Random numbers generation

Vasilis Tsioulos*

*Project 2: Computational mathematics II 2025

ABSTRACT

This study investigates the random number generation from the inverse function method. The first step was to explain the inverse function method to generate randomly distributed numbers with a desired distribution (pdf) starting from a uniform one. . The next step was to compare my explanationwith Deepseek's answer. Then the method to generate exponentially distributed random numbers with $\lambda = 2$, and compare it to the numpy library **random.exponential()**, and use a Q-Q plot to check if my numbers are exponentially distributed. Then the Q-Q plot is explained.

1 Introduction

1.1 Inverse function method

From the probability theory we know that any random variable can be written as a function of a uniform random variable **U** on the (0,1) ($U \sim U(0,1)$). We need the reverse cumulative distribution function (CDF) of the desired random variable.

If X is a random variable with the CDF $F(x) = P(X \leq x)$. The inverse can be defined as:

$$F^{-1}(y) = inf\{x : F(x) \geq y\} \tag{1}$$

2 Deepseek's explanation

Inverse Function Method for Random Number Generation

The **inverse function method** (also called the **inverse transform method**) is a technique for generating random numbers that follow a desired probability distribution function (PDF) by transforming uniformly distributed random numbers. Here's how it works:

Key Idea:

If X is a random variable with a cumulative distribution function (CDF) $F_X(x)$, then $F_X(X)$ follows a uniform distribution on $[0,1]$. Conversely, if U is uniformly distributed on $[0,1]$, then $F_X^{-1}(U)$ has the same distribution as X.

Steps to Generate Random Numbers with Desired PDF:

1. **Find the CDF $F(x)$ of the desired distribution:**

$$F(x) = \int_{-\infty}^{x} f(t)\,dt$$

   where $f(x)$ is the given PDF.

2. **Compute the inverse CDF $F^{-1}(u)$:** Solve $F(x) = u$ for x to get $x = F^{-1}(u)$.

3. **Generate uniform random numbers U in $[0,1]$:** Use a standard uniform random number generator (e.g., `rand()` in many programming languages).

4. **Apply the inverse CDF to U:** The transformed random variable $X = F^{-1}(U)$ will follow the desired distribution.

Example: Exponential Distribution

Suppose we want to generate numbers from an exponential distribution with PDF:

$$f(x) = \lambda e^{-\lambda x}, \quad x \geq 0$$

1. **Compute the CDF:**

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

2. **Find the inverse CDF:**

$$u = 1 - e^{-\lambda x} \implies x = -\frac{1}{\lambda}\ln(1-u)$$

3. **Generate $U \sim$ Uniform$(0,1)$ and apply $F^{-1}(U)$:**

$$X = -\frac{1}{\lambda}\ln(1-U)$$

   (Since $1-U$ is also uniform, we can simplify to $X = -\frac{1}{\lambda}\ln(U)$.)

Why It Works:

- The CDF $F(x)$ maps values of X to the interval $[0,1]$.
- The inverse $F^{-1}(u)$ maps back from $[0,1]$ to the domain of X, preserving the probability structure.

Limitations:

- Requires an analytically computable (and invertible) CDF.
- May be inefficient for some distributions (e.g., Gaussian, where other methods like the Box-Muller transform are preferred).

Final Formula:

$$X = F^{-1}(U), \quad \text{where} \quad U \sim \text{Uniform}(0,1)$$

This method is widely used for distributions like exponential, Cauchy, and Pareto, where the inverse CDF is easily obtainable. For more complex distributions, alternative methods (e.g., rejection sampling) may be needed.

3 Coding the method

3.1 Generating exponential random numbers

The exponential distribution is a probability distribution commonly used to model the time until the next event. A particularly useful distribution when studying situations where events occur randomly and independently at a constant average rate over time. The characteristic of the exponential distribution is its parameterization (scale and size). For the custom exponential distribution, all the code was just a few lines:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

lambda_ = 2
sample_size = 100000
u = np.random.uniform(0,1,sample_size)
x = np.log(1-u)/(-lambda_)
```

Listing 1. Custom exponential number generator

3.2 Numpy's exponential distribution

The Numpy's exponential distribution is defined by a single parameter which is called scale and determines the average rate of events. The scale controls the shape of distribution and must be greater than zero. The exact equation of the numpy's scale parameter is:

$$scale = 1/\lambda \tag{2}$$

where $\lambda = 2$ in our case. The function **np.random.exponential()** returns random samples from the standard exponential distribution, where the scale parameter is one.

The code to replicate the generation of random numbers drawn from exponential distribution is:

```
x_numpy = np.random.exponential(scale=1/lambda_,size=sample_size)
```

Listing 2. Numpy's exponential distribution sampling

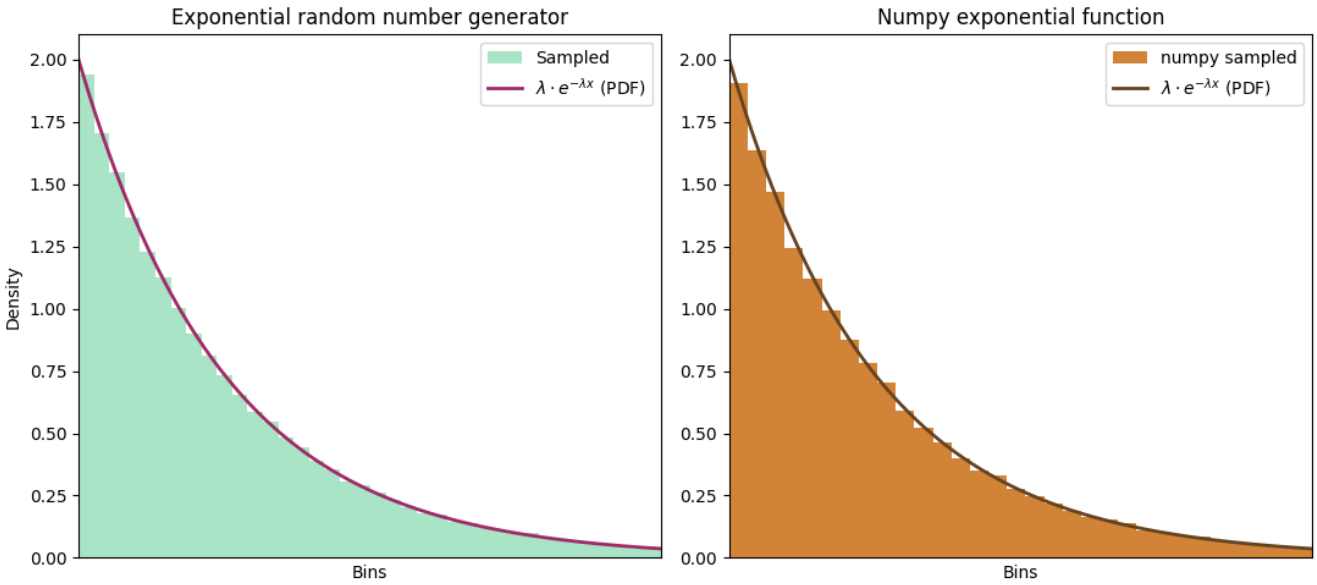

Figure 1. Comparison of two methods of exponential distribution sampling

The results of both methods were plotted in two subplots and compared to each other.

As one can see, the results of the two plots are almost identical, but the next step is to use the Q-Q plot to see better the differences.

4 Q-Q plot

The Q-Q plot, also known as quantile-quantile plot, is a graphical tool to help us discover if the set of data comes from some theoretical distribution such as normal or exponential. In our case we can use it to check our assumption that our residuals from the statistical analysis are exponentially distributed (visually). The way the Q-Q plot works is that it scatters two datasets on two sets of quantiles against one another. If both quantiles come from the same distribution, they will form a straight line. Another example that we may see the same philosophy in plots is the actual data and the predicted data that are scattered on a plot, in order to see if we performed a good prediction of some machine learning model. The following Q-Q plots are created based on the samples created by the exponential functions implemented.

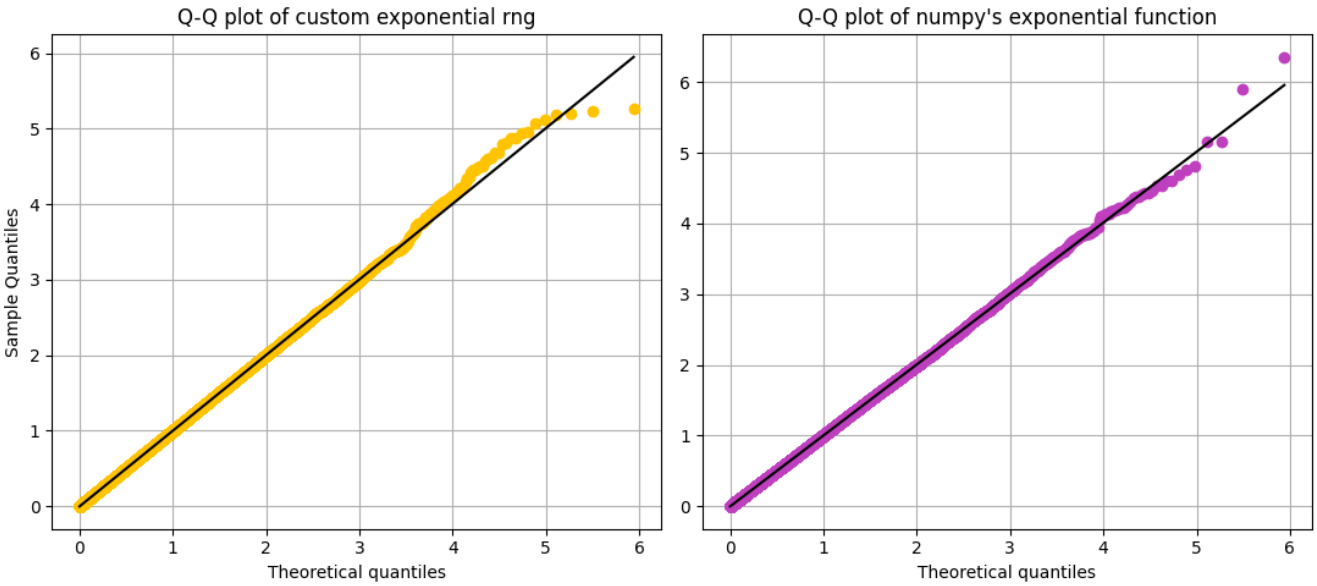

Figure 2. Q-Q plots created by the exponential functions samples.

The results show that the quantiles bisect the center line so they follow the exponential distribution.