Seung Jun Baek

Topics: FIR filter design

1. The frequency response of a desired filter $h_d[n]$ is shown in Fig. 1. In this problem, we wish to design an $(M+1)$-point causal linear-phase FIR filter $h[n]$ that minimizes the integral-squared error

$$\epsilon^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} |A(e^{j\omega}) - H_d(e^{j\omega})|^2 d\omega,$$

where the frequency response of the filter $h[n]$ is

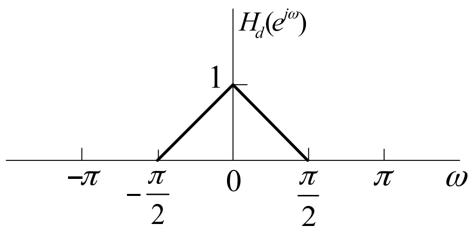

Fig 1:

$$H(e^{j\omega}) = A(e^{j\omega})e^{-j\omega M/2}$$

and M is an even integer

(a) Determine $h_d[n]$.

(b) What symmetry should $h[n]$ have in the range $0 \le n \le M$? Briefly explain your reasoning.

(c) Determine $h[n]$ in the range $0 \le n \le M$.

(d) Determine an expression for the minimum integral-squared error ϵ^2 as a function of $h_d[n]$ and M.

Sol:

(a)

$$H_d(e^{jw}) = \text{rect}\Big(\frac{2w}{\pi}\Big) * \text{rect}\Big(\frac{2w}{\pi}\Big)$$

$$\text{rect}\Big(\frac{2w}{\pi}\Big) \overset{\mathcal{F}}{\leftrightarrow} \frac{1}{4}\text{sinc}\Big(\frac{n}{4}\Big)$$

$$H_d(e^{jw}) \overset{\mathcal{F}}{\leftrightarrow} h_d[n] = \Big(\frac{1}{4}\text{sinc}\Big(\frac{1}{4}\Big)\Big)^2$$

(b) $h[n]$ must be symmetric with respect to $n = \frac{N-1}{2}$ due to the linear phase requirement.

(c) Shifting $h[n]$ by $\frac{N-1}{2}$ because $h[n]$ should be causal.

$$h[n] = \Big(\frac{1}{4}\text{sinc}\Big(\frac{1}{4}\Big(n - \frac{N-1}{2}\Big)\Big)\Big)^2 w[n], \quad \text{where } w[n] = \begin{cases} 1, & 0 \le n \le N-1 \\ 0, & else \end{cases}$$

(d)

$$\epsilon = \frac{1}{2\pi}\int_{-\pi}^{\pi} |A(e^{jw} - H_d(e^{jw})|^2 dw$$

Using Parseval's theorem

$$\begin{aligned}\epsilon &= \sum_{-\infty}^{\infty} |a[n] - h_d[n]|^2 \\ &= 2\sum_{0}^{\infty} |a[n] - h_d[n]|^2 \\ &= 2\sum_{\frac{N-1}{2}+1}^{\infty} |h_d[n]|^2\end{aligned}$$

$$a[n] = \begin{cases} h_d[n], & -\frac{N-1}{2} \leq n \leq -\frac{N-1}{2} \\ 0, & else \end{cases}$$

2. Consider the function $D(x) = e^x$ and

$$H(x) = \sum_{k=0}^{L} b_k x^k$$

where $L = 1$, so $H(x)$ is a first-order polynomial function. In the interval $0 \leq x \leq 1$, find the coefficients b_0, b_1 that minimizes the maximum error between $H(x)$ and $D(x)$,

$$\max_{0\leq x\leq 1} |H(x) - D(x)|.$$

(Hints: The number of points where the maximum error occurs is $L + 2 = 3$ where two of the m occur at $x = 0$ and $x = 1$. Also note that you need three points to solve for two coefficients b_0 and b_1.)

Sol:

$$H(x) = \sum_{k=0}^{1} b_k x^k = b_0 + b_1 x, D(x) = e^x$$

$$|H(0) - D(0)| = |H(x_0) - D(x_0)| = |H(1) - D(1)|$$

$$1 - b_0 = b_0 + b_1 x_0 - e^{x_0} = e - b_0 - b_1$$

$$1 - b_0 = e - b_0 - b_1 \Rightarrow b_1 = e - 1$$

$$(b_0 + b_1 x - e^x)' = b_1 - e^x = 0$$

$$x_0 = \ln(e - 1)$$

$$1 - b_0 = b_0 + b_1 x_0 - e^{x_0}$$

$$2b_0 = 2 - (e - 1)\ln(e - 1)$$

$$b_0 = \frac{e - (e - 1)\ln(e - 1)}{2}$$

$$H(x) = \frac{e - (e - 1)\ln(e - 1)}{2} + (e - 1)x$$

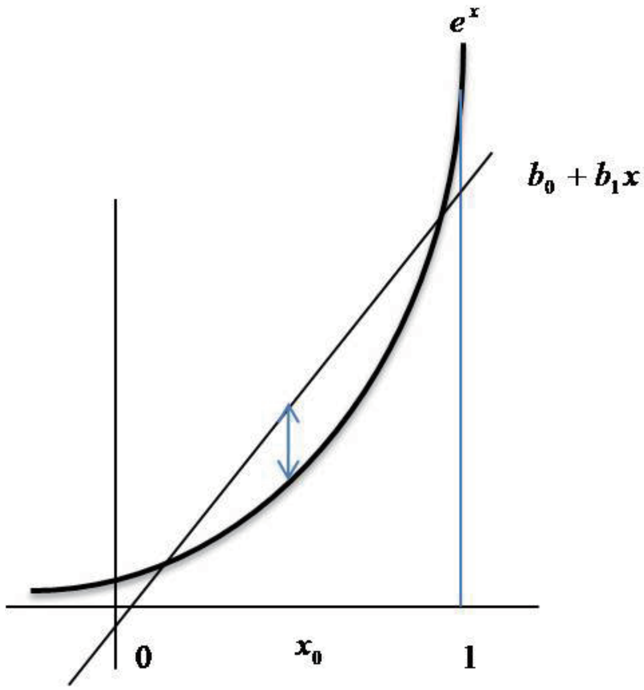

Figure 2:

3. An optimal equiripple FIR linear-phase filter wa designed by the Parks-McClellan algorithm. The magnitude of its frequency response is shown in Fig. 3. The maximum approximation error in the passband and stopband is $\delta_1 = \delta_2 = 0.085$. The passband and stopband cutoff frequencies are $\omega_p = 0.4\pi$ and $\omega = 0.58\pi$.

   (a) Carefully sketch the approximation error; i.e., sketch

$$E(\omega) = H_d(e^{j\omega}) - A_e(e^{j\omega}).$$

   (Note that Fig. 3 shows $|A_e(e^{j\omega})|$.)

   (b) What is the length of the impulse response of the system?

Sol:

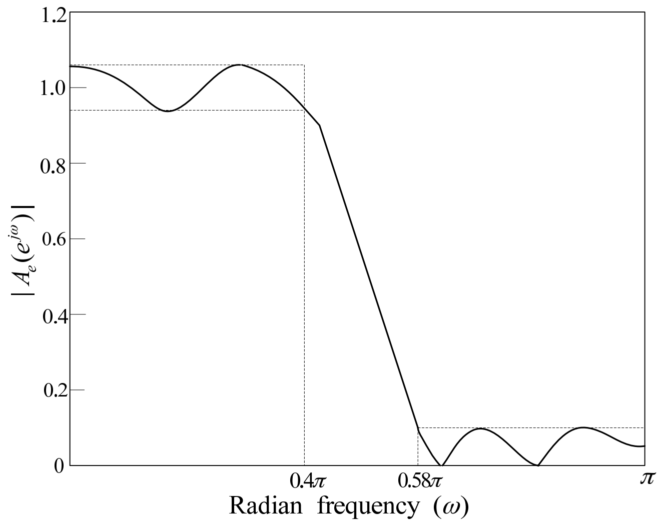

Figure 3:

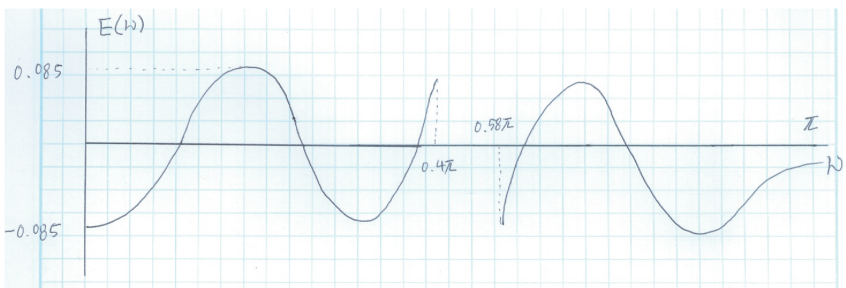

Figure 4:

(a) The plot is given by Fig. 3

(b) Since this filter does not have a alternation at $w = \pi$. So number of alternation $= L + 2 = 7, \;\; L = 5$
filter length $= 2L + 1 = 11$

4. **(Migrating to Python.)** From now on, we will use Python for computation and plotting for signal processing. Python is a free tool where Matlab is proprietary software – that is you have to pay for the tool. Typically the development environment like IDE, Matlab is convenient, but other open-source benefits can be obtained from Python, especially if you have to organize a big project because Python is a well structured, general purpose programming language (compared with Matlab built for matrix operation).

   An example website for Matlab to Python is

   `https://towardsdatascience.com/python-for-matlab-users-ac3e0b8463a5` (you can search web for many other examples), and also for more detailed Matlab to Python function/notation mapping is given

   `https://docs.scipy.org/doc/numpy-1.15.0/user/numpy-for-matlab-users.html`

   Use your favorite IDE for python – I personally use Pycharm at (`www.jetbrains.com/pycharm`).

(a) **Reproducing previous HW with Python**: Convert the MATLAB problem in HW2 to an equivalent code in python. You may want to import libraries such as `numpy`, `matplotlib` and `signal` from `scipy`. For example, if you want to do a convolution between two sequences, you may use `signal.convolve`; if you want to plot figures using matlab-like commands (`subplot, xlabel, ylabel,...`), you may want to use `matplotlib.pyplot`. For example, your python code may start with

```
from scipy import signal
import numpy as np
import matplotlib.pyplot as plt
```

(b) **Gibb's phenomenon:** Consider the following impulse response of truncated low-pass filter

$$h_M[n] = \begin{cases} \frac{\sin(\frac{\pi n}{5})}{\pi n} & -M \leq n \leq M \\ 0 & \text{otherwise} \end{cases}$$

Consider a Gaussian-like input signal given by

$$x[n] = \begin{cases} \exp\left(-\frac{n^2}{2}\right), & -N \leq n \leq N \\ 0 & \text{otherwise} \end{cases}$$

Let $N = 100$. For the variable in frequency domain, consider ω vector $-2\pi \leq \omega \leq 2\pi$ with step size $\Delta\omega = 0.001\pi$. Also let us denote the output signal by $y_M[n]$.

i. Assume $M = 20$. Let DTFTs of $x[n]$, $h_{20}[n]$ and $y_{20}[n]$ be $X(e^{j\omega})$, $H_{20}(e^{j\omega})$ and $Y_{20}(e^{j\omega})$. Plot the magnitude of DTFT $X(e^{j\omega})$, $H_{20}(e^{j\omega})$ and $Y_{20}(e^{j\omega})$ in one page using subplot.

ii. Repeat the same with $M = 70$: plot the magnitudes of $X(e^{j\omega})$, $H_{70}(e^{j\omega})$ and $Y_{70}(e^{j\omega})$. Do you observe the Gibb's phenomenon?

```
from scipy import signal
import numpy as np
import matplotlib.pyplot as plt

N=20
M=5
n=np.arange(N)
u=(n >= 0).astype(int)
ut=(n > M).astype(int)
x=(1/2)**n*u
h=1/(M+1)*(u-ut)
print(h)

fig=plt.figure(1)

plt.subplot(211)
plt.stem(n,x)
plt.xlabel('n')
plt.ylabel('x[n]')
plt.title('(a)')
```

```
plt.subplot(212)
plt.stem(n,h)
plt.xlabel('n')
plt.ylabel('h[n]')

# discrete convolution
y=signal.convolve(x,h)
nn=np.arange(len(y))
plt.figure(2)
plt.stem(nn,y)
plt.xlabel('nn')
plt.ylabel('y[n]')
plt.title('(b)')

# plt.show()

# (c) computing DTFT
pi=np.pi
W=np.arange(-3*pi,3*pi,0.01*pi)

# create column vectors
nv=n.reshape(n.size,1)
nnv=nn[:,None]
xv=x[:,None]
Wv=W[:,None]
hv=h[:,None]
yv=y[:,None]

X=xv.T@np.exp(-1j*nv@Wv.T)
H=hv.T@np.exp(-1j*nv@Wv.T)
Y=yv.T@np.exp(-1j*nnv@Wv.T)

plt.figure(3)
plt.title('magnitude X(jw)')
plt.subplot(211)
plt.plot(W,abs(X[0,:]))

plt.subplot(212)
plt.title('angle X(jw)')
plt.plot(W,np.angle(X[0,:]))

plt.figure(4)
plt.title('magnitude H(jw)')
plt.subplot(211)
plt.plot(W,abs(H[0,:]))

plt.subplot(212)
plt.title('angle H(jw)')
```

```
plt.plot(W,np.angle(H[0,:]))

plt.figure(5)
plt.title('magnitude Y(jw)')
plt.subplot(211)
plt.plot(W,abs(Y[0,:]))

plt.subplot(212)
plt.title('angle Y(jw)')
plt.plot(W,np.angle(Y[0,:]))
plt.show()

#Gibbs phenomenon (b)-(i) -- for (ii), change M=20 to 70
pi=np.pi
N = 100
M = 20

n1 = np.arange(-M,M)
n2 = np.arange(-N,N)
W = np.arange(-2 * pi,2 * pi,0.001 * pi)

h20 = np.sinc(n1/5)/5   # sin(pi * n1 / 5). / (pi * n1 + eps);
x20 = np.exp(-n2**2)/2  # exp(-(n2. ^ 2) / 2);
y20 = signal.convolve(x20,h20)  # conv(x20, h20);

nn = np.arange(len(y20))    #0:length(y20) - 1;

x20v=x20[:,None]
h20v=h20[:,None]
y20v=y20[:,None]
n2v=n2[:,None]
n1v=n1[:,None]
nnv=nn[:,None]
Wv=W[:,None]

X20=x20v.T @ np.exp(-1j*n2v@Wv.T)
H20=h20v.T @ np.exp(-1j*n1v@Wv.T)
Y20=y20v.T @ np.exp(-1j*nnv@Wv.T)

fig=plt.figure(6)
plt.title('Magnitude when M=20')
plt.subplot(311)
plt.plot(W, abs(X20[0,:]))
plt.xlabel('W')
plt.ylabel('|X(jw)|')

plt.subplot(312)
plt.plot(W, abs(H20[0,:]))
```

```
plt.xlabel('W')
plt.ylabel('|H(jw)|')

plt.subplot(313)
plt.plot(W, abs(Y20[0,:]))
plt.xlabel('W')
plt.ylabel('|Y(jw)|')

plt.show()
```