# ENGINEERING TRIPOS PART II A

EIETL MODULE EXPERIMENT 3F3

## RANDOM VARIABLES and RANDOM NUMBER GENERATION
**Short Report Template**

**Name: Olly Parker**

**College: Emmanuel**

**Lab Group Number: Student 7, 1st Nov 2024**

**This is a template suitable for the short report write-up. Simply edit the Latex or Word document to include your calculations/ results/ code.**

## 1. Uniform and normal random variables.

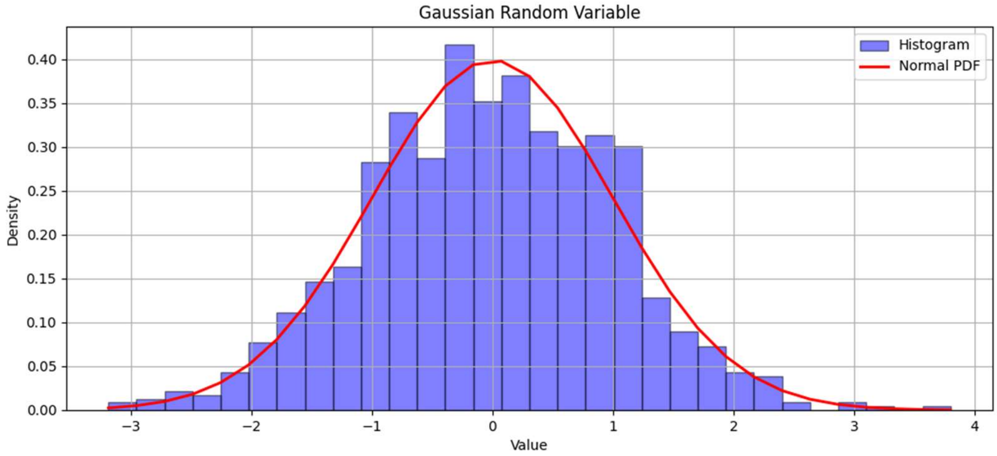


*Figure 1 Histogram of Gaussian random numbers overlaid on exact Gaussian curve (scaled)*

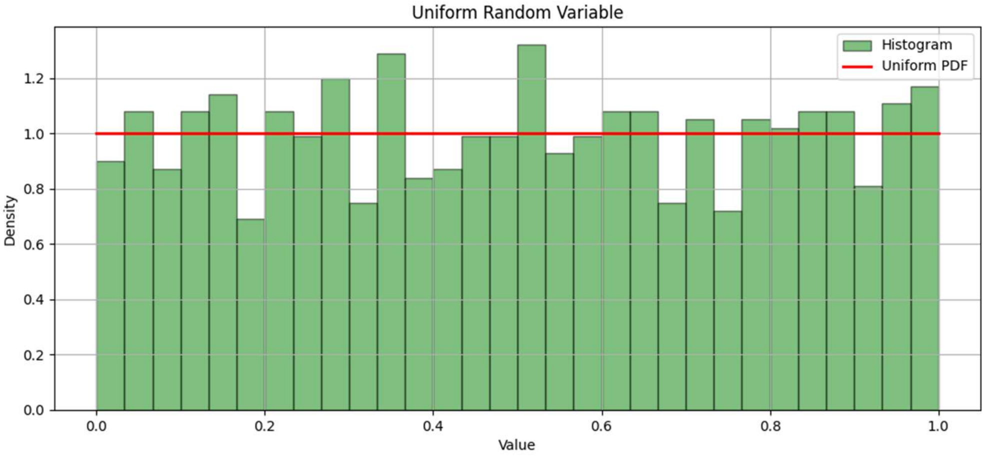


*Figure 2 Histogram of Uniform random numbers overlaid on exact Uniform curve (scaled):*

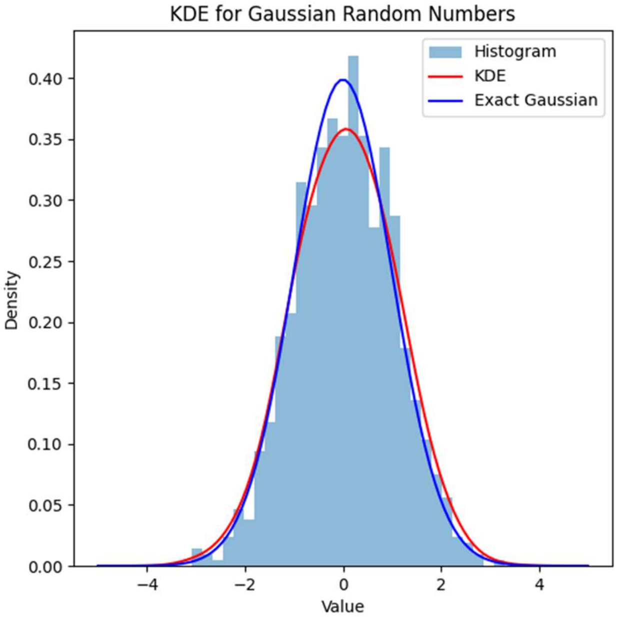


*Figure 3 Kernel density estimate for Gaussian random numbers overlaid on exact Gaussian curve*

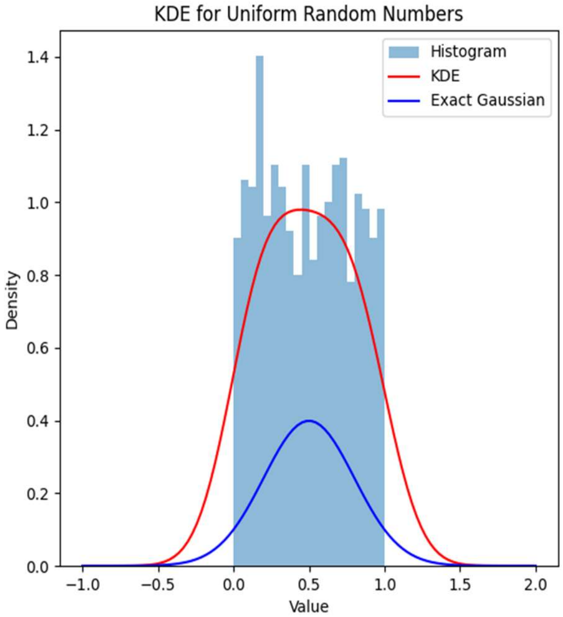


*Figure 4 Kernel density estimate for Uniform random numbers*

Comment on the advantages and disadvantages of the kernel density method compared with the histogram method for estimation of a probability density from random samples:

The kernel density method does provide a smooth curve, which is characteristic of a probability density function. However, in cases such as the uniform distribution where there is a large step, the size of the convolution is too broad and there is an erroneous sloped edge to the kernel density estimate. In this case the histogram provides a far better representation of the density function. Despite this, in the gaussian case which does not characteristically have steps in probability density, the kernel density estimate does provide a reliable estimate of the true density.

Theoretical mean and standard deviation calculation for uniform density as a function of *N*:

The probability $p_j$ of a randomly generated number being in a bin *j* (one bin of a total number of bins *M*) when generated with a uniform distribution is simply:

$$p_j = \frac{1}{M} = \frac{1}{\sum_j j}$$

Thus, when generating *N* random data points, the expected vale of the number of data points in each bin $x_j$ will be:

$$\mathbb{E}[x_j] = Np_j = \frac{N}{M}$$

The variance of the number of values in each bin $Var(x_j)$ can be calculated using the variance of the binomial distribution:

$$X_j \sim B(N, p_j)$$

$$Var(x_j) = N \cdot p_j \cdot (1 - p_j)$$

$$\therefore \ \sigma = \sqrt{Np_j(1 - p_j)}$$

For an arbitrary number of bins *M*, the variance will be:

$$\sigma^2 = N\frac{1}{M}\left(1 - \frac{1}{M}\right)$$

$$\sigma^2 = N\frac{(M-1)}{M^2}$$

$$\sigma = \sqrt{N\frac{(M-1)}{M^2}}$$

Explain behaviour as *N* becomes large:

We see that as *N* progresses from $10^2$ to $10^4$ that the *sample mean*, the mean number of elements in each bin, converges to the *true mean*, calculated above.

While the value of the standard deviation can be seen to increase, we still observe that the data seems to more closely represent the true uniform distribution. This is because while the standard deviation is increasing, the ratio $\frac{\sigma}{N}$ is decreasing with *N*.

$$\frac{\sigma}{N} = \frac{\sqrt{N\frac{M-1}{M^2}}}{N}$$

$$\frac{\sigma}{N} = \frac{A}{\sqrt{N}}$$

(where A is a constant that describes the number of bins used)

This parameter is indicative of the shape of the sampled distribution and demonstrates the behaviour predicted by the Weak Law of Large Numbers.

Plot of histograms for $N$ = 100, $N$ = 1000 and $N$ = 10000 with theoretical mean and ±3 standard deviation lines:

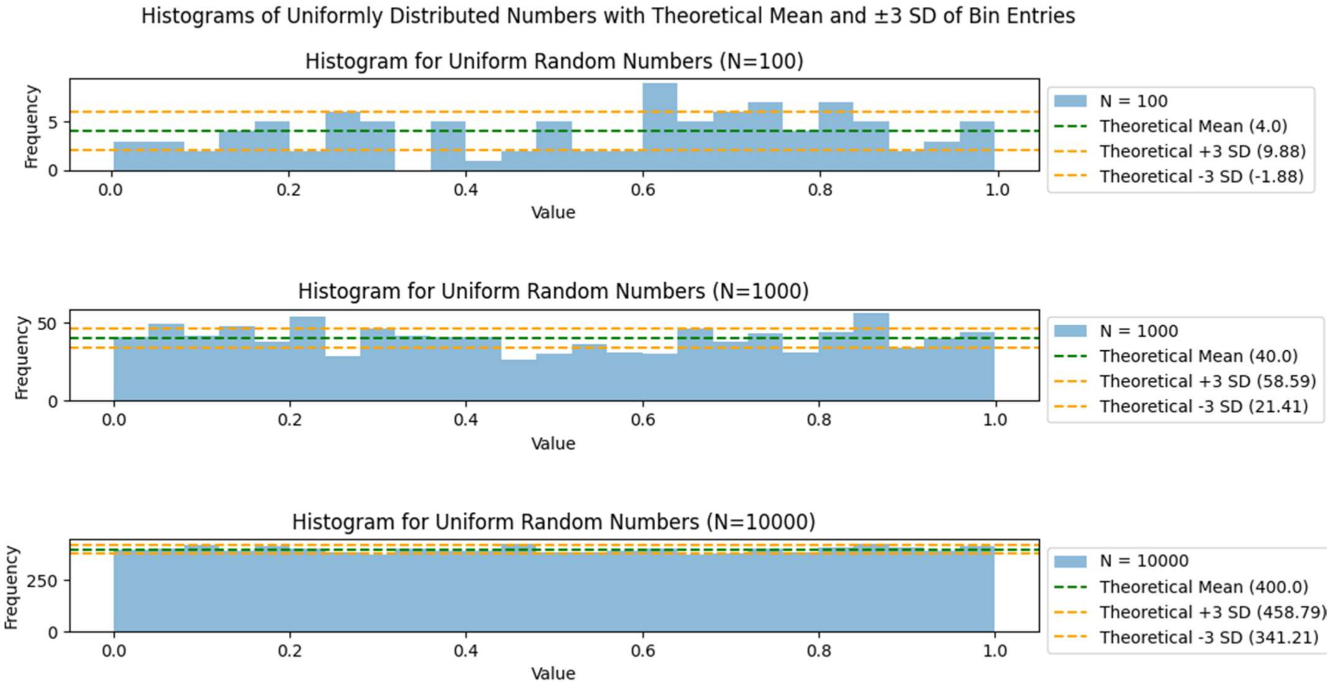


*Figure 5 Histograms of varying sample size generations from a uniform random variable, sorted into 25 bins across the range 0 to 1.*

Are your histogram results consistent with the multinomial distribution theory?

The results shown in figure 5 are consistent with the multinomial distribution, which predicts that the mean of the count data in bin $j$ is $Np_j$ , i.e. it is independent of the bin – meaning that each bin should have the same mean. The histograms can be seen to demonstrate each bin converging to having the same mean as the sample size $N$ increases.

2. **Functions of random variables**

For normally distributed $N(x|0,1)$ random variables, take $y = f(x) = ax + b$.
Calculate $p(y)$ using the Jacobian formula:

$$X \sim N(0,1) \qquad y = f(x) = ax + b$$

We can use the formula:

$$p(y) = \frac{p(x)}{\left|\frac{dy}{dx}\right|}\Bigg|_{x = f^{-1}(y)}$$

With:

$$f^{-1}(y) = \frac{y-b}{a}$$

To find:

$$p(y) = \frac{p\left(\frac{y-b}{a}\right)}{a}$$

$$p(y) = \frac{1}{a\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y-b}{a}\right)^2}$$

It can be seen from the transformed pdf for y that y is a Gaussian distribution defined by:

$$Y \sim N(b, a^2)$$

Explain how this is linked to the general normal density with non-zero mean and non-unity variance:

The general normal density is defined by *X~N(0, 1)*, i.e. zero mean and unity variance. In the case above, for the transformed pdf, y, the mean of the normal density is b and its variance is $a^2$.

Verify this formula by transforming a large collection of random samples $x^{(i)}$ to give $y^{(i)} = \boldsymbol{f}(x^{(i)})$, histogramming the resulting *y* samples, and overlaying a plot of your formula calculated using the Jacobian:

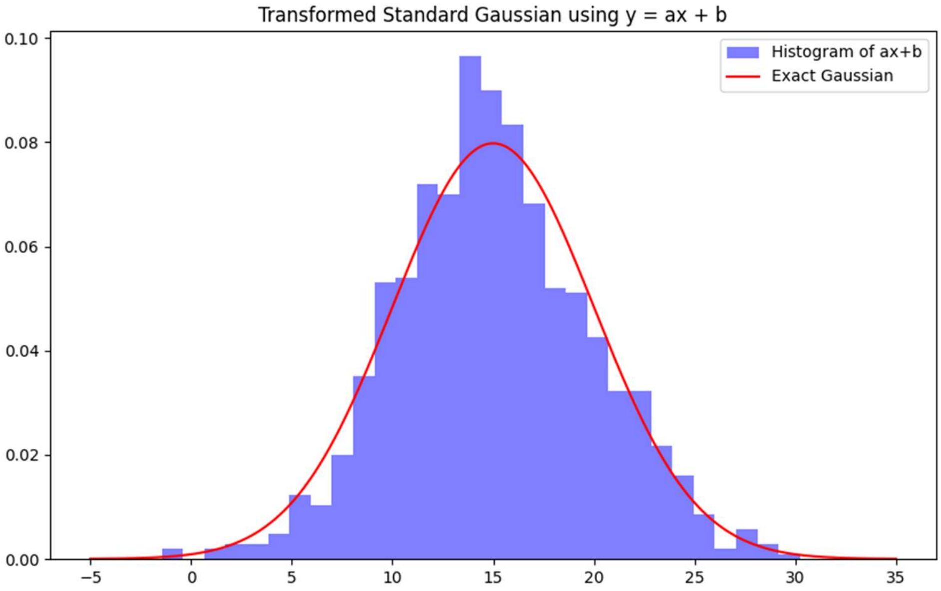


*Figure 6 Transformed Gaussian distribution under the function f(X) = 15 + 5X.*

Now take $p(x) = N(x|0, 1)$ and $\boldsymbol{f}(x) = x^2$. Calculate $p(y)$ using the Jacobian formula:

Again, we use the formula:

$$p(y) = \sum \frac{p(x)}{\left|\frac{dy}{dx}\right|}\Bigg|_{x=f^{-1}(y)}$$

Where we sum the values generated by the inverse.

The desired inverse and derivate in this case are defined by:

$$f^{-1}(y) = \pm\sqrt{y} \qquad \text{and} \qquad \frac{dy}{dx} = 2x$$

Simply using the formula, with the pdf p(x) defined by X~N(0,1), the following result can be found:

$$p(y) = \sum_{x=\pm\sqrt{y}} \frac{p(x)}{|2x|}$$

$$\therefore p(y) = \frac{1}{2\sqrt{y}}\left(\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}(\sqrt{y})^2} + \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}(-\sqrt{y})^2}\right)$$

$$p(y) = \frac{1}{\sqrt{2\pi y}}e^{-\frac{y}{2}}$$

Verify your result by histogramming of transformed random samples:

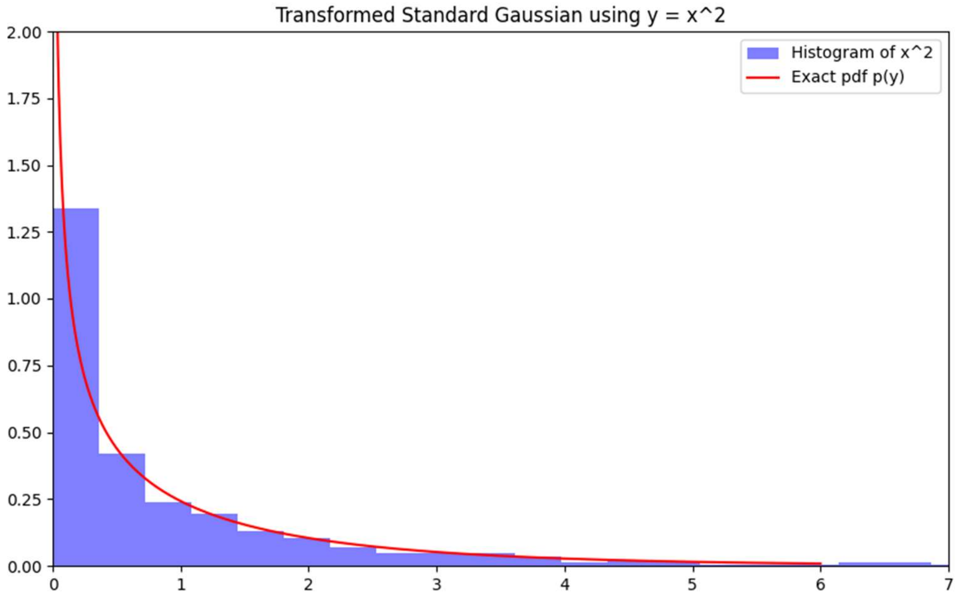


*Figure 7 Transformed Gaussian random variable under the transformation f(X) = X²*

## 3. **Inverse CDF method**

Calculate the CDF and the inverse CDF for the exponential distribution:

The pdf $p(y)$ is defined by:

$$p_Y(y) = e^{-y}$$

The cdf $F(y)$ can simply be found by the following integral:

$$F_Y(y) = \int_{-\infty}^{y} e^{-y} dy = 1 - e^{-y}$$

The inverse cdf $F^{-1}(x)$ can be found by setting x = F(y):

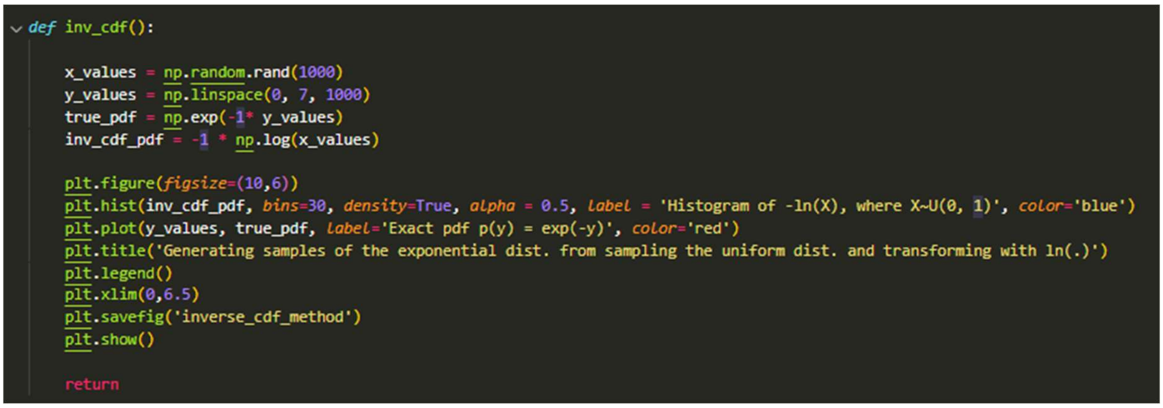
$$y = F^{-1}(x)$$
$$1 - e^{-y} = x$$
$$e^{-y} = 1 - x$$
$$-y = \ln(1 - x)$$
$$y = -\ln(1 - x)$$

However, as our distribution is uniform between 0 and 1, this is equivalent to:

$$y = -\ln(x)$$

$$\therefore F^{-1}(x) = -\ln(x)$$

Python code for inverse CDF method for generating samples from the exponential distribution:



```python
def inv_cdf():

    x_values = np.random.rand(1000)
    y_values = np.linspace(0, 7, 1000)
    true_pdf = np.exp(-1* y_values)
    inv_cdf_pdf = -1 * np.log(x_values)

    plt.figure(figsize=(10,6))
    plt.hist(inv_cdf_pdf, bins=30, density=True, alpha = 0.5, label = 'Histogram of -ln(X), where X~U(0, 1)', color='blue')
    plt.plot(y_values, true_pdf, label='Exact pdf p(y) = exp(-y)', color='red')
    plt.title('Generating samples of the exponential dist. from sampling the uniform dist. and transforming with ln(.)')
    plt.legend()
    plt.xlim(0,6.5)
    plt.savefig('inverse_cdf_method')
    plt.show()

    return
```

Plot histograms/ kernel density estimates and overlay them on the desired exponential density:

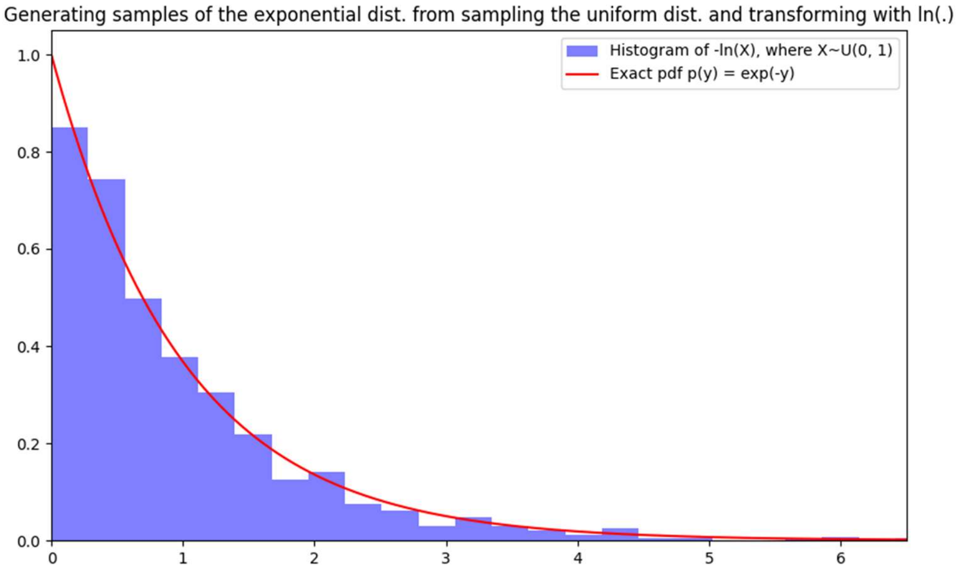
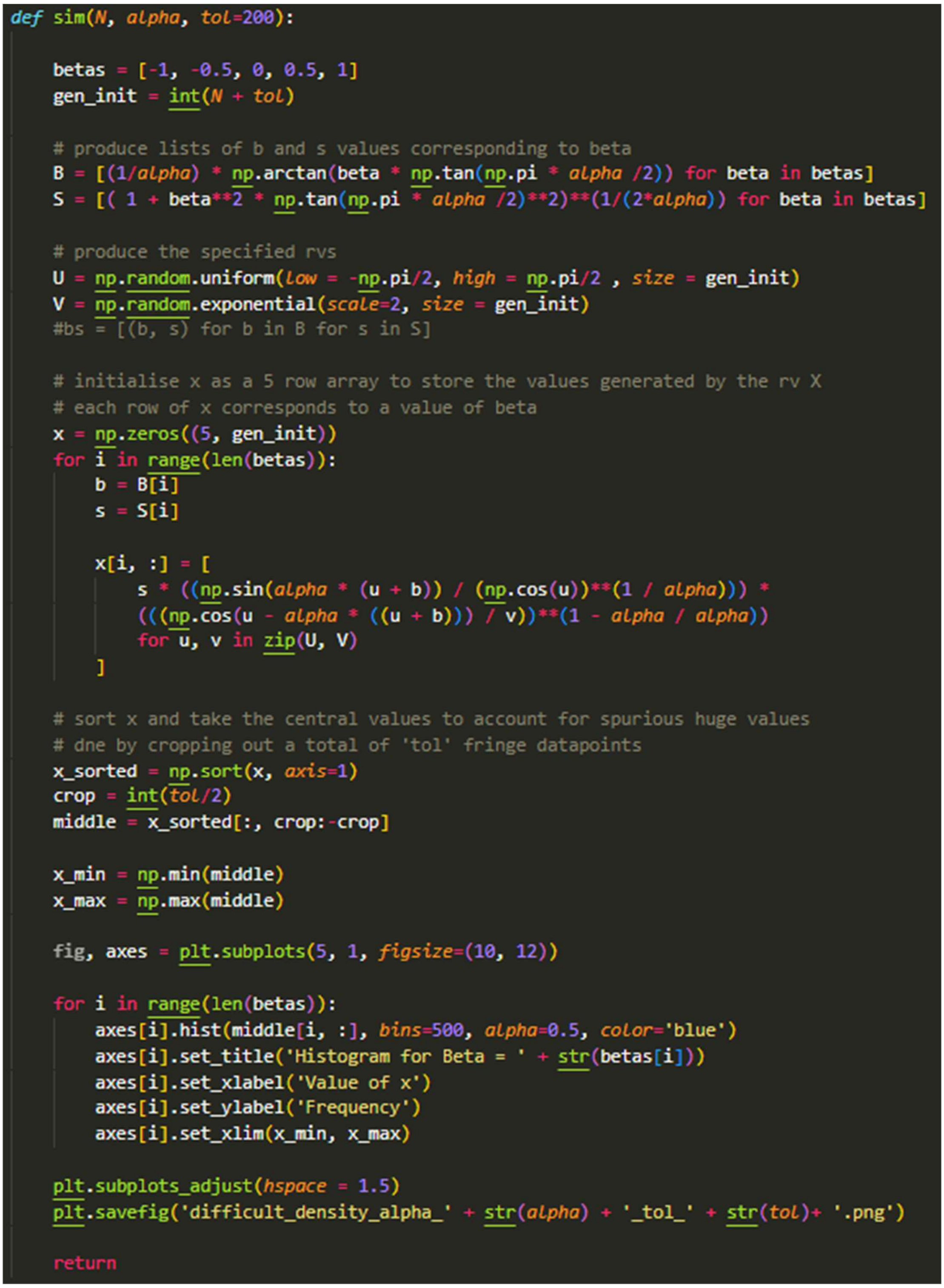


*Figure 8 Samples of the exponential distribution generated using the inverse cdf method.*

4. **Simulation from a `difficult' density.**

Python code to generate N random numbers drawn from the distribution of X:



```python
def sim(N, alpha, tol=200):

    betas = [-1, -0.5, 0, 0.5, 1]
    gen_init = int(N + tol)

    # produce lists of b and s values corresponding to beta
    B = [(1/alpha) * np.arctan(beta * np.tan(np.pi * alpha /2)) for beta in betas]
    S = [( 1 + beta**2 * np.tan(np.pi * alpha /2)**2)**(1/(2*alpha)) for beta in betas]

    # produce the specified rvs
    U = np.random.uniform(low = -np.pi/2, high = np.pi/2 , size = gen_init)
    V = np.random.exponential(scale=2, size = gen_init)
    #bs = [(b, s) for b in B for s in S]

    # initialise x as a 5 row array to store the values generated by the rv X
    # each row of x corresponds to a value of beta
    x = np.zeros((5, gen_init))
    for i in range(len(betas)):
        b = B[i]
        s = S[i]

        x[i, :] = [
            s * ((np.sin(alpha * (u + b)) / (np.cos(u))**(1 / alpha))) *
            (((np.cos(u - alpha * ((u + b))) / v))**(1 - alpha / alpha))
            for u, v in zip(U, V)
        ]

    # sort x and take the central values to account for spurious huge values
    # dne by cropping out a total of 'tol' fringe datapoints
    x_sorted = np.sort(x, axis=1)
    crop = int(tol/2)
    middle = x_sorted[:, crop:-crop]

    x_min = np.min(middle)
    x_max = np.max(middle)

    fig, axes = plt.subplots(5, 1, figsize=(10, 12))

    for i in range(len(betas)):
        axes[i].hist(middle[i, :], bins=500, alpha=0.5, color='blue')
        axes[i].set_title('Histogram for Beta = ' + str(betas[i]))
        axes[i].set_xlabel('Value of x')
        axes[i].set_ylabel('Frequency')
        axes[i].set_xlim(x_min, x_max)

    plt.subplots_adjust(hspace = 1.5)
    plt.savefig('difficult_density_alpha_' + str(alpha) + '_tol_' + str(tol)+ '.png')

    return
```

Plot some histogram density estimates with alpha= 0.5, 1.5 and several values of beta:

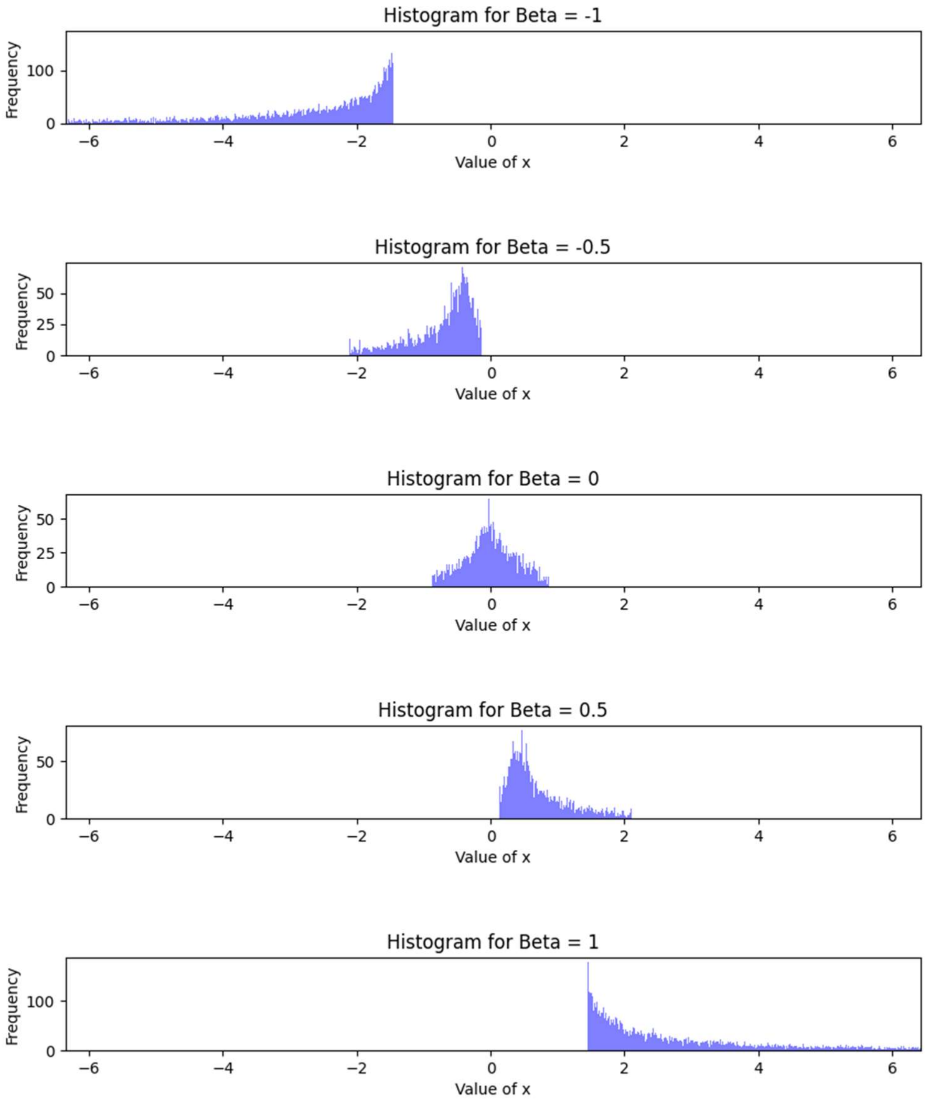


*Figure 9 Histograms of random numbers generated from X. In this case alpha=0.5, the plots above show the central 1,000 datapoints when generated 10,000 datapoints were generated for each value of beta.*

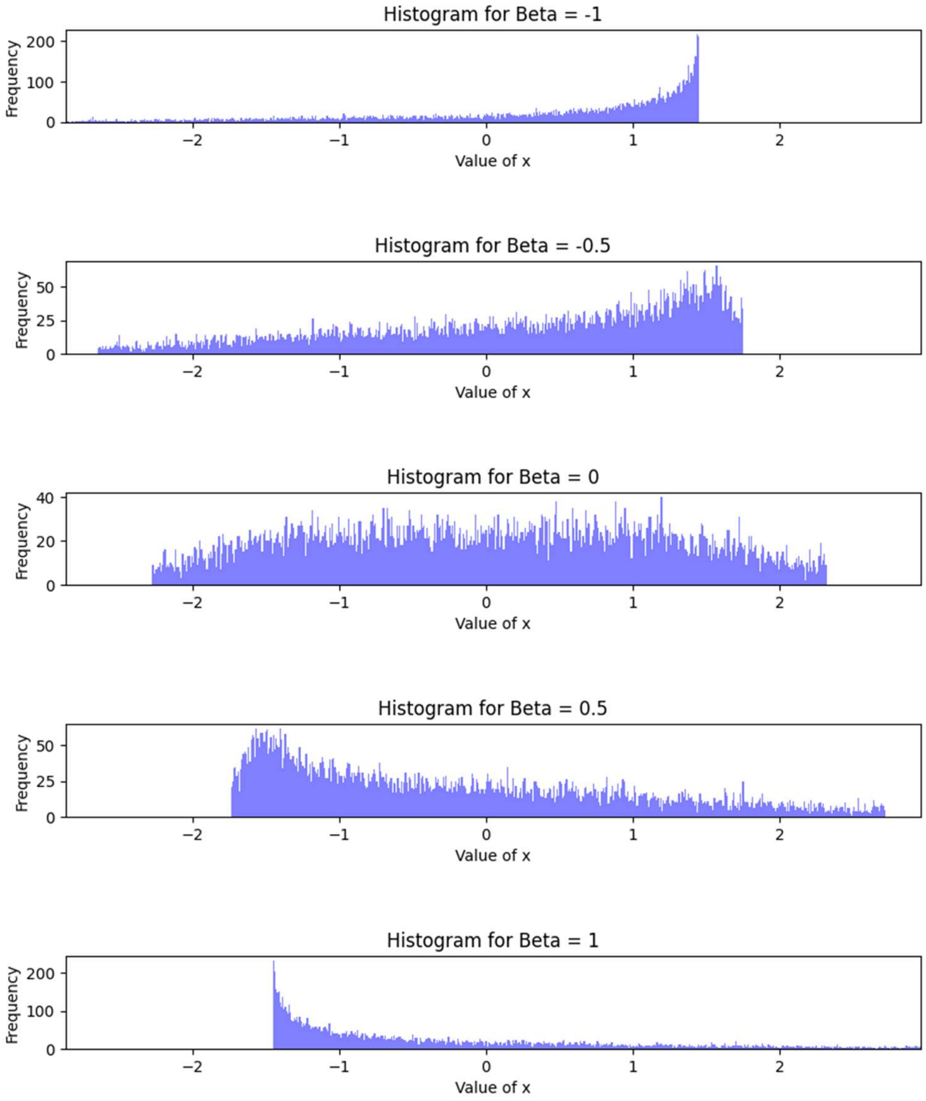


*Figure 10 Histograms of random numbers generated from X. In this case alpha=1.5, the plots above show the central 8,000 datapoints when generated 10,000 datapoints were generated for each value of beta.*

Hence comment on the interpretation of the parameters alpha and beta:

The x-shift caused by the changing value of beta suggests that beta influences the mean of the overall distribution, as well as its symmetry. A beta value of zero gives a perfectly symmetric distribution centered on zero – regardless of alpha. Whereas other beta values induce ‘tailedness’, skewing the distribution depending on the value of beta.

Alpha can be seen to impact the spread of the distribution, in figure 9 (alpha = 0.5) only the central 10% of datapoints are shown, in figure 10 (alpha = 1.5) the central 80% of datapoints are shown. Despite this, the shown histograms in the figures have a similar range, indicating that the *smaller* alpha value results in a far *greater* spread of datapoints. Additionally, while alpha does not affect the general shape of the ‘tailedness’ (aside from the stretch factor caused by the spread) it does induce an x-shift for non-zero values of beta.