ENGINEERING TRIPOS PART II A

EIETL MODULE EXPERIMENT 3F3 RANDOM VARIABLES and RANDOM NUMBER GENERATION

Mark 8.5/10. Markers Comments: Overall, this is a strong report. Please note:
It would be beneficial to try various values of 'width' for the ksdensity method and comment on the trade-off between using a smaller width and the smoothness of the attained approximation.
For y=ax+b; the scalar in the resultant distribution should be 1/|a| in lieu of 1/a, as per the Jacobian Formula.
For y=sin(x), you missed a few details out of the Jacobian calculations, e.g. the boundaries, -1=<y<=1, should be clearly stated in the final expression for p_{Y}(y). Additionally, in the latter, it should be a function of y NOT x (I suspect this is a typo).
Alpha and beta dictate the spread and skewness of the distribution respectively. The treatment of the question of analysing the tail behaviour is incomplete and an expression for gamma can be notably improved.

Contents

1 Introduction

The study of random variables and random number generation is an increasingly useful tool for engineers due to the large number of physical processes that can be modelled in this way. For example, random processes are integral to communications engineering where noise is not-negligible, quantum processes and even mechanical processes such as disturbances in the wind for when planes and wind turbines are being designed.

Computers are discrete and quantized systems and hence, cannot ever generate truly random numbers. Therefore, engineers have developed a good alternative in pseudo-random number generation. It is these pseudo-random numbers and their transformations that will be discussed in this report. The estimations of a number of distributions from the generation of pseudo-random numbers is compared to the theoretical distributions and it is found that these generation methods are a very reliable alternative to theoretical calculations.

In the last section a case in which this estimation and generation method is more useful than the theoretical distributions is discussed.

2 Results and Discussion

2.1 Uniform and Gaussian Random Variables

There are three standard distributions that will be used in this report: the uniform, the Gaussian and the exponential distributions. Their definitions are given below.

$$\mathcal{U}(x|min, max) = \begin{cases} \frac{1}{(max - min)}, & \min < x < \max \\ 0, & otherwise \end{cases}$$

(Equ. 1)

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)}$$

(Equ. 2)

$$\mathcal{E}(x|\mu) = \frac{1}{\mu} e^{-\frac{x}{\mu}}$$

(Equ. 3)

2.1.1 The Kernel Density Method Compared to the Histogram Method for p.d.f. Estimation

In this lab, there were two methods of pdf (probability density function) estimation: Gaussian kernel density smoothing, and histogram estimation.

Histogram Estimation:

This method takes N randomly generated numbers, according to some distribution (Gaussian or Uniform in this lab), and groups them into bins, e.g. numbers between 0.4 and 0.5 might be grouped into a bin together. The bin sizes are given by the number of random numbers that fall within each bin limit. In order to make a valid probability density function, these bin sizes are normalized to give a total area of 1.

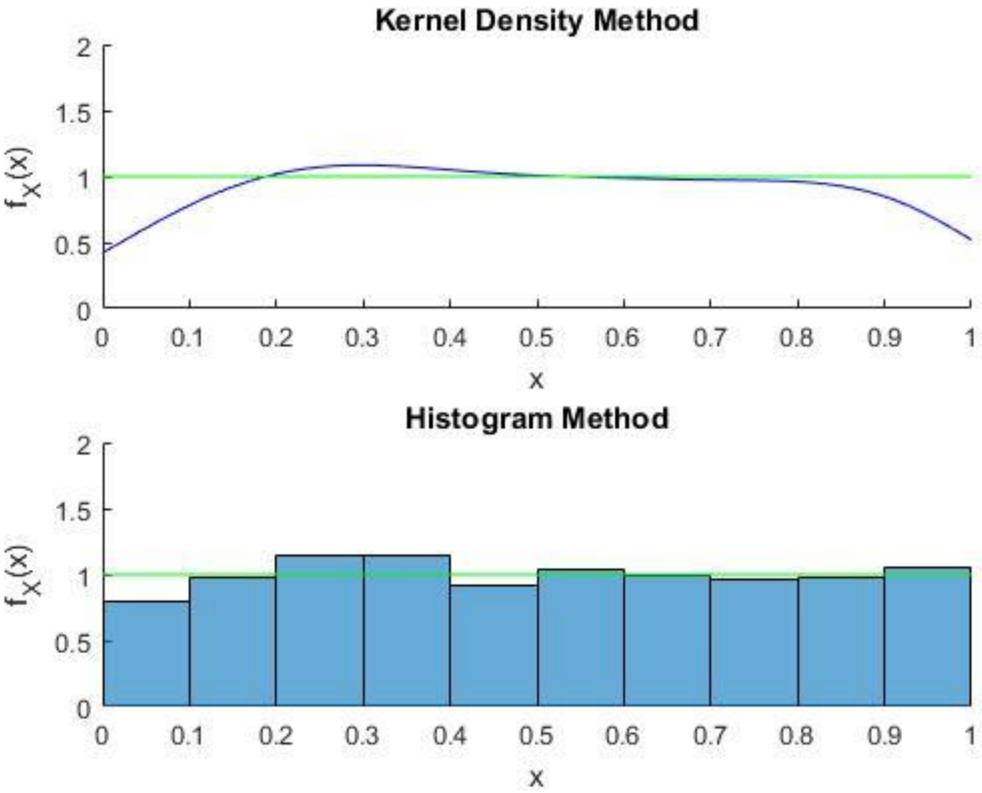
Kernel Density Estimation:

A Gaussian kernel is convolved with the distribution of randomly generated numbers. This can be visualized as a Gaussian kernel being flipped and, as it is shifted along the x-axis, adjacent values are multiplied and summed according to the Gaussian distribution to give a value for the smoothed distribution at that location. This can be thought of as a weighting and summing function where values closer to the center of the kernel hold more weight than those further away. The Gaussian kernel used is:

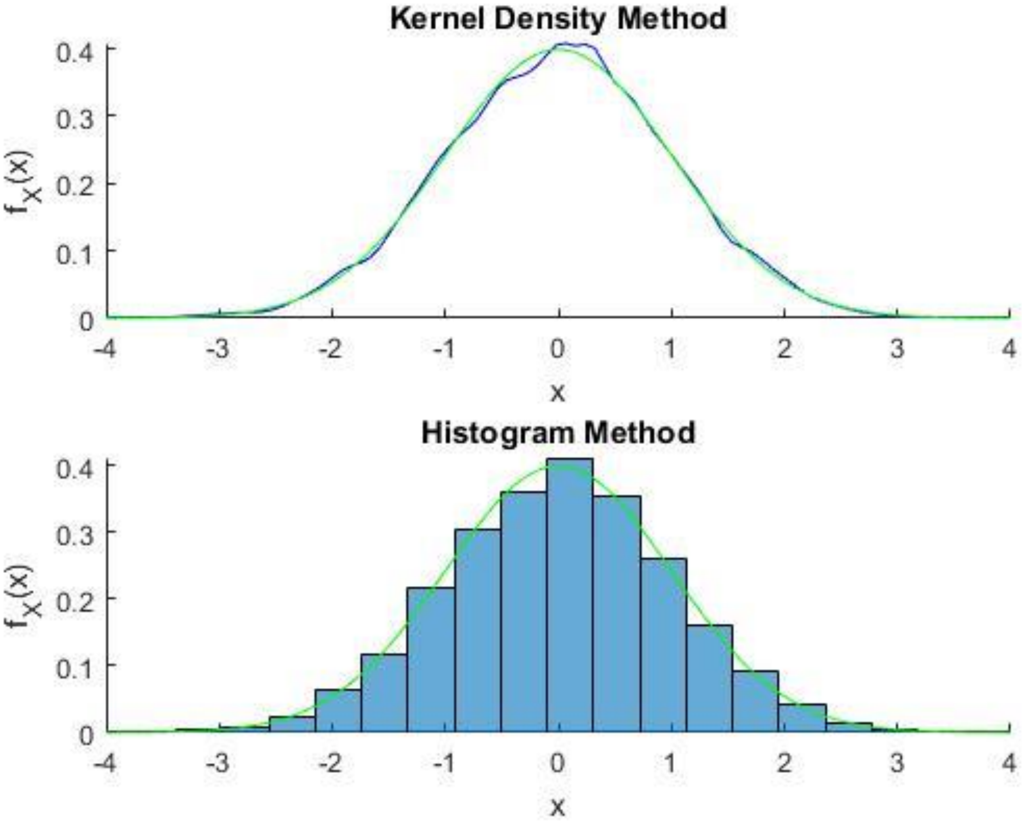
$$\pi_{ks}(x) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\sigma}\mathcal{K}\left(\frac{x - x^{(i}}{\sigma}\right)$$

(Equ. 4)

Where $\mathcal{K}(x) = \mathcal{N}(x|0, 1)$. Vectors of 1000 randomly generated numbers (both Gaussian and uniformly) were generated and the kernel smoothing and histogram methods were used on both to estimate the pdf of each. The Estimations can be seen in fig.1 and fig.2 with the expected pdf overlaid in green.


Kernel Density Method
2
1.5
1
0.5
0
$f_X(x)$
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
x
Histogram Method
2
1.5
1
0.5
0
$f_X(x)$
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
x

(Fig. 2)


Kernel Density Method
0.4
0.3
0.2
0.1
0
$f_X(x)$
-4
-3
-2
-1
0
1
2
3
4
x
Histogram Method
0.4
0.3
0.2
0.1
0
$f_X(x)$
-4
-3
-2
-1
0
1
2
3
4
x

(Fig. 2)

For the uniform distribution, the histogram method gives a much better estimate of the expected pdf because it enforces upper and lower bounds of 1 and 0, whereas, near the edges, the kernel density method averages with values outside the range of values generated by the random number generator and so the pdf (weighted average value as seen by the kernel) decreases to about 0.5 at the edges.

This, however, is only a disadvantage for bounded distributions and as such does not affect the estimate of a Gaussian pdf.

The histogram method has the disadvantage that its output is a discrete distribution and so to get the value of the distribution at some non-discrete value you need to interpolate the distribution between the two nearest points which is not ideal, especially since the bin probabilities are not exact either.

Conversely, this is less of a problem for the kernel density method as it returns a continuous function.

A disadvantage of both methods is that they are very dependent on the number of bins and the kernel size. Increasing the number of bins and decreasing the kernel size both lead to distributions that match the data more, but also means that there is a larger error between the bins/kernel distribution and the true distribution.

2.1.2 The Multinomial Distribution

The binomial distribution gives the probability of n successes in N independent Bernoulli trials and is given by:

$$\underbrace{\binom{N}{0} p^0 \cdot (1-p)^N}_{probability\ of\ zero\ successes} + \underbrace{\binom{N}{1} p^1 \cdot (1-p)^{N-1}}_{probability\ of\ one\ success} + \cdots + \underbrace{\binom{N}{N} p^0 \cdot (1-p)^N}_{probability\ of\ N\ successes} = 1$$

(Equ. 5)

Where $\binom{N}{n} = \frac{N!}{n!(N-n)!}$ is the binomial coefficient and is intuitively the number of ways to choose n elements from a set of N different & independent elements disregarding the order in which they are chosen. The probability mass function (the probability of the random variable X having k successes) is therefore one of the terms in the binomial distribution above.

$$p(X = n) = \binom{N}{n} p^n \cdot (1-p)^{N-n}$$

(Equ. 6)

The multinomial distribution is the generalization of the binomial distribution in which the number of possible outcomes of a trial is greater than two. Similarly, to the binomial distribution, the multinomial distribution gives the probability of n successes in N independent trials. Suppose there are k possible outcomes (as opposed to two – success or failure – in the binomial distribution) and each outcome, i, has a probability of p_i to occur. Suppose then that the outcome, i, occurs n_i times. The probability of this is simply:

$$p_1(X_i = n_i) = p_i^{n_i}$$

(Equ. 7)

Since the events are independent, the joint probability is simply the probabilities multiplied together:

$$p_1(X_1 = n_1, X_2 = n_2, \dots, X_k = n_k) = p_1^{n_1} p_2^{n_2} \dots p_k^{n_k}$$

(Equ. 8)

But there are multiple ways in which to choose n_1 of element 1 and n_2 of element 2 etc. Therefore, the total probability of choosing this is the sum of these. The total number of ways in which to pick these outcomes is given by the multinomial coefficient:

$$c = \frac{N!}{n_1!\,n_2!\ldots n_k!}$$

(Equ. 9)

The form of this is obvious from comparison with the binomial coefficient noting that on the denominator: $n + (N - n) = n_1 + n_2 \ = N$ is a special case with k = 2 of the general formula $n_1 + n_2 + n_3 + \cdots + n_k = N$.

Histograms can be considered to be multinomial distributions with each bin being a different element and therefore knowing the theoretical mean for the jth bin, μ_j, and the variance of the jth bin, σ_j^2, of the multinomial distribution is useful for histogram manipulation. These are given by:

$$\mu_j = Np_j = N\int_{c_j-\delta/2}^{c_j+\delta/2} p(x)dx$$

(Equ. 10)

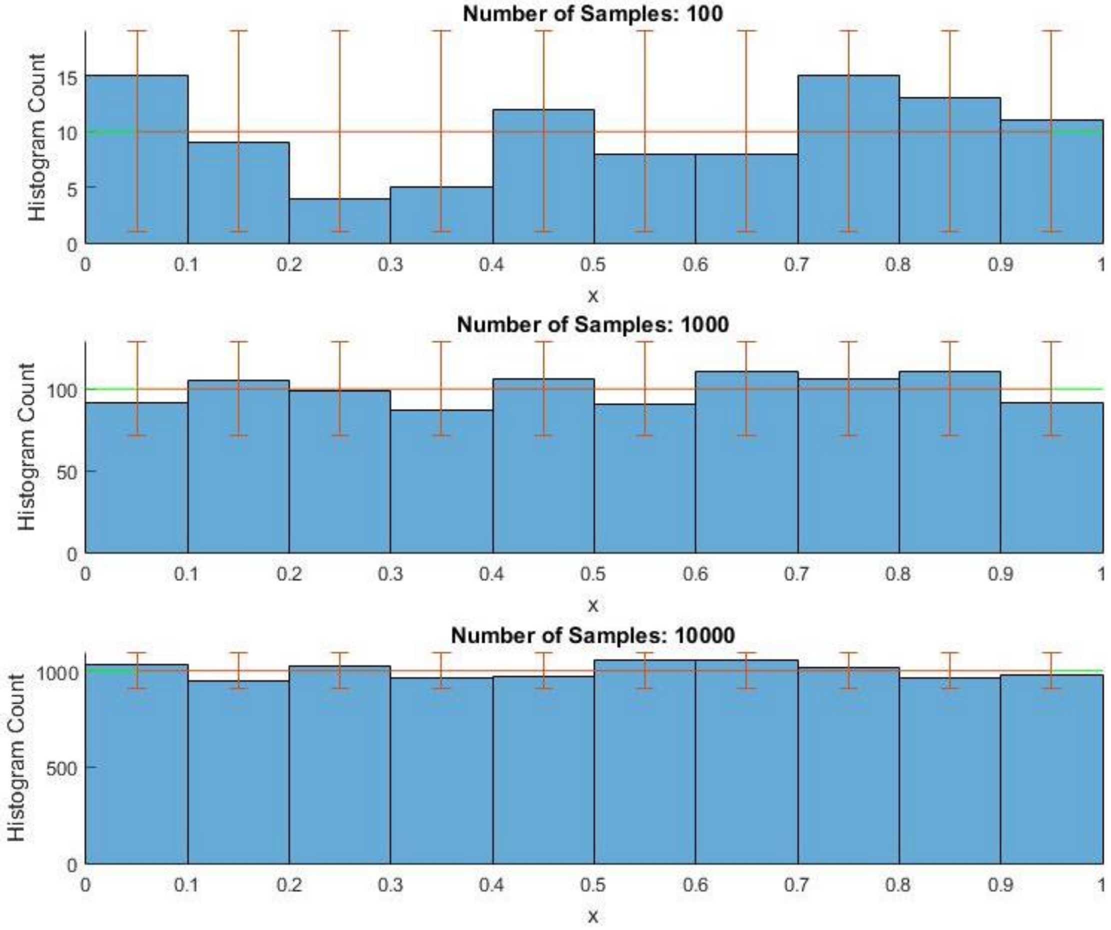
$$\sigma_j^2 = Np_j\left(1 - p_j\right)$$

(Equ. 11)

Where p_j is the probability that the sample, x_i, lies within the jth bin of the histogram and there are N samples generated in total. δ is the width of the jth bin.

2.1.3 The Uniform Distribution

If there are J bins, then for a uniform distribution between 0 and 1 the bin width and number of bins is related by $\delta = 1/\mathrm{J}$. Therefore the mean and variance are given by:



(Fig. 3)

$$\mu_j = N\int_{c_j-\delta/2}^{c_j+\delta/2} 1dx = N\delta = \frac{N}{J}$$

(Equ. 12)

$$\sigma_j^2 = N\delta(1-\delta) = \frac{N}{J^2}(J-1)$$

(Equ. 13)

Fig.3 Shows plots of histograms with varying sample sizes with error bars showing a ±3σ range. The standard deviation is proportional to $\sqrt{N}$ and so it increases with the number of samples, but the mean is proportional to N and so it increases faster. Therefore the deviation as a proportion of the mean decreases with $\frac{1}{\sqrt{N}}$ making it look as if the error is decreasing with increasing N. It is important to note that if the histogram data is normalized to find an estimate of the pdf then the deviation would also decrease with $\frac{1}{\sqrt{N}}$ as N increases. This means that the estimate becomes better as N increases.

The multinomial theory laid out above agrees well with intuition and the experimental data in fig.3

2.1.4 The Gaussian distribution

The mean and variance of the Gaussian distribution can be calculated in a similar way to that of the uniform distribution:

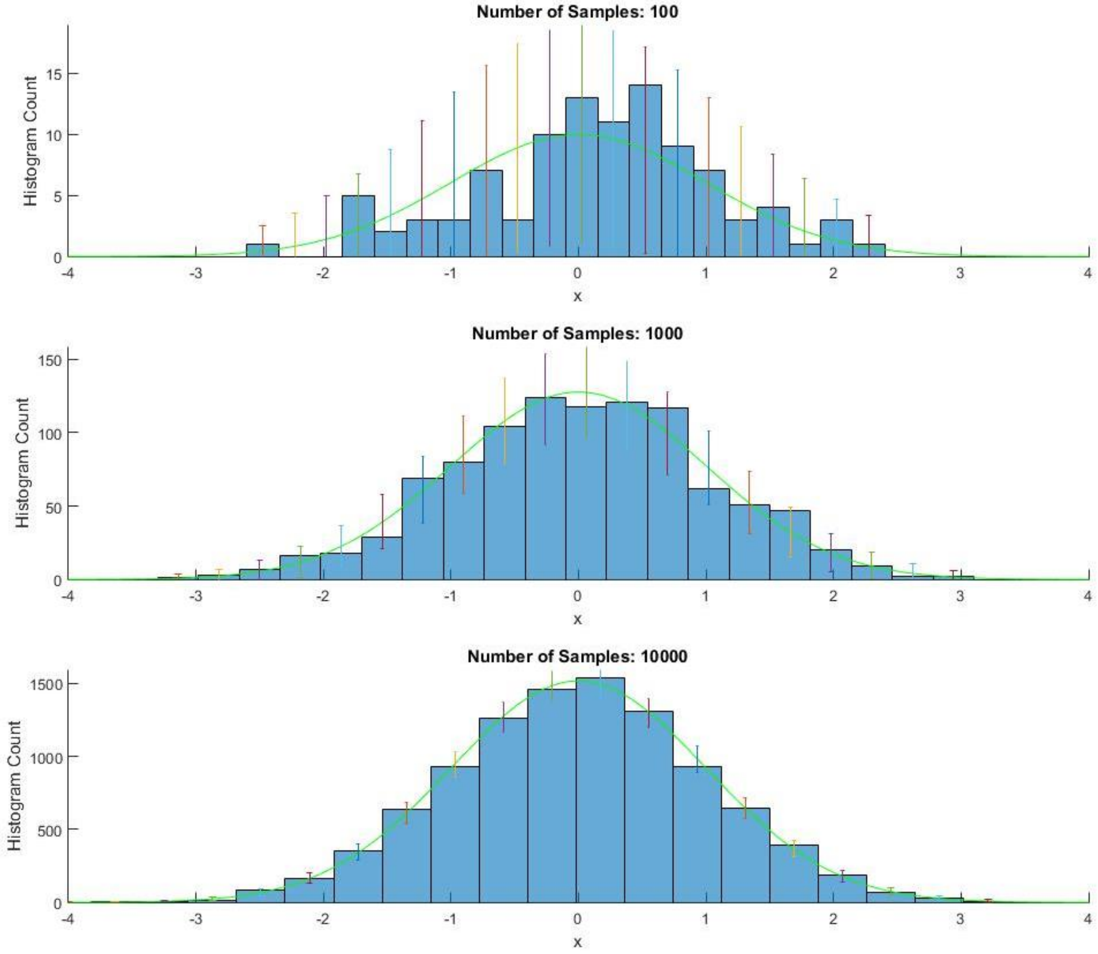
$$\mu_j = Np_j = N\int_{c_j-\delta/2}^{c_j+\delta/2} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}\, dx = N\left[\Phi\left(c_j + \frac{\delta}{2}\right) - \Phi\left(c_j - \frac{\delta}{2}\right)\right]$$

(Equ. 14)

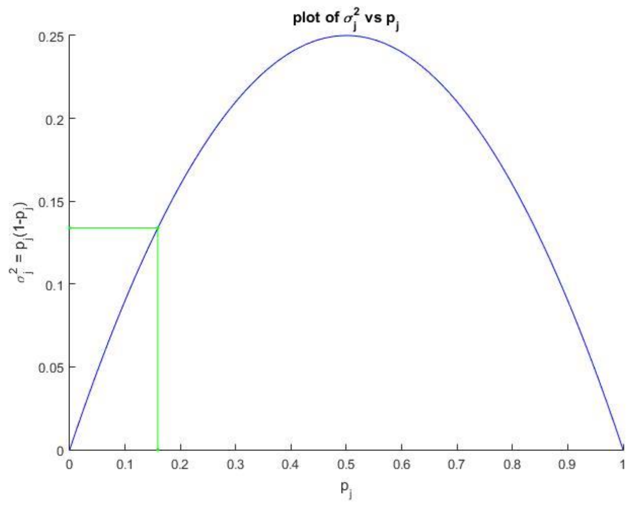
$$\sigma_j^2 = Np_j(1 - p_j)$$

(Equ. 15)

Fig.4 Shows plots of histograms with varying sample sizes with error bars showing a ±3σ range. Similarly to the uniform case, the mean varies with N and the standard deviation with $\sqrt{N}$ meaning that the relative error decreases with $\frac{1}{\sqrt{N}}$ again. The difference between the uniform and Gaussian case is that the probability p_j changes between bins and so σ_j^2 is not constant for each bin. The relationship between bin probabilities and the variance is given by fig.5 and also knowing



(Fig. 4)



(Fig. 5)

that the maximum bin probability is $\frac{1}{\sqrt{2\pi}} \times BinWidth \approx 0.16$ means that $\sigma_j^2 < 1.4$ which is a roughly linear region. Therefore the variance varies with bin probability as $\sigma_j^2 \approx kp_j$.

2.2 Functions of Random Variables

In probability, a useful thing to be able to do is to transform between domains of random variables. This may be used in practice to simplify difficult probability densities, for example, as in 2.2.1 the inverse transformation of this can be used to convert to the standard Gaussian distribution which is often tabulated. Another example might be to square the random values in order to find the magnitude of 'significant' events against background noise. When a random distribution is transformed, there is both an experimental method and a theoretical method to obtain the transformed distribution.

The experimental method is simply transforming the values given by the random number generator with the transformation function $f(x)$. The theoretical method involves substituting the x variables for that of the new y variable and then transforming with the Jacobian.

2.2.1 $Transformation\colon\ y = ax + b$

For normally distributed $\mathcal{N}(x|0,1)$ random variables, the transformation y = ax + b can be used to transform the pdf in the x domain to the y domain $\left(f_X(x) \rightarrow f_Y(y)\right)$.

$$f_X(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$$

(Equ. 16)

$$f_Y(y) = \sum_{k=1}^{n} f_X(x) \left|\frac{dx}{dy}\right|$$

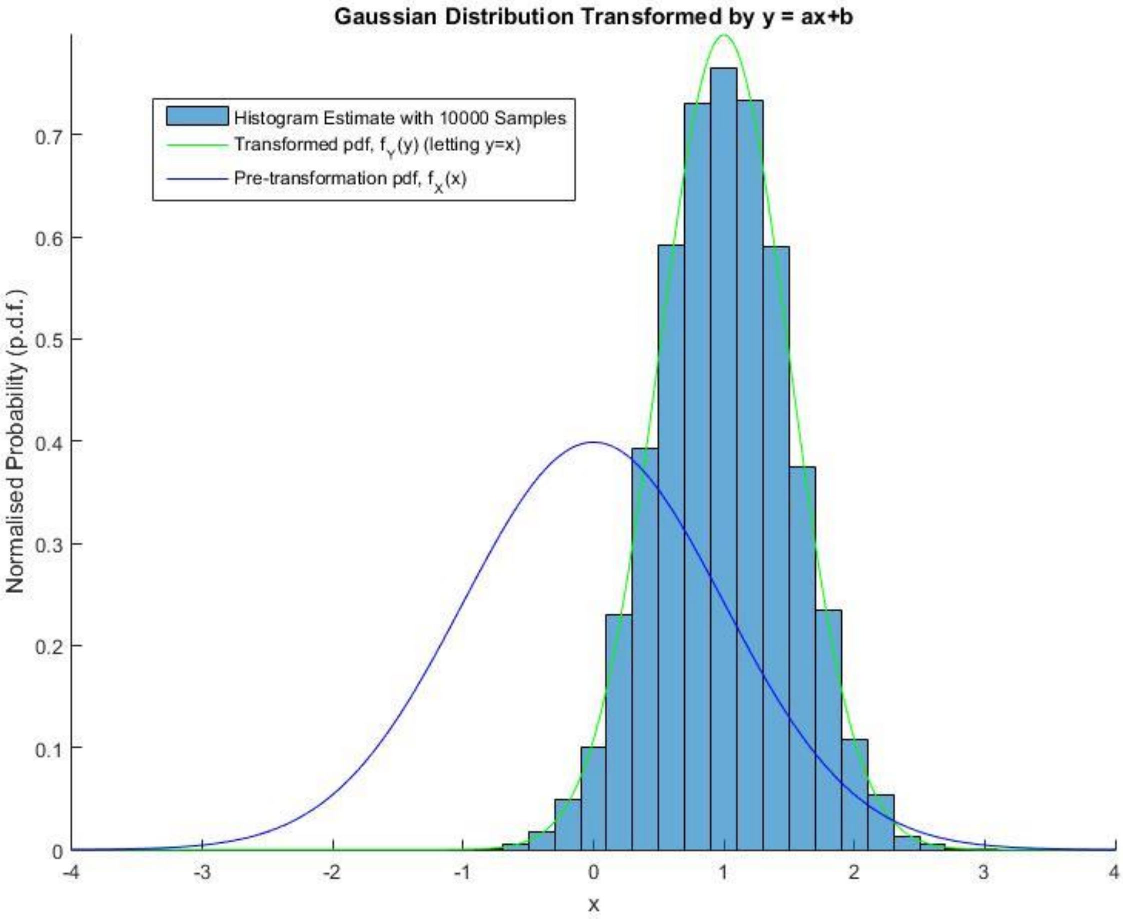
(Equ. 17)

$$y = ax + b, \qquad x = \frac{y-b}{a}, \qquad \left|\frac{dx}{dy}\right| = \frac{1}{a}$$

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

(Equ. 18)

Hence the transformed pdf is a Gaussian distribution with a variance of a^2 and a mean of b. Fig.6 shows the expected probability density functions when a = 0.5 and b = 1 overlaid against the experimental estimate of the histogram and the original standard Gaussian from which this was transformed. Note that in fig.6 x = y in the transformed substitution so that the original and transformed distributions can be plotted on the same graph



(Fig. 6)

2.2.2 $Transformation: \ y = x^2$

The probability distribution of a Gaussian transformed with $y = x^2$ can be calculated in a similar fashion:

$$f_x(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$$

(Equ. 19)

$$f_Y(y) = \sum_{k=1}^{n} \mathrm{f_x}(x) \left|\frac{dx}{dy}\right|$$

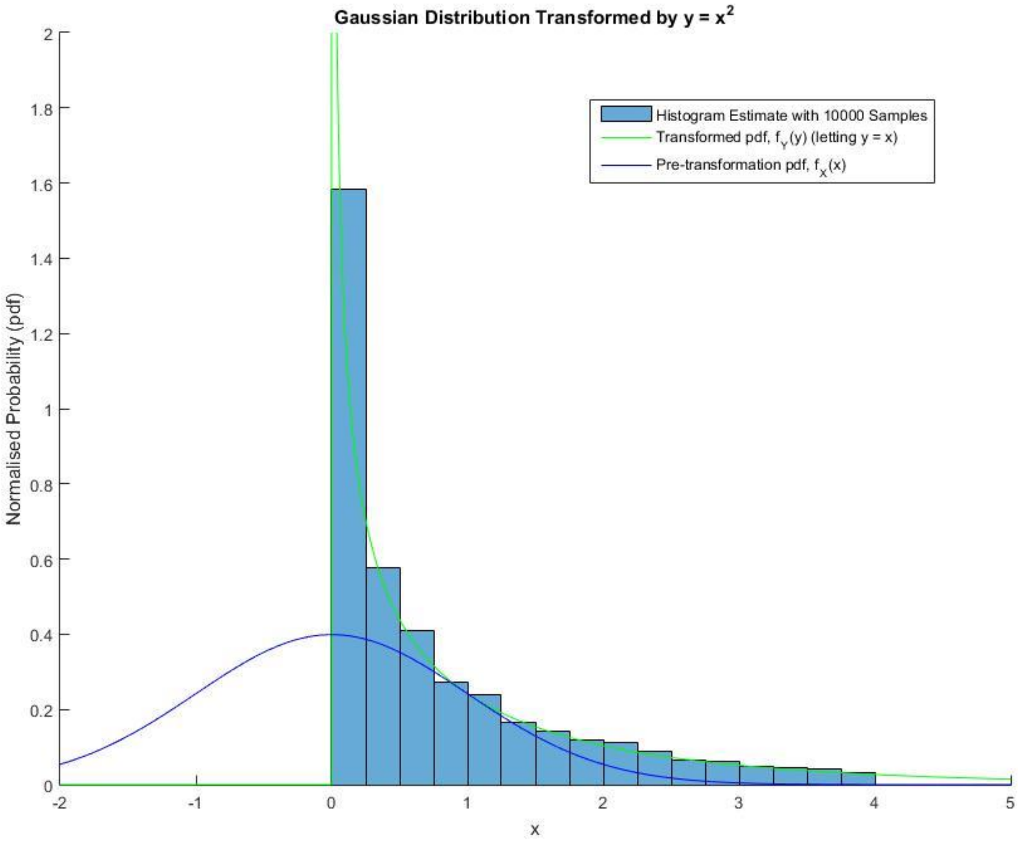
(Equ. 20)

$$y = x^2, \qquad x_{k=1} = \sqrt{y}\,, \qquad x_{k=2} = \sqrt{y}\,, \qquad \left|\frac{dx}{dy}\right| = \frac{1}{2\sqrt{y}}$$

$$f_Y(y) = \sum_{k=1}^{2} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(x_k)^2\right)\frac{1}{2\sqrt{y}} = \frac{1}{\sqrt{2\pi y}} \exp(-\frac{1}{2}y)$$

(Equ. 21)

Fig.7 shows the theoretical transformed distribution in green against the experimental histogram estimate and the standard Gaussian distribution from which both were transformed.



(Fig. 7)

2.2.3 $Transformation: y = \sin(x)$

Transforming a uniform distribution is slightly different because it is discontinuous. With a probability distribution given by:

$$\mathrm{f}_{\mathrm{X}}(x) = \mathcal{U}(x|0, 2\pi) = \begin{cases} \frac{1}{2\pi}, & 0 < x < 2\pi \\ 0, & otherwise \end{cases}$$

(Equ. 22)

And a using the transform $y = \sin(x)$ also has the problem that there are two values of x to which the function maps the uniform distribution to. The theoretical transformed distribution is derived as follows:

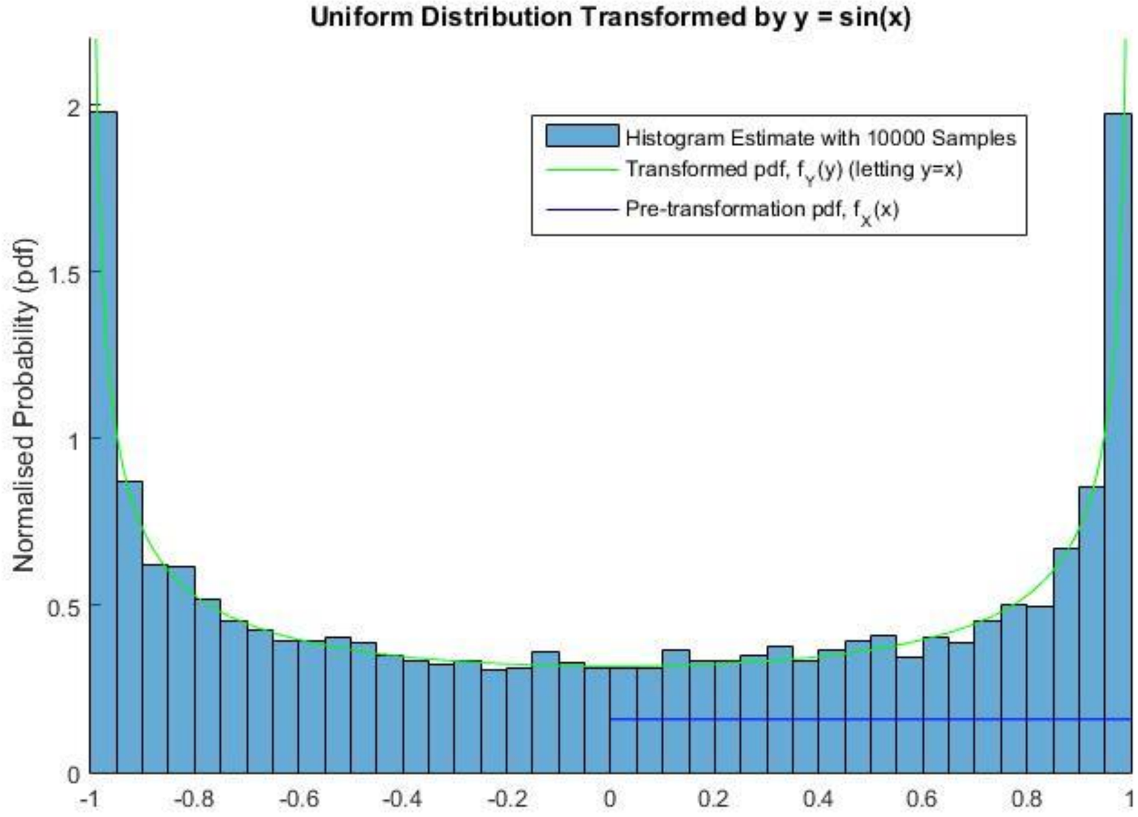
$$f_Y(y) = \sum_{k=1}^{n} \frac{1}{2\pi} \left|\frac{dx}{dy}\right|$$

(Equ. 23)

$$\left|\frac{dx}{dy}\right| = \frac{1}{\left|{}^{dy}/_{dx}\right|} = \frac{1}{\left|\frac{d}{dx}(\sin(x))\right|} = \frac{1}{|\cos(x)|} = \frac{1}{|\cos(\sin^{-1}(y))|} = \frac{1}{\left|\sqrt{1-y^2}\right|}$$

(Equ. 24)

Due to the fact that this is a one-to-many-function we need to sum to n=2.



(Fig. 8)

$$f_Y(y) = \sum_{k=1}^{2} \frac{1}{2\pi} \frac{1}{\left|\sqrt{1-y^2}\right|} = \frac{1}{\pi\left|\sqrt{1-y^2}\right|}, \qquad -1 < y < 1$$

(Equ. 25)

The probability density function of the pre-transformed, transformed and histogram estimated distribution are shown in fig.8

2.2.4 $Transformation$: $y = \min[\sin(x), 0.7]$

This limited sin function is an idealized version of clipping that can occur when electronics saturate. We can represent this as:

$$y = \begin{cases} 0.7, & \sin^{-1}(0.7) < x < \pi - \sin^{-1}(0.7) \\ \sin(x), & otherwise \end{cases}$$

(Equ. 26)

And transforming the uniform distribution from section 2.2.3 and using the same Jacobian:

$$\left|\frac{dx}{dy}\right| = \frac{1}{\left|\sqrt{1-y^2}\right|}$$

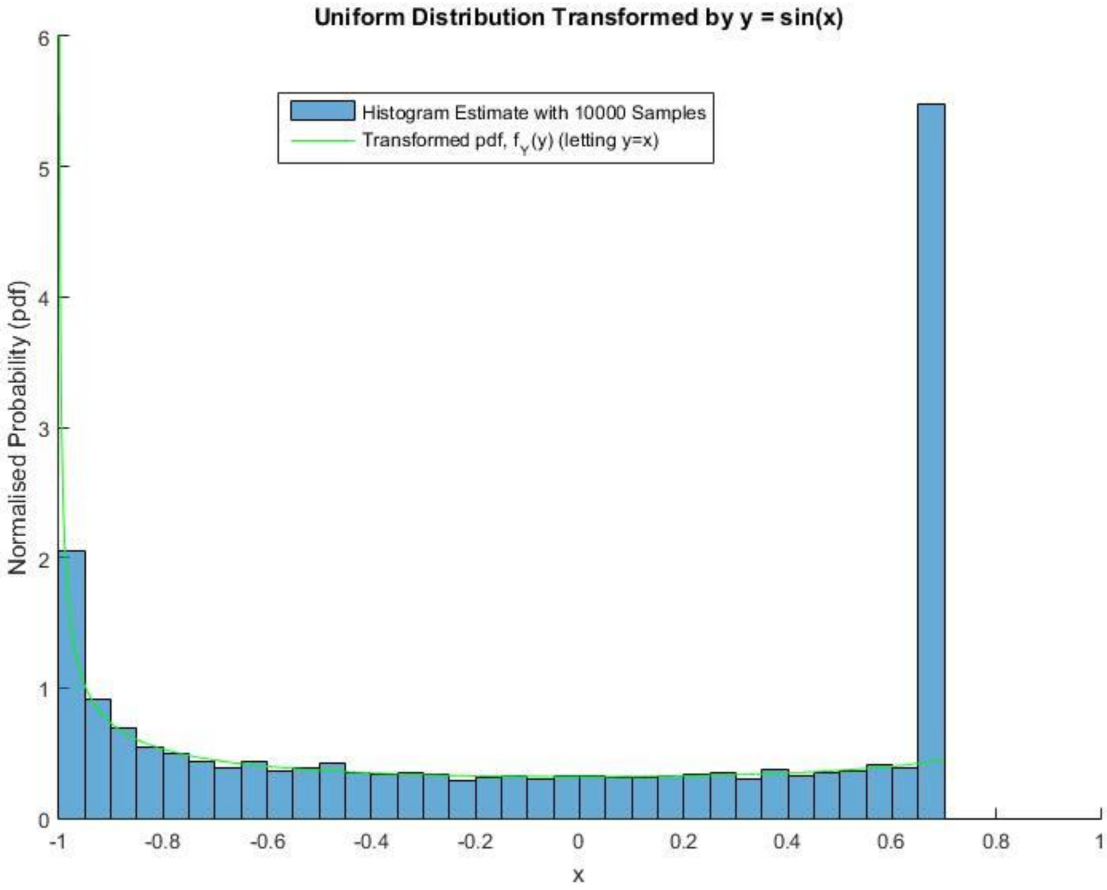
$$f_Y(y = 0.7) = 2 \cdot \int_{y=0.7}^{1} \frac{1}{2\pi} \frac{1}{\left|\sqrt{1-y^2}\right|} dy = \frac{1}{\pi}\left[\frac{\pi}{2} - \sin^{-1}(0.7)\right] \approx 0.2532, \qquad y = 0.7$$

(Equ. 27)

$$f_y(y) = \frac{1}{\pi\left|\sqrt{1-y^2}\right|}, \qquad -1 \le y < 0.7$$

(Equ. 28)

Note that the integral preserves the constraint of a total area of 1. An estimate of the bin height at y=0.7 can be obtained from the area of the pdf greater than 0.7 divided by a bin width $= \frac{0.2532}{0.05} = 5.06$. The normalized pdf of the clipped sine function is shown in fig.9 and it is clear that this estimate is close to the actual bin height.



(Fig. 9)

2.3 Inverse CDF method

The inverse CDF method is a very useful and general method of transforming random variables to a desired distribution. The inverse CDF of the desired distribution is calculated and can then be used in a similar same way to the 'Functions of random variables' method as seen in section 2.2.

2.3.1 Generating Random Samples of the Exponential Distribution from the Uniform Distribution

An example of this might be if we have a uniform random generator: $\mathrm{f_X}(x) = \mathcal{U}(x|0,1)$. Let's say that we want to generate the exponential distribution from this: $f_Y(y) = e^{-y}, \quad y \geq 0$. What is the algebraic transformation needed to generate this?

$$x = F_Y(y) = \int_0^y e^{-y} dy = [-e^{-y}]_0^y = 1 - e^{-y}$$

(Equ. 29)

Set the inverse distribution that you want to

$$F_Y(y) = fn(y) = F_X^{-1}(x)): \quad x = 1 - e^{-y}$$

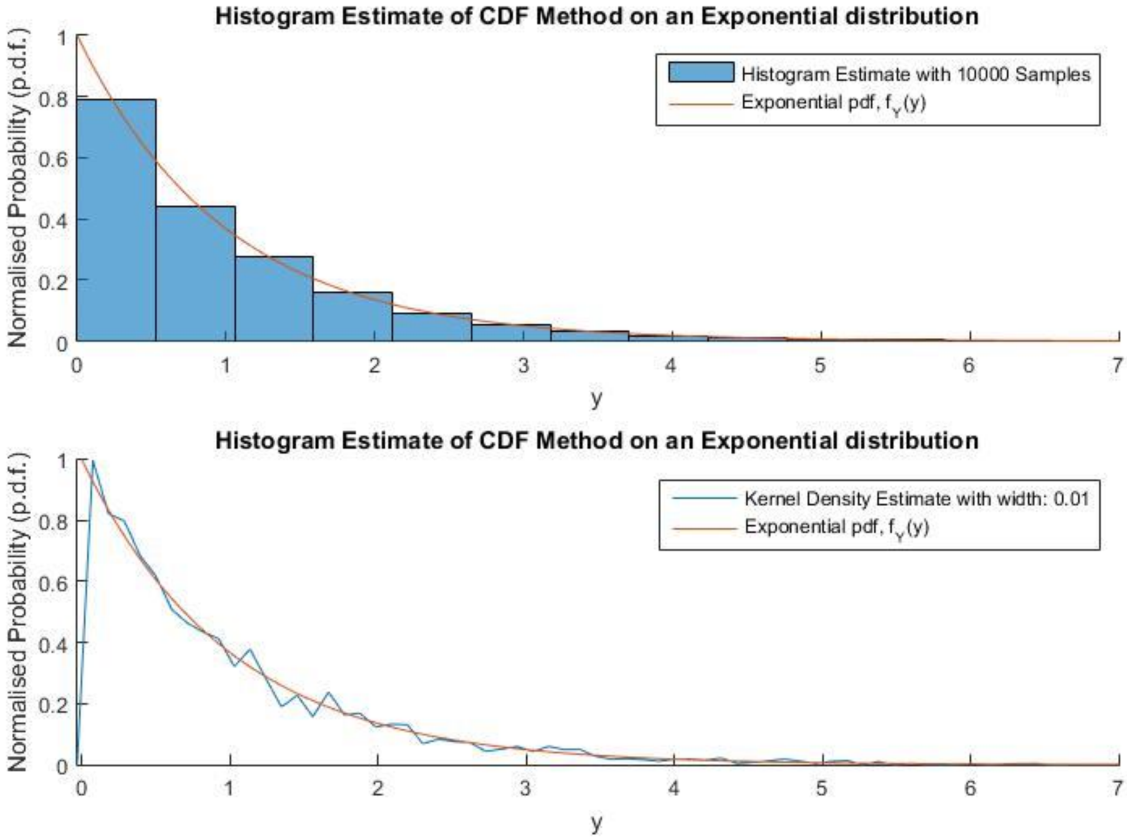
(Equ. 30)

Therefore rearranging to get the transformation from y to x:

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

(Equ. 31)

The outcome of this can be seen in fig.10 which compares kernel density and histogram estimates of this transformation with the expected exponential distribution.



(Fig. 10)

2.3.2 Using Monte Carlo to Estimate Means and Variances

Monte Carlo is a simple way to estimate the variance and mean of distributions by simply taking the arithmetic mean of the distribution and the arithmetic mean of the squared-values. Estimations of parameters are denoted as that parameter with a hat.

$$\mu = \mathbb{E}[X] = \int_{x-0}^{\infty} x f_X(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} y^{(i)} = \hat{\mu}$$

(Equ. 32)

$$\sigma^2 = \mathbb{E}[X^2] - \mathbb{E}[X]^2 = \int_{x-0}^{\infty} x^2 f_X(x)dx - \mu^2 \approx \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)}\right)^2 - \hat{\mu} = \hat{\sigma}^2$$

(Equ. 33)

We can show that $\hat{\mu}$ is unbiased using the linearity of the expectation of the random variable X.:

$$\mathbb{E}[\hat{\mu}] = \frac{1}{N}(\mathbb{E}[x_1] + \mathbb{E}[x_2] + \cdots + \mathbb{E}[x_N]) = \frac{1}{N}(N\mu) = \mu$$

(Equ. 34)

Additionally we can show that the expected mean squared error is inversely proportional to N and, therefore, that the Monte Carlo mean estimator has variance $\frac{\sigma^2}{N}$. This can be used to show that $\lim_{N\to\infty}\left(\frac{1}{N}\sum_{i=1}^{N} y^{(i)}\right) = \mu$.

$$\mathbb{E}[\hat{\mu}^2 - \mu^2] = \mathbb{E}[\hat{\mu}^2] - \mu^2 = \mathbb{E}\left[\left(\frac{1}{N}\sum_{i=1}^{N} y^{(i)}\right)^2\right] - \mu^2$$

(Equ. 35)

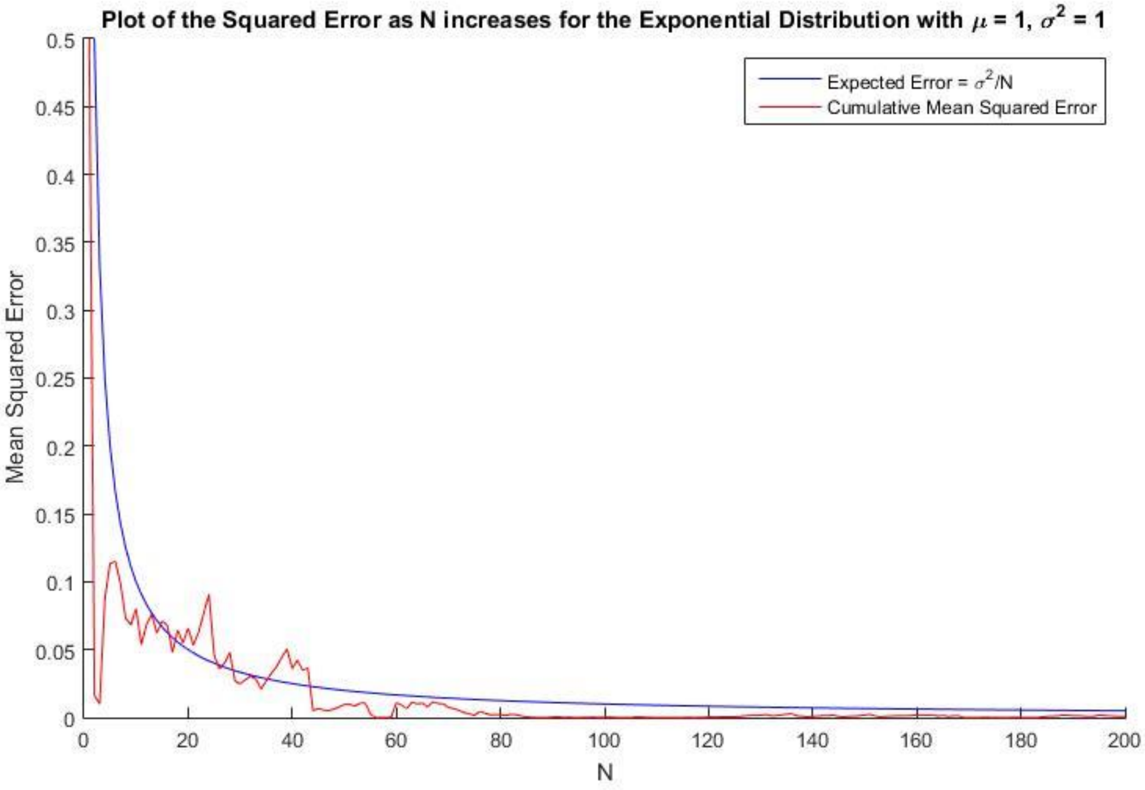
$$\rightarrow \mathbb{E}\left[\left(\frac{1}{N}\sum_{i=1}^{N} y^{(i)}\right)^2\right] = \frac{1}{N^2}\mathbb{E}\left[\left(\sum_{i=1}^{N} y^{(i)}\right)^2\right] = \frac{1}{N^2}\mathbb{E}\left[\sum_{i}^{N}\left(y^{(i)}\right)^2 + \sum_{i\neq j}^{N} y^{(i)}y^{(j)}\right]$$

$$= \frac{1}{N^2}\left\{\sum_{i=1}^{N}\mathbb{E}\left[y^{(i)}\right]^2 + \sum_{i\neq j}^{N}\mathbb{E}\left[y^{(i)}\right]\mathbb{E}\left[y^{(j)}\right]\right\} = \frac{1}{N^2}\{N(\sigma^2 + \mu^2) + (N^2 - N)\mu^2\} = \frac{\sigma^2}{N} + \mu^2$$

$$\therefore\ \mathbb{E}[\hat{\mu}^2 - \mu^2] = \frac{\sigma^2}{N} + \mu^2 - \mu^2 = \frac{\sigma^2}{N}$$

(Equ. 36)

This property is shown in fig,11 and displays that as N increases the Monte Carlo estimate becomes better and better as predicted by the theory above.



(Fig. 11)

2.4 Simulation from a `difficult' Density

2.4.1 Generation of the alpha-stable distribution

The alpha stable distribution is a non-analytically expressible distribution that can vary widely through the adjustment of a couple of parameters:

- The stability parameter: $\{0 < \alpha \leq 2\}$. The stability parameter is roughly correlated with the variance of the distribution (see fig.13), however, the variance is undefined because, except for special cases i.e. a Gaussian $(\alpha = 2)$ or Cauchy distribution $(\alpha = 1)$, all of the moments above $\mathbb{E}[X^1]$ are undefined. α is known as the stability parameter because if it, along with the skewness parameter, are the same for two independent distributions then any linear combination of the two will have the same distribution, $(same\ \alpha\ \&\ \beta)$ upto location and scale parameters.
- The Skewness parameter: $\{-1 \leq \beta \leq 1\}$ where negative values correspond to negative skew, zero gives no skew and positive values give positive skew. The skew of the alpha distribution is also undefined as $\mathbb{E}[X^3]$ is undefined. For the special case of $\beta = 0$, the first moment is the same as the mode and hence, is defined.

This method is useful for simulating a number of different distributions that may have similar - but not similar enough – distributions to a Gaussian $(\alpha = 2)$ or Cauchy distribution $(\alpha = 1)$. This is often the case for signal processing engineering and communications. The distribution is formed through the transformation of exponential and uniform distributions by first calculating two intermediate variables, a and b, and then substituting these and some randomly generated numbers into equ.39 to get the ‘difficult’ density, X.

$$b = \frac{1}{\alpha}\tan^{-1}\left(\beta\tan\left(\frac{\pi\alpha}{2}\right)\right), \qquad\qquad s = \left(1+\beta^2\tan^2\left(\frac{\pi\alpha}{2}\right)\right)^{\frac{1}{2\alpha}}$$

(Equ. 37a & 37b)

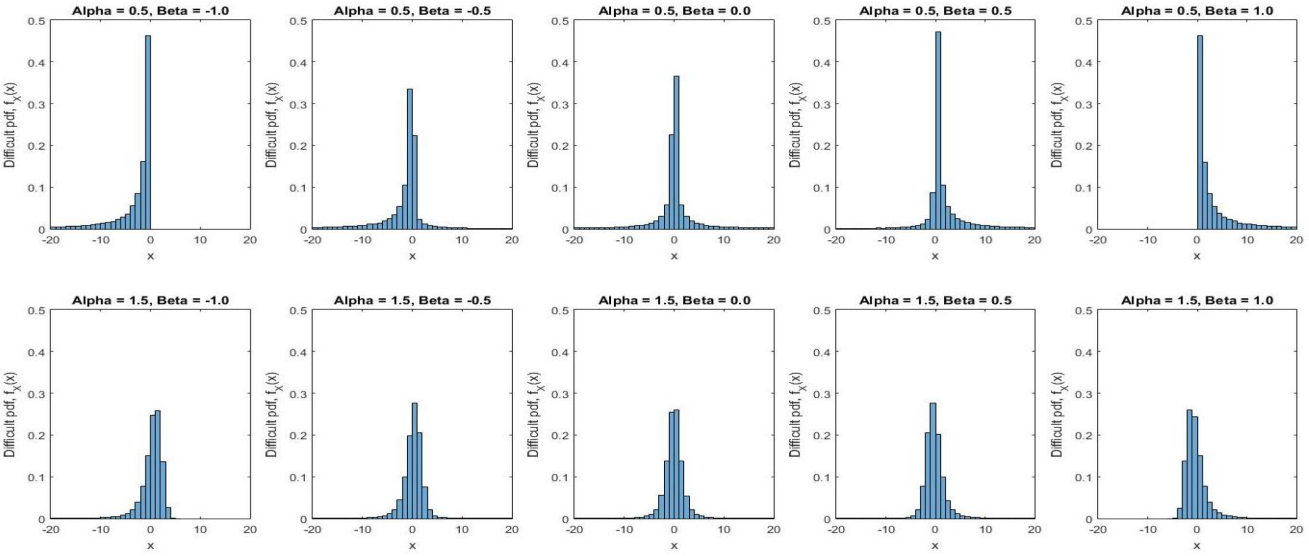
$$\mathcal{E}(x|1) = e^{-x}, \qquad \mathcal{U}(x|-\frac{\pi}{2},\frac{\pi}{2}) = \begin{cases}\frac{1}{\pi}, & -\frac{\pi}{2} < x < \frac{\pi}{2}\\ 0, & otherwise\end{cases}$$

(Equ. 38a & 38b)

$$X = \frac{s\sin\big(\alpha(U+b)\big)}{(\cos(U))^{\frac{1}{\alpha}}}\left(\frac{\cos\big(U - \ \alpha(U+b)\big)}{V}\right)^{\frac{1-\alpha}{\alpha}}$$

(Equ. 39)

Fig.12 shows the alpha stable distribution for different values of α and β.



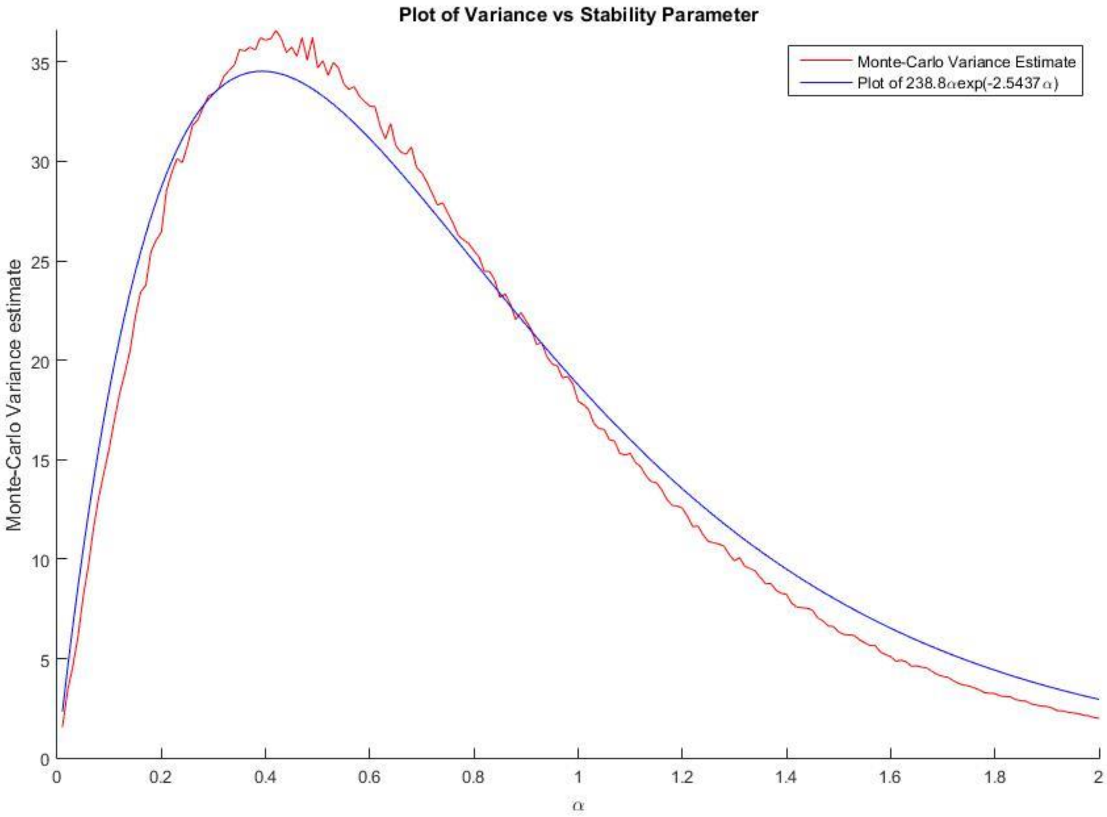
(Fig. 12)

2.4.2 Characteristics of the Alpha-Stable Distribution

From fig.12 we can guess that α has something to do with the spread of the data. Therefore we can use the Monte-Carlo estimator from section 2.3.2, equ.33 to estimate the variance and guess its relationship with α when β=0.

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)}\right)^2 - \hat{\mu}$$

(Equ. 40)



(Fig. 13)

Fig.14 shows the plot of alpha vs the variance estimate with N = 100,000 so as to reduce the estimation error as much as possible. The curve formed can be estimated roughly by a Boltzmann distribution with the formula:

$$\hat{\sigma}^2_{Monte\ Carlo} \approx 238.8\alpha e^{-2.5437\alpha}$$

(Equ. 41)

Therefore showing that, at least for non-skewed data, the stability is correlated with the stability parameter.

For a non-skewed distribution, ($\beta = 0$), the distribution simplifies to equ.40.

$$X = \frac{\sin\big(\alpha(U)\big)}{(\cos(U))^{\frac{1}{\alpha}}}\left(\frac{\cos(U(1-\ \alpha))}{V}\right)^{\frac{1-\alpha}{\alpha}}$$

(Equ. 42)

In addition to this as $\alpha \to 2$ the distribution can be simplified to:

$$X = \frac{\sin(2U)}{(\cos(U))^{\frac{1}{2}}}\left(\frac{\cos(-U)}{V}\right)^{-\frac{1}{2}} = \frac{sin(2U)}{\cos(U)}V^{\frac{1}{2}} = 2\sin(U)\,V^{\frac{1}{2}}$$

(Equ. 43)

We can simplify this further by noting that $\mathcal{E}(x|0) = -\ln(\ \mathcal{U}(x|0,1))$ and therefore:

$$V = \mathcal{E}(x|0) = -\ln\left[\pi \cdot \mathcal{U}\left(x\middle|-\frac{\pi}{2},\frac{\pi}{2}\right)\right] = -\ln(\pi U)$$

(Equ. 44)

$$X = 2\sin(U)\sqrt{-\ln(\pi U)} = \sqrt{2}\left(\sin(U)\sqrt{-2\ln(\pi U)}\right)$$

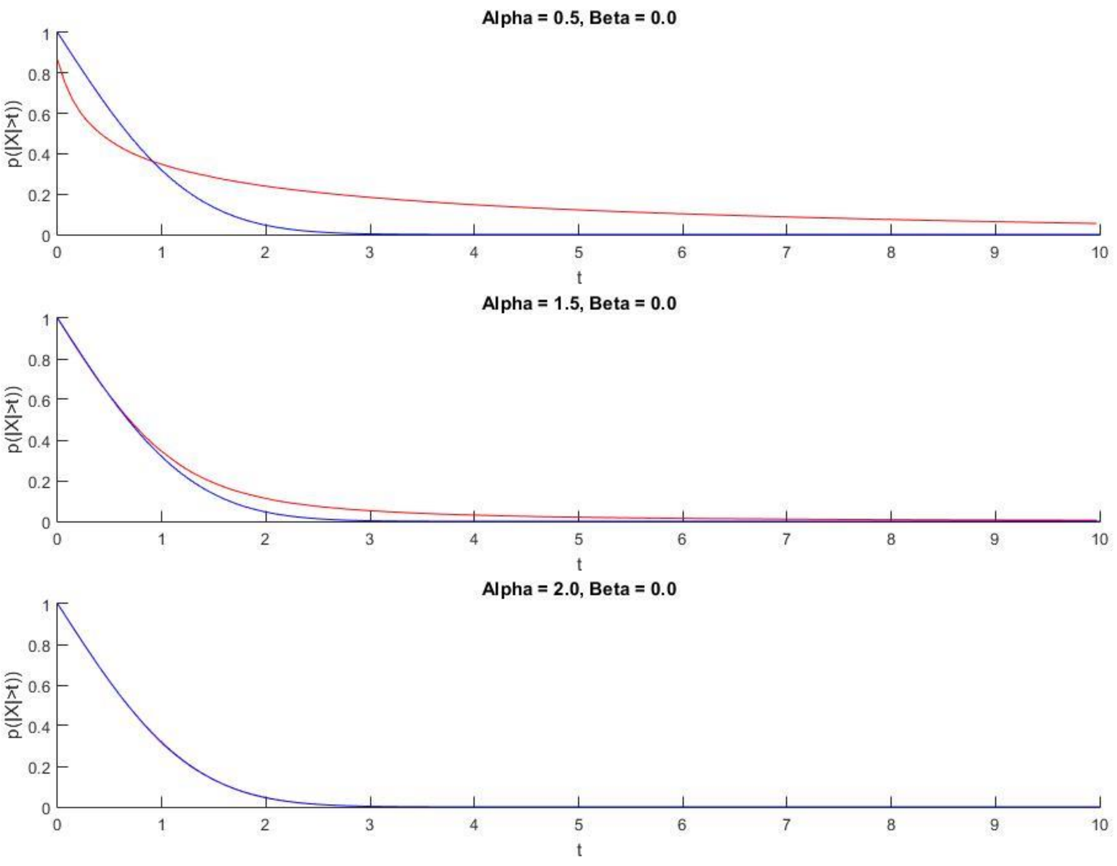
(Equ. 45)

This is known as the Box-Muller transformation for generating standard Gaussian distributions from uniform distributions and stretched horizontally by a factor of $\sqrt{2}$.

2.4.3 Tail Probabilities of the Alpha-Stable Distribution when $\beta = 0$

The tail probability of a distribution is simply the probability that the random variable $|X| > t$ where t is some value > 0. Note that by taking the absolute value of X this is both the negative and positive tail probabilities.

The tail probabilities as a function of t were plotted for different values of α and presented in fig.13 where the blue plot is the tail probability of a standard Gaussian for comparison and the red plot if for the alpha stable distribution. When $\alpha = 2$ it is clear that the tail probabilities are the same for the alpha stable, and Gaussian distributions. This is because when $\alpha = 2$ the alpha stable distribution becomes a Gaussian distribution as proven in equ.43. Note that the tail probabilities for the alpha stable distribution were scaled horizontally by a factor of $\sqrt{2}$.



(Fig. 14)

Values for t = 0, 3 and 6 have been picked from the graph and laid out in table 1 for comparison.

Table.1	$p(\lvert X\rvert > t)$		
Distribution	$t = 0$	$t = 3$	$t = 6$
$X, \alpha = 0.5, \beta = 0$	1.000	0.370	0.283
$X, \alpha = 1.5, \beta = 0$	1.000	0.104	0.034
$X \sim \mathcal{N}(x\lvert 0, 1^2)$	1.000	0.003	0.000

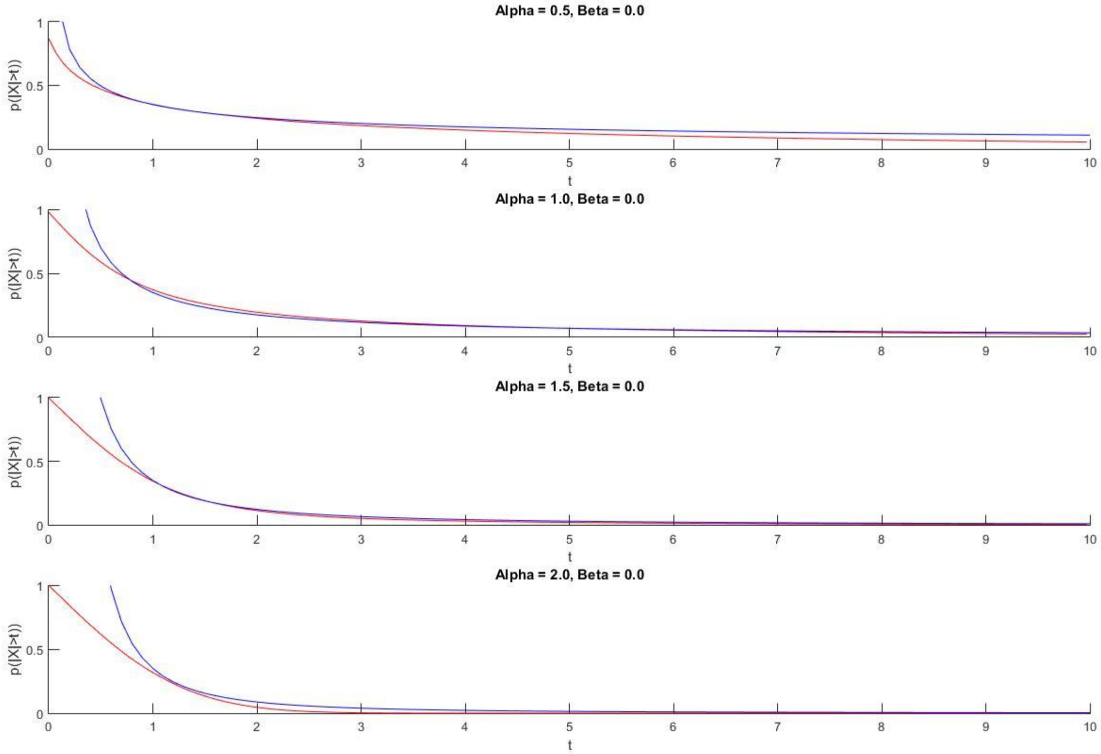
2.4.4 Estimation of Tail Probabilities with a Closed-Form Function

The tail probabilities (the red plots in fig.13) can be estimated with a closed form function of the form:

$$p(|X| > t) \approx cX^{\gamma}$$

(Equ. 46)

Where γ is some function of alpha to be determined and c is a constant. Guessing that $\gamma = -\alpha$ and then adjusting the constant c to fit the curves proved to be a good way of estimating the tail probabilities curve. The curves with $c = 0.35$ Are shown in fig.15 where the red curves are the actual tail probabilities and the blue curves are the estimates using equ.45. These closely match the tail probabilities for large values of t, and so is a good estimate.



(Fig. 15)

3 Conclusions

The report above can be cut into 4 distinct sections and each has its own conclusions which can be drawn.

3.1.1 Section 1

This section looked at two different methods of estimating the pdf of distributions, specifically the Gaussian distribution and the Uniform distribution. The two methods used were the Kernel Smoothing method and a histogram method.

The Kernel density method involved the convolution of a Gaussian kernel across the randomly generated numbers to give a weighted average. This method was better for estimating the Gaussian distribution because it output a smooth distribution estimate.
The histogram method could be thought of as a multinomial distribution in which the bins are events in which the random numbers can belong to. This method was found to be much better for estimating the uniform distribution because it is bounded.

For both estimation methods as the number of random numbers increase the error decreases with $\frac{1}{\sqrt{N}}$. And as such, more samples give better estimates.

3.1.2 Section 2

This section looked at methods of transforming one distribution to another using certain functions of random variables.

$$f_Y(y) = \sum_{k=1}^{n} f_X(x) \left|\frac{dx}{dy}\right|$$

(Equ. 47)

A formula can be used to transform between these distributions by using the Jacobian of the transformation $y = f(x)$.

3.1.3 Section 3

The most pressing problem with the Jacobian method laid out in section 2 was that there was no way to determine what the transformed distribution was without transforming it first. Therefore, a method of transforming from a uniform distribution to a desired distribution was described, this is known as the inverse CDF method. It involves simply taking the inverse CDF of the desired distribution and setting this at your function of random variables. The same method of finding the new pdf laid out in section 2 could then be used to find the transformed pdf, however, this is now obsolete as we already know what the transformed function will be. The use of this is finding a function for which to transform the pseudo –random numbers generated by the computer.

$$y^{(i)} = F_X^{-1}\left(x^{(i)}\right)$$

(Equ. 48)

In addition to the CDF method of generating random distributions, the Monte-Carlo method of estimating the mean and variance of random distributions was discussed. It was found that increasing the number of samples increases the accuracy of this method of estimation.

3.1.4 Section 4

In this section the alpha-stable distribution was discussed. This was a method of producing a probability density similar to a Gaussian but with a number of parameters to change it slightly. This made it very useful in the study of random processes that cannot simply be described by a Gaussian. The tail probabilities of the alpha-stable distribution was then calculated and it was found that the stability parameter was both correlated with the variance of the distribution and as it was increased, the distribution tended to a Gaussian.

4 Appendix (Matlab Code)

4.1 Q1 - Uniform and Gaussian Random Variables

```
clear all
close all
%%
%smoothed uniform distribution:
Num_U = 1000;                       %number random numbers generated
U1=rand(Num_U,1);                   %generate Num_U uniformly distributed random 
numbers in a 1xNum_U column vector
E_Ux = linspace(0, 1, Num_U);       %x components of expected uniform distribution
E_Uy = ones(Num_U, 1);              %expected distribution
```

```
[K_U, K_x] = ksdensity(U1, 'width', 0.1); %returns smoothed distribution K_U =
fn(K_x)
                                          %sets random distribution to smooth as
U1, with kernel width = 0.1

%plot histogramed uniform distribution
figure(1)
j = 1;
for Num_Uh = [100 1000 10000]
subplot(3, 1, j),                 %generate three vertically aligned subplots
U2 = rand(Num_Uh, 1);             %generate Num_Uh uniformly distributed random
numbers

hold on
H_U = histogram(U2, 10);                       %stores bin information in a struct
and plots 10 bins
plot(E_Ux, E_Uy.*(Num_Uh/H_U.NumBins) , 'g')   %plot expected distribution

i=1:1:10;                                    %calculate mean and std for each of
the 10 bars
mean(i) = Num_Uh/H_U.NumBins;
std(i) = sqrt(Num_Uh*H_U.BinWidth*(1-H_U.BinWidth)); %note division by average
numel per bin
errorbar((i/10)-0.05, mean, 3*std)           %plot 3*std errorbars for all 10 bars

axis([0 1 0 inf]);
ttl = sprintf('Number of Samples: %i', Num_Uh);
title(ttl);
ylabel('Histogram Count');
xlabel('x');
hold off

j=j+1;  %next subplot
end

%comparison of kernel density and histogram plot
figure(2)
subplot(2, 1, 1),
title('Kernel Density Method');
ylabel('f_X(x)');
xlabel('x');
axis([0 1 0 2]);

hold on
plot(K_x, K_U, 'b')
plot(E_Ux, E_Uy, 'g')
hold off


subplot(2, 1, 2),
title('Histogram Method');
ylabel('f_X(x)');
xlabel('x');
axis([0 1 0 2]);
```

```
hold on
H_U = histogram(U1, 10, 'Normalization', 'pdf');
plot(E_Ux, E_Uy, 'g')
hold off
%%




clear all
%smoothed gaussian distribution
Num_G = 10000;                              %number random numbers generated
G1 = randn(Num_G,1);                        %generate Num_U normally distributed
random numbers in a 1xNum_G column vector
E_Gx = linspace(-5, 5, Num_G);              %x components of expected Gaussian
distribution
E_Gy = (1/sqrt(2*pi)).*exp(-0.5*(E_Gx.^2)); %y components of expected Gaussian
distribution

[K_G, K_x] = ksdensity(G1, 'width', 0.1);  %returns smoothed distribution K_G =
f(K_x)
                                            %sets random distribution to smooth as
G1, with kernel width = 0.1

%plot histogrammed normal distribution:
figure(3)
j = 1;
for Num_Gh = [100 1000 10000]
subplot(3, 1, j),            %generate three vertically aligned subplots
G2 = randn(Num_Gh, 1);       %generate Num_Gh uniformly distributed random numbers

hold on
H_G = histogram(G2, 20, 'Normalization', 'count');  %gives a histogram where the
heights are the number of elements per bin
plot(E_Gx, E_Gy*H_G.BinWidth*Num_Gh, 'g')            %plot scaled Gaussian
distribution (use bin width and not num bins because delta =! 1/J anymore)222
start = H_G.BinLimits(1);                    %define the x coordinate of the edge of
the far left bin

mean = zeros(1, 20);  std = zeros(1, 20); prob = zeros(1, 20);

    for i=1:1:20 %calculate mean and std for each of the 20 bars
        binStart = start+((i-1)*H_G.BinWidth);  %left edge coordinate of bin i
        binEnd = start+(i*H_G.BinWidth);        %right edge coordinate of bin i
        q = normcdf([binStart binEnd]);         %returns the cdf at x=binStart and
x=binEnd of a gaussian
        prob(i) = q(2) - q(1);                  %returns p(binStart < x < binEnd)


        mean(i) = prob(i)*Num_Gh;               %for un-normalised histogram mean,
multiply by Num_Gh
                                                %note, this is the theoretical
mean, not the experimental one
        std(i) = sqrt(Num_Gh*prob(i)*(1-prob(i)));
        errorbar(binStart+(H_G.BinWidth/2), mean(i), 3*std(i))          %plot 3*std
errorbars for all 20 bars
```

```
    end
ttl = sprintf('Number of Samples: %i', Num_Gh);
title(ttl);
ylabel('Histogram Count');
xlabel('x');
axis([-4 4 0 inf])      %sets ymin=0 but allows for auto ymax

hold off
j=j+1;
end

%comparison of kernel density and histogram plot
figure(4)
subplot(2, 1, 1),
title('Kernel Density Method');
ylabel('f_X(x)');
xlabel('x');
axis([-4 4 0 inf]);

hold on
plot(K_x, K_G, 'b')
plot(E_Gx, E_Gy, 'g')
hold off

subplot(2, 1, 2),
title('Histogram Method');
ylabel('f_X(x)');
xlabel('x');
axis([-4 4 0 inf]);

hold on
H_G = histogram(G1, 20, 'Normalization', 'pdf');
plot(E_Gx, E_Gy, 'g')
hold off

clear all
figure(5) %plot of bin probabilities vs variance

p = linspace(0,1, 200);
one = ones(1, 200);
var = p.*(one - p);
p_max = 1/(2*pi);

hold on
plot(p, var, 'b')
plot([0 p_max p_max], [p_max*(1-p_max) p_max*(1-p_max) 0], 'g.-')
hold off

xlabel('p_j')
ylabel('\sigma_j^2 = p_j(1-p_j)')
title('plot of \sigma_j^2 vs p_j')
```

4.2 Q2 - Functions of Random Variables

```
close all
clear all

%question 2a y=ax+b
Num_a = 10000; a = 0.5; b = 1;
E_xa = linspace(-10, 10, 1000);  %x components of expected transformed pdf
f_xa=randn(Num_a,1);            %generates Num_a normally distributed random
numbers in a 1xNum_a column vector

T_ya= a*f_xa + b;                %transforms random numbers in 'x' to 'y'
E_fy = (1/sqrt(2*pi))*exp(-0.5*((E_xa-b)/a).^2)/a;   %expected transformed
distribution

figure(1)   %plot graphs
hold on
H_a= histogram(T_ya, 20, 'normalization', 'pdf');    %normalised histogram of
tranformed distribution
plot(E_xa, E_fy, 'g')   %plot transformed distribution
plot(E_xa, (1/sqrt(2*pi))*exp(-0.5*((E_xa)).^2), 'b') %plot distribution before
transformation
hold off

title('Gaussian Distribution Transformed by y = ax+b'); %format plots
xlabel('x');
ylabel('Normalised Probability (p.d.f.)');
axis([-4 4 0 inf]);
txt = sprintf('Histogram Estimate with %i Samples', Num_a);
legend(txt, 'Transformed pdf, f_Y(y) (letting y=x)', 'Pre-transformation pdf,
f_X(x)')

clear all




%question 2b y=x^2
Num_b = 10000;
E_xb = linspace(-10, 10, Num_b); %x components of expected transformed pdf
f_xb=randn(Num_b,1);    %generates Num_b normally distributed random numbers in a
1xNum_b column vector
```

```
T_yb = f_xb.^2;                                 %transforms random numbers in 'x'
to 'y'
E_fy = (1./sqrt(2*pi*E_xb)).*exp(-0.5*E_xb);    %expected transformed distribution

figure(2)
hold on
H_b = histogram(T_yb, 20, 'Normalization', 'pdf', 'BinLimits', [-1 4]);      %plot
normalised histogram plot
plot(E_xb, E_fy, 'g')                                   %plot transformed
distribution
plot(E_xb, (1/sqrt(2*pi))*exp(-0.5*((E_xb)).^2), 'b')   %plot distribution before
transformation
hold off

title('Gaussian Distribution Transformed by y = x^2'); %format plots
xlabel('x');
ylabel('Normalised Probability (pdf)');
axis([-2 5 0 2]);
txt = sprintf('Histogram Estimate with %i Samples', Num_b);
legend(txt, 'Transformed pdf, f_Y(y) (letting y = x)', 'Pre-transformation pdf,
f_X(x)')

clear all




%FTR (c) y = sin(x)
Num_c = 10000; Uni = 1/(2*pi).*ones(Num_c, 1);

E_xc = linspace(0, 1/2*pi, Num_c);%xcomponents of pre-transformed pdf
E_yc = linspace(-1, 1, Num_c);  %x components of expected transformed pdf

f_xc = (2*pi).*rand(Num_c,1);    %generates Num_c uniformly distributed random
numbers between 0&2*pi
                                   %so cdf=1 for interval 0<x<1/2pi (E_xc)

T_yc = sin(f_xc);                  %transforms random numbers in 'x' to 'y'
E_fy = (1/pi)./abs(sqrt(1-E_yc.^2));       %expected transformed distribution

figure(3)
hold on
H_c = histogram(T_yc, 40, 'Normalization', 'pdf', 'BinLimits', [-1 1]); %plots
histogram estimation
plot(E_yc, E_fy, 'g')       %plots transformed distribution
plot(E_xc, Uni, 'b')        %plots original transformation
hold off

axis([-1 1 0 2.2]);
title('Uniform Distribution Transformed by y = sin(x)'); %format plots
xlabel('x');
ylabel('Normalised Probability (pdf)');
txt = sprintf('Histogram Estimate with %i Samples', Num_c);
```

```
legend(txt, 'Transformed pdf, f_Y(y) (letting y=x)', 'Pre-transformation pdf,
f_X(x)')

clear all
```

```
%FTR (d) y = min(sin(x), 0.7)
Num_d = 10000; Uni = 1/(2*pi).*ones(Num_d, 1);

E_xd = linspace(0, 1/2*pi, Num_d);%xcomponents of pre-transformed pdf
E_yd = linspace(-1, 0.7, Num_d);  %x components of expected transformed pdf

f_xd = (2*pi).*rand(Num_d, 1);    %generates Num_c uniformly distributed random
numbers between 0&2*pi

T_yd = min(sin(f_xd), 0.7);                  %transforms random numbers in 'x' to
'y'
E_fy = (1/pi)./abs(sqrt(1-E_yd.^2));       %expected transformed distribution

figure(4)
hold on
H_c = histogram(T_yd, 40, 'Normalization', 'pdf', 'BinLimits', [-1 1]); %plots
histogram estimation
plot(E_yd, E_fy, 'g')       %plots transformed distribution
hold off

axis([-1 1 0 6]);
title('Uniform Distribution Transformed by y = sin(x)'); %format plots
xlabel('x');
ylabel('Normalised Probability (pdf)');
txt = sprintf('Histogram Estimate with %i Samples', Num_d);
legend(txt, 'Transformed pdf, f_Y(y) (letting y=x)')
```

4.3 Q3 - Inverse CDF method

```
clear all
close all

%question 3 exponential distribution
figure(1)
Num = 10000;
x=rand(Num,1);          %set up uniform distribution
w=0:0.01:7;
T = -log(1-x);

subplot(211),
hold on
U = histogram(T, 20, 'Normalization', 'pdf');  %plot histogram estimate of pdf
plot(w, exp(-w))                               %overlay exponential distribution
hold off
```

```
title('Histogram Estimate of CDF Method on an Exponential distribution'); %format
plots
xlabel('y');
ylabel('Normalised Probability (p.d.f.)');
axis([0 7 0 inf]);
txt = sprintf('Histogram Estimate with %i Samples', Num);
legend(txt, 'Exponential pdf, f_Y(y)')


subplot(212),
width = 0.01;
hold on
ksdensity(T, 'width', width)    %plot smoothed kernel
plot(w, exp(-w))                %overlay exponential distribution
hold off

title('Histogram Estimate of CDF Method on an Exponential distribution'); %format
plots
xlabel('y');
ylabel('Normalised Probability (p.d.f.)');
axis([-inf 7 0 inf]);
txt = sprintf('Kernel Density Estimate with width: %.2f', width);
legend(txt, 'Exponential pdf, f_Y(y)')



%FTR plot of mean squared error
N=200; spacing = 1;
ux=rand(N, 1);              %set up uniform distribution with N random values
Random_exp = -log(1-ux);            %get random samples

n = 1:1:N;
one_vector = ones(1, N);    %allows calculation of cumulative 1/n vector

MC_mean = (one_vector./n)'.*cumsum(Random_exp);     %monte carlo mean u(n) =
(1/n)*sum(random variables up to n)
                                                    %note ' = transpose, and this
is used because otherwise dimensions arent equal
%var_exp = (one_vector./n)'.*cumsum(Random_exp.^2) - MC_mean.^2;
error = (MC_mean' - 1).^2;                          %the arithmetic (monte carlo)
mean minus the theoretical mean (1)

figure(2)
hold on
plot(n, one_vector./n, 'b')
plot(n, error, 'r')
hold off

title('Plot of the Squared Error as N increases for the Exponential Distribution
with \mu = 1, \sigma^2 = 1'); %format plots
xlabel('N');
ylabel('Mean Squared Error');
legend('Expected Error = \sigma^2/N', 'Cumulative Mean Squared Error')
axis([0 N 0 0.5]);
```

4.4 Q4 - Simulation from a `difficult' Density

```
close all
clear all

%question 4

%plot differing values of alpha and beta to compare effects
figure(1)
j=1;
for alpha=[0.5 1.5]      %Plot different values of alpha
    for beta=-1:0.5:1    %Plot different values of beta

    Num = 100000;

    b = atan(beta*tan(pi*alpha/2))/alpha;           %difficult density algebra
    s =(1+(beta*tan(pi*alpha/2))^2)^(1/(2*alpha));

    U_rand = -pi/2 + pi*rand(Num, 1);   %generate random numbers between -pi/2 < x
< pi/2
    E_rand = exprnd(1, Num, 1);         %generate exponential distribution with
mean = 1

    X = s*(sin(alpha*(U_rand+b))./(cos(U_rand)).^(1/alpha)).*((cos(U_rand -
alpha*(U_rand+b)))./E_rand).^((1-alpha)/alpha);     %generate 'difficult' density

    X(abs(X) > 30) = 0;             %replace values of X > 30 with zero (removes
outliers)
    graph = sprintf('Alpha = %0.1f, Beta = %0.1f', alpha, beta);

    subplot(2, 5, j),               %plot histograms
    histogram(X, 'BinWidth', 1, 'normalization', 'pdf')

    title(graph)
    xlabel('x')
    ylabel('Difficult pdf, f_X(x)')
    axis([-20 20 0 0.5])       %graph size limits so comparable

    j = j+1;

    end
end

clear all





%relationship between alpha and the variance
figure(2)
Num = 100000;   beta = 0;
```

```
i=1;
for alpha=0.01:0.01:2     %Plot different values of alpha

%same calculation as fisrt part
b = atan(beta*tan(pi*alpha/2))/alpha;          %difficult density algebra
s =(1+(beta*tan(pi*alpha/2))^2)^(1/(2*alpha));

U_rand = -pi/2 + pi*rand(Num, 1);   %generate random numbers between -pi/2 < x <
pi/2
E_rand = exprnd(1, Num, 1);         %generate exponential distribution with mean =
1

X = s*(sin(alpha*(U_rand+b))./(cos(U_rand)).^(1/alpha)).*((cos(U_rand -
alpha*(U_rand+b)))./E_rand).^((1-alpha)/alpha);     %generate 'difficult' density
X_abs = abs(X);
X(abs(X) > 30) = 0;             %replace values of X > 30 with zero (removes
outliers)



%Monte-Carlo variance estimator
q = 1:1:Num;
one_vector = ones(1, Num);    %allows calculation of cumulative 1/n vector
MC_mean = (one_vector./q)'.*cumsum(X);     %monte carlo mean u(n) =
(1/n)*sum(random variables up to n)
MC_var = (one_vector./q)'.*cumsum(X.^2) - MC_mean.^2;   %monte carlo variance
estimate

variance_est(i) = MC_var(numel(MC_var)-1);

i = i+1;    %next value
end

alpha=0.01:0.01:2;
hold on
plot(alpha, variance_est, 'r')
plot(alpha, 238.8.*alpha.*exp(-2.5437.*alpha), 'b')
hold off


title('Plot of Variance vs Stability Parameter')
legend('Monte-Carlo Variance Estimate', 'Plot of 238.8\alphaexp(-2.5437\alpha)')
xlabel('\alpha')
ylabel('Monte-Carlo Variance estimate')
axis([0 inf 0 inf])





clear all
```

```
%plot tail probabilities
figure(3)
Num = 100000;   beta = 0; alpha = [0.5 1 1.5 2];


for i=1:1:numel(alpha)    %Plot different values of alpha

%same calculation as last part
b = atan(beta*tan(pi*alpha(i)/2))/alpha(i);          %difficult density algebra
s =(1+(beta*tan(pi*alpha(i)/2))^2)^(1/(2*alpha(i)));

U_rand = -pi/2 + pi*rand(Num, 1);   %generate random numbers between -pi/2 < x <
pi/2
E_rand = exprnd(1, Num, 1);         %generate exponential distribution with mean =
1

X_abs = abs(s*(sin(alpha(i)*(U_rand+b))./(cos(U_rand)).^(1/alpha(i))).*((cos(U_rand
- alpha(i)*(U_rand+b)))./E_rand).^((1-alpha(i))/alpha(i)));     %generate
'difficult' density
X_abs(X_abs > 30) = 0;             %replace values of X > 30 with zero (removes
outliers)



    k=1;    %calculate tail probabilities
    for t = 0:0.1:10*sqrt(2);
        m=1;
        n = 0;
        while(m < Num+1)    %limit size of tail probability
            if (X_abs(m)>t)     %check if the mth element is greater than the tail
probability
                n = n+1;        %if it is then increase the number of elements
greater than t
            end
            m = m+1;        %check next element
        end
        tailP(k) = n/Num;   %(num_random_values > t) normalised = tail probability
        gausTailP(k) = (1-normcdf(t))*2;    %gaussian tail prob for comparison
        k=k+1;              %next tail prob for next value of t
    end

subplot(numel(alpha), 1, i),
t = 0:0.1:10*sqrt(2);
hold on
plot(t./sqrt(2), tailP, 'r')
plot(t, gausTailP, 'b')
hold off


graph = sprintf('Alpha = %0.1f, Beta = %0.1f', alpha(i), beta);
```

```
title(graph)
xlabel('t')
ylabel('p(|X|>t))')
axis([0 10 0 1])       %graph size limits so comparable

tail_alpha(:, i) = tailP;   %store tailP in columns of differing alpha for use in
next section.

end



%plot tail probabilities vs gamma estimates
figure(4)

for r = 1:1:numel(alpha)

subplot(numel(alpha), 1, r),
hold on
plot(t./sqrt(2), tail_alpha(:, r), 'r')
plot(t, 0.35.*(t.^(-alpha(r))), 'b')
hold off


graph = sprintf('Alpha = %0.1f, Beta = %0.1f', alpha(r), beta);
title(graph)
xlabel('t')
ylabel('p(|X|>t))')
axis([0 10 0 1])       %graph size limits so comparable

end
```