Signal&System CA#5

Coding Message Using Fourier series

SeyedMahdi HajiSeyedHossein

810100118

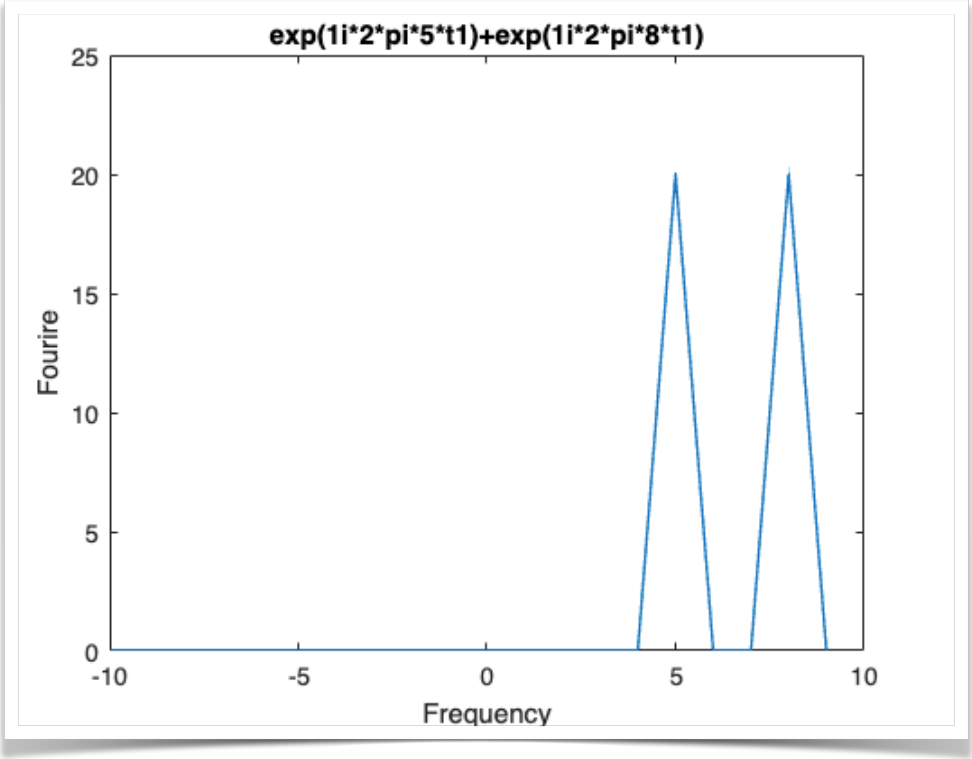
PART 1

QUESTION 0 :

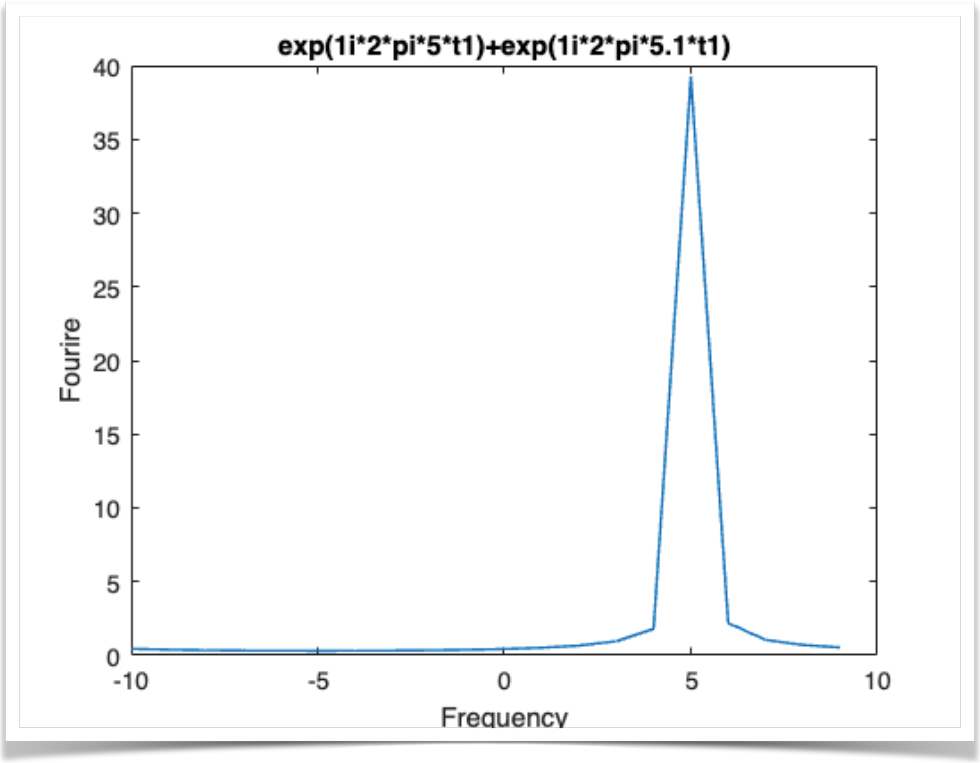
Frequency Resolution

میدانیم رزولوشن فرکانسی قدرت تفکیک فرکانسی در حوزه فوریه را نشان میدهد. در این سوال رزولوشن فرکانسی برابر 1 Hz در نظر گرفته شده .

برای همین سیگنال (x0= e^(j2pi*8t)+e^(j2pi*5t (که اختلاف فرکانس های دو سینگنال تک تن آن بیشتر از ۱ است) در حوزه فوریه دو قله ی ۸ و ۵ را داراست .



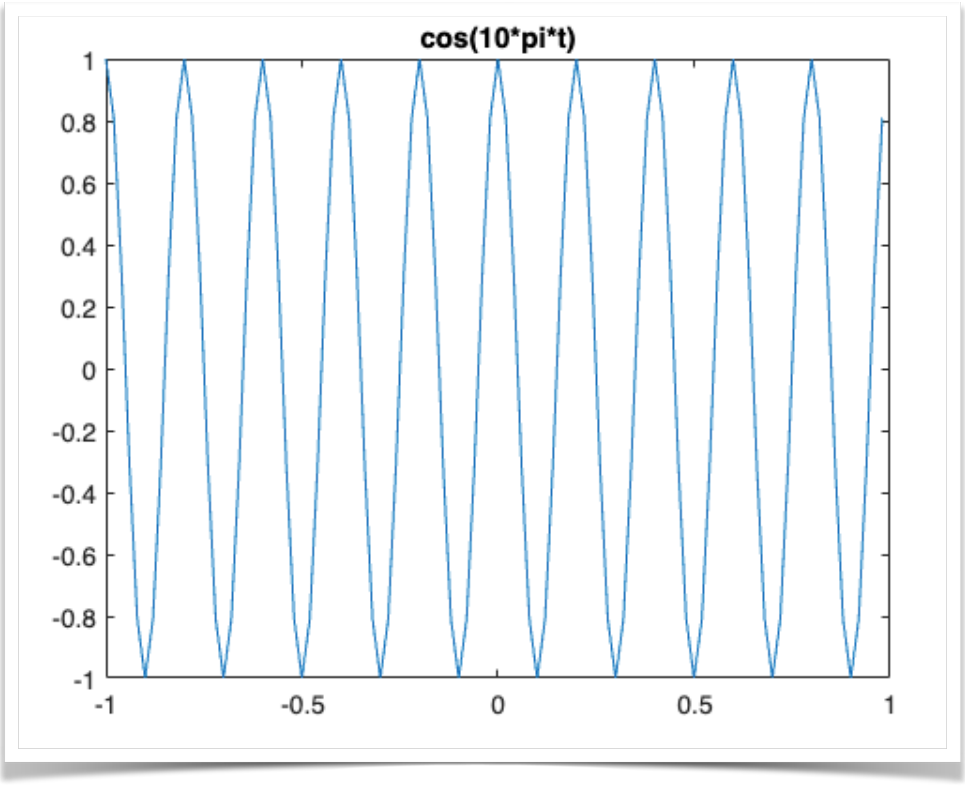
اما برای سیگنال x0.1 = e^(j2pi*5t)+e^(2pi*5.1t) که اختلاف بین دو فرکانس های آن کمتر از یک است ، در حوزه فوریه تفکیک پذیر نیست ، پس فقط یک قله در فرکانس 5Hz خواهد داشت .



QUESTION 1 :

الف)

با توجه به فرمول t(sample) = 1/fs ، t را به صورت -1:0.02:0.98 تعریف میکنیم و سیگنال cos(10pi*t) را رسم میکنیم :

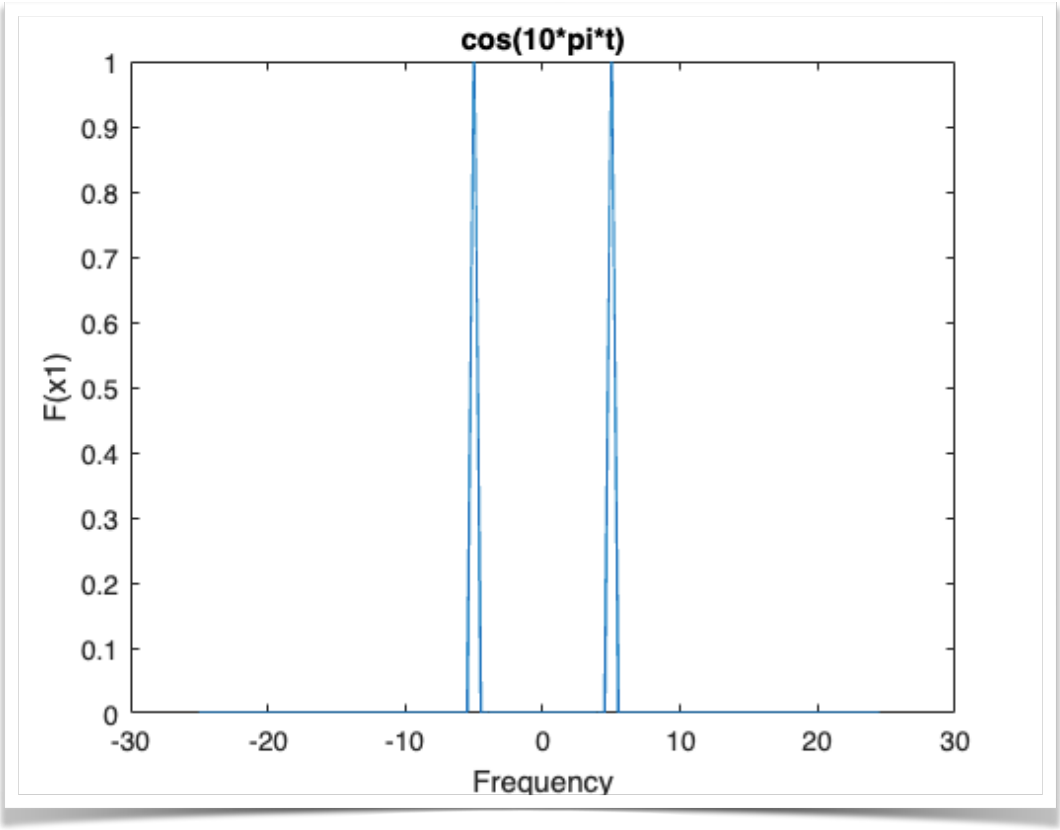


ب)

حالا با دستور fftshift(fft(x1)) سینگال cos(10pi*t) را به حوزه فوریه میبریم .

و با f0 = -25:0.5:24.5 سینگال را در حورزه فوریه به صورت

دوتایی (f0 , |(y0)/max(|y0 |)|) رسم میکنیم :



X = cos(10pi*t) = 1/2*(e^j10pi*t + e^-j10pi*t)

حالا اگر سیگنال را در حوزه فوریه ببریم :

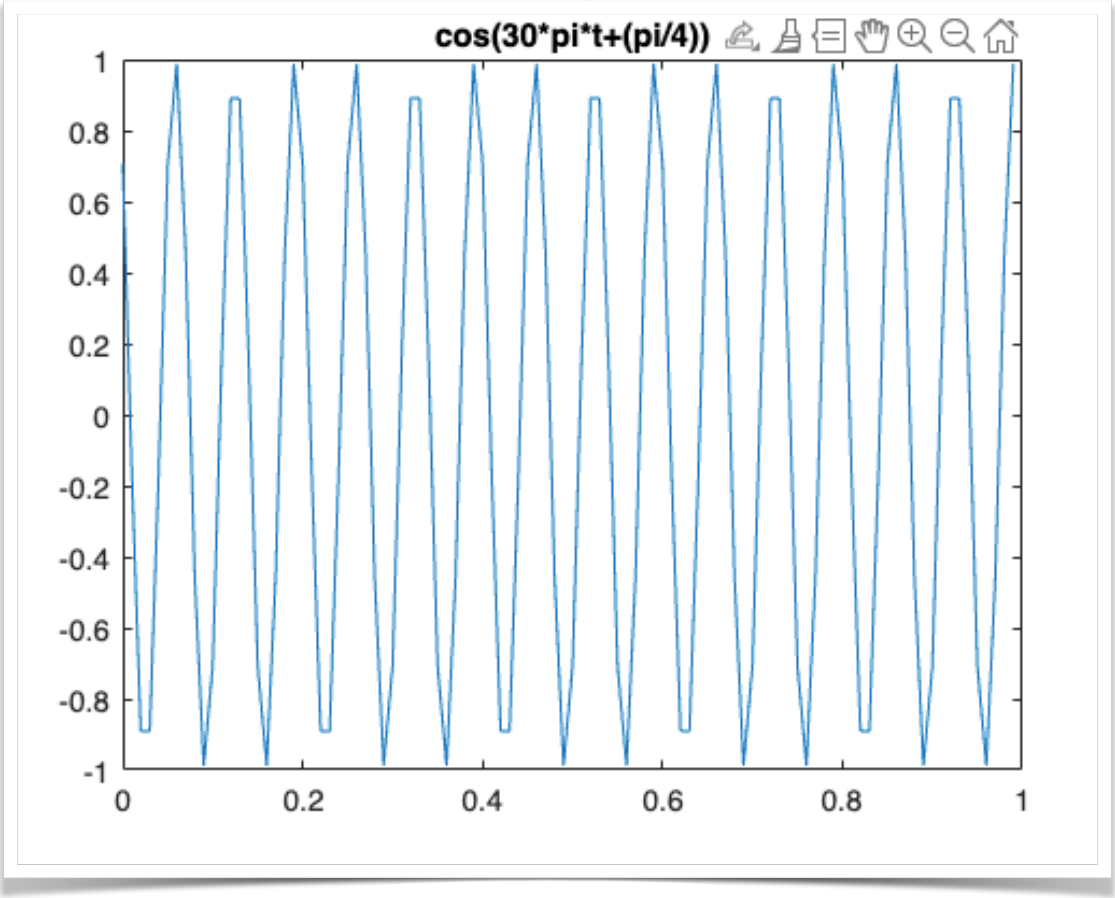
F{x} = pi . delta(w-10) + pi . delta(w+10)

همان طور که در شکل مشخص است دو قله نزدیک به دلتای دیراک در ۵ ، -۵ مشاهده میکنیم .
چون در اینجا فرکانس زاویه ای را به صورت omega = f/2*pi تعریف کرده ایم.

QUESTION 2:

الف)

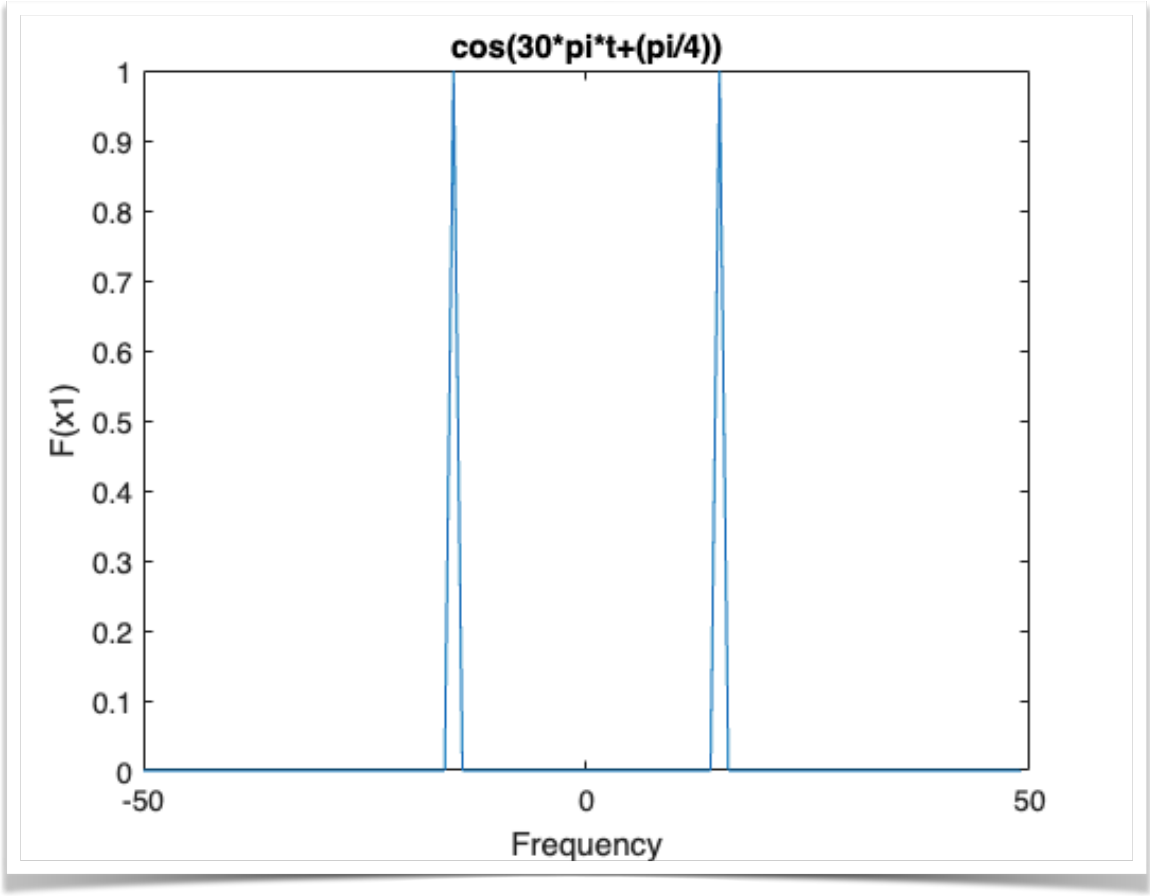
نمودار خواسته شده را به دوتایی t = 0: 0.01 : 0.99 , x = cos(30pi*t + pi/4) رسم میکنیم :



ب)

سپس با همان دستور fftshift(fft(x)) ، x را به حوزه فوریه میبریم . و در متغیر y ذخیره میکنیم .

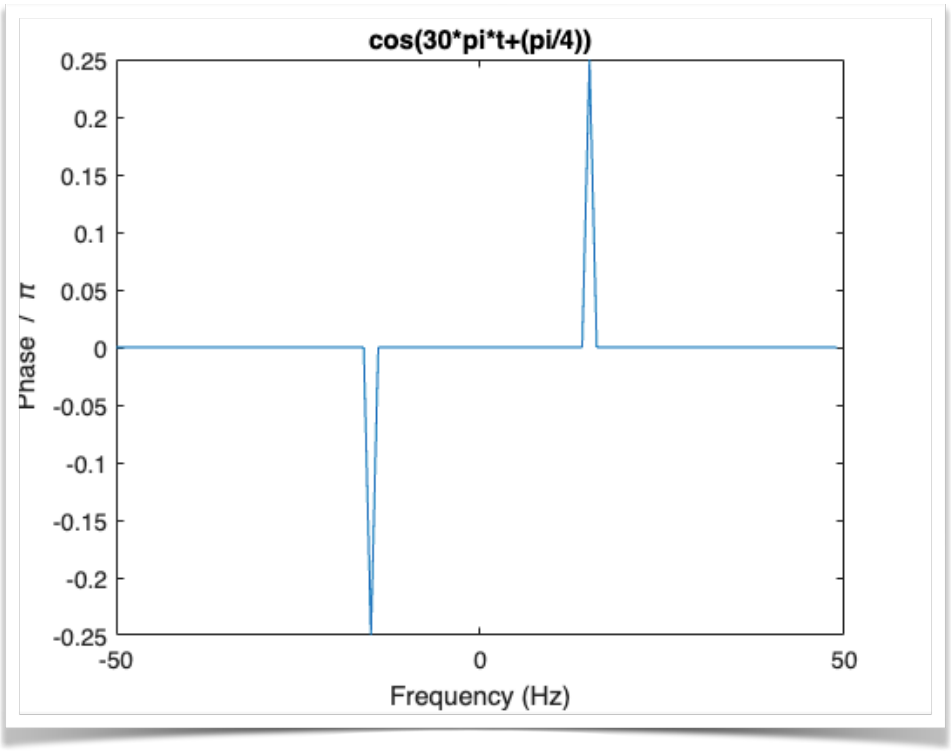
f= -50 :1 :49 را تعریف میکنیم . و با استفاده از دوتایی (|y0)/max(|y0 |) , |f0) در حوزه فوریه اندازه را رسم میکنیم .



X = cos(30pi*t + pi/4) = 1/2(e^j30pi*t+pi/4 + e^-j30pi*t-pi/4)

F{X} = pi . e^jpi/4 * delta(w - 30) + pi . e^-jpi/4 * delta(w + 30)

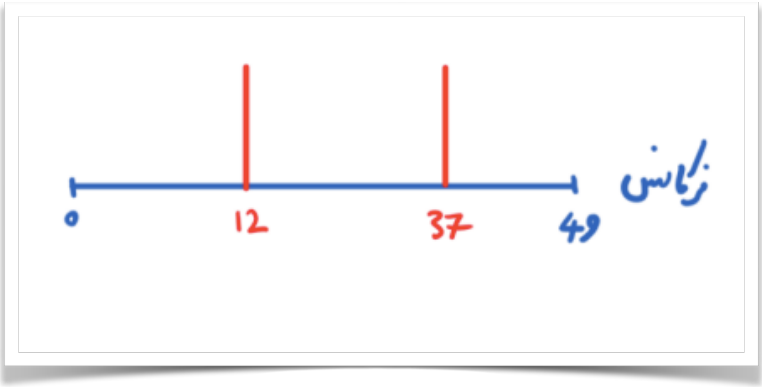
ج)



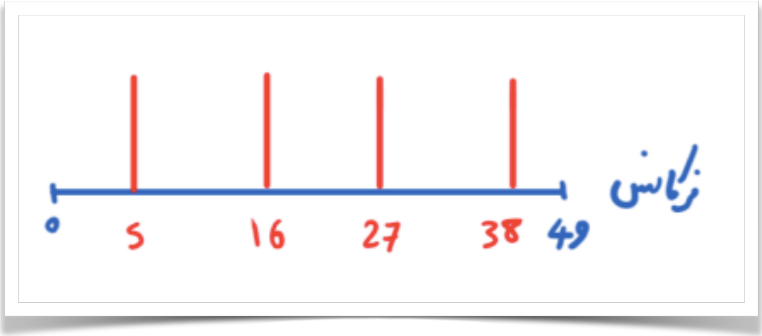
طبق سری فوریه بالا که در قسمت ب یادداشت شد ، فازهای سیگنال -pi/4 , pi/4 است .

PART 1

مشابه آنچه در صورت پروژه هم ذکر شده است . من برای ارسال 1bit/sec از فرکانس هایی که بیشترین فاصله را دارند استفاده میکنم :



هم چنین برای ارسال 2bit/sec هم مشابه صورت پروژه بیشترین فاصله را برای انتخاب فرکانس استفاده میکنم تا خطایمان هنگام اضافه کردن نویز کاهش یابد :



QUESTION 1:

MapSet Preparation:

مشابه تمرین قبلی همان مپست را تکرار میکنیم . تا بتوان ۳۲ کاراکتر را بتوانیم رمز گذاری کنیم . چون ۳۲ کاراکتر داریم پس باید هر کاراکتر را با ۵ بیت صفر و یک مشخص میکنیم .

```
function Mapset=MapSet()
    Mapset=cell(2,32);
    alphabet = 'abcdefghijklmnopqrstuvwxyz .,!";';
    for i = 1:32
        Mapset{1,i} = alphabet(i);
        Mapset{2,i} = dec2bin(i-1, 5);
    end

end
```

QUESTION 2 :

freq_coding:

تابع خواسته شده با نام coding_freq را توضیح میدهیم :

ابتدا در اول با استفاده strcmp کاراکتر های ورودی را با MapSet مقاسه میکنیم عدد باینری مربوط به هرکدام را در باینری مسیج میریزیم .

حالا با استفاده از ، یک لوپ باینری مسیج را جدا میکنیم با توجه به bit_rate .

```
binaryMessage=cell2mat(binaryMessage);

loopCounter = 1;
for k = 1 : bit_rate : length(binaryMessage)
    index1 = k;
    index2 = min(k + bit_rate - 1, length(binaryMessage));
    out{loopCounter} = binaryMessage(index1 : index2);
    loopCounter = loopCounter + 1;
end

binaryMessage = out;
```

حالا در x در تمام حالت های مختلف باینری را نگه میداریم ، و در متغییر y مقدار فرکانس مرتبط با هر x را نگه میداریم .

حالا باینری مسیج هایی را که جدا کرده ایم را با متغییر های x نگه میداریم ، مقایسه میکنیم ، و فرکانس مربوط به ان هارا از y استخراج میکنیم ، و سپس فرکانس های مربوط را به موج sin(2*pi*f*t) تبدین میکنیم .

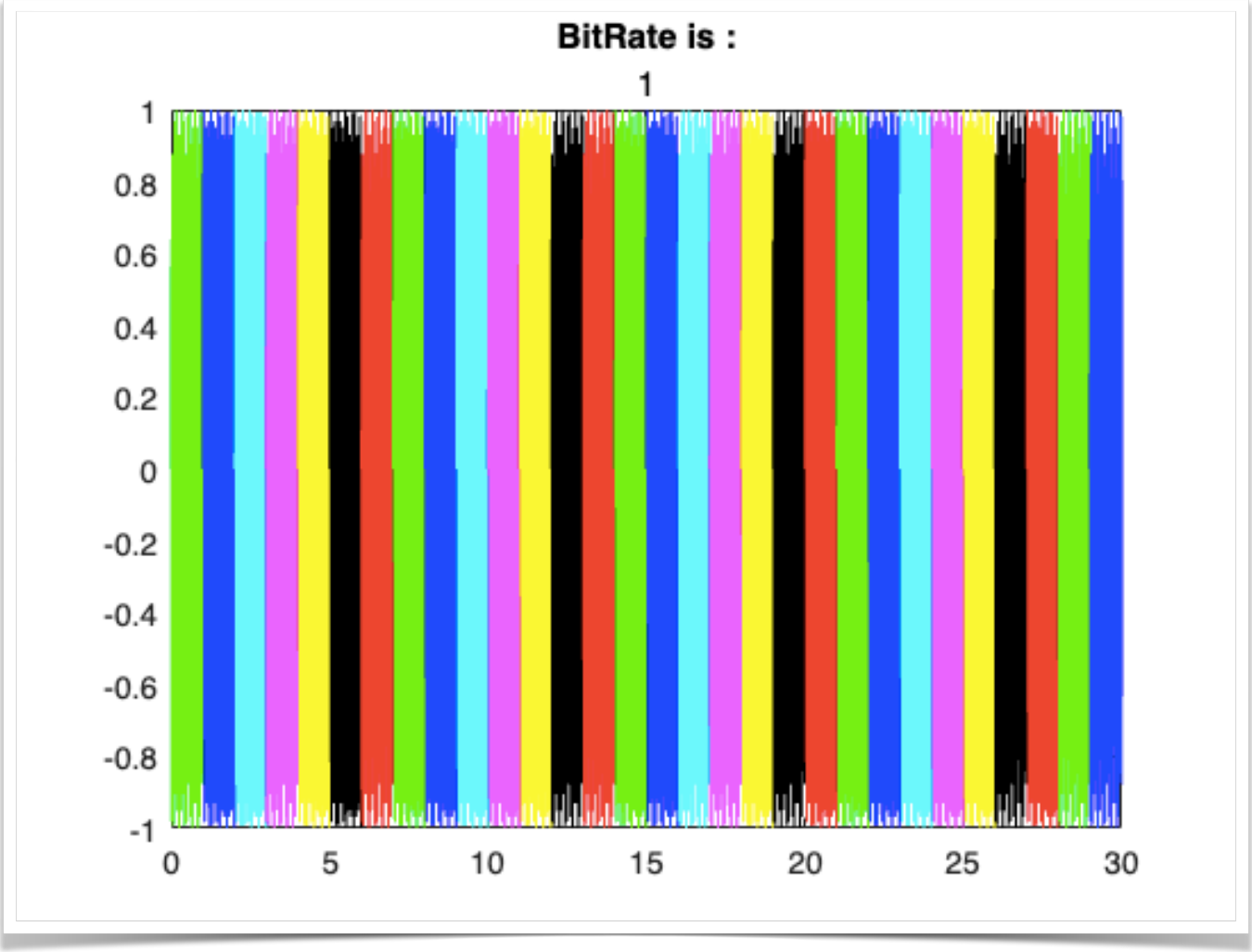
```
44
45          y=zeros(1,2^bit_rate);
46          w=fix(50/(2^bit_rate)+1);
47          z=fix(w/2);
48
49
50          for i=1:2^bit_rate
51              y(1,i)=z;
52              z=z+w;
53
54          end
55
56
57          t1=zeros(n,100);
58          for i=1:n
59              t1(i,:)=linspace(i-1,i,fs);
60          end
61
62
63          ts=1/fs;
64          t=0:ts:1-ts;
65          f=[];
66
67          for i=1:n
68              for j=1:2^bit_rate
69                  if strcmp(binaryMessage(1,i),x(1,j))==1
70                      f=[f y(1,j)];
71                  end
72              end
73          end
74          encoded_message=zeros(n,100);
75          for i=1:n
76              encoded_message(i,:)=sin(2*pi*f(1,i)*t);
77
78          end
```

```
65          end
66
67          figure()
68          colors = {'r', 'g', 'b', 'c', 'm', 'y', 'k'};
69          for k=1:n
70              colorID = mod(k , 7) + 1 ;
71              plot(t1(k,:),encoded_message(k,:) ,'Color', colors{colorID});
72              title('BitRate is :',bit_rate)
73              hold on
74          end
75      end
76
```

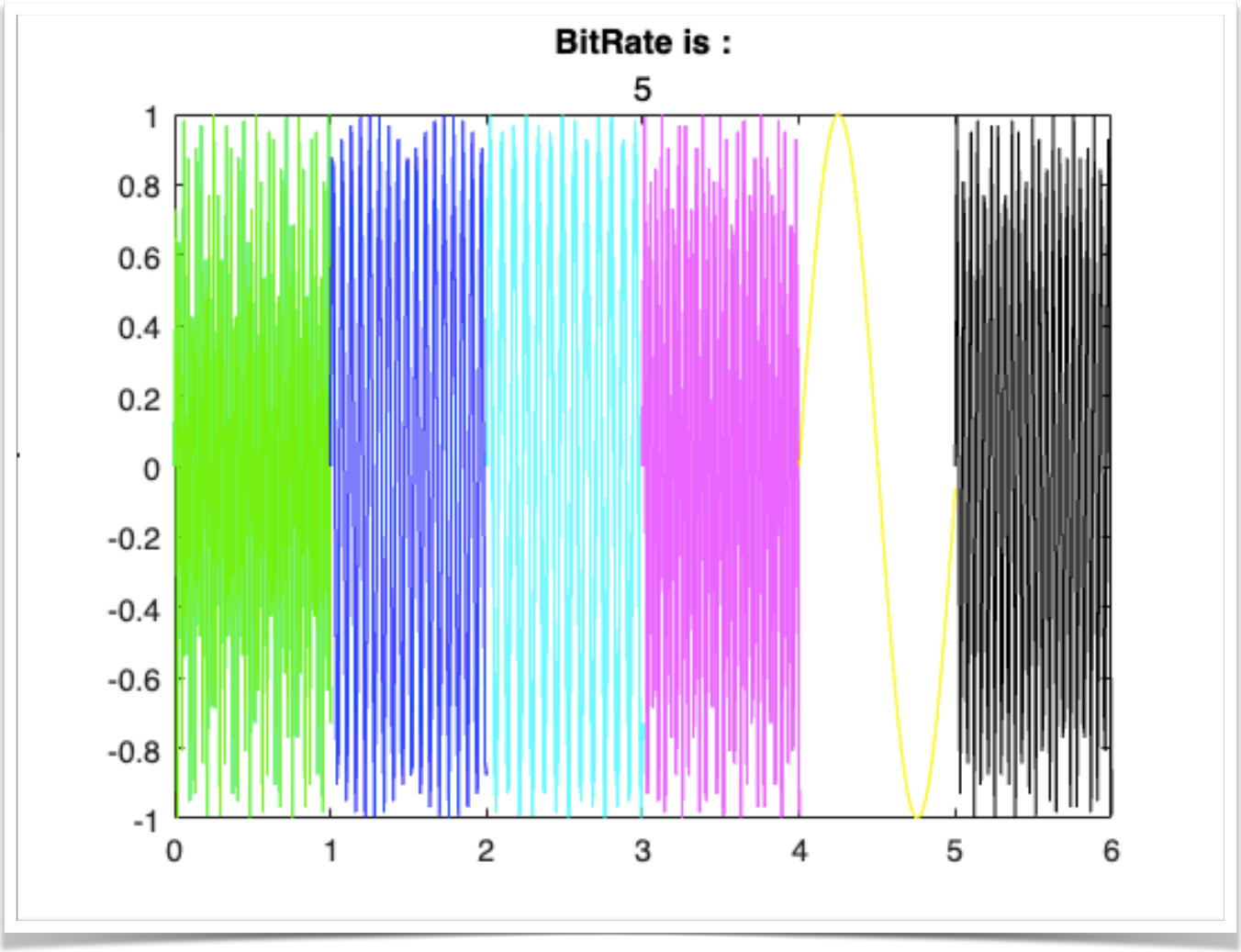
QUESTION 3 :

freq_coding sample :

For Bit Rate = 1 bit/sec



For Bit Rate = 5 bit/sec



QUESTION 4 :

freq_decoding:

```
>> signal_coded_bitRate_1 = coding_freq("signal" , 1);
>> decoding_freq(signal_coded_bitRate_1 , 1)
signal

ans =

    'signal'

>> |
```

```
>> signal_coded_bitRate_5 = coding_freq("signal" , 5);
>> decoding_freq(signal_coded_bitRate_5 , 5)
signal

ans =

    'signal'

>> |
```

روش کار بدین صورت است که مانند بخش کدنیگ یک سلول با نام x تمام جایگشت های مربوط به بیت های مربوط به آن بیت ریت را ذخیره میکند. و در ماتریس y فرکانس های مربوط به x های متناظر را نگه میداریم که همان ضرایب درونی آرگومان سینوس هستند .

حالا متغییر freq تمام فرکانس های از بازه ی ۵۰- تا ۴۹ را با فاصله یک (چون تعداد نمونه ها در هر ثانیه ۱۰۰ تاست و fs هم ۱۰۰ است) نگه میداریم .

حالا با استفاده از دستور fftshitf(fft(codedMessage)) پیام را به حوزه فوریه میبریم ، و قدر مطلق گرفتن از آن اندازه آن را حساب میکنیم . و در متغییر اندازه سیو میکنم .

سپس با با دستورmax قله های این موج را پیدا میکنیم ، و در متغییر loc ذخیره میکنم .

حالا با دیدن اینکه این قله مربوط به کدام فرکانس است را پیدا میکنیم . و بدین تربیت متوجه شده ایم که هر پیام در هر ثانیه در بردارنده ی چه فرکانسی است .

QUESTION 5 :

Gaussian Noise:

دقیقا مشابه پروژه قبل همان استراتژی را برای اضافه کردن نویز گاوسی به پیام کد شده پیش میگیریم :

/MATLAB Drive/P2/AddNoise.m

```
1  function message_with_noise=AddNoise(coded , sigma)
2      num=size(coded);
3      fs=100;
4      encoded_message_size=size(coded);
5
6      noise=sigma*randn(encoded_message_size(1),encoded_message_size(2));
7
8      message_with_noise=coded+noise;
9      t=zeros(num(1),fs);
10     figure()
11
12     for i=1:num(1)
13         t(i,:)=linspace(i-1,i,fs);
14     end
15
16     for k=1:num(1)
17         plot(t(k,:),message_with_noise(k,:),'b');
18         hold on
19     end
20 end
21
```

/MATLAB Drive/P2/TestAddingNoise.m

```
1
2
3       r1 = coding_freq("signal" , 1);
4       r5 = coding_freq("signal" , 5);
5
6
7       sigma = 0.0001;
8
9
10      r1n = AddNoise(r1 , sigma);
11      r5n = AddNoise(r5 , sigma);
12
13
14      decoding_freq(r1n , 1);
15      decoding_freq(r5n , 5);
16
```

```
>> TestAddingNoise
signal
signal
>>
```

همان طور که مشاهده میشود برای واریانس 0.0001 هم به درستی سیگنال ها دیکود شدند .

QUESTION 6 :

Testing Different Power Of Noise:

به ترتیب پیام های با سرعت 1bit/sec و 5bit/sec :

واریانس 0.25

```
>> TestAddingNoise
signal
signal
>>
```

واریانس 1

```
>> TestAddingNoise
signal
signal
>>
```

واریانس 2.5

```
>> TestAddingNoise
ham"y
signlh
>>
```

واریانس 3

```
>> TestAddingNoise
 yaurl
shqnaw
>>
```

پس از مشاهدات من :

سیگنال با بیت ریت .1bit/sec مقاوم تر بود به این علت که فاصله بین فرکانس های حاوی پیام آن بیشتر است بنابراین با توجه به اینکه بیشتر فاصله را دارند خطا کمتر خواهد بود. (مراجعه به توضیحات اول همین تمرین در گزارش من)

QUESTION 7 :

Accepting Power Of Noise:

بیشترین واریانس نویزی که پیام با سرعت ارسال 5 نسبت به آن مقاوم بود 2.1و برای پیام با سرعت ارسال 1 برابر با 3.1بود.

QUESTION 8 :

Make Stronger Against Noise:

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 اطالعات را ارسال کنیم و در عین حال نسبت به نویز مقاوم باشیم.

QUESTION 9 :

Make Stronger Against Noise:

وقتی پهنای باند را تغییر نمیدهیم افزایش فرکانس نمونهبرداری تغییری در فاصله بین فرکانسها ایجاد نمیشود به همین دلیل تاثیری در مقاوم بودن نسبت به نویز نخواهد داشت.