EE 511
Fall 2018
Prof. John Silvester

Project 1 - Uniform Random Numbers

Pavan Athreya Narasimha Murthy
USC ID: 9129210968
Section: Wednesday 9:00 AM
E-mail: pavanatn@usc.edu

1. *Let* $X \sim U(0,1)$, evaluate the mean, μ and variance, σ_X^2 .

Problem Statement:

To simulate a uniform probability distribution and compute the mean and variance. This is to get a general idea and uniform distribution and its properties

Theory/Analysis:

Uniform distribution is a derivative of the Beta distribution which are a family of continuous distributions defined from interval [0, 1].

Beta(α, β)

$$f(x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1} (1-x)^{\beta-1}$$

$$\mu_x = \frac{\alpha}{\alpha+\beta}$$

$$\sigma_x^2 = \frac{\alpha \beta}{(\alpha+\beta+1)(\alpha+\beta)^2}$$

$$\Downarrow$$

Uniform

$$\alpha = \beta = 1$$

$$\mu = 1/2 \text{ or } 0.5$$

$$\sigma_x^2 = 1/12 \text{ or } 0.08\bar{3}$$

Uniform distribution

$$f(x) = \frac{1}{b-a}$$

b - upper limit

a - lower limit

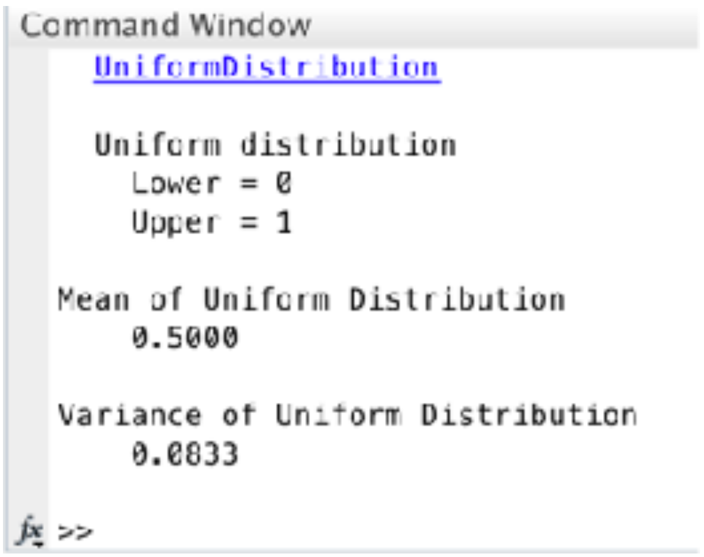
$$\mu_x = \frac{b+a}{2} \qquad \sigma_x^2 = \frac{(b-a)^2}{12}$$

Simulation Methodology:

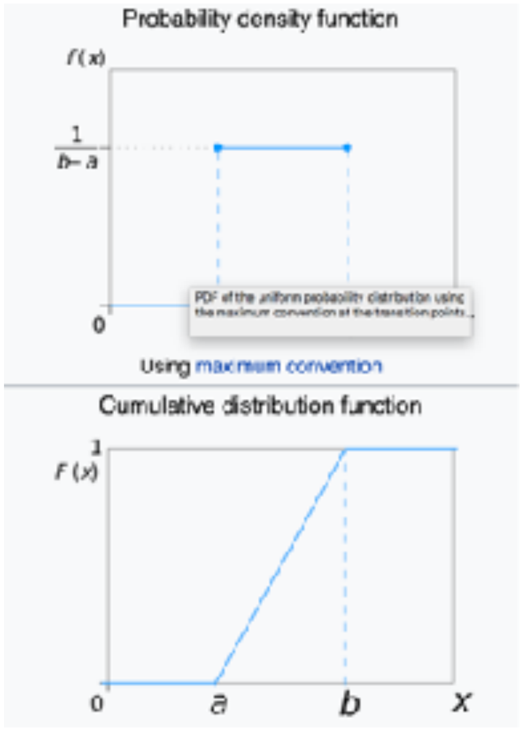
There are built-in functions in Matlab which can create probability distribution classes using the variables/parameters required. For Example, in our case, we need the lower and the upper bound of the uniform distribution.

Results:

The mean and variance across multiple runs remain the same as explained/expected in the analysis section.

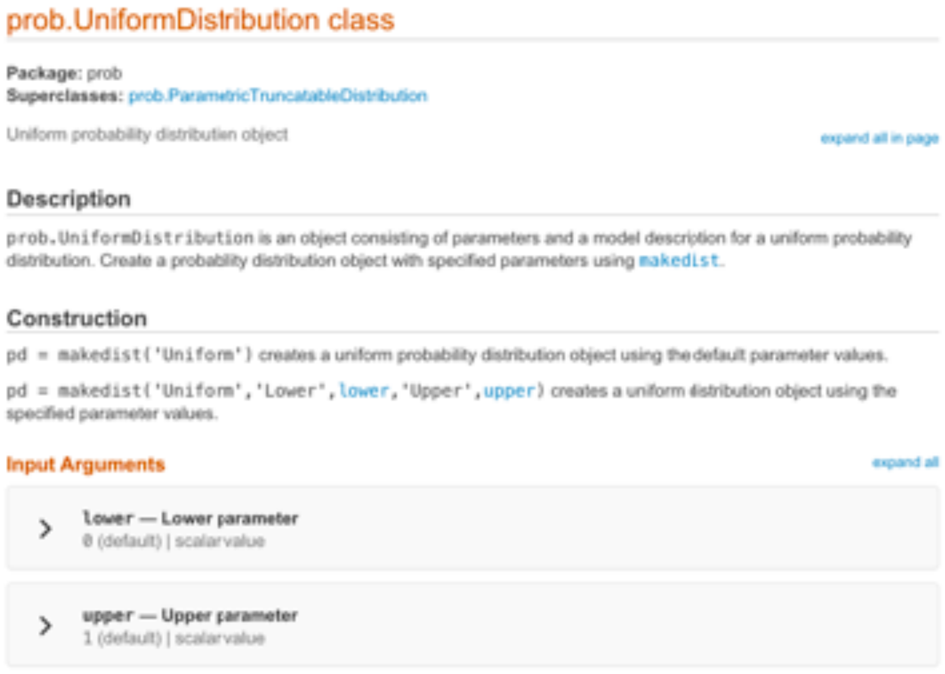


```
Command Window
  UniformDistribution

    Uniform distribution
      Lower = 0
      Upper = 1

Mean of Uniform Distribution
    0.5000

Variance of Uniform Distribution
    0.0833

fx >>
```



References:

Matlab Documentation for built-in Probability distribution class and https://en.wikipedia.org/wiki/Uniform_distribution_(continuous)



prob.UniformDistribution class

Package: prob
Superclasses: prob.ParametricTruncatableDistribution

Uniform probability distribution object

expand all in page

Description

`prob.UniformDistribution` is an object consisting of parameters and a model description for a uniform probability distribution. Create a probability distribution object with specified parameters using `makedist`.

Construction

`pd = makedist('Uniform')` creates a uniform probability distribution object using the default parameter values.

`pd = makedist('Uniform','Lower',lower,'Upper',upper)` creates a uniform distribution object using the specified parameter values.

Input Arguments

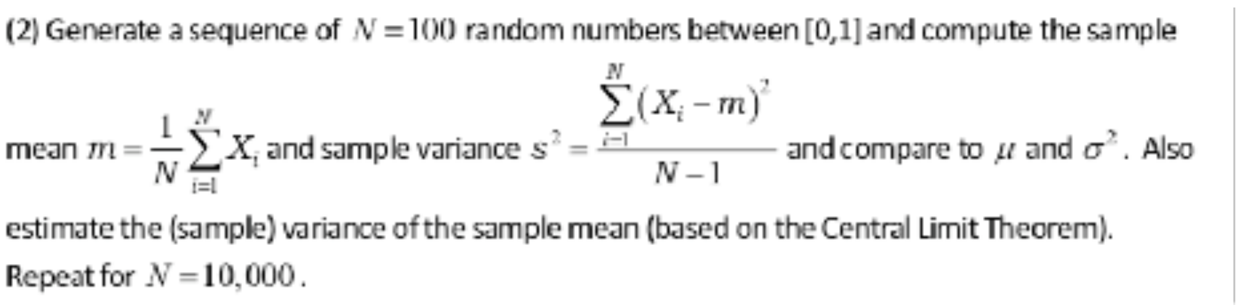
expand all

`lower` — Lower parameter
0 (default) | scalar value

`upper` — Upper parameter
1 (default) | scalar value

Source Code:

```
%Name: Pavan Athreya Narasimha Murthy
%USC ID: 9129210968
%E-mail: pavanatn@usc.edu
%Ph: +1(323)-684 5715
%Term: Fall 2018
%Course: EE511
%Professor: John Silvester

%Clear the Workspace variables and command window for every run
clear all;
clc;

% First part of Project 1
%Class: Uniform Distribution:
ProbabilityDistribution = makedist('Uniform','lower',0,'upper',1);
MeanOfUniformDis = mean(ProbabilityDistribution);
VarianceOfUniformDis = var(ProbabilityDistribution);

disp(ProbabilityDistribution);
disp("Mean of Uniform Distribution");
disp(MeanOfUniformDis);
disp("Variance of Uniform Distribution");
disp(VarianceOfUniformDis);
```



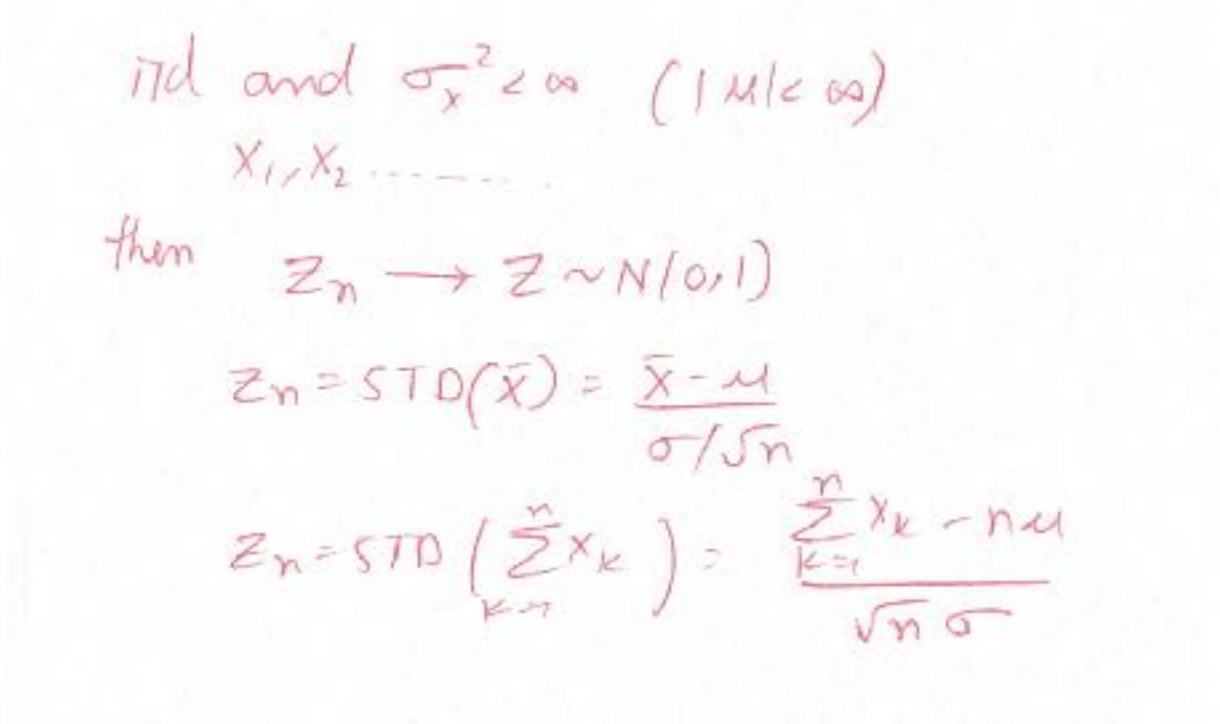
(2) Generate a sequence of $N = 100$ random numbers between [0,1] and compute the sample mean $m = \frac{1}{N}\sum_{i=1}^{N} X_i$ and sample variance $s^2 = \frac{\sum_{i=1}^{N}(X_i - m)^2}{N-1}$ and compare to μ and σ^2. Also estimate the (sample) variance of the sample mean (based on the Central Limit Theorem). Repeat for $N = 10,000$.

Problem Statement:

Compare the sample mean to population mean and sample variance to population variance to verify the central limit theorem which states that sample mean converges to population mean in normal distribution provided that the sample and independent and identically distributed with finite variance.

Theory/Analysis:

In probability theory, the central limit theorem (CLT) establishes that, in some situations, when independent random variables are added, their properly normalized sum tends toward a normal distribution (informally a "bell curve") even if the original variables themselves are not normally distributed. The theorem is a key concept in probability theory because it implies that probabilistic and statistical methods that work for normal distributions can be applicable to many problems involving other types of distributions.



iid and $\sigma_x^2 < \infty$ $(|\mu| < \infty)$

$X_1, X_2 \ldots\ldots$

then $Z_n \rightarrow Z \sim N(0,1)$

$$Z_n = STD(\bar{X}) = \frac{\bar{X} - \mu}{\sigma/\sqrt{n}}$$

$$Z_n = STD\left(\sum_{k=1}^{n} X_k\right) = \frac{\sum_{k=1}^{n} X_k - n\mu}{\sqrt{n}\,\sigma}$$

Simulation Methodology:

Random samples are created using Matlab's in-built function "RAND" after which the population mean and population variances are computed using "MEAN" and "VAR" methods. Then, we calculate the sample mean and sample variance manually using the formula given in the problem statement.

Results:
Output from Matlab command window:

```
Question 2 Answer: Part1
Number of samples:
   100

Sample Mean
    0.5414

Sample Variance
    0.0829

variance of the sample mean:
    0.0821

Question 2 Answer: Part 2
Number of samples:
       10000

Sample Mean
    0.5008

Sample Variance
    0.0830

variance of the sample mean:
    0.0830
```

References:

EE503 notes on Central limit theorem and https://en.wikipedia.org/wiki/Central_limit_theorem

Source Code:

```
%Name: Pavan Athreya Narasimha Murthy
%USC ID: 9129210968
%E-mail: pavanatn@usc.edu
%Ph: +1(323)-684 5715
%Term: Fall 2018
%Course: EE511
%Professor: John Silvester

%Clear the Workspace variables and command window for every run
clear all;
clc;
```

```
% Second part of Project 1
%For 100 samples
NumberOfSamples1 = 100;
RandomSamples1 = rand(100,1);
PopulationMean1 = mean(RandomSamples1);
PopulationVariance1 = var(RandomSamples1);

SumOfSamples1 = sum(RandomSamples1);
SampleMean1 = SumOfSamples1/NumberOfSamples1;
disp("Question 2 Answer: Part1");
disp("Number of samples:");
disp(NumberOfSamples1)
disp("Sample Mean");
disp(SampleMean1);
z1 = 0;
for i=1:100
   y1(i) = (RandomSamples1(i)-SampleMean1)*(RandomSamples1(i)-SampleMean1);
   z1 = z1 + y1(i);
end
SampleVariance1 = z1/(NumberOfSamples1-1);
disp("Sample Variance");
disp(SampleVariance1);
disp("variance of the sample mean:");
VarianceBasedOnSMean1 = z1/(NumberOfSamples1);
disp(VarianceBasedOnSMean1);

%For 10000 samples
NumberOfSamples = 10000;
RandomSamples = rand(10000,1);
PopulationMean = mean(RandomSamples);
PopulationVariance = var(RandomSamples);

SumOfSamples = sum(RandomSamples);
SampleMean = SumOfSamples/NumberOfSamples;
disp("Question 2 Answer: Part 2");
disp("Number of samples:");
disp(NumberOfSamples)
disp("Sample Mean");
disp(SampleMean);
z = 0;
for i=1:10000
   y(i) = (RandomSamples(i)-SampleMean)*(RandomSamples(i)-SampleMean);
   z = z + y(i);
end
SampleVariance = z/(NumberOfSamples-1);
disp("Sample Variance");
disp(SampleVariance);
disp("variance of the sample mean:");
VarianceBasedOnSMean = z/(NumberOfSamples);
disp(VarianceBasedOnSMean);
```

(3) The Central Limit Theorem says that $m = \frac{\sum_{i=1}^{n} X_i}{n} \to N(\mu, \sigma^2 / n)$. Repeat the experiment in (2 with $N = 100$) 50 times to generate a set of sample means $\{m_j, j = 1..50\}$. Do they appear to be approximately normally distributed values with mean μ and variance σ^2 / n ?

Problem Statement:

To verify the Statement of the central limit theorem as explain in the previous problem.

Theory/Analysis:

Explained in the previous problem analysis.

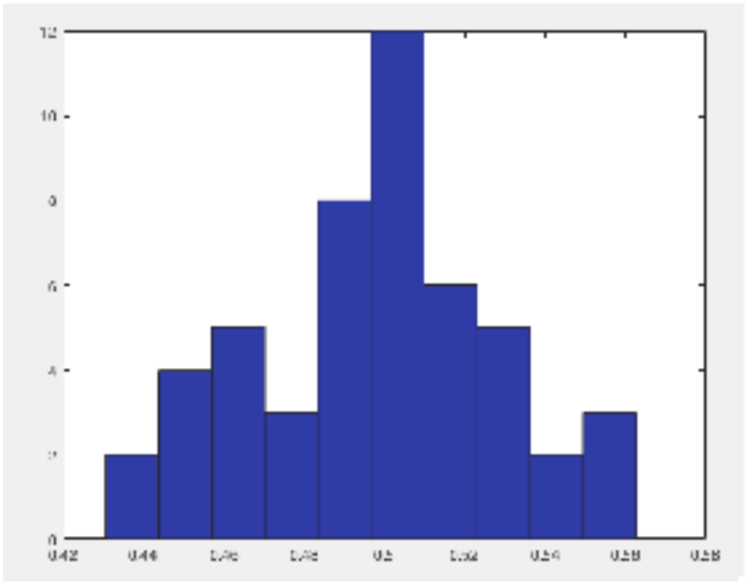
Simulation Methodology:

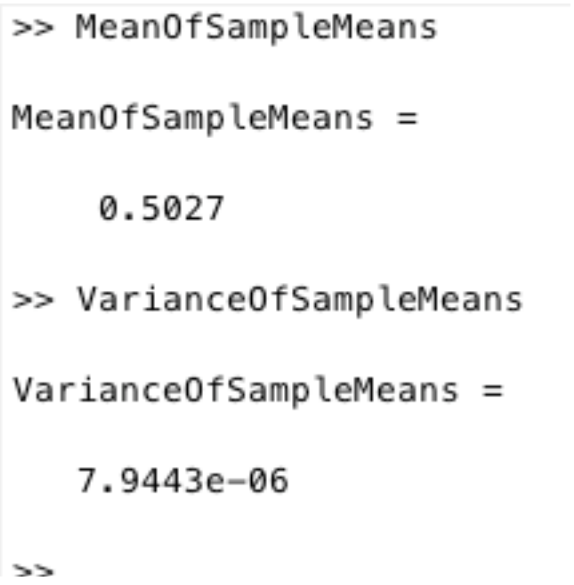
We generate 100 random samples and compute their sample means and repeat this process 50 times to calculate 50 means generated from 50 of the hundred random samples. Following that we calculate the mean and variance of the 50 means and variances. Using that data we plot of the mean of the means to verify that it resembles a bell curve(Gaussian distribution)

Note: The plot need many trials before it started to resemble a bell curve.

Results:

Yes, They appear to be approximately normally distributed values with mean and variance shown below

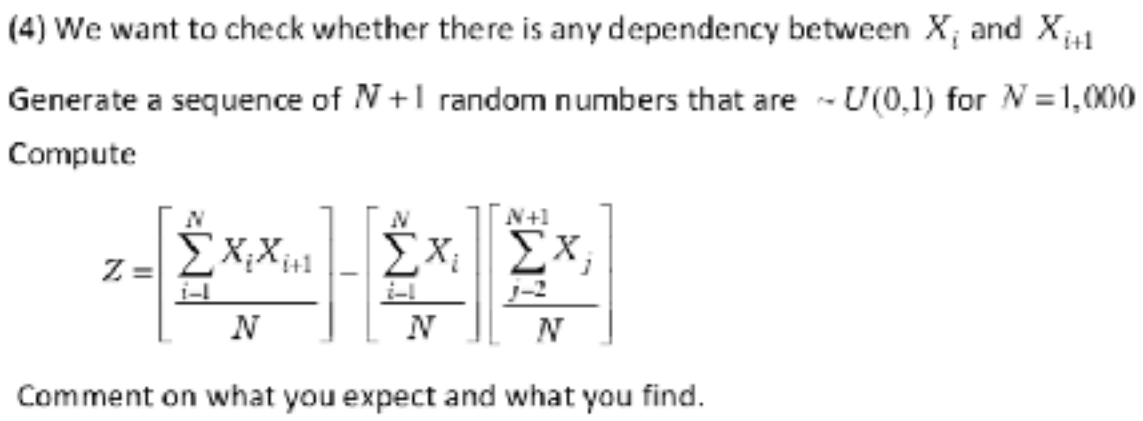




```
>> MeanOfSampleMeans

MeanOfSampleMeans =

    0.5027

>> VarianceOfSampleMeans

VarianceOfSampleMeans =

   7.9443e-06

>>
```

Source Code:

```
%Name: Pavan Athreya Narasimha Murthy
%USC ID: 9129210968
%E-mail: pavanatn@usc.edu
%Ph: +1(323)-684 5715
%Term: Fall 2018
%Course: EE511
%Professor: John Silvester

%Clear the Workspace variables and command window for every run
clear all;
clc;

% Third part of Project 1
for i=1:50
NumberOfSamples = 100;
RandomSamples = rand(100,1);
%PopulationMean = mean(RandomSamples);
%PopulationVariance = var(RandomSamples);
SumOfSamples(i) = sum(RandomSamples);
SampleMean(i) = SumOfSamples(i)/NumberOfSamples;
end
MeanOfSampleMeans = mean(SampleMean);
VarianceOfSampleMeans = var(SampleMean)/NumberOfSamples;
% Norm = normpdf(SampleMean);
% plot(SampleMean,Norm);
hist(SampleMean);
```


(4) We want to check whether there is any dependency between X_i and X_{i+1}

Generate a sequence of $N+1$ random numbers that are $\sim U(0,1)$ for $N=1{,}000$

Compute

$$Z = \left[\frac{\sum_{i=1}^{N} X_i X_{i+1}}{N}\right] - \left[\frac{\sum_{i=1}^{N} X_i}{N}\right]\left[\frac{\sum_{j=2}^{N+1} X_j}{N}\right]$$

Comment on what you expect and what you find.

Problem Statement:

Generate a 1000 random samples that are uniformly distributed. Then, check if the samples are dependent on one another by using the equation provided in the problem statement.

Theory/Analysis:

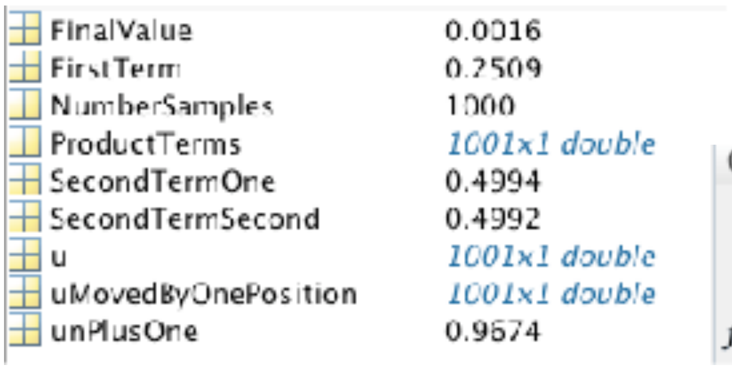
The consecutive random variables should not be dependent since uniform distribution has a condition that the sample should be independent and identically distributed with finite variance.
Other theory and analysis on uniform distributions are provided earlier in this report.

Simulation Methodology:

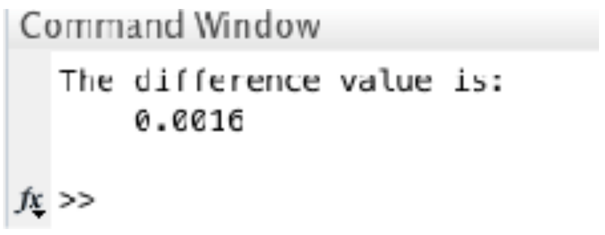
Implement a Matlab script to generate 1000 random sample ~U[0,1] and use the equation to calculate the co-relation between two consecutive terms. The source code clearly shows the implementation of the equation.

Results:

We see that the consecutive terms have very less co-relation. This proves that the uniform distribution is indeed independent and identically distributed random variables.



FinalValue	0.0016
FirstTerm	0.2509
NumberSamples	1000
ProductTerms	1001x1 double
SecondTermOne	0.4994
SecondTermSecond	0.4992
u	1001x1 double
uMovedByOnePosition	1001x1 double
unPlusOne	0.9674



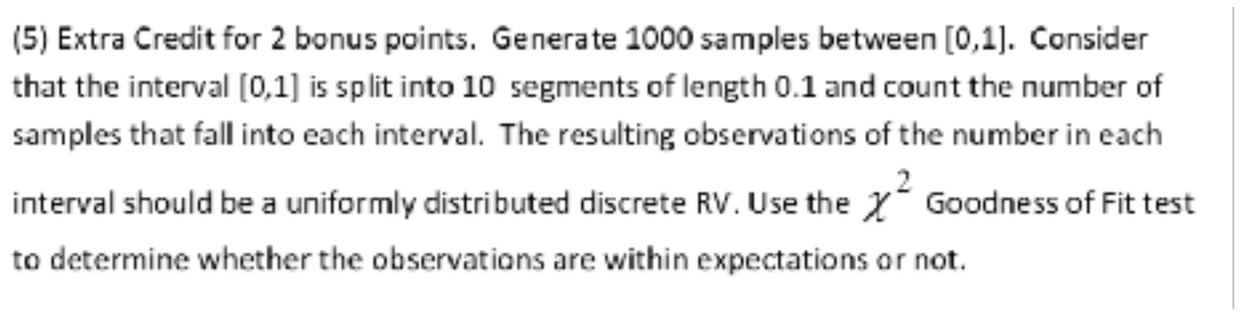
Command Window

```
The difference value is:
    0.0016

>>
```

Source Code:

```
%Name: Pavan Athreya Narasimha Murthy
%USC ID: 9129210968
%E-mail: pavanatn@usc.edu
%Ph: +1(323)-684 5715
%Term: Fall 2018
%Course: EE511
%Professor: John Silvester

%Clear the Workspace variables and command window for every run
clear all;
clc;

% Fourth part of Project 1
NumberSamples = 1000;
u = rand(1001, 1);
unPlusOne = u(1);
uMovedByOnePosition = vertcat(u(2:1001),unPlusOne);
ProductTerms = u.*uMovedByOnePosition;
FirstTerm = (sum(ProductTerms(1:1000))/NumberSamples);
SecondTermOne = (sum(u(1:1000))/NumberSamples);
SecondTermSecond = (sum(uMovedByOnePosition(1:1000))/NumberSamples);
FinalValue = FirstTerm - (SecondTermOne*SecondTermSecond);
disp("The difference value is:");
disp(FinalValue);
```

Extra Credit Question:


(5) Extra Credit for 2 bonus points. Generate 1000 samples between [0,1]. Consider that the interval [0,1] is split into 10 segments of length 0.1 and count the number of samples that fall into each interval. The resulting observations of the number in each interval should be a uniformly distributed discrete RV. Use the χ^2 Goodness of Fit test to determine whether the observations are within expectations or not.

Problem Statement:

To verify that the random samples generated between [0,1] and split into 10 segments of range 0.1 still resembles a uniform distribution. Also, to use the Goodness fit test to check if our results meets expectations.

Theory/Analysis:

The goodness of fit of a statistical model describes how well it fits a set of observations. Measures of goodness of fit typically summarize the discrepancy between observed values and the values expected under the model in question. Such measures can be used in statistical hypothesis testing, e.g. to test for normality of residuals, to test whether two samples are drawn from identical distributions, or whether outcome frequencies follow a specified distribution. In the analysis of variance, one of the components into which the variance is partitioned may be a lack-of-fit sum of squares.

Analysis and Theory on uniform distribution is provided earlier in this report.

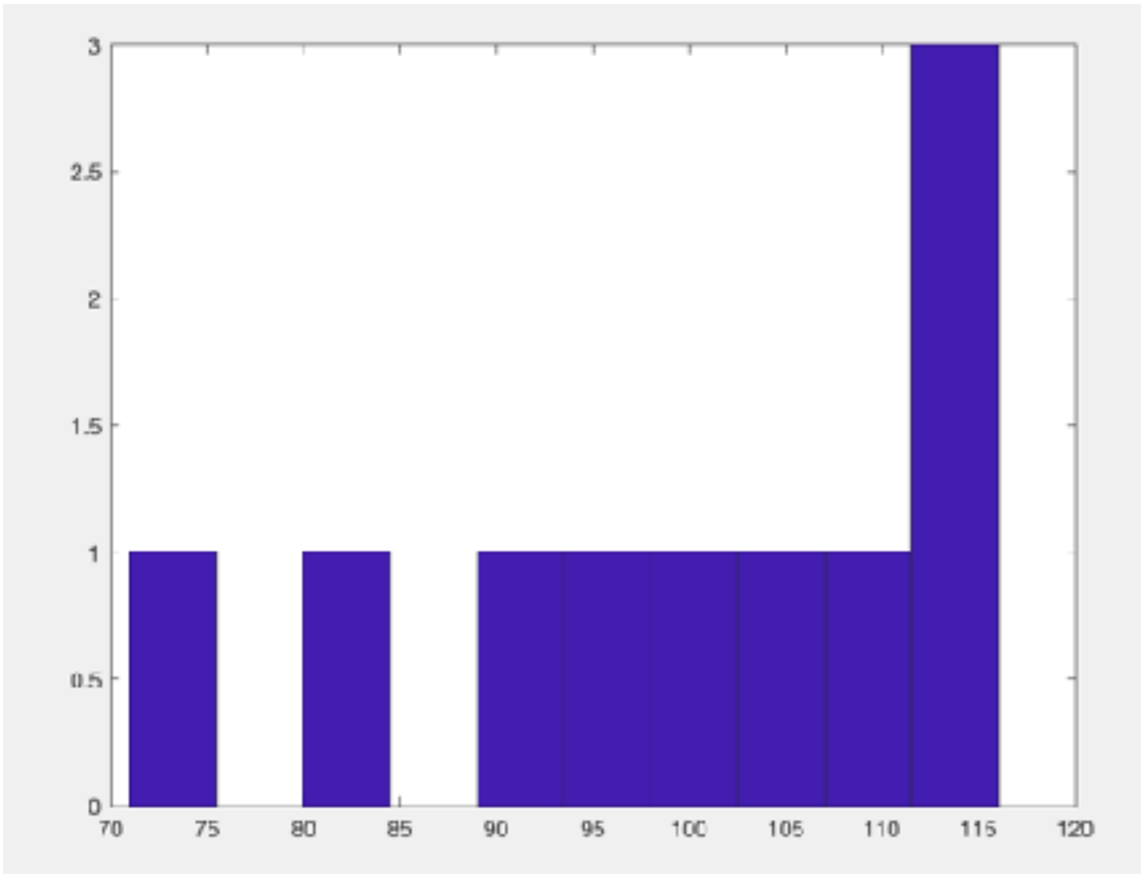
Simulation Methodology:

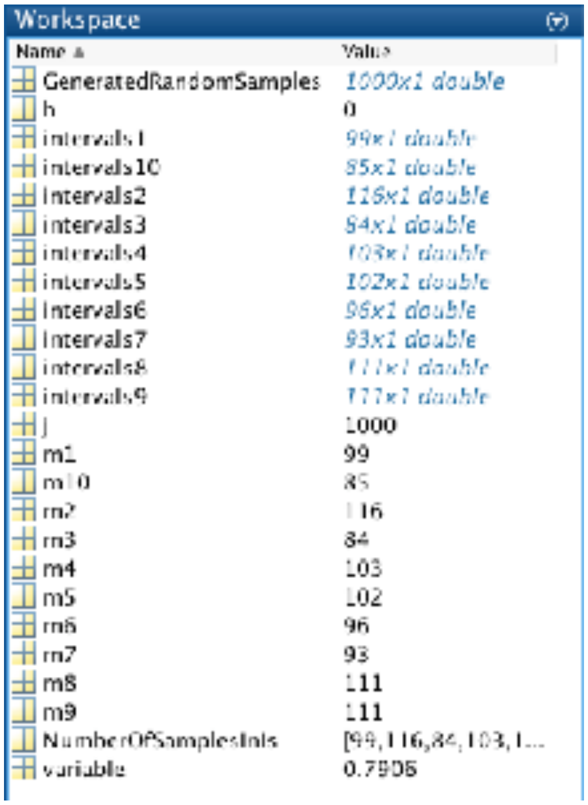
The code to generate 1000 random numbers between 0 and 1 is written and also to divide them in to different segments and add them to their respective segments.
Then a histogram is plotted to verify the desired result

References:

Matlab documentation on Chi-Square Goodness fit test: “chi2gof”
Khan Academy Goodness of fit: https://www.khanacademy.org/math/statistics-probability/inference-categorical-data-chi-square-tests/chi-square-goodness-of-fit-tests/v/pearson-s-chi-square-test-goodness-of-fit

Results:




Workspace

Name ▲	Value
GeneratedRandomSamples	*1000x1 double*
h	0
intervals1	*99x1 double*
intervals10	*85x1 double*
intervals2	*116x1 double*
intervals3	*84x1 double*
intervals4	*103x1 double*
intervals5	*102x1 double*
intervals6	*96x1 double*
intervals7	*93x1 double*
intervals8	*111x1 double*
intervals9	*111x1 double*
j	1000
m1	99
m10	85
m2	116
m3	84
m4	103
m5	102
m6	96
m7	93
m8	111
m9	111
NumberOfSamplesInIs	[99,116,84,103,1...
variable	0.7906

The results are as expected, which shows that each of the sample segments resemble uniform distribution.

Source Code:

```
%Name: Pavan Athreya Narasimha Murthy
%USC ID: 9129210968
%E-mail: pavanatn@usc.edu
%Ph: +1(323)-684 5715
%Term: Fall 2018
%Course: EE511
%Professor: John Silvester

%Clear the Workspace variables and command window for every run
clear all;
clc;

%Fifth portion of Project5

%Lists to have the randoms samples in a segment of 0.1 increments
intervals1 = zeros(1000,1);
intervals2 = zeros(1000,1);
intervals3 = zeros(1000,1);
intervals4 = zeros(1000,1);
intervals5 = zeros(1000,1);
intervals6 = zeros(1000,1);
intervals7 = zeros(1000,1);
intervals8 = zeros(1000,1);
intervals9 = zeros(1000,1);
intervals10 = zeros(1000,1);

%rando samples in ~U[0,1]
GeneratedRandomSamples = rand(1000, 1);

%segregatin the samples
for j = 1:1000
   variable = GeneratedRandomSamples(j,1);
   if variable>0 & variable<=0.1
         intervals1(j) = variable;
   elseif variable>0.1 & variable<=0.2
         intervals2(j) = variable;
   elseif variable>0.2 & variable<=0.3
         intervals3(j) = variable;
   elseif variable>0.3 & variable<=0.4
         intervals4(j) = variable;
   elseif variable>0.4 & variable<=0.5
         intervals5(j) = variable;
   elseif variable>0.5 & variable<=0.6
         intervals6(j) = variable;
   elseif variable>0.6 & variable<=0.7
         intervals7(j) = variable;
   elseif variable>0.7 & variable<=0.8
         intervals8(j) = variable;
   elseif variable>0.8 & variable<=0.9
         intervals9(j) = variable;
   elseif variable>0.9 & variable<=1
```

```
            intervals10(j) = variable;
    end
end

%removing unwanted initialized zeroes in the intervals
intervals1 = intervals1(intervals1 ~= 0);
intervals2 = intervals2(intervals2 ~= 0);
intervals3 = intervals3(intervals3 ~= 0);
intervals4 = intervals4(intervals4 ~= 0);
intervals5 = intervals5(intervals5 ~= 0);
intervals6 = intervals6(intervals6 ~= 0);
intervals7 = intervals7(intervals7 ~= 0);
intervals8 = intervals8(intervals8 ~= 0);
intervals9 = intervals9(intervals9 ~= 0);
intervals10 = intervals10(intervals10 ~= 0);

%calculatig the size of the intervals
[m1,n1] = size(intervals1);
[m2,n2] = size(intervals2);
[m3,n3] = size(intervals3);
[m4,n4] = size(intervals4);
[m5,n5] = size(intervals5);
[m6,n6] = size(intervals6);
[m7,n7] = size(intervals7);
[m8,n8] = size(intervals8);
[m9,n9] = size(intervals9);
[m10,n10] = size(intervals10);

%Plotting a histogram to verify that the samples still resemble uniform
%distribution
NumberOfSamplesInIs = [m1, m2, m3, m4, m5, m6, m7, m8, m9, m10];
hist(NumberOfSamplesInIs);
h=chi2gof(NumberOfSamplesInIs);
disp(h);
```