “Create Plots”.

References

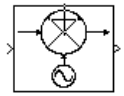
[1] Gupta, K.C., G. Ramesh, I. Bahl, and P. Bhartia, *Microstrip Lines and Slotlines*, Second Edition, Artech House, 1996. pp. 102-109.

See Also

Coaxial Transmission Line, Coplanar Waveguide Transmission Line, General Passive Network, Transmission Line, Parallel-Plate Transmission Line, Two-Wire Transmission Line

Mixer (Idealized Baseband)

Complex baseband model of mixer and local oscillator with phase noise



Library

Mathematical

Description

The Mixer block generates a complex baseband model of the following:

- A mixer
- A local oscillator with phase noise whose spectrum is characterized by a 1/*f* slope

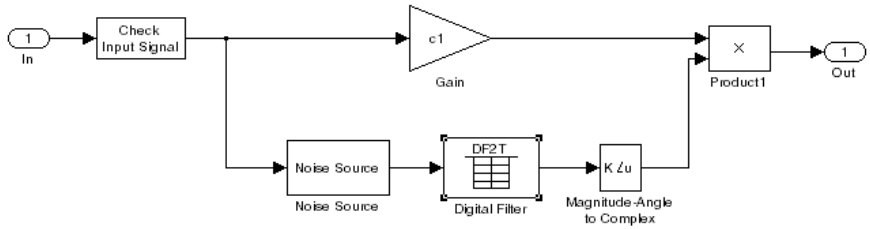
The Mixer block includes both the IF and RF signals as complex-baseband equivalent signals. Both the IF and RF center frequencies are represented as 0 hertz. The amplitude of the noise spectrum is specified by the noise power contained in a 1-hertz bandwidth offset from the carrier by a specified frequency.

Note This block assumes a nominal impedance of 1 ohm.

The block applies the phase noise to the signal as follows:

1. Generates additive white Gaussian noise (AWGN) and filters it with a digital filter.
2. Adds the resulting phase noise to the angle component of the input signal.

You can view the block's implementation of phase noise by right-clicking the block and selecting **Look under mask** from the pop-up menu. The following figure shows the implementation.



You can view the construction of the Noise Source subsystem by double-clicking it.

Parameters

You can change parameters that are marked as tunable in the following descriptions while the model is running.

Conversion gain (dB)

Scalar specifying the conversion gain for the mixer. Use a negative value to specify loss. Tunable.

Phase noise level (dBc/Hz)

Scalar specifying the phase noise level in decibels relative to the carrier, per hertz. Tunable.

Frequency offset (Hz)

Scalar specifying the frequency offset. Tunable.

Initial seed

Nonnegative integer specifying the initial seed for the random number generator the block uses to generate noise.

References

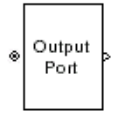
[1] Kasdin, N.J., “Discrete Simulation of Colored Noise and Stochastic Processes and 1/(f^alpha); Power Law Noise Generation,” The Proceedings of the IEEE, May, 1995, Vol. 83, No. 5.

See Also

Amplifier, Bandpass RF Filter, Bandstop RF Filter, Highpass RF Filter, Lowpass RF Filter

Output Port

Connection block from RF physical blocks to Simulink environment



Library

Input/Output Ports sublibrary of the Physical library

Description

The Output Port block produces the baseband-equivalent time-domain response of an input signal traveling through a series of RF physical components. The Output Port block

1. Partitions the RF physical components into linear and nonlinear subsystems.
2. Extracts the complex impulse response of the linear subsystem for baseband-equivalent modeling of the RF linear system.
3. Extracts the nonlinear AMAM/AMPM modeling for RF nonlinearity.

The Output Port block also serves as a connecting port from an RF physical part of the model to the Simulink, or mathematical, part of the model. For more information about how the Output Port block converts the physical modeling environment signals to mathematical Simulink signals, see “Convert to and from Simulink Signals”.

Note Some RF blocks require the sample time to perform baseband modeling calculations. To ensure the accuracy of these calculations, the Input Port block, as well as the mathematical RF blocks, compare the input sample time to the sample time you provide in the mask. If they do not match, or if the input sample time is missing because the blocks are not connected, an error message appears.

Linear Subsystem

For the linear subsystem, the Output Port block uses the Input Port block parameters and the interpolated S-parameters calculated by each of the cascaded physical blocks to calculate the baseband-equivalent impulse response. Specifically, it

1. Determines the modeling frequencies f as an N-element vector. The modeling frequencies are a function of the center frequency f_c, the sample time t_s, and the finite impulse response filter length N, all of which you specify in the Input Port block dialog box.

   The nth element of f, f_n, is given by

   $$f_n = f_{\min} + \frac{n-1}{t_s N} \quad n = 1, \ldots, N$$

   where

   $$f_{\min} = f_c - \frac{1}{2t_s}$$

2. Calculates the passband transfer function for the frequency range as

   $$H(f) = \frac{V_L(f)}{V_S(f)}$$

   where V_S and V_L are the source and load voltages, and f represents the modeling frequencies. More specifically,

   $$H(f) = \frac{S_{21}(1+\Gamma_l)(1-\Gamma_s)}{2(1-S_{22}\Gamma_l)(1-\Gamma_{in}\Gamma_s)}$$

   where

   $$\Gamma_l = \frac{Z_l - Z_o}{Z_l + Z_o}$$

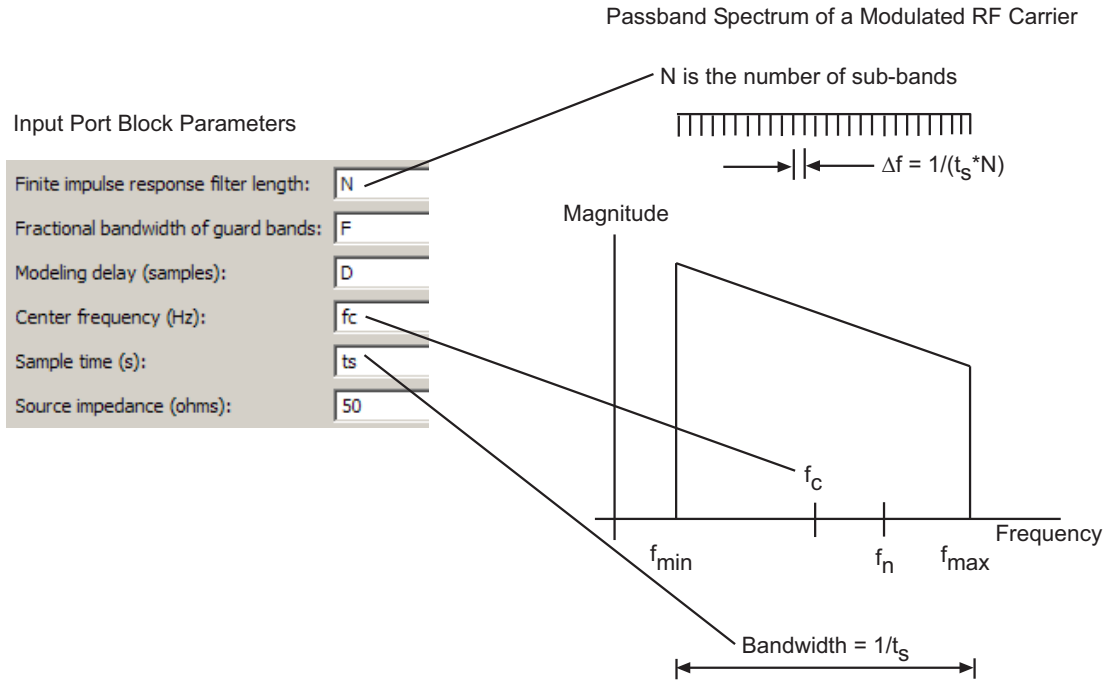
   $$\Gamma_s = \frac{Z_s - Z_o}{Z_s + Z_o}$$

   $$\Gamma_{in} = S_{11} + \left(S_{12}S_{21}\frac{\Gamma_l}{(1-S_{22}\Gamma_l)}\right)$$

   and

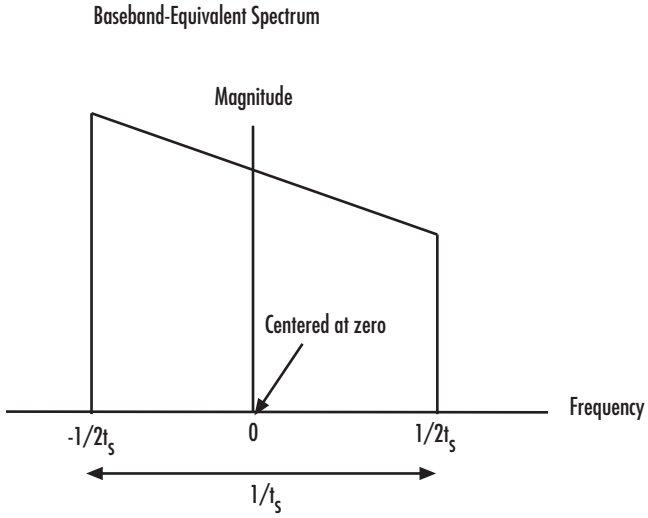
- Z_S is the source impedance.
- Z_L is the load impedance.
- S_{ij} are the S-parameters of a two-port network.

The blockset derives the passband transfer function from the Input Port block parameters as shown in the following figure:



3 Translates the passband transfer function to baseband as $H(f - f_c)$, where f_c is the specified center frequency.

The baseband transfer function is shown in the following figure.

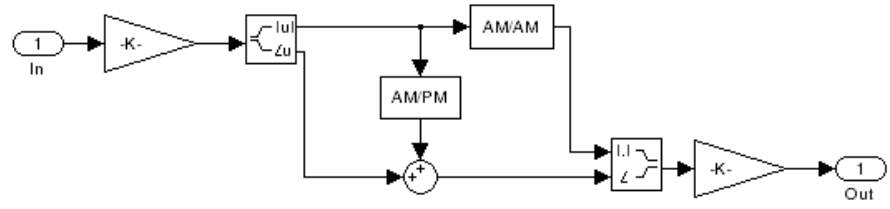


4 Obtains the baseband-equivalent impulse response by calculating the inverse FFT of the baseband transfer function. For faster simulation, the block calculates the IFFT using the next power of 2 greater than the specified finite impulse response filter length. Then, it truncates the impulse response to a length equal to the filter length specified.

For the linear subsystem, the Output Port block uses the calculated impulse response as input to the DSP System Toolbox Digital Filter Design block to determine the output.

Nonlinear Subsystem

The nonlinear subsystem is implemented by AM/AM and AM/PM nonlinear models, as shown in the following figure.



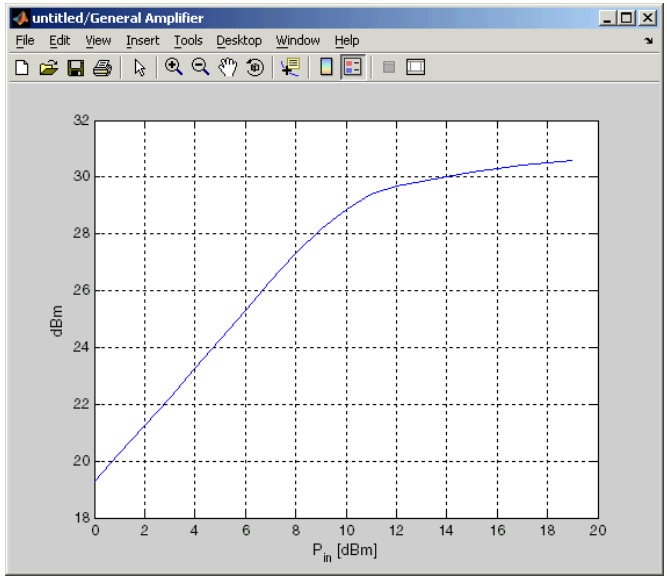
The nonlinearities of AM/AM and AM/PM conversions are extracted from the power data of an amplifier or mixer by the equations

$$AM_{out} = \sqrt{R_l P_{out}}$$
$$PM_{out} = \varphi$$
$$AM_{in} = \sqrt{R_s P_{in}}$$

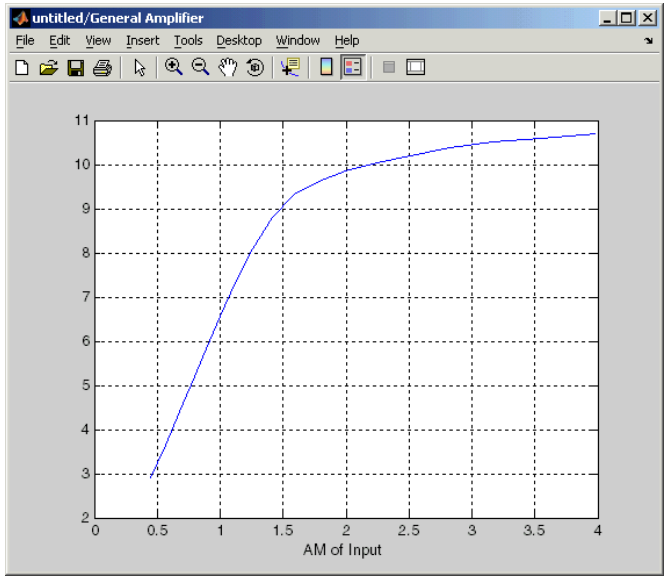
where AM_{in} is the AM of the input voltage, AM_{out} and PM_{out} are the AM and PM of the output voltage, R_s is the source resistance (50 ohms), R_l is the load resistance (50 ohms), P_{in} is the input power, P_{out} is the output power, andϕ is the phase shift between the input and output voltage.

Note You can provide power data via a `.amp` file. See “AMP File Data Sections” (RF Toolbox) for information about this format.

The following figure shows the original power data of an amplifier.



This figure shows the extracted AM/AM nonlinear conversion.



Parameters

Main Tab

Load impedance (ohms)

Load impedance of the RF network described in the physical model to which it connects.

Visualization Tab

This tab shows parameters for creating plots if you display the Output Port mask after you perform one or more of the following actions:

- Run a model with two or more blocks between the Input Port block and the Output Port block.
- Click the Update Diagram button to initialize a model with two or more blocks between the Input Port block and the Output Port block.

For information about plotting, see “Create Plots”.

See Also

Input Port

`s2y` (RF Toolbox)

Parallel-Plate Transmission Line

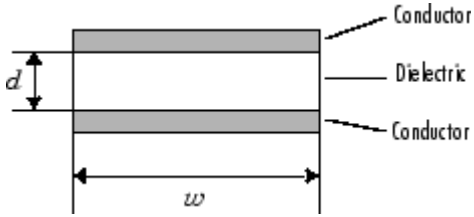
Model parallel-plate transmission line

Library

Transmission Lines sublibrary of the Physical library

Description 

The Parallel-Plate Transmission Line block models the parallel-plate transmission line described in the block dialog box in terms of its frequency-dependent S-parameters. 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*.



The block lets you model the transmission line as a stub or as a stubless line.

Stubless Transmission Line

If you model a parallel-plate transmission line as a stubless line, the Parallel-Plate Transmission Line block 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 block 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, a vector of modeling frequencies. Both can be expressed in terms of the resistance (R), inductance (L), conductance (G), and capacitance (C) per unit length (meters) as follows:

$$Z_0 = \sqrt{\frac{R + j\omega L}{G + j\omega C}}$$

$$k = k_r + jk_i = \sqrt{(R + j\omega L)(G + j\omega C)}$$

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:

- 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 **Relative permittivity constant** parameter value.

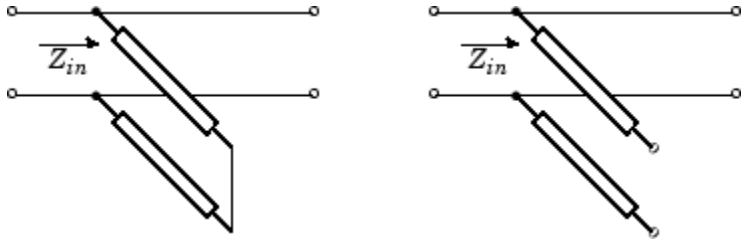
  - tan δ is the **Loss tangent of dielectric** parameter value.
- δ_{cond} is the skin depth of the conductor, which the block calculates as $1/\sqrt{\pi f \mu \sigma_{cond}}$.
- f is a vector of modeling frequencies determined by the Output Port block.

Shunt and Series Stubs

If you model the transmission line as a shunt or series stub, the Parallel-Plate Transmission Line block first calculates the ABCD-parameters at each frequency contained in the vector of modeling frequencies. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

Shunt ABCD-Parameters

When you set the **Stub mode** parameter in the mask dialog box to `Shunt`, the two-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown here.

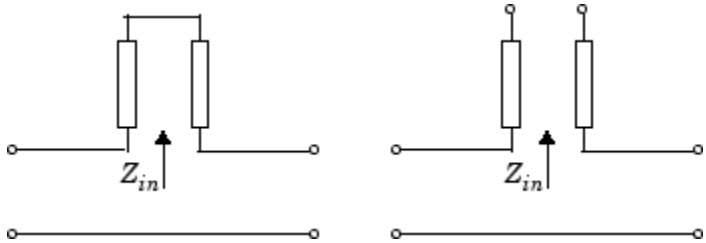


Z_{in} is the input impedance of the shunt circuit. The ABCD-parameters for the shunt stub are calculated as

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

Series ABCD-Parameters

When you set the **Stub mode** parameter in the mask dialog box to `Series`, the two-port network consists of a series transmission line that you can terminate with either a short circuit or an open circuit as shown here.



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

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

Parameters

Main Tab

Plate width (m)

Physical width of the parallel-plate transmission line.

Plate separation (m)

Thickness of the dielectric separating the plates.

Relative permeability constant

Relative permeability of the dielectric expressed as the ratio of the permeability of the dielectric to permeability in free space μ_0.

Relative permittivity constant

Relative permittivity of the dielectric expressed as the ratio of the permittivity of the dielectric to permittivity in free space ε_0.

Loss tangent of dielectric

Loss angle tangent of the dielectric.

Conductivity of conductor (S/m)

Conductivity of the conductor in siemens per meter.

Transmission line length (m)

Physical length of the transmission line.

Stub mode

Type of stub. Choices are `Not a stub`, `Shunt`, or `Series`.

Termination of stub

Stub termination for stub modes `Shunt` and `Series`. Choices are `Open` or `Short`. This parameter becomes visible only when **Stub mode** is set to `Shunt` or `Series`.

Visualization Tab

For information about plotting, see “Create Plots”.

References

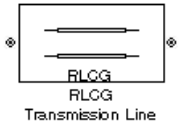
[1] Pozar, David M. *Microwave Engineering*, John Wiley & Sons, Inc., 2005.

See Also

Coaxial Transmission Line, Coplanar Waveguide Transmission Line, General Passive Network, Transmission Line, Microstrip Transmission Line, Two-Wire Transmission Line

RLCG Transmission Line

Model RLCG transmission line

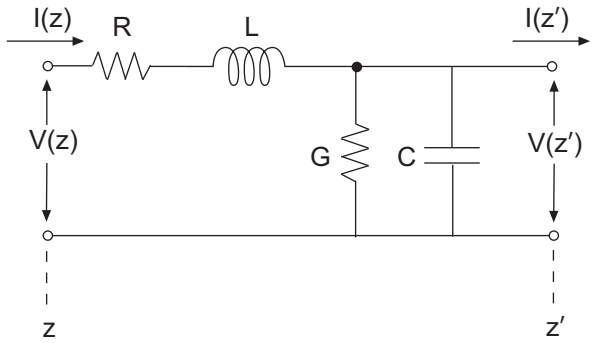


Library

Transmission Lines sublibrary of the Physical library

Description

The RLCG Transmission Line block models the RLCG transmission line described in the block dialog box in terms of its frequency-dependent resistance, inductance, capacitance, and conductance. The transmission line, which can be lossy or lossless, is treated as a two-port linear network.



where $z' = z + \Delta z$.

The block lets you model the transmission line as a stub or as a stubless line.

Stubless Transmission Line

If you model an RLCG transmission line as a stubless line, the RLCG Transmission Line block 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 block 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, a vector of modeling frequencies. Both can be expressed in terms of the resistance (R), inductance (L), conductance (G), and capacitance (C) per unit length (meters) as follows:

$$Z_0 = \sqrt{\frac{R + j\omega L}{G + j\omega C}}$$

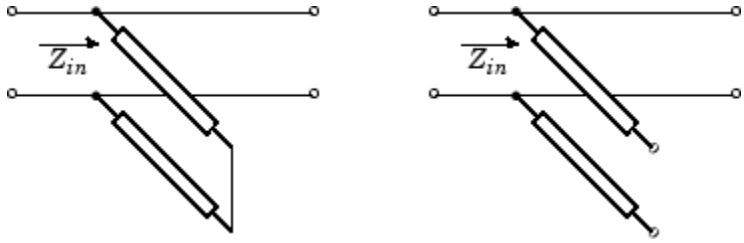
$$k = k_r + jk_i = \sqrt{(R + j\omega L)(G + j\omega C)}$$

Shunt and Series Stubs

If you model the transmission line as a shunt or series stub, the RLCG Transmission Line block first calculates the ABCD-parameters at each frequency contained in the vector of modeling frequencies. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

Shunt ABCD-Parameters

When you set the **Stub mode** parameter in the mask dialog box to `Shunt`, the two-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown here.

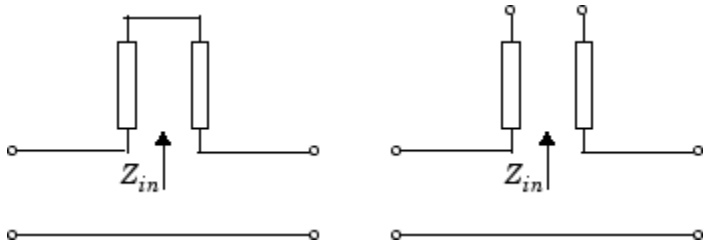


Z_{in} is the input impedance of the shunt circuit. The ABCD-parameters for the shunt stub are calculated as

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

Series ABCD-Parameters

When you set the **Stub mode** parameter in the mask dialog box to `Series`, the two-port network consists of a series transmission line that you can terminate with either a short circuit or an open circuit as shown here.



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

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

Parameters

Main Tab

Resistance per length (ohms/m)

Vector of resistance values in ohms per meter.

Inductance per length (H/m)

Vector of inductance values in henries per meter.

Capacitance per length (F/m)

Vector of capacitance values in farads per meter.

Conductance per length (S/m)

Vector of conductance values in siemens per meter.

Frequency (Hz)

Vector of frequency values at which the resistance, inductance, capacitance, and conductance values are known.

Interpolation method

Specify the interpolation method the block uses to calculate the parameter values at the modeling frequencies. Your choices are `Linear`, `Spline`, or `Cubic`.

Transmission line length (m)

Physical length of the transmission line.

Stub mode

Type of stub. Your choices are `Not a stub`, `Shunt`, or `Series`.

Termination of stub

Stub termination for stub modes `Shunt` and `Series`. Choices are `Open` or `Short`. This parameter becomes visible only when **Stub mode** is set to `Shunt` or `Series`.

Visualization Tab

For information about plotting, see “Create Plots”.

References

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

See Also

Coaxial Transmission Line, Coplanar Waveguide Transmission Line, General Passive Network, Parallel-Plate Transmission Line, Transmission Line, Microstrip Transmission Line, Two-Wire Transmission Line

S-Parameters Amplifier

Model nonlinear amplifier using S-parameters

Library

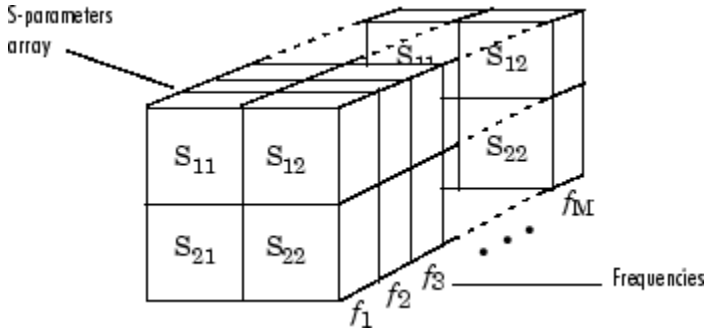
Amplifiers sublibrary of the Physical library

Description

The S-Parameters Amplifier block models the nonlinear amplifier described in the block dialog box, in terms of its frequency-dependent S-parameters, the frequencies and reference impedance of the S-parameters, noise data, and nonlinearity data.

Network Parameters

In the **S-parameters** field of the block dialog box, provide the S-parameters for each of M frequencies as a 2-by-2-by-M array. In the **Frequency** field, specify the frequencies for the S-parameters as an M-element vector. The elements of the frequencies vector must be in the same order as the S-parameters. All frequencies must be positive. For example, the following figure shows the correspondence between the S-parameters array and the vector of frequencies.



The S-Parameters Amplifier block interpolates the given S-parameters to determine their values at the modeling frequencies. See “Map Network Parameters to Modeling Frequencies” for more details.

Nonlinearity

You can introduce nonlinearities into your model by specifying parameters in the **Nonlinearity Data** tab of the S-Parameters Amplifier block dialog box. Depending on which of these parameters you specify, the block computes up to four of the coefficients c_1, c_3, c_5, and c_7 of the polynomial

$$F_{AM/AM}(s) = c_1 s + c_3|s|^2 s + c_5|s|^4 s + c_7|s|^6 s$$

that determines the AM/AM conversion for the input signal s. The block automatically calculates c_1, the linear gain term. If you do not specify additional nonlinearity data, the block operates as a linear amplifier. If you do, the block calculates one or more of the remaining coefficients as the solution to a system of linear equations, determined by the following method.

1. The block checks whether you have specified a value other than `Inf` for:

   - The third-order intercept point ($OIP3$ or $IIP3$).
   - The output power at the 1-dB compression point ($P_{1dB,\,out}$).
   - The output power at saturation ($P_{sat,\,out}$).

   In addition, if you have specified $P_{sat,\,out}$, the block uses the value for the gain compression at saturation (GC_{sat}). Otherwise, GC_{sat} is not used. You define each of these parameters in the block dialog box, on the **Nonlinearity Data** tab.

2. The block calculates a corresponding input or output value for the parameters you have specified. In units of dB and dBm,

   $$\begin{aligned} P_{sat,\,out} + GC_{sat} &= P_{sat,\,in} + G_{lin} \\ P_{1dB,\,out} + 1 &= P_{1dB,\,in} + G_{lin} \\ OIP3 &= IIP3 + G_{lin} \end{aligned}$$

   where G_{lin} is c_1 in units of dB.

3. The block formulates the coefficients c_3, c_5, and c_7, where applicable, as the solutions to a system of one, two, or three linear equations. The number of equations used is

equal to the number of parameters you provide. For example, if you specify all three parameters, the block formulates the coefficients according to the following equations:

$$\sqrt{P_{sat,out}} = c_1\sqrt{P_{sat,in}} + c_3\left(\sqrt{P_{sat,in}}\right)^3 + c_5\left(\sqrt{P_{sat,in}}\right)^5 + c_7\left(\sqrt{P_{sat,in}}\right)^7$$

$$\sqrt{P_{1dB,out}} = c_1\sqrt{P_{1dB,in}} + c_3\left(\sqrt{P_{1dB,in}}\right)^3 + c_5\left(\sqrt{P_{1dB,in}}\right)^5 + c_7\left(\sqrt{P_{1dB,in}}\right)^7$$

$$0 = \frac{c_1}{IIP3} + c_3$$

The first two equations are the evaluation of the polynomial $F_{AM/AM}(s)$ at the points $(\sqrt{P_{sat,in}}, \sqrt{P_{sat,out}})$ and $(\sqrt{P_{1dB,in}}, \sqrt{P_{1dB,out}})$, expressed in linear units (such as W or mW) and normalized to a 1-Ω impedance. The third equation is the definition of the third-order intercept point.

The calculation omits higher-order terms according to the available degrees of freedom of the system. If you specify only two of the three parameters, the block does not use the equation involving the parameter you did not specify, and eliminates any c_7 terms from the remaining equations. Similarly, if you provide only one of the parameters, the block uses only the solution to the equation involving that parameter and omits any c_5 or c_7 terms.

If you provide vectors of nonlinearity and frequency data, the block calculates the polynomial coefficients using values for the parameters interpolated at the center frequency.

Active Noise

You can specify active block noise in one of the following ways:

- Spot noise data in the S-Parameters Amplifier block dialog box.
- Noise figure, noise factor, or noise temperature value in the S-Parameters Amplifier block dialog box.

If you specify block noise as spot noise data, the block uses the data to calculate noise figure. The block first interpolates the noise data for the modeling frequencies, using the specified **Interpolation method**. It then calculates the noise figure using the resulting values.

Parameters

Main Tab

S-Parameters

S-parameters for a nonlinear amplifier in a 2-by-2-by-M array. M is the number of S-parameters.

Frequency (Hz)

Frequencies of the S-parameters as an M-element vector. The order of the frequencies must correspond to the order of the S-parameters in **S-Parameters**. All frequencies must be positive.

Reference impedance (ohms)

Reference impedance of the S-parameters as a scalar or a vector of length M. The value of this parameter can be real or complex. If you provide a scalar value, then that value is applied to all frequencies.

Interpolation method

The method used to interpolate the network parameters. The following table lists the available methods describes each one.

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

Noise Data Tab

Noise type

Type of noise data. The value can be `Noise figure`, `Spot noise data`, `Noise factor`, or `Noise temperature`. This parameter is disabled if the data source contains noise data.

Noise figure (dB)

Scalar ratio or vector of ratios, in decibels, of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Noise figure`.

Minimum noise figure (dB)

Minimum scalar ratio or vector of minimum ratios of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Spot noise data`.

Optimal reflection coefficient

Optimal amplifier source impedance. This parameter is enabled if **Noise type** is set to `Spot noise data`. The value can be a scalar or vector.

Equivalent normalized resistance

Resistance or vector of resistances normalized to the resistance value or values used to take the noise measurement. This parameter is enabled if **Noise type** is set to `Spot noise data`.

Noise factor

Scalar ratio or vector of ratios of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Noise factor`.

Noise temperature (K)

Equivalent temperature or vector of temperatures that produce the same amount of noise power as the amplifier. This parameter is enabled if **Noise type** is set to `Noise temperature`.

Frequency (Hz)

Scalar value or vector corresponding to the domain of frequencies over which you are specifying the noise data. If you provide a scalar value for your noise data, the block ignores the **Frequency (Hz)** parameter and uses the noise data for all frequencies. If you provide a vector of values for your noise data, it must be the same size as the vector of frequencies. The block uses the **Interpolation method** specified in the **Main** tab to interpolate noise data.

Nonlinearity Data Tab

IP3 type

Type of third-order intercept point. The value can be `IIP3` (input intercept point) or `OIP3` (output intercept point). This parameter is disabled if the data source contains power data or IP3 data.

IP3 (dBm)

Value of third-order intercept point. This parameter is disabled if the data source contains power data or IP3 data. Use the default value, `Inf`, if you do not know the

IP3 value. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

1 dB gain compression power (dBm)

Output power value ($P_{1dB,\,out}$) at which gain has decreased by 1 dB. This parameter is disabled if the data source contains power data or 1-dB compression point data. Use the default value, `Inf`, if you do not know the 1-dB compression point. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Output saturation power (dBm)

Output power value ($P_{sat,\,out}$) that the amplifier produces when fully saturated. This parameter is disabled if the data source contains output saturation power data. Use the default value, `Inf`, if you do not know the saturation power. If you specify this parameter, you must also specify the **Gain compression at saturation (dB)**. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Gain compression at saturation (dB)

Decrease in gain (GC_{sat}) when the power is fully saturated. The block ignores this parameter if you do not specify the **Output saturation power (dBm)**. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Frequency (Hz)

Scalar or vector value of frequency points corresponding to the third-order intercept and power data. This parameter is disabled if the data source contains power data or IP3 data. If you use a scalar value, the **IP3 (dBm)**, **1 dB gain compression power (dBm)**, and **Output saturation power (dBm)** parameters must all be scalars. If you use a vector value, one or more of the **IP3 (dBm)**, **1 dB gain compression power (dBm)**, and **Output saturation power (dBm)** parameters must also be a vector.

Visualization Tab

For information about plotting, see “Create Plots”.

Examples

Plotting Parameters with the S-Parameters Amplifier Block

The following example specifies S-parameters [-.33+.71i, -.03i; 8.12-.02i, -.37-.37i] and [0.16+.20i, -.03-.04i; 7.71-8.04i, -.70-.12i] at frequencies 2.0 GHz and 2.1 GHz respectively, with a reference impedance of 50 ohms. The example uses the MATLAB `cat` function to create the 2-by-2-by-2 S-parameters array.

```
cat(3,[-.33+0.71i,     -.03i;  8.12-.02i, -.37-.37i],...
      [ .16+0.20i, -.03-.04i; 7.71-8.04i, -.70-.12i])
```

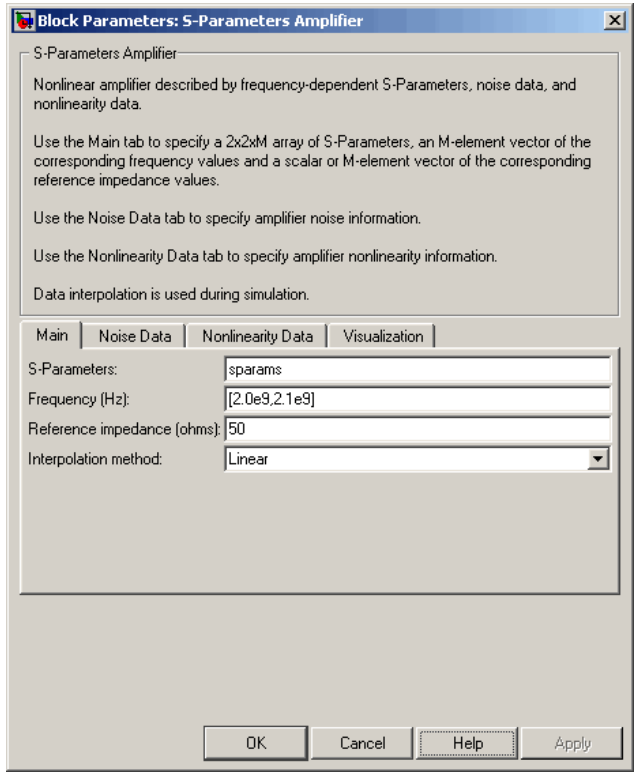
1. Type the following command at the MATLAB prompt to create a variable called `sparams` that stores the values of the S-parameters.

   ```
   sparams = cat(3,...
         [-.33+0.71i, -.03i;  8.12-.02i, -.37-.37i],...
         [ .16+0.20i, -.03-.04i; 7.71-8.04i, -.70-.12i])
   ```

2. Set the S-Parameters Amplifier block parameters on the **Main** tab as follows:

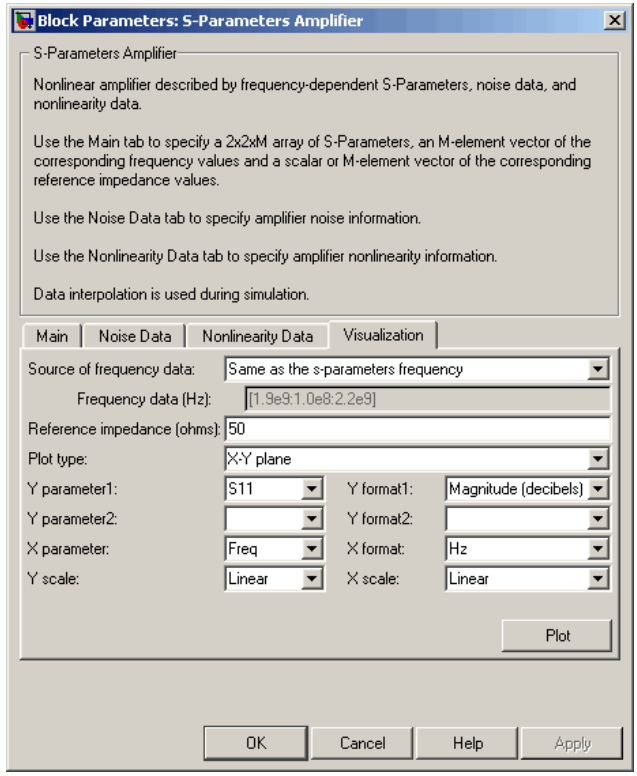
   - Set the **S-Parameters** parameter to `sparams`.
   - Set the **Frequency (Hz)** parameter to `[2.0e9,2.1e9]`.

   Click **Apply**. This action applies the specified settings.

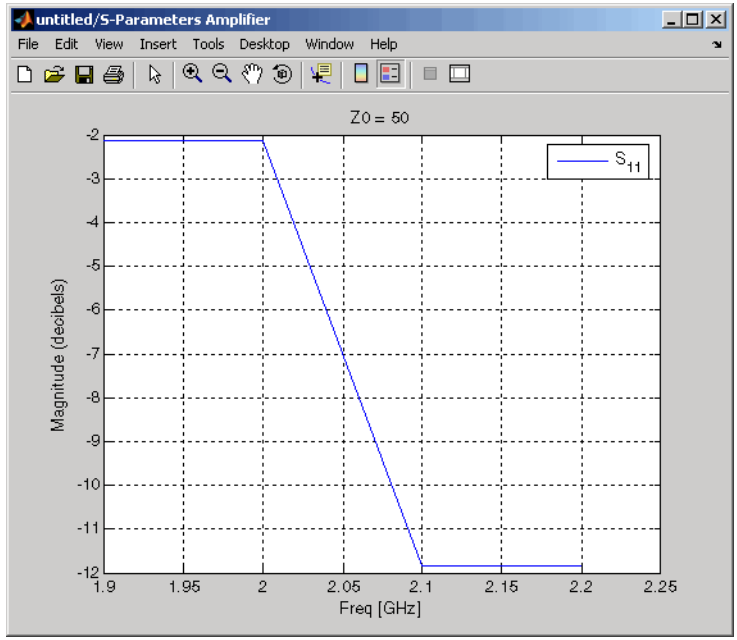


3 Set the S-Parameters Amplifier block parameters on the **Visualization** tab as follows:

- In the **Plot type** list, select `X-Y plane`.
- In the **Y parameter1** list, select `S11`.



Click **Plot**. This action creates an X-Y Plane plot of the S_{11} parameters using the frequencies taken from the **Frequency (Hz)** parameter on the **Main** tab.



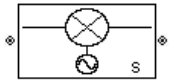
See Also

General Amplifier, Output Port, Y-Parameters Amplifier, Z-Parameters Amplifier

`interp1` (MATLAB)

S-Parameters Mixer

Model mixer and local oscillator using S-parameters



Library

Mixer sublibrary of the Physical library

Description

The S-Parameters Mixer block models the nonlinear mixer described in the block dialog box, in terms of its frequency-dependent S-parameters, the frequencies and reference impedance of the S-parameters, noise data (including phase noise data), and nonlinearity data.

Network Parameters

The S_{21} parameter values describe the conversion gain as a function of frequency, referred to the mixer input frequency. The other S-parameters also refer to the mixer input frequency.

The S-Parameters Mixer block interpolates the given S-parameters to determine their values at the modeling frequencies the Output Port block calculates. For more details about how the Output Port block calculates the modeling frequencies, see “Map Network Parameters to Modeling Frequencies”.

RF Blockset Equivalent Baseband software computes the reflected wave at the mixer input (b_1) and at the mixer output (b_2) from the interpolated S-parameters as

$$\begin{bmatrix} b_1(f_{in}) \\ b_2(f_{out}) \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} a_1(f_{in}) \\ a_2(f_{out}) \end{bmatrix}$$

where

- f_{in} and f_{out} are the mixer input and output frequencies, respectively.
- a_1 and a_2 are the incident waves at the mixer input and output, respectively.

The interpolated S_{21} parameter values describe the conversion gain as a function of frequency, referred to the mixer input frequency.

Active Noise

You can specify active block noise in one of the following ways:

- Spot noise data in the S-Parameters Mixer block dialog box.
- Noise figure, noise factor, or noise temperature value in the S-Parameters Mixer block dialog box.

If you specify block noise as spot noise data, the block uses the data to calculate noise figure. The block first interpolates the noise data for the modeling frequencies, using the specified **Interpolation method**. It then calculates the noise figure using the resulting values.

Phase Noise

The S-Parameters Mixer block applies phase noise to a complex baseband signal. The block first generates additive white Gaussian noise (AWGN) and filters the noise with a digital FIR filter. It then adds the resulting noise to the angle component of the input signal.

The blockset computes the digital filter by:

1. Interpolating the specified phase noise level to determine the phase noise values at the modeling frequencies.
2. Taking the IFFT of the resulting phase noise spectrum to get the coefficients of the FIR filter.

Note If you specify phase noise as a scalar value, the blockset assumes that the phase noise is the phase noise is constant at all modeling frequencies and does not have a *1/f* slope. This assumption differs from that made by the Mathematical Mixer block.

Nonlinearity

You can introduce nonlinearities into your model by specifying parameters in the **Nonlinearity Data** tab of the S-parameters Mixer block dialog box. Depending on which of these parameters you specify, the block computes up to four of the coefficients c_1, c_3, c_5, and c_7 of the polynomial

$$F_{AM/AM}(s) = c_1 s + c_3|s|^2 s + c_5|s|^4 s + c_7|s|^6 s$$

that determines the AM/AM conversion for the input signal s. The block automatically calculates c_1, the linear gain term. If you do not specify additional nonlinearity data, the block operates as a mixer with a linear gain. If you do, the block calculates one or more of the remaining coefficients as the solution to a system of linear equations, determined by the following method.

1 The block checks whether you have specified a value other than `Inf` for:

- The third-order intercept point ($OIP3$ or $IIP3$).
- The output power at the 1-dB compression point ($P_{1dB,\,out}$).
- The output power at saturation ($P_{sat,\,out}$).

In addition, if you have specified $P_{sat,\,out}$, the block uses the value for the gain compression at saturation (GC_{sat}). Otherwise, GC_{sat} is not used. You define each of these parameters in the block dialog box, on the **Nonlinearity Data** tab.

2 The block calculates a corresponding input or output value for the parameters you have specified. In units of dB and dBm,

$$\begin{aligned} P_{sat,\,out} + GC_{sat} &= P_{sat,\,in} + G_{lin} \\ P_{1dB,\,out} + 1 &= P_{1dB,\,in} + G_{lin} \\ OIP3 &= IIP3 + G_{lin} \end{aligned}$$

where G_{lin} is c_1 in units of dB.

3 The block formulates the coefficients c_3, c_5, and c_7, where applicable, as the solutions to a system of one, two, or three linear equations. The number of equations used is equal to the number of parameters you provide. For example, if you specify all three parameters, the block formulates the coefficients according to the following equations:

$$\sqrt{P_{sat,out}} = c_1\sqrt{P_{sat,in}} + c_3\left(\sqrt{P_{sat,in}}\right)^3 + c_5\left(\sqrt{P_{sat,in}}\right)^5 + c_7\left(\sqrt{P_{sat,in}}\right)^7$$

$$\sqrt{P_{1dB,out}} = c_1\sqrt{P_{1dB,in}} + c_3\left(\sqrt{P_{1dB,in}}\right)^3 + c_5\left(\sqrt{P_{1dB,in}}\right)^5 + c_7\left(\sqrt{P_{1dB,in}}\right)^7$$

$$0 = \frac{c_1}{IIP3} + c_3$$

The first two equations are the evaluation of the polynomial $F_{AM/AM}(s)$ at the points $(\sqrt{P_{sat,in}}, \sqrt{P_{sat,out}})$ and $(\sqrt{P_{1dB,in}}, \sqrt{P_{1dB,out}})$, expressed in linear units (such as W or mW) and normalized to a 1-Ω impedance. The third equation is the definition of the third-order intercept point.

The calculation omits higher-order terms according to the available degrees of freedom of the system. If you specify only two of the three parameters, the block does not use the equation involving the parameter you did not specify, and eliminates any c_7 terms from the remaining equations. Similarly, if you provide only one of the parameters, the block uses only the solution to the equation involving that parameter and omits any c_5 or c_7 terms.

If you provide vectors of nonlinearity and frequency data, the block calculates the polynomial coefficients using values for the parameters interpolated at the center frequency.

Parameters

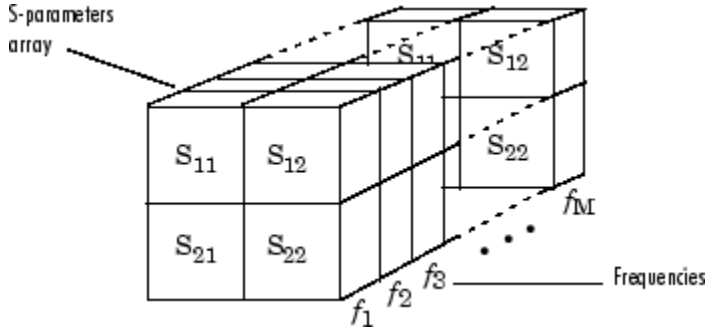
Main Tab

S-Parameters

S-parameters for a nonlinear mixer in a 2-by-2-by-M array. M is the number of S-parameters.

Frequency (Hz)

Frequencies of the S-parameters as an M-element vector. The order of the frequencies must correspond to the order of the S-parameters in **S-Parameters**. All frequencies must be positive. The following figure shows the correspondence between the S-parameters array and the vector of frequencies.



Reference impedance (ohms)

Reference impedance of the S-parameters as a scalar or a vector of length M. The value of this parameter can be real or complex. If you provide a scalar value, then that value is applied to all frequencies.

Interpolation method

The method used to interpolate the network parameters. The following table lists the available methods describes each one.

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

Mixer Type

Type of mixer. Choices are `Downconverter` (default) and `Upconverter`.

LO frequency (Hz)

Local oscillator frequency. If you choose `Downconverter`, the blockset computes the mixer output frequency, f_{out}, from the mixer input frequency, f_{in}, and the local oscillator frequency, f_{lo}, as $f_{out} = f_{in} - f_{lo}$. If you choose `Upconverter`, $f_{out} = f_{in} + f_{lo}$.

Note For a downconverting mixer, the local oscillator frequency must satisfy the condition $f_{in} - f_{lo} \geq 1/(2t_s)$, where t_s is the sample time specified in the Input Port block. Otherwise, an error appears.

Noise Data Tab

Phase noise frequency offset (Hz)

Vector specifying the frequency offset.

Phase noise level (dBc/Hz)

Vector specifying the phase noise level.

Noise type

Type of noise data. The value can be `Noise figure`, `Spot noise data`, `Noise factor`, or `Noise temperature`. This parameter is disabled if the data source contains noise data.

Noise figure (dB)

Scalar ratio or vector of ratios, in decibels, of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Noise figure`.

Minimum noise figure (dB)

Minimum scalar ratio or vector of minimum ratios of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Spot noise data`.

Optimal reflection coefficient

Optimal mixer source impedance. This parameter is enabled if **Noise type** is set to `Spot noise data`. The value can be a scalar or vector.

Equivalent normalized resistance

Resistance or vector of resistances normalized to the resistance value or values used to take the noise measurement. This parameter is enabled if **Noise type** is set to `Spot noise data`.

Noise factor

Scalar ratio or vector of ratios of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Noise factor`.

Noise temperature (K)

Equivalent temperature or vector of temperatures that produce the same amount of noise power as the mixer. This parameter is enabled if **Noise type** is set to `Noise temperature`.

Frequency (Hz)

Scalar value or vector corresponding to the domain of frequencies over which you are specifying the noise data. If you provide a scalar value for your noise data, the block ignores the **Frequency (Hz)** parameter and uses the noise data for all frequencies. If you provide a vector of values for your noise data, it must be the same size as the vector of frequencies. The block uses the **Interpolation method** specified in the **Main** tab to interpolate noise data.

Nonlinearity Data Tab

IP3 type

Type of third-order intercept point. The value can be `IIP3` (input intercept point) or `OIP3` (output intercept point). This parameter is disabled if the data source contains power data or IP3 data.

IP3 (dBm)

Value of third-order intercept point. This parameter is disabled if the data source contains power data or IP3 data. Use the default value, `Inf`, if you do not know the IP3 value. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

1 dB gain compression power (dBm)

Output power value ($P_{1dB,\,out}$) at which gain has decreased by 1 dB. This parameter is disabled if the data source contains power data or 1-dB compression point data. Use the default value, `Inf`, if you do not know the 1-dB compression point. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Output saturation power (dBm)

Output power value ($P_{sat,\,out}$) that the mixer produces when fully saturated. This parameter is disabled if the data source contains output saturation power data. Use the default value, `Inf`, if you do not know the saturation power. If you specify this parameter, you must also specify the **Gain compression at saturation (dB)**. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Gain compression at saturation (dB)

Decrease in gain (GC_{sat}) when the power is fully saturated. The block ignores this parameter if you do not specify the **Output saturation power (dBm)**. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Frequency (Hz)

Scalar or vector value of frequency points corresponding to the third-order intercept and power data. This parameter is disabled if the data source contains power data or IP3 data. If you use a scalar value, the **IP3 (dBm)**, **1 dB gain compression power (dBm)**, and **Output saturation power (dBm)** parameters must all be scalars. If you use a vector value, one or more of the **IP3 (dBm)**, **1 dB gain compression power (dBm)**, and **Output saturation power (dBm)** parameters must also be a vector.

Visualization Tab

For information about plotting, see “Create Plots”.

See Also

General Mixer, Output Port, Y-Parameters Mixer, Z-Parameters Mixer

S-Parameters Passive Network

Model passive network using S-parameters

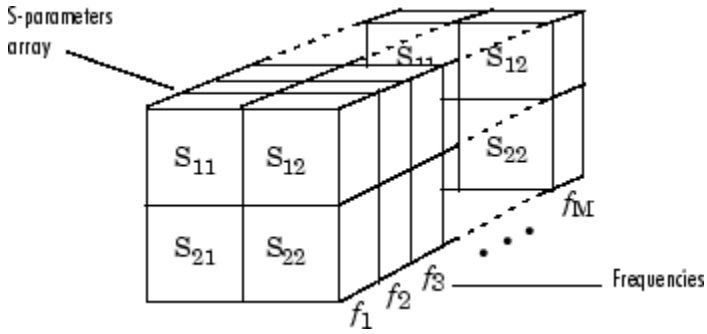
Library

Black Box Elements sublibrary of the Physical library

Description

The S-Parameters Passive Network block models the two-port passive network described in the block dialog box, in terms of its S-parameters and the frequencies and reference impedance of the S-parameters.

In the **S-Parameters** field of the block dialog box, provide the S-parameters for each of M frequencies as a 2-by-2-by-M array. In the **Frequency** field, specify the frequencies for the S-parameters as an M-element vector. The elements of the vector must be in the same order as the S-parameters. All frequencies must be positive. For example, the following figure shows the correspondence between the S-parameters array and the vector of frequencies.



The S-Parameters Passive Network block interpolates the given S-parameters to determine their values at the modeling frequencies. The modeling frequencies are

determined by the Output Port block. See “Map Network Parameters to Modeling Frequencies” for more details.

Parameters

Main Tab

S-Parameters

S-parameters for a two-port passive network in a 2-by-2-by-M array. M is the number of S-parameters.

Frequency (Hz)

Frequencies of the S-parameters as an M-element vector. The order of the frequencies must correspond to the order of the S-parameters in **S-Parameters**. All frequencies must be positive.

Reference impedance (ohms)

Reference impedance of the network as a scalar or a vector of length M. The value of this parameter can be real or complex. If you provide a scalar value, then that value is applied to all frequencies.

Interpolation method

The method used to interpolate the network parameters. The following table lists the available methods describes each one.

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

Visualization Tab

For information about plotting, see “Create Plots”.

Examples

Plotting Parameters with the S-Parameters Passive Network Block

The following example specifies S-parameters [-.96-.23i, .03-.12i; .03-.12i, -.96-.23i] and [-.96-.11i, .02-.21i; .02-.21i, -.96-.11i] at frequencies 2.0 GHz and 2.1 GHz respectively. The example then plots these parameters.

The example first uses the MATLAB `cat` function to create the 2-by-2-by-2 S-parameters array.

```
cat(3,[-.96-.23i, .03-.12i; .03-.12i, -.96-.23i],...
      [-.96-.11i, .02-.21i; .02-.21i, -.96-.11i])
```

You could also use the MATLAB `reshape` function. The following command produces the same result as previous command.

```
reshape([-.96-.23i;.03-.12i;.03-.12i;-.96-.23i;...
         -.96-.11i;.02-.21i;.02-.21i;-.96-.11i],2,2,2)
```

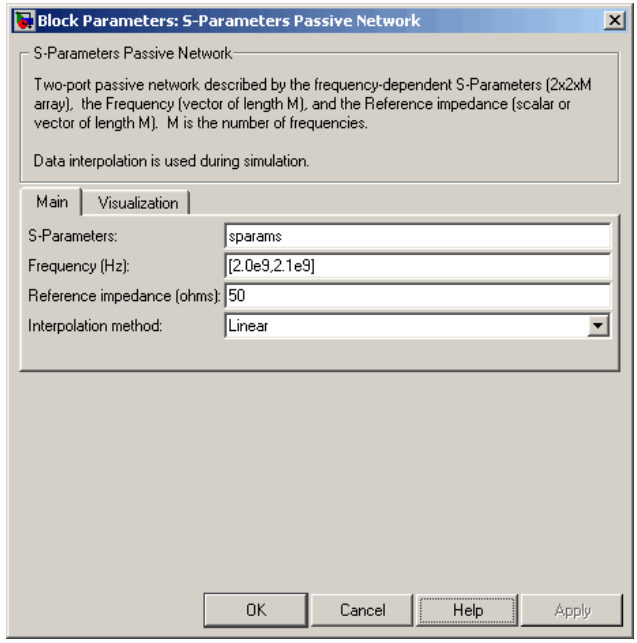
1. Type the following command at the MATLAB prompt to create a variable called `sparams` that stores the values of the S-parameters.

   ```
   sparams = cat(3,...
         [-.96-.23i, .03-.12i; .03-.12i, -.96-.23i],...
         [-.96-.11i, .02-.21i; .02-.21i, -.96-.11i])
   ```

2. Set the S-Parameters Passive Network block parameters on the **Main** tab as follows:

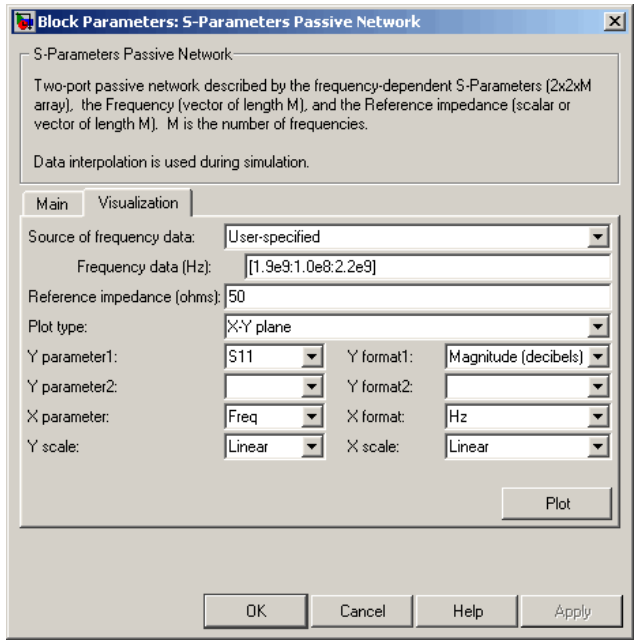
   - Set the **S-Parameters** parameter to `sparams`.
   - Set the **Frequency (Hz)** parameter to `[2.0e9,2.1e9]`.

   Click **Apply**. This action applies the specified settings.

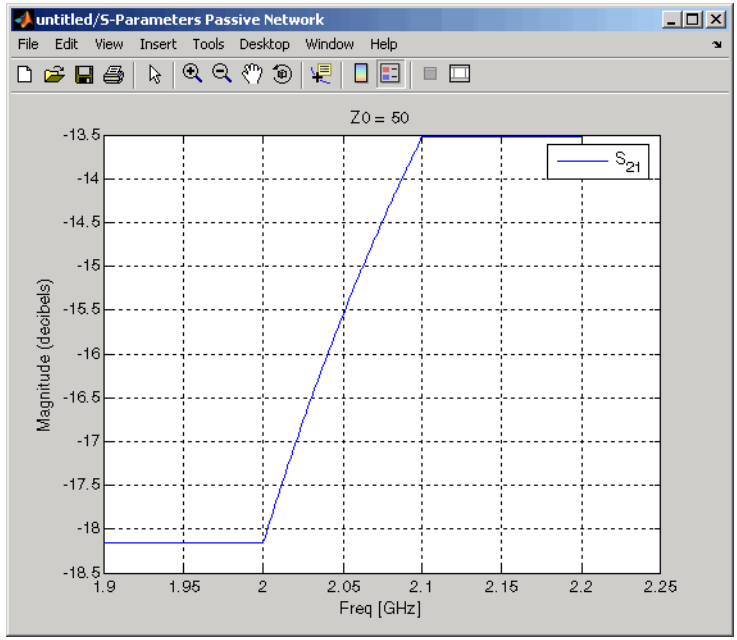


3 Set the S-Parameters Passive Network block parameters on the **Visualization** tab as follows:

- In the **Source of frequency data** list, select `User-specified`.
- Set the **Frequency data (Hz)** parameter to `[1.9e9:1.0e8:2.2e9]`.
- In the **Y parameter1** list, select `S21`.



Click **Plot**. This action creates an X-Y Plane plot of the magnitude of the S_{21} parameters, in decibels, in the frequency range 1.9 to 2.2 GHz.



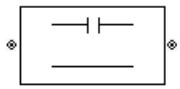
See Also

General Circuit Element, General Passive Network, Output Port, Y-Parameters Passive Network, Z-Parameters Passive Network

`interp1` (MATLAB)

Series C

Model series capacitor



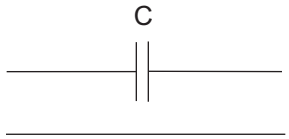
Library

Ladders Filters sublibrary of the Physical library

Description

The Series C block models the series capacitor described in the block dialog box, in terms of its frequency-dependent S-parameters.

The series C object is a two-port network, as shown in the following circuit diagram.



Parameters

Main Tab

Capacitance (F)

Scalar value for the capacitance. The value must be nonnegative.

Visualization Tab

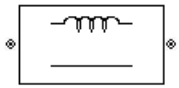
For information about plotting, see “Create Plots”.

See Also

General Passive Network, LC Bandpass Pi, LC Bandpass Tee, LC Bandstop Pi, LC Bandstop Tee, LC Highpass Pi, LC Highpass Tee, LC Lowpass Pi, LC Lowpass Tee, Series L, Series R, Series RLC, Shunt C, Shunt L, Shunt R, Shunt RLC

Series L

Model series inductor



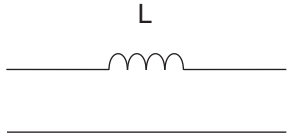
Library

Ladders Filters sublibrary of the Physical library

Description

The Series L block models the series inductor described in the block dialog box, in terms of its frequency-dependent S-parameters.

The series L object is a two-port network, as shown in the following circuit diagram.



Parameters

Main Tab

Inductance (H)

Scalar value for the inductance. The value must be nonnegative.

Visualization Tab

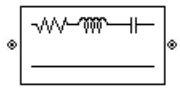
For information about plotting, see “Create Plots”.

See Also

General Passive Network, LC Bandpass Pi, LC Bandpass Tee, LC Bandstop Pi, LC Bandstop Tee, LC Highpass Pi, LC Highpass Tee, LC Lowpass Pi, LC Lowpass Tee, Series C, Series R, Series RLC, Shunt C, Shunt L, Shunt R, Shunt RLC

Series RLC

Model series RLC network



Library

Ladders Filters sublibrary of the Physical library

Description

The Series RLC block models the series RLC network described in the block dialog box, in terms of its frequency-dependent S-parameters.

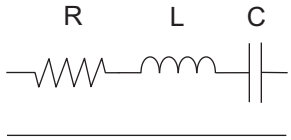
For the given resistance, inductance, and capacitance, the block first calculates the ABCD-parameters at each frequency contained in the vector of modeling frequencies, and then converts the ABCD-parameters to S-parameters using the RF Toolbox `abcd2s` function. See the Output Port block reference page for information about determining the modeling frequencies.

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 series RLC object is a two-port network as shown in the following circuit diagram.



Parameters

Main Tab

Resistance (ohms)

Scalar value for the resistance. The value must be nonnegative.

Inductance (H)

Scalar value for the inductance. The value must be nonnegative.

Capacitance (F)

Scalar value for the capacitance. The value must be nonnegative.

Visualization Tab

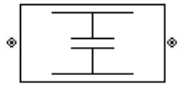
For information about plotting, see “Create Plots”.

See Also

General Passive Network, LC Bandpass Pi, LC Bandpass Tee, LC Bandstop Pi, LC Bandstop Tee, LC Highpass Pi, LC Highpass Tee, LC Lowpass Pi, LC Lowpass Tee, Series C, Series L, Series R, Shunt C, Shunt L, Shunt R, Shunt RLC

Shunt C

Model shunt capacitor



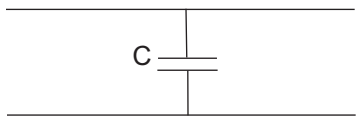
Library

Ladders Filters sublibrary of the Physical library

Description

The Shunt C block models the shunt capacitor described in the block dialog box, in terms of its frequency-dependent S-parameters.

The shunt C object is a two-port network, as shown in the following circuit diagram.



Parameters

Main Tab

Capacitance (F)

Scalar value for the capacitance. The value must be nonnegative.

Visualization Tab

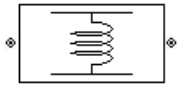
For information about plotting, see “Create Plots”.

See Also

General Passive Network, LC Bandpass Pi, LC Bandpass Tee, LC Bandstop Pi, LC Bandstop Tee, LC Highpass Pi, LC Highpass Tee, LC Lowpass Pi, LC Lowpass Tee, Series C, Series L, Series R, Series RLC, Shunt L, Shunt R, Shunt RLC

Shunt L

Model shunt inductor



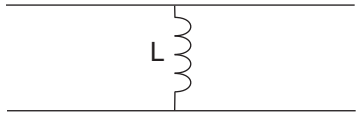
Library

Ladders Filters sublibrary of the Physical library

Description

The Shunt L block models the shunt inductor described in the block dialog box, in terms of its frequency-dependent S-parameters.

The shunt L object is a two-port network, as shown in the following circuit diagram.



Parameters

Main Tab

Inductance (H)

Scalar value for the inductance. The value must be nonnegative.

Visualization Tab

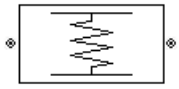
For information about plotting, see “Create Plots”.

See Also

General Passive Network, LC Bandpass Pi, LC Bandpass Tee, LC Bandstop Pi, LC Bandstop Tee, LC Highpass Pi, LC Highpass Tee, LC Lowpass Pi, LC Lowpass Tee, Series C, Series L, Series R, Series RLC, Shunt C, Shunt R, Shunt RLC

Shunt R

Model shunt resistor



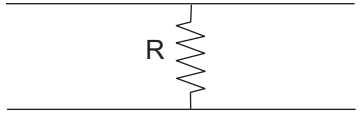
Library

Ladders Filters sublibrary of the Physical library

Description

The Shunt R block models the shunt resistor described in the block dialog box, in terms of its frequency-dependent S-parameters.

The shunt R object is a two-port network, as shown in the following circuit diagram.



Parameters

Main Tab

Resistance (ohms)

Scalar value for the resistance. The value must be nonnegative.

Visualization Tab

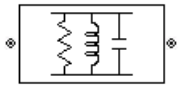
For information about plotting, see “Create Plots”.

See Also

General Passive Network, LC Bandpass Pi, LC Bandpass Tee, LC Bandstop Pi, LC Bandstop Tee, LC Highpass Pi, LC Highpass Tee, LC Lowpass Pi, LC Lowpass Tee, Series C, Series L, Series R, Series RLC, Shunt C, Shunt L, Shunt RLC

Shunt RLC

Model shunt RLC network



Library

Ladders Filters sublibrary of the Physical library

Description

The Shunt RLC block models the shunt RLC network described in the block dialog box, in terms of its frequency-dependent S-parameters.

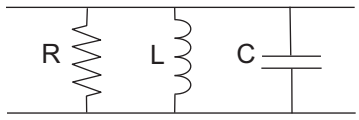
For the given resistance, inductance, and capacitance, the block first calculates the ABCD-parameters at each frequency contained in the vector of modeling frequencies, and then converts the ABCD-parameters to S-parameters using the RF Toolbox `abcd2s` function. See the Output Port block reference page for information about determining the modeling frequencies.

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 shunt RLC object is a two-port network as shown in the following circuit diagram.



Parameters

Main Tab

Resistance (ohms)

Scalar value for the resistance. The value must be nonnegative.

Inductance (H)

Scalar value for the inductance. The value must be nonnegative.

Capacitance (F)

Scalar value for the capacitance. The value must be nonnegative.

Visualization Tab

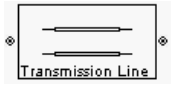
For information about plotting, see “Create Plots”.

See Also

General Passive Network, LC Bandpass Pi, LC Bandpass Tee, LC Bandstop Pi, LC Bandstop Tee, LC Highpass Pi, LC Highpass Tee, LC Lowpass Pi, LC Lowpass Tee, Series C, Series L, Series R, Series RLC, Shunt C, Shunt L, Shunt R

Transmission Line (Equivalent Baseband)

Model general transmission line



Library

Transmission Lines sublibrary of the Physical library

Description

The Transmission Line block models the transmission line described in the block dialog box in terms of its physical parameters. The transmission line, which can be lossy or lossless, is treated as a two-port linear network.

The block enables you to model the transmission line as a stub or as a stubless line.

Stubless Transmission Line

If you model the transmission line as a stubless line, the Transmission Line block 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 block 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 block calculates k from the specified parameters 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

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

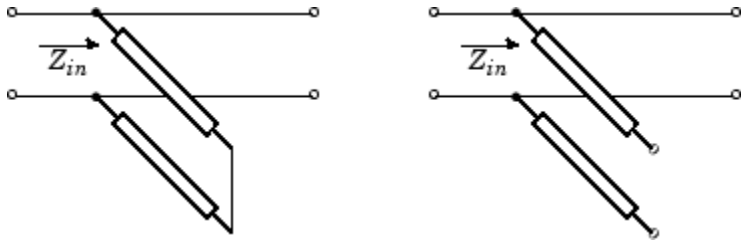
The phase velocity V_P is also known as the *wave propagation velocity*.

Shunt and Series Stubs

If you model the transmission line as a shunt or series stub, the Transmission Line block first calculates the ABCD-parameters at each frequency contained in the vector of modeling frequencies. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

Shunt ABCD-Parameters

When you set the **Stub mode** parameter in the mask dialog box to `Shunt`, the two-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown here.

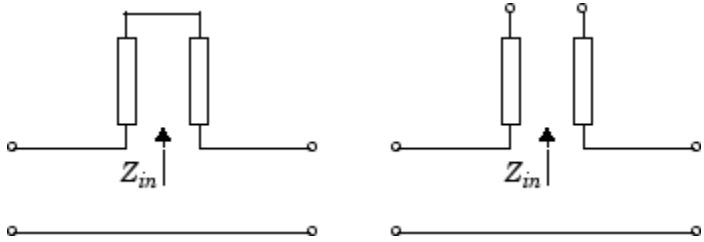


Z_{in} is the input impedance of the shunt circuit. The ABCD-parameters for the shunt stub are calculated as

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

Series ABCD-Parameters

When you set the **Stub mode** parameter in the mask dialog box to `Series`, the two-port network consists of a series transmission line that you can terminate with either a short circuit or an open circuit as shown here.



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

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

Parameters

Main Tab

Characteristic impedance (ohms)

Characteristic impedance of the transmission line. The value can be complex.

Phase velocity (m/s)

Propagation velocity of a uniform plane wave on the transmission line.

Loss (dB/m)

Reduction in strength of the signal as it travels over the transmission line. Must be positive.

Frequency (Hz)

Vector of modeling frequencies. The block performs the calculations listed in the Description section at each frequency you provide.

Transmission line length (m)

Physical length of the transmission line.

Stub mode

Type of stub. Choices are `Not a stub`, `Shunt`, or `Series`.

Termination of stub

Stub termination for stub modes `Shunt` and `Series`. Choices are `Open` or `Short`. This parameter becomes visible only when **Stub mode** is set to `Shunt` or `Series`.

Visualization Tab

For information about plotting, see “Create Plots”.

References

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

See Also

Coaxial Transmission Line, Coplanar Waveguide Transmission Line, General Passive Network, Microstrip Transmission Line, Parallel-Plate Transmission Line, Two-Wire Transmission Line

Two-Wire Transmission Line

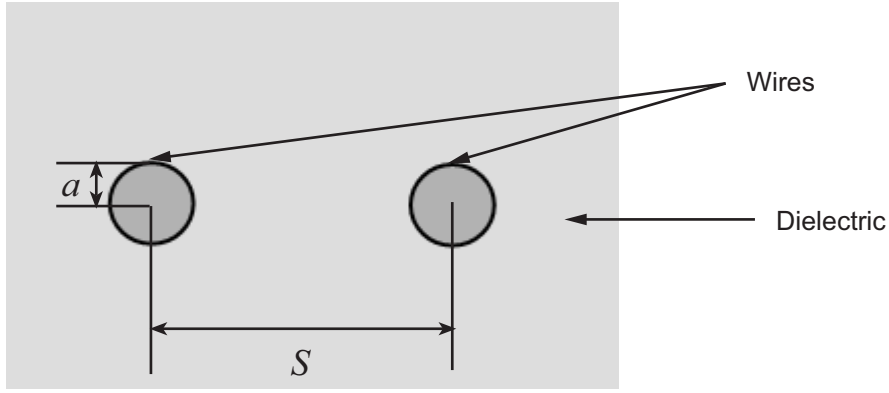
Model two-wire transmission line

Library

Transmission Lines sublibrary of the Physical library

Description

The Two-Wire Transmission Line block models the two-wire transmission line described in the block dialog box in terms of its frequency-dependent S-parameters. 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 Blockset Equivalent Baseband software assumes the relative permittivity and permeability are uniform.



The block enables you to model the transmission line as a stub or as a stubless line.

Stubless Transmission Line

If you model a two-wire transmission line as a stubless line, the Two-Wire Transmission Line block 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 block 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, a vector of modeling frequencies. Both can be expressed in terms of the resistance (R), inductance (L), conductance (G), and capacitance (C) per unit length (meters) as follows:

$$Z_0 = \sqrt{\frac{R + j\omega L}{G + j\omega C}}$$

$$k = k_r + jk_i = \sqrt{(R + j\omega L)(G + j\omega C)}$$

where

$$R = \frac{1}{\pi a \sigma_{cond} \delta_{cond}}$$

$$L = \frac{\mu}{\pi} \operatorname{acosh}\left(\frac{D}{2a}\right)$$

$$G = \frac{\pi \omega \varepsilon''}{\operatorname{acosh}\left(\frac{D}{2a}\right)}$$

$$C = \frac{\pi \varepsilon}{\operatorname{acosh}\left(\frac{D}{2a}\right)}$$

and $\omega = 2\pi f$.

In these equations:

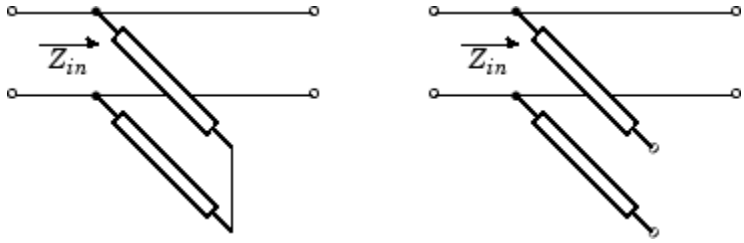
- σ_{cond} is the conductivity in the conductor.
- μ is the permeability of the dielectric.
- ε is the permittivity of the dielectric.
- ε'' is the imaginary part of ε, $\varepsilon'' = \varepsilon_0\varepsilon_r \tan\delta$, where:
  - ε_0 is the permittivity of free space.
  - ε_r is the **Relative permittivity constant** parameter value.
  - $\tan\delta$ is the **Loss tangent of dielectric** parameter 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 Output Port block.

Shunt and Series Stubs

If you model the transmission line as a shunt or series stub, the Two-Wire Transmission Line block first calculates the ABCD-parameters at each frequency contained in the vector of modeling frequencies. It then uses the `abcd2s` function to convert the ABCD-parameters to S-parameters.

Shunt ABCD-Parameters

When you set the **Stub mode** parameter in the mask dialog box to `Shunt`, the two-port network consists of a stub transmission line that you can terminate with either a short circuit or an open circuit as shown here.

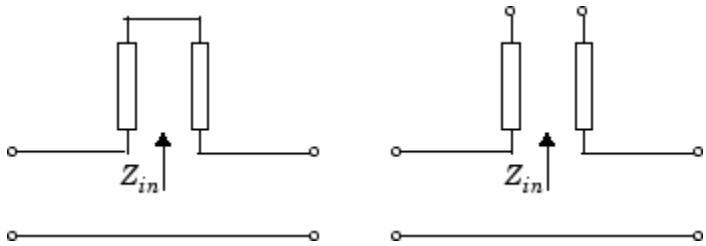


Z_{in} is the input impedance of the shunt circuit. The ABCD-parameters for the shunt stub are calculated as

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

Series ABCD-Parameters

When you set the **Stub mode** parameter in the mask dialog box to `Series`, the two-port network consists of a series transmission line that you can terminate with either a short circuit or an open circuit as shown here.



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

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

Parameters

Main Tab

Wire radius (m)

Radius of the conducting wires of the two-wire transmission line.

Wire separation (m)

Physical distance between the wires.

Relative permeability constant

Relative permeability of the dielectric expressed as the ratio of the permeability of the dielectric to permeability in free space μ_0.

Relative permittivity constant

Relative permittivity of the dielectric expressed as the ratio of the permittivity of the dielectric to permittivity in free space ε_0.

Loss tangent of dielectric

Loss angle tangent of the dielectric.

Conductivity of conductor (S/m)

Conductivity of the conductor in siemens per meter.

Transmission line length (m)

Physical length of the transmission line.

Stub mode

Type of stub. Choices are `Not a stub`, `Shunt`, or `Series`.

Termination of stub

Stub termination for stub modes `Shunt` and `Series`. Choices are `Open` or `Short`. This parameter becomes visible only when **Stub mode** is set to `Shunt` or `Series`.

Visualization Tab

For information about plotting, see “Create Plots”.

References

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

See Also

Coaxial Transmission Line, Coplanar Waveguide Transmission Line, General Passive Network, Transmission Line, Microstrip Transmission Line, Parallel-Plate Transmission Line

Y-Parameters Amplifier

Model nonlinear amplifier using Y-parameters

Library

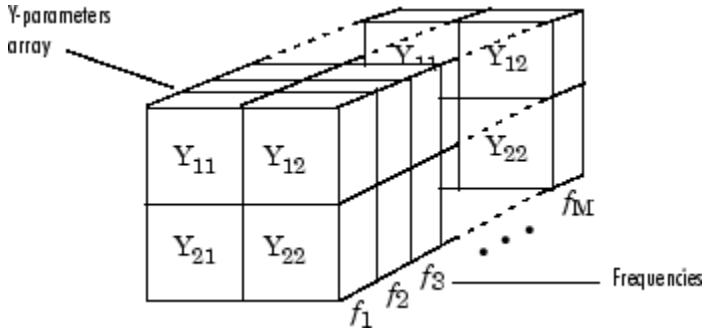
Amplifiers sublibrary of the Physical library

Description

The Y-Parameters Amplifier block models the nonlinear amplifier described in the block dialog box, in terms of its frequency-dependent, the frequencies of the Y-parameters, noise data, and nonlinearity data

Network Parameters

In the **Y-Parameters** field of the block dialog box, provide the Y-parameters for each of M frequencies as a 2-by-2-by-M array. In the **Frequency** field, specify the frequencies for the Y-parameters as an M-element vector. The elements of the frequencies vector must be in the same order as the Y-parameters. All frequencies must be positive. For example, the following figure shows the correspondence between the Y-parameters array and the vector of frequencies.



The Y-Parameters Amplifier block uses the RF Toolbox `y2s` function to convert the Y-parameters to S-parameters, and then interpolates the resulting S-parameters to determine their values at the modeling frequencies. See “Map Network Parameters to Modeling Frequencies” for more details.

Nonlinearity

You can introduce nonlinearities into your model by specifying parameters in the **Nonlinearity Data** tab of the Y-Parameters Amplifier block dialog box. Depending on which of these parameters you specify, the block computes up to four of the coefficients c_1, c_3, c_5, and c_7 of the polynomial

$$F_{AM/AM}(s) = c_1 s + c_3|s|^2 s + c_5|s|^4 s + c_7|s|^6 s$$

that determines the AM/AM conversion for the input signal s. The block automatically calculates c_1, the linear gain term. If you do not specify additional nonlinearity data, the block operates as a linear amplifier. If you do, the block calculates one or more of the remaining coefficients as the solution to a system of linear equations, determined by the following method.

1. The block checks whether you have specified a value other than `Inf` for:

   - The third-order intercept point ($OIP3$ or $IIP3$).
   - The output power at the 1-dB compression point ($P_{1dB,\,out}$).
   - The output power at saturation ($P_{sat,\,out}$).

   In addition, if you have specified $P_{sat,\,out}$, the block uses the value for the gain compression at saturation (GC_{sat}). Otherwise, GC_{sat} is not used. You define each of these parameters in the block dialog box, on the **Nonlinearity Data** tab.

2. The block calculates a corresponding input or output value for the parameters you have specified. In units of dB and dBm,

   $$\begin{aligned} P_{sat,\,out} + GC_{sat} &= P_{sat,\,in} + G_{lin} \\ P_{1dB,\,out} + 1 &= P_{1dB,\,in} + G_{lin} \\ OIP3 &= IIP3 + G_{lin} \end{aligned}$$

   where G_{lin} is c_1 in units of dB.

3 The block formulates the coefficients c_3, c_5, and c_7, where applicable, as the solutions to a system of one, two, or three linear equations. The number of equations used is equal to the number of parameters you provide. For example, if you specify all three parameters, the block formulates the coefficients according to the following equations:

$$\sqrt{P_{sat,out}} = c_1\sqrt{P_{sat,in}} + c_3\left(\sqrt{P_{sat,in}}\right)^3 + c_5\left(\sqrt{P_{sat,in}}\right)^5 + c_7\left(\sqrt{P_{sat,in}}\right)^7$$

$$\sqrt{P_{1dB,out}} = c_1\sqrt{P_{1dB,in}} + c_3\left(\sqrt{P_{1dB,in}}\right)^3 + c_5\left(\sqrt{P_{1dB,in}}\right)^5 + c_7\left(\sqrt{P_{1dB,in}}\right)^7$$

$$0 = \frac{c_1}{IIP3} + c_3$$

The first two equations are the evaluation of the polynomial $F_{AM/AM}(s)$ at the points $(\sqrt{P_{sat,in}}, \sqrt{P_{sat,out}})$ and $(\sqrt{P_{1dB,in}}, \sqrt{P_{1dB,out}})$, expressed in linear units (such as W or mW) and normalized to a 1-Ω impedance. The third equation is the definition of the third-order intercept point.

The calculation omits higher-order terms according to the available degrees of freedom of the system. If you specify only two of the three parameters, the block does not use the equation involving the parameter you did not specify, and eliminates any c_7 terms from the remaining equations. Similarly, if you provide only one of the parameters, the block uses only the solution to the equation involving that parameter and omits any c_5 or c_7 terms.

If you provide vectors of nonlinearity and frequency data, the block calculates the polynomial coefficients using values for the parameters interpolated at the center frequency.

Active Noise

You can specify active block noise in one of the following ways:

- Spot noise data in the Y-Parameters Amplifier block dialog box.
- Noise figure, noise factor, or noise temperature value in the Y-Parameters Amplifier block dialog box.

If you specify block noise as spot noise data, the block uses the data to calculate noise figure. The block first interpolates the noise data for the modeling frequencies, using the specified **Interpolation method**. It then calculates the noise figure using the resulting values.

Parameters

Main Tab

Y-Parameters

Y-parameters for a nonlinear amplifier in a 2-by-2-by-M array. M is the number of Y-parameters.

Frequency (Hz)

Frequencies of the Y-parameters as an M-element vector. The order of the frequencies must correspond to the order of the Y-parameters in **Y-Parameters**. All frequencies must be positive.

Interpolation method

The method used to interpolate the network parameters. The following table lists the available methods describes each one.

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

Noise Data Tab

Noise type

Type of noise data. The value can be `Noise figure`, `Spot noise data`, `Noise factor`, or `Noise temperature`. This parameter is disabled if the data source contains noise data.

Noise figure (dB)

Scalar ratio or vector of ratios, in decibels, of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Noise figure`.

Minimum noise figure (dB)

Minimum scalar ratio or vector of minimum ratios of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Spot noise data`.

Optimal reflection coefficient

Optimal amplifier source impedance. This parameter is enabled if **Noise type** is set to `Spot noise data`. The value can be a scalar or vector.

Equivalent normalized resistance

Resistance or vector of resistances normalized to the resistance value or values used to take the noise measurement. This parameter is enabled if **Noise type** is set to `Spot noise data`.

Noise factor

Scalar ratio or vector of ratios of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Noise factor`.

Noise temperature (K)

Equivalent temperature or vector of temperatures that produce the same amount of noise power as the amplifier. This parameter is enabled if **Noise type** is set to `Noise temperature`.

Frequency (Hz)

Scalar value or vector corresponding to the domain of frequencies over which you are specifying the noise data. If you provide a scalar value for your noise data, the block ignores the **Frequency (Hz)** parameter and uses the noise data for all frequencies. If you provide a vector of values for your noise data, it must be the same size as the vector of frequencies. The block uses the **Interpolation method** specified in the **Main** tab to interpolate noise data.

Nonlinearity Data Tab

IP3 type

Type of third-order intercept point. The value can be `IIP3` (input intercept point) or `OIP3` (output intercept point). This parameter is disabled if the data source contains power data or IP3 data.

IP3 (dBm)

Value of third-order intercept point. This parameter is disabled if the data source contains power data or IP3 data. Use the default value, `Inf`, if you do not know the IP3 value. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

1 dB gain compression power (dBm)

Output power value ($P_{1dB,\,out}$) at which gain has decreased by 1 dB. This parameter is disabled if the data source contains power data or 1-dB compression point data. Use the default value, `Inf`, if you do not know the 1-dB compression point. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Output saturation power (dBm)

Output power value ($P_{sat,\,out}$) that the amplifier produces when fully saturated. This parameter is disabled if the data source contains output saturation power data. Use the default value, `Inf`, if you do not know the saturation power. If you specify this parameter, you must also specify the **Gain compression at saturation (dB)**. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Gain compression at saturation (dB)

Decrease in gain (GC_{sat}) when the power is fully saturated. The block ignores this parameter if you do not specify the **Output saturation power (dBm)**. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Frequency (Hz)

Scalar or vector value of frequency points corresponding to the third-order intercept and power data. This parameter is disabled if the data source contains power data or IP3 data. If you use a scalar value, the **IP3 (dBm)**, **1 dB gain compression power (dBm)**, and **Output saturation power (dBm)** parameters must all be scalars. If you use a vector value, one or more of the **IP3 (dBm)**, **1 dB gain compression power (dBm)**, and **Output saturation power (dBm)** parameters must also be a vector.

Visualization Tab

For information about plotting, see “Create Plots”.

Examples

Plotting Parameters with the Y-Parameters Amplifier Block

The following example specifies Y-parameters [-.06+.58i, -.08i; 1.14-1.82i, -.07+.28i] and [.02-.21i, 0.03i; -.21+.72i, .03-.11i] at frequencies 2.0 GHz and 2.1 GHz respectively. It uses the MATLAB `cat` function to create the 2-by-2-by-2 Y-parameters array

```
cat(3,[-.06+.58i, -.08i; 1.14-1.82i, -.07+.28i],...
      [ .02-.21i, 0.03i;  -.21+.72i,  .03-.11i])
```

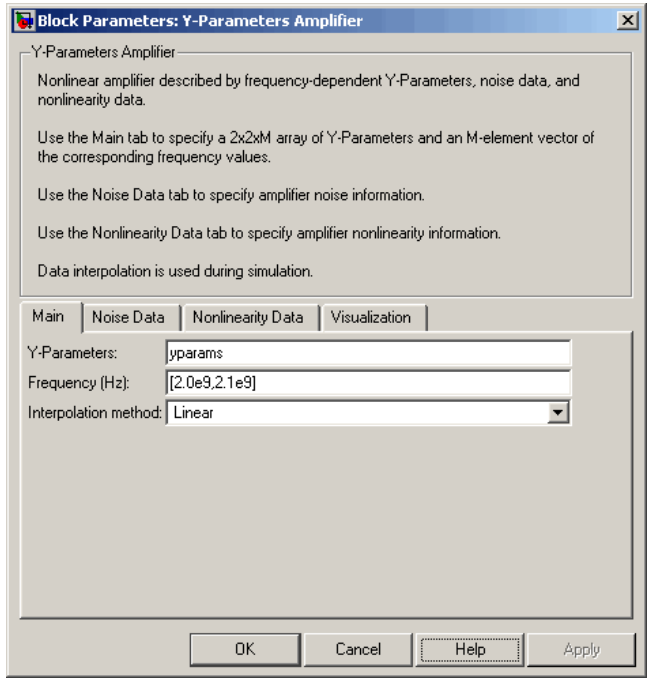
1. Type the following command at the MATLAB prompt to create a variable called `yparams` that stores the values of the Y-parameters.

   ```
   yparams = cat(3,...
         [-.06+.58i, -.08i; 1.14-1.82i, -.07+.28i],...
         [ .02-.21i, 0.03i;  -.21+.72i,  .03-.11i])
   ```

2. Set the Y-Parameters Amplifier block parameters on the **Main** tab as follows:

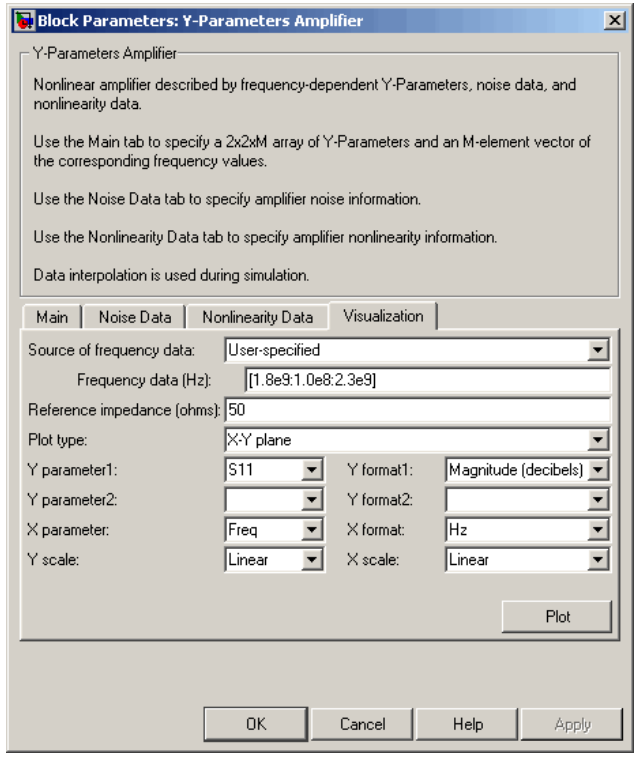
   - Set the **Y-Parameters** parameter to `yparams`.
   - Set the **Frequency (Hz)** parameter to `[2.0e9,2.1e9]`.

   Click **Apply**. This action applies the specified settings.

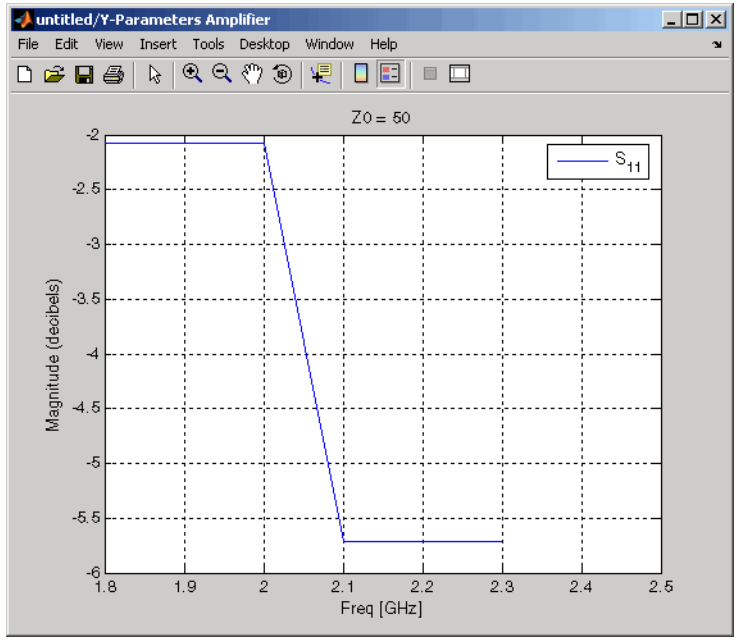


3 Set the Y-Parameters Amplifier block parameters on the **Visualization** tab as follows:

- In the **Source of frequency data** list, select `User-specified`.
- Set the **Frequency data (Hz)** parameter to `[1.8e9:1.0e8:2.3e9]`.
- In the **Plot type** list, select `X-Y plane`.
- In the **Y parameter1** list, select `S11`.



Click **Plot**. This action creates an X-Y Plane plot of the S_{11} parameters in the frequency range 1.8 to 2.3 GHz.



See Also

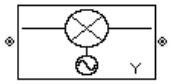
General Amplifier, Output Port, S-Parameters Amplifier, Z-Parameters Amplifier

`y2s` (RF Toolbox)

`interp1` (MATLAB)

Y-Parameters Mixer

Model mixer and local oscillator using Y-parameters



Library

Mixer sublibrary of the Physical library

Description

The Y-Parameters Mixer block models the nonlinear mixer described in the block dialog box in terms of its frequency-dependent Y-parameters, the frequencies of the Y-parameters, noise data (including phase noise data), and nonlinearity data.

Network Parameters

The Y-parameter values all refer to the mixer input frequency.

The Y-Parameters Mixer block uses the RF Toolbox `y2s` function to convert the Y-parameters to S-parameters and then interpolates the resulting S-parameters to determine their values at the modeling frequencies. See “Map Network Parameters to Modeling Frequencies” for more details.

RF Blockset Equivalent Baseband software computes the reflected wave at the mixer input (b_1) and at the mixer output (b_2) from the interpolated S-parameters as

$$\begin{bmatrix} b_1(f_{in}) \\ b_2(f_{out}) \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} a_1(f_{in}) \\ a_2(f_{out}) \end{bmatrix}$$

where

- f_{in} and f_{out} are the mixer input and output frequencies, respectively.

- a_1 and a_2 are the incident waves at the mixer input and output, respectively.

The interpolated S_{21} parameter values describe the conversion gain as a function of frequency, referred to the mixer input frequency.

Active Noise

You can specify active block noise in one of the following ways:

- Spot noise data in the Y-Parameters Mixer block dialog box.
- Noise figure, noise factor, or noise temperature value in the Y-Parameters Mixer block dialog box.

If you specify block noise as spot noise data, the block uses the data to calculate noise figure. The block first interpolates the noise data for the modeling frequencies, using the specified **Interpolation method**. It then calculates the noise figure using the resulting values.

Phase Noise

The Y-Parameters Mixer block applies phase noise to a complex baseband signal. The block first generates additive white Gaussian noise (AWGN) and filters the noise with a digital FIR filter. It then adds the resulting noise to the angle component of the input signal.

The blockset computes the digital filter by:

1. Interpolating the specified phase noise level to determine the phase noise values at the modeling frequencies.
2. Taking the IFFT of the resulting phase noise spectrum to get the coefficients of the FIR filter.

Note If you specify phase noise as a scalar value, the blockset assumes that the phase noise is the phase noise is constant at all modeling frequencies and does not have a *1/f* slope. This assumption differs from that made by the Mathematical Mixer block.

Nonlinearity

You can introduce nonlinearities into your model by specifying parameters in the **Nonlinearity Data** tab of the Y-Parameters Mixer block dialog box. Depending on which of these parameters you specify, the block computes up to four of the coefficients c_1, c_3, c_5, and c_7 of the polynomial

$$F_{AM/AM}(s) = c_1 s + c_3|s|^2 s + c_5|s|^4 s + c_7|s|^6 s$$

that determines the AM/AM conversion for the input signal s. The block automatically calculates c_1, the linear gain term. If you do not specify additional nonlinearity data, the block operates as a mixer with a linear gain. If you do, the block calculates one or more of the remaining coefficients as the solution to a system of linear equations, determined by the following method.

1 The block checks whether you have specified a value other than `Inf` for:

- The third-order intercept point ($OIP3$ or $IIP3$).
- The output power at the 1-dB compression point ($P_{1dB,\,out}$).
- The output power at saturation ($P_{sat,\,out}$).

In addition, if you have specified $P_{sat,\,out}$, the block uses the value for the gain compression at saturation (GC_{sat}). Otherwise, GC_{sat} is not used. You define each of these parameters in the block dialog box, on the **Nonlinearity Data** tab.

2 The block calculates a corresponding input or output value for the parameters you have specified. In units of dB and dBm,

$$\begin{aligned} P_{sat,\,out} + GC_{sat} &= P_{sat,\,in} + G_{lin} \\ P_{1dB,\,out} + 1 &= P_{1dB,\,in} + G_{lin} \\ OIP3 &= IIP3 + G_{lin} \end{aligned}$$

where G_{lin} is c_1 in units of dB.

3 The block formulates the coefficients c_3, c_5, and c_7, where applicable, as the solutions to a system of one, two, or three linear equations. The number of equations used is equal to the number of parameters you provide. For example, if you specify all three parameters, the block formulates the coefficients according to the following equations:

$$\sqrt{P_{sat,out}} = c_1\sqrt{P_{sat,in}} + c_3\left(\sqrt{P_{sat,in}}\right)^3 + c_5\left(\sqrt{P_{sat,in}}\right)^5 + c_7\left(\sqrt{P_{sat,in}}\right)^7$$

$$\sqrt{P_{1dB,out}} = c_1\sqrt{P_{1dB,in}} + c_3\left(\sqrt{P_{1dB,in}}\right)^3 + c_5\left(\sqrt{P_{1dB,in}}\right)^5 + c_7\left(\sqrt{P_{1dB,in}}\right)^7$$

$$0 = \frac{c_1}{IIP3} + c_3$$

The first two equations are the evaluation of the polynomial $F_{AM/AM}(s)$ at the points $(\sqrt{P_{sat,in}}, \sqrt{P_{sat,out}})$ and $(\sqrt{P_{1dB,in}}, \sqrt{P_{1dB,out}})$, expressed in linear units (such as W or mW) and normalized to a 1-Ω impedance. The third equation is the definition of the third-order intercept point.

The calculation omits higher-order terms according to the available degrees of freedom of the system. If you specify only two of the three parameters, the block does not use the equation involving the parameter you did not specify, and eliminates any c_7 terms from the remaining equations. Similarly, if you provide only one of the parameters, the block uses only the solution to the equation involving that parameter and omits any c_5 or c_7 terms.

If you provide vectors of nonlinearity and frequency data, the block calculates the polynomial coefficients using values for the parameters interpolated at the center frequency.

Parameters

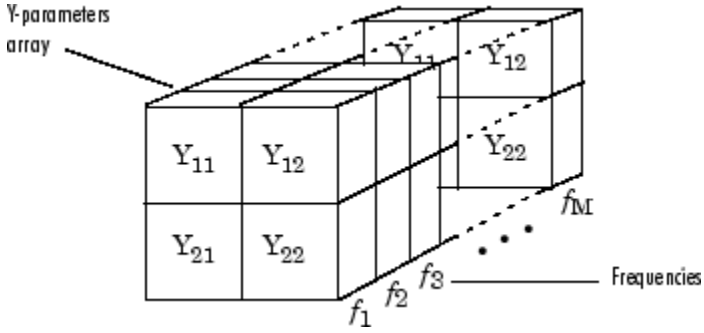
Main Tab

Y-Parameters

Y-parameters for a nonlinear mixer in a 2-by-2-by-M array. M is the number of Y-parameters.

Frequency (Hz)

Frequencies of the Y-parameters as an M-element vector. The order of the frequencies must correspond to the order of the Y-parameters in **Y-Parameters**. All frequencies must be positive. The following figure shows the correspondence between the Y-parameters array and the vector of frequencies.



Interpolation method

The method used to interpolate the network parameters. The following table lists the available methods describes each one.

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

Mixer Type

Type of mixer. Choices are `Downconverter` (default) and `Upconverter`.

LO frequency (Hz)

Local oscillator frequency. If you choose `Downconverter`, the blockset computes the mixer output frequency, f_{out}, from the mixer input frequency, f_{in}, and the local oscillator frequency, f_{lo}, as $f_{out} = f_{in} - f_{lo}$. If you choose `Upconverter`, $f_{out} = f_{in} + f_{lo}$.

Note For a downconverting mixer, the local oscillator frequency must satisfy the condition $f_{in} - f_{lo} \geq 1/(2t_s)$, where t_s is the sample time specified in the Input Port block. Otherwise, an error appears.

Noise Data Tab

Phase noise frequency offset (Hz)

Vector specifying the frequency offset.

Phase noise level (dBc/Hz)

Vector specifying the phase noise level.

Noise type

Type of noise data. The value can be `Noise figure`, `Spot noise data`, `Noise factor`, or `Noise temperature`. This parameter is disabled if the data source contains noise data.

Noise figure (dB)

Scalar ratio or vector of ratios, in decibels, of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Noise figure`.

Minimum noise figure (dB)

Minimum scalar ratio or vector of minimum ratios of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Spot noise data`.

Optimal reflection coefficient

Optimal mixer source impedance. This parameter is enabled if **Noise type** is set to `Spot noise data`. The value can be a scalar or vector.

Equivalent normalized resistance

Resistance or vector of resistances normalized to the resistance value or values used to take the noise measurement. This parameter is enabled if **Noise type** is set to `Spot noise data`.

Noise factor

Scalar ratio or vector of ratios of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Noise factor`.

Noise temperature (K)

Equivalent temperature or vector of temperatures that produce the same amount of noise power as the mixer. This parameter is enabled if **Noise type** is set to `Noise temperature`.

Frequency (Hz)

Scalar value or vector corresponding to the domain of frequencies over which you are specifying the noise data. If you provide a scalar value for your noise data, the block ignores the **Frequency (Hz)** parameter and uses the noise data for all frequencies. If you provide a vector of values for your noise data, it must be the same size as the

vector of frequencies. The block uses the **Interpolation method** specified in the **Main** tab to interpolate noise data.

Nonlinearity Data Tab

IP3 type

Type of third-order intercept point. The value can be `IIP3` (input intercept point) or `OIP3` (output intercept point). This parameter is disabled if the data source contains power data or IP3 data.

IP3 (dBm)

Value of third-order intercept point. This parameter is disabled if the data source contains power data or IP3 data. Use the default value, `Inf`, if you do not know the IP3 value. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

1 dB gain compression power (dBm)

Output power value ($P_{1dB,\,out}$) at which gain has decreased by 1 dB. This parameter is disabled if the data source contains power data or 1-dB compression point data. Use the default value, `Inf`, if you do not know the 1-dB compression point. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Output saturation power (dBm)

Output power value ($P_{sat,\,out}$) that the mixer produces when fully saturated. This parameter is disabled if the data source contains output saturation power data. Use the default value, `Inf`, if you do not know the saturation power. If you specify this parameter, you must also specify the **Gain compression at saturation (dB)**. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Gain compression at saturation (dB)

Decrease in gain (GC_{sat}) when the power is fully saturated. The block ignores this parameter if you do not specify the **Output saturation power (dBm)**. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Frequency (Hz)

Scalar or vector value of frequency points corresponding to the third-order intercept and power data. This parameter is disabled if the data source contains power data or IP3 data. If you use a scalar value, the **IP3 (dBm)**, **1 dB gain compression power**

(dBm), and **Output saturation power (dBm)** parameters must all be scalars. If you use a vector value, one or more of the **IP3 (dBm)**, **1 dB gain compression power (dBm)**, and **Output saturation power (dBm)** parameters must also be a vector.

Visualization Tab

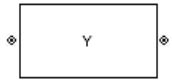
For information about plotting, see “Create Plots”.

See Also

General Mixer, Output Port, S-Parameters Mixer, Z-Parameters Mixer

Y-Parameters Passive Network

Model passive network using Y-parameters



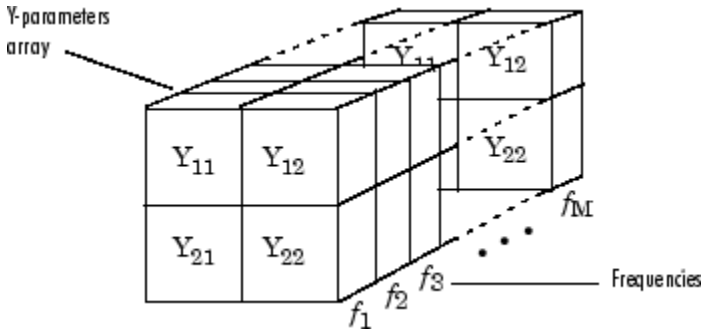
Library

Black Box Elements sublibrary of the Physical library

Description

The Y-Parameters Passive Network block models the two-port passive network described in the block dialog box, in terms of its Y-parameters and their associated frequencies.

In the **Y-Parameters** field of the block dialog box, provide the Y-parameters for each of M frequencies as a 2-by-2-by-M array. In the **Frequency** field, specify the frequencies for the Y-parameters as an M-element vector. The elements of the vector must be in the same order as the Y-parameters. All frequencies must be positive. For example, the following figure shows the correspondence between the Y-parameters array and the vector of frequencies.



The Y-Parameters Passive Network block uses the RF Toolbox `y2s` function to convert the Y-parameters to S-parameters, and then interpolates the resulting S-parameters to

determine their values at the modeling frequencies. The modeling frequencies are determined by the Output Port block. See “Map Network Parameters to Modeling Frequencies” for more details.

Parameters

Main Tab

Y-Parameters

Y-parameters for a two-port passive network in a 2-by-2-by-M array. M is the number of Y-parameters.

Frequency (Hz)

Frequencies of the Y-parameters as an M-element vector. The order of the frequencies must correspond to the order of the Y-parameters in **Y-Parameters**. All frequencies must be positive.

Interpolation method

The method used to interpolate the network parameters. The following table lists the available methods describes each one.

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

Visualization Tab

For information about plotting, see “Create Plots”.

Examples

Plotting Parameters with the Y-Parameters Passive Network Block

The following example specifies Y-parameters [.23i, -.12i; -.12i, .23i] and [.02-.13i, -.02+.25i; -.02+.25i, .02-.13i] at frequencies 2.0 GHz and 2.1 GHz respectively. It uses the MATLAB `cat` function to create the 2-by-2-by-2 Y-parameters array.

```
cat(3,[.23i,-.12i;-.12i,.23i],...
      [.02-.13i,-.02+.25i;-.02+.25i, .02-.13i])
```

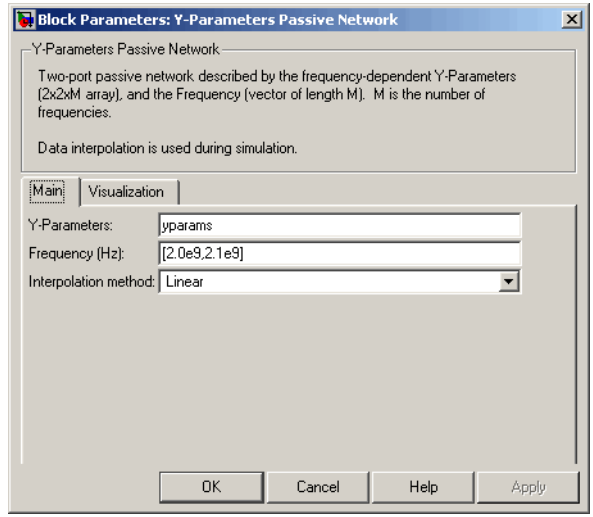
1. Type the following command at the MATLAB prompt to create a variable called `yparams` that stores the values of the Y-parameters.

   ```
   yparams = cat(3,[.23i,-.12i;-.12i,.23i],...
         [.02-.13i,-.02+.25i;-.02+.25i, .02-.13i])
   ```

2. Set the Y-Parameters Passive Network block parameters on the **Main** tab as follows:

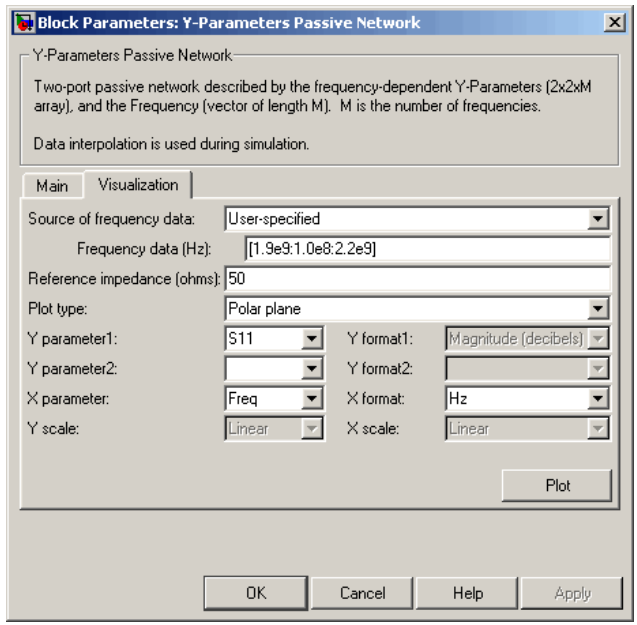
   - Set the **Y-Parameters** parameter to `yparams`.
   - Set the **Frequency (Hz)** parameter to `[2.0e9,2.1e9]`.

   Click **Apply**. This action applies the specified settings.

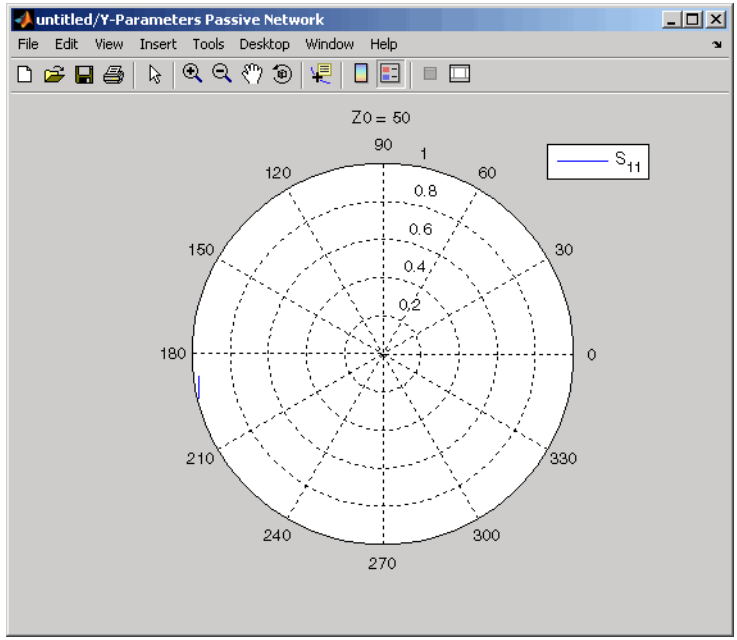


3 Set the Y-Parameters Passive Network block parameters on the **Visualization** tab as follows:

- In the **Source of frequency data** list, select `User-specified`.
- Set the **Frequency data (Hz)** parameter to `[1.9e9:1.0e8:2.2e9]`.
- In the **Plot type** list, select `Polar plane`.



Click **Plot**. This action creates a polar plane plot of the S_{11} parameters in the frequency range 1.9 to 2.2 GHz.



See Also

General Circuit Element, General Passive Network, Output Port, S-Parameters Passive Network, Z-Parameters Passive Network

`y2s` (RF Toolbox)

`interp1` (MATLAB)

Z-Parameters Amplifier

Model nonlinear amplifier using Z-parameters

Library

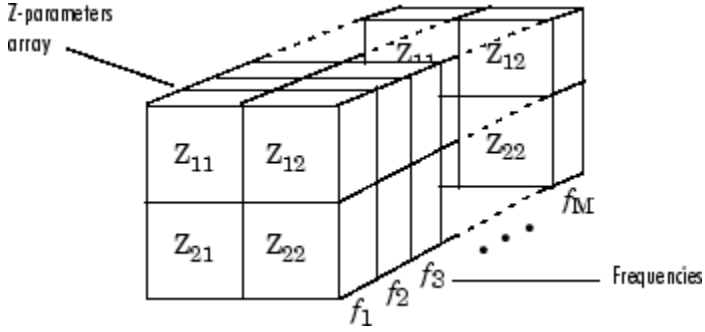
Amplifiers sublibrary of the Physical library

Description

The Z-Parameters Amplifier block models the nonlinear amplifier described in the block dialog box, in terms of its frequency-dependent Z-parameters, the frequencies of the Z-parameters, noise data, and nonlinearity data

Network Parameters

In the **Z-Parameters** field of the block dialog box, provide the Z-parameters for each of M frequencies as a 2-by-2-by-M array. In the **Frequency** field, specify the frequencies for the Z-parameters as an M-element vector. The elements of the frequencies vector must be in the same order as the Z-parameters. All frequencies must be positive. For example, the following figure shows the correspondence between the Z-parameters array and the vector of frequencies.



The Z-Parameters Amplifier block uses the RF Toolbox `z2s` function to convert the Z-parameters to S-parameters, and then interpolates the resulting S-parameters to determine their values at the modeling frequencies. See “Map Network Parameters to Modeling Frequencies” for more details.

Nonlinearity

You can introduce nonlinearities into your model by specifying parameters in the **Nonlinearity Data** tab of the Z-Parameters Amplifier block dialog box. Depending on which of these parameters you specify, the block computes up to four of the coefficients c_1, c_3, c_5, and c_7 of the polynomial

$$F_{AM/AM}(s) = c_1 s + c_3|s|^2 s + c_5|s|^4 s + c_7|s|^6 s$$

that determines the AM/AM conversion for the input signal s. The block automatically calculates c_1, the linear gain term. If you do not specify additional nonlinearity data, the block operates as a linear amplifier. If you do, the block calculates one or more of the remaining coefficients as the solution to a system of linear equations, determined by the following method.

1 The block checks whether you have specified a value other than `Inf` for:

- The third-order intercept point ($OIP3$ or $IIP3$).
- The output power at the 1-dB compression point ($P_{1dB,\,out}$).
- The output power at saturation ($P_{sat,\,out}$).

In addition, if you have specified $P_{sat,\,out}$, the block uses the value for the gain compression at saturation (GC_{sat}). Otherwise, GC_{sat} is not used. You define each of these parameters in the block dialog box, on the **Nonlinearity Data** tab.

2 The block calculates a corresponding input or output value for the parameters you have specified. In units of dB and dBm,

$$\begin{aligned} P_{sat,\,out} + GC_{sat} &= P_{sat,\,in} + G_{lin} \\ P_{1dB,\,out} + 1 &= P_{1dB,\,in} + G_{lin} \\ OIP3 &= IIP3 + G_{lin} \end{aligned}$$

where G_{lin} is c_1 in units of dB.

3 The block formulates the coefficients c_3, c_5, and c_7, where applicable, as the solutions to a system of one, two, or three linear equations. The number of equations used is equal to the number of parameters you provide. For example, if you specify all three parameters, the block formulates the coefficients according to the following equations:

$$\sqrt{P_{sat,out}} = c_1\sqrt{P_{sat,in}} + c_3\left(\sqrt{P_{sat,in}}\right)^3 + c_5\left(\sqrt{P_{sat,in}}\right)^5 + c_7\left(\sqrt{P_{sat,in}}\right)^7$$

$$\sqrt{P_{1dB,out}} = c_1\sqrt{P_{1dB,in}} + c_3\left(\sqrt{P_{1dB,in}}\right)^3 + c_5\left(\sqrt{P_{1dB,in}}\right)^5 + c_7\left(\sqrt{P_{1dB,in}}\right)^7$$

$$0 = \frac{c_1}{IIP3} + c_3$$

The first two equations are the evaluation of the polynomial $F_{AM/AM}(s)$ at the points $(\sqrt{P_{sat,in}}, \sqrt{P_{sat,out}})$ and $(\sqrt{P_{1dB,in}}, \sqrt{P_{1dB,out}})$, expressed in linear units (such as W or mW) and normalized to a 1-Ω impedance. The third equation is the definition of the third-order intercept point.

The calculation omits higher-order terms according to the available degrees of freedom of the system. If you specify only two of the three parameters, the block does not use the equation involving the parameter you did not specify, and eliminates any c_7 terms from the remaining equations. Similarly, if you provide only one of the parameters, the block uses only the solution to the equation involving that parameter and omits any c_5 or c_7 terms.

If you provide vectors of nonlinearity and frequency data, the block calculates the polynomial coefficients using values for the parameters interpolated at the center frequency.

Active Noise

You can specify active block noise in one of the following ways:

- Spot noise data in the Z-Parameters Amplifier block dialog box.
- Noise figure, noise factor, or noise temperature value in the Z-Parameters Amplifier block dialog box.

If you specify block noise as spot noise data, the block uses the data to calculate noise figure. The block first interpolates the noise data for the modeling frequencies, using the specified **Interpolation method**. It then calculates the noise figure using the resulting values.

Parameters

Main Tab

Z-Parameters

Z-parameters for a nonlinear amplifier in a 2-by-2-by-M array. M is the number of Z-parameters.

Frequency (Hz)

Frequencies of the Z-parameters as an M-element vector. The order of the frequencies must correspond to the order of the Z-parameters in **Z-Parameters**. All frequencies must be positive.

Interpolation method

The method used to interpolate the network parameters. The following table lists the available methods describes each one.

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

Noise Data Tab

Noise type

Type of noise data. The value can be `Noise figure`, `Spot noise data`, `Noise factor`, or `Noise temperature`. This parameter is disabled if the data source contains noise data.

Noise figure (dB)

Scalar ratio or vector of ratios, in decibels, of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Noise figure`.

Minimum noise figure (dB)

Minimum scalar ratio or vector of minimum ratios of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Spot noise data`.

Optimal reflection coefficient

Optimal amplifier source impedance. This parameter is enabled if **Noise type** is set to `Spot noise data`. The value can be a scalar or vector.

Equivalent normalized resistance

Resistance or vector of resistances normalized to the resistance value or values used to take the noise measurement. This parameter is enabled if **Noise type** is set to `Spot noise data`.

Noise factor

Scalar ratio or vector of ratios of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Noise factor`.

Noise temperature (K)

Equivalent temperature or vector of temperatures that produce the same amount of noise power as the amplifier. This parameter is enabled if **Noise type** is set to `Noise temperature`.

Frequency (Hz)

Scalar value or vector corresponding to the domain of frequencies over which you are specifying the noise data. If you provide a scalar value for your noise data, the block ignores the **Frequency (Hz)** parameter and uses the noise data for all frequencies. If you provide a vector of values for your noise data, it must be the same size as the vector of frequencies. The block uses the **Interpolation method** specified in the **Main** tab to interpolate noise data.

Nonlinearity Data Tab

IP3 type

Type of third-order intercept point. The value can be `IIP3` (input intercept point) or `OIP3` (output intercept point). This parameter is disabled if the data source contains power data or IP3 data.

IP3 (dBm)

Value of third-order intercept point. This parameter is disabled if the data source contains power data or IP3 data. Use the default value, `Inf`, if you do not know the IP3 value. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

1 dB gain compression power (dBm)

Output power value ($P_{1dB,\,out}$) at which gain has decreased by 1 dB. This parameter is disabled if the data source contains power data or 1-dB compression point data. Use the default value, `Inf`, if you do not know the 1-dB compression point. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Output saturation power (dBm)

Output power value ($P_{sat,\,out}$) that the amplifier produces when fully saturated. This parameter is disabled if the data source contains output saturation power data. Use the default value, `Inf`, if you do not know the saturation power. If you specify this parameter, you must also specify the **Gain compression at saturation (dB)**. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Gain compression at saturation (dB)

Decrease in gain (GC_{sat}) when the power is fully saturated. The block ignores this parameter if you do not specify the **Output saturation power (dBm)**. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Frequency (Hz)

Scalar or vector value of frequency points corresponding to the third-order intercept and power data. This parameter is disabled if the data source contains power data or IP3 data. If you use a scalar value, the **IP3 (dBm)**, **1 dB gain compression power (dBm)**, and **Output saturation power (dBm)** parameters must all be scalars. If you use a vector value, one or more of the **IP3 (dBm)**, **1 dB gain compression power (dBm)**, and **Output saturation power (dBm)** parameters must also be a vector.

Visualization Tab

For information about plotting, see “Create Plots”.

Examples

Plotting Parameters with the Z-Parameters Amplifier Block

The following example specifies Z-parameters [12.60+3.80i, 3.77-0.17i; 80.02+54.68i, 26.02+3.84i] and [15.12+3.55i, 4.14-0.92i; 92.10+23.67i, 27.59+2.71i] at frequencies

2.0 GHz and 2.1 GHz respectively. It uses the MATLAB `cat` function to create the 2-by-2-by-2 Z-parameters array.

```
cat(3,...
   [12.60+3.80i, 3.77-0.17i; 80.02+54.68i, 26.02+3.84i],...
   [15.12+3.55i, 4.14-0.92i; 92.10+23.67i, 27.59+2.71i])
```

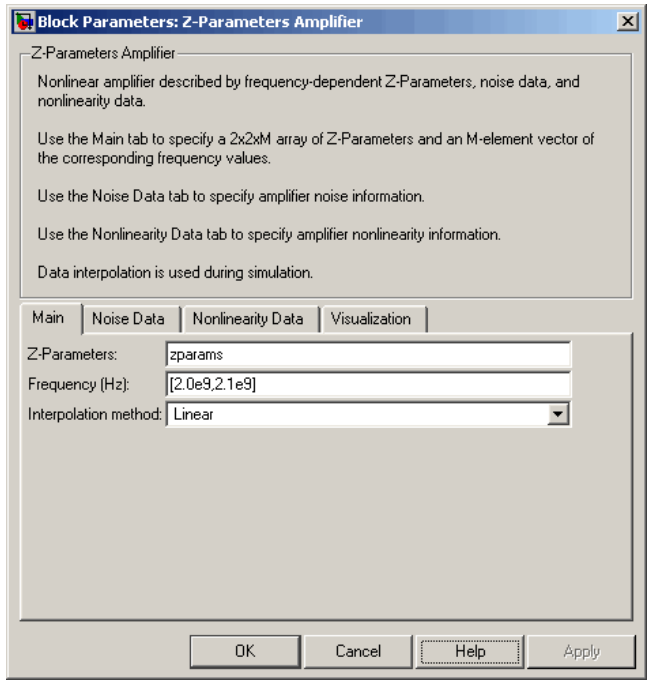
1. Type the following command at the MATLAB prompt to create a variable called `zparams` that stores the values of the Z-parameters.

   ```
   zparams = cat(3,...
    [12.60+3.80i, 3.77-0.17i; 80.02+54.68i, 26.02+3.84i],...
    [15.12+3.55i, 4.14-0.92i; 92.10+23.67i, 27.59+2.71i])
   ```

2. Set the Z-Parameters Amplifier block parameters on the **Main** tab as follows:

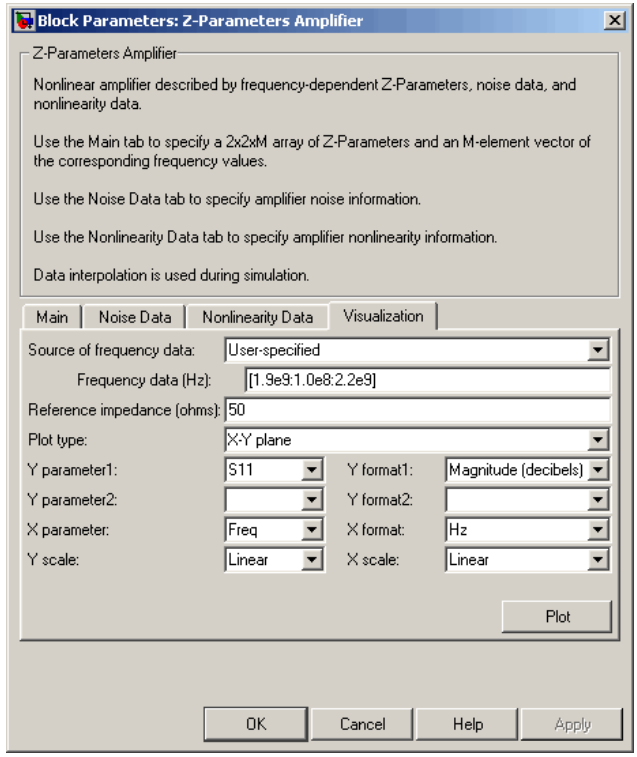
   - Set the **Z-Parameters** parameter to `zparams`.
   - Set the **Frequency (Hz)** parameter to `[2.0e9,2.1e9]`.

   Click **Apply**. This action applies the specified settings.

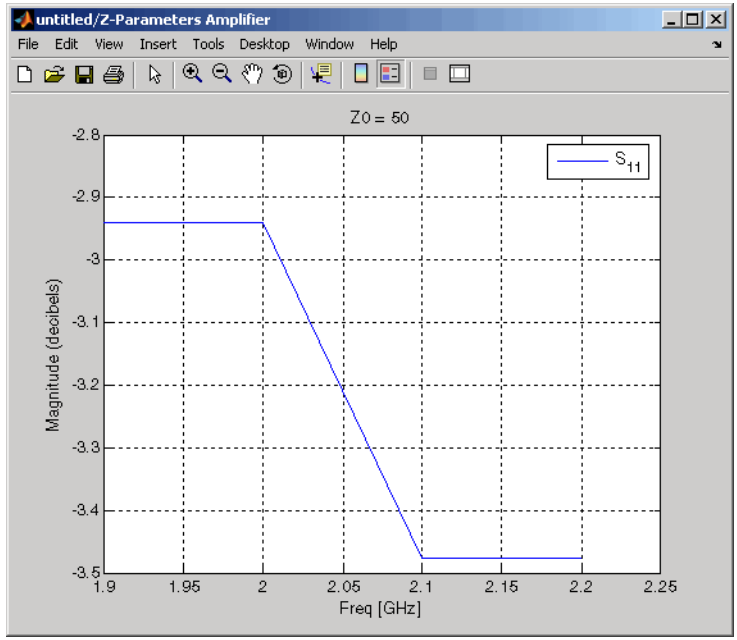


3 Set the Z-Parameters Amplifier block parameters on the **Visualization** tab as follows:

- In the **Source of frequency data** list, select `User-specified`.
- Set the **Frequency data (Hz)** parameter to `[1.9e9:1.0e8:2.2e9]`.
- In the **Plot type** list, select `X-Y plane`.
- In the **Y parameter1** list, select `S11`.



Click **Plot**. This action creates an X-Y Plane plot of the S_{11} parameters in the frequency range 1.9 to 2.2 GHz.



See Also

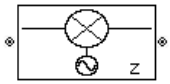
General Amplifier, Output Port, S-Parameters Amplifier, Y-Parameters Amplifier

`z2s` (RF Toolbox)

`interp1` (MATLAB)

Z-Parameters Mixer

Model mixer and local oscillator using Z-parameters



Library

Mixer sublibrary of the Physical library

Description

The Z-Parameters Mixer block models the nonlinear mixer described in the block dialog box in terms of its frequency-dependent Z-parameters, the frequencies of the Z-parameters, noise data (including phase noise data), and nonlinearity data.

Network Parameters

The Z-parameter values all refer to the mixer input frequency.

The Z-Parameters Mixer block uses the RF Toolbox `z2s` function to convert the Z-parameters to S-parameters and then interpolates the resulting S-parameters to determine their values at the modeling frequencies. See “Map Network Parameters to Modeling Frequencies” for more details.

RF Blockset Equivalent Baseband software computes the reflected wave at the mixer input (b_1) and at the mixer output (b_2) from the interpolated S-parameters as

$$\begin{bmatrix} b_1(f_{in}) \\ b_2(f_{out}) \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} a_1(f_{in}) \\ a_2(f_{out}) \end{bmatrix}$$

where

- f_{in} and f_{out} are the mixer input and output frequencies, respectively.

- a_1 and a_2 are the incident waves at the mixer input and output, respectively.

The interpolated S_{21} parameter values describe the conversion gain as a function of frequency, referred to the mixer input frequency.

Active Noise

You can specify active block noise in one of the following ways:

- Spot noise data in the Z-Parameters Mixer block dialog box.
- Noise figure, noise factor, or noise temperature value in the Z-Parameters Mixer block dialog box.

If you specify block noise as spot noise data, the block uses the data to calculate noise figure. The block first interpolates the noise data for the modeling frequencies, using the specified **Interpolation method**. It then calculates the noise figure using the resulting values.

Phase Noise

The Z-Parameters Mixer block applies phase noise to a complex baseband signal. The block first generates additive white Gaussian noise (AWGN) and filters the noise with a digital FIR filter. It then adds the resulting noise to the angle component of the input signal.

The blockset computes the digital filter by:

1. Interpolating the specified phase noise level to determine the phase noise values at the modeling frequencies.
2. Taking the IFFT of the resulting phase noise spectrum to get the coefficients of the FIR filter.

Note If you specify phase noise as a scalar value, the blockset assumes that the phase noise is the phase noise is constant at all modeling frequencies and does not have a *1/f* slope. This assumption differs from that made by the Mathematical Mixer block.

Nonlinearity

You can introduce nonlinearities into your model by specifying parameters in the **Nonlinearity Data** tab of the Z-Parameters Mixer block dialog box. Depending on which of these parameters you specify, the block computes up to four of the coefficients c_1, c_3, c_5, and c_7 of the polynomial

$$F_{AM/AM}(s) = c_1 s + c_3|s|^2 s + c_5|s|^4 s + c_7|s|^6 s$$

that determines the AM/AM conversion for the input signal s. The block automatically calculates c_1, the linear gain term. If you do not specify additional nonlinearity data, the block operates as a mixer with a linear gain. If you do, the block calculates one or more of the remaining coefficients as the solution to a system of linear equations, determined by the following method.

1 The block checks whether you have specified a value other than `Inf` for:

- The third-order intercept point ($OIP3$ or $IIP3$).
- The output power at the 1-dB compression point ($P_{1dB,\,out}$).
- The output power at saturation ($P_{sat,\,out}$).

In addition, if you have specified $P_{sat,\,out}$, the block uses the value for the gain compression at saturation (GC_{sat}). Otherwise, GC_{sat} is not used. You define each of these parameters in the block dialog box, on the **Nonlinearity Data** tab.

2 The block calculates a corresponding input or output value for the parameters you have specified. In units of dB and dBm,

$$\begin{aligned} P_{sat,\,out} + GC_{sat} &= P_{sat,\,in} + G_{lin} \\ P_{1dB,\,out} + 1 &= P_{1dB,\,in} + G_{lin} \\ OIP3 &= IIP3 + G_{lin} \end{aligned}$$

where G_{lin} is c_1 in units of dB.

3 The block formulates the coefficients c_3, c_5, and c_7, where applicable, as the solutions to a system of one, two, or three linear equations. The number of equations used is equal to the number of parameters you provide. For example, if you specify all three parameters, the block formulates the coefficients according to the following equations:

$$\sqrt{P_{sat,out}} = c_1\sqrt{P_{sat,in}} + c_3\left(\sqrt{P_{sat,in}}\right)^3 + c_5\left(\sqrt{P_{sat,in}}\right)^5 + c_7\left(\sqrt{P_{sat,in}}\right)^7$$

$$\sqrt{P_{1dB,out}} = c_1\sqrt{P_{1dB,in}} + c_3\left(\sqrt{P_{1dB,in}}\right)^3 + c_5\left(\sqrt{P_{1dB,in}}\right)^5 + c_7\left(\sqrt{P_{1dB,in}}\right)^7$$

$$0 = \frac{c_1}{IIP3} + c_3$$

The first two equations are the evaluation of the polynomial $F_{AM/AM}(s)$ at the points $(\sqrt{P_{sat,in}}, \sqrt{P_{sat,out}})$ and $(\sqrt{P_{1dB,in}}, \sqrt{P_{1dB,out}})$, expressed in linear units (such as W or mW) and normalized to a 1-Ω impedance. The third equation is the definition of the third-order intercept point.

The calculation omits higher-order terms according to the available degrees of freedom of the system. If you specify only two of the three parameters, the block does not use the equation involving the parameter you did not specify, and eliminates any c_7 terms from the remaining equations. Similarly, if you provide only one of the parameters, the block uses only the solution to the equation involving that parameter and omits any c_5 or c_7 terms.

If you provide vectors of nonlinearity and frequency data, the block calculates the polynomial coefficients using values for the parameters interpolated at the center frequency.

Parameters

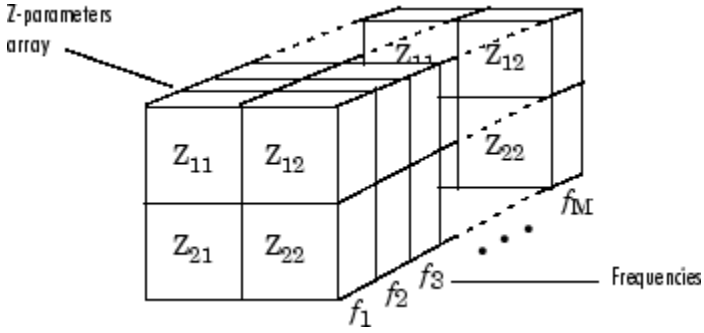
Main Tab

Z-Parameters

Z-parameters for a nonlinear mixer in a 2-by-2-by-M array. M is the number of Z-parameters.

Frequency (Hz)

Frequencies of the Z-parameters as an M-element vector. The order of the frequencies must correspond to the order of the Z-parameters in **Z-Parameters**. All frequencies must be positive. The following figure shows the correspondence between the Z-parameters array and the vector of frequencies.



Interpolation method

The method used to interpolate the network parameters. The following table lists the available methods describes each one.

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

Mixer Type

Type of mixer. Choices are `Downconverter` (default) and `Upconverter`.

LO frequency (Hz)

Local oscillator frequency. If you choose `Downconverter`, the blockset computes the mixer output frequency, f_{out}, from the mixer input frequency, f_{in}, and the local oscillator frequency, f_{lo}, as $f_{out} = f_{in} - f_{lo}$. If you choose `Upconverter`, $f_{out} = f_{in} + f_{lo}$.

Note For a downconverting mixer, the local oscillator frequency must satisfy the condition $f_{in} - f_{lo} \geq 1/(2t_s)$, where t_s is the sample time specified in the Input Port block. Otherwise, an error appears.

Noise Data Tab

Phase noise frequency offset (Hz)

Vector specifying the frequency offset.

Phase noise level (dBc/Hz)

Vector specifying the phase noise level.

Noise type

Type of noise data. The value can be `Noise figure`, `Spot noise data`, `Noise factor`, or `Noise temperature`. This parameter is disabled if the data source contains noise data.

Noise figure (dB)

Scalar ratio or vector of ratios, in decibels, of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Noise figure`.

Minimum noise figure (dB)

Minimum scalar ratio or vector of minimum ratios of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Spot noise data`.

Optimal reflection coefficient

Optimal mixer source impedance. This parameter is enabled if **Noise type** is set to `Spot noise data`. The value can be a scalar or vector.

Equivalent normalized resistance

Resistance or vector of resistances normalized to the resistance value or values used to take the noise measurement. This parameter is enabled if **Noise type** is set to `Spot noise data`.

Noise factor

Scalar ratio or vector of ratios of the available signal-to-noise power ratio at the input to the available signal-to-noise power ratio at the output, $(S_i/N_i)/(S_o/N_o)$. This parameter is enabled if **Noise type** is set to `Noise factor`.

Noise temperature (K)

Equivalent temperature or vector of temperatures that produce the same amount of noise power as the mixer. This parameter is enabled if **Noise type** is set to `Noise temperature`.

Frequency (Hz)

Scalar value or vector corresponding to the domain of frequencies over which you are specifying the noise data. If you provide a scalar value for your noise data, the block ignores the **Frequency (Hz)** parameter and uses the noise data for all frequencies. If you provide a vector of values for your noise data, it must be the same size as the

vector of frequencies. The block uses the **Interpolation method** specified in the **Main** tab to interpolate noise data.

Nonlinearity Data Tab

IP3 type

Type of third-order intercept point. The value can be `IIP3` (input intercept point) or `OIP3` (output intercept point). This parameter is disabled if the data source contains power data or IP3 data.

IP3 (dBm)

Value of third-order intercept point. This parameter is disabled if the data source contains power data or IP3 data. Use the default value, `Inf`, if you do not know the IP3 value. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

1 dB gain compression power (dBm)

Output power value ($P_{1dB,\,out}$) at which gain has decreased by 1 dB. This parameter is disabled if the data source contains power data or 1-dB compression point data. Use the default value, `Inf`, if you do not know the 1-dB compression point. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Output saturation power (dBm)

Output power value ($P_{sat,\,out}$) that the mixer produces when fully saturated. This parameter is disabled if the data source contains output saturation power data. Use the default value, `Inf`, if you do not know the saturation power. If you specify this parameter, you must also specify the **Gain compression at saturation (dB)**. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Gain compression at saturation (dB)

Decrease in gain (GC_{sat}) when the power is fully saturated. The block ignores this parameter if you do not specify the **Output saturation power (dBm)**. This parameter can be a scalar (to specify frequency-independent nonlinearity data) or a vector (to specify frequency-dependent nonlinearity data).

Frequency (Hz)

Scalar or vector value of frequency points corresponding to the third-order intercept and power data. This parameter is disabled if the data source contains power data or IP3 data. If you use a scalar value, the **IP3 (dBm)**, **1 dB gain compression power**

(dBm), and **Output saturation power (dBm)** parameters must all be scalars. If you use a vector value, one or more of the **IP3 (dBm)**, **1 dB gain compression power (dBm)**, and **Output saturation power (dBm)** parameters must also be a vector.

Visualization Tab

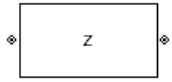
For information about plotting, see “Create Plots”.

See Also

General Mixer, Output Port, S-Parameters Mixer, Y-Parameters Mixer

Z-Parameters Passive Network

Model passive network using Z-parameters



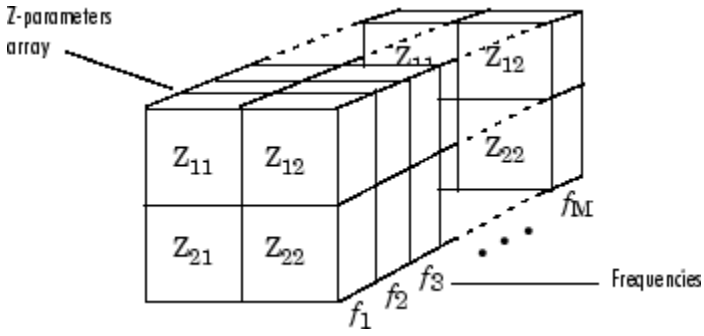
Library

Black Box Elements sublibrary of the Physical library

Description

The Z-Parameters Passive Network block models the two-port passive network described in the block dialog box, in terms of its Z-parameters and their associated frequencies.

In the **Z-Parameters** field of the block dialog box, provide the Z-parameters for each of M frequencies as a 2-by-2-by-M array. In the **Frequency** field, specify the frequencies for the Z-parameters as an M-element vector. The elements of the vector must be in the same order as the Z-parameters. All frequencies must be positive. For example, the following figure shows the correspondence between the Z-parameters array and the vector of frequencies.



The Z-Parameters Passive Network block uses the RF Toolbox `z2s` function to convert the Z-parameters to S-parameters, and then interpolates the resulting S-parameters to

determine their values at the modeling frequencies. The modeling frequencies are determined by the Output Port block. See “Map Network Parameters to Modeling Frequencies” for more details.

Parameters

Main Tab

Z-Parameters

Z-parameters for a two-port passive network in a 2-by-2-by-M array. M is the number of Z-parameters.

Frequency (Hz)

Frequencies of the Z-parameters as an M-element vector. The order of the frequencies must correspond to the order of the Z-parameters in **Z-Parameters**. All frequencies must be positive.

Interpolation method

The method used to interpolate the network parameters. The following table lists the available methods describes each one.

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

Visualization Tab

For information about plotting, see “Create Plots”.

Examples

Plotting Parameters with the Z-Parameters Passive Network Block

The following example specifies Z-parameters [0.13 - 5.93i, .03-3.16i; 0.03-3.16i, .13-5.93i] and [0.27-2.86i, -.09-5.41i; -.09-5.41i, .27-2.86i] at frequencies 2.0 GHz and 2.1 GHz respectively. It uses the MATLAB `cat` function to create the 2-by-2-by-2 Z-parameters array.

```
cat(3,[0.13-5.93i, .03-3.16i; 0.03-3.16i, .13-5.93i],...
      [0.27-2.86i,-.09-5.41i; -.09-5.41i, .27-2.86i])
```

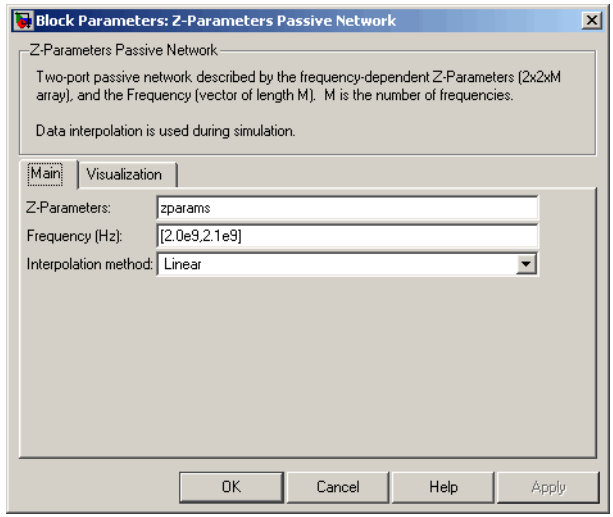
1. Type the following command at the MATLAB prompt to create a variable called `zparams` that stores the values of the Z-parameters.

   ```
   zparams = cat(3,...
         [0.13-5.93i,.03-3.16i; 0.03-3.16i,.13-5.93i],...
         [0.27-2.86i,-.09-5.41i; -.09-5.41i,.27-2.86i])
   ```

2. Set the Z-Parameters Passive Network block parameters on the **Main** tab as follows:

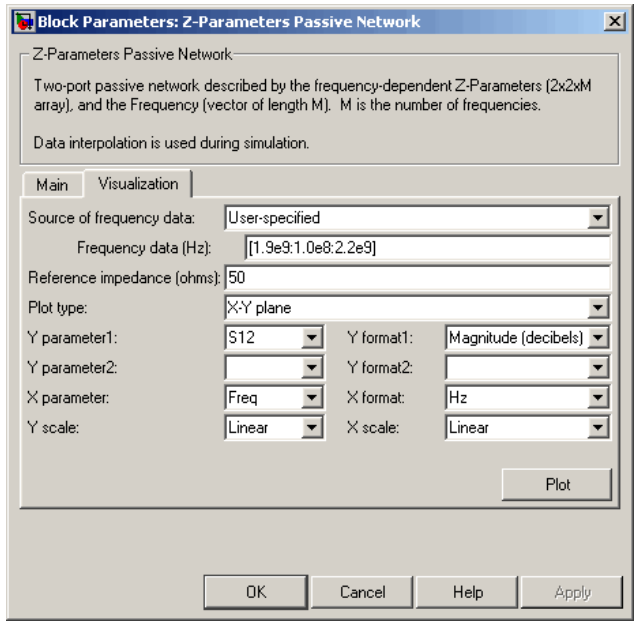
   - Set the **Z-Parameters** parameter to `zparams`.
   - Set the **Frequency (Hz)** parameter to `[2.0e9,2.1e9]`.

   Click **Apply**. This action applies the specified settings.

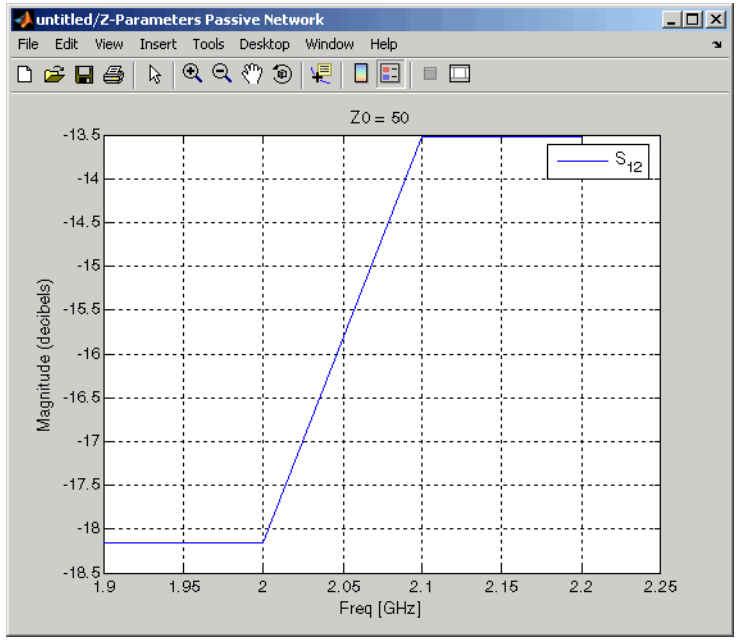


3 Set the Z-Parameters Passive Network block parameters on the **Visualization** tab as follows:

- In the **Source of frequency data** list, select `User-specified`.
- Set the **Frequency data (Hz)** parameter to `[1.9e9:1.0e8:2.2e9]`.
- In the **Y parameter1** list, select `S12`.



Click **Plot**. This action creates an X-Y plane plot of the S_{12} parameters in the frequency range 1.9 to 2.2 GHz.



See Also

General Circuit Element, General Passive Network, Output Port, S-Parameters Passive Network, Y-Parameters Passive Network

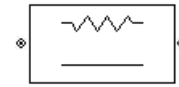
`z2s` (RF Toolbox)

`interp1` (MATLAB)

Series R

Model series resistor

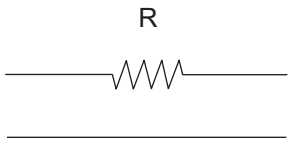
Library: RF Blockset / Equivalent Baseband / Series / Shunt RLC



Description

The Series R block models the series resistor described in the block dialog box, in terms of its frequency-dependent S-parameters.

The series R object is a two-port network, as shown in the following circuit diagram.



Parameters

Main

`Resistance (ohms)` — Resistance value

`1` (default) | scalar

Resistance value, specified as a scalar.

Visualization

`Source of frequency data` — Source of frequency data

`User-Specified` (default)

Source of frequency data, specified as `User-Specified`.

`Frequency data (Hz)` — Frequencies

`[1:1.0e5:4e6]` (default) | vector

Frequencies, specified as vector in hertz..

`Reference impedance (ohms)` — Impedance to use when plotting small-signal parameters
`50` (default) | scalar

Impedance to use when plotting small-signal parameters, specified as scalar in ohms.

`Plot type` — Plot type
`X-Y plane` (default) | `Composite data` | `Polar plane` | `Z Smith chart` | `Y Smith chart` | `ZY Smith chart`

Plot type, specified as `Composite data`, `X-Y plane`, `Polar plane`, `Z Smith chart`, `Y Smith chart`, or `ZY Smith chart`. For more information see, “Create Plots”.

`Y Parameter 1` — First parameter on Y-axis
`S11` (default) | `S12` | `S21` | `S22` | `GroupDelay` | `OIP3` | `NF` | `NFactor` | `NTemp`

First parameter on Y-axis, specified as `S11`, `S12`, `S21`, `S22`, `GroupDelay`, `OIP3`, `NF`, `NFactor`, or `NTemp`.

`Y Parameter 2` — Second parameter on Y-axis
`S11` (default) | `S12` | `S21` | `S22` | `GroupDelay` | `OIP3` | `NF` | `NFactor` | `NTemp`

Second parameter on Y-axis (optional), specified as `S11`, `S12`, `S21`, `S22`, `GroupDelay`, `OIP3`, `NF`, `NFactor`, or `NTemp`.

`Y Format1` — Plot format of first Y-parameter
`Magnitude (decibels)` (default) | `Magnitude (linear)` | `Angle(degrees)` | `Angle(radians)` | `Real` | `Imaginary`

Plot format, specified as `Magnitude (decibels)`, `Angle(degrees)`, `Real`, or `Imaginary`.

`Format2` — Plot format
`Magnitude (decibels)` (default) | `Angle(degrees)` | `Real` | `Imaginary`

Plot format, specified as `Magnitude (decibels)`, `Angle(degrees)`, `Real`, or `Imaginary`.

`Y-axis scale` — Y-axis scale
`Linear` (default) | `Logarithmic`

Y-axis scale, specified as `Linear` or `Logarithmic`.

`X-axis scale` — X-axis scale
`Linear` (default) | `Logarithmic`

X-axis scale, specified as `Linear` or `Logarithmic`.

`Plot` — Plot specified data
button

Plot specified data using plot button.

See Also

Inductor | Three-Winding Transformer

Introduced before R2006a

Functions — Alphabetical List

simrfSupportPackages

Start Add-On Explorer to download, install, or uninstall RF Blockset models and supporting software for third–party hardware

Syntax

```
simrfSupportPackages
```

Description

`simrfSupportPackages` opens the Add-On Explorer to install the Analog Devices models and testbenches.

Examples

Install RF Blockset System Models

1. Start Add-On Explorer using:

   ```
   simrfSupportPackages
   ```
2. Click `SimRF Models for Analog Devices Rf Transceivers`.
3. Click **Install** and follow the instructions to install the models.
4. Complete the installation process. You can now access the following Analog Devices® models:

AD9361 models
AD9361 testbenches
AD9371 models
AD9371 testbenches

Note To run the models, you require these additional licenses:

- Communications Toolbox
- Stateflow®
- Fixed-Point Designer™
- DSP System Toolbox
- Simscape

When you update your MATLAB software, you must repeat the installation process for the latest updates. You can also check for updates between releases.

Introduced in R2014b

SpectrumAnalyzerConfiguration

Configure Spectrum Analyzer for programmatic access

Description

The `spbscopes.SpectrumAnalyzerConfiguration` object contains the scope configuration information for the Spectrum Analyzer block.

Creation

`MyScopeConfiguration = get_param(gcbh,'ScopeConfiguration')` constructs a new Spectrum Analyzer Configuration object. You must first select the block in the model or give the full path to the block.

Properties

Frequently Used

`NumInputPorts` — Number of input ports

`"1"` (default) | character vector | string scalar

Number of input ports on a scope block, specified by a character vector or string scalar. Maximum number of input ports is 96.

UI Use

Select **File** > **Number of Input Ports**.

Data Types: `char` | `string`

`SpectrumType` — Type of spectrum to show

`"Power"` (default) | `"Power density"` | `"RMS"`

Specify the spectrum type to display.

`"Power"` — Power spectrum

`"Power density"` — Power spectral density. The power spectral density is the magnitude squared of the spectrum normalized to a bandwidth of 1 hertz.

`"RMS"` — Root mean square. The root-mean-square shows the square root of the sum of the squared means. This option is useful when viewing the frequency of voltage or current signals.

Tunable: Yes

UI Use

Open the **Spectrum Settings**. In the **Main options** section, set **Type**.

Data Types: `char` | `string`

`SampleRateSource` — Source of input sample rate

`"Inherited"` (default) | `"Property"`

Specify the source of the input sample rate as:

- `"Inherited"` — Spectrum Analyzer inherits the input sample rate from the model.
- `"Property"` — Specify the sample rate input directly using the `SampleRate` property.

UI Use

Open the **Spectrum Settings**. In the **Main options** section, in the **Sample rate (Hz)** combo box, enter a custom sample rate or select `Inherited`.

Data Types: `char` | `string`

`SampleRate` — Sample rate of input

`"10e3"` (default) | character vector | string scalar

Specify the sample rate of the input signals in hertz as a character vector or string scalar.

Dependency

To enable this property, set `SampleRateSource` to `"Property"`.

UI Use

Open the **Spectrum Settings**. In the **Main options** section, enter a **Sample rate (Hz)** in the combo box.

Data Types: `char` | `string`

`PlotAsTwoSidedSpectrum` — Two-sided spectrum flag

`false` (default) | `true`

- `true` — Compute and plot two-sided spectral estimates. When the input signal is complex-valued, you must set this property to `true`.
- `false` — Compute and plot one-sided spectral estimates. If you set this property to `false`, then the input signal must be real-valued.

  When this property is `false`, Spectrum Analyzer uses power-folding. The *y*-axis values are twice the amplitude that they would be if this property were set to `true`, except at `0` and the Nyquist frequency. A one-sided power spectral density (PSD) contains the total power of the signal in the frequency interval from DC to half of the Nyquist rate. For more information, see `pwelch`.

UI Use

Open the **Spectrum Settings**. In the **Trace options** section, select **Two-sided spectrum**.

Data Types: `logical`

`FrequencyScale` — Frequency scale

`"Linear"` (default) | `"Log"`

- `"Log"` — displays the frequencies on the *x*-axis on a logarithmic scale. To use the `"Log"` setting, you must also set the `PlotAsTwoSidedSpectrum` property to `false`.
- `"Linear"` — displays the frequencies on the *x*-axis on a linear scale. To use the `"Linear"` setting, you must also set the `PlotAsTwoSidedSpectrum` property to `true`.

Tunable: Yes

UI Use

Open the **Spectrum Settings**. In the **Trace options** section, set **Scale**.

Data Types: `char` | `string`

Advanced

RBWSource — Source of resolution bandwidth value

`"Auto"` (default) | `"Property"` | `"InputPort"`

Specify the source of the resolution bandwidth (RBW) as `"Auto"`, `"Property"`, or `"InputPort"`.

- `"Auto"` — The Spectrum Analyzer adjusts the spectral estimation resolution to ensure that there are 1024 RBW intervals over the defined frequency span.
- `"Property"` — Specify the resolution bandwidth directly using the RBW property.
- `"InputPort"` — An input port is added to the Spectrum Analyzer block to read the RBW. This option is only applicable to frequency input.

UI Use

Open the **Spectrum Settings**. In the **Frequency input options** section, set **RBW (Hz)**.

Data Types: `char` | `string`

RBW — Resolution bandwidth

`"9.76"` (default) | character vector | string scalar

RBW controls the spectral resolution of the Spectrum Analyzer. Specify the resolution bandwidth in hertz as a character vector or string scalar. You must specify a value to ensure that there are at least two RBW intervals over the specified frequency span. Thus, the ratio of the overall span to RBW must be greater than two:

$$\frac{span}{RBW} > 2$$

Dependency

To enable, set:

- RBWSource to `"Property"`

UI Use

Open the **Spectrum Settings**. In the **Main options** section, set **RBW (Hz)**.

Data Types: `char` | `string`

`OverlapPercent` — Overlap percentage

`"0"` (default) | character vector of a real scalar | string scalar of a real scalar

The percentage overlap between the previous and current buffered data segments, specified as a character vector or string scalar of a real scalar. The overlap creates a window segment that is used to compute a spectral estimate. The value must be greater than or equal to zero and less than 100.

UI Use

Open the **Spectrum Settings**. In the **Window options** section, set **Overlap (%)**.

Data Types: `char` | `string`

`Window` — Window function

`"Hann"` (default) | `"Rectangular"`

Specify a window function for the spectral estimation. The following table shows preset windows. For more information, follow the link to the corresponding function reference in the Signal Processing Toolbox documentation.

Window Option	Corresponding Signal Processing Toolbox Function
`"Rectangular"`	`rectwin`
`"Hann"`	`hann`

UI Use

Open the **Spectrum Settings**. In the **Window options** section, set **Window**.

Data Types: `char` | `string`

`SpectrumUnits` — Units of the spectrum

`"dBm"` (default)

This property is read-only.

Specify the units in which the Spectrum Analyzer displays power values. To change the spectrum units you must have DSP System Toolbox.

`AveragingMethod` — Smoothing method

`"Running"` (default) | `"Exponential"`

Specify the smoothing method as:

- `Running` — Running average of the last *n* samples. Use the `SpectralAverages` property to specify *n*.
- `Exponential` — Weighted average of samples. Use the `ForgettingFactor` property to specify the weighted forgetting factor.

For more information about the averaging methods, see “Averaging Method” (DSP System Toolbox).

UI Use

Open the **Spectrum Settings**. In the **Trace options** section, set **Averaging method**.

Data Types: `char` | `string`

`SpectralAverages` — Number of spectral averages

`"1"` (default) | character vector | string scalar

Specify the number of spectral averages as a character vector or string scalar. The Spectrum Analyzer computes the current power spectrum estimate by computing a running average of the last *N* power spectrum estimates. This property defines *N*.

Dependency

To enable this property, set `AveragingMethod` to `"Running"`.

UI Use

Open the **Spectrum Settings**. In the **Trace options** section, set **Averages**.

Data Types: `char` | `string`

`ForgettingFactor` — Weighting forgetting factor

`"0.9"` (default) | string scalar of scalar in the range (0,1] | character vector of scalar in the range (0,1]

Specify the exponential weighting as a scalar value greater than 0 and less than or equal to 1, specified as a string scalar or character vector.

Dependency

To enable this property, set `AveragingMethod` to `"Exponential"`.

UI Use

Open the **Spectrum Settings**. In the **Trace options** section, set **Forgetting factor**.

Data Types: `char` | `string`

`ReferenceLoad` — Reference load

`"1"` (default) | character vector of a real positive scalar | string scalar of a real positive scalar

The load the scope uses as a reference to compute power levels.

UI Use

Open the **Spectrum Settings**. In the **Trace options** section, set **Reference load**.

Data Types: `char` | `string`

`FrequencyOffset` — Frequency offset

`"0"` (default) | numeric scalar character vector | numeric vector character vector | numeric scalar string scalar | numeric vector string scalar

- Numeric scalar (specified as a character vector or string scalar) — Apply the same frequency offset to all channels, specified in hertz as a character vector.
- Numeric vector (specified as a character vector or string scalar) — Apply a specific frequency offset for each channel, specify a vector of frequencies. The vector length must be equal to number of input channels.

  The frequency-axis values are offset by the values specified in this property. The overall span must fall within the “Nyquist frequency interval” on page 1-197.

UI Use

Open the **Spectrum Settings**. In the **Trace options** section, set **Offset (Hz)**.

Data Types: `char` | `string`

`TreatMby1SignalsAsOneChannel` — Treat unoriented sample-based input signal as a column vector

`true` (default) | `false`

Set this property to `true` to treat *M*-by-1 and unoriented sample-based inputs as a column vector, or one channel. Set this property to `false` to treat *M*-by-1 and unoriented sample-based inputs as a 1-by-*M* row vector.

Data Types: `logical`

Measurements

`MeasurementChannel` — Channel for which measurements are obtained

`"1"` (default) | character vector | string scalar

Channel over which the measurements are obtained, specified as a character vector or a string scalar which evaluates to a positive integer greater than 0 and less than or equal to 100. The maximum number you can specify is the number of channels (columns) in the input signal.

Tunable: Yes

UI Use

Click on **Tools > Measurements** and open the **Trace Selection** settings.

Data Types: `char` | `string`

`PeakFinder` — Peak finder measurement

`PeakFinderSpecification` object

Enable peak finder to compute and display the largest calculated peak values. The `PeakFinder` property uses the `PeakFinderSpecification` properties.

The `PeakFinderSpecification` properties are:

- `MinHeight` –– Level above which peaks are detected, specified as a scalar value.

  Default: `-Inf`
- `NumPeaks` –– Maximum number of peaks to show, specified as a positive integer scalar less than 100.

  Default: `3`
- `MinDistance` –– Minimum number of samples between adjacent peaks, specified as a positive real scalar.

  Default: `1`
- `Threshold` –– Minimum height difference between peak and its neighboring samples, specified as a nonnegative real scalar.

Default: `0`

- `LabelFormat` -- Coordinates to display next to the calculated peak value, specified as a character vector or a string scalar. Valid values are `"X"`, `"Y"`, or `"X + Y"`.

  Default: `"X + Y"`
- `Enable` -- Set this property to `true` to enable peak finder measurements. Valid values are `true` or `false`.

  Default: `false`

All `PeakFinderSpecification` properties are tunable.

Tunable: Yes

UI Use

Open the **Peak Finder** pane () and modify the **Settings** options.

`CursorMeasurements` — Cursor measurements

`CursorMeasurementsSpecification` object

Enable cursor measurements to display screen or waveform cursors. The `CursorMeasurements` property uses the `CursorMeasurementsSpecification` properties.

The `CursorMeasurementsSpecification` properties are:

- `Type` -- Type of the display cursors, specified as either `"Screen cursors"` or `"Waveform cursors"`.

  Default: `"Waveform cursors"`
- `ShowHorizontal` -- Set this property to `true` to show the horizontal screen cursors. This property applies when you set the `Type` property to `"Screen cursors"`.

  Default: `true`
- `ShowVertical` -- Set this property to `true` to show the vertical screen cursors. This property applies when you set the `Type` property to `"Screen cursors"`.

  Default: `true`

- `Cursor1TraceSource` -- Specify the waveform cursor 1 source as a positive real scalar. This property applies when you set the `Type` property to `"Waveform cursors"`.

  Default: `1`

- `Cursor2TraceSource` -- Specify the waveform cursor 2 source as a positive real scalar. This property applies when you set the `Type` property to `"Waveform cursors"`.

  Default: `1`

- `LockSpacing` -- Lock spacing between cursors, specified as a logical scalar.

  Default: `false`

- `SnapToData` -- Snap cursors to data, specified as a logical scalar.

  Default: `true`

- `XLocation` -- *x*-coordinates of the cursors, specified as a real vector of length equal to 2.

  Default: `[-2500 2500]`

- `YLocation` -- *y*-coordinates of the cursors, specified as a real vector of length equal to 2. This property applies when you set the `Type` property to `"Screen cursors"`.

  Default: `[-55 5]`

- `Enable` -- Set this property to `true` to enable cursor measurements. Valid values are `true` or `false`.

  Default: `false`

All `CursorMeasurementsSpecification` properties are tunable.

UI Use

Open the **Cursor Measurements** pane () and modify the **Settings** options.

`DistortionMeasurements` — Distortion measurements

`DistortionMeasurementsSpecification` object

Enable distortion measurements to compute and display the harmonic distortion and intermodulation distortion. The `DistortionMeasurements` property uses the `DistortionMeasurementsSpecification` properties.

The `DistortionMeasurementsSpecification` properties are:

- `Algorithm` –– Type of measurement data to display, specified as either `"Harmonic"` or `"Intermodulation"`.

  Default: `"Harmonic"`
- `NumHarmonics` –– Number of harmonics to measure, specified as a real, positive integer. This property applies when you set the `Algorithm` to `"Harmonic"`.

  Default: `6`
- `Enable` –– Set this property to `true` to enable distortion measurements.

  Default: `false`

All `DistortionMeasurementsSpecification` properties are tunable.

UI Use

Open the **Distortion Measurements** pane () and modify the **Distortion** and **Harmonics** options.

Visualization

`Name` — Window name

`"Spectrum Analyzer"` (default) | character vector | string scalar

Title of the scope window.

Tunable: Yes

Data Types: `char` | `string`

`Position` — Window position

screen center (default) | `[left bottom width height]`

Spectrum Analyzer window position in pixels, specified by the size and location of the scope window as a four-element double vector of the form [left bottom width height]. You

can place the scope window in a specific position on your screen by modifying the values to this property.

By default, the window appears in the center of your screen with a width of `800` pixels and height of `450` pixels. The exact center coordinates depend on your screen resolution.

Tunable: Yes

`PlotType` — Plot type for normal traces

`"Line"` (default) | `"Stem"`

Specify the type of plot to use for displaying normal traces as either `"Line"` or `"Stem"`. Normal traces are traces that display free-running spectral estimates.

Tunable: Yes

UI Use

Open the **Style** properties and set **Plot type**.

Data Types: `char` | `string`

`ReducePlotRate` — Improve performance with reduced plot rate

`true` (default) | `false`

The simulation speed is faster when this property is set to `true`.

- `true` — the scope logs data for later use and updates the display at fixed intervals of time. Data occurring between these fixed intervals might not be plotted.
- `false` — the scope updates every time it computes the power spectrum. Use the `false` setting when you do not want to miss any spectral updates at the expense of slower simulation speed.

UI Use

Select **Simulation** > **Reduce plot rate to improve performance**.

Data Types: `logical`

`Title` — Display title

`''` (default) | character vector | string scalar

Specify the display title as a character vector or string.

Tunable: Yes

UI Use

Open the **Configuration Properties**. Set **Title**.

Data Types: `char` | `string`

`YLabel` — Y-axis label

`''` (default) | character vector | string scalar

Specify the text for the scope to display to the left of the *y*-axis.

Regardless of this property, Spectrum Analyzer always displays power units as one of the `SpectrumUnits` values.

Tunable: Yes

UI Use

Open the **Configuration Properties**. Set **Y-label**.

Data Types: `char` | `string`

`ShowLegend` — Show legend

`false` (default) | `true`

To show a legend with the input names, set this property to `true`.

From the legend, you can control which signals are visible. This control is equivalent to changing the visibility in the **Style** dialog box. In the scope legend, click a signal name to hide the signal in the scope. To show the signal, click the signal name again. To show only one signal, right-click the signal name. To show all signals, press **Esc**.

Note The legend only shows the first 20 signals. Any additional signals cannot be viewed or controlled from the legend.

Tunable: Yes

UI Use

Open the **Configuration Properties**. On the **Display** tab, select **Show legend**.

Data Types: `logical`

`ChannelNames` — Channel names

empty cell (default) | cell array of character vectors

Specify the input channel names as a cell array of character vectors. The names appear in the legend, **Style** dialog box, and **Measurements** panels. If you do not specify names, the channels are labeled as `Channel 1`, `Channel 2`, etc.

Tunable: Yes

Dependency

To see channel names, set `ShowLegend` to `true`.

UI Use

On the legend, double-click the channel name.

Data Types: `char`

`ShowGrid` — Grid visibility

`true` (default) | `false`

Set this property to `true` to show gridlines on the plot.

Tunable: Yes

UI Use

Open the **Configuration Properties**. On the **Display** tab, set **Show grid**.

Data Types: `logical`

`YLimits` — Y-axis limits

`[-80, 20]` (default) | `[ymin ymax]`

Specify the *y*-axis limits as a two-element numeric vector, `[ymin ymax]`.

Example: `scope.YLimits = [-10,20]`

Tunable: Yes

UI Use

Open the **Configuration Properties**. Set **Y-limits (maximum)** and **Y-limits (minimum)**.

`AxesScaling` — Axes scaling mode

`"Auto"` (default) | `"Manual"` | `"OnceAtStop"` | `"Updates"`

Specify when the scope automatically scales the axes. Valid values are:

- `"Auto"` — The scope scales the axes as-needed to fit the data, both during and after simulation.
- `"Manual"` — The scope does not scale the axes automatically.
- `"OnceAtStop"` — The scope scales the axes when the simulation stops.
- `"Updates"` — The scope scales the axes once after 10 updates.

UI Use

Select **Tools** > **Axes Scaling**.

Data Types: `char` | `string`

`AxesScalingNumUpdates` — Number of updates before scaling

`"10"` (default) | integer character vector | integer string scalar

Set this property to delay auto scaling the y-axis.

Dependency

To enable this property, set `AxesScaling` to `"Updates"`.

UI Use

Open the **Axes Scaling** dialog box and set **Number of updates**.

Data Types: `char` | `string`

`OpenAtSimulationStart` — Open scope when starting simulation

`true` (default) | `false`

Set this property to `true` to open the scope when the simulation starts. Set this property to `false` to prevent the scope from opening at the start of simulation.

UI Use

Select **File** > **Open at Start of Simulation**.

Data Types: `logical`

`Visible` — Visibility of the Spectrum Analyzer

`false` | `true`

Set this property to `true` to show the spectrum analyzer window, or `false` to hide the spectrum analyzer window.

Data Types: `logical`

Examples

Construct a Spectrum Analyzer Configuration Object

Create the configuration object for a Spectrum Analyzer block.

Create a new Simulink® model with a randomly-generated name.

```
sysname=char(randi(26,1,7)+96);
new_system(sysname);
```

Add a new Spectrum Analyzer block to the model.

```
add_block('built-in/SpectrumAnalyzer',[sysname,'/SpectrumAnalyzer'])
```

Call the `get_param` function to retrieve the default Spectrum Analyzer block configuration properties.

```
config = get_param([sysname,'/SpectrumAnalyzer'],'ScopeConfiguration')
```

```
config =

  SpectrumAnalyzerConfiguration with properties:

                 NumInputPorts: '1'
                  SpectrumType: 'Power'
               SampleRateSource: 'Inherited'
        PlotAsTwoSidedSpectrum: 1
                FrequencyScale: 'Linear'

   Advanced
                      RBWSource: 'Auto'
                 OverlapPercent: '0'
                         Window: 'Hann'
                   SpectrumUnits: 'dBm'
                AveragingMethod: 'Running'
               SpectralAverages: '1'
                  ReferenceLoad: '1'
                FrequencyOffset: '0'
    TreatMby1SignalsAsOneChannel: 1
```

```
Measurements
           MeasurementChannel: '1'
                   PeakFinder: [1×1 PeakFinderSpecification]
              CursorMeasurements: [1×1 CursorMeasurementsSpecification]
       DistortionMeasurements: [1×1 DistortionMeasurementsSpecification]

Visualization
                         Name: 'SpectrumAnalyzer'
                     Position: [560 375 800 450]
                     PlotType: 'Line'
                ReducePlotRate: 1
                        Title: ''
                       YLabel: ''
                   ShowLegend: 0
                 ChannelNames: {''}
                     ShowGrid: 1
                      YLimits: [-80 20]
                  AxesScaling: 'Auto'
        OpenAtSimulationStart: 1
                      Visible: 0
```

See Also

Blocks

Spectrum Analyzer

Topics

“Control Scope Blocks Programmatically” (Simulink)

Introduced in R2016b