**Experiment No.: 5**

**Title: Implementation of Uniformity / Independence test**

### Batch: A3 Roll No.: 16010421119 Experiment No.: 5

**Aim:** To implement Kolmogorov -Smirnov test / Chi-square / Runs test to perform uniformity / Independence test of generated random numbers.

**Resources needed:** Turbo C / Java / python

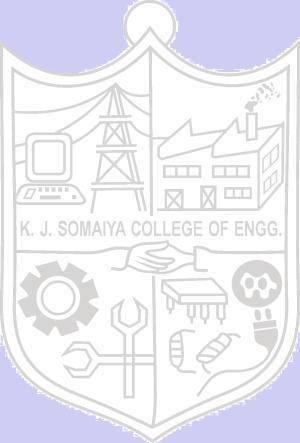
### Theory

**Problem Statement:**

Write function in C / C++ / java / python or macros in MS-excel to implement KolmogorovSmirnov / Chi-square / Runs test.

### Concepts:

Random Numbers generated using a known process or algorithm is called Pseudo random Number.The random numbers generates must possess the property of :

1. Uniformity
2. Independence

### Uniformity :

If the interval (0, 1) is divided into „n‟ classes or subintervals of equal length , the expected number of observations in each interval is N/n, where N is total number of observations.

### Tests for Random numbers

1. **Uniformity Test**

A basic test that is to be performed to validate a new generator is the test of uniformity. Two different testing methods are available, they are

* 1. Kolmogorov- Smirnov Test
  2. Chi-square Test

Both of these measure the degree of agreement between distance of sample of generated random numbers and the theoretical uniform distributions.

1) **Kolmogorov-Smirnov Test:** This test compares the continuous cdf F(x) of the uniform distribution to the empirical cdf SN(x) of sample of N distribution

By definition,

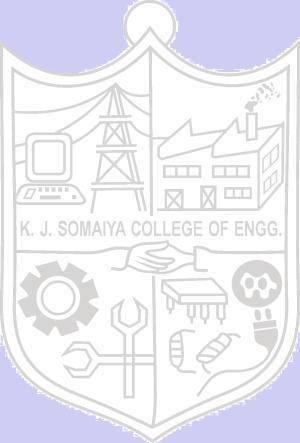
F(x) = x 0 ≤ x ≤ 1

If the sample from random no. generated is R1, R2, … ,RN then the empirical cdf SN(x) is defined as

No. of R1, R2, … ,RN which are x SN(x) =

N

As N becomes larger SN(x) should become better approximation to F(x) provided the null hypothesis is true. The Kolmogorov-Smirnov distance test is best on largest absolute deviation between F(x) & SN(x) over range of random variable.



**2) Chi-square Test**: The chi-square test uses sa mple statistic

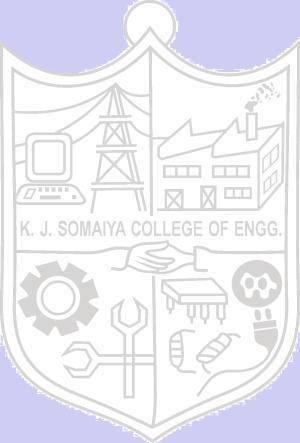
2

O i – E i

(**χ**0)2 = Σ

𝑛

𝑖 =1 E i

Where, Oi = Observed frequency in ith class E = Expected frequency in ith class n = is the no.

of classes i

### Independence:

The probability of observing a value in a particular interval is independent of the previous drawn value.

Each random number R must be an independent sample drawn from a continuous uniform distribution between 0 & 1

i.e.p.d.f. is given by

f(x) = 1 0 ≤ x ≤ 1

0 othe rwise

The expected value of each Ri is given by

E(R) =

And variance is given by

1

0*∫*x dx =[x2/2] = 1/2

1

V(R) = 0*∫*x2 dx = [x3/3 ] = 1/3 – 1/4 = 1/12

Tests for Independence:

These tests are done to check the independence of sequence of random numbers.

### Runs Test

This test analyses an orderly grouping of numbers in a sequence to test the hypothesis of independence. A Run is defined as a succession of similar events preceded and followed by a different. The length of the run is the number of events that occur in the run.

In all cases, actual values are compared with expected values using chi square test.

The Runs test used re:

* 1. Runs Up and Down ii) Runs above and below the mean iii) Runs test for testing length of runs

### Runs Up and Down:

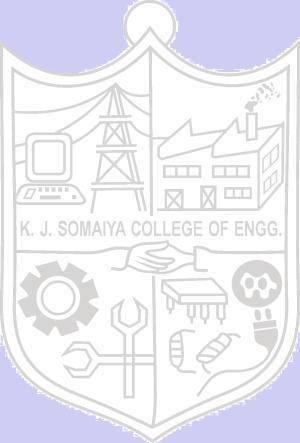
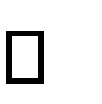
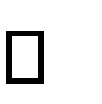
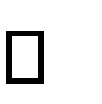
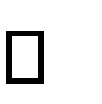
In a sequence of numbers, if a number is followed by a larger number, this is an upward run. Likewise, a number followed by a smaller number is a downstream run. The numbers are given + and – depending on whether they are followed by larger or smaller number. The last number is followed by no event. Eg. 10 numbers there will be 9 +or -. If the numbers are truly random, one would expect to find a certain numbers of runs up and down.

In a sequence of N numbers, a is the total no of runs, the mean and variance is given by the following equation

For N > 20, the distribution of “a” is approximated by a normal distribution, N(0,1). This approximation can be used to test the independence of numbers from a generator. Finally, the standardised normal test statistics ,Zo is developed and compared with critical value

Z 0 = (a - µ) / σ

Where a is total no of runs.



a/2 ≤ Z0 ≤ Z

- Z

/ 2

Z

/ 2

Acceptance region for hypothesis of

independence -Z a/2

/ 2 / 2

1. **Auto correlation Test:** The test for auto correlation is concerned with dependence between numbers in a sequence**.** The test computes auto correlation between every m numbers starting with the ith number.
2. **Gap Test:** The gap test is used to determine the significance of the interval between reoccurrence of the same digit. A gap of length x occurs between reoccurrence of same digit.
3. **Poker Test:** The poker test for independence is based on frequency with which certain digits are repeated in a series of numbers in each case a pair of like digits appear in the numbers that were generated. In 3 digit sample of numbers there are three possibilities which are as follows:
   1. The individual numbers can all be different ii) The individual numbers can all be same iii) There can be one pair of like digits.

### Procedure:

*(Write the algorithm for the test to be implemented and follow the steps given below)*

Steps:

* Implement either Kolmogorov-Smirnov Test or Chi-square Test or Runs test using C / C++ / java or macros in MS-excel
* Generate 5 sample sets (Each set consisting of 100 random numbers) of Pseudo random numbers using Linear Congruential Method.
* Execute the test using all the five sample sets of random numbers as input and using α=0.05.
* Draw conclusions on the acceptance or rejection of the null hypothesis of independence.

Activity:

* 1. Hypothesis
  2. Null Hypothesis
  3. Uniformity of data
  4. Equation of KS Test
  5. Why KS test is called Goodness of Fit Test?

## ANS:

1. Hypothesis: In modeling simulation, a hypothesis is an educated guess about how a system behaves or how certain factors influence its outcomes. For example, if simulating traffic flow, a hypothesis could be that increasing the number of lanes on a highway reduces congestion.
2. Null Hypothesis: The null hypothesis in modeling simulation states that there is no significant effect or difference caused by the factors being studied. For instance, in a simulation testing the impact of different advertising strategies on sales, the null hypothesis might be that there's no difference in sales between the strategies.
3. Uniformity of data: In modeling simulation, uniformity of data refers to the even distribution of data points across different parameters or dimensions. For example, in a weather simulation, uniformity would mean that temperature readings are evenly spread across various locations and times.
4. Equation of KS Test: The equation of the Kolmogorov-Smirnov (KS) test in modeling simulation measures the maximum difference between the cumulative distribution functions of observed data and expected distributions. It helps assess how well the simulated data matches the expected distribution.
5. Why KS test is called Goodness of Fit Test?: The KS test is known as a Goodness of Fit Test in modeling simulation because it evaluates how well the simulated data fits an expected distribution or model. For instance, if simulating the distribution of exam scores based on a normal distribution assumption, the KS test can assess if the simulated scores match the expected normal distribution. If the simulated data closely matches the expected distribution, it indicates a good fit.

**Results: (Program printout with output)**

# Program:

import numpy as np

from scipy.stats import chi2

def generate\_random\_numbers(multiplier, increment, modulus, seed\_value, num\_random\_numbers):

    random\_numbers = []

    current\_value = seed\_value

    for \_ in range(num\_random\_numbers):

        current\_value = (multiplier \* current\_value + increment) % modulus

        random\_numbers.append(round(current\_value / modulus, 2))

    return random\_numbers

def chi\_square\_test(observed\_frequencies, expected\_frequencies, significance\_level, df):

    chi2\_stat = np.sum((observed\_frequencies - expected\_frequencies)\*\*2 / expected\_frequencies)

    critical\_value = chi2.ppf(1 - significance\_level, df)

    return chi2\_stat, critical\_value

def print\_chi\_square\_table\_stepwise(observed\_frequencies, expected\_frequencies, significance\_level, df):

    print("Step-wise Chi-square Table:")

    print("Step | Interval | Observed(O) | Expected(E) | (O-E) | (O- E)^2/E")

    print("-" \* 70)

    for i in range(len(observed\_frequencies)):

        interval = f"{i/10:}-{(i+1)/10:}"

        diff\_observed\_expected = observed\_frequencies[i] - expected\_frequencies[i]

        chi\_square\_step = (diff\_observed\_expected)\*\*2 / expected\_frequencies[i]

        print(f" {i+1:<4} | {interval:^9} | {observed\_frequencies[i]:^11} | {expected\_frequencies[i]:^11} | {diff\_observed\_expected:^5} | {chi\_square\_step:^9.4f}")

    print("-" \* 70)

    chi2\_stat, critical\_value = chi\_square\_test(observed\_frequencies, expected\_frequencies, significance\_level, df)

    print(f"Sample test statistics: {chi2\_stat:.4f}")

    print(f"Critical value (alpha={significance\_level}, df={df}): {critical\_value:.4f}")

    if chi2\_stat <= critical\_value:

        print("Null hypothesis accepted. The observed frequencies indicate uniformity.")

    else:

        print("Null hypothesis rejected. The observed frequencies do not indicate uniformity.")

for i in range(5):

    print(f"\nSet {i+1}:")

    multiplier = int(input("Enter multiplier (a): "))

    increment = int(input("Enter increment (c): "))

    modulus = int(input("Enter modulus (m): "))

    seed\_value = int(input("Enter seed value: "))

    num\_random\_numbers = 100

    significance\_level = 0.05

    df = 9

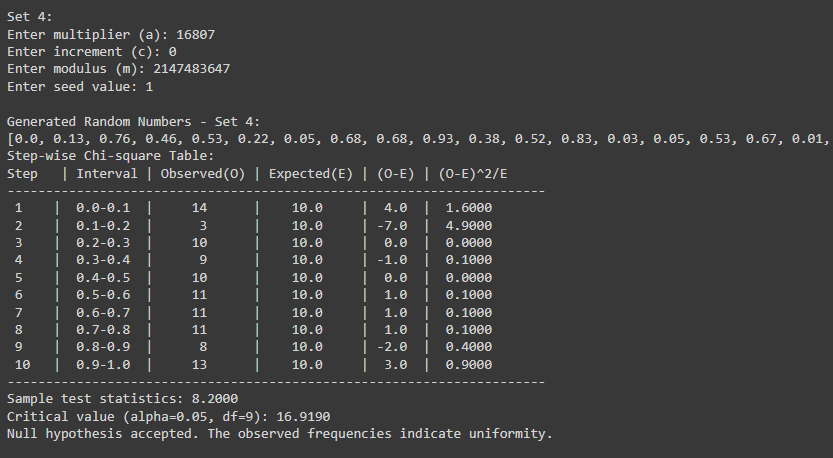
    random\_numbers = generate\_random\_numbers(multiplier, increment, modulus, seed\_value, num\_random\_numbers)

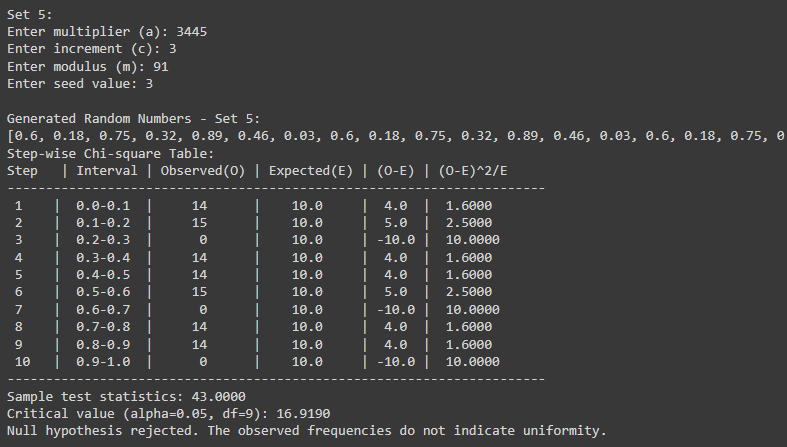
    observed\_frequencies, \_ = np.histogram(random\_numbers, bins=10, range=(0, 1))

    expected\_frequencies = np.full(10, fill\_value=num\_random\_numbers/10)

    print\_chi\_square\_table\_stepwise(observed\_frequencies, expected\_frequencies, significance\_level, df)

# Output:





### Questions:

1. List down the pros and cons of the Kolmogorov - Smirnov test and Chi- Square test.

## ANS:

### Pros and Cons of Kolmogorov-Smirnov Test and Chi-Square Test: Kolmogorov-Smirnov Test:

**Pros:**

* + Suitable for continuous and discrete distributions.
  + It does not require assumptions about the distribution parameters.
  + It's sensitive to differences in both location and shape of the empirical and theoretical distributions.

### Cons:

* + Less powerful than other tests for some distributions.
  + Can be sensitive to outliers in small samples.
  + Requires continuous or ordinal data.

### Chi-Square Test:

**Pros:**

* + Well-understood and widely used for testing goodness-of-fit and independence.
  + Can handle large sample sizes efficiently.
  + Applicable to categorical data.

### Cons:

* + Requires the observations to be independent.
  + Sensitive to sample size; may yield unreliable results with small sample sizes.
  + Assumes that each cell in the contingency table has an expected frequency of at least 5.

1. What is the minimum sample size to apply each of the uniformity and independence tests?

## ANS:

The minimum sample size depends on the specific test being used and the desired level of significance. As a general guideline:

* For the Kolmogorov-Smirnov test, a sample size of at least 20 is recommended for robustness.
* For the Chi-Square test for uniformity, a minimum expected frequency of 5 in each cell of the contingency table is desirable. This typically requires a sample size of at least 100 or more, depending on the number of categories.
* For the Chi-Square test for independence, the sample size should be large enough to ensure that the expected frequency in each cell is at least 5. This often corresponds to a sample size of several hundred or more.

1. Why is it essential to test the random number generator?

## ANS:

* Random number generators (RNGs) are used in various applications, including simulations, cryptography, and statistical sampling.
* Testing RNGs ensures that they produce numbers that are sufficiently random and meet the desired statistical properties.
* Flaws or biases in RNGs can lead to incorrect results or security vulnerabilities in applications that rely on randomness.
* Testing helps identify and mitigate any issues with the RNG algorithm or implementation, ensuring the reliability and integrity of the generated random numbers.

1. List out Chi – Square Test Applications?

## ANS:

* **Goodness-of-fit test:** Determines whether the observed frequency distribution of a variable fits a theoretical distribution.
* **Independence test:** Examines the association between two categorical variables to determine if they are independent.
* **Homogeneity test:** Compares the frequency distributions of two or more groups to determine if they come from the same population.
* **Test of uniformity:** Determines whether observed frequencies within different categories are uniformly distributed.

**Outcomes:**

**CO2: Generate pseudorandom numbers and perform empirical tests to measure the quality of a pseudorandom number generator.**

**Conclusion:**

Successfully implemented Chi square test for uniformity.

### References:

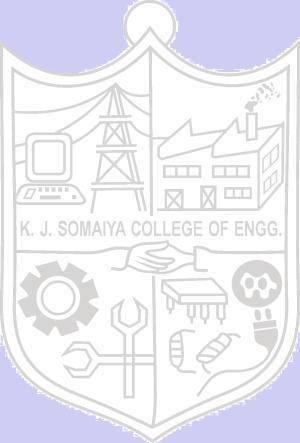
**Books/ Journals/ Websites:**

1. ["Linear Congruential Generators"](http://demonstrations.wolfram.com/LinearCongruentialGenerators/) by Joe Bolte,  [Wolfram Demonstrations Project.](http://en.wikipedia.org/wiki/Wolfram_Demonstrations_Project)
2. Severance, Frank (2001). *System Modeling and Simulation*. John Wiley & Sons, Ltd.

p. 86.  [ISBN](http://en.wikipedia.org/wiki/International_Standard_Book_Number) [0-471-49694-4.](http://en.wikipedia.org/wiki/International_Standard_Book_Number)

1. The GNU C library's *rand()* i[n](http://en.wikipedia.org/wiki/Stdlib.h) [stdlib.h](http://en.wikipedia.org/wiki/Stdlib.h) [uses](http://en.wikipedia.org/wiki/Stdlib.h) a simple (single state) linear congruential generator only in case that the state is declared as 8 bytes. If the state is larger (an array), the generator becomes an additive feedback generator and the period increases. See the [simplified code](http://www.mscs.dal.ca/~selinger/random/) [th](http://www.mscs.dal.ca/~selinger/random/)at reproduces the random sequence from this library.
2. ["A collection of](http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.53.3686&rep=rep1&type=pdf) [selected](http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.53.3686&rep=rep1&type=pdf) [pseudorandom number](http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.53.3686&rep=rep1&type=pdf) [generators with linear structures, K.](http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.53.3686&rep=rep1&type=pdf) [Entacher, 1997".](http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.53.3686&rep=rep1&type=pdf) Retrieved 16 June 2012.
3. ["How](http://support.microsoft.com/kb/231847) [Visual Basic](http://support.microsoft.com/kb/231847) [Generates Pseudo-Random Numbers for](http://support.microsoft.com/kb/231847) [the RND](http://support.microsoft.com/kb/231847) [Function".](http://support.microsoft.com/kb/231847)

*Microsoft Support*. Microsoft. Retrieved 17 June 2011.

1. In spite of documentation on [MSDN,](http://msdn.microsoft.com/en-us/library/bb432429%28VS.85%29.aspx) RtlUniform uses LCG, and not Lehmer's algorithm, implementations before [Windows Vista](http://en.wikipedia.org/wiki/Windows_Vista) [are](http://en.wikipedia.org/wiki/Windows_Vista) flawed, because the result of multiplication is cut to 32 bits, before modulo is applied
2. [GNU Scientific](http://www.gnu.org/software/gsl/manual/html_node/Other-random-number-generators.html) [Library:](http://www.gnu.org/software/gsl/manual/html_node/Other-random-number-generators.html) [Other random number](http://www.gnu.org/software/gsl/manual/html_node/Other-random-number-generators.html) [generators](http://www.gnu.org/software/gsl/manual/html_node/Other-random-number-generators.html)
3. [Novice](http://www.forth.org/novice.html) [Forth library](http://www.forth.org/novice.html)
4. Matsumoto, Makoto, and Takuji Nishimura (1998) ACM Transactions on Modeling and Computer Simulation
5. S.K. Park and K.W. Miller (1988)[.](http://portal.acm.org/citation.cfm?id=63042) ["Random Number](http://portal.acm.org/citation.cfm?id=63042) [Generators: Good](http://portal.acm.org/citation.cfm?id=63042) [Ones Are](http://portal.acm.org/citation.cfm?id=63042) [Hard To](http://portal.acm.org/citation.cfm?id=63042) [Find".](http://portal.acm.org/citation.cfm?id=63042) [*Commun*](http://portal.acm.org/citation.cfm?id=63042)*ications o*[*f*](http://portal.acm.org/citation.cfm?id=63042)[*the*](http://portal.acm.org/citation.cfm?id=63042)[*ACM***31**](http://portal.acm.org/citation.cfm?id=63042)(10): 1192–1201. [doi:10.1145/63039.63042.](http://en.wikipedia.org/wiki/Digital_object_identifier)
6. [D. E.](http://en.wikipedia.org/wiki/Donald_Knuth) [Knuth.](http://en.wikipedia.org/wiki/Donald_Knuth) *The Art of Computer Programming*, Volume 2: *Seminumerical Algorithms*, Third Edition. Addison-Wesley, 1997. ISBN 0-201-89684-2. Section

3.2.1: The Linear Congruential Method, pp. 10–26.

1. P. L'Ecuyer (1999). ["Tables of](http://citeseer.ist.psu.edu/132363.html) [Linear Congruential Generators of](http://citeseer.ist.psu.edu/132363.html) [Different Sizes and](http://citeseer.ist.psu.edu/132363.html) [Good](http://citeseer.ist.psu.edu/132363.html) [Lattice](http://citeseer.ist.psu.edu/132363.html) [Structure".](http://citeseer.ist.psu.edu/132363.html) [*Mathematics of Computati*](http://citeseer.ist.psu.edu/132363.html)[*on***68**](http://citeseer.ist.psu.edu/132363.html)(225): 249–260. [doi:10.1090/S0025-5718-99-0099](http://en.wikipedia.org/wiki/Digital_object_identifier) [6-5.](http://en.wikipedia.org/wiki/Digital_object_identifier)
2. Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007), ["](http://apps.nrbook.com/empanel/index.html#pg%3D343) [Section 7.1.1.](http://apps.nrbook.com/empanel/index.html#pg%3D343) [Some History",](http://apps.nrbook.com/empanel/index.html#pg%3D343) *Numerical Recipes: The Art of Scientific Computing* (3rd ed.), New York: Cambridge University Press,  [ISBN](http://en.wikipedia.org/wiki/International_Standard_Book_Number) [978-0-521-88068-8](http://en.wikipedia.org/wiki/International_Standard_Book_Number)
3. Gentle, James E., (2003). *Random Number Generation and Monte Carlo Methods*, 2nd edition, Springer, ISBN 0-387-00178-6.
4. Joan Boyar (1989).  ["Inferring](http://portal.acm.org/citation.cfm?id=59305&dl=ACM&coll=portal) [sequences produced](http://portal.acm.org/citation.cfm?id=59305&dl=ACM&coll=portal) [by](http://portal.acm.org/citation.cfm?id=59305&dl=ACM&coll=portal) [pseudo-random number](http://portal.acm.org/citation.cfm?id=59305&dl=ACM&coll=portal) [generators".](http://portal.acm.org/citation.cfm?id=59305&dl=ACM&coll=portal)  [*Journal of the*](http://portal.acm.org/citation.cfm?id=59305&dl=ACM&coll=portal)[*ACM***36**](http://portal.acm.org/citation.cfm?id=59305&dl=ACM&coll=portal)(1): 129–141.  [doi:10.1145/58562.59305.](http://en.wikipedia.org/wiki/Digital_object_identifier) (in this paper, efficient algorithms are given for inferring sequences produced by certain pseudo-random number generators).