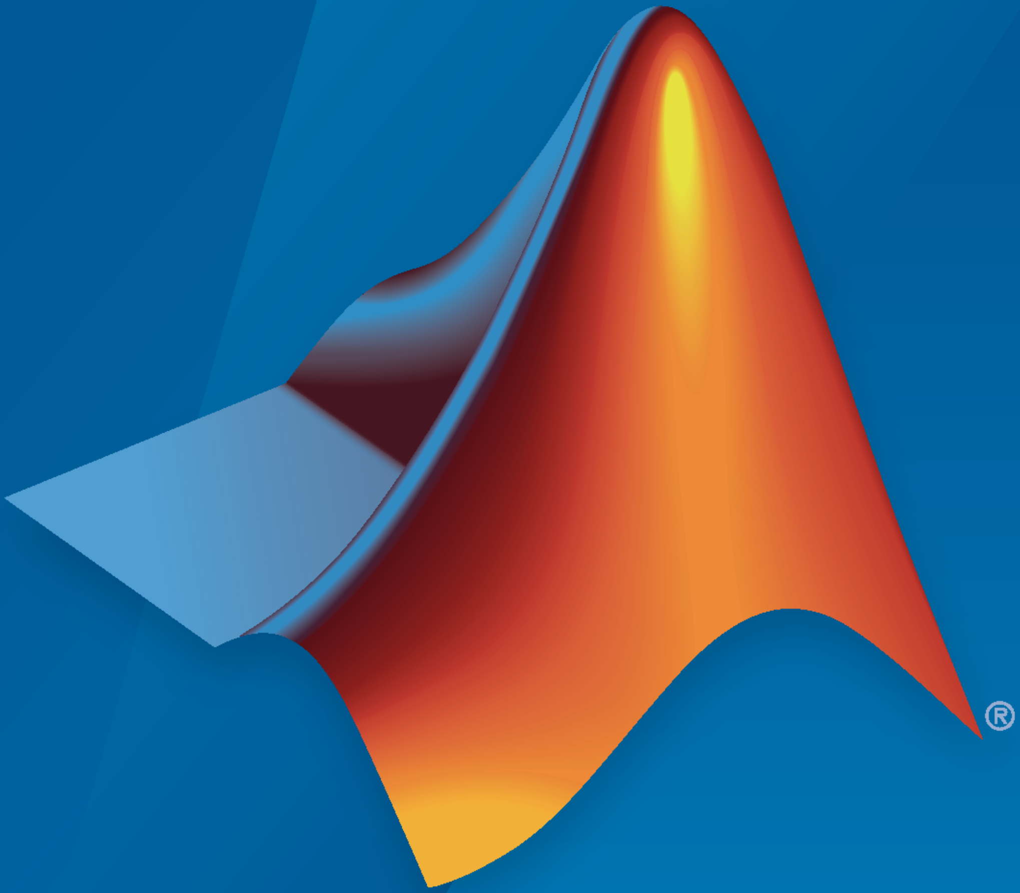
RF Blockset™

User's Guide



MATLAB® & SIMULINK®

R2020b

MathWorks®

RF Blockset™ User's Guide

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)
March 2020	Online only	Revised for Version 7.4 (Release 2020a)
September 2020	Online only	Revised for Version 8.0 (Release 2020b)

Contents

Circuit Envelope

4 Testbenches

RF Blockset Models

5 Analog Devices Transceiver Models

A RF Blockset Equivalent Baseband Algorithms

B Model Mixers

8 Examples

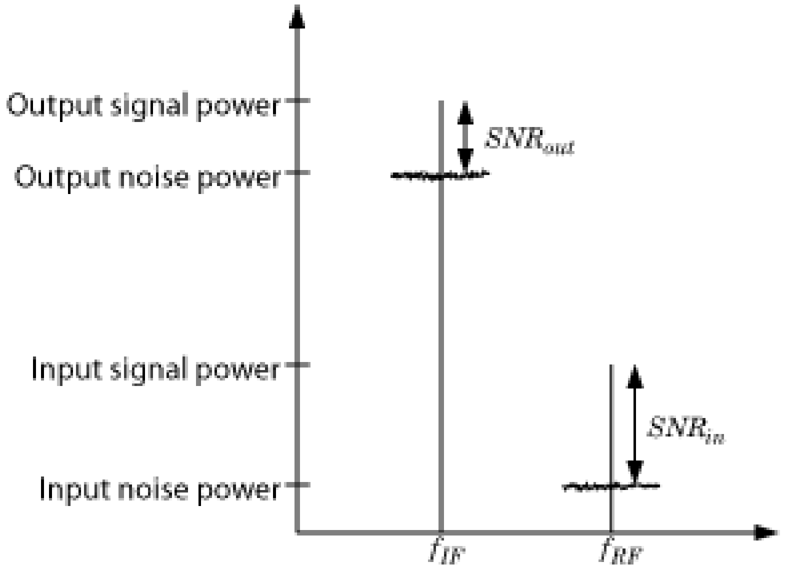
9 RF Blockset Examples

Circuit Envelope

Sensitivity

Model System Noise Figure

RF receivers amplify signals and shift them to lower frequencies. The receiver itself introduces noise that degrades the received signal. The signal-to-noise ratio (SNR) at the receiver output ultimately determines the usability of the receiver.



The preceding figure illustrates the effect of the receiver on the signal. The receiver amplifies a low-power RF signal at the carrier f_{RF} with a high SNR and downconverts the signal to f_{IF}. The noise figure (NF) of the system determines the difference between the SNR at the output and the SNR at the input:

$$SNR_{out} = SNR_{in} - NF_{sys}$$

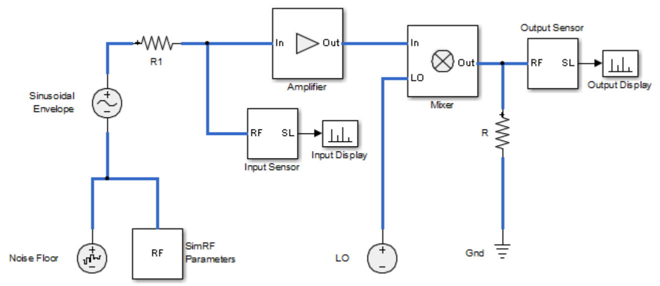
where the difference is calculated in decibels. Excessive noise figure in the system causes the noise to overwhelm the signal, making the signal unrecoverable.

Create a Low-IF Receiver Model

The model `ex_simrf_snr` simulates a simplified IF receiver architecture. A Sinusoid block and a Noise block model a two-tone input centered at f_{RF} and low-level thermal noise. The RF system amplifies the signal and mixes it with the local oscillator f_{LO} down to an intermediate frequency f_{IF}. A voltage sensor recovers the signal at the IF.

To open this model, at MATLAB® command line, enter:

```
addpath(fullfile(docroot,'toolbox','simrf','examples'))
ex_simrf_snr
```



The amplifier contributes 40 dB of gain and a 15-dB noise figure, and the mixer contributes 0 dB of gain and a 20-dB noise figure, which are values characteristic of a relatively noisy, high-gain receiver. The two-tone input has a specified level of .1 μV. A 1-V level in the local oscillator ensures consistency with the formulation of the conversion gain of the mixer.

To run the model:

1. Open the model by clicking the link or by typing the model name at the Command Window prompt.
2. Click **Run**.

Set Up the RF Blockset Environment

To maximize performance, the **Fundamental tones** and **Harmonic order** parameters specify the simulation frequencies explicitly in the Configuration block:

- f_{LO}, the frequency of the LO in the first mixing stage, equals 1.9999 GHz. and appears in the list of fundamental tones as `carriers.LO`.
- f_{RF}, the carrier of the desired signal, equals 2 GHz and appears in the list of fundamental tones as `carriers.RF`.
- f_{IF}, the intermediate frequency, equals $f_{RF} - f_{LO}$. The frequency is a linear combination of the first-order (fundamental) harmonics of f_{LO} and f_{RF}. Setting **Harmonic order** to `1` is sufficient to ensure this frequency appears in the simulation frequencies. This minimal value for the harmonic order ensures a minimum of simulation frequencies.

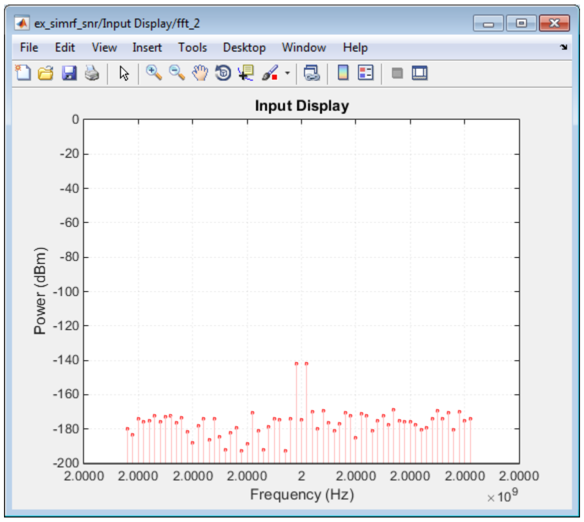
Solver conditions and noise settings are also specified for the Configuration block:

- The **Solver type** is set to `auto`. For more information on choosing solvers, see the reference page for the Configuration block or see Choosing Simulink and Simscape Solvers.
- The **Sample time** parameter is set to `1/(mod_freq*64)`. This setting ensures a simulation bandwidth 64 times greater than the envelope signals in the system.
- The **Simulate noise** box is checked, so the environment includes noise parameters during simulation.

View Simulation Output

The model uses subsystems with a MATLAB Coder™ implementation of a fast Fourier transform (FFT) to generate two plots. The FFT uses 64 bins, so for a sampling frequency of 64 Hz, the bandwidth of each bin is 1 Hz. Subsequently, the power levels shown in the figures also represent the power spectral density (PSD) of the signals in dBm/Hz.

- The Input Display plot shows the power spectrum of the signal and noise at the input of the receiver.



The measured power of each tone is consistent with the expected power level of a 0.1-µV two-tone envelope:

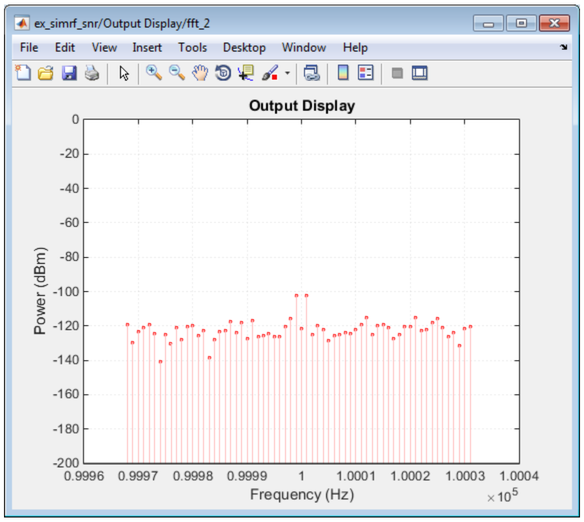
$$P_{in} = 10\log_{10}\left(\frac{V^2}{2R}\right) + 30$$

$$= 10\log_{10}\left(\frac{\left(\frac{1}{2}\cdot\frac{10^{-7}}{2}\right)^2}{2\cdot 50}\right) + 30 = -142\ \mathrm{dBm}$$

A factor of 1/2 is due to voltage division across source and load resistors, and another factor of 1/2 is due to envelope scaling. See the featured example Two-Tone Envelope Analysis Using Real Signals for more discussion on scaling envelope signals for power calculation.

The measured noise floor at -177 dBm/Hz is reduced by 3 dB from the specified -174 dBm/Hz noise floor. The difference is due to power transfer from the source to the input of the amplifier. The amplifier also models a thermal noise floor, so although this decrease is unrealistic, it does not affect accuracy at the output stage.

- The Output Display plot shows the power spectrum of the signal and noise at the output of the receiver.



The measured PSD of -102 dBm/Hz for each tone is consistent with the 40-dB combined gain of the amplifier and mixer. The noise PSD in the figure is shown to be approximately 50 dB higher at the output, due to the gain and noise figure of the system.

If you have DSP System Toolbox™ software installed, you can replace the MATLAB Coder subsystems with a Spectrum Analyzer block.

Simulating Thermal Noise Floor

Thermal noise power can be modeled according to the equation

$$P_{noise} = 4k_B T R_s \Delta f$$

where:

- k_B is Boltzmann's constant, equal to 1.38065×10^{-23} J/K.
- T is the noise temperature, specified as 293.15 K in this example.
- R_s is the noise source impedance, specified as 50 Ω in this example to agree with the resistance value of the Resistor block labeled `R1`.
- Δf is the noise bandwidth.

To model the noise floor on the RF signal at the resistor, the model includes a Noise block:

- The **Noise Power Spectral Density (Watts/Hz)** parameter is calculated as $P_{noise}/\Delta f = 4k_B T R_S$.
- The **Carrier frequencies** parameter, set to `carriers.RF`, places noise on the RF carrier only.

Computing System Noise Figure

To model RF noise from component noise figures:

1. Select **Simulate noise** in the RF Blockset Parameters block dialog box, if it is not already selected.
2. Specify a value for the **Noise figure (dB)** parameter of an Amplifier and Mixer blocks.

The noise figures are not strictly additive. The amplifier contributes more noise to the system than the mixer because it appears first in the cascade. To calculate the total noise figure of the RF system with *n* stages, use the Friis equation:

$$F_{sys} = F_1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 G_2} + \ldots + \frac{F_n - 1}{G_1 G_2 \ldots G_{n-1}}$$

where F_i and G_i are the noise factor and gain of the *i*th stage, and $NF_i = 10\log_{10}(F_i)$.

In this example, the noise figure of the amplifier is 10 dB, and the noise figure of the mixer is 15 dB, so the noise figure of the system is:

$$10\log_{10}\left(10^{10/10} + \frac{10^{15/10} - 1}{10000}\right) = 10.0\ \mathrm{dB}$$

The Friis equation shows that although the mixer has a higher noise figure, the amplifier contributes more noise to the system.

See Also

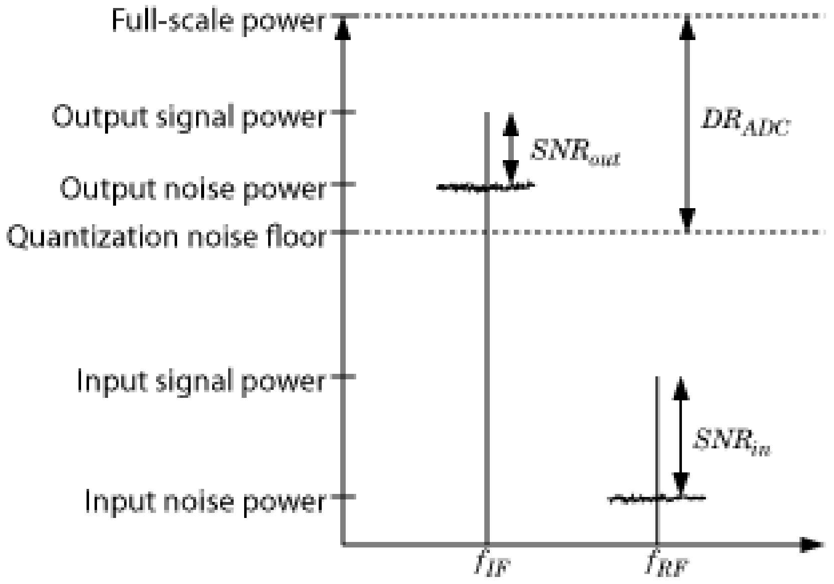
Amplifier | Mixer | Noise

More About

- “Noise in RF Systems” on page 3-8

Designing a Receiver with an ADC

Most RF receivers in modern communications or radar systems feed signals to an analog-to-digital converter (ADC). Due to their finite resolution, ADCs introduce quantization error into the system. The resolution of the ADC is determined by the number of bits and the full-scale (FS) range of the ADC.



The preceding figure illustrates an RF signal that falls within the dynamic range (DR) of an ADC. The input signal and noise at the carrier f_{RF} has high signal-to-noise ratio (SNR). The received signal at f_{IF} has reduced SNR due to system noise figure. However, if the quantization error is near or above the receiver noise, system performance degrades.

To ensure that the ADC contributes no more than 0.1 dB of noise to the signal at f_{IF}, the quantization noise floor must be 16 dB lower than the receiver noise. This condition can be met by:

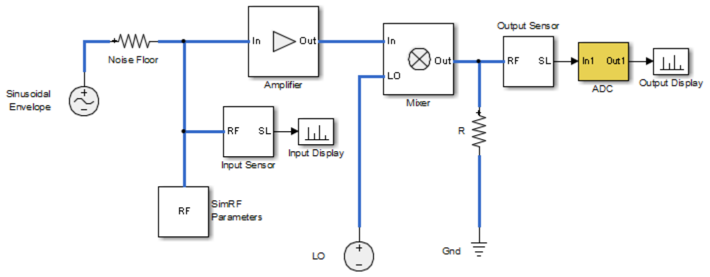
- Reducing the full-scale (FS) range or increasing the resolution of the ADC, which lowers the quantization noise floor.
- Increasing the gain of the RF receiver, which raises the receiver noise floor.

Overcome Quantization Error of an ADC

The model `ex_simrf_adc` simulates a low-IF receiver with an ADC. This model is based on the model `ex_simrf_snr` described in the section “Create a Low-IF Receiver Model” on page 1-2. At the output of the RF system, the ADC subsystem models an ADC with an FS range of `sqrt(100e-3)` V and a resolution of 16 bits.

To open this model, at MATLAB command line, enter:

```
addpath(fullfile(docroot,'toolbox','simrf','examples'))
ex_simrf_adc
```



The power of a voltage signal at the full-scale range of the ADC is

$$10\log_{10}\sqrt{100}^{-3} + 30 = 0\,\text{dBm}$$

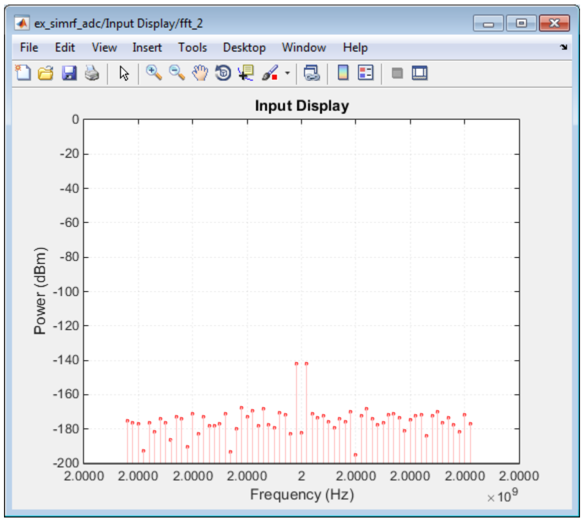
To maximize performance, the model uses the same simulation settings as `ex_simrf_snr`. To run this model:

1. Open the model by clicking the link or by typing the model name at the Command Window prompt.
2. Click **Run**.

View Simulation Output

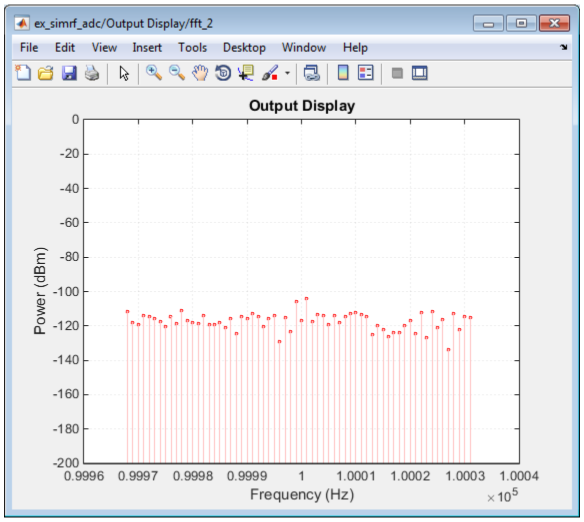
The model uses subsystems with a MATLAB Coder implementation of a fast Fourier transform (FFT) to generate two plots. The FFT uses 64 bins, so for a sampling frequency of 64 Hz, the bandwidth of each bin is 1 Hz. Subsequently, the power levels shown in the figures also represent the power spectral density (PSD) of the signals in dBm/Hz.

- The Input Display plot shows the power spectrum of the two-tone signal and noise at the input of the receiver-ADC system.



The measured power of each tone of -142 dBm is consistent with the expected power level of a .1-µV signal. The power level of the noise is consistent with a -174 dBm/Hz noise floor.

- The Output Display plot shows power spectrum of the output signal.



The quantization error exceeds the receiver noise.

If you have DSP System Toolbox software installed, you can replace the MATLAB Coder subsystems with a Spectrum Analyzer block.

Measuring the Quantization Noise Floor

To calculate the quantization noise floor (QNF) of the ADC, subtract the dynamic range from the full-scale power, which is 0 dBm. To calculate the dynamic range PSD for the ADC, use the equation:

$$DR_{ADC} = 6.02 \cdot N_{bits} + 10\log_{10}(\Delta f) + 1.76 = 116.1 \text{ dBm/Hz}$$

where

- N_{bits} is the resolution. The ADC in this example uses 16 bits.
- Δf is the bandwidth of the FFT, which is 64 in this example. Oversampling in an ADC yields lower quantization noise.
- The value 1.76 is a correction factor for a pure sinusoidal input.

Therefore, the quantization noise floor is -116 dBm/Hz, in agreement with the measured output levels.

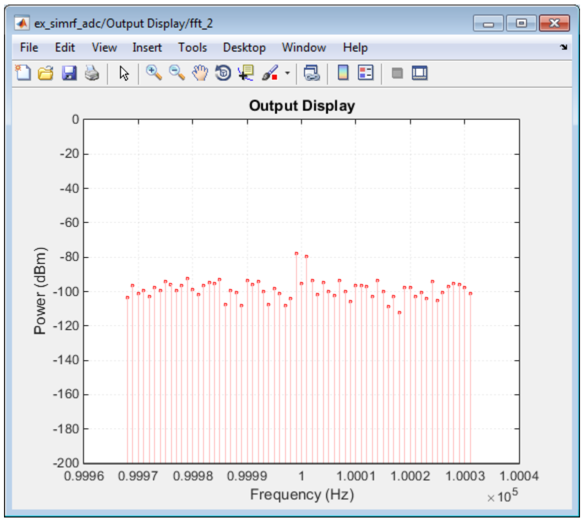
Improving Receiver-ADC Performance

Increasing the gain in the mixer raises the receiver noise without increasing the noise figure. Calculate the mixer gain required to achieve a 16-dB margin between the quantization noise floor and the receiver noise:

$$\begin{aligned} G_{mixer} &= (QNF_{ADC} + 16) - (-174 + G_{sys} + NF_{sys}) \\ &= (-116.1 + 16) - (-174 + 40 + 10.0) \\ &= -100.1 + 124.0 \\ &= 23.9 \text{ dB} \end{aligned}$$

To simulate a receiver that clears the quantization noise floor:

1 Set the **Available power gain** parameter of the mixer to `23.9`.
2 Click **Run**.



The figure shows that the receiver noise is 16 dB above the quantization noise floor.

See Also

Amplifier | Mixer

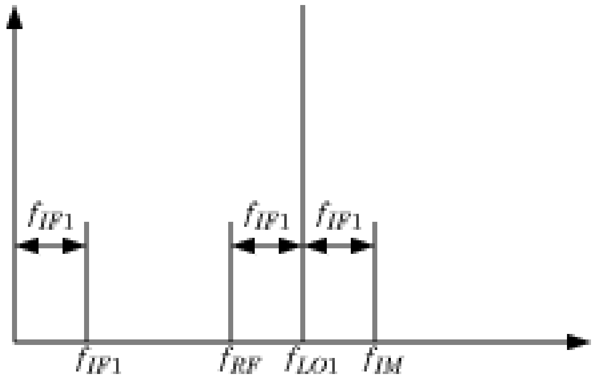
More About

- “RF Noise Modeling” on page 9-199
- “Noise in RF Systems” on page 3-8

Interference

Carrier to Interference Performance of Weaver Receiver

A classic superheterodyne architecture filters images prior to frequency conversion. In contrast, image-reject receivers remove the images at the output without filtering but are sensitive to phase offsets.

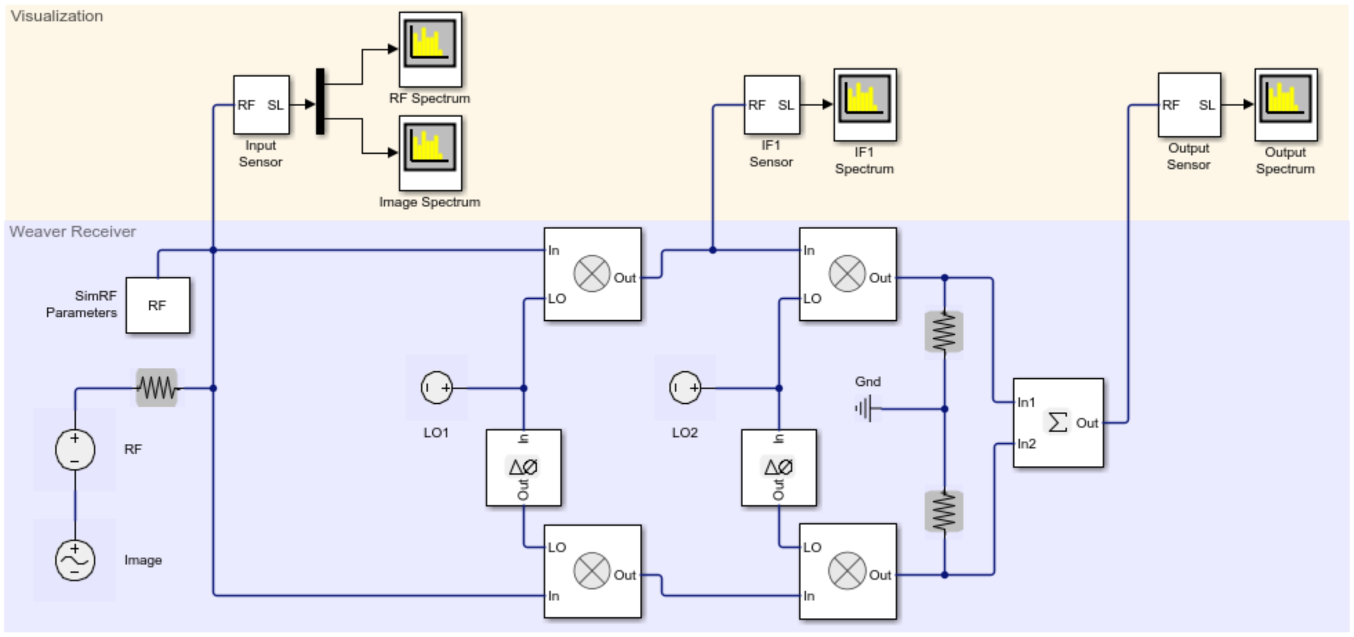


The preceding figure illustrates two input signals at the carriers f_{RF} and f_{IM} that both differ from the LO frequency, f_{LO1} , by an amount f_{IF1}. Mixing translates both input signals down to f_{IF1}. Perfect image rejection in the final stage of the receiver removes the image signal from the output entirely.

Create Model with RF Interference

The model `ex_simrf_ir` performs image rejection with a Weaver architecture. The receiver downconverts the signals f_{RF} and f_{IM} to f_{IF1} and f_{IF2} in two sequential stages.

```
open_system('ex_simrf_ir')
```



Set Up RF Blockset Environment

To maximize performance, the **Fundamental tones** and **Harmonic order** parameters are explicitly set in the `Configuration` block. To create the minimal set of simulation frequencies, the following carrier frequencies are set or created within the model.

- f_{RF}, the RF carrier, equals 100 MHz.
- f_{IM}, the image of the RF carrier relative to f_{LO1}, equals 200 MHz.
- f_{LO1}, the frequency of the LO in the first mixing stage, equals 150 MHz.
- f_{IF1}, the intermediate frequency of the signal after the first mixing stage equals: $f_{LO1} - f_{RF} = f_{IM} - f_{LO1}$ = 50 MHz.
- f_{LO2}, the frequency of the LO second mixing stage equals 75 MHz.
- f_{IF2}, the intermediate frequency of the signal after the second mixing stage equals: $f_{LO2} - f_{IF1}$ = 25 MHz.

In this system, every carrier is a multiple of f_{IF2}. The largest carrier, f_{IM}, is the 8th harmonic of f_{IF2}, so setting **Fundamental tones** to f_{IF2} and the **Harmonic order** to 8 ensures that every carrier is in the set of simulation frequencies.

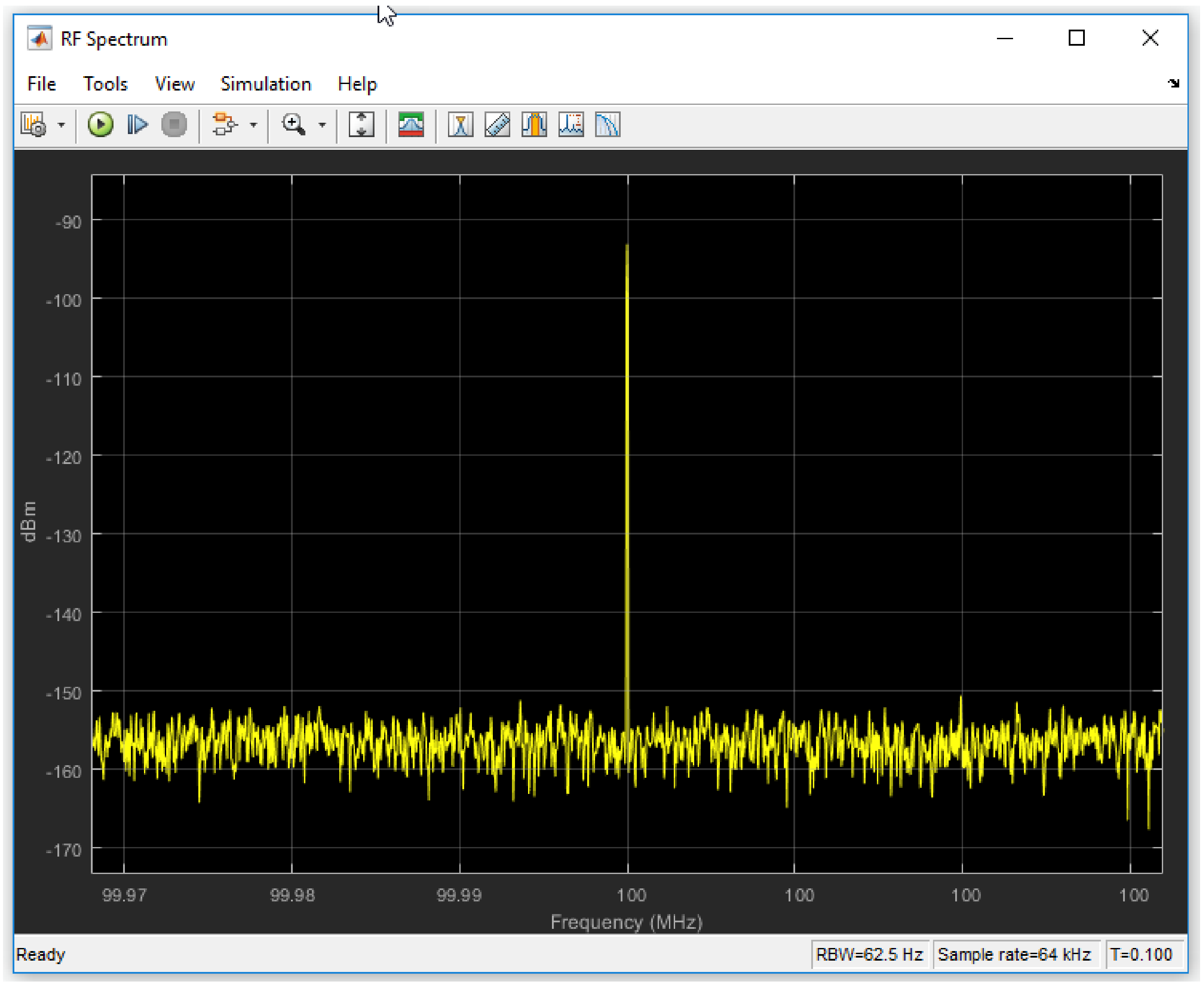
Solver conditions and noise settings are also specified for the `Configuration` block:

- The **Solver type** is set to auto. For more information on choosing solvers, see the reference page for the Configuration block or see Choosing Simulink and Simscape Solvers.
- The **Step size** parameter is set to 1/(mod_freq*64). This setting ensures a simulation bandwidth 64 times greater than the envelope signals in the system.
- The **Simulate noise** box is checked, so the environment includes noise in the simulation.

View Simulation Output

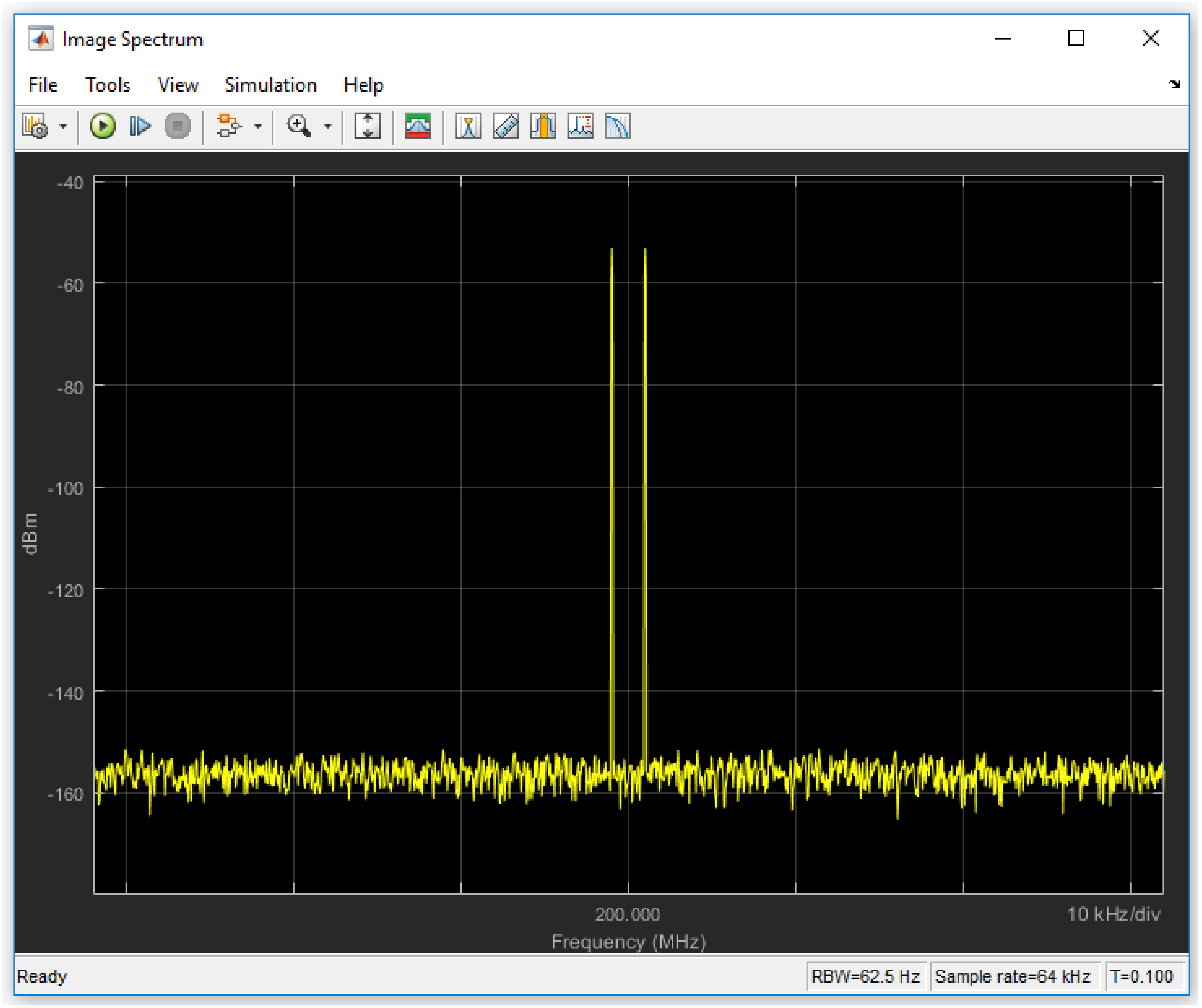
The model uses `Spectrum Analyzer` to generate four plots.

The RF spectrum scope shows the power spectrum of the carrier signal f_{RF}, specified as `carriers.RF` in the **Carrier frequencies** parameter of the Input Sensor `Outport` block.



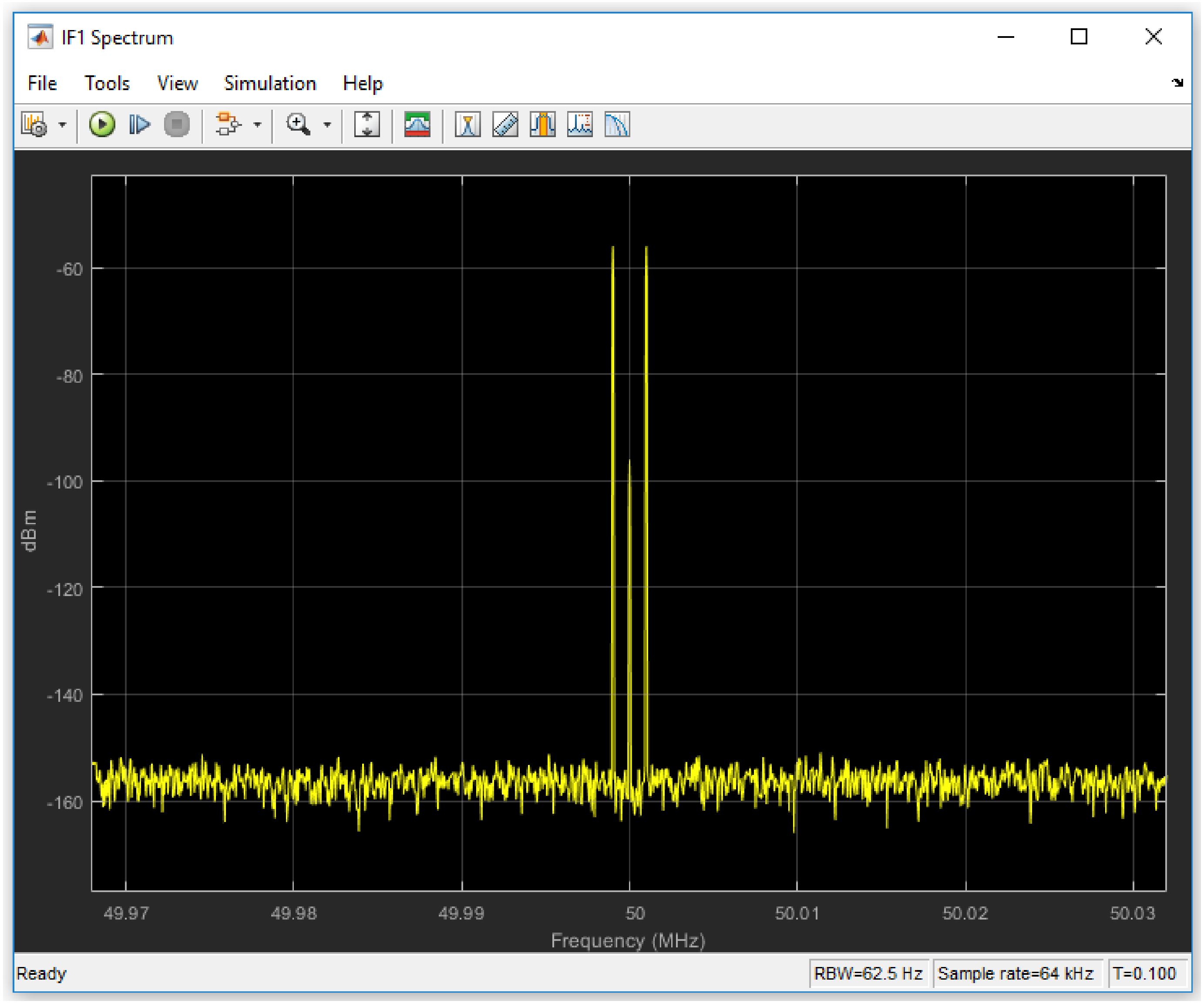
The modulation of the RF carrier is a constant envelope generated by a `Continuous Wave` block which generates a single peak centered at the carrier.

The Image Spectrum scope shows the power spectrum of the image. The signal is recovered from the carrier f_{IM}, specified as `carriers.IM` in the **Carrier frequencies** parameter of the Input Sensor `Outport` block.

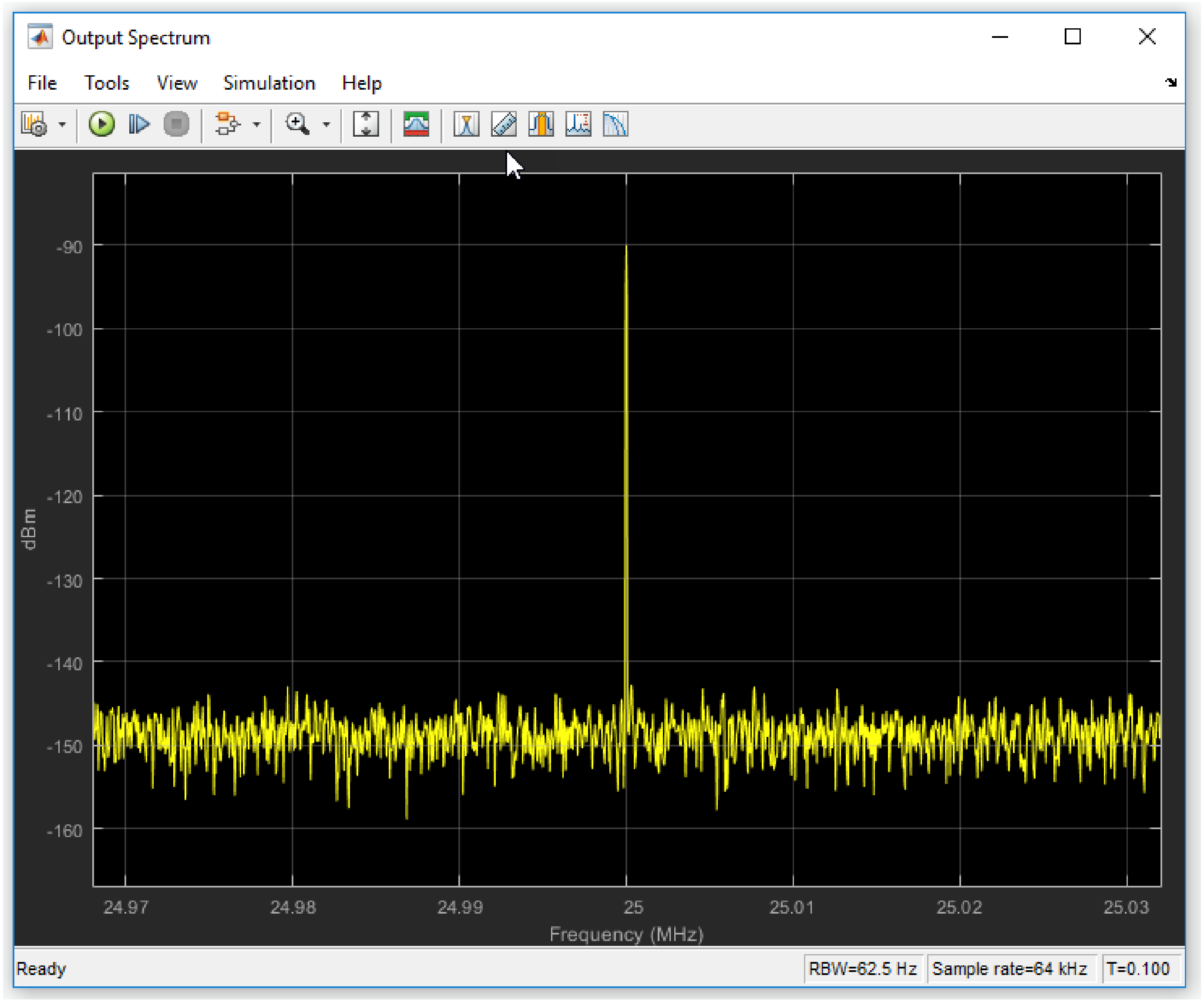


The Image `Sinusoidal Source` block generates a two-tone signal centered at f_{IM}.

The IF1 spectrum scope plot shows a power spectrum centered at the first intermediate frequency, measured between the first and second stages. The sensor outputs the modulation from the carrier f_{IF1}, specified as `carriers.IF1` in the **Carrier frequencies** parameter.



The Output spectrum scope shows the complete effects of the RF system. The sensor outputs the modulation from the carrier f_{IF2}, specified as `carriers.IF2` in the **Carrier frequencies** parameter.



Model RF and Blocker Sources

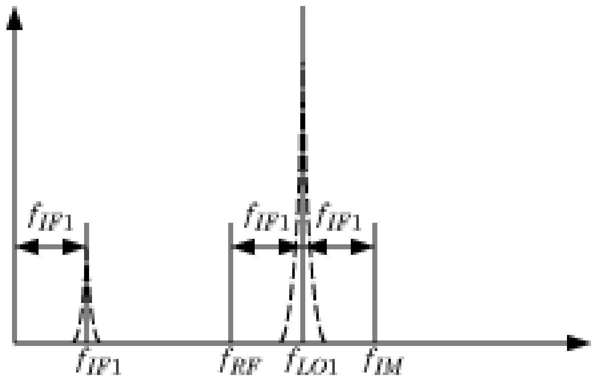
To model more robust input signals, you can use a RF Blockset `Inport` block to specify a circuit envelope signal generated using blocks from other Simulink™ libraries. For example, see the featured example `Impact of an RF Receiver on Communication System Performance`. This example uses Communications System Toolbox™ to model a QPSK-modulated waveform of random bits with RF Blockset Inport that brings the signal into the RF Blockset environment.

Simulating LO Phase Offset

The phase shifters have specified Phase shift parameters of 90. Deviation from this value results in a phase offset and causes imperfect image rejection. The featured example `Measuring Image Rejection Ratio in Receivers` analyzes the IRR of the Weaver and Hartley architectures several times, calculating the image rejection ratio (IRR) for several different phase offsets.

Model LO Phase Noise

A mixer transfers local oscillator (LO) phase noise directly to its output.

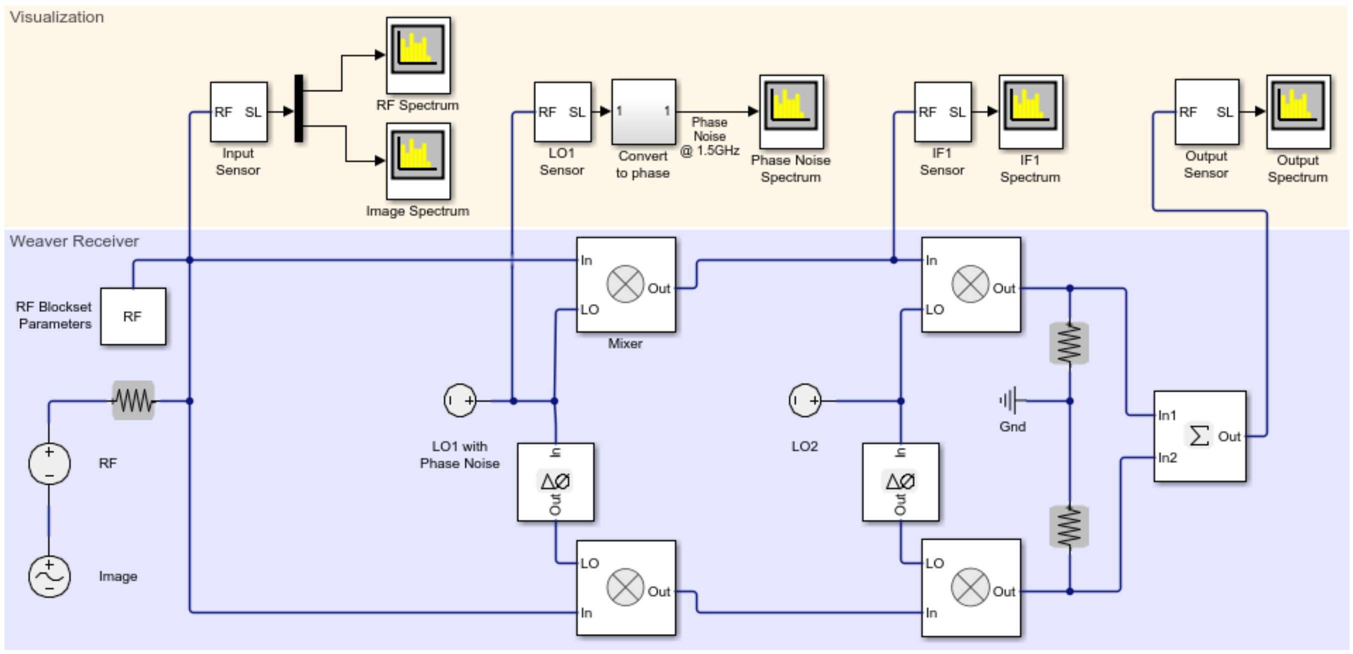


The preceding figure shows the transfer of phase noise from f_{LO1} to f_{IF1} .

Create Model with Phase Noise

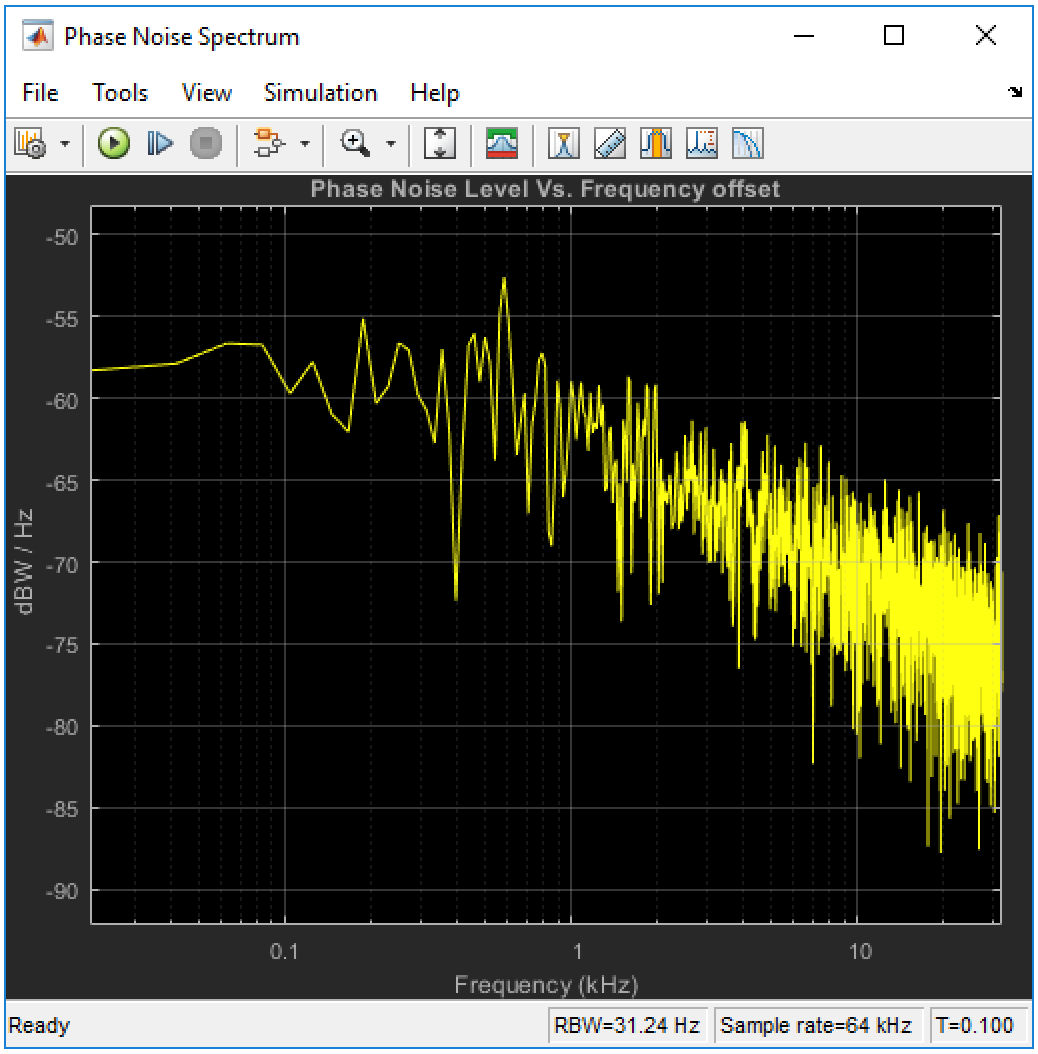
The model `ex_simrf_phase_noise` introduces phase noise into the model from the section `Create a Model with RF Interference`. The first mixing stage downconverts the RF and image to f_{IF}. To open the model:

```
open_system('ex_simrf_phase_noise')
```



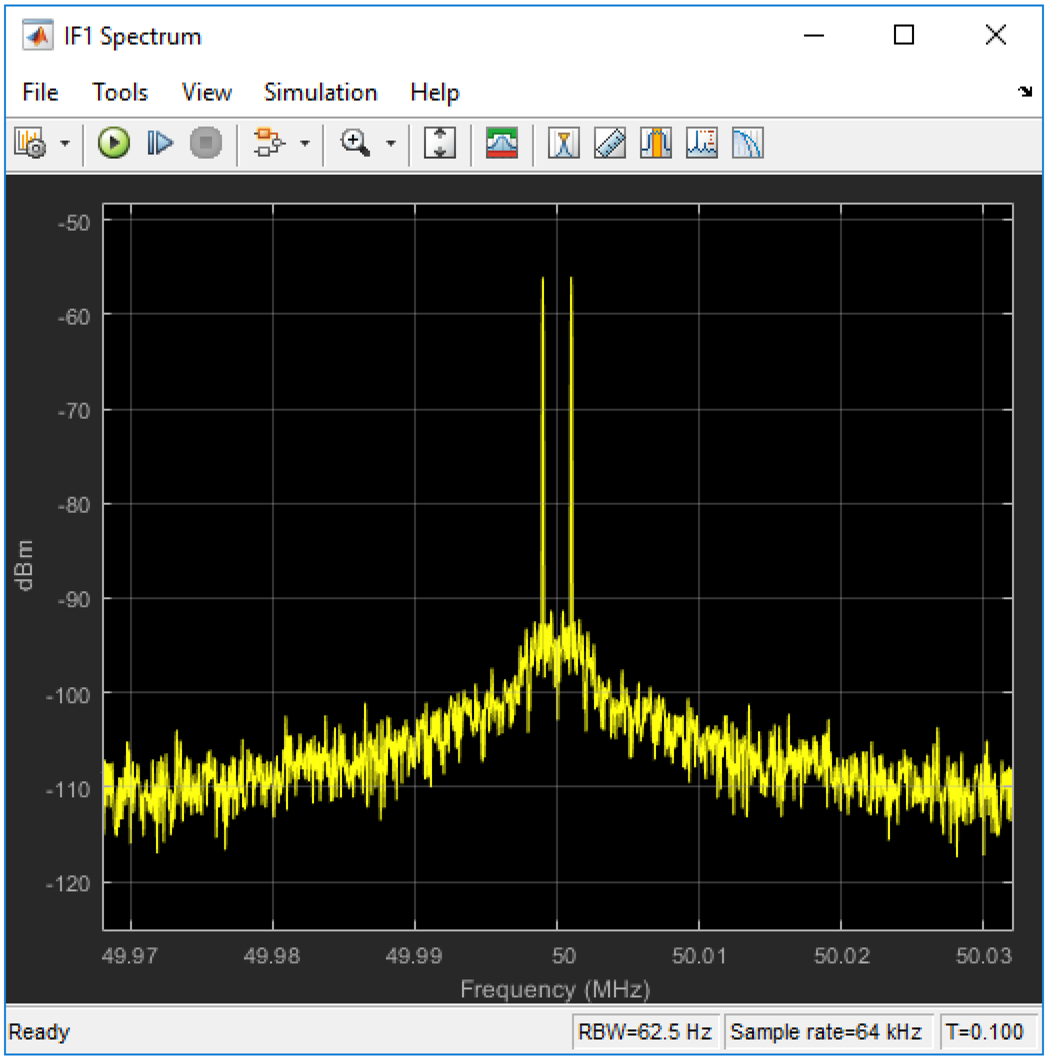
View Simulation Output

The model uses `Spectrum Analyzer` to generate 5 plots.



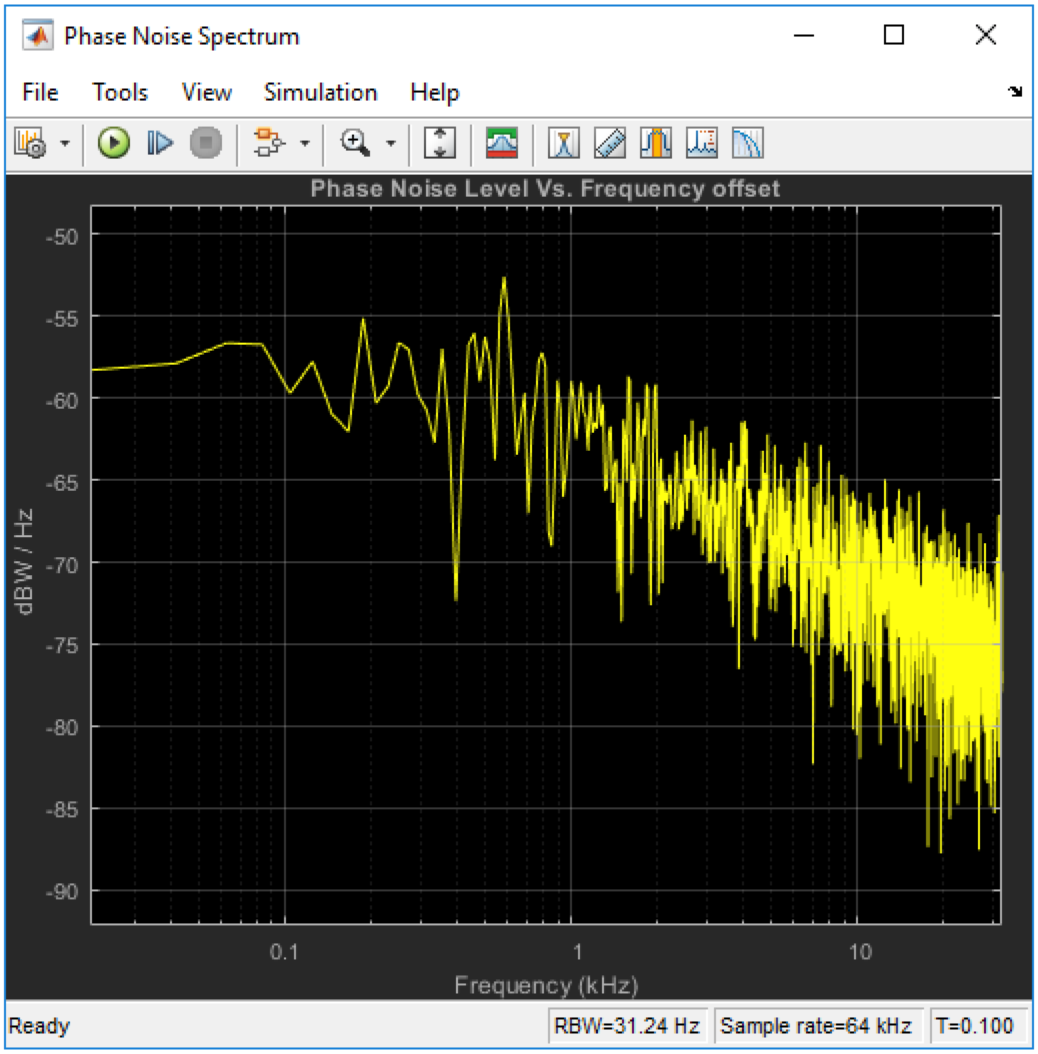
The Phase Noise spectrum scope shows a single-sided power density spectrum measuring the phase noise level at the LO1 source versus frequency offset shown in logarithmic scale.

The IF1 spectrum scope shows a power spectrum centered at the first intermediate frequency, measured between the first and second stages.



The scope shows that the LO phase noise has been transferred to the image. The RF signal on the carrier f_{IF1} is not visible in the figure because its power level is below the phase noise power of the downconverted image signal.

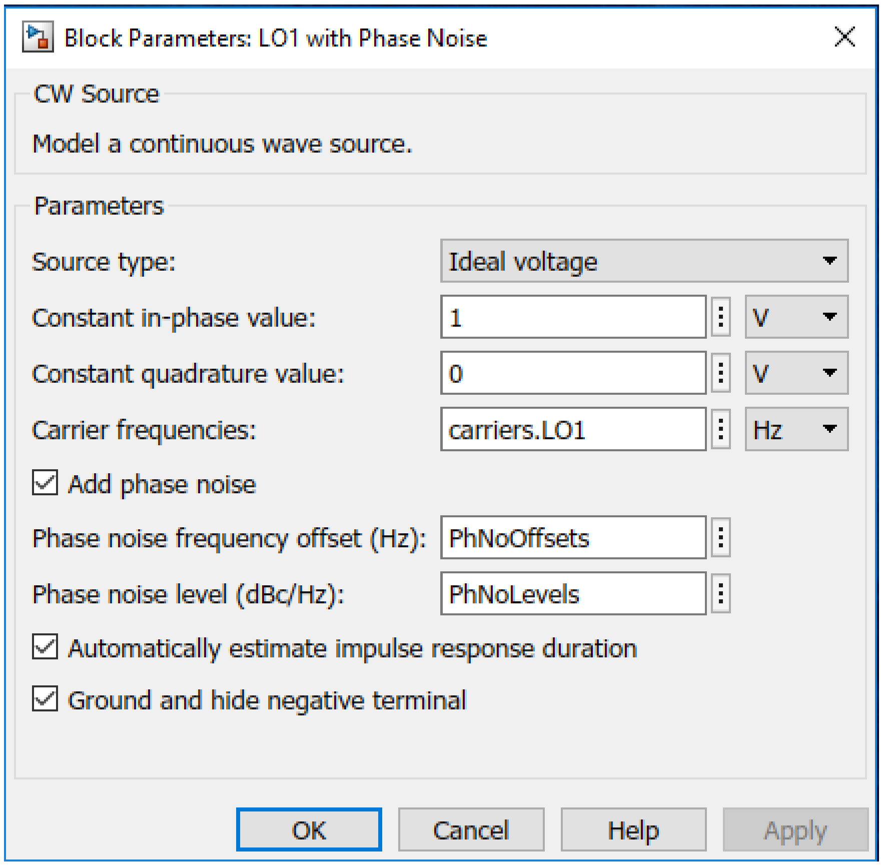
The Output spectrum scope shows the downconverted RF with the images removed.



The LO phase noise has been transferred to the receiver output.

Shaping LO Noise Skirt

As seen in the phase noise scope, the added phase noise is pink (1/f) and is specified within the CW source LO1. Specifically, the Add phase noise checkbox is checked in the blocks parameters dialog:


Block Parameters: LO1 with Phase Noise

CW Source

Model a continuous wave source.

Parameters

Source type: Ideal voltage

Constant in-phase value: 1 V

Constant quadrature value: 0 V

Carrier frequencies: carriers.LO1 Hz

☑ Add phase noise

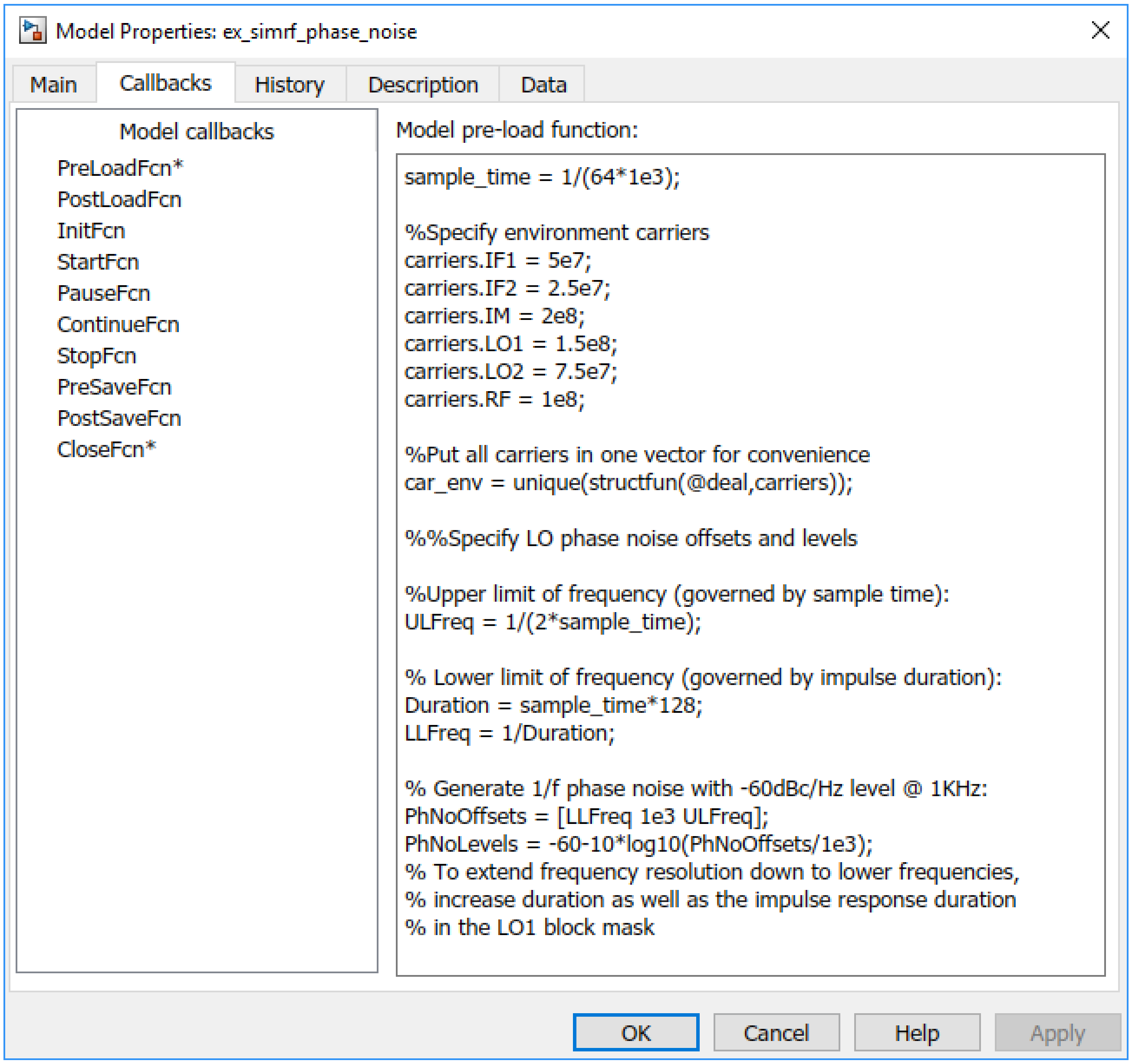
Phase noise frequency offset (Hz): PhNoOffsets

Phase noise level (dBc/Hz): PhNoLevels

☑ Automatically estimate impulse response duration

☑ Ground and hide negative terminal

The phase noise frequency offset and phase noise level variables PhNoOffsets and PhNoLevels are defined in the models PreLoadFcn callback, accessible through File > Model Properties > Model Properties:



```
sample_time = 1/(64*1e3);

%Specify environment carriers
carriers.IF1 = 5e7;
carriers.IF2 = 2.5e7;
carriers.IM = 2e8;
carriers.LO1 = 1.5e8;
carriers.LO2 = 7.5e7;
carriers.RF = 1e8;

%Put all carriers in one vector for convenience
car_env = unique(structfun(@deal,carriers));

%%Specify LO phase noise offsets and levels

%Upper limit of frequency (governed by sample time):
ULFreq = 1/(2*sample_time);

% Lower limit of frequency (governed by impulse duration):
Duration = sample_time*128;
LLFreq = 1/Duration;

% Generate 1/f phase noise with -60dBc/Hz level @ 1KHz:
PhNoOffsets = [LLFreq 1e3 ULFreq];
PhNoLevels = -60-10*log10(PhNoOffsets/1e3);
% To extend frequency resolution down to lower frequencies,
% increase duration as well as the impulse response duration
% in the LO1 block mask
```

The upper limit of the offset frequency is governed by the sample time and is limited to the envelope bandwidth of the simulation. The lower limit of the offset frequency is governed by the duration of the impulse response generating the phase noise frequency profile. Increasing the duration length in time steps from 128 to 256 will double the frequency resolution and allow simulation of the 1/f profile down to 250Hz (as opposed to the current lower limit of 500Hz).

See Also

Noise Figure Testbench | Mixer

Related Topics

“RF Noise Modeling” on page 9-199

Intermodulation Distortion

Model a Direct Conversion Receiver

In this section...

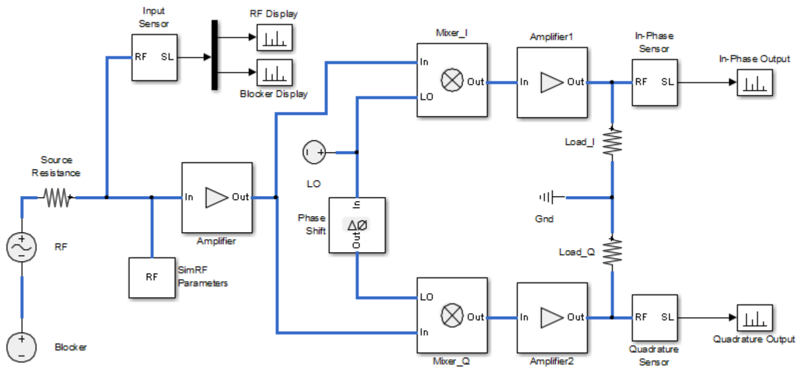
Direct-conversion receivers are sensitive to second-order intermodulation products because they transfer the RF signal directly to baseband.

Create a Direct Conversion Receiver Model

The model `ex_simrf_dc` models a direct-conversion receiver within the RF Blockset environment. The RF system consists of a low-noise amplification (LNA) stage, a direct-conversion stage, and a final amplification stage.

To open this model, at MATLAB command line, enter:

```
addpath(fullfile(docroot,'toolbox','simrf','examples'))
ex_simrf_dc
```



To run the model:

1. Open the model by clicking the link or by typing the model name at the command prompt.
2. Click **Run**.

Set Up the RF Blockset Environment

The model runs according to the following environment settings:

- In the Configuration dialog box, the **Fundamental tones** parameter specifies the carriers in the RF Blockset environment:

  - $f_{RF} = f_{LO}$, the carrier of the RF and the local oscillator.
  - f_{BL}, the blocker carrier

  The RF Blockset environment always simulates the 0- Hz carrier, regardless of whether the RF Blockset Parameters block specifies it.

- In the Solver Configuration dialog box, the **Use local solver** box is selected. This setting causes the RF Blockset environment to simulate with a local solver with the following settings:
  - **Solver type** is `Trapezoidal rule`.
  - **Sample time** is `sample_time`, defined as `1.25e-4` in the model initialization function.

  Since the model uses a local solver, the global solver settings do not affect the simulation within the RF Blockset environment. For more information on global and local solvers, see Choosing Simulink and Simscape Solvers.

To maximize performance, the **Fundamental tones** and **Harmonic order** parameters specify the simulation frequencies explicitly in the Configuration block:

- $f_{RF} = f_{LO}$, the carrier of the RF and the local oscillator, appears as a fundamental tone.
- f_{BL}, the blocker carrier, appears as a fundamental tone.
- A carrier of 0 Hz, representing the passband signal, is included in the set of first-order harmonics of both fundamental tones. Therefore, setting **Harmonic order** to `1` is sufficient to ensure that this frequency appears in the simulation frequencies. This minimal value for the harmonic order ensures a minimum of simulation frequencies.

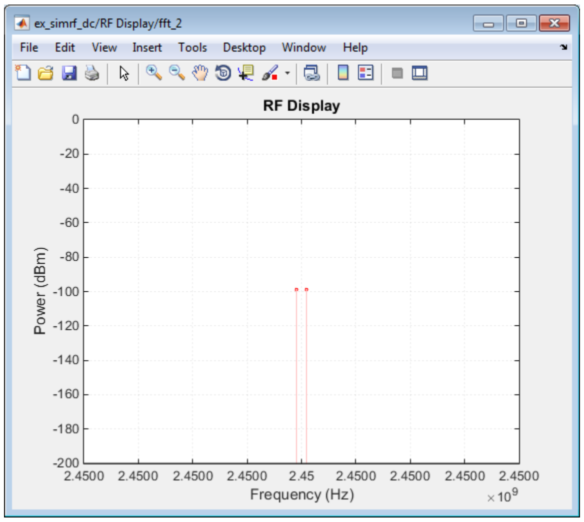
Solver conditions and noise settings are also specified for the Configuration block:

- The **Solver type** is set to `auto`. For more information on choosing solvers, see the reference page for the Configuration block or see Choosing Simulink and Simscape Solvers.
- The **Sample time** parameter is set to `sample_time`, which is equal to `1/(mod_freq*64)`. This setting ensures a simulation bandwidth 64 times greater than the envelope signals in the system.
- The **Simulate noise** box is checked, so the environment includes noise parameters during simulation.

View Simulation Output

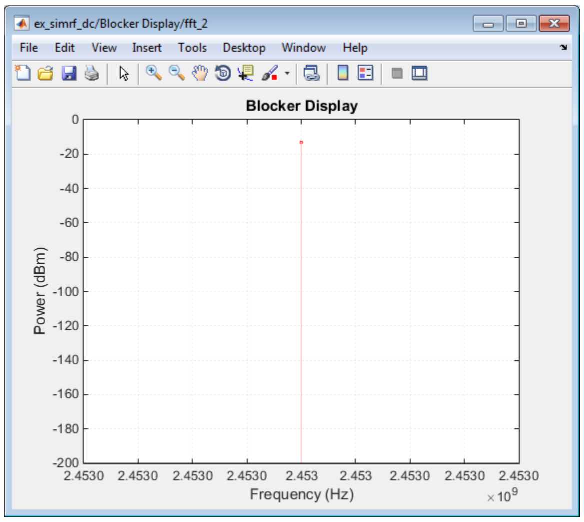
The model uses subsystems with a MATLAB Coder implementation of a fast Fourier transform (FFT) to generate four plots:

- The RF Display plot shows the power level of the RF signal.



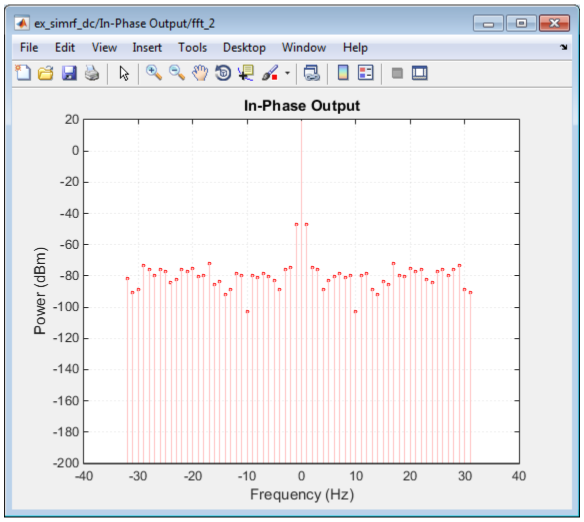
The power level of the RF is about 100 dBm.

- The Blocker Display plot shows the power spectrum centered at the carrier f_{BL}.



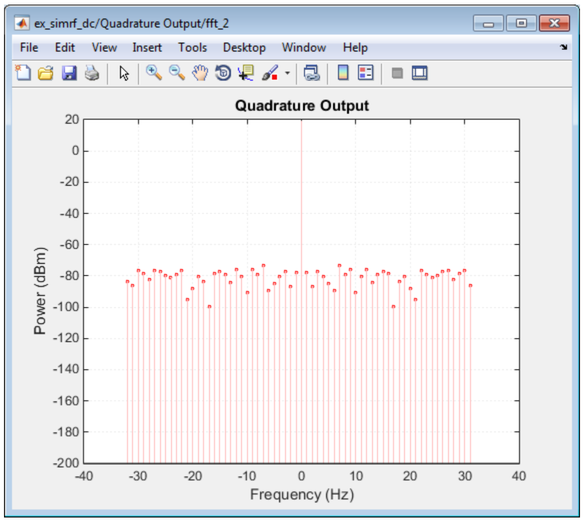
The power level of the blocker is about 90 dB higher than the signal power of the RF.

- The In-Phase Output plot shows the power spectrum of the in-phase signal at baseband.



In the figure, DC power is a direct result of the blocker and the IP2 in the mixers.

- The Quadrature Output plot shows the power spectrum of the quadrature signal at baseband.



If you have DSP System Toolbox software installed, you can replace the MATLAB Coder subsystems with a Spectrum Analyzer block.

Modeling IMD in System-Level Components

The **IP2** and **IP3** parameters specify the second- and third-order intercept points of Amplifier and Mixer blocks:

- The amplifiers have infinite **IP2** and **IP3**, so the amplifiers are linear.
- **IP2** of the mixer is 15 dB

Amplifier and Mixer components have specified gains and noise figures:

- The gain and noise figure in the LNA stage are 25 dB and 6 dB, respectively.
- The gain and noise figure in the mixing stage are 10 dB and 10 dB. The **Input impedance (ohms)** parameters of the two mixers are both `100`, which sum in parallel to a resistance of 50 Ω to match the output impedance of the LNA.
- The gain and noise figure in the final amplification stage are 20 dB and 15 dB, respectively.

To calculate RF system noise figure, use the Friis equation:

$$F_{sys} = F_1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 G_2} + \ldots + \frac{F_n - 1}{G_1 G_2 \ldots G_{n-1}}$$

where F_i and G_i are the noise factor and gain of the ith stage.

Examining DC Impairments

In addition to intermodulation distortion from IP2, direct-conversion receivers are subject to additional DC impairments. For example, coupling between mixer input and local oscillator (LO) ports causes self-mixing of the LO. For more information, see the featured example “Executable Specification of a Direct Conversion Receiver” on page 9-72

See Also

Related Examples

- “Top-Down Design of an RF Receiver” on page 9-179

Intermodulation Distortion in Amplifiers and Mixers

Intermodulation distortion or IMD is a measure of linearity of amplifiers, mixers, gain blocks, and other RF components. IMD is the result of two or more signals interacting in a non-linear device such as an amplifier or a mixer to produce additional unwanted signals.

Intermodulation Distortion in Amplifiers

Intermodulation distortion or IMD is an undesirable quality in power amplifiers.

Noise in RF Systems

In this section...

Noise in an RF system is generated internally by active components in the system or introduced externally like channel interference or antenna.

White and Colored Noise

White noise: Noise with a flat frequency spectrum is called `white noise`. White noise has equal power across all frequencies of the system band width.

Colored noise: Noise with power that varies according to frequencies in an RF system bandwidth is called `colored noise`.

To simulate white or colored noise in RF Blockset, use the Noise block.

Thermal Noise

Thermal noise is the most common noise introduced in an RF system. This noise is generated internally by active components in the system or externally due to channel interference or antenna. Thermal noise is also known as Johnson or Nyquist noise. The equation for thermal noise is:

$$P_N = k_B T B$$

- k_B is Boltzmann's constant, equal to 1.38065×10^{-23} J/K.
- T is the noise temperature, specified as 293.15 K in this example.
- R_s is the noise source impedance, specified as 50 Ω in this example to agree with the resistance value of the Resistor block labeled `R1`.
- B is the bandwidth.

At room temperature, the thermal noise generated by system with a band width of 1 Hz is `-174 dBm`.

To generate thermal noise in a RF Blockset system use Resistor and Configuration block. You can also generate thermal noise using the S-Parameters block if the S-parameter is passive.

Thermal noise floor in RF Blockset is defined by the equation:

$$P_{noise} = 4 k_B T R_s \Delta f$$

where:

- k_B is Boltzmann's constant, equal to 1.38065×10^{-23} J/K.
- T is the noise temperature, specified as 293.15 K in this example.
- R_s is the noise source impedance, specified as 50 Ω in this example to agree with the resistance value of the Resistor block labeled `R1`.

- Δf is the noise bandwidth.

Phase Noise

Phase noise is a short-term fluctuation in the phase of an oscillator signal. This noise introduces uncertainty in the detection of digitally modulated signals. Phase noise is defined as the ratio of power ins one-phase modulation sideband to the total signal power per unit bandwidth. It is expressed in decibels relative to the carrier power per hertz of bandwidth (dBc/Hz). To know how to model LO phase noise in an oscillator see, “Model LO Phase Noise” on page 2-8.

Noise Figure

Noise figure value determines the degradation of signal to noise ratio of a signal as it goes through the network. Noise figure is defined by the equation:

$$N_f = \frac{\frac{Signal}{Noise}\text{at the input}}{\frac{Signal}{Noise}\text{at the output}}$$

Excessive noise figure in the system causes the noise to overwhelm the signal, making the signal unrecoverable. Noise figure is a function of frequency but it is independent of the bandwidth of the system. Noise figure is expressed in dB.

In RF Blockset, you can specify noise figure for an RF system using the Amplifier or Mixer blocks.

See Also

More About

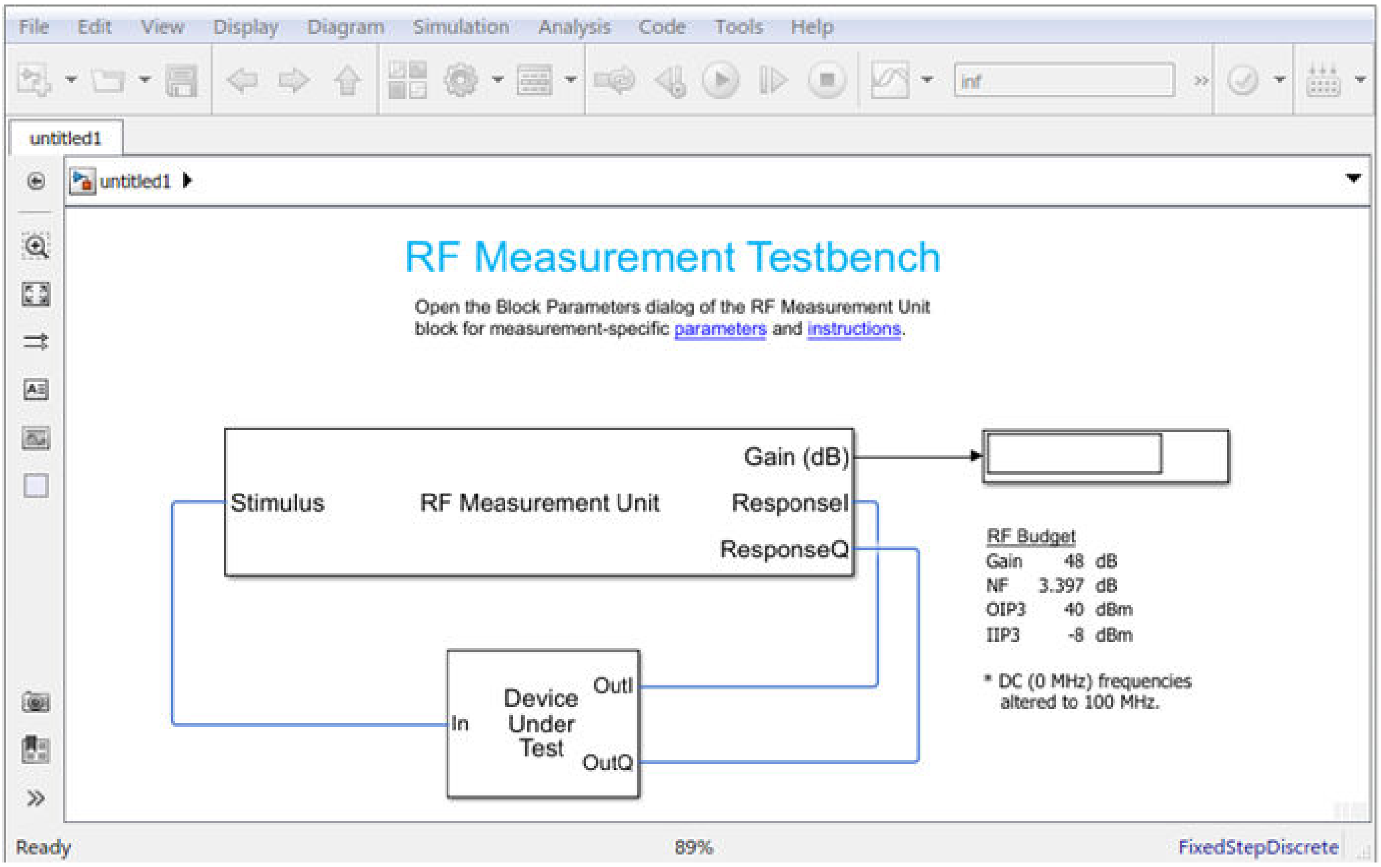
- “Model System Noise Figure” on page 1-2

Testbenches

Use RF Measurement Testbench for RF-to-IQ Converter

In this section...

Use the RF Measurement Testbench to measure various quantities of an RF-to-IQ converter. Measurable quantities include cumulative gain, noise figure, and nonlinearity (IP3) values. To open the testbench and measure the quantities, use the **RF Budget Analyzer** app to create an RF-to-IQ converter and then click `Export` > `Measurement testbench`.



RF Budget

Gain	48	dB
NF	3.397	dB
OIP3	40	dBm
IIP3	-8	dBm

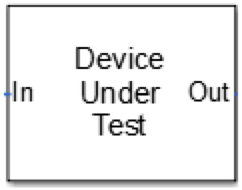
* DC (0 MHz) frequencies altered to 100 MHz.

The testbench has two subsystems:

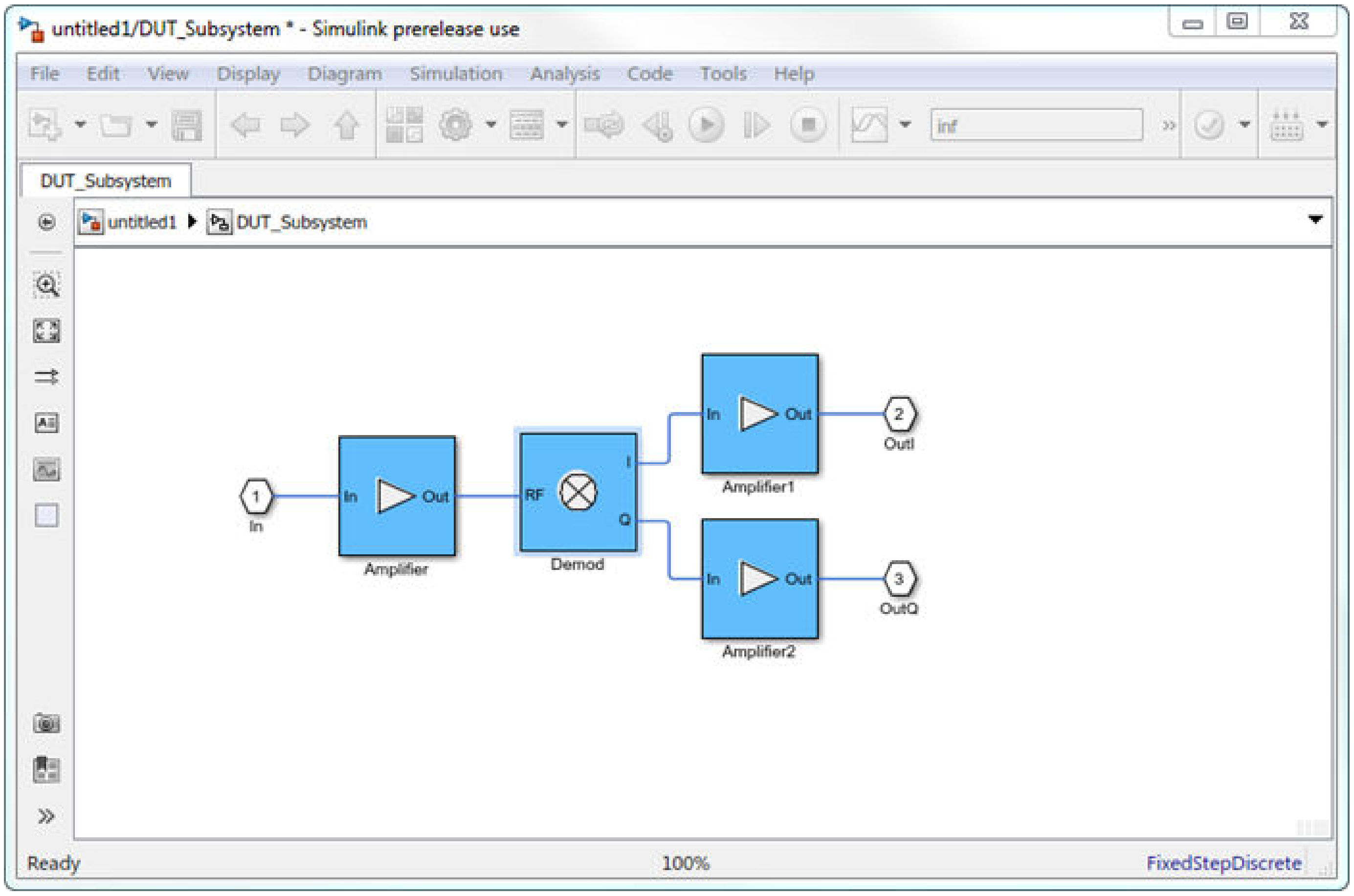
- `RF Measurement Unit`
- `Device Under Test`

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

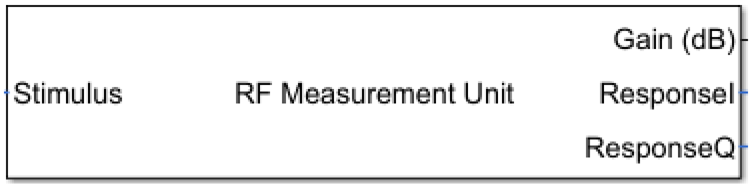
Device Under Test



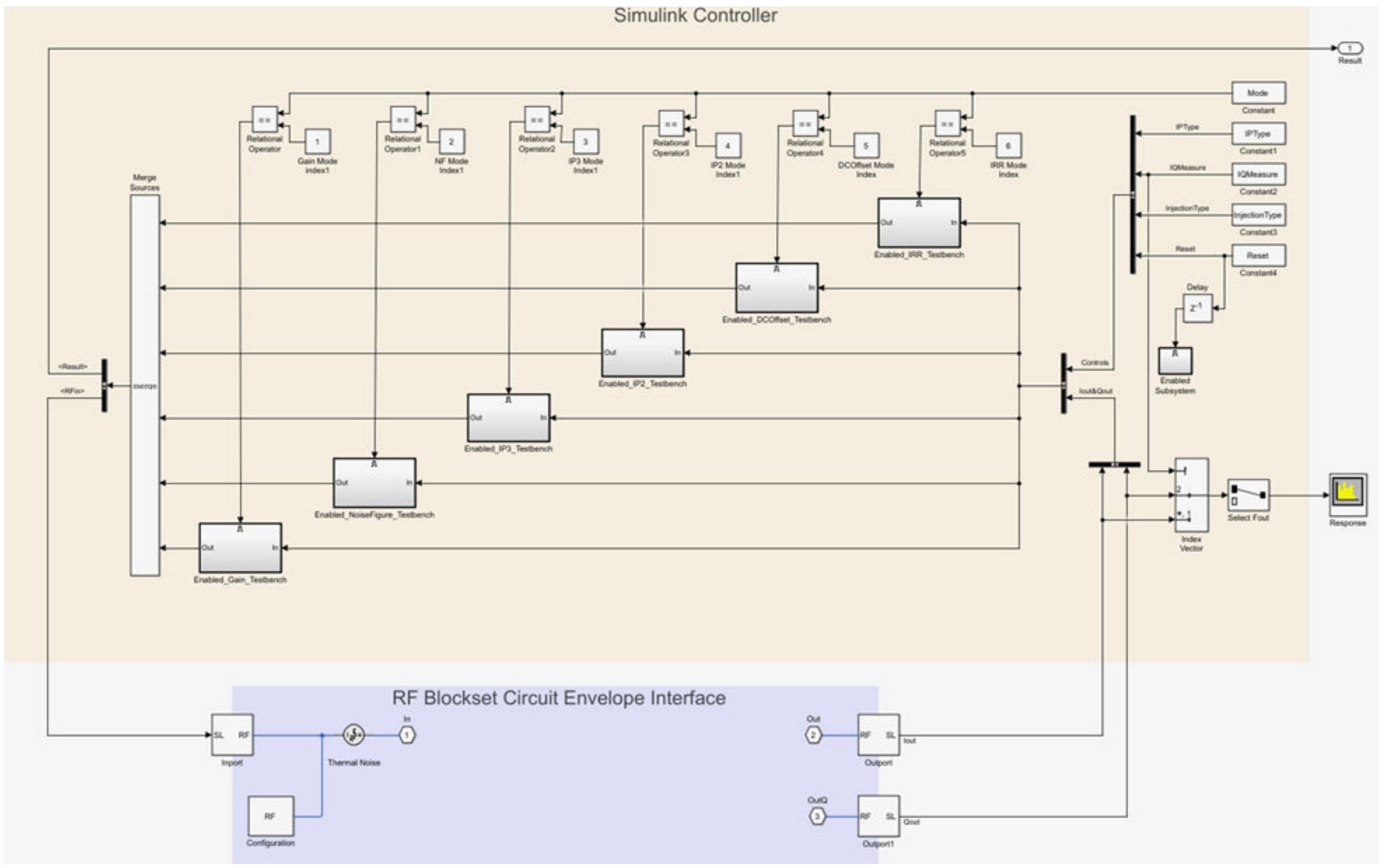
The Device Under Test subsystem contains the RF system exported from the app.



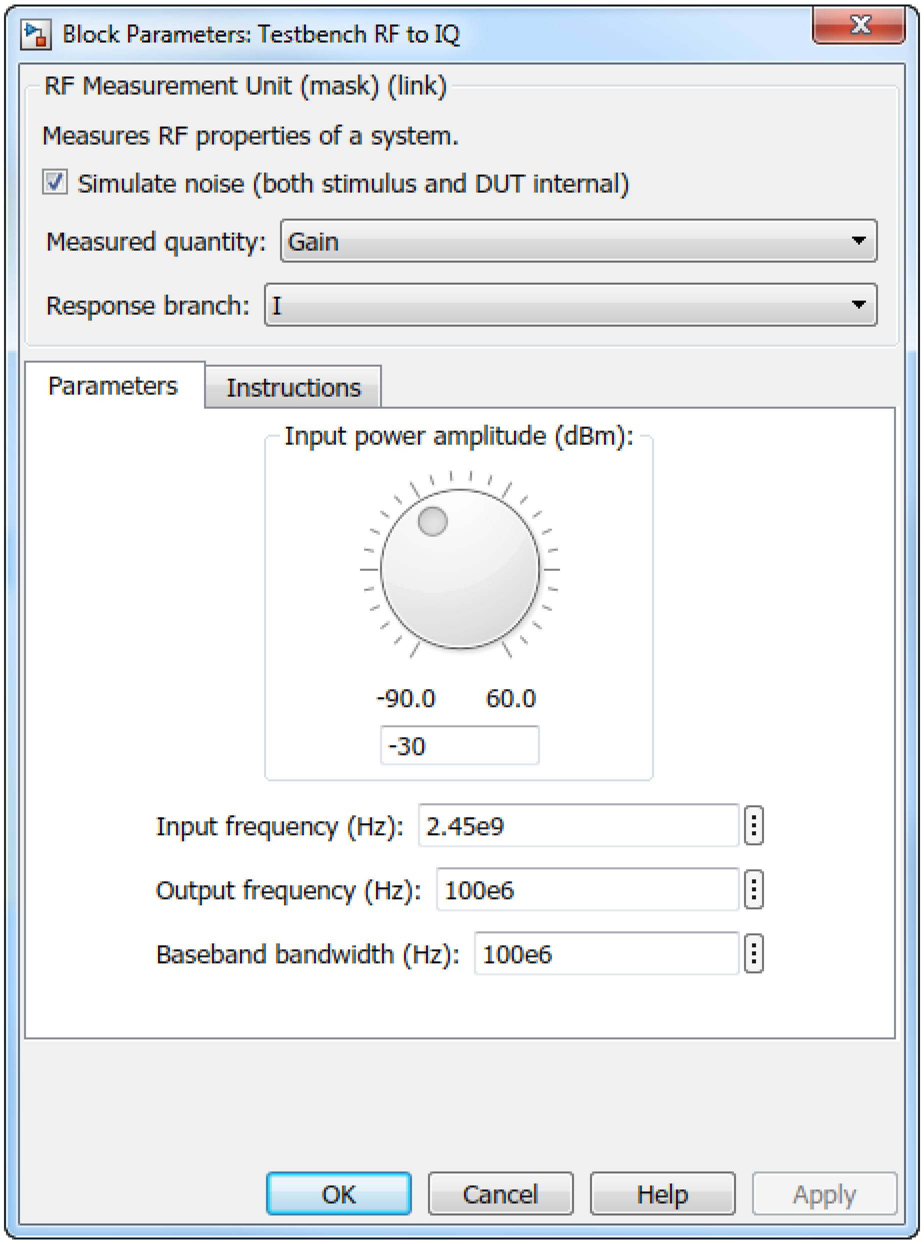
RF Measurement Unit



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


Simulink Controller
Result
Mode
Constant
IPType
Constant1
IQMeasure
Constant2
InjectionType
Constant3
Reset
Constant4
Relational Operator
Gain Mode index1
Relational Operator1
NF Mode Index1
Relational Operator2
IP3 Mode Index1
Relational Operator3
IP2 Mode Index1
Relational Operator4
DCOffset Mode Index
Relational Operator5
IRR Mode Index
Merge Sources
merge
<Result>
<RFin>
Enabled_IRR_Testbench
Enabled_DCOffset_Testbench
Enabled_IP2_Testbench
Enabled_IP3_Testbench
Enabled_NoiseFigure_Testbench
Enabled_Gain_Testbench
Out
In
Controls
Iout&Qout
Delay
Enabled Subsystem
Index Vector
Select Fout
Response
RF Blockset Circuit Envelope Interface
Inport
Thermal Noise
Configuration
Outport
Outport1
OutQ
Iout
Qout

RF Measurement Unit Parameters



Block Parameters: Testbench RF to IQ

RF Measurement Unit (mask) (link)

Measures RF properties of a system.

☑ Simulate noise (both stimulus and DUT internal)

Measured quantity: Gain

Response branch: I

Input power amplitude (dBm):

-30

Input frequency (Hz): 2.45e9

Output frequency (Hz): 100e6

Baseband bandwidth (Hz): 100e6

- **Simulate noise (both stimulus and DUT)** — Select this check box to enable noise modeling in the stimulus signal entering the DUT and inside the DUT.
- **Measured quantity** — Choose the quantity you want to measure:

  - `Gain` – Measure the transducer gain of the converter, assuming a load of 50 ohm. If you choose only `I` or only `Q` from **Response branch**, you see only half the value of the measured gain.
  - `NF` – Measure the noise figure value at the output of the converter.
  - `IP3` – Measure the output or input third-order intercept (IP3).
  - `IP2` – Measure the output or input second-order intercept (IP2).

  - `DC Offset` – Measure the DC level interference centered on the desired signal due to LO leakage mixing with input signal.
  - `Image Rejection Ratio` – Measure the image rejection ratio required to cancel the effect of images at the RF input signal.

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

  By default, the testbench measures `Output referred`. This option is available when you set the **Measured quantity** to `IP2` or `IP3`.
- **Injection Type** – Choose the local oscillator (LO) injection for image rejection ratio: `Low-side` or `High-side`.

  By default, the testbench measures the `Low-side`. This option is available when you set the **Measured quantity** to `Image Rejection Ratio`.
- **Response branch** — Choose the output branch you want to measure from:

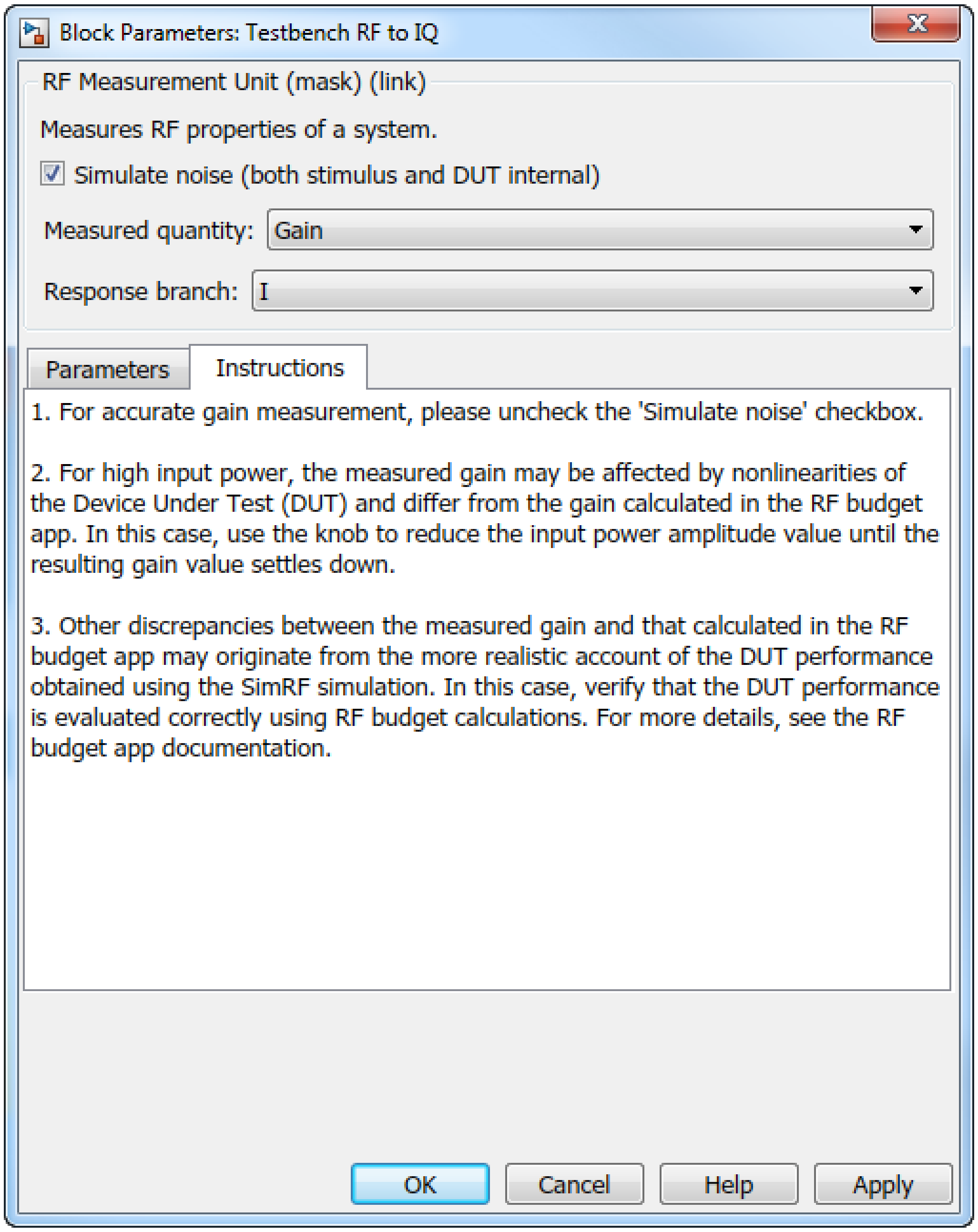
  - `I only(Q=0)` – Output signal measured at the in-phase branch.
  - `Q only(I=0)` – Output signal measured at the quadrature branch.

  This option is not available when you set the **Measured quantity** to `Image Rejection Ratio`.

Parameters Tab

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

Instructions Tab


Block Parameters: Testbench RF to IQ
RF Measurement Unit (mask) (link)
Parameters
Instructions
OK
Cancel
Help
Apply

Measures RF properties of a system.

☑ Simulate noise (both stimulus and DUT internal)

Measured quantity: Gain

Response branch: I

1. For accurate gain measurement, please uncheck the 'Simulate noise' checkbox.

2. For high input power, the measured gain may be affected by nonlinearities of the Device Under Test (DUT) and differ from the gain calculated in the RF budget app. In this case, use the knob to reduce the input power amplitude value until the resulting gain value settles down.

3. Other discrepancies between the measured gain and that calculated in the RF budget app may originate from the more realistic account of the DUT performance obtained using the SimRF simulation. In this case, verify that the DUT performance is evaluated correctly using RF budget calculations. For more details, see the RF budget app documentation.

Instructions for Gain Measurement

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

Instructions for NF Measurement

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

Instructions for IP3 and IP2 Measurement

- Clear **Simulate noise (both stimulus and DUT)** for accurate IP3 and IP2 measurement.
- Change **Input power amplitude (dBm)** or turn the knob to reduce the input power amplitude. For high input power, higher-order nonlinearities in the DUT can affect the OIP3 and IIP3 measurements.

Instructions for DC Offset Measurement

- Clear **Simulate noise (both stimulus and DUT)** for accurate DC offset measurement.
- Correct calculation of the DC offset assumes a frequency-independent system in the frequencies surrounding the test tones. Reduce the frequency separation between the test tones or reduce the baseband bandwidth until this condition is fulfilled. In common RF systems, the bandwidth is reduced below `1 KHz` for DC offset testing.
- . Change **Input RMS voltage amplitude (dBmV)** or turn the knob to reduce the input RMS voltage amplitude. For high input RMS voltage, higher-order nonlinearities in the DUT can affect the DC offset measurements

Instructions for Image Rejection Ratio

- Clear **Simulate noise (both stimulus and DUT)** for accurate OIP3 and IIP3 measurement.
- Correct calculation of the image rejection ratio (IRR) assumes a frequency-independent system in the frequencies surrounding the test tones. Reduce the frequency separation between the test tones or reduce the baseband bandwidth until this condition is fulfilled. In common RF systems, the bandwidth is reduced below `1 KHz` for IRR testing.
- . Change **Input power amplitude (dBm)** or turn the knob to reduce the input power amplitude. For high input power, higher-order nonlinearities in the DUT can affect the image rejection ratio measurement.

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

See Also

Amplifier | Mixer | **RF Budget Analyzer**

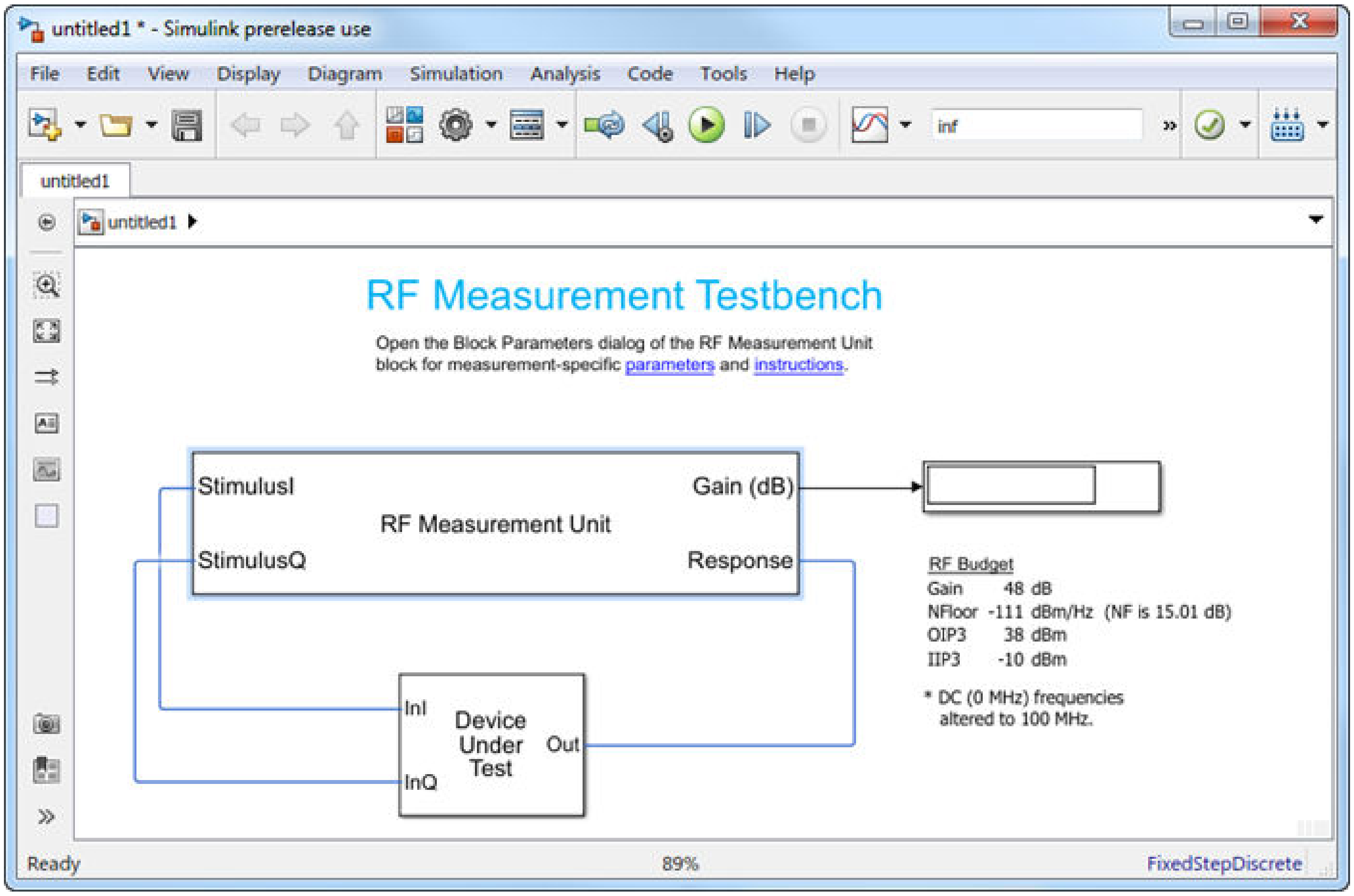
More About

- "Using RF Measurement Testbench for IQ-to-RF Converter" on page 4-10
- "Using RF Measurement Testbench"

Using RF Measurement Testbench for IQ-to-RF Converter

In this section...
"Device Under Test" on page 4-11
"RF Measurement Unit" on page 4-11
"RF Measurement Unit Parameters" on page 4-13

Use the RF Measurement Testbench to measure various quantities of an IQ-to-RF converter system. Measurable quantities include cumulative gain, noise figure, and nonlinearity (IP3) values. To open the testbench and measure the quantities, use the **RF Budget Analyzer** app to create an RF system and then click `Export` > `Measurement testbench`.



RF Measurement Testbench

Open the Block Parameters dialog of the RF Measurement Unit block for measurement-specific parameters and instructions.

RF Budget
Gain 48 dB
NFloor -111 dBm/Hz (NF is 15.01 dB)
OIP3 38 dBm
IIP3 -10 dBm

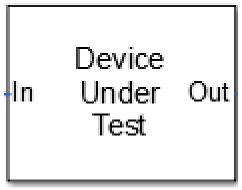
* DC (0 MHz) frequencies altered to 100 MHz.

The testbench has two subsystems:

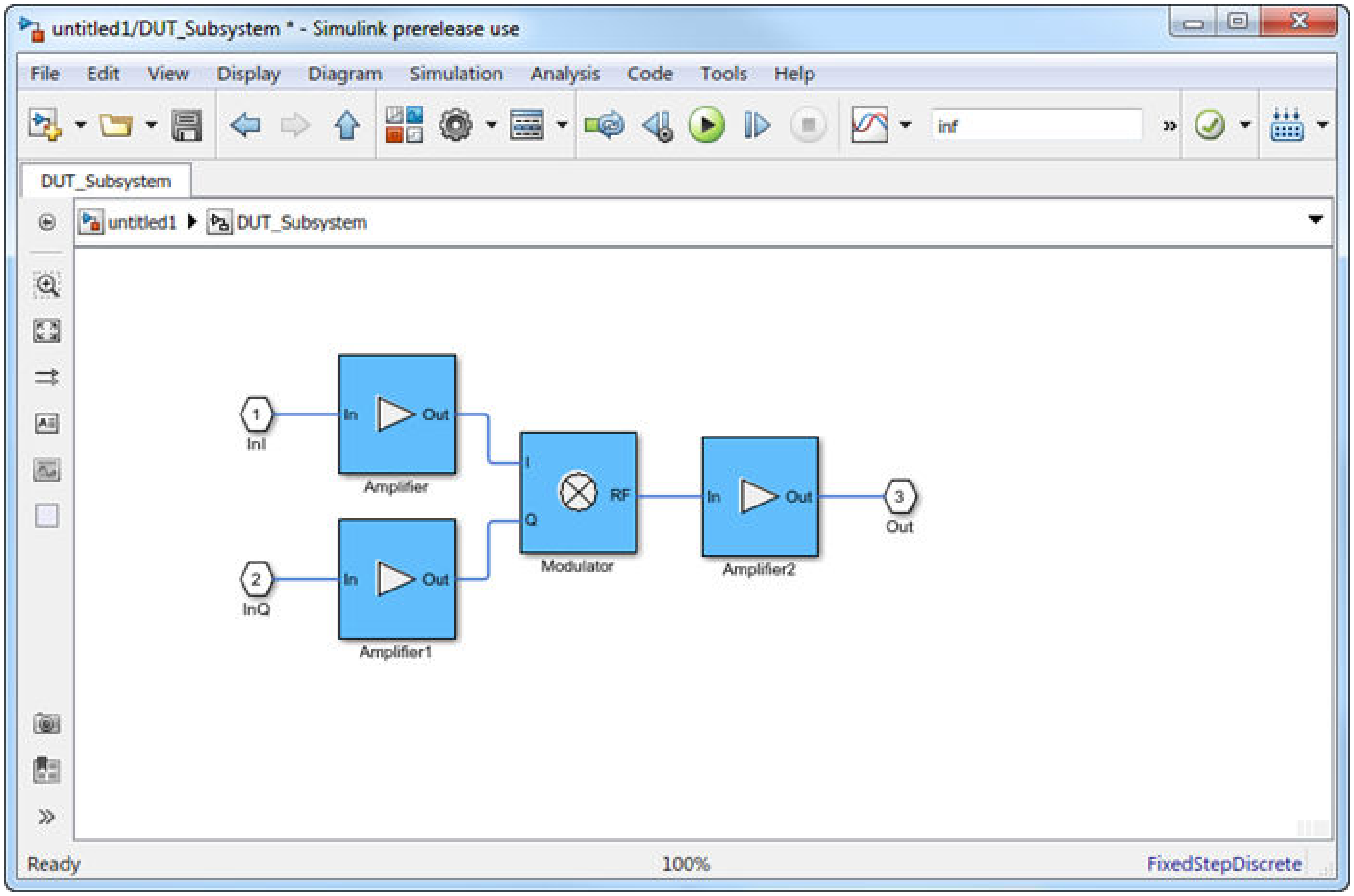
- `RF Measurement Unit`
- `Device Under Test`

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

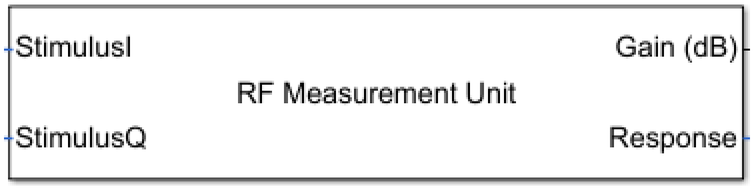
Device Under Test



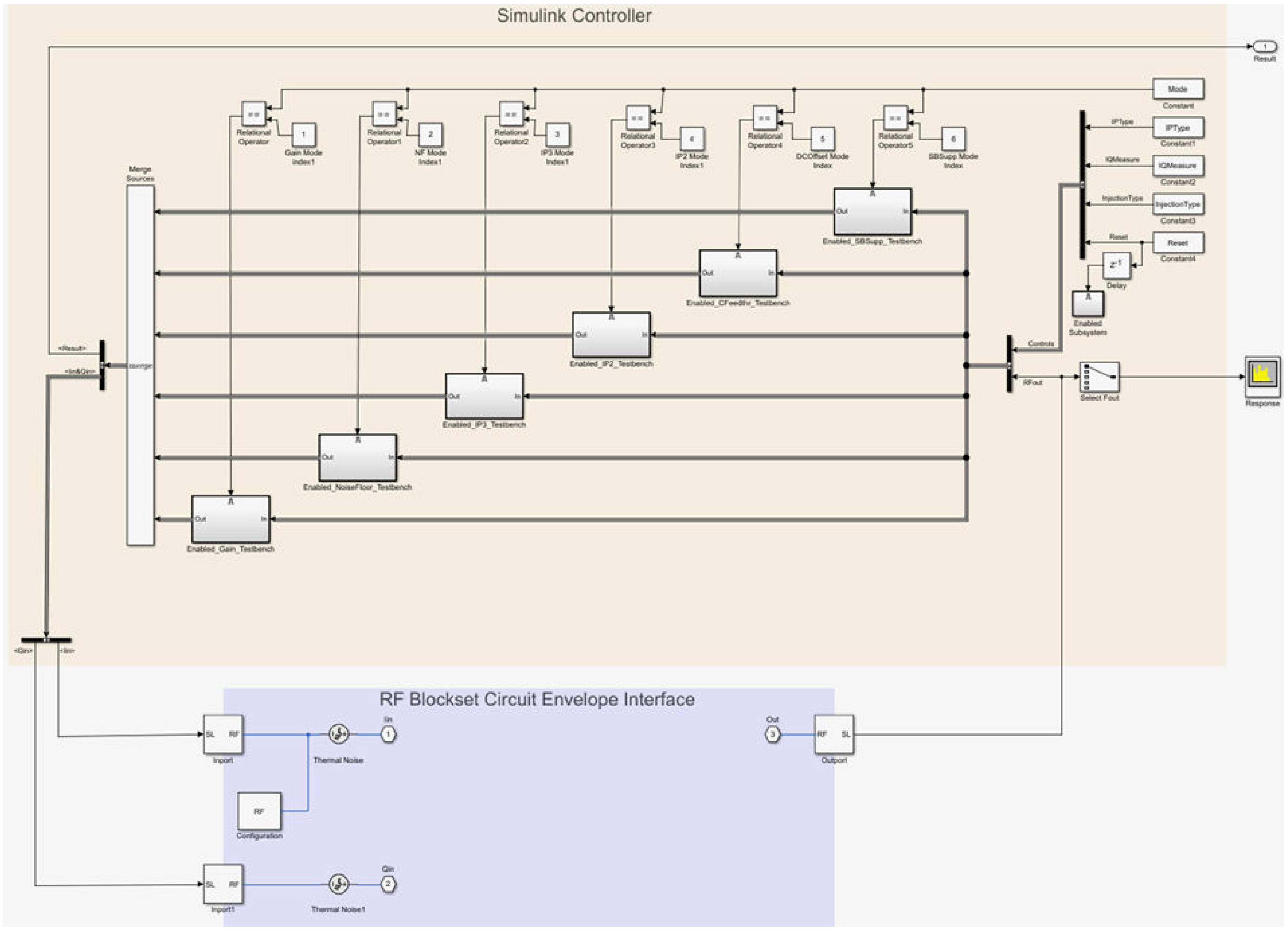
The Device Under Test subsystem contains the RF system exported from the app.



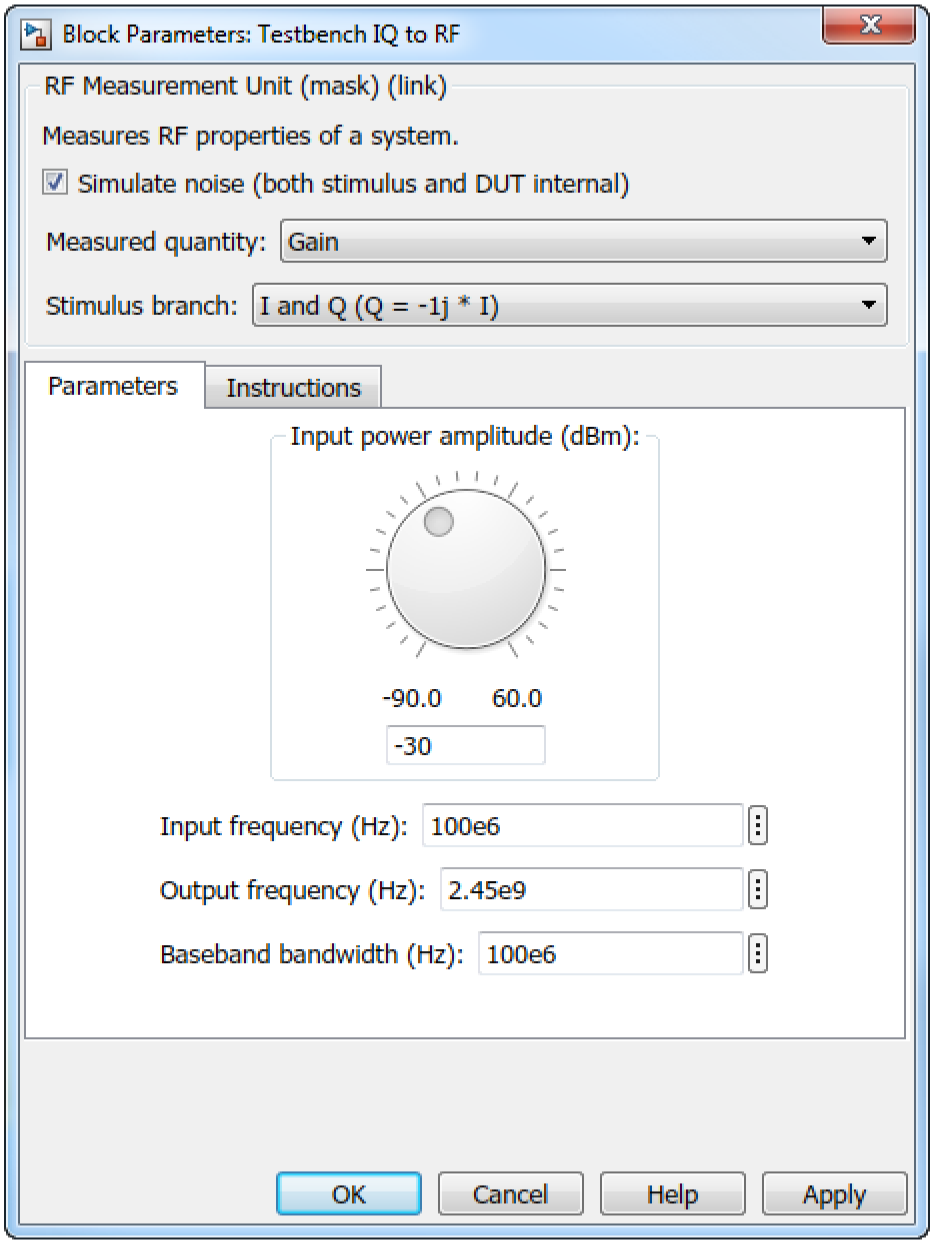
RF Measurement Unit



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


Simulink Controller
Result
Mode
Constant
IPType
Constant1
IQMeasure
Constant2
InjectionType
Constant3
Reset
Constant4
Delay
Enabled Subsystem
Controls
RFout
Select Fout
Response
Relational Operator
Gain Mode index1
NF Mode Index1
IP3 Mode Index1
IP2 Mode Index1
DCOffset Mode Index
SBSupp Mode Index
Merge Sources
merge
Enabled_SBSupp_Testbench
Enabled_CFeedthr_Testbench
Enabled_IP2_Testbench
Enabled_IP3_Testbench
Enabled_NoiseFloor_Testbench
Enabled_Gain_Testbench
<Result>
<Iin&Qin>
<Qin>
<Iin>
RF Blockset Circuit Envelope Interface
Inport
Thermal Noise
Iin
Configuration
Inport1
Thermal Noise1
Qin
Out
Outport

RF Measurement Unit Parameters


Block Parameters: Testbench IQ to RF

RF Measurement Unit (mask) (link)

Measures RF properties of a system.

☑ Simulate noise (both stimulus and DUT internal)

Measured quantity: Gain

Stimulus branch: I and Q (Q = -1j * I)

Input power amplitude (dBm):

-30

Input frequency (Hz): 100e6

Output frequency (Hz): 2.45e9

Baseband bandwidth (Hz): 100e6

- **Simulate noise (both stimulus and DUT)** — Select this check box to enable noise modeling in the stimulus signal entering the DUT and inside the DUT.
- **Measured quantity** — Choose the quantity you want to measure:

  - `Gain` – Measure the transducer gain of the converter. If you choose only `I` or only `Q` from **Stimulus branch**, you only see half the value of the measured gain.
  - `Noise Floor` – Measure the noise floor value of the converter.
  - `IP3` – Measure the output or input third-order intercept (IP3).
  - `IP2` – Measure the output or input second-order intercept (IP2).

- `Carrier Feedthrough` – Measure the leakage of carrier tone into the RF spectrum due to imbalances in the in-phase and quadrature phase inputs.
- `Sideband Suppression` – Measure the sideband suppression required for the ideal cancellation of image signals around the RF output signal.

By default, the testbench measures `Gain`. The contents in the **Instructions** tab changes according to the **Measured quantity** value.

- **IP Type** – Choose the type of intercept points (IP) to measure: `Output referred` or `Input referred`,

  By default, the testbench measures `Output referred`. This option is available when you set the **Measured quantity** to `IP2` or `IP3`.
- **Injection Type** – Choose the local oscillator (LO) injection for sideband suppression: `Low-side` or `High-side`,

  By default, the testbench measures `Low-side`. This option is available when you set the **Measured quantity** to `Sideband Suppression`.
- **Stimulus branch** — Choose the branch you want to use as input stimulus for the measurement:

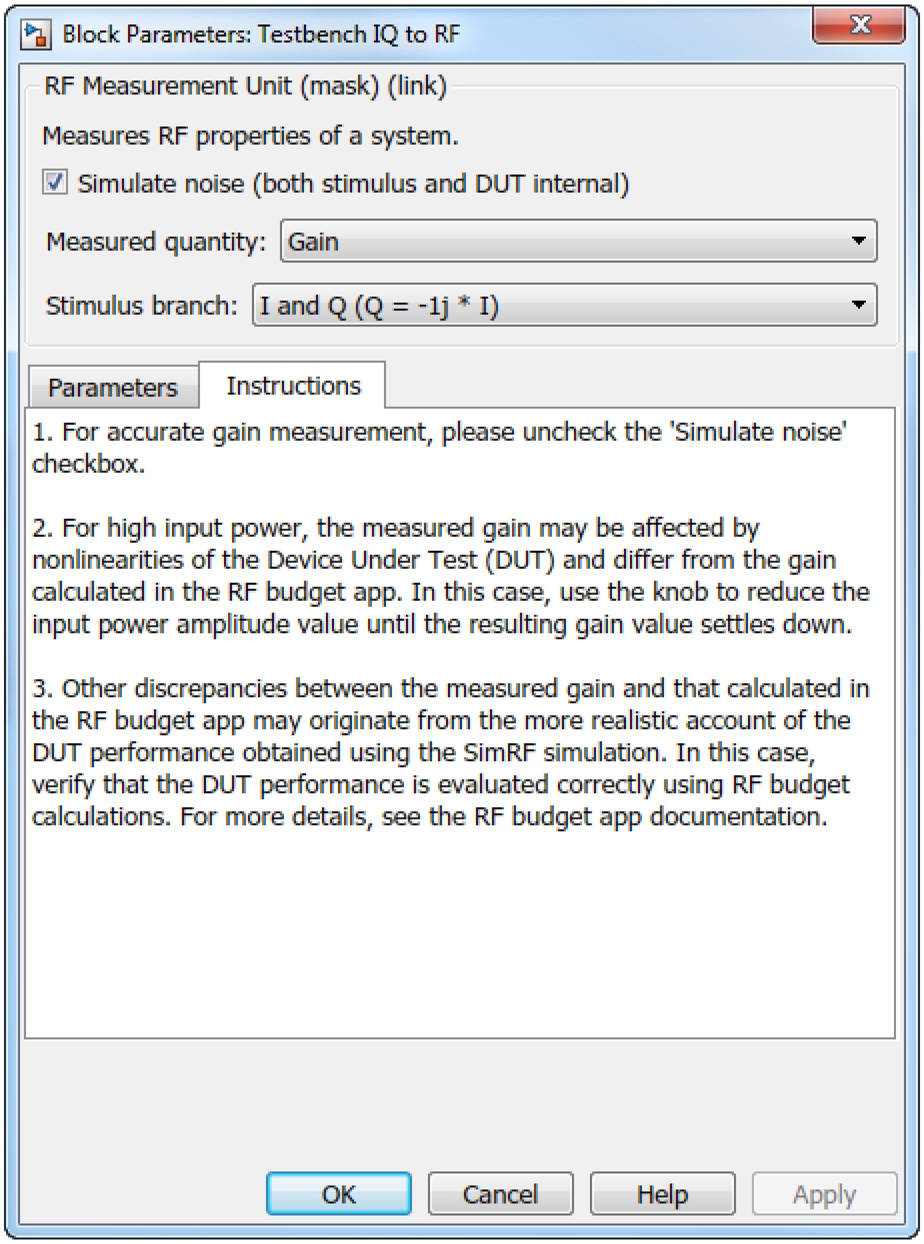
  - `I and Q (Q=-j*I)` — Signal measured is based on a combination of input signals. Quadrature input is same as the in-phase input but is –90 degrees out of phase.
  - `I and Q (Q=j*I)` — Signal measured is based on a combination of input signals. Quadrature input is same as the in-phase input but is 90 degrees out of phase.
  - `I only(Q=0)` — Signal measured is only the output of the in-phase input signal. Gain measured using this input is only one quarter of the total output signal gain.
  - `Q only(I=0)` — Signal verified is only the output of the quadrature input signal. Gain measured using this input is only one quarter of the total output signal gain.

Parameters Tab

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

  This option is only available when you set **Measured quantity** to `IP2`, `IP3`, and `Carrier Feedthrough`.

Instructions Tab


Block Parameters: Testbench IQ to RF
RF Measurement Unit (mask) (link)
Parameters
Instructions
OK
Cancel
Help
Apply

Measures RF properties of a system.

☑ Simulate noise (both stimulus and DUT internal)

Measured quantity: Gain

Stimulus branch: I and Q (Q = -1j * I)

1. For accurate gain measurement, please uncheck the 'Simulate noise' checkbox.

2. For high input power, the measured gain may be affected by nonlinearities of the Device Under Test (DUT) and differ from the gain calculated in the RF budget app. In this case, use the knob to reduce the input power amplitude value until the resulting gain value settles down.

3. Other discrepancies between the measured gain and that calculated in the RF budget app may originate from the more realistic account of the DUT performance obtained using the SimRF simulation. In this case, verify that the DUT performance is evaluated correctly using RF budget calculations. For more details, see the RF budget app documentation.

Instructions for Gain Measurement

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

Instructions for Noise Floor Measurement

- The testbench measures the spot noise floor calculated. This calculation assumes a frequency-independent system within a given bandwidth. To simulate a frequency-independent system and calculate the correct noise floor value, reduce the baseband bandwidth until this condition is fulfilled. In common RF systems, the bandwidth is reduced below 1 kHz for noise floor testing.

- Change **Input power amplitude (dBm)** or turn the knob to reduce or increase the input power amplitude. For high input power, nonlinearities in the DUT can affect the noise floor measurements.

Instructions for IP3 and IP2 Measurement

- Clear **Simulate noise (both stimulus and DUT)** for accurate IP3 and IP2 measurement.
- Change **Input power amplitude (dBm)** or turn the knob to reduce the input power amplitude. For high input power, higher-order nonlinearities in the DUT can affect the IP3 and IP2 measurements.

Instructions for Carrier Feedthrough Measurement

- Clear **Simulate noise (both stimulus and DUT)** for accurate IP3 and IP2 measurement.
- Change **Input RMS voltage amplitude (dBmV)** or turn the knob to reduce the input RMS voltage amplitude. For high input RMS voltage, higher-order nonlinearities in the DUT can affect the carrier feedthrough measurements
- Correct calculation of the carrier feedthrough assumes a frequency-independent system in the frequencies surrounding the test tones. Reduce the frequency separation between the test tones or reduce the baseband bandwidth until this condition is fulfilled. In common RF systems, the bandwidth is reduced below `1 KHz` for carrier feedthrough testing.

Instructions for Sideband Suppression Measurement

- Clear **Simulate noise (both stimulus and DUT)** for accurate IP3 and IP2 measurement.
- Change **Input power amplitude (dBm)** or turn the knob to reduce the input power amplitude. For high input power, higher-order nonlinearities in the DUT can affect the sideband suppression measurement.

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

See Also

RF Budget Analyzer

RF Blockset Models

Analog Devices Transceiver Models

AD9361 Models

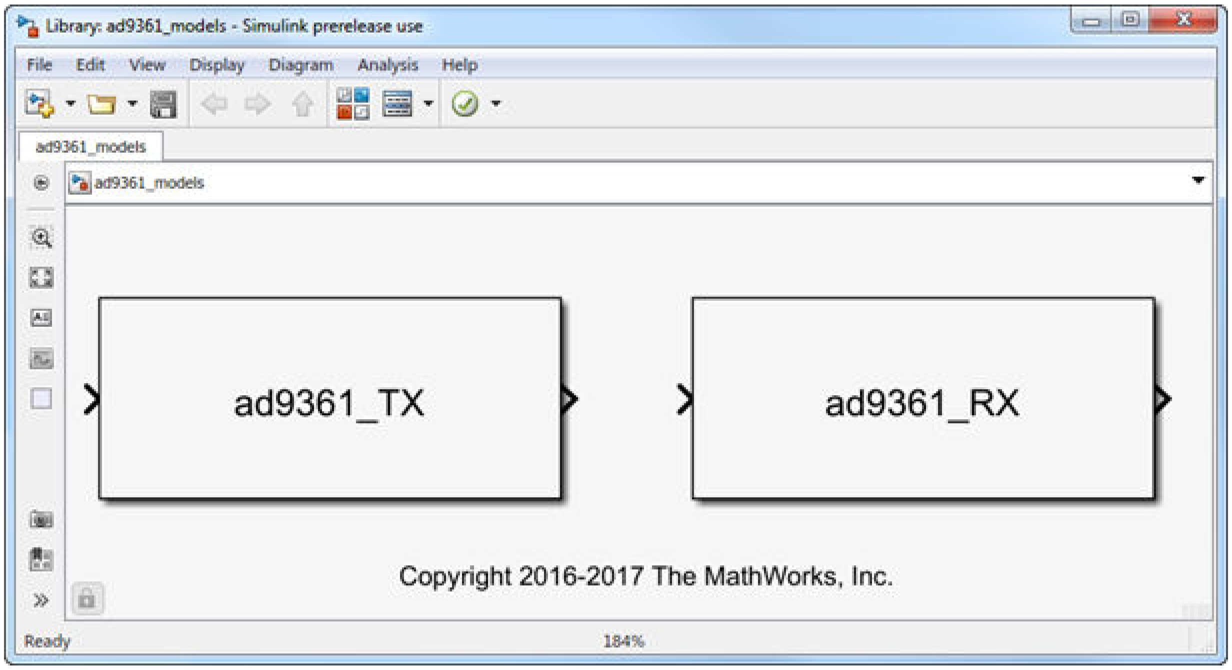
In this section...

You can use the AD9361 models to simulate Analog Devices® AD9361 RF transmitter or receiver designs. These models also helps to see the impact of RF imperfections on your transmitted or received signal.

Install Analog Devices RF Transceivers using, `simrfSupportPackages`. You can open models using the Simulink Library Browser and opening RF Blockset Models for Analog Devices RF Transceivers, or by typing the following in the MATLAB command prompt:

```
open ad9361_models
```

Choose the model you want from the library:


Library: ad9361_models - Simulink prerelease use
File Edit View Display Diagram Analysis Help
ad9361_models
ad9361_models
ad9361_TX
ad9361_RX
Copyright 2016-2017 The MathWorks, Inc.
Ready
184%

Note You require these additional licenses to run this model:

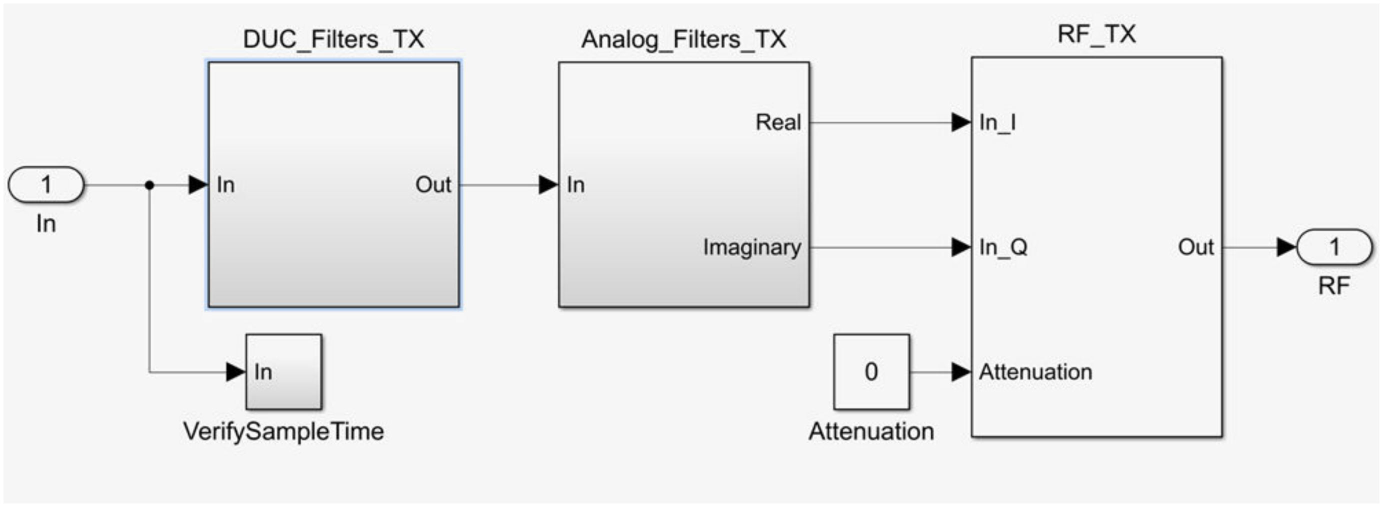
- Communications Toolbox™
- Stateflow®
- Fixed-Point Designer™

Complete documentation on how to use the models is available with the software download.

```
open(ad93xx_getdoc)
```

AD9361_TX Analog Devices Transmitter

Model Description



The transmitter model consists of three stages:

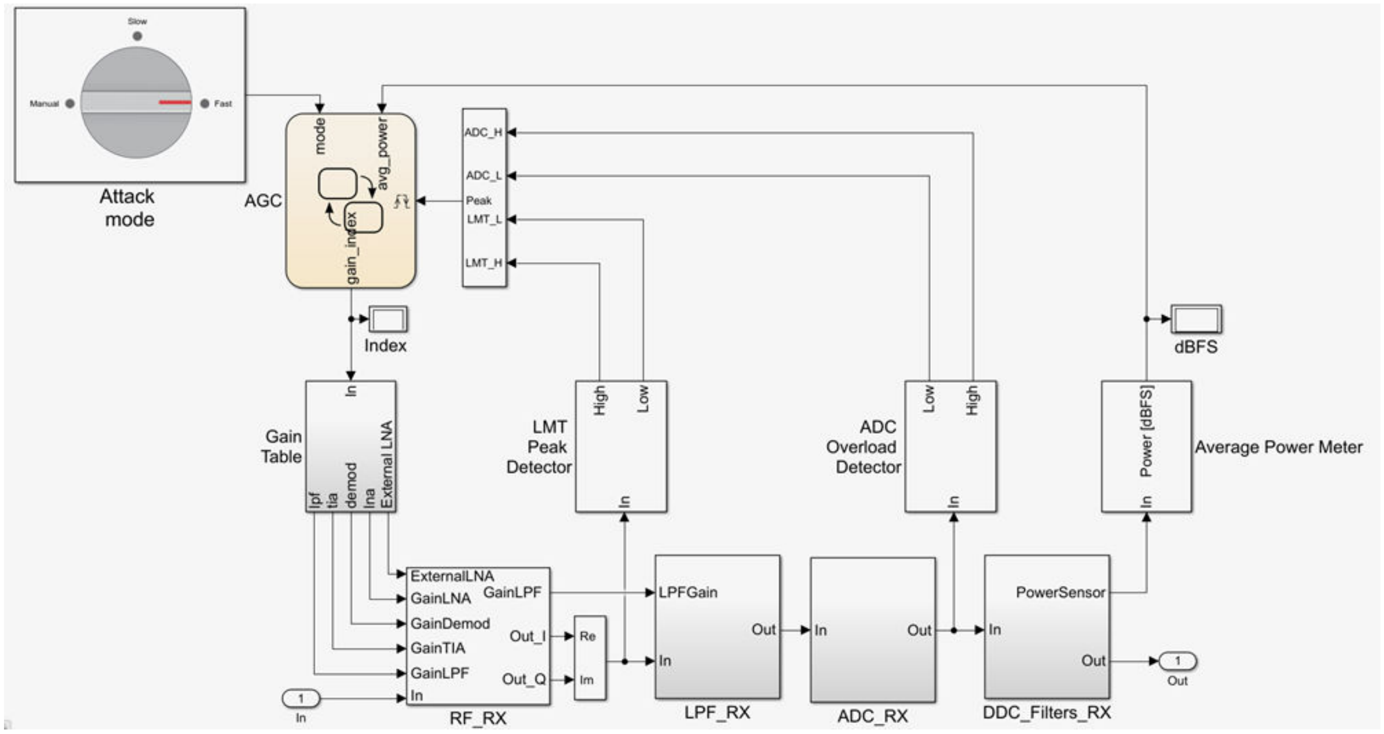
- Digital up-conversion filters (DUC_Filters_TX))
- Analog filters (Analog_Filters_TX)
- RF front end (RF_TX)

You can use the transmitter model to simulate the following behaviors:

- Tunable attenuation
- Oscillator phase noise
- Carrier-dependent noise floor
- Attenuation and carrier-dependent nonlinearities like output-referred third-order intercept (OIP3)
- Attenuation dependent gain imbalance
- Attenuation dependent local oscillator (LO) carrier leak

AD9361_RX Analog Devices Receiver

Model Description



The receiver model consists of:

- RF receiver (RF_RX)
- Analog filters (Analog_Filters_RX)
- Analog to Digital Converter (ADC_RX)
- Digital down-conversion filters (DDC)
- Receiver signal strength indicators
- Automatic gain control (AGC) state machine
- Gain table

You can use the receiver model to simulate the following behaviors:

- Tunable low-noise amplifier, mixer, and trans-impedance amplifier (LMT) gains
- Carrier-dependent noise floor
- Gain and carrier-frequency dependent nonlinearities like output-referred third-order intercept (OIP3)
- Gain and carrier-frequency dependent nonlinearities like output–referred second–order intercept (OIP2)
- Gain dependent gain imbalance
- Gain dependent local oscillator (LO) carrier leak

See Also

`simrfSupportPackages`

More About

- “AD9361 Testbenches” on page 5-6
- “AD9371 Models” on page 5-9

AD9361 Testbenches

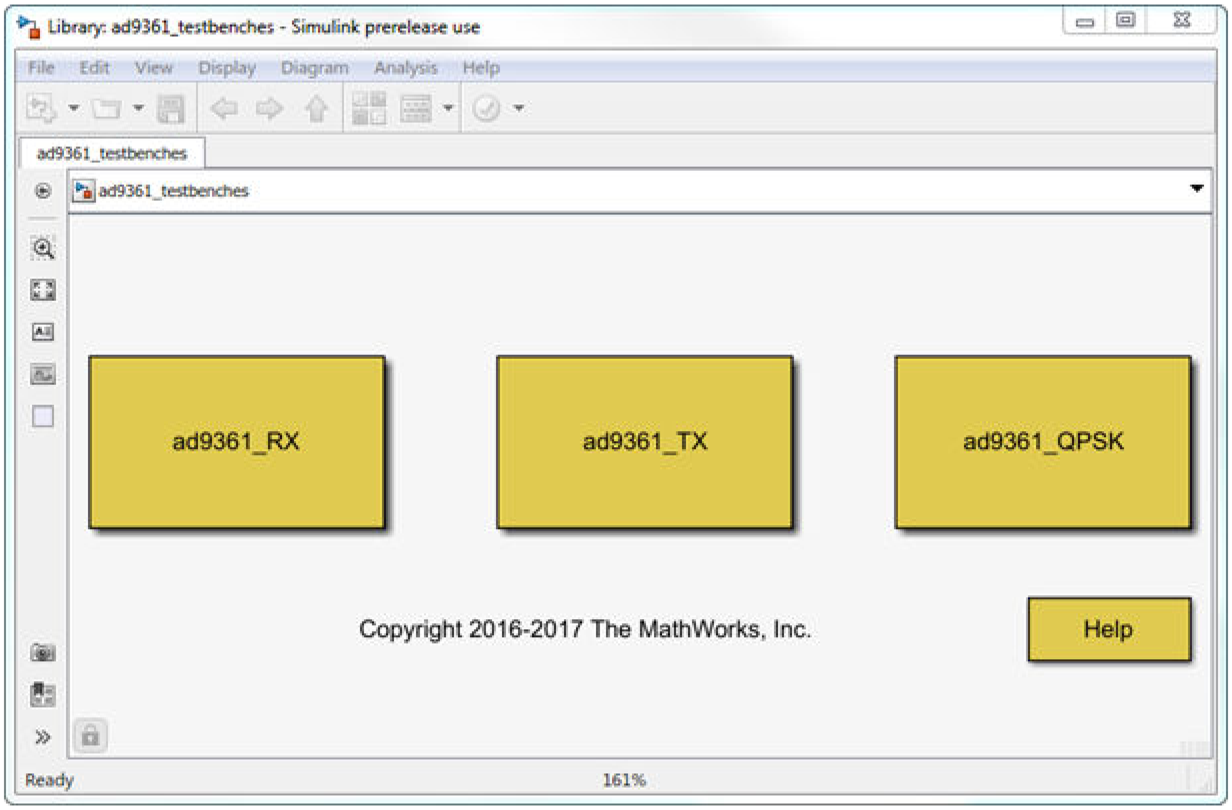
In this section...

You can use the AD9361 testbench models to analyze the functioning and values of Analog Devices AD9361 RF transmitter, receiver, or end-to-end designs.

Install Analog Devices RF Transceivers using `simrfSupportPackages`. You can open the testbench models using the Simulink library browser and opening RF Blockset Models for Analog Devices RF Transceivers, or by typing the following in the MATLAB command prompt:

```
open ad9361_testbenches
```

Click to open the AD9361 testbench model from the library:


Library: ad9361_testbenches - Simulink prerelease use
File Edit View Display Diagram Analysis Help
ad9361_testbenches
ad9361_testbenches
ad9361_RX
ad9361_TX
ad9361_QPSK
Copyright 2016-2017 The MathWorks, Inc.
Help
Ready
161%

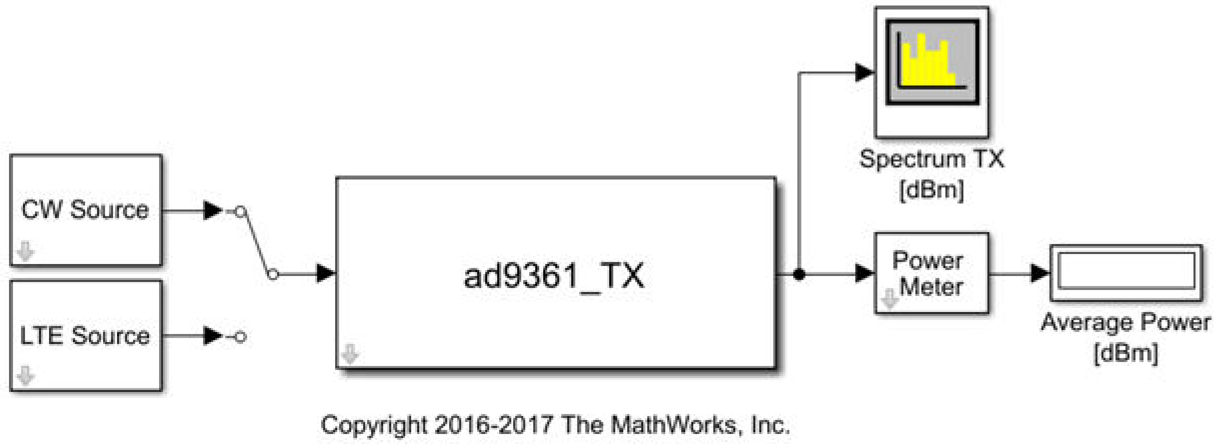
Note You require these additional licenses to run this model:

- Communications Toolbox
- Stateflow
- Fixed-Point Designer

Complete documentation on how to use the models is available with the software download.

```
open(ad93xx_getdoc)
```

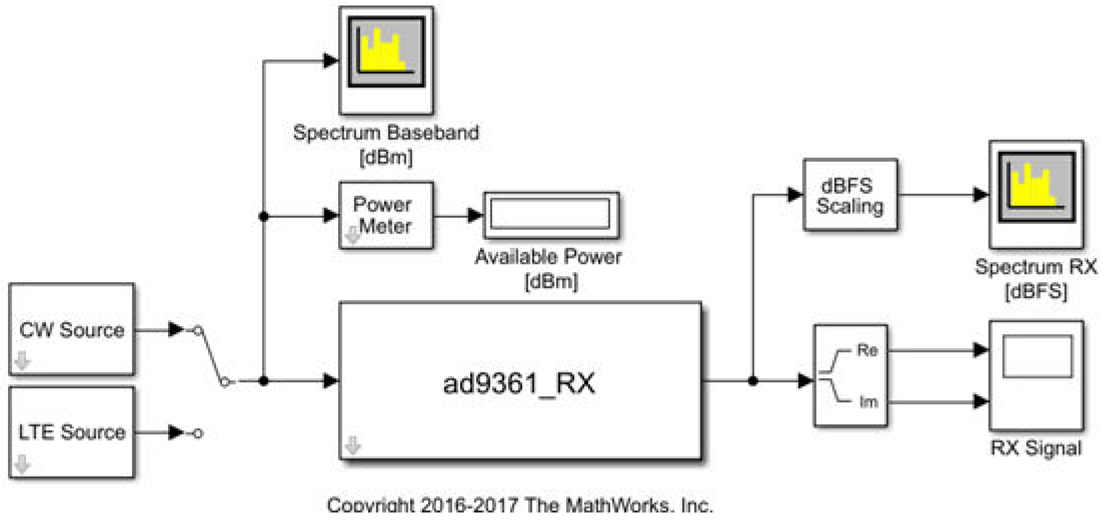
AD9361_TX Analog Devices Transmitter Testbench



The transmitter testbench consists of:

- CW and LTE Signal sources
- AD9361 transmitter which is the device under test
- Spectrum analyzer and power meter for signal visualization

AD9361_RX Analog Devices Receiver Testbench

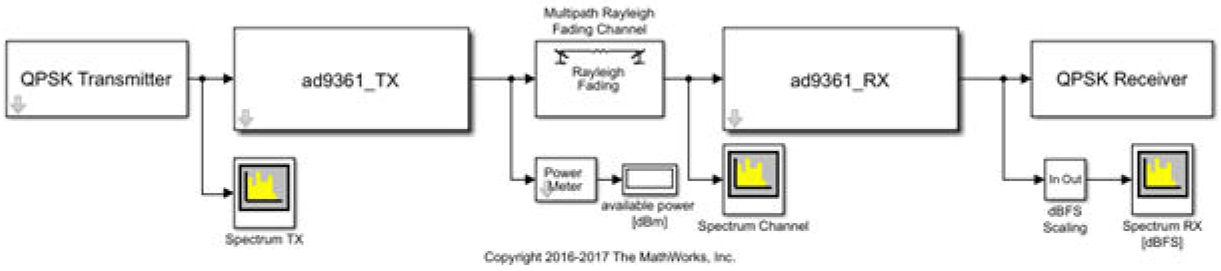


The receiver testbench consists of:

- CW and LTE Signal sources

- AD9361 receiver which is the device under test
- Spectrum analyzer and power meter for signal visualization

AD9361_QPSK Analog Devices Testbench



The QPSK testbench consists of:

- QPSK signal generator
- AD9361 transmitter operating at 2 GHz, with default LTE 5-MHz filter configuration
- Multi-path Rayleigh fading channel
- Multi-path Rayleigh fading channel
- AD9361 receiver operating at 2 GHz, with default LTE 5-MHz filter configuration
- QPSK baseband signal demodulator decoder.

See Also

`simrfSupportPackages`

More About

- “AD9371 Testbenches” on page 5-15
- “AD9361 Models” on page 5-2

AD9371 Models

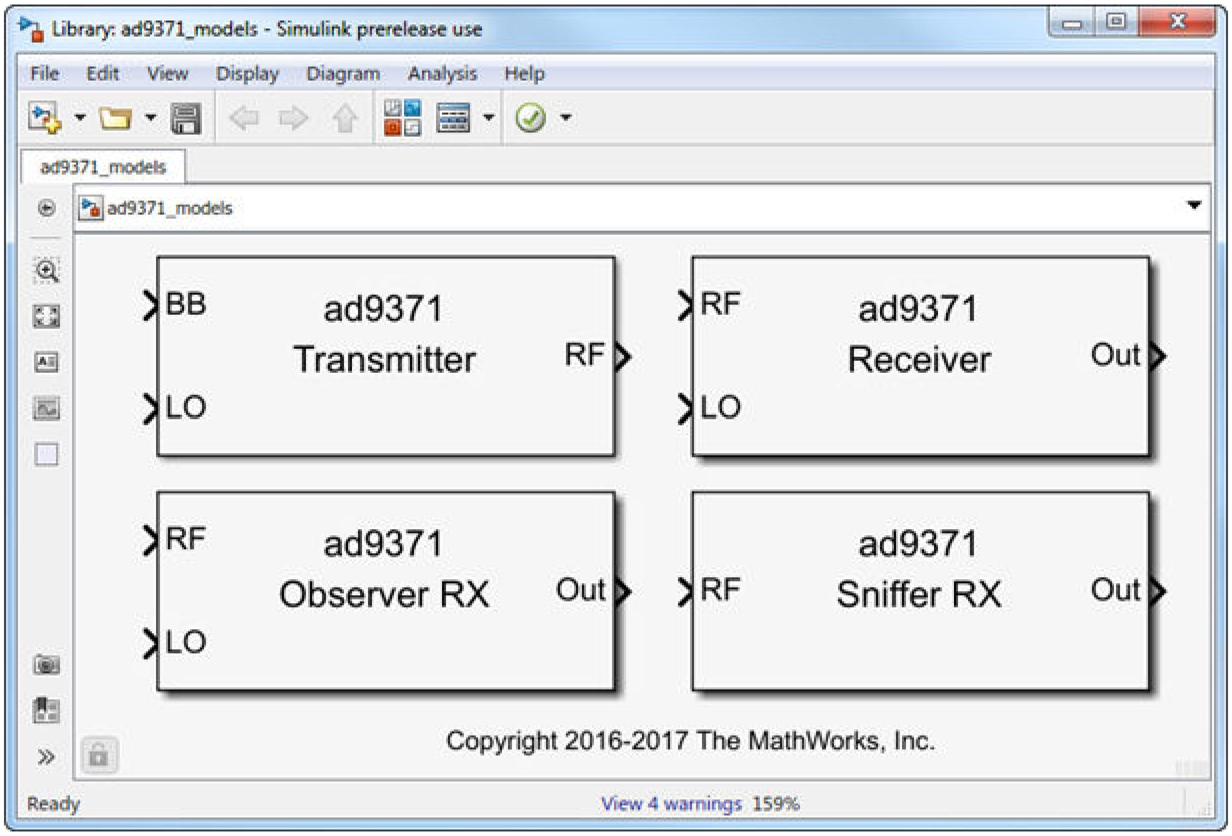
In this section...

You can use the AD9371 models to simulate Analog Devices AD9371 RF transmitter, receiver, observer, and sniffer designs. These models also helps to see the impact of RF imperfections on your transmitted or received signal.

Install Analog Devices RF Transceivers using `simrfSupportPackages`. You can open models using the Simulink Library Browser and opening RF Blockset Models for Analog Devices RF Transceivers, or by typing the following in the MATLAB command prompt:

```
open ad9371_models
```

Choose the model you want from the library:



Note You require these additional licenses to run this model:

- Communications Toolbox

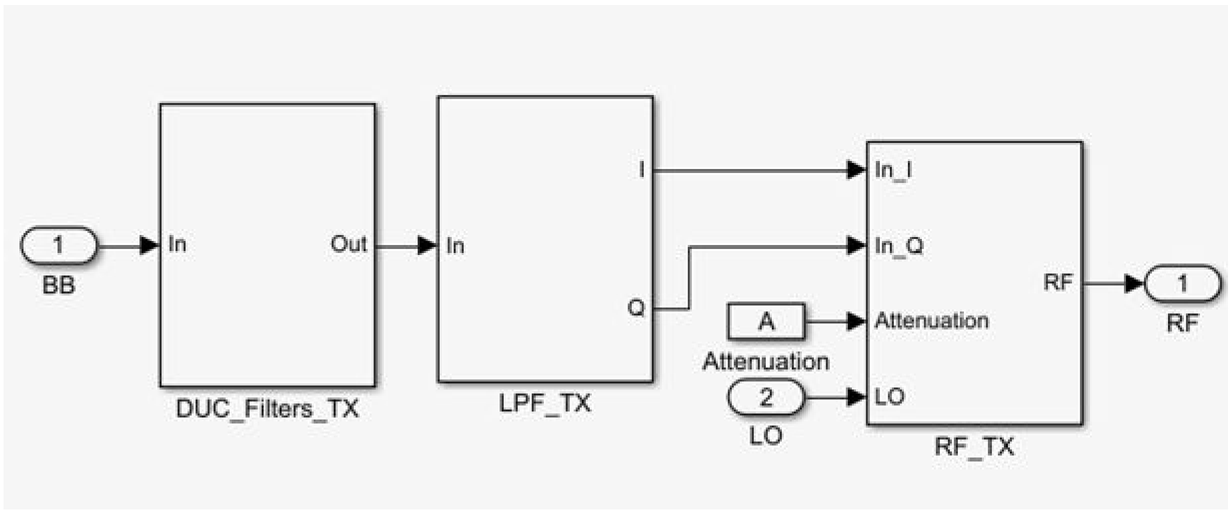
- Stateflow
- Fixed-Point Designer

Complete documentation on how to use the models is available with the software download.

```
open(ad93xx_getdoc)
```

AD9371_TX Analog Devices Transmitter

Model Description



The transmitter model consists of three stages:

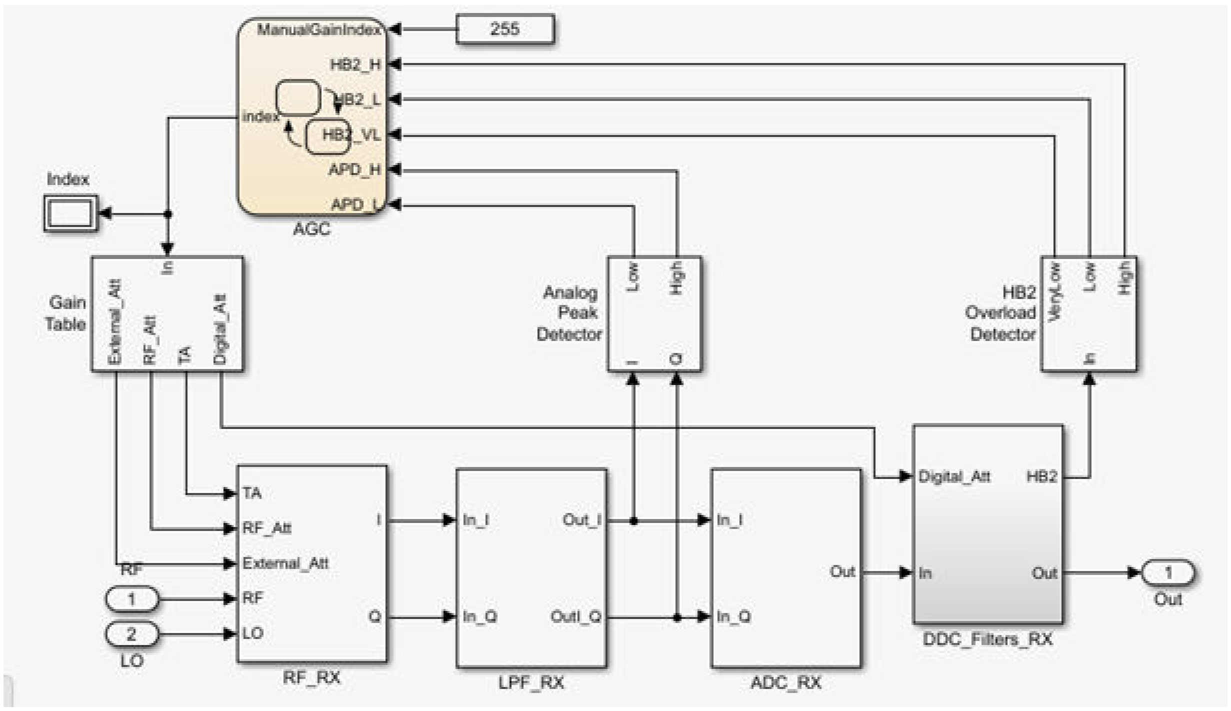
- Digital up-conversion filters (DUC_Filters_TX))
- Analog filters (Analog_Filters_TX)
- RF front end (RF_TX)

You can use the transmitter model to simulate the following behaviors:

- Tunable attenuation
- Attenuation and carrier-frequency dependent noise floor
- Attenuation and carrier-frequency dependent output-referred third-order intercept (OIP3) and output-referred second-order intercept (OIP2)
- Attenuation and carrier-frequency dependent gain imbalance (to model finite image rejection)
- Attenuation and carrier-frequency dependent local oscillator (LO) carrier leakage

AD9371_RX Analog Devices Receiver

Model Description



The receiver model consists of three stages:

- RF receiver (RF_RX)
- Analog filters (LPF_RX)
- Analog-to-Digital converter (ADC_RX)
- Digital down-conversion filters (DDC_Filters_RX)
- Receiver signal strength indicator (RSSI): two power meters to detect the strength of the received signal at different section of the chain (Analog Peak Detector, and HB2 Overload Detector)
- Automatic gain control state machine (AGC)
- Programmable gain table (Gain Table)

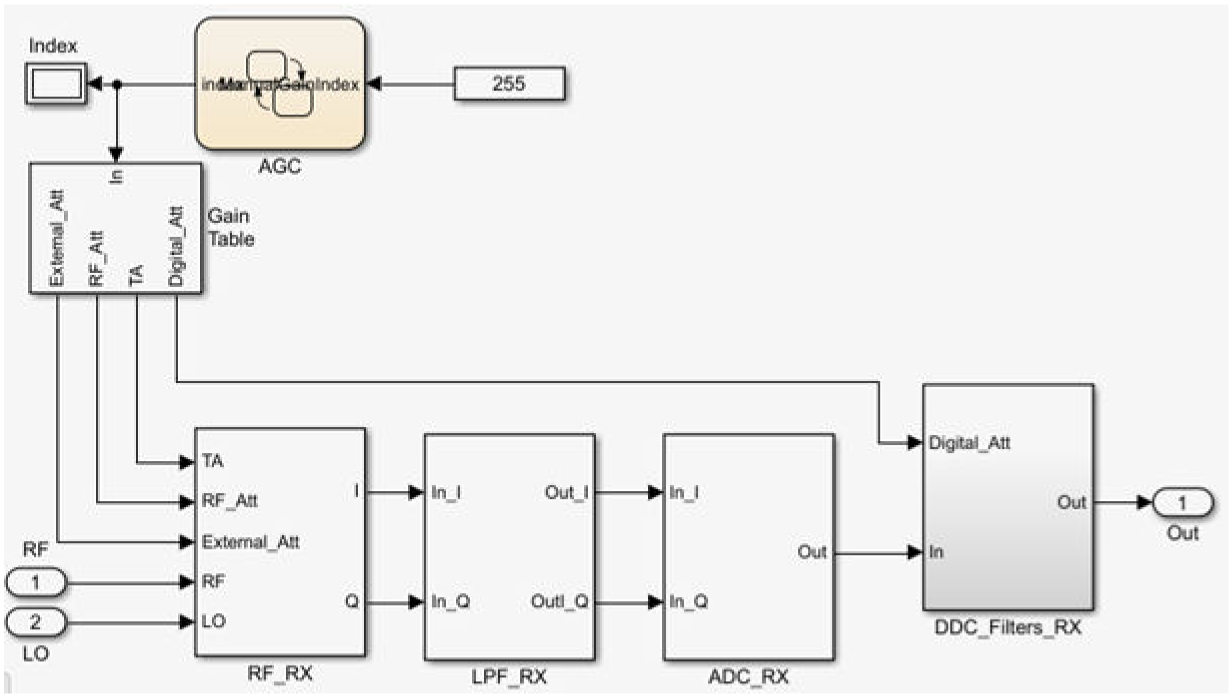
You can use the receiver model simulate the following behaviors:

- Tunable internal attenuation
- Attenuation and carrier-frequency dependent noise figure
- Attenuation and carrier-frequency dependent output-referred third-order intercept (OIP3) and output-referred second-order intercept (OIP2)
- Attenuation and carrier-frequency dependent gain imbalance (to model finite image rejection)
- Attenuation and carrier-frequency dependent local oscillator (LO) carrier leakage
- Analog filters provide continuous time signal
- ADC models a high-sampling rate third order delta-sigma modulator.

- Digital down conversion digital filters converts the highly sampled signal at the output of the ADC to a lower baseband rate.
- Received signal strength indicator measures power at two stages, at the RF receiver output and after the first half-band filter
- AGC changes the index of the gain table according to the flags of threshold crossing reported by the RSSI.
- Default gain table is read from the MATLAB file, `DefaultGainTables`. You can customize this file.

AD9371_ORX Analog Devices Observer Receiver

Model Description



The observer receiver model consists of three stages:

- RF receiver (RF_RX)
- Analog filters (LPF_RX)
- Analog to Digital converter (ADC_RX)
- Digital down-conversion filters (DDC_Filters_RX)
- Automatic gain control state machine (AGC) operating in manual control mode
- Programmable gain table (Gain Table)

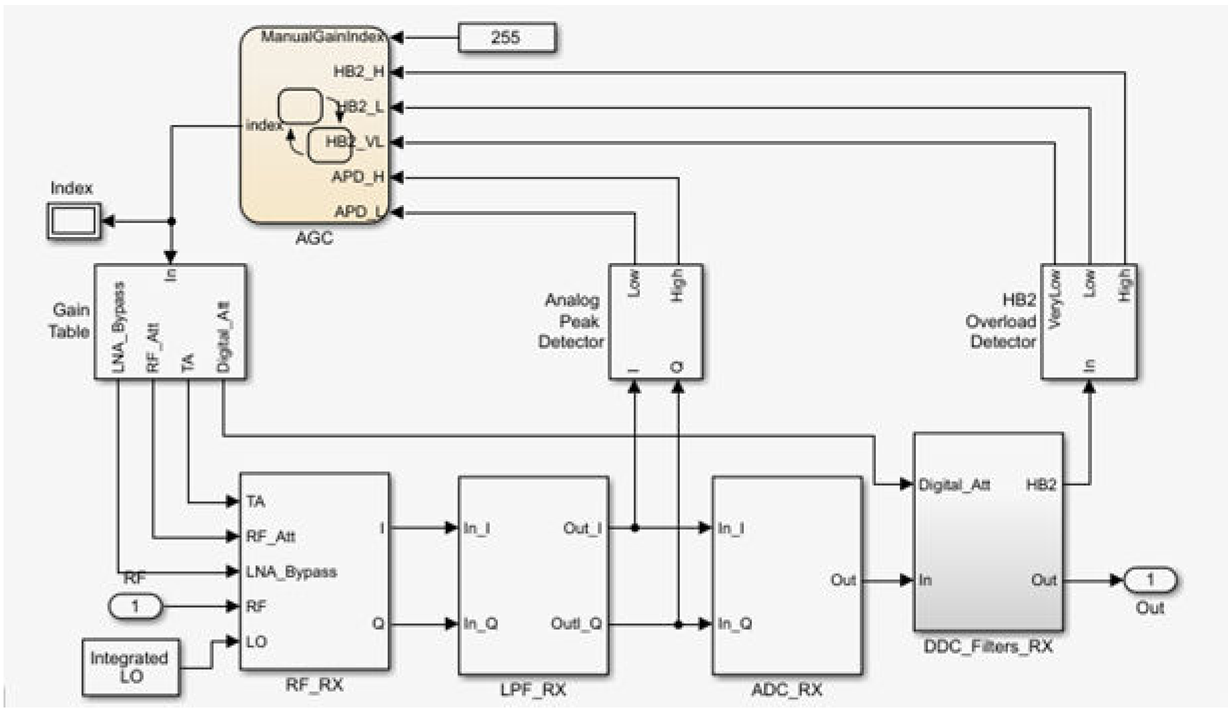
You can use the observer receiver model to simulate the following behaviors:

- Tunable internal attenuation
- Attenuation and carrier-frequency dependent noise figure

- Attenuation and carrier-frequency dependent output-referred third-order intercept (OIP3) and output-referred second-order intercept (OIP2)
- Attenuation and carrier-frequency dependent gain imbalance (to model finite image rejection)
- Attenuation and carrier-frequency dependent local oscillator (LO) carrier leakage
- Analog filters provide continuous time signal.
- ADC models a high-sampling rate third-order delta-sigma modulator.
- Digital down conversion digital filters converts the highly sampled signal at the output of the ADC to a lower baseband rate.
- Received signal strength indicator measures power at two stages, at the RF receiver output and after the first half-band filter
- Default gain table is read from the MATLAB file, `DefaultGainTables`. You can customize this file.

AD9371_SNF Analog Devices Sniffer Receiver

Model Description



The sniffer receiver model consists of three stages:

- RF receiver (RF_RX)
- Analog filters (LPF_RX)
- Analog to Digital converter (ADC_RX)
- Digital down-conversion filters (DDC_Filters_RX)
- Receiver signal strength indicator (RSSI): two power meters to detect the strength of the received signal at different section of the chain (Analog Peak Detector, and HB2 Overload Detector)

- Automatic gain control state machine (AGC) operating in manual control mode
- Programmable gain table (Gain Table)

You can use the AD9371 sniffer receiver model to simulate the following behaviors:

- Tunable internal attenuation
- Carrier-frequency dependent noise figure
- Attenuation and carrier-frequency dependent output-referred third-order intercept (OIP3) and output-referred second-order intercept (OIP2)
- Attenuation and carrier-frequency dependent gain imbalance (to model finite image rejection)
- Attenuation and carrier-frequency dependent local oscillator (LO) carrier leakage
- Analog filters provide continuous time signal.
- ADC models a high-sampling rate third-order delta-sigma modulator.
- Digital down conversion digital filters converts the highly sampled signal at the output of the ADC to a lower baseband rate.
- Received signal strength indicator measures power at two stages, at the RF receiver output and after the first half-band filter
- AGC changes the index of the gain table according to the flags of threshold crossing reported by the RSSI.
- Default gain table is read from the MATLAB file, `DefaultGainTables`. You can customize this file.

See Also
`simrfSupportPackages`

More About
- “AD9371 Testbenches” on page 5-15
- “AD9361 Models” on page 5-2

AD9371 Testbenches

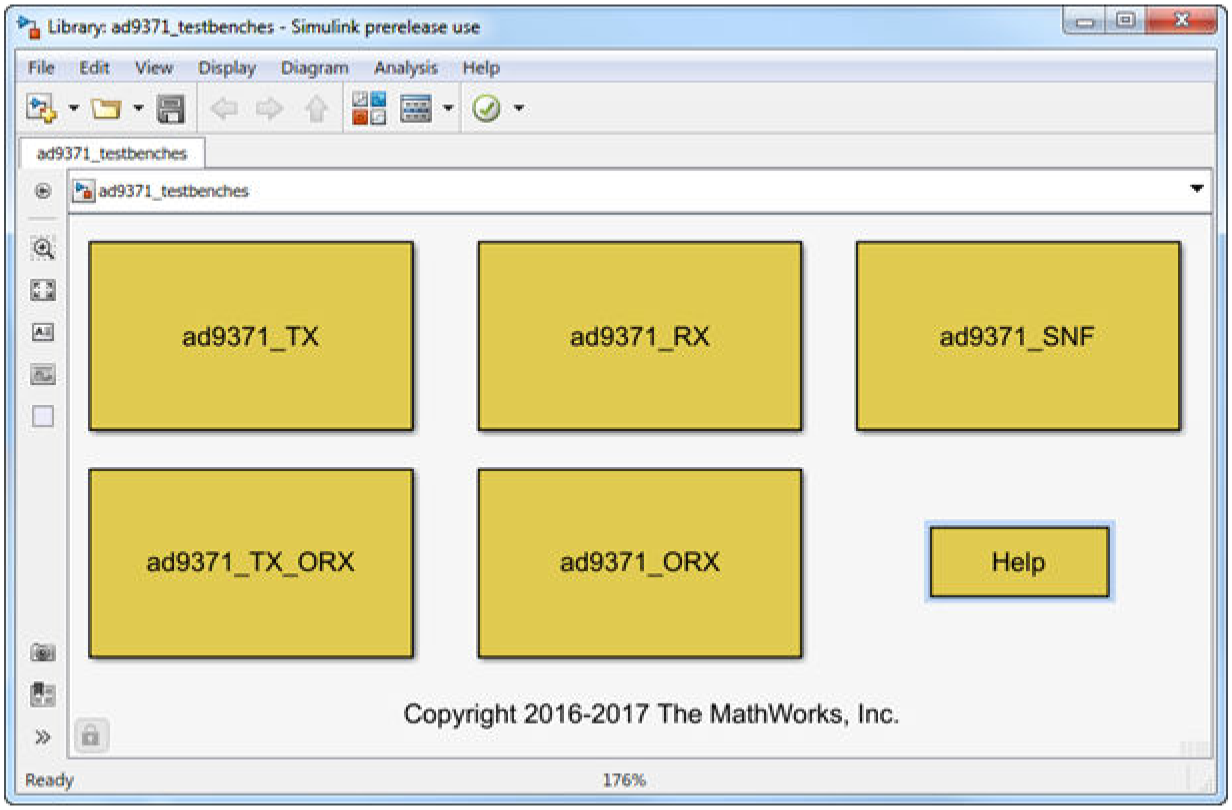
In this section...

You can use the AD9371 testbench models to analyze the functioning and values of Analog Devices AD9371 RF transmitter, receiver, observer, sniffer, or end-to-end designs.

Install Analog Devices RF Transceivers using `simrfSupportPackages`. You can open the testbench models using the Simulink library browser and opening RF Blockset Models for Analog Devices RF Transceivers, or by typing the following in the MATLAB command prompt:

```
open ad9371_testbenches
```

Click to open the AD9371 testbench model from the library:



Note You require these additional licenses to run this model:

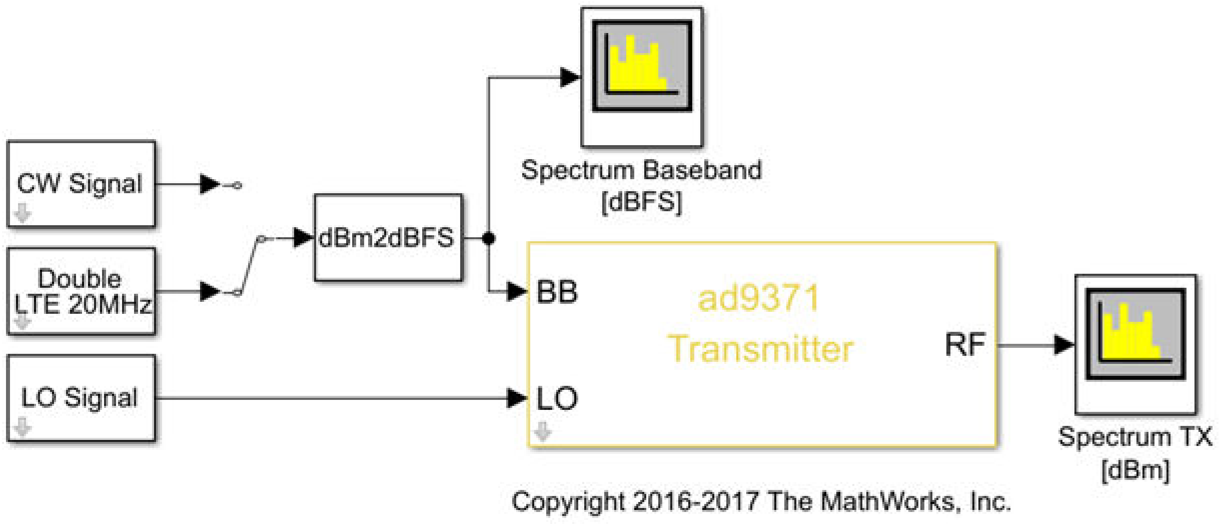
- Communications Toolbox

- Stateflow
- Fixed-Point Designer

Complete documentation on how to use the models is available with the software download.

```
open(ad93xx_getdoc)
```

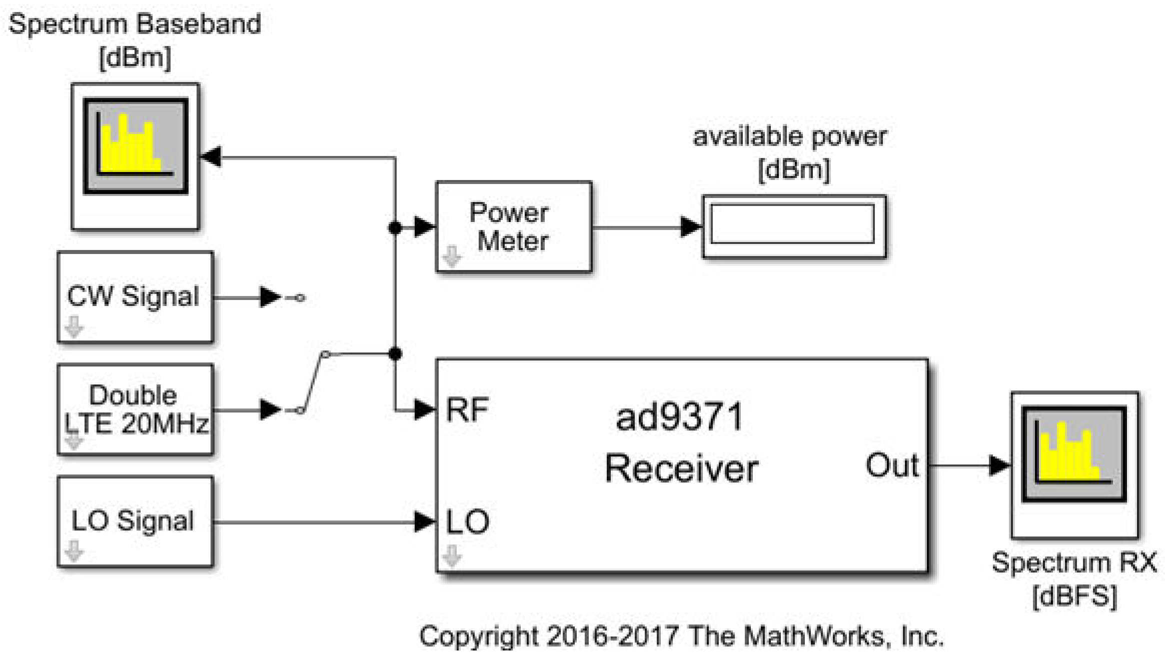
AD9371_TX Analog Devices Transmitter Testbench



The transmitter testbench consists of:

- CW and LTE Signal sources
- External local oscillator signal source
- AD9371 transmitter which is the device under test
- Spectrum analyzer and power meter for signal visualization

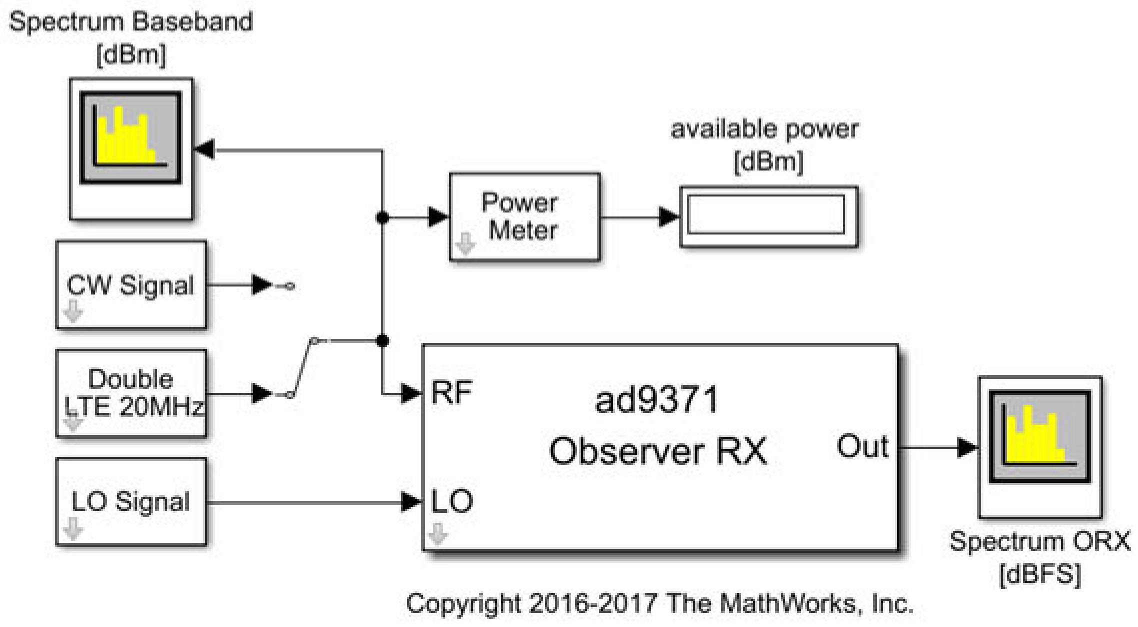
AD9371_RX Analog Devices Receiver Testbench



The receiver testbench consists of:

- CW and LTE 20 MHz Signal sources
- External local oscillator signal source
- AD9371 receiver which is the device under test
- Spectrum analyzer and power meter for signal visualization

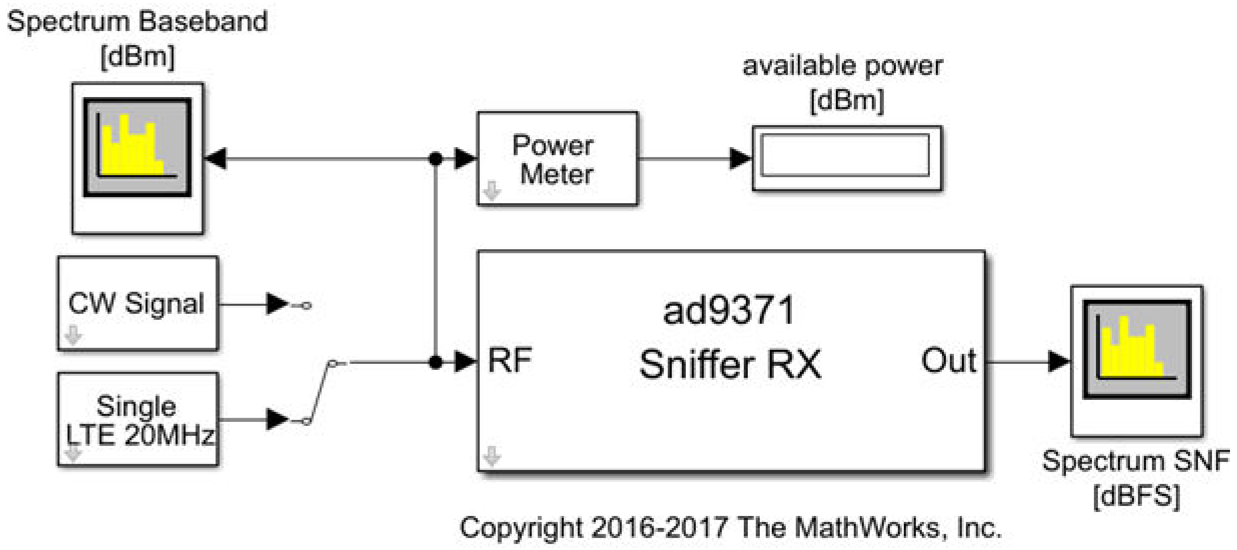
AD9371_ORX Analog Devices Observer Receiver Testbench



The observer receiver testbench consists of:

- CW and LTE 20 MHz Signal sources
- External local oscillator signal source
- AD9371 receiver which is the device under test
- Spectrum analyzer and power meter for signal visualization

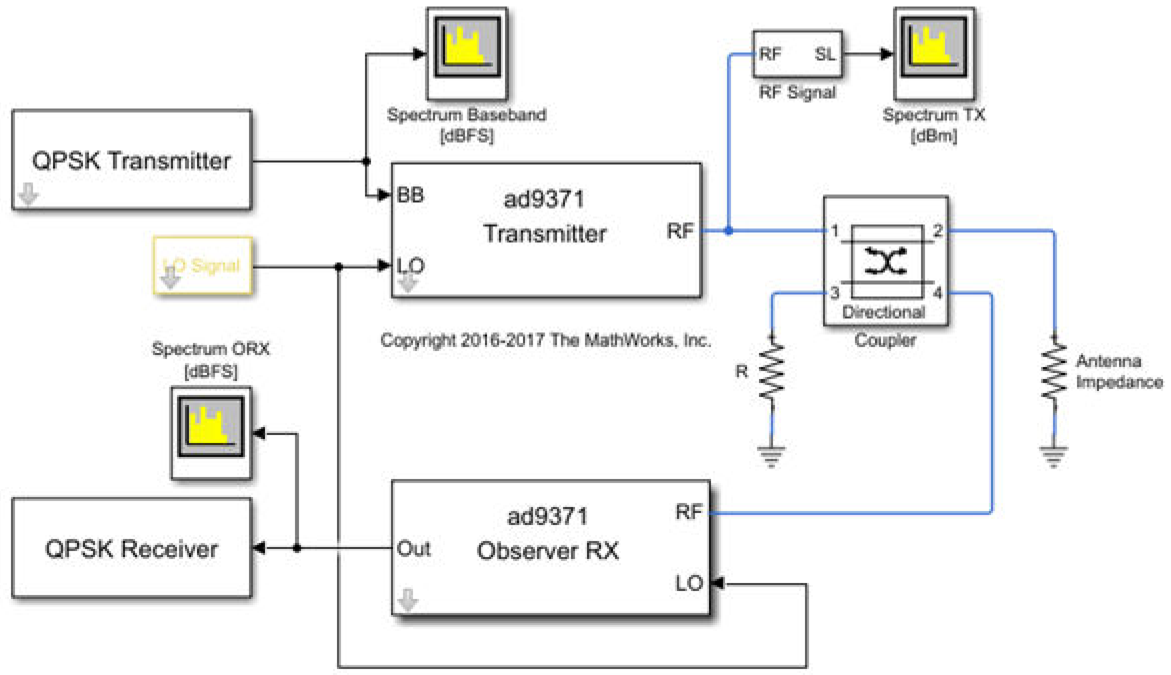
AD9371_SNF Analog Devices Sniffer Receiver Testbench



The sniffer receiver testbench consists of:

- CW and LTE 20 MHz Signal sources
- AD9371 sniffer receiver which is the device under test
- Spectrum analyzer and power meter for signal visualization

AD9371_TX_ORX Analog Devices Transmitter-Observer Testbench



The transmitter-observer testbench consists of:

- CW and LTE 20 MHz Signal sources
- External local oscillator signal source driving both transmitter and observer
- AD9371 transmitter and observer connected via directional coupler
- Spectrum analyzer and power meter for signal visualization

See Also

`simrfSupportPackages`

More About

- “AD9361 Testbenches” on page 5-6
- “AD9371 Models” on page 5-9

Equivalent Baseband

Model an RF System

Model RF Components

In this section...
“Add RF Blocks to a Model” on page 6-2
“Connect Model Blocks” on page 6-3

Add RF Blocks to a Model

You can include blocks from the RF Blockset Equivalent Baseband Physical and Mathematical libraries in a Simulink model. For more information on the libraries and the available RF blocks, see “RF Blockset Equivalent Baseband Libraries”.

This section contains the following topics:

- “Input Signal Requirements” on page 6-2
- “How to Add RF Blocks to a Model” on page 6-2

Input Signal Requirements

Most RF Blockset Equivalent Baseband blocks only support complex single-channel signals. The signals can be either sample-based or frame-based. The following blocks have this requirement:

- All Physical blocks
- Mathematical Amplifier and Mixer blocks

You can model the effect of these components on a multichannel signal as follows:

1. Use a Simulink Demux block to split the multichannel signal into single-channel signals.
2. Create duplicate RF models, with one model for each channel, and pass each single-channel signal into a separate model.
3. Use a Simulink Mux block multiplex the signals at the output of the RF models.

How to Add RF Blocks to a Model

To add RF blocks to a Simulink model:

1. Type `rflib` at the MATLAB prompt to open the RF Blockset Equivalent Baseband library.
2. Navigate to the desired library or sublibrary.
3. Drag instances of RF Blockset Equivalent Baseband blocks into the model window using the mouse.

Note You can also access RF Blockset Equivalent Baseband blocks and other Simulink blocks from the Simulink Library Browser window. Open this window by typing `simulink` at the MATLAB prompt. Add blocks to the model by dragging them from this window and dropping them into the model window.

Connect Model Blocks

You follow the same procedure for connecting RF Blockset Equivalent Baseband blocks as for connecting Simulink blocks: you click a port and drag the mouse to draw a line to another port on a different block.

You can only connect blocks that use the same type of signal. Physical library blocks use different types of signals than Mathematical library blocks, and are represented graphically by a different port style. Therefore, you can freely connect pairs of Mathematical modeling blocks. You can also freely connect pairs of Physical modeling blocks. However, you cannot directly connect Physical blocks to Mathematical blocks. Instead, you must use the Input Port and Output Port blocks to bridge them.

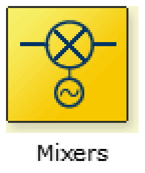
For more information on the Physical and Mathematical libraries, including how to open the libraries and a description of the available blocks, see “Overview of RF Blockset Equivalent Baseband Libraries”.

This section contains the following topics:

- “Connect Mathematical Blocks” on page 6-3
- “Connect Physical Blocks” on page 6-3
- “Bridge Physical and Mathematical Blocks” on page 6-4

Connect Mathematical Blocks

The Mathematical library blocks use the same input and output ports as standard Simulink blocks. These ports show the direction of the signal at the port, as shown in the following diagram.

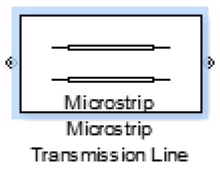


Similar to standard Simulink blocks, you draw lines between the ports of the Mathematical modeling blocks, called *signal lines*, to represent signals that are inputs to and outputs from the mathematical functions represented by the blocks. Therefore, you can connect Simulink, DSP System Toolbox, and RF Blockset Equivalent Baseband mathematical blocks by drawing signal lines between their ports.

You can connect a port to multiple ports by branching the signal line, or you can leave a port unconnected.

Connect Physical Blocks

The Physical library blocks have specialized *connector ports*. These ports only represent physical connections; they do not imply signal direction.



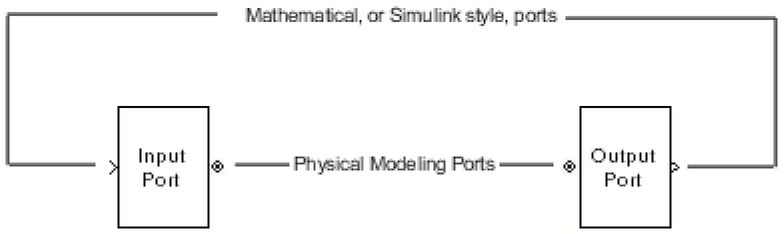
The lines you draw between the physical modeling blocks, called *connection lines*, represent physical connections among the block components. Connection lines appear as solid black when connected and as dashed red lines when either end is unconnected.

You can draw connection lines only between the connector ports of physical modeling blocks. You cannot branch these connection lines. You cannot leave connector ports unconnected.

Bridge Physical and Mathematical Blocks

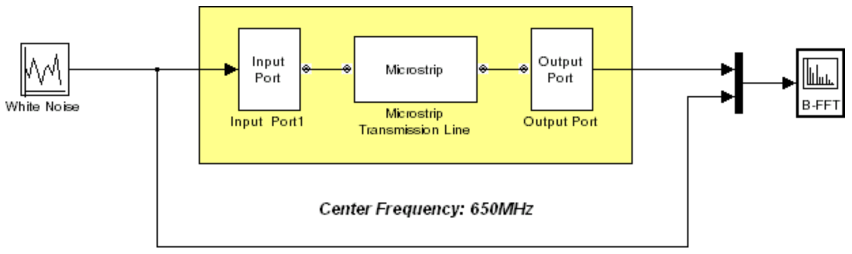
The blockset provides the Input Port and Output Port blocks to connect the physical and mathematical parts of the model. These blocks convert mathematical signals to and from the physical modeling environment.

The Input Port and Output Port blocks have one of each kind of connector port: a standard Simulink style input port and a physical modeling port. These ports are shown in the following figure:



The Input Port and Output Port blocks must bound a physical subsystem to connect it to the mathematical part of a model.

For example, a simple RF model of a coaxial transmission line might resemble the following figure.



The Microstrip Transmission Line block uses an Input Port block to get its white noise input from a Random Source block, and an Output Port block to pass its output to a Spectrum Scope block. The Random Source and Spectrum Scope blocks are from DSP System Toolbox library.

For information on how RF Blockset Equivalent Baseband software converts mathematical signals to and from the physical modeling environment, see “Convert to and from Simulink Signals” on page A-22.

See Also

Input Port | Microstrip Transmission Line | Output Port

More About

- “Simulate High Frequency Components”

Specify or Import Component Data

In this section...

Specify Parameter Values

There are two ways to set block parameter values:

- Using the GUI — Enter information in the block dialog boxes, which open when you double-click a block in the Simulink window.
- Using commands — Use the Simulink `set_param` and `get_param` commands to set and get parameter values of the blocks, respectively. For more information on these commands, see the `set_param` and `get_param` reference pages.

Supported File Types for Importing Data

The blockset also lets you import the following types of data files:

- Industry-standard file formats — Touchstone S2P, Y2P, Z2P, and H2P formats specify the network parameters and noise information for measured and simulated data.

  For more information on Touchstone files, see `https://ibis.org/connector/touchstone_spec11.pdf`.
- Agilent® P2D file format — Specifies amplifier and mixer large-signal, power-dependent network parameters, noise data, and intermodulation tables for several operating conditions, such as temperature and bias values.

  The P2D file format lets you import system-level verification models of amplifiers and mixers.
- Agilent S2D file format — Specifies amplifier and mixer network parameters with gain compression, power-dependent S_{21} parameters, noise data, and intermodulation tables for several operating conditions.

  The S2D file format lets you import system-level verification models of amplifiers and mixers.
- MathWorks amplifier (AMP) file format — Specifies amplifier network parameters, power data, noise data, and third-order intercept point

  For more information about `.amp` files, see "AMP File Data Sections".
- MATLAB circuits — RF Toolbox™ circuit objects in the MATLAB workspace specify network parameters, noise data, and third-order intercept point information of circuits with different topologies.

  For more information about RF circuit objects, see "RF Circuit Objects".

Import Data Files into RF Blocks

The blockset lets you import industry-standard data files, Agilent P2D and S2D files, and MathWorks AMP files into specific blocks to simulate the behavior of measured components in the Simulink modeling environment.

This section contains the following topics:

- "Blocks Used to Import Data" on page 6-6
- "How to Import Data Files" on page 6-6

Blocks Used to Import Data

Three blocks in the Physical library accept data from a file. The following table lists the blocks and any corresponding data format that each supports.

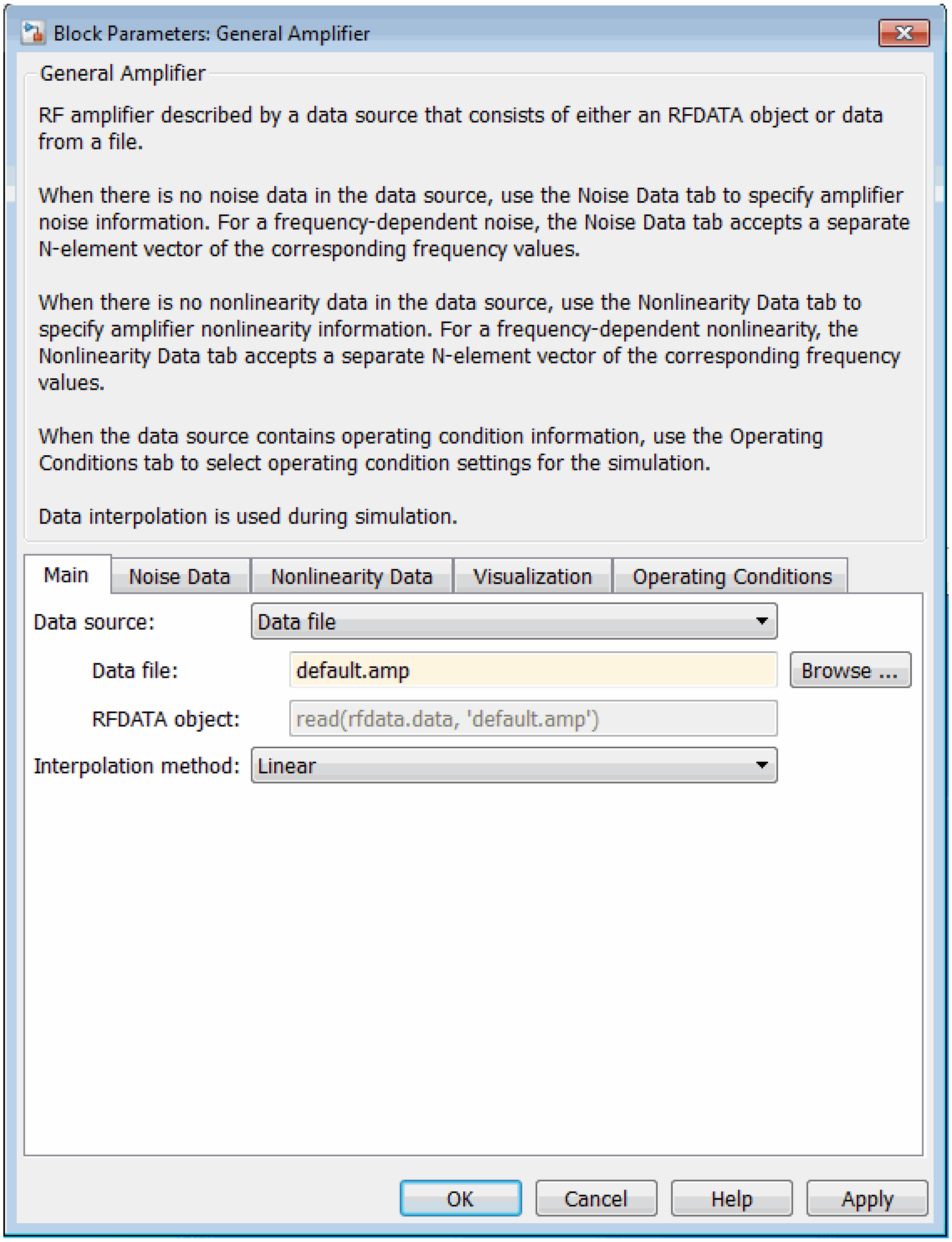
Block	Description	Supported Format(s)
General Amplifier	Generic amplifier	Touchstone, AMP, P2D, S2D
General Mixer	Generic mixer	Touchstone, AMP, P2D, S2D
General Passive Network	Generic passive component	Touchstone

How to Import Data Files

To import a data file:

1. Choose the block that best represents your component from the list of blocks that accept file data shown in "Blocks Used to Import Data" on page 6-6.
2. Open the Physical library, and navigate to the sublibrary that contains the block.
3. Click and drag the block into your Simulink model.
4. In the block dialog box, enter the name of your data file for the **Data file** parameter. The file name must include the extension. If the file is not in your MATLAB path, specify the full path to the file or use the **Browse** button to find the file.

Note The **Data file** parameter is only enabled when the **Data source** parameter is set to `Data file`. This is the default setting and it means the block data comes from a file.


Block Parameters: General Amplifier

General Amplifier

RF amplifier described by a data source that consists of either an RFDATA object or data from a file.

When there is no noise data in the data source, use the Noise Data tab to specify amplifier noise information. For a frequency-dependent noise, the Noise Data tab accepts a separate N-element vector of the corresponding frequency values.

When there is no nonlinearity data in the data source, use the Nonlinearity Data tab to specify amplifier nonlinearity information. For a frequency-dependent nonlinearity, the Nonlinearity Data tab accepts a separate N-element vector of the corresponding frequency values.

When the data source contains operating condition information, use the Operating Conditions tab to select operating condition settings for the simulation.

Data interpolation is used during simulation.

Main | Noise Data | Nonlinearity Data | Visualization | Operating Conditions

Data source: Data file

Data file: default.amp

RFDATA object: read(rfdata.data, 'default.amp')

Interpolation method: Linear

The following section shows an example of this procedure.

Example — Import a Touchstone Data File into an RF Model

In this example, you model the frequency response of a passive component using data from a Touchstone file, `defaultbandpass.s2p`.

You use a model from one of the RF Blockset Equivalent Baseband examples to perform the following tasks:

- "Import Data into a General Passive Network Block" on page 6-8
- "Validate the Passive Component" on page 6-9

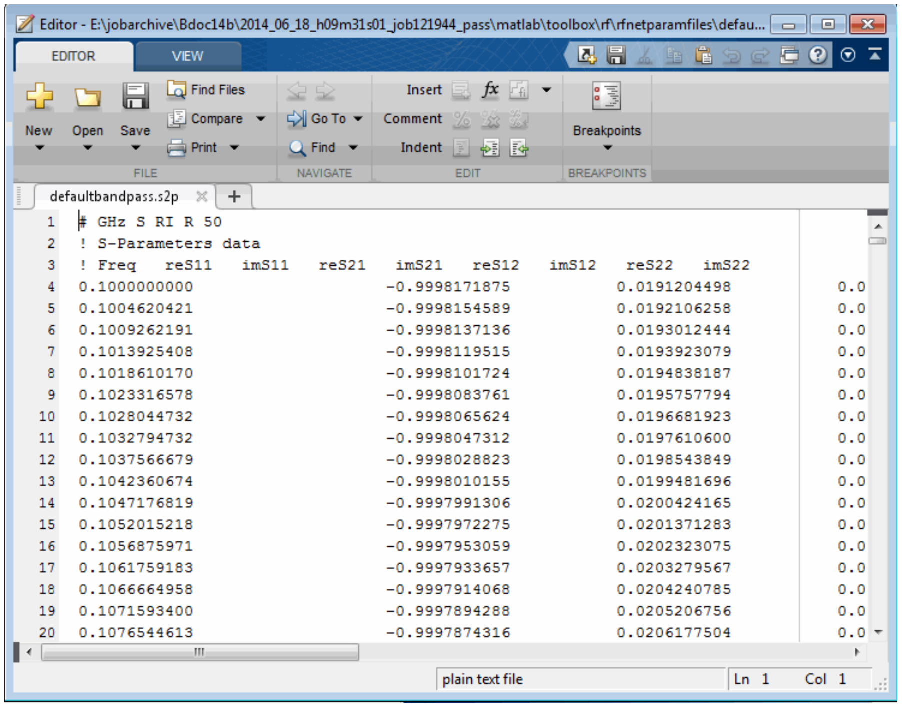
Import Data into a General Passive Network Block

In this part of the example, you inspect the `defaultbandpass.s2p` file and import data into the RF model using the General Passive Network block.

1 Type the following at the MATLAB prompt to open the `defaultbandpass.s2p` file:

```
edit defaultbandpass.s2p
```

The following figure shows a portion of the `.s2p` file.



```
# GHz S RI R 50
! S-Parameters data
! Freq    reS11    imS11    reS21    imS21    reS12    imS12    reS22    imS22
0.1000000000                    -0.9998171875          0.0191204498           0.0
0.1004620421                    -0.9998154589          0.0192106258           0.0
0.1009262191                    -0.9998137136          0.0193012444           0.0
0.1013925408                    -0.9998119515          0.0193923079           0.0
0.1018610170                    -0.9998101724          0.0194838187           0.0
0.1023316578                    -0.9998083761          0.0195757794           0.0
0.1028044732                    -0.9998065624          0.0196681923           0.0
0.1032794732                    -0.9998047312          0.0197610600           0.0
0.1037566679                    -0.9998028823          0.0198543849           0.0
0.1042360674                    -0.9998010155          0.0199481696           0.0
0.1047176819                    -0.9997991306          0.0200424165           0.0
0.1052015218                    -0.9997972275          0.0201371283           0.0
0.1056875971                    -0.9997953059          0.0202323075           0.0
0.1061759183                    -0.9997933657          0.0203279567           0.0
0.1066664958                    -0.9997914068          0.0204240785           0.0
0.1071593400                    -0.9997894288          0.0205206756           0.0
0.1076544613                    -0.9997874316          0.0206177504           0.0
```

The option line

```
# GHz S RI R 50
```

specifies the following information about the contents of the data file:

- GHz — Frequency units.
- S — Network parameters are S-parameters.

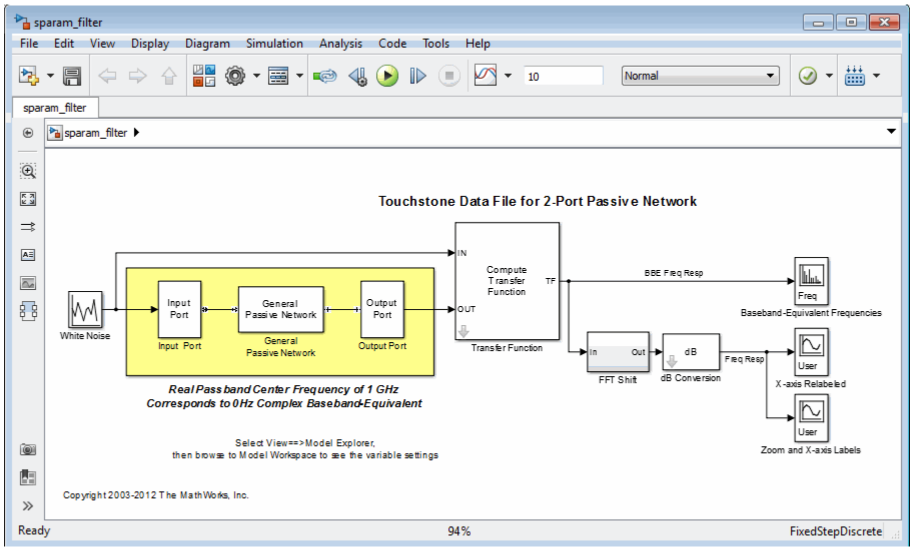
- RI — Network parameters are specified as the real and imaginary parts.
- R 50 — Reference impedance is 50 ohms.

For more information about the Touchstone specification, including the option line, see `https://ibis.org/connector/touchstone_spec11.pdf`.

2 At the MATLAB prompt, type

```
sparam_filter
```

This command opens the RF Blockset Equivalent Baseband example called “Touchstone Data File for 2-Port Bandpass Filter,” as shown in the following figure.


sparam_filter
File Edit View Display Diagram Simulation Analysis Code Tools Help
10
Normal
Touchstone Data File for 2-Port Passive Network
White Noise
Input Port
General Passive Network
Output Port
Real Passband Center Frequency of 1 GHz
Corresponds to 0Hz Complex Baseband-Equivalent
IN
OUT
Compute Transfer Function
TF
Transfer Function
BBE Freq Resp
Freq
Baseband-Equivalent Frequencies
In Out
FFT Shift
dB
dB Conversion
Freq Resp
User
X-axis Relabeled
User
Zoom and X-axis Labels
Select View==>Model Explorer,
then browse to Model Workspace to see the variable settings
Copyright 2003-2012 The MathWorks, Inc.
Ready
94%
FixedStepDiscrete

3 Double-click the General Passive Network block to display its parameters.

The **Data source** parameter is set to `Data file`, so the **Data file** parameter specifies the data file to import. The **Data file** parameter is set to `defaultbandpass.s2p`. The block uses this data with the other block parameters during simulation.

Note When the imported file contains data that is measured at frequencies other than the modeling frequencies, use the **Interpolation method** parameter to specify how the block determines the data values at the modeling frequencies. For more information, see “Determine Modeling Frequencies” on page A-3 and “Map Network Parameters to Modeling Frequencies” on page A-4.

Validate the Passive Component

In this part of the example, you plot the network parameters of the General Passive Network block to validate the data you imported in “Import Data into a General Passive Network Block” on page 6-8.

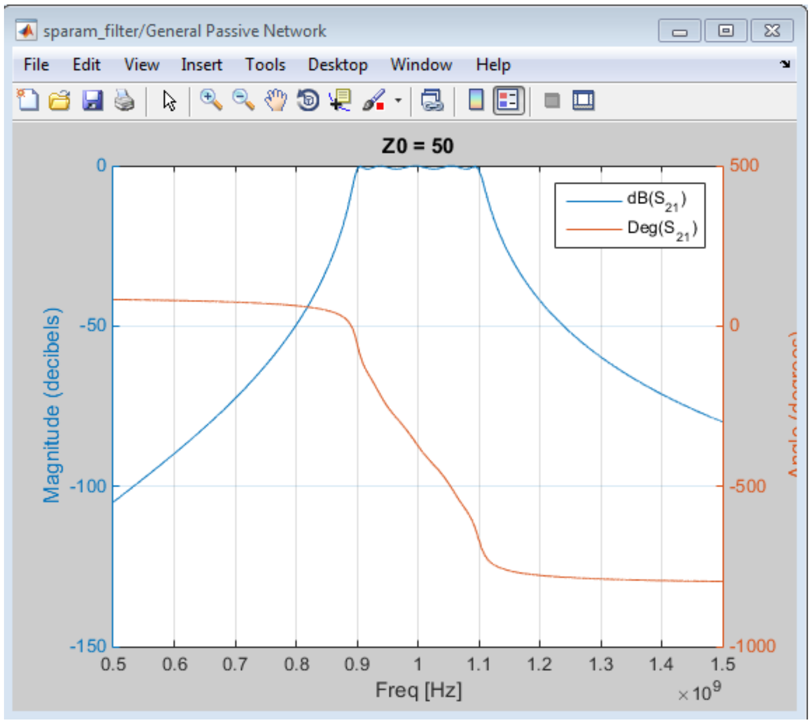
1 Open the General Passive Network block dialog box, and select the **Visualization** tab.

2 Set the **Source of frequency data** parameter to `User-specified`.

3 Set the **Frequency data (Hz)** parameter to `[0.5e9:0.1e6:1.5e9]`.

4 Click **Plot**.

These actions create a plot of the magnitude and phase of S_{21} as a function of frequency.



S_{21} versus Frequency for the Imported Data

Import Circuits from the MATLAB Workspace

You can only connect Physical library blocks in cascade. However, the blockset works with RF Toolbox software to let you include additional circuit topologies in an RF model. To model circuit topologies that contain other types of connections, you must define a circuit in the MATLAB workspace and import it into an RF model.

To import a circuit from the MATLAB workspace:

1. Define the circuit object in the MATLAB workspace using the RF Toolbox functions.

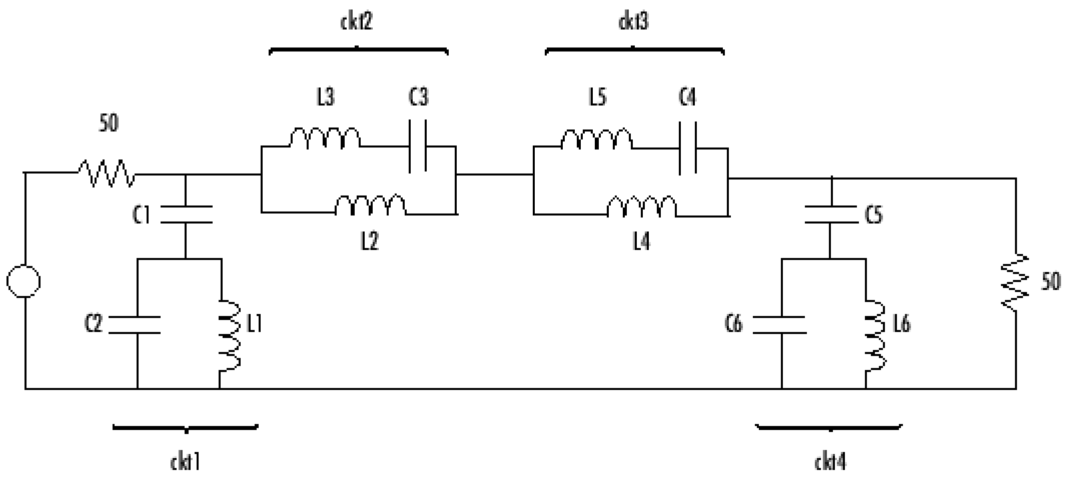
   For more information about RF circuit objects, see the RF Toolbox documentation for “RF Circuit Objects”.
2. Add a General Circuit Element block to your RF model from the Black Box Elements sublibrary of the Physical library. For information on how to open this library, see “Open RF Blockset Equivalent Baseband Libraries”.
3. Enter the circuit object name in the **RFCKT object** parameter in the General Circuit Element block dialog box.

This procedure is illustrated by example in the following section.

Example — Import a Bandstop Filter into an RF Model

In this example, you simulate the frequency response of a filter that you model using circuit objects from the MATLAB workspace.

The filter in this example is the 50-ohm bandstop filter shown in the following figure.



Bandstop Filter Diagram

You represent the filter using four circuit objects that correspond to the four parts of the filter, `ckt1`, `ckt2`, `ckt3`, and `ckt4` in the diagram. You use an input signal with random, complex input values that have a Gaussian distribution to stimulate the filter. The scope block displays the output signal.

This example illustrates how to perform the following tasks:

- "Create Circuit Objects in the MATLAB Workspace" on page 6-11
- "Build the Model" on page 6-12
- "Specify and Import Component Data" on page 6-13
- "Run the Simulation and Plot the Results" on page 6-14

Create Circuit Objects in the MATLAB Workspace

In this part of the example, you define MATLAB variables to represent the physical properties of the filter shown in the previous figure, "Bandstop Filter Diagram" on page 6-11, and use functions from RF Toolbox software to create RF circuit objects that model the filter components.

1 Type the following at the MATLAB prompt to define the filter's capacitance and inductance values in the MATLAB workspace:

```
C1 = 1.734e-12;
C2 = 4.394e-12;
C3 = 7.079e-12;
C4 = 7.532e-12;
C5 = 1.734e-12;
C6 = 4.394e-12;
L1 = 25.70e-9;
L2 = 3.760e-9;
L3 = 17.97e-9;
L4 = 3.775e-9;
L5 = 17.63e-9;
L6 = 25.70e-9;
```

2 Type the following at the MATLAB prompt to create RF circuit objects that model the components labeled `ckt1`, `ckt2`, `ckt3`, and `ckt4` in the circuit diagram:

```
ckt1 = ...
    rfckt.series('Ckts',{rfckt.shuntrlc('C',C1),...
    rfckt.shuntrlc('L',L1,'C',C2)});
ckt2 = ...
    rfckt.parallel('Ckts',{rfckt.seriesrlc('L',L2),...
    rfckt.seriesrlc('L',L3,'C',C3)});
ckt3 = ...
    rfckt.parallel('Ckts',{rfckt.seriesrlc('L',L4),...
    rfckt.seriesrlc('L',L5,'C',C4)});
ckt4 = ...
    rfckt.series('Ckts',{rfckt.shuntrlc('C',C5),...
    rfckt.shuntrlc('L',L6,'C',C6)});
```

For more information about the RF Toolbox objects used in this example, see the `rfckt.series`, `rfckt.parallel`, `rfckt.seriesrlc`, and `rfckt.shuntrlc` object reference pages in the RF Toolbox documentation.

Build the Model

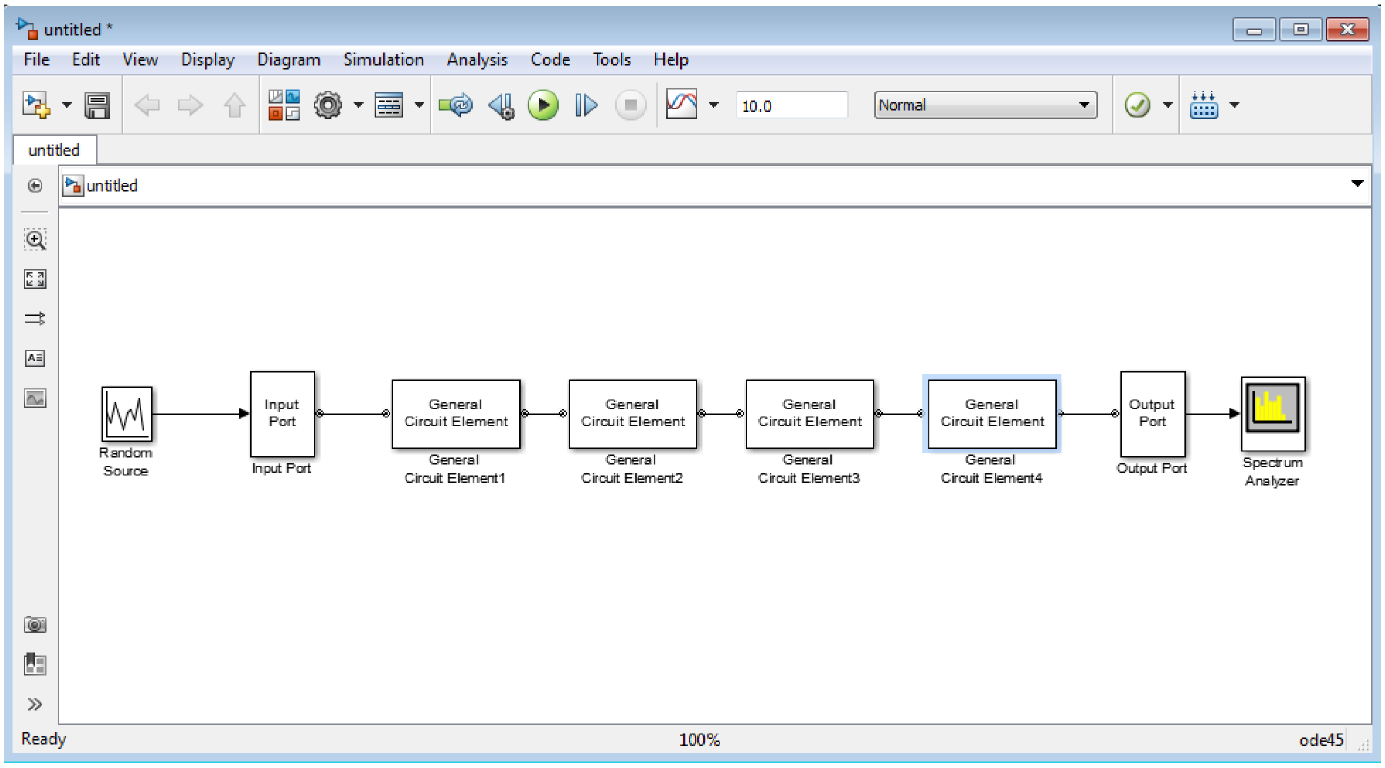
In this portion of the example, you create a Simulink model. For more information about adding and connecting components, see “Model RF Components” on page 6-2.

1 Create a new model.
2 Add to the model the blocks shown in the following table. The Library column of the table specifies the hierarchical path to each block.

Block	Library	Quantity
Random Source	**DSP System Toolbox > Sources**	1
Input Port	**RF Blockset > Equivalent Baseband > Input/Output Ports**	1
General Circuit Element	**RF Blockset > Equivalent Baseband > Black Box Elements**	4
Output Port	**RF Blockset > Equivalent Baseband > Input/Output Ports**	1
Spectrum Analyzer	**DSP System Toolbox > Sinks**	1

3 Connect the blocks as shown in the following figure.

Change the names of your General Circuit Element blocks to match those in the figure by double-clicking the text below the block and typing a new name.



Specify and Import Component Data

In this portion of the example, you specify block parameters. To open the parameter dialog box for each block, double-click the block.

1 In the Random Source block dialog box:

- Set the **Source type** parameter to `Gaussian`.
- Set the **Sample time** parameter to `1/100e6`.
- Set the **Samples per frame** parameter to `256`.
- Set the **Complexity** parameter to `Complex`.

Selecting these settings creates an input signal with random, complex input values that have a Gaussian distribution.

2 In the Input Port block dialog box:

- Set the **Treat input Simulink signal as** parameter to `Incident power wave`.
- Set the **Finite impulse response filter length** parameter to `256`.
- Set the **Center frequency (Hz)** parameter to `400e6`.
- Set the **Sample time** parameter to `1/100e6`.
- Clear the **Add noise** check box.

Selecting these settings defines the physical characteristics and modeling bandwidth of the filter.

3 Set the parameters of the General Circuit Element blocks as follows:

- In the General Circuit Element1 block dialog box, set the **RFCKT object** parameter to `ckt1`.
- In the General Circuit Element2 block dialog box, set the **RFCKT object** parameter to `ckt2`.
- In the General Circuit Element3 block dialog box, set the **RFCKT object** parameter to `ckt3`.
- In the General Circuit Element4 block dialog box, set the **RFCKT object** parameter to `ckt4`.

Selecting these settings imports the circuit objects that model the filter components into the model.

4 In the Output Port block dialog box, set the **Load impedance** parameter to `50`.

5 Set the Spectrum Analyzer block parameters as follows:

- In the **View** tab, under `Spectrum Settings` , set the **Averages** under `Trace options` to `100`.

  This parameter establishes the number of spectra that the scope averages to produce the displayed signal. You use a value of 100 because the input signal is random and you want to display the average filter response over a large number of input values.
- In the **View** tab, under `Spectrum Settings` , set the **Units** under `Trace options` to `dBm/Hertz`.
- In the **View** tab, under `Configuration Properties`, set the **Minimum Y-limit** parameter to `-75` and the **Maximum Y-limit** parameter to `-45`.

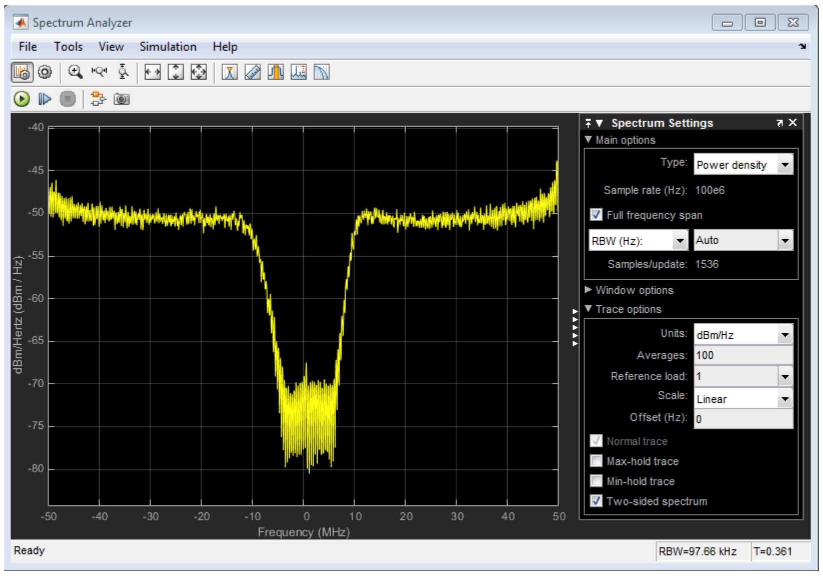
  These values set the range of *x*- and *y*-values on the display such that the entire signal is visible when you run the simulation.
- Set the **Y-axis label** parameter to `dBm/Hertz`.

Run the Simulation and Plot the Results

In this part of the example, you run the simulation and examine the frequency response of the filter.

Click **Run** in the model window to start the simulation.

The Spectrum Scope window appears automatically and displays the following plot, which shows the frequency response of the filter.



Spectrum Settings

Main options

Type: Power density

Sample rate (Hz): 100e6

✓ Full frequency span

RBW (Hz): Auto

Samples/update: 1536

Window options

Trace options

Units: dBm/Hz

Averages: 100

Reference load: 1

Scale: Linear

Offset (Hz): 0

✓ Normal trace

Max-hold trace

Min-hold trace

✓ Two-sided spectrum

Ready | RBW=97.66 kHz | T=0.361

Frequency Response of Bandstop Filter

The Spectrum Scope block displays the frequency response at the shifted (baseband-equivalent) frequencies, not at the selected passband frequencies. You can relabel the *x*-axis of the Spectrum Scope window to display the passband signal by entering the **Center frequency** parameter value of `400e6` (from the Input Port block) for the **Frequency display offset (Hz)** parameter in the **Axis Properties** tab of the Spectrum Scope block. For more information on complex-baseband modeling, see “Create Complex Baseband-Equivalent Model” on page A-9.

References

Geffe, P.R., “Novel designs for elliptic bandstop filters,” *RF Design*, February 1999.

See Also

Configuration | Input Port | Output Port

More About

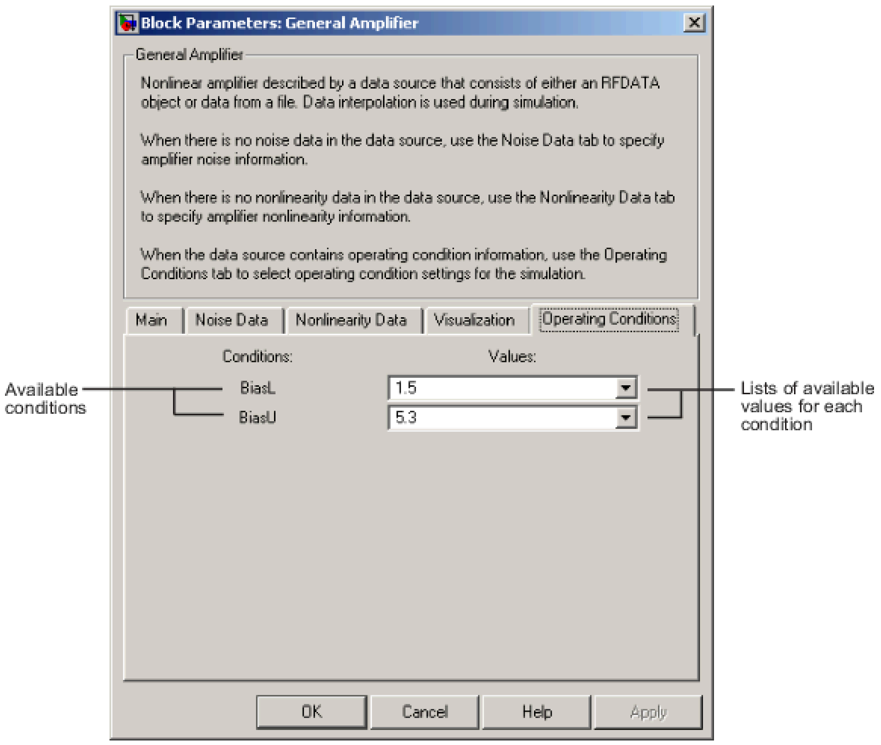
- “Create Complex Baseband-Equivalent Model” on page A-9

Specify Operating Conditions

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

When you import component data from a `.p2d` or `.s2d` file into a General Amplifier or General Mixer block, the block contains parameter values for several operating conditions. The available conditions depend on the data in the file. By default, the blockset defines the object behavior using the property values that correspond to the operating conditions that appear first in the file. To use other property values, you must select a different operating condition in the block dialog box.

If the block 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.


Block Parameters: General Amplifier

General Amplifier

Nonlinear amplifier described by a data source that consists of either an RFDATA object or data from a file. Data interpolation is used during simulation.

When there is no noise data in the data source, use the Noise Data tab to specify amplifier noise information.

When there is no nonlinearity data in the data source, use the Nonlinearity Data tab to specify amplifier nonlinearity information.

When the data source contains operating condition information, use the Operating Conditions tab to select operating condition settings for the simulation.

Conditions:	Values:
BiasL	1.5
BiasU	5.3

Block Dialog Box Showing Operating Conditions

To specify the operating condition values for a simulation:

1. Double-click the block to open the block dialog box.
2. Select the **Operating Conditions** tab.
3. In the **Conditions** column, find the condition to specify. Select the corresponding pull-down list in the **Values** column, and choose the desired operating condition value.

Repeat the preceding step as needed to specify the desired operating condition values.

Model Nonlinearity

In this section...

Amplifier and Mixer Nonlinearity Specifications

You define nonlinearity for the physical amplifier and mixer blocks at one or more frequency points through one of the following specifications:

- Power data, consisting of output power as a function of input power, imported into the block.
- Third-order intercept data, with or without power parameters, in the block dialog box. The available power parameters are gain compression power (defined as the ratio of output power to input power at small input power) and output saturation power.

The following table summarizes the nonlinearity specification options for each type of physical amplifier and mixer block.

Block	Nonlinearity Specification
General Amplifier	You can choose either of the following specifications: Power data (using a P2D, S2D, or AMP data file) or Third-order intercept data or one or more power parameters, in the block dialog box.
S-Parameters Amplifier Y-Parameters Amplifier Z-Parameters Amplifier	Third-order intercept data or one or more power parameters, in the block dialog box.
General Mixer	You can choose either of the following specifications: Power data (using a P2D, S2D, or AMP data file) or Third-order intercept data or one or more power parameters, in the block dialog box.
S-Parameters Mixer Y-Parameters Mixer Z-Parameters Mixer	Third-order intercept data or one or more power parameters, in the block dialog box.

Add Nonlinearity to Your System

To simulate the nonlinearity of an amplifier or mixer, you must specify or import nonlinearity data at one or more frequency points into the block.

The method you use to add nonlinearity data to a block depends on whether you specify the data manually or import the data into a block.

The following table provides instructions for adding nonlinearity data.

Nonlinearity Specification	Instructions
IP3	In the **Nonlinearity Data** tab of the block dialog box: • Set the **IP3 type** parameter to `IIP3` or `OIP3`. • Enter input third-order intercept values at one or more frequency points in the **IP3 (dBm)** parameter. • Enter corresponding frequency values in the **Frequency (Hz)** parameter.
Power parameters	Enter the gain compression power in the **1 dB gain compression power (dBm)** parameter or the saturation power in the **Output saturation power (dBm)** parameter. If you choose a scalar value for the **Frequency (Hz)** parameter, then you must also use scalar values for the power parameters. If you choose a vector value for the **Frequency (Hz)** parameter, then you can use either scalar or vector values for the power parameters.
Power data (from a file)	Import file data that includes power information into the **Data file** or **RFCKT object** parameter of the General Amplifier or General Mixer block.

Note If you import file data with no power information into a General Amplifier or General Mixer block, the **Nonlinearity Data** tab lets you add nonlinearity data manually in the block dialog box.

For information on how the blockset simulates nonlinearity data of an amplifier or mixer, see the block reference page.

See Also

General Amplifier | General Mixer | S-Parameters Amplifier | S-Parameters Mixer | Y-Parameters Amplifier | Y-Parameters Mixer | Z-Parameters Amplifier | Z-Parameters Mixer

More About

- "User-Defined Nonlinear Amplifier Model" on page 9-148

Model Noise

In this section...

Amplifier and Mixer Noise Specifications

You only need to specify noise information for the physical amplifier and mixer blocks that generate noise other than resistor noise. For the other blocks, the blockset calculates the noise automatically based on the resistor values.

You define noise for the physical amplifier and mixer blocks through one of the following specifications:

- 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.
- Frequency-independent 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 following table summarizes the noise specification options for each type of physical amplifier and mixer block.

Block	Noise Specification
General Amplifier	Spot noise data (using a Touchstone, P2D, S2D, or AMP data file) **OR** Spot noise data, noise figure value, noise factor value, noise temperature value, `rfdata.noise`, or `rfdata.nf` object in the block dialog box
S-Parameters Amplifier Y-Parameters Amplifier Z-Parameters Amplifier	Spot noise data, noise figure value, noise factor value, noise temperature value, `rfdata.noise`, or `rfdata.nf` object in the block dialog box
General Mixer	Spot noise data (using a Touchstone, P2D, S2D, or AMP data file) **OR** Spot noise data, noise figure value, noise factor value, noise temperature value, `rfdata.noise`, or `rfdata.nf` object in the block dialog box

Block	Noise Specification
S-Parameters Mixer Y-Parameters Mixer Z-Parameters Mixer	Spot noise data, noise figure value, noise factor value, noise temperature value, `rfdata.noise`, or `rfdata.nf` object in the block dialog box

Add Noise to Your System

To simulate the noise of a physical subsystem, you perform the following tasks:

- "Specify or Import Noise Data" on page 6-20
- "Add Noise to the Simulation" on page 6-21

Specify or Import Noise Data

The method you use to add noise data to a block depends on whether you are specifying noise data manually or importing spot-noise data.

The following table provides instructions for adding noise data.

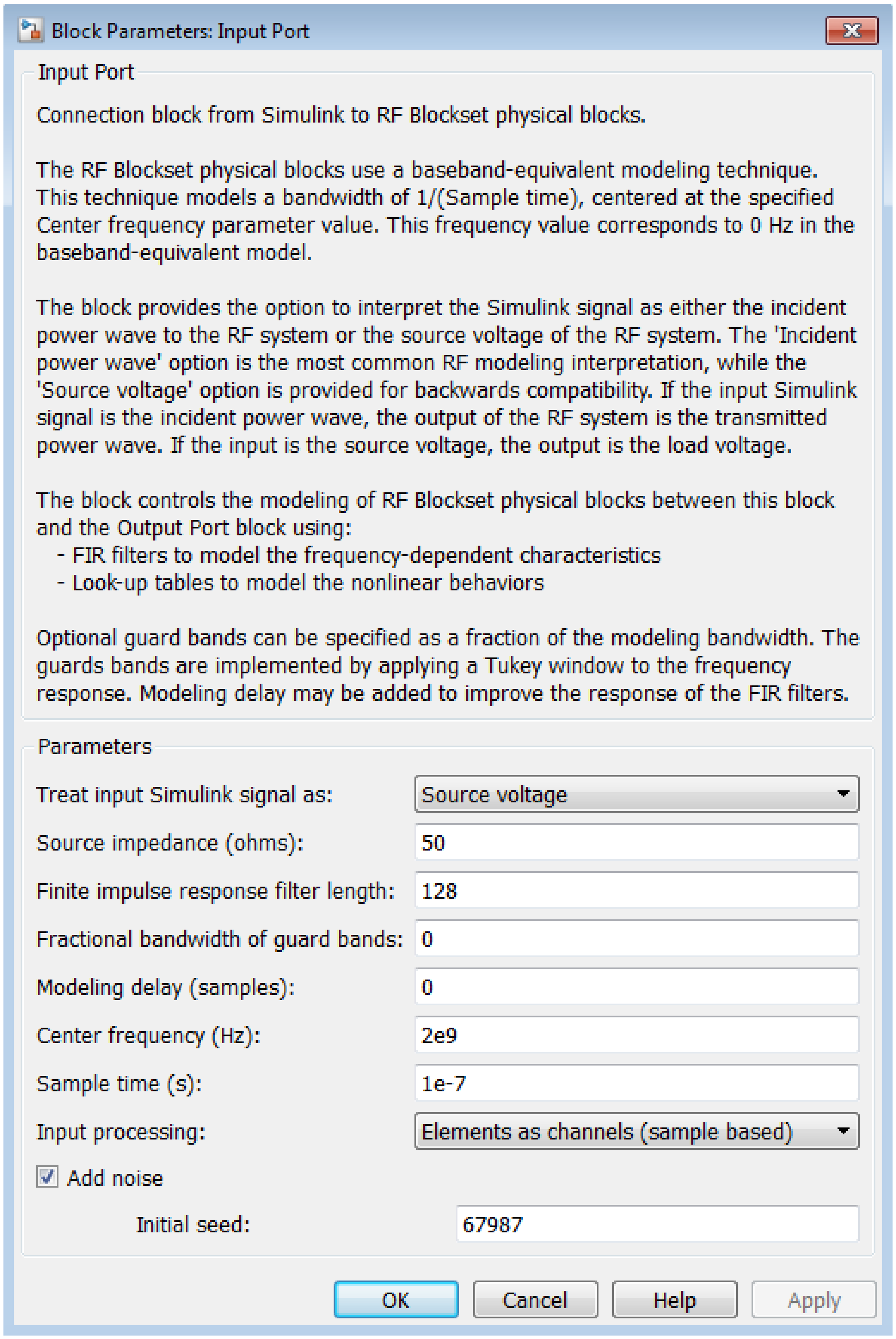
Noise Specification	Instructions
Frequency-independent noise figure	In the **Noise Data** tab of the block dialog box, set the **Noise type** parameter to `Noise figure`, and enter the noise figure value in the **Noise figure (dB)** parameter.
Frequency-dependent noise figure	In the **Noise Data** tab of the block dialog box, set the **Noise type** parameter to `Noise figure`, and enter the name of the `rfdata.nf` object in the **Noise figure (dB)** parameter.
Noise factor	In the **Noise Data** tab of the block dialog box, set the **Noise type** parameter to `Noise factor`, and enter the noise factor value in the **Noise factor** parameter.
Noise temperature	In the **Noise Data** tab of the block dialog box, set the **Noise type** parameter to `Noise temperature`, and enter the noise temperature value in the **Noise temperature (K)** parameter.
Spot noise data (in a block dialog box)	In the **Noise Data** tab of the block dialog box, set the **Noise type** parameter to `Spot noise data`. Enter the spot noise information in the **Minimum noise figure (dB)**, **Optimal reflection coefficient**, and **Equivalent normalized noise resistance** parameters.
Spot noise data (from a data object)	In the **Noise Data** tab of the block dialog box, set the **Noise type** parameter to `Noise figure` and enter the name of the `rfdata.noise` object in the **Noise figure (dB)** parameter.

Noise Specification	Instructions
Spot noise data (from a file)	Import file data that includes noise information into the **Data file** or **RFCKT object** parameter of the General Amplifier or General Mixer block.

Note If you import file data with no noise information into a General Amplifier or General Mixer block, the **Noise Data** tab lets you add noise data manually in the block dialog box.

Add Noise to the Simulation

To include noise in the simulation, you must select the **Add noise** check box on the Input Port block dialog box. This check box is selected by default.



Block Parameters: Input Port

Input Port

Connection block from Simulink to RF Blockset physical blocks.

The RF Blockset physical blocks use a baseband-equivalent modeling technique. This technique models a bandwidth of 1/(Sample time), centered at the specified Center frequency parameter value. This frequency value corresponds to 0 Hz in the baseband-equivalent model.

The block provides the option to interpret the Simulink signal as either the incident power wave to the RF system or the source voltage of the RF system. The 'Incident power wave' option is the most common RF modeling interpretation, while the 'Source voltage' option is provided for backwards compatibility. If the input Simulink signal is the incident power wave, the output of the RF system is the transmitted power wave. If the input is the source voltage, the output is the load voltage.

The block controls the modeling of RF Blockset physical blocks between this block and the Output Port block using:
- FIR filters to model the frequency-dependent characteristics
- Look-up tables to model the nonlinear behaviors

Optional guard bands can be specified as a fraction of the modeling bandwidth. The guards bands are implemented by applying a Tukey window to the frequency response. Modeling delay may be added to improve the response of the FIR filters.

Parameters

Treat input Simulink signal as:	Source voltage
Source impedance (ohms):	50
Finite impulse response filter length:	128
Fractional bandwidth of guard bands:	0
Modeling delay (samples):	0
Center frequency (Hz):	2e9
Sample time (s):	1e-7
Input processing:	Elements as channels (sample based)
✓ Add noise	
Initial seed:	67987

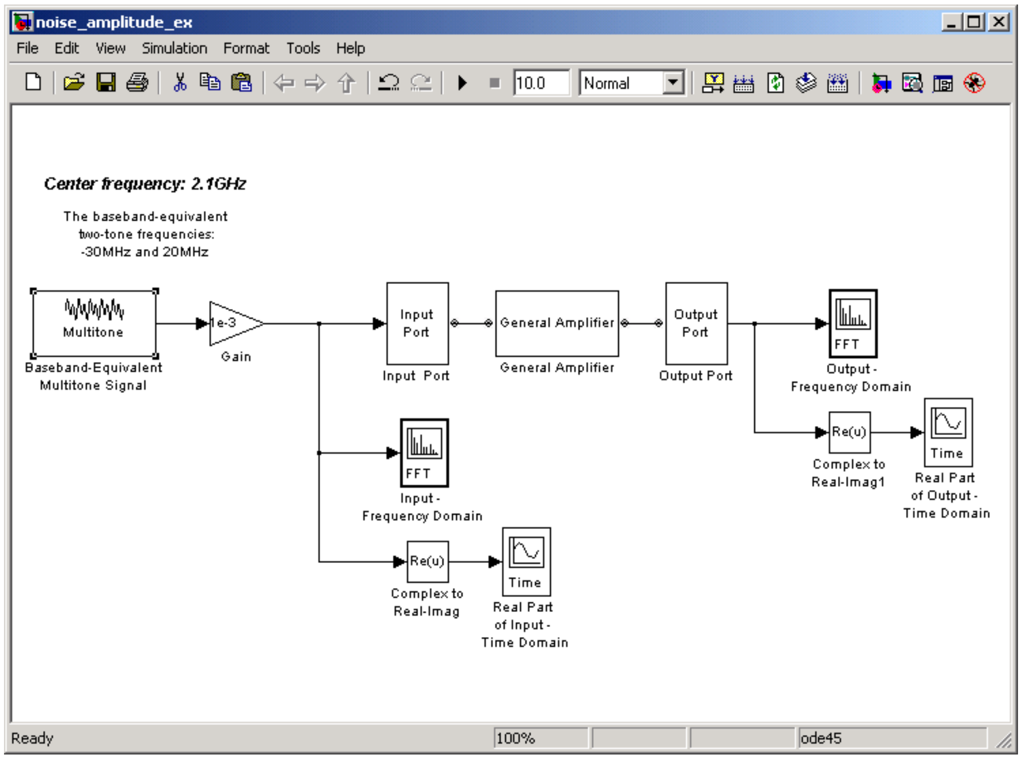
For information on how the blockset simulates noise, see “Model Noise in an RF System” on page A-5.

Plot Noise

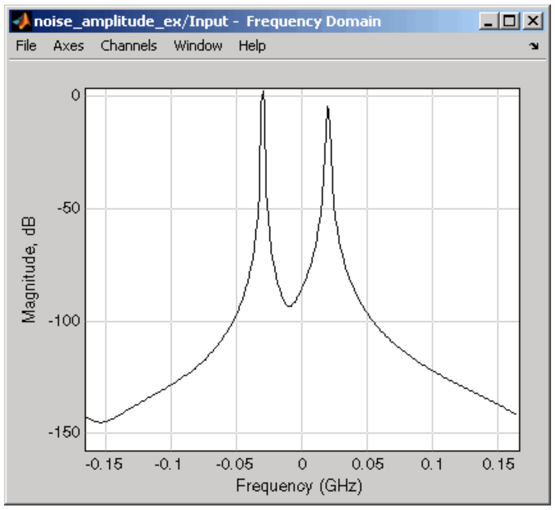
RF Blockset Equivalent Baseband software models communications systems. The noise in these systems has a very small amplitude, typically from 1e-6 to 1e-12 Watts. In contrast, the default signal power of a Communications Toolbox modulator block is 1 Watt at a nominal 1 ohm. Therefore, the signal-to-noise ratio in an RF system simulation is large, making it difficult to view the noise that the RF system adds to your signal.

To display the noise on a plot, you might need to attenuate the signal amplitude to a value within a couple orders of magnitude of the noise.

For example, suppose you have the following model that contains a multitone test signal source.

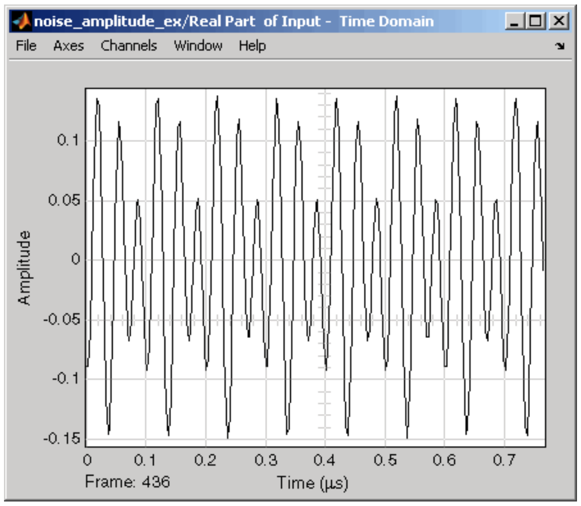


When you simulate this model, Simulink brings up several windows showing the input and output for the physical subsystem. The Input - Frequency Domain window shown in the following figure displays the input signal in the frequency domain.



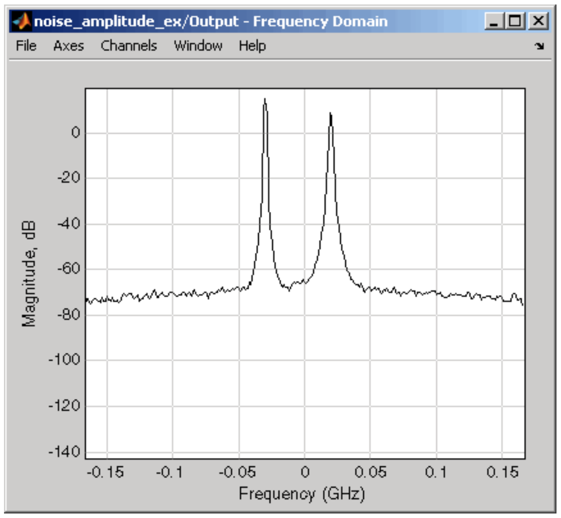
Input Signal Spectrum

The Real Part of Input - Time Domain window displays the real part of the complex-valued input signal in the time domain.



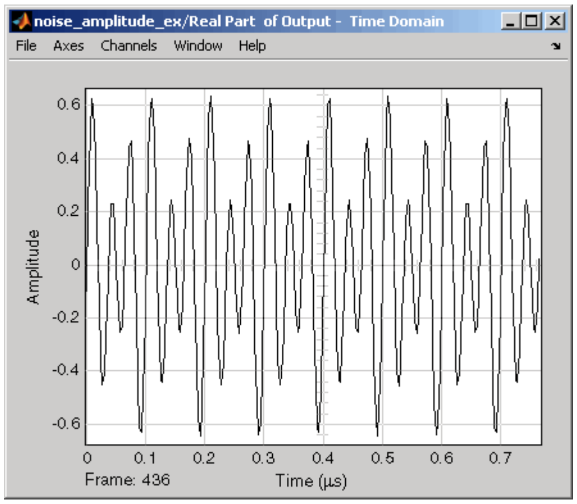
Real Part of Input Signal

In the model, the physical subsystem adds noise to the input signal. The Output - Frequency Domain window shows the noisy output signal in the frequency domain.



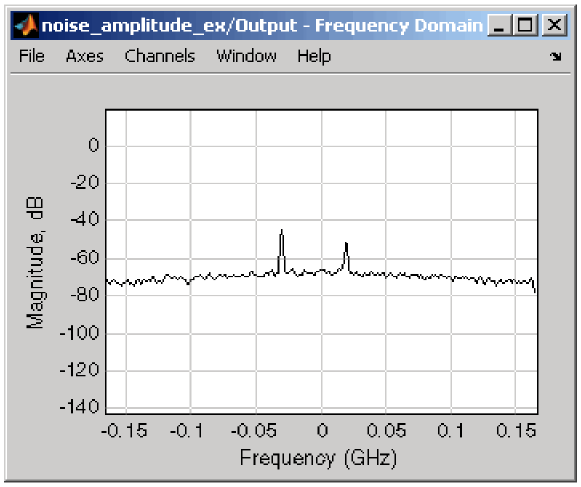
Output Signal Spectrum

The amplitude of the signal is large compared to the amplitude of the noise, so the noise is not visible in the Real Part of Output - Time Domain window that shows the real part of the time-domain output signal. Therefore, you must attenuate the amplitude of the input signal to display the noise of the time-domain output signal.



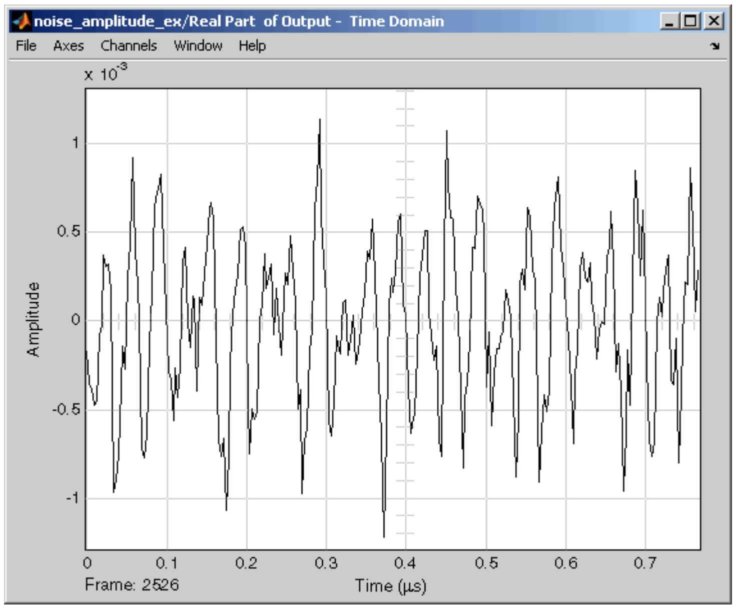
Real Part of Output Signal

Attenuate the amplitude of the input signal by setting the **Gain** parameter to `1e-3`. This is equivalent to attenuating the input signal by 60 dB. When you run the model again, the two signal peaks are not as pronounced in the Output - Frequency Domain window.



Output Signal Spectrum for Attenuated Input

You can now view the noise that the RF system adds to your signal in the Real Part of Output - Time Domain window.



Real Part of Output Signal Showing Noise

See Also

General Amplifier | Input Port | Output Port

More About

- “Model Nonlinearity” on page 6-17
- “Noise in RF Systems” on page 3-8

Plot Model Data

Create Plots

In this section...

Available Data for Plotting

RF Blockset Equivalent Baseband software lets you validate the behavior of individual RF components and physical subsystems in your model by plotting the following data:

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

Note When you plot information about a physical block, the blockset plots the actual frequency response of the block, as specified in the block dialog box. The blockset does not plot the frequency response of the complex-baseband model that it uses to simulate the block, in which the frequency response is centered at zero.

Validate Individual Blocks and Subsystems

You can plot model data for an individual physical block or for a physical subsystem. A *subsystem* is a collection of one or more physical blocks bracketed by an Input Port block and an Output Port block. To understand the behavior of specific subsystems, plot the data of the corresponding Output Port block after you run a simulation.

To validate the behavior of individual RF components in the model, plot the data of the corresponding physical blocks. You can plot data for individual blocks from each of these components either before or after you run a simulation.

You create a plot by selecting options in the block dialog box, as shown in “Create and Modify Subsystem Plots” on page 7-23. To learn about the available plots, see “Types of Plots” on page 7-3. For more information about creating plots, see “How to Create a Plot” on page 7-10.

Types of Plots

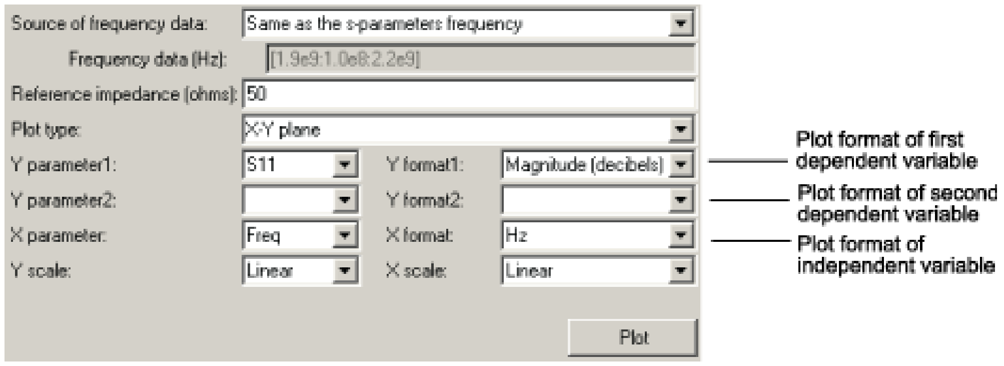
RF Blockset Equivalent Baseband software provides a variety of plots for analyzing the behavior of RF components and subsystems. The following table summarizes the available plots and charts and describes each one.

Plot Type	Plot Contents
X-Y Plane (Rectangular) Plot	Parameters as a function of frequency, input power, or operating condition, such as • S-parameters • Noise figure (NF), Noise factor (NFactor), and Noise Temperature (NTemp) • Voltage standing-wave ratio (VSWR) • Output third-order intercept point (OIP3) • Input and output reflection coefficients (GammaIn and GammaOut)
Link Budget Plot (3-D)	Parameters as a function of frequency for each component in a physical subsystem *where* The curve for a given component represents the cumulative contribution of each RF component up to and including the parameter value of that component. For more information, see “Link Budget” on page 7-8.
Polar Plane Plot	Magnitude and phase of parameters as a function of frequency or operating condition, such as • S-parameters • Input and output reflection coefficients (GammaIn and GammaOut)
Smith® Chart	Real and imaginary parts of S-parameters as a function of frequency or operating condition, used for analyzing the reflections caused by impedance mismatch.
Composite Plot	Multiple plots and charts in one figure.

To learn how to create these plots, see “How to Create a Plot” on page 7-10.

Plot Formats

When you create a plot from a block dialog box, you must specify the format of the data for both the *x*- and *y*-axes.


Source of frequency data: Same as the s-parameters frequency
Frequency data (Hz): [1.9e9:1.0e8:2.2e9]
Reference impedance (ohms): 50
Plot type: X-Y plane
Y parameter1: S11 Y format1: Magnitude (decibels)
Y parameter2: Y format2:
X parameter: Freq X format: Hz
Y scale: Linear X scale: Linear

Plot
Plot format of first dependent variable
Plot format of second dependent variable
Plot format of independent variable

These plot options define how RF Blockset Equivalent Baseband software displays the data on the plot.

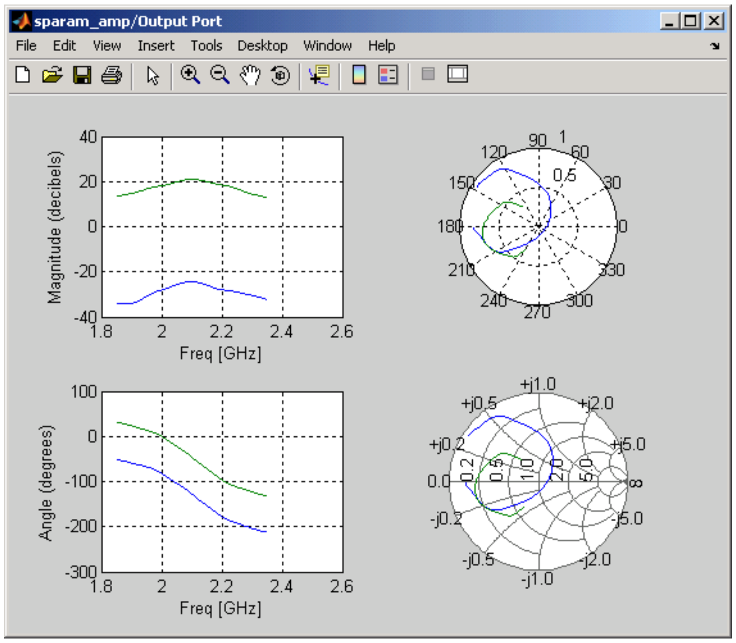
The available formats vary with the data you select to plot. The data you can plot depends on the plot type you select. The plot formats determine whether the blockset converts the data to a new set of units, or performs a calculation on the data. For example, setting the format to `Real` tells the blockset to compute and plot the real part of the parameter.

The following topics describe the available parameters and formats for each plot type:

- "Composite Data" on page 7-4
- "X-Y Plane" on page 7-6
- "Link Budget" on page 7-8
- "Polar Plane Plots and Smith Charts" on page 7-9

Composite Data

The composite data plot automatically generates four separate plots in one figure window, showing the frequency dependence of several parameters. The following figure shows an example of such a plot.



Example — Composite Data Plot

Note For composite data plots, you do not need to specify the parameters or the formats—they are set automatically.

The combination of plots differs based on the type of block and the specified block data. The following table describes the contents of the composite data plot for each specification. The Plot Contents column lists the types of plots as they appear on the composite plot, counterclockwise and starting in the upper-left corner. The blockset plots all data as a function of frequency.

Block	Specified Data	Plot Contents
General Amplifier or General Mixer	Network parameters **OR** Network parameters and noise	• X-Y plot, magnitude of S_{12} and S_{21} in decibels • X-Y plot, phase of S_{12} and S_{21} in degrees • Z Smith Chart, real and imaginary parts of S_{11} and S_{22} • Polar plot, magnitude and phase of S_{11} and S_{22}
	Network parameters and power **OR** Network parameters, noise, and power	• X-Y plot, magnitude of S_{12} and S_{21} in decibels • X-Y plot, output power (P_{out}) in dBm (decibels referenced to one milliwatt) • Z Smith Chart, real and imaginary parts of S_{11} and S_{22} • Polar plot, magnitude and phase of S_{11} and S_{22}

Block	Specified Data	Plot Contents
Other Physical block	Network parameters **OR** Network parameters and noise (S-, Y-, and Z-Parameters Amplifiers and Mixers only) **Note** Only the General Amplifier and General Mixer blocks accept power data.	• X-Y plot, magnitude of S_{12} and S_{21} in decibels • X-Y plot, phase of S_{12} and S_{21} in degrees • Z Smith Chart, real and imaginary parts of S_{11} and S_{22} • Polar plot, magnitude and phase of S_{11} and S_{22}

X-Y Plane

You can plot any parameters that are relevant to your block on an X-Y plane plot. For this type of plot, you specify data for both the *x*- and *y*-axes. If you specify two Y parameters, and you specify different formats for the two Y parameters, the blockset plots the second Y parameter on the right *y*-axis.

The following table summarizes the available Y parameters and formats. The parameters and formats are the same for both the left and right *y*-axes.

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

Y Parameter	Y Format

`S11`, `S12`, `S21`, `S22`
 	
`LS11`, `LS12`, `LS21`, `LS22` (General Amplifier and General Mixer blocks with multiple operating conditions only)
	`Magnitude (decibels)`
	`Magnitude (linear)`
	`Angle (degrees)`
	`Angle (radians)`
	`Real`
	`Imaginary`

`NF`	`Magnitude (decibels)`
`NFactor`	`None` This format tells the blockset to plot the noise factor as it is specified to or calculated by the block.
`NTemp`	`Kelvin`

`OIP3`	`dBm`
	`dBW`
	`W`
	`mW`

`VSWRIn`, `VSWROut`	`Magnitude (decibels)`
	`None`
	This format tells the blockset to plot the voltage standing-wave ratio as it is specified to or calculated by the block.

Y Parameter	Y Format
Pout (General Amplifier and General Mixer blocks with power data only)	dBm dBW W mW
Phase (General Amplifier and General Mixer blocks with power data only)	Angle (degrees) Angle (radians)
AM/AM (General Amplifier and General Mixer blocks with power data only)	Magnitude (decibels) None This format tells the blockset to plot the AM/AM conversion as it is specified to or calculated by the block.
AM/PM (General Amplifier and General Mixer blocks with power data only)	Angle (degrees) Angle (radians)
PhaseNoise (Mixer blocks only)	dBc/Hz
FMIN (Amplifier and Mixer blocks with spot noise data only)	Magnitude (decibels) None This format tells the blockset to plot the minimum noise figure as it is specified to or calculated by the block.
GammaIn, GammaOut(Output Port block only)	Magnitude (decibels) Magnitude (linear) Angle (degrees) Angle (radians) Real Imaginary
GAMMAOPT (Amplifier and Mixer blocks with spot noise data only)	Magnitude (decibels) Magnitude (linear) Angle (degrees) Angle (radians) Real Imaginary
RN (Amplifier and Mixer blocks with spot noise data only)	None This format tells the blockset to plot the noise resistance as it is specified to or calculated by the block.

The available X parameters depend on the Y parameters you select. The following table summarizes the available X parameters for each of the Y parameters in the preceding table.

Y Parameter	X Parameter
Pout, Phase, LS11, LS12, LS21, LS22	Pin Freq
S11, S12, S21, S22, NF, OIP3, VSWRIn, VSWROut, GAMMAIn, GAMMAOut, FMIN, GAMMAOPT, RN	Freq
AM/AM, AM/PM	AM

The following table shows the X formats that are available for the X parameters listed in the preceding table.

X Parameter	X Format

`Pin`	`dBm`
	`dBW`
	`W`
	`mW`

`Freq`	`THz`
	`GHz`
	`MHz`
	`KHz`
	`Hz`
	`Auto` (`xformat` is chosen to provide the best scaling for the given `xparameter` values.)

`AM`	`Magnitude (dB)`
	`Magnitude (linear)`

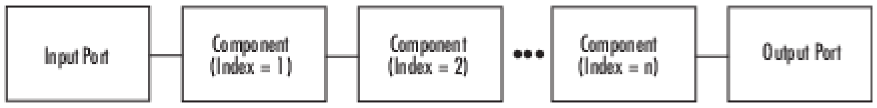
When you import block data from a `.p2d` or `.s2d` file, you can also plot Y parameters as a function of any operating condition from the file that has numeric values, such as bias. You can specify an operating condition as the X parameter only when validating individual blocks, and the format is always `None`. This format tells the blockset to plot the operating condition values as they are specified in the file.

Link Budget

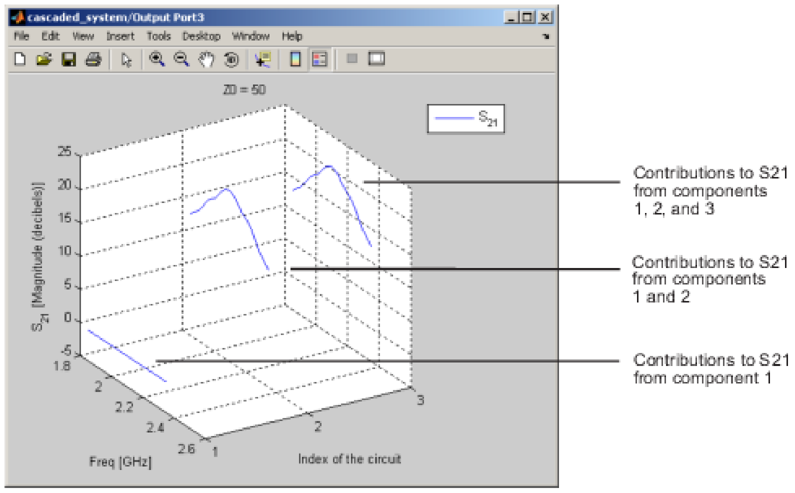
You use the `Link budget` plot to understand the individual contribution of each block to a plotted Y parameter value in a physical subsystem with multiple components between the Input Port and the Output Port blocks.

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

The following figure shows how the circuit index is assigned to a component in a physical subsystem based on its sequential position in the subsystem.



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



Example — Link Budget Plot

The following table summarizes the Y parameters and formats that are available for a link budget plot.

Y Parameter	Y Format

S11, S12, S21, S22	`Magnitude (decibels)`
	`Magnitude (linear)`
	`Angle (degrees)`
	`Real`
	`Imaginary`

`OIP3`	`dBm`
	`dBW`
	`W`
	`mW`

`NF`	`Magnitude (decibels)`
	`Magnitude (linear)`

`NFactor`	`None` This format tells the blockset to plot the noise factor as it is specified to the block.
`NTemp`	`Kelvin`

If you specify two Y parameters, the blockset puts both parameters in a single plot. The Y parameters must have the same formats.

For a link budget plot, the X parameter is always `Freq`. The format of the X parameter specifies the units of the *x*-axis.

Polar Plane Plots and Smith Charts

You can use RF Blockset Equivalent Baseband software to generate Polar plots and Smith Charts. When you select these plot types, you do not need to specify the format of any Y parameters—the formats are set automatically. If you specify two Y parameters, the blockset puts both parameters in a single plot.

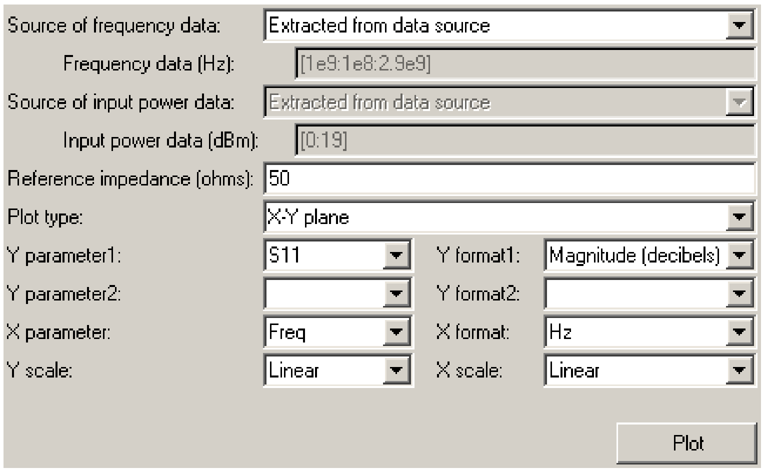
The following table describes the Polar plot and Smith Chart options. It also lists the available Y parameters.

Plot Type	Y Parameter
`Polar plane`	`S11`, `S12`, `S21`, `S22` `LS11`, `LS12`, `LS21`, `LS22` (General Amplifier and General Mixer blocks with data from a P2D file only) `GammaIn`, `GammaOut` (Output Port block only)
`Z Smith chart`	`S11`, `S22` `LS11`, `LS22` (General Amplifier and General Mixer blocks with data from a P2D file only) `GammaIn`, `GammaOut` (Output Port block only)
`Y Smith chart`	`S11`, `S22` `LS11`, `LS22` (General Amplifier and General Mixer blocks with data from a P2D file only) `GammaIn`, `GammaOut` (Output Port block only)
`ZY Smith chart`	`S11`, `S22` `LS11`, `LS22` (General Amplifier and General Mixer blocks with data from a P2D file only) `GammaIn`, `GammaOut` (Output Port block only)

By default, the X parameter is `Freq`. The format of the X parameter specifies the units of the *x*-axis. When you import block data from a `.p2d` or `.s2d` file, you can also plot Y parameters as a function of any operating condition from the file that has numeric values, such as bias. You can specify an operating condition as the X parameter only when validating individual blocks, and the format is always `None`.

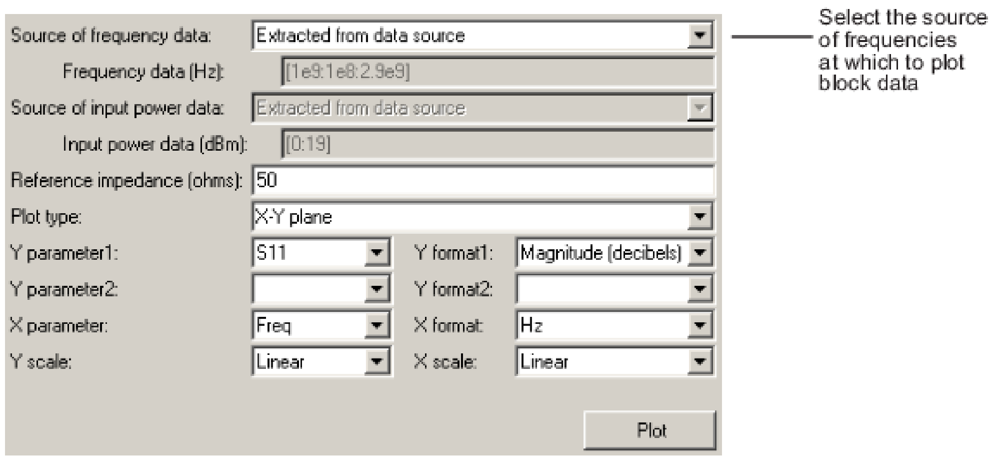
How to Create a Plot

1 Double-click the block to open the block dialog box, and select the **Visualization** tab. The following figure shows the contents of the tab.



Source of frequency data: Extracted from data source
Frequency data (Hz): [1e9:1e8:2.9e9]
Source of input power data: Extracted from data source
Input power data (dBm): [0:19]
Reference impedance (ohms): 50
Plot type: X-Y plane
Y parameter1: S11 Y format1: Magnitude (decibels)
Y parameter2: Y format2:
X parameter: Freq X format: Hz
Y scale: Linear X scale: Linear

2 Select the **Source of frequency data**.



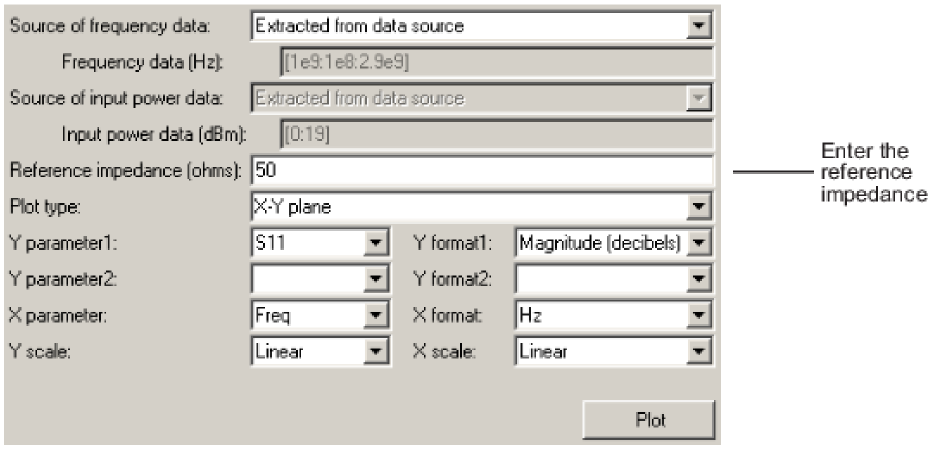
Source of frequency data: Extracted from data source
Frequency data (Hz): [1e9:1e8:2.9e9]
Source of input power data: Extracted from data source
Input power data (dBm): [0:19]
Reference impedance (ohms): 50
Plot type: X-Y plane
Y parameter1: S11 Y format1: Magnitude (decibels)
Y parameter2: Y format2:
X parameter: Freq X format: Hz
Y scale: Linear X scale: Linear

Plot
Select the source of frequencies at which to plot block data

This value is the source of the frequency values at which to plot block data. The following table summarizes the available types of sources for the various types of blocks.

Source of frequency data	Description	Blocks
`User-specified`	Vector of frequencies that you enter. When you select `User-specified` in the **Source of frequency data** list, the **Frequency range (Hz)** field is displayed. Enter a vector specifying the range of frequencies you want to plot. For example, to plot block data from 0.3 MHz to 5 GHz by 0.1 MHz, enter `[0.3e6:0.1e6:5e9]`. **Note** When you select **PhaseNoise** in the **Parameter** list and `User-specified` in the **Source of frequency data** list, the **Frequency range (Hz)** field is disabled. You use the **Phase noise frequency offset (Hz)** block parameter to specify the frequency values at which to plot block data.	All physical blocks
`Derived from Input Port parameters` (Available after running a simulation or clicking the **Update Diagram** button)	Modeling frequencies derived from the Input Port block parameters. For information on how the blockset computes the modeling frequencies, see "Determine Modeling Frequencies" on page A-3.	All physical blocks
`Same as the S-parameters`	Frequency values specified in the **Frequency** block parameter.	S-Parameters Passive Network, S-Parameters Amplifier, S-Parameters Mixer
`Same as the Y-parameters`	Frequency values specified in the **Frequency** block parameter.	Y-Parameters Passive Network, Y-Parameters Amplifier, Y-Parameters Mixer
`Same as the Z-parameters`	Frequency values specified in the **Frequency** block parameter.	Z-Parameters Passive Network, Z-Parameters Amplifier, Z-Parameters Mixer
`Extracted from data source`	Frequency values imported into the **Data file** or **RFDATA object** block parameter.	General Passive Network, General Amplifier, and General Mixer

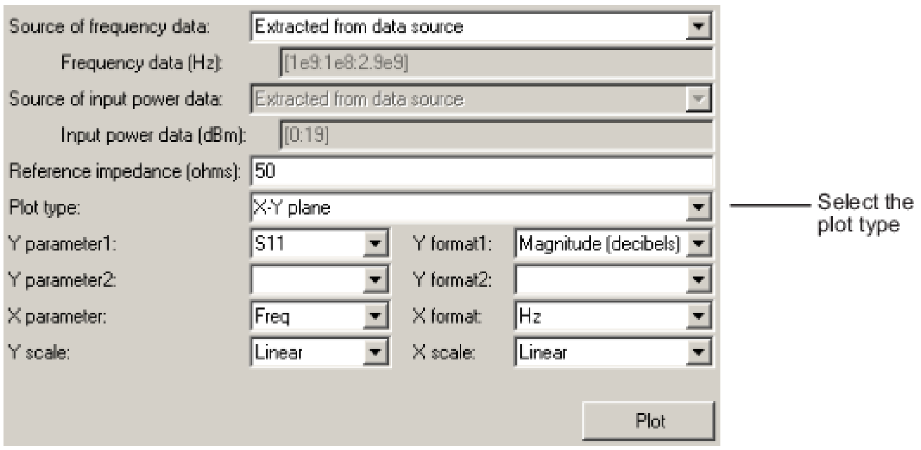
3 Enter the **Reference impedance**.



Source of frequency data:	Extracted from data source		
Frequency data (Hz):	[1e9:1e8:2.9e9]		
Source of input power data:	Extracted from data source		
Input power data (dBm):	[0:19]		
Reference impedance (ohms):	50		
Plot type:	X-Y plane		
Y parameter1:	S11	Y format1:	Magnitude (decibels)
Y parameter2:		Y format2:	
X parameter:	Freq	X format:	Hz
Y scale:	Linear	X scale:	Linear

This value is the reference impedance to use when plotting small-signal parameters.

4 Select the **Plot type**.

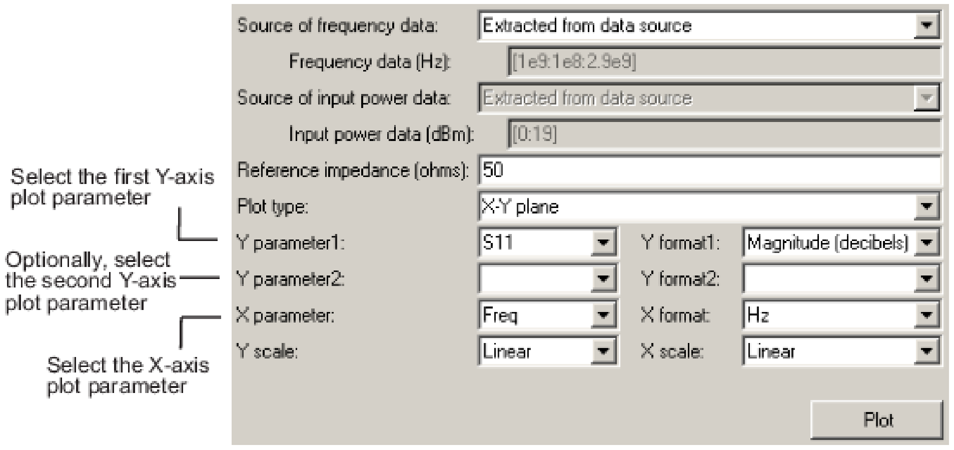


Source of frequency data:	Extracted from data source		
Frequency data (Hz):	[1e9:1e8:2.9e9]		
Source of input power data:	Extracted from data source		
Input power data (dBm):	[0:19]		
Reference impedance (ohms):	50		
Plot type:	X-Y plane		
Y parameter1:	S11	Y format1:	Magnitude (decibels)
Y parameter2:		Y format2:	
X parameter:	Freq	X format:	Hz
Y scale:	Linear	X scale:	Linear

This value is the type of plot. For a description of the options, see “Types of Plots” on page 7-3.

5 Select the following parameters:

- **Y Parameter1** — The first parameter for the Y-axis.
- **Y Parameter2** — The second parameter for the Y-axis (optional).
- **X Parameter** — The parameter for the X-axis.

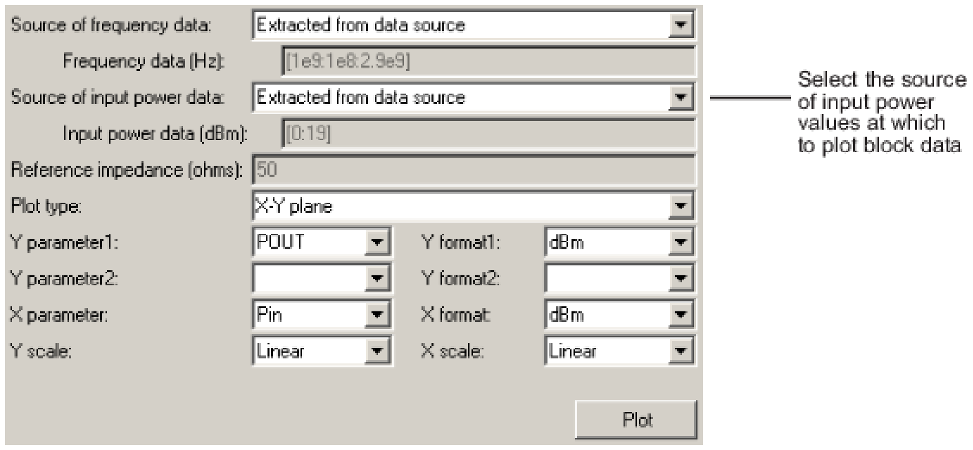

Select the first Y-axis plot parameter
Optionally, select the second Y-axis plot parameter
Select the X-axis plot parameter
Plot

```
Source of frequency data:     Extracted from data source
Frequency data (Hz):          [1e9:1e8:2.9e9]
Source of input power data:   Extracted from data source
Input power data (dBm):       [0:19]
Reference impedance (ohms):   50
Plot type:                    X-Y plane
Y parameter1:  S11            Y format1:  Magnitude (decibels)
Y parameter2:                 Y format2:
X parameter:   Freq           X format:   Hz
Y scale:       Linear         X scale:    Linear
```

These parameters specify the data to be plotted. The available choices vary with the type of plot. For a description of the options for a particular plot type, see the topic on that plot type in “Plot Formats” on page 7-3.

6 If you select a large-signal parameter for one or more *y*-axis parameters, select the **Source of power data**.

Note Large-signal parameters are available only for General Amplifier or General Mixer blocks that contain power data.


Select the source of input power values at which to plot block data
Plot

```
Source of frequency data:     Extracted from data source
Frequency data (Hz):          [1e9:1e8:2.9e9]
Source of input power data:   Extracted from data source
Input power data (dBm):       [0:19]
Reference impedance (ohms):   50
Plot type:                    X-Y plane
Y parameter1:  POUT           Y format1:  dBm
Y parameter2:                 Y format2:
X parameter:   Pin            X format:   dBm
Y scale:       Linear         X scale:    Linear
```

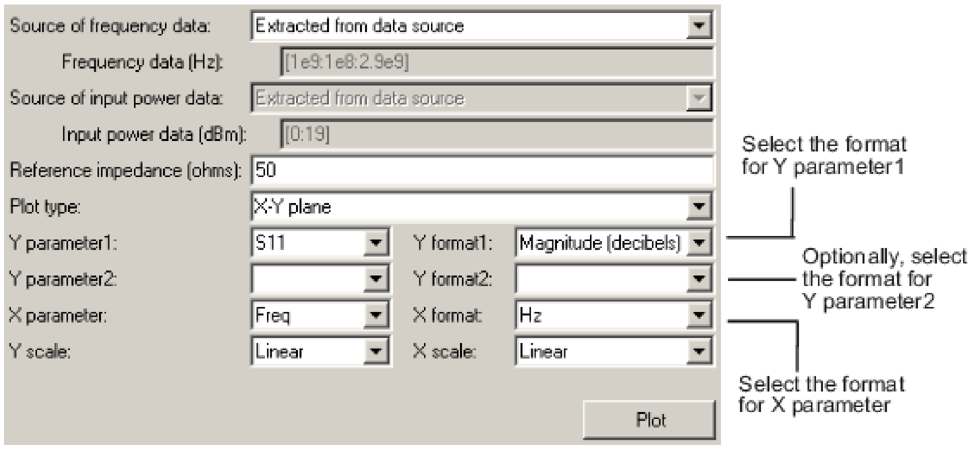
This value is the source of the input power values at which to plot block data. The following table summarizes the available types of sources for the General Amplifier and General Mixer blocks.

Source of frequency data	Description
`Extracted from data source`	Input power values imported into the **Data file** or **RFDATA object** block parameter.

Source of frequency data	Description
`User-specified`	Vector of power values that you enter. When you select `User-specified` in the **Source of power data** list, the **Input power data (dBm)** field is displayed. Enter a vector specifying the range of power values you want to plot. For example, to plot block data from 1 dBm to 10 dBm by 2 dBm, enter `[1:2:10]`.

7 Select the following formats:

- **Y Format1** — The format for the first Y parameter.
- **Y Format2** — The format for the second Y parameter (optional).
- **X Format** — The format for the X parameter.

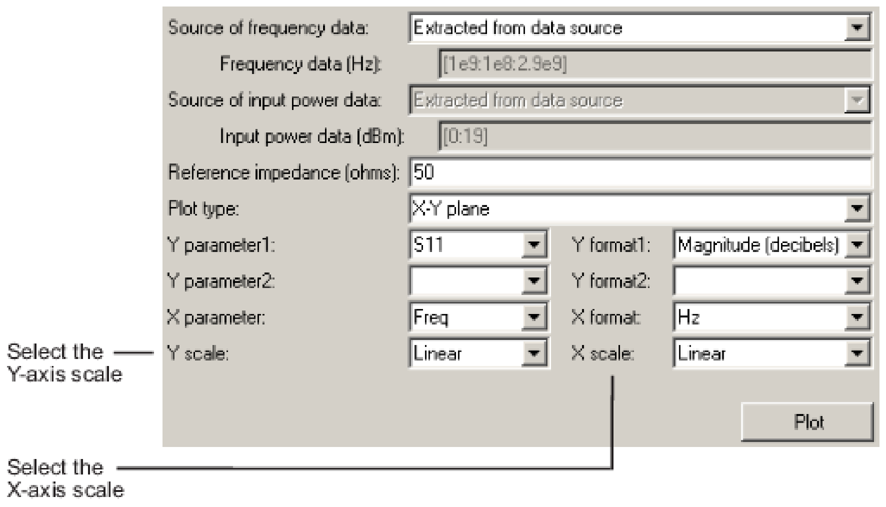


Source of frequency data:	Extracted from data source
Frequency data (Hz):	[1e9:1e8:2.9e9]
Source of input power data:	Extracted from data source
Input power data (dBm):	[0:19]
Reference impedance (ohms):	50
Plot type:	X-Y plane
Y parameter1:	S11
Y format1:	Magnitude (decibels)
Y parameter2:	
Y format2:	
X parameter:	Freq
X format:	Hz
Y scale:	Linear
X scale:	Linear

Select the format for Y parameter1
Optionally, select the format for Y parameter2
Select the format for X parameter
Plot

These are the X and Y formats for plotting the selected parameter. The available choices vary based on the selected parameter. For a description of the options for a particular plot type, see the topic on that plot type in “Plot Formats” on page 7-3.

8 Select the **X Scale** and **Y Scale**.


Source of frequency data: Extracted from data source
Frequency data (Hz): [1e9:1e8:2.9e9]
Source of input power data: Extracted from data source
Input power data (dBm): [0:19]
Reference impedance (ohms): 50
Plot type: X-Y plane
Y parameter1: S11 Y format1: Magnitude (decibels)
Y parameter2: Y format2:
X parameter: Freq X format: Hz
Y scale: Linear X scale: Linear

These are the scales on which to plot the data. The available choices are `Linear` and `Log`.

9 Click **Plot**.

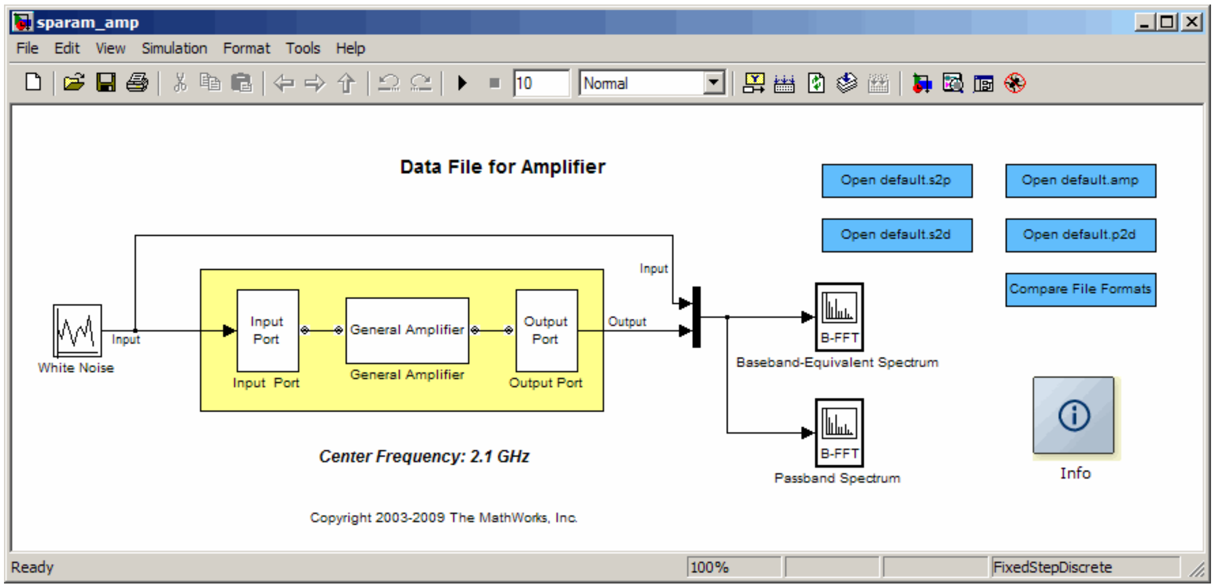
Note By default, the blockset does not add a legend to some plots. To display the plot legend, type `legend show` at the MATLAB prompt.

Example — Plot Component Data on a Z Smith Chart

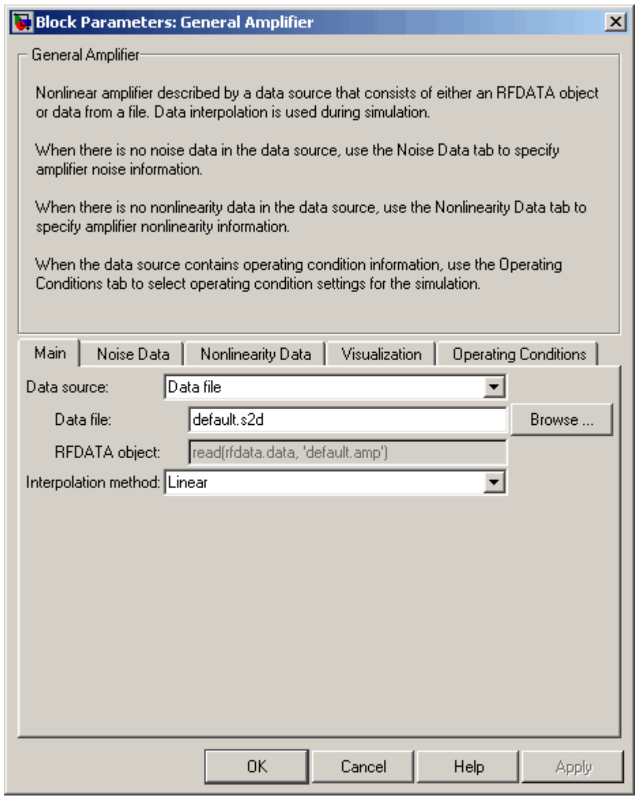
In this example, you simulate the frequency response of an amplifier using data from the `default.s2d` S2D file.

Using a RF Blockset Equivalent Baseband example model, you import the data file into a General Amplifier block and validate the amplifier by plotting the S-parameters of the block on a Z Smith Chart.

1 Type `sparam_amp` at the MATLAB prompt to open the “AMP Data File for Amplifier” example.



2 Double-click the General Amplifier block to display its parameters.


Block Parameters: General Amplifier

General Amplifier

Nonlinear amplifier described by a data source that consists of either an RFDATA object or data from a file. Data interpolation is used during simulation.

When there is no noise data in the data source, use the Noise Data tab to specify amplifier noise information.

When there is no nonlinearity data in the data source, use the Nonlinearity Data tab to specify amplifier nonlinearity information.

When the data source contains operating condition information, use the Operating Conditions tab to select operating condition settings for the simulation.

Data source: Data file

Data file: default.s2d

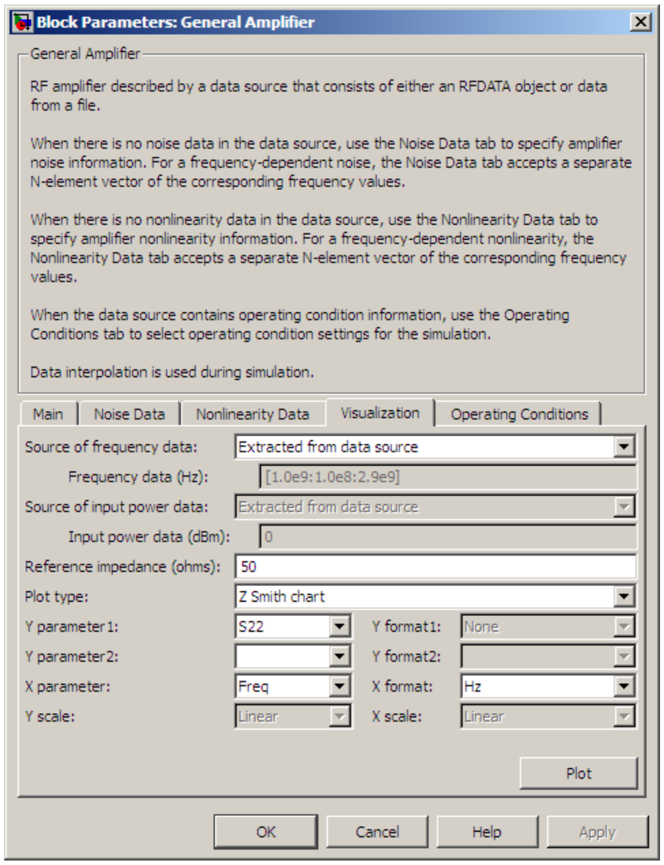
RFDATA object: read(rfdata.data, 'default.amp')

Interpolation method: Linear

As shown in the preceding figure, the **Data source** parameter is set to `Data file`and the **Data file** parameter is set to `default.s2d`. These values tell the blockset to import data from the file `default.s2d`. The block uses this data, along with the other block parameters, in simulation.

3 Select the **Visualization** tab and set the General Amplifier block parameters as follows:

- In the **Plot type** list, select `Z Smith chart`.
- In the **Y Parameter1** list, select S22.



Block Parameters: General Amplifier

General Amplifier

RF amplifier described by a data source that consists of either an RFDATA object or data from a file.

When there is no noise data in the data source, use the Noise Data tab to specify amplifier noise information. For a frequency-dependent noise, the Noise Data tab accepts a separate N-element vector of the corresponding frequency values.

When there is no nonlinearity data in the data source, use the Nonlinearity Data tab to specify amplifier nonlinearity information. For a frequency-dependent nonlinearity, the Nonlinearity Data tab accepts a separate N-element vector of the corresponding frequency values.

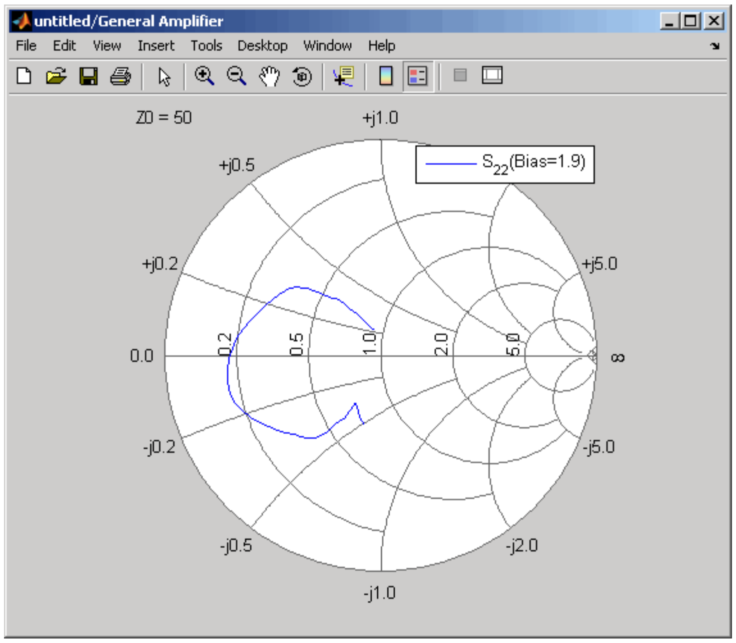
When the data source contains operating condition information, use the Operating Conditions tab to select operating condition settings for the simulation.

Data interpolation is used during simulation.

Source of frequency data:	Extracted from data source		
Frequency data (Hz):	[1.0e9:1.0e8:2.9e9]		
Source of input power data:	Extracted from data source		
Input power data (dBm):	0		
Reference impedance (ohms):	50		
Plot type:	Z Smith chart		
Y parameter1:	S22	Y format1:	None
Y parameter2:		Y format2:	
X parameter:	Freq	X format:	Hz
Y scale:	Linear	X scale:	Linear

4 Click **Plot**.

This action creates a Z Smith Chart of the S_{22} parameters using the frequency data from the `default.s2d` file.



General Amplifier Frequency Response

Note To display data tips for a plotted line, select **Tools > Data Cursor**. Click the data cursor on the plotted line to see the frequency and the parameter value at that point. See the MATLAB documentation for more information.

See Also

More About

- "Modify Plots" on page 7-21
- "Update Plots" on page 7-20
- "Create and Modify Subsystem Plots" on page 7-23

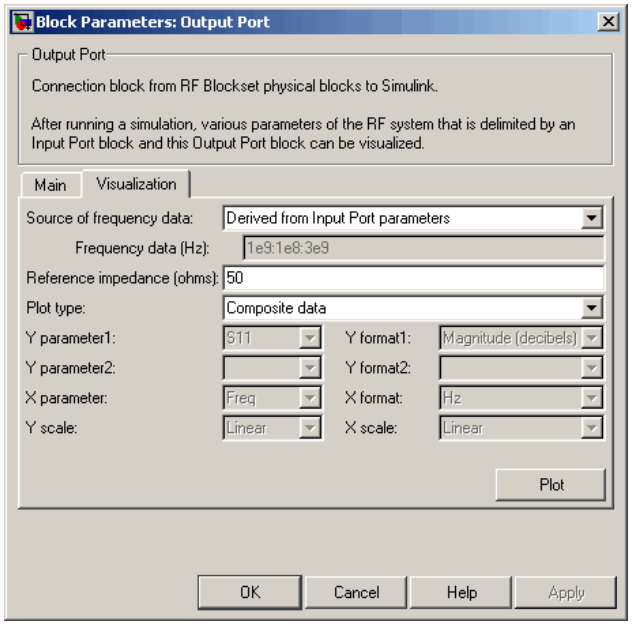
Update Plots

When you run a simulation, the blockset continues to display any open plots, but does not update the plots to reflect new simulation results. You must update the subsystem plots after the simulation to display the behavior of the revised subsystem.

When you make changes to the parameters of blocks that represent individual RF components, you need to update any open plots, because the blockset does not automatically redraw the plots.

To update an existing plot:

1 Double-click the block to open the block dialog box, and select the **Visualization** tab.



Block Parameters: Output Port

Output Port

Connection block from RF Blockset physical blocks to Simulink.

After running a simulation, various parameters of the RF system that is delimited by an Input Port block and this Output Port block can be visualized.

Source of frequency data: Derived from Input Port parameters
Frequency data (Hz): 1e9:1e8:3e9
Reference impedance (ohms): 50
Plot type: Composite data
Y parameter1: S11
Y format1: Magnitude (decibels)
Y parameter2:
Y format2:
X parameter: Freq
X format: Hz
Y scale: Linear
X scale: Linear

Example Block Dialog Box Showing Visualization Tab

2 Click **Plot**.

See Also

More About

- “Create Plots” on page 7-2
- “Modify Plots” on page 7-21
- “Create and Modify Subsystem Plots” on page 7-23

Modify Plots

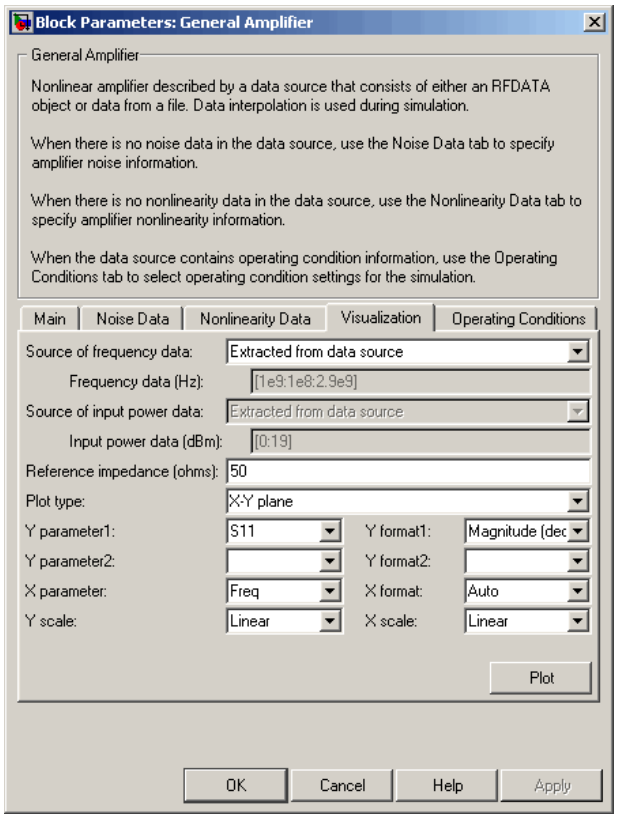
You can modify an existing plot by changing the plot options. The outcome depends on the parameter you change.

The following table summarizes the results of changing the plot options.

Block Parameter	Plot Change
Source of frequency data OR **Frequency data (Hz)**	Redraws plot using the new frequency data.
Source of power data OR **Input power data (dBm)**	Redraws plot using the new power data.
Plot type	Draws plot in a new figure using the new plot type. **Note** If the current plot options are valid for the new plot type, they retain their values. Otherwise, they revert to their default values.
Y Parameter1 OR **Y Parameter2**	If the new parameter has the same independent variable and format as the one on the plot, the blockset adds the new parameter to the existing plot. Otherwise, it redraws the plot for the new parameter and independent variable.
Y Format1 OR **Y Format2**	Redraws plot using the new format.
X Parameter	Redraws plot using the new independent variable.
X Format	Redraws plot using the new format.
X Scale	Redraws plot using the new scale.
Y Scale	Redraws plot using the new scale.

To modify a plot:

1 Double-click the block to open the block dialog box, and select the **Visualization** tab.


Block Parameters: General Amplifier

General Amplifier

Nonlinear amplifier described by a data source that consists of either an RFDATA object or data from a file. Data interpolation is used during simulation.

When there is no noise data in the data source, use the Noise Data tab to specify amplifier noise information.

When there is no nonlinearity data in the data source, use the Nonlinearity Data tab to specify amplifier nonlinearity information.

When the data source contains operating condition information, use the Operating Conditions tab to select operating condition settings for the simulation.

Source of frequency data: Extracted from data source
Frequency data (Hz): [1e9:1e8:2.9e9]
Source of input power data: Extracted from data source
Input power data (dBm): [0:19]
Reference impedance (ohms): 50
Plot type: X-Y plane
Y parameter1: S11 Y format1: Magnitude (dec
Y parameter2: Y format2:
X parameter: Freq X format: Auto
Y scale: Linear X scale: Linear

Example Block Dialog Box Showing Plot Parameters

2 Change the plot options.

3 Click **Plot**.

See Also

More About

- "Create and Modify Subsystem Plots" on page 7-23
- "Create Plots" on page 7-2
- "Update Plots" on page 7-20

Create and Modify Subsystem Plots

In this section...

Plot the Network Parameters of a Subsystem

In this part of the example, you open and run a RF Blockset Equivalent Baseband example that uses file data to specify an amplifier in a physical subsystem. Then, you plot the network parameters of the physical subsystem, which consists of the General Amplifier, the Input Port, and the Output Port blocks.

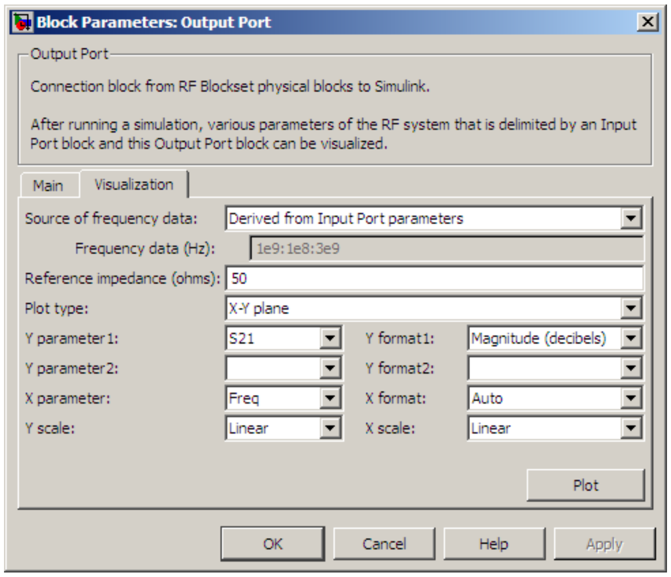
1 Type `sparam_amp` at the MATLAB prompt to open the RF Blockset Equivalent Baseband example called "AMP Data File for Amplifier".

2 In the model window, click **Run** to run the simulation. Once the simulation has reached steady state, click **Debug** > **Simulate** > **Stop** to stop the simulation.

3 Double-click the Output Port block to open the block dialog box.

4 Select the **Visualization** tab and set the Output Port block parameters as follows:

- In the **Source of frequency data** list, select `Derived from Input Port parameters`.
- In the **Plot type** list, select `X-Y plane`.
- In the **Y Parameter1** list, select `S21`.


Block Parameters: Output Port

Output Port

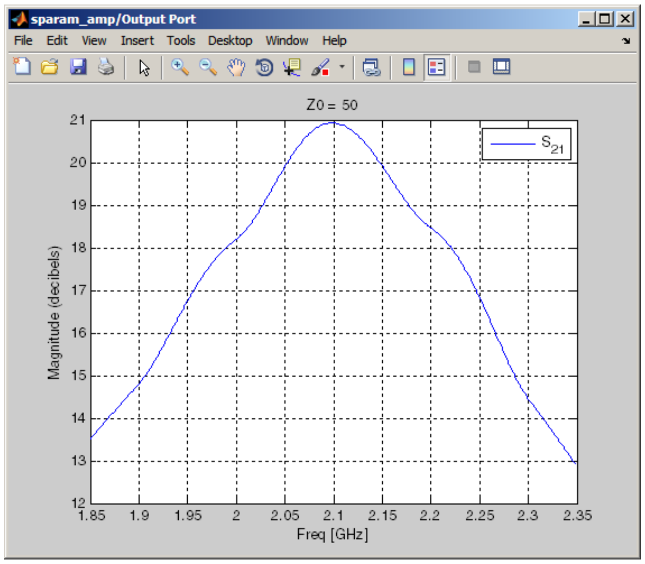
Connection block from RF Blockset physical blocks to Simulink.

After running a simulation, various parameters of the RF system that is delimited by an Input Port block and this Output Port block can be visualized.

Source of frequency data: Derived from Input Port parameters
Frequency data (Hz): 1e9:1e8:3e9
Reference impedance (ohms): 50
Plot type: X-Y plane
Y parameter1: S21 Y format1: Magnitude (decibels)
Y parameter2: Y format2:
X parameter: Freq X format: Auto
Y scale: Linear X scale: Linear

5 Click **Plot**.

This action plots the magnitude of S_{21} (in decibels) as a function of frequency on an X-Y plot.



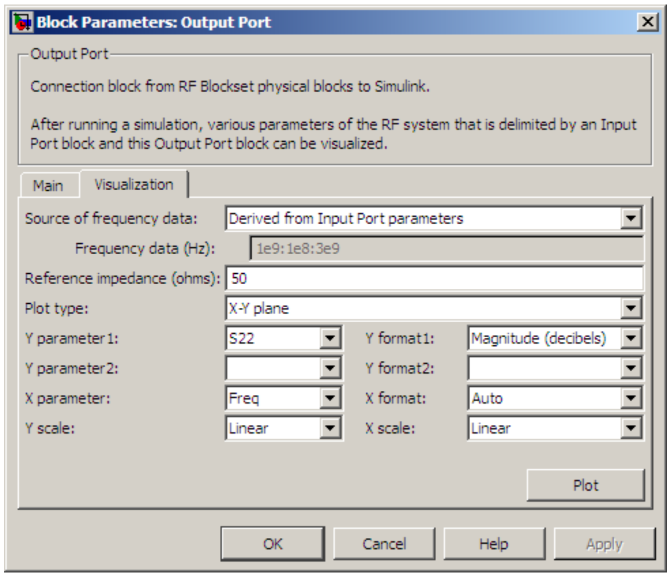
S_{21} versus Frequency for a Physical Subsystem

Add Data to an Existing Plot

In this part of the example, you add data to the plot you created in “Plot the Network Parameters of a Subsystem” on page 7-23.

1 Double-click the Output Port block to open the block dialog box.

2 Select the **Visualization** tab and change the value of **Y Parameter1** to S22.


Block Parameters: Output Port

Output Port

Connection block from RF Blockset physical blocks to Simulink.

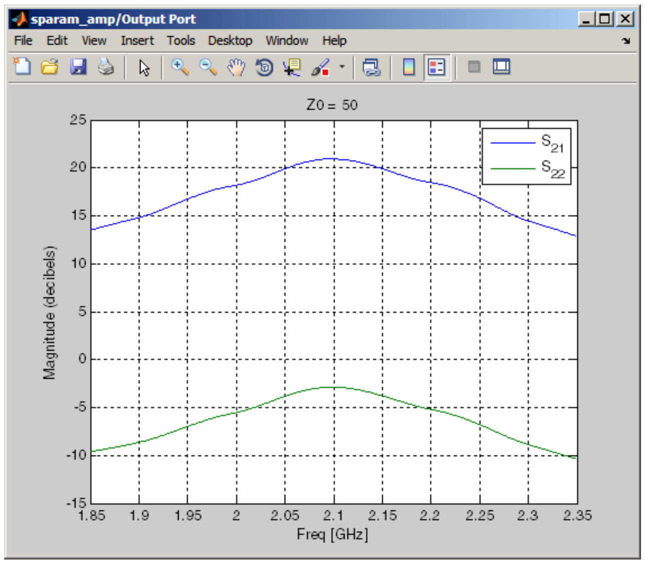
After running a simulation, various parameters of the RF system that is delimited by an Input Port block and this Output Port block can be visualized.

Main | Visualization

Source of frequency data:	Derived from Input Port parameters		
Frequency data (Hz):	1e9:1e8:3e9		
Reference impedance (ohms):	50		
Plot type:	X-Y plane		
Y parameter1:	S22	Y format1:	Magnitude (decibels)
Y parameter2:		Y format2:	
X parameter:	Freq	X format:	Auto
Y scale:	Linear	X scale:	Linear

3 Click **Plot**.

This action adds S_{22} to the plot.



S_{21} and S_{22} versus Frequency for a Physical Subsystem

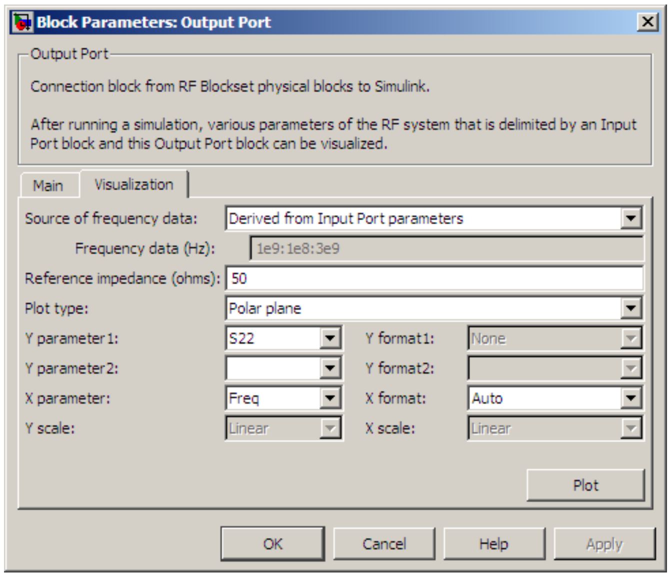
Change Data on an Existing Plot

In this part of the example, you change the data on the plot you created in the previous steps of the example by modifying the **Plot type**.

1 Double-click the Output Port block to open the block dialog box.

2 Select the **Visualization** tab and change the value of **Plot type** to `Polar plane`, as shown in the following figure.

As the figure shows, the value of **Y Parameter1** remains as S_{22}, the last parameter selected for the previous plot.


Block Parameters: Output Port

Output Port

Connection block from RF Blockset physical blocks to Simulink.

After running a simulation, various parameters of the RF system that is delimited by an Input Port block and this Output Port block can be visualized.

Source of frequency data: Derived from Input Port parameters

Frequency data (Hz): 1e9:1e8:3e9

Reference impedance (ohms): 50

Plot type: Polar plane

Y parameter1: S22 Y format1: None

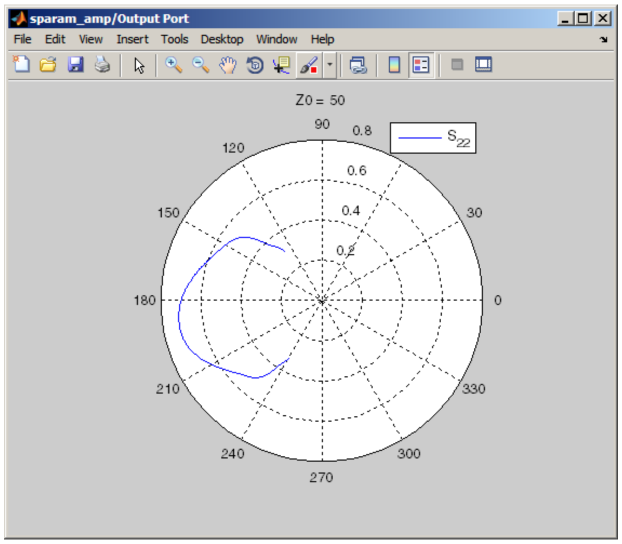
Y parameter2: Y format2:

X parameter: Freq X format: Auto

Y scale: Linear X scale: Linear

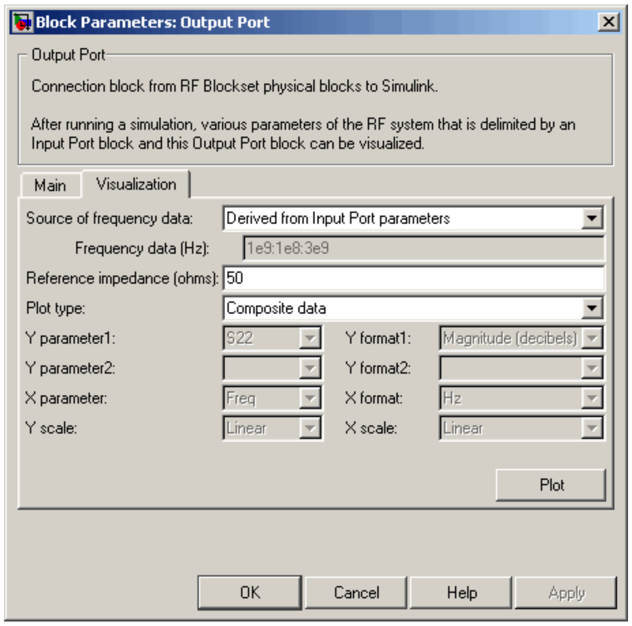
3 Click **Plot**.

This action creates a Polar plane plot of S_{22} as a function of frequency.



S_{22} versus Frequency for a Physical Subsystem

4 In the Output Port block dialog box, change the **Plot type** to `Composite data` to generate four plots in one figure. The parameters for the plots are defined by the block, so the **Y Parameter1**, **Y Parameter2**, and **X Parameter** fields becomes invisible.



Block Parameters: Output Port

Output Port

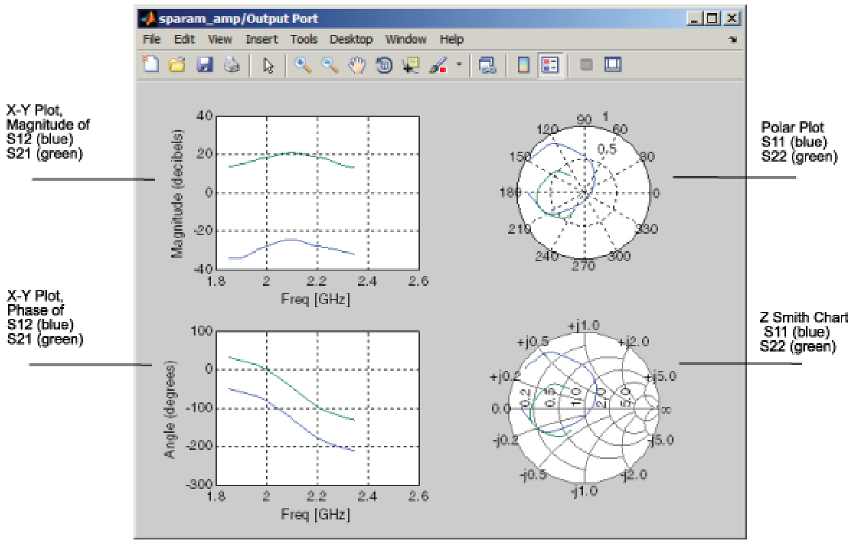
Connection block from RF Blockset physical blocks to Simulink.

After running a simulation, various parameters of the RF system that is delimited by an Input Port block and this Output Port block can be visualized.

Source of frequency data:	Derived from Input Port parameters		
Frequency data (Hz):	1e9:1e8:3e9		
Reference impedance (ohms):	50		
Plot type:	Composite data		
Y parameter1:	S22	Y format1:	Magnitude (decibels)
Y parameter2:		Y format2:	
X parameter:	Freq	X format:	Hz
Y scale:	Linear	X scale:	Linear

5 Click **Plot**.

This action creates a composite plot.


sparam_amp/Output Port
File Edit View Insert Tools Desktop Window Help
X-Y Plot, Magnitude of S12 (blue) S21 (green)
Magnitude (decibels)
Freq [GHz]
X-Y Plot, Phase of S12 (blue) S21 (green)
Angle (degrees)
Freq [GHz]
Polar Plot S11 (blue) S22 (green)
Z Smith Chart S11 (blue) S22 (green)

Composite Plot for a Physical Subsystem

See Also

More About

- “Modify Plots” on page 7-21
- “Create Plots” on page 7-2
- “Update Plots” on page 7-20

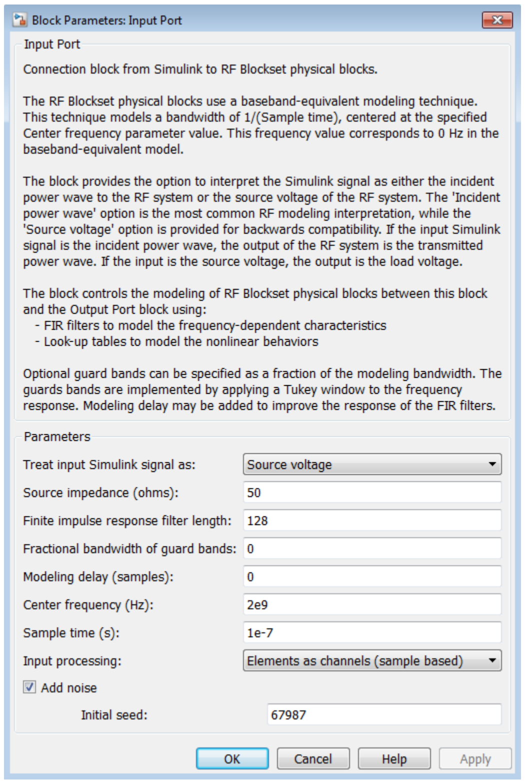
RF Blockset Equivalent Baseband Algorithms

Simulate an RF Model

When you simulate a model that contains physical blocks, RF Blockset Equivalent Baseband software determines the modeling frequencies of the physical system using the Input Port block parameters. The *modeling frequencies* are the frequencies at which the blockset takes information from the blocks to construct the baseband-equivalent model. Then, the software determines the block parameter values at those frequencies and uses the information to create a baseband-equivalent model for Simulink time-domain simulation.

Determine Modeling Frequencies

When you simulate an RF model, the Output Port block uses Input Port block parameters to determine the modeling frequencies *f* for the physical system that is bracketed between the Input Port block and the Output Port block. *f* is an *N*-element vector, where *N* is the finite impulse response filter length. The modeling frequencies are a function of the center frequency f_c and the sample time t_s. The following figure shows the Input Port block parameters that determine the modeling frequencies.



Block Parameters: Input Port

Input Port

Connection block from Simulink to RF Blockset physical blocks.

The RF Blockset physical blocks use a baseband-equivalent modeling technique. This technique models a bandwidth of 1/(Sample time), centered at the specified Center frequency parameter value. This frequency value corresponds to 0 Hz in the baseband-equivalent model.

The block provides the option to interpret the Simulink signal as either the incident power wave to the RF system or the source voltage of the RF system. The 'Incident power wave' option is the most common RF modeling interpretation, while the 'Source voltage' option is provided for backwards compatibility. If the input Simulink signal is the incident power wave, the output of the RF system is the transmitted power wave. If the input is the source voltage, the output is the load voltage.

The block controls the modeling of RF Blockset physical blocks between this block and the Output Port block using:
- FIR filters to model the frequency-dependent characteristics
- Look-up tables to model the nonlinear behaviors

Optional guard bands can be specified as a fraction of the modeling bandwidth. The guards bands are implemented by applying a Tukey window to the frequency response. Modeling delay may be added to improve the response of the FIR filters.

Parameters

Treat input Simulink signal as:	Source voltage
Source impedance (ohms):	50
Finite impulse response filter length:	128
Fractional bandwidth of guard bands:	0
Modeling delay (samples):	0
Center frequency (Hz):	2e9
Sample time (s):	1e-7
Input processing:	Elements as channels (sample based)

✓ Add noise

Initial seed:	67987

f_n is the *n*th element of the vector of modeling frequencies, *f*, and is given by

$$f_n = f_{\min} + \frac{n-1}{t_s N} \quad n = 1, \ldots, N$$

where

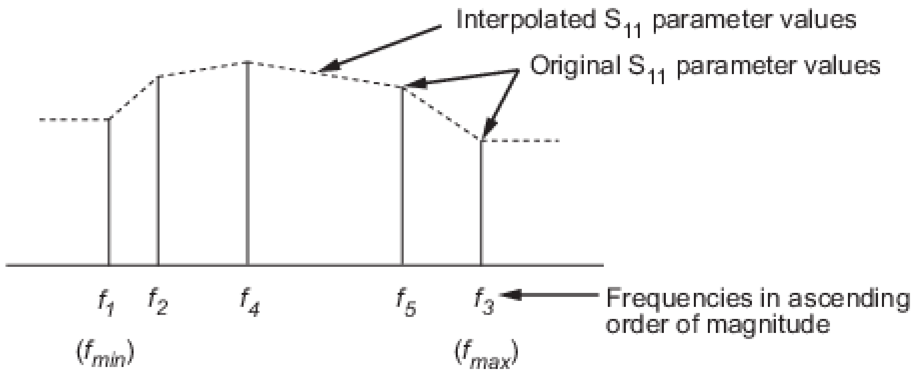
$$f_{\min} = f_c - \frac{1}{2t_s}$$

Map Network Parameters to Modeling Frequencies

In a physical system, each block provides network parameters at different frequencies. These frequencies are not necessarily the modeling frequencies for the physical system in which the block resides. To create a baseband-equivalent model, RF Blockset Equivalent Baseband software must calculate the values of the S-parameters at the modeling frequencies.

Individual physical blocks calculate the S-parameters at the modeling frequencies determined by the Input Port block parameters. Each block interpolates its S-parameters to determine the S-parameters at the modeling frequencies. If the block contains network Y- or Z-parameters, it first converts them to S-parameters.

Specifically, the block orders the S-parameters in the ascending order of their frequencies, f_n. Then, it interpolates the S-parameters using the MATLAB `interp1` function. For example, the curve in the following diagram illustrates the result of interpolating the S_{11} parameters at original frequencies f_1 through f_5.


Interpolated S_{11} parameter values
Original S_{11} parameter values
f_1
f_2
f_4
f_5
f_3
Frequencies in ascending order of magnitude
(f_{min})
(f_{max})

The **Interpolation method** field in the individual block dialog boxes enables you to specify the interpolation method as `Cubic`, `Linear` (default), or `Spline`. For more information about these methods, see the `interp1` reference page in the MATLAB documentation.

As shown in the previous diagram, each block uses the parameter values at f_{min} for all modeling frequencies smaller than f_{min}. The block uses the parameter values at f_{max} for all modeling frequencies greater than f_{max}. In both cases, the results may not be accurate, so you need to specify network parameter values over a range of frequencies that is wide enough to account for the block behavior.

Model Noise in an RF System

In this section...

The RF Blockset Equivalent Baseband Physical library blocks can model noise. The Input Port block parameters specify whether to include noise in a simulation. When you include noise information in your model, the blockset simulates the noise of the physical system by combining the noise contributions from each individual block. This section explains how the blockset simulates noise from user-specified information. For information on how to add noise to an RF model, see "Model Noise" on page 6-19.

Output-Referred Noise in RF Models

In general, you can specify output-referred noise in one of three ways:

- `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 the standard reference noise temperature of 290 K. In terms of noise factor

  Noise figure = 10log(Noise factor)

These three specifications are equivalent, because you can compute each one from any of the others.

The blockset lets you simulate the noise associated with any physical block in your RF model.

The blockset automatically determines the noise properties of passive blocks from their network parameters. The blockset gets these network parameters either explicitly from the block dialog box or specified data files, or implicitly by calculating them from the specified block parameters.

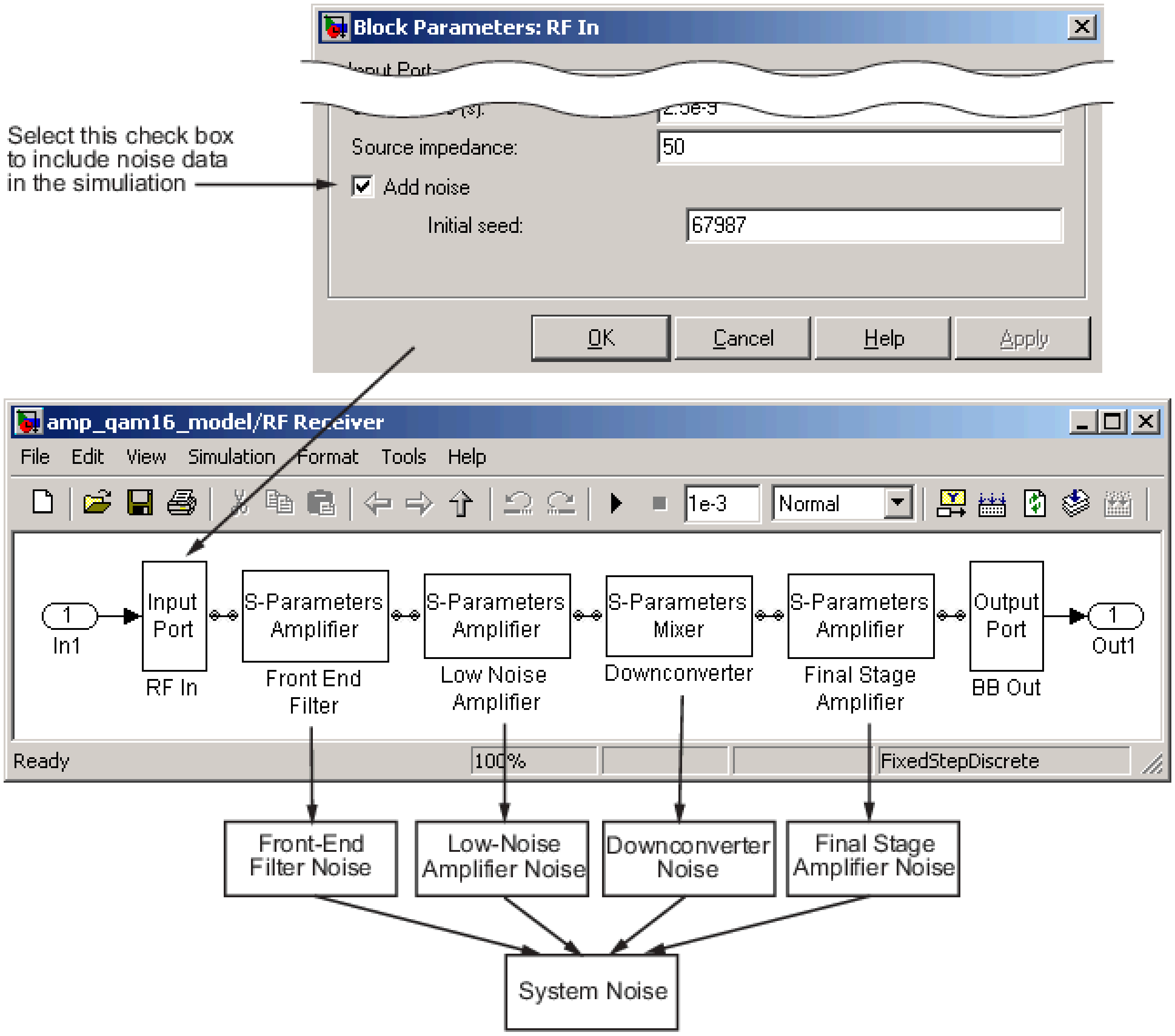
For active devices such as amplifiers and mixers, the noise properties cannot be inferred from network parameters. Therefore, for the amplifier and mixer blocks, you must specify the noise information explicitly, either in the dialog block or the associated data file.

For physical amplifier and mixer blocks, you can specify active block noise in one of the following ways:

- Spot noise data
- Frequency-independent noise figure, noise factor, or noise temperature values
- Frequency-dependent noise figure data (`rfdata.nf`) or spot noise data (`rfdata.noise`) object

These noise specification options are described in "Amplifier and Mixer Noise Specifications" on page 6-19.

When you run the simulation, the blockset first computes the noise figure values for each individual block at the modeling frequencies. Then, it computes the noise figure of the physical system from the individual noise figure values and uses the system noise figure information to calculate the output noise power. This process is shown in the following figure.



Calculate Noise Figure at Modeling Frequencies

To include noise information in a simulation, the blockset must compute the noise figure values of each individual block at the modeling frequencies.

If you specify the frequency-independent noise figure value directly, or if the blockset computes the noise figure value from the block resistance, the blockset uses this value for the noise figure value at each of the modeling frequencies.

If you specify the noise factor or noise temperature value, the blockset computes the noise figure value from the specified value using the equations in the preceding section and uses the computed value for the noise figure value at each of the modeling frequencies.

If you specify frequency-dependent noise figure values using an `rfdata.nf` object, the blockset interpolates the values using the **Interpolation method** specified in the block dialog box to get the noise figure value at each of the modeling frequencies.

If you specify spot noise data, the blockset computes frequency-dependent noise figure information from this data. It takes the minimum noise figure, NF_{min}, equivalent noise resistance, R_n, and optimal source admittance, Y_{opt}, values in the file and interpolates to find the values at the modeling frequencies. Then, the blockset uses the following equation to calculate the noise correlation matrix, C_A:

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

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

The blockset then calculates the noise factor, F, from the noise correlation matrix as follows:

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

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

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

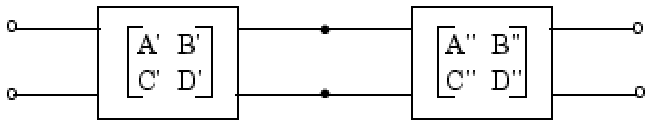
The blockset obtains the noise figure, NF, from the noise factor:

$$NF = 10\log(F)$$

Calculate System Noise Figure

The blockset uses a recursive process to calculate system noise figure. The noise correlation matrices for the first two elements of the cascade are combined into a single matrix, and the process is repeated.

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


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

First, the blockset calculates noise correlation matrices C_A' and C_A'' for the two networks. Then, the blockset combines C_A' and C_A'' into a single correlation matrix C_A using the equation

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

The ABCD-parameter matrices in the cascade combine according to matrix multiplication:

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

If there is another element in the cascade, the same calculations will be performed using these ABCD-parameters as well as the ABCD-parameters corresponding to the following element. The recursion will terminate with a noise correlation matrix pertaining to the entire system. The blockset then calculates the system noise figure from this matrix.

For more information about these calculation techniques, see the following article:

Hillbrand, H. and P.H. Russer, “An Efficient Method for Computer Aided Noise Analysis of Linear Amplifier Networks,” *IEEE Transactions on Circuits and Systems*, Vol. CAS-23, Number 4, pp. 235–238, 1976.

Calculate Output Noise Power

The blockset uses noise power to determine the amplitude of the noise that it adds to the physical system using a Gaussian distributed pseudorandom number generator. It uses both the noise temperature and the modeling bandwidth to calculate the noise power:

Noise power = kTB

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

The blockset computes noise temperature from the specified or calculated noise figure values for the system, and it computes the modeling bandwidth from the model's sample time and center frequency.

Create Complex Baseband-Equivalent Model

In this section...

Baseband-Equivalent Modeling

RF Blockset Equivalent Baseband software simulates the physical system in the time domain using a complex baseband-equivalent model that it creates from the passband frequency-domain parameters of the physical blocks. This type of modeling is also known as lowpass equivalent (LPE), complex envelope, or envelope modeling.

To create a complex baseband-equivalent model in the time domain based on the network parameters of the physical system, the blockset performs a mathematical transformation that consists of the following three steps:

1. "Calculate the Passband Transfer Function" on page A-9
2. "Calculate the Baseband-Equivalent Transfer Function" on page A-11
3. "Calculate the Baseband-Equivalent Impulse Response" on page A-11

Calculate the Passband Transfer Function

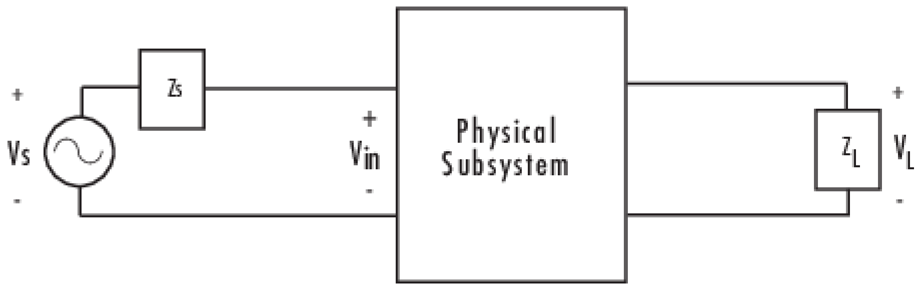
The blockset calculates the passband transfer function from the physical block parameters at the modeling frequencies by calculating the transfer function of the physical subsystem and then applying the Tukey window to obtain the passband transfer function.

Note To learn how the blockset uses the specified network parameters to compute the network parameters at the modeling frequencies, see "Map Network Parameters to Modeling Frequencies" on page A-4.

The transfer function of the physical subsystem is defined as:

$$H(f) = \frac{V_L(f)}{V_S(f)}$$

where V_S and V_L are the source and load voltages shown in the following figure, 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

$$\Gamma_l = \frac{Z_l - Z_o}{Z_l + Z_o}$$

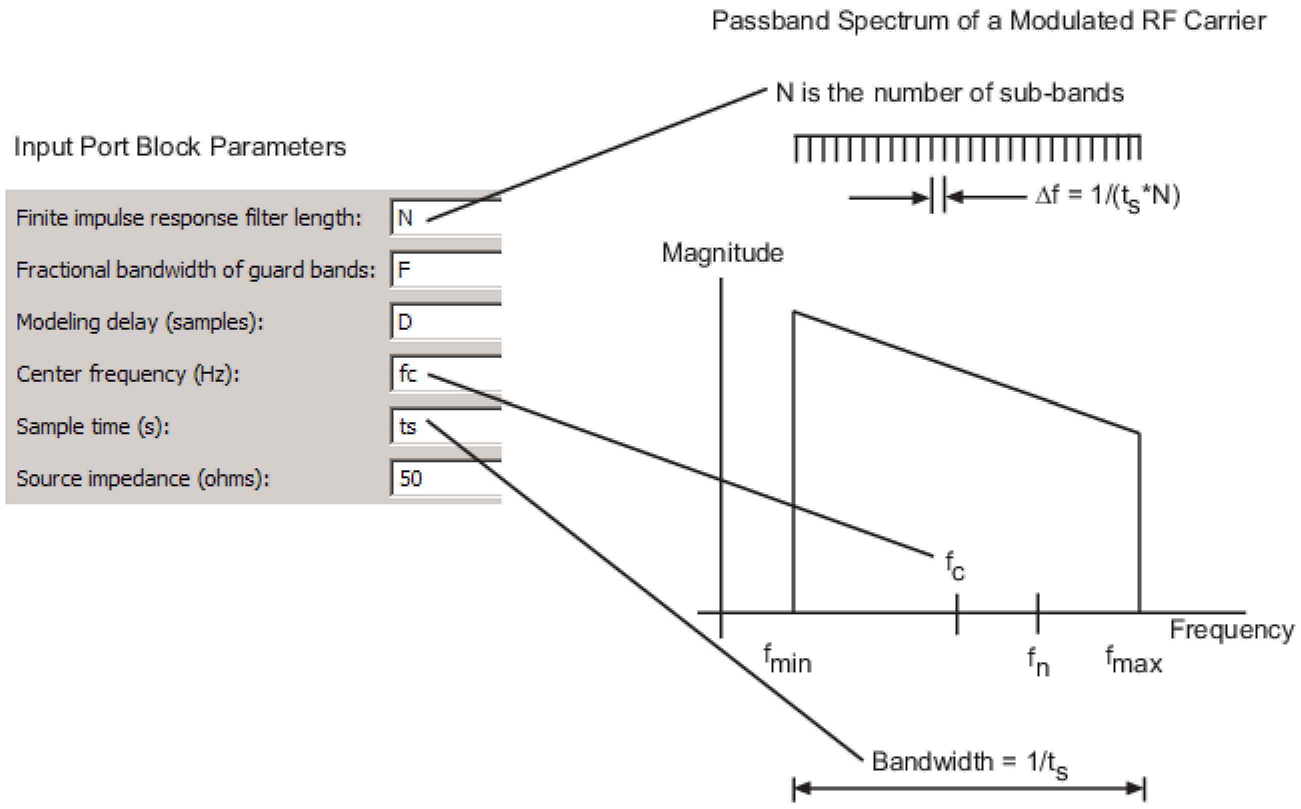
$$\Gamma_s = \frac{Z_s - Z_o}{Z_s + Z_o}$$

$$\Gamma_{in} = S_{11} + \left(S_{12}S_{21}\frac{\Gamma_l}{(1 - S_{22}\Gamma_l)}\right)$$

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 transfer function of the physical subsystem from the Input Port block parameters as shown in the following figure.


Passband Spectrum of a Modulated RF Carrier
Input Port Block Parameters
Finite impulse response filter length: N
Fractional bandwidth of guard bands: F
Modeling delay (samples): D
Center frequency (Hz): fc
Sample time (s): ts
Source impedance (ohms): 50
N is the number of sub-bands
Δf = 1/(ts*N)
Magnitude
fc
Frequency
fmin
fn
fmax
Bandwidth = 1/ts

The blockset then applies the Tukey window to obtain the passband transfer function:

$$H_{passband}(f) = H(f) \cdot tukeywin(N, F)$$

where `tukeywin` is the Signal Processing Toolbox™ `tukeywin` function.

Calculate the Baseband-Equivalent Transfer Function

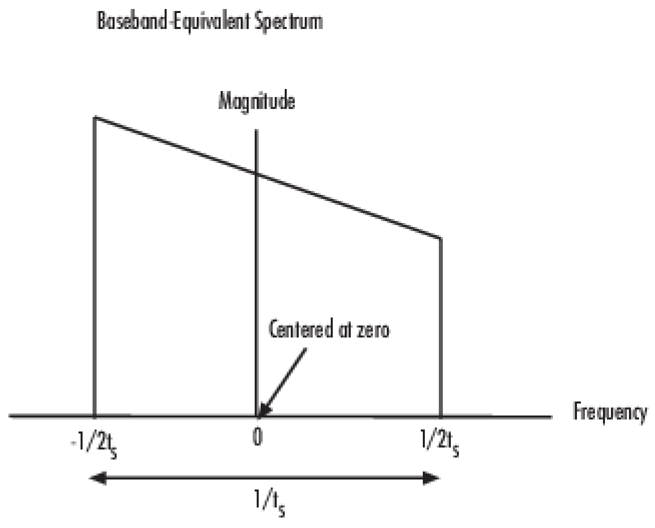
The blockset calculates the baseband transfer function, $H_{baseband}(f)$, by translating the passband transfer function to its equivalent baseband transfer function:

$$H_{baseband}(f) = H_{passband}(f + f_c)$$

where f_c is the specified center frequency.

The resulting baseband-equivalent spectrum is centered at zero, so the blockset can simulate the system using a much larger time step than Simulink can use for the same system. For information on why this translation allows for a larger time step, see “Simulation Efficiency of a Baseband-Equivalent Model” on page A-12.

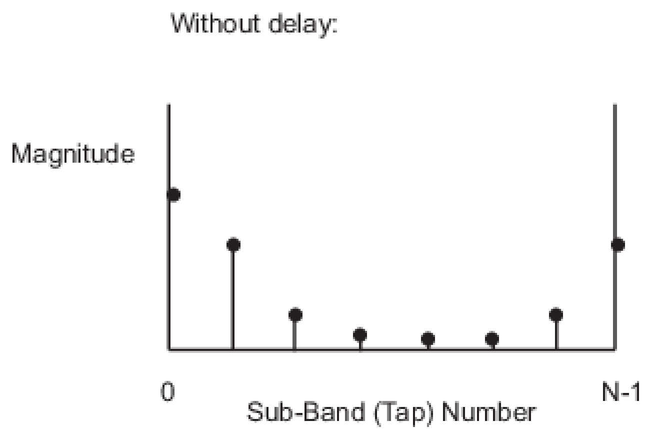
The baseband transfer function is shown in the following figure.

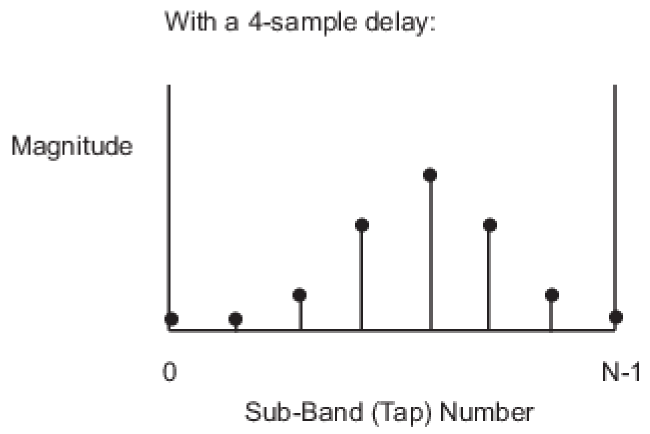


Calculate the Baseband-Equivalent Impulse Response

The blockset calculates the baseband-equivalent impulse response by performing the following steps:

1. Calculate 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. When the finite impulse response is truncated to the length specified by the user, the effect of the truncation is similar to windowing with a rectangular window.
2. Apply the delay specified by the **Modeling delay (samples)** parameter in the Input Port block dialog box. Selecting an appropriate value for this delay ensures that the baseband-equivalent model has a causal response by moving the time window such that the model energy is concentrated at the center of the window, as shown in the following figure:





Simulation Efficiency of a Baseband-Equivalent Model

The baseband-equivalent modeling technique improves simulation speed by allowing the simulator to take larger time steps. To simulate a system in the time domain, Simulink would require a step size of:

$$t_{step} = \frac{1}{2f_{\max}}$$

Using the baseband-equivalent model of the same system, whose spectrum has been shifted down by f_c, allows for a much larger time step of:

$$t_{step} = \frac{1}{2(f_{\max} - f_c)} = \frac{1}{f_{\max} - f_{\min}}$$

Example — Select Parameter Values for a Baseband-Equivalent Model

- "Baseband-Equivalent Modeling Example Overview" on page A-13
- "Create the Model" on page A-13
- "Specify Model Parameters" on page A-15

- “Run the Simulation and Analyze the Results” on page A-19
- “Reducing Acausal Response in the Baseband-Equivalent Model” on page A-19
- “Introduce Delay into the Baseband-Equivalent Model” on page A-20

Baseband-Equivalent Modeling Example Overview

In this example, you model an RF transmission line stimulated by a pulse and plot the baseband-equivalent model that the blockset uses to simulate the transmission line in the time domain. You compare the effects of using different parameter values for the baseband-equivalent model. This example helps you understand how to use these parameters to best apply the baseband-equivalent modeling paradigm of performing time-domain simulation using a limited band of frequency data.

Create the Model

In this part of the example, you perform the following tasks:

- “Select Blocks to Represent System Components” on page A-13
- “Build the Model” on page A-13
- “Specify Model Variables” on page A-14

Select Blocks to Represent System Components

In this part of the example, you select the blocks to represent:

- The input signal
- The RF transmission line
- The baseband-equivalent model display

The following table lists the blocks that represent the system components and a description of the role of each block.

Block	Description
Discrete Impulse	Generates a frame-based pulse input signal.
Real-Imag to Complex	Converts the real pulse signal into a complex pulse signal.
Input Port	Establishes parameters that are common to all blocks in the RF transmission line subsystem, including the source impedance of the subsystem that is used to convert Simulink signals to the physical modeling environment.
RLCG Transmission Line	Models the signal attenuation caused by an RF transmission line.
Output Port	Establishes parameters that are common to all blocks in the RF transmission line subsystem. These parameters include the load impedance of the subsystem, which is used to convert RF signals to Simulink signals.
Complex to Magnitude-Angle	Converts the complex signal from the Output Port block into magnitude-angle format.

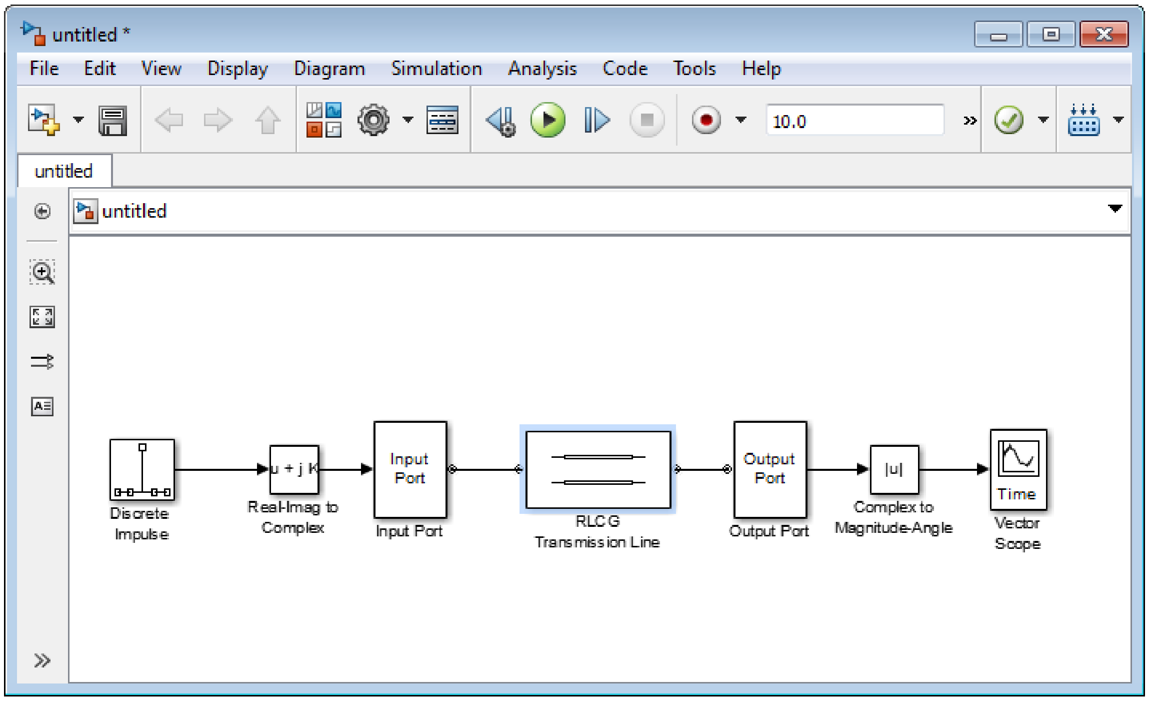
Build the Model

In this part of the example, you create a Simulink model, add blocks to the model, and connect the blocks.

1 Create a model.
2 Add to the model the blocks shown in the following table. The Library Path column of the table specifies the hierarchical path to each block.

Block	Library Path	Quantity
Discrete Impulse	**DSP System Toolbox > Sources**	1
Real-Imag to Complex	**Simulink > Math Operations**	1
Input Port	**RF Blockset > Equivalent Baseband > Input/Output Ports**	1
RLCG Transmission Line	**RF Blockset > Equivalent Baseband > Transmission Lines**	1
Output Port	**RF Blockset > Equivalent Baseband > Input/Output Ports**	1
Complex to Magnitude-Angle	**Simulink > Math Operations**	1

3 Connect the blocks as shown in the following figure.



Now you are ready to specify model variables.

Specify Model Variables

Type the following at the MATLAB prompt to set up workspace variables for the model:

```
t_s = 5e-10;   % Sample time
f_c = 3e9;         % Center frequency
taps = 64;     % Filter length
```

Now you are ready to specify the block parameters.

Specify Model Parameters

In this part of the example, you specify the following parameters to represent the behavior of the system components:

- "Input Signal Parameters" on page A-15
- "Transmission Line Subsystem Parameters" on page A-16
- "Signal Display Parameters" on page A-18

Input Signal Parameters

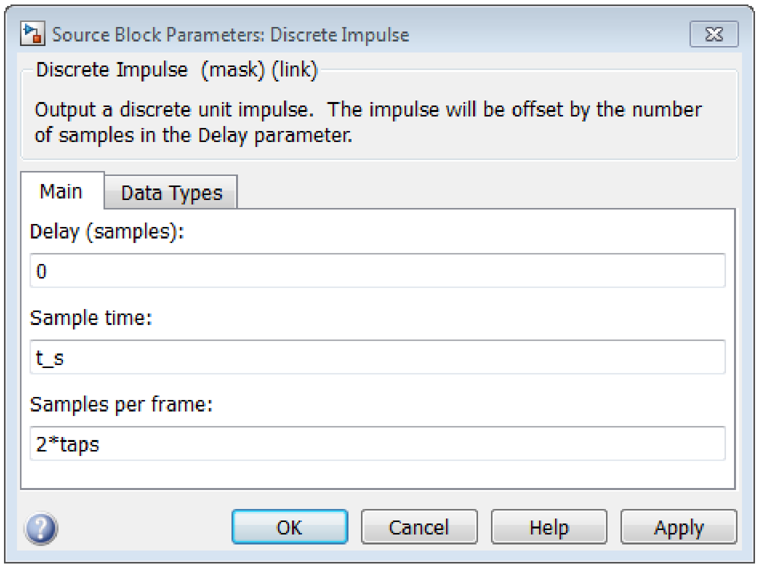
You generate the frame-based complex pulse source signal using two blocks:

- The Discrete Impulse block generates a real pulse signal.
- The Real-Imag to Complex block converts the real signal to a complex signal.

Note All signals in the RF model must be complex to match the signals in the physical subsystem, so you create a complex input signal.

1 In the Discrete Impulse block parameters dialog box:

   - Set **Sample time** to `t_s`.
   - Set **Samples per frame** to `2*taps`.


Source Block Parameters: Discrete Impulse

Discrete Impulse (mask) (link)

Output a discrete unit impulse. The impulse will be offset by the number of samples in the Delay parameter.

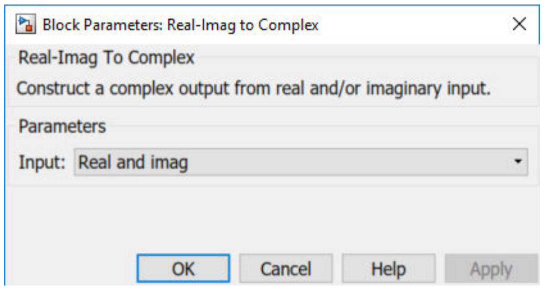
Main | Data Types

Delay (samples):
0

Sample time:
t_s

Samples per frame:
2*taps

2 Set the Real-Imag to Complex block **Input** parameter to `Real`. Changing this parameter changes the number of block inputs from two to one, making the block fully connected.


Block Parameters: Real-Imag to Complex

Real-Imag To Complex

Construct a complex output from real and/or imaginary input.

Parameters

Input: Real and imag

Transmission Line Subsystem Parameters

In this part of the example, you configure the blocks that model the RF filter subsystem—the Input Port, Transmission Line, and Output Port blocks.

1 In the Input Port block parameters dialog box:

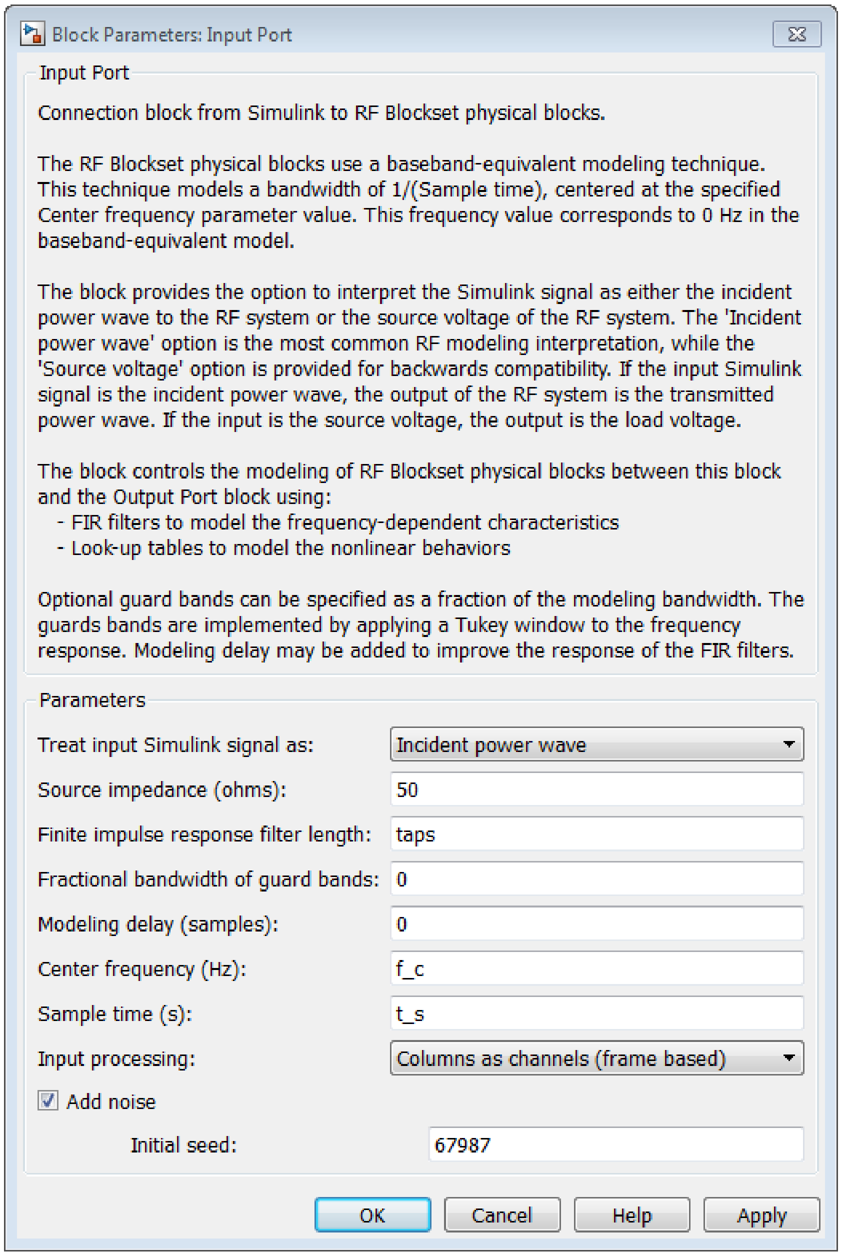
- Set **Treat input Simulink signal as** to `Incident power wave`.

  This option tells the blockset to interpret the input signal as the incident power wave to the RF subsystem, and not the source voltage of the RF subsystem.

  Note If you use the default value for this parameter, the software interprets the input Simulink signal as the source voltage. As a result, the source and the load that model the Input Port and Output Port blocks, respectively, introduce 6 dB of loss into the physical system at all frequencies. For more information on why this loss occurs, see the note in “Convert to and from Simulink Signals” on page A-22.

- Set **Finite impulse response filter length** to `taps`.
- Set **Center frequency** to `f_c`.
- Set **Sample time (s)** to `t_s`.

  This sample time is equivalent to a modeling bandwidth of 1/`t_s` seconds.
- Set **Input Processing** to `Columns as channels (frame based)`.



Block Parameters: Input Port

Input Port

Connection block from Simulink to RF Blockset physical blocks.

The RF Blockset physical blocks use a baseband-equivalent modeling technique. This technique models a bandwidth of 1/(Sample time), centered at the specified Center frequency parameter value. This frequency value corresponds to 0 Hz in the baseband-equivalent model.

The block provides the option to interpret the Simulink signal as either the incident power wave to the RF system or the source voltage of the RF system. The 'Incident power wave' option is the most common RF modeling interpretation, while the 'Source voltage' option is provided for backwards compatibility. If the input Simulink signal is the incident power wave, the output of the RF system is the transmitted power wave. If the input is the source voltage, the output is the load voltage.

The block controls the modeling of RF Blockset physical blocks between this block and the Output Port block using:
- FIR filters to model the frequency-dependent characteristics
- Look-up tables to model the nonlinear behaviors

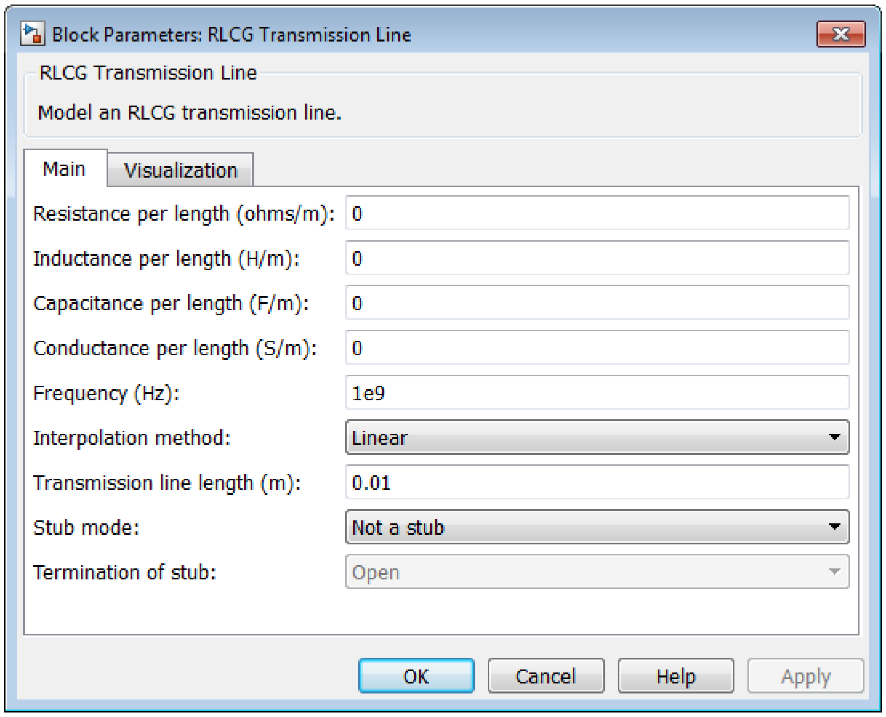
Optional guard bands can be specified as a fraction of the modeling bandwidth. The guards bands are implemented by applying a Tukey window to the frequency response. Modeling delay may be added to improve the response of the FIR filters.

Parameters

Treat input Simulink signal as:	Incident power wave
Source impedance (ohms):	50
Finite impulse response filter length:	taps
Fractional bandwidth of guard bands:	0
Modeling delay (samples):	0
Center frequency (Hz):	f_c
Sample time (s):	t_s
Input processing:	Columns as channels (frame based)
✓ Add noise	
Initial seed:	67987

2 In the RLCG Transmission Line block parameters dialog box:

- Set **Inductance per length (H/m)** to `50`.
- Set **Capacitance per length (F/m)** to `.02`.
- Set **Frequency (Hz)** to `f_c`.
- Set **Transmission line length (m)** to `0.5*t_s`.


Block Parameters: RLCG Transmission Line

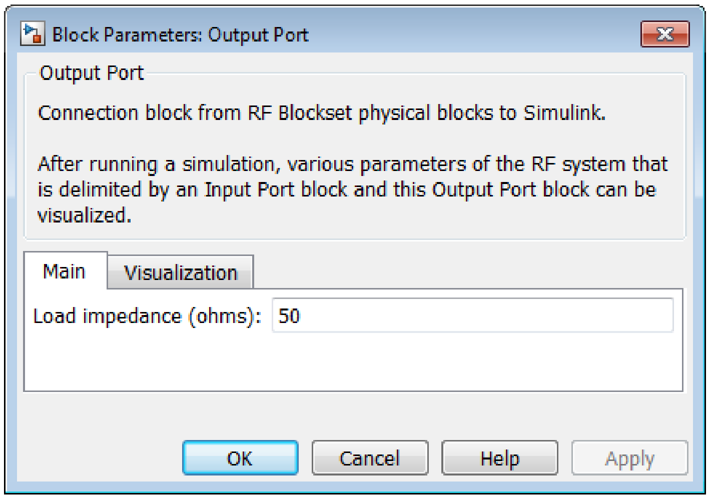
RLCG Transmission Line

Model an RLCG transmission line.

Main | Visualization

Resistance per length (ohms/m):	0
Inductance per length (H/m):	0
Capacitance per length (F/m):	0
Conductance per length (S/m):	0
Frequency (Hz):	1e9
Interpolation method:	Linear
Transmission line length (m):	0.01
Stub mode:	Not a stub
Termination of stub:	Open

3 Accept the default parameters for the Output Port block to use a load impedance of 50 ohms.


Block Parameters: Output Port

Output Port

Connection block from RF Blockset physical blocks to Simulink.

After running a simulation, various parameters of the RF system that is delimited by an Input Port block and this Output Port block can be visualized.

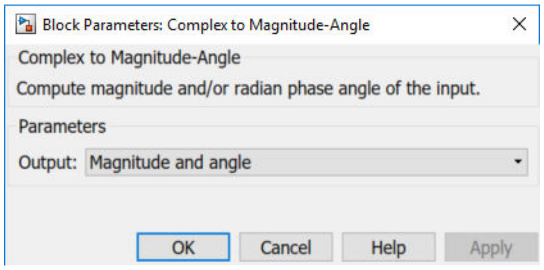
Main | Visualization

Load impedance (ohms): 50

Signal Display Parameters

In this part of the example, you specify the parameters that set up the baseband-equivalent model display. You use the Complex to Magnitude-Angle block to convert the RF subsystem output to magnitude format.

1 Set the Complex to Magnitude-Angle block **Output** parameter to `Magnitude`. Changing this parameter changes the number of block outputs from two to one, making the block fully connected.


Block Parameters: Complex to Magnitude-Angle

Complex to Magnitude-Angle

Compute magnitude and/or radian phase angle of the input.

Parameters

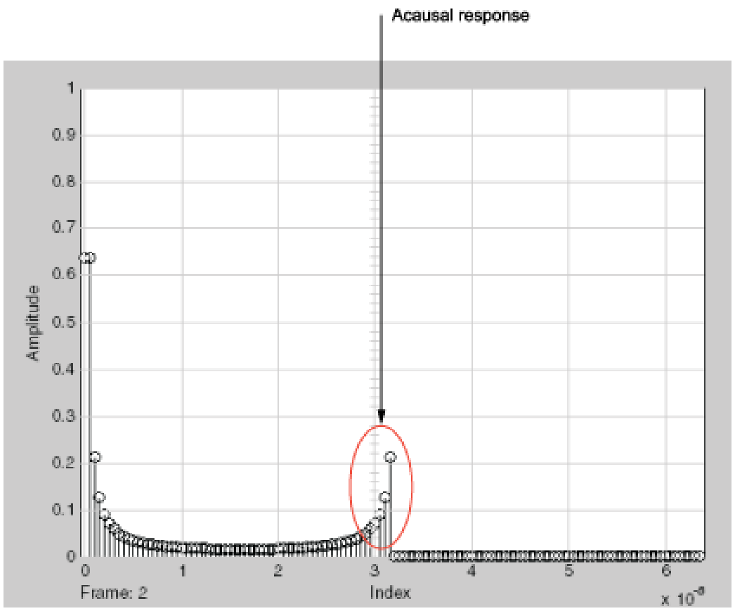
Output: Magnitude and angle

Run the Simulation and Analyze the Results

Before you run the simulation, set the stop time. Click **Simulation** In the **PREPARE**, click **Model Settings** in **Configuration and Simulation**. Enter `2*t_s*(taps-1)` for the **Stop time** parameter.

To run the simulation, click **Run** in the model window.

This window appears automatically when you start the simulation. The following plot shows the baseband-equivalent model, which contains a significant amount of acausal energy because of the limited bandwidth of the model.



Baseband-Equivalent Model

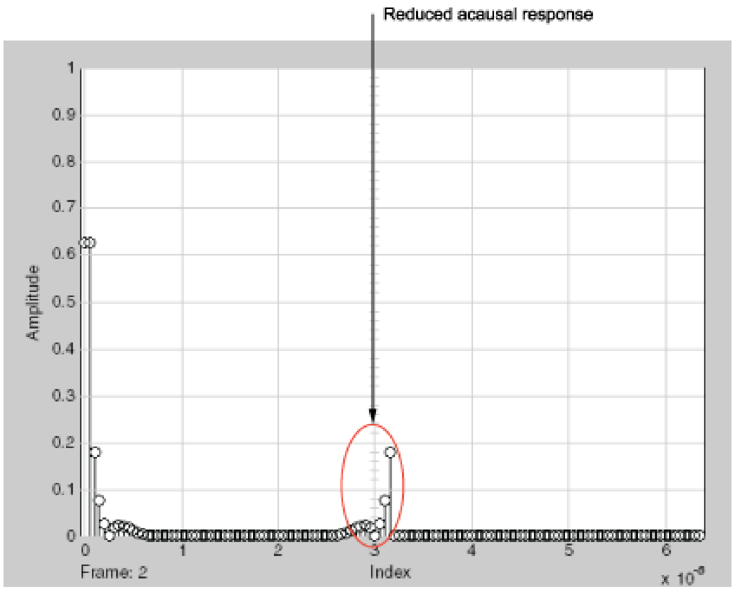
The next part of the example shows you how to reduce this acausal response.

Reducing Acausal Response in the Baseband-Equivalent Model

In this part of the example, you adjust the **Fractional bandwidth of guard bands** parameter. This parameter controls the shaping of the filter that the blockset applies to create the baseband-equivalent model.

1. Set the Input Port **Fractional bandwidth of guard bands** parameter to `0.2`.
2. Rerun the simulation.

You can see that the acausal response is lower than it was for the previous simulation, but there is still some energy wrapping around the end of the model because it is periodic.



Baseband-Equivalent Model with Filter Shaping

Note You can further reduce the acausal response in the baseband-equivalent model by increasing the value of the **Fractional bandwidth of guard bands** parameter above 0.2, but if you use a high value, you compromise the fidelity of the gain of the transmission line.

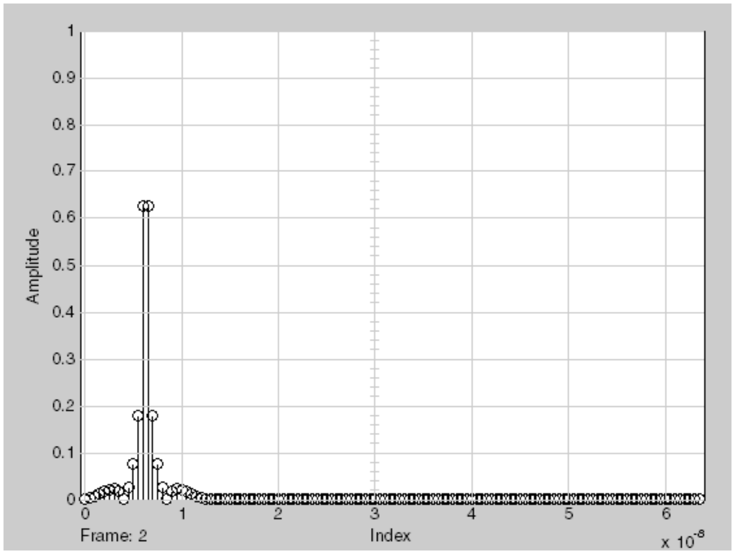
The next section shows you how to shift the response to avoid this wrapping.

Introduce Delay into the Baseband-Equivalent Model

In this part of the example, you adjust the **Modeling delay (samples)** parameter. This parameter controls the delay the blockset applies to create the baseband-equivalent model.

1 Set the Input Port **Modeling delay (samples)** parameter to 12.
2 Rerun the simulation.

The response of the baseband-equivalent model is concentrated in a small time window. This model provides the most accurate time-domain simulation of the specified band of frequency data.



Baseband-Equivalent Model with Filter Shaping and Delay

See Also

Input Port | Output Port | RLCG Transmission Line

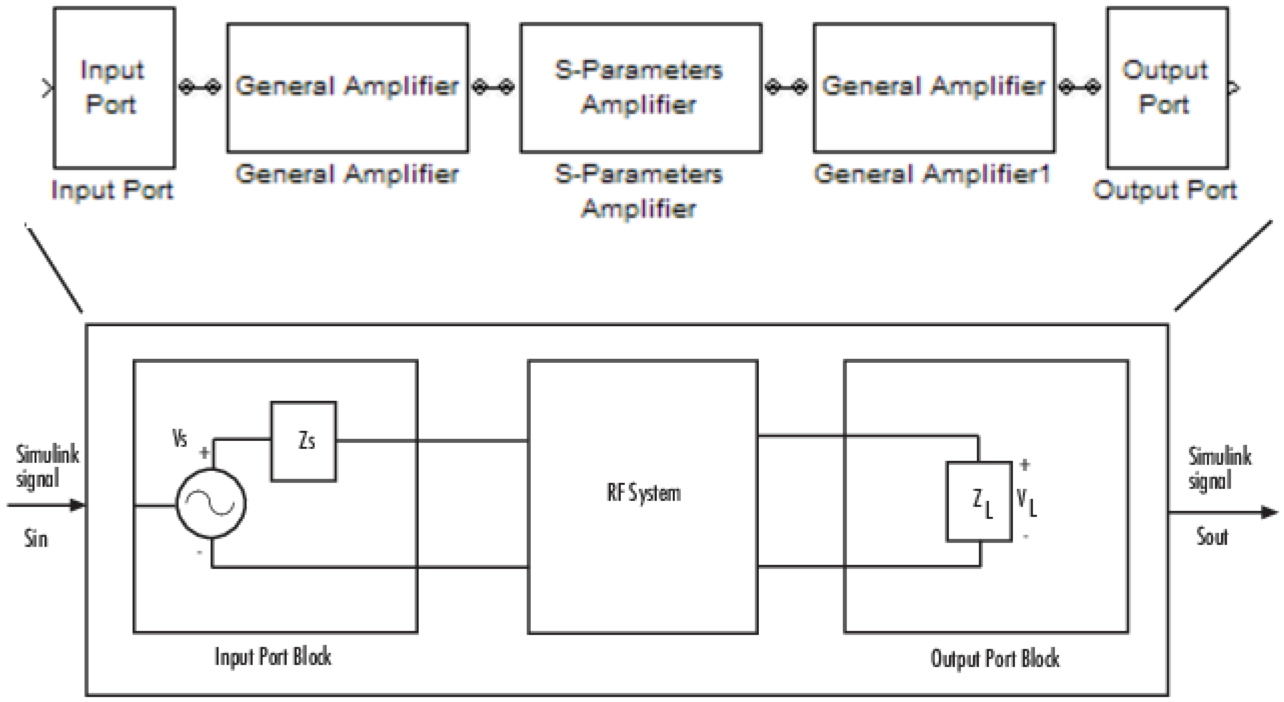
More About

- “Model Nonlinearity” on page 6-17
- “Convert to and from Simulink Signals” on page A-22

Convert to and from Simulink Signals

Signal Conversion Specifications

When you simulate an RF model, the blockset must convert the mathematical Simulink signals to and from the physical modeling environment. The following figure shows the signals involved in the conversion.



Where:

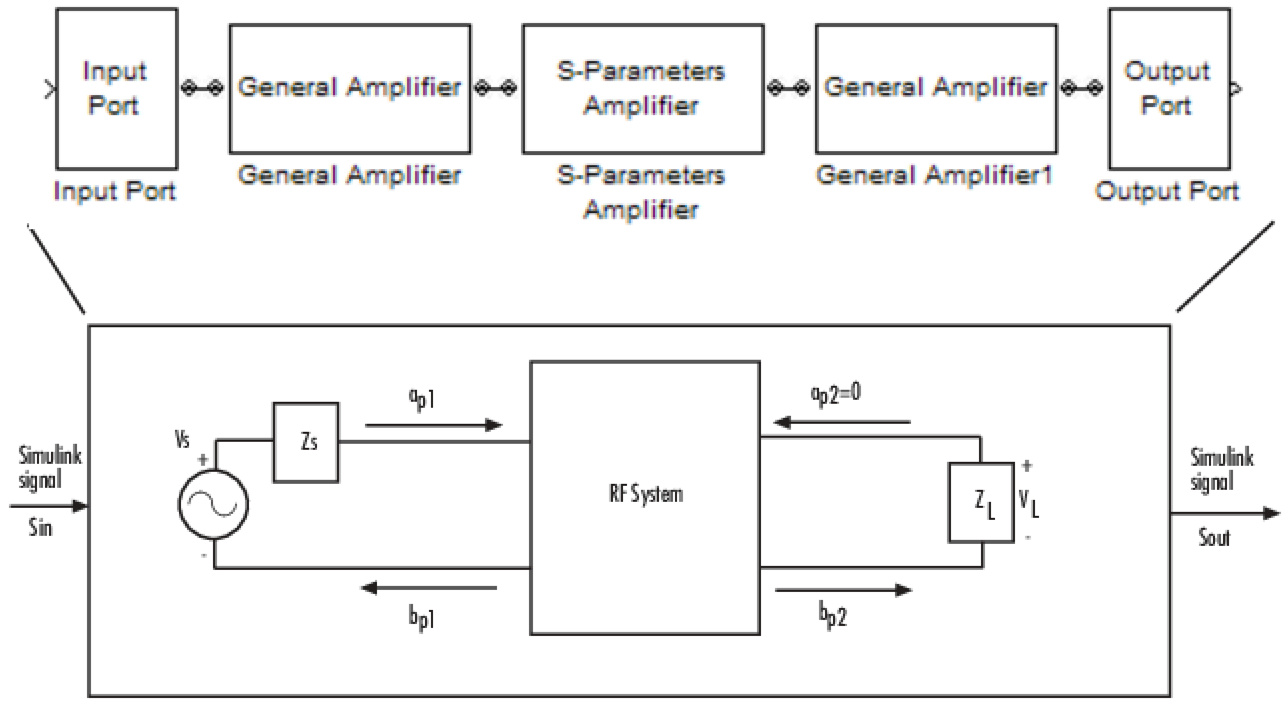
- Z_S is the **Source impedance (ohms)** parameter of the Input Port block.
- Z_L is the **Load impedance (ohms)** parameter of the Output Port block.

There are two options for interpreting the Simulink signal that enters the Input Port block:

- S_{in} is the incident power wave. For more information about this option, see “Interpret Simulink Signals as Incident Power Waves” on page A-22.
- S_{in} is the source voltage. For more information about this option, see “Interpret Simulink Signals as Source Voltages” on page A-24.

Interpret Simulink Signals as Incident Power Waves

The blockset provides the option to interpret the input Simulink signal, S_{in}, as the incident power wave, a_{p1}, at the first port of the RF system. The following figure shows the model for this interpretation.



In the figure, b_{p2} is the transmitted power wave at the second port of the RF system. This is the signal at the output of the Output Port block, S_{out}.

For a 2-port RF system, the incident and transmitted power waves are defined as:

$$a_{p1} = \frac{V_S}{2\sqrt{R_S}}$$

$$b_{p2} = \frac{\sqrt{R_L}}{Z_L}V_L$$

where:

- Z_S, the **Source impedance (ohms)** parameter of the Input Port block, is defined as:

  $Z_S = R_S + jX_S$
- Z_L, the **Load impedance (ohms)** parameter of the Output Port block, is defined as:

  $Z_L = R_L + jX_L$

Solving the power wave equations for S_{in} and S_{out} gives the following relationships:

$$S_{in} = \frac{V_S}{2\sqrt{R_S}}$$

$$S_{out} = \frac{\sqrt{R_L}}{Z_L}V_L$$

The implications of this interpretation are:

- $|S_{in}|^2$ is equal to the power available from the source, P_{avs}.
- $|S_{out}|^2$ is equal to the power delivered to the load, P_{out}.

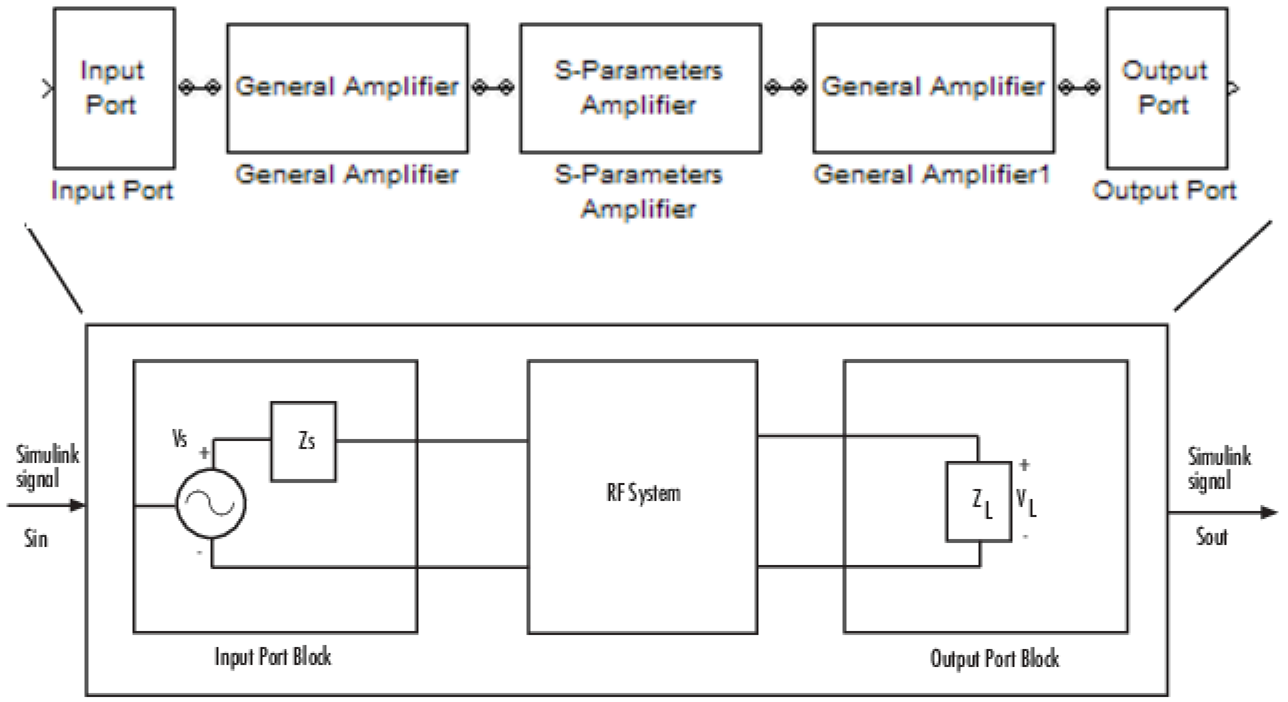
For a linear RF system, $P_{out} = G_t P_{avs}$ where G_t is the transducer power gain. As a result, the Simulink signals at the input and output of the RF system have the following relationship:

$$|S_{out}|^2 = G_t|S_{in}|^2$$

Note You can plot G_t from the Output Port block's **Visualization** tab.

Interpret Simulink Signals as Source Voltages

The blockset provides the option to interpret the input Simulink signal, S_{in}, as the source voltage, V_{S}, of the RF system. The following figure shows the model for this interpretation.



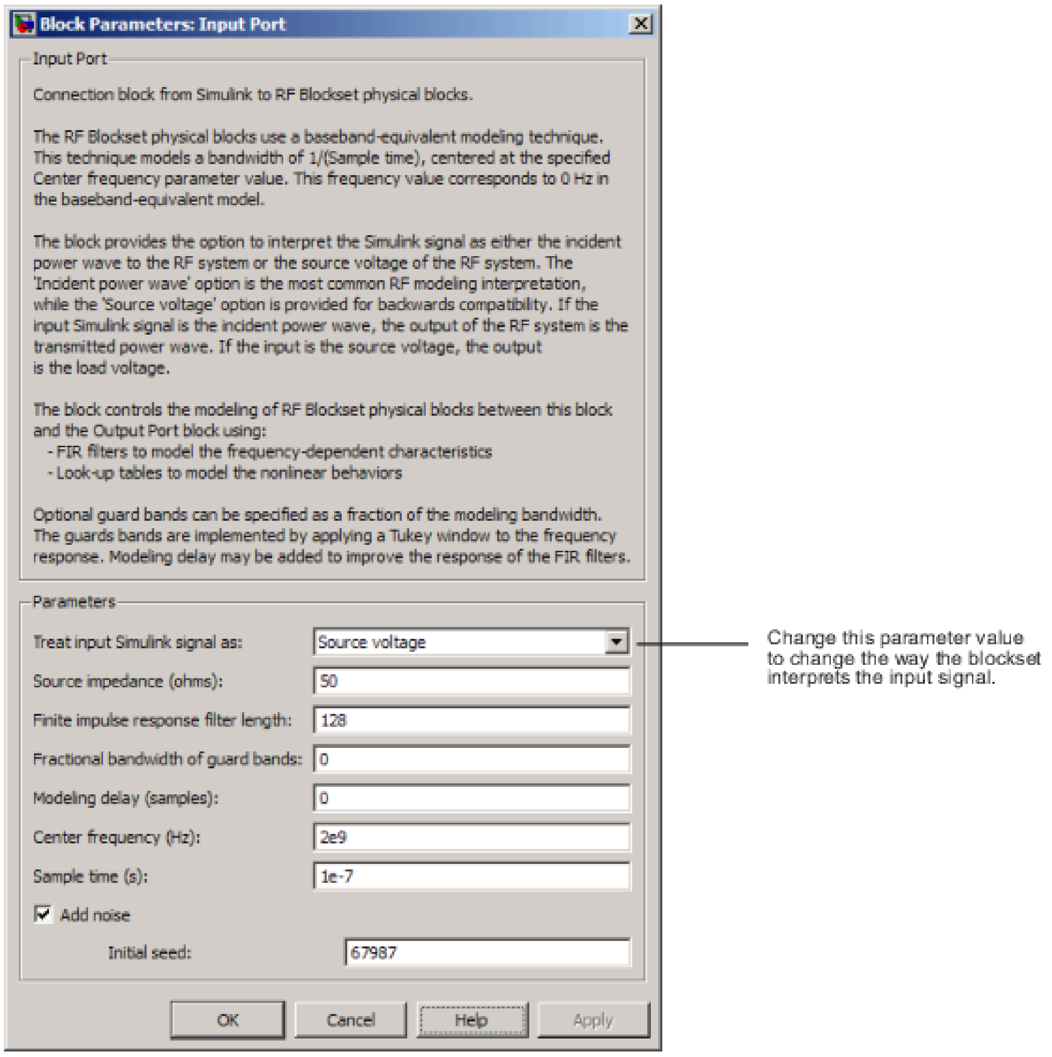
With this interpretation, the signal at the output of the Output Port block is the load voltage, V_L.

The blockset interpretation of the input Simulink signal as the source voltage, V_{S}, produces different results than the interpretation where the input Simulink signal is the incident power wave. When the source and load impedances are the same and real, the former interpretation produces 6 dB of loss compared to the latter.

Specify Input Signal Conversions

To specify the way in which the blockset interprets the input Simulink signal, you change the value of the **Treat input Simulink signal as** parameter in the Input Port dialog box. The available parameter values are:

- `Incident power wave` — Interpret the input signal as the incident power wave.
- `Source voltage` — Interpret the input signal as the source voltage.


Block Parameters: Input Port

Input Port

Connection block from Simulink to RF Blockset physical blocks.

The RF Blockset physical blocks use a baseband-equivalent modeling technique. This technique models a bandwidth of 1/(Sample time), centered at the specified Center frequency parameter value. This frequency value corresponds to 0 Hz in the baseband-equivalent model.

The block provides the option to interpret the Simulink signal as either the incident power wave to the RF system or the source voltage of the RF system. The 'Incident power wave' option is the most common RF modeling interpretation, while the 'Source voltage' option is provided for backwards compatibility. If the input Simulink signal is the incident power wave, the output of the RF system is the transmitted power wave. If the input is the source voltage, the output is the load voltage.

The block controls the modeling of RF Blockset physical blocks between this block and the Output Port block using:
- FIR filters to model the frequency-dependent characteristics
- Look-up tables to model the nonlinear behaviors

Optional guard bands can be specified as a fraction of the modeling bandwidth. The guards bands are implemented by applying a Tukey window to the frequency response. Modeling delay may be added to improve the response of the FIR filters.

Parameters

Treat input Simulink signal as:	Source voltage
Source impedance (ohms):	50
Finite impulse response filter length:	128
Fractional bandwidth of guard bands:	0
Modeling delay (samples):	0
Center frequency (Hz):	2e9
Sample time (s):	1e-7
[✓] Add noise	
Initial seed:	67987

OK
Cancel
Help
Apply
Change this parameter value to change the way the blockset interprets the input signal.

See Also

General Amplifier | Input Port | Output Port | S-Parameters Amplifier

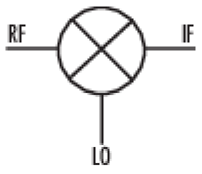
More About

- “Model RF Components” on page 6-2

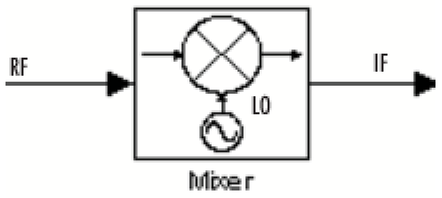
Model Mixers

2-Port Mixer Blocks

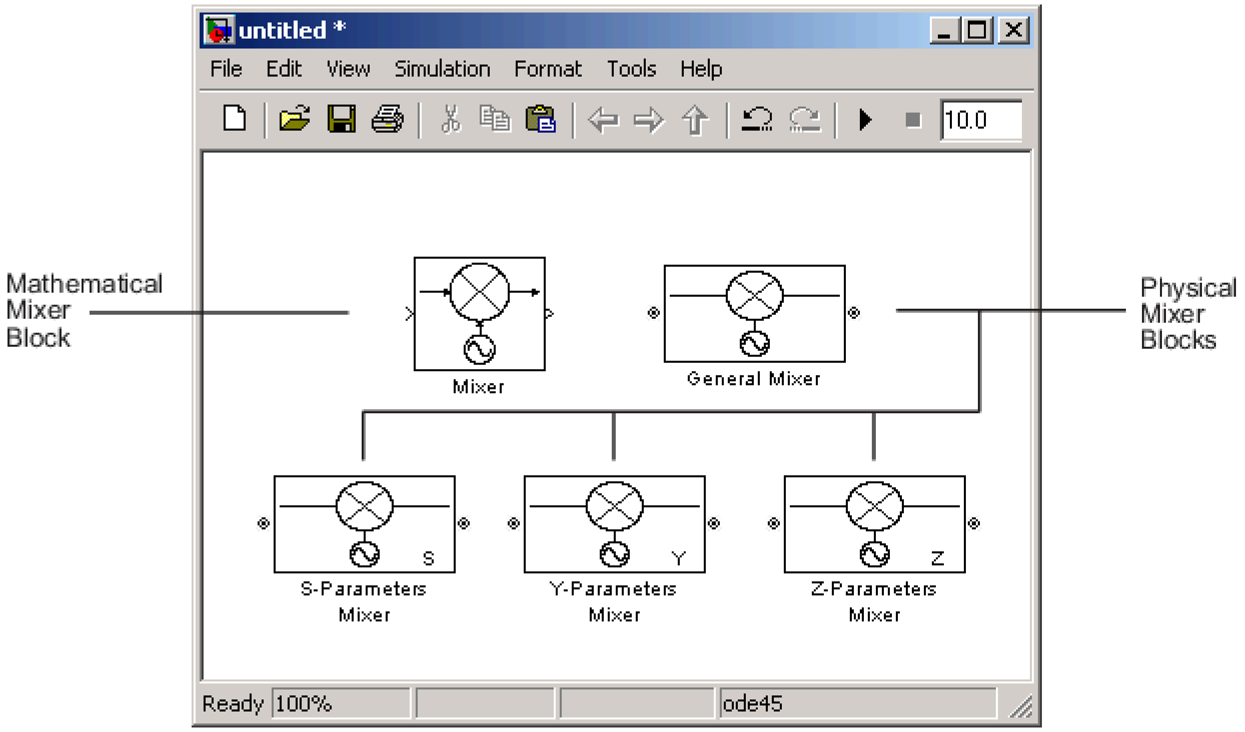
Typically, the block diagram of a mixer has three ports, as shown in the following representation of a downconversion mixer.



The mathematical and physical mixer blocks model both a mixer and a local oscillator, so they have only two ports.



The following figure shows the icons of the mixer blocks. The icons of all the mixer blocks show the internal local oscillator.



For the physical blocks, you can use the **LO frequency (Hz)** parameter to specify the local oscillator's frequency.

For more information, see the individual block reference pages.

Model a Mixer Chain

RF Blockset Equivalent Baseband software uses a baseband equivalent model to simulate RF components in the time domain. The blockset only models a band of frequencies around the carrier frequency of each component; the frequency band is determined by the following parameters of the corresponding Input Port block:

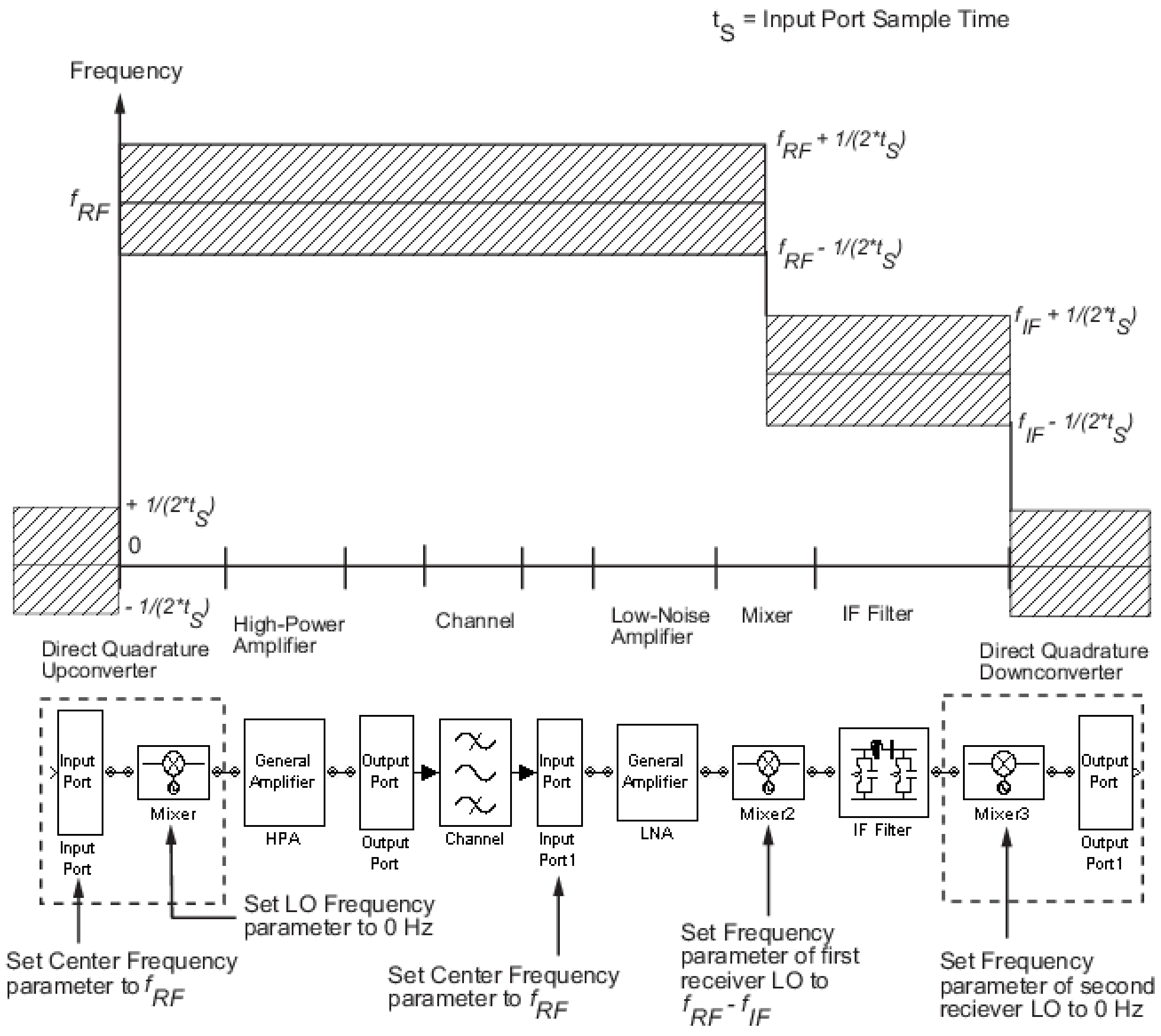
- Reciprocal of the **Sample time (s)**
- **Center frequency (Hz)**

When a mixer is present in a physical subsystem, it shifts the carrier frequency of the signal. This shift affects the frequencies that are used to create baseband equivalent model.

To illustrate how the mixer works, consider a typical RF mixer chain that consists of the following components:

- Direct Quadrature Upconverter
- High-Power Amplifier
- Channel
- Low-Noise Amplifier
- Downconverting Mixer
- IF Filter
- Direct Quadrature Downconverter

The following diagram shows these components and the band of frequencies that are simulated for each component. The signals at the input and output of the cascade are baseband complex. For the cascade, the diagram shows the real passband frequencies that are used to create the baseband-equivalent model, which is centered at zero. For a detailed explanation of how to use the blockset to model quadrature mixers, see “Quadrature Mixers” on page B-6.


t_S = Input Port Sample Time
Frequency
f_RF
f_RF + 1/(2*t_S)
f_RF - 1/(2*t_S)
f_IF + 1/(2*t_S)
f_IF - 1/(2*t_S)
+ 1/(2*t_S)
0
- 1/(2*t_S)
Direct Quadrature Upconverter
High-Power Amplifier
Channel
Low-Noise Amplifier
Mixer
IF Filter
Direct Quadrature Downconverter
Input Port
Mixer
General Amplifier
HPA
Output Port
Channel
Input Port1
LNA
Mixer2
IF Filter
Mixer3
Output Port1
Set LO Frequency parameter to 0 Hz
Set Center Frequency parameter to f_RF
Set Center Frequency parameter to f_RF
Set Frequency parameter of first receiver LO to f_RF - f_IF
Set Frequency parameter of second reciever LO to 0 Hz

See Also

General Mixer

More About

- “Model an RF Mixer”

Quadrature Mixers

In this section...

Use RF Blockset Equivalent Baseband Software to Model Quadrature Mixers

RF Blockset software lets you model upconversion and downconversion quadrature mixers using Physical blocks. These mixers convert a complex baseband signal up to and down from the desired carrier frequency by mixing the real and imaginary parts of the signal with a cosine and sine of the same frequency.

Model Upconversion I/Q Mixers

You use the Input Port block to model the upconversion of in-phase/quadrature baseband signals to modulated signals at a finite real carrier frequency. The real component of the block input represents the in-phase signal. The imaginary component of the block input represents the quadrature signal.

To model a perfect quadrature upconversion mixer, use the Input Port block with the **Center frequency (Hz)** parameter set to the carrier frequency.

To model an imperfect quadrature upconversion mixer, use the Input Port block with the **Center frequency (Hz)** parameter set to the carrier frequency. Follow this block immediately by a mixer block with the **LO frequency (Hz)** parameter set to `0`. Specify imperfections as follows:

- S-parameters — Use S-parameters to specify imperfections such as frequency response. For a mixer, S_{21} describes the conversion gain, as explained in the Network Parameters section of the reference page for each mixer block. Use purely real and purely imaginary S_{21} parameters to represent multiplying the input signal by a pure cosine and a pure sine, respectively. Use a complex S_{21} parameter to represent multiplying the input signal by a combination of sine and cosine.
- Thermal noise — Use thermal noise to specify temperature-dependent random noise.
- Phase noise — Use the phase noise to specify noise to add to the angle component of the input signal.
- Nonlinearity — Use nonlinearity (specified as output power and phase as a function of input power and frequency in an AMP file or as third-order intercept point) to specify nonlinear mixer behavior as a function of input power.

Note If you specify a nonzero value for the local oscillator frequency of the mixer and set the **Type** parameter to `Upconverter`, the blockset converts the signal to a frequency above the center frequency. The final IF value is the sum of the Input Port center frequency and the mixer local oscillator frequency.

Model Downconversion I/Q Mixers

You use the Output Port block to model the downconversion of in-phase/quadrature modulated carrier signals to baseband signals. The real component of the block output represents the in-phase signal. The imaginary component of the block output represents the quadrature signal.

The finite real carrier frequency is set automatically as the sum of the center frequency of the Input Port block and the LO frequencies in any mixer blocks in the cascade.

Note In the cascade, upconversion mixers increase the carrier frequency and downconversion mixers decrease the carrier frequency.

The Output Port block models a perfect quadrature downconversion mixer. To model an imperfect quadrature downconversion mixer, precede the Output Port block immediately by a mixer block with the **LO frequency (Hz)** parameter set to `0`. Specify imperfections as follows:

- S-parameters — Use S-parameters to specify imperfections such as frequency response. For a mixer, S_{21} describes the conversion gain, as explained in the Network Parameters section of the reference page for each mixer block. Use purely real and purely imaginary S_{21} parameters to represent multiplying the input signal by a pure cosine and a pure sine, respectively. Use a complex S_{21} parameter to represent multiplying the input signal by a combination of sine and cosine.
- Thermal noise — Use thermal noise to specify temperature-dependent random noise.
- Phase noise — Use the phase noise to specify noise to add to the angle component of the input signal.
- Nonlinearity — Use nonlinearity (specified as output power and phase as a function of input power and frequency in an AMP file or as third-order intercept point) to specify nonlinear mixer behavior as a function of input power.

Note The mixer output frequency must be positive. This means that if you choose a downconverting mixer, the input carrier frequency f_{in} must be greater than the local oscillator frequency f_{lo}. Otherwise, an error appears.

Simulate I/Q Mixers

When you model an I/Q mixer in the blockset, the center frequency you specify in the Input Port block dialog is only used to build a complex-baseband equivalent model of the cascade that represents the mixer. The blockset simulates this model using a fixed time step equal to the sample time that you specify in the Input Port block dialog box.

To examine the model in the Simulink window:

1. Click **Modeling** > **Compile Diagram** > **Update Model** to update the model diagram.
2. Right-click the Output Port block and select **Mask** > **Look Under Mask**.

For more information on baseband-equivalent modeling, see “Model RF Components” on page 6-2.

See Also

General Mixer

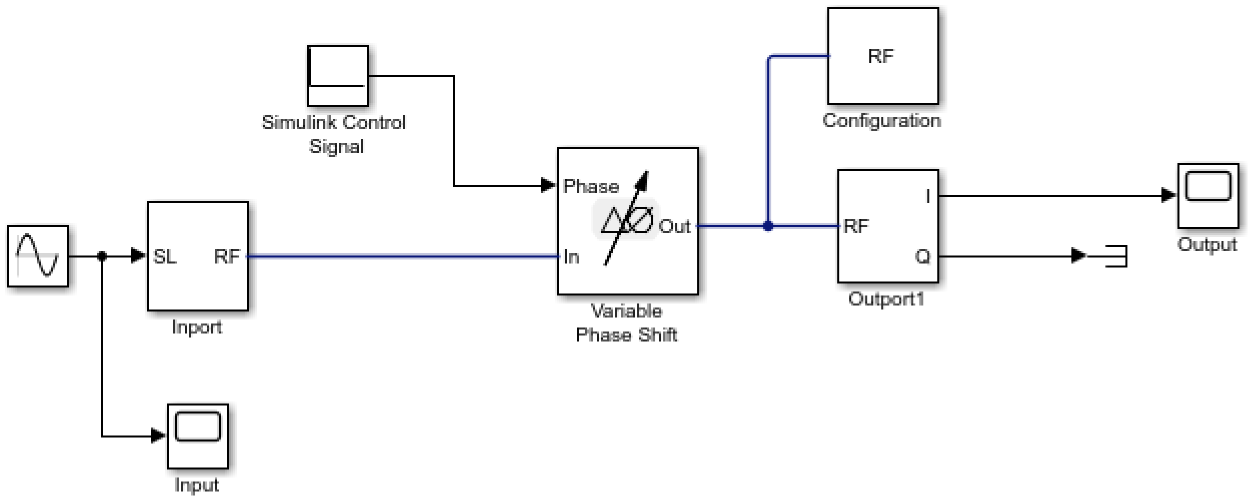
More About

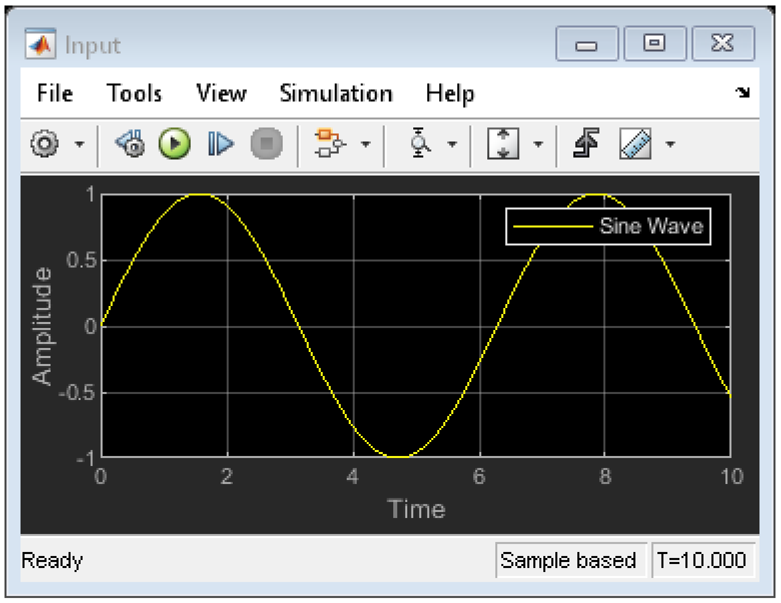
- “Model a Mixer Chain” on page B-4
- “Model RF Components” on page 6-2
- “Model an RF Mixer”

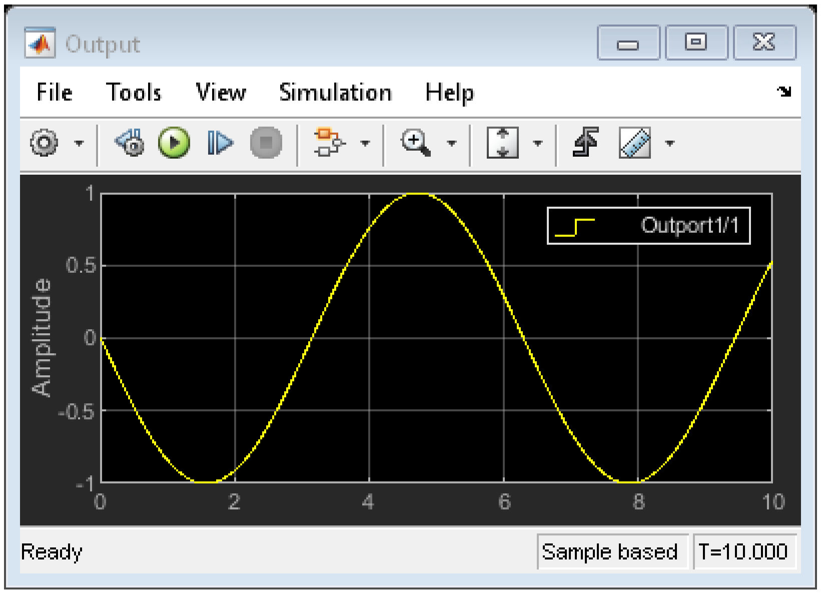
Examples

Vary Phase Of Signal During Simulation

Use the Variable Phase Shift block to shift the phase of a sine wave to 180 degrees.Use Repeating Sequence Stair block as a Simulink control signal to control the phase of the signal. To see the variation in phase to 180 degrees, first open and run the model. During simulation, change the value of the Simulink control signal to 90 degrees and see a change in phase in the Output Scope.

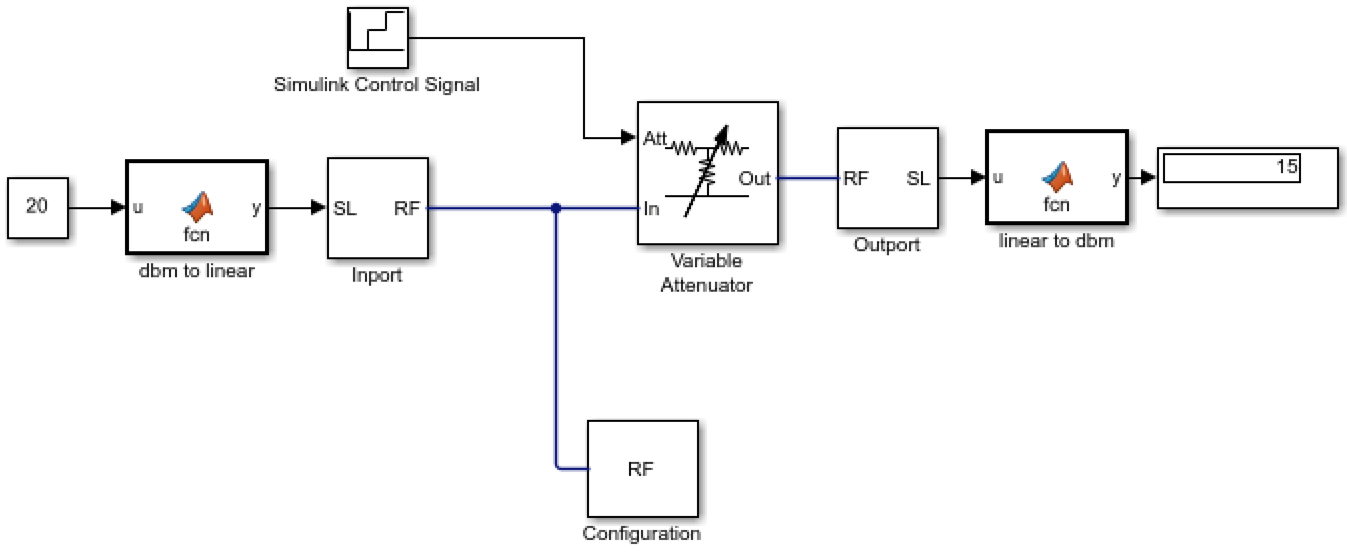





Output
File
Tools
View
Simulation
Help
Outport1/1
Amplitude
1
0.5
0
-0.5
-1
0
2
4
6
8
10
Ready
Sample based
T=10.000

Vary Attenuation of Signal During Simulation

Use the Variable Attenuator block to attenuate a 20 dB constant signal. Use the Repeating Sequence Stair block as a Simulink control signal to vary the attenuation of the signal. In this model,the signal attenuation varies between 5 and 10 dB during simulation. To see the variation in attenuation, open and run the model.

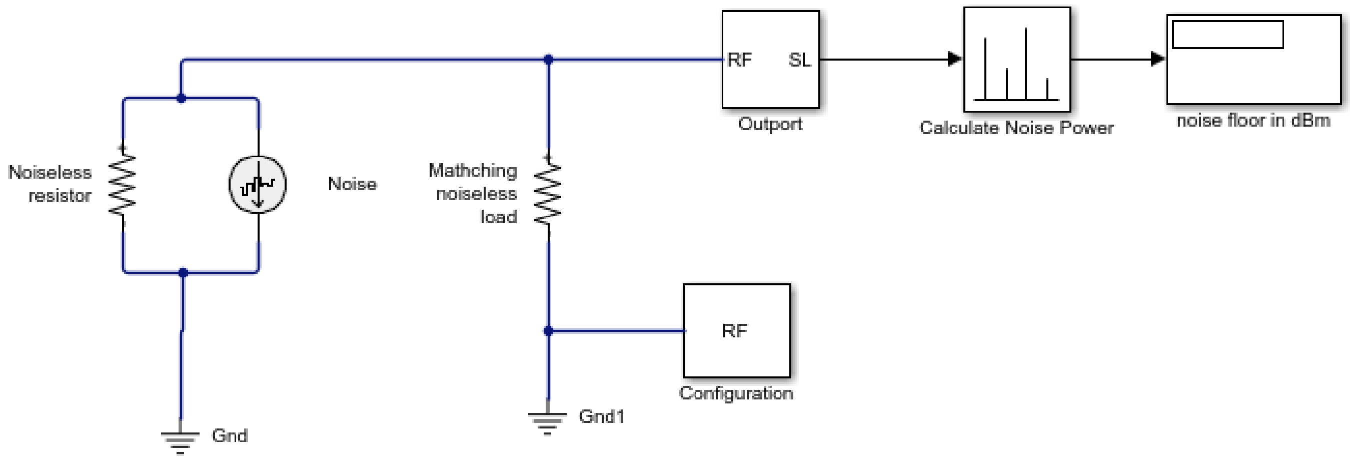


Explicitly Simulate Resistor Thermal Noise

Use the `Noise` block to calculate the classic thermal noise floor, `kT`, for a matched resistor circuit. Model configuration is as follows:

- Time step of the model is 1e-6 and frequency is 2 GHz.
- The `Resistor` noise source is modelled explicitly to make it noiseless. The resistance is 50 ohms. In the Resistor blocks, Simulate Noise is not selected.
- `Noise` current source parallel to the `Resistor` block models the noise. In the `Noise` block, the Source type is set to Ideal current to make it a current source. The Noise spectral density is defined as (4kT/R)(A2/Hz). The value of `k` is * The Masked block, Calculate Noise Power, calculates the noise floor as a standard deviation of the output signal.

```
open_system('model_simrf_noise_source1')
```



To run the model, select Simulation > Run. With the bandwidth included using the Configuration block, noise power is in the range of -173.98 to 174.1 dBm

See Also

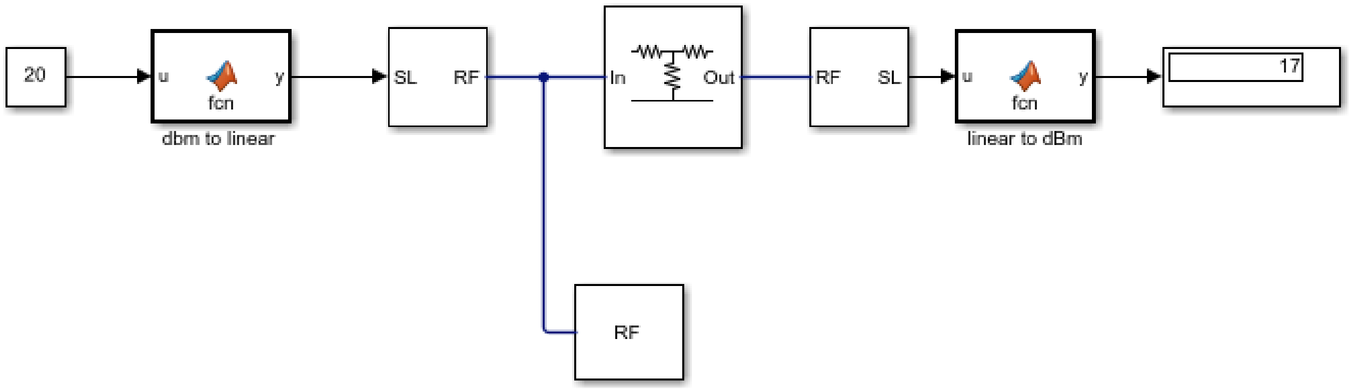
Configuration | Outport | Noise

Related Topics

“Spot Noise Data in Amplifiers and Effects on Measured Noise Figure” on page 8-16 | “RF Noise Modeling” on page 9-199

Attenuate Signal Power

Use the `Attenuator` block to attenuate a constant signal of 20 dB by 3 dB.



Demodulate Two-Tone RF Signal Using IQ Demodulator

Use the IQ Demodulator block to demodulate a two-tone RF signal to DC level. Observe the impairments in the demodulated output signal such as images due to gain imbalance, intermodulation distortion, and output third-order intercept (OIP3).

The two tones are at 10MHz and 15 MHz. The power of each tone is -30 dBm. The carrier frequency is 2 GHz.

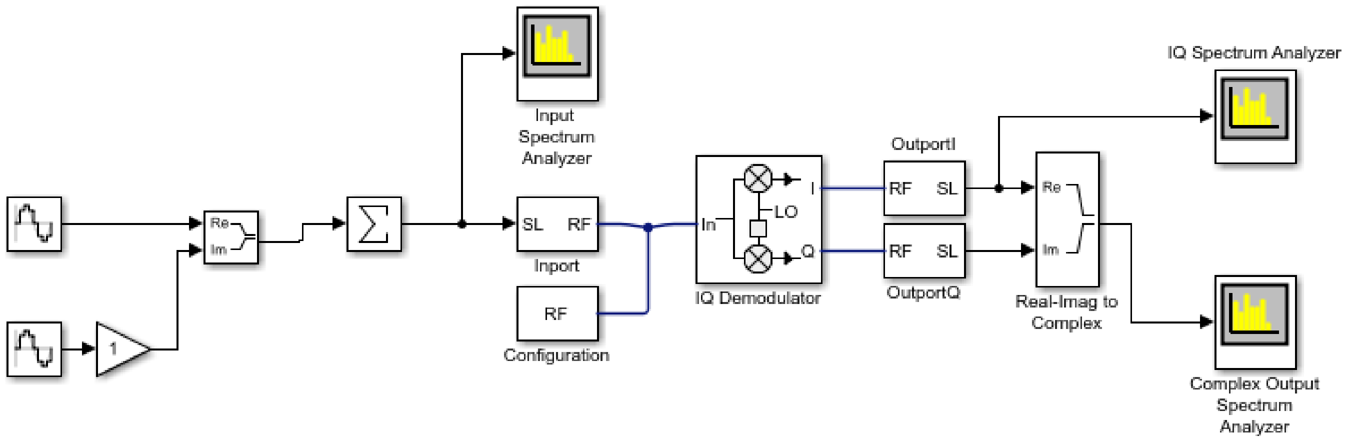
IQ Demodulator

The IQ Demodulator parameters are:

- Available power gain: 10 dB
- Local oscillator frequency: 2 GHz
- I/Q gain mismatch: 0.1 dB
- LO to RF Isolation: 90 dB
- Noise Figure: 6 dBm/Hz

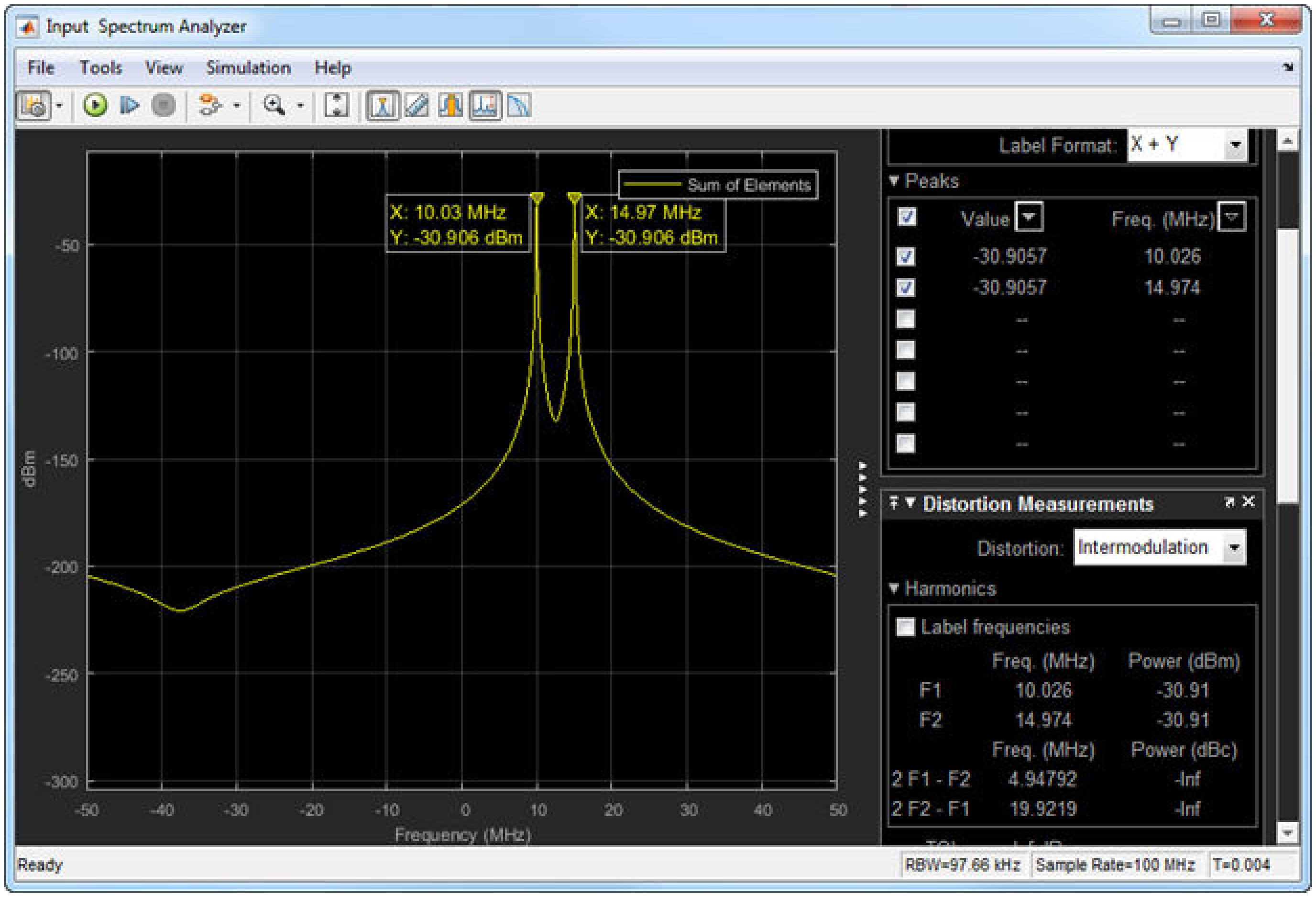
Open the model.

```
open('model_IQdemod')
```



Run the model and observe the spectrum analyzers.

Input Spectrum Analyzer



Label Format: X + Y

Peaks

✓	Value	Freq. (MHz)
✓	-30.9057	10.026
✓	-30.9057	14.974
	--	--
	--	--
	--	--
	--	--
	--	--

Distortion Measurements

Distortion: Intermodulation

Harmonics

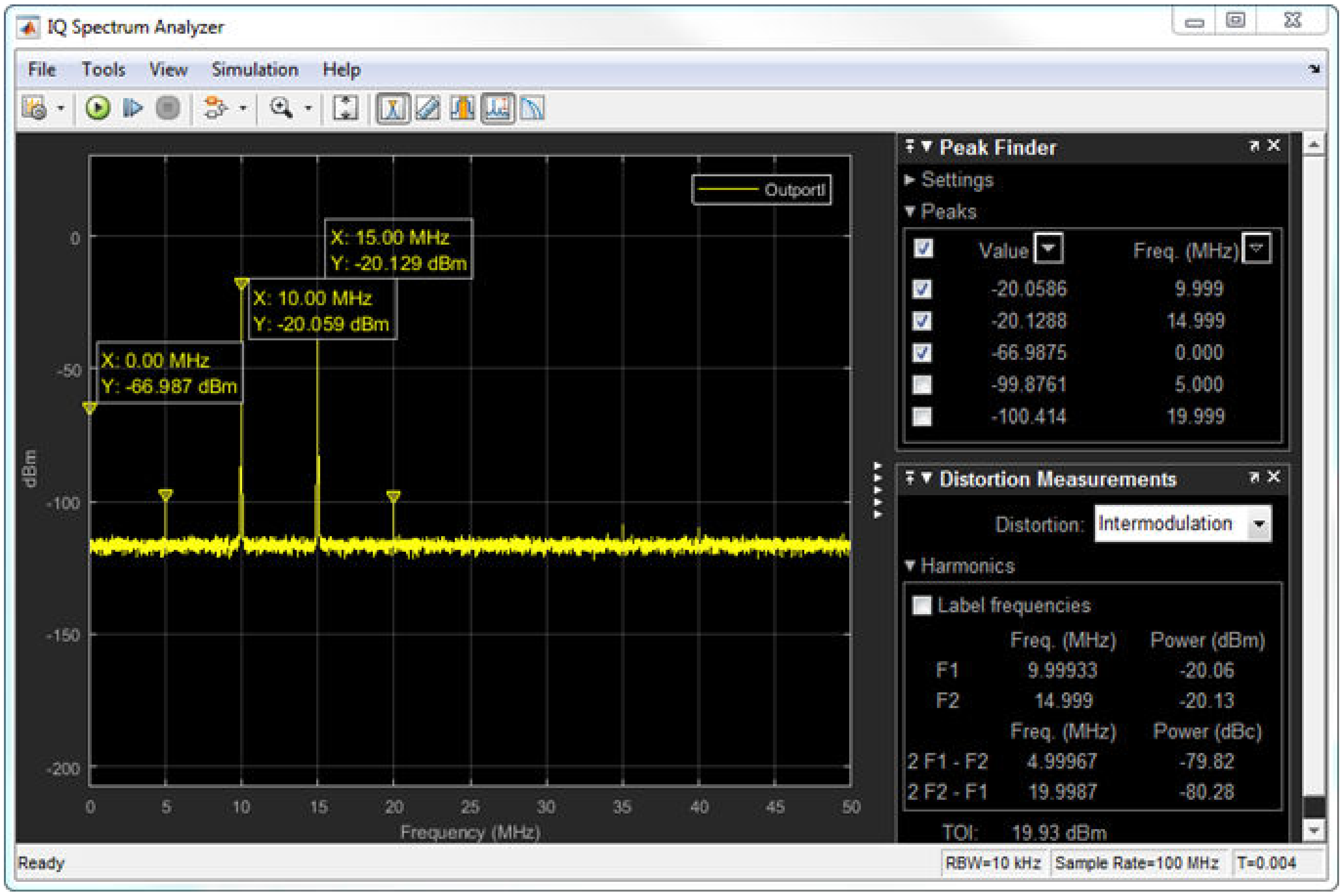
Label frequencies

	Freq. (MHz)	Power (dBm)
F1	10.026	-30.91
F2	14.974	-30.91
	Freq. (MHz)	Power (dBc)
2 F1 - F2	4.94792	-Inf
2 F2 - F1	19.9219	-Inf

RBW=97.66 kHz Sample Rate=100 MHz T=0.004

In the input spectrum analyzer, you see the input RF signal with the two tones at 10 MHz and 15 MHz. The power level of each tone is -30 dBm. The carrier frequency is 2 GHz.

I/Q Spectrum Analyzer



Peak Finder

Value	Freq. (MHz)
-20.0586	9.999
-20.1288	14.999
-66.9875	0.000
-99.8761	5.000
-100.414	19.999

Distortion Measurements

Distortion: Intermodulation

Harmonics

Label frequencies

	Freq. (MHz)	Power (dBm)
F1	9.99933	-20.06
F2	14.999	-20.13
	Freq. (MHz)	Power (dBc)
2 F1 - F2	4.99967	-79.82
2 F2 - F1	19.9987	-80.28

TOI: 19.93 dBm

RBW=10 kHz Sample Rate=100 MHz T=0.004

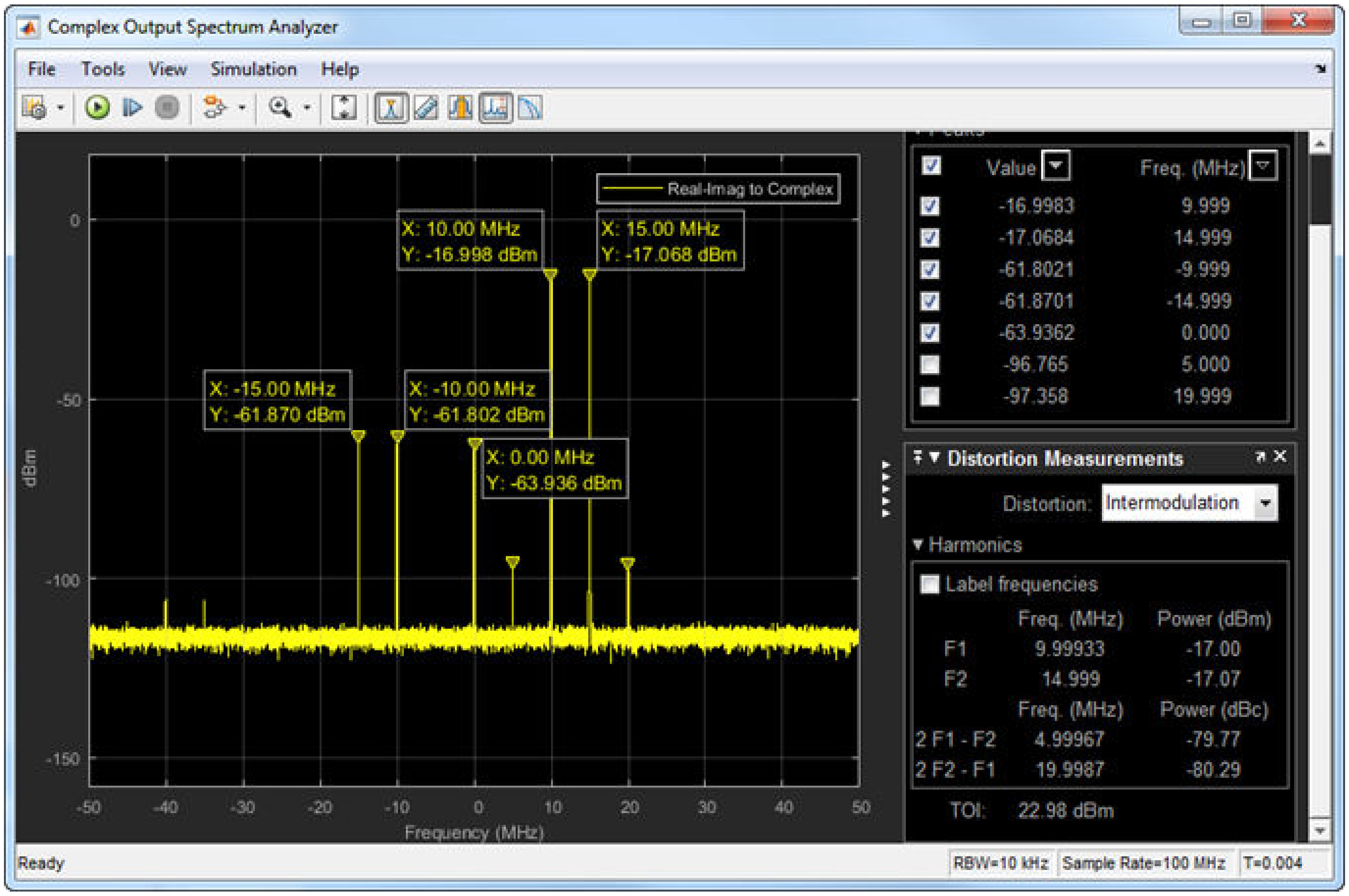
In the I/Q spectrum analyzer, you see the inphase part of the demodulated signal including the DC signal level and the two tones. The following formula gives you the DC power level of the signal:

$$DClevelI/Q = PowerLO - dBmIsolation + Gain$$

$$DclevelI/Q = 20 * log(1/sqrt(50)) + 30 - 90 + 10 = -67dBm$$

The output power level of the two tones are -20 dBm. You also see the OIP3 value (measured by the spectrum analyzer) at approximately 20 dBm.

Complex Output Spectrum Analyzer



	Value	Freq. (MHz)
✓	-16.9983	9.999
✓	-17.0684	14.999
✓	-61.8021	-9.999
✓	-61.8701	-14.999
✓	-63.9362	0.000
☐	-96.765	5.000
☐	-97.358	19.999

Distortion Measurements

Distortion: Intermodulation

Harmonics

☐ Label frequencies

	Freq. (MHz)	Power (dBm)
F1	9.99933	-17.00
F2	14.999	-17.07
	Freq. (MHz)	Power (dBc)
2 F1 - F2	4.99967	-79.77
2 F2 - F1	19.9987	-80.29

TOI: 22.98 dBm

In the complex output spectrum analyzer, you see the whole demodulated signal including the imaginary parts. The output power level of the two tones (10 MHz and 15 MHz) is -17 dBm.

Image Rejection Ratio

The images of the two tones are at -10 MHz and -15 MHz. The output power level of the images are -61.78 dBm. Image rejection ratio is given by the formula:

$$IMRR = [(gainimbalance)^2 + 1 - 2*(gainimbalance)]/[(gainimbalance)^2 + 1 + 2*(gainimbalance)]$$

$$GainImbalance = 10^{(}0.1/20) = 1.0116$$

$$IMRRdB = 10*log10(3.3253e-05) = -44.78dB$$

$$Imagelevel = -17dBm - 44.78dB = -61.78dBm$$

See Also

IQ Demodulator | Configuration | Inport | Outport

Related Topics

"Modulate Two-Tone DC Signal Using IQ Modulator" on page 8-51

Modulate Two-Tone DC Signal Using IQ Modulator

Use the IQ Modulator block to Modulate a two-tone DC signal to RF level. Observe the impairments in the modulated output signal such as images due to gain imbalance, intermodulation distortion, and output third-order intercept (OIP3).

The two tones are at 10MHz and 15 MHz.The power of each tone is -30 dBm. The carrier frequency is 0 GHz.

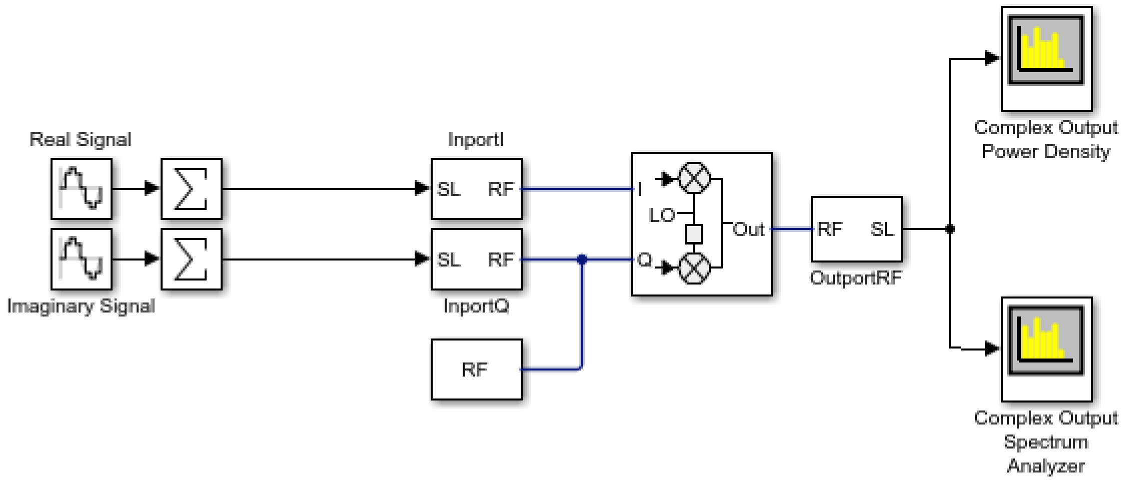
IQ Modulator

The IQ modulator parameters are :

- Available power gain: 10 dB
- Local oscillator frequency: 2 GHz
- I/Q gain mismatch: 0.1 dB
- LO to RF Isolation: 90 dB
- Noise Floor: -160 dBm/Hz
- IP3: 10 dBm

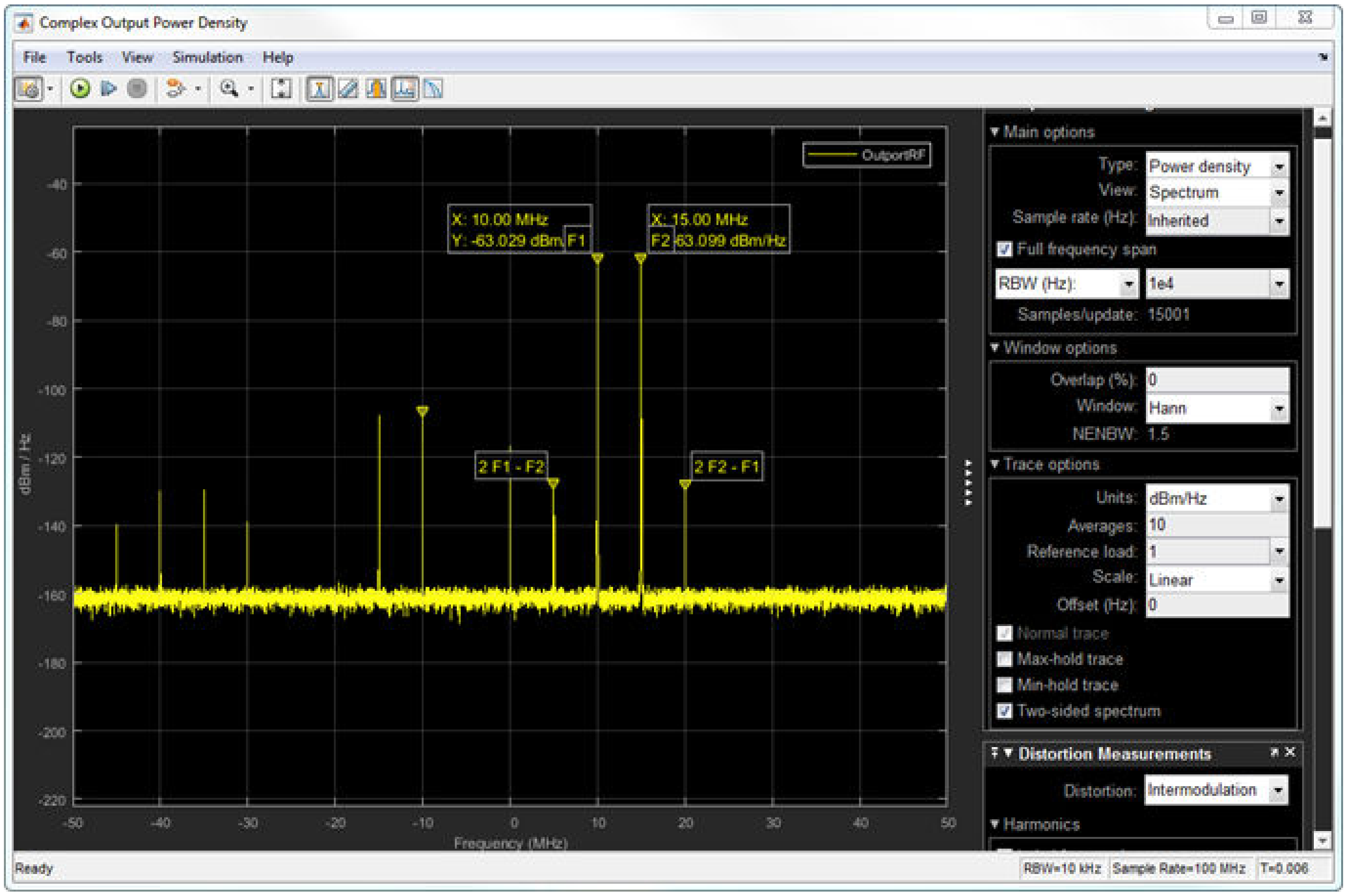
Open the model.

```
open('model_IQmod')
```



Run the model and observe the spectrum analyzers.

Complex Output Power Density



▾ Main options

Type: Power density
View: Spectrum
Sample rate (Hz): Inherited
☑ Full frequency span
RBW (Hz): 1e4
Samples/update: 15001

▾ Window options

Overlap (%): 0
Window: Hann
NENBW: 1.5

▾ Trace options

Units: dBm/Hz
Averages: 10
Reference load: 1
Scale: Linear
Offset (Hz): 0
☑ Normal trace
☐ Max-hold trace
☐ Min-hold trace
☑ Two-sided spectrum

▾ Distortion Measurements

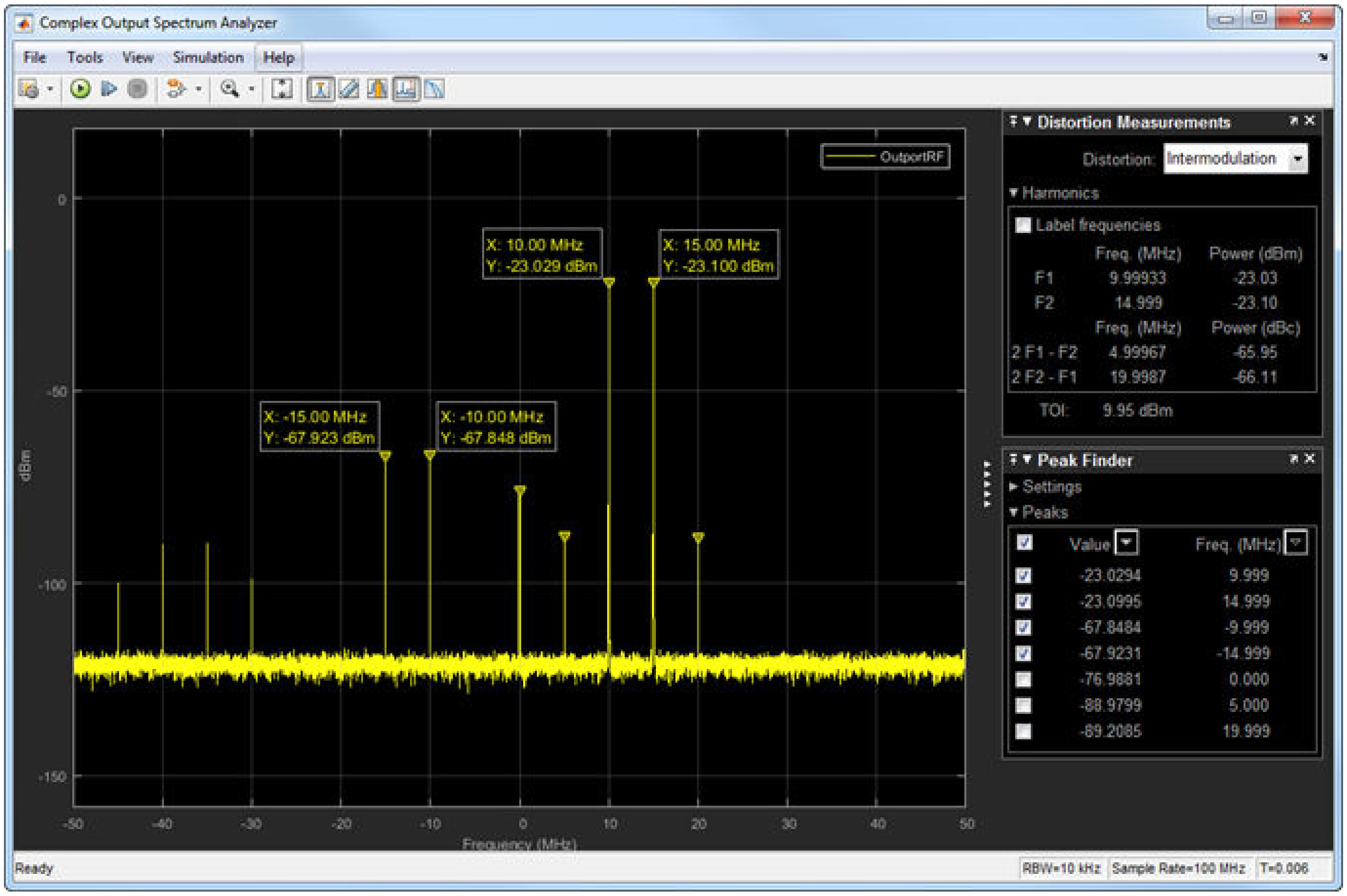
Distortion: Intermodulation

▾ Harmonics

Ready | RBW=10 kHz | Sample Rate=100 MHz | T=0.006

In the complex output power density spectrum analyzer, you see the noise floor of the signal at -160 dBm/Hz.

Complex Output Spectrum Analyzer



Distortion Measurements

Distortion: Intermodulation

Harmonics

☐ Label frequencies

	Freq. (MHz)	Power (dBm)
F1	9.99933	-23.03
F2	14.999	-23.10
	Freq. (MHz)	Power (dBc)
2 F1 - F2	4.99967	-65.95
2 F2 - F1	19.9987	-66.11

TOI: 9.95 dBm

Peak Finder

Settings

Peaks

✓	Value	Freq. (MHz)
✓	-23.0294	9.999
✓	-23.0995	14.999
✓	-67.8484	-9.999
✓	-67.9231	-14.999
☐	-76.9881	0.000
☐	-88.9799	5.000
☐	-89.2085	19.999

In the complex output spectrum analyzer, you see the whole modulated signal including the imaginary parts. The output power level of the two tones (10 MHz and 15 MHz)is -20 dBm.

$$Outputpowerlevel = Inputpowerlevel + gain = -30dBm + 10dB = -20dBm$$

The output third-order intercept(OIP3) is at 10 dBm. The spectrum analyzer measures this value.

Image Rejection Ratio

The images of the two tones are at -10 MHz and -15 MHz. The output power level of the two images are -67.8 dBm. Image rejection ratio is given by the formula:

$$IMRR = [(gainimbalance)^2 + 1 - 2 * (gainimbalance)]/[(gainimbalance)^2 + 1 + 2 * (gainimbalance)]$$

where,

$$GainImbalance = 10^(0.1/20) = 1.0116$$

$$IMRRdB = 10 * log10(3.3253e - 05) = -44.78dB$$

The image power level is given by the formula:

$$Imagelevel = -23dBm - 44.78dB = -67.78dBm$$

See Also

IQ Modulator

Related Topics

“Demodulate Two-Tone RF Signal Using IQ Demodulator” on page 8-46

Spot Noise Data in Amplifiers and Effects on Measured Noise Figure

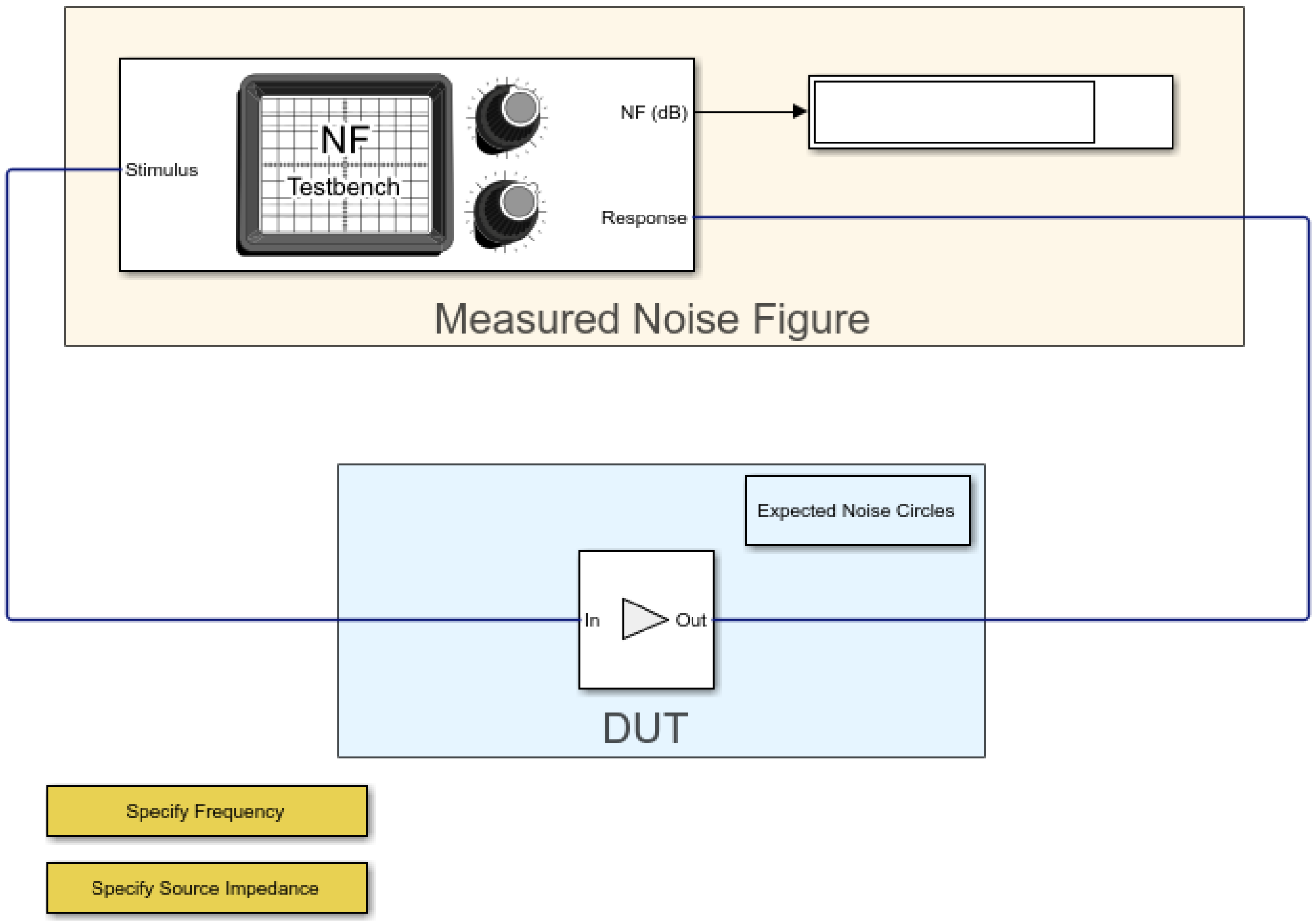
This example shows a test bench model to describe the noise introduced by a 2-port device.

The spot noise data parameters, `Fmin`, Γ `opt`, and `Rn`, fully describe the noise introduced by a 2-port device. These parameters along with the source impedance, `Zs` uniquely determine the measured noise figure of the device. You can use noise circles plotted on a Smith chart to show interaction between `Zs` and the noise figure.

Measure Noise Figure in RF Blockset

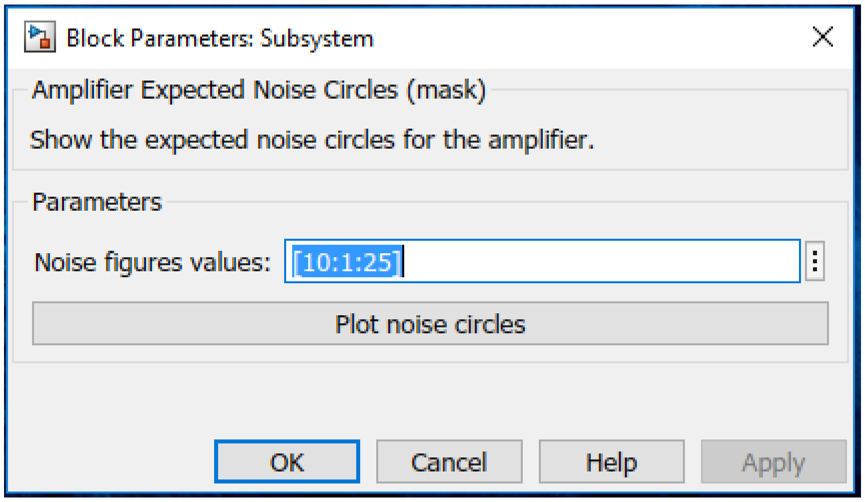
The model `Noise_figure_ex` simulates a simple noise figure measurement. In this model, the device under test comprises of a single amplifier. To open the model,

```
open_system('Noise_figure_ex.slx')
```



Please click on the **Open Script** button before running the model. To run the model, select **Simulation > Run**.

You see that the displayed value settles down at 10.0 dB of the measured noise figure. In this testbench, the amplifier parameters represent a simple attenuator of 10 dB matched to 25 Ohms at both input and output. Due to thermal equilibrium, the expected noise figure of the attenuator is 10 dB when matched, corresponding to the measured value. To gain a broader view of the expected noise figure values for the amplifier when the source impedance deviates from the matched value of 25 Ohm, double-click the `Expected Noise Circles` subsystem placed in the vicinity of the amplifier:


Block Parameters: Subsystem

Amplifier Expected Noise Circles (mask)

Show the expected noise circles for the amplifier.

Parameters

Noise figures values: [10:1:25]

Click `Plot noise circles` to bring up a figure showing a Smith chart with circles corresponding to the noise figure values specified above the button:

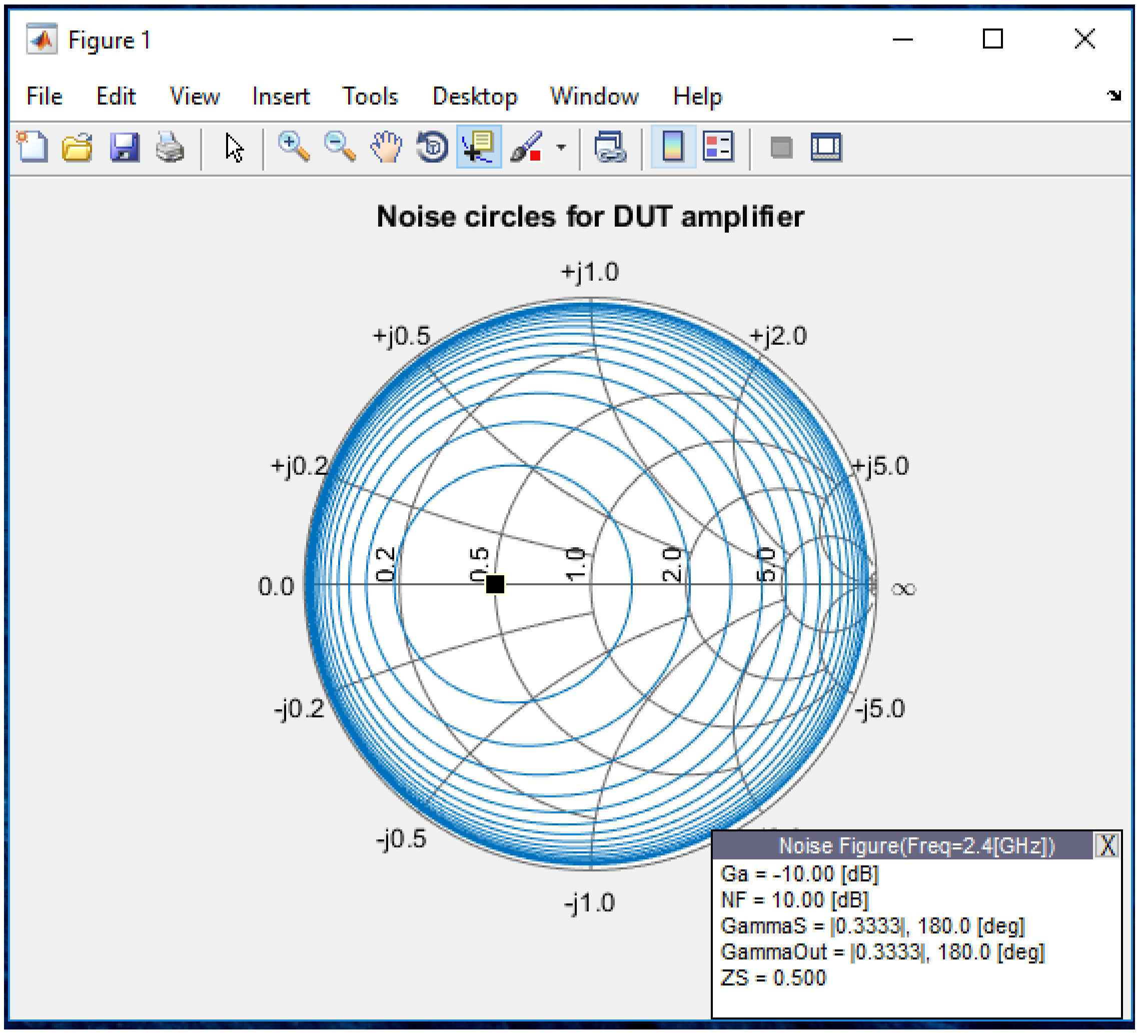


Figure 1
File Edit View Insert Tools Desktop Window Help
Noise circles for DUT amplifier
+j1.0
+j0.5
+j2.0
+j0.2
+j5.0
0.0
0.2
0.5
1.0
2.0
5.0
∞
-j0.2
-j5.0
-j0.5
-j1.0

Noise Figure(Freq=2.4[GHz])
Ga = -10.00 [dB]
NF = 10.00 [dB]
GammaS = |0.3333|, 180.0 [deg]
GammaOut = |0.3333|, 180.0 [deg]
ZS = 0.500

The values shown in the Smith chart represent the expected noise figures obtained theoretically from the parameters specified in the amplifier [1]. The Smith chart is interactive and you can place the data cursor on any circle to view the corresponding noise figure, source impedance (normalized to the reference impedance of the amplifier, Z0), and other RF properties. The initial data cursor position corresponds to a point on a noise circle that is closest to the source impedance specified. To control the complex value of this source impedance, double-click the `Specify Source Impedance` subsystem, and specify the desired value in the edit box.

To validate that the simulated measured noise figure corresponds to theoretical values, specify a reference impedance from the Smith chart, using the `Specify Source Impedance` subsystem. Run the model again.

Measuring Other RF Systems

You can replace the amplifier in the model by any other RF Blockset system and measure its noise figure. In case the system is frequency depended, you can change the frequency for the measurements by double-clicking the `Specify Frequency` subsystem and specifying the desired frequency. This frequency is also used for the amplifier noise circles plot.

The plotted expected noise circles apply to the `Amplifier` block alone. The plotted circles capture correctly all types of data inputs specified in the amplifier, including s2p based network and noise data. Note that the `Available Gain` shown in the data window of the Smith chart is based on the original data and ignores inaccuracies introduced by the amplifier modeling method. The plotting fails if a block named `Amplifier` does not exist in the model.

Using Other RF Blockset Blocks

Three additional ways to implement the attenuator specified in the amplifier are:

1. Use an RF Blockset `Attenuator` block with attenuation of 10dB, with input and output impedances set to 25 Ohms.
2. Implement the attenuator using three resistors arranged in a T or π topology.
3. Use an `S-parameter` block with the same Scattering matrix used in the amplifier. Select `Simulate noise` in the block. Simulating noise in a passive S-parameter block accounts for the resistive noise introduced by the S-parameters.

See Also

Noise

More About

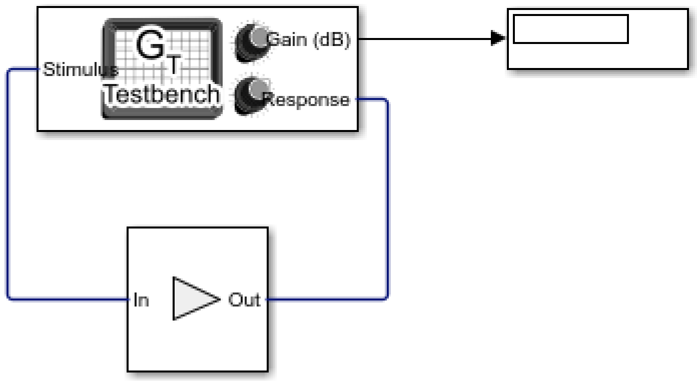
- “RF Noise Modeling” on page 9-199

Transducer Gain TestBench

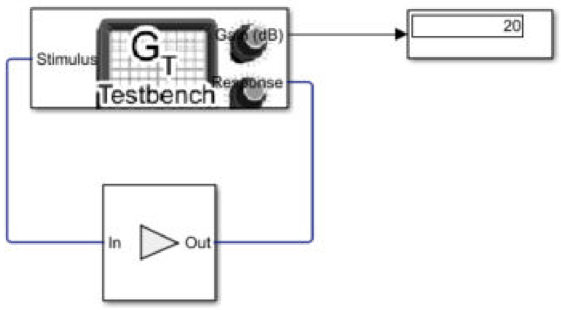
Use the `Transducer Gain Testbench` block to verify the gain of an `Amplifier` block.

Open the model and change the gain of the amplifier block to 20 dB.

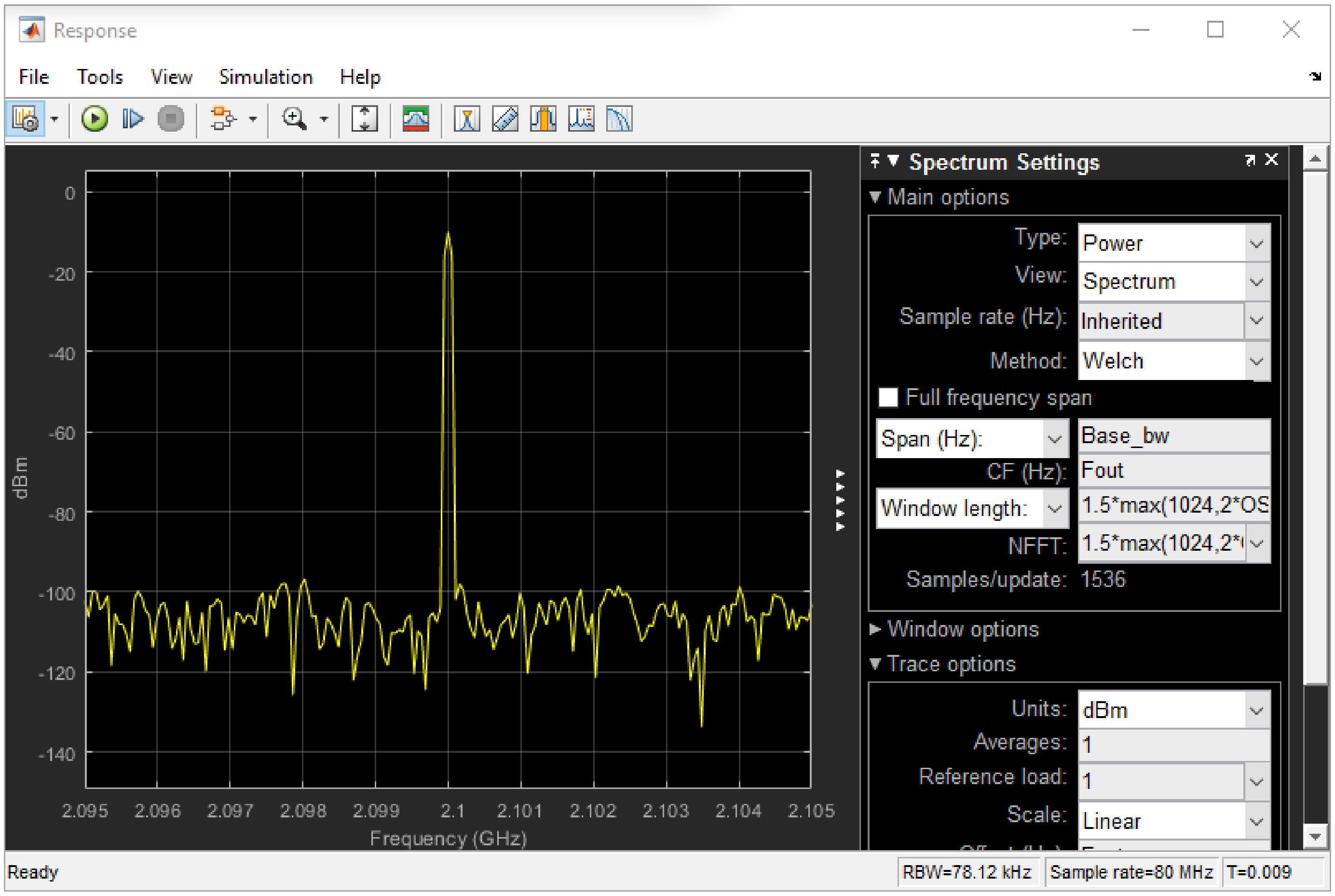
```
open_system('transducer_gain_example.slx')
```



Run the model. You will see that the output display shows a gain of 20 dB.



The scope response shows a gain of 20 dBm at 2.1 GHz, which is the specified frequency in the testbench dialogue box.



Spectrum Settings

Main options

Type: Power
View: Spectrum
Sample rate (Hz): Inherited
Method: Welch
Full frequency span
Span (Hz): Base_bw
CF (Hz): Fout
Window length: 1.5*max(1024,2*OS
NFFT: 1.5*max(1024,2*
Samples/update: 1536

Window options

Trace options

Units: dBm
Averages: 1
Reference load: 1
Scale: Linear

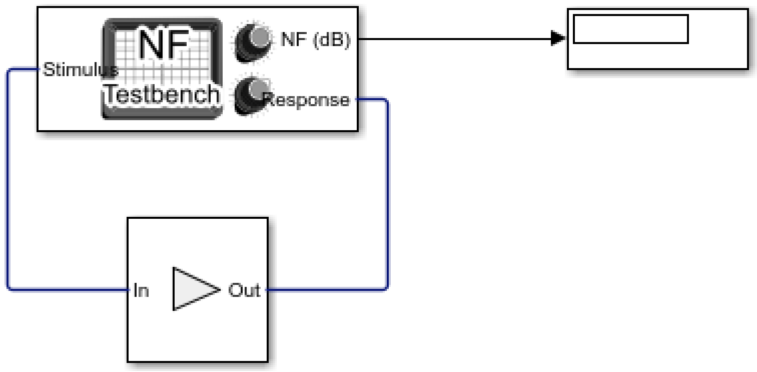
Ready
RBW=78.12 kHz
Sample rate=80 MHz
T=0.009

Noise Figure Testbench

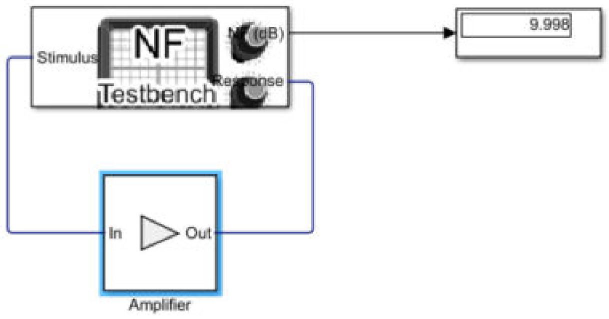
Use the `Noise Figure Testbench` block to verify the noise figure of an `Amplifier` block.

Open the model and change the noise figure of the amplifier block to 10 dB. In the Noise Figure Testbench block, clear `Show Response spectrum`.

```
open_system('noise_figure_example.slx')
```



Run the model. You will see that the display shows a noise figure close to 10 dB.

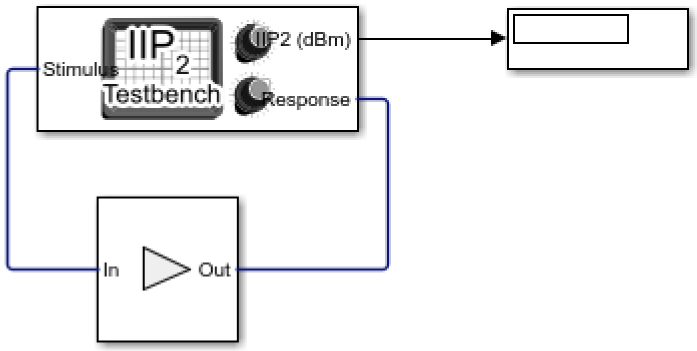


IIP2 Testbench

Use the `IIP2 Testbench` block to verify the input second order intercept (iip2) of an `Amplifier` block.

Open the model.

```
open_system('iip2_example.slx')
```



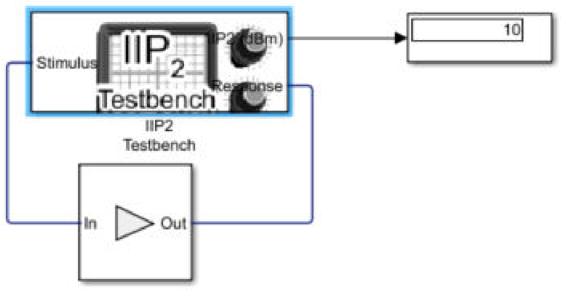
Open the amplifier block and change the following under **Nonlinearity**:

- Intercept points convention = Input
- IP2 = 10 dB

In the IIP2 Testbench block, clear the following:

- Simulate noise (both stimulus and DUT internal)
- Show Response spectrum

Run the model. You will see that the display shows a iip2 value of 10 dB.

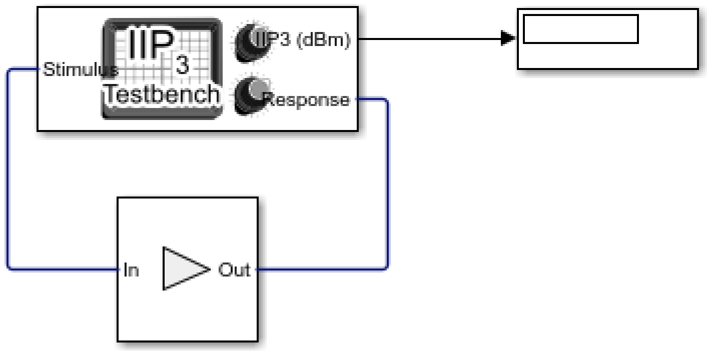


IIP3 Testbench

Use the `IIP3 Testbench` block to verify the input third order intercept (IIP3) of an `Amplifier` block.

Open the model.

```
open_system('iip3_example.slx')
```



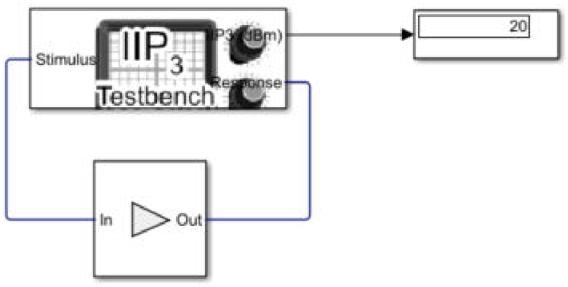
Open the Amplifier block and change the following under **Nonlinearity**:

- Intercept points convention = Input
- IP3 = 20 dB

In the IIP3 Testbench block, clear the following:

- Simulate noise (both stimulus and DUT internal)
- Show Response spectrum

Run the model. You will see that the display shows a IIP3 value of 20 dB.

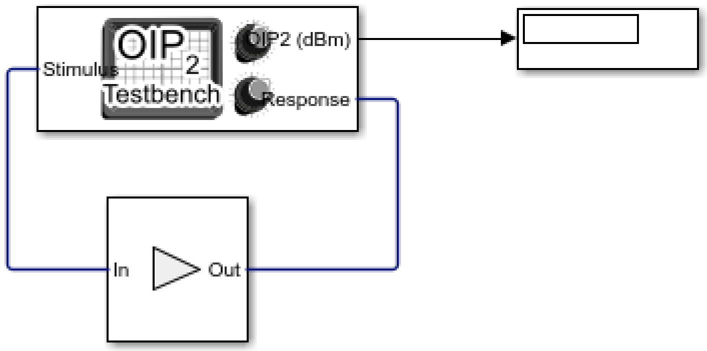


OIP2 Testbench

Use the `OIP2 Testbench` block to verify the output second order intercept (OIP2) of an `Amplifier` block.

Open the model.

```
open_system('oip2_example.slx')
```



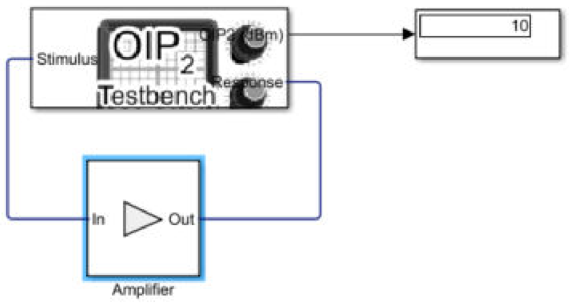
Open the amplifier block and change the following under **Nonlinearity**:

- Intercept points convention = Output
- IP2 = 10

In the OIP2 Testbench block, clear the following:

- Simulate noise (both stimulus and DUT internal)
- Show Response spectrum

Run the model. You will see that the display shows a OIP2 value of 10 dB.

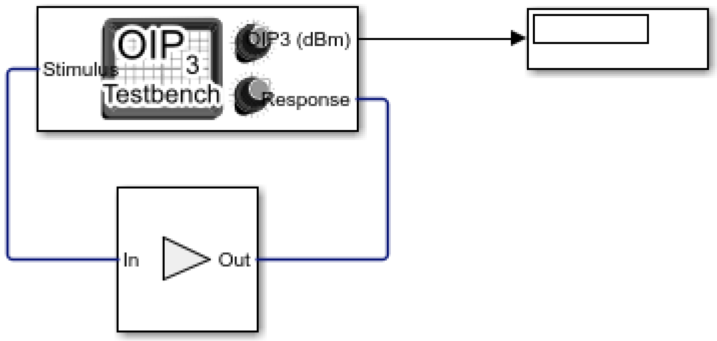


OIP3 Testbench

Use the `OIP3 Testbench` block to verify the output third order intercept (oip3) of an `Amplifier` block.

Open the model.

```
open_system('oip3_example.slx')
```



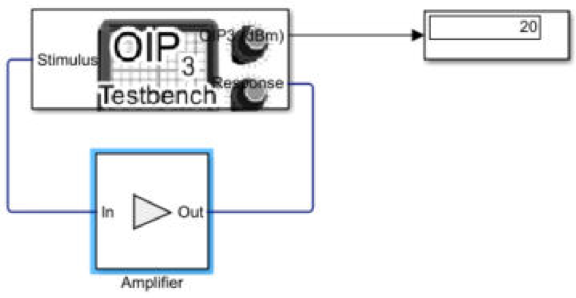
Open the amplifier block and change the following under **Nonlinearity**:

- Intercept points convention = Output
- IP3 = 20

In the OIP3 Testbench block, clear the following:

- Simulate noise (both stimulus and DUT internal)
- Show Response spectrum

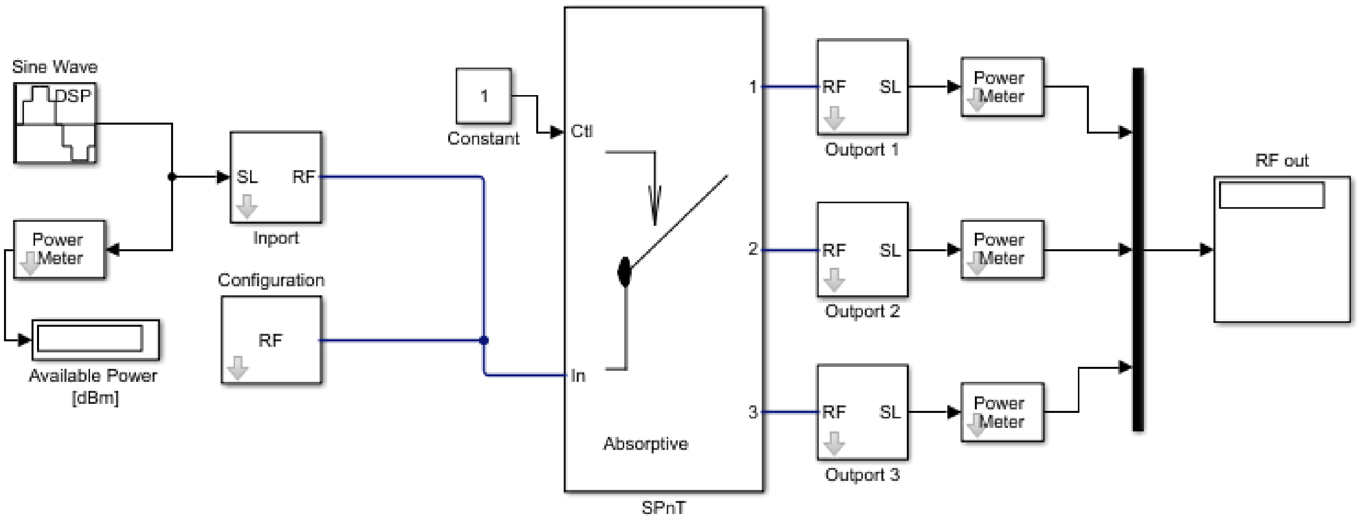
Run the model. You will see that the display shows a oip3 value of 20 dB.



Single Pole Triple Throw Switch

Use the SPnT block to create a single pole triple throw switch to switch a signal between three outputs.

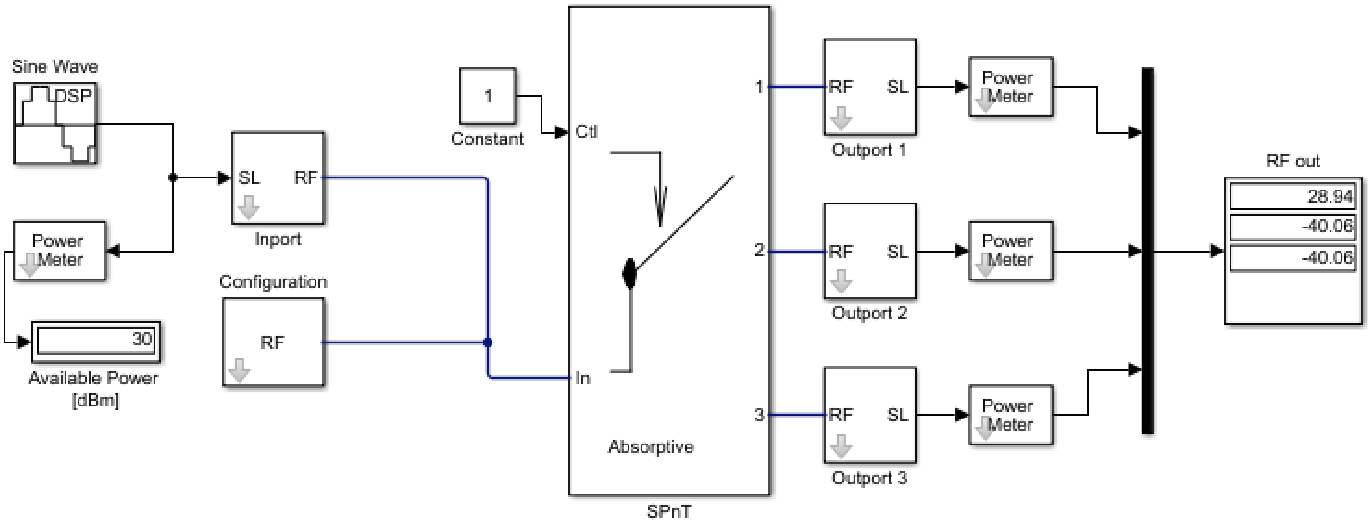
Open the model.



The model consists of:

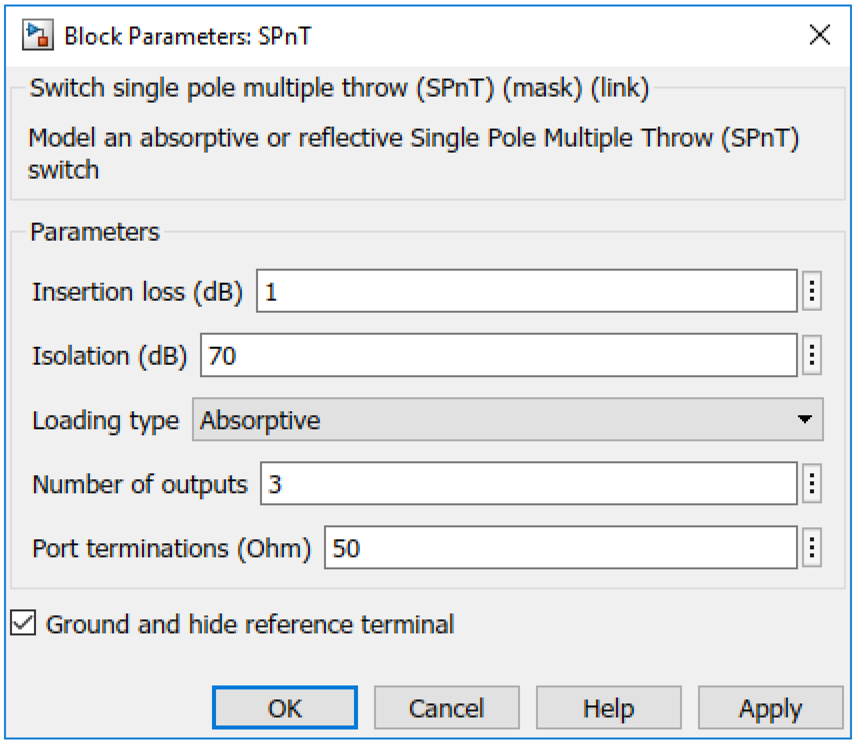
- Sine Wave block to generate a sine wave of amplitude 1.
- Power Meter block to determine the power of the sine wave. This is the input power. The input power is 30 dB.
- Inport and Configuration blocks connected to the "In" port of the SPnT block.
- A Constant block connected to the "Ctl" port of the SPnT block. This signal is used to control the switch outputs.
- SPnT switch block with 3 output ports.
- Outport 1, Outport 2, Outport 3, and Power meter blocks to calculate the power of each output signal.
- Display block to display the three outputs.

Run the model.



The Display block shows that the signal power is available through the first port of the switch as the "Ctl" port is set to 1.

Open the SPnT block to see set values. Currently the switch is set to "Absorptive" using the "Load Type" parameter.


Block Parameters: SPnT

Switch single pole multiple throw (SPnT) (mask) (link)

Model an absorptive or reflective Single Pole Multiple Throw (SPnT) switch

Parameters

Insertion loss (dB) 1

Isolation (dB) 70

Loading type Absorptive

Number of outputs 3

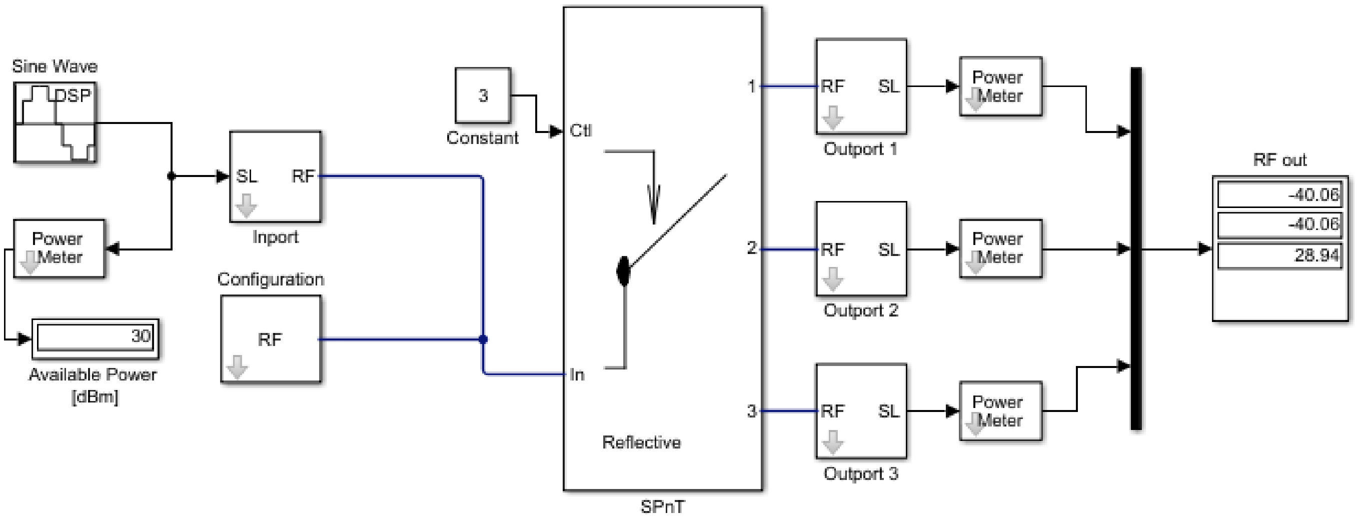
Port terminations (Ohm) 50

☑ Ground and hide reference terminal

Change the "Load Type" value to "Reflective".

Change the value of the Constant block to 3.

Run the model again.



The Display block shows that the signal power is available through the third port of the switch as the "Ctl" port is set to 3.

See Also

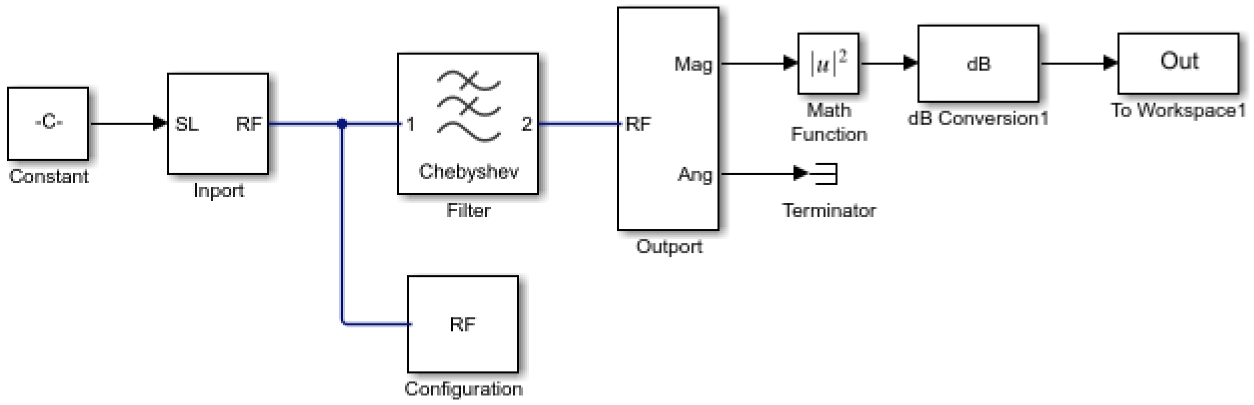
Inport | Outport | SPnT

Frequency Response of Lowpass Chebyshev Filter

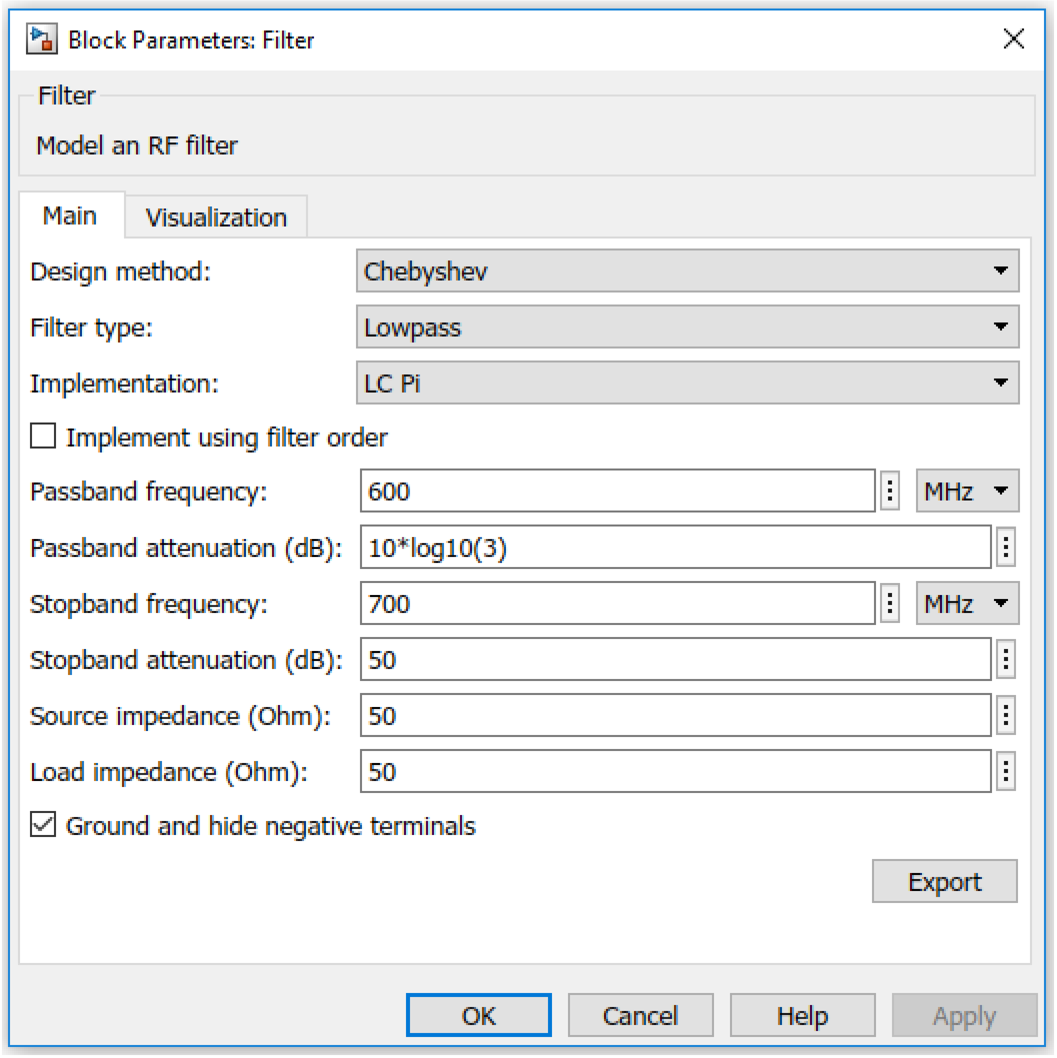
Use the `Filter` block to study the frequency response of a lowpass Chebyshev filter.

From the MATLAB command prompt, open the model.

```
open_system('ex_simrf_filter_lowpass_cheby_resp')
```



The `Constant` block sets the amplitude of the 201 carrier signals to ones (1, 201). The `Inport` block generates the 201 carrier frequencies for the mask value of logspace (7, 9, 201). Generate an 11th order lowpass LC Pi Chebyshev filter by setting appropriate block parameters in the `Filter` block.


Block Parameters: Filter

Filter

Model an RF filter

Design method: Chebyshev

Filter type: Lowpass

Implementation: LC Pi

☐ Implement using filter order

Passband frequency: 600 MHz

Passband attenuation (dB): 10*log10(3)

Stopband frequency: 700 MHz

Stopband attenuation (dB): 50

Source impedance (Ohm): 50

Load impedance (Ohm): 50

☑ Ground and hide negative terminals

The output signal from the `Filter` block is fed into the `Outport` block. The `Outport` block is configured to give both magnitude and angle of the signal. The angle output is terminated using the `Terminator` block. The magnitude output is squared and converted to dB using `Math Function` and `dB Conversion` blocks.

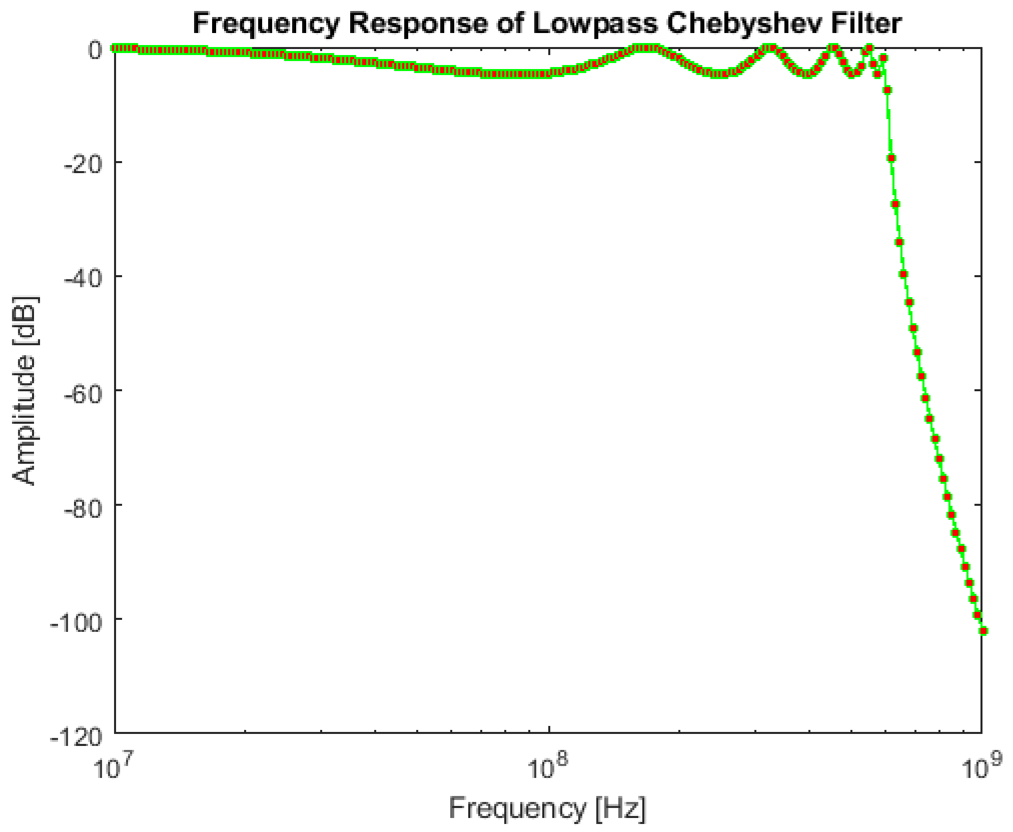
To run the model, select Simulation > Run. You can also use the following command:

```
sim('ex_simrf_filter_lowpass_cheby_resp')
```

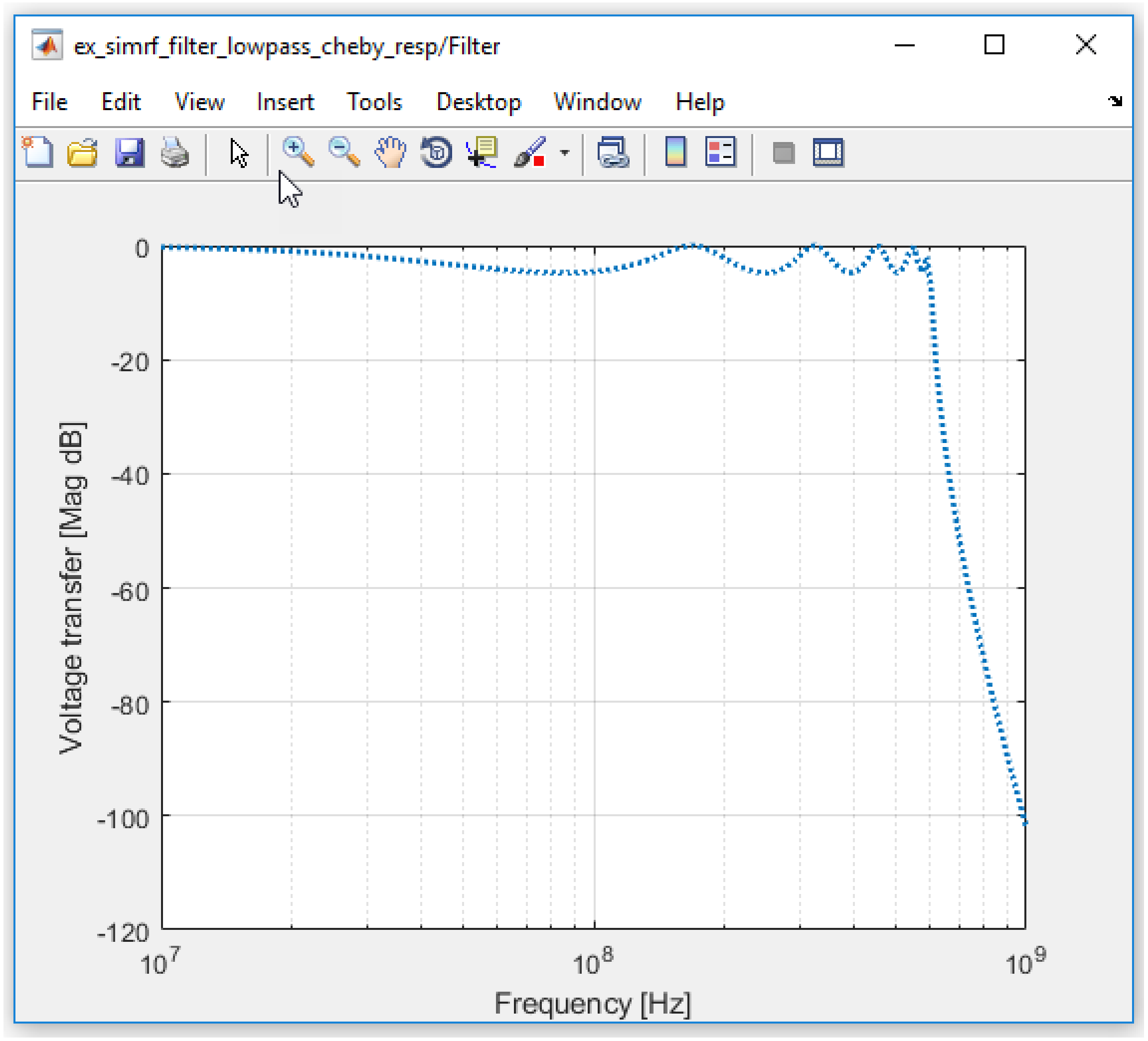
The model creates an `Out` array in the MATLAB workspace. Since the simulation stop time is set to 0, the frequency response corresponds to the steady state solution.

To plot the frequency response, use the following commands in the MATLAB command window.

```
figure
freq = logspace(7,9,201);
h = semilogx(freq, Out, '-gs', 'LineWidth',1, 'MarkerSize',3, 'MarkerFaceColor','r');
xlabel('Frequency [Hz]');
ylabel('Amplitude [dB]');
title('Frequency Response of Lowpass Chebyshev Filter');
```



You can also use the Plot button in the Visualization tab of the `Filter` block parameters. Set the Frequency points to logspace (7, 9, 201) and X-axis scale to Logarithmic to achieve a similar plot.



See Also

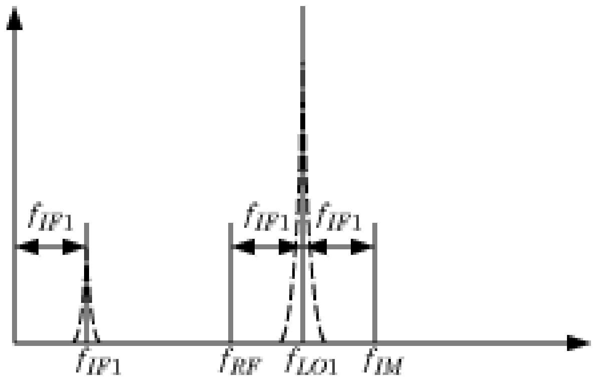
Filter | Inport | Outport | Configuration

Related Topics

“Model RF Filter Using Circuit Envelope”

Model LO Phase Noise

A mixer transfers local oscillator (LO) phase noise directly to its output.

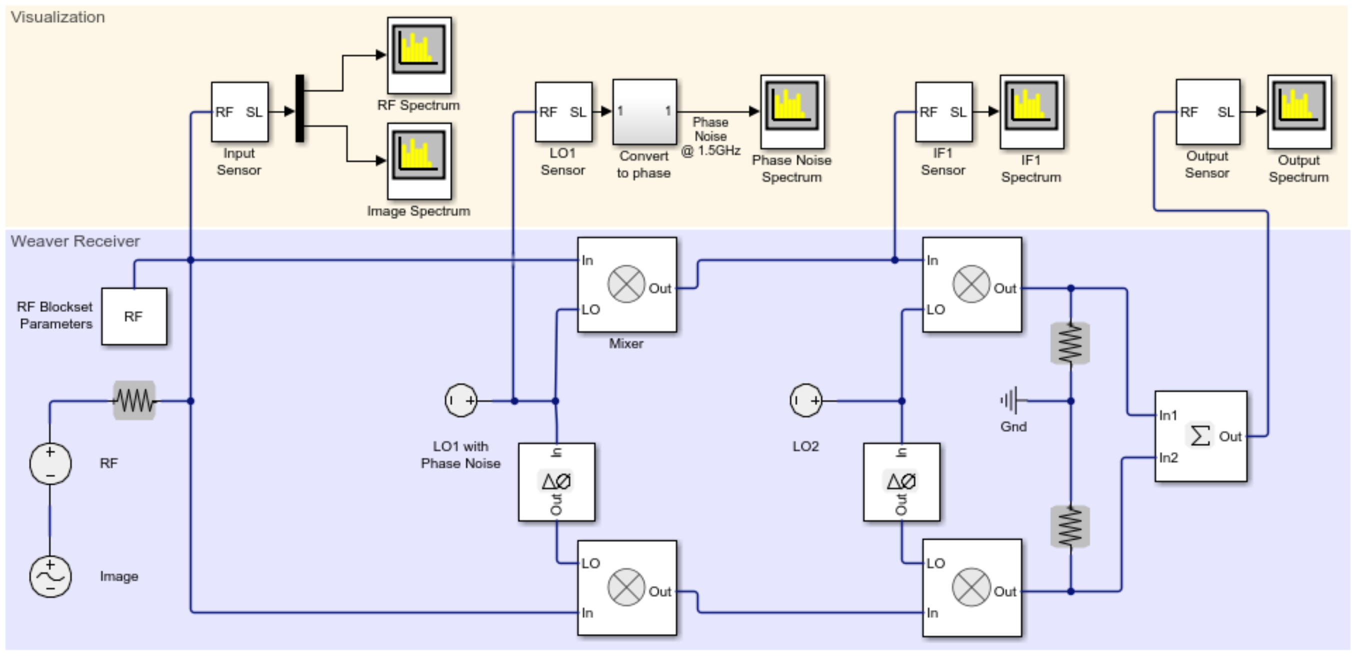


The preceding figure shows the transfer of phase noise from f_{LO1} to f_{IF1} .

Create Model with Phase Noise

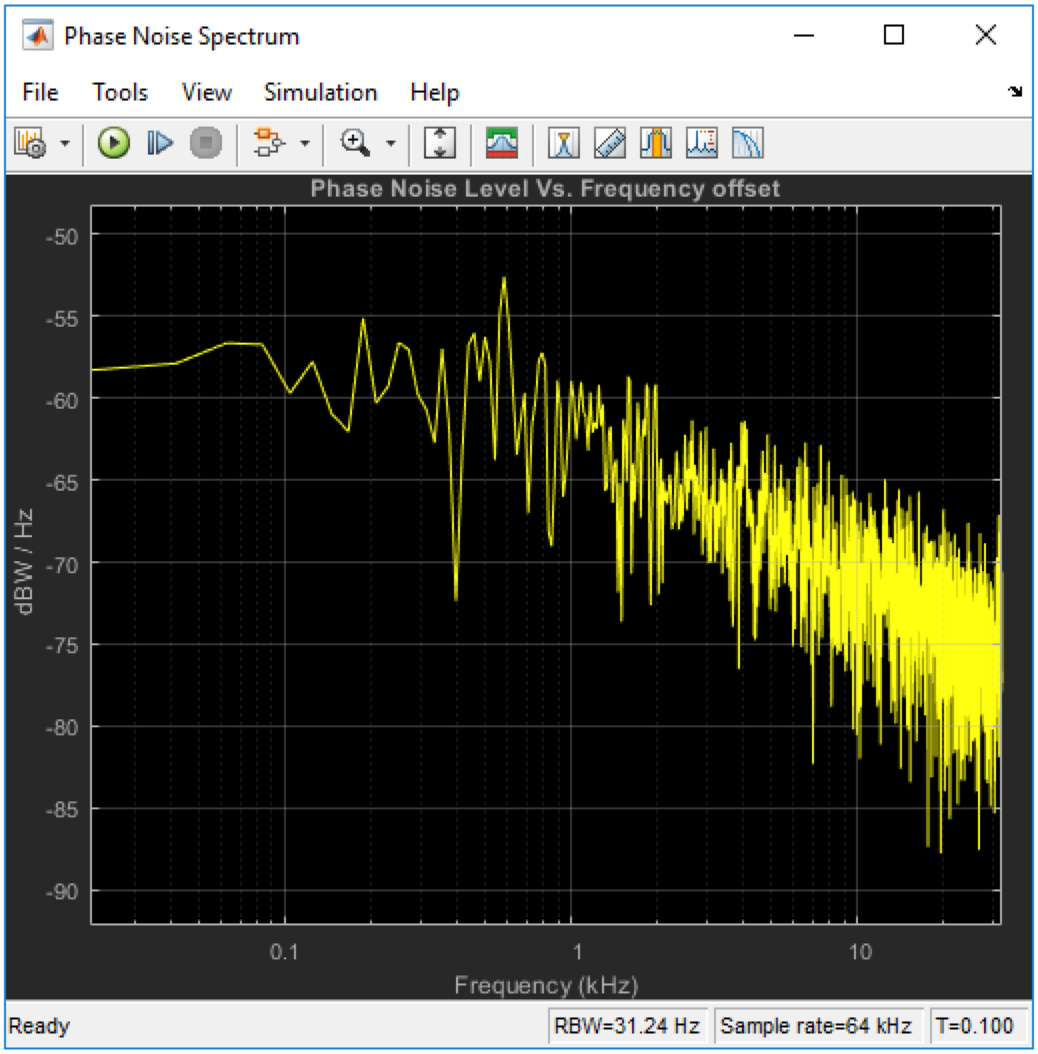
The model `ex_simrf_phase_noise` introduces phase noise into the model from the section `Create a Model with RF Interference`. The first mixing stage downconverts the RF and image to f_{IF}. To open the model:

```
open_system('ex_simrf_phase_noise')
```



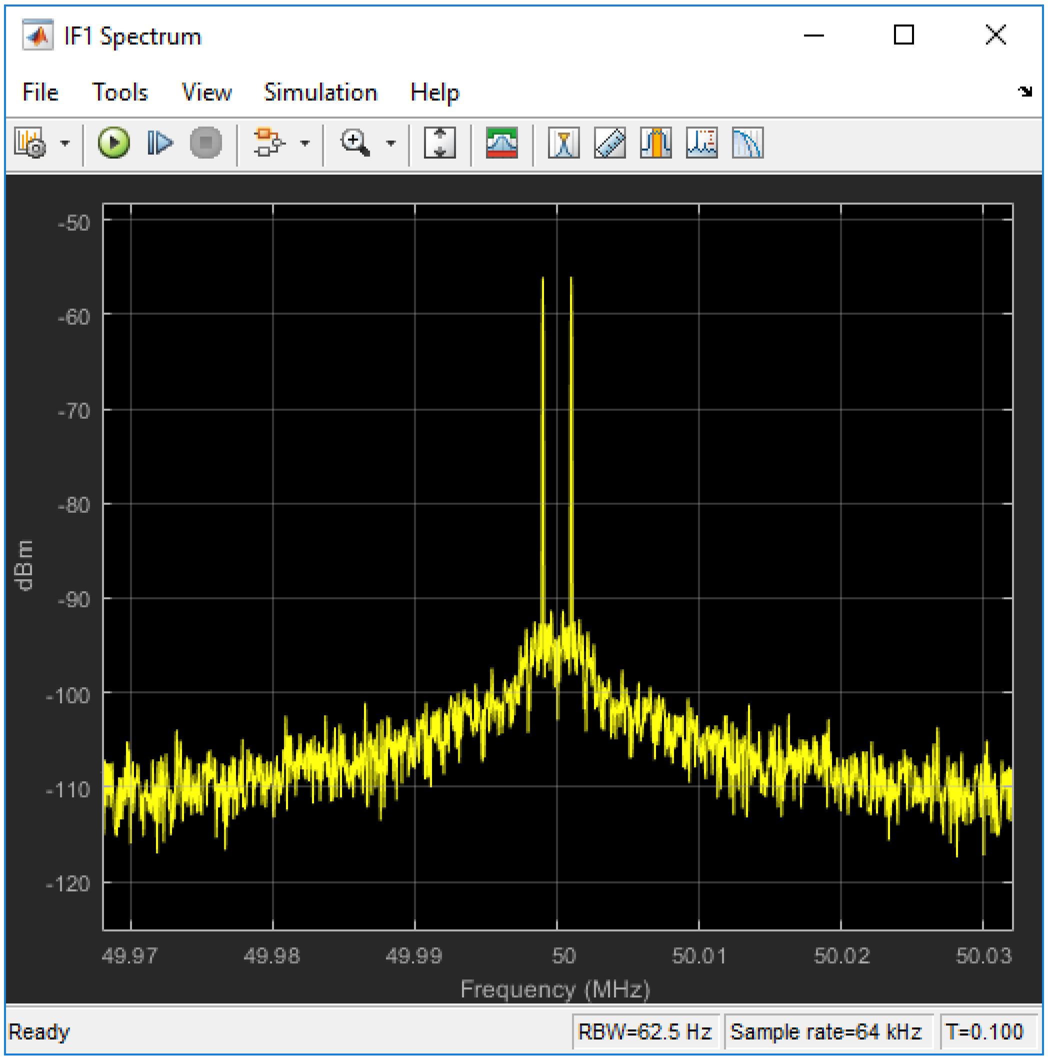
View Simulation Output

The model uses `Spectrum Analyzer` to generate 5 plots.



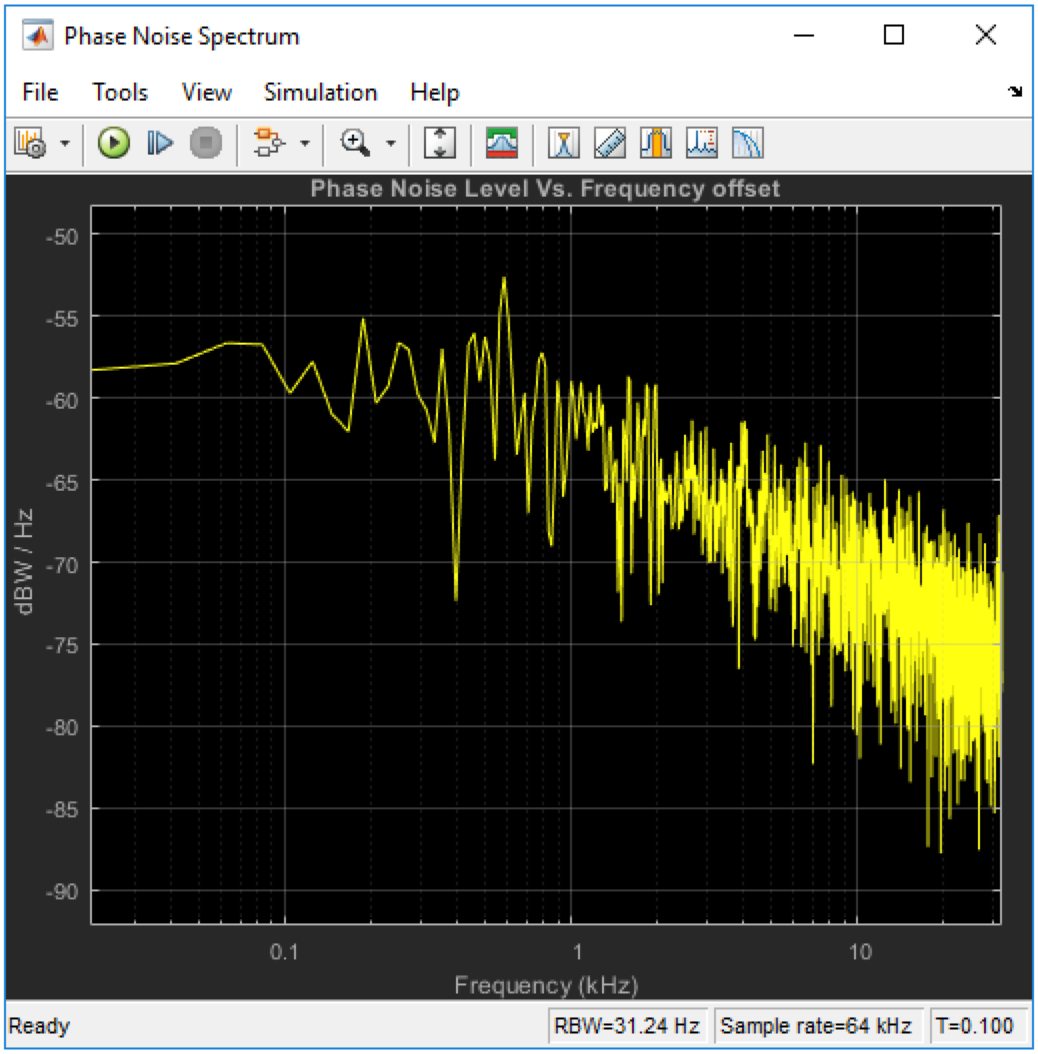
The Phase Noise spectrum scope shows a single-sided power density spectrum measuring the phase noise level at the LO1 source versus frequency offset shown in logarithmic scale.

The IF1 spectrum scope shows a power spectrum centered at the first intermediate frequency, measured between the first and second stages.



The scope shows that the LO phase noise has been transferred to the image. The RF signal on the carrier f_{IF1} is not visible in the figure because its power level is below the phase noise power of the downconverted image signal.

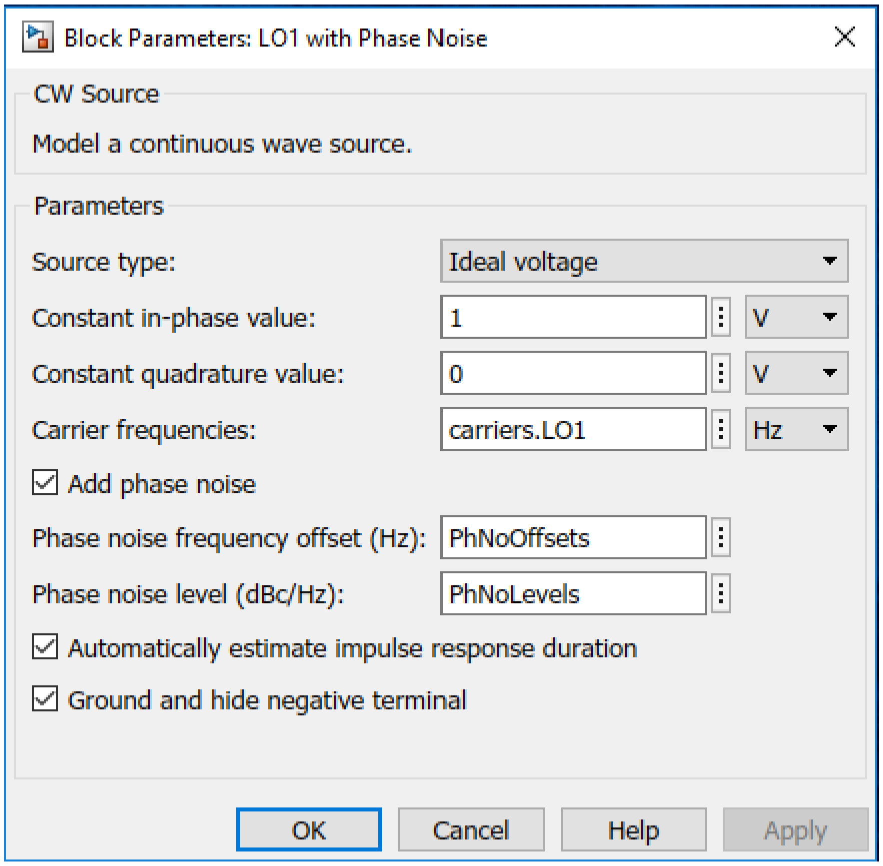
The Output spectrum scope shows the downconverted RF with the images removed.



The LO phase noise has been transferred to the receiver output.

Shaping LO Noise Skirt

As seen in the phase noise scope, the added phase noise is pink (1/f) and is specified within the CW source LO1. Specifically, the Add phase noise checkbox is checked in the blocks parameters dialog:



Block Parameters: LO1 with Phase Noise

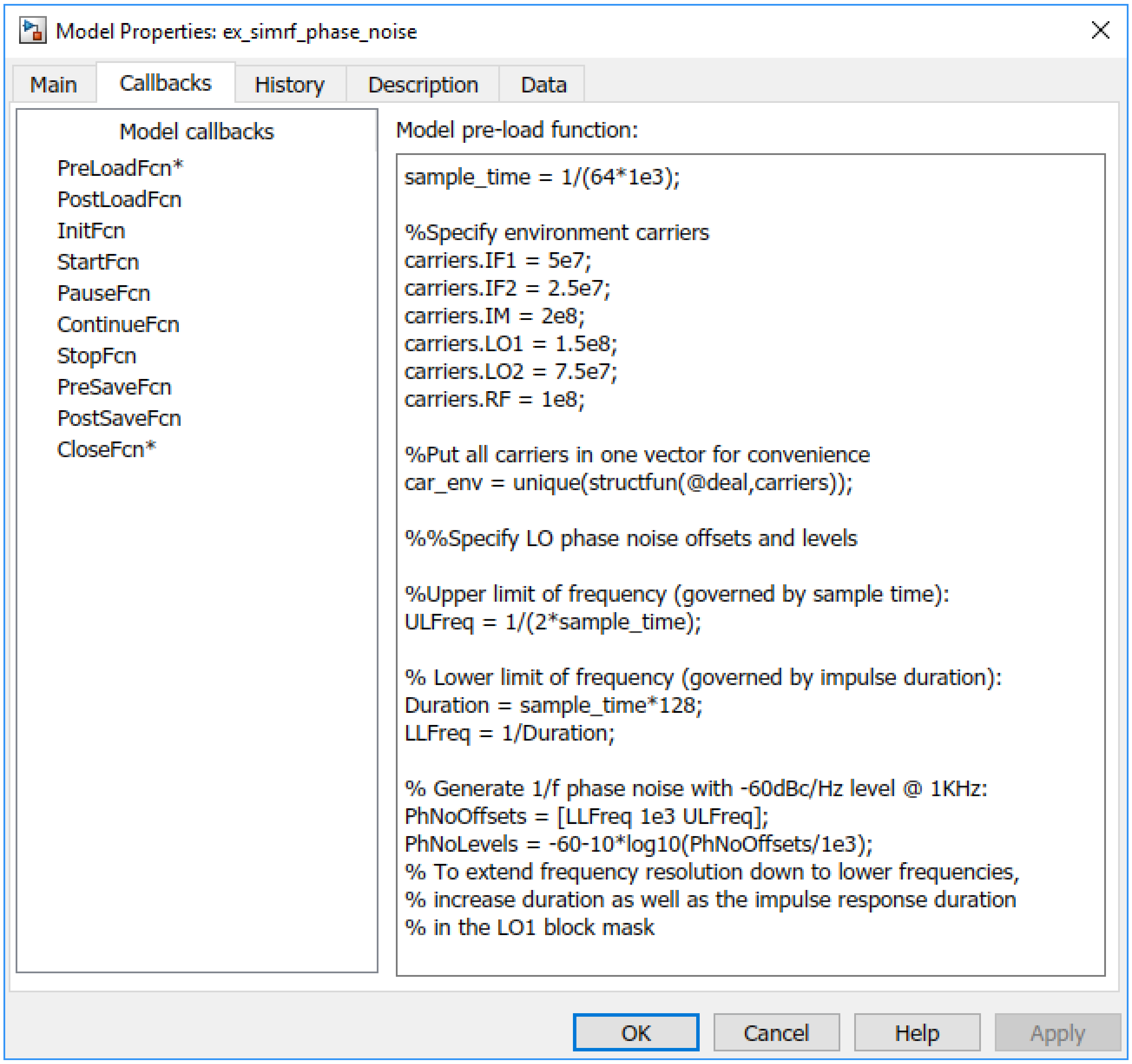
CW Source

Model a continuous wave source.

Parameters

Source type:	Ideal voltage	
Constant in-phase value:	1	V
Constant quadrature value:	0	V
Carrier frequencies:	carriers.LO1	Hz
☑ Add phase noise		
Phase noise frequency offset (Hz):	PhNoOffsets	
Phase noise level (dBc/Hz):	PhNoLevels	
☑ Automatically estimate impulse response duration		
☑ Ground and hide negative terminal		

The phase noise frequency offset and phase noise level variables PhNoOffsets and PhNoLevels are defined in the models PreLoadFcn callback, accessible through File > Model Properties > Model Properties:



```
sample_time = 1/(64*1e3);

%Specify environment carriers
carriers.IF1 = 5e7;
carriers.IF2 = 2.5e7;
carriers.IM = 2e8;
carriers.LO1 = 1.5e8;
carriers.LO2 = 7.5e7;
carriers.RF = 1e8;

%Put all carriers in one vector for convenience
car_env = unique(structfun(@deal,carriers));

%%Specify LO phase noise offsets and levels

%Upper limit of frequency (governed by sample time):
ULFreq = 1/(2*sample_time);

% Lower limit of frequency (governed by impulse duration):
Duration = sample_time*128;
LLFreq = 1/Duration;

% Generate 1/f phase noise with -60dBc/Hz level @ 1KHz:
PhNoOffsets = [LLFreq 1e3 ULFreq];
PhNoLevels = -60-10*log10(PhNoOffsets/1e3);
% To extend frequency resolution down to lower frequencies,
% increase duration as well as the impulse response duration
% in the LO1 block mask
```

The upper limit of the offset frequency is governed by the sample time and is limited to the envelope bandwidth of the simulation. The lower limit of the offset frequency is governed by the duration of the impulse response generating the phase noise frequency profile. Increasing the duration length in time steps from 128 to 256 will double the frequency resolution and allow simulation of the 1/f profile down to 250Hz (as opposed to the current lower limit of 500Hz).

See Also

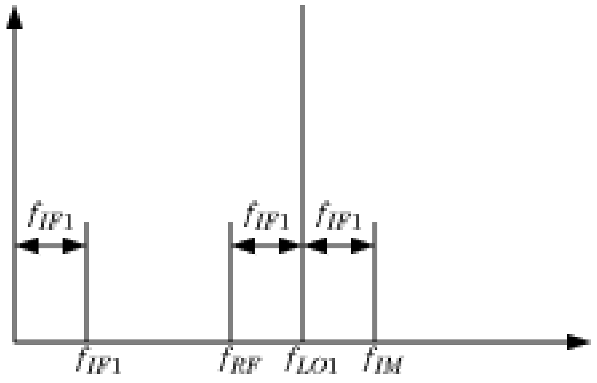
Noise Figure Testbench | Mixer

Related Topics

"RF Noise Modeling" on page 9-199

Carrier to Interference Performance of Weaver Receiver

A classic superheterodyne architecture filters images prior to frequency conversion. In contrast, image-reject receivers remove the images at the output without filtering but are sensitive to phase offsets.

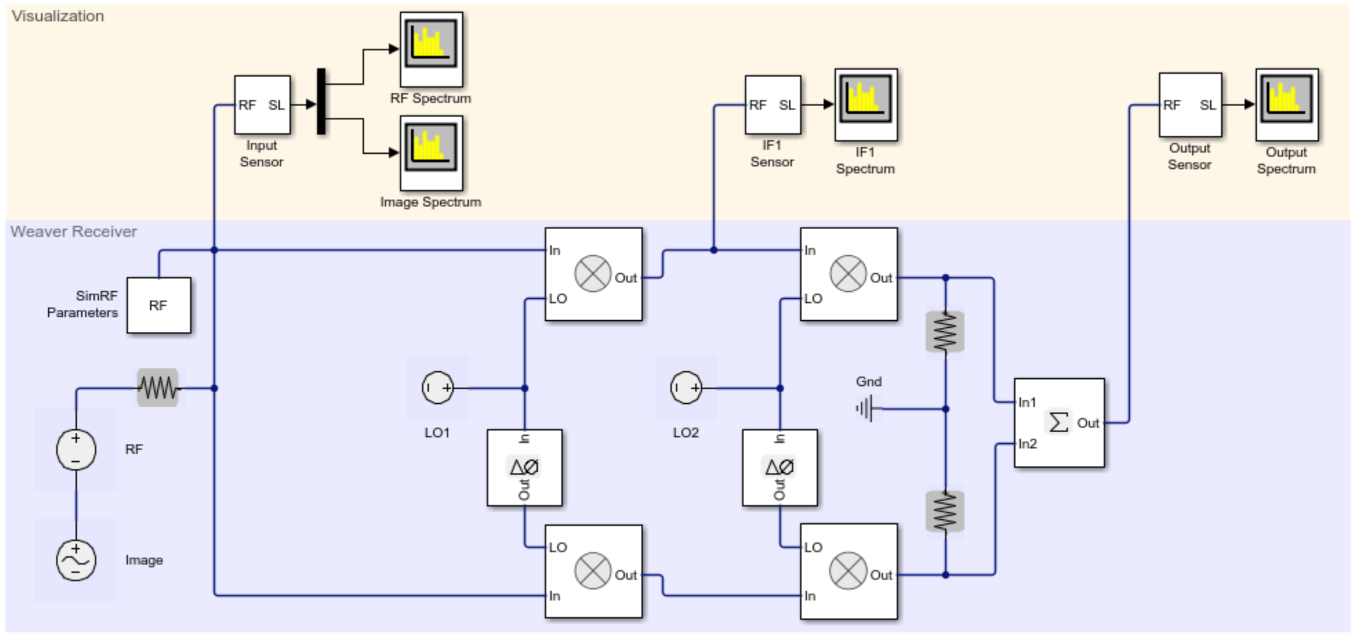


The preceding figure illustrates two input signals at the carriers f_{RF} and f_{IM} that both differ from the LO frequency, f_{LO1} , by an amount f_{IF1}. Mixing translates both input signals down to f_{IF1}. Perfect image rejection in the final stage of the receiver removes the image signal from the output entirely.

Create Model with RF Interference

The model `ex_simrf_ir` performs image rejection with a Weaver architecture. The receiver downconverts the signals f_{RF} and f_{IM} to f_{IF1} and f_{IF2} in two sequential stages.

```
open_system('ex_simrf_ir')
```



Set Up RF Blockset Environment

To maximize performance, the **Fundamental tones** and **Harmonic order** parameters are explicitly set in the `Configuration` block. To create the minimal set of simulation frequencies, the following carrier frequencies are set or created within the model.

- f_{RF}, the RF carrier, equals 100 MHz.
- f_{IM}, the image of the RF carrier relative to f_{LO1}, equals 200 MHz.
- f_{LO1}, the frequency of the LO in the first mixing stage, equals 150 MHz.
- f_{IF1}, the intermediate frequency of the signal after the first mixing stage equals: $f_{LO1} - f_{RF} = f_{IM} - f_{LO1}$ = 50 MHz.
- f_{LO2}, the frequency of the LO second mixing stage equals 75 MHz.
- f_{IF2}, the intermediate frequency of the signal after the second mixing stage equals: $f_{LO2} - f_{IF1}$ = 25 MHz.

In this system, every carrier is a multiple of f_{IF2}. The largest carrier, f_{IM}, is the 8th harmonic of f_{IF2}, so setting **Fundamental tones** to f_{IF2} and the **Harmonic order** to 8 ensures that every carrier is in the set of simulation frequencies.

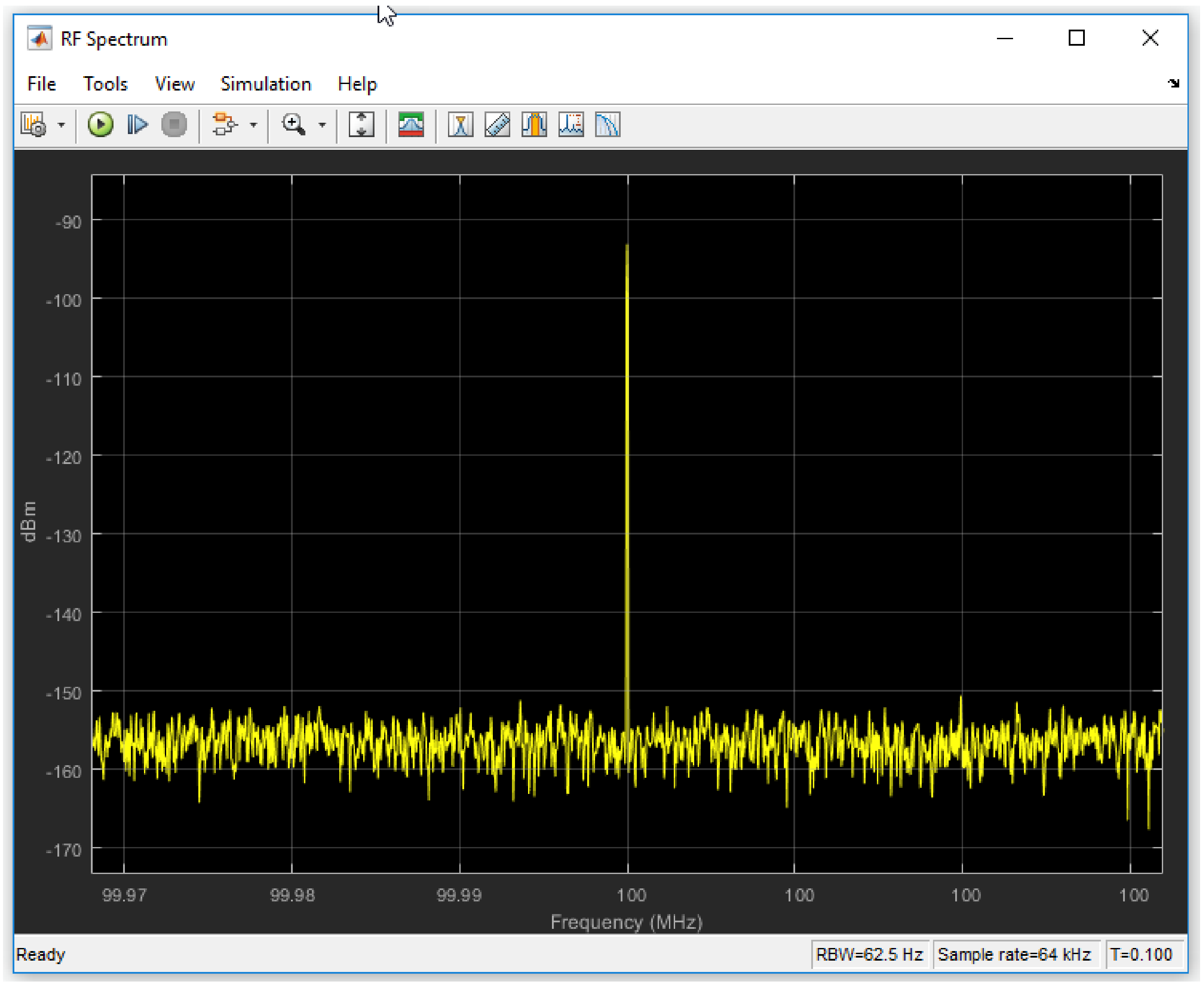
Solver conditions and noise settings are also specified for the `Configuration` block:

- The **Solver type** is set to auto. For more information on choosing solvers, see the reference page for the Configuration block or see Choosing Simulink and Simscape Solvers.
- The **Step size** parameter is set to 1/(mod_freq*64). This setting ensures a simulation bandwidth 64 times greater than the envelope signals in the system.
- The **Simulate noise** box is checked, so the environment includes noise in the simulation.

View Simulation Output

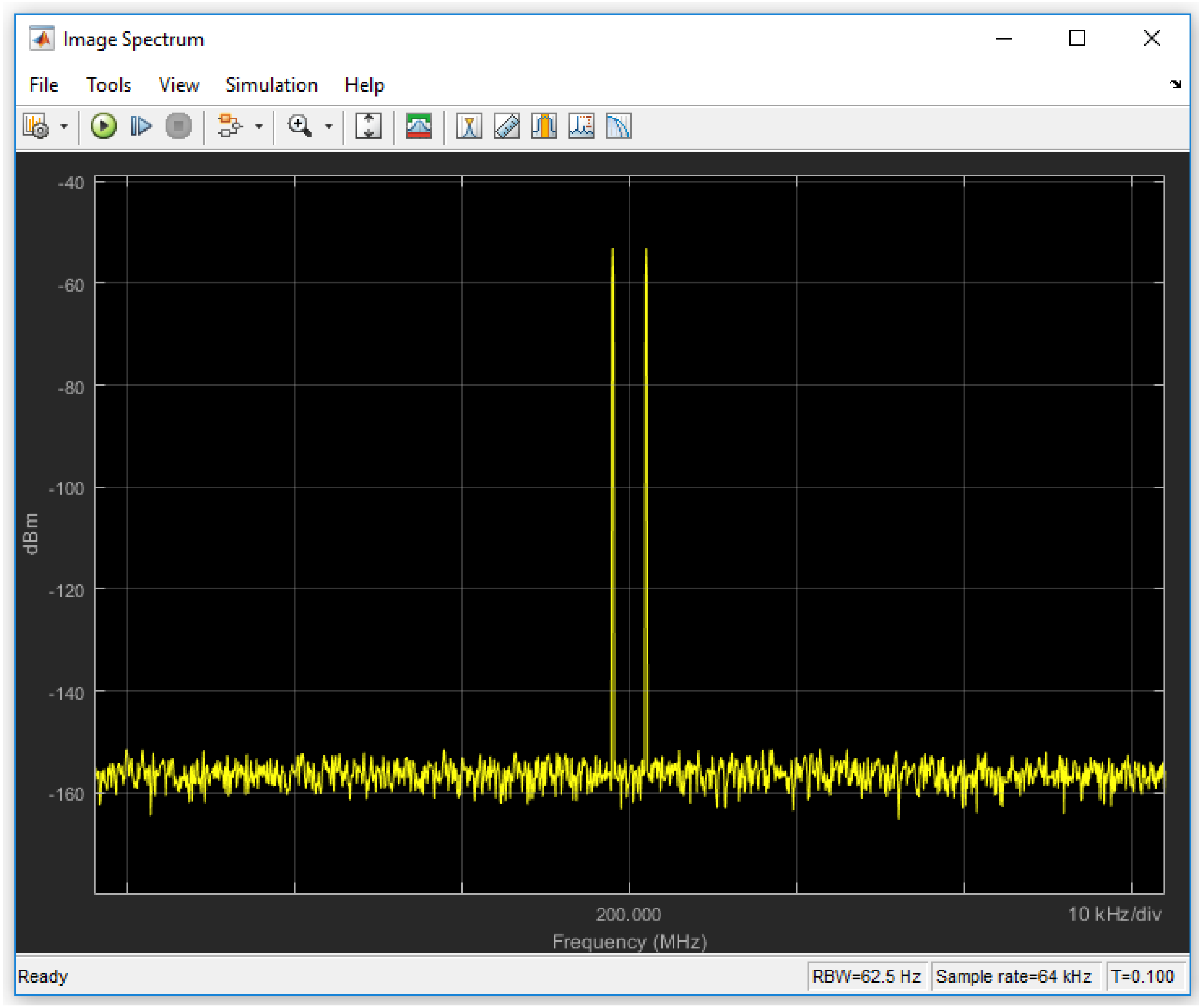
The model uses `Spectrum Analyzer` to generate four plots.

The RF spectrum scope shows the power spectrum of the carrier signal f_{RF}, specified as `carriers.RF` in the **Carrier frequencies** parameter of the Input Sensor `Outport` block.



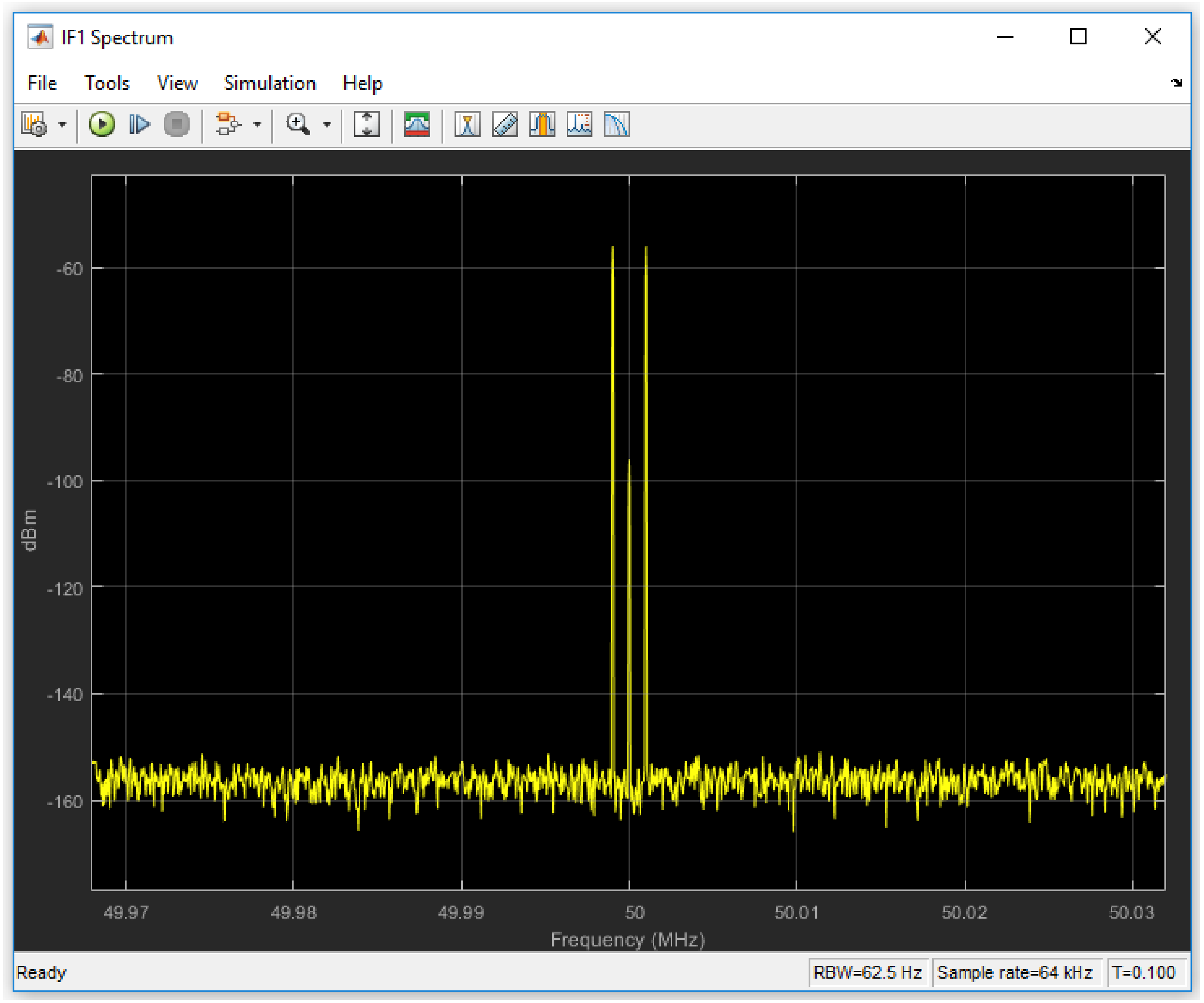
The modulation of the RF carrier is a constant envelope generated by a `Continuous Wave` block which generates a single peak centered at the carrier.

The Image Spectrum scope shows the power spectrum of the image. The signal is recovered from the carrier f_{IM}, specified as `carriers.IM` in the **Carrier frequencies** parameter of the Input Sensor `Outport` block.

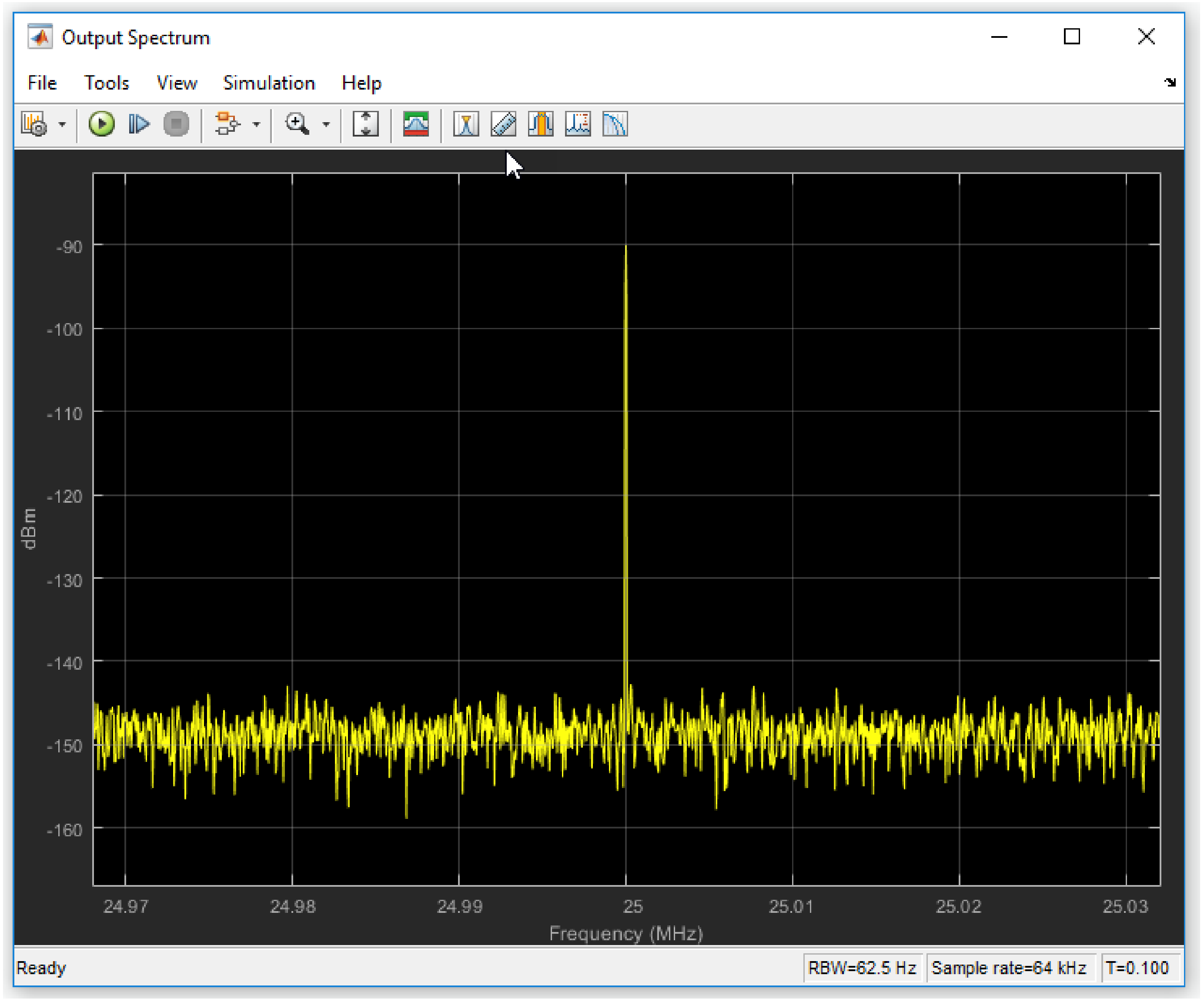


The Image `Sinusoidal Source` block generates a two-tone signal centered at f_{IM}.

The IF1 spectrum scope plot shows a power spectrum centered at the first intermediate frequency, measured between the first and second stages. The sensor outputs the modulation from the carrier f_{IF1}, specified as `carriers.IF1` in the **Carrier frequencies** parameter.



The Output spectrum scope shows the complete effects of the RF system. The sensor outputs the modulation from the carrier f_{IF2}, specified as `carriers.IF2` in the **Carrier frequencies** parameter.



Model RF and Blocker Sources

To model more robust input signals, you can use a RF Blockset `Inport` block to specify a circuit envelope signal generated using blocks from other Simulink™ libraries. For example, see the featured example `Impact of an RF Receiver on Communication System Performance`. This example uses Communications System Toolbox™ to model a QPSK-modulated waveform of random bits with RF Blockset Inport that brings the signal into the RF Blockset environment.

Simulating LO Phase Offset

The phase shifters have specified Phase shift parameters of 90. Deviation from this value results in a phase offset and causes imperfect image rejection. The featured example `Measuring Image Rejection Ratio in Receivers` analyzes the IRR of the Weaver and Hartley architectures several times, calculating the image rejection ratio (IRR) for several different phase offsets.

Demodulate Two-Tone RF Signal Using IQ Demodulator

Use the IQ Demodulator block to demodulate a two-tone RF signal to DC level. Observe the impairments in the demodulated output signal such as images due to gain imbalance, intermodulation distortion, and output third-order intercept (OIP3).

The two tones are at 10MHz and 15 MHz. The power of each tone is -30 dBm. The carrier frequency is 2 GHz.

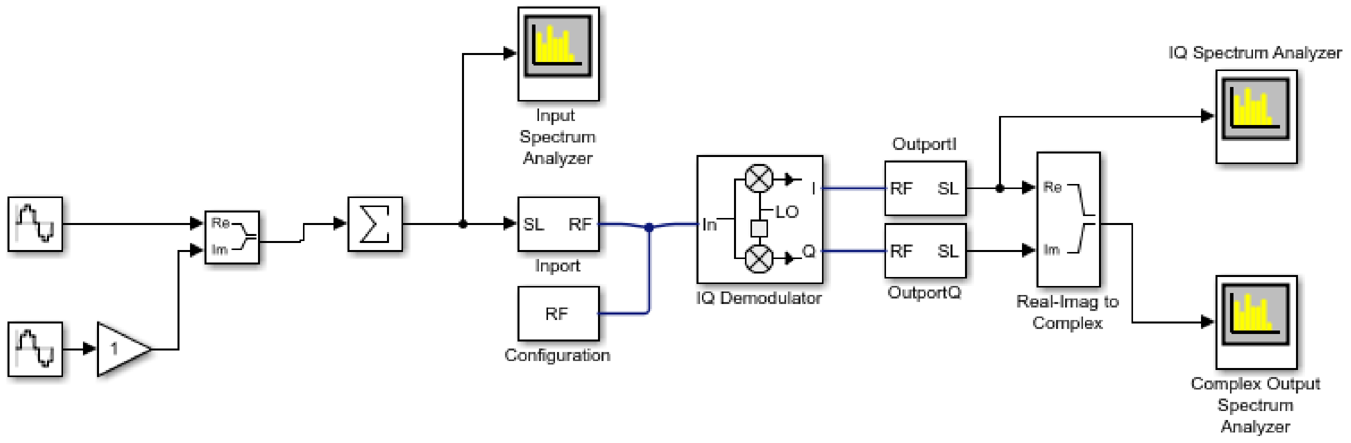
IQ Demodulator

The IQ Demodulator parameters are:

- Available power gain: 10 dB
- Local oscillator frequency: 2 GHz
- I/Q gain mismatch: 0.1 dB
- LO to RF Isolation: 90 dB
- Noise Figure: 6 dBm/Hz

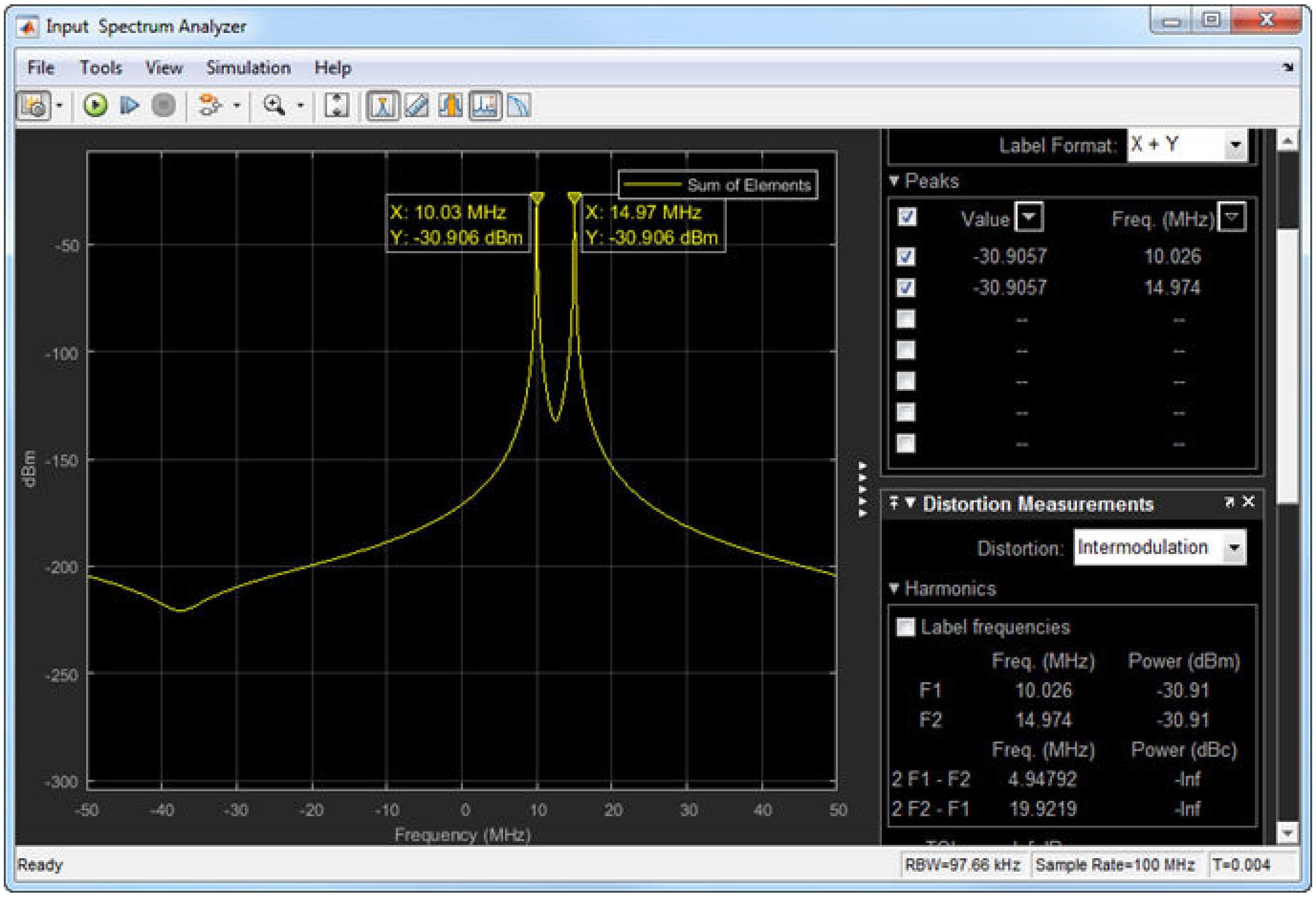
Open the model.

```
open('model_IQdemod')
```



Run the model and observe the spectrum analyzers.

Input Spectrum Analyzer



Label Format: X + Y

▼ Peaks

✓	Value	Freq. (MHz)
✓	-30.9057	10.026
✓	-30.9057	14.974
☐	--	--
☐	--	--
☐	--	--
☐	--	--
☐	--	--

▼ Distortion Measurements

Distortion: Intermodulation

▼ Harmonics

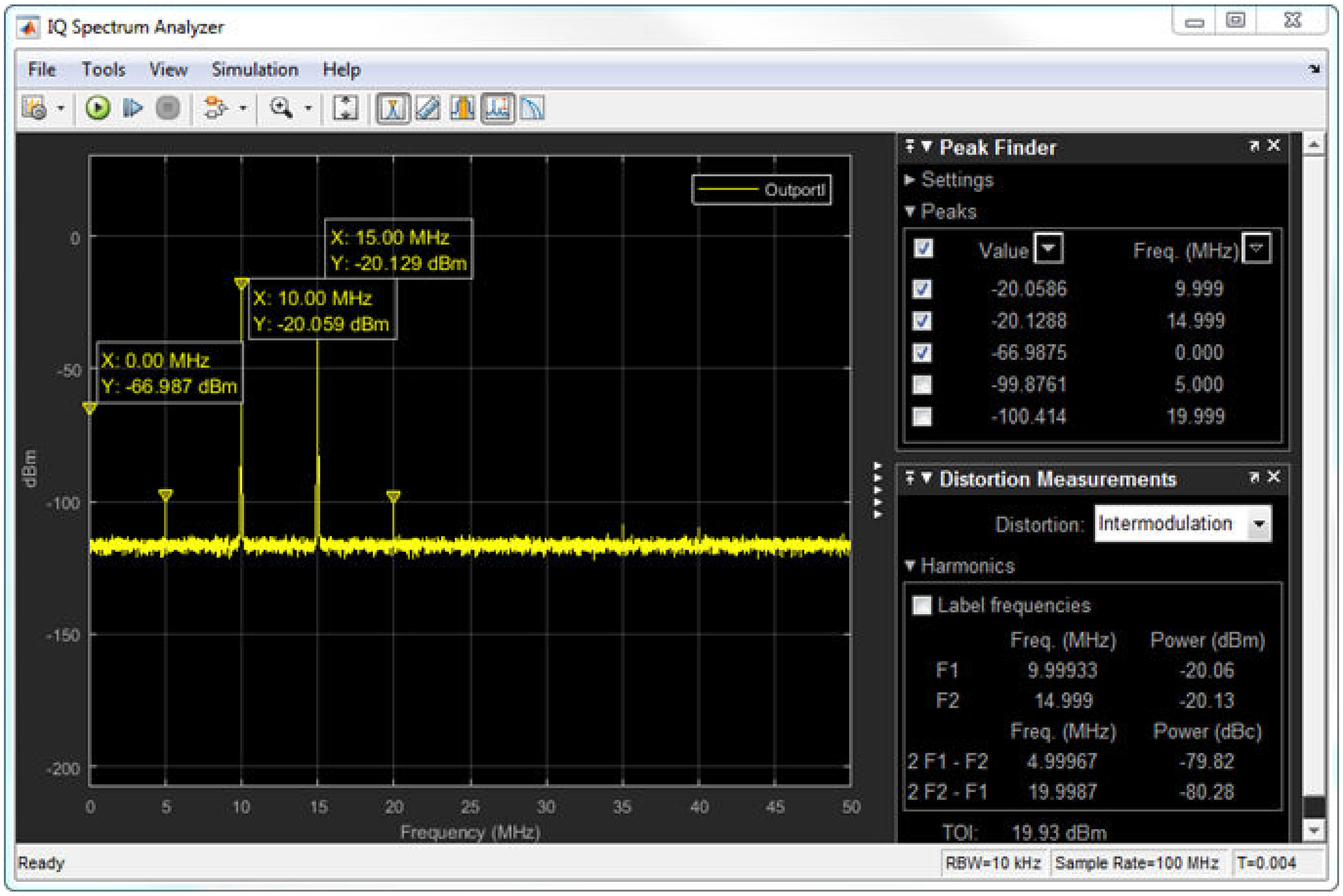
☐ Label frequencies

	Freq. (MHz)	Power (dBm)
F1	10.026	-30.91
F2	14.974	-30.91
	Freq. (MHz)	Power (dBc)
2 F1 - F2	4.94792	-Inf
2 F2 - F1	19.9219	-Inf

RBW=97.66 kHz Sample Rate=100 MHz T=0.004

In the input spectrum analyzer, you see the input RF signal with the two tones at 10 MHz and 15 MHz. The power level of each tone is -30 dBm. The carrier frequency is 2 GHz.

I/Q Spectrum Analyzer



✓	Value	Freq. (MHz)
✓	-20.0586	9.999
✓	-20.1288	14.999
✓	-66.9875	0.000
	-99.8761	5.000
	-100.414	19.999

Distortion: Intermodulation

	Freq. (MHz)	Power (dBm)
F1	9.99933	-20.06
F2	14.999	-20.13
	Freq. (MHz)	Power (dBc)
2 F1 - F2	4.99967	-79.82
2 F2 - F1	19.9987	-80.28

TOI: 19.93 dBm

RBW=10 kHz Sample Rate=100 MHz T=0.004

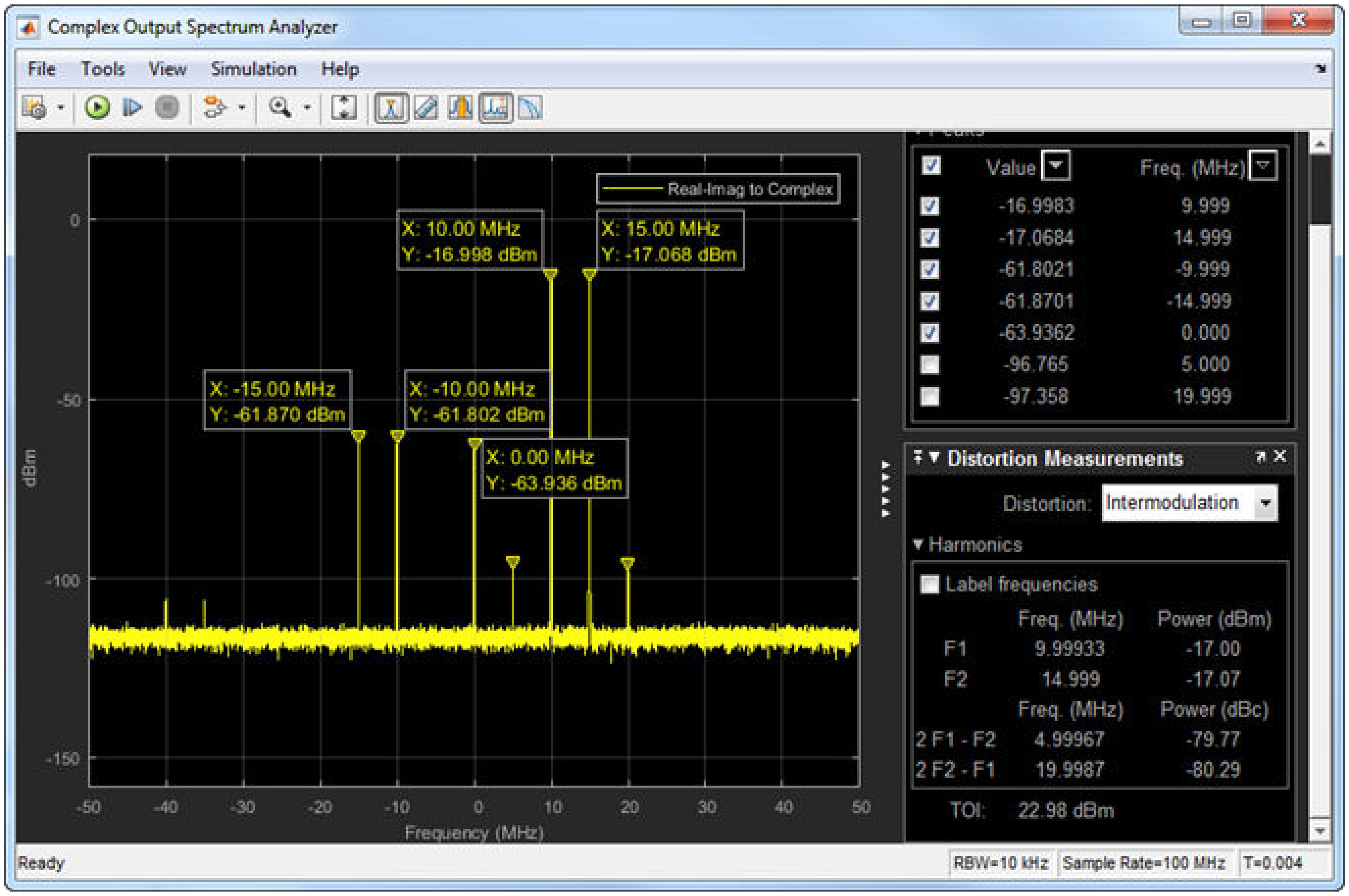
In the I/Q spectrum analyzer, you see the inphase part of the demodulated signal including the DC signal level and the two tones. The following formula gives you the DC power level of the signal:

$$DClevelI/Q = PowerLO - dBmIsolation + Gain$$

$$DclevelI/Q = 20 * log(1/sqrt(50)) + 30 - 90 + 10 = -67dBm$$

The output power level of the two tones are -20 dBm. You also see the OIP3 value (measured by the spectrum analyzer) at approximately 20 dBm.

Complex Output Spectrum Analyzer



✓	Value	Freq. (MHz)
✓	-16.9983	9.999
✓	-17.0684	14.999
✓	-61.8021	-9.999
✓	-61.8701	-14.999
✓	-63.9362	0.000
	-96.765	5.000
	-97.358	19.999

Distortion Measurements

Distortion: Intermodulation

Harmonics

Label frequencies

	Freq. (MHz)	Power (dBm)
F1	9.99933	-17.00
F2	14.999	-17.07
	Freq. (MHz)	**Power (dBc)**
2 F1 - F2	4.99967	-79.77
2 F2 - F1	19.9987	-80.29

TOI: 22.98 dBm

In the complex output spectrum analyzer, you see the whole demodulated signal including the imaginary parts. The output power level of the two tones (10 MHz and 15 MHz) is -17 dBm.

Image Rejection Ratio

The images of the two tones are at -10 MHz and -15 MHz. The output power level of the images are -61.78 dBm. Image rejection ratio is given by the formula:

$$IMRR = [(gainimbalance)^2 + 1 - 2 * (gainimbalance)]/[(gainimbalance)^2 + 1 + 2 * (gainimbalance)]$$

$$GainImbalance = 10^{(}0.1/20) = 1.0116$$

$$IMRRdB = 10 * log10(3.3253e - 05) = -44.78dB$$

$$Imagelevel = -17dBm - 44.78dB = -61.78dBm$$

See Also

IQ Demodulator | Configuration | Inport | Outport

Related Topics

"Modulate Two-Tone DC Signal Using IQ Modulator" on page 8-51

Modulate Two-Tone DC Signal Using IQ Modulator

Use the IQ Modulator block to Modulate a two-tone DC signal to RF level. Observe the impairments in the modulated output signal such as images due to gain imbalance, intermodulation distortion, and output third-order intercept (OIP3).

The two tones are at 10MHz and 15 MHz.The power of each tone is -30 dBm. The carrier frequency is 0 GHz.

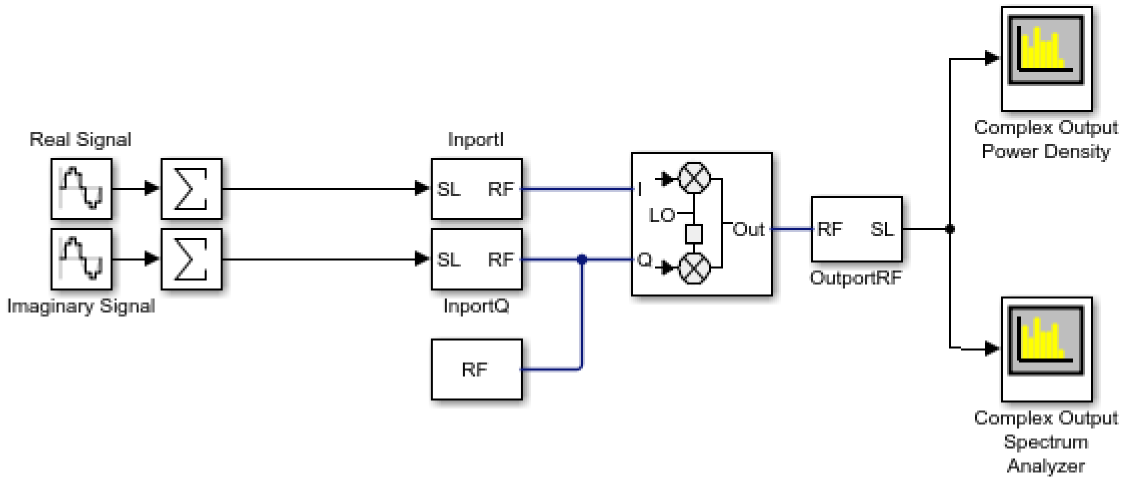
IQ Modulator

The IQ modulator parameters are :

- Available power gain: 10 dB
- Local oscillator frequency: 2 GHz
- I/Q gain mismatch: 0.1 dB
- LO to RF Isolation: 90 dB
- Noise Floor: -160 dBm/Hz
- IP3: 10 dBm

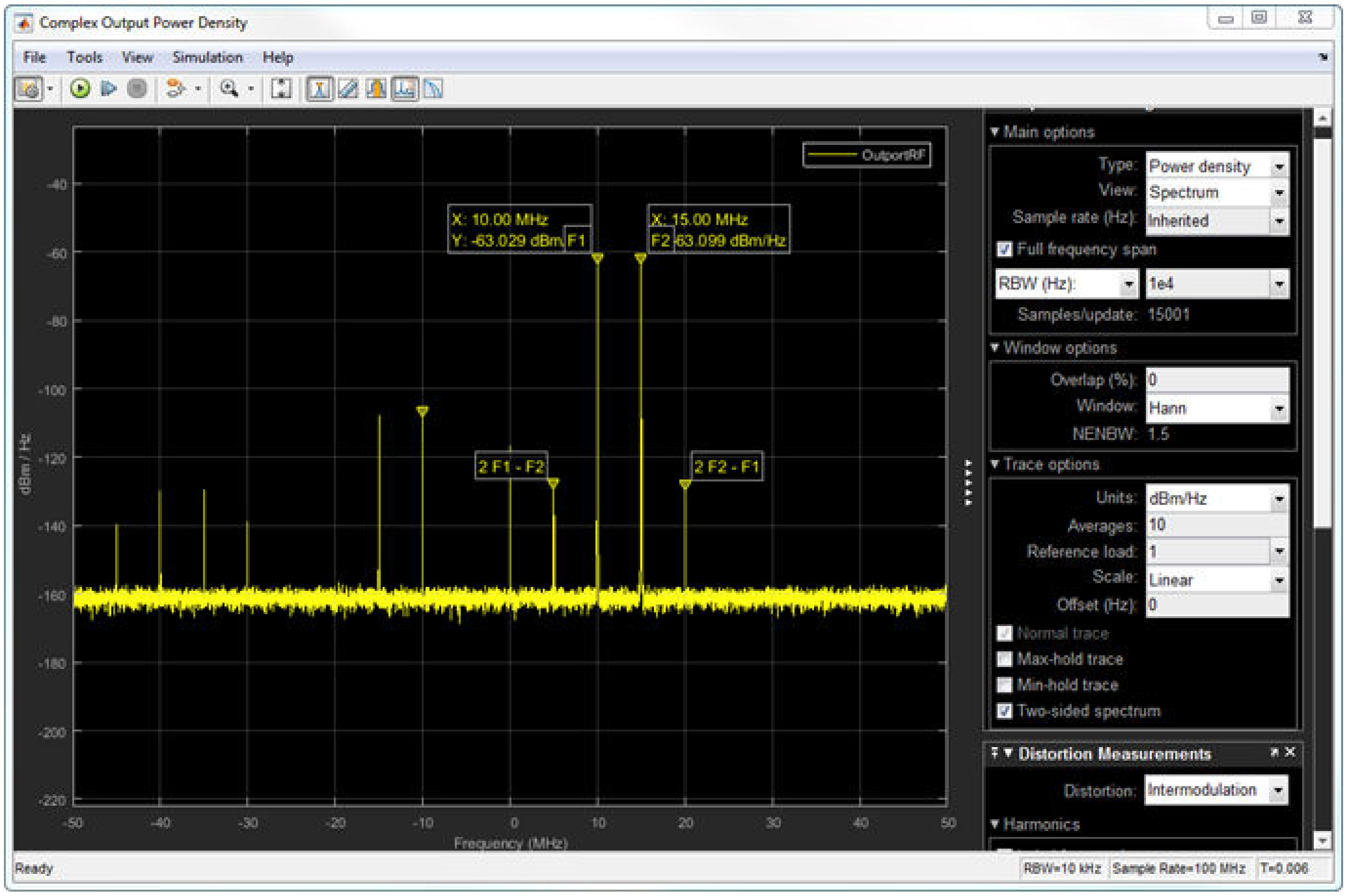
Open the model.

```
open('model_IQmod')
```



Run the model and observe the spectrum analyzers.

Complex Output Power Density



▾ Main options

Type: Power density

View: Spectrum

Sample rate (Hz): Inherited

☑ Full frequency span

RBW (Hz): 1e4

Samples/update: 15001

▾ Window options

Overlap (%): 0

Window: Hann

NENBW: 1.5

▾ Trace options

Units: dBm/Hz

Averages: 10

Reference load: 1

Scale: Linear

Offset (Hz): 0

☑ Normal trace

☐ Max-hold trace

☐ Min-hold trace

☑ Two-sided spectrum

▾ Distortion Measurements

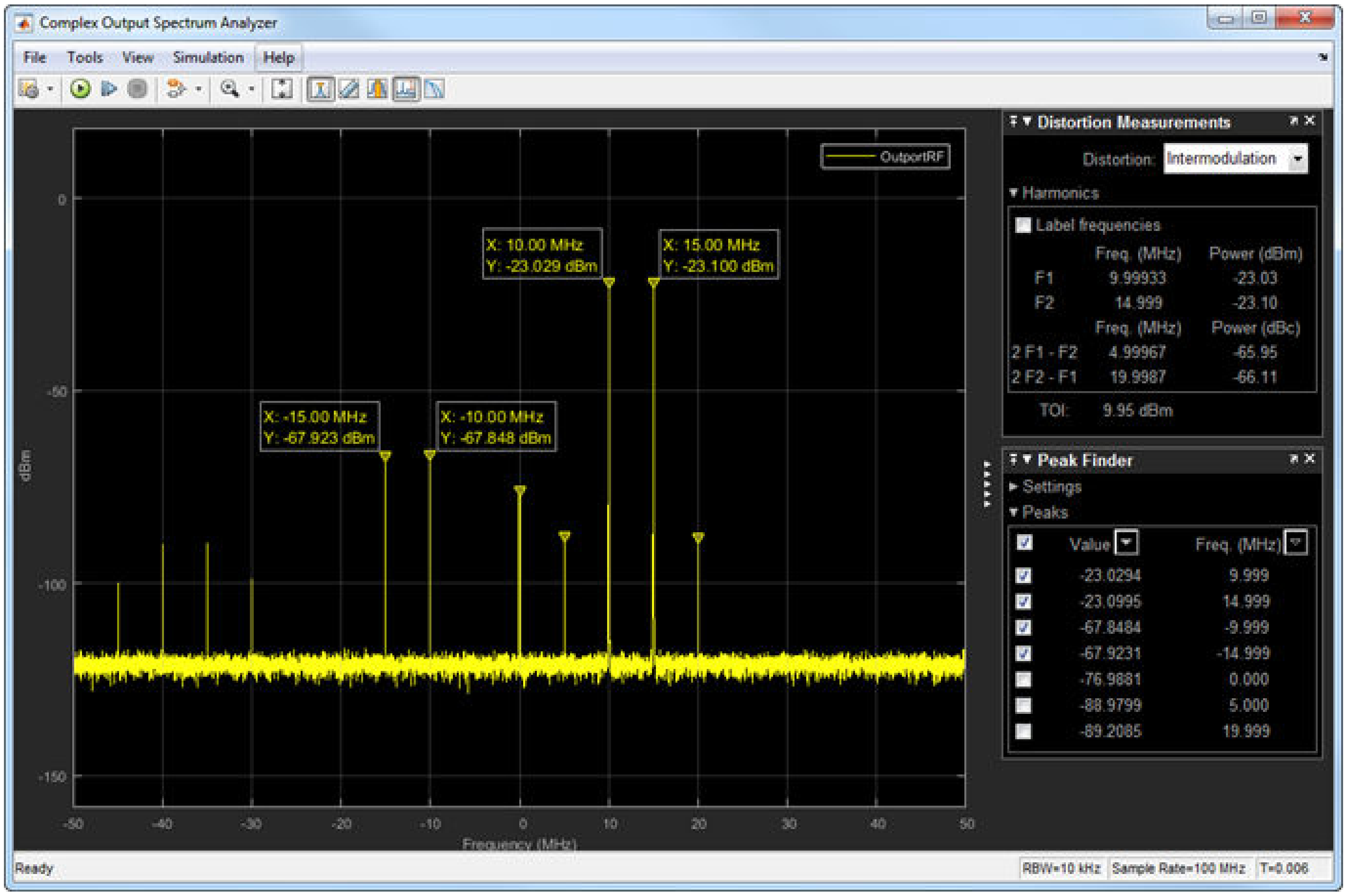
Distortion: Intermodulation

▾ Harmonics

RBW=10 kHz Sample Rate=100 MHz T=0.006

In the complex output power density spectrum analyzer, you see the noise floor of the signal at -160 dBm/Hz.

Complex Output Spectrum Analyzer



Distortion Measurements

Distortion: Intermodulation

Harmonics

Label frequencies

	Freq. (MHz)	Power (dBm)
F1	9.99933	-23.03
F2	14.999	-23.10
	Freq. (MHz)	Power (dBc)
2 F1 - F2	4.99967	-65.95
2 F2 - F1	19.9987	-66.11

TOI: 9.95 dBm

Peak Finder

Settings

Peaks

✓	Value	Freq. (MHz)
✓	-23.0294	9.999
✓	-23.0995	14.999
✓	-67.8484	-9.999
✓	-67.9231	-14.999
☐	-76.9881	0.000
☐	-88.9799	5.000
☐	-89.2085	19.999

Ready RBW=10 kHz Sample Rate=100 MHz T=0.006

In the complex output spectrum analyzer, you see the whole modulated signal including the imaginary parts. The output power level of the two tones (10 MHz and 15 MHz)is -20 dBm.

$$Outputpowerlevel = Inputpowerlevel + gain = -30dBm + 10dB = -20dBm$$

The output third-order intercept(OIP3) is at 10 dBm. The spectrum analyzer measures this value.

Image Rejection Ratio

The images of the two tones are at -10 MHz and -15 MHz. The output power level of the two images are -67.8 dBm. Image rejection ratio is given by the formula:

$$IMRR = [(gainimbalance)^2 + 1 - 2 * (gainimbalance)]/[(gainimbalance)^2 + 1 + 2 * (gainimbalance)]$$

where,

$$GainImbalance = 10^(0.1/20) = 1.0116$$

$$IMRRdB = 10 * log10(3.3253e - 05) = -44.78dB$$

The image power level is given by the formula:

$$Imagelevel = -23dBm - 44.78dB = -67.78dBm$$

See Also

IQ Modulator

Related Topics

"Demodulate Two-Tone RF Signal Using IQ Demodulator" on page 8-46

Measurement of Gain and Noise Figure Spectrum

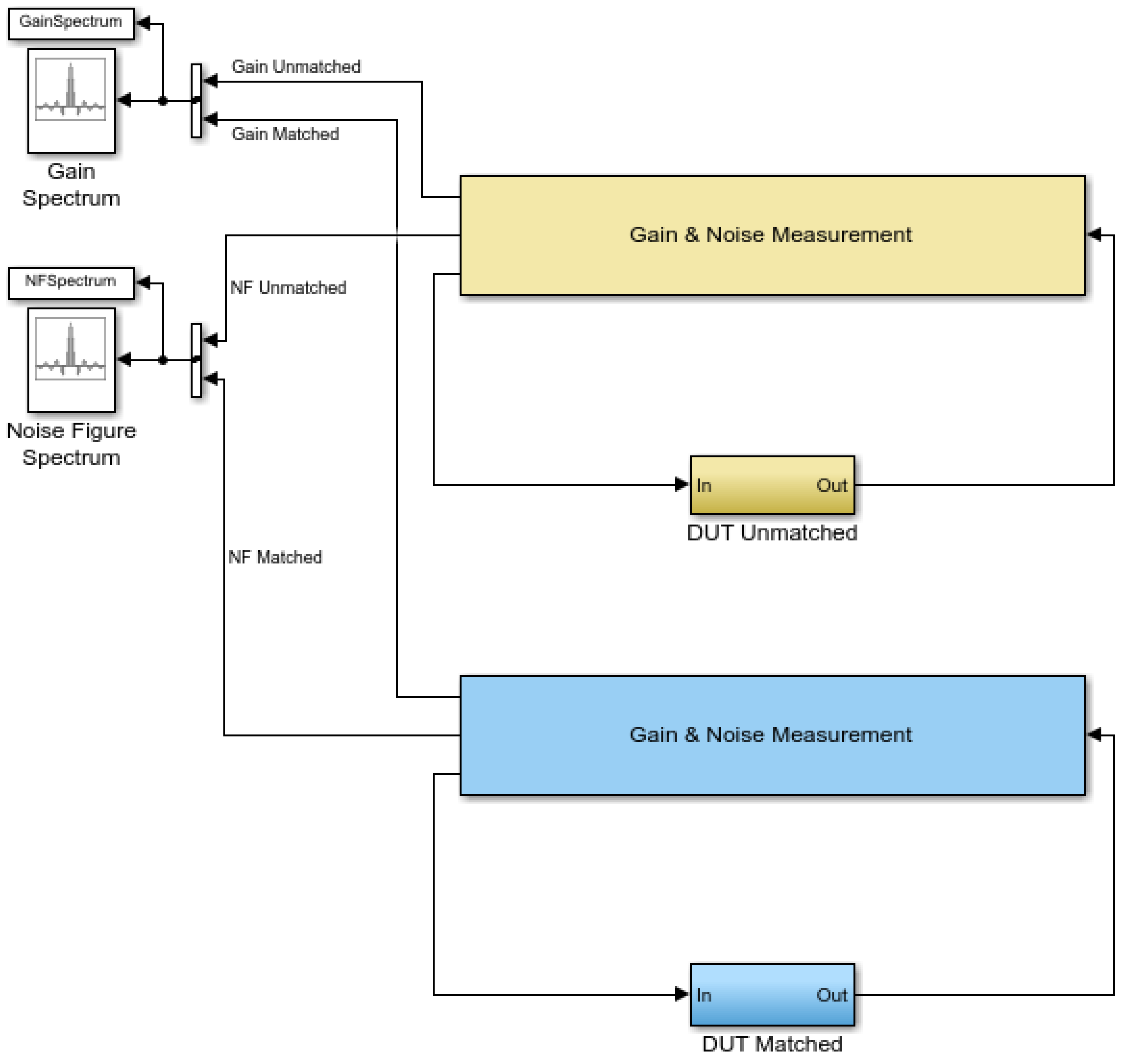
This example shows how to use RF Blockset to measure the Gain and Noise Figure of an RF system over a given spectral range.

The example requires DSP System Toolbox™.

Introduction

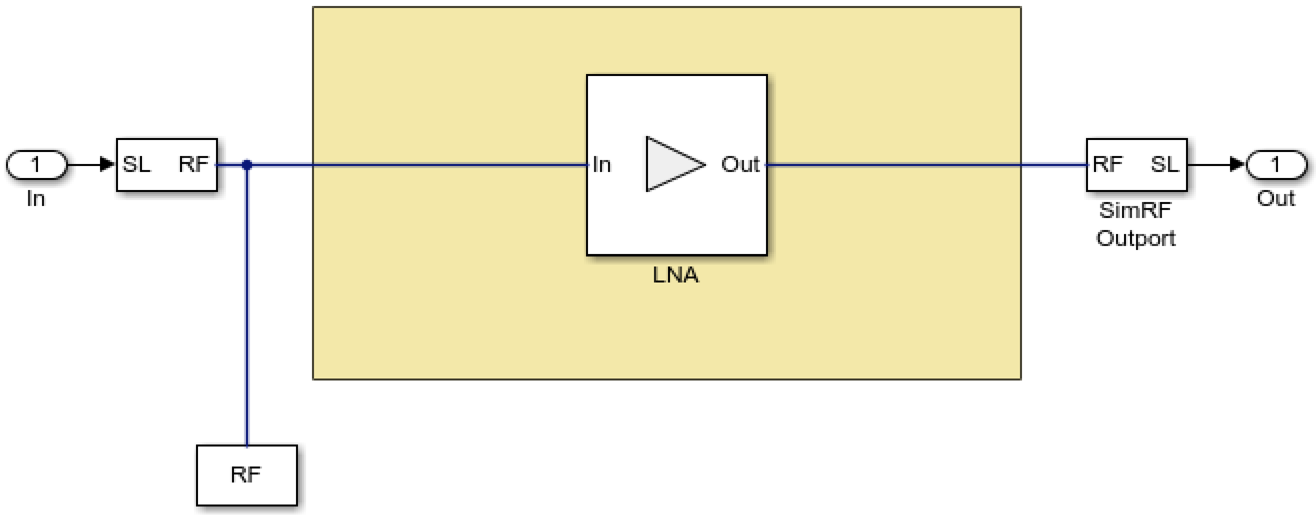
In this example, a method for measuring the frequency-dependent gain and noise figure of an RF system is described. These spectral properties are measured for two RF systems; A single Low Noise Amplifier and the same amplifier when matched. The model used for the measurement is shown below:

```
model = 'GainNoiseMeasurementExample';
open_system(model);
```



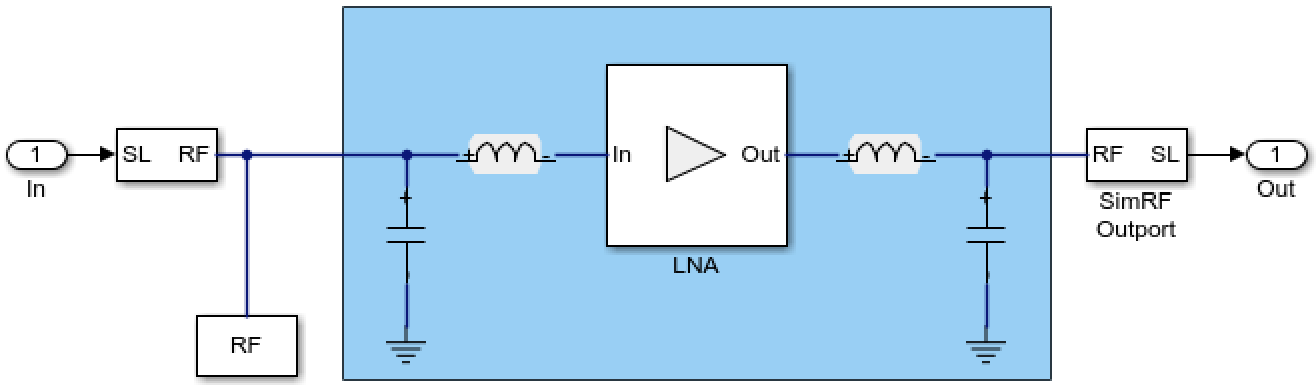
The model has two measurement units, each connected to a different subsystem containing the DUT. The upper measurement unit is connected to an unmatched LNA in the DUT subsystem with yellow background:

```
open_system([model '/DUT Unmatched']);
```



The lower measurement unit is connected to a matched LNA in the DUT subsystem with blue background:

```
open_system([model '/DUT Matched']);
```



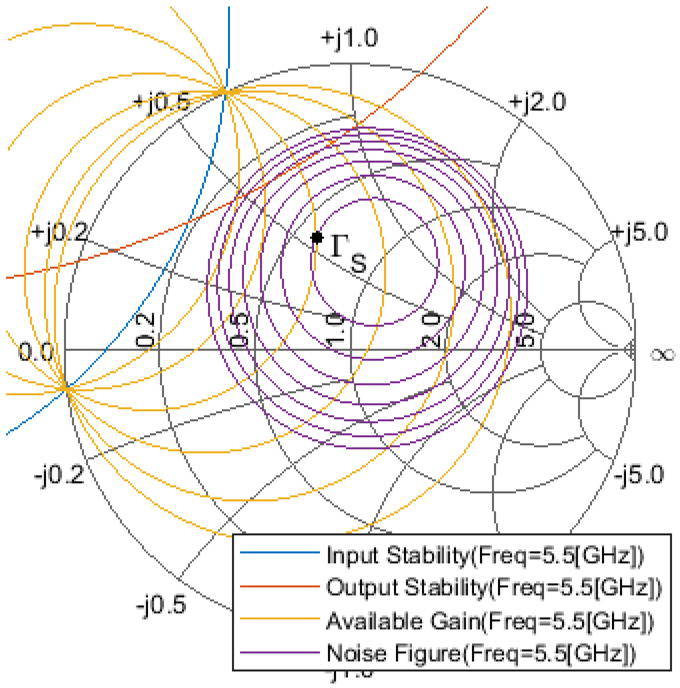
Each measurement unit outputs two vector signals representing the spectrums of the Gain and Noise Figure of the corresponding DUT and those are inputted into two Array Plot blocks that plot the above properties versus frequency, comparing the unmatched and matched DUT systems. In the following sections, the matching network design process is described, the simulation results are given and compared with these expected from LNA and matching network properties. Finally, the procedure used within the measurement units to obtain the spectral Gain and Noise results is explained.

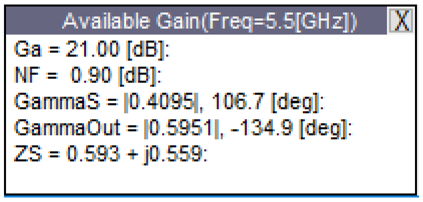
Design of the matching network

The matching network used in the matched DUT subsystem comprises a single stage L-C network that is designed following the same procedure as the one described in the RF Toolbox example "Design Matching Networks - Part 1". Since the LNA used here is different, the design is described below

Initially, an `rfckt.amplifier` object is created to represent an Heterojunction Bipolar Transistor based low noise amplifier that is specified in the file, 'RF_HBT_LNA.S2P'. Then, the `circle` method of the `rfckt.amplifier` object is used to place the constant available gain and the constant noise figure circles on a Smith chart, and select an appropriate source reflection coefficient, GammaS, that provides a suitable compromise between gain and noise. The GammaS value chosen yields an available gain of Ga=21dB, and a noise figure of NF=0.9dB at the center frequency fc=5.5GHz:

```
unmatched_amp = read(rfckt.amplifier, 'RF_HBT_LNA.S2P');
fc = 5.5e9; % Center frequency (Hz)
circle(unmatched_amp,fc,'Stab','In','Stab','Out','Ga',15:2:25, ...
    'NF',0.9:0.1:1.5);

% Choose GammaS and show it on smith chart:
hold on
GammaS = 0.411*exp(1j*106.7*pi/180);
plot(GammaS,'k.','MarkerSize',16)
text(real(GammaS)+0.05,imag(GammaS)-0.05,'\Gamma_{S}','FontSize', 12, ...
    'FontUnits','normalized')
hLegend = legend('Location','SouthEast');
hLegend.String = hLegend.String(1:end-1);
hold off
```





```
Available Gain(Freq=5.5[GHz])
Ga = 21.00 [dB]:
NF =  0.90 [dB]:
GammaS = |0.4095|, 106.7 [deg]:
GammaOut = |0.5951|, -134.9 [deg]:
ZS = 0.593 + j0.559:
```

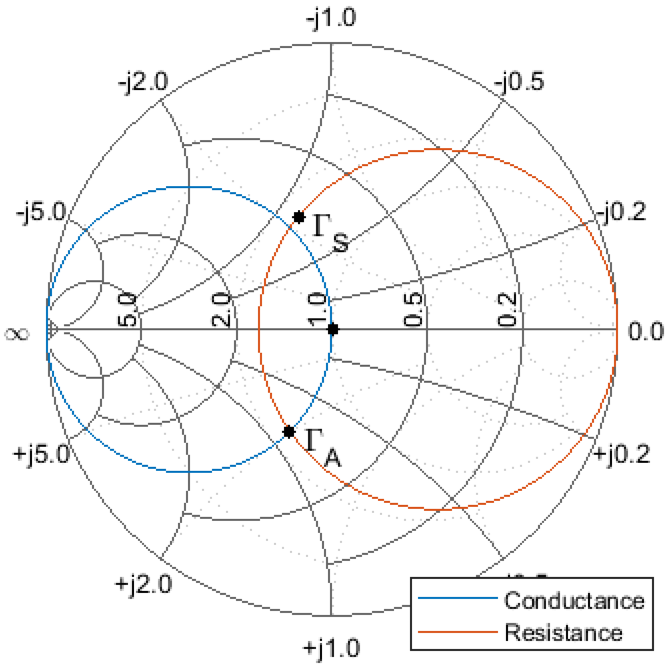
For the chosen GammaS, the following properties can be obtained:

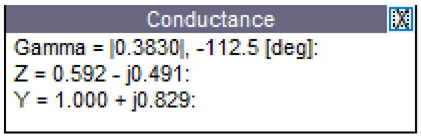
```
% Normalized source impedance:
Zs = gamma2z(GammaS,1);

% Matching |GammaL| that is equal to the complex conjugate of
% |GammaOut| shown on the data tip:
GammaL = 0.595*exp(1j*135.0*pi/180);

% Normalized load impedance:
Zl = gamma2z(GammaL,1);
```

The input matching network consists of one shunt capacitor, Cin, and one series inductor, Lin. The Smith chart is used to find the component values. To do this, the constant conductance circle that crosses the center of the Smith chart and the constant resistance circle that crosses `GammaS` are plotted and the intersection points (Point Γ_A) is found:

```
[~, hsm] = circle(unmatched_amp,fc,'G',1,'R',real(Zs));
hsm.Type = 'YZ';
```

```
% Choose GammaA and show points of interest on smith chart:
hold on
plot(GammaS,'k.','MarkerSize',16)
text(real(GammaS)+0.05,imag(GammaS)-0.05,'\Gamma_{S}','FontSize', 12, ...
    'FontUnits','normalized')
plot(0,0,'k.','MarkerSize',16)
GammaA = 0.384*exp(1j*(-112.6)*pi/180);
plot(GammaA,'k.','MarkerSize',16)
text(real(GammaA)+0.05,imag(GammaA)-0.05,'\Gamma_{A}','FontSize', 12, ...
    'FontUnits','normalized')
hLegend = legend('Location','SouthEast');
hLegend.String = hLegend.String(1:end-3);
hold off
```





Conductance

Gamma = |0.3830|, -112.5 [deg]:
Z = 0.592 - j0.491:
Y = 1.000 + j0.829:

Using the chosen GammaA, the input matching network components, Cin and Lin, are obtained:

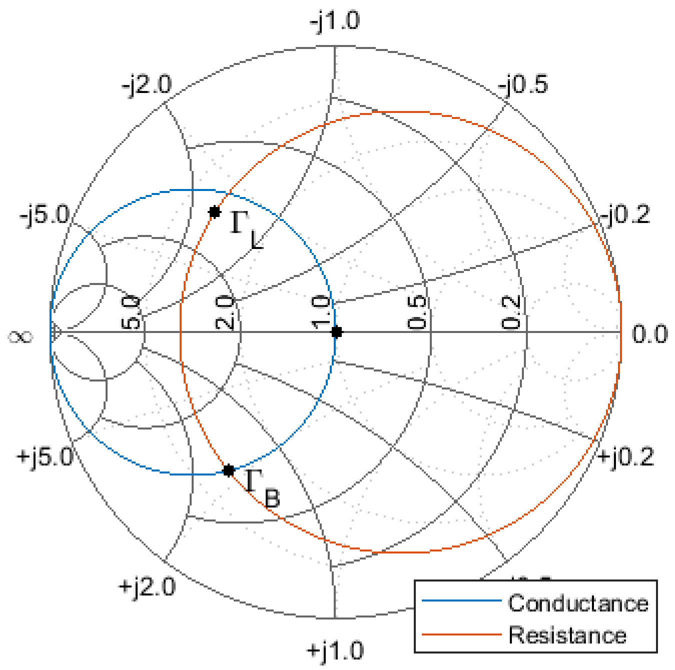
```
% Obtain admittance Ya corresponding to GammaA:
Za = gamma2z(GammaA,1);
Ya = 1/Za;

% Using Ya, find Cin and Lin:
```

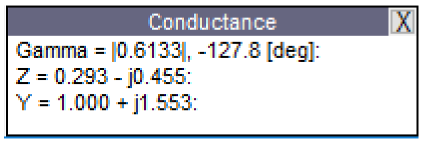
```
Cin = imag(Ya)/50/2/pi/fc
Lin = (imag(Zs) - imag(Za))*50/2/pi/fc
```

```
Cin =

   4.8145e-13


Lin =

   1.5218e-09
```

In a similar manner, the output matching network components are obtained using the intersection points (Point Γ_B) between a constant conductance circle that crosses the center of the Smith chart and the constant resistance circle that crosses `GammaL`:

```
[hLine, hsm] = circle(unmatched_amp,fc,'G',1,'R',real(Zl));
hsm.Type = 'YZ';

% Choose GammaB and show points of interest on smith chart:
hold on
plot(GammaL,'k.','MarkerSize',16)
text(real(GammaL)+0.05,imag(GammaL)-0.05,'\Gamma_{L}','FontSize', 12, ...
    'FontUnits','normalized')
plot(0,0,'k.','MarkerSize',16)
GammaB = 0.612*exp(1j*(-127.8)*pi/180);
plot(GammaB,'k.','MarkerSize',16)
text(real(GammaB)+0.05,imag(GammaB)-0.05,'\Gamma_{B}','FontSize', 12, ...
    'FontUnits','normalized')
hLegend = legend('Location','SouthEast');
hLegend.String = hLegend.String(1:end-3);
hold off
```





Conductance

Gamma = |0.6133|, -127.8 [deg]:
Z = 0.293 - j0.455:
Y = 1.000 + j1.553:

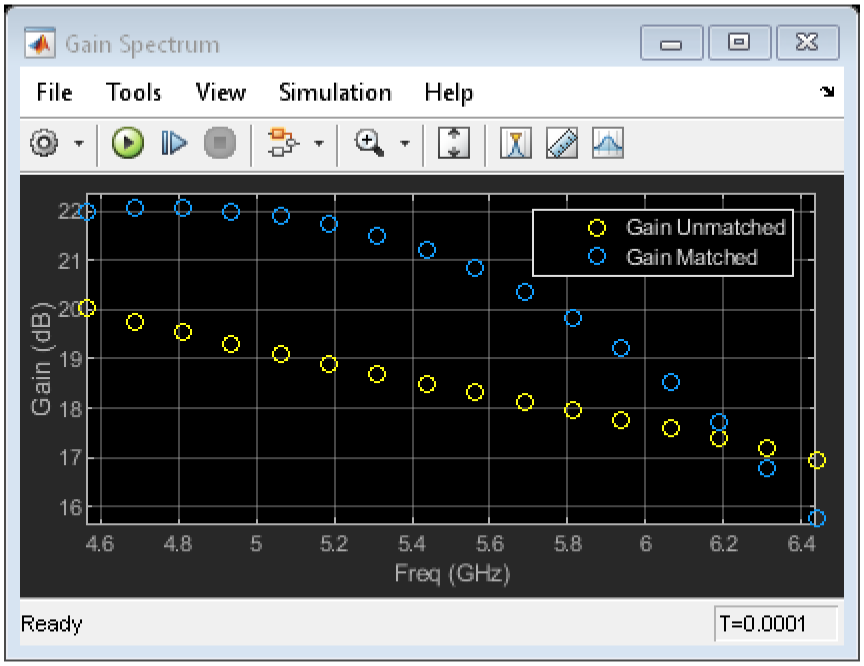
Using the chosen GammaB, the input matching network components, Cout and Lout, are obtained:

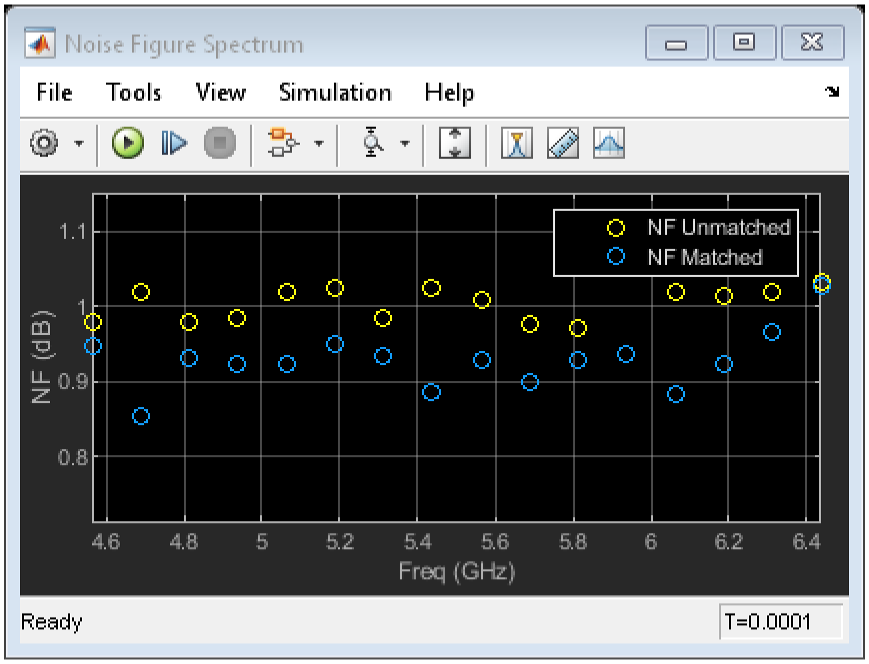
```
% Obtain admittance Yb corresponding to GammaB:
Zb = gamma2z(GammaB, 1);
Yb = 1/Zb;

% Using Yb, find Cout and Lout:
Cout = imag(Yb)/50/2/pi/fc


Cout =

   8.9651e-13


Lout = (imag(Zl) - imag(Zb))*50/2/pi/fc


Lout =

   1.2131e-09
```

Simulation results for gain and noise figure spectrum measurement model

The above input and output network component values are used in the simulation of the matched DUT in the gain and noise figure spectrum measurement model described earlier. The spectral results displayed in the Array Plot blocks are given below:

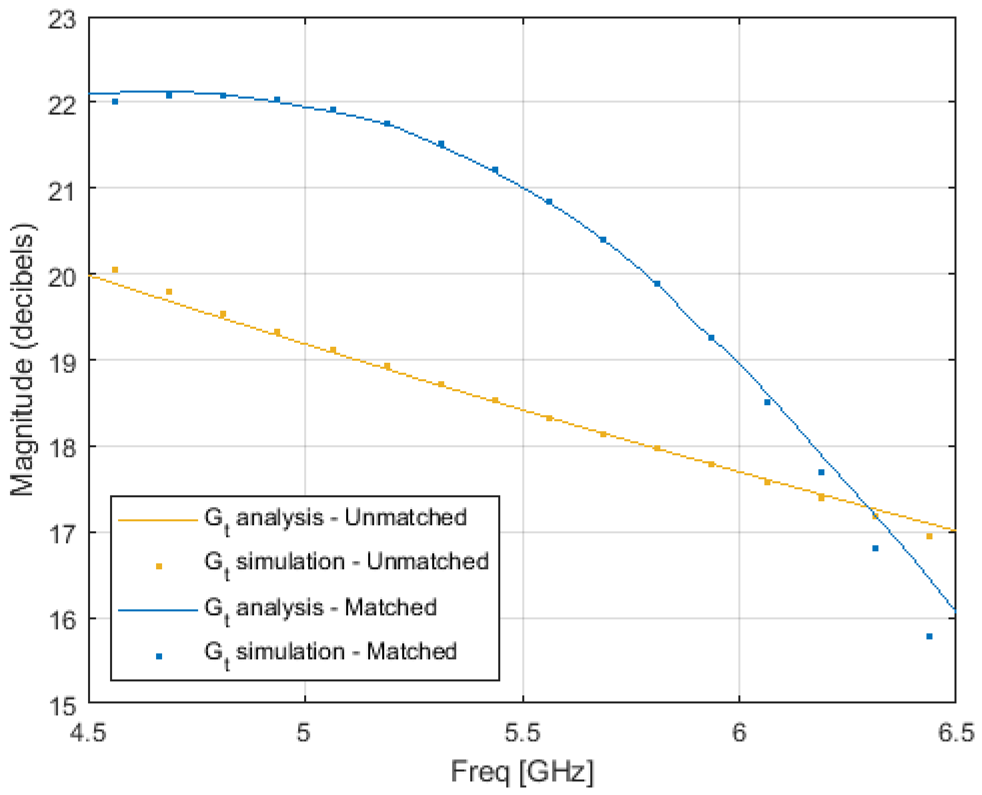
```
open_system([model '/Gain Spectrum']);
open_system([model '/Noise Figure Spectrum']);
sim(model, 1e-4);
```

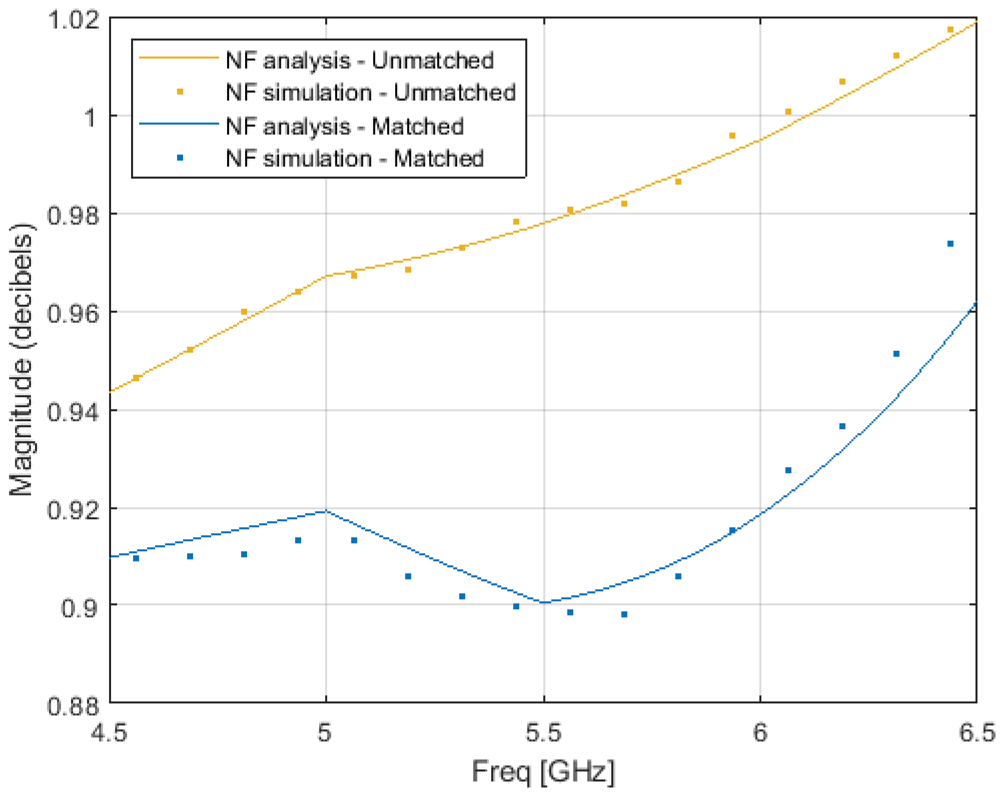




Next, the simulation results are compared with those expected analytically. To facilitate the comparison, the unmatched and matched amplifier networks are analyzed using RF Toolbox. In addition, as finer details are required, the simulation is run for a longer time. The results of the longer simulation are given in the file 'GainNoiseResults.mat'.

```
% Analyze unmatched amplifier
BW_analysis = 2e9; % Bandwidth of the analysis (Hz)
f_analysis = (-BW_analysis/2:1e6:BW_analysis/2)+fc;
analyze(unmatched_amp, f_analysis);

% Create and analyze an RF network for the matched amplifier
input_match = rfckt.cascade('Ckts', ...
    {rfckt.shuntrlc('C',Cin),rfckt.seriesrlc('L',Lin)});
output_match = rfckt.cascade('Ckts', ...
    {rfckt.seriesrlc('L',Lout),rfckt.shuntrlc('C',Cout)});
matched_amp = rfckt.cascade('ckts', ...
    {input_match,unmatched_amp,output_match});
analyze(matched_amp,f_analysis);

% Load results of a longer simulation
load 'GainNoiseResults.mat' f GainSpectrum NFSpectrum;

% Plot expected and simulated Transducer Gain
StdBlue = [0 0.45 0.74];
StdYellow = [0.93,0.69,0.13];
hLineUM = plot(unmatched_amp, 'Gt', 'dB');
hLineUM.Color = StdYellow;
hold on
plot(f, GainSpectrum(:,1), '.', 'Color', StdYellow);
hLineM = plot(matched_amp, 'Gt', 'dB');
hLineM.Color = StdBlue;
plot(f, GainSpectrum(:,2), '.', 'Color', StdBlue);
legend({'G_t analysis - Unmatched', ...
        'G_t simulation - Unmatched', ...
        'G_t analysis - Matched', ...
        'G_t simulation - Matched'}, 'Location','SouthWest');

% Plot expected and simulated Noise Figure
hFig = figure;
hLineUM = plot(unmatched_amp, 'NF', 'dB');
hLineUM.Color = StdYellow;
legend('Location','NorthWest')
hold on
plot(f, NFSpectrum(:,1), '.', 'Color', StdYellow);
hLineM = plot(matched_amp, 'NF', 'dB');
hLineM.Color = StdBlue;
plot(f, NFSpectrum(:,2), '.', 'Color', StdBlue);
legend({'NF analysis - Unmatched', ...
        'NF simulation - Unmatched', ...
        'NF analysis - Matched', ...
        'NF simulation - Matched'}, 'Location','NorthWest');
```

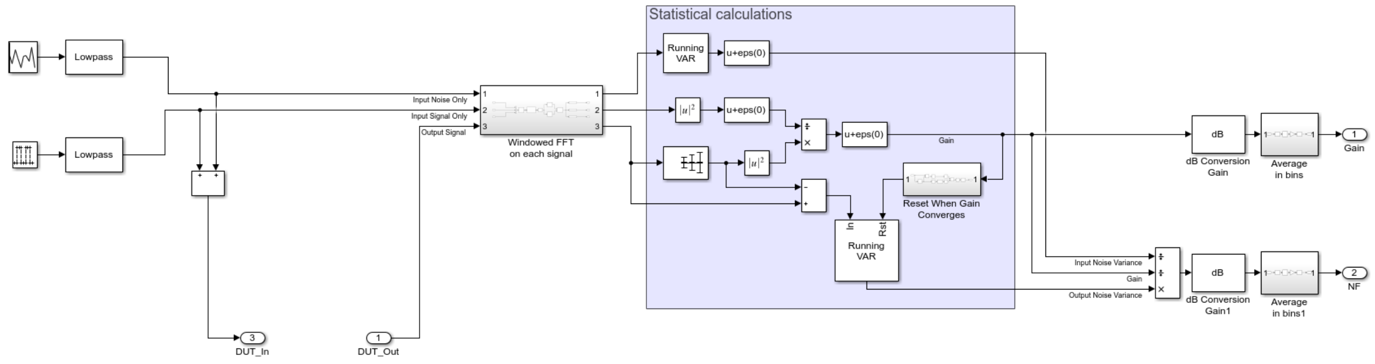

Magnitude (decibels)
Freq [GHz]
G_t analysis - Unmatched
G_t simulation - Unmatched
G_t analysis - Matched
G_t simulation - Matched
23
22
21
20
19
18
17
16
15
4.5
5
5.5
6
6.5



Operation of the measurement unit

The measurement unit produces an input signal, DUT_in, that is composed of zero-mean white noise and zero-variance impulse response signal. The latter is used to determine the frequency response of the DUT gain and together with the white noise determine the DUT noise figure. The measurement unit collects the DUT output signal, performs a windowed FFT on it and then facilitates statistical calculations to obtain the gain and the noise figure of the DUT.

```
open_system([model '/Noise and Gain Measurement'], 'force');
```

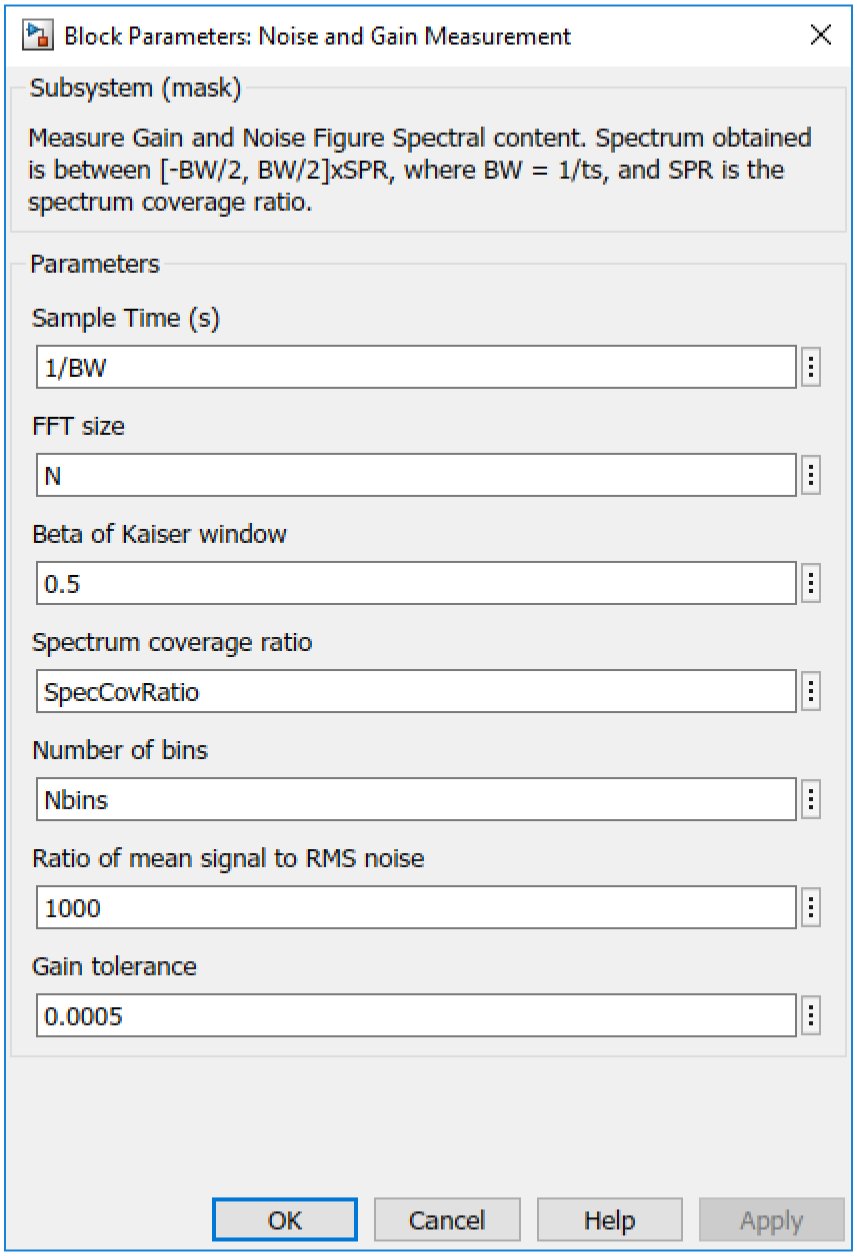


The statistical calculations are done in the area marked in blue. The calculations use three inputs in the frequency domain; Input Noise Only, Input Signal Only, and Output Signal. The Input Signal Only is compared with the mean of the Output Signal to determine the DUT's gain, G, at each frequency bin. The variance of the Output Signal, with mean signal removed, yields the output noise of the DUT system, N_o, Together with the input noise fed to the DUT, N_i, calculated by taking the variance of the Input Noise Only, the Noise Figure, NF, can be calculated using the following formula:

$$NF = \frac{SNR_{in}}{SNR_{out}} = \frac{N_o}{N_i G}$$

Where, SNR_{in} and SNR_{out} in the above equation are the Signal-to-Noise ratios at the input and output of the DUT. Finally, after conversion to decibels, the spectral results are divided into bins and averaged within them to facilitate faster convergence. Also, to improve the noise calculation convergence, the output noise variance is reset once the gain has reached convergence.

The properties affecting the operation of the measurement unit are specified in the block's mask parameter dialog box as shown below:



Subsystem (mask)

Measure Gain and Noise Figure Spectral content. Spectrum obtained is between [-BW/2, BW/2]xSPR, where BW = 1/ts, and SPR is the spectrum coverage ratio.

Parameters

Sample Time (s): 1/BW

FFT size: N

Beta of Kaiser window: 0.5

Spectrum coverage ratio: SpecCovRatio

Number of bins: Nbins

Ratio of mean signal to RMS noise: 1000

Gain tolerance: 0.0005

These parameters are described below:

- Sample time - Sample time of the signal created by the measurement unit. The sample time also governs the total simulation bandwidth captured by the measurement unit.
- FFT size - Number of FFT bins used to obtain the frequency domain representation of the signals within the measurement unit.
- Beta of Kaiser window - The β parameter of the Kaiser window used in all FFT calculations within the measurement unit. Increasing β widens the mainlobe and decreases the amplitude of the sidelobes of the frequency response of the window.
- Spectrum coverage ratio - Value between 0 and 1, representing the part of total simulation bandwidth processed by the measurement unit.

- Number of bins - Number of output frequency bins in the Gain and NF signals created by the measurement unit. The FFT bins within the covered spectrum are re-distributed into those output bins. Multiple FFT bins falling into the same output bin are averaged.
- Ratio of mean signal to RMS noise - The ratio of the mean signal amplitude to the RMS noise in the DUT_in signal created by the measurement unit. A large value improves the convergence of the DUT gain calculation, but reduces the accuracy of noise calculation due to numerical inaccuracies.
- Gain tolerance - The threshold of gain variation relative to its average. When the threshold is hit, the gain is considered as converged, triggering a reset for the output noise calculation.

```
close(hFig);
bdclose(model);
clear model hLegend hsm hLine hLegend StdBlue StdYellow hLineUM hLineM hFig;
clear GammaS Zs GammaL Zl GammaA Za Ya GammaB Zb Yb;
clear unmatched_amp BW_analysis f_analysis input_match output_match matched_amp;
```

See Also

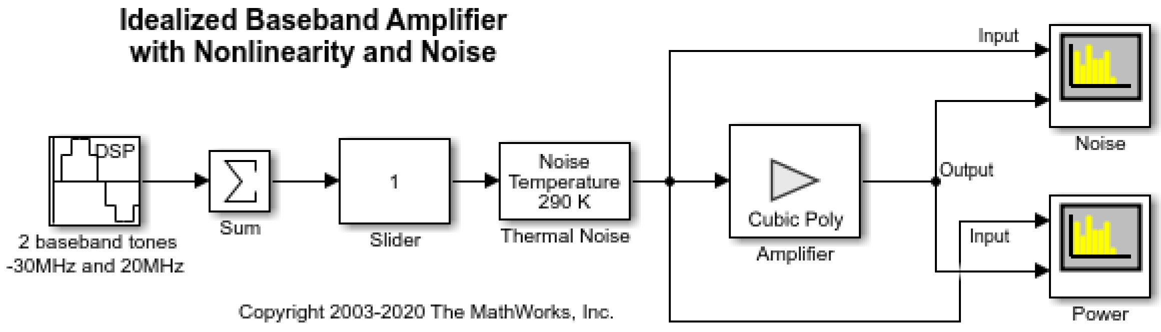
Amplifier | Configuration | Inport | Outport

Related Topics

“RF Noise Modeling” on page 9-199 | “Model LO Phase Noise” on page 2-8

Idealized Baseband Amplifier with Nonlinearity and Noise

The example shows how to use the idealized baseband library Amplifier block to amplify a signal with nonlinearity and noise. The Amplifier uses the `Cubic Polynomial` model with a `Linear power gain` of 10 dB, an `Input IP3` nonlinearity of 30 dBm, and a `Noise figure` of 3 dB.



System Architecture

The DSP Sine Wave block inputs two complex baseband tones with a power level of -20 dBm and -25 dBm at frequencies of -30 MHz and 20 MHz. In this block you can also:

- Increase the samples per frame to increase the simulation speed.
- Use output complexity and phase offset to control the I-Q relationship of each baseband signal
- Control the bandwidth of the scopes using the inverse of the sample time parameter.

The Amplifier block only accepts a vector input. The Sum block combines the two baseband signals into a vector length equal to the samples per frame in the DSP Sine Wave block.

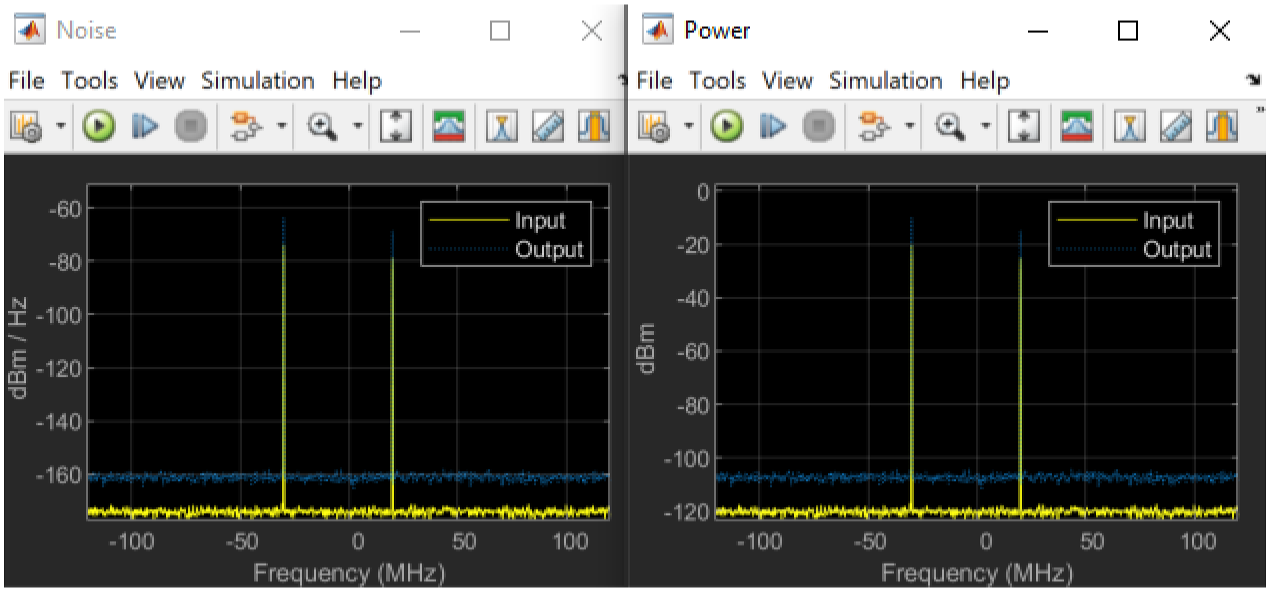
The Thermal Noise block creates a thermal noise floor input of -174 dBm/Hz.

Simulation Analysis

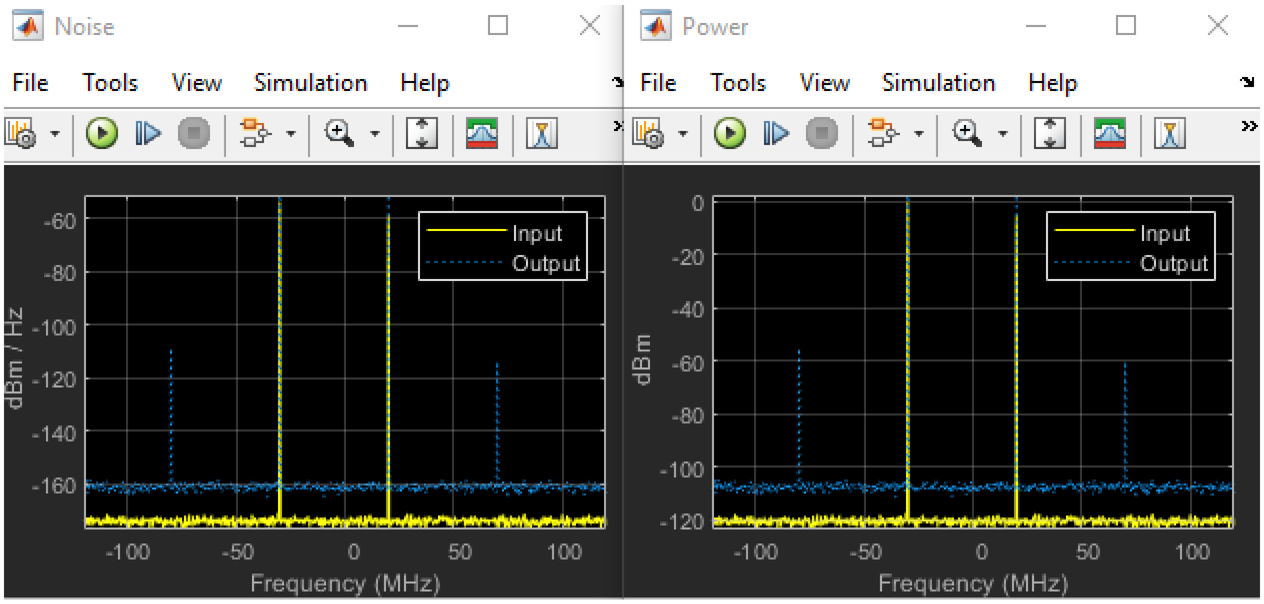
The Amplifier block with `Linear power gain` of 10 dB outputs tone with magnitude -10 dBm and -15 dBm as seen in the Power plot. The Amplifier also increases the thermal noise floor to -161 dBm/Hz. You can calculate the output thermal noise using this equation:

$$InputNoiseFloor + linearpowergain + NoiseFigure = -174dBm/Hz + 10dBm + 3dBm = -161dBm/Hz$$

The following plots illustrate the differences in the input and output noise floors. The spurs appear at 70 MHz (2*20 MHz + 30 MHz) and -80 MHz (2*(-30 MHz) - 20 MHz). This shows the third order intercept nature of the spurs.



Increasing the Slider value from 1 to 10, shows nonlinear effects in the plots. These are the Noise and Power plots when the gain of the Slider is 10.

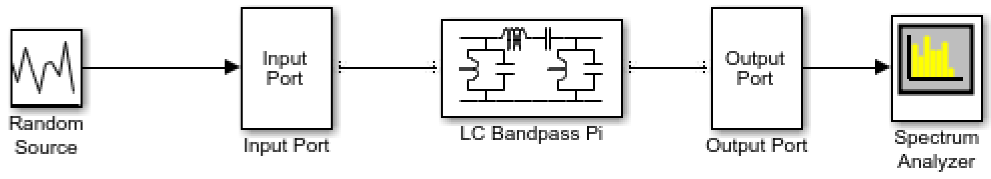


See Also

Amplifier

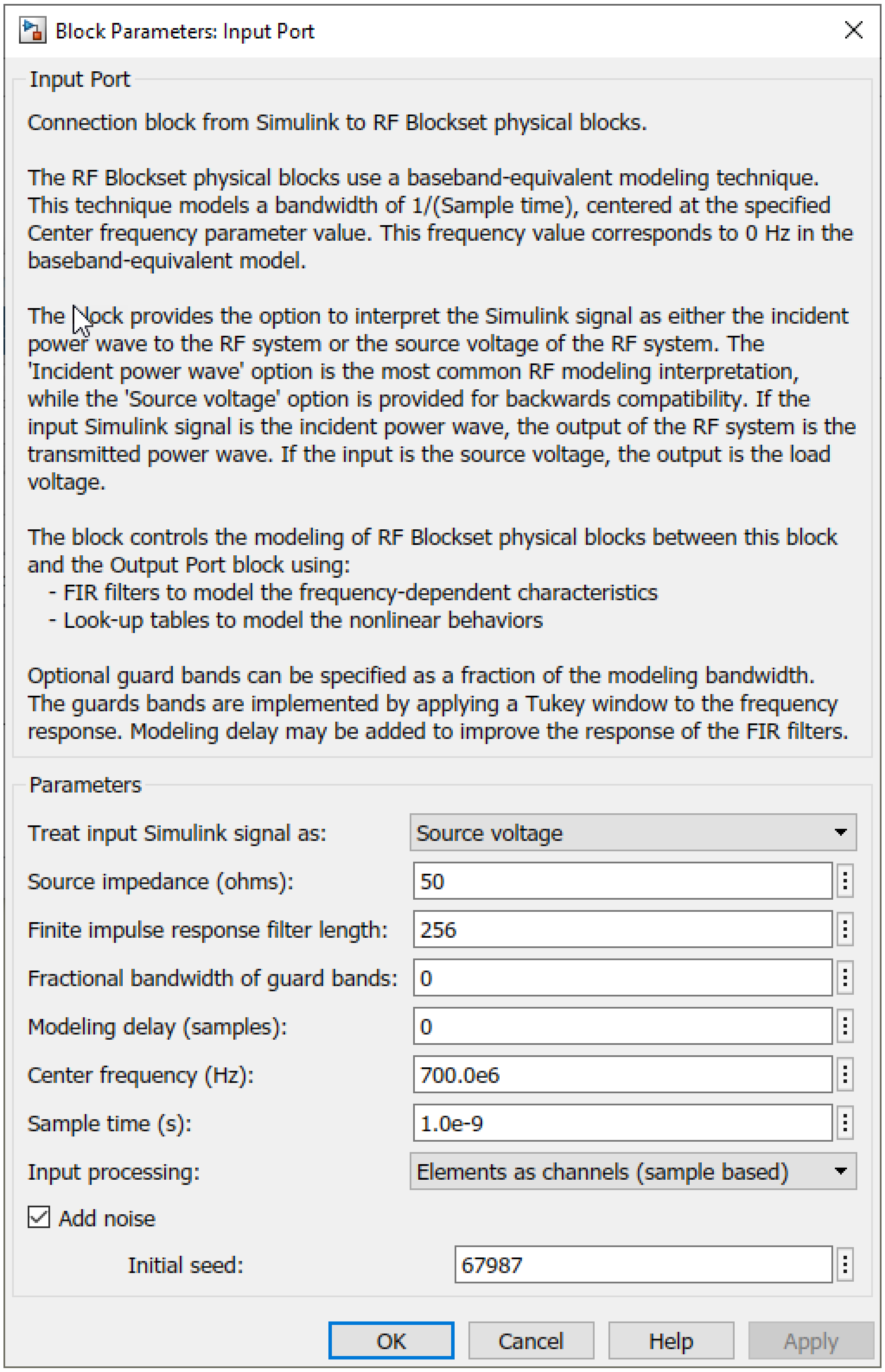
Using a Ladder Filter Block to Filter Gaussian Noise

This example provides complex random noise in Gaussian form as input to an LC Bandpass Pi block. A DSP System Toolbox fallback for Spectrum Scope block plots the filtered output.



The DSP System Toolbox fallback for 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.


Block Parameters: Input Port

Input Port

Connection block from Simulink to RF Blockset physical blocks.

The RF Blockset physical blocks use a baseband-equivalent modeling technique. This technique models a bandwidth of 1/(Sample time), centered at the specified Center frequency parameter value. This frequency value corresponds to 0 Hz in the baseband-equivalent model.

The block provides the option to interpret the Simulink signal as either the incident power wave to the RF system or the source voltage of the RF system. The 'Incident power wave' option is the most common RF modeling interpretation, while the 'Source voltage' option is provided for backwards compatibility. If the input Simulink signal is the incident power wave, the output of the RF system is the transmitted power wave. If the input is the source voltage, the output is the load voltage.

The block controls the modeling of RF Blockset physical blocks between this block and the Output Port block using:
- FIR filters to model the frequency-dependent characteristics
- Look-up tables to model the nonlinear behaviors

Optional guard bands can be specified as a fraction of the modeling bandwidth. The guards bands are implemented by applying a Tukey window to the frequency response. Modeling delay may be added to improve the response of the FIR filters.

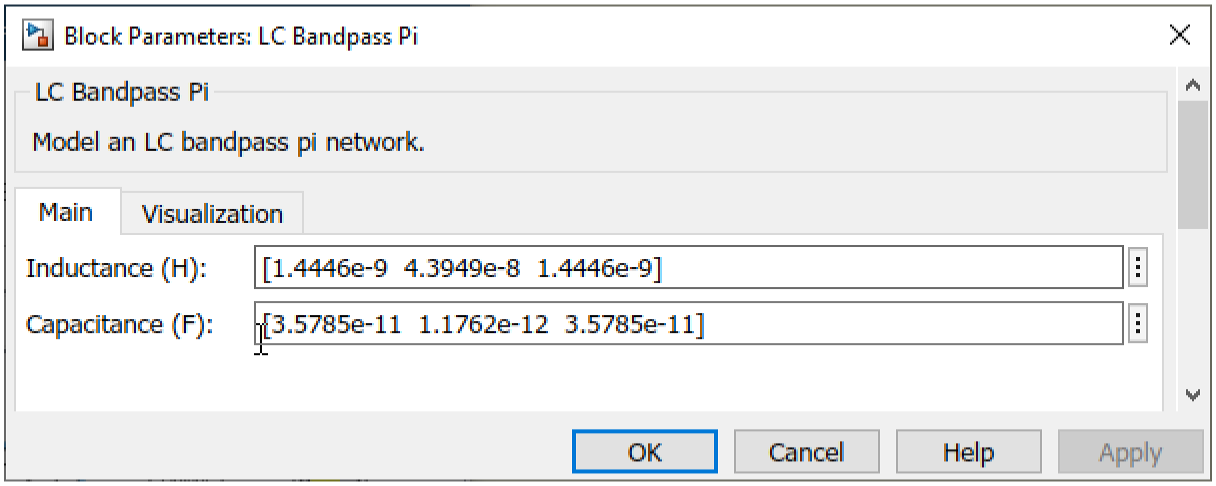
Parameters

Treat input Simulink signal as:	Source voltage
Source impedance (ohms):	50
Finite impulse response filter length:	256
Fractional bandwidth of guard bands:	0
Modeling delay (samples):	0
Center frequency (Hz):	700.0e6
Sample time (s):	1.0e-9
Input processing:	Elements as channels (sample based)

✓ Add noise

Initial seed:	67987

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].


Block Parameters: LC Bandpass Pi

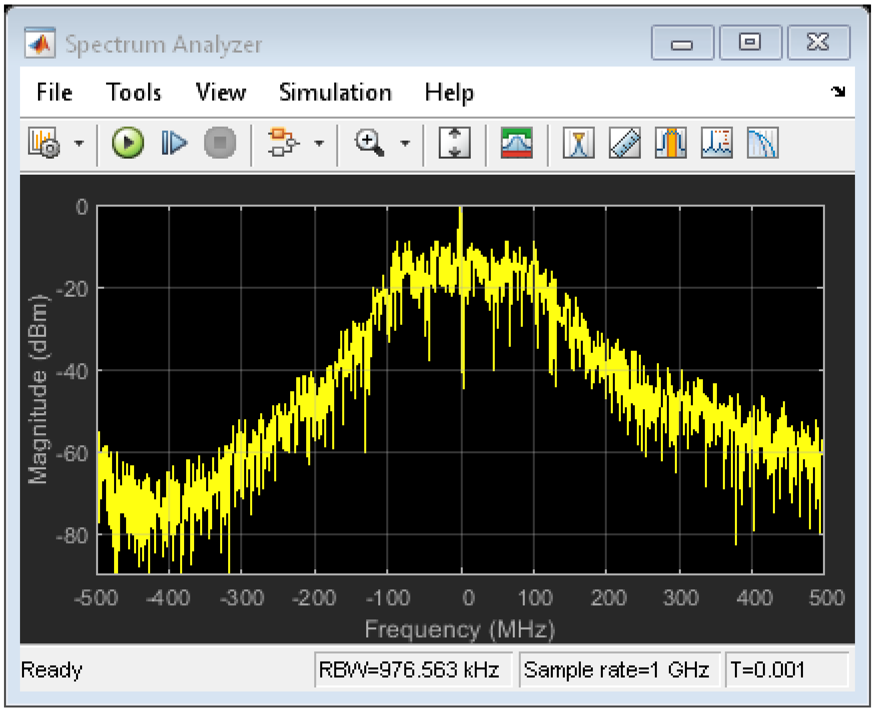
LC Bandpass Pi

Model an LC bandpass pi network.

Inductance (H): [1.4446e-9 4.3949e-8 1.4446e-9]

Capacitance (F): [3.5785e-11 1.1762e-12 3.5785e-11]

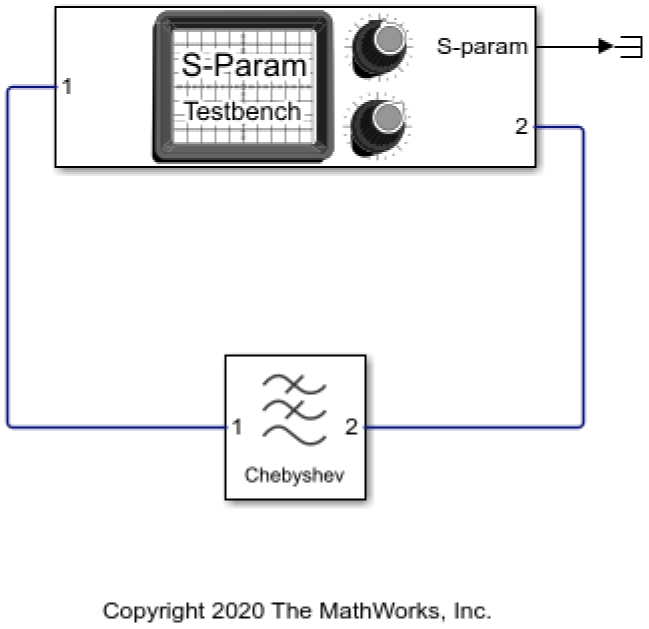
The following plot shows a sample of the baseband-equivalent RF signal generated by this LC Bandpass Pi block. Zero (0) on the frequency axis corresponds to the center frequency specified in the Input Port block. The bandwidth of the frequency spectrum is 1/sample time. You specify the Sample time parameter in the Input Port block.



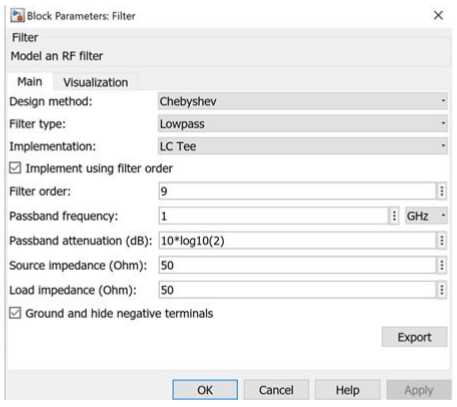
The Axis Properties of the Spectrum Scope block have been adjusted to show the frequencies above and below the carrier. The Minimum Y-limit parameter is -90, and Maximum Y-limit is 0.

Measure S-Parameter Data of Chebyshev Filter

Use the S-Parameter Testbench block to measure S-parameter data of a ninth-order Chebyshev Tee LC low-pass filter.



Set the parameters of the Filter block as shown in this image.



Block Parameters: Filter

Filter

Model an RF filter

Main | Visualization

Design method:	Chebyshev
Filter type:	Lowpass
Implementation:	LC Tee

☑ Implement using filter order

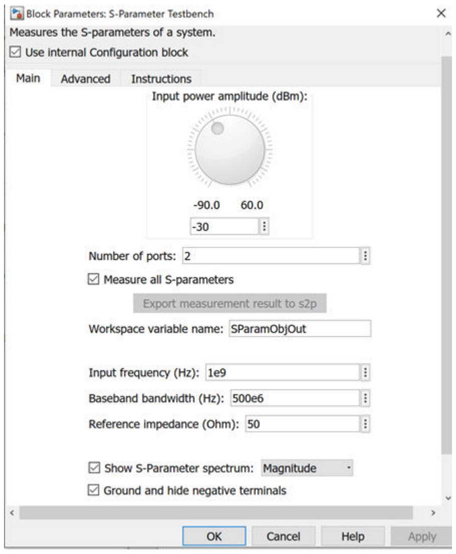
Filter order:	9
Passband frequency:	1 GHz
Passband attenuation (dB):	10*log10(2)
Source impedance (Ohm):	50
Load impedance (Ohm):	50

☑ Ground and hide negative terminals

Export

OK Cancel Help Apply

Set the parameters of the S-Parameter Testbench block to measure the S-parameters of the Chebyshev filter.


Block Parameters: S-Parameter Testbench

Measures the S-parameters of a system.

☑ Use internal Configuration block

Main | Advanced | Instructions

Input power amplitude (dBm): -30 (-90.0 to 60.0)

Number of ports: 2

☑ Measure all S-parameters

Export measurement result to s2p

Workspace variable name: SParamObjOut

Input frequency (Hz): 1e9

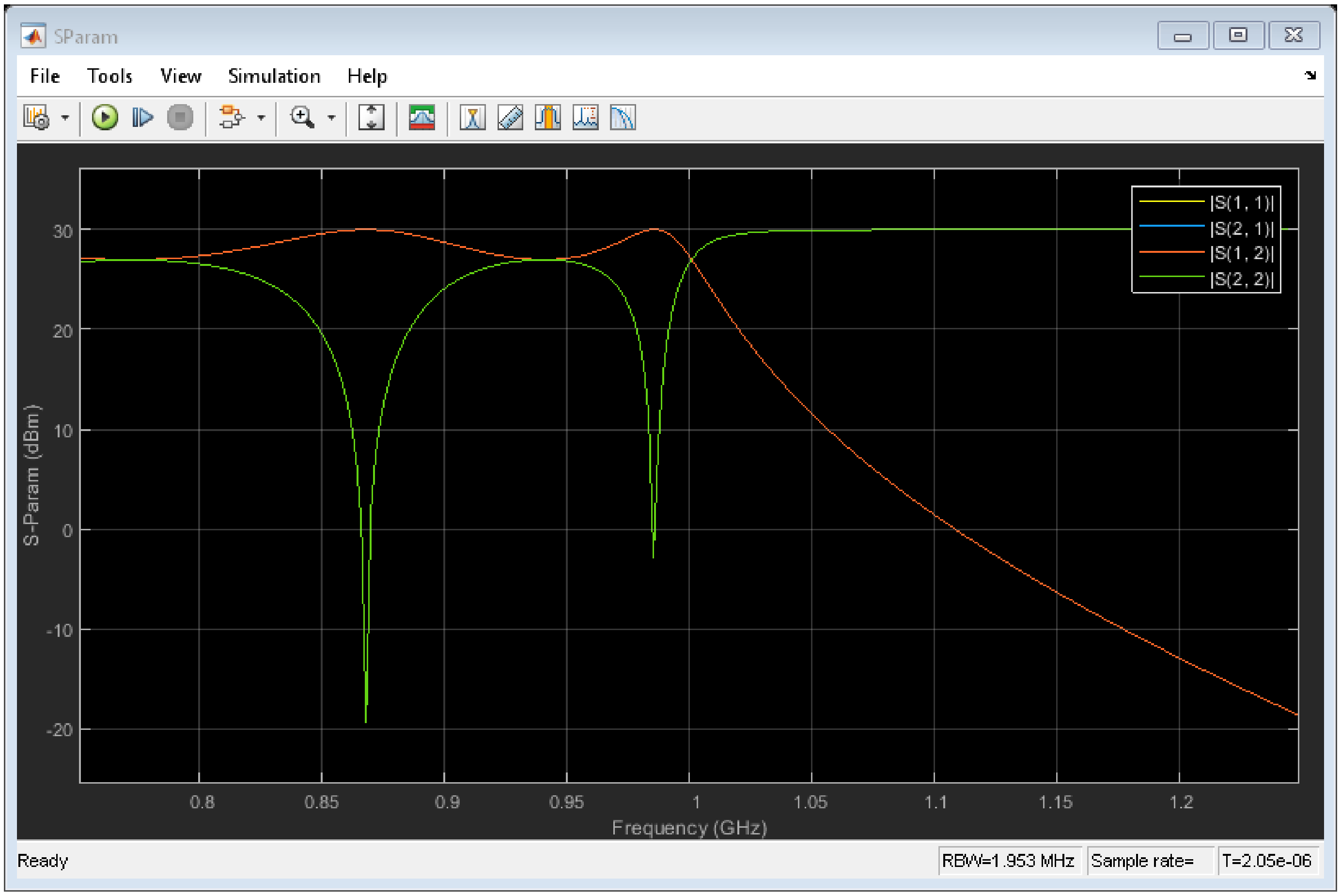
Baseband bandwidth (Hz): 500e6

Reference impedance (Ohm): 50

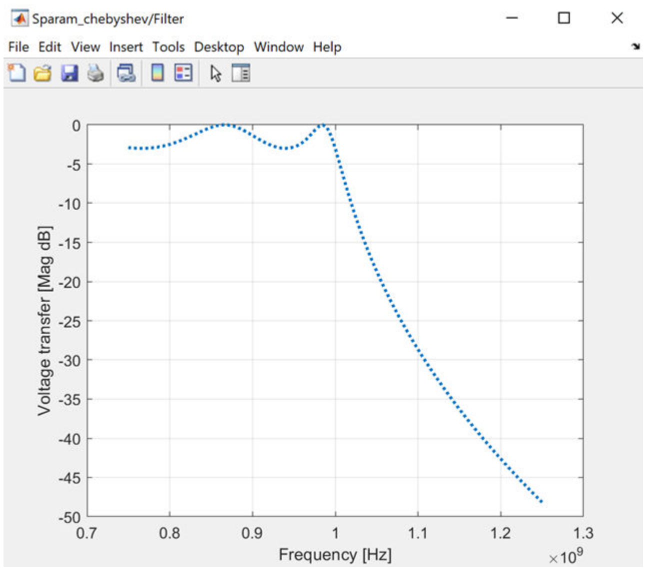
☑ Show S-Parameter spectrum: Magnitude

☑ Ground and hide negative terminals

Run the simulation to yield the measured S-parameter data for the Chebyshev filter.



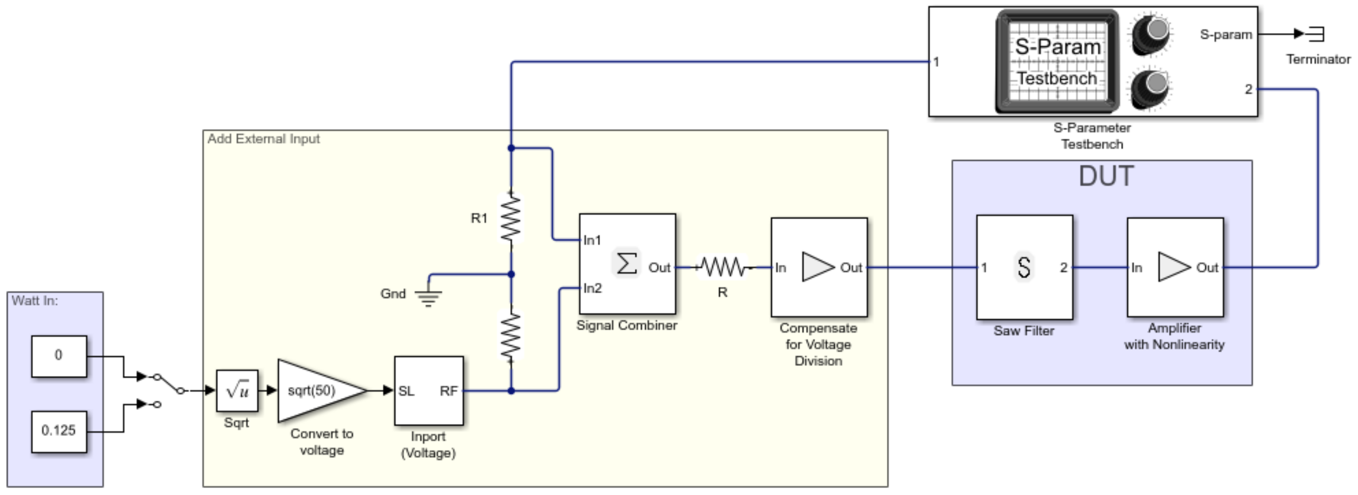
Note that due to the symmetry and reciprocity of the filter, the plots for S11 and S22 and S12 and S21 overlap each other. The results can be compared with the plot obtained from the **Visualization** tab of the Filter block.



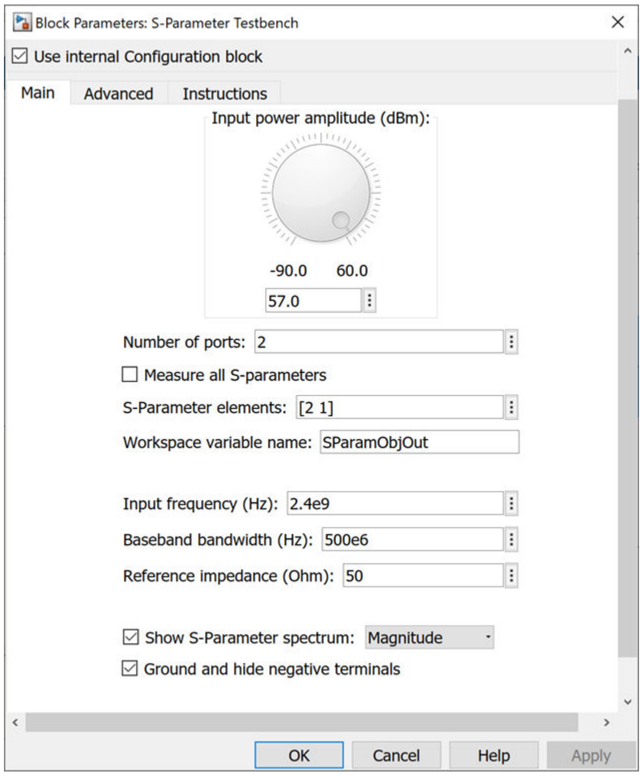
In addition to viewing the results in the Spectrum Analyzer, you can export the result as an RF Toolbox™ S-parameter object or as a Touchstone® file for further processing.

Measure S-Parameter of Nonlinear System

Use the S-Parameter Testbench block to measure the S-parameters of a nonlinear system. The system shown here includes a device under test (DUT) with a linear filter followed by a nonlinear amplifier. The system allows an external steady state input that is switched off initially. When the DUT is nonlinear, the stimulus signal created in the S-Parameter Testbench block must be small enough to operate in the linear region.



Set the S-Parameter Testbench block parameters as shown.


Block Parameters: S-Parameter Testbench

☑ Use internal Configuration block

Input power amplitude (dBm): 57.0

Number of ports: 2

☐ Measure all S-parameters

S-Parameter elements: [2 1]

Workspace variable name: SParamObjOut

Input frequency (Hz): 2.4e9

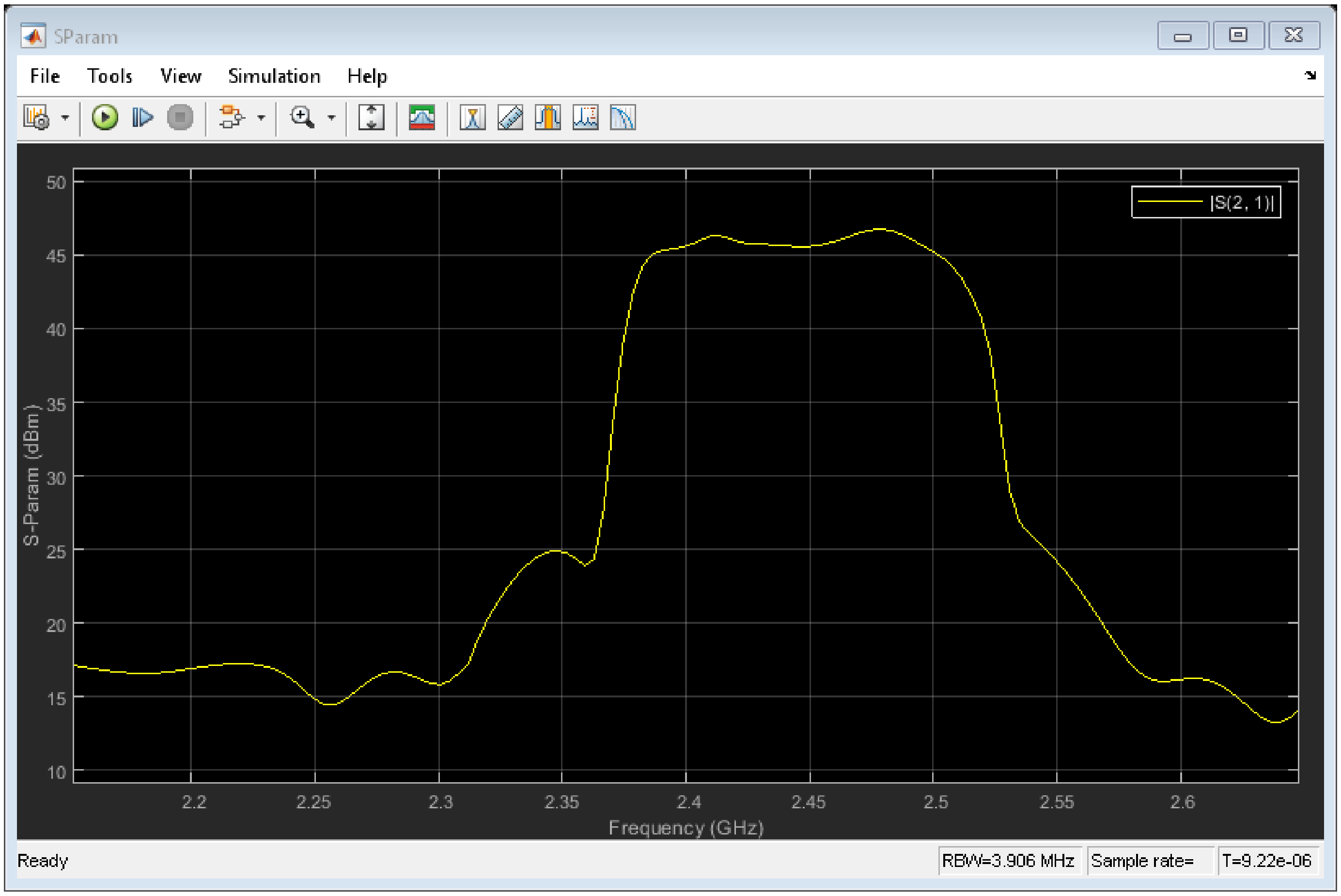
Baseband bandwidth (Hz): 500e6

Reference impedance (Ohm): 50

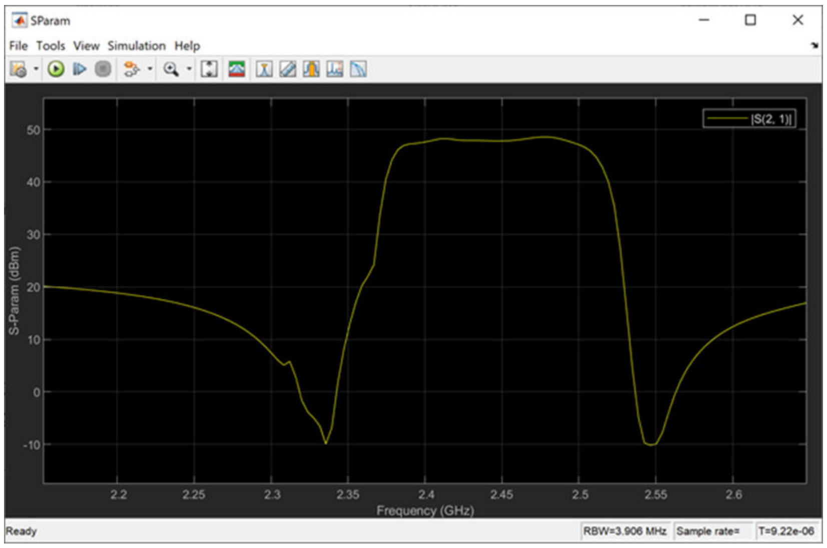
☑ Show S-Parameter spectrum: Magnitude

☑ Ground and hide negative terminals

Select **Run** to display the output of the S-Parameter Testbench block which is the S21 magnitude.

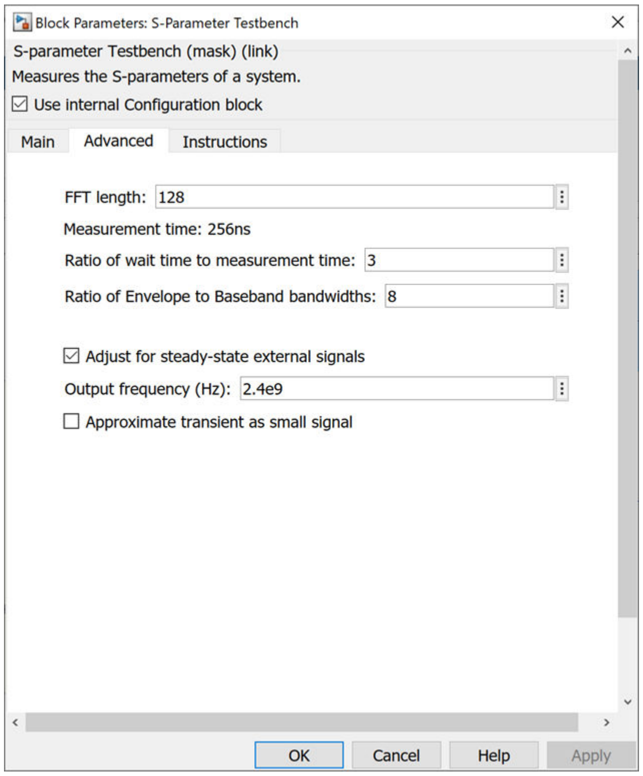


Open the mask parameters dialog box of the S-Parameter Testbench block and set the **Input power amplitude (dBm)** parameter to 15 dBm. You can also vary the **Input power amplitude (dBm)** parameter while the simulation is running. Rerun the simulation and observe the change in the magnitude of the S21 of the system.



Note the difference between the results. When the input power is significant enough to excite the nonlinearity of the system, the dips in the spectrum of the filter fill up due to spectral regrowth. The simulation results are correct in both cases, but the initial S-parameter measurement is invalid due to nonlinearity.

While it is recommended that the stimulus signal be kept small when measuring S-parameter data, it is also possible to measure the S-parameter data with a small stimulus signal around a large signal operating point. To do this, set the switch to an input of 0.125 watts. In addition, in the **Advanced** tab in the block parameters dialog box of the S-Parameter Testbench block, select the **Adjust for steady-state external signals** parameter. This allows the testbench to first measure the output of the system with an external signal and then subtract the external signal from the output to ensure that only the small signal stimulus is accounted for in the calculation of the S-parameters.


Block Parameters: S-Parameter Testbench

S-parameter Testbench (mask) (link)

Measures the S-parameters of a system.

☑ Use internal Configuration block

Main | Advanced | Instructions

FFT length: 128

Measurement time: 256ns

Ratio of wait time to measurement time: 3

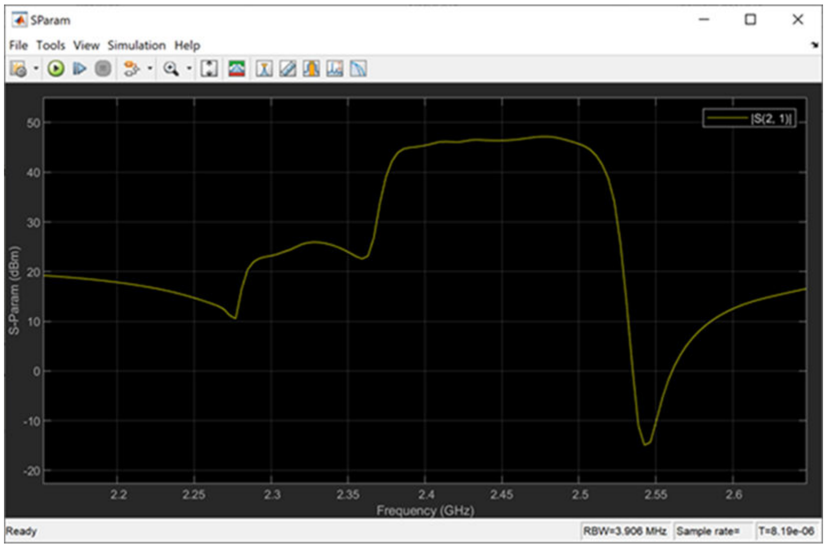
Ratio of Envelope to Baseband bandwidths: 8

☑ Adjust for steady-state external signals

Output frequency (Hz): 2.4e9

☐ Approximate transient as small signal

Rerun the simulation and observe the S21 magnitude.

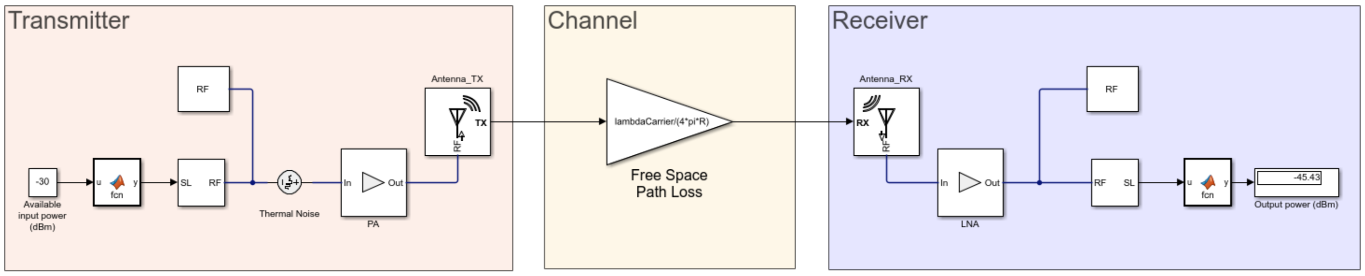


When you compare the result with a zero-watt external signal and a 0.125-watt signal, the effect of a large signal operation point is evident.

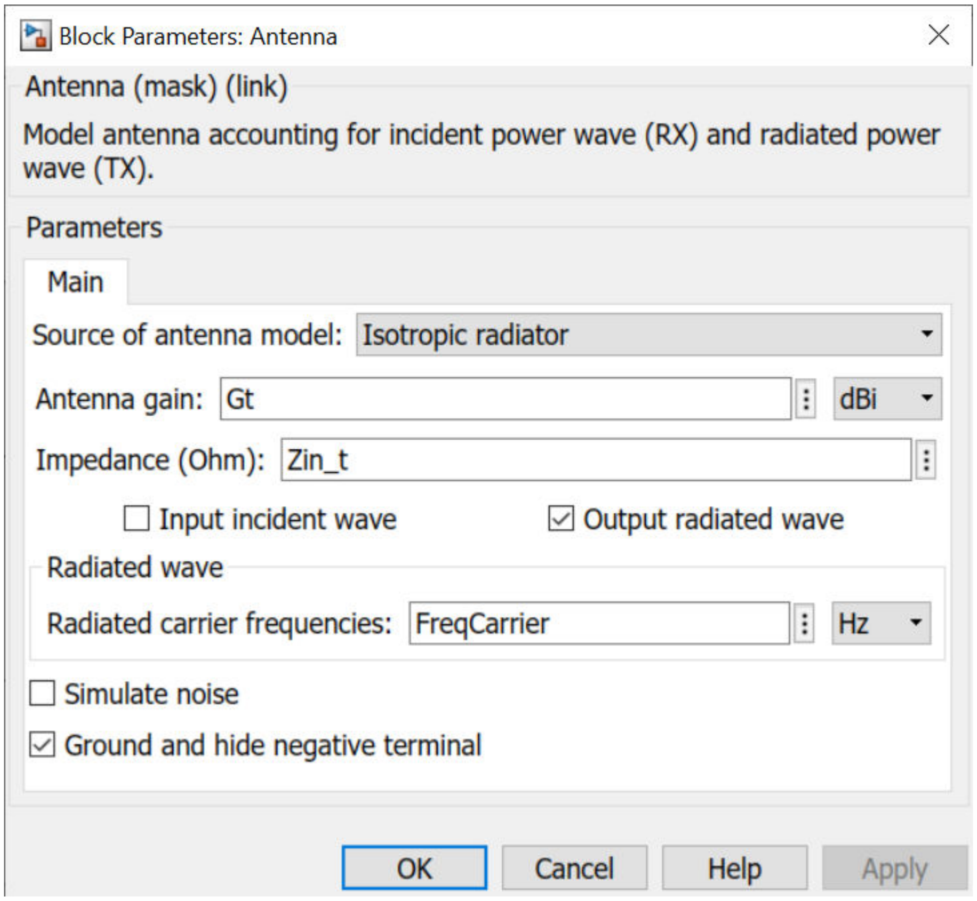
Simulation of RF Systems with Antenna Blocks

Use the antenna block to incorporate the effect of an antenna into an RF simulation. In this model, a single tone is fed to the transmitter and the power of the received signal at the output of the receiver is calculated.

```
model = 'Antenna_block_example';
open(model)
sim(model);
```



Set the **Antenna_TX** and **Antenna_RX** blocks to be isotropic radiators with the following parameters:



Block Parameters: Antenna

Antenna (mask) (link)

Model antenna accounting for incident power wave (RX) and radiated power wave (TX).

Parameters

Main

Source of antenna model: Isotropic radiator

Antenna gain: Gt dBi

Impedance (Ohm): Zin_t

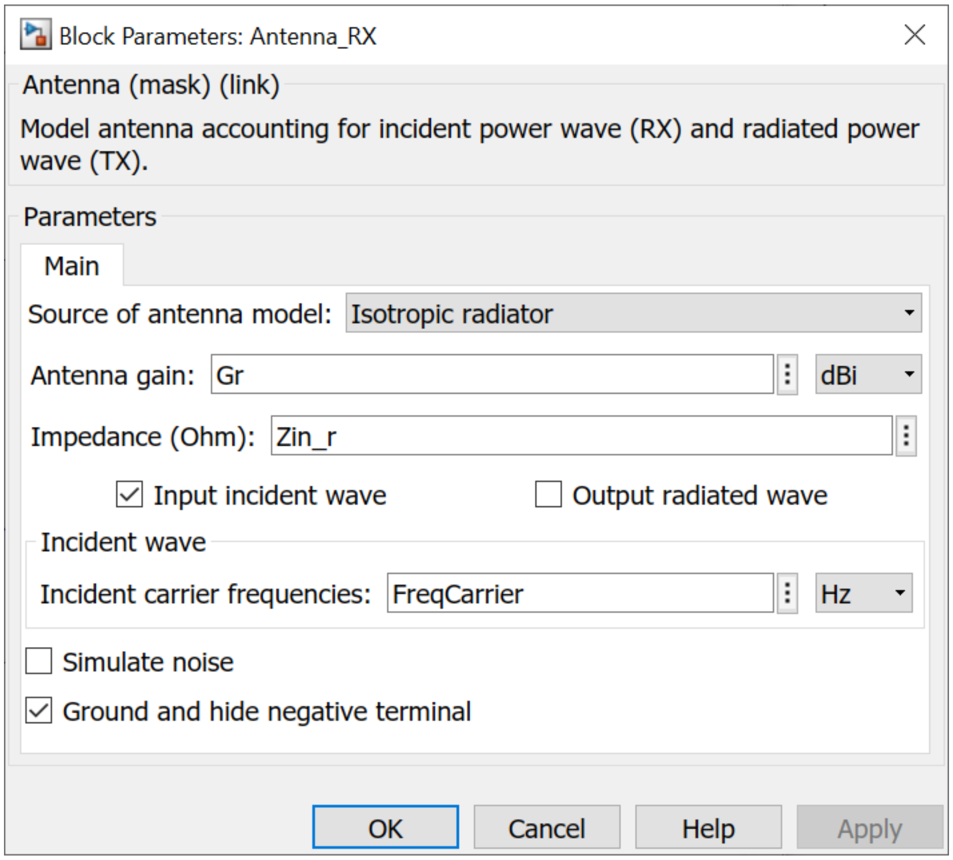
☐ Input incident wave ☑ Output radiated wave

Radiated wave

Radiated carrier frequencies: FreqCarrier Hz

☐ Simulate noise

☑ Ground and hide negative terminal


Block Parameters: Antenna_RX

Antenna (mask) (link)

Model antenna accounting for incident power wave (RX) and radiated power wave (TX).

Parameters

Main

Source of antenna model: Isotropic radiator

Antenna gain: Gr dBi

Impedance (Ohm): Zin_r

☑ Input incident wave ☐ Output radiated wave

Incident wave

Incident carrier frequencies: FreqCarrier Hz

☐ Simulate noise

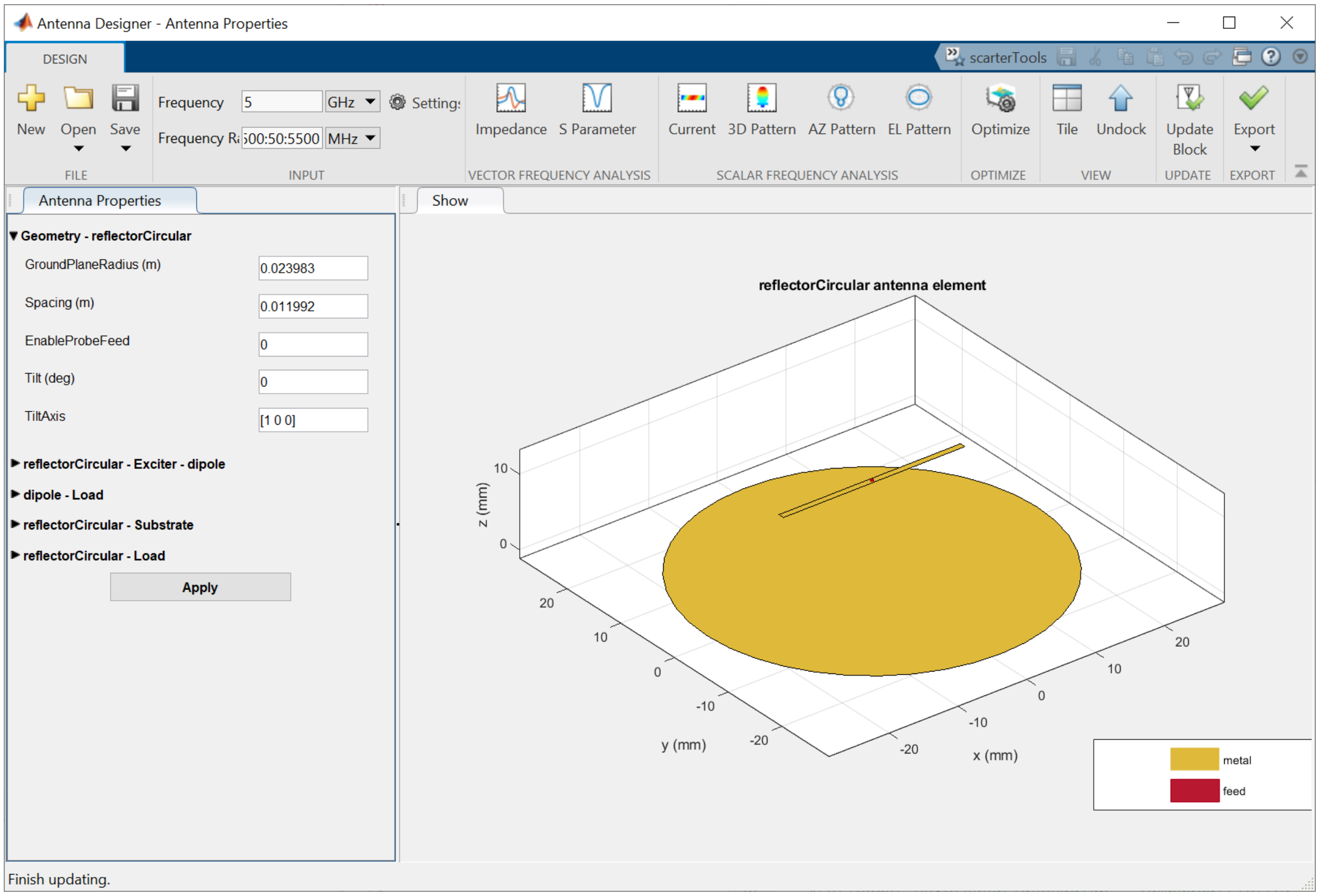
☑ Ground and hide negative terminal

The following values are set upon loading the model:

- R = 100 [m]
- FreqCarrier = 5.0 [GHz]
- Gt = Gr = 7.9988 [dBi]
- Zin_t = Zin_r = 56.2947 - 4.2629i [Ohm]

where the antenna gains and impedances were calculated beforehand from dipoles backed by circular reflectors.

With Antenna Toolbox, it is possible to design the antenna using the Antenna Designer app invoked directly from the block. To do so, change the choice of **Source of the antenna model** to **Antenna Designer** and press the **Create antenna** button. Within the Antenna Designer app, create a new antenna, choose **Dipole** from the Antenna Gallery, **Circular** from the Backing Structure Gallery in the app toolstrip and press **Accept**. Note that the design frequency was prepopulated with the RF system frequency of 5 GHz.



Frequency: 5 GHz

Frequency Range: 500:50:5500 MHz

Geometry - reflectorCircular

GroundPlaneRadius (m)	0.023983
Spacing (m)	0.011992
EnableProbeFeed	0
Tilt (deg)	0
TiltAxis	[1 0 0]

reflectorCircular - Exciter - dipole

dipole - Load

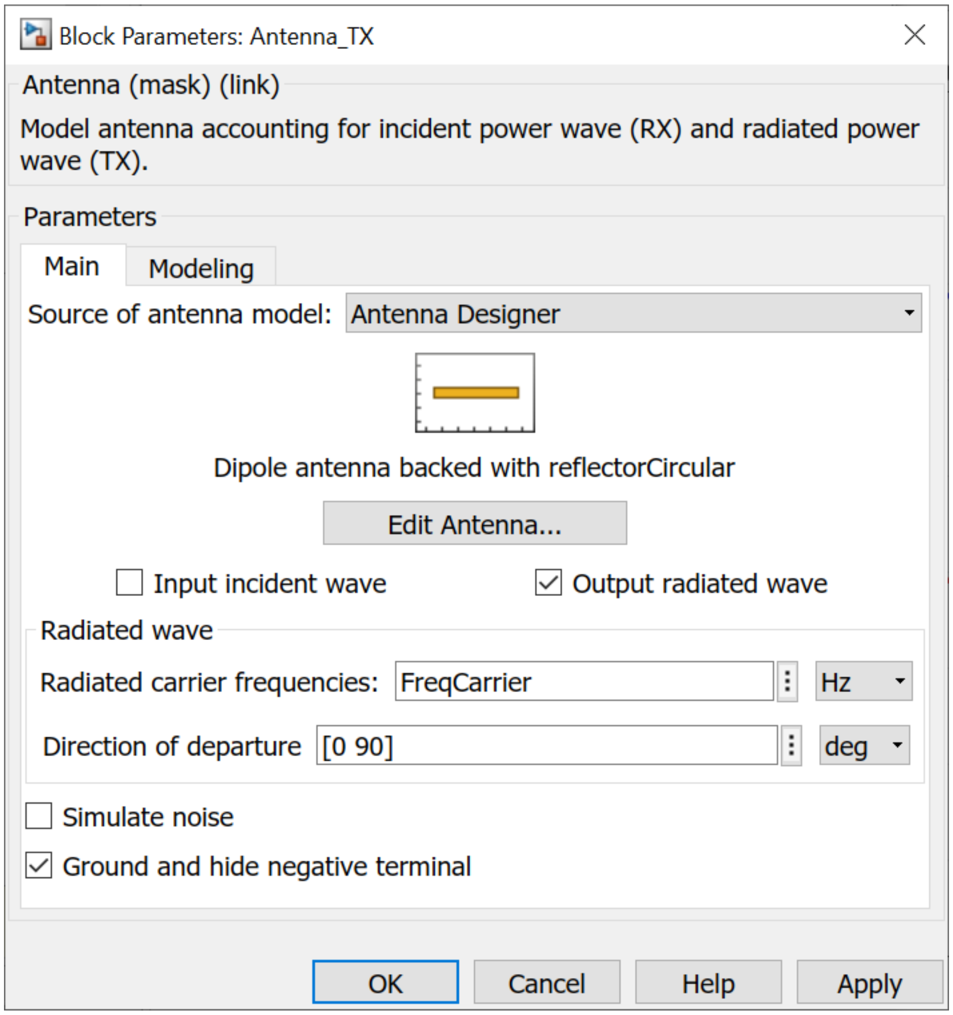
reflectorCircular - Substrate

reflectorCircular - Load

Finish updating.

Press the **Impedance** button in the app toolstrip to analyze the structure and press the **Update Block** button to update the block with the chosen antenna. Note that the Antenna block requires that the designed antenna be analyzed for at least one frequency in the Antenna Designer app before updating and using it in the block.

In the **Antenna_TX** block mask parameter dialog box, change the default **Direction of departure** to 0 degrees in azimuth and 90 degrees in elevation:


Block Parameters: Antenna_TX

Antenna (mask) (link)

Model antenna accounting for incident power wave (RX) and radiated power wave (TX).

Parameters

Source of antenna model: Antenna Designer

Dipole antenna backed with reflectorCircular

☐ Input incident wave ✓ Output radiated wave

Radiated wave

Radiated carrier frequencies: FreqCarrier Hz

Direction of departure [0 90] deg

☐ Simulate noise

✓ Ground and hide negative terminal

Repeat the above steps to design **Antenna_RX**. However, the receiving antenna needs to be rotated to face the transmitting antenna. To do so, in the Antenna Properties panel of the Antenna Designer app, change **Tilt** to 180 degrees. Again, press the **Impedance** button and then press the **Update Block** button to update the block. In the **Antenna_RX** block mask parameter dialog box, change the **Direction of arrival** to 0 degrees in azimuth and -90 degrees in elevation. We choose -90 degrees in elevation since the radiated signal that was transmitted in the positive z direction in the coordinate system of the transmitter, is now arriving from the negative z direction in the coordinate system of the receiver.

Run the model again, and note that the output power remained almost exactly the same. This is since the original gain and impedance values used for the isotropically radiating antenna in the beginning were calculated from the same antennas and spatial settings. However, it is now possible to change the antenna properties and observe the effect on the output power in the model. For example: press the 'Edit Antenna' button in the **Antenna_TX** block mask parameter dialog box to reopen the Antenna Designer app. In the Antenna Properties panel of the Antenna Designer app, change **Tilt** to 30 degrees and the **TiltAxis** to [0 1 0]. Press the **Update Block** button to update the block and rerun the model to observe reduction of 2.5 dB in the output received power due to the mismatch in antenna orientation.

Power Amplifier Characterization

This example shows how to characterize a power amplifier (PA) using measured input and output signals of an NXP Airfast PA. Optionally, you can use a hardware test setup including an NI PXI chassis with a vector signal transceiver (VST) to measure the signals at run time.

You can use the characterization results to simulate the PA using the `comm.MemorylessNonlinearity` (Communications Toolbox) System object or Memoryless Nonlinearity (Communications Toolbox) block. For a PA model with memory, you can use Power Amplifier block. You can use these models to design digital predistortion (DPD) using comm.DPD and comm.DPDCoefficientEstimator System objects or DPD (Communications Toolbox) and DPD Coefficient Estimator (Communications Toolbox) blocks. For more information, see “Digital Predistortion to Compensate for Power Amplifier Nonlinearities” (Communications Toolbox).

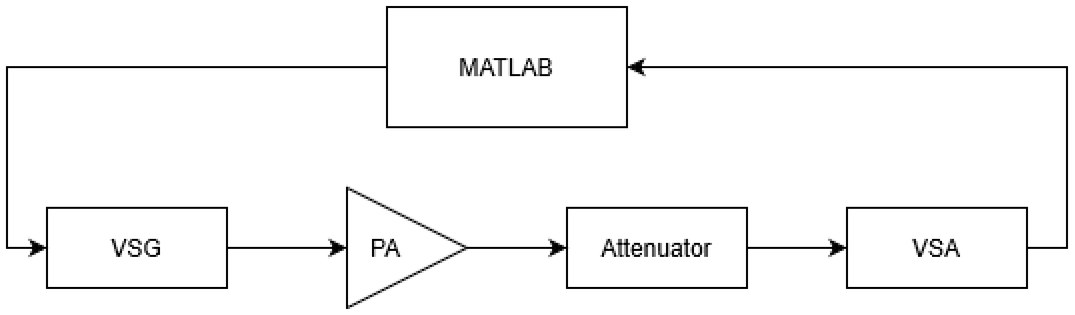
Optional Hardware and Software

This example can run on an NI PXI chassis with a VST to measure PA input and output signals during run time. The VST is a high-bandwidth RF instrument that combines a Vector Signal Generator (VSG) with a Vector Signal Analyzer (VSA). The following NI PXI chassis configuration was used to capture the saved signal:

- NI PXIe-5840 Vector Signal Transceiver (VST)
- NI PXIe-4139 Source Measure Unit (SMU)
- NI PXIe-4145 SMU
- NI RFmx SpecAn software
- NI-RFSG software
- NI-RFSG Playback Library software

As the device under test (DUT), this example uses an NXP Airfast LDMOS Doherty PA with operating frequency 3.6-3.8 GHz and 29 dB gain. This PA requires 29V, 5V, 3 V, 1.6V and 1.4V DC bias, which are provided using PXIe-4139 and PXIe-4145 SMUs.

Install MATLAB® on the NI PXI controller to run this example with the hardware setup, which is illustrated in the following figure. MATLAB, running on the PXI controller, generates test waveform and downloads the waveform to the VSG. The VSG transmits this test waveform to the PA and the VSA receives the impaired waveform at the PA output. MATLAB collects the PA output from the VSA and performs PA characterization.



Set dataSource variable to `"Hardware"` to run a test signal though the PA using the hardware setup described above. The test signal can be either a 5G-like OFDM waveform or two tones, as described in the following section. Set dataSource variable to `"From file"` to use prerecorded data.

```
dataSource = From file ;
```

Generate Test Signals

If `testSignal` is `"OFDM"`, this example uses a 5G-like OFDM waveform with 64-QAM modulated signals for each subcarrier. If `testSignal` is `"Tones"`, this example uses two tones at 1.8 MHz and 2.6 MHz, to test the intermodulation caused by the PA.

```
testSignal = OFDM ;
switch testSignal
  case "OFDM"
    bw = 100 MHz ;
    [txWaveform,sampleRate,numFrames] = helperPACharGenerateOFDM(bw);
  case "Tones"
    bw = 3e6;
    [txWaveform,sampleRate,numFrames] = helperPACharGenerateTones();
end
```

To identify high order nonlinearities, the test signal must be oversampled at least by the amount of expected order of nonlinearity. In this example, we run a grid search up to nonlinearity order of seven. Upsample by seven to cover possible seventh order nonlinearities. Also, normalize the waveform amplitude.

```
overSamplingRate = 7;
filterLength = 6*70;
lowpassfilter = firpm(filterLength, [0 8/70 10/70 1], [1 1 0 0]);
firInterp = dsp.FIRInterpolator(overSamplingRate, lowpassfilter);
txWaveform = firInterp([txWaveform; zeros(filterLength/overSamplingRate/2,1)]);
txWaveform = txWaveform((filterLength/2)+1:end,1);      % Remove transients
txWaveform = txWaveform/max(abs(txWaveform));    % Normalize the waveform
sampleRate = sampleRate * overSamplingRate;
```

Hardware Test

If dataSource variable is set to `"From file"`, load the prerecorded data. `If dataSource` variable is set to `"Hardware",` run the test signal through the PA using the VST. Create a helperVSTDriver object to communicate with the VST device. Set the resource name to the resource name assigned to the VST device. This example uses `'VST_01'`. For NI devices, you can find the resource name using the NI Measurement & Automation Explorer (MAX) application.

```
if strcmp(dataSource, "Hardware")
  VST = helperVSTDriver('VST_01');
```

Set the expected gain values of the DUT and the attenuator. Since PA output is connected to a 30 dB attenuator, set VSA external attenuation to 30. Set the expected gain of the DUT to 29 dB and gain accuracy to 1 dB. Set the acquisition time to a value that will result in about 40k samples. Set the target input power to 8 dBm. You can increase this value to drive the PA more into the non-linear region.

```
  VST.DUTExpectedGain     = 29;     % dB
  VST.ExternalAttenuation = 30;     % dB
  VST.AcquisitionTime     = 0.9e-3*(53.76e6/sampleRate); % seconds

  VST.DUTTargetInputPower = 8                    ;  % dBm
  VST.CenterFrequency     = 3.7e9   % Hz
```

Download the test waveform to the VSG. Measure PA output.

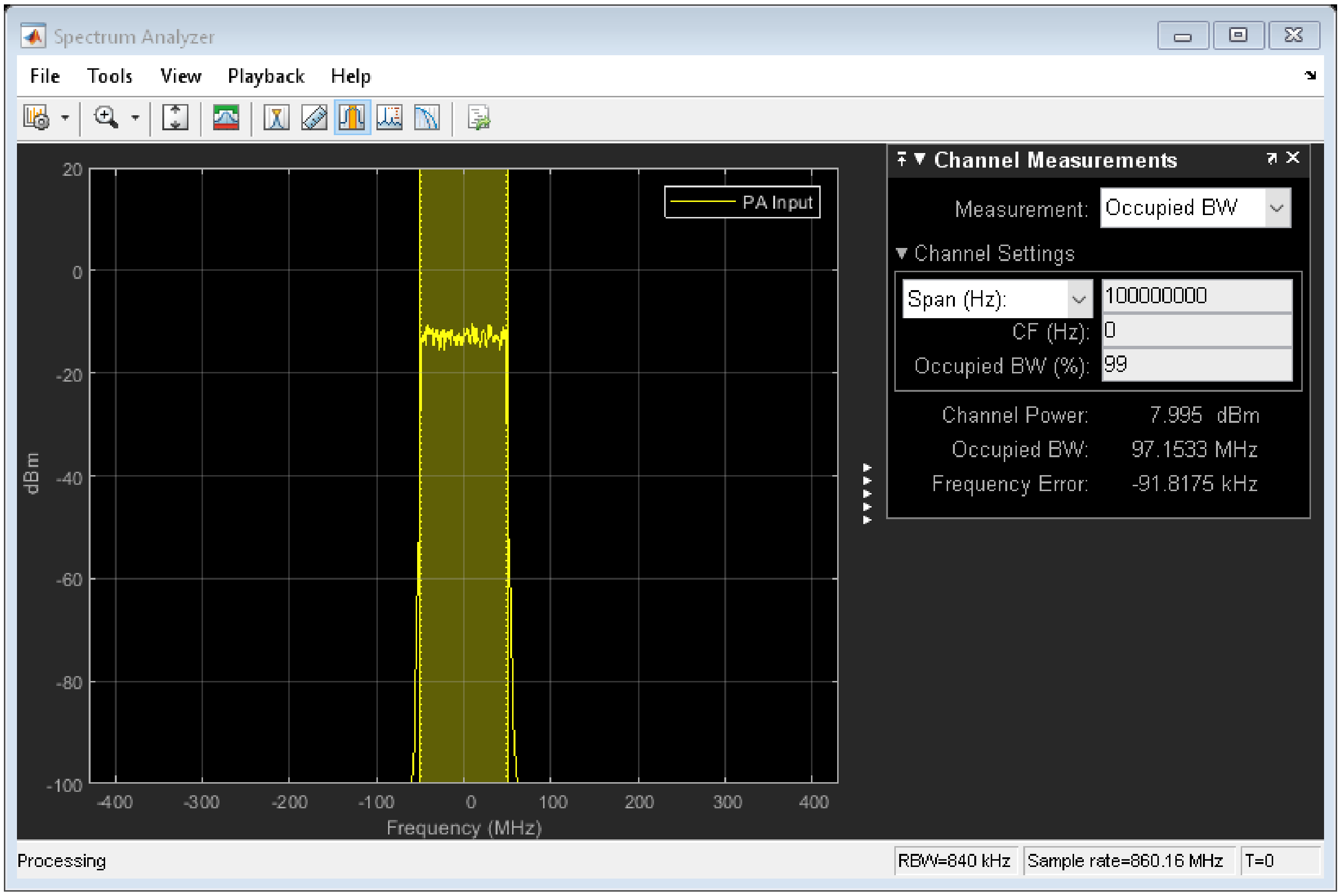
```
  writeWaveform(VST,txWaveform,sampleRate,testSignal)
  results = runPAMeasurements(VST);
  release(VST)
else
  % Load the prerecorded results from VST
  switch testSignal
    case "OFDM"
      dataFileName = sprintf("helperPACharSavedData%dMHz",bw/1e6);
    case "Tones"
      dataFileName = "helperPACharSavedDataTones";
  end
  load(dataFileName,"results","sampleRate","overSamplingRate","testSignal","numFrames")
end
```

Map results into local variables.

```
referencePower = results.ReferencePower;
measuredAMToAM = results.MeasuredAMToAM;
paInput = results.InputWaveform;
paOutput = results.OutputWaveform;
linearGaindB = results.LinearGain;
```

Plot the spectrum of the test signal using `dsp.SpectrumAnalyzer` (DSP System Toolbox) System object.

```
saInput = helperPACharPlotInput(paInput, sampleRate, testSignal, bw);
```



Measurement: Occupied BW

Channel Settings

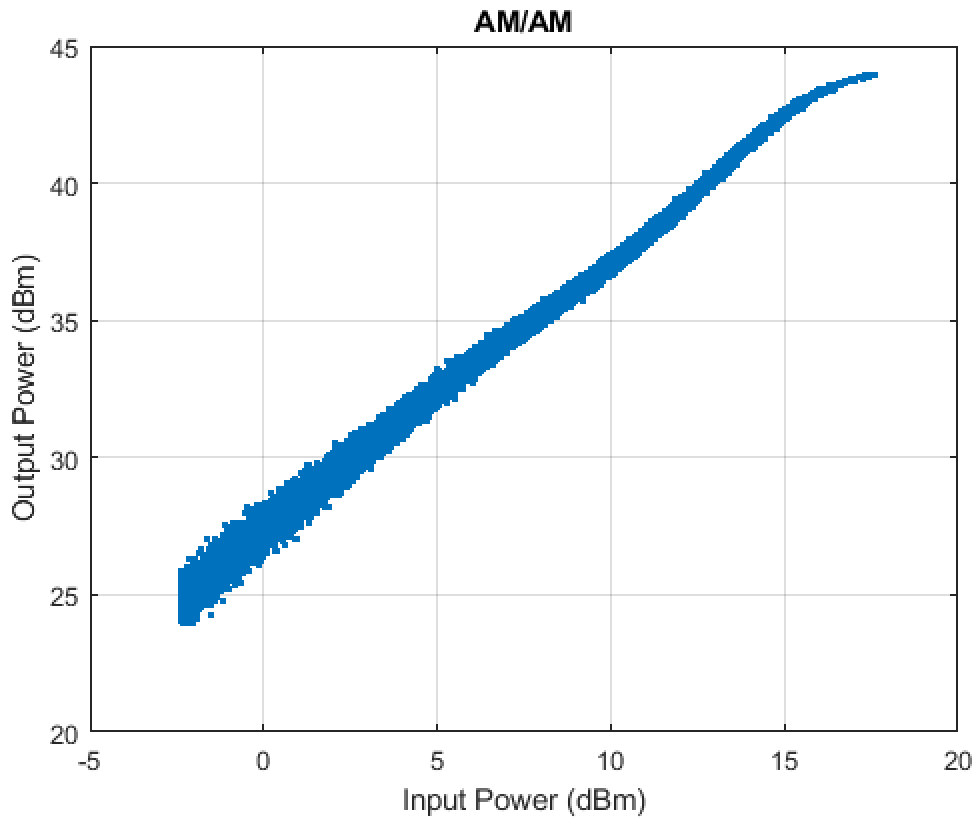
Span (Hz): 100000000
CF (Hz): 0
Occupied BW (%): 99

Channel Power: 7.995 dBm
Occupied BW: 97.1533 MHz
Frequency Error: -91.8175 kHz

Processing | RBW=840 kHz | Sample rate=860.16 MHz | T=0

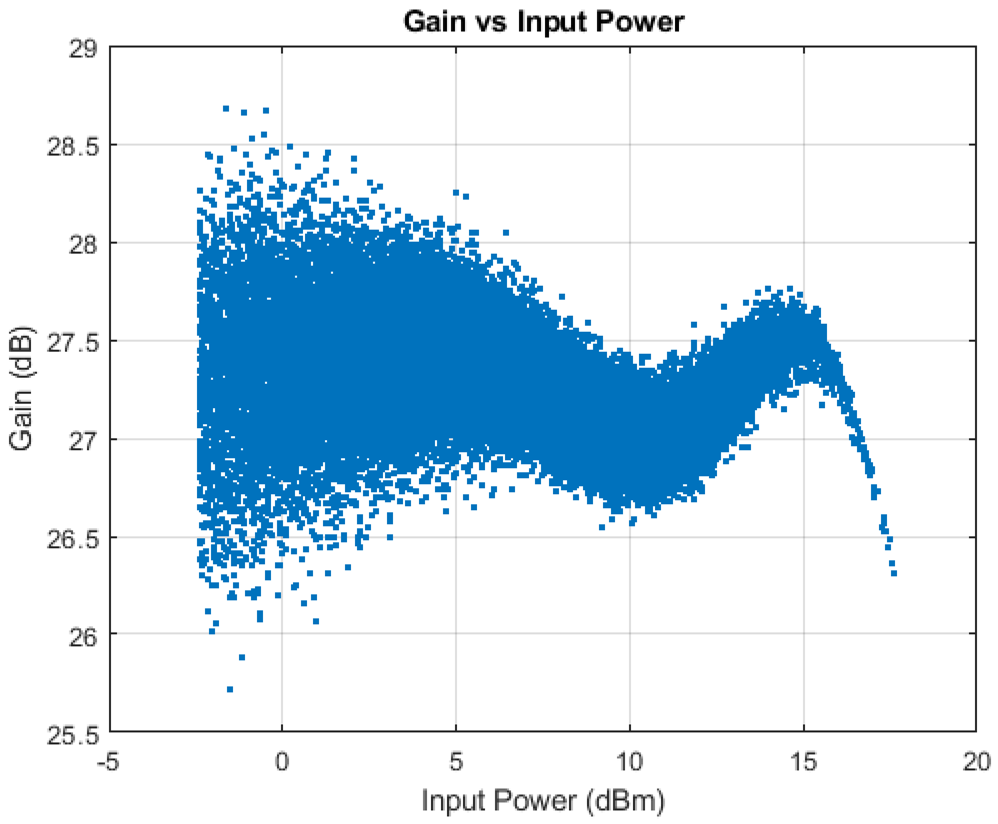
Plot the AM/AM characteristics of the PA.

```
helperPACharPlotSpecAnAMAM(referencePower, measuredAMToAM)
```



For a better view, focus on gain vs input power instead of output power vs input power and plot again.

```
helperPACharPlotSpecAnGain(referencePower, measuredAMToAM)
```



The PA is mostly linear of the input power range -1 to 17 dBm, with only about 1dB variation over that range. The width of the gain curve is due to the memory effects of the PA.

PA Characterization

Use the measured PA input and output data to model the PA. Then, you can use this model to simulate a system that contains this PA and fine tune the parameters. This example considers three models: memoryless nonlinearity, memory polynomial and memory polynomial with cross terms.

Memoryless Nonlinearity Model

Memoryless nonlinear impairments distort the input signal amplitude and phase. The amplitude distortion is amplitude-to-amplitude modulation (AM/AM) and the phase distortion is amplitude-to-phase modulation (AM/PM). The `comm.MemorylessNonlinearity` (Communications Toolbox) System object and Memoryless Nonlinearity (Communications Toolbox) block implements several such distortions. Use the PA input and output data to create a lookup table to use with this object or block.

To characterize the AM/AM transfer function, calculate the average output power for a range of input power values. Measurements are in Volts over 50 Ohm impedance. First convert the measured baseband samples to power values in dBm. +30 dB term is for dBW to dBm conversion and -20 dB term is for 50 Ohm impedance.

```
paInputdBm  = mag2db(abs(paInput)) + 30 - 20;
paOutputdBm  = mag2db(abs(paOutput)) + 30 - 20;
```

First partition the input power values into bins. The `edges` variable contains the bin edges, and the `idx` variable contains the index of the bin values for each input power value.

```
[N,edges,idx] = histcounts(paInputdBm, 'BinWidth', 0.5);
```

For each bin, calculate the midpoint of the bin, average output power and average phase shift. Do not include any input power value that is less than -10 dBm. Store the results in a three-column matrix where the first column is the input power in dBm, second column is the output power in dBm and last column is the phase shift.

```
minInPowerdBm = max(paInputdBm) - 20;
minIdx = find(edges < minInPowerdBm, 1, 'last');
tableLen = length(edges)-minIdx-1;
inOutTable = zeros(tableLen,2);
for p = minIdx+1:length(edges)-1
  inOutTable(p-minIdx,1) = mean(paInputdBm(idx == p));   % Average input power for current bin
  inOutTable(p-minIdx,2) = mean(paOutputdBm(idx == p));  % Average output power for current bin
  inOutTable(p-minIdx,3) = mean(angle(paOutput(idx == p)./paInput(idx == p)));  % Average phase s
end
```

comm.MemorylessNonlinearity assumes 1 Ohm load, however the measurements are done with 50 Ohm. Adjust the table entries with +20 dB correction.

```
inOutTable(:,1:2) = inOutTable(:,1:2) + 20;
```

Use the table in the comm.MemorylessNonlinearity System object to model the PA. Compare the estimated output with the actual output.

```
pa = comm.MemorylessNonlinearity('Method','Lookup table','Table',inOutTable)
```

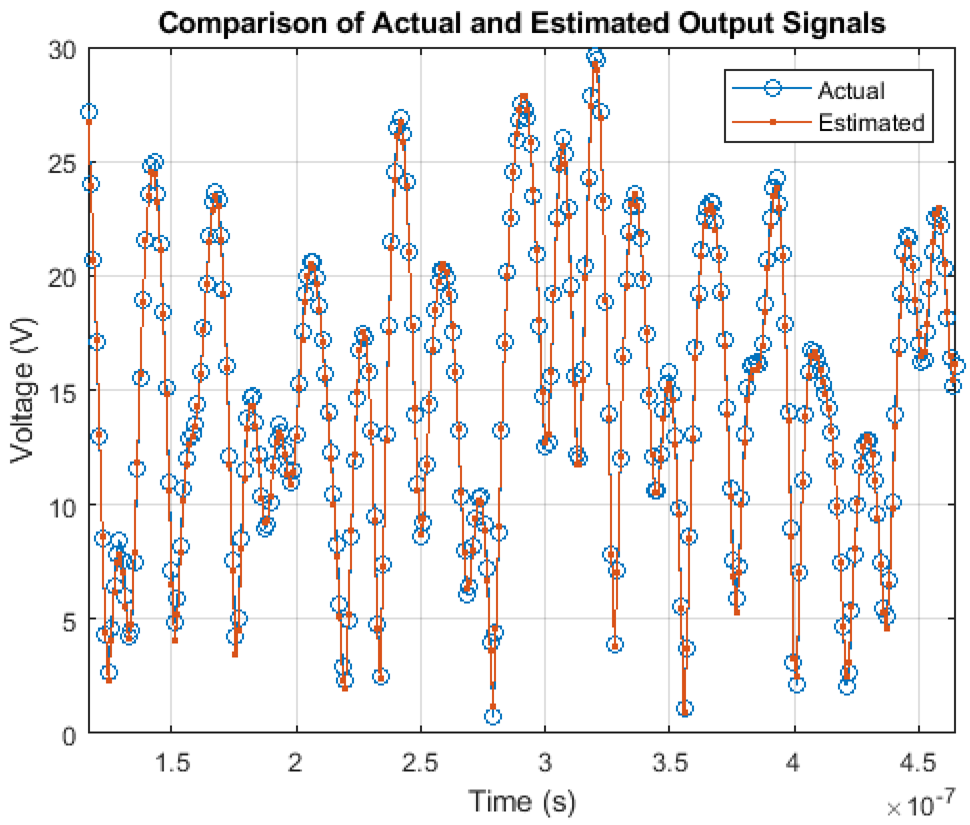
```
pa = 
  comm.MemorylessNonlinearity with properties:

    Method: 'Lookup table'
     Table: [40x3 double]
```

```
paOutputFitMemless = pa(paInput);
err = abs(paOutput - paOutputFitMemless)./abs(paOutput);
rmsErrorMemless = rms(err)*100;
disp(['Percent RMS error in time domain is ' num2str(rmsErrorMemless) '%'])
```

```
Percent RMS error in time domain is 12.1884%
```

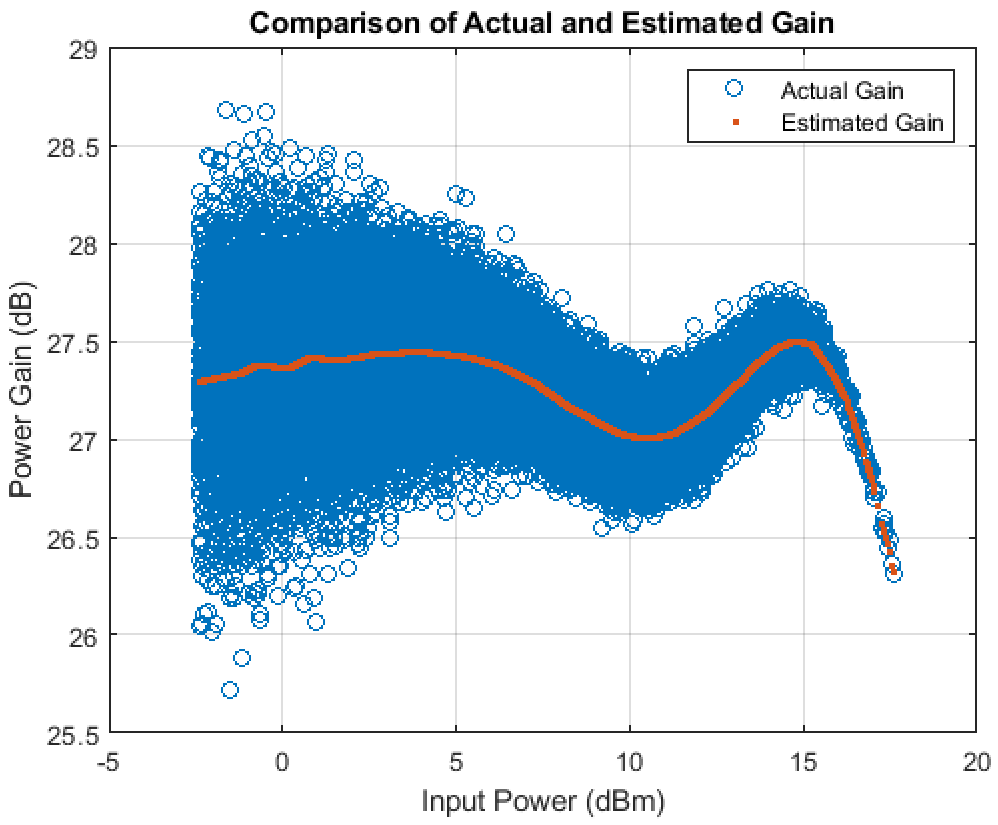
To visualize both the measured output signal and the fitted output signal, plot the actual and fitted time-domain output voltages.

```
helperPACharPlotTime(paOutput, paOutputFitMemless, sampleRate)
```



Plot the magnitude of the gain.

```
helperPACharPlotGain(paInput, paOutput, paOutputFitMemless)
```



Memory Polynomial Model

The memory polynomial model includes the memory effects of the PA in addition to the nonlinear gain. Use the multipurpose helper function helperPACharMemPolyModel to determine the complex coefficients of a memory polynomial model for the amplifier characteristics. Set the model type to `'Memory Polynomial'`.

```
modType = Memory Polynomial ;
```

Perform a grid search as shown in Appendix Grid Search for Memory Length and Polynomial Order on page 8-0 . Based on this grid search results, the best fit is obtained when memory length and polynomial degree values are as follows:

```
memLen = 5;
degLen = 5;
```

Perform the fit and RMS error calculation for these values. Only half of the data is used to compute the fitting coefficients, as the whole data set will be used to compute the relative error. The helper function helperPACharMemPolyModel calculates the coefficients of the model.

```
numDataPts = length(paInput);
halfDataPts = round(numDataPts/2);
```

The helper function helperPACharMemPolyModel is editable for custom modifications, and to return the desired matrix. The PA model has some zero valued coefficients, which results in a rank deficient matrix.

```
fitCoefMatMem = helperPACharMemPolyModel('coefficientFinder',          ...
  paInput(1:halfDataPts),paOutput(1:halfDataPts),memLen,degLen,modType);
```

```
Warning: Rank deficient, rank = 24, tol =  1.870573e-01.
```

```
disp(abs(fitCoefMatMem))
```

```
   23.1553    8.8539   17.8403   13.3042    3.2174
         0   11.7661   26.4612   23.1878    5.5464
   20.9748   16.8495   25.7229   22.1851    5.0674
   32.6203    8.4018    9.4817   10.6939    2.5605
   15.3880    2.3642    2.0892    2.9342    0.7370
```

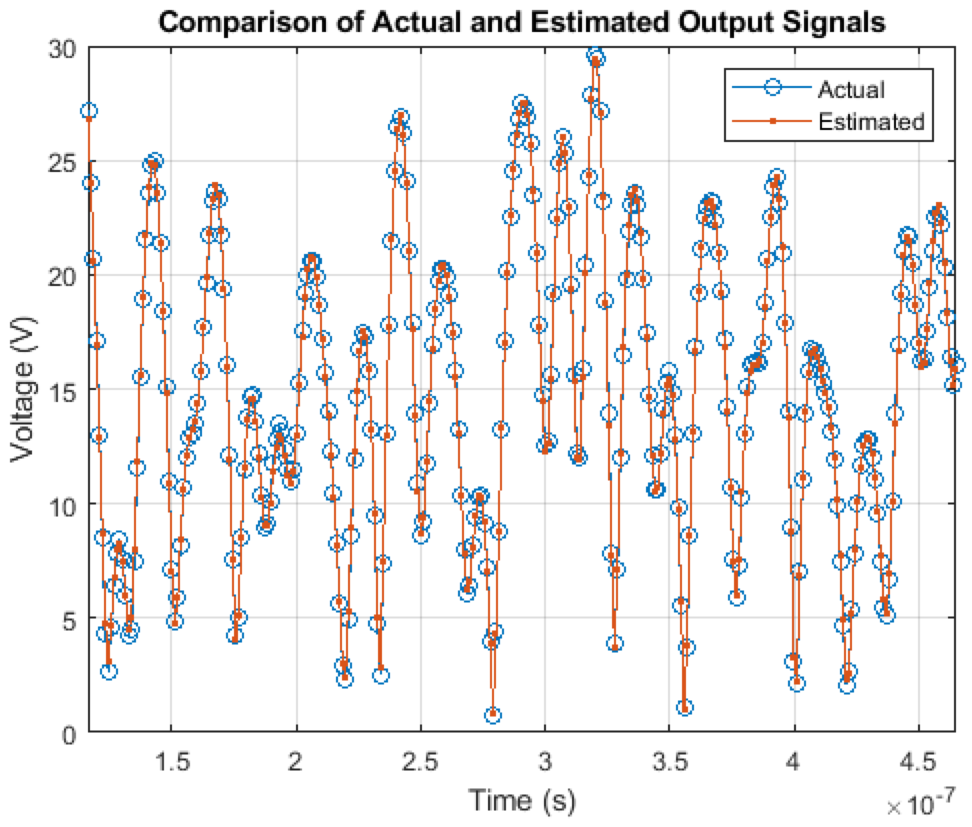
To validate the fitting, use the helper function to compute percent RMS error with respect to the measured signal.

```
rmsErrorTimeMem = helperPACharMemPolyModel('errorMeasure', ...
  paInput, paOutput, fitCoefMatMem, modType);
disp(['Percent RMS error in time domain is ' num2str(rmsErrorTimeMem) '%'])
```

```
Percent RMS error in time domain is 6.1056%
```

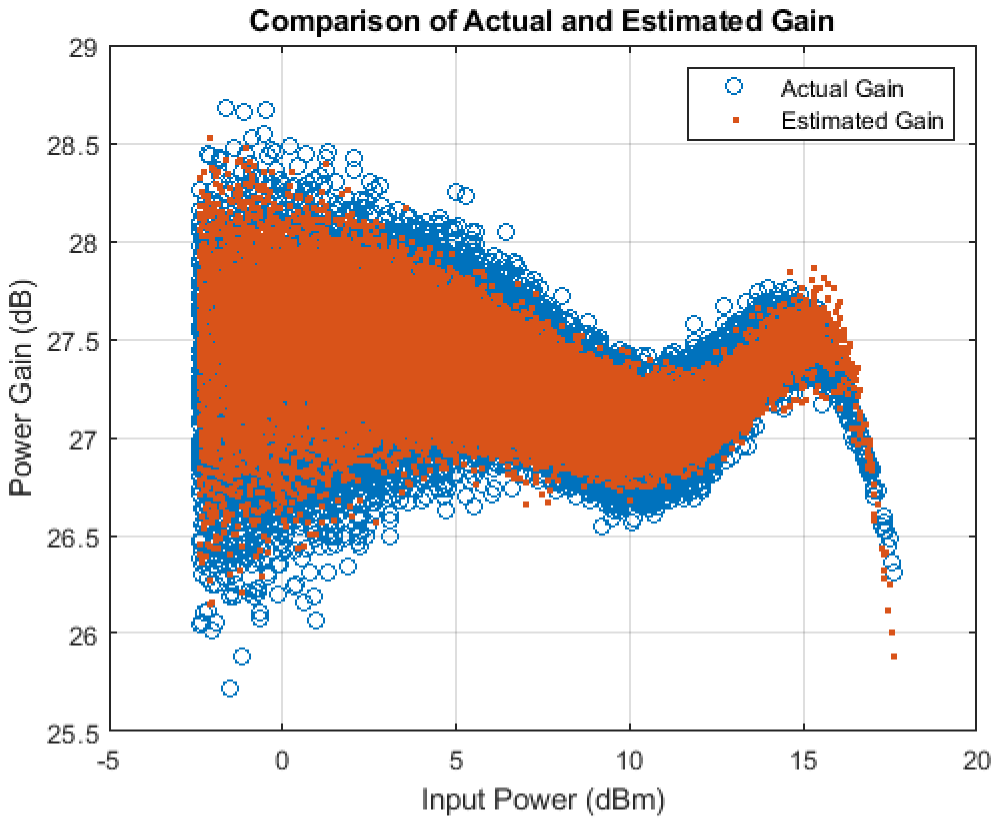
To visualize both the measured output signal and the fitted output signal, plot the actual and fitted time-domain output voltages.

```
paOutputFitMem = helperPACharMemPolyModel('signalGenerator', ...
  paInput, fitCoefMatMem, modType);
helperPACharPlotTime(paOutput, paOutputFitMem, sampleRate)
```



Plot the magnitude of the gain.

```
helperPACharPlotGain(paInput, paOutput, paOutputFitMem)
```

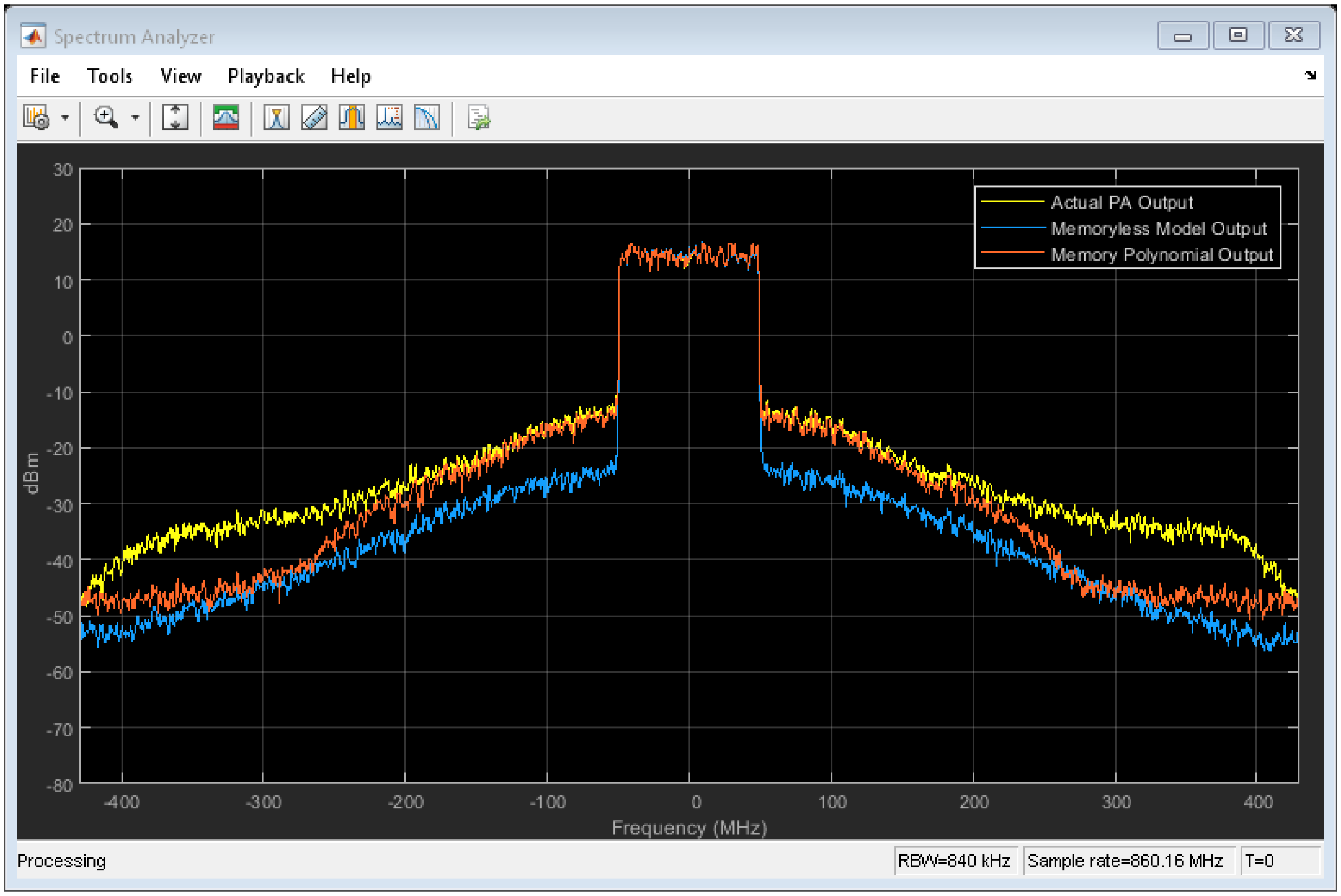


Discussions

The percent RMS estimation error in time domain for the memoryless nonlinearity model, which is between 9% and 13%, is about 3 to 4 times more than the error for the memory polynomial model is, which is between 2% and 6%, for the OFDM signals with different bandwidths.

Check the estimation error in frequency domain by plotting the spectrum of the actual PA output together with the spectrum of the estimated PA output for all three models. The memoryless nonlinearity table lookup model is not able to simulate the spectral growth seen in the measured PA output. For this PA, memory polynomial model provides a good approximation of the PA characteristics.

```
sa = helperPACharPlotSpectrum(...
  [paOutput paOutputFitMemless paOutputFitMem],...
  {'Actual PA Output','Memoryless Model Output', ...
  'Memory Polynomial Output'},...
  sampleRate,testSignal);
```



The helper function helperPACharMemPolyModel can also use the memory polynomial with cross terms model, which includes the leading and lagging memory cross terms in addition to the memory effects of the PA and the nonlinear gain. Set the model type to `'Cross-Term Memory'` to explore this model.

For further exploration, try different memory length and polynomial degree combinations. Modify the oversampling factor and explore its effect on the PA model performance. Modify the helper function helperPACharMemPolyModel to try different PA models.

Using PA Model for DPD Testing

Save the coefficient matrix of the PA model to be used in the Power Amplifier for simulation at the system-level in the “Digital Predistortion to Compensate for Power Amplifier Nonlinearities” (Communications Toolbox).

```
frameSize = floor(length(paInput)/numFrames);
paIn.signals.values = double(reshape(paInput(1:frameSize*numFrames,1),numFrames,frameSize));
paIn.signals.dimensions = frameSize;
paIn.time = [];
save('PAcoefficientsAndInput.mat','modType','fitCoefMatMem','memLen','degLen','paIn','linearGainc
```

Appendix: Grid Search for Memory Length and Polynomial Order

Uncomment following lines to perform the grid search when the cost function is the percent RMS error in time. First choose the model type.

```
modType = Memory Polynomial ;
% rmsErrorTime = helperPACharGridSearchTime(paInput,paOutput,modType,overSamplingRate)
```

Repeat the search when the cost function is the percent RMS error in frequency.

```
% rmsErrorFreq = helperPACharGridSearchFrequency(paInput,paOutput,modType,overSamplingRate)
```

RF Blockset Examples

Getting Started with RF Modeling

Learn how to use the **RF Budget Analyzer** app to build a simple RF receiver, and then create an RF Blockset™ Circuit Envelope multi-carrier model to perform simulation.

Build a Cascade (row vector) of RF Elements.

You can build and analyze an RF cascade by adding elements characterized by their data sheet specifications.

You can use the **RF Budget Analyser** app and drag and drop new elements, or you can script the chain elements using MATLAB® commands. If you are not familiar with the syntax, you can start with app and generate a MATLAB script.

Add elements to your chain in the following order:

- Filter specified by an S-parameters Touchstone file
- Low Noise Amplifier (LNA)
- Direct conversion demodulator
- Baseband amplifier

```
elements(1) = nport('sawfilterpassive.s2p');
elements(2) = amplifier(                                                   ...
    'Name','LNA',                                                          ...
    'Gain',18,                                                             ...
    'NF',3,                                                                ...
    'OIP3',10);
elements(3) = modulator(                                                   ...
    'Name','Demod',                                                        ...
    'Gain',10,                                                             ...
    'NF',6.4,                                                              ...
    'OIP3',36,                                                             ...
    'LO',2.45e9,                                                           ...
    'ConverterType','Down');
elements(4) = amplifier(                                                   ...
    'Gain',20,                                                             ...
    'NF',11.3,                                                             ...
    'OIP3',42);
```

Inspect RF Budget Using RF Budget Analyzer App

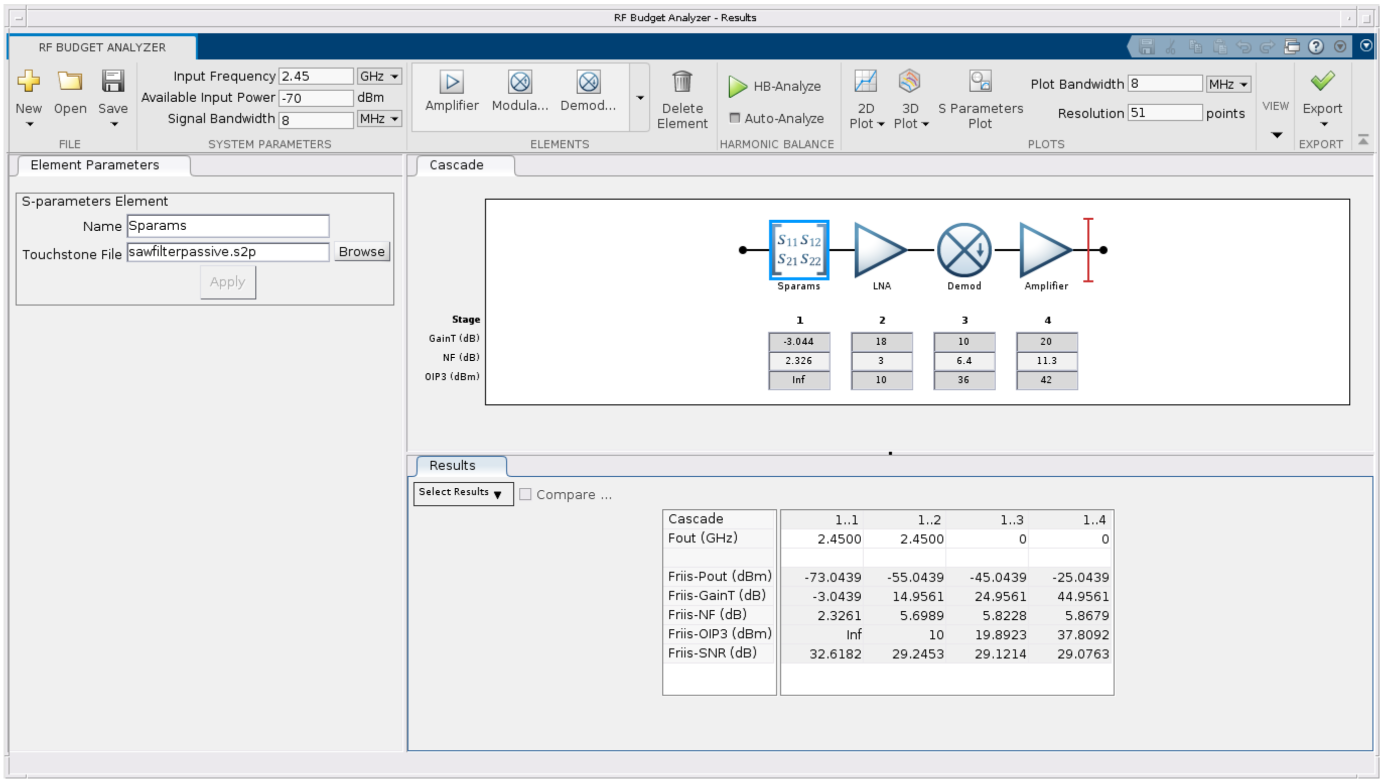
Create an `rfbudget` object. The MATLAB command window dynamically displays the budget analysis results.

```
b = rfbudget(                                                              ...
    'Elements',elements,                                                   ...
    'InputFrequency',2.45e9,                                               ...
    'AvailableInputPower',-70,                                             ...
    'SignalBandwidth',8e6)
```

```
b = 

  rfbudget with properties:

             Elements: [1x4 rf.internal.rfbudget.Element]
```

```
          InputFrequency: 2.45 GHz
     AvailableInputPower:  -70 dBm
         SignalBandwidth:    8 MHz
                  Solver: Friis
              AutoUpdate: true

    Analysis Results
         OutputFrequency: (GHz) [  2.45    2.45       0       0]
             OutputPower: (dBm) [-73.04  -55.04  -45.04  -25.04]
          TransducerGain: (dB)  [-3.044   14.96   24.96   44.96]
                      NF: (dB)  [ 2.326   5.699   5.823   5.868]
                    IIP2: (dBm) []
                    OIP2: (dBm) []
                    IIP3: (dBm) [   Inf  -5.674  -5.782  -7.865]
                    OIP3: (dBm) [   Inf      10   19.89   37.81]
                     SNR: (dB)  [ 32.62   29.25   29.12   29.08]
```

Or you can visualize the `rfbudget` object in the app using the MATLAB command `show(b)`.



Input Frequency 2.45 GHz
Available Input Power -70 dBm
Signal Bandwidth 8 MHz
Plot Bandwidth 8 MHz
Resolution 51 points

S-parameters Element
Name Sparams
Touchstone File sawfilterpassive.s2p

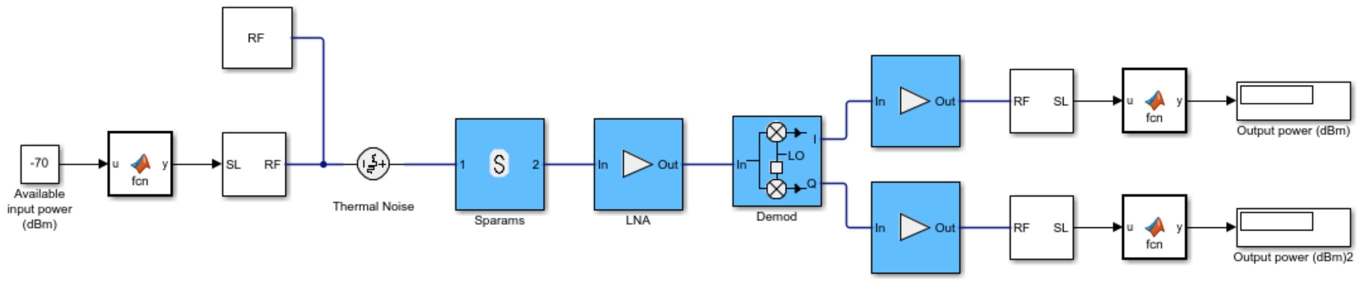
Stage	1	2	3	4
GainT (dB)	-3.044	18	10	20
NF (dB)	2.326	3	6.4	11.3
OIP3 (dBm)	Inf	10	36	42

Cascade	1..1	1..2	1..3	1..4
Fout (GHz)	2.4500	2.4500	0	0
Friis-Pout (dBm)	-73.0439	-55.0439	-45.0439	-25.0439
Friis-GainT (dB)	-3.0439	14.9561	24.9561	44.9561
Friis-NF (dB)	2.3261	5.6989	5.8228	5.8679
Friis-OIP3 (dBm)	Inf	10	19.8923	37.8092
Friis-SNR (dB)	32.6182	29.2453	29.1214	29.0763

Generate RF Blockset Model

Use the **Export** button in the **RF Budget Analyzer** app to create an RF Blockset model or:

```
exportRFBlockset(b)
save_system(gcs,'model_1')
```



You can use this model for multi-carrier circuit envelope simulation. The Input Port / Output Port ports and Configuration block are set up correctly and you can copy the model for use in any other Simulink® testbench.

- The input port specifies a complex powerwave signal centered at 2.45 GHz.
- The output ports terminate the cascade and extract the envelope centered at DC (0 Hz). The I and Q signals are real baseband signals.
- The Configuration block runs the simulation for a total of 8 simulation frequencies in order to capture the non linearity introduced by the demodulator and amplifiers.
- The simulation stop time in this case is set equal to 0. This means that the simulation does only a static analysis of the model (harmonic balance).

Observe and understand the model blocks:

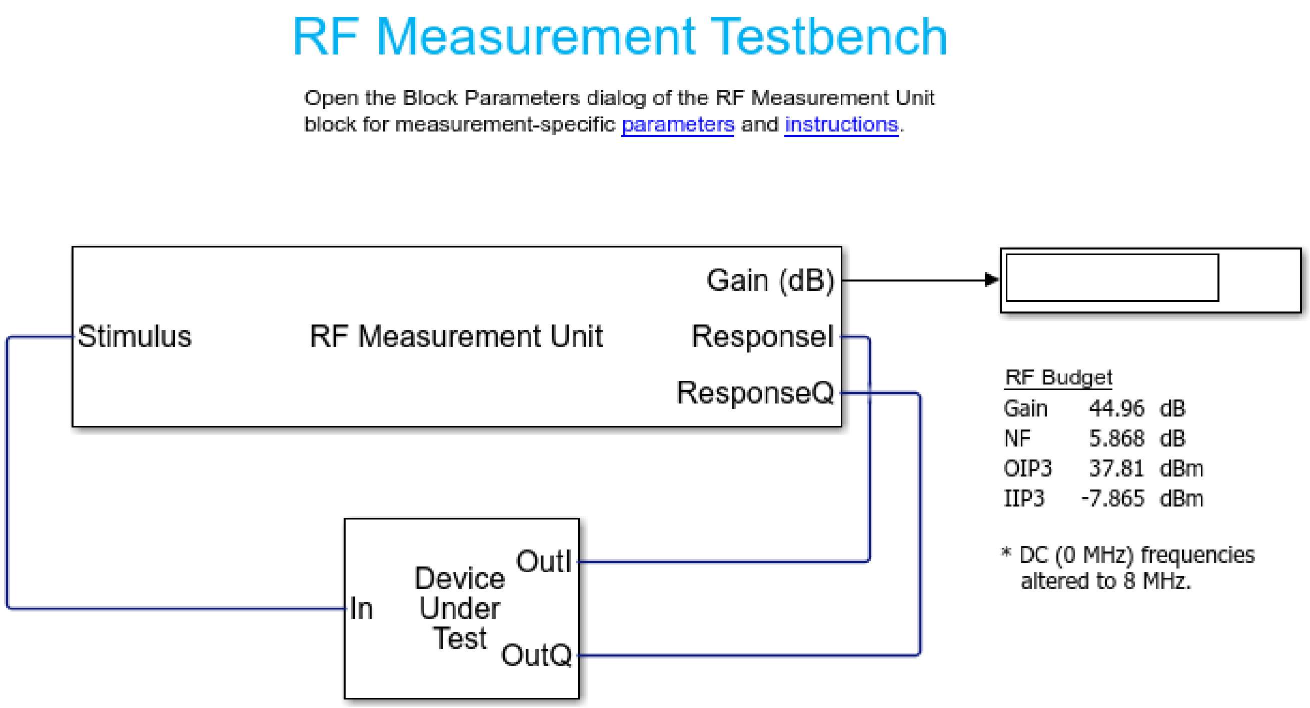
- The S-parameter block describing the filter uses rational fitting in order to simulate frequency data in the time domain. Notice that at 2.45 GHz it introduces a phase rotation of approximately -58 degrees.
- Both amplifiers specify IP3, but you can also specify IP2.
- The demodulator includes ideal channel selection filters. Additional impairments can be added such as LO leakage and I/Q imbalance.

Simulate the model to compare the output power values with the **RF Budget Analyzer** app values. Notice that due to the phase rotation introduced by the S-parameter block, the complex input signal is partly downconverted on the I and on the Q branch, and thus the output power on the two branches is different. For this reason, the gain and other specs of direct conversion receivers, are measured at an arbitrary low frequency.

Generate Measurement Testbench

Use the **Export** button in the **RF Budget Analyzer** app to create a measurement testbench or:

```
exportTestbench(b)
save_system(gcs,'model_2')
```


RF Measurement Testbench
Open the Block Parameters dialog of the RF Measurement Unit block for measurement-specific parameters and instructions.
Stimulus
RF Measurement Unit
Gain (dB)
ResponseI
ResponseQ
In
Device Under Test
OutI
OutQ

RF Budget

Gain	44.96	dB
NF	5.868	dB
OIP3	37.81	dBm
IIP3	-7.865	dBm

* DC (0 MHz) frequencies altered to 8 MHz.

To measure the gain, noise figure, and OIP3 use the RF Measurement Unit dialog box to choose the value you want to verify.

Observe and understand the testbench block:

- You can measure the output on the I or Q branches.
- Measurements are done at an arbitrary low frequency
- Measurements are done in the time domain over an arbitrary signal bandwidth

Run the following simulation:

- Measure the gain (disable the noise for accurate measurements).
- Measure the NF. Reduce the baseband bandwidth to 8e3 for narrowband measurements. In this way, the noise figure measurement is not affected by the filter selectivity.
- Measure the OIP3. Keep the smaller baseband bandwidth and disable noise for accurate measurements.

On comparing, you will see that the values of gain, noise figure, and IP3 match the values in the **RF Budget Analyzer** app reported in the testbench.

See Also

RF Budget Analyzer| “Using RF Blockset for the First Time”| “Power Ports and Signal Power Measurement in RF Blockset” on page 9-11| “Create Custom RF Blockset™ Models” on page 9-108

Passband Signal Representation in Circuit Envelope

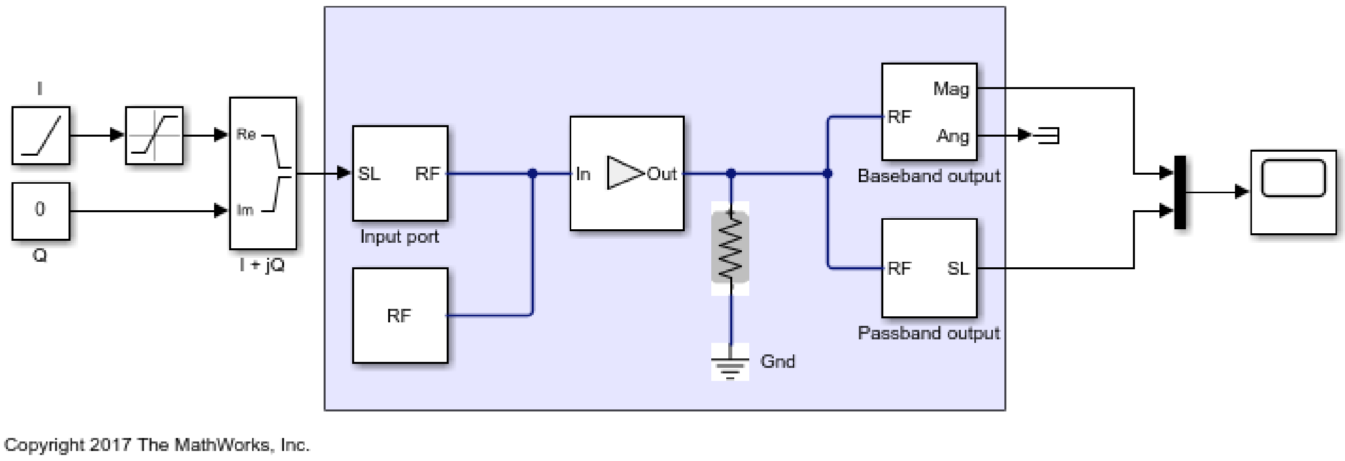
This model shows the relationship between two signal representations in RF Blockset™ Circuit Envelope: complex baseband (envelope) signal and passband (time domain) signal. The step size of a RF Blockset solver is usually much larger than the period of the carrier, so upsampling is necessary to construct a reasonable passband signal.

System Architecture

The system consists of:

- Simulink blocks that generate a complex **I+jQ** input baseband signal.
- A RF Blockset Inport block that specifies the carrier frequency of the signal as **f = 3GHz** .
- A simple RF Blockset system that consists of an Amplifier with **0dB** gain and matching **50 Ohm** load (that is, its input and output signals are identical). It has two outports: baseband (where the complex envelope signal **I+jQ** is represented as magnitude and angle) and passband, where the actual time domain signal is reconstructed.
- A Scope block that displays baseband magnitude (that is, the envelope of the signal) versus the passband (actual) signal.

```
model = 'simrfV2_passband';
open_system(model)
```



Definition of the Passband Signal

RF Blockset interprets the complex signal $I(t) + jQ(t)$, as a modulation (envelope) of the sinusoidal carrier signal with a frequency f. By default, RF Blockset assumes that the carrier signal is normalized (that is, its average power is equal to 1), so the passband signal is

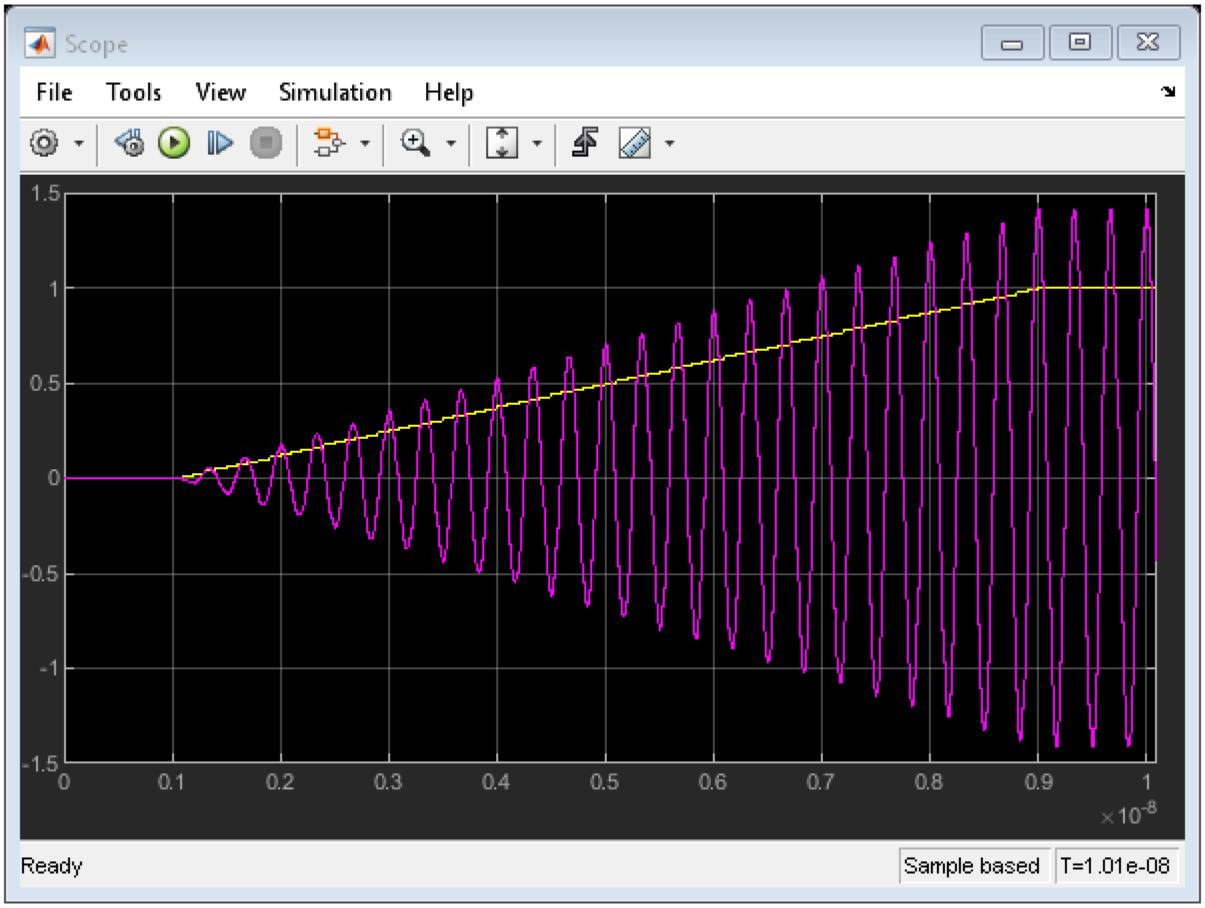
$$s(t) = I(t)\sqrt{2}\cos 2\pi ft - Q(t)\sqrt{2}\sin 2\pi ft$$

With this definition, the average power of the signal is

$$\overline{s^2(t)} = I^2 + Q^2.$$

In this example, I is a ramp that goes from 0 to 1 and $Q = 0$.

```
scope = [model '/Scope'];
set_param(scope, 'YMax','1.5');
set_param(scope, 'YMin','-1.5');
open_system(scope)
sim(model);
```



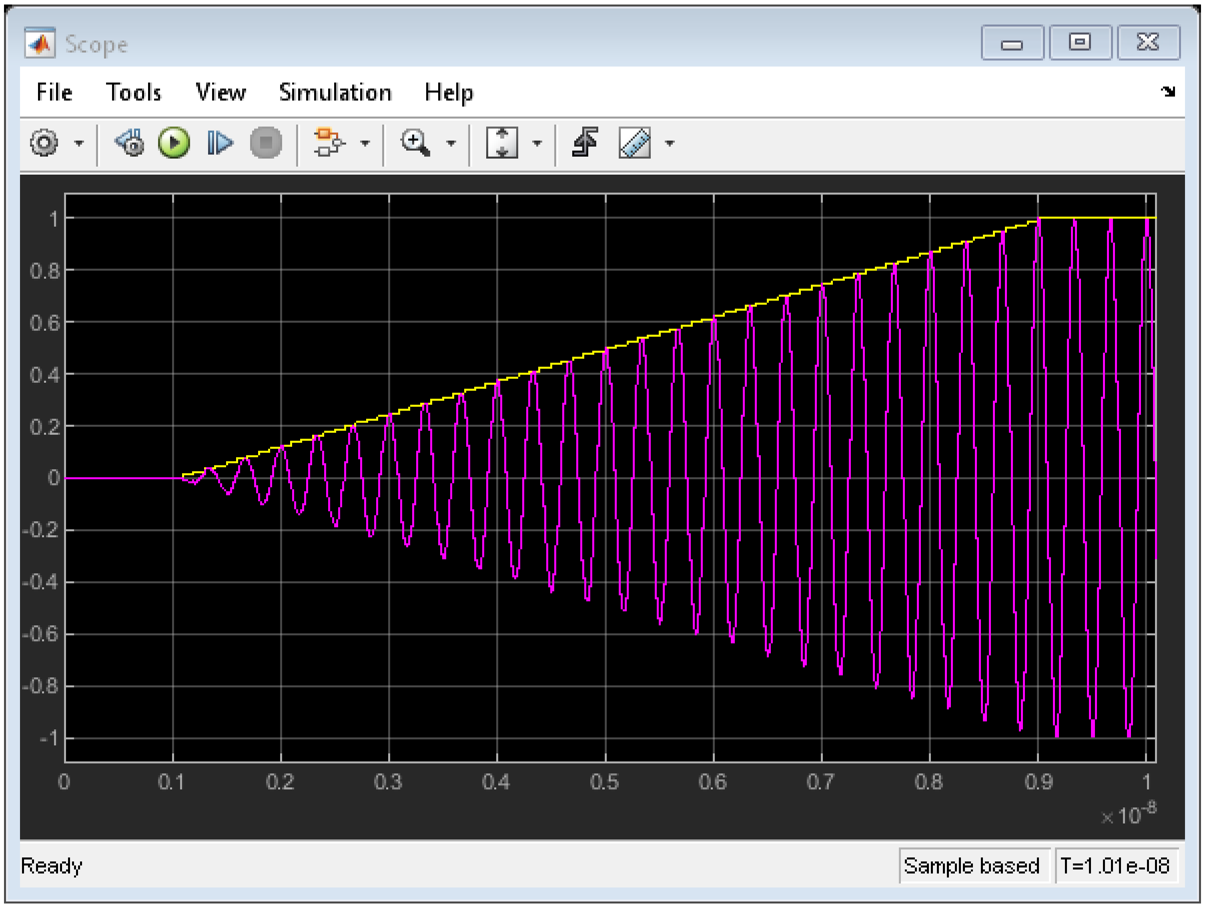
When the **Normalize Carrier Power** Option on the **Configuration** block is not selected, RF Blockset assumes that $I(t) + jQ(t)$ represent the peak values of the carrier, that is

$$s(t) = I(t)\cos 2\pi ft - Q(t)\sin 2\pi ft$$

and the average power of the signal is therefore

$$\overline{s^2(t)} = (I^2 + Q^2)/2$$

```
params = [model '/Configuration'];
set_param(params, 'NormalizeCarrierPower', 'off')
set_param(scope, 'YMax','1.1');
set_param(scope, 'YMin','-1.1');
sim(model);
```



Effects of the Normalize Carrier Power Option

It is very important to understand that when you change the **Normalize Carrier Power** option, RF Blockset changes the interpretation of the complex input/output $I(t) + jQ(t)$ baseband signal. Consider the simple case when the input baseband voltage is constant, $I = 1$ and $Q = 0$. The amplifier has a gain of **0dB**, which means that the output signal is the same as the input.

When the **Normalize** option is checked, the output baseband voltage is equal to $I_{out} = 1$, the output passband voltage is $\sqrt{2}\cos 2\pi ft$, and the average power at the **R = 50 Ohm** load is $1^2/50 = 0.02$.

When the **Normalize** option is unchecked, the output baseband signal does not change, $I_{out} = 1$, while the output passband signal is now $\cos 2\pi ft$, which means that the average power is $1^2/50/2 = 0.01$.

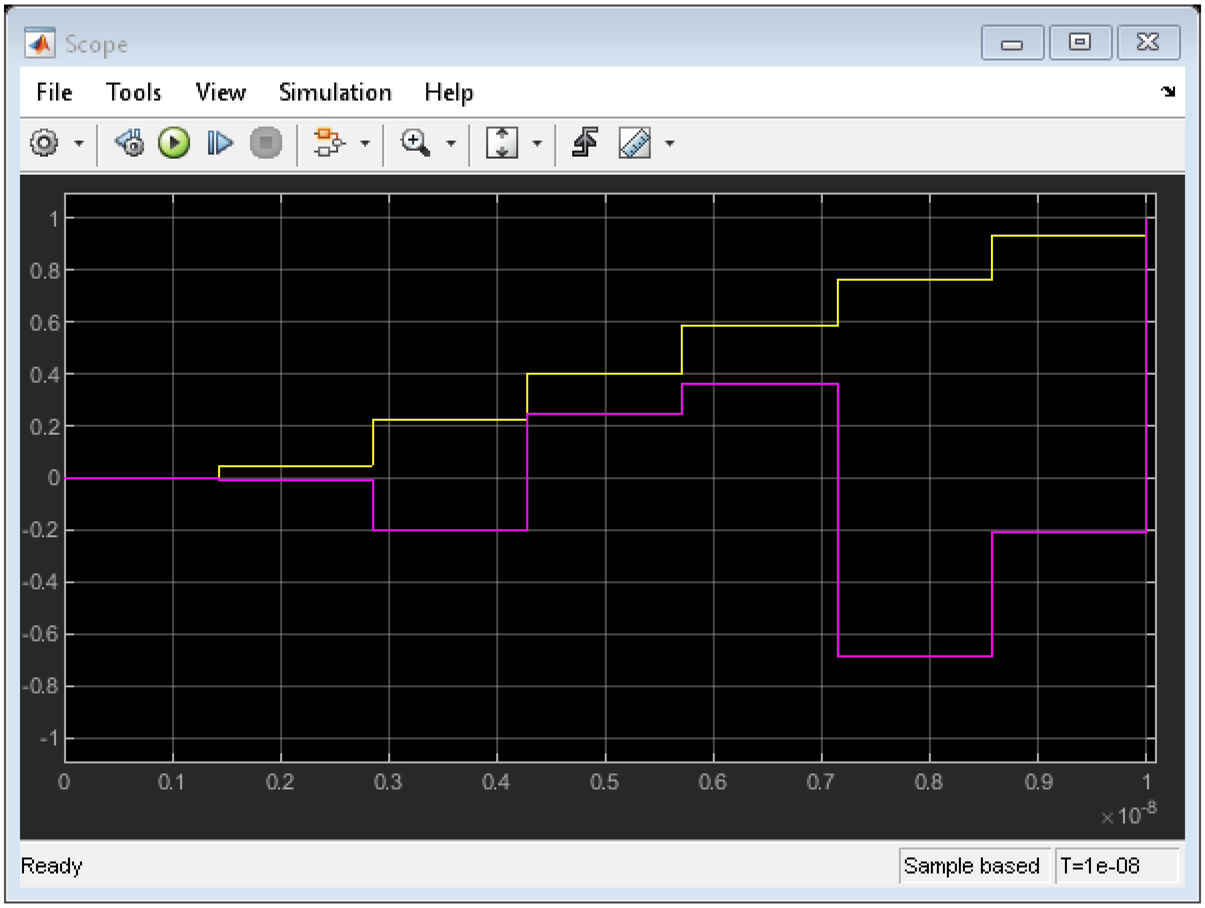
In other words, for linear models the **Normalize** option does not affect the baseband output, but affects the actual passband signal and the average power formula.

Note that the zero carrier frequency is special: the passband and baseband representations for $f = 0$ are always the same: $s(t) = I(t)$

Simulation Step Size Versus Passband Output Step Size

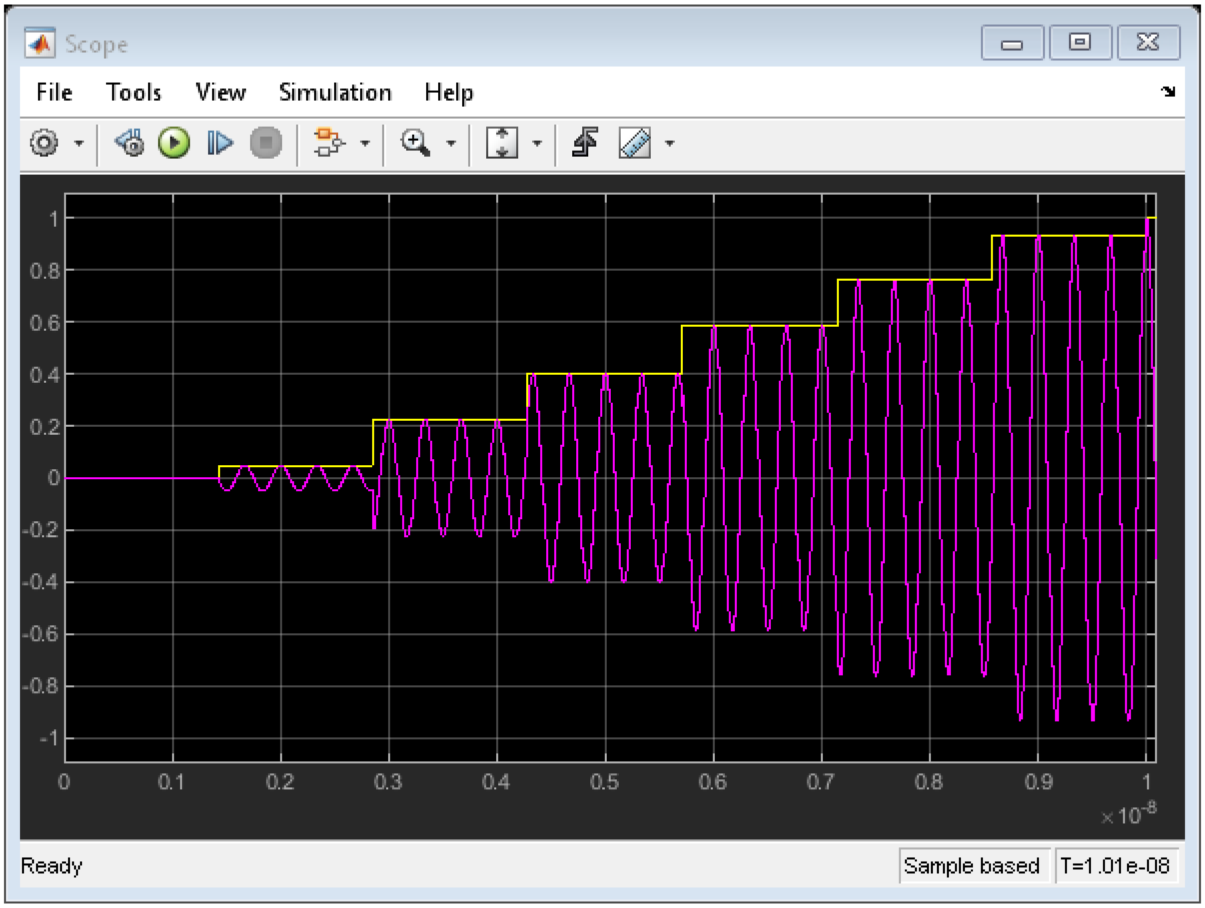
In general, the RF Blockset simulation step is much larger than the period of the carrier, which allows faster simulation compared to regular methods. For such time steps the passband output is severely undersampled and exhibits aliasing effects. Set the **Step Size** value of the **Configuration** block to the large value **1e-8/7**

```
set_param(params, 'StepSize', '1e-8/7')
sim(model);
```



To obtain a realistic passband signal, resample the signal in the outport. Change the **Step Size** parameter of the **Passband output** block from **-1** (which means that the step size is inherited from RF Blockset simulation) to **1e-11** .

```
outport = [model '/Passband output'];
set_param(outport, 'StepSize', '1e-11');
sim(model);
```



Notes:

- Generating passband output at a higher rate (compared to RF Blockset simulation) requires resampling the signal's envelope. Current implementation uses a zero-hold resampling method that introduces "stepping" artifacts. Better interpolation techniques require delaying the output by several time steps.
- The **'auto'** time step option is available on the RF Blockset Outport block (the time step is selected to resolve the highest output carrier frequency).
- Passband output might slow RF Blockset simulation because of the higher output sampling rate.

```
bdclose(model)
```

See Also

Configuration | Amplifier

Related Topics

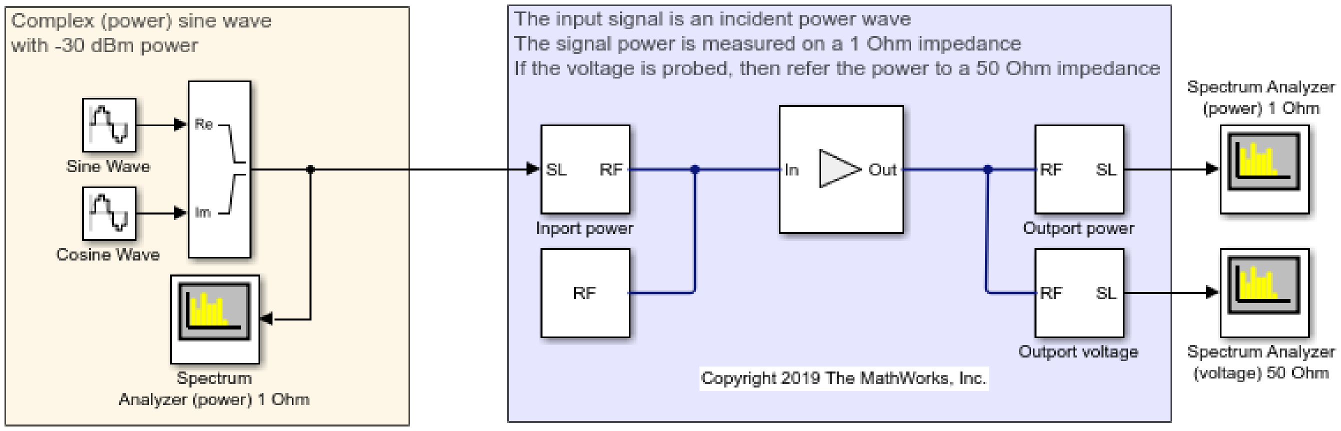
“Getting Started with RF Modeling” on page 9-2

Power Ports and Signal Power Measurement in RF Blockset

This example shows how to use power ports and measure the signal power using the spectrum analyzer.

Power Model

```
model = 'simrfV2_powerportdefinition_1.slx';
open(model)
```


Complex (power) sine wave
with -30 dBm power
Re
Im
Sine Wave
Cosine Wave
Spectrum
Analyzer (power) 1 Ohm
The input signal is an incident power wave
The signal power is measured on a 1 Ohm impedance
If the voltage is probed, then refer the power to a 50 Ohm impedance
SL
RF
Inport power
In
Out
Outport power
Outport voltage
Copyright 2019 The MathWorks, Inc.
Spectrum Analyzer
(power) 1 Ohm
Spectrum Analyzer
(voltage) 50 Ohm

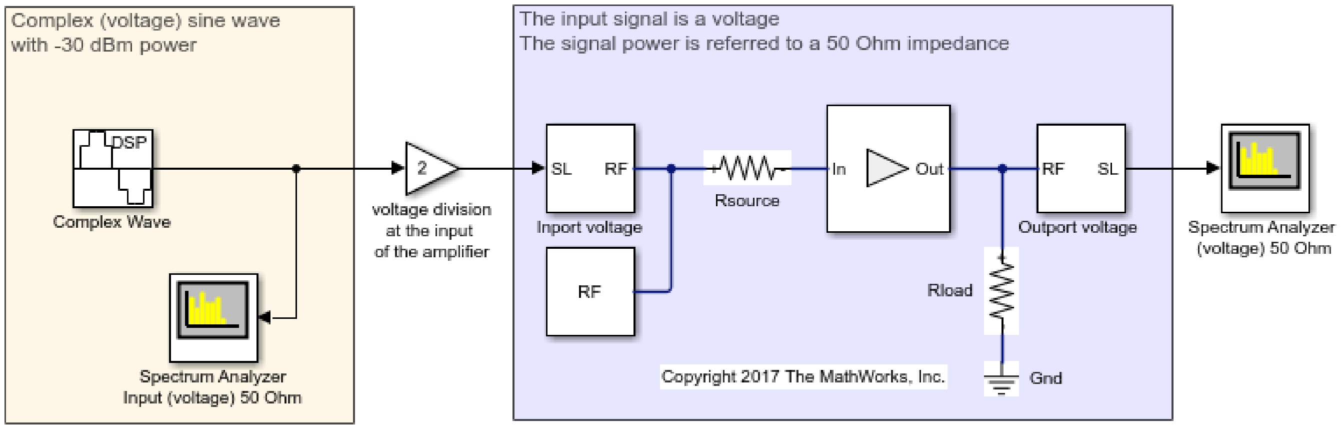
In this model, the Simulink signal is interpreted as a `|Vrms^2 signal|` referred to 1 Ohm. This is the implicit convention used in the DSP System Toolbox.

When measuring the signal power with the spectrum analyzer, refer the measurement to 1 Ohm. Use the power option for the ports at the input and output of the system. Notice that the ports automatically add source and load impedances, and scale the signal to refer it to the specified impedance. In this way, the available power of the input signal is referred to 50 Ohm. The RF Blockset amplifier is thus processing a signal with the same power as the Simulink signal.

When probing internal nodes, use the voltage option in the "sensing" port to avoid mismatch in the circuit. Also, adjust the reference impedance in the spectrum analyzer to 50 Ohm.

Voltage Model

```
model = 'simrfV2_powerportdefinition_2.slx';
open(model)
```



In this model, the Simulink signal is interpreted as a Voltage signal. When measuring the signal power with the spectrum analyzer, refer the measurement to 50 Ohm. Notice that the two sources and the "Real-Imag to Complex" of the Power Model are replaced using a single Sine Wave block with property `Output complexity` set to "Complex".

In RF Blockset, use the voltage option for the ports at the input and output of the system. Add source and load impedances to avoid mismatch in the amplifier. Also add a factor of 2 to make sure that the voltage at the input of the RF Blockset amplifier has the same value as the Simulink signal. The factor of 2 takes into account the voltage division between the source and input impedances.

Simulate both the models and observe.

See Also

Amplifier | Inport | Spectrum Analyzer

Related Topics

“Getting Started with RF Modeling” on page 9-2 | “Passband Signal Representation in Circuit Envelope” on page 9-6

Communications System with Embedded RF Receiver

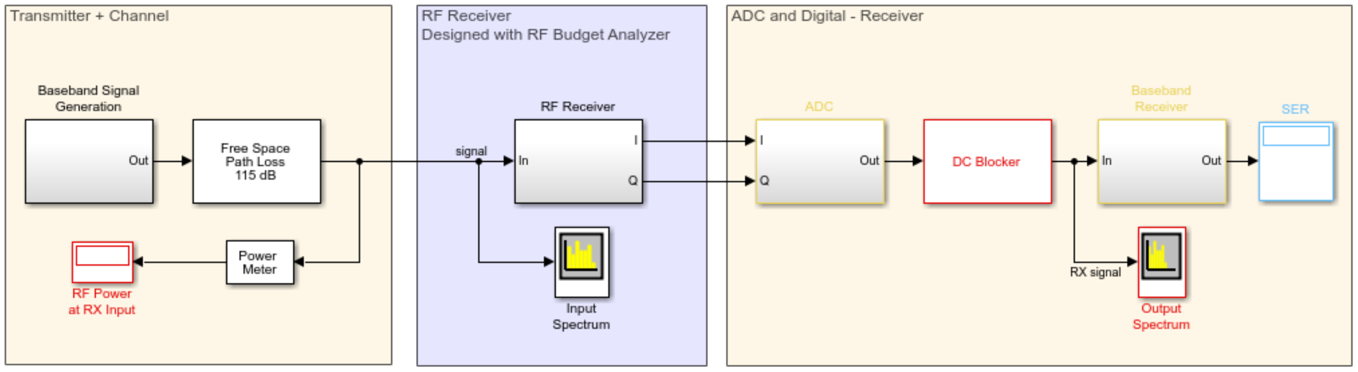
This example shows how to integrate an RF receiver together with baseband signal processing algorithms to model an end-to-end communications system.

The example requires Communications Toolbox™.

Part 1: Baseband Communications Link with Integrated RF Receiver Model

The following model includes a baseband signal generator, a simple channel, an RF receiver initially designed using the RF budget analyzer as described in Getting Started with RF Modeling, an analog-to-digital conversion, a demodulation scheme, and a computation block for the symbol error rate.

```
model = 'simrfV2_comms_rf_example';
open_system(model);
```



For this model, blocks from Communications Toolbox and DSP System Toolbox™ are used to perform baseband signal processing. The nonstandard compliant baseband signal has a rectangular QAM constellation with raised cosine filtering and the baseband receiver does not include carrier/clock synchronization. Parameters for the baseband signal generation are defined in the `Model Properties` -> `Model callbacks` **PreLoadFcn**, which sets these parameters in the MATLAB workspace when the model is loaded:

- BW = 8 MHz;
- Tstep = 125 ns; % 1/BW
- FrameLength = 128;
- M = 4; % Constellation size 2^M
- Tsymbol = 64 us; % M*FrameLength*Tstep

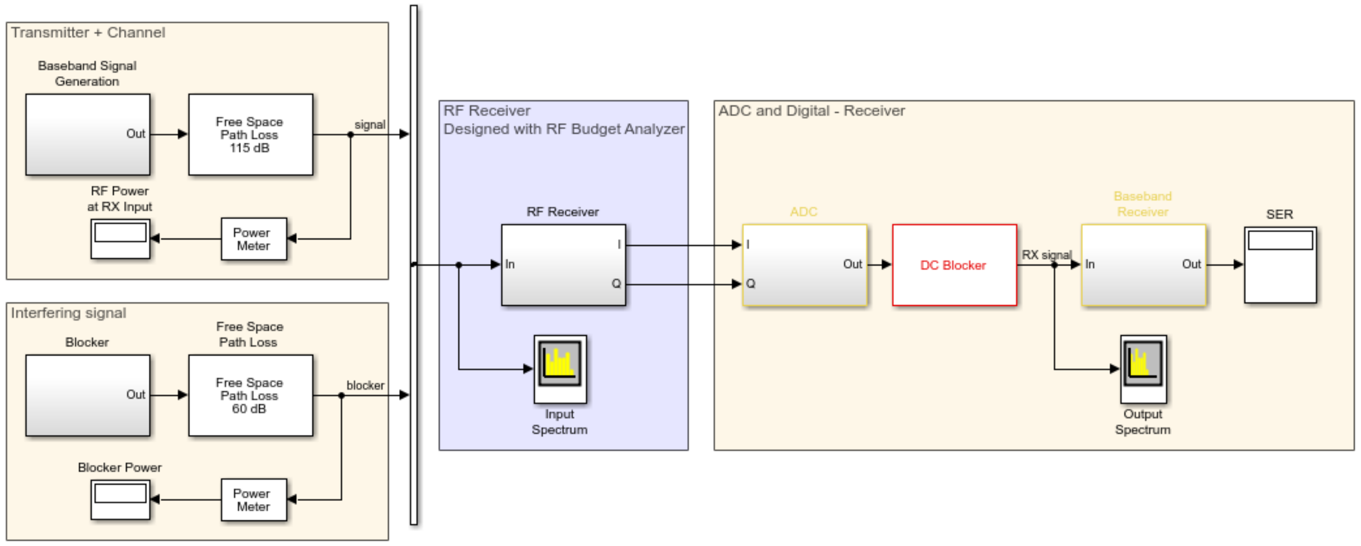
`Sample time` for the baseband signal and `Step size` of the RF Blockset receiver Configuration block have the same value. This guarantees that the RF simulation bandwidth is consistent with the sampling rate of the input signal. The RF Blockset Receiver has input and output ports that convert the Simulink signals into RF domain quantities and scale their power to a 50 Ohm reference impedance. The input port centers the baseband signal at a specified center frequency of 2.45 GHz, and the RF IQ Demodulator downconverts the input signal to baseband with a single quadrature stage.

```
bdclose(model);
```

Part 2: Include an Out-of-Band Interfering Blocker Signal

The model `simrfV2_comms_rf_interferer` shows how to add a high power out-of-band interferer centered around 2.5 GHz. This blocker affects the RF receiver by driving it into the nonlinear region. Use the following steps to complete this task.

```
model = 'simrfV2_comms_rf_interferer';
open_system(model);
```



Add an 8-PSK Modulator Baseband block source to include a blocking signal with a higher power level than the transmitter signal. Using the Vector Concatenate block, combine the baseband and blocker signals. The input signal to the RF receiver is now composed of two complex baseband signals. It is important that the two baseband sources use the same sample rate to insure equal simulation bandwidths for each signal (same envelope bandwidth). If the two signals don't have the same sample time, they need to be resampled before combining. This is the recommended best practice for simulating blocker signals when they are "far away" from the desired signal in the frequency spectrum and cannot be included in the same envelope for a particular carrier. To display the spectral positioning of the two input signals in the Spectrum Analyzer block, the `Offset` option has two frequencies specified for the two baseband signals.

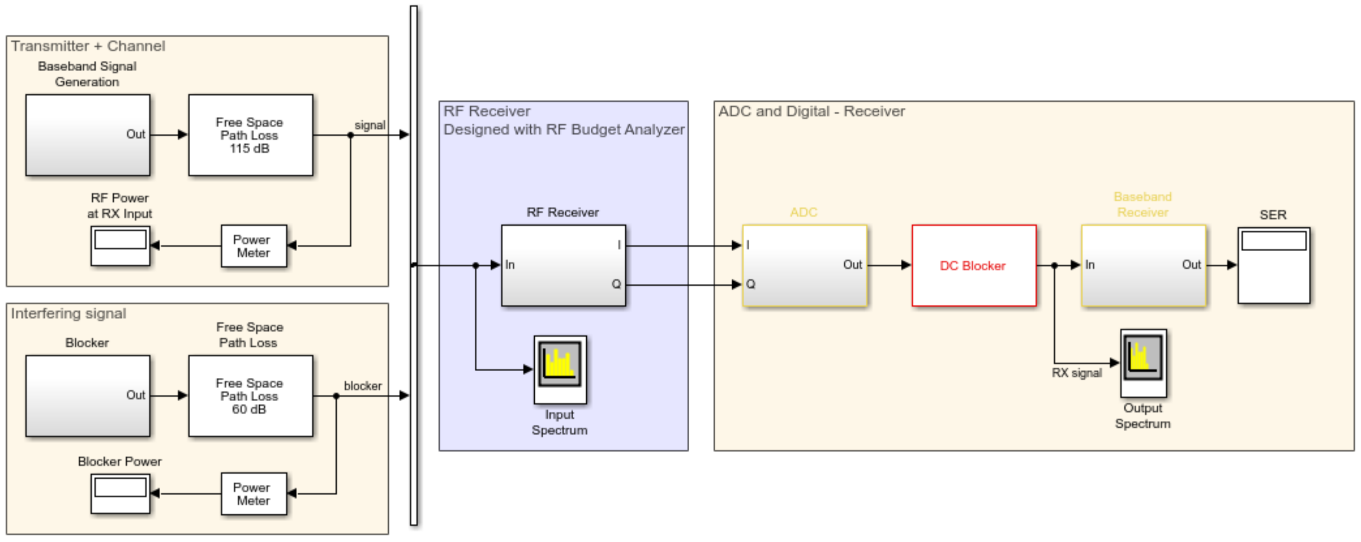
The input port of the RF receiver has been modified to include the two carrier (`Carrier frequencies`) signals (2.45 GHz and 2.5 GHz). Initially we leave the configuration block to automatically select the fundamental tones and the harmonic order.

```
bdclose(model);
```

Part 3: Add Imperfections to the RF Receiver

The model `simrfV2_comms_rf_impairments` shows how to add impairments to the RF receiver that were initially not estimated in the link budget of the RF Budget Analyzer.

```
model = 'simrfV2_comms_rf_impairments';
open_system(model);
```



Under the mask of the RF receiver, modify the RF demodulator to add imperfections that will be driven by the blocking signal. In the mask of the IQ demodulator change these parameters:

- `I/Q gain mismatch` = 0.5 dB
- `I/Q phase mismatch` = 1 degree
- `LO to RF isolation` = 85 dB
- `IIP2` = 45 dB
- `Phase noise frequency offset` = [1e5 5e5 2e6] Hz
- `Phase noise level` = [-95 -120 -140] dBc/Hz

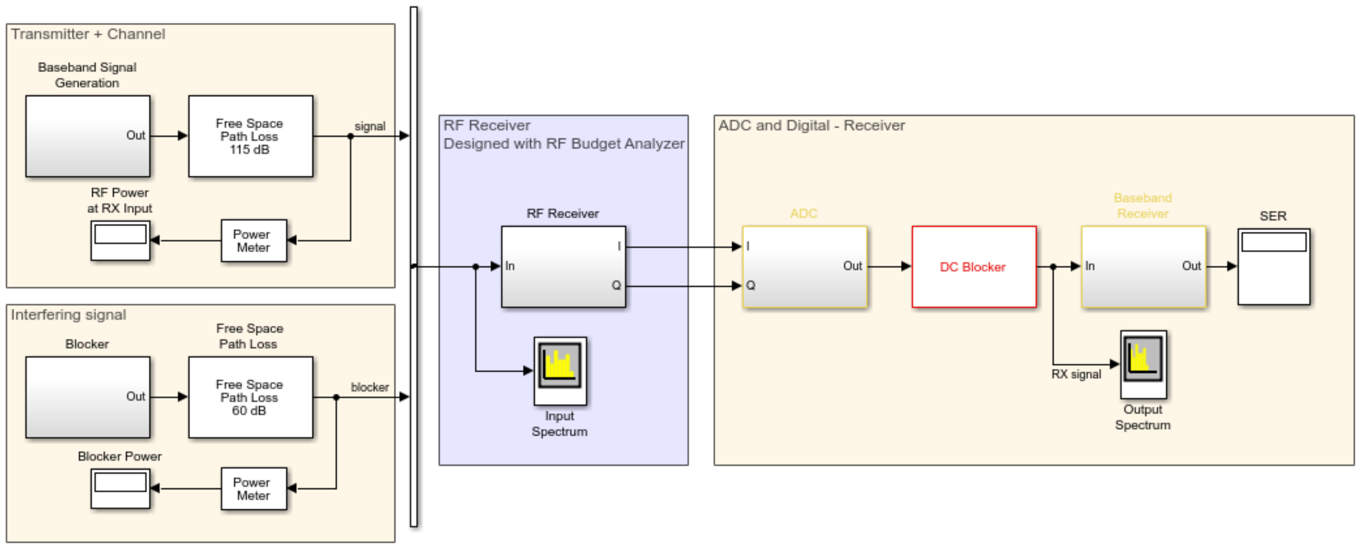
Each of these imperfections separately increases the bit error rate. These imperfections cause finite image rejection and a DC offset that is removed in the baseband domain. As observed, the DC offset correction requires time to integrate the signal power and remove the DC component. To further modify the structure of the I/Q demodulator system, you can click on the "Edit System" button. With this operation you disable the link to the library, inline the value of the parameters, and have the ability to manually modify the block parameters as well as the block architecture.

```
bdclose(model);
```

Part 4: How to Decrease Simulation Time

The model `simrfV2_comms_rf_speed` shows how to decrease the simulation time of the previous model described in this example. Follow these steps to speed up the simulation of the model.

```
model = 'simrfV2_comms_rf_speed';
open_system(model);
```



In Simulink, select `Accelerator` mode to speed up the simulation by leveraging automatic C code generation.

In the RF Blockset section, to speed up simulation reduce the `Harmonic order` of the Circuit Envelope configuration block. Uncheck `Automatically select fundamental tones and harmonic order` and set the `Harmonic order` equal to 3. The `Total simulation frequencies` is reduced from 61 to 25, equivalent to an approximate 2.5 times speed up. After reducing the Harmonic order, verify that simulation results do not change.

To further increase simulation speed, use `Frequency domain` modeling instead of `Time domain` modeling for the S-parameters SAW filter block. You need to verify that when changing the S-parameters modeling approach the simulated transfer function is still correct and that the model uses a sufficiently long `Impulse response duration`.

With the above modifications, the simulation is approximately five times faster without significantly affecting the simulation results.

```
bdclose(model);
clear model;
```

Related Topics

"Getting Started with RF Modeling" on page 9-2

Automatic Sample-Time Interpolation at Input Port

This example shows how to manage models consisting of both digital communication and RF systems that process signals at different sampling rates. To perform a model simulation where the Nyquist sampling rate of the digital communication signal is less than the inverse of the RF section time step an interpolation filter will be employed. The use of this interpolation filter diminishes the introduction of artificial signal artifacts at the boundaries of the communication and RF systems resulting from the sampling rate differences.

Part 1: Single signal entering the RF system

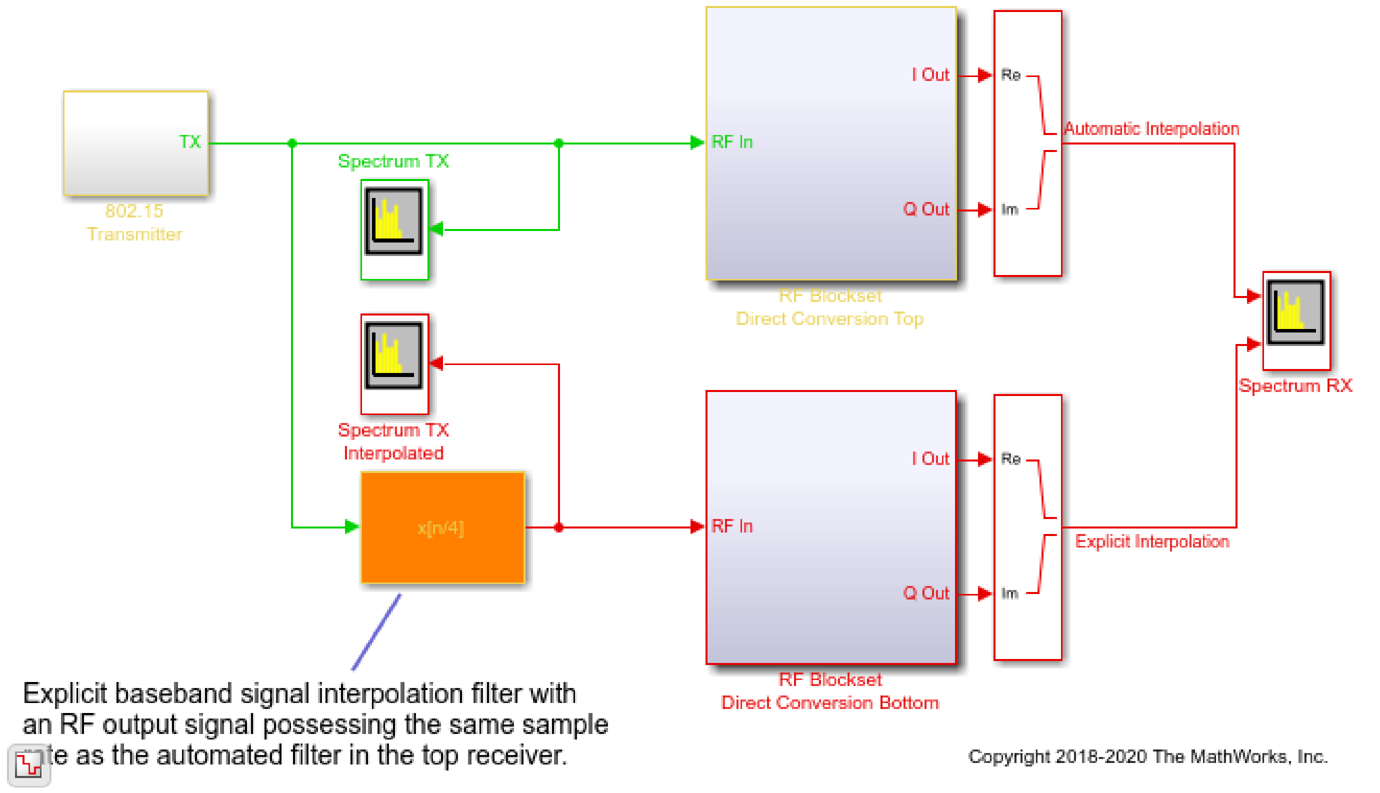
The following model includes a Zigbee (802.15) baseband signal feeding a direct conversion RF receiver. The ZigBee baseband transmitter is built using blocks from Communications Toolbox™ and DSP System Toolbox™ while the RF receiver is constructed using blocks from the RF Blockset™ Circuit Envelope library.

For the RF Blockset Circuit Envelope solver it is recommended to use a simulation time step that is 4 to 8 times smaller than the reciprocal of the input baseband signal sample time. This provides a simulation bandwidth that is sufficient for the RF solver to capture artifacts at the edge of the bandwidth accurately and the physical effects that require additional bandwidth such as spectral regrowth. In general, using an interpolation factor of 4 to 8 increases the simulation bandwidth beyond the Nyquist rate of the baseband signal generated in the transmitter.

In this model, the two different signal sample rates are:

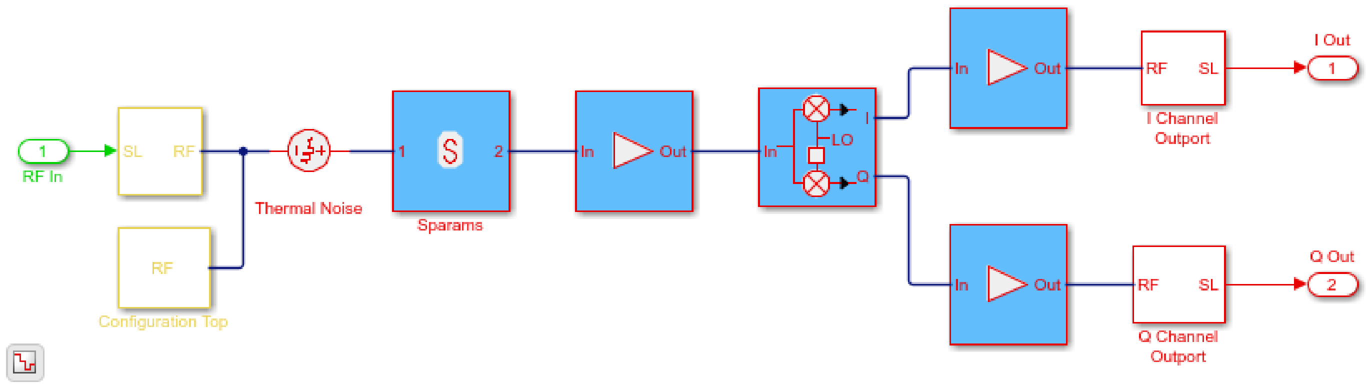
- green for the communications baseband signal
- red for the RF circuit envelope signal

```
model = 'simrfV2_sampletime_example';
open_system(model)
sim(model)

% Hide all scopes (see PostLoadFcn Model Callback for more details):
SpTxScopeConf.Visible = false;
SpTXiScopeConf.Visible = false;
SpRxScopeConf.Visible = false;
```

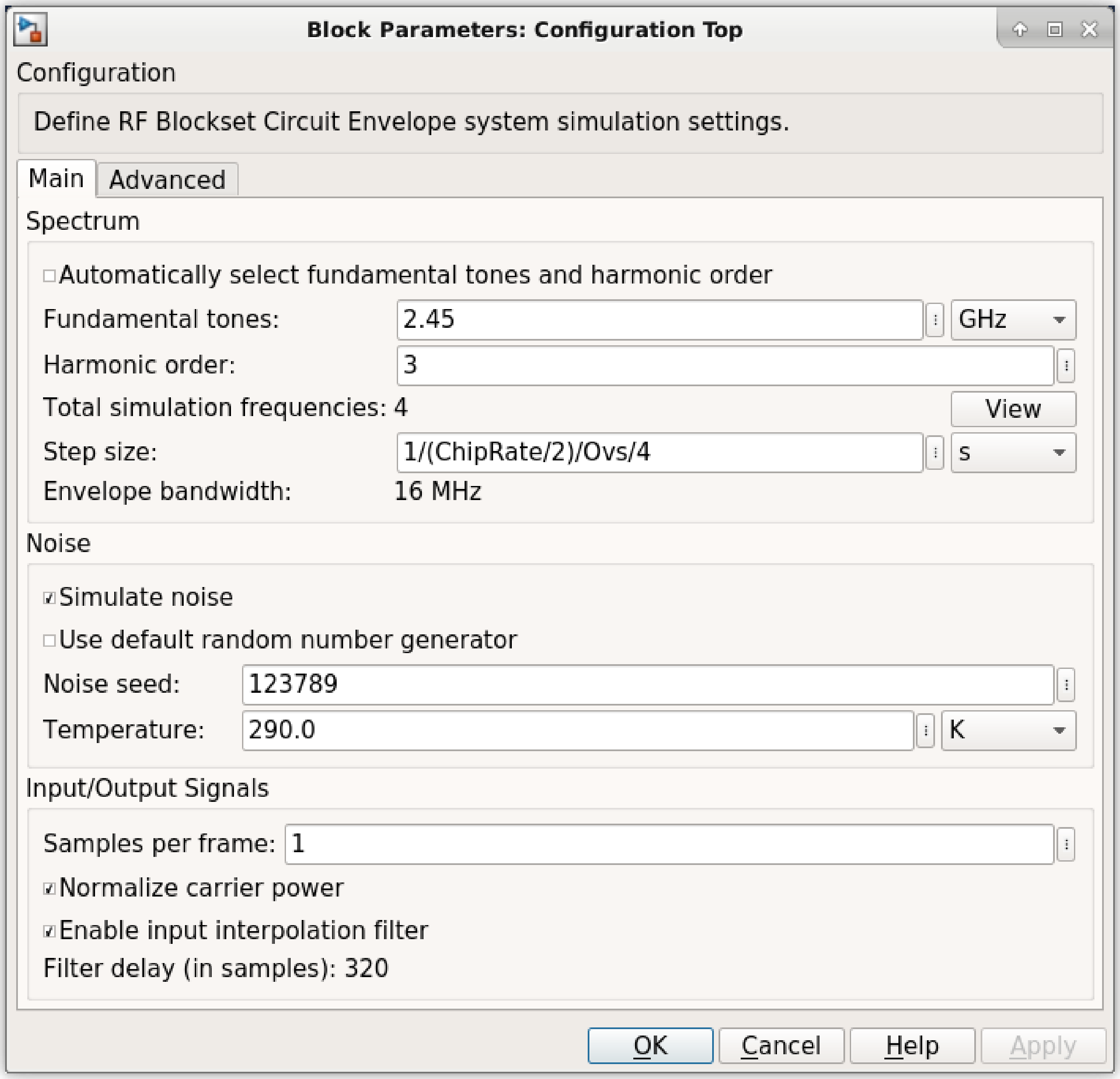

TX
802.15 Transmitter
Spectrum TX
Spectrum TX Interpolated
x[n/4]
Explicit baseband signal interpolation filter with an RF output signal possessing the same sample rate as the automated filter in the top receiver.
RF In
I Out
Q Out
RF Blockset Direct Conversion Top
RF Blockset Direct Conversion Bottom
Re
Im
Automatic Interpolation
Explicit Interpolation
Spectrum RX
Copyright 2018-2020 The MathWorks, Inc.

The Top and Bottom RF receiver systems in the model are identical and consist of Pre-LNA filter, followed by an LNA, quadrature demodulator, and another amplification stage. All RF components include typical impairments such as noise, nonlinearity and finite isolation.

```
open_system([model '/RF Blockset Direct Conversion Top'])
```



As specified in the Configuration block Mask Parameters dialog box, the simulation is performed with the input interpolation filter enabled for the top receiver,



Block Parameters: Configuration Top

Configuration

Define RF Blockset Circuit Envelope system simulation settings.

Spectrum

☐ Automatically select fundamental tones and harmonic order

Fundamental tones:	2.45	GHz
Harmonic order:	3	
Total simulation frequencies:	4	View
Step size:	1/(ChipRate/2)/Ovs/4	s
Envelope bandwidth:	16 MHz	

Noise

☑ Simulate noise

☐ Use default random number generator

Noise seed:	123789	
Temperature:	290.0	K

Input/Output Signals

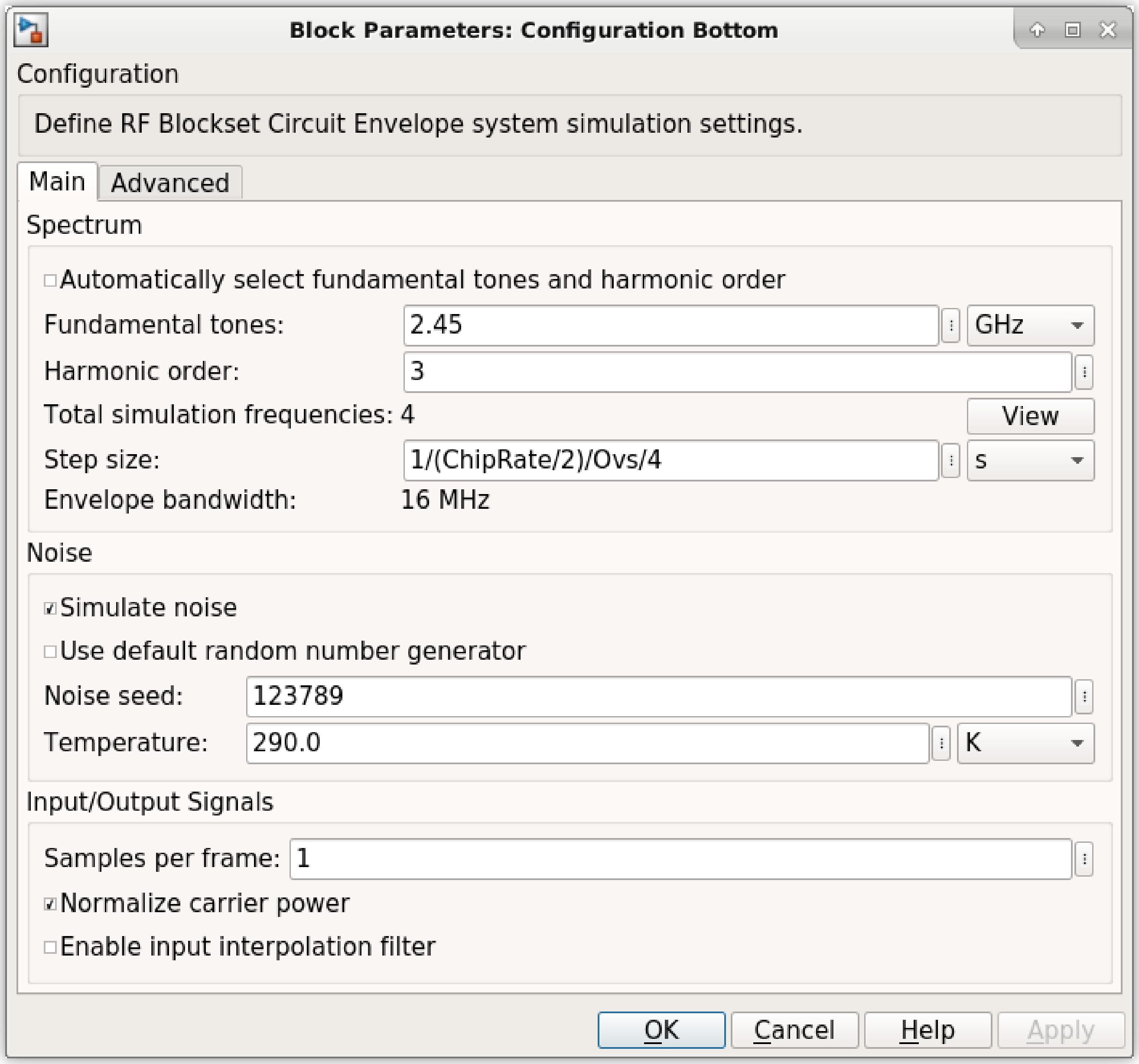
Samples per frame: 1

☑ Normalize carrier power

☑ Enable input interpolation filter

Filter delay (in samples): 320

and disabled for the bottom receiver.


Block Parameters: Configuration Bottom

Configuration

Define RF Blockset Circuit Envelope system simulation settings.

Main | Advanced

Spectrum

☐ Automatically select fundamental tones and harmonic order

Fundamental tones: 2.45 GHz

Harmonic order: 3

Total simulation frequencies: 4 View

Step size: 1/(ChipRate/2)/Ovs/4 s

Envelope bandwidth: 16 MHz

Noise

☑ Simulate noise

☐ Use default random number generator

Noise seed: 123789

Temperature: 290.0 K

Input/Output Signals

Samples per frame: 1

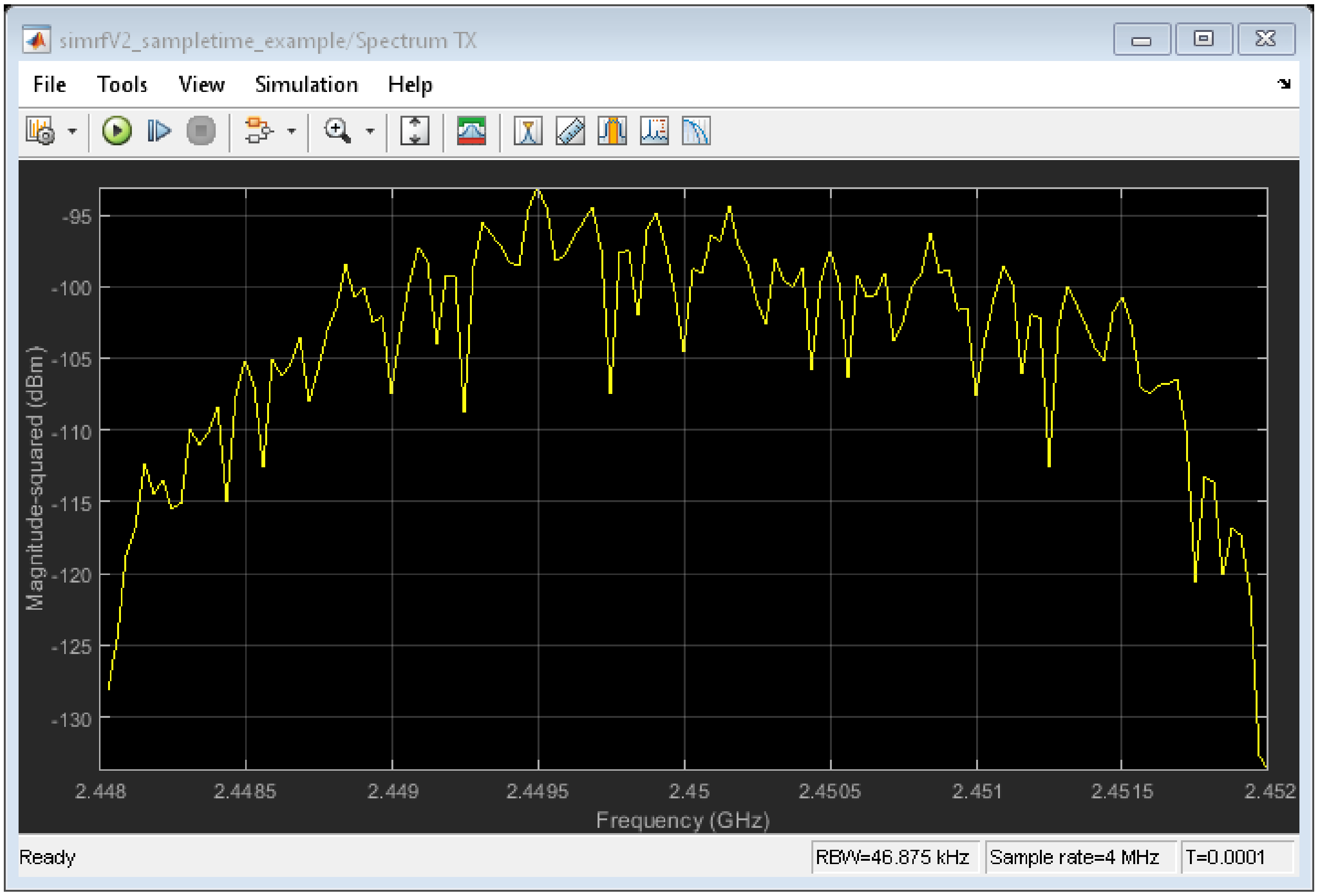
☑ Normalize carrier power

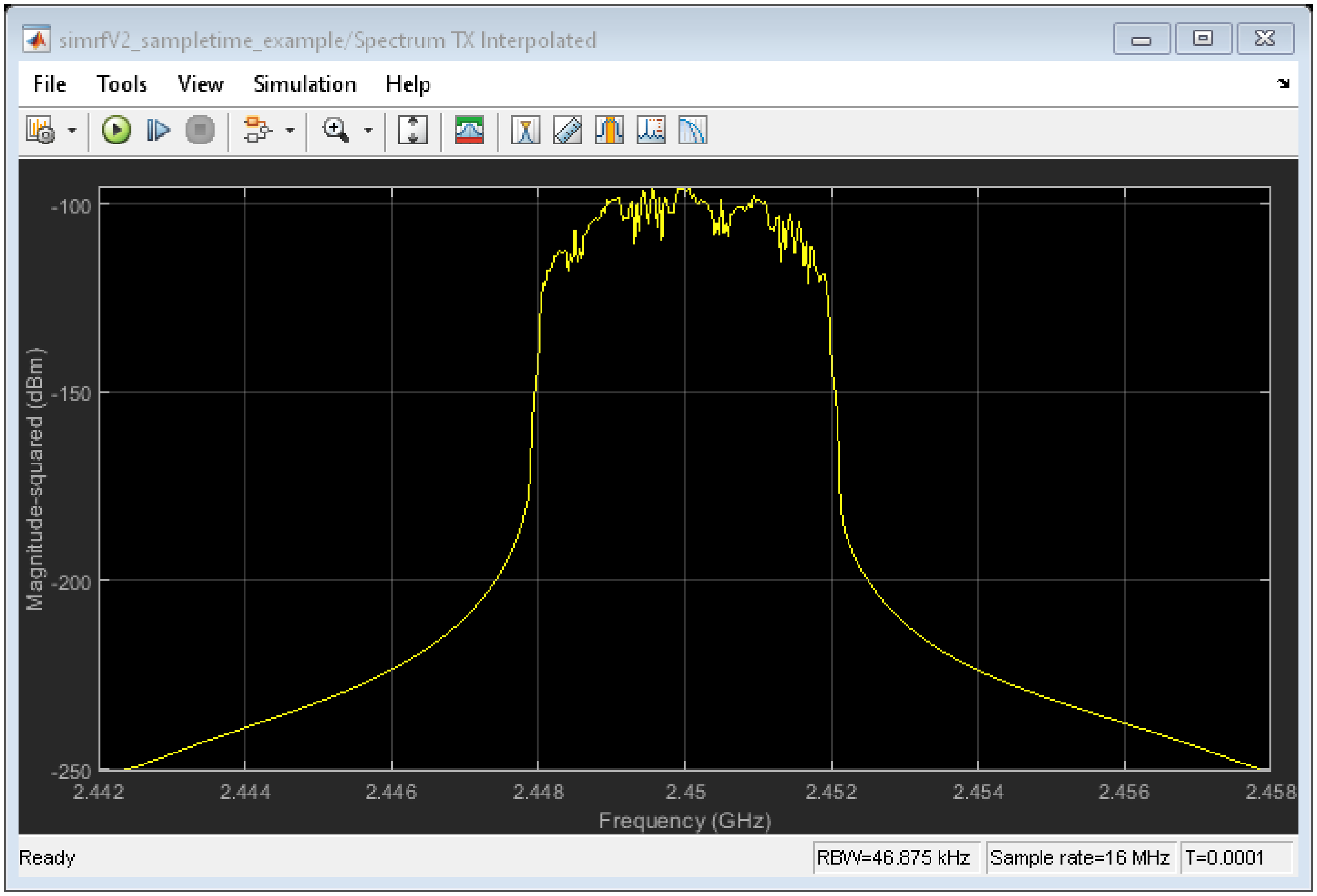
☐ Enable input interpolation filter

The top RF receiver is fed with a baseband signal possessing a sample rate 4 times slower than the reciprocal of the RF simulation step size set in its Configuration block. The RF Inport block automatically interpolates the input signal at the required RF rate.

The bottom RF receiver is fed with a baseband signal sample rate equaled to the reciprocal of the step size specified in its RF Configuration block. The bottom RF receiver uses an explicit interpolation filter highlighted in orange to up-sample the communication baseband signal.

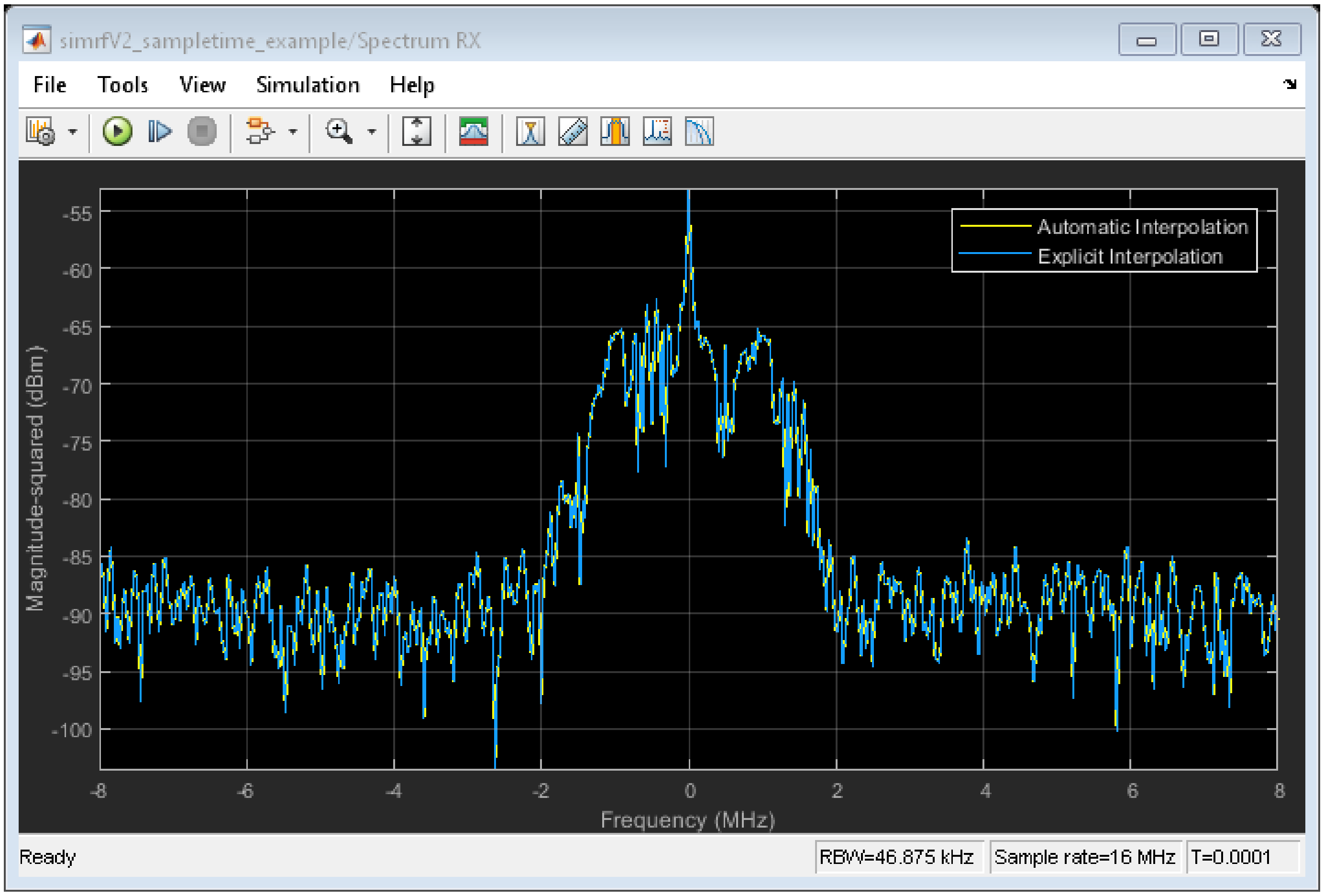
```
% Show these two scope results:
SpTxScopeConf.Visible = true;
SpTXiScopeConf.Visible = true;
```


simrfV2_sampletime_example/Spectrum TX
File
Tools
View
Simulation
Help
-95
-100
-105
-110
-115
-120
-125
-130
Magnitude-squared (dBm)
2.448
2.4485
2.449
2.4495
2.45
2.4505
2.451
2.4515
2.452
Frequency (GHz)
Ready
RBW=46.875 kHz
Sample rate=4 MHz
T=0.0001



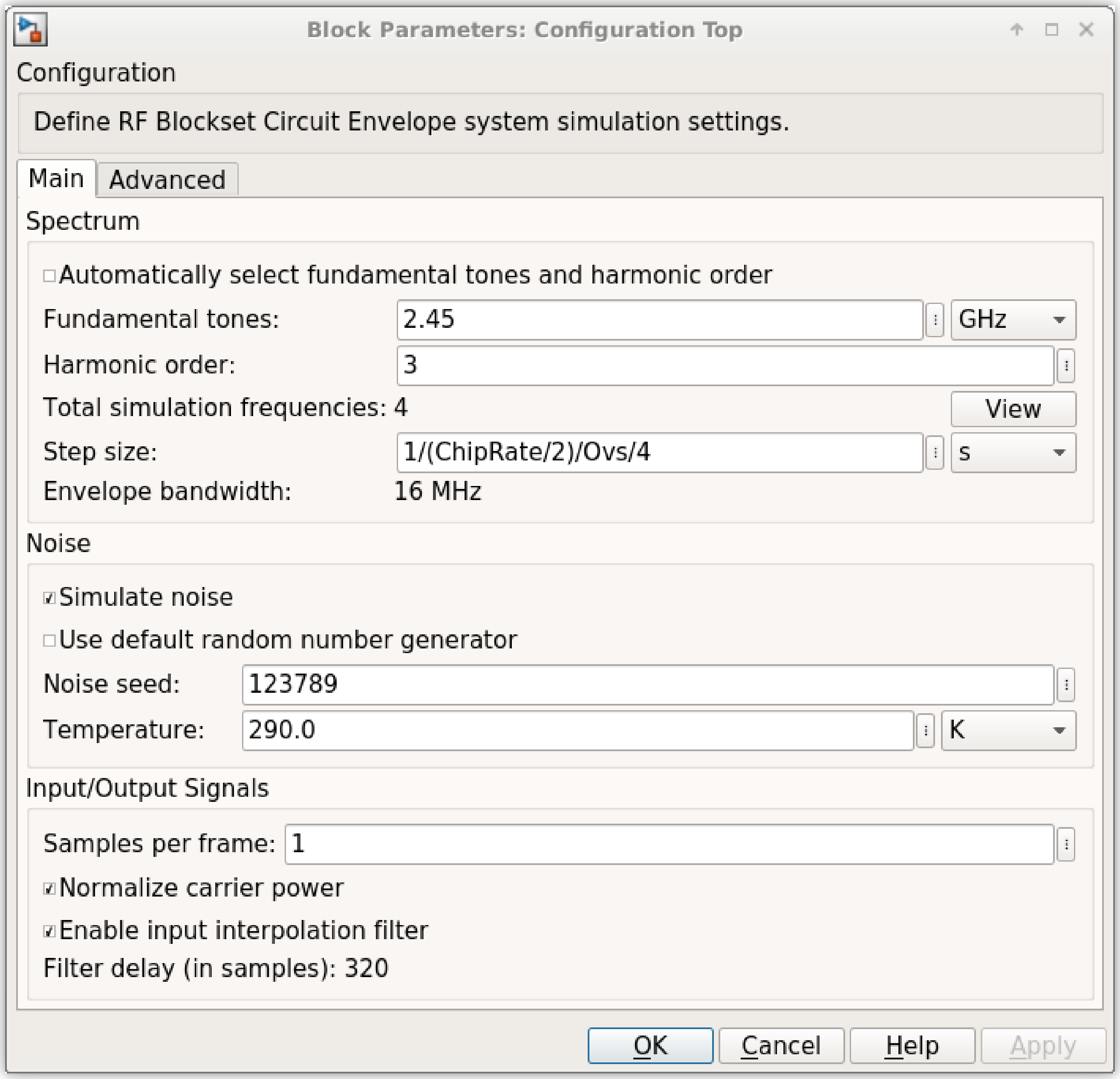
The outputs of both receivers are the same, since both input signals are resampled by interpolation filters to reduce sample rate transition aliasing effects. In the top receiver, the sample rate transition is automatically managed by the circuit envelope Inport block. In the bottom receiver, the sample rate transition is explicitly managed by the addition of the interpolation filter.

```
% Show this scope result:
SpRxScopeConf.Visible = true;
```



Using an interpolation filter improves the spectral results of the simulation, but comes at a price: it introduces a delay. Since an FIR filter is used for the interpolation, the delay corresponds to half the number of filter coefficients. In this case, the filter has 640 taps and introduces a delay of 320 time steps at the faster RF sample rate or 80 time steps at the slower baseband communication sample rate. In case of multiple baseband communication signal inputs, it may be necessary to compensate for the delay by aligning all signals entering the RF system.

When an input interpolation filter is enabled in the Configuration block Mask Parameters dialog box, the RF signal delay introduced will be displayed next to the enabling switch.


Block Parameters: Configuration Top

Configuration

Define RF Blockset Circuit Envelope system simulation settings.

Main | Advanced

Spectrum

☐ Automatically select fundamental tones and harmonic order

Fundamental tones: 2.45 GHz

Harmonic order: 3

Total simulation frequencies: 4

Step size: 1/(ChipRate/2)/Ovs/4 s

Envelope bandwidth: 16 MHz

Noise

☑ Simulate noise

☐ Use default random number generator

Noise seed: 123789

Temperature: 290.0 K

Input/Output Signals

Samples per frame: 1

☑ Normalize carrier power

☑ Enable input interpolation filter

Filter delay (in samples): 320

By default, RF Blockset automatically inserts an interpolation filter and resamples the input signal. You might decide to disable the default option and explicitly insert an interpolation filter if you have:

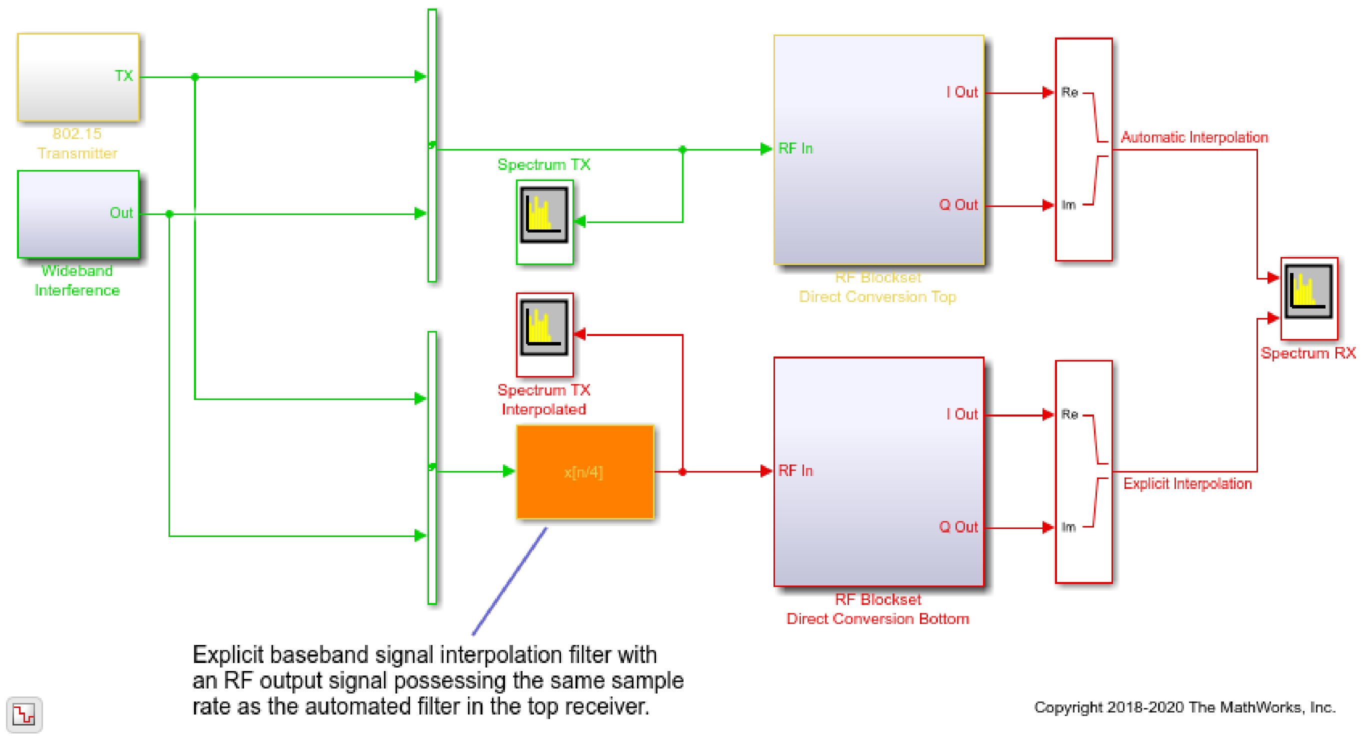
- specific requirements regarding the specifications of the interpolation filter;
- multiple input signals requiring different input ports (case described below);
- Simulink control signals (e.g. applied to VGA, variable phase shifter or switch blocks) that are intrinsically slower than the RF signal and do not necessitate resampling.

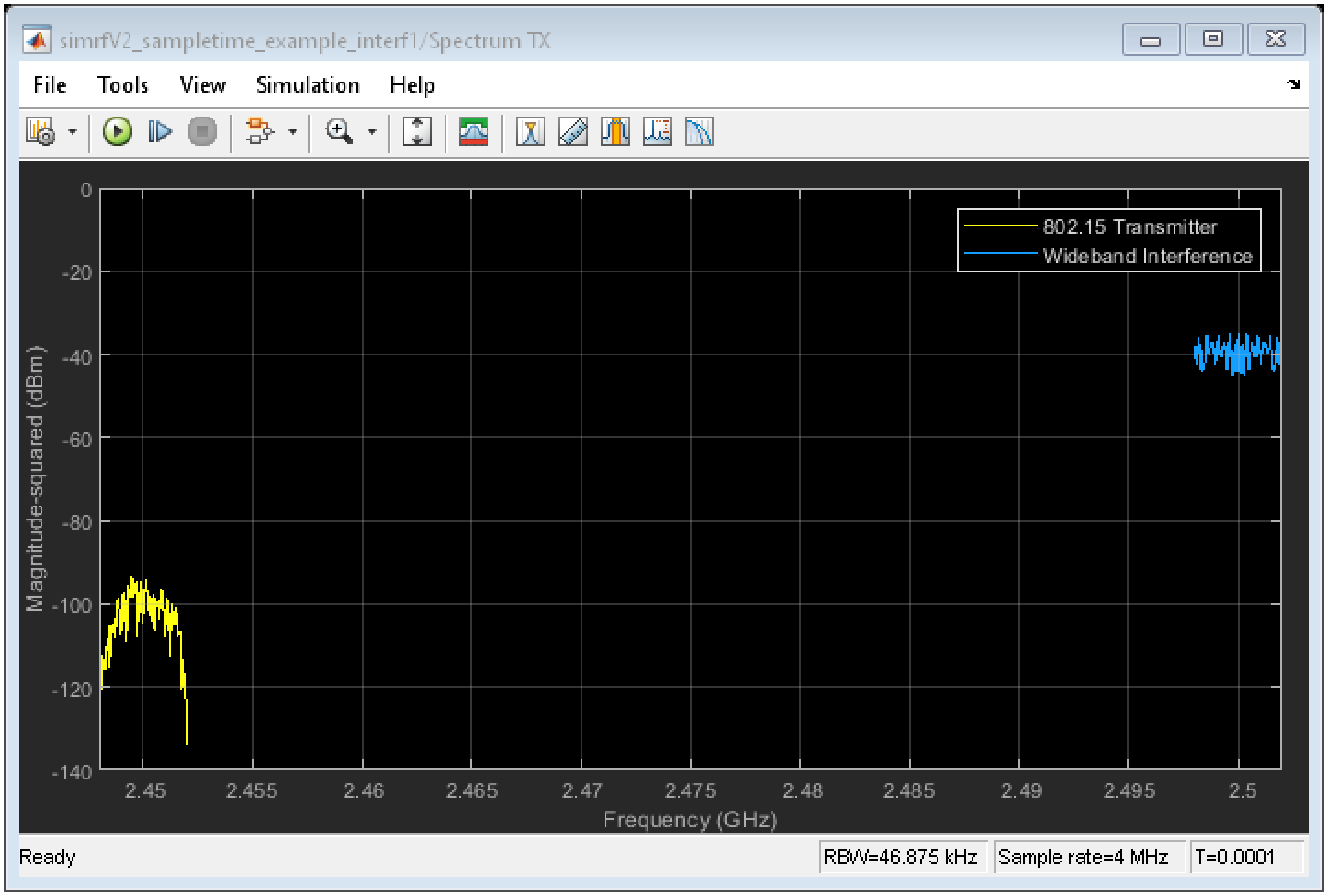
Part 2: Multiple signals entering the RF system

The automatic interpolation option discussed above can only support a single RF Inport block. When using multiple Inport blocks, the user is required to manually insert interpolation filters before these blocks. The interpolation filters are then adjusted to have all entering communication signals resampled at the rate specified in the RF Configuration block.

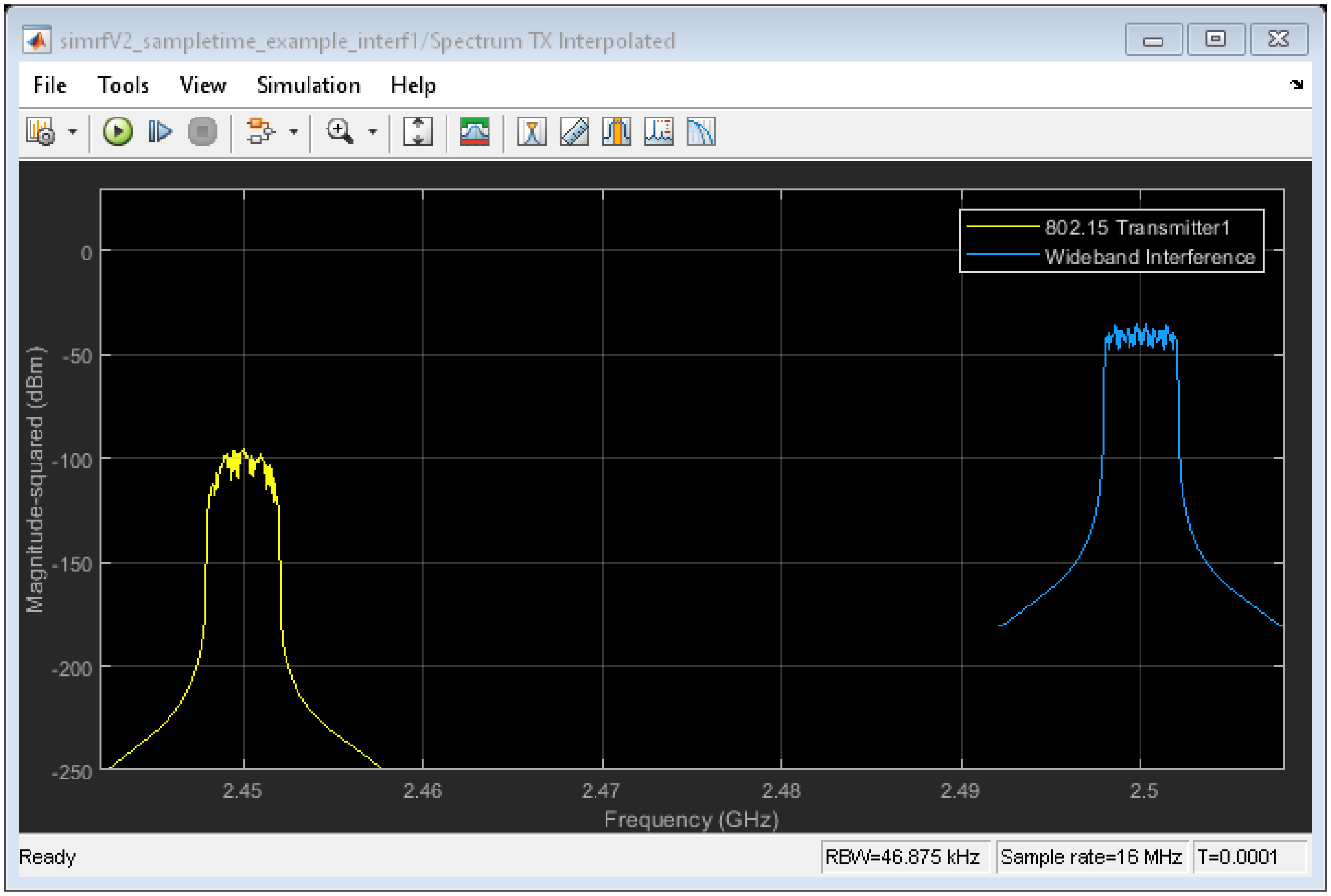
While the RF Blockset Inport block can accept a vector of multiple signals each specified at a different carrier frequency, these signals must have the same sample rates. The following model describes two RF systems with multiple inputs centered on different carriers and correctly resampled. The model is like the one in Part 1 of this example, but also includes a wideband interfering signal that is generated using blocks from Communications Toolbox and DSP System Toolbox. The two input signals have the same sample rate and the RF Blockset Configuration block has a Step size that samples the RF signal 4 times faster than the baseband communications signal.

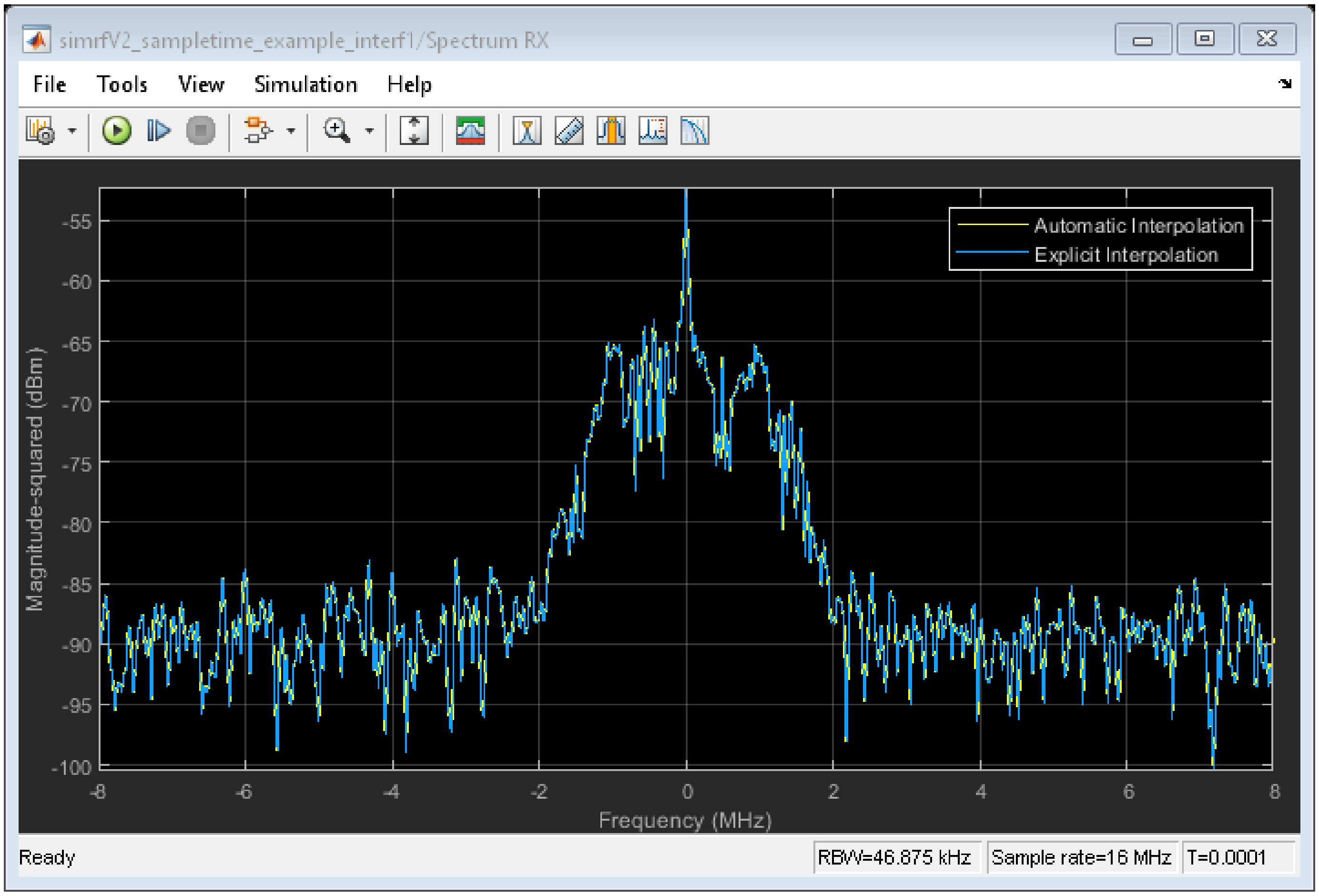
```
bdclose(model);
model = 'simrfV2_sampletime_example_interf1';
open_system(model);
sim(model);
```


TX
802.15 Transmitter
Out
Wideband Interference
Spectrum TX
Spectrum TX Interpolated
x[n/4]
RF In
I Out
Q Out
RF Blockset Direct Conversion Top
RF Blockset Direct Conversion Bottom
Re
Im
Automatic Interpolation
Explicit Interpolation
Spectrum RX
Explicit baseband signal interpolation filter with an RF output signal possessing the same sample rate as the automated filter in the top receiver.
Copyright 2018-2020 The MathWorks, Inc.



Ready RBW=46.875 kHz Sample rate=4 MHz T=0.0001


simrfV2_sampletime_example_interf1/Spectrum TX Interpolated
File
Tools
View
Simulation
Help
802.15 Transmitter1
Wideband Interference
Magnitude-squared (dBm)
0
-50
-100
-150
-200
-250
2.45
2.46
2.47
2.48
2.49
2.5
Frequency (GHz)
Ready
RBW=46.875 kHz
Sample rate=16 MHz
T=0.0001

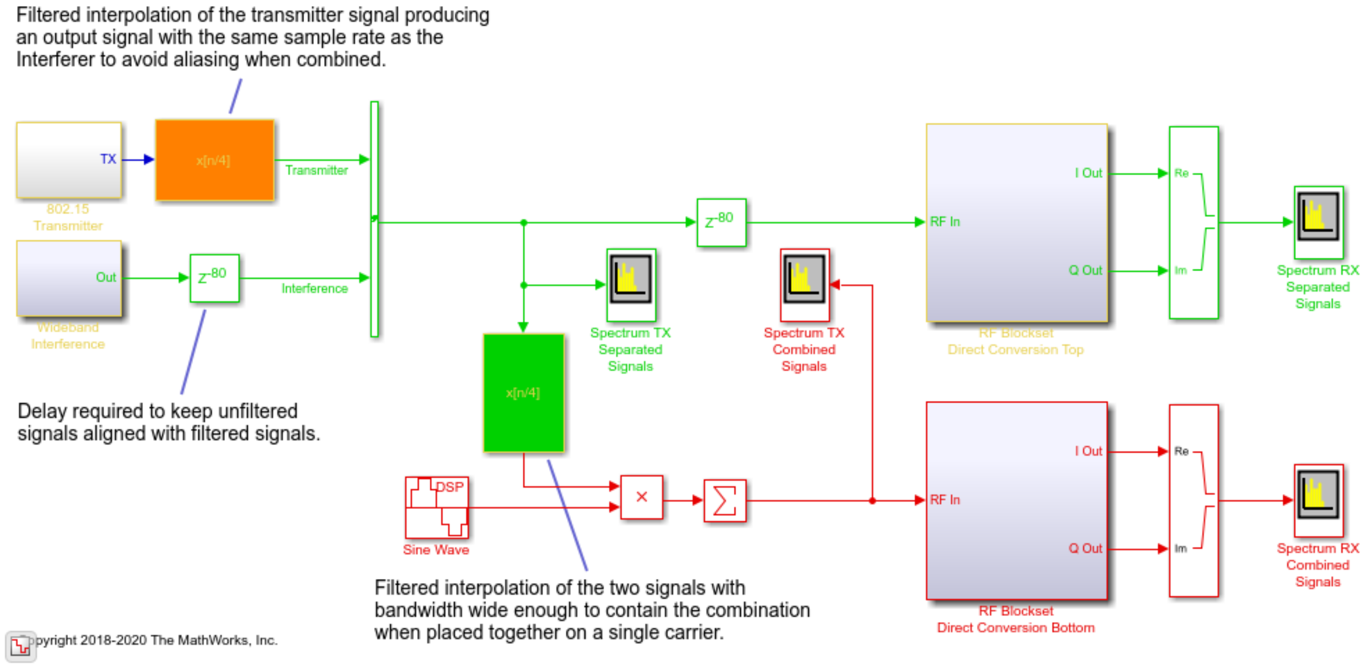


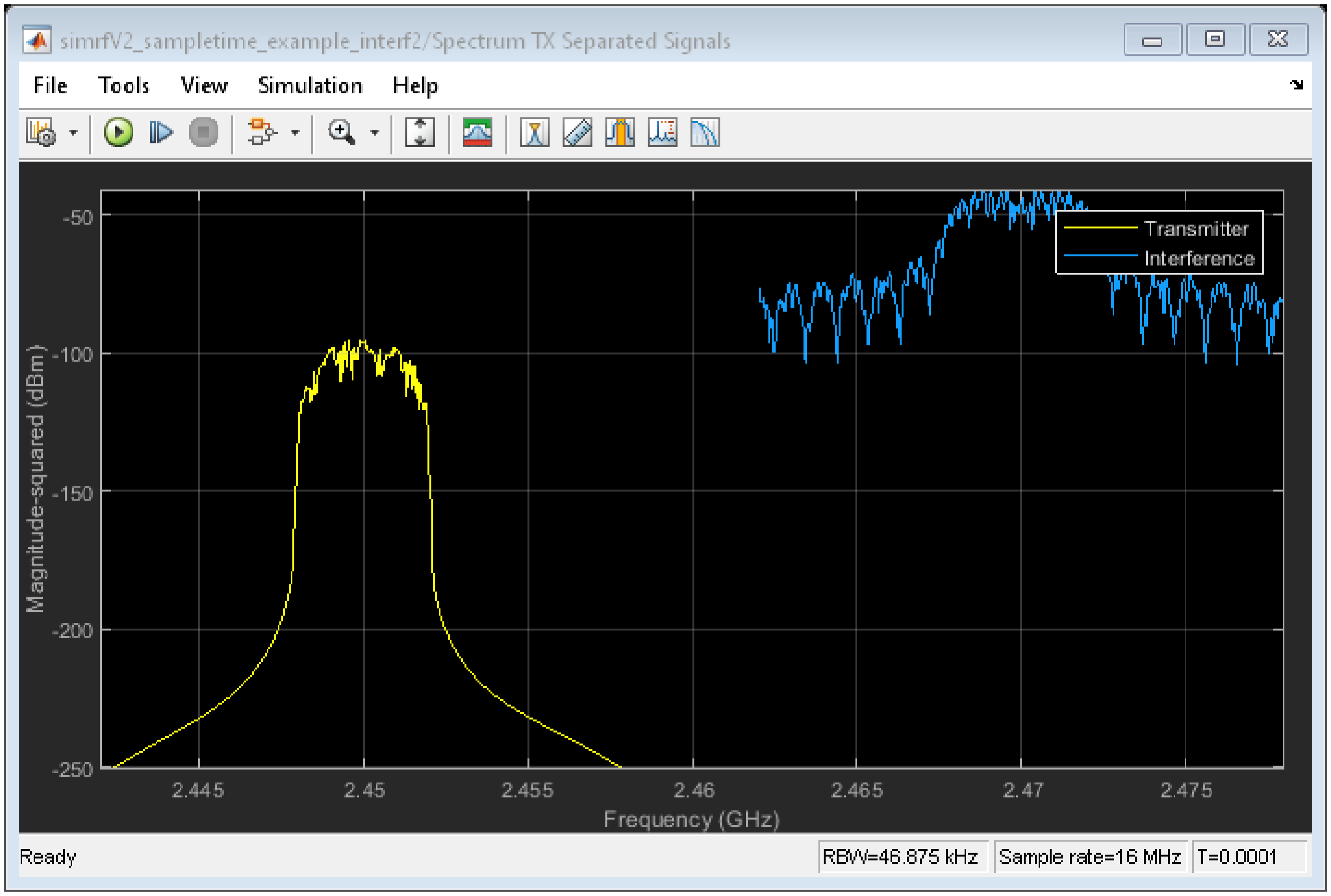
The model is like the one described in Part 1 of this example. The interpolation filter is necessary to avoid aliasing effects due to rate transition.

A more interesting scenario occurs in the following model when the desired and interferer signals have different sample rates. In this model, the desired signal is explicitly interpolated by the filter (highlighted in orange) and then combined with the wideband interferer as a vector.

To avoid the aliasing effect, the slower rate of the desired input signal is interpolated and filtered before combining with the faster rate interfering signal.

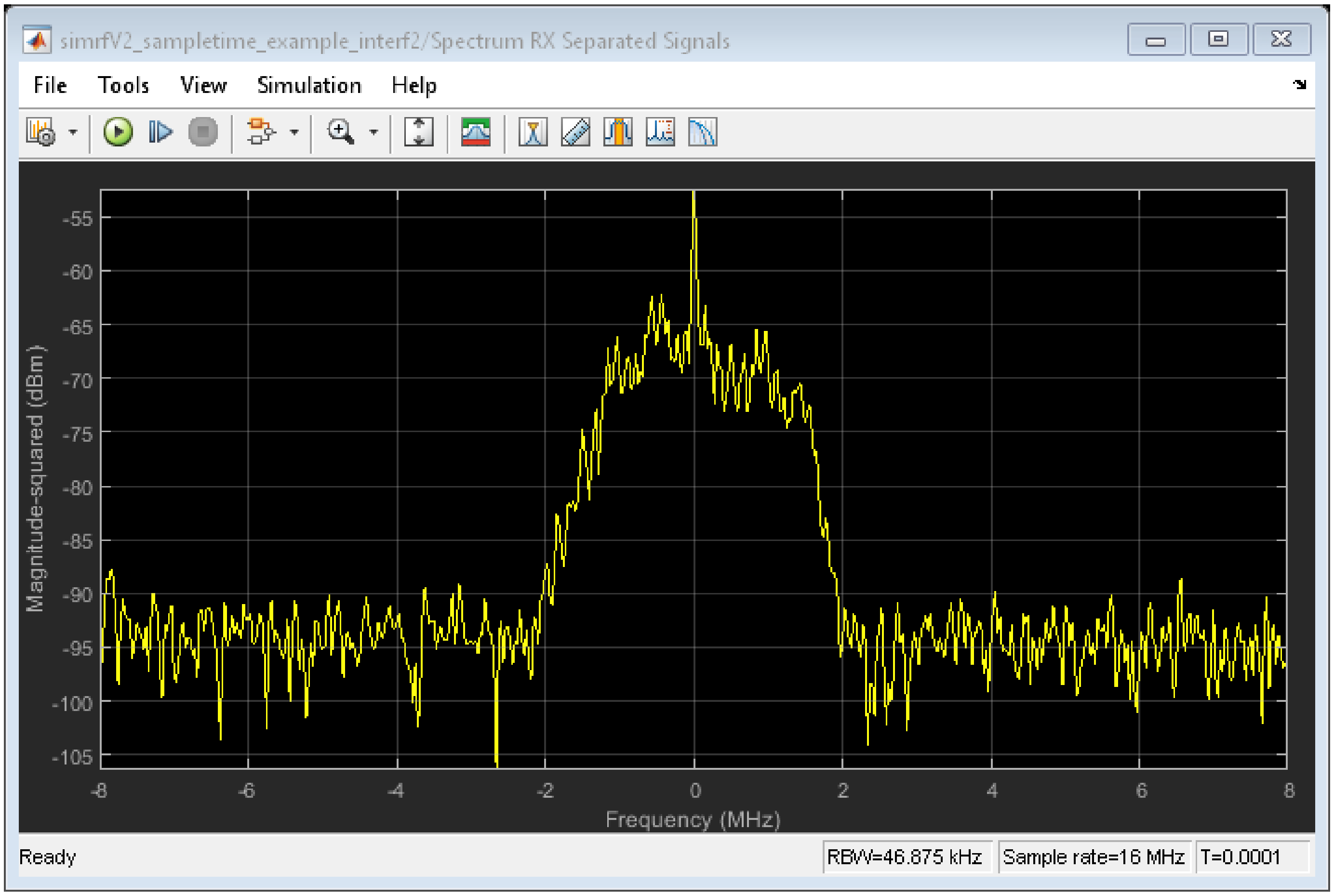
```
bdclose(model);
model = 'simrfV2_sampletime_example_interf2';
open_system(model);
sim(model);

% Hide all scope results (see PostLoadFcn Model Callback for more details):
SpTXComScopeConf.Visible = false;
SpRxSepScopeConf.Visible = false;
SpRxComScopeConf.Visible = false;
```


Filtered interpolation of the transmitter signal producing an output signal with the same sample rate as the Interferer to avoid aliasing when combined.
TX
802.15 Transmitter
x[n/4]
Transmitter
Out
Wideband Interference
z^{-80}
Interference
Delay required to keep unfiltered signals aligned with filtered signals.
Spectrum TX Separated Signals
x[n/4]
DSP
Sine Wave
Filtered interpolation of the two signals with bandwidth wide enough to contain the combination when placed together on a single carrier.
z^{-80}
Spectrum TX Combined Signals
RF In
I Out
Q Out
RF Blockset Direct Conversion Top
Re
Im
Spectrum RX Separated Signals
RF In
I Out
Q Out
RF Blockset Direct Conversion Bottom
Re
Im
Spectrum RX Combined Signals
Copyright 2018-2020 The MathWorks, Inc.


simrfV2_sampletime_example_interf2/Spectrum TX Separated Signals
File
Tools
View
Simulation
Help
Transmitter
Interference
-50
-100
-150
-200
-250
Magnitude-squared (dBm)
2.445
2.45
2.455
2.46
2.465
2.47
2.475
Frequency (GHz)
Ready
RBW=46.875 kHz
Sample rate=16 MHz
T=0.0001

In the top RF receiver, the two signals entering the RF system are centered on different carriers. Note that the sample rate of the signal entering the top RF system is the same as defined in the RF Configuration block. In this case, enabling the automatic input interpolation filter in the RF Configuration block does not introduce any interpolation.

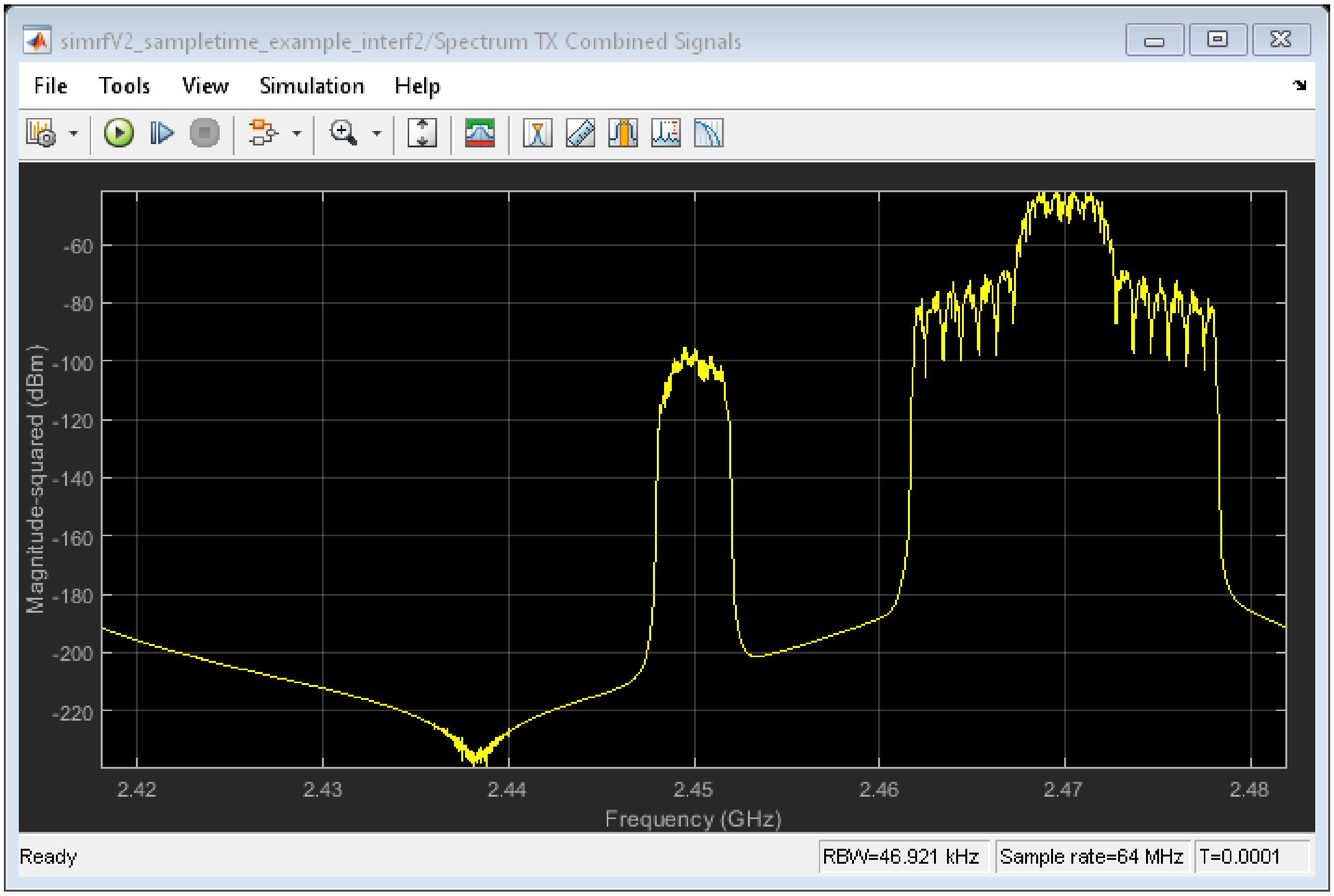
```
SpRxSepScopeConf.Visible = true;
```



RBW=46.875 kHz | Sample rate=16 MHz | T=0.0001

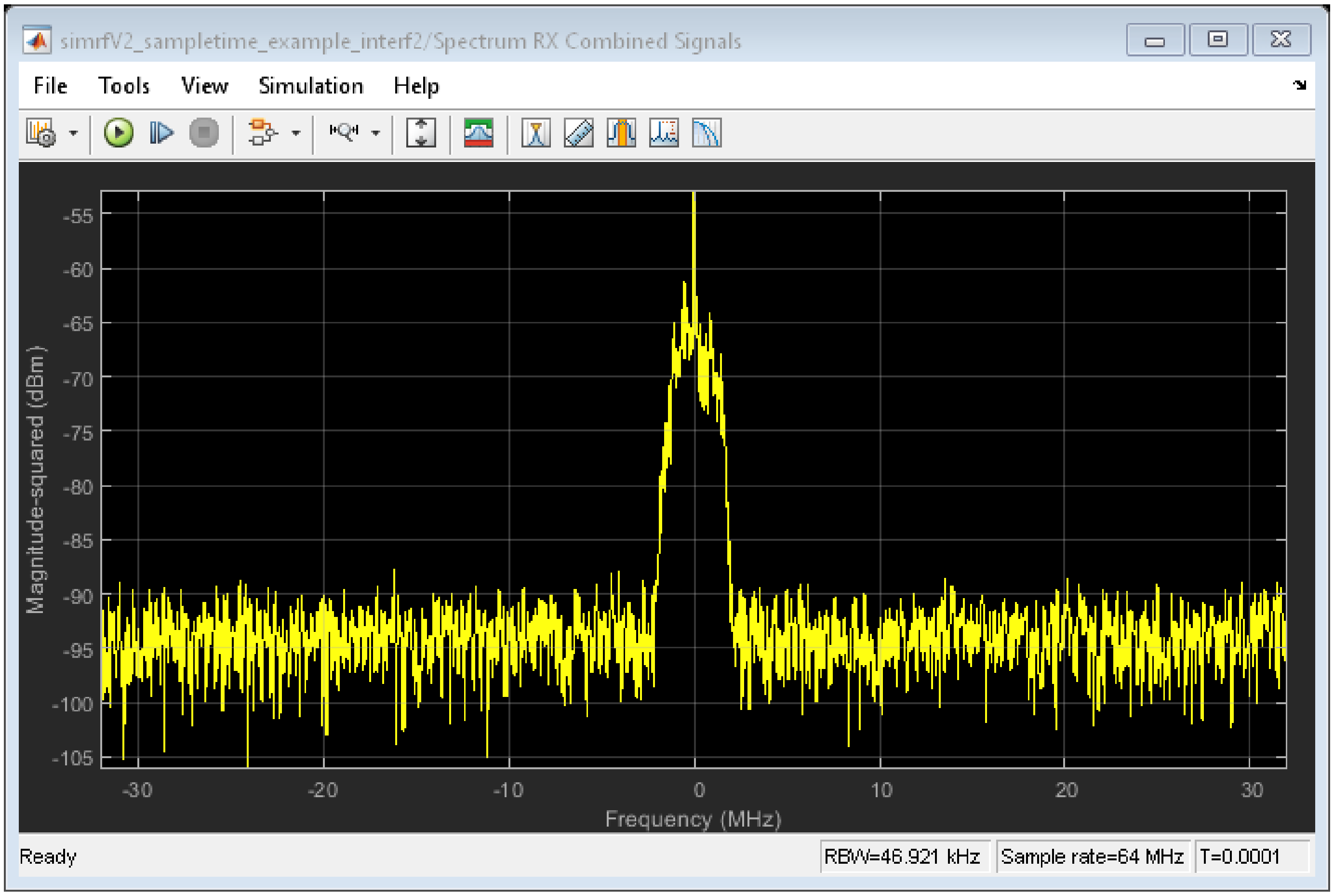
The last scenario discussed occurs when the two signals entering the RF system are placed on carriers that are relatively close to each other. Since the number of mixing harmonics required for simulation can be large in strongly nonlinear systems, it is recommended to combine the two signals onto one carrier when they are close by.

```
SpTXComScopeConf.Visible = true;
```


simrfV2_sampletime_example_interf2/Spectrum TX Combined Signals
File Tools View Simulation Help
Magnitude-squared (dBm)
Frequency (GHz)
Ready
RBW=46.921 kHz
Sample rate=64 MHz
T=0.0001

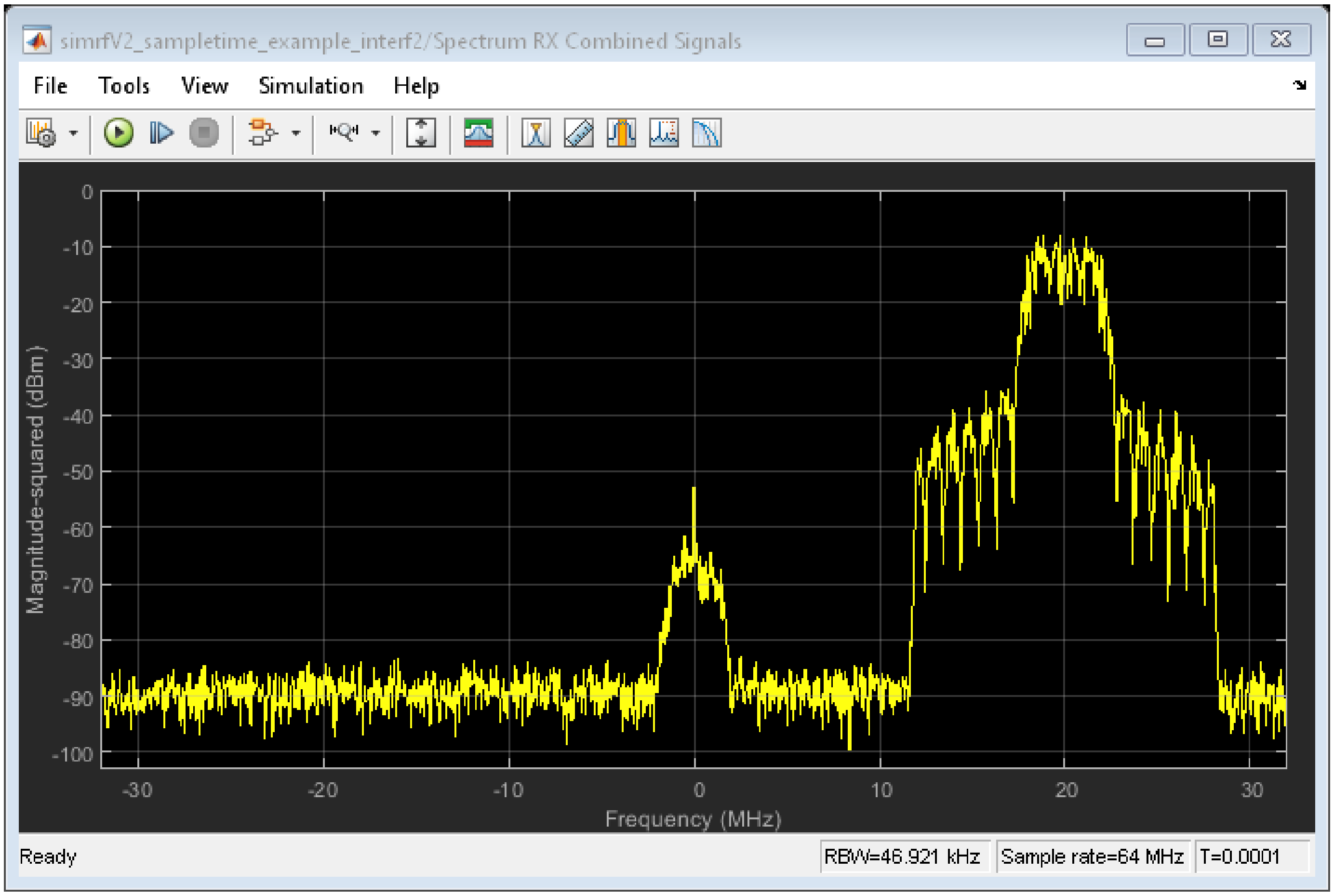
In the bottom receiver, the RF system is fed with the desired signals combined onto a single carrier signal. The combined signal is achieved by multiplying the interfering signal with a complex exponent to shift its operation frequency by 20MHz relative to the frequency of the desired signal. Note that the bandwidth needed to capture both signals when combined on a single carrier is larger than the bandwidth of each individual carrier signal. This is the reason for introducing the interpolation filter highlighted in green before combining the signals.

```
SpRxComScopeConf.Visible = true;
```



The results of the two RF systems (top and bottom) in the above model show excellent correspondence. The interferer signal is missing from the spectrum in the top RF system since the output port behaves as an ideal filter and only selects the real passband signal centered at DC. The interferer signal is missing from the spectrum in the bottom RF system since the IQ Demodulator includes a channel select filter. To see the effects of the interferer signal, turn off the filter by unchecking the 'Add Channel Select filter' checkbox in the IQ Demodulator block Mask Parameter dialog. The resulting spectrum is

```
set_param([model '/RF Blockset Direct Conversion Bottom/IQ Demodulator'], ...
    'AddCSFilters', 'off');
sim(model);

% Do not show other scopes and rescale Y axis:
SpTxSepScopeConf.Visible = false;
SpTXComScopeConf.Visible = false;
SpRxSepScopeConf.Visible = false;
SpRxComScopeConf.YLimits = [-103 0];
```



```
bdclose(model);
clear model;
```

See Also

Amplifier | Configuration

Related Topics

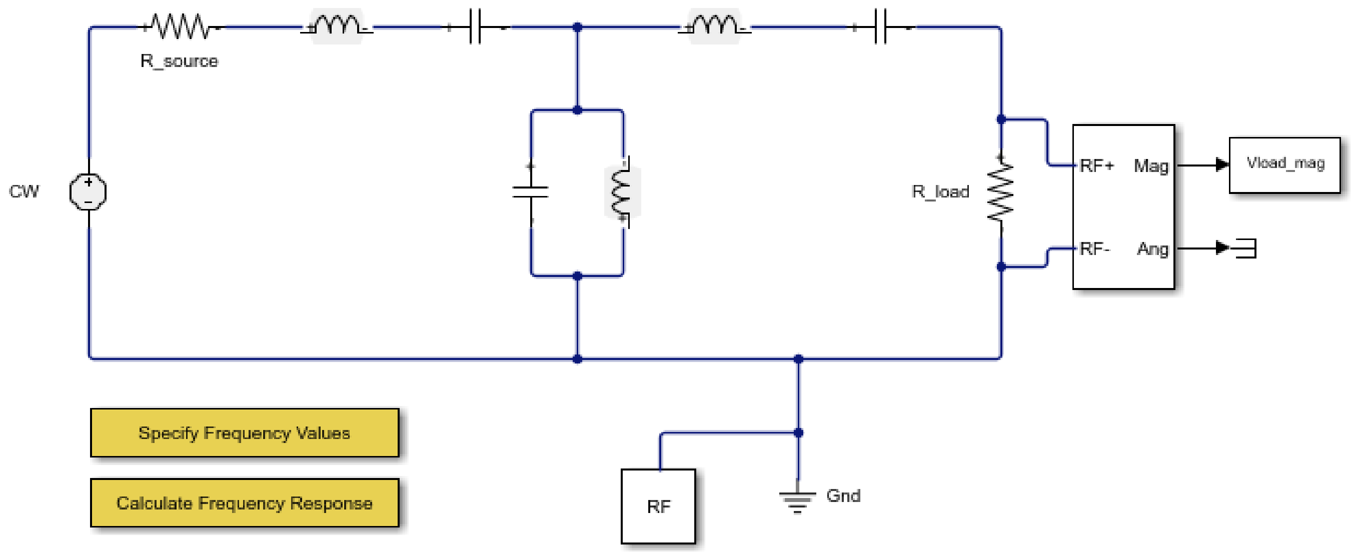
"Power Ports and Signal Power Measurement in RF Blockset" on page 9-11

Analysis of Frequency Response of RF System

This example uses a few techniques to calculate the steady-state frequency response for a filter-based RF system built from RF Blockset™ Circuit Envelope library blocks. The first technique performs static analysis (harmonic balance) on a circuit comprising of inductors and capacitors. The second technique does time domain simulation using a similar circuit built with the Filter library block. The third technique facilitates small-signal analysis to obtain the frequency response of a filtering system that exhibits nonlinearity at a given operation point. This example helps you validate a circuit envelope model using a static analysis in the frequency domain, a time domain simulation, and small signal analysis in cases where the system exhibits non-linearity.

Frequency Domain Analysis

```
model = 'simrfV2_ac_analysis';
open_system(model);
```



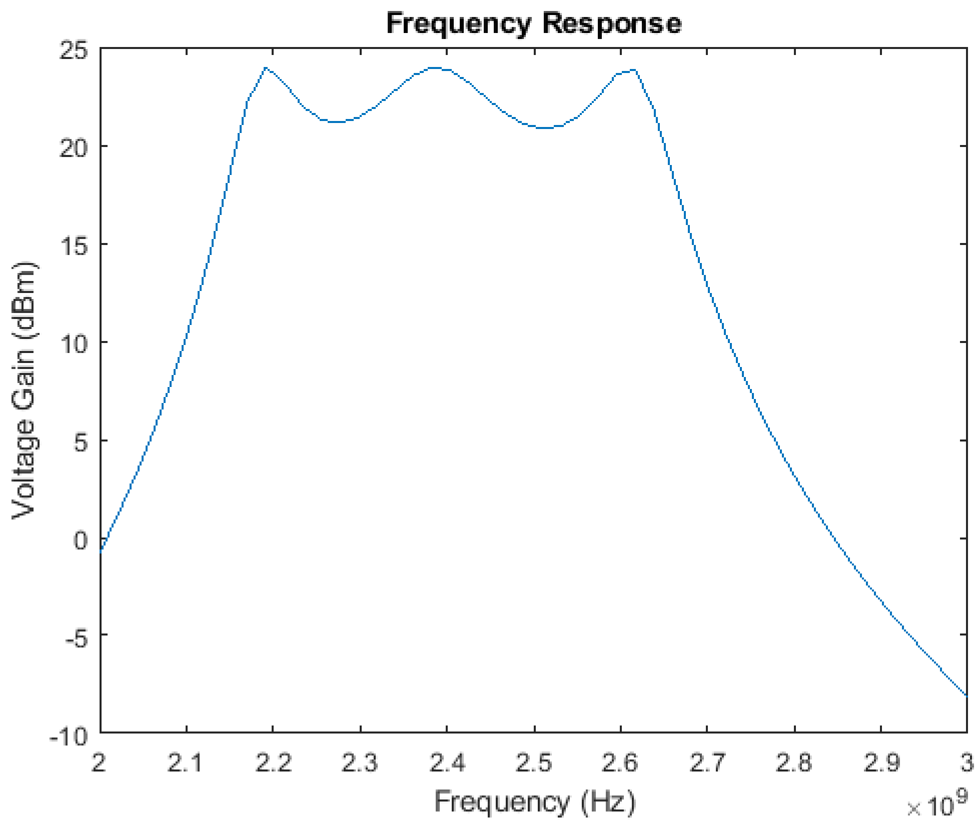
Copyright 2010-2018 The MathWorks, Inc.

The system consists of:

- A Continuous Wave source and a series resistor to model a voltage source with internal source impedance.
- Inductor and Capacitor blocks configured to model a third-order Chebyshev filter with a center frequency of 2.4 GHz.
- An Outport block configured as a voltage sensor to measure the voltage across a load resistor.
- A Configuration block, which sets up the circuit envelope simulation environment. As the system is linear, the harmonic balance analysis is done with a single simulation frequency and corresponds to an AC analysis.

1. Type `open_system('simrfV2_ac_analysis')` at the Command Window prompt.
2. Double-click the block labeled 'Specify Frequency Values' to provide a vector of frequencies.
3. Double-click the block labeled 'Calculate Frequency Response' to execute a script, `simrfV2_ac_analysis_callback`, that analyzes the model at the specified frequencies and plots the response.

```
simrfV2_ac_analysis_callback([model '/Subsystem'], 'OpenFcn');
```



To configure a model with circuit envelope library blocks for harmonic balance:

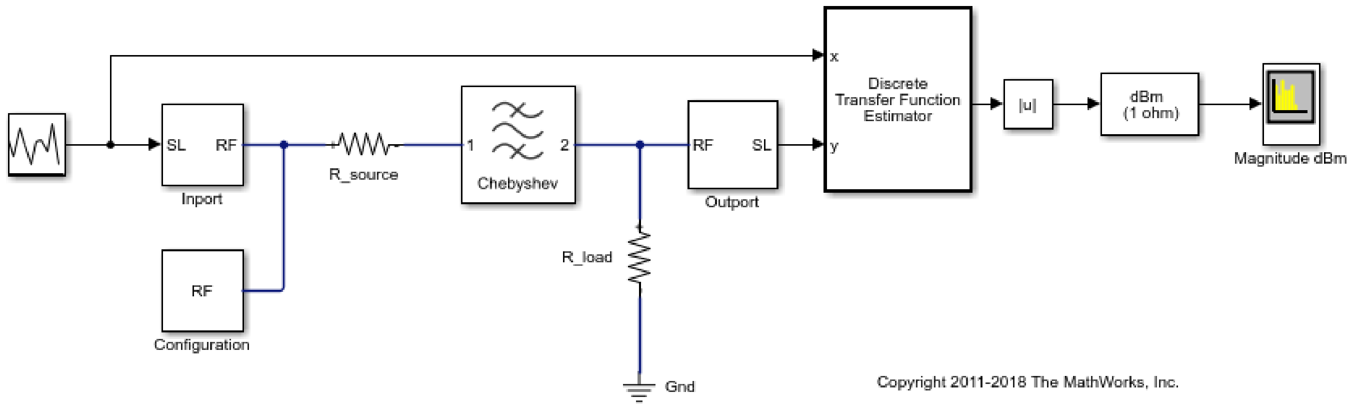
- In the Model Configuration parameters dialog box, set the **Stop time** parameter to zero.
- Use a Continuous Wave block to drive the system.
- Set the **Carrier frequencies** parameter in the Continuous Wave, Outport blocks, and the **Fundamental Tones** parameter in the Configuration block to the same vector of frequencies.

Close the open model

```
bdclose(model)
```

Time Domain Simulation

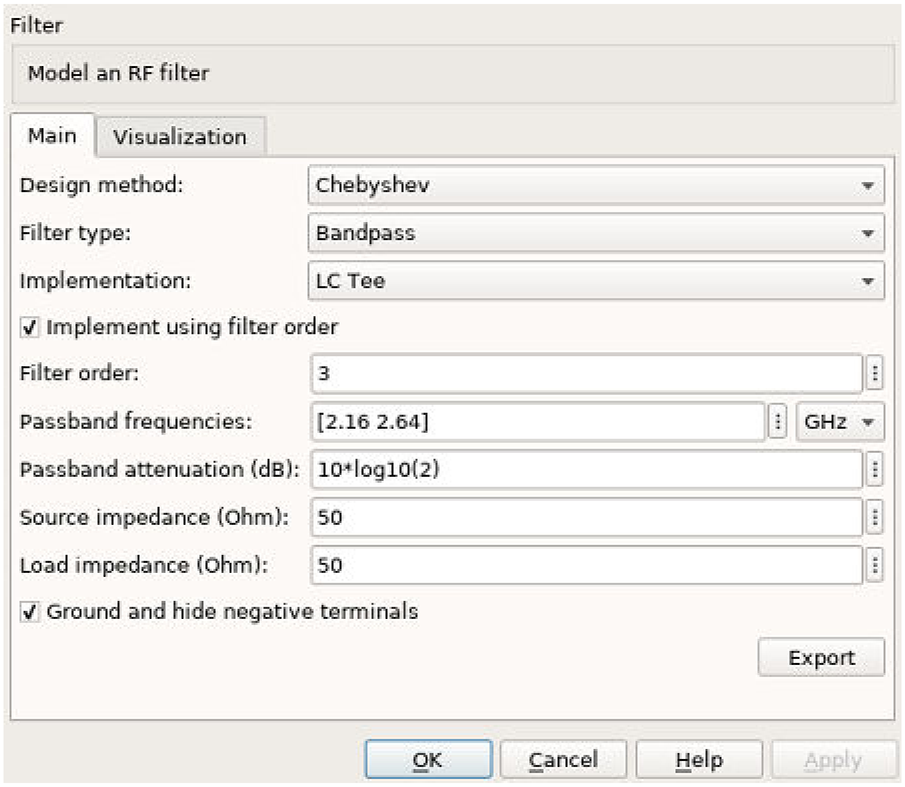
```
model = 'simrfV2_ac_analysis_tf';
open_system(model)
```



The system consists of:

- A Random source generator that outputs a continuous random signal.
- A Chebyshev filter constructed using the Filter library block and designed with a center frequency of 2.4 GHz and a bandwidth of 480 MHz.
- Discrete Transfer Function Estimator block to view the frequency domain output of a time domain simulation.
- Spectrum Analyzer to view the output.

View the filter designed parameters used in the Filter block mask.


Filter

Model an RF filter

Design method: Chebyshev

Filter type: Bandpass

Implementation: LC Tee

[✓] Implement using filter order

Filter order: 3

Passband frequencies: [2.16 2.64] GHz

Passband attenuation (dB): 10*log10(2)

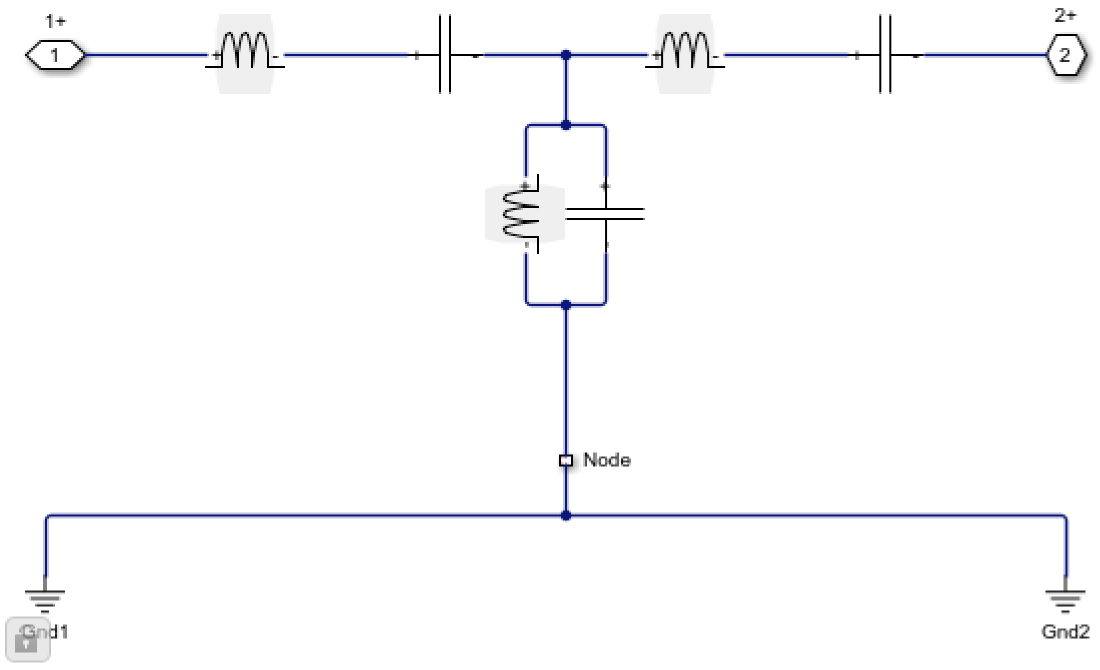
Source impedance (Ohm): 50

Load impedance (Ohm): 50

[✓] Ground and hide negative terminals

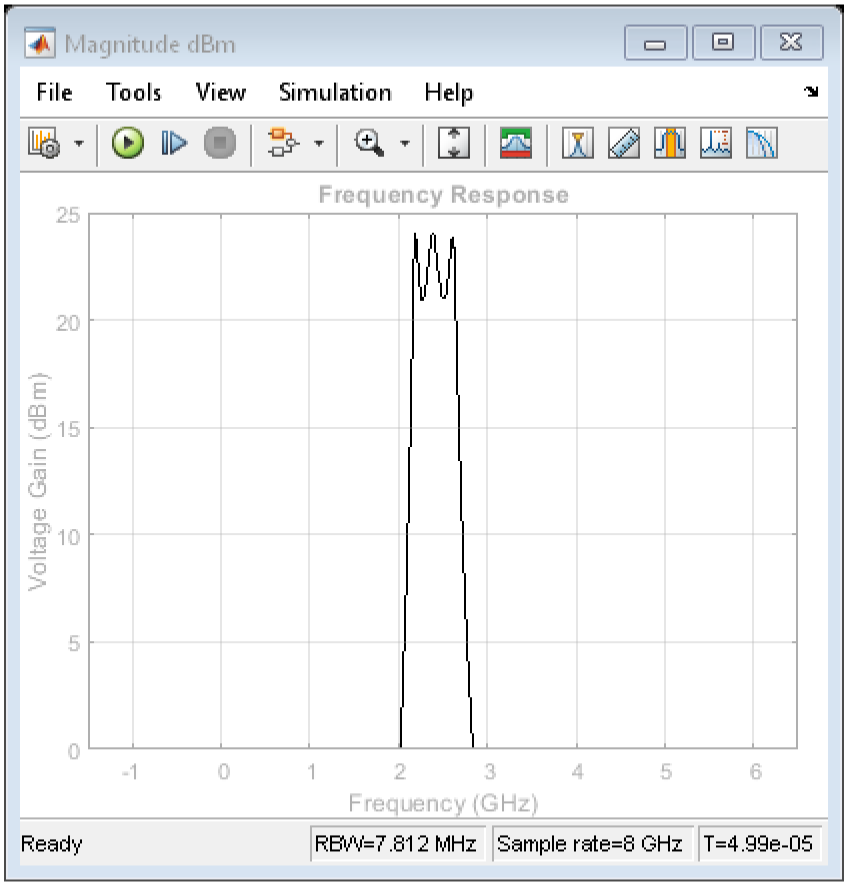
View the implemented filter under the Filter block mask.

```
open_system([model '/Filter'],'force')
```



Simulate the transfer system model.

```
sim(model,5e-5)
```

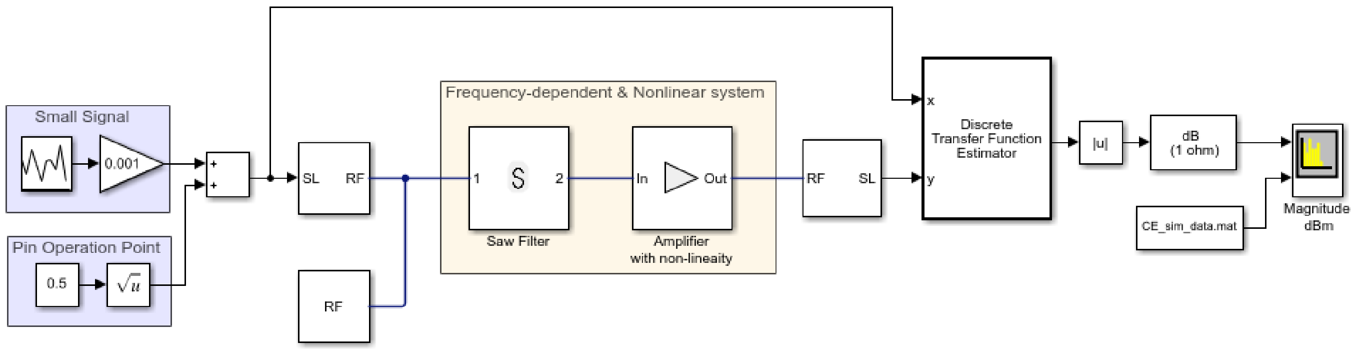


Compare the outputs of the first and second model.

```
bdclose(model)
```

Small Signal Analysis

```
model = 'simrfV2_ac_analysis_ss';
open_system(model)
```

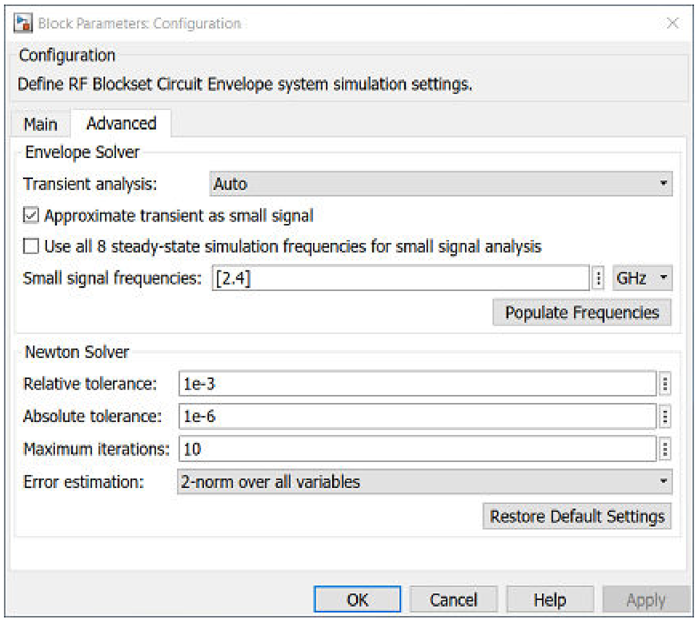


The system consists of:

- A Random source generator that outputs a continuous random signal that is subsequently attenuated to ensure small signal input.

- A constant source added to the random source to determine the non-linear operation point. Both signals are centered at 2.4 GHz.
- An RF System comprising two elements; A saw filter constructed using the S-parameter library block with a center frequency of 2.45 GHz and a bandwidth of 112 MHz and an amplifier with 20dB of available power gain and non-linearity described by a 3rd-order intercept point of 30dBm.
- Discrete Transfer Function Estimator block to view the frequency domain output of a time domain simulation measured over the 2.4 GHz carrier.
- Spectrum Analyzer to view the output and compare it to saved output data.

Since the transient signal is small while the operation point is determined based on carrier-constant large signals, it is possible to use the transient small-signal approximation. In this approximation, non-linear interaction between transient signals is ignored, however the non-linear interaction between carrier-constant signals and its effect on the small signals is captured accurately. The small signal analysis is enabled in the advanced tab of the Configuration block mask.


Block Parameters: Configuration

Configuration

Define RF Blockset Circuit Envelope system simulation settings.

Main | Advanced

Envelope Solver

Transient analysis: Auto

[✓] Approximate transient as small signal

[ ] Use all 8 steady-state simulation frequencies for small signal analysis

Small signal frequencies: [2.4] GHz

Populate Frequencies

Newton Solver

Relative tolerance: 1e-3

Absolute tolerance: 1e-6

Maximum iterations: 10

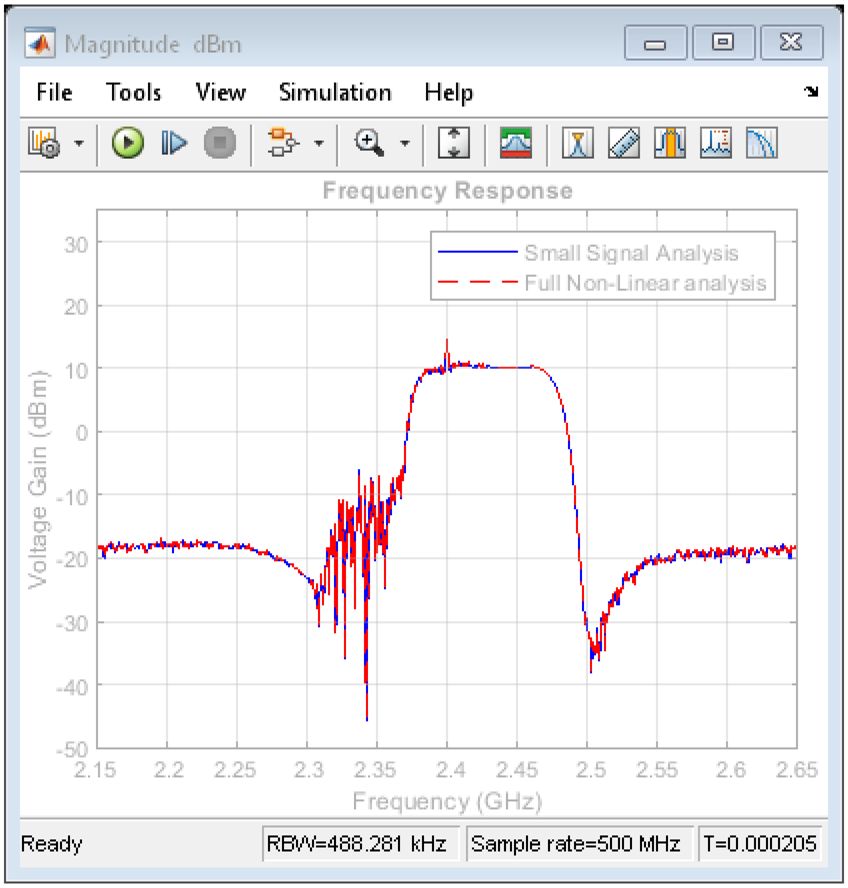
Error estimation: 2-norm over all variables

Restore Default Settings

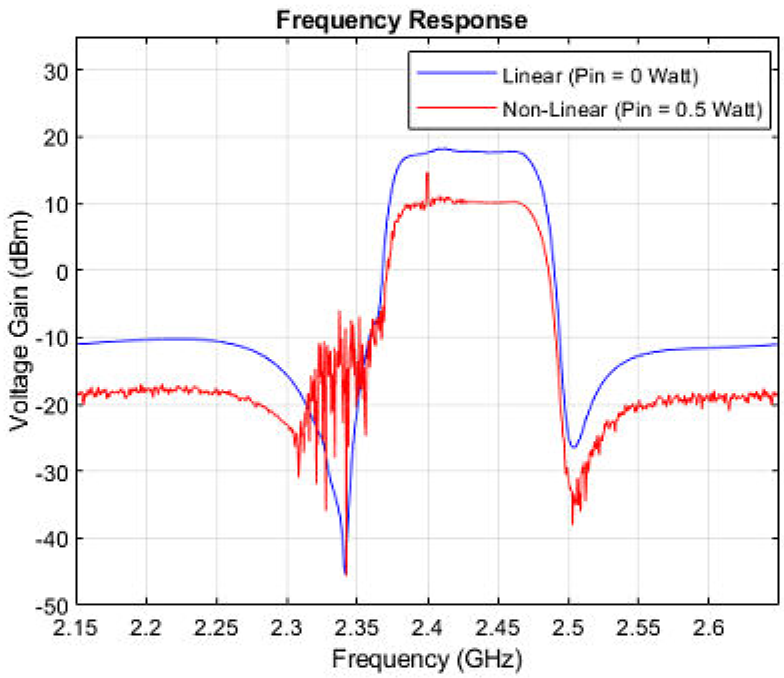
OK Cancel Help Apply

Using small signal analysis, a subset of the full set of carriers used for steady state solution can be chosen for transient simulation. In this example, only the 2.4 GHz is of interest for transient analysis. Reducing the number of simulated carriers, accelerates the simulation. In this case, the small signal simulation is more than 15 times faster than a full non-linear circuit-envelope based simulation. Comparing the small-signal simulation results with those of a full circuit envelope simulation loaded from a file, it is evident that the results are practically identical.

```
sim(model)
```



Decreasing the operation point power in the constant block from 0.5 Watt down to zero, the system becomes effectively linear. A comparison between the curves illustrates the effect of the non-linearity on the transfer function. These effects include a decrease in overall amplitude due to compression and a widening of the filter profile at the lower-frequency side. The widening can be explained as the result of the cubic term in the amplifier polynomial response folding the original RF frequency of 2.4 GHz back onto itself, but with a frequency response that is flipped around its central frequency since 2.4 GHz is reached by reflection from -2.4 GHz. Since the Saw filter is centered at 2.45GHz, the flipped frequency response is centered at 2.35GHz. Summing the linear and cube terms effects yields a widened profile.



```
bdclose(model)
```

References

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

Mass A. Stephen, *Nonlinear Microwave and RF Circuits*. Artech House, 2003.

See Also

"Compare Time and Frequency Domain Simulation Options for S-parameters" on page 9-42

Compare Time and Frequency Domain Simulation Options for S-parameters

This example shows how to use two different options for modeling S-parameters with the RF Blockset™ Circuit Envelope library. The Time-domain (rationalfit) technique creates an analytical rational model that approximates the whole range of the data. This is a preferable technique when a good fit could be achieved with a small number of poles. When the data has a lot of details or high level of noise, this model becomes large and slow to simulate.

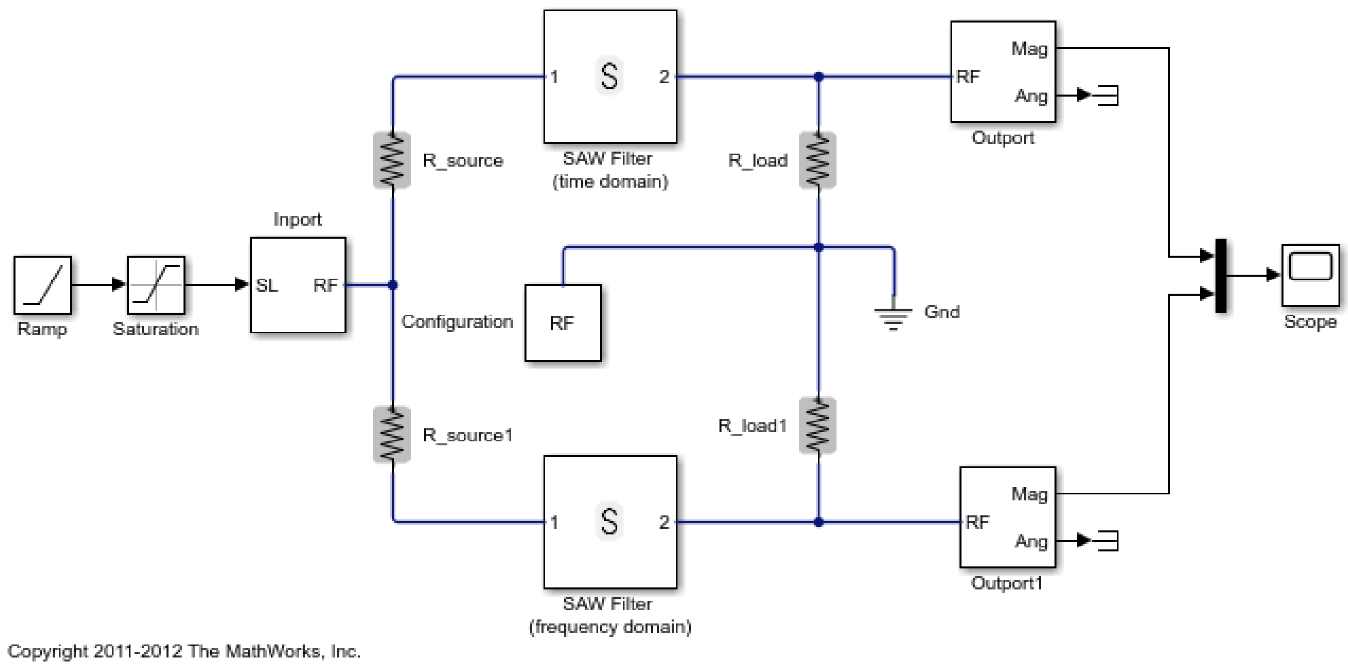
The frequency-domain technique is based on convolution, where the baseband impulse response depends on the simulation time step and the carrier frequency.

System Architecture

The system consists of:

- An input envelope signal modeled with Simulink blocks. The input signal is a ramp that goes from 0 to 1 in `TF_RAMP_TIME`; the initial value of `TF_RAMP_TIME` is set to `1e-6` s. The carrier frequency of the signal is `TF_FREQ`; the initial value of `TF_FREQ` is set to `2.4e9` Hz.
- Two SAW filters, modeled by two S-parameter blocks using the same data file, `sawfilter.s2p`. The block labeled `SAW Filter (time domain)` has its **Modeling options** parameter in the Modeling tab set to `Time domain (rationalfit)`. The block labeled `SAW Filter (frequency domain)` has its **Modeling options** parameter in the Modeling tab set to `Frequency domain` and the **Automatically estimate impulse response duration** is checked.
- A Scope block that displays the outputs of the two S-parameter blocks.

```
model = 'simrfV2_sparam_t_vs_f';
open_system(model);
```

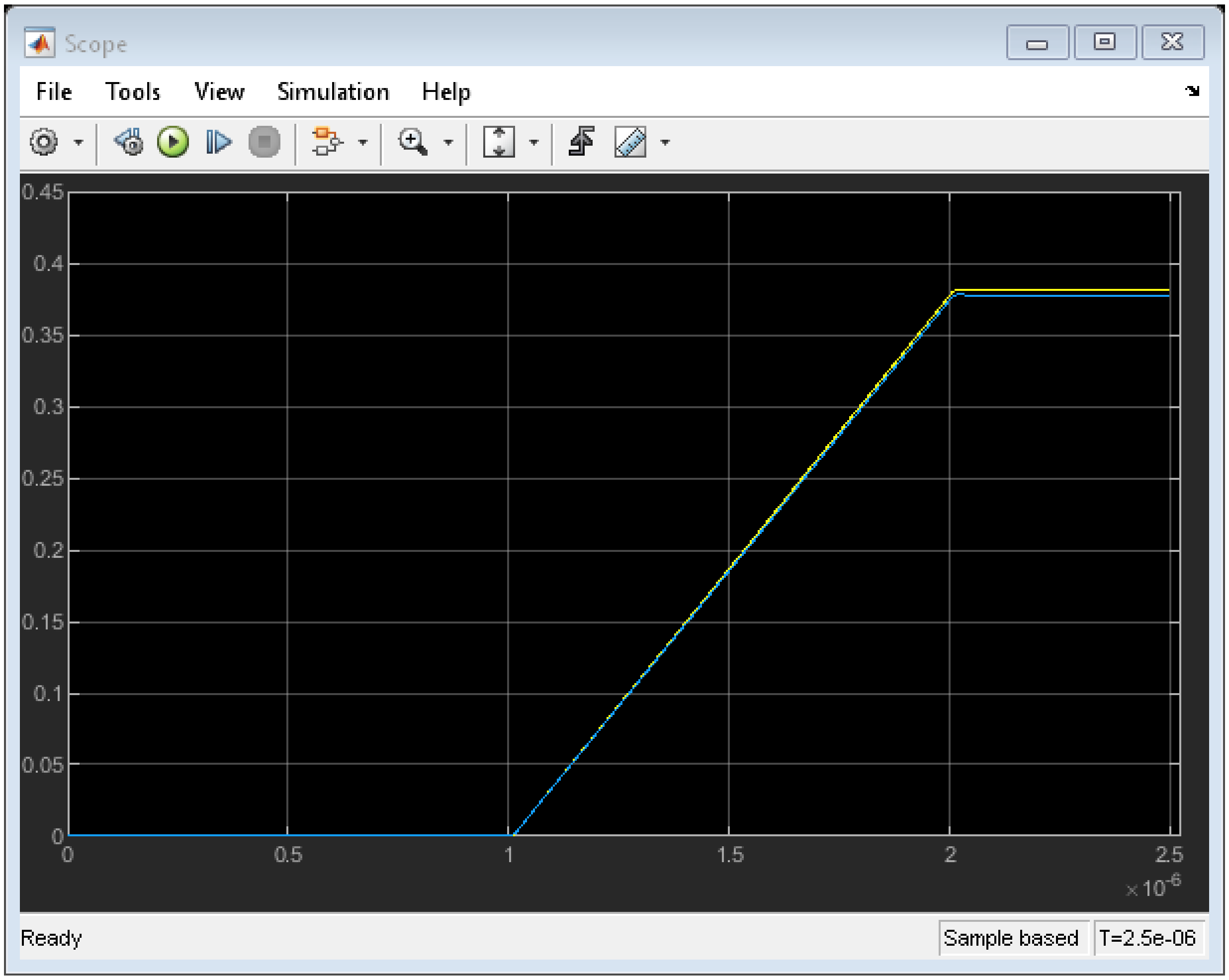


Run Simulation with the Default Settings

1 Type `open_system('simrfV2_sparam_t_vs_f')` at the Command Window prompt.
2 Select **Simulation** > **Run**.

The outputs from both methods are very close to each other. The frequency-domain model (purple curve) captures the transfer function (steady-state value) a bit better.

```
scope = [model '/Scope'];
open_system(scope);
set_param(scope, 'YMax','0.45');
set_param(scope, 'YMin','0');
set_param(scope, 'TimeRange',num2str(1.01*TF_END_TIME));
sim(model);
```



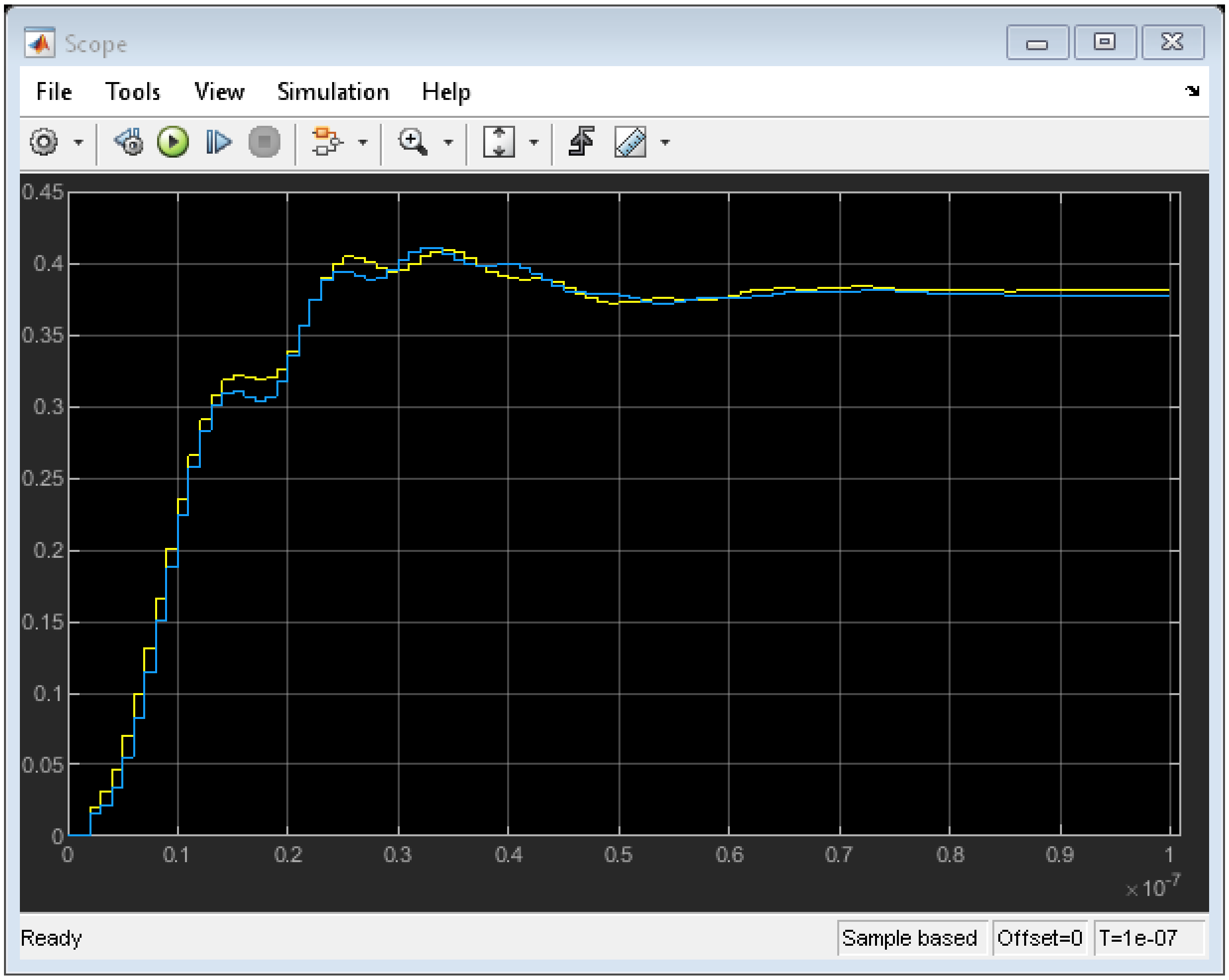
Run the Simulation with the Very Steep Ramp

In the previous simulation, the rise time of the envelope `TF_RAMP_TIME = 1e-6` was many orders of magnitude greater than the period of the carrier signal `T = 1/TF_FREQ = 4.1667e-10`. In other words, the envelope was much slower than the carrier. As the ramp time approaches the period of the carrier, the corresponding time effects are better captured by the time-domain model (yellow curve).

To continue the example:

1 Type `TF_RAMP_TIME = 1e-9; TF_END_TIME = 1e-7;` at the Command Window prompt.
2 Select **Simulation** > **Run**.

```
TF_RAMP_TIME = 1e-9;
TF_END_TIME = 1e-7;
set_param(scope, 'TimeRange',num2str(1.01*TF_END_TIME));
sim(model);
open_system(scope);
```



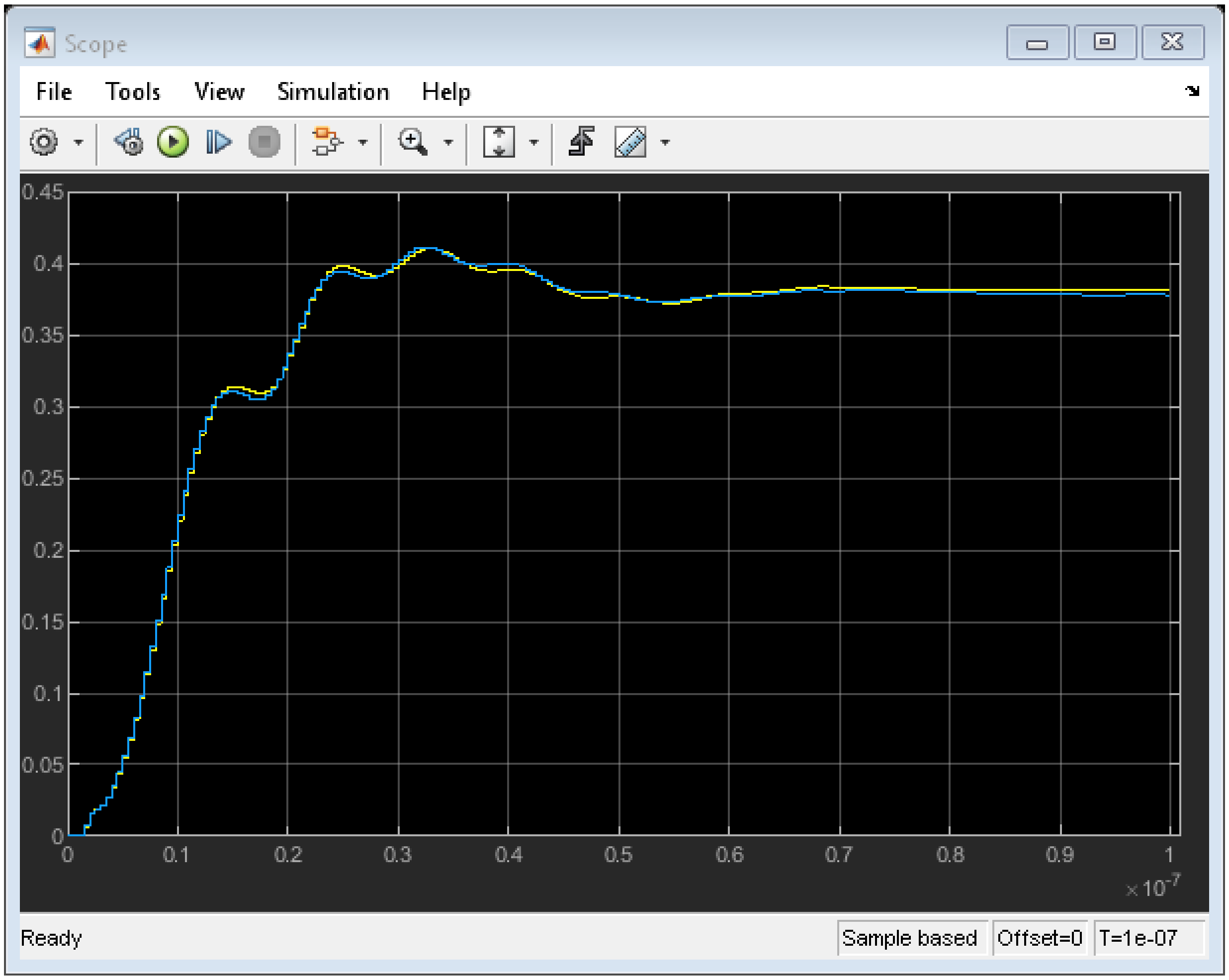
The result of the frequency-domain simulation can be improved by decreasing the time step of the simulation and manually setting the impulse duration time.

To continue the example:

1 Type `TF_STEP = 5e-10;` at the Command Window prompt.
2 Uncheck **Automatically estimate impulse response duration** in the modeling pane of `Saw filter (frequency domain)` block and specify the Impulse Response Duration as `1e-7`.
3 Select **Simulation** > **Run**.

```
TF_STEP = 5e-10;
sparam_freq  = [model '/SAW Filter  (frequency domain)'];
```

```
set_param(sparam_freq, 'AutoImpulseLength', 'off');
set_param(sparam_freq, 'ImpulseLength', '1e-7');
sim(model);
open_system(scope);
```



Run Simulation with Different Frequency

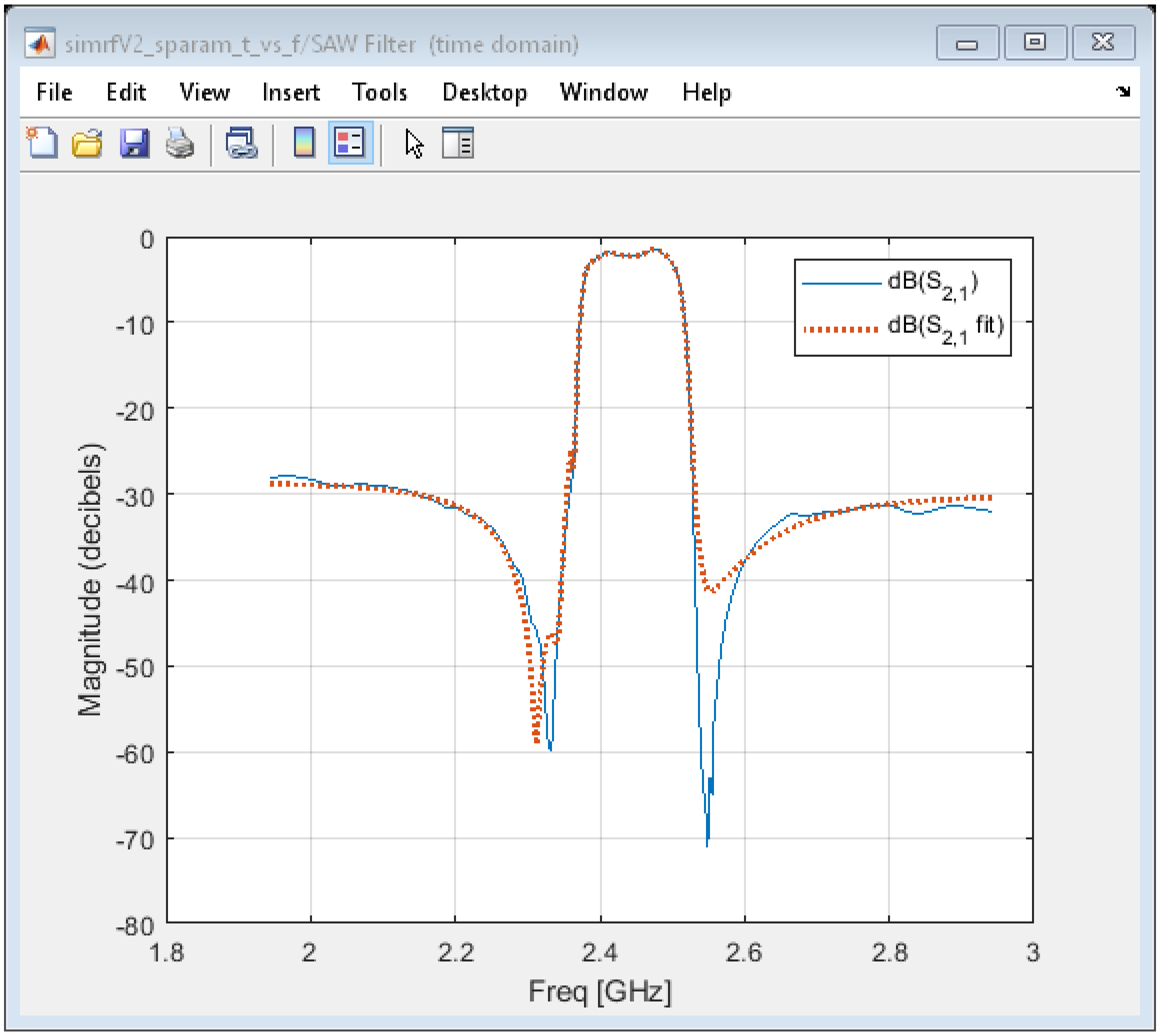
Rational-function approximation is not exact. To see the approximation error, double-click the "SAW Filter (time domain)" block. Information about the approximation appears under "Rational fitting results" in the bottom of the dialog 'Modeling' pane.

```
open_system([model sprintf('/SAW Filter  (time domain)')]);
```

For more details, select 'Visualization' panel, and click the 'Plot' button.

The rationalfit algorithm (dotted curve) does a very good job for the most of the frequencies. However, sometimes it does not capture the sharp changes of S-parameter data.

```
simrfV2_click_dialog_button('Block Parameters: SAW Filter  (time domain)', 'PlotButton');
```



Conversely, the frequency-domain method exactly reproduces the steady-state behavior at all carrier frequencies (by definition). Running the simulation for `TF_FREQ = 2.54e9` produces drastically different results between the two S-parameter methods.

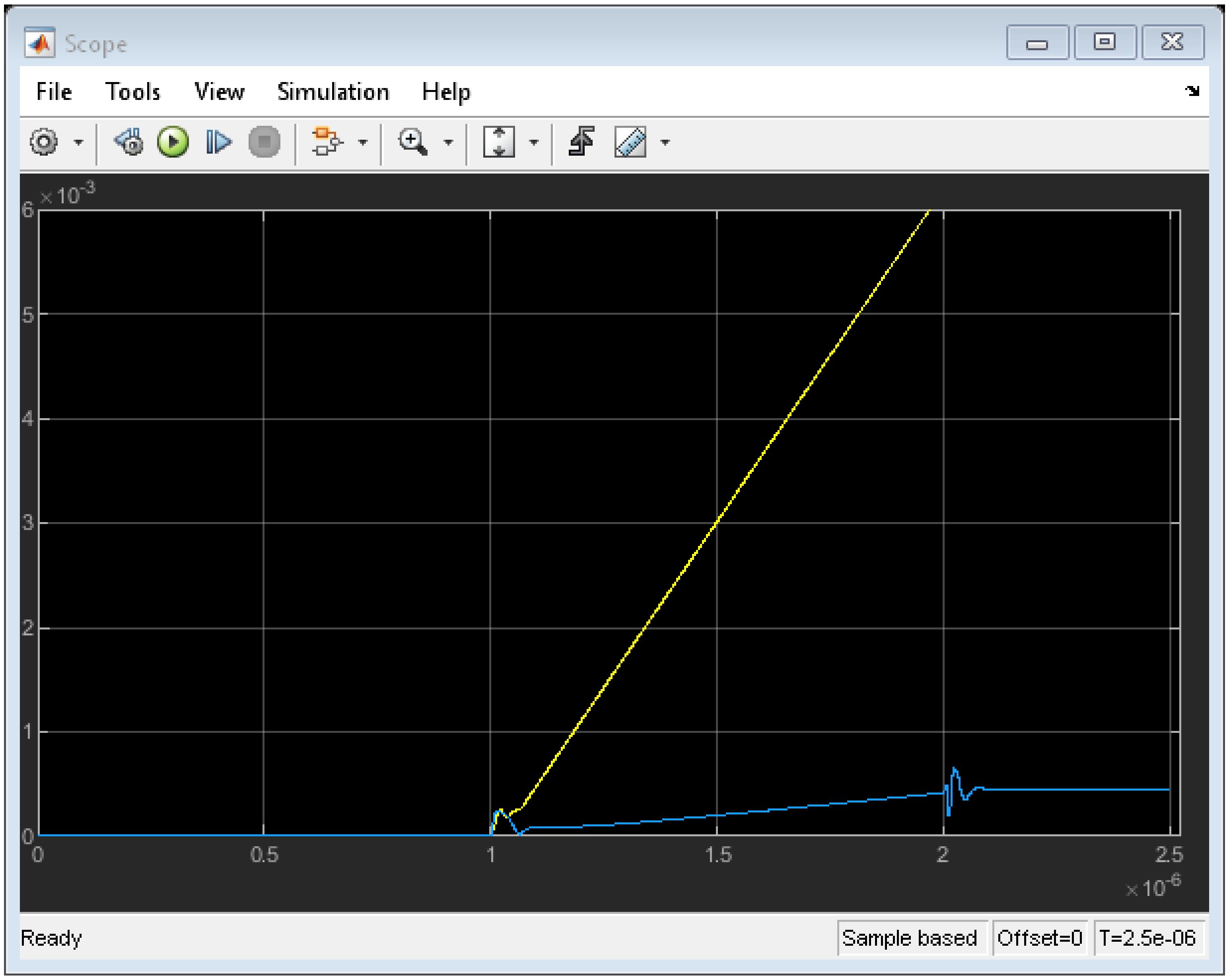
To continue the example:

1 Type `TF_FREQ = 2.54e9; TF_RAMP_TIME = 1e-6; TF_STEP = 3e-9; TF_END_TIME = 2.5e-6;` at the Command Window prompt.

2 Select **Simulation** > **Run**.

In this case, the frequency-domain model provides a better approximation of the original data.

```
TF_STEP = 3e-9;
TF_RAMP_TIME = 1e-6;
TF_FREQ = 2.54e9;
TF_END_TIME = 2.5e-6;
set_param(scope, 'YMax','6e-3');
set_param(scope, 'TimeRange',num2str(1.01*TF_END_TIME));
sim(model);
open_system(scope);
```



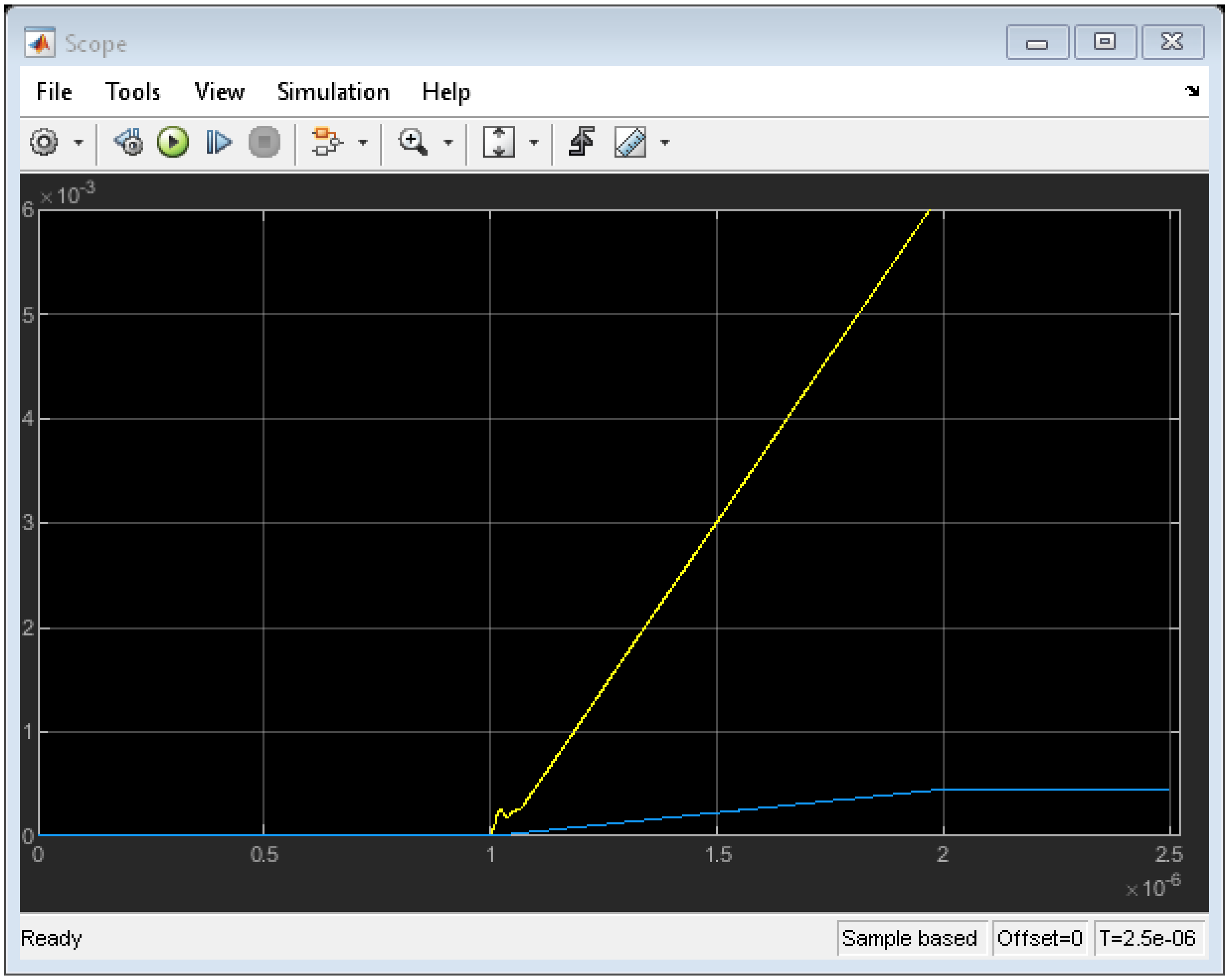
Run Simulation with Impulse Duration Set to Zero.

There is a special case that could be very helpful in practice. When the "Impulse Response Duration" of the s-parameters block is set to zero, the history of the input is no longer taken into consideration. Still, the model captures the transfer function (steady-state value) correctly. This is a fast and reliable way to model the ideal devices when the transient effects could be ignored.

To continue the example:

1 Specify the `Impulse Response Duration` of `Saw filter (frequency domain)` block as `0`.
2 Select **Simulation** > **Run**.

```
set_param(sparam_freq, 'ImpulseLength', '0');
sim(model);
open_system(scope);
```



Conclusion

In most practical RF systems, time- and frequency-domain techniques give similar answers. The time-domain method better captures the time-domain effects of the fast-changing envelopes, but relies on a rationalfit approximation of the original data. The frequency-domain method is sensitive to the simulation time step; this option is recommended when the time-domain model does not provide a good fit.

```
close gcf;
bdclose(model);
clear model scope;
```

See Also

S-Parameters | Configuration | Inport | Outport

Related Topics

“Analysis of Frequency Response of RF System” on page 9-34 | “Transmission Lines, Delay-based and Lumped Models” on page 9-49

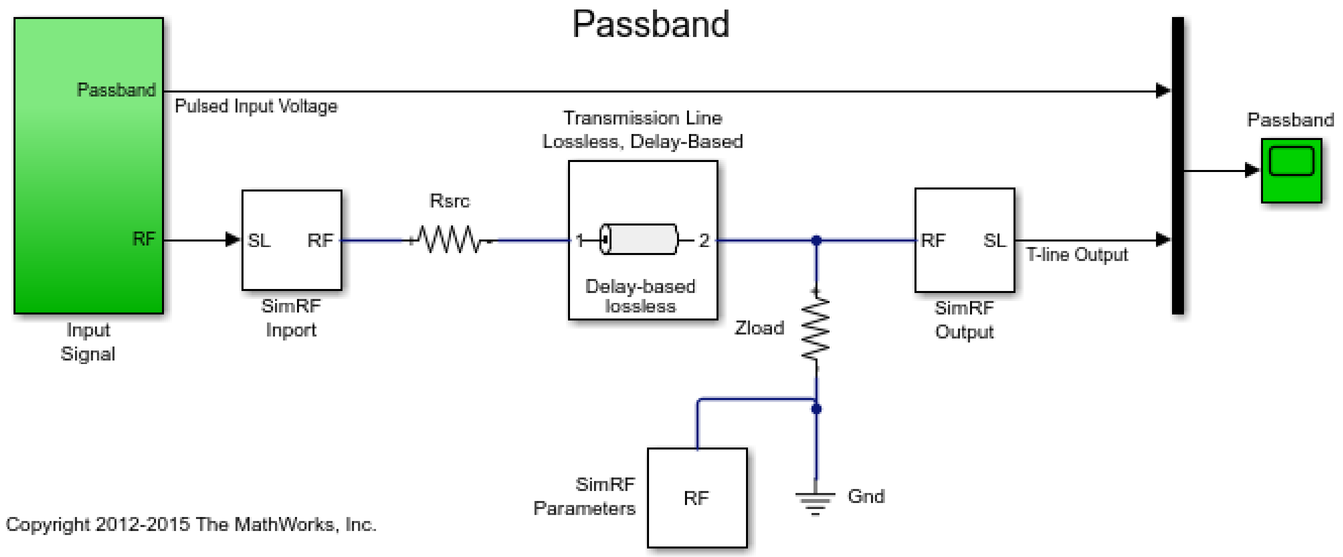
Transmission Lines, Delay-based and Lumped Models

This example shows how to simulate delay-based and lumped-element Transmission Line using blocks in the RF Blockset™ Circuit Envelope library. The example is sequenced to examine circuit envelope and passband differences, delay-based lossy transmission line sectioning, and lumped element implementation of delay.

System Architecture for Lossless Delay-Based Transmission Line

In this section, two RF Blockset™ models, `simrf_xline_pb` and `simrf_xline_ce`, illustrate lossless delay-based transmission line effects and the computational benefit of circuit envelope techniques.

```
model_pb = 'simrf_xline_pb';
model_ce = 'simrf_xline_ce';
load_system(model_ce)
open_system(model_pb)
```

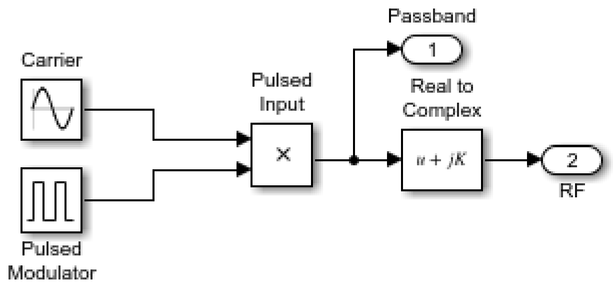


The model, `simrf_xline_pb`, represents a passband signal as:

$$I(t)\cos 2\pi f_c t - Q(t)\sin 2\pi f_c t$$

The input is a pulse-modulated sinusoidal passband signal. For this particular case, I(t) equals zero, and Q(t) is the pulse modulation. The carrier frequencies are set to zero in the RF Blockset Inport and Outport blocks.

```
open_system([model_pb '/Input Signal']);
```

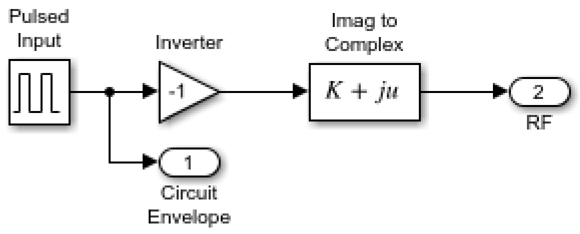


The circuit envelope model, `simrf_xline_ce`, represents an envelope signal as:

$$I(t) + jQ(t)$$

Again, I(t) equals zero, and Q(t) is the pulse modulation, but the carrier signal is not specified as part of the input signal. To model the carrier, the `Carrier Frequencies` parameter is set to f_c in the RF Blockset Inport and Outport blocks.

```
open_system([model_ce '/Input Signal']);
```



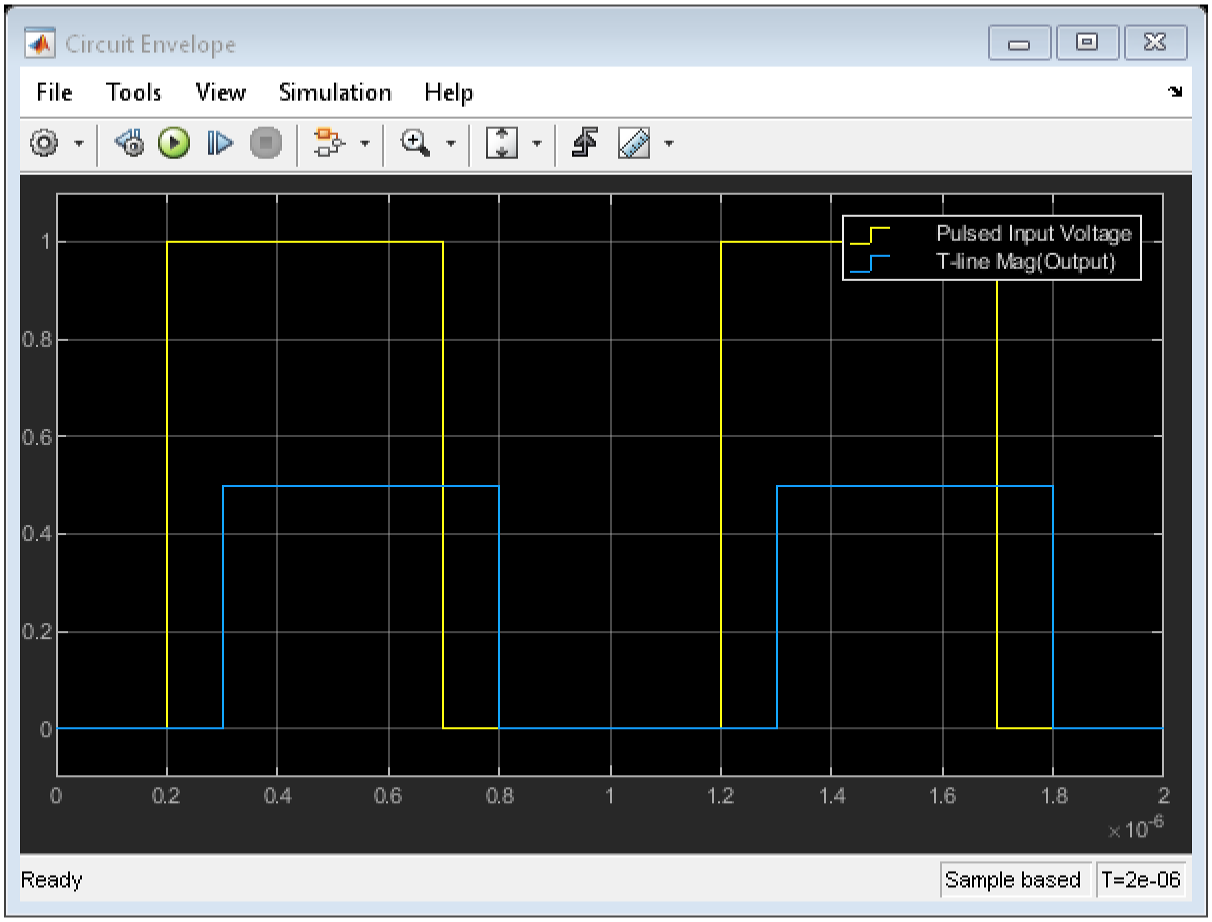
Removal of the explicit sinusoidal carrier in the circuit-envelope model allows the simulation to reduce time-steps relative to the passband model.

Running the Lossless Delay-Based Transmission Line

1 Type `open_system('simrf_xline_pb')` or `open_system('simrf_xline_ce')` at the Command Window prompt.
2 Select **Simulation** > **Run**.

After simulating, the transmission delay is observable in a plot of input and output signals.

```
open_system([model_ce '/Circuit Envelope']);
sim(model_ce);
```

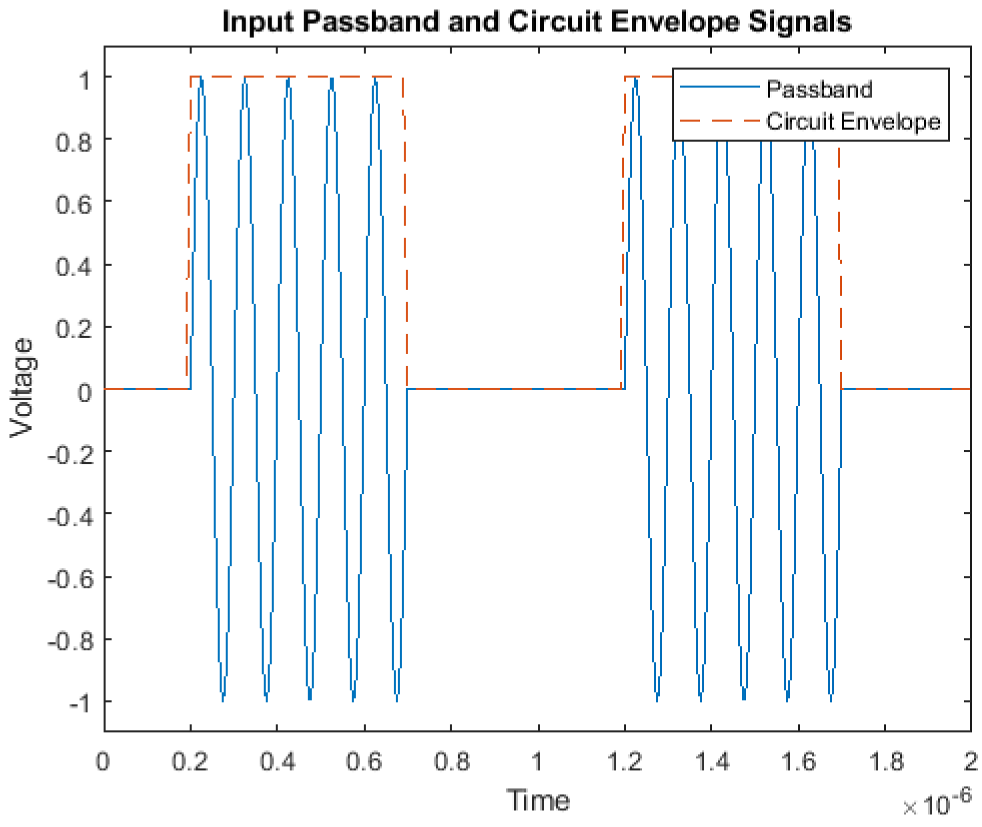


The carriers in modulated waveforms appear in passband signals, but only the modulation envelopes appear in circuit-envelope signals. Passband signals can be reconstructed from circuit envelope signals as:

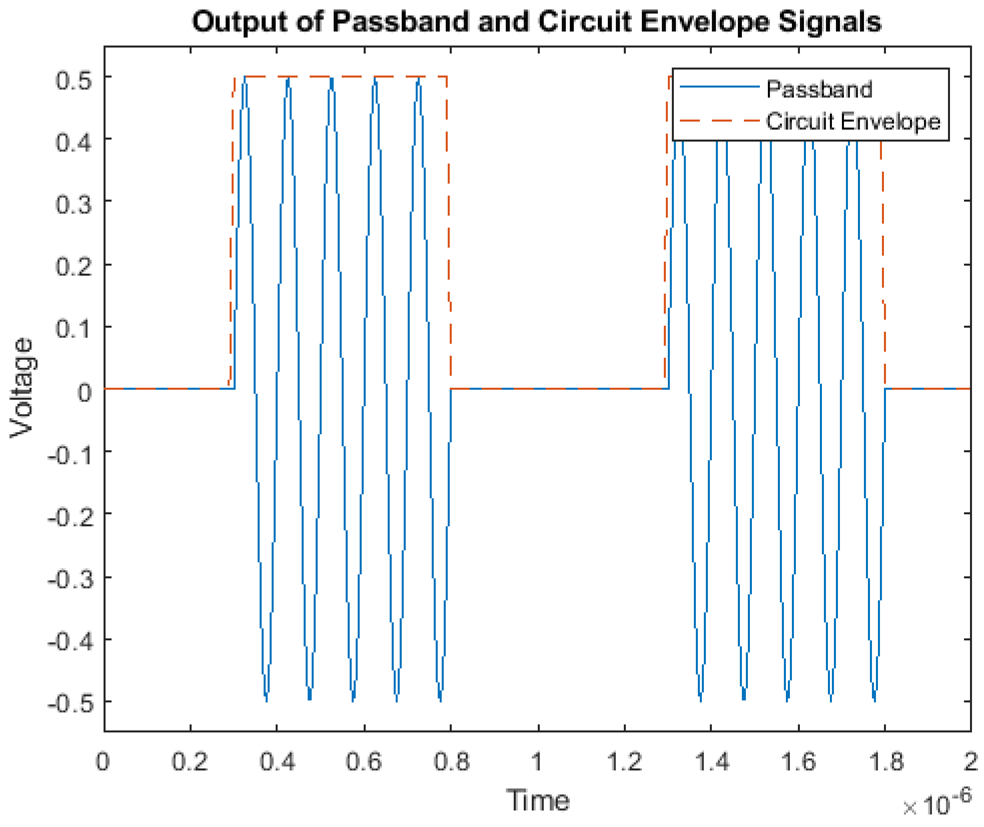
$$Re((I(t) + jQ(t))e^{j2\pi f_c t})$$

However, reconstruction of the passband signal this way requires additional time steps for the carrier.

```
sim(model_pb);
hline = plot(SPB_Data(:,1),SPB_Data(:,2),SCE_Data(:,1),SCE_Data(:,2),'--');
legend('Passband', 'Circuit Envelope')
title('Input Passband and Circuit Envelope Signals')
xlabel('Time')
ylabel('Voltage')
ylim([-1.1 1.1])
```



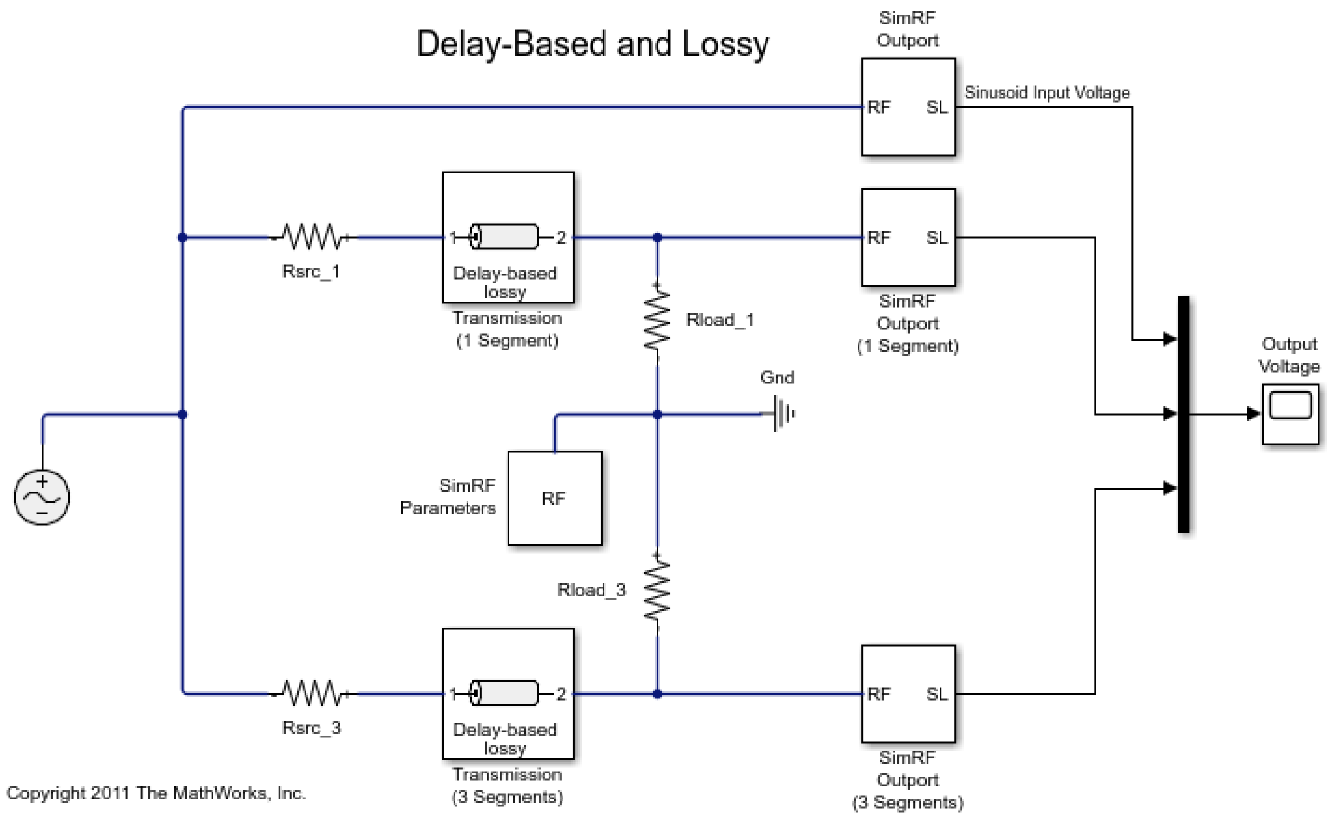
```
haxis = get(hline(1),'Parent');
plot(haxis,SPB_Data(:,1),SPB_Data(:,3),SCE_Data(:,1),SCE_Data(:,3),'--')
legend('Passband', 'Circuit Envelope')
title('Output of Passband and Circuit Envelope Signals')
xlabel('Time')
ylabel('Voltage')
ylim([-.55 .55])
```



Partitioning Delay-Based Lossy Transmission Lines

A conventional method for modeling distributed lossy transmission lines employs *N* two-port segments in cascade. Each segment consists of an ideal lossless delay line and resistance, where the segment delay equals the total line delay divided by *N* and the segment resistance equals the total line resistance divided by *N*. As the number of segments increases, the lumped model will more accurately represent the distributed system. This methodology requires a compromise between simulation time and model accuracy for increasing *N*. In RF Blockset, the `Number of segments`, the `Resistance per unit length` and the `Line length` are specified as dialog box parameters in the transmission line block.

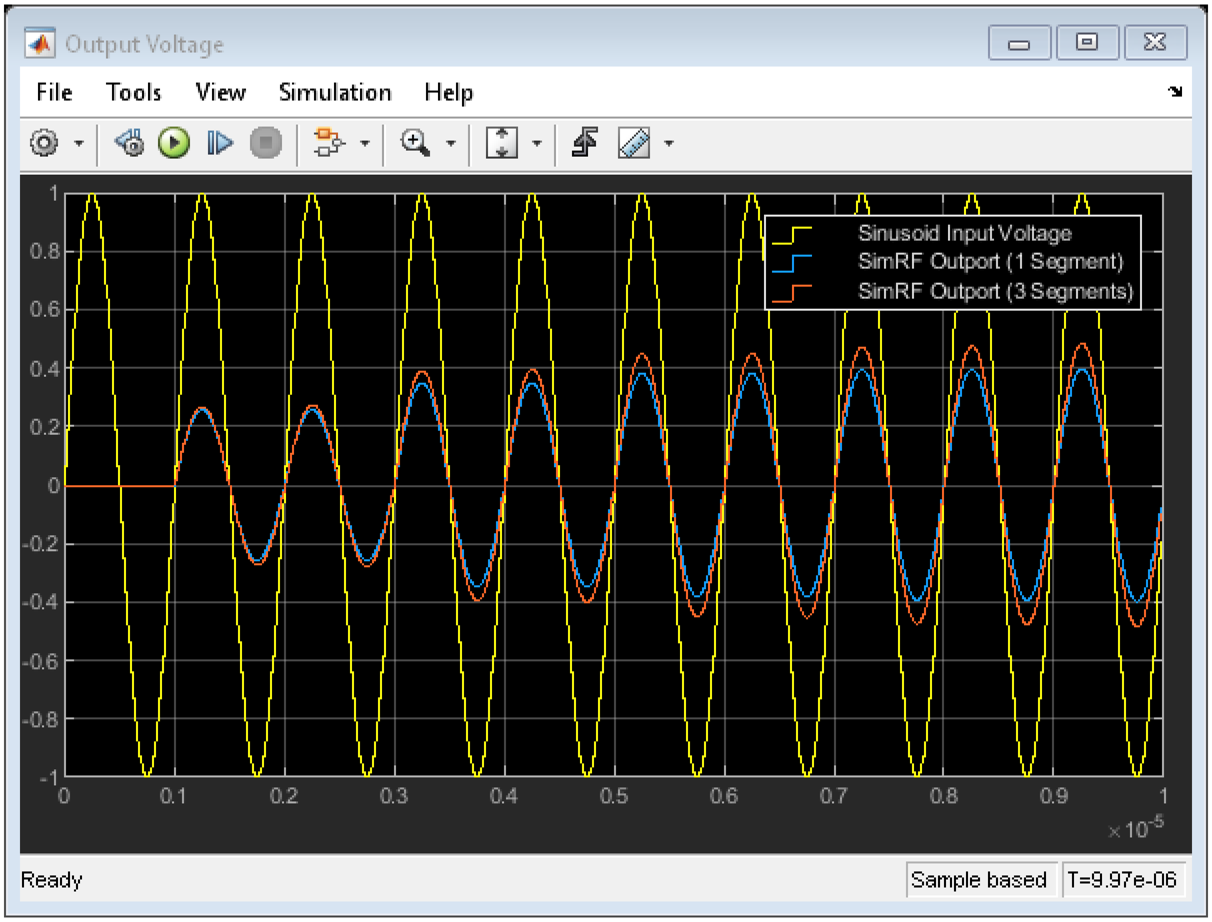
```
model_seg = 'simrf_xline_seg';
open_system(model_seg)
```


Delay-Based and Lossy
SimRF Outport
RF
SL
Sinusoid Input Voltage
Rsrc_1
1
2
Delay-based lossy Transmission (1 Segment)
Rload_1
SimRF Outport (1 Segment)
Output Voltage
Gnd
SimRF Parameters
Rload_3
Rsrc_3
Delay-based lossy Transmission (3 Segments)
SimRF Outport (3 Segments)
Copyright 2011 The MathWorks, Inc.

System Architecture for Lossy Delay-Based Transmission Line

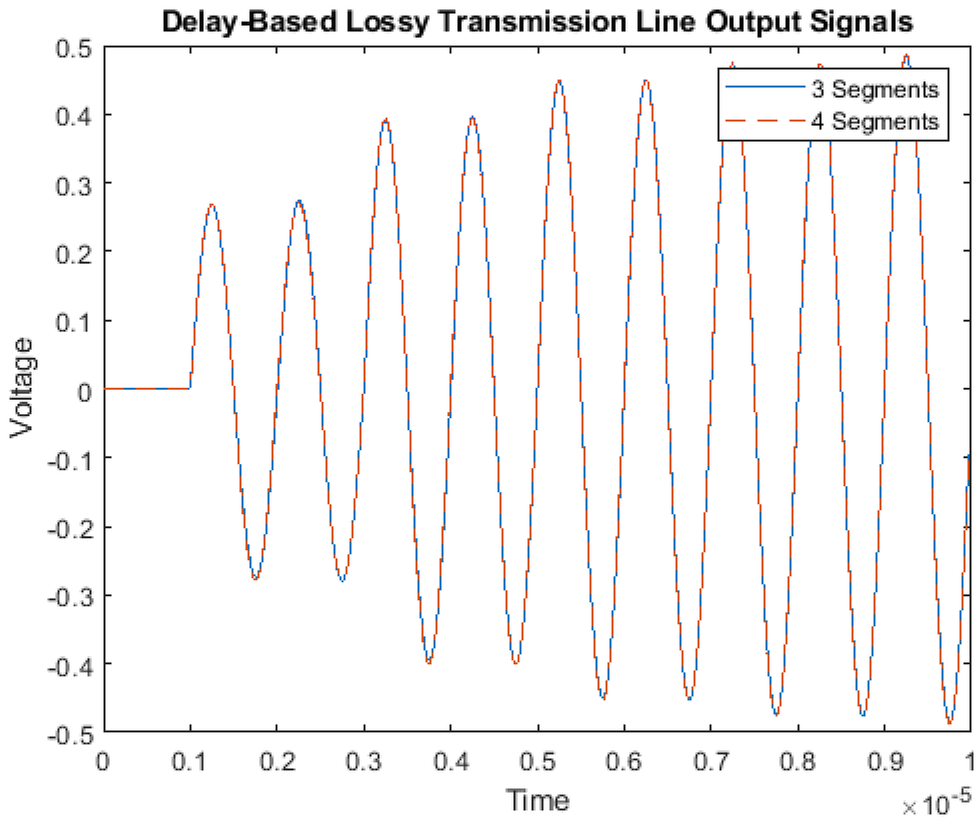
The lossy delay-based transmission line model, `simrf_xline_seg`, consists of two parallel arms excited by a RF Blockset sinusoidal source. The top arm employs a single segment transmission line, while the bottom arm uses a line consisting of 3 segments. The source and load resistances are not equal to the characteristic impedance of the transmission line. These differences affect the shape of the output response. For example, the output response will be overdamped when the source and load resistances are less than the characteristic impedance.

```
open_system([model_seg '/Output Voltage']);
sim(model_seg);
```



Increasing the number of line segments in the bottom arm from three to four and comparing responses show that three segments suffice for this configuration.

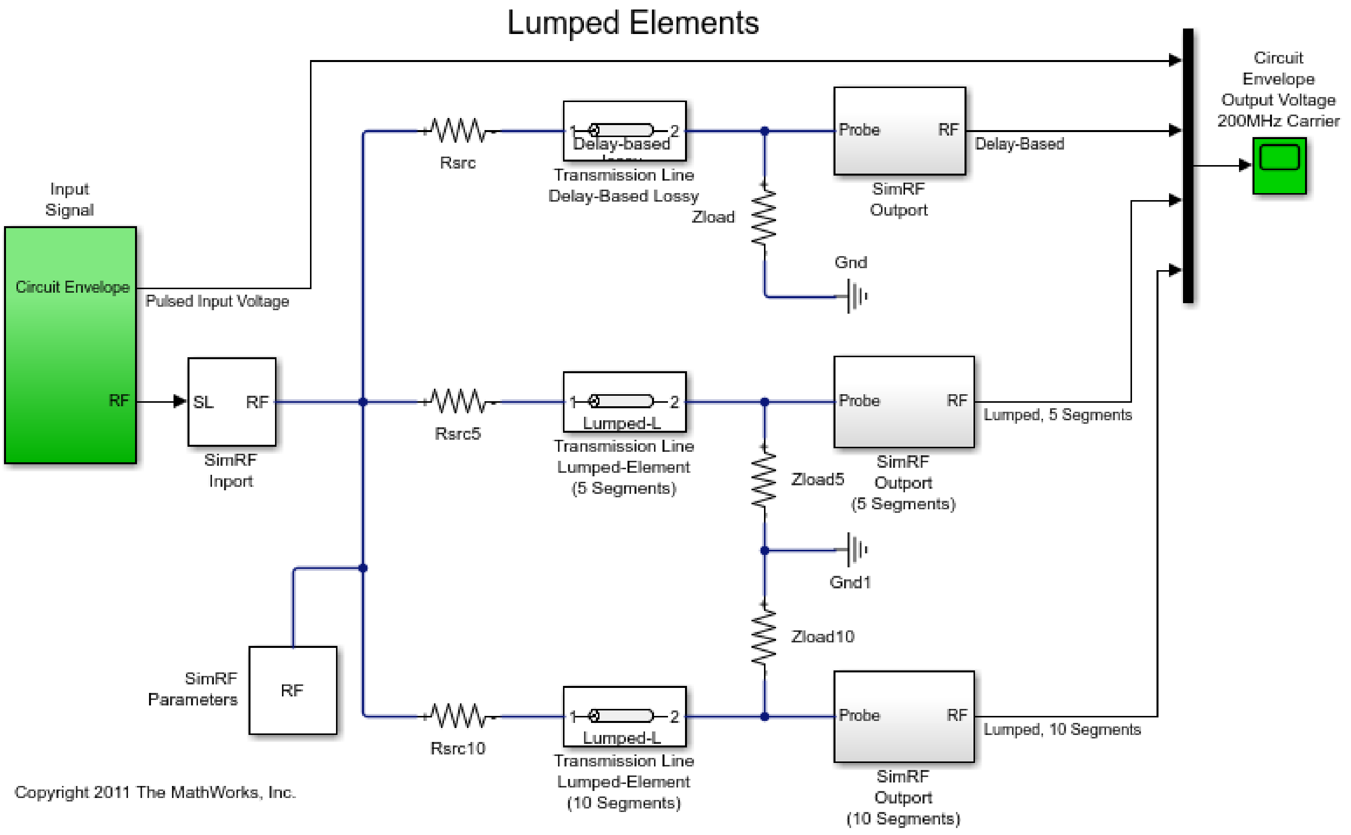
```
close_system([model_seg '/Output Voltage']);
ST_Data3 = ST_Data;
set_param([model_seg '/Transmission (3 Segments)'],'NumSegments','4')
sim(model_seg);
plot(haxis, ST_Data3(:,1), ST_Data3(:,4), ST_Data(:,1), ST_Data(:,4), '--')
legend('3 Segments', '4 Segments')
title('Delay-Based Lossy Transmission Line Output Signals')
xlabel('Time')
ylabel('Voltage')
```



System Architecture for Lumped Element Transmission Line

Differences between the lumped element and delay-based transmission lines are now examined. Consider the model `simrf_xline_ll`, where the dialog box parameter `Model_type` is `Delay-based and lossy` for the top arm and `Lumped parameter L-section` for the other two arms. The `Inductance per unit length` and `Capacitance per unit length` parameters values for the L-section lines are similar to a 50 Ω coaxial cable. Basic first order approximations for these lines are $Z_0 = \sqrt{L/C}$ and $T_D = \sqrt{L*C}*Length.$

```
model_ll = 'simrf_xline_ll';
open_system(model_ll)
```

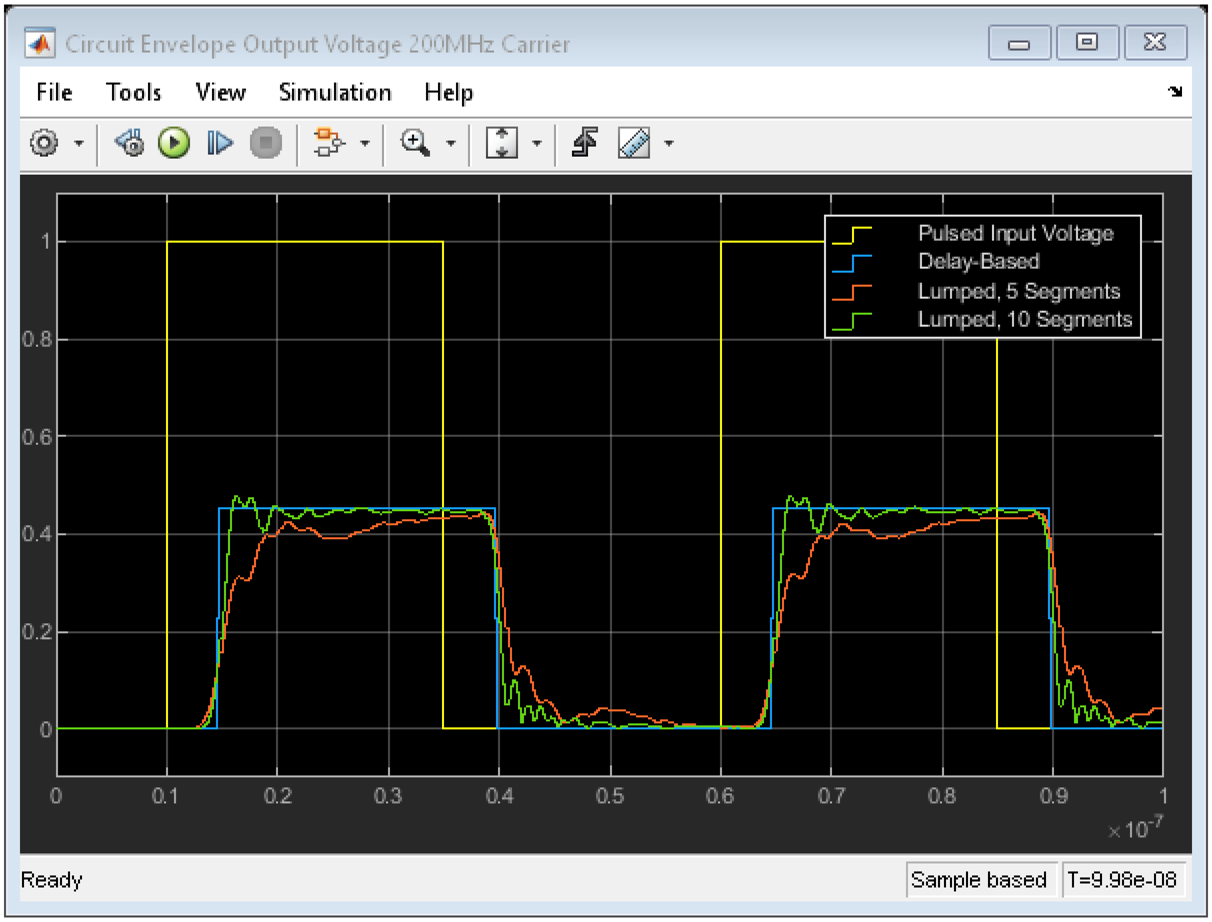

Lumped Elements
Input Signal
Circuit Envelope
RF
Pulsed Input Voltage
SL
RF
SimRF Inport
SimRF Parameters
RF
Rsrc
Rsrc5
Rsrc10
Delay-based Lossy
Transmission Line Delay-Based Lossy
Lumped-L
Transmission Line Lumped-Element (5 Segments)
Lumped-L
Transmission Line Lumped-Element (10 Segments)
Zload
Zload5
Zload10
Gnd
Gnd1
Probe
RF
SimRF Outport
SimRF Outport (5 Segments)
SimRF Outport (10 Segments)
Delay-Based
Lumped, 5 Segments
Lumped, 10 Segments
Circuit Envelope Output Voltage 200MHz Carrier
Copyright 2011 The MathWorks, Inc.

Running the Lumped Element Transmission Line

1. Type `open_system('simrf_xline_ll')` at the Command Window prompt.
2. Select **Simulation** > **Run**.

The following graph shows how the number of lumped element segments affects the output. Speed and accuracy must be balanced when using the lumped-element transmission line block.

```
open_system([model_ll '/Circuit Envelope Output Voltage 200MHz Carrier']);
sim(model_ll);
```



Cleaning Up

Close the model and remove workspace variables.

```
close(get(haxis,'Parent'))
clear haxis hline;
bdclose({model_pb model_ce model_seg model_ll});
clear SCE_Data SPB_Data ST_Data ST_Data3 SLL_Data;
clear model_pb model_ce model_seg model_ll;
```

References

Sussman-Fort and Hantgan, *SPICE Implementation of Lossy Transmission Line and Schottky Diode Models*. IEEE Transactions on Microwave Theory and Techniques, Vol. 36, No. 1, January 1988

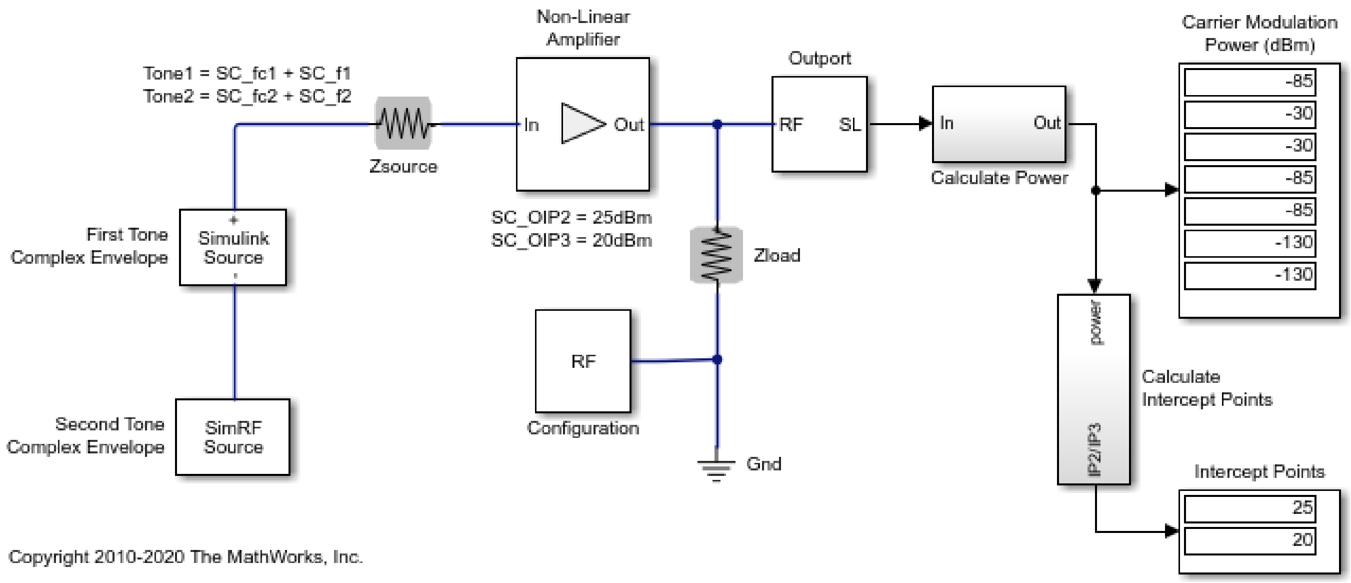
True Kenneth M, *Data Transmission Lines and Their Characteristics*. Application Note 806, April 1992

See Also

"Compare Time and Frequency Domain Simulation Options for S-parameters" on page 9-42

Validating IP2/IP3 Using Complex Signals

This example shows how to use the RF Blockset™ Circuit Envelope library to run a two-tone experiment that measures the second- and third-order intercept points of an amplifier. The model computes the intercept points of the amplifier from the modulated signal power measured on each carrier, verifying the behavior of the RF Blockset system. These values are confirmed using the RF Budget Analyzer app and a measurement testbench.



System Architecture

The system consists of:

- Two complex voltage sources connected in series. The first voltage source is modeled with Simulink® blocks, and the second with blocks from the RF Blockset circuit envelope library. In the Simulink Source subsystem, two series Sine Wave blocks model in-phase and quadrature components of the first tone. An Inport block assigns the Simulink signal to the carrier `fc1`. In the RF Blockset Source subsystem, two series Sinusoid blocks model in-phase and quadrature voltage signals that modulate the carrier `fc2`.
- A resistor modeling the voltage source impedance.
- An amplifier with input impedance, output impedance specified in the Main tab; output IP2, and output IP3 specified in the Nonlinearity tab.
- An Outport block that probes the output voltage of the amplifier across a shunt Resistor block. The ordering of the output signals is determined by the ordering of the carriers specified in the Outport block dialog.
- A subsystem to compute running rms power levels at the input, IP2 and IP3 frequencies.
- A subsystem to compute IP2 and IP3 intercept points [1].

The example model defines variables for block parameters using a callback function. To access model callbacks, select **MODELING** > **Model Settings** > **Model Properties** and click the Callbacks tab in the Model Properties window.

Running the Example

1 Type `open_system('simrfV2_carriers')` at the Command Window prompt.
2 Select **Simulation** > **Run**.

Output power and amplifier output intercept points are displayed on the right side of the model. The Calculate Power subsystem computes the power in dBm of each intermodulation product using a running root-mean-square (RMS) average.

Modeling Nonlinear RF Blockset Components

To model nonlinearities in the RF Blockset circuit envelope environment:

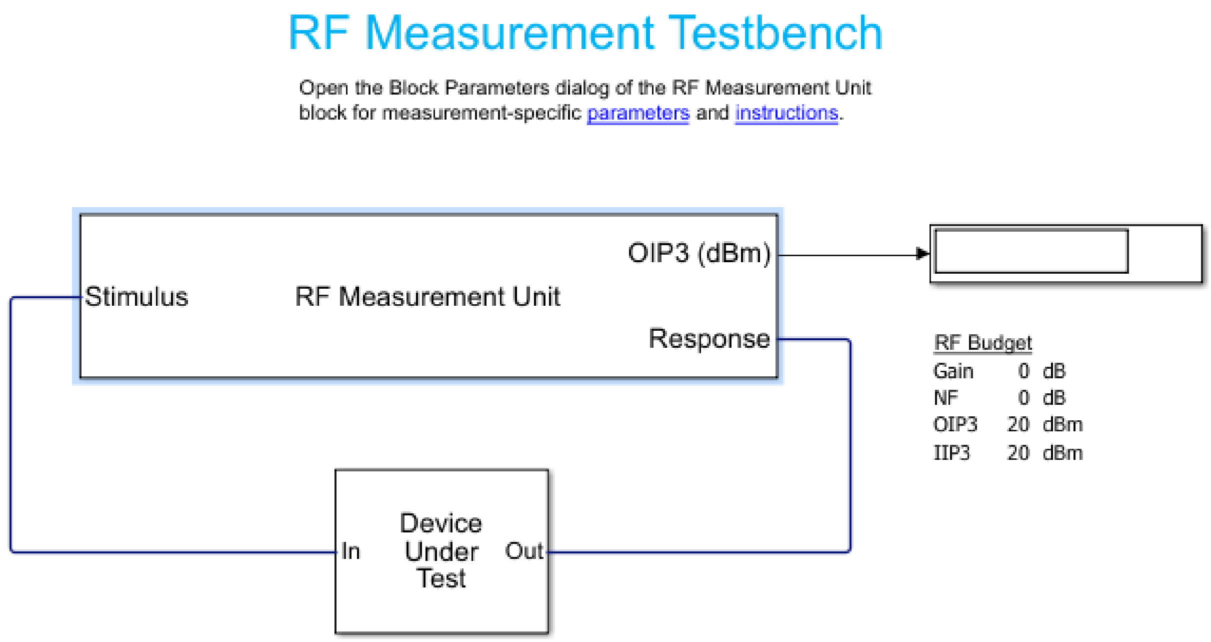
- Place an Amplifier or Mixer block in your model.
- Specify parameters that generate nonlinearities, such as **IP2** and **IP3**, taking care to specify the convention or specify a polynomial directly in the Nonlinearity tab of the block dialog.
- Specify any additional carrier frequencies for simulation in the Configuration block. In this example, the Configuration block specifies a total of twenty five frequencies : `fc1` and `fc2`, as the **Fundamental Tones** of the input signals; a **Harmonic Order** of 3 for each tone resulting in a complete set of second, third, fourth-order intermodulation products(second and third-order harmonic products included), and a partial set of fifth and sixth-order intermodulation products.

In order to calculate the power level of each envelope, the measured voltage signals are scaled with the inverse of the square root of the characteristic impedance. An additional scaling of `1/sqrt(2)` in the Calculate Power subsystem normalizes the complex-valued output signal.

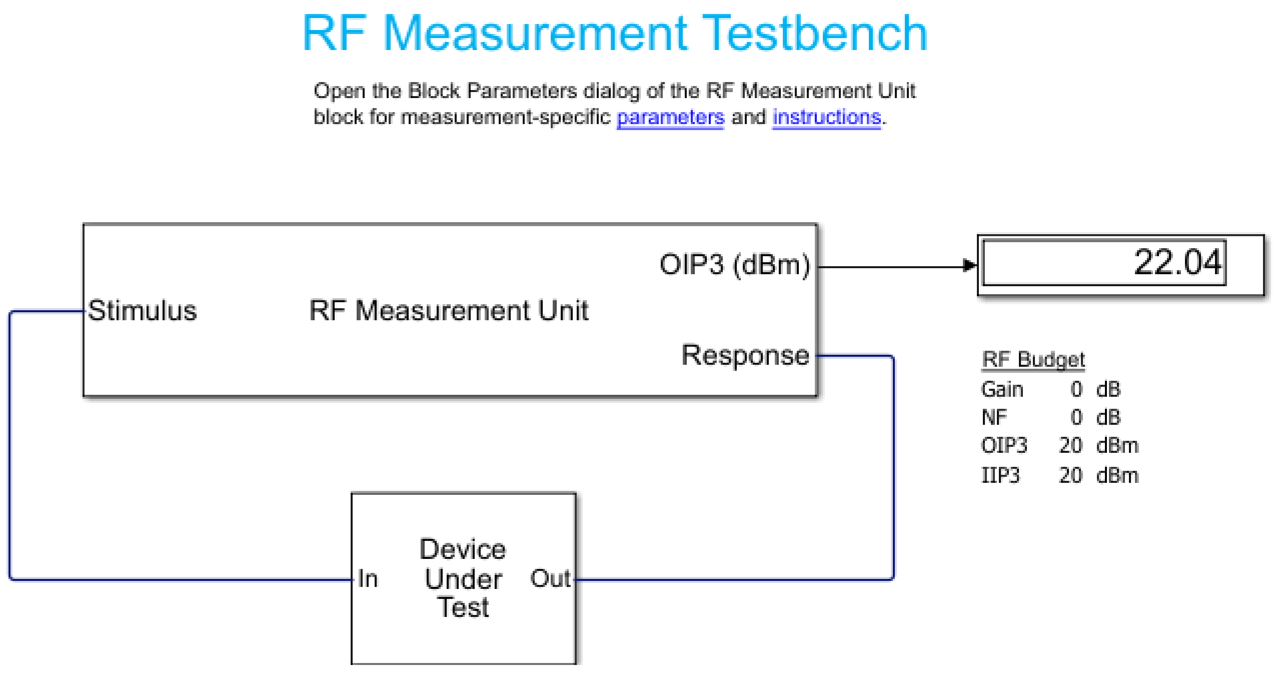
Validate Using Measurement Testbench

The same measurements can be performed using the RF Budget Analyzer app to automatically generate the model and testbench.

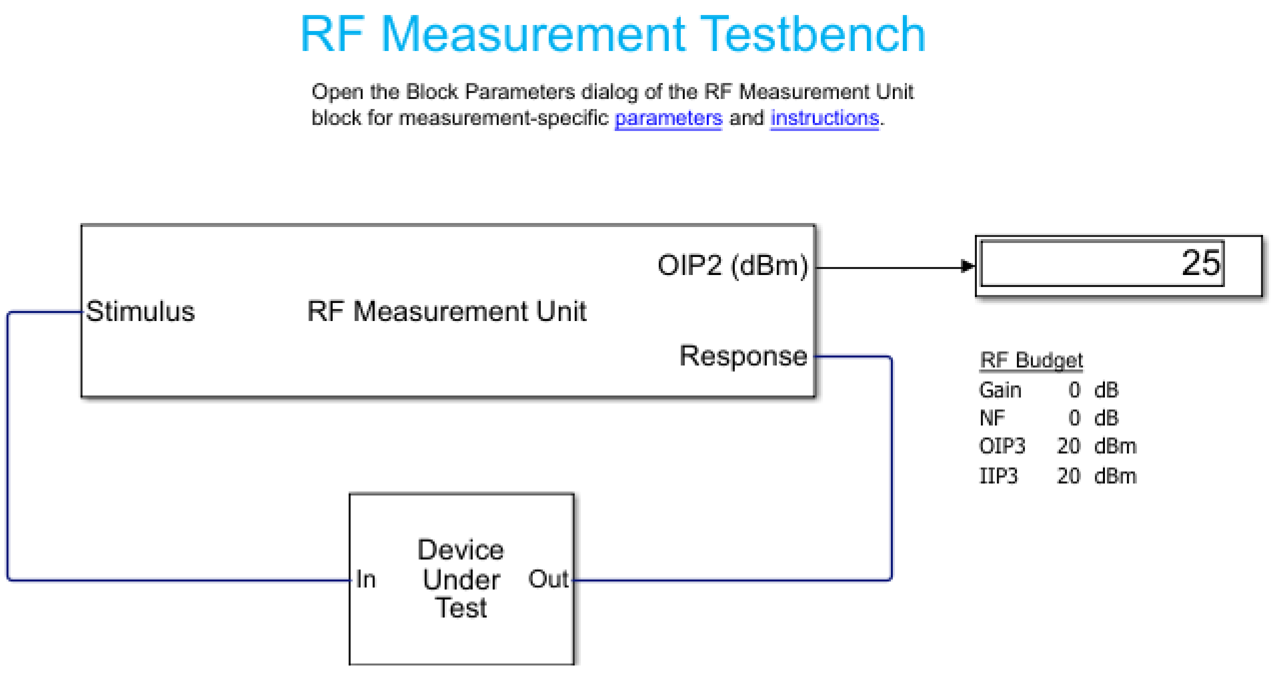
- Open the RF Budget Analyzer app and specify an amplifier.
- Define its IP3 (IP2 cannot be specified at this time).
- Generate the measurement testbench.



- Open Device Under Test to reveal the amplifier. Specify the IP2 value.
- Disable the noise to make an accurate measurement for IP2 and IP3. (Use the RF Measurement Unit dialog box).
- Run the simulation and measure IP3.



- Change the quantity to be verified to IP2 and rerun the simulation.



If you look under the mask of the testbench, you will find the logic to measure IP2 and IP3. This methodology is very similar to what is described in the initial model.

Reference

1 Kundert, Ken. "Accurate and Rapid Measurement of IP2 and IP3." *The Designers Guide Community*, Version 1b, May 22, 2002.

See Also

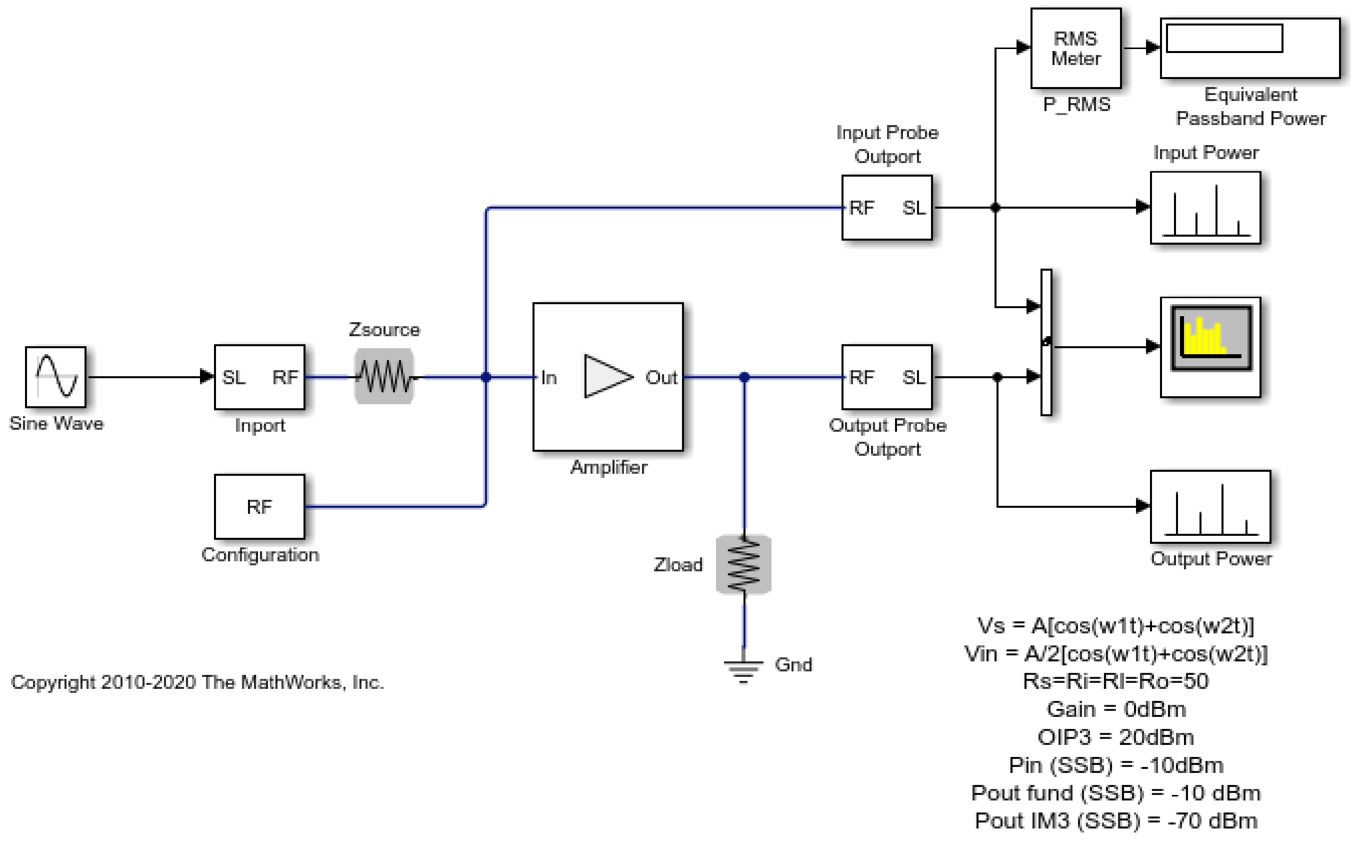
IIP2 Testbench | IIP3 Testbench

Related Topics

“Two-Tone Envelope Analysis Using Real Signals” on page 9-63

Two-Tone Envelope Analysis Using Real Signals

This example shows how to use the RF Blockset™ Circuit Envelope library to test intermodulation distortion of an amplifier using two-carrier envelope analysis.



System Architecture

The system consists of:

- A Simulink® sinusoidal input with frequency `(f1-f2)/2`, and an Inport that assigns the input modulation to the carrier `(f1+f2)/2`. This formulation is equivalent to the sum of two sinusoids with frequency `f1` and `f2`, according to the sum-product formula for cosines:

$$\cos 2\pi f_1 t + \cos 2\pi f_2 t = 2\cos\left(2\pi\frac{f_1+f_2}{2}t\right)\cos\left(2\pi\frac{f_1-f_2}{2}t\right)$$

- An amplifier with specified 0 dB linear gain and -20 dB OIP3. For an input signal u, the amplifier computes the output v according to the polynomial

$$v(t) = 2u(t) - \frac{4}{15}u^3(t)$$

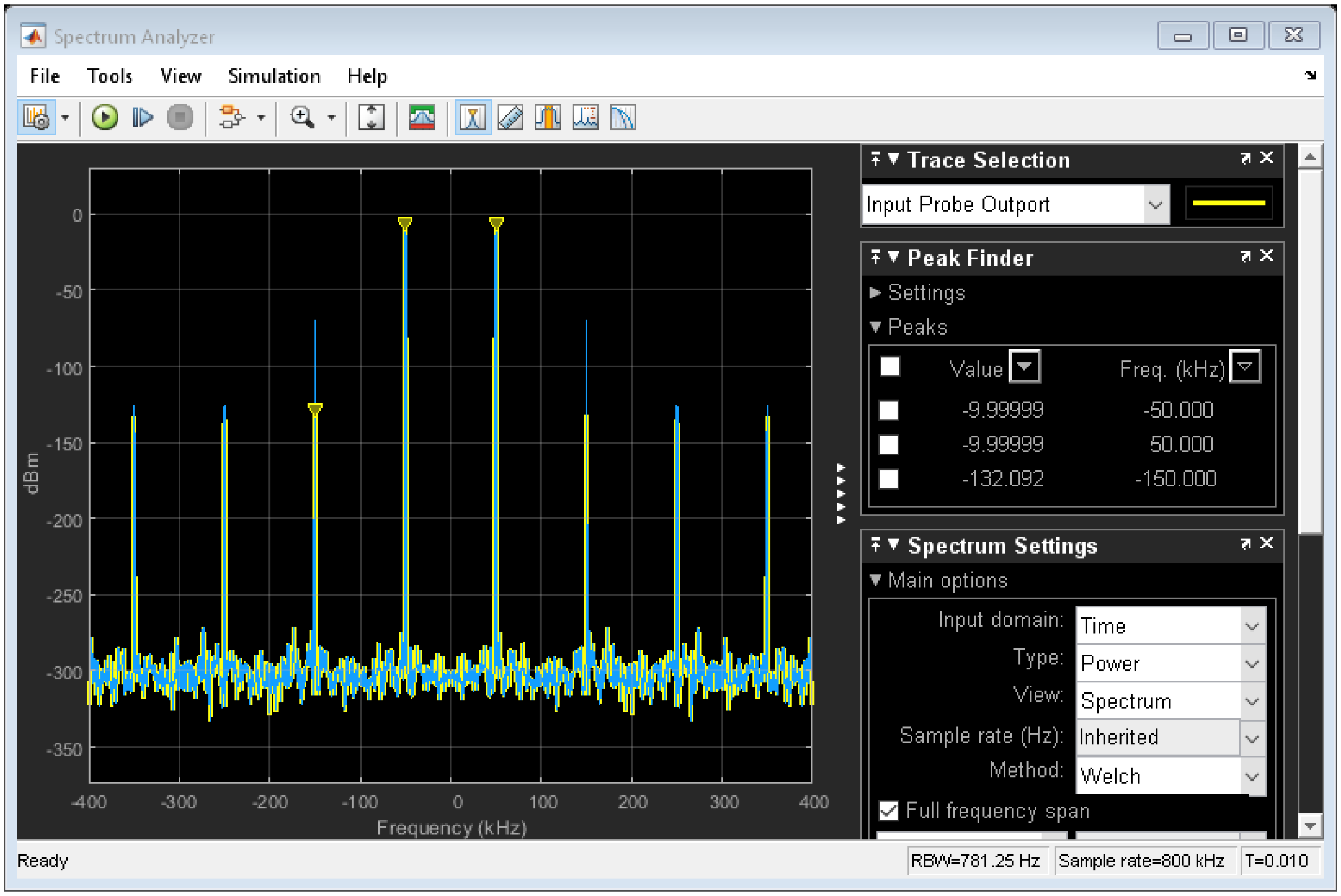
- Two Outport blocks that probe the input and output voltages (across shunt resistors) at the carrier frequency.

- A spectrum analyzer

The example model defines variables for block parameters using a callback function. To access model callbacks, select **MODELING** > **Model Settings** > **Model Properties** and click the Callbacks tab in the Model Properties window.

Running the Example

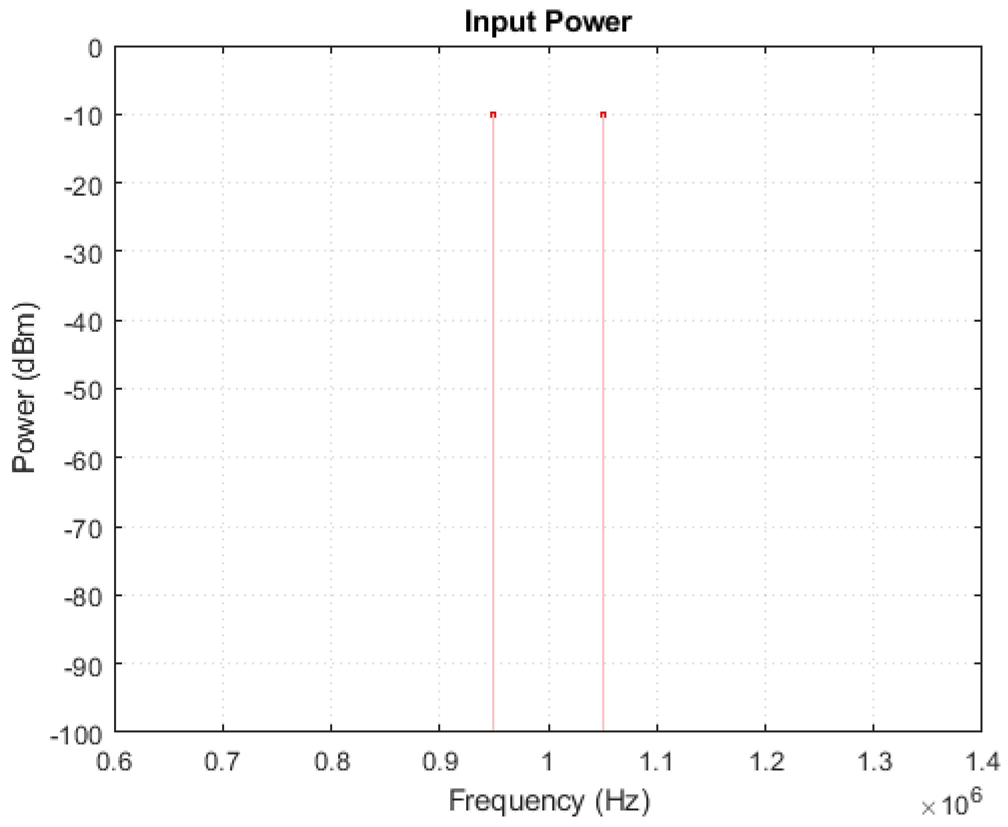
1. Type `open_system('simrfV2_power_imd')` at the Command Window prompt.
2. Select **Simulation** > **Run**.

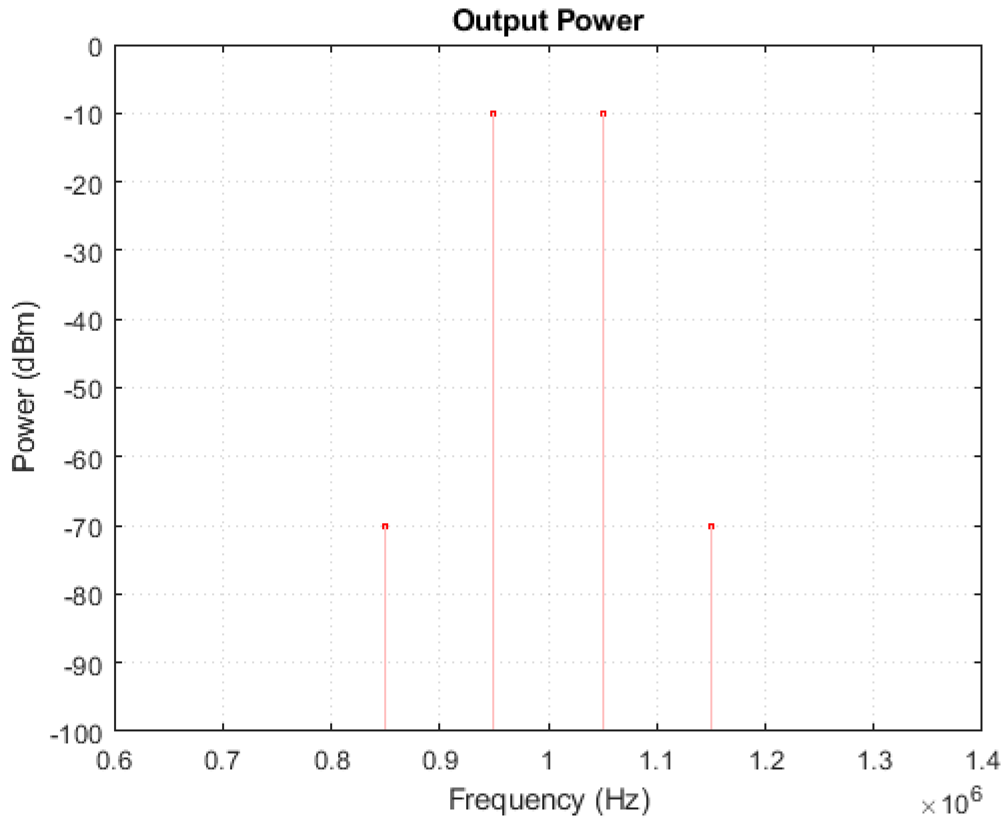


	Value	Freq. (kHz)
	-9.99999	-50.000
	-9.99999	50.000
	-132.092	-150.000

Input domain: Time
Type: Power
View: Spectrum
Sample rate (Hz): Inherited
Method: Welch

RBW=781.25 Hz | Sample rate=800 kHz | T=0.010


Input Power
Power (dBm)
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
0.6
0.7
0.8
0.9
1
1.1
1.2
1.3
1.4
Frequency (Hz)
$\times 10^6$



The resulting output power spectrum, labeled 'Output Power', shows third-order intermodulation distortion. The Display block shows the power of the bandpass waveform, which is half of power of the envelope waveform.

The spectrum analyzer is set to probe for the Output Power. On comparing values, you will see that the spectrum analyzer matches the output power spectrum.

The input drives the nonlinear amplifier into compression, so the linear component of the output is attenuated. To simulate the amplifier in a linear region:

- Specify `Inf` for the IP3 parameter of the Amplifier block located in the Nonlinearity tab of the dialog, or
- Reduce the power of the input signal by decreasing the value of the Amplitude parameter of the Sine Wave block.

Performing Two-Tone Analysis Using Circuit Envelope Simulation

This example takes advantage of the properties of real signals--namely, the sum-product equivalence of sinusoids. To perform the same experiment on a different RF system:

1. Choose `f1` and `f2`, the frequencies of the test tones.
2. Use a Simulink Sine source and an Inport block to model modulation with frequency `(f1-f2)/2` on a carrier with frequency `(f1+f2)/2`. Alternatively, you can use a Sinusoid block from the RF Blockset Circuit Envelope library to specify both a modulation and a carrier simultaneously.
3. Specify `(f1+f2)/2` as one of your carrier frequencies in the Configuration block dialog.

4 Use an Outport block to probe the distorted signal.

References

Cripps, Steve C. *RF Power Amplifiers for Wireless Communications*. Artech House, Inc., 2006.

See Also

Amplifier | Configuration | Inport

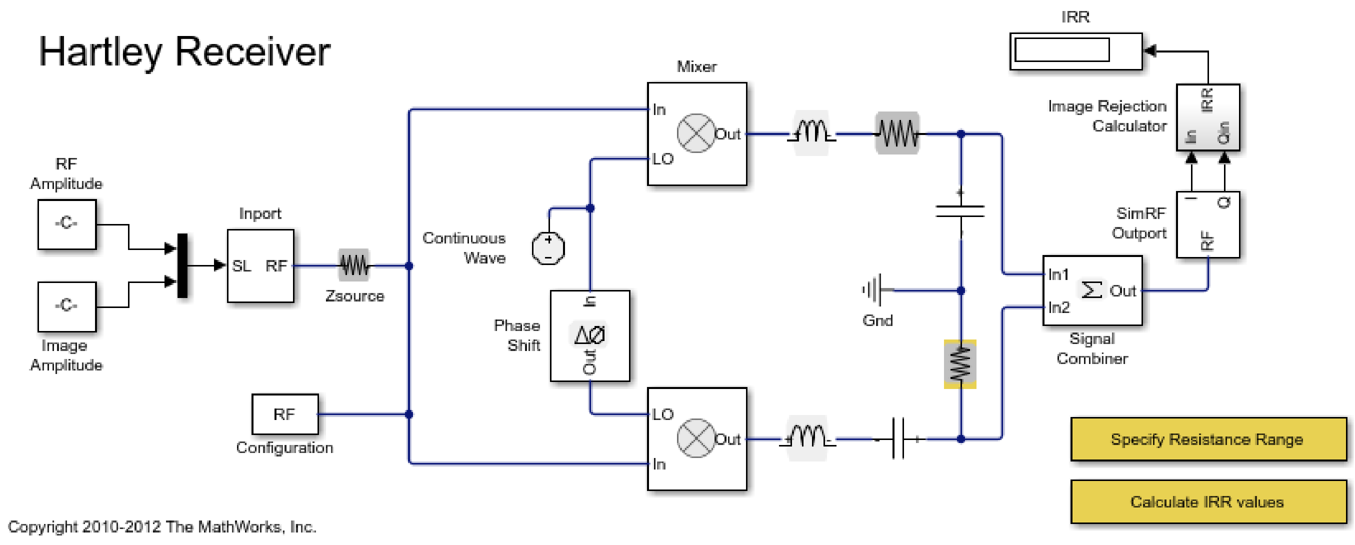
Related Topics

“Validating IP2/IP3 Using Complex Signals” on page 9-59

Measuring Image Rejection Ratio in Receivers

This example shows how to use the RF Blockset™ Circuit Envelope library to calculate the image rejection ratio (IRR) for high-side-injection in Weaver and Hartley receivers. The Weaver receiver shows the effect of phase offset on IRR, and the Hartley receiver shows a similar effect for resistor variation.

```
model1 = 'simrfV2_hartley';
open_system(model1);
```



System Architecture

The RF system consists of:

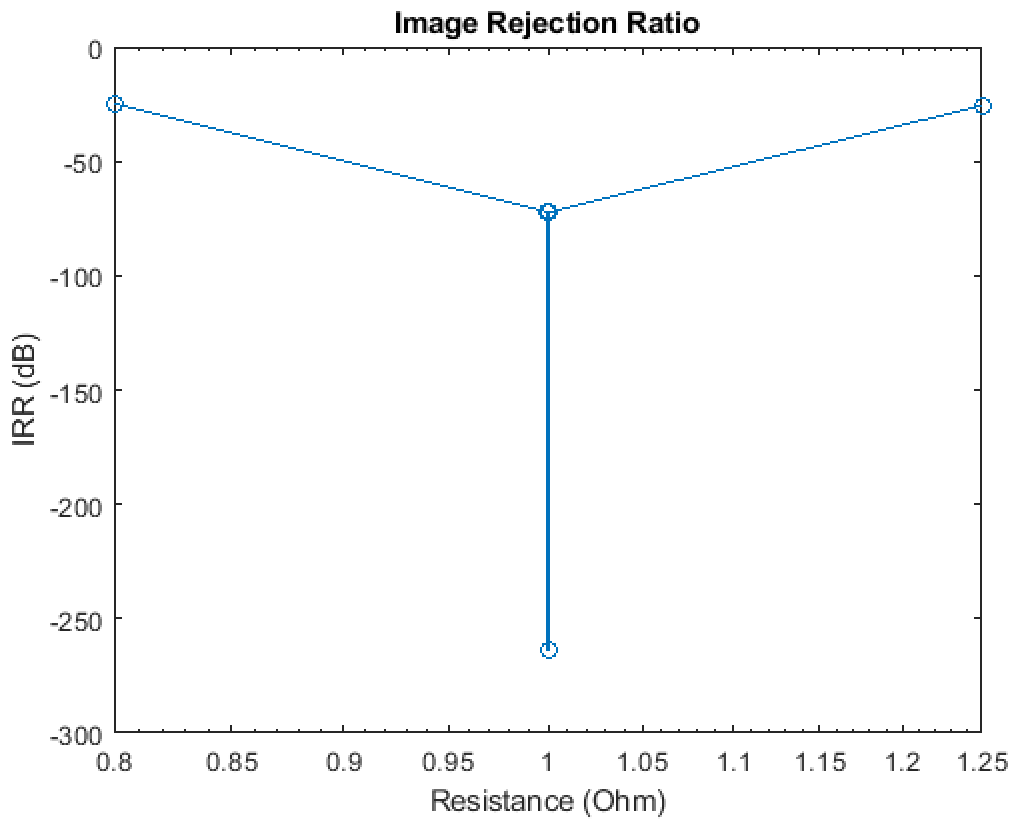
- An Inport block that assigns multiplexed outputs of the RF and Image, by using Simulink® Constant blocks, to the carriers `fc_RF` and `fc_IM` respectively. Real and imaginary values of the Constant blocks are matched to in-phase and quadrature carrier components.
- First stage mixers that mix the input signal with a local oscillator modeled by a Continuous Wave block with frequency `fc_LO`. The LO frequency is the average of the RF and image frequencies, so both signals are mixed down to the same frequency, `fc_IF`. The LO phase is shifted 90 degrees in one mixer relative to the other.
- The second stage of the Hartley uses a frequency independent RC-CR network to produce an additional 90 degree phase shift between the two signal paths, while the Weaver employs two additional mixers for channel selection.
- A Signal Combiner block that sums the voltage signals at its two inputs to yield the RF signal. If the Signal Combiner block is used to perform subtraction, the image can be obtained instead of the RF signal at its output. For low-side-injection, the Signal Combiner block needs to perform subtraction.
- The values of the in-phase and quadrature components of the RF and image signals are chosen to reduce the number of IRR calculations and facilitate reuse of the Image Rejection Calculator.

Simulating the Hartley Receiver

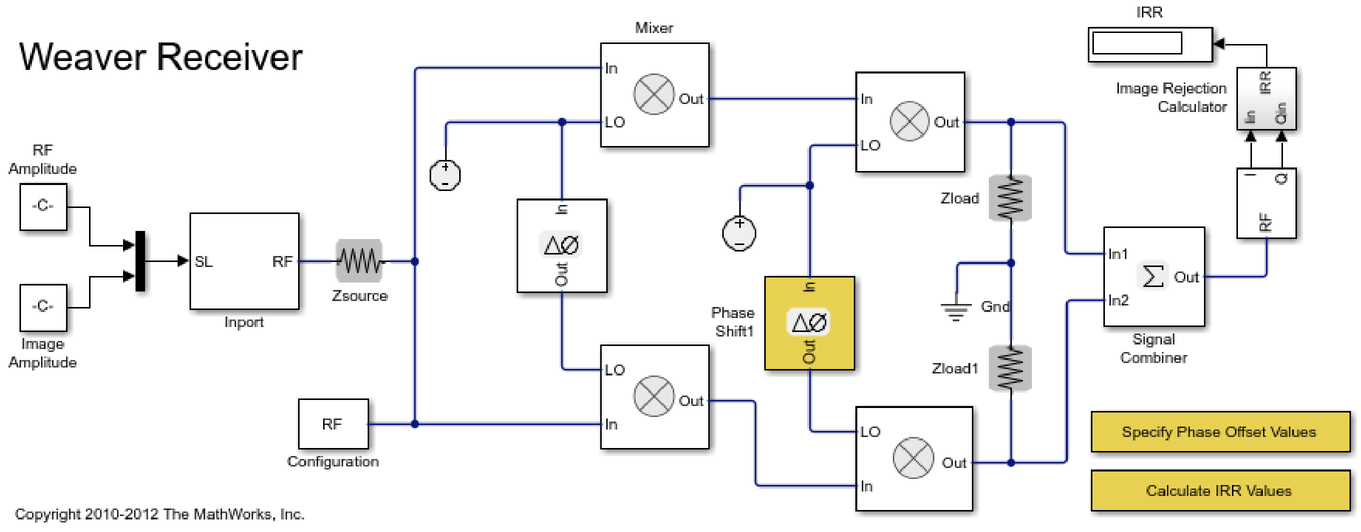
1. Type `open_system('simrfV2_hartley')` at the Command Window prompt.
2. Double-click 'Specify Resistance Range' and specify a set of resistance values for the highlighted resistor.
3. Double-click 'Calculate IRR values' to execute a script, `simrfV2_hartley_callback`, that simulates the model once for each specified resistance value and generates a plot.

The sensitivity of the architecture to the component variation is shown by simulating the system multiple times, varying the resistance of the highlighted Resistor block at each iteration. When the highlighted resistor has a resistance of 1 Ohm, the images sum to zero in the Signal Combiner block and the IRR is minus infinity.

```
evalc('simrfV2_hartley_callback');
```



```
bdclose(model1);
model2 = 'simrfV2_weaver';
open_system(model2);
```

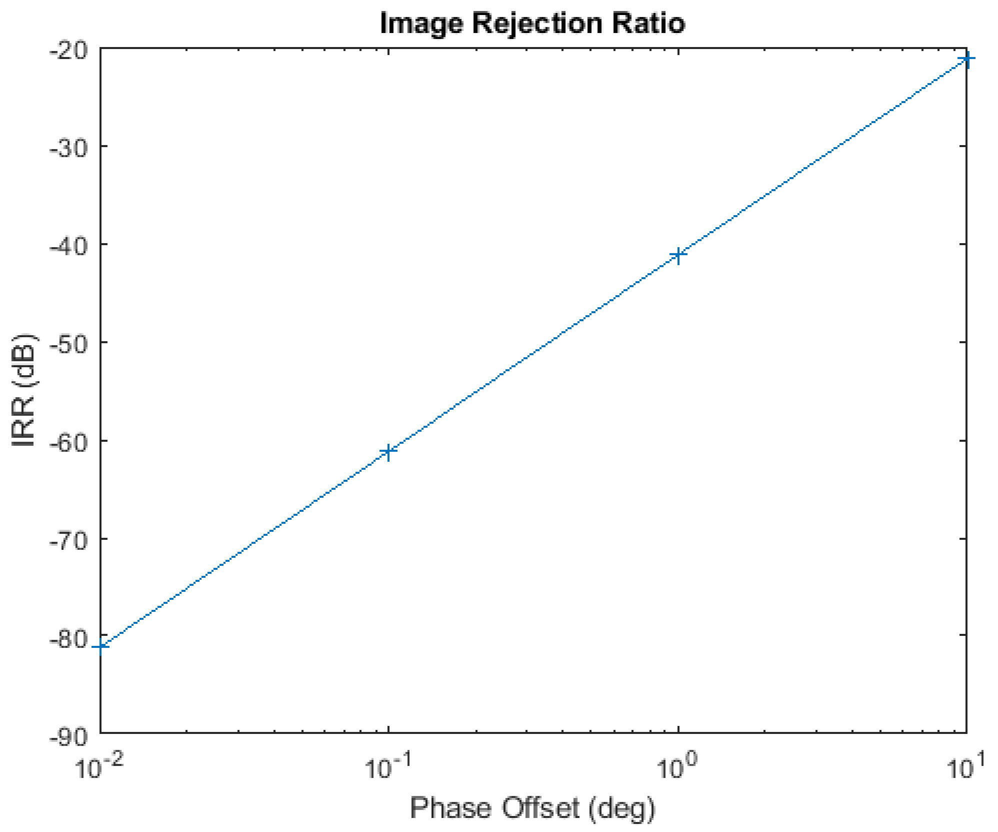


Simulating the Weaver Receiver

1. Type `open_system('simrfV2_weaver')` at the Command Window prompt.
2. Double-click 'Specify Phase Offset Values' and specify a set of phase offset values.
3. Double-click 'Calculate IRR values' to execute a script, `simrfV2_weaver_callback`, that simulates the model once for each specified offset and generates a plot.

The sensitivity of the architecture to LO phase offset is shown by simulating the system multiple times, varying the phase offset of the highlighted Phase Shift block at each iteration. When the phase offset of the highlighted Phase Shift block is zero, the images sum to zero in the Signal Combiner block and the IRR is minus infinity.

```
evalc('simrfV2_weaver_callback');
```



```
bdclose(model2)
```

See Also

Mixer

Related Topics

"Measuring Image Rejection Ratio in Receivers" on page 9-68

Executable Specification of a Direct Conversion Receiver

This example shows how to use the RF Blockset™ Circuit Envelope library to simulate the sensitivity performance of a direct conversion architecture with the following RF impairments:

- Component noise
- LO-RF isolation
- Interference from blocker signals
- Local oscillator phase offset
- ADC dynamic range
- Component mismatch

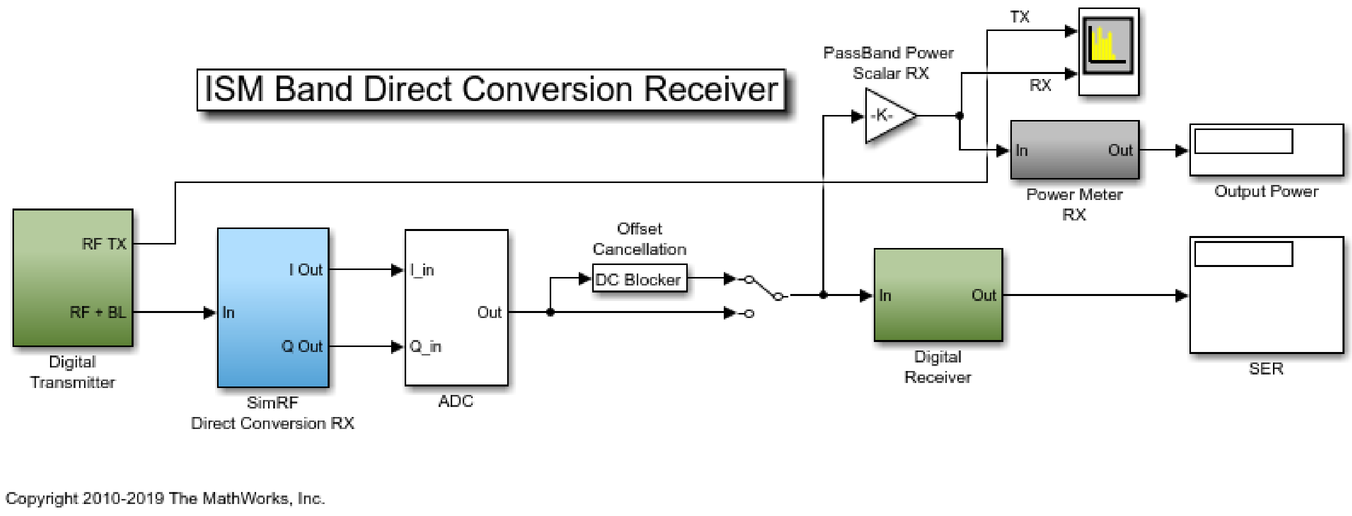
The RF portion of the model includes the explicit specification of gain, noise figure, IP2 and IP3, input \output impedance, and LO phase offset. The transmitter side of the RF interface includes modulation scheme, signal power, and blocker power. The carrier frequencies for the transmitted waveforms are specified in the Inport block. The baseband side includes the number of symbols and full scale range of the ADC.

System Architecture:

This system model illustrates the design and simulation of a Direct Conversion ISM Band Receiver. The model is comprised of blocks from RF Blockset, Communications Toolbox™, DSP System Toolbox™, and Simulink® libraries. The primary subsystems in the model include a digital transmitter, an RF receiver, an ADC, a DC offset correction, and a digital receiver. Blocks and plotted signals are color coded:

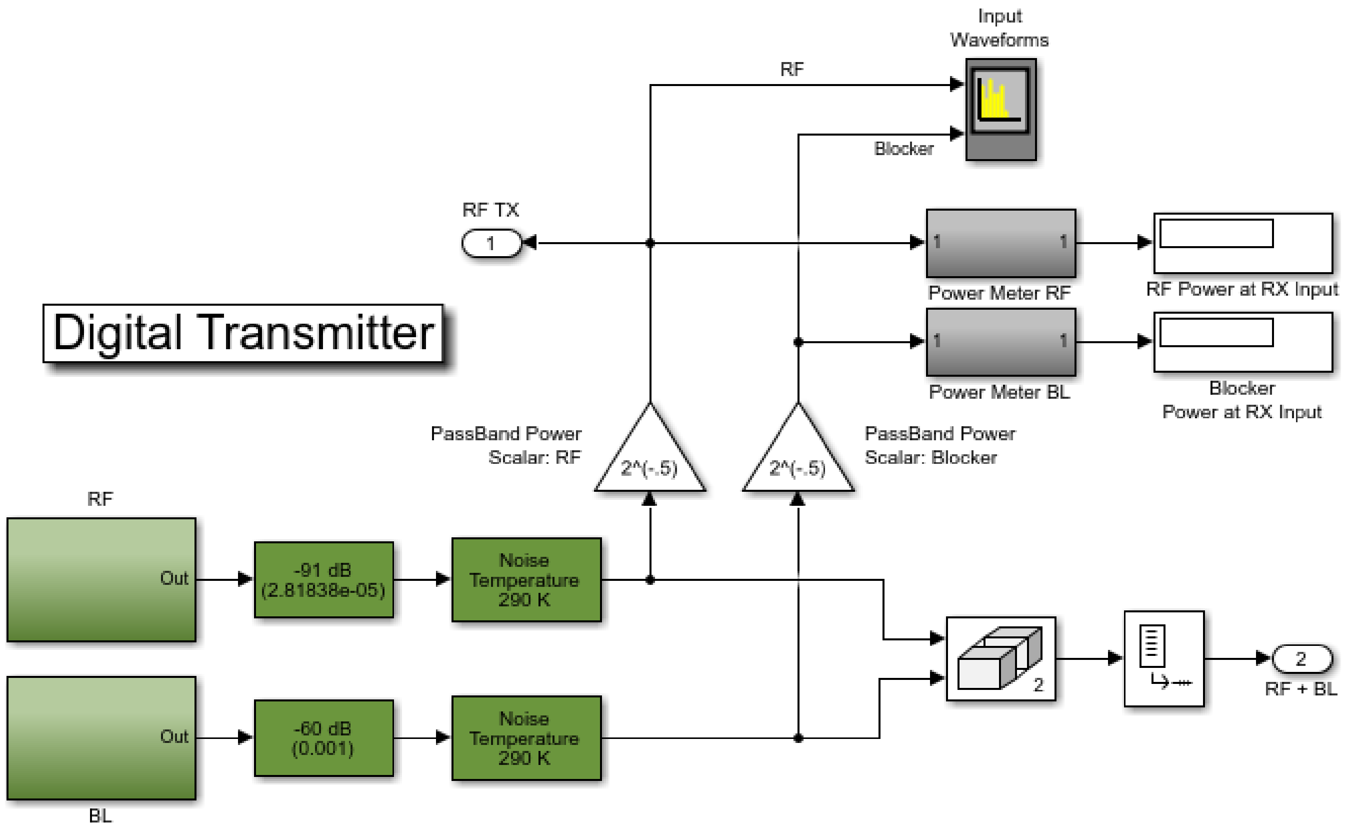
- RF Blockset: Light Blue
- Communications Toolbox: Green
- DSP System Toolbox: Grey
- Simulink: White

```
model = 'simrfV2_direct_conv';
open_system(model)
```

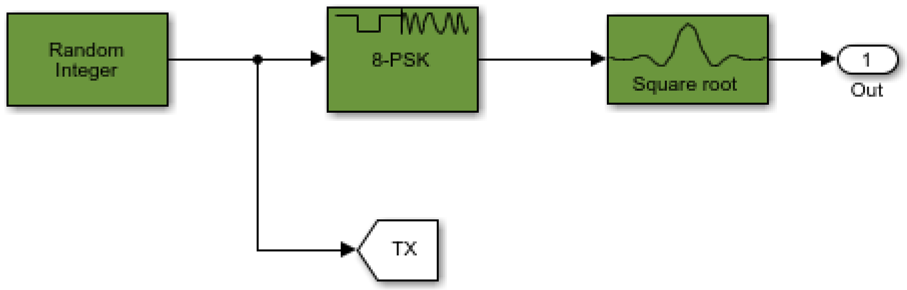


The digital transmitter consists of two 8-PSK modulated waveforms, a target waveform and an interfering waveform. The waveforms are scaled by 1/sqrt(2) for passband-waveform power measurement and spectrum visualization.

```
open_system([model '/Digital Transmitter'])
```

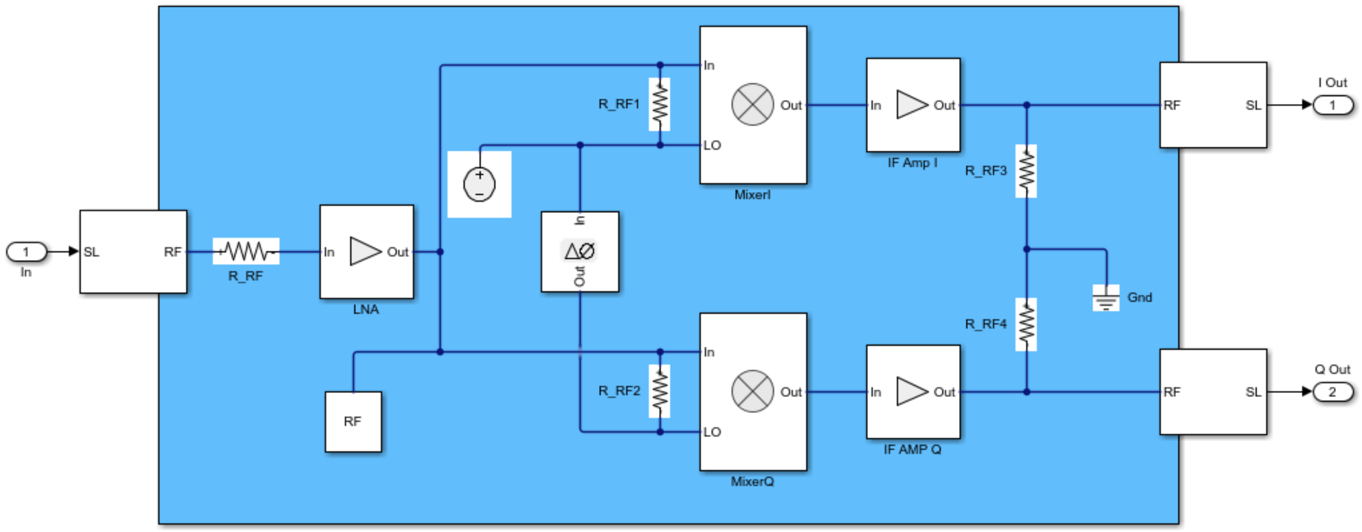


```
open_system([model '/Digital Transmitter/RF'])
```



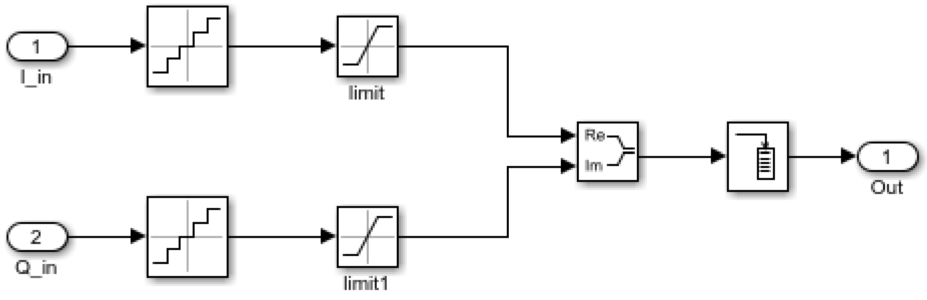
The direct-conversion RF receiver has a frequency conversion stage and two gain stages. Resistors model input and output impedances of the RF system as well as the isolation between the LO and RF ports of the mixers. Each of the blocks captures RF impairments relevant to this design. Each of the nonlinear blocks is specified by noise figure. The LNA non-linearity is specified by IP3, and the nonlinearity in the IF amplifiers are specified by both IP2 and IP3. The mixer nonlinearity is specified by IP2. A single LO and a phase shift block provide the cosine and sine terms to the I and Q branches, respectively. To model a thermal noise floor in the RF Blockset environment, the Temperature parameter in the Configuration block specifies a noise temperature of 290.0 K.

```
open_system([model '/SimRF Direct Conversion RX'])
```



The ADC uses an N-bit quantizer followed by a saturation block to model the full-scale range. Hence, the ADC properly models the system quantization noise floor.

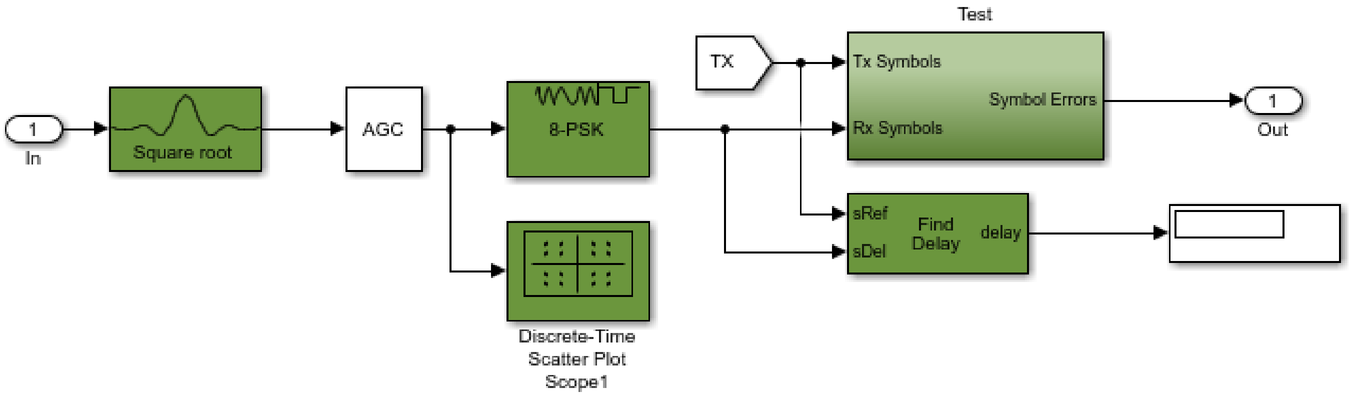
```
open_system([model '/ADC'], 'Force')
```



The DC offset cancellation block employs an IIR algorithm for estimating offset. This DC offset correction is enable or disable during simulation using the manual switch that follows the offset block.

The digital receiver applies a matched filter to the received waveform followed by an AGC function, and demodulates the waveform for symbol-error-rate calculation.

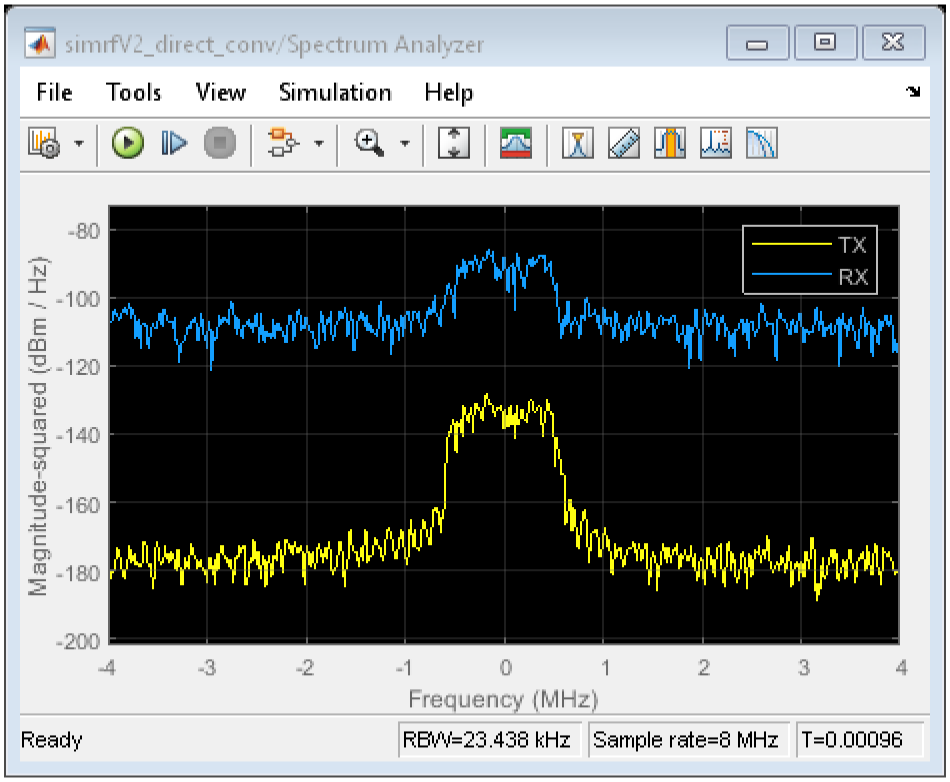
```
open_system([model '/Digital Receiver'])
```

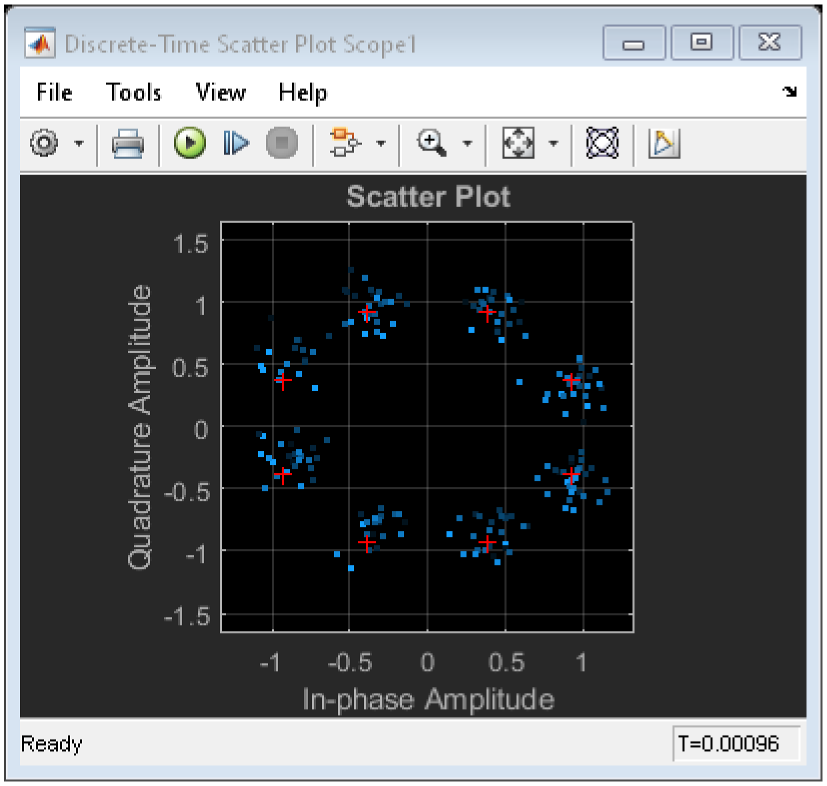


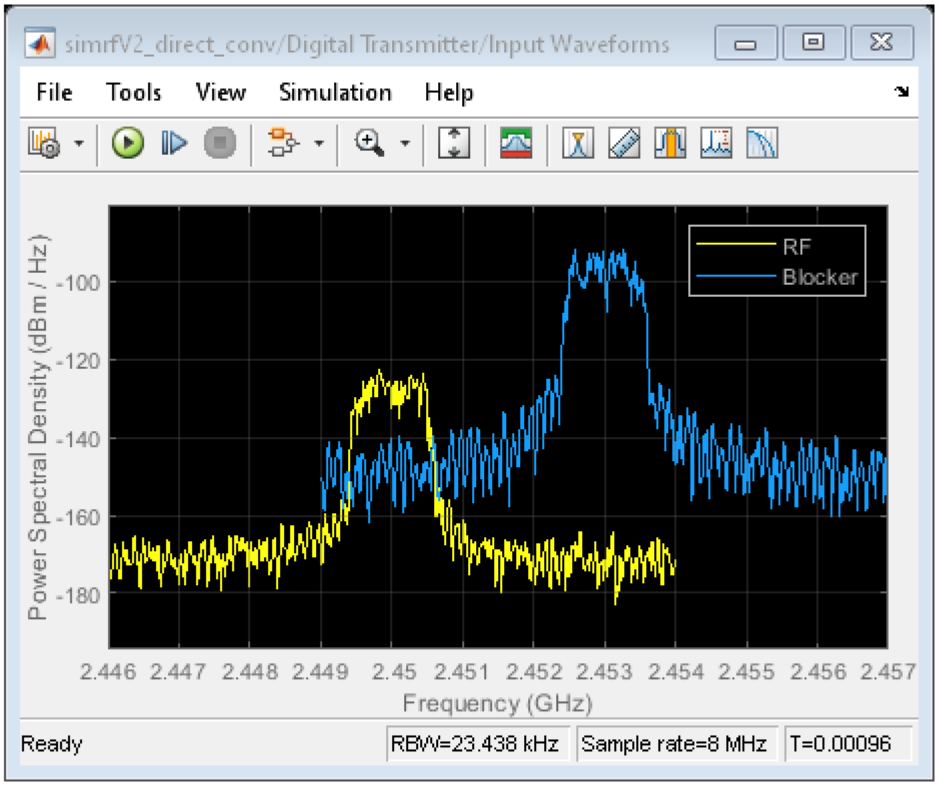
Running the Example

Running the example simulates a design that meets an uncoded 0.5% SER specification. Modifications to the signal power levels and component specifications in the receiver and ADC have a direct impact on the receiver performance. The manual switch in the design enables the user switch the offset correction subsystem in and out to visualize the DC offset effect associated with RF-LO isolation.

```
sim(model,'stoptime','1e-3');
```



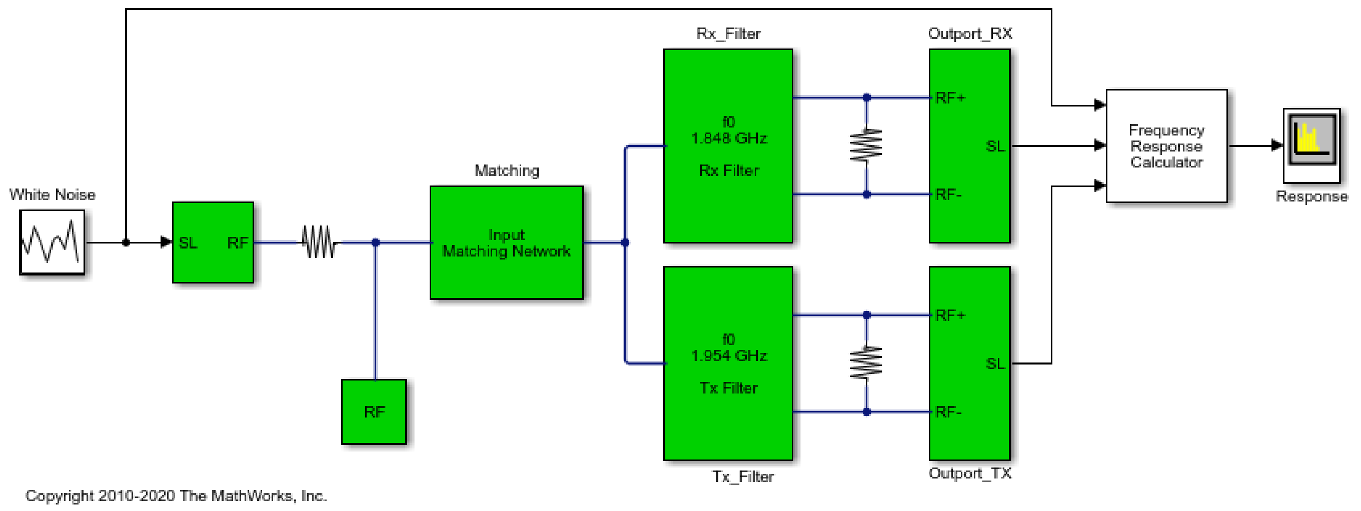

Discrete-Time Scatter Plot Scope1
File
Tools
View
Help
Scatter Plot
Quadrature Amplitude
In-phase Amplitude
1.5
1
0.5
0
-0.5
-1
-1.5
-1
-0.5
0
0.5
1
Ready
T=0.00096


simrfV2_direct_conv/Digital Transmitter/Input Waveforms
File
Tools
View
Simulation
Help
RF
Blocker
Power Spectral Density (dBm / Hz)
-100
-120
-140
-160
-180
2.446
2.447
2.448
2.449
2.45
2.451
2.452
2.453
2.454
2.455
2.456
2.457
Frequency (GHz)
Ready
RBW=23.438 kHz
Sample rate=8 MHz
T=0.00096

```
bdclose(model)
clear model
```

Frequency Response of an RF Transmit/Receive Duplex Filter

This example shows how to use blocks from the RF Blockset™ Circuit Envelope library to simulate a transmit/receive duplex filter and calculate frequency response curves from a broadband white-noise input. Blocks from the DSP System Toolbox™ libraries generate the input signal, process the output signal, and display the results.

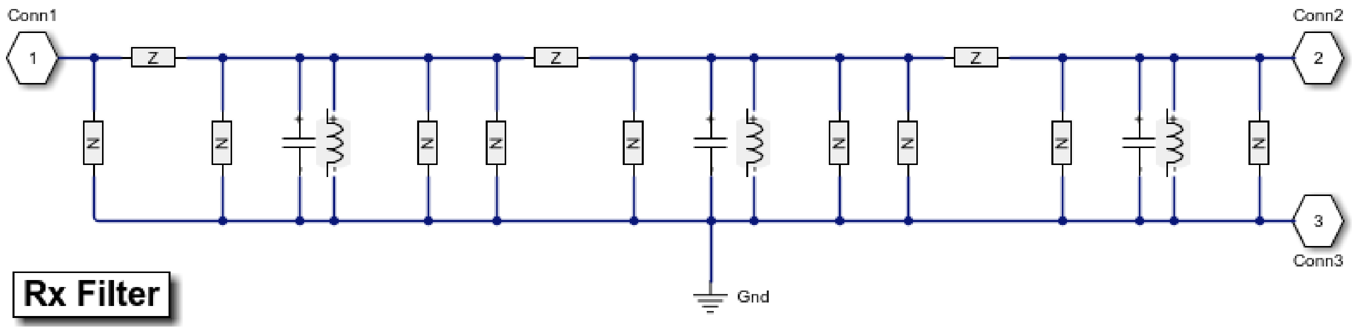


System Architecture

The system consists of:

- A DSP System Toolbox Random Source block, which generates broadband white noise.
- Inport and Resistor blocks from the RF Blockset circuit envelope library, which model a controlled voltage source with internal impedance.
- An input matching network, which maximizes the power of the signal received by the filters.
- Duplexed transmit and receive filters, composed of circuit envelope blocks from the RF Blockset Elements sublibrary. The center frequencies of the filters are 1.954 GHz and 1.848 GHz, and the bandwidths are 27 MHz and 18 MHz, respectively. The receive filter is shown in the figure below.
- Two Outports acting as voltage sensors, measuring the voltage across two load resistors.
- The Frequency Response Calculator subsystem, which computes the voltage transfer functions.
- A DSP System Toolbox Spectrum Analyzer, which displays the frequency response curves.

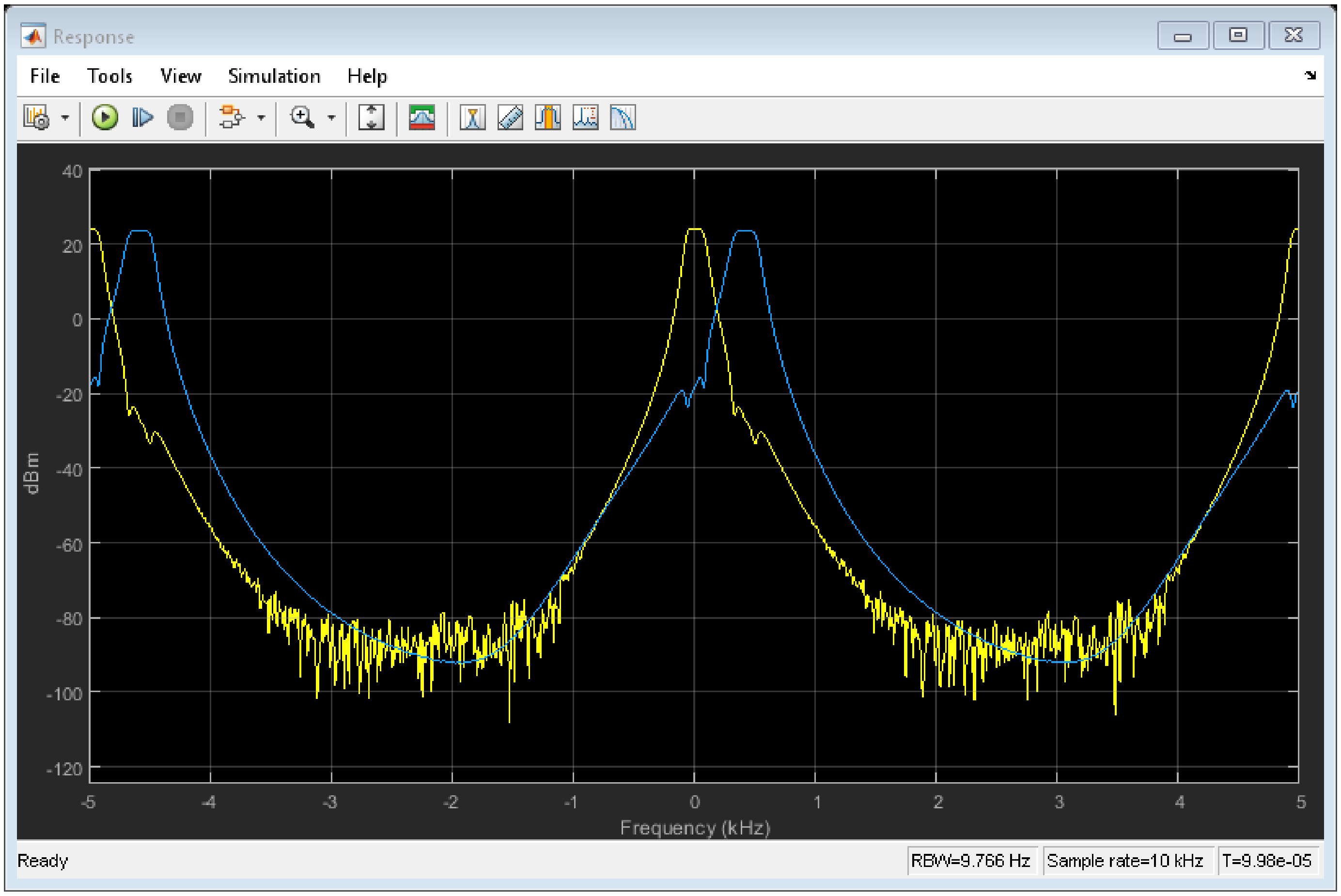
Block parameters in this example are specified by variables stored in the model workspace. To alter the model workspace, select **Modeling** > **Model Explorer** and click on the Model Workspace node for this model in the Model Hierarchy pane.



Running the Example

1. Type `open_system('simrfV2_duplexer')` at the Command Window prompt.
2. Select **Simulation** > **Run**.

The Spectrum Analyzer displays the envelope voltage transfer functions of the two filters. Duplexing is evident in the frequency response of the two filters. The frequency values along the x-axis are relative to the RF carrier. To compute the absolute frequencies, add the value of the **Carrier frequencies** parameter of the Inport block to the frequencies shown.



Calculating a Transfer Function Using a Broadband White Noise Source

The Frequency Response Calculator subsystem processes the Simulink® signals converted from the RF Blockset environment at the Outport blocks. Within this subsystem, two Compute Transfer Function subsystems correlate the outputs to the input signal.

The Compute Transfer Function subsystems operate on discrete signals. To configure the RF Blockset circuit envelope environment for discrete-time simulation:

- In the Configuration block dialog, select `Auto` from the **Solver** drop-down menu, and specify the **Step size** parameter. Within the RF Blockset circuit envelope environment, the local solver overrides the solver options specified in the Configuration Parameters dialog box.
- Use a continuous source, or use a discrete source with a sample time equal to the value specified in the Configuration block. In this example, the **Sample mode** parameter of the Random Source block is `Continuous`.

See Also

Inductor | Capacitor | Z (Impedance)

Related Topics

"Architectural Design of a Low IF Receiver System" on page 9-193

Digital Predistortion to Compensate for Power Amplifier Nonlinearities

This demo shows an example of using digital predistortion (DPD) in a transmitter to offset the effects of nonlinearities in a power amplifier. In this case, we use a model of a power amplifier that was obtained using the “Power Amplifier Characterization” (Communications Toolbox). In the first simulation, the RF transmitter sends two tones. In the second simulation, the RF transmitter sends a 5G-like OFDM waveform with 100 MHz bandwidth.

DPD with Two Sinusoidal Test Signals

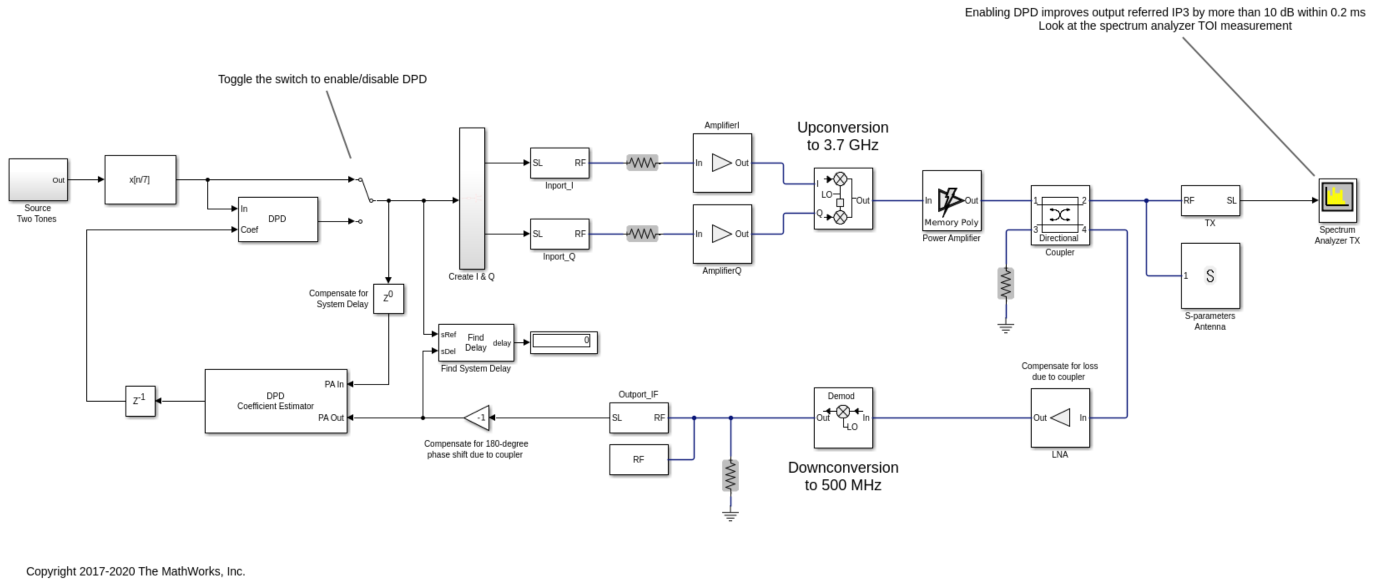
Open the Simulink RF Blockset model including the PA model and the adaptive DPD algorithm [2]:

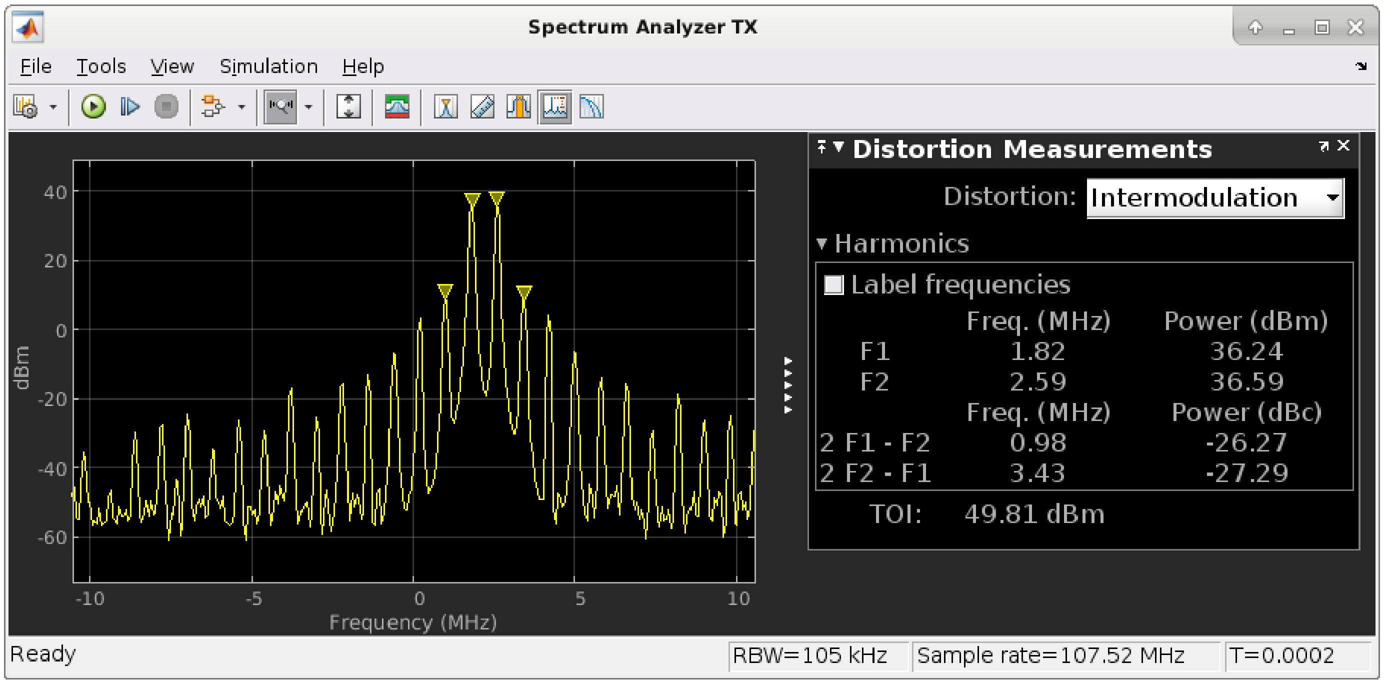
System-level model PA + DPD with two tones

The model includes a two-tone signal generator that is used for testing the output-referred third-order intercept point of the system. The model includes upconversion to RF frequency using an I-Q modulator, the PA model, a coupler to sniff the output of the PA, and an S-parameter block representing the antenna loading effect. The receiver chain performs downconversion to low intermediate frequency. Notice that the simulation bandwidth of this system is 107.52 MHz.

The model can be simulated without the DPD when the toggle switch is in the up position.

```
model = 'simrfV2_powamp_dpd';
open_system(model)
sim(model)
```


Enabling DPD improves output referred IP3 by more than 10 dB within 0.2 ms
Look at the spectrum analyzer TOI measurement
Toggle the switch to enable/disable DPD
Source Two Tones
x[n/7]
DPD
Compensate for System Delay
Create I & Q
Inport_I
Inport_Q
AmplifierI
AmplifierQ
Upconversion to 3.7 GHz
Memory Poly Power Amplifier
Directional Coupler
TX
S-parameters Antenna
Spectrum Analyzer TX
Find System Delay
DPD Coefficient Estimator
Compensate for 180-degree phase shift due to coupler
Outport_IF
Demod
Downconversion to 500 MHz
Compensate for loss due to coupler
LNA
Copyright 2017-2020 The MathWorks, Inc.



Distortion Measurements

Distortion: Intermodulation

Harmonics

Label frequencies

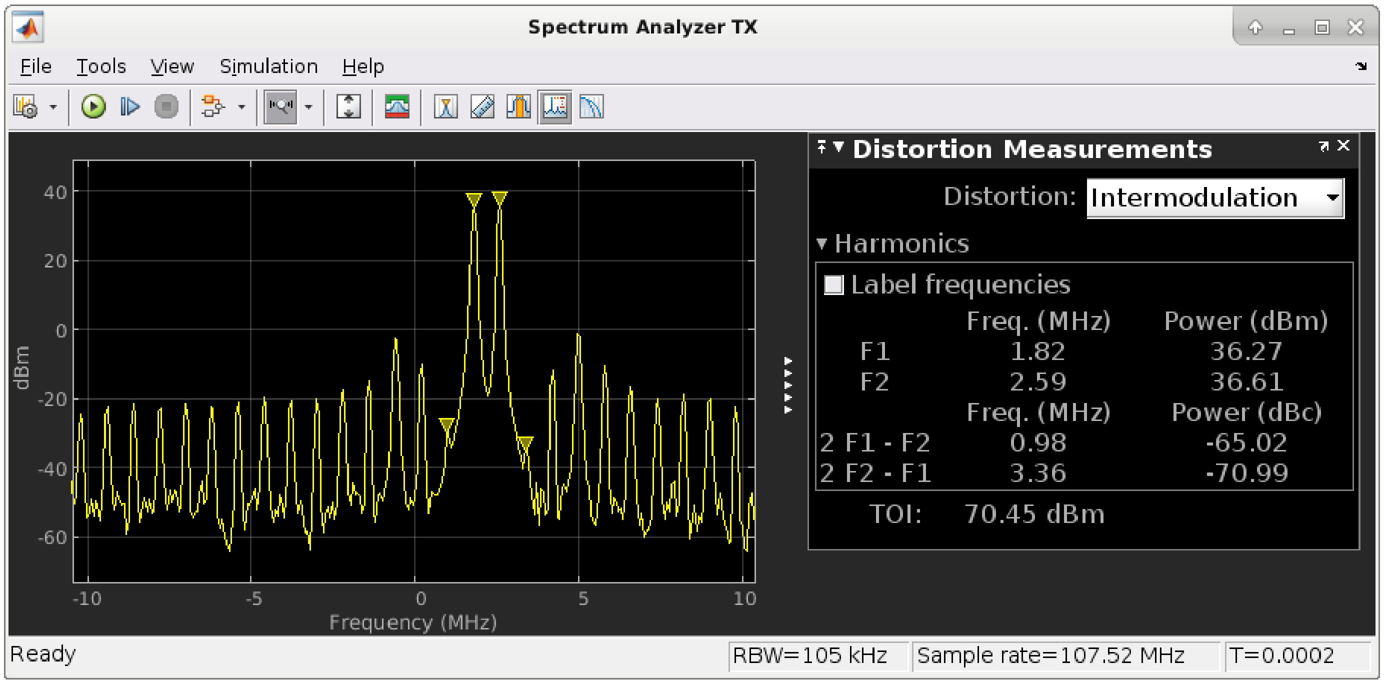
	Freq. (MHz)	Power (dBm)
F1	1.82	36.24
F2	2.59	36.59
	Freq. (MHz)	Power (dBc)
2 F1 - F2	0.98	-26.27
2 F2 - F1	3.43	-27.29

TOI: 49.81 dBm

RBW=105 kHz   Sample rate=107.52 MHz   T=0.0002

The manual switch is toggled to enable the DPD algorithm. When toggled, the TOI (third-order intercept point) is improved by more than 10 dB within 0.2 ms of simulation time. Inspect the distortion measurement in the Spectrum Analyzer to validate these results and see how the power of the harmonics is reduced thanks to the DPD linearization.

```
set_param([model '/Manual Switch'], 'action', '1')
sim(model)
```



Distortion Measurements

Distortion: Intermodulation

Harmonics

Label frequencies

	Freq. (MHz)	Power (dBm)
F1	1.82	36.27
F2	2.59	36.61
	Freq. (MHz)	Power (dBc)
2 F1 - F2	0.98	-65.02
2 F2 - F1	3.36	-70.99

TOI: 70.45 dBm

RBW=105 kHz   Sample rate=107.52 MHz   T=0.0002

By changing the degree and the memory depth defined in the DPD Coefficient Estimator block, you can find the most suitable tradeoff between performance and implementation cost.

To estimate the DPD coefficients correctly, the input signals to the DPD Coefficient Estimator block, PA In and PA Out, must be aligned in the time domain. This is verified by the Find Delay block which shows that the delay introduced by the RF system is 0. The estimator's desired amplitude gain is important to make sure that the total linear gain of the system does not change when the DPD is active. For this particular model, the gain has been estimated using separate techniques [2].

```
close_system(model,0)
close all; clear
```

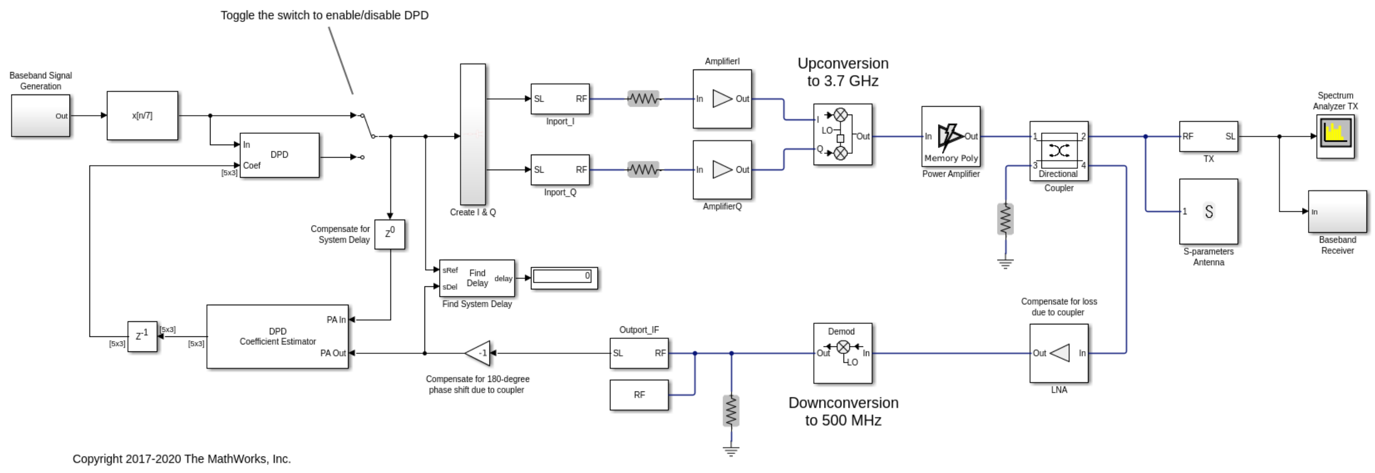
DPD with a 5G-like OFDM Waveform

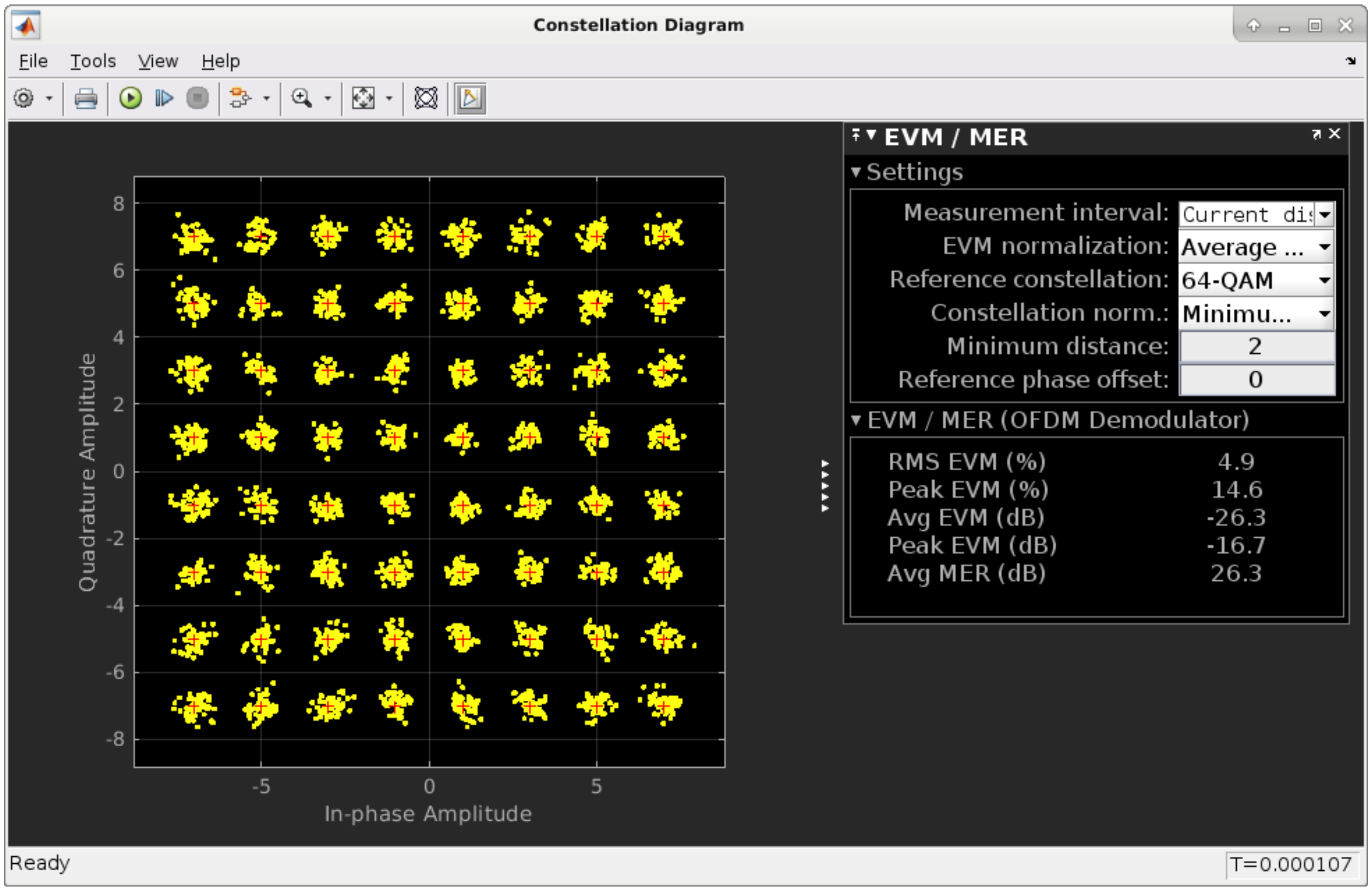
Open the Simulink RF Blockset model including the PA model and the adaptive DPD algorithm excited with a 5G-like OFDM waveform with 100 MHz bandwidth.

System-level model PA + DPD with a 5G-like OFDM waveform

Without DPD linearization, the system has a Modulation Error Ratio of 26 dB, as it can be seen from the constellation plot measurement.

```
model = 'simrfV2_powamp_dpd_comms';
open_system(model)
sim(model)
```





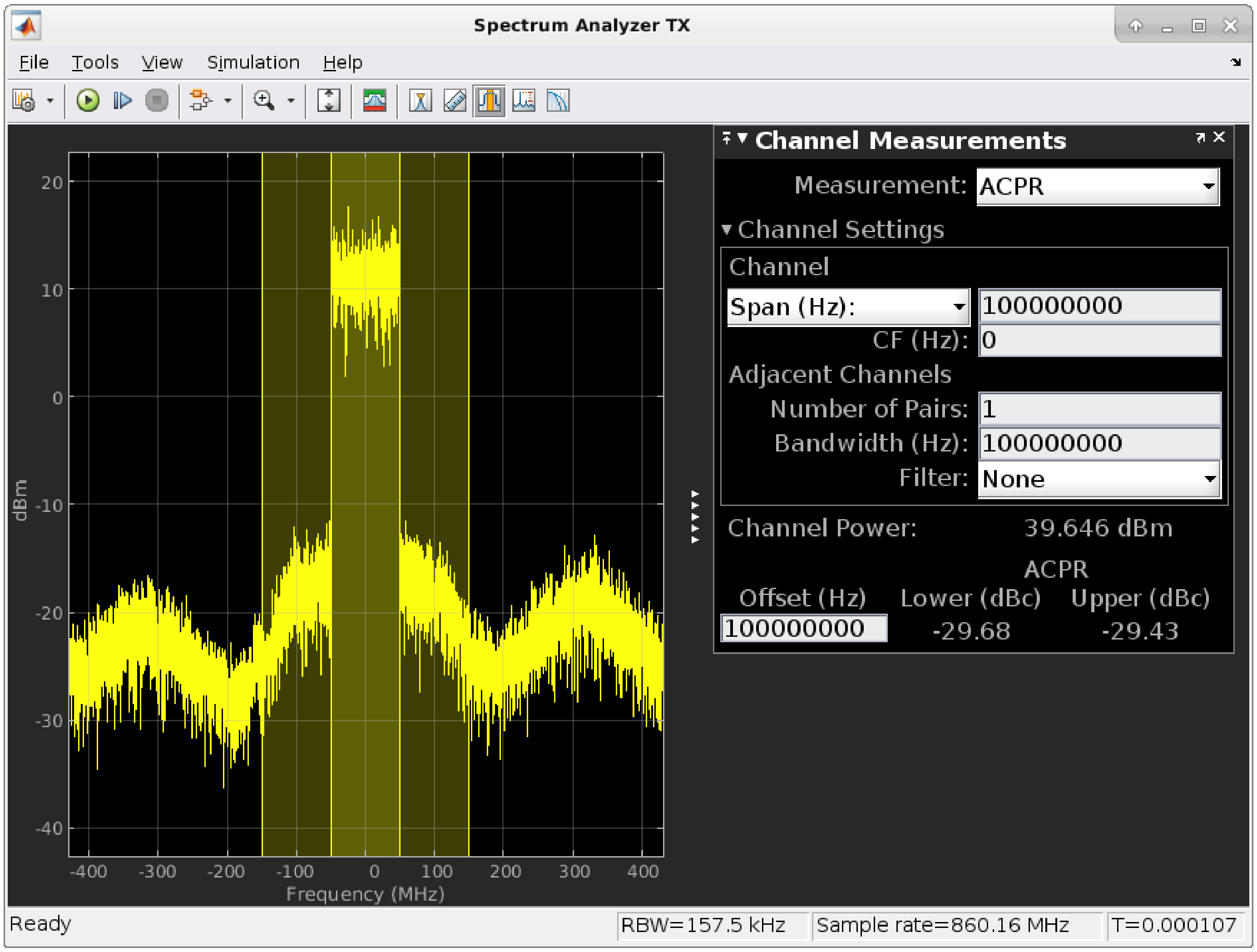
EVM / MER

Settings

Measurement interval:	Current di
EVM normalization:	Average ...
Reference constellation:	64-QAM
Constellation norm.:	Minimu...
Minimum distance:	2
Reference phase offset:	0

EVM / MER (OFDM Demodulator)

RMS EVM (%)	4.9
Peak EVM (%)	14.6
Avg EVM (dB)	-26.3
Peak EVM (dB)	-16.7
Avg MER (dB)	26.3



Channel Measurements

Measurement: ACPR

Channel Settings

Channel

Span (Hz): 100000000

CF (Hz): 0

Adjacent Channels

Number of Pairs: 1

Bandwidth (Hz): 100000000

Filter: None

Channel Power: 39.646 dBm

ACPR

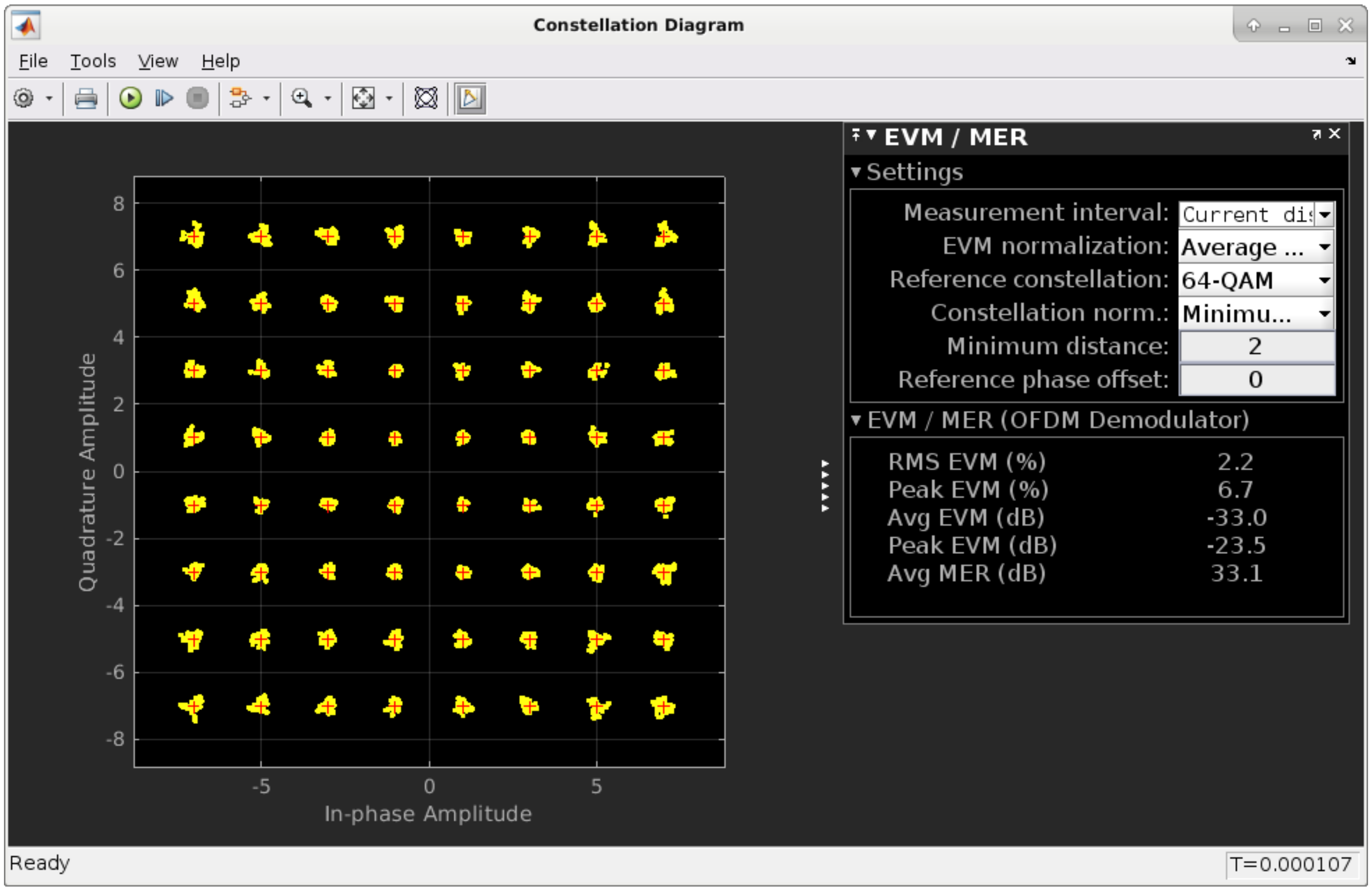
Offset (Hz)	Lower (dBc)	Upper (dBc)
100000000	-29.68	-29.43

Ready

RBW=157.5 kHz | Sample rate=860.16 MHz | T=0.000107

The manual switch is toggled to enable the DPD algorithm. When toggled, the average MER is improved from 26 dB to approximately 33 dB in about 0.1 ms of simulation time.

```
set_param([model '/Manual Switch'], 'action', '1')
sim(model)
```



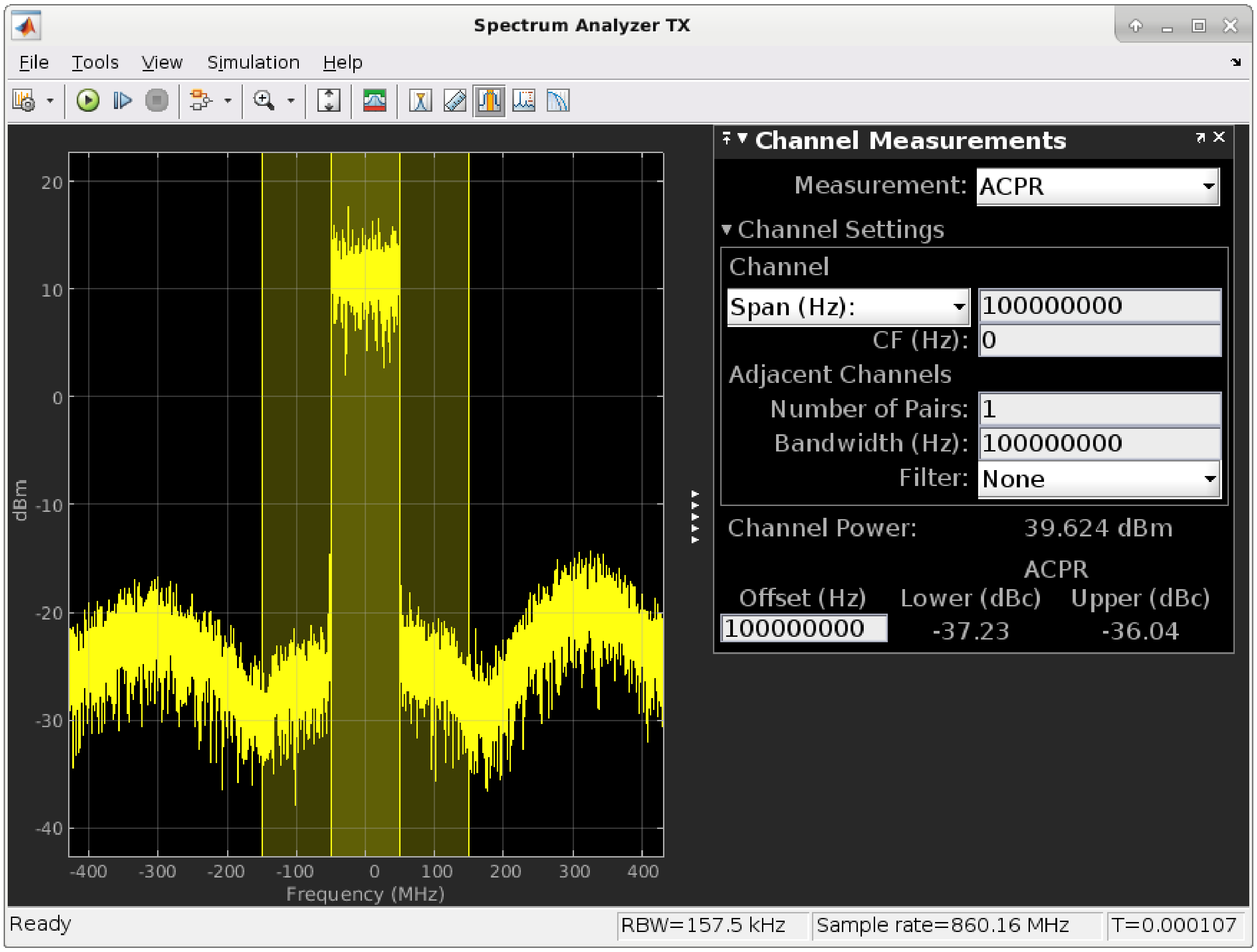
EVM / MER

Settings

Measurement interval: Current dis
EVM normalization: Average ...
Reference constellation: 64-QAM
Constellation norm.: Minimu...
Minimum distance: 2
Reference phase offset: 0

EVM / MER (OFDM Demodulator)

```
RMS EVM (%)      2.2
Peak EVM (%)     6.7
Avg EVM (dB)   -33.0
Peak EVM (dB)  -23.5
Avg MER (dB)    33.1
```



Measurement: ACPR

Channel Settings

Channel

Span (Hz): 100000000

CF (Hz): 0

Adjacent Channels

Number of Pairs: 1

Bandwidth (Hz): 100000000

Filter: None

Channel Power: 39.624 dBm

	ACPR	
Offset (Hz)	Lower (dBc)	Upper (dBc)
100000000	-37.23	-36.04

Ready RBW=157.5 kHz Sample rate=860.16 MHz T=0.000107

```
close_system(model,0)
close all; clear
```

Selected Bibliography

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
Power Amplifier

More About

- “Two-Tone Envelope Analysis Using Real Signals” on page 9-63

Radar System Modeling

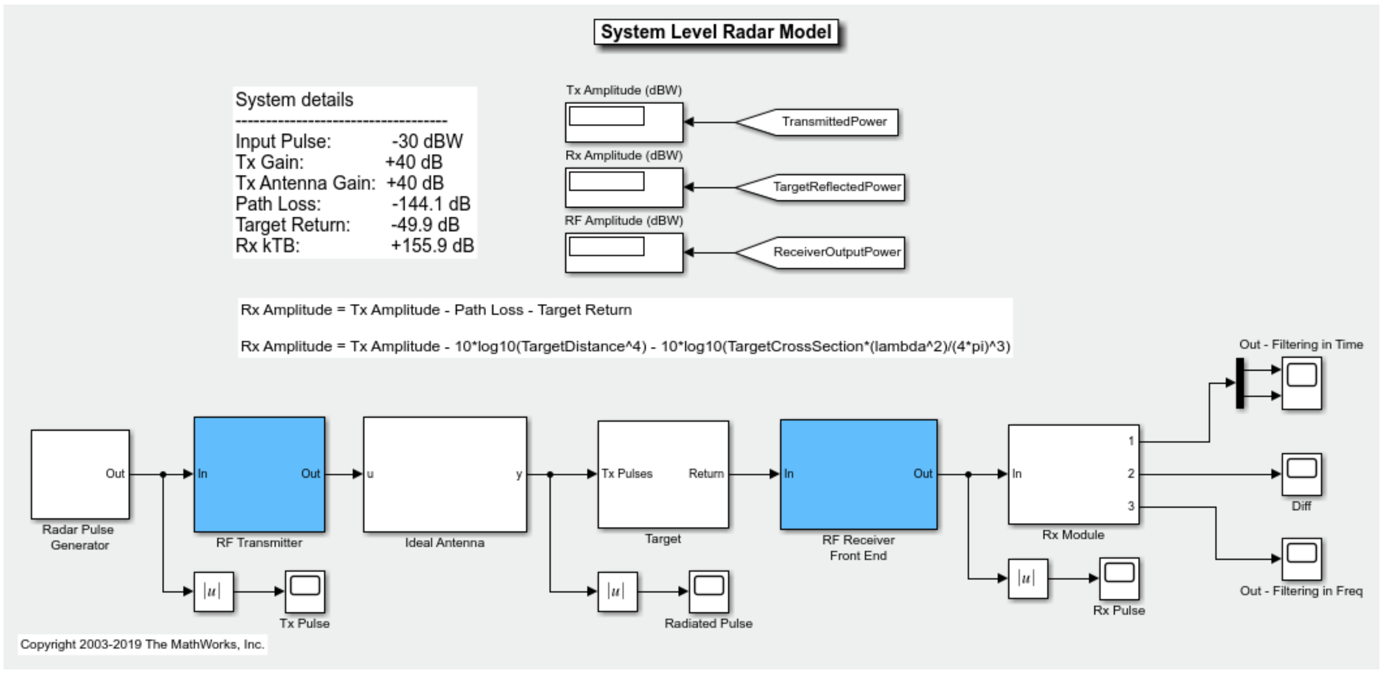
This example shows how to set up a radar system simulation consisting of a transmitter, a channel with a target, and a receiver. For the Aerospace Defense industry, this is an important multi-discipline problem. RF Blockset is used for modeling the RF transmitter and receiver sections.

System Architecture

The system consists of:

- A radar pulse generator, which outputs a chirp with a power of 1 mW at a 2% duty cycle (On time = 2 ms, period = 100 ms).
- An RF transmitter section consisting of a filter and an Amplifier implemented using RF Blockset Circuit Envelope library blocks. Since the filter is a linear device and the amplifier is a non-linear device, they are split into two separate independent subsystems. This separation allows the use of different simulation frequency sets in each subsystem. This separation also permits a trade-off between faster simulation speed and the loss of inter-stage loading effects available in a cascaded chain.
- An ideal antenna element with specified boresight gain operating at 2.1 GHz.
- A moving target implementation that reflects the entire incident signal from its cross-sectional surface. The target surface is perpendicular to the incident radar pulse direction of travel.
- An RF Receiver built using the RF Blockset Circuit Envelope library. The direct conversion structure is implemented in the receiver together with LNA and matching networks. The LNA is describe in a touchstone file and the local oscillator includes a phase noise model. Similar to the RF transmitter section, the receiver is split into independent linear and non-linear subsystems. The matching networks, LNA and filter are in the linear section, while the Mixer and final stage amplifiers are in the non-linear section.

The Receive Module in this example serves two purposes. First, the module contains a matched filter detector for target detection. Second, the module serves as a testbench where a theoretical filter implementation is realized via Simulink blocks. The output of each of these filters is compared and their differences plotted.



System details

Input Pulse:	-30 dBW
Tx Gain:	+40 dB
Tx Antenna Gain:	+40 dB
Path Loss:	-144.1 dB
Target Return:	-49.9 dB
Rx kTB:	+155.9 dB

Rx Amplitude = Tx Amplitude - Path Loss - Target Return

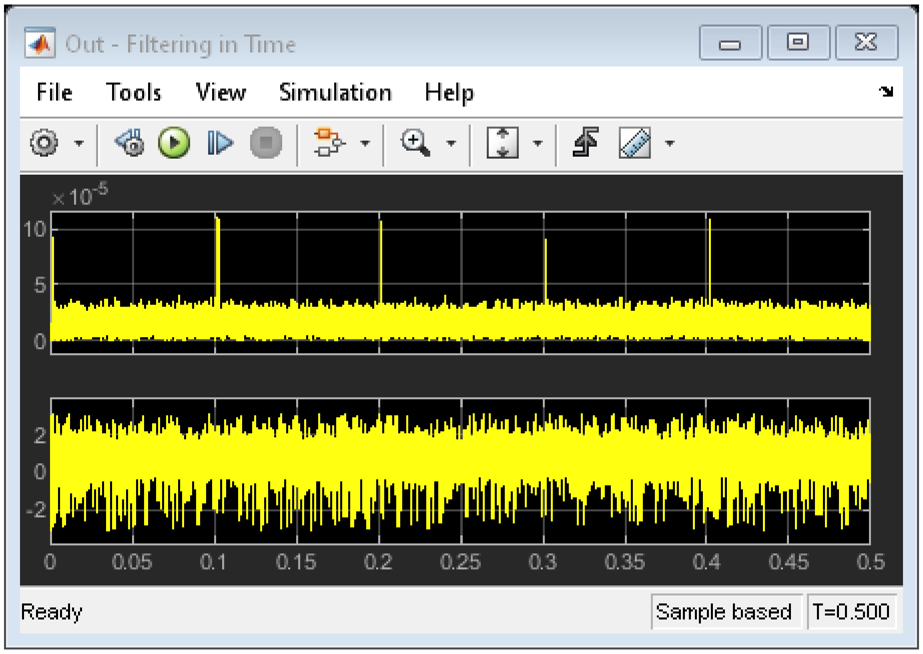
Rx Amplitude = Tx Amplitude - 10*log10(TargetDistance^4) - 10*log10(TargetCrossSection*(lambda^2)/(4*pi)^3)

Running Example Using Default Settings

Set the target cross section, target speed, and relative distance to the target by double-clicking the Target icon. At sufficiently large distances or if the target cross section is too small, the return signal cannot be detected because of the noise.

To start the example simulation:

1 Select **Simulation** > **Run**



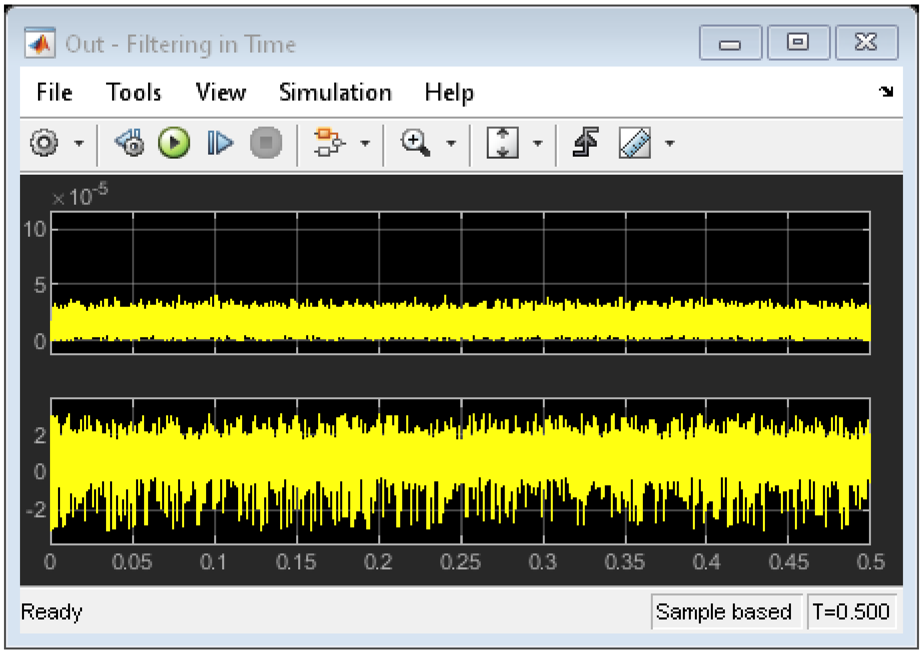
The scope output shows the results for a 0.5 second simulation, while received pulses indicate the presence of the target.

Effect of Antenna Gain/Direction

Open the 'Ideal Antenna' block and change the transmit gain to 10 dB. The target will no longer receive the signal from the main beam of the transmit antenna.

To run the example under this scenario:

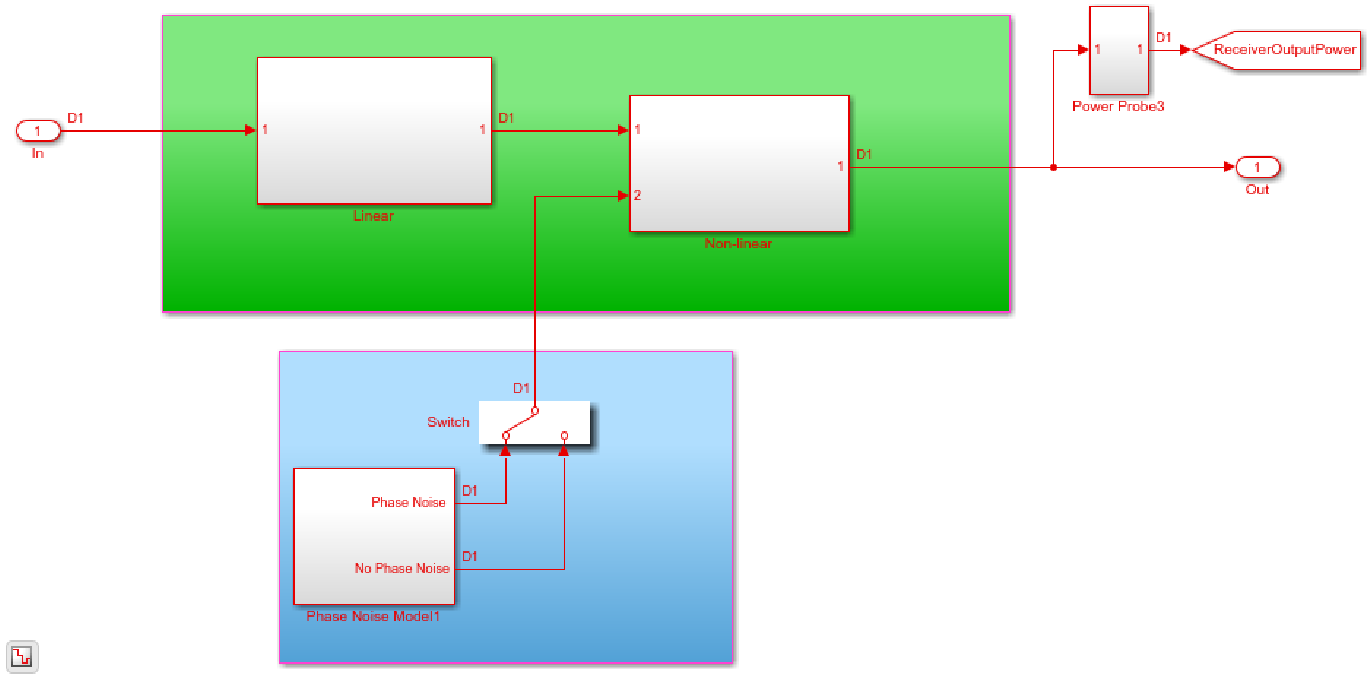
1 Select **Simulation** > **Run**



The effect of the change in antenna gain is observed in the scope. Notice that the pulses are now buried in the noise, rendering the object electromagnetically invisible.

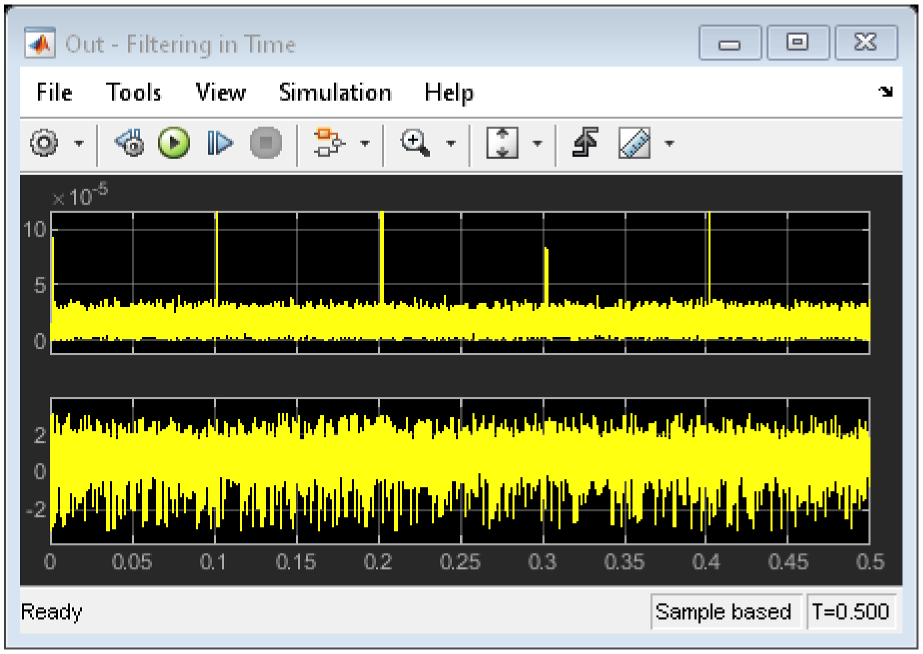
Phase Noise Enabled On Receiver LO

Open the Receiver Front-End subsystem, and use the manual switch to include the phase noise model for the Local Oscillator.



Effect of Phase Noise

1 Double click on the 'Ideal Antenna' block and change the transmit gain back to 40 dB.
2 Select **Simulation** > **Run**



The effect of the phase noise from the Local oscillator is observed in the varying strength of the detected pulses. This varying pulse strength can have an impact on the probability of detection and will result in the target being detected only at certain times.

See Also

“Modeling RF Front End in Radar System Simulation” on page 9-94| “Radar Tracking System” on page 9-142

Modeling RF Front End in Radar System Simulation

In a radar system, the RF front end often plays an important role in defining the system performance. For example, since the RF front end is the first section in the receiver chain, the design of its low noise amplifier is critical to achieving the desired signal-to-noise ratio (SNR). This example shows how to incorporate RF front end behavior into an existing radar system design.

This example requires Phased Array System Toolbox™.

Available Example Implementations

This example includes two Simulink® models:

- Monostatic Radar with One Target: simrfV2_monostatic_radar.slx
- FMCW Radar Range and Speed Estimation: simrfV2_fmcw_radar.slx

Introduction

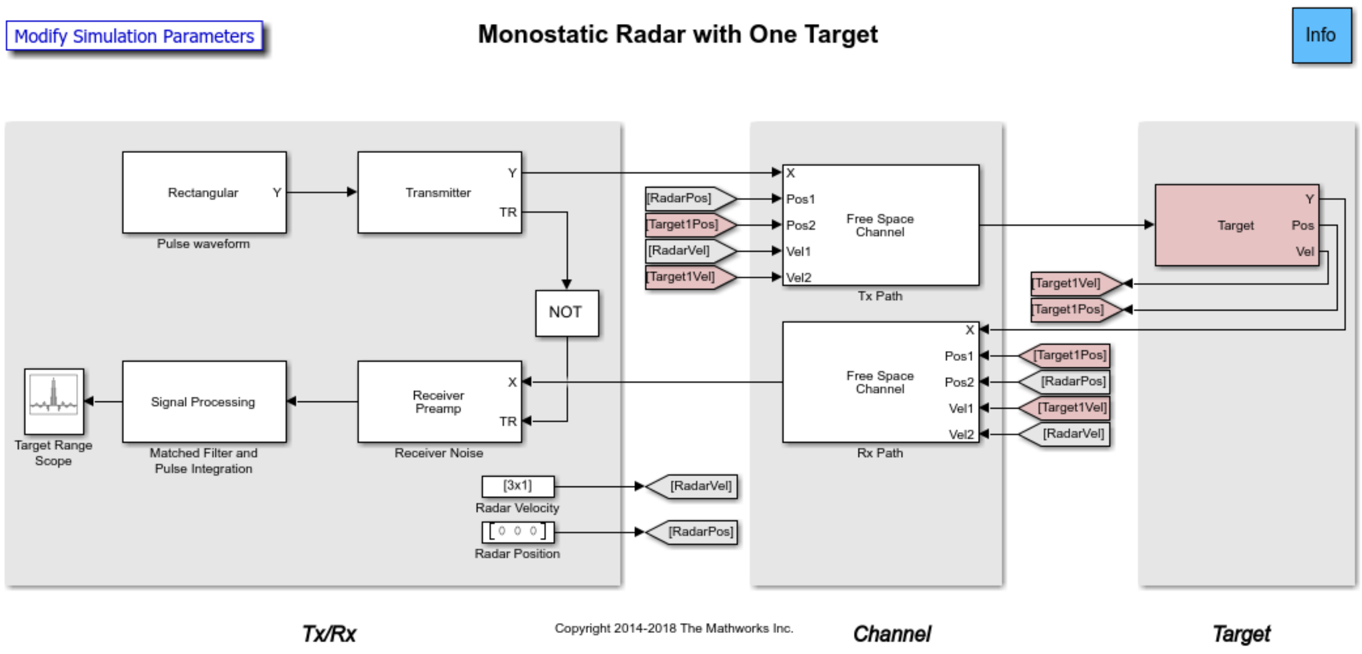
Several examples, such as End-to-End Monostatic Radar and Automotive Adaptive Cruise Control Using FMCW and MFSK Technology, have shown that one can build end-to-end radar systems in Simulink using Phased Array System Toolbox. In many cases, once the system model is built, the next step would add more fidelity in different subsystems and components. A popular candidate for such a component is the RF front end. One advantage of modeling the system in Simulink is the capability of performing multidomain simulations.

The following sections show two examples of incorporating RF Blockset™ modeling capability in radar systems built with Phased Array System Toolbox.

Monostatic Radar with One Target

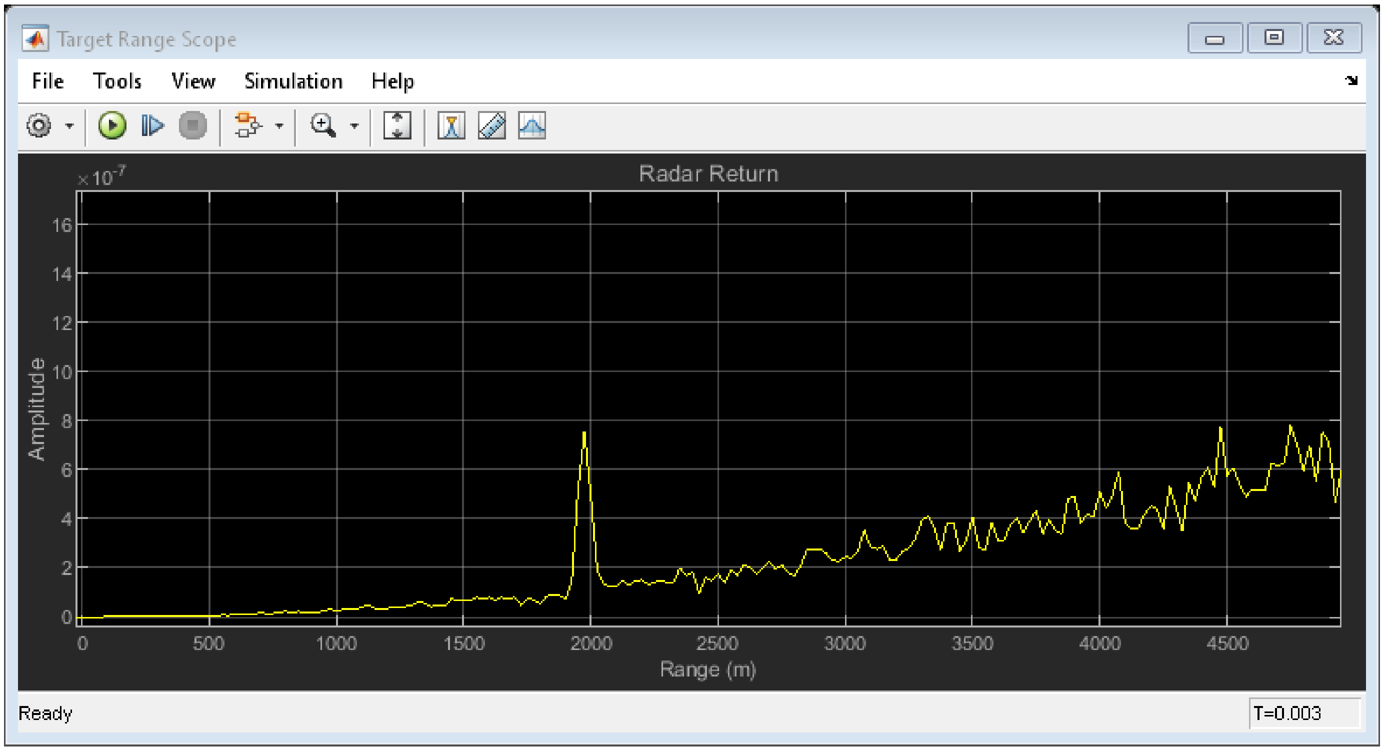
The first model is adapted from example Modeling Mutual Coupling in Large Arrays Using Embedded Element Pattern, which simulates a monostatic pulse radar with one target. From the diagram itself, the model below looks identical to the model shown in that example.

```
model = 'simrfV2_monostatic_radar';
open_system(model);
```



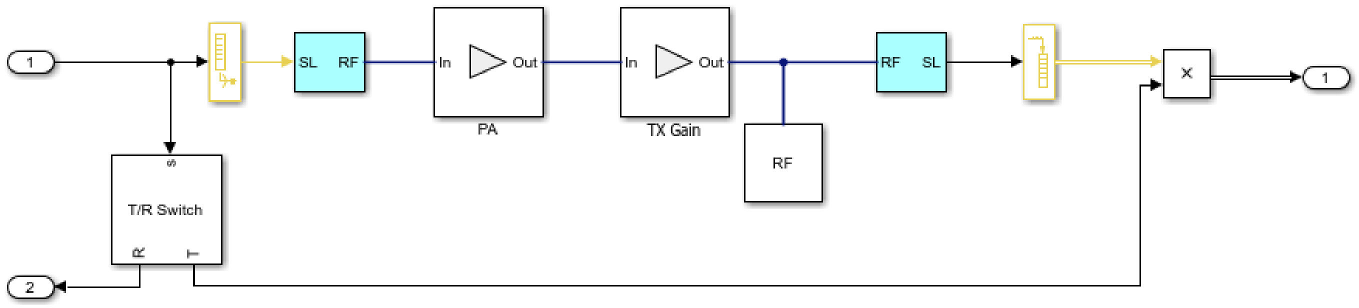
When the model is executed, the resulting plot is also the same.

```
sim(model);
```



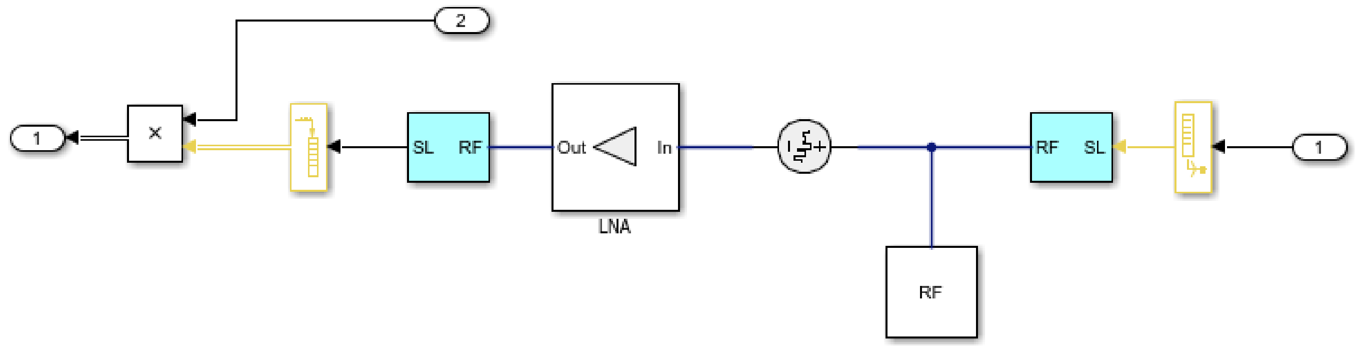
However, a deeper look at the transmitter subsystem shows that the transmitter now employs two RF Blockset amplifiers.

```
open_system([model '/Transmitter']);
```



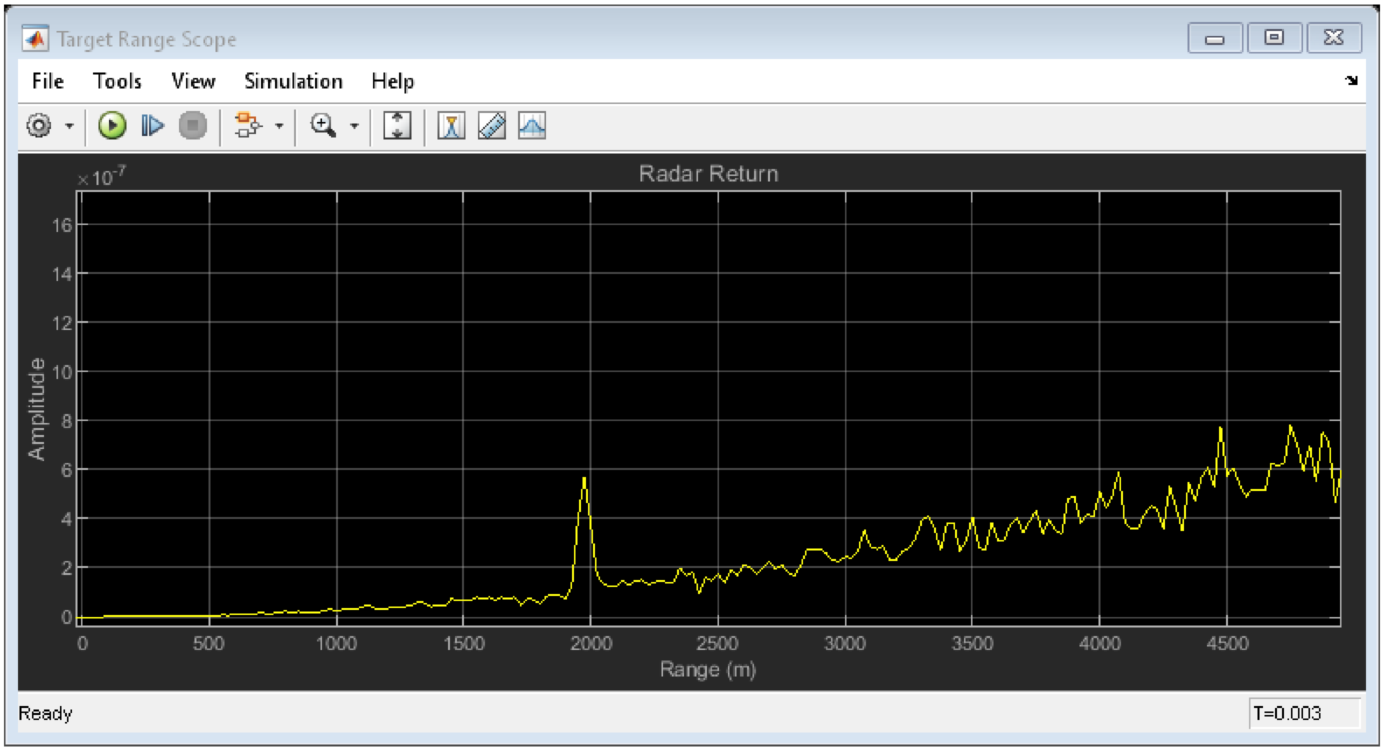
Similar changes are also implemented on the receiver side.

```
open_system([model '/Receiver Noise']);
```



With these changes, the model is capable of simulating RF behaviors. For example, the simulation result shown above assumes a perfect power amplifier. In real applications, the amplifier suffers from many nonlinearities. If one sets the IP3 of the transmitter to 70 dB and runs the simulation again, the peak corresponding to the target is no longer as dominant. This gives the engineer some knowledge regarding the system's performance under different situations.

```
set_param([model '/Transmitter/PA'],'IP3','70');
sim(model);
```

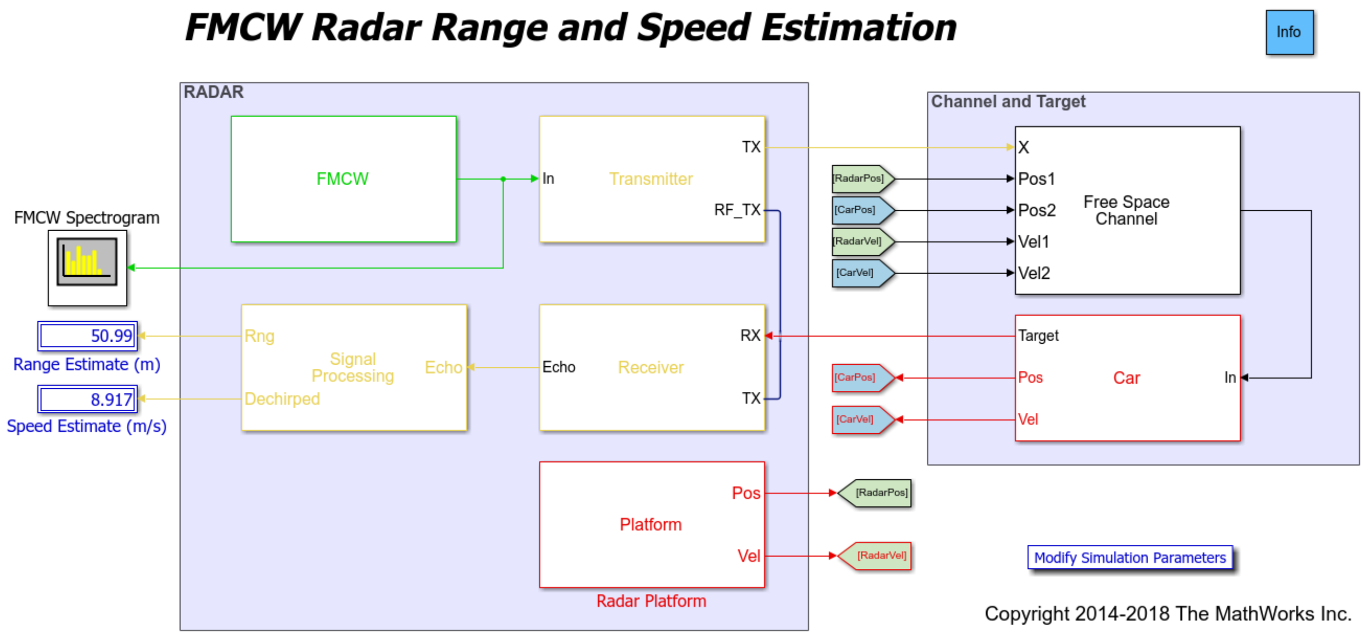


```
bdclose(model);
clear model;
```

FMCW Radar Range and Speed Estimation

The second example is adapted from Automotive Adaptive Cruise Control Using FMCW Technology. However, this model uses a triangle sweep waveform instead, so the system can estimate range and speed simultaneously. At the top level, the model is similar to what gets built from Phased Array System Toolbox. Once executed, the model shows the estimated range and speed values that match the distance and relative speed of the target car.

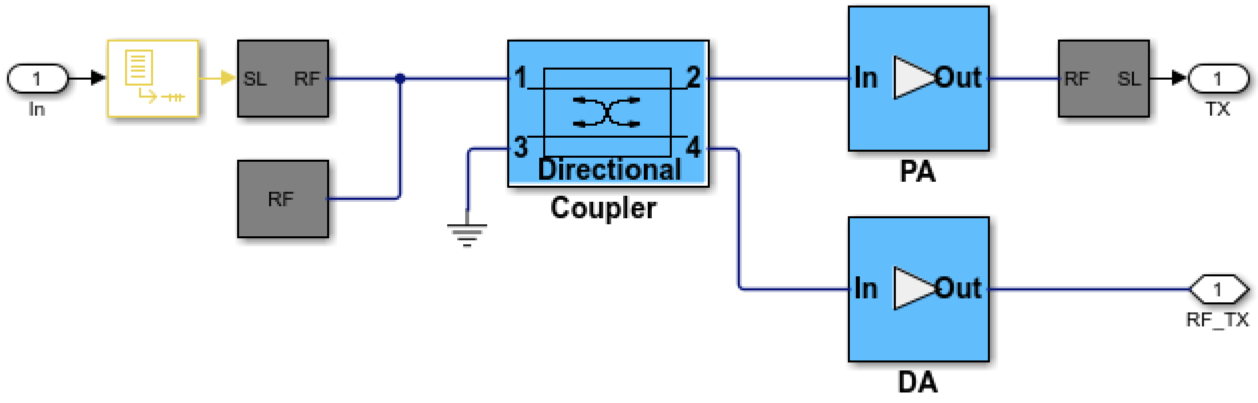
```
model = 'simrfV2_fmcw_radar';
open_system(model);
sim(model);
```



Similar to the first example, the transmitter and receiver subsystems are now built with RF Blockset blocks.

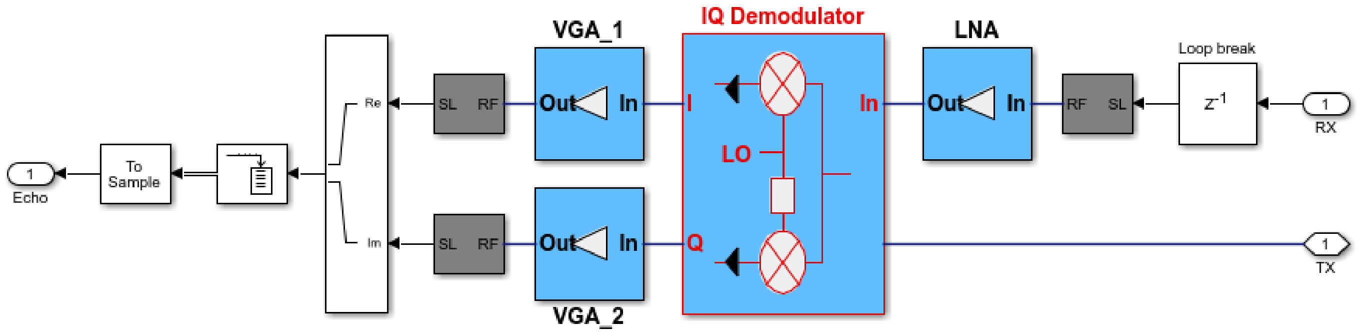
The following figure shows the transmitter subsystem.

```
open_system([model '/Radar Transmitter']);
```



The following figure shows the receiver subsystem.

```
open_system([model '/Radar Receiver']);
```



In a continuous wave radar system, part of the transmitted waveform is used as a reference to dechirp the received target echo. From the diagrams above, one can see that the transmitted waveform is sent to the receiver via a coupler and the dechirping is performed via an I/Q mixer. Therefore, by adjusting parameters in those RF components, higher simulation fidelity can be achieved.

```
bdclose(model);
clear model;
```

Summary

This example shows two radar models that were originally built with Phased Array System Toolbox and later incorporated RF models from RF Blockset. The simulation fidelity is greatly improved by combining the two products together.

See Also

Amplifier | Configuration | Power Amplifier

More About

- “Radar System Modeling” on page 9-89
- “RF Receiver Modeling for LTE Reception” on page 9-100

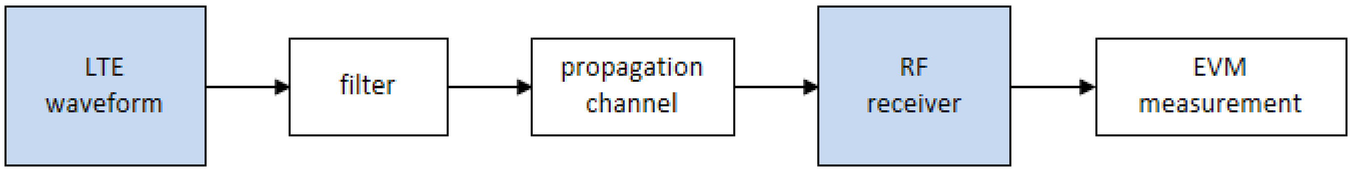
RF Receiver Modeling for LTE Reception

This example demonstrates how to model and test an LTE RF receiver using LTE Toolbox™ and RF Blockset™.

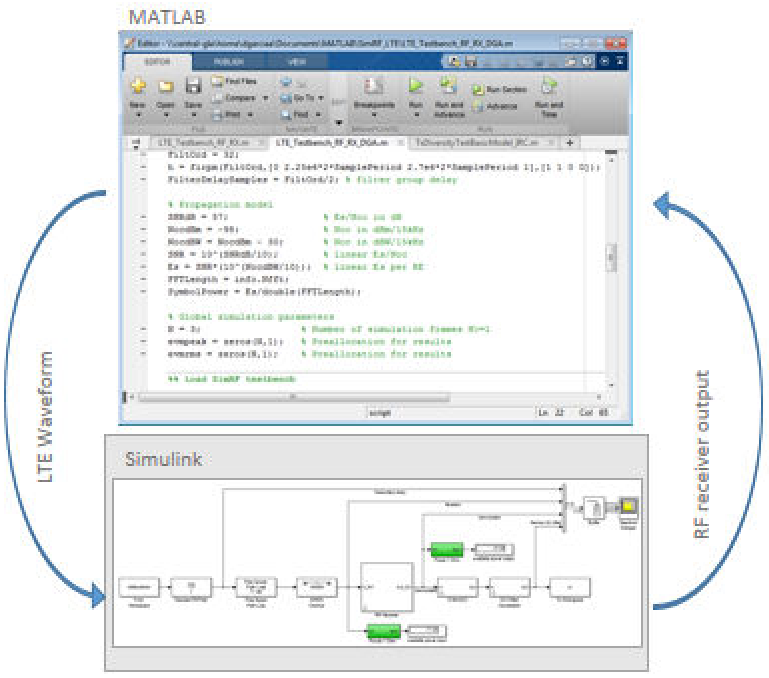
The example requires LTE Toolbox.

Model Description

The figure below shows the system model for this example. An LTE waveform is generated, filtered, transmitted through a propagation channel and fed into an RF Blockset receiver model. The model can be assembled using commercially available parts. EVM measurements are performed on the RF receiver output.



This example is implemented using MATLAB® and Simulink®, which interact at runtime. The functional partition is:



```
FiltOrd = 32;
h = firpm(FiltOrd,[0 2.25e6*2*SamplePeriod 2.7e6*2*SamplePeriod 1],[1 1 0 0]);
FilterDelaySamples = FiltOrd/2; % filter group delay

% Propagation model
SNRdB = 37;                 % Es/Noc in dB
NocdBm = -98;               % Noc in dBm/15kHz
NocdBW = NocdBm - 30;       % Noc in dBW/15kHz
SNR = 10^(SNRdB/10);        % linear Es/Noc
Es = SNR*(10^(NocdBW/10));  % linear Es per RE
FFTLength = info.Nfft;
SymbolPower = Es/double(FFTLength);

% Global simulation parameters
N = 3;                 % Number of simulation frames N>=1
evmpeak = zeros(N,1);  % Preallocation for results
evmrms = zeros(N,1);   % Preallocation for results

%% Load SimRF testbench
```

The measurement testbench is implemented with a MATLAB script using the Simulink model as the device under test (DUT). LTE frames are streamed between testbench and DUT.

Generate LTE Waveform

In this section we generate the LTE waveform using the LTE Toolbox. We use the reference measurement channel (RMC) R.6 as defined in TS 36.101 [1]. This RMC specifies a 25 resource elements (REs) bandwidth, equivalent to 5 MHz. A 64 QAM modulation is used. All REs are allocated. Additionally, OCNG noise is enabled in unused REs.

Only one frame is generated. This frame will then be repeated a number of times to perform the EVM measurements.

```
% Configuration TS 36.101 25 REs (5 MHz), 64-QAM, full allocation
rmc = lteRMCDL('R.6');
rmc.OCNGPDSCHEnable = 'On';

% Create eNodeB transmission with fixed PDSCH data
rng(2);                 % Fixed random seed (arbitrary)
data = randi([0 1], sum(rmc.PDSCH.TrBlkSizes),1);

% Generate 1 frame, to be repeated to simulate a total of N frames
[tx, ~, info] = lteRMCDLTool(rmc, data); % 1 frame

% Calculate the sampling period and the length of the frame.
```

```
SamplePeriod = 1/info.SamplingRate;
FrameLength = length(tx);
```

Initialize Simulation Components

This section initializes some of the simulation components:

- Band limiting filter: design the filter coefficients, which will be used by the Simulink model. The filter has order 32, with passband frequency equal to 2.25 MHz, and stopband frequency equal to 2.7 MHz.
- Number of frames: this is the number of times the generated frame is repeated
- Preallocate result vectors

```
% Band limiting interpolation filter
FiltOrd = 32;
h = firpm(FiltOrd,[0 2.25e6*2*SamplePeriod 2.7e6*2*SamplePeriod 1],[1 1 0 0]);
FilterDelaySamples = FiltOrd/2; % filter group delay

% Number of simulation frames N>=1
N = 3;

% Preallocate vectors for results for N-1 frames
% EVM is not measured in the first frame to avoid transient effects
evmpeak = zeros(N-1,1);   % Preallocation for results
evmrms = zeros(N-1,1);    % Preallocation for results
```

Load RF Blockset Testbench

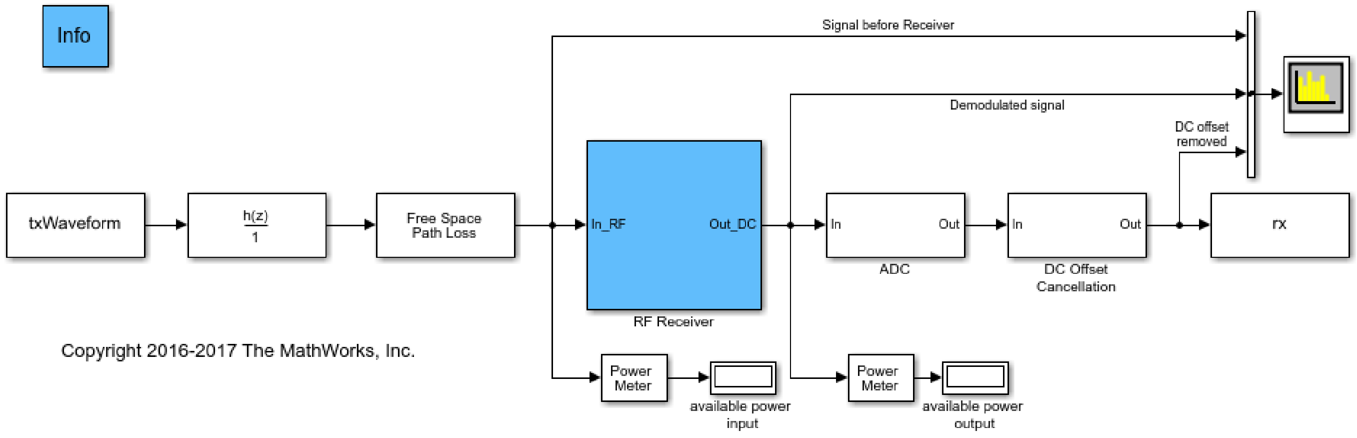
This section loads the Simulink model shown below. The model performs the following functions:

- Reading the LTE waveform and the sampling period from the workspace
- Bandlimiting filtering
- Saving results to workspace

The model includes the following components:

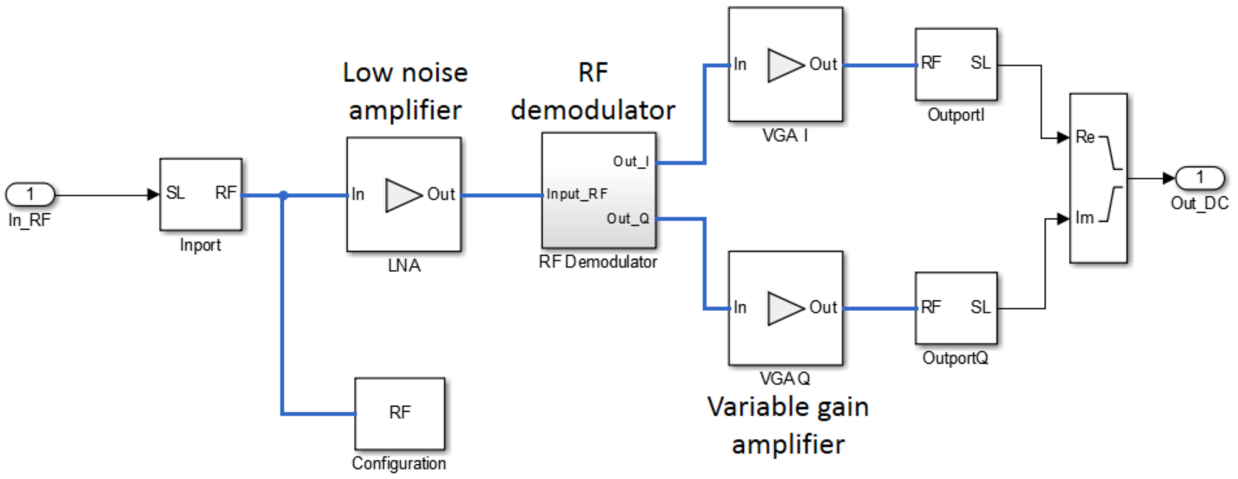
- Channel model: includes free space path loss
- RF receiver: includes direct conversion demodulator
- ADC and DC offset cancellation

```
% Specify and open Simulink model
model = 'simrfV2_lte_receiver';
open_system(model);
```



RF Receiver Model

The RF receiver model includes the elements shown below



This model was initially generated using the RF Budget Analyzer app, and later modified to include additional impairments. Initial RF Budget Analysis of RF Receiver

Simulate Frames

This section simulates the specified number of frames. This is done in two stages:

- Simulate the first frame
- Simulate the rest of the frames in a loop

The reason for splitting the processing in these two stages is to simplify the code. During the processing of the first frame we need to take into account the delay of the band limiting filter. This is not the case for subsequent frames, since the filter state is maintained between frames. Therefore, the length of the first frame has to be increased slightly to take into account the delay introduced by the filter.

Simulate First LTE Frame

As mentioned for the first simulated frame we need to increase the length of the signal fed to the Simulink model to compensate for the delay introduced by the filter. Next we launch the simulation of the Simulink model without loading any initial state. The model is set to simulate in `Accelerator` mode to decrease run time. After processing the first frame with the Simulink model, its state (`xFinal`) is stored and assigned to `xInitial` for the next time the model is run.

The output of the Simulink model is stored in the variable `rx`, which is available in the workspace. Any delays introduced to this signal are removed after performing synchronization. The EVM is measured on the resulting waveform.

```
% Generate test data for RF receiver
time = (0:FrameLength+FilterDelaySamples)*SamplePeriod;
% Append to the end of the frame enough samples to compensate for the delay
% of the filter
txWaveform = timeseries([tx; tx(1:FilterDelaySamples+1)],time);

% Simulate RF Blockset model of RF RX
set_param(model, 'LoadInitialState', 'off');
sim(model, time(end));
% Save the final sate of the model in xInitial for next frame processing
xInitial = xFinal;

% Synchronize to received waveform
Offset = lteDLFrameOffset(rmc,squeeze(rx),'TestEVM');
% In this case Offset = FilterDelaySamples therefore the following
% frames do not require synchronization
```

Simulate Successive LTE Frames

Now the rest of the frames can be simulated. First, the model state is set using the value stored in `xInitial` at the output of the previous iteration.

```
% Load state after execution of previous frame. Since we are repeating the
% same frame the model state will be the same after every frame execution.
set_param(model, 'LoadInitialState', 'on', 'InitialState','xInitial');
% Modify input vector to take into account the delay of the bandlimiting
% filter
RepeatFrame = [tx(FilterDelaySamples+1:end); tx(1:FilterDelaySamples+1)];
EVMalg.EnablePlotting = 'Off';
cec.PilotAverage = 'TestEVM';

for n = 2:N % for all remaining frames
    % Generate data
    time = ((n-1)*FrameLength+(0:FrameLength)+FilterDelaySamples)  *    ...
        SamplePeriod;
    txWaveform = timeseries(RepeatFrame,time);

    % Execute Simulink RF Blockset testbench
    sim(model, time(end));
    xInitial = xFinal; % Save model state
    % Compute and display EVM measurements
    evmmeas = simrfV2_lte_receiver_evm_cal(rmc,cec,squeeze(rx),EVMalg);
    evmpeak(n-1) = evmmeas.Peak;
    evmrms(n-1) = evmmeas.RMS;
end
```

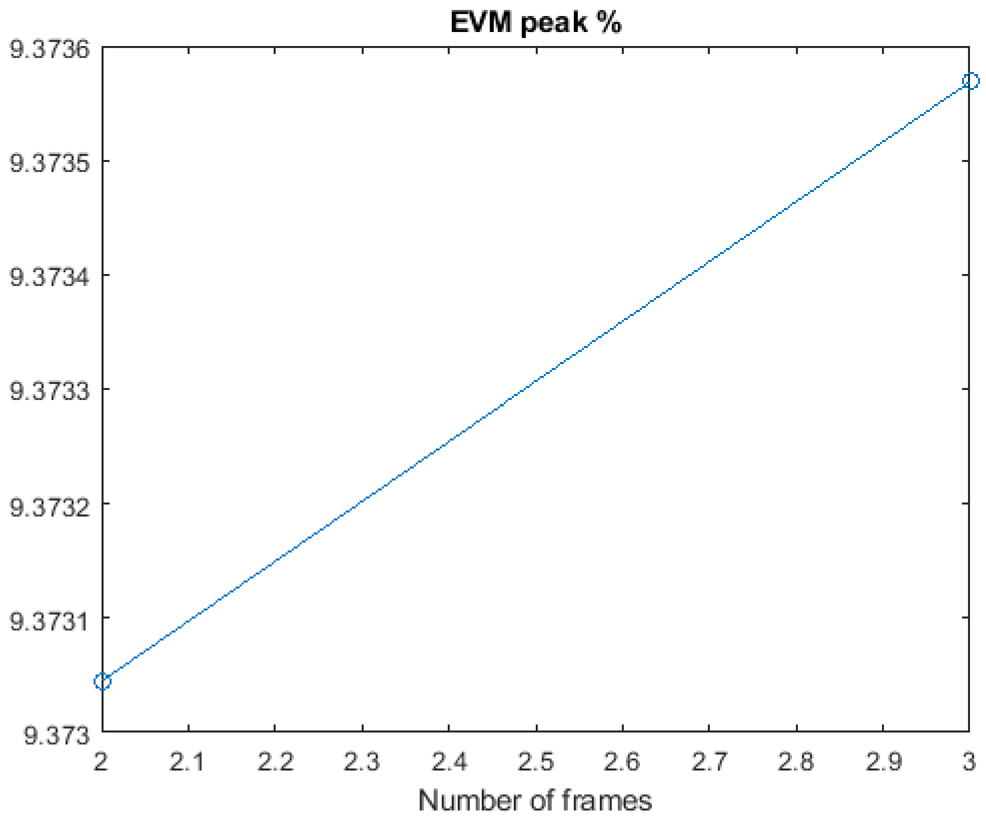
```
Low edge EVM, subframe 0: 3.165%
High edge EVM, subframe 0: 3.173%
Low edge EVM, subframe 1: 3.024%
High edge EVM, subframe 1: 3.036%
Low edge EVM, subframe 2: 3.004%
High edge EVM, subframe 2: 2.992%
Low edge EVM, subframe 3: 2.964%
High edge EVM, subframe 3: 2.927%
Low edge EVM, subframe 4: 3.003%
High edge EVM, subframe 4: 3.012%
Low edge EVM, subframe 6: 3.001%
High edge EVM, subframe 6: 2.983%
Low edge EVM, subframe 7: 3.005%
High edge EVM, subframe 7: 3.002%
Low edge EVM, subframe 8: 3.019%
High edge EVM, subframe 8: 3.002%
Low edge EVM, subframe 9: 3.050%
High edge EVM, subframe 9: 3.041%
Averaged low edge EVM, frame 0: 3.024%
Averaged high edge EVM, frame 0: 3.017%
Averaged EVM frame 0: 3.024%
Averaged overall EVM: 3.024%
Low edge EVM, subframe 0: 3.165%
High edge EVM, subframe 0: 3.173%
Low edge EVM, subframe 1: 3.024%
High edge EVM, subframe 1: 3.036%
Low edge EVM, subframe 2: 3.004%
High edge EVM, subframe 2: 2.992%
Low edge EVM, subframe 3: 2.964%
High edge EVM, subframe 3: 2.927%
Low edge EVM, subframe 4: 3.003%
High edge EVM, subframe 4: 3.012%
Low edge EVM, subframe 6: 3.001%
High edge EVM, subframe 6: 2.983%
Low edge EVM, subframe 7: 3.005%
High edge EVM, subframe 7: 3.003%
Low edge EVM, subframe 8: 3.019%
High edge EVM, subframe 8: 3.002%
Low edge EVM, subframe 9: 3.050%
High edge EVM, subframe 9: 3.041%
Averaged low edge EVM, frame 0: 3.024%
Averaged high edge EVM, frame 0: 3.017%
Averaged EVM frame 0: 3.024%
Averaged overall EVM: 3.024%
```

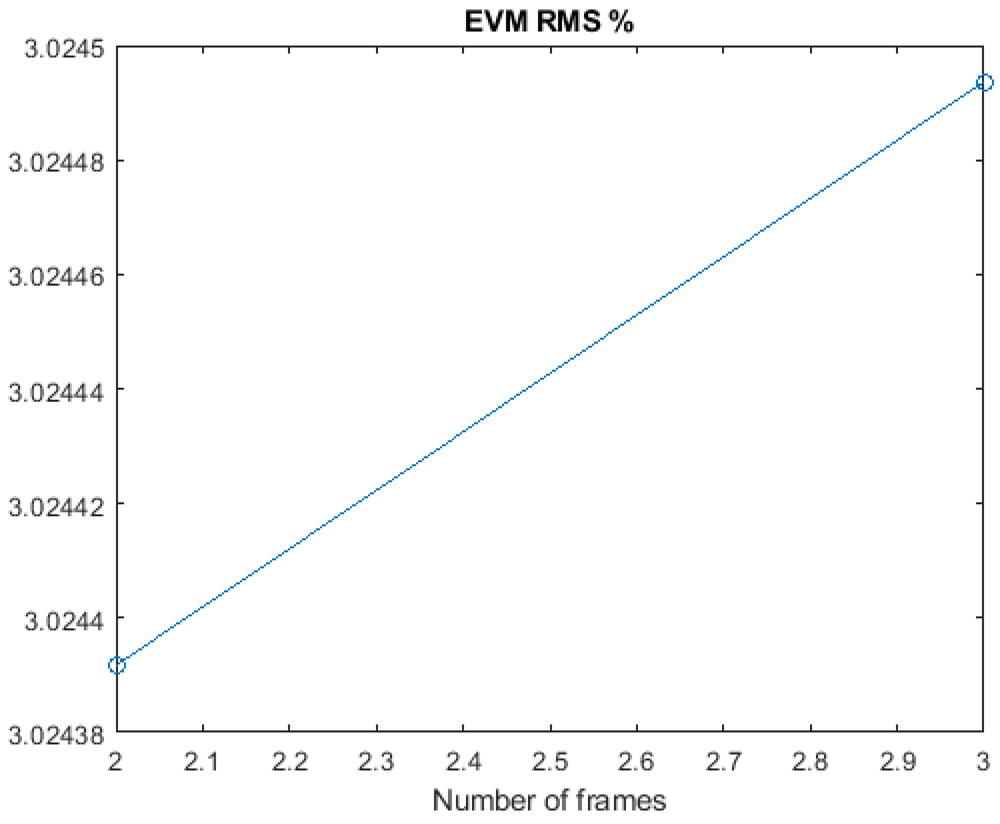
Visualize Measured EVM

This section plots the measured peak and RMS EVM for each simulated frame.

```
hf(1) = figure;
plot((2:N), 100*evmpeak,'o-')
title('EVM peak %');
```

```
xlabel('Number of frames');
hf(2) = figure;
plot((2:N), 100*evmrms,'o-');
title('EVM RMS %');
xlabel('Number of frames');
```





Cleaning Up

Close the Simulink model and remove the generated files.

```
bdclose(model);
close(hf)
clear 'hf'
```

Appendix

This example uses the following helper function:

- simrfV2_lte_receiver_evm_cal.m

Selected Bibliography

1 3GPP TS 36.101 "User Equipment (UE) radio transmission and reception"

See Also

Amplifier | VGA | Configuration | Inport | Outport

Related Topics

“Carrier to Interference Performance of Weaver Receiver” on page 2-2

Create Custom RF Blockset™ Models

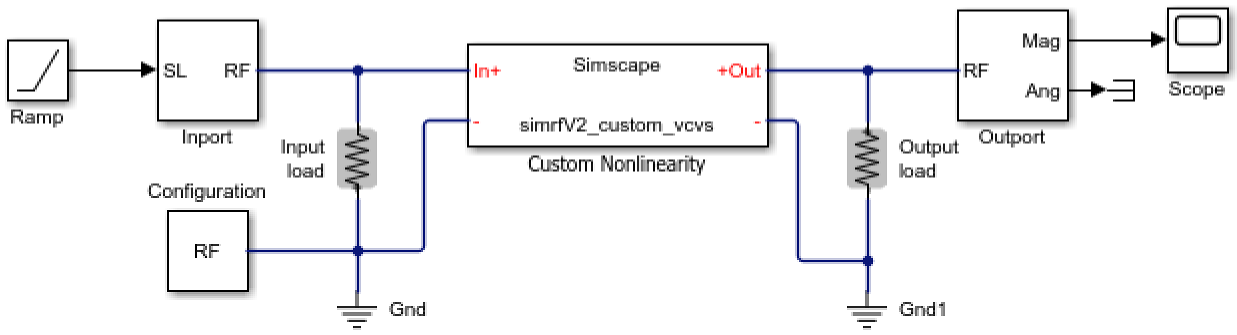
This example shows how to write your own RF Blockset Circuit Envelope model in Simscape® language for complex baseband simulation. An RF Circuit Envelope complex baseband signal resides on a carrier with specified frequency. This baseband signal will modulate with other signals when the system is nonlinear. The example nonlinearity is implemented with a Simscape Component block and includes a Simscape ssc-file to describe the nonlinear voltage polynomial.

System Architecture

The system consists of:

- An input voltage signal, linearly increasing in time and generated with a Simulink Ramp block.
- An RF Blockset Inport block to specify the `Carrier frequencies` (**Input_Freq**) of the input voltage signal. This setup allows observation of the system nonlinear behavior for different input settings.
- A custom nonlinear voltage amplifier (polynomial voltage controlled voltage source), modeled with a Simscape Component block. The device equations are written in passband (time) domain and assume instantaneous voltage V(t) and current I(t) values. These equations are interpreted by RF Blockset envelope solver in both passband and baseband domains (zero and nonzero carrier frequencies).
- An Outport block to specify output `Carrier frequencies` (**Output_Freqs**). The output carrier frequencies are higher order harmonics (integer multiples) of the Inport frequency resulting from the amplifier nonlinearity.
- A Scope to display the magnitudes of the output voltages at **Output_Freqs** frequencies as specified in the Outport block.
- Load resistors and ground nodes needed to make the circuit electrically sound. By construction, the resistor values do not affect the output voltage.
- A Configuration block to control the system carrier frequencies required for accurate simulation and other simulation properties.

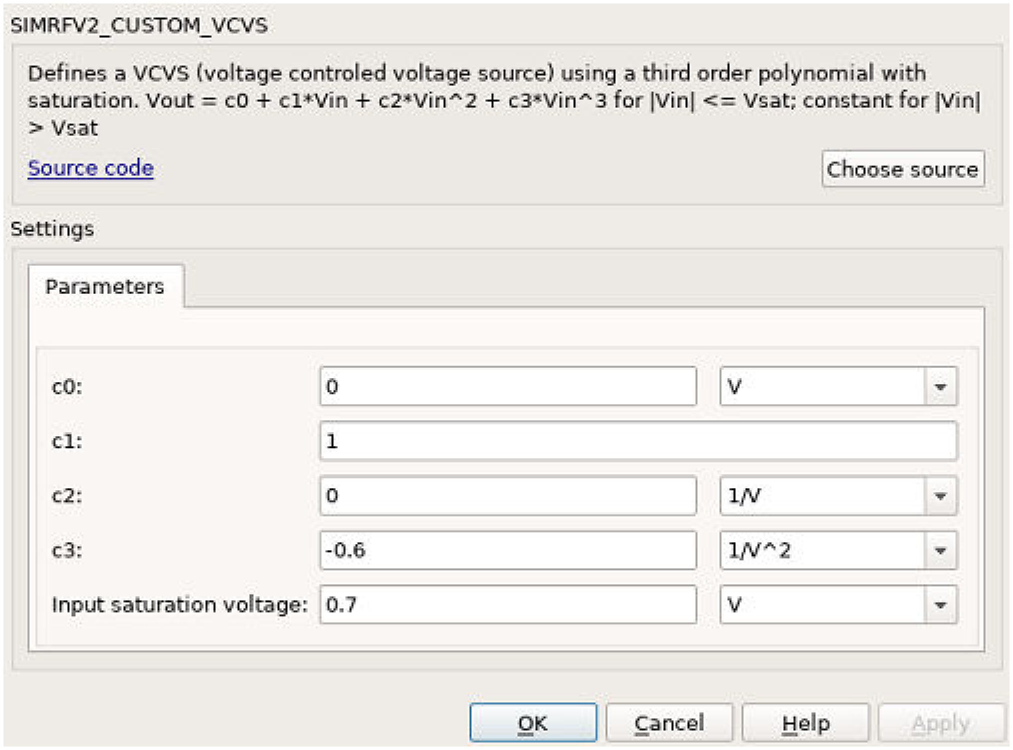
```
model = 'simrfV2_custom_polynomial';
open_system(model);
```



Copyright 2013-2018 The MathWorks, Inc.

Examine the Model

Double-click the "Custom Nonlinearity" block or type `open_system([model '/Custom Nonlinearity'])` in the command window to open the Custom Nonlinearity block mask.


SIMRFV2_CUSTOM_VCVS

Defines a VCVS (voltage controled voltage source) using a third order polynomial with saturation. Vout = c0 + c1*Vin + c2*Vin^2 + c3*Vin^3 for |Vin| <= Vsat; constant for |Vin| > Vsat

Source code

Settings

c0:	0	V
c1:	1	
c2:	0	1/V
c3:	-0.6	1/V^2
Input saturation voltage:	0.7	V

The file `simrfV2_custom_vcvs.ssc` describes the custom device. View the source code by clicking the block mask "Source code" link or typing `edit simrfV2_custom_vcvs` at the command prompt.

Copy the file `simrfV2_custom_vcvs.ssc` to a directory where you have write permission to rename and modify the file. Click the block mask "Choose source" button to replace the current device implementation with yours. Use the mask `Help` button for additional information.

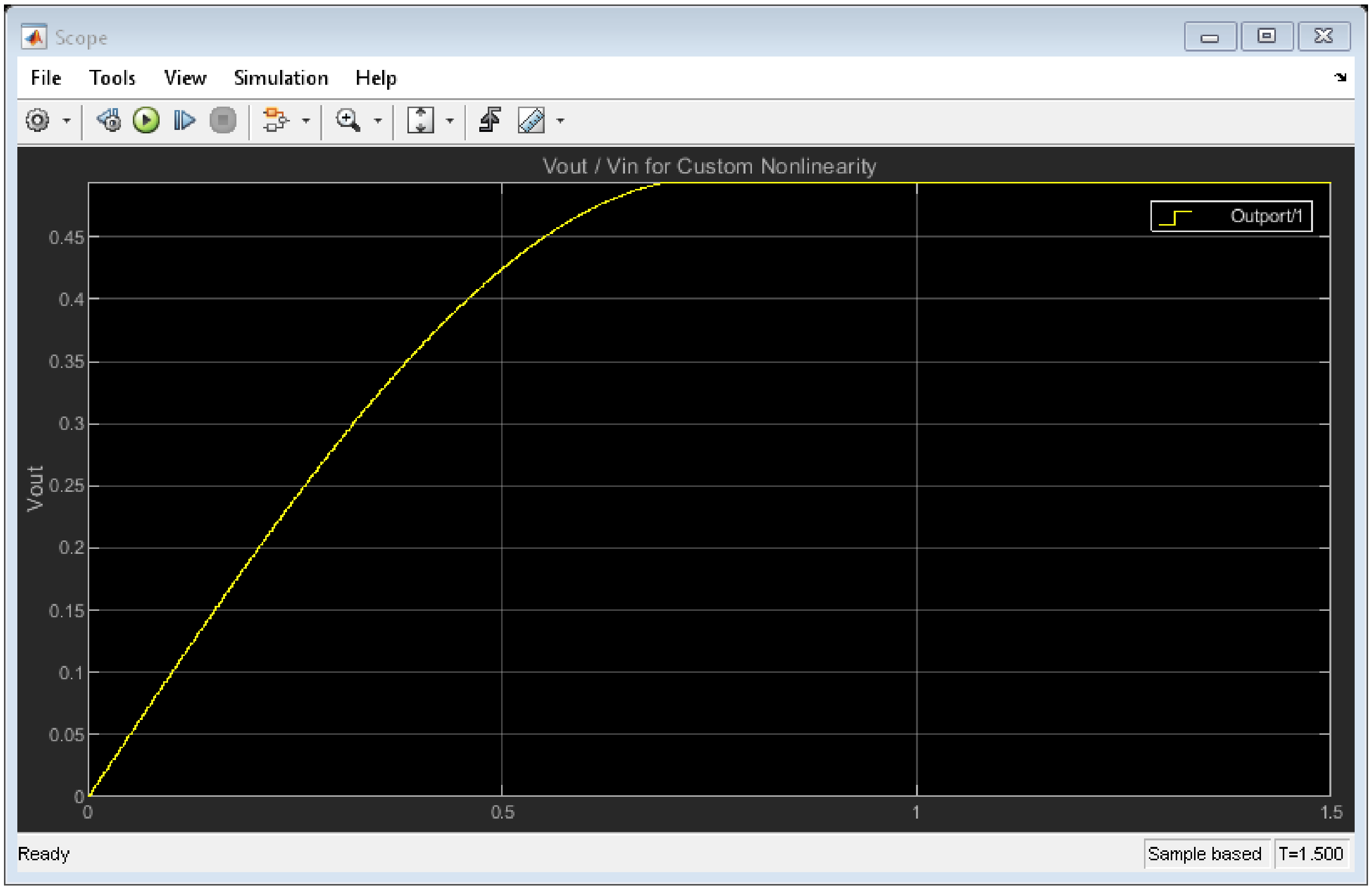
The above method uses the Simscape Component block from the Simscape Utilities library to avoid the library build process. See Creating Custom Components documentation for more details.

Run the Model Using Default Settings

For this example, default input and output frequencies are set to `0` and the result is a passband simulation. The input voltage magnitude is linearly increasing in time, **Vin(t) = t**, and the custom nonlinearity relationship **Vout(Vin)** is shown in the scope.

The model is simulated after entering the following into the command window

```
sim(model);
```



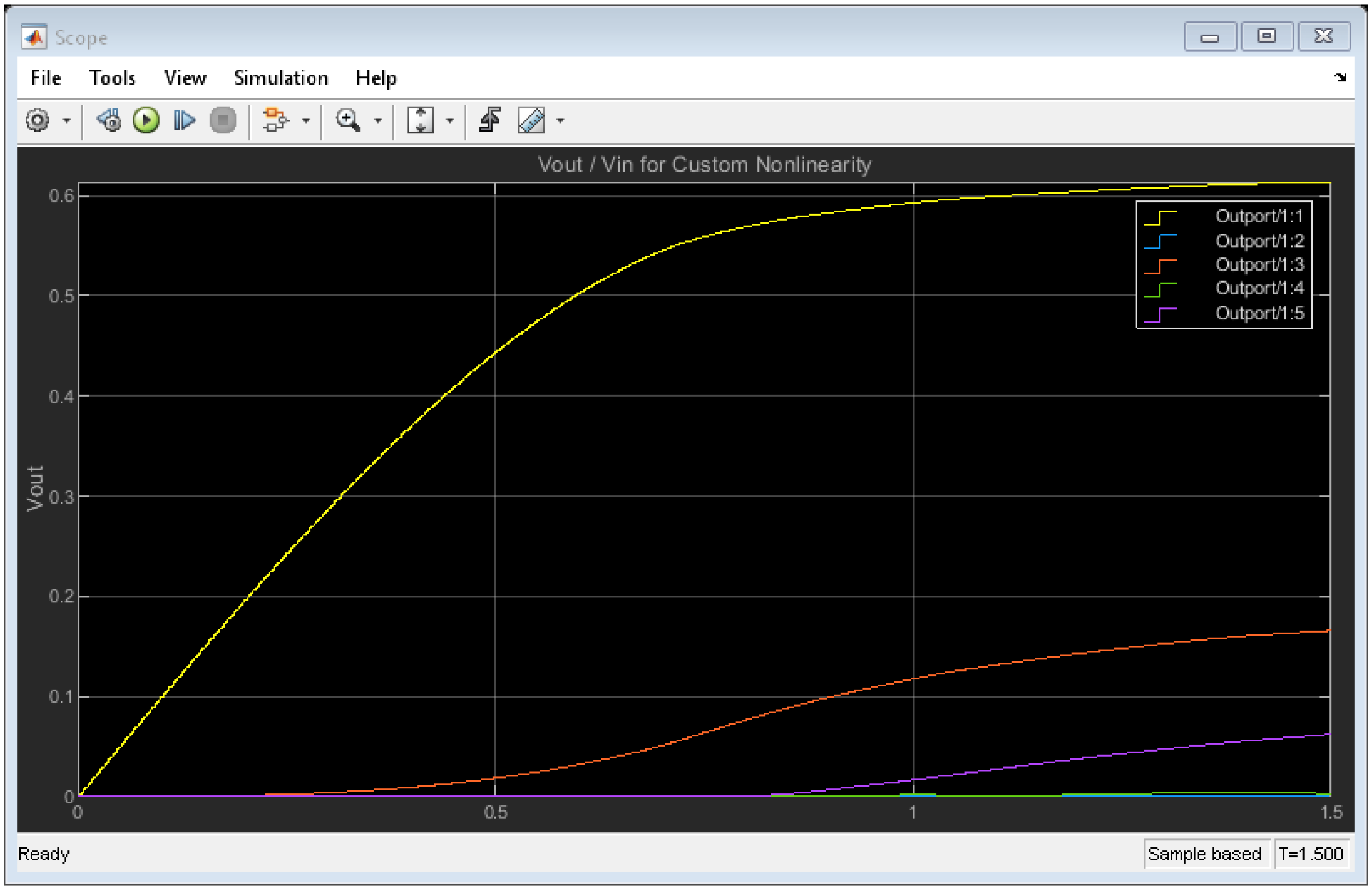
Observe the response produced by the cubic polynomial specified in the 'Custom Nonlinearity' model. The saturated output voltage occurs at time `0.7` seconds and corresponds to the input voltage of `0.7 V`.

Run the Model with Non-zero Input Carrier

Set the input carrier frequency to `1 GHz` and the output frequencies to the first five harmonics of the input. For non-zero carrier input frequency, RF Blockset interprets the input as a complex baseband signal. This complex baseband signal only has a in-phase part specified.

Type the following at the Command Prompt:

```
Input_Freq = 1e9;
Output_Freqs = (1:5)*Input_Freq;
sim(model);
```



Since the coefficients **c0** and **c2** are zero, the output has only odd harmonics (`1 GHz`, `3 GHz` and `5 GHz`) until the output voltage reaches saturation. Other harmonics are introduced for large values of input voltage because of saturation effects.

The relationship between the output curves, polynomial coefficients and IP2/IP3/P1db coefficients is well-studied in the literature [1,2].

Conclusion

An RF Blockset model can be written as a time-domain electrical model in the Simscape language. The model equation can include many types of characteristics, such as derivatives and history (not shown in this example). As with any other model description language, the modeler is responsible for the validity of the model:

- The equations are consistent.
- The equations cannot be degenerate, unstable, or discontinuous. Avoid negative resistances, large nonlinearities and sharp transitions.
- The model does not produce convergence errors during the simulation.

Bibliography

1 Kundert, Ken. "Accurate and Rapid Measurement of IP2 and IP3." *The Designers Guide Community*, Version 1b, May 22, 2002.

2 Chen, Jesse. "Modeling RF systems." *The Designers Guide Community*, Version 1, 6 March 2005.

```
bdclose(model)
```

See Also

Configuration | Inport | Outport

Related Topics

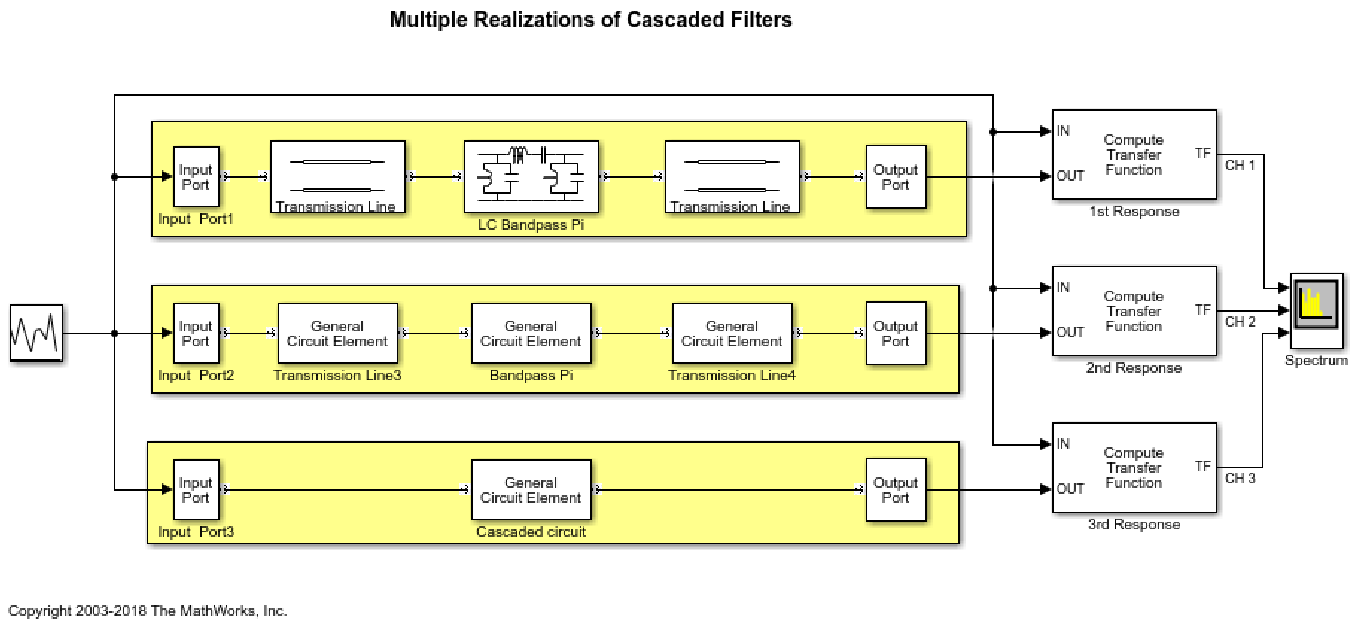
“Getting Started with RF Modeling” on page 9-2

Multiple Realizations of Cascaded Filters

This model shows three different ways to use RF Blockset™ Equivalent Baseband library blocks and RF Toolbox™ objects to implement filters.

Example Model

Example model of multiple realizations of cascaded filters:



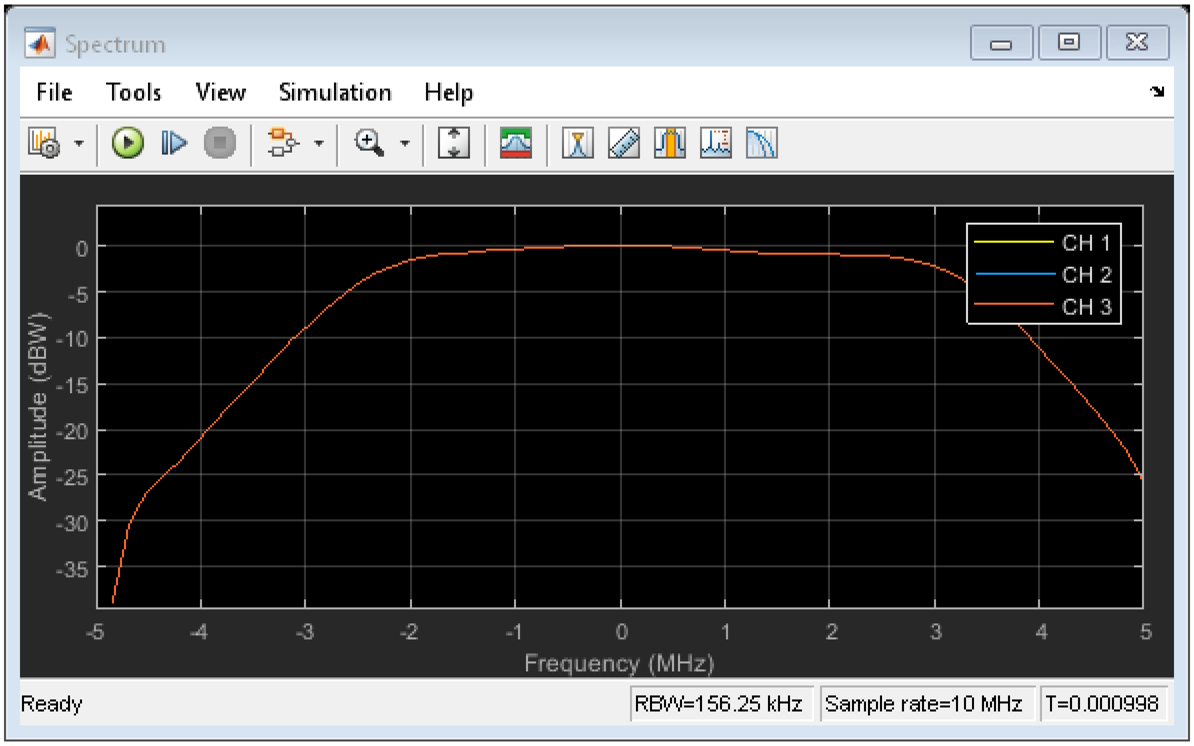
The first method creates a filter using transmission line blocks from the Transmission Lines sublibrary and a LC Bandpass Filter block from the Ladder Filters sublibrary.

The second method creates a filter using three General Circuit Element blocks and three RF circuit objects.

The third method creates a filter using a General Circuit Element block and an RF circuit object, which represents a cascade of three RF circuit objects.

Simulation Results

The spectrum scope shows the frequency responses for the three filter implementations from 70 MHz (-5.0 MHz) to 80 MHz (5.0 MHz).



Ready
RBW=156.25 kHz
Sample rate=10 MHz
T=0.000998

See Also

LC Bandpass Pi | Transmission Line (Equivalent Baseband)

Related Topic

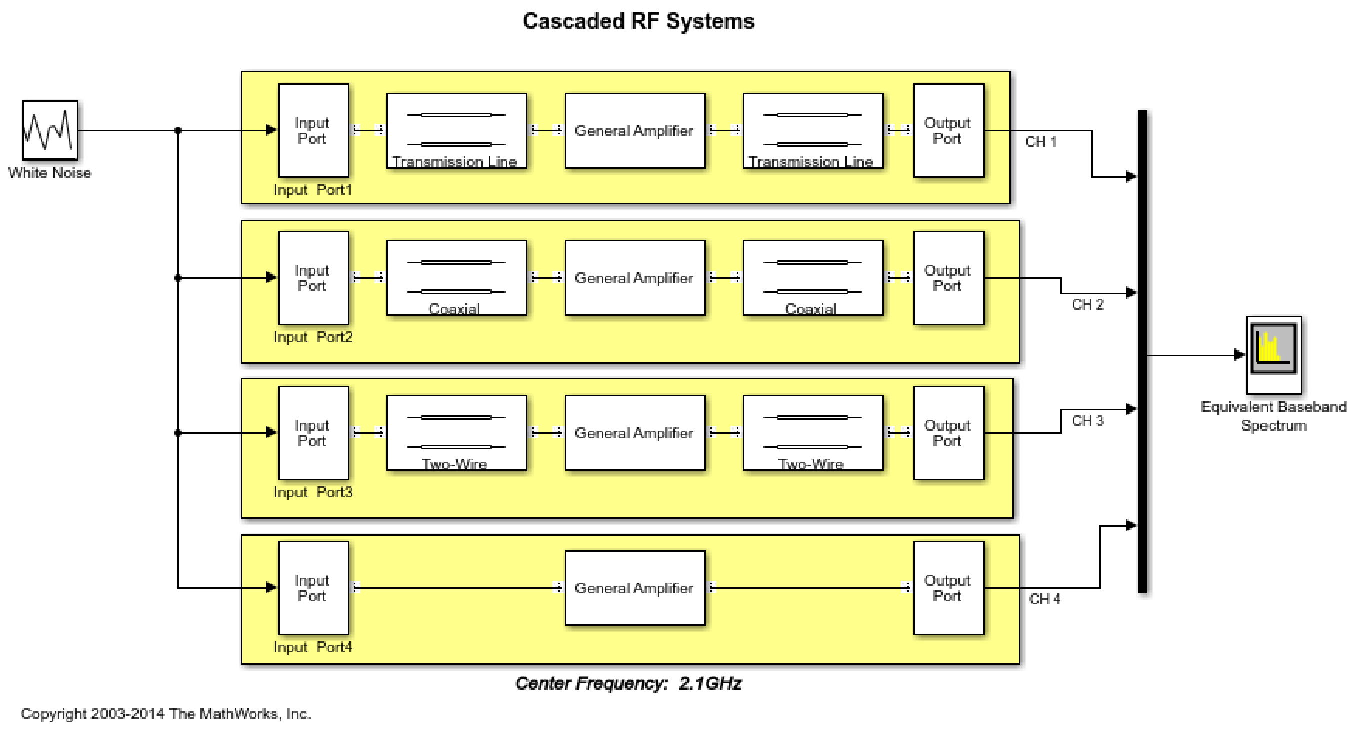
"Cascaded RF Systems" on page 9-115

Cascaded RF Systems

This model shows how to use blocks from the RF Blockset™ Equivalent Baseband library to build cascaded RF systems.

Example Model

Fig. 1 Example model of cascaded RF systems:

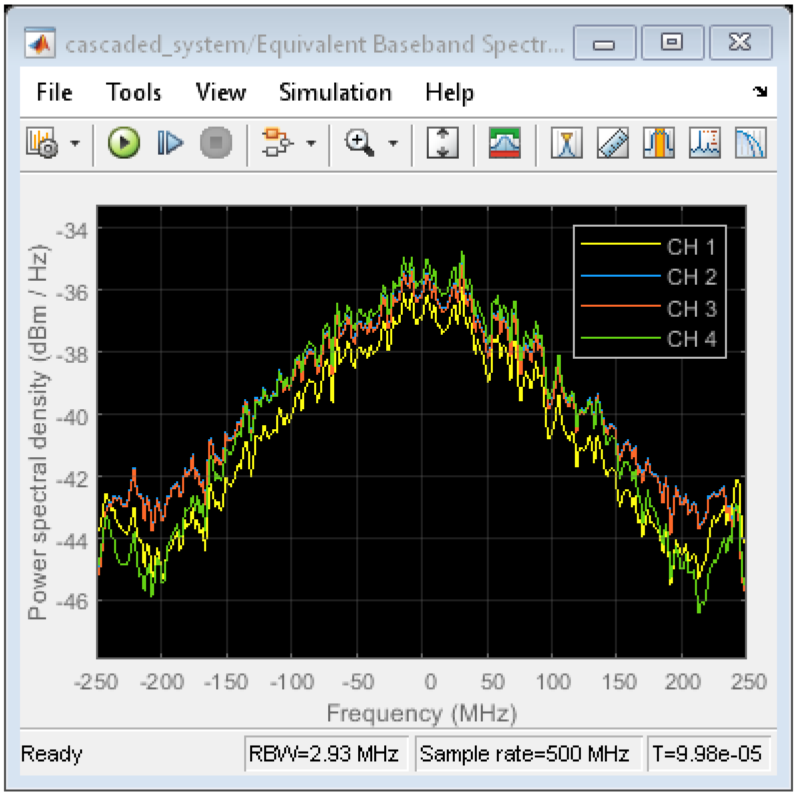


Simulation Results

In the first three systems, an amplifier with S-parameters taken from the default.s2p data file is cascaded with different types of transmission lines. The fourth system represents the amplifier itself.

The following figure shows the equivalent baseband frequency response of four RF systems ranging from 1.85 GHz (-250 MHz) to 2.35 GHz (250 MHz).

Fig. 2 Frequency response:



To see the frequency response centered at another RF frequency, change the Center frequency parameter in the Input Port block.

See Also

Equivalent Baseband Transmission Line | LC Bandpass Pi

More About

- “Multiple Realizations of Cascaded Filters” on page 9-113

Power in Simulink® Sources and Signals

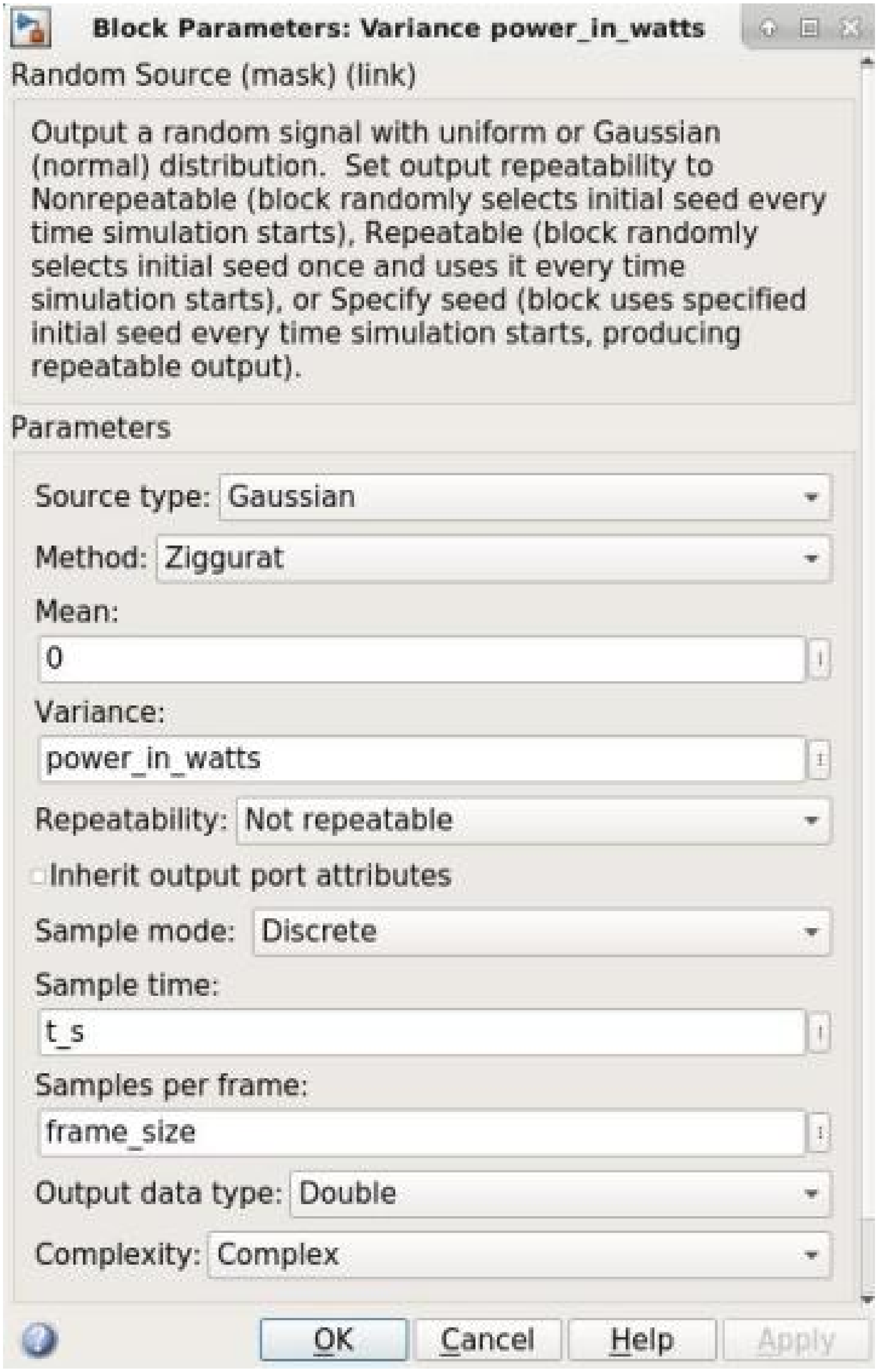
This example shows how to use the Input Port and Output Port blocks of the RF Blockset™ Equivalent Baseband library to convert between dimensionless Simulink signals and equivalent-baseband signals.

In general, signals in Simulink are dimensionless, so their amplitudes do not correspond to a particular voltage or power. However, in an RF system, power is a quantity of interest. When you use blocks from the RF Blockset Equivalent Baseband library in a Simulink model, you must specify how the software interprets the Simulink signals that exist outside the boundaries of the Input Port and Output Port blocks. RF Blockset Equivalent Baseband software provides two options to interpret the Simulink signal: power wave or voltage. The amplitude of a source in Simulink determines the signal power level and affects the signal power and power spectrum.

All the models used in this example interpret the Simulink signal as a power wave with dimensions of $W^{0.5}$. This means for an RF system, the source signal generated by regular Simulink blocks is treated as the incident power wave to the RF system, and the RF output signal is the transmitted power wave of the RF system. If you choose to interpret the Simulink signal as a voltage, you need to modify the models by considering the impedance effects when you calculate the powers. For more details, see Converting to and from Simulink Signals.

White Noise Source

This part of the example shows how to create a complex baseband-equivalent White Noise Source. This type of source is useful, for example, as a stimulus for visualizing the frequency response of an RF system. Use a Random Source block from the DSP System Toolbox™ Sources sublibrary to create this source. In the Random Source block dialog box, set the **Complexity** parameter to `Complex` and in the **Variance** parameter enter the desired noise power in watts using the expression `power_in_watts`.



Block Parameters: Variance power_in_watts

Random Source (mask) (link)

Output a random signal with uniform or Gaussian (normal) distribution. Set output repeatability to Nonrepeatable (block randomly selects initial seed every time simulation starts), Repeatable (block randomly selects initial seed once and uses it every time simulation starts), or Specify seed (block uses specified initial seed every time simulation starts, producing repeatable output).

Parameters

Source type: Gaussian

Method: Ziggurat

Mean: 0

Variance: power_in_watts

Repeatability: Not repeatable

☐ Inherit output port attributes

Sample mode: Discrete

Sample time: t_s

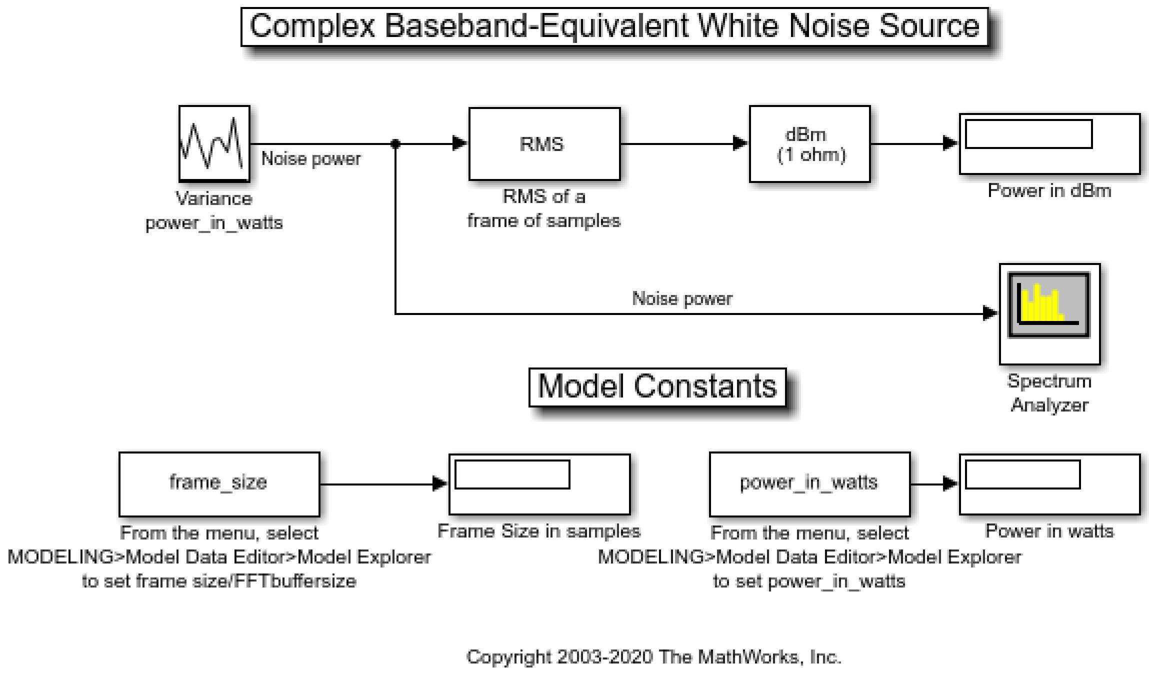
Samples per frame: frame_size

Output data type: Double

Complexity: Complex

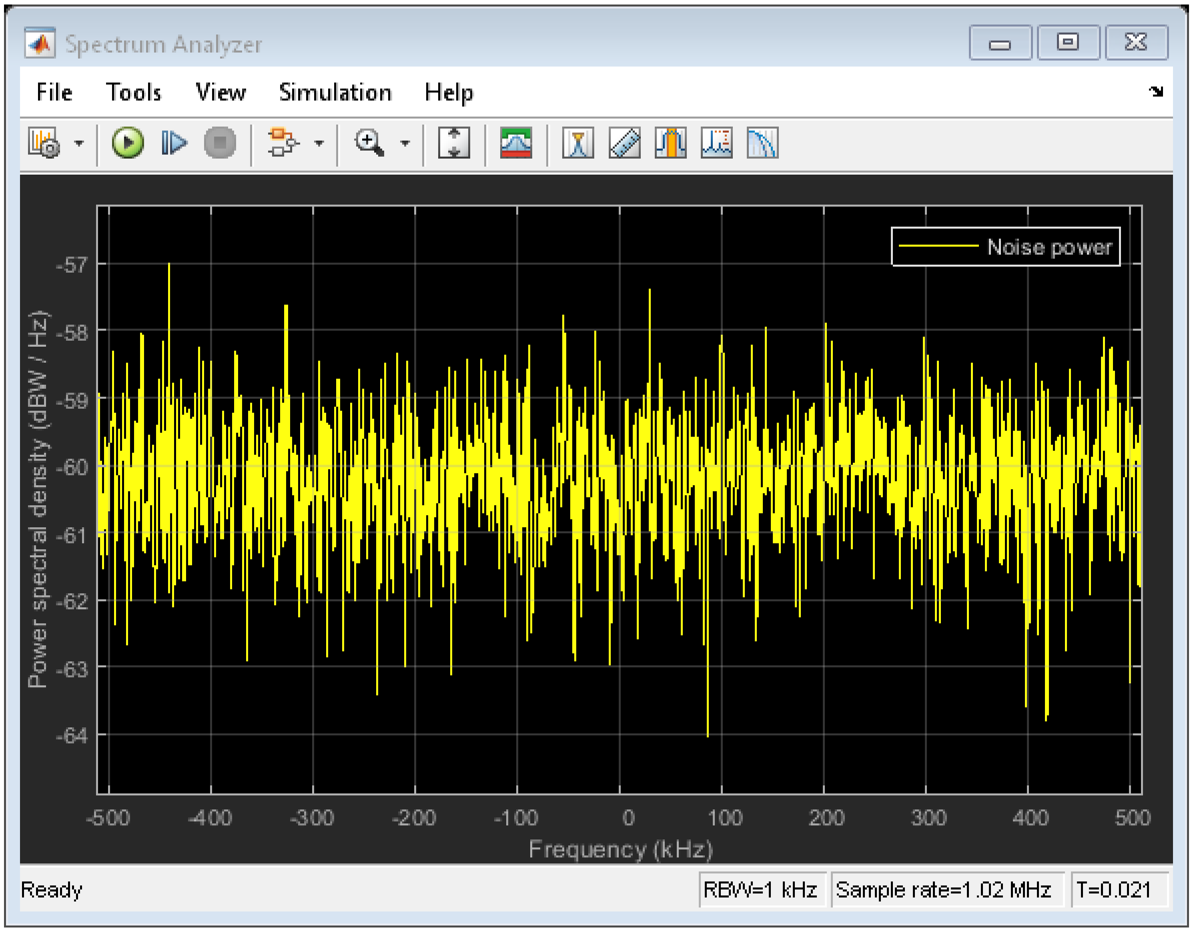
To calculate a signal power in dBm, use an RMS block (from the DSP System Toolbox Statistics sublibrary), followed by a dB Conversion block (from the DSP System Toolbox Math Functions/Math Operations sublibrary). In the dB Conversion block dialog box, set **Convert to** parameter to `dBm` , **Input Signal** parameter to `Amplitude` , and **Load resistance (ohms)** parameter to `1` .

```
open_system('rfb_power_examples_white_noise')
```



To display the power spectrum of a signal, use the Spectrum Analyzer block (from the RF Blockset Circuit Envelope Utilities sublibrary). To show the Spectrum Settings, in the Spectrum Analyzer menu, select **View** > **Spectrum Settings** or use the far left button in the toolbar. **Two-sided spectrum** is checked by default in the trace options. This is the desired frequency range because a complex baseband-equivalent representation translates the carrier frequency to zero hertz. The real-world frequencies above and below the carrier (i.e. higher and lower sidebands) are represented as positive and negative frequencies, respectively. A few of the **Spectrum Settings** options (**Type**, **Window**, **Units**, **Averaging method**, **Averages**) have been changed from their default.

```
sim('rfb_power_examples_white_noise')
```



In addition, note that the selected Spectrum Scope Window Type can affect how the power is distributed amongst the channels closest to the actual frequency. For example, if a pure sine wave falls between two channels, you may need to sum the power in one or two channels either side of the actual frequency to determine the exact total power.

Complex Sine Source

The next model, of a Complex Sine Wave, shows how to use power to set the amplitude of a complex sine wave source block for an RF system. Complex sine wave sources are often used in baseband-equivalent Simulink models. These sources have the following time-domain output:

```
signal(t) = amplitude * (cos(2*pi*f*t+phi)+j*sin(2*pi*f*t+phi))
```

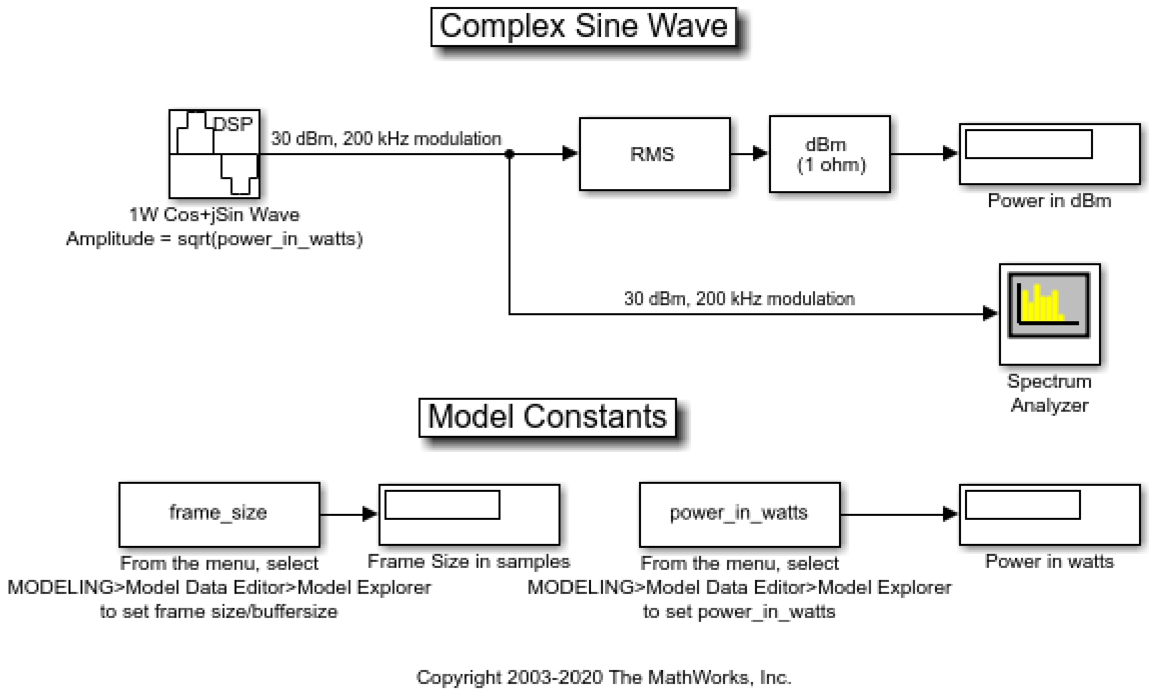
The mean square power of the output, `signal`, is `amplitude^2`.

By contrast, the time-domain output of a real sine wave source is:

```
signal2(t) = amplitude * sin(2*pi*f*t+phi)
```

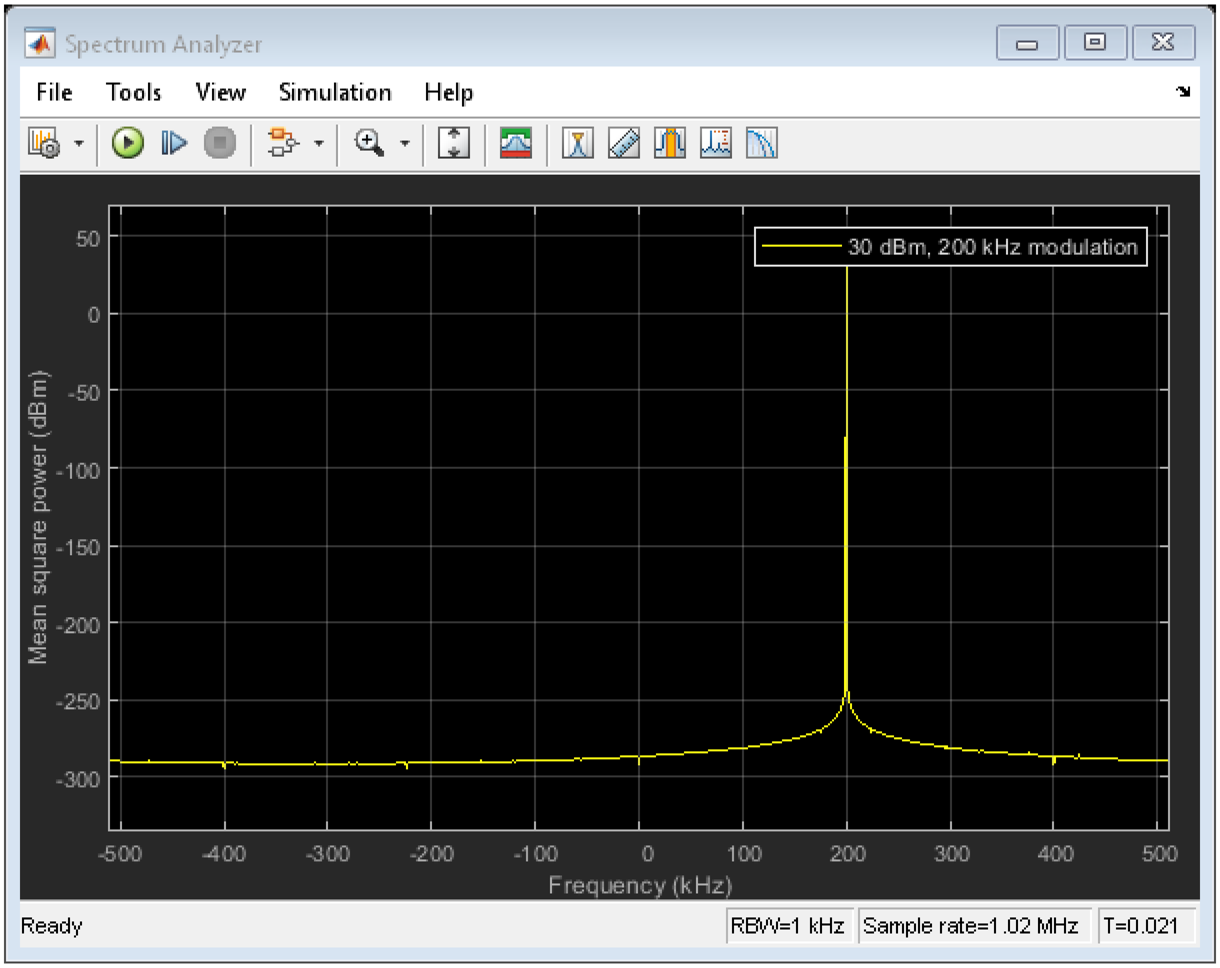
where the mean square power of `signal2` is `amplitude^2/2`, half that of a complex sine wave with the same `amplitude`.

```
bdclose('rfb_power_examples_white_noise');
open('rfb_power_examples_cis_wave.slx')
```


Complex Sine Wave
DSP
30 dBm, 200 kHz modulation
RMS
dBm
(1 ohm)
Power in dBm
1W Cos+jSin Wave
Amplitude = sqrt(power_in_watts)
30 dBm, 200 kHz modulation
Spectrum
Analyzer
Model Constants
frame_size
From the menu, select
MODELING>Model Data Editor>Model Explorer
to set frame size/buffersize
Frame Size in samples
power_in_watts
From the menu, select
MODELING>Model Data Editor>Model Explorer
to set power_in_watts
Power in watts
Copyright 2003-2020 The MathWorks, Inc.

Use the Sine Wave block to create a complex sine source. In the block dialog box, set the **Output complexity** parameter to `Complex` and the **Amplitude** parameter to `sqrt(power_in_watts)`. By default, the Spectrum Analyzer displays power spectral density normalized to the unit sampling frequency in units of dBm/Hertz. For this section, the Spectrum Scope displays the tone as a positive frequency (upper side band).

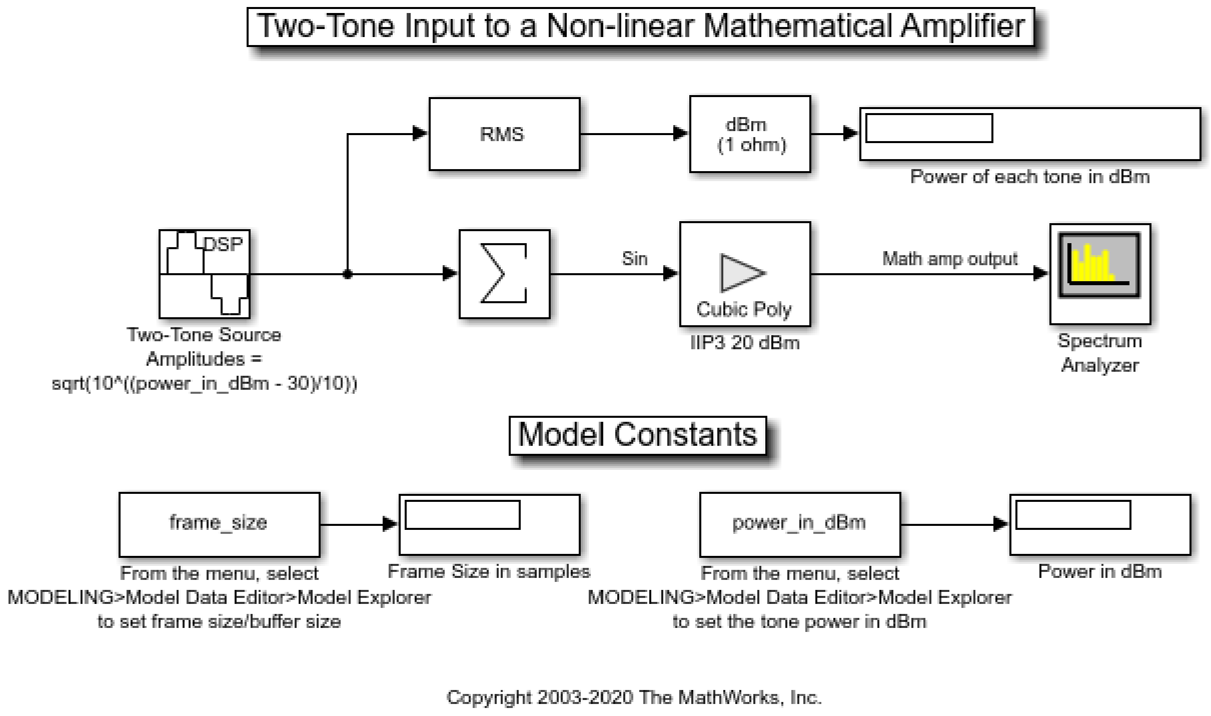
```
sim('rfb_power_examples_cis_wave')
```



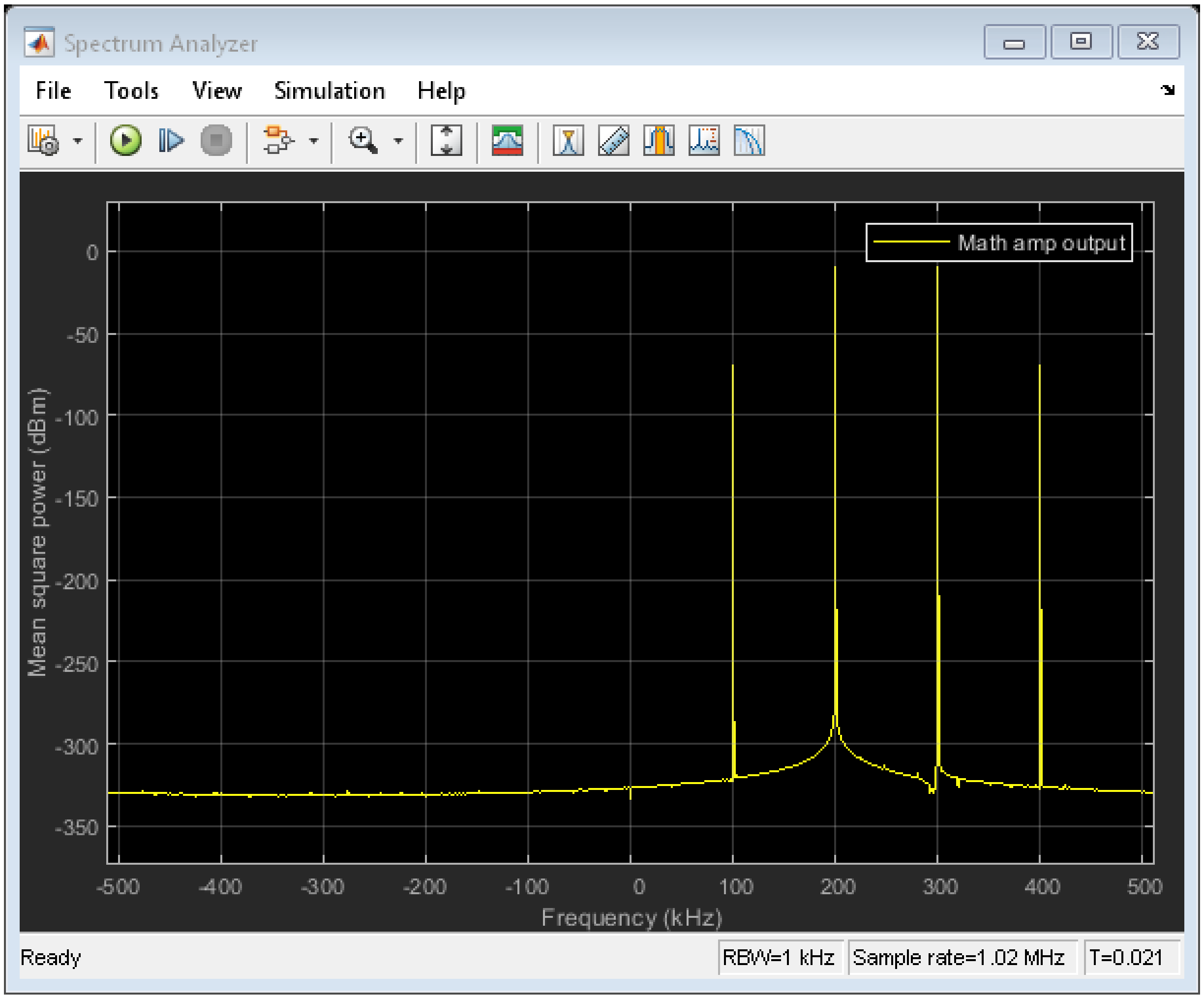
Two-Tone Input to Idealized Baseband Nonlinear Amplifier

The third model, of a Two-Tone Input to an Idealized Baseband Nonlinear Amplifier, shows how the Amplifier block in the RF Blockset Idealized Baseband library affects the signal. In the Amplifier block dialog box set the **IIP3 (dBm)** parameter to `20` dBm. In the Sine Wave block dialog box, set **Amplitude** parameter to `sqrt(10^((power_in_dBm - 30)/10))`. Setting `power_in_dBm = -10` in the model workspace results in -10 dBm per tone. Note that we must use a Matrix Sum block with the **Sum along** parameter set to "Rows" after the source block to sum the two-channel output of the source. Without the Row Sum, a two-channel signal would be created, all blocks downstream would have two independent channels, and no mixing would occur.

```
bdclose('rfb_power_examples_cis_wave');
open('rfb_power_examples_two_tone_math_amp.slx')
```


Two-Tone Input to a Non-linear Mathematical Amplifier
RMS
dBm
(1 ohm)
Power of each tone in dBm
DSP
Two-Tone Source
Amplitudes =
sqrt(10^((power_in_dBm - 30)/10))
Sin
Cubic Poly
IIP3 20 dBm
Math amp output
Spectrum
Analyzer
Model Constants
frame_size
From the menu, select
MODELING>Model Data Editor>Model Explorer
to set frame size/buffer size
Frame Size in samples
power_in_dBm
From the menu, select
MODELING>Model Data Editor>Model Explorer
to set the tone power in dBm
Power in dBm
Copyright 2003-2020 The MathWorks, Inc.

```
sim('rfb_power_examples_two_tone_math_amp')
```



The Spectrum Scope displays the power level in each intermodulation tone. The power level of each is:

`-10dBm - 2*(20dBm - -10dBm) = -70dBm`.

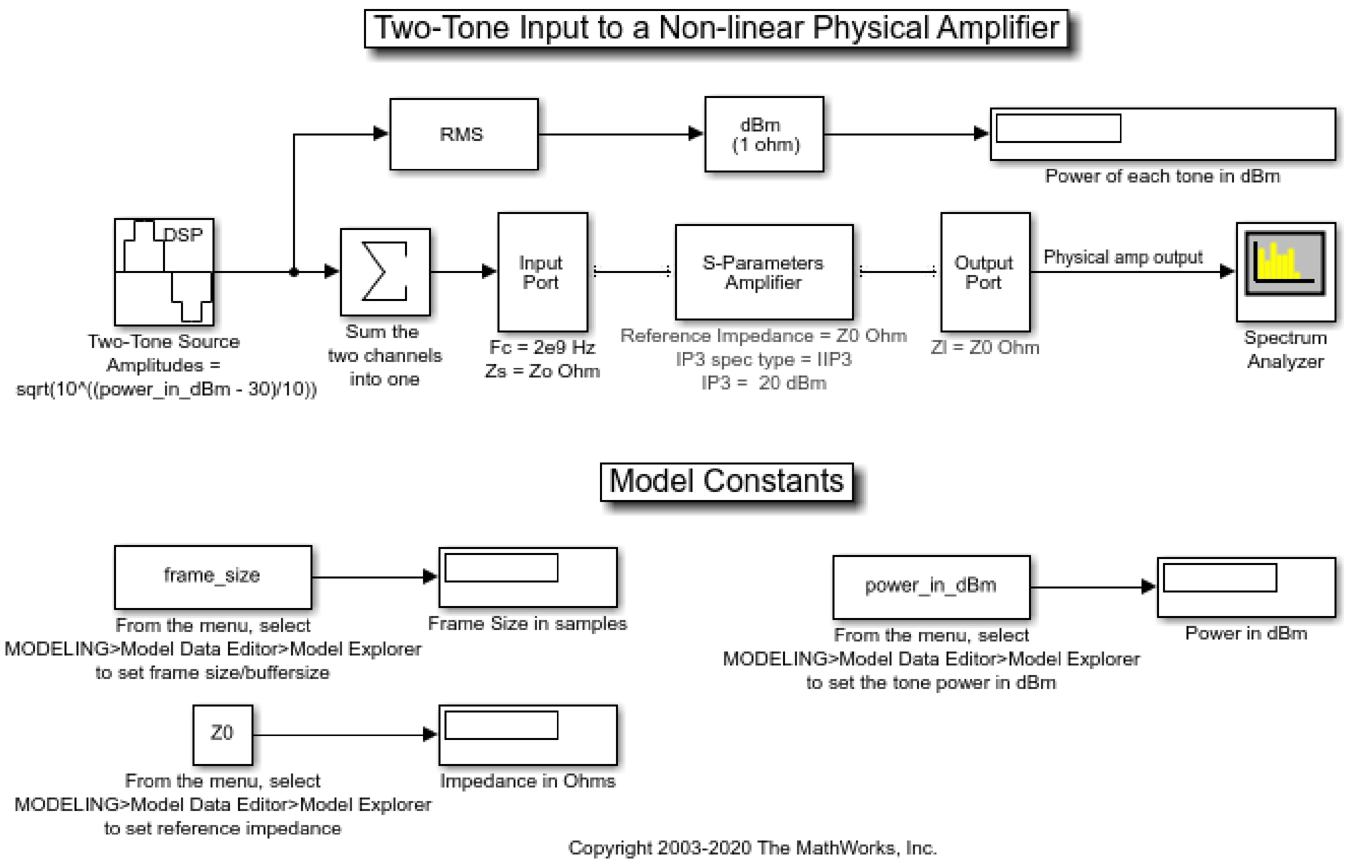
Two-Tone Input to Equivalent Baseband Nonlinear Amplifier

Like the third model, the fourth model, of a Two-Tone Input to an Equivalent Baseband Nonlinear Amplifier, shows how an Amplifier block affects the signal. However, this time we use the S-Parameters Amplifier block from the Equivalent Baseband library of RF Blockset. Unlike the Idealized Baseband blocks, the Equivalent Baseband blocks allow you to set the center frequency and impedances. Thus, if you want to model an RF system at real RF frequencies, the loading and reflection effects, we recommend these physical blocks.

In this model, we set several parameters to `Z0`:

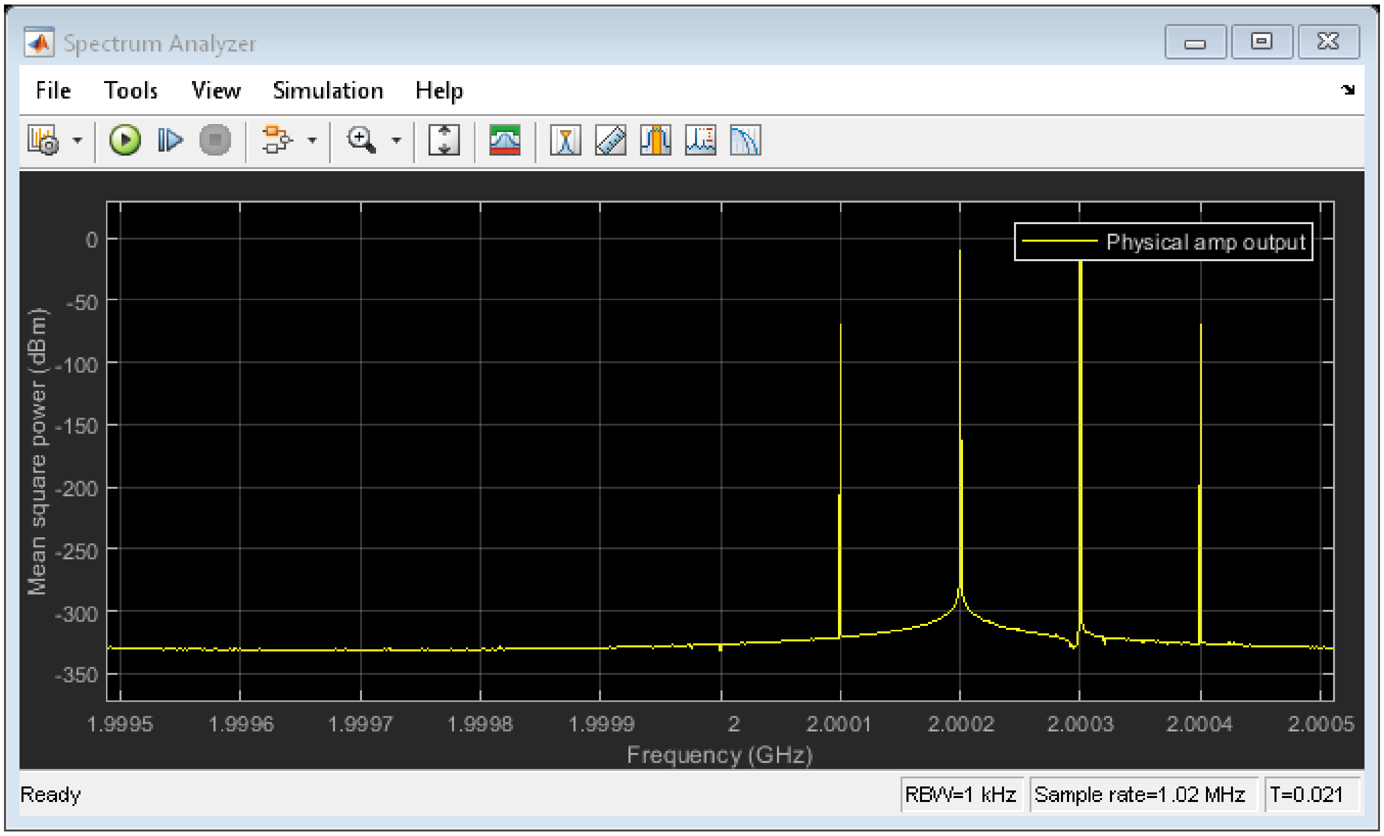
- **Source impedance** parameter of the Input Port block dialog box
- **Reference impedance** parameter of the S-Parameters Amplifier block dialog box
- **Load impedance** parameter of the Output Port block dialog box

```
bdclose('rfb_power_examples_two_tone_math_amp');
open('rfb_power_examples_two_tone_physical_amp.slx')
```


Two-Tone Input to a Non-linear Physical Amplifier
RMS
dBm (1 ohm)
Power of each tone in dBm
DSP
Two-Tone Source
Amplitudes = sqrt(10^((power_in_dBm - 30)/10))
Sum the two channels into one
Input Port
Fc = 2e9 Hz
Zs = Zo Ohm
S-Parameters Amplifier
Reference Impedance = Z0 Ohm
IP3 spec type = IIP3
IP3 = 20 dBm
Output Port
Zl = Z0 Ohm
Physical amp output
Spectrum Analyzer
Model Constants
frame_size
Frame Size in samples
From the menu, select MODELING>Model Data Editor>Model Explorer to set frame size/buffersize
power_in_dBm
Power in dBm
From the menu, select MODELING>Model Data Editor>Model Explorer to set the tone power in dBm
Z0
Impedance in Ohms
From the menu, select MODELING>Model Data Editor>Model Explorer to set reference impedance
Copyright 2003-2020 The MathWorks, Inc.

In the Input Port block, set the **Center frequency (Hz)** parameter to `2e9` (2 GHz). The baseband frequencies of the two-tone complex Simulink signal are 200 kHz and 300 kHz. Thus, in the RF system (the Equivalent Baseband blocks connected between the Input Port and Output Port blocks), the real RF two-tone frequencies are 2.0002 GHz and 2.0003 GHz. By default, the Spectrum Scope displays in baseband. To display the desired tones at 2.0002 GHz and 2.0003 GHz (-10 dBm each) and the intermodulation tones at 2.0001 GHz and 2.0004 GHz (-70 dBm each), set the **Frequency display offset:** parameter in the "Axis Properties" tab to the value of the Center frequency, in this case it is 2e9 (2 GHz).

```
sim('rfb_power_examples_two_tone_physical_amp')
```

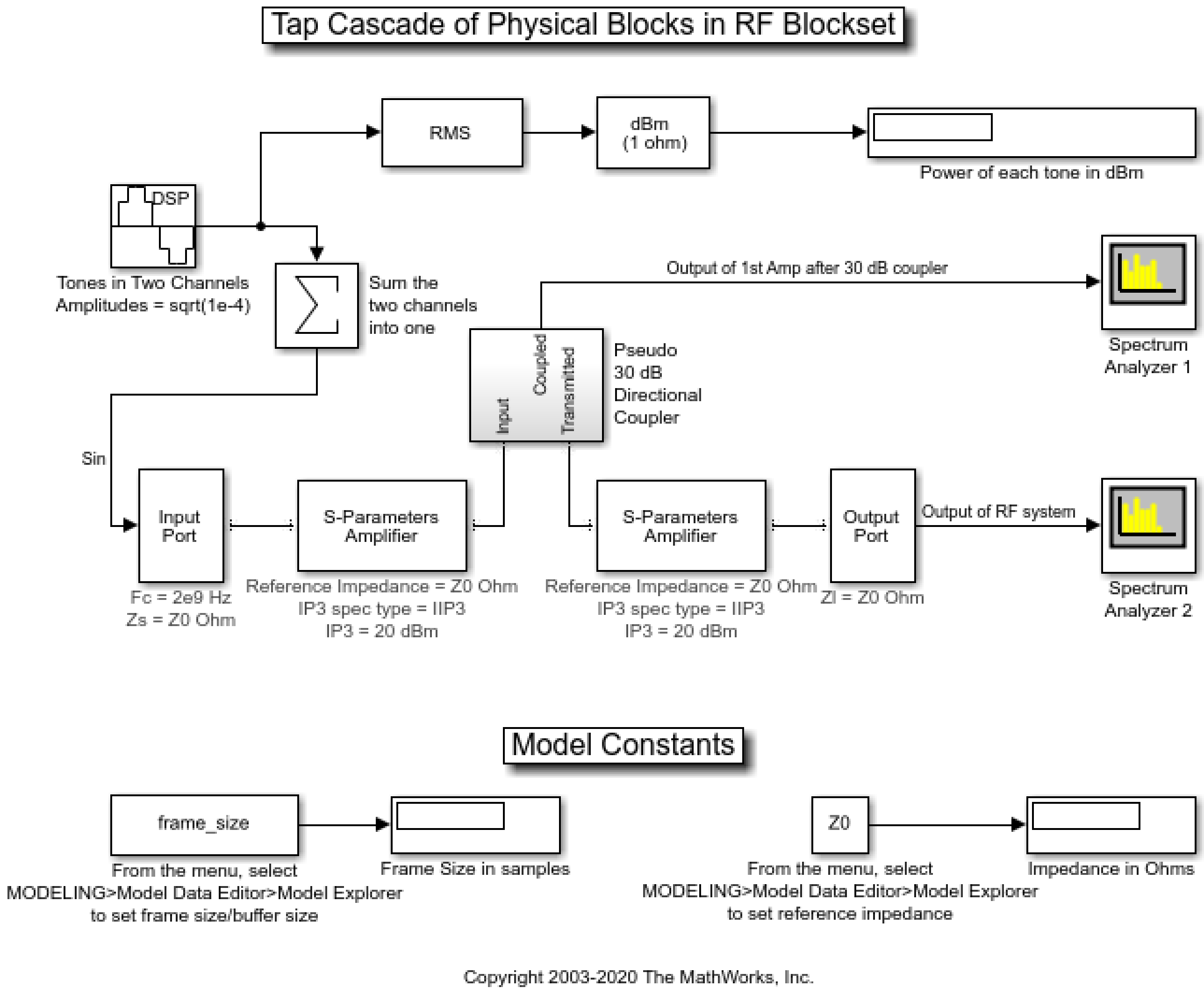


Displaying Power Spectrum Within Cascade of RF Blockset Equivalent Baseband Blocks

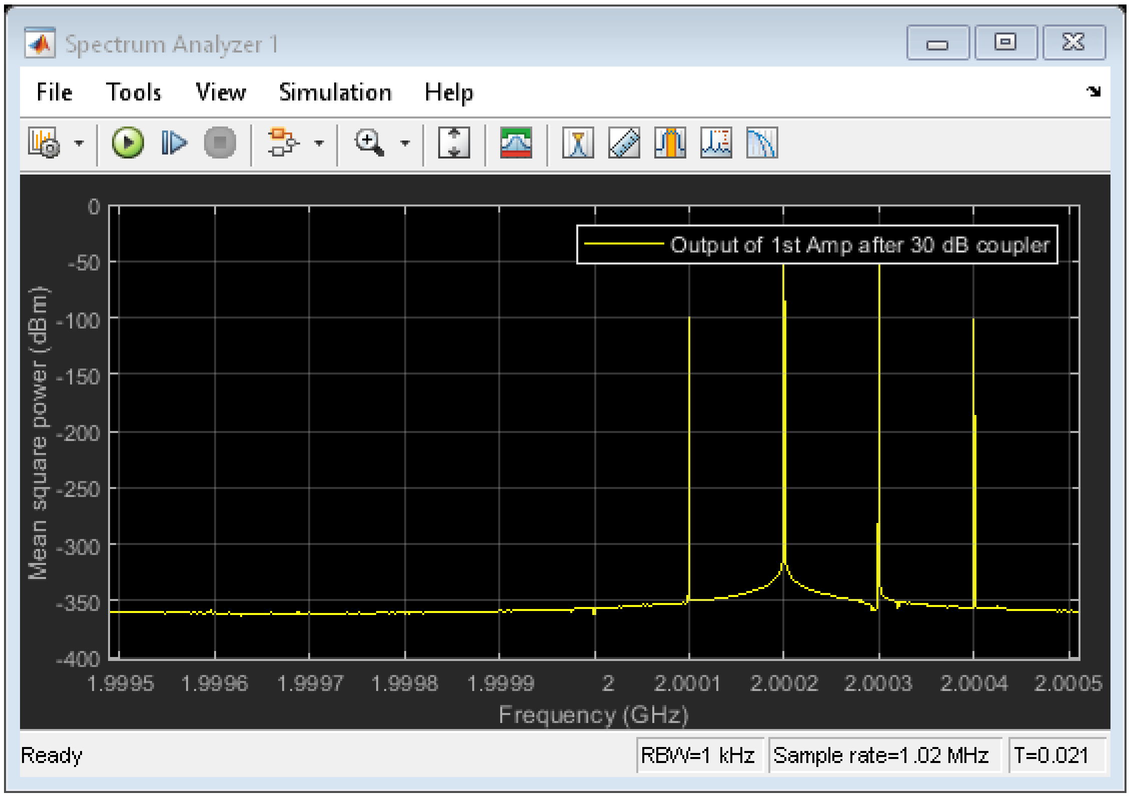
The Output Port block lets you create a link budget plot for multi-block cascades. This feature allows you to visualize the characteristics of the cascade non-intrusively. Therefore, it is not usually necessary to tap a cascade of RF Blockset Equivalent Baseband blocks. However, it is sometimes useful to do so, for example, to see the modulated spectrum at an intermediate point. The final model of a Tap Cascade of Equivalent Baseband Blocks in RF Blockset, achieves this with a subsystem that approximately models a real-world directional coupler. As with its real-world counterpart, the tapping is intrusive in that it presents a load impedance to the downstream part of the cascade and it drives the upstream part with a source impedance.

Double click on the subsystem "Pseudo 30 dB Directional Coupler" to open it and see how the model works. The Output Port and Input Port blocks correspond to the input and output impedance of the mainline of a real-world directional coupler, respectively. However, the phase behavior of a real-world directional coupler is not modeled here.

```
bdclose('rfb_power_examples_two_tone_physical_amp');
open('rfb_power_examples_tap_cascade.slx')
```

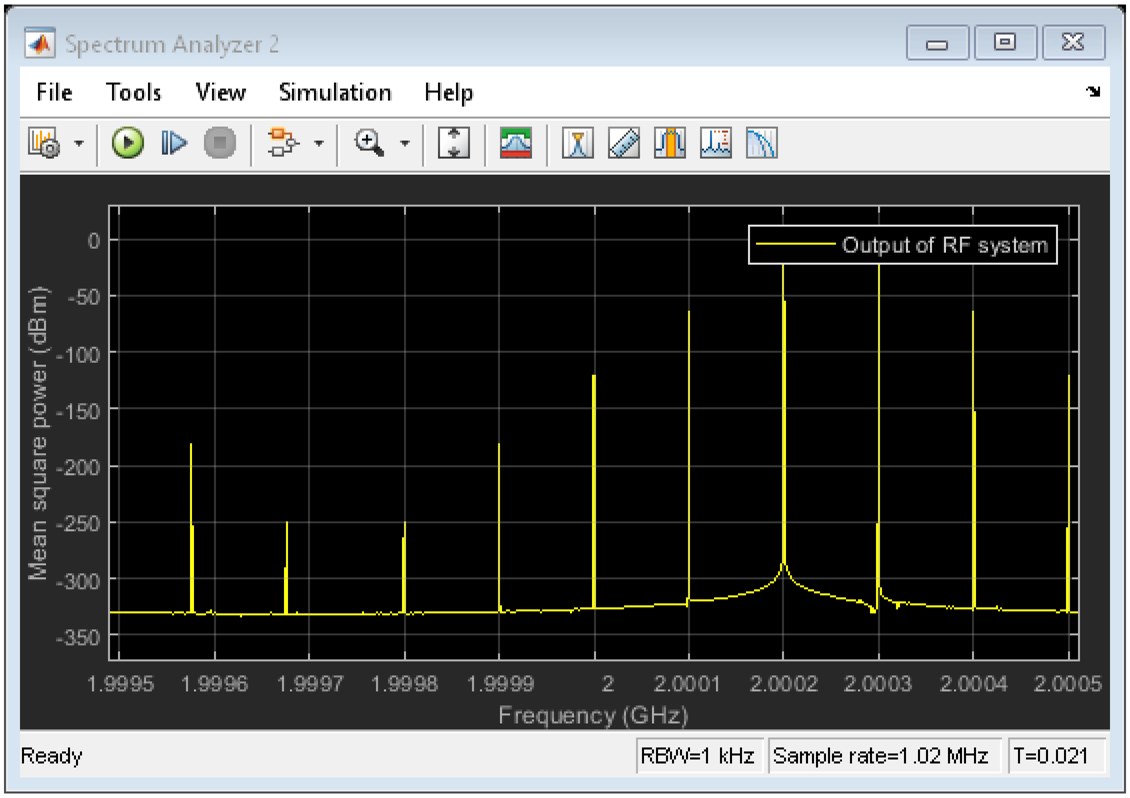


```
sim('rfb_power_examples_tap_cascade')
close_system(['rfb_power_examples_tap_cascade/Spectrum',newline,'Analyzer 2']);
```



The first Spectrum Scope shows the intermodulation tones after one amplifier. Note that the power is 30 dB down because of the characteristics of the "Pseudo 30 dB Directional Coupler" subsystem. You could calibrate this out with a gain block, or even modify the subsystem to model a 0 dB-loss "active" directional coupler.

```
open_system(['rfb_power_examples_tap_cascade/Spectrum',newline,'Analyzer 2']);
```



The second Spectrum Scope shows an increased level of intermodulation tones after a cascade of two non-linear amplifiers.

```
bdclose('rfb_power_examples_tap_cascade');
```

See Also

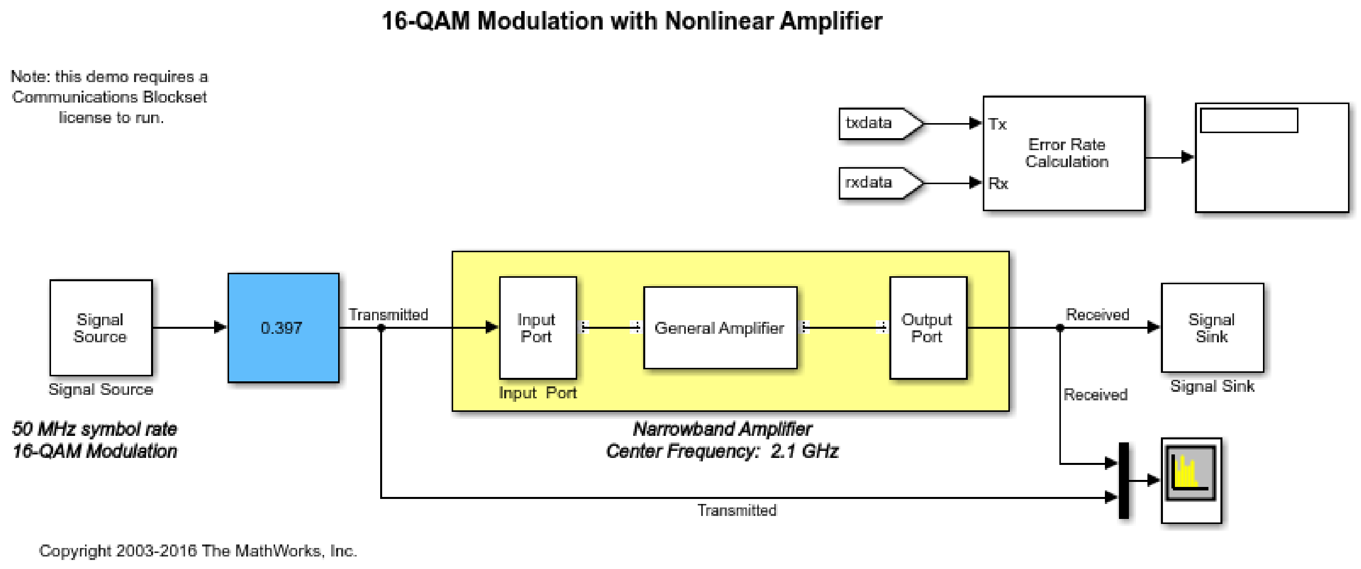
“Power Ports and Signal Power Measurement in RF Blockset” on page 9-11

Effect of Nonlinear Amplifier on 16-QAM Modulation

This model shows the nonlinear effect of a RF Blockset™ Equivalent Baseband amplifier on a 16-QAM modulated signal.

Example Model

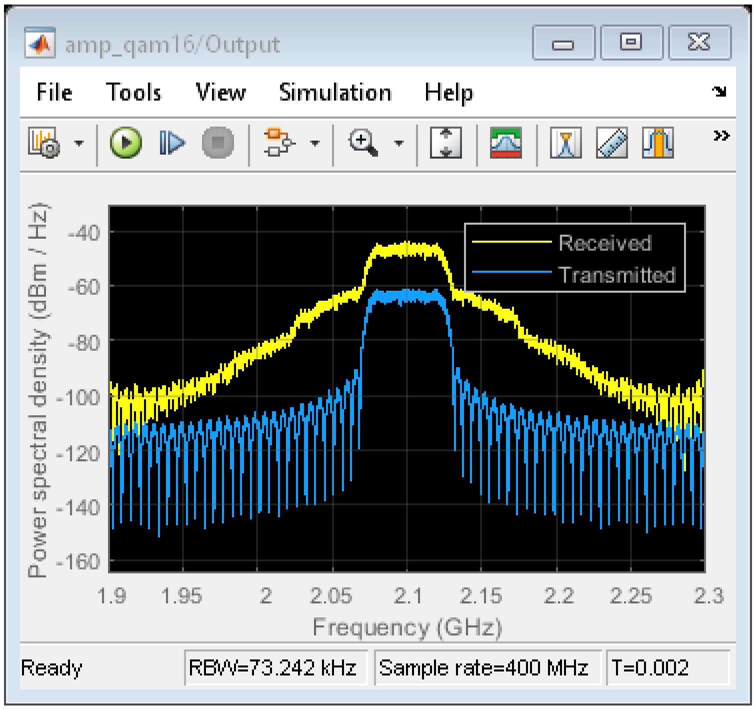
Fig. 1 16-QAM modulation example model:



Simulation Results

When you compare the spectrums of transmitted and received signals (Fig. 2), you can see the spectrum regrowth at the received signal. This regrowth is due to the nonlinearity of the amplifier.

Fig. 2 Spectrums of transmitted and received signals:



When you compare the constellations of transmitted and received signals (Figs. 3 and 4 respectively), you can see the signal distortion at the received signal. This distortion is due to the nonlinearity of the amplifier.

Fig. 3 Constellation of transmitted signal:

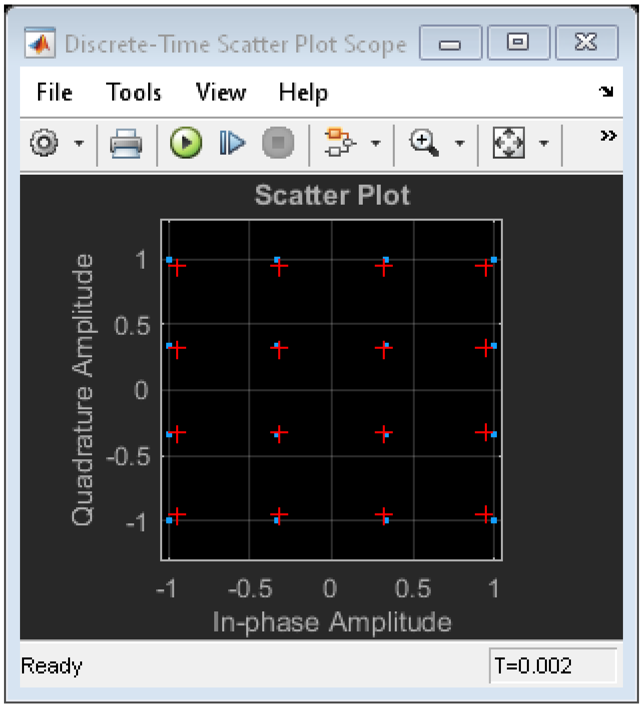
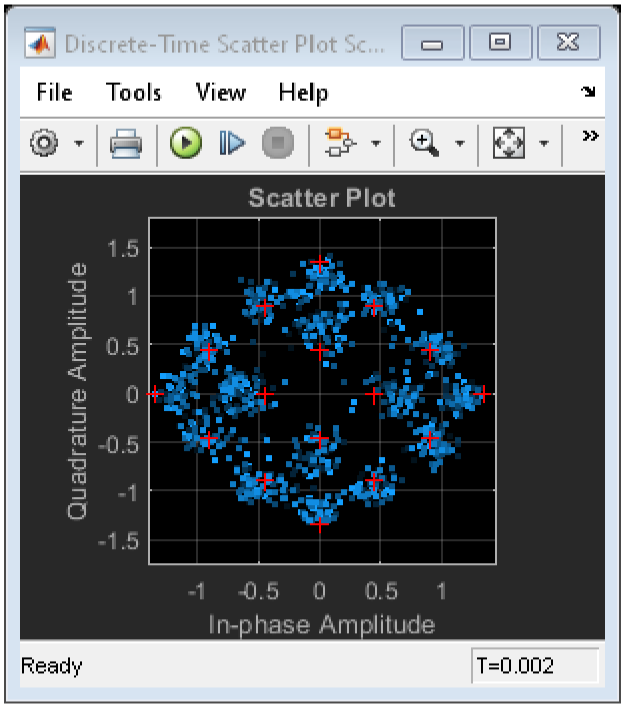


Fig. 4 Constellation of received signal:



To view the nonlinear effect, double-click the Slider Gain block and change the value of the gain while the model is running.

See Also

General Amplifier | Input Port | Output Port

Related Topics

“User-Defined Nonlinear Amplifier Model” on page 9-148

Executable Specification for System Design

This example shows how to use the Model-Based Design methodology to overcome the challenge of exchanging specifications, design information, and verification models between multiple design teams working on a single project. The example uses a simple project: an executable specification that encapsulates information from all teams. The example includes information on how to use Signal Processing Toolbox™, DSP System Toolbox™, Communications Toolbox™, RF Toolbox™, and RF Blockset™ in a multi-domain design.

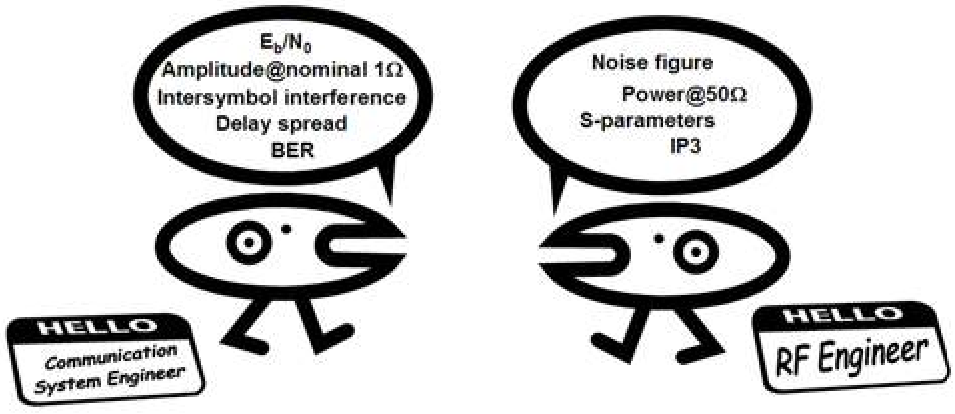


Figure 1: Bridging the Jargon Gap between RF and System Engineers

Model-Based Design

Model-Based Design uses a system-level model at the center of the development process. Before partitioning the system-level model among various design teams, the initial system model, developed by the system engineer, is validated against requirements and standards. With a validated error-free executable specification, design and implementation go smoothly. As the design progresses, verification can include co-simulation and testing with hardware-in-the-loop.

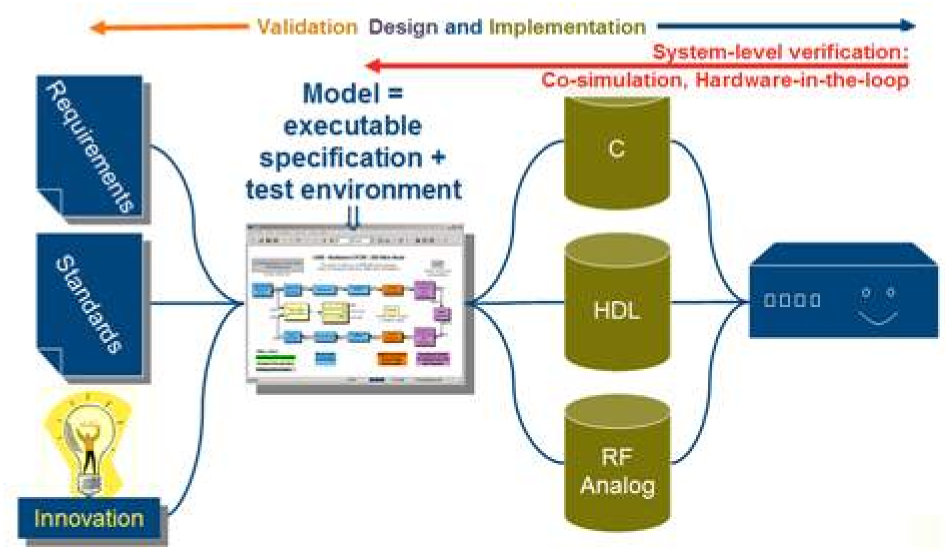
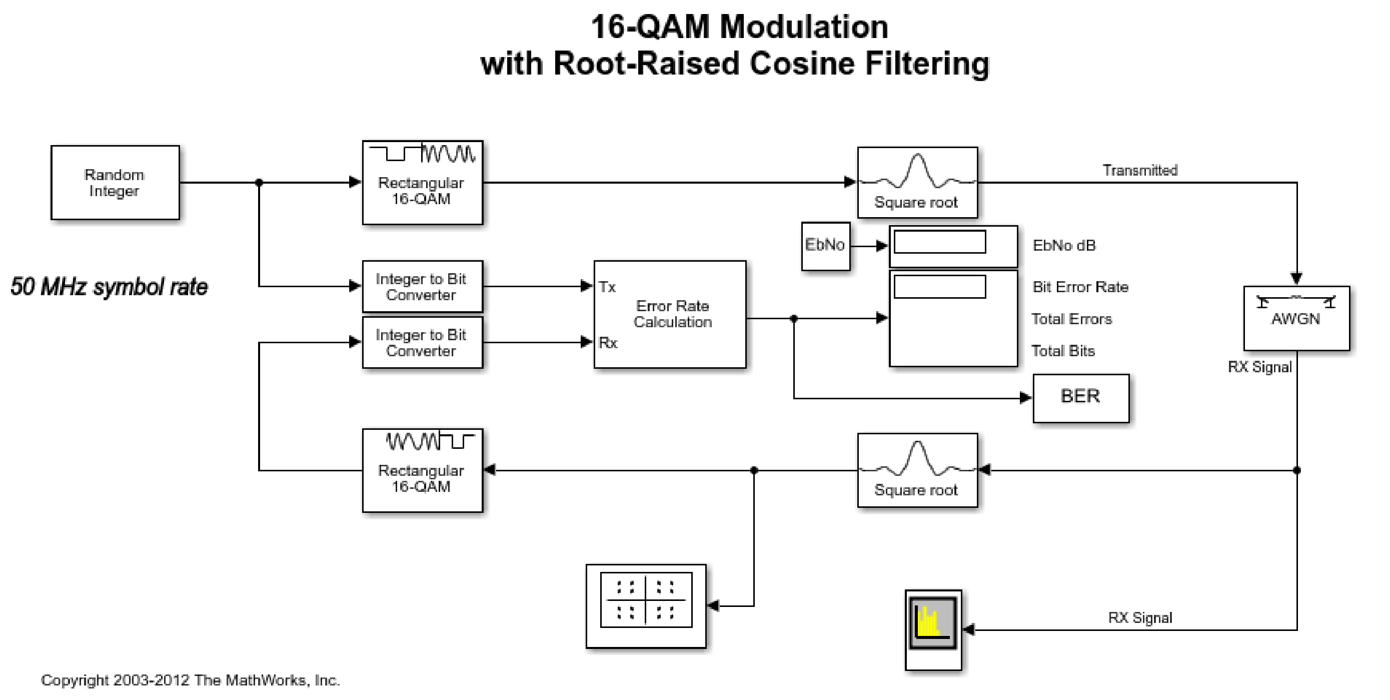

Validation
Design and Implementation
System-level verification:
Co-simulation, Hardware-in-the-loop
Model = executable specification + test environment
Requirements
Standards
Innovation
C
HDL
RF Analog

Figure 2: Model-Based Design -- A system-level model is at the center of the development process

Rather than talking about all the elements in the development flow, this example focuses on how Model-Based Design aids your engineering teams. The idea is to enable the System Engineer to initially create an executable specification in the form of a Simulink model that can be distributed to design teams. A team, such as the RF team, will devise a subsystem, extract a verification model and import it into the RF Toolbox. The RF team then returns the solution to the System Engineer, who reevaluates the overall performance of the system with the impairments from the RF subsystem. The design teams can go back and forth, iterating to find an optimal solution as the design proceeds. Perhaps the RF section can use a more efficient or less costly device if the signal processing algorithms are altered. Or, perhaps a small increase in fixed-point wordlength can free up some of the implementation loss budgeted, and enable a lower cost RF component to be used. The opportunities for cross-domain optimization are enhanced by this Model-Based design methodology.

Baseline Model: Communications Toolbox™ with No RF Modeling

```
open('rfb_receiver_0.slx')
```



The model rfb_receiver_0.slx shows the kind of Communication System Toolbox model that inspired the creation of the RF Blockset Equivalent Baseband library. Note that this is a simple model for illustrative purposes. Communications Toolbox includes more complex models of WCDMA, 802.11, DVB-S2, etc. However, the concepts presented can be applied to more complex models as well.

The simple wireless communication system consists of a message source, QAM modulator, root raised cosine filter and an AWGN channel. The model is an executable specification, and is used to validate the specification against requirements and acceptance criteria, "At a BER of 1e-3, the Eb/No must be no greater than 1dB above the theoretical bound for 16QAM."

To validate the spec, you can use a previously saved BERTool session file `rfb_receiver_0.ber`. To find this file, type the following command at the MATLAB prompt

```
which rfb_receiver_0.ber
```

Open the BERTool using the MATLAB command `bertool`. From the File==>Open Session... dialog box, navigate to the saved session `rfb_receiver_0.ber`. Now click on the Monte Carlo tab, and then click on the Run button. A figure like the one below is generated:

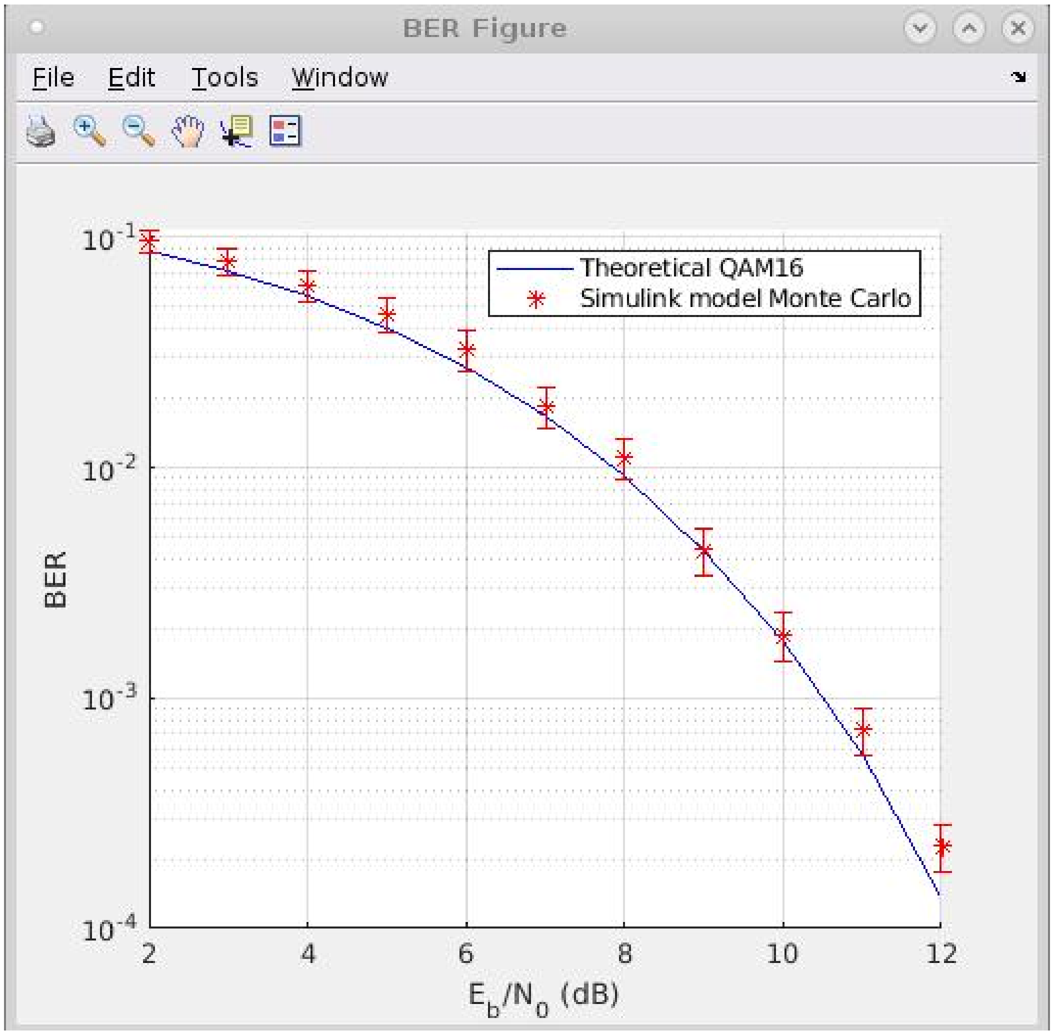
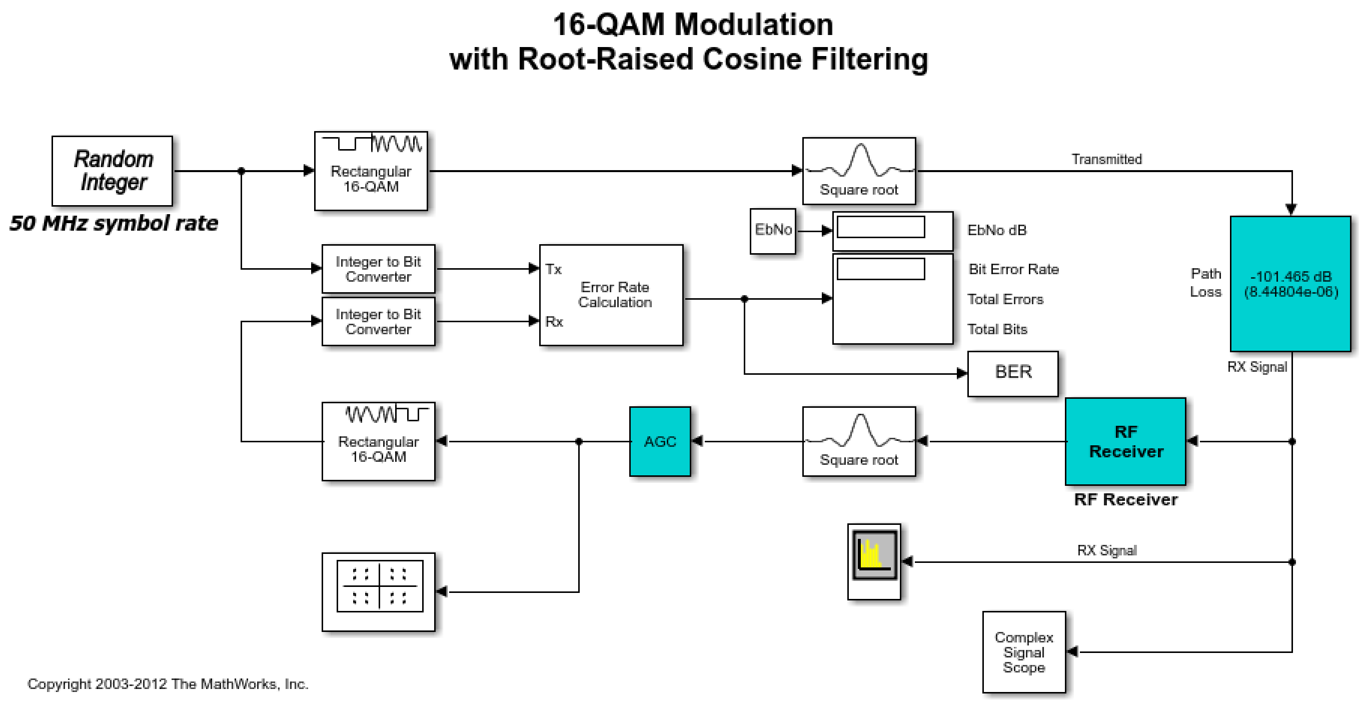


Figure 3: BER versus Eb/No plot without RF impairments

The Eb/No for a given BER value is a little higher than the theoretical bound because of implementation losses. (In the present case, the main loss is due to the finite length of the root raised cosine filters.) But the degradation is within acceptance criteria.

Adding RF Specifications to the Baseline Model

```
open('rfb_receiver_1.slx')
```



Let's elaborate the baseline model and see how it changes with additional refinement using RF Blockset components. The first step is to replace the AWGN block with a path loss block (shown in the preceding figure in cyan); this will lower the signal level close to the end of range value. The path loss (in dB) required to bring the unit power (1W) down to a given Eb/No (also in dB) at the receiver input is:

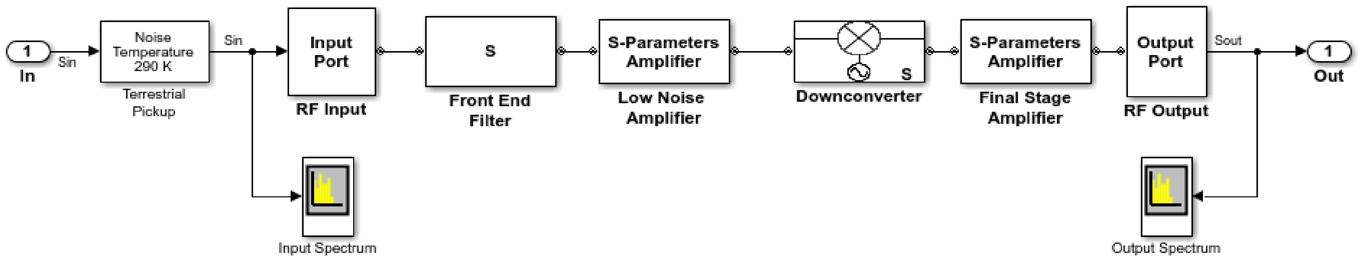
```
path_loss = 10*log10(k*T_ref*B*M) + EbNo + NF
```

where `k` is Boltzmann's constant (~1.38e-23 J/K), `T_ref` is the IEEE® standard noise reference temperature (290K), `B` is the noise bandwidth (~50 MHz in this case), and `NF` is the receiver noise figure in dB.

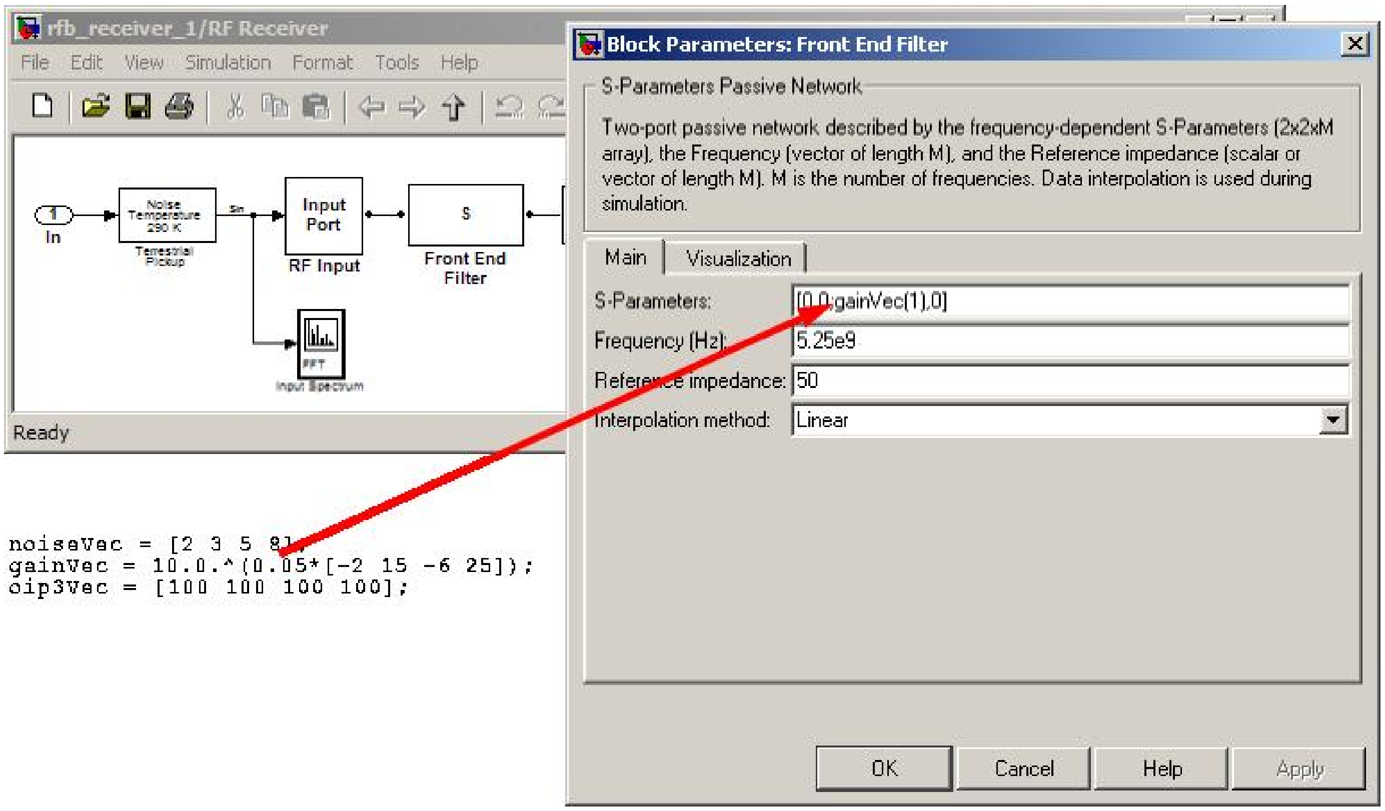
Next, the cyan-colored RF receiver subsystem and AGC Blocks are included. The AGC Block is a consequence of using realistic signal levels required by the demodulator.

The RF Receiver Subsystem Examined

```
open('rfb_receiver_1.slx')
open_system('rfb_receiver_1/RF Receiver')
```



Now examine the RF Receiver subsystem, which is a cascaded model of a super heterodyne receiver. The receiver uses blocks from the RF Blockset Equivalent Baseband library. The Simulink signal enters the RF domain through a gateway "Input Port" block. Notice that the connectors after the gateway are different. The standard Simulink arrows have been replaced with RF connection lines. This is to remind us that RF signals are bidirectional. The receiver is a cascade of components each represented as a 2-port network: a filter, a LNA, a mixer, and an IF strip. The Output Port, in this case, is not only the gateway back to Simulink but also represents an ideal quadrature down conversion mixer. Here is a framework or architecture for a receiver that is not yet designed. An executable specification for the RF engineer has been created. Each stage of the RF subsystem includes a budget for the overall gain, noise and nonlinearities, as shown in the following figure.



S-Parameters Passive Network

Two-port passive network described by the frequency-dependent S-Parameters (2x2xM array), the Frequency (vector of length M), and the Reference impedance (scalar or vector of length M). M is the number of frequencies. Data interpolation is used during simulation.

S-Parameters: [0,0;gainVec(1),0]

Frequency (Hz): 5.25e9

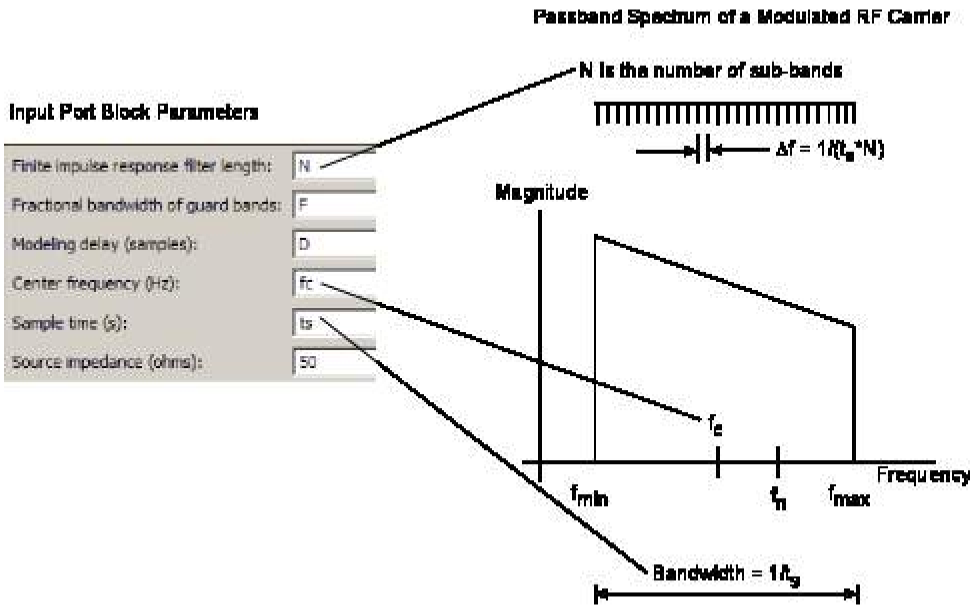
Reference impedance: 50

Interpolation method: Linear

```
noiseVec = [2 3 5 8];
gainVec = 10.0.^(0.05*[-2 15 -6 25]);
oip3Vec = [100 100 100 100];
```

Figure 4: Specification of Amplifier Block Parameters

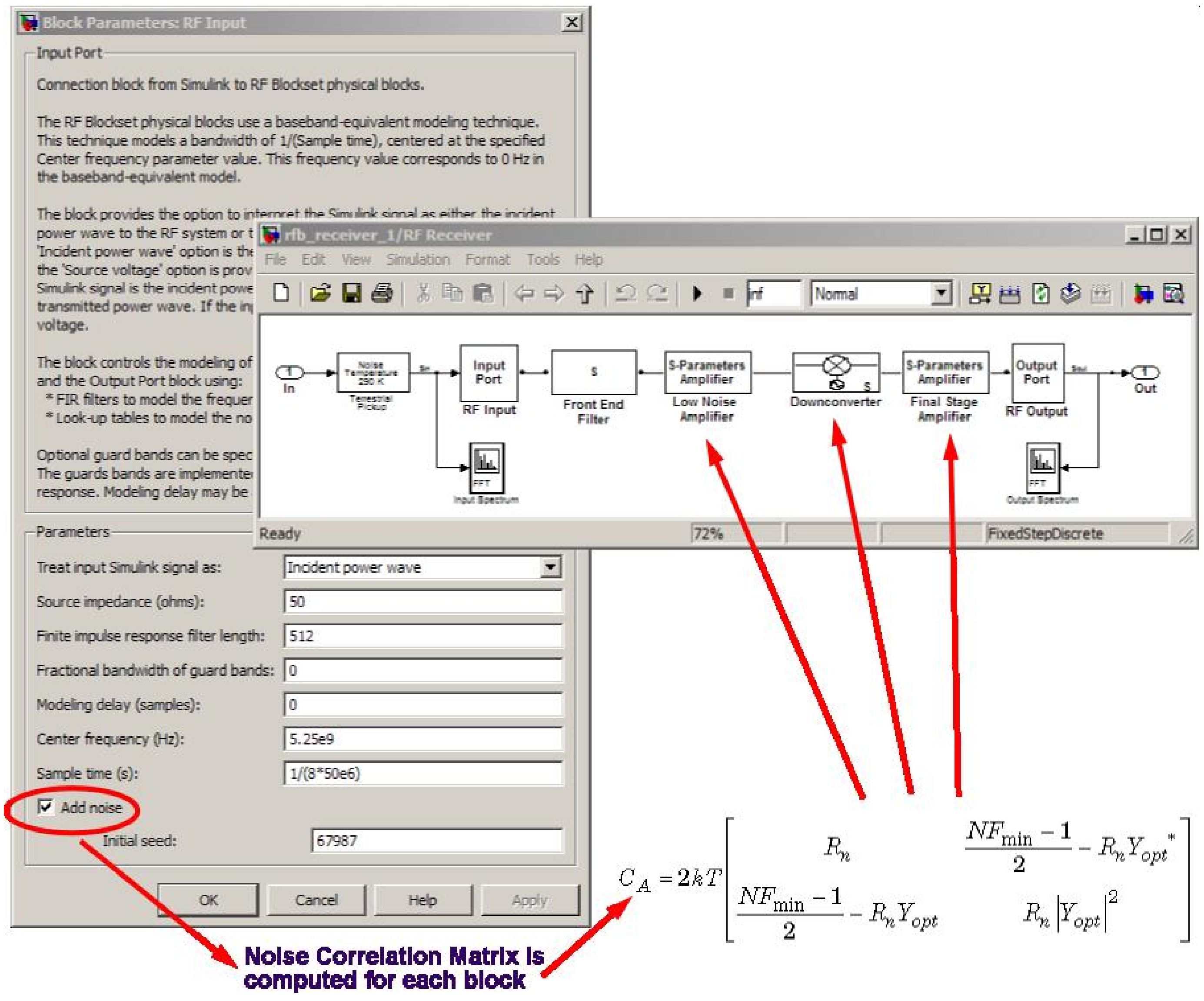
As an example of budgeting, consider the front end filter in the above figure. The S-parameters are specified at a single frequency point using the first element of the gainVec array that was entered into the base workspace using the **PostLoadFcn*** under the Callbacks tab in the Model Properties panel. Each element of the array refers to a stage, so the index 1 refers to the first stage. Values for OIP3, on the Nonlinearity data tab, and for Noise Figure, on the Noise data tab, are similarly specified.


Input Port Block Parameters

Finite impulse response filter length:	N
Fractional bandwidth of guard bands:	F
Modeling delay (samples):	D
Center frequency (Hz):	fc
Sample time (s):	ts
Source impedance (ohms):	50

Figure 5: Specification of Complex Baseband-Equivalent Simulation Parameters

Now open the Input Port block. This port contains parameters that apply to the overall RF subsystem. A narrowband modeling approach is used to capture the in-band effects that impact downstream signal processing blocks. The range of frequencies is specified through the **Center frequency** parameter, the **Sample time** parameter (which is 1/Bandwidth), and the **Finite impulse response filter length** parameter (which is the length of the impulse response filters that are used in modeling RF components). A longer length time-domain filter will give finer frequency-domain resolution within the specified bandwidth. To model mismatch at the input of the first component, source impedance is also specified here. Notice the "Add noise" checkbox. To include noise in the simulation, you must select this "Add noise" checkbox.



Block Parameters: RF Input

Input Port

Connection block from Simulink to RF Blockset physical blocks.

The RF Blockset physical blocks use a baseband-equivalent modeling technique. This technique models a bandwidth of 1/(Sample time), centered at the specified Center frequency parameter value. This frequency value corresponds to 0 Hz in the baseband-equivalent model.

The block provides the option to interpret the Simulink signal as either the incident power wave to the RF system or t 'Incident power wave' option is the the 'Source voltage' option is prov Simulink signal is the incident powe transmitted power wave. If the in voltage.

The block controls the modeling of and the Output Port block using:
* FIR filters to model the frequer
* Look-up tables to model the no

Optional guard bands can be spec The guards bands are implemente response. Modeling delay may be

Parameters

Treat input Simulink signal as: Incident power wave

Source impedance (ohms): 50

Finite impulse response filter length: 512

Fractional bandwidth of guard bands: 0

Modeling delay (samples): 0

Center frequency (Hz): 5.25e9

Sample time (s): 1/(8*50e6)

[✓] Add noise

Initial seed: 67987

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

Figure 6: Noise Modeling with the RF Blockset Equivalent Baseband library

The AWGN block models overall noise as a signal-to-noise ratio. By contrast, blocks from the RF Blockset Equivalent Baseband library model noise by adding the noise contribution of each block individually. For each block, the noise is modeled using an appropriate formulation determined by the set of noise parameters supplied for that block. Once the noise for each block is calculated, the overall system noise model is developed. This overall model includes the position of each block in the cascade (i.e., includes the gain of the subsequent stages).

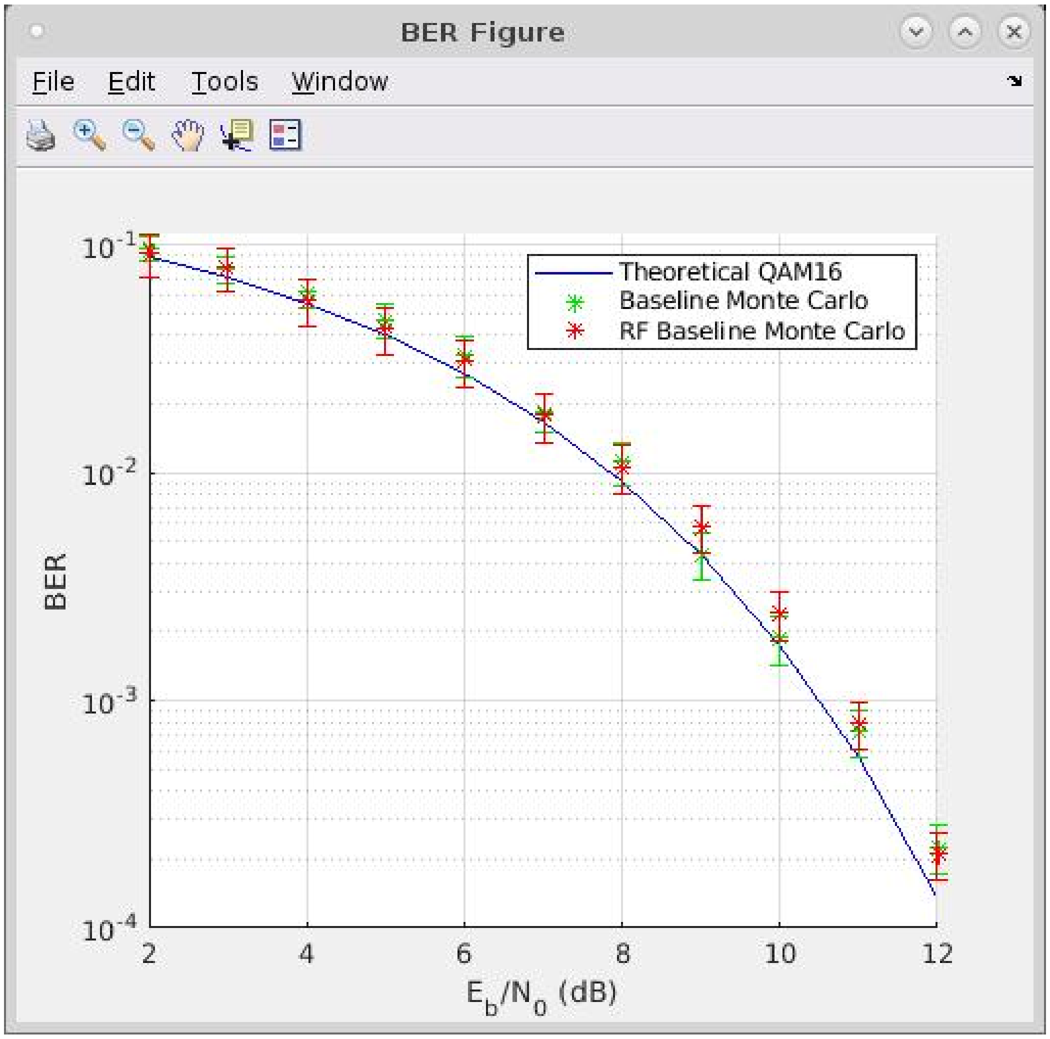


Figure 7: BER versus Eb/No plot with RF impairments

Plots of BER versus Eb/No comparing the theoretical, Baseline and Baseline with RF impairments models are given in Figure 7. This is a simple illustration of the convenience afforded by the Model-Based Design methodology. At this point in the process, an executable specification has been developed. This specification will be used by teams to design their subsystems. In the case of the RF subsystem, the abstract RF blocks will be replaced by discrete components. As each RF block is realized, its effect on the system's design criteria can be assessed.

```
bdclose('rfb_receiver_0');
bdclose('rfb_receiver_1');
```

See Also

S-Parameters Amplifier

Related Topics

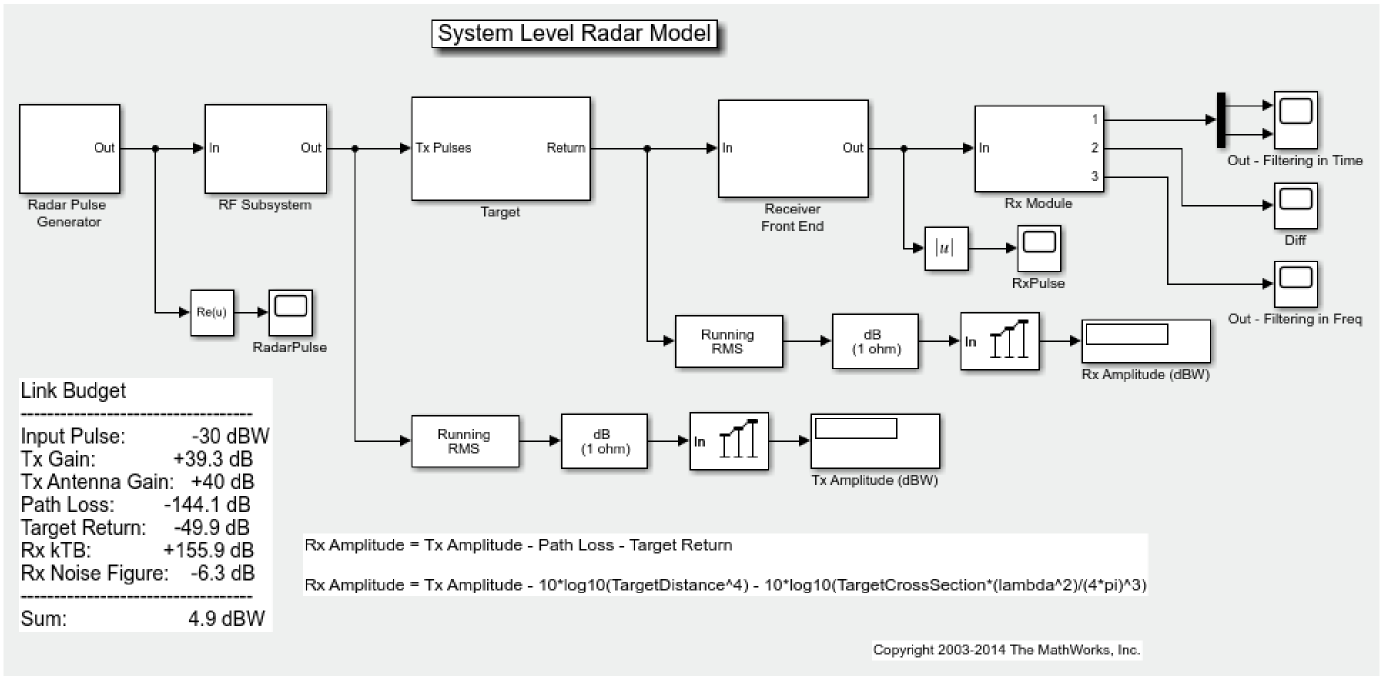
“Effect of Nonlinear Amplifier on 16-QAM Modulation” on page 9-130

Radar Tracking System

This model shows how to simulate a key multi-discipline design problem from the Aerospace Defense industry sector.

Structure of the Example

This example contains subsystems that model the essential features of a radar system. The model is typical of a radar system that is used for target position and velocity detection. The example includes a radar pulse generator, an RF Transmitter subsystem, a Simulink representation of a moving target, an RF Receiver and a Receive Module (Rx Module).



```
Link Budget
-----------------------------------
Input Pulse:          -30 dBW
Tx Gain:            +39.3 dB
Tx Antenna Gain:     +40 dB
Path Loss:         -144.1 dB
Target Return:      -49.9 dB
Rx kTB:            +155.9 dB
Rx Noise Figure:     -6.3 dB
-----------------------------------
Sum:                 4.9 dBW
```

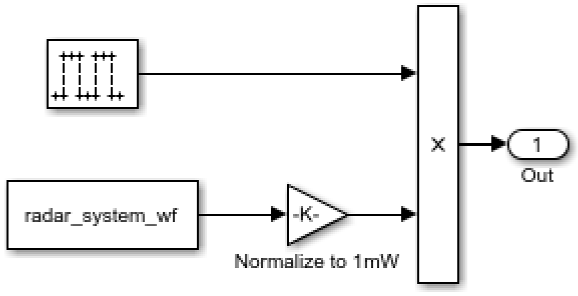
Rx Amplitude = Tx Amplitude - Path Loss - Target Return

Rx Amplitude = Tx Amplitude - 10*log10(TargetDistance^4) - 10*log10(TargetCrossSection*(lambda^2)/(4*pi)^3)

Copyright 2003-2014 The MathWorks, Inc.

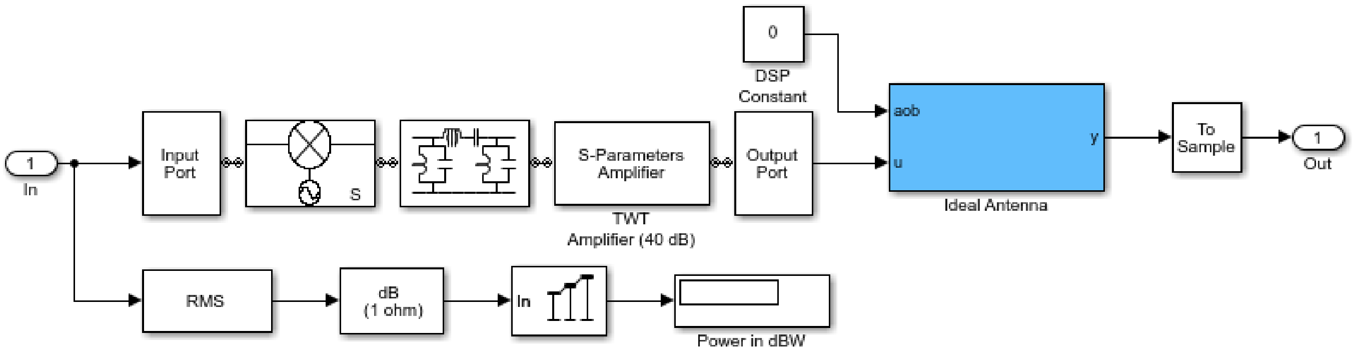
Radar Pulse Generator

The radar Pulse Generator creates a swept frequency signal (chirp signal) that has a 10 percent duty cycle. The subsystem is implemented by using Simulink® blocks and a signal from the MATLAB workspace that represents a chirp signal.



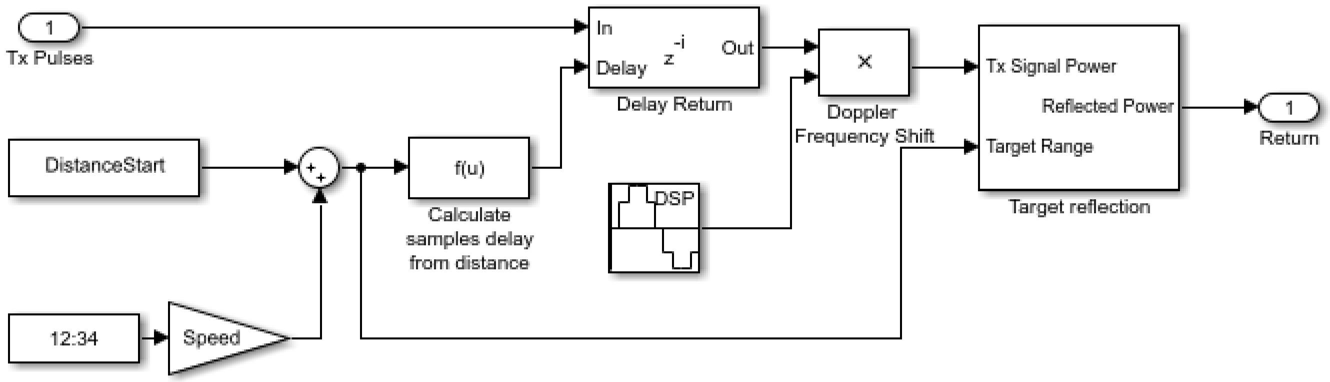
RF Transmitter Subsystem

This Subsystem is implemented with both core Simulink blocks as well as blocks from the RF Blockset Equivalent Baseband library. The RF Blockset subsystem represents a traveling wave tube amplifier. An ideal antenna is implemented via a Simulink gain block. Within the subsystem, there are DSP System Toolbox blocks used for calculating the power level of the baseband signal.



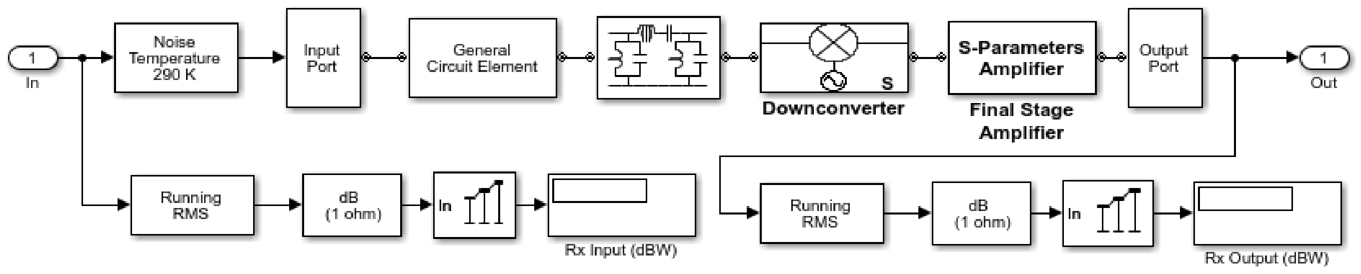
Target

The Target is implemented using theoretical implementations of a moving target that fully reflects all of the incident signal off of its cross-sectional surface, which is perpendicular to the direction of travel of the incident radar pulses.



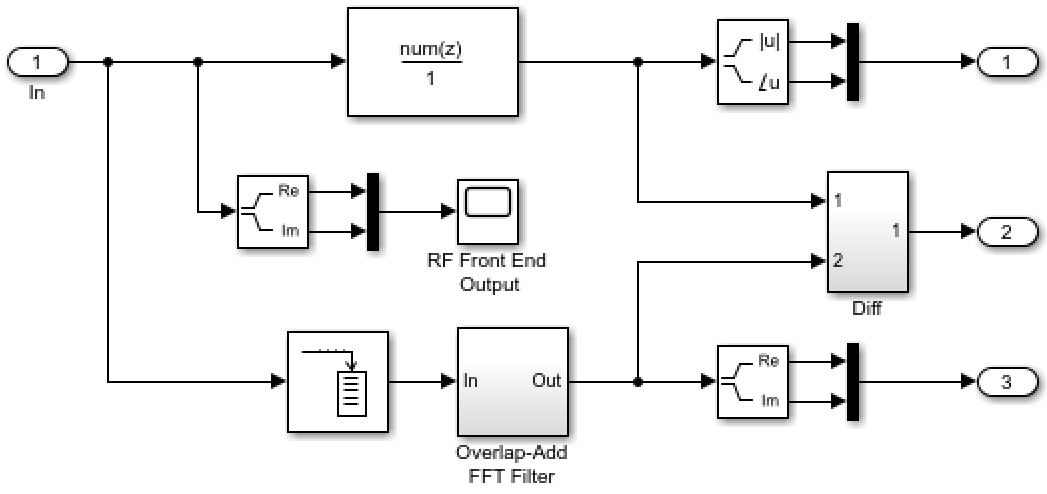
RF Receiver

The RF Receiver is implemented using the RF Blockset Equivalent Baseband library. The RF receiver is a super heterodyne receiver. The LNA is a matched amplifier. For broadband impedance matching, see the example of RF Toolbox: “Designing Broadband Matching Networks (Part 2: Amplifier)”. The LNA is represented by a Touchstone® data file with noise data. The name of the datafile is samplebjt.s2p. Following the amplifier are behavioral models for a bandpass filter, mixer and a high gain, high noise amplifier.



Rx Module

The Rx Module in this example serves two purposes. First, the module contains a matched filter detector for target detection. Also, this module serves as a testbench where a theoretical filter implementation is realized via Simulink blocks, the output of each of these filters is compared, and the difference is plotted.



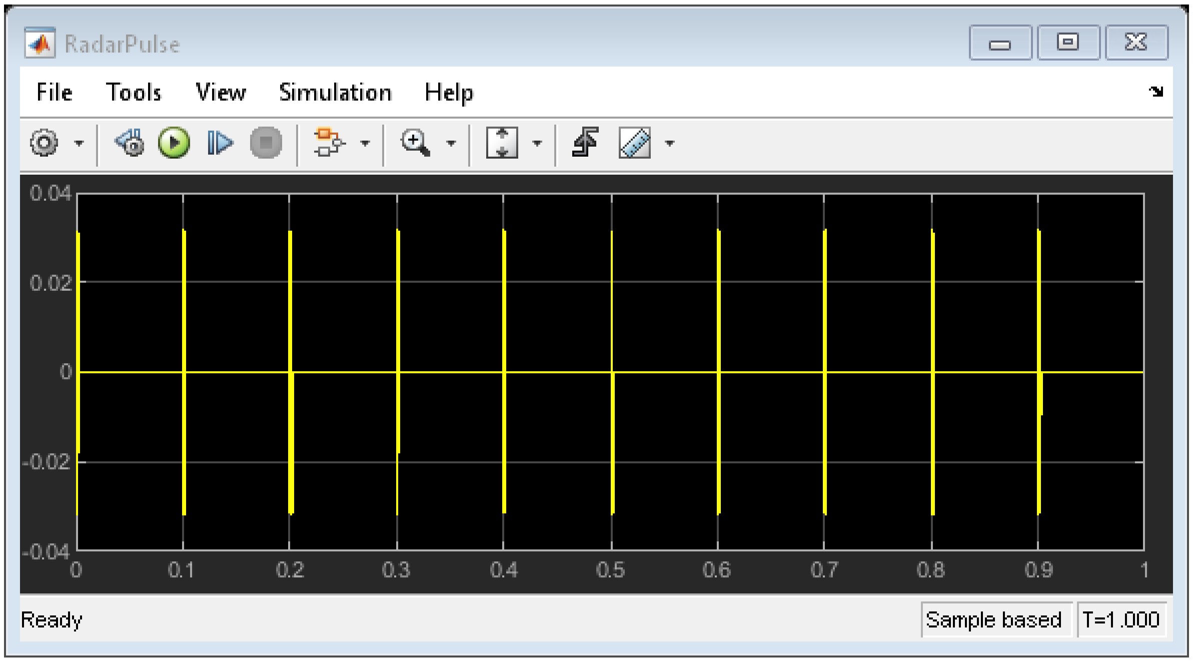
Exploring the Example

You can set the target cross section, target speed, and relative distance to the target by double-clicking the Target icon and specifying the corresponding parameters. At sufficiently large distances, the return signal cannot be detected within the noise. Similarly, the return signal cannot be detected in the noise if the target cross section is too small.

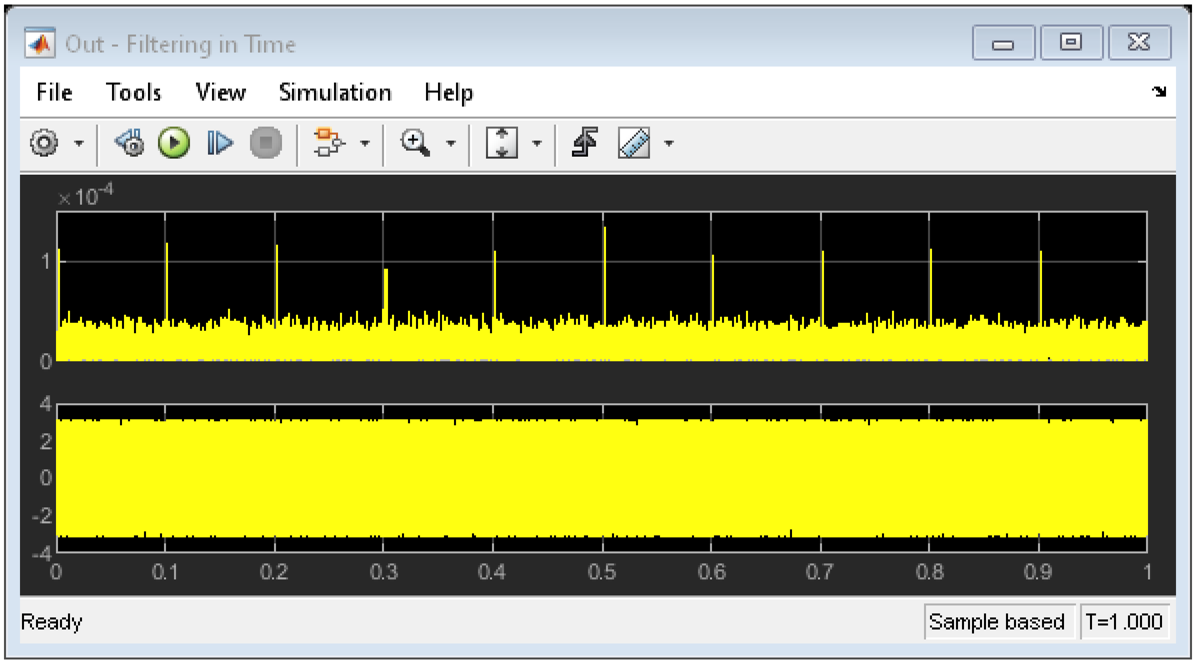
Results and Displays

After simulation, open four time-domain signal graphs of interest for examination.

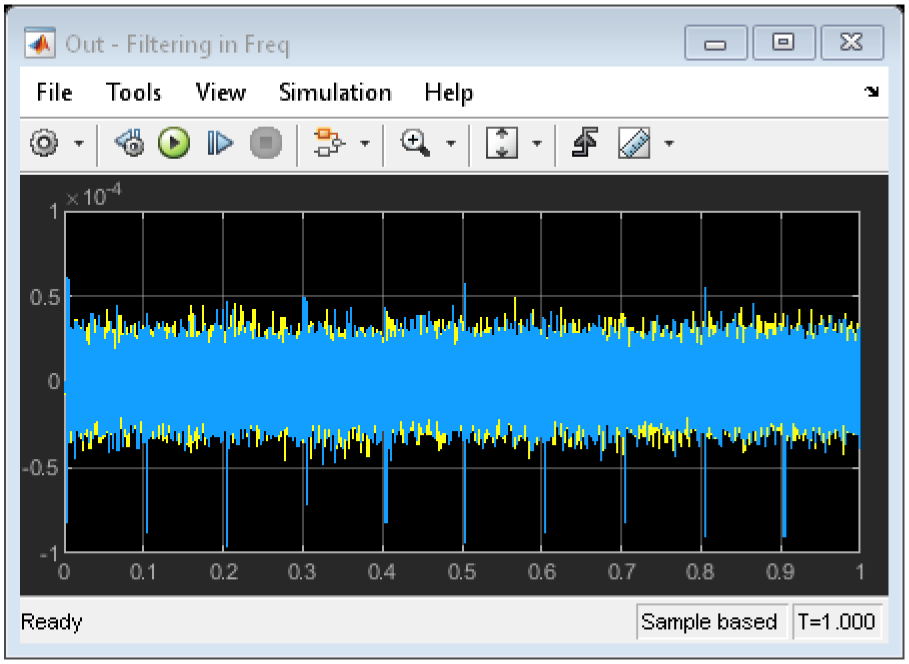
The first graph, "RadarPulse", displays the time-domain representation of a chirp signal with a 10% duty cycle.



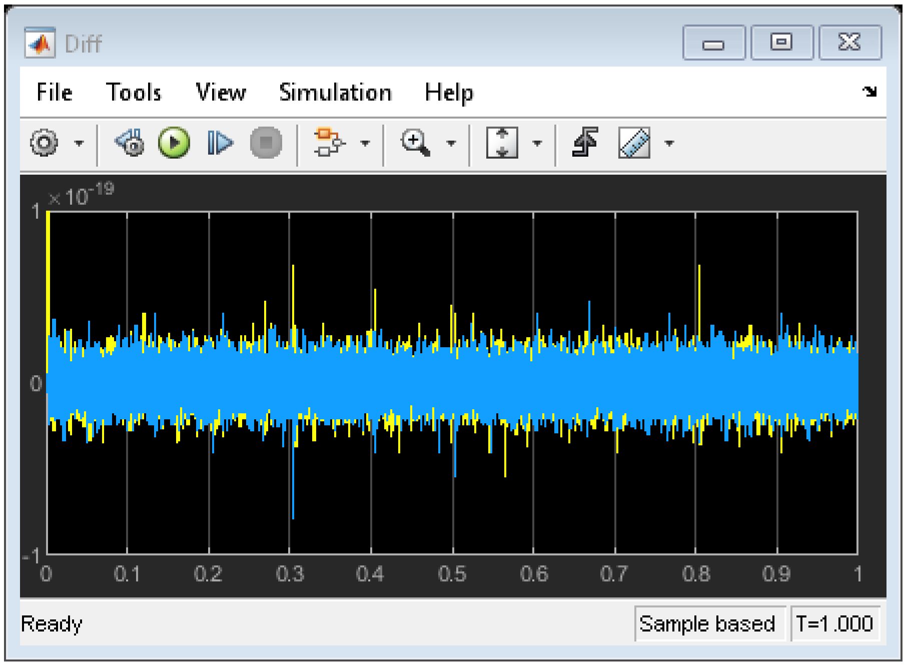
The second graph, "Out - Filtering in Time", displays the magnitude and phase of the filtered return signal with noise.



The third graph, "Out - Filtering in Frequency", displays the real and imaginary response of the filtered return signal with noise through an ideal filter implementation.



The fourth graph, "Diff", displays the difference between the results obtained in calculating the results for graphs two and three.



The following blocks display numerical results:

The Tx Amplitude (dBW) block displays the power transmitted in dBW.

The Rx Amplitude (dBW) block displays the target return power in dBW.

See Also

“Radar System Modeling” on page 9-89 | “Executable Specification for System Design” on page 9-133 | S-Parameters Amplifier | General Mixer

User-Defined Nonlinear Amplifier Model

This example shows how to:

- Generate user-defined custom models by creating an RF Toolbox™ object in the MATLAB® workspace and importing it into an Equivalent Baseband amplifier block.
- Create a nonlinear amplifier with an adjustable Pin-Pout curve.
- Simulate intermodulation products in a two-tone test model.

In the example, the Vin-Vout relationship for the amplifier is a simple polynomial. The example uses this voltage relationship to generate a Pin-Pout curve that is incorporated into the amplifier model. Although the example uses a specific polynomial function, the same approach can be used to create arbitrary functions of power out and phase versus power in and frequency. Indeed, it can be used to set any of the writable properties of a component, such as S-parameters or noise properties.

Define the Voltage-Out Versus Voltage-In Relationship

Use MATLAB commands to create a vector of polynomial coefficients that define the desired Vin-Vout relationship. A MATLAB convention is to store nth order polynomial coefficients in a row vector of length n+1 with the nth power in the first element and the zeroth power (constant) in the last (n+1 th) element. The power series here is for illustration, and can easily be altered. You may recognize the values as the first twelve terms of the polynomial series expansion of the hyperbolic tangent function.

```
TanhSeries = [-1382/155925 0 62/2835 0 -17/315 0 2/15 0 -1/3 0 1 0];
```

Next choose a range for the independent variable, Vin, and define a vector of input voltage values over this range. The low end (1 mV here) should be sufficiently low that the linear term dominates Vout. The high end should be chosen such that Vout just reaches its local maximum value. The resulting Vout will be extrapolated for all Vin greater than 1.23. Next compute the vector of output voltage values, Vout, using the power series.

```
Vin = linspace(0.001,1.23,100);     % volts
Vout = polyval(TanhSeries, Vin);    % volts
```

Create a Nonlinear Amplifier and Generate the Pin-Pout Curve

Create an RF Toolbox amplifier object with the default property values. Then, generate the Pin-Pout data (in watts) for the amplifier by dividing the square of the input and output voltage vectors by the reference impedance of the amplifier. Use the Pin-Pout data to specify the nonlinearity of the amplifier object. For this example, the Pin-Pout curve is defined for one frequency point (2.1 GHz) and used (by extrapolation) at all frequency points. See the RF Toolbox documentation for information on how to create a component with separately defined curves for any number of frequency points.

```
amp = rfckt.amplifier;
amp.NonlinearData.Freq = 2.1e9;            % Hz
Zref = 50;                                 % ohm
amp.NonlinearData.Pin  = {(Vin.^2)./Zref};  % watts
amp.NonlinearData.Pout = {(Vout.^2)./Zref}; % watts
```

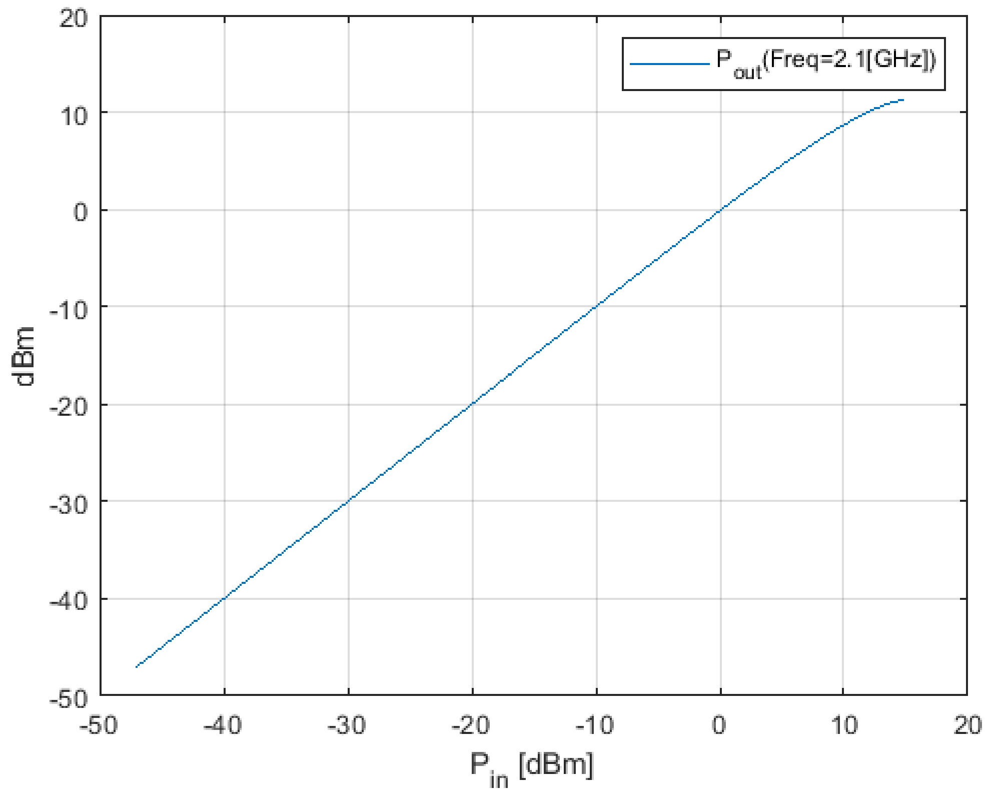
In this example, we define the phase change to be zero for all Pin and all frequencies, but RF Toolbox lets you set it to be a function of Pin and frequency.

```
amp.NonlinearData.Phase = {zeros(size(Vin))};
```

Make the Small-Signal Gain Consistent with the Power Gain Slope at Low Power

Define the S21 parameter of the amplifier at the frequency point for which you specified the power data. The S21 network parameter must be consistent with the gain slope at the low power end of the Pin-Pout curve at that frequency point. If these values are inconsistent, RF Blockset software will attempt to reconcile the data and issue a warning that is has done so. To ensure consistency, define the S21 parameter to be the linear term of the power series that defines the Vin-Vout relationship. The linear term is the next-to-last element in the vector. Plot the Pin-Pout curve.

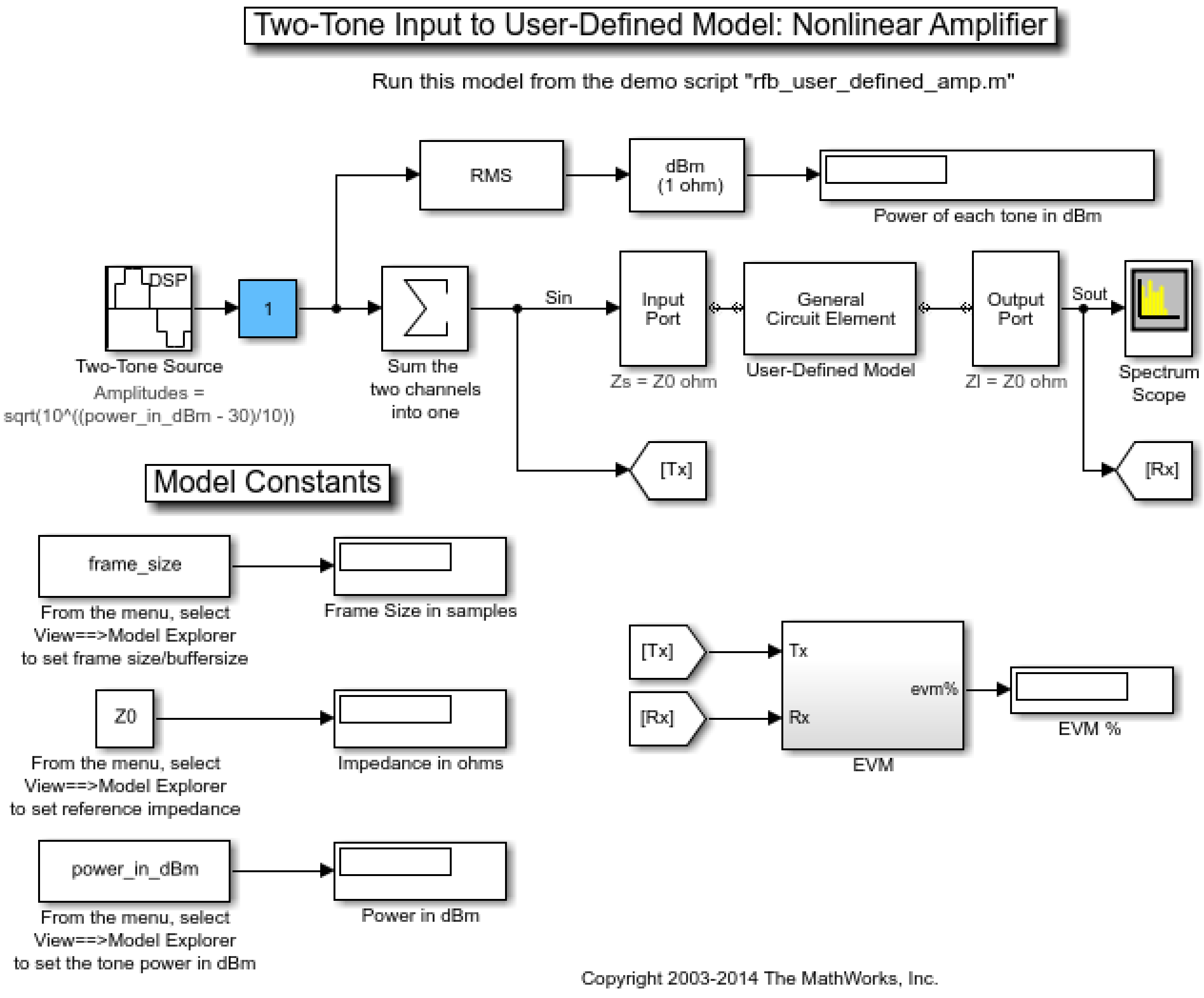
```
amp.NetworkData.Freq = amp.NonlinearData.Freq;
amp.NetworkData.Data = [0 0; TanhSeries(end-1) 0];
fig = figure;
plot(amp,'Pout');
```



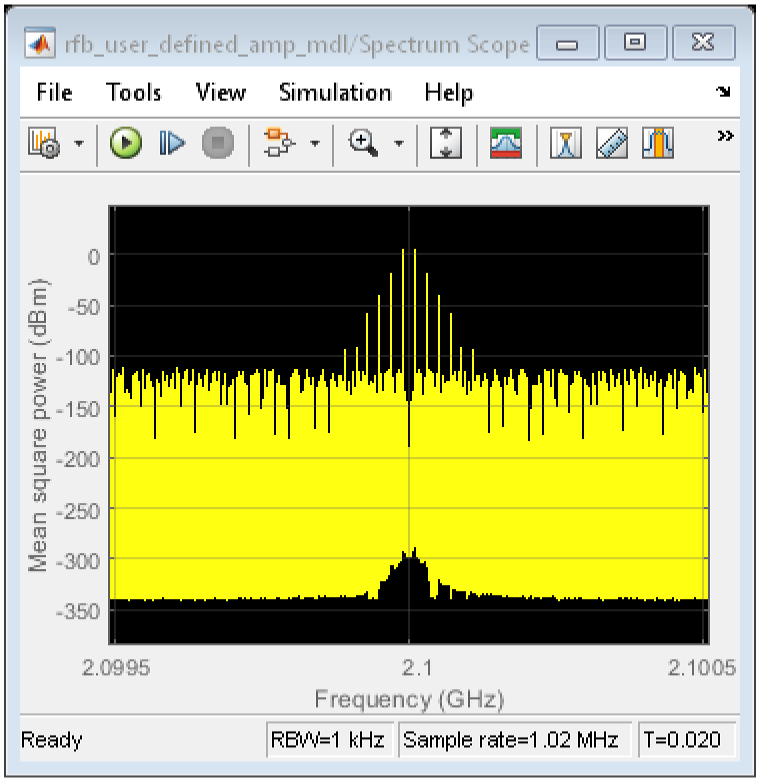
Run the Test Harness with an Input Power of 7 dBm Per Tone

The following figure shows the test harness for your new amplifier. The input signal consists of the sum of two tones, one 10kHz below the center frequency, and one 10kHz above it. The spectrum scope shows the various intermodulation products at higher and lower frequencies than the two test tones. The EVM subsystem calculates error vector magnitude. Open and Run "rfb_user_defined_amp_mdl" model. To view and edit the preset model workspace values, click the **Modeling** Tab in the Toolstrip and select **Model Explorer** .

```
%

open('rfb_user_defined_amp_mdl.slx');
```


Two-Tone Input to User-Defined Model: Nonlinear Amplifier
Run this model from the demo script "rfb_user_defined_amp.m"
RMS
dBm (1 ohm)
Power of each tone in dBm
DSP
Two-Tone Source
Amplitudes = sqrt(10^((power_in_dBm - 30)/10))
1
Sum the two channels into one
Sin
Input Port
Zs = Z0 ohm
General Circuit Element
User-Defined Model
Output Port
Zl = Z0 ohm
Sout
Spectrum Scope
[Tx]
[Rx]
Model Constants
frame_size
From the menu, select View==>Model Explorer to set frame size/buffersize
Frame Size in samples
Z0
From the menu, select View==>Model Explorer to set reference impedance
Impedance in ohms
power_in_dBm
From the menu, select View==>Model Explorer to set the tone power in dBm
Power in dBm
[Tx]
[Rx]
Tx
Rx
evm%
EVM
EVM %
Copyright 2003-2014 The MathWorks, Inc.

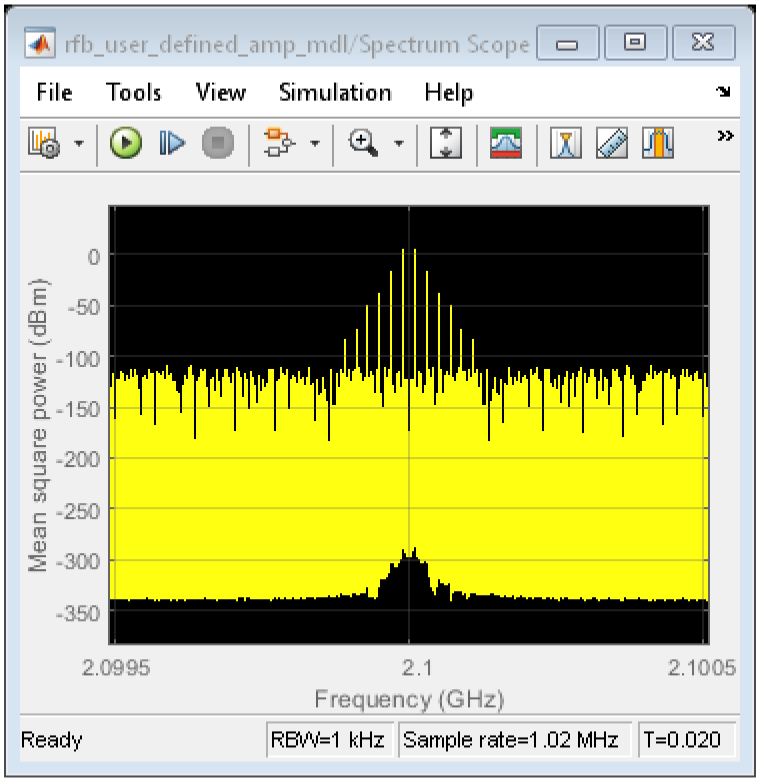
```
sim('rfb_user_defined_amp_mdl');
```



Rerun the Simulation Using an Input Power Level of 8 dBm Per Tone

To increase the source power from 7dBm to 8 dBm from a MATLAB script, first get the handle to the current Simulink® model workspace. Then set the appropriate workspace variable (power_in_dBm in this example) to the value 8. Rerun the simulation. Notice that 11th order intermods (outermost tones on the spectrum scope) increase by about 11 dB.

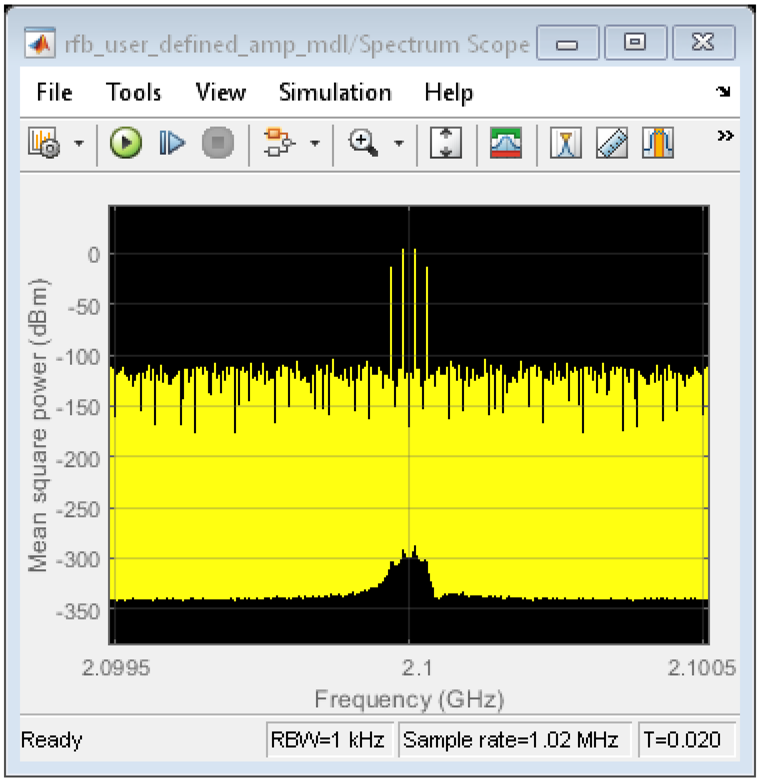
```
hws=get_param(bdroot, 'modelworkspace');
hws.assignin('power_in_dBm', 8);
sim('rfb_user_defined_amp_mdl');
```



Truncate the Hyperbolic Tangent Series to Only the First Four Terms

Reset the input power to 7 dBm. Then, specify the power series for the hyperbolic tangent using only the first four terms of the series. Recalculate the output voltage values and the amplifier Pin-Pout data. Rerun the simulation. The spectrum scope shows that only the third intermodulation products are produced. You can also set the stop time to inf, rerun the simulation, and experiment to see the effect of the slider gain (Blue box on the Simulink model).

```
hws.assignin('power_in_dBm', 7);
TanhSeries = [-1/3 0 1 0];
Vin = linspace(0.001,1,100);                % volts
Vout = polyval(TanhSeries, Vin);            % volts
amp.NonlinearData.Pin  = {(Vin.^2)./Zref};  % watts
amp.NonlinearData.Pout = {(Vout.^2)./Zref}; % watts
sim('rfb_user_defined_amp_mdl');
```



```
bdclose('rfb_user_defined_amp_mdl');
close(fig);
```

See Also

Output Port | Input Port | Configuration | `rfckt.amplifier`

Related Topics

“Effect of Nonlinear Amplifier on 16-QAM Modulation” on page 9-130

Modeling and Simulation of MIMO RF Receiver Including Beamforming

This example shows how to model a MIMO RF receiver with a baseband beamforming algorithm. It considers antenna coupling effects and RF imperfections. The simulation of the system-level model includes the RF receiver baseband beamforming algorithms, RF imperfections, and the antenna array radiation pattern.

In the following sections, you will see more details about the transmitter, receiver, and beamforming algorithm.

Transmitter and Channel

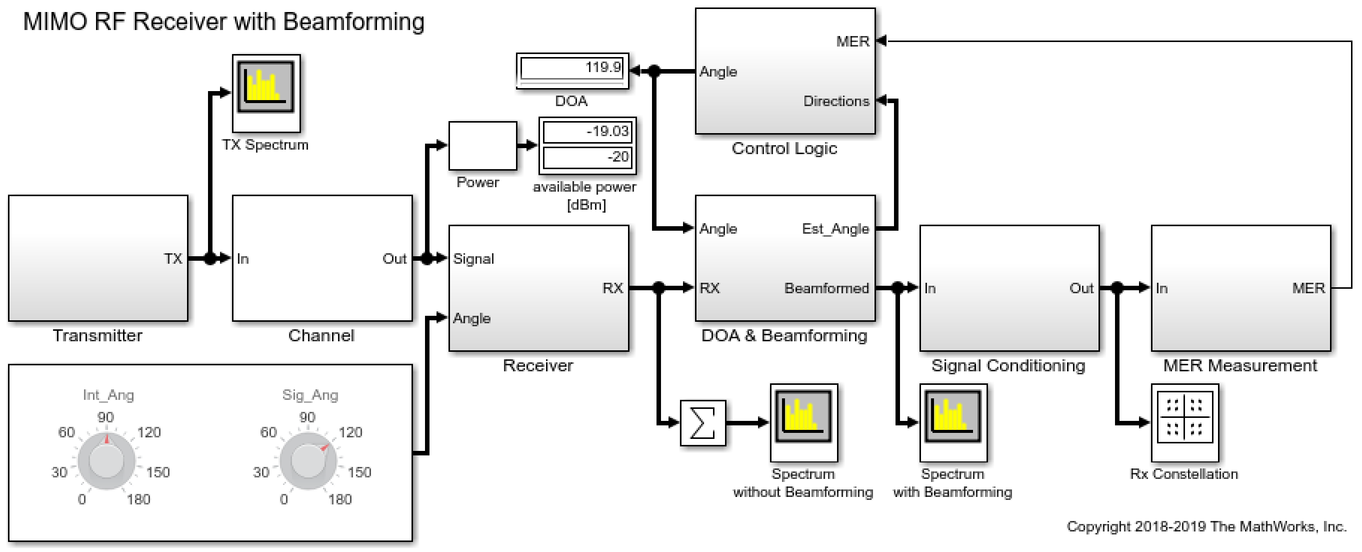
The transmitter and channel models are ideal.

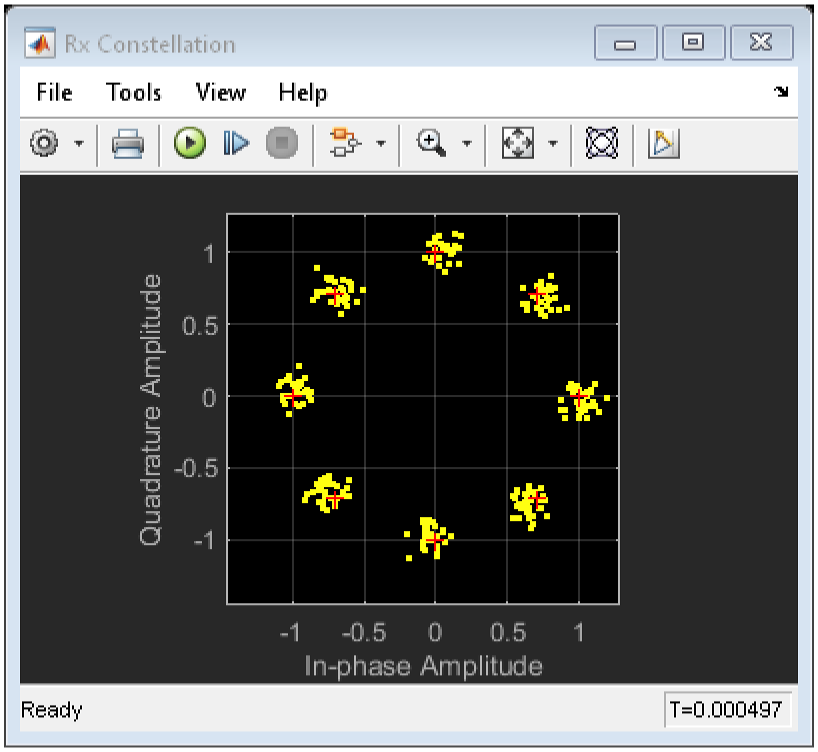
- The transmitter constructs a simple modulated signal transmitted using a single antenna.
- The channel model introduces path loss attenuation and adds an interfering narrowband signal with power level similar to the desired signal.

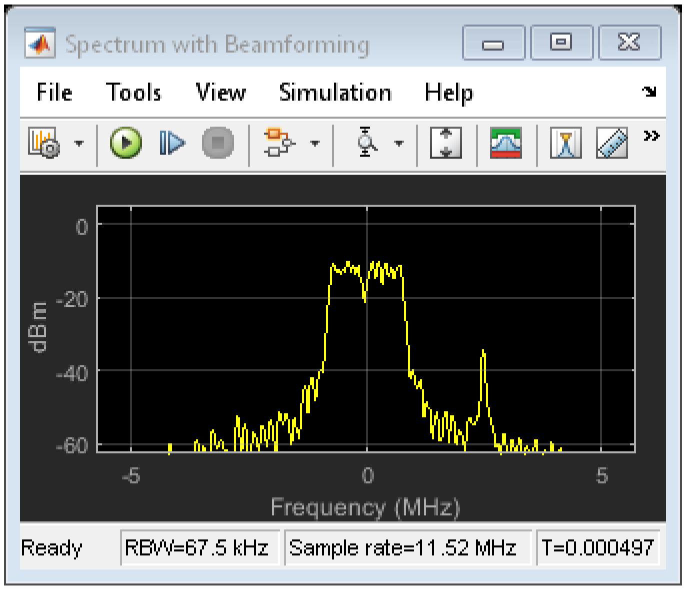
The model assumes that transmitter and receiver are positioned on the same plane. You can change the angle of arrival of the desired transmitted signal and of the interfering signal by turning the dials on the Simulink diagram.

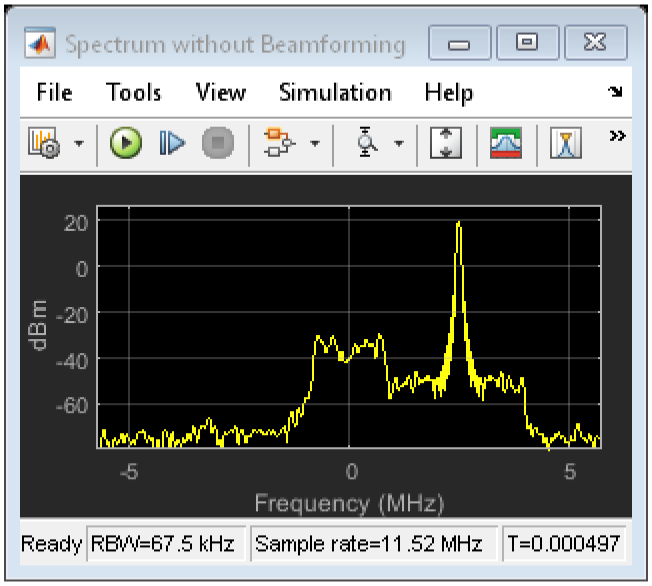
- An angle of 90 degrees indicates that the transmitter is in front of the receiver, where the main lobe of the antenna array radiation pattern is located.
- An angle of 120 degrees indicates that the transmitter is 30 degrees away from the normal axis to the array, where a null of the radiation pattern is positioned.

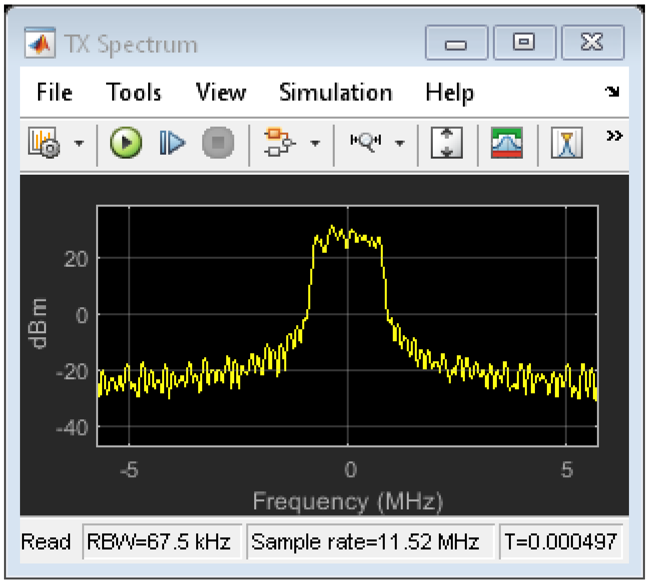
Changing the relative angle of arrival for desired and interfering signals will change the relative signal powers in the Spectrum Analyzer scope "Spectrum without Beamforming". In this case, all 8 received signals are just summed together, without applying any beamforming algorithm.




Rx Constellation
File
Tools
View
Help
Quadrature Amplitude
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
In-phase Amplitude
Ready
T=0.000497


Spectrum with Beamforming
File
Tools
View
Simulation
Help
dBm
0
-20
-40
-60
-5
0
5
Frequency (MHz)
Ready
RBW=67.5 kHz
Sample rate=11.52 MHz
T=0.000497



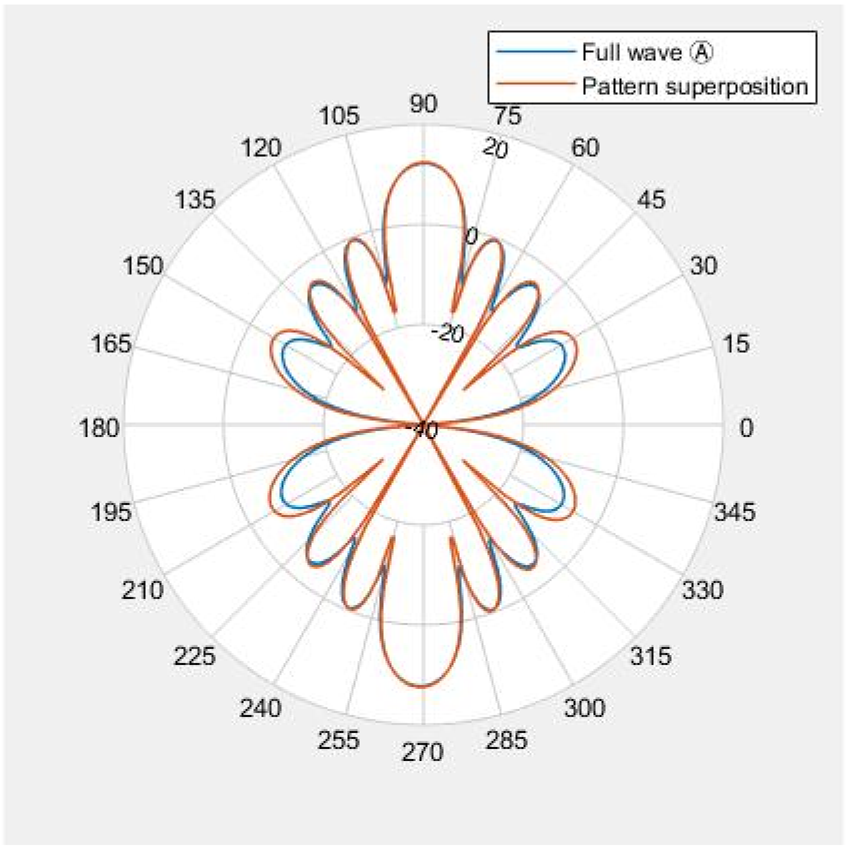


Design the Receiver Antenna Array

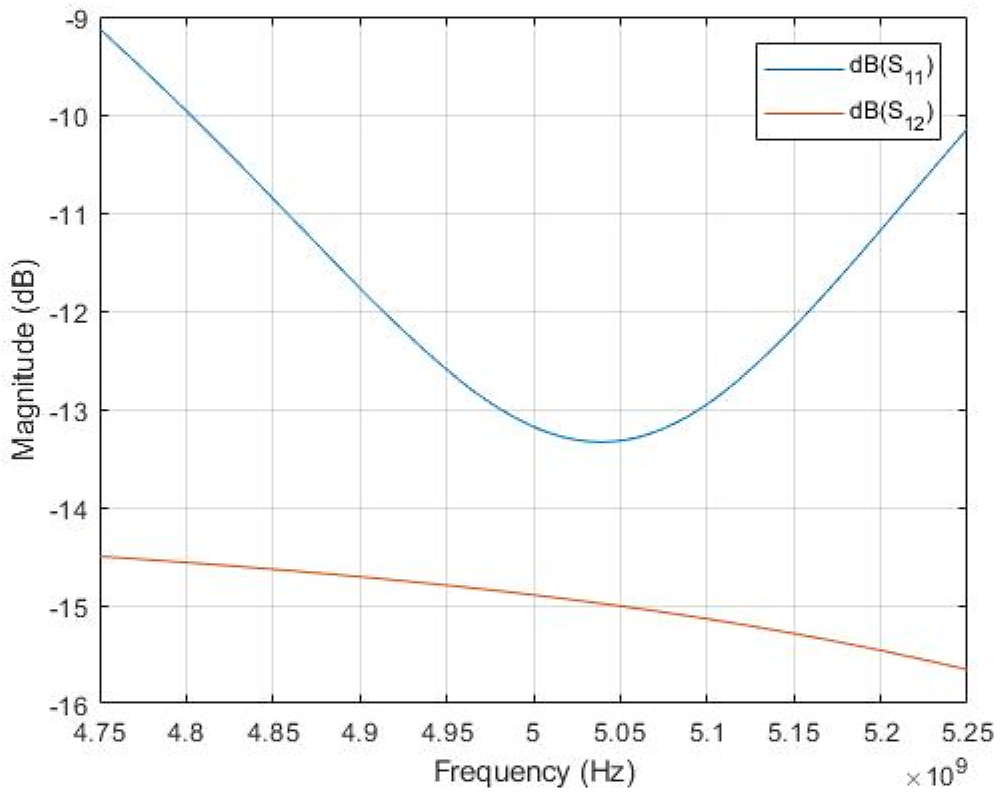
The receiver antenna array is designed using Antenna Toolbox. The Antenna Toolbox helps you design an antenna at the desired operating frequency and verify that the pattern superposition of the isolated element is an acceptable approximation for the array simulation.

Script to design and verify the antenna array

As you can see, the antenna array consists of 8 dipole antennas resonating at 5 GHz. The comparison of the far-field radiation pattern of the array computed with full-wave analysis and pattern superposition of the isolated element shows modest differences:



However, the S-parameters show a non-negligible leakage between adjacent antennas.



RF Receiver

The receiver model includes:

- Model of receiver antenna array. The receiver antenna array is composed using 8 dipole antennas operating at 5 GHz. The array radiation pattern is modeled with Phased Array System Toolbox "Narrowband Rx array". The array is simulated using pattern superposition of the isolated element stored in the variable P_antenna computed using Antenna Toolbox and the script. You can visualize the radiation pattern by clicking the **Analyze** button in the sensor array tab.
- Model of RF receiver. The RF receiver is composed with eight non-linear superheterodyne receivers and filters described with S-parameters. Each chain is designed with the RF Toolbox "RF Budget Analyzer app" as described in: RF Receiver Design example.
- Antenna array impedance is described with the eight-port S-parameters computed using Antenna Toolbox. The S-parameters capture the loading of the antenna array on the RF receiver as well as the coupling between the antenna elements. A lumped inductance for each receiver is used to retune the respective antenna.
- Eight 12-bit ADCs capturing the finite dynamic range of the data converters by modeling saturation and quantization.

DOA & Beamforming

The baseband receiver algorithm consists of four main elements in a closed feedback loop.

- Root MUSIC algorithm to determine the Direction of Arrival assuming that 2 signals are present. The two estimated DOA angles are passed to a state machine that determines which angle produces the higher Modulation Error Ratio (MER). This state machine includes some time delay in between state transitions to avoid decision jitter.
- MVDR Beamforming algorithm for the receiver to focus on the desired signal and suppress interference and noise from other directions. It uses the angle chosen by the Control Logic to maximize the MER.
- Signal Conditioning and estimation of the Modulation Error Ratio. The MER is used to determine which angle to select for the beamforming algorithm.

Related Topics

"Modeling an RF mmWave Transmitter with Hybrid Beamforming" on page 9-160

Modeling an RF mmWave Transmitter with Hybrid Beamforming

This example illustrates a methodology for system-level modeling and simulation of a 66 GHz QPSK RF transmit and receive system with a 32-element hybrid beamforming antenna. The system includes RF imperfections, transmit array radiation effects, a narrowband receive array and a baseband receiver with corrections for system impairments and message decoding. The antenna beamforming direction is defined using azimuth and elevation angles and it is estimated in the RF receive antenna using a Root Music DOA algorithm.

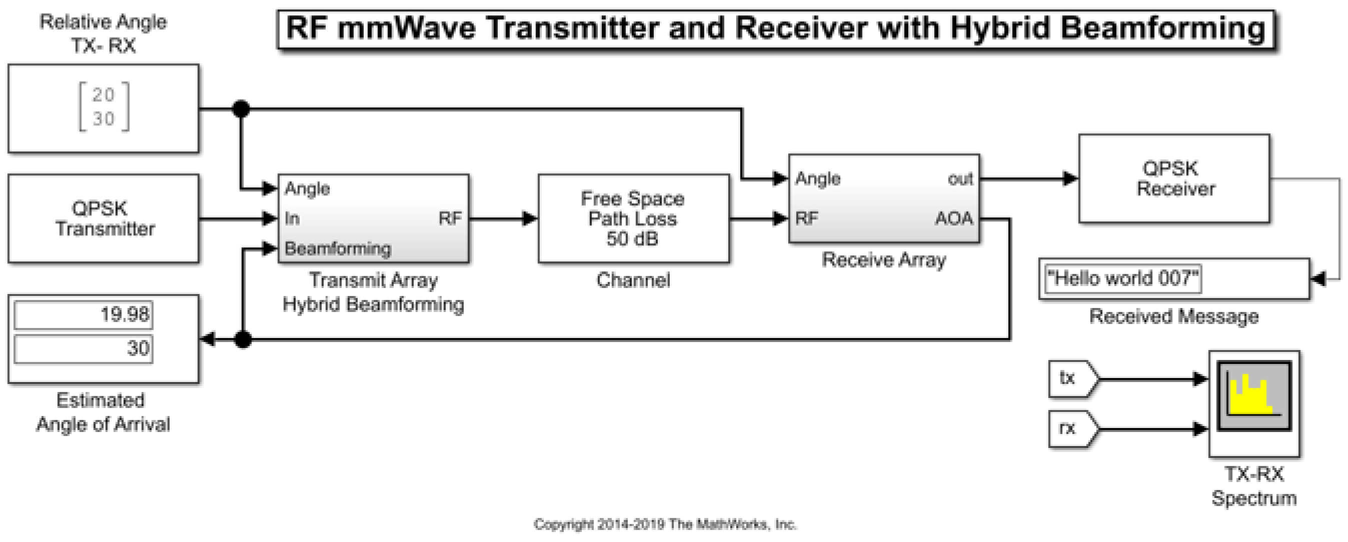
In the following sections you will see more details about the system design.

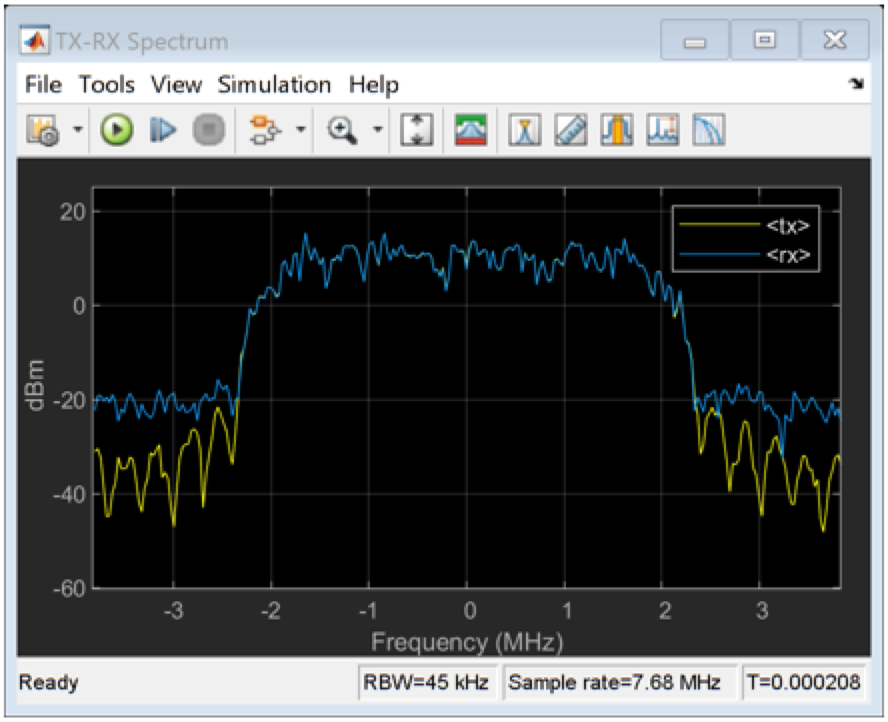
Model Description

The top-level of this example consists of five sub-system blocks, a block to control the relative angle between transmitter and receiver, and 2 displays:

- A QPSK baseband transmitter encodes the message "Hello World ###".
- An RF transmitter with IQ modulation, mixing, amplification and hybrid beamforming with control circuitry. The RF transmitter model includes RF imperfections such as noise, non-linear effects and antenna element coupling.
- An ideal channel attenuating the transmitted signal with a free space path loss model.
- An RF receiver with two narrowband receive array antennas, receiver gain and SNR, 12-bit ADC with finite dynamic range, and two root MUSIC algorithms for angle of arrival estimation along azimuth and elevation.
- A QPSK receiver, including carrier and frame synchronization, demodulation and data decoding.
- A block where the user sets the relative angle between the transmitter and the receiver.
- A spectrum analyzer scope comparing normalized transmitted and received signals and a display for the received message.

```
model = 'simrfV2_qpsk';
open_system(model)
sim(model)
```

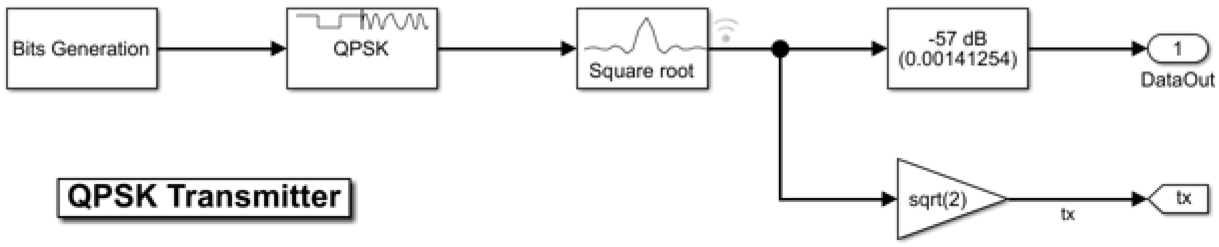

RF mmWave Transmitter and Receiver with Hybrid Beamforming
Relative Angle TX- RX
20
30
QPSK Transmitter
Angle
In
RF
Beamforming
Transmit Array Hybrid Beamforming
Free Space Path Loss 50 dB
Channel
Angle
out
RF
AOA
Receive Array
QPSK Receiver
"Hello world 007"
Received Message
19.98
30
Estimated Angle of Arrival
tx
rx
TX-RX Spectrum
Copyright 2014-2019 The MathWorks, Inc.



QPSK Transmitter

The QPSK transmitter includes a Bit Generation subsystem, a QPSK Modulator block, a Raised Cosine Transmit Filter block for pulse shaping, and a Gain block. The Bit Generation subsystem generates frames. Each frame contains 26 header bits followed by a payload of 174 bits, 105 bits for the message 'Hello world ###' and 69 random bits. The payload is scrambled to guarantee a balanced distribution of zeros and ones for the timing recovery operation in the receiver model.

```
open_system([model '/QPSK TX'],'force')
```

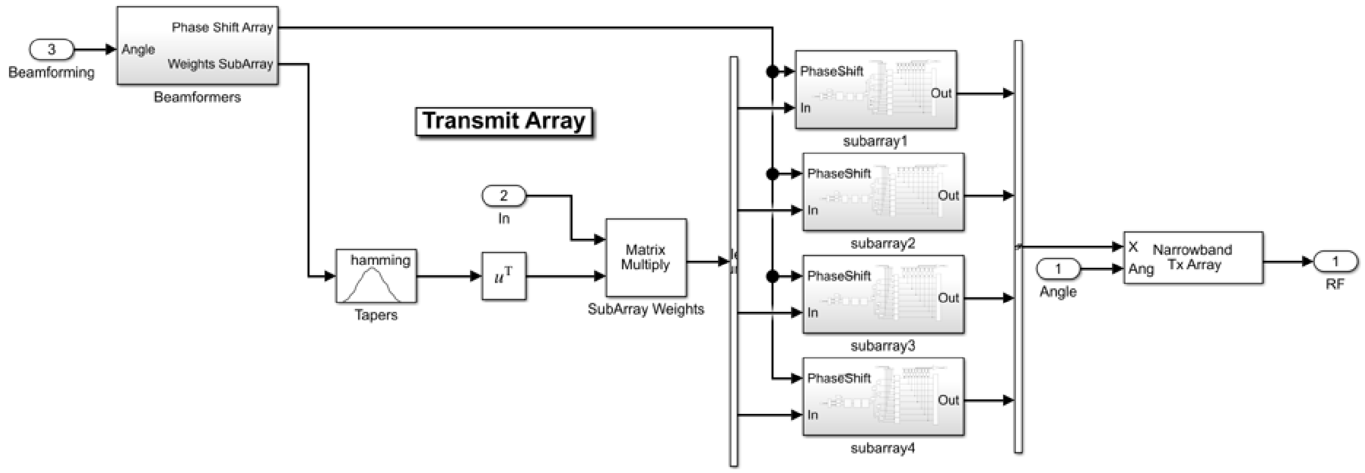


RF Transmitter

The RF transmitter is composed of three sections: array beamformers, a hybrid beamforming antenna and a Narrowband Transmit Array block. The 32-element hybrid beamforming antenna is divided in 4 sub-arrays. Each subarray consists of 8 RF transmitters operating at 66 GHz. The antennas are microstrip patches. These antenna elements and the subarrays have been designed and verified with a MATLAB script that uses Antenna Toolbox™.

The far field antenna array gain is computed with the Phased Array System Toolbox™ Narrowband Transmit Array block. The computed radiation pattern is the superposition of the fields generated by the isolated microstrip patches.

```
open_system([model '/Transmit Array Hybrid Beamforming'])
```



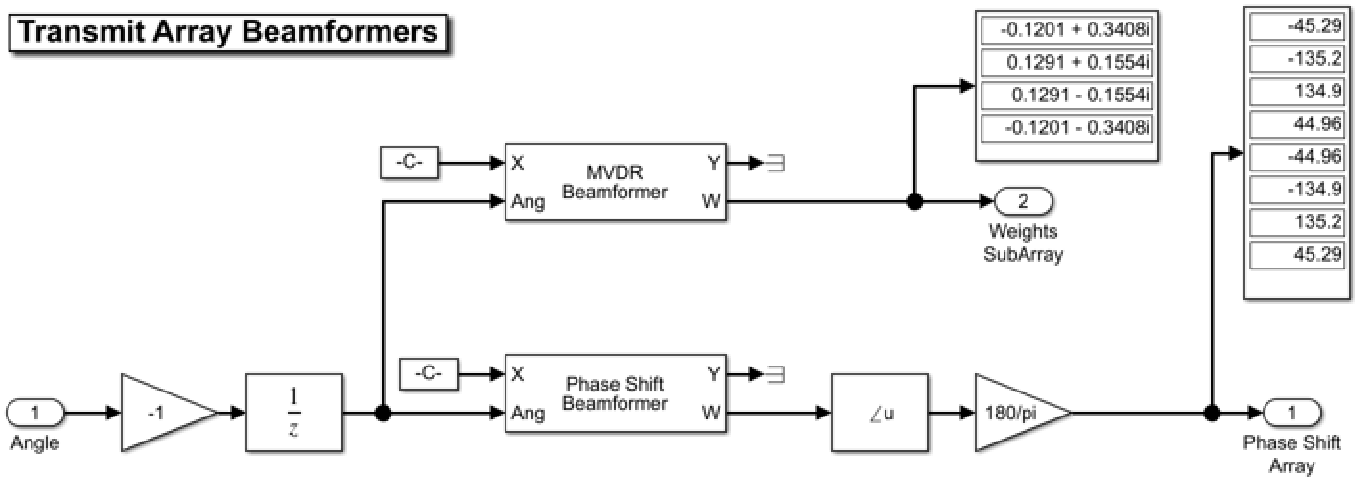
Transmit Array Beamformers

The transmit array is steered towards the direction estimated by the receiver. For demonstration purposes, two different beamforming algorithms are used to compute the weights applied to the four subarrays and to the elements of each subarray.

The subarrays weights are computed with an MVDR beamformer. A complex multiplication in the MVDR beamformer combines the transmitted signal and subarrays weights, steering the transmitted signal along the azimuth direction. Tapering is used to reduce the effects of grating lobes.

The phase shifts applied to the eight subarray elements are computed with a phase shifter beamforming algorithm. The four subarrays apply the same phase shifts that steer the transmitter along the elevation direction.

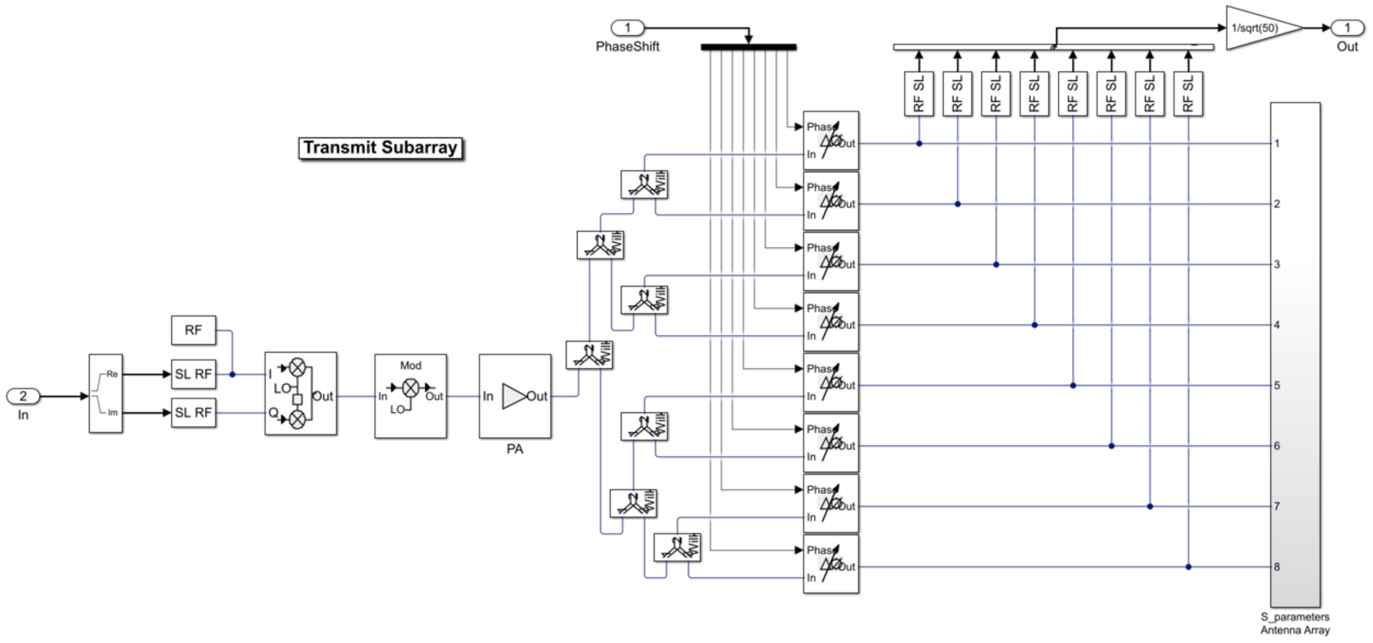
```
open_system([model '/Transmit Array Hybrid Beamforming/Beamformers'])
```



Transmit Subarrays

The four transmit subarrays are identical. Each subarray first performs upconversion at 5 GHz using a quadrature modulator, and then performs upconversion at 66 GHz using a superhet modulator including image and channel select filters. Each stage introduces impairments such as noise, I/Q imbalance, LO leakage, and non-linearity. A non-linear power amplifier increases the transmitter gain, and a network of Wilkinson power dividers connect the PA to the 8 antennas. Eight variable phase shifters are used to steer the beam. The loading of the antenna subarray and the coupling in between the antenna elements is modeled by its S-parameters.

```
open_system([model '/Transmit Array Hybrid Beamforming/subarray1'])
```



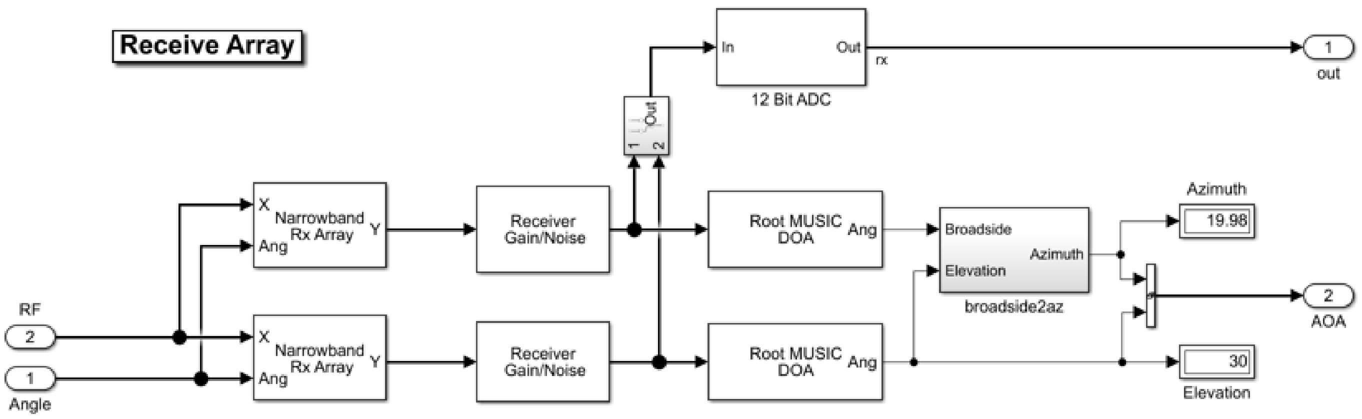
Receive Array

The receiver is modeled at a higher abstraction level compared to the transmitter. The receiver uses two orthogonal linear arrays, each with 4 isotropic antenna elements. The arrays are used to provide spatial diversity for the identification of the angle of arrival. The receiver does not implement any beamforming algorithm.

The receiver finite gain and SNR is modeled for each of the received signals followed by a 12-bit ADC with finite dynamic range including saturation and quantization effects.

Two root MUSIC algorithms are used to estimate the direction of arrival using the linear array signals. Each algorithm operates across one dimension, thus together can estimate the transmitter position in terms of azimuth and elevation angles.

```
open_system([model '/Receive Array'])
```

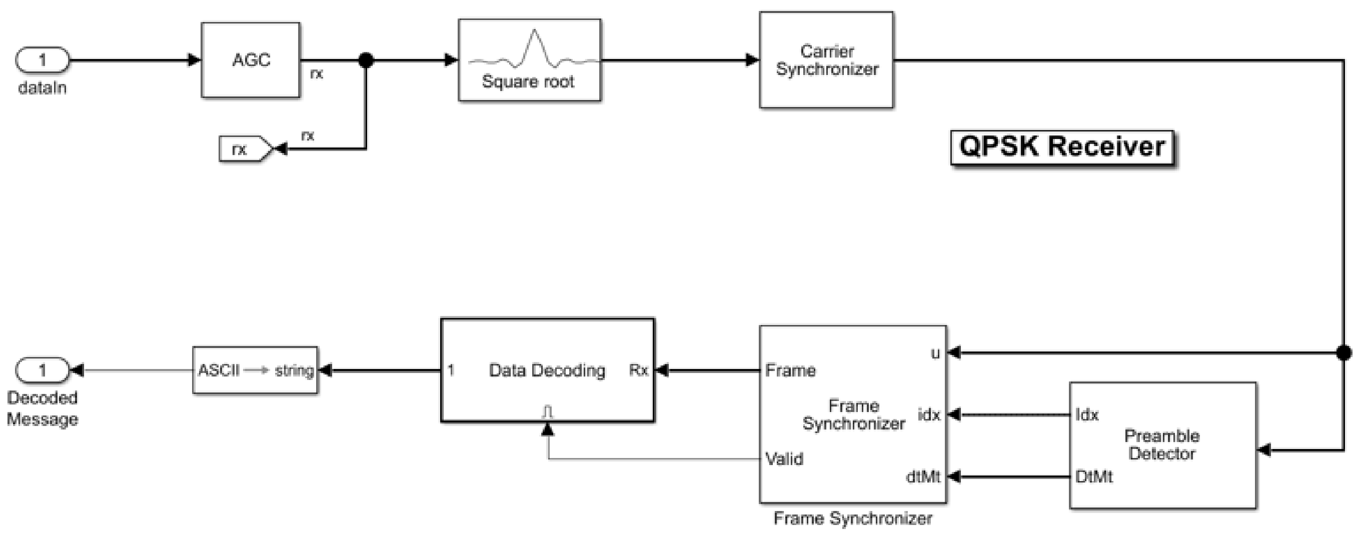


QPSK Receiver

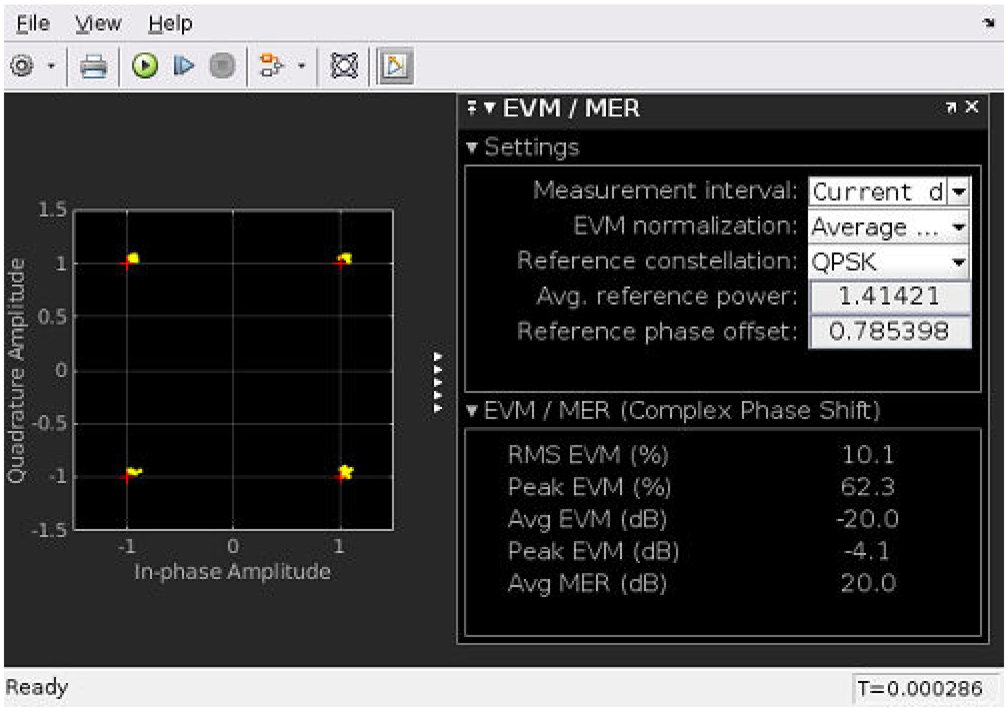
The QPSK receiver from the Communications Toolbox™ example “QPSK Transmitter and Receiver” (Communications Toolbox) is used in this example with modification. These modifications remove blocks from this receiver when the signal impairment is absent.

- The AGC controls and stabilizes the received signal amplitude which affects the accuracy of the carrier symbol synchronizer.
- The Raised Cosine Receive Filter provides matched filtering for the transmitted waveform.
- The Carrier Synchronizer Block performs fine frequency compensation.
- The Preamble Detector block uses the known frame header (QPSK-modulated Barker code) to correlate against the received QPSK symbols to find the location of the frame header.
- The Frame Synchronizer block uses the frame location information from the Preamble Detector to align the frame boundaries. The second output of the block is a boolean scalar indicating if the first output is a valid frame with the desired header and if so, enables the Data Decoding subsystem to run.
- The Data Decoding enabled subsystem performs phase ambiguity resolution, demodulation and text message decoding.

```
open_system([model '/QPSK Receiver'])
```



The input signal constellation for the data decoder QPSK Demodulator is



EVM / MER

Settings

Measurement interval: Current d
EVM normalization: Average ...
Reference constellation: QPSK
Avg. reference power: 1.41421
Reference phase offset: 0.785398

EVM / MER (Complex Phase Shift)

RMS EVM (%)	10.1
Peak EVM (%)	62.3
Avg EVM (dB)	-20.0
Peak EVM (dB)	-4.1
Avg MER (dB)	20.0

```
bdclose(model)
clear model;
```

See Also

IQ Demodulator | Mixer | Power Amplifier

Related Topics

“Modeling and Simulation of MIMO RF Receiver Including Beamforming” on page 9-154 | “Wireless Digital Video Broadcasting with RF Beamforming” on page 9-167

Wireless Digital Video Broadcasting with RF Beamforming

This example shows how to model a digital video broadcasting system which includes a 16 antenna phased array receiver operating at 28 GHz. The baseband transmitter, receiver and channel are realized with Communications Toolbox™. The RF receiver is implemented with the RF Blockset™ Circuit Envelope library, and the receive phased array antennas are constructed using Phased Array System Toolbox™. The 4 x 4 planar phased array feeds a 16 channel receive module that includes phase shifters to enable RF beamforming.

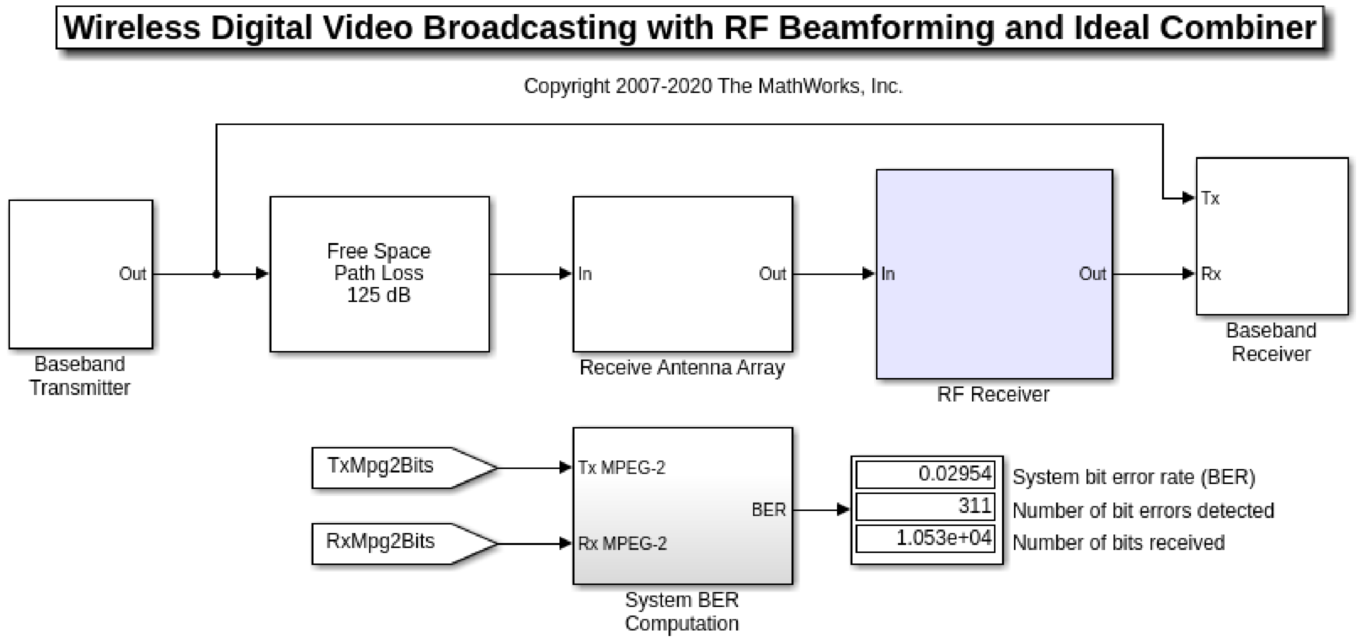
System Architecture

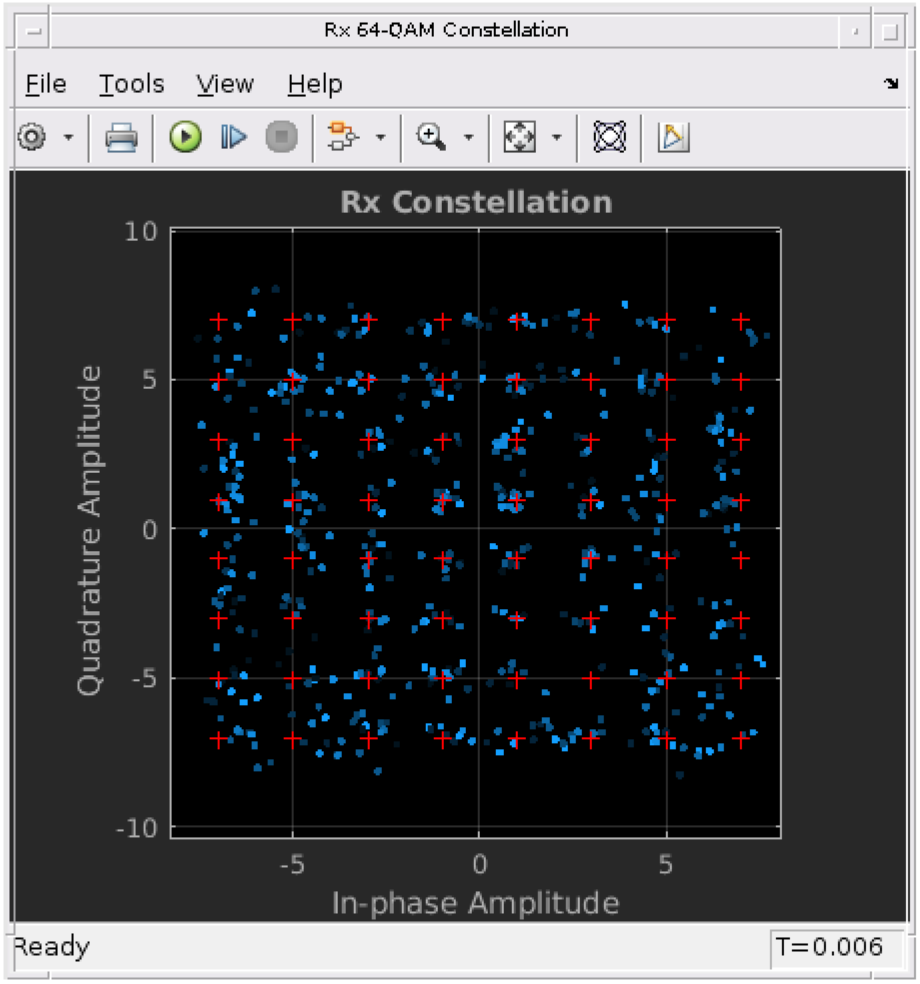
The system consists of:

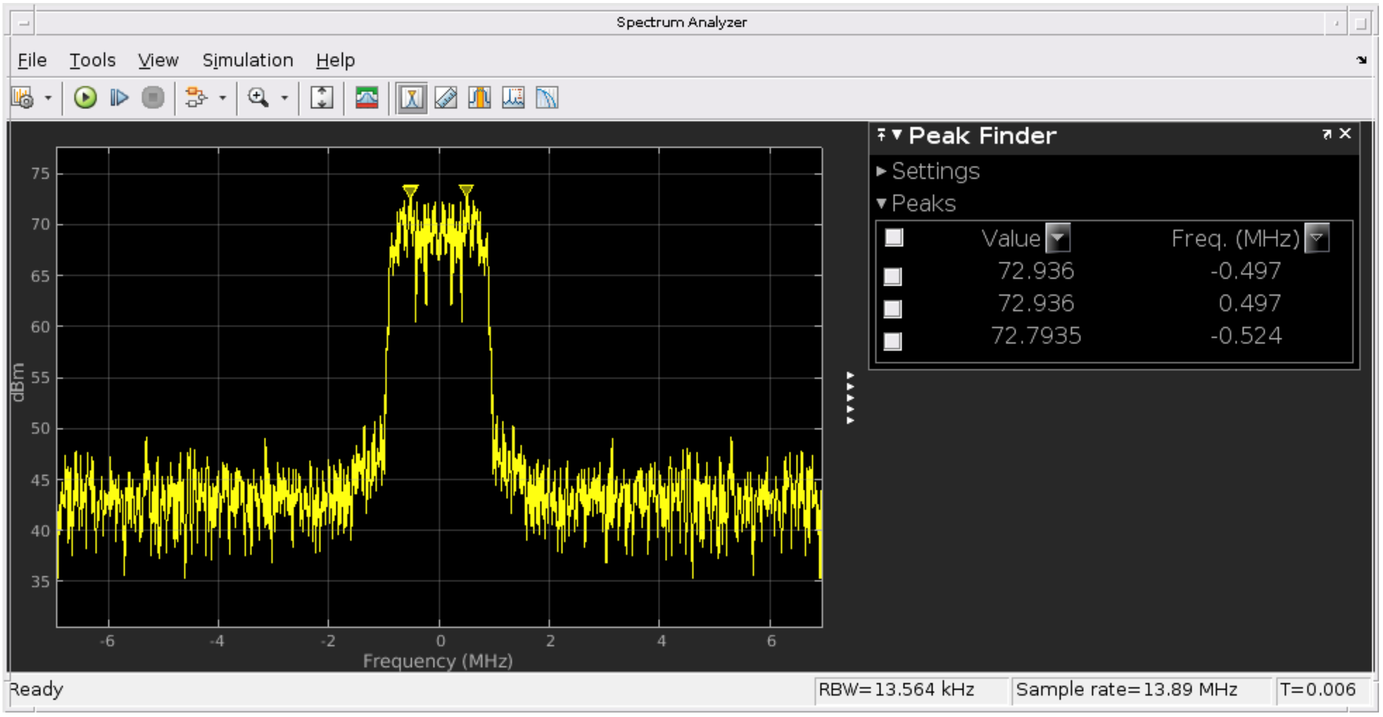
- A Baseband Transmitter subsystem that is responsible for generating a 64-QAM signal occupying 2 MHz of bandwidth that adheres to the DVB-C standard.
- Channel effects in the form of path loss.
- A 16 element phased array receiver arranged in a 4 X 4 rectangular grid. This includes design parameters for operating frequency, element radiation pattern, and receive direction.
- An RF receiver module consisting of 16 paths combined with a network of 2:1 power combiners and then downconverted to baseband. Each path includes LNAs and variable phase shifters for RF beamforming. The network of 2:1 power combiners is constructed twice to emulate a typical design process. The initial design employs ideal Wilkinson power dividers as behavioral combiners, while the second implementation uses actual hardware modeled with `S-Parameters` blocks and measured data supplied via a Touchstone™ file (wireless.s3p).
- A Baseband Receiver subsystem that is responsible for extracting the transmitted signal. The receiver includes simple models for correcting phase offsets and gain control effects.

Diagnostics are available at various stages in the system using the received constellation, the bit error rate calculation, and the received spectrum.

```
model_ideal = 'simrfV2_wirelessdvb_beamform_ideal';
open_system(model_ideal)
sim(model_ideal)
```


Wireless Digital Video Broadcasting with RF Beamforming and Ideal Combiner
Copyright 2007-2020 The MathWorks, Inc.
Out
Baseband Transmitter
Free Space Path Loss 125 dB
In
Out
Receive Antenna Array
In
Out
RF Receiver
Tx
Rx
Baseband Receiver
TxMpg2Bits
RxMpg2Bits
Tx MPEG-2
Rx MPEG-2
BER
System BER Computation
0.02954
311
1.053e+04
System bit error rate (BER)
Number of bit errors detected
Number of bits received


Rx 64-QAM Constellation
File
Tools
View
Help
Rx Constellation
Quadrature Amplitude
In-phase Amplitude
10
5
0
-5
-10
-5
0
5
Ready
T=0.006

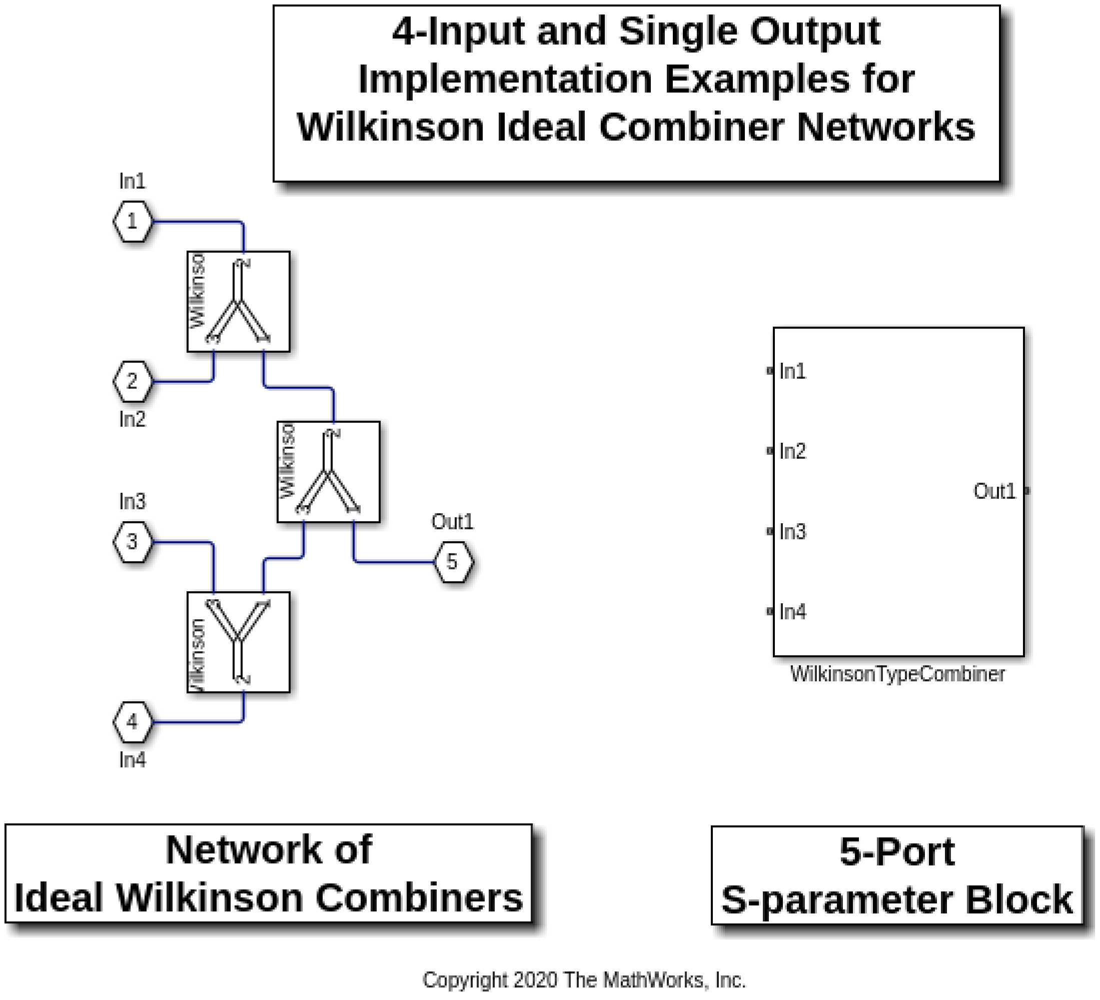


	Value	Freq. (MHz)
	72.936	-0.497
	72.936	0.497
	72.7935	-0.524

Designing with Ideal Components

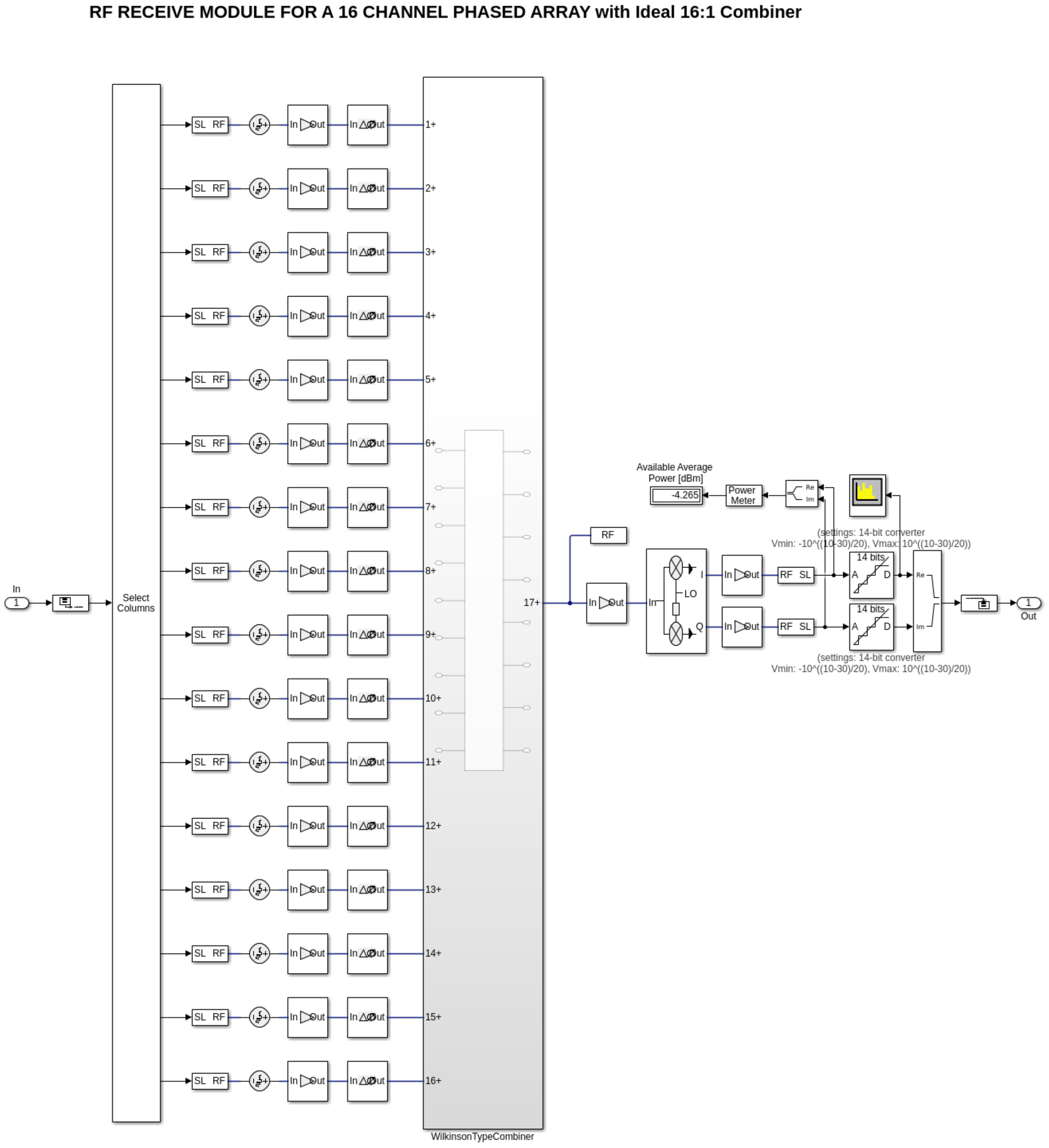
An initial design may use ideal components to speed up the overall design process. For example, ideal Wilkinson dividers from the RF Blockset Junctions library can be used to build a combiner system in the Receive Antenna Array. This combiner system can be consolidated into a 17-port `S-parameters` block to improve simulation performance. A simplified example of consolidation is shown where the 4:1 combiner system on the left is replaced with the 5-port S-parameters block on the right. The `S-parameters` block entries are calculated in the WilkinsonTypeCombiner block mask initialization commands.

```
model_combiner = 'simrfV2_wirelessdvb_beamform_prototype_combiner';
open_system(model_combiner)
```


4-Input and Single Output
Implementation Examples for
Wilkinson Ideal Combiner Networks
In1
1
2
In2
In3
3
4
In4
Out1
5
Wilkinson
In1
In2
In3
In4
Out1
WilkinsonTypeCombiner
Network of
Ideal Wilkinson Combiners
5-Port
S-parameter Block
Copyright 2020 The MathWorks, Inc.

Each input channel branch of the RF Receiver employs a separate amplifier to introduce thermal noise and non-linearities. The quadrature demodulator following the combiner performs direct downconversion and includes its non-linearities, LO leakage, I/Q mismatch, and noise impairments.

```
open_system([model_ideal '/RF Receiver'], 'force')
```

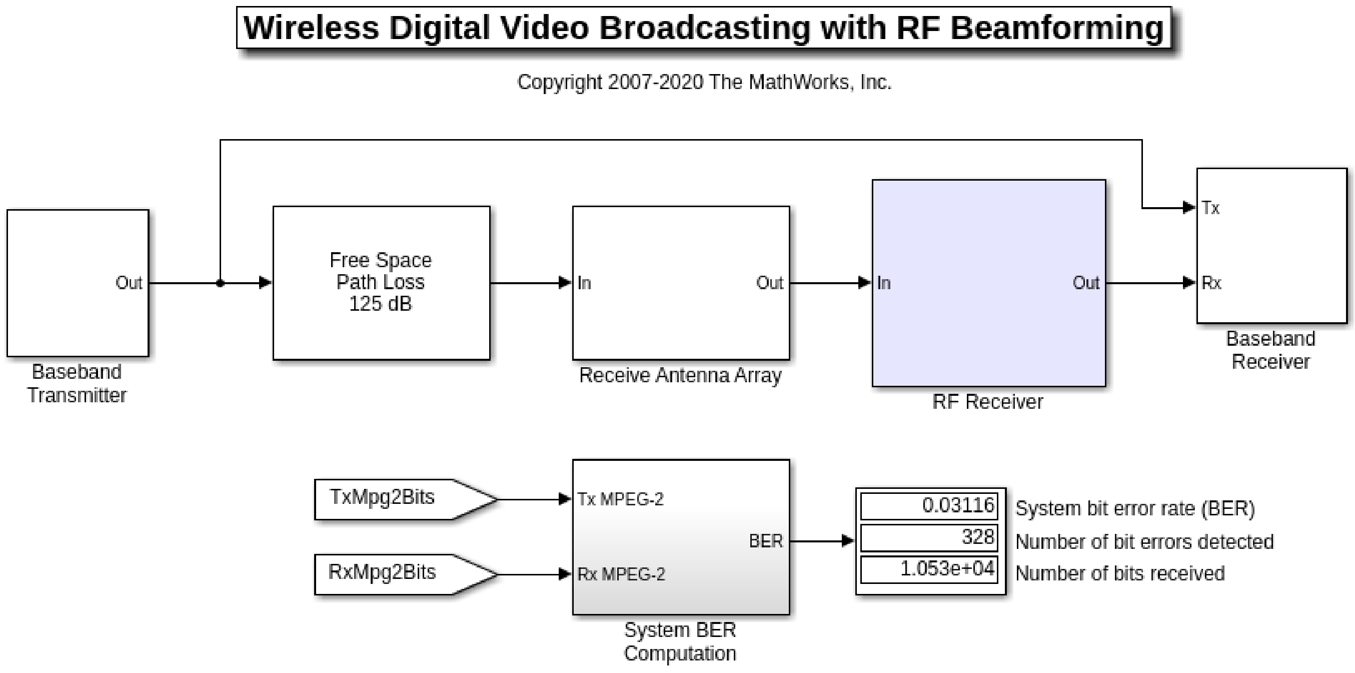

RF RECEIVE MODULE FOR A 16 CHANNEL PHASED ARRAY with Ideal 16:1 Combiner
In
1
Select Columns
SL RF
In Out
1+
2+
3+
4+
5+
6+
7+
8+
9+
10+
11+
12+
13+
14+
15+
16+
17+
WilkinsonTypeCombiner
RF
In
LO
I
Q
RF SL
14 bits
A D
Re
Im
Available Average Power [dBm]
-4.265
Power Meter
(settings: 14-bit converter
Vmin: -10^((10-30)/20), Vmax: 10^((10-30)/20))
Out

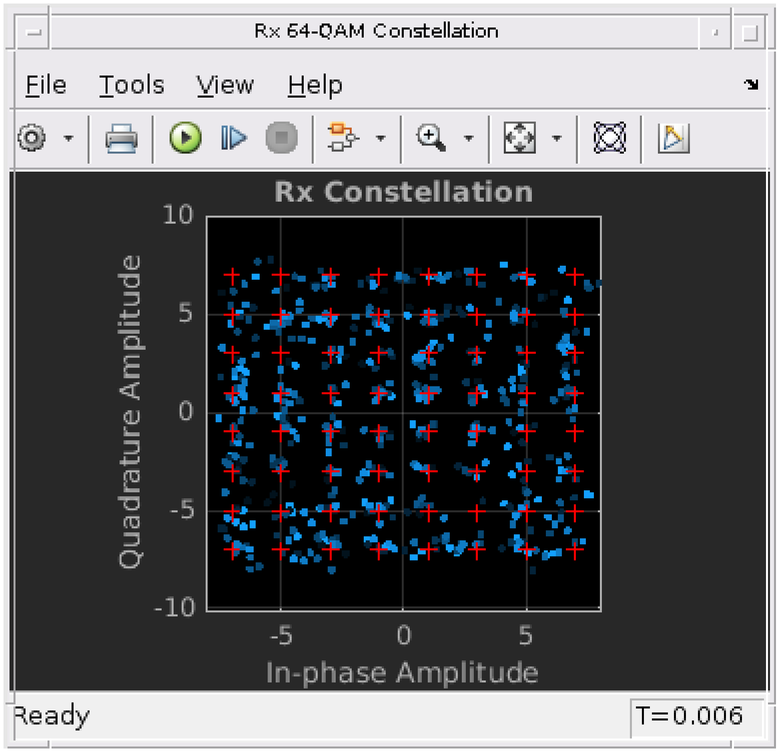
```
bdclose(model_combiner)
bdclose(model_ideal)
clear model_ideal
```

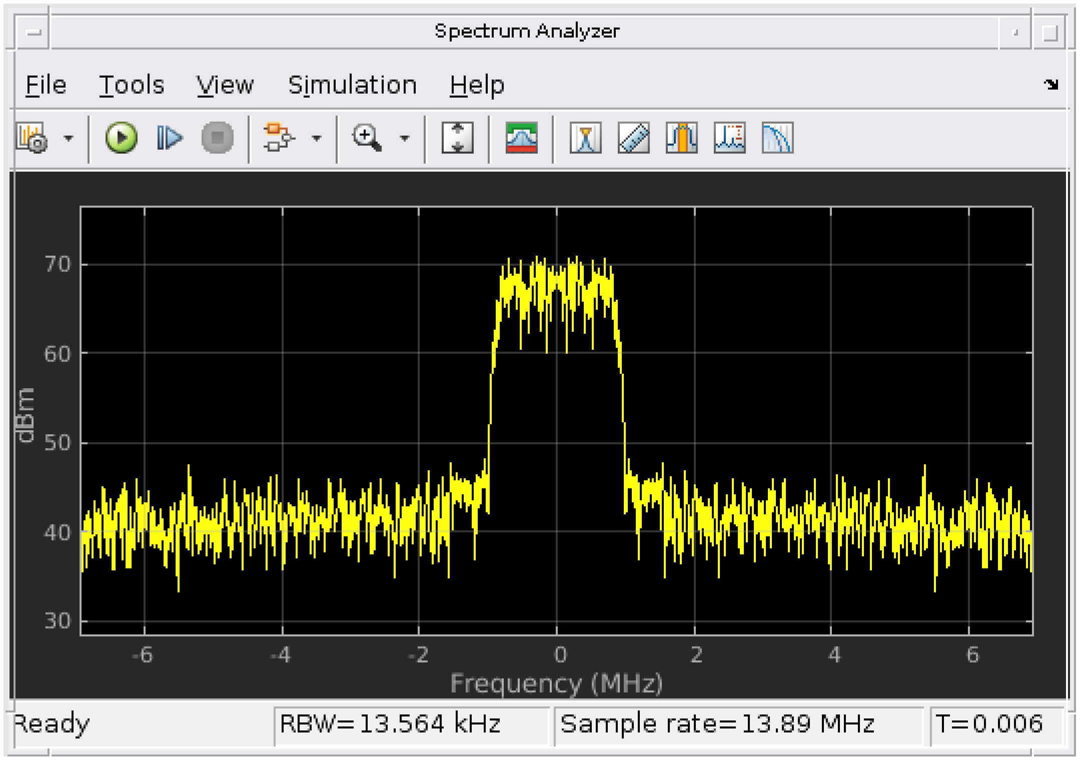
Designing with Real Components

Use `S-Parameters` blocks to model a real combiner system. There are several options available to characterize the behavior of each individual block in the combiner system; one approach utilizes a data file directly while another approach provides a rational model of the data. For the latter approach, utilize the `rational` function in RF Toolbox™, save the resulting parameters in the base workspace and use them in the `S-Parameters` blocks. For this example, the measured data is described in the Touchstone file wireless.s3p and used directly. To improve simulation performance the combiner system js replaced with a 17-port `S-Parameters` block. The `S-Parameters` block entries are calculated using the function simrfV2_wirelessdvb_beamforming_findcombinerspars in the Initization commands of the the RF Receiver subsystem mask.

```
model = 'simrfV2_wirelessdvb_beamforming';
open_system(model)
sim(model)
```





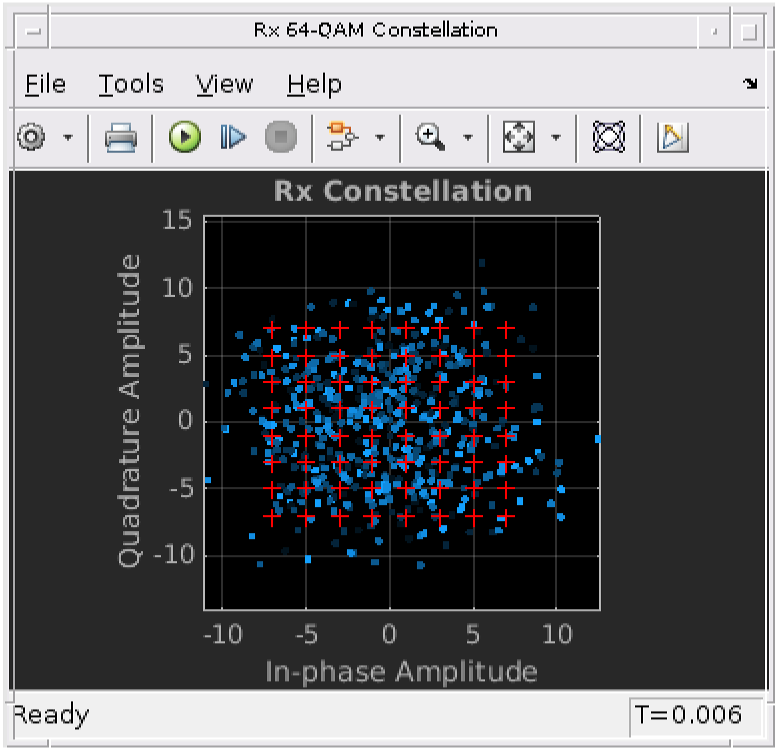


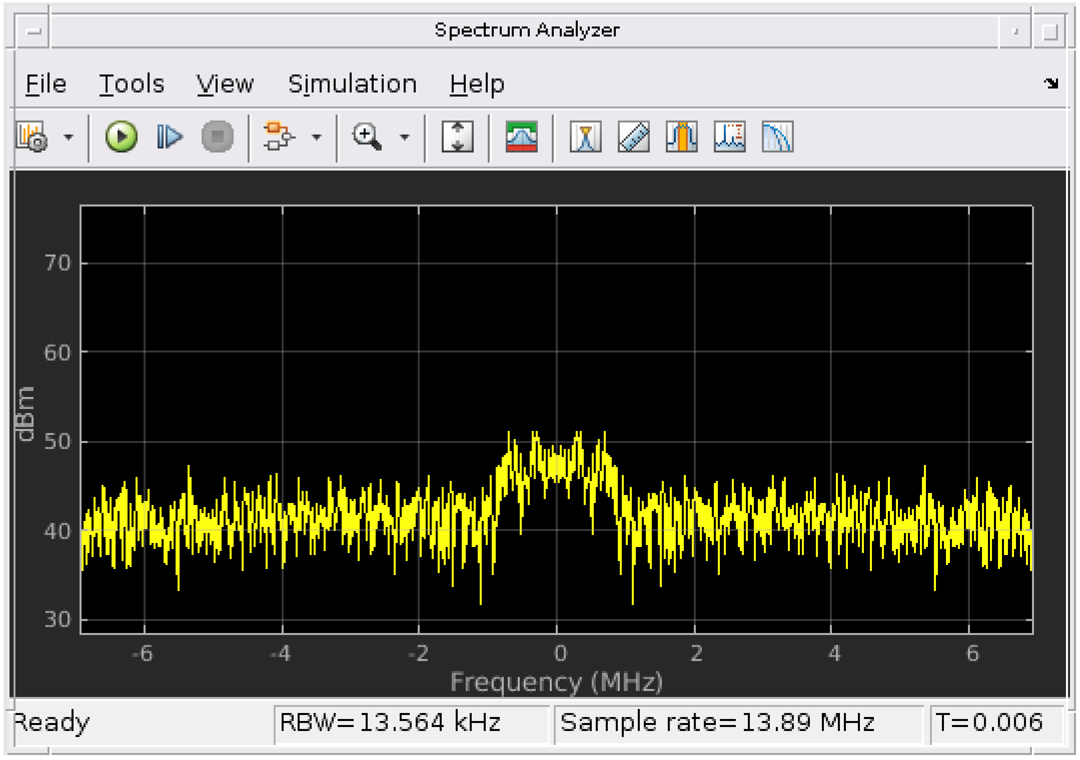
The transmit side planar array is chosen to have 16 elements and transmits along the main beam (azimuth = 0 deg. and elevation = 0 deg.) at a frequency of 28 GHz. An isotropic radiation pattern is chosen for each element. Note that the power dividers introduce a phase shift at 28 GHz. This is estimated and corrected in the Baseband receiver subsystem.

Modify the Receive Direction and Simulate

Modify the receive direction by changing the **Receive Direction** mask dialog parameter of the 16-element Receive Antenna Array. The angle chosen decreases the signal strength due to the proximity of a null in the array radiation pattern.

```
open_system(model)
set_param([model '/Receive Antenna Array'],'RecDir','[20;25]')
sim(model)
```

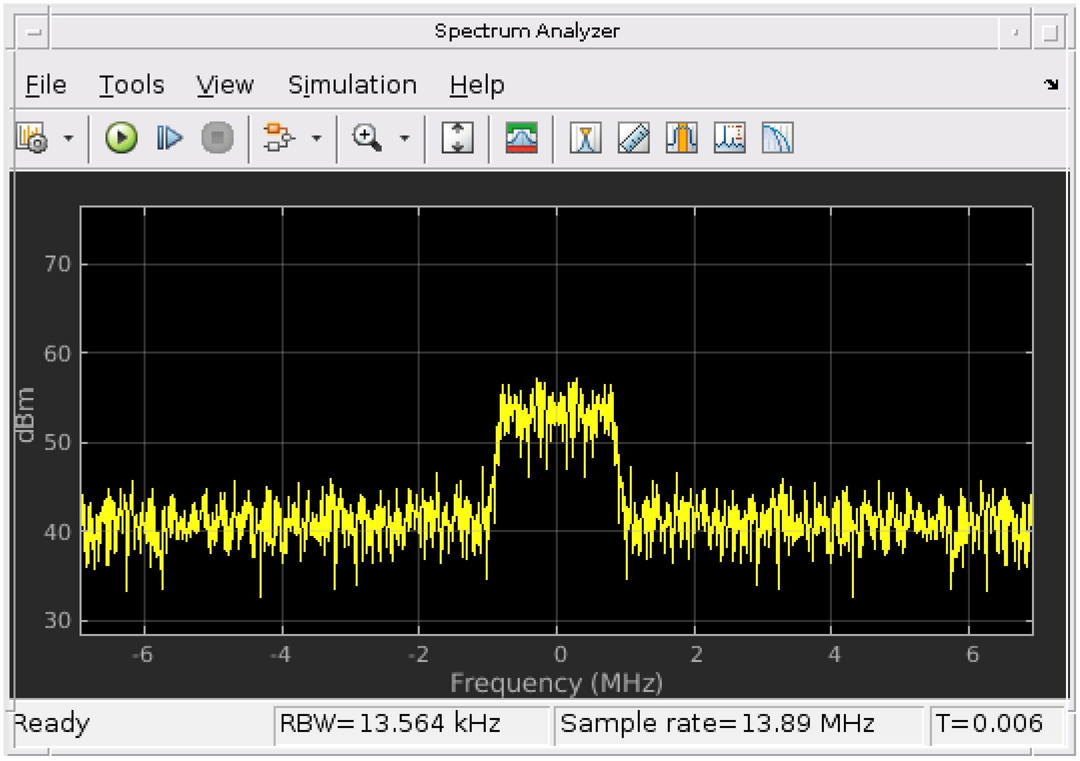


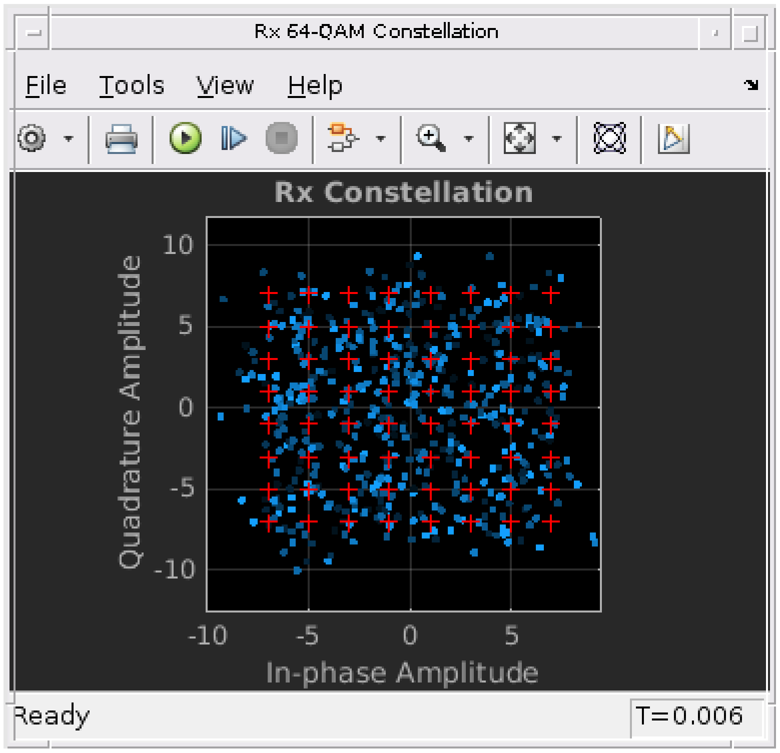


Improve RF Reception with Beamforming

Modify the **Beamforming direction** parameter for the 4 X 4 phased array on the receive side. This mask parameter will automatically adjust the phase shift of each channel in the RF Receiver subsystem. Run the simulation to observe an increase in the received signal level.

```
open_system(model)
set_param([model '/RF Receiver'],'BeamDir','[20;25]')
sim(model)
```





```
bdclose(model)
clear model
```

References

S. Emami, R. F. Wiser, E. Ali, M. G. Forbes, M. Q. Gordon, X. Guan, S. Lo, P. T. McElwee, J. Parker, J. R. Tani, J. M. Gilbert,, and C. H. Doan, "A 60 GHz CMOS Phased-Array Transceiver Pair for Multi-Gb/s Wireless Communications," in IEEE Int. Solid-State Circuits Conf. Tech. Dig., Feb. 2011, pp. 164-165

See Also

More About

- “Modeling an RF mmWave Transmitter with Hybrid Beamforming” on page 9-160
- “Modeling and Simulation of MIMO RF Receiver Including Beamforming” on page 9-154

Top-Down Design of an RF Receiver

This example designs an RF receiver for a ZigBee®-like application using a top-down methodology. It verifies the BER of an impairment-free design, then analyzes BER performance after the addition of impairment models. The example uses the `RF Budget Analyzer` App to rank the elements contributing to the noise and nonlinearity budget.

Design specifications:

- Data rate = 250 kbps
- OQPSK modulation with half sine pulse shaping, as specified in IEEE® 802.15.4 for the physical layer of ZigBee
- Direct sequence spread spectrum with chip rate = 2 Mchips/s
- Sensitivity specification = -100 dBm
- Bit Error Rate (BER) specification = 1e-4
- Analog to digital converter (ADC) with 10 bits and 0 dBm saturation power

To create fully standard-compliant ZigBee waveforms, you can use the Communications Toolbox Library for the ZigBee Protocol Add-on.

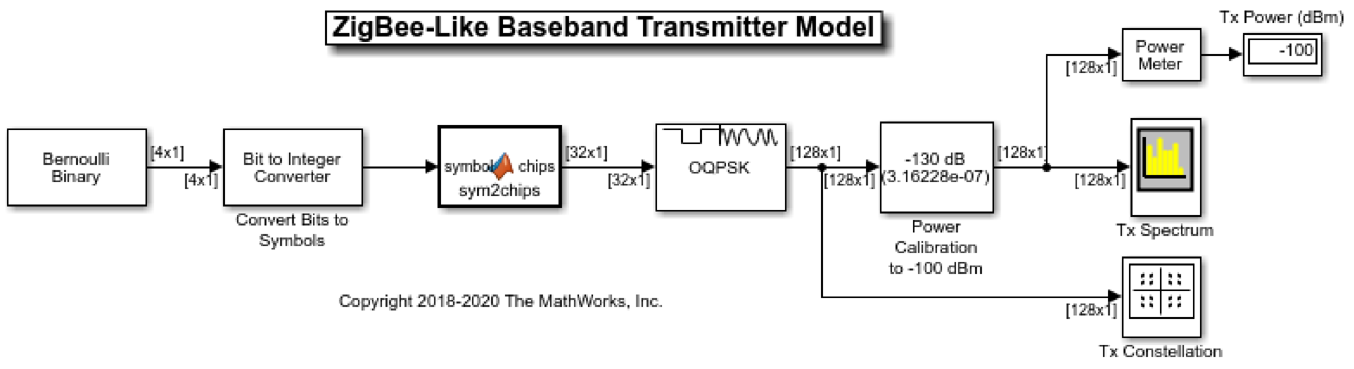
This example guides you through the following steps:

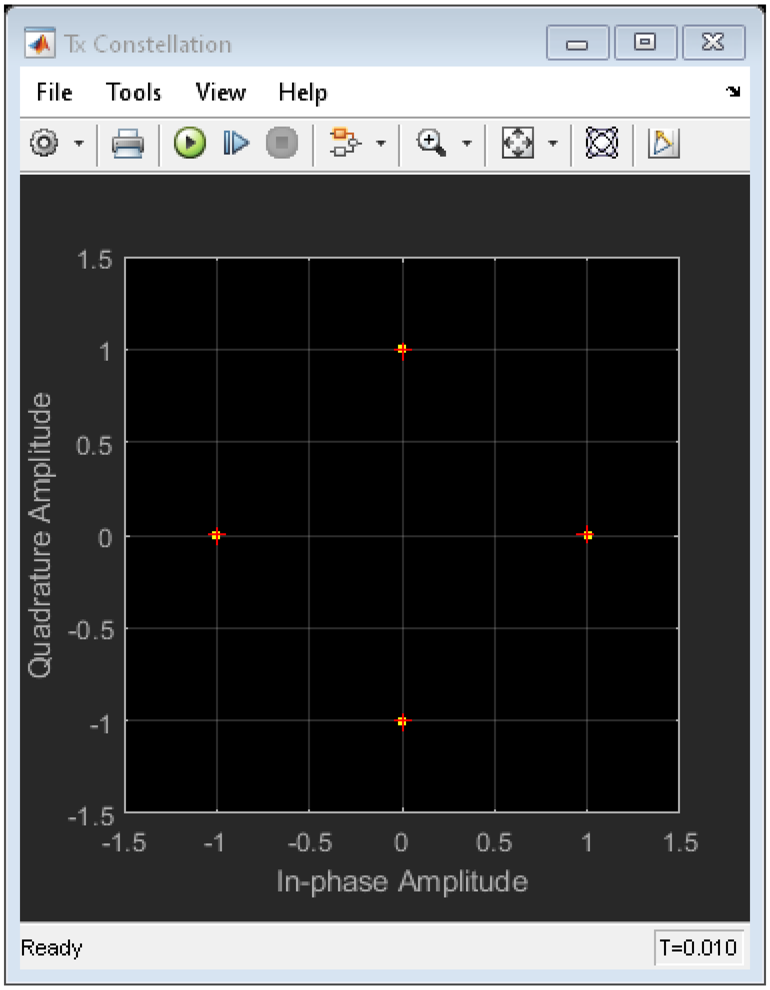
- Develop the baseband transmitter model for waveform generation
- Determine SNR specification to achieve the 1e-4 BER from a link-level idealized baseband model
- Derive RF subsystem specifications from equivalent-baseband model of RF receiver and ADC
- Derive direct conversion specifications from circuit envelope model of RF receiver
- Perform multi-carrier simulation including interfering signals and derive the specifications of the DC offset compensation algorithm

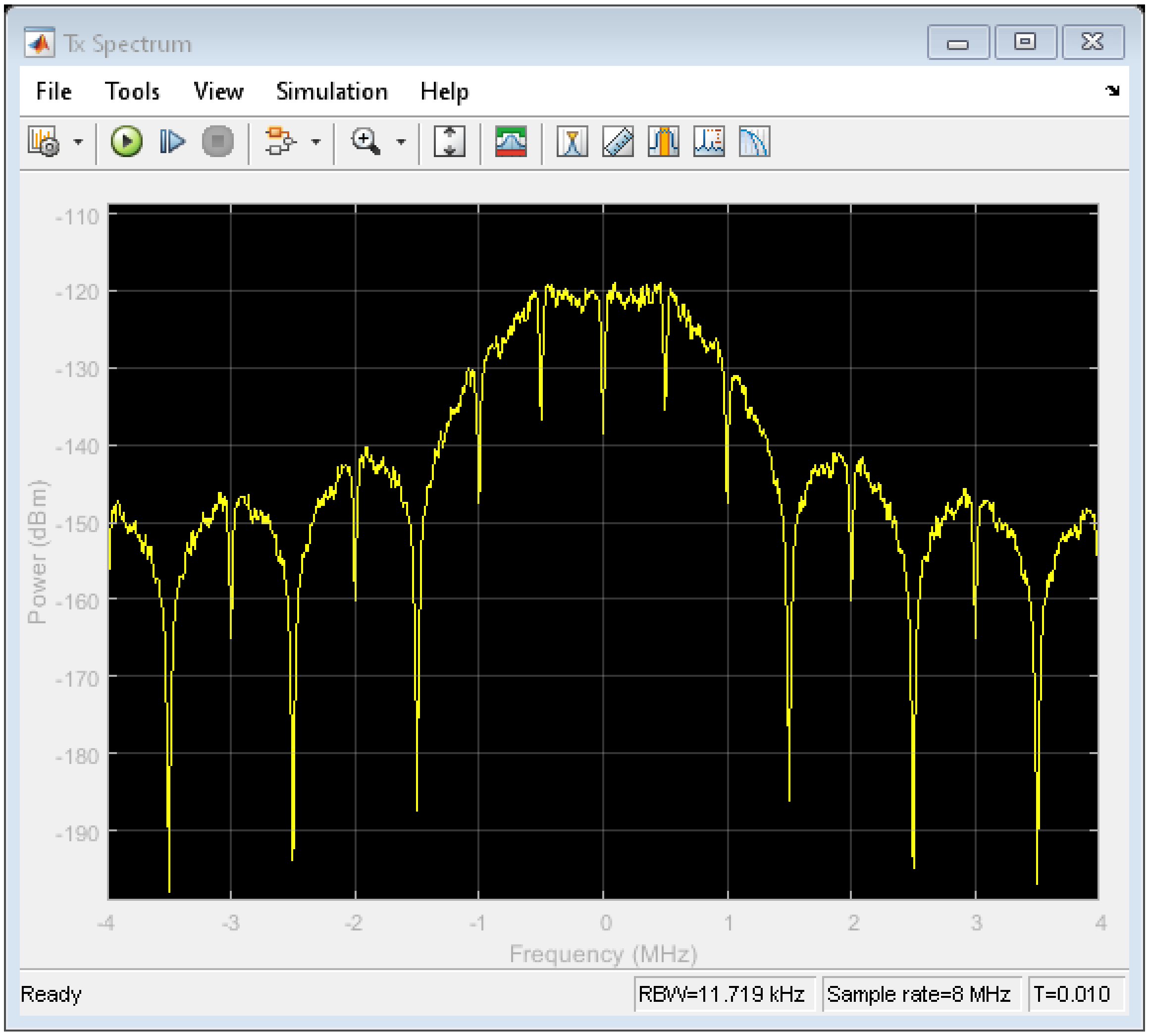
Design and Verify Baseband Transmitter

To evaluate the performance of the RF receiver design, it is necessary and sufficient to use a signal spectrally representative of an 802.15.4 waveform.

The baseband transmitter model creates and illustrates a spectrally representative ZigBee waveform in the spectral and constellation domains. This model and all the subsequent models use callbacks to create MATLAB workspace variables that parameterize the systems.



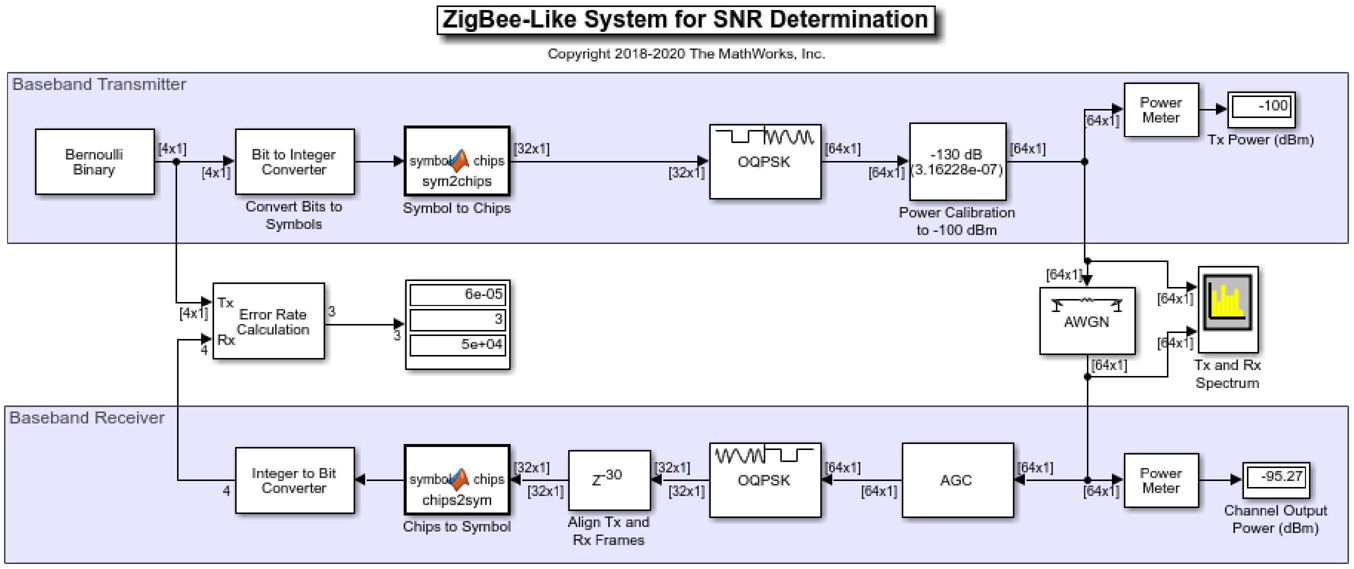

Tx Constellation
File
Tools
View
Help
1.5
1
0.5
0
-0.5
-1
-1.5
Quadrature Amplitude
-1.5
-1
-0.5
0
0.5
1
1.5
In-phase Amplitude
Ready
T=0.010



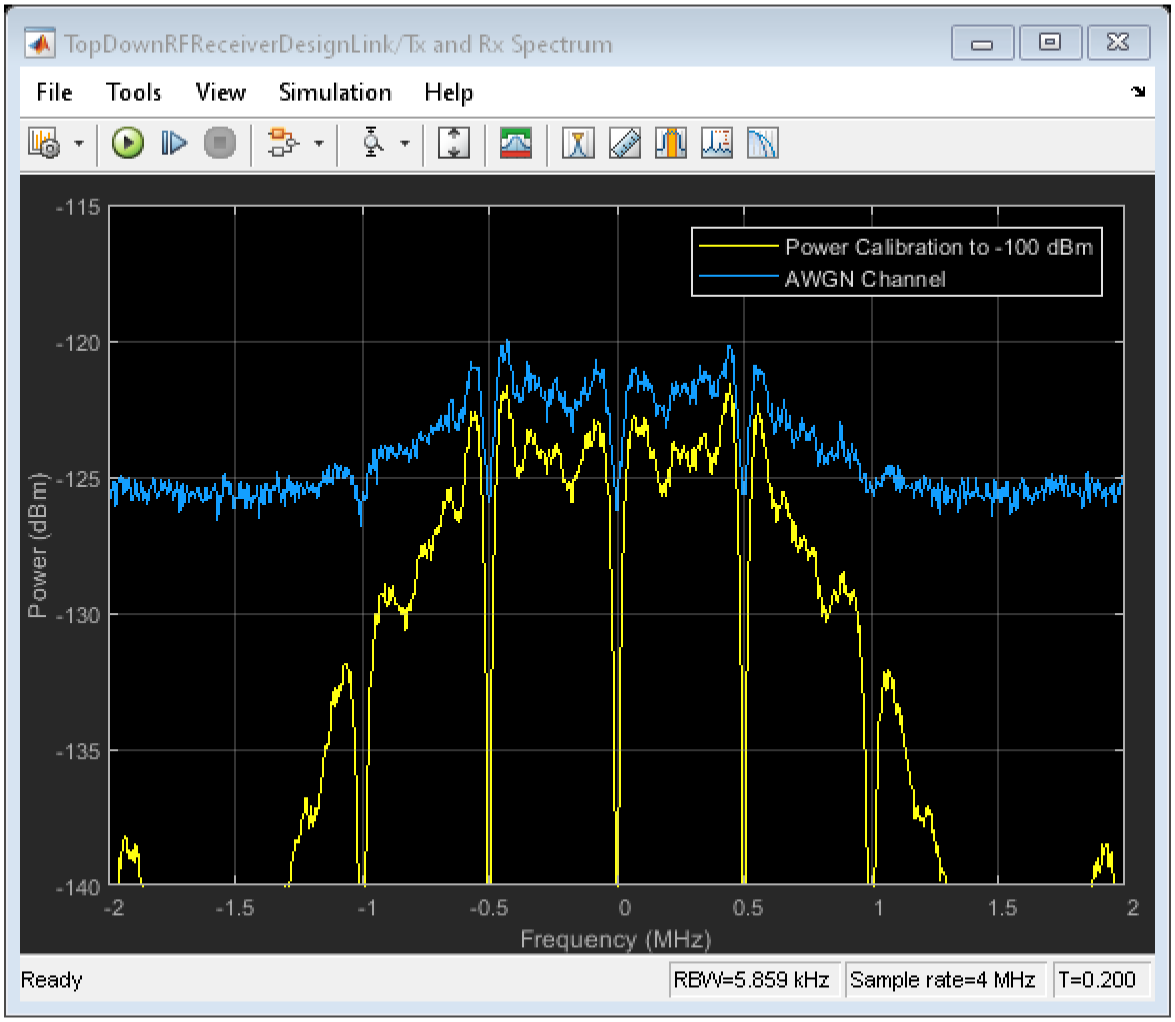
Determine Receiver SNR Requirement

To design the receiver, first determine the SNR needed to achieve the specified BER less than 1e-4. calculated in the simulation bandwidth of 4 MHz. Run the link-level model to simulate the receiver processing required to achieve the target BER.

Computing the BER accurately requires alignment of the transmit and receive signals. The simulation must compensate for a two-sample delay of the received signal compared to the transmitted signal. Also, to ensure correct chip-to-symbol-to-bit mapping, the simulation must align the chips to frame boundaries at the input to the Chips to Symbol block on a frame boundary. Accounting for the receive signal delay and the frame boundary alignment requires addition of a **Delay** block set to a 32-2=30 delay on the receiver branch before recovering the received symbols.



ZigBee-Like System for SNR Determination
Copyright 2018-2020 The MathWorks, Inc.
Baseband Transmitter
Bernoulli Binary
Bit to Integer Converter
Convert Bits to Symbols
symbol chips sym2chips
Symbol to Chips
OQPSK
-130 dB (3.16228e-07)
Power Calibration to -100 dBm
Power Meter
-100
Tx Power (dBm)
Tx Rx Error Rate Calculation
6e-05
3
5e+04
AWGN
Tx and Rx Spectrum
Baseband Receiver
Integer to Bit Converter
symbol chips chips2sym
Chips to Symbol
z^{-30}
Align Tx and Rx Frames
OQPSK
AGC
Power Meter
-95.27
Channel Output Power (dBm)
[4x1]
[32x1]
[64x1]



TopDownRFReceiverDesignLink/Tx and Rx Spectrum
File
Tools
View
Simulation
Help
Power Calibration to -100 dBm
AWGN Channel
Power (dBm)
-115
-120
-125
-130
-135
-140
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
Frequency (MHz)
Ready
RBW=5.859 kHz
Sample rate=4 MHz
T=0.200

The model achieves a 1e-4 BER at an SNR of -2.7 dB, which can be verified by collecting 100 bit errors.

In the link-level model, the AWGN block accounts for the overall channel and RF receiver SNR budget.

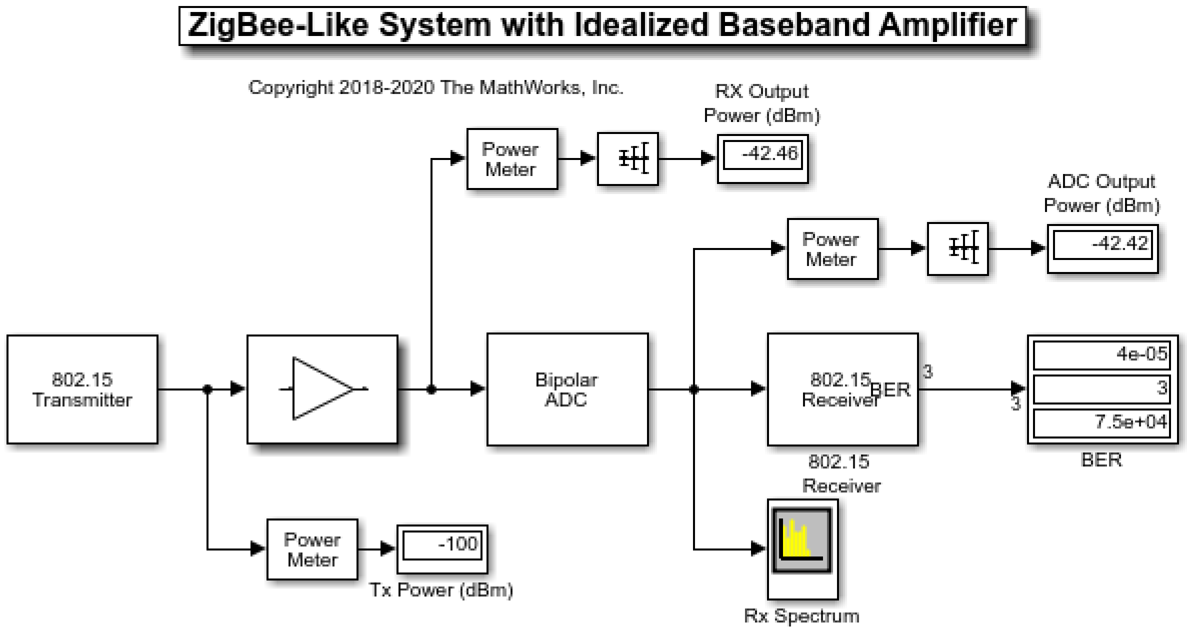
Add ADC and Determine Receiver Total Gain and Noise Figure (NF)

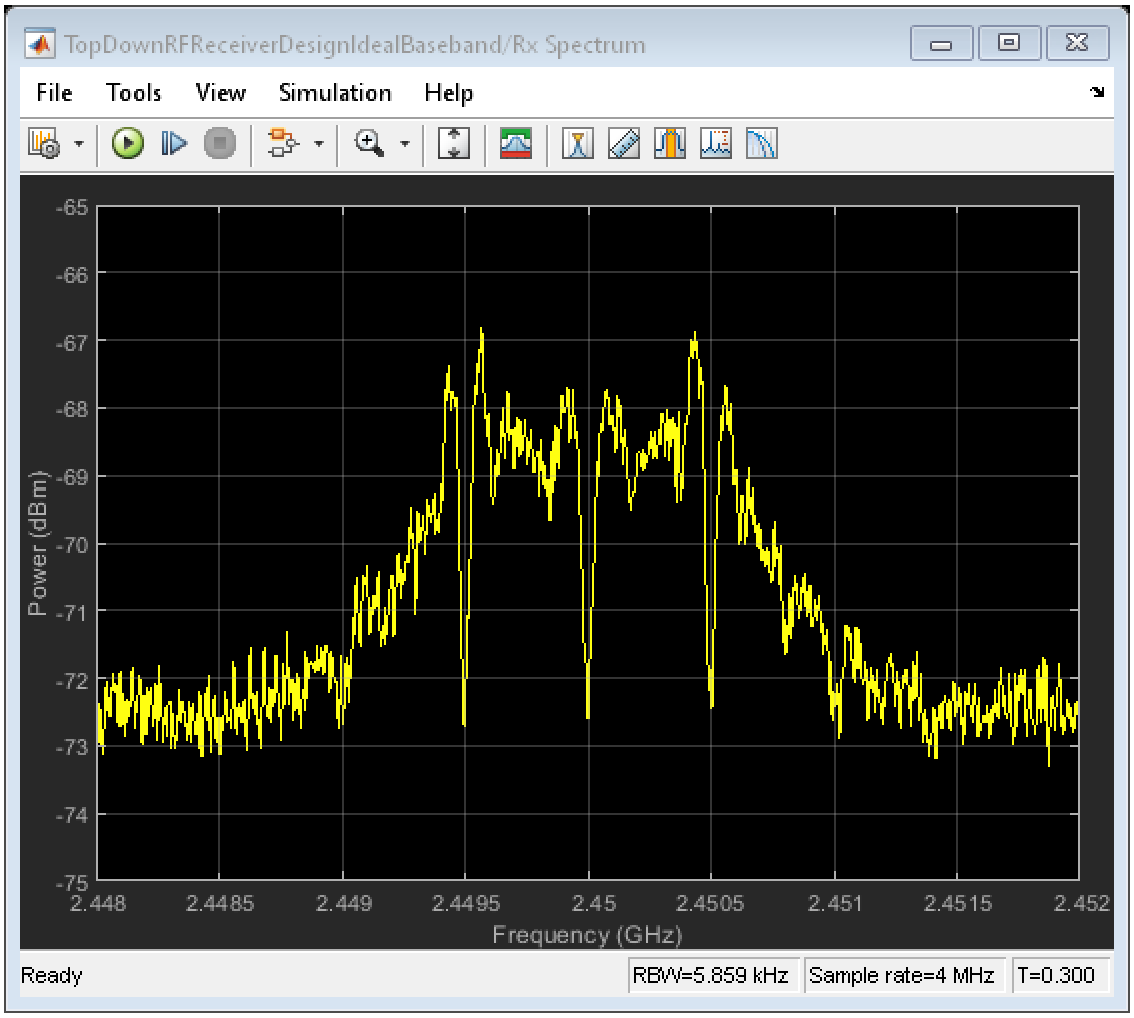
This section uses traditional heuristic derivations to determine the high-level specifications of the RF receiver and ADC.

- B = 4 MHz = simulation bandwidth = simulation sampling frequency
- kT = 174 dBm/Hz = thermal noise floor power
- Sensitivity = -100 dBm = receiver sensitivity
- SNR = -2.7 dB
- Noise power in simulation bandwidth = Pn = sensitivity-SNR = -100 dBm - (-2.7 dB) = -97.3 dBm

Simulating an idealized baseband model of the RF Receiver, verify the preliminary RF receiver specifications (NF = 10.7 dB and receiver gain = 53.4 dB). This can be done by collecting 100 errors.

The spectrum analyzer shows that the received spectrum with the ADC is roughly identical in shape to the spectrum of the previous section, without the ADC.





Refine Architectural Description of RF Receiver

In this section the RF receiver, and its noise figure and gain budget specifications, are modelled by using four discrete subcomponents with these characteristics:

- SAW Filter: Noise Figure = 2.5 dB, Gain = -3 dB
- LNA: Noise Figure = 6 dB, Gain = 22 dB
- Passive Mixer: Noise Figure = 10 dB, Gain = -5 dB
- VGA: Noise Figure = 14 dB, Gain = 40 dB

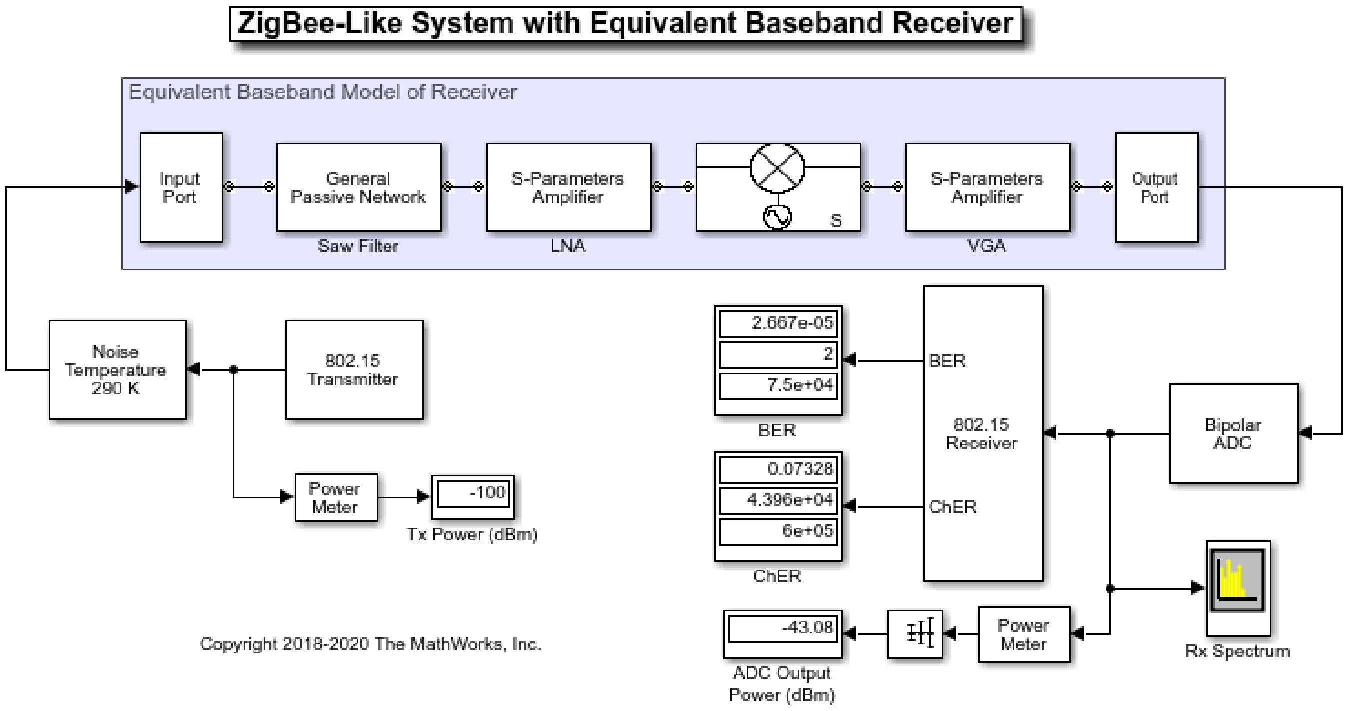
The SAW filter performance is derived from a Touchstone file that specifies S-parameters characteristics. You can verify the gain by visualizing the S21 parameter in the X-Y plane at the operating frequency of 2.45 GHz. You can verify the noise figure by visualizing the NF parameter in the X-Y plane at the operating frequency of 2.45 GHz. Typically, an LNA with low noise and high gain follows the SAW filter, which greatly reduces the impact of the noise figure of the components after the LNA. Also, the passive mixer is specified with a high IP2. Similar to the SAW filter, you can verify the mixer gain by visualizing the S21 parameter in the X-Y plane over a user-specified frequency range of [2e9 3e9].

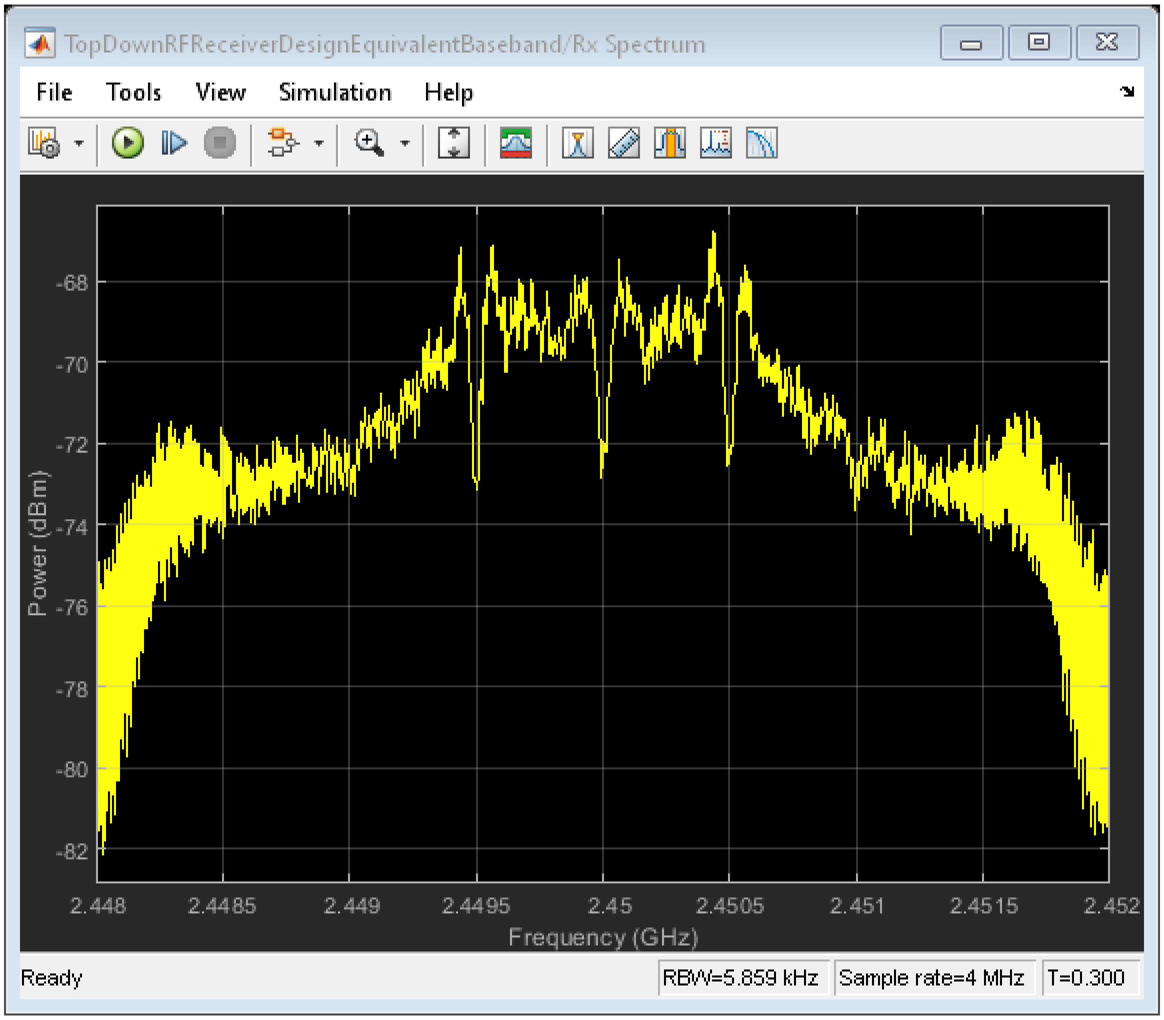
An equivalent baseband model simulates the refined RF receiver.

Run the simulation and verify the RF receiver link budget by using the output port visualization pane. The total noise figure and gain across the four stages has been divided according to the following budget:

- Component NF (dB) = [2.5, 6, 10, 14]
- Component noise factor F (linear) = 10^(NF/10) = [1.78 3.98 10.0 25.1]
- Power gain (dB) = [-3, 22, -5, 40] = 54 dB > 53.4 dB
- Voltage gain VG (linear) = 10^(Power gain/20) = [0.71 12.59 0.56 100.0]
- System noise factor Fsys (linear) =
$$1 + [F(1) - 1] + \frac{[F(2) - 1]}{VG(1)} + \frac{[F(3) - 1]}{VG(1) \times VG(2)} + \frac{[F(4) - 1]}{VG(1) \times VG(2) \times VG(3)} = 11.8$$
- System noise figure NFsys (dB) = 10*log10(Fsys) = 10.7 dB

With this model you can verify that a BER < 1e-4 corresponds to a Chip Error Rate (ChER) around 7%. By computing ChER, you can run the subsequent models for less time and still collect accurate BER statistics.





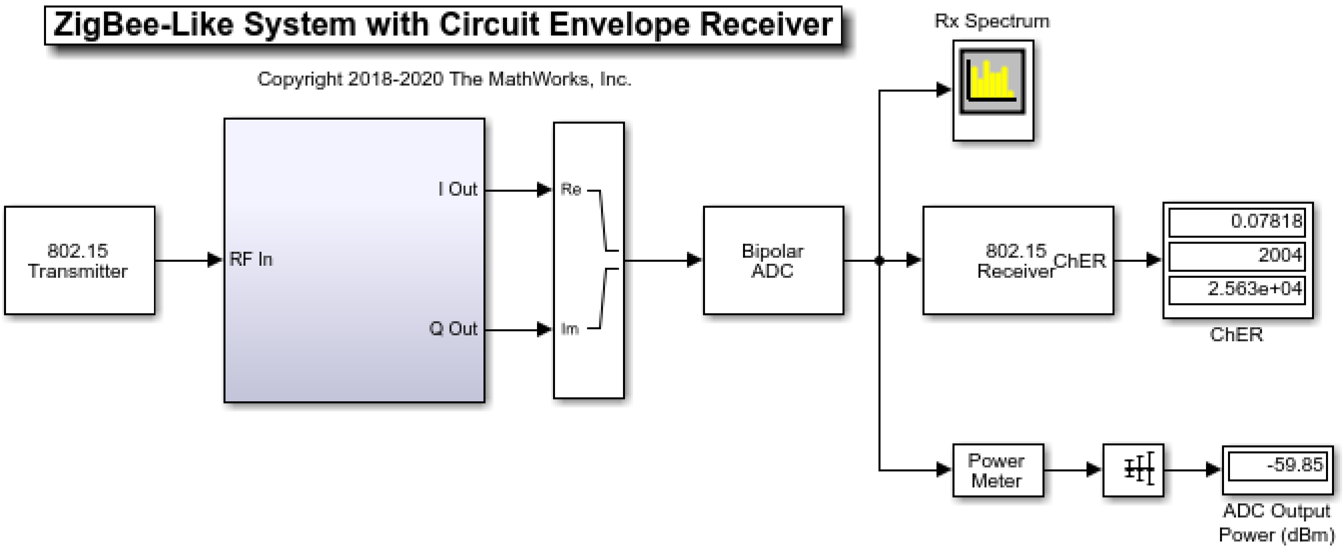
Use Circuit Envelope to Simulate Additional RF Impairments

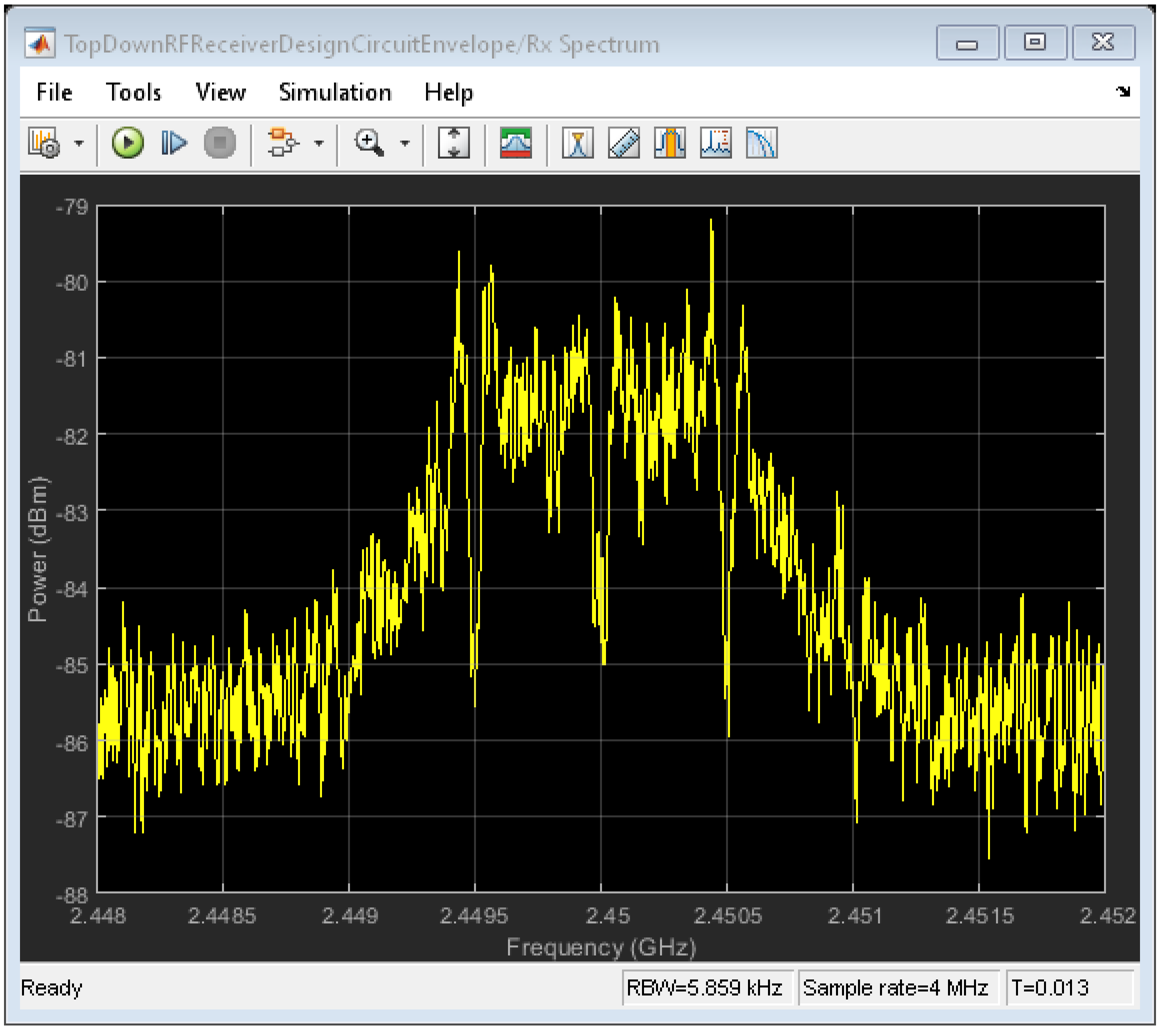
Using the circuit envelope modeling approach, continue refining the RF receiver architecture by adding more realistic impairments.

The circuit envelope model of the RF Receiver differs from the equivalent baseband model as it:

- Replaces the equivalent baseband mixer with a quadrature modulator, consisting of parameterizable I and Q mixers and phase shifter block, and an LO with impairments
- Uses broadband impedances (50 ohm) to explicitly model the power transfer between blocks

Comparing spectra, power measurements, and ChER to the equivalent baseband model, there are no significant performance differences. However, with the circuit envelope model, you can include even order nonlinearity effects, I/Q imbalance, and specifications of colored noise distributions for each of the components.

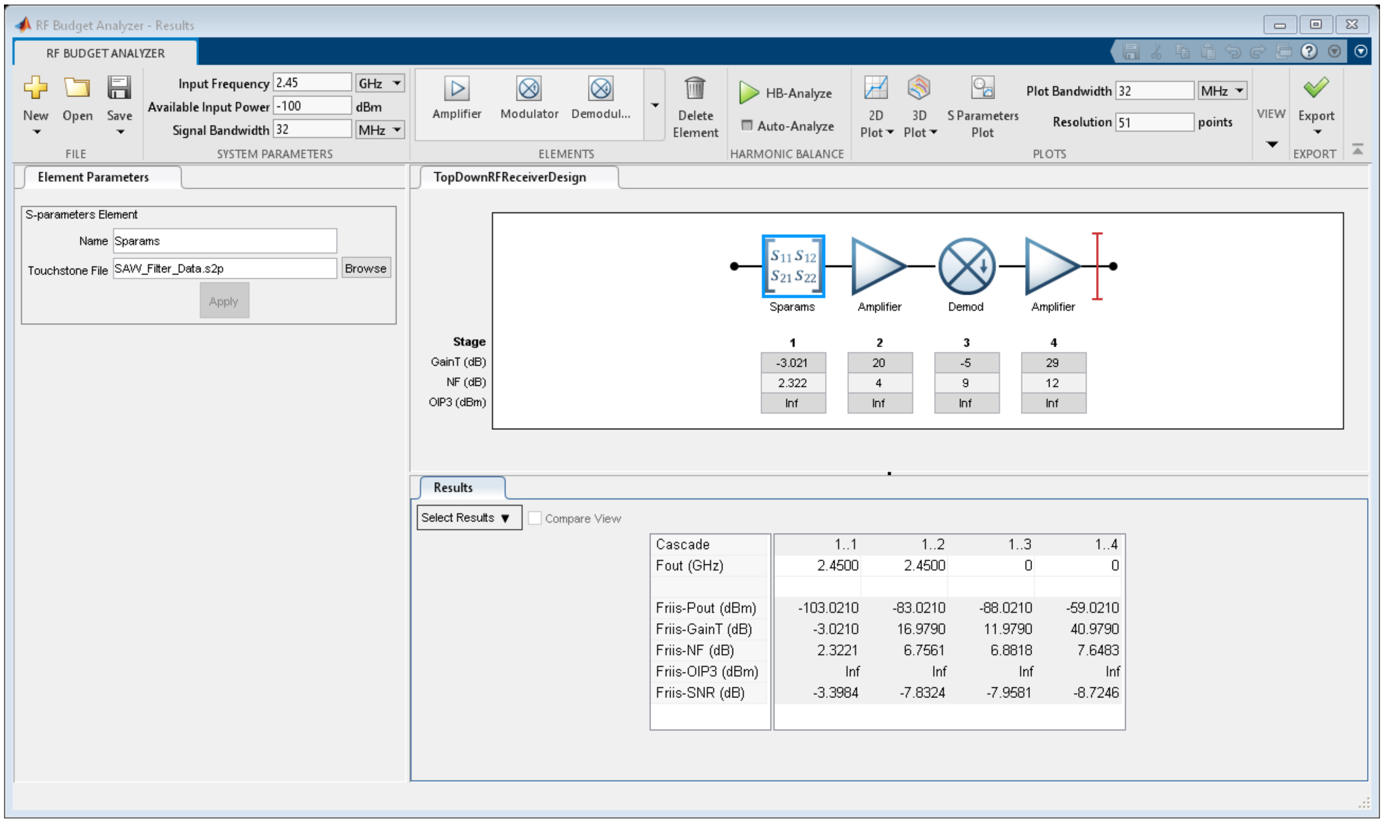

ZigBee-Like System with Circuit Envelope Receiver
Copyright 2018-2020 The MathWorks, Inc.
802.15 Transmitter
RF In
I Out
Q Out
Re
Im
Bipolar ADC
Rx Spectrum
802.15 Receiver
ChER
0.07818
2004
2.563e+04
ChER
Power Meter
-59.85
ADC Output Power (dBm)


TopDownRFReceiverDesignCircuitEnvelope/Rx Spectrum
File
Tools
View
Simulation
Help
Power (dBm)
Frequency (GHz)
-79
-80
-81
-82
-83
-84
-85
-86
-87
-88
2.448
2.4485
2.449
2.4495
2.45
2.4505
2.451
2.4515
2.452
Ready
RBW=5.859 kHz
Sample rate=4 MHz
T=0.013

You can manually build the circuit envelope model of the RF Receiver by using blocks from the Circuit Envelope library, or it can be automatically generated using the RF Budget Analyzer App.

The RF Budget Analyzer App

- Uses Friis equations to determine the noise, gain, and nonlinearity budget of an RF chain
- Allows you to explore the receiver design space and determine how to break down the specifications across the elements of the chain
- Helps you determine which element has the largest contribution to the noise and nonlinearity budget
- Can generate an RF receiver model with which you can perform multi-carrier simulation and further modify.

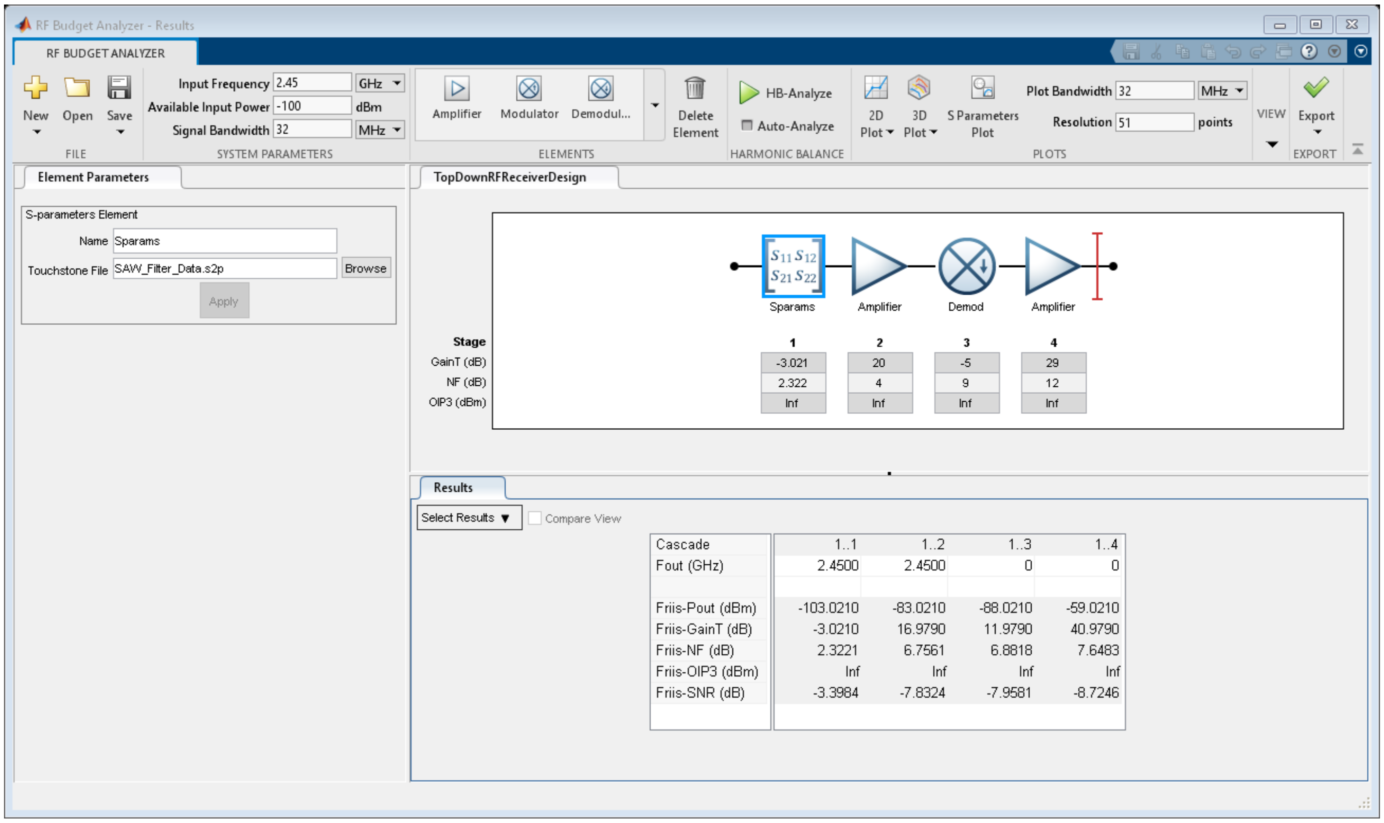


Input Frequency 2.45 GHz
Available Input Power -100 dBm
Signal Bandwidth 32 MHz
Plot Bandwidth 32 MHz
Resolution 51 points

S-parameters Element
Name Sparams
Touchstone File SAW_Filter_Data.s2p

Stage	1	2	3	4
GainT (dB)	-3.021	20	-5	29
NF (dB)	2.322	4	9	12
OIP3 (dBm)	Inf	Inf	Inf	Inf

Cascade	1..1	1..2	1..3	1..4
Fout (GHz)	2.4500	2.4500	0	0
Friis-Pout (dBm)	-103.0210	-83.0210	-88.0210	-59.0210
Friis-GainT (dB)	-3.0210	16.9790	11.9790	40.9790
Friis-NF (dB)	2.3221	6.7561	6.8818	7.6483
Friis-OIP3 (dBm)	Inf	Inf	Inf	Inf
Friis-SNR (dB)	-3.3984	-7.8324	-7.9581	-8.7246



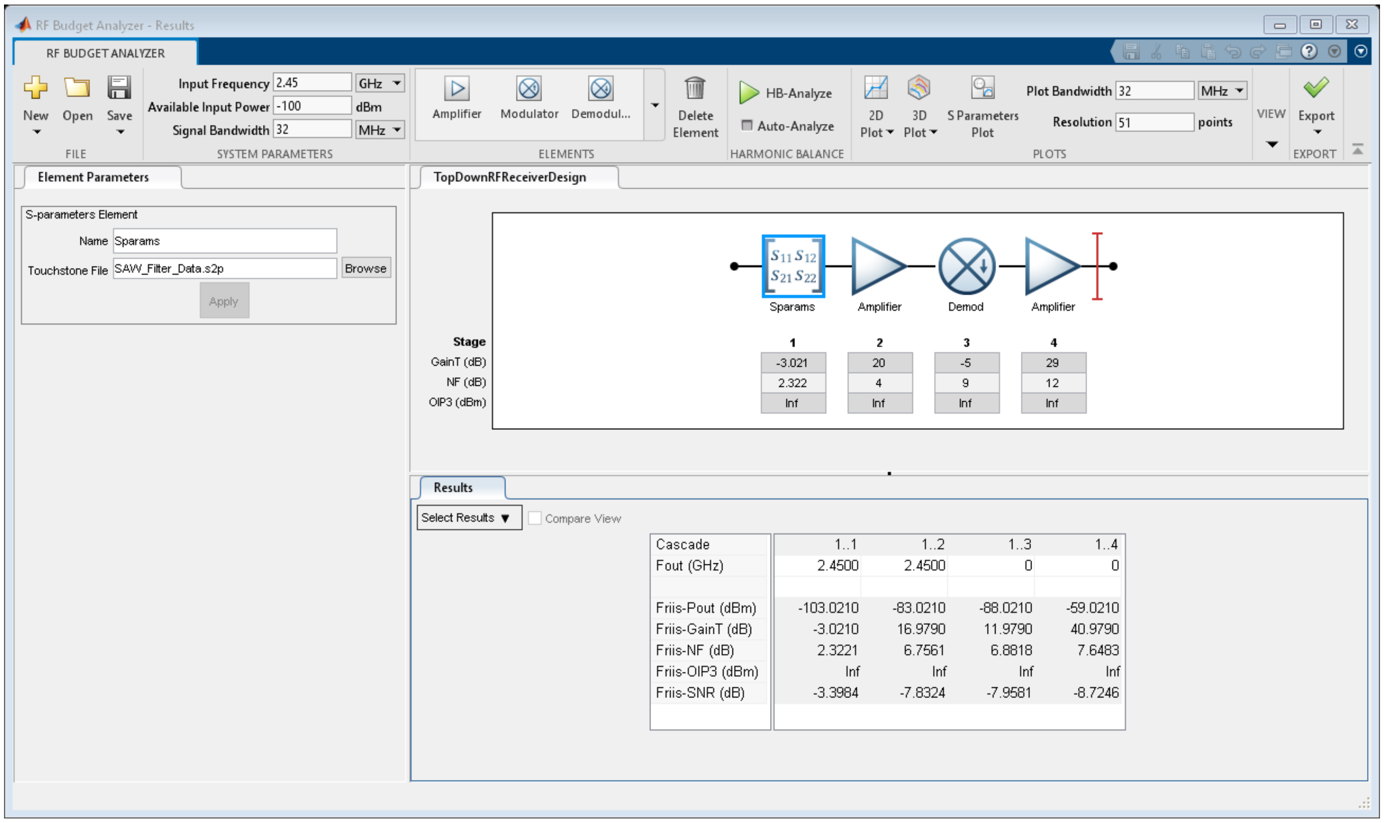
Input Frequency 2.45 GHz
Available Input Power -100 dBm
Signal Bandwidth 32 MHz
SYSTEM PARAMETERS

Plot Bandwidth 32 MHz
Resolution 51 points

S-parameters Element
Name Sparams
Touchstone File SAW_Filter_Data.s2p

Stage	1	2	3	4
GainT (dB)	-3.021	20	-5	29
NF (dB)	2.322	4	9	12
OIP3 (dBm)	Inf	Inf	Inf	Inf

Cascade	1..1	1..2	1..3	1..4
Fout (GHz)	2.4500	2.4500	0	0
Friis-Pout (dBm)	-103.0210	-83.0210	-88.0210	-59.0210
Friis-GainT (dB)	-3.0210	16.9790	11.9790	40.9790
Friis-NF (dB)	2.3221	6.7561	6.8818	7.6483
Friis-OIP3 (dBm)	Inf	Inf	Inf	Inf
Friis-SNR (dB)	-3.3984	-7.8324	-7.9581	-8.7246


Input Frequency 2.45 GHz
Available Input Power -100 dBm
Signal Bandwidth 32 MHz
SYSTEM PARAMETERS

Plot Bandwidth 32 MHz
Resolution 51 points

S-parameters Element
Name Sparams
Touchstone File SAW_Filter_Data.s2p

Stage	1	2	3	4
GainT (dB)	-3.021	20	-5	29
NF (dB)	2.322	4	9	12
OIP3 (dBm)	Inf	Inf	Inf	Inf

Cascade	1..1	1..2	1..3	1..4
Fout (GHz)	2.4500	2.4500	0	0
Friis-Pout (dBm)	-103.0210	-83.0210	-88.0210	-59.0210
Friis-GainT (dB)	-3.0210	16.9790	11.9790	40.9790
Friis-NF (dB)	2.3221	6.7561	6.8818	7.6483
Friis-OIP3 (dBm)	Inf	Inf	Inf	Inf
Friis-SNR (dB)	-3.3984	-7.8324	-7.9581	-8.7246

Add Wideband Interference, LO Leakage, and DC Offset Cancellation

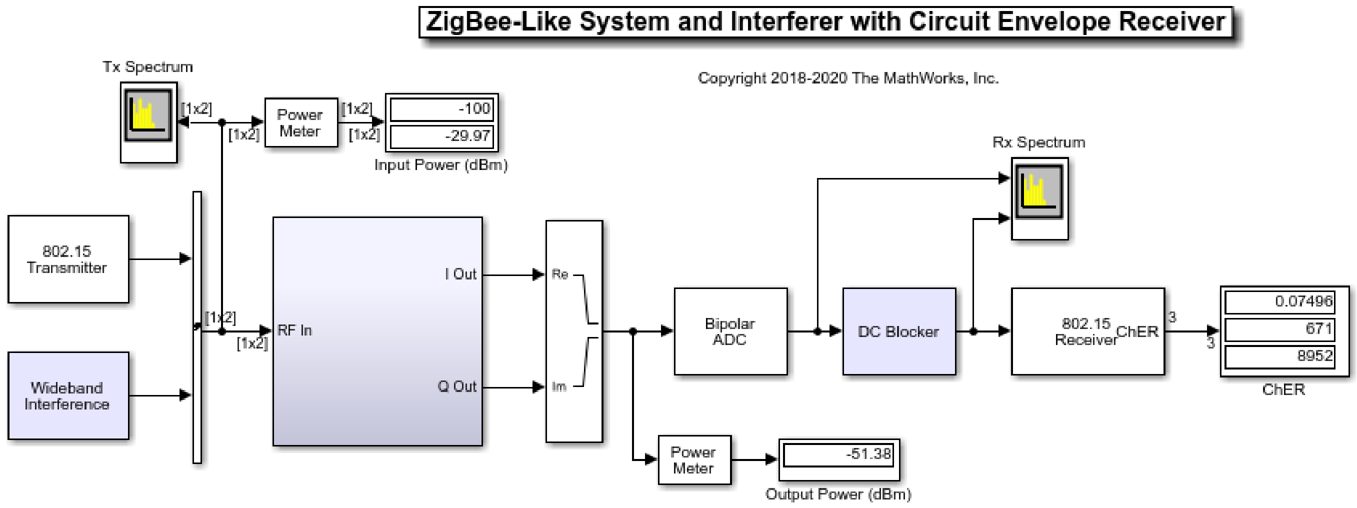
This section modifies the circuit envelope model to create this circuit envelope with interferer model. The circuit envelope with interferer model includes a wideband interfering signal and these impairments:

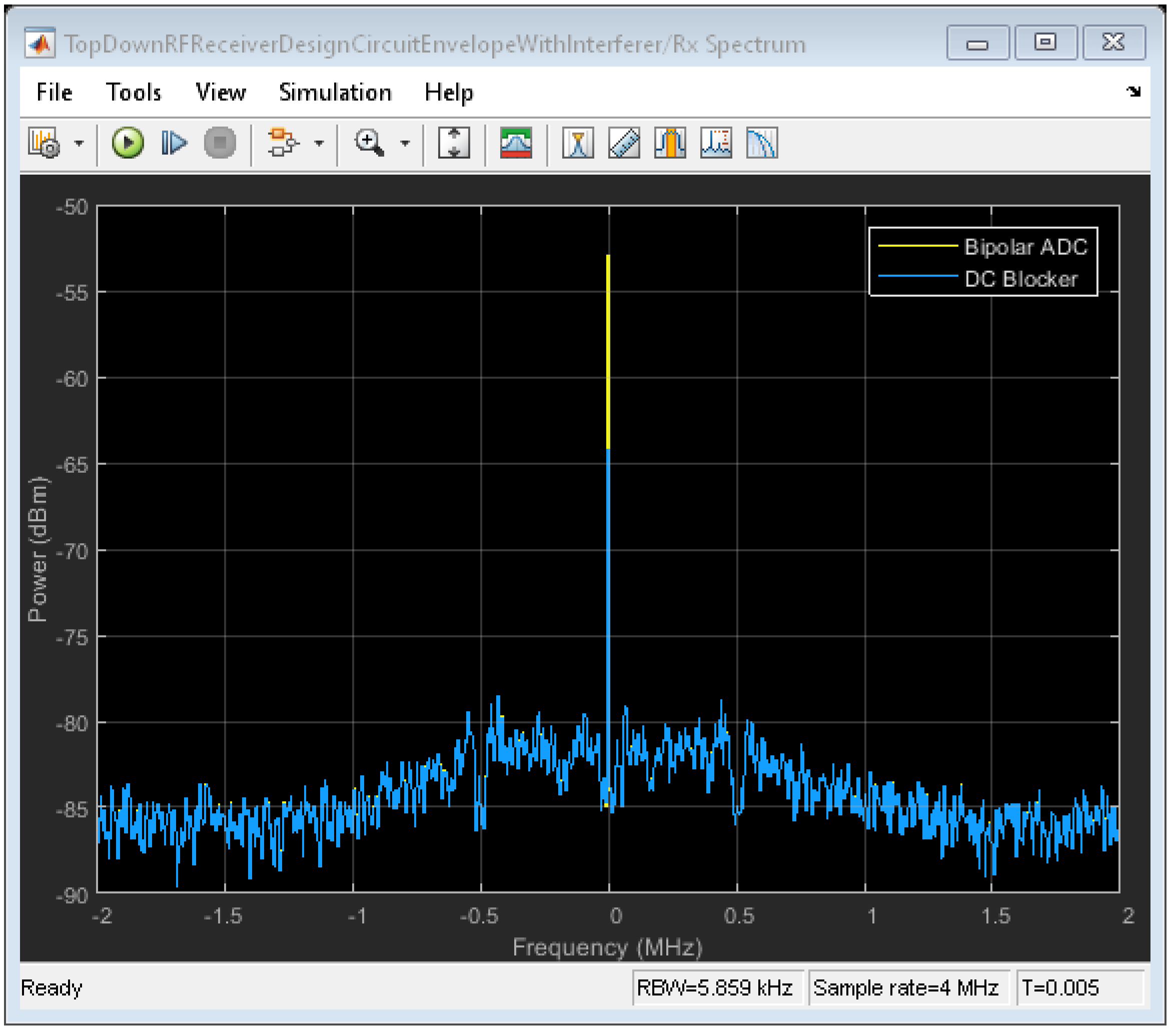
- LO-RF isolation of 90 dB in the quadrature demodulator
- OIP2 equal to 70 dBm in the quadrature demodulator
- WCDMA-like blocker of -30 dBm at 2500 MHz

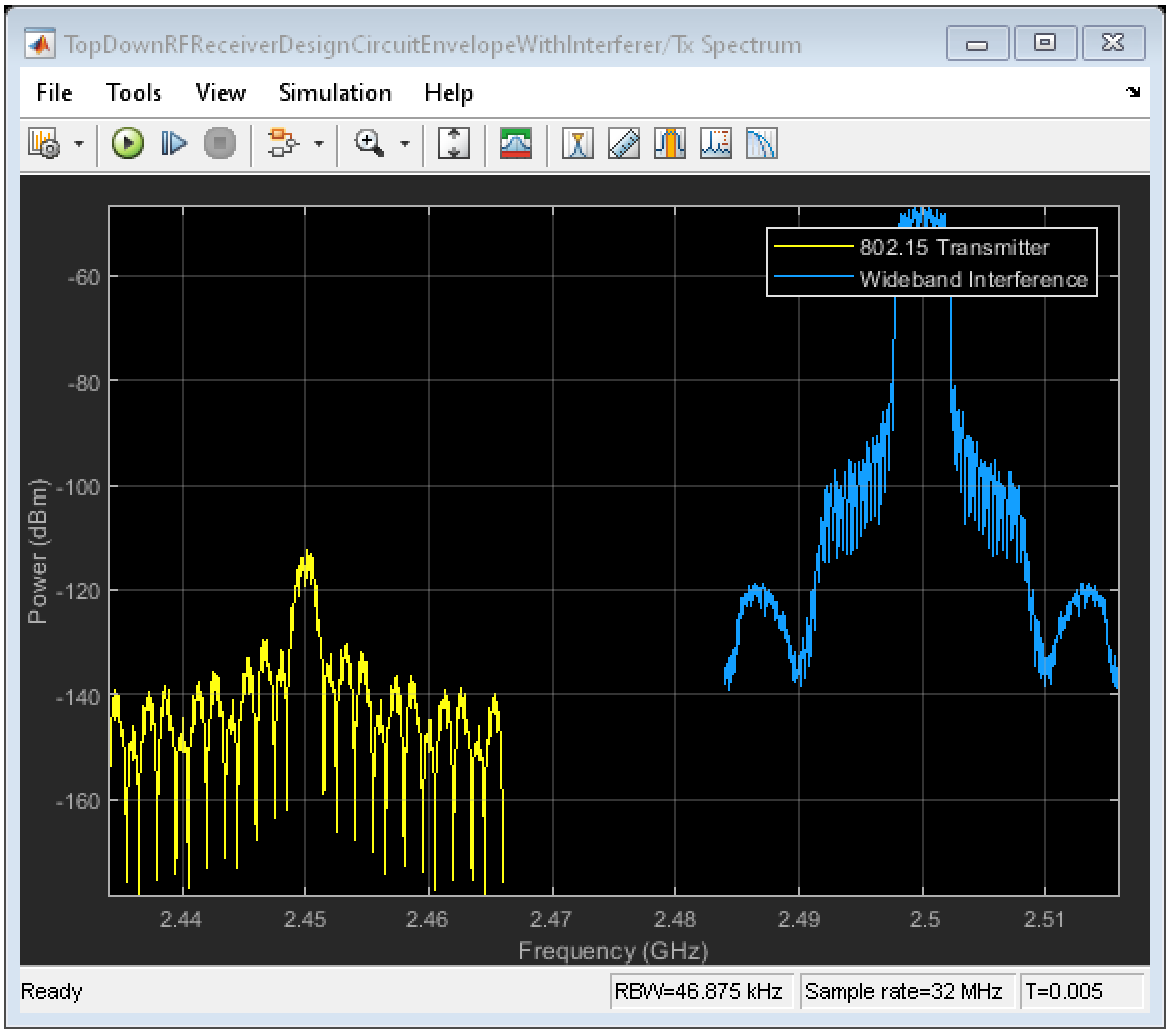
This simulation models a non-standard-compliant interfering signal that has power and spectral distribution characteristics realistic for a WCDMA signal.

The design requires a DC offset compensation algorithm to achieve the desired ChER due to the DC offset that results from the LO leakage and the nonlinearity in the demodulator caused by the high out-of-band interfering signal power. In this case you include a very selective filter, that introduces a long latency with corresponding computation delay increases in the ChER measurement block.

The spectrum centered at 0 Hz shows the DC offset compensation reducing the DC offset. As you run the model, note that the DC offset is eventually completely removed.




TopDownRFReceiverDesignCircuitEnvelopeWithInterferer/Rx Spectrum
File
Tools
View
Simulation
Help
-50
-55
-60
-65
-70
-75
-80
-85
-90
Power (dBm)
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
Frequency (MHz)
Bipolar ADC
DC Blocker
Ready
RBW=5.859 kHz
Sample rate=4 MHz
T=0.005



Conclusion

Following a top-down design methodology, RF receiver components specifications were derived. Impairment, interferer, and RF receiver subcomponent models were iteratively refined to increase fidelity and validated at each stage to confirm overall system performance goals were achieved.

See Also

General Passive Network | Mixer | S-Parameters Amplifier | VGA

More About

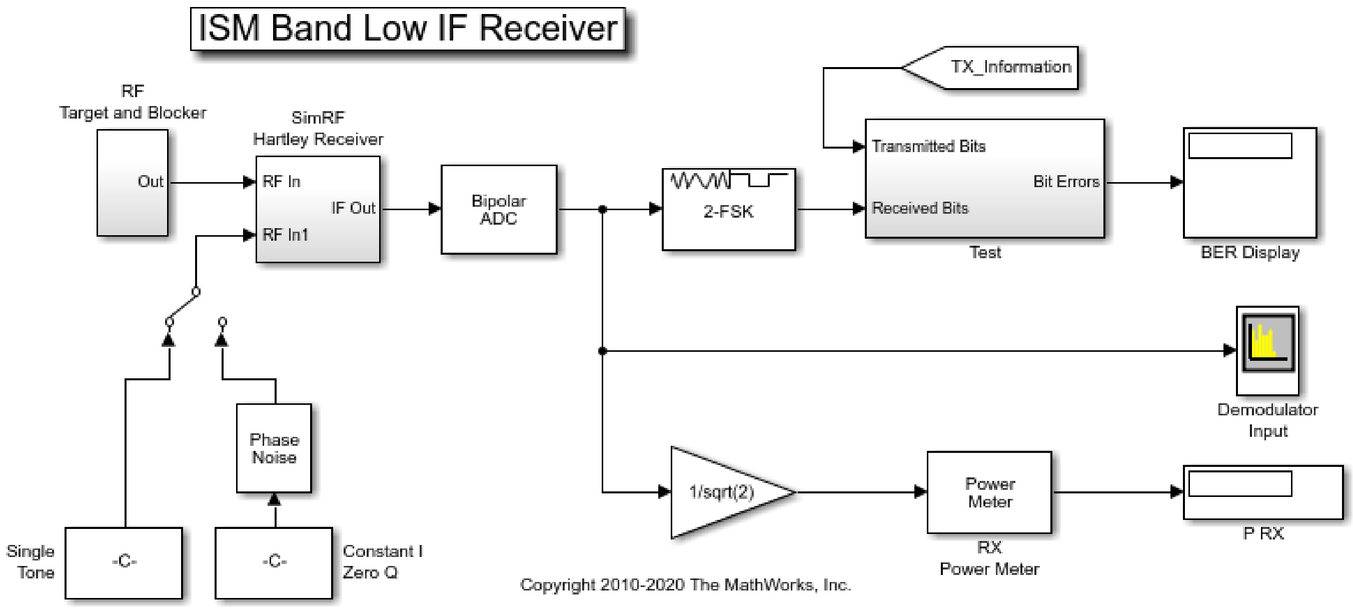
- “Architectural Design of a Low IF Receiver System” on page 9-193
- “Carrier to Interference Performance of Weaver Receiver” on page 8-40

Architectural Design of a Low IF Receiver System

This example shows how to use the RF Blockset™ Circuit Envelope library to simulate the performance of a Low IF architecture with the following RF impairments:

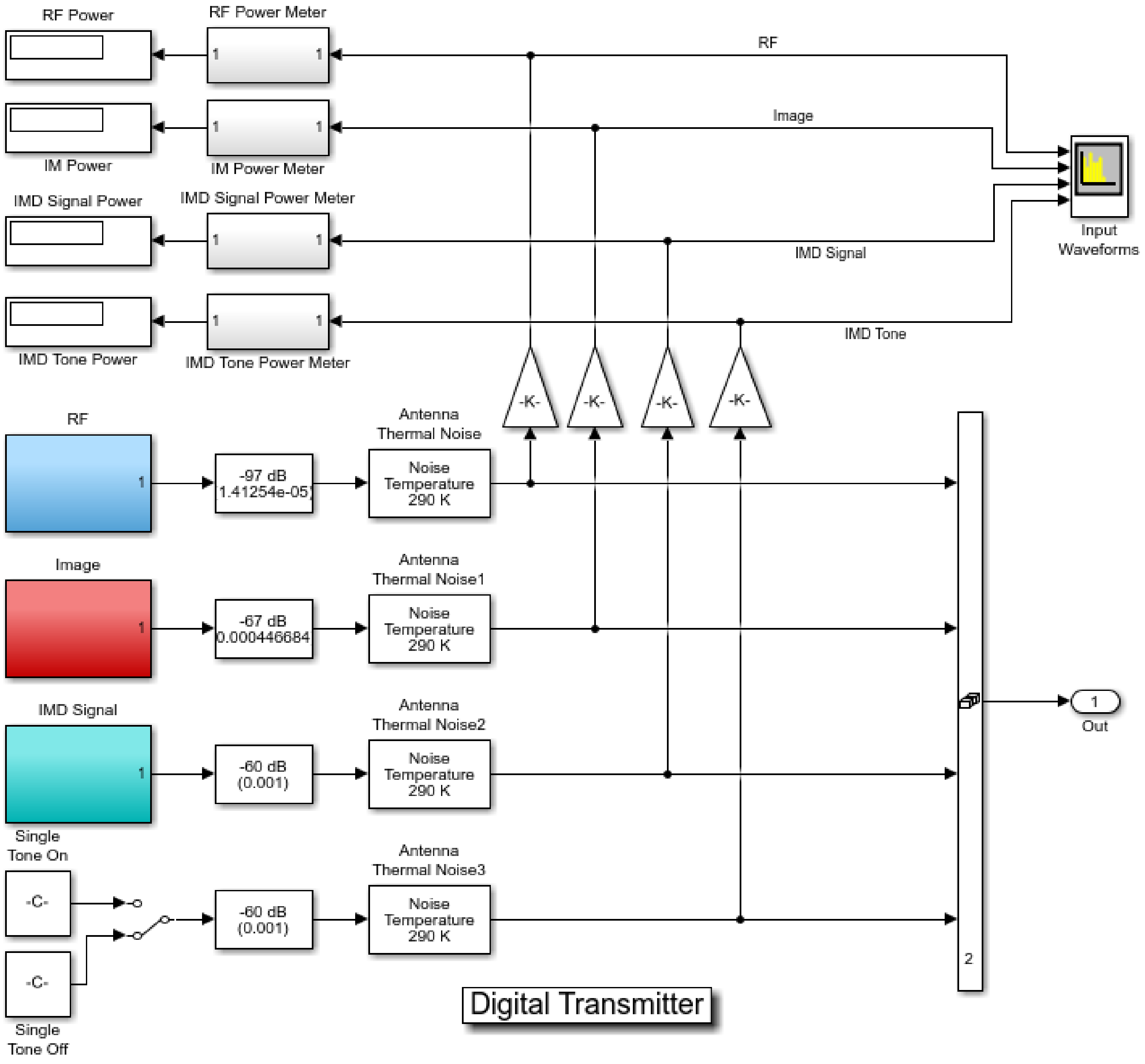
- Component noise
- Interference from blocker signals
- LO phase noise
- Analog-to-digital converter (ADC) dynamic range
- Component mismatch

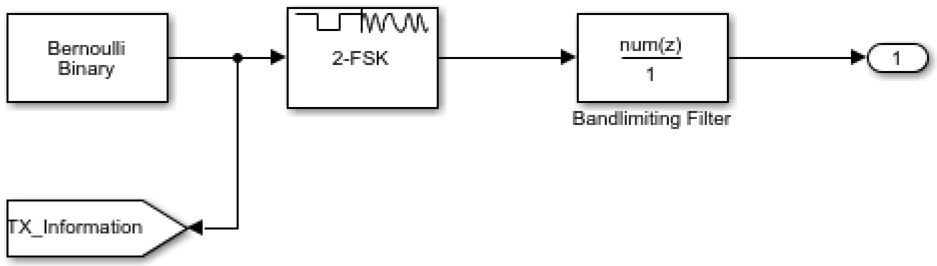
Design variables in the RF portion of the model include explicit specification of gain, noise figure, IP3, input/output impedance, LO phase offset, and LO phase noise. Carrier frequencies for waveforms entering RF Blockset subsystems are specified in the Inport blocks. Design variables for the transmitter side of the RF interface include carrier frequency, modulation scheme, signal power, and blocker power level. Baseband design variables are number of bits and full scale range of the ADC.



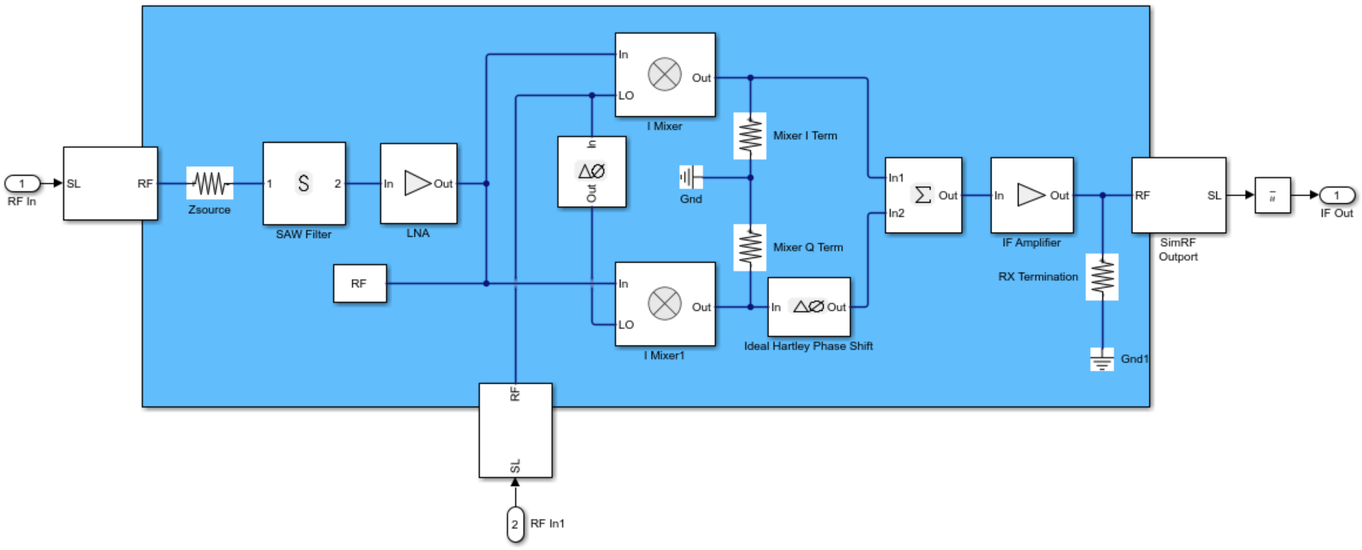
System Architecture:

This model illustrates the design and simulation of an ISM Band Receiver. Primary subsystems include a digital transmitter, an RF receiver, an ADC, a phase noise block for noisy LO modeling, and a digital receiver. The remaining blocks are used for analysis.

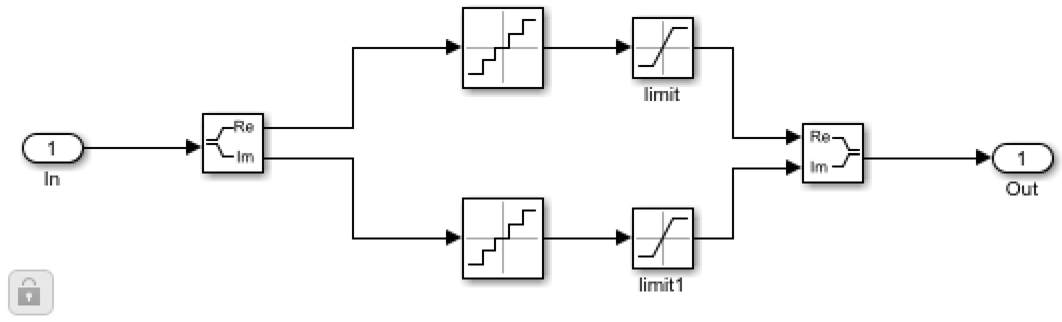

RF Power
RF Power Meter
IM Power
IM Power Meter
IMD Signal Power
IMD Signal Power Meter
IMD Tone Power
IMD Tone Power Meter
RF
Image
IMD Signal
IMD Tone
Input Waveforms
-K-
RF
Image
IMD Signal
Single Tone On
Single Tone Off
-C-
-97 dB (1.41254e-05)
-67 dB (0.000446684)
-60 dB (0.001)
Antenna Thermal Noise
Antenna Thermal Noise1
Antenna Thermal Noise2
Antenna Thermal Noise3
Noise Temperature 290 K
1 Out
2
Digital Transmitter


Bernoulli Binary
2-FSK
num(z)
1
Bandlimiting Filter
1
TX_Information

The digital transmitter consists of three FSK modulated waveforms and a high power tone. The three FSK waveform generators use a bandlimiting filter that suppresses the FSK sidebands below the expected thermal noise level. The target waveform at 2450 MHz has a 1 ohm referenced passband power level of approximately -70 dBm. Similarly defined image and intermodulation distortion (IMD) blocker waveforms have passband powers of approximately -40 dBm and -33 dBm, respectively. The IMD tone that couples with the IMD blocker to generate in-band IM3 products has a passband power of -33 dBm. Since the baseband processing defines the complex envelope waveforms, computing passband power requires the insertion of 1/sqrt(2) gain as shown in the design. An IF of 2 MHz can be inferred by inspecting the demodulator input signal spectrum, where a 2 MHz offset is specified for the display.



The Low IF receiver is comprised of a receive band SAW filter, a frequency conversion stage, an image rejection stage, and two gain stages. Resistors are used to model input and output impedances. Each nonlinear block has a noise figure specification. Power nonlinearities in the low noise amplifier (LNA), IF amplifier and mixers are specified by IP3. Image rejection is accomplished with a Hartley design, and single LO and phase shift blocks provide cosine and sine terms to mix with the I and Q branches, respectively. The summation block recombines the signals on the I branch and the phase-shifted Q branch. Image rejection quality can be controlled directly by setting a non-ideal phase offset in the Phase Shift block. To capture the RF, Image, IMD Signal and IMD Tone waveforms/spectra, choose the **Fundamental tones** to be 2450 MHz, 1 MHz and the **Harmonic Order** as 1 for the first tone and 8 for the second tone within the Configuration block. To model a thermal noise floor in the RF Blockset environment, the **Temperature** within the System Parameters section in the Configuration block is set to a noise temperature of 290.0 K.



The ADC is modeled using an a 12-bit quantizer. The quantizer takes into account the full-scale and dynamic ranges of the ADC, properly modeling its quantization noise floor.

A digital receiver demodulates the waveform for bit error rate calculation. This noncoherent FSK receiver assumes perfect timing synchronization, such that each FSK pulse is integrated over one and only one symbol.

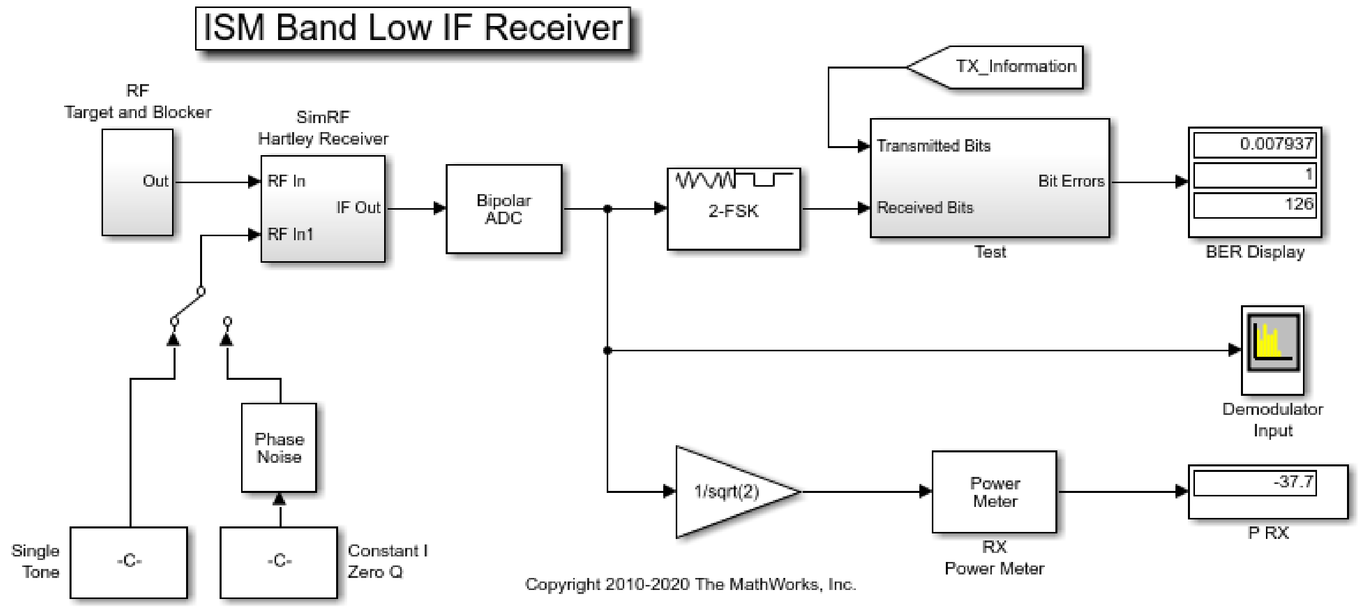
Running the Example

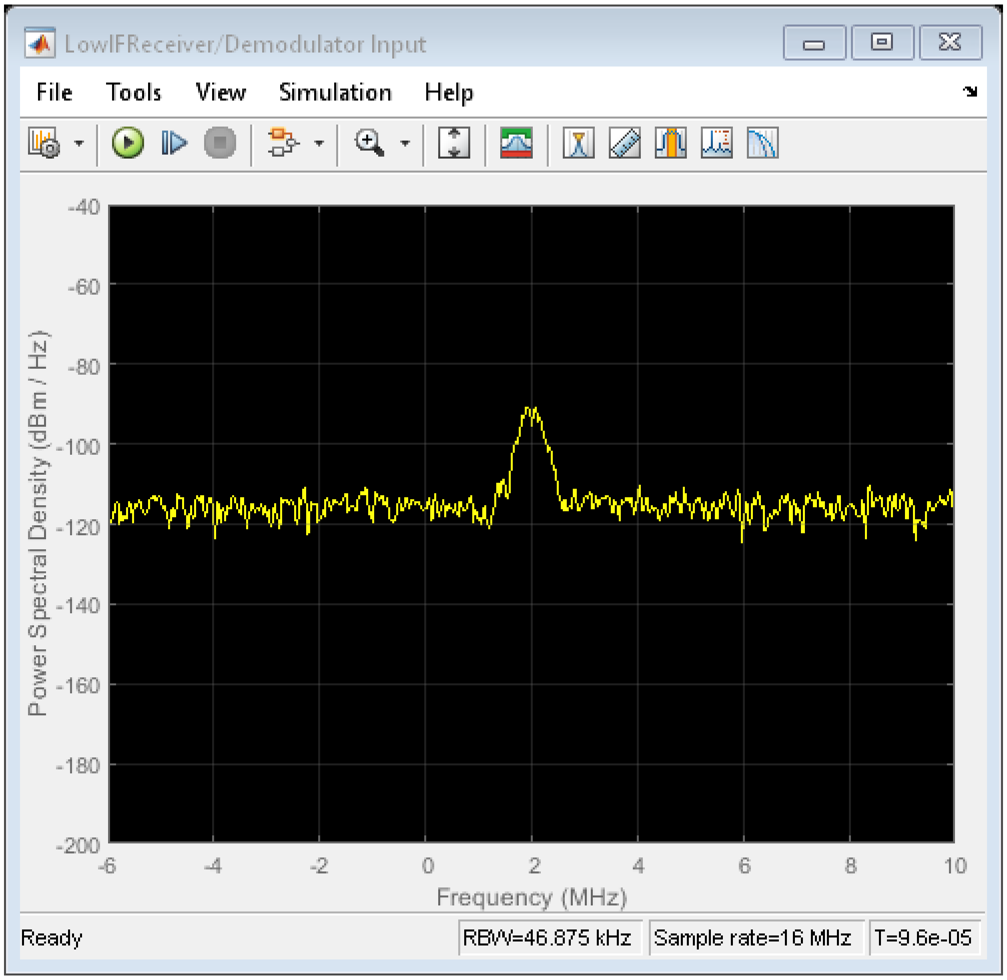
Running the example simulates a design that meets an uncoded BER spec of less than 1%. Modifications to the signals and component specifications in the receiver and ADC have a direct impact on the receiver performance. Manual switches enable you to:

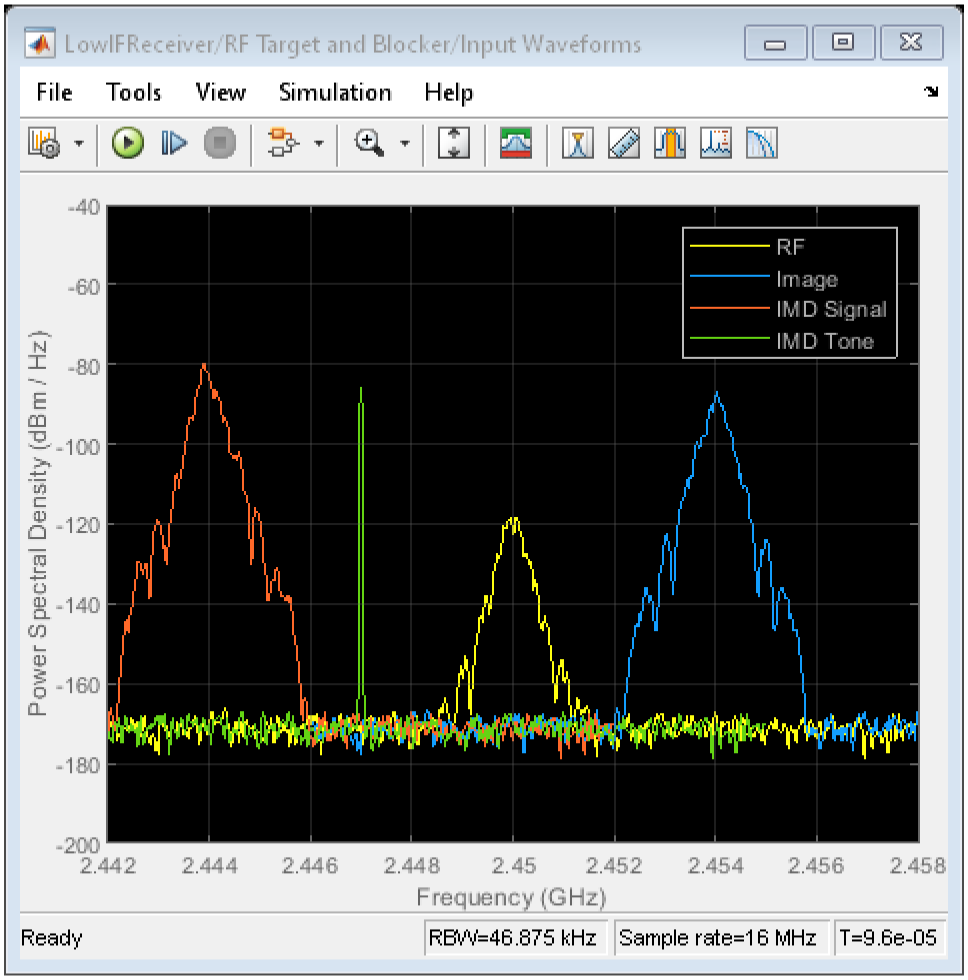
1. Select a power level for the IMD blocker tone of -33 dBm or -45 dBm
2. Select an ideal or noisy LO.

Other possible changes to the design include:

- Image rejection ratio (IRR) of the Hartley design. The IRR of the present design (dPhi=0.01 degrees) is -40 dB. For more information on calculating IRR, see the example `“Measuring Image Rejection Ratio in Receivers” on page 9-68` Measuring Image Rejection Ratio in Receivers>.
- Modulation schemes
- Baseband filtering options
- Signal power levels
- Signal carrier frequencies
- Noise figures
- Non-linear gain parameters
- Interstage matching
- ADC bit length and full scale range


ISM Band Low IF Receiver
RF
Target and Blocker
Out
SimRF
Hartley Receiver
RF In
IF Out
RF In1
Bipolar
ADC
2-FSK
TX_Information
Transmitted Bits
Bit Errors
Received Bits
Test
0.007937
1
126
BER Display
Demodulator
Input
Phase
Noise
Single
Tone
-C-
-C-
Constant I
Zero Q
1/sqrt(2)
Power
Meter
RX
Power Meter
-37.7
P RX
Copyright 2010-2020 The MathWorks, Inc.


LowIFReceiver/Demodulator Input
File
Tools
View
Simulation
Help
Power Spectral Density (dBm / Hz)
-40
-60
-80
-100
-120
-140
-160
-180
-200
-6
-4
-2
0
2
4
6
8
10
Frequency (MHz)
Ready
RBW=46.875 kHz
Sample rate=16 MHz
T=9.6e-05



See Also

Amplifier | Mixer | Outport

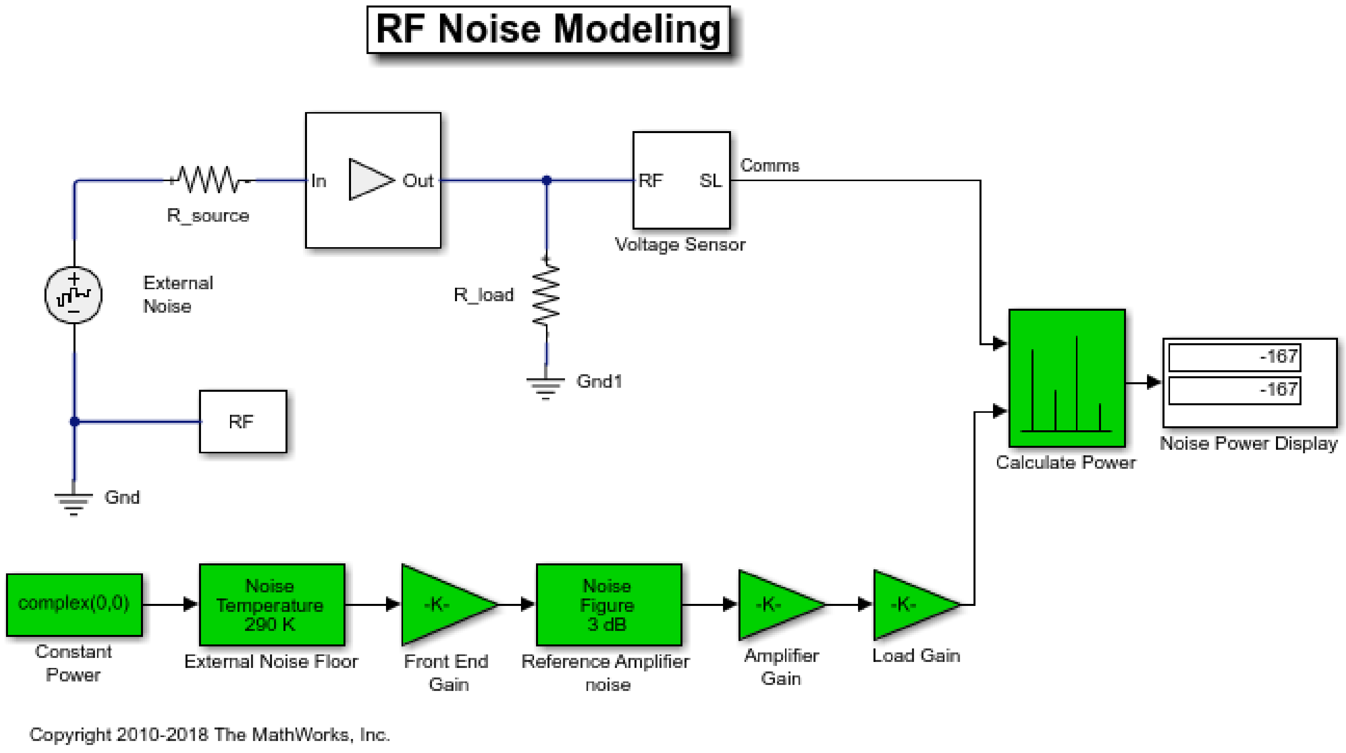
More About

- “Top-Down Design of an RF Receiver” on page 9-179
- “Carrier to Interference Performance of Weaver Receiver” on page 8-40

RF Noise Modeling

This example shows how to use the RF Blockset™ Circuit Envelope library to simulate noise and calculate noise power. Results are compared against theoretical calculations and a Communications Toolbox™ reference model.

The model is shown below.



System Architecture

The RF system, shown in white, consists of:

- A `Configuration` block, which sets global simulation parameters for the RF Blockset system. With the **Simulate Noise** option checked, noise is included in the simulation.
- An `External Noise` source with a power spectral density of $4kT_sR$ applied at the input. In this equation, k is the Boltzmann constant, T_s is the temperature of the source, and R is the noise reference impedance. The calculated noise level of -174 dBm/Hz is used in this example. The `External Noise` source is an explicit signal.
- An `Amplifier` block with a specified power gain and noise figure.
- A `Voltage Sensor` (i.e. `Outport`) block, with the **Source type** parameter set to **Voltage**.
- Source and load resistors.

The Communications Toolbox reference system, shown in green, consists of:

- `Gain` blocks that model amplifier gain and loading effects.
- Two `Receiver Thermal Noise` blocks that model the external noise and the amplifier noise, respectively.

The `Calculate Power` block computes RMS noise power. Note that the Communications Toolbox signal is referenced to 1 ohm, while RF Blockset power is computed for the actual load R_{load} .

The example model defines variables for block parameters using a callback function. To access model callbacks, select **MODELING** > **Model Settings** > **Model Properties** and click the Callbacks tab in the Model Properties window.

Running the Example

1. Type `open_system('RFNoiseExample')` at the Command Window prompt.
2. Select **Simulation** > **Run**.

The `Noise Power Display` block verifies that the RF Blockset and Communications Toolbox noise models are equivalent.

Computing RF System Noise

To enable noise in the RF Blockset circuit envelope environment:

- In the `Configuration` block dialog, select **Simulate noise**.
- Specify a **Temperature**. RF Blockset uses this value to calculate the equivalent noise temperature inside the amplifier.
- Specify the **Noise figure (dB)** parameter of any amplifiers or mixers in the system.

In the example, for a specified LNA gain of 4 dB and noise figure of 3 dB, the output noise is calculated using the following equations:

$$G_1 = 2.5119 \text{ (4 dB)}$$

$$F_1 = 1.9953 \text{ (3 dB)}$$

The next equation converts the noise factor to an equivalent noise temperature. T is the **Temperature** parameter of the RF Blockset `Configuration` block.

$$T_e = (F_1 - 1) * T = 288.63$$

The final equation calculates the output noise power. T_s is the temperature of the SimRF™ `External Noise` block and the Communications Toolbox `External Noise Floor` block.

$$N_{out,sys} = 10\log_{10}(k(T_s + T_e)G_1) + 30 = -166.97 \text{ dBm/Hz}$$

The available noise power is the power that can be supplied by a resistive source when it is feeding a noiseless resistive load equal to the source resistance. The green `External Noise Floor` block generates an available power referenced to 50 ohms.

The `Front End Gain` block models the voltage divider due to the source resistance and the input impedance of the amplifier.

The green `Reference Amplifier Noise` and `Amplifier Gain` blocks model the noise added by the amplifier and the amplifier gain, respectively.

The output of the Communications Toolbox `Amplifier Gain` block is equal to the voltage across the RF Blockset `R_load` block.

See Also

Noise | Noise Figure Testbench

More About

- “Noise in RF Systems” on page 3-8
- “Model LO Phase Noise” on page 2-8

Impact of Thermal Noise on Communication System Performance

This example shows how to use the RF Blockset Circuit Envelope library to model the thermal noise of a super-heterodyne RF receiver and measure its effects on the bit error rate (BER) of a communications system. The RF receiver model does not include impedance mismatches or nonlinearities. A Communications Toolbox reference model with parameters computed using Friis equations is used to verify the results.

RF Receiver System Architecture

The Modulator and Channel subsystems consist of blocks that model:

- A QPSK-modulated waveform of random bits
- A raised cosine pulse-shaping filter for spectral limiting
- Free-space path loss

The RF receiver subsystem, shown in the light purple area, consists of:

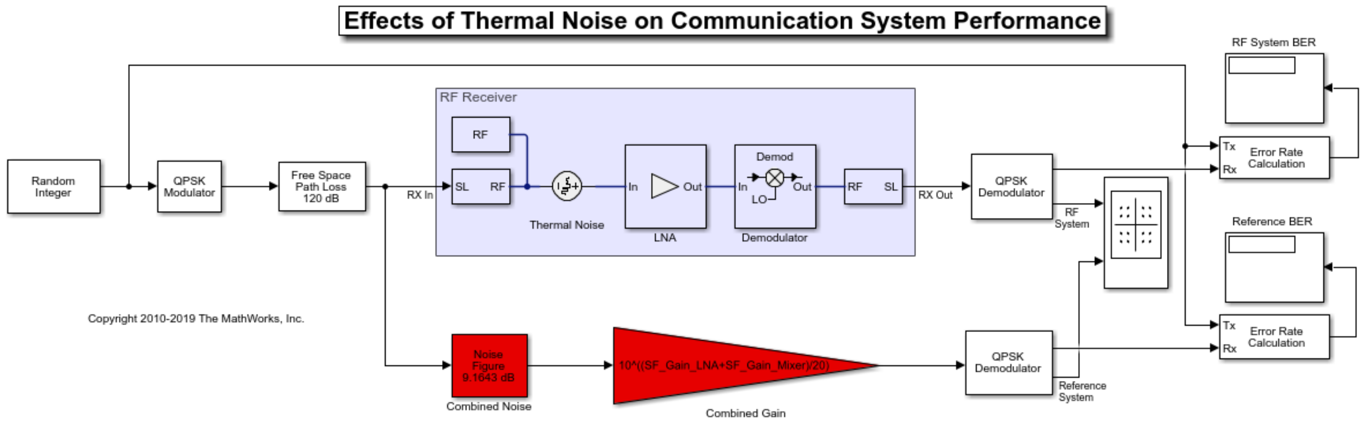
- An Inport block, which assigns the complex input waveform to the specified RF carrier. In this case the variable SF_RF is set to 2.1 GHz. To ensure that the RF Blockset model has the same available power referred to a 50 Ohm impedance as the reference model, the Source type parameter is set to Power.
- A cascaded RF amplifier and RF demodulator with specified noise figure and gain. The amplifier and the mixer do not specify any nonlinearity, so the harmonic order of the simulation specified in the configuration block is set to 1.
- An Outport block, with the Source type parameter set to Power and center frequency equal to the intermediate frequency given by the absolute difference between the RF center frequency specified in the Inport and Local Oscillator specified in the demodulator. In this case, the intermediate frequency is equal to the variable SF_IF = 500 MHz.
- A thermal white noise source at the input of the chain to model the equivalent noise floor introduced by an ideally matched antenna (50 Ohm) at 290 K.
- An image reject filter is added into the Demodulator block using a mask checkbox to ensure that only signals centered around the 2.1 GHz carrier enter the demodulator. This filter also prevents the down-conversion of thermal noise centered around the image frequency of 2.6 GHz (image frequency is SL_RF+SL_IF) into the intermediate frequency. If the image rejection filter is removed, the noise contribution on the output signal increases above the estimation provided by Friis equations and the BER will deteriorate.
- Note that all blocks in the RF receiver are ideally matched to 50 Ohm. To understand the effects of impedance mismatches on noise simulation see the example "RF Noise Modeling" on page 9-199.

The reference system, shown in red, consists of:

- A Receiver Thermal Noise block that adds noise to the signal according to the value calculated by the Friis Equation. You can find the calculation in the model's pre-load callback function. This block also includes the equivalent thermal noise introduced by the antenna.
- A Gain block that models the combined gain of the RF receiver.
- Baseband filters and demodulators process the received signal.

Circuit Envelope Simulation of the RF Receiver

```
open_system('RFReceiverImpactExample')
```



Select Simulation > Run.

Error Rate Calculation blocks compute the BER for the system and reference. To observe the BER as it approaches steady state, increase the total simulation time. For this example, the steady-state bit error rate is approximately 1e-4.

Computing RF Receiver Noise Figure

To model noise in the RF Blockset circuit envelope environment:

- In the Configuration block dialog, select Simulate noise.
- Specify the Noise figure (dB) parameter of RF Amplifier and RF Mixer blocks in your system. The following specifications for the RF receiver in this example produce a combined noise figure of 9.16 dB (as per the Friis Equation): LNA gain of 20 dB, LNA noise figure of 9 dB, and RF mixer noise figure of 15 dB.

$$G_1 = 100\ (20\ \text{dB})$$

$$F_1 = 7.94\ (6\ \text{dB})$$

$$F_2 = 31.62\ (15\ \text{dB})$$

$$F_{sys} = F_1 + \frac{F_2 - 1}{G_1} = 8.25$$

$$NF_{sys} = 10\log_{10} F_{sys} = 9.16\ \text{dB}$$

```
bdclose('RFReceiverImpactExample')
```

See Also

Noise | Amplifier | Demodulator

Related Topics

“RF Noise Modeling” on page 9-199 | “Explicitly Simulate Resistor Thermal Noise” on page 8-5