بـه نـام خدا

سید محمد حسین مظهری ۸۱۰۱۰۱۵۲۰ پروژه ۴ درس سیگنال و سیستم

پارسا دقیق 810101419

بخش اول :

تمرین ۱-۱)

برای ایجاد mapset یک تابع ایجاد میکنیم که در سطر اول کاراکتر ها را قرار دهد و در سطر دوم عدد متناظر باینری آنها را بگذارد .

تابع ایجاد کردن mapset :

```
function output = create_mapset()

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

end
```

:

بخشی از خروجی تابع :

1	2	3	4	5	6	7	8	9	10	11	12
a	b	c	d	e	f	g	h	i	j	k	l
00000	00001	00010	00011	00100	00101	00110	00111	01000	01001	01010	01011

تمرین ۱-۲)

در تابع coding_amp ، پیام مورد نظر را با توجه به mapset به عددی باینری تبدیل می کنیم .

هر کاراکتر را جدا جدا تبدیل کرده و به انتهای یک آرایه که نشان دهنده کل پیام است اضافه می کنیم .

حال برای ایجاد سیگنال ، بیت های پیام را به اندازه ی bit rate جدا می کنیم و عدد decimal متناظر با آن را حساب می کنیم .

عدد ثابتی که قرار است در سینوس ضرب شود ، از رابطه ی زیر به دست می آید :

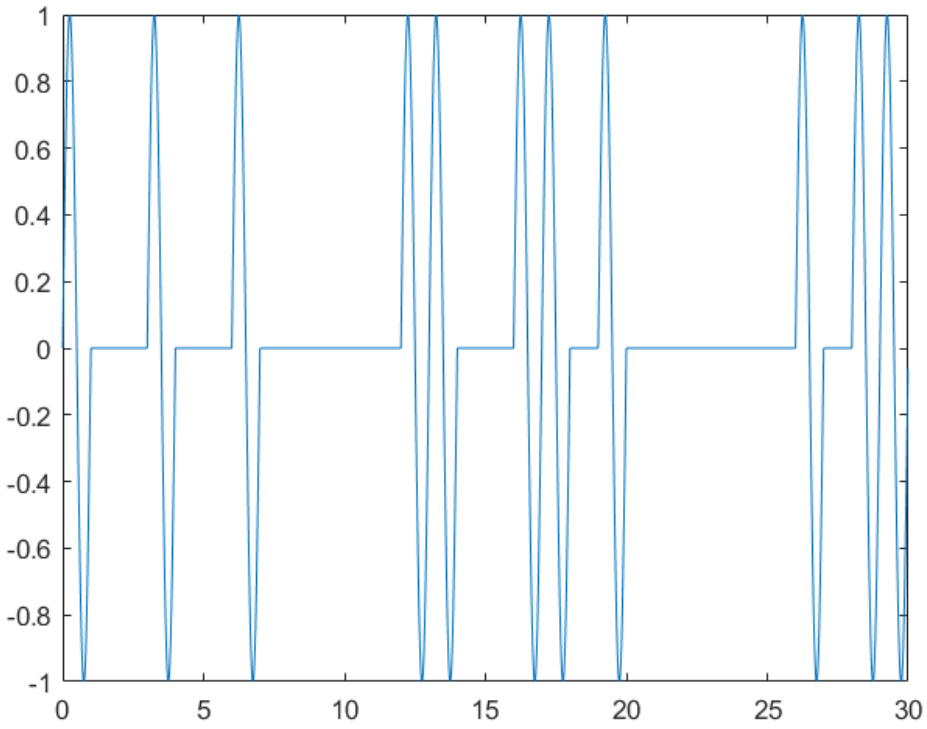
$$const = \frac{decimal}{2^{bit\ rate} - 1}$$

با توجه به رابطه ی گفته شده عدد ثابت را به دست می آوریم و در یک ثانیه با فرکانس نمونه برداری fs = 100 Hz سیگنال سینوسی را تولید می کنیم . در پایان کار سیگنال کد گذاری شده ما درست شده است .

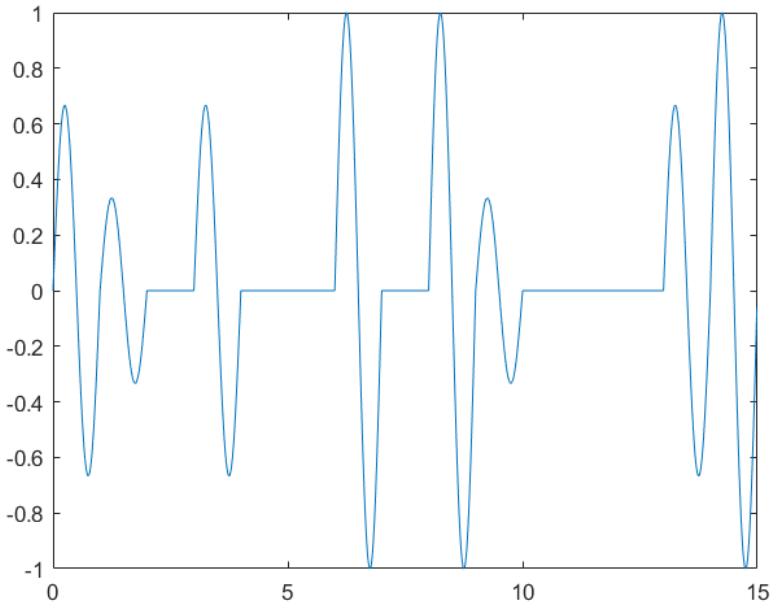
```
function coded_sig = coding_amp(message,bit_rate)

    fs = 100;
    coded_sig =[];
    mapset = create_mapset();

    % 1- Turning the message into a binary number
    Nch = 32;
    message_len = strlength(message);
    message_bin = cell(1,message_len);
    for i=1:message_len
        ch = message(i);
        for j=1:Nch
            if ch == mapset{1,j}
                message_bin{i} = mapset{2,j};
            end
        end
    end
    binarymessage = cell2mat(message_bin);
    binarymessage_len = length(binarymessage);

    % 2- Creating the coded signal
    for z=1:bit_rate:binarymessage_len
        bin_to_write = binarymessage(z:z+bit_rate-1);
        decimal = bin2dec(bin_to_write);
        cons = decimal/((2^bit_rate)-1);
        t=(z/bit_rate):(1/fs):(z/bit_rate)+1-(1/fs);
        to_write = cons*sin(2*pi*(t-(z/bit_rate)));
        coded_sig = [coded_sig to_write];
    end
```

تمرین ۱-۳)

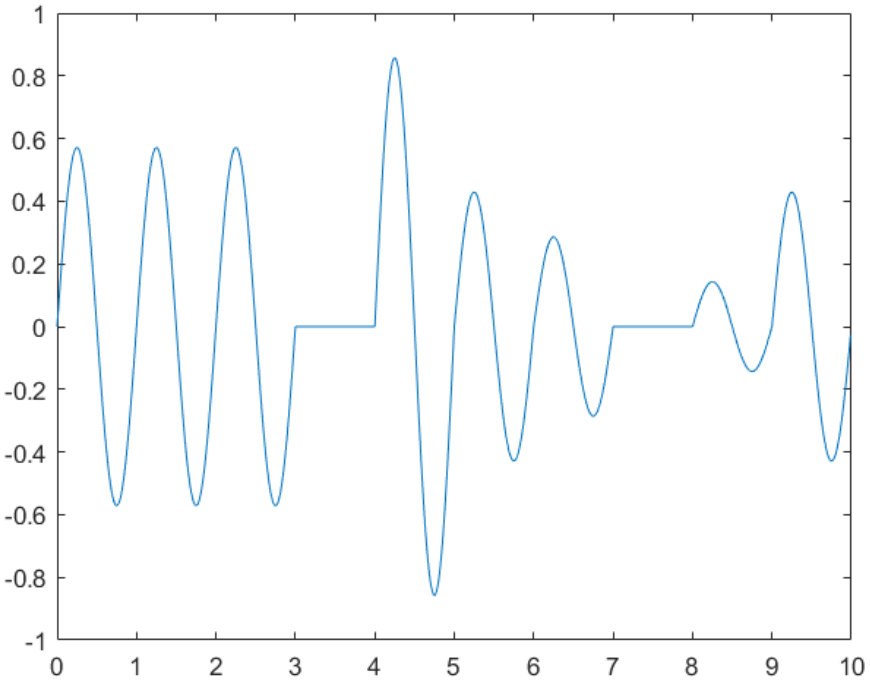
Bit Rate = 1



Bit Rate = 2



Bit Rate = 3



تمرین ۱-۴)

در تابع decoding_amp ، ابتدا پیام باینری را از سیگنال استخراج می کنیم .

یعنی ثانیه به ثانیه سیگنال را جدا کرده و با $2\sin(2\pi t)$ ، correlation میگیریم .

عدد حاصل از correlation ، همان عدد ثابت بیان شده در تمرین ۱-۲ است .

از آن رابطه ما عدد ثابت و bit rate را داریم ، پس عدد decimal نیز به دست می آید . حال اعداد decimal به دست آمده را به عدد باینری متناظر تبدیل میکنیم و به انتهای یک آرایه که کل پیام را تشکیل می دهد اضافه می کنیم .سپس این ارقام باینری را ۵ بیت ۵ بیت جدا کرده و درون mapset جست و جو میکنیم و کاراکتر متناظر را پیدا می کنیم .از کنار هم قرار دادن این کاراکتر ها ، پیام رمزگذاری شده به دست می آید .

اسکریپت مربوط به این تابع :

```
function decoded_message = decoding_amp(coded_sig,bit_rate)

    fs = 100;
    decoded_message = [];
    mapset = create_mapset();

    % Calculating correlations and extracting binary message
    bin_message = [];
    thresholds=[];
    for k=1:(2^bit_rate)-1
        thresholds(k) = ((2*k)-1)/(2*((2^bit_rate)-1));
    end
    t = 0:(1/fs):1-(1/fs);
    ref = 2*sin(2*pi*t);
    for j=0:(length(coded_sig)/fs)-1
        to_check = coded_sig((j*fs)+1:((j+1)*fs));
        corr = 0.01*sum(to_check.*ref);
        initial_corr = thresholds(1);
        for t=1:length(thresholds)
            if corr<=thresholds(t)
                corr = ((initial_corr*2*((2^bit_rate)-1))-t)/((2^bit_rate)-1);
                break;
            elseif t==length(thresholds)
                corr = 1;
            else
                initial_corr = thresholds(t+1);
            end
        end
        bin_message = [bin_message dec2bin(round(corr*((2^bit_rate)-1)),bit_rate)];
    end

    % Turning the binary message into string
    for p = 1:5:length(bin_message)-4
        message_to_check = bin_message(p:p+4);
        message_to_check = mapset{1,bin2dec(message_to_check)+1};
        decoded_message = [decoded_message message_to_check];
```

کد main مربوط به این ۴ تمرین :

```
close all;
clc;
clear;

% 1- Creating the mapset
mapset = create_mapset();

% 2- Coding the message
fs = 100;
message = 'signal';
bit_rate = 3;
coded_signal = coding_amp(message,bit_rate);

% 3- Plotting the result
t = 0:(1/fs):(length(coded_signal)/fs)-(1/fs);
figure
plot(t,coded_signal)

% 3- Decoding the coded signal
decoded_message = decoding_amp(coded_signal,bit_rate)
```

خروجی تابع به ازای bit rate های مختلف (۱و۲و۳) :

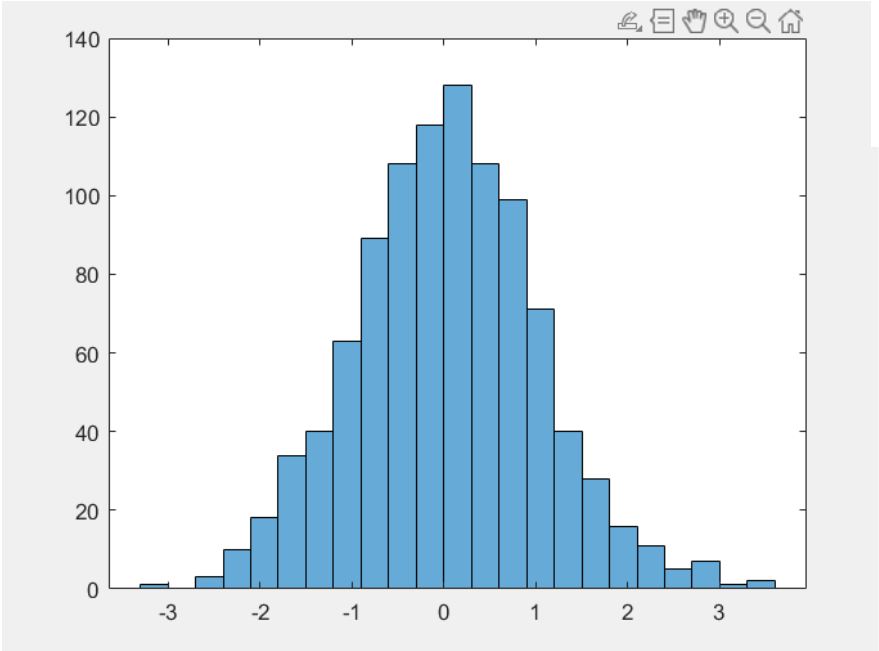
```
decoded_message =

    'signal'
```

تمرین ۱-۵)

```
% 5- Noise
column = length(coded_signal);
noise=randn(1,column);
figure
histogram(noise)
Mean=mean(noise)
variance=var(noise)
```

با استفاده از کد بالا یک نویز ایجاد کردیم(با استفاده از دستور randn)

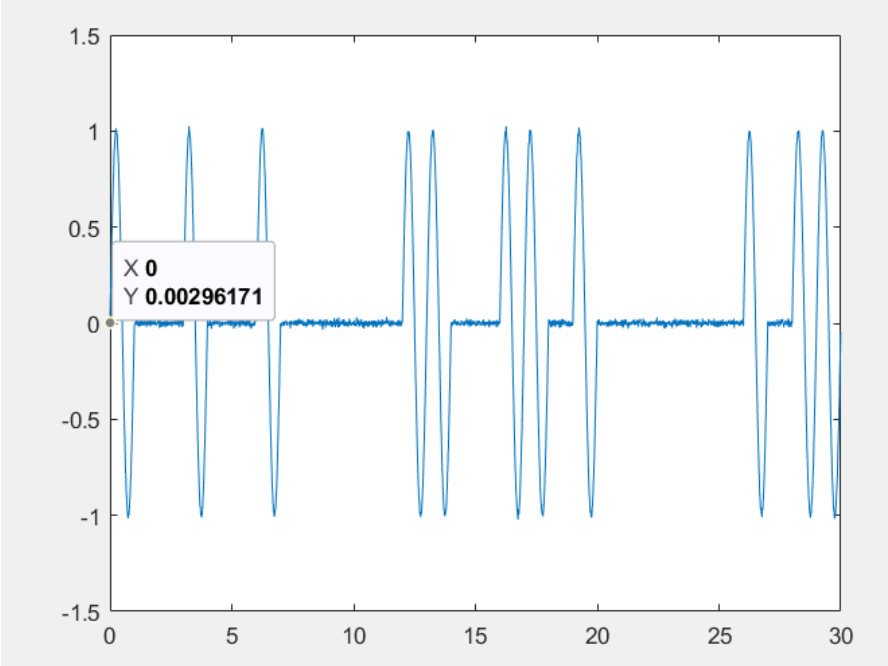
سپس هیسوگرام را میکشیم تا چک کنیم نویز گوسی باشد. همچنین میانگین و واریانس را هم چک میکنیم که به ترتیب ۰ و ۱ باشند که هستند.



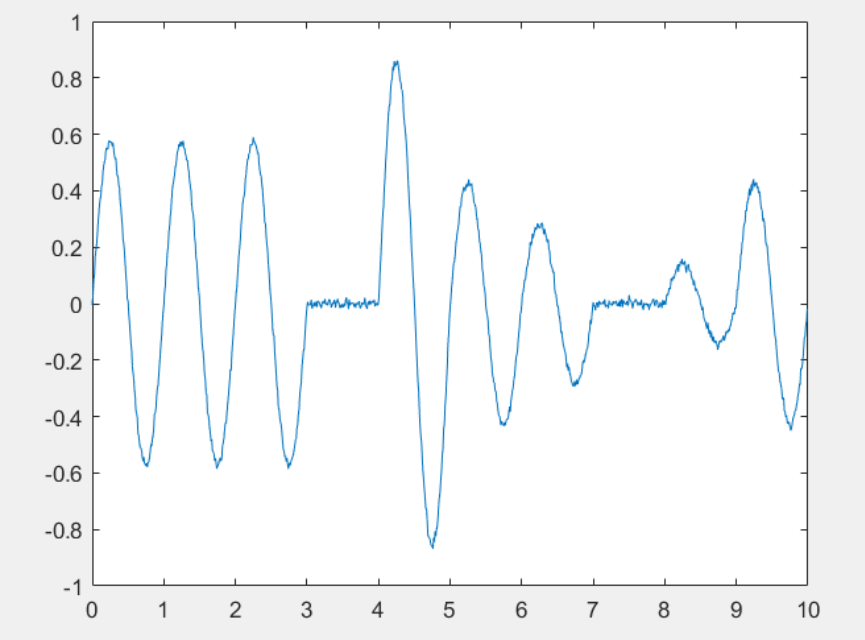

```
Mean =

    0.0369


variance =

    0.9972
```

تمرین ۱-۶)

نویز با bit-rate=1 و نویز با واریانس ۰,۰۰۰۱

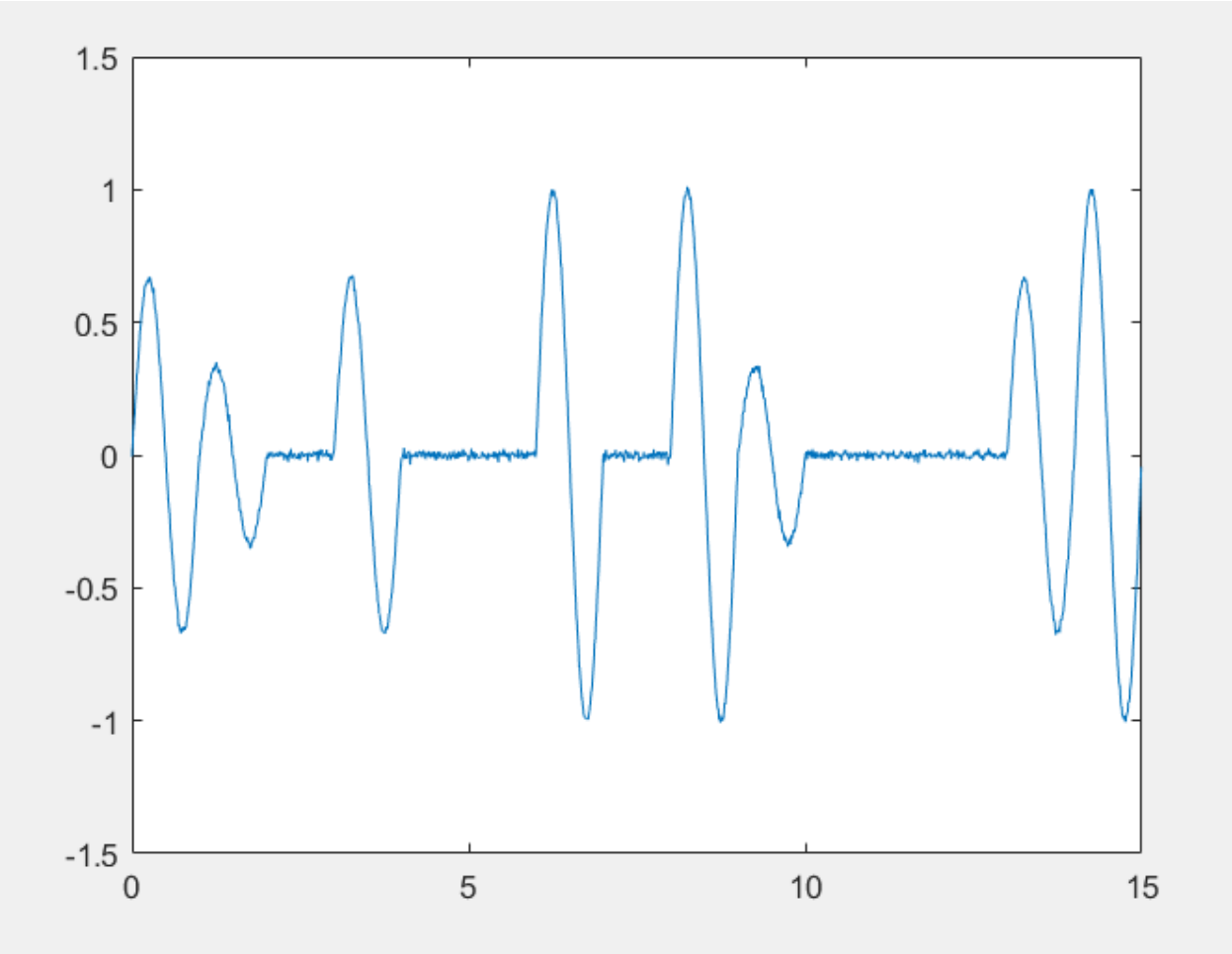


نویز با bit-rate=2 و نویز با واریانس ۰,۰۰۰۱



نویز با bit-rate=3 و نویز با واریانس ۰,۰۰۰۱

میبینیم که سرعت ۳ کمی نویز قابل مشاهده است.




```
% 6- change variance of noise
m = 0.01;
noisy = m * noise;
coded1 = noisy + coded_signal;
decoded = decoding_amp(coded1,bit_rate)
figure
plot(t,coded1)
```

تمرین ۱-۷)

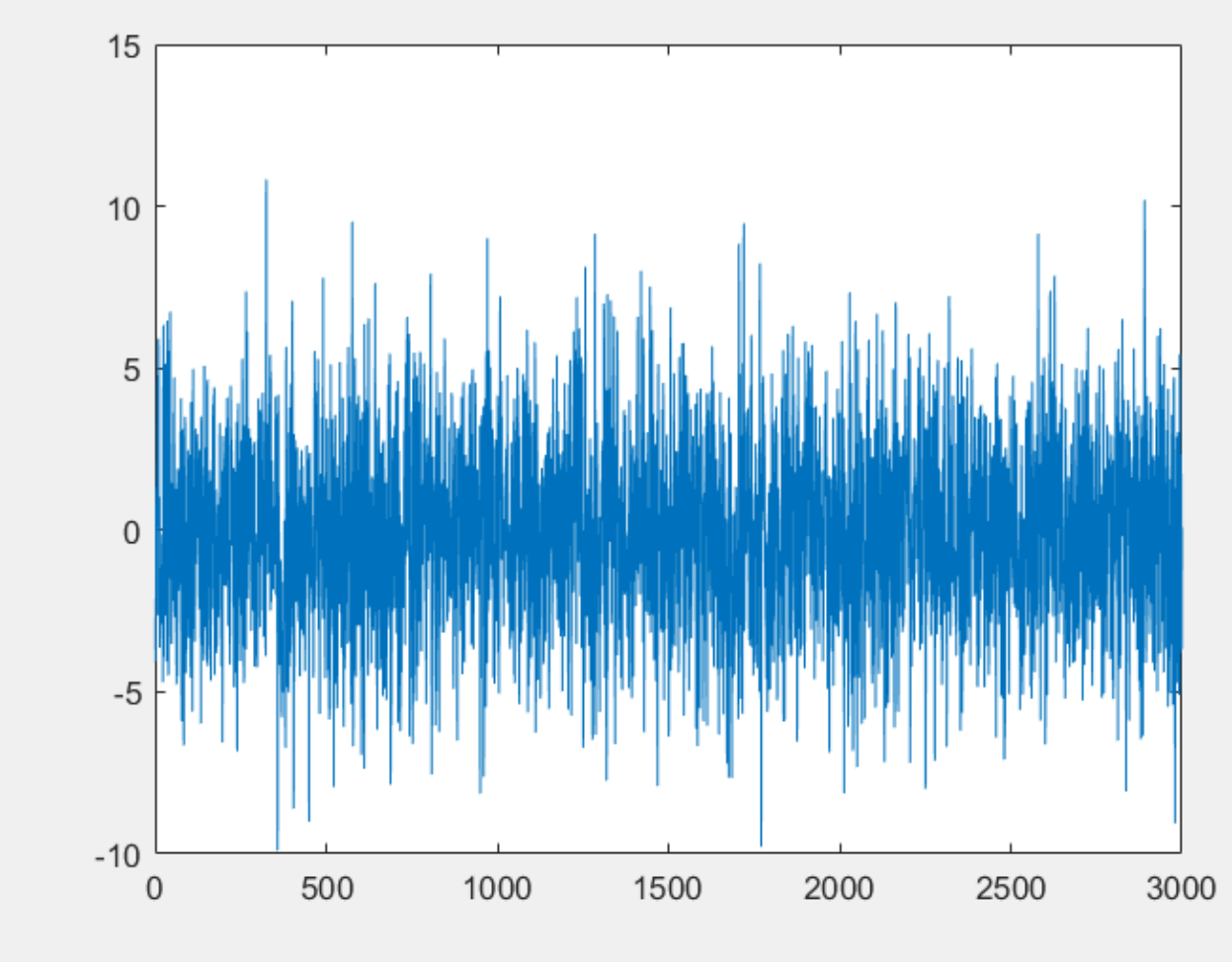
با یک حلقه به صورت زیر نویز تدریجی میدهیم. اگر اشتباه بود پیام شکل را نشان میدهد. طبق شکل ها در bit-rateهای بالاتر نویز سریع تر اثر کرده و n کمتری خروجی اشتباه میسازد.

```
%7 -sligth change in noise
n=0.01;
for i=1:500
    noisy=n*noise;
    coded=noisy+coded_signal;
    decoded=decoding_amp(coded,bit_rate);
    if ~strcmp(decoded, message)
        variance=n*n;
        disp(variance)
        disp(decoded)
        figure
        plot(coded)
        break
    end
    n=n+0.01;
```

هر چقدر سرعت بالاتر است حساسیت به نویز بیشتر است.

تمرین۱-۸)

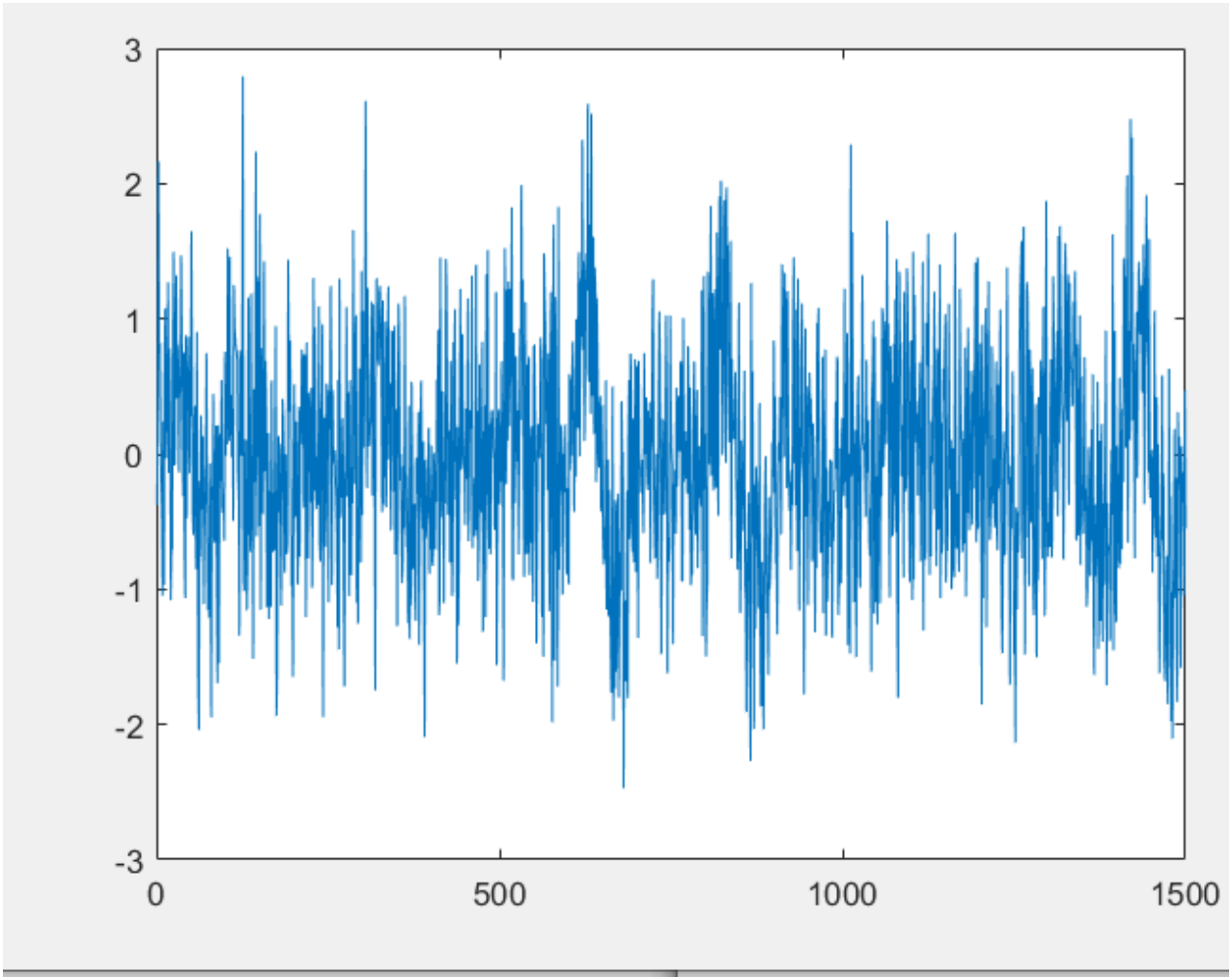
در bit-rate = 1 در واریانس ۸,۸۴ این اتفاق میفتد.



mmand Window

```
decoded =

    'signal'

    8.8804

sig!al
```

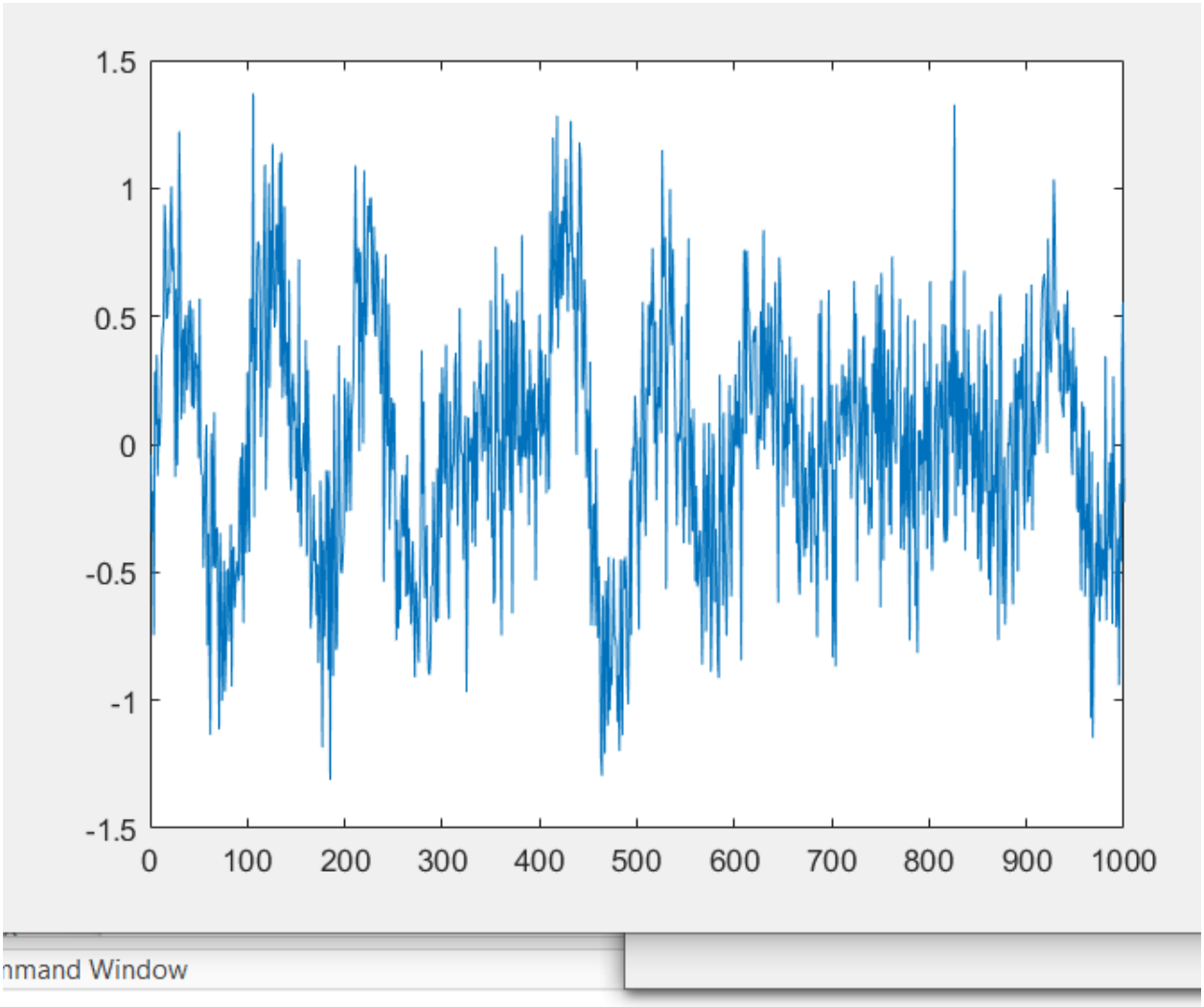
در bit-rate = 2 در واریانس 0.54این اتفاق می افتد.



mmand Window

```
decoded =

    'signal'

    0.5476

kignal
```

در bit-rate = 3 در واریانس ۰,۱۰۸ این اتفاق می افتد.



nmand Window

```
decoded =

    'signal'

    0.1089

signcl
```

تمرین ۱-۹)

در واقع، زمانی که ما دامنه سیگنال ارسالی را افزایش می‌دهیم، به طور طبیعی آستانه‌های تشخیص ما نیز افزایش می‌یابد. این به معنی است که سیستم ما توانایی بهتری در تفکیک سیگنال اصلی از نویز خواهد داشت. به عبارت دیگر، با افزایش توان مصرفی و ارسال سیگنال با دامنه بالاتر، سیگنال ما در برابر نویز مقاومت بیشتری خواهد داشت و دقت تصمیم‌گیری‌ها بهبود می‌یابد. این به این دلیل است که یک سیگنال با دامنه بالاتر می‌تواند از پس نویزهای محیطی و تداخلات بهتر برآید، و تشخیص‌ها و شناسایی‌ها را دقیق‌تر کند.

اضافه بر این، افزایش دامنه سیگنال به معنی بهبود نسبت سیگنال به نویز است، که یکی از معیارهای کلیدی در سیستم‌های مخابراتی برای تعیین کیفیت و قابلیت اطمینان ارتباط است. بهبود SNR به طور مستقیم منجر به کاهش خطاهای احتمالی در انتقال داده‌ها و افزایش قابلیت اطمینان ارتباط می‌شود. بنابراین، در بسیاری از موارد افزایش دامنه و توان سیگنال ارسالی به عنوان یک راهکار موثر برای بهبود عملکرد سیستم‌های مخابراتی و مقاوم‌سازی آن‌ها در برابر نویز و تداخلات محیطی در نظر گرفته می‌شود.

تمرین ۱-۱۰)

در صورت عدم وجود نویز در سیستم کدینگ دامنه، بیت‌ریت می‌تواند به طور تئوری به بی‌نهایت نزدیک شود. به عبارت دیگر، در یک محیط بدون نویز، محدودیتی برای افزایش بیت‌ریت وجود ندارد. البته، در عمل موارد دیگری مانند محدودیت‌های سخت‌افزاری، پهنای باند کانال و محدودیت‌های فیزیکی سیستم‌ها تاثیرگذار خواهند بود.

۱. پهنای باند کانال: حتی در صورت عدم وجود نویز، پهنای باند کانال محدودیتی برای افزایش بیت‌ریت ایجاد می‌کند. ظرفیت کانال بر اساس پهنای باند و تکنیک‌های مدولاسیون استفاده شده تعیین می‌شود.

۲. محدودیت‌های سخت‌افزاری:سرعت پردازش و ذخیره‌سازی داده‌ها نیز نقشی مهم در تعیین بیت‌ریت دارد. حتی اگر نویزی نباشد، سخت‌افزار باید بتواند با سرعت بالای بیت‌ریت هماهنگ شود.

تمرین ۱-۱۱)

در اصل، اهمیت دامنه سیگنال ارسالی به مراتب بیشتر از آن است که سیگنال دریافتی با چه سیگنالی برای عملیات correlation یا ضرب شود. سیگنال ارسالی باید قدرت کافی داشته باشد تا در مقابل نویز محیطی مقاومت کند.

به این ترتیب، اگر سیگنال ارسالی دارای دامنه کافی نباشد، حتی اگر سیگنال دریافتی را با اعداد مختلف ضرب کنیم، در نهایت تأثیر نویز کاهش نخواهد یافت. به عبارتی دیگر، زمانی که سیگنال از فرستنده خارج می‌شود، اگر نویز محیطی از حدی بیشتر باشد، سیگنال اصلی دیگر قابل تشخیص نخواهد بود. اعداد ۲ و ۱۰ که برای ضرب استفاده شده‌اند، تنها برای راحتی در انجام عملیات ریاضی و بدون تغییر در اصول پایه‌ای تأثیر نویز بر سیگنال است.

تمرین ۱-۱۲)

سرعت اینترنت‌های ADSL معمولاً بین ۸ تا ۱۶ مگابیت بر ثانیه است، که معادل حدود ۸ میلیون بیت در ثانیه می‌باشد. در این تمرین، ما با سرعت‌هایی مانند ۱ بیت در ثانیه، ۲ بیت در ثانیه و ۳ بیت در ثانیه داده‌ها را ارسال کردیم. این تفاوت عظیم در سرعت ارسال داده‌ها نشان می‌دهد که تمرین ما در مقایسه با سرعت‌های واقعی اینترنت بسیار آهسته‌تر است.

این مثال می‌تواند به وضوح نشان دهد که در مقیاس‌های کوچک‌تر و سرعت‌های پایین‌تر، چگونه مشکلات انتقال داده‌ها و تأثیر نویز بررسی و تحلیل می‌شود، اما در دنیای واقعی اینترنت و ارتباطات پر سرعت، مسائل بسیار پیچیده‌تر هستند.