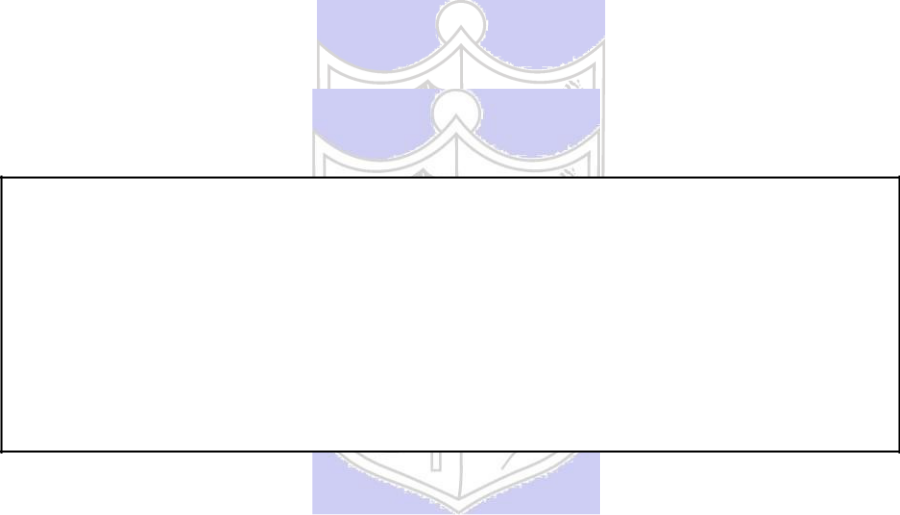
KJSCE/IT/TY BTECH /SEMVI/SM/2021-22



**Experiment No.: 2**

**Title: Implementation of Uniformity test**

**Autonomous College Affiliated to University of Mumbai)**

KJSCE/IT/TY BTECH /SEMVI/SM/2021-22

**Batch:A4** **Roll No.:1914078** **Experiment No.: 2**

**Aim:** To implement Kolmogorov –Smirnov (K S) test / Chi-square test on the random number generator implemented in experiment no 1 for uniformity testing.

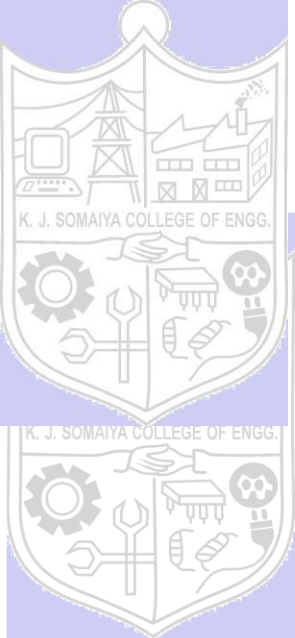
**Resources needed:** Turbo C / Java / python

**Theory**

**Problem Statement:**

Write function in C / C++ / java / python or macros in MS-excel to implement Kolmogorov-Smirnov ( KS) / Chi-square 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

1. 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

**Autonomous College Affiliated to University of Mumbai)**

KJSCE/IT/TY BTECH /SEMVI/SM/2021-22

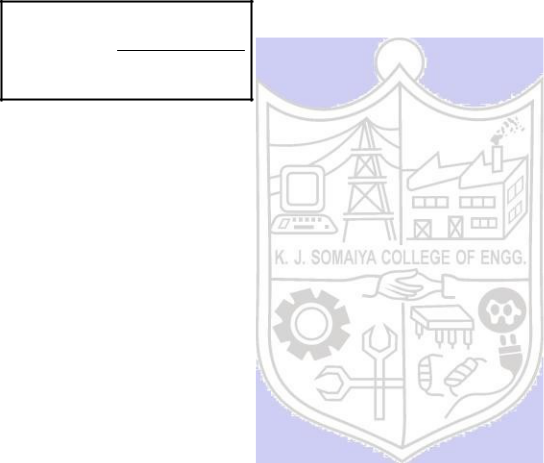
As N becomes larger SN(x) should become better approximation to F(x) provided the null

hypothesis is true. The Kolmogorov-Smirnov distance test is best on largest absolute deviation between F(x) & SN(x) over range of random variable.

**b)) Chi square test:** The Chi square test sample test statistics is:

* 2*n* (*Oi* *Ei* )2

0*i*1*Ei*



Where, Oi = Observed frequency in ith class Ei = Expected frequency in ith class n = is the no. of classes

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Procedure:**

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

Steps:

* Make a null hypothesis for uniformity

 Generate 5 sample sets (Each set consisting of 100 random numbers) of Pseudo random numbers using Linear Congruential Method implemented in expt 1

* Implement either Kolmogorov-Smirnov Test or Chi-square Test
  + 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 uniformity.

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

# X i+1= (a X i + c) mod m

def linearCongruenceGenerator(seed, a, c, m):

    Xn = None

    Xn\_1 = seed

    randomNumbers = []

    while(Xn != seed):

        randomNumbers.append(round(Xn\_1/m, 4))

        Xn = (Xn\_1\*a + c) % m

        Xn\_1 = Xn

    n = len(randomNumbers)

    density = (max(randomNumbers)-min(randomNumbers))/n

    return randomNumbers, n, density

#int(input("Seed Value: "))

seed = 1

#int(input("Multiplier: "))

a = 13

#int(input("Inceremnt: "))

c = 1

#int(input("Modulus: "))

m = 256

randomList, Period, Density = linearCongruenceGenerator(seed, a, c, m)

print("Random numbers: ", end=" ")

for i in range(len(randomList)):

    if i % 10 == 0:

        print()

    print(randomList[i], end=" ")

print("\nPeriod: ", Period)

print("Density: ", Density)

N = 100

n = 10

Ei = N/n

final = []

# defining the classes

Oi = [0]\*n

# Selecting the first 100 numbers

testList = randomList[:N]

# sorting the list

testList.sort()

print("Test List: ")

for i in range(len(testList)):

    if i % 10 == 0:

        print()

    print(testList[i], end=" ")

for number in testList:

    Oi[int(number//0.1)] += 1

for each in Oi:

    final.append((each-Ei)\*\*2/Ei)

Chi\_o = sum(final)

print(f"\nChi\_o: {Chi\_o}")

# float(input("Enter Chi apliha value for 0.05: "))

Chi\_aplha = 16.9

if(Chi\_aplha > Chi\_o):

    print("H\_o is accepted")

else:

    print("H\_o is rejected")

Sample sets:

1. X0 = 1, a = 13, c=1, m =256



**Questions:**

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

**Kolmogorov-Smirnov Test:**

Advantages include:

* The test is distribution free. That means you don’t have to know the underlying population distribution for your data before running this test.
* The D statistic (not to be confused with Cohen’s D) used for the test is easy to calculate.
* It can be used as a goodness of fit test following regression analysis.
* There are no restrictions on sample size; Small samples are acceptable.
* Tables are readily available.

Disadvantages:

* In order for the test to work, you must specify the location, scale, and shape parameters. If these parameters are estimated from the data, it invalidates the test. If you don’t know these parameters, you may want to run a less formal test (like the one outlined in the empirical distribution function article).
* It generally can’t be used for discrete distributions, especially if you are using software (most software packages don’t have the necessary extensions for discrete K-S Test and the manual calculations are convoluted).
* Sensitivity is higher at the center of the distribution and lower at the tails.

**Chi square test:**

Advantages include:

* One of the largest strengths of chi-square is that it is easier to compute than some statistics.
* Also it can be used with data that has been measured on a categorical scale.

Disadvantages:

* One of the limitations is that all participants measured must be independent, meaning that an individual cannot fit in more than one category.
* Another limitation with using chi-square is that the data must be frequency data.

1. What is the minimum sample size to apply each of the uniformity and independence tests?
2. Why is it essential to test the random number generator?

It is necessary to test random numbers because the random numbers we generate are pseudo random numbers and not real and pseudo random number generator must generate a sequence of such random numbers which are uniformly distributed and they should not be correlated, they should not repeat itself.

**Autonomous College Affiliated to University of Mumbai**

KJSCE/IT/TY BTECH /SEMVI/SM/2021-22



**Outcomes:**

Generate pseudorandom numbers and perform empirical tests to measure the quality of a pseudo random number generator.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Conclusion:**

Implemented Chi square test to test the random numbers generated using

1. Xo = 53, a=5, c=19, m = 256
   1. Period = 256, Density = 0.0195312
   2. Chi square = 5.8
   3. Null hypothesis is accepted
2. Xo = 123, a=5, c=0, m = 512
   1. Period = 128, Density = 0.03125
   2. Chi square = 1.8
   3. Null hypothesis is accepted
3. Xo = 35412, a=27753, c=18751, m = 65536
   1. Period = 65536, Density = 0.24794
   2. Chi square = 9.6
   3. Null hypothesis is accepted
4. Xo = 667, a=435, c=65, m = 4096
   1. Period = 2048, Density = 0.0795898
   2. Chi square = 5
   3. Null hypothesis is accepted
5. Xo = 347, a = 699, c = 0, m = 1024
   1. Period = 256, Density = 0.523438
   2. Chi square = 6
   3. Null hypothesis is accepted

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of faculty in-charge with date**

**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 0-471-49694-4.](http://en.wikipedia.org/wiki/International_Standard_Book_Number)
3. The GNU C library's *rand()* [in 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/) that reproduces the random sequence from this library.
4. ["A collection of selected pseudorandom number 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.
5. ["How Visual Basic Generates Pseudo-Random Numbers for the RND Function".](http://support.microsoft.com/kb/231847)
6. 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 flawed, because the result of multiplication is cut to 32 bits, before modulo is applied

**(Autonomous College Affiliated to University of Mumbai)**

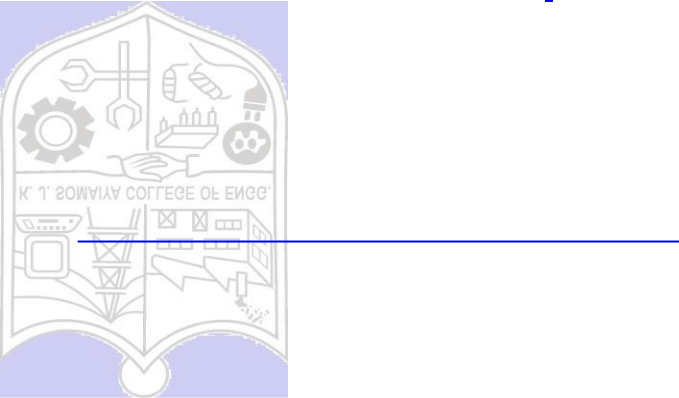
KJSCE/IT/TY BTECH /SEMVI/SM/2021-22

1. [GNU Scientific Library: Other random number generators](http://www.gnu.org/software/gsl/manual/html_node/Other-random-number-generators.html)
2. [Novice Forth library](http://www.forth.org/novice.html)
3. Matsumoto, Makoto, and Takuji Nishimura (1998) ACM Transactions on Modeling
4. S.K. Park and K.W. Miller (1988)[. "Random Number Generators: Good Ones Are](http://portal.acm.org/citation.cfm?id=63042)

[Hard To Find". *Communications of the 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)

1. [D. E. 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.
2. P. L'Ecuyer (1999). ["Tables of Linear Congruential Generators of Different Sizes and](http://citeseer.ist.psu.edu/132363.html) [Good Lattice Structure". *Mathematics of Computation***68**](http://citeseer.ist.psu.edu/132363.html) (225): 249–260. [doi:10.1090/S0025-5718-99-00996-5.](http://en.wikipedia.org/wiki/Digital_object_identifier)
3. Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007), ["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 978-0-521-88068-8](http://en.wikipedia.org/wiki/International_Standard_Book_Number)
4. Gentle, James E., (2003). *Random Number Generation and Monte Carlo Methods*, 2nd edition, Springer, ISBN 0-387-00178-6.
5. Joan Boyar (1989). ["Inferring sequences produced by pseudo-random number](http://portal.acm.org/citation.cfm?id=59305&dl=ACM&coll=portal) [generators". *Journal of the 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).



**(Autonomous College Affiliated to University of Mumbai)**

KJSCE/IT/TY BTECH /SEMVI/SM/2021-22

**(Autonomous College Affiliated to University of Mumbai)**