ECE 863 Final Project

Sums of Random Variables and Correlated Gaussian Processes

Jacob Honer
College of Electrical and Computer Engineering
Michigan State University
East Lansing, United States of America
honerja1@msu.edu

December 14, 2022

Abstract

This project will contain a series of random experiments demonstrating some of the core concepts pertaining to the ECE 863 course at Michigan State University. The focus of the project is sums of random variables, as well as correlated Gaussian processes. All experiments contained within this report were conducted through Matlab. The complete Matlab script can be found at the end of the document in pdf form; an understanding of the attached script requires only a basic understanding of programming tools, as well as some of the probabilistic concepts outlined by this report.

1 Random Samples

The first experiment of this report utilizes the well-known **rand** function in Matlab:

```
X = rand(N,1);
```

The above function creates an **N** length array of uniformly distributed random variables between 0 and 1. It is obvious that the pdf of such a function is a uniform distribution between 0 and 1.

To confirm this, the **rand** function is used with various values of **N** and histograms are generated using the resulting **X**. It is expected that as **N** increases in value, the histograms will smooth out, better reflecting the shape of the expected pdf. Note that if it was desired to approximate the real-valued pdf, one would have to normalize the histograms to **N**, but as only an observation of the shape of the histogram is desired, this is not done.

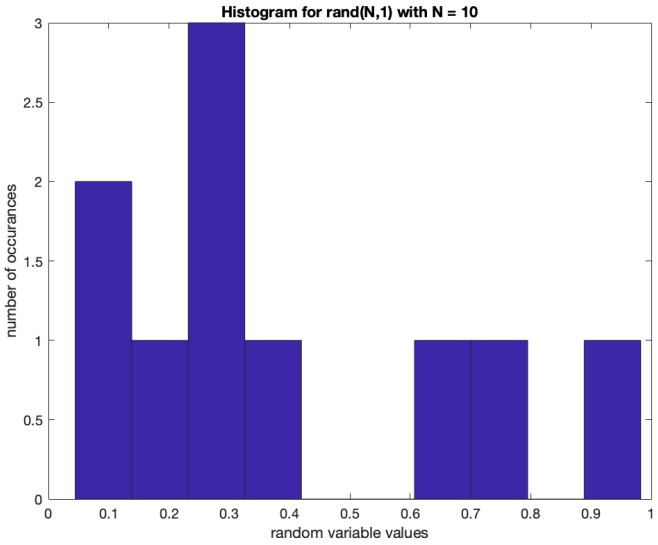


Figure 1: Generated histogram using the rand(N,1) function with N = 10.

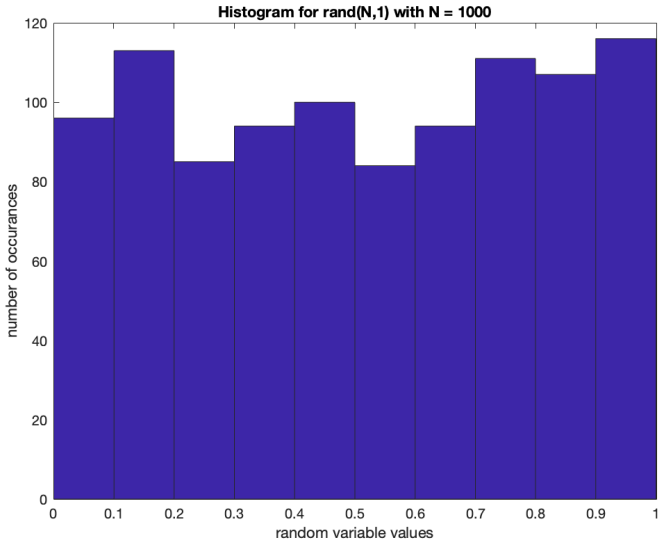


Figure 2: Generated histogram using the rand(N,1) function with N = 1,000.

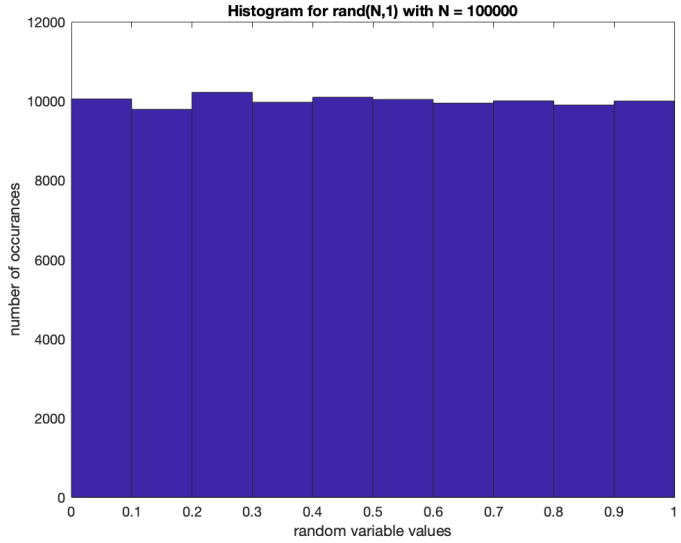


Figure 3: Generated histogram using the rand(N,1) function with N = 100,000.

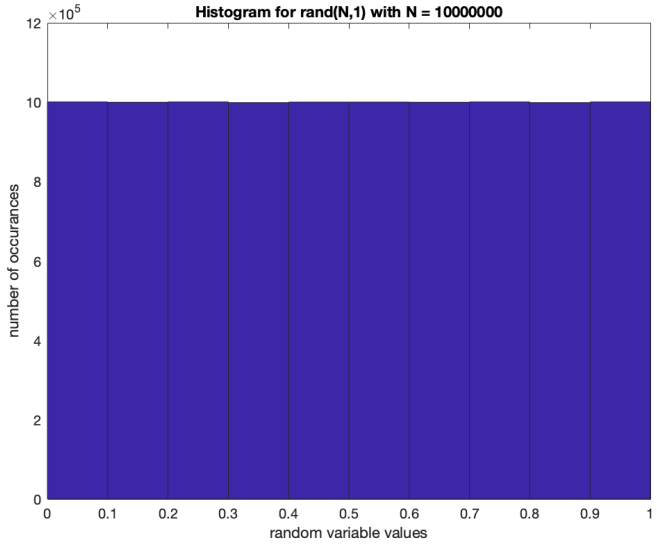


Figure 4: Generated histogram using the rand(N,1) function with N = 10,000,000.

Observing Fig 1-4, it is clear that as **N** increases, the shape of the histogram better approximates that of a uniform distribution between 0 and 1. For a very small **N**, the histogram appears almost random, although it is not, and with a very large **N**, the shape almost perfectly reflects the expected distribution. This is exactly what was anticipated.

2 Exponential Random Variable

Extending from above, the previously generated uniformly distributed samples are transformed into new random samples with an exponential distribution. This means that **X**, a uniformly distributed random variable between 0 and 1, is inputted into a function, $\mathbf{g(X)}$, resulting in **Y**, which must have an exponential distribution.

We solve for $\mathbf{g(X)}$ below using the Transformation method [1] (page 172): Starting with the provided pdf for an exponential distribution:

$$f_Y(y) = e^{-y}$$

We solve for the cdf by integration from 0 to y:

$$F_Y(y) = \int_0^y e^{-y}\, dy = 1 - e^{-y}$$

According to the Transformation method, our desired random distribution can be derived by inverting the cdf of the desired distribution with the cdf of the current random variable as input to that distribution. Noting that $\mathbf{X}$, a uniform distribution from 0 to 1, has the cdf, U, defined from 0 to 1:

$$Y = 1 - e^{-U}$$

$$Y - 1 = -e^{-U}$$

$$\ln(1 - U) = -Y$$

$$Y = -\ln(1 - U)$$

therefore, since $X = U$ here, $\mathbf{g(X)} = -\ln(1 - X)$, and $\mathbf{g(X)}$ is the transform that can be used to generate a new variable $\mathbf{Y}$ with an exponential distribution, from the original random variable $\mathbf{X}$ with a uniform distribution from 0 to 1.

Repeating the above procedure for

$$f_Y(y) = \lambda e^{-\lambda y}$$

the cdf is

$$F_Y(y) = \int_0^y \lambda e^{-\lambda y}\, dy = 1 - e^{-\lambda y}$$

and again, according to the Transformation method:

$$Y = 1 - e^{-\lambda U}$$

$$Y - 1 = -e^{-\lambda u U}$$

$$\ln(1 - U) = -\lambda U$$

$$Y = -\frac{1}{\lambda}\ln(1-U)$$

Therefore, since $X = U$ here, $\mathbf{g(X)} = -\frac{1}{\lambda}\ln(1-X)$. Note that the first $\mathbf{g(X)}$ derived is identical to this $\mathbf{g(X)}$ if $\lambda = 1$ in the first derivation, which it does.

Using the above expression, samples of $\mathbf{X}$ were converted into samples of $\mathbf{Y}$ and histograms of their results were made.

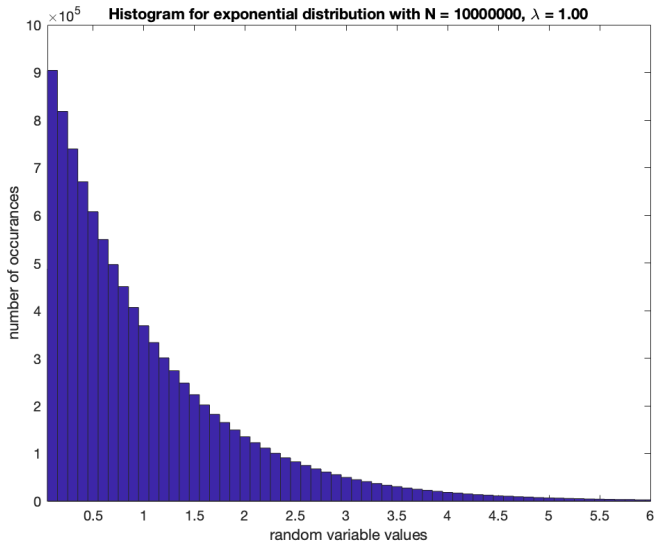


Figure 5: Generated histogram of exponential random variable samples with N = 100,000, $\lambda = 1$

It is clear in the above, Fig. 5, that the histogram does resemble the shape for an exponential distribution. Also note that for Fig. 5, $\lambda = 1$.

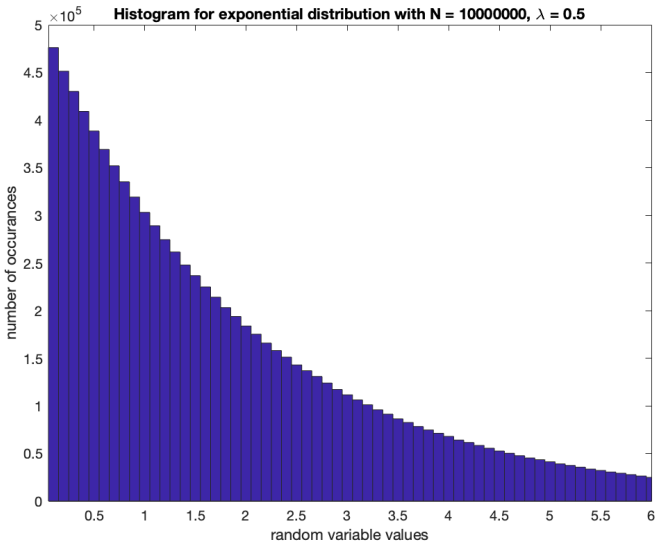


Figure 6: Generated histogram of exponential random variable samples with N = 100,000, $\lambda = .5$

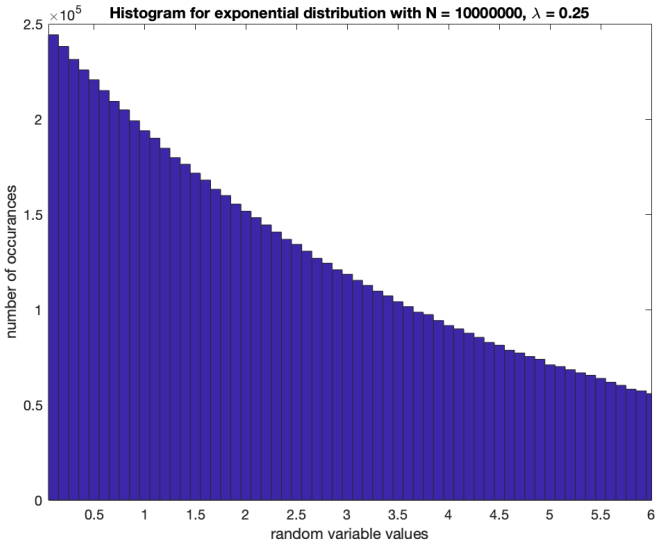


Figure 7: Generated histogram of exponential random variable samples with N = 100,000, $\lambda = .25$

Varying λ, the expected effect is observed. As λ decreases, the rate at which the distribution decays is slower; the inverse is also true. This is obvious when Fig. 5, 6, and 7 are compared. Fig. 5 has the steepest decay and the largest λ, while Fig. 7 has the flattest decay and the smallest λ.

3 Random Samples for a Gaussian Random Variable

The next experiment involves generating a random variable, **Z**, with a Gaussian distribution.

Using the built-in **norminv** function in Matlab, the uniformly distributed samples from above are transformed into Gaussian samples with zero mean and unit variance:

```
Z = norminv(X);
```

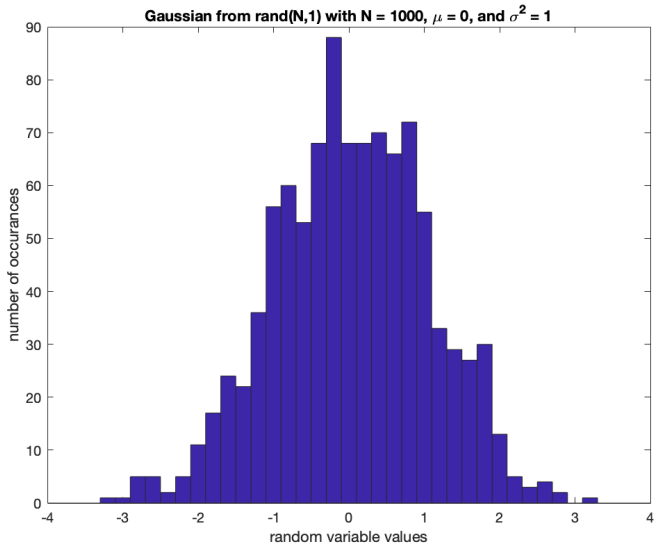


Figure 8: Generated Gaussian distribution samples with N = 1,000

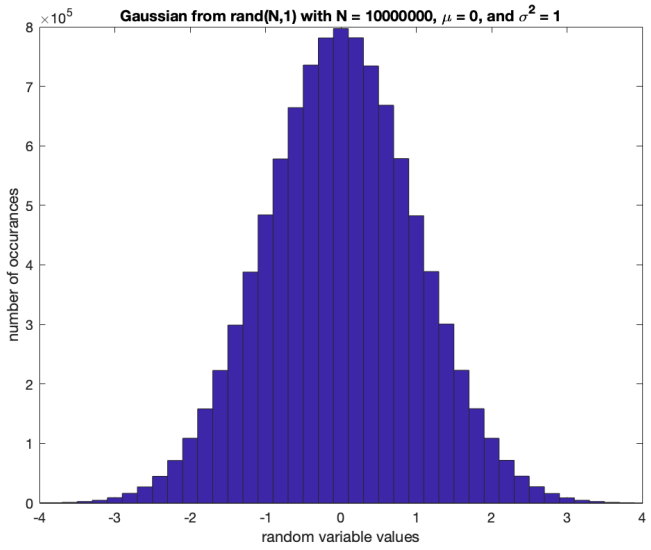


Figure 9: Generated Gaussian distribution samples with N = 10,000,000

Histograms of the **Z** samples can be seen in Fig. 8 and 9. It is clear that as **N**, the number of samples of the original uniform distribution, increases, our Gaussian approximation becomes tighter to a true Gaussian curve.

For the above, Matlab handles the math for us, but it is important to understand the underlying approach. The function **norminv** returns the inverse of the standard normal cumulative distribution function (cdf), evaluated at the probability values in p. Variable p is of course defined from 0 to 1, the bounds of probability. There are a variety of transform methods and ways in which **norminv** could be implemented and optimized; one such method is the Transformation method that was followed above in the previous section. Knowing the cdf for the input random variable and the desired distribution, a transformation can be derived. This method is convenient as it is known that there is no closed-form expression for a Gaussian pdf; typically approximations are required. In this case, as stated above, the cdf for a uniformly distributed random variable bounded between 0 and 1 is:

$$F_U(u) = \begin{cases} u & 0 \leq u \leq 1 \\ 0 & otherwise \end{cases}$$

and for a zero mean unit variance Gaussian random variable:

$$F_Z(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}$$

From here, a Transformation expression can be derived, and this is likely the methodology that Matlab uses. This is supported by looking into the Matlab function **norminv**; it takes an input variable and then performs linear transformations using a with that input variable and a function called **erfcinv**. The **erfcinv** function is a function for the inverse of the complementary error function. This process sounds awfully familiar to the previous Transformation method in this report, especially if the reader recalls that the error function is also known as a function describing the cdf of a Gaussian distribution with zero mean and $\frac{1}{2}$ variance.

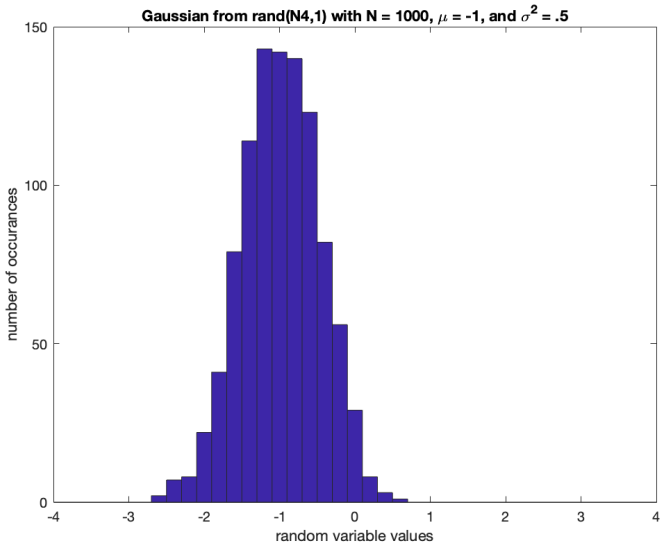


Figure 10: Generated Gaussian distribution samples with N = 1,000, $\mu = -1$, and $\sigma^2 = .5$

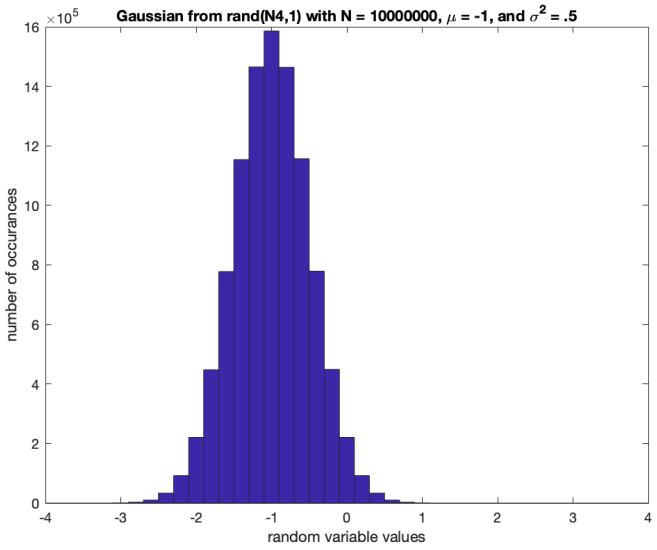


Figure 11: Generated Gaussian distribution samples with N = 10,000,000, $\mu = -1$, and $\sigma^2 = .5$

Continuing with this experiment, using **norminv**, Gaussian distributions were generated with different means (mu) and variances (var) for different length arrays of uniformly distributed random variables:

```
Z = norminv(X, mu, var);
```

Two such examples can be seen in Fig. 10 and 11. As expected, the shifted mean is evident in each Figure, as well as the decreased variance.

As an exercise, the actual means and variances of samples were computed and compared to the "true" means and variances. The results were:

```
Mean for N = 10000000, true mean = -1, true var = 1: -0.999467
Var for N = 10000000, true mean = -1, true var = 1: 1.000411
Mean for N = 10000000, true mean = 1, true var = 1: 1.000533
Var for N = 10000000, true mean = 1, true var = 1: 1.000411
Mean for N = 10000000, true mean = -1, true var = .5: -0.999734
Var for N = 10000000, true mean = -1, true var = .5: 0.250103
Mean for N = 10000000, true mean = 1, true var = .5: 1.000266
```

```
Var for N = 10000000, true mean = 1, true var = .5: 0.250103
Mean for N = 1000, true mean = -1, true var = 1: -0.963631
Var for N = 1000, true mean = -1, true var = 1: 1.068855
Mean for N = 1000, true mean = 1, true var = 1: 1.036369
Var for N = 1000, true mean = 1, true var = 1: 1.068855
Mean for N = 1000, true mean = -1, true var = .5: -0.981815
Var for N = 1000, true mean = -1, true var = .5: 0.267214
Mean for N = 1000, true mean = 1, true var = .5: 1.018185
Var for N = 1000, true mean = 1, true var = .5: 0.267214
```

Upon inspection, one will observe that the actual means and variances more closely resembled the true means for N = 10000000 compared to N = 1000. This makes sense when Fig. 8 9 10 and 11 are analyzed, but also makes sense intuitively, if the core ideas behind the laws of large numbers are recalled.

4 Independent and Identically Distributed Random Variables and their Sum

For our next experiment, it is desired to generate **M** random variables with **N** random samples each.

To do this, the same **rand** function from earlier is used, except that we specify both **N** and **M**:

```
X = rand(N,M);
```

This leaves a matrix of random samples, where each column is a collection of random samples, referred to as its own random variable.

As a demonstration of the central limit theorem, summing up these random variables is desired.

This is achieved through the **sum** function in Matlab:

```
S_x = sum(Y2,2);
```

Here, the **2** in **sum(Y2,2)** indicates summing in the row direction, as required, since we are summing random variables by their random samples **N** times.

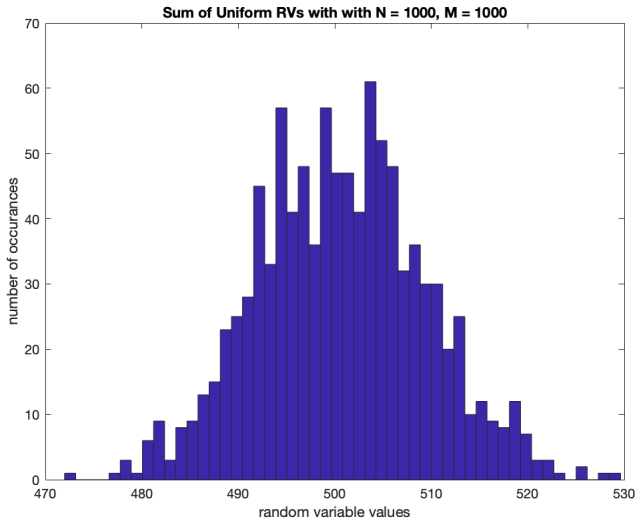


Figure 12: Generated Gaussian distribution from summing M = 1,000 uniformly distributed random variables and N = 1,000 samples from each.

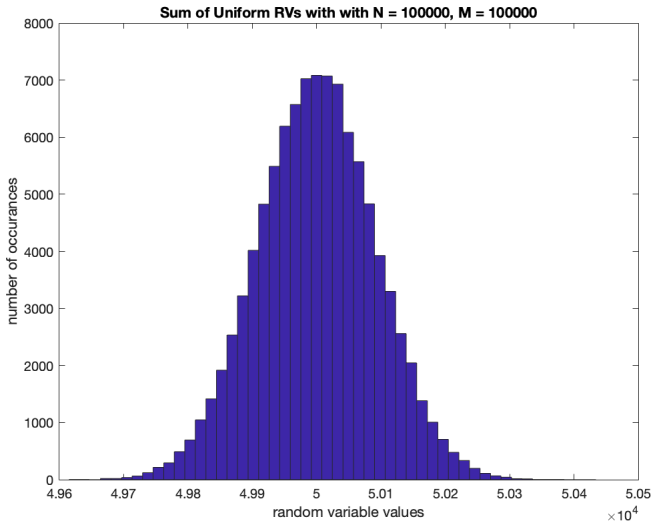


Figure 13: Generated Gaussian distribution from summing M = 100,000 uniformly distributed random variables and N = 100,000 samples from each.

Two examples for various summations of $\mathbf{N}$ random samples over $\mathbf{M}$ random variables can be seen in Fig. 12 and 13. As $\mathbf{N}$ and $\mathbf{M}$ increase, the curve converges onto the familiar Gaussian shape. Given knowledge of the central limit theorem, it it known that Gaussian approximations of the sum of iid random variables with finite means and finite variances improve as the number of random samples and random variables increases. This is the case here, evident by Fig. 13 better reflecting a Gaussian curve compared to Fig. 12, although both of them have the general Gaussian shape. This reinforces the central limit theorem.

Note that for very large $\mathbf{N}$ and $\mathbf{M}$, Matlab does not allow for this generation and summation to occur in one step, as too much memory is required. Therefore, each random variable was generated and summed recursively to generate Fig. 13. The code to achieve this can be found in the attached Matlab code at the end of this document.

The above process was repeated but for exponential random variables. It is expected that similar Gaussian curves with be generated, in accordance with the proclamations of the central limit theorem.

The transformation derived in the previous part [$\mathbf{g(X)} = -\ln(1 - X)$] was used to convert the set of $\mathbf{M}$ uniformly distributed random variables with $\mathbf{N}$ samples each to $\mathbf{M}$ exponentially distributed random variables with $\mathbf{N}$ samples each, as done prior.

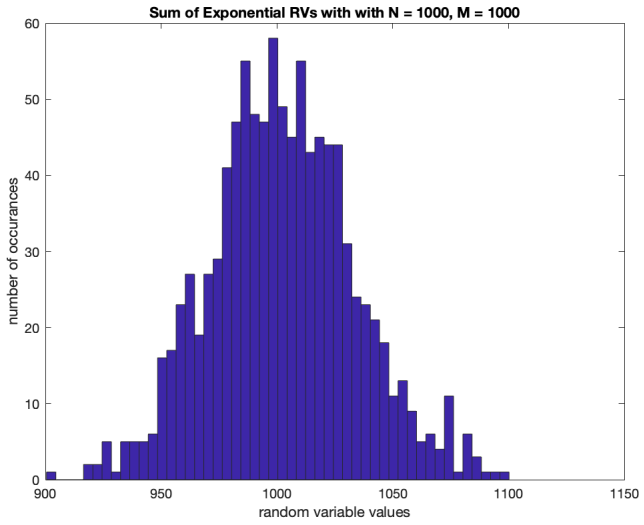


Figure 14: Generated Gaussian distribution from summing M = 1,000 exponentially distributed random variables and N = 1,000 samples from each.

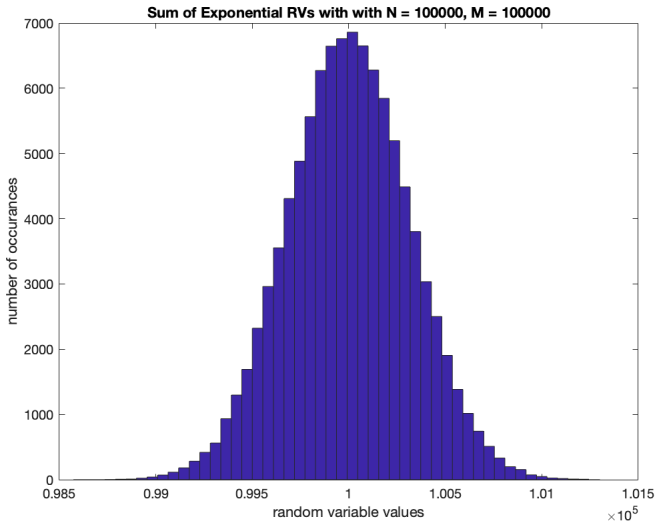


Figure 15: Generated Gaussian distribution from summing M = 1,000 exponentially distributed random variables and N = 1,000 samples from each.

Again, using the **sum** function in Matlab, these random variables were summed and their distributions plotted and displayed, as shown in Fig. 14 and 15. As expected, the distribution is seemingly Gaussian, and tightens to a Gaussian shape as **N** and **M** increase.

5 Correlated Gaussian Random Variables using Linear Transformations

The objective for this experiment is to generate **M** correlated Gaussian random variables from a set of **M** independent Gaussian random variables through a linear transformation.

Unlike previously for generating linear independent Gaussian random variables, here, a linear transformation approach is used.

According to the textbook, for this method [1] (page 270), it is required to:

1. Generate U_1 and U_2, two independent random variables uniformly distributed in the unit interval.

This is easy and only requires use of the **rand** function from earlier.

2. Let $R^2 = -2\log(U_1)$ and $\Theta = 2\pi U_2$

3. Let $X_1 = Rcos(\Theta) = \sqrt{-2\log(U_1)}cos(2\pi U_2)$ and
$X_2 = Rcos(\Theta) = \sqrt{-2\log(U_1)}sin(2\pi U_2)$

This results in two independent, zero-mean, unit-variance Gaussian random variables.

For the presented desired outcome, it is desired to start the problem with **M** independent Gaussian random variables, so the above steps for 2 random variables must somehow be extended to **M** random variables. Since the random variables that are being considered are iid random variables, one can simply replicate the process **M/2** times to obtain **M** random variables.

Matlab code similar to the following was used to generate **M** independent Gaussian random variables with **N** samples each, represented by **Z**, using the linear transformation approach:

```
Z = zeros(N, M);
for i = 1:M/2
    U_1 = rand(N,1);
    U_2 = rand(N,1);

    Z(:,(i*2)-1) = sqrt(-2*log(U_1)).*cos(2*pi*U_2);
    Z1_2(:,(i*2)) = sqrt(-2*log(U_1)).*sin(2*pi*U_2);
end
```

The attached Matlab code is slightly different to account for various values of **M** and **N**, as well as other parameter changes.

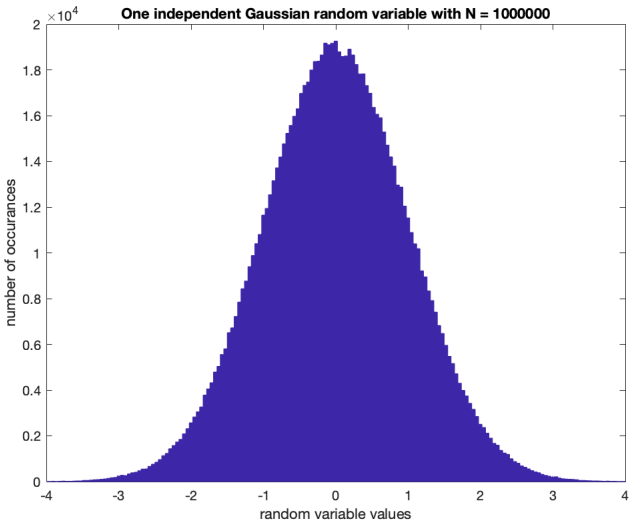


Figure 16: Generated Gaussian distribution for a single random variable from the linear transformation approach with N = 1,000,000 samples.

To verify this method, a single random variable with N = 1,000,000 samples was plotted in a histogram, evident in Fig. 16. The expected Gaussian curve was apparent, and this established confidence in the above linear transformation approach for its validity.

In order to generate a new vector of correlated Gaussian random variables, a linear transformation was used: $\mathbf{W} = A\mathbf{Z}$. A is an $\mathbf{M} \times \mathbf{M}$ square matrix, derived from a $\mathbf{M} \times \mathbf{M}$ covariance matrix. $\mathbf{W}$ is the new set of correlated

Gaussian random variables, derived from the old set of uncorrelated Gaussian random variables, **Z**.

It was given that the covariance matrix be defined by:

$$C = \begin{bmatrix} 1 & \rho & \cdots & \rho^{M-1} \\ \rho & 1 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \rho^{M-1} & \cdots & \cdots & 1 \end{bmatrix}$$

which as a Matlab function is something like:

```
function C = get_covariance_matrix(rho,M)
    C = zeros(M, M);
    for i = 1:M
        for j = 1:M
            power = i - j;
            C(i,j) = rho^(abs(power));
        end
    end
end
```

By computing the eigenvalues and eigenvectors of C in Matlab, it is rather straight forward to solve for the A matrix, especially since the covariance matrix of **Z** is an identity matrix, as it is uncorrelated [1] (page 269-270). This is coded in Matlab by:

```
C = get_covariance_matrix(rho, M);
[P,D] = eig(C);
A = P * sqrt(D);
```

Knowing A, only Matrix multiplication is require to solve for **W**. Matrix A must be multiplied by **Z** random variables for each set of **M** samples. In Matlab, this can be written as:

```
W = zeros(M);
for i = 1:N
    W(:,i) = A1*(Z(i,:)');
end
```

Finally, the built **cov** function is used in Matlab, which returns the covariance of a matrix whose columns represent random variables and whose rows represent observations, as had here, to verify the covariances for $\mathbf{Z}$ and $\mathbf{W}$. Variable $\mathbf{W}$ is transposed to match the Matlab input requirements of rows and columns for random variables and observations:

```
C_w = cov(W')
C_z = cov(Z)
```

The above work outlined the methods in generating and verifying that $\mathbf{M}$ correlated Gaussian random variables were created. The details of the code and computations can be again found in the attached Matlab script at the end of this report.

Now, for the results of the above computations for different values of ρ. It is hoped that the computed covariance matrices resemble C.

For $\rho = 0$, $\mathbf{M} = 2$ and $\mathbf{N} = 100$, our computed covariance matrix for $\mathbf{W}$ was:

```
 0.8620   0.0960
-0.0960   0.9049
```

and for $\mathbf{Z}$:

```
 0.8620  -0.0960
-0.0960   0.9049
```

This shows basically no correlation, as expected when $\rho = 0$. In this experiment, reflecting the theory, the covariance matrices converge to an identity matrix for $\rho = 0$ as $\mathbf{N}$ increases, evident by the computed covariance matrices for $\rho = 0$, $\mathbf{M} = 2$ and $\mathbf{N} = 1{,}000{,}000$ for $\mathbf{W}$ being:

```
 0.9980  -0.0002
-0.0002   1.0002
```

and for $\mathbf{Z}$:

```
 0.9980  -0.0002
-0.0002   1.0002
```

This same concept was observed for the $\rho = 0$ case for larger $\mathbf{M}$ as well.

Next, the computed covariance matrices for the case of $\rho = .5$, $\mathbf{M} = 6$ and $\mathbf{N} = 100$ is shown. For $\mathbf{W}$:

```
 1.0319    0.4528    0.2962    0.0507    0.0740    0.0695
 0.4528    0.9109    0.5362    0.2217    0.0848    0.1035
 0.2962    0.5362    1.1861    0.6528    0.2555    0.3365
 0.0507    0.2217    0.6528    1.1323    0.4916    0.3746
 0.0740    0.0848    0.2555    0.4916    0.6822    0.3486
 0.0695    0.1035    0.3365    0.3746    0.3486    0.9449
```

and for $\mathbf{Z}$:

```
 0.9052    0.0933    0.0316    0.0977   -0.1468   -0.0149
 0.0933    1.0266   -0.0890    0.0061    0.1054    0.2781
 0.0316   -0.0890    1.0419    0.0477    0.0062   -0.1318
 0.0977    0.0061    0.0477    0.9595   -0.1545   -0.1066
-0.1468    0.1054    0.0062   -0.1545    0.8630   -0.0175
-0.0149    0.2781   -0.1318   -0.1066   -0.0175    1.0494
```

The above is not very precise, but as expected, using the same parameters but increasing the number of samples ($\mathbf{N}$ = 1,000,000) results in computed covariance matrices that much better fit to what is expected. For $\mathbf{W}$:

```
 1.0016    0.5000    0.2503    0.1246    0.0628    0.0301
 0.5000    0.9996    0.5007    0.2495    0.1245    0.0619
 0.2503    0.5007    0.9987    0.4995    0.2502    0.1244
 0.1246    0.2495    0.4995    1.0004    0.4993    0.2489
 0.0628    0.1245    0.4995    1.0004    0.9983    0.4992
 0.0301    0.0619    0.1244    0.2489    0.4992    0.9982
```

and for $\mathbf{Z}$:

```
 0.9997   -0.0008    0.0003    0.0022   -0.0008    0.0004
-0.0008    0.9970    0.0020    0.0003    0.0002   -0.0002
 0.0003    0.0020    1.0011    0.0000   -0.0006    0.0006
 0.0022    0.0003    0.0000    0.9995   -0.0008   -0.0004
-0.0008    0.0002   -0.0006   -0.0008    1.0001   -0.0010
 0.0004   -0.0002    0.0006   -0.0004   -0.0010    0.9991
```

Lastly, for good measure, the computed covariance matrices for $\rho = .95$, $\mathbf{M} = 2$ and $\mathbf{N} = 1{,}000{,}000$ are shown. For $\mathbf{W}$:

```
 1.0002    0.9503
 0.9503    1.0001
```

and for $\mathbf{Z}$:

```
 0.9980   -0.0002
-0.0002    1.0002
```

precisely as expected for a large $\mathbf{N}$.

Note that for all cases of the computed covariance matrices of $\mathbf{Z}$, an identity matrix is expected. This is because $\mathbf{Z}$ is a set of uncorrelated Gaussian random variables with unit variance. Covariance matrices for $\mathbf{W}$ on the other hand reflect what is expected of C. Both expectations were found to be true in the experiment.

It is concluded that the computed covariance matrices agree with the theoretical ones overall, and this is especially true for a large $\mathbf{N}$.

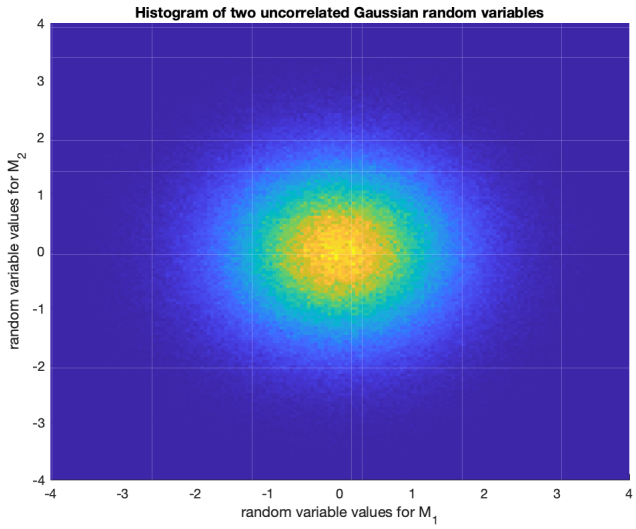


Figure 17: Generated histogram distribution of two uncorrelated Gaussian random variables (M = 2) for N = 1,000,000 samples each ($\rho = 0$).

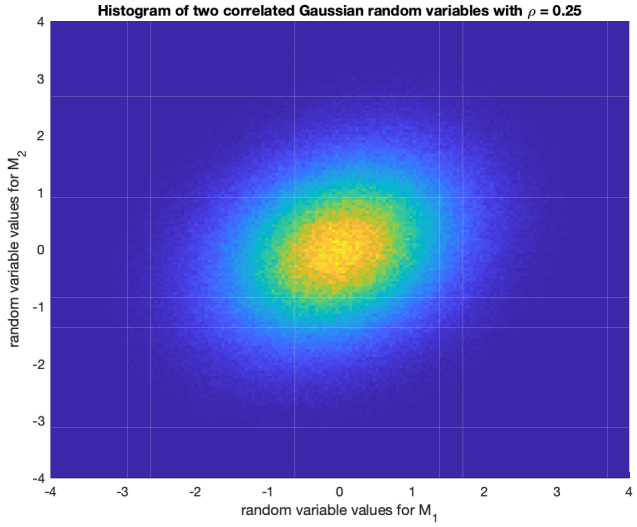


Figure 18: Generated histogram distribution of two correlated Gaussian random variables (M = 2) for N = 1,000,000 samples each and $\rho = .25$.

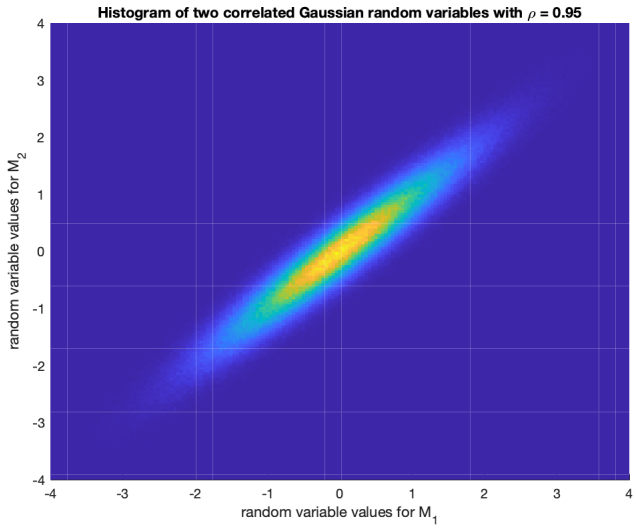


Figure 19: Generated histogram distribution of two correlated Gaussian random variables (M = 2) for N = 1,000,000 samples from each $\rho = .95$.

As an additional exercise, 3D histograms showing the distributions of two Gaussian random variables ($\mathbf{M} = 2$), with $\mathbf{N} = 1{,}000{,}000$, and various values of ρ, were generated, as seen in Fig. 17, 18, and 19. This was not a part of the original assignment description, but it was good to discover that the distributions for two correlated Gaussian random variables from the class notes were able to be replicated with the above works.

6 Conclusion

With this, this project is complete. Through this work, sums of random variables and correlated Gaussian processes were thoroughly explored, as well as some other core probabilistic concepts, such as transformation methods, the laws of large numbers, and the central limit theorem.

All Matlab code used in generating the included figures, code, and data can be found, in published form, following the references page.

References

[1] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering*, 2nd ed. Pearson Education Inc., 1998, pp. 172 267–271.

Contents

- ECE 863 Final Project
- 1)
- a)
- b)
- c)
- 2)
- a)
- b)
- c)
- 3
- a)
- b)
- c)
- 4)
- a)
- b)
- 5)
- a)
- b) and c)

ECE 863 Final Project

```
clc
close all
clear all
```

1)

a)

```
N1 = 10;
X1 = rand(N1,1);

N2 = 1000;
X2 = rand(N2,1);

N3 = 100000;
X3 = rand(N3,1);

N4 = 10000000;
X4 = rand(N4,1);
```

b)

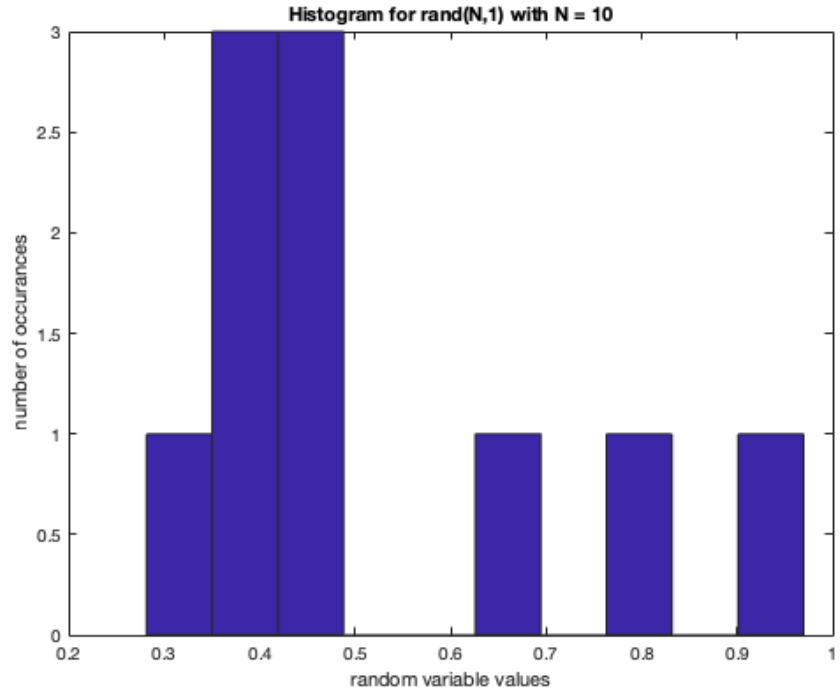
the pdf of this is a normaliy distributed pdf between 0 and 1

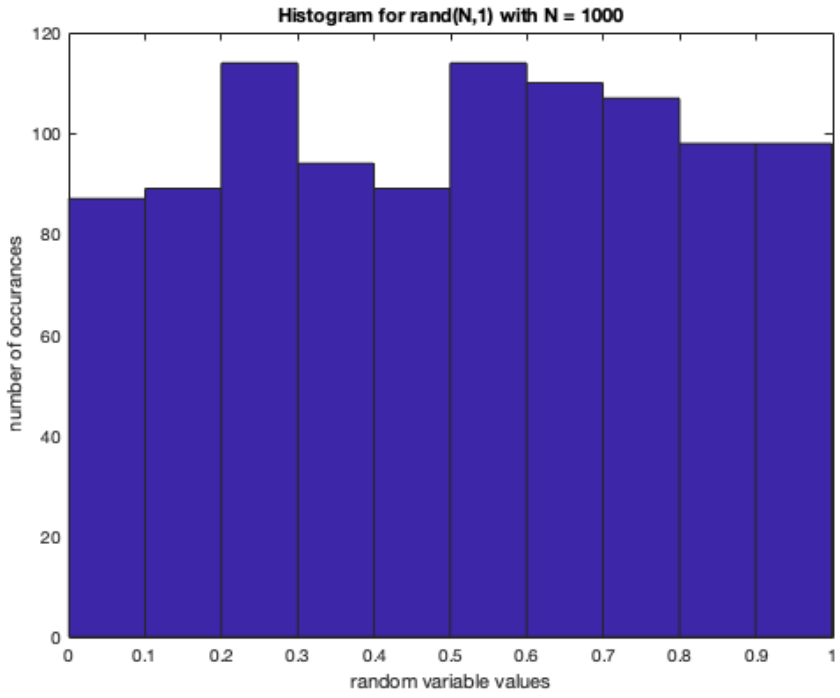
c)

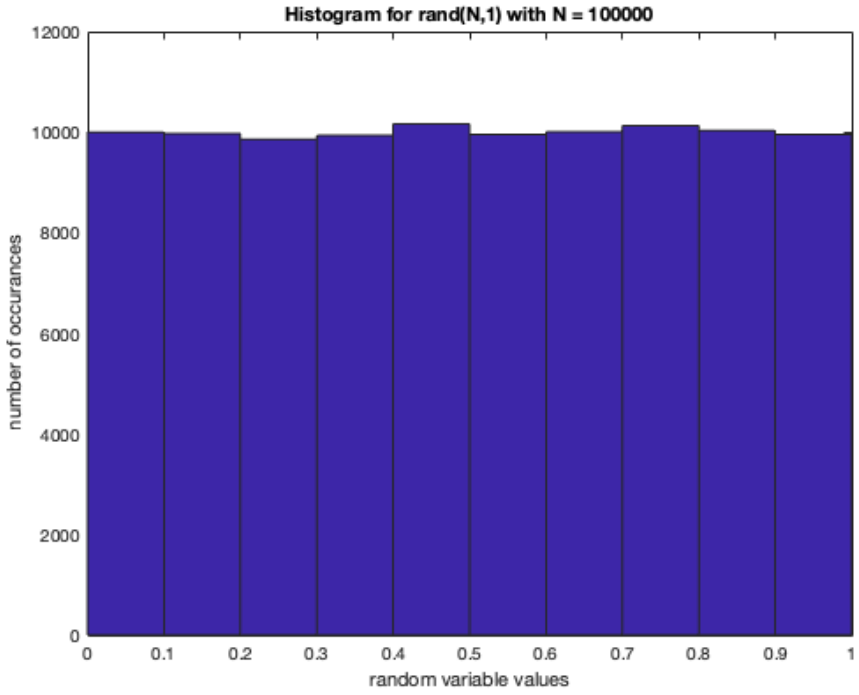
plots of various histograms

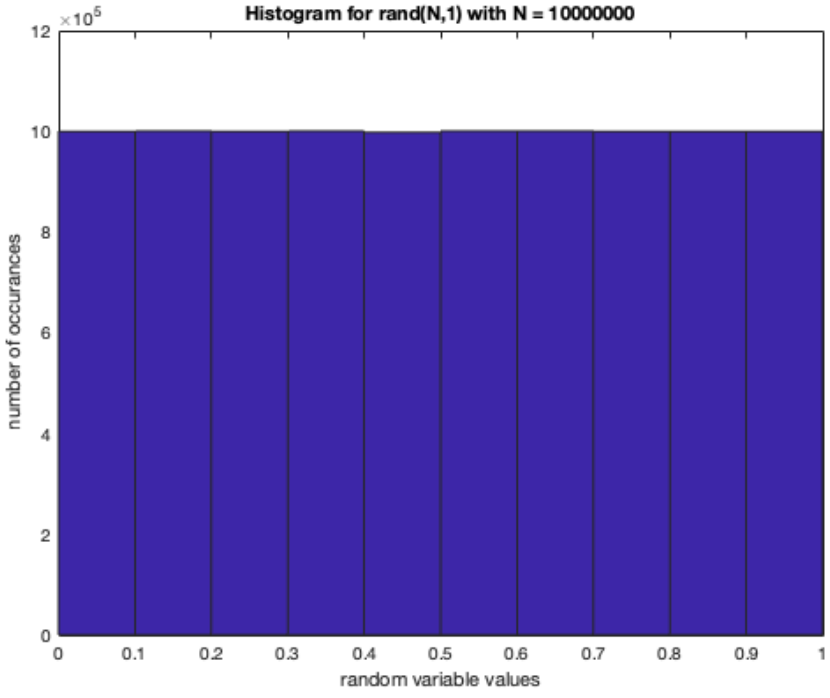

```
figure()
hist(X1)
title(sprintf("Histogram for rand(N,1) with N = " + N1))
```

```
xlabel("random variable values")
ylabel("number of occurances")

figure()
hist(X2)
title(sprintf("Histogram for rand(N,1) with N = " + N2))
xlabel("random variable values")
ylabel("number of occurances")

figure()
hist(X3)
title(sprintf("Histogram for rand(N,1) with N = " + N3))
xlabel("random variable values")
ylabel("number of occurances")

figure()
hist(X4)
title(sprintf("Histogram for rand(N,1) with N = " + N4))
xlabel("random variable values")
ylabel("number of occurances")

% we see that as N increases, the shape converges to what we expect for a
% normal distribution

% histogram(X,10,'Normalization','probability') - this normalizes histogram
```


Histogram for rand(N,1) with N = 10
number of occurances
random variable values
0
0.5
1
1.5
2
2.5
3
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Histogram for rand(N,1) with N = 1000
number of occurances
random variable values
0
20
40
60
80
100
120
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1





2)

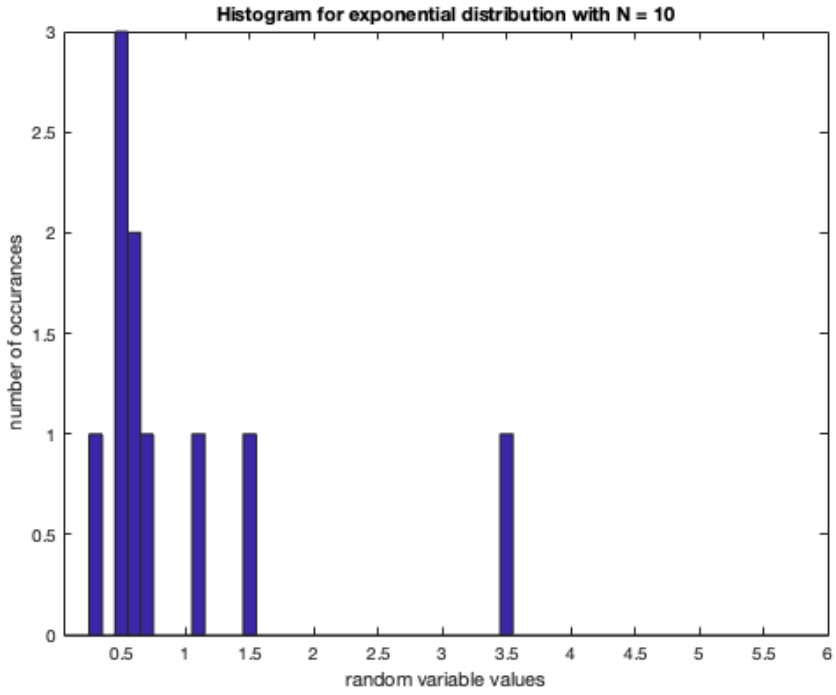
a)

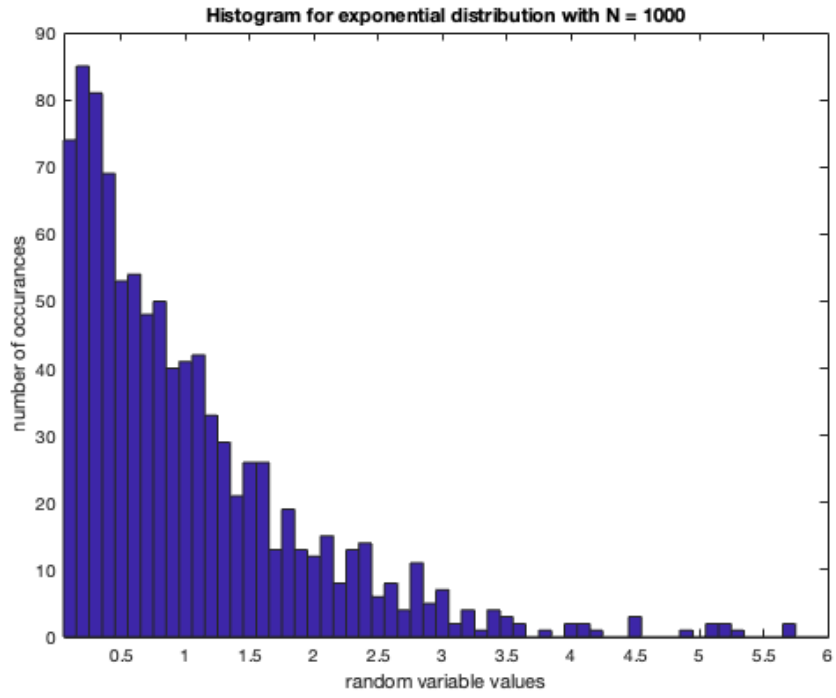
see page 172 of book for this transformation

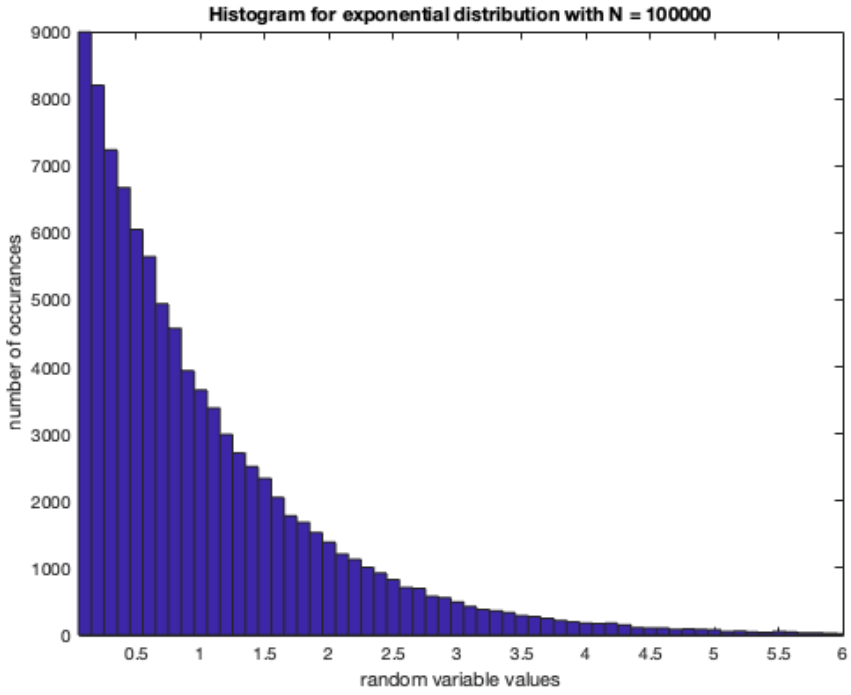
b)

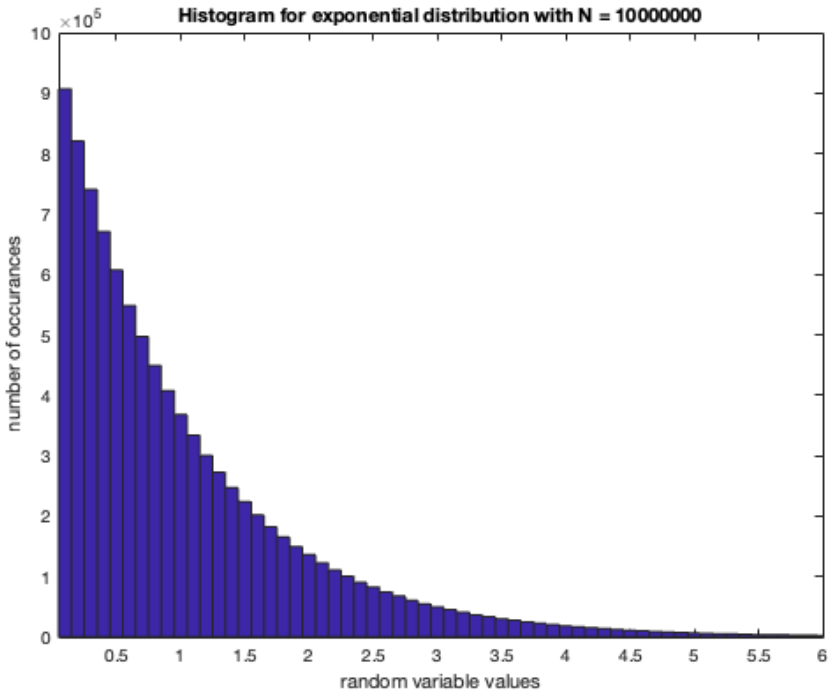
```
xbins = 0:.1:6.1;

Y1 = -log(1-X1); % see page 172 of book for this transformation
figure()
hist(Y1,xbins)
```

```
xlim([.05 6])
title(sprintf("Histogram for exponential distribution with N = " + N1))
xlabel("random variable values")
ylabel("number of occurances")

Y2 = -log(1-X2); % see page 172 of book for this transformation
figure()
hist(Y2,xbins)
xlim([.05 6])
title(sprintf("Histogram for exponential distribution with N = " + N2))
xlabel("random variable values")
ylabel("number of occurances")

Y3 = -log(1-X3); % see page 172 of book for this transformation
figure()
hist(Y3,xbins)
xlim([.05 6])
title(sprintf("Histogram for exponential distribution with N = " + N3))
xlabel("random variable values")
ylabel("number of occurances")

Y4 = -log(1-X4); % see page 172 of book for this transformation
figure()
hist(Y4,xbins)
xlim([.05 6])
title(sprintf("Histogram for exponential distribution with N = " + N4))
xlabel("random variable values")
ylabel("number of occurances")
```


Histogram for exponential distribution with N = 10
number of occurances
random variable values
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6


Histogram for exponential distribution with N = 1000
number of occurances
random variable values
0
10
20
30
40
50
60
70
80
90
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6

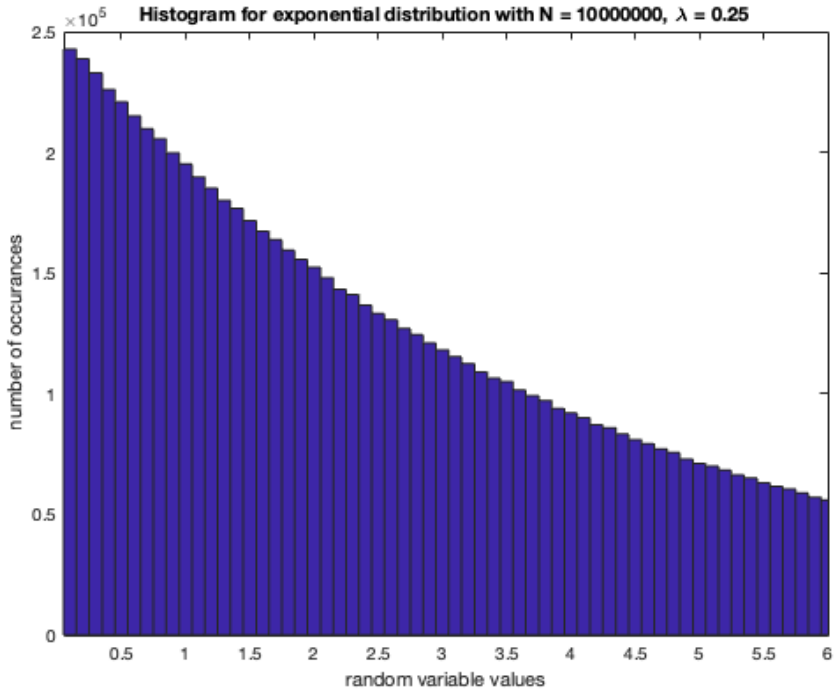


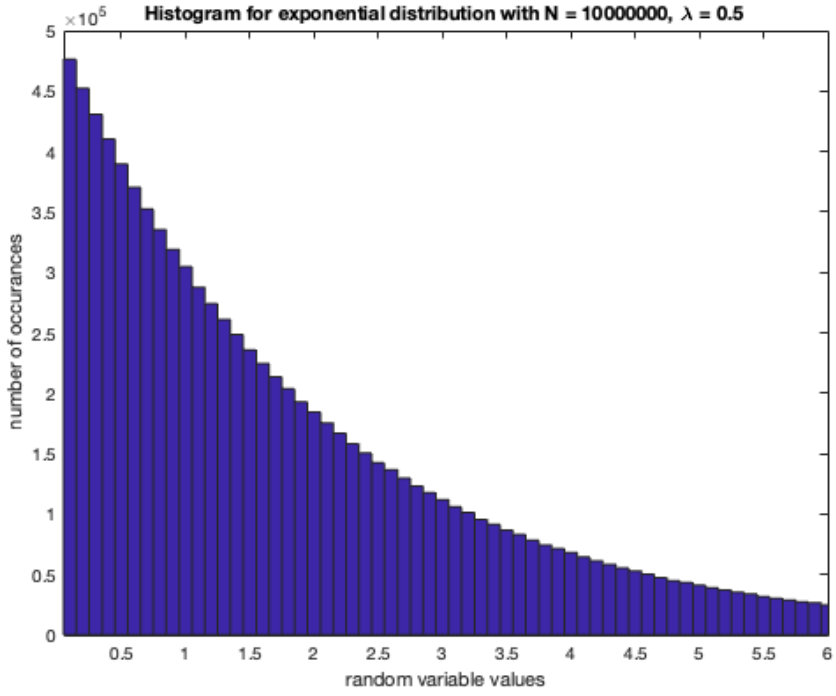


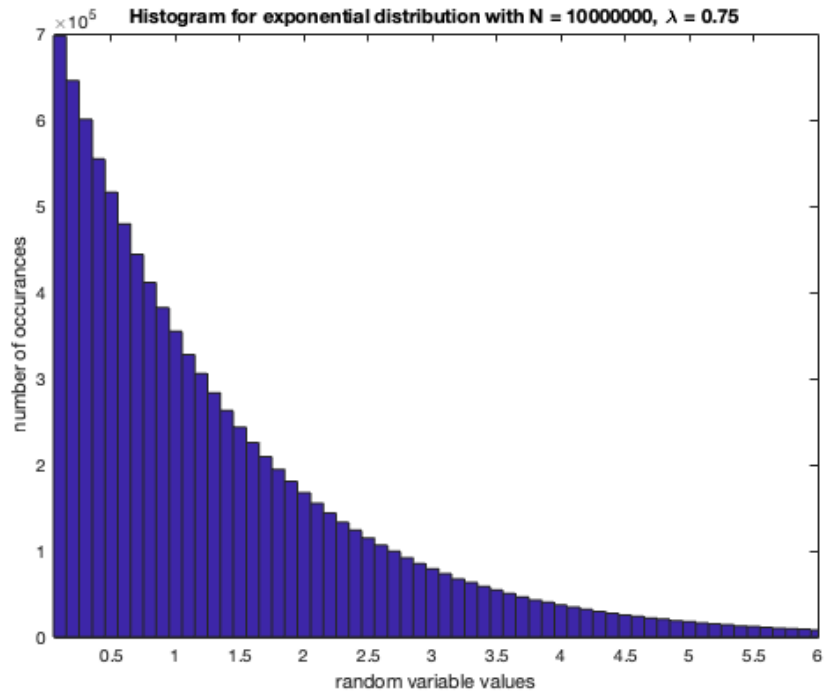
c)

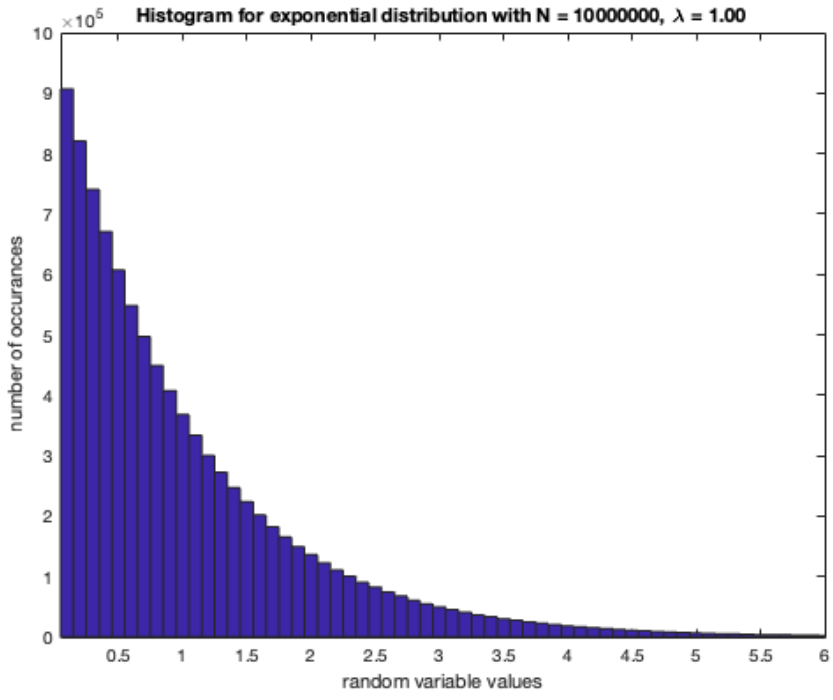
```
Y4 = -(1/.25)*log(1-X4); % see page 172 of book for this transformation
figure()
hist(Y4,xbins)
xlim([.05 6])
title(sprintf("Histogram for exponential distribution with N = " + N4) + ", \lambda = 0.25",'interpreter', 'tex')
xlabel("random variable values")
ylabel("number of occurances")

Y4 = -(1/.5)*log(1-X4); % see page 172 of book for this transformation
figure()
hist(Y4,xbins)
```

```
xlim([.05 6])
title(sprintf("Histogram for exponential distribution with N = " + N4) + ", \lambda = 0.5",'interpreter', 'tex')
xlabel("random variable values")
ylabel("number of occurances")

Y4 = -(1/.75)*log(1-X4); % see page 172 of book for this transformation
figure()
hist(Y4,xbins)
xlim([.05 6])
title(sprintf("Histogram for exponential distribution with N = " + N4) + ", \lambda = 0.75",'interpreter', 'tex')
xlabel("random variable values")
ylabel("number of occurances")

Y4 = -(1)*log(1-X4); % see page 172 of book for this transformation
figure()
hist(Y4,xbins)
xlim([.05 6])
title(sprintf("Histogram for exponential distribution with N = " + N4) + ", \lambda = 1.00",'interpreter', 'tex')
xlabel("random variable values")
ylabel("number of occurances")
```


Histogram for exponential distribution with N = 10000000, λ = 0.25
×10^5
number of occurances
random variable values
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6


Histogram for exponential distribution with N = 10000000, λ = 0.5
×10^5
number of occurances
random variable values
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6



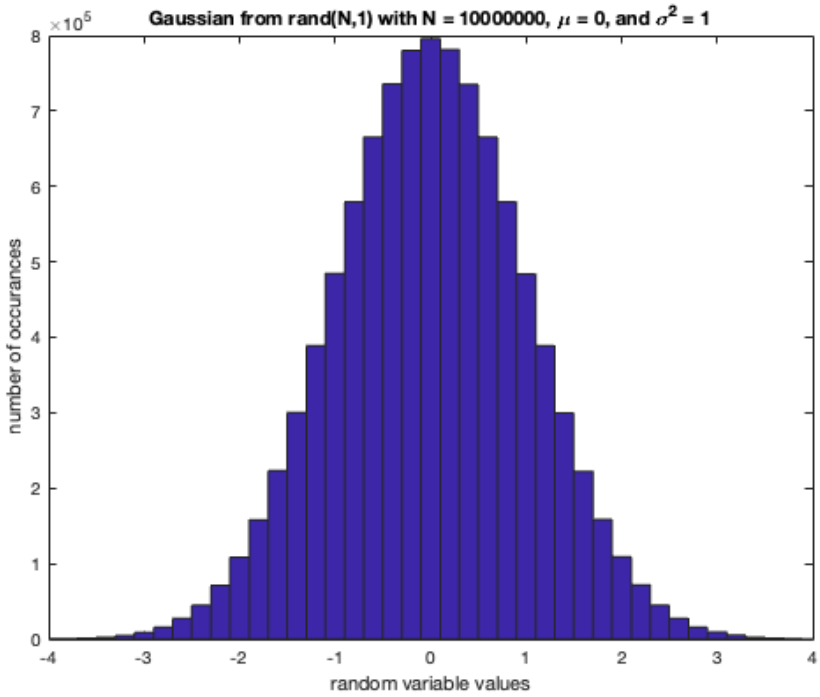


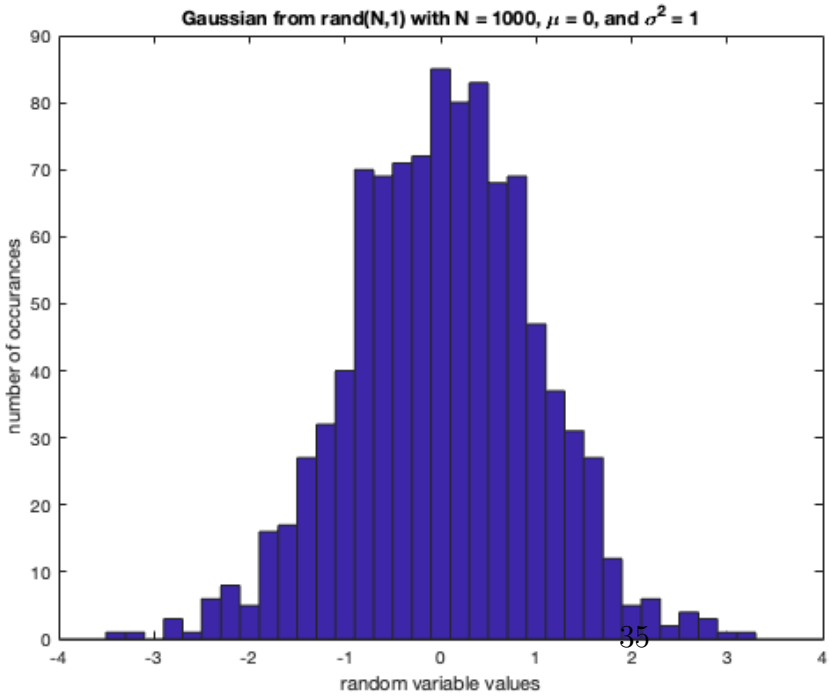
3

a)

```
xbins = -4:.2:4.2;

Z = norminv(X4);

figure()
hist(Z,xbins)
xlim([-4 4])
title(sprintf("Gaussian from rand(N,1) with N = " + N4) + ", \mu = 0, and \sigma^2 = 1",'interpreter', 'tex')
xlabel("random variable values")
```

```
ylabel("number of occurances")

Z = norminv(X2);

figure()
hist(Z,xbins)
xlim([-4 4])
title(sprintf("Gaussian from rand(N,1) with N = " + N2) + ", \mu = 0, and \sigma^2 = 1",'interpreter', 'tex')
xlabel("random variable values")
ylabel("number of occurances")
```





b)

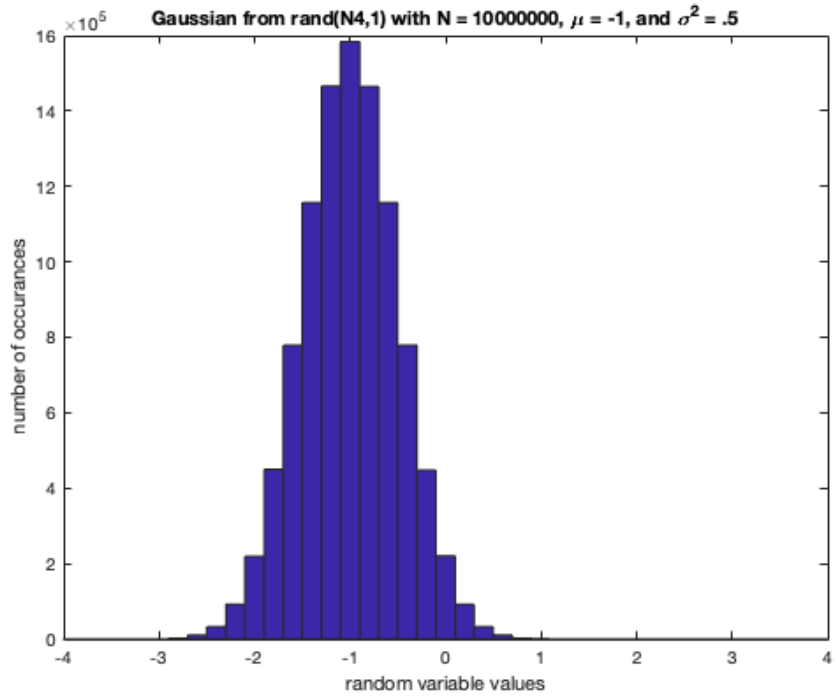
include derivation and explanation in report

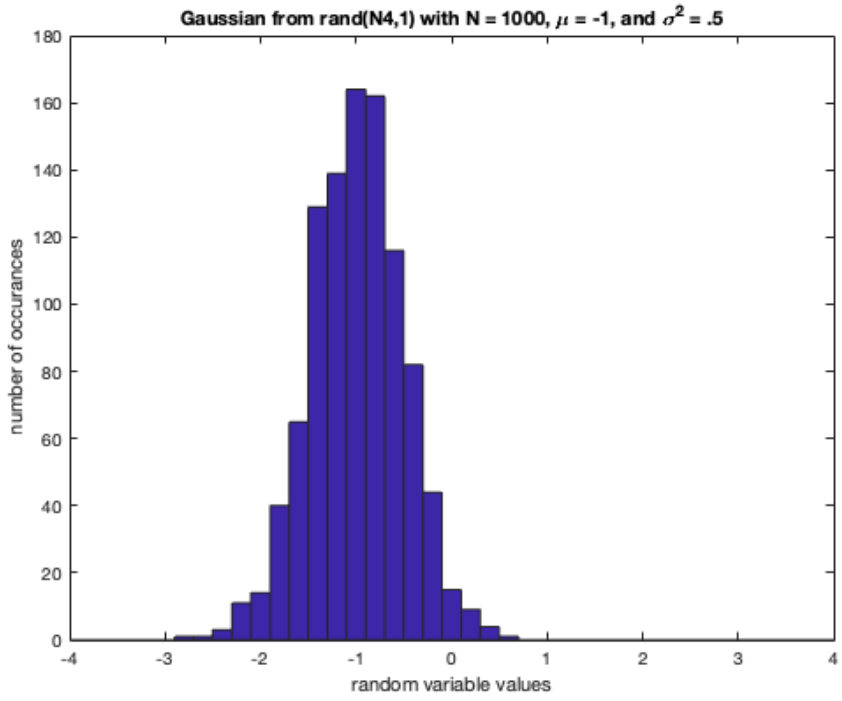
c)

```
Z = norminv(X4, -1, 1);
mu = mean(Z);
variance = var(Z);

fprintf("Mean for N = %d, true mean = -1, true var = 1: %f \n", N4, mu);
fprintf("Var for N = %d, true mean = -1, true var = 1: %f \n", N4, variance);

Z = norminv(X4, 1, 1);
mu = mean(Z);
variance = var(Z);

fprintf("Mean for N = %d, true mean = 1, true var = 1: %f \n", N4, mu);
fprintf("Var for N = %d, true mean = 1, true var = 1: %f \n", N4, variance);

Z = norminv(X4, -1, .5);
mu = mean(Z);
variance = var(Z);

fprintf("Mean for N = %d, true mean = -1, true var = .5: %f \n", N4, mu);
fprintf("Var for N = %d, true mean = -1, true var = .5: %f \n", N4, variance);

figure()
hist(Z,xbins)
xlim([-4 4])
title(sprintf("Gaussian from rand(N4,1) with N = " + N4) + ", \mu = -1, and \sigma^2 = .5",'interpreter', 'tex')
xlabel("random variable values")
ylabel("number of occurances")

Z = norminv(X4, 1, .5);
mu = mean(Z);
variance = var(Z);

fprintf("Mean for N = %d, true mean = 1, true var = .5: %f \n", N4, mu);
fprintf("Var for N = %d, true mean = 1, true var = .5: %f \n", N4, variance);

Z = norminv(X2, -1, 1);
mu = mean(Z);
variance = var(Z);

fprintf("Mean for N = %d, true mean = -1, true var = 1: %f \n", N2, mu);
fprintf("Var for N = %d, true mean = -1, true var = 1: %f \n", N2, variance);

Z = norminv(X2, 1, 1);
mu = mean(Z);
variance = var(Z);

fprintf("Mean for N = %d, true mean = 1, true var = 1: %f \n", N2, mu);
fprintf("Var for N = %d, true mean = 1, true var = 1: %f \n", N2, variance);

Z = norminv(X2, -1, .5);
mu = mean(Z);
variance = var(Z);

fprintf("Mean for N = %d, true mean = -1, true var = .5: %f \n", N2, mu);
fprintf("Var for N = %d, true mean = -1, true var = .5: %f \n", N2, variance);

figure()
hist(Z,xbins)
xlim([-4 4])
title(sprintf("Gaussian from rand(N4,1) with N = " + N2) + ", \mu = -1, and \sigma^2 = .5",'interpreter', 'tex')
xlabel("random variable values")
```

```
ylabel("number of occurances")

Z = norminv(X2, 1, .5);
mu = mean(Z);
variance = var(Z);

fprintf("Mean for N = %d, true mean = 1, true var = .5: %f \n", N2, mu);
fprintf("Var for N = %d, true mean = 1, true var = .5: %f \n", N2, variance);
```

```
Mean for N = 10000000, true mean = -1, true var = 1: -0.999986
Var for N = 10000000, true mean = -1, true var = 1: 0.999689
Mean for N = 10000000, true mean = 1, true var = 1: 1.000014
Var for N = 10000000, true mean = 1, true var = 1: 0.999689
Mean for N = 10000000, true mean = -1, true var = .5: -0.999993
Var for N = 10000000, true mean = -1, true var = .5: 0.249922
Mean for N = 10000000, true mean = 1, true var = .5: 1.000007
Var for N = 10000000, true mean = 1, true var = .5: 0.249922
Mean for N = 1000, true mean = -1, true var = 1: -0.974172
Var for N = 1000, true mean = -1, true var = 1: 0.943474
Mean for N = 1000, true mean = 1, true var = 1: 1.025828
Var for N = 1000, true mean = 1, true var = 1: 0.943474
Mean for N = 1000, true mean = -1, true var = .5: -0.987086
Var for N = 1000, true mean = -1, true var = .5: 0.235869
Mean for N = 1000, true mean = 1, true var = .5: 1.012914
Var for N = 1000, true mean = 1, true var = .5: 0.235869
```





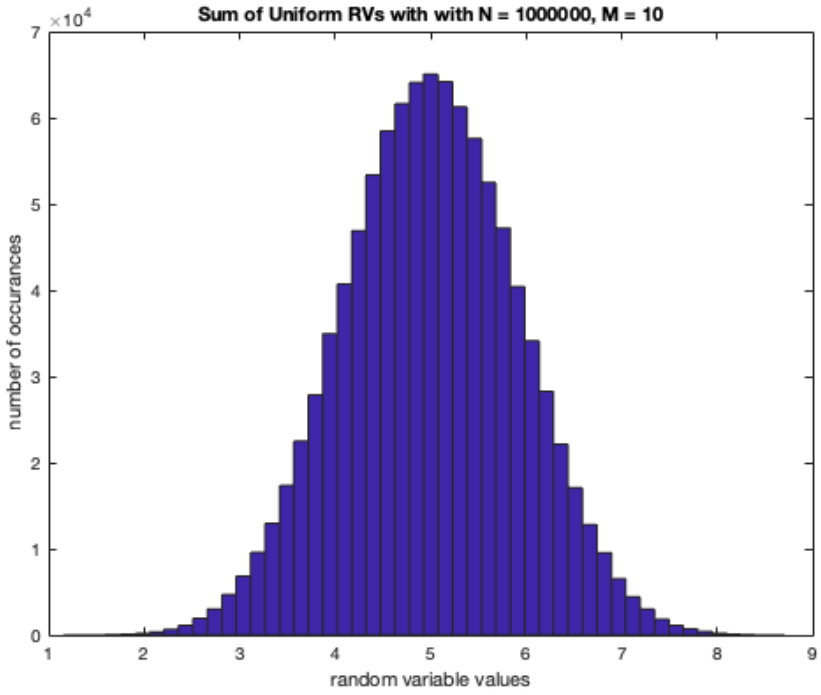
4)

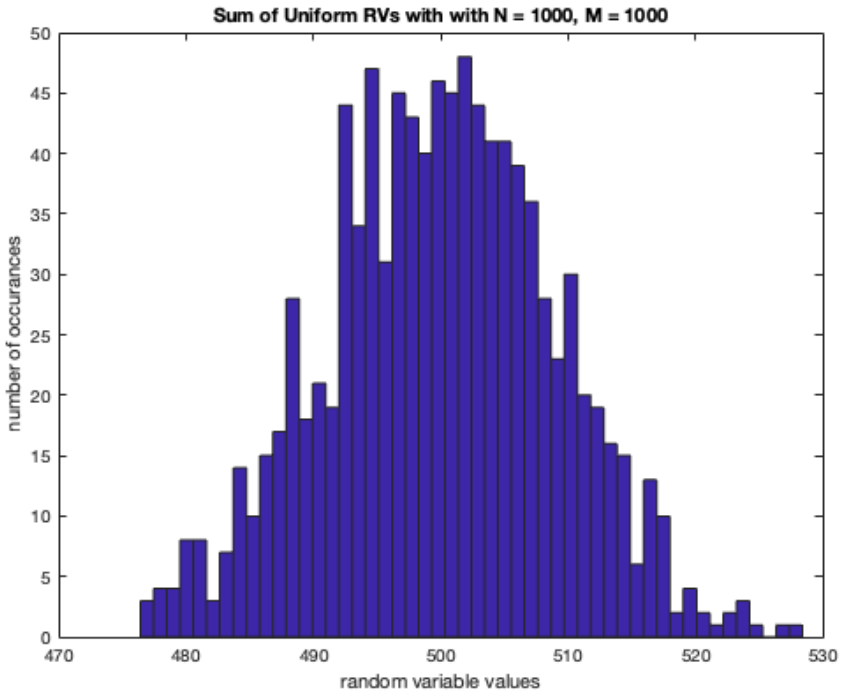
a)

start with new values for Ms and Ns so computer doesnt overflow:

```
M1 = 10;
M2 = 1000;
M3 = 10000;
M4 = 100000;

N1 = 1000000;
N2 = 1000;
```

```
N3 = 10000;
N4 = 100000;

X1 = rand(N1,M1);
Sx1 = sum(X1,2);
figure()
hist(Sx1,50)
title(sprintf("Sum of Uniform RVs with with N = " + N1 + ", M = " + M1))
xlabel("random variable values")
ylabel("number of occurances")

X2 = rand(N2,M2);
Sx2 = sum(X2,2);
figure()
hist(Sx2,50)
title(sprintf("Sum of Uniform RVs with with N = " + N2 + ", M = " + M2))
xlabel("random variable values")
ylabel("number of occurances")

% comment back in to run, but make take a while:

% for larger since Matlab runs out of memory - go one by one
Sx3 = zeros(M3, 1);
for i = 1:N3
    X3_i = rand(1,N3);
    summation = sum(X3_i,'all');
    Sx3(i) = summation;
end
figure()
hist(Sx3,50)
title(sprintf("Sum of Uniform RVs with with N = " + N3 + ", M = " + M3))
xlabel("random variable values")
ylabel("number of occurances")

Sx4 = zeros(M4, 1);
for i = 1:N4
    X4_i = rand(1,N4);
    summation = sum(X4_i,'all');
    Sx4(i) = summation;
end
figure()
hist(Sx4,50)
title(sprintf("Sum of Uniform RVs with with N = " + N4 + ", M = " + M4))
xlabel("random variable values")
ylabel("number of occurances")
```

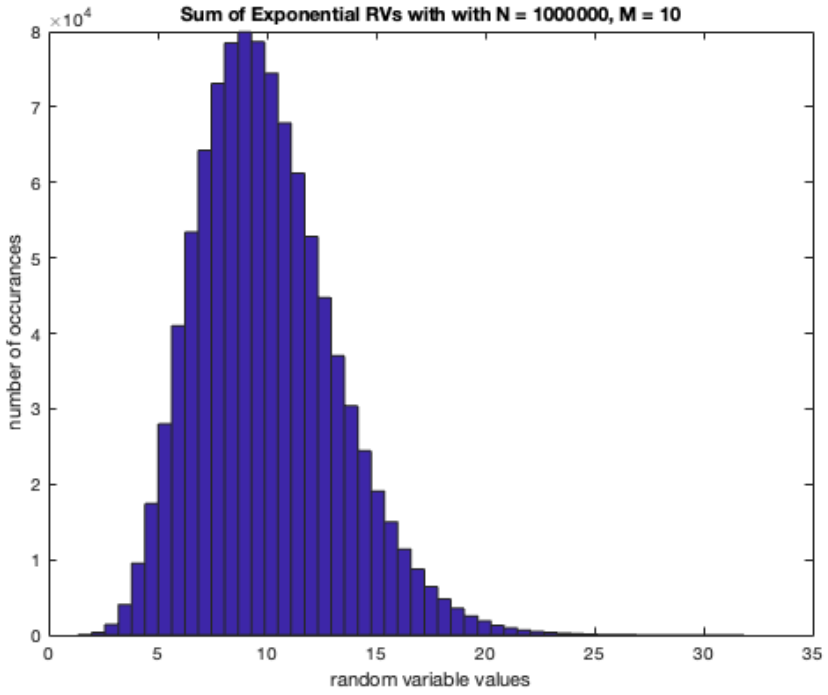


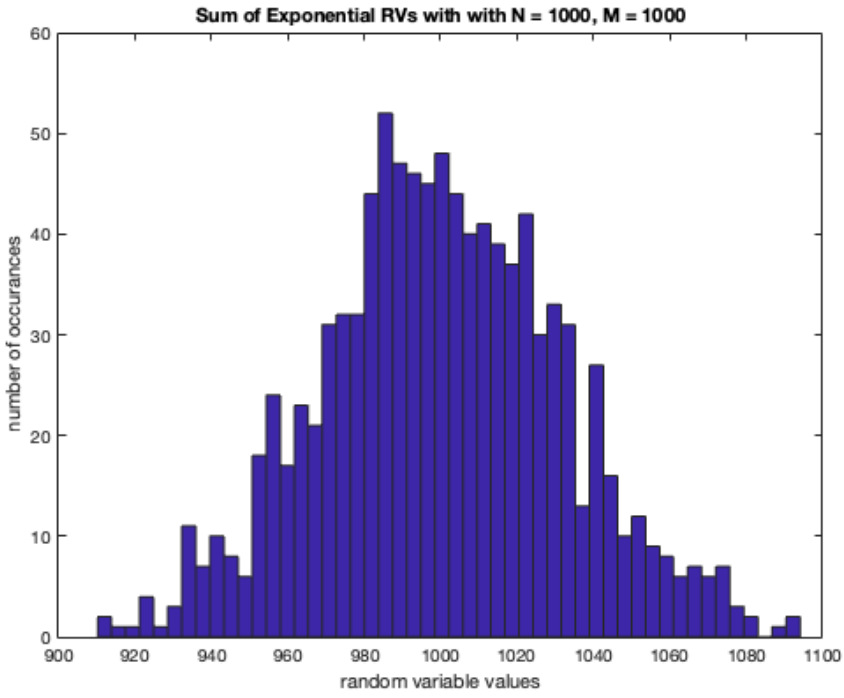


b)

```
Y1 = -log(1-X1);
Sx1 = sum(Y1,2);
figure()
hist(Sx1,50)
title(sprintf("Sum of Exponential RVs with with N = " + N1 + ", M = " + M1))
xlabel("random variable values")
ylabel("number of occurances")

Y2 = -log(1-X2);
Sx2 = sum(Y2,2);
figure()
```

```
hist(Sx2,50)
title(sprintf("Sum of Exponential RVs with with N = " + N2 + ", M = " + M2))
xlabel("random variable values")
ylabel("number of occurances")

% comment back in to run, but make take a while:

% for larger since Matlab runs out of memory - go one by one
Sx3 = zeros(M3, 1);
for i = 1:N3
    X3_i = rand(1,N3);
    Y3_i = -log(1-X3_i);
    summation = sum(Y3_i,'all');
    Sx3(i) = summation;
end
figure()
hist(Sx3,50)
title(sprintf("Sum of Exponential RVs with with N = " + N3 + ", M = " + M3))
xlabel("random variable values")
ylabel("number of occurances")

Sx4 = zeros(M4, 1);
for i = 1:N4
    X4_i = rand(1,N4);
    Y4_i = -log(1-X4_i);
    summation = sum(Y4_i,'all');
    Sx4(i) = summation;
end
figure()
hist(Sx4,50)
title(sprintf("Sum of Exponential RVs with with N = " + N4 + ", M = " + M4))
xlabel("random variable values")
ylabel("number of occurances")
```





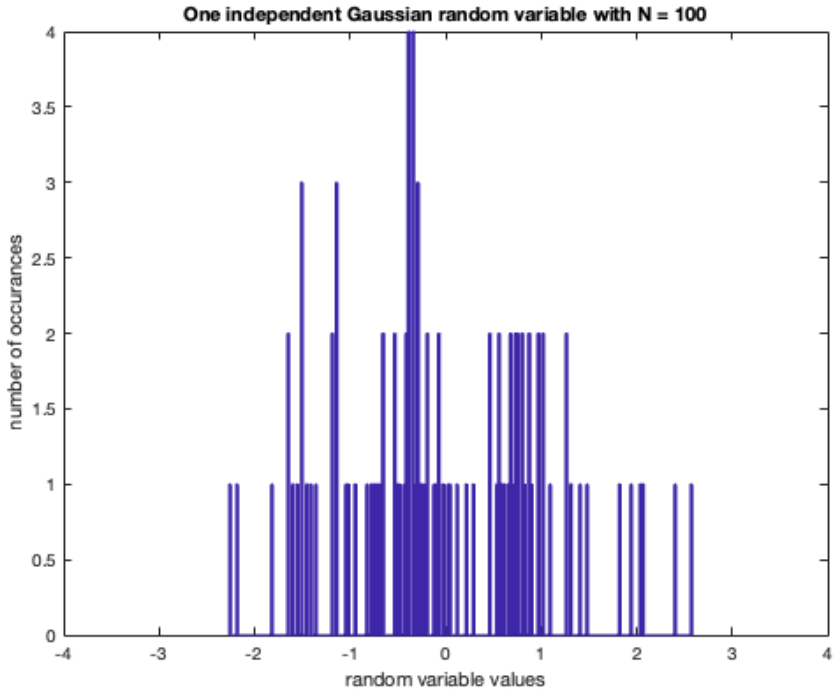
5)

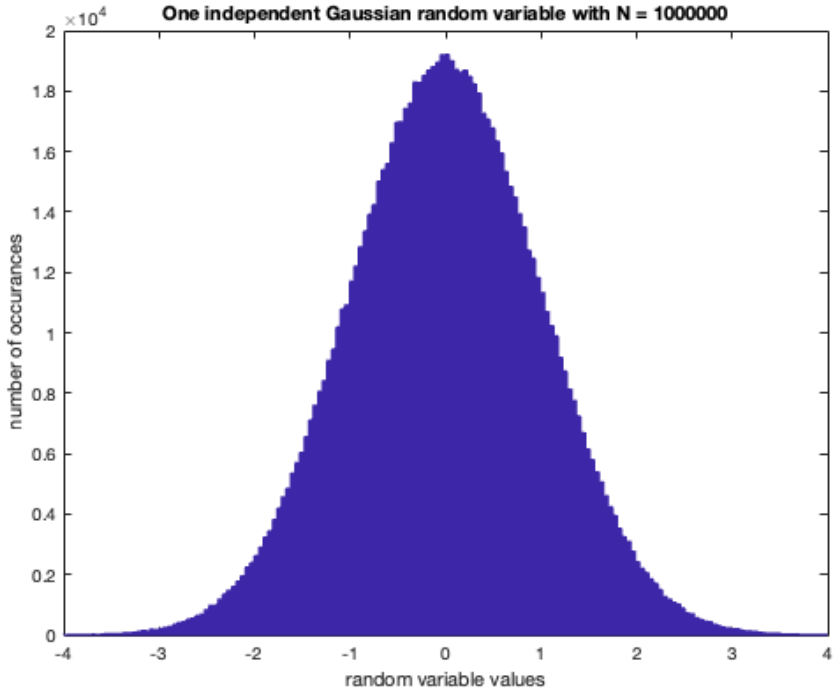
```
M1 = 2;
M2 = 6;

N1 = 100;
N2 = 1000000;
```


a)

generate random samples for M independent Gaussian random variables using a linear transformation approach

```
Z1_1 = zeros(N1, M1);
for i = 1:M1/2
    U_1s_1 = rand(N1,1);
    U_2s_1 = rand(N1,1);

    Z1_1(:,(i*2)-1) = sqrt(-2*log(U_1s_1)).*cos(2*pi*U_2s_1);
    Z1_1(:,(i*2)) = sqrt(-2*log(U_1s_1)).*sin(2*pi*U_2s_1);

end

figure()
hist(Z1_1(:,1),200)
xlim([-4 4])
title(sprintf("One independent Gaussian random variable with N = " + N1))
xlabel("random variable values")
ylabel("number of occurances")

Z2_1 = zeros(N2, M1);
for i = 1:M1/2
    U_1s_2 = rand(N2,1);
    U_2s_2 = rand(N2,1);

    Z2_1(:,(i*2)-1) = sqrt(-2*log(U_1s_2)).*cos(2*pi*U_2s_2);
    Z2_1(:,(i*2)) = sqrt(-2*log(U_1s_2)).*sin(2*pi*U_2s_2);

end

figure()
hist(Z2_1(:,1),200)
xlim([-4 4])
title(sprintf("One independent Gaussian random variable with N = " + N2))
xlabel("random variable values")
ylabel("number of occurances")

Z1_2 = zeros(N1, M2);
for i = 1:M2/2
    U_1s_1 = rand(N1,1);
    U_2s_1 = rand(N1,1);

    Z1_2(:,(i*2)-1) = sqrt(-2*log(U_1s_1)).*cos(2*pi*U_2s_1);
    Z1_2(:,(i*2)) = sqrt(-2*log(U_1s_1)).*sin(2*pi*U_2s_1);

end

Z2_2 = zeros(N2, M2);
for i = 1:M2/2
    U_1s_2 = rand(N2,1);
    U_2s_2 = rand(N2,1);

    Z2_2(:,(i*2)-1) = sqrt(-2*log(U_1s_2)).*cos(2*pi*U_2s_2);
    Z2_2(:,(i*2)) = sqrt(-2*log(U_1s_2)).*sin(2*pi*U_2s_2);

end
```





b) and c)

```
% start with Covariance Matrix with zero correlation:
fprintf("rho 0:\n");
rho = 0;

% rho 0, M = 2, N = 100
C1 = get_covariance_matrix(rho, M1);
[P1,D1] = eig(C1);
A1 = P1 * sqrt(D1);

W1_1_rho0 = zeros(M1);
for i = 1:N1
```

```
    W1_1_rho0(:,i) = A1*(Z1_1(i,:)');
end

C_w = cov(W1_1_rho0')
C_z = cov(Z1_1)

% rho 0, M = 2, N = 1000000
C1 = get_covariance_matrix(rho, M1);
[P1,D1] = eig(C1);
A1 = P1 * sqrt(D1);

W2_1_rho0 = zeros(M1);
for i = 1:N2
    W2_1_rho0(:,i) = A1*(Z2_1(i,:)');
end

figure()
hist3(W2_1_rho0','Nbins',[200 200],'CDataMode','auto','FaceColor','interp')
shading flat;
xlim([-4 4])
ylim([-4 4])
xlabel("random variable values for M_1")
ylabel("random variable values for M_2")
title("Histogram of two uncorrelated Gaussian random variables")
view(2)
grid off;

C_w = cov(W2_1_rho0')
C_z = cov(Z2_1)

% rho 0, M = 10, N = 100
C1 = get_covariance_matrix(rho, M2);
[P1,D1] = eig(C1);
A1 = P1 * sqrt(D1);

W1_2_rho0 = zeros(M2);
for i = 1:N1
    W1_2_rho0(:,i) = A1*(Z1_2(i,:)');
end

C_w = cov(W1_2_rho0')
C_z = cov(Z1_2)

% rho 0, M = 10, N = 1000000
C1 = get_covariance_matrix(rho, M2);
[P1,D1] = eig(C1);
A1 = P1 * sqrt(D1);

W2_2_rho0 = zeros(M2);
for i = 1:N2
    W2_2_rho0(:,i) = A1*(Z2_2(i,:)');
end

C_w = cov(W2_2_rho0')
C_z = cov(Z2_2)

% repeat with rho = .25:
fprintf("rho .1:\n");
rho = .25;

% rho 0, M = 2, N = 100
C1 = get_covariance_matrix(rho, M1);
[P1,D1] = eig(C1);
A1 = P1 * sqrt(D1);

W1_1_rho0 = zeros(M1);
```

```
for i = 1:N1
    W1_1_rho0(:,i) = A1*(Z1_1(i,:)');
end

C_w = cov(W1_1_rho0')
C_z = cov(Z1_1)

% rho 0, M = 2, N = 1000000
C1 = get_covariance_matrix(rho, M1);
[P1,D1] = eig(C1);
A1 = P1 * sqrt(D1);

W2_1_rho0 = zeros(M1);
for i = 1:N2
    W2_1_rho0(:,i) = A1*(Z2_1(i,:)');
end

figure()
hist3(W2_1_rho0','Nbins',[200 200],'CDataMode','auto','FaceColor','interp')
shading flat;
xlim([-4 4])
ylim([-4 4])
xlabel("random variable values for M_1")
ylabel("random variable values for M_2")
title("Histogram of two correlated Gaussian random variables with \rho = " + rho)
view(2)
grid off;

C_w = cov(W2_1_rho0')
C_z = cov(Z2_1)

% rho 0, M = 10, N = 100
C1 = get_covariance_matrix(rho, M2);
[P1,D1] = eig(C1);
A1 = P1 * sqrt(D1);

W1_2_rho0 = zeros(M2);
for i = 1:N1
    W1_2_rho0(:,i) = A1*(Z1_2(i,:)');
end

C_w = cov(W1_2_rho0')
C_z = cov(Z1_2)

% rho 0, M = 10, N = 1000000
C1 = get_covariance_matrix(rho, M2);
[P1,D1] = eig(C1);
A1 = P1 * sqrt(D1);

W2_2_rho0 = zeros(M2);
for i = 1:N2
    W2_2_rho0(:,i) = A1*(Z2_2(i,:)');
end

C_w = cov(W2_2_rho0')
C_z = cov(Z2_2)

% repeat with rho = .5:
fprintf("rho .5:\n");
rho = .5;
```



```
% rho 0, M = 2, N = 100
C1 = get_covariance_matrix(rho, M1);
[P1,D1] = eig(C1);
A1 = P1 * sqrt(D1);
```

```
W1_1_rho0 = zeros(M1);
for i = 1:N1
    W1_1_rho0(:,i) = A1*(Z1_1(i,:)');
end

C_w = cov(W1_1_rho0')
C_z = cov(Z1_1)

% rho 0, M = 2, N = 1000000
C1 = get_covariance_matrix(rho, M1);
[P1,D1] = eig(C1);
A1 = P1 * sqrt(D1);

W2_1_rho0 = zeros(M1);
for i = 1:N2
    W2_1_rho0(:,i) = A1*(Z2_1(i,:)');
end

C_w = cov(W2_1_rho0')
C_z = cov(Z2_1)

figure()
hist3(W2_1_rho0','Nbins',[200 200],'CDataMode','auto','FaceColor','interp')
shading flat;
xlim([-4 4])
ylim([-4 4])
xlabel("random variable values for M_1")
ylabel("random variable values for M_2")
title("Histogram of two correlated Gaussian random variables with \rho = " + rho)
view(2)
grid off;

% rho 0, M = 10, N = 100
C1 = get_covariance_matrix(rho, M2);
[P1,D1] = eig(C1);
A1 = P1 * sqrt(D1);

W1_2_rho0 = zeros(M2);
for i = 1:N1
    W1_2_rho0(:,i) = A1*(Z1_2(i,:)');
end

C_w = cov(W1_2_rho0')
C_z = cov(Z1_2)

% rho 0, M = 10, N = 1000000
C1 = get_covariance_matrix(rho, M2);
[P1,D1] = eig(C1);
A1 = P1 * sqrt(D1);

W2_2_rho0 = zeros(M2);
for i = 1:N2
    W2_2_rho0(:,i) = A1*(Z2_2(i,:)');
end

C_w = cov(W2_2_rho0')
C_z = cov(Z2_2)

% repeat with rho = .95:
fprintf("rho .95:\n");
rho = .95;

% rho 0, M = 2, N = 100
C1 = get_covariance_matrix(rho, M1);
[P1,D1] = eig(C1);
A1 = P1 * sqrt(D1);
```

```
W1_1_rho0 = zeros(M1);
for i = 1:N1
    W1_1_rho0(:,i) = A1*(Z1_1(i,:)');
end

C_w = cov(W1_1_rho0')
C_z = cov(Z1_1)

% rho 0, M = 2, N = 1000000
C1 = get_covariance_matrix(rho, M1);
[P1,D1] = eig(C1);
A1 = P1 * sqrt(D1);

W2_1_rho0 = zeros(M1);
for i = 1:N2
    W2_1_rho0(:,i) = A1*(Z2_1(i,:)');
end

C_w = cov(W2_1_rho0')
C_z = cov(Z2_1)

figure()
hist3(W2_1_rho0','Nbins',[200 200],'CDataMode','auto','FaceColor','interp')
shading flat;
xlim([-4 4])
ylim([-4 4])
xlabel("random variable values for M_1")
ylabel("random variable values for M_2")
title("Histogram of two correlated Gaussian random variables with \rho = " + rho)
view(2)
grid off;

% rho 0, M = 10, N = 100
C1 = get_covariance_matrix(rho, M2);
[P1,D1] = eig(C1);
A1 = P1 * sqrt(D1);

W1_2_rho0 = zeros(M2);
for i = 1:N1
    W1_2_rho0(:,i) = A1*(Z1_2(i,:)');
end

C_w = cov(W1_2_rho0')
C_z = cov(Z1_2)

% rho 0, M = 10, N = 1000000
C1 = get_covariance_matrix(rho, M2);
[P1,D1] = eig(C1);
A1 = P1 * sqrt(D1);

W2_2_rho0 = zeros(M2);
for i = 1:N2
    W2_2_rho0(:,i) = A1*(Z2_2(i,:)');
end

C_w = cov(W2_2_rho0')
C_z = cov(Z2_2)

function C = get_covariance_matrix(rho,M)
    C = zeros(M, M);
    for i = 1:M
        for j = 1:M
            power = i - j;
            C(i,j) = rho^(abs(power));
        end
```

```
    end
end
```

```
rho 0:

C_w =

    1.0722    0.1664
    0.1664    0.9323


C_z =

    1.0722    0.1664
    0.1664    0.9323


C_w =

    0.9982   -0.0014
   -0.0014    1.0021


C_z =

    0.9982   -0.0014
   -0.0014    1.0021


C_w =

    0.9910   -0.0624   -0.1506    0.0146   -0.0253   -0.1197
   -0.0624    0.9235   -0.0760   -0.0508    0.1021   -0.0580
   -0.1506   -0.0760    1.2316    0.0375    0.0713    0.0442
    0.0146   -0.0508    0.0375    0.9462    0.1329   -0.1767
   -0.0253    0.1021    0.0713    0.1329    1.0472    0.0760
   -0.1197   -0.0580    0.0442   -0.1767    0.0760    1.2272


C_z =

    0.9910   -0.0624   -0.1506    0.0146   -0.0253   -0.1197
   -0.0624    0.9235   -0.0760   -0.0508    0.1021   -0.0580
   -0.1506   -0.0760    1.2316    0.0375    0.0713    0.0442
    0.0146   -0.0508    0.0375    0.9462    0.1329   -0.1767
   -0.0253    0.1021    0.0713    0.1329    1.0472    0.0760
   -0.1197   -0.0580    0.0442   -0.1767    0.0760    1.2272


C_w =

    0.9975   -0.0002   -0.0007   -0.0014    0.0005   -0.0007
   -0.0002    1.0011   -0.0007    0.0002    0.0019    0.0010
   -0.0007   -0.0007    0.9990   -0.0011    0.0005   -0.0010
   -0.0014    0.0002   -0.0011    0.9987    0.0007    0.0010
    0.0005    0.0019    0.0005    0.0007    0.9989    0.0001
   -0.0007    0.0010   -0.0010    0.0010    0.0001    0.9990


C_z =

    0.9975   -0.0002   -0.0007   -0.0014    0.0005   -0.0007
   -0.0002    1.0011   -0.0007    0.0002    0.0019    0.0010
   -0.0007   -0.0007    0.9990   -0.0011    0.0005   -0.0010
   -0.0014    0.0002   -0.0011    0.9987    0.0007    0.0010
```

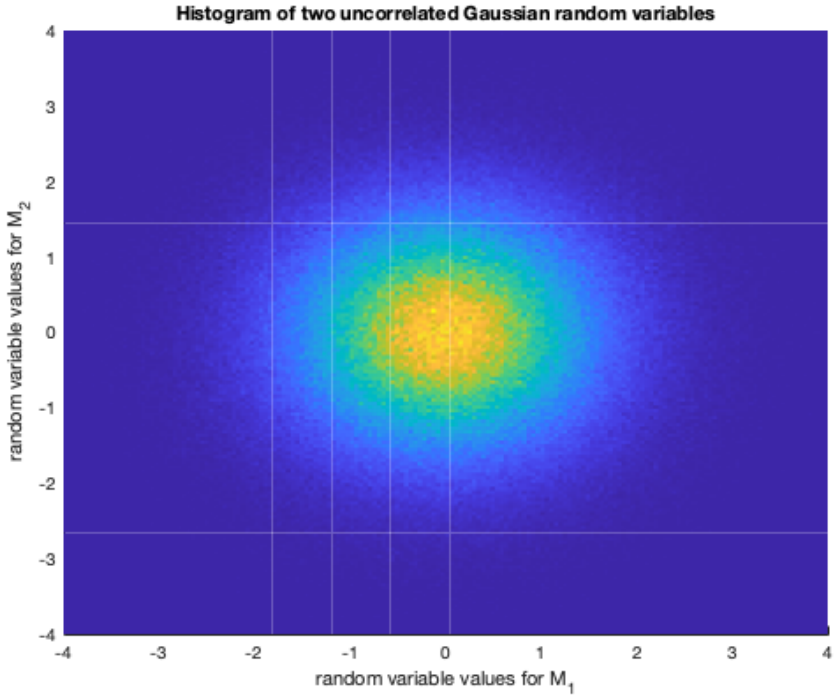
```
    0.0005    0.0019    0.0005    0.0007    0.9989    0.0001
   -0.0007    0.0010   -0.0010    0.0010    0.0001    0.9990

rho .1:

C_w =

    0.8237    0.1806
    0.1806    1.1459


C_z =

    1.0722    0.1664
    0.1664    0.9323


C_w =

    1.0020    0.2520
    0.2520    0.9993


C_z =

    0.9982   -0.0014
   -0.0014    1.0021


C_w =

    0.9599    0.1109    0.0305    0.1511   -0.0504   -0.0397
    0.1109    0.9273    0.3799    0.2140   -0.0236    0.0002
    0.0305    0.3799    1.2140    0.3295    0.0241    0.0312
    0.1511    0.2140    0.3295    1.3399    0.4375    0.0808
   -0.0504   -0.0236    0.0241    0.4375    1.0552    0.3518
   -0.0397    0.0002    0.0312    0.0808    0.3518    0.9937


C_z =

    0.9910   -0.0624   -0.1506    0.0146   -0.0253   -0.1197
   -0.0624    0.9235   -0.0760   -0.0508    0.1021   -0.0580
   -0.1506   -0.0760    1.2316    0.0375    0.0713    0.0442
    0.0146   -0.0508    0.0375    0.9462    0.1329   -0.1767
   -0.0253    0.1021    0.0713    0.1329    1.0472    0.0760
   -0.1197   -0.0580    0.0442   -0.1767    0.0760    1.2272


C_w =

    0.9980    0.2494    0.0625    0.0164    0.0021    0.0017
    0.2494    0.9999    0.2499    0.0624    0.0167    0.0055
    0.0625    0.2499    0.9987    0.2491    0.0615    0.0156
    0.0164    0.0624    0.2491    0.9977    0.2497    0.0620
    0.0021    0.0167    0.0615    0.2497    1.0003    0.2504
    0.0017    0.0055    0.0156    0.0620    0.2504    0.9994


C_z =
```

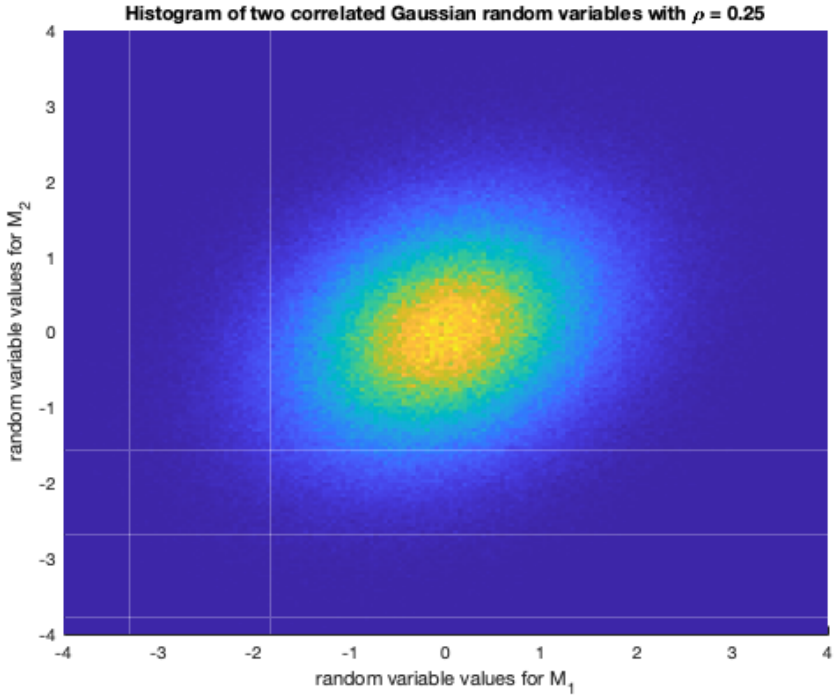


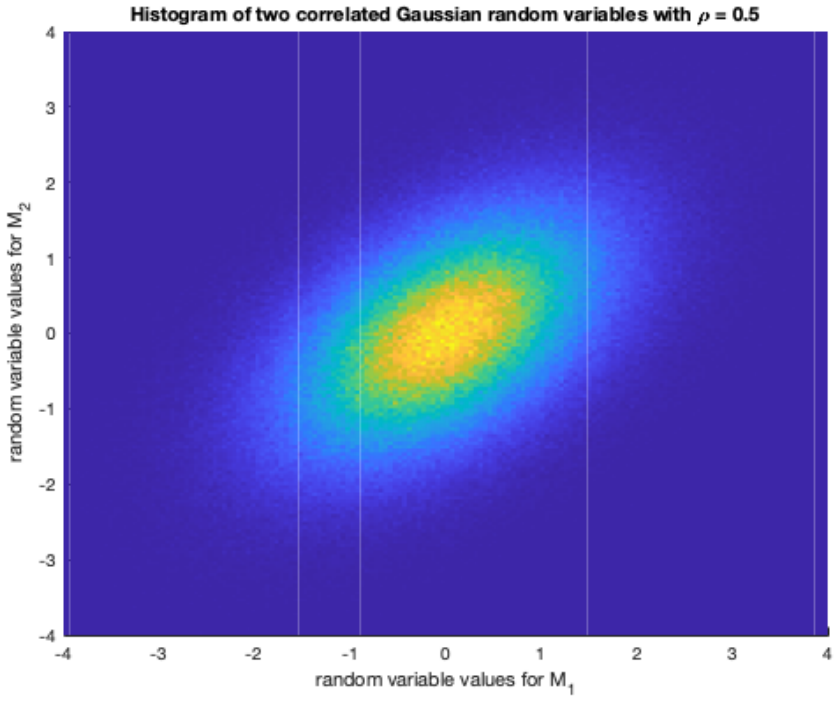
```
    0.9975   -0.0002   -0.0007   -0.0014    0.0005   -0.0007
   -0.0002    1.0011   -0.0007    0.0002    0.0019    0.0010
   -0.0007   -0.0007    0.9990   -0.0011    0.0005   -0.0010
   -0.0014    0.0002   -0.0011    0.9987    0.0007    0.0010
    0.0005    0.0019    0.0005    0.0007    0.9989    0.0001
```

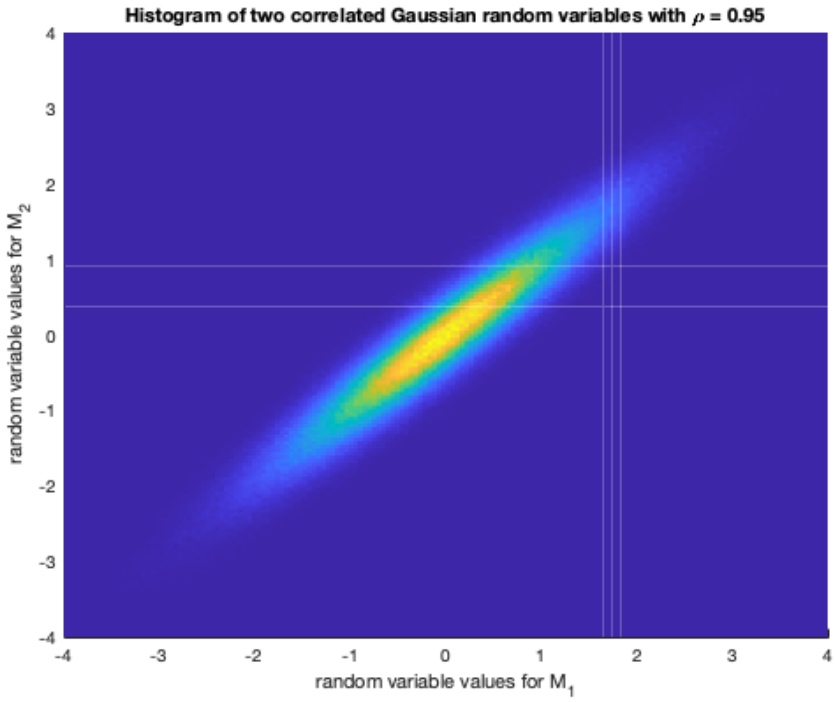
```
   -0.0007    0.0010   -0.0010    0.0010    0.0001    0.9990

rho .5:

C_w =

    0.8232    0.4312
    0.4312    1.1114


C_z =

    1.0722    0.1664
    0.1664    0.9323


C_w =

    1.0024    0.5020
    0.5020    0.9999


C_z =

    0.9982   -0.0014
   -0.0014    1.0021


C_w =

    0.7909    0.4624    0.1978    0.2215    0.1309   -0.0477
    0.4624    1.0793    0.6659    0.4113    0.1926    0.0127
    0.1978    0.6659    1.2012    0.6479    0.2995    0.1525
    0.2215    0.4113    0.6479    1.3607    0.7221    0.3519
    0.1309    0.1926    0.2995    0.7221    1.1478    0.5838
   -0.0477    0.0127    0.1525    0.3519    0.5838    1.0799


C_z =

    0.9910   -0.0624   -0.1506    0.0146   -0.0253   -0.1197
   -0.0624    0.9235   -0.0760   -0.0508    0.1021   -0.0580
   -0.1506   -0.0760    1.2316    0.0375    0.0713    0.0442
    0.0146   -0.0508    0.0375    0.9462    0.1329   -0.1767
   -0.0253    0.1021    0.0713    0.1329    1.0472    0.0760
   -0.1197   -0.0580    0.0442   -0.1767    0.0760    1.2272


C_w =

    0.9974    0.4998    0.2497    0.1255    0.0625    0.0324
    0.4998    1.0001    0.4994    0.2491    0.1248    0.0625
    0.2497    0.4994    0.9988    0.4991    0.2492    0.1256
    0.1255    0.2491    0.4991    0.9984    0.4991    0.2505
    0.0625    0.1248    0.2492    0.4991    0.9979    0.4996
    0.0324    0.0625    0.1256    0.2505    0.4996    1.0014


C_z =

    0.9975   -0.0002   -0.0007   -0.0014    0.0005   -0.0007
   -0.0002    1.0011   -0.0007    0.0002    0.0019    0.0010
   -0.0007   -0.0007    0.9990   -0.0011    0.0005   -0.0010
   -0.0014    0.0002   -0.0011    0.9987    0.0007    0.0010
    0.0005    0.0019    0.0005    0.0007    0.9989    0.0001
   -0.0007    0.0010   -0.0010    0.0010    0.0001    0.9990
```

```
rho .95:

C_w =

    0.8839    0.8822
    0.8822    0.9878


C_z =

    1.0722    0.1664
    0.1664    0.9323


C_w =

    1.0024    0.9521
    0.9521    1.0015


C_z =

    0.9982   -0.0014
   -0.0014    1.0021


C_w =

    1.2860    1.2246    1.1254    1.1056    1.0465    1.0045
    1.2246    1.2564    1.1603    1.1295    1.0775    1.0397
    1.1254    1.1603    1.1445    1.1001    1.0550    1.0212
    1.1056    1.1295    1.1001    1.1685    1.1222    1.0759
    1.0465    1.0775    1.0550    1.1222    1.1859    1.1444
    1.0045    1.0397    1.0212    1.0759    1.1444    1.2158


C_z =

    0.9910   -0.0624   -0.1506    0.0146   -0.0253   -0.1197
   -0.0624    0.9235   -0.0760   -0.0508    0.1021   -0.0580
   -0.1506   -0.0760    1.2316    0.0375    0.0713    0.0442
    0.0146   -0.0508    0.0375    0.9462    0.1329   -0.1767
   -0.0253    0.1021    0.0713    0.1329    1.0472    0.0760
   -0.1197   -0.0580    0.0442   -0.1767    0.0760    1.2272


C_w =

    0.9987    0.9488    0.9017    0.8565    0.8132    0.7727
    0.9488    0.9989    0.9494    0.9019    0.8564    0.8137
    0.9017    0.9494    0.9995    0.9494    0.9014    0.8567
    0.8565    0.9019    0.9494    0.9993    0.9488    0.9016
    0.8132    0.8564    0.9014    0.9488    0.9985    0.9488
    0.7727    0.8137    0.8567    0.9016    0.9488    0.9989


C_z =

    0.9975   -0.0002   -0.0007   -0.0014    0.0005   -0.0007
   -0.0002    1.0011   -0.0007    0.0002    0.0019    0.0010
   -0.0007   -0.0007    0.9990   -0.0011    0.0005   -0.0010
   -0.0014    0.0002   -0.0011    0.9987    0.0007    0.0010
    0.0005    0.0019    0.0005    0.0007    0.9989    0.0001
   -0.0007    0.0010   -0.0010    0.0010    0.0001    0.9990
```


Histogram of two uncorrelated Gaussian random variables
random variable values for M2
random variable values for M1
4
3
2
1
0
-1
-2
-3
-4


Histogram of two correlated Gaussian random variables with ρ = 0.25
random variable values for M2
random variable values for M1
4
3
2
1
0
-1
-2
-3
-4





Published with MATLAB® R2022b