SIMULATION TASK

Task 1

Here, we intend to simulate a realtime communication system in MATLAB. Consider the general block diagram of Fig. 1.

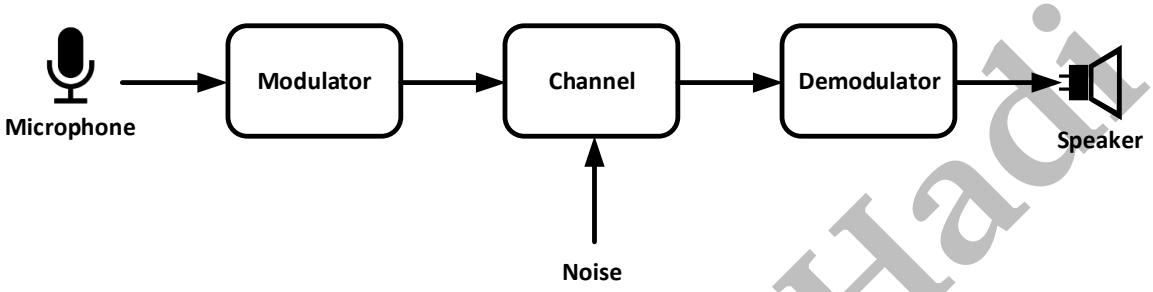

Figure 1: Block diagram of an analog communication system.

Note

At first I wrote the code for part a, b and d for continous form. But because voice is discrete I should've used digital to analog converter which is tooooooo slow. So I wrote another code which is discrete so there is no need for digital to analog converter. But for part a, b and d method 1 is okay. So for these parts I uploaded both methods.
In part a, b and d by "method 1" I mean the continuous method and by "method 2" I mean the discrete method.

How to run the code

Running the codes for method 1:
Please go "Codes" folder and then go to "Method 1" folder and run Main.m
In Main.m the results for PM and DSB after running will be plotted. But I commented the plot of SNR because it takes too much time. To test the SNR please uncomment lines 16 and 17
Running the codes for method 2:
Please go "Codes" folder and then go to "Method 2" folder. To run part e for DSB open real-time-dsb, to run part e for PM open real-time-pm and to run part a, b, c and d open main.m. Note that in main there are three sections. To test part b,d(DSB) run section 1, to test part c(DSB) run section 2, to test part a,c,d(PM) run section 3

Task 1 continued on next page...

(a) Assume that the modulator and demodulator are PM. Write a MATLAB code to simulate the PM communication system. Create separate MATLAB functions for the modulator, demodulator, and channel. Then, connect them in a main mfile. Name the functions pm_modulator, channel, and pm_demodulator.

METHOD 1

For PM modulation we know that

$$u(t) = A_c cos(2\pi f_c t + \phi(t)) = A_c cos(2\pi f_c t + k_p m(t))$$

where k_p is called phase deviation constant. So the code for this formula will be:

```
function Y = PM_modulator(Ac,fc,Kp,m)
    Y = @(t) Ac * cos(2*pi*fc*t+Kp*m(t));
end
```

Also for demodulation, we first calculate the derivative of the input signal, and then after finding the envelope, we calculate its integral:

```
function Y = PM_demodulator(Ac,fc,Kp,PMOD,t)
    du = diff(PMOD);
    [Envel,chert] = envelope( du );
    phi_p = Envel./Ac - 2*pi*fc;
    phi_p = phi_p - mean(phi_p);
    Y = cumsum(phi_p);
end
```

for channel we only add an awgn noise to the input signal
Channel:

```
function Y = channel(X,snr)
    Y = awgn(X,snr);
end
```

in the following code, we have defined a function named "PM-System" which uses the above three functions and gets an input signal with modulation characteristics and gives the demodulated output and plots the results. The results are plotted in figure 2.

```
clc;

fc = 1E6;
Ac = 10;
Kp = 10;
```

Task 1 continued on next page...

```
6
7  m = @(t) cos(2*pi*1E4*t) + cos(2*pi*0.7E4*t) + cos(2*pi*0.4E4*t)
       ;
8  fMax = 1E4;
9  Tmax = 10 / fMax;
10 PM_System(Ac,fc,Kp,Tmax,m,0.1);
11
12
13 function Y = PM_System(Ac,fc,Kp,Tmax,m,snr)
14
15     t = linspace(0,Tmax,Tmax*10*fc);
16     Modulated = PM_modulator(Ac,fc,Kp,m);
17     PMOD = [];
18     M = [];
19     for i=1:length(t)
20         PMOD(i) = Modulated(t(i));
21         M(i) = m(t(i));
22     end
23     subplot(4,1,1)
24     plot(M,'linewidth',2,'color','black')
25     title("First Signal")
26     subplot(4,1,2)
27     plot(PMOD,'linewidth',2,'color','red')
28     title("Modulated Signal")
29     PMOD1 = channel(PMOD,snr);
30     subplot(4,1,3)
31     plot(PMOD1,'linewidth',2,'color','green')
32     title("Modulated Signal Passed Through Channel")
33     PDEMOD = PM_demodulator(Ac,fc,Kp,PMOD1,t);
34     subplot(4,1,4)
35     plot(PDEMOD,'linewidth',2,'color','blue')
36     title("Demodulated Recieved Signal")
37     Y = PDEMOD;
38 end
```

METHOD 2

This method is discrete which is faster for voice in part c and e. For PM modulation we calculate lowpass of upsampled signal and then according to the formula of PM modulatoin we calculate the modulated signal (in general for modulator we also need a bandpass filter(for fdm). But because in this project we only have one message so here we do not need a bandpass filter. But because we have noise so to find snr we need a bandpass filter in demodulator)

```
1 function [msg_mod, t_for_demod] = pm_modulator(fs, msg)
2 global Fs
3 msg_mod_prime = upsample(msg, Fs / fs);
4 N = numel(msg_mod_prime);
```

Task 1 continued on next page...

```
t_for_demod = 0:(1/Fs):(N-1)/Fs;
msg_zegond = lowpass(msg_mod_prime, 4000, Fs, 'ImpulseResponse',
    'iir', 'Steepness', 0.95);
msg_mod = cos(2*pi*(1e5)*t_for_demod + (msg_zegond')/max(abs(
    msg_zegond)));
end
```

Then we should pass the modulated signal through the channel. For channel we get the modulated signal, W and N0.Then using the following formulas we find the noise power and the signal power.

$$noise\ power = \frac{2N0}{2B} \qquad signal\ power = mean(mod.^2)$$

Then by dividing the powers we find the snr. Note that awgn function in MATLAB gets snr. So by the above formulats in our function we find the snr.

```
function [chn, t] = channel(mod, t, W, N0)
B = 2*W;
n_pow = 2*N0/2*B;
s_pow = mean(mod.^2);
snr_ch = 10*log10(s_pow/n_pow);
chn = awgn(mod, snr_ch, 'measured');
end
```

Since we have noise so to find snr we need a bandpass filter in demodulator. After the bandpass filter we use FM2AM method to find the demodulated signal. In this method we first calculate the derivative of the input, then we use an envelope detecter and after calculating the integral we remove the DC part. Figures 3 and 4 are to better illustrate these things.

```
function msg_demod = pm_demodulator(msg_ch, t_for_demod)
global Fs
msg_ch_prime = bandpass(msg_ch, [9e4 11e4], Fs);
derivative = diff(msg_ch_prime) / (t_for_demod(2) - t_for_demod
    (1));
envelope = sqrt(derivative.^2 + imag(hilbert(derivative)).^2);
DCR = envelope - movmean(envelope,2500);
msg_demod_with_DC = (t_for_demod(2) - t_for_demod(1))*cumsum(DCR
    );
DCR_msg = msg_demod_with_DC - movmean(msg_demod_with_DC,2500);
msg_demod = DCR_msg;
end
```

Task 1 continued on next page...

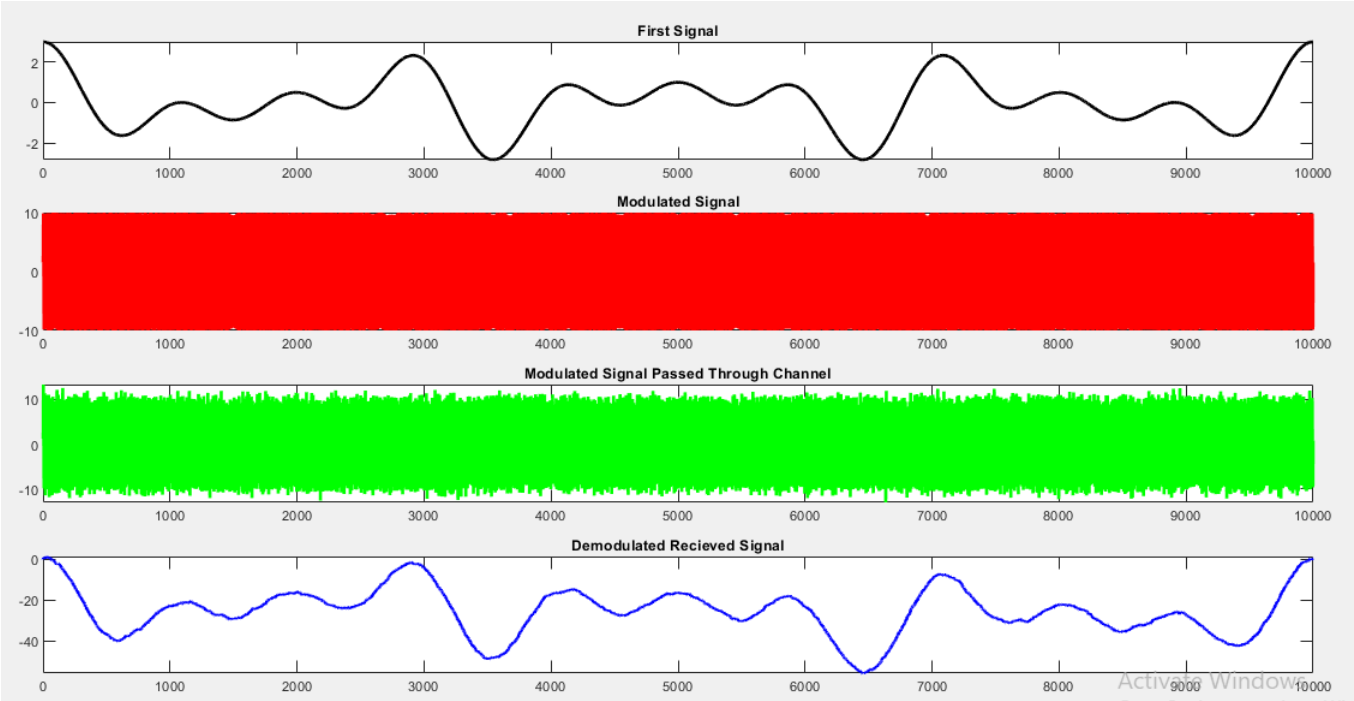

Figure 2: PM

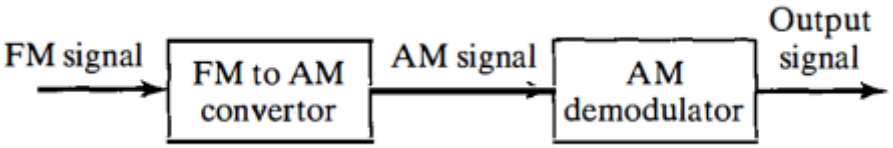

Figure 3: FM to AM demodulator with differentiator

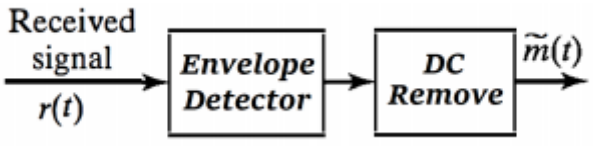

Figure 4: Block diagram of the AM envelope Demodulator

Task 1 continued on next page...

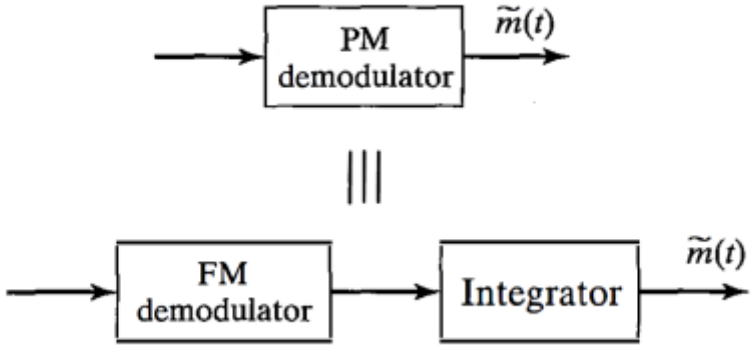

Figure 5: : A comparison of frequency and phase demodulators

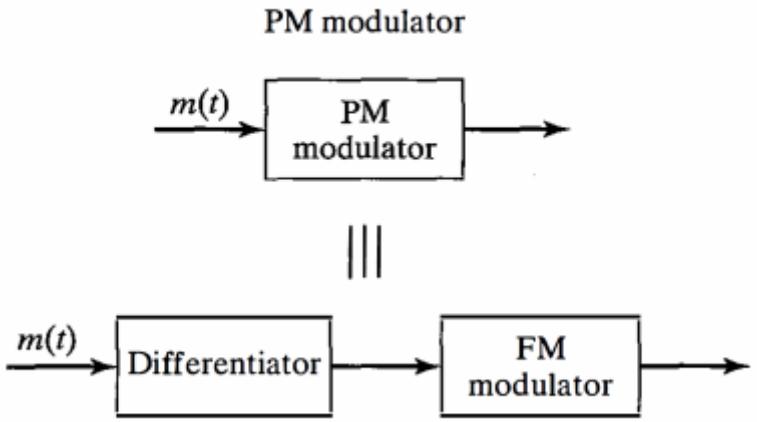

Figure 6: A comparison of frequency and phase modulators

(b) Repeat the previous part for a DSB communication system. Note that the channel function does not change. You only need to code two new functions for the DSB modulator and demodulator. Name these new functions dsb_modulator and dsb_demodulator.

METHOD 1

The formula for dsb modulation that was introduced in slides is

$$u(t) = A_c m(t) cos(2\pi f_c t)$$

So like this formula, its code will be

```
function Y = DSB_modulator(Ac, fc ,m)
    Y = @(t) Ac * m(t) * cos(2*pi*fc*t);
end
```

Also for demodulation we first multiply it by $cos(2\pi f_c t)$ then we pass it through a lowpass filter. So

```
function Y = DSB_demodulator(Ac,fc,dsbMOD,t,fpass)
    fsample = length(t)/t(length(t));
    D = dsbMOD.*cos(2*pi*fc.*t);
    Y = lowpass(D,fpass,fsample);

end
```

In Main.m we defined a function named DSB-System which gets a signal(with modulation characteristics) as an input and then by using three functions gives the demodulated output signal and plots it. The results are plotted in figure 7.

```
clc;

fc = 1E6;
Ac = 10;
Kp = 10;

m = @(t) cos(2*pi*1E4*t) + cos(2*pi*0.7E4*t) + cos(2*pi*0.4E4*t)
    ;
fMax = 1E4;
Tmax = 10 / fMax;
DSB_System(Ac,fc,Tmax,m,0.1);

function Y = DSB_System(Ac,fc,Tmax,m,snr)
    dsbMOD_T = DSB_modulator(Ac,fc,m);
    t = linspace(0,Tmax,Tmax*10*fc);
    dsbMOD = [];
    M = [];
    for i=1:length(t)
        dsbMOD(i) = dsbMOD_T(t(i));
        M(i) = m(t(i));
    end
    subplot(4,1,1)
    plot(M,'linewidth',2,'color','black')
    title("First Signal")
    subplot(4,1,2)
    plot(dsbMOD,'linewidth',2,'color','red')
    title("Modulated Signal")
    dsbMOD1 = channel(dsbMOD,snr);
    subplot(4,1,3)
    plot(dsbMOD1,'linewidth',2,'color','green')
    title("Modulated Signal Passed Through Channel")
```

Task 1 continued on next page...

```
    dsbDEMOD = DSB_demodulator(Ac,fc,dsbMOD1,t,2E4);
    subplot(4,1,4)
    plot(dsbDEMOD,'linewidth',2,'color','blue')
    title("Demodulated Recieved Signal")
    Y = dsbDEMOD;
end
```

METHOD 2

First we do a mixing by multiplying the message and the carrier. Then we apply a band-pass filter on modulated signal. Then we find the convolution of mixed and bandpass filter.

```
function [mod, t] = dsb_modulator(msg, t, W, fc, Ac)
mixed = Ac*msg.*cos(2*pi*fc*t);
B = 2*W;
bpf = abs(t(2)-t(1))*2*B*sinc(B*t).*cos(2*pi*fc*t);
mod = conv(mixed, bpf);
mod = mod(1:numel(t));
end
```

Again, for demodulation we do mixing, then by applying a lowpass filter then we find the convolution pf mixed signal and lowpass filter. So we have

```
function [demod, t] = dsb_demodulator(chn, t, W, fc)
%chn = bandpass(chn, [0.9*fc 1.1*fc], Fs);
mixed = chn.*cos(2*pi*fc*t);
BL = 1*W;
lpf = abs(t(2)-t(1))*BL*sinc(BL*t);
demod = conv(mixed, lpf);
demod = demod(1:numel(t));
end
```

We combine these functions in main. So we have

```
clear;
clc;
T = 300;
K = 1.38e-23;
N0 = K*T;
Ac = 1;
fc = 400000;

t=0:0.0005:20;
W = 5000;
```

Task 1 continued on next page...

```
msg = cos(2*pi*W*t);

mod = dsb_modulator(msg, t, W, fc, Ac);
chn = channel(mod, t, W, N0);
demod = dsb_demodulator(chn, t, W, fc);
```

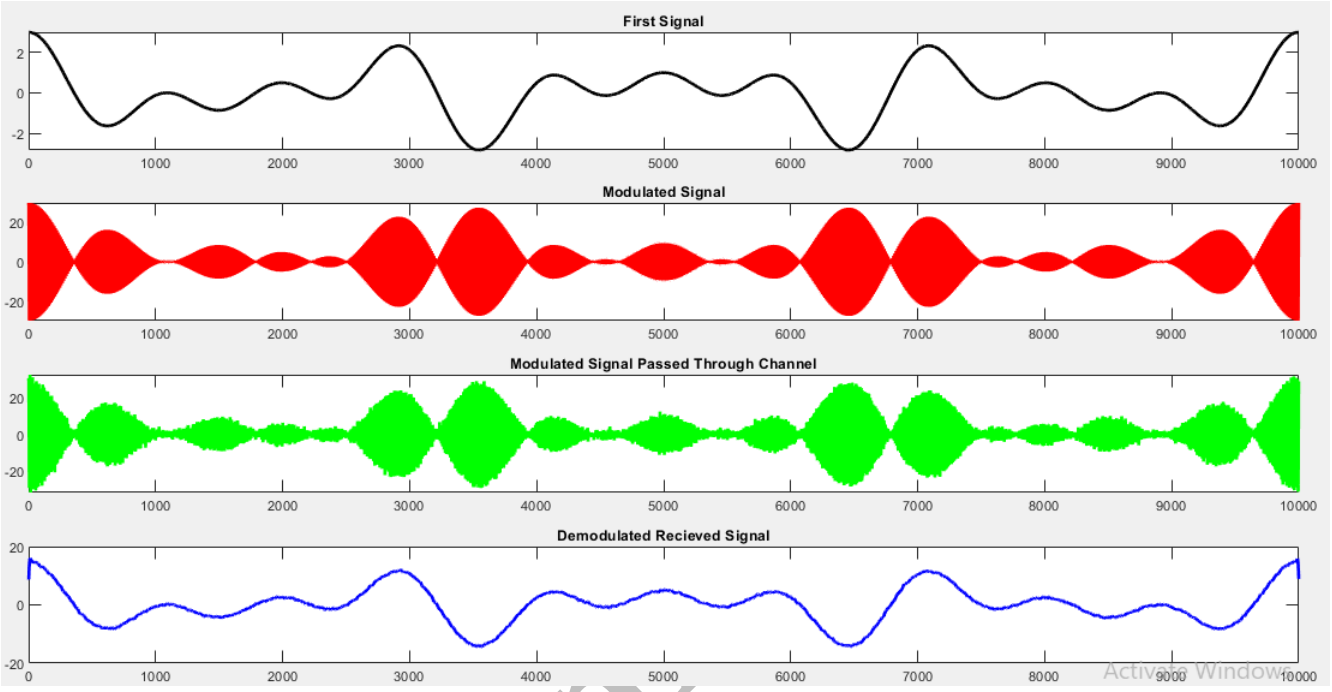

Figure 7: DSB

(c) Feed your simulation codes with a recorded audio file and play the demodulated signal and hear it for different noise levels in the channel. How do you feel when you hear the demodulated signal? Note that you can record your voice from your laptop microphone and feed it to the modulator. You can also play the demodulated signal and hear it from your laptop speaker. MATLAB has useful internal commands for working with microphones and speakers!

For audio we have two options. First option is recording our voice and after a few seconds play it. The second option is using a recorded audio. To explain both options for PM we used option 1 and for DSB we used option 2.
The code for PM using option 1(voice) is

```
Fs = 4e5;
recObj1 = audiorecorder;
disp("Start Recording");
recordblocking(recObj1, 5);
disp("Stop Recording");
fs = recObj1.SampleRate;
msg = getaudiodata(recObj1);
```

Task 1 continued on next page...

```
% pm
T = 300;
K = 1.38e-23;
N0 = K*T;
[msg_mod, t_for_demod] = pm_modulator(fs, msg);
t = 0:1/fs:(numel(msg)-1)/fs;
msg_ch = channel(msg_mod, t, fs, N0);
msg_demod = pm_demodulator(msg_ch, t_for_demod);
m = downsample(msg_demod, Fs / fs);
player=audioplayer(m, fs);
play(player)
```

As we increase the noise of the channel or the bandwidth of the channel, our voice becomes more noisy and finally our voice is not understandable at all.

The code for DSB using option 2(audio) is

```
%part c
load audio;
Fs = 8000;
t = 0:1/Fs:(numel(y)-1)*1/Fs;
y = y';
ymod = dsb_modulator(y, t, Fs, fc, Ac);
ych = channel(ymod, t, Fs, N0);
ydemod = dsb_demodulator(ych, t, Fs, fc);
player = audioplayer(ydemod, Fs);
play(player);
```

As we increase the noise of the channel or the bandwidth of the channel, the audio becomes more noisy and finally it is not understandable at all.

(d) Compare the SNR performance of the simulated PM and DSB communication systems. To do this, you can plot the output SNR of both systems in terms of the channel noise level, message bandwidth, and so on.

METHOD 1

To find the SNR in output, first we pass noise from the modulator, channel and demodulator, and then we pass the signal. The output signal is a noisy signal. So if we remove the noise we get the signal. So now we have the signal and noise so by "cumsum" we can find the power of noise and signal and by dividing them, we get the snr. We plot the snr with respect to different snrs of the channel. The plot is in figure 8.
As we can see from the figure 8: 1)the snr of PM is bigger than the snr of DSB,

Task 1 continued on next page...

2)if we increase the snr of the channel(or equivalently decrease the bandwidth) the snr of DSB and PM increases.(it is true that PM is a little random but by average it increases)

```
clear;clc;
m1 = @(t) 0;
m2 = @(t) cos(2*pi*1E4*t) + cos(2*pi*0.7E4*t) + cos(2*pi*0.4E4*t
    );
fc = 1E6;
Ac = 10;
Kp = 4000;
fMax = 1E4;
Tmax = 10 / fMax;
snrChannel = linspace(0,10,100);
snr_output_dsb = [];
snr_output_pm = [];

for j = 1:length(snrChannel)
    t = linspace(0,Tmax,Tmax*500*fc);
    Modulated = PM_modulator(Ac,fc,Kp,m1);
    PMOD = [];
    for i=1:length(t)
        PMOD(i) = Modulated(t(i));
    end
    PMOD = channel(PMOD,snrChannel(j));
    PDEMOD = PM_demodulator(Ac,fc,Kp,PMOD,t);
    Y1 = PDEMOD;
    t = linspace(0,Tmax,Tmax*500*fc);
    Modulated = PM_modulator(Ac,fc,Kp,m2);
    PMOD = [];
    for i=1:length(t)
        PMOD(i) = Modulated(t(i));
    end
    PDEMOD = PM_demodulator(Ac,fc,Kp,PMOD,t);
    Y2 = PDEMOD;
    snr_output_pm(j) = cumsum((Y2 - Y1).*(Y2 - Y1))
                       / cumsum((Y1).*(Y1));

    t = linspace(0,Tmax,Tmax*500*fc);
    Modulated = DSB_modulator(Ac,fc,m1);
    PMOD = [];
    for i=1:length(t)
        PMOD(i) = Modulated(t(i));
    end
    PMOD = channel(PMOD,snrChannel(j));
    PDEMOD = DSB_demodulator(Ac,fc,PMOD,t,fMax);
    Y1 = PDEMOD;
    t = linspace(0,Tmax,Tmax*500*fc);
    Modulated = DSB_modulator(Ac,fc,m2);
```

Task 1 continued on next page...

```
    PMOD = [];
    for i=1:length(t)
        PMOD(i) = Modulated(t(i));
    end
    PDEMOD = DSB_demodulator(Ac,fc,PMOD,t,fMax);
    Y2 = PDEMOD;
    snr_output_dsb(j) = cumsum((Y2 - Y1).*(Y2 - Y1))
                        / cumsum((Y1).*(Y1));
end

plot(snrChannel,snr_output_pm,'linewidth',2,'color','black')
hold on
plot(snrChannel,snr_output_dsb,'linewidth',2,'color','red')
xlabel("Channel SNR")
ylabel("Output SNR")
legend("Output SNR of PM System","Output SNR of DSB System")
title("Output SNR per Channel SNR")
```

METHOD 2

My PM code doesn't work properly but the logic behind the code is true. We first pass the signal through the ideal channel and we get the modulated signal and we find the power of the modulated signal. Then again we pass the same signal through the noisy channel. We get the noisy demodulated signal and if we remove the modulated signal from it, we get the noise. We find the power of the noise. By dividing the power of the demodulated signal by the power of noise we can find the snr. By doing this in a for loop for different values of bandwidth we find snr for each bandwidth. The code for this algorithm for PM is as follows

```
T = 300;
K = 1.38e-23;
N0 = K*T;
t = 0:0.05:20;
fs = 100:50000:200000;
snr_pm_dB = 100:50000:200000;
msg = cos(2*pi*W*t);

for i = 1:length(fs)
    [msg_mod, t_for_demod] = pm_modulator(fs(i), msg);
    msg_ch = channel(msg_mod, t, fs(i), 0);
    msg_demod = pm_demodulator(msg_ch, t_for_demod);
    signal = msg_demod;
    s_pow = mean(signal.^2);
    [msg_mod, t_for_demod] = pm_modulator(fs(i), msg);
    msg_ch = channel(msg_mod, t, fs(i), N0);
    msg_demod = pm_demodulator(msg_ch, t_for_demod);
    noisySignal = msg_demod;
    noise = noisySignal - signal;
    n_pow = mean(noise.^2);
```

Task 1 continued on next page...

```
    snr_pm = abs(s_pow / n_pow);
    snr_pm_dB(i) = 10*log10(snr_pm);
end
plot(fs,snr_pm_dB)
```

For calculating the SNR of DSB, we first pass the signal through the ideal channel and we get the modulated signal and we find the power of the modulated signal. Then again we pass the same signal through the noisy channel. We get the noisy demodulated signal and if we remove the modulated signal from it, we get the noise. We find the power of the noise. By dividing the power of the demodulated signal by the power of noise we can find the snr. By doing this in a for loop for different values of bandwidth we find snr for each bandwidth. The code for this algorithm for DSB is as follows

```
clear;
clc;
T = 300;
K = 1.38e-23;
N0 = K*T;
Ac = 1;
fc = 400000;

% signal
t=0:0.0005:20;
WW = linspace(0.1,1000,100);
snr_dsb_dB = linspace(5,5000,100);
for i = 1:100
    W = WW(i);
    msg = cos(2*pi*W*t);
    mod = dsb_modulator(msg, t, W, fc, Ac);
    chn = channel(mod, t, W, 0);
    demod = dsb_demodulator(chn, t, W, fc);
    signal = demod;
    s_pow = mean(demod.^2);
    mod = dsb_modulator(msg, t, W, fc, Ac);
    chn = channel(mod, t, W, N0);
    demod = dsb_demodulator(chn, t, W, fc);
    noisySignal = demod;
    noise = noisySignal - signal;
    n_pow = mean(noise.^2);
    snr_dsb = (s_pow / n_pow);
    snr_dsb_dB(i) = 10*log10(snr_dsb);
end
plot(WW, snr_dsb_dB)
```

The results for DSB is in figure 9
As we can see from the figure 9, if the bandwidth increases, the SNR decreases(as we

Task 1 continued on next page...

expected from what we learned in the course)

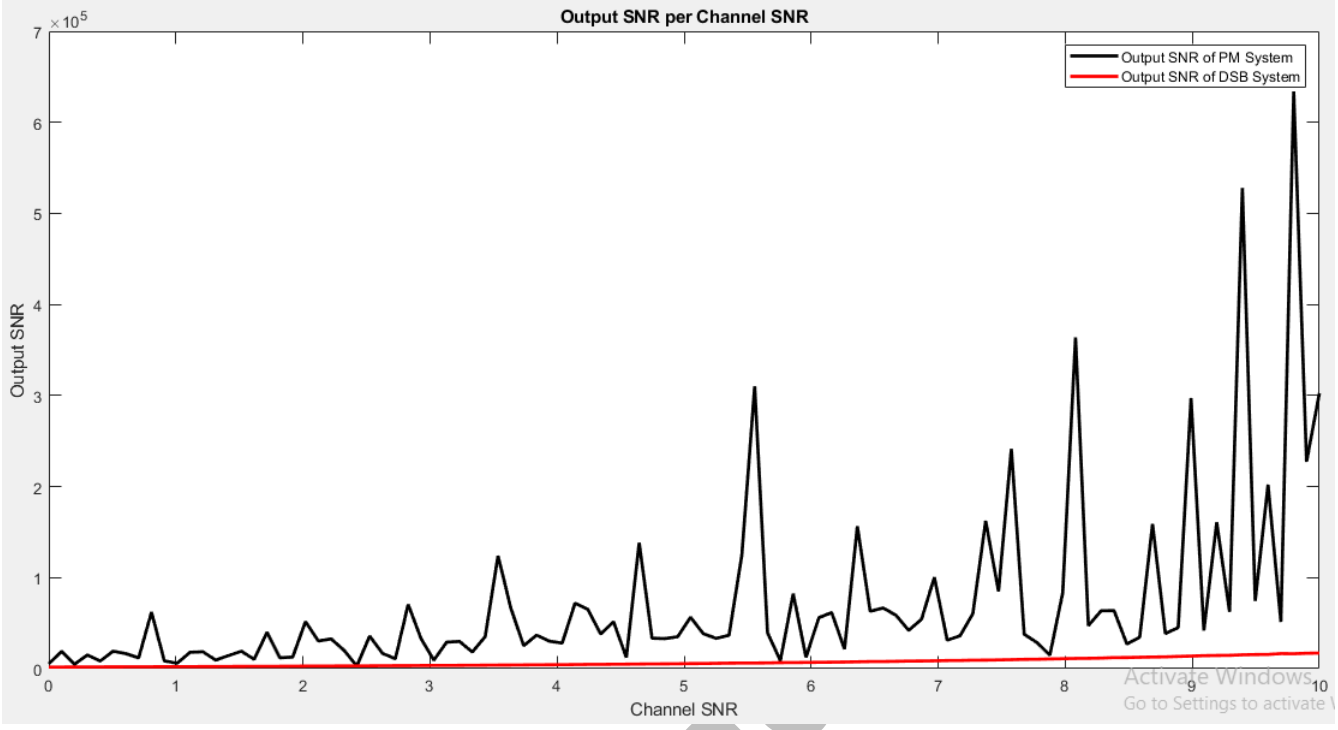

Figure 8: Output SNR(of DSB and PM) with respect to Channel SNR

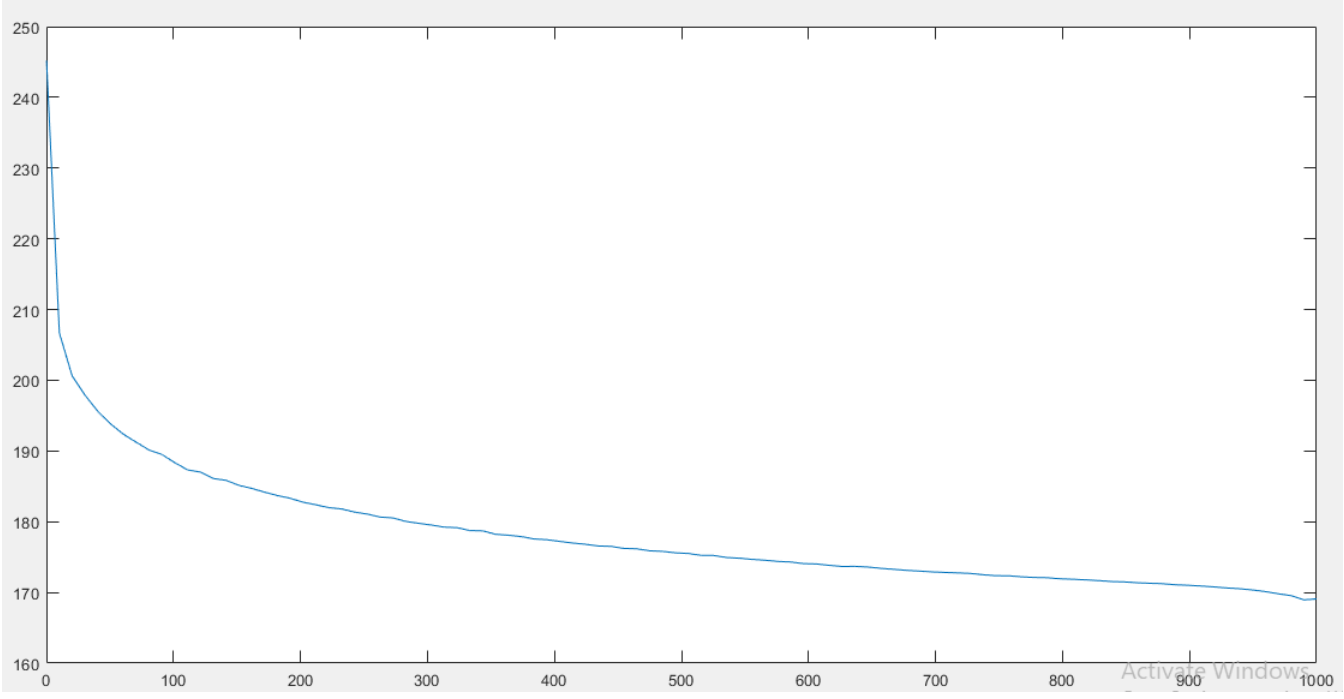

Figure 9: SNR of DSB with respect to the bandwidth

Task 1 continued on next page...

(e) Make your simulation setup realtime. In this way, you talk to the microphone and hear the demodulated signal from the speaker simultaneously without any delay and lag.

We use an infinite loop and inside it, we get the input and then by using modulator, channel and demodulator function that we developed in part a and b, we can hear the demodulated signal
For DSB we have

```
clear; clc;
deviceReader = audioDeviceReader(1e5);
deviceWriter = audioDeviceWriter('SampleRate',
                        deviceReader.SampleRate);
T = 300;
K = 1.38e-23;
N0 = K*T;
fs = 8000;
disp('Begin Signal Input...')

while 1
    y = deviceReader();
    t = 0:1/fs:(numel(y)-1)*1/fs;
    myProcessedSignal = dsb_modulator(y', t, 8000, 400000, 1);
    myProcessedSignal_chn = channel(myProcessedSignal, t, fs, 0)
        ;
    m = dsb_demodulator(myProcessedSignal_chn, t, 8000, 400000);
    deviceWriter(m');

    % pause(0.05);
end
disp('End Signal Input')

release(deviceReader)
release(deviceWriter)
```

And for PM we have

```
clear; close all; clc;
global Fs
T = 300;
K = 1.38e-23;
N0 = K*T;
Fs = 400000;

deviceReader = audioDeviceReader(8000);
deviceWriter = audioDeviceWriter('SampleRate',
                    deviceReader.SampleRate);
fs = 8000;
```

Task 1 continued on next page...

```
disp('Begin Signal Input...')
while 1
    y = deviceReader();
    [myProcessedSignal, t] = pm_modulator(fs,y);
    myProcessedSignal_chn = channel(myProcessedSignal, t, fs, N0
        );
    m = pm_demodulator(myProcessedSignal_chn, t);
    m = downsample(m, (400000) / fs);
    deviceWriter(m');
    %pause(0.05);
end
disp('End Signal Input')

release(deviceReader)
release(deviceWriter)
```

(f) Prepare a short report and describe your work concisely. Use suitable figures to better describe the developed codes and to make your report more readable and understandable. Attach a sample of the recorded audios as well as the developed codes to your sent report.

(g) ***Bonus!*** *Write your report in LaTeX.*