Signals and Systems

Term Project

2014312436

School of Mechanical Engineering

Joung Byeong In

2019.05.25

Table

1. Theories and objectives

1) Fourier transform

The fourier transform which was introduced by Jean Baptiste Joseph Fourier[1] can transform from time-domain function to frequency-domain function. It can by following equation.

$$\mathrm{x(t)} = \sum_{k=-\infty}^{\infty} X[k]e^{jk\omega_0 t} \leftarrow \mathrm{FT} \rightarrow \mathrm{X(j\omega)} = 2\pi \sum_{k=-\infty}^{\infty} X[k]\delta(\omega - k\omega_0) \text{ for continuous function} \cdots (1)$$

$$\text{where } \mathrm{X[k]} = \frac{1}{T}\int_0^T x(t)e^{-jk\omega_0 t}dt, \text{ T is period and } \omega_0 = \frac{2\pi}{T}$$

$$\mathrm{x[n]} = \sum_{k=0}^{N-1} X[k]e^{jk\Omega_0 n} \leftarrow DTFT \rightarrow X(e^{j\Omega}) = 2\pi \sum_{k=-\infty}^{\infty} X[k]\delta(\Omega - k\Omega_0) \text{ for discrete function} \cdots (2)$$

$$\text{where N is period and } \Omega_0 = \frac{2\pi}{N}$$

2) Sampling

In signal system, sampling means converting from continuous to discrete. It can be easily by using following equation.

$$\mathrm{x_\delta(t)} = x(t)\sum_{n=-\infty}^{\infty} \delta(t - nT_s) \text{ where } \mathrm{T_s} \text{ is interval} \cdots (3)$$

3) Sampling theorem

If sampling frequency of arbitrary function x(t) is greater than or equal to highest frequency of it two times, there are only one x(t) exists determined by that sampling frequency. This is called the sampling theorem. When sampling frequency is equal to two times of highest frequency, that sampling frequency is called by nyquist frequency.[2]

4) Convolution

Convolution is mathematical operation which make new function by using two functions. After let one of them be fixed and the other one be reversed, moving reversed function and calculate overlapping area of both function. It can be expressed like this :

$$y(t) = f(t) * g(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \quad \cdots (4)$$

However, in frequency domain, by using fourier transform, (4) turn to be :

$$Y(j\omega) = F(j\omega) \times G(j\omega)$$

where $Y(j\omega)$, $F(j\omega)$ and $G(j\omega)$ are fourier transform of $y(t), f(t)$ and $g(t)$.

5) Low pass filter(LPF)

Low pass filter is passing only low frequency of input signal.

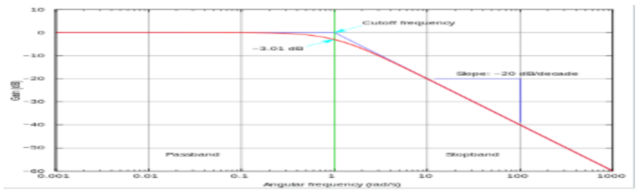


Figure 1. Low pass filter in frequency domain[3]

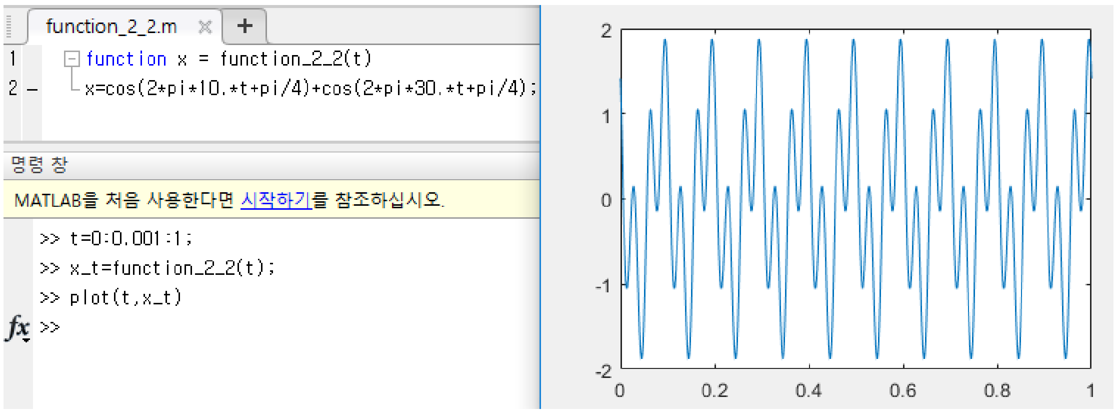
Figure 1 shows low pass filter in frequency domain.

6) Objectives

I learned convolutions, fourier transform and sampling which are frequently used in real life(especially telephone, radio and other communications) from signals and system class. Conducting them by using Matlab software, enhance understanding of discrete and continuous signals.

2. Generation of Signals

1) Generate a continuous-time signal $x(t) = \cos\left(2\pi 10t + \frac{\pi}{4}\right) + \cos(2\,\pi 30t + \frac{\pi}{4})$.


function_2_2.m

```
function x = function_2_2(t)
x=cos(2*pi*10.*t+pi/4)+cos(2*pi*30.*t+pi/4);
```

명령 창

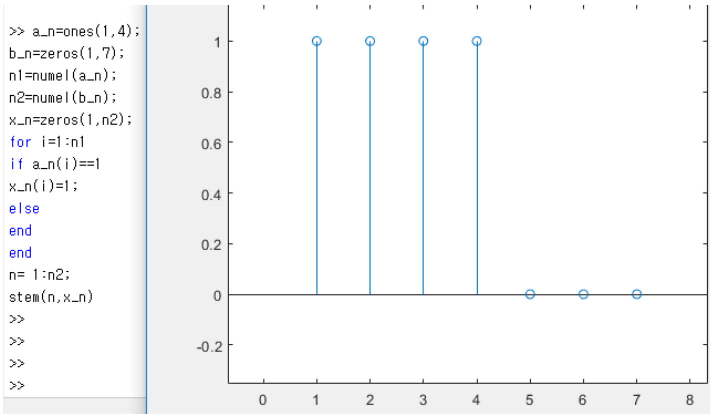
MATLAB를 처음 사용한다면 시작하기를 참조하십시오.

```
>> t=0:0.001:1;
>> x_t=function_2_2(t);
>> plot(t,x_t)
>>
```

Figure 2. Continuous-time signal of x(t)

Figure 2 shows continuous-time signal of given x(t) and matlab code. I made 'function_2_2.m' file to make given function and plot x(t).

2) Generate a discrete-time signal $x[n] = \left(\frac{1}{4}\right)(u[n] - u[n-4])$.



```
>> a_n=ones(1,4);
b_n=zeros(1,7);
n1=numel(a_n);
n2=numel(b_n);
x_n=zeros(1,n2);
for i=1:n1
if a_n(i)==1
x_n(i)=1;
else
end
end
n= 1:n2;
stem(n,x_n)
>>
>>
>>
>>
```

Figure 3. Discrete-time signal of x[n]

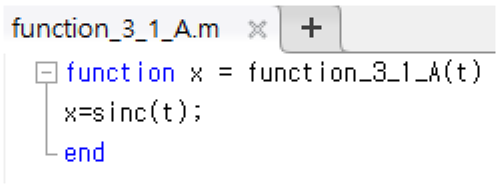
Figure 3 shows discrete-time signal of given x[n]. I could make it by using 'for' command and 'if' command.

3. Continuous-Time Non-periodic Signal

1) Convolution and Fourier Transform(TF)

(a) Generate a signal(sinc function) for $-10 \le t \le 10$.

$$x_1(t) = sinc(t) = \frac{\sin(\pi t)}{\pi t}.$$



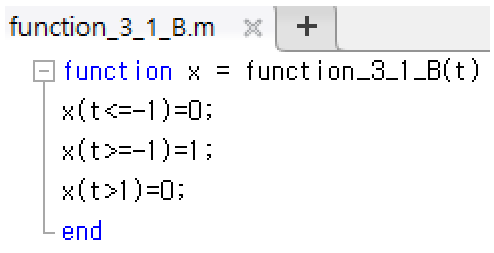
```
function_3_1_A.m
function x = function_3_1_A(t)
x=sinc(t);
end
```

Figure 4. Sinc function

Figure 4 shows sinc function m-file code.

(b) Generate a function $h_1(t)$ for $-10 \le t \le 10$.

$$h_1(t) = \begin{cases} 1, & -1 \le t \le 1 \\ 0, & otherwise \end{cases}$$



```
function_3_1_B.m
function x = function_3_1_B(t)
x(t<=-1)=0;
x(t>=-1)=1;
x(t>1)=0;
end
```

Figure 5. $h_1(t)$

Figure 5 shows $h_1(t)$ function m-file code.

(c) Using the conv function in Matlab, compute the convolution $y_1(t) = x_1(t) * h_1(t)$.

```
t=-10:0.01:10;
x_1=function_3_1_A(t);
h_1=function_3_1_B(t);
y_1=conv(x_1,h_1,'same')/100;
```

Figure 6. Convolution of $x_1(t)$ and $y_1(t)$

(d) Using the fft function in Matlab, find the FT of $x_1(t)$ and $h_1(t)$.

```
>> X_1=fft(x_1);
>> H_1=fft(h_1);
```

Figure 7. FFT of $x_1(t)$ and $y_1(t)$

Figure 7 shows fourier transform of $x_1(t)$ and $h_1(t)$.

(e) Multiply the FT of $x_1(t)$ by that of $h_1(t)$, that is $Y_1(j\omega) = X_1(j\omega)H_1(j\omega)$.

```
>> Y_1=X_1.*H_1;
```

Figure 8. Multiplication

Figure 8 shows multiplication of $x_1(t)$ and $h_1(t)$. This is simple.

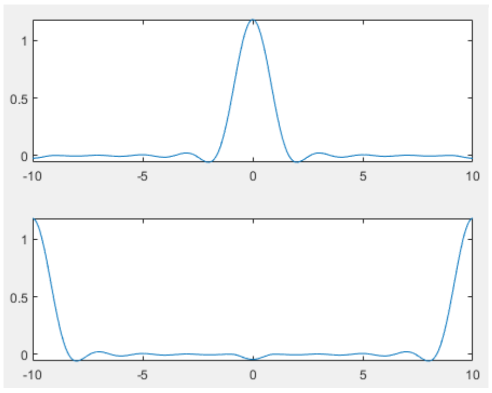
(f) Using ifft function in Matlab, find the inverse FT(IFT) of $Y_1(j\omega)$ to find $y_1(t)$.

```
>> y_1_IFT=ifft(Y_1);
```

Figure 9. IFT of $Y_1(j\omega)$

Figure 9 shows inverse fourier transform in matlab.

(g) Compare the signal $y_1(t)$ found from the convolution in time domain and $y_1(t)$ found from the multiplication in frequency domain.



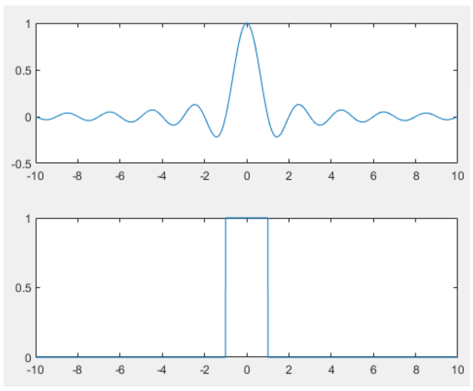
```
t=-10:0.01:10;
x_1=function_3_1_A(t);
h_1=function_3_1_B(t);
y_1=conv(x_1,h_1,'same')/100;
subplot(2,1,1)
plot(t,y_1)
X_1=fft(x_1);
H_1=fft(h_1);
y_1_ift=ifft(X_1.*H_1);
subplot(2,1,2)
plot(t,y_1_ift/100)
```

Figure 10. Plot $y_1(t)$ found from different method

Figure 10 shows two $y_1(t)$ functions. Upper plot shows function came from 'conv' command and the other one shows function came from 'ifft' command. I can observe they are different and one of them should be shifted. The code is named as 'function_3_1_G.m'.

2) Plot

(a) Plot $x_1(t)$ and $h_1(t)$ using the plot function in Matlab for $-10 \leq t \leq 10$.

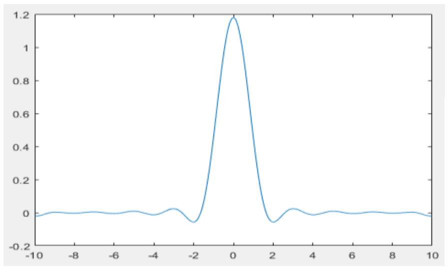


```
t=-10:0.01:10;
x_1=function_3_1_A(t);
h_1=function_3_1_B(t);
subplot(2,1,1)
plot(t,x_1)
subplot(2,1,2)
plot(t,h_1)
```

Figure 11. $x_1(t)$ **and** $h_1(t)$

Figure 11 shows plot of $x_1(t)$ and $h_1(t)$ and code of Matlab to plot it.. Upper graph shows $x_1(t)$, sinc function and Below one shows $h_1(t)$.

(b) Plot $y_1(t)$ from the convolution in time domain using the plot function in Matlab for $-10 \leq t \leq 10$.



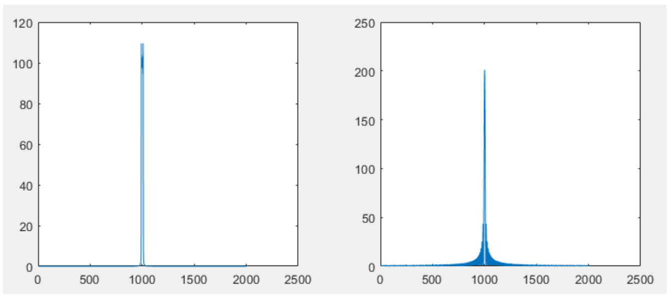
```
>> t=-10:0.01:10;
x_1=function_3_1_A(t);
h_1=function_3_1_B(t);
>> y_1=conv(x_1,h_1,'same')/100;
>> plot(t,y_1)
```

Figure 12. $y_1(t)$

Figure 12 shows plot fo $y_1(t)$ and Matlab code to plot it.

(c) Plot FTs of $x_1(t)$ and $h_1(t)$ using the plot.

(d)

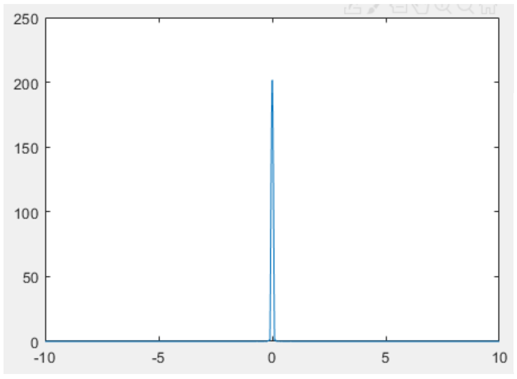


```
t=-10:0.01:10;
x_1=function_3_1_A(t);
h_1=function_3_1_B(t);
X_1=fft(x_1);
H_1=fft(h_1);
subplot(1,2,1)
plot(abs(fftshift(X_1)))
subplot(1,2,2)
plot(abs(fftshift(H_1)))
```

Figure 13. FTs of $x_1(t)$ **and** $h_1(t)$

Figure 13 shows FTs of $x_1(t)$ and $h_1(t)$ and Matlab code for plot. To plot and arrange complex number, I used 'abs' command and 'fftshift' command.

(e) Plot $Y_1(j\omega) = X_1(j\omega)H_1(j\omega)$ using the plot.

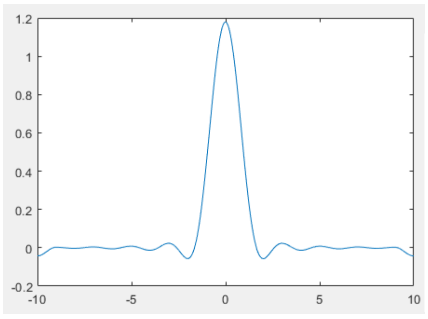


```
t=-10:0.01:10;
x_1=function_3_1_A(t);
h_1=function_3_1_B(t);
X_1=fft(x_1);
H_1=fft(h_1);
Y_1=X_1.*H_1;
plot(t,fftshift(abs(Y_1))/100)
```

Figure 14. $\mathbf{Y_1(j\omega)}$

Figure 14 shows graph of $Y_1(j\omega)$ and code to plot it. To find $Y_1(j\omega)$, use '.*' command due to multiply component of matrices, $X_1(j\omega)$ and $H_1(j\omega)$.

(f) Plot $y_1(t)$ obtained from the multiplication $Y_1(j\omega) = X_1(j\omega)H_1(j\omega)$ in frequency domain and IFT using the plot function in Matlab for $-10 \leq t \leq 10$.

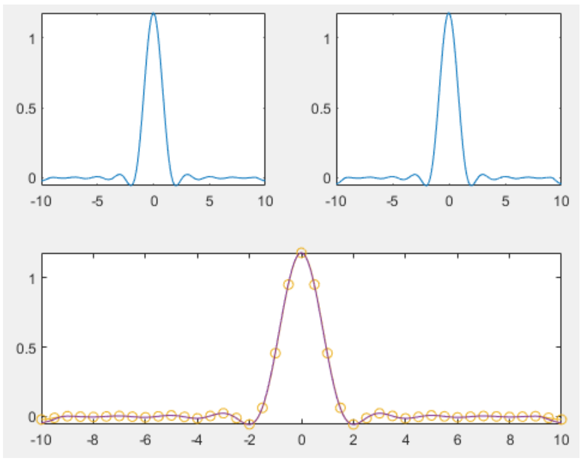


```
t=-10:0.01:10;
x_1=function_3_1_A(t);
h_1=function_3_1_B(t);
X_1=fft(x_1);
H_1=fft(h_1);
Y_1=X_1.*H_1;
plot(t,fftshift(ifft(Y_1))/100)
```

Figure 15. $\mathbf{y_1(t)}$

Figure 15 shows $y_1(t)$ obatained from the multiplication and Matlab code to plot it.

(g) Plot $y_1(t)$ in (b) and that in (e) together in a graph using subplot.



```
t=-10:0.01:10;
x_1=function_3_1_A(t);
h_1=function_3_1_B(t);
y_1=conv(x_1,h_1,'same')/100;
X_1=fft(x_1);
H_1=fft(h_1);
y_1_ift=ifft(X_1.*H_1)/100;
subplot(2,2,1);
plot(t,y_1);
subplot(2,2,2);
plot(t,fftshift(y_1_ift));
subplot(2,2,[3, 4]);
plot(t,y_1,'-o','MarkerIndices',1:50:length(y_1));
hold on
plot(t,fftshift(y_1_ift));
```

Figure 16. Plot two types of $y_1(t)$

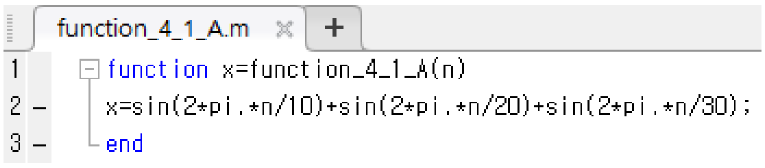
Figure 16 shows two small graph in up and combined graphs in down and Matlab code. Both results ($y_1(t)$) are looked very similar so I plotted one more graph below of them with different expression method to compare easily. At below subplot, graph shown as line is $y_1(t)$ found from inverse fourier transform and graph which is shown as dot is $y_1(t)$ found from 'conv' command. The dots are marked as one for each of the 5 components. I can find they are same.

4. Discrete-Time Periodic Signal

1) Convolution and Discrete-Time Fourier Series(DTFS)

(a) Generate a periodic signal $x_2[n]$ with the fundamental period N. Find the fundamental frequency $\Omega_0 = 2\pi/N$ with the fundamental period N.

$$x_2[n] = \sin\left(\frac{2\pi n}{10}\right) + \sin\left(\frac{2\pi n}{20}\right) + \sin\left(\frac{2\pi n}{30}\right),\ \text{for}\ 0 \le n \le N-1$$



```
function x=function_4_1_A(n)
x=sin(2*pi.*n/10)+sin(2*pi.*n/20)+sin(2*pi.*n/30);
end
```

Figure 17. $x_2[n]$

Figure 17 shows matlab code to make $x_2[n]$. Using 'stem' command will be helpful to find fundamental period of $x_2[n]$.

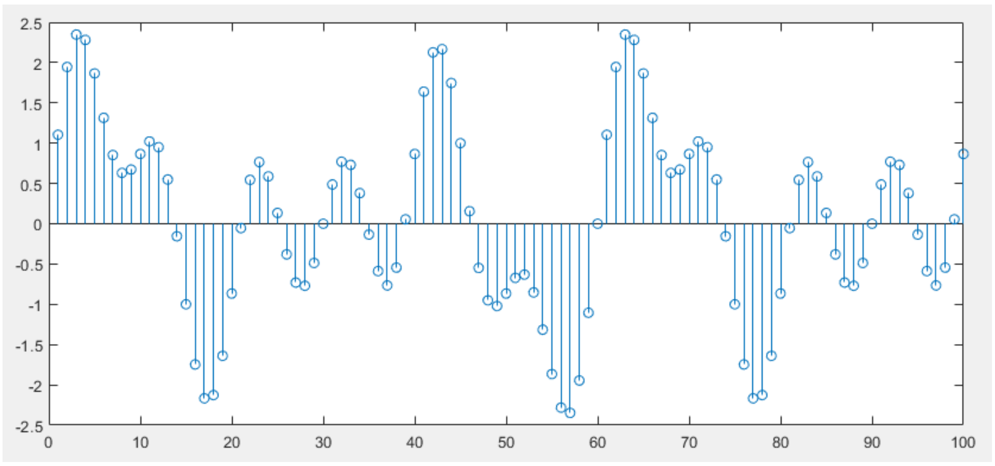
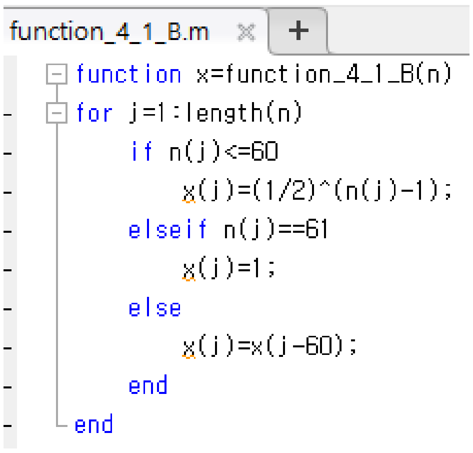


Figure 18. Plot of $x_2[n]$

Figure 18 shows $x_2[n]$. I could find fundamental period N from the figure 17, N=60.

(b) Generate a periodic signal $h_2[n]$ with the fundamental period N.

$$h_2[n] = \left(\frac{1}{2}\right)^n, \text{ for } 0 \le n \le N-1$$



```
function x=function_4_1_B(n)
for j=1:length(n)
    if n(j)<=60
        x(j)=(1/2)^(n(j)-1);
    elseif n(j)==61
        x(j)=1;
    else
        x(j)=x(j-60);
    end
end
```

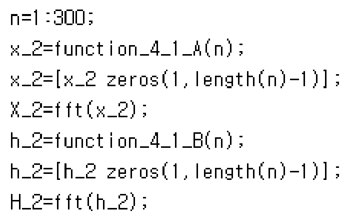
Figure 19. $h_2[n]$

Figure 19 shows $h_2[n]$.

(c) Using the conv function in Matlab, compute the convolution $y_2[n] = x_2[n] * h_2[n]$.

I could compute the convolution like this : 'y_2=conv(x_2,h_2);' in matlab workspace.

(d) Using the fft function in Matlab, find the DTFS $X_2[k]$ of $x_2[n]$ and $H_2[k]$ of $h_2[n]$.



```
n=1:300;
x_2=function_4_1_A(n);
x_2=[x_2 zeros(1,length(n)-1)];
X_2=fft(x_2);
h_2=function_4_1_B(n);
h_2=[h_2 zeros(1,length(n)-1)];
H_2=fft(h_2);
```

Figure 20. DTFS

Figure 20 shows DTFS of $x_2[n]$ and $y_2[n]$ by using 'fft' function. Before convolution, I should make two array(x_2 and h_2) long because the product of x_2 and h_2 is fourier transform of circular convolution. 'conv' command is linear convolution, so I should make x_2 and h_2 have same length with conv(x_2,h_2).

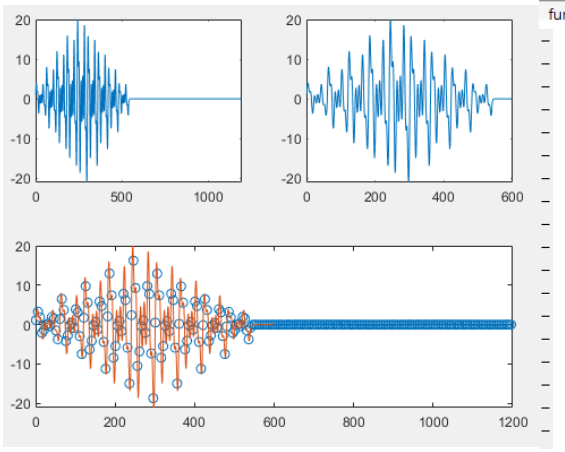
(e) Multiply the DTFSs of $x_2[n]$ and $h_2[n]$, that is $Y_2[k] = NX_2[k]H_2[k]$.

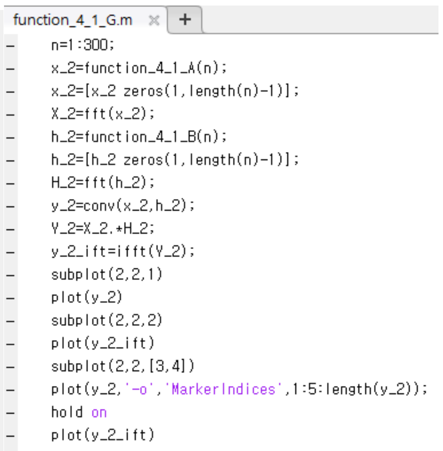
I could find $Y_2[k]$ by simple code : 'Y_2[n]=X_2[n].*H_2[n];' in matlab workspace.

(f) Using the ifft function in Matlab, find the inverse DTFS of $Y_2[k]$ to find $y_2[n]$.

Let inverse DTFS of Y_2[n] is y_2_ift, then inverse DTFS of Y_2[n] is 'y_2_ift=ifft(Y_2);' in matlab workspace.

(g) Compare the signal $y_2[n]$ found from the convolution in time domain and $y_2[n]$ found from the multiplication in frequency domain and IDTFS.

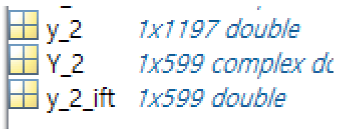



function_4_1_G.m

```
n=1:300;
x_2=function_4_1_A(n);
x_2=[x_2 zeros(1,length(n)-1)];
X_2=fft(x_2);
h_2=function_4_1_B(n);
h_2=[h_2 zeros(1,length(n)-1)];
H_2=fft(h_2);
y_2=conv(x_2,h_2);
Y_2=X_2.*H_2;
y_2_ift=ifft(Y_2);
subplot(2,2,1)
plot(y_2)
subplot(2,2,2)
plot(y_2_ift)
subplot(2,2,[3,4])
plot(y_2,'-o','MarkerIndices',1:5:length(y_2));
hold on
plot(y_2_ift)
```

Figure 21. Result of function_4_1_G.m

Figure 21 shows the result of function_4_1_G.m which show two result. The left upper figure shows y_2 which is made by 'conv' command and the right upper figure shows y_2 which is made by 'ifft' command. I could find first to 600th component of y_2 which is made by 'conv' command are same with y_2 which is made by 'ifft' command. More details will introduced at 4-2-(f).



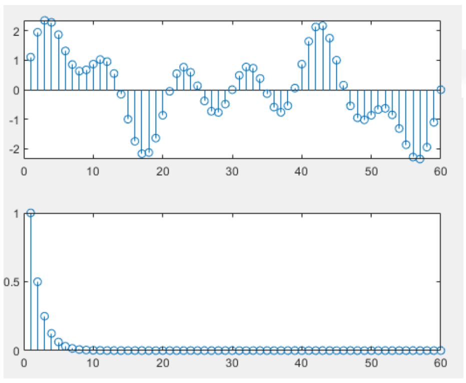
```
y_2      1x1197 double
Y_2      1x599 complex dc
y_2_ift  1x599 double
```

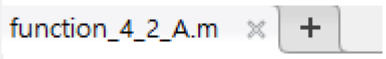
Figure 22. The number of components of y_2 and y_2_ift

Figure 22 shows the number of components of y_2 and y_2_ift. As you can see, y_2 has about two times components lather tan y_2_ift. Because of convolution, y_2 should have 2*599-1=1197 components.

2) Plot

(a) Plot $x_2[n]$ and $h_2[n]$ for $0 \le n \le N-1$.



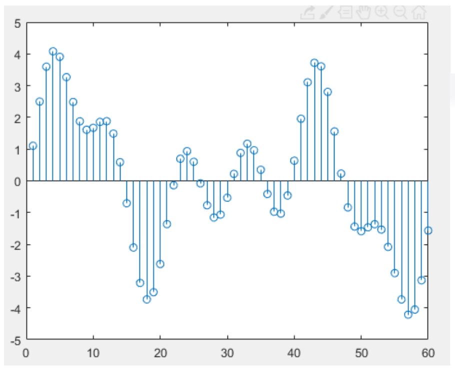


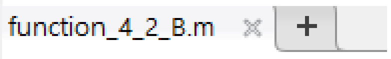
```
function_4_2_A.m
n=1:60;
x_2=function_4_1_A(n);
h_2=function_4_1_B(n);
subplot(2,1,1)
stem(x_2)
subplot(2,1,2)
stem(h_2)
```

Figure 23. $x_2[n]$ **and** $h_2[n]$

Figure 23 shows $x_2[n]$ and $h_2[n]$ for $0 \le n \le N-1$. The left figure shows $x_2[n]$ at upside and $h_2[n]$ at downside. The right figure shows code to plot $x_2[n]$ and $h_2[n]$.

(b) Plot $y_2[n]$ from the convolution in time domain for $0 \le n \le N-1$.



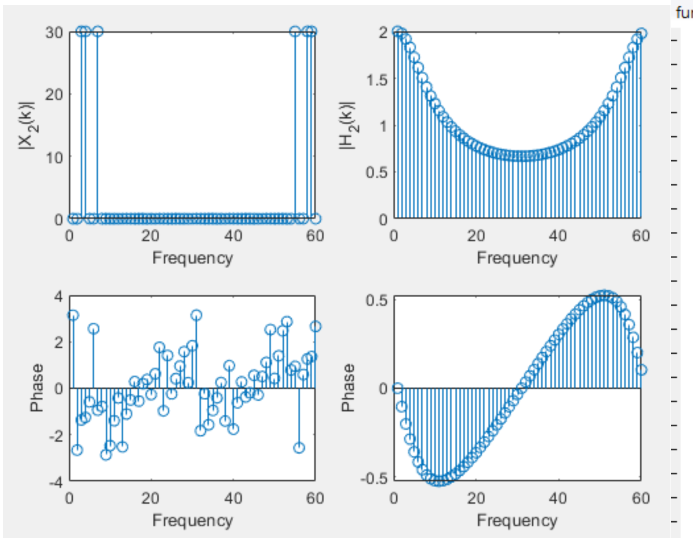


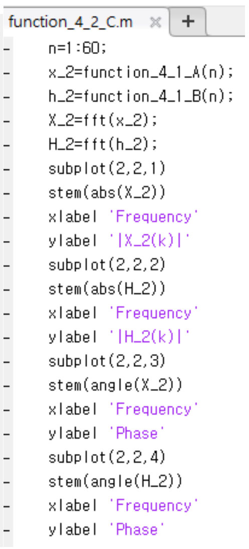
```
function_4_2_B.m
n=1:60;
x_2=function_4_1_A(n);
h_2=function_4_1_B(n);
y_2=conv(x_2,h_2);
stem(y_2)
xlim([0 60])
```

Figure 24. $y_2[n]$

Figure 24 shows the result of convolution. The left figure shows $y_2[n]$ and right figure shows code to plot $y_2[n]$.

(c) Plot DTFS of the $x_2[n]$ and $h_2[n]$.



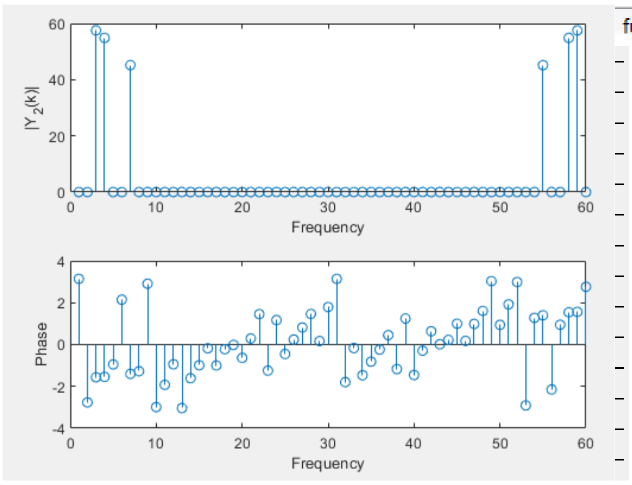


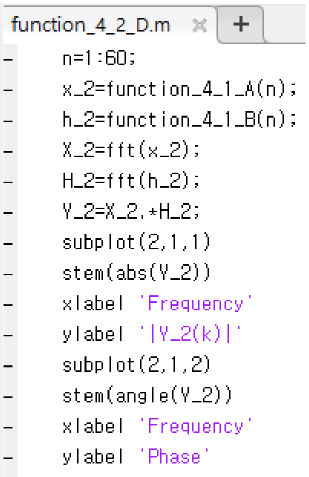
```
n=1:60;
x_2=function_4_1_A(n);
h_2=function_4_1_B(n);
X_2=fft(x_2);
H_2=fft(h_2);
subplot(2,2,1)
stem(abs(X_2))
xlabel 'Frequency'
ylabel '|X_2(k)|'
subplot(2,2,2)
stem(abs(H_2))
xlabel 'Frequency'
ylabel '|H_2(k)|'
subplot(2,2,3)
stem(angle(X_2))
xlabel 'Frequency'
ylabel 'Phase'
subplot(2,2,4)
stem(angle(H_2))
xlabel 'Frequency'
ylabel 'Phase'
```

Figure 25. DTFS of $x_2[n]$ and $h_2[n]$

Figure 25 shows DTFS of $x_2[n]$ and $h_2[n]$. the graphs in upside shows magnitude of DTFS at each frequency and the graphs in downside shows phase angle of DTFS at each frequency.

(d) Plot $Y_2[k] = NX_2[k]H_2[k]$.



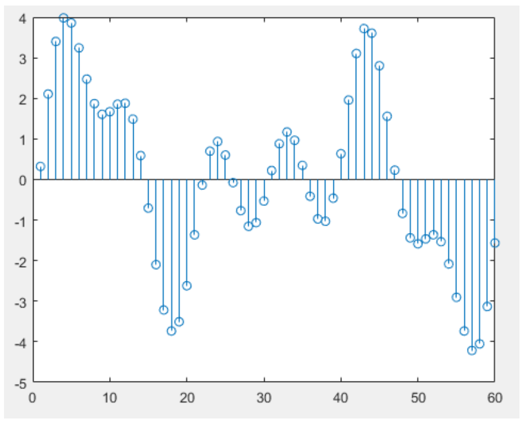


```
n=1:60;
x_2=function_4_1_A(n);
h_2=function_4_1_B(n);
X_2=fft(x_2);
H_2=fft(h_2);
Y_2=X_2.*H_2;
subplot(2,1,1)
stem(abs(Y_2))
xlabel 'Frequency'
ylabel '|Y_2(k)|'
subplot(2,1,2)
stem(angle(Y_2))
xlabel 'Frequency'
ylabel 'Phase'
```

Figure 26. $Y_2[k]$

Figure 26 shows magnitude and phase angle of $Y_2[k]$ and code to plot it. The upside graph shows magnitude of $Y_2[k]$ and the other one shows phase angle of $Y_2[k]$.

(e) Plot $y_2[n]$ obtained from the multiplication in frequency domain and IDTFS.



```
n=1:60;
x_2=function_4_1_A(n);
h_2=function_4_1_B(n);
X_2=fft(x_2);
H_2=fft(h_2);
y_2_ift=ifft(X_2.*H_2);
stem(y_2_ift)
```

Figure 27. $y_2[n]$ **obtained from IDFTS**

Figure 27 shows $y_2[n]$ obtained from IDFTS. The left figure shows $y_2[n]$ which made by 'stem' command in the right figure.

(f) Plot the signal $y_2[n]$ found from the convolution in time domain and $y_2[n]$ found from the multiplication $Y_2[k] = NX_2[k]H_2[k]$in frequency domain and IDTFS in a single graph using subplot.

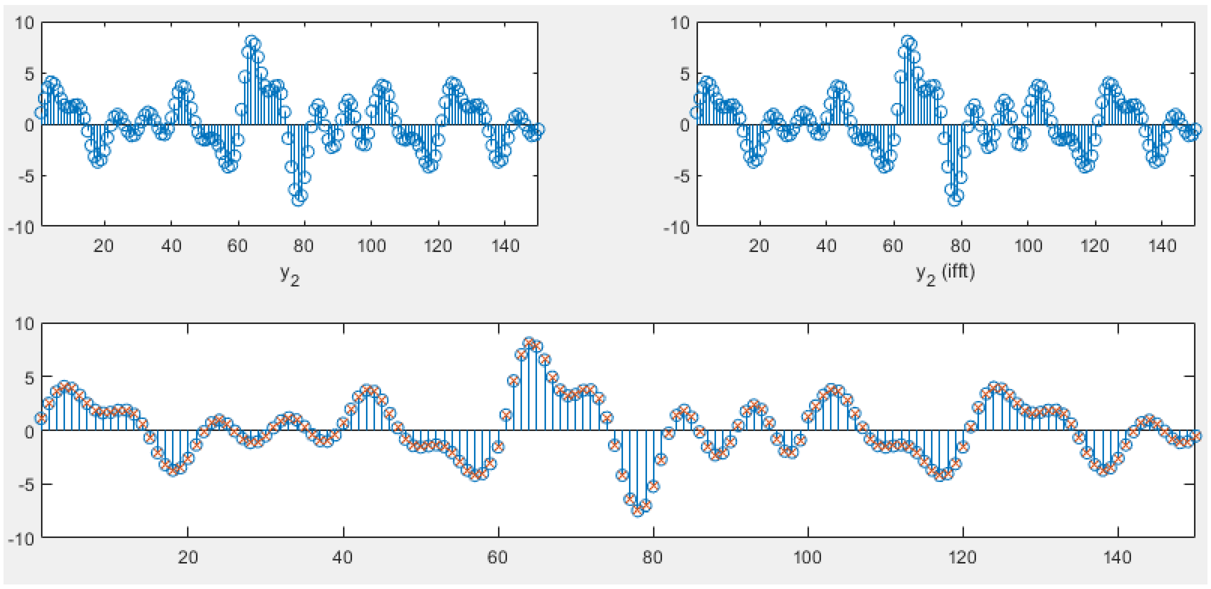


Figure 28. $y_2[n]$ **found from the convolution and** $y_2[n]$ **found from IDTFS**

Figure 28 shows $y_2[n]$ found from the convolution in time domain and $y_2[n]$ found from the IDFTS. The left upside shows what found from the 'conv' command, the right upside shows what found from the 'ifft' command. The last one shows both in same graph. Which one is shown by the red 'x' is what found from the IDFTS and the other one is what found from 'conv' function. I could identify those two graphs are exactly same.

5. Real Audio Signal

1) Generate and Process Audio Signal

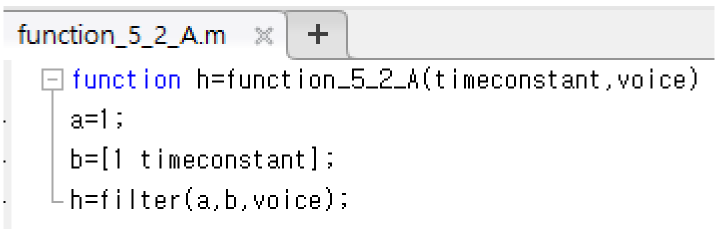
(a) Follow the steps to collect your own voice signal.

```
recObj = audiorecorder

disp('Start speaking.')
recordblocking(recObj, 5); % 5 seconds
disp('End of Recording.');

play(recObj);

x = getaudiodata(recObj);

plot(x);
```

Figure 29. Collect Voice

2) Filter the Voice Signal

(a) Create a low pass filter $h[n]$ to process your voice signal $x[n]$.



```
function h=function_5_2_A(timeconstant,voice)
a=1;
b=[1 timeconstant];
h=filter(a,b,voice);
```

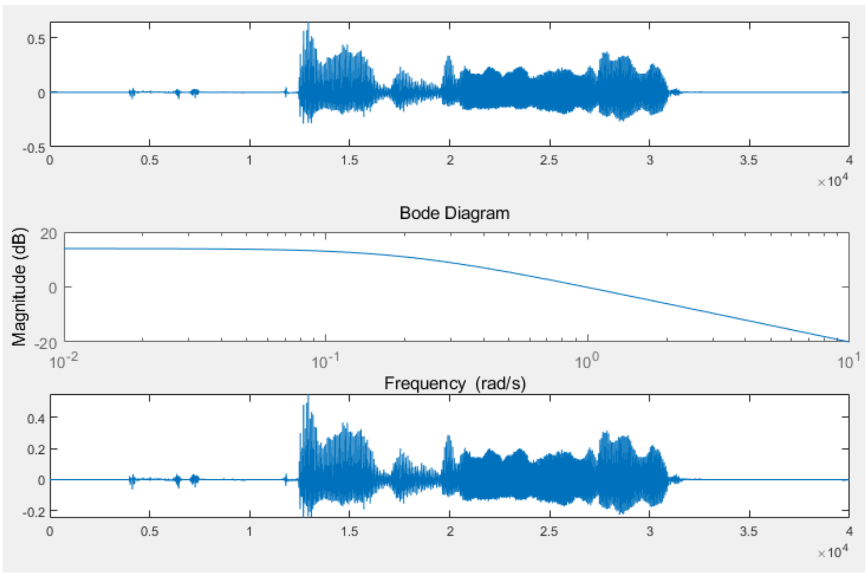
Figure 30. A Moving Average Filter

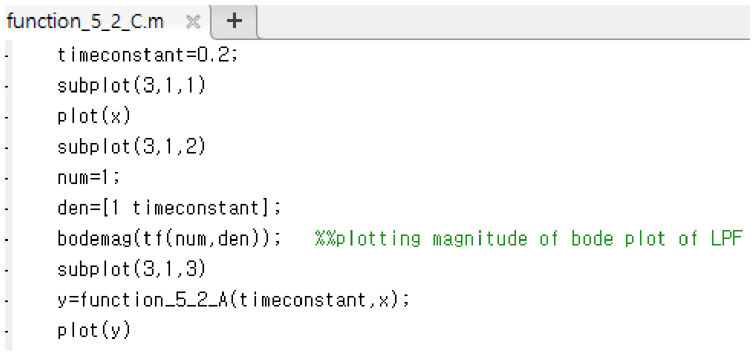
Figure 30 shows code of a low pass filter. 'cutoff' get cut-off frequency and 'time' get .

(b) Filter your voice signal x[n] by the filter h[n] to get y[n]. Use the filter function filter().

I could filter my voice by putting code : 'y=function_5_2_A(timeconstant,x)'. I set timeconstant as 0.2.

(c) Plot the voice signals and filter, x[n], h[n] and y[n] in a figure window.



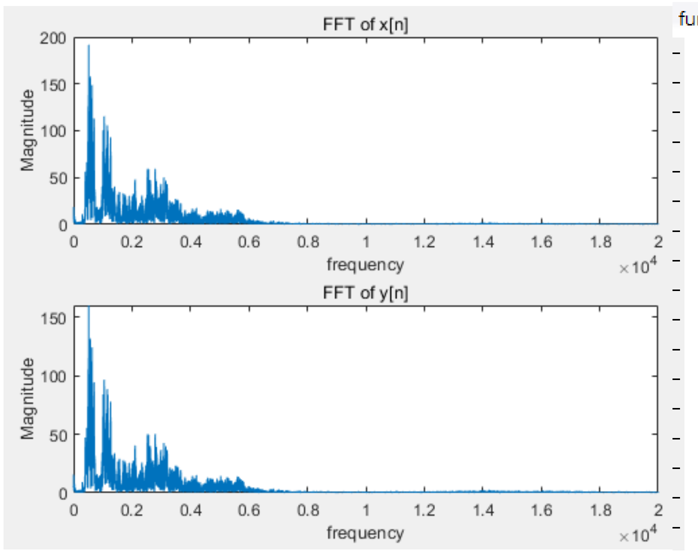

function_5_2_C.m

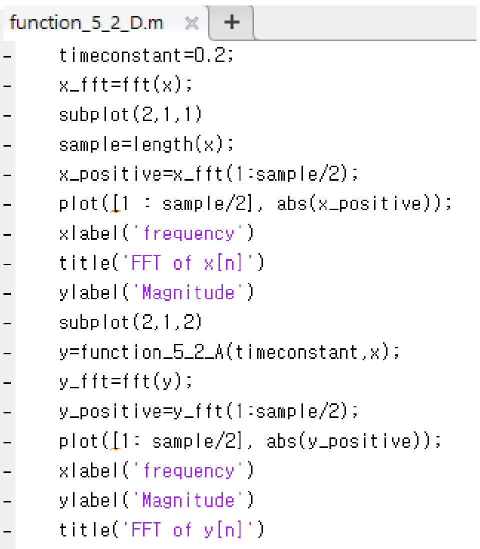
```
timeconstant=0.2;
subplot(3,1,1)
plot(x)
subplot(3,1,2)
num=1;
den=[1 timeconstant];
bodemag(tf(num,den));   %%plotting magnitude of bode plot of LPF
subplot(3,1,3)
y=function_5_2_A(timeconstant,x);
plot(y)
```

Figure 31. Plot of x[n], h[n] and y[n] and the Code to Plot

Figure 31 shows x[n], h[n] which is filter in dB-frequency graph and y[n] at upper figure and the code to plot them. To plot h[n] in frequency domain, I used 'bodemag' command. It plot magnitude part of bode plot. The y[n] is filtered x[n].

(d) Plot the frequency spectrums of the signals and the filter.





```
timeconstant=0.2;
x_fft=fft(x);
subplot(2,1,1)
sample=length(x);
x_positive=x_fft(1:sample/2);
plot([1 : sample/2], abs(x_positive));
xlabel('frequency')
title('FFT of x[n]')
ylabel('Magnitude')
subplot(2,1,2)
y=function_5_2_A(timeconstant,x);
y_fft=fft(y);
y_positive=y_fft(1:sample/2);
plot([1: sample/2], abs(y_positive));
xlabel('frequency')
ylabel('Magnitude')
title('FFT of y[n]')
```

Figure 32. Frequency Spectrum of x[n] and y[n]

Figure 32 shows frequency spectrum of x[n] and y[n]. The frequency spectrum of filter(h[n]) was shown at figure 31, second figure. The right figure of figure 31 shows code to plot them. I could observe both spectrums are same.

(e) Listen to the original voice x[n] and the filtered voice y[n] and compare them.

I could hear x[n] and y[n] by using 'sound' command. Honestly, I don't know what is different.

(f) Save the signals into .wav audio files. Use the function audiowrite().

```
audiowrite('original_voice.wav',x,10000);

audiowrite('filtered_voice.wav',y,10000);
```

Figure 33. Save original voice and filtered voice

Figure 33 shows the code to save voices by using 'audiowrite' command.

6. Conclusion

Through this project, I could be able to find out what forms invisible signals exist in real life. It was possible to create basic signals by using Matlab in (2), and it was found that the two functions combined through convolution in time-domain were equal to multiplication in frequency-domain in (3) and (4). Also, I could record my voice and check the results obtained my voice by filter in (t).

In addition, I made and conducted the following plan using the principle of AM radio that came out as an example of textbook to experiment with sampling theorem which I learned in class without using the 'filter' command of Matlab.

1) Plot the voice signal in the frequency domain by using 'fft' command
2) Make r(t) by multiplying $\cos(\omega_c t)$ with voice signal and plot r(t) in frequency domain.
3) Make g(t) by multiplying $\cos(\omega_c t)$ with r(t) and plot g(t) in frequency domain.
4) After filter g(t) by 'LPF' function what I made, get original signal by using 'ifft' command.

The result is in below.

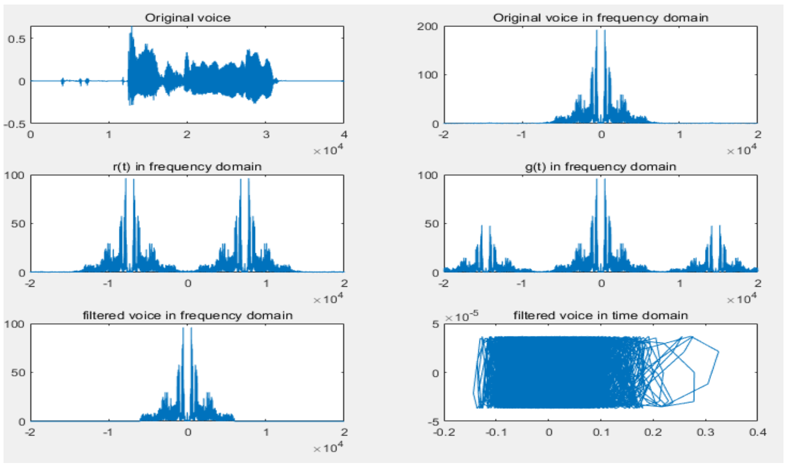


Figure 34. Additional experiment

Figure 34 shows original voice in time domain(left upside), original voice in frequency domain(right upside), r(t) in frequency domain(left middle), (t) in frequency domain(right middle), filtered voice in frequency domain(left downside), filtered voice in time domain(right downside). The filter was successful. It was possible to verify that the magnitude of the filtered voice was half of the original voice's. However, unlike the original voice in frequency domain, all of the larger values than ω_n were zero in the filtered voice, so I couldn't get my voice because the complex numbers are still remained even I used 'ifft' command. Although I couldn't find solution to solve this eventually, I could enjoy this term project by understanding of design of real radio. (I also saved code of this additional experiment with 'function_additional_experiment.m', 'frequency_s.m' and 'LPF.m' file)

7. Reference

[1] Erwin kreyszig, Advanced Engineering Mathematics, 10th edition, WILEY, 2011,p.473

[2] Simon Haykin, Barry Van Veen, Signals and Systems 2nd edition, WILEY, 2003, p.362

[3] Wikipedia, low pass filter,

https://commons.wikimedia.org/wiki/File:Butterworth_response.png#filehistory, on [2019.05.25]