

MATLAB で Audio Plugin 開発

#02 イコライザ

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 で Audio Plugin 開発

Audio Plugin とは

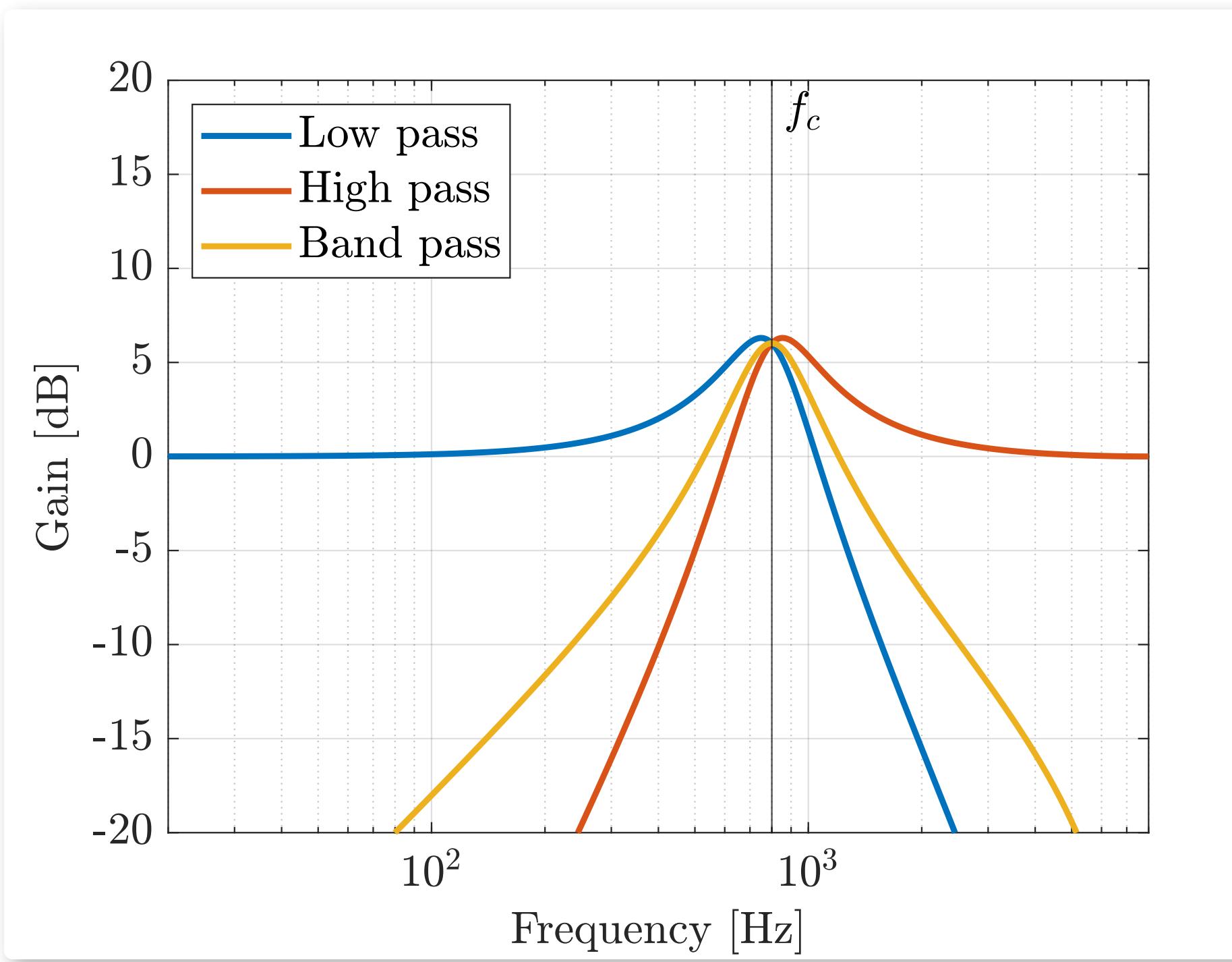
- DAW 等の音楽制作ソフトウェア上で動作する拡張機能
- MATLAB の **Audio Toolbox**  で手軽な開発が可能

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 

イコライザ

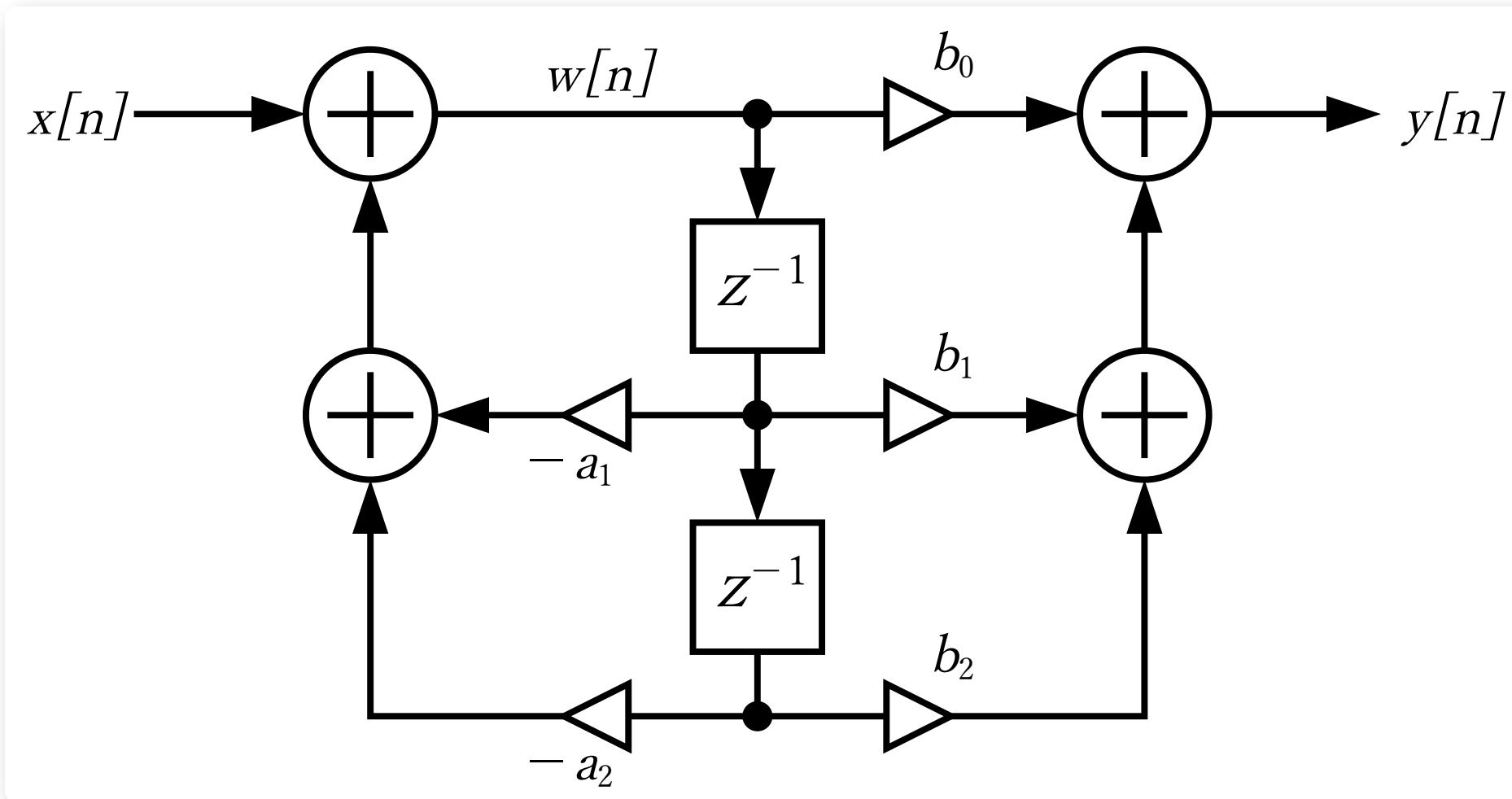
イコライザとは



- 周波数帯域ごとの音量を整えるツール
- 信号処理的には線形フィルタと呼ばれる
- 今回は三種類のフィルタを実装

種類	効果
Low Pass	低域を通過（ハイカット）
High Pass	高域を通過（ローカット）
Band Pass	カットオフ周波数を中心とした 帯域を通過

双二次フィルタ



$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}}$$

双二次フィルタ（上）とその伝達関数（下）

画像 : https://en.wikipedia.org/wiki/File:Biquad_filter_DF-IIRx.svg

- イコライザでは**双二次フィルタ (biquad filter)**を用いることが多い
- IIR (Infinite Impulse Response) フィルタの一種
- 分子係数 b_0, b_1, b_2 , 分母係数 a_1, a_2 の設定により様々なフィルタを実現できる

要素	記号	働き
加算器	⊕	信号を加算
乗算器	→ a	信号を a 倍
遅延器	→ z^{-1}	1 サンプル遅延

Low Pass Filter の係数の計算式

式 (双一次変換を用いて導出)

$$\left\{ \begin{array}{l} b_0 = \frac{1 - \cos \omega_c}{2} \\ b_1 = 1 - \cos \omega_c \\ b_2 = \frac{1 - \cos \omega_c}{2} \end{array} \right\} \quad \left\{ \begin{array}{l} a_0 = 1 + \alpha \\ a_1 = -2 \cos \omega_c \\ a_2 = 1 - \alpha \end{array} \right.$$

- f_c : カットオフ周波数, Q : レゾナンス
- f_s : サンプリング周波数
- $\omega_c = 2\pi f_c / f_s$, $\alpha = \sin \omega_c / 2Q$
- 導出や他のフィルタについては
Audio EQ Cookbook  で確認

実装

```
fc      = 400;
Q       = 2;
fs      = 16000;
wc      = 2*pi*fc/fs;
alpha   = sin(wc)/(2*Q);
coswc  = cos(wc);
```

```
% b = [b0 b1 b2], a = [a0 a1 a2]
b = [ (1-coswc)/2, 1-coswc, (1-coswc)/2];
a = [1+alpha, -2*coswc, 1-alpha];
```

フィルタ特性の確認 / フィルタの適用

- `freqz` : 得られたフィルタ (分子係数 `b` と分母係数 `a`) の特性を確認

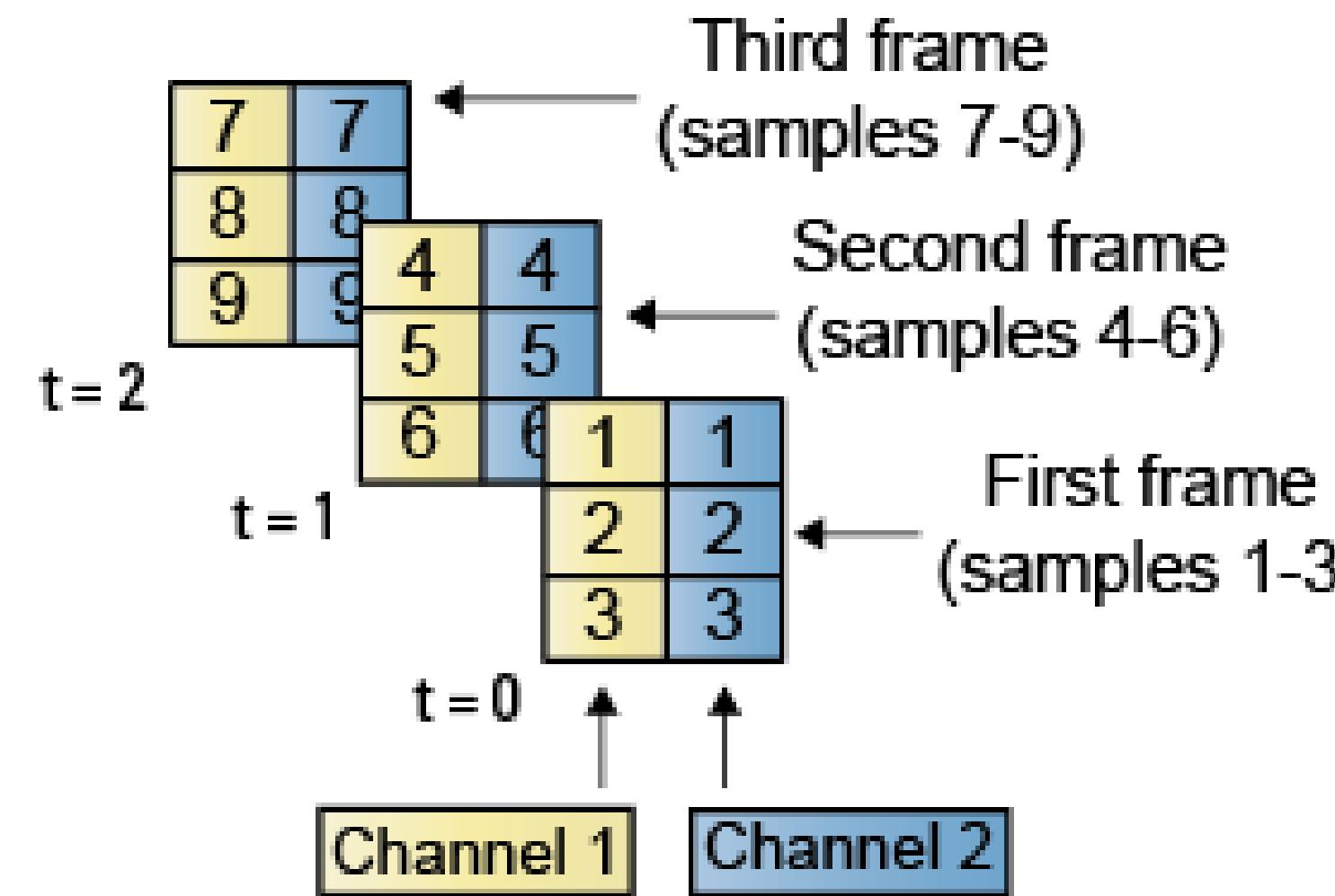
```
[h,f]=freqz(b,a,[],fs); % 周波数特性を計算.  
plot(f,mag2db(abs(h)),LineWidth=2); % 振幅特性 (= h の絶対値を dB に変換したもの) をプロット
```

- `filter` : 信号にフィルタ (分子係数 `b` と分母係数 `a`) を適用する

```
x = randn(fs*0.5,1); % 0.5 秒の雑音を生成  
y = filter(b,a,x); % フィルタを適用  
soundsc(y,fs); % 再生
```

フレームベースの処理

フレームベースの処理

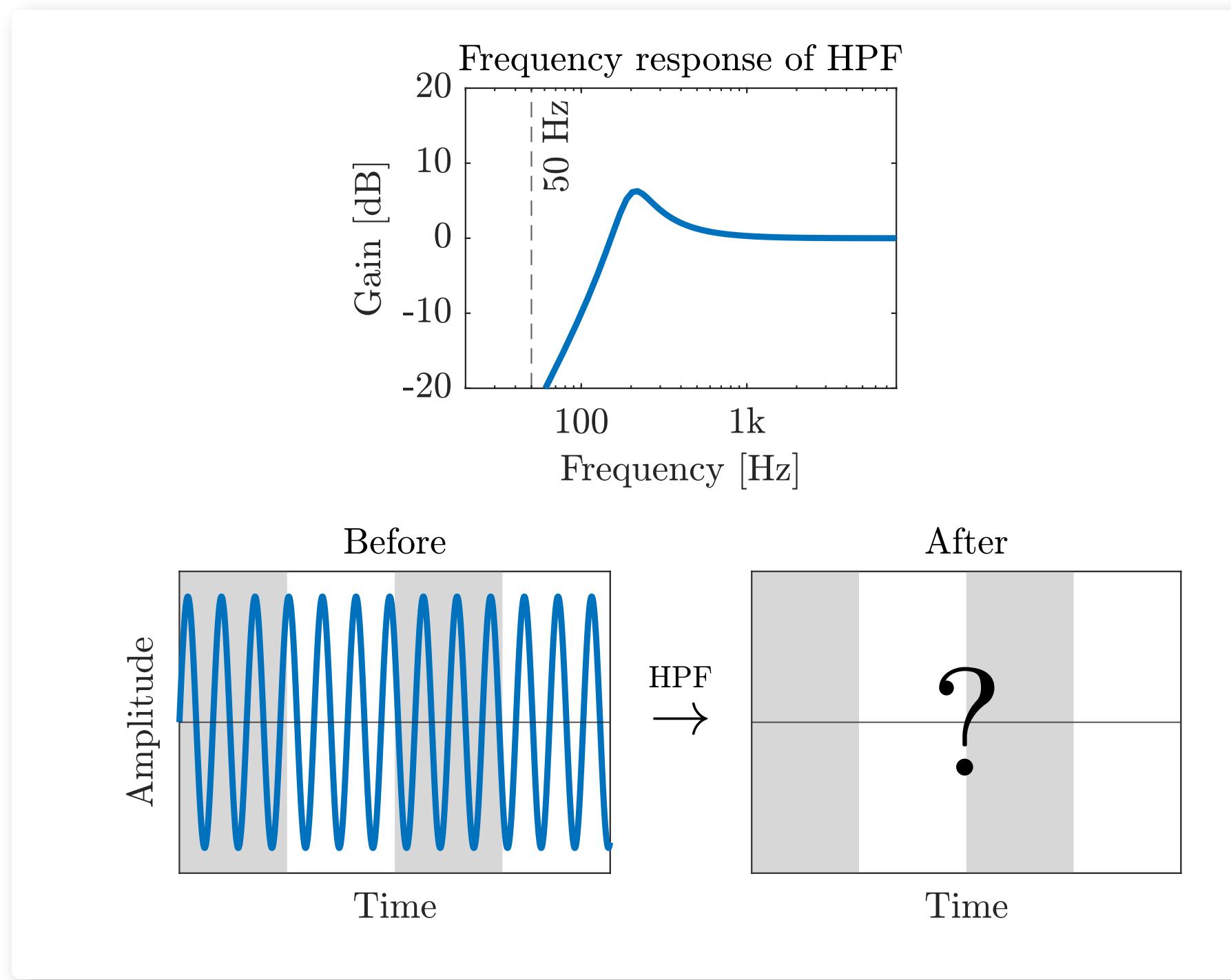


フレームベース処理

- 信号をリアルタイム処理したい
 - ▶ ある程度の長さのフレームに分割し、逐次的に処理
- フレームのつながりを意識した実装が必須
- イコライザの実装では、
遅延器の状態をフレーム間で伝搬する必要がある

画像 : <https://www.mathworks.com/help/dsp/ug/sample-and-frame-based-concepts.html>

フレームベースのフィルタの実装例



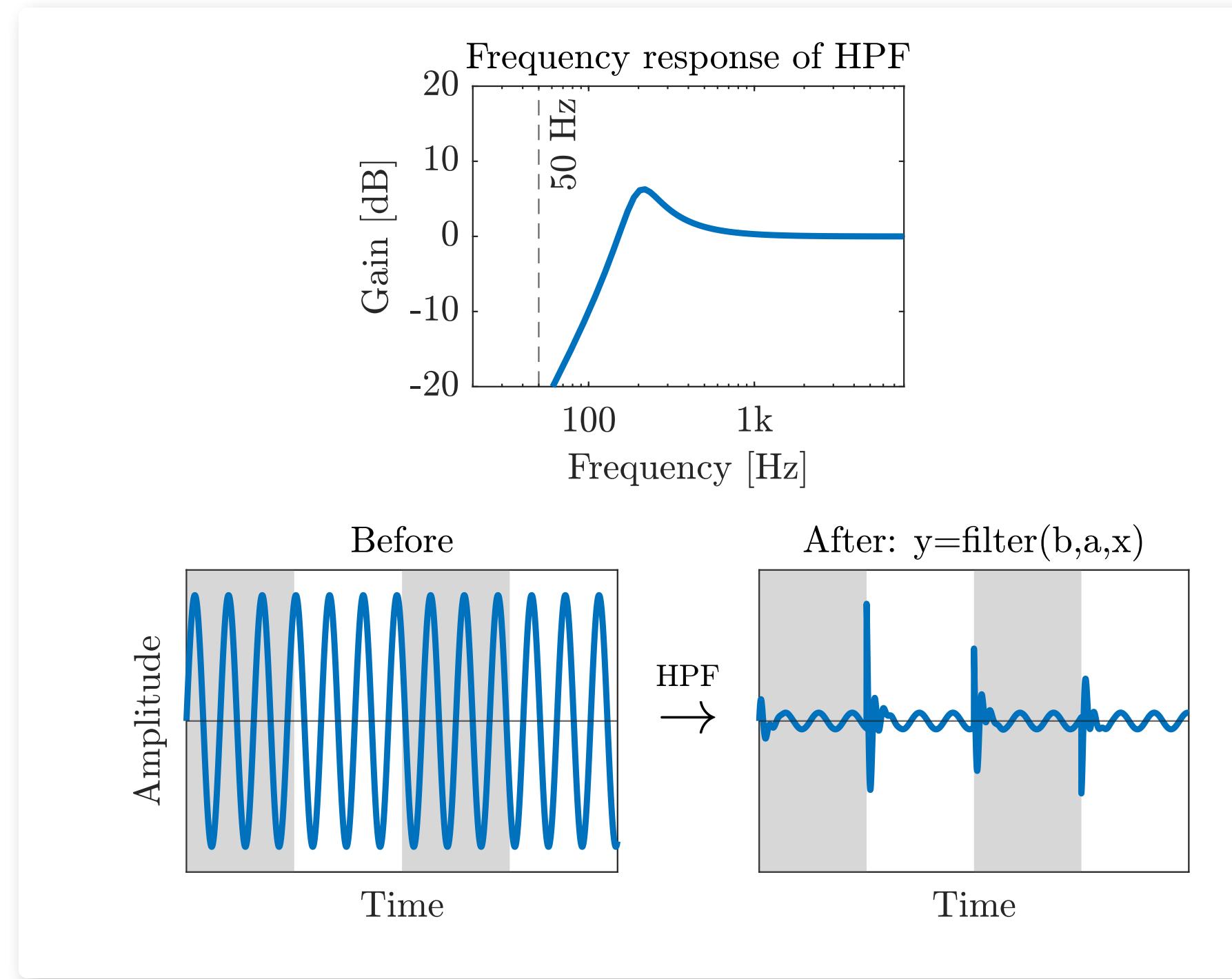
フレームベースのフィルタ適用例

以下のようなケースを考える

- 50 Hz の正弦波（左下）に HPF（上）を適用
- サンプリング周波数は 16 kHz , フレーム長は 1024

```
fs = 16000;
t = (0:4095)'/fs;
x = sin(2 * pi * 50 * t);
x1 = x(1:1024,:);
x2 = x(1025:2048,:);
x3 = x(2049:3072,:);
x4 = x(3073:4096,:);
```

フレームベースのフィルタの実装例



よくない実装の例（フレームの境界で不連続）

よくない実装の例： `filter` を前から順に適用

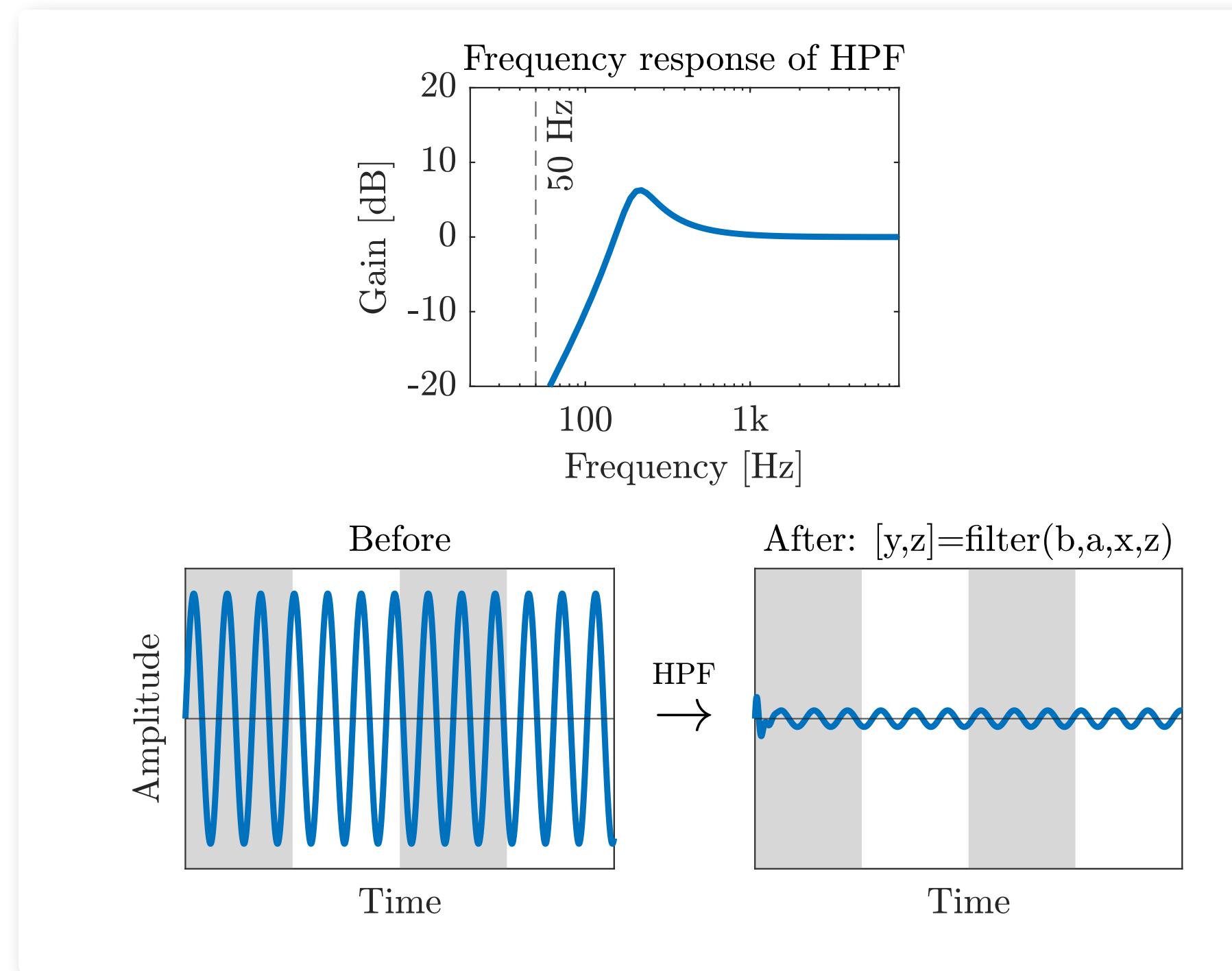
```

y1 = filter(b,a,x1); % 1 フレーム目
y2 = filter(b,a,x2); % 2 フレーム目
y3 = filter(b,a,x3); % 3 フレーム目
y4 = filter(b,a,x4); % 4 フレーム目
y = [y1;y2;y3;y4]; % 出力

```

✗ フレーム間で信号が途切れてしまう

フレームベースのフィルタの実装例



適切な実装の例

適切な実装の例 : 遅延器の状態 z を伝搬

```

z = zeros(2,1); % 遅延器の状態の初期化
[y1,z] = filter(b,a,x1,z); % 1 フレーム目
[y2,z] = filter(b,a,x2,z); % 2 フレーム目
[y3,z] = filter(b,a,x3,z); % 3 フレーム目
[y4,z] = filter(b,a,x4,z); % 4 フレーム目
y = [y1;y2;y3;y4]; % 出力

```

- z はフィルタ次数 (遅延器の数) × チャネル数の行列
- モノラルの双二次フィルタの場合は 2×1

プラグインの実装

フィルタ係数を計算する関数

Low Pass Filter (calcLPFCoeffs.m)

```
function [b,a] = calcLPFCoeffs(fc,fs,Q)
    wc      = 2*pi*fc/fs;
    alpha   = sin(wc)/(2*Q);
    coswc  = cos(wc);

    b = [ (1-coswc)/2, ...
           1-coswc, ...
           (1-coswc)/2];

    a = [1+alpha, ...
          -2*coswc, ...
          1-alpha];
end
```

フィルタ係数を計算する関数

High Pass Filter (calcHPFCoeffs.m)

```
function [b,a] = calcHPFCoeffs(fc,fs,Q)
    wc      = 2*pi*fc/fs;
    alpha   = sin(wc)/(2*Q);
    coswc  = cos(wc);

    b = [ (1+coswc)/2, ...
           -(1+coswc), ...
           (1+coswc)/2];

    a = [1+alpha, ...
          -2*coswc, ...
          1-alpha];
end
```

フィルタ係数を計算する関数

Band Pass Filter (calcBPFCoefs.m)

```
function [b,a] = calcBPFCoefs(fc,fs,Q)
    wc      = 2*pi*fc/fs;
    alpha   = sin(wc)/(2*Q);
    coswc  = cos(wc);

    b = [ Q*alpha, ...
            0, ...
        -Q*alpha];

    a = [1+alpha, ...
          -2*coswc, ...
        1-alpha];
end
```

プラグインのクラスを実装

プロパティの定義 (MATLAB_EQ.m から抜粋)

```
classdef MATLAB_EQ < audioPlugin

properties
    fs % サンプリング周波数
    fc = 2000 % カットオフ周波数
    Q = 1/sqrt(2) % Quality factor
    z = zeros(2,2) % フィルタの内部状態 (2次 x 2ch)
    type = 'LowPass'; % フィルタの種類
end
```

プラグインのクラスを実装

操作画面の定義 (1/2, MATLAB_EQ.m から抜粋)

```
properties(Constant)
    PluginInterface = ...
        audioPluginInterface( ...
            audioPluginParameter(...
                'type',... % プロパティ名 type
                'DisplayName','Type',...
                % Type
                'Mapping', {'enum','LowPass',...
                    'HighPass',...
                    'BandPass'} ... % フィルタの種類
            ),...
        )
```

プラグインのクラスを実装

操作画面の定義 (2/2, MATLAB_EQ.m から抜粋)

```
audioPluginParameter(...  
    'fc',... % プロパティ名 fc  
    'DisplayName','Cutoff Frequency',... % 画面に表示する名前  
    'Mapping',{'log',10,20000},... % 対数的に, 10 Hz から 20 kHz まで  
    'Label','Hz'... % 単位は Hz  
,...  
audioPluginParameter(...  
    'Q',... % プロパティ名 Q  
    'DisplayName','Quality Factor',... % 画面に表示する名前  
    'Mapping',{'log',0.025,20}... % 線形に 0.025 から, 40 まで  
)...  
);  
end
```

プラグインのクラスを実装

フィルタ処理を実装 (MATLAB_EQ.m から抜粋)

```
function y = process(p,x)
p.fs  = getSampleRate(p); % サンプリング周波数の取得

switch p.type % フィルタ係数の計算
    case 'LowPass'
        [b,a] = calcLPFCoeffs(p.fc,p.fs,p.Q);
    case 'HighPass'
        [b,a] = calcHPFCoeffs(p.fc,p.fs,p.Q);
    otherwise
        [b,a] = calcBPFCoeffs(p.fc,p.fs,p.Q);
end

[y,p.z] = filter(b,a,x,p.z); % フィルタの適用
end
```

おわりに

まとめ

- イコライザ（フィルタ）の実装
- フレームベースの処理

Further Practice

- **Audio EQ Cookbook**  に記載されている他の種類のフィルタを実装してみる

役立つ資料

- **Audio EQ Cookbook**  : イコライザで用いられるフィルタの導出
- **やる夫で学ぶディジタル信号処理**  : 信号処理全般