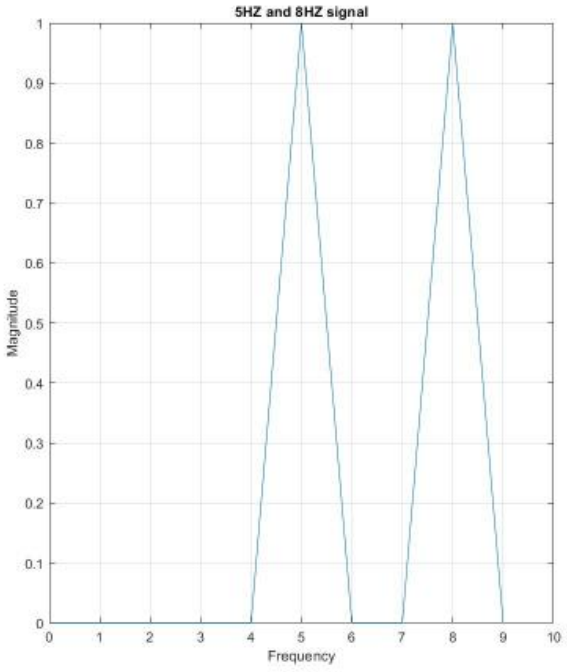
امیرعلی رحیمی 8101100146

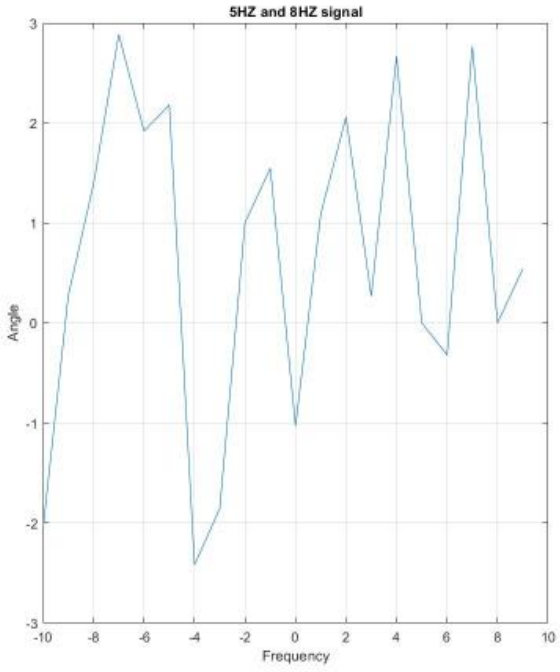
پروژه ی پنجم سیگنال و سیستم ها

بخش اول)

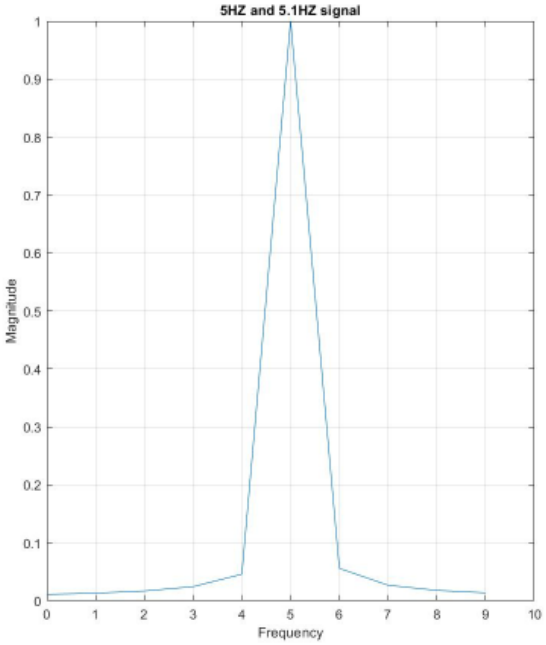
0-1)

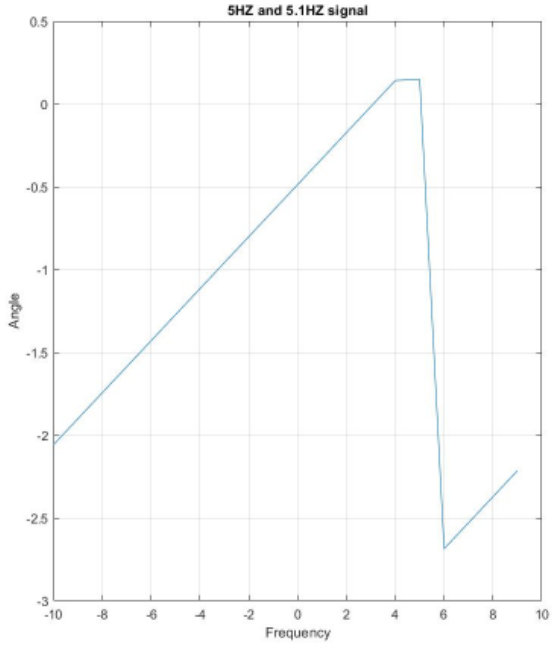
تبدیل فوریه سیگنالی که از دو سیگنال تک تن به فرکانس های 5 و 8 تشکیل شده است:





تبدیل فوریه سیگنالی که از دو سیگنال تک تن به فرکانس های 5.1 و 5 تشکیل شده است:



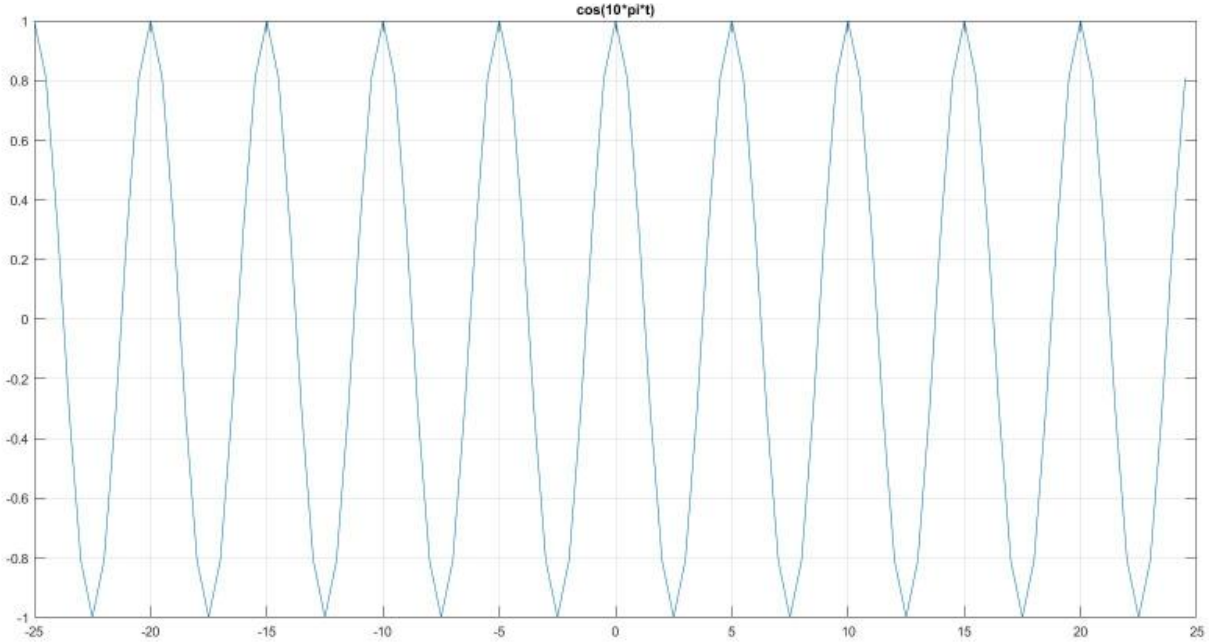


همانطور که مشاهده میشود، از آنجایی که رزولوشن فرکانسی ما، برابر 1 می باشد، توانایی تشخیص سیگنال با فرکانس 5.1 را نداریم.

1-1)

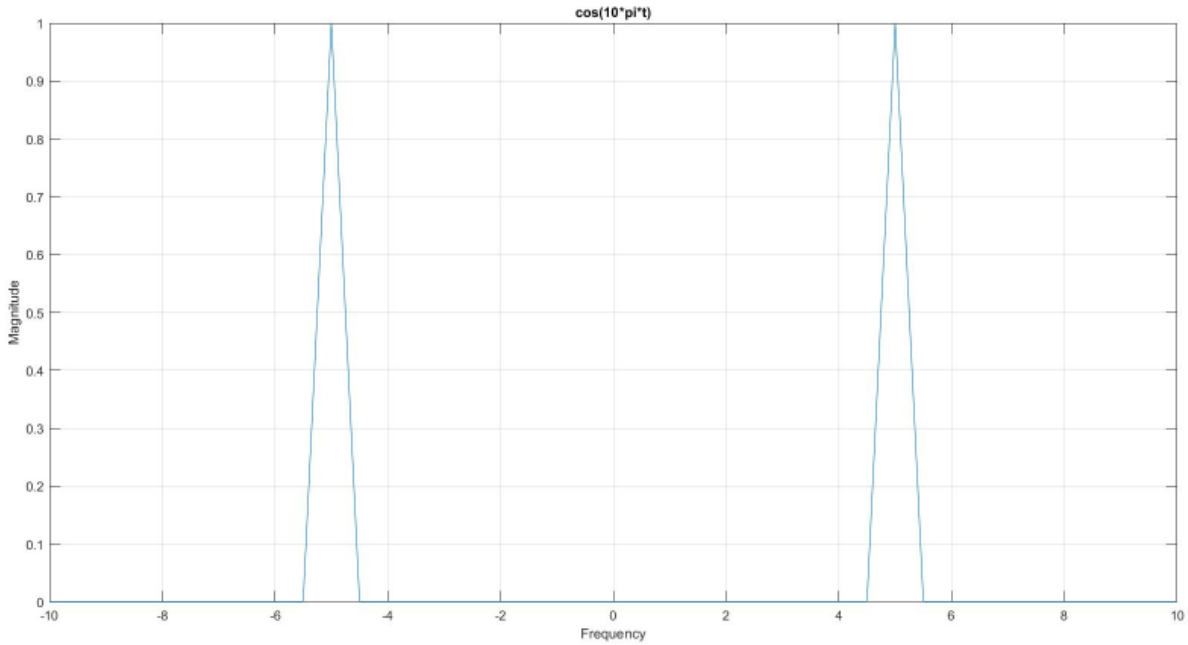
الف)

رسم سیگنال گفته شده:



ب)

رسم تبدیل فوریه سیگنال گفته شده:



می‌دانیم تبدیل فوریه تابع $cos(\omega_0 t)$ به صورت زیر محاسبه می‌شود.

$$\mathcal{F}\{\cos(\omega_0 t)\} = \pi\delta(\omega - \omega_0) + \pi\delta(\omega + \omega_0)$$

از طرفی با توجه به اینکه در متلب تبدیل فوریه را normalize می‌کنیم، ضرایب π را از پاسخ حذف می‌کنیم.

با جایگذاری مقادیر، نتیجه به صورت زیر خواهد بود:

$$\omega_0 = 10\pi \longrightarrow \mathcal{F}_{\mathcal{N}}\{cos(10\pi t)\} = \delta(\omega - 10\pi) + \delta(\omega + 10\pi)$$

از طرفی نمودارها به جای اینکه بر اساس ω رسم شده باشند، بر حسب f رسم شده‌اند. در نتیجه باید این تغییر متغیر را نیز لحاظ کنیم:

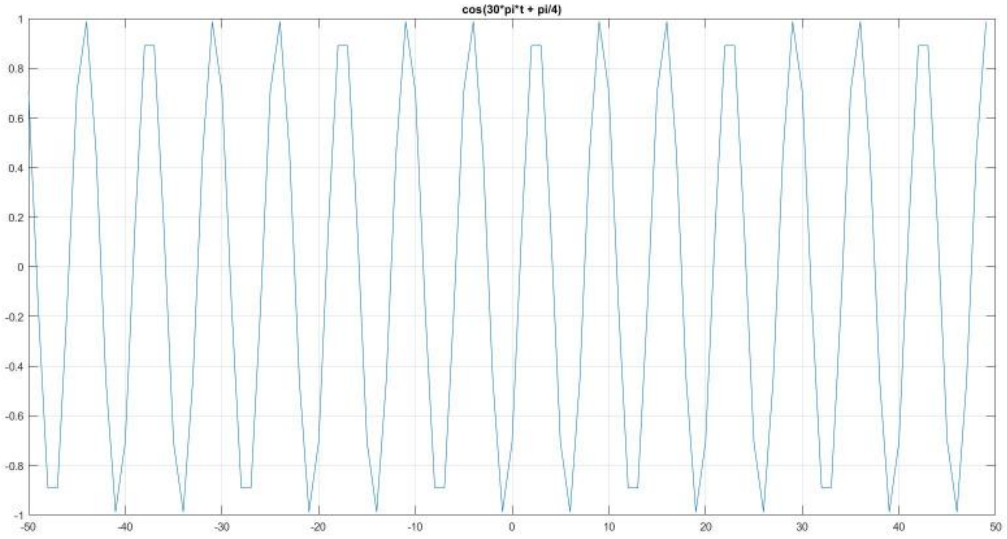
$$\omega = 2\pi f \longrightarrow \mathcal{F}_{\mathcal{N}}\{cos(10\pi t)\} = \delta(f - 5) + \delta(f + 5)$$

بنابر روابط بدست آمده، انتظار داریم که نمودار تبدیل فوریه بدست آمده، در فرکانس های 5- و 5، پیک بزند، که این اتفاق رخ میدهد.

2-1)

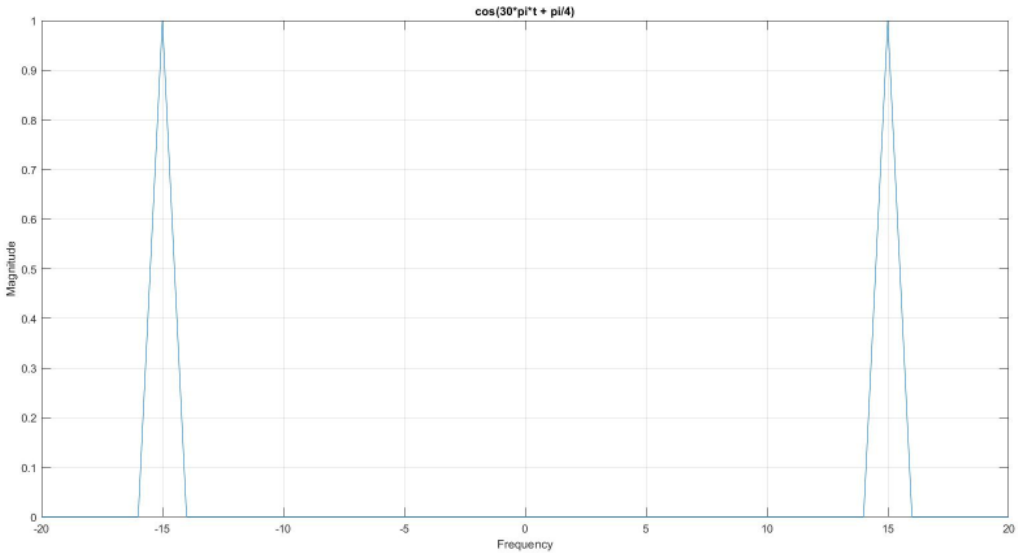
الف)

رسم سیگنال گفته شده:



ب)

رسم تبدیل فوریه سیگنال گفته شده:



ابتدا تبدیل فوریه تابع $cos(\omega_0 t + t_0)$ را محاسبه می‌کنیم:

$$\mathcal{F}\{cos(\omega_0 t + t_0)\} = \int_{-\infty}^{+\infty} \frac{e^{j(\omega_0 t + t_0)} + e^{-j(\omega_0 t + t_0)}}{2} e^{-j\omega t}\, dt$$
$$= \pi e^{-jt_0}\delta(\omega + \omega_0) + \pi e^{jt_0}\delta(\omega - \omega_0)$$

حال تبدیل فوریه تابع $cos\left(30\pi t + \frac{\pi}{4}\right)$ را به صورت زیر بدست می‌آوریم:

$$\mathcal{F}\{cos\left(30\pi t + \frac{\pi}{4}\right)\} = \pi e^{-j\frac{\pi}{4}}\delta(\omega + 30\pi) + \pi e^{j\frac{\pi}{4}}\delta(\omega - 30\pi)$$

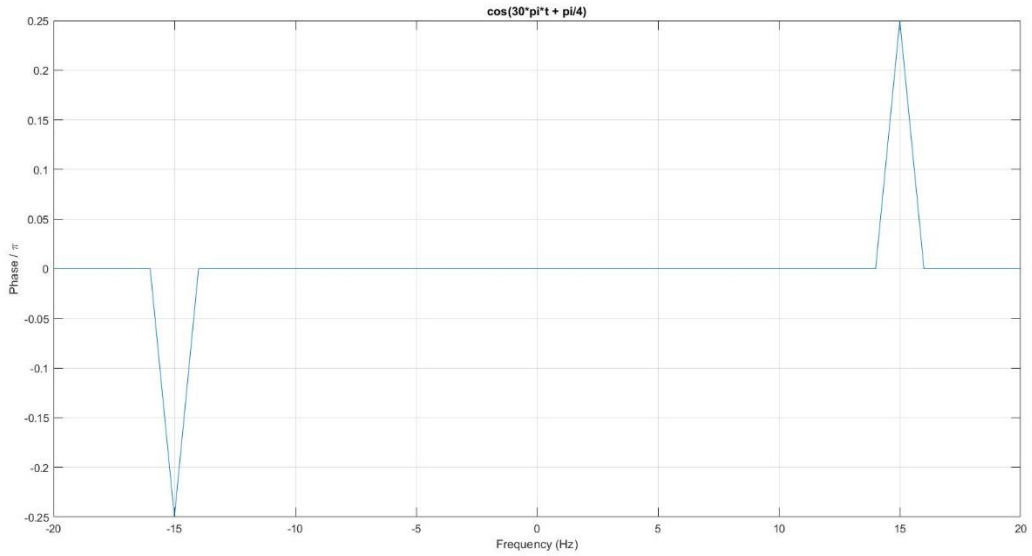
حال تغییر متغیر $\omega = 2\pi f$ را انجام می‌دهیم و با توجه به اینکه باید اندازه تابع را نرمالایز کنیم، ضریب π را در نظر نمی‌گیریم:

$$\mathcal{F}_{\mathcal{N}}\{cos\left(30\pi t + \frac{\pi}{4}\right)\} = e^{-j\frac{\pi}{4}}\delta(f + 15) + e^{j\frac{\pi}{4}}\delta(f - 15)$$

بنابر روابط بدست آمده، انتظار داریم که نمودار تبدیل فوریه بدست آمده، در فرکانس های 15- و 15، پیک بزند، که این اتفاق رخ میدهد.

ج)

نمایش فاز این سیگنال در حوزه ی فوریه:



بنابر رابطه ی بدست آمده در قسمت ب، انتظار میرود در فرکانس 15- فاز این سیگنال در حوزه ی فوریه برابر $-\frac{\pi}{4}$ و در فرکانس 15، برابر $\frac{\pi}{4}$ باشد، که نمودار رسم شده دقیقا نشان دهنده همین مورد است.

بخش دوم)

1-2)

ایجاد map set گفته شده، دقیقا مشابه پروژه قبل میباشد:

```
codedLetters = cell(2,32);
letters = ['a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n' 'o' 'p' ...
           'q' 'r' 's' 't' 'u' 'v' 'w' 'x' 'y' 'z' ' ' '.' ',' '!' '"' ';'];
numbers = 0:1:31;
binaryNums = dec2bin(numbers, 5);
for i=1:32
    codedLetters{1,i} = letters(i);
    codedLetters{2,i} = binaryNums(i, :);
end
```

2-2)

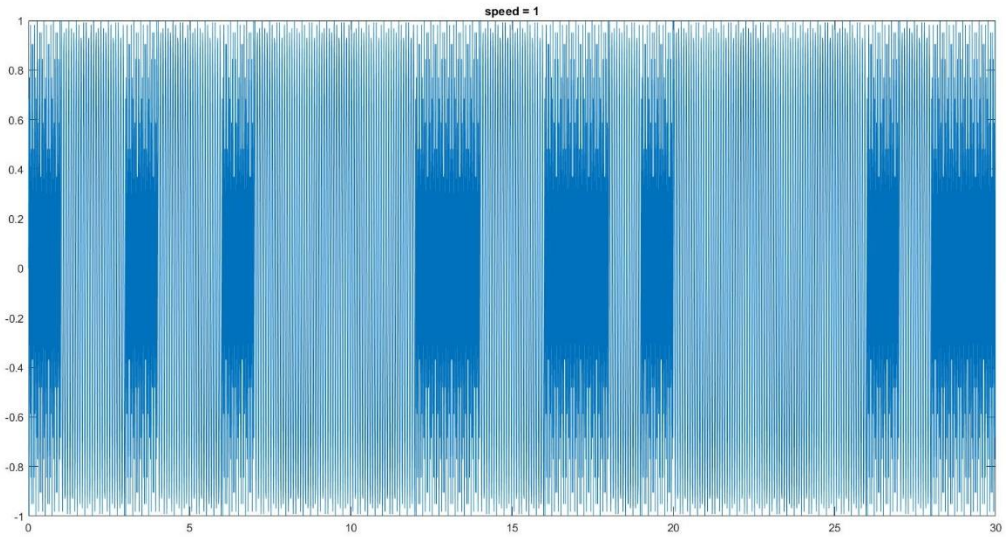
ایده ی پیاده سازی، تقریبا مشابه تمرین قبلی میباشد.

ابتدا پیام داده شده را تبدیل به یک رشته باینری میکنیم. سپس هر speed رشته باینری را به یک فرکانس نگاشت میکنیم و سیگنال 1 ثانیه ای مربوط به آن را تولید کرده و به سیگنال تولید شده ی قبلی میچسبانیم. تصویر نحوه ی پیاده سازی:

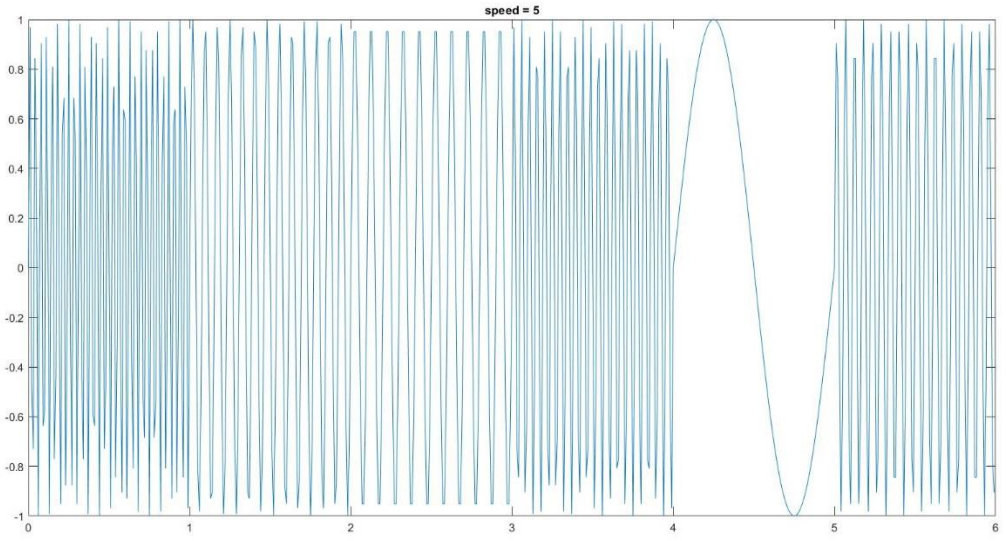
```
function coded = coding_freq(mapSet, message, speed)
    fs = 100;

    binarizedMessage = [];
    for i=1:size(message,2)
        for j=1:size(mapSet,2)
            if(message(1,i)==mapSet{1,j})
                binarizedMessage = [binarizedMessage mapSet{2,j}];
            end
        end
    end
    signal = [];
    t = 0:1/fs:0.99;
    for i=1:speed:size(binarizedMessage, 2)
        this = binarizedMessage(i:i+speed-1);
        freq = floor(fs/(2^(speed+2))) + floor(bin2dec(this)*(fs/(2^(speed+1)))) + 1;
        appendSignal = sin(2*pi*freq*t);
        signal = [signal appendSignal];
    end
    coded = signal;
end
```

3-2) سیگنال تولید شده برای speed=1:

(در p2.m این دو سیگنال تولید شده اند.)



سیگنال تولید شده برای speed=5:



(4-2

ایده ی پیاده سازی این بخش هم تقریبا مشابه پروژه قبلی می باشد.

در هر حلقه، ابتدا یک ثانیه از سیگنال را جدا میکنیم، سپس از آن تبدیل فوریه گرفته و اندیس جایی که مقدار تبدیل قوریه در آن ماکسیمم میشود را پیدا میکنیم. این اندیس، به ما نشان میدهد که این یک ثانیه از سیگنال، با چه فرکانسی فرستاده شده است. که آن را با چند آستانه مقایسه میکنیم(مشابه پروژه قبل) و یک فرکانس به آن نسبت میدهیم. حال از روی فرکانس متوجه میشویم که این سیگنال، مربوط به کدام رشته ی باینری است. آن را به رشته ی باینری ای که تا اینجا بدست آوردیم میچسبانیم و اینکار را تمام شدن سیگنال ورودی ادامه میدهیم. حال از روی mapSet، متوجه میشویم که پیام کد شده در این سیگنال چیست. پیاده سازی به شکل زیر است:

```
function message = decoding_freq(mapSet, signal, speed)
    fs = 100;
    t = 0:1/fs:0.99;
    binarizedMessage = [];
    for i=1:fs:size(signal, 2)
        partSignal = signal(i:i+fs-1);
        [maxValue, maxIdx] = max(abs(fftshift(fft(partSignal))));
        maxIdx = fs/2 - maxIdx + 1;
        for j=1:(2^speed)
            if(maxIdx < floor(j*fs/(2^(speed+1)))+1)
                binarizedMessage = [binarizedMessage dec2bin(j-1, speed)];
                break;
            end
        end
    end
    message = [];
    for i=1:5:size(binarizedMessage,2)
        for j=1:size(mapSet,2)
            if(binarizedMessage(i:i+4)==mapSet{2,j})
                message = [message mapSet{1,j}];
            end
        end
    end
end
```

خروجی کد برای سیگنال هایی که با سرعت 1 و 5 بیت بر ثانیه در قسمت coding_freq ساخته بودیم:

(در p2.m این تابع تست شده است.)

```
>> p2
decoded message for speed=1 signal is: signal
decoded message for speed=5 signal is: signal
```

(5-2

بعد از ساختن سیگنال توسط تابع coding_freq، به آن نویزی به واریانس 0.0001 اضافه کرده و آن را توسط تابع decoding_freq، دیکود کرده و پیام را استخراج می کنیم. نحوه ی پیاده سازی:

(این قسمت در p2.m نوشته شده است.)

```
28      %adding noise with variance 0.0001
29 -    noiseSD= 0.01;
30 -    signal5x = coding_freq(codedLetters, 'signal', 5);
31 -    noise5 = noiseSD*randn(size(signal5x));
32 -    signal5x = signal5x + noise5;
33 -    signal1x = coding_freq(codedLetters, 'signal', 1);
34 -    noise1 = noiseSD*randn(size(signal1x));
35 -    signal1x = signal1x + noise1;
36 -    message1 = decoding_freq(codedLetters, signal1x, 1);
37 -    message5 = decoding_freq(codedLetters, signal5x, 5);
38 -    fprintf('noise standard derivation is: %.2f \n', noiseSD);
39 -    fprintf('decoded message for speed 1: %s \n', message1);
40 -    fprintf('decoded message for speed 5: %s \n', message5);
```

خروجی داده شده:

```
noise standard derivation is: 0.01
decoded message for speed 1: signal
decoded message for speed 5: signal
```

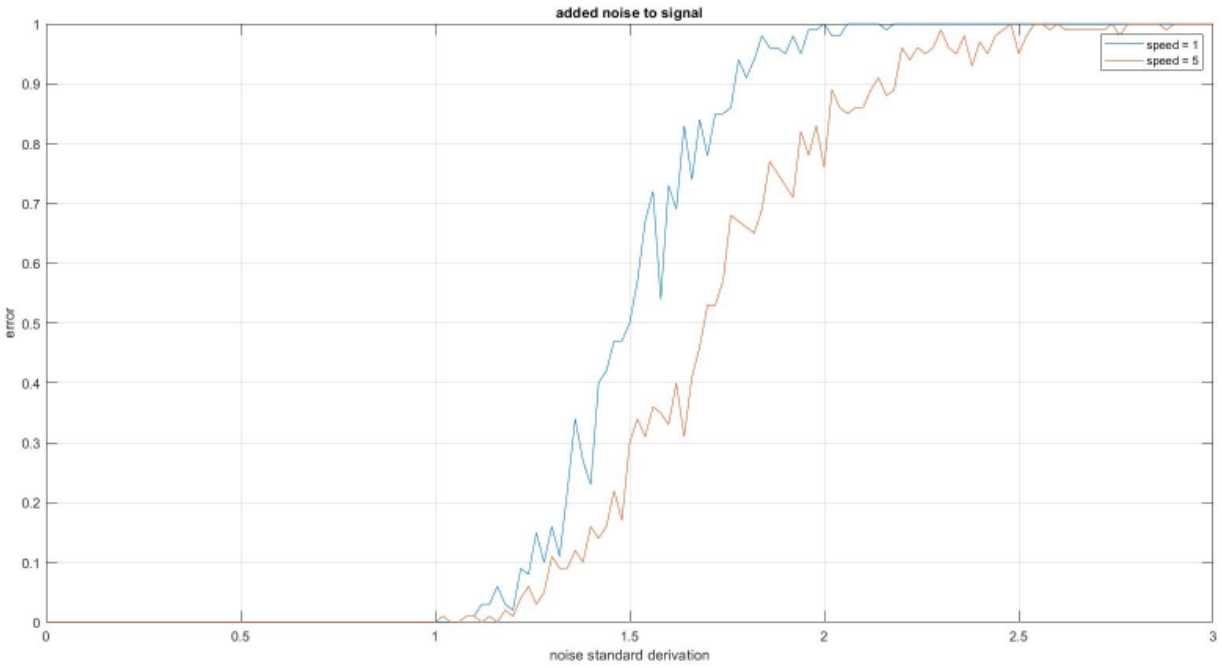
مشاهده می شود برای هر دو نرخ ارسال پیام، پیام مورد نظر به درستی استخراج شده است.

(6-2

ابتدا یک آرایه ای از نویز ها در نظر میگیریم که از 0 تا 3 به گام های 0.02 می باشد. حال به ازای هر نویز، 100 نویز متفاوت ایجاد کرده و با هر دو سیگنال جمع میکنیم و آن سیگنال ها را به تابع decoding می دهیم. میانگین خطا را برابر جمع خروجی های غیر از signal تابع decoding در نظر می گیریم. حال به ازای هر نویز، این میانگین خطا را محاسبه میکنیم و در نهایت این میانگین خطا ها را به ازای هر دو سیگنال با سرعت 1 و 5 رسم می کنیم. نحوه ی پیاده سازی:

```
44 -    primaryMessage = 'signal';
45 -    errors = zeros(2, 100);
46 -    noiseSDs = 0:0.02:3;
47 -    for i=1:length(noiseSDs)
48 -        noiseSD= noiseSDs(i);
49 -        wrongFor1x = 0;
50 -        wrongFor5x = 0;
51 -        for j=1:100
52 -            signal5x = coding_freq(codedLetters, primaryMessage, 5);
53 -            noise5 = noiseSD*randn(size(signal5x));
54 -            signal5x = signal5x + noise5;
55 -            signal1x = coding_freq(codedLetters, primaryMessage, 1);
56 -            noise1 = noiseSD*randn(size(signal1x));
57 -            signal1x = signal1x + noise1;
58 -            message1 = decoding_freq(codedLetters, signal1x, 1);
59 -            message5 = decoding_freq(codedLetters, signal5x, 5);
60 -            if(~strcmp(message1, primaryMessage))
61 -                wrongFor1x = wrongFor1x + 1;
62 -            end
63 -            if(~strcmp(message5, primaryMessage))
64 -                wrongFor5x = wrongFor5x + 1;
65 -            end
66 -        end
67 -        errors(1, i) = wrongFor1x / 100;
68 -        errors(2, i) = wrongFor5x / 100;
69 -    end
```

رسم نمودار گفته شده:



همانطور که مشاهده میشود، در مقادیر بعد از انحراف معیار 1، هر دو سرعت خطاهای تقریبا نزدیک به همی دارند و همچنین خطای سرعت 5، مقداری کمتر از سرعت 1 می باشد. در صورتی که انتظار می رفت سیگنالی که با سرعت 1 فرستاده شده است، خطای بسیار کمتری داشته باشد.

7-2)

برای این بخش، یک قطعه به کد قبل اضافه میکند و در حلقه چک میکند که اگر مقدار error کمتر از 10 درصد بود، آن را به عنوان آخرین نویزی که این سیگنال ها به آن مقاوم است در نظر میگیرد. حال این دو مقدار را برای هر یک از سیگنال های با سرعت 1 و 5 چاپ میکنیم:

```
last noise that we had less than 10% error for speed = 1 is: 1.24
last noise that we had less than 10% error for speed = 5 is: 1.36
>>
```

8-2)

در صورتی که اختلاف بین فرکانس ها بیشتر باشد، باعث میشود که سیگنال ها به نویز مقاوم تر شوند. برای اینکه اختلاف فرکانس ها را بیشتر کنیم، میتوانیم پهنای باند بیشتری مصرف کنیم. بنابراین با افزایش پهنای باند، هم میتوان سیگنال ها را به نویز مقاوم تر کرد و هم میتوان سرعت انتقال اطلاعات را افزایش داد.

9-2)

خیر، زیرا با تغییر فرکانس نمونه برداری، پهنای باند تغییری نمیکند و فواصل بین فرکانس های انتخاب شده، ثابت میمانند که این باعث میشود مقاومتی در برابر نویز به ارمغان نیاید. حتی ممکن است نویز موجود در سیستم افزایش پیدا کند زیرا بیشترین قدرت سیگنال ها در پایین ترین بخش پهنای باند متمرکز هستند. برای افزایش مقاومت در برابر نویز، میتوانیم فرکانس نمونه برداری را همراه با پهنای باند افزایش دهیم.