MATLAB Projects

Problem 7.36

Speech enhancement:
A digitally recorded speech in the noisy environment can be enhanced using a lowpass filter if the recorded speech with a sampling rate of 8000 Hz contains information within 1600 Hz.
Design a lowpass filter to remove the high-frequency noise above 1600 Hz with following filter specifications:

- passband frequency range: 0~1600 Hz;
- passband ripple: 0.02 dB;
- stop-band frequency range: 1800~4000 Hz;
- stop-band attenuation: 50 dB.

Use the designed low-pass filter to filter the noisy speech and adopt the following code to simulate the noisy speech:

```
load speech.dat
t=[0:length(speech)-1]*T;
th=mean(speech.speech)/4; %Noise power =(1/4) speech power
noise=sqrt(th)*randn([1,length(speech)]); %Generate Gaussian noise
nspeech=speech+noise; % Generate noisy speech
```

In this project, plot the speech samples and spectra for both noisy speech and the enhanced speech and use MATLAB sound() function to evaluate the sound qualities. For example, to hear the noisy speech:

```
sound(nspeech / max(abs(nspeech)), 8000);
```

solution

Look up **Table 7.7** in page 254, based on the requirements:

- Passband ripple: 0.02 dB;
- Stop-band attenuation: 50 dB.

1. Window method: **Hamming**
2. Compute the transition band

$$\Delta f = \frac{|f_{pass} - f_{stop}|}{f_s} = (1800 - 1600)/8000$$

Then use formula in **Table 7.7**

$$N = \frac{3.3}{\Delta f} = 132$$

Find the closest odd number **133**

3. Then estimate the cutoff frequency

$$f_c = (f_{pass} + f_{stop})/2$$

The cutoff frequency are **1700** Hz

4. $2 \times M + 1 = 133 \Rightarrow M = 66$

$\Omega_c = 2\pi \times \frac{f_c}{f_s} = 1.3352$

$$h(n) = \begin{cases} \frac{\Omega_c}{\pi} & \text{for } n = 0 \\ \frac{\sin(\Omega_c n)}{n\pi} & \text{for } n \neq 0 \quad -M \leq n \leq M \end{cases}$$

$h(-M) \rightarrow h(M)$ are

```
[0.000754, -0.004524, -0.002923, 0.003281, 0.004574,
-0.001218, -0.005305, -0.001259, 0.00489, 0.003627, -0.003341,
-0.005347, 0.000922, 0.005987, 0.001892, -0.005322, -0.004502,
0.003394, 0.006307, -0.000531, -0.006835, -0.002707, 0.005853,
0.005629, -0.003441, -0.007549, -0.0, 0.007936, 0.003803,
-0.006542, -0.007153, 0.00348, 0.009247, 0.000757, -0.00946,
-0.005365, 0.007503, 0.009359, -0.003513, -0.011753,
-0.001915, 0.011763, 0.007796, -0.008988, -0.012892, 0.003538,
0.015915, 0.003911, -0.015756, -0.01216, 0.011694, 0.019605,
-0.003557, -0.02441, -0.008197, 0.024673, 0.022508, -0.01848,
-0.037841, 0.003568, 0.052398, 0.024362, -0.06438, -0.080682,
0.072255, 0.309515, 0.425, 0.309515, 0.072255, -0.080682,
-0.06438, 0.024362, 0.052398, 0.003568, -0.037841, -0.01848,
0.022508, 0.024673, -0.008197, -0.02441, -0.003557, 0.019605,
0.011694, -0.01216, -0.015756, 0.003911, 0.015915, 0.003538,
-0.012892, -0.008988, 0.007796, 0.011763, -0.001915,
-0.011753, -0.003513, 0.009359, 0.007503, -0.005365, -0.00946,
0.000757, 0.009247, 0.00348, -0.007153, -0.006542, 0.003803,
0.007936, -0.0, -0.007549, -0.003441, 0.005629, 0.005853,
-0.002707, -0.006835, -0.000531, 0.006307, 0.003394,
-0.004502, -0.005322, 0.001892, 0.005987, 0.000922, -0.005347,
-0.003341, 0.003627, 0.00489, -0.001259, -0.005305, -0.001218,
0.004574, 0.003281, -0.002923, -0.004524, 0.000754]
```

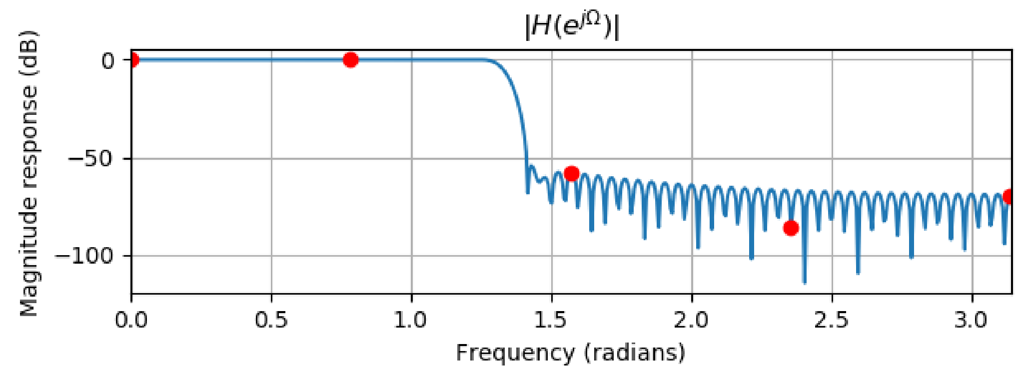
5. Hamming window function

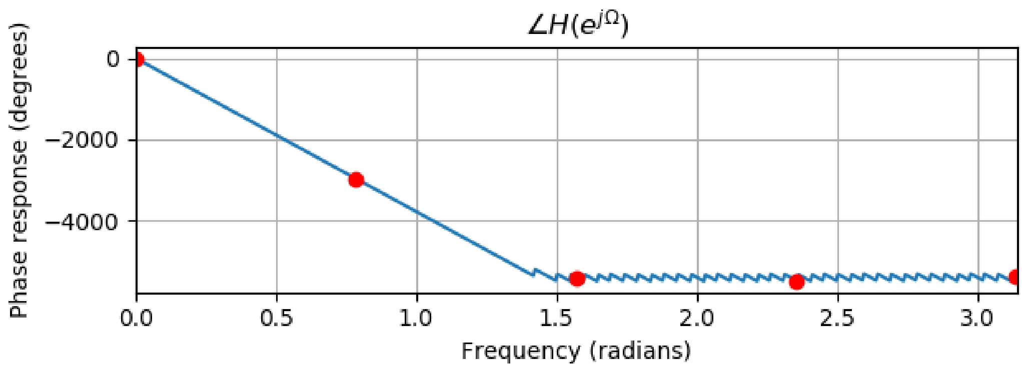
$$w_{\text{ham}}(n) = 0.54 + 0.46 \cos\left(\frac{n\pi}{M}\right), -M \leq n \leq M$$

$h_w(n) = h(n) \cdot w(n)$ are

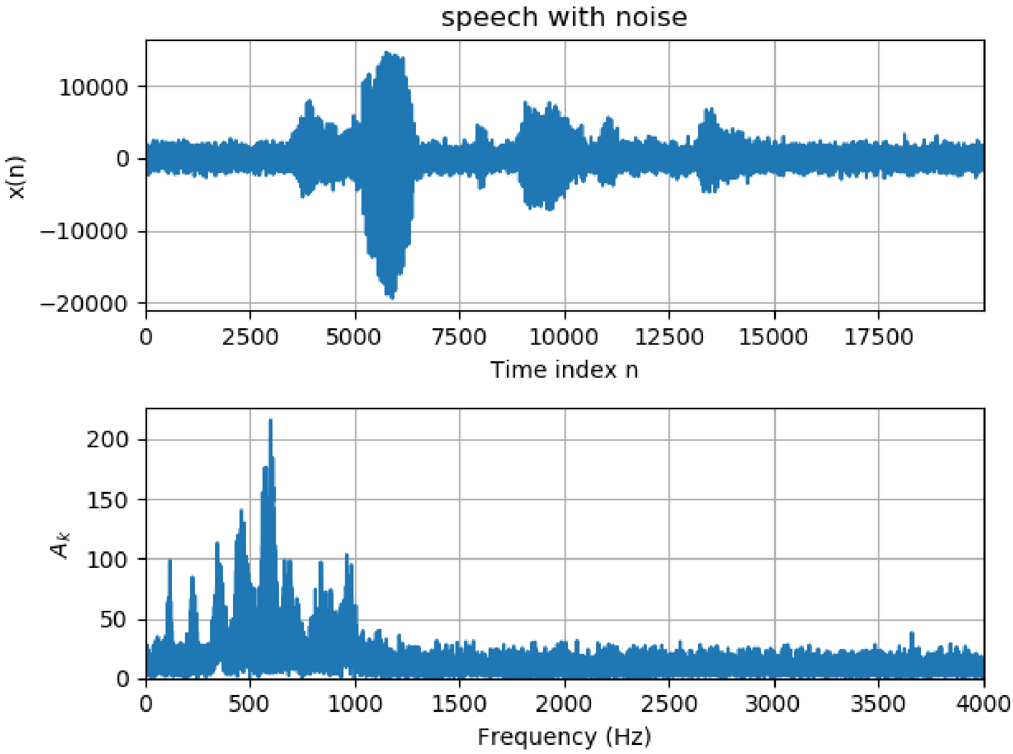
```
[6e-05, -0.000364, -0.00024, 0.000278, 0.000404, -0.000113,
-0.000523, -0.000133, 0.000552, 0.000441, -0.000438,
-0.000757, 0.000141, 0.00099, 0.000337, -0.001024, -0.000932,
0.000755, 0.001506, -0.000136, -0.001867, -0.000789, 0.001814,
0.001853, -0.0012, -0.002786, -0.0, 0.003257, 0.001641,
-0.002963, -0.003394, 0.001727, 0.004791, 0.000409, -0.005316,
-0.003132, 0.004543, 0.005868, -0.002278, -0.00787, -0.001322,
0.008363, 0.005699, -0.006748, -0.009927, 0.002791, 0.012841,
0.003224, -0.013255, -0.010427, 0.010208, 0.017403, -0.003207,
-0.022328, -0.007598, 0.023153, 0.021357, -0.017711,
-0.036594, 0.003477, 0.051422, 0.024046, -0.063844, -0.080304,
0.072104, 0.309354, 0.425, 0.309354, 0.072104, -0.080304,
-0.063844, 0.024046, 0.051422, 0.003477, -0.036594, -0.017711,
0.021357, 0.023153, -0.007598, -0.022328, -0.003207, 0.017403,
0.010208, -0.010427, -0.013255, 0.003224, 0.012841, 0.002791,
-0.009927, -0.006748, 0.005699, 0.008363, -0.001322, -0.00787,
-0.002278, 0.005868, 0.004543, -0.003132, -0.005316, 0.000409,
0.004791, 0.001727, -0.003394, -0.002963, 0.001641, 0.003257,
-0.0, -0.002786, -0.0012, 0.001853, 0.001814, -0.000789,
-0.001867, -0.000136, 0.001506, 0.000755, -0.000932,
-0.001024, 0.000337, 0.00099, 0.000141, -0.000757, -0.000438,
0.000441, 0.000552, -0.000133, -0.000523, -0.000113, 0.000404,
0.000278, -0.00024, -0.000364, 6e-05]
```

The magnitude and phase of $H(e^{j\Omega})$ are

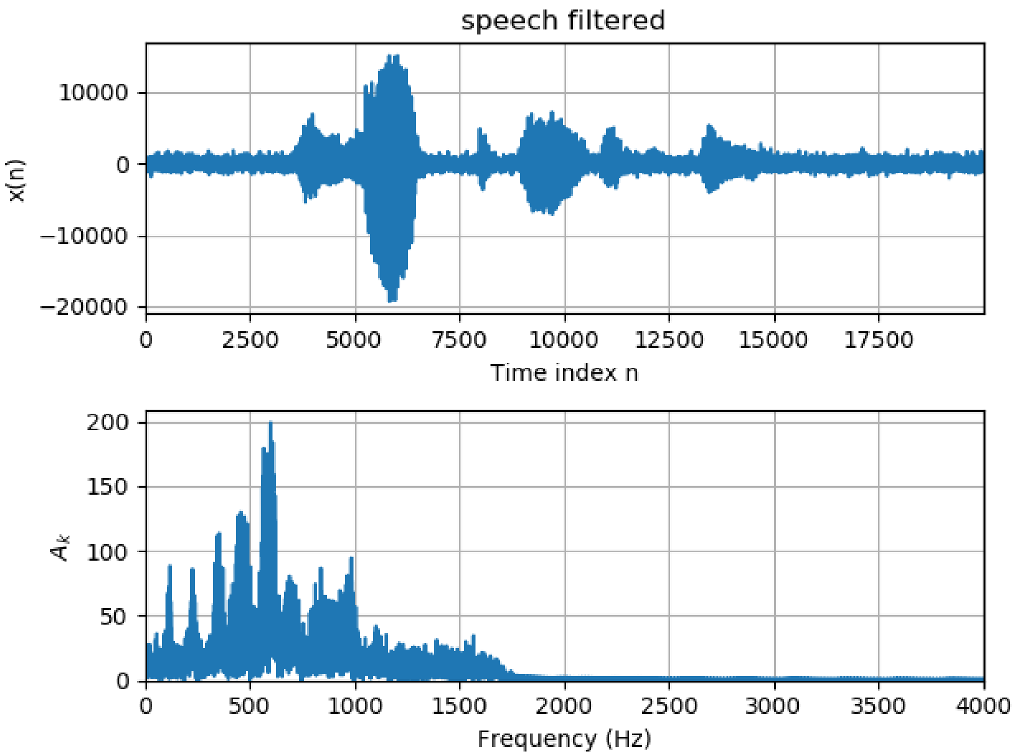




6. Speech with noise:



Speech filtered:



comment

1. The low-pass filter eliminates the high-frequency component of noise.
2. The quality of speech improves
3. The results and figures (except Problem 7.19, 7.20) are generated by **Python** scripts

   I used the python library written on my own to replace the function in MATLAB.

Here is the main script for problem 7.36

```
from fir_filter.choose_window_type import *
from fir_filter.calc_window_len import *
from fir_filter.calc_freq_cutoff import *
from fir_filter.fir_filter import *
from fir_filter.window import *
from fir_filter.calc_mag_angle import *
from fir_filter.filter import *
from fir_filter.fft1d import *

# choose window type
passband_ripple = 0.02
stopband_attenuation = 50
str_window_type = choose_window_type(passband_ripple,
stopband_attenuation)
print(str_window_type)
# calc length of window
f_s = 8000
list_transient_band = [ [1600, 1800] ]
window_len = calc_window_len(str_window_type,
list_transient_band, f_sample=f_s)
print(window_len)
# calc the cutoff frequency & normalized cutoff frequency
list_freq_cutoff = calc_freq_cutoff(list_transient_band)
print(list_freq_cutoff)
print_approx([calc_omega(list_freq_cutoff[0], f_s)])
# calc h(-M)~h(M) list of filter
list_filter = fir_filter(list_freq_cutoff, f_s, window_len,
"low_pass")
print_approx(list_filter, precision=6)
# Hamming window: calc h_w(n) = h(n) * w(n)
str_window_type = "Hamming"
path_fig = "../p7_36_hamm.png"
list_filter_window = window(list_filter,
str_window_type=str_window_type)
print_approx(list_filter_window, precision=6)
# plot Magnitude & Phase of H(e^{j Omega})
list_mag, list_angle, list_omega =
calc_mag_angle(list_filter_window)
plot_mag_angle(list_mag, list_angle, list_omega,
path_fig=path_fig)
# read data: "speech.dat"
import numpy as np
filename = "./speech.dat"
speech = np.loadtxt(filename)
```

```
th = np.mean( speech**2 ) / 4 # Noise power =(1/4) speech
power
noise = np.sqrt(th) * np.random.randn( len(speech) ) #
Generate Gaussian noise
nspeech = speech + noise # Generate noisy speech
# low-pass filter to filter the noisy speech
nspeech_filtered = filter(nspeech, list_filter_window)
nspeech_filtered = np.asarray(nspeech_filtered)
# save sounds
from scipy.io.wavfile import write
write("../nspeech.wav", f_s, nspeech)
write("../nspeech_filtered.wav", f_s, nspeech_filtered)
# plot speech with noise & filtered speech
plot_spectrum(nspeech, f_s, path_fig="../p7_36_noise.png",
str_title="speech with noise")
plot_spectrum(nspeech_filtered, f_s,
path_fig="../p7_36_filtered.png", str_title="speech filtered")
```