

デジタル信号処理の基礎

8

December 1, 2025

Assignment #7

iFFTを用いたBPF

clc; clear; close all;

==== 基本設定 ====
frameLength = 4096; % フレーム長
FFT_LENGTH = 4096; % iFFT 用のポイント数
Fs = 44100; % サンプリング周波数

==== 通過域設定（4–10 kHz） ====
Fp1 = 4000;
Fp2 = 10000;

=====

% ★★ ① 周波数領域で理想 BPF を作成 ★★

=====

H = zeros(FFT_LENGTH, 1);
f = (0:FFT_LENGTH-1) * (Fs / FFT_LENGTH);

% 正の周波数 & 対応する負側
idx = (f >= Fp1 & f <= Fp2) | ...
(f >= (Fs - Fp2) & f <= (Fs - Fp1));
H(idx) = 1;

=====

% ★★ ② iFFT → インパルス応答 h[n] (FIR係数) ★★

=====

h_full = real(ifft(H));

% フィルタ長 N を決めて切り出す

N = 200;
h = h_full(1:N+1);
% 窓（ハミング）をかけて滑らかに
h = h .* hamming(N+1);

=====

% ★★ ③ FIRフィルタとして dsp.FIRFilter を作成 ★★

=====

bandpassFIR = dsp.FIRFilter('Numerator', h);

==== フィルタ特性を表示 ====
fvtool(bandpassFIR, 'Fs', Fs, 'Color', 'White');

title('iFFT-based 4–10 kHz Bandpass Filter');

=====

% ★★ 固定パスで WAV を読み込む（あなたの指定ファイル） ★★

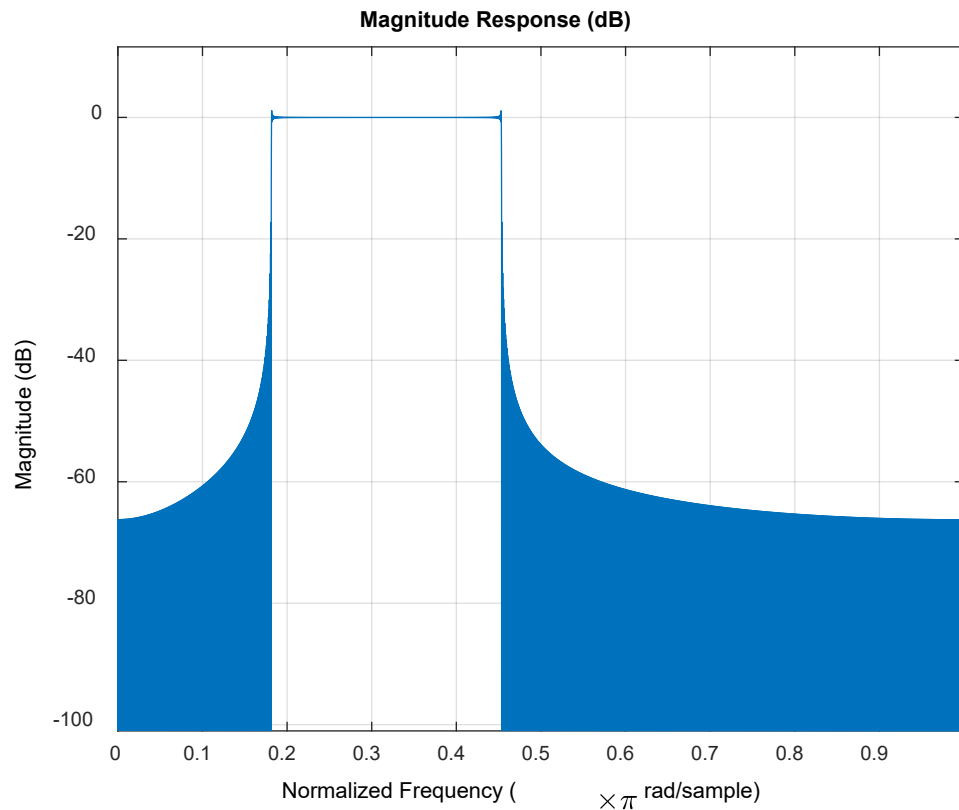
=====

fullpath = "/Users/daitaikiakaunto/Library/Mobile
Documents/com~apple~CloudDocs/Documents/Music/PioneerDJ/Demo Tracks/Demo Track 1.mp3";
fileReader = dsp.AudioFileReader(fullpath, ...
'SamplesPerFrame', frameLength);
deviceWriter = audioDeviceWriter('SampleRate', fileReader.SampleRate);

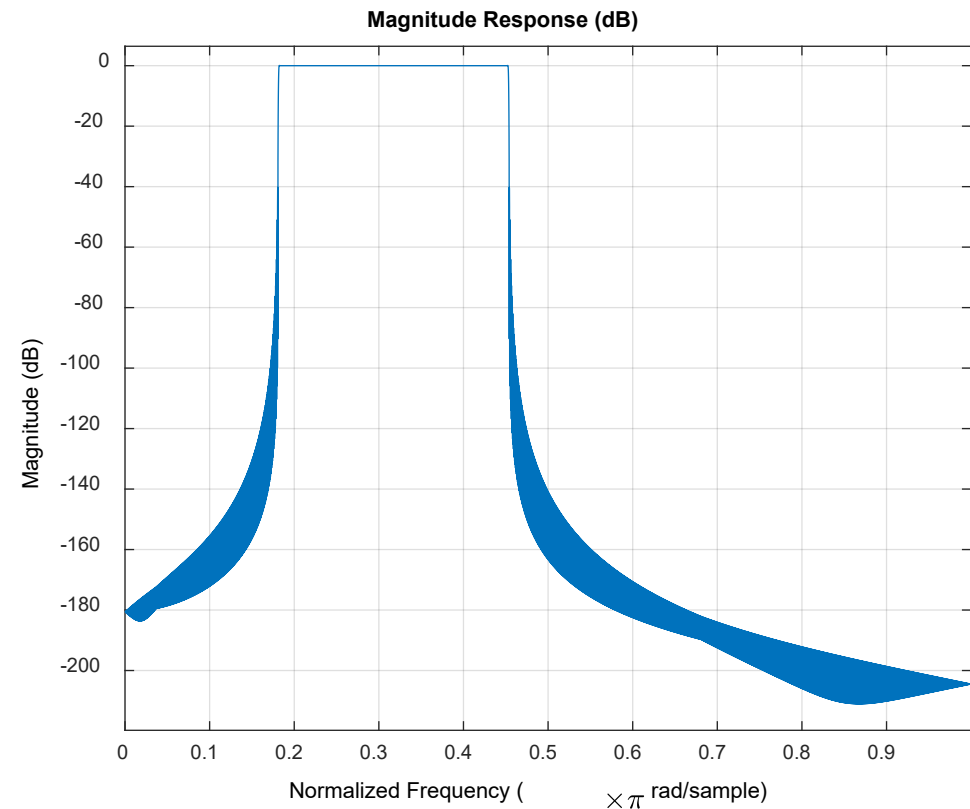
=====

```
% 富田萌, 72206043
% tomita_7.m
% iFFTを用いた 4-10 kHz バンドパスフィルタ + MP3 プレーヤー
clc; clear; close all;
%% === 基本設定 ===
frameLength = 4096;
Fs = 44100;
%% === 周波数領域でBPFを作ってiFFT ===
N = 4096; % FFTサイズ
f = (0:N-1) * Fs/N; % 周波数軸
H = zeros(1, N); % 周波数特性
% 4kHz~10kHzを1、それ以外を0にする
H(f >= 4000 & f <= 10000) = 1;
H(f >= Fs-10000 & f <= Fs-4000) = 1; % 対応する負の周波
数側
% iFFT → 実フィルタ化
h = real(ifft(H));
h = fftshift(h); % 中心揃え
h = h .* hann(N)'; % 窓で滑らかに
%% === FIR フィルタをシステム化 ===
BPF = dsp.FIRFilter('Numerator', h);
```

Windowの効果



Windowなし
パスバンド特性急峻

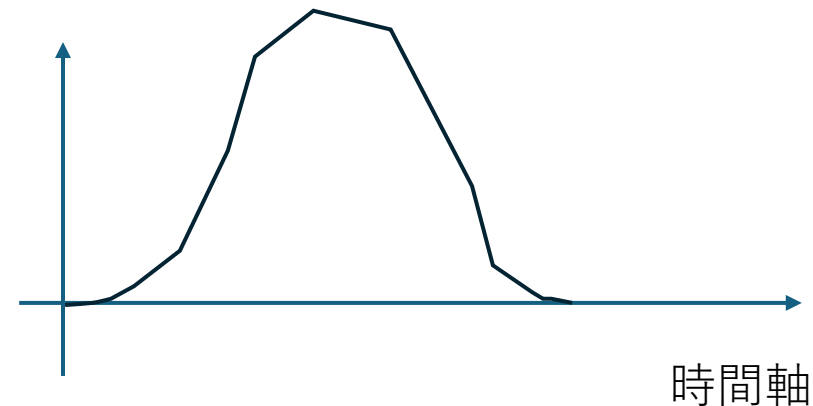


Han Windowあり
パスバンドが若干劣化
ストップバンドの抑圧量が良好

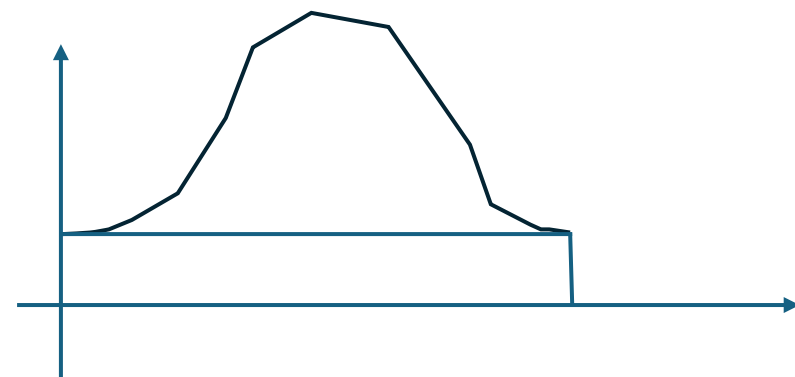
Hanning and Hamming

Hann window = a type of raised cosine

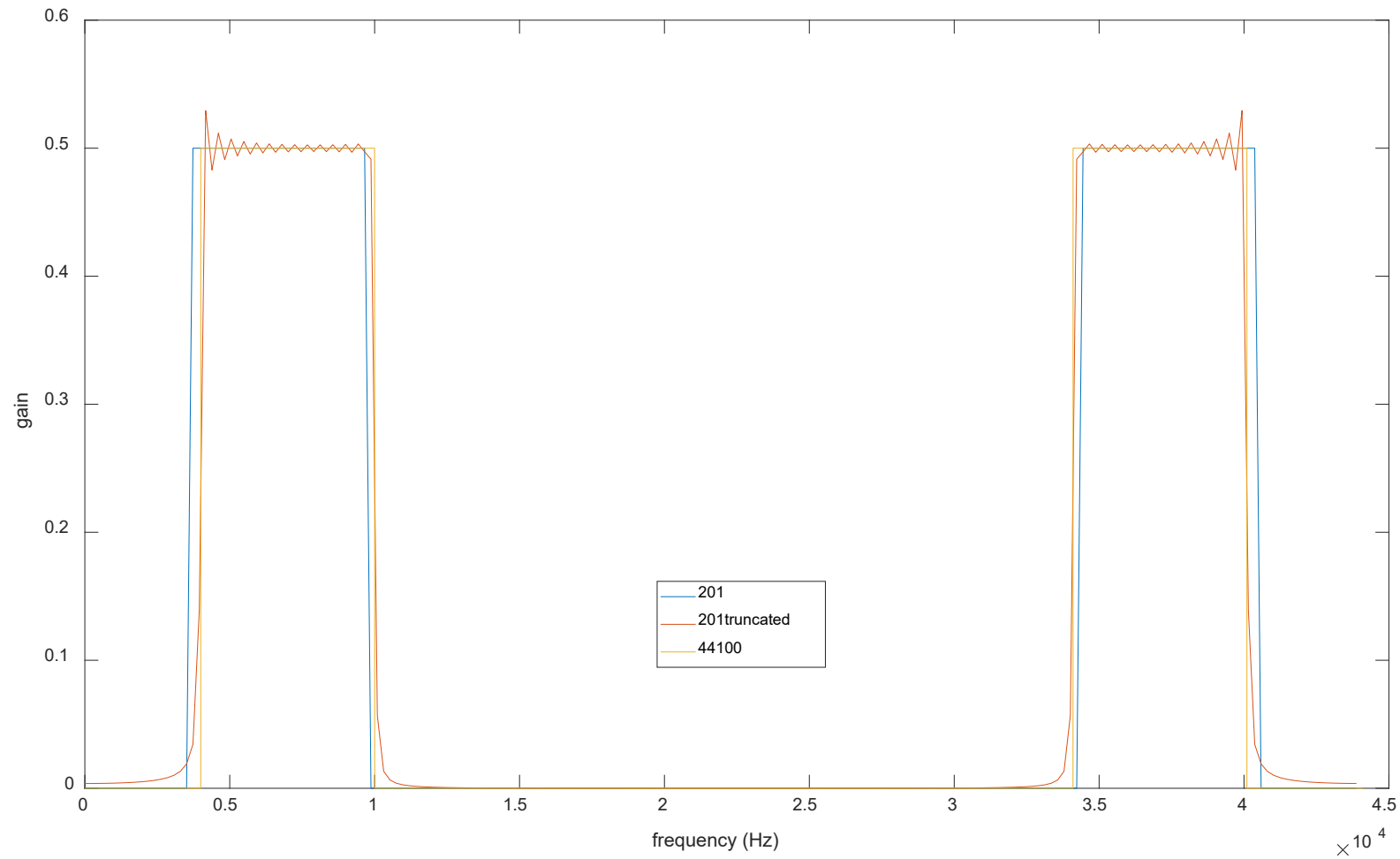
$$\text{Hann}_n = 0.5 - 0.5 \cos(2\pi n / (N-1))$$



Hamming window = 矩形Windowと Raised Cosineの組み合わせ
抑圧帯域での切れがよい



タップ数による周波数特性の違い



Advanced: 複素フィルタ

複素フィルタを用いると全帯域でのフィルタを実現

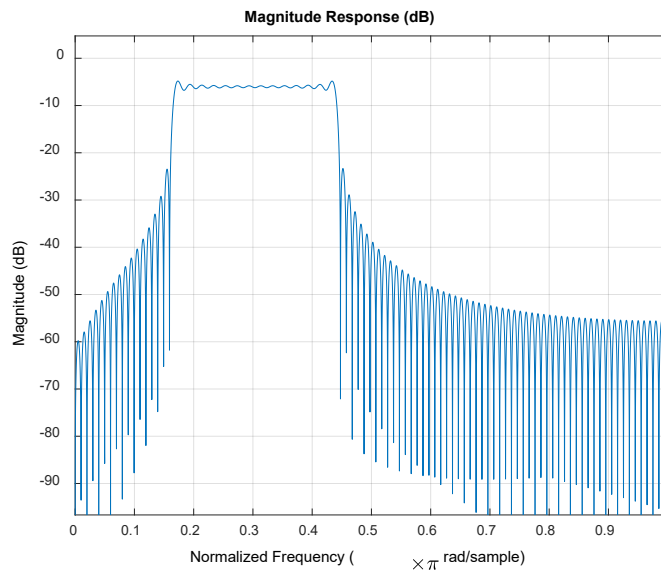
```
yt = ifft(f); % inverse FFT
```

```
L2 = floor(L/2);
```

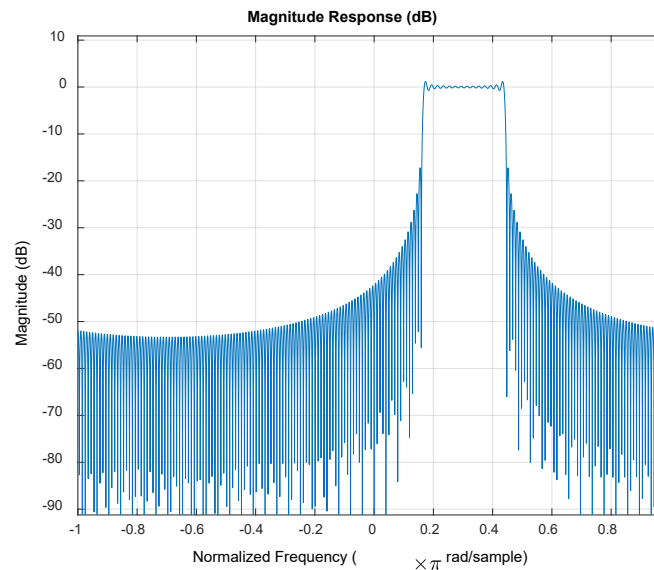
```
yt = [yt(end-L2:end) yt(1:L2)];
```

```
bpf = dsp.FIRFilter('Numerator', yt); % produce a complex filter
```

musicdsp_base_kyle



Real filter (always symmetric)



Complex filter (can be unsymmetric)

#assignment 8 画像のリサイズ

imresize関数を用いず、画像サイズを正の実数で指定して変更するMATLAB mスクリプトを作成する

仕様

- `uigetfile` 関数を用いて変形する画像を選択
- `input` 関数を用いて寸法縮小拡大係数（正の実数）を指定
- 変形前と変形後の写真を並べて表示せよ

`scaleimage`

2つのFigをサイズを変えないで並べるためには例えば`figure;`で以下のようにサイズ自動変更しないオプションを選択する

```
figure('Name', 'Original', 'NumberTitle', 'off', 'Resize', 'off') ;
```

12月8日（月）休講
次回は12月15日（月）です。