PH 415 Simulation Techniques

Assignment-I

Name: Ujjval Charpota

Roll No.: 210121058

Date: 16-08-2024

1. Introduction

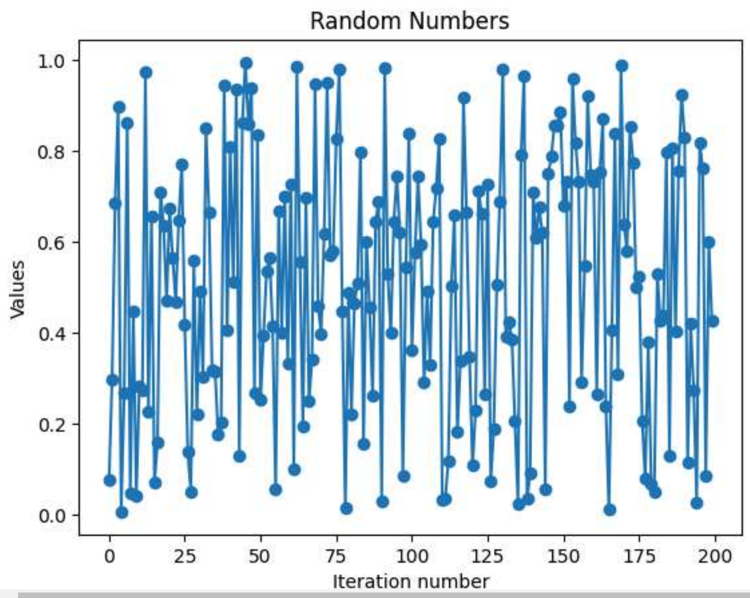
In this assignment, we are required to generate 2000 random numbers using the Linear Congruential Method (LCM). The modulus p is selected based on the last four digits of my Roll Number. The generated sequence will be tested using the Square test, Cube test, and Chi-Square test to evaluate its randomness. The results, along with a discussion, are presented below.

2. Python code

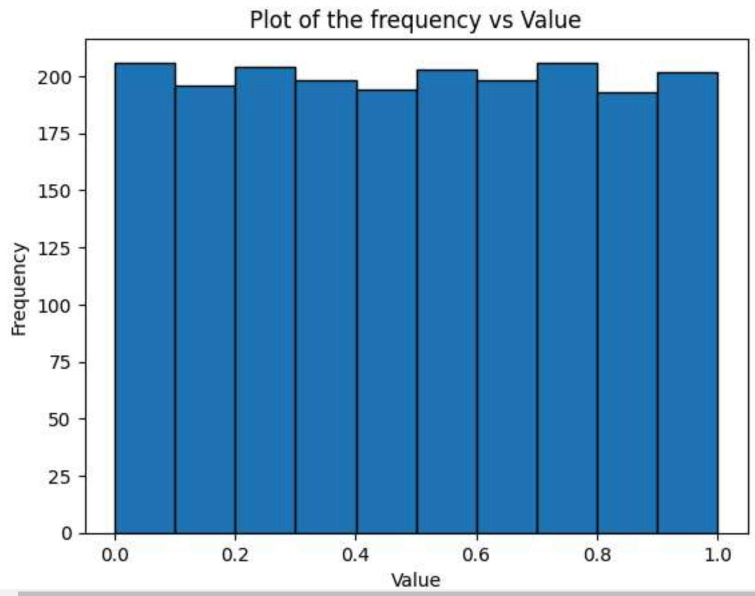
```
import numpy as np
from scipy.stats import chisquare


import matplotlib.pyplot as plt


def random_numbers(seed, a, c, p, count):
    x = seed
    result = np.zeros(count, dtype=np.int32)
    for i in range(count):
        x = (a*x+c)%p
        result[i] = x
    return result


p = 1058 # Last 4 digit of my roll number 210121058
a = 29 # Coprime with p not too close to p
c = 53 # Coprime wiht p and smaller than p
seed = 1 # Arbitrary seed
count = 2000 # count of random numbers


y = random_numbers(seed, a, c, p, count)
y = y/(p-1)
```

Plot of first 200 random numbers generated

```
 def plot(random_numbers):
    plot_range = 200
    plt.plot(y[:200], marker="o")
    plt.title("Random Numbers")
    plt.xlabel("Iteration number")
    plt.ylabel("Values")
    plt.show()

plot(random_numbers)
```

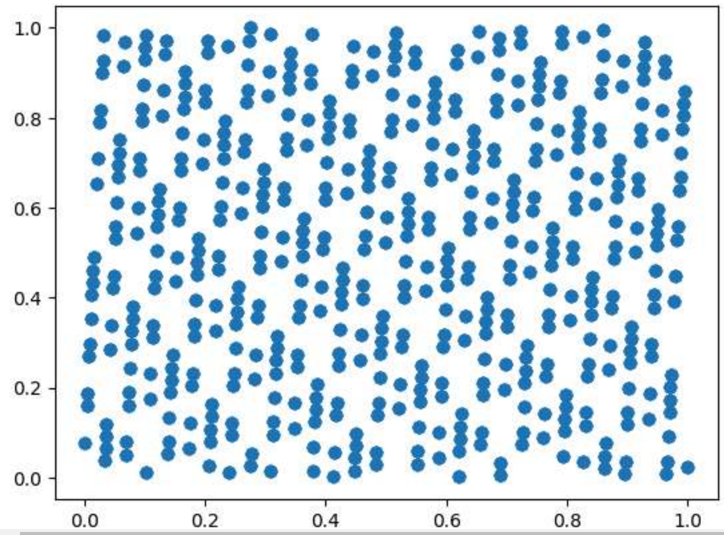


```
num_bins = 10 # Too large then the data will be noisy and too small then data will be oversimplified


plt.hist(y,bins=num_bins ,edgecolor="black")
plt.title("Plot of the frequency vs Value")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```



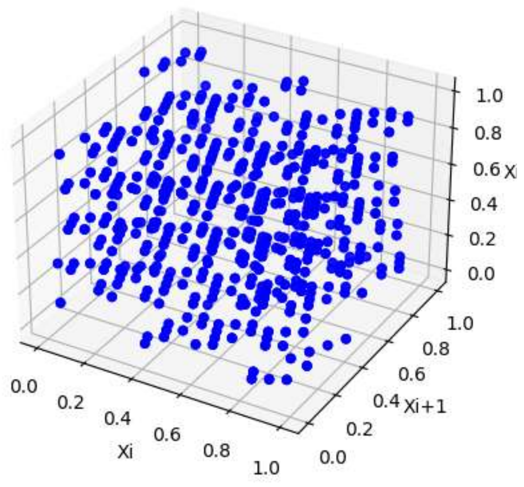
Square test:

```
def cube_test(nums):
    x = nums[:-1]
    y = nums[1:]
    plt.scatter(x,y)
    plt.show()

cube_test(y)
```



Cube test

```
# %matplotlib notebook


def cube_test(nums):
    x = nums[:-2]
    y = nums[1:-1]
    z = nums[2:]
    fig = plt.figure()
    ax = fig.add_subplot(111, projection="3d")
    ax.scatter(x, y, z, color="blue")
    ax.set_xlabel("Xi")
    ax.set_ylabel("Xi+1")
    ax.set_zlabel("Xi+2")
    plt.show()

cube_test(y)
```



Chi Squared test:

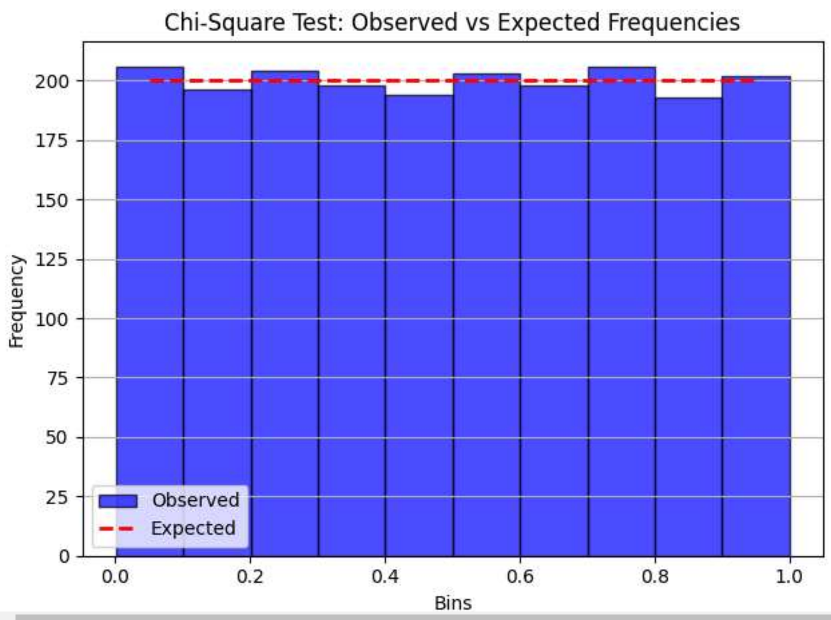
```
num_bins = 10
# Observed counts
Oi, bins = np.histogram(y, bins=num_bins)
# Expected counts
Ei = np.ones(num_bins)*len(y)/num_bins
# Perform chi-squared test
chi2_value, p_value = chisquare(Oi, Ei)
```

```
print(f"chi squared value: {chi2_value:.6f}")
```

```
chi squared value: 1.050000
```

These are relatively high p-value so it doesn't provide any strong evidence for discarding null hypothesis so the distribution of random numbers are quite close to the expected distribution(uniform distribution)

```
# Plotting
def chi_square_test(nums, num_bins):
    # Observed counts
    Oi, bins = np.histogram(y, bins=num_bins)
    # Expected counts
    Ei = np.ones(num_bins)*len(y)/num_bins
    bin_centers = (bins[:-1] + bins[1:]) / 2  # Calculate the centers of the bins for the x-axis

    plt.hist(y, bins=num_bins, alpha=0.7, label='Observed', color='blue', edgecolor="black")
    plt.plot(bin_centers, Ei, 'r--', label='Expected', linewidth=2)

    plt.xlabel('Bins')
    plt.ylabel('Frequency')
    plt.title('Chi-Square Test: Observed vs Expected Frequencies')
    plt.legend()
    plt.grid(axis='y')
    plt.tight_layout()
    plt.show()

chi_square_test(y, num_bins)
```



Observed frequencies of numbers

```
print(Oi)
```

```
[206 196 204 198 194 203 198 206 193 202]
```

Expected frequencies of numbers (uniform distribution)

```
print(Ei)
```

```
[200. 200. 200. 200. 200. 200. 200. 200. 200. 200.]
```

Report

3. Results

Plot of the First 200 Random Numbers: The plot of the first 200 numbers generated by the LCM shows a scattered distribution, which indicates a lack of immediate patterns or correlations in the sequence.

Square Test:

The plot of random number pairs (x0,x1), (x1,x2),...,(xn-1,xn) shows a pattern. This is not a good random generator. This is due to the fact that choosen p (1058) is not a Mersenne Prime number. Even after trying different mulpliers "a" and increament "c" I couldn't get a patternless plot.

Cube Test:

Here we plotted (x0,x1,x2), (x1,x2,x3),....,(xn-2,xn-1,xn). Here we also observed a pattern that numbers were lying on equidistant planes. The reason is the same as mentioned in square test.

Chi-Square Test:

Chi-Squared Value: 1.05 This suggests that the LCM with the chosen parameters generates numbers that pass the Chi-Square test for randomness.

4. Discussion

Parameter Selection:

Here we were restricted to choose p = 1058(in my case) as the roll last 4 digits of my roll number 210121058. This is not a Mersenne Prime number. It is not even a prime number. This affected the quality of random number generators. We choose multiplier "a=29" such that it is coprime of p which is not too close to p and increment "c=53" such that it is coprime of "p". This got as a good p-value in chi square test but pattern was still visible.

Conclusion:

The Linear Congruential Method, with the chosen parameters, appears to generate a sequence that passes standard tests for randomness. The sequence does exhibit some patterns, as evidenced by the Square and Cube tests. The Chi-Square test confirms the uniformity of the distribution, indicating that the generated numbers are suitable for applications requiring pseudo-random sequences. But it doesn't pass square test and cube test so well.

```
Start coding or generate with AI.
```