

دانشگاه صنعتی خواجه نصیرالدین طوسی

گزارشکار تمرین کامپیوتری سری سوم

درس پردازش سیگنال‌های دیجیتال

آنائیس گلبداغیانس ۴۰۱۲۲۱۱۳

سوال اول

در این سوال از همان تابعی که برای تمرینات کامپیوتری سری قبل استفاده شد، استفاده شده است.

```
function fftANDplot(x)
    X = fft(x);
    figure
    % w_sample=1/length(x)*(0:length(x)-1);
    subplot(1,2,1)
    stem(abs(X),"LineWidth",1);
    xlabel('\omega');
    ylabel(' |H(e^{j\omega})| ')
    hold on
    subplot(1,2,2)
    stem(angle(X),"LineWidth",1);
    xlabel('\omega');
    ylabel('arg(H(e^{j\omega}))')
end
```

نام تابع تغییر پیدا کرده، تبدیل فوریه داخل تابع محاسبه می‌شود. فرق این تابع با تمرینات سریع قبل این است که دیگر نمی‌خواهیم تبدیل فوریه پیوسته رسم کنیم؛ پس بخش امگا را کامنت کرده و بهجای `stem` از `plot` استفاده شده تا نمایش گسسته داشته باشیم.

```
%3rd computer asignment, DSP
%Anaies Golboudaghians 40122113
```

```
%Q1
clc; clear; close all; close all hidden

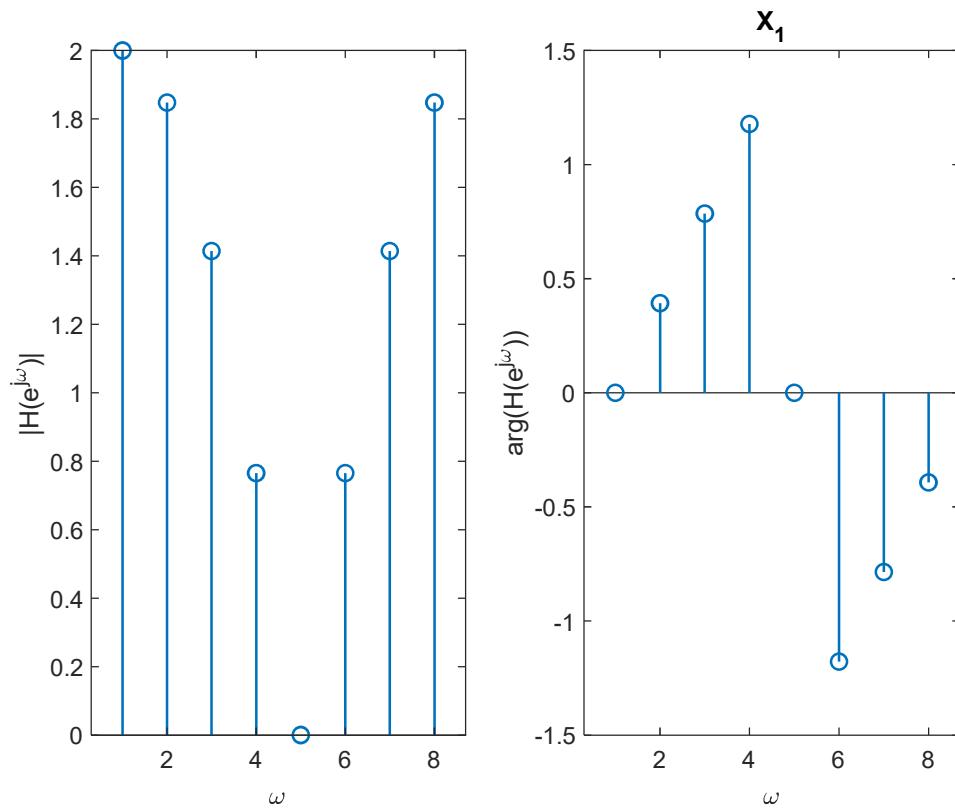
x1 = [1 0 0 0 0 0 1];
fftANDplot(x1);
title('X_1');

x2 = [0 1 1 0 0 0 -1];
fftANDplot(x2);
title("X_2")

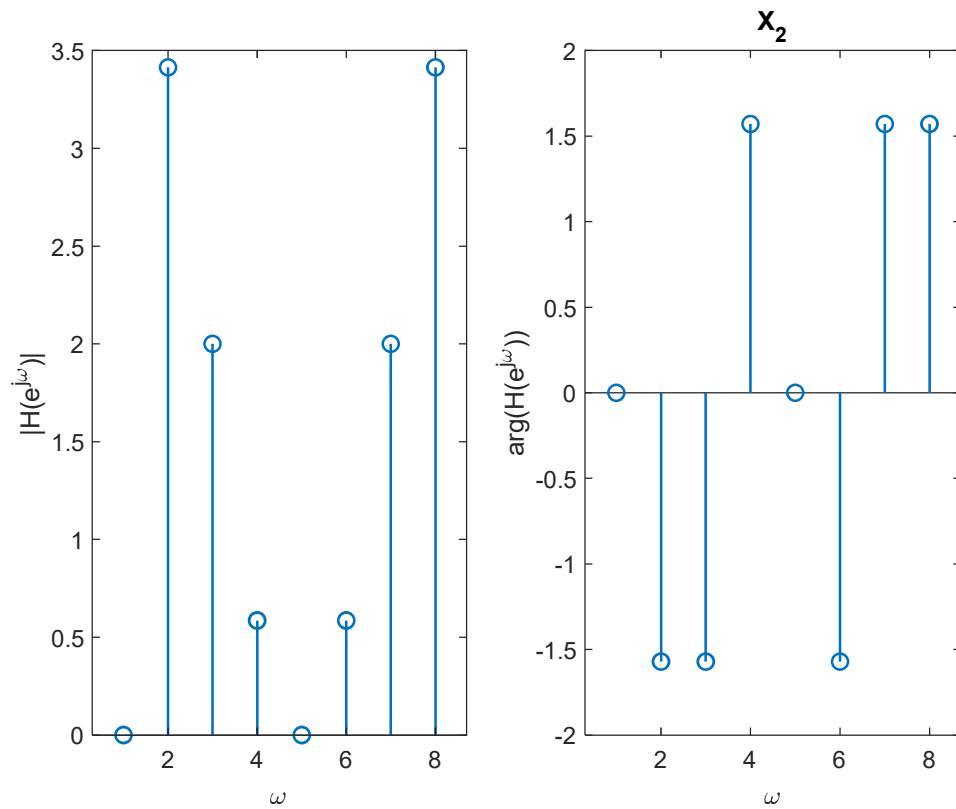
x3 = [1 1 1 1 1 1 1 1];
fftANDplot(x3);
title("X_3")
```

بردارهای سوال را تعریف می‌کنیم و از تابع نوشته شده استفاده می‌کنیم.

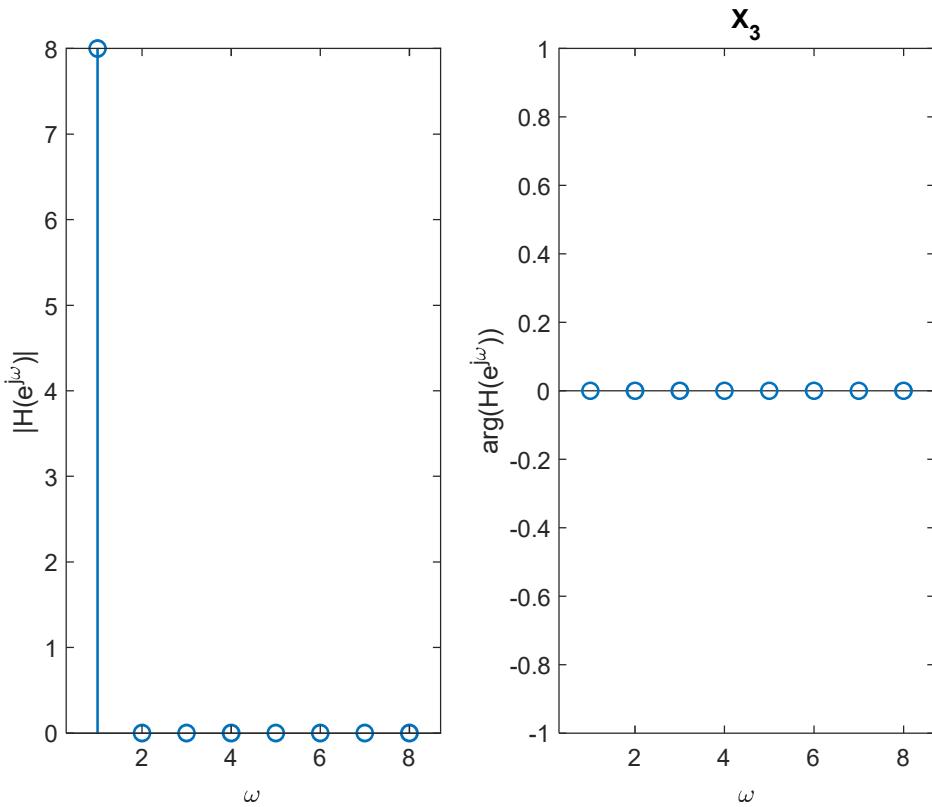
خروجی:



همان‌طور که در x_1 تقارن داشتیم، در حوزه فرکانس هم همان تقارن را مشاهده می‌کنیم.



اندازه متقارن و فاز پادمتری می باشد.



سوال دوم

بخش اول

ابتدا دنباله طول محدود را تولید می‌کنیم. صفرها را با دستور zeros تولید می‌کنیم و به صورت زیر آرایه‌ها را به هم وصل می‌کنیم.

```
%3rd computer asignment, DSP
%Anaies Golboudaghians 40122113
```

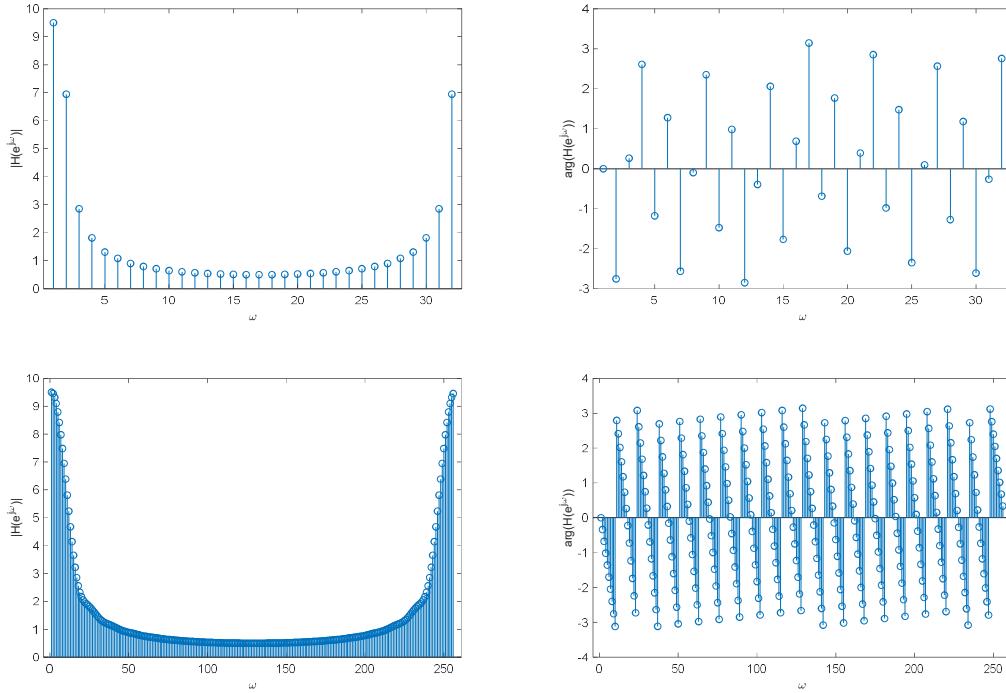
```
%Q2
clc; clear; close all; close all hidden

%% Part 1
n = 0:19;
x= 0.5.* (1-cos(pi.*n./20));
zs = zeros(1,12);
x1 = [x, zs];
fftANDplot(x1);

zs = zeros(1,236);
x2 = [x, zs];
fftANDplot(x2);
```

از همان تابع سوال قبل برای نمایش حوزه فوریه استفاده می کنیم.

خروجی:



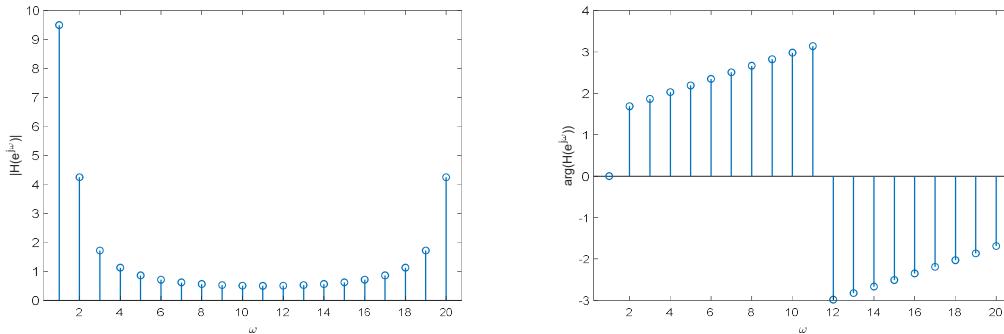
مشاهده می کنیم که با افزایش صفرها، به گونه ای که تعداد نمونه ها توان دو باشد، پیوستگی نمایش فوریه بیشتر می شود. در واقع zero padding به همین کار گفته می شود. نتیجه آن اعتبار و دقیقیت بیشتر نمایش فوریه است.

بخش دوم

```
%% Part 2
zs = zeros(1,16);
x3 = [zs, x];
fftANDplot(x);

function fftANDplot(x)
    X = fft(x);
    figure
    % w_sample=1/length(x)*(0:length(x)-1);
    subplot(1,2,1)
    stem(abs(X), "LineWidth",1);
    xlabel('j\omega');
    ylabel('|H(e^{j\omega})|')
    hold on
    subplot(1,2,2)
    stem(angle(X),"LineWidth",1);
    xlabel('j\omega');
    ylabel('arg(H(e^{j\omega}))')
end
```

این بار صفرها را به ابتدای سیگنال اضافه کردیم و تعداد نمونه‌ها توان دو نمی‌شود!



همین طور که می‌بینیم شکل فاز تغییر پیدا کرده. تعداد نمونه‌های حوزه فوریه همان ۲۰ باقی‌مانده. پس باید در zero padding به دو نکته توجه کنیم:

1. تعداد صفرها را به گونه‌ای تنظیم کنیم که مجموع نمونه‌ها توانی از ۲ باشد.
2. صفرها را به انتهای نمونه اضافه کنیم.

سوال سوم

در این سوال نیز از تابع سوال‌های قبل استفاده شده.

```
%3rd computer asignment, DSP
%Anaies Golboudaghians 40122113
```

```
%Q3
clc; clear; close all; close all hidden

n =0:31;

x1 = cos((3*pi).*n./8);
x2 = cos((3*pi).*n./16);
x3 = cos((3*pi).*n./17);

fftANDplot(x1);
fftANDplot(x2);
fftANDplot(x3);

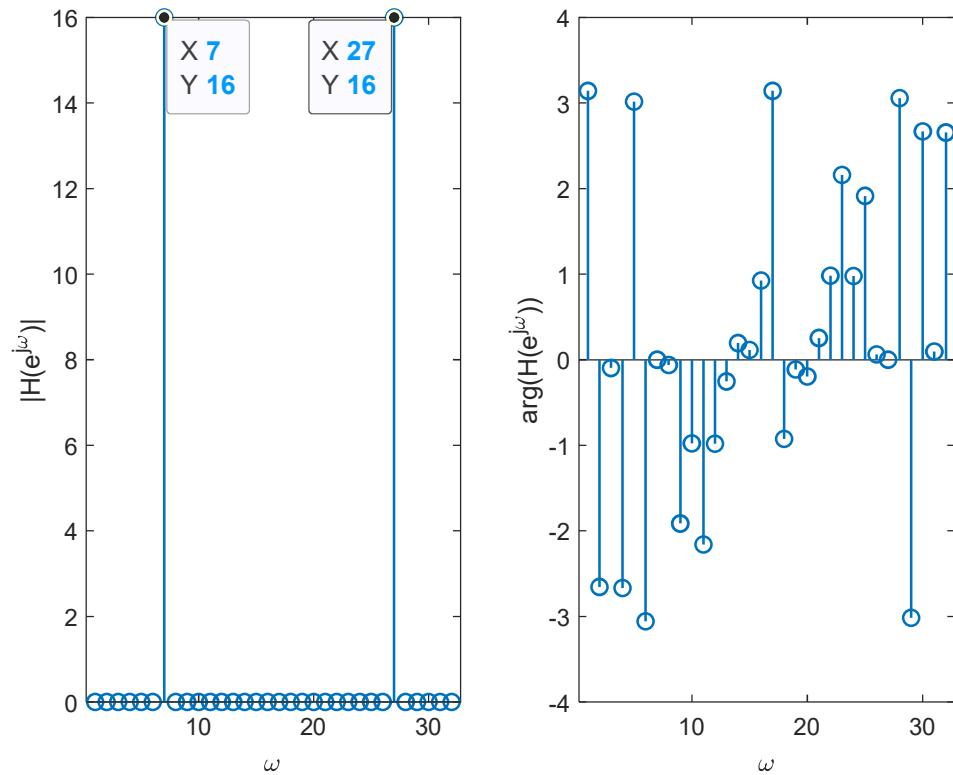
function fftANDplot(x)
    X = fft(x);
    figure
    % w_sample=1/length(x)*(0:length(x)-1);
    subplot(1,2,1)
    stem(abs(X), "LineWidth",1);
    xlabel('|\omega|');
    ylabel('|H(e^{j\omega})|')
    hold on
    subplot(1,2,2)
```

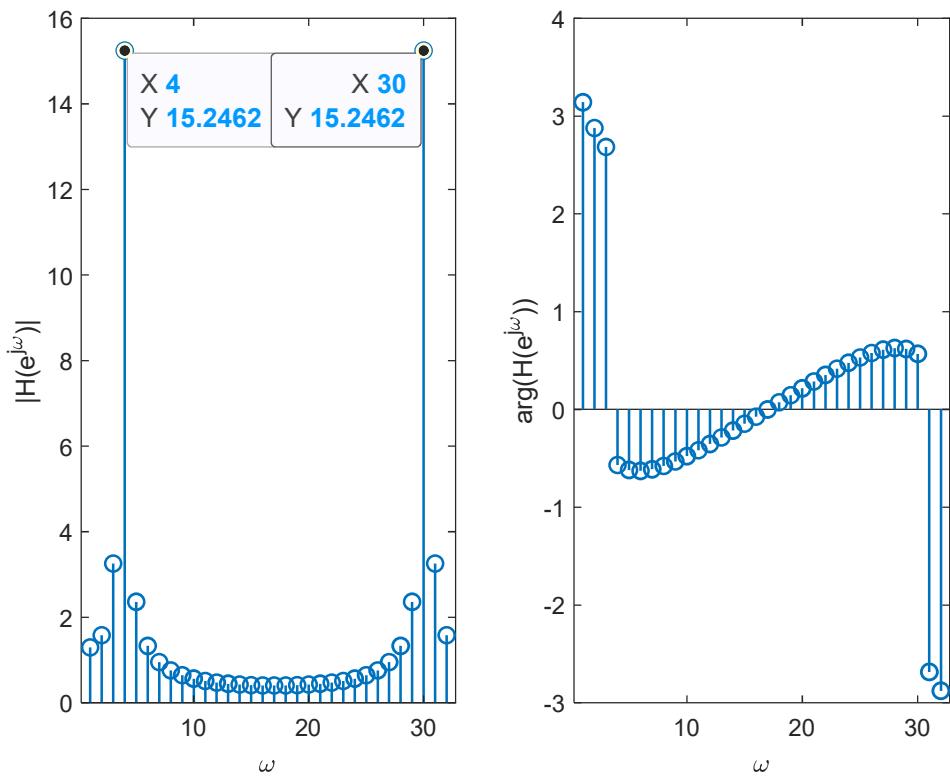
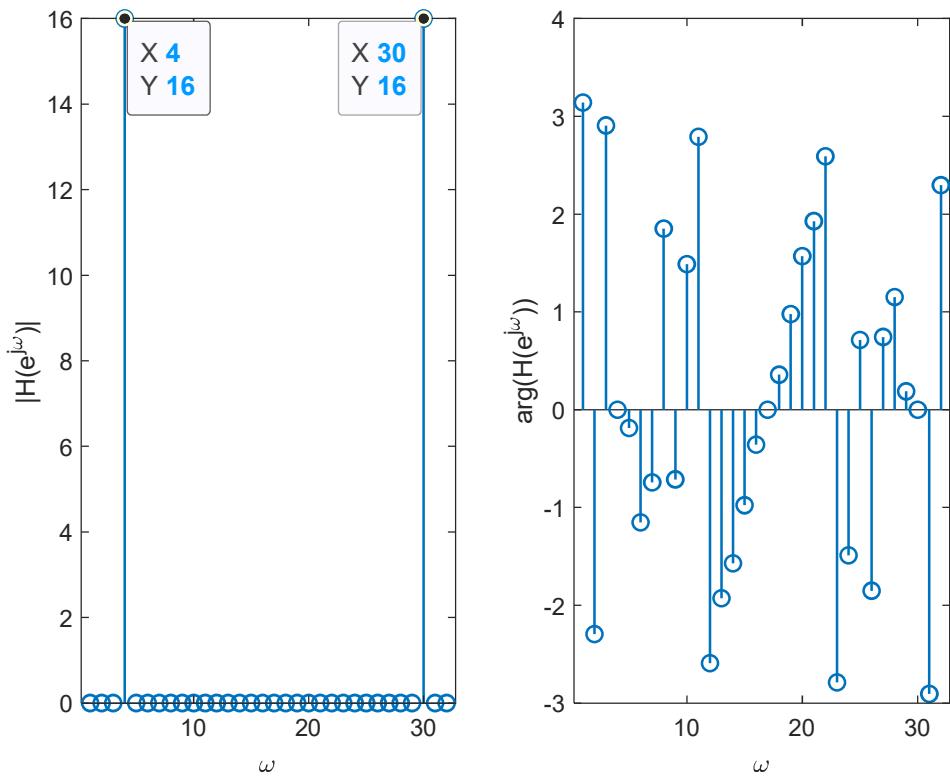
```

stem(angle(X),"LineWidth",1);
xlabel('\omega');
ylabel('arg(H(e^{j\omega}))')
end

```

خروجی





در فرکانس‌هایی که در شکل مشخص شده، اندازه به مقدار ماکسیمم می‌رسد.

سوال چهارم

در این سوال تغییر جزئی در تابع همیشگی داده شده، این تابع علاوه بر رسم، خروجی مقادیر فوریه محاسبه شده را نیز می‌دهد.

```
function X = fftANDplot(x)
    X = fft(x);
    figure
    % w_sample=1/length(x)*(0:length(x)-1);
    subplot(1,2,1)
    stem(abs(X), "LineWidth",1);
    xlabel('|\omega|');
    ylabel('|H(e^{j\omega})|')
    hold on
    subplot(1,2,2)
    stem(angle(X), "LineWidth",1);
    xlabel('|\omega|');
    ylabel('arg(H(e^{j\omega}))')
end
```

بخش اول

```
%3rd computer asignment, DSP
%Anaies Golboudaghians 40122113
```

```
%Q4
clc; clear; close all; close all hidden

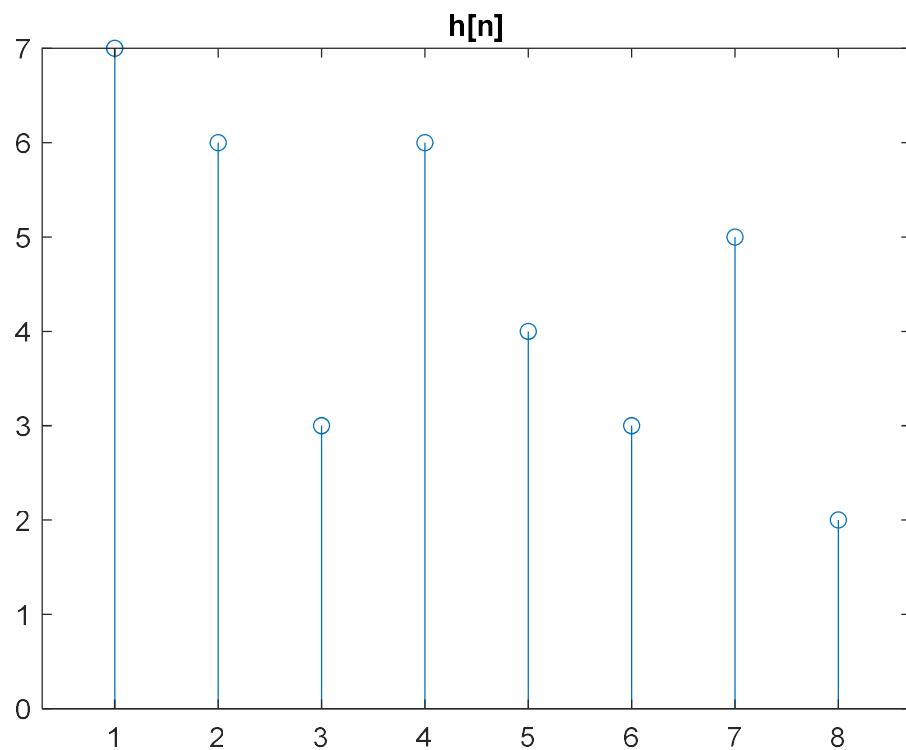
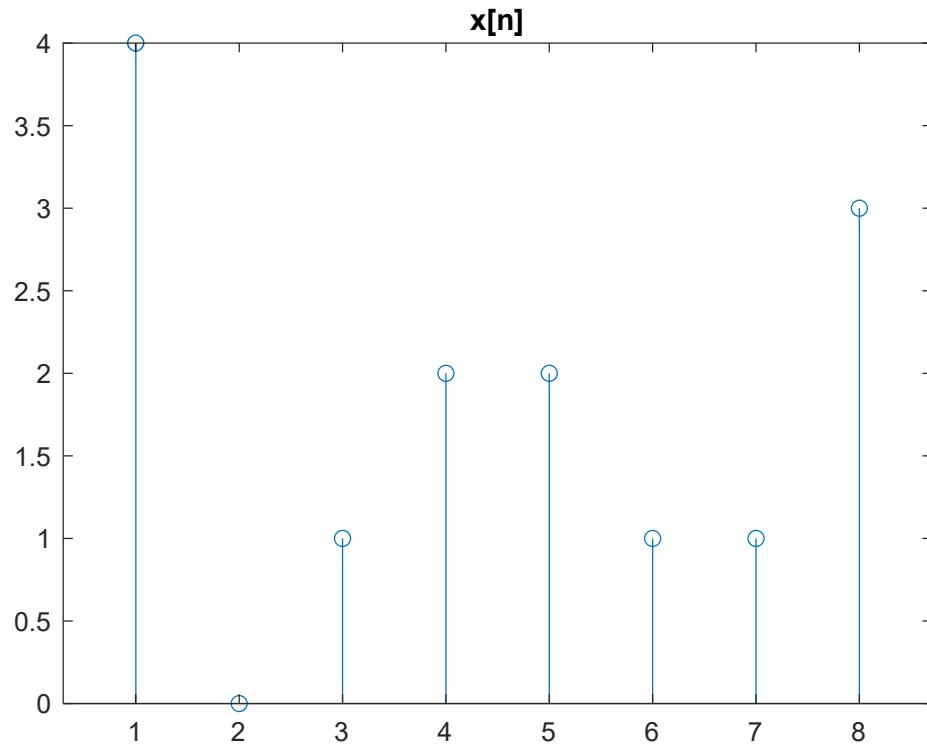
%% Part 1
seed = 40122113; % Set the seed value for the random number generator
rng(seed);
% Generate random numbers between 0 and 9
h = randi ([0, 9], 8, 1);
x = [4; 0; 1; 2; 2; 1; 1; 3];

stem(x);
title('x[n]');
figure
stem(h);
title('h[n]')
```

از بخش راهنمایی، کد گفته شده را وارد می‌کنیم. تعیین `seed` مثل اینکه برای منحصر به فرد بودن خروجی تولید مقادیر تصادفی است.

از جایی که `h` تولید شده هشت سطر و یک ستون دارد، با قرار دادن نقطه ویرگول به جای ویرگول، آرایه‌ی `x` را نیز این‌چنان درست می‌کنیم.

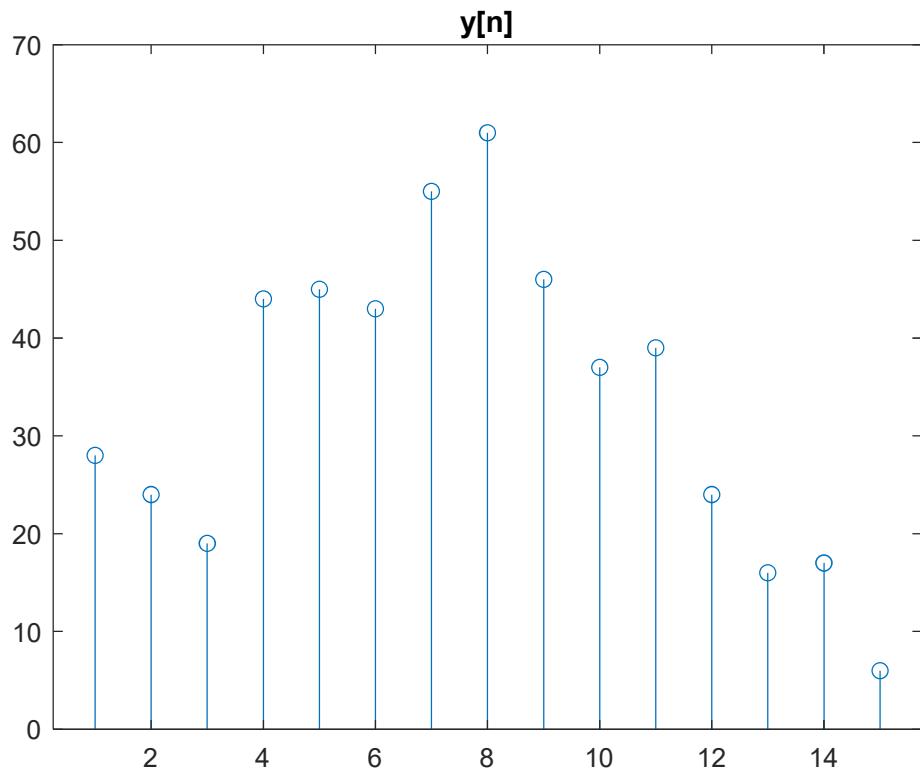
خروجی:



بخش دوم

```
%% Part 2  
y = conv(x,h);  
figure  
stem(y);  
title('y[n]');
```

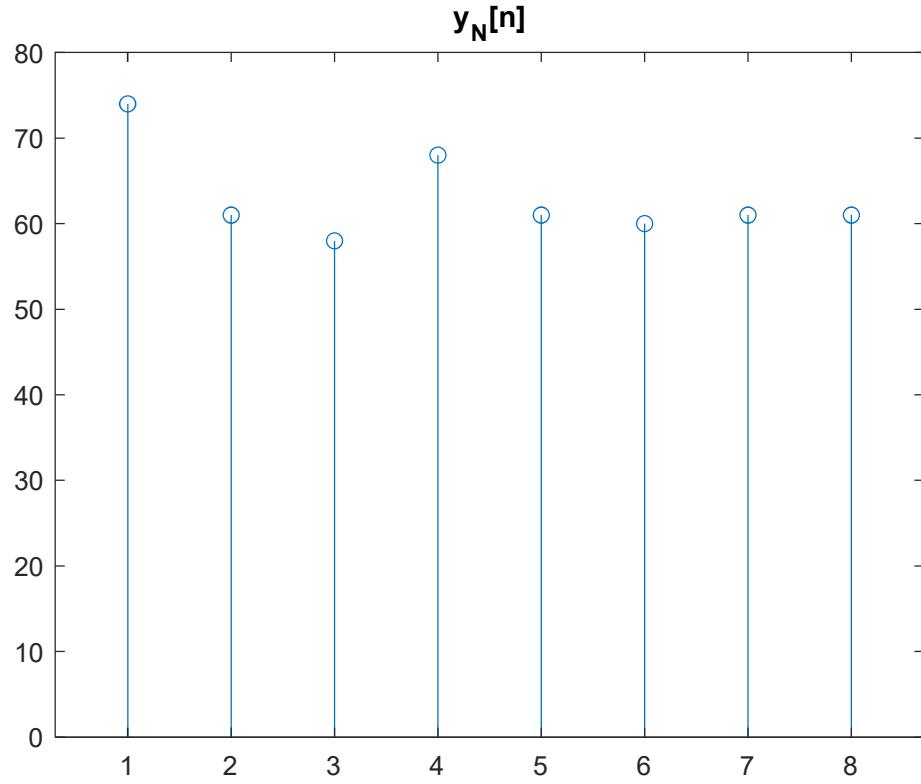
از تابع `conv` برای محاسبه کانولوشن خطی استفاده می‌کنیم و آن را رسم می‌کنیم.



بخش سوم

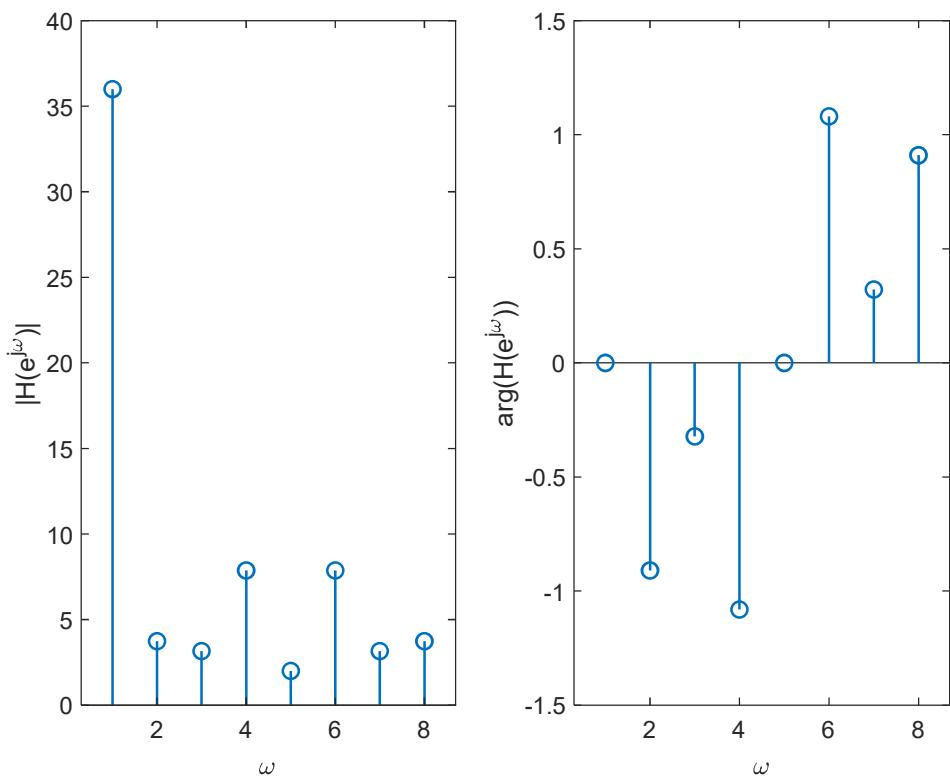
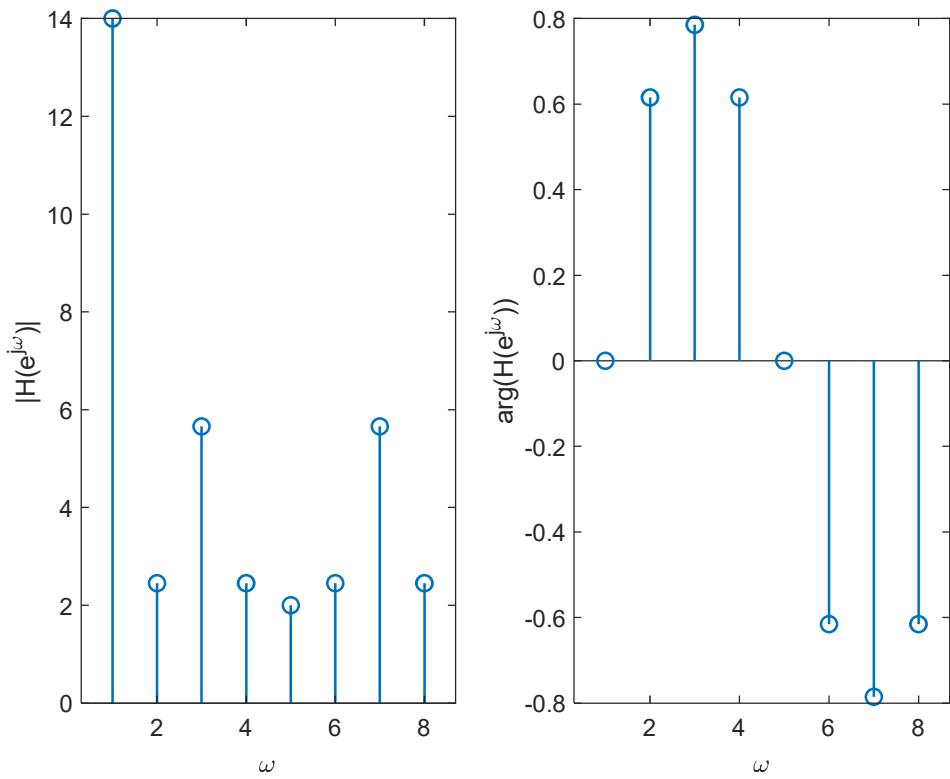
```
%% Part 3  
y_c = cconv(x,h,8);  
figure  
stem(y_c);  
title('y_N[n]');
```

از `cconv` برای کانولوشن دایروی ۸ نقطه‌ای استفاده می‌کنیم.



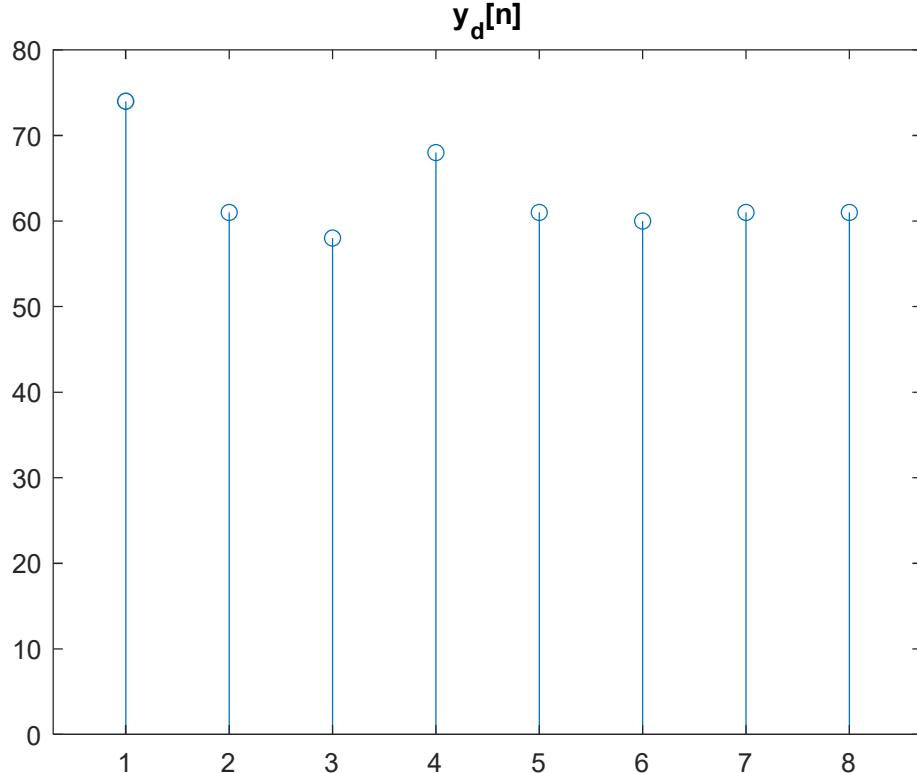
بخش چهارم

```
%% Part 4  
X = fftANDplot(x);  
H = fftANDplot(h);
```



بخش پنجم

```
%% Part 5
Y = X.*H;
y_d = ifft(Y);
figure
stem(y_d);
title("y_d[n]")
```



شکل به دست آمده مانند شکل بخش سوم شد.

سوال پنجم

بخش صفرم

از همان تابع سوال قبل استفاده می‌کنیم. (تابع‌ها در انتهای کد می‌آیند)

```
function X = fftANDplot(x)
X = fft(x);
figure
% w_sample=1/length(x)*(0:length(x)-1);
subplot(1,2,1)
stem(abs(X), "LineWidth",1);
xlabel('|\omega|');
ylabel('|H(e^{j\omega})|')
```

```

hold on
subplot(1,2,2)
stem(angle(X),"LineWidth",1);
xlabel('omega');
ylabel('arg(H(e^{j\omega}))')
end

```

در ابتدا و قبل از بخش اول، دستورات اولیه را می‌نویسیم و سیگنال را شکل می‌دهیم.

```

%3rd computer assignment, DSP
%Anaies Golboudaghians 40122113

```

```

%Q7
clc; clear; close all; close all hidden

```

```

n = 0:255;
s = cos((pi*5).*n./32) + cos((pi*21).*n./64);

```

```

e = sqrt(0.3)*randn(1,256);
x = s+e;

```

انحراف معیار را در نمونه‌های تولید شده ضرب می‌کنیم تا واریانس مورد نظر را پیدا کند.

بخش اول

```

%% Part 1

```

```

E = sum((x-s).^2);
disp(E);

```

انرژی سیگنال خطأ را با فرمول گفته شده محاسبه می‌کنیم.

خروجی:

83.6575

بخش دوم

```

%% Part 2

```

```

S = fftANDplot(s);
X = fftANDplot(x);

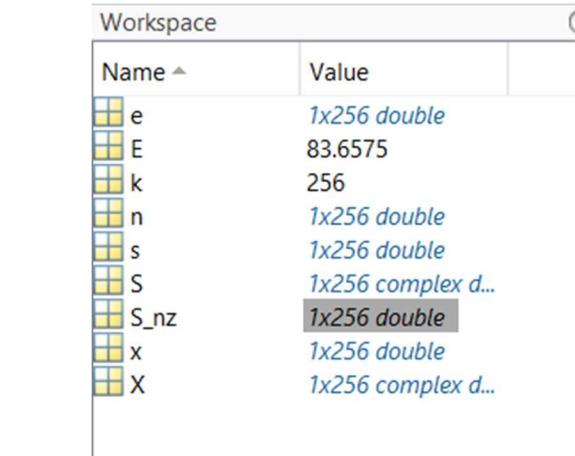
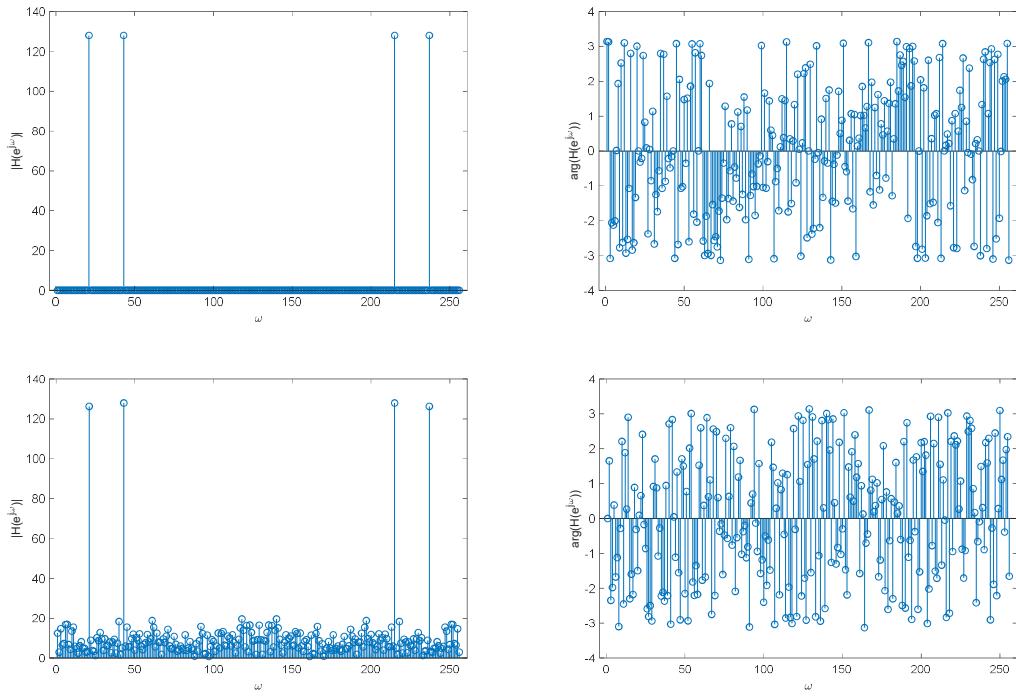
```

```

for k=1:length(S)
    if S(k)~=0
        S_nz(k)=k;
    end
end

```

همهی مقادیر k در شرایط گفته شده صدق می‌کرد.



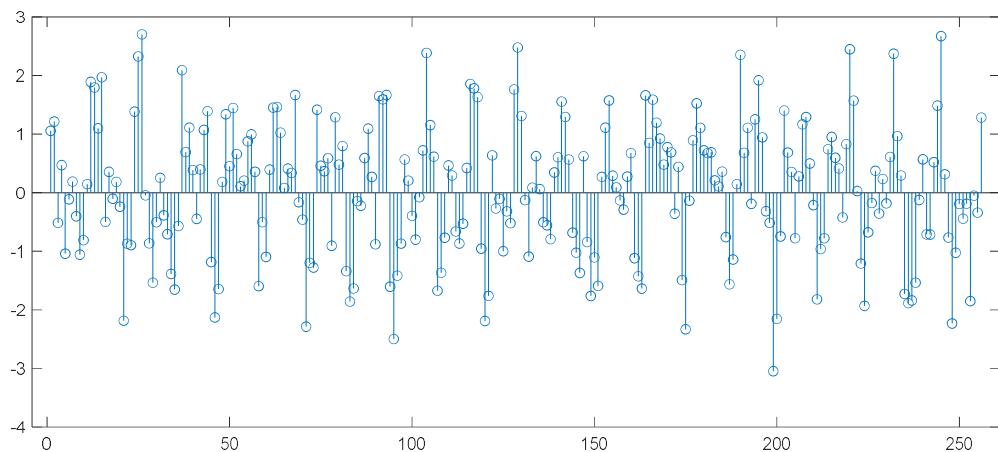
بخش سوم

```
%% Part 3
for k=1:length(X)
    if ismember(k,S_nz)
        X_h(k)= X(k);
    else
        X_h(k)= 0;
    end
end

x_h = ifft(X_h);
figure
stem(x_h);
```

```
E_h = sum((x_h-s).^2);
disp(E_h);
```

چون همه مقادیر k را داشتیم، سیگنال متناوب جدید برابر با سیگنال قبلی است.



83.6575

انرژی سیگنال خطای هم به طبع همان مقدار درمی‌آید.

سوال ششم

```
%3rd computer assignment, DSP
%Anaies Golboudaghians 40122113
```

```
%Q6
clc; clear; close all; close all hidden
```

بخش اول

```
%% Part 1
pic = imread('image1.jpg');
pic = rgb2gray(pic);
imshow(pic);
title('main image')
```

```
E_Signal = mean(pic(:));
noise = pic-E_Signal;
E_Noise = mean(noise(:));
```

```
SNR = 10*log10((E_Signal^2)/(E_Noise^2));
fprintf('SNR of main image is %.3f dB\n', SNR);
```

برای پردازش بهتر باید تصویر را سیاه سفید کنیم. به کمک دستور `rgb2gray` انجام شد.

برای محاسبه SNR از فرمول زیر استفاده شد.

$$\text{SNR} = \frac{\mathbb{E}[S^2]}{\mathbb{E}[N^2]},$$

برای بهدست آوردن در مقیاس دسیبل:

$$\text{SNR}_{\text{dB}} = 10 \log_{10} \left[\left(\frac{A_{\text{signal}}}{A_{\text{noise}}} \right)^2 \right] = 20 \log_{10} \left(\frac{A_{\text{signal}}}{A_{\text{noise}}} \right) = A_{\text{signal,dB}} - A_{\text{noise,dB}}.$$

یعنی باید از سیگنال اصلی و نویز میانگین بگیریم. با دستور `mean` در متلب می توانیم میانگین بگیریم. برای محاسبه نویز، داده اولیه عکس را از میانگین اصلی کم می کنیم. از داده های بهدست آمده میانگین می گیریم و میانگین نویز بهدست می آید.

خروجی ها:

main image



SNR of main image is 13.483 dB

بخش دوم

%% Part 2

```

PIC = fft2(pic);
pic_r = ifft2(PIC);
figure
imshow(pic_r,[]);
title('reconstructed image')

```

برای تبدیل فوریه دو بعدی، از `fft2` و `ifft2` برای معکوس آن استفاده شده.

خروجی:

reconstructed image



بخش سوم

```

%% Part 3
E_Signal_r = mean(pic_r(:));
noise_r = pic_r-E_Signal_r;
E_Noise_r = mean(noise_r(:));

SNR_r = 10*log10((E_Signal_r^2)/(E_Noise_r^2));
fprintf('SNR of reconstructed image is %.3f dB\n', SNR_r);

```

SNR of reconstructed image is 295.955 dB

بخش چهارم

خیر، اتفاقاً خیلی بیشتر شده. این نشان گر کمتر شدن تاثیر نویز پس از بازسازی مجدد تصویر است.

بخش پنجم

```
%% Part 5
figure
imshow(abs(pic_r),[]);
title('magnitude of reconstructed image')
figure
imshow(angle(pic_r),[]);
title('phase of reconstructed image')
```

از `abs` برای اندازه و `angle` برای فاز استفاده می‌کنیم. چون مقادیر بازسازی شده حقیقی‌اند، انتظار خاصی از مقادیر فاز نباید داشته باشیم.

magnitude of reconstructed image

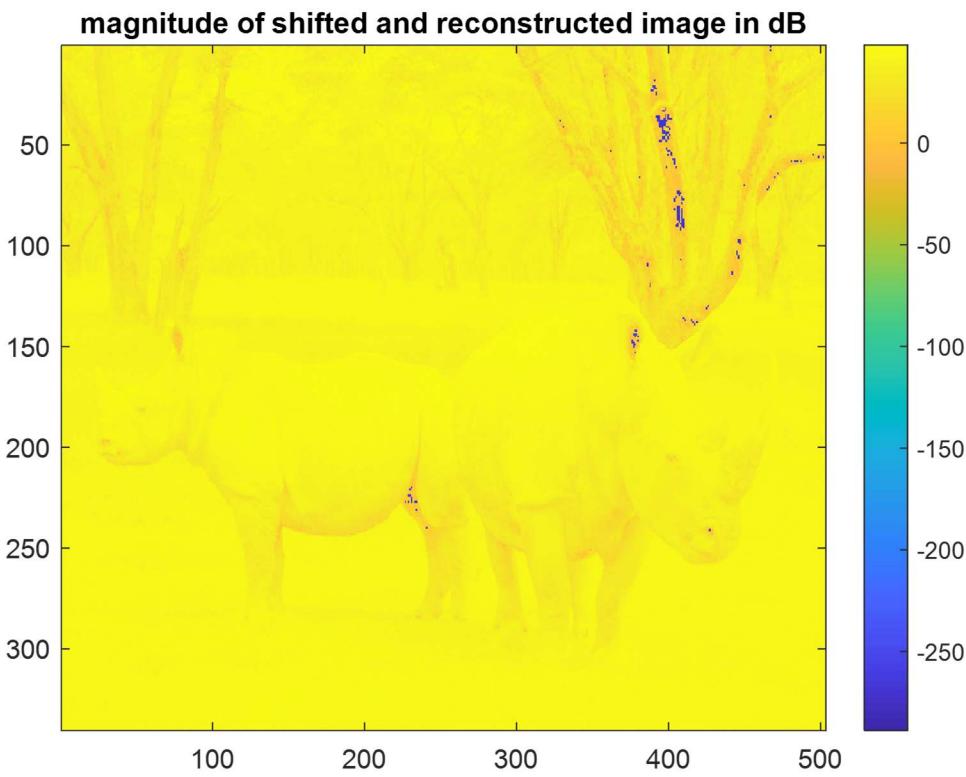


phase of reconstructed image



بخش ششم

```
%% Part 6
PIC_sh = fftshift(PIC);
pic_r_sh = ifft2(PIC_sh);
phase = angle(pic_r_sh);
figure
imagesc(20*log10(abs(pic_r_sh)));
title('magnitude of shifted and reconstructed image in dB')
colorbar;
```



بخش هفتم

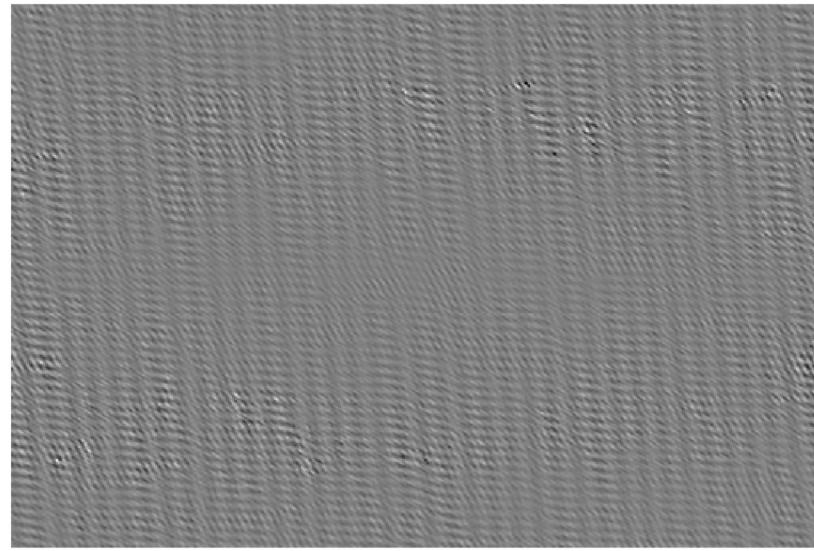
در بخش قبلی فازی را به دست آوردیم که در این بخش از آن استفاده می‌کنیم برای ساخت سیگنال جدید.

```
%% Part 7, A
PIC_7 = fft2(pic);
New_PIC_7 = abs(PIC_7).*exp(1j*phase);
pic_7_r = ifft2(New_PIC_7);
figure
imshow(abs(pic_7_r),[])
title('magnitude of reconstructed image with the phase of part 6')
figure
imshow(angle(pic_7_r),[])
title('phase of reconstructed image with the phase of part 6')
```

magnitude of reconstructed image with the phase of part 6



phase of reconstructed image with the phase of part 6



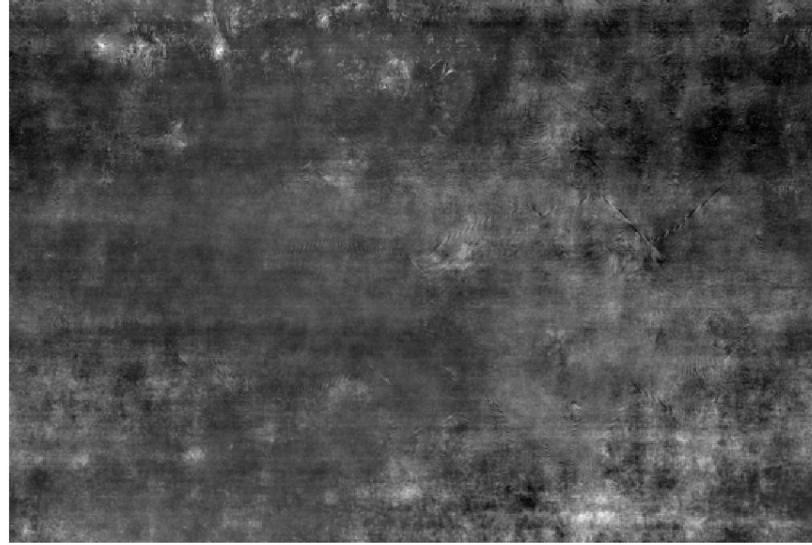
احتمالاً چون از فازی که در حوزه زمان داشتیم استفاده کردیم، تصویر جدیدی که ساختیم نتیجه‌ی جالبی نداده است.

حال از فاز بخش شش در حوزه فوریه استفاده می‌کنیم:

%% Part 7, B

```
PIC_7 = fft2(pic);
New_PIC_7 = abs(PIC_7).*exp(1j*angle(PIC_sh));
pic_7_r = ifft2(New_PIC_7);
figure
imshow(abs(pic_7_r),[]);
title('magnitude of reconstructed image with the phase of part 6')
figure
imshow(angle(pic_7_r),[])
title('phase of reconstructed image with the phase of part 6')
```

magnitude of reconstructed image with the phase of part 6



phase of reconstructed image with the phase of part 6

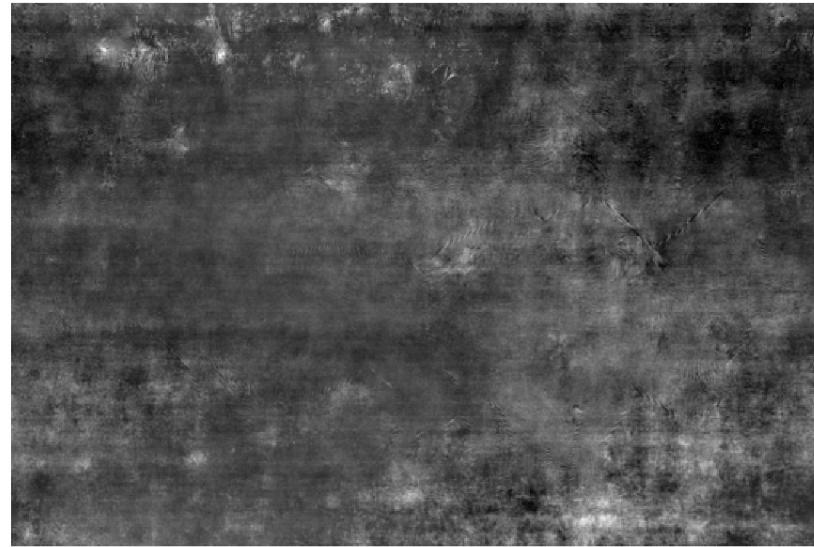


این شیفت داده شده در بخش شش، باید روی اندازه هم اعمال می شد تا نتیجه مطلوب می شد.

بخش هشتم

```
%% Part 8
New_PIC_8 = abs(PIC_sh).*exp(1j*angle(PIC));
pic_8_r = ifft2(New_PIC_8);
figure
imshow(abs(pic_8_r),[]);
title('magnitude of shifted and reconstructed image with the phase of input image')
figure
imshow(angle(pic_8_r),[])
title('phase of shifted and reconstructed image with the phase of input image')
```

magnitude of shifted and reconstructed image with the phase of input image



phase of shifted and reconstructed image with the phase of input image

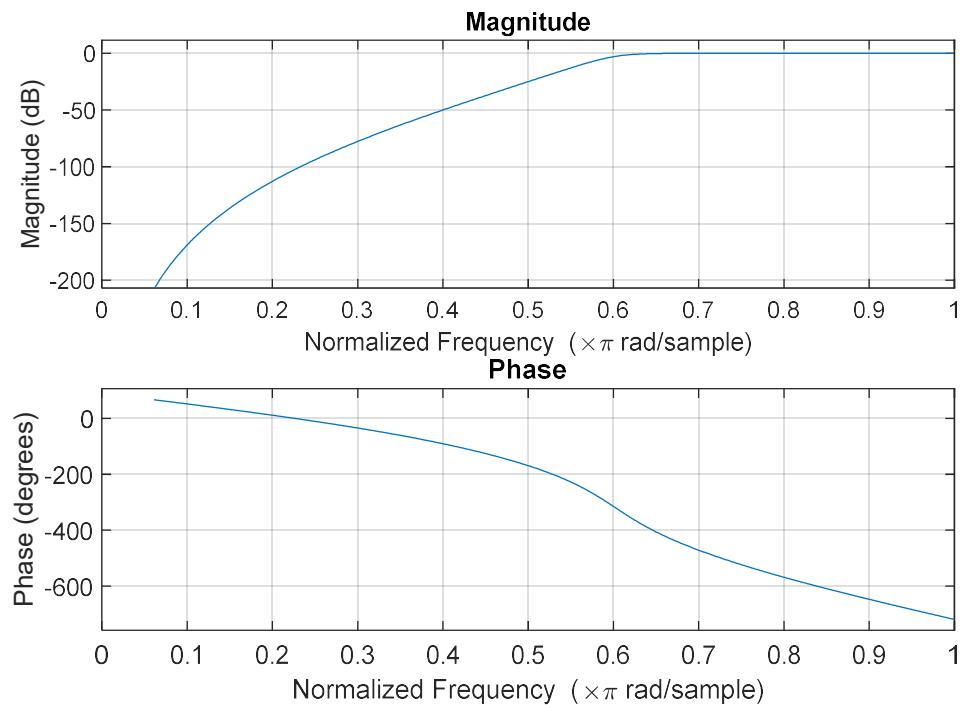


بخش نهم

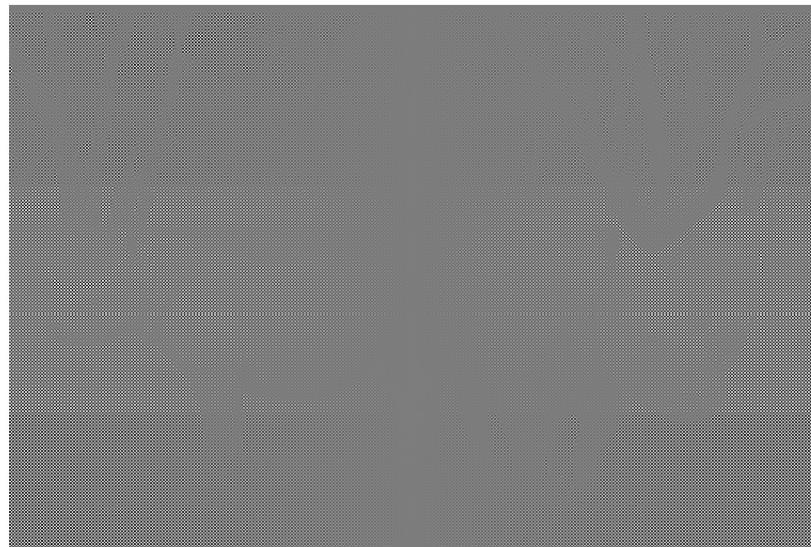
دیدیم که در بخش‌های قبل، با تغییر فاز و استفاده از فاز دیگر، تصویر بهم ریخت و از بین رفت. پس فاز نقش مهمی در پردازش تصویر دارد.

```
%% Part 10
[b,a] = butter(9,300/500,"high");
freqz(b,a,1000)
yy = filter(b,a,pic_r_sh);
figure
imshow(yy,[]);
title('filtered image');
```

یک فیلتر با ترورث دلخواه از نوع بالاگذر انتخاب کردیم.



filtered image



به نظر می‌رسد اطلاعاتی از تصویر که در فرکانس‌های میانی وجود داشت از بین رفت.

بخش یازدهم

اگر در بخش دهم به جای فیلتر بالا گذر، از پایین گذر استفاده می‌کردیم کل تصویر از بین می‌رفت. پهنه‌ای باند گذر را نیز باید بیشتر در نظر می‌گرفتیم زیرا مثل این که در فرکانس‌های نیم تا 0.6. اطلاعاتی وجود داشت.

پس به طور کلی در تصاویر فرکانس‌های بالا اطلاعات بیشتری از تصویر را در خود دارند.

سوال هفتم

```
%3rd computer assignment, DSP  
%Anaies Golboudaghians 40122113
```

```
%Q7  
clc; clear; close all; close all hidden
```

بخش اول

```
%% Part 1  
pic = imread('image2.png');  
PIC = fft2(pic);  
imshow(pic,[])  
figure  
imshow(abs(PIC),[]);
```

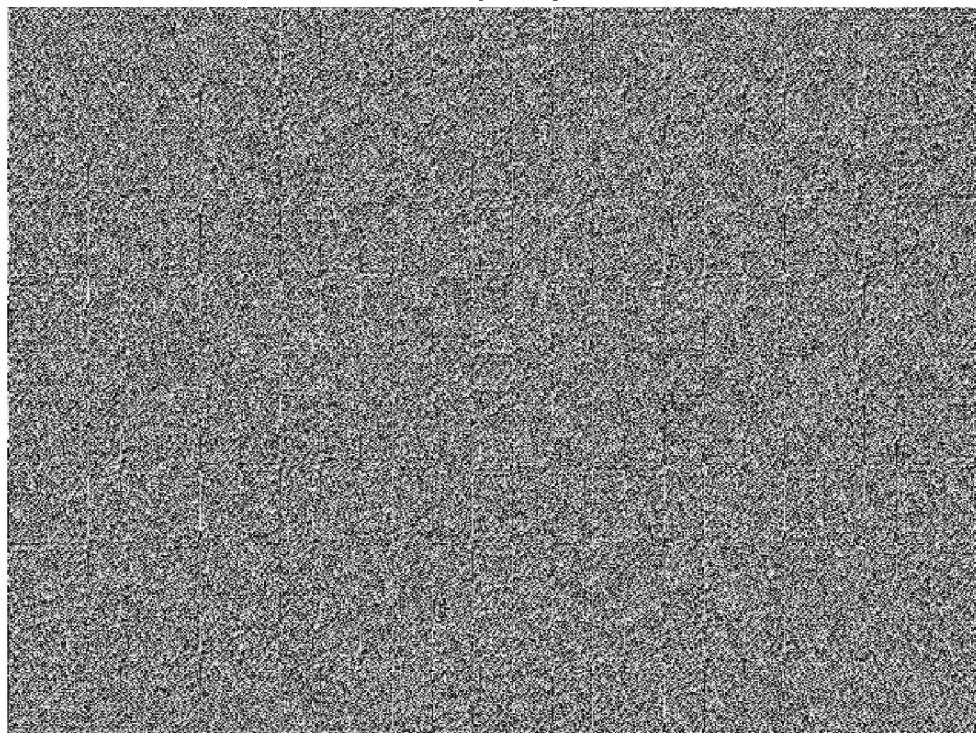
```
title('Magnitude in frequency domain')
figure
imshow(angle(PIC),[]);
title('Phase in frequency domain')
```



Magnitude in frequency domain

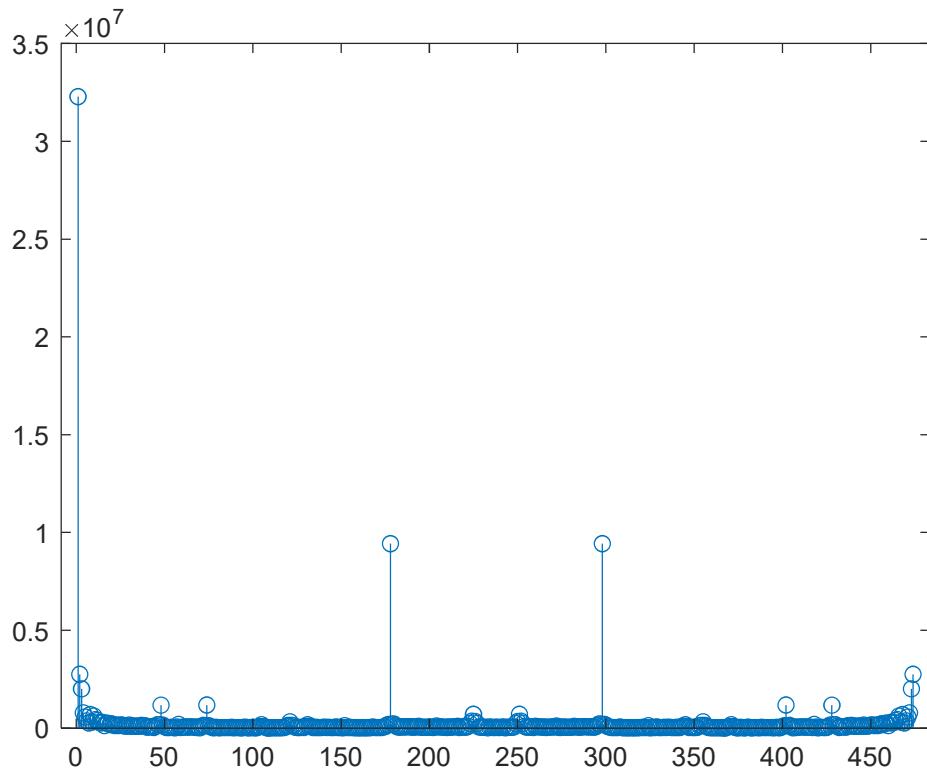


Phase in frequency domain



```
figure  
stem(abs(PIC(:,1)));
```

برای ستون اول داده تصویر، اندازه آن را در حوزه فوریه رسم می کنیم.



به نظر می‌رسد نویز در اواسط آرایه است.

بخش دوم

```
%% Part 2
fs = 1000;
N = size(PIC);

f1 = (0:N(2)-1)*(fs/N(2));
cutoff = [100 900];
H0 = double((f1 <= cutoff(1))|(f1 >= cutoff(2)));
Y0 = PIC.*H0;

f2 = (0:N(1)-1)*(fs/N(1));
H = double((f2 <= cutoff(1))|(f2 >= cutoff(2)));
Y = Y0.*H';
```

یک فیلتر ایده‌آل متشكل از صفر و یک به صورت آرایه درست می‌کنیم. باید توجه داشت که هم عمودی و هم افقی، داده‌ها را فیلتر کنیم. ده درصد اول و آخر را نگه می‌داریم و بقیه را فیلتر می‌کنیم.

بخش سوم

```
%% Part 3
y = ifft2(Y);
figure
```

```
imshow(y,[])
title('after filtering')
y_8b = uint8(y);
imwrite(y_8b,"filteredimage2.png");
```

از نتیجه تبدیل فوریه معکوس می‌گیریم و نمایشش می‌دهیم. باید توجه داشته باشیم که موقع ذخیره‌سازی آن را به صورت هشت‌بیتی ذخیره کنیم، در غیر این صورت چیز درستی ذخیره نمی‌شود.

خروجی:

after filtering

