

MATH 512 - Project 2

**Wasif Ahmed, Haoxiang Deng, Jacob Fein-Ashley,
Kanav Malhotra, Longzhe Yang**

Question 1 (a)

We use the *Kolmogorov-Smirnov* test to test for the uniformity of the random numbers generated by the LCG. The test statistic is given by

$$D_n = \max_{1 \leq i \leq n} \left(\frac{i}{n} - U_{(i)} \right) \vee \max_{1 \leq i \leq n} \left(U_{(i)} - \frac{i-1}{n} \right)$$

where $U_{(i)}$ is the i -th order statistic of the U_i 's.

- H_0 : the random numbers are uniformly distributed.

We find that $D_n = 0.0069$ and a **p-value of 0.708**. This means that we fail to reject H_0 at the 5% significance level and conclude that the random numbers **are uniformly distributed**.

Question 1 (b)

Parameters: $a = 6, m = 11, x_0 = 3, c = 0, n = 10$

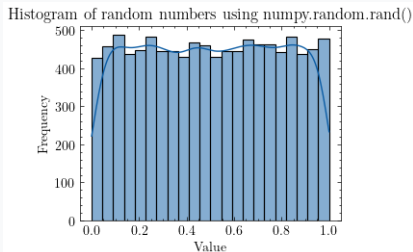
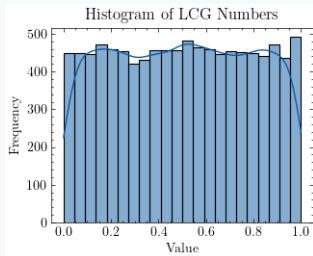
- The sequence is $\{3, 7, 9, 10, 5, 8, 4, 2, 1, 6\}$
- The period is 10

Parameters: $a = 6, m = 10, x_0 = 3, c = 0, n = 10$

- The sequence is $\{3, 8, 8, 8, 8, 8, 8, 8, 8, 8\}$
- The period is 1

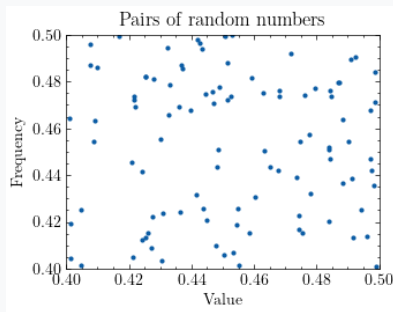
We notice that even a *small* change in the parameters results in a seemingly non-random sample.

Question 1 (d)



- The two histograms look relatively similar, meaning both look relatively uniform.

Question 1 (e)



- The pairs of numbers seem to have a random pattern.

Question 1 (f)

Disadvantages of LCG:

- It can appear random with the right set of parameters, but as we saw, it can get “stuck” in a loop.
- The randomness depends on the choice of parameters.
- Particularly sensitive to parameter choices m and a

Question 2

$$P(X = k) = \begin{cases} 0.3 & \text{for } k = 1 \\ 0.2 & \text{for } k = 2 \\ 0.35 & \text{for } k = 3 \\ 0.15 & \text{for } k = 4 \end{cases}$$

We draw 10000 random Uniform random numbers using the following rule:

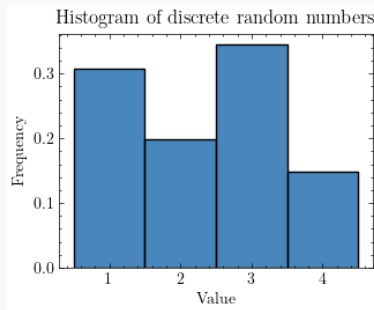
$$U \leq 0.3 \implies X = 1$$

$$0.3 < U \leq 0.5 \implies X = 2$$

$$0.5 < U \leq 0.85 \implies X = 3$$

$$0.85 < U \leq 1 \implies X = 4$$

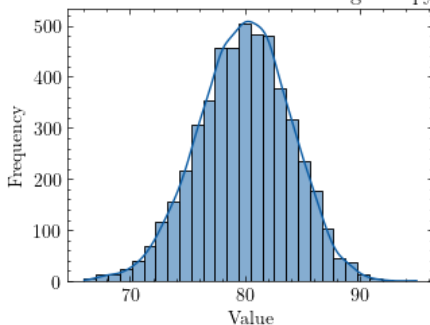
Question 2



Question 3(a) NumPy r.v. Generator

- Observed Probability that $X < 50$: 0.000000000000.

Histogram of binomial random numbers using numpy.random.rand()



- Theoretical Probability:

$$P(X < 50) = \sum_{k=0}^{49} \binom{100}{k} (0.8)^k (0.2)^{100-k} = 2.14 \times 10^{-11}$$

Question 3(b) Inverse Transformation Method

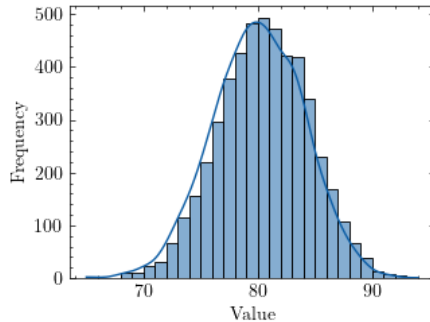
Generating a sum of random Bernoulli random variables is much faster. Nonetheless, we use the following algorithm for selecting a Binomial distribution:

Algorithm Binomial RV Generation

```
1: for  $n = 1, \dots, n_{sim}$  do  
2:   pmf = [ $\binom{n}{k} \cdot p^k \cdot (1-p)^{n-k}$  for  $k \in \{1, \dots, k, k+1\}$ ]  
3:   cdf =  $\sum_{i=1}^{k+1} pmf_i$   
4:   Draw  $U \sim Uniform[0, 1]$   
5:    $x[i] = \arg \max(cdf > U)$   
6: end for
```

Question 3(b) Inverse Transformation Method

Histogram of binomial random numbers using inverse transformation



Question 3(c)

- Time for NumPy r.v. Generator: 0.0726s
- Time for inverse transformation method: 0.8775s

Question 4

Formula:

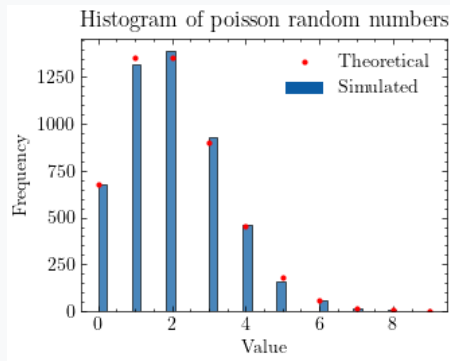
$$P(X = k) = \frac{e^{-\lambda} \lambda^k}{k!}$$

Poisson r.v. with $\lambda = 2$. We use the following algorithm:

Algorithm Poisson RV Generation

```
1: for  $n = 1, \dots, n_{sim}$  do  
2:    $x[i] = 0$   
3:    $j = 0$   
4:   while  $x[i] < \lambda$  do  
5:      $x[i]- = \log(U)$   
6:      $j+ = 1$   
7:   end while  
8:    $x[i] = j - 1$   
9: end for
```

Question 4



Question 5

Exponential r.v. with $\lambda = 5$.

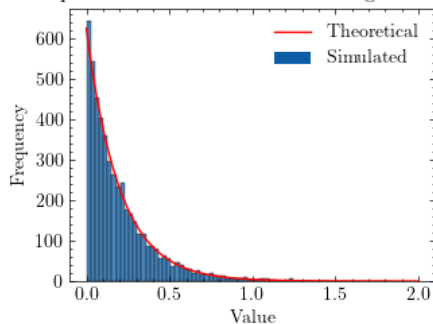
Inv. Transformation Method:

$$F(x) = 1 - e^{-\lambda x}$$

$$F^{-1}(u) = -\frac{\log(1-u)}{\lambda}$$

Question 5

Histogram of exponential random numbers using inverse transformation



Question 6

- The Cauchy r.v. has the following pdf:

$$f(x) = \frac{1}{\pi(1+x^2)}$$

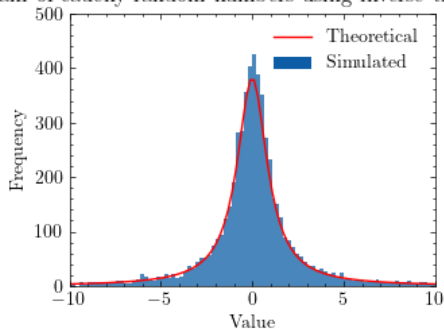
- We use the inverse transformation method to generate the Cauchy r.v.

$$F(x) = \frac{1}{\pi} \arctan(x) + \frac{1}{2}$$

$$F^{-1}(u) = \tan\left(\pi\left(u - \frac{1}{2}\right)\right)$$

Question 6

Histogram of cauchy random numbers using inverse transformation



Question 7

Simulation:

- For $x=1,2,\dots,100$, shuffle the position of $x[i]$
- After shuffling, if $x[i]=i$, count 1, o.w. 0.
- Sum up all the counting.

With a repetition of 10000 simulations, we find that the expectation is 1.00 and the variance is 1.01.

We can calculate $\mathbb{E}[X]$ and $\mathbb{V}[X]$ exactly based on Poisson Distribution and compare them with our estimates.

$$\mathbb{E}[X] = \sum_{i=1}^{100} i \cdot \frac{1^i \cdot e^{-1}}{i!} = 1$$

$$\mathbb{V}[X] = \sum_{i=1}^{100} i^2 \cdot \frac{1^i \cdot e^{-1}}{i!} - (\mathbb{E}[X])^2 = 1$$

The estimates are very close to the exact values. This is expected because the number of simulations is large.

Question 8: Rolling Two Die

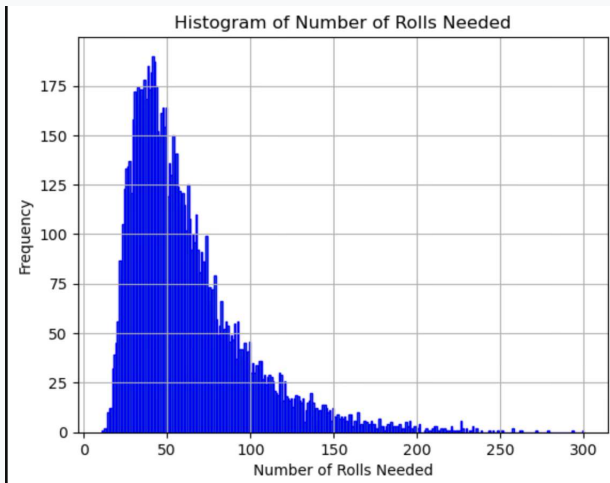
Algorithm:

- Generate two random integers in between $[1, 6]$ and take the sum.
- Keep on generating the sum until all the integers in between $[2, 12]$ are generated.
- Count the number of iterations it requires.
- Repeat the same process 10000 times.

Result: The expected number of rolls required is **60.8352**.

Note: Calculating expected values by hand would require some Markov Chain (simulation is easier!)

Question 8: Rolling Two Die



Question 9: Random Selection of 10 Balls

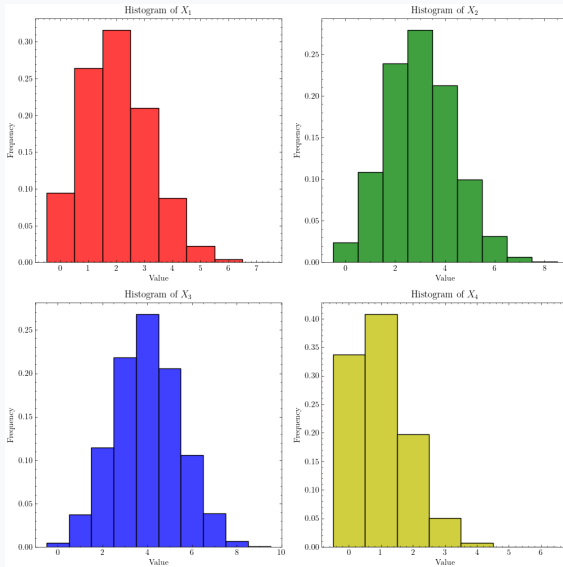
Algorithm

- Make an array of the following form:

$$urn = \{0, 0, \dots 20 \text{ times}; 1, 1, \dots 30 \text{ times}; \\ 2, 2, \dots 40 \text{ times}; 3, 3, \dots 10 \text{ times}\}$$

- Reshuffle urn .
- Pick the first 10 numbers and count the number of times we have $\{X_1, X_2, X_3, X_4\} = \{0, 1, 2, 3\}$.
- Repeat.

Question 9: Random Selection of 10 Balls



Question 10(a): Box-Muller Method

Algorithm:

- Generate two independent $U[0, 1]$ set of random numbers: u_1 and u_2 .
- Define:

$$R = \sqrt{-2 \log(u_1)}$$

$$x = \cos(2\pi u_2)$$

$$y = \sin(2\pi u_2)$$

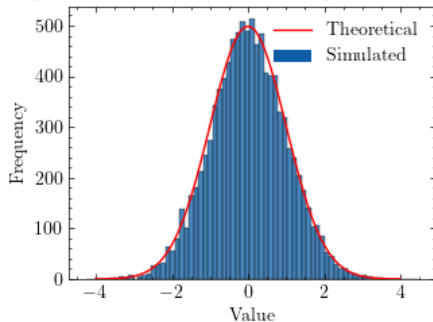
$$z_1 = r * x$$

$$z_2 = r * y$$

- Return z_1 and z_2 .

Question 10 (a): Box-Muller Method

Histogram of random numbers using Box-Muller method



Question 10(b): Marsaglia-Bray Method

Algorithm:

- Generate two independent $U[0, 1]$ set of random numbers: u_1 and u_2 .
- Define:

$$w_1 = (2u_1) - 1$$

$$w_2 = (2u_2) - 1$$

$$s = w_1^2 + w_2^2$$

- if $s < 1$,

$$t = \sqrt{-2 \frac{\log(s)}{s}}$$

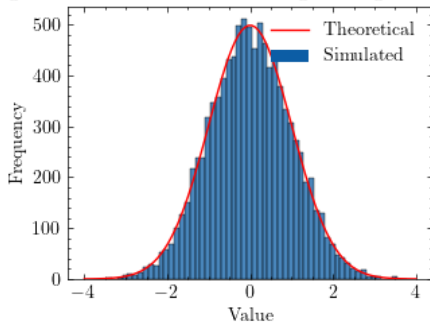
$$z_1 = w_1 * t$$

$$z_2 = w_2 * t$$

- Return z_1 and z_2 .

Question 10(b): Marsaglia-Bray Method

Histogram of random numbers using Marsaglia-Bray method



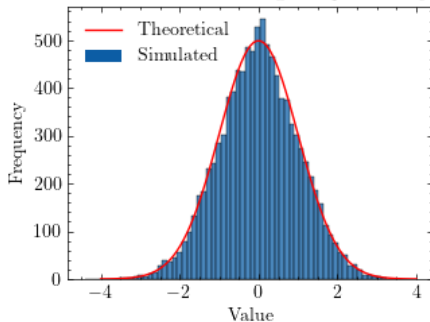
Question 10(c): Acceptance-Rejection Method

Algorithm:

- Define:
Normal Density Function (NDF): $\exp(-\frac{x^2}{2})/2\pi$
Proposal Density Function (PDF): $\exp(-x^2/2)$
- Generate $u \sim U[0, 1]$ and $x \sim \text{PDF}$.
- $\sim 1.32 = \sqrt{2e/\pi}$
- if $u < \frac{\text{NDF}[x]}{1.32 * \text{PDF}[x]}$, then accept x .

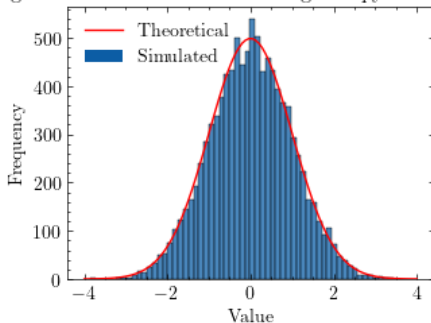
Question 10(c): Acceptance-Rejection Method

Histogram of random numbers using acceptance-rejection method



Question 10(d): `random.randn()`

Histogram of random numbers using `numpy.random.randn()`



All of the methods are very close to the theoretical normal distribution.

Question 10: Timing

Table: Execution Times of Various Random Number Generation Methods

Method	Time (ms)	Comments
Box-Muller Method	1.2269	Fast execution
Marsaglia-Bray Method	16.712	Moderate execution
Acceptance-Rejection Method	75.590	Slow execution
<i>random.randn()</i>	0.1469	Very fast execution

Questions?