

دانیال امامیان

پروژه درس احتمال مهندسی دکتر نریمانی

پروژه اول:

قسمت الف:

پارامتر a در این سوال متغیر فرض شده که برای بنده مساوی 3 خواهد بود ، سپس خواسته های سوال را به ترتیب انجام میدهیم:

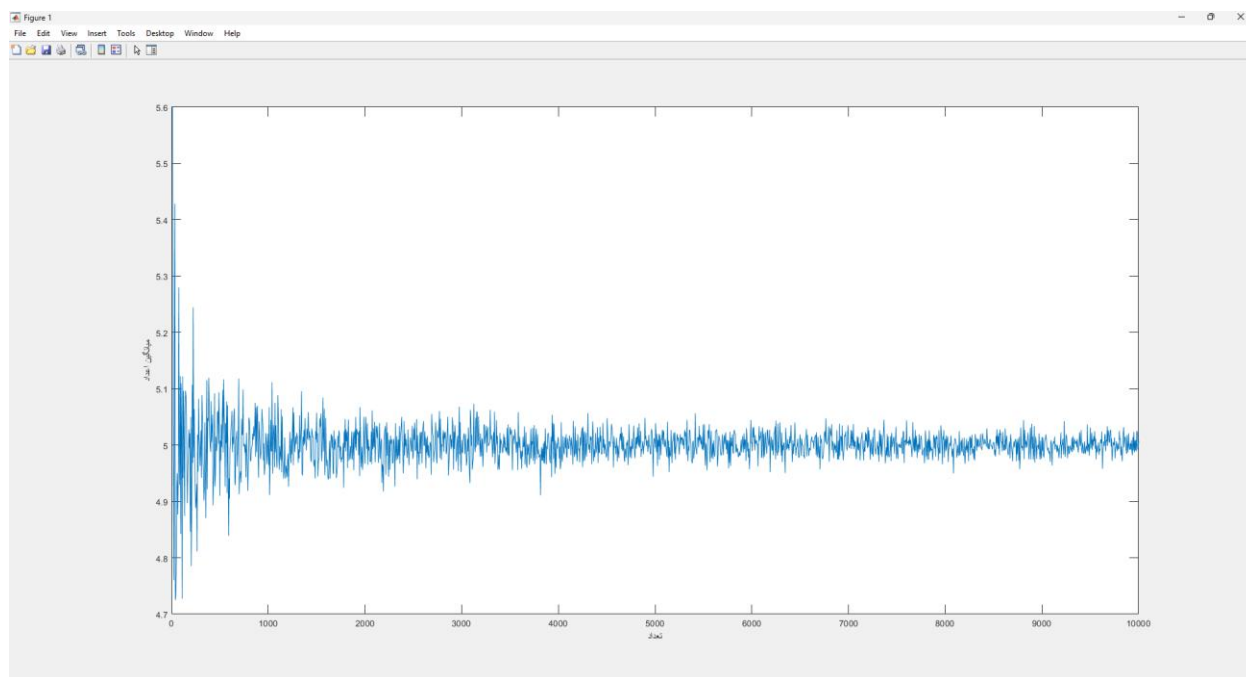
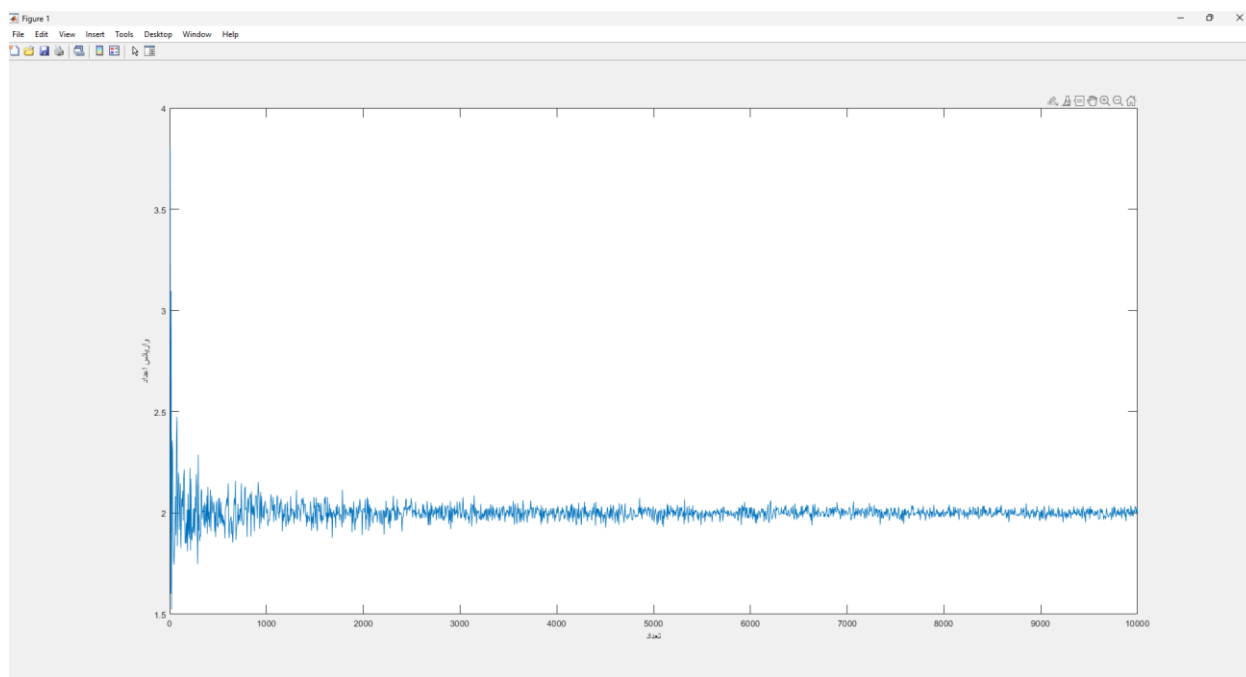
```
a = 3;
b = (2*a)+1;
start_from = 0;
end_to = 10000;
step = 5;
requested_var = [];
requested_mean = [];
for count = start_from:step:end_to
    random_num = randi([a,b], 1, count);
    var_in_each_loop = var(random_num);
    mean_in_each_loop = mean(random_num);
    requested_mean = [requested_mean mean_in_each_loop];
    requested_var = [requested_var var_in_each_loop];
end
figure;
```

حال با دستورات زیر نمودار هر یک از واریانس و میانگین را رسم میکنیم:

```
plot(start_from:step:end_to , requested_var);
xlabel('تعداد');
ylabel('واریانس اعداد');
```

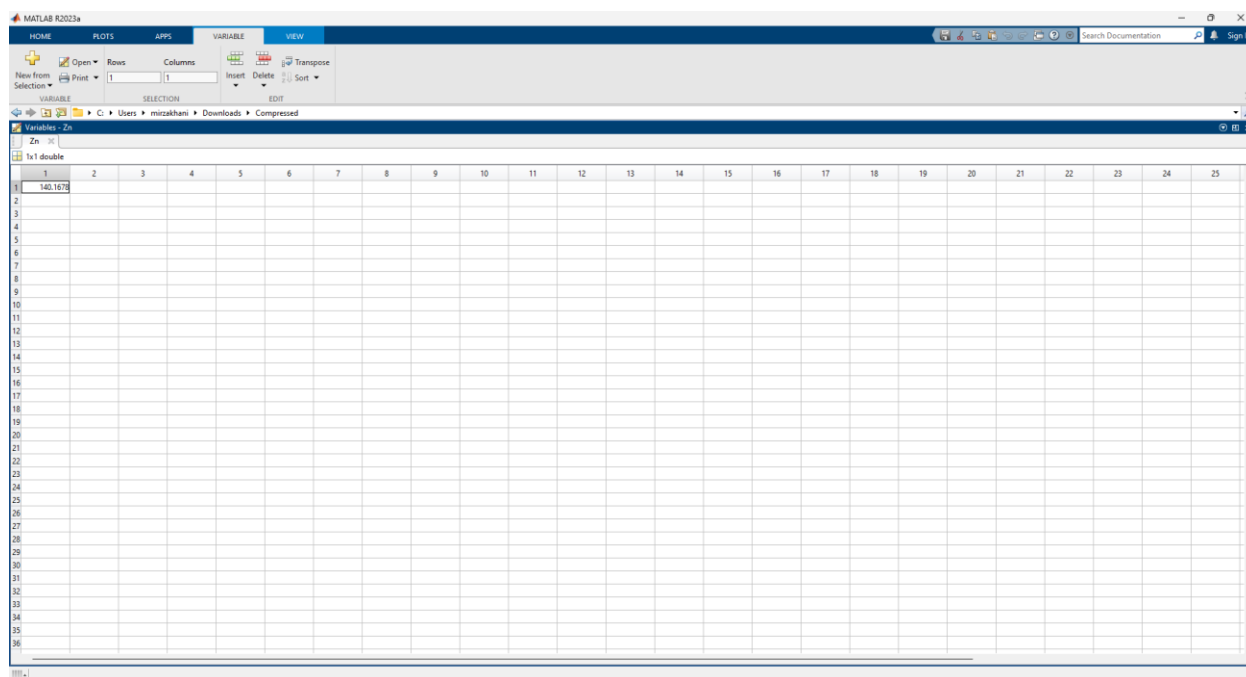
```
plot(start_from:step:end_to , requested_mean);
xlabel('تعداد');
ylabel('میانگین اعداد');
```

نمودار ها به صورت زیر خواهد شد:



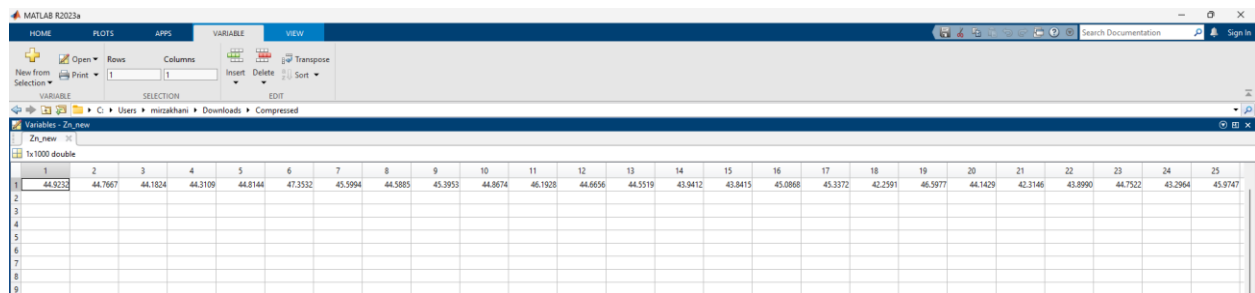
قسمت ب:

در ابتدا، در حلقه قسمت قبل، یک متغیر به نام Zn تعریف میکنیم و مقادیر آن را را طبق گفته سوال بدست می آوریم:



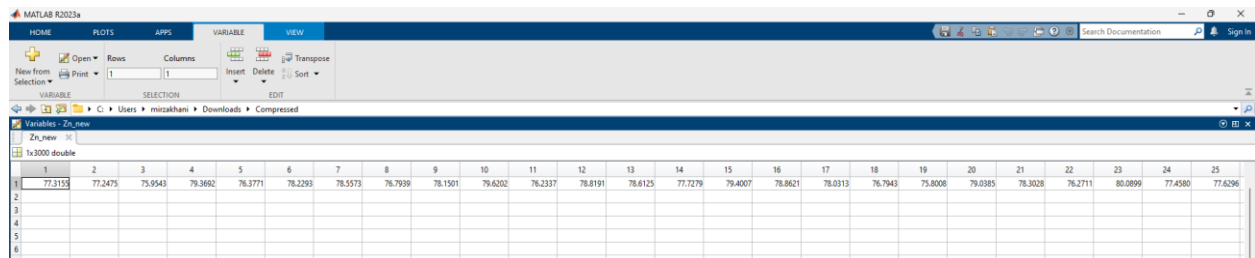
سپس می‌آیم و یک متغیر n با تعداد 1000 انتخاب می‌کنیم و zn را در هر مرحله در یک حلقه حساب می‌کنیم:

```
% 1.2.2:  
a = 3;  
b = (2*a)+1;  
n = 1000;  
Zn_new = zeros(1,n);  
for x = 1:n  
    random_num = randi([a,b], 1, n);  
    mean_in_each_loop = mean(random_num);  
    var_in_each_loop = var(random_num);  
    Zn_new(x) = ((mean_in_each_loop - a) / sqrt(var_in_each_loop)) * sqrt(n);  
end
```



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|---|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| 1 | 44.9232 | 44.7667 | 44.1824 | 44.3109 | 44.8144 | 47.3532 | 45.5994 | 44.5885 | 45.3953 | 44.8674 | 46.1928 | 44.6656 | 44.5519 | 43.9412 | 43.8415 | 45.0868 | 45.3372 | 42.2591 | 46.5977 | 44.1429 | 42.3146 | 43.8990 | 44.7522 | 43.2964 | 45.9747 |
| 2 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | | | | | | | | |

برای افزایش دقت می‌توانیم n را افزایش دهیم ، برای مثال n را مساوی 3000 قرار می‌دهیم:



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|---|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| 1 | 77.3155 | 77.2475 | 75.9543 | 79.3692 | 76.3771 | 78.2293 | 78.5573 | 76.7939 | 78.1501 | 79.6202 | 76.2337 | 78.8191 | 78.6125 | 77.7279 | 79.4007 | 78.8621 | 78.0313 | 76.7943 | 75.8008 | 79.0385 | 78.3028 | 76.2711 | 80.0899 | 77.4580 | 77.6296 |
| 2 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | | | | | | | | |

با رسم تابع هیستگرام مشاهده می‌شود که Zn ما ، بسیار شبیه به توزیع نرمال 0 و 1 می‌شود که این نتیجه باعث می‌شود قضیه حد مرکزی را صدق دهد.

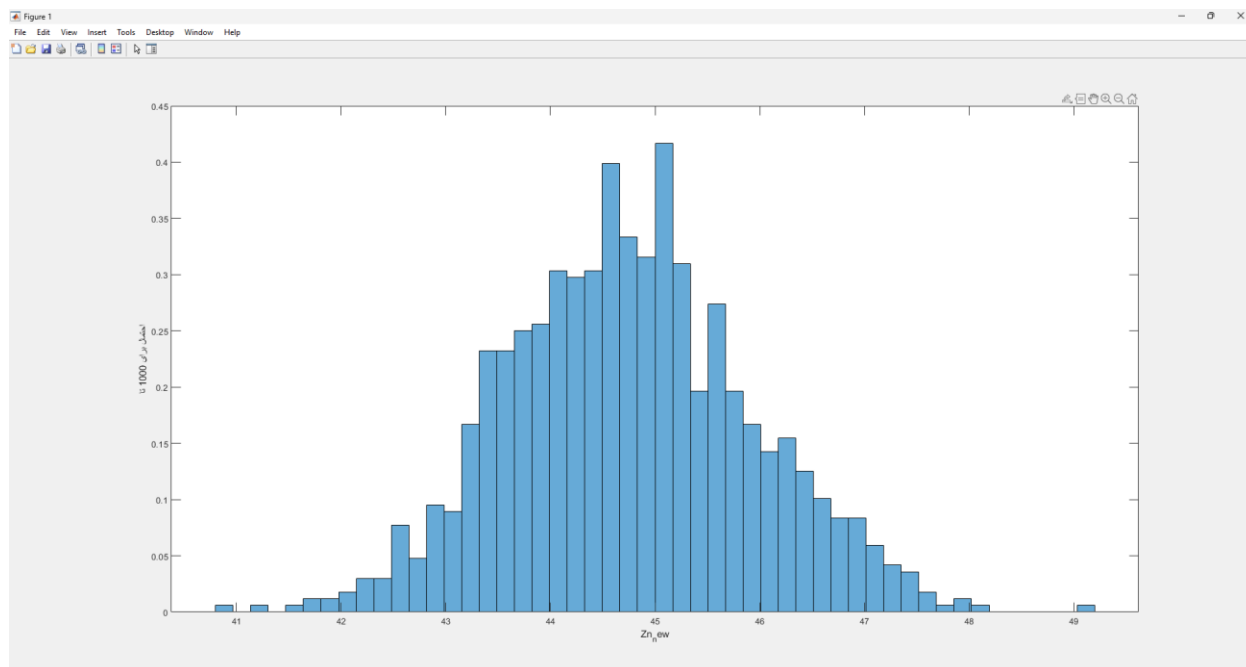
تابع هیستوگرام را دستور زیر اجرا میکنیم:

```
histogram(Zn_new, 50, 'Normalization', 'pdf');
```

در اینجا عدد مساوی یعنی تعداد ستون های رسم شده است و نرمالیزیشن پی دی اف باعث می شود که ارتفاع میله های هیستوگرام به جای تعداد داده ها، به شکلی نرمال سازی شود که مساحت کل زیر نمودار برابر با 1 باشد.

در نتیجه ما تابع چگالی احتمال را خواهیم داشت:

```
% 1.2.2:  
a = 3;  
b = (2*a)+1;  
n = 1000;  
Zn_new = zeros(1,n);  
for x = 1:n  
    random_num = randi([a,b], 1, n);  
    mean_in_each_loop = mean(random_num);  
    var_in_each_loop = var(random_num);  
    Zn_new(x) = ((mean_in_each_loop - a) / sqrt(var_in_each_loop)) * sqrt(n);  
end  
figure;  
histogram(Zn_new, 50, 'Normalization', 'pdf');  
xlabel('Zn_new');  
ylabel('احتمال برای 1000 تا');
```



قسمت ج:

حال با توزیع نمایی و متغیر دهگان می‌خواهیم تکرار کنیم.

```
% 1.3:
a = 1+1;
n = 1000;
count = 1000;
param = 1 / (mod(a, 10) + 1);
exp = exprnd(1/param, n, count);

Sum_exp = sum(exp);
var_exp = var(exp(:));
mean_exp = mean(exp(:));

Zn_exp = (Sum_exp - n * mean_exp) ./ sqrt(n * var_exp);

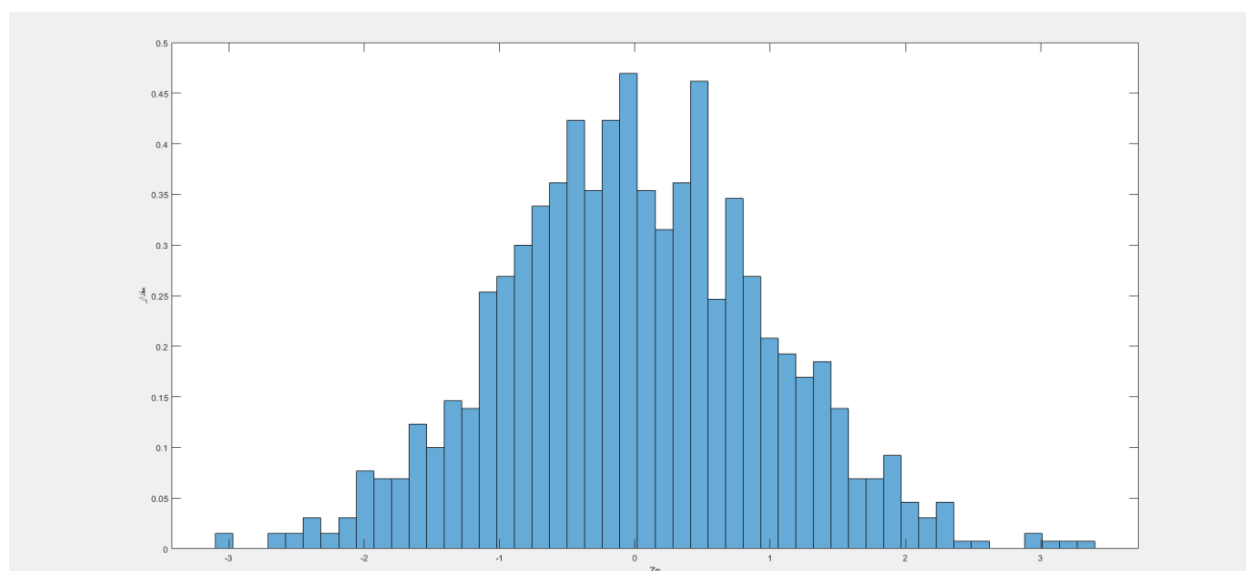
figure;
histogram(Zn_exp, 50, 'Normalization', 'pdf');
xlabel('Zn');
ylabel('مقدار');
```

در اینجا نیز ابتدا پارامتر تابع نمایی را تعریف می‌کنیم و سپس خود تابع را.

سپس مجموع اعداد در هر ستون را محاسبه می‌کنیم

$\exp(:)$ ماتریس \exp را به یک بردار تبدیل می‌کند و سپس با

$\text{var}(\exp(:))$ واریانس تمامی مقادیر موجود در \exp را محاسبه می‌کنیم.

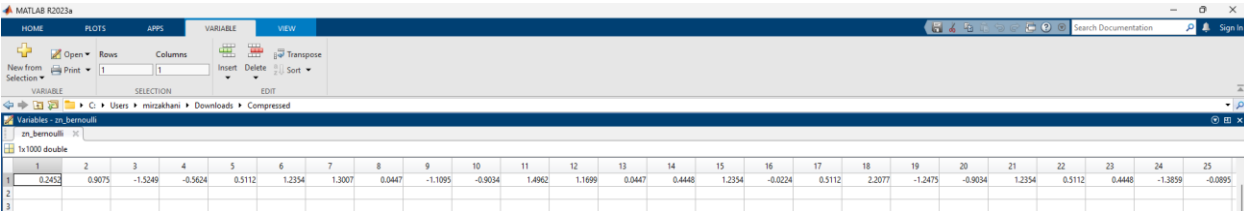


با داشتن تابع Zn ، مقدار میانگین و واریانس را به دست می آوریم:

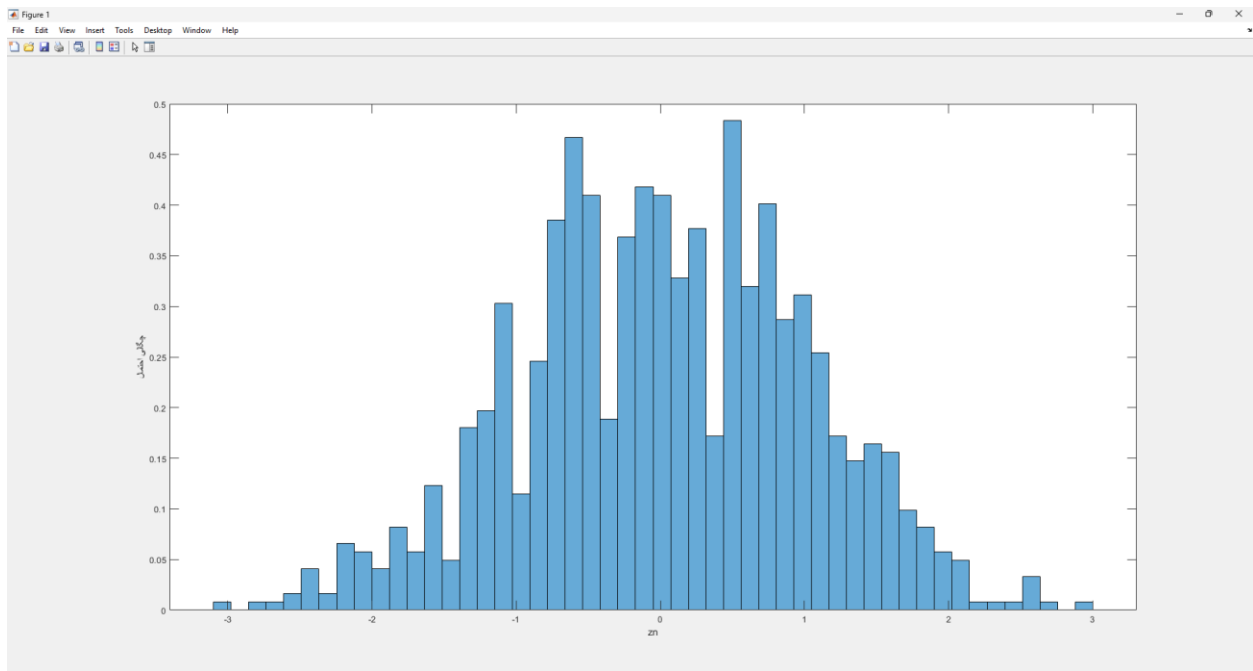
```
mean_zn = mean(Zn_exp);  
disp(['mean Zn: ', num2str(mean_zn)]);  
  
var_zn = var(Zn_exp);  
disp(['var Zn: ', num2str(var_zn)]);
```

قسمت د:

حال توزیع برنولی مورد سوال است:



حال تابع هیستوگرام را برای این توزیع رسم میکنیم:



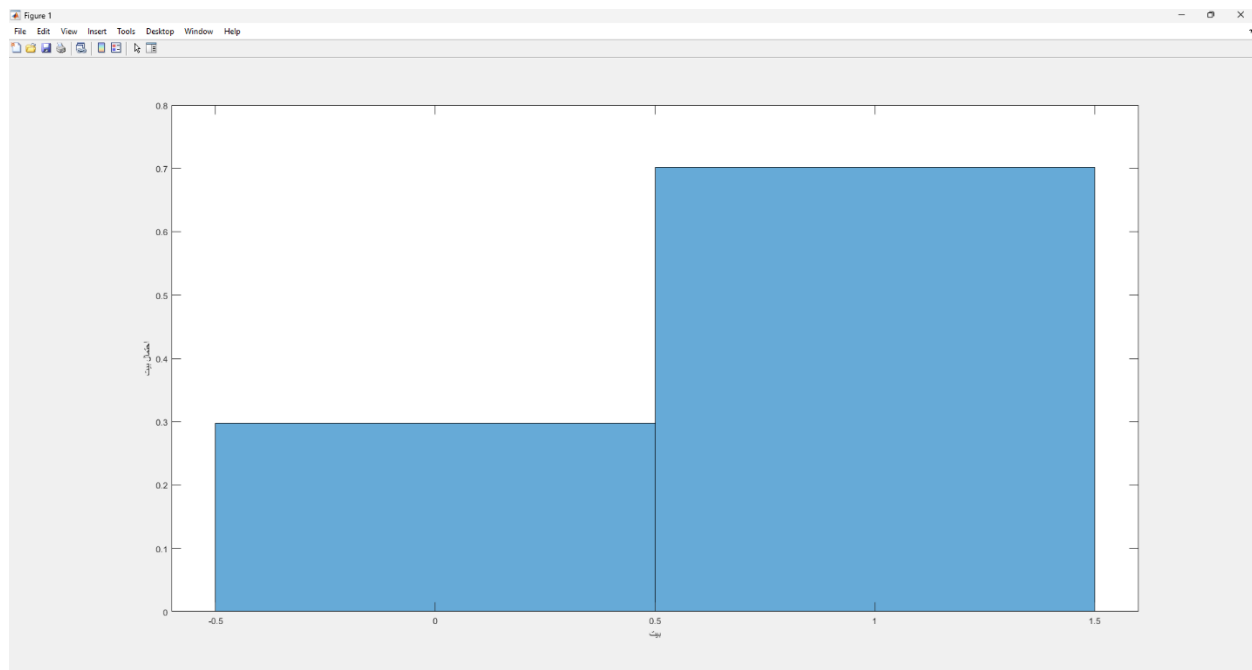
پروژه دوم:

قسمت الف: ابتدا احتمال های داده شده صورت سوال را به صورت دو متغیر تعریف می کنیم.

سپس به کمک تابع rand ، n عدد بین صفر و یک تولید میکنیم. حال اگر عدد کوچکتر از 0.7 باشد آن را بیت 1 در نظر میگیریم و اگر نباشد بیت 0.

```
% 2.1:
p_0 = 0.3;
p_1 = 1 - p_0 ;
n = 20000;
numbers = rand(1 , n) < p_1;

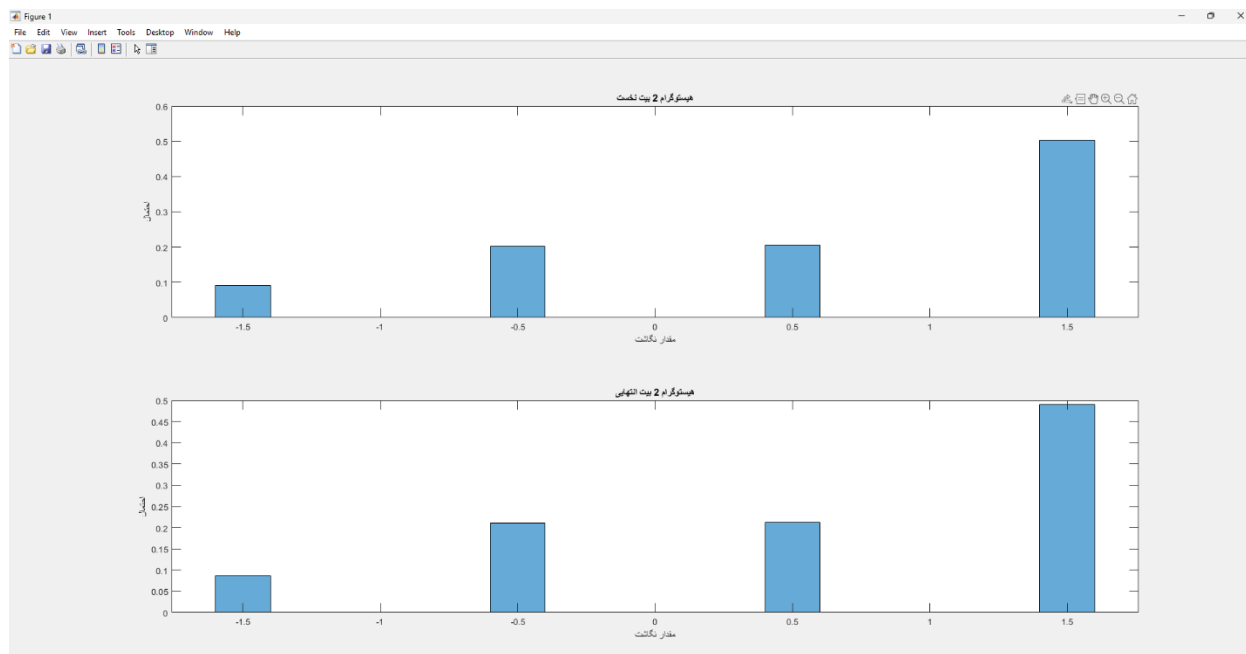
histogram(numbers, 'Normalization', 'pdf');
xlabel('بیت');
ylabel('احتمال بیت');
```



قسمت دوم:

در اینجا می بایست طبق گفته سوال بیت ها را دو به دو تقسیم کنیم و در دو گروه بریزیم
برای مثال دو بیت ابتدایی در گروه اول و دو بیت دوم در گروه دوم.

کد این بخش طولانی بود در گزارش نیاوردم!



الان ما دو گروه 5 هزار عددی از عدد های جدول شماره 1 داریم.

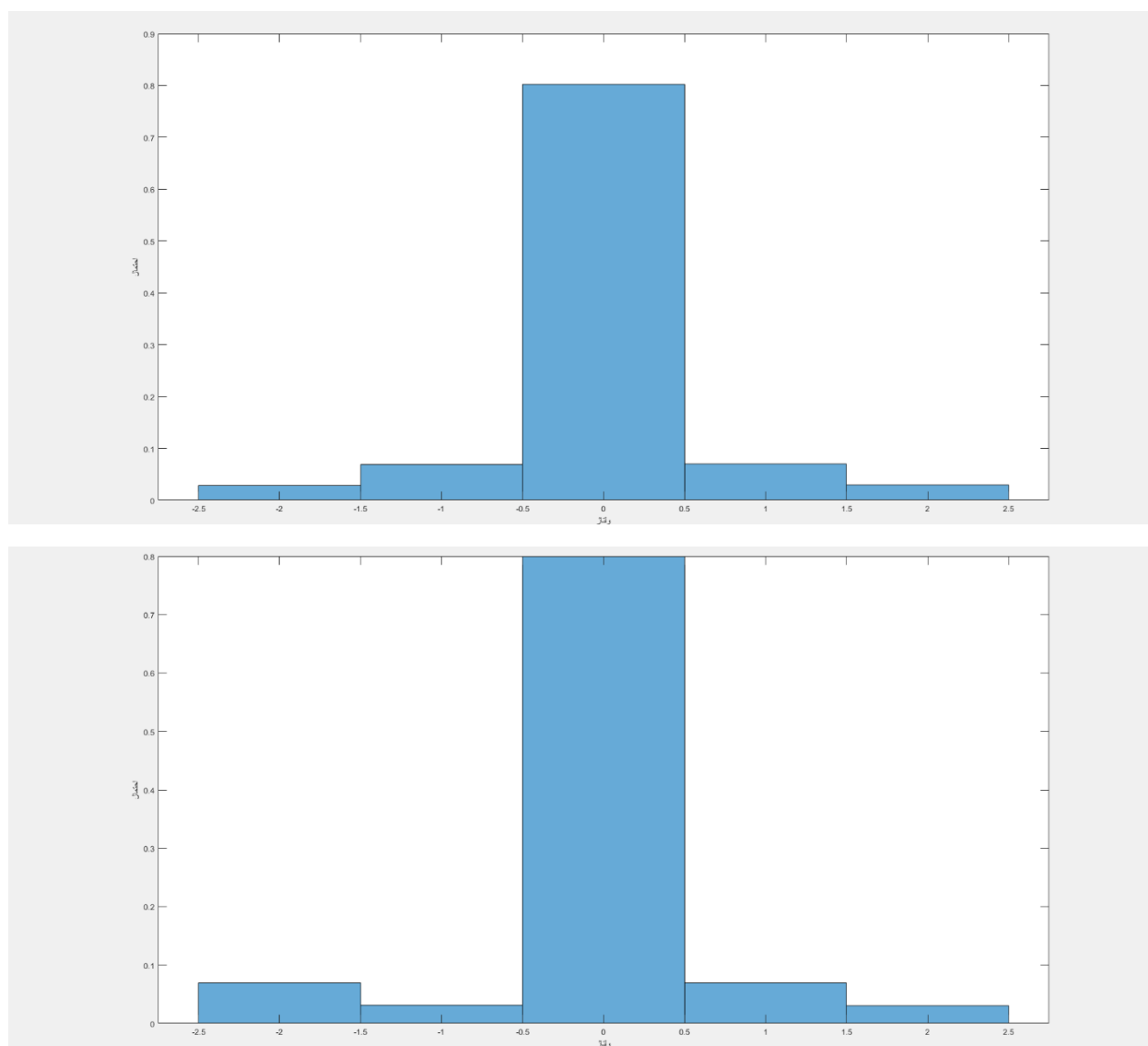
قسمت ج:

در این بخش سوال گفته است که ولتاژ ها را به کانال a و b نگاشت کنیم. در نمودار های داده شده احتمال ها داده شده است.

تنها کافی است هر عدد رو نمودار را با احتمالش متناظر کنیم.

```
% 2.3:  
n = 5000  
numbers = rand(1, n) < p_1;  
voltage = [-2, -1, 0, 1, 2];  
prob_voltage_a = [0.03, 0.07, 0.8, 0.07, 0.03];  
prob_voltage_b = [0.07, 0.03, 0.8, 0.07, 0.03];  
noise_a = randsrc(1,n,[voltage; prob_voltage_a]);  
noise_b = randsrc(1,n,[voltage; prob_voltage_b]);
```

برای a و b به ترتیب خواهیم داشت:



قسمت د:

حال ما هم بردار های ورودی را داریم ، دو گروه 5 هزارتایی ؛ و هم دو گروه 5000 تایی نویز.
صرفا کافی است این گروه ها متناظر باهم جمع شوند.

| | |
|--------------|---------------|
| group_first | 1x5000 double |
| group_second | 1x5000 double |
| i | 19997 |
| index | 5001 |
| n | 5000 |
| noise_a | 1x5000 double |
| noise_b | 1x5000 double |

حال می آییم گروه final_a را که 5000 هزار عدد است ، هر عدد را تبدیل به دو بیت میکنیم تا در نهایت 10000 بیت شود و این را در کنار 10000 بیت دوم قرار دهیم تا 20000 بیتی که در ابتدا داشتیم دوباره ساخته شود.

```
% 2.4:
finall_a = noise_a + group_first;
finall_b = noise_b + group_second;

n = length(finall_a);
mapped_bits_a = "";

for i = 1:n
    if finall_a(i) <= -1
        mapped_bits_a = strcat(mapped_bits_a, "00");
    elseif finall_a(i) > -1 && finall_a(i) <= 0
        mapped_bits_a = strcat(mapped_bits_a, "01");
    elseif finall_a(i) > 0 && finall_a(i) <= 1
        mapped_bits_a = strcat(mapped_bits_a, "10");
    elseif finall_a(i) > 1
        mapped_bits_a = strcat(mapped_bits_a, "11");
    end
end

bit_sequence = char(mapped_bits_a);
disp(['طول دنباله بیت نهایی: ', num2str(length(bit_sequence))]);

mapped_bits_b = "";
for i = 1:n
    if finall_b(i) <= -1
        mapped_bits_b = strcat(mapped_bits_b, "00");
    elseif finall_b(i) > -1 && finall_b(i) <= 0
        mapped_bits_b = strcat(mapped_bits_b, "01");
    elseif finall_b(i) > 0 && finall_b(i) <= 1
        mapped_bits_b = strcat(mapped_bits_b, "10");
    elseif finall_b(i) > 1
        mapped_bits_b = strcat(mapped_bits_b, "11");
    end
end

bit_sequence_2 = char(mapped_bits_b);
disp(['طول دنباله بیت نهایی: ', num2str(length(bit_sequence_2))]);
```

حال کافی است این دو 10000 را با ترتیبی درست در کنار هم قرار بدهیم:

```
if length(mapped_bits_a) ~= length(mapped_bits_b)
    error('باید برابر باشد a و b طول رشته‌های');
end

n = length(mapped_bits_a);
result = "";

for i = 1:2:length(bit_sequence)-1
    part_a = bit_sequence(i:i+1);

    part_b = bit_sequence_2(i:i+1);

    result = result + part_a + part_b;
end
```