

**Experiment No.: 3**

**Title: Implementation of Independence test**

# Batch: A4 Roll No.: 1914078 Experiment No.: 3

**Aim:** To implement Autocorrelation test / Runs test to perform Independence test of generated random numbers.

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

# Theory

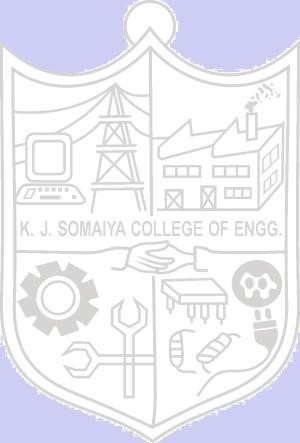
**Problem Statement:**

Write function in C / C++ / java / python or macros in MS-excel to implement Autocorrelation

/ 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.

# 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

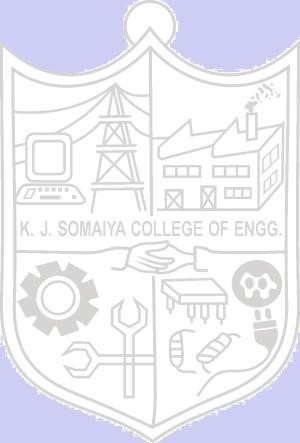
0 otherwise

The expected value of each Ri is given by

E(R) = 1

1 0

0*∫*x dx = [x2/2 ] = 1/2 And variance is given by

1

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

# Tests for Random numbers 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 otherwise

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
  2. Runs above and below the mean
  3. Runs test for testing length of runs

# Runs Up and Down:

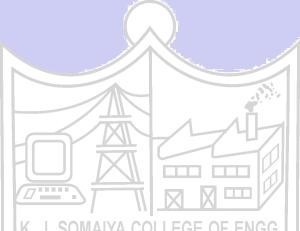
In a sequence of numbers, if a number is followed by a larger number, this is an upward run. Likewise, a number followed by a smaller number is a downstream run. The numbers are given + and – depending on whether they are followed by larger or smaller number. The last number is followed by no event. 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 anddown.

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.

Acceptance region for hypothesis of independence -Za/2 ≤ Z0 ≤ Za/2

**Autonomou sity of Mumbai)**

**s College Affiliated to Univer**

 / 2  / 2

-Z / 2 Z / 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. Thus autocorrelation limit between following numbers would be of interest.

*Ri , Ri+m , Ri+2m , Ri+(M+1)m*

where *M* is the largest integer such that *i+(M+1)m* ≤ N where N is total number of values in the sequence.

Since the nonzero autocorrelation implies a lack of independence, the following test is appropriate:

*H* 0 :

*H*1 :

*im*  0,

*im*  0,

if numbers are independent if numbers are dependent

For large values of M, the distribution of the estimator of ρ*im*, denoted**ˆ*im*

normal,if the values *Ri , Ri+m , Ri+2m , Ri+(M+1)m* are uncorrelated. The test statistics is

is approximately

*Z* 0 

**ˆ*im*

**ˆ **ˆ

*im*

with a mean of 0 and variance of 1,under the assumption of independence , for large M.

If -Zα/2 ≤ Z0 ≤ Zα/2 , H0 is not rejecte for the significance level α .

1. **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.
2. **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:
3. The individual numbers can all be different
4. The individual numbers can all be same
5. There can be one pair of likedigits.

# Procedure:

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

Steps:

* Implement either Autocorrelation 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.

# Results:

# Code:

from random import random

import math

def display(x, len):

    for i in range(0, len):

        if(i % 10 == 0 and i != 0):

            print(x[i])

        else:

            print(x[i], end = ", ")

def lcg(x0, a, c, m, len, random\_numbers):

    count = 1       #check number of random numbers

    integers = [0] \* len    #to store Xi values

    integers[0] = x0

    random\_numbers[0] = integers[0] / m     #storing the first random number

    for i in range(1, len):

        integers[i] = ((integers[i - 1] \* a) + c) % m       #formula

        random\_numbers[i] = integers[i] / m

        count = count + 1

    #printing the array

    print("Random Numbers: ")

    display(random\_numbers, len)

    #to check if the numbers are repeated

    for j in range(0, i):

        if(random\_numbers[i] == random\_numbers[j]):

            print("repeated")

    print("\nCount = ", count)

def auto\_correlation(random\_numbers, i, m):

    N = 100

    M = int((N - i) / m - 1) #from formula i+(M+1)m <= N

    sum = 0

    print()

    for k in range(0, M+1):

        if((i+(k+1)\*m) < 100):

            temp = random\_numbers[i+k\*m] \* random\_numbers[i+(k+1)\*m]

            print(random\_numbers[i+k\*m], end=", ")

            print(random\_numbers[i+(k+1)\*m])

            sum = sum + temp

    print("\nSum = ", sum)

    print("M = ", M)

    rho = sum / (M + 1) - 0.25

    print("rho = ", rho)

    sigma = (math.sqrt(13\*M + 7)) / (12\*(M+1))

    print("sigma = ", sigma)

    z\_alpha = 1.96

    z0 = rho / sigma

    print("\nz0 = rho / sigma = ", z0)

    if(-z\_alpha <= z0 and z0 <= z\_alpha):

        print("z-alpha = ", z\_alpha)

        print("H0 is accepted.")

    else:

        print("H0 is rejected.")

def main():

    print("\nHypothesis H0: The pseudo random numbers generated are independent.")

    # TEST CASE 1

    print("\nTEST CASE 1: ")

    x0 = 17

    a = 21      # a = 5+8k

    c = 0

    m = 512     # m = 2^9 => period = m/4 = 128

    len = 100      #number of random numbers

    random\_numbers = [0] \* len

    i = 1

    M = 8

    lcg(x0, a, c, m, len, random\_numbers)

    auto\_correlation(random\_numbers, i, M)

    # TEST CASE 2

    print("\n\nTEST CASE 2: ")

    x0 = 23

    a = 9      # a = 1+4k

    c = 31

    m = 128     # m = 2^7 => period = m = 128

    len = 100      #number of random numbers

    random\_numbers = [0] \* len

    i = 1

    M = 7

    lcg(x0, a, c, m, len, random\_numbers)

    auto\_correlation(random\_numbers, i, M)

    # TEST CASE 3

    print("\n\nTEST CASE 3: ")

    x0 = 41

    a = 29      # a = 5+8k

    c = 0

    m = 512     # m = 2^9 => period = m/4 = 128

    len = 100      #number of random numbers

    random\_numbers = [0] \* len

    i = 4

    M = 6

    lcg(x0, a, c, m, len, random\_numbers)

    auto\_correlation(random\_numbers, i, M)

    # TEST CASE 4

    print("\n\nTEST CASE 4: ")

    x0 = 29

    a = 13      # a = 1+4k

    c = 41

    m = 128     # m = 2^7 => period = m = 128

    len = 100      #number of random numbers

    random\_numbers = [0] \* len

    i = 2

    M = 5

    lcg(x0, a, c, m, len, random\_numbers)

    auto\_correlation(random\_numbers, i, M)

    # TEST CASE 5

    print("\n\nTEST CASE 5: ")

    x0 = 7

    a = 17      # a = 1+4k

    c = 27

    m = 128     # m = 2^7 => period = m = 128

    len = 100      #number of random numbers

    random\_numbers = [0] \* len

    i = 3

    M = 4

    lcg(x0, a, c, m, len, random\_numbers)

    auto\_correlation(random\_numbers, i, M)

main()

# Output:

# 

# 

# 

# 

# 

**Questions:**

1. Give an example and interpret the need of Independence test.

**Answer:**

The random numbers generated using LCM are pseudo random numbers. These numbers cannot be considered as true random numbers. Therefore, it is necessary to check the randomness of these numbers, which is done by carrying out uniformity and independence tests on them. Independence tests are used to check if the random numbers are correlated in any manner.

Example:

45, **99**, 26, 0, 79, **96**, 6, 34, 64, **98**, 84, 2, 19, **93**, 49

The above numbers appear to be random, but we can see that the 2nd, 6th, 10th and 14th numbers are large numbers. This means that every 4th number after the 2nd number is large. This hints that the random numbers generated are somehow correlated.

Independence tests are required to find such correlation between the numbers. If the random number generator passes these tests, along with uniformity tests, then it can be considered as acceptable.

1. What is Type 1 and Type 2 error?

**Answer:**

**Type I error** is a false positive conclusion. A Type I error means rejecting the null hypothesis when it’s actually true. It means concluding that results are statistically significant when, in reality, they came about purely by chance or because of unrelated factors.

**Type II error** is a false negative conclusion. A Type II error means not rejecting the null hypothesis when it’s actually false. This is not quite the same as accepting the null hypothesis, because hypothesis testing can only tell you whether to reject the null hypothesis.

1. What of the independence tests make use of Chi square test?

**Answer:**

The Chi square test is used to check whether 2 variables are likely to be related or not, in the

runs test we use the chi square test to chi square test to determine whether the random numbers are independent or not.

**Outcomes:**

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

# Conclusion:

# In this experiment, we successfully carried out auto-correlation independence test on pseudo random numbers generated using the linear congruential method for five different test cases.

# Hypothesis H0: The pseudo random numbers generated are independent.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sr. No | X0 | a | c | m | i | m(test) | Result |
| 1 | 17 | 21 | 0 | 512 | 1 | 8 | H0 is accepted |
| 2 | 23 | 9 | 31 | 128 | 1 | 7 | H0 is accepted |
| 3 | 41 | 29 | 0 | 512 | 4 | 6 | H0 is accepted |
| 4 | 29 | 13 | 41 | 128 | 2 | 5 | H0 is accepted |
| 5 | 7 | 17 | 27 | 128 | 3 | 4 | H0 is accepted |

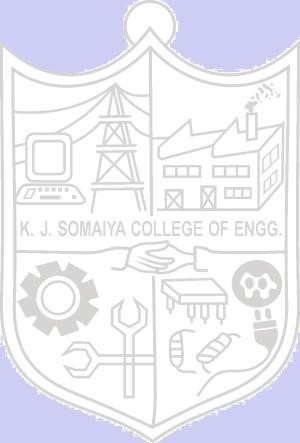
# Grade: AA / AB / BB / BC / CC / CD /DD

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

**References:**

**Books/ Journals/ Websites:**

.

* 1. Jerry Banks, Jerry Banks, John Carson, Barry Nelson, and David M. Nicol; “Discrete EventSystem System Simulation”, Third Edition, Pearson Education
  2. "[Linear Congruential Generators"](http://demonstrations.wolfram.com/LinearCongruentialGenerators/) by Joe Bolte[, Wolfram Demonstrations Project.](http://en.wikipedia.org/wiki/Wolfram_Demonstrations_Project)
  3. 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)
  4. ["A collection of selected pseudorandom number generators with linear structures, K. Entacher, 1997".](http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.53.3686&rep=rep1&type=pdf) Retrieved 16 June 2012.
  5. [GNU Scientific Library: Other random number generators](http://www.gnu.org/software/gsl/manual/html_node/Other-random-number-generators.html)
  6. [Novice Forth library](http://www.forth.org/novice.html)
  7. Matsumoto, Makoto, and Takuji Nishimura (1998) ACM Transactions on Modeling and Computer Simulation
  8. S.K. Park and K.W. Miller (1988)[. "Random Number Generators: Good Ones Are 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)
  9. [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.
  10. P. L'Ecuyer (1999). ["Tables of Linear Congruential Generators of Different Sizes and 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)
  11. Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007), ["Section 7.1.1. 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)
  12. Gentle, James E., (2003). *Random Number Generation and Monte Carlo Methods*, 2nd edition, Springer, ISBN 0-387-00178-6.
  13. Joan Boyar (1989). ["Inferring sequences produced by pseudo-random number 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).