Creating Audio Plugins with MATLAB

Gabriele Bunkheila, MathWorks
gbunkhei@mathworks.com

146th Convention of the AES, Dublin

2019 – 2nd MATLAB Plugin AES Student Competition!

http://www.aes.org/students/awards/mpsc/

First Stop for MATLAB Resources

MathWorks® Products Solutions Academia Support Community Events

Academia

Search MathWorks.com

Student Home | MATLAB Student | Examples | Student Competitions | Books | Hardware Support

AES Student Competition: MATLAB Plugin

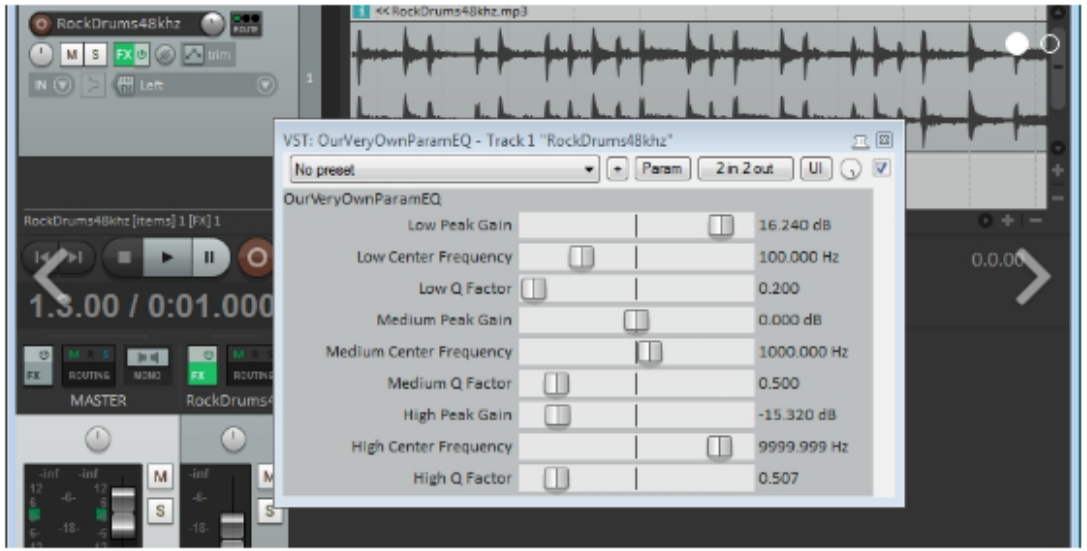
Design a new kind of audio production VST plugin using MATLAB and your wits!

MathWorks is actively supporting the second edition of AES MATLAB Plugin Competition, a student competition held by the Audio Engineering Society.

The competition provides students with the opportunity to challenge both their signal processing skills and creativity, and to share their results with the audio engineering community.

The shortlisted teams will be selected to present their work in person, for a chance to win cash and software prizes. The showcase and award ceremony will take place at the 147th Convention of the AES from October 16-19, 2019 in New York City.

For full details and rules, please visit the AES competition website.



https://www.mathworks.com/academia/student-competitions/aes-matlab.html

2018 Shortlist and Awards

- Plate reverb – https://www.youtube.com/watch?v=-lSVdCZhfrk
  [Aalborg University, Denmark]

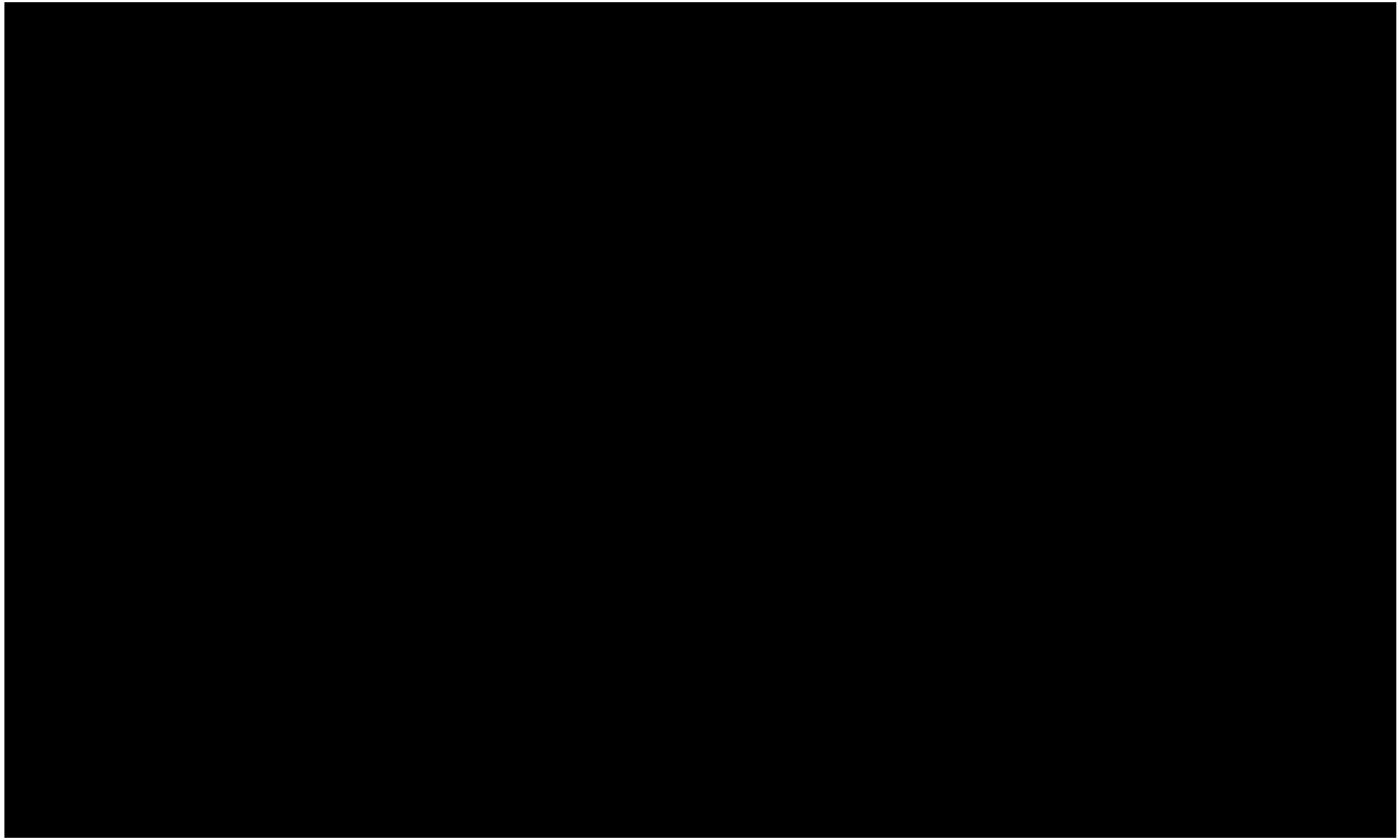
- Neural Reverberator – https://www.youtube.com/watch?v=_gJ-e1SsPkE
  [Clemson University, US]

- Binaural Piano – https://youtu.be/HUh6qq0oPyk
  [RWTH Aachen, Germany]

- Escalator – https://www.youtube.com/watch?v=G_vV2ysFgX4
  [York University, UK]

> Code all in MATLAB Central File Exchange – Search for *tag:"aescomp"*

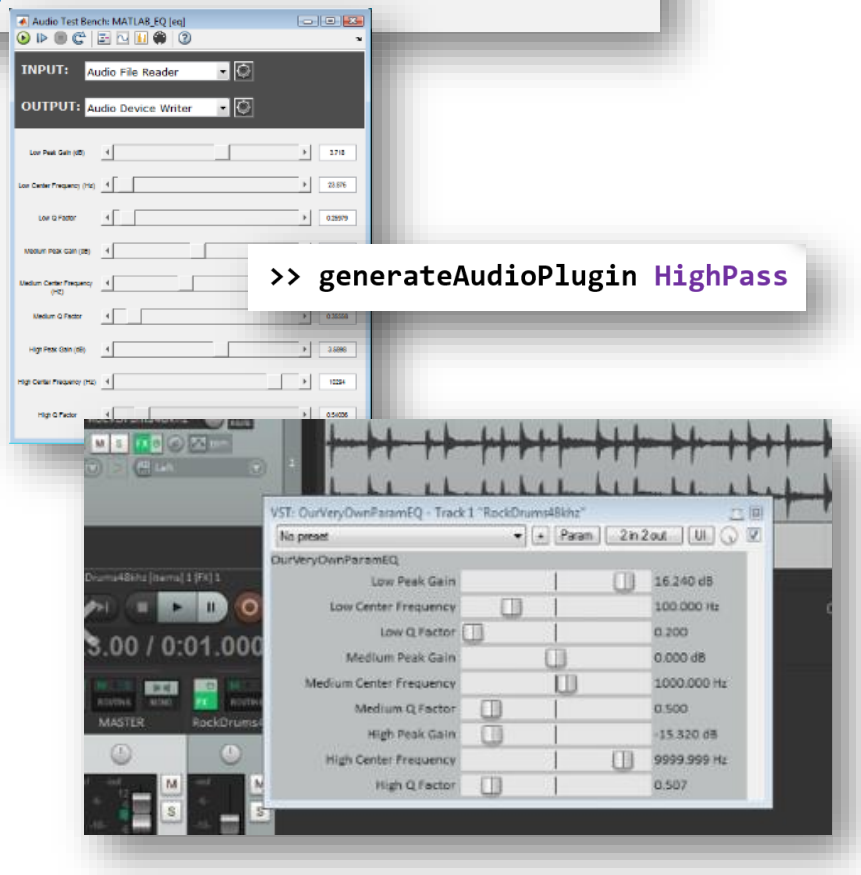
This Session – 2-Minute Deep Dive



Agenda – Key Ideas

- Structure MATLAB code for plugin generation:
  - Correct interface
  - Ready to generate C++
- Generate VST plugin from MATLAB
- Use generated plugin in DAW

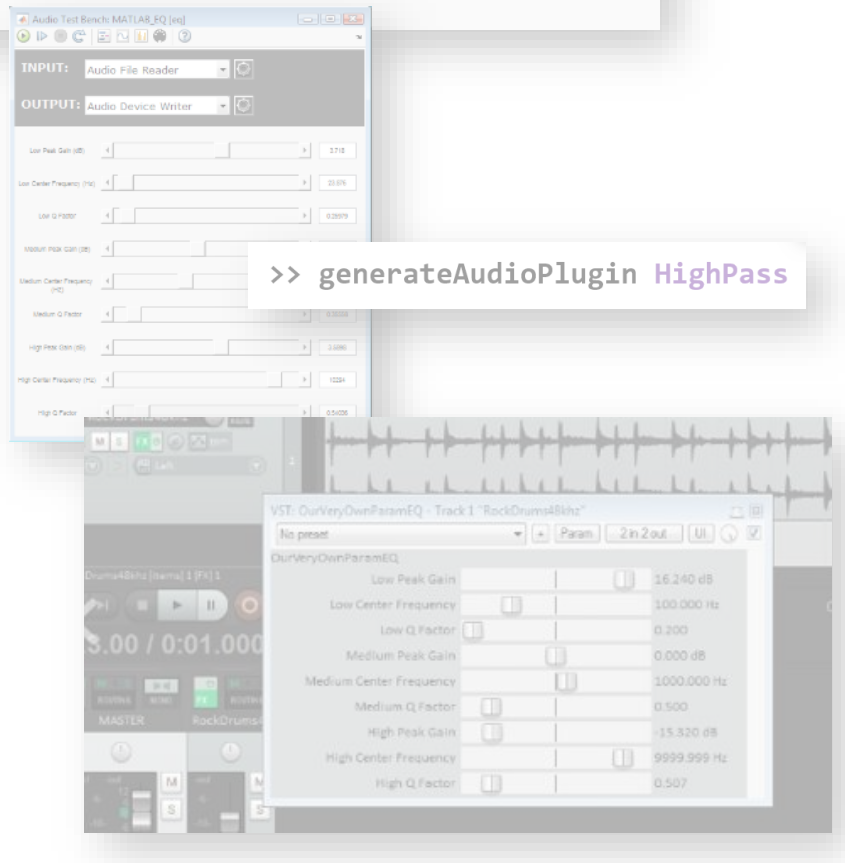
```
classdef stereoWidthExpander < audioPlugin
% Stereo width expander example

  properties
    Width = 1
  end

  properties (Constant)
    PluginInterface = audioPluginInterface( ..
      audioPluginParameter('Width', ...
      'Mapping', {'pow', 2, 0, 4}))
  end
```



Key Workflow Ideas

- Structure MATLAB code for plugin generation:
- Generate VST plugin from MATLAB
- Use generated plugin in DAW
- Not in scope for this tutorial
  - Building plugin UIs
  - Developing production plugins
  - Maximizing creative value

```
classdef stereoWidthExpander < audioPlugin
% Stereo width expander example

  properties
    Width = 1
  end

  properties (Constant)
    PluginInterface = audioPluginInterface( ..
      audioPluginParameter('Width', ...
      'Mapping', {'pow', 2, 0, 4}))
  end
```



Anatomy of Basic Real-Time streaming

- Simple `while` or `for` loop
  (N samples per frame)
- Audio I/O designed for streaming
  (decoupled setup vs. runtime)

```
% Initialize a file reader to produce an input signal
inReader = dsp.AudioFileReader('FunkyDrums-44p1-stereo-25secs.mp3');
Fs = inReader.SampleRate;
% Initialize a device writer to play back the output
outWriter = audioDeviceWriter('SampleRate', Fs);
% Initialize a spectrum analyzer to visualize the results over frequ
spectViewer = dsp.SpectrumAnalyzer('SampleRate',Fs,...
    'PlotAsTwoSidedSpectrum',false,'FrequencyScale','Log');

% Filter parameters
Fc = 1000;
z = zeros(2);

tic
while(toc < 11)
    % Read from file
    x = inReader();

    % Filter input block
    [y, z] = highPassFilter(Fc, Fs, x, z);

    % Write to audio device
    outWriter(y);

    % Visualize spectrum
    spectViewer([x(:,1),y(:,1)])

end
```

Packaging algorithms for real-time processing (1/4)

Plain functions can create issues, e.g.

- Single entry point
  (Coefficients computed even if not needed)
- Time-limited scope
  (States need maintaining outside)
- Unclear who are in, out, and parameters

```
function [out,State] = highPassFilter(Cutoff, Fs, in, varargin)
% Note: this function is inefficient in that it calculates new coef
% every time it is called regardless of whether Cutoff/Q have chang
[Num,Den] = computeCoefficients(Cutoff, Fs);
[out, State] = filter(Num,Den, in, varargin{:});


function [Num,Den] = computeCoefficients(Cutoff, Fs)
% Function to compute filter coefficients
w0 = 2*pi*Cutoff/Fs;
alpha = sin(w0)/sqrt(2);
cosw0 = cos(w0);
norm = 1/(1+alpha);
Num = (1 + cosw0)*norm * [.5 -1 .5];
Den = [1 -2*cosw0*norm (1 - alpha)*norm];
```

Packaging algorithms for real-time processing (2/4)

Objects provide solutions

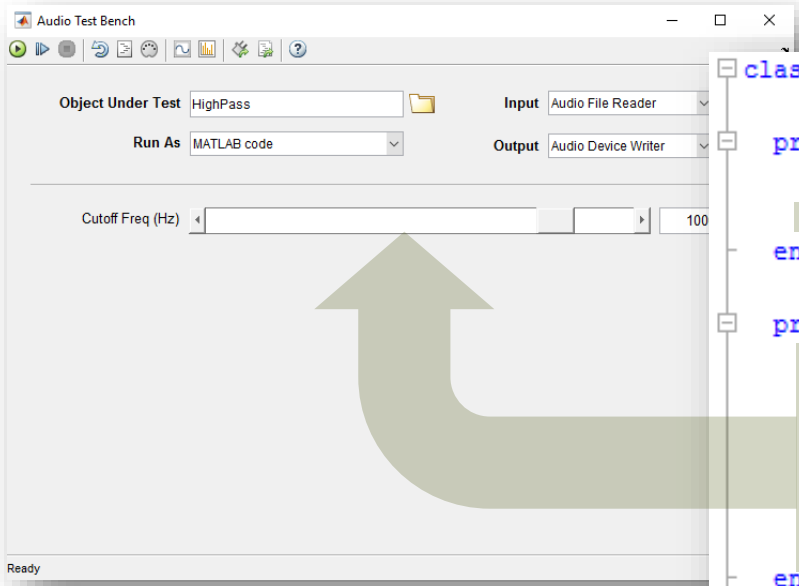
- Accessible parameters exposed as properties
- Parameter-led computations only triggered when parameters changed or initialized
- Pre-computed internal values and states are remembered
- Extra-lean runtime code
- Consistent signature for **in** and **out**

```
classdef HighPass < handle

  properties
    % public interface
    Fc = 1000
  end

  properties (Access = private)
    % internal state
    z = zeros(2)
    b = [1, zeros(1,2)]
    a = [1, zeros(1,2)]
  end

  methods

    function out = process(p, in)
      [out,p.z] = filter(p.b, p.a, in, p.z);
    end

    function reset(p)
      % initialize internal state
      p.z = zeros(2);
      Fs = getSampleRate(p);
      [p.b, p.a] = highPassCoeffs(p.Fc, Fs);
    end

    function set.Fc(p, Fc)
      p.Fc = Fc;
      Fs = getSampleRate(p);
      [p.b, p.a] = highPassCoeffs(Fc, Fs); %#
    end

  end

end
```

Packaging algorithms for real-time processing (3/4)

Objects can also store parameter metadata

- Define how parameters should be exposed to users
- Enable automatic generation of prototyping UI



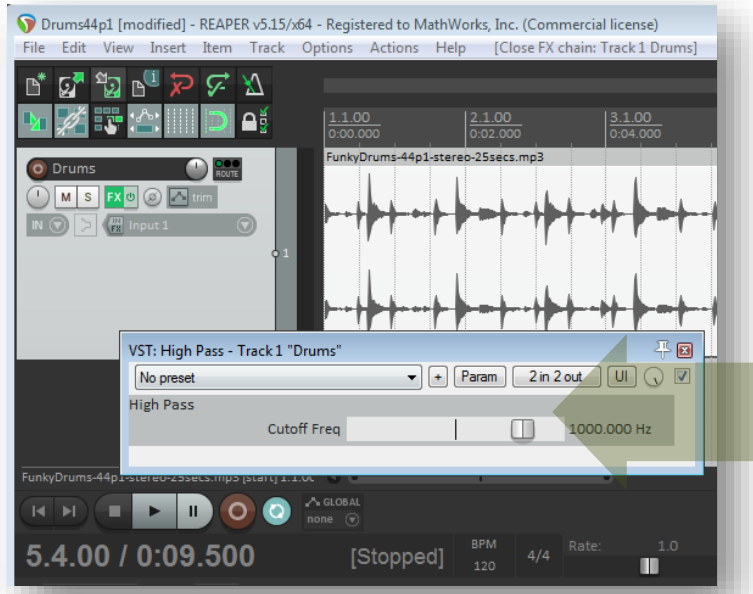

```
classdef HighPass < audioPlugin

  properties
    % public interface
    Fc = 1000
  end

  properties (Constant, Hidden)
    PluginInterface = audioPluginInterface( ...
      'PluginName','High Pass',...
      audioPluginParameter('Fc', ...
        'DisplayName',  'Cutoff Freq', ...
        'Label',  'Hz', ...
        'Mapping', { 'log', 20, 2000}))
  end

  properties (Access = private)
    % internal state
    z = zeros(2)
    b = [1, zeros(1,2)]
    a = [1, zeros(1,2)]
  end

  methods
```

```
>> audioTestBench HighPass
```

Packaging algorithms for real-time processing (4/4)

Objects can also store parameter metadata

- Define how parameters should be exposed to users
- Enable automatic generation of prototyping UI
- *…or **VST plugins***




```
classdef HighPass < audioPlugin

  properties
    % public interface
    Fc = 1000
  end

  properties (Constant, Hidden)
    PluginInterface = audioPluginInterface(
      'PluginName','High Pass',...
      audioPluginParameter('Fc', ...
        'DisplayName',  'Cutoff Freq', ...
        'Label',  'Hz', ...
        'Mapping', { 'log', 20, 2000}))
  end

  properties (Access = private)
    % internal state
    z = zeros(2)
    b = [1, zeros(1,2)]
    a = [1, zeros(1,2)]
  end

  methods
```

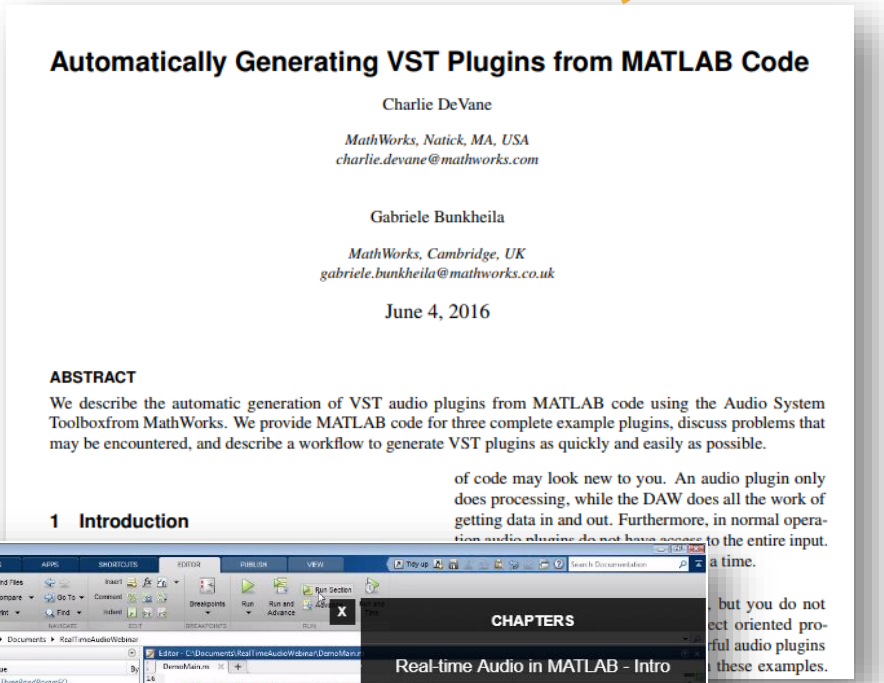
```
>> generateAudioPlugin HighPass
```

Plugin Generation – Additional Resources

- Automatically Generating VST Plugins from MATLAB Code (140 AES Convention Eng. Brief)
- Real-time Audio Processing for Algorithm Prototyping and Custom Measurements (Recorded Webinar – 45min)

- Design an Audio Plugin in MATLAB (Doc)
- Convert MATLAB Code to an Audio Plugin (Doc)
- Export a MATLAB Plugin to a DAW (Doc)
- Tips and Tricks for Plugin Authoring (Doc)
- Object-Oriented Design with MATLAB (Doc)

- Audio Plugin Example Gallery (Doc – Long collection of **`audioPlugin`** examples)


Automatically Generating VST Plugins from MATLAB Code

Charlie DeVane

MathWorks, Natick, MA, USA
charlie.devane@mathworks.com

Gabriele Bunkheila

MathWorks, Cambridge, UK
gabriele.bunkheila@mathworks.co.uk

June 4, 2016

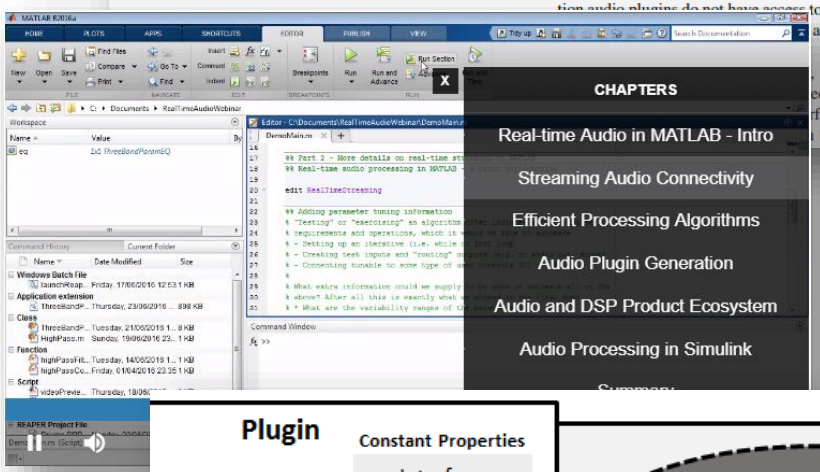
ABSTRACT

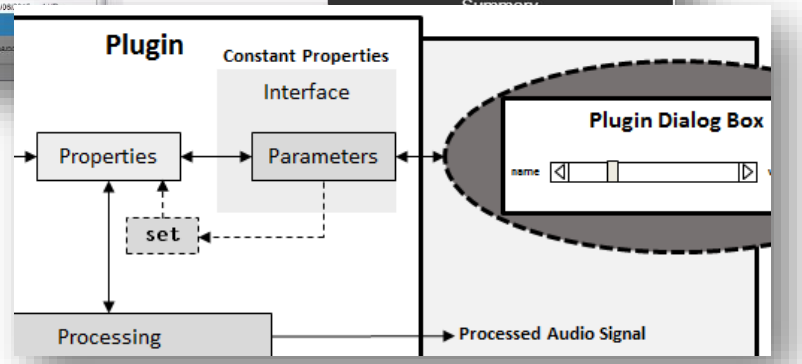
We describe the automatic generation of VST audio plugins from MATLAB code using the Audio System Toolboxfrom MathWorks. We provide MATLAB code for three complete example plugins, discuss problems that may be encountered, and describe a workflow to generate VST plugins as quickly and easily as possible.

1 Introduction

of code may look new to you. An audio plugin only does processing, while the DAW does all the work of getting data in and out. Furthermore, in normal operation audio plugins do not have access to the entire input. a time.

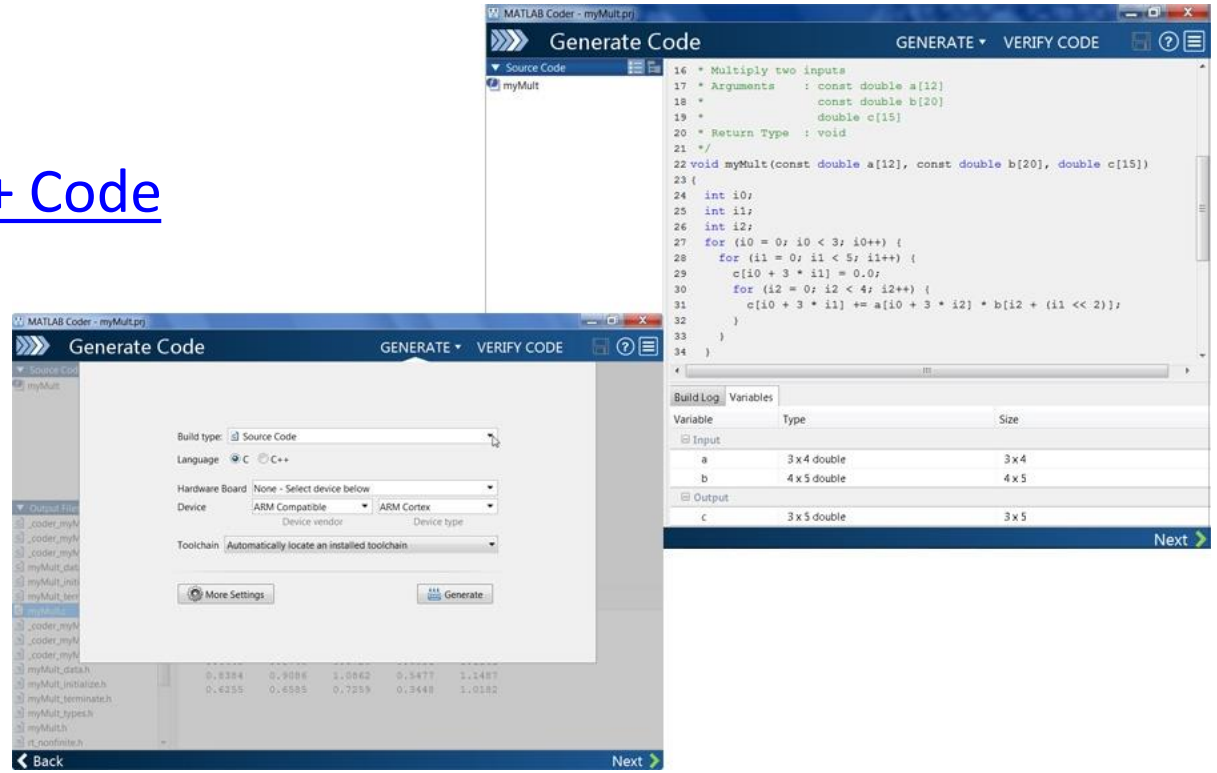
but you do not ect oriented pro- ful audio plugins these examples.



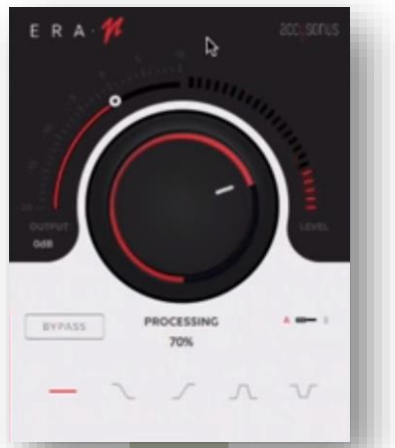


Generating C/C++ Code from MATLAB – Additional Resources

- MATLAB Code Design Considerations for Code Generation (doc)
- MATLAB Language Features Supported for C/C++ Code Generation (doc)
- Functions and Objects Supported for C/C++ Code Generation — Alphabetical List (doc)



Testing or Using External Plugins Programmatically



Existing VST or AU plugins can be driven via the same MATLAB interface

- Same requirements: designed for real-time audio processing

```
>> eq = loadAudioPlugin('ERA-N.dll')
>> y = process(eq, x);
```

```
>> noiseSuppressor = loadAudioPlugin('ERA-N.dll')

noiseSuppressor =

  VST plugin 'ERA-N'  2 in, 2 out

    Processing: 40 %
          Gain: 0 dB
          Tilt: 'NoTilt'
        Bypass: 0
```

Investigate Efficiency of MATLAB Code – Two Approaches

- tic/toc

```
% start timer
tic

% execute code
out = myFunction(in);

% stop timer (and store
% elapsed time)
et = toc;
```

- How long did it take?

- profile

```
% turn on profiler
profile on

% execute code
out = myFunction(in);

% turn off profiler
profile off
% open html report
profile report
```

- Where are the bottlenecks?

Two popular good practices for MATLAB programmers – any guess?

- Vectorisation
- Pre-allocation
- Correct indexing
- …

Optimizing MATLAB Performance – Additional Resources

- Speeding Up MATLAB Applications (Lauren's blog)
- Techniques to Improve Performance (doc)
- Vectorization (doc)
- Preallocation (doc)
- Understanding Array Preallocation (Lauren's blog)
- Indexing: Programming Patterns: Maximizing Code Performance by Optimizing Memory Access (Technical article)
- Performance and memory (doc)

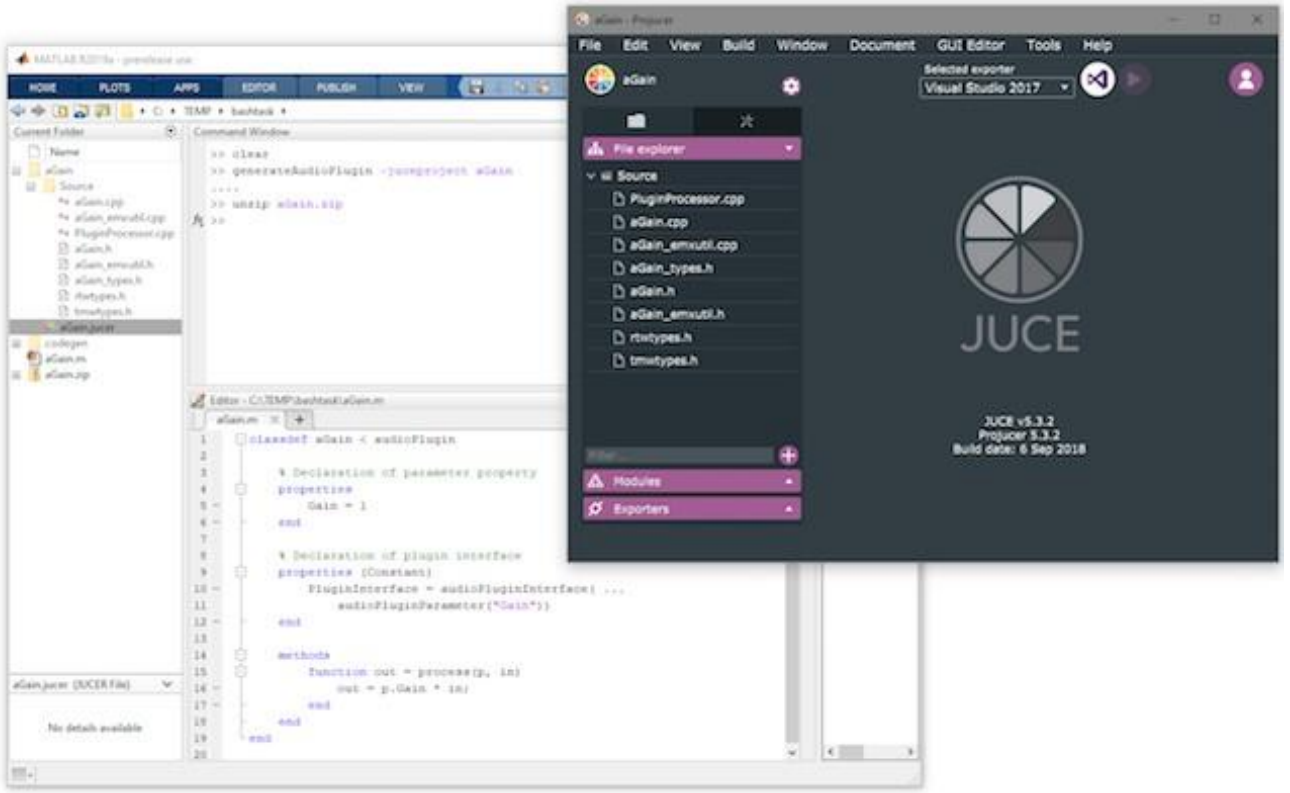
MATLAB vs. C++ – Which is Faster?

- It depends!
- MATLAB can be very fast (multi-threaded under the hood):
  - Vectorized Linear Algebra (BLAS, LAPACK)
  - FFT/IFFT (FFTW)
  - FIR filters (IPP, TBB)
  - ... (and more)
- C++ in generated plugin will generally use (by default):
  - Efficient algorithm implementations
  - Single-threaded code

New with MATLAB R2019a – Generate JUCE Projects

```
generateAudioPlugin -juceproject yourPlugin
```

- Creates
  - .zip archive with
  - C++ source files
  - JUCER
- Requires MATLAB Coder
- Not required nor evaluated for Competition!



Agenda – Key Ideas

- Structure MATLAB code for plugin generation:
  - Correct interface
  - Ready to generate C++
- Generate VST plugin from MATLAB
- Use generated plugin in DAW

```
classdef stereoWidthExpander < audioPlugin
% Stereo width expander example

  properties
    Width = 1
  end

  properties (Constant)
    PluginInterface = audioPluginInterface( ..
      audioPluginParameter('Width', ...
      'Mapping', {'pow', 2, 0, 4}))
  end
```