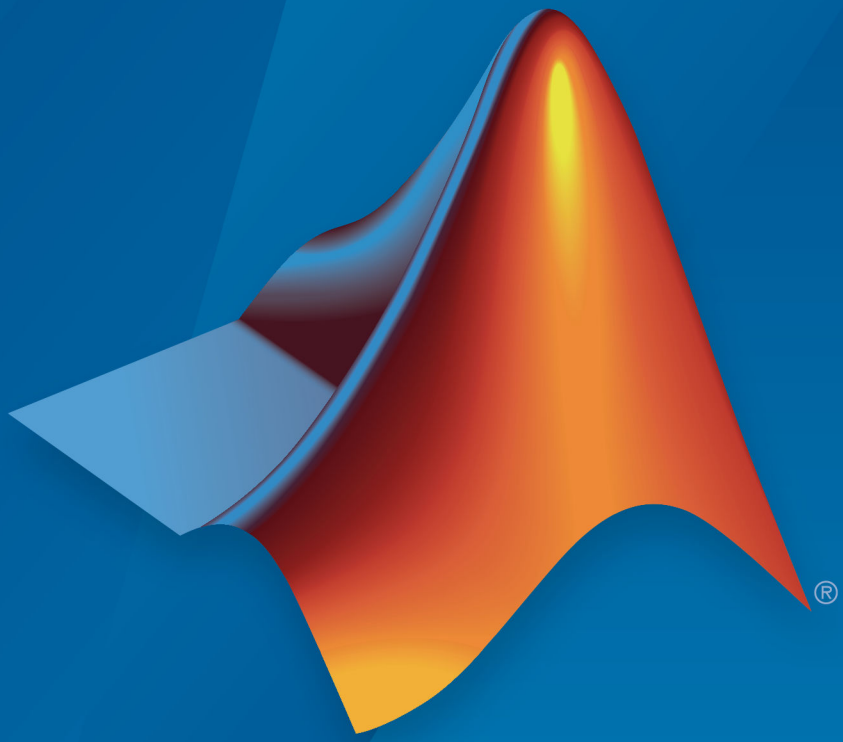
Audio Toolbox™ Release Notes



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.
1 Apple Hill Drive
Natick, MA 01760-2098

Audio Toolbox™ Release Notes

Contents

R2019a

R2018b

R2018a

R2017b

R2017a

R2016b

R2016a

R2019a

Version: 2.0

New Features

Compatibility Considerations

Modified Discrete Cosine Transform (MDCT)

Audio Toolbox enables you to transform to and from a compact frequency domain representation with perfect reconstruction:

- `mdct` –– Transform a signal into a compact frequency-domain representation using the modified discrete cosine transform (MDCT)
- `imdct` –– Transform a signal from a compact frequency-domain representation to the time domain using the inverse MDCT.
- `kbdwin` –– Create a Kaiser-Bessel derived window. This window enables perfect reconstruction when used with `mdct` and `imdct`.

Gammatone Filter Bank: Mimic the human auditory system

Use `gammatoneFilterBank` to decompose a signal by passing it through a bank of gammatone filters equally spaced on the equivalent rectangular bandwidth (ERB) scale. Gammatone filter banks are designed to model the human auditory system.

Mel-Spaced Spectrogram: Transform signals into perceptually-spaced compact time-frequency representations

Use `melSpectrogram` to compute the mel spectrogram of an audio signal. Mel spectrograms are popular features in deep-learning applications.

See “Speech Command Recognition Using Deep Learning” (Deep Learning Toolbox) and “Acoustic Scene Recognition Using Late Fusion” for examples.

Feature Extraction: Gammatone cepstral coefficients (GTCC)

Use `gtcc` to extract gammatone cepstral coefficients from audio signals. You can specify the window length and overlap length used for analysis, and optionally return the delta and delta-delta features calculated with look-ahead. Gammatone cepstral coefficients are a biologically inspired modification to mel frequency cepstral coefficients (`mfcc`), which have been shown to be robust to noise when used in machine-learning applications.

See “Classify Gender Using Long Short-Term Memory Networks” for an example.

Feature Extraction: Characterize level of harmonicity in audio signals

Use `harmonicRatio` to describe how much of the total energy of a signal is harmonic.

Feature Extraction: Characterize spectral shape of audio signals

Audio Toolbox now includes a suite of features that describe spectral shape, or timbre:

- `spectralCentroid`
- `spectralCrest`
- `spectralDecrease`
- `spectralEntropy`
- `spectralFlatness`
- `spectralFlux`
- `spectralKurtosis`
- `spectralRolloffPoint`
- `spectralSkewness`
- `spectralSlope`
- `spectralSpread`

See “Spectral Descriptors” for an overview of spectral descriptors and common applications.

Feature Extraction: Enhancements to cepstral feature extractors

`cepstralFeatureExtractor` and the Cepstral Feature Extractor block can now return gammatone cepstral coefficients (GTCC). Use `cepstralFeatureExtractor` in MATLAB® and the Cepstral Feature Extractor block in Simulink® when computing cepstral features for streaming audio.

Enhancements to Audio Datastore: Combine datastores and define custom read functions

`audioDatastore` has been enhanced to include the following functions:

- `transform` –– Define a custom read function on a datastore
- `combine` –– Combine data from multiple audio datastores into a single datastore

Convert between Hz, Bark, ERB, and mel domains

Audio Toolbox now includes conversion functions between Hz and popular auditory scales: Bark, equivalent rectangular bandwidth (ERB), and mel.

- `erb2hz` –– Convert from ERB scale to Hz
- `hz2erb` –– Convert from Hz to ERB scale
- `bark2hz` –– Convert from Bark scale to Hz
- `hz2bark` –– Convert from Hz to Bark scale
- `mel2hz` –– Convert from mel scale to Hz
- `hz2mel` –– Convert from Hz to mel scale

Generate JUCE projects from your audio plugins

`generateAudioPlugin` can now generate C/C++ source code and JUCE project files suitable for use with JUCE 5.3.2. This functionality requires a MATLAB Coder™ license.

Octave Filter Bank: Decompose signal into octave or fractional-octave subbands

Use `octaveFilterBank` to decompose signals into octave or fractional-octave subbands.

Graphically tune audio plugins and Audio Toolbox objects while streaming

Use `parameterTuner` to graphically tune parameters of audio plugins while streaming. `parameterTuner` is compatible with classes that inherit from `audioPlugin` and define tunable parameters.

You can also tune parameters of Audio Toolbox objects, including:

- `compressor`
- `expander`
- `limiter`
- `noiseGate`
- `octaveFilter`
- `crossoverFilter`
- `multibandParametricEQ`
- `graphicEQ`
- `audioOscillator`
- `wavetableSynthesizer`

Enhanced Parametric Equalization in Simulink

The Parametric EQ Filter block has been renamed as Parametric EQ and enhanced to use the `designParamEQ` algorithm. Instances of the Parametric EQ Filter block in existing models will not be automatically updated.

Improved Swept Sine Generation and Impulse Response Estimation

`sweeptone` and `impzest` have been improved to calculate more accurate impulse response estimations. The output of the `sweeptone` function has changed. `impzest` can be used with recordings using `sweeptone` from early releases as long as the corresponding excitation is specified to `impzest`.

New examples for deep learning, active noise control, pitch tracking, and MIDI

Examples for machine learning and deep learning:

- “Cocktail Party Source Separation Using Deep Learning Networks”
- “Voice Activity Detection in Noise Using Deep Learning”
- “Acoustic Scene Recognition Using Late Fusion”

- “Spoken Digit Recognition with Wavelet Scattering and Deep Learning”

Examples for active noise control:

- “Active Noise Control with Simulink Real-Time”

Example for pitch tracking:

- “Pitch Tracking Using Multiple Pitch Estimations and HMM”

Example for MIDI:

- “Convert MIDI Files into MIDI Messages”

R2018b

Version: 1.5

New Features

Audio Labeler App: Interactively define and visualize ground-truth labels for audio datasets

Use the **Audio Labeler** app for interactive audio labeling. The Audio Labeler app enables you to:

- Visualize the time-domain waveform during playback.
- Assign labels at the file level and region level.
- Create label definitions for consistent and fast labeling.
- Record audio.

Audio Datastore: Handle large collections of audio recordings for batch processing or machine and deep learning applications

Use `audioDatastore` to handle large collections of audio signals and associated labels that are too large to fit in memory. With audio datastores, you can:

- Point to a collection of audio files in a specified location.
- Associate labels with audio files.
- Split datastores according to label definitions and specified proportions.
- Randomize the order of audio files.
- Read files consecutively while monitoring progress.
- Process audio files in parallel when using a machine with multiple cores (requires Parallel Computing Toolbox™).

Octave Level Metering: Measure sound pressure level for octave and fractional-octave bands of audio signals

The `splMeter` System object™ now enables you to measure the sound pressure level (SPL) of octave and fractional-octave bands.

HRTF Interpolation: Compute Head-Related Transfer Functions (HRTF) for arbitrary positions from space-discrete datasets

Use `interpolateHRTF` to interpolate between HRTFs that were measured at known positions.

Impulse Response Measurements: Estimate impulse responses of acoustical systems using MATLAB code

Use `impzest` to estimate the impulse response of an audio system given a known excitation signal and a recorded signal. The `impzest` function supports the maximum length sequence (MLS) technique and the exponential sine sweep (ESS) technique for impulse response estimation. Use `mls` and `sweeptone` to generate the excitation signals.

Audio Test Bench enhancements

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

With the Audio Test Bench, you now can:

- Open custom visualizations for audio plugins.
- Use the visualization and tuning capabilities of the Audio Test Bench without writing audio to a device or file.

Additional examples for machine learning, deep learning, and spatial audio

Examples for machine learning and deep learning:

- Classify Gender Using Long Short-Term Memory Networks –– Use MFCC, pitch, harmonicity, and spectral centroid with a multilayer LSTM network to classify speaker gender.
- Denoise Speech Using Deep Learning Networks –– Use a spectrogram transformation and a deep CNN network to solve an audio regression problem.
- Music Genre Classification Using Wavelet Time Scattering –– Use wavelet time scattering and the audio datastore to classify music by genre.

Examples for spatial audio:

- Ambisonic Binaural Decoding –– Decode ambisonic audio into binaural audio using virtual loudspeakers.
- Ambisonic Plugin Generation –– Create ambisonic plugins using higher order ambisonic (HOA) demo functions.

R2018a

Version: 1.4

New Features

Impulse Response Measurer App: Interactively measure impulse and frequency responses of acoustic systems

The **Impulse Response Measurer** app enables you to acquire, analyze, and export impulse response and frequency response measurements through a user interface.

MIDI Message Interface: Send and receive MIDI messages of any type in MATLAB

You can now send and receive MIDI messages using the following features:

- `mididevice` –– Interface to a MIDI device in MATLAB. `mididevice` acts as a conduit between the MATLAB environment and your real-world MIDI device.
- `midimsg` –– Create a MIDI message in MATLAB
- `midisend` –– Send a MIDI message to an external MIDI device
- `midireceive` –– Receive a MIDI message from an external MIDI device

See MIDI Device Interface for a walkthrough of sending and receiving MIDI messages in MATLAB.

Voice Activity Detection: Automate the detection of speech content in audio signals

The `voiceActivityDetector` System object returns a confidence metric indicating the presence of speech in streaming audio signals. The input to the object can be time-domain or frequency-domain signals.

In the Simulink environment, use the Voice Activity Detector block.

Feature Extraction: Compute features of audio signals, such as pitch and MFCC

Detect the fundamental frequency of audio signals using the `pitch` function. You can choose between pitch detection algorithms, including the pitch estimation filter, normalized correlation function, cepstrum, log-harmonic summation, and summation of residual harmonics.

The Audio Toolbox enables batch and streaming approaches to cepstral feature extraction:

- `cepstralFeatureExtractor` –– Use the `cepstralFeatureExtractor` System object to process frame-based audio signals. You can specify your input in the time or frequency domain. This feature enables you to fine-tune the extracted features using the `BandEdges`, `FilterBankNormalization`, and `FilterBankDesignDomain` properties.

  In the Simulink environment, use the Cepstral Feature Extractor block.
- `mfcc` –– Use the `mfcc` function to process whole audio signals. You can specify the window length and overlap length used for analysis, and optionally return the delta and delta-delta features calculated with look-ahead.

Sound Pressure Level (SPL) Metering: Measure the level of acoustic signals in decibels relative to a standard perceptual reference

Use the `splMeter` System object to compute fast or slow, A-weighted or C-weighted, equivalent continuous, peak, and maximum sound pressure level measurements. You can calibrate your `splMeter` for your hardware and environment using the `CalibrationFactor` and `TimeInterval` properties. You can calculate your `CalibrationFactor` according to standards using the `calibrate` method.

Improved Audio Test Bench: Persistent I/O Settings and Bypass Mode

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

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

- Persistent input and output settings across sessions.
- A/B test your algorithm by bypassing the object under test.
- Visualize, analyze, and play unprocessed audio by not specifying an object under test.

Multichannel Support for RaspberryPi and STM Discovery Hardware

New multi-channel support for Mic In, Line In, and Speaker Out blocks for hardware support packages. See the hardware package release notes for more details:

- Release Notes for Embedded Coder Support Package for STMicroelectronics Discovery Boards (Embedded Coder Support Package for STMicroelectronics Discovery Boards)
- Release Notes for Simulink Support Package for Raspberry Pi Hardware (Simulink Support Package for Raspberry Pi Hardware)

Additional examples for word recognition and dataset recording

New examples include:

- Deep Learning Speech Recognition

  Use an auditory-based spectrogram to train a speaker-independent small vocabulary isolated word recognition system. Use `audioexample.Datastore` to manage large datasets. After training, you can run a streaming version for real-time word recognition. This example requires the Neural Network Toolbox™.

- Record Audio Datasets

  Record and label audio datasets using a user interface.

Speech-to-Text Transcription Using 3rd-Party Speech API

To perform speech-to-text transcription in MATLAB, use the `speech2text` function available on File Exchange. The function enables you to interface third-party speech-to-text APIs, including:

- Google® Speech API
- IBM® Watson Speech API
- Microsoft® Azure Speech API

The File Exchange submission includes a tutorial to help get you started.

R2017b

Version: 1.3

New Features

AU Plugin Hosting: Run and test Audio Units (AU) plugins in MATLAB on macOS

You can now load Audio Units (AU) plugins into MATLAB using the `loadAudioPlugin` function. You can interact with the hosted plugin graphically using the **Audio Test Bench**.

Graphic Equalization: Boost and cut standard octave or fractional octave frequency bands in MATLAB and Simulink

You can use the `graphicEQ` System object to perform equalization. The graphic equalizer uses the ANSI S1.11-2004 and ISO 266:1997(E) standards to determine and label the center frequencies of individual bandpass filters.

In the Simulink environment, use the Graphic EQ block.

Real-World Parameter Values for Hosted Plugins: Set and get values of hosted plugin parameters directly, using standard dot notation

Plugins loaded into the MATLAB environment using `loadAudioPlugin` are now populated with properties with real-world values. You can interact with the properties directly using standard dot notation.

See Host External Audio Plugins for a description of how normalized parameter values are heuristically interpreted as real-world values.

VST and AU plugins continue to support interaction through normalized parameter values.

MATLAB Code Generation from Audio Test Bench: Automatically generate MATLAB code for real-time audio streaming and processing

You can now generate MATLAB code from the **Audio Test Bench**. The generated MATLAB code is the script implementation of your **Audio Test Bench**. Settings in the

Audio Test Bench, such as plugin parameter values and scopes opened through the test bench, are also ported to the MATLAB code.

Direct Access to ASIO Configuration Panel: Open configuration panel of ASIO drivers directly from MATLAB

You can now open an ASIO settings user interface directly from MATLAB using the `asiosettings` function.

Additional input ports for Audio Toolbox blocks

The table describes the new optional input ports for tuning your block parameters.

Block	Parameters Tuned by New Optional Input Ports
Compressor	**Threshold (dB)**, **Ratio**, **Knee width (dB)**, **Attack time (s)**, **Release time (s)**
Expander	**Threshold (dB)**, **Ratio**, **Knee width (dB)**, **Attack time (s)**, **Release time (s)**, **Hold time (s)**
Limiter	**Threshold (dB)**, **Knee width (dB)**, **Attack time (s)**, **Release time (s)**, **Hold time (s)**
Noise Gate	**Threshold (dB)**, **Attack time (s)**, **Release time (s)**, **Hold time (s)**
Octave Filter	**Center frequency (Hz)**

Additional examples for machine learning, spatial audio, device measurements, and deployment to android

New examples include:

- Speaker Identification Using Pitch and MFCC
- Acoustic Beamforming Using a Microphone Array
- Measure Frequency Response of an Audio Device

- Parametric Audio Equalizer for Android Devices

Enhancements to existing examples include:

- Measure Impulse Response of an Audio System
- Measure Audio Latency

R2017a

Version: 1.2

New Features

Enhanced VST Workflow in Audio Test Bench: Interactively tune hosted VST plugins and test MATLAB objects in VST mode

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

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

- Host external VST and VST3 plugins. You can now interact with external plugins using the graphical UI of the **Audio Test Bench**, as you would in a DAW.
- Run audio plugins created in MATLAB as VST plugins.

Synchronized Playback and Acquisition: Play back and acquire audio signals synchronously in MATLAB via a single audioPlayerRecorder object

The `audioPlayerRecorder` System object reads and writes from an audio device simultaneously, enabling real-time system measurements when using ASIO, Core Audio, or ALSA drivers. Combining the play and record into a single object enables easy configuration and consistent latency between input and output.

WASAPI Driver Support on Windows: Stream signals from and to audio devices equipped with WASAPI drivers

The `audioDeviceWriter` and `audioDeviceReader` System objects and the Audio Device Reader and Audio Device Writer blocks now support WASAPI drivers on Windows machines.

File browsing in Audio Test Bench

You can now browse for audio input files and the object under test directly from the **Audio Test Bench**.

Additional fractional bandwidth option for octave filtering

The `octaveFilter` System object and Octave Filter block now support 2/3 octave bandwidth. This octave bandwidth is popular in graphic equalizer designs.

configureMIDI support for hosted audio plugins

You can now use the `configureMIDI` function to quickly synchronize your hosted audio plugins with MIDI devices.

Tab completion for parameter names and options

You can now use tab completion to complete parameter names and options for all System objects in Audio Toolbox. Tab completions also work for the `validateAudioPlugin`, `loadAudioPlugin`, and `generateAudioPlugin` functions.

Additional audio plugin examples

The Audio Plugin Gallery contains audio plugin example code that can be used as building blocks in larger systems, as models for design patterns, or as benchmarks for comparison.

New plugins in the gallery include:

- `audiopluginexample.FastConvolver`
- `audiopluginexample.Phaser`
- `audiopluginexample.MultiNotchFilter`
- `audiopluginexample.SpeechPitchDetector`
- `audiopluginexample.BeatDetector`

Enhancements to existing plugins in the gallery include:

- The `audiopluginexample.SpectralSubtractor` example now includes an analysis and synthesis buffering object. The `audiopluginexample.private.AnalysisAndSynthesisBuffer` object enables easy input/output buffering for the audio plugin API so that you can concentrate on your algorithm.

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 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 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 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 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 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 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.