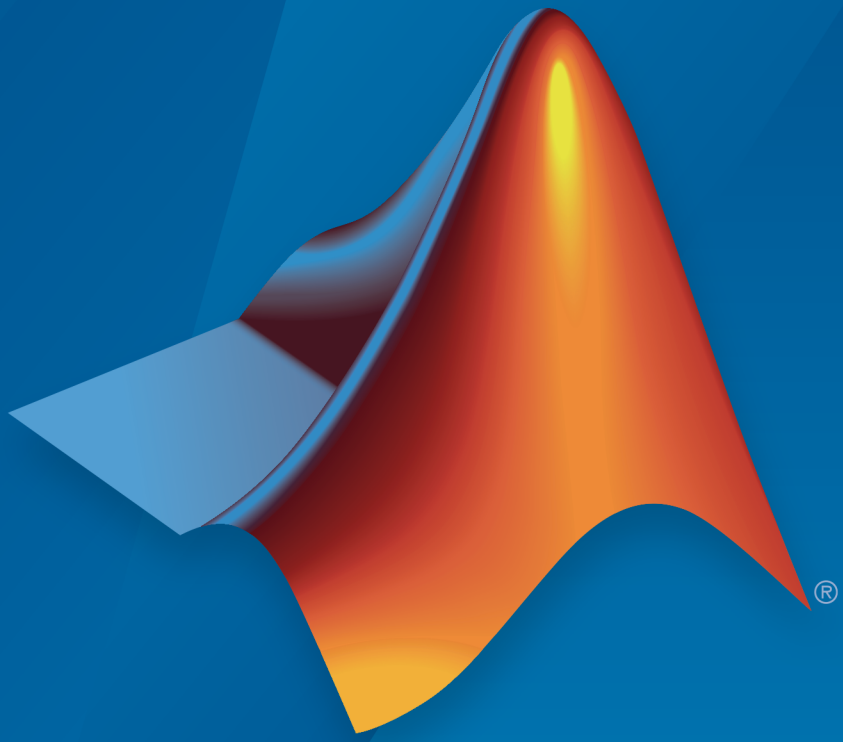
Audio System Toolbox™ Release Notes



MATLAB® & SIMULINK®

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

Audio System Toolbox™ Release Notes

Contents

R2016b

R2016a

R2016b

Version: 1.1

New Features

Audio Plugin Hosting: Run and test VST plugins directly in MATLAB

The `loadAudioPlugin` function enables you to host external VST and VST3 plugins in MATLAB®. You can process audio using the algorithm of the hosted plugin. You can interact with the hosted plugin programmatically by getting and setting parameters.

Improved Audio Test Bench: Choose from a wider range of input signals, and generate VST plugins directly from the app

The Audio Test Bench is a graphical debugging and testing suite for audio processing modules.

New abilities of the **Audio Test Bench** include:

- Switch the object under test in a single instance of the test bench.
- New input choices: wavetableSynthesizer, audioOscillator, dsp.Chirp, and dsp.ColoredNoise.
- Validate and generate VST plugins directly from the test bench.
- Track overrun and underrun in frames, seconds, or samples.

Loudness Metering: Measure standard-compliant loudness parameters

Measure integrated loudness and loudness range of an audio signal using the `integratedLoudness` function.

Measure momentary loudness, short-term loudness, integrated loudness, loudness range, and true-peak of streaming audio using the loudnessMeter System object™. You can also open an 'EBU-Mode' visualization for loudness metering.

Measure momentary loudness, short-term loudness, and true-peak in the Simulink® environment using the `Loudness Meter` block.

Octave-Band Filters: Select octave and fractional-octave signal bands using standard-compliant digital filters

Perform octave-band and fractional octave-band filtering for arbitrary center frequency using the octaveFilter System object. With this object, you can tune center frequency

and bandwidth while the simulation is running. To check your compliance to the ANSI S1.11-2004 standard, use the `isStandardCompliant` method. To visualize and validate your filter response, use the `visualize` method.

In the Simulink environment, use the `Octave Filter` block.

Audio Weighting Filters: Compensate signal magnitude for perceptual measurements using standard-compliant A-, C-, and K-weighted filters

Perform frequency-weighted filtering using the weightingFilter System object. With this object, you can design A-weighted and C-weighted filters based on the ANSI S1.42-2001 standard, or K-weighted filters based on the ITU-R BS.1770-4 standard. To check your compliance to the IEC 61672-1:2002 standard, use the `isStandardCompliant` method. To visualize and validate your filter response, use the `visualize` method.

In the Simulink environment, use the `Weighting Filter` block.

Plugin class creation and MIDI support for multiband parametric equalizer

New functionality for the multibandParametricEQ System object includes:

- Plugin class creation using `createAudioPluginClass`
- MIDI support using `configureMIDI`

`multibandParametricEQ` is now enabled for the **Audio Test Bench**.

Simpler way to call System objects

Instead of using the `step` method to perform the operation defined by a System object, you can call the object with arguments, as if it were a function. The `step` method will continue to work. This feature improves the readability of scripts and functions that use many different System objects.

For example, if you create a `weightingFilter` System object named `Cweight`, then you call the System object as a function with that name.

```
Cweight = weightingFilter('C-weighting');
Cweight(x)
```

The equivalent operation using the `step` method is:

```
Cweight = weightingFilter('C-weighting');
step(Cweight,x)
```

When the `step` method has the System object as its only argument, the function equivalent has no arguments. This function must be called with empty parentheses. For example, `step(sysobj)` and `sysobj()` perform equivalent operations.

R2016a

Version: 1.0

New Features

VST plugin generation for digital audio workstations

Audio System Toolbox™ enables the design and generation of VST plugins.

For more information, see Export a MATLAB Plugin to a DAW.

Interfaces to ASIO, ALSA, CoreAudio, and Windows Direct Sound

Audio System Toolbox enables real-time audio processing using low-latency audio drivers.

For more information, see Audio I/O: Buffering, Latency, and Throughput.

Interfaces to MIDI controls for real-time tuning of MATLAB and Simulink simulations

Audio System Toolbox enables real-time tuning in MATLAB and Simulink using MIDI controls.

For more information, see configureMIDI and Musical Instrument Digital Interface (MIDI).

Audio processing algorithms, sources, and measurements for MATLAB and Simulink

Audio System Toolbox provides algorithms and tools for the design, simulation, and desktop prototyping of audio processing systems.

For more information, see Audio Processing Algorithm Design.

Audio test bench to automatically generate an interactive audio simulation environment

Audio System Toolbox provides an all-in-one graphical debugging and testing suite.

For more information, see Audio Test Bench and Use the Audio Test Bench.

Support for C code generation

You can use MATLAB Coder™ to generate efficient C and C++ code for most Audio System Toolbox functions, classes, and System objects.

For a list of supported functions and objects, see Audio System Toolbox.

For a guide to developing code capable of C code generation, see MATLAB Programming for Code Generation.

Support for MATLAB Compiler

You can use MATLAB Compiler™ to share MATLAB programs as standalone applications.

For an example, see Deploy Audio Applications with MATLAB Compiler.