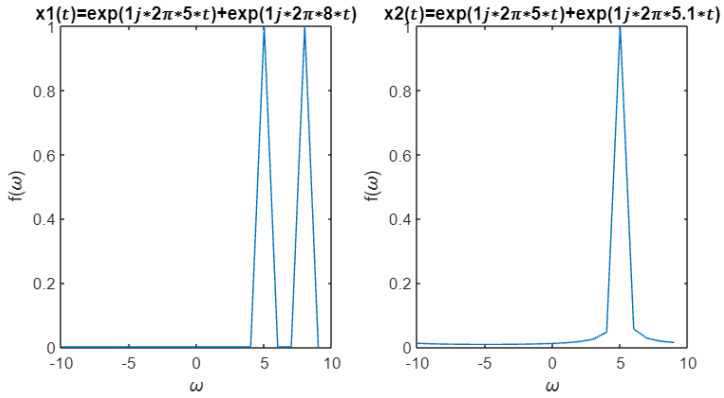
Computer Assignment 4 - Signals & Systems - Dr Akhavan

Amirali Dehghani – 810102443

Question 0-1

همانطور که در این بخش انتظار می‌رفت، هدف این است که نشان بدهیم که فرکانس ها اگر از رزولوشن فرکانسی کمتر باشند، دیگر نمی‌توان آن‌ها را در حوزه‌ی فرکانسی تفکیک کرد. که همینطور که مشاهده شد برای $x_1 = e^{j*2\pi*5*t} + e^{j*2\pi*8*t}$ فاصله‌ی بین این دو ۳ است که به مشکلی برنمیخوریم اما برای $x_2 = e^{j*2\pi*5*t} + e^{j*2\pi*5.1*t}$ چون که ۰٫۱ است و از ۱ کمتر است، نمایش داده نمی‌شود.

```
clc, clearvars, close all;
fs = 20;
t_start = 0;
t_end = 1;
ts = 1 / fs;
t = t_start:ts:t_end - ts;
T = t_end - t_start;
N = T * fs;
f = -fs/2 : fs/N : fs/2 - fs/N;
x1 = exp(1j * 2 * pi * 5  * t) + exp(1j * 2 * pi * 8  * t);
x2 = exp(1j * 2 * pi * 5  * t) + exp(1j * 2 * pi * 5.1  * t);
figure('Position', [0,0,1000,500]);
y1 = fftshift(fft(x1));
y2 = fftshift(fft(x2));
subplot(1,2,1);
plot(f, abs(y1) / max(abs(y1)));
title("x1(t)=exp(1j*2π*5*t)+exp(1j*2π*8*t)");
xlabel("\omega"); ylabel("f(\omega)")
subplot(1,2,2);
plot(f, abs(y2) / max(abs(y2)));
title("x2(t)=exp(1j*2π*5*t)+exp(1j*2π*5.1*t)");
xlabel("\omega"); ylabel("f(\omega)")
```



Part 1

```
clc, clearvars, close all;
```

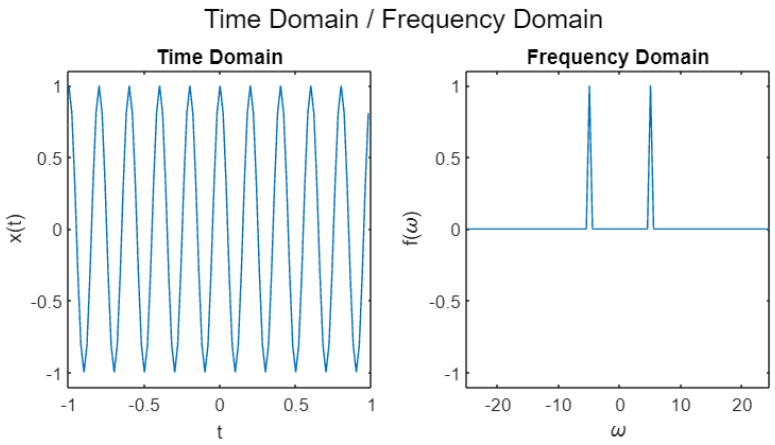
1 - 1)

مطابق توضیحاتی که در ابتدای پروژه داده شد، با ویژگی های داده شده تابع $x_1 = 10\cos(5\pi t)$ را رسم می‌کنیم. و سپس تبدیل فوریه آن را محاسبه و اندازه‌ی آن را نیز نمایش می‌دهیم که فقط در ۵ و ۵- مقدار دارد. همانطور که گفته شد همچنین باید آن را به مقدار ماکزیمم تقسیم کنیم تا عدد حاصل بین ۰ و ۱ باشد.

```
fs = 50;
t_start = -1;
t_end = 1;
ts = 1 / fs;
t = t_start:ts:t_end - ts;
T = t_end - t_start;
N = T * fs;
f = -fs/2 : fs/N : fs/2 - fs/N;
x1 = cos(10 * pi * t);

figure('Position', [0,0,1000,500]);
sgtitle("Time Domain / Frequency Domain");
subplot(1,2,1)
plot(t, x1);
title ("Time Domain");
xlabel("t"); ylabel("x(t)");
ylim([-1.1,1.1])
y1 = fftshift(fft(x1));
y1 = y1 / max(abs(y1));

subplot(1,2,2)
plot(f, abs(y1));
ylim([-1.1,1.1])
title ("Frequency Domain");
xlabel("\omega"); ylabel("f(\omega)");
```

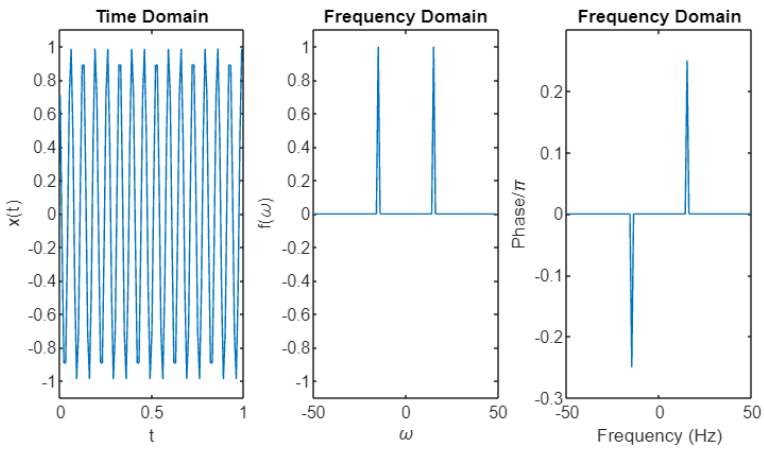


1 - 2)

حال تابع $x_2 = \cos\left(30\pi t + \frac{\pi}{4}\right)$ را رسم می‌کنیم و سپس تبدیل فوریه آن را محاسبه و اندازه‌ی آن را نیز نمایش می‌دهیم که فقط در ۱۵ و ۱۵- مقدار دارد. همانطور که گفته شد همچنین باید آن را به مقدار ماکزیمم تقسیم کنیم تا عدد حاصل

بین ۰ و ۱ باشد. در این بخش علاوه بر اندازه، فاز آن را نیز نمایش می‌دهیم که به ازای فرکانس های ۱۵- و ۱۵، به ترتیب مقادیر ۰٫۲۵- و ۰٫۲۵ دارد که مطلوب است.

```
fs = 100;
t_start = 0;
t_end = 1;
ts = 1 / fs;
t = t_start:ts:t_end - ts;
T = t_end - t_start;
N = T * fs;
f = -fs/2 : fs/N : fs/2 - fs/N;
x2 = cos(30 * pi * t + pi/4);
figure('Position', [0,0,1000,500]);
subplot(1,3,1)
plot(t, x2);
title ("Time Domain");
xlabel("t"); ylabel("x(t)");
ylim([-1.1,1.1])
y2 = fftshift(fft(x2));
y2 = y2 / max(abs(y2));
subplot(1,3,2)
plot(f, abs(y2));
title ("Frequency Domain");
xlabel("\omega"); ylabel("f(\omega)");
ylim([-1.1,1.1])
tol = 1e-6;
y2(abs(y2) < tol) = 0;
theta = angle(y2);
subplot(1,3,3)
plot(f, theta/pi);
title ("Frequency Domain");
xlabel("Frequency (Hz)"); ylabel("Phase/\pi");
ylim([-0.3,0.3])
```



Part 2

```
clc, clearvars, close all;
```

2 - 1) Creating mapset

در ابتدا ما باید مپ ست مورد نظر خودمان را بسازیم. در اینجا یک آرایه‌ی دو بعدی از حروف الفبای انگلیسی و چند کاراکتر داریم که دومین عضو هر کدام، باینری شده‌ی آن کاراکتر تا ۵ بیت است.

```
Nch = 32;
mapset = cell(2,Nch);
Alphabet ='abcdefghijklmnopqrstuvwxyz .,!";';
for i = 1:Nch
    mapset{1,i}=Alphabet(i);
    mapset{2,i}=dec2bin(i-1,5);
end
```

2 - 2)

تابع message2binary، پیام حاوی متن را که به صورت یک رشته است، دریافت کرده و تا زمانی که رشته تمام نشده یا به کاراکتر ; نرسیده، ادامه می‌دهد و در نهایت خروجی آن، یک باینری است.

```
function binarizedMessage = message2binary(message, mapset)
    binarizedMessage = '';
    for charInMessage = message
        found = false;
        for charInMapSet = mapset
            if charInMessage == charInMapSet{1};
                binarizedMessage = [binarizedMessage, charInMapSet{2}];
                found = true;
                break;
            end
        end
        if ~found
            fprintf('Character "%c" not found in mapset. Skipping...\n',
charInMessage);
        end
    end
end
```

تابع coding_freq، پیام، بیت‌ریت و مپ‌ست را به عنوان ورودی می‌گیرد. در ابتدا بررسی می‌کند که اگر تعداد بیت‌های پیام به بیت‌ریت بخش‌پذیر نیست، به آن تا جایی کاراکتر ; را اضافه می‌کند تا تعداد بیت‌ها به بیت‌ریت بخش پذیر بشود. سپس با جدا کردن بیت ها به اندازه‌ی بیت‌ریت، رزولوشن فرکانس مربوط به هر بیت ریت را با توجه فرکانس نمونه برداری محاسبه کرده(علت استفاده از $\frac{f_s}{2}$ این است که به بخش منفی سینوس نیازی نداریم.)و در نهایت در هرسری، فرکانس هرکدام را به دست می‌آوریم و به صورت یک سینوس خروجی می‌دهیم.

```
function codedMessage = coding_freq(message ,bitrate, mapset)
while(mod(length(message) * 5, bitrate) ~= 0)
    message = [message, ';'];
end
fs = 100;
ts = 1/fs;
```

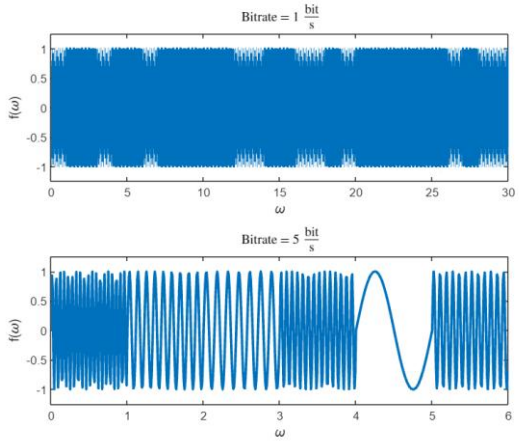
```
t_start = 0; t_end = 1;
t = t_start:ts:t_end - ts;

fStep = floor((fs / 2) / 2 ^ bitrate);
fStart = ceil(fStep / 2);
codedMessage = [];
binarizedMessage = message2binary(message, mapset);
for i = 1:bitrate:length(binarizedMessage)
    bit = binarizedMessage(i: i+bitrate-1);
    f = fStart + fStep * bin2dec(bit);
    signal = sin(2*pi*f*t);
    codedMessage = [codedMessage, signal];
end
end
```

2 - 3)

در این بخش پیام مورد نظر که signal است را به تابع coding_freq با بیت‌ریت های به ۱ و ۵ می‌دهیم و خروجی هرکدام را به صورت یک نمودار مشاهده می کنیم.

```
message = 'signal';
fs = 100;
ts = 1/fs;
bitrates = [1,5];
codedMessage = cell(1, 2);
figure('Position', [0,0,1000,800]);
for i = 1:2
    codedMessage{i} = coding_freq(message, bitrates(i), mapset);
    t = 0:(1/fs):(length(codedMessage{i})/fs)-(1/fs);
    subplot(2, 1, i);
    plot(t,codedMessage{i}, 'LineWidth', 2)
    title("Bitrate = " + bitrates(i) + "
$\frac{\mathrm{bit}}{\mathrm{s}}$",'Interpreter','latex')
    xlabel("\omega"); ylabel("f(\omega)");
    ylim([-1.25, 1.25]);
end
```



2 - 4)

برای این بخش هدف این است از سیگنال‌هایی که با استفاده از تابع decoding_amp تولید کردیم، پیام مخفی شده را رمز گشایی کنیم.

این تابع با گرفتن باینری هر کاراکتر، آن را در مپ‌ست پیدا می‌کند و کاراکتر آن را خروجی می‌دهد.

```
function character = getCharacter(binary, mapset)
    for char = mapset
        if char{2} == binary
            character = char{1};
            break;
        end
    end
end
```

حال تابع decoding_amp، سیگنال‌های رمزگذاری شده را به همراه بیت‌ریت و مپ‌ست ورودی می‌گیرد. این تابع هر بخش از سیگنال را جدا کرده و تبدیل فوریه آن را حساب می‌کند و اندیس مقدار ماکسیمم را برمیدارد. سپس تمامی فرکانس‌ها را با این اندیس به دست آمده بررسی می‌کند. اگر تفاوت آن ها با هم کمتر از $\pm\frac{f_s}{2}$ باشد، باینری آن را ذخیره می‌کند و در نهایت ۵ تا ۵ تا باینری ها جدا می‌شوند و حرف معادل ان در مپ ست را خروجی می‌دهد.

```
function decodedMessage = decoding_freq(codedMessage, bitrate, mapset)
fs = 100;
fStep = floor((fs / 2) / 2 ^ bitrate);
fStart = ceil(fStep / 2);
binarizedMessage = '';
decodedMessage = '';
for i = 1:length(codedMessage)/fs
    segment = codedMessage((i-1) * fs + 1: i * fs);
    [~, maxF] = max(abs(fftshift(fft(segment))));
    maxF = abs(maxF - fs / 2 -1);
    for j = 0:(2^bitrate - 1)
        if (fStart + j * fStep - fStep/2) <= maxF && maxF <= (fStart + j *
fStep + fStep/2)
            binarizedMessage = [binarizedMessage, dec2bin(j,bitrate)];
            break;
        end
    end
end
for i = 1:5:floor(length(binarizedMessage) /5) * 5
    character = getCharacter(binarizedMessage(i:i+4), mapset);
    decodedMessage = [decodedMessage, character];
end
end
```

```
 for i = 1:2
     decodedMessage = decoding_freq(codedMessage{i}, bitrates(i), mapset);
     fprintf ('Bitrate: %d Decoded Message: %s\n', bitrates(i),
decodedMessage)
 end
```

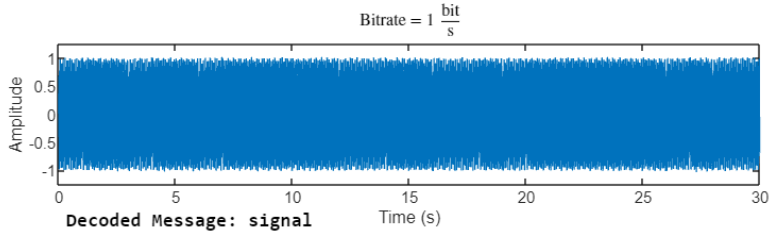
```
  Bitrate: 1 Decoded Message: signal
  Bitrate: 5 Decoded Message: signal
```

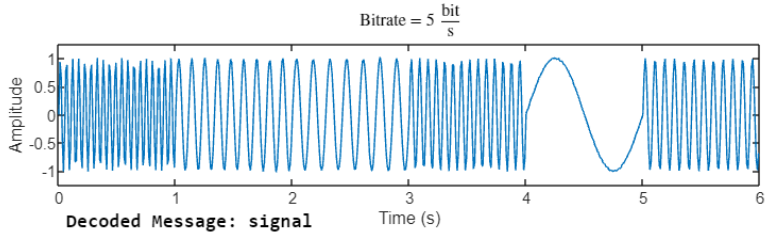
2 - 5)

برای این بخش، به سیگنال‌هایی که از قسمت ۱-۳ به دست آوردیم، یک نویز با واریانس ۰/۰۰۰۱ (انحراف معیار ۰/۰۱) اضافه می‌کنیم و آن را مشاهده و دوباره رمز گشایی می‌کنیم. چون که انحراف معیار ما کم بود، خیلی نویز اضافه شده تاثیری نداشت و همچنان پیام رمز گشایی شد اما همانطور که مشاهده می‌شود، در بیت‌ریت ۵ نویز تاثیر بیشتری گذاشته.

```
 message = 'signal';
 fs = 100;
 std = 0.01;
 codedMessage = cell(1,3);
 noisyMessage = cell(1,3);

 for i = 1:2
     codedMessage{i} = coding_freq(message, bitrates(i), mapset);
     noise = std * randn(1, length(codedMessage{i}));
     noisyMessage{i} = codedMessage{i} + noise;
     t = 0:1/fs:(length(noisyMessage{i})/fs) - 1/fs;
     figure('Position',[0,0,1500,500]);
     subplot(5,1,2:4)
     plot(t, noisyMessage{i})
     title("Bitrate = " + bitrates(i) + "
$\frac{\mathrm{bit}}{\mathrm{s}}$",'Interpreter','latex')
     xlabel("Time (s)")
     ylabel("Amplitude")
     ylim([-1.25, 1.25])
     decodedMessage = decoding_freq(noisyMessage{i}, bitrates(i), mapset);
     subplot(5,1,5)
     axis off
     text(0.01, 0.5, sprintf('Decoded Message: %s', decodedMessage),
'FontName','consolas', 'FontSize', 10, 'FontWeight','bold')
 end
```



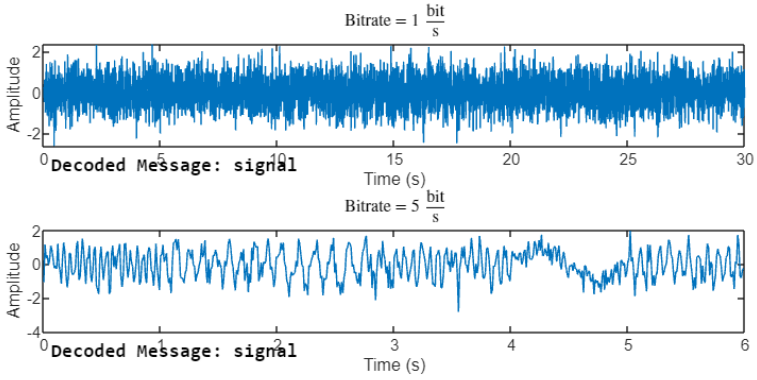


2 - 6)

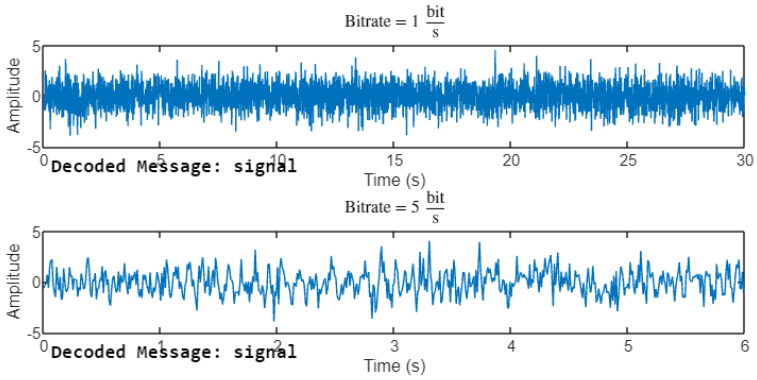
در این گام ما ۳ انحراف معیار مختلف را بررسی می‌کنیم و مشاهده می‌کنیم که با افزایش انحراف معیار، تاثیر آن در ابتدا بر روی سیگنال با بیت‌ریت بیشتر است و سپس با افزایش انحراف معیار، نویز به قدری تاثیرگذار تر می‌شود که دیگر تابع decode_freq خروجی مد نظر ما را نمی‌دهد.

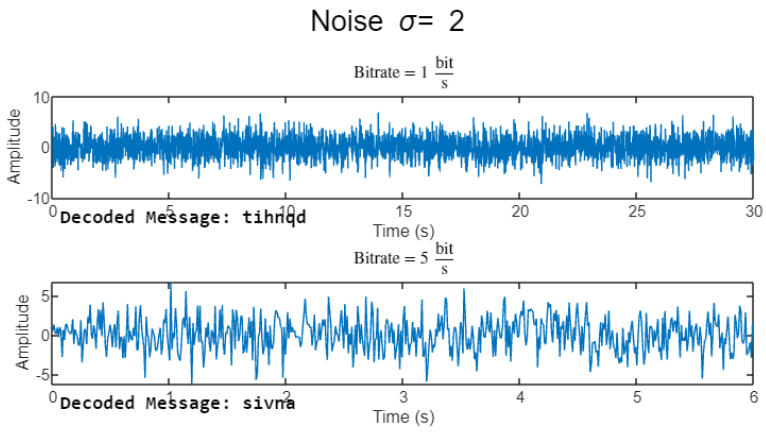
```
message = 'signal';
fs = 100;
codedMessage = cell(1,3);
noisyMessage = cell(1,3);
stds = [0.5, 1, 2];
for std = stds
    figure('Position',[0, 0, 1500, 1200]);
    sgtitle("Noise \sigma= " + std);
    for i = 1:2
        codedMessage{i} = coding_freq(message, bitrates(i), mapset);
        noise = std * randn(1, length(codedMessage{i}));
        noisyMessage{i} = codedMessage{i} + noise;
        t = 0:1/fs:(length(noisyMessage{i})/fs) - 1/fs;
        subplot(15,1,((i-1)*5 + 1):((i-1)*5 + 3));
        plot(t, noisyMessage{i})
        title("Bitrate = " + bitrates(i) + "
$\frac{\mathrm{bit}}{\mathrm{s}}$",'Interpreter','latex')
        xlabel("Time (s)"); ylabel("Amplitude");
        decodedMessage = decoding_freq(noisyMessage{i}, bitrates(i),
mapset);
        subplot(15,1,(i-1)*5 + 4);
        axis off
        text(0.01, 0.5, sprintf('Decoded Message: %s', decodedMessage),
'FontName','consolas', 'FontSize', 10, FontWeight='bold')
    end
end
```

Noise σ= 0.5



Noise σ= 1



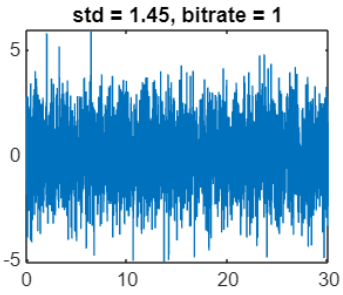


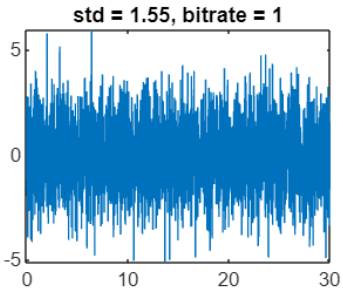
2 - 7)

برای این بخش، می‌خواهیم که بیشترین انحراف معیار ممکن برای هر بیت‌ریت را پیدا کنیم. با آزمون و خطا، انحراف معیارها طبق آن چیزی که در متغیر stds ذخیره شده است می‌باشد که برای آن، دو نمونه حروجی گرفته شده است که همانطور که مشاهده می‌شود، در این بازه‌ها که تقریبا می‌توان گفت مرز هستند، پیام دیکود شده یا خود پیام یا با اختلاف جزئی از پیام است که با هرسری اجرا کردن کد، خروجی به دلیل استفاده از randn متفاوت خواهد بود.

```
message = 'signal';
fs = 100;
codedMessage = cell(1,2);
stds = {[1.45, 1.55], [1.4,1.45]};
for i = 1:2
    codedMessage{i} = coding_freq(message, bitrates(i), mapset);
    std = stds{i};
    noise1 = std(1) * randn(1, length(codedMessage{i}));
    noise2 = std(2) * randn(1, length(codedMessage{i}));
    noisyMessage1 = codedMessage{i} + noise1;
    noisyMessage2 = codedMessage{i} + noise2;
    t = 0:1/fs:(length(codedMessage{i})/fs) - 1/fs;
    figure
    subplot(2,2,1);
    plot(t, noisyMessage1);
    title("std = " + std(1) + ", bitrate = " + bitrates(i));
    subplot(2,2,2);
    plot(t, noisyMessage1);
    title("std = " + std(2) + ", bitrate = " + bitrates(i));
    subplot(2,2,3);
```

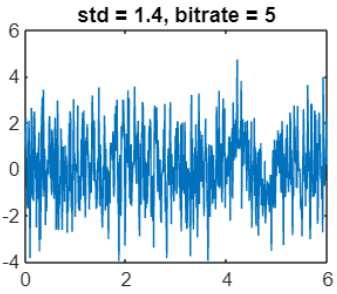
```
    text(0.01, 0.5, sprintf('Decoded Message: %s',
decoding_freq(noisyMessage1, bitrates(i), mapset)), 'FontName','consolas',
'FontSize', 10, FontWeight='bold')
    axis off
    subplot(2,2,4);
    text(0.01, 0.5, sprintf('Decoded Message: %s',
decoding_freq(noisyMessage2, bitrates(i), mapset)), 'FontName','consolas',
'FontSize', 10, FontWeight='bold')
    axis off
 end
```

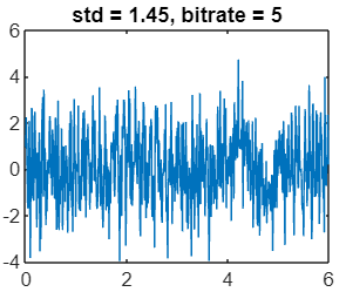





```
Decoded Message: signal
```

```
Decoded Message: sicnal
```






```
Decoded Message: signal
```

```
Decoded Message: signal
```

2 - 8)

همانطور که در صورت پروژه گفته شده است: " هر چه فاصله ی فرکانس های انتخابی بیشتر باشند کدگذاری نسبت به نویز مقاوم تر می شود .بنابراین هر چه پهنای باند بیشتری مصرف کنیم می توانیم با سرعت بیشتری اطلاعات را ارسال کنیم و در عین حال نسبت به نویز مقاوم باشیم."

2 - 9)

خیر – وقتی پهنای‌باند ثابت بماند، افزایش نرخ نمونه‌برداری بیت‌ریت را بالا نمی‌برد و لذا مقاومت در برابر نویز تغییری نمی‌کند.