Experiment No.: 6
Title: Implementation of Uniformity / Independence test

Batch: B2 **Roll No.:16010421059** **Experiment No.:6**

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

a. Kolmogorov- Smirnov Test
b. 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 $S_N(x)$ of sample of N distribution

By definition,

$$F(x) = x \qquad 0 \leq x \leq 1$$

If the sample from random no. generated is R_1, R_2, … ,R_N then the empirical cdf $S_N(x)$ is defined as

$$S_N(x) = \frac{\text{No. of } R_1, R_2, \ldots, R_N \text{ which are } x}{N}$$

As N becomes larger $S_N(x)$ 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) & $S_N(x)$ over range of random variable.

2) Chi-square Test: The chi-square test uses sample statistic

$$(\chi_0)^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

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

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

$$f(x) = \begin{cases} 1 & 0 \le x \le 1 \\ 0 & \text{othe rwise} \end{cases}$$

The expected value of each Ri is given by

$$E(R) = \int_0^1 x\,dx = [x^2/2] = 1/2$$

And variance is given by

$$V(R) = \int_0^1 x^2\,dx = [x^3/3] = 1/3 - 1/4 = 1/12$$

Tests for Independence:

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

1) 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:

i) 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

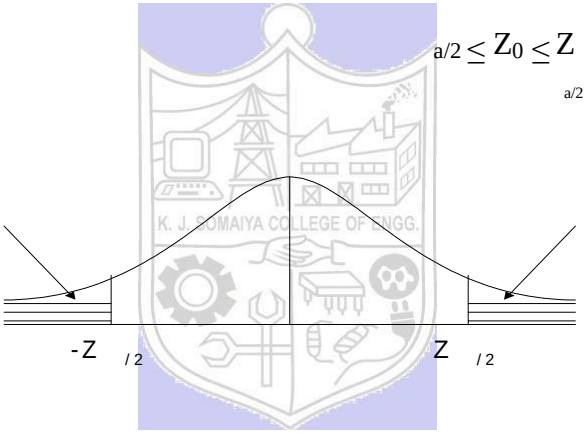
$$\mu = \frac{(2N-1)}{3} \qquad \sigma^2 = \frac{(16N-29)}{90}$$

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

$$Z_0 = (a - \mu) / \sigma$$

Where a is total no of runs.

Acceptance region for hypothesis of independence $-Z_{a/2} \leq Z_0 \leq Z_{a/2}$



2) 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.

3) 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.

4)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:

i) 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?

Results: (Program printout with output)

Code:-

```
import numpy as np
import scipy.stats as stats

class LinearCongruentialGenerator:
```

```
    def __init__(self, seed, a, c, m):
        self.seed = seed
        self.a = a
        self.c = c
        self.m = m

    def generate_uniform(self, n):
        random_numbers = []
        xn = self.seed
        for _ in range(n):
            xn = (self.a * xn + self.c) % self.m
            random_numbers.append(round(xn / self.m, 2))
        return random_numbers

    def generate_non_uniform(self, n):
        random_numbers = []
        xn = self.seed
        for _ in range(n):
            xn = (self.a * xn + self.c) % self.m
            # Introduce non-uniformity
            if xn % 2 == 0:
                random_numbers.append(round((xn / self.m) * 0.2, 2))
            else:
                random_numbers.append(round(xn / self.m, 2))
        return random_numbers

def perform_chi_square_test(data):
    observed, _ = np.histogram(data, bins=10, range=(0, 1))
    expected = np.full_like(observed, fill_value=10)
    chi2_stat, p_val = stats.chisquare(observed, f_exp=expected)
    return observed, expected, chi2_stat, p_val

def display_matrix(data):
    matrix = np.array(data).reshape(10, 10)
    print("Random Numbers Matrix:")
    print(matrix)

def main():
    seed = 12345
    a = 1103515245
    c = 12345
    m = 2**31
    generator = LinearCongruentialGenerator(seed, a, c, m)
    alpha = 0.05
    for i in range(5):
        if i == 3:
            random_set = generator.generate_non_uniform(100)
        else:
            random_set = generator.generate_uniform(100)
        print(f"Sample Set {i+1}:")
        display_matrix(random_set)  # Display 10x10 matrix
        print("Interval\tOi\t\tEi\t\t(Oi - Ei)^2 / Ei")
        observed, expected, chi2_stat, p_val = perform_chi_square_test(random_set)
```

```
    for interval, oi, ei in zip(range(1, 11), observed, expected):
        oi_ei_squared_over_ei = ((oi - ei) ** 2) / ei
        print(f"{interval}\t\t{oi}\t\t{ei}\t\t{oi_ei_squared_over_ei}")
    # Perform hypothesis test
    dof = len(observed) - 1
    critical_value = stats.chi2.ppf(1 - alpha, dof)
    print("\nHypothesis Test:")
    print(f"Chi-square statistic: {chi2_stat}")
    print(f"Critical value (alpha = {alpha}): {critical_value}")
    if chi2_stat < critical_value:
        print("Null hypothesis accepted. The numbers are likely generated from a uniform distribution.")
    else:
        print("Null hypothesis rejected. The numbers are not generated from a uniform distribution.")
    print()

if __name__ == "__main__":
    main()
```

Output:

```
Sample Set 1:
Random Numbers Matrix:
[[0.66 0.3  0.67 0.11 0.52 0.49 0.6  0.37 0.26 0.37]
 [0.83 0.17 0.3  0.64 0.79 0.99 0.8  0.46 0.54 0.63]
 [0.25 0.7  0.72 0.98 0.33 0.45 0.71 0.74 0.17 0.02]
 [0.78 0.04 0.6  0.25 0.56 0.52 0.4  0.18 0.65 0.72]
 [0.3  0.97 0.83 0.39 0.7  0.13 0.61 0.55 0.7  0.9 ]
 [0.4  0.83 0.62 0.02 0.37 0.11 0.56 0.37 0.15 0.8 ]
 [0.07 0.09 0.2  0.9  0.55 0.49 0.38 0.1  0.45 0.31]
 [0.38 0.12 0.95 0.79 0.43 0.57 0.02 0.96 0.16 0.35]
 [0.75 0.72 0.83 0.18 0.7  0.05 0.99 0.03 0.16 0.99]
 [0.68 0.04 0.24 0.64 0.16 0.91 0.44 0.86 0.5  0.16]]
Interval        Oi              Ei              (Oi - Ei)^2 / Ei
1               9               10              0.1
2               14              10              1.6
3               8               10              0.4
4               10              10              0.0
5               9               10              0.1
6               11              10              0.1
7               13              10              0.9
8               9               10              0.1
9               7               10              0.9
10              10              10              0.0

Hypothesis Test:
Chi-square statistic: 4.2
Critical value (alpha = 0.05): 16.918977604620448
Null hypothesis accepted. The numbers are likely generated from a uniform distribution.
```

```
Sample Set 2:
Random Numbers Matrix:
[[0.66 0.3  0.67 0.11 0.52 0.49 0.6  0.37 0.26 0.37]
 [0.83 0.17 0.3  0.64 0.79 0.99 0.8  0.46 0.54 0.63]
 [0.25 0.7  0.72 0.98 0.33 0.45 0.71 0.74 0.17 0.02]
 [0.78 0.04 0.6  0.25 0.56 0.52 0.4  0.18 0.65 0.72]
 [0.3  0.97 0.83 0.39 0.7  0.13 0.61 0.55 0.7  0.9 ]
 [0.4  0.83 0.62 0.02 0.37 0.11 0.56 0.37 0.15 0.8 ]
 [0.07 0.09 0.2  0.9  0.55 0.49 0.38 0.1  0.45 0.31]
 [0.38 0.12 0.95 0.79 0.43 0.57 0.02 0.96 0.16 0.35]
 [0.75 0.72 0.83 0.18 0.7  0.05 0.99 0.03 0.16 0.99]
 [0.68 0.04 0.24 0.64 0.16 0.91 0.44 0.86 0.5  0.16]]
Interval        Oi              Ei              (Oi - Ei)^2 / Ei
1               9               10              0.1
2               14              10              1.6
3               8               10              0.4
4               10              10              0.0
5               9               10              0.1
6               11              10              0.1
7               13              10              0.9
8               9               10              0.1
9               7               10              0.9
10              10              10              0.0

Hypothesis Test:
Chi-square statistic: 4.2
Critical value (alpha = 0.05): 16.918977604620448
Null hypothesis accepted. The numbers are likely generated from a uniform distribution.
```

```
Sample Set 3:
Random Numbers Matrix:
[[0.66 0.3  0.67 0.11 0.52 0.49 0.6  0.37 0.26 0.37]
 [0.83 0.17 0.3  0.64 0.79 0.99 0.8  0.46 0.54 0.63]
 [0.25 0.7  0.72 0.98 0.33 0.45 0.71 0.74 0.17 0.02]
 [0.78 0.04 0.6  0.25 0.56 0.52 0.4  0.18 0.65 0.72]
 [0.3  0.97 0.83 0.39 0.7  0.13 0.61 0.55 0.7  0.9 ]
 [0.4  0.83 0.62 0.02 0.37 0.11 0.56 0.37 0.15 0.8 ]
 [0.07 0.09 0.2  0.9  0.55 0.49 0.38 0.1  0.45 0.31]
 [0.38 0.12 0.95 0.79 0.43 0.57 0.02 0.96 0.16 0.35]
 [0.75 0.72 0.83 0.18 0.7  0.05 0.99 0.03 0.16 0.99]
 [0.68 0.04 0.24 0.64 0.16 0.91 0.44 0.86 0.5  0.16]]
Interval        Oi              Ei              (Oi - Ei)^2 / Ei
1               9               10              0.1
2               14              10              1.6
3               8               10              0.4
4               10              10              0.0
5               9               10              0.1
6               11              10              0.1
7               13              10              0.9
8               9               10              0.1
9               7               10              0.9
10              10              10              0.0

Hypothesis Test:
Chi-square statistic: 4.2
Critical value (alpha = 0.05): 16.918977604620448
Null hypothesis accepted. The numbers are likely generated from a uniform distribution.
```

```
Sample Set 4:
Random Numbers Matrix:
[[0.13 0.3  0.13 0.11 0.1  0.49 0.12 0.37 0.05 0.37]
 [0.17 0.17 0.06 0.64 0.16 0.99 0.16 0.46 0.11 0.63]
 [0.05 0.7  0.14 0.98 0.07 0.45 0.14 0.74 0.03 0.02]
 [0.16 0.04 0.12 0.25 0.11 0.52 0.08 0.18 0.13 0.72]
 [0.06 0.97 0.17 0.39 0.14 0.13 0.12 0.55 0.14 0.9 ]
 [0.08 0.83 0.12 0.02 0.07 0.11 0.11 0.37 0.03 0.8 ]
 [0.01 0.09 0.04 0.9  0.11 0.49 0.08 0.1  0.09 0.31]
 [0.08 0.12 0.19 0.79 0.09 0.57 0.   0.96 0.03 0.35]
 [0.15 0.72 0.17 0.18 0.14 0.05 0.2  0.03 0.03 0.99]
 [0.14 0.04 0.05 0.64 0.03 0.91 0.09 0.86 0.1  0.16]]
Interval        Oi              Ei              (Oi - Ei)^2 / Ei
1               29              10              36.1
2               36              10              67.6
3               3               10              4.9
4               6               10              1.6
5               4               10              3.6
6               3               10              4.9
7               4               10              3.6
8               4               10              3.6
9               3               10              4.9
10              8               10              0.4

Hypothesis Test:
Chi-square statistic: 131.2
Critical value (alpha = 0.05): 16.918977604620448
Null hypothesis rejected. The numbers are not generated from a uniform distribution.
```

```
Sample Set 5:
Random Numbers Matrix:
[[0.66 0.3  0.67 0.11 0.52 0.49 0.6  0.37 0.26 0.37]
 [0.83 0.17 0.3  0.64 0.79 0.99 0.8  0.46 0.54 0.63]
 [0.25 0.7  0.72 0.98 0.33 0.45 0.71 0.74 0.17 0.02]
 [0.78 0.04 0.6  0.25 0.56 0.52 0.4  0.18 0.65 0.72]
 [0.3  0.97 0.83 0.39 0.7  0.13 0.61 0.55 0.7  0.9 ]
 [0.4  0.83 0.62 0.02 0.37 0.11 0.56 0.37 0.15 0.8 ]
 [0.07 0.09 0.2  0.9  0.55 0.49 0.38 0.1  0.45 0.31]
 [0.38 0.12 0.95 0.79 0.43 0.57 0.02 0.96 0.16 0.35]
 [0.75 0.72 0.83 0.18 0.7  0.05 0.99 0.03 0.16 0.99]
 [0.68 0.04 0.24 0.64 0.16 0.91 0.44 0.86 0.5  0.16]]
Interval        Oi              Ei              (Oi - Ei)^2 / Ei
1               9               10              0.1
2               14              10              1.6
3               8               10              0.4
4               10              10              0.0
5               9               10              0.1
6               11              10              0.1
7               13              10              0.9
8               9               10              0.1
9               7               10              0.9
10              10              10              0.0

Hypothesis Test:
Chi-square statistic: 4.2
Critical value (alpha = 0.05): 16.918977604620448
Null hypothesis accepted. The numbers are likely generated from a uniform distribution.
```

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 data.
- No assumptions about the distribution of data are needed.
- Can be used to compare any two distributions, not just normal distributions.
- Sensitive to differences in both location and shape of distributions.

Cons:

- Less powerful than the chi-square test for detecting differences in the tails of distributions.
- More sensitive to differences in the center of distributions than in the tails.
- Requires larger sample sizes for small effect sizes compared to other tests.

Chi-Square Test:

Pros:

- Suitable for categorical data analysis.
- Easy to understand and interpret.
- Provides a single statistic and p-value to summarize the degree of discrepancy between observed and expected frequencies.
- Powerful for detecting differences in the tails of distributions.

Cons:

- Assumes that observed frequencies are independent.
- Assumes that the sample size is large enough for the chi-square distribution to approximate the sampling distribution of the test statistic.
- Less suitable for small sample sizes or when expected frequencies are low.

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

Ans: Minimum Sample Size for Uniformity and Independence Tests:

- The minimum sample size required depends on several factors including the desired level of significance, the effect size, and the specific test being used. However, generally:

- For the Kolmogorov-Smirnov test, there is no strict minimum sample size requirement, but it becomes more reliable with larger sample sizes, especially for detecting small effect sizes.

- For the Chi-Square test, a common guideline is to have expected frequencies greater than 5 in each cell of the contingency table. So, the minimum sample size depends on the number of categories and the expected frequencies.

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

Ans: Importance of Testing Random Number Generators:

Testing random number generators (RNGs) is essential for several reasons:

- **Ensuring randomness:** RNGs are used in various applications such as simulations, cryptography, and statistical sampling. It's crucial to ensure that the numbers generated are statistically indistinguishable from true random numbers.
- **Detecting biases:** RNGs may have biases or patterns that can lead to inaccurate results or vulnerabilities in cryptographic systems.
- **Verifying compliance:** Many industries and standards require the use of validated RNGs, especially in fields such as finance, gambling, and cryptography.
- **Improving reliability:** Testing RNGs helps identify flaws and improve the quality and reliability of random number generation algorithms.
- **Establishing trust:** By testing RNGs rigorously and transparently, users can have confidence in the randomness and integrity of the generated numbers.

4.List out Chi – Square Test Applications?

Ans:

The chi-square test is a statistical method used to determine whether there is a significant association between categorical variables. It has various applications across different fields. Some common applications of the chi-square test include:

1. Goodness-of-fit test: Assessing whether observed frequency data fit a theoretical distribution (e.g., testing whether observed genetic ratios fit expected Mendelian ratios).

2. Independence test: Determining whether there is a significant association between two categorical variables in a contingency table (e.g., testing whether there is a relationship between smoking habits and lung cancer).

3. Homogeneity test: Comparing the distributions of two or more independent populations to determine if they have the same distribution (e.g., comparing the distribution of blood types in different ethnic groups).

4. Test for association in case-control studies: Assessing whether there is an association between an exposure and an outcome in case-control studies (e.g., testing whether there is an association between a certain genetic variant and a particular disease).

5. Test for goodness of fit in regression analysis: Evaluating the goodness of fit of a regression model by comparing observed and expected frequencies of data points across different categories or ranges.

6. Testing for independence in survey analysis: Determining whether there is a significant relationship between responses to different survey questions (e.g., testing whether there is a relationship between gender and political affiliation in a survey).

Outcomes:

CO2 : Generate pseudorandom numbers and perform empirical tests to measure the quality of a

pseudorandom number generator

Conclusion:

Learnt about chi square test and their implementations using Linear Congruential method

References:

Books/ Journals/ Websites:

1. "Linear Congruential Generators" by Joe Bolte, Wolfram Demonstrations Project.
2. Severance, Frank (2001). *System Modeling and Simulation*. John Wiley & Sons, Ltd. p. 86. ISBN 0-471-49694-4.
3. The GNU C library's *rand()* in stdlib.h uses 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 that reproduces the random sequence from this library.
4. "A collection of selected pseudorandom number generators with linear structures, K. Entacher, 1997". Retrieved 16 June 2012.
5. "How Visual Basic Generates Pseudo-Random Numbers for the RND Function". *Microsoft Support*. Microsoft. Retrieved 17 June 2011.
6. In spite of documentation on MSDN, RtlUniform uses LCG, and not Lehmer's algorithm, implementations before Windows Vista are flawed, because the result of multiplication is cut to 32 bits, before modulo is applied
7. GNU Scientific Library: Other random number generators
8. Novice Forth library
9. Matsumoto, Makoto, and Takuji Nishimura (1998) ACM Transactions on Modeling and Computer Simulation
10. S.K. Park and K.W. Miller (1988). "Random Number Generators: Good Ones Are Hard To Find". *Communications of the ACM***31** (10): 1192–1201. doi:10.1145/63039.63042.
11. D. E. 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.
12. P. L'Ecuyer (1999). "Tables of Linear Congruential Generators of Different Sizes and Good Lattice Structure". *Mathematics of Computati on***68** (225): 249–260. doi:10.1090/S0025-5718-99-0099 6-5.
13. Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007), " Section 7.1.1. Some History", *Numerical Recipes: The Art of Scientific Computing* (3rd ed.), New York: Cambridge University Press, ISBN 978-0-521-88068-8
14. Gentle, James E., (2003). *Random Number Generation and Monte Carlo Methods*, 2nd edition, Springer, ISBN 0-387-00178-6.
15. Joan Boyar (1989). "Inferring sequences produced by pseudo-random number generators". *Journal of the ACM***36** (1): 129–141. doi:10.1145/58562.59305. (in this paper, efficient algorithms are given for inferring sequences produced by certain pseudo-random number generators).