

Computer Assignment 5

مصطفی کرمانی نیا 810101575

امیر نداف فهمیده 810101540

بخش اول

(تمرين 1)

ابتدا محاسبات دستی را انجام میدهیم تا بدانیم انتظار چه چیزی داریم.

$$u_{(+)}) = e^{j\omega_0 t} + e^{j\omega_1 t} \xrightarrow{\mathcal{F}} 2\pi (\delta_{(\omega - \omega_0)} + \delta_{(\omega - \omega_1)})$$

پس $f_1 = \omega_0$, $f_2 = \omega_1$ و $\omega_0 > \omega_1$

نمودار

$$u_{(+)}) = e^{j\omega_0 t} + e^{j\omega_1 t} \xrightarrow{\mathcal{F}} 2\pi (\delta_{(\omega - \omega_0)} + \delta_{(\omega - \omega_1)})$$

پس $f_1 = \omega_1$, $f_2 = \omega_0$ و $\omega_0 < \omega_1$

نمودار

```

%% Part 0: پاکسازی محیط
close all;
clc;
clear;

%% Part 1
fs = 20; % فرکانس نمونهبرداری (هرتز)
t_start = 0; % زمان شروع نمونهبرداری (ثانیه)
t_end = 1; % زمان پایان نمونهبرداری (ثانیه)
ts = 1/fs; % فاصله زمانی بین دو نمونه (ثانیه)
t = t_start : ts : (t_end - ts);
N = length(t); % تعداد کل نمونهها

x1 = exp(1i * 2 * pi * 5 * t) + exp(1i * 2 * pi * 8 * t);
x2 = exp(1i * 2 * pi * 5 * t) + exp(1i * 2 * pi * (5.1) * t);

y1 = fftshift(fft(x1)); % تبدیل فوریه و جابجایی فرکانسها برای سیگنال اول
y1 = y1 / max(abs(y1)); % نرمالسازی دامنه

y2 = fftshift(fft(x2)); % تبدیل فوریه و جابجایی فرکانسها برای سیگنال دوم
y2 = y2 / max(abs(y2)); % نرمالسازی دامنه

f = -fs/2 : fs/N : fs/2 - fs/N; % بردار فرکانسی

```

- ابتدا که مثل همیشه تمامی پنجره های باز را بسته، صفحه نمایش را پاک کرده و تمام متغیرها را پاک می کنیم.
- حالا فرکانس نمونه برداری داده شده که 20 هرتز است و در نتیجه فاصله زمانی بین دو نمونه که ts است، و زمان شروع و پایان نمونه برداری که 0 تا 1 است را تعریف کرده و بردار زمان را با نمونه برداری با فاصله ts می سازیم.
- سپس تعداد کل نمونه ها را که از روی $T=1$ و $fs=20$ هم میتوانستیم به آن برسیم، از روی طول تعداد نمونه های زمانی، می یابیم که در این مثال 20 است.

-
- سپس سیگنال ها را تعریف کرده و با فانکشن های گفته شده، تبدیل فوریه گرفته و آن ها را حول مبدأ می برمی و اندازه ی آن ها را هم نرمال سازی می کنیم.
 - سپس بردار فرکانس ها را با رزولوشن N/f_s می سازیم.

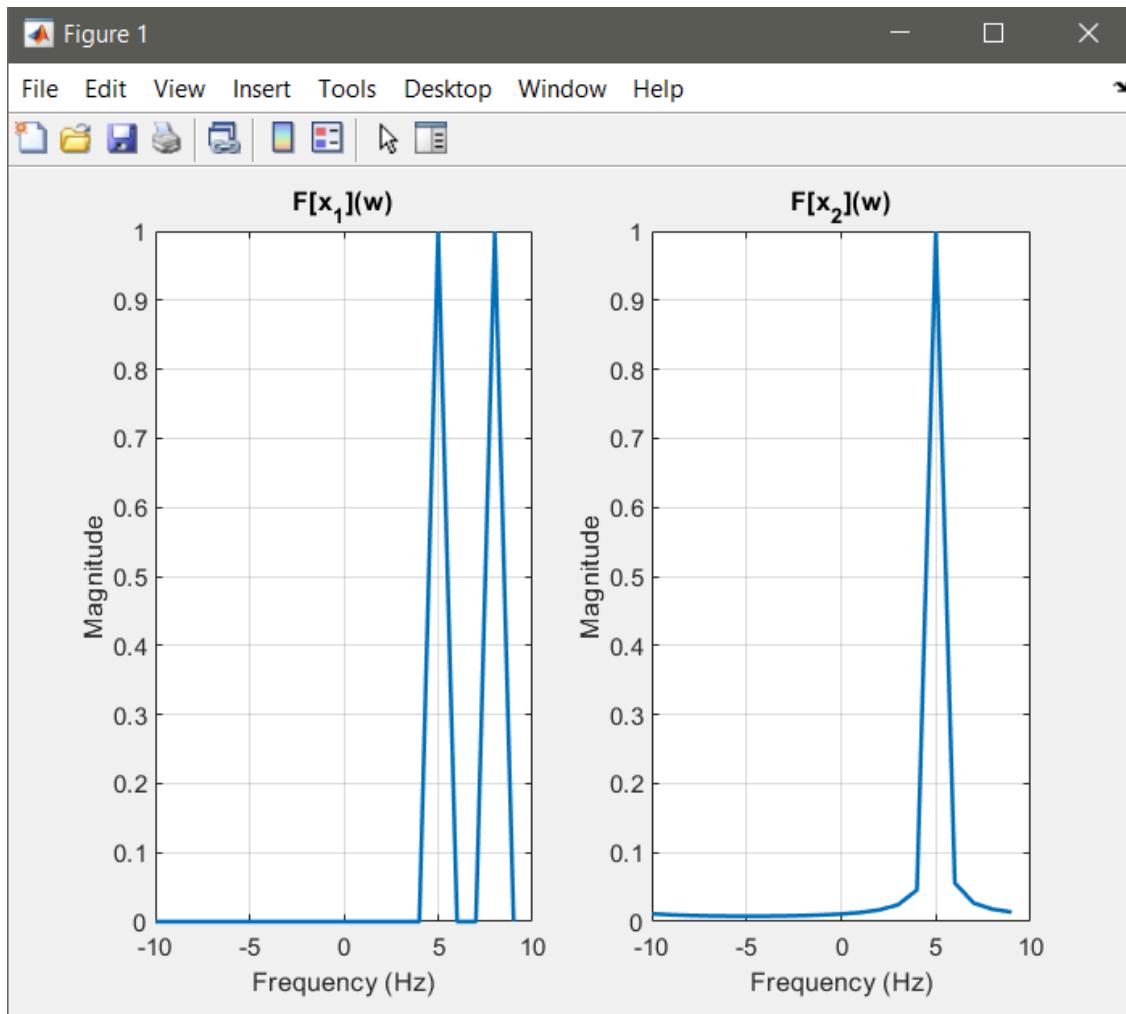
۸۸ Part 2: رسم نمودارها

```
figure;
```

```
subplot(1, 2, 1);
plot(f, abs(y1), 'LineWidth', 1.5);
title('F[x_1](w)');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;

subplot(1, 2, 2);
plot(f, abs(y2), 'LineWidth', 1.5);
title('F[x_2](w)');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
```

- سپس یک فیگور ساخته و در دو تا subplot هر دو نمودار را رسم می کنیم



- نهایتا همانطور که انتظار می رفت، مشاهده شد که در حالت اول دو قله در فرکانس 5 و 8 دیده شد اما در حالت دوم فقط یک قله در اندازه‌ی سیگنال در حوزه‌ی فوریه در فرکانس 5 می بینیم و دیگر توانایی تفکیک این دو سیگنال را در حوزه‌ی فوریه نداریم چون اختلاف فرکانس دو سیگنال تک تن از رزولوشن که 1 هرتز است کمتر بوده و 0.1 است.

(1-1) تمرین

(الف)

محاسبات دستی:

$$x(t) = \cos(10\pi t) \xrightarrow{\text{F}} X(\delta_{(w-10\pi)} + \delta_{(w+10\pi)})$$

$f = \pm 10\pi$ با همان $w = \pm 10\pi$ داشت

%% Part 0: پاکسازی محیط

close all;

clc;

clear;

%% Part 1: الف

fs = 50;

ts = 1 / fs;

t_start = -1;

t_end = 1;

t = t_start : ts : (t_end - ts);

x1 = cos(10 * pi * t);

رسم سیگنال در حوزه زمان

figure;

plot(t, x1, 'b', 'LineWidth', 0.5);

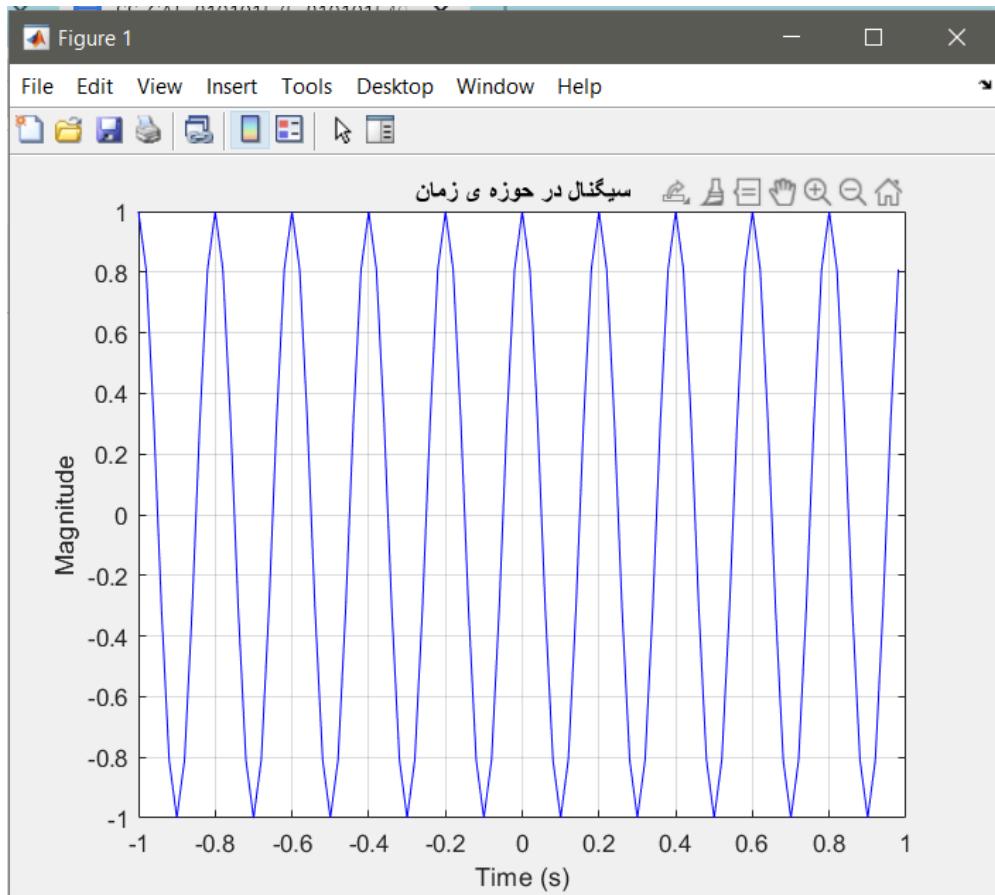
grid on;

xlabel('Time (s)');

ylabel('Magnitude');

title('سیگنال در حوزه زمان', 'FontWeight', 'bold');

- رسم ساده‌ی تابع نکته‌ی خاصی ندارد و مثل همیشه با تعریف فرکانس و فاصله‌ی بین سمپل‌ها و تعریف تابع و سپس رسم نمودار در فیگور انجام می‌شود:



(ب)

%% Part 2: ب

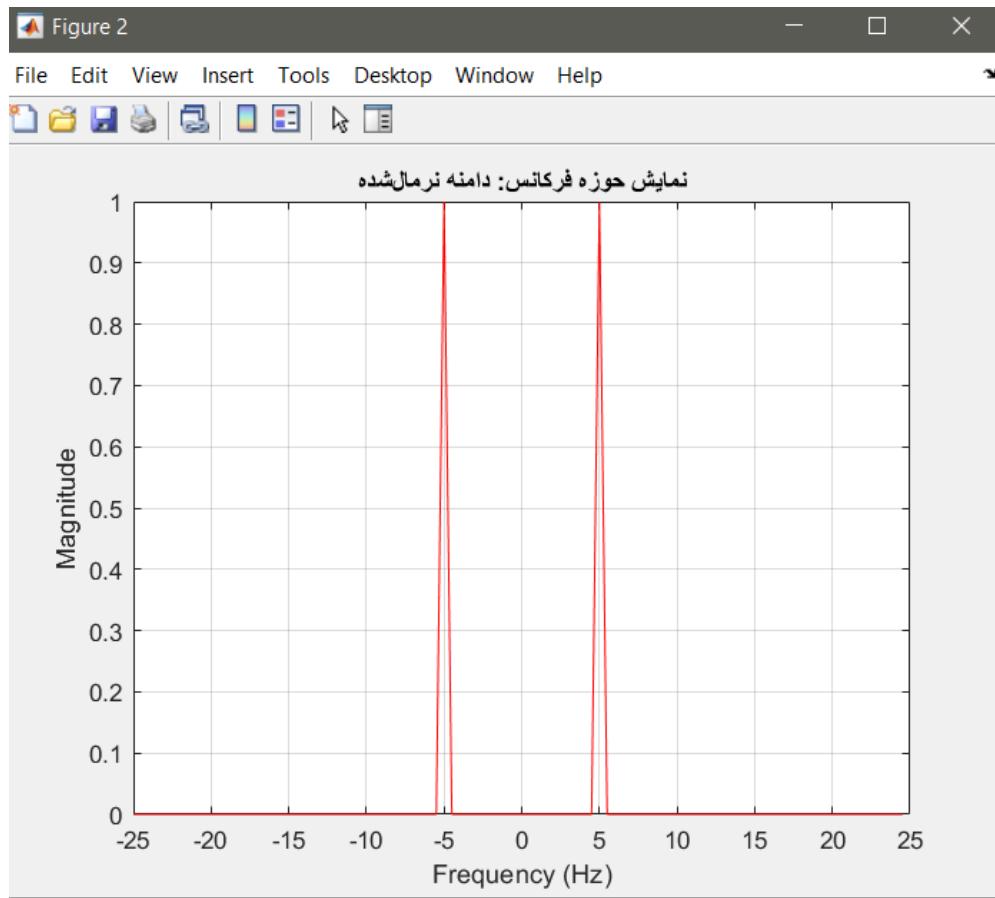
```
y1 = fftshift(fft(x1));
out = y1 / max(abs(y1)); % تبدیل فوریه نرمال شده (دامنه مقیاس شده به 1)

N = length(t); % تعداد کل نمونه ها

f = -fs/2 : fs/N : fs/2 - fs/N; % بردار فرکانسی

figure;
plot(f, abs(out), 'r', 'LineWidth', 0.5);
grid on;
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('نمایش حوزه فرکانس: دامنه نرمال شده', 'FontWeight', 'bold');
```

- حالا همانطور که در بخش صفر هم دیدیم، تبدیل فوریه را با توابع موجود محاسبه کرده و آن را جایه جا کرده و حول صفر می آوریم و سپس هم دامنه ای آن را اسکیل کرده و با تعریف بردار فرکانس ها با رزولوشن مدنظر، آن را رسم میکنیم بر حسب فرکانس.
- البته دقت شود که صرفا اندازه را رسم می کنیم با تابع `abs`، چون فعلا تحلیل هایمان روی اندازه بوده و اصلا تابع کسینوس چون حقیقی و زوج است، تبدیلش هم حقیقی و زوج است و فاز ندارد.



- نمودار اندازه‌ی تبدیل فوریه‌ی اینتابع هم طبق انتظارمان در محاسبات دستی، در ۵ و ۵- پیک میزند و چون فاصله‌ی ایندو فرکانس از رزولوشن که ۱ است بیشتر است، به راحتی قابل تمایز هستند.
- پس نتیجه‌ی نهایی کاملا با دانسته‌های ما و محاسباتمان تطابق دارد.

تمرین 1-2

محاسبات دستی:

$$\begin{aligned}
 H_{R(t)} &= C_1 \left(e^{j\omega t} \left(t + \frac{1}{f_R \tau_R} \right) \right) = C_1 \left(e^{j\omega t} \left(t + \frac{1}{f_R \cdot 10} \right) \right) \\
 \xrightarrow{F} e^{j\omega \frac{1}{f_R \cdot 10}} \quad f_{(C_1 e^{j\omega t})} &= e^{j\omega \frac{1}{f_R \cdot 10}} \delta(f_{(\omega - 10\pi)} + \delta_{(\omega + 10\pi)}) \\
 f = \frac{\omega}{2\pi} \rightarrow \hat{H}_{(f)} &= e^{-jf \frac{1}{10}} \delta(f_{(f-10)} + \delta_{(f+10)}) \\
 \rightarrow |\hat{H}_{(f)}| &= \sqrt{\delta_{(f-10)}^2 + \delta_{(f+10)}^2} \xrightarrow{f = \pm 10 \text{ پیک من زند و در قیسی نواحی صفر است}}
 \end{aligned}$$

سه نمودار اندازه‌ی تبدیل فوریه، در
 هر کدام از اندازه‌ها مقدار موج می‌باشد.
 جزوی از مقدار موج می‌باشد.

$$\begin{aligned}
 \rightarrow \angle \hat{H}_{(f)} &= \angle e^{-jf \frac{1}{10}} = f \frac{\pi}{4} \xrightarrow{f = -10 \Rightarrow \angle \hat{H} = \frac{-10\pi}{4} = -\frac{\pi}{2}} \\
 &\qquad\qquad\qquad f = 10 \Rightarrow \angle \hat{H} = \frac{10\pi}{4} = \frac{\pi}{2}
 \end{aligned}$$

سه انتشار داریم نمودار $\frac{|\hat{H}|}{\pi}$ در $f = 10$ مقدار $\frac{1}{2}$ بود.
 در $f = -10$ مقدار $\frac{1}{2}$ بود.

- نتیجه این است که اندازه‌ی تبدیل فوریه باید در $f = 15$ پیک بزند و در بقیه‌ی نقاط صفر باشد.

- از طرف دیگر بعلت وجود اندازه‌ی صفر و سبک تحلیل ما از فاز که در بخش ج خواهیم دید، فاز هم در تمام نقاط صفر است بجز $f = 15$ که مقدار $\frac{\pi}{4}$ میدهد و $f = 15$ که مقدار $\frac{3\pi}{4}$ می‌دهد.

الف)

%% Part 0: باکسازی محیط

```
close all;  
clc;  
clear;
```

%% Part 1: الف

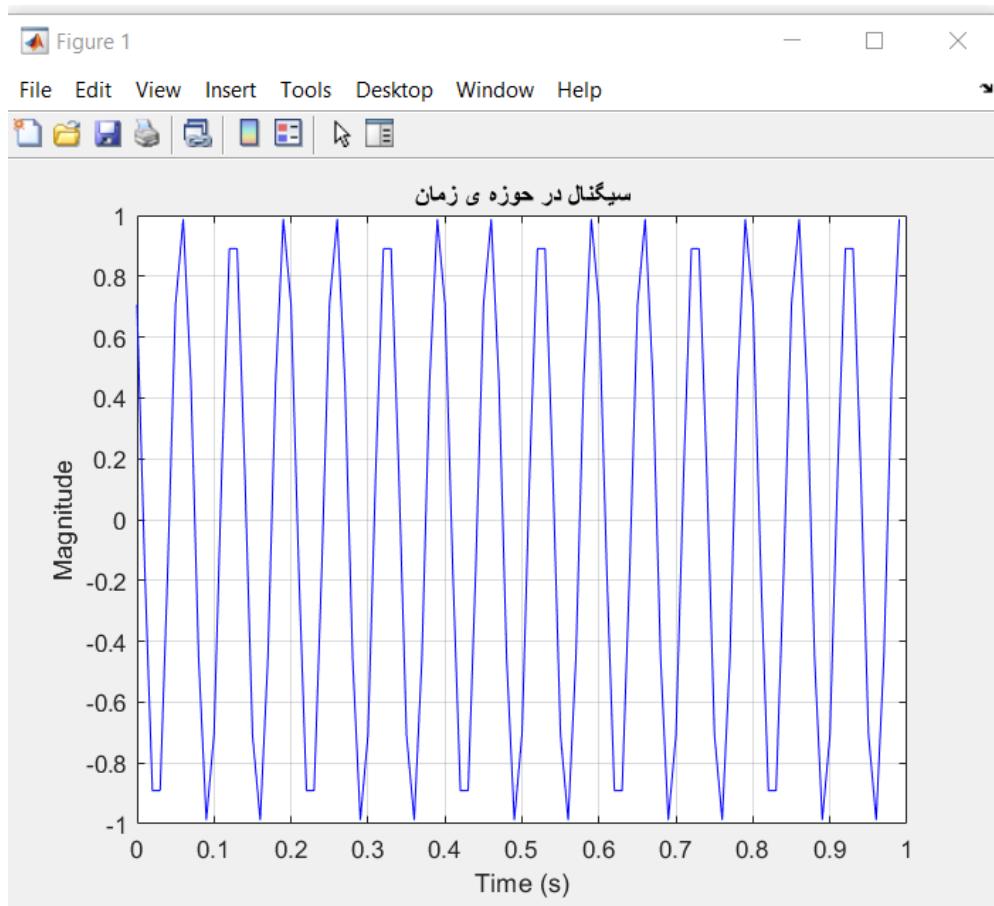
```
fs = 100;  
ts = 1 / fs;  
  
t_start = 0;  
t_end = 1;  
t = t_start : ts : (t_end - ts);  
  
x2=cos(30*pi*t + pi/4);
```

% رسم سیگنال در حوزه زمان

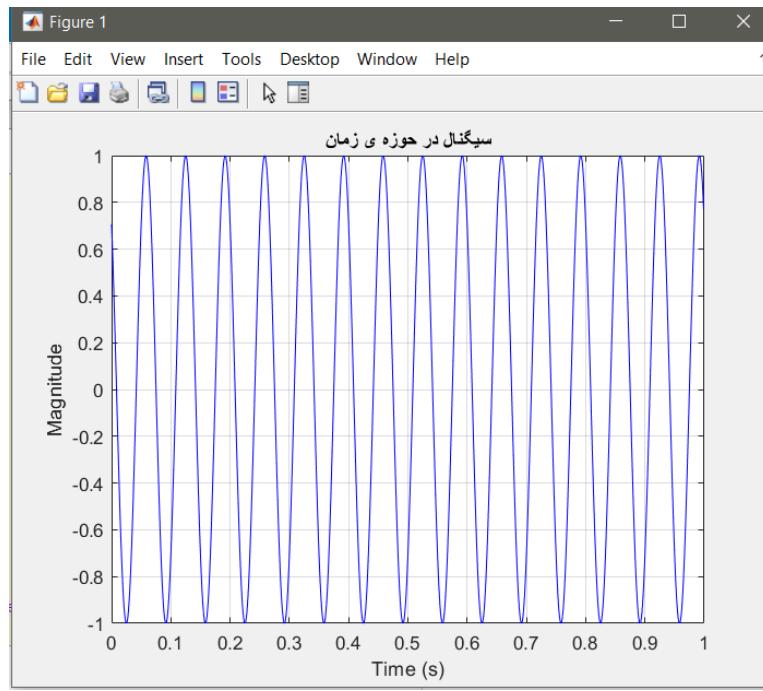
```
figure;  
plot(t, x2, 'b', 'LineWidth', 0.5);  
grid on;  
xlabel('Time (s)');  
ylabel('Magnitude');  
title('سیگنال در حوزه زمان', 'FontWeight', 'bold');
```

- در این بخش کار جدیدی نمیکنیم و دقیقاً مثل بخش های قبلی با توجه به دستورات

صورت سوال، تابع را در حوزه زمان رسم می کنیم:



- نمودار مشاهده شده، همان چیزی بود که انتظار داشتیم، یعنی $\cos(30\pi t + \pi/4)$
- دلیل مشکل جزیی در رسم پیک های نمودار هم فرکانس نمونه برداری است که مثلا اگر آن را به 1000 بررسانیم، رزولوشن نمودار بهتر می شود (در بخش قبلی هم همین مورد به چشم می خورد، اما چیز مهمی نیست در کل):



(ب)

%% Part 2: ب

```

y1 = fftshift(fft(x2));
out = y1 / max(abs(y1)); % تبدیل فوریه نرمال شده (دامنه مقیاس شده به 1)

N = length(t); % تعداد کل نمونه ها
% روشن دوم تعریف
% T = (t_end - t_start);
% N = T/ts;

f = -fs/2 : fs/N : fs/2 - fs/N; % بردار فرکانسی

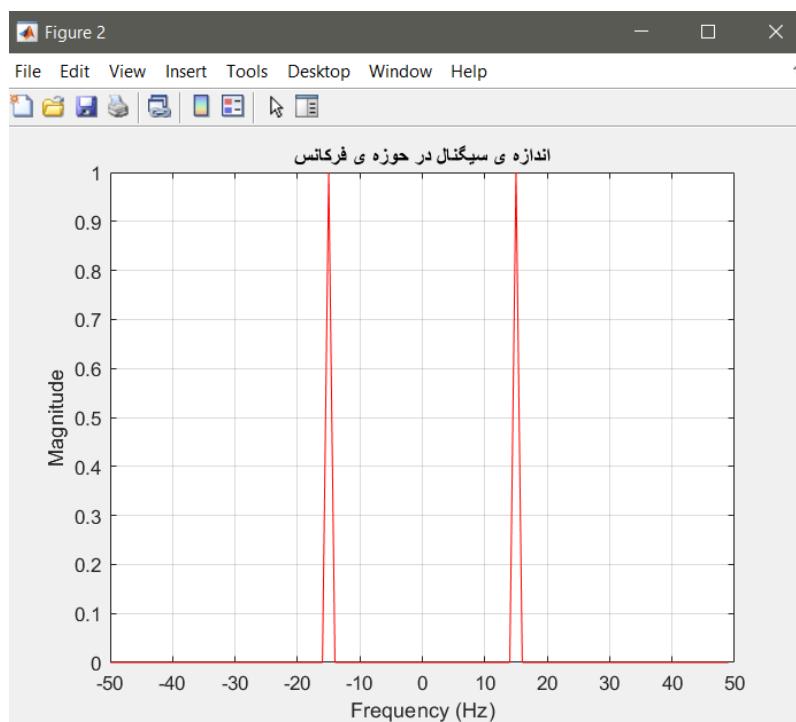
figure;
plot(f, abs(out), 'r', 'LineWidth', 0.5);
grid on;
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('نمایش حوزه فرکانس: دامنه نرمال شده', 'FontWeight', 'bold');

```

- در این بخش باز هم مانند قبل عمل کرده و با تعریف بردار فرکانس ها و گرفتن تبدیل فوریه و شیفت دادن و نرمال کردن آن، آن را رسم می کنیم.
- نکته ای که اینجا دیدیم و جالب بود، نحوه ای ذخیره ای وکتوری از اعداد مختلط مثل `out` در متلب بود که داریم:

	11	12	13	14	15	
1	6... -1.8976e-1...	8.6546e-18...	-2.2642e-1...	-1.9896e-1...	-3.3079e-1...	2.
2						
3						

- نهایتا `plot` نهایی به این شکل است:

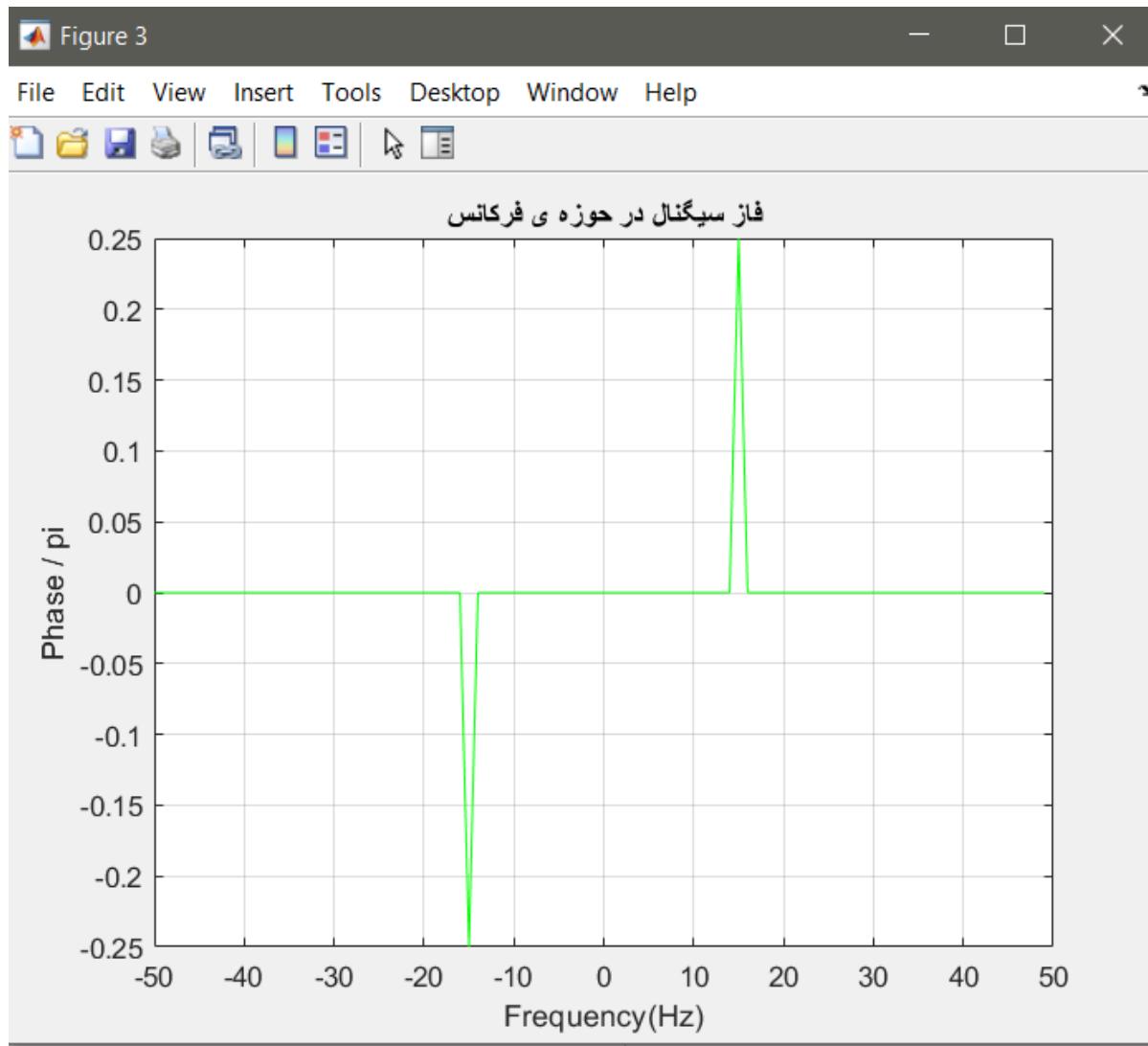


- همانطور که مشاهده می شود، نتیجه دقیقا طبق انتظارمان است، یعنی همانطور که در برگه حساب کردیم، اندازه f تبدیل فوریه‌ی این سیگنال، در نقاط $f = 15$ و $f = -15$ پیک می‌زند و در بقیه‌ی نواحی، صفر است.

(ج)

```
%% Part 3: ج
tol=1e-6;
out(abs(out)<tol)=0;
theta=angle(out);
figure
plot(f,theta/pi)
grid on;
xlabel('Frequency(Hz)')
ylabel('Phase / pi')
```

- طبق چیزی که در دستور پروژه راهنمایی شده بود عمل کردیم و ابتدا فاز جاها بی که اندازه تقریباً صفر بوده است را صفر کردیم، و سپس با تابع angle فاز را بدست آورده و فاز تقسیم بر pi را رسم می‌کنیم تا از شر عدد پی خلاص شویم در نمودار، و نمودار حاصل این می شود:



- باز هم نتیجه دقیقاً مطابق با محاسبات دستی در کاغذ بود، که همانطور که انتظار میرفت، در $f = 15$ مقدار $\frac{1}{4}$ را در نمودار مشاهده می‌کنیم.
- در نهایت هم نمودار فاز و اندازه هر دو با دانش ما مطابق بودند زیرا با اعداد و نتایج محاسبه شده در ابتدا روی برگه، یکسان بودند.

بخش دوم

(1-2) تمرین

```
%% Part 0
close all;
clc;
clear;
```

- پیش از هر کاری تمام فیگورهای باز را بسته، متغیرها و ترمینال را هم کاملا پاک می‌کنیم.

```
%% Part 1
% Define our char array which contains alphabet and additional characters
chars = ['a':'z', ' ', '.', ',', '!', "'", ";"];
% chars_len = length(chars); % its 32

% Creating a mapset with 2 rows and columns :
% Row 1 is the characters, Row 2 is the binary coded value of each character
Mapset = cell(2, 32);

% Set the values into Mapset
for i = 1:32
    % Store the character in the first row of Mapset
    Mapset{1, i} = chars(i);

    % Add the binary coded value for each character (i-1 in binary)
    Mapset{2, i} = dec2bin(i - 1, 5);
end
```

- حالا ابتدا تمام کاراکترهای مدنظر در سوال را در یک رشته به طول 32 بصورت زیر میریزیم:

 chars
'abcdefghijklmnopqrstuvwxyz .,!;'

- سپس یک سلول 2 در 32 برای نگهداری از کاراکترها و کد 5 بیتی متناظر آنها می‌سازیم به نام

.Mapset

- حالا در یک حلقه، روی تک ستون های Mapset حرکت کرده و هر بار در ردیف اول، یک کاراکتر گذاشته و در ردیف دوم هم کد 5 بیتی متناظر آن را قرار می دهیم که این کد شامل اعداد باینری ۰ تا ۳۱ است.

- نهایتاً Mapset بصورت زیر خواهد بود:

9	10	11	12	13	14	15
i	j	k	l	m	n	o
01000	01001	01010	01011	01100	01101	01110

(2-2) تمرین

```
%% part 2
function Coded_signal=coding_amp(message, BitRate, mapset, show_plt)

fs=100;
% Convert message to binary sequence using mapset
message_bin = '';
for k = 1:length(message)
    char_index = find(strcmp(mapset(1,:), message(k))); % Find the character index in mapset
    if isempty(char_index)
        error(['Character ', message(k), ' not found in mapset.']);
    end
    char_bin = mapset(2, char_index); % Get binary representation of each character
    message_bin = [message_bin, char_bin]; % Append to the binary message sequence
end
Coded_signal=[];
message_bin_len = length(message_bin);

t=0.01:(1/fs):1;
a = [];
a = [a,zeros(1,length(t))];
```

- در ابتدا فرکانس نمونه برداری تعریف شده و متغیر message_bin برای ذخیره کردن پیام باینری شده‌ی نهایی استفاده می‌کنیم و یک حلقه‌ی می‌زنیم و تک تک کاراکتر‌های پیام ابتدایی را برداشته و تلاش می‌کنیم آن را در ردیف اول Mapset پیدا کنیم. اگر پیدا نشد که ارور بر می‌گردانیم و اگر پیدا شد، معادل باینری آن کاراکتر را از ردیف دوم Mapset برداشته و به message_bin اضافه می‌کنیم.

```

100 -
109 - fs = 100;
110 - ts = 1/fs;
111 - t = 0 : ts : (1-ts);
112 -
113 - frequency=cell(1,5);
114 - frequency{1,1}=[12,37];
115 - frequency{1,2}=[5,16,27,38];
116 - frequency{1,3}=[4,10,16,22,28,34,40,46];
117 - frequency{1,4}=[2,5,8,11,14,17,20,23,26,29,32,35,38,41,44,47];
118 - frequency{1,5}=[1,2,4,5,7,8,10,11,13,14,16,17,19,20,22,23,25,26,28,29,31,32,34,35,37,38,40,41,43,44,46,47];
119 -
120 - f_i = cell2mat(frequency(speed));
121

```

سپس فرکانس نمونه برداری، ts و بردار t که از صفر تا یک هست را تعریف می‌کنیم. بعد از آن یک $cell$ به این صورت تعریف می‌کنیم که ۵ خانه برای ۵ بیت ریت ما دارد و سپس در هر خانه آرایه‌ای از فرکانس‌ها با سایز ۲^۸(شماره خانه) دارد که فرکانس‌های انتخابی ما از بین ۰ تا ۴۹ برای کدگذاری پیام است و در آخر هم f_i هایی که استفاده می‌کنیم را به کمک متغیر $speed$ یا همان BitRate انتخاب می‌کنیم.

```

% Creating the coded signal by choosing the right frequency
Coded_signal=[];
for i=1:speed:length(message_bin)
    code_number = bin2dec(message_bin(i:i+speed-1));
    Coded_signal = [Coded_signal sin(2*pi*f_i(code_number+1)*t)];
end

```

در مرحله بعد نوبت به ساختن سیگنال رمز شده است. به این صورت است که از پیام باینری شده در هر بار اجرای حلقه، ایندکس i تا $i+speed-1$ که در واقع به تعداد $speed$ بیت می‌شود را بر می‌داریم و سپس آن را به مقدار دسیمال تبدیل می‌کنیم. این عدد به علاوه یک (چون ایندکس‌های متباعد از یک شروع می‌شوند) برابر با ایندکس فرکانس معادل کدگذاری شده بیت‌های برداشته شده از پیام باینری است و سپس به مدت یک ثانیه (با توجه به تعریف t ، یک تابع سینوسی به صورت $\sin(2\pi f_i t)$ که در آن f_i انتخاب شده است) را به سیگنال کد شده اضافه می‌کنیم.

```

    % Plot the result if needed
    if (show_plt == 1)
        x_axis=zeros(1,length(message_bin)/speed * 100);
        for i=1:length(message_bin)/speed*100
            x_axis(i)=i/100;
        end
        figure;
        plot(x_axis,Coded_signal)
    end
end

```

در آخر هم اگر نیاز بود به کمک متغیر show_plt، تابع کدگذاری شده را نمایش می‌دهیم.

تمرین 2 (3-2)

حال با فراخوانی تابع coding_freq و دادن مقدار یک به show_plt، پیام signal را رمز گذاری کرده و با دو بیت ریت 1 و 5 نمایش می‌دهیم.

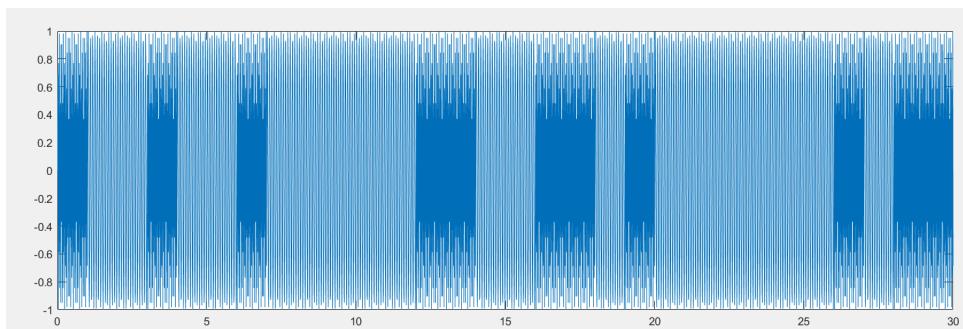
```

%% Part 3
message = 'signal';
coded_signal1 = coding_freq(message, 1, Mapset, 1);
coded_signal2 = coding_freq(message, 5, Mapset, 1);

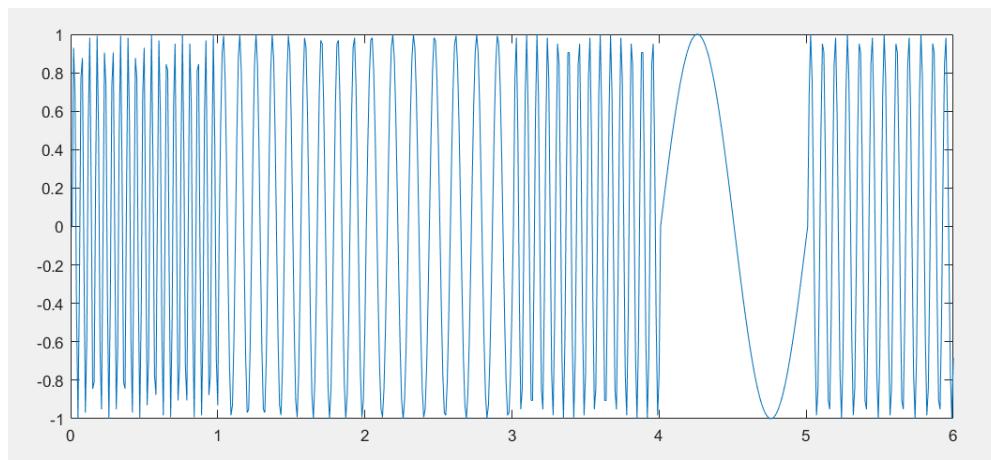
```

شکل سیگنال های کدگذاری شده به صورت زیر است

:BitRate =1 برای



:BitRate =5 برای



(4-2 تمرین

```
140
141 %% Part 4
142 function Decoded_message = decoding_freq(Coded_signal, speed, mapset)
143 fs = 100;
144
145 frequency=cell(1,5);
146 frequency{1,1}=[12,37];
147 frequency{1,2}=[5,16,27,38];
148 frequency{1,3}=[4,10,16,22,28,34,40,46];
149 frequency{1,4}=[2,5,8,11,14,17,20,23,26,29,32,35,38,41,44,47];
150 frequency{1,5}=[1,2,4,5,7,8,10,11,13,14,16,17,19,20,22,23,25,26,28,29,31,32,34,35,37,38,40,41,43,44,46,47];
151
152 f_i = cell2mat(frequency(speed));
153
```

در ابتدا مانند بخش قبل f_i را تعیین می‌کنیم.

```

153
154     % Find the frequency of each 1sec in the coded signal
155     freq_indexes = [];
156     for i = 1:fs:length(Coded_signal)
157         X = Coded_signal(i:fs+i-1);
158         Y = fftshift(fft(X));
159         Y = abs(Y/max(Y));
160         vals_selected_f = [];
161         for j= 1:length(f_i)
162             vals_selected_f = [vals_selected_f Y(51+f_i(j))];
163         end
164         [val idx] = max(abs(vals_selected_f));
165         freq_indexes = [freq_indexes idx];
166     end
167

```

سپس به پیدا کردن پیام باینری می‌پردازیم. به این صورت این کار را می‌کنیم که در هر مرحله اجرای حلقه یک ثانیه از سیگنال کد گذاری شده را بر می‌داریم سپس با استفاده از `fft` و `fftshift` تبدیل فوریه این یک ثانیه را می‌گیریم و در ۷ ذخیره می‌کنیم، سپس ۷ را نرمالایز کرده و اندازه آن را ذخیره می‌کنیم (زیرا ماکسیمم اندازه را می‌خواهیم و با فاز ها کاری نداریم). سپس با توجه به `i_f` انتخاب شده مقدار ۷ را در فرکانس های انتخاب شده پیدا می‌کنیم (ایندکس ۵۱ ام مربوط به فرکانس ۰ است و چون ما با بخش مثبت `f` سر و کار داریم از این ایندکس به بعد را چک می‌کنیم) و ذخیره می‌کنیم. از بین مقادیر مربوط به فرکانس های انتخاب شده، فرکانسی مدنظر ما است که بیشترین مقدار را دارد پس ایندکس این فرکانس را پیدا می‌کنیم (که معادل با ایندکس آن در آرایه `i_f` می‌باشد) و آن را ذخیره می‌کنیم تا در مرحله بعد از آن استفاده کنیم.

```

167
168     % Finding the binary message
169     message_bin=[];
170     for i = 1 : length(freq_indexes)
171         message_bin = [message_bin dec2bin(freq_indexes(i)-1, speed)];
172     end
173

```

سپس ایندکس ها را منهای یک کرده (چون رمز های ما از صفر شروع می‌شوند ولی ایندکس ها از یک) و با توجه به `speed` به باینری تبدیل می‌کنیم و به ادامه پیام باینری استخراج شده اضافه می‌کنیم.

```

173
174 -     Decoded_message = [];
175 -     for i = 1 : length(message_bin)/5
176 -         ch = message_bin(5*i-4 : 5*i);
177 -         number = bin2dec(ch);
178 -         Decoded_message = [Decoded_message mapset{1, number+1}];
179 -     end
180 - end
181

```

سپس در آخر، یک حلقه داریم که هر بار 5 تا بیت از پیام باینری جدا می‌کند و سپس آن را به مقدار دسیمال تبدیل می‌کند و در آخر کاراکتر مربوط به این عدد را که در ایندکس number+1 ردیف اول قرار دارد، به پیام دیکود شده اضافه می‌کند.

حال سیگنال های کد گذاری شده را به تابع decode_freq می‌دهیم و پیام های دیکود شده را نمایش می‌دهیم.

```

22 %% Part 3
23 - message = 'signal';
24 - coded_signal1 = coding_freq(message, 1, Mapset, 1);
25 - coded_signal2 = coding_freq(message, 5, Mapset, 1);
26
27 %% Part 4
28 - msg1 = decoding_freq(coded_signal1, 1, Mapset);
29 - msg2 = decoding_freq(coded_signal2, 5, Mapset);
30
31 - disp(['decoded msg1 is: ', msg1, ' decoded msg2 is: ', msg2]);
32

```

همانطور که دیده می‌شود تابع ما به درستی کار می‌کند و پیام به درستی رمزگشایی می‌شود.

Command Window

New to MATLAB? See resources for [Getting Started](#).

```

decoded msg1 is: signal decoded msg2 is: signal

```

(5-2) تمرین

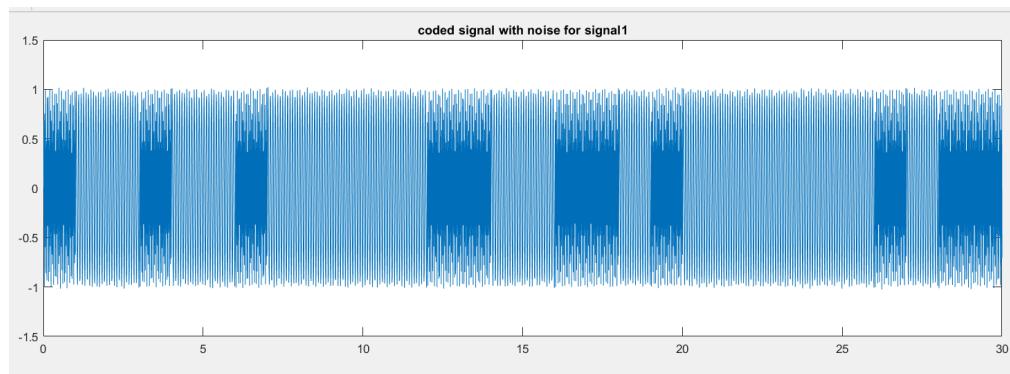
حال ابتدا با BitRate=1 دوباره پیام signal را کد گذاری می‌کنیم سپس به سیگنال خروجی به وسیله تابع randn یک نویز گوسی با انحراف معیار 0.01 (واریانس 0.0001) و میانگین صفر اضافه می‌کنیم.

سپس محور افقی را به این صورت می سازیم که یک ماتریس 1 در طول سیگنال کد گذاری شده است و مقدار هر درایه آن را برابر ایندکس آن تقسیم بر 100 (بین دلیل که در هر ثانیه صد نمونه داریم) می گذاریم. سپس سیگنال کد گذاری شده به همراه نویز را پلات می کنیم و بعد از آن سیگنال نویزدار را به تابع دیکود می دهیم تا آن را دیکود کند و پیام استخراج شده را نمایش می دهیم. همین کار را برای BitRate تابع دیکود می دهیم تا آن را دیکود کند و پیام استخراج شده را نمایش می دهیم. 5 نیز انجام می دهیم.

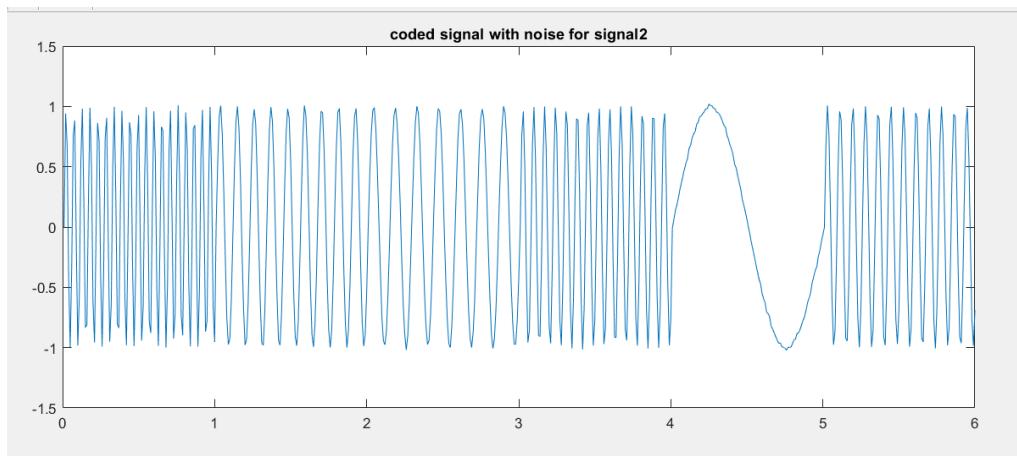
```
52
53 %% Part 5
54
55 % speed = 1
56 message = 'signal';
57 coded_signal1 = coding_freq(message, 1, Mapset, 0);
58 coded_signal1 = coded_signal1 + 0.01*randn(1, length(coded_signal1)); % Add noise
59 x_axis=zeros(1,length(coded_signal1));
60 for j=1:length(coded_signal1)
61     x_axis(j)=j/100;
62 end
63 figure;
64 plot(x_axis,coded_signal1)
65 title('coded signal with noise for signal1');
66 decoded_msg1 = decoding_freq(coded_signal1, 1, Mapset);
67 disp(['decoded message for coded signal with noise is:',decoded_msg1, ' for speed = 1']);
68
69 % speed = 5
70 coded_signal2 = coding_freq(message, 5, Mapset, 0);
71 coded_signal2 = coded_signal2 + 0.01*randn(1, length(coded_signal2)); % Add noise
72 x_axis=zeros(1,length(coded_signal2));
73 for j=1:length(coded_signal2)
74     x_axis(j)=j/100;
75 end
76 figure;
77 plot(x_axis,coded_signal2)
78 title('coded signal with noise for signal2');
79 decoded_msg2 = decoding_freq(coded_signal2, 1, Mapset);
80 disp(['decoded message for coded signal with noise is:',decoded_msg2, ' for speed = 5']);
81
82
```

خروجی ها به صورت زیر می باشد

برای BitRate=1 سیگنال نویز دار به صورت زیر است.



برای BitRate=5 سیگنال نویز دار به صورت زیر است.



و پیام ها هم در هر دو BitRate به درستی رمزگشایی می شود.

decoded message for coded signal with noise is: signal for speed = 1
decoded message for coded signal with noise is: signal for speed = 5

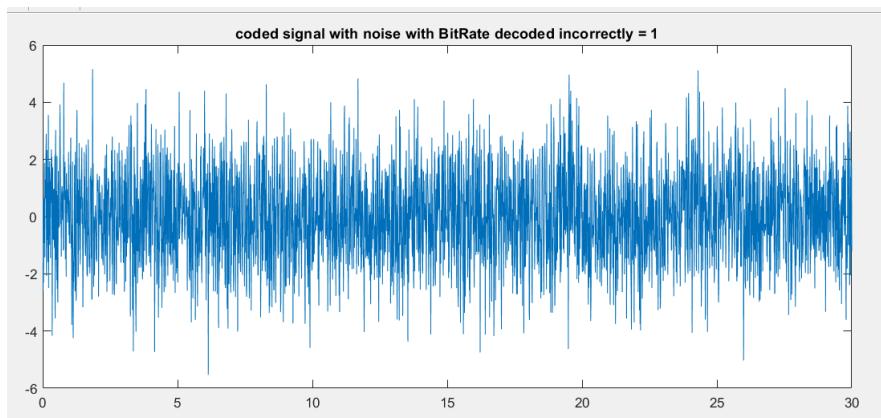
تمرین 2-6 و 2-7

```
63 %% Part 6 and 7
64 message = 'signal';
65 for Speed=1:4:5
66 sum_var = 0;
67 max_var = 0;
68 for i=1:300
69 std = 0.01;
70 decoded_message = 'signal';
71 while message == decoded_message
72 coded_signal = coding_freq(message, Speed, Mapset, 0);
73 coded_signal = coded_signal + std*randn(1, length(coded_signal));
74 decoded_message=decoding_freq(coded_signal, Speed, Mapset);
75 if message == decoded_message
76 std = std + 0.01;
77 end
78 end
79 sum_var = sum_var + (std^2);
80 if max_var < (std-0.01)^2
81 max_var = (std-0.01)^2;
82 end
83 end
84 x_axis=zeros(1,length(coded_signal));
85 for j=1:length(coded_signal)
86 x_axis(j)=j/100;
87 end
88 figure;
89 plot(x_axis,coded_signal)
90 title(['coded signal with noise with BitRate decoded incorrectly = ', num2str(Speed)]);
91 disp(['decoded_message, ' for BitRate = ', num2str(Speed), ' mean noise variance = ', num2str(sum_var/300)]);
92 disp(['Max variance for BitRate = ', num2str(Speed), ' decodes correctly is about', num2str(max_var)]);
93 end
```

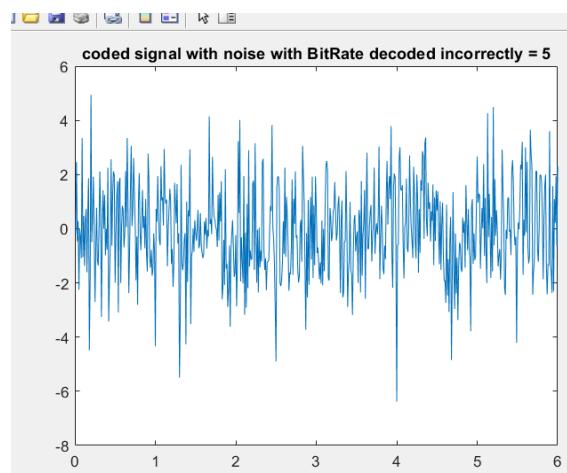
مثل تمرین قبل برای این بخش ابتدا پیام را تعریف کرده سپس در یک حلقه فور که به Speed دو مقدار 1 و 4 را می‌دهد، به این صورت عمل می‌کنیم. ابتدا مقادیر sum_var و max_var را تعریف می‌کنیم تا به ترتیب مقدار ماکریم واریانسی که تابع دیکود پیام را اشتباه تشخیص می‌دهد و مجموع تمام واریانس هایی که منجر به اشتباه شدن پیام دیکود شده می‌شود را ذخیره کنیم. سپس برای اینکه تاثیر رندوم بودن نویز را کمتر کنیم در یک حلقه‌ای که 300 بار اجرا می‌شود، هر بار از انحراف معیار 0.01 شروع کرده و در حلقه واپل پیام را رمزگذاری کرده و به سیگنال رمزگذاری شده نویز اضافه می‌کنیم. سپس پیام را توسط تابع دیکود رمزگشایی می‌کنیم اگر به درستی رمزگشایی شد مقدار انحراف معیار را 0.01 اضافه می‌کنیم. این کار را آنقدر تکرار می‌کنیم تا پیام استخراج شده اشتباه باشد سپس مقادیر sum_var و max_var را آپدیت می‌کنیم. در آخر هم آخرین سیگنال نویزی که اشتباه تشخیص داده شده را به همراه پیام اشتباهی که دیکود شده گزارش می‌کنیم.

خروجی ها به صورت زیر است.

سیگنال نویز دار برای BitRate=1



سیگنال نویز دار برای BitRate=5



مقادیر میانگین واریانس و ماکسیمم واریانس هم به صورت زیر گزارش شده.

```
message for coded signal with noise variance for signal for BitRate = 1 mean noise variance = 2.3033  
Max variance for BitRate = 1 decodes correctly is about 3.2041  
signal for BitRate = 5 mean noise variance = 1.8697  
Max variance for BitRate = 5 decodes correctly is about 2.6569
```

که همانطور که انتظار می‌رفت BitRate=1 نسبت به نویز مقاوم تر است زیرا فاصله بین فرکانس‌هایی که برای کد گذاری انتخاب شده اند بیشتر است. اما در BitRate=5 این فاصله بین فرکانس‌ها کمتر است و در نتیجه نسبت به نویز حساس تر است.

(8-2) تمرین

همانطور که در شرح پرتو نیز نوشته شده است هرچه فاصله‌ی بین فرکانس‌های انتخاب شده بیشتر باشد، کد گذاری انجام شده مقاومت بیشتری در برابر نویز خواهد داشت. این به این معناست که با افزایش پهنای باند، می‌توان اطلاعات بیشتری را با سرعت بالاتری منتقل کرد. در چنین شرایطی، داده‌ها در برابر اثرات نویز مقاوم‌تر باقی می‌مانند، زیرا فضای بیشتری برای تکیک و بازسازی سیگنال‌ها در اختیار سیستم قرار می‌گیرد.

(9-2) تمرین

خیر، افزایش نرخ نمونه‌برداری به تنهایی (بدون افزایش پهنای باند مورد استفاده) مقاومت در برابر نویز را افزایش نمی‌دهد. دلیل این امر آن است که فاصله‌ی بین فرکانس‌های انتخابی نقشی کلیدی در مقاومت در برابر نویز دارد. اگر این فاصله‌ها ثابت بمانند (یعنی پهنای باند افزایش نیابد)، صرفاً با افزایش نرخ نمونه‌برداری دقت نمایش سیگنال بیشتر می‌شود، اما نسبت سیگنال به نویز و مقاومت در برابر آن بهبود نخواهد یافت.