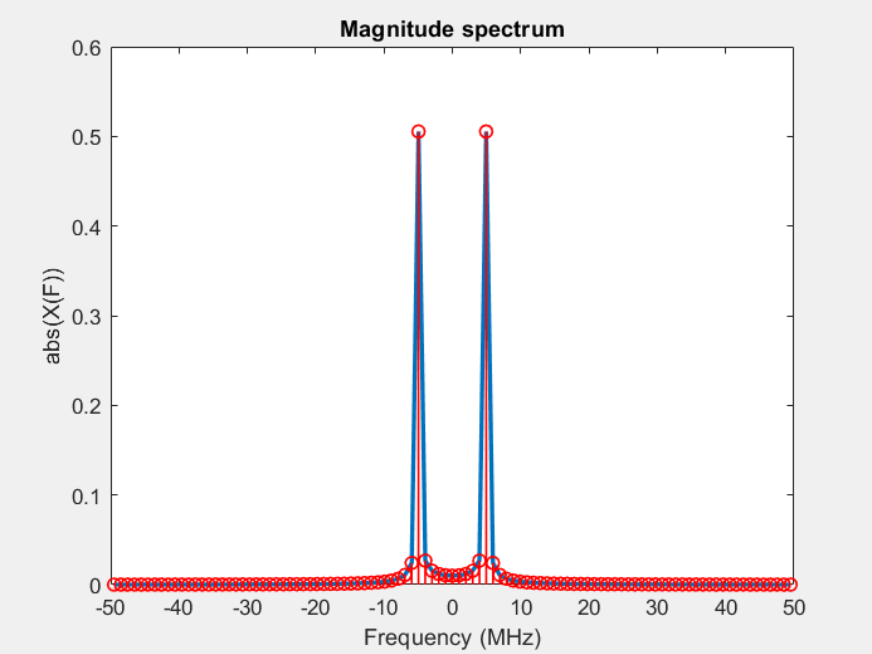
EECS2020 常安彥 107081028 Computer HW2 05/21/2020

Part 1

1. Go through ComputerHW3_SampleCodes.m, run the codes, and tell whether or not the computed magnitude spectrum is the same as what you expect. Please elaborate why you have such a computed magnitude spectrum for a cosine signal.

Magnitude spectrum:



Yes, the graph looks as I expected resulting from rewriting cosine into

$$\cos(\theta) = \frac{e^{j\theta}+e^{-j\theta}}{2}.$$

Through the code, it is expected to occur two peaks at about (-5) and 5 with absolute value at about 0.5. If actually calculate the values with the information in the code, the peak value is 0.5055.

```
F0 = 5; % in MHz
Fs = 100; % sampling rate/sampling frequency, in MHz or Msamples/sec
T = 1/Fs;  % time resolution, i.e., sampling interval in time domain
total_time = 1; % in us

% !!! Sampling in time
t_axis = (0:T:total_time);  % time axis
x = cos(2*pi*F0*t_axis);  % sampled cosine/discrete time sinusoid ,time domain
Npoint = length(x);   % number of points in sampled cosine
```

```
% !!! Sampling in frequency
dF = Fs/Npoint; % frequency resolution, i.e., sampling interval in frequency domain
%f_axis = (0:1:(Npoint-1))*dF;   % frequency axis (from 0 to Fs or equivalently from 0 tc
f_axis = ((1:1:Npoint)-(Npoint+1)/2)*dF; % frequency axis (from -Fs/2 to Fs/2 or equivale
X = zeros(1,length(f_axis)); % spectrum

% 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
```

```
%% Part 1.7
% !!! You can compare the result with that from MATLAB fft()
%X = fft(x); % spectrum of sampled cosine, frequency domain, complex; Any difference from
%f_axis_forFFT = ???; %frequency axis for fft(x), from 0 to Fs or equivalently from 0 to

mag_X = abs(X);   % magnitude
pha_X = angle(X); % phase
```

2. Which Fourier representation (CTFS, CTFT, DTFT, DTFS) does the sample codes actually implement although I am trying to implement CTFT? Please verify your answer.

Continuous time cannot be modeled in computer because of the impossibility to set a value for every interval with the limited memory. Furthermore, periodicity cannot be examined due to the infinity.

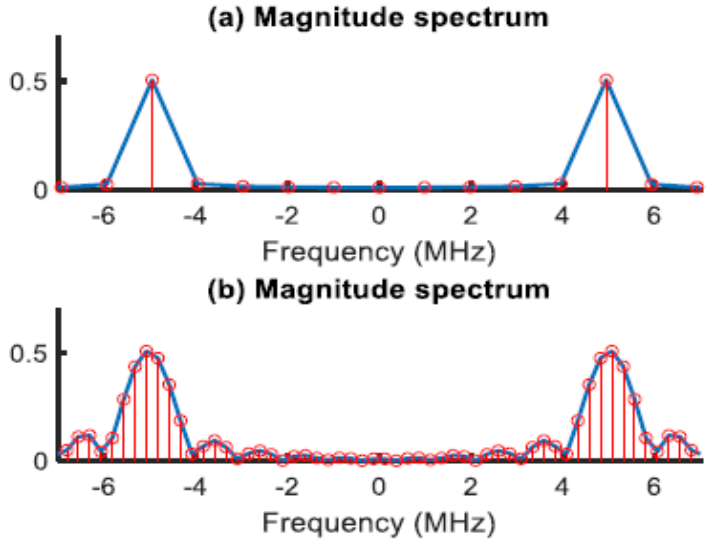
DTFT is the closest representation the sample codes implement in my opinion. Nevertheless, using different sampling frequency results in different signals, which are enlonged as the increase of Fs. T, the sampling interval, appeared to scale the interval: the value can be set to 0 out of the interval T in order to create a periodic circumstance with a period exist though impossible to determine.

The formula fits CTFT although computers can only model discrete values.

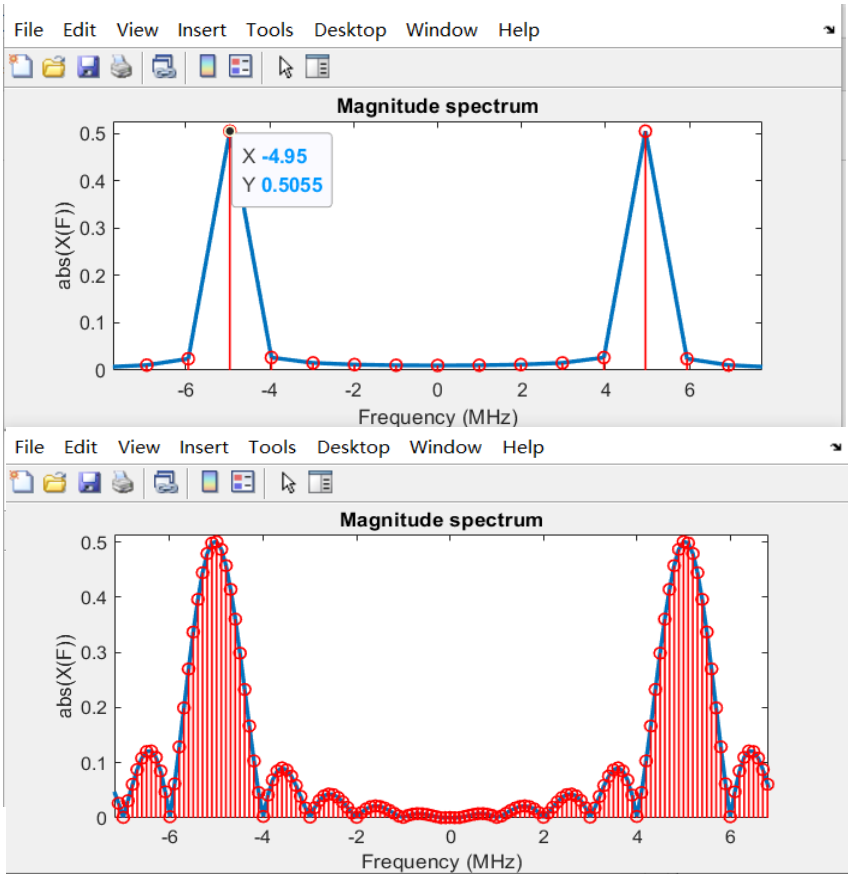
The time domain and frequency domain are both discrete in DTFS, which a computer can act without approximation. However, the periodicity is not perfect.

3. In the following figure, (a) is the magnitude spectrum generated by the provided sample codes, which is poor sampled, i.e., not so smooth, and (b) is the smoother magnitude spectrum which is desired in practice because it is closer to the ideal CTFT

spectrum. Please provide your strategy, elaborate your idea, and modify the provided sample codes accordingly to obtain a smoother magnitude spectrum as shown in (b). Remember to show how you modify the codes in your report.



Through the figure above, we can know that the difference between figure(a) and figure(b) is the number of samples between the same interval. Therefore, I modified the t_axis part, and increased the samples through the same sampling interval. The consequence:

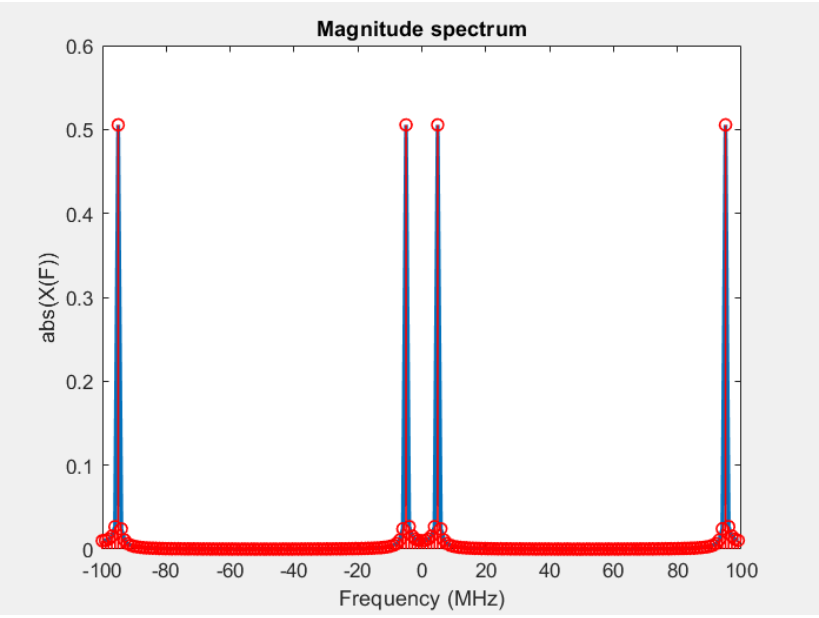


Code:

```
F0 = 5; % in MHz
Fs = 100; % sampling rate/sampling frequency, in MHz or Msamples/sec
T = 1/Fs;  % time resolution, i.e., sampling interval in time domain
total_time = 1; % in us

% !!! Sampling in time
t_axis = (0:T*0.1:total_time);  % time axis
x = 0.1*cos(2*pi*F0*t_axis);  % sampled cosine/discrete time sinusoid ,time domain
Npoint = length(x);   % number of points in sampled cosine

figure
```

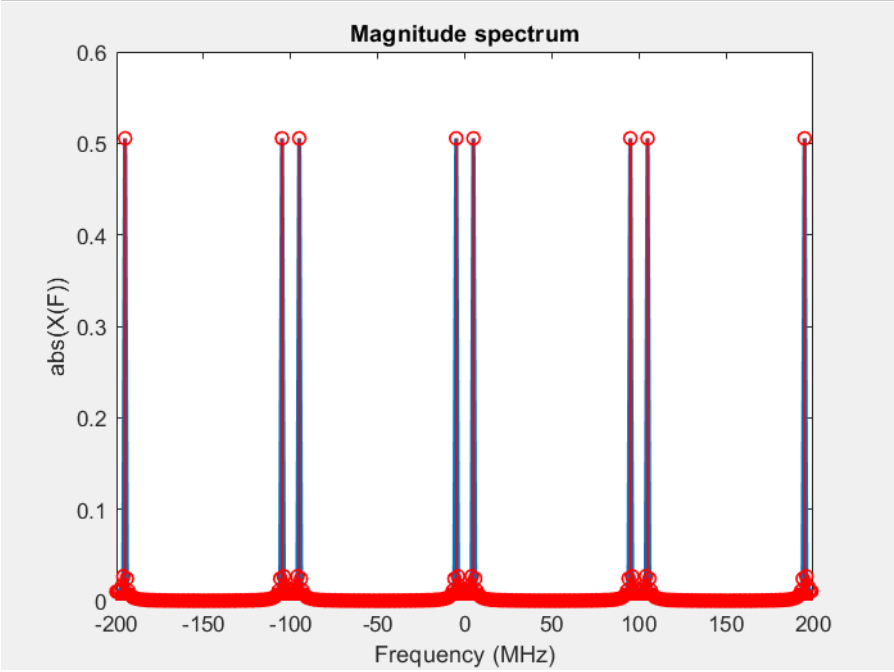
4. In the provided sample codes, the magnitude spectrum is computed and observed in the frequency range [–Fs/2 Fs/2] where Fs is the sampling rate used to sample the CT cosine signal. Please modify the codes so that the magnitude spectrum is computed (or observed) in the frequency range [–Fs Fs] and [–2Fs 2Fs], respectively. Please tell the change in the magnitude spectrum as the frequency range changes, and elaborate what you observe. Then try to show the magnitude spectrum with the normalized frequency axis and elaborate what you observe. Remember to explain why.

```
%% ---------- Fourier transform - Analysis ---------
% !!! Sampling in frequency
dF = Fs/Npoint; % frequency resolution,
%f_axis = (0:1:(Npoint-1))*dF;   % frequ
f_axis = ((1:1:Npoint*2)-(Npoint+1))*dF;
X = zeros(1,length(f_axis)); % spectrum
```



([-Fs Fs])

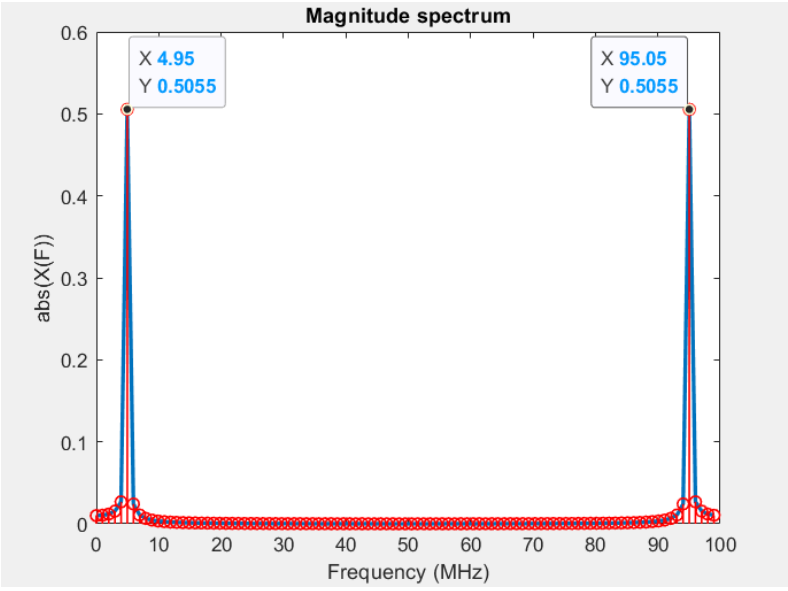
```
dF = Fs/Npoint; % frequency resolution, i.e., sam
%f_axis = (0:1:(Npoint-1))*dF;   % frequency axis
f_axis = ((1:1:Npoint*4)-(Npoint*2+1))*dF; % frec
X = zeros(1,length(f_axis)); % spectrum
```



([-2Fs 2Fs])

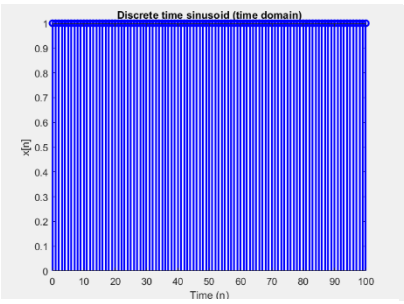
Changing f_axis affects the width of the magnitude spectrum, also showing the magnitude is periodic. The original spectrum repeats as the width increases with the period same as the sampling rate. Below is [0 Fs]:

```
dF = Fs/Npoint; % frequency resolution,
f_axis = (0:1:(Npoint-1))*dF;   % frequ
%f_axis = ((1:1:Npoint*4)-(Npoint*2+1))
X = zeros(1,length(f_axis)); % spectrum
```

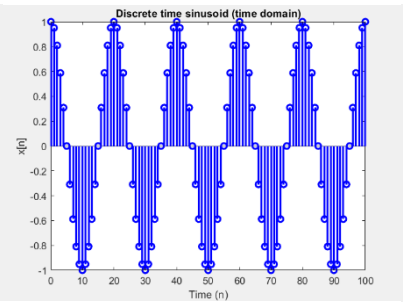


5. Change the value of **F0** in the sample codes to 0, 5, 25, 50, 75, 95, and 100MHz, respectively, and plot the discrete-time sinusoid x[n] with different **F0** as a function of time n. Please tell the change in the rate of oscillation of the discrete-time sinusoid as **F0** changes, and elaborate what you observe in time domain as well as in frequency domain. Remember to explain why. In your elaboration, you can try to convert **F0** to normalized frequency, and explain from the point of view of normalized frequency.

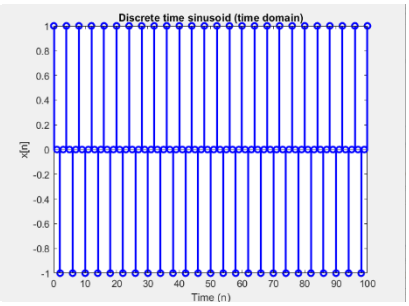
F0 = 0



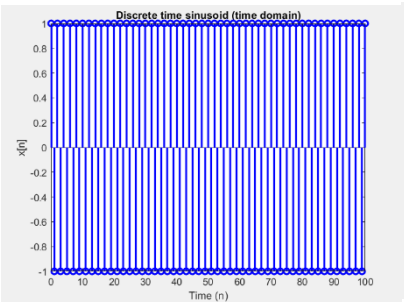
F0 = 5



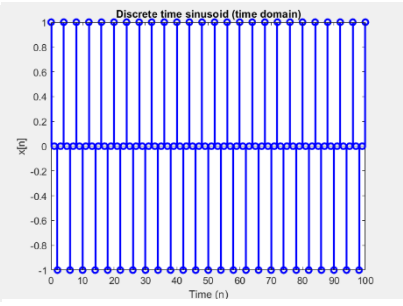
F0 =25



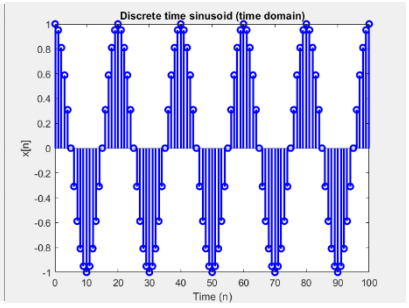
F0 = 50



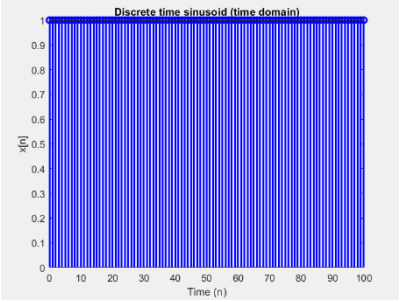
F0 = 75



F0 = 95



F0 = 100



As the frequency increases, the sinusoid becomes more intense. And when it is larger than 50, it slowly returns back, meaning that 50 has reached the max in the discrete form. That is, for frequency 75, 95, and 100, the wave devolves into 25, 5, and 0 respectively.

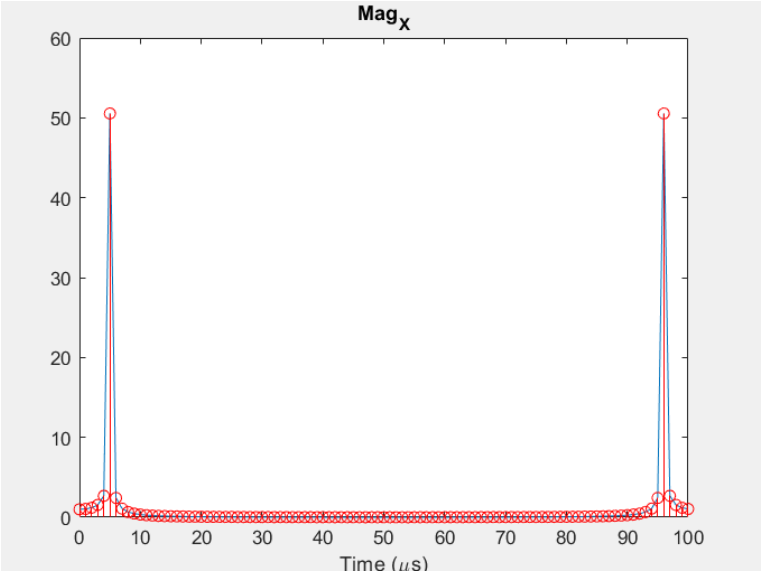
It acts like $\cos(2\pi\frac{f0}{fs}t)$ before, and acts like $\cos(2\pi(\frac{f0}{fs}-\frac{1}{2})t)$ after half of the sampling rate.

6. From problems 2, 3, 4, and 5, what is the working frequency range in the continuous-time domain for the provided CTFT codes? Justify your answer. The working frequency

range is the frequency range where the CTFT codes can truly show the representative CTFT spectrum of the corresponding CT signal.

When the sampling rate is larger than twice the original frequency, the frequency will be the same; when original frequency is between half and whole of the sampling rate, the frequency remains at most time; when original frequency is larger, it would be the same as itself minuses 2nπ.

7. Perform Fourier analysis using MATLAB built function fft(), i.e., **X**= fft(**x**) (see the provided sample codes). Plot the magnitude spectrum obtained by fft() and tell the differences between fft() spectrum and that obtained by using the provided sample codes. Once you know the differences, later on, you can use fft() to help you perform Fourier analysis instead of using our slow CTFT codes.



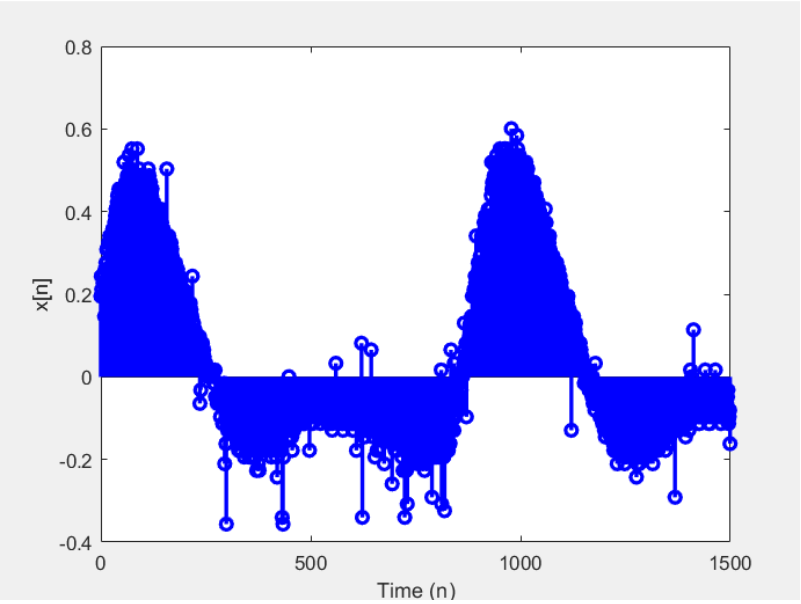
Code:

```
X = fft(x); % spectrum of sampled cosine, frequency do
f_axis_forFFT = ((0:1:(Npoint-1))); %frequency axis fo
Mag_X = abs(X);

figure
plot(f_axis_forFFT, Mag_X);
hold
stem(f_axis_forFFT, Mag_X,'r');
xlabel('Time (\mus)');
title('Mag_X');
```

$$\sum_{n=0}^{N-1} x(nT)e^{-j\frac{2\pi}{N}knFsT}T \ \ (CTFT) \qquad v.s. \qquad \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}kn} \ \ (fft)$$

Part2

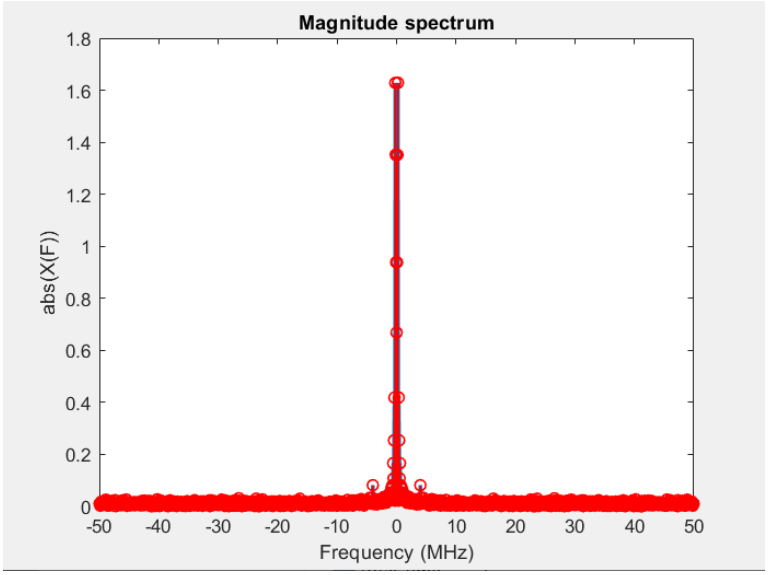
1. Given a MATLAB data file –PPG.mat where a raw PPG signal and Fs (in Hz) used to acquire the PPG signal are stored, perform Fourier analysis over (a) one single-cycle PPG signal(i.e., one heart-beat cycle) and (b) the whole PPG signal, and then elaborate which ((a) or (b)) magnitude spectrum can better provide the spectral information needed for PPG front end circuit design, e.g., band width of pre-amplifier for signal amplification, frequency response of analog filter for noise reduction, and proper ADC selection (sampling rate). (Note that the PPG signal was acquired by a digital oscilloscope with sufficiently high sampling rate, and you can use fft() for Fourier analysis instead of my slow CTFT codes if you know what fft() does for you)
   (a)one cycle



code:

```
load PPG % PPG: PPG signal, Fs: sampling rate in Hz

Fs = 100; % sampling rate/sampling frequency, in MHz or Msamples/sec
T = 1/Fs; % time resolution, sampling interval in time domain
t_axis = (1:T:1500*T); % time axis
%x = cos(2*pi*F0*t_axis); % sampled cosine/discrete time sinusoid ,time domain
x= PPG(1:1401);
Npoint = length(x); % number of points in sampled cosine

figure
plot(0:1:Npoint-1, x, 'linewidth', 2);
hold
stem(0:1:Npoint-1, x,'b', 'linewidth', 2);
xlabel('Time (n)')
ylabel('x[n]');
figure
```

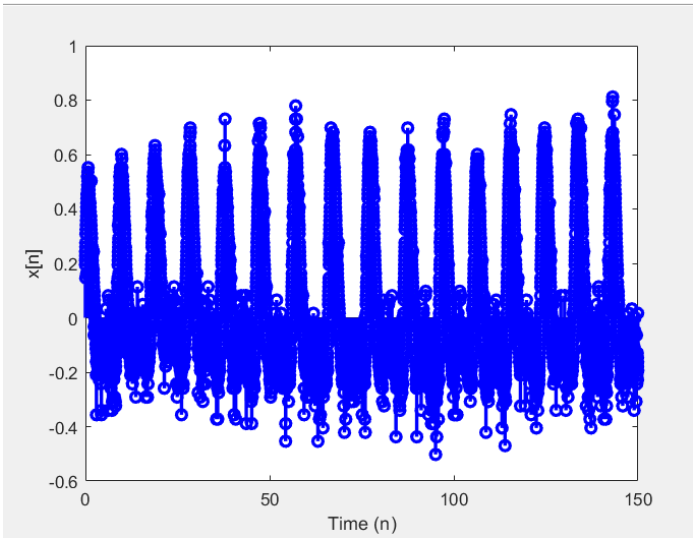
magnitude spectrum:



code:

```
f_axis = ((1:1:Npoint)-(Npoint+1)/2)*dF; % frequency axis (from -Fs/2 to Fs/2 or equivale
X = zeros(1,length(f_axis)); % spectrum

% 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
   X(iFreq)
end

mag_X = abs(X);
```

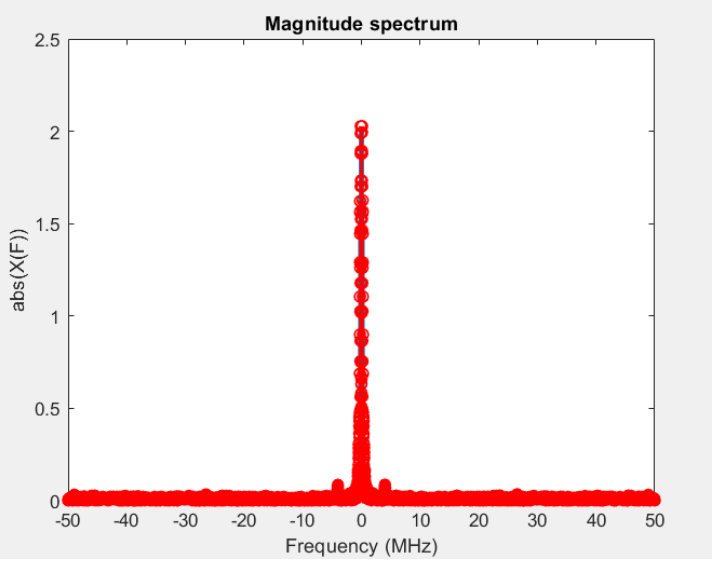
(b)whole signal



Code:

```
dF = Fs/Npoint; % frequency resolution, i.e., sampling interval in frequency domain
%f_axis = (0:1:(Npoint-1))*dF;   % frequency axis (from 0 to Fs or equivalently from 0 t
f_axis = ((1:1:Npoint)-(Npoint+1)/2)*dF; % frequency axis (from -Fs/2 to Fs/2 or equival
X = zeros(1,length(f_axis)); % spectrum
% 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
   X(iFreq)
end
mag_X = abs(X);
figure
plot(f_axis, mag_X,'linewidth',2);
hold
stem(f_axis, mag_X, 'r', 'linewidth',1)
xlabel('Frequency (MHz)');
```

Magnitude spectrum:

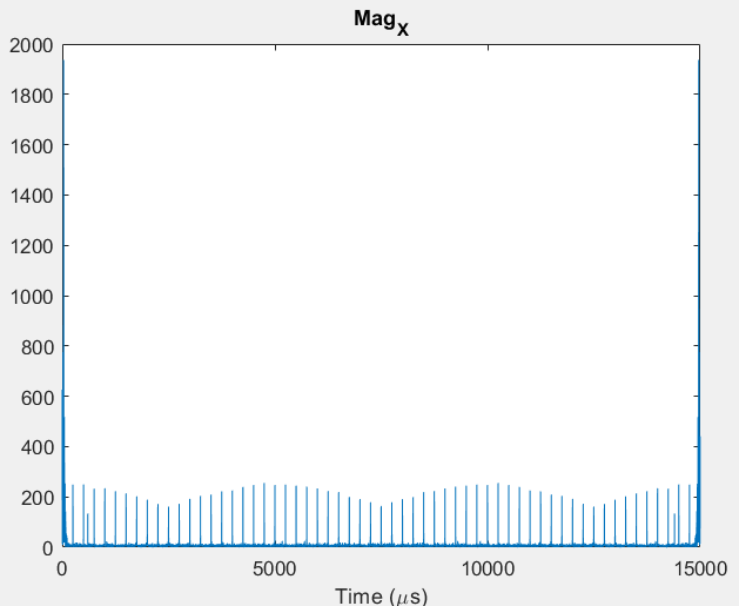


Code:

```
load PPG % PPG: PPG signal, Fs: sampling rate in Hz

Fs = 100; % sampling rate/sampling frequency, in MHz or Msamples/sec
T = 1/Fs; % time resolution, sampling interval in time domain
t_axis = (1:T:14998*T); % time axis
%x = cos(2*pi*F0*t_axis); % sampled cosine/discrete time sinusoid ,time domain
x= PPG;
Npoint = length(x); % number of points in sampled cosine

figure
plot((0:length(PPG)-1)/Fs, x, 'linewidth', 2);
hold
stem((0:length(PPG)-1)/Fs, x,'b', 'linewidth', 2);
xlabel('Time (n)')
ylabel('x[n]');
figure
```

Ans. (a) provides the spectral information needed for PPG front end circuit design because its signal is disturbed less.

2. Following Part 2.1, once you see the magnitude spectrum, you will find that the acquired PPG signal suffers 60-Hz power line noise. Based on what you have learned from our lectures so far, design an LTI filter/system to remove the 60-Hz power line noise, implement the filter in time domain or frequency domain, and verify whether or not your filter works by comparing the unfiltered and filtered PPG signals in time domain and in frequency domain (i.e., check the spectra). Remember to elaborate your filter design and implementation.

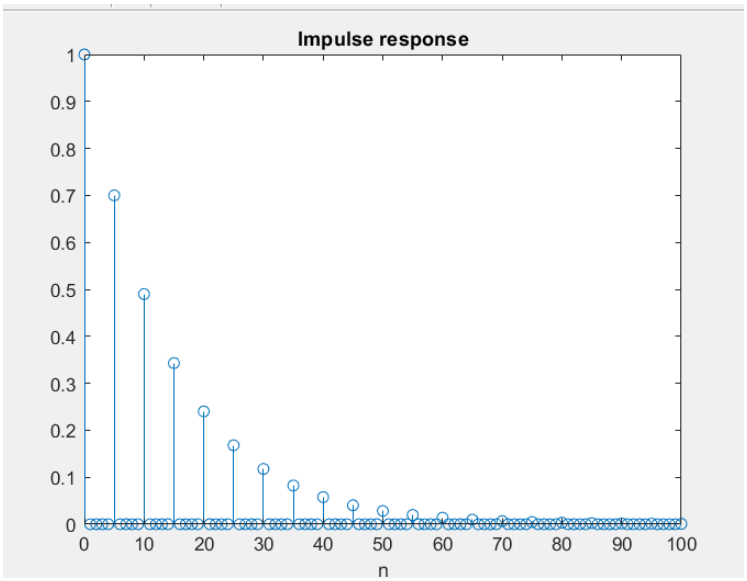


Code:

```
UnitImpulse = [1 zeros(1,59)];
z = UnitImpulse;
h = [repmat(z,1,249) zeros(1,57)]; % impulse response of the noise remover
g = x + fliplr(h);
PPG_NoiseSuppressed = PPG + fliplr(g);

figure
plot((0:length(PPG)-1), PPG_NoiseSuppressed);
xlabel('Time (n)')
ylabel('x[n]');

PPG_NoiseSuppressedfft = fft(PPG_NoiseSuppressed);
f_axis_forFFT = ((0:1:(Npoint-1))); %frequency axis for fft(x), from 0 to Fs
Mag_X = abs(PPG_NoiseSuppressedfft);

figure
plot(f_axis_forFFT, Mag_X);
xlabel('Time (\mus)');
title('Mag_X');
```

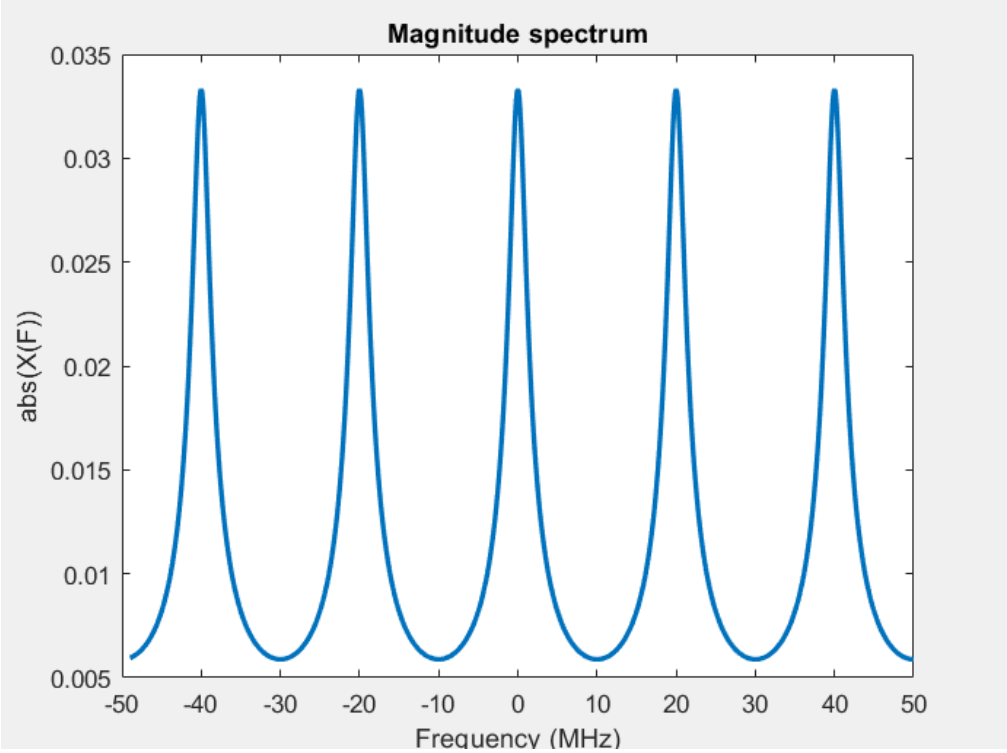
Using fliplr()

3. Derive the frequency response of the comb reverberator in our Topic 2 lecture (see slide 48, in Topic2_LTISystems_Part2 _HandWriting0415_2020.pdf), and plot the magnitude response (i.e., magnitude of the frequency response) using MATLAB (by sampling in frequency domain). Please use the same values of *a* and *D* in Part1 (d) of your computer homework 2 for the frequency response plot. Then based on the implemented CTFT codes (you can use fft() if you know what fft() does for you), **y(input)**, and **ye(output)** in the provided **EchoGen.m**, please compute the magnitude response using the FT convolution property and compare with your derivation. Explain why it is called as "Comb" reverberator".



code:

```
D = 5;
a = 0.7;
UnitImpulse = [1 zeros(1,100)]; % creat unit impulse, starting from n=0;
x = UnitImpulse;
y = filter(1, [1 zeros(1, D-1) -a], x);
n = 0:(length(y)-1); % check length of y[n]

figure
stem(n, y); % is it the same as that in your derivation?
xlabel('n')
title('Impulse response')
```

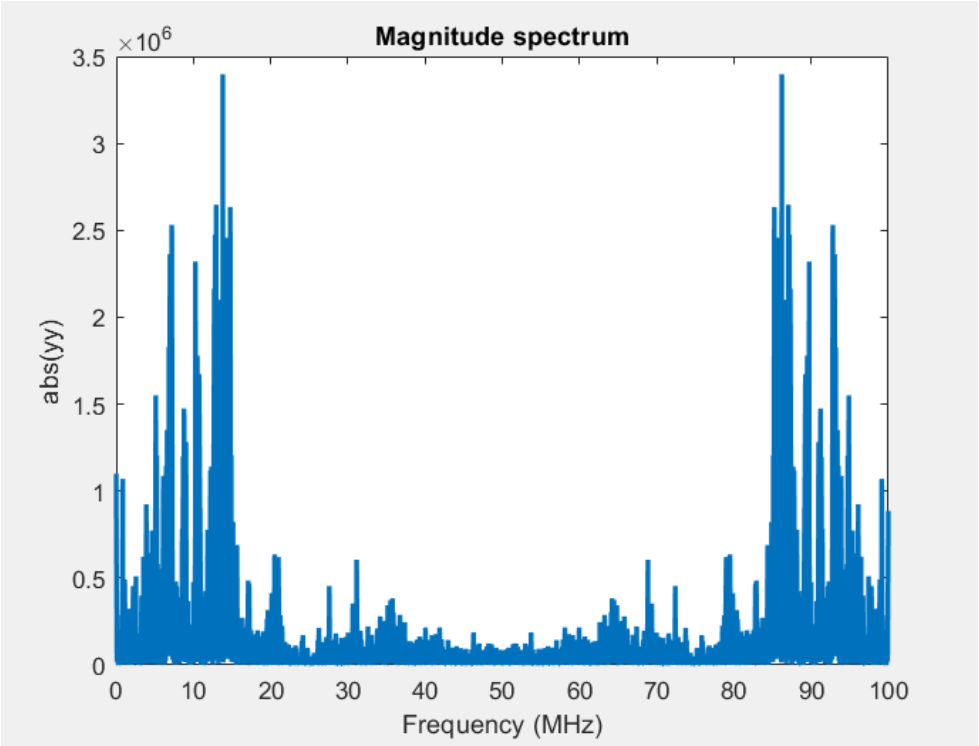


code:

```
F_axis = (1:0.1:100)-50;
t_axis = (0:0.01:1);
X = zeros(1,length(F_axis));
x = y;
for iFreq = 1:length(F_axis)
    for iTime = 1:length(t_axis)
        X(iFreq) = X(iFreq) + x(iTime)*exp(-sqrt(-1)*2*pi*F_axis(iFreq)*t_axis(iTime))*0.01;
    end
end
mag_X = abs(X); % magnitude

figure
plot(F_axis, mag_X,'linewidth',2);
xlabel('Frequency (MHz)');
ylabel('abs(X(F))')
title('Magnitude spectrum')
```

It is called the comb “Comb reverberator” because of the plot in frequency domain symbols a comb with teeth.

EchoGen:



code:

```
Npoint = length(ye);
Fs = 100;
df = Fs/Npoint; % frequency resolution
f_axis = (0:1:(Npoint-1))*df;

yy = fft(ye);
mag_yy = abs(yy);

figure
plot(f_axis, mag_yy,'linewidth',2);
xlabel('Frequency (MHz)');
ylabel('abs(yy)')
title('Magnitude spectrum')
```