デジタル信号処理の基礎
#7

November 17, 2025

#6 assignment バンドパスフィルタ(BPF)

通過域が4kHz～10kHzのバンドパスフィルタを作成し、
Example 4.2のmusicdspに組み込みmファイルを提出せよ。

2つの問題

- firceqripはmスクリプトで使えるが、BPFオプションがない
- filterDesigner でBPFは設計できるが、係数を保存せねばならない。

```matlab
% 富田萌, 72206043
% musicdsp_bandpass.m
% 通過域が4kHz～10kHzのバンドパスフィルタを設計し、音声ストリーミングに適用

clc; clear; close all;

%=== 基本設定 ===%
frameLength = 4096; % フレーム長
N = 200; % フィルタ次数
Fs = 44100; % サンプリング周波数 [Hz]

%=== 通過域設定 ===%
Fp1 = 4000; % 下限 (4 kHz)
Fp2 = 10000; % 上限 (10 kHz)

% 正規化周波数（ナイキスト周波数=Fs/2で割る）
Wn = [Fp1, Fp2] / (Fs/2);

%=== FIRバンドパスフィルタ設計 ===%
eqnum = fir1(N, Wn, 'bandpass', hamming(N+1));

% dsp.FIRFilter オブジェクト作成
bandpassFIR = dsp.FIRFilter('Numerator', eqnum);

% フィルタ特性を表示
fvtool(bandpassFIR, 'Fs', Fs, 'Color', 'White');
title('4–10 kHz Bandpass FIR Filter');
fn = uigetfile('*.mp3');
%=== 音声ファイル読み込み設定 ===%
fileReader = dsp.AudioFileReader(...
fn, ... % ★使用する音声ファイルに変更
'SamplesPerFrame', frameLength);

deviceWriter = audioDeviceWriter('SampleRate', fileReader.SampleRate);

scope = spectrumAnalyzer( ...
'SampleRate', fileReader.SampleRate, ...
'NumInputPorts', 2, ...
'ShowLegend', true, ...
'ChannelNames', {'Original', 'Filtered'});

%=== ストリーミング処理 ===%
while ~isDone(fileReader)
signal = fileReader(); % 音声フレームを取得
yy = bandpassFIR(signal); % フィルタを適用
deviceWriter(yy); % 出力（再生）
scope(signal, yy); % スペクトラム表示
end

%=== リソース解放 ===%
release(fileReader);
release(deviceWriter);
release(scope);
```

```
frameLength = 4096;
N = 100; % FIR filter order
Fp = 1e3; % 1 kHz passband-edge frequency
Fs = 44100;
%filterDesigner
Num=[0.0054 0.0039 -0.0010 -0.0019 0.0013 0.0018 -
0.0036 -0.0081 -0.0045 0.0036 0.0062 0.0019 -0.0003
0.0050 0.0095 0.0036 -0.0080 -0.0113 -0.0041 5.4352e-04
-0.0056 -0.0114 -0.0027 0.0140 0.0183 0.0064 -0.0019
0.0061 0.0143 0.0011 -0.0239 -0.0297 -0.0098 0.0050 -
0.0067 -0.0197 0.0024 0.0453 0.0548 0.0165 -0.0146
0.0072 0.0357 -0.0159 -0.1363 -0.1876 -0.0608 0.1651
0.2783 0.1651 -0.0608 -0.1876 -0.1363 -0.0159 0.0357
0.0072 -0.0146 0.0165 0.0548 0.0453 0.0024 -0.0197 -
0.0067 0.0050 -0.0098 -0.0297 -0.0239 0.0011 0.0143
0.0061 -0.0019 0.0064 0.0183 0.0140 -0.0027 -0.0114 -
0.0056 0.0005 -0.0041 -0.0113 -0.0080 0.0036 0.0095
0.0050 -0.0003 0.0019 0.0062 0.0036 -0.0045 -0.0081 -
0.0036 0.0018 0.0013 -0.0019 -0.0010 0.0039 0.0054 -
0.0000 -0.0063 -0.0058 0.0011 0.0068 0.0055 -0.0003 -
0.0045 -0.0039 -0.0009 0.0011 0.0012 4.7959e-04]
bandpassFIR = dsp.FIRFilter('Numerator', Num);
% show the characteristics of the filter
fvtool(bandpassFIR, 'Fs', Fs, 'Color', 'White');
% specify an audio file
fileReader = dsp.AudioFileReader(...
'music.mp3',... % replace with a mp3 file
'SamplesPerFrame',frameLength);
deviceWriter = audioDeviceWriter(...
'SampleRate',fileReader.SampleRate);
scope = dsp.SpectrumAnalyzer('SampleRate',
fileReader.SampleRate);
while ~isDone(fileReader)
% acquire frame lenth audio stream
signal = fileReader();
% apply LPF by convolutional product
yy = bandpassFIR(signal);
% write to speaker
deviceWriter(yy);
% show
scope([signal,yy]);
end
release(fileReader);
release(deviceWriter);
release(scope);
```

Derive band pass filter(BPF) 係数をLPFとHPFから求める

LPF coefficients: ai i=0⋯N

HPF coefficients: bj j=0⋯M

$$y(n) = \sum_{i=0}^{N} a(i)x(n-i)$$

1st filtering

2nd filtering using the output of the 1st filtering

$$z(n) = \sum_{j=0}^{M} b(j)y(n-j) = \sum_{j=0}^{M}\sum_{i=0}^{N} a(i)b(j)y(n-j) = \sum_{j=0}^{M}\sum_{i=0}^{N} a(i)b(j)x(n-j-i)$$

$we\ introduce$

$$r = i + j \qquad 0 \le r \le N + M$$

$$z(n) = \sum_{r=0}^{N+M}\sum_{i=0}^{r} a(i)b(r-i)x(n-r)$$

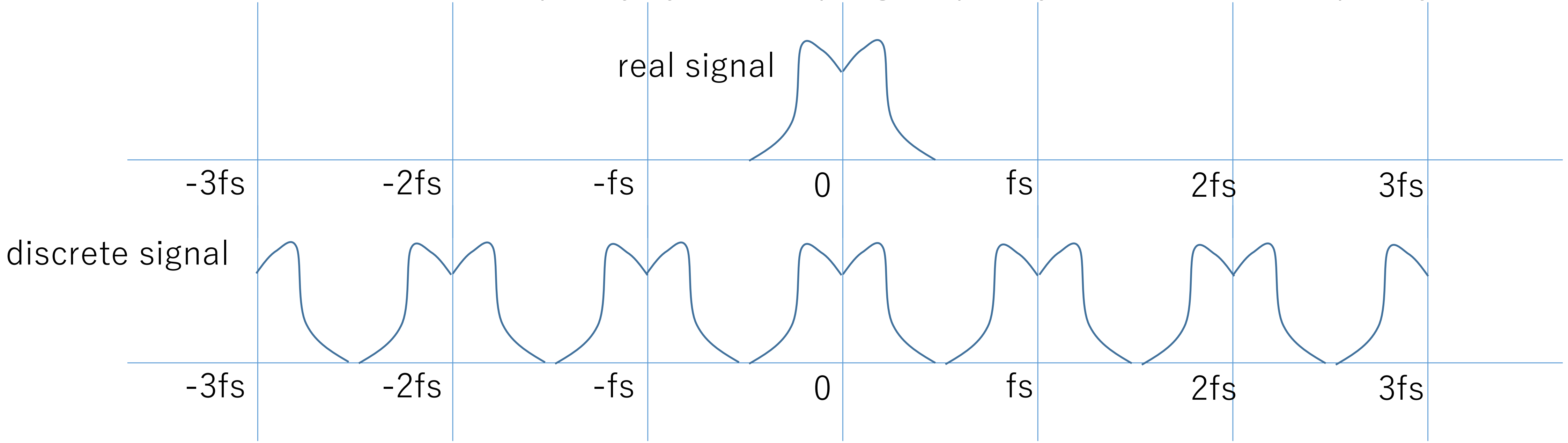
Overall filter coefficients can be produced by a convolutional products of the 1st and 2nd filter coefficients.
This can be done by “conv” facility.

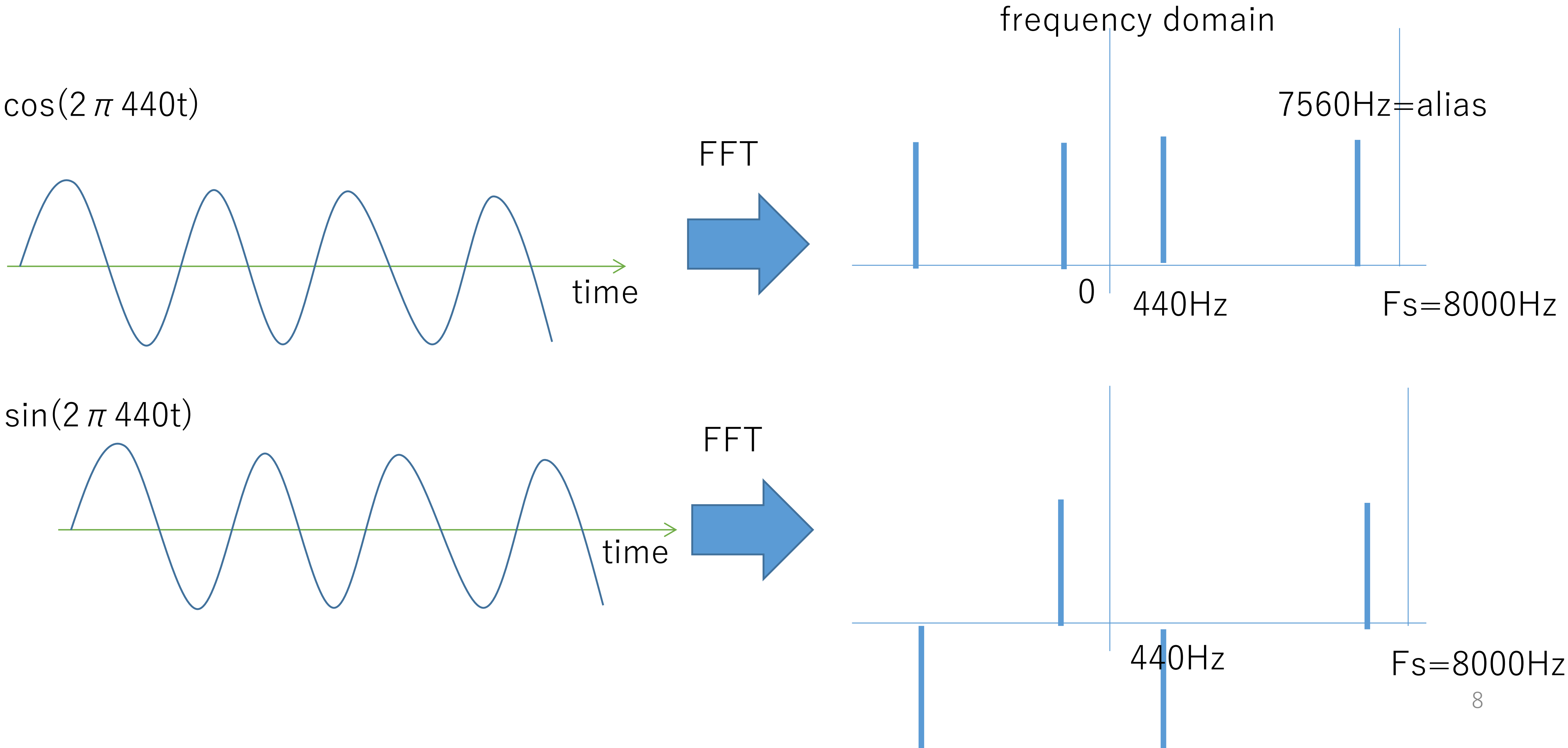
逆フーリエ変換

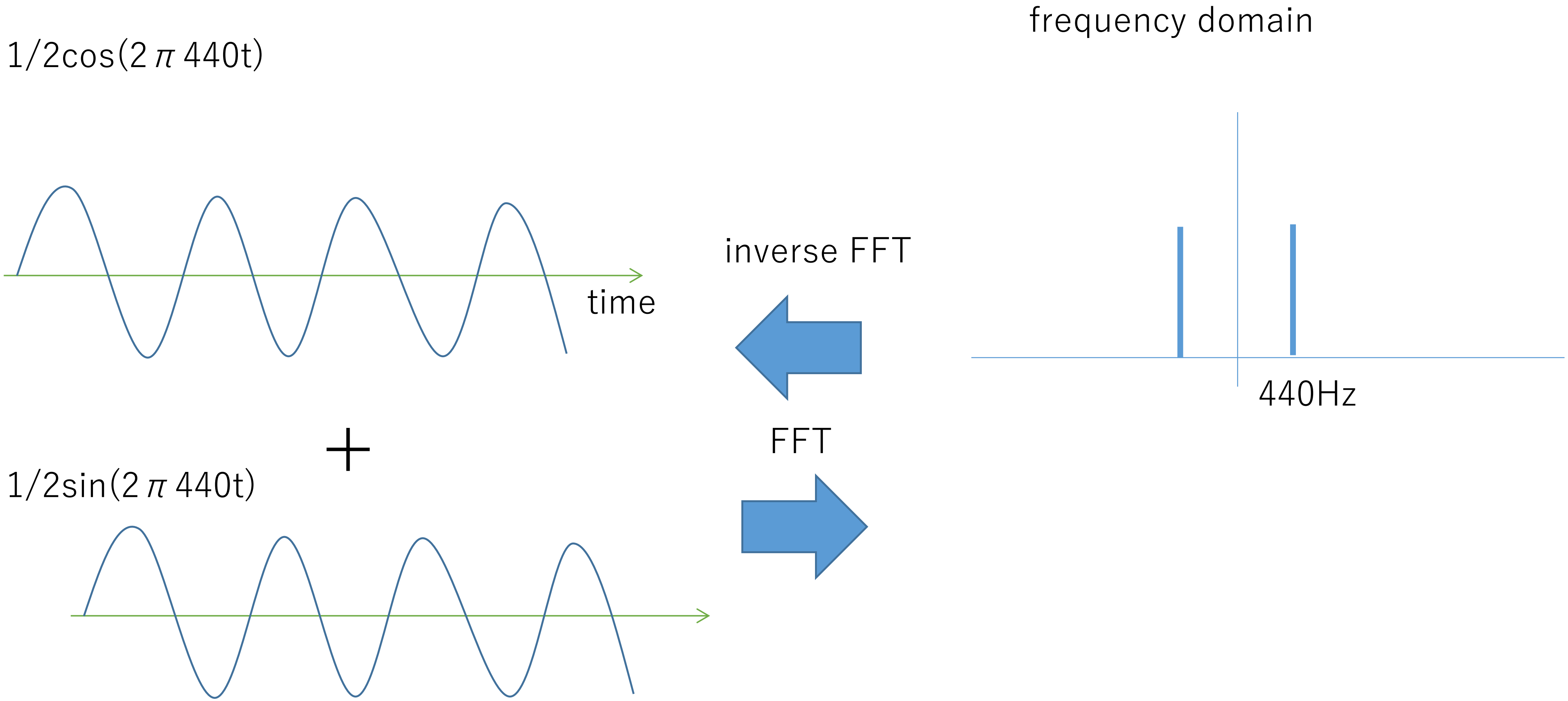
Filter function H(z)=H(jω)

フィルタの周波数特性が判れば、FIR は逆FFT (iFFT)で求めることができる。

Divide the real-world frequency by the sampling frequency = Normalized frequency




frequency domain
cos(2π440t)
FFT
time
7560Hz=alias
0
440Hz
Fs=8000Hz
sin(2π440t)
FFT
time
440Hz
Fs=8000Hz


1/2cos(2π440t)
time
+
1/2sin(2π440t)
inverse FFT
FFT
frequency domain
440Hz

Example 4.4 *(inverseA.m)*

The following script produce a time series of tone 440Hz for one second.

```
%% inverseA.m
%
%   genrate a 440Hz tone with inverse FFT.
%
Fs = 8000; % sampling rate
L =  8000; % sample length
f = zeros(L,1);
f(441) = L; % the first component is f=0
yt = ifft(f); % inverse FFT
sound(real(yt), Fs);
```

`ifft` *takes a real vector to represent the frequency components and produces a complex series of time signal. The real part represents the in-phase (cosine) component and the imaginary part represents the quadrature as is in Fourier Transform. In order to produce a sound we should take either the real part or the imaginary part. If we take the absolute value, it would produce no sound because the absolute value of the* `ifft` *result is constant because* $\cos^2 t + \sin^2 t = 1$.

Note that the lowest frequency after FFT is 0, rather than $\frac{1}{period}$*, 440 Hz frequency is located at* $f(441)$ *rather than* $f(440)$*. The alias of 440 Hz emerges at 7560 Hz when the sampling frequency is 8000 Hz as shown in the above example, The 7560 Hz component appears at* $f(7561)$*, accordingly.*

Time series produced from a designated filter shape

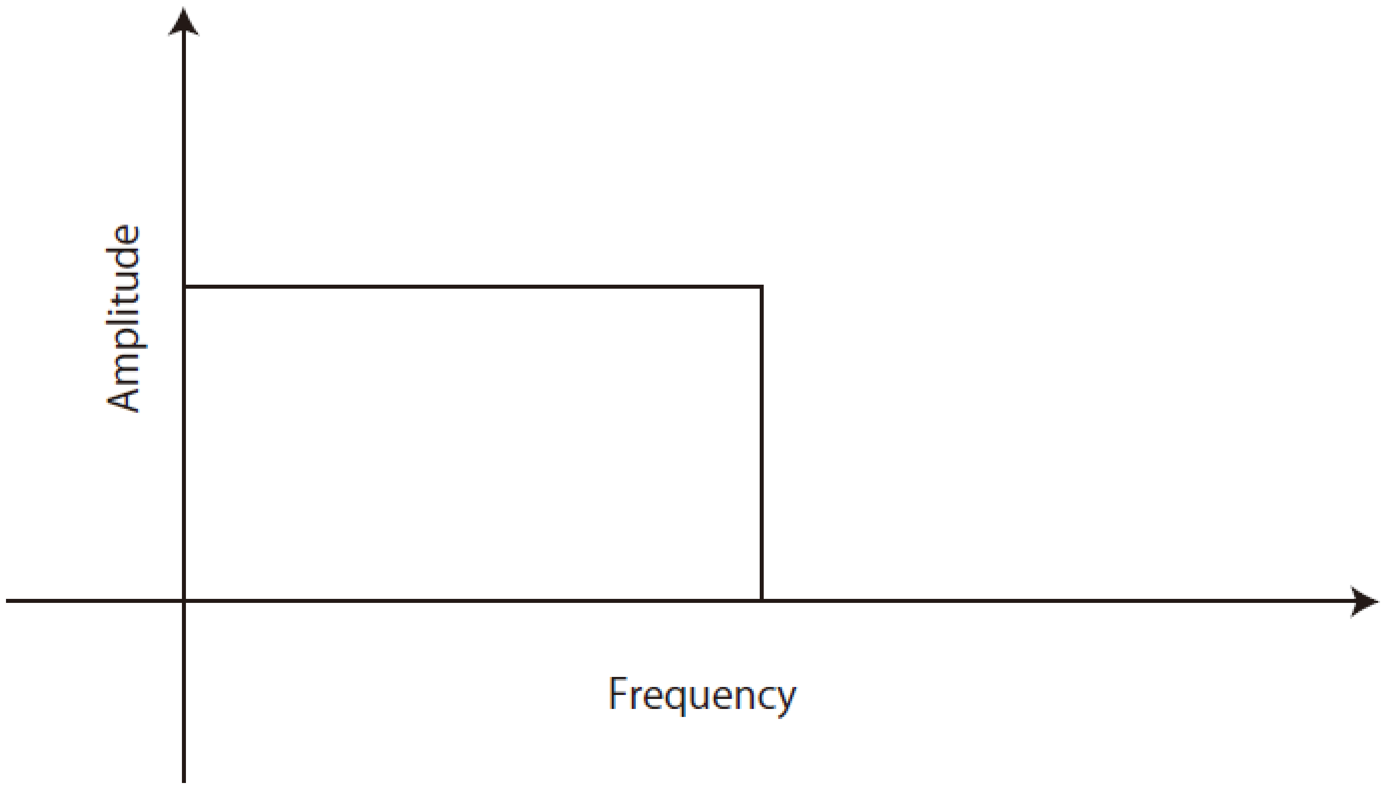


Figure 4.12: Allowing only a part of frequency components

Example 4.4 *(ifftlpf_base)*

The following script produces coefficients of LPF with 1kHz pass band edge.

```
%% ifftlpf_base
%
Fs = 44100; % sampling rate
L =  44100; % sample length
N = 1000; % passband edge

f = zeros(L,1);
for i=1:N
    f(i) = L ;
end
yt = ifft(f); % inverse FFT
plot(real(yt));
sound(real(yt), Fs);
```

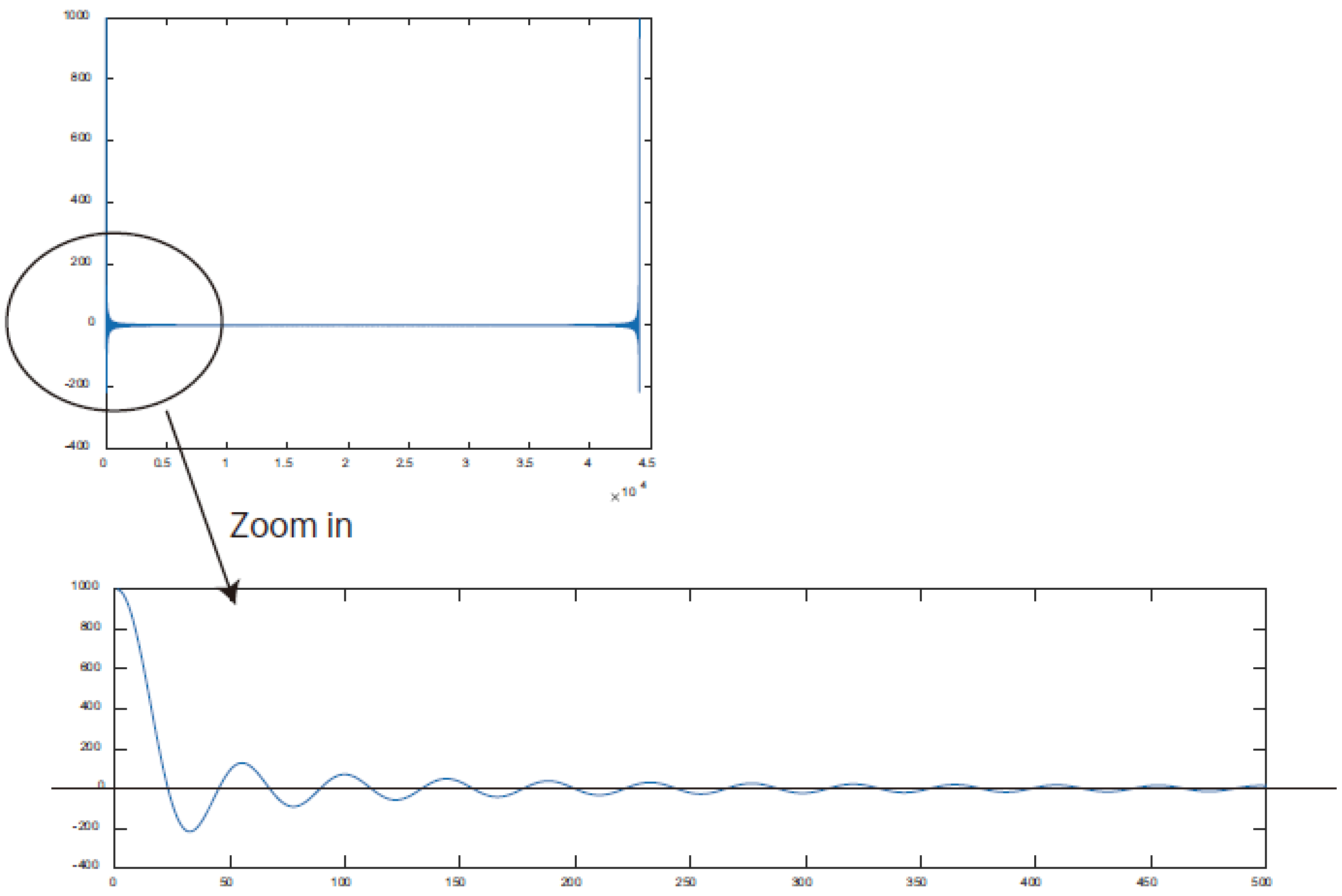


Figure 4.13: Ideal LPF coefficients

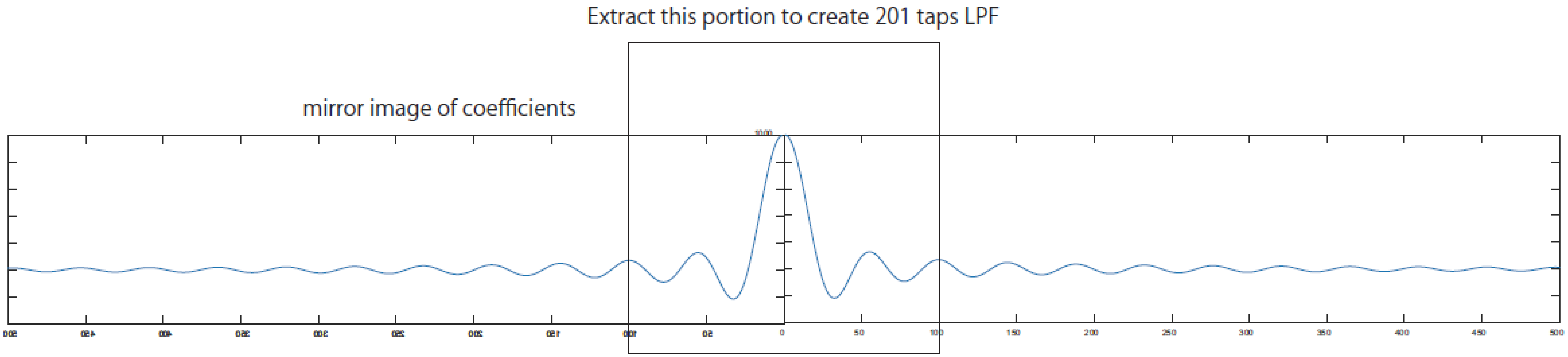


Figure 4.14: Extract LPF coefficients by concatenating mirror image

Assignment #7
iFFTを用いてBPFを作成する

- iFFTを用いて4kHz から 10kHzをパスバンドとするBPFを作成し、それを前回課題で作成したmp3プレーヤーに組み込め。

次回は１２月１日

- １１月２４日は三田祭休み