“MATLAB in the Loop” for Audio Signal Processing

Darel A. Linebarger, Ph.D.
Senior Manager, Signal Processing and Communications
MathWorks, Inc.

Introduction: Who am I and why am I here?

- **Why:** To demonstrate that you can **use MATLAB and your laptop to develop and test real time audio signal processing algorithms**
- **Who:**
  - I manage a development group at MathWorks focused on DSP and Communications
    - Includes fixed-point modeling and deployment to C or HDL
  - Audio is a focus area for DSP System Toolbox
- **What:**
  - I am on the road to channel customer input directly into development
  - I am seeking a few customers to work closely with and by helping them succeed, to make our tools better for everyone.
  - Could you be one of those customers?

Goals for today

- Help you use our tools better so that you are
  - More productive
  - More efficient
- Take your input for our product plans to help you with your workflow(s)
  - What new directions should we be considering?

- Initiate contact with key people or groups to help drive this area forward.

- NOTE: Most of today's presentation is also available as a webinar from DSP System Toolbox product page on mathworks.com

Where would you like us to invest next?
How can we best help you?

- What customers are saying: We want plugin support (autogenerate for deployment, hosting)
  - Which plugin formats? (Apple, VST, etc.)
- Other possible priorities:
  - Performance:
    - How many biquads can we run and maintain real-time?
    - Reduce latency in our processing chain?
  - Asynchronous sample rate conversion
  - More audio algorithms
    - Codecs? Recognition? Effects for music production?
  - More drivers or environments (OSC, JACK, JUCE, WASAPI, etc.)
  - Your good idea goes here …
- What would you suggest?

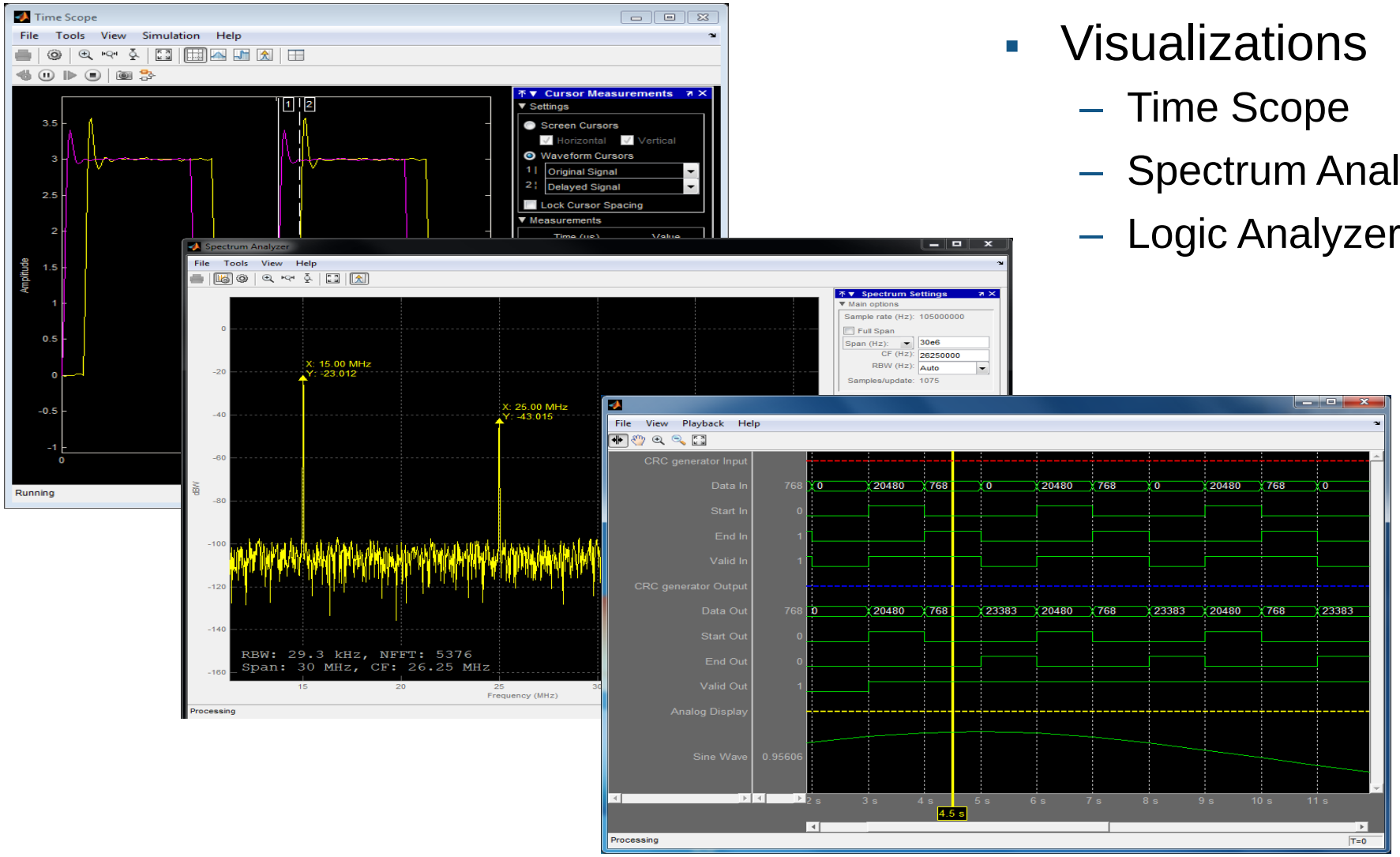
Start with demos

- Live audio to scopes and file
- Simple demo:
  - audioIn=dsp.AudioRecorder('SamplesPerFrame',1e5, 'NumChannels', 1)
  - sound(yin,44100)
  - audioFileOut=dsp.AudioFileWriter;
  - step(audioFileOut,yin);
  - release(audioFileOut);
- Parametric equalizer

I said “real time”. What did I really mean?

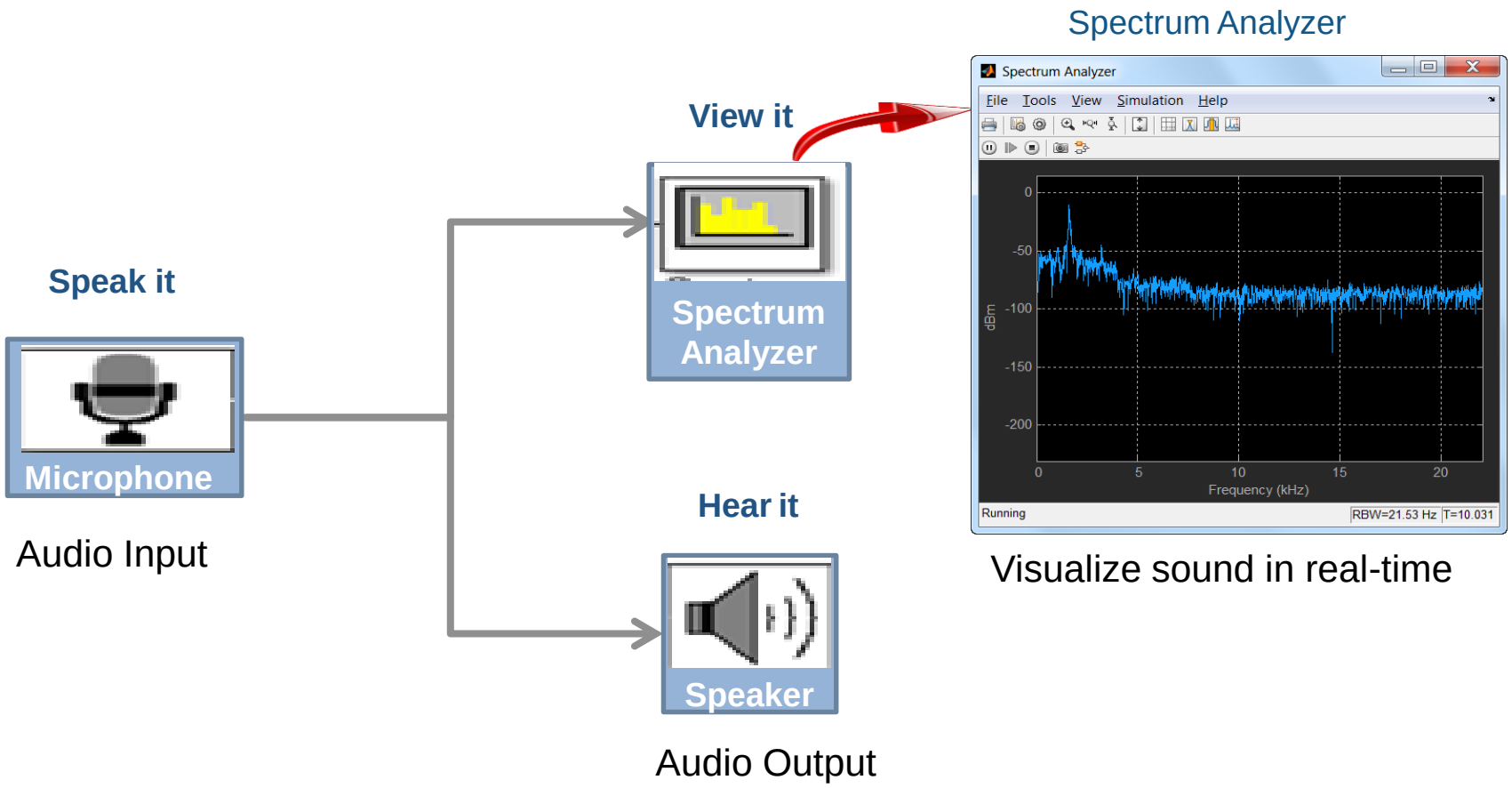
- A laptop does not provide a true real time environment. On the other hand, if it can process the data fast enough and reliably enough, it might work just fine.
  - E.g. We use our PCs for voice and video (Skype) communications frequently. That’s real time communications.
- For audio signal processing, real time is only important when either or both input and output are live audio.
  - Audio input comes from microphone, audio output goes to speakers or headphones.
- What about latency?
  - Not important if either input or output are not live. E.g. consider playing recorded music. As long as the latency is not ridiculous, users will not notice it.
  - If both input and output are live, then latency must be small (< 30 ms).
  - We have a shipping example in 14a demonstrating how to measure latency.

New scopes in DSP System Toolbox



- Visualizations
  - Time Scope
  - Spectrum Analyzer
  - Logic Analyzer

How to create a streaming test bench



How to create test bench in MATLAB

```
%% Create and Initialize
SamplesPerFrame = 1024;
Fs = 44100;

Microphone=dsp.AudioRecorder('SamplesPerFrame',SamplesPerFrame);

Spectra=dsp.SpectrumAnalyzer ('SampleRate',Fs);
```

```
%% Stream processing loop
tic;

while toc < 20
     % Read frame from microphone
       audioIn = step(Microphone);

     % View audio spectrum
       step(Spectra,audioIn);
end
```

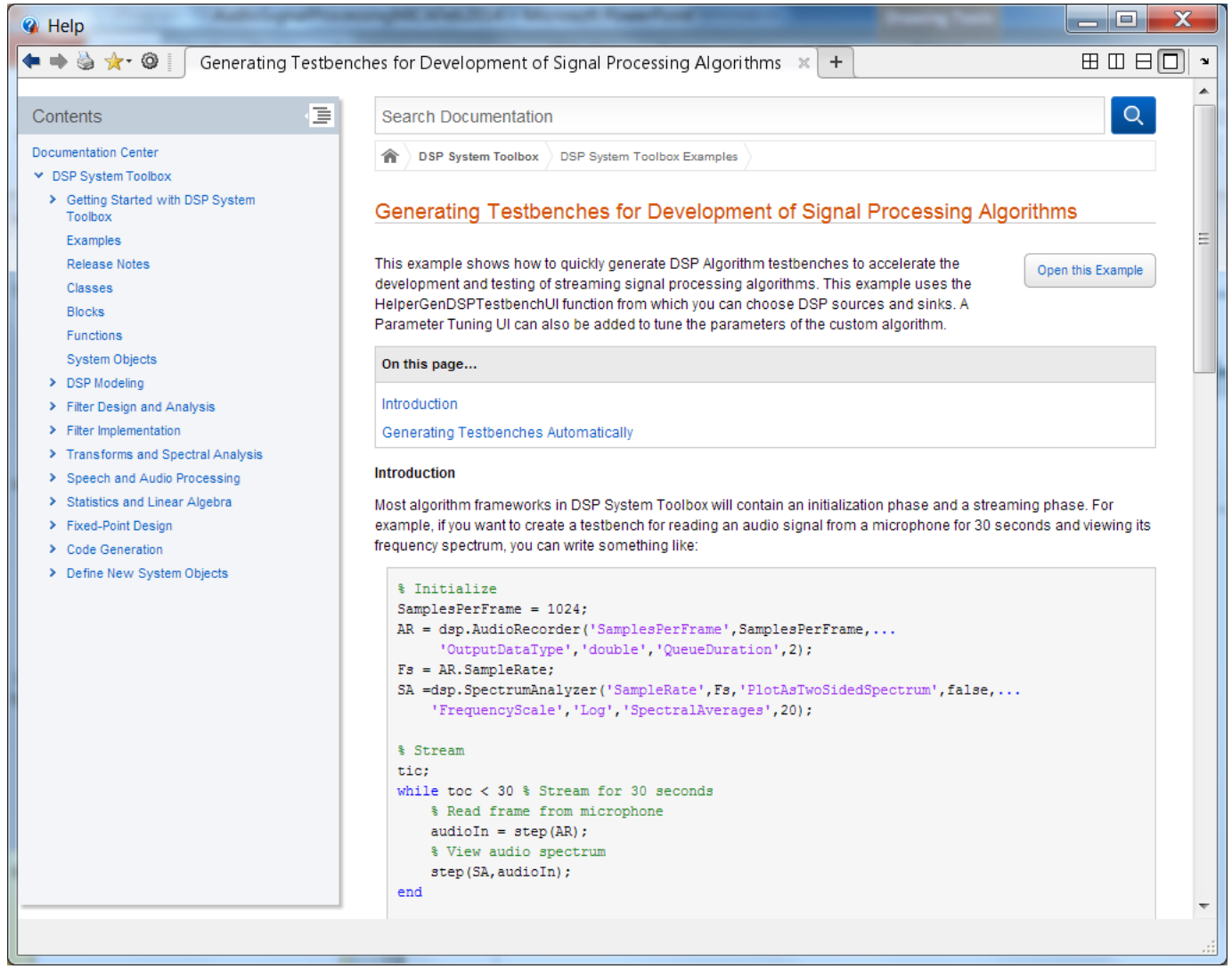
```
%% Terminate
release(Microphone)
release(Spectra)
```

Initialize

Process in-the-loop

Terminate

Use test bench App from in product example to create a test bench



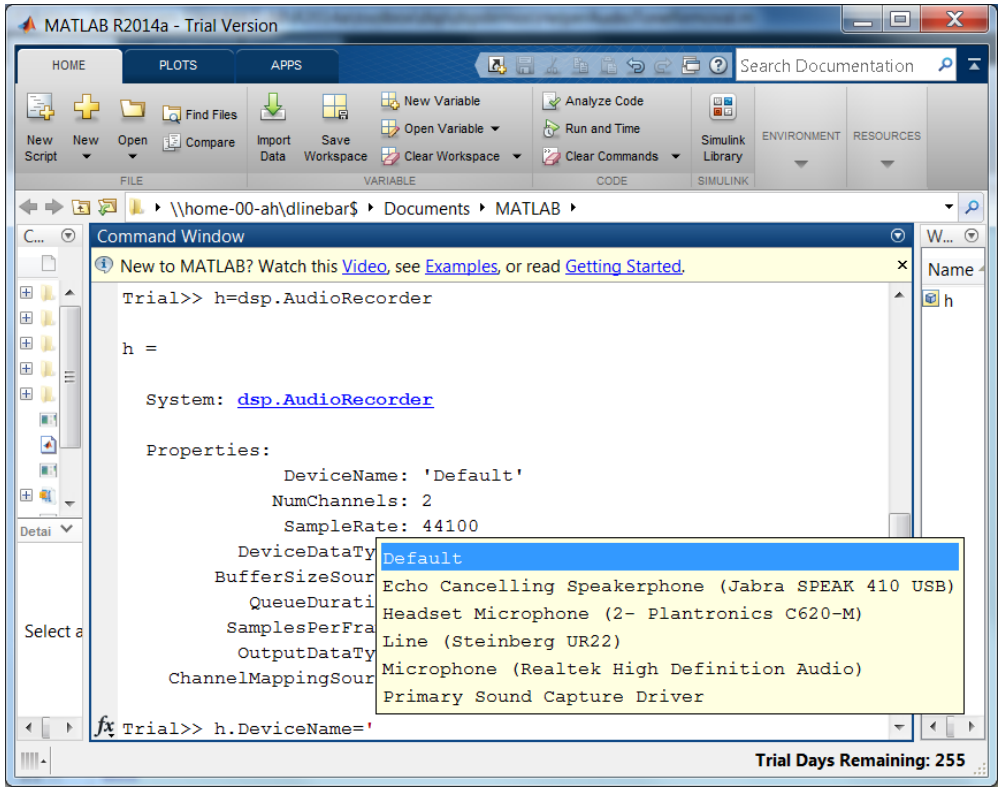
DSP System Toolbox audio related components (supported on Apple/Windows/Linux)

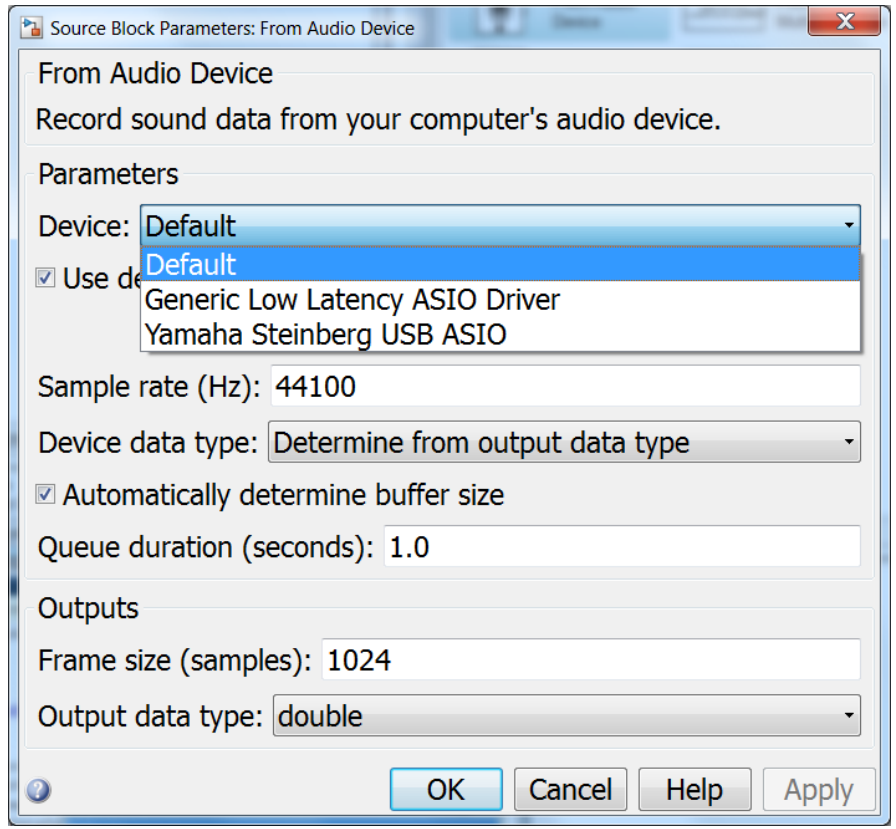
- **Multichannel audio I/O** (Number of channels depends on hardware)
  - Audio Player/Recorder - Supports multiple devices, one sound driver per MATLAB session
  - Audio File Reader/Writer
  - ASIO low latency driver support on Windows(R)
  - Custom channel mapping
- **Audio signal analysis**
  - Scopes: time, spectrum analyzer, array plot
  - Transfer function estimator
  - Measurements: Average power, PeaktoRMS ratio, mean, variance, ...
- **Signal processing algorithms**
  - FIR, Biquad, Multirate FIR, FFT, LMS, ...
  - Variable fractional delay (useful for audio beamforming)
- **Connectivity**
  - UDP, MIDI (simultaneous support for multiple controls on multiple devices)

Audio I/O with MATLAB: The gear

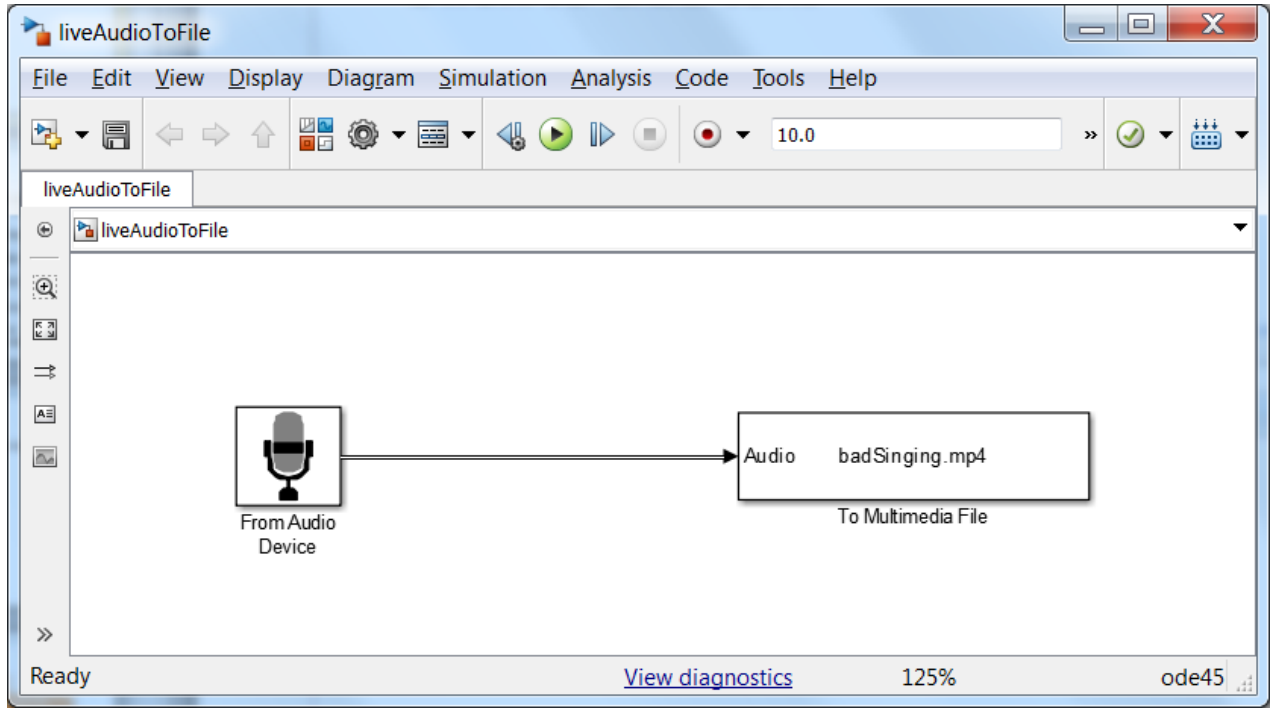
Audio Hardware is Automatically Detected

- Audio device I/O components (in both MATLAB and Simulink) detect audio devices registered with OS and dynamically populate pick lists





Choice of Modern File Formats Allows Interplay with other Common Audio Players



Audio demos

- Live feed into scopes and file write
- Sample rate conversion
- Parametric equalizer with run time interaction (real time on laptop)
- Auto generate code for audio test bench
- Fourier
- Reverb (uses ASIO)
- Plugin with Reaper

Optional additional topics

- Latency – measuring, minimizing, ASIO (for low latency on Windows)
- Filter design
  - Sample Rate Conversion
- Plugins – generating them from MATLAB Code
- Codecs – speech or audio
- Code generation for acceleration or deployment

Filter Design and Sample Rate Conversion

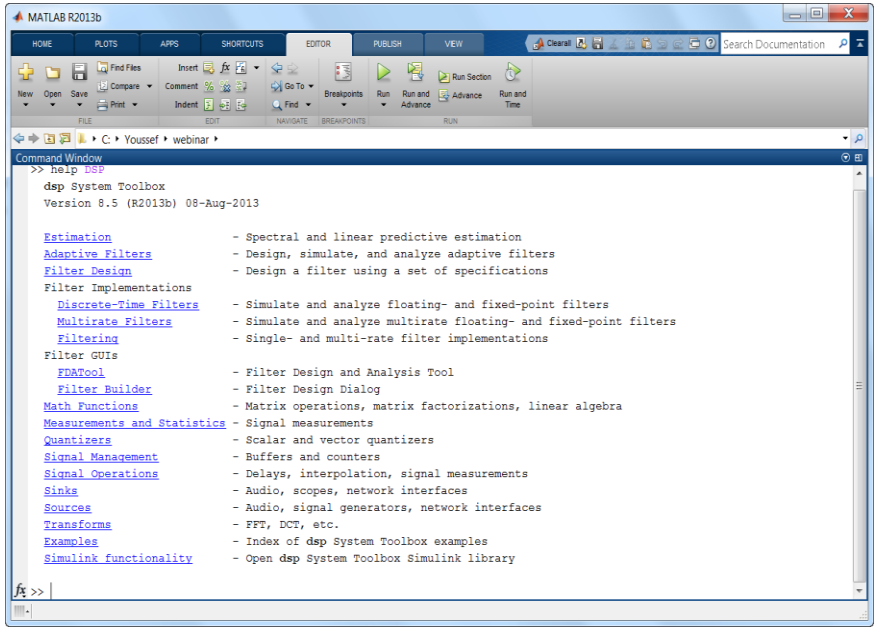
- filterbuilder
  - Generates MATLAB code for given design
  - Optionally generates HDL code
- Sample rate conversion
  - Baseline: FIR Decimation, FIR Interpolation and FIR Rate Converter
  - New design assist: dspdemo.SampleRateConverter (see associated demo in 14a)
- Preview of 14b sample rate converter
  - Allows “tolerance” to find smaller factors if approximate rate conversion acceptable
  - Can reduce number of operations and/or number of stages
- Is there interest in Asynchronous sample rate conversion?
  - What would you expect for interface?
- Demo

DSP System Toolbox *

Over 300 algorithms for modeling, designing, implementing and deploying dynamic system applications

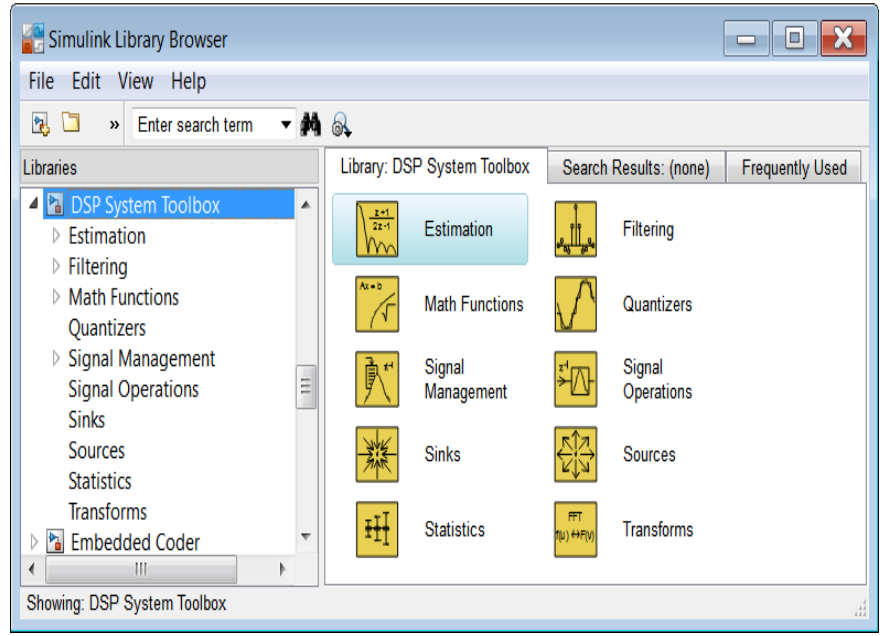
- Advanced Filter Design, Adaptive, Multistage and Multi-rate Filters
- FFT, DCT & other Transforms
- Signal processing blocks for Simulink
- Support for Fixed-Point, C/C++ code generation and HDL
- Visualization in Time and Frequency-domain
- System objects and functions in MATLAB
- Stream signal Processing
- ARM Cortex-M support for hardware prototype

Algorithm libraries in MATLAB



*http://www.mathworks.com/products/dsp-system/index.html

Algorithm libraries in Simulink



THANK YOU!

Where would you like us to invest next?
How can we best help you?

- What customers are saying: We want plugin support (autogenerate for deployment, hosting)
  - Which plugin formats? (Apple, VST, etc.)
- Other possible priorities:
  - Performance:
    - How many biquads can we run and maintain real-time?
    - Reduce latency in our processing chain?
  - Asynchronous sample rate conversion
  - More audio algorithms
    - Codecs? Recognition? Effects for music production?
  - More drivers or environments (OSC, JACK, JUCE, WASAPI, etc.)
  - Your good idea goes here …
- What would you suggest?

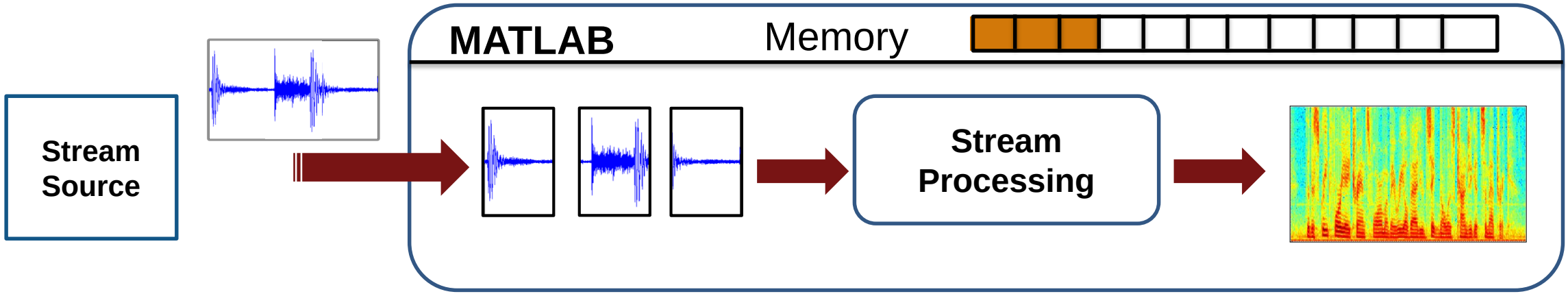
Agenda

- Tunable parametric equalizer example
- Dynamic range audio expander example

1. How to create a streaming test bench for audio processing in MATLAB
2. How to develop algorithms and incorporate them into the test bench
3. How to accelerate simulation for real-time performance

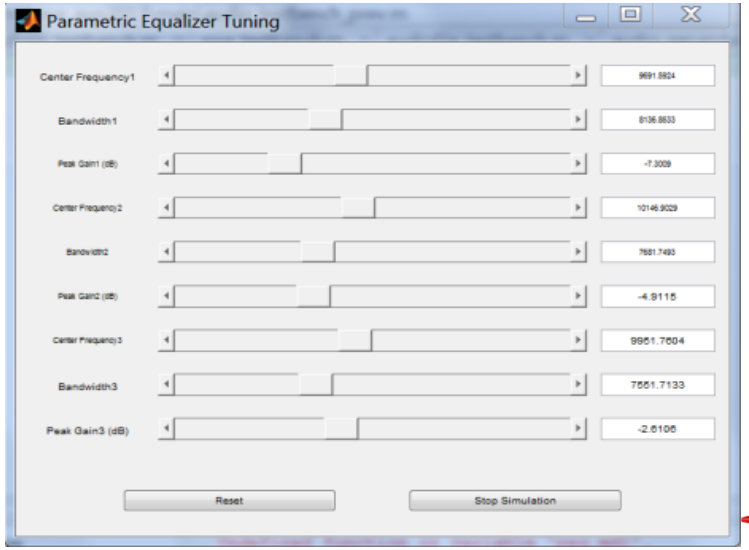
Stream processing in MATLAB

- *Streaming techniques* process continuous data from a captured signal or large file by dividing it into "frames"* and fully processes each frame before the next one arrives
  - ✓ Memory efficient
- Streaming algorithms in DSP System Toolbox provide
  - ✓ Implicit data buffering, state management and indexing
  - ✓ Simulation speed-up by reducing overhead



*http://www.mathworks.com/discovery/stream-processing.html

Tunable parameter equalizer example



Tune parameters in real-time

Tune it

Audio Input

Guitar10min.ogg a 44.1Khz stereo audio

Play it

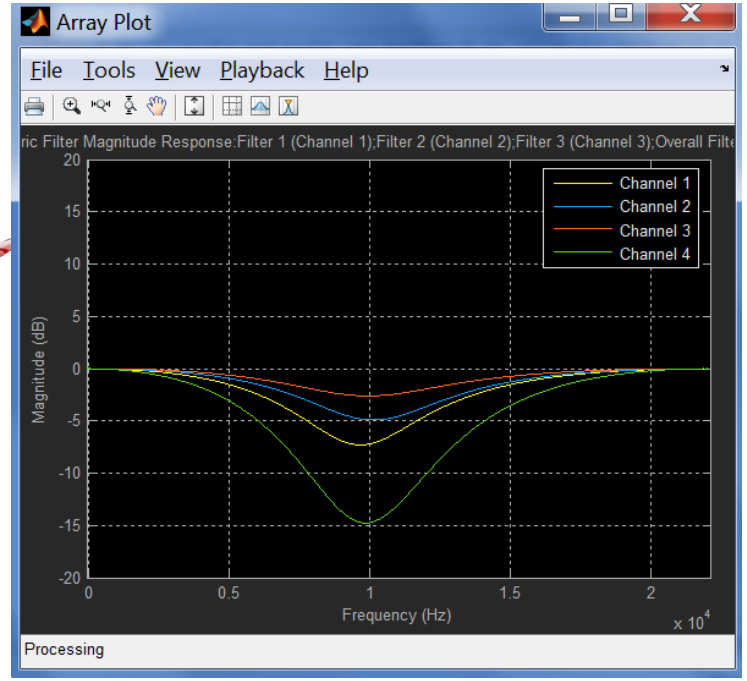
Parameter Equalizer Filters

Custom Audio Algorithm

Create it

See it

Array Plot



Visualize audio waveforms in real-time

Audio Output

Speaker

Hear it

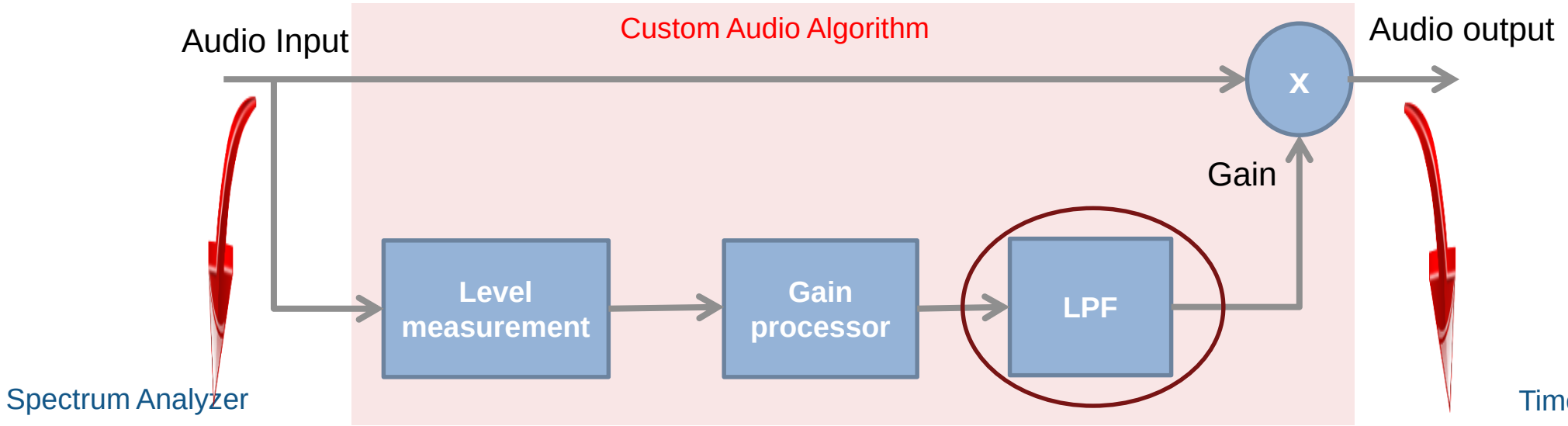
Part 1: Test bench and peripheral access

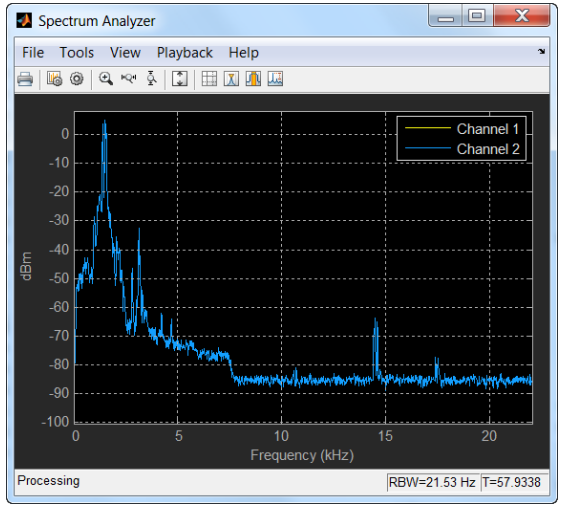
1. **How to create a streaming test bench for audio processing in MATLAB**
2. How to develop algorithms and incorporate them into the test bench
3. How to accelerate simulation for real-time performance

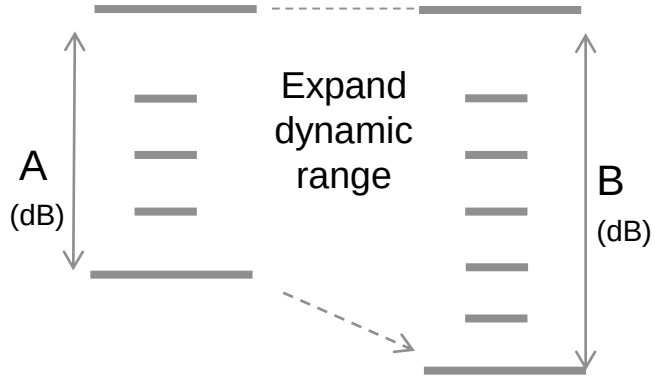
Part 2: Algorithms

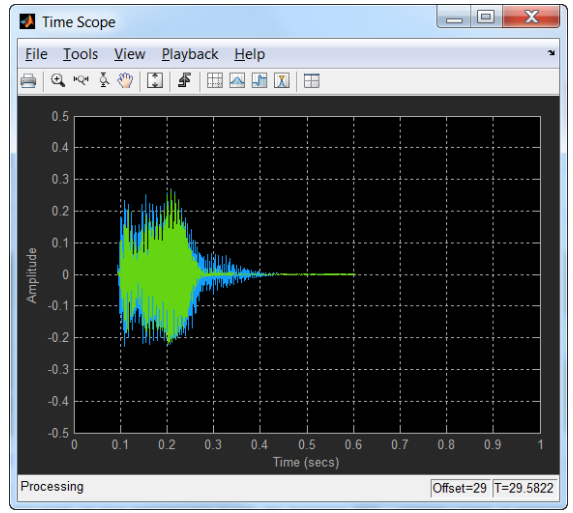
1. How to create a streaming test bench for audio processing in MATLAB
2. **How to develop algorithms and incorporate them into the test bench**
3. How to accelerate simulation for real-time performance

Example 1: Dynamic audio range expander









Webinar: Digital Filter Design Made Easy (https://www.mathworks.com/company/events/webinars/wbnr64770.html)

How to incorporate algorithm into test bench

```
%% Create & Initialize
SamplesPerFrame = 1024;
Fs = 44100;
Microphone = dsp.AudioRecorder('SamplesPerFrame');
MyTimeScope  = dsp.TimeScope('SampleRate',Fs);

h = fdesign.lowpass('fp,fst,ap,ast',4750,5250,0.5,80,Fs);
FIR =design(h,'equiripple','MinOrder','any,,'StopbandShape','flat','SystemObject', true);
z = zeros(1,Microphone.NumChannels);
```

initialize

```
%% Stream processing loop
tic;
while toc < 15
     % Read frame from microphone
       audioIn = step(Microphone);
   % Dynamic range expansion algorithm
     [audioOut,z]=expander_vec(audioIn,FIR,z);
     % View audio waveform
       step(MyTimeScope,[audioIn,audioOut]);
 end
```

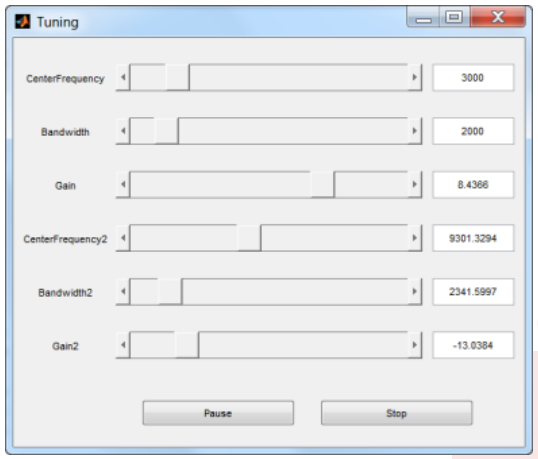
Algorithm

Process in-the-loop

```
%% Terminate
release(Microphone)
release(MyTimeScope)
```

Terminate

Example 2: Tunable audio parametric equalizer



Tune parameter equalizer in real-time

Tune it

See it

Play it

Guitar file @ 44.1Khz stereo audio

Audio Input

X

Parameter Equalizer Filters

Y

X

H=Y/X

Y

Transfer Function Estimator

H

Array Plot

Hear it

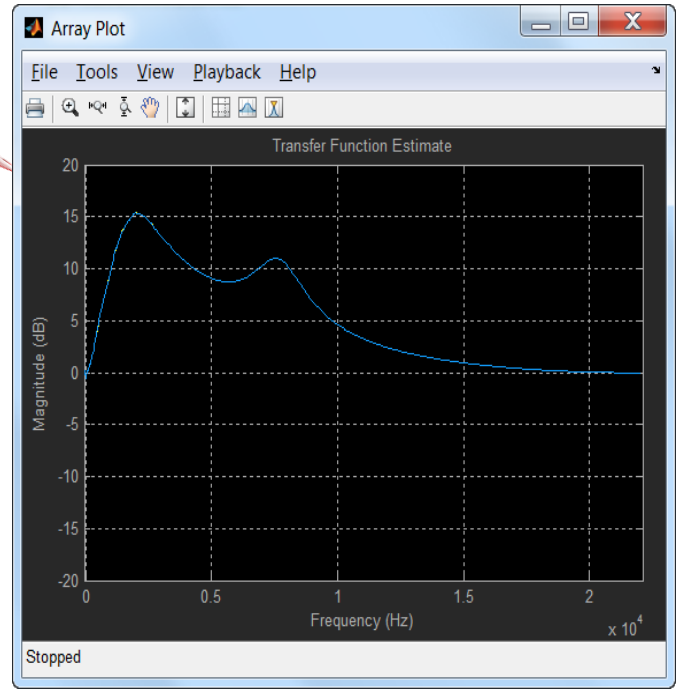
Y

Speaker

Audio Output

Custom Audio Algorithm

Create it



Visualize audio waveform in real-time

How to incorporate algorithms into test bench

```
%% Create & Initialize
SamplesPerFrame = 1024;
FReader=dsp.AudioFileReader('guitar2min.ogg','SamplesPerFrame',SamplesPerFrame)
Fs = FR.SampleRate;
TransferFuncEstimate= dsp.TransferFunctionEstimator('SampleRate',Fs,'FrequencyRange','onesided','SpectralAverages',1);
MyArrayPlot = dsp.ArrayPlot('PlotType','Line','Title','Transfer Function Estimate');

Speaker = dsp.AudioPlayer('SampleRate',Fs);
GUI = CreateParamTuningGUI(param, 'Tuning');
```

Initialize

```
%% Stream processing loop
tic;
while ~isDone(FR)
       audioIn = step(FReader); % Read frame from file
       [pauseSim,stopSim,tunedparams] = callbacks(param);

        % Audio processing algorithms - custom algorithm - PEQ
         audioOut = audio_algorithm_peqso(audioIn,tunedparams);
         H = step(TransferFuncEstimate,audioIn,audioOut);

   step(MyArrayPlot,20*log10(abs(H)));  %View estimated transfer function
   step(Speaker,audioOut);              %Play resulting audio
End
```

Algorithms

Process In-the-loop

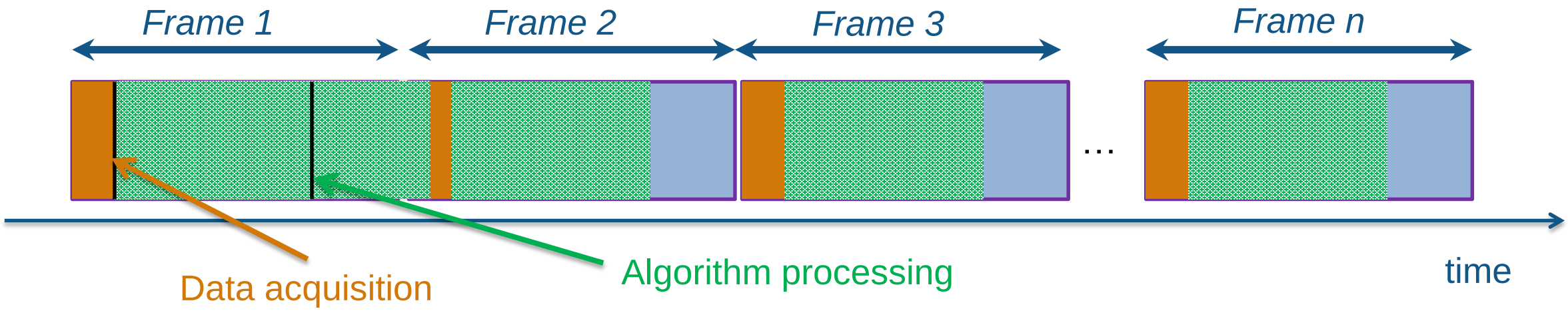
```
%% Terminate
release(FReader)
release(TransferFuncEstimate)
release(MyArrayPlot)
release(Speaker);
close(GUI);
```

Terminate

Part 3: Acceleration of simulation

1. How to create a streaming test bench for audio processing in MATLAB
2. How to develop algorithms and incorporate them into the test bench
3. **How to accelerate simulation for real-time performance**

Stream processing:
Data acquisition & algorithm times



As long as

Data acquisition **+** Algorithm processing **<=** Frame time

We have

Real-time signal processing

MATLAB to C code generation*

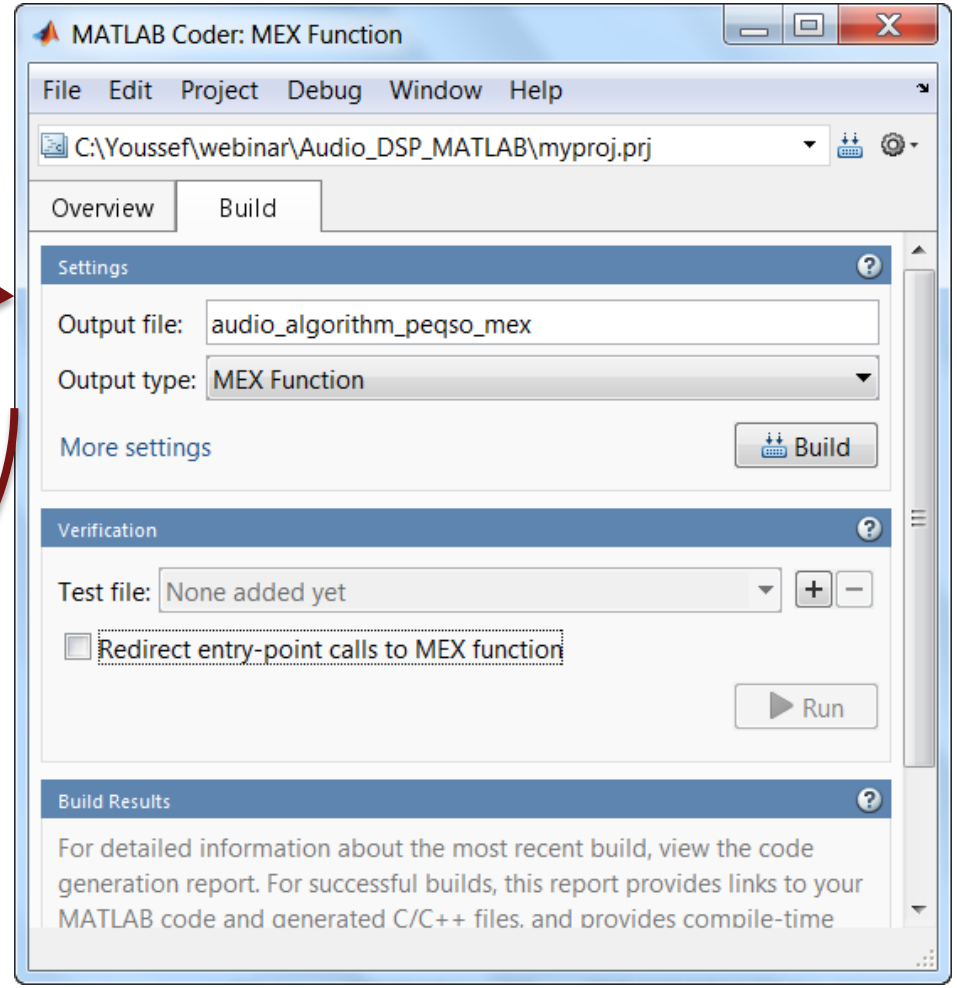
```
function y = audio_algorithm_peqso(u,tunedparams)
% Copyright 2013 The MathWorks, Inc.
persistent PE1 PE2
if isempty(PE1)
   PE1 = parametricEQFilter('Bandwidth',2000,…
                  'CenterFrequency',3000,'PeakGaindB',6.02);
   PE2 = ParametricEQFilter('Bandwidth',2000,…
                  'CenterFrequency',1000,'PeakGaindB',-6.02);
end
[PE1,PE2] = processtunedparams(tunedparams,PE1,PE2);
v = step(PE1,u);
y = step(PE2,v);
%-------------------------------------
function [PE1,PE2] = processtunedparams(tunedparams,PE1,PE2)

if ~isnan(tunedparams.CenterFrequency)
    PE1.CenterFrequency = tunedparams.CenterFrequency;
end
if ~isnan(tunedparams.Bandwidth)
    PE1.Bandwidth = tunedparams.Bandwidth;
end
if ~isnan(tunedparams.Gain)
    PE1.PeakGaindB = tunedparams.Gain;
end
if ~isnan(tunedparams.CenterFrequency2)
    PE2.CenterFrequency = tunedparams.CenterFrequency2;
end
if ~isnan(tunedparams.Bandwidth2)
    PE2.Bandwidth = tunedparams.Bandwidth2;
end
if ~isnan(tunedparams.Gain2)
    PE2.PeakGaindB = tunedparams.Gain2;
end
```

Algorithm.m

Algorithm.mex

MATLAB Coder



(*) Design and Prototype Real-Time DSP Systems with MATLAB (Conference Presentation):

http://www.mathworks.com/company/events/conferences/matlab-virtual-conference/2013/proceedings/design-and-prototype-real-time-dsp-systems-with-matlab.html

Simulation acceleration benchmarks

2-band parametric equalizer algorithm	Processing time
MATLAB code	**23.37** seconds
MEX code	**2.84** seconds

Speed up of ~ 8 x

Audio signal processing is everywhere!

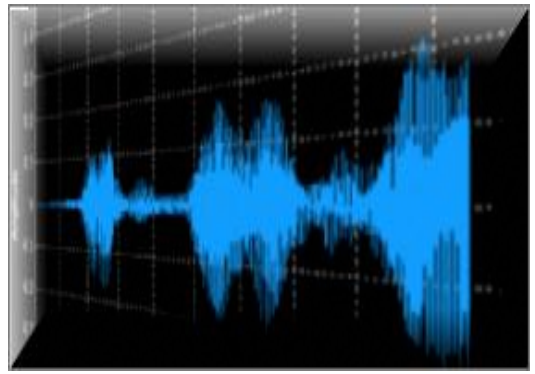
Tablet/ MP3 Player & Smart Phone

Professional Audio & Music

Automotive Audio & Navigation System

Gaming System

Medical Devices Hearing Aids



Create Your Own System Objects

```
classdef RemoveMean < matlab.System
% Remove estimated running mean.

properties
    % Memory weighting
    Weight = 0.8
end

properties (DiscreteState)
    Mean = 0 % Initial value
end

methods (Access=protected)
    function y = stepImpl(obj,u)
        y = u - obj.Mean;
        obj.Mean = u - y*obj.Weight;
    end
end

end
```

- `classdef` defines a System object using `matlab.System`
- `properties` defines parameters and states of your system
- `methods` implements the functions specific to your system
- `stepImpl` implements the kernel of the `step` function
- Other methods to consider: `setupImpl`, `resetImpl`, `releaseImpl`

http://www.mathworks.nl/help/dsp/basic-operations.html