Computer Assignment 4 - Signals & Systems - Dr Akhavan

Amirali Dehghani - 810102443

Question 1

در این بخش، می‌خواهیم یک پیام را به صورت یک سیگنال در بیاوریم و آن را بفرستیم. برای این کار با استفاده از $sin(2\pi t)$ و به دست آوردن یک ضریب، پیام را ارسال می‌کنیم و با کورلیشن‌گیری، پیام را تشخیص می‌دهیم.

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

1 - 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
```

1 - 2) coding_amp Function

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
```

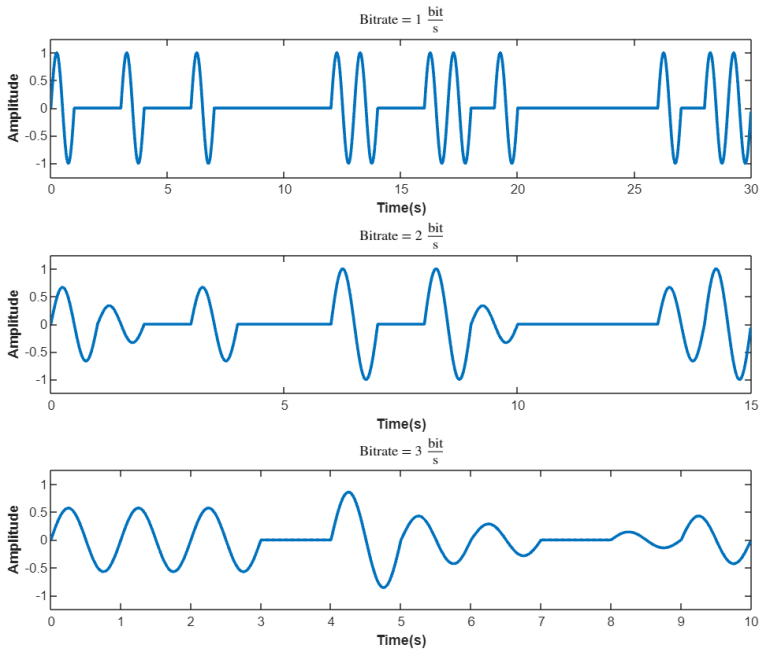
تابع coding_amp، پیام، بیت‌ریت و مپ‌ست را به عنوان ورودی می‌گیرد. در ابتدا بررسی می‌کند که اگر تعداد بیت‌های پیام به بیت‌ریت بخش‌پذیر نیست، به آن تا جایی کاراکتر ; را اضافه می‌کند تا تعداد بیت‌ها به بیت‌ریت بخش پذیر بشود. سپس با جدا کردن بیت ها به اندازه‌ی بیت‌ریت، ضریب آن‌ها را از فرمول $\alpha = \frac{decimal}{2^{bitrate}-1}$ به دست می‌آوریم. سپس هر سیگنال را از رابطه $\alpha \sin(2\pi t)$ محاسبه می‌کنیم.

```
function codedMessage = coding_amp(message ,bitrate, mapset)
while(mod(length(message), bitrate) ~= 0)
    message = [message, ';'];
end
fs = 100;
codedMessage = [];
t = 0:(1/fs):1-(1/fs);
binarizedMessage = message2binary(message, mapset);
for i = 1:bitrate:length(binarizedMessage)
    bit = binarizedMessage(i: i+bitrate-1);
    alpha = bin2dec(bit) / (2^bitrate -1);
    signal = alpha*sin(2*pi*t);
    codedMessage = [codedMessage, signal];
end
end
```

1 - 3) Encoding signal

در این بخش پیام مورد نظر که signal است را به تابع coding_amp با بیت‌ریت های به ترتیب ۱، ۲ و ۳ می‌دهیم و خروجی هرکدام را به صورت یک نمودار مشاهده می کنیم.

```
message = 'signal';
fs = 100;
codedMessage = cell(1, 3);
figure('Position', [0,0,1000,800]);
for bitrate = 1:3
    codedMessage{bitrate} = coding_amp(message, bitrate, mapset);
    t = 0:(1/fs):(length(codedMessage{bitrate})/fs)-(1/fs);
    subplot(3, 1, bitrate);
    plot(t,codedMessage{bitrate}, 'LineWidth', 2)
    title("Bitrate = " + bitrate + "
$\frac{\mathrm{bit}}{\mathrm{s}}$",'Interpreter','latex')
    xlabel("Time(s)", FontName="Arial", FontWeight="bold");
ylabel("Amplitude", FontName="Arial", FontWeight="bold");
    ylim([-1.25, 1.25]);
end
```



1 - 4) decoding_amp function

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

در ابتدا تابع correlation را تعریف می‌کنیم که با توجه به صورت پروژه و نرخ نمونه برداری ۱۰۰ هرتز، از فرمول $corr = 0.01\sum AB$ استفاده می‌کنیم و آن را طوری تنظیم می‌کنیم که بین ۰ و ۱ باشد.

```
function correlation = corr(a, b)
correlation = 0.01 * dot(a, b);
    correlation = max(0, correlation);
    correlation = min(correlation, 1);
end
```

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، آن را با $2\sin(2\pi t)$ کورلیشن می‌گیرد، مقدار به دست آمده، همان ضریبی است که درهنگام رمزگذاری، به دست آوردیم. اما برای در نظر گرفتن احتمال خطا، اگر ضریب را α در نظر بگیریم، باید چک کنیم که مقدار به دست آمده در بازه‌ی $\alpha \pm \frac{1}{2(2^{bitrate}-1)}$ باشد، درصورتی که این اتفاق بیوفتد، آن بیت را برداشته و به عنوان بیت مورد نظر به پیام اضافه می‌کنیم. در نهایت از پیام‌نهایی ۵ تا ۵ تا بیت‌ها را جدا کرده و به کاراکتر تبدیل می‌کنیم و در نهایت پیام را به صورت رشته خروجی می‌دهیم.

```
function decodedMessage = decoding_amp(codedMessage, bitrate, mapset)
fs = 100;
t = 0:1/fs:1-1/fs;
binarizedMessage = '';
decodedMessage = '';
template = 2 * sin (2 * pi * t);
for i = 1:length(codedMessage)/fs
    segment = codedMessage((i-1) * fs + 1: i * fs);
    correlation = corr(segment, template);
    for j = 0:(2^bitrate - 1)
        if max(0, j/(2^bitrate-1) - (0.5/(2^bitrate-1))) <= correlation &&
correlation <= min(1, j/(2^bitrate-1) + (0.5/(2^bitrate-1)))
            binarizedMessage = [binarizedMessage, dec2bin(j,bitrate)];
            break;
        end
    end
end
for i = 1:5:length(binarizedMessage)
    character = getCharacter(binarizedMessage(i:i+4), mapset);
    decodedMessage = [decodedMessage, character];
end
end
```

در اینجا خروجی‌هایی که از قسمت قبلی به دست آوردیم را با بیت‌ریت خودشان به تابع decoding_amp می‌دهیم و خروجی می‌گیریم که همانطور که مشاهده می‌شود سیگنال با موفقیت رمزگشایی شد.

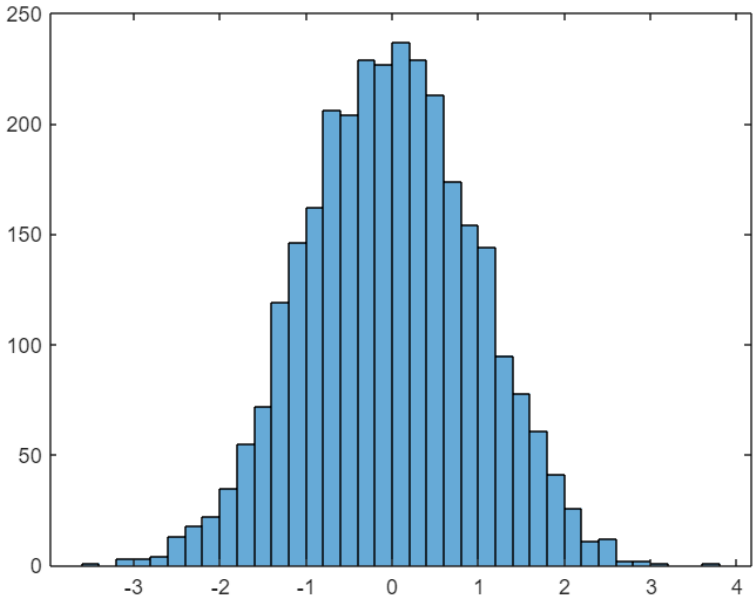
```
for bitrate = 1:3
    decodedMessage = decoding_amp(codedMessage{bitrate}, bitrate, mapset);
    fprintf ('Bitrate: %d Decoded Message: %s\n', bitrate, decodedMessage)
end
```

```
Bitrate: 1 Decoded Message: signal
Bitrate: 2 Decoded Message: signal
Bitrate: 3 Decoded Message: signal
```

1 - 5) Creating noise

برای نشان دادن اینکه خروجی تابع randn، گوسی می‌باشد، باید histogram آن را رسم کنیم، که خروجی آن برای randn(1, 3000) به شکل زیر می باشد که مشاهده می شود توزیع نرمال داریم. همچنین مشاهده می‌کنیم که میانگین به طور تقریبی ۰ و واریانس ۱ است.

```
noise = randn(1, 3000);
figure;
histogram(noise);
```




```
Mean = mean(noise)
```

```
Mean = -0.0160
```

```
variance = var(noise)
```

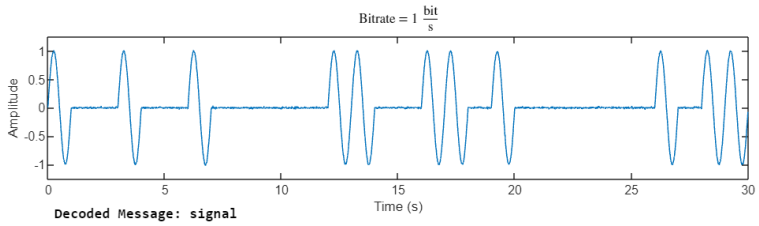
```
variance = 0.9734
```

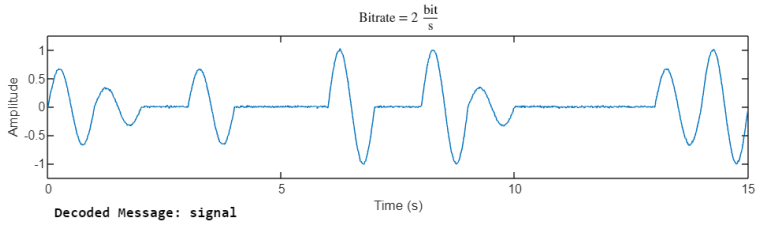
1 - 6) Adding noise to signal

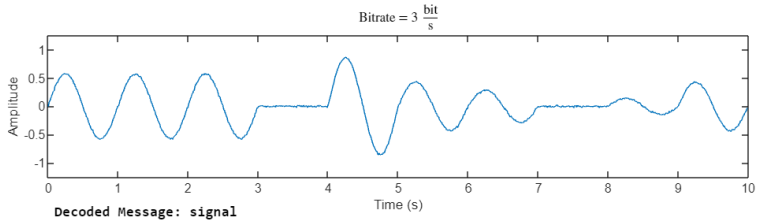
برای این بخش، به سیگنال‌هایی که از قسمت ۱-۳ به دست آوردیم، یک نویز با واریانس ۰/۰۰۰۱ (انحراف معیار ۰/۰۱) اضافه می‌کنیم و آن را مشاهده و دوباره رمز گشایی می‌کنیم. چون که انحراف معیار ما کم بود، خیلی نویز اضافه شده تاثیری نداشت و همچنان پیام رمز گشایی شد اما همانطور که مشاهده می‌شود، در بیت‌ریت ۳ نویز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

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

```
 for bitrate = 1:3
     codedMessage{bitrate} = coding_amp(message, bitrate, mapset);
     noise = std * randn(1, length(codedMessage{bitrate}));
     noisyMessage{bitrate} = codedMessage{bitrate} + noise;
     t = 0:1/fs:(length(noisyMessage{bitrate})/fs) - 1/fs;
     figure('Position',[0,0,1500,500]);
     subplot(5,1,2:4)
     plot(t, noisyMessage{bitrate})
     title("Bitrate = " + bitrate + "
$\frac{\mathrm{bit}}{\mathrm{s}}$",'Interpreter','latex')
     xlabel("Time (s)")
     ylabel("Amplitude")
     ylim([-1.25, 1.25])
     decodedMessage = decoding_amp(noisyMessage{bitrate}, bitrate, mapset);
     subplot(5,1,5)
     axis off
     text(0.01, 0.5, sprintf('Decoded Message: %s', decodedMessage),
'FontName','consolas', 'FontSize', 10, 'FontWeight','bold')
 end
```







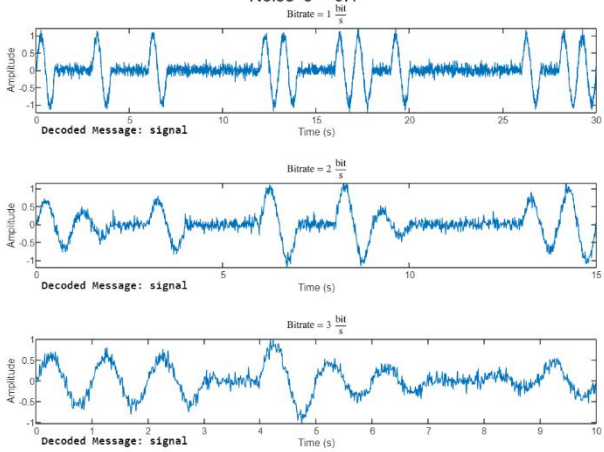
1 - 7) Trying different standard deviation

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 encode_amp خروجی مد نظر ما را نمی‌دهد.

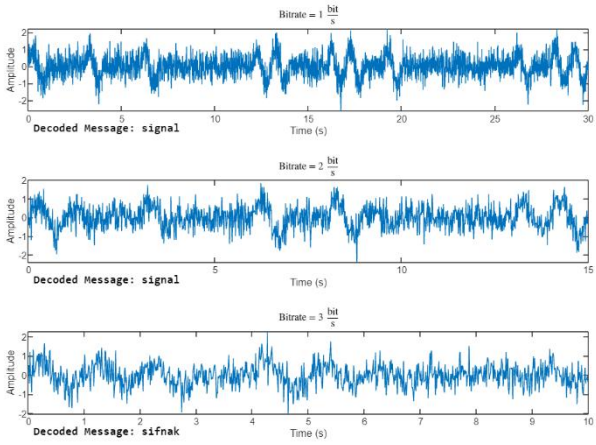
```
message = 'signal';
fs = 100;
codedMessage = cell(1,3);
noisyMessage = cell(1,3);
stds = [0.1, 0.5, 1];
for std = stds
    figure('Position',[0, 0, 1500, 1200]);
    sgtitle("Noise \sigma= " + std);
    for bitrate = 1:3
        codedMessage{bitrate} = coding_amp(message, bitrate, mapset);
        noise = std * randn(1, length(codedMessage{bitrate}));
        noisyMessage{bitrate} = codedMessage{bitrate} + noise;
        t = 0:1/fs:(length(noisyMessage{bitrate})/fs) - 1/fs;
        subplot(15,1,((bitrate-1)*5 + 1):((bitrate-1)*5 + 3));
        plot(t, noisyMessage{bitrate})
        title("Bitrate = " + bitrate + "
$\frac{\mathrm{bit}}{\mathrm{s}}$",'Interpreter','latex')
        xlabel("Time (s)"); ylabel("Amplitude");
        decodedMessage = decoding_amp(noisyMessage{bitrate}, bitrate,
mapset);
        subplot(15,1,(bitrate-1)*5 + 4);
        axis off
        text(0.01, 0.5, sprintf('Decoded Message: %s', decodedMessage),
'FontName','consolas', 'FontSize', 10, FontWeight='bold')
    end
end
```

(نمودار ها در صفحه‌ی بعد می‌باشد.)

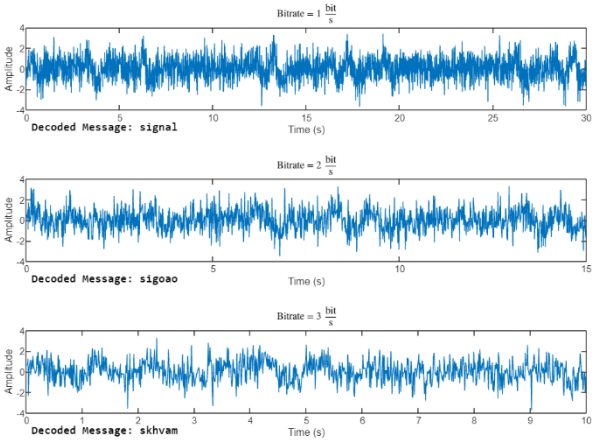
Noise σ= 0.1



Noise σ= 0.5



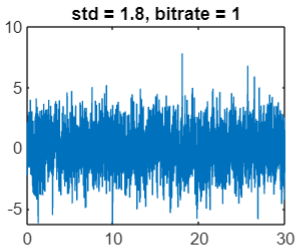
Noise σ= 1



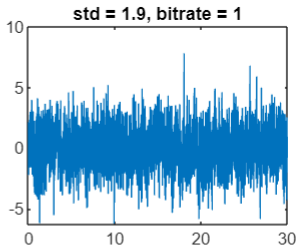
1 - 8) Finding maximum variance for each bitrate

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.

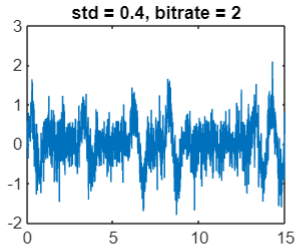
```
 message = 'signal';
 fs = 100;
 codedMessage = cell(1,3);
 stds = {[1.8, 1.9], [0.4,0.5], [0.2,0.25]};
 for bitrate = 1:3
     codedMessage{bitrate} = coding_amp(message, bitrate, mapset);
     std = stds{bitrate};
     noise1 = std(1) * randn(1, length(codedMessage{bitrate}));
     noise2 = std(2) * randn(1, length(codedMessage{bitrate}));
     noisyMessage1 = codedMessage{bitrate} + noise1;
     noisyMessage2 = codedMessage{bitrate} + noise2;
     t = 0:1/fs:(length(codedMessage{bitrate})/fs) - 1/fs;
     figure
     subplot(2,2,1);
     plot(t, noisyMessage1);
     title("std = " + std(1) + ", bitrate = " + bitrate);
     subplot(2,2,2);
     plot(t, noisyMessage1);
     title("std = " + std(2) + ", bitrate = " + bitrate);
     subplot(2,2,3);
     text(0.01, 0.5, sprintf('Decoded Message: %s',
decoding_amp(noisyMessage1, bitrate, mapset)), 'FontName','consolas',
'FontSize', 10, FontWeight='bold')
     axis off
     subplot(2,2,4);
     text(0.01, 0.5, sprintf('Decoded Message: %s',
decoding_amp(noisyMessage2, bitrate, mapset)), 'FontName','consolas',
'FontSize', 10, FontWeight='bold')
     axis off
 end
```



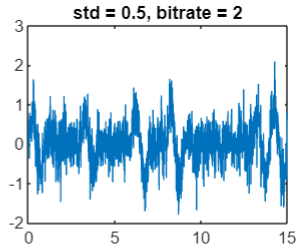

```
Decoded Message: signal
```



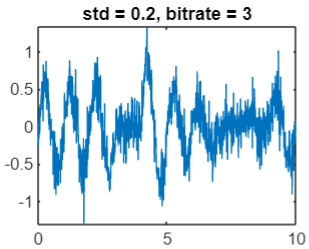

```
Decoded Message: sigpal
```



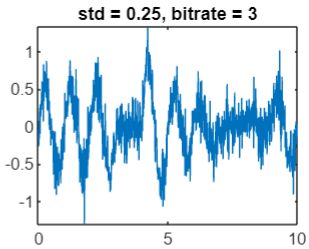

```
Decoded Message: signal
```




```
Decoded Message: signal
```




```
Decoded Message: signal
```




```
Decoded Message: signad
```

1 – 9)

همان‌طور که در صورت پروژه نیز گفته شده است، راهکار برای مقاوم کردن سیگنال نسبت به نویز، افزایش دامنه است. زیرا ضرایبی که برای هر بیت در نظر می‌گیریم، حال فاصله‌ی بیشتری از هم می‌گیرند و احتمال اشتباه کمتر می‌شود و حساسیت نسبت به نویز کمتر می‌شود.

1 – 10)

می‌توان تا هرچقدر افزایش داد اما بدلیل اینکه هر عدد اعشاری را تنها در ۳۲ بیت میتوان ذخیره کرد، با افزایش بیت‌ریت از یک جایی به بعد تمایز دوسیگنال متفاوت، غیر ممکن خواهد بود چون که دو سیگنال بر یکدیگر منطبق می‌شوند.

1 – 11)

بی‌تاثیر است. چون با اینکه با این کار، دامنه‌ی سیگنال را ۵ برابر کرده‌ایم اما در کنار آن، نویز نیز ۵ برابر شده است.

1 – 12)

بیشینه سرعت ADSL خانگی برابر با ۱۶ مگابیت بر ثانیه است که یعنی در هر ثانیه ۱۶ میلیون بیت را جابه‌جا می‌کنند و ما در این پروژه بیشینه بیت‌ریتی که با آن کار کردیم، ۳ بیت بر ثانیه بود.