S&S Computer HW3 Solution (EECS202002 Spring 2021)

Part 1:

(a)

$$F\{\cos(2\pi f_0 t)\} = \delta(f - f_0) + \delta(f + f_0)$$

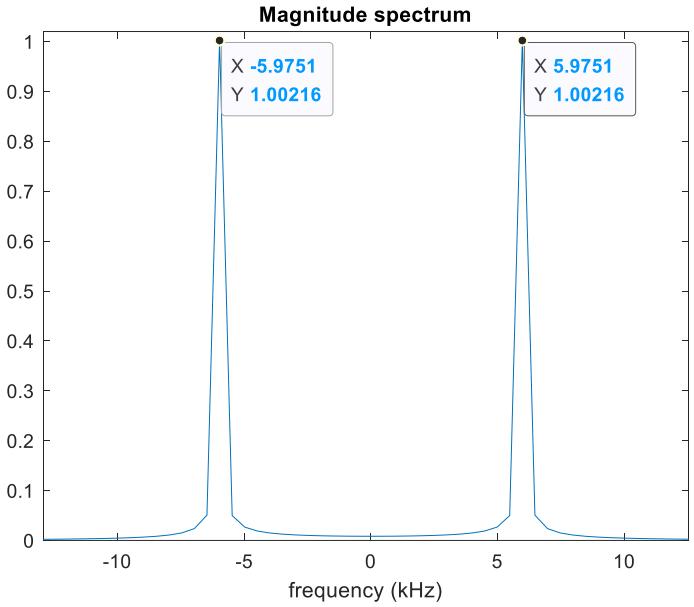
Assume $f_0 = 6\ kHz$,

$$F\{\cos(2\pi * 6 * t)\} = \delta(f - 6\ kHz) + \delta(f + 6\ kHz)$$

After applying Fourier transform to cosine signal, we expect to get 2 delta functions located at $\pm 6\ kHz$. However, from the FT result in the following figure (a-1), we can observe that the two peaks located nearby $\pm 6\ kHz$ are not 2 delta functions exactly. This is due to the finite length of the signal which is with N sampling points only (i.e., the cosine signal is windowed and thus its spectrum suffers window effect). In addition, from the code implementing X(iFreq), the spectrum is sampled and the sampling interval in frequency domain is $\frac{2\pi}{N}$ in normalized angular frequency or Fs/N in absolute frequency Hz. Therefore, the two peaks may not be located at exact $\pm 6\ kHz$ unless $\pm 6\ kHz$ happens to be a multiple of the sampling interval($i.e., \frac{kF_s}{N}, k \in Z$).

見 Slide 61, Topic5_TimeAndFreqAnalysisOfSignalsAndSystems_HandWriting0602_2021.pdf
如果我們在 time domain 讓一個訊號和 window function 相乘，那麼在 CTFT frequency domain 就是訊號的 FT 和 window function 的 FT (i.e., sinc function) 做 convolution，而不會是原本訊號的 spectrum，這就是所謂的 window effect。以此題及 slide 61 上的例子來說，$\cos \omega_0 t$的 FT 原本是在$\pm\omega_0$各有一個 impulse function，但因為 window effect 的關係，frequency domain 變成在$\pm\omega_0$各有一個 sinc function。其中 frequency domain 的 sinc function 寬度與 time domain 的 window function 寬度有關。time domain 的 window function 寬度越窄，frequency domain 的 sinc function 寬度越寬；time domain 的 window function 寬度越寬，frequency domain 的 sinc function 寬度越窄。若 time domain 的 window function 寬度為無限寬，則 frequency domain 的 sinc function 寬度為無限窄，也就會是完美的 impulse function。下圖若要能看到 sinc function (實際上是 digital sinc function)的樣子，作圖時或計算 X(iFreq)時的 frequency domain

sampling interval 就要夠小，這樣才能做出更接近 CTFT (實際上是 DTFT) 的 spectrum。因所提供的程式碼中，frequency domain sampling interval = Fs/N，N 為 time domain 信號取樣的點數，可以透過對 time domain signal 做 zero padding 的手段讓等效的 N 變大，使得讓 frequency domain sampling interval 變小，而得到更平滑更接近連續的 spectrum (see Problem 1(d))。



(b)

According to the code:

```
% implementatoin of X(Fk) = summation x(nT)*exp(-j*2*pi*Fk*(nT))*T
for iFreq = 1:length(f_axis),
    iFreq
    for iTime = 1:length(t_axis),
        X(iFreq) = X(iFreq) + x(iTime)*exp(-sqrt(-1)*2*pi*f_axis(iFreq)*t_axis(iTime))*T;
    end
end
```

We have:

$$X(iFreq) = \sum_{iTime} x(iTime) \times e^{-j2\pi \times iFreq \times iTime} \times T$$

$$X\left(\frac{2\pi k}{N}\right) = T \sum_{n} x(nT) \times e^{-j\frac{2\pi k}{NT} \times nT}$$

$$X[k] = T \sum_{n=0}^{N-1} x[n] \times e^{-\frac{j2\pi kn}{N}} \quad -(1)$$

And the equation of DTFS is.

$$a_k = \frac{1}{N}\sum_{n=0}^{N-1} x[n] \times e^{-\frac{j2\pi kn}{N}} = \frac{1}{N*T}X[k] \quad -(2)$$

After converting the sample code to mathematical formula, we can observe that the sample code is equal to **DTFS result being multiplied by N*T. (By comparing equation (1) of the implemented code and DTFS Equation(2))**

(c)

From (b), the equation implemented in MATLAB:

$$X'[k] = T\sum_{n=0}^{N-1} x[n] \times e^{-\frac{j2\pi kn}{N}}$$

If we perform Fourier analysis using MATLAB fft() function, we find that fft() function actually implements DFT (Discrete Fourier Transform). (Look up the documentation of fft() function in MATLAB and see the lecture of FFT usage on EECLASS).

The equation of DFT:

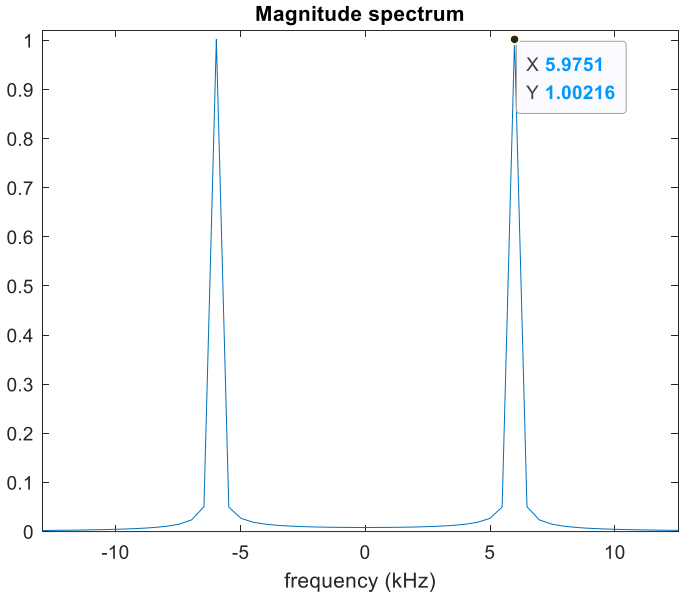
$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{k2\pi n}{N}}$$

By Comparing DFT equation and the implemented CTFT equation, we can see that the equation of our code is equal to T times of the equation of DFT (See the lecture of FFT usage on EECLASS). From code, we have T=1/Fs=1/120≈ 0.083

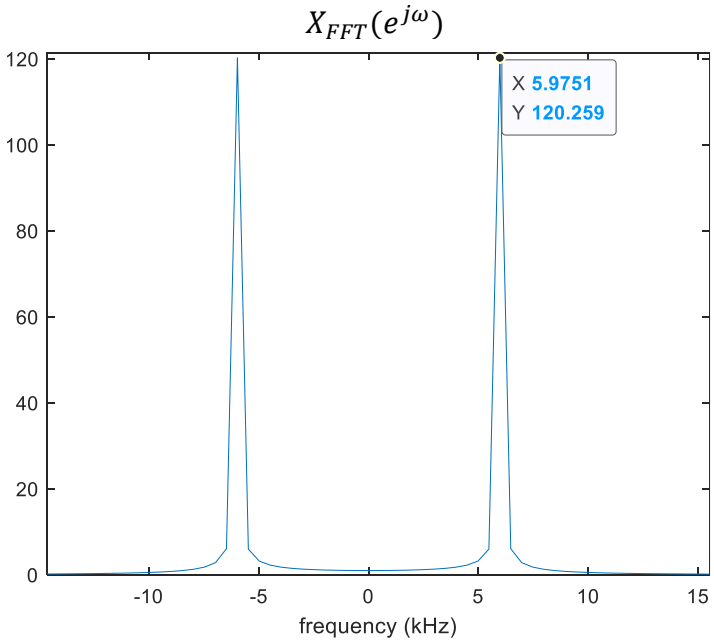
```
%% ---------- Generate sampled cosine/discrete-time sinusoid ----------
F0 = 6; % in kHz
Fs = 120; % sampling rate/sampling frequency, in kHz or ksamples/sec
T = 1/Fs;  % time resolution, i.e., sampling interval in time domain, in ms
total_time = 2; % in ms
```

We validate it by plotting the two spectra - checking whether the implemented CTFT code (equation (1)) $= T * X_{FFT}[k]$) or not.

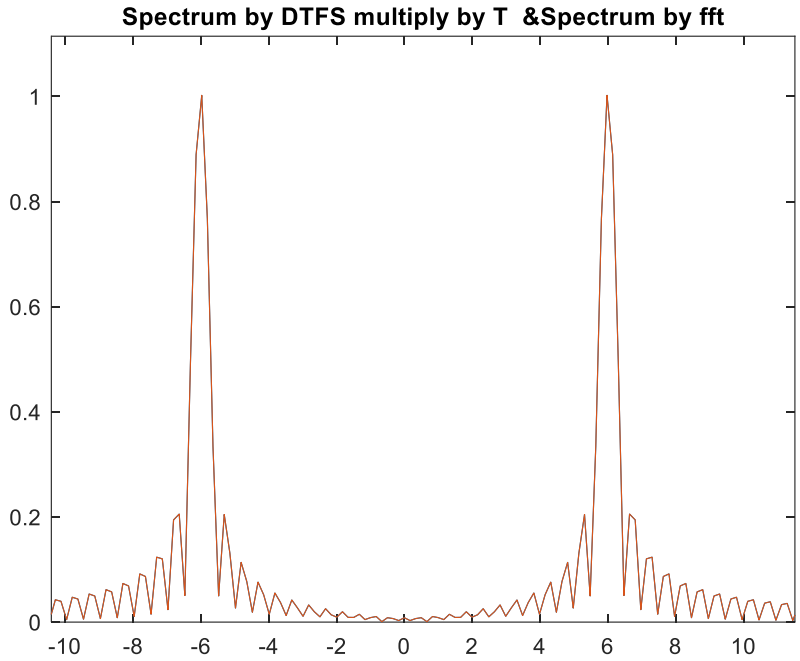
The spectrum generated by the given CTFT code:



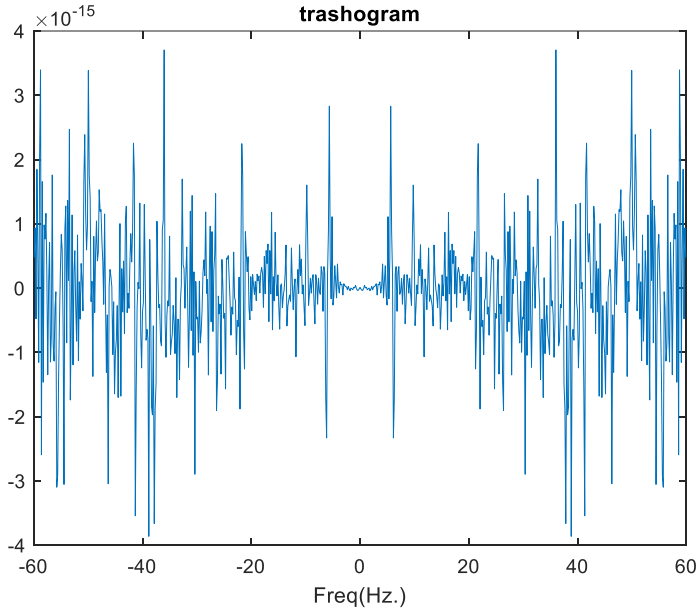
The spectrum generated by MATLAB built-in function fft() (i.e., DFT):



Checking if the given CTFT code $= T * X_{FFT}[k]$



Plot $the\ implemented\ CTFT - T * X_{FFT}[k]$



The difference between the two is extremely small, which confirms the result describing in the above, i.e., the implemented CTFT = T*X_{FFT}[k].

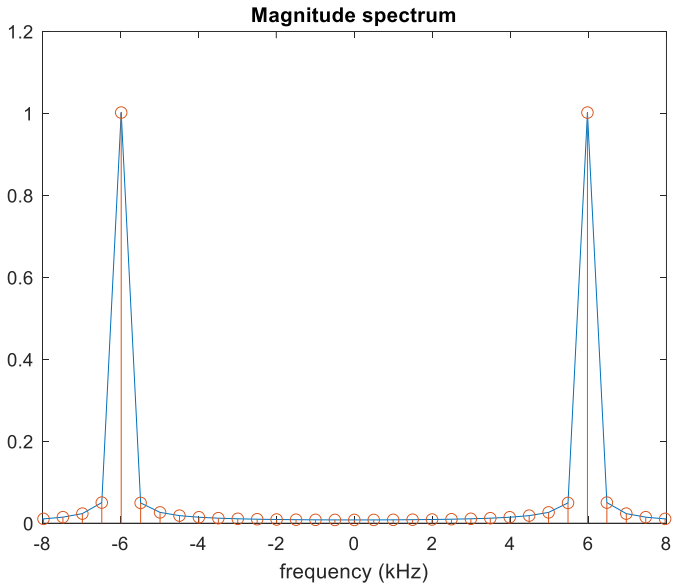
(d)

In problem 1(b) we know the implemented CTFT is equal to DTFS*N*T. For DTFS, we know that $N \uparrow$ (N: the number of sampling points), then $\Delta F \downarrow$ ($\Delta F = \frac{Fs}{N}\ from\ DTFS\ or\ from\ the\ give\ code$). Thus, in order to improve the frequency resolution (i.e., the frequency-domain sampling interval), **we apply zero padding to the original signal**

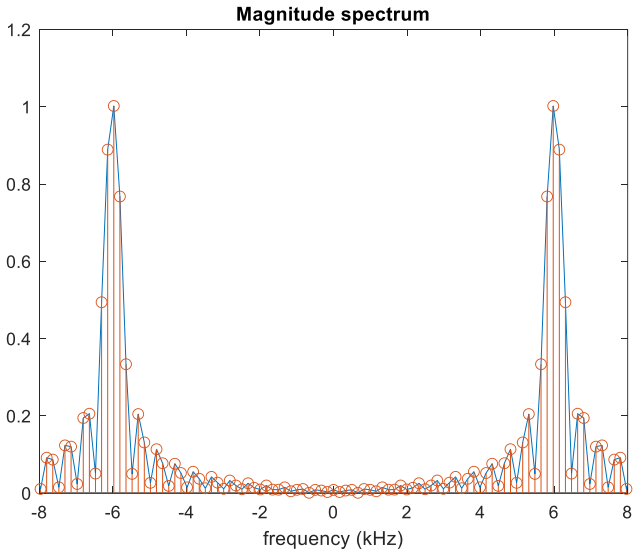
to increase the number of sampling points. In the following code, 2 times length(x) of zeroes are padded after the original signal x.

```
t_axis = (0:T:total_time);  % time axis
x = cos(2*pi*F0*t_axis);  % sampled cosine/discrete time sinusoid ,time domain
x = [x, zeros(1, 2*length(x))];   zero padding
```

The spectrum before zero-padding:



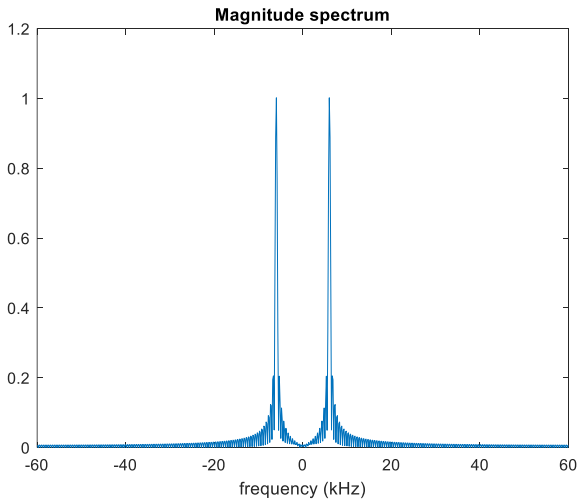
The spectrum after zero-padding (Now we can see two sinc functions (in fact, should be two digital sinc functions), resulting from window effect, in [-Fs/2 Fs/2] or fundamental frequency range in normalized angular frequency):

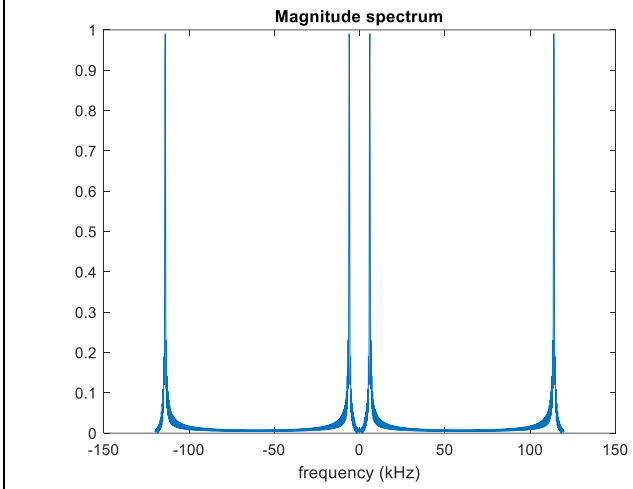


(e)

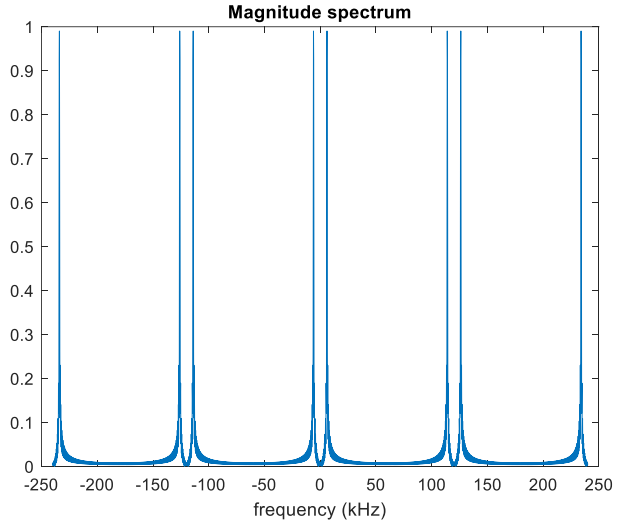
The sampling rate Fs is 120Hz. The result shown below is three different frequency ranges.

[-Fs/2 Fs/2]	[-Fs Fs]





[-2Fs 2Fs]



From the result, we can find that the generated CTFT spectrum by the give CTFT codes becomes periodic if the display frequency range is increased. This is because CT signal cannot be presented in computer simulation, so the original signal is sampled in the code, i.e., we perform Fourier analysis over a DT signal, which spectrum is periodic with its period = sampling frequency = 1 in normalized frequency = 2 π in normalized angular frequency .

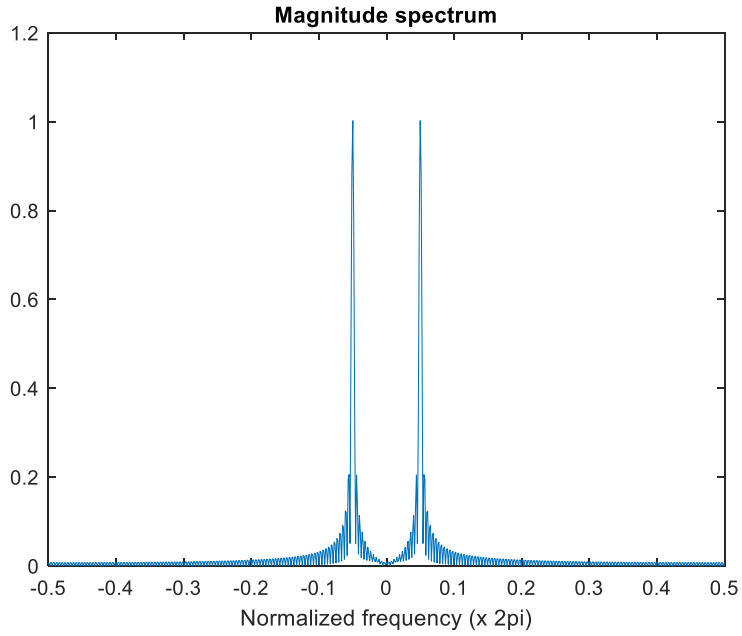
Then we try to show the magnitude spectrum with the normalized frequency axis. We can just divide original frequency axis by sampling frequency.

$$Normalized\ angular\ frequency = \frac{Absolute\ frequency}{Fs} \times 2\pi$$

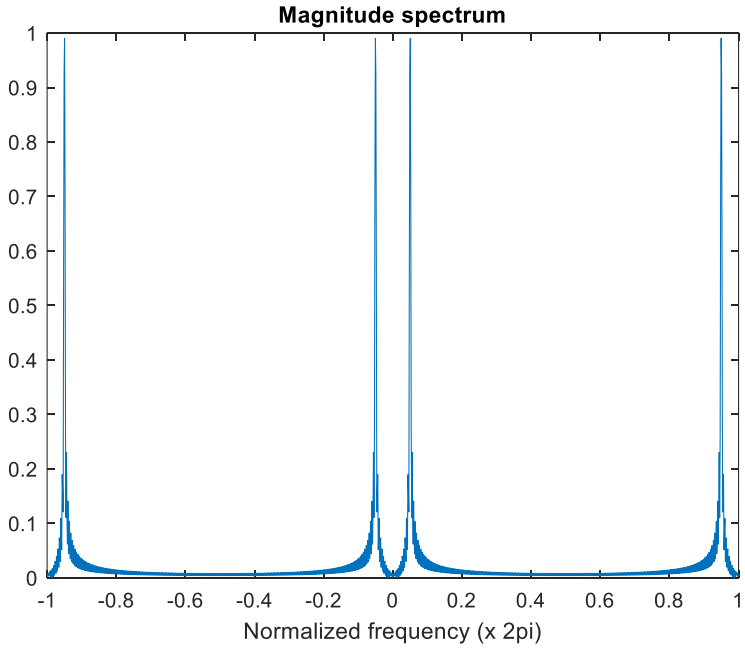
(From slide 7, Topic3_FourierRepresentation_Part4_withNotes.pdf).

Here we let frequency range [-Fs/2 Fs/2] and convert it to normalize angular frequency [-

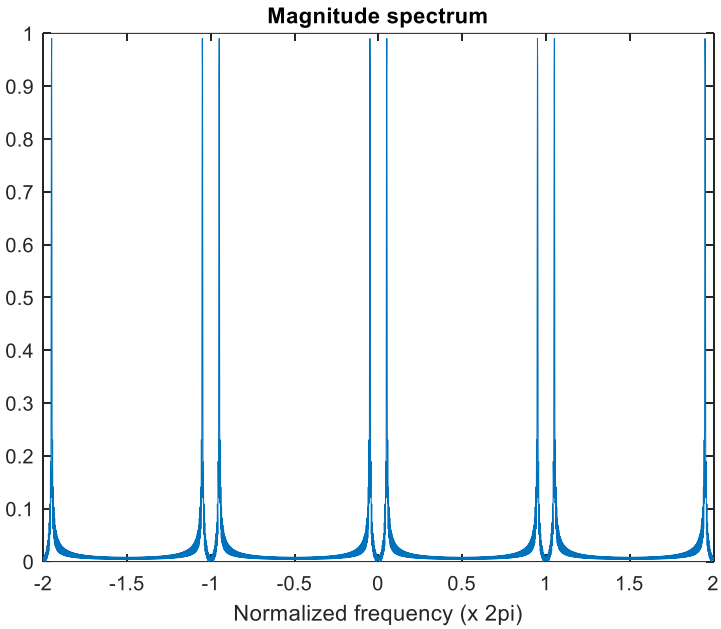
π π] , which is the fundamental frequency range. The result is shown below.



If we change frequency range to [-Fs Fs], then the normalized frequency is changed to [-2π 2π], which is 2 times fundamental frequency range and thus two periods of the frequency domain signal can be found in the following spectrum plot. The result is shown below.

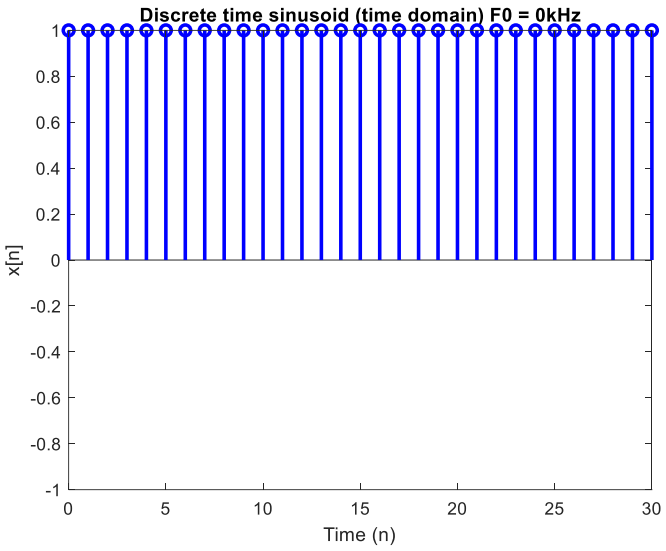
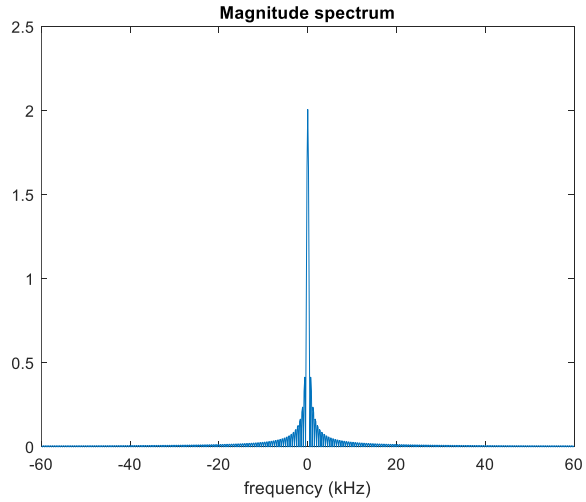


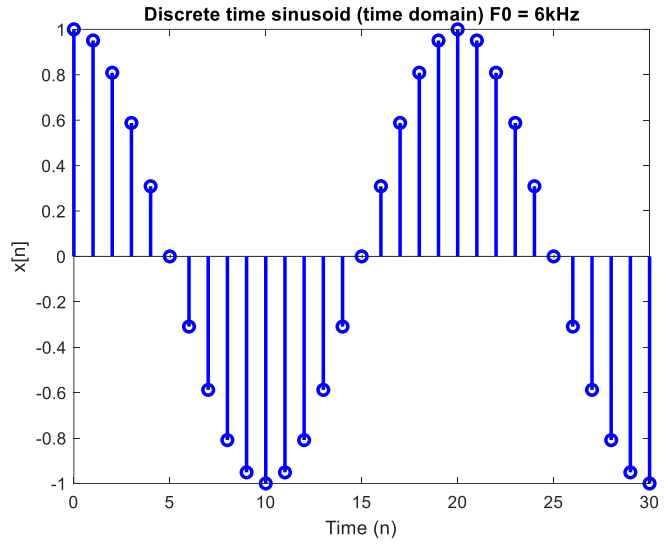
If we change frequency range to [-2Fs 2Fs], then normalized frequency is changed to [-4 π 4 π], which is 4 times fundamental frequency range and thus 4 periods of the frequency domain signal can be found in the following spectrum plot. The result is shown below.

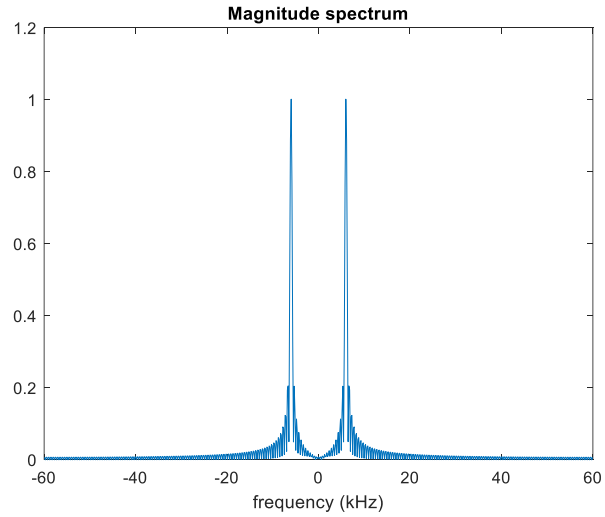


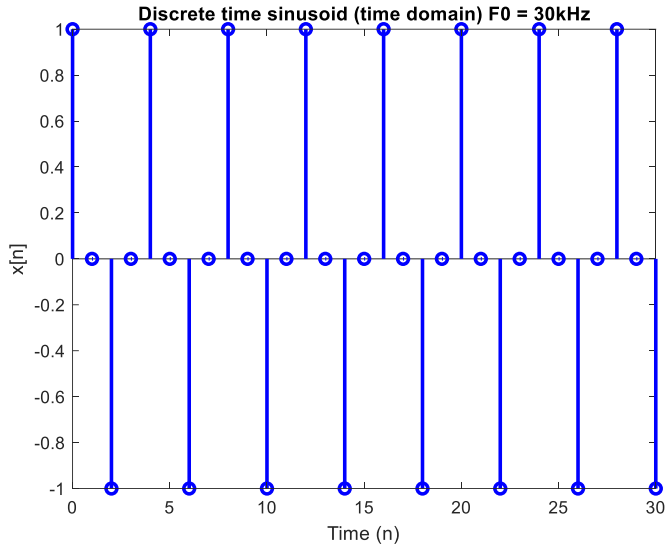
We can find that magnitude spectrum in normalized angular frequency is periodic with its period = 2π.

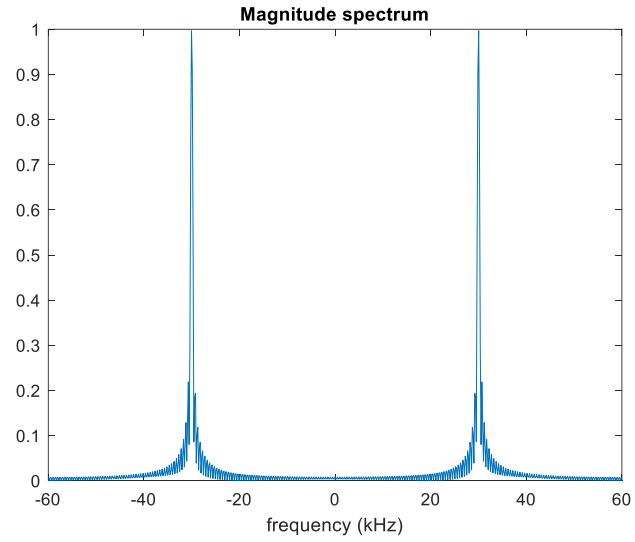
(f)

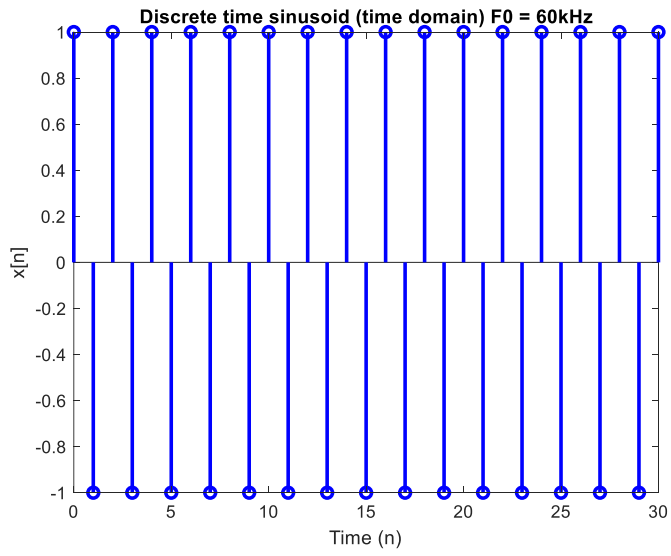
Time domain	Frequency domain
	

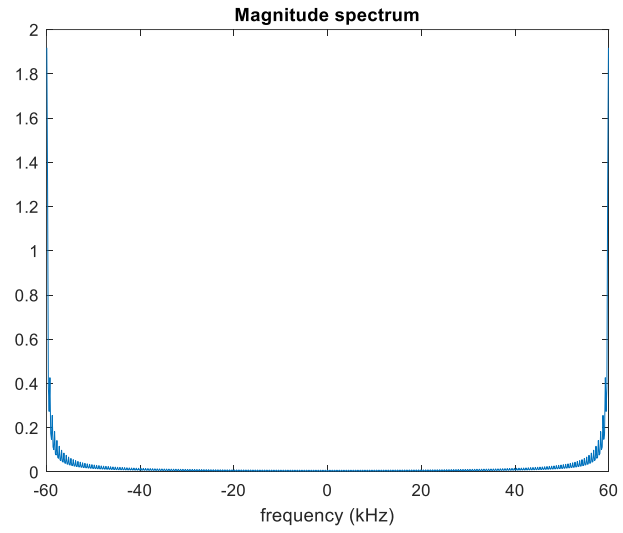

Discrete time sinusoid (time domain) F0 = 6kHz
x[n]
Time (n)

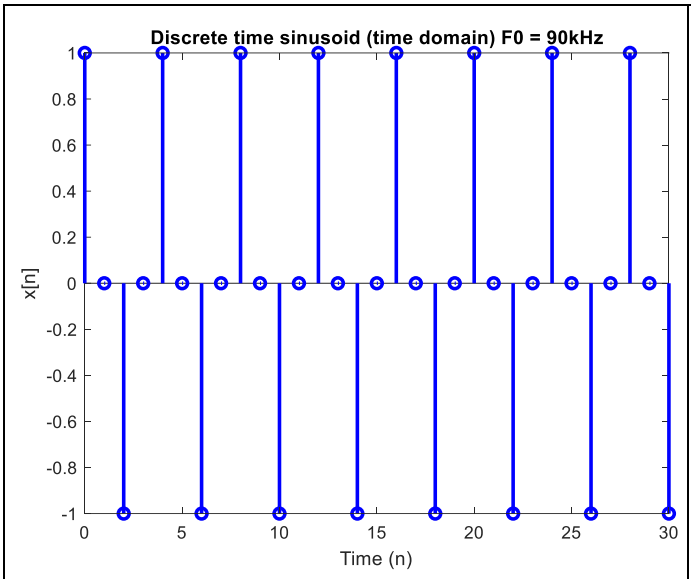

Magnitude spectrum
frequency (kHz)

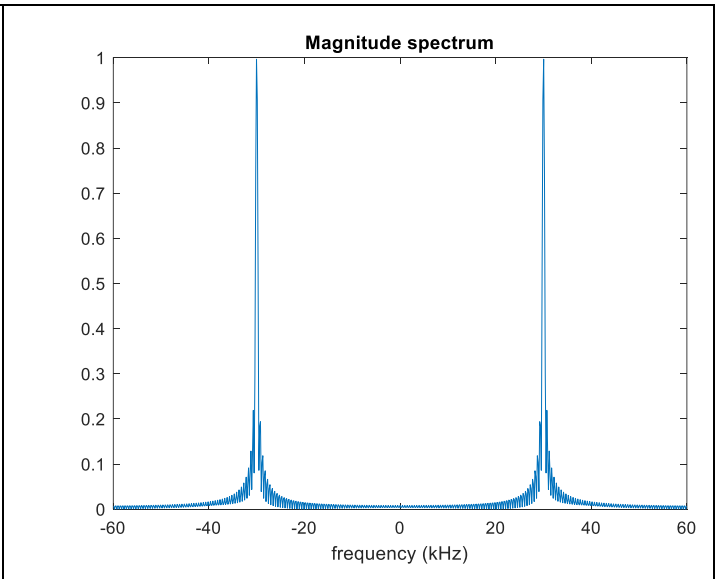

Discrete time sinusoid (time domain) F0 = 30kHz
x[n]
Time (n)

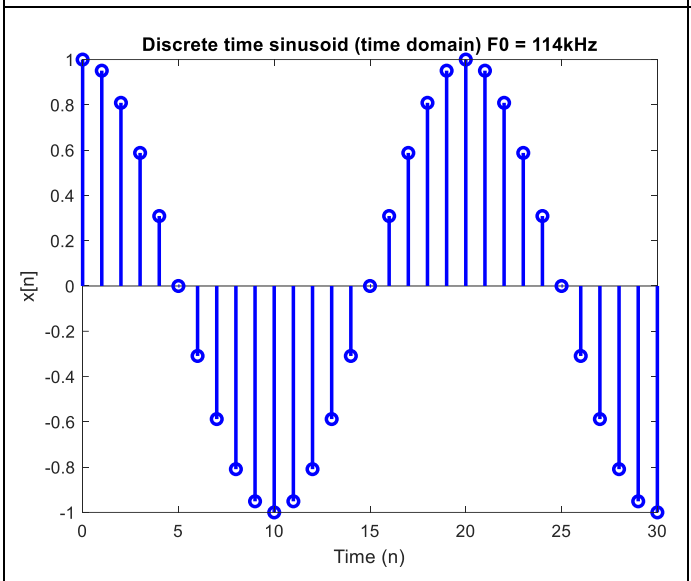

Magnitude spectrum
frequency (kHz)

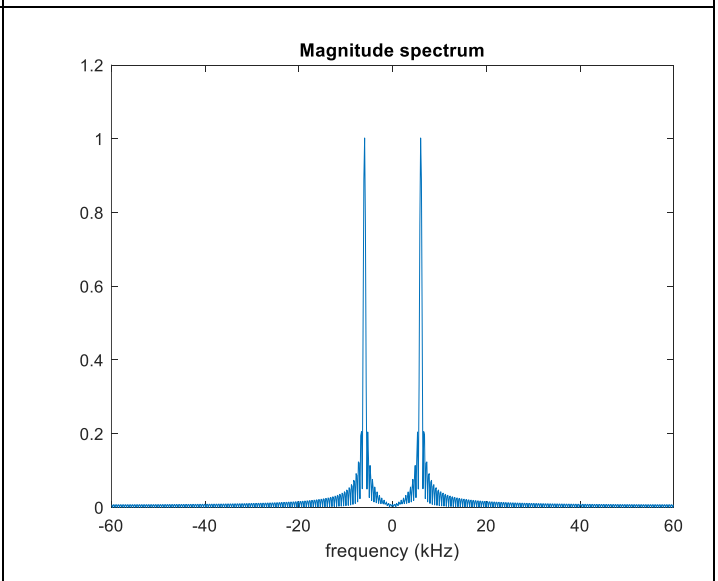

Discrete time sinusoid (time domain) F0 = 60kHz
x[n]
Time (n)

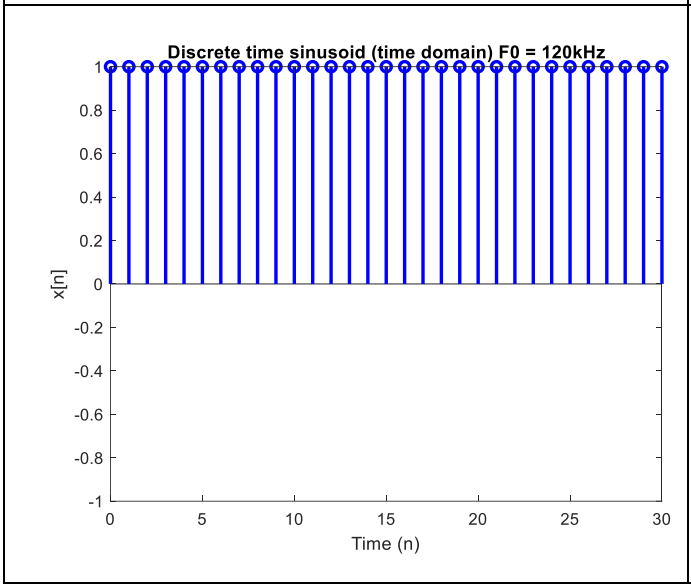

Magnitude spectrum
frequency (kHz)

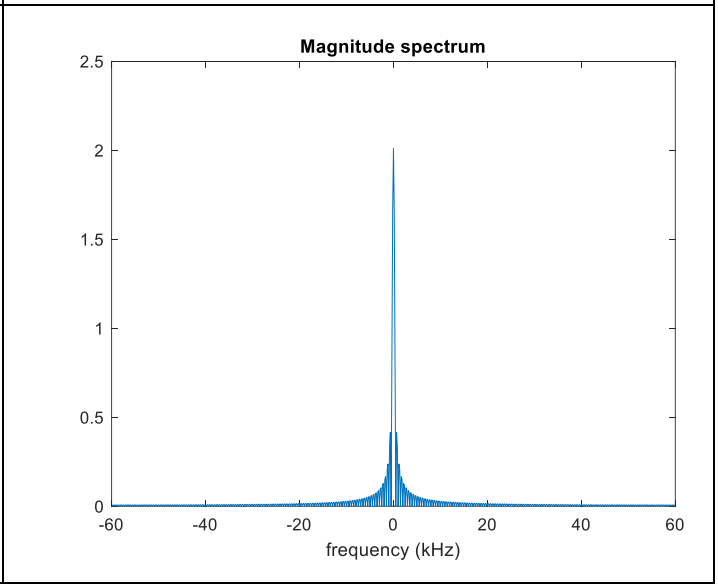












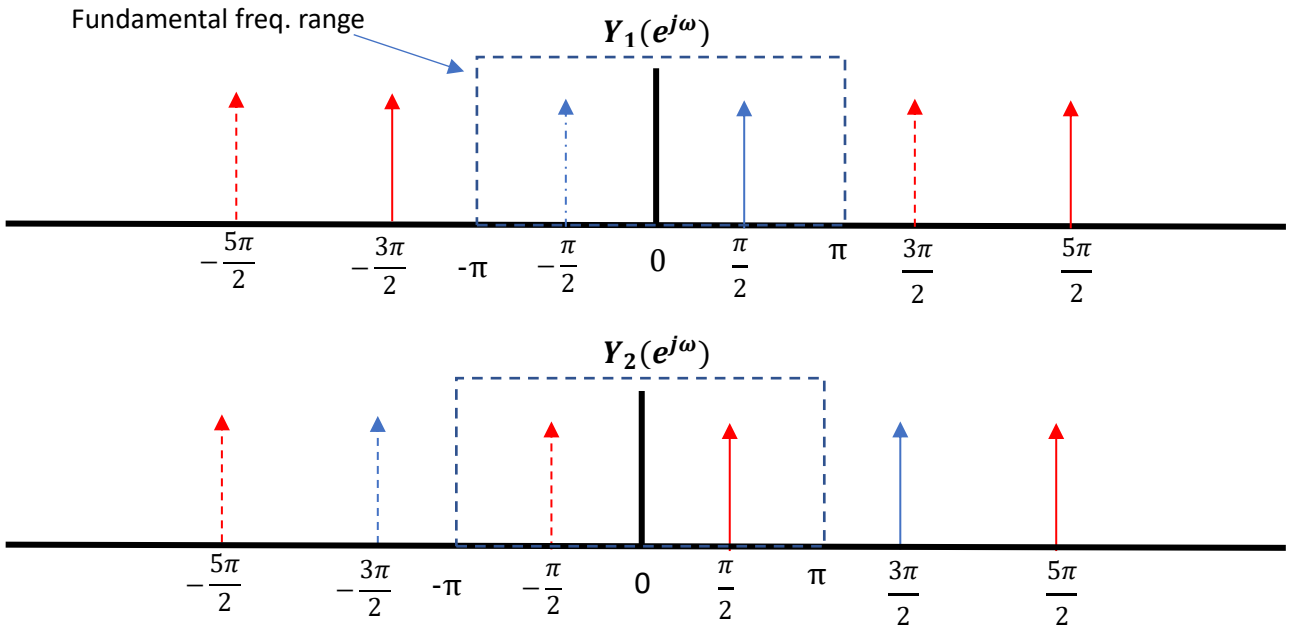
Comparing sinusoid signals and their spectra, we find that if F0<60kHz, the peaks of the spectra appear at correct frequencies. If F0>60kHz, the spectra are not the expected spectra because the peaks appear at incorrect frequencies, which are aliased frequencies. The signals which F0 are 0 kHz and 120 kHz (0 kHz is its aliased frequency) look the same in time

domain. So do the signals which F0 are 6 kHz and 114 kHz (6 kHz is its aliased frequency) and the signals which F0 are 30 kHz and 90 kHz (30 kHz is its aliased frequency).

Same as Part 1 (e), we convert absolute frequency axis into normalized angular frequency axis. From slides 40-45 of Topic3_FourierRepresentation_Part3_1_HandWriting0505_2021.pdf, note that the DT signals are from the multiplication of CT signals and impulse train, which spectra will be periodic in normalized angular frequency and are with the fundamental period = 2π. We know that the sampling frequency Fs is 120 kHz and the corresponding normalized angular frequency is 2π.

F0	normalized angular frequency
0	0
6	$\pi/10$
30	$\pi/2$
60	π
90	$3\pi/2$
114	$19\pi/10$
120	2π

Below figures are the spectra of the DT sinusoid signals which normalized frequencies are$\pi/2$ and $3\pi/2$ (which aliased frequency is $\pi/2$), respectively, named as $Y_1(e^{j\omega})$ and $Y_2(e^{j\omega})$. The blue delta functions in frequency domain are the expected original spectra. Since we use T*DFT to get the spectrum (or DTFS*N*T which the implemented CTFT code does), it is periodic in frequency. And the other delta function replica (since the spectra are periodic) are red.



According to sampling theorem, sampling frequency Fs must be larger than twice the maximum frequency of the signal. From the point of view of normalized frequency, we can know that the DT (complex) sinusoidal signal is periodic, there is a fundamental frequency range and the highest oscillation rate (i.e., the highest frequency) is π. All these properties of the DT sinusoids come from sampling and sampling theorem. That is, these properties can be explained by sampling and sampling theorem.

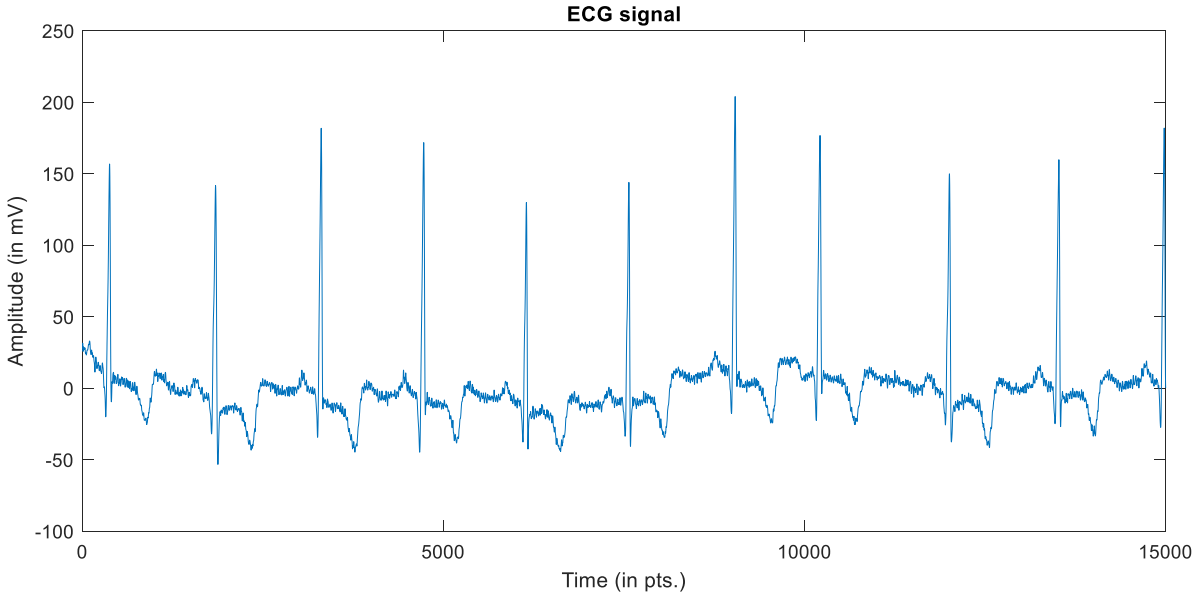
(g)

We can know that the working frequency range is [-Fs/2 Fs/2]. From the result of (f), if F0 is bigger than Fs/2, aliasing in frequency domain happens.

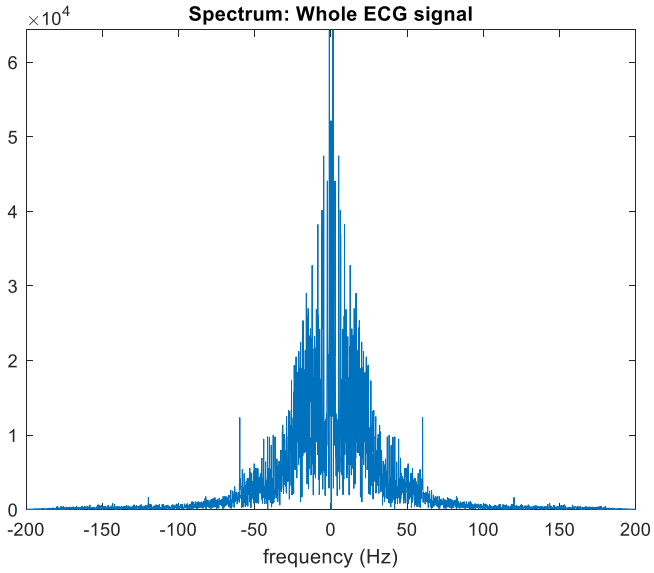
Part 2:

(a)

Whole Signal

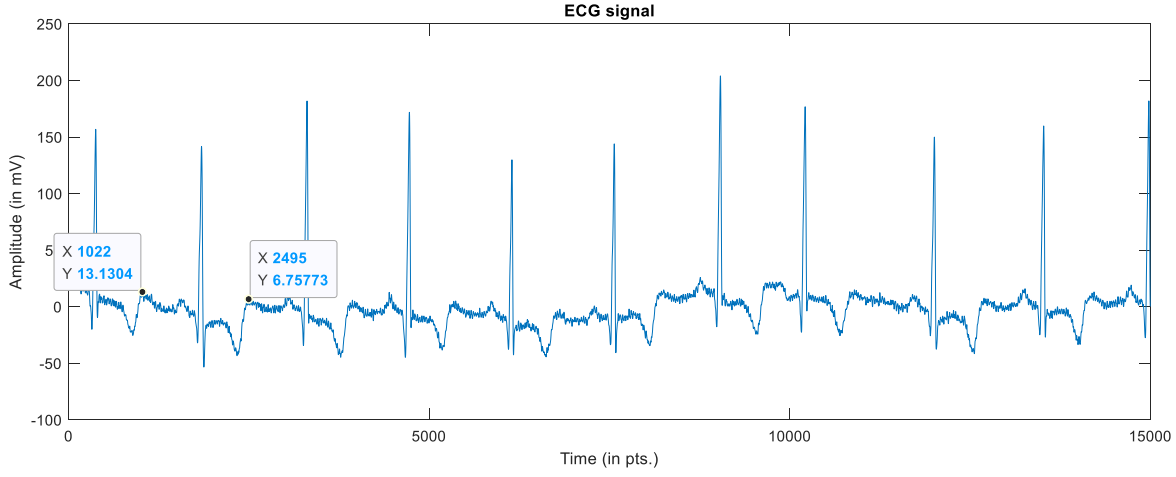


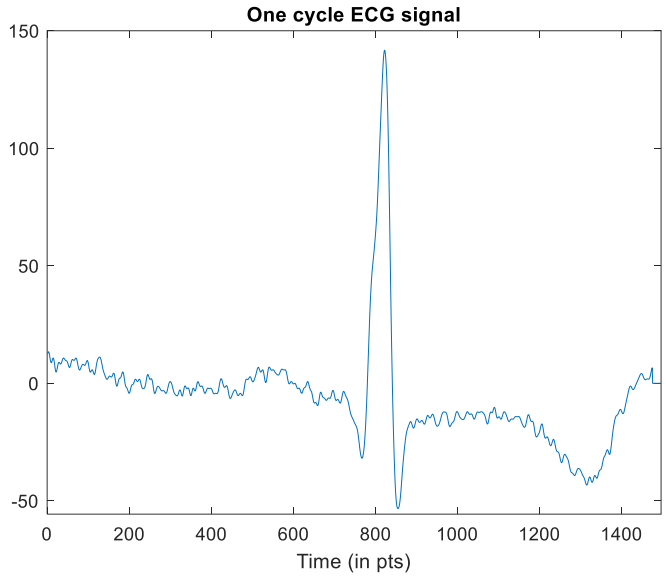
The spectrum of the full ECG signal is shown below.



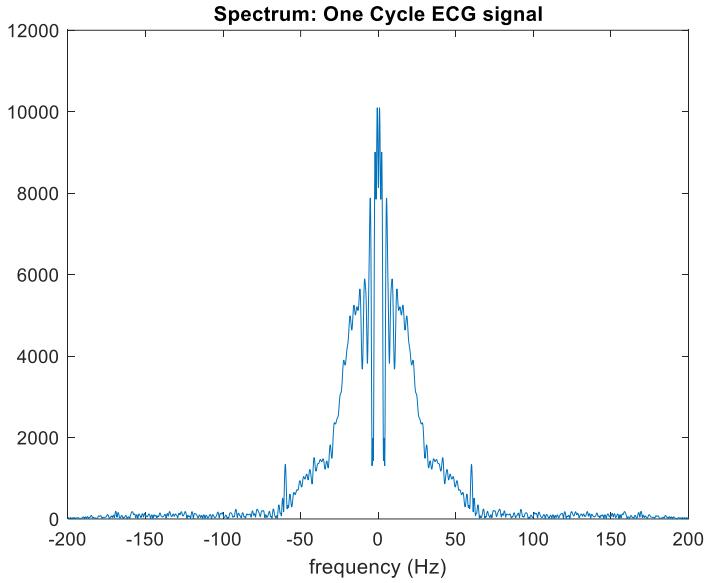
Single-cycle ECG Signal

We can acquire a cycle ECG wavelet from data points 1022-2495, as the figure below.

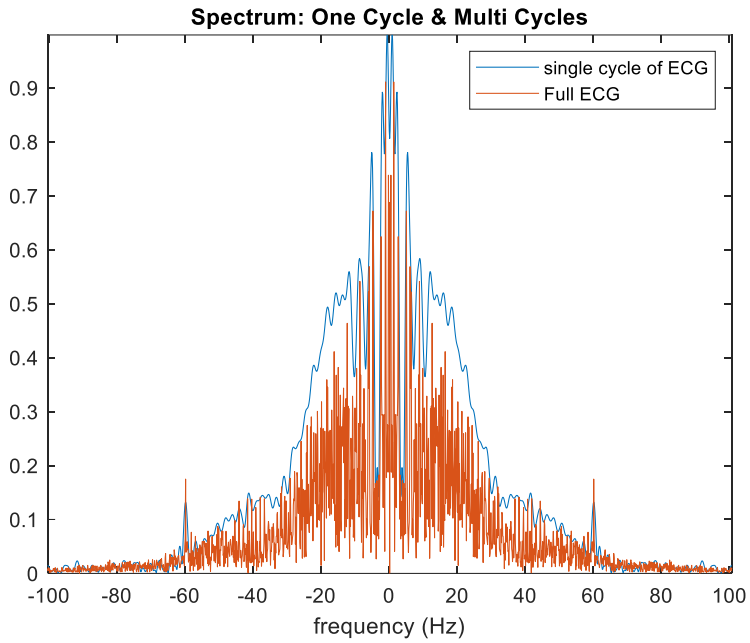




The spectrum is plotted below.



Compare the two spectra by plotting them together and zooming in.



We can see that:

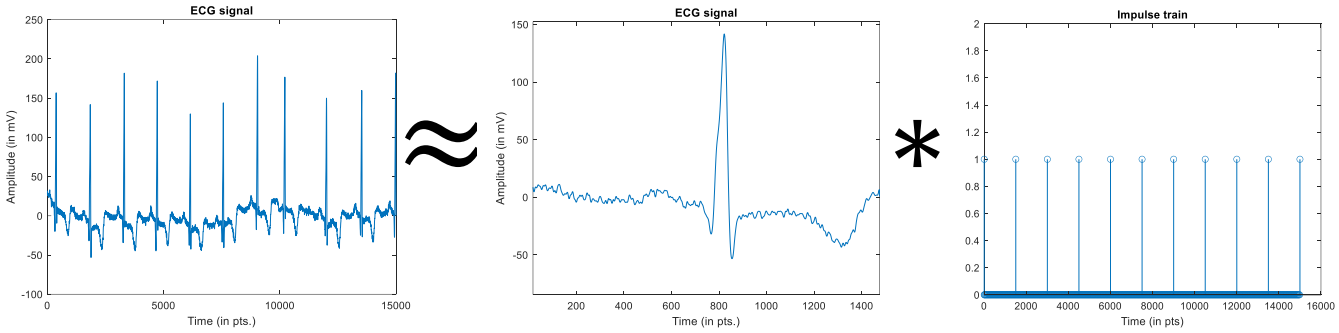
(a) The spectrum of full ECG (more peaks in the spectrum) is slightly different from that of single ECG (relatively smooth spectrum).

(b) The existence of power line noise at 60Hz and its harmonics (not so obvious) in both of the spectra.

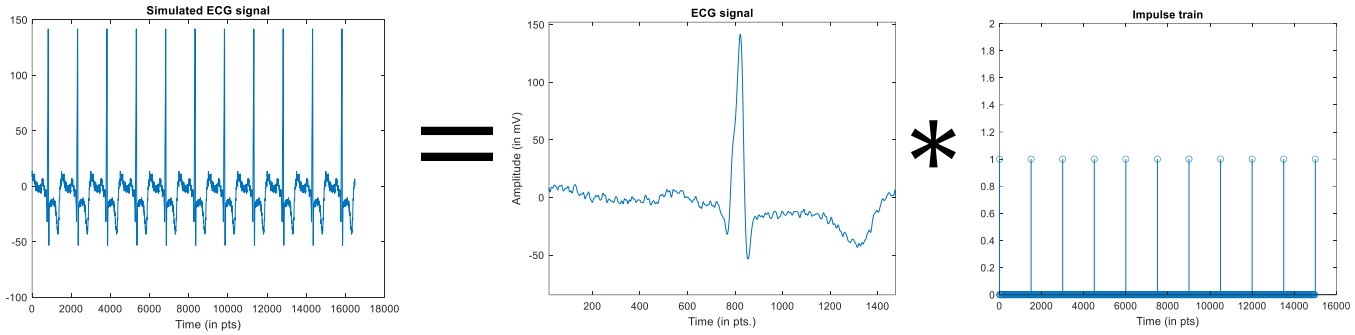
(c) The bandwidth (i.e., the maximum frequency components) of the two are almost the same.

Therefore, from the above observation, we conclude that both magnitude spectra can provide the spectral information needed for ECG front end circuit design. (e.g., determine bandwidth of a pre-amplifier for signal amplification, frequency response of an analog filter for noise reduction, and a proper ADC sampling rate.)

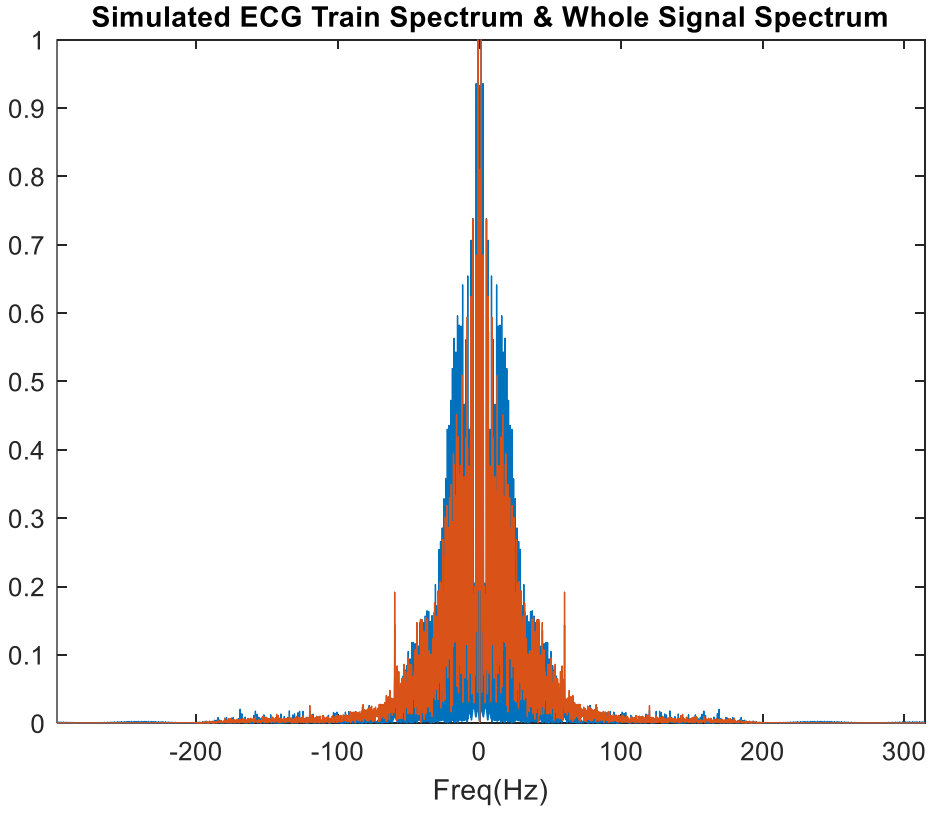
The reason causing the difference in the two spectra is explained as follows. If we compare the signal of the whole ECG signal and one-cycle ECG signal, we can approximately view the whole ECG signal as the result of one-cycle ECG signals convolving with a truncated or windowed impulse train (see the following illustration and slide 29, Topic3_FourierRepresentation_Part2_3_withNotes.pdf). If the heart rate is constant over the data acquisition time (though in fact, it is impossible to have a constant (time-invariant) heart rate. Heart rate is time-varying).



The result of such convolution is shown below (The heart rate, i.e., the inverse of R-R interval, is constant).



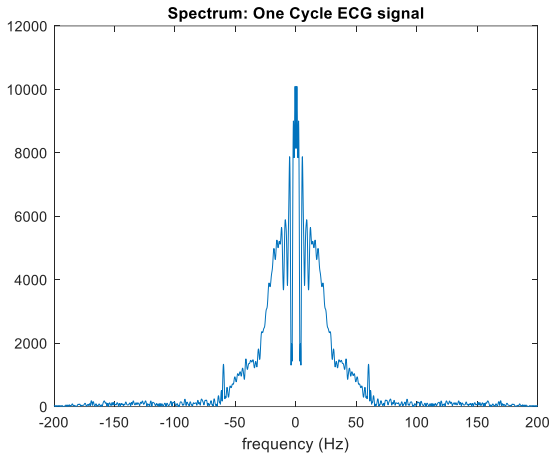
The spectrum of the simulated ECG train (using the above convolution procedure) is shown below (in blue), which is similar to the spectrum of the full ECG signal (in red).

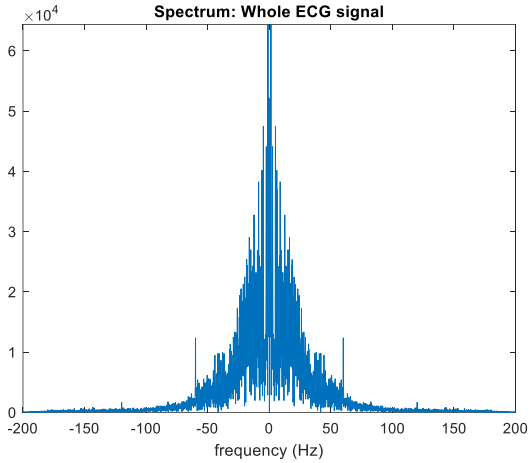


Based on such a convolution approximation, the spectrum of the full ECG signal tells no

more frequency-domain information than a single cycle signal can provide because the full ECG spectrum is the single cycle ECG spectrum times an impulse train in frequency domain. That is, both of the spectra can provide the same maximum frequency information and show the same power line noises. However, the spectrum of the single-cycle ECG is more clear owing to being without the multiplication of FT{truncated impulse train}) , so it is relatively easier to identify the maximum frequency and noises.

(b) (See also the above rationale)

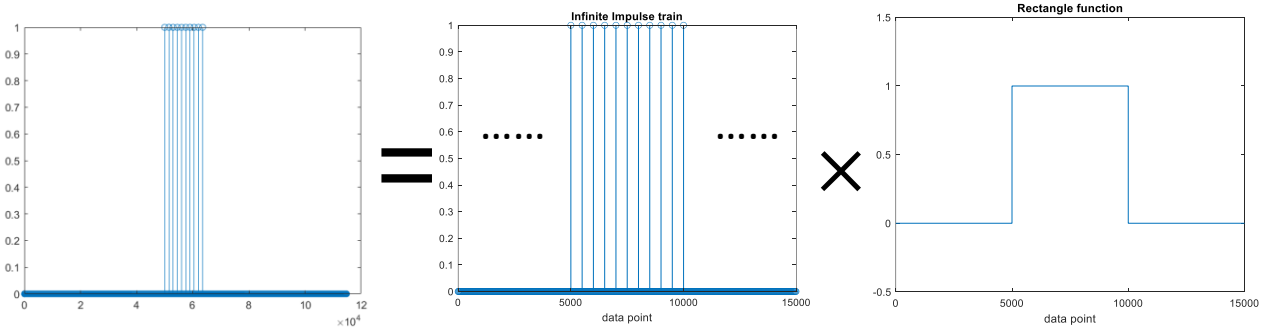




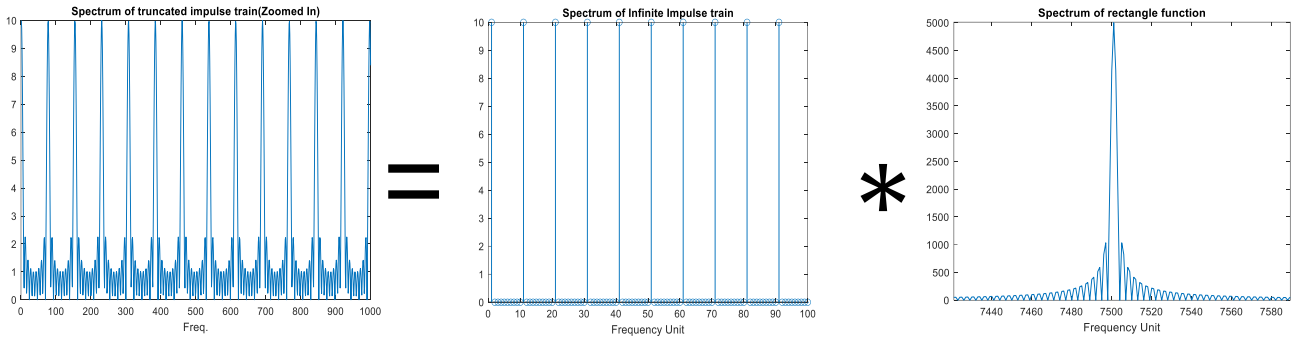
The spectrum of one cycle ECG signal looks smoother. From (a) , we can approximately view the whole ECG signal as the convolution of One-cycle ECG signal and truncated impulse train. That is, we can also approximately view the whole ECG spectrum as the multiplication of One-cycle ECG spectrum and FT{truncated impulse train}.

The truncated impulse signal can be seen as the windowed impulse train in time domain. That is, truncated impulse train is the multiplication of impulse train and rectangle function. Thus, we can see window effect in frequency domain or in the spectrum of the truncated signal (From slide 42-43, Topic3_FourierRepresentation_Part3_1_withNotes.pdf), as the figure shown below.

In time domain:

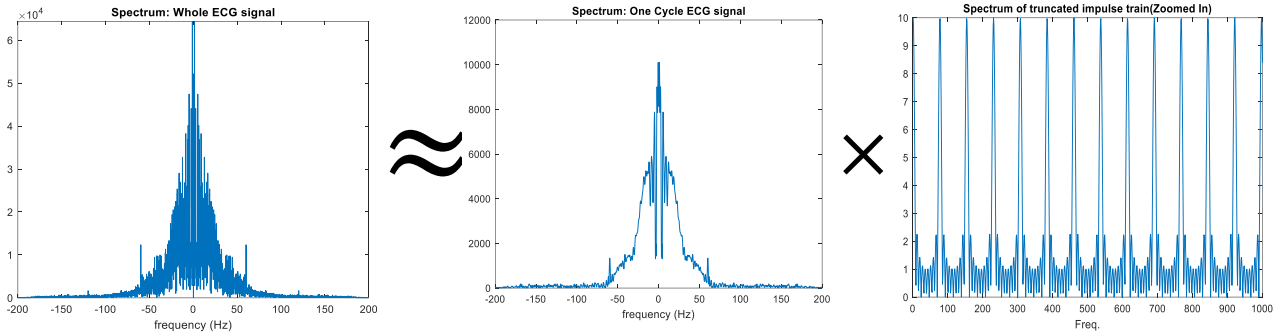


In frequency domain:

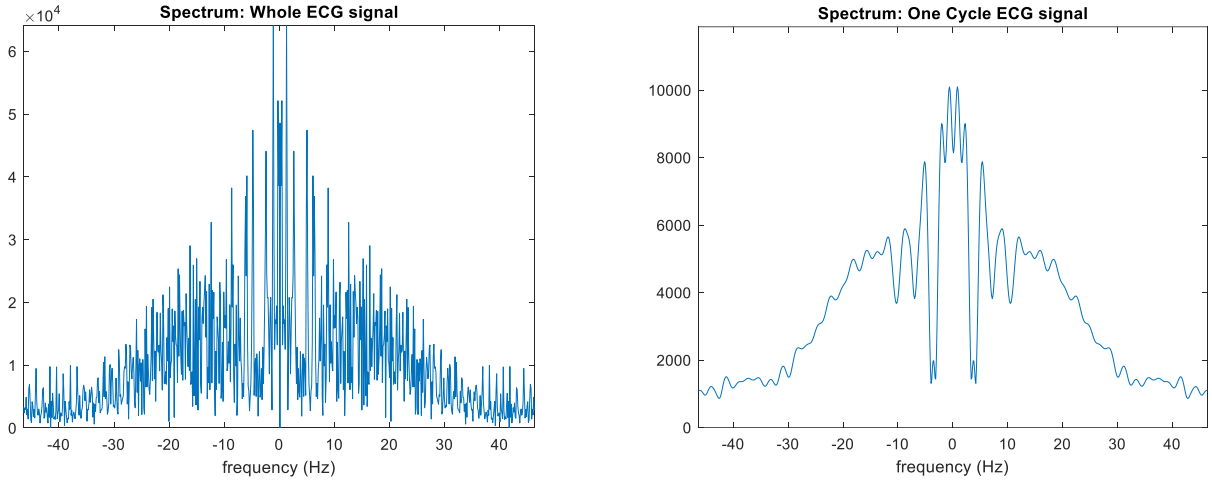


(Note that Fourier transform of impulse train is also impulse train with fundamental period = 1/ time domain fundamental peirod. (See slide 28, Topic3_FourierRepresentation_Part2_3_withNotes.pdf).

Therefore, the whole ECG spectrum is the multiplication of One-cycle spectrum and the spectrum of the truncated impulse train spectrum.



And we zoom in the two spectra.



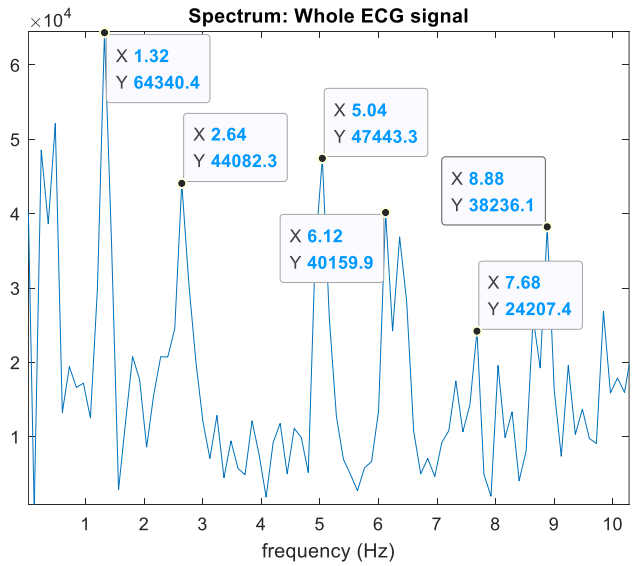
The left spectrum shows a lot of peaks which are caused by the peaks in the spectrum of the truncated impulse train spectrum (i.e., from the multiplication).

(c)

First, as mentioned in the lectures, please note that the definition of the heartbeat frequency (the inverse of R-R interval) is different from that of the frequency in Fourier analysis, which is the frequency of a (complex) sinusoid or the inverse of the fundamental period of a (complex) sinusoid. Intuitively, the heartbeat frequency cannot be figured out simply from the frequency components in the Fourier spectrum.

However, if we treat multiple cycle ECG signals (whole ECG signal) as impulse train, R-R interval is the fundamental period of the impulse train if the heart rate is constant over all the data acquisition time. Based on this assumption, the inverse of the R-R interval will be the fundamental period of FT{impulse train} in frequency domain (See slide 28, Topic 3 Fourier Representation of Signals and Systems Part 2-3 withNote.pdf). **That is, if the R-R interval is time-invariant, we can figure out the inverse of R-R interval (the heart rate) by finding out the fundamental period of the FT{impulse train} in the frequency domain, which will be the peak-to-peak interval in the whole ECG spectrum.**

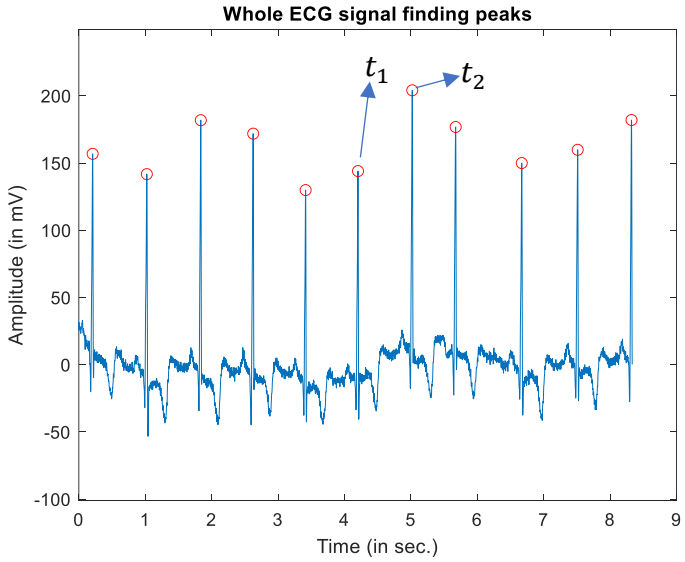
Unfortunately, in practice, R-R interval is time-varying which you can find from the given whole ECG signal; thus, the heart rate (i.e., the heartbeat frequency; the inverse of the R-R interval) cannot be figured out from the spectrum of the given whole ECG signal. Below is the zoom-in of the spectrum of the whole ECG signal.



Because the heart rate (the inverse of R-R interval) is time-varying, we can see the peak-to-peak intervals between successive peaks in the spectrumare not the same and thus it is difficult to identify the heart rate from the spectrum. In addition, even if the heart rate

is constant, the inverse of the R-R interval can be found from the constant peak-to-peak interval in the whole ECG spectrum. It is not the INSTANTANEOUS heart rate (i.e., heart rate counted for each heartbeat). It is the AVERAGE heart rate over the data acquisition time (the overall time use for the Fourier analysis). Therefore, in general, we estimate the instantaneous heart rate by locating the R peak position and finding the R-R interval in TIME domain.

We know that R-R interval is the interval of two peaks of ECG signals. Therefore, we can get the approximate heart rate by findpeak() function(see the documentation of MATLAB) or just using our eye without Fourier transform. Then R-R interval is $t_2 - t_1$. It is better way to find out the heart rate.



(d)

In this case, we need to remove 60Hz and its harmonics (i.e., 120Hz, 180Hz, etc.). A moving average filter can be used as a multiple-notch filter to remove the power line noises at 60 Hz and its harmonics by making 60Hz and the harmonics on the zero-crossing points in its frequency response.

The model of a moving average filter:

$$y[n] = \frac{1}{N}\sum_{k=0}^{N-1} x[n-k] \quad => h[n] = \begin{cases} \frac{1}{N}, 0 \le k \le N-1 \\ 0, otherwise \end{cases}$$

$h[n]$ is a rectangular pulse, and the frequency response is $H(e^{j\omega}) = \frac{1}{N}\frac{\sin\left(\frac{N\omega}{2}\right)}{\sin\left(\frac{\omega}{2}\right)}$,where N

is the length of the moving average filter.

If $H(e^{j\omega}) = 0$, $\omega = the\ normalized\ angular\ frequencies\ of\ 60Hz\ and\ the\ harmonics,$ we can remove 60 Hz power line noises (See slides 54-55, Topic 5 Time and Frequency Characterization of Signals and Systems_withNotes.pdf).

Convert 60 Hz and its harmonic frequencies (e.g., 120 Hz and 180 Hz) to the corresponding normalized angular frequency ω_c

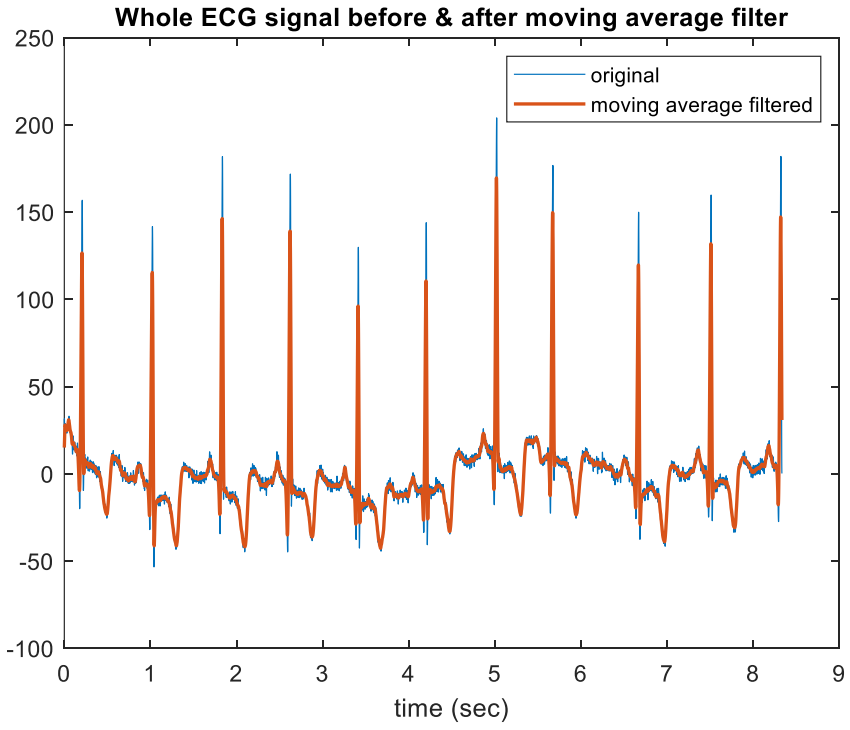
$$\omega_c = \frac{60\times n}{Fs} \times 2\pi = \frac{60\times n}{1800} \times 2\pi = \frac{n\pi}{15}, n = \pm1, \pm2, \pm3, \dots$$

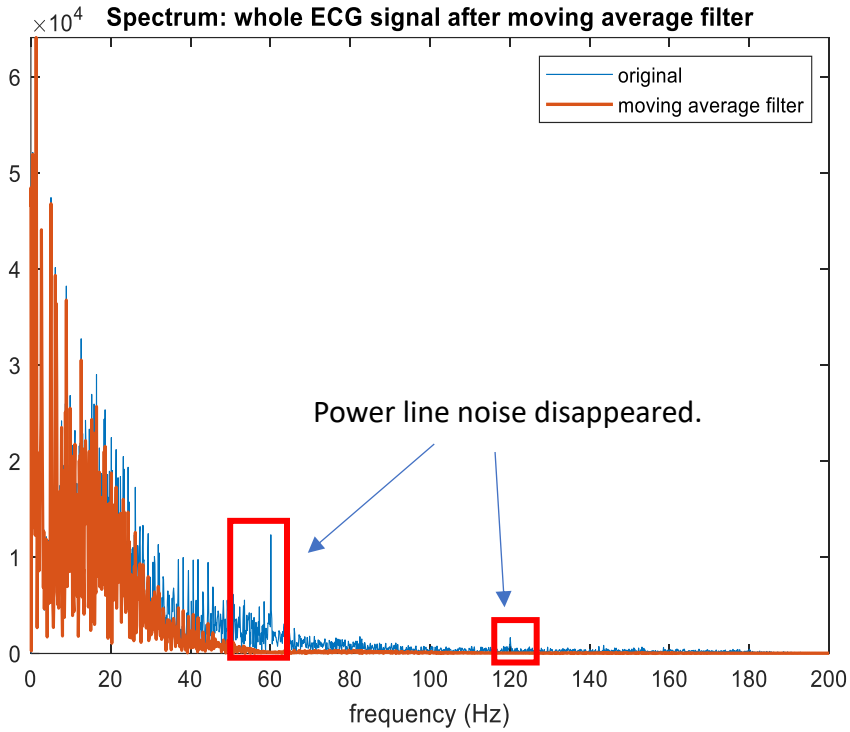
Utilize the zero-crossing in the moving-average-filter frequency response to remove the power line noises.

$$\because \frac{\sin\left(\frac{N\omega_c}{2}\right)}{\sin(\frac{\omega_c}{2})} = \frac{\sin\left(\frac{N}{2}\times\frac{k\pi}{15}\right)}{\sin\left(\frac{k\pi}{30}\right)} = \frac{\sin\left(\frac{k\pi}{30}N\right)}{\sin\left(\frac{k\pi}{30}\right)} = 0 \text{ when } \frac{k\pi}{30}N = k\pi, \text{where } k = \pm1, \pm2, \pm3, \dots$$

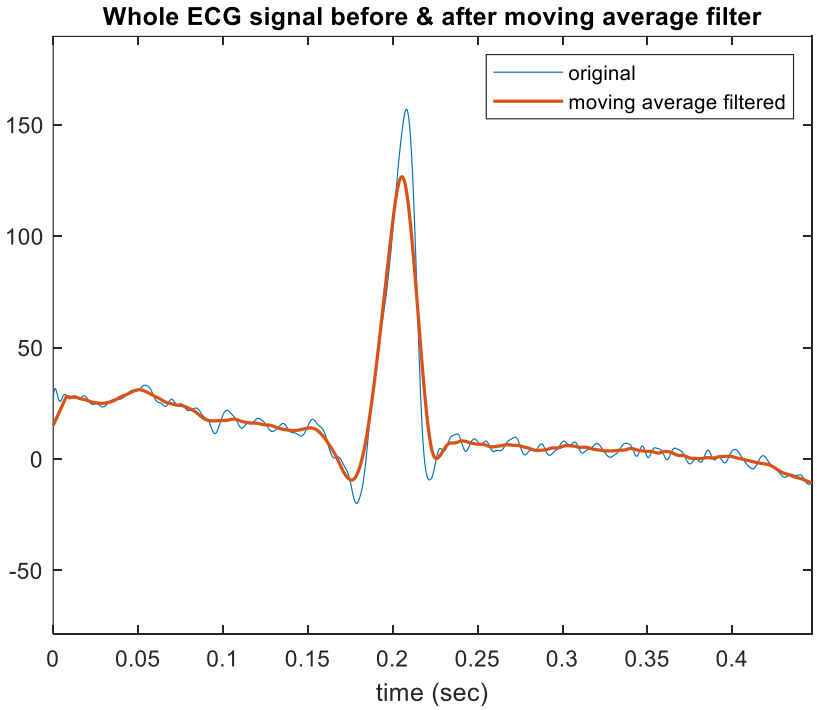
$$\therefore N = 30_{\#}$$

From the above result, we set the length of moving average filter to 30. From the time domain and frequency domain ECG signals before and after moving average filtering shown below, it is found that the power-line noises are suppressed by the designed moving average filter.





Zoom in the result in time domain.



(e)

In EchoGen.m, we set parameters a=0.7 D=819, same as Computer HW2 Part 2. Then we derive the magnitude response from the equation provided in Computer HW2 Part 2.

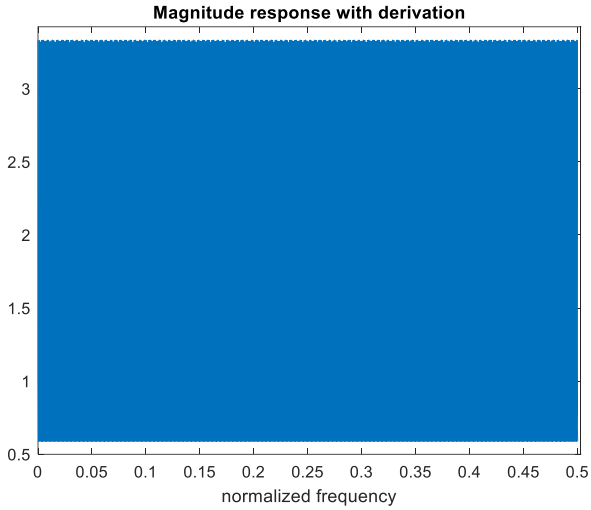
Derivation of magnitude response:

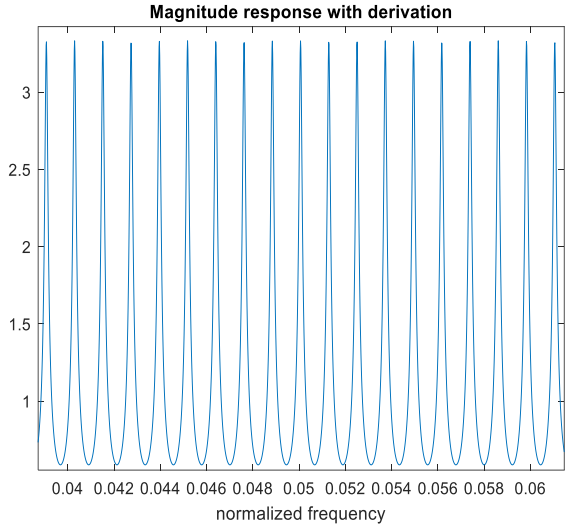
$$y[n] = 0.7y[n-819] + x[n]$$

$$Y(e^{-j\omega}) - 0.7e^{-j819\omega}Y(e^{-j\omega}) = X(e^{-j\omega})$$

$$H(e^{-j\omega}) = \frac{Y(e^{-j\omega})}{X(e^{-j\omega})} = \frac{1}{1 - 0.7e^{-j819\omega}}$$

Plot the magnitude response based on the above derivation and zoom it in.





The result of zooming in shows that there are lots of multiple stopbands, and frequencies of stopbands are located in times of certain fundamental frequencies. It makes the spectrum look like a comb. Therefore, we called it "Comb" reverberator.

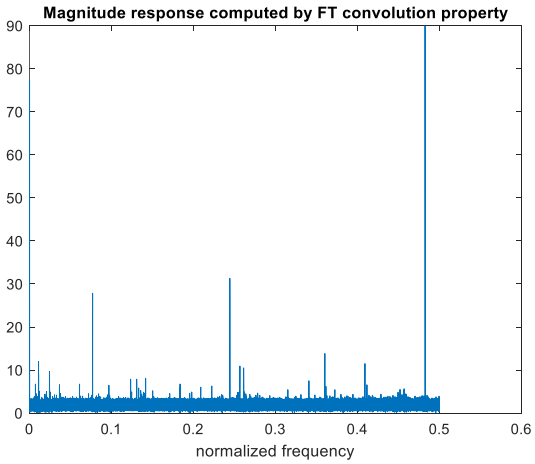
Then $y[n]$ is input , and $y_e[n]$ is output. Then we can use FT convolution property to find out $H(e^{-j\omega})$. The plots of the magnitude response and its zoom in are shown below.

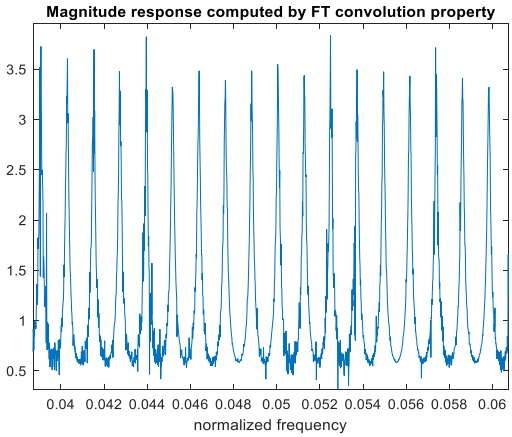
By FT convolution property:

$$y_e[n] = y[n] * h[n]$$

$$Y_e(e^{-j\omega}) = Y(e^{-j\omega}) \times H(e^{-j\omega})$$

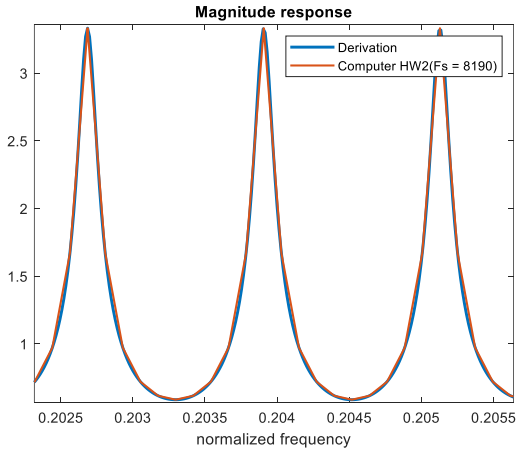
$$\rightarrow H(e^{-j\omega}) = \frac{Y_e(e^{-j\omega})}{Y(e^{-j\omega})}$$



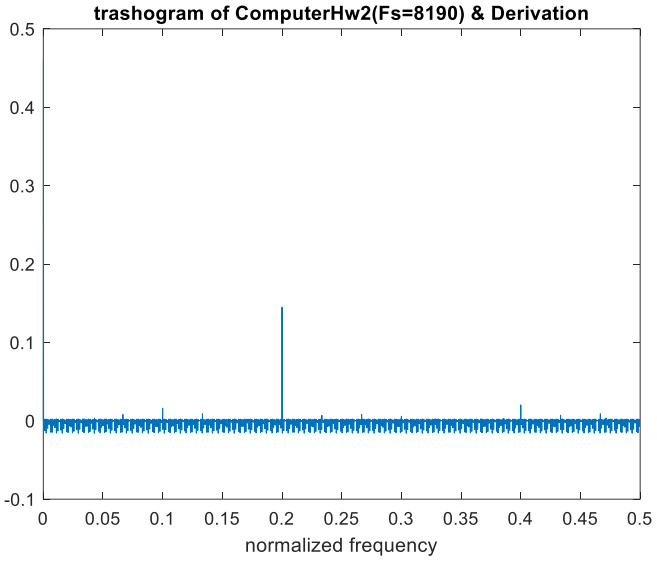


Compare the results with magnitude response we obtained in Part2(g) of Computer HW2.

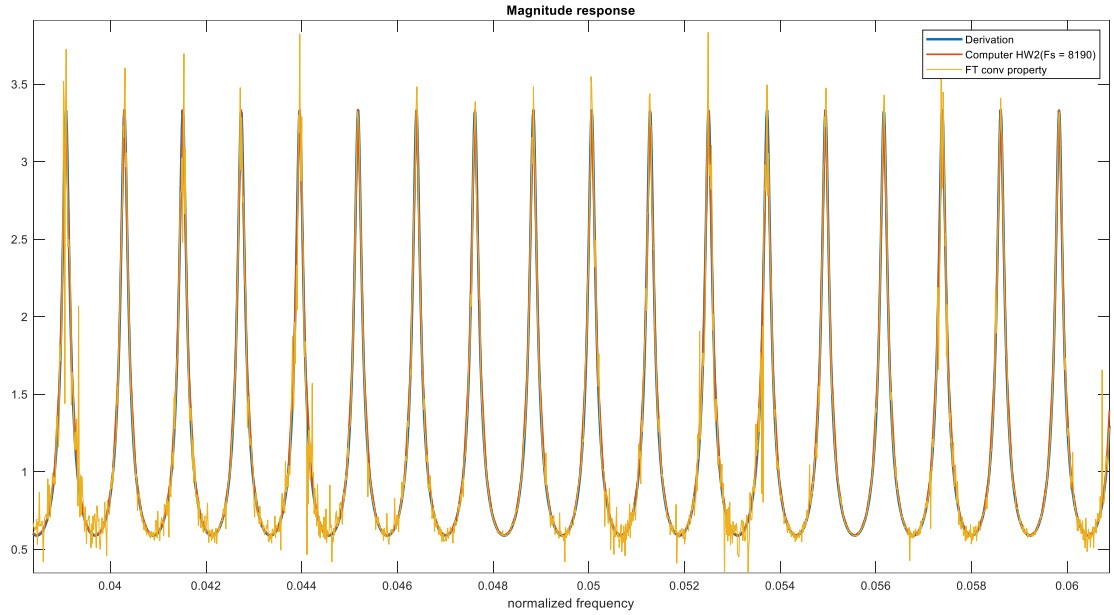
Derivation Result and Part2(g) of Computer HW2.



If we substract the results, we get the trashogram. Plot the trashogram as follows.



In the following figure, we compare the three results: Part2(g) of Computer HW2、Derivation Result and FT convolution result. **The FT convolution property result slightly deviates from the results of computer HW2 and the derivation, which results from numerical errors of the implemented CTFT codes or MATLAB function fft().**


Magnitude response
Derivation
Computer HW2(Fs = 8190)
FT conv property
3.5
3
2.5
2
1.5
1
0.5
0.04
0.042
0.044
0.046
0.048
0.05
0.052
0.054
0.056
0.058
0.06
normalized frequency