گزارش تمرین کامپیوتری چهارم سیگنال سیستم دکتر اخوان

پاییز 1403

محمدمهدی صمدی - 810101465

سپهر جمالی - 810101400

بخش اول:

1_1) تابع زیر mapset گفته شده را می‌سازد.

```
function mapset=create_mapset(chars)
    num_chars = length(chars);
    coded_binary_length = ceil(log2(num_chars));
    mapset=cell(2, num_chars);
    for i=1:num_chars
        mapset{1,i}=chars(i);
        mapset{2,i}=dec2bin(i-1, coded_binary_length);
    end
end
```

2_1) در این بخش، تابع coding amp را توضیح می‌دهیم.

در ابتدا معادل هر کاراکتر را در مپست پیدا می‌کنیم. اگر کاراکتری در مپست نبود پیغامی چاپ می‌شود و بعد از آن می‌توان دو روش را پیش گرفت:

- از تابع بازگردیم و ارور دهیم.
- حرف ناشناخته با را حرف قرارداده خاصی جایگزین کنیم. من با '!' کردم.

این که کدام روش انجام شود بستگی به فلگ ignore not founds دارد. اگر 1 باشد روش دوم و در غیر این صورت روش اول انجام می‌شود.

```
function [valid, coded_message] = coding_amp(message, rate, mapset, ignore_not_founds)
    valid = 1;
    coded_message = [];
    message_len = length(message);
    fs = 100;

    found_all_chars = 1;
    for i=1:message_len
        found_char = 0;
        for j=1:length(mapset)
            if strcmp(message(i), mapset(1, j)) == 1
                found_char = 1;
                coded_message = cat(2, coded_message, mapset(2, j));
            end
        end
        if ~found_char
            fprintf("did not find %c in the mapset\n", message(i));
            if ignore_not_founds
                coded_message = cat(2, coded_message, '11110');
            end
            found_all_chars = 0;
        end
    end
    if ~found_all_chars
        if ~ignore_not_founds
            valid = 0;
            return;
        end
    end
```

در ادامه پیام باینری code شده به تکه های rate بیتی تقسیم می‌شود تا هر تکه در یک بازه زمانی 0.01 ثانیه‌ای سیگنال ظاهر شود.

برای این مینیمم گرفته شده که اندیس راست بازه از سایز بیشتر نشود. در حالتی این اتفاق می‌افتد که نمایش باینری پیام ورودی به rate تقسیم پذیر نباشد که این شرط از قبل چک شده است.

```
    coded_message = cell2mat(coded_message);
    coded_message_len = length(coded_message);
    for l = 1 : rate : coded_message_len
        % l and r are start and end indices of current segment
        % min is to prevent exceeding size of the message
        r = min(l + rate - 1, coded_message_len);
        segmented_message{floor((l-1)/rate) + 1} = coded_message(l:r);
    end
```

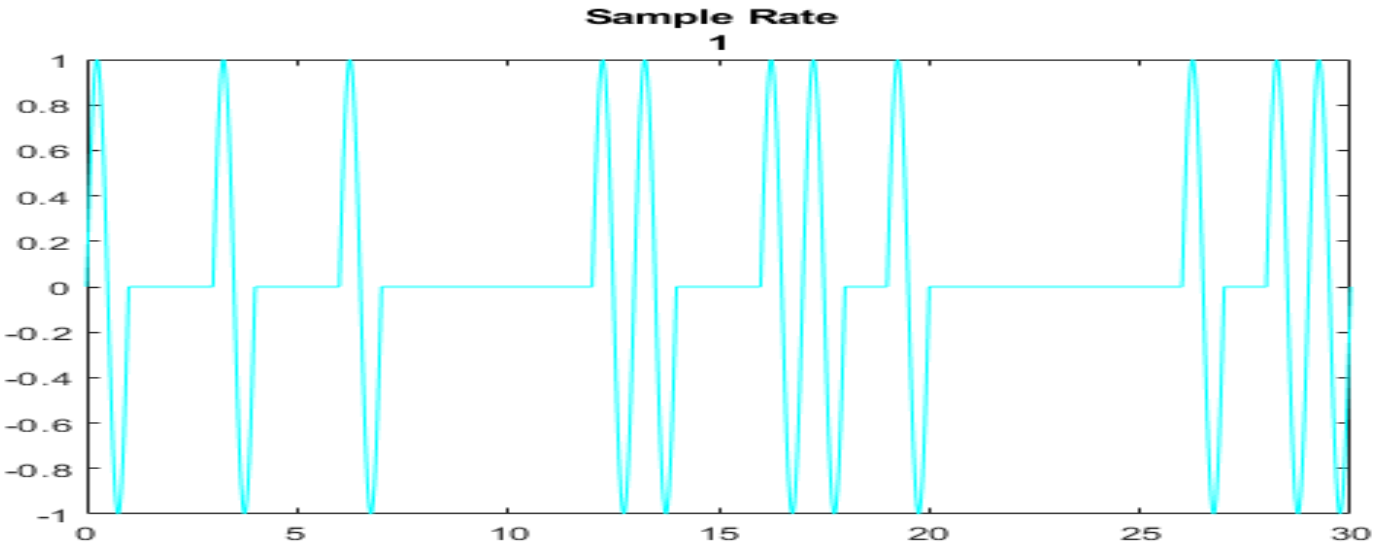
در ادامه به هر rate بیت متوالی یک ضریب برای تابع سینوسی $sin(2\pi t)$ اختصاص می‌دهیم تا با کنار هم قرار دادن آن‌ها سیگنال خروجی را بسازیم. اگر هر rate بیت متوالی ناشناخته بود اروری داده می‌شود.

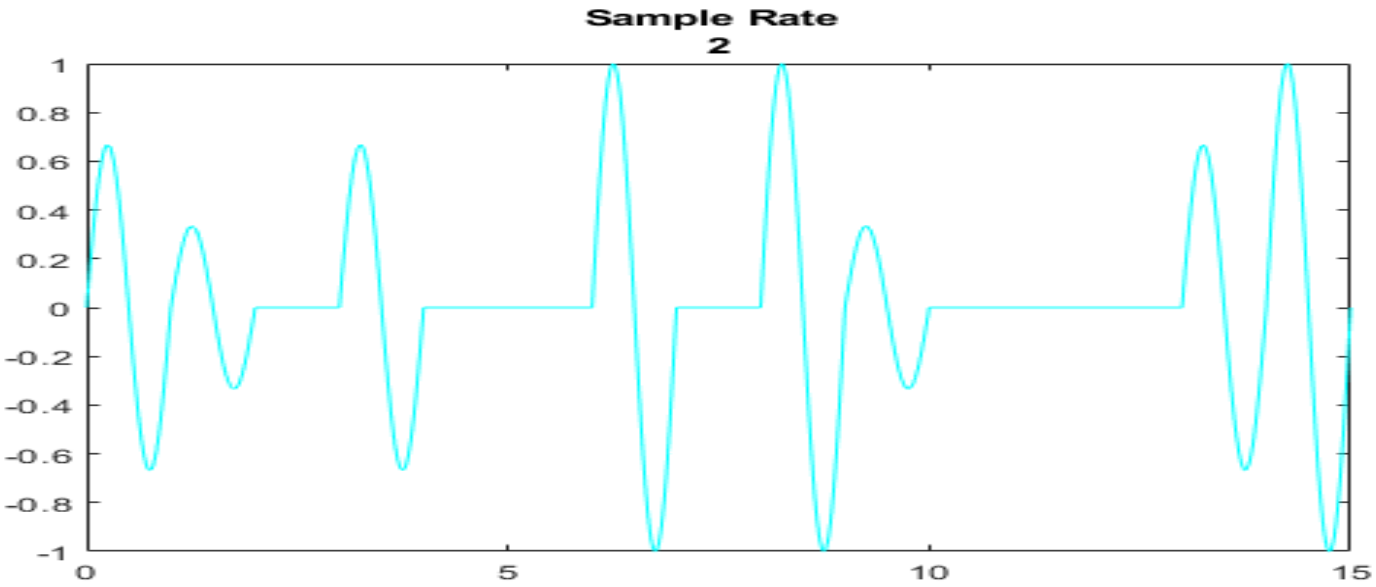
```
58      encoded_message_coef = [];
59      found_all_bits = 1;
60      for i=1:numof_bits
61          found_bit = 0;
62          for j=1:2^rate
63              if strcmp(coded_message(1,i), binary_codes(1,j)) == 1
64                  found_bit = 1;
65                  encoded_message_coef = horzcat(encoded_message_coef, coefficients(1, j));
66              end
67          end
68          if ~found_bit
69              fprintf("did not find %s in bits\n", char(coded_message(1, i)));
70              found_all_bits = 0;
71          end
72      end
73      if ~found_all_bits
74          return;
75      end
```

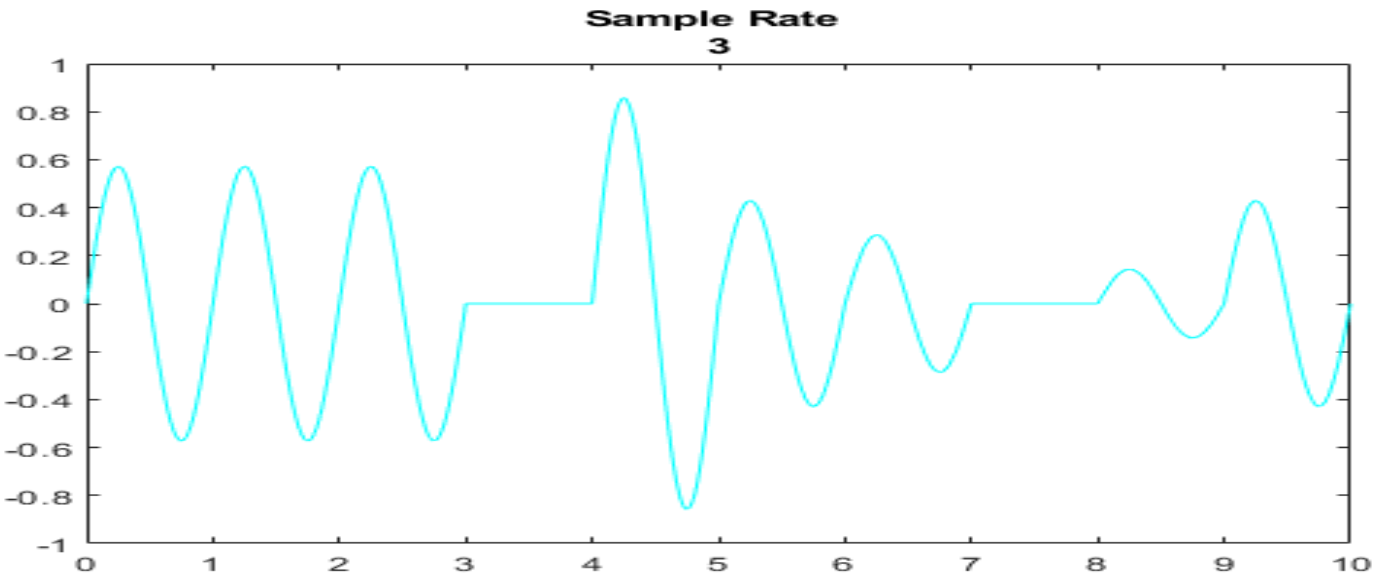
در انتهای این بخش سیگنال ساخته و پلات می‌شود.

```
77        coded_message=zeros(numof_bits, fs);
78        for i=1:numof_bits
79                coded_message(i,:)=encoded_message_coef(1,i).*sin(2*pi*t(i,:));
80        end
81
82        for startindex=1:numof_bits
83            plot(t(startindex,:),coded_message(startindex,:),'r');
84            hold on;
85        end
86        title(["Rate = " , int2str(rate)]);
87    end
```

3_1) سه خروجی داده شده برای پیام "signal" به صورت زیر می‌باشند.







4_1) در این بخش، تابع decode کردن را توضیح می‌دهیم.

در ابتدا correlation سیگنال ورودی را با تابع $sin(2\pi t)$ می‌گیریم. ضریب $\frac{1}{100}$ برای این است که راحت‌تر شدن کار است.

```
function decoded_message=decoding_amp(coded_message, rate, mapset)
    corr = [];
    len = size(coded_message);
    fs = 100;
    t=zeros(len(1),100);
    coded_binary_length = ceil(log2(length(mapset)));
    for i=1:len(1)
        t(i,:)=linspace(i-1, i, fs);
    end
    for l=1:len(1)
        corr_with_sin = 0.01*sum((2*sin(2*pi*t(l,:))).*(coded_message(l,:)));
        corr = horzcat(corr, corr_with_sin);
        corr = double(corr);
    end
    coefficients= linspace(0, 1, 2^rate);
    binary_codes = cell(1, 2^rate);
    for i=1:2^rate
        binary_codes{i} = dec2bin(i-1, rate);
    end
```

در ادامه rate تا rate تا بیت از سیگنال خارج می‌کنیم و در decoded bits می‌ریزیم. برای این کار فاصله correlation هر بخش با تابع $sin(2\pi t)$ را از هر محدوده چک می‌کنیم و آن را به نزدیک‌ترین محدوده نسبت می‌دهیم. بعد از آن تمام این بیت‌ها را به هم می‌چسبانیم تا تشکیل یک استرینگ باینری دهند.

```
    decoded_bits = [];
    half_coeffs_diff = (coefficients(1, 2) - coefficients(1, 1)) / 2;
    for i=1:len(1)
        for idx=1:2^rate
            if (abs(corr(1, i) - coefficients(1, idx))) <= half_coeffs_diff
                decoded_bits = [decoded_bits binary_codes(1, idx)];
            end
        end
    end

    cat_bits = '';
    for i=1:length(decoded_bits)
        cur_rate_bits = cell2mat(decoded_bits(i));
        for j=1:rate
            cat_bits(end+1) = cur_rate_bits(j);
        end
    end
```

در نهایت باید 5 تا 5 تا بیت‌ها را جدا کنیم و در مپ‌ست بگردیم تا کاراکتر معادلش را پیدا کنیم.

```
39          cat_len = length(cat_bits);
40          decoded_message = '';
41          for i=1:coded_binary_length:cat_len
42              binary_coded_char = cat_bits(i:i+coded_binary_length-1);
43              for j=1:length(mapset)
44                  if binary_coded_char == cell2mat(mapset(2,j))
45                      decoded_message(end+1) = mapset{1, j};
46                  end
47              end
48          end
```

برای اجرای توابع بالا در کنار هم script زیر نوشته شده است.

اگر تعداد بیت‌های مسیج بعد از تبدیل به حالت باینری بر rate بخش‌پذیر نباشد، یعنی در آخر پیام یک تعداد بیت کمتر rate باقی می‌مانند که نمی‌توانیم آن‌ها را code کنیم پس ارور می‌دهیم.

در ادامه تابع coding amp همان‌طور که قبلا توضیح داده شد کاراکترهای ناشناخته را هندل می‌کند.

در نهایت نیز تابع decoding amp سیگنال را گرفته و decode می‌کند.

```
1   clc;
2   clearvars;
3   close all;
4
5   chars ='abcdefghijklmnopqrstuvwxyz .,;!"';
6   mapset = create_mapset(chars);
7   message = 'spread your wings and fly away... fly away... far away';
8   rate = 3;
9
10  if rem(length(message) * ceil(log2(length(chars))), rate) ~= 0
11      fprintf("size of binary coded message is not divisble by the rate\n");
12      return;
13  end
14
15  ignore_not_founds = 1;
16  [valid, coded_signal] = coding_amp(message, rate, mapset, ignore_not_founds);
17  if ~valid
18      fprintf("invalid input\n");
19      return;
20  end
21
22  decoded_message = decoding_amp(coded_signal, rate, mapset);
23  disp(decoded_message)
24
```

Command Window

```
spread your wings and fly away... fly away... far away
>>
```

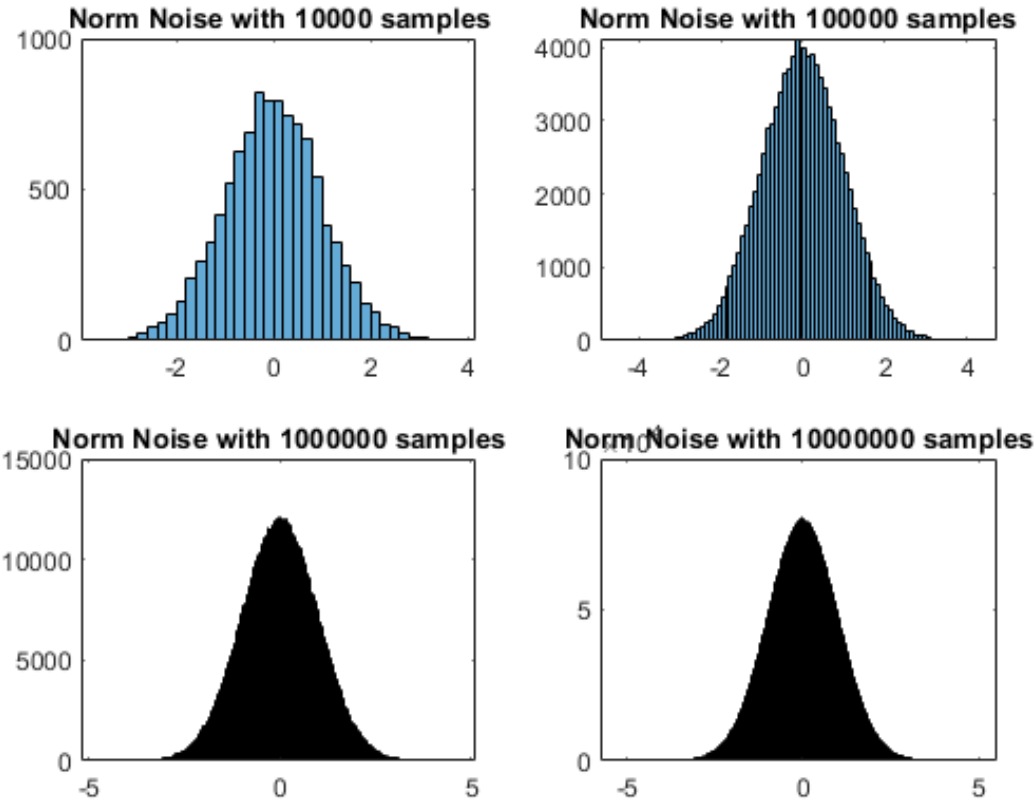
(5_1

کد زیر نویز گوسی با تعداد سمپل مختلف می‌سازد و برای نشان دادن ویژگی‌های گفته شده میانگین، انحراف معیار و هیستوگرام هر کدام را نشان می‌دهد.

```
clearvars;
close all;

figure;

for i=1:4
    subplot(2, 2, i);
    noise = randn(1, 10^(i+3));
    mu= mean(noise);
    std = sqrt(var(noise));
    num_samples = num2str(10^(i+3));
    disp(['Mean of Gaussian noise with ', num_samples, ' samples: ', num2str(mu)]);
    disp(['Standard deviation of Gaussian noise with ', num_samples, ' samples: ', num2str(std)]);

    histogram(noise);
    title(['Norm Noise with ', num_samples, ' samples']);
end
```

Command Window

```
Mean of Gaussian noise with 10000 samples: -0.0087037
Standard deviation of Gaussian noise with 10000 samples: 1.0012
Mean of Gaussian noise with 100000 samples: -0.0031117
Standard deviation of Gaussian noise with 100000 samples: 1.003
Mean of Gaussian noise with 1000000 samples: -0.00067353
Standard deviation of Gaussian noise with 1000000 samples: 0.99962
Mean of Gaussian noise with 10000000 samples: -0.0001988
Standard deviation of Gaussian noise with 10000000 samples: 0.99999
```



6_1)

نتیجه سیگنال با نویز نرمال و انحراف معیار $\frac{1}{1000}$ برای سه rate مختلف در تصویر زیر است. در هر سه حالت به درستی decode می‌شود.

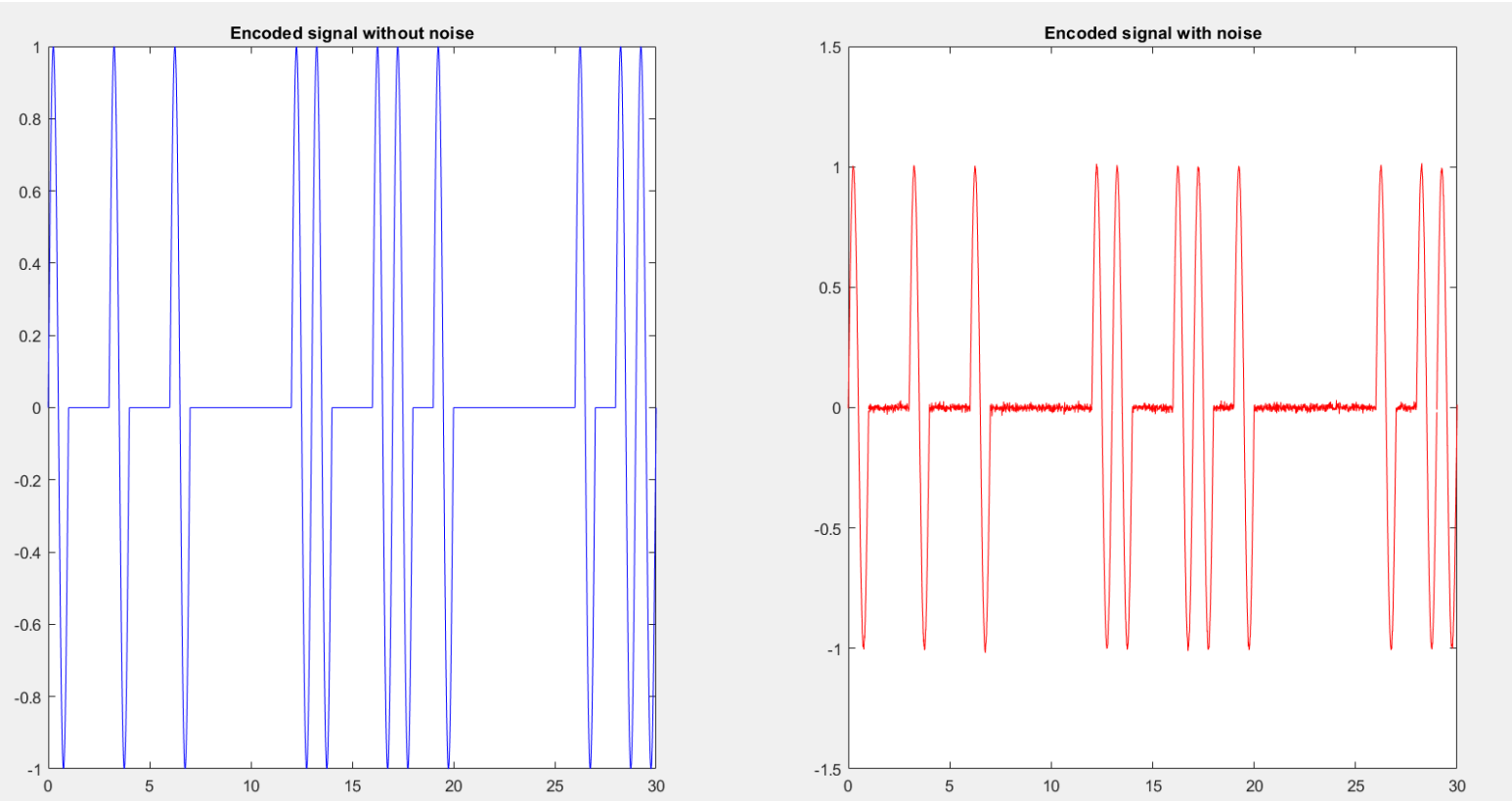
```
clc;
clearvars;
close all;

chars ='abcdefghijklmnopqrstuvwxyz .,;!"';
mapset = create_mapset(chars);
message = 'spread your wings and fly away... fly away... far away';
rate = 3;

if rem(length(message) * ceil(log2(length(chars))), rate) ~= 0
    fprintf("size of binary coded message is not divisble by the rate\n");
    return;
end

ignore_not_founds = 1;
[valid, coded_signal] = coding_amp(message, rate, mapset, ignore_not_founds);
if ~valid
    fprintf("invalid input\n");
    return;
end
std = 0.001;
noisy_signal = add_noise(coded_signal, std);
decoded_message = decoding_amp(noisy_signal, rate, mapset);


disp(['added normal noise (rate=', num2str(rate), '):', decoded_message])
```

Command Window

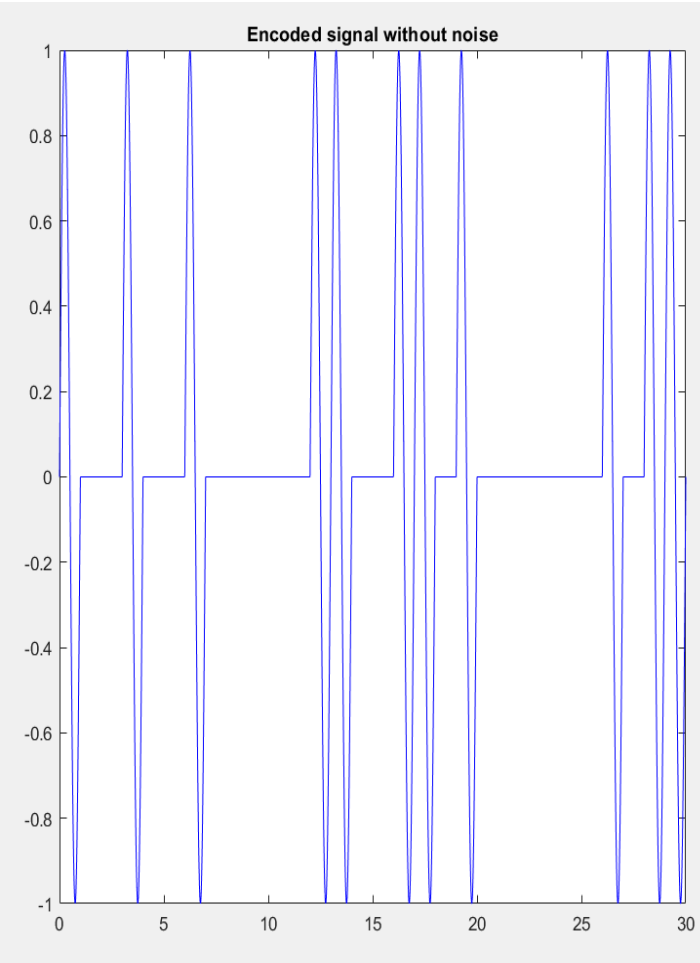
```
>> run_with_noise
added normal noise (rate=1):spread your wings and fly away... fly away... far away
>> run_with_noise
added normal noise (rate=2):spread your wings and fly away... fly away... far away
>> run_with_noise
added normal noise (rate=3):spread your wings and fly away... fly away... far away
>>
```

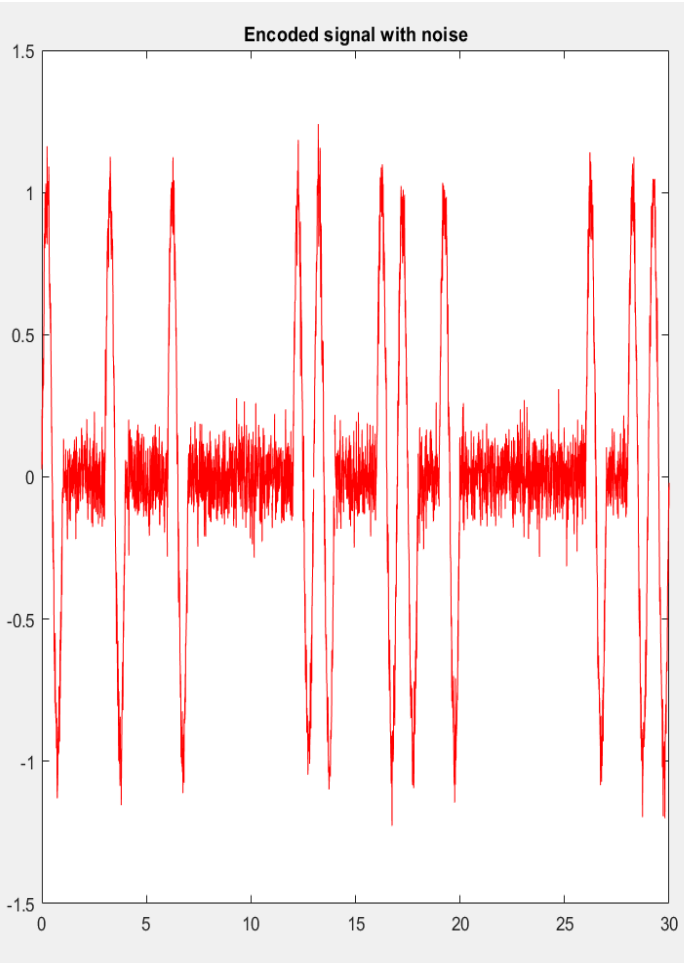
(7_1

Bit rate=1

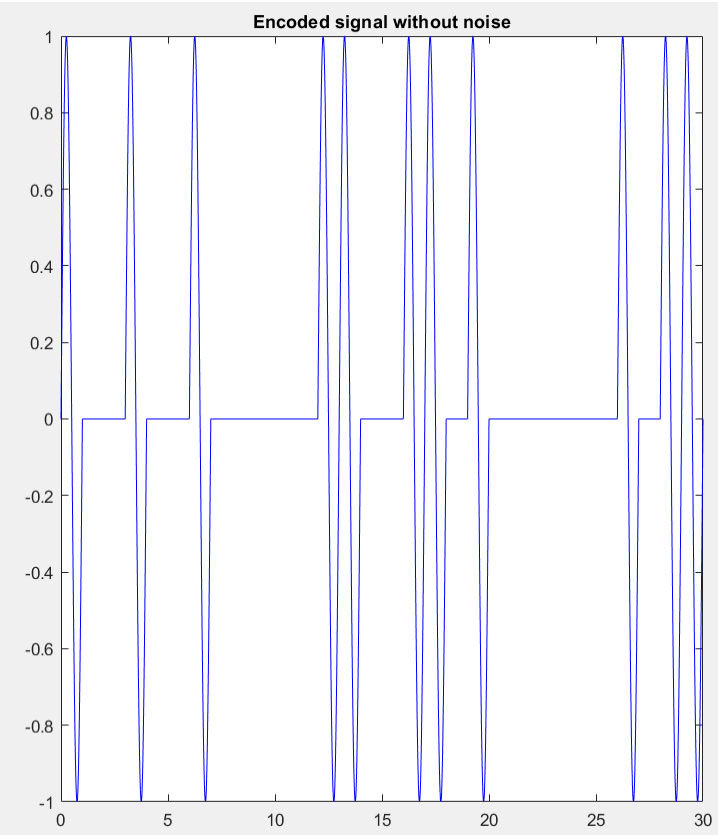


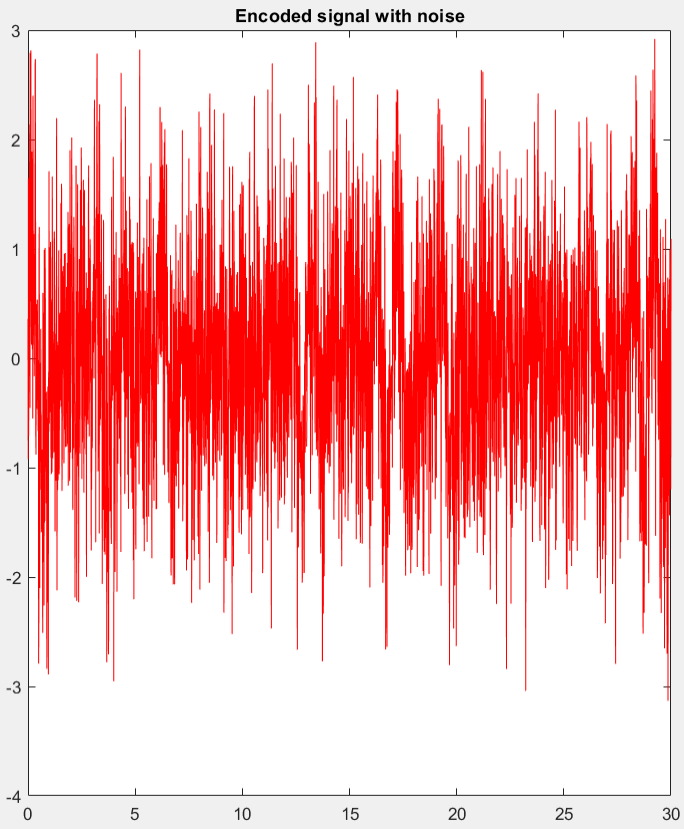
std=0.009


Encoded signal without noise
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
5
10
15
20
25
30


Encoded signal with noise
1.5
1
0.5
0
-0.5
-1
-1.5
0
5
10
15
20
25
30

std=0.09





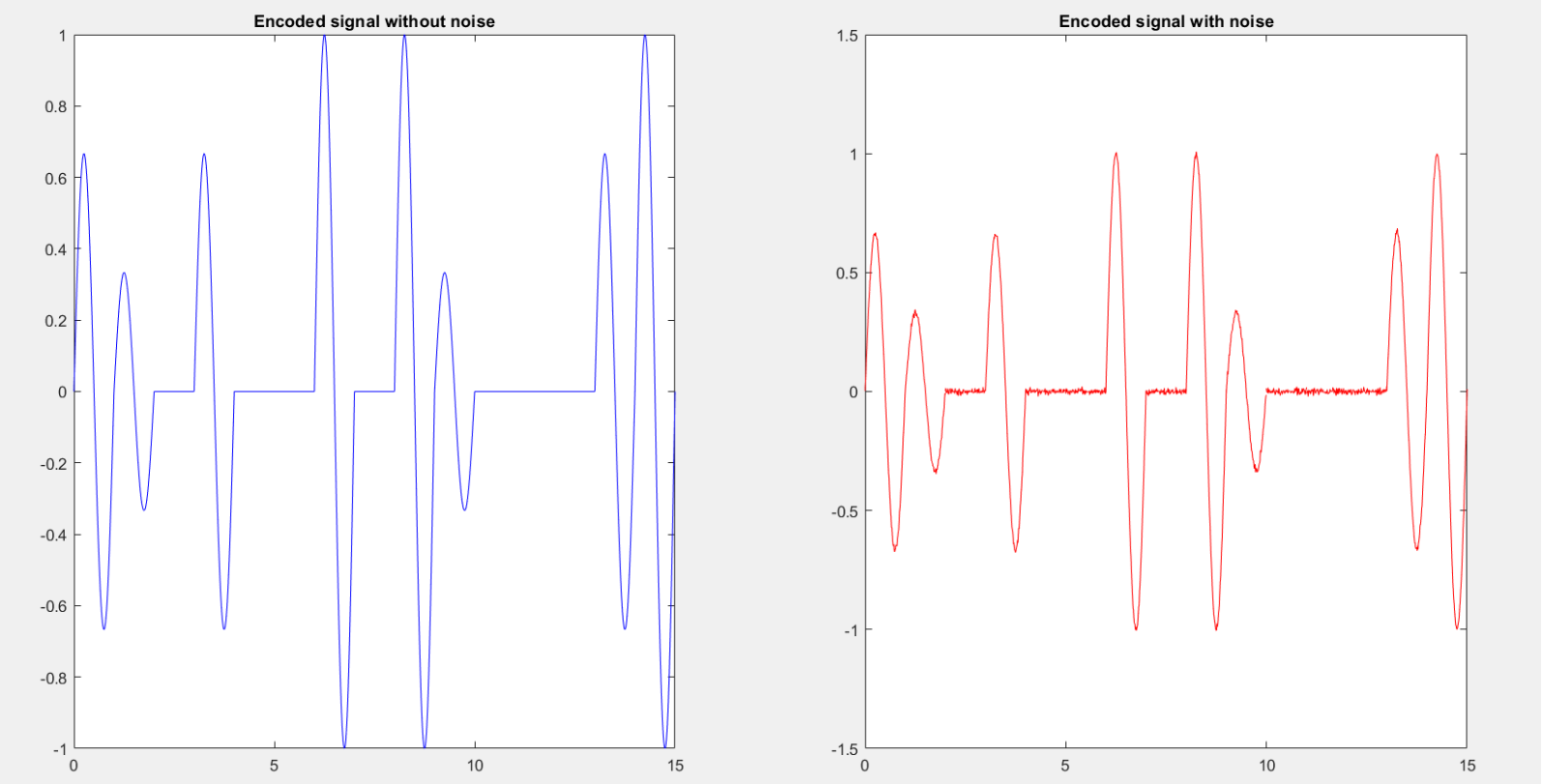
std=0.9

خروجی به ازای std های مختلف:

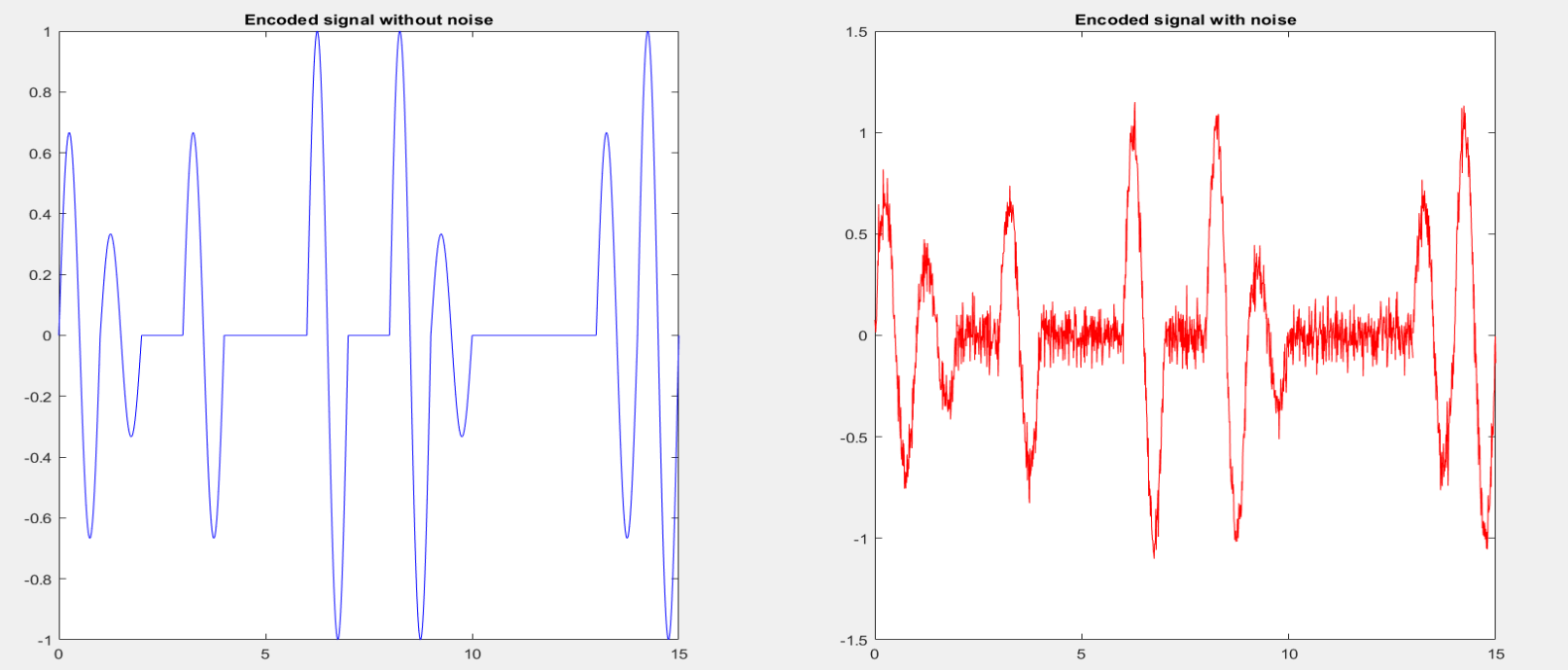
```
added normal noise (rate=1) (std=0.009):signal
added normal noise (rate=1) (std=0.09):signal
added normal noise (rate=1) (std=0.9):signal
```

خروجی در همه حالات درست می‌باشد.

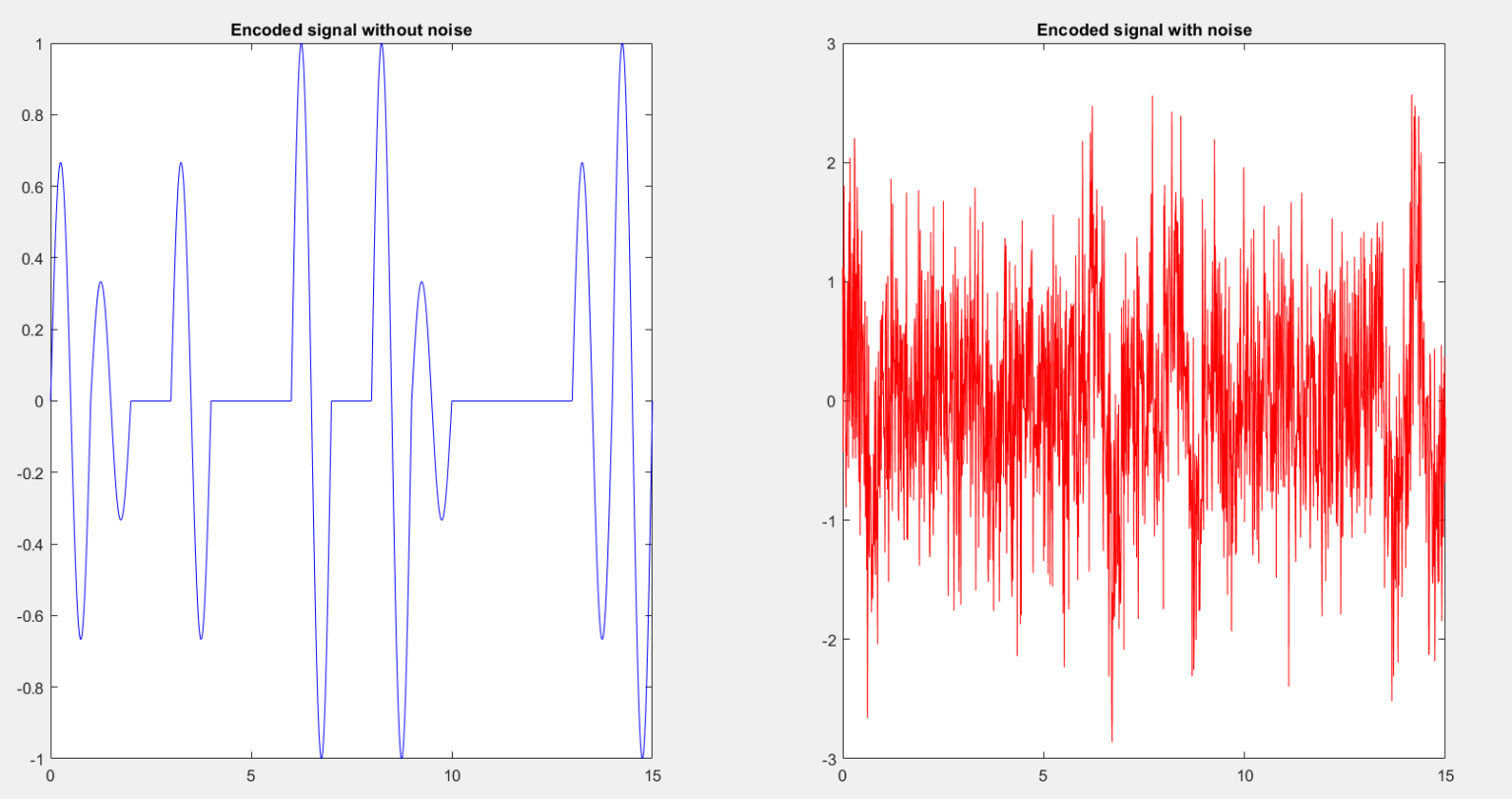
Bit rate=2



std=0.0075



std=0.075



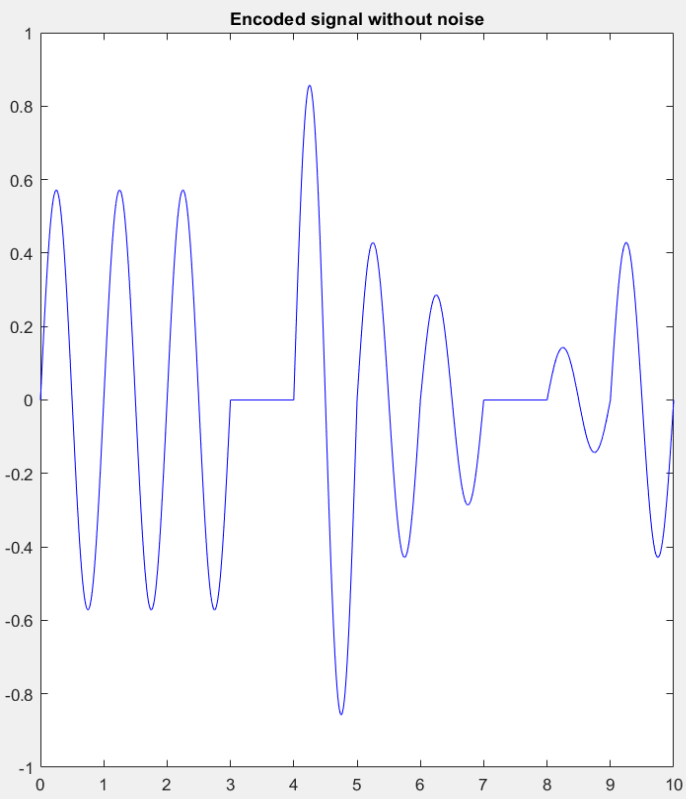
std=0.75

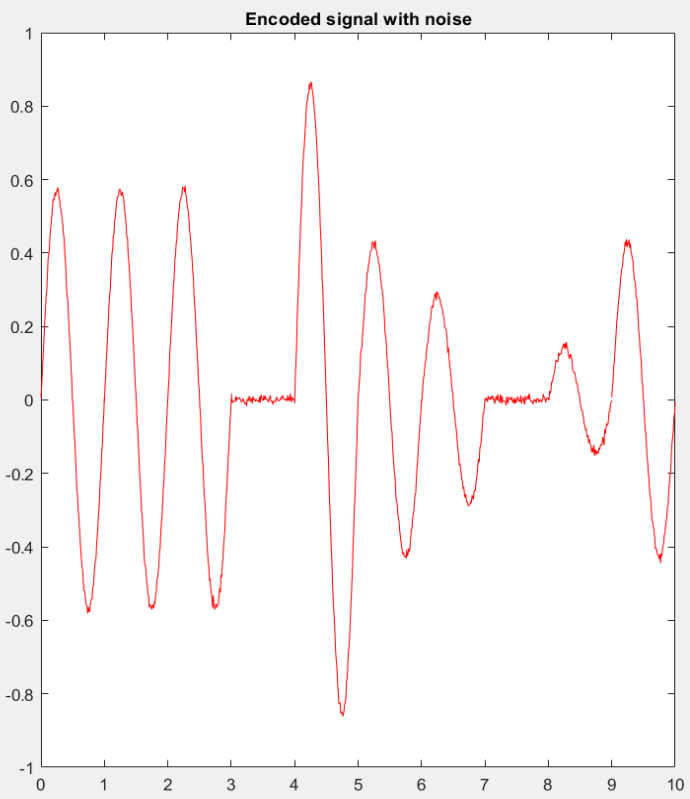
خروجی به ازای std های مختلف:

```
added normal noise (rate=2) (std=0.0075):signal
added normal noise (rate=2) (std=0.075):signal
added normal noise (rate=2) (std=0.75):yegnal
```

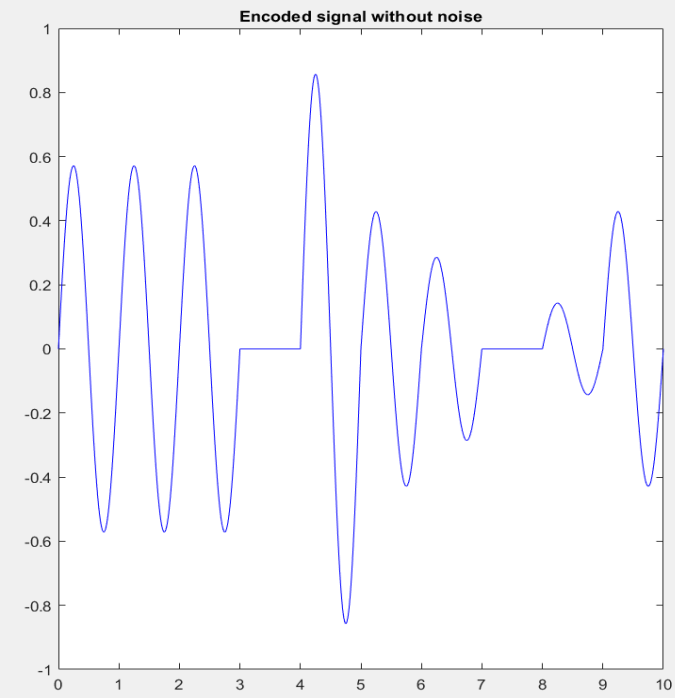
خروجی برای std=0.75 دچار مقدار کمی نویز می‌شود. (که البته با چند بار اجرا می‌توان به خروجی signal رسید)

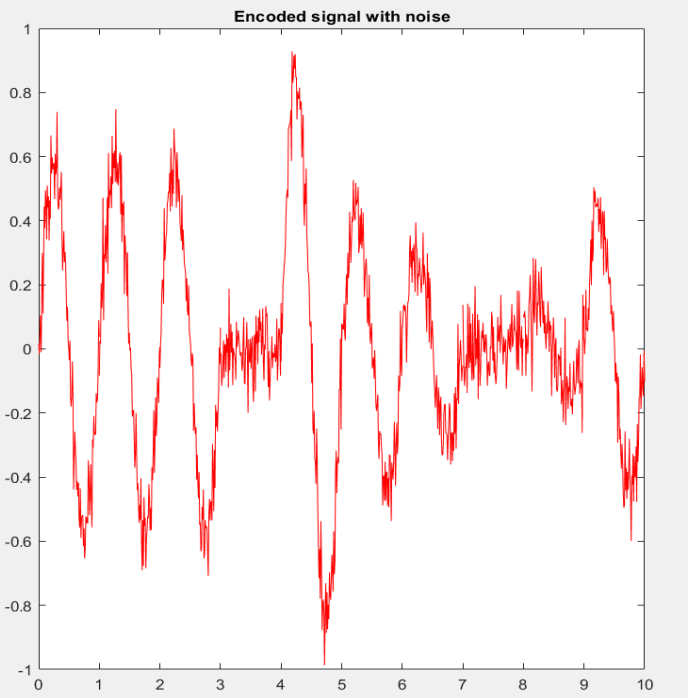
Bit rate=3



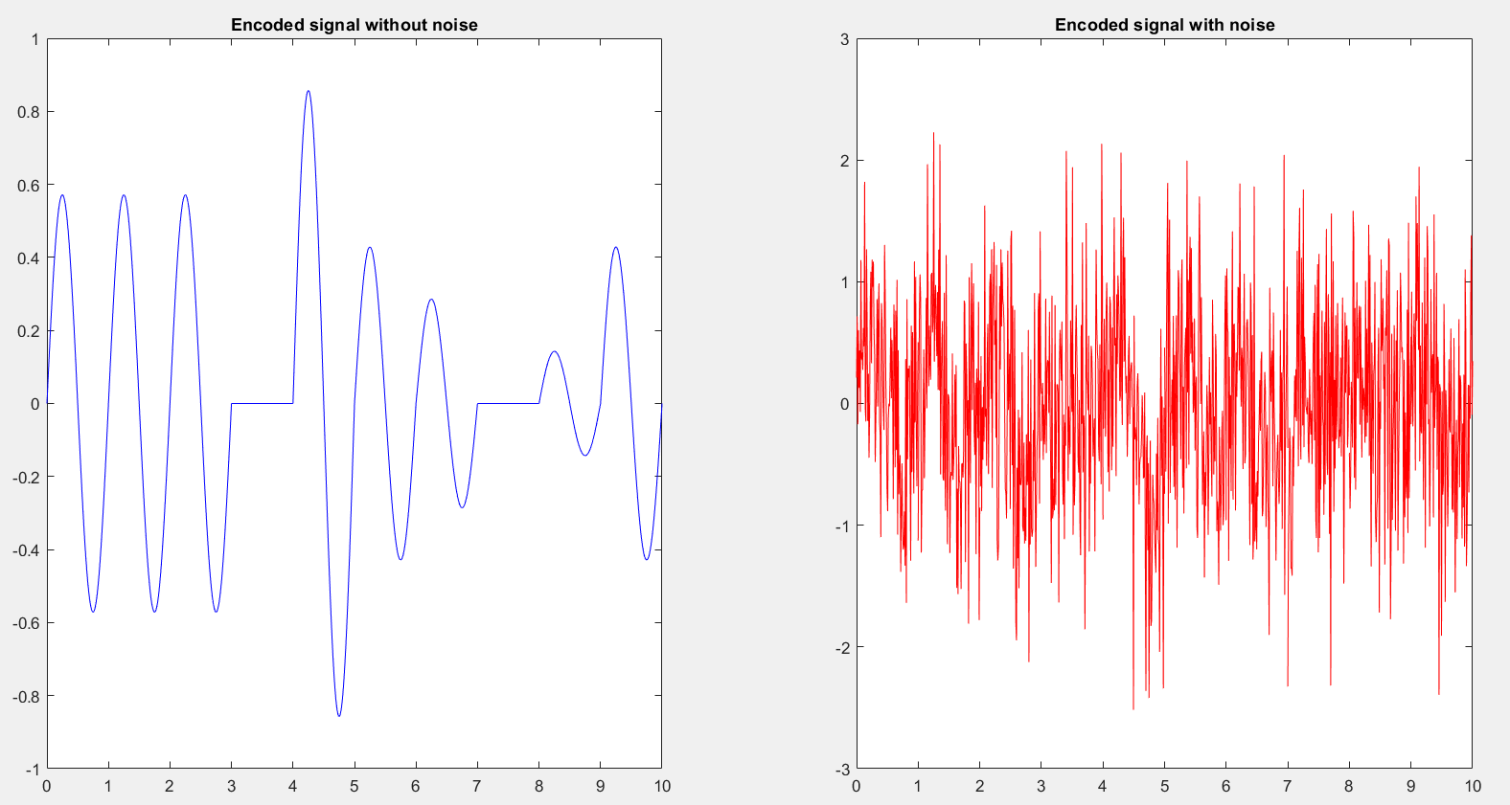


std=0.0075





std=0.075



std=0.75

خروجی به ازای std های مختلف:

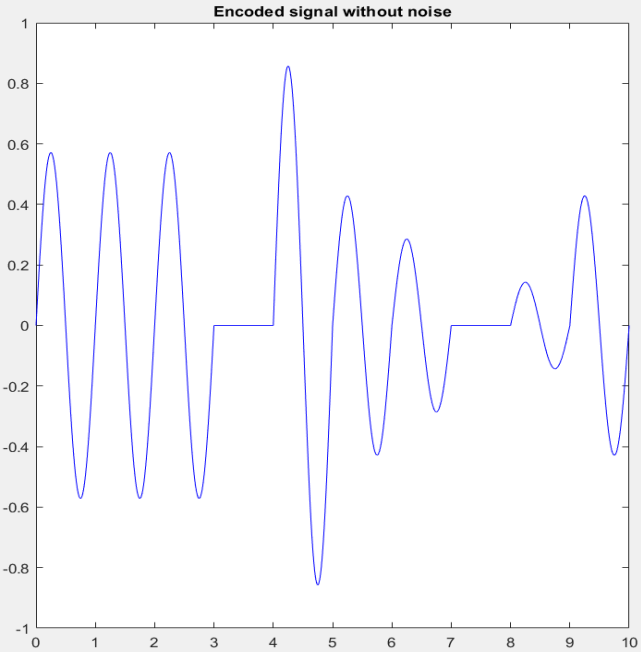
```
added normal noise (rate=3) (std=0.0075):signal
added normal noise (rate=3) (std=0.075):signal
added normal noise (rate=3) (std=0.75):nygjak
```

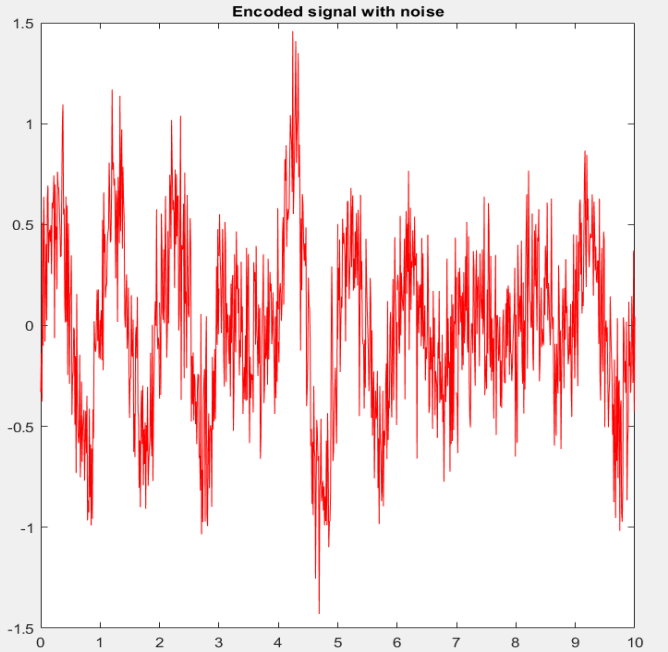
خروجی برای std=0.75 نویز زیادی دارد و اکثر حروف اشتباه decode شده‌اند.

مطابق آنچه در مقدمه بیان شد با افزایش bit rate احتمال افزایش خطا در رمزگشایی بیشتر می‌شود. به دلیل اینکه با افزایش bit rate مرز حالات مختلف به هم نزدیک‌تر شده و تشخیص آنها دشوارتر می‌باشد.

(8_1

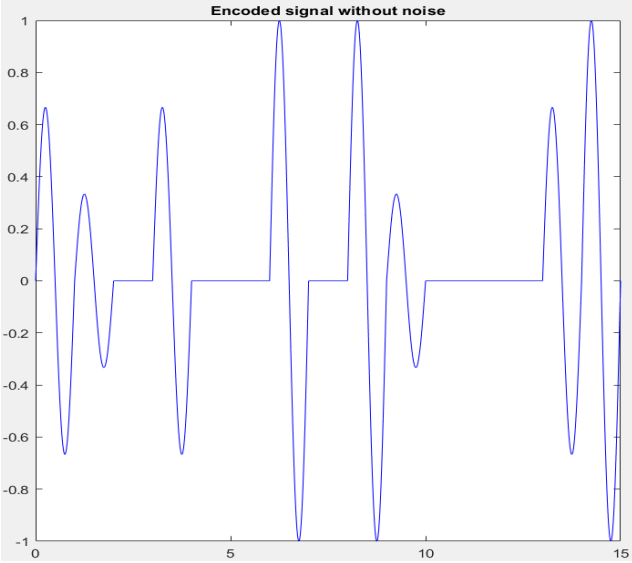
Bit rate=3

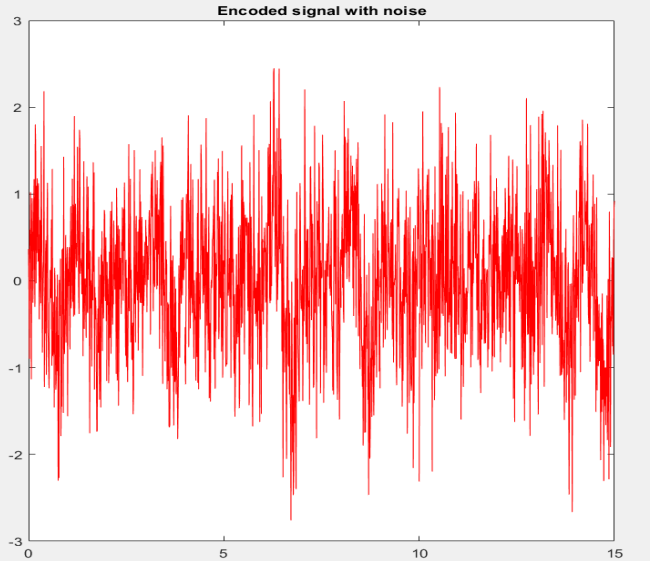




std=0.28

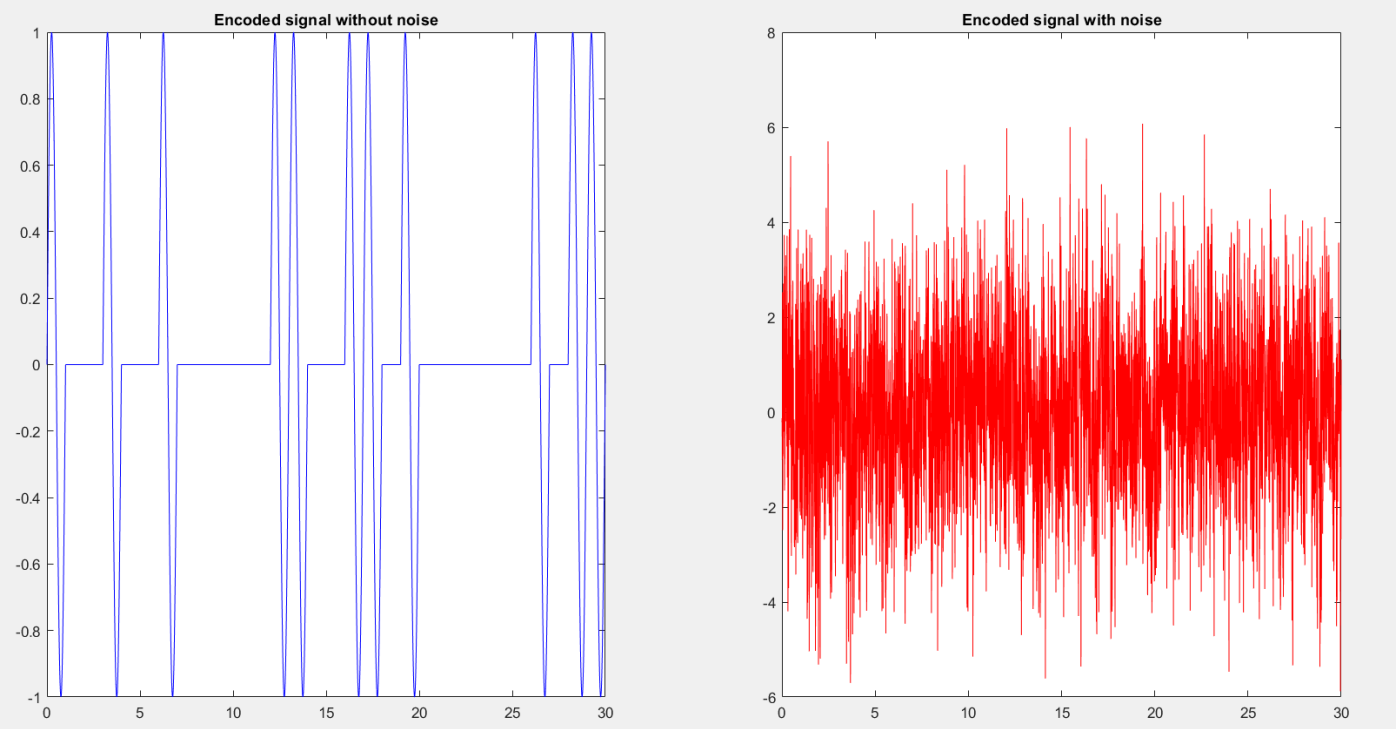
Bit rate=2





std=0.75

Bit rate=1



std=1.2

(9_1

با افزایش دامنه ارسالی threshold ها نیز افزایش پیدا می‌کنند که باعث می‌شود تصمیم‌گیری نسبت به نویز مقاوم‌تر باشد. در نتیجه با power بیشتر به نویز مقاوم‌تر خواهد بود.

(10_1

اگر نویز نداشته باشیم پیام در یک ثانیه دریافت می‌شود و از این نظر محدودیتی برای bit rate نداریم اما ممکن است از لحاظ حافظه برای bit rate دچار محدودیت شویم.

(11_1

به دلیل اینکه دامنه سیگنال ارسالی برای ما مهم است و همواره بین صفر و یک است قرار دادن ضریب پشت سیگنال دریافتی باعث ایجاد تغییر در آن نمی‌شود پس عملکرد نهایی نسبت به نویز تفاوتی ندارد.

(12_1

سرعت adsl حدودا بین 8 تا 16 مگابیت بر ثانیه است که ما در این تمرین با 1 و 2 و 3 بیت بر ثانیه کار کردیم.