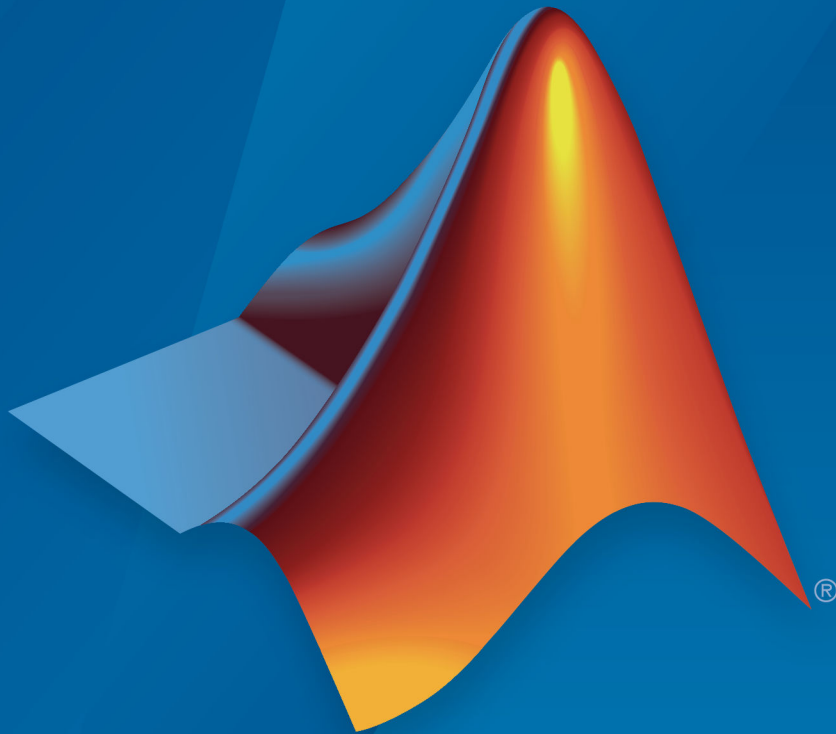


DSP System Toolbox™

Getting Started Guide



MATLAB® & SIMULINK®

R2018b

 MathWorks®

How to Contact MathWorks



Latest news: www.mathworks.com
Sales and services: www.mathworks.com/sales_and_services
User community: www.mathworks.com/matlabcentral
Technical support: www.mathworks.com/support/contact_us



Phone: 508-647-7000



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

DSP System Toolbox™ Getting Started Guide

© COPYRIGHT 2011–2018 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

Revision History

April 2011	First printing	Revised for Version 8.0 (R2011a)
September 2011	Online only	Revised for Version 8.1 (R2011b)
March 2012	Online only	Revised for Version 8.2 (R2012a)
September 2012	Online only	Revised for Version 8.3 (R2012b)
March 2013	Online only	Revised for Version 8.4 (R2013a)
September 2013	Online only	Revised for Version 8.5 (R2013b)
March 2014	Online only	Revised for Version 8.6 (R2014a)
October 2014	Online only	Revised for Version 8.7 (R2014b)
March 2015	Online only	Revised for Version 9.0 (R2015a)
September 2015	Online only	Revised for Version 9.1 (R2015b)
March 2016	Online only	Revised for Version 9.2 (R2016a)
September 2016	Online only	Revised for Version 9.3 (R2016b)
March 2017	Online only	Revised for Version 9.4 (R2017a)
September 2017	Online only	Revised for Version 9.5 (R2017b)
March 2018	Online only	Revised for Version 9.6 (R2018a)
September 2018	Online only	Revised for Version 9.7 (R2018b)

1 Introduction

DSP System Toolbox Product Description	1-2
Key Features	1-2
ARM Cortex-M and ARM Cortex-A Optimization	1-3
Configure the Simulink Environment for Signal Processing Models	1-5
About DSP Simulink Model Templates	1-5
Create Model Using the DSP System Toolbox Simulink Model Template	1-5
DSP Simulink Model Templates	1-6
See Also	1-12

2 Design Filters in Simulink

Design and Implement a Filter	2-2
Design a Digital Filter in Simulink	2-2
Add a Digital Filter to Your Model	2-6
Adaptive Filters	2-10
Design an Adaptive Filter in Simulink	2-10
Add an Adaptive Filter to Your Model	2-13
View the Coefficients of Your Adaptive Filter	2-18
Summary	2-22

Introduction

- “DSP System Toolbox Product Description” on page 1-2
- “ARM Cortex-M and ARM Cortex-A Optimization” on page 1-3
- “Configure the Simulink Environment for Signal Processing Models” on page 1-5

DSP System Toolbox Product Description

Design and simulate streaming signal processing systems

DSP System Toolbox provides algorithms, apps, and scopes for designing, simulating, and analyzing signal processing systems in MATLAB® and Simulink®. You can model real-time DSP systems for communications, radar, audio, medical devices, IoT, and other applications.

With DSP System Toolbox you can design and analyze FIR, IIR, multirate, multistage, and adaptive filters. You can stream signals from variables, data files, and network devices for system development and verification. The Time Scope, Spectrum Analyzer, and Logic Analyzer let you dynamically visualize and measure streaming signals. For desktop prototyping and deployment to embedded processors, including ARM® Cortex® architectures, the toolbox supports C/C++ code generation. It also supports bit-accurate fixed-point modeling and HDL code generation from filters, FFT, IFFT, and other algorithms.

Algorithms are available as MATLAB functions, System objects, and Simulink blocks.

Key Features

- Signal processing and linear algebra blocks for Simulink
- Streaming signal processing in MATLAB
- Single-rate, multirate, FIR, IIR, and adaptive filter design
- Time Scope, Spectrum Analyzer, and Logic Analyzer for visualizing and measuring streaming signals
- Fixed-point modeling and simulation of signal processing algorithms
- Support for C and C++ code generation
- Support for HDL code generation

ARM Cortex-M and ARM Cortex-A Optimization

The DSP System Toolbox supports optimized C code generation for popular algorithms like FIR filtering and FFT on ARM Cortex-M and ARM Cortex-A processors.

You can generate C code that can be linked with the CMSIS library or calls the Ne10 library functions and compiled to provide optimized executables to run on ARM Cortex-M or ARM Cortex-A processors.

To use the DSP System Toolbox support packages for ARM Cortex-M and ARM Cortex-A processors, you must have the following products in addition to the DSP System Toolbox: Simulink, Simulink Coder™, Embedded Coder®, and MATLAB Coder.

To obtain more information and download the DSP System Toolbox support packages for the ARM Cortex processors, see <https://www.mathworks.com/hardware-support.html>.

See Also

Related Examples

- “Verify FIR Filter on ARM Cortex-A Processor in MATLAB” (DSP System Toolbox Support Package for ARM Cortex-A Processors)
- “Real-Time ECG QRS Detection on ARM Cortex-A Processor” (DSP System Toolbox Support Package for ARM Cortex-A Processors)
- “Verify FIR Filter on ARM Cortex-M Processor in MATLAB” (DSP System Toolbox Support Package for ARM Cortex-M Processors)
- “Real-Time ECG QRS Detection on ARM Cortex-M Processor” (DSP System Toolbox Support Package for ARM Cortex-M Processors)

More About

- “Installation and Setup” (DSP System Toolbox Support Package for ARM Cortex-A Processors)
- “Installation and Setup” (DSP System Toolbox Support Package for ARM Cortex-M Processors)
- “Deployment” (DSP System Toolbox Support Package for ARM Cortex-M Processors)

- “Deployment” (DSP System Toolbox Support Package for ARM Cortex-A Processors)

External Websites

- Port the Generated ARM Cortex-M CRL Code from MATLAB to KEIL μ Vision IDE
- Port the Generated ARM Cortex-M CRL Code from MATLAB to IAR Embedded Workbench

Configure the Simulink Environment for Signal Processing Models

In this section...

“About DSP Simulink Model Templates” on page 1-5

“Create Model Using the DSP System Toolbox Simulink Model Template” on page 1-5

“DSP Simulink Model Templates” on page 1-6

“See Also” on page 1-12

About DSP Simulink Model Templates

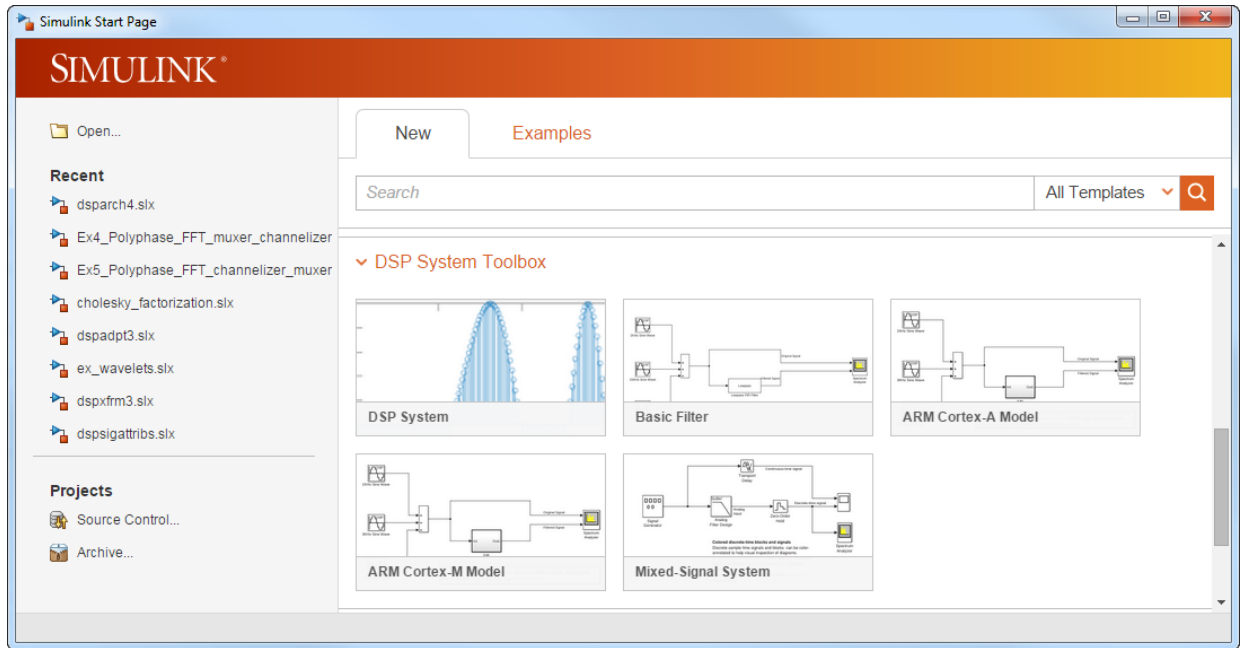
The DSP Simulink model templates let you automatically configure the Simulink environment with the recommended settings for digital signal processing modeling. DSP Simulink model templates enable reuse of settings, including configuration parameters. You can create models from templates that use best practices and take advantage of previous solutions to common problems. Instead of the default canvas of a new model, select a template model to help you get started.

For more information on Simulink model templates, see “Create a Model” (Simulink).

Create Model Using the DSP System Toolbox Simulink Model Template

To create a new blank model and open the library browser:

- 1 On the MATLAB **Home** tab, click **Simulink**.
- 2 Click on **DSP System** to create an empty model with settings suitable for use with DSP System Toolbox. The new model opens. To access the library browser, click the **Library Browser** button on the model toolbar.



The new model using the template settings and contents appears in the Simulink Editor. The model is only in memory until you save it.

DSP Simulink Model Templates

When you create a model by choosing one of the DSP Simulink model templates, the model is configured to use the settings recommended for DSP System Toolbox. Some of these settings are:

Configuration Parameter	Setting
SingleTaskRateTransMsg	error
multiTaskRateTransMsg	error
Solver	fixedstepdiscrete
EnableMultiTasking	Off
StartTime	0.0
StopTime	inf

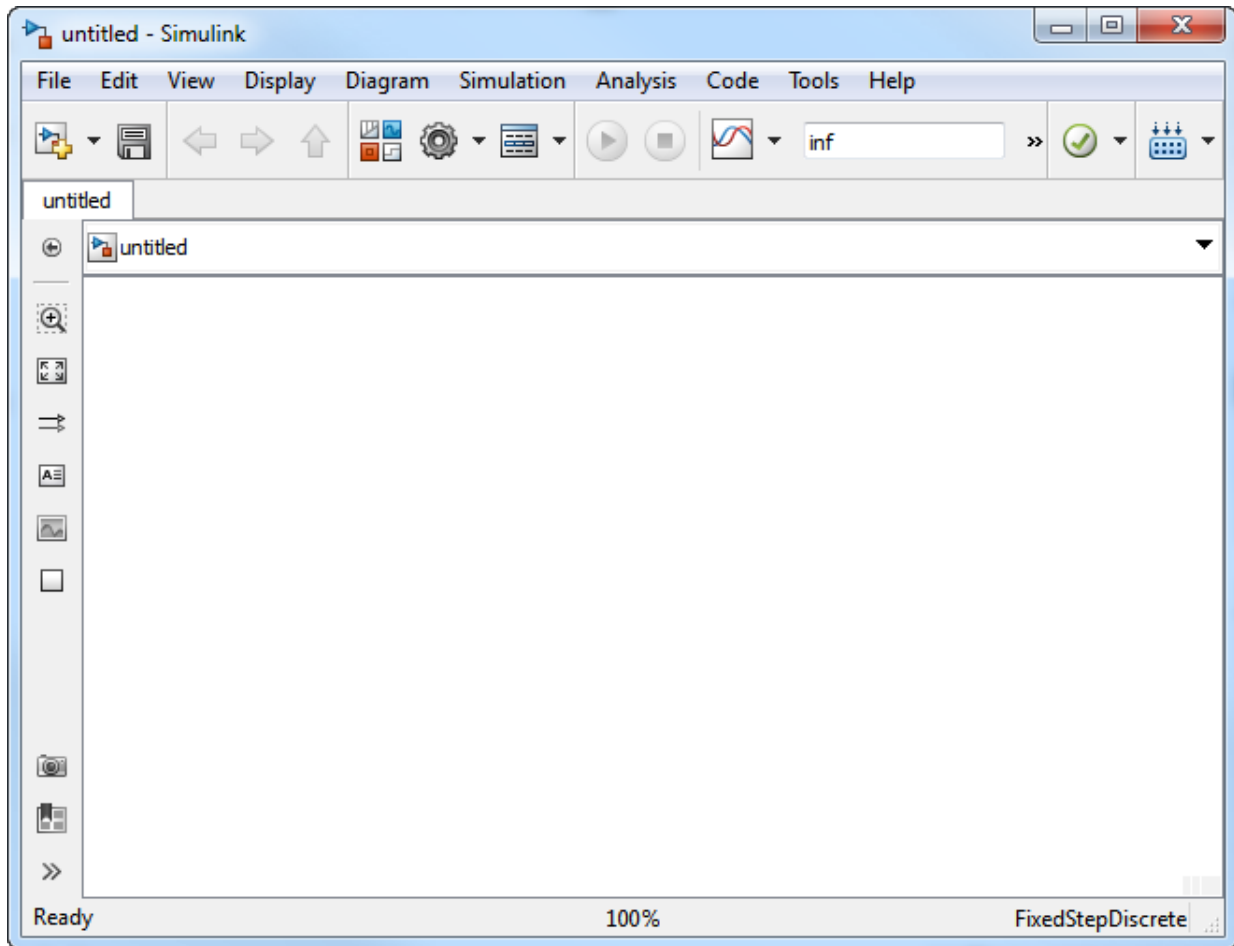
Configuration Parameter	Setting
FixedStep	auto
SaveTime	off
SaveOutput	off
AlgebraicLoopMsg	error
SignalLogging	off
FrameProcessingCompatibilityMsg	error

The DSP Simulink model templates are:

- “DSP System Template” on page 1-7
- “Basic Filter Template” on page 1-8
- “ARM Cortex-A Model Template” on page 1-9
- “ARM Cortex-M Model Template” on page 1-10
- “Mixed-Signal System Template” on page 1-11

DSP System Template

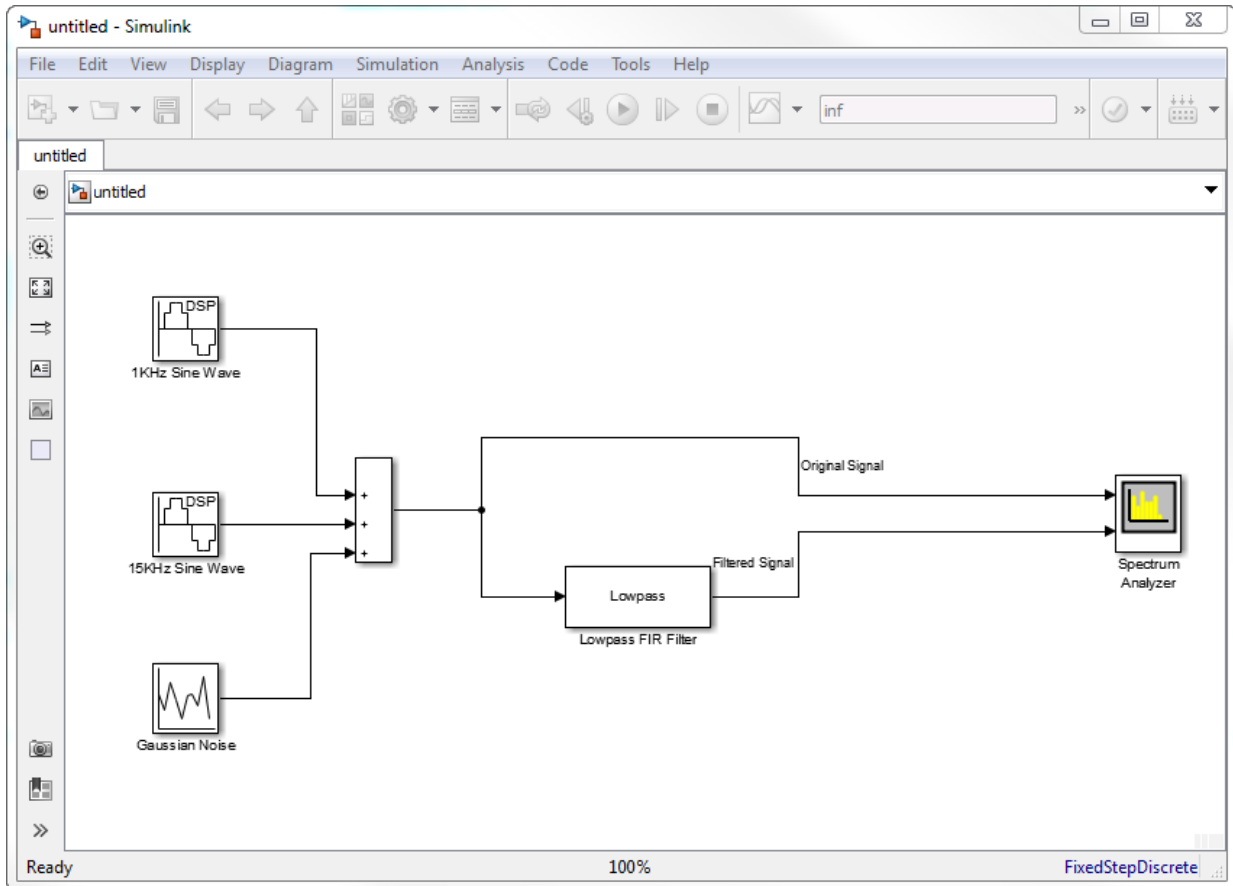
Click on **DSP System** to create a blank model configured with settings recommended for DSP System Toolbox.



Basic Filter Template

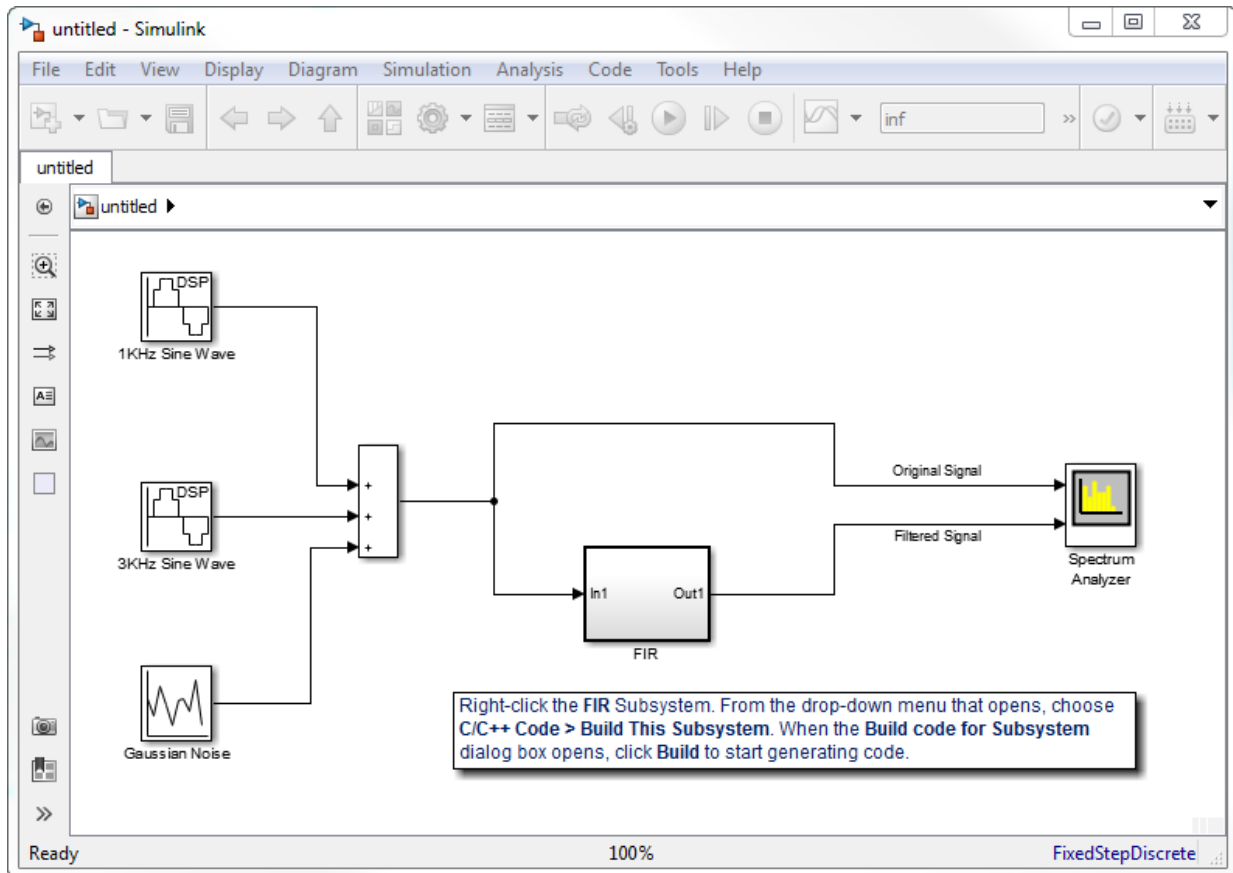
Click on **Basic Filter** to create a basic filtering model configured with settings recommended for DSP System Toolbox.

This model implements a low pass filter and enables you to compare the filtered signal with the original signal. The model acts as a starting point for modeling filtering algorithms in Simulink using DSP System Toolbox.



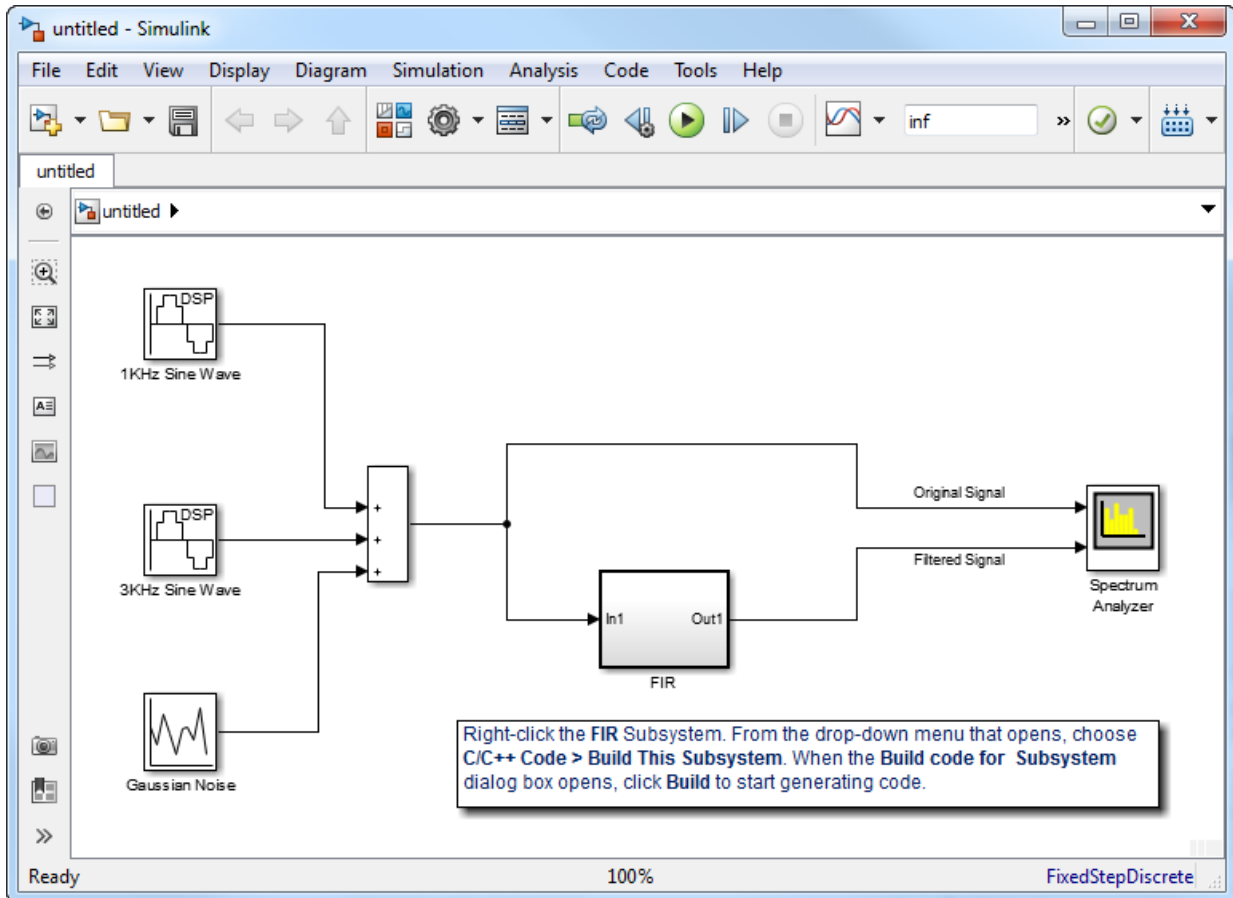
ARM Cortex-A Model Template

Click the **ARM Cortex-A Model** to create a basic model for code replacement with the Ne10 library, which is optimized for ARM Cortex-A Processors. This model is configured with settings recommended for DSP System Toolbox and ARM Cortex-A processors.



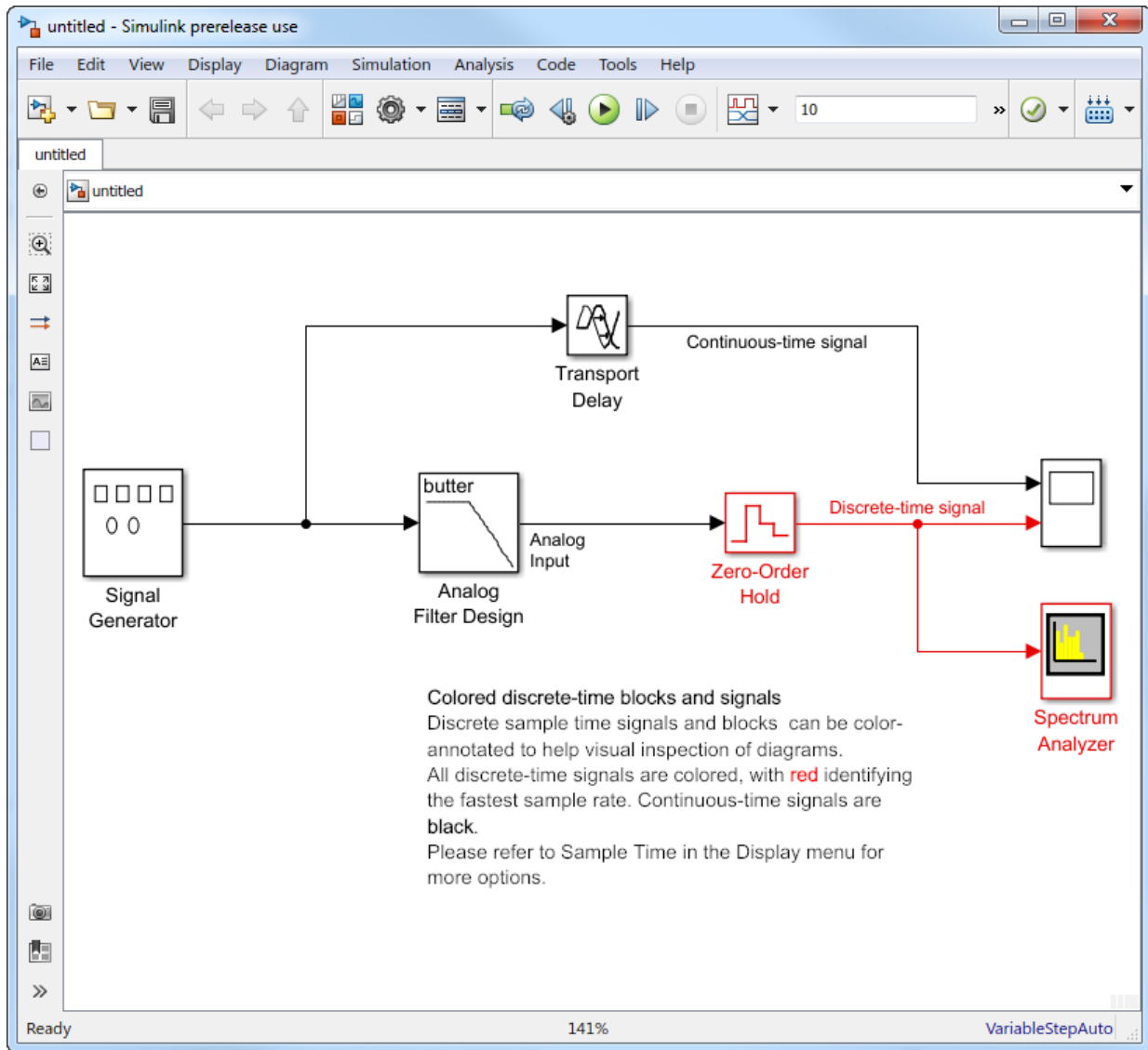
ARM Cortex-M Model Template

Click the **ARM Cortex-M Model** to create a basic model for code replacement with the CMSIS library, which is optimized for ARM Cortex-M processors. This model is configured with settings recommended for DSP System Toolbox and ARM Cortex-M processors.



Mixed-Signal System Template

Click the **Mixed-Signal System** template to create a basic A/D converter model configured with settings recommended for DSP System Toolbox and mixed-signal systems. This model performs A/D conversion by implementing an analog anti aliasing filter followed by a zero-order hold circuit. The model acts as a starting point for modeling mixed-signal systems in Simulink using DSP System Toolbox. All discrete-time signals are colored in red to indicate the fastest sample rate. Continuous-time signals are colored in black. For additional sample time options, from the **Display** menu, select **Sample Time**.



See Also

- “Filter Frames of a Noisy Sine Wave Signal in Simulink”

Design Filters in Simulink

- “Design and Implement a Filter” on page 2-2
- “Adaptive Filters” on page 2-10

Design and Implement a Filter

In this section...
“Design a Digital Filter in Simulink” on page 2-2
“Add a Digital Filter to Your Model” on page 2-6

Design a Digital Filter in Simulink

You can design lowpass, highpass, bandpass, and bandstop filters using either the Digital Filter Design block or the Filter Realization Wizard. These blocks are capable of calculating filter coefficients for various filter structures. In this section, you use the Digital Filter Design block to convert white noise to low frequency noise so you can simulate its effect on your system.

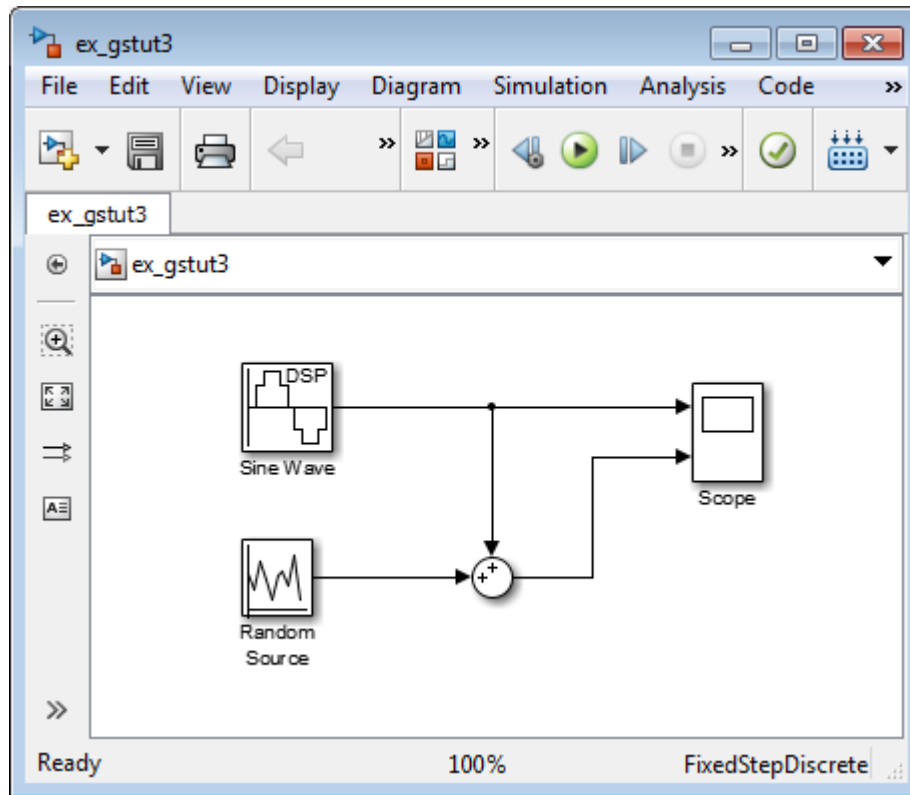
As a practical application, suppose a pilot is speaking into a microphone within the cockpit of an airplane. The noise of the wind passing over the fuselage is also reaching the microphone. A sensor is measuring the noise of the wind on the outside of the plane. You want to estimate the wind noise inside the cockpit and subtract it from the input to the microphone so that only the pilot's voice is transmitted. In this chapter, you first learn how to model the low frequency noise that is reaching the microphone. Later, you learn how to remove this noise so that only the pilot's voice is heard.

In this topic, you use a Digital Filter Design block to create low frequency noise, which models the wind noise inside the cockpit:

- 1 Open the model by typing

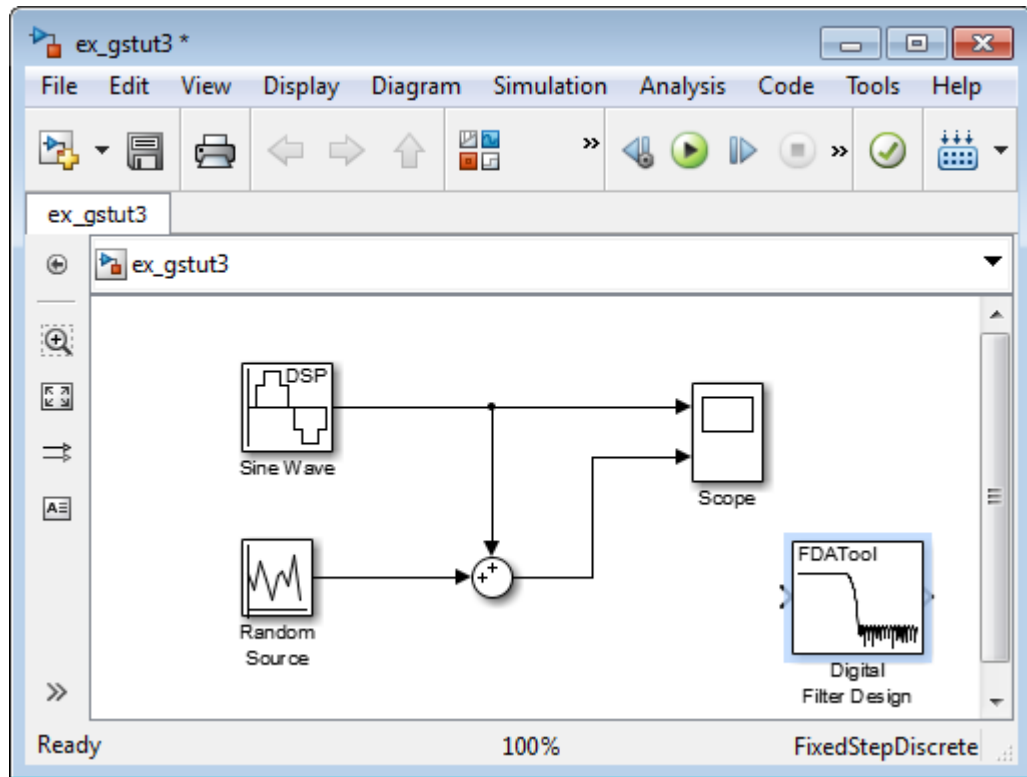
```
ex_gstut3
```

at the MATLAB command prompt. This model contains a Scope block that displays the original sine wave and the sine wave with white noise added.



- 2 Open the DSP System Toolbox library by typing `dspLib` at the MATLAB command prompt.
- 3 Convert white noise to low frequency noise by introducing a Digital Filter Design block into your model. In the airplane scenario, the air passing over the fuselage creates white noise that is measured by a sensor. The Random Source block models this noise. The fuselage of the airplane converts this white noise to low frequency noise, a type of colored noise, which is heard inside the cockpit. This noise contains only certain frequencies and is more difficult to eliminate. In this example, you model the low frequency noise using a Digital Filter Design block. This block uses the functionality of the Filter Design and Analysis Tool (FDATool) to design a filter.

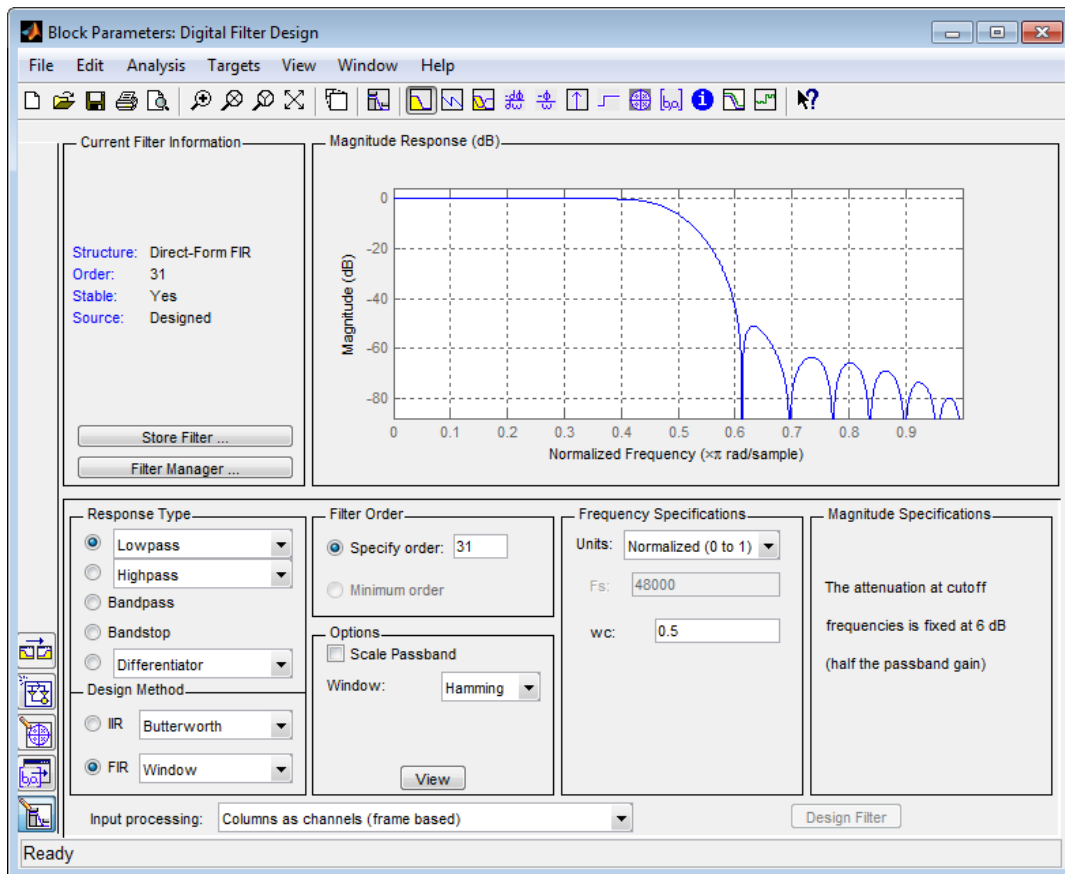
Double-click the Filtering library, and then double-click the Filter Implementations sublibrary. Click-and-drag the Digital Filter Design block into your model.



- 4 Set the Digital Filter Design block parameters to design a lowpass filter and create low frequency noise. Open the block parameters dialog box by double-clicking the block. Set the parameters as follows:
 - **Response Type** = Lowpass
 - **Design Method** = **FIR** and, from the list, choose Window
 - **Filter Order** = **Specify order** and enter 31
 - **Scale Passband** – Cleared
 - **Window** = Hamming
 - **Units** = Normalized (0 to 1)
 - **wc** = 0.5

Based on these parameters, the Digital Filter Design block designs a lowpass FIR filter with 32 coefficients and a cutoff frequency of 0.5. The block multiplies the time-domain response of your filter by a 32 sample Hamming window.

- 5 Click **Design Filter** at the bottom center of the dialog box to view the magnitude response of your filter in the **Magnitude Response** pane. The Digital Filter Design dialog box should now look similar to the following figure.



You have now designed a digital lowpass filter using the Digital Filter Design block.

You can experiment with the Digital Filter Design block in order to design a filter of your own. For more information on the block functionality, see the Digital Filter Design block reference page.

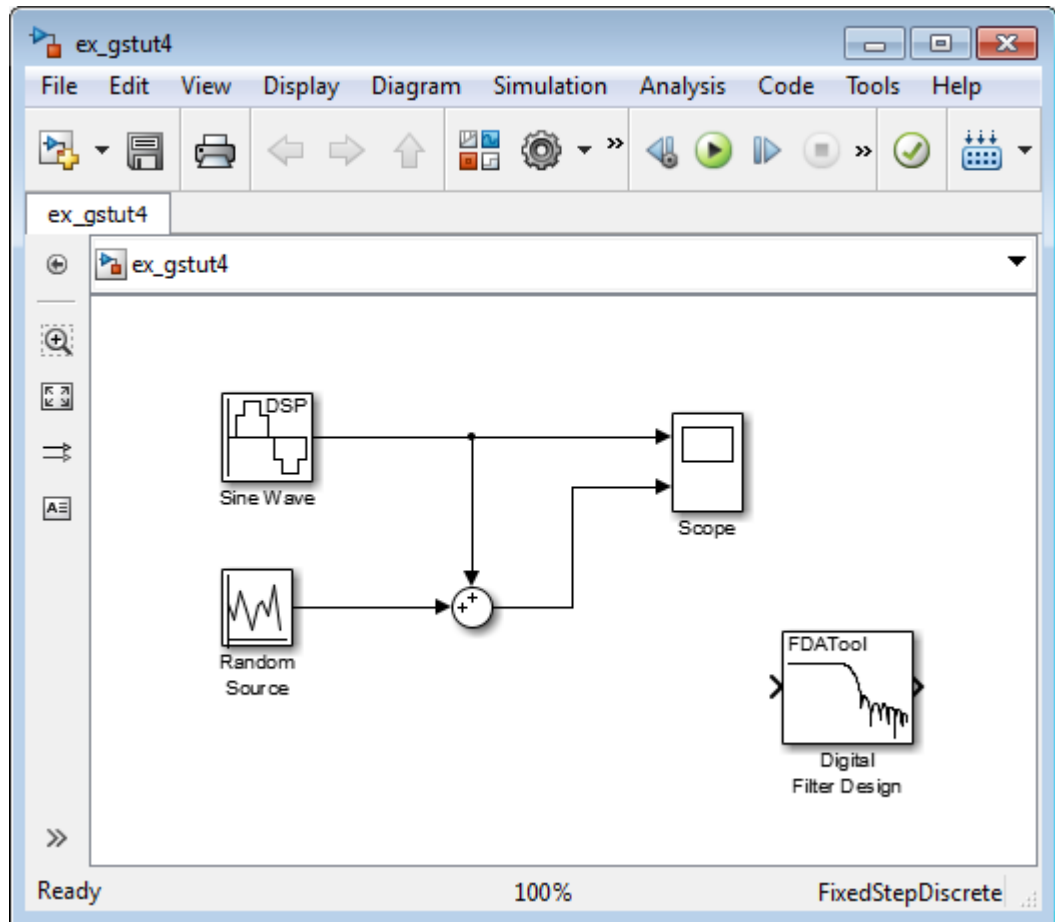
Add a Digital Filter to Your Model

In this topic, you add the lowpass filter you designed in “Design a Digital Filter in Simulink” on page 2-2 to your block diagram. Use this filter, which converts white noise to colored noise, to simulate the low frequency wind noise inside the cockpit:

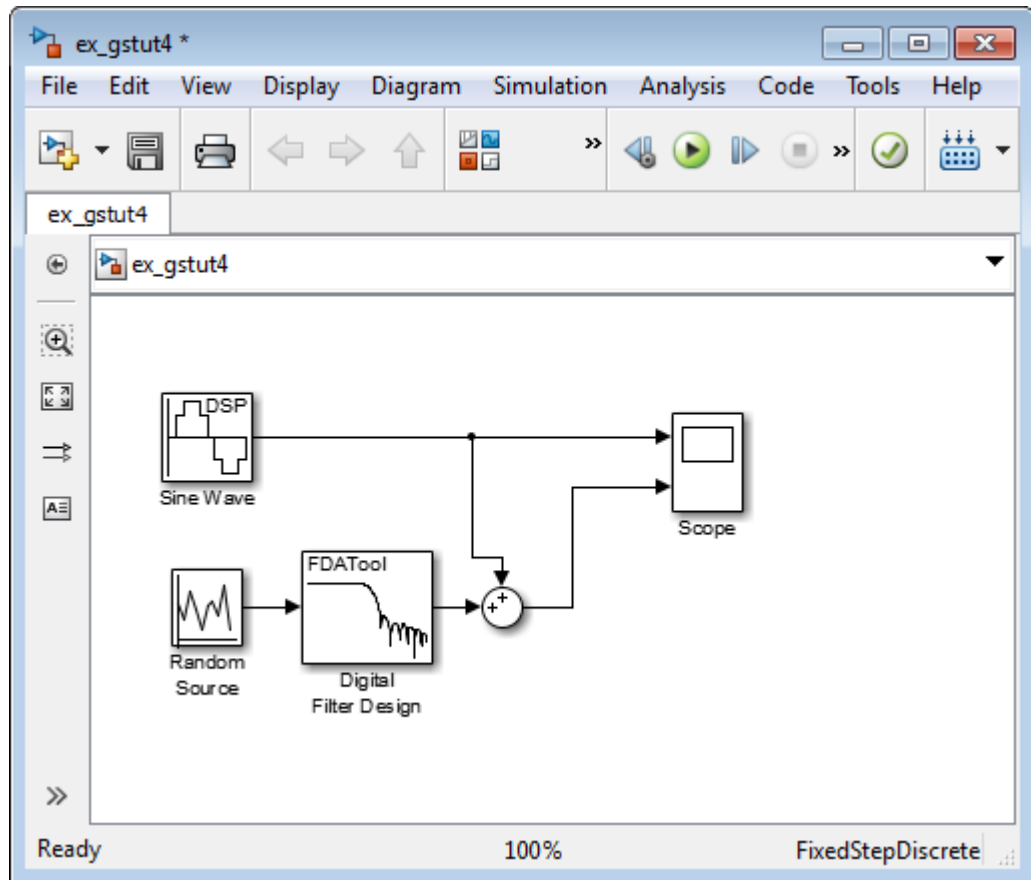
- 1 If the model you created in “Design a Digital Filter in Simulink” on page 2-2 is not open on your desktop, you can open an equivalent model by typing

`ex_gstut4`

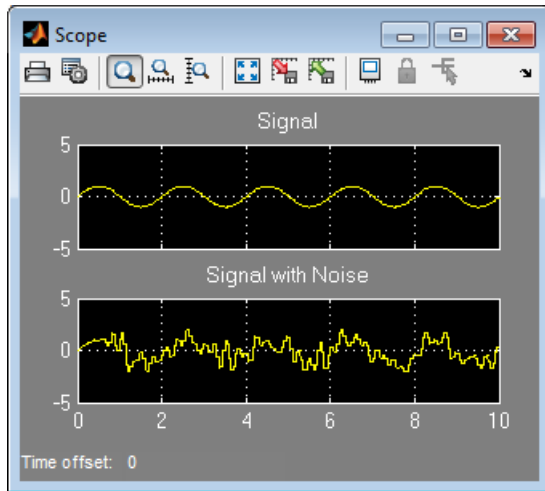
at the MATLAB command prompt.



- 2 Incorporate the Digital Filter Design block into your block diagram by placing it between the Random Source block and the Sum block.



- 3 Run your model and view the results in the Scope window. This window shows the original input signal and the signal with low frequency noise added to it.



You have now built a digital filter and used it to model the presence of colored noise in your signal. This is analogous to modeling the low frequency noise reaching the microphone in the cockpit of the aircraft. Now that you have added noise to your system, you can experiment with methods to eliminate it.

Adaptive Filters

In this section...
“Design an Adaptive Filter in Simulink” on page 2-10
“Add an Adaptive Filter to Your Model” on page 2-13
“View the Coefficients of Your Adaptive Filter” on page 2-18
“Summary” on page 2-22

Adaptive filters track the dynamic nature of a system and allow you to eliminate time-varying signals. The DSP System Toolbox libraries contain blocks that implement least-mean-square (LMS), block LMS, fast block LMS, and recursive least squares (RLS) adaptive filter algorithms. These filters minimize the difference between the output signal and the desired signal by altering their filter coefficients. Over time, the adaptive filter's output signal more closely approximates the signal you want to reproduce.

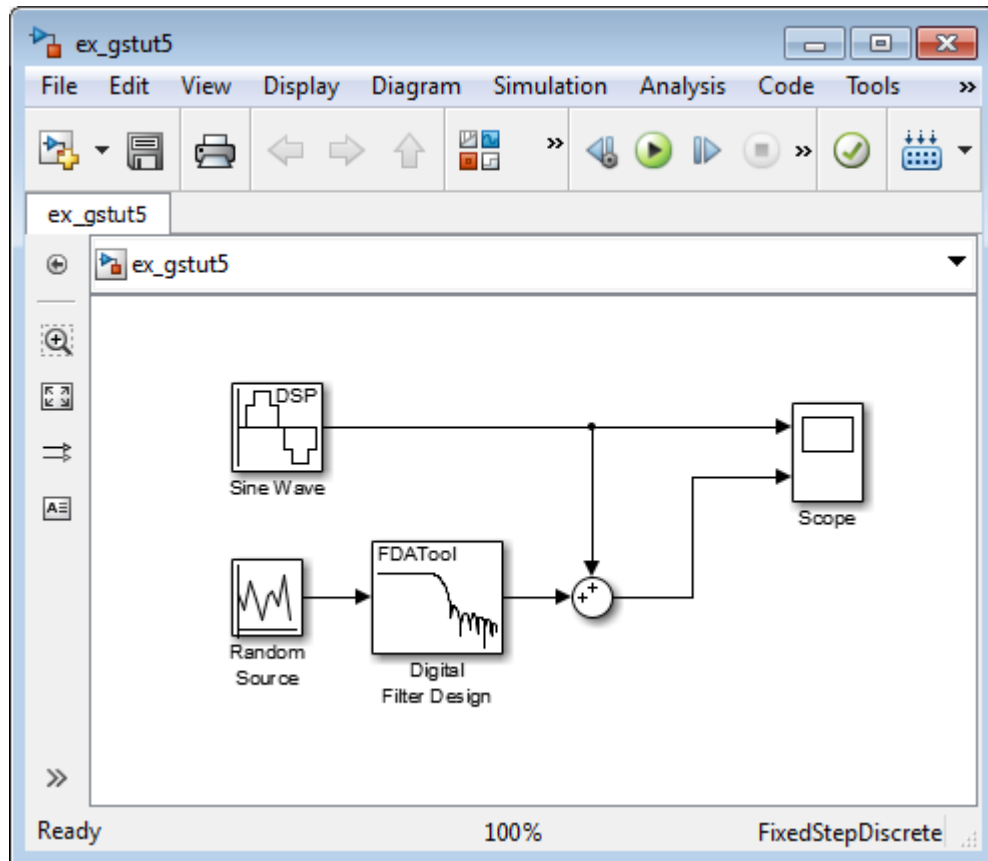
Design an Adaptive Filter in Simulink

In this example, you design an LMS adaptive filter to remove the low frequency noise in your signal:

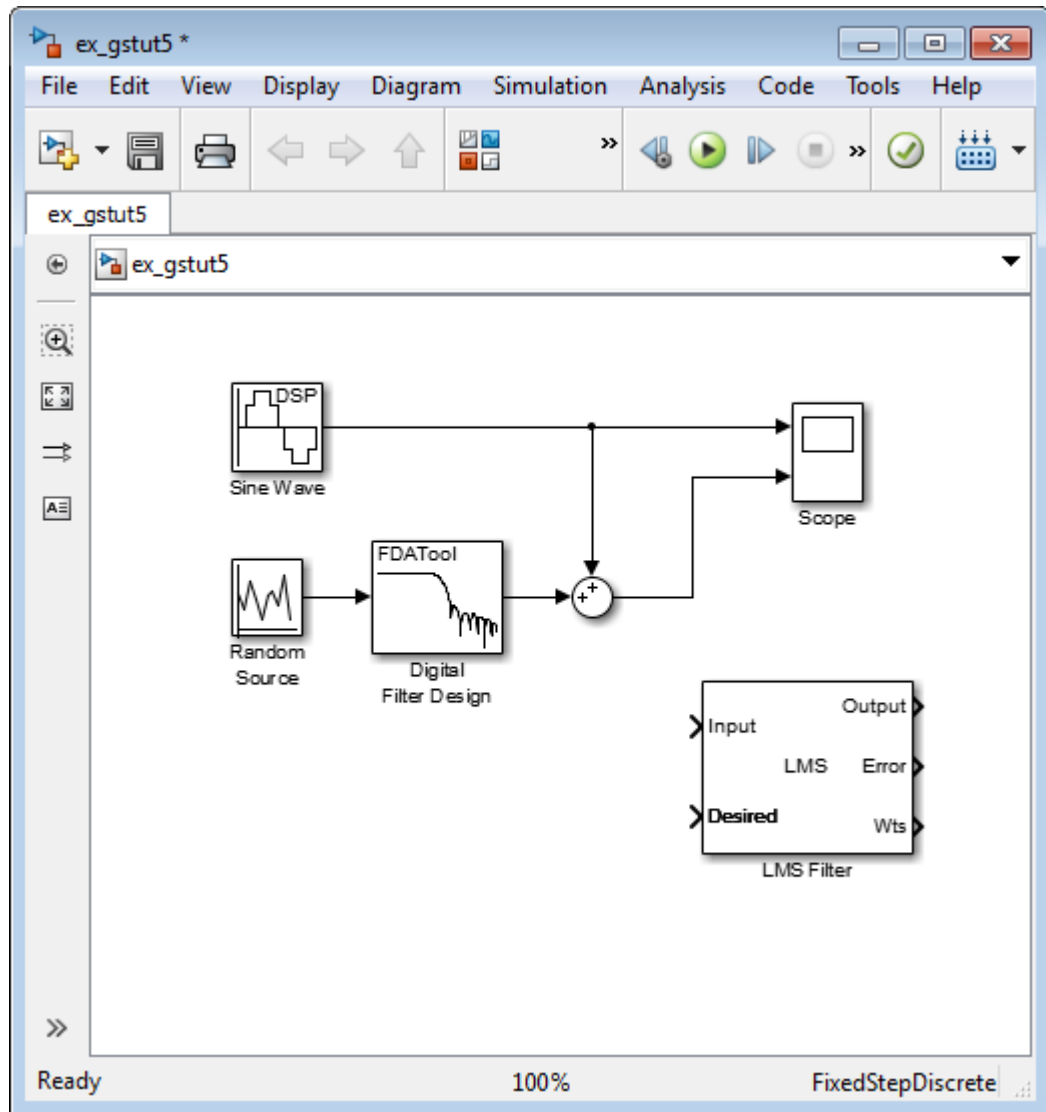
- 1 If the model you created in “Add a Digital Filter to Your Model” on page 2-6 is not open on your desktop, you can open an equivalent model by typing

`ex_gstut5`

at the MATLAB command prompt.



- 2 Open the DSP System Toolbox library by typing `dspLib` at the MATLAB command prompt.
- 3 Remove the low frequency noise from your signal by adding an LMS Filter block to your system. In the airplane scenario, this is equivalent to subtracting the wind noise inside the cockpit from the input to the microphone. Double-click the Filtering library, and then double-click the Adaptive Filters library. Add the LMS Filter block into your model.



- 4 Set the LMS Filter block parameters to model the output of the Digital Filter Design block. Open the LMS Filter dialog box by double-clicking the block. Set and apply the block parameters as follows:
 - **Algorithm** = Normalized LMS

- **Filter length** = 32
- **Specify step size via** = Dialog
- **Step size (μ)** = 0.1
- **Leakage factor (0 to 1)** = 1.0
- **Initial value of filter weights** = 0
- Clear the **Adapt port** check box.
- **Reset port** = None
- Select the **Output filter weights** check box.

Based on these parameters, the LMS Filter block computes the filter weights using the normalized LMS equations. The filter order you specified is the same as the filter order of the Digital Filter Design block. The **Step size (μ)** parameter defines the granularity of the filter update steps. Because you set the **Leakage factor (0 to 1)** parameter to 1.0, the current filter coefficient values depend on the filter's initial conditions and all of the previous input values. The initial value of the filter weights (coefficients) is zero. Since you selected the **Output filter weights** check box, the Wts port appears on the block. The block outputs the filter weights from this port.

Now that you have set the block parameters of the LMS Filter block, you can incorporate this block into your block diagram.

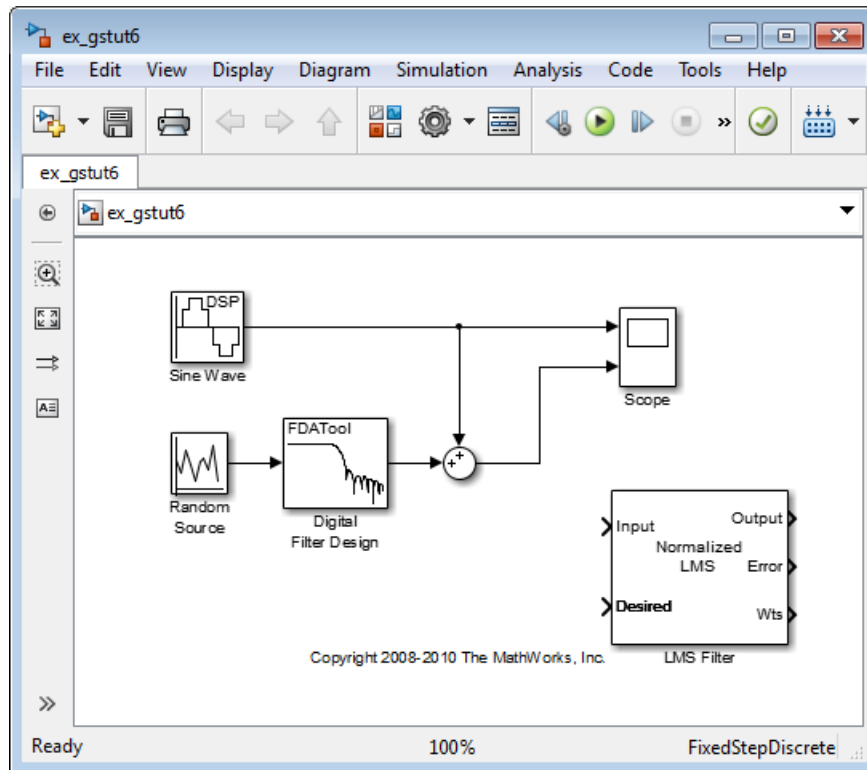
Add an Adaptive Filter to Your Model

In this example, you recover your original sinusoidal signal by incorporating the adaptive filter you designed in “Design an Adaptive Filter in Simulink” on page 2-10 into your system. In the aircraft scenario, the adaptive filter models the low frequency noise heard inside the cockpit. As a result, you can remove the noise so that the pilot's voice is the only input to the microphone:

- 1 If the model you created in “Design an Adaptive Filter in Simulink” on page 2-10 is not open on your desktop, you can open an equivalent model by typing

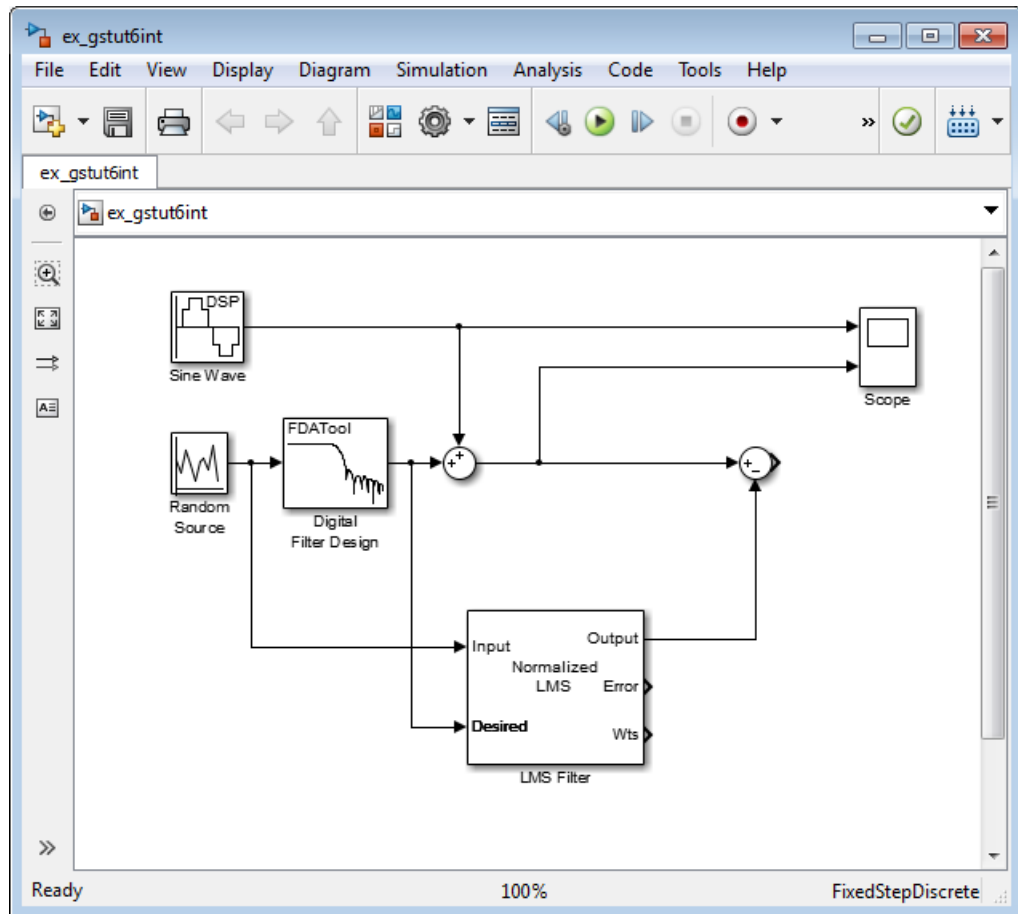
```
ex_gstut6
```

at the MATLAB command prompt.



- 2 Add a Sum block to your model to subtract the output of the adaptive filter from the sinusoidal signal with low frequency noise. From the Simulink Math Operations library, drag a Sum block into your model. Open the Sum dialog box by double-clicking this block. Change the **List of signs** parameter to $|+-$ and then click **OK**.
- 3 Incorporate the LMS Filter block into your system.
 - a Connect the output of the Random Source block to the Input port of the LMS Filter block. In the aircraft scenario, the random noise is the white noise measured by the sensor on the outside of the airplane. The LMS Filter block models the effect of the airplane's fuselage on the noise.
 - b Connect the output of the Digital Filter Design block to the Desired port on the LMS Filter block. This is the signal you want the LMS block to reproduce.
 - c Connect the output of the LMS Filter block to the negative port of the Sum block you added in step 2.

- d Connect the output of the first Sum block to the positive port of the second Sum block. Your model should now look similar to the following figure.



The positive input to the second Sum block is the sum of the input signal and the low frequency noise, $s(n) + y$. The negative input to the second Sum block is the LMS Filter block's best estimation of the low frequency noise, y' . When you subtract the two signals, you are left with an approximation of the input signal.

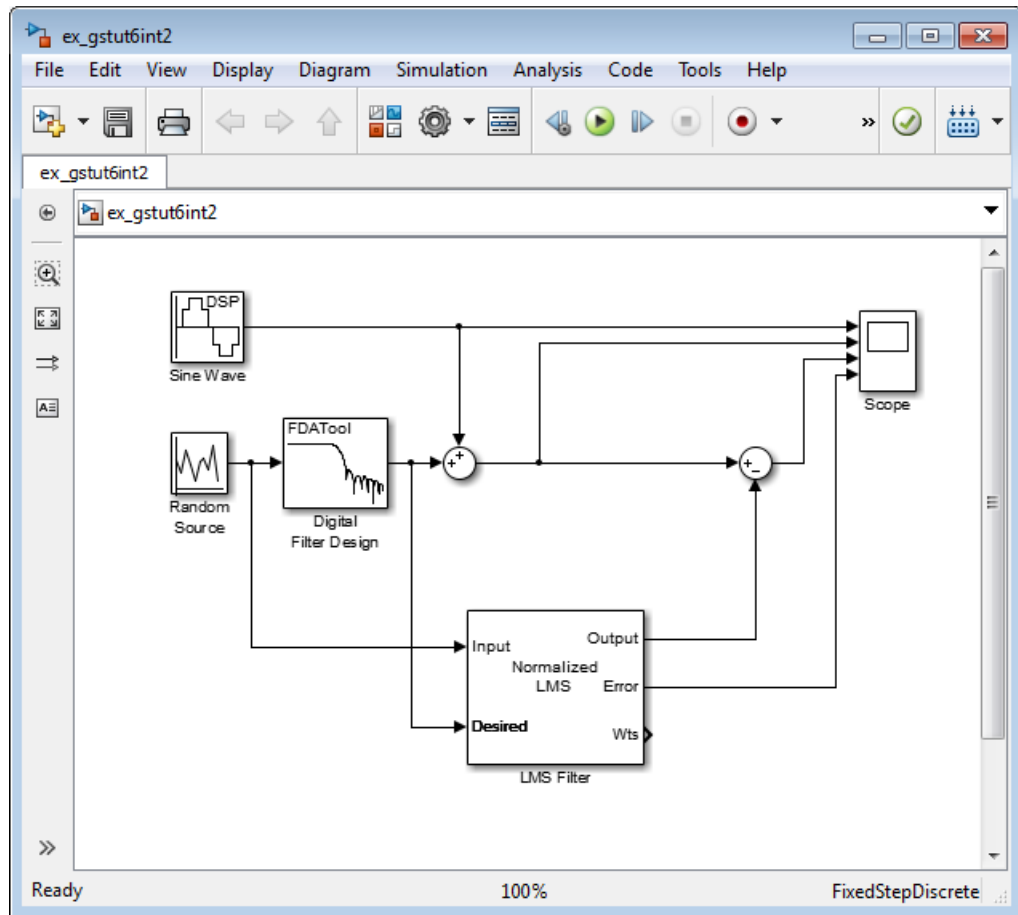
$$s(n)_{\text{approx}} = s(n) + y - y'$$

In this equation:

- $s(n)$ is the input signal
- $s(n)_{\text{approx}}$ is the approximation of the input signal
- y is the noise created by the Random Source block and the Digital Filter Design block
- y' is the LMS Filter block's approximation of the noise

Because the LMS Filter block can only approximate the noise, there is still a difference between the input signal and the approximation of the input signal. In subsequent steps, you set up the Scope block so you can compare the original sinusoidal signal with its approximation.

- 4 Add two additional inputs and axes to the Scope block. Open the Scope dialog box by double-clicking the Scope block. Click the **Parameters** button. For the **Number of axes** parameter, enter 4. Close the dialog box by clicking **OK**.
- 5 Label the new Scope axes. In the Scope window, right-click on the third axes and select **Axes properties**. The Scope properties: axis 3 dialog box opens. In the **Title** box, enter **Approximation of Input Signal**. Close the dialog box by clicking **OK**. Repeat this procedure for the fourth axes and label it **Error**.
- 6 Connect the output of the second Sum block to the third port of the Scope block.
- 7 Connect the output of the **Error** port on the LMS Filter block to the fourth port of the Scope block. Your model should now look similar to the following figure.



In this example, the output of the **Error** port is the difference between the LMS filter's desired signal and its output signal. Because the error is never zero, the filter continues to modify the filter coefficients in order to better approximate the low frequency noise. The better the approximation, the more low frequency noise that can be removed from the sinusoidal signal. In the next topic, "View the Coefficients of Your Adaptive Filter" on page 2-18, you learn how to view the coefficients of your adaptive filter as they change with time.

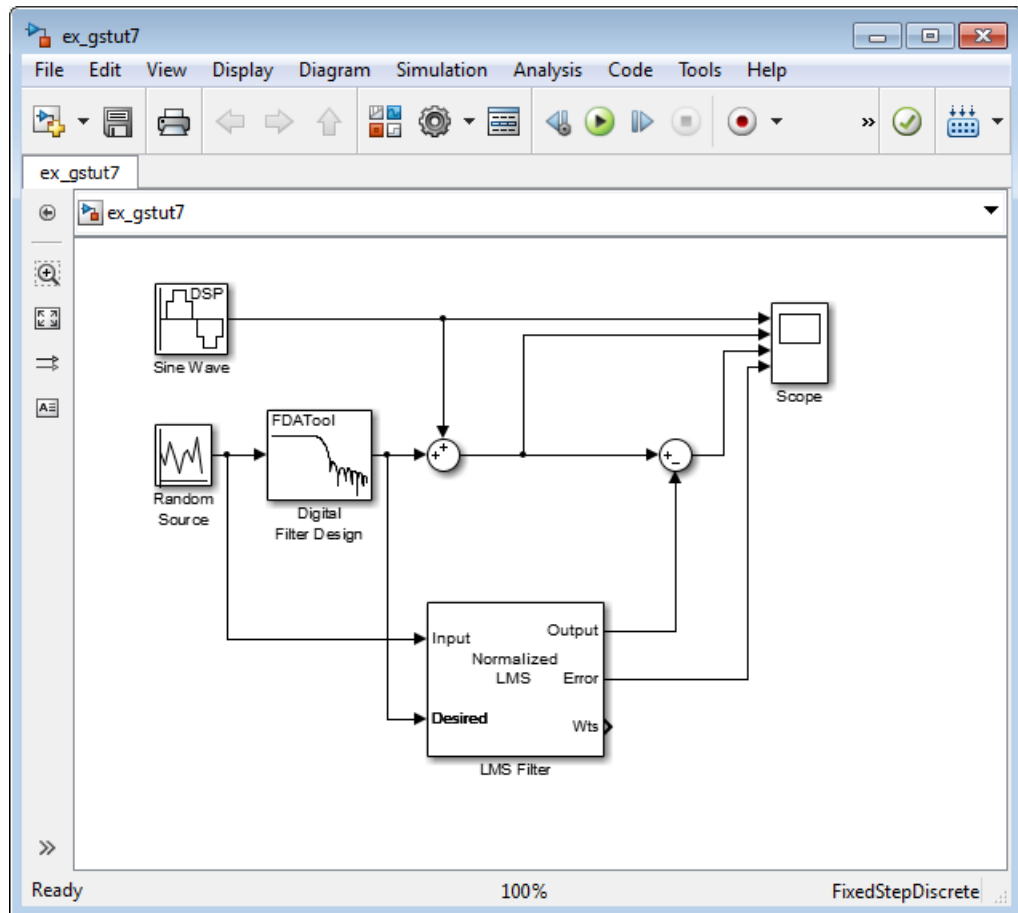
View the Coefficients of Your Adaptive Filter

The coefficients of an adaptive filter change with time in accordance with a chosen algorithm. Once the algorithm optimizes the filter's performance, these filter coefficients reach their steady-state values. You can view the variation of your coefficients, while the simulation is running, to see them settle to their steady-state values. Then, you can determine whether you can implement these values in your actual system:

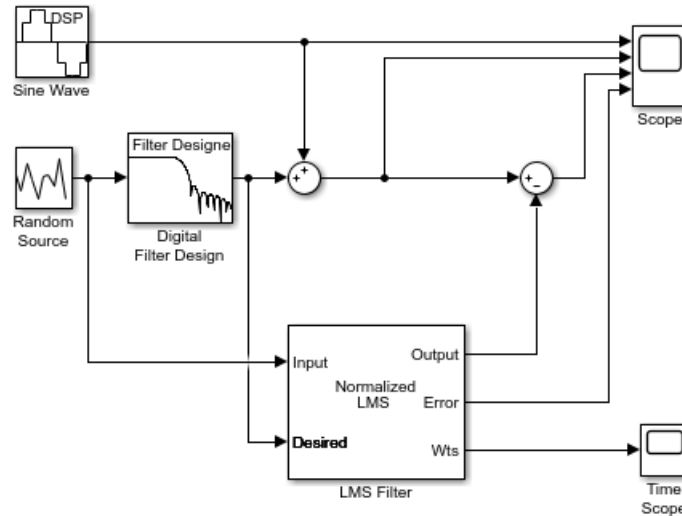
- 1 If the model you created in “Add an Adaptive Filter to Your Model” on page 2-13 is not open on your desktop, you can open an equivalent model by typing

```
ex_gstut7
```

at the MATLAB command prompt. Note that the **Wts** port of the adaptive filter, which outputs the filter weights, still needs to be connected.



- 2 Open the DSP System Toolbox library by typing `dsplib` at the MATLAB command prompt.
- 3 View the filter coefficients by connecting a Time Scope block to the **Wts** port of the LMS Filter block.
- 4 On the Time Scope block, modify the following configuration properties:
 - **Time span** = 0.05
 - **Y-limits (Minimum)** = -0.15
 - **Y-limits (Maximum)** = 0.5



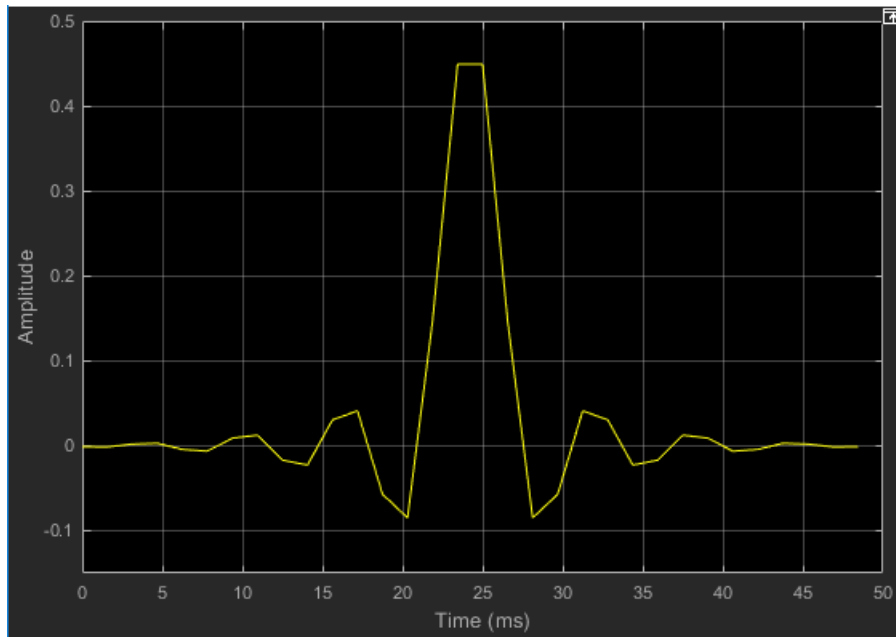
- 5 Open the Configuration Parameters dialog box. In the **Solver** pane, set these configuration parameters:
 - a **Stop time** = `inf`
 - b **Type** = Fixed-step
 - c **Solver** = `discrete` (no continuous states)

We recommend these configuration parameters for models that contain DSP System Toolbox blocks. Because these blocks calculate values directly rather than solving differential equations, you must configure the Simulink Solver to behave like a scheduler. The Solver, while in scheduler mode, uses a block's sample time to determine when the code behind each block is executed. For example, the sample time of the Sine Wave and Random Source blocks in this model is `0.05`. The Solver executes the code behind these blocks, and every other block with this sample time, once every 0.05 second.

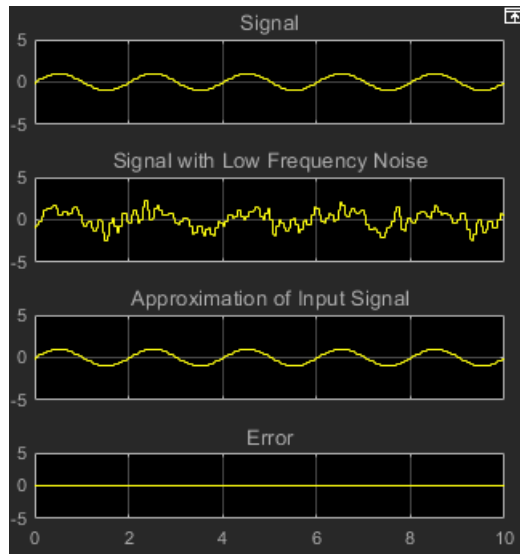
Note When working with models that contain DSP System Toolbox blocks, use source blocks that enable you to specify their sample time. If your source block does not have a **Sample time** parameter, you must add a Zero-Order Hold block in your model and use it to specify the sample time. For more information, see “Continuous-

Time Source Blocks". The exception to this rule is the Constant block, which can have a constant sample time. When it does, Simulink executes this block and records the constant value once at the start of the simulation and any time you tune a parameter. This allows for faster simulations and more compact generated code.

- 6 Run your model and view the behavior of your filter coefficients in the Time Scope, which opens automatically when your simulation starts. Over time, the filter coefficients approach their steady-state values.



You can also view the behavior of the system in the Scope window. Over time, you see the error decrease and the approximation of the input signal more closely match the original sinusoidal input signal.



You have now created a model capable of adaptive noise cancellation.

Summary

You have learned how to design a lowpass filter using the Digital Filter Design block. You also learned how to create an adaptive filter using the LMS Filter block. The DSP System Toolbox product has other blocks capable of designing and implementing digital and adaptive filters. For more information on the filtering capabilities of this product, see “Filter Design” and “Filter Analysis”.

Because all blocks in this model have the same sample time, this model is single rate, and Simulink ran it in `SingleTasking` solver mode. If the blocks in your model have different sample times, your model is multirate and Simulink might run it in `MultiTasking` solver mode. For more information on solver modes, see “Recommended Settings for Discrete-Time Simulations”.

To learn how to generate code from your model using the Simulink Coder product, see “Generate C Code from Simulink Model”.

See Also

Digital Filter Design | LMS Filter

More About

- “Filter Design”
- “Filter Analysis”

