

MATLAB で Audio Plugin 開発

#01 ゲイン (+プラグインの作りの基礎)

松本 和樹 (早稲田大学, MATLAB Student Ambassador)

はじめに

自己紹介



氏名 : 松本 和樹
所属 : 早稲田大学
研究 : 音響信号処理
趣味 : 作曲
仕事 : MATLAB Student Ambassador
@km_MATLAB_Amb

MATLAB とは



MATLAB® は思考や作業プロセスに合うように設計されています。

MATLAB® は、反復的に実施する分析や設計プロセスに適したデスクトップ環境、そして行列と配列の数学を直接表現するプログラミング言語が 1 つに組み合わさっています。

各分野の専門家による製品開発

MATLAB ツールボックスは各分野の専門家により開発され、厳密なテストを経ており、仕様はドキュメンテーションに詳細にまとめられています。

対話型アプリ

MATLAB アプリを使用することで、異なるアルゴリズムが特定のデータを使用してどのように動作するかを確認できます。望ましい結果が得られるまで作業を繰り返し、その後 MATLAB プログラムを自動的に生成して作業の再現や自動化が可能です。

スケーラビリティ

わずかなコード変更で、クラスター、GPU、またはクラウドでの実行まで分析を拡張できます。コードを書き直したり、ビッグデータ プログラミングや out-of-memory 処理の技術を学習したりする必要はありません。

<https://jp.mathworks.com/products/matlab.html>

MathWorks 社の数値解析ソフトウェア

- プログラミング言語
- データの処理や可視化が簡単

様々な Toolbox[®] が研究開発を支援

- 信号処理
- 機械学習

Campus-Wide License と Student Ambassador



**MATLABによる
データの可視化**

Student Ambassador 主催 / 理工メディアセンター共催

講座概要
可視化はデータの特徴をわかりやすく伝えるための重要な手段です。本講習会では、数値計算ソフトウェア MATLAB の基礎から、レポートで映えるグラフを作成するための基本的な技術までを紹介します

対象者
● 早稲田大学に所属する学生・教職員
● プログラミングに触れたことがある方

参加方法
QRコードから理工コンピュータセミナーのページにアクセスし「MATLABによるデータの可視化」の申し込みを申請

実施場所
対面実施：西早稲田キャンパス
63号館3階PCルーム<Aルーム>

12/14(木) 18:55 開始
20:35 終了

参加登録
[登録する](#)

Student Ambassadorとは
MATLAB Student Ambassadorとは、Campus-Wide Licenseを導入している大学等の教育機関において、MathWorks社と協力し、MATLABの普及を促進するための活動を担う学生のことです。MATLAB講習会の企画・主催、TwitterやQiitaでの情報発信、MATLABに関する質問の要付など、幅広い活動を実施しています。

お問い合わせは下記まで
ご連絡ください！

Twitter
@km_MATLAB_Amb

E-mail
km.matlabamb@gmail.com

MATLABを無料で使えるライセンス

- 大学などの教育機関が MathWorks と提携
- MATLAB 本体だけでなく、Toolbox も無料
- 使わないともったいない！

MATLAB Student Ambassador

Campus-Wide License を導入する大学で
MATLABの利用を促進するために活動する学生

- MATLAB 講習会の開催
- 質問への対応
- X : @km_MATLAB_Amb

MATLAB で Audio Plugin 開発

Audio Plugin とは

- DAW 等の音楽制作ソフトウェア上で動作する拡張機能
- MATLAB の **Audio Toolbox**  で手軽な開発が可能

シリーズの内容

- 第1回 : ゲイン (+プラグインの作りの基礎) 
- 第2回 : イコライザ 
- 第3回 : ディストーション 

プログラムは : <https://github.com/kzkmtmt/audioPlugins>  からダウンロード

MATLAB で Audio Plugin 開発

こんな人におすすめ

- 自作の Audio Plugin を作ってみたい / MATLAB で信号処理をしてみたい

必要なもの

- MATLAB 2024b
- Toolbox (Signal Processing Toolbox, DSP System Toolbox, Audio Toolbox)
MATLAB をインストール済みの場合 : ホーム > アドオン  から追加できる
- C/C++ コンパイラ  (Audio Toolbox をカバーするもの)
 - Windowsの場合 : Microsoft Visual C++ 2022  など
※ インストール時に Desktop development with C++  を選択してください
- VSTプラグインを読み込む DAW (**REAPER**  など)

Audio Plugin 開発の流れ

Audio Plugin 開発の流れ

1. アルゴリズムの開発
2. プラグインの実装
 - `audioPlugin` クラスを継承
 - `process` メソッド内にアルゴリズムを実装
 - `pluginInterface` で操作画面を実装
3. `audioTestBench` を用いたテスト
4. `validateAudioPlugin` を用いた検証
5. `generateAudioPlugin` を用いた VST プラグイン生成
6. DAW 上で動作確認

ゲインの実装

Audio Plugin 開発の流れ

1. アルゴリズムの開発 
2. プラグインの実装
 - `audioPlugin` クラスを継承
 - `process` メソッド内にアルゴリズムを実装
 - `pluginInterface` で操作画面を実装
3. `audioTestBench` を用いたテスト
4. `validateAudioPlugin` を用いた検証
5. `generateAudioPlugin` を用いた VST プラグイン生成
6. DAW 上で動作確認

ゲインのアルゴリズム



- ゲイン：音量を変化させるプラグイン
- 入力 $x(t)$ に係数 a を掛けて出力 (t は時刻)

$$y(t) = ax(t)$$

- a が大きければ音は大きく、小さければ小さくなる
- a の値はデシベル単位のゲイン g から計算

$$g = 20 \log_{10} a \quad \Rightarrow \quad a = 10^{g/20}$$

- プログラムは以下の通り (6 dB で $a = 1.99 \approx 2$)

```

x = [1;2;3];    % 入力信号の例
g = 6;          % ゲイン [dB]
a = 10^(g/20); % デシベル値から係数を計算。a = db2mag(g) でも良い
y = a * x;      % 入力 x に定数 a を掛けて出力

```

Audio Plugin 開発の流れ

1. アルゴリズムの開発 
2. プラグインの実装
 - `audioPlugin` クラスを継承
 - `process` メソッド内にアルゴリズムを実装
 - `pluginInterface` で操作画面を実装
3. `audioTestBench` を用いたテスト
4. `validateAudioPlugin` を用いた検証
5. `generateAudioPlugin` を用いた VST プラグイン生成
6. DAW 上で動作確認

Audio Plugin 開発の流れ

1. アルゴリズムの開発 
2. プラグインの実装
 - `audioPlugin` クラスを継承 
 - `process` メソッド内にアルゴリズムを実装
 - `pluginInterface` で操作画面を実装
3. `audioTestBench` を用いたテスト
4. `validateAudioPlugin` を用いた検証
5. `generateAudioPlugin` を用いた VST プラグイン生成
6. DAW 上で動作確認

プラグインのクラスを実装

MATLABGain.m を作成

 クラスや考え方・使い方は [オブジェクト指向プログラミング入門](#) などをチェック

audioPlugin クラスの継承

```
classdef クラス名 < audioPlugin  
  
properties  
    % プラグインが持つ変数を定義  
end  
  
methods  
    % 信号を処理する関数  
    function y = process(p,x)  
        y = x; % 入力をそのまま返す  
    end  
end  
end
```

クラス名

- プラグインの名前
- ファイル名と同じ名前に設定

properties

- プラグイン `p` が持つ変数を定義する

process メソッド

- 入力信号 `x` から出力信号 `y` を計算する関数
- `p` はプラグイン自身を参照するためのもの

Audio Plugin 開発の流れ

1. アルゴリズムの開発 
2. プラグインの実装
 - `audioPlugin` クラスを継承 
 - `process` メソッド内にアルゴリズムを実装
 - `pluginInterface` で操作画面を実装
3. `audioTestBench` を用いたテスト
4. `validateAudioPlugin` を用いた検証
5. `generateAudioPlugin` を用いた VST プラグイン生成
6. DAW 上で動作確認

Audio Plugin 開発の流れ

1. アルゴリズムの開発 
2. プラグインの実装
 - `audioPlugin` クラスを継承 
 - `process` メソッド内にアルゴリズムを実装 
 - `pluginInterface` で操作画面を実装
3. `audioTestBench` を用いたテスト
4. `validateAudioPlugin` を用いた検証
5. `generateAudioPlugin` を用いた VST プラグイン生成
6. DAW 上で動作確認

process メソッド内にアルゴリズムを実装

```
classdef MATLAB_Gain < audioPlugin

properties
    g = 0 % ゲイン [dB]
end

methods
    function y = process(p,x)
        a = 10^(p.g/20); % g から a を計算
        y = a * x;         % x に a をかけて出力
    end
end
```

クラス名

- ファイル名に併せて `MATLAB_Gain` に変更
- 自分で名前を決めたい場合はファイル名も変更

properties

- 音量のパラメータ `g` を用意し 0 で初期化

process メソッド

- `p.g` の値を参照して係数 `a` を計算
- 入力 `x` に `a` をかけて出力

Audio Plugin 開発の流れ

1. アルゴリズムの開発 
2. プラグインの実装
 - `audioPlugin` クラスを継承 
 - `process` メソッド内にアルゴリズムを実装 
 - `pluginInterface` で操作画面を実装
3. `audioTestBench` を用いたテスト
4. `validateAudioPlugin` を用いた検証
5. `generateAudioPlugin` を用いた VST プラグイン生成
6. DAW 上で動作確認

Audio Plugin 開発の流れ

1. アルゴリズムの開発 ✓
2. プラグインの実装
 - `audioPlugin` クラスを継承 ✓
 - `process` メソッド内にアルゴリズムを実装 ✓
 - `pluginInterface` で操作画面を実装 📝
3. `audioTestBench` を用いたテスト
4. `validateAudioPlugin` を用いた検証
5. `generateAudioPlugin` を用いた VST プラグイン生成
6. DAW 上で動作確認

audioPluginInterface で操作画面を定義

```
properties(Constant)
    PluginInterface = ...
        audioPluginInterface( ...
            audioPluginParameter(...
                'g',...
                'DisplayName','Gain',...
                'Mapping',{'lin',-20,20},...
                'Label','dB'...
            )...
        );
end
```

- プロパティに `PluginInterface` を追加
- 画面上で操作したいパラメータの数だけ `audioPluginParameter` を作成して `audioPluginInterface` に渡す

設定項目	値
プロパティ	'g'
表示名	Gain
マッピング	線形軸 (lin) 範囲は -20 から 20
ラベル	'dB'

💡 Design User Interface for Audio Plugin 🔒

Audio Plugin 開発の流れ

1. アルゴリズムの開発 ✓
2. プラグインの実装
 - `audioPlugin` クラスを継承 ✓
 - `process` メソッド内にアルゴリズムを実装 ✓
 - `pluginInterface` で操作画面を実装 📝
3. `audioTestBench` を用いたテスト
4. `validateAudioPlugin` を用いた検証
5. `generateAudioPlugin` を用いた VST プラグイン生成
6. DAW 上で動作確認

Audio Plugin 開発の流れ

1. アルゴリズムの開発 ✓
2. プラグインの実装
 - `audioPlugin` クラスを継承 ✓
 - `process` メソッド内にアルゴリズムを実装 ✓
 - `pluginInterface` で操作画面を実装 ✓
3. `audioTestBench` を用いたテスト 📝
4. `validateAudioPlugin` を用いた検証
5. `generateAudioPlugin` を用いた VST プラグイン生成
6. DAW 上で動作確認

audioTestBench を用いたテスト

以下のコマンドで audioTestBench を起動 & パラメータを操作し、音量の変化を確認

```
>> audioTestBench MATLAB_Gain
```

Audio Plugin 開発の流れ

1. アルゴリズムの開発 ✓
2. プラグインの実装
 - `audioPlugin` クラスを継承 ✓
 - `process` メソッド内にアルゴリズムを実装 ✓
 - `pluginInterface` で操作画面を実装 ✓
3. `audioTestBench` を用いたテスト 📝
4. `validateAudioPlugin` を用いた検証
5. `generateAudioPlugin` を用いた VST プラグイン生成
6. DAW 上で動作確認

Audio Plugin 開発の流れ

1. アルゴリズムの開発 ✓
2. プラグインの実装
 - `audioPlugin` クラスを継承 ✓
 - `process` メソッド内にアルゴリズムを実装 ✓
 - `pluginInterface` で操作画面を実装 ✓
3. `audioTestBench` を用いたテスト ✓
4. `validateAudioPlugin` を用いた検証 📄
5. `generateAudioPlugin` を用いた VST プラグイン生成
6. DAW 上で動作確認

validateAudioPlugin を用いた検証

validateAudioPlugin を用いてプラグインの動作を検証する

```
>> validateAudioPlugin MATLAB_Gain
```

プラグインのクラスをチェック中 'MATLAB_Gain'... 合格.
テストベンチ ファイルの生成中 'testbench_MATLAB_Gain.m'... 完了.
テストベンチの実行中... 合格.
MEX ファイルの生成中 'testbench_MATLAB_Gain_mex.mexw64'... 完了.
MEX テストベンチの実行中... 合格.
テストベンチの削除中.
オーディオ プラグインを生成する準備ができました.

Audio Plugin 開発の流れ

1. アルゴリズムの開発 ✓
2. プラグインの実装
 - `audioPlugin` クラスを継承 ✓
 - `process` メソッド内にアルゴリズムを実装 ✓
 - `pluginInterface` で操作画面を実装 ✓
3. `audioTestBench` を用いたテスト ✓
4. `validateAudioPlugin` を用いた検証 📄
5. `generateAudioPlugin` を用いた VST プラグイン生成
6. DAW 上で動作確認

Audio Plugin 開発の流れ

1. アルゴリズムの開発 ✓
2. プラグインの実装
 - `audioPlugin` クラスを継承 ✓
 - `process` メソッド内にアルゴリズムを実装 ✓
 - `pluginInterface` で操作画面を実装 ✓
3. `audioTestBench` を用いたテスト ✓
4. `validateAudioPlugin` を用いた検証 ✓
5. `generateAudioPlugin` を用いた VST プラグイン生成 📝
6. DAW 上で動作確認

generateAudioPlugin を用いた VST プラグイン生成

generateAudioPlugin を用いてプラグインを生成

- コンパイラのインストールが必要になる
- デフォルトではVST規格のDLLファイルが生成される
- オプションでAU規格のプラグインやEXEファイルも生成できる

 詳しくは **generateAudioPlugin**  を参照

```
>> generateAudioPlugin MATLAB_Gain
```

Audio Plugin 開発の流れ

1. アルゴリズムの開発 ✓
2. プラグインの実装
 - `audioPlugin` クラスを継承 ✓
 - `process` メソッド内にアルゴリズムを実装 ✓
 - `pluginInterface` で操作画面を実装 ✓
3. `audioTestBench` を用いたテスト ✓
4. `validateAudioPlugin` を用いた検証 ✓
5. `generateAudioPlugin` を用いた VST プラグイン生成 📝
6. DAW 上で動作確認

Audio Plugin 開発の流れ

1. アルゴリズムの開発 ✓
2. プラグインの実装
 - `audioPlugin` クラスを継承 ✓
 - `process` メソッド内にアルゴリズムを実装 ✓
 - `pluginInterface` で操作画面を実装 ✓
3. `audioTestBench` を用いたテスト ✓
4. `validateAudioPlugin` を用いた検証 ✓
5. `generateAudioPlugin` を用いた VST プラグイン生成 ✓
6. DAW 上で動作確認 📎

6. DAW 上で動作確認

プラグインの生成が終わったら、DAW 上で動作を確認！

- 様々な DAW 上で動作確認するのが望ましい
- 本シリーズでは REAPER を用いる
- DAW ごとに VST Plugin の読み込み方法が異なる

【REAPER の場合】

- Option > Preferences > VST > Edit path list > add path で
- DLL ファイルの格納されたディレクトリを追加し Rescan ボタンを押す

<https://www.REAPER.fm/>

Audio Plugin 開発の流れ

1. アルゴリズムの開発 ✓
2. プラグインの実装
 - `audioPlugin` クラスを継承 ✓
 - `process` メソッド内にアルゴリズムを実装 ✓
 - `pluginInterface` で操作画面を実装 ✓
3. `audioTestBench` を用いたテスト ✓
4. `validateAudioPlugin` を用いた検証 ✓
5. `generateAudioPlugin` を用いた VST プラグイン生成 ✓
6. DAW 上で動作確認 ✓

VST プラグイン開発の一連の流れを確認 100

おわりに

まとめ

- ゲインの実装
- Audio Plugin 開発の流れ

役立つ資料

- **MATLAB 入門** : MATLAB の基本的な使い方
- **オブジェクト指向プログラミング入門** : オブジェクト指向プログラミング
- **Audio Plugin Example Gallery** : Audio Plugin の MATLAB 実装例
- **Audio Plugin Creation and Hosting** : Audio Plugin 関連のドキュメント