

# Analysis of Random Variables Using MATLAB

---

*Instructor: Dr. Samy Soliman*

*Course Code: CIE 327*

**Aml Tarek**

**Mohammad Mahmoud Ibrahim**

**Youssef Allam**

## Table of Contents

1) Project Description.....	3
2) Single Variables.....	3
Probability Distribution and Cumulative Distribution .....	3
Statistical Expectations .....	4
Moment Generating Function .....	5
3) Single Variable Testing.....	6
Test Case 1: $X \sim \mathcal{U}(-5, 2)$ .....	6
Test Case 2: $X \sim \mathcal{N}(3, 4)$ .....	8
Test Case 3: $X \sim \text{Bin}(5, 0.3)$ .....	10
Test Case 4: $X \sim \text{Poisson}(10)$ .....	11
4) Joint Random Variables .....	12
Joint Probability Distribution .....	13
Marginal Probability Distributions .....	13
Statistical Expectations .....	14
5) Joint Variable Testing.....	16
Test Case 1: $X \sim \mathcal{N}(3, 4)$ ; $Y \sim \mathcal{N}(-5, 2)$ :.....	16
Test Case 2: $X \sim \Gamma(2, 10)$ ; $Y \sim \text{Bin}(4, 0.5)$ .....	18
Test Case 3: $X \sim \text{Exp}(0.05)$ ; $Y = 3X + 2$ .....	19
Test Case 4: $X \in \{-1, 1\}$ ; $Y = X + n$ ; $n \sim \mathcal{N}(0, 0.5)$ .....	21
6) Functions of RVs .....	22
7) Function of RV Testing.....	23
Test Case 1: $X \sim \mathcal{N}(3, 4)$ ; $Y \sim \mathcal{N}(-5, 2)$ :.....	24
Test Case 2 : $X \sim \Gamma(2, 10)$ ; $Y \sim \text{Bin}(4, 0.5)$ .....	25
Test Case 3: $X \sim \text{Exp}(0.05)$ ; $Y = 3X + 2$ .....	26
Test Case 4: $X \in \{-1, 1\}$ ; $Y = X + n$ ; $n \sim \mathcal{N}(0, 0.5)$ .....	27
8) GUI Generation .....	28

## 1) Project Description

The goal of this project is to use MATLAB to perform the random variable analysis that we did throughout the course whether for single random variables or joint random variables. The data used will be a sample space of large enough size to allow us to perform statistical calculations on. The program will be presented in the form of GUI that performs the following functionalities for each of 3 cases discussed in this report.

## 2) Single Variables

For the case of single random variables, the user will enter a single 1D array of the sample space.

### Probability Distribution and Cumulative Distribution

Plotting the histogram would give us a very similar plot to the probability density function, however it would need to be normalized to meet the axioms of probability. This could be done through the built in MATLAB function *histcounts*.

The concept behind histcounts is that it takes a large sample space of various scattered numbers and divides them into bins. It then calculates the total number of samples within that bin. It can also take a parameter of 'Normalization' and 'PDF'/'CDF'/'Probability'. The Parameters 'Normalization' and 'PDF' tell the function to normalize the bins into a PDF while 'Probability' normalizes it into a PMF. The parameters 'Normalization' and 'CDF' tell the function to normalize the bins into CDFs by adding the values of the previous bins to the next bin.

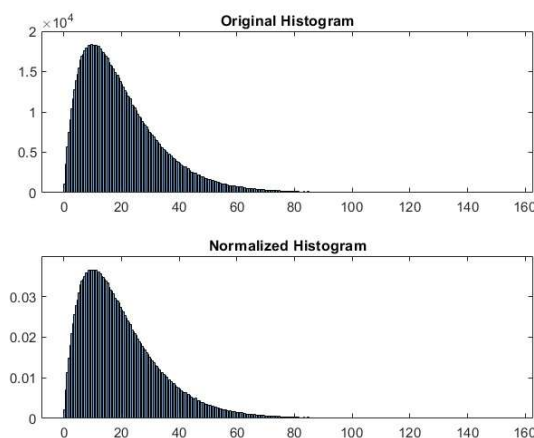


Figure 2.1 (Original vs Normalized Histograms)

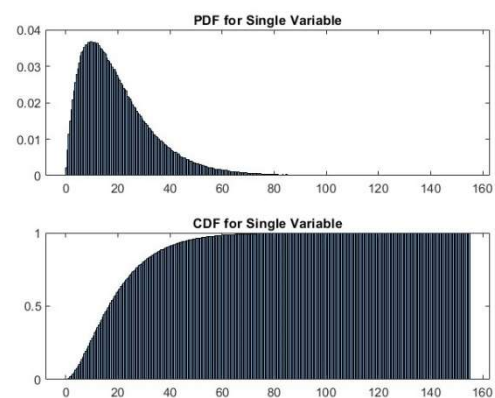


Figure 2.2 (PDF and CDF for Sample Data)

```

if(size(unique_values)/size(X)<0.75)
[pdf_counts, pdf_edges] = histcounts(X,'Binwidth',0.01,'Normalization', 'probability');
histogram('BinEdges',pdf_edges,'BinCounts',pdf_counts,'Normalization','probability');
xlabel('Value');
ylabel('Probability');
title('Probability Mass Function (PMF)');
end

if(size(unique_values)/size(X)>0.75)
[pdf_counts, pdf_edges] = histcounts(X,'Binwidth',0.01,'Normalization', 'pdf');
histogram('BinEdges',pdf_edges,'BinCounts',pdf_counts,'Normalization','pdf');
xlabel('Value');
ylabel('Probability');
title('Probability Mass Function (PMF)');
end

```

The function returns 2 arrays *counts* and *edges*. The *edges* array represents the values on edges of the bins while the *counts* array represents the value between 2 edges. In other words, the value of the function between edges(i) to edges(i+1) is count(i). Note that we use different calls to the histcounts depending on whether a function is discrete or not. The check for a discrete function assumes that if the count of unique values is less than 75% of the count of the sample then it is discrete else it is continuous.

## Statistical Expectations

Then we use the definitions of mean, variance and third moment to calculate them.

The mean is by definition  $E\{x\} = \sum x f(x)$  where  $f(x) = P(X = x)$  and  $X$  is a random variable. In our case we can easily calculate the mean by  $\frac{1}{N} \sum_{i=1}^N x_i$  where  $x_i$  is the  $i$ th sample. Since the sample space contains all possible values for  $X$  as well and the more probable values will be more frequent making them have more effect over the average.

```

%Mean
sum_single=0;
for i=1 : length(X)
sum_single=sum_single+X(i);
end
mean=sum_single/length(X)

```

```
mean: 20.0127
```

The same logic can be applied to the variance as well as the third moment. The variance will be given by  $\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2$  where  $\mu$  is the mean calculated from before. The standard deviation could also be found by taking the square root of the variance. Similarly to the mean the third moment will be found by  $\frac{1}{N} \sum_{i=1}^N x_i^3$ .

```

%Variance
sum_single=0;
for i=1:length(X)
    sum_single=sum_single+(X(i)-mean)^2;
end
var=sum_single/length(X)
std=sqrt(var)

```

```

%Third Moment
sum_single=0;
for i=1:length(X)
    sum_single=sum_single+(X(i))^3;
end
moment3=sum_single/length(X)

```

```

var: 200.0529
std: 14.1440
moment3: 2.4003e+04

```

## Moment Generating Function

Finally, we calculate and plot the MGF as well as it's first and second derivative then evaluate them at  $T=0$ .

To ease the calculation of the derivative we chose to use symbolic variables (*syms*). However, in this case we will not be able to operate directly on the entire input array since the program will probably run out of memory trying to deal with a lot of numbers in symbolic variables. Alternatively, however, we could use the arrays returned by the `histcounts` function. We will calculate the centers between each 2 edges and multiply that by the bin width between these 2 edges to calculate the probability of that bin and use the definition for the MGF  $M_x(t) = \mathcal{E}\{e^{xt}\}$  when we apply that to our current situation we get  $M_x(t) = \sum_{i=1}^{counts} e^{t*center(i)} f(x_i) = \sum_{i=1}^{counts} e^{t*center(i)} * pdf_{counts(i)} * bin\_width(i)$

```

%MGF
syms t;
syms mgf(t);
mgf=0;
bin_widths = diff(pdf_edges);
for j=1 :length(pdf_counts)
    center=(pdf_edges(j)+pdf_edges(j+1))/2;
    mgf=mgf+bin_widths(j)*pdf_counts(j)*exp(center*t);
end

```

We next use the `diff` function to differentiate the MGF with respect to  $t$  once and twice and evaluate both derivatives at  $t=0$

```

%1st Derivative of MGF
syms m1(t);
m1=diff(mgf,t);
em1=simplify(subs(m1,t,0))

%2nd Derivative of MGF
syms m2(t);
m2=diff(m1,t);
em2=simplify(subs(m2,t,0))

em1: 20.0127
em2: 600.5811

```

Of course, the values of the first derivative at 0 must be equal to the previously calculated mean and the second derivative at 0 must be equal to the variance + the mean squared which if we check is the case.

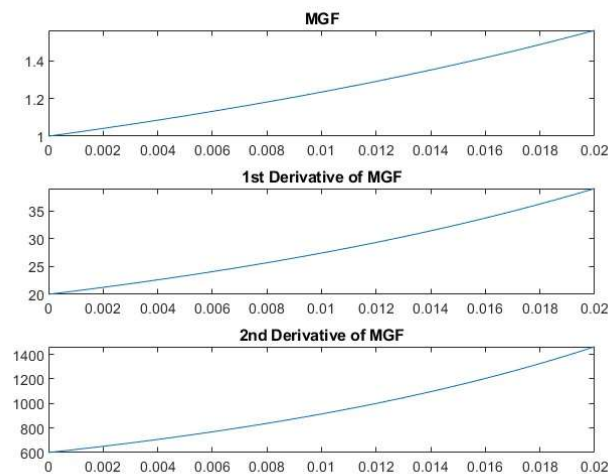


Figure 2.3 (Plots of MGF and its Derivatives)

### 3) Single Variable Testing

We used the test case provided above to explain how the program works. We will next generate the required test cases and run our program for them. Most of these data samples are generated using the *machine learning and statistics* toolbox to provide the most accurate data samples. We defined a variable size with a value of one million for usage in all the test cases. At the end of each section a table will be shown displaying statistics calculated by the program and manually calculated statistics from the definition of the distribution.

#### Test Case 1: $X \sim \mathcal{U}(-5, 2)$

Before generating the function, we can calculate the expected value and variance using the fact that it is a uniform distribution.

$$\varepsilon(x) = \frac{b+a}{2} = \frac{-3}{2} = -1.5$$

$$\sigma^2 = \frac{(b-a)^2}{12} = \frac{49}{12} \approx 4.0833$$

We generated a uniform distribution with a minimum of -5 and a maximum of 2 by creating an array of size `size`. After setting the first element in the array to the minimum (-5) we linearly added  $\frac{7}{size}$  across `i` until `X` reaches a value of  $-5+7=2$ .

```
%X2=U~(-5,2)
Single_X2=zeros(1,size);
for i=1: size
    Single_X2(i)=-5+7*i/size;
end
```

After generating the test case we run our program on it.

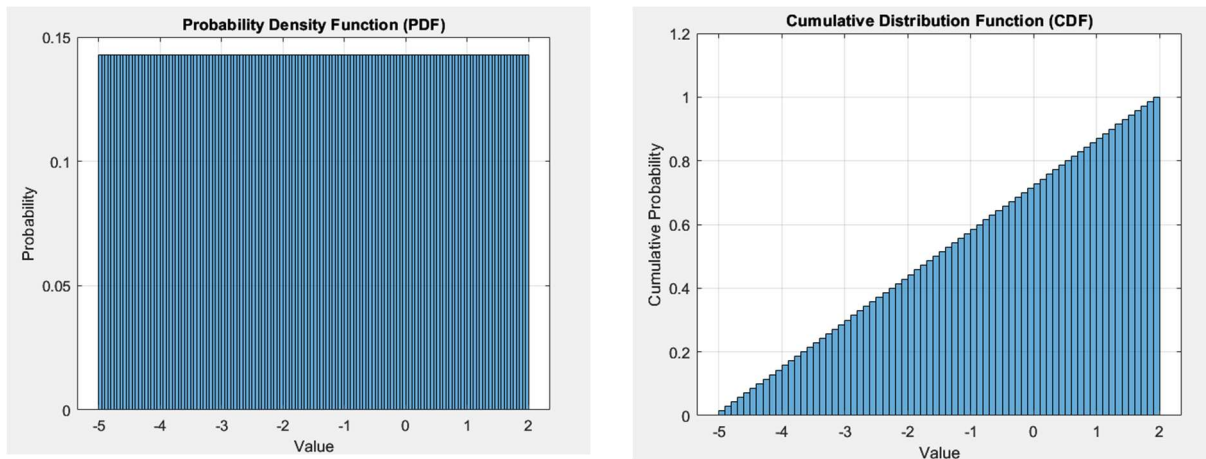


Figure 3.1.1 (PDF and CDF for Test Case 1)

One interesting thing to note here is that the integration of the pdf is  $0.14286 \times 7 \approx 1$  indicating that it is a valid pdf.

Lastly, the plot of the MGF as well as its initial values that agree with the previously generated statistics and manual calculations.

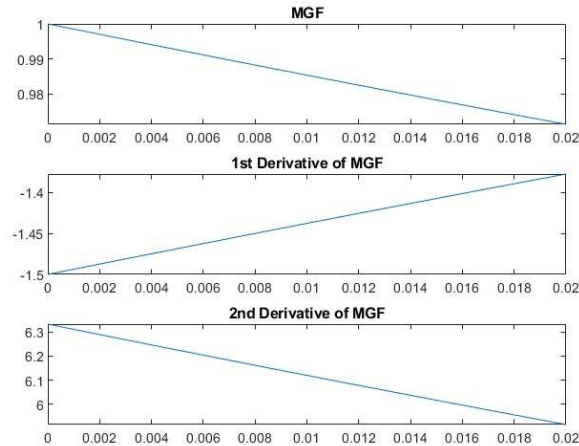


Figure 3.1.2 (MGF and its Derivatives for Test Case 1)

	Mean	Variance	Standard Deviation	$Mx'(0)$	$Mx''(0)$
<b>Theoretical Value</b>	-1.50	4.08	2.02	-1.50	6.33
<b>Calculated Value</b>	-1.50	4.08	2.02	-1.50	6.33
<b>Relative Error(%)</b>	0.00	0.00	0.00	0.00	0.00

Table 1

### Test Case 2: $X \sim \mathcal{N}(3, 4)$

To generate a normal distribution with a mean of 3 and a standard deviation of 4 we use the function *randn* which generates a normal standard distribution which has a mean of 0 and a standard deviation of 1. We use the transformation  $Z = \frac{X - \mu}{\sigma}$  where Z is normal standard distribution and X is the normal distribution with mean  $\mu$  and standard deviation  $\sigma$ . The inverse transformation which is the one we need will be given  $X = Z\sigma + \mu$  where Z is the normal standard distribution generated by *randn*.

```
%X3~N(3,4)
Single_X3=3+4*randn(1,size);
```

We then run our program and analyze the results.



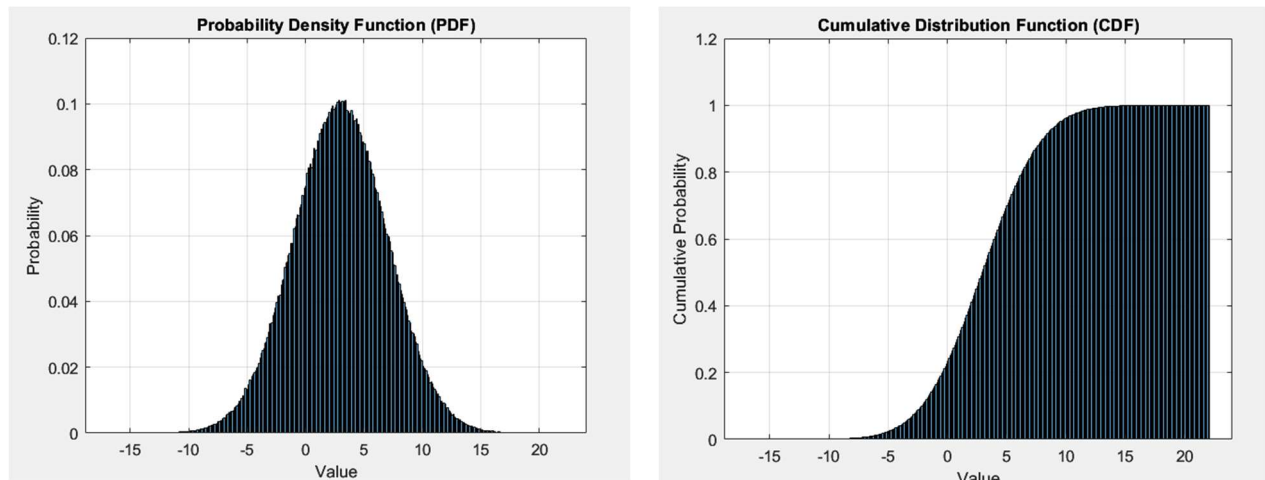


Figure 3.2.2 (PDF and CDF of Test Case 2)

We see that both plots look like normal distributions, however only the normalized could be a probability distribution as we can verify that its integration is 1.

Given that the distribution is a normal distribution with mean 3 and standard deviation 4 and variance 16 we naturally expect these values to be close to the generated statistics.

Finally, we graph the MGF as well as its derivatives.

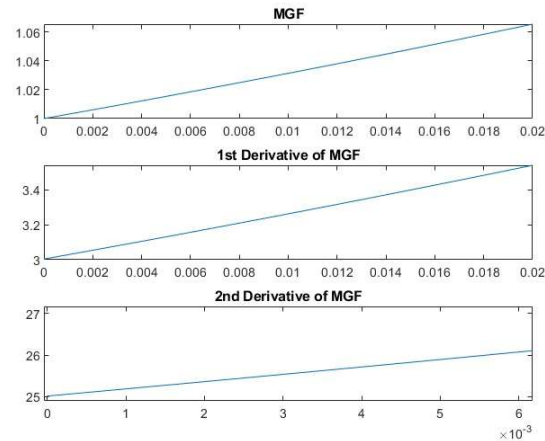


Figure 3.2.2 (MGF and its derivatives for Test Case 2)

If we check the initial values of the first and second derivatives of the MGF we see that they agree with our previously calculated values.

	Mean	Variance	Standard Deviation	$Mx'(0)$	$Mx''(0)$
Theoretical Value	3.00	16.00	4.00	3.00	25.00
Calculated Value	3.00	15.99	4.00	3.00	25.01
Relative Error(%)	0.13	0.06	0.03	0.13	0.05

Table 2

### Test Case 3: $X \sim \text{Bin}(5, 0.3)$

The next test case will be the generation of binary distribution with a sample size of 5 and a probability of success 0.3.

We can easily calculate the statistics of binomial distribution.

$$\varepsilon(x) = np = 1.5$$

$$\sigma^2 = np(1 - p) = 1.05$$

We will use the *binornd* function which generates the samples of a binomial distribution.

```
%X4~Bin(5,0.3)
Single_X4=binornd(5,0.3,1,size);
```

Following the same procedures as before we first generate the histograms, PDF, and CDF.

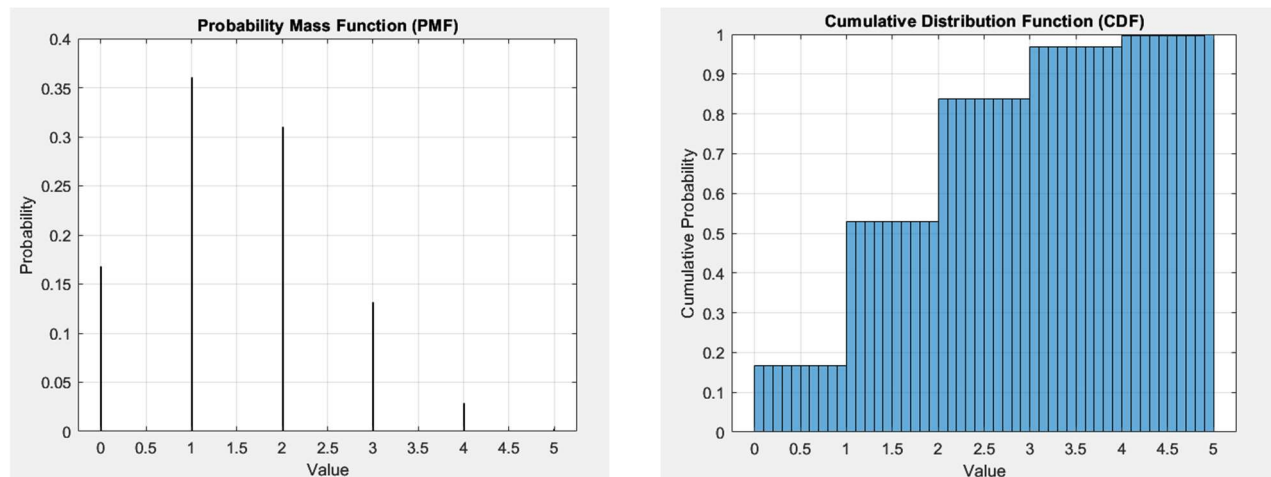
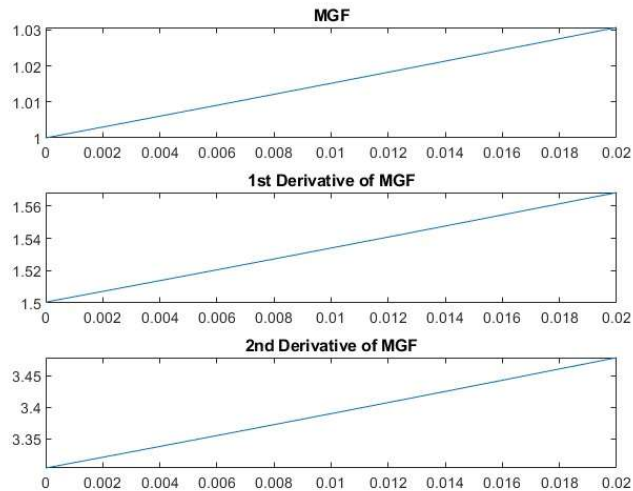


Figure 3.3.1 (PMF and CDF for Test Case 3)

Note that the sum of the PMF values is equal to approximately 1.

Finally, we plot the MGF and its derivatives and check their initial values.



**Figure 3.3.2 (MGF and its derivatives for Test Case 3)**

Looking at the initial values of the MGF derivatives we see they agree with the calculated and generated statistics.

	Mean	Variance	Standard Deviation	$Mx'(0)$	$Mx''(0)$
<b>Theoretical Value</b>	1.50	1.05	1.02	1.50	3.30
<b>Calculated Value</b>	1.50	1.05	1.03	1.50	3.30
<b>Relative Error(%)</b>	0.03	0.14	0.07	0.03	0.09

**Table 3**

### **Test Case 4: $X \sim \text{Poisson}(10)$**

The last test case for our single random variable program will be a Poisson distribution which will be generated using the *poissrnd* function.

```
%X5~Poisson(10)
Single_X5=poissrnd(10, 1,size);
```

We know that a Poisson distribution has a mean of  $\lambda$  and a variance of  $\lambda$ .  $\lambda$  in this case is 10.

Following the same sequence of calculations.

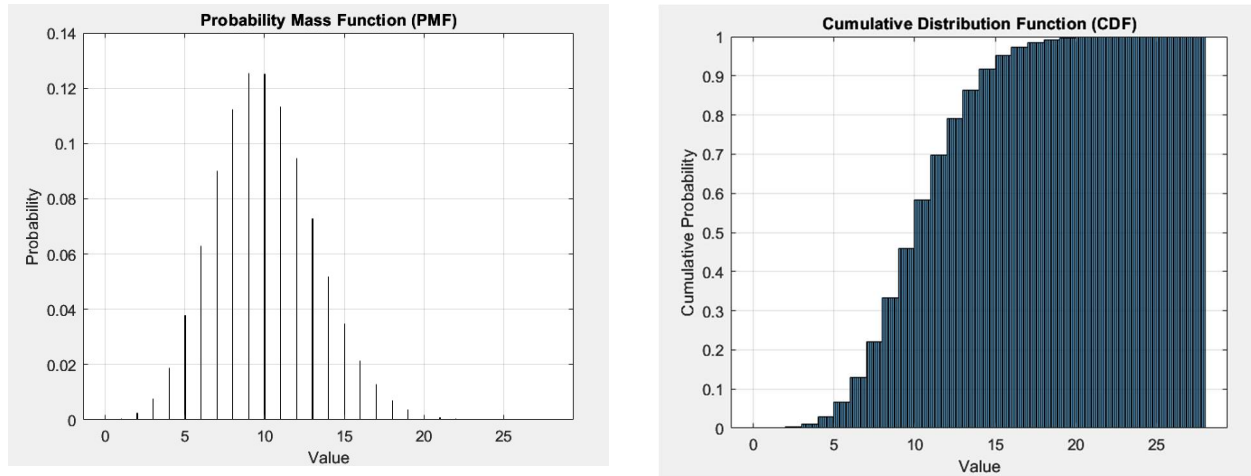


Figure 3.4.1 (PMF and CDF of Test Case 4)

Finally, the graph of the MGF as well as its derivatives is shown below with initial values that agree with the generated and calculated statistics

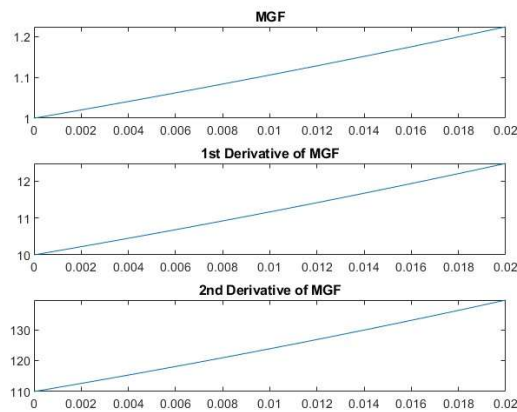


Figure 3.4.2 (Plot of MGF and its derivatives for Test Case 4)

	Mean	Variance	Standard Deviation	$Mx'(0)$	$Mx''(0)$
<b>Theoretical Value</b>	10.00	10.00	3.16	10.00	110.00
<b>Calculated Value</b>	10.00	9.99	3.16	10.00	109.90
<b>Relative Error(%)</b>	0.05	0.05	0.03	0.05	0.09

Table 4

#### 4) Joint Random Variables

The other type of random variables our program will deal with is joint random variables where it will be given a 2d array XY where 1 row represents the X and the second row represents the Y. Similar to

how we presented the single variable we will explain how our program works on the provided test case then show the results for the remaining test cases/

## Joint Probability Distribution

Similar to how we used the function *histcounts* another function *histcounts2* will be used to generate to normalize the sample data and generate the joint probability distribution.

**%Probability Distribution**

```
[N_pdf,Xedges_pdf,Yedges_pdf]=histcounts2(X,Y,'Normalization','pdf');  
[N_cdf,Xedges_cdf,Yedges_cdf]=histcounts2(X,Y,'Normalization','cdf');  
figure()  
histogram2('XBinEdges', Xedges_pdf, 'YBinEdges', Yedges_pdf, 'BinCounts', N_pdf);  
title("Joint Probability Distribution");
```

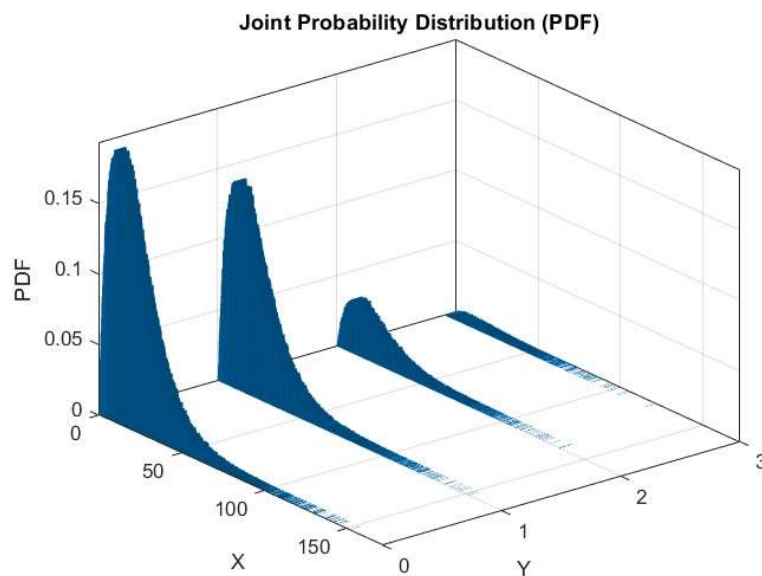


Figure 4.1.1 (Joint Probability Distribution of Provided Test Case)

## Marginal Probability Distributions

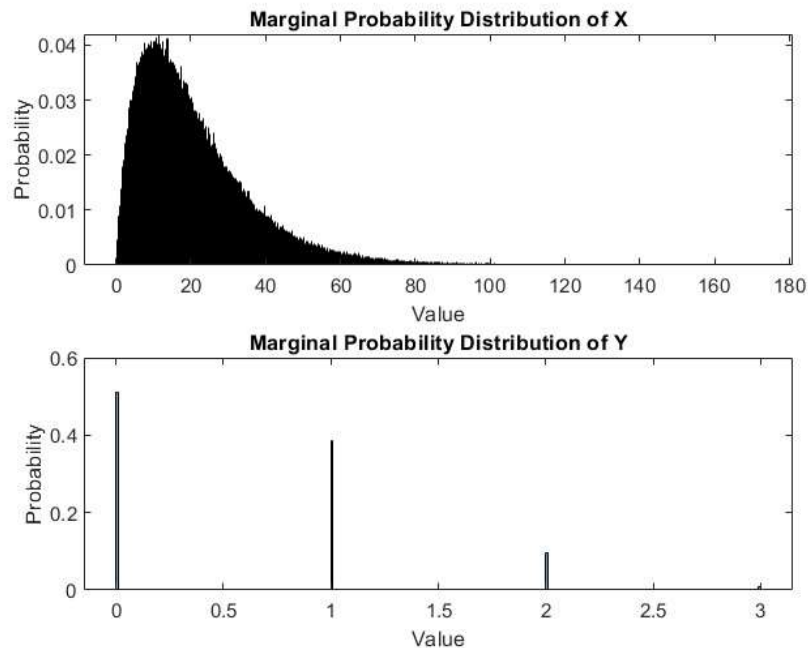
Next, we plot the marginal probability for both X and Y. By looking at the joint probability distribution we can already have an intuition of how the PDFs will look. However, to accurately plot them we will use the same method we used in the single random variables for X and Y independently.

```

%Marginal Probabilities
figure()
[X_pdf_counts,X_pdf_edges]=histcounts(X,'Normalization','pdf');
subplot(2,1,1);
histogram('BinEdges',X_pdf_edges,'BinCounts',X_pdf_counts);
title("Marginal Probability Distribution of X");

[Y_pdf_counts,Y_pdf_edges]=histcounts(Y,'Normalization','pdf');
subplot(2,1,2);
histogram('BinEdges',Y_pdf_edges,'BinCounts',Y_pdf_counts);
title("Marginal Probability Distribution of Y");

```



**Figure 4.1.2 (Marginal Probability Distribution of X and Y)**

The 2 PDFs above must satisfy the conditions of a marginal probability distribution such that the sum of all probabilities must equal 1. This is satisfied due to the use of the “*Normalization*” function like single random variables.

## Statistical Expectations

Next, we look to calculate the mean for both X and Y as well as the covariance and correlation coefficient.

The mean will be calculated in the same way we did in part 1 by dealing with each random variable separately.

```
%Mean
mean_x=0;
mean_y=0;
for i=1 : length(XY(1,:))
    mean_x=mean_x+XY(1,i);
end
mean_x=mean_x/(length(XY(1,:)))

for i=1 : length(XY(2,:))
    mean_y=mean_y+XY(2,i);
end
mean_y=mean_y/length(XY(2,:))

mean_x: 19.9998
mean_y: 0.6006
```

The covariance is by definition  $\sigma_{XY} = \varepsilon\{(X - \mu_X)(Y - \mu_Y)\}$  this can be given in our case as  $\frac{1}{N} \sum_{i=1}^N (X_i - \mu_X)(Y_i - \mu_Y)$ .

```
%Covariance
covar=0;
for i=1 : length(XY(1,:))
    cx=(XY(1,i)-mean_x);
    cy=(XY(2,i)-mean_y);
    covar=covar+cx*cy;
end
covar=covar/length(XY(1,:))
```

```
covar: -0.0031
```

To calculate the correlation coefficient, we must first calculate the standard deviation of both X and Y since the correlation coefficient is given by  $\rho = \frac{\text{covar}(x,y)}{\sigma_x \sigma_y}$ . We already know how to calculate the standard deviation from the single variable part.

```

%STD
std_x=0;
std_y=0;
for i=1:length(XY(1,:))
    std_x=std_x+(XY(1,i)-mean_x)^2;
end
std_x=std_x/length(XY(1,:));
std_x=sqrt(std_x)

for i=1:length(XY(2,:))
    std_y=std_y+(XY(2,i)-mean_y)^2;
end
std_y=std_y/length(XY(2,:));
std_y=sqrt(std_y)

%Correlation
correlation_coeff=covar/(std_x*std_y)

std_x: 14.1442
std_y: 0.6926
correlation_coeff: -3.1360e-04

```

## 5) Joint Variable Testing

Next, we will test the joint variables on the required test cases. We've already shown how to generate most kinds of probability distributions so this will not be different from Single Variable Testing. However, this time we will be generating 2 random variables per test case.

### Test Case 1: $X \sim \mathcal{N}(3, 4)$ ; $Y \sim \mathcal{N}(-5, 2)$ :

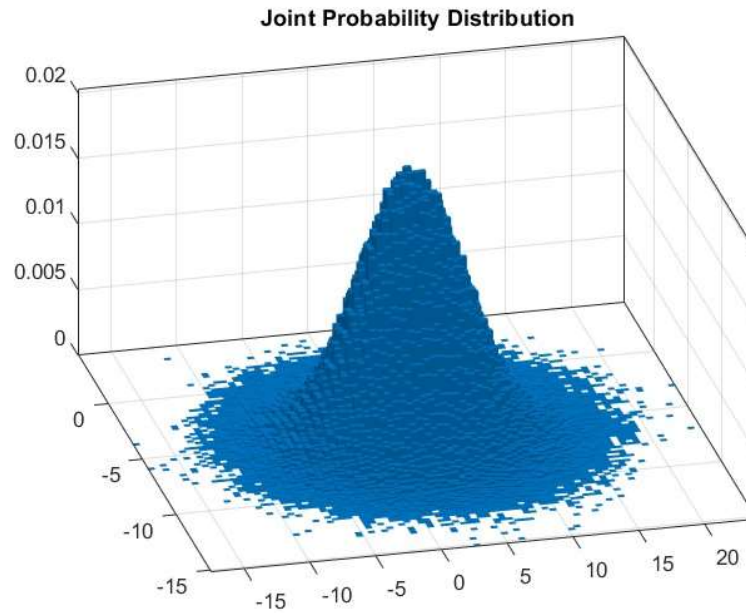
The first test case for the joint variables will be X and Y both normally distributed with X having a mean of 3 and standard deviation of 4 while Y having a mean of -5 and standard deviation of 2.

```

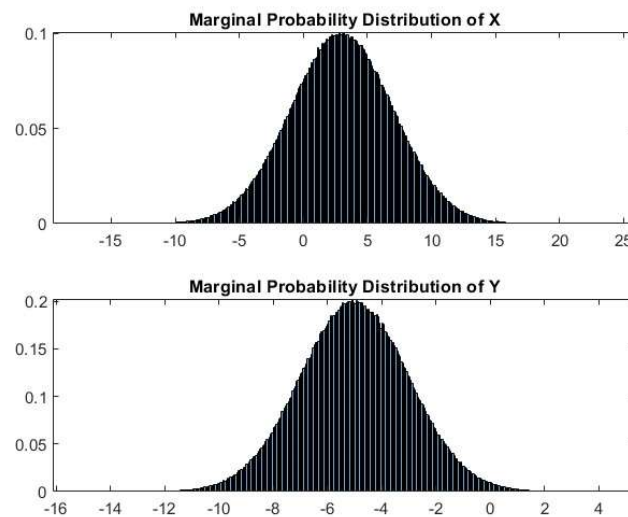
%X2~N(3,4) Y2~N(-5,2)
X2J=3+4*randn(1,size);
Y2J=-5+2*randn(1,size);
XY2=[X2J; Y2J];

```





**Figure 5.1.1 (Joint Probability Distribution of Test Case 1)**



**Figure 5.1.2 (Marginal PDFs for X and Y in Test Case 1)**

Looking at the graph we see that X and Y come out normally distributed with the correct mean and variances as expected. To ensure that they are equal we look at the exact statistics generated.

	Mean of X	Mean of Y	STD of X	STD of Y
Theoretical Value	3	-5	4	2
Generated Value	3.009	-5.004	4.005	2.007
Relative Error(%)	0.3	0.08	0.125	0.35

Table 5

The covariance is approximately -0.0052 and the correlation coefficient is -6.4495e-04 indicating a weak relationship between the variables.

### Test Case 2: $X \sim \Gamma(2, 10)$ ; $Y \sim \text{Bin}(4, 0.5)$

The only new function we will have to introduce here will be *gamrnd* which similar to all the previous functions generates a sample of gamma distribution. The mean of a gamma distribution is given by  $\alpha\theta$  in this case 20. While the standard deviation is given by  $\sqrt{\alpha\theta}$ . In this case it has a value of  $10\sqrt{2}$ .

```
%X3~Gamma(2,10) Y3~Bin(4,0.5)
X3J=gamrnd(2,10,1,size);
Y3J=binornd(4,0.5,1,size);
XY3=[X3J;Y3J];
```

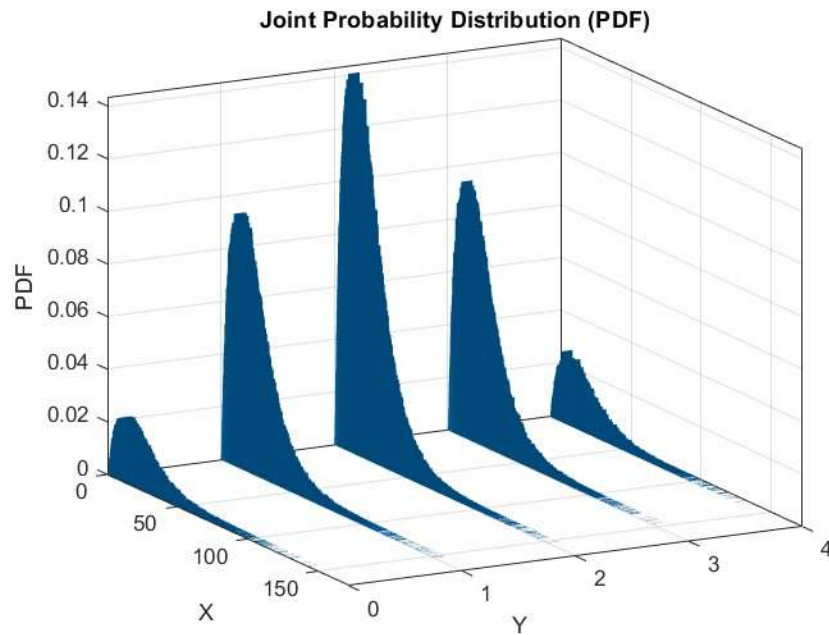


Figure 5.2.1 (Joint Probability Distribution of Test Case 2)

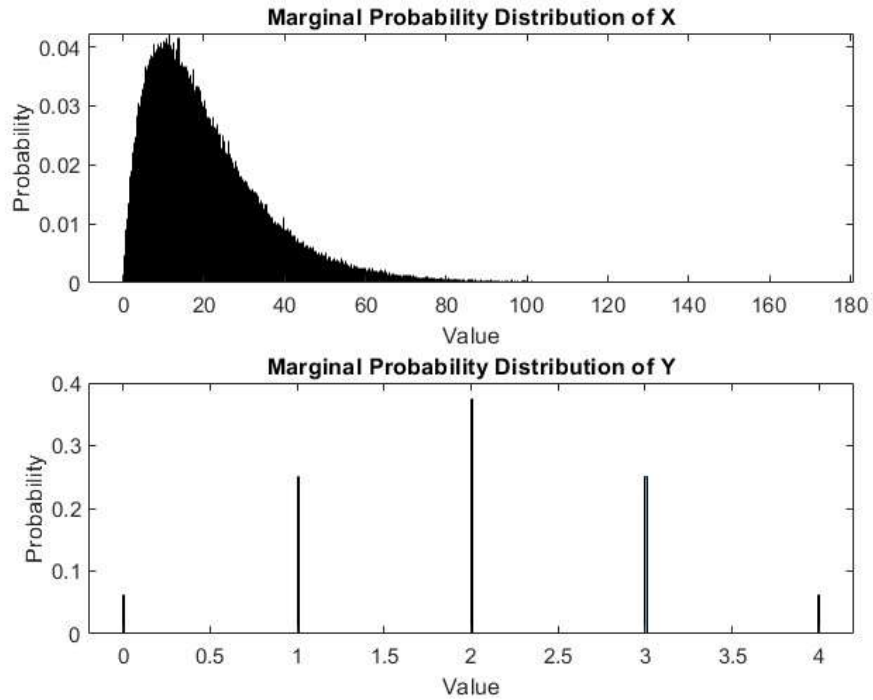


Figure 5.2.2 (Marginal PDFs of X and Y in Test Case 2)

	Mean of X	Mean of Y	STD of X	STD of Y
<b>Theoretical Value</b>	20	2	14.142	1
<b>Generated Value</b>	20.0137	1.9964	14.1556	1.003
<b>Relative Error(%)</b>	0.0685	0.18	0.096167	0.3

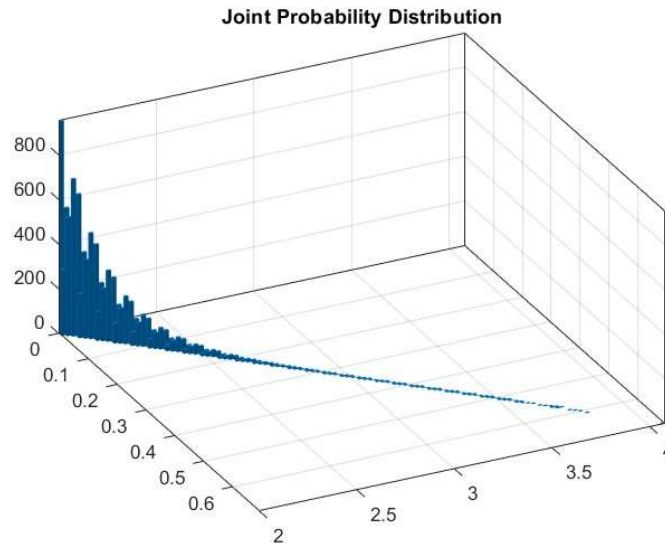
Table 6

The covariance is approximately 0.0120 and the correlation coefficient is approximately  $8.454e-04$  also indicating a weak relationship.

### Test Case 3: $X \sim \text{Exp}(0.05)$ ; $Y = 3X + 2$

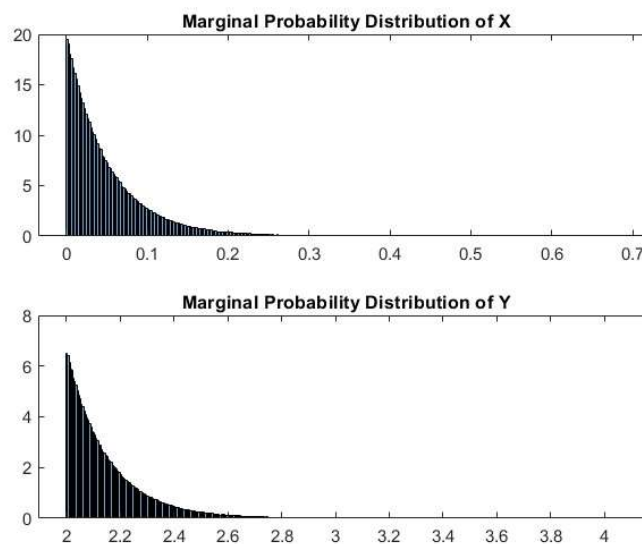
In this case the random variable Y is a function of X so we only need to generate X and then use the relationship between X and Y to evaluate the values of Y. To generate the X values, we use `exprnd` function which generates a sample of values from an exponential distribution. The mean of the exponential distribution is given by  $\frac{1}{\lambda}$  in this case 0.05 and so is the standard deviation since the variance is given by  $\frac{1}{\lambda^2}$ .

```
%X4~exp(0.05) Y4=3X4+2
X4J=exprnd(0.05,1,size);
Y4J=3*X4J+2;
XY4=[X4J; Y4J];
```



**Figure 5.3.1 (Joint Probability Distribution of Test Case 3)**

We note that the top view of the joint probability distribution would represent a straight line this makes sense since there is a direct linear relationship between Y and X.



**Figure 5.3.2 (Marginal PDFs for Test Case 3)**

We note that X and Y are very similar in shape with a difference of scaling and shifting. This is also due to the linear relationship between X and Y.

	Mean of X	Mean of Y	STD of X	STD of Y
Theoretical Value	0.05	2.15	0.05	0.15
Generated Value	0.05	2.1499	0.0499	0.1498
Relative Error(%)	0	0.004651	0.2	0.133333

Table 7

The covariance is 0.0075 and the correlation coefficient is 1 indicating a strong relationship between X and Y. This is naturally expected as the random variable Y is a function of the random variable X.

#### Test Case 4: $X \in \{-1, 1\}$ ; $Y = X + n$ ; $n \sim \mathcal{N}(0, 0.5)$

In this case we have X as a discrete variable with a value of 1 or -1 (both equally probable). Y is a function of X but added to a normal distribution of mean 0 and variance 0.5. To generate the samples for X we make half of the sample size equal to 1 and the other half equal -1. Then we generate a normal distribution with a mean of 0 and a variance of 0.5 to add to X and get Y.

```
%X5~U={-1,1} Y5=X5+n n~N(0,0.5)
X5J=zeros(1,size);
for i=1: 2: size
    X5J(i)=-1;
    X5J(i+1)=1;
end
n1=0.5*randn(1,size);
Y5J=X5J+n1;
XY5=[X5J;Y5J];
```

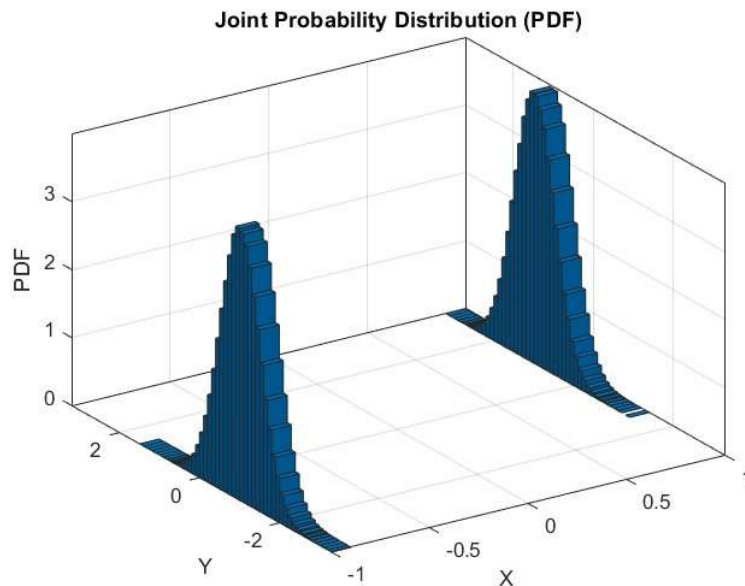


Figure 5.4.1 (Probability Distribution of Test Case 4)

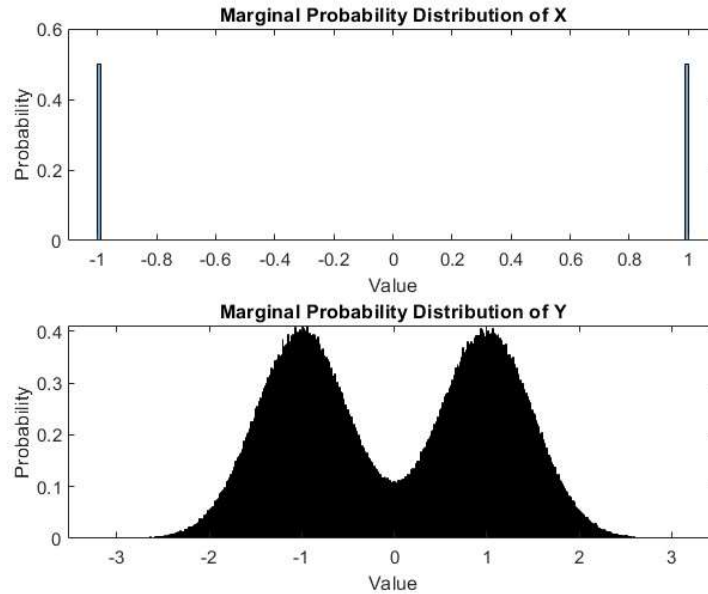


Figure 5.4.2 (Marginal PDFs for Test Case 5)

The graphs for the PDFs show us that  $X$  takes a value of 1 or -1 with an equal probability of 0.5 for both. It also shows the effect of adding continuous noise to  $X$  while maintaining the 2 peaks in  $X$ .

The expected value of  $X$  will simply be the average of 1 and -1, which is 0 while the expected value of  $Y$  will be the expected value of  $X$  plus the expected value of  $n$  which are both equal to 0.

## 6) Functions of RVs

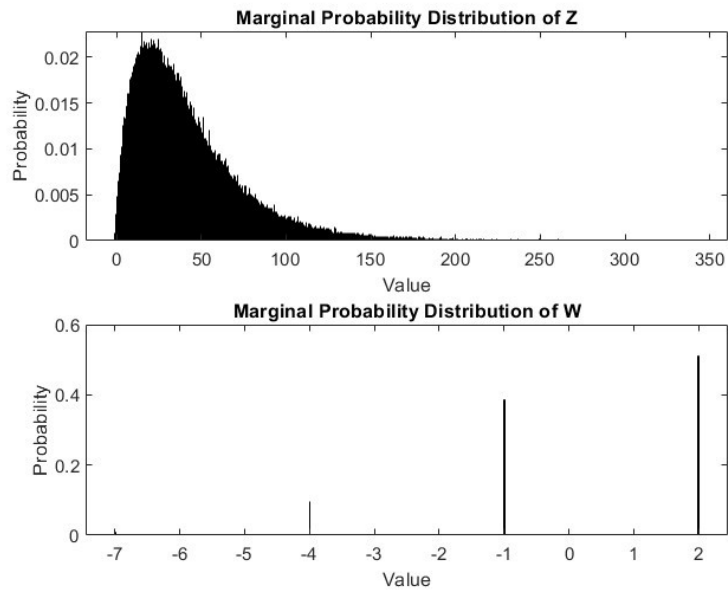
The last functionality we will develop is to allow the user to enter a function of the RVs  $X$  and  $Y$  where  $Z = f(X)$ ,  $W = f(Y)$ . We plot the probability distributions of  $Z$  and  $W$  as well as their joint probability distribution.

To get the function of  $Z$  in terms of  $X$  and  $W$  in terms of  $Y$  we use the *generateAlgebraicFunction* function.

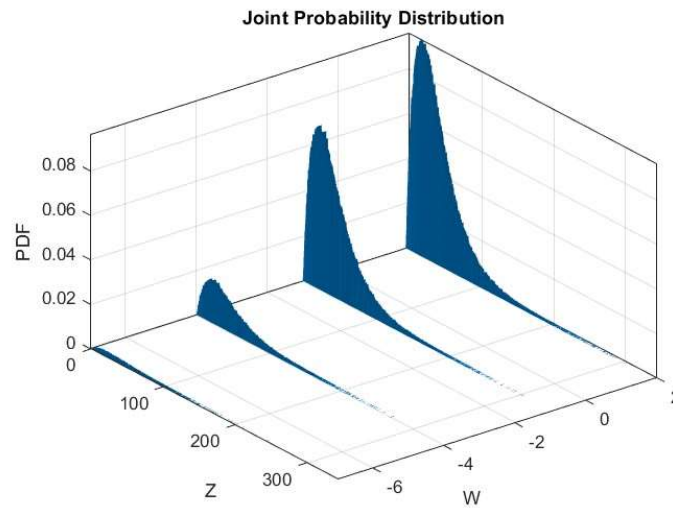
```
% Get Z fn of X
funcz=app.EnterZFunctionofXEditField.Value;
algebraicFuncz=generateAlgebraicFunction(app,funcz,'X');
Z=algebraicFuncz(X);

% Get W fn of Y
funcw=app.EnterWFunctionofYEditField.Value;
algebraicFuncw = generateAlgebraicFunction(app,funcw,'Y');
W=algebraicFuncw(Y);
```

After we generate  $W$  and  $X$  we treat them exactly as we treated  $X$  and  $Y$  in the joint plotting the marginal distribution for each and the joint distribution. We will test our program on the sample case for the function of  $Z = 2X - 1$  and  $W = 2 - 3Y$ .



**Figure 6.1 (Probability Distributions of Sample Data)**

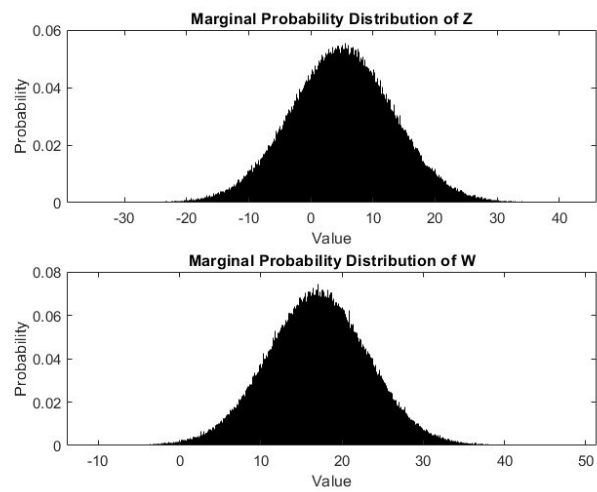


**Figure 6.2 (Joint Probability Distribution of Sample Data)**

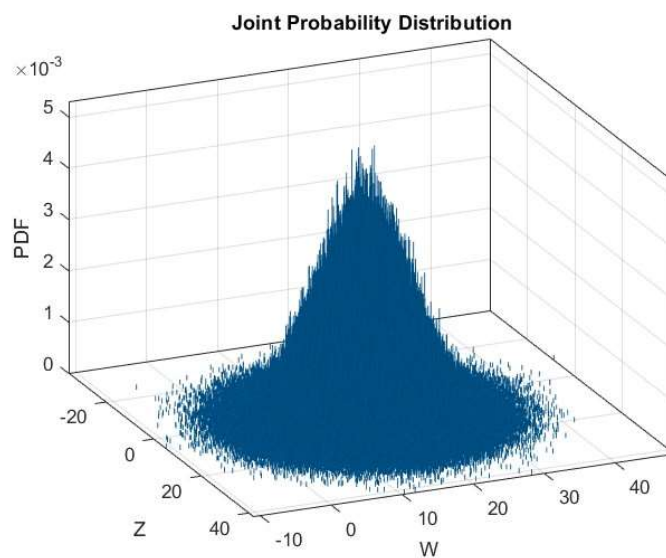
## 7) Function of RV Testing

We will go over the same test cases we went through in single and joint variables and for each test case we will plot the probability distribution of each function as well as their joint probability distribution.

**Test Case 1:  $X \sim \mathcal{N}(3, 4)$  ;  $Y \sim \mathcal{N}(-5, 2)$ :**



**Figure 7.1.1 (Probability Distribution of Z and W for Test Case 1)**



**Figure 7.1.2 (Joint Probability Distribution of Test Case 1)**



Since the original X and Y functions are normal distributions the functions of them are shifted and scaled normal distributions similarly for joint distribution.

### Test Case 2 : $X \sim \Gamma(2, 10)$ ; $Y \sim \text{Bin}(4, 0.5)$

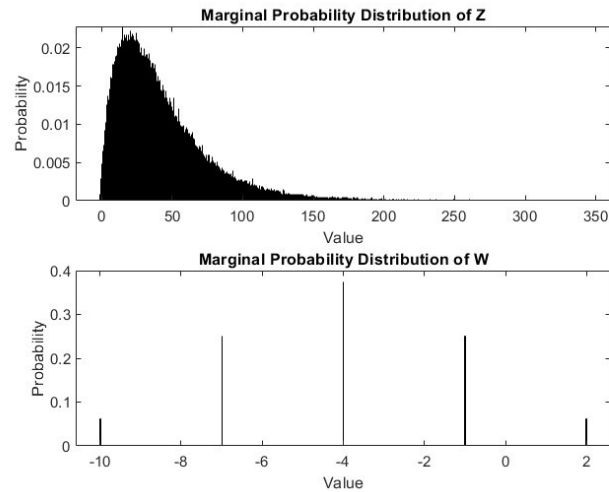


Figure 7.2.1 (Probability Distributions of Z and W for Test Case 2)

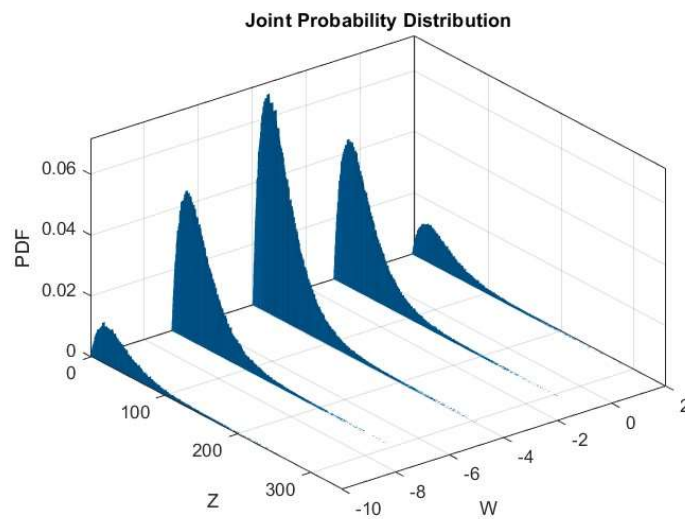


Figure 7.2.2 (Joint Probability Distributions of Test Case 2)

### Test Case 3: $X \sim \text{Exp}(0.05)$ ; $Y = 3X + 2$

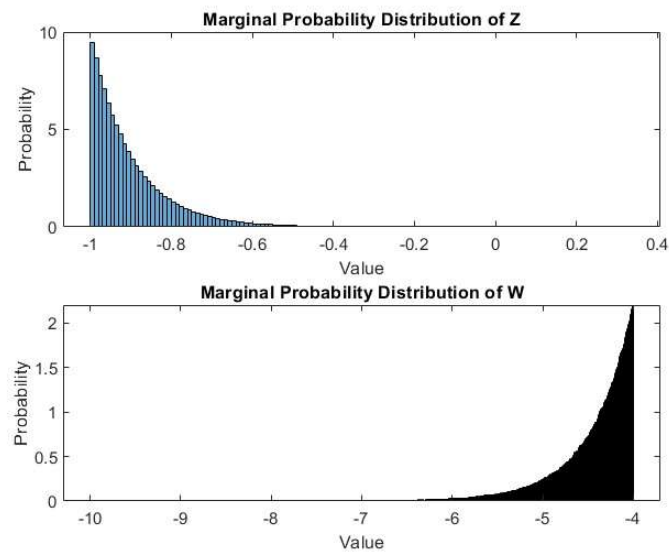


Figure 7.3.1 (Probability Distribution of Z and W for Test Case 3)

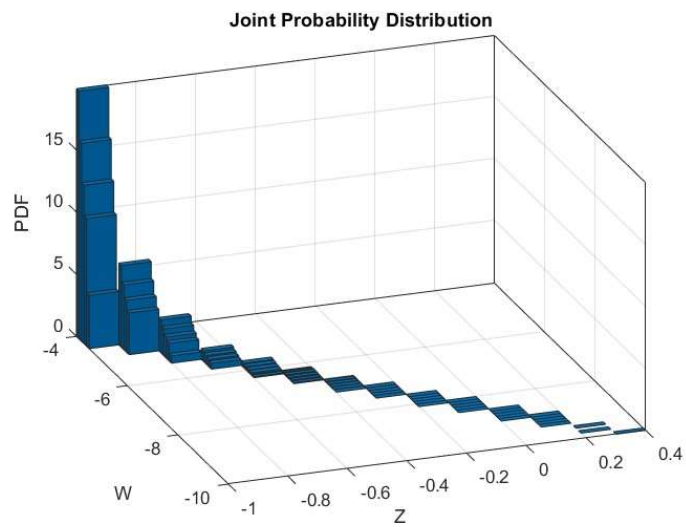
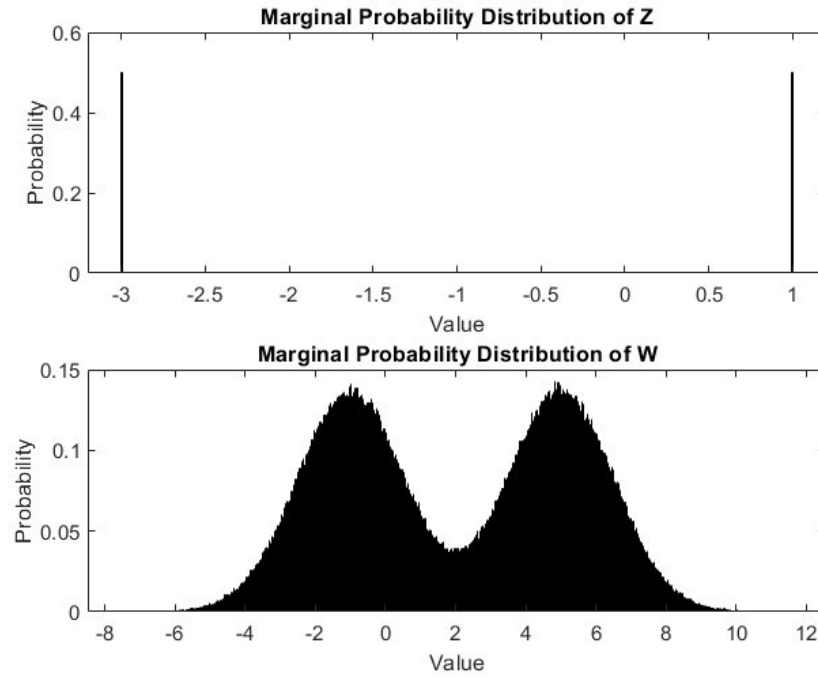
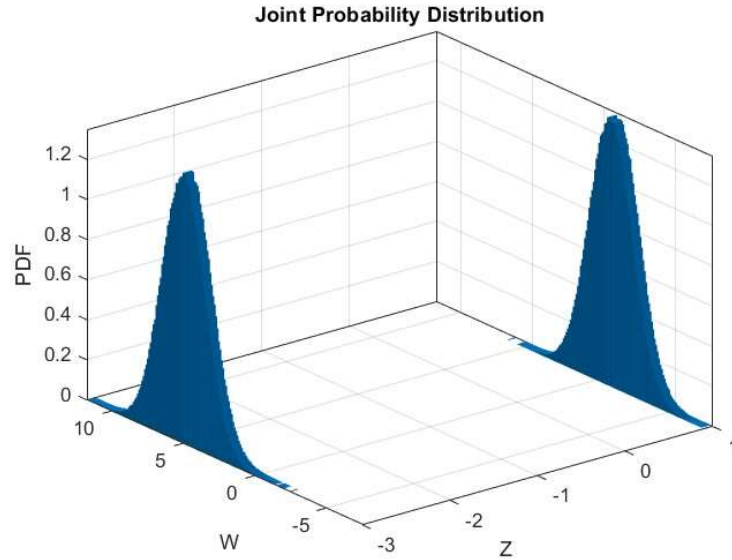


Figure 7.3.2 (Probability Distribution of Test Case 3)

**Test Case 4:  $X \in \{-1, 1\}$ ;  $Y = X + n$ ;  $n \sim \mathcal{N}(0, 0.5)$**



**Figure 7.4.1 (Probability Distributions of Z and Y for Test Case 4)**



**Figure 7.4.2 (Probability Distributions for Test Case 4)**


## 8) GUI Generation

To implement the program as a GUI we used the MATLAB app designer and Designed 3 separate apps for the cases of Single Random Variable, Joint Random Variable, and Function of RV. Below are screenshots for each of the 3 GUIs.

The screenshot shows the MATLAB App window titled 'MATLAB App'. The app interface has a dark blue background. In the top left corner, there is a logo for 'ZEWAIL CITY' with the text 'ESTABLISHED 2000' and 'INAUGURATED 2011'. The main title 'Single Random Variable' is displayed in large white font. Below the title, there is a text input field labeled 'Enter Filename'. Underneath this, there are three buttons: 'View PDF', 'Caculate Statistics' (note the typo), and 'View CDF'. Further down, there are three labels: 'MGF(0)', 'MGF'(0)', and 'MGF''(0)'. To the right of these labels, there is a text input field labeled 'End Value of T' with the value '0.02' entered. Below this, there is a button labeled 'View MGF'. At the bottom, there are three buttons: 'Mean', 'Variance', and 'Third moment'.

The screenshot shows the MATLAB App window titled 'MATLAB App'. The app interface has a dark blue background. In the top left corner, there is a logo for 'ZEWAIL CITY' with the text 'ESTABLISHED 2000' and 'INAUGURATED 2011'. The main title 'Joint Random Variable' is displayed in large white font. Below the title, there is a text input field labeled 'Enter Filename'. Underneath this, there are two buttons: 'Single Statistics' and 'Joint Statistics'. Below these buttons, there are two columns of buttons. The left column has four buttons: 'Mean of X', 'Mean of Y', 'Std of X', and 'Std of Y'. The right column has two buttons: 'Covariance' and 'Correlation Coefficient'.

MATLAB App



# Functions Of Random Variable

Enter Filename

Joint\_XY5

Enter Z Function of X

$2 \cdot X - 1$

Enter W Function of Y

$2 \cdot 3 \cdot Y$

Single Statistics

Joint Statistics

Mean of Z: -1

Mean of W: 2

Standard Deviation of Z: 2

Standard Deviation of W: 3.35

Covariance: -6

Correlation Coefficient: -0.895

Page 29 of 29