Assignment - 2

Farhan Alam 12340740

January 23, 2025

Question 1

Imagine you are studying two distinct processes that generate random numbers between 0 and 1, modeled as continuous random variables X with different distributions:

- The first process generates numbers following an **exponential distribution** with $\lambda = 2$.
- The second process generates numbers following a **uniform distribution** between 0 and 1.

You collect n random numbers from each process and define a new random variable:

$$Y = F_X(x)$$

where $F_X(x)$ represents the **Cumulative Distribution Function (CDF)** of X. The goal is to derive the Probability Density Function (PDF) of Y for both processes and analyze their histograms for different values of n.

Data

Generated during the program by using np.random.uniform(0,1) and then stored in n size arrays after transforming it thorugh cdf of expoentitial distribution with lambda =2 and using uniform distribution between (0,1)

Methodology

The cdf of exponential distribution is with lambda =2 is given by :

$$F_X(x) = \begin{cases} 1 - e^{-2x} & x \geq 0, \\ 0 otherwise \end{cases}$$

The cdf of uniform distribution between (0,1) is given by:
F(x)=x 0¡=x¡=1
0 otherwise
Now the uniform random variable is being transformed through these two functions so to find the distribution lets rather solve a general case by considering some one to one monotonic disribution:
Let $X \sim U(0,1)$ be a uniform random variable with CDF $F_X(x)$:

$$F_X(x) = \begin{cases} 0, & x < 0 \\ x, & 0 \leq x \leq 1 \\ 1, & x > 1 \end{cases}$$

We apply a one-to-one monotonic transformation $Y = g(X)$, and aim to find the CDF and PDF of Y.

1 CDF of Y

The CDF of Y is:

$$F_Y(y) = P(Y \leq y) = P(g(X) \leq y) = P(X \leq g^{-1}(y))$$

Using the CDF of X, we get:

$$F_Y(y) = F_X(g^{-1}(y)) = g^{-1}(y), \quad g(0) \leq y \leq g(1)$$

Thus, the CDF of Y is:

$$F_Y(y) = \begin{cases} 0, & y < g(0) \\ g^{-1}(y), & g(0) \leq y \leq g(1) \\ 1, & y > g(1) \end{cases}$$

2 PDF of Y by Differentiation

To find the PDF of Y, we differentiate the CDF $F_Y(y)$ with respect to y:

$$f_Y(y) = \frac{d}{dy} F_Y(y) = \frac{d}{dy} \left(g^{-1}(y) \right)$$

Using the chain rule, we get:

$$f_Y(y) = \frac{1}{g'(g^{-1}(y))}$$

Thus, the PDF of Y is:

$$f_Y(y) = \frac{1}{g'(g^{-1}(y))}, \quad g(0) \leq y \leq g(1)$$

3 Conclusion

By differentiating the CDF of Y, we obtain the PDF of Y as:

$$f_Y(y) = \frac{1}{g'(g^{-1}(y))}$$

This result is general for any monotonic transformation $Y = g(X)$ of a uniform random variable X with length (1).
So we can directly say that both of these cdfs would give us a uniform distribution between 0 and 1
Now we are going to proove it using a python cod. . The code is as follows :-

```
import numpy as np
import matplotlib.pyplot as plt
p = [10, 100, 1000, 10000]

for i in p:
    k = np.random.exponential(scale=1/2, size=i)
    l = (1 - np.exp(-2*k))
    m = np.random.uniform(0, 1, size=i)

    plt.figure(figsize=(12, 6))

    plt.subplot(1, 2, 1)
    plt.hist(l, bins=20, density=True, color='orange')
    plt.title(f"Exponential Process Histogram (n={i})")
    plt.xlabel("Y")
    plt.ylabel("Distribution")

    #plt.figure()
    plt.subplot(1, 2, 2)
    plt.hist(m, bins=20, density=True, color='red')
    plt.title(f"Uniform Process Histogram (n={i})")
    plt.xlabel("Y")
    plt.ylabel("Distribution")

    plt.show()
```

The code technically does the same as the questions with use of x for uniform number generation l for converting to y using exponetial and m ofr converting to y using uniform asks us to then plots the histogram for the observation .

Result

For the output the code gives two histograms(exponential,uniform) for various n between (10,10000) .The result was as follows://

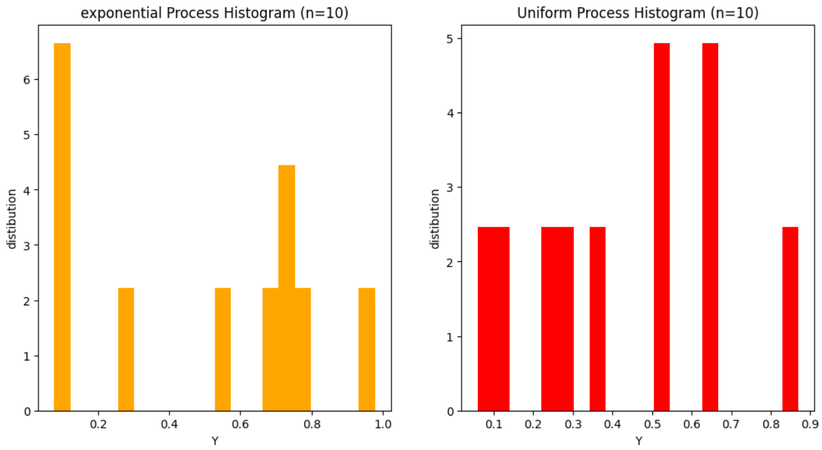

Figure 1:

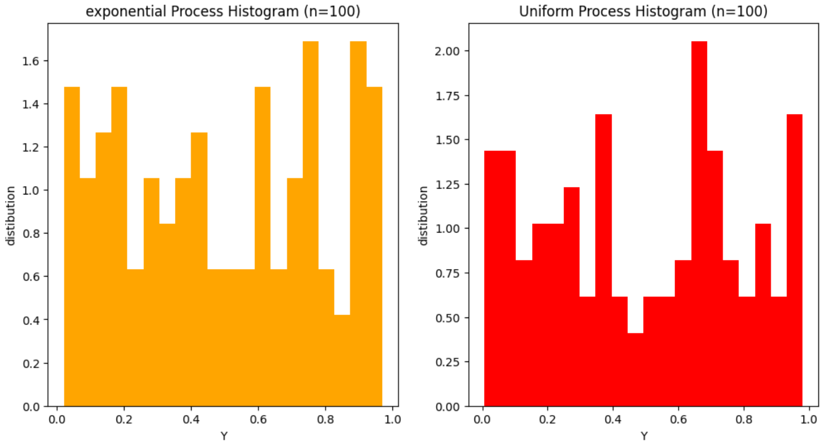

Figure 2:

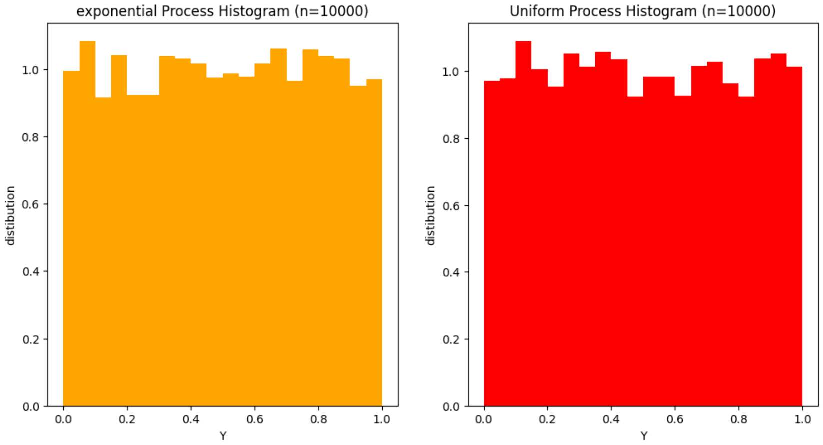

Figure 3:

As we can see, as n increase the ditribution starts tending to uniform distribution of (0,1) henceforth we can say that the statement we porrved above hold true

Conclusion

The conclusion is that if we transform a uniform distribution between 0 and 1 using any cdf we are then bound to get a unifrom distribution between (0,1) whih we have provved mathematically as well as using a python program.

Question 2

Imagine you are an archivist analyzing a dusty old text file from a forgotten library. Your task is to uncover the hidden patterns in the text by doing the following:

1. **Count the Words:** Create a histogram of how often each word appears.
2. **Focus on the Key Players:** Identify the top 30 most frequent words from your list.
3. **CDF Transformation:** Once you have this list of important words, calculate the Cumulative Distribution Function (CDF) for their frequencies. Use this CDF as a transformation function to remap the word frequencies.

What new insights can you uncover about the text after applying this transformation?

Data

`file_text.txt`

Methodology

We first open the file and then extract words from that then using counter function get a dictionary with each unique word and its count. Then we convert it into a dataframe and sort it according to the frequency of each word and the display the barplot. then we caculate the probablity of thse 30 words by considering only these 30 words and their occurences as the universal space after whihwe caculate the cdf and plot it using bar graph. The code used is as follows :

```python
import re
from collections import Counter
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd


with open('text_file.txt','r') as file:
    text = file.read()
    words = re.findall(r'\b\w+\b', text.lower())
    counts = Counter(words)


l = pd.DataFrame(counts.items(), columns=['words', 'frequency'])
l = l.sort_values('frequency', ascending=False)


f = l.head(30)


plt.figure(figsize=(30, 12))
plt.subplot(1, 2, 1)
plt.bar(f['words'], f['frequency'], color='blue')
plt.title("Original␣Word␣freq")
plt.xticks(rotation=90)
plt.xlabel("Words")
plt.ylabel("Frequency")


plt.subplot(1, 2, 2)
k = 0
q = []
for i in f['frequency']:
    k = k + i
    q.append(k)
for i in q:
    i = i / k
```

```

plt.bar(f['words'], q, color='blue')
plt.title("CDF-Transformed␣Word␣freq")
plt.xlabel("Words")
plt.ylabel("Transformed␣Frequency␣(CDF)")

plt.show()
```

The code follows these steps:

1. It reads the contents of `text_file.txt` and processes it to extract words.

2. Using the `Counter` class from the `collections` module, it counts the frequency of each word.

3. The frequencies of the top 30 words are plotted on a bar chart (Figure 1).

4. It then calculates the Cumulative Distribution Function (CDF) for the top 30 words by summing the frequencies progressively.

5. A second bar chart (Figure 2) displays the CDF-transformed word frequencies.

The two subplots help visualize the original word frequencies and how the CDF transformation normalizes the frequencies across the top words.

Result

The output we received was as follows:

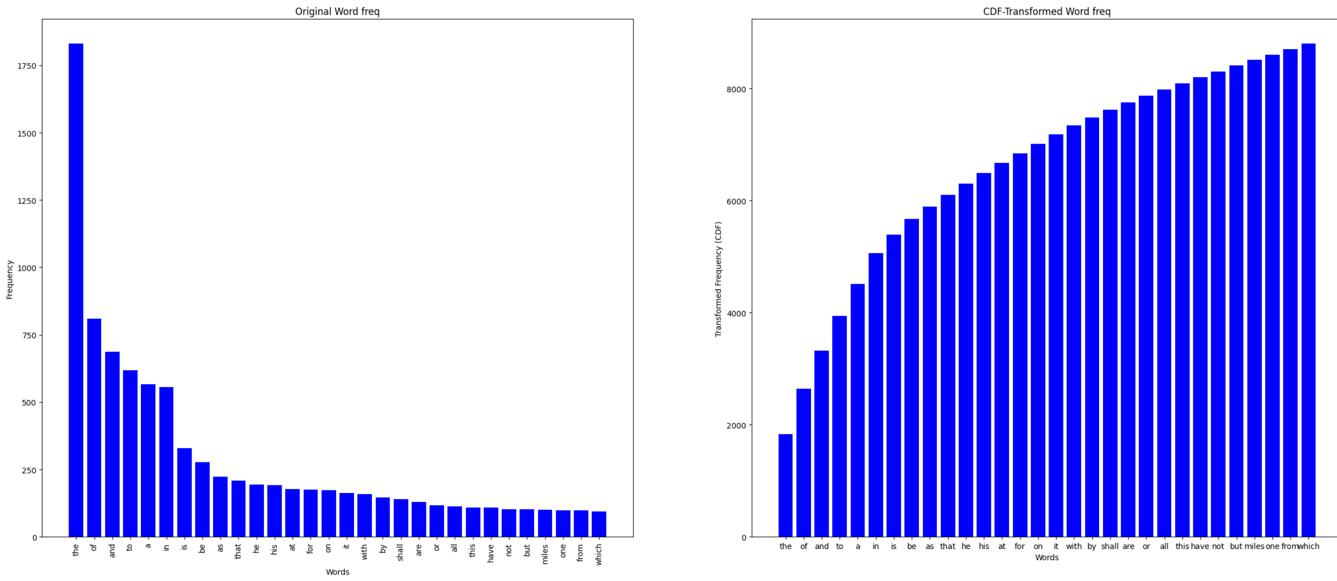

Figure 4:

After applying the Cumulative Distribution Function (CDF) transformation to the word frequencies, the remapped values display a more balanced distribution. Words that initially had high frequencies are now adjusted based on the cumulative distribution, resulting in a smoother and more uniform representation of the word frequency structure.

The CDF plot effectively illustrates how the cumulative frequency increases as we progress through the list of most frequent words. This visualization highlights how a small set of high-frequency words dominate the majority of the text, while the remaining words contribute to the frequency incrementally.

Conclusion

The CDF transformation smooths out extreme word frequency biases. In the original histogram, a few high-frequency words dominate the text, skewing the analysis. By normalizing the distribution with the CDF, we can reveal less obvious patterns in the text.

Question no.3

Imagine you are working with a random number U, which is drawn from a uniform distribution between 0 and 1. You have a tool that allows you to transform this number by using the inverse CDF of a distribution X. By applying this transformation, you create a new number Y.

Now, your task is to determine what kind of random variable Y becomes after the transformation. Consider the following two cases:

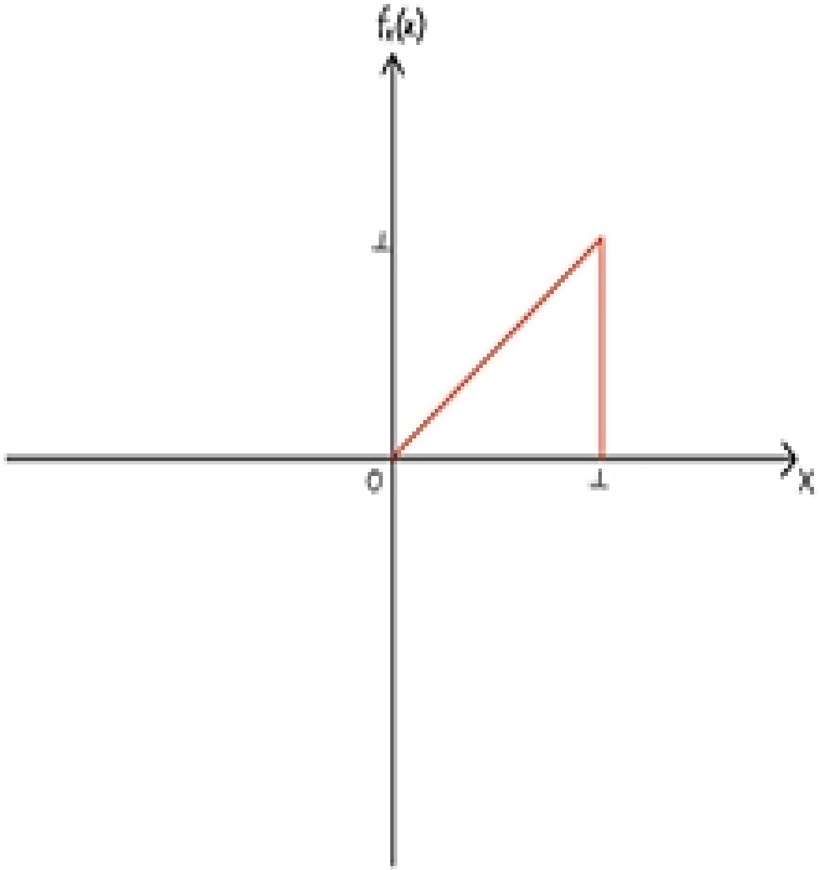

Figure 5:

Data

Produced in the same manner as done in Question no.1

Methodology

For this as done in question no.1 first we are producing random uniform no. between 0 abd 1 and then tranforming through inverse cdf of expoentital distribution i.e -log(1-x)/lambda and y=x dirstiburiotn that is sqrt(2x). In this case we need to find diribution of the ew vairable formed for that we can easily find

$$P(Y \leq F^{-1}(y))$$

i.e F(y) it self as here we are already taking the inverse which then again becomes The original cdf itself and as we have uniform function whose cdf is x itself so we get F(y) again hence the distribution will be the istribution used to ranfomr the uniform distribution. This can be proove by code as follows:

```python
import numpy as np
import matplotlib.pyplot as plt

n = 100000
l_exp = 1

# Generate uniform random variables
U = np.random.uniform(0, 1, n)

# Exponential distribution transformation
Y = -np.log(1 - U) / l_exp

# Triangular distribution transformation
X = np.sqrt(2 * U)

# Plot Exponential Distribution
plt.subplot(1, 2, 1)
plt.hist(Y, bins=30, density=True, color=’skyblue’,
   edgecolor=’black’)
plt.title("Exponential␣Distribution␣log(1-x)")
plt.xlabel("Y")
plt.ylabel("Density")

# Plot Triangular Distribution
plt.subplot(1, 2, 2)
plt.hist(X, bins=30, density=True, color='orange',
    edgecolor='black')
plt.title("Triangular Distribution sqrt(x)")
plt.xlabel("Y")
plt.ylabel("Density")

# Adjust layout and show the plot
plt.tight_layout()
plt.show()
```

Result

The output was as follows:

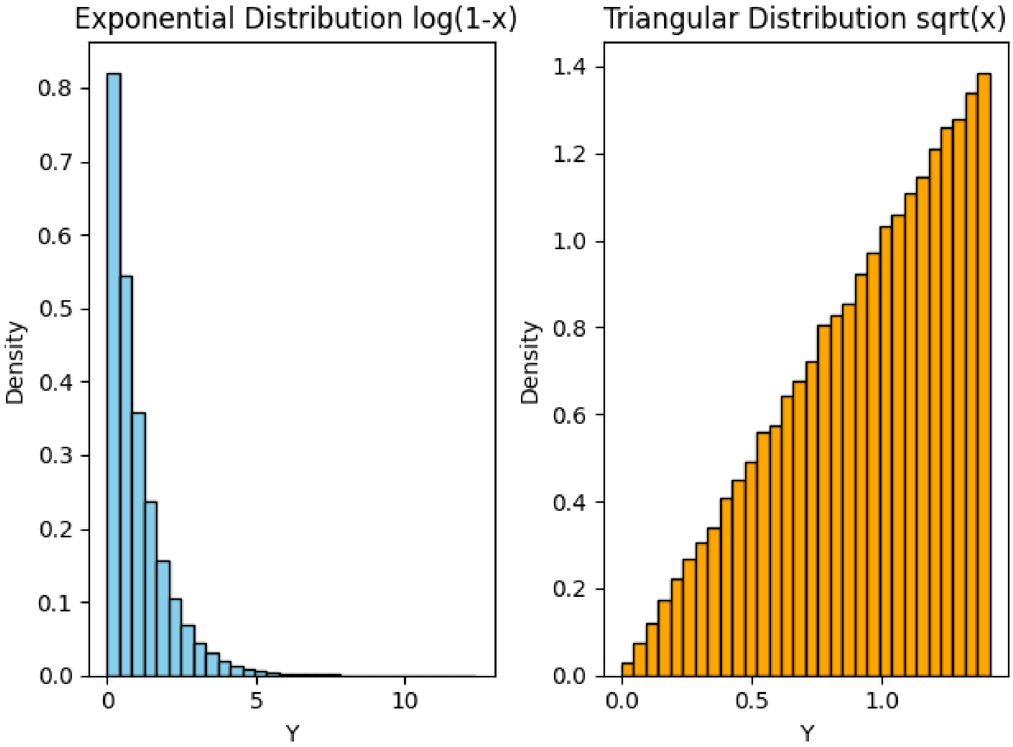

Figure 6:

As we can see they resemble the disrtibution given used to tranform hecne we can say that if we use inverse tranform of a cdf to change a uniform distribution then we we get distribution of the function which was used to transform it.

Conclusion

If a unifrom function is tranformed through innverse cdf of a function then the resultant function will the function whose cdfs inverse was taken