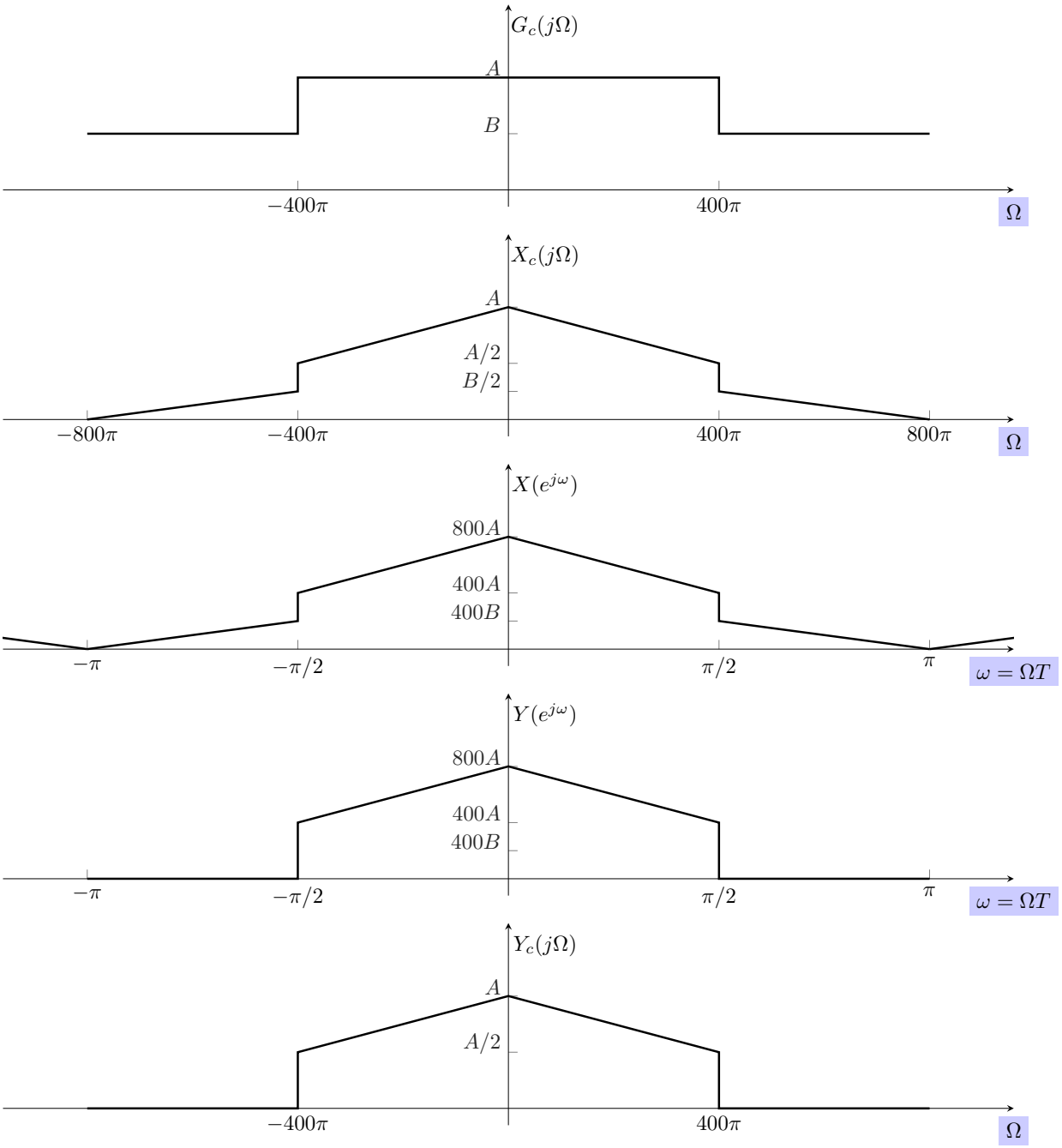
Problem 1

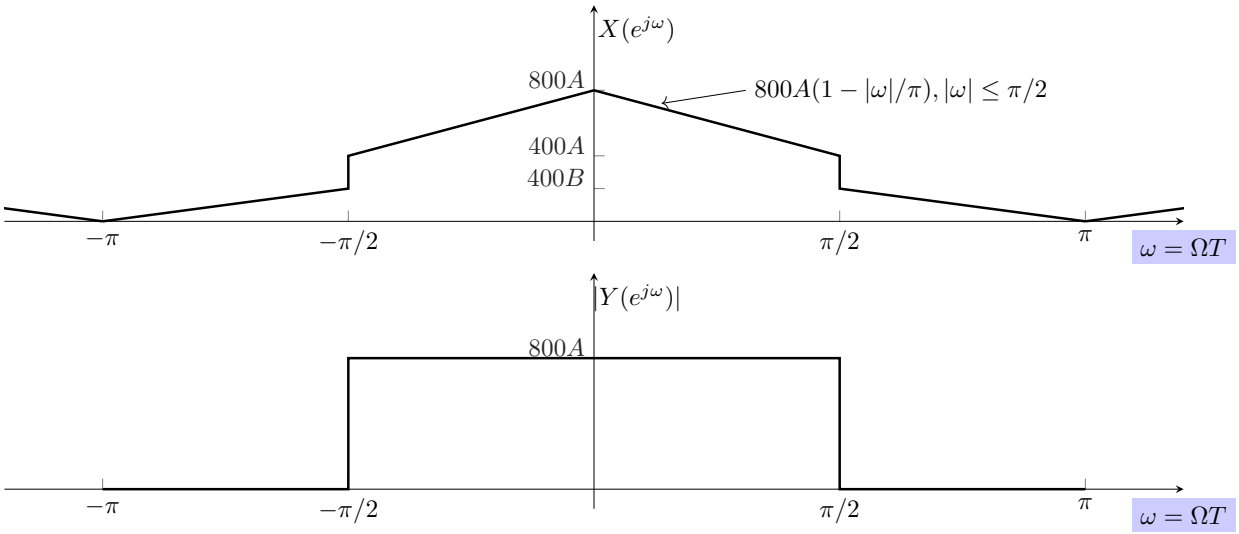
(a)



(b)

We want to obtain $y_c(t) = f_c(t - 0.1)$. In the frequency domain,

$$Y_c(j\Omega) = F_c(j\Omega)e^{-j0.1\Omega} \qquad \text{(Delay property of the Fourier transform)}$$
$$Y(e^{j\omega}) = F_c(j\omega/T)e^{-j0.1\omega/T} \qquad \text{(Since } \Omega = \omega/T \text{ and there is no aliasing)}$$
$$Y(e^{j\omega}) = F_c(j\omega/T)e^{-j80\omega} \qquad (T = 1/800)$$

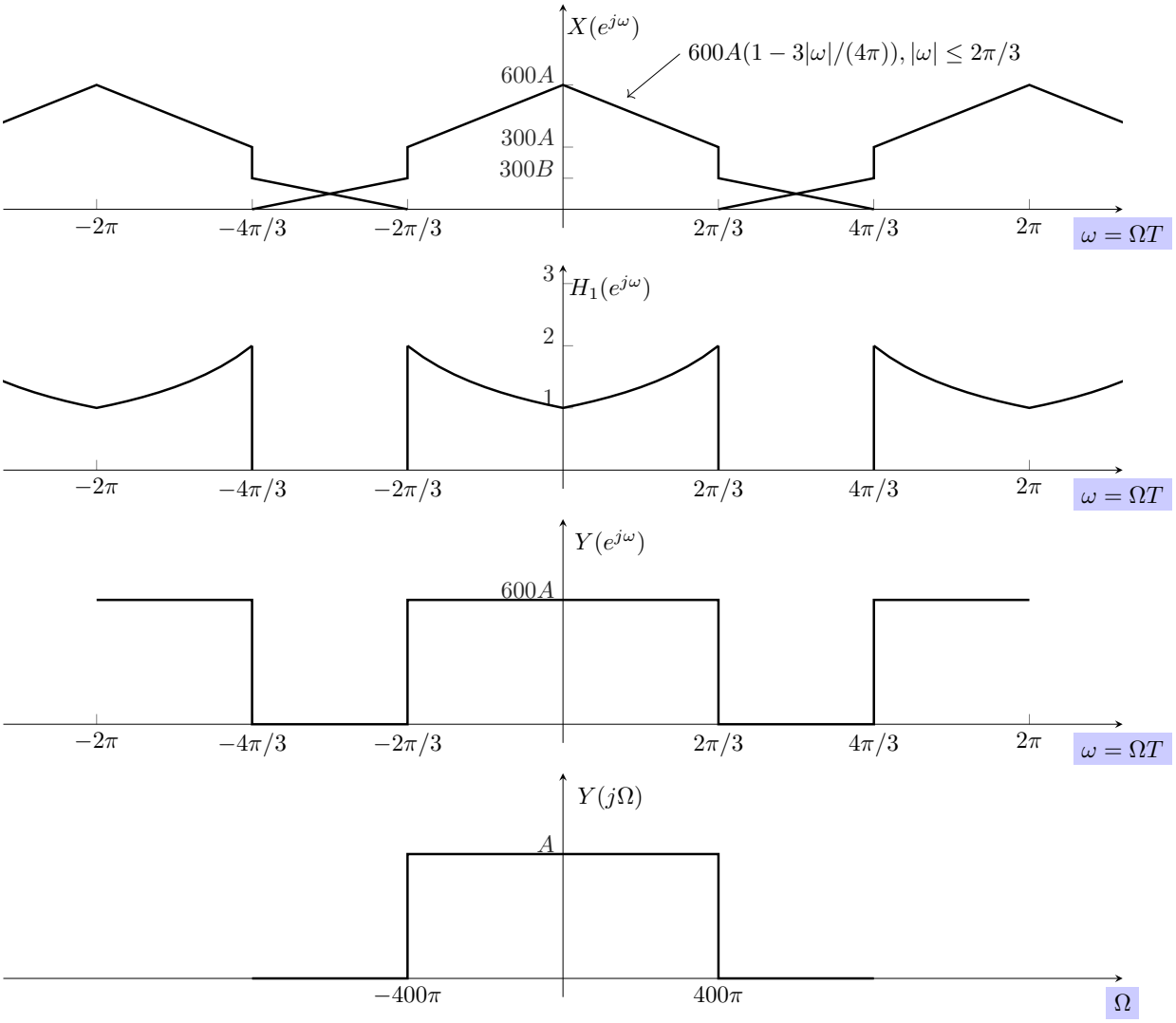


Now we need to find $H_1(e^{j\omega})$ such that $Y(e^{j\omega}) = H_1(e^{j\omega})X(e^{j\omega})$. By inspecting the plots of the figure above, it is clear that the filter $H_1(e^{j\omega})$ needs to flatten the part of the spectrum between $|\omega| < \pi/2$, and it needs to be zero for $|\omega| \le \pi/2 < \pi$. Therefore, we can write:

$$H_1(e^{j\omega}) = \begin{cases} \dfrac{1}{1-|\omega|/\pi}e^{-j80\omega}, & |\omega| \le \pi/2 \\ 0, & \pi/2 < |\omega| \le \pi/2 \end{cases} \tag{1}$$

Note that the factor $e^{-j80\omega}$ appears from the delay requirement and not from inspecting the magnitude plots.

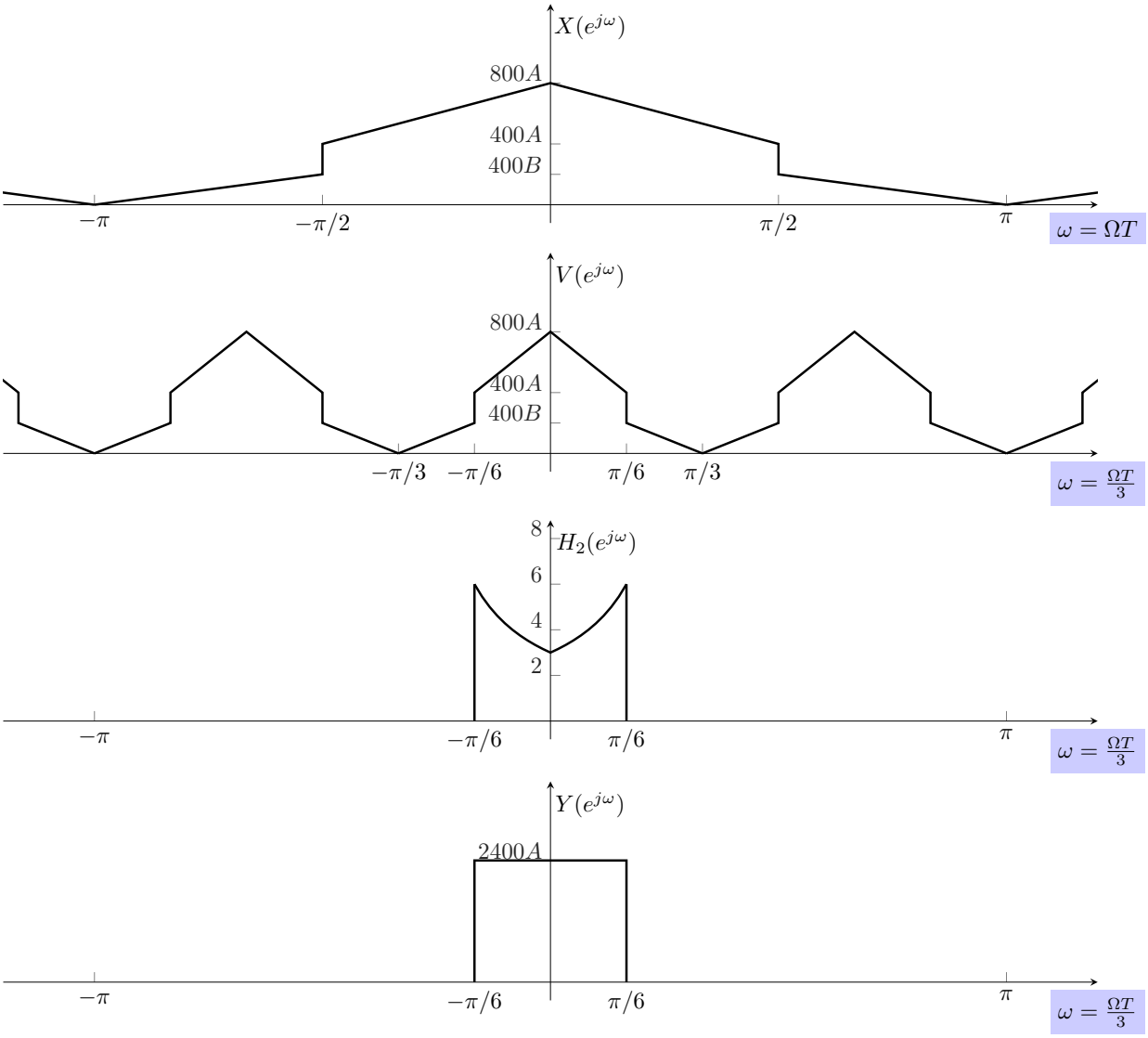
(c)



The filter $H_1(e^{j\omega})$ only needs to flatten the frequency response of $X(e^{j\omega})$ over the interval $\omega < 2\pi/3$. Therefore,

$$H_1(e^{j\omega}) = \begin{cases} \dfrac{1}{1-\dfrac{3|\omega|}{4\pi}}, & |\omega| \le \frac{2\pi}{3} \\ 0, & \frac{2\pi}{3} < |\omega| \le \pi \end{cases} \tag{2}$$

(d)



The filter $H_2(e^{j\omega})$ only needs to flatten the frequency response of $V(e^{j\omega})$ over the interval $\omega < \pi/6$. Therefore,

$$H_2(e^{j\omega}) = \begin{cases} \frac{3}{3|\omega|}, & |\omega| \leq \frac{\pi}{6} \\ 1 - \frac{}{\pi} \\ 0, & \frac{\pi}{6} < |\omega| \leq \pi \end{cases} \tag{3}$$

Problem 2

(a)

To maintain a constant time scale we need

$$T_2 = \frac{2}{3} T_1 \tag{4}$$

(b)

To avoid aliasing in the sampling of the input

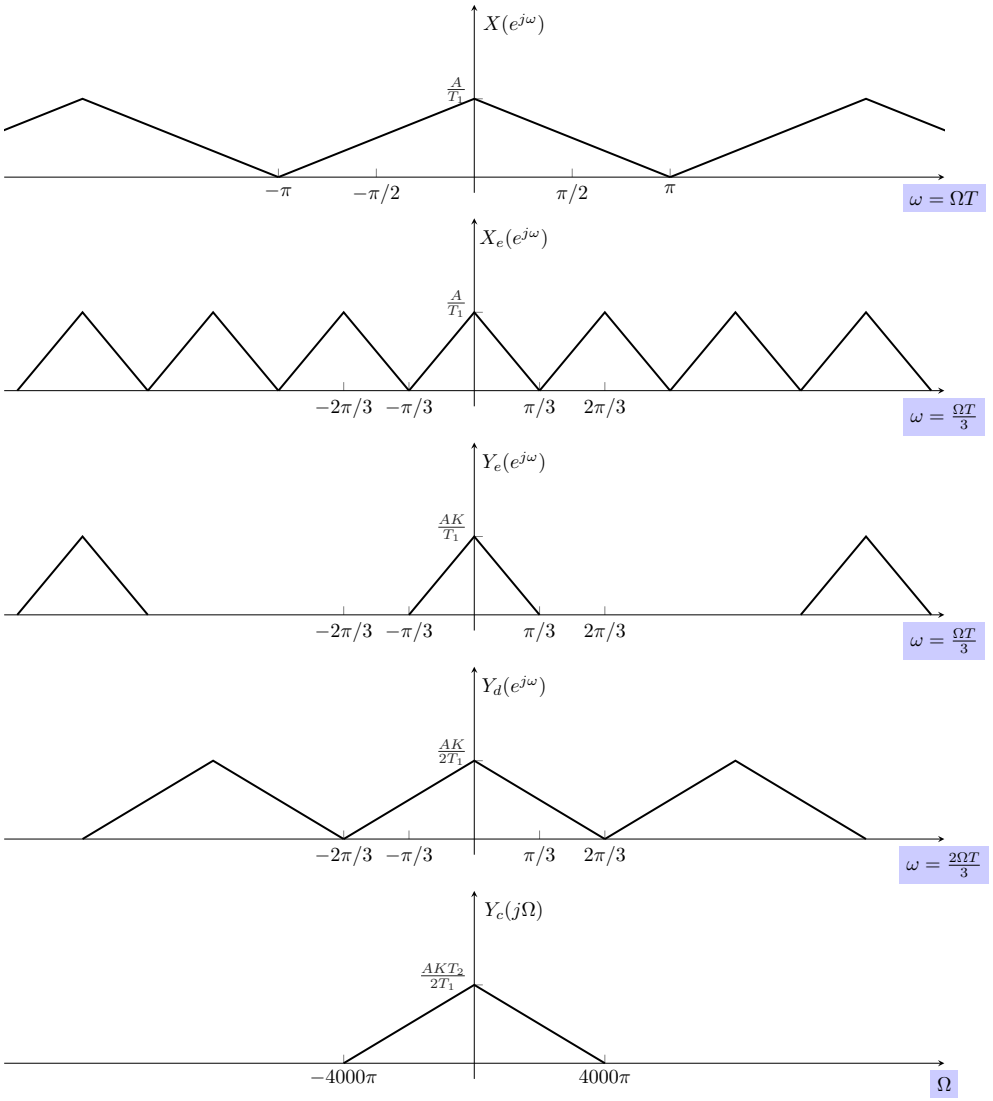
$$\Omega_s = \frac{2\pi}{T_1} > 2\Omega_N = 8000\pi \tag{5}$$

$$T_1 < 1/4000 \tag{6}$$

To perfectly reconstruct the input, we should preserve the same time scale between input and output. Hence, using the result from part (a):

$$T_2 = \frac{2}{3}T_1 \implies T_2 > 1/6000 \tag{7}$$

(c)



For perfect reconstruction we need:

$$\frac{AKT_2}{2T} = A \implies K = \frac{2T_1}{T_2} = 3 \tag{8}$$

Therefore,

$$H(e^{j\omega}) = \begin{cases} 3, & |\omega| \leq \pi/3 \\ 0, & \pi/3 < |\omega| \leq \pi \end{cases} \tag{9}$$

(d)

$$y_c(t) = 2x_c(t - 4T_1) \Longleftrightarrow Y_c(j\Omega) = 2e^{-j4T_1\Omega}X_c(j\Omega) \tag{10}$$

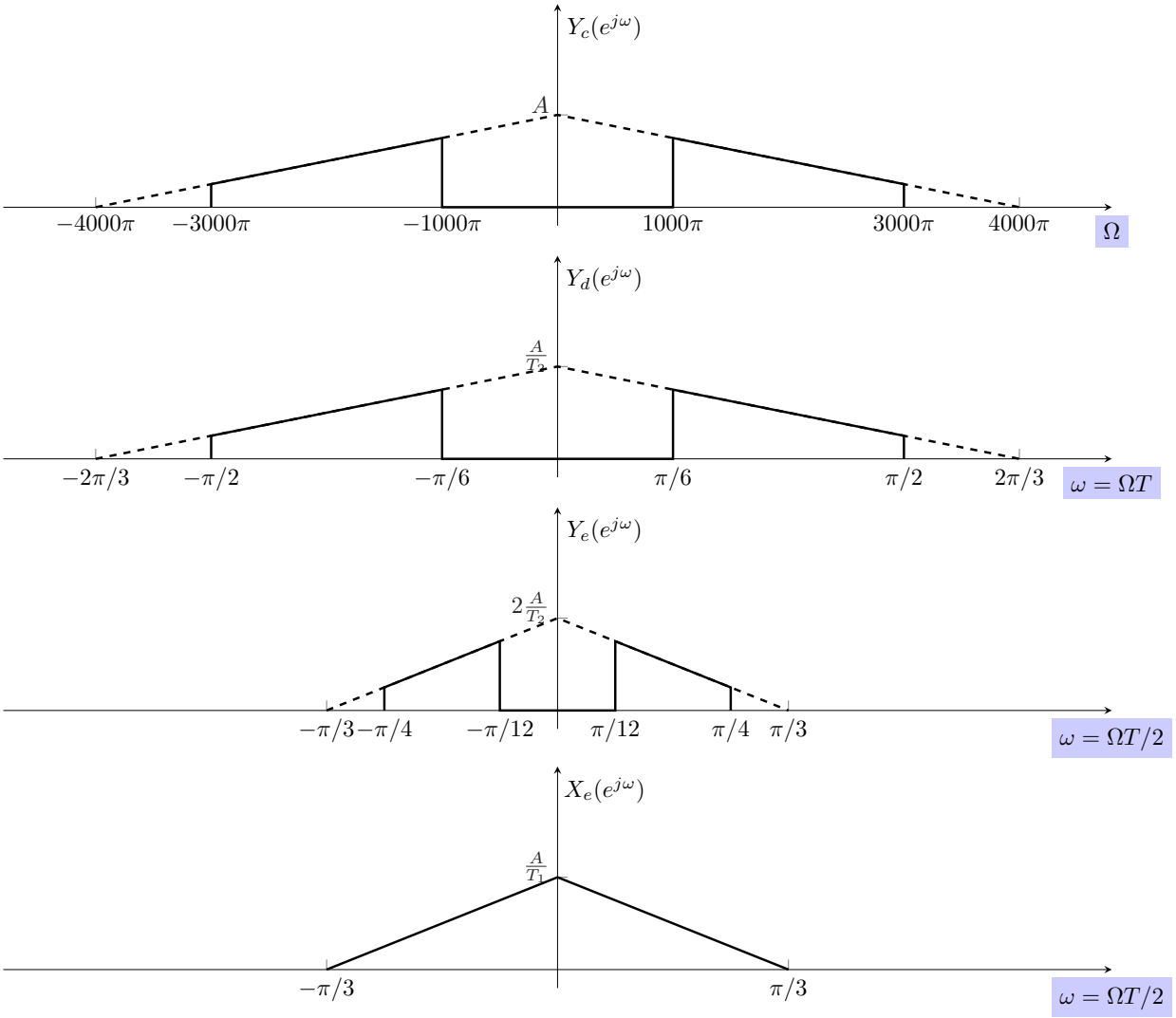
Similarly to previous items, the shape of the spectrum will not be changed, so we can focus on the amplitude at $\Omega = 0$ in order to determine the amplitude of $H(e^{j\omega})$.

$$\begin{aligned} Y_c(j\Omega = 0) &= 2Ae^{-j4T_1\Omega} \\ Y_d(e^{j\omega} = 1) &= \frac{2A}{T_2}e^{-j4T_1\omega/T_2} = \frac{2A}{T_2}e^{-j6\omega} && \text{(since we can interpret } y_d[n] = y_c(nT_2)\text{)} \\ Y_e(e^{j\omega} = 1) &= \frac{4A}{T_2}e^{-j16\omega} && \text{(upsampling by 2 with filter scaling 2)} \end{aligned}$$

By comparing $Y_e(e^{j\omega} = 1)$ and $X_e(j\Omega)$, we can write

$$\begin{aligned} H(e^{j\omega}) &= \begin{cases} 4\frac{T_1}{T_2}e^{-j12\omega}, & |\omega| \leq \pi/3 \\ 0, & \pi/3 < |\omega| \leq \pi \end{cases} \\ &= \begin{cases} 6e^{-j12\omega}, & |\omega| \leq \pi/3 \\ 0, & \pi/3 < |\omega| \leq \pi \end{cases} \end{aligned} \tag{11}$$

(e)



Therefore, to obtain $Y_e(e^{j\omega})$ from $X(e^{j\omega})$, the filter $H(e^{j\omega})$ must satisfy

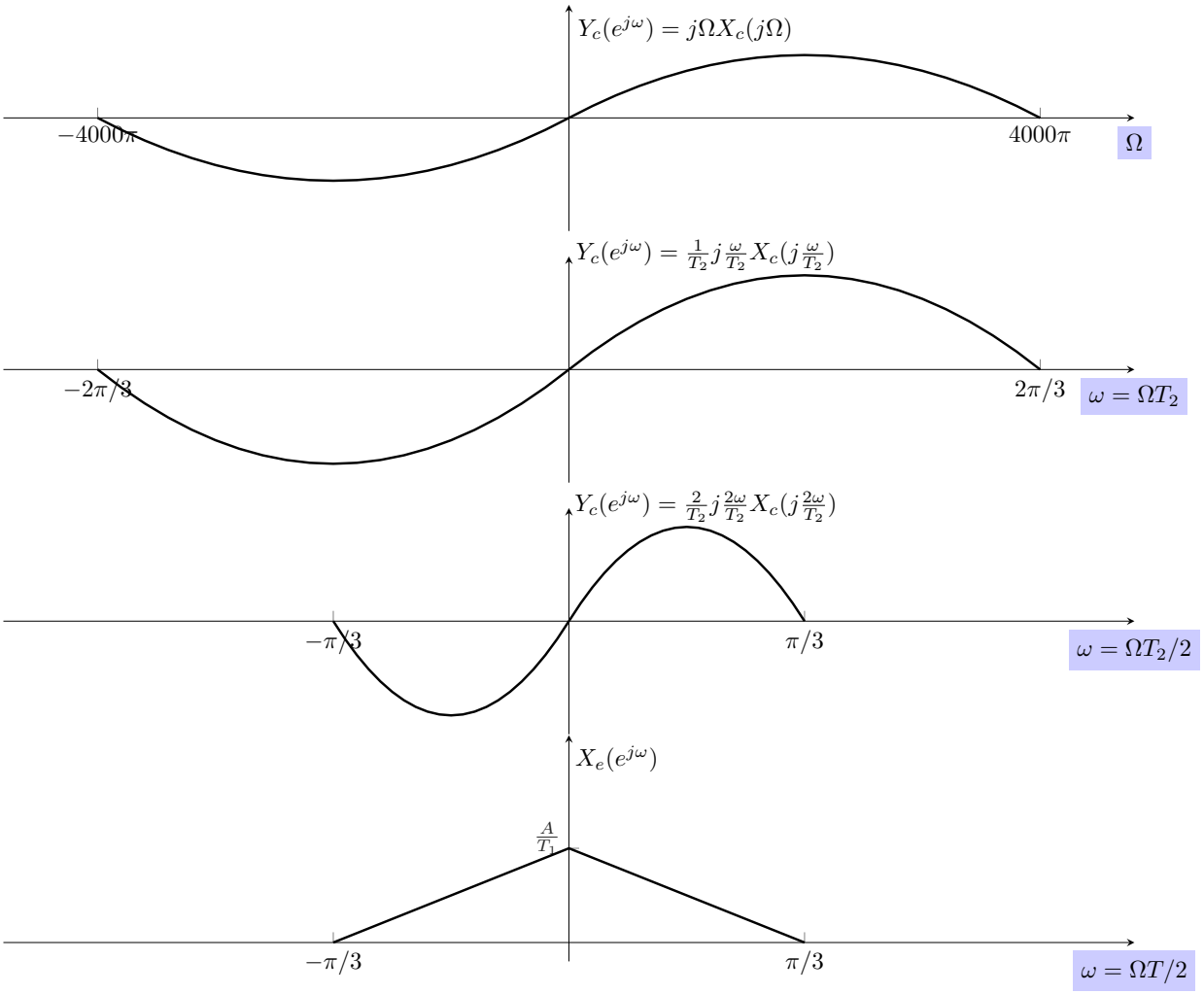
$$H(e^{j\omega}) = \begin{cases} 0, & |\omega| \le \pi/12 \\ \frac{2T_1}{T_2}, & \pi/12 < |\omega| \le \pi/4 \\ 0, & \pi/4 < |\omega| \le \pi \end{cases}$$

$$= \begin{cases} 0, & |\omega| \le \pi/12 \\ 3, & \pi/12 < |\omega| \le \pi/4 \\ 0, & \pi/4 < |\omega| \le \pi \end{cases}, \tag{12}$$

which is a band-pass filter.

(f)

$$y_c(t) = \frac{d}{dt} x_c(t) \Longleftrightarrow Y_c(j\Omega) = j\Omega X_c(j\Omega) \tag{13}$$



Since for $|\omega| \leq \pi/3$, $X_c(j2\omega/T_2) = X_c(j3\omega/T_1) = T_1 X_e(e^{j\omega})$. Thus,

$$\begin{aligned} Y_e(e^{j\omega}) &= 4j\frac{\omega}{T_2} X_c(j2\frac{\omega}{T_2}) \\ &= 4j\frac{\omega}{T_2^2} T_1 X_e(e^{j\omega}) \\ &= 4j\frac{T_1}{T_2^2}\omega X_e(e^{j\omega}), |\omega| \leq \pi/3 \end{aligned} \tag{14}$$

Therefore,

$$H(e^{j\omega}) = \begin{cases} 4j\frac{T_1}{T_2^2}\omega, & |\omega| \leq \pi/3 \\ 0, & \pi/3 < |\omega| \leq \pi \end{cases} \tag{15}$$

Problem 3

(a)

For the original system

$$\begin{aligned}\mathcal{F}\{\phi_1[m]\} &= |X_1(e^{j\omega})|^2 \\ &= |X_e(e^{j\omega L})H_1(e^{j\omega})|^2 \\ &= \begin{cases} L^2|X_e(e^{j\omega L})|^2, & |\omega| \le \pi/L \\ 0, & \pi/L < |\omega| \le \pi \end{cases}\end{aligned} \tag{16}$$

For the system proposed by the student

$$\begin{aligned}\mathcal{F}\{\phi_3[m]\} &= \mathcal{F}\{\phi_{2e}[m]\}H_2(e^{j\omega}) \\ &= |X(e^{j\omega L})|^2 H_2(e^{j\omega})\end{aligned} \tag{17}$$

The two systems will be equivalent if

$$H_2(e^{j\omega}) = \begin{cases} L^2, & |\omega| \le \pi/L \\ 0, & \pi/L < |\omega| \le \pi \end{cases} \tag{18}$$

(b)

$$\mathcal{F}\{\phi_5[m]\} = |X(e^{j\omega L})|^2 H_3(e^{j\omega})$$

This is exactly the same condition of part (a). Therefore,

$$H_3(e^{j\omega}) = \begin{cases} L^2, & |\omega| \le \pi/L \\ 0, & \pi/L < |\omega| \le \pi \end{cases} \tag{19}$$

Problem 4

Part 1: Implementing a quantizer

(a)

The quantizer function implemented is shown below. Many other implementations are possible and will be accepted.

```
1  function [xq, e] = quantizer(x, B, range_lims)
2  %% B-bit quantizer with range determined by range_lims
3
4  DeltaX = range_lims(2) - range_lims(1); % dynamic range
5  Delta = DeltaX/(2^B); % step size
6
7  % Quantization levels start at range_lims(1) and go up to range_lims(2) in
```

```
% Delta increments. This is the codebook for the quantiz function
levels = range_lims(1):Delta:range_lims(2);
[~, xq] = quantiz(x, levels(1:end-1)+Delta/2, levels);
% The codebook is shifted by Delta/2 in order to obtain the partitions

xq = xq.'; % transpose to preserve same dimensionality as x
e = x - xq; % quantization error

```

(b)

The code for part (b) is included on the next page. The empirical autocorrelation function is shown below. As we can see, the assumption that the noise is white is more accurate for finer quantization.

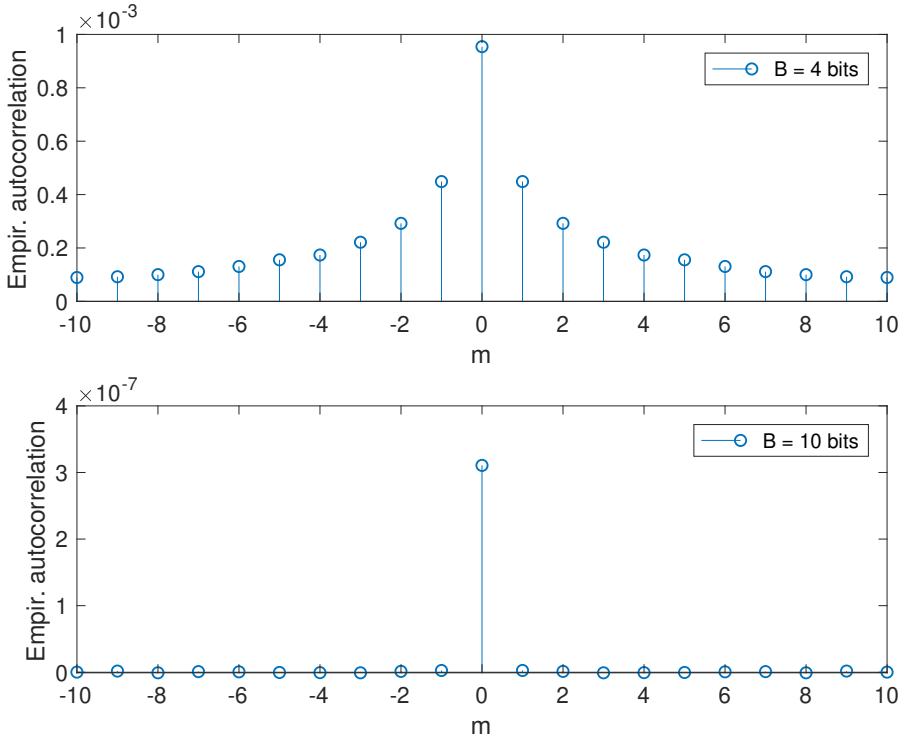


Figure 1: Empirical autocorrelation functions for $B = 4$ and $B = 10$ bits.

The average power is simply the autocorrelation function evaluated at 0. Therefore, we have:

$$\hat{\sigma}_e^2 = \begin{cases} 0.001, & B = 4 \text{ bits} \\ 3.1841 \times 10^{-7}, & B = 10 \text{ bits} \end{cases}$$

Recall that as discussed in class, by assuming quantization noise as an uniformly distributed noise:

$\sigma_e^2 = \Delta^2/12$. For each case,

$$\sigma_e^2 = \begin{cases} 0.0013, & B = 4 \text{ bits} \\ 3.1789 \times 10^{-7}, & B = 10 \text{ bits} \end{cases}$$

Note that these estimates are very accurate even when quantization is coarse.

Part 2: Noise shaping without oversampling

(c)

The code for part (c) is included on the next page.

Part 3: Noise shaping with oversampling

(d)

The code for part (d) is included on the next page.

(e)

The code for part (e) is included on the next page.

The noise power after noise shaping and filtering is $\tilde{\sigma}_e^2 = 1.113 \times 10^{-4}$. We can define the signal-to-noise ratio (SNR) improvement

$$\text{SNR}_{\text{improvement}} = 10 \log_{10} \left(\frac{\tilde{\sigma}_e^2}{\sigma_e^2} \right) = 9.6858 \text{ dB (or 9.3021 in linear units).} \tag{20}$$

Since for every extra bit of resolution we have a 6.02 dB improvement of SNR. This improvement in SNR would be equivalent to increase the quantizer resolution by 1.6 bits. More significant improvements could be achieved either by using higher oversampling factor M or higher-order noise shaping (different filter $H(z)$).

Matlab code for parts (b) through (e)

```
%% Template for Homework 3: Problem 3 on quantization and quantization noise shaping
clear, clc, close all % clear variables

% Load speech signal
[x, Fs] = audioread('speech_dft.wav');          % Fs is sampling frequency
T = 1/Fs;                                       % sampling period

%% Important:
% If your simulations take too long because you're using a slow computer,
% you can uncomment the following line in order to use just part of the
% speech signal. Specifically, you may use just the first 43000 samples,
% which correspond to the phrase "The discrete Fourier transform".
% In my computer, the simulations with the whole signal took < 1 s.
% x = x(1:43e3); % play just part of the speech

%% Your code goes here
```

```
%sound(x, Fs) % Play speech

% Define parameters
range_lims = [-1 1];           % range limits
B = 4; % quantizer resolution

%% Part (b)
maxLag = 10;
[~, e4bits] = quantizer(x, 4, range_lims);
[~, e10bits] = quantizer(x, 10, range_lims);

c4bits = xcorr(e4bits, e4bits, maxLag, 'unbiased');
c10bits = xcorr(e10bits, e10bits, maxLag, 'unbiased');

e4bits_power = c4bits(maxLag+1); % autocorrelation at m = 0;
e10bits_power = c10bits(maxLag+1); % autocorrelation at m = 0;

% Plot autocorrelation function
figure, subplot(211), box on
stem(-maxLag:maxLag, c4bits)
legend('B = 4 bits')
xlabel('m', 'FontSize', 12)
ylabel('Empir. autocorrelation', 'FontSize', 12)
set(gca, 'FontSize', 12)
subplot(212), box on
stem(-maxLag:maxLag, c10bits)
legend('B = 10 bits')
xlabel('m', 'FontSize', 12)
ylabel('Empir. autocorrelation', 'FontSize', 12)
set(gca, 'FontSize', 12)
saveas(gca, '../figs/hw03_q4_empir_xcorr', 'epsc')

%% Part (c)
B = 4; % quantizer resolution
b = [1 -1]; % shaping filter coefficients

[xq, e] = quantizer(x, B, range_lims);

eshaped = filter(b, 1, e);

xqshaped = x + eshaped;
%sound(xqshaped, Fs)

%% Part (d)
B = 4; % quantizer resolution
```

```
62 b = [1 -1]; % shaping filter coefficients
63 M = 3; % upsampling factor
64 [xq, e] = quantizer(x, B, range_lims);
65
66 eup = upsample(e, M);
67 eshaped = filter(b, 1, eup);
68 ef = M*decimate(eshaped, M); % Note multiplication by M in order to account
69 % for the fact that in Fig. 6a the signal is also decimated
70
71 xqshaped = x + ef;
72 sound(xqshaped, Fs)
73
74 ef_power = mean(abs(ef).^2); % can use var only if ef is zero mean
75
76 SNRimprovement = 10*log10(e4bits_power/ef_power)
77
78 Bimprovement = SNRimprovement/6.02
```