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

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.

Contents

R2022a

R2021b

R2021a

R2020b

R2020a

R2019b

R2019a

R2017b

R2017a

R2022b

Version: 3.3

New Features

Bug Fixes

Compatibility Considerations

OpenL3, OpenL3 Preprocess, and OpenL3 Embeddings Blocks: Extract audio feature embeddings in Simulink using pretrained OpenL3 neural network

Use the OpenL3 Embeddings block to extract semantically meaningful feature vectors (embeddings). You can use the feature vectors as an input for a classification model. The block has functionality equivalent to the `openl3Embeddings` function.

Use the OpenL3 block to interact with the pretrained network directly. You can use this pretrained model for feature extraction. The block has functionality equivalent to the `openl3` function.

Use the OpenL3 Preprocess block to generate spectrograms that you can then feed to the OpenL3 pretrained network. You can use this block in transfer learning workflows. The block has functionality equivalent to the `openl3Preprocess` function.

The OpenL3 functionality requires Deep Learning Toolbox™.

Audio Plugin Block and generateSimulinkAudioPlugin: Use audio plugins in Simulink

This release introduces the Audio Plugin block and the `generateSimulinkAudioPlugin` function.

- The Audio Plugin block lets you use any audio plugin as a Simulink® block.
- Use the `generateSimulinkAudioPlugin` function to generate a System object™ compatible with Simulink from an audio plugin. Use the generated object in Simulink with the MATLAB System block.

generateAudioPlugin: Configure audio plugin generation through new UI

This release introduces a new UI to generate audio plugins when you call `generateAudioPlugin` with no input arguments. The UI has the same functionality as the command-line interface for `generateAudioPlugin`.

Audio Test Bench: Choose from multiple plugin formats and specify coder configuration when generating audio plugin binaries

The **Audio Test Bench** now has the same functionality as `generateAudioPlugin` and `audioPluginConfig` for generating audio plugin binaries. You can choose from multiple plugin binary formats, and you can specify the coder configuration for deep learning and code replacement libraries.

MFCC, Cepstral Coefficients, and Audio Delta Blocks: Extract cepstral features from audio signals in Simulink

This release introduces new blocks to the Audio Toolbox Simulink library for extracting cepstral coefficients and their delta features from audio signals. These features are commonly used to characterize speech signals.

- The MFCC and Cepstral Coefficients blocks return cepstral coefficients from audio signals. The blocks have functionality equivalent to the `mfcc` function and `cepstralCoefficients` function, respectively.
- The Audio Delta block returns the delta features of the input audio feature. The block has functionality equivalent to the `audioDelta` function.

speech2text: Convert speech to text using pretrained wav2vec 2.0 model

This release enhances the `speech2text` API to provide the wav2vec 2.0 pretrained neural network for state-of-the-art speech-to-text transcription.

Using `speech2text` with wav2vec 2.0 requires Deep Learning Toolbox and a one-time download of the pretrained network weights.

Signal Labeler: Automatically label speech signals with transcriptions using pretrained wav2vec 2.0 model

This release enhances the automatic speech-to-text labelling in the **Signal Labeler** app to use the wav2vec 2.0 pretrained neural network. You can choose the wav2vec 2.0 model or one of the third-party speech-to-text APIs to perform automatic labelling.

This functionality requires downloading the pretrained network weights for wav2vec 2.0 and Deep Learning Toolbox.

audioFeatureExtractor: Visualize features extracted from audio signals

This release introduces a new object function, `plotFeatures`, for the `audioFeatureExtractor` object. Use this function to visualize features such as pitch, cepstral coefficients, and spectral descriptors.

audioFeatureExtractor: GPU support for zerocrossrate

The `audioFeatureExtractor` object now supports `gpuArray` objects with the `zerocrossrate` feature for code acceleration.

You must have Parallel Computing Toolbox™ to use `gpuArray` objects with supported functions. See "Run MATLAB Functions on a GPU" (Parallel Computing Toolbox) for more details and "GPU Computing Requirements" (Parallel Computing Toolbox) to see which GPUs are supported.

Impulse Response Measurer: Automatic latency compensation

The **Impulse Response Measurer** can now use a loopback cable to measure the audio device latency that delays the measured impulse response. You can optionally remove this latency from the captured measurements.

Added Support for OPUS File Format

This release introduces support for working with OPUS audio files in the following features:

- `dsp.AudioFileReader` and `dsp.AudioFileWriter` System objects
- From Multimedia File and To Multimedia File Simulink blocks
- **Signal Labeler** app and **Audio Test Bench** app
- `audioDatastore` object

audioOscillator and wavetableSynthesizer: New MaxSamplesPerFrame property

This release introduces a new property, `MaxSamplesPerFrame`, to the `audioOscillator` and `wavetableSynthesizer` System objects. Use this property to allow more than the default max of 192000 samples per frame. Alternatively, set `MaxSamplesPerFrame` to a lower value to save memory when using code generation.

Pretrained Model on GitHub: Deep Speech speech-to-text model

To learn how to load a pretrained Deep Speech model into MATLAB®, see the Speech-to-Text Transcription Using Deep Speech repository. The Deep Speech model is suitable for transfer learning, code generation, and streaming.

To find the latest pretrained models for deep learning in MATLAB, see MATLAB Deep Learning Model Hub.

Additional examples for AI, audio system design, and audio filter design

AI

- "Audio Transfer Learning Using Experiment Manager"
- "Audio-Based Anomaly Detection for Machine Health Monitoring"
- "Train 3-D Speech Enhancement Network Using Deep Learning"
- "3-D Speech Enhancement Using Trained Filter and Sum Network"

Audio System Design

- "Model Smart Speaker in Simulink"

Audio Filter Design

- "Automated Design of Audio Filters for Room Equalization"

Functionality being removed or changed

Single-Band Parametric EQ Block: Input sample rate (Hz) parameter is now tunable
Behavior change

You can now tune the **Input sample rate (Hz)** parameter of the Single-Band Parametric EQ block during simulation.

sweeptone and Impulse Response Measurer: Exponential swept sine supports longer duration
Behavior change

The `sweeptone` function and the **Impulse Response Measurer** app now support longer durations for the exponential swept sine signal. For `sweeptone`, the duration can be a maximum of 60 seconds. For **Impulse Response Measurer**, the duration must be less than 60 seconds.

The Cepstral Feature Extractor block will be removed
Still runs

The Cepstral Feature Extractor block will be removed in a future release. Use the MFCC block or a combination of the Auditory Spectrogram, Cepstral Coefficients, and Audio Delta blocks instead.

Cepstral Feature Extractor Configuration	Recommended Replacement
Filter bank type parameter set to `Mel`	Use the MFCC block.
Filter bank type parameter set to `Gammatone`	Use the Auditory Spectrogram block combined with the Cepstral Coefficients block. See "Extract GTCC from Audio in Simulink" for an example.
Output delta or **Output delta-delta** parameters selected	If using the MFCC block, select the **Append delta** or **Append delta-delta** parameters. If using the Cepstral Coefficients block instead, use the Audio Delta block to extract delta features.
Log energy usage parameter set to `Append` or `Replace`	No replacement
Band edges of Mel filter bank (Hz) parameter specified	No replacement
Domain for Mel filter bank design parameter set to `Bin`	No replacement

cepstralFeatureExtractor will be removed
Warns

The `cepstralFeatureExtractor` System object will be removed in a future release. Use the `mfcc` and `gtcc` functions to compute the same features for batch signals. For streaming applications, improve performance by designing the filter bank once with `designAuditoryFilterBank`, and then apply the filter bank and extract the same features with `cepstralCoefficients` and `audioDelta` in the streaming loop. If you are extracting multiple audio features, use the `audioFeatureExtractor` object.

cepstralFeatureExtractor Configuration	Recommended Replacement

FilterBank property set to "Mel"	Use the mfcc function.
	
	`[audioIn,fs] = audioread("Counting-16-44p1-mono-15secs.wav");`
	`[coeffs,delta,deltaDelta] = mfcc(audioIn,fs);`
	
	Alternatively, use a combination of designAuditoryFilterBank and cepstralCoefficients. See "Mel Frequency Cepstral Coefficients" for an example. To extract delta features with this approach, use audioDelta.

FilterBank property set to "Gammatone"	Use the gtcc function.
	
	`[audioIn,fs] = audioread("Counting-16-44p1-mono-15secs.wav");`
	`[coeffs,delta,deltaDelta] = gtcc(audioIn,fs);`
	
	Alternatively, use a combination of designAuditoryFilterBank and cepstralCoefficients. See "Gammatone Frequency Cepstral Coefficients" for an example. To extract delta features with this approach, use audioDelta.

FilterBankDesignDomain property set to "Bin"	No replacement
FilterBankNormalization property set to "Area" or "None"	Use the designAuditoryFilterBank function, with the Normalization name-value argument set to "area" or "none", combined with the cepstralCoefficients function.

Audio Test Bench: Time Scope and Spectrum Analyzer can no longer be in the same window
Behavior change

Due to new interfaces for **Time Scope** and **Spectrum Analyzer** in the **Audio Test Bench** app, each scope can only be in its own separate window.

R2022a

Version: 3.2

New Features

Bug Fixes

Compatibility Considerations

VGGish, VGGish Preprocess, and VGGish Embeddings Blocks: Extract audio embeddings in Simulink using pretrained neural network

Use the VGGish Embeddings block to extract semantically meaningful 128-dimensional feature vectors (embeddings). You can use the feature vectors as input for a classification model. The block has functionality equivalent to the `vggishEmbeddings` function.

Use the VGGish block to interact with the pretrained network directly. You can use this pretrained model for feature extraction. The block has functionality equivalent to the `vggish` function.

Use the VGGish Preprocess block to generate mel spectrograms that can be fed to the VGGish pretrained network. You can use this block in transfer learning workflows. The block has functionality equivalent to the `vggishPreprocess` function.

The VGGish functionality requires Deep Learning Toolbox.

Access audio pretrained networks in Deep Network Designer

Starting this release, you can access pretrained audio deep networks from the **Deep Network Designer** (Deep Learning Toolbox) app. These pretrained networks are supported:

- `crepe`
- `openl3`
- `vggish`
- `yamnet`

You can use **Deep Network Designer** to visualize, edit, and train using the pretrained networks. **Deep Network Designer** requires Deep Learning Toolbox.

Shelving Filter: Boost or attenuate low or high frequencies in MATLAB and Simulink

This release introduces the `shelvingFilter` System object and the Shelving Filter block. You can specify or tune the filter gain, slope, and cutoff frequency, and you can visualize the filter response. The object supports MIDI devices and plugin class creation.

Auditory Spectrogram Blocks: Generate filter banks and extract auditory features in Simulink

This release enhances the Audio Toolbox Simulink library by adding blocks that can extract features from audio signals.

- The Design Auditory Filter Bank block returns frequency-domain auditory filter banks. The block has functionality equivalent to the `designAuditoryFilterBank` function. To create the popular mel filter bank, use the new Design Mel Filter Bank block.
- The Auditory Spectrogram enables users to extract mel, Bark, or equivalent rectangular bandwidth (ERB) spectrograms from audio signals. The block has functionality that builds on the `audioFeatureExtractor` object. To extract the popular mel spectrogram, use the new Mel Spectrogram block.

Signal Labeler App: Automatically detect speech regions in audio signals and label spoken words

The **Signal Labeler** app now provides automatic labeling of detected regions of speech and speech-to-text transcription.

- To automate the detection of speech content, the app uses the `detectSpeech` function.
- To perform speech-to-text transcription, the app uses the `speech2text` function available on File Exchange. The function interfaces with third-party speech-to-text APIs, including:

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

The `speech2text` entry on File Exchange includes a tutorial to help get you started.

Compatibility Considerations

The **Audio Labeler** app will be removed in a future release. Use **Signal Labeler** instead.

audioFeatureExtractor: Extract zero-cross rate and short-time energy

The `audioFeatureExtractor` object can now be used to extract zero-cross rates and short-time energies of audio signals.

ivectorSystem: Calibrate scores and toggle training progress display

This release introduces enhancements to the `ivectorSystem` object. The `calibrate` function calibrates the scores output by an i-vector system so they can be interpreted as a measure of confidence in a positive decision. The `Verbose` property can be set to display or suppress information about training progress.

Compatibility Considerations

- The `identify` object function now throws a warning if the scores from the i-vector system are not calibrated. Use `calibrate` to calibrate the scores.
- The `verify` object function now throws a warning if it is called with two output arguments and the scores from the i-vector system are not calibrated. Use `calibrate` to calibrate the scores.

Impulse Response Measurer app: Multichannel support and other enhancements

This release introduces enhancements to the **Impulse Response Measurer** app. Now you can:

- Automatically save device, method, and advanced settings and use them in later measurement sessions.
- Acquire impulse responses from one or more input channels to create filters and generate models for offline simulations. You can also have multiple output channels.

- Toggle between linear and logarithmic frequency axes in magnitude and phase plots.
- Adjust the settings of your selected ASIO™ device on Windows® platforms.
- Specify the number of samples per frame.
- See markers corresponding to individual samples when you zoom into a plot.

Dynamic Range Control and Filter System Objects: Handle visualizations programmatically and compute static characteristics

Starting this release, the `visualize` object function can output a figure handle for enhanced programmatic control. These System objects support the feature:

- **Dynamic Range Control** — `compressor`, `expander`, `limiter`, and `noiseGate`
- **Filters** — `crossoverFilter`, `graphicEQ`, `multibandParametricEQ`, `octaveFilter`, and `weightingFilter`
- **Loudness** — `loudnessMeter`

This release also introduces the `staticCharacteristic` function for the `compressor`, `expander`, and `limiter` System objects.

Compatibility Considerations

In previous releases, calling the `visualize` function of the dynamic range control System objects with an output argument returned static characteristic data. Starting in R2022a, that syntax returns a figure handle instead. Use the new `staticCharacteristic` function to compute static dynamic-range characteristics.

Audio Filter Blocks: Export filters to the MATLAB Workspace

Starting this release, every Simulink filter block in Audio Toolbox enables users to generate a filter System object whose properties are the same as the block parameters. The generated System object appears in the MATLAB Workspace. These blocks support the feature:

- Crossover Filter
- Weighting Filter
- Octave Filter
- Octave Filter Bank
- Gammatone Filter Bank
- Graphic EQ
- Multiband Parametric EQ

Convenience plots for spectral descriptors and other functions

This release introduces convenience plots for spectral descriptors and other functions. These functions support the feature:

- **Spectral descriptors** — `spectralCentroid`, `spectralCrest`, `spectralDecrease`, `spectralEntropy`, `spectralFlatness`, `spectralFlux`, `spectralKurtosis`, `spectralRolloffPoint`, `spectralSkewness`, `spectralSlope`, and `spectralSpread`

- **Periodicity** — `harmonicRatio` and `pitch`
- **Cepstrum** — `gtcc` and `mfcc`

Audio Plugins: Support deep learning networks in Macintosh and Linux platforms and generate VST 3 binaries

This release enhances the `audioPluginConfig` object and the `generateAudioPlugin` function. Users can now validate and generate audio plugins that use deep learning pretrained networks in computational environments with different architectures.

- The `audioPluginConfig` object now supports the Intel® Math Kernel Library for Deep Neural Networks (Intel MKL-DNN) on Macintosh platforms.
- The `audioPluginConfig` object can now leverage the Intel DSP AVX2-FMA code replacement library on Macintosh and Linux® platforms.
- You can now use the `generateAudioPlugin` function to generate universal binaries for Intel and ARM® architectures on Macintosh platforms. Starting this release, the function also generates VST 3 audio plugin binaries.

Pretrained Networks: Feature embeddings extraction functions renamed

The `openl3Features` and `vggishFeatures` functions have been renamed to `openl3Embeddings` and `vggishEmbeddings`, respectively. The old functions and any scripts containing them continue to work.

Compatibility Considerations

The `openl3Features` and `vggishFeatures` functions will be removed in a future release. Use `openl3Embeddings` and `vggishEmbeddings` instead.

GPU code acceleration for audio augmentation and feature extraction

- The `audioDataAugmenter` object now supports `gpuArray` (Parallel Computing Toolbox) objects for code acceleration.
- The `pitch` function now supports `gpuArray` objects for code acceleration. This release also introduces GPU acceleration for the `pitch` property of `audioFeatureExtractor`.

You must have Parallel Computing Toolbox to use `gpuArray` objects with supported functions. For more details, see Run MATLAB Functions on a GPU (Parallel Computing Toolbox). To see which GPUs are supported, see GPU Support by Release (Parallel Computing Toolbox).

Additional examples for AI, room acoustics, and electroacoustic modeling

Machine Learning and Deep Learning

- Train 3-D Sound Event Localization and Detection (SELD) Using Deep Learning
- 3-D Sound Event Localization and Detection Using Trained Recurrent Convolutional Neural Network

- Feature Selection for Audio Classification
- Transfer Learning with Pretrained Audio Networks in Deep Network Designer
- Investigate Audio Classifications Using Deep Learning Interpretability Techniques
- Import Audacity Labels to Signal Labeler
- Speech Command Recognition on Raspberry Pi Using Simulink
- Speech Command Recognition Code Generation with Intel MKL-DNN Using Simulink

Room Acoustics

- Room Impulse Response Simulation with the Image-Source Method and HRTF Interpolation
- Room Impulse Response Simulation with Stochastic Ray Tracing

Electroacoustic Modeling

- Loudspeaker Modeling with Simscape™

Functionality being removed or changed

Audio Labeler will be removed
Warns

The **Audio Labeler** app will be removed in a future release. Use **Signal Labeler** instead.

The **Signal Labeler** app:

- Replaces file-level labels with attribute labels to define full-signal characteristics.
- Uses different line colors for channels of an audio file by default.
- Does not expose label tags of label definitions. You cannot interactively see, add, or edit label tags for label definitions.
- Does not automatically create label definitions when running automation algorithms. To automate the detection of speech content, you must first add logical region-of-interest (ROI) label definitions. To perform speech-to-text transcription, you must first add string ROI label definitions.
- Requires automation algorithms be run on one channel at a time. For multichannel audio files, you can choose which channel to use as input.
- Does not provide audio recording.

openl3Features and vggishFeatures will be removed
Still runs

The `openl3Features` and `vggishFeatures` functions will be removed in a future release. Use `openl3Embeddings` and `vggishEmbeddings` instead.

setExtractorParams will be removed
Still runs

The `setExtractorParams` function will be removed in a future release. Use `setExtractorParameters` instead.

fdesign.parameq will be removed
Warns

The `fdesign.parameq` filter specification object will be removed in a future release. Existing instances of the object continue to run. For new code, use the `designParamEQ` function instead. For more details on how to replace your existing code, see the **Compatibility Considerations** section on the `fdesign.parameq` object reference page.

visualize object function outputs figure handles
Behavior change

In previous releases, calling the `visualize` function of the dynamic range control System objects with an output argument returned static characteristic data. Starting in R2022a, that syntax returns a figure handle instead. Use the new `staticCharacteristic` function to compute static characteristics.

R2021b

Version: 3.1

New Features

Bug Fixes

Compatibility Considerations

Audio Playback Controls in Signal Labeler: Listen to audio signals while annotating them interactively

The **Signal Labeler** app has been enhanced with new audio functionality that plays back audio signals imported from audio files and folders.

Code generation support for audio plugins that use deep learning networks or code replacement libraries

This release introduces the `audioPluginConfig` object.

- The `audioPluginConfig` object enables users to use `generateAudioPlugin` and `validateAudioPlugin` to validate and generate audio plugins that use deep learning pretrained networks. The generated code can either leverage the Intel Math Kernel Library for Deep Neural Networks (Intel MKL-DNN) or use no third-party libraries.
- The `audioPluginConfig` object also lets users pass code replacement libraries to `generateAudioPlugin`. The generated code can leverage the Intel AVX and DSP AVX2-FMA code replacement libraries on Windows platforms.

You must have MATLAB Coder™ Interface for Deep Learning Libraries to use deep learning pretrained networks. You must have Embedded Coder® to use code replacement libraries. For more information, see Prerequisites for Deep Learning with MATLAB Coder (MATLAB Coder).

Pretrained i-Vector System: Apply and customize pretrained speaker verification

This release introduces the `speakerRecognition` function, which returns a pretrained speaker recognition system. You can modify the system to suit your application.

Enhanced ivectorSystem training, evaluation, and inference

This release enhances the `ivectorSystem` object to speed up training and evaluation using parallel computing and GPUs. The `trainExtractor`, `trainClassifier`, `enroll`, `detectionErrorTradeoff`, and `ivector` functions now include `ExecutionEnvironment` and `DispatchInBackground` options. Use `ExecutionEnvironment` to specify whether to execute on a CPU, GPU, multiple GPUs, or a remote parallel environment. Set `DispatchInBackground` to `true` to perform prefetch queueing. This feature requires a Parallel Computing Toolbox license.

The i-vector system inference functions `verify` and `identify` have been enhanced to include default values for scorers and corresponding decision thresholds.

Sound Classifier, YAMNet, and YAMNet Preprocess Blocks: Classify sound signals with deep learning using Simulink

This release introduces three blocks that enable users to perform deep learning sound classification in Simulink. The blocks combine signal preprocessing and network inference in one convenient package.

- Sound Classifier combines signal preprocessing and network inference for sound classification.

- YAMNet implements a pretrained neural network that predicts 521 audio event classes. The network can be used for classification and transfer learning.
- YAMNet Preprocess generates mel spectrograms that can be fed to YAMNet or to a network that accepts the same inputs as YAMNet and performs tasks such as transfer learning.

Multiband Parametric EQ Library Block: Design a multiband parametric equalizer in Simulink

This release adds multiband parametric EQ functionality to Simulink. The Multiband Parametric EQ block enables users to perform multiband parametric equalization, with functionality equivalent to the `multibandParametricEQ` System object. It is possible to add lowshelf and highshelf filters, as well as highpass (low cut) and lowpass (high cut) filters.

Compatibility Considerations

The Parametric EQ block has been renamed to Single-Band Parametric EQ.

Gammatone Filter Bank Block: Model the human auditory system in Simulink

Gammatone filter banks have applications in auditory modeling, psychoacoustics, and deep learning and machine learning for speech. This release enhances the Audio Toolbox Simulink library by including Gammatone Filter Bank, a block with functionality equivalent to the `gammatoneFilterBank` System object.

GPU acceleration for preprocessing inputs of pretrained deep learning models, speech detection, and feature extraction

- The `yamnetPreprocess`, `vggishPreprocess`, `crepePreprocess`, and `openl3Preprocess` functions now support `gpuArray` (Parallel Computing Toolbox) objects for code acceleration.
- The `detectSpeech` and `harmonicRatio` functions now support `gpuArray` (Parallel Computing Toolbox) objects for code acceleration. This release also introduces `gpuArray` support for the `harmonicRatio` property of audioFeatureExtractor.

You must have Parallel Computing Toolbox to use `gpuArray` objects with supported functions. For more details, see Run MATLAB Functions on a GPU (Parallel Computing Toolbox). To see which GPUs are supported, see GPU Support by Release (Parallel Computing Toolbox).

MATLAB Online support for Audio Toolbox

Starting this release, Audio Toolbox is supported in MATLAB Online™.

Audio Datastores: Specify FileSet objects as data locations

Starting this release, you can use `FileSet` objects to provide file locations for `audioDatastore` objects. `FileSet` objects provide increased performance compared to file paths or `DsFileSet` objects. For more information, see `matlab.io.datastore.FileSet`.

Additional examples for deep learning and deployment

This release introduces examples that use Audio Toolbox functionality to perform deep learning tasks and deploy deep learning audio systems to hardware.

- End-to-End Deep Speech Separation
- Acoustics-Based Machine Fault Recognition
- Acoustics-Based Machine Fault Recognition Code Generation with Intel MKL-DNN
- Acoustics-Based Machine Fault Recognition Code Generation on Raspberry Pi
- Accelerate Audio Deep Learning Using GPU-Based Feature Extraction

Functions being removed

fdesign.parameq will be removed
Still runs

The `fdesign.parameq` function will be removed in a future release. Existing instances of the function continue to run. For new code, use the `designParamEQ` function instead.

R2021a

Version: 3.0

New Features

Bug Fixes

OpenL3 Pretrained Network: Extract deep audio embeddings with pretrained OpenL3 convolutional neural network

`openl3` produces audio embeddings that can be used as input to deep learning classification networks or general machine learning systems. The layers of `openl3` can also be incorporated as part of a larger network.

- Use the `openl3` function to return the pretrained neural network.
- Use the `openl3Preprocess` function to preprocess the audio signal and generate the input to the network.
- Use the `openl3Features` function to combine preprocessing and network inference to generate deep audio embeddings.

The `openl3` and `openl3Features` functions require Deep Learning Toolbox.

CREPE: Deep learning pitch estimation

This release introduces the CREPE convolutional deep learning neural network and associated pre and postprocessing functions. Depending on the setup, CREPE can be integrated into a larger deep learning model or used as a feature extractor.

- Use the `crepe` function to return the pretrained neural network.
- Use the `crepePreprocess` function to preprocess the audio signal into a format acceptable by `crepe`.
- Use the `crepePostprocess` function to postprocess the output of the CREPE network and convert it to a pitch estimate.
- Use the `pitchnn` function to incorporate audio pre and postprocessing with network inference and produce the pitch estimation.

The `crepe` and `pitchnn` functions require Deep Learning Toolbox.

ivectorSystem: Produce compact representations of audio signals

Use `ivectorSystem` object to produce compact representations of audio signals for applications such as speaker verification, speaker identification, speaker diarization, speech emotion recognition, and acoustic machine fault detection.

The `ivectorSystem` object enables a streamlined workflow including system training, label enrolling and unenrolling, label verification, label identification, as well as open-set classification threshold determination.

vggishPreprocess and yamnetPreprocess

This release introduces functions that preprocess audio signals for classification tasks using the VGGish (`vggish`) or YAMNet (`yamnet`) pretrained neural networks. With the `vggishPreprocess` and `yamnetPreprocess` functions, users can process large audio sets through the networks more efficiently.

Enhanced audio workflow within Signal Labeler

The **Signal Labeler** app has been enhanced with new audio functionality that enables data importing from audio files and folders.

acousticRoughness: Measure perceived roughness of an acoustic signal

Apply the `acousticRoughness` function for perceptual sound quality analysis or noise vibration harshness evaluation.

Using algorithms based on the Zwicker loudness method (ISO 532-1), the `acousticRoughness` function quantifies perceived amplitude modulations (between approximately 10 Hz and 400 Hz) due to amplitude or frequency modulations in the audio signal.

The `acousticRoughness` function returns roughness in asper units and specific roughness in asper/Bark.

Octave Filter Bank Block: Octave and fractional-octave filter bank

The Octave Filter Bank Simulink block simplifies the design and visualization of ANSI S1.11-2204 compliant filter banks. The Octave Filter Bank block maintains the properties of the `octaveFilterBank` object, with additional enhancements enabling individual frequency band access, as well as specifying or inheriting sample rate.

Sidechain input capability for dynamic range objects and blocks

The Sidechain input lets you use one audio input signal to provide dynamic range compression or expansion of a second, separate audio signal. Use the Sidechain input for channel-linked stereo audio processing, ducking, and de-essing.

The Sidechain input applies to four objects and their associated blocks:

- The `compressor` object and Compressor block
- The `expander` object and Expander block
- The `limiter` object and Limiter block
- The `noiseGate` object and Noise Gate block

GPU code acceleration for audioFeatureExtractor object

The `audioFeatureExtractor` object now supports `gpuArray` objects.

You must have Parallel Computing Toolbox to use `gpuArray` objects with supported functions. See Run MATLAB Functions on a GPU for more details and GPU Support by Release to see which GPUs are supported.

New rectification option for cepstralCoefficients

The `cepstralCoefficients` function now allows users to implement custom rectification outside of the function.

- `'log'` — Performs logarithmic `'Rectification'`.
- `'cubic-root'` — Performs cubic-root `'Rectification'`.
- `'none'` — No `'Rectification'` is carried out. This option enables you to implement your own rectification algorithms outside of the function.

Extended JUCE project support

`generateAudioPlugin` function can now generate C/C++ source code and JUCE project files suitable for use with JUCE 5.3.2 to 6.0.1. This functionality requires a MATLAB Coder license.

Set nondefault time stretching and pitch shifting parameters using audioDataAugmenter

`audioDataAugmenter` has been enhanced with two new object functions allowing default parameter value modification of time stretching and pitch shifting augmentation algorithms.

Additional examples for deep learning and psychoacoustics

Design and Use Deep Learning Systems

- Speaker Recognition Using x-vectors
- Speaker Diarization Using x-vectors
- Train Spoken Digit Recognition Network Using Out-of-Memory Audio Data
- Train Spoken Digit Recognition Network Using Out-of-Memory Features
- Speaker Identification Using Custom SincNet Layer and Deep Learning
- Dereverberate Speech Using Deep Learning Networks
- Speech Command Recognition in Simulink

Deploy Deep Learning Systems

- Keyword Spotting in Noise Code Generation with Intel MKL-DNN
- Keyword Spotting in Noise Code Generation on Raspberry Pi

Psychoacoustics

- Effect of Soundproofing on Perceived Noise Levels

R2020b

Version: 2.3

New Features

Bug Fixes

Compatibility Considerations

YAMNet Pretrained Network: Classify sounds with pretrained YAMNet neural network

Use the `classifySound` function to preprocess, perform sound classification using YAMNet, and then postprocess the network outputs. You can specify smoothing and thresholding parameters for sound classification, and analyze the output by viewing the most likely sounds for a whole signal, or a table of detected sounds and the corresponding decision confidence. YAMNet is a pretrained neural network that predicts 521 audio event classes.

Use the `yamnet` function to interact with the network object directly. The network is returned as a `SeriesNetwork` object. You can use this pretrained model for classification and transfer learning.

Use the `yamnetGraph` function to explore the audio event class ontology.

The YAMNet functionality requires Deep Learning Toolbox.

VGGish Pretrained Network: Extract audio embeddings with pretrained VGGish neural network

Use the `vggishFeatures` function to extract semantically meaningful 128-dimensional feature vectors (embeddings). You can use the feature vectors as input for a classification model.

Use the `vggish` function to interact with the network object directly. The network is returned as a `SeriesNetwork` object. You can use this pretrained model for feature extraction and transfer learning.

The VGGish functionality requires Deep Learning Toolbox.

Extract cepstral coefficients from spectrograms and auditory spectrograms

Use the `cepstralCoefficients` function to extract cepstral coefficients from spectrograms and auditory spectrograms. The `cepstralCoefficients` function supports streaming and nonstreaming audio signals.

You can now use modular functions such as `stft`, `designAuditoryFilterBank`, `cepstralCoefficients`, and `audioDelta` in combination for efficient extraction of audio features such as mel frequency cepstral coefficients (MFCC), gammatone frequency cepstral coefficients (GTCC), and Bark frequency cepstral coefficients (BFCC). You can also use these modular functions to explore, extend, or modify default implementations.

Compute delta of audio features

Use the `audioDelta` function to compute the delta of any audio feature. The `audioDelta` function supports streaming and nonstreaming audio signals.

You can now use modular functions such as `stft`, `designAuditoryFilterBank`, `cepstralCoefficients`, and `audioDelta` in combination for efficient extraction of audio features such as mel frequency cepstral coefficients (MFCC), gammatone frequency cepstral coefficients (GTCC), and Bark frequency cepstral coefficients (BFCC). You can also use these modular functions to explore, extend, or modify default implementations.

Enhanced control of designAuditoryFilterBank

The designAuditoryFilterBank function now includes parameters to enable the following functionality:

Parameter	New Functionality
'OneSided'	Specify whether the filter bank returned is one-sided or two-sided. Depending on your processing pipeline, using a two-sided filter bank may increase speed.
'FilterBankDesignDomain'	Specify whether the filter shapes are designed in the linear (Hz) or warped (mel or Bark) domain. This parameter is only applicable when the frequency scale is mel or Bark.

Enhanced control of audioFeatureExtractor and Extract Audio Features

The audioFeatureExtractor object and the **Extract Audio Features** task now include parameters to enable the following functionality:

audioFeatureExtractor Parameter	Extract Audio Features UI label	New Functionality
WindowNormalization	Window normalization	Specify whether to normalize a spectrum by the time-domain window energy. This functionality applies when using or extracting a linear, mel, Bark, or ERB spectrum.
FilterBankDesignDomain	Filter bank domain	Specify whether the filter shapes are designed in the linear (Hz) or warped (mel or Bark) domain. This functionality applies when using or extracting a mel or Bark spectrum.

Enhanced control of melSpectrogram

The melSpectrogram function now includes parameters to enable the following functionality:

Parameter	New Functionality
'Window'	Specify the desired time-domain window to apply prior to the discrete Fourier transform (DFT).
'WindowNormalization'	Specify whether or not to normalize the spectrum by the window energy.

Parameter	New Functionality
`'FilterBankNormalization'`	Specify the type of normalization applied to each filter in the filter bank. You can choose to ignore normalization, or normalize by the individual filter bandwidth or area.

Enhanced control of time-domain windowing for mfcc and gtcc

The `mfcc` and `gtcc` functions include a new parameter, `Window`, that enables you to specify the window applied in the time domain.

Extract spectral flux from streaming signals

To extract spectral flux from streaming signals, you can now pass state in and out of the `spectralFlux` function.

Generate MATLAB function from audioFeatureExtractor

Use `generateMATLABFunction` to create a MATLAB function from the `audioFeatureExtractor` object. The generated MATLAB function performs equivalent audio feature extraction as the object, and is compatible with C/C++ code generation.

You can also generate a MATLAB function that is optimized for frame-based, streaming audio signals. When you generate a MATLAB function for stream processing, required states are maintained by the function.

GPU code acceleration for new and existing features

The following new features support `gpuArray` objects:

- `audioDelta`
- `cepstralCoefficients`

The following existing features now support `gpuArray` objects:

- `stretchAudio`
- `shiftPitch`
- `spectralCentroid`
- `spectralCrest`
- `spectralDecrease`
- `spectralEntropy`
- `spectralFlatness`
- `spectralFlux`
- `spectralKurtosis`
- `spectralRolloffPoint`
- `spectralSkewness`

- `spectralSlope`
- `spectralSpread`

You must have Parallel Computing Toolbox to use `gpuArray` objects with supported functions. See Run MATLAB Functions on a GPU for more details and GPU Support by Release to see which GPUs are supported.

GPU code generation support for the melSpectrogram function

The `melSpectrogram` function now supports code generation for graphical processing units (GPUs). If you have GPU Coder™, you can generate optimized CUDA® code from the `melSpectrogram` function for machine learning and deep learning systems.

Measure perceived acoustic fluctuation strength

Use `acousticFluctuation` to measure perceived fluctuation in accordance with the Zwicker model of fluctuation and ISO 532-1:2017(E) (the Zwicker loudness method). The `acousticFluctuation` function returns fluctuation in vacil and specific fluctuation in vacil/Bark.

Enhanced control over sound pressure level (SPL) metering

You can now use the `FrequencyRange` property of the `splMeter` object to specify the frequency range to analyze.

Improved low-frequency and high-frequency octave filtering using octaveFilter

The `octaveFilter` object has been enhanced so that you can specify a larger range of center frequencies. This enhancement includes more accurate filtering at low and high frequencies.

Improved low-frequency octave filtering using octaveFilterBank

The `octaveFilterBank` object has been enhanced to provide more accurate filtering at low frequencies.

Enhanced Audio Test Bench workflow

The **Audio Test Bench** app has been enhanced to maintain settings applied to scopes between sessions.

Report audio plugin latency to host

Use the `setLatencyInSamples` method in your `audioPlugin` class definition to report latency to your host application. Typically, digital audio workstations (DAWs) use this information to compensate for algorithmic latency and align tracks.

Refresh audio device list from audio I/O blocks

The Audio Device Reader block and the Audio Device Writer block now include the ability to refresh a list of available audio devices. This functionality enables you to add and remove audio devices during a single MATLAB session.

Additional examples for deep learning

Deploy Deep Learning Systems

- Speech Command Recognition Code Generation on Raspberry Pi
- Speech Command Recognition Code Generation with Intel MKL-DNN

Functionality being removed or changed

Specify window for mfcc, gtcc, and melSpectrogram functions
Behavior change in future release

The `WindowLength` parameter will be removed from `mfcc`, `gtcc`, and `melSpectrogram` in a future release. Use the `Window` parameter instead.

For example, in releases prior to R2020b, you could only specify the length of the time-domain window. The window was always designed as a periodic Hamming window. You can replace instances of the following code:

```
a = mfcc(audioIn,fs,'WindowLength',1024);
b = gtcc(audioIn,fs,'WindowLength',1024);
c = melSpectrogram(audioIn,fs,'WindowLength',1024);
```

With this code:

```
a = mfcc(audioIn,fs,'Window',hamming(1024,'periodic'));
b = gtcc(audioIn,fs,'Window',hamming(1024,'periodic'));
c = melSpectrogram(audioIn,fs,'Window',hamming(1024,'periodic'));
```

Delta computation for mfcc, gtcc, and audioFeatureExtractor
Behavior change

The delta and delta-delta calculations are now computed using the `audioDelta` function, which has a different startup behavior than the previous algorithm. The default `DeltaWindowLength` has changed from 2 to 9. A delta window length of 2 is no longer supported.

Window normalization parameter renamed for audioFeatureExtractor
Behavior change in future release

The `FilterBankNormalization` parameter replaces the `Normalization` parameter when using or extracting the `melSpectrum`, `barkSpectrum`, or `erbSpectrum`. The `Normalization` parameter will be removed in a future release.

For example, replace instances of the following code:

```
afe = audioFeatureExtractor('melSpectrum',true,'barkSpectrum',true,'erbSpectrum',true);
setExtractorParams(afe,'melSpectrum','Normalization','none')
setExtractorParams(afe,'barkSpectrum','Normalization','none')
setExtractorParams(afe,'erbSpectrum','Normalization','none')
```

With this code:

```
afe = audioFeatureExtractor('melSpectrum',true,'barkSpectrum',true,'erbSpectrum',true);
setExtractorParams(afe,'melSpectrum','FilterBankNormalization','none')
setExtractorParams(afe,'barkSpectrum','FilterBankNormalization','none')
setExtractorParams(afe,'erbSpectrum','FilterBankNormalization','none')
```

SOS returned instead of FOS from octaveFilterBank
Behavior change

The `coeffs` function of `octaveFilterBank` now returns the filter in second-order sections (SOS) instead of fourth-order sections (FOS). This new format reflects an updated internal representation, which has been enhanced to remain stable at very low frequencies.

cepstralFeatureExtractor will be removed
Still runs

`cepstralFeatureExtractor` will be removed. Use the `mfcc` and `gtcc` functions to compute the same features for batch signals. For time-critical processing, use a combination of `designAuditoryFilterBank`, `cepstralCoefficients`, and `audioDelta` to compute the same features. If you are extracting multiple audio features, use the `audioFeatureExtractor`.

designAuditoryFilterBank scaling changed for ERB filter banks
Behavior change

The half-sided ERB filter bank returned from `designAuditoryFilterBank` is now scaled by 2. This change provides consistent results when applying one-sided or two-sided filtering, without requiring multiplications in the processing loop.

R2020a

Version: 2.2

New Features

Measure perceived loudness according to ISO 532-1 or ISO 532-2

Use `acousticLoudness` to measure perceived loudness in accordance with ISO 532-1:2017(E) (the Zwicker method) and ISO 532-2:2017(E) (the Moore-Glasberg method). Both methods can return loudness in sone and specific loudness. When using the Zwicker method, you can also measure the percentile loudness of time-varying signals, as described in ISO 532-1:2017(E).

Measure perceived sharpness according to DIN 45692

Use `acousticSharpness` to measure perceived sharpness of an acoustic signal, as specified by the ISO 532-1:2017(E) and DIN 45692 standards.

Determine calibration factor for microphone

Use `calibrateMicrophone` to determine a calibration factor for your audio input system.

Convert between acoustic loudness units phon and sone

Audio Toolbox now includes conversion functions for loudness units phon and sone.

- `phon2sone` –– Convert from phon to sone
- `sone2phon` –– Convert from sone to phon

Detect boundaries of speech in audio

Use `detectSpeech` to determine the start and end indices of regions of speech in audio.

Speech detection is a common preprocessing step for machine learning and deep learning workflows. See Classify Gender Using LSTM Networks and Keyword Spotting in Noise Using MFCC and LSTM Networks for examples.

Streamline audio feature extraction in the Live Editor

Use the **Extract Audio Features** task to configure an optimized feature extraction pipeline by selecting features and parameters graphically. You can reuse the output from the **Extract Audio Features** task to apply feature extraction to entire data sets.

GPU code generation support for the mfcc function

The `mfcc` function now supports code generation for graphical processing units (GPUs). If you have GPU Coder, you can generate optimized CUDA code from the `mfcc` function for machine learning and deep learning systems.

Audio Datastore: Write data from audio datastore using writeall

You can now write data from an `audioDatastore` object to files on disk using the `writeall` function.

GPU code acceleration for mfcc and melSpectrogram functions

The `mfcc` and `melSpectrogram` functions now support `gpuArray` objects. You must have Parallel Computing Toolbox to use `gpuArray` objects with supported functions. See Run MATLAB Functions on a GPU for more details and GPU Support by Release to see which GPUs are supported.

Text-to-speech conversion using third-party speech API

To perform text-to-speech conversion in MATLAB, use the `text2speech` function available on File Exchange. The function enables you to interface third-party text-to-speech 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.

Cubic root rectification for MFCC and GTCC calculations

You can now choose between cubic-root and log rectification when extracting mel frequency cepstral coefficients and gammatone cepstral coefficients. This enhancement applies to the `mfcc` and `gtcc` functions, the `cepstralFeatureExtractor` and `audioFeatureExtractor` objects, and the Cepstral Feature Extractor block.

Enhanced look and feel for Audio Test Bench

The **Audio Test Bench** app has been enhanced with a modern look and feel. The app now supports custom UI layouts and parameter styles defined in your plugin interface.

Enhanced visualization for loaded plugins

The `parameterTuner` function now renders parameter-specific widgets for plugins loaded using `loadAudioPlugin`.

Generate standalone executable from audio plugin

You can now use the -exe option of the generateAudioPlugin function to generate a binary standalone executable from an audio plugin. When you evaluate the generated code, the UI you defined in your audio plugin opens. You can tune parameters and control the input and output of the plugin using the UI.

Generate sine, square, and sawtooth waveforms in Simulink

Use the Audio Oscillator block in Simulink to generate tunable sine, square, and sawtooth waveforms. You can tune parameters of the waveforms, such as frequency, amplitude, and DC offset, using input ports or a UI.

Generate periodic signal from single-cycle waveforms in Simulink

Use the Wavetable Synthesizer block in Simulink to generate a periodic signal from a single-cycle waveform you define. You can tune parameters of the signal, such as frequency, amplitude, and DC offset, using input ports of a UI.

Additional input ports for Audio Toolbox blocks

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

Block	New Optional Input Ports
Reverberator	Pre-delay (s), Highcut frequency (Hz), Diffusion, Decay factor, High frequency damping, Wet/dry mix
Voice Activity Detector	silence-to-speech probability, speech-to-silence probability
Crossover Filter	Crossover frequency (Hz), Crossover order

Additional examples for deep learning and machine learning

- Train Generative Adversarial Network (GAN) for Sound Synthesis
- Speaker Verification Using Gaussian Mixture Model
- Speaker Verification Using i-Vectors

R2019b

Version: 2.1

New Features

AU Plugin Generation: Generate AU plugins for macOS

You can now generate an Audio Unit (AU) v2 audio plugin binary using `generateAudioPlugin`.

Custom Plugin UI: Generate VST and AU plugins with custom UIs

Define custom user interfaces (UIs) using the updated `audioPluginInterface` and `audioPluginParameter` functions, and the new `audioPluginGridLayout` function. The custom UI is visible once the plugin is generated or when using `parameterTuner` in MATLAB. Customization abilities include:

- Define parameter controls that render as knobs, sliders, rocker switches, toggle switches, check boxes, or drop-downs.
- Position controls and labels on a grid layout.
- Define custom background color, background image, or both.
- Define custom images for controls (generated plugins only).

See Design User Interface for Audio Plugin for more information.

Enhanced Parameter Tuner UI

`parameterTuner` now renders parameters as knobs, sliders, rocker switches, toggle switches, check boxes, or drop-downs when used with objects that inherit from `audioPlugin`. You can also define a custom background color, background image, or both. Custom images for controls are not supported by `parameterTuner`.

Audio Data Augmentation: Enlarge your dataset using audio-specific augmentation

`audioDataAugmenter` enables you to enlarge your audio dataset using audio-specific augmentation techniques like pitch shifting, time-scale modification, time shifting, noise addition, and volume control. You can create a cascaded augmentation pipeline to apply multiple algorithms probabilistically, or a parallel augmentation pipeline to apply algorithms deterministically. You can also add custom augmentation algorithms to `audioDataAugmenter`.

See Keyword Spotting in Noise Using MFCC and LSTM Networks for an example.

Audio Feature Extraction: Streamline audio feature extraction

Use `audioFeatureExtractor` to extract multiple audio features using an efficient processing pipeline. `audioFeatureExtractor` encapsulates the extraction pipeline so that your code is cleaner and more modular.

See Sequential Feature Selection for Speech Emotion Recognition and Classify Gender Using LSTM Networks for examples.

Pitch shifting: Increase or decrease the pitch of audio signals

Use `shiftPitch` to increase or decrease the pitch of an audio signal by a given number of semitones. To achieve better fidelity with the original audio, you can optionally apply phase locking and formant preservation.

Time stretching: Stretch the time scale of audio signals

Use `stretchAudio` to apply time-scale modification (TSM) to an audio signal. You can speed up or slow down audio while preserving the original pitch.

For streaming applications, use `audioTimeScaler` to apply TSM. The `audioTimeScaler` enables you to tune the speedup factor while streaming.

Auditory Filter Banks: Design common frequency-domain auditory filter banks

Use `designAuditoryFilterBank` to design a mel, Bark, or ERB filter bank. You can use the filter bank to apply computationally efficient frequency-domain filtering.

Enhanced MFCC extraction

The `mfcc` function now accepts frequency-domain input so that you can reuse your DFT computation.

The `mfcc` function now enables you to specify the bandedges of the filter bank. You can use this ability to fine-tune the MFCC feature extraction or mimic other implementations.

Enhanced GTCC extraction

The `gtcc` function now accepts frequency-domain input so that you can reuse your DFT computation.

Audio Labeler App: Automatically label regions of speech

The **Audio Labeler** app now provides automatic labeling of detected regions of speech.

Audio Labeler App: Speech-to-text transcription using third-party speech API

The **Audio Labeler** app now supports the `speech2text` function available on File Exchange. Through the Audio Labeler app, you can interface with third-party speech-to-text APIs, including:

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

The `speech2text` entry on File Exchange includes a tutorial includes a tutorial to help get you started.

Pink Noise: Generate noise signals common to audio applications

Use `pinknoise` to generate a single channel or multiple independent channels of pink noise that is bounded between –1 and 1. The power spectral density of pink noise is inversely proportional to frequency and falls off at 10 dB/decade (3 dB/octave). Pink noise is commonly used to test and equalize loudspeakers and to mimic background noise encountered in real-world situations.

Tune reverberator parameters graphically

You can now tune parameters of the `reverberator` object graphically using `parameterTuner`.

Specify coefficient orientation output from designParamEQ, designShelvingEQ, and designVarSlopeFilter

`designParamEQ`, `designShelvingEQ`, and `designVarSlopeFilter` now enable you to specify the orientation of the returned filter coefficients. Specify the orientation as `'row'` for interoperability with **FVTool**, `dsp.DynamicFilterVisualizer`, and `dsp.FourthOrderSectionFilter`.

Visualize and analyze the filters designed by weightingFilter and octaveFilter

The `weightingFilter` object and the `octaveFilter` object now include additional filter analysis tools.

Additional examples for deep learning, active noise control, positional audio, and time-frequency masking

- Keyword Spotting in Noise Using MFCC and LSTM Networks
- Sequential Feature Selection for Speech Emotion Recognition
- Active Noise Control: From Modeling to Real-Time Prototyping
- Binaural Audio Rendering Using Head Tracking
- Time-Frequency Masking for Harmonic-Percussive Source Separation

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.