CUED - Engineering Tripos Part IIA 2018-2019 **Module Full Technical Report**

Student name:	Harvey Hughes	CRSID:	hh458	Module number:	3f3

Feedback to the student ☐ **See also comments in the text**		Very good	**Good**	Needs improvmt
CONTENT	**Completeness, quantity of content:** Has the report covered all aspects of the lab? Has the analysis been carried out thoroughly?			
	Correctness, quality of content Is the data correct? Is the analysis of the data correct? Are the conclusions correct?			
	Depth of understanding, quality of discussion Does the report show a good technical understanding? Have all the relevant conclusions been drawn?			
	Comments:			
PRESENTATION	**Structure, organisation of content** Has the report content been structured clearly and in a logical order?			
	Attention to detail, typesetting and typographical errors Is the report free of typographical errors? Are the figures/tables/references presented professionally?			
	Comments:			

Raw report mark	**/ 10**
Penalty for lateness	

The weighting of comments is not intended to be equal, and the relative importance of criteria may vary between modules. A good report should attract about 7 marks.

2 marks / week or part week.
Please use allowance forms to inform the teaching office about mitigating circumstances.

Marker: Date:

Random variables and random number generation FTR

Harvey Hughes
Emmanuel College
Lab Date : 17/10/18
hh458@cam.ac.uk

December 3, 2018

Abstract

The purpose of this lab was to introduce random variables and how transformations can be used to generate a different probability density. This would be achieved using Matlab to back up theoretical results of random variables. Jacobian, inverse CDF and non linear transformations were investigated. The theory agreed with Matlabs random number generation throughout. The α-stable distribution proved to be particularly interesting as it approaches a Gaussian as α goes to two and its pdf can be expressed in the form cx^{γ} for large x. The effect of sample size was observed to improve unbiased estimates such as Monte Carlo by $1/\sqrt{N}$ and produce smoother histogram or kernel density plots.

I. Introduction

During the course of the lab various methods and techniques were used to generate distributions of random variables using Matlab. The techniques investigated involved Jacobian transformation, inverse CDF method and α-stable distributions. Histogram plots and kernel smoothing functions will be used in order to analyse the pdf of these functions and how sample size effects these estimates.
With the objectives being:

- Introduce the idea of random variables and functions of them
- To use the Jacobian to transform random variables
- To experiment with non uniform random number generation

II. Results and Discussion

i. 1. Uniform and normal random variables

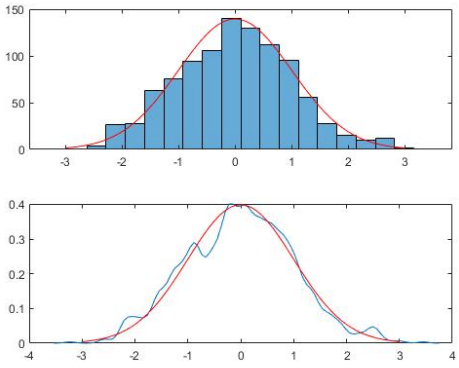

Figure 1: *Histogram and kernel density plot using a Gaussian kernel $\mathcal{N}(0,1)$ with sample size 1000*

When a Gaussian kernel is used, as in figure

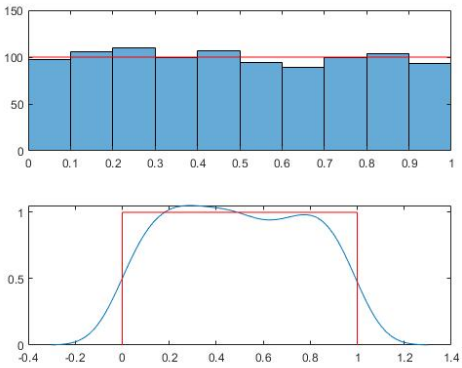

Figure 2: *Histogram and kernel density plot using a uniform distribution $\mathcal{U}(0,1)$ with sample size 1000*

1, it is clear that a kernel density method to approximate the distribution leads to a more accurate curve. This is due to the smoothing inherent in the kernel density method which removes the jaggedness that is shown in the histogram plot.

The opposite is true for distributions with a large discontinuity in density function such as a uniform distribution pictured in figure 2. This figure shows the smoothing present with the kernel method either side of the bounds [0,1]. This leads to an incorrect shape of density function. The histogram plot more accurately depicts the distribution due to bin height tending towards the mean as N increases and therefore all bin heights being close to the pdf.

i.1 Multinominal distribution

When a random vector is drawn from a distribution p(x) and then a histogram plot is generated the multinomial distribution can be calculated. There is a probability ,p_j , that each variable x^i from the random vector lies within the bin limits [a,b].

$$p_j = \int_a^b p(x)dx \tag{1}$$

All the random variables within the random vector are generated independently and therefore the number of x^i's residing in each bin follows a binomial distribution $\mathcal{B}(p_j, N)$. With the two events being x^i outside or inside the bin limits. This results in the mean and variable of bin height following equations 3. The probability of getting n_i variables located in bin p_i follows equation 2.

$$\begin{aligned}
p &= {}^N C_{n_i} p_i^{n_i} (1-p_i)^{N-n_i} \\
p &= \frac{N!}{n_i!(N-n_i)!} p_i^{n_i} (1-p_i)^{N-n_i} \\
&(1-p_i)^{N-n_i} \textit{ is the probability of not in bin } i \\
(1-p_i)^{N-n_i} &= \Pi_j p_j^{n_j} \frac{N!}{n_j!(N-n_j)!} \; i \neq j \textit{ as independant} \\
\Pi_j \frac{N!}{n_j!(N-n_j)!} &= \frac{N!}{n_1!n_2!...n_J!} \\
p &= \frac{N!}{n_1!n_2!...n_J!} \Pi_{j=1}^{J} p_j^{n_i}
\end{aligned} \tag{2}$$

i.2 Variation with sample size in uniform variables

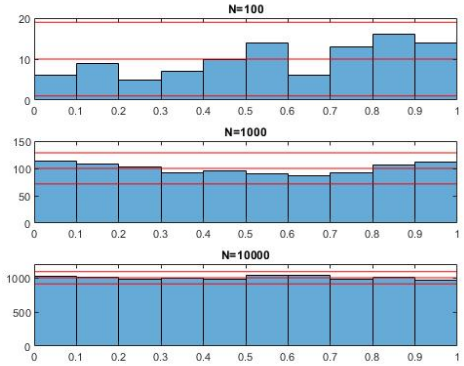

Figure 3: *Histogram plots using a uniform distribution $\mathcal{U}(0,1)$, showing how the theoretical mean and $\pm 3\sigma$ of bin height varies with sample size*

$$\begin{aligned}
\mu &= Np_j \\
\sigma^2 &= Np_j(1-p_j)
\end{aligned} \tag{3}$$

The theoretical mean and standard deviation of bin height in histogram plots can be calculated using equations 3. These values

have been calculated and plotted in figure 3 for three different sample sizes from a uniform distribution. P_j is the probability of the distribution being located in one bin, for the plot mentioned $p_j = 0.1$ as 10 bins are present. Table 1 shows the values calculated. The results observed in figure 3 show that Matlab accurately generates uniform random variable that agree with the multinomial distribution theory which equation 3 is derived from. As sample size increases the 3σ lines are relatively closer to the mean height due to standard deviation increasing with $\sqrt{N}$ and mean increasing with N. This agrees with the bin heights becoming far less varied and residing within the 3σ bounds. These bounds should contain about 99% of all possible bin heights.

Table 1: *Theoretical mean and standard deviation of bin height for a histogram plot of a uniformly distributed random variable*

Sample Size N	Mean	Standard Deviation
100	10	3
1000	100	$3\sqrt{10}$
10000	1000	30

i.3 Variation with sample size in normal variables

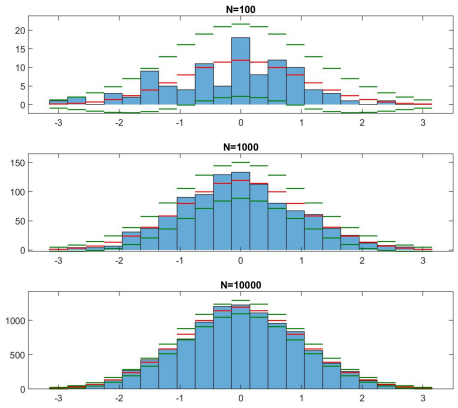

Figure 4: *Histogram plots using a Gaussian distribution $\mathcal{N}(0,1)$, showing how the theoretical mean and $\pm 3\sigma$ of bin height varies with sample size*

The multinomial distribution above can also be applied to non uniform random variables. Figure 4 shows the calculated bin mean heights and variances for a $\mathcal{N}(0,1)$ distribution at various sample sizes. The bin probabilities to be used in equation 3 have been calculated using $p_j = \Phi(upper\,bin\,edge) - \Phi(lower\,bin\,edge)$. Once again it's seen that the relatively proximity of the standard deviation lines to the mean line decreases as N increases. As p_j approaches 0 in the tails of the distribution the -3σ lines show a negative density being possible. This is particularly visible for small sample sizes, as variance at small bin probabilities is nearly equal to the mean height (equation 4) meaning $\mu - 3\sqrt{\mu}$ is likely to be negative. The minima of the -3σ line is observed at ± 1.8 when N=100, this is at an intermediate bin due to the rate at which the Gaussian distribution decreases here (which is linked to the mean bin height) compared to how the variance decreases. When bin probability is one then we only have one histogram bin with height N and no variance which is to be expected.

$$\begin{aligned} &\mu = Np_j \\ &\sigma^2 = Np_j(1-p_j) = Np_j - Np_j^2 \\ &When\ p_j\ small\ \sigma^2 \approx Np_j = \mu \\ &This\ is\ more\ true\ when\ N\ is\ also\ small \end{aligned} \tag{4}$$

i.4 Matlab code

For the Matlab code see section 1.0 and 1.1 of the appendix. With figures 1 to 3 generated in section 1.0 and figure 4 in 1.1.

ii. 2. Function of random variables

ii.1 Transformation Y=aX+b

Starting with the distribution $X \sim \mathcal{N}(0,1)$ and performing the transformation Y=aX+b leads to the distribution plotted in figure 5. This is the distribution $Y \sim \mathcal{N}(a\mu + b, \sigma^2 a^2) = \mathcal{N}(b, a^2)$. The derivation for this can be seen in equations 5. Figure 5 shows the histogram data for a sample of 1000 variables in addition to each variable being transformed individually

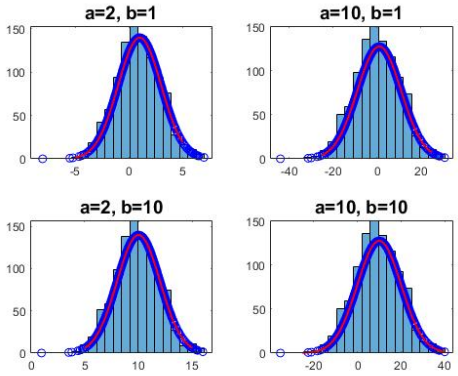

Figure 5: *Histogram plot of transforming a Gaussian kernel $\mathcal{N}(0,1)$ by $f(x) = ax + b$ with sample size 1000*

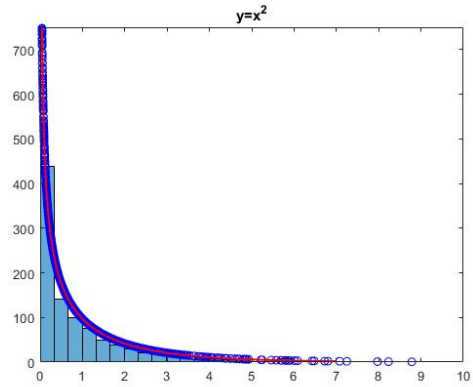

Figure 6: *Histogram plot of transforming a Gaussian kernel $\mathcal{N}(0,1)$ by $f(x) = x^2$ with sample size 1000*

using the Jacobian to form the dotted distribution shown.

$$
\begin{aligned}
&f(X) = aX + b \\
&J = |f'(x)| = |a| \\
&p(y) = \sum_{1}^{1} \frac{p(x)}{J} = \frac{p(\frac{y-b}{a})}{|a|} \\
&p(y) = \frac{1}{a\sigma\sqrt{2\pi}} exp(-\frac{1}{2}[\frac{y-b-a\mu}{a\sigma}]^2) \\
&Y \sim \mathcal{N}(a\mu + b, a^2\sigma^2)
\end{aligned}
\tag{5}
$$

for a given x.

$$
\begin{aligned}
&f(X) = X^2 \\
&J = |f'(x)| = 2|X| = 2\sqrt{y} \\
&p(y) = \sum_{1}^{2} \frac{p(x)}{J} = \frac{p(+\sqrt{y}) + p(-\sqrt{y})}{2\sqrt{y}} \\
&p(y) = \frac{p(\sqrt{y}))}{\sqrt{y}} \; by \; symmetry \\
&p(y) = \frac{1}{\sigma\sqrt{y2\pi}} exp(-\frac{1}{2}[\frac{\sqrt{y}-\mu}{\sigma}]^2) \\
&\sigma = 1, \; \mu = 0
\end{aligned}
\tag{6}
$$

ii.2 Transformation $Y = X^2$

The transformation $Y = X^2$ was applied to another Gaussian distribution following $X \sim \mathcal{N}(0,1)$. Figure 6 shows the result of this. The distribution $p(y) = \frac{1}{\sqrt{y2\pi}} exp(-\frac{1}{2}y)$ is overlain. This distribution was calculated using the same Jacobian method and is shown in equations 6. As $Y = X^2$ is not a 1:1 function the derivation required summing over all possible y values

ii.3 Sine transformations

The transformation $Y = sin(X)$ was applied to a uniform distribution $\mathcal{U}(0, 2\pi)$. Figure 8 shows the end result. A distribution of $p(y) = \frac{1}{2\pi|cos(sin^{-1}(y))|}$ was generated. This is shown overlaid on the histogram plot and each random sample transformed individually. The derivation for this using the Jacobian can be

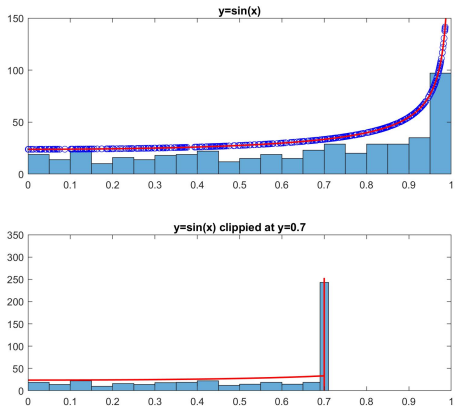

Figure 7: *Histogram plot of transforming a uniform kernel $\mathcal{N}(0,2\pi)$ by $f_1(x) = sin(x)$ and $f_2(x) = min[sin(x),x]$ with sample size 1000*

seen in equations 7.

$$
\begin{aligned}
&f(X) = Sin(X)\\
&y = Sin^{-1}(x)\ this\ is\ simplified\ from\ the\\
&2:1\ case\ due\ to\ p(x)\ being\ uniformly\\
&distributed\ therefore\ each\ inversion\ case\\
&will\ contribute\ half\ towards\ p(y)\\
&J\ is\ also\ periodic\ so\ each\ half\\
&is\ the\ same\\
&J = |f'(x)| = |cos(x)| = |cos(sin^{-1}(y))|\\
&p(y) = \sum_{1}^{1} \frac{p(x)}{J} = \frac{p(sin^{-1}(y))}{|cos(sin^{-1}(y))|}\\
&p(y) = \frac{1/2\pi}{|Cos(sin^{-1}(y))|}\\
&p(y) = \frac{1}{2\pi|Cos(sin^{-1}(y))|}
\end{aligned}
\tag{7}
$$

A clipped version of this transformation was considered next $f(x) = min[sin(x),x]$. This can be also be seen in figure 8. p(y) follows the same distribution as before $p(y) = \frac{1}{2\pi|cos(sin^{-1}(y))|}$ for 0<y<0.7, with a spike of magnitude $\frac{\pi - 2sin^{-1}(0.7)}{2\pi}$ at y=0.7. The derivation for this extra part are shown in equations 8, for 0<y<0.7 the derivation in equations 7 still holds.

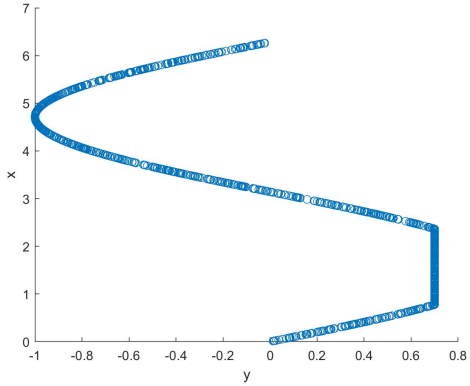

Figure 8: *Plot of the transformation $f(x) = min[sin(x),x]$, with*

The magnitude of this spike should equal the size of interval in x being truncated, multiplied by the constant p(x) in order for $\int p(y)dy = 1$. This agrees with the derivation shown.

$$
\begin{aligned}
&f(x) = min[sin(x),x]\\
&sin^{-1}(0.7) < x < \pi - sin^{-1}(0.7)\\
&y = 0.7\\
&\frac{1}{J} = |\frac{dx}{dy}| = 2[\frac{\pi}{2} - sin^{-1}(0.7)]\delta(y-0.7)\\
&p(y) = \int_{y=0.7}^{y=0.7} p(x)\frac{1}{J}dy\\
&p(y) = \frac{(\pi - 2sin^{-1}(0.7))}{2\pi}\delta(y-0.7)
\end{aligned}
\tag{8}
$$

ii.4 Matlab code

The Matlab code for the transformation Y=aX+b can be seen in section 2.0 of the appendix, the transformation $Y = X^2$ is shown in section 2.1. Section 2.2 includes the code for generating the sine two transformation.

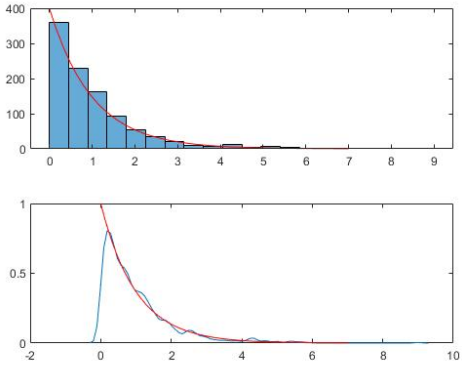

Figure 9: *Histogram and kernel density plot of an exponential distribution generated using the inverse CDF method on a uniform kernel $\mathcal{U}(0,1)$ with sample size 1000*

iii. 3. Inverse CDF method

iii.1 Generating an exponential distribution

$$\begin{aligned} CDF &= \int_0^y p(y)dy = \int_0^y e^{-y} \\ CDF &= 1 - e^{-y} = F(y) \\ F^{-1}(y) &= -ln(1-y) \end{aligned} \tag{9}$$

The cumulative distribution function (CDF) for an exponential distribution p(y) = e^{-y} is calculated in equations 9. Though the use of the inverse CDF and equation 10 an exponentially distributed random variable can be generated using a uniform kernel $\mathcal{U}(0,1)$. This generated distribution is shown in figure 9 with p(y) overlain. The curves match up with great accuracy especially the kernel density plot. This shows the validity of this method in generating different random variable densities.

$$y^i = F^{-1}(x^i) \tag{10}$$

iii.2 Monte Carlo estimation

The Monte Carlo estimate for mean and variance (equation 11) was used to estimate the these parameters for the generated exponential distribution. This estimate regularly produces

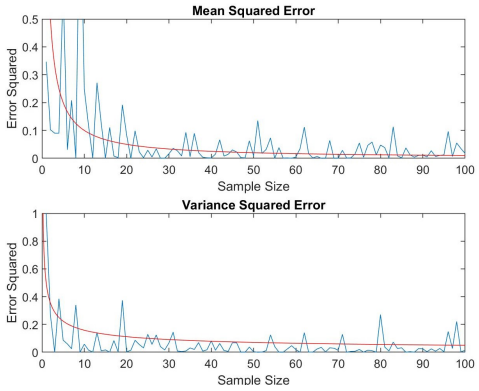

Figure 10: *Error squared of the Monte Carlo estimator on the exponential distribution, for sample sizes up to 100. Mean squared error is stretched by 5 to more closely match the line.*

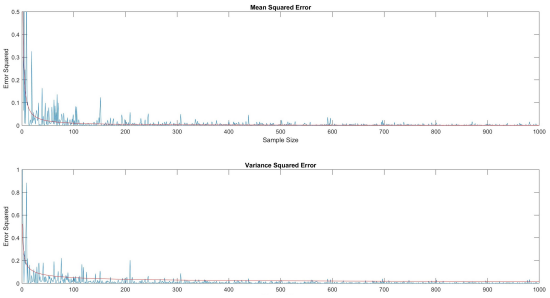

Figure 11: *Error squared of the Monte Carlo estimator on the exponential distribution, for sample sizes up to 1000*

means and variances within 0.05 of the theoretical value when sample size is 1000. This is an unbiased estimator as when sample size increases the estimator tends closer to the true value μ therefore $\mathbb{E}(\hat{\mu}) = \mu$. With sample size 100 million the estimator is within 0.001 of the true value.

$$\mu = \mathbb{E}[Y] = \int_{y=0}^{\infty} yp(y)dy \approx \frac{1}{N}\sum_{i=1}^{N} y^i = \hat{\mu} \tag{11}$$

The convergence of this estimator towards the true mean follows a $1/\sqrt{N}$ rate. This can be seen in figures 10 and 11 where the squared error is plotted as sample size increases. This error squared follows a 1/N line. The estimators in the graph have been generated from

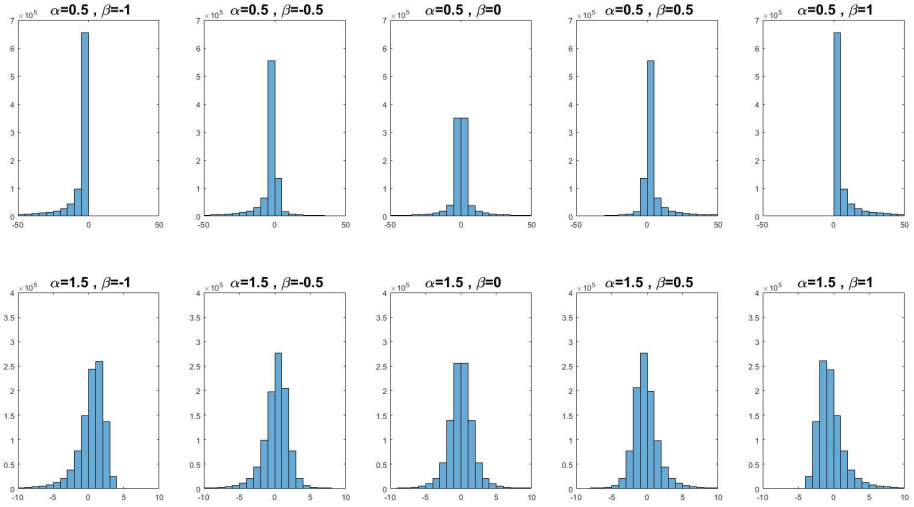

Figure 12: *Histogram plots of distribution X in equation 12 using various values for constants a and b, all from the same kernels* $\mathcal{U}(-\frac{\pi}{2}, \frac{\pi}{2})$ *and* $\mathcal{E}(1)$ *with sample size 1000.*

different random sequences for each sample size and repeated three times for each size. The sequences were generated using the exponential distribution previously discussed. From figure 11 the error can be seen to be very small at large sample sizes, with the occasional spike due to the variance of this estimator.

iii.3 Matlab code

The Matlab code for the inverse CDF method can be seen in section 3.0 of the appendix. The code for the Monte Carlo estimator is shown in section 3.1.

iv. 4. Simulation from a 'difficult' density

By using the non linear transformation listed in equations 12 and various values for α and β the graphs in figure 12 can be produced. These graphs indicate that α controls the tail thinness and sharpness of central peak, with a low alpha resulting in a thin tail and sharp centre. A low α such as 0.5 results in a very sharp density peak at 0. β appears to affect the skew of the distribution. A negative β results in negative skew and vice versa.

$$\begin{aligned} &\alpha \in (0,2)\, \beta \in [-1,1] \\ &b = \frac{1}{\alpha} tan^{-1}(\beta tan(\frac{\pi\alpha}{2})) \\ &s = (1+\beta^2 tan^2(\frac{\pi\alpha}{2}))^{\frac{1}{2\alpha}} \\ &U \sim \mathcal{U}(-\frac{\pi}{2}, \frac{\pi}{2})\, V \sim \mathcal{E}(1) \\ &X = s\frac{sin(\alpha(U+b))}{(cosU)^{1/\alpha}} \end{aligned} \tag{12}$$

iv.1 Tail behaviour

The tail probabilities of this distribution can be calculated by counting how many random variables lie outside a certain value. This was conducted with the distribution with $\beta = 0$ and $\alpha = 0.5, 1.5$, the results are shown in table 2. The Gaussian distribution's tail is far smaller and reaches less then 1% at t=3 however the generated distributions both still have significant probabilities at this t value. The tail is thinner when alpha is 1.5 compared to 0.5 as previously noticed.

The tail behaviour of the pdf p(x) can be approximated to a curve cx^{γ} at large x values.

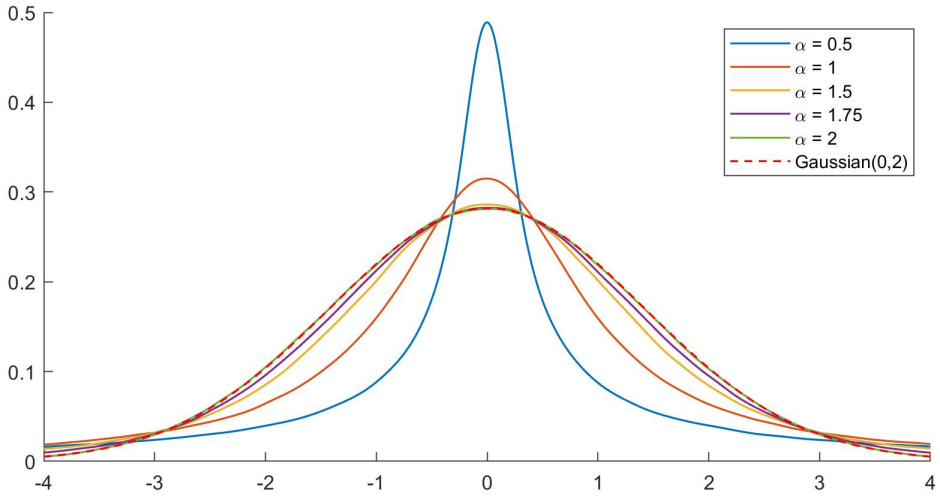

Figure 13: *Pdf of the distribution for various α values.*

Table 2: *Tail probabilities for the distribution with $\beta = 0$ and $\alpha = 0.5, 1.5$ for four locations(t)*

Alpha	t=0	1	3	6
0.5	1	0.5467	0.3749	0.2850
1.5	1	0.4896	0.1053	0.0304
Gaussian	1	0.3173	0.0027	0.0000

In order to work out this relationship a log plot can be constructed of the pdf. The pdf is calculated using a kernel smoothing technique of the random vector X. A large sample size is required to accurately get this relationship as p(x) at x=100 is approximately 10^{-4} and $5x10^{-6}$ for alpha = 0.5 and 1.5. This means in order to get sufficient random variables in this region sample size has to be large. A sample size of 100 million was used. The pdf and log plots can be seen in figure 14. From these plots γ is the gradient at large x on the log plot, it was found to be -0.008 and -0.015 for alpha = 0.5 and 1.5 respectively. LogC varied across repeats but was in the range -2.8 and -3.8 for the two distributions. In order to see how γ is related to α distributions with other *alpha* values were tested, these are plotted in 15. The linear relationship $\gamma = -0.0078\alpha - 0.0035$ was calculated.

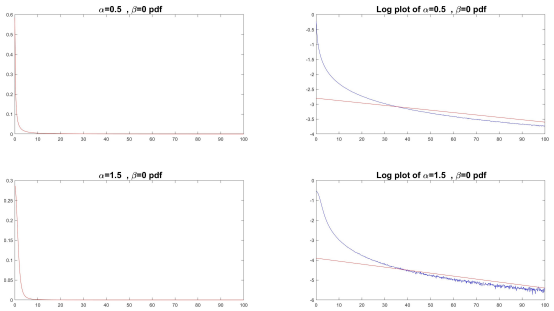

Figure 14: *Pdf and log(pdf)of the distribution with $\beta = 0$ and $\alpha = 0.5, 1.5$*

iv.2 Links with the Gaussian

By plotting the pdf of the distribution for various α values as shown in figure 13 links between the Gaussian can be found. When α approaches 2 it closely matches and eventually is a Gaussian distribution $\mathcal{N}(0, 2)$.

iv.3 Matlab code

The Matlab code for this transformation can be seen in sections 4.0 - 4.3 of the appendix.

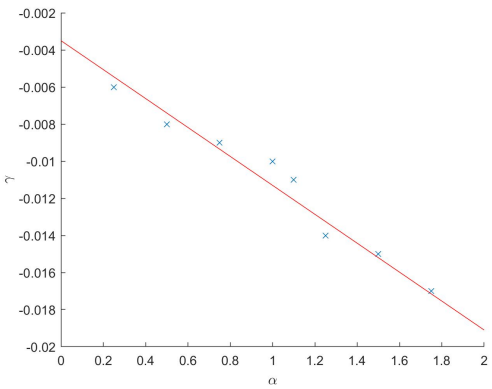

Figure 15: *Relationship between α and γ at large x values.* $\gamma = -0.0078\alpha - 0.0035$

III. Conclusion

Through the investigations carried out through this lab it has become clear that Matlab random number generation follows the theory such as the multinomial distribution. Through the use of transformations either directly or using inverse CDF complicated distributions can be generated from much simpler ones such as uniform and Gaussian. Sample size decreased spread throughout the experiments producing smoother curves, histograms and better estimators. The alpha stable distribution is an example of a difficult density that can be generated using these methods. This distribution tends to a Gaussian as α tends to 2 and skewness is created by the β variable. This distribution can be expressed in the form cx^{γ} for large x. The sample mean of distributions can be estimated using a Monte Carlo approach which produced an unbiased estimator with accuracy increasing with $1/\sqrt{N}$. Transforming distributions proves very useful in real word applications as it means lots of desired distributions can be generated from the same technology that for reliably produces a uniform distribution.

IV. Appendix

1.0 Uniform and normal random variables

```
%Plot Normal Distributions
figure(1)
x=randn(1000,1);
subplot(211),
histogram(x,20)
hold on
n = [-3:.1:3];
norm = normpdf(n,0,1)*350;
plot(n,norm,'red')
hold off
subplot(212),
ksdensity(x,'width',0.1)
hold on
n = [-3:.1:3];
norm = normpdf(n,0,1);
plot(n,norm,'red')
hold off


   %plot Uniform Distribution
figure(2)
y=rand(1000,1);
subplot(211),
histogram(y,10)
axis([0 1 0 150])
hold on
height = 100.* ones(length(n),1)
plot(n,height,'red')
hold off
subplot(212),
ksdensity(y,'width',0.1)
hold on
x1=[0,0,1,1];
y1=[0,1,1,0];
plot(x1,y1,'red')
hold off


   %plot Uniform no.2 Distribution
N=[100,1000,10000]
height=[20,150,1200]
figure(3)
for i =1:3
y=rand(N(i),1);
subplot(3,1,i),
histogram(y,10)
title(['N=',num2str(N(i))])
hold on
mean=N(i)*0.1;
sd= sqrt(N(i)*0.1*0.9);
line([0,1],[mean,mean],'color','red')
```

```
line([0,1],[mean+3*sd,mean+3*sd],'color','red')
line([0,1],[mean-3*sd,mean-3*sd],'color','red')
axis([0 1 0 height(i)])
hold off
end
```

1.1 Gaussian bin probabilities

```
%plot Normal distribution Distribution
N=[100,1000,10000]
height=[20,150,1200]
figure(1)

for i =1:3
botrange=0;
y=randn(N(i),1);
subplot(3,1,i),
numberofbins = 21;
binranges = zeros(numberofbins,1);
for j = 1:numberofbins+1
binranges(j) = (3–3)/(numberofbins-1) *(j-1) +
-3.15;
end
histogram(y,binranges)
hold on
for j = 1:length(binranges)-1
pj = normcdf(binranges(j+1)) - norm-
cdf(binranges(j));
mean = N(i)*pj;
sd = sqrt(N(i)*pj*(1-pj))*3;
if j==length(binranges)/2
toprange = (mean + sd )*1.1;
end
tempbot= (mean-sd)*1.1;
if tempbot <botrange
botrange=tempbot;
end
line([binranges(j),binranges(j+1)],
[mean,mean],'color','red', 'LineWidth',1)
line([binranges(j),binranges(j+1)],[mean+sd,mean+sd],
'color',[0 0.5 0],'LineWidth',1)
line([binranges(j),binranges(j+1)],[mean-sd,mean-
sd], 'color',[0 0.5 0],'LineWidth',1)
end
hold off
axis([-3.5 3.5 botrange toprange])
title(['N=',num2str(N(i))])
end
```

2.0 Transforming a Gaussian by Y=aX+b

```
x=randn(1000,1);
a=2;
b=1;
y=a*x+b;
J=abs(a);

py=0;
for i = 1:1 %only one possible value for x given y
py = normpdf((y-b)/a,0,1)/J ;
end

%Plot Transformed Distributions
figure(1)
subplot(221),
histogram(y,20)
hold on
scatter(y,py*700,'blue')
f = @(x) 1/(sqrt(2*pi)*a) *exp(-0.5*(x-b)^ 2/(a^ 2))
*700
fplot(f,[-5,5],'red','LineWidth',1.5)
hold off
title('\fontsize{14} a=2, b=1')

subplot(222),
a=10;
b=1;
y=a*x+b;
J=abs(a);
py=0;
for i = 1:1 %only one possible value for x given y
py = normpdf((y-b)/a,0,1)/J ;
end
histogram(y,20)
hold on
scatter(y,py*3200,'blue')
f = @(x) 1/(sqrt(2*pi)*a) *exp(-0.5*(x-b)^2/(a^2))
*3200
fplot(f,[-25,25],'red','LineWidth',1.5)
hold off
title('\fontsize{14} a=10, b=1')

subplot(223),
a=2;
b=10;
y=a*x+b;
```

```
J=abs(a);
py=0;
for i = 1:1 %only one possible value for x given y
py = normpdf((y-b)/a,0,1)/J ;
end
histogram(y,20)
hold on
scatter(y,py*700,'blue')
f = @(x) 1/(sqrt(2*pi)*a) *exp(-0.5*(x-b)^2/(a^2))
*700
fplot(f,[5,15],'red','LineWidth',1.5)
hold off
title('\fontsize{14} a=2, b=10')


   subplot(224),
a=10;
b=10;
y=a*x+b;
J=abs(a);
py=0;
for i = 1:1 %only one possible value for x given y
py = normpdf((y-b)/a,0,1)/J ;
end
histogram(y,20)
hold on
scatter(y,py*3200,'blue')
f = @(x) 1/(sqrt(2*pi)*a) *exp(-0.5*(x-b)^2/(a^2))
*3200
fplot(f,[-25,40],'red','LineWidth',1.5)
hold off
title('\fontsize14 a=10, b=10')
```

2.1 Transforming a Gaussian by $Y = X^2$

```
x=randn(1000,1);
y=x.^2;
py=zeros(length(y),1);
for i = 1:2 %y=x^2 is a 2:1 function
J=abs(x)*2;
py = py + normpdf((-1)^i*sqrt(y),0,1)./J ;
end


   %Plot Transformed distribution
figure(1)
histogram(y,40)
hold on
scatter(y,py*400,'blue')
f = @(x) 1/(sqrt(2*pi*x)) *exp(-0.5*(sqrt(x))^2) *400
fplot(f,[0,7],'red','LineWidth',1.5)
hold off
title('y=x^2')
axis([0 10 0 750])
```

2.2 Transforming a Gaussian by $y = sin(x)$ and $y = min[sin(x), x]$

```
x=rand(1000,1)*2*pi; %uniform distribution
y=sin(x);
figure(2)
yclip=min(y,0.7);
scatter(yclip,x)
xlabel('y')
ylabel('x')
py=zeros(length(y),1);
for i = 1:1 %y=sin(x) is a 1:1 function in range
0-2pi
J=abs(cos(x));
py = py + (1/(2*pi))./J ; % as p(x) = 1/2pi
everywhere
end


   binrange=[0,0.05,0.1,0.15,0.2,0.25,0.3,0.35,0.4,0.45,
0.5,0.55,0.6,0.65,0.7,0.75,0.8,0.85,0.9,0.95,1]
%Plot Transformed distribution
figure(1)
subplot(211)
histogram(y,binrange)
hold on
scatter(y,py*150,'blue')
f = @(x) 1/(2*pi*abs(cos(asin(x)))) *150;
fplot(f,[0,1],'red','LineWidth',1.5)
hold off
title('y=sin(x)')
axis([0 1 0 150])
binrange=[0,0.05,0.1,0.15,0.2,0.25,0.3,0.35,0.4,0.45,
0.5,0.55,0.6,0.65,0.69,0.71] % in order to see peak
at 0.7
subplot(212)
histogram(yclip,binrange)
hold on
f = @(x) 1/(2*pi*abs(cos(asin(x)))) *150;
fplot(f,[0,0.7],'red','LineWidth',1.5)
top=1/(2*pi)*(pi-2*asin(0.7))*1000 % times 1000
as its a p(y)*N = number in histogram here
line([0.7 0.7],[0 top],'color', 'red', 'LineWidth',1.5)
hold off
title('y=sin(x) clippied at y=0.7')
```

```
axis([0 1 0 350])
```

3.0 Generating an exponential distribution using CDF method

```
x=rand(1000,1);
y= -log(-x+1);

   figure(1)
subplot(211),
histogram(y,20);
hold on
f = @(x) 400*exp(-x)
fplot(f,[0,7],'red')
hold off
subplot(212),
ksdensity(y,'width',0.1)
hold on
f = @(x) exp(-x)
fplot(f,[0,7],'red')
hold off
```

3.0 Generating an exponential distribution using CDF method

```
M=100;
x=rand(M,1);
y=-log(-x+1);

N=3;
mean = zeros(M,1);
variance = zeros(M,1);
meansquarederror = zeros(M,N);
mse=zeros(M,1);
vse=zeros(M,1);
variancesquarederror = zeros(M,N);
range=zeros(M,1);

   for j =1:N
xs=rand(M,M);
f = @(x) -log(-x+1);
ys= arrayfun(f,xs);
for i = 1:M
mean(i)=sum(ys(i,:))/M;
variance(i)=sum(ys(i,:).*ys(i,:))/M -
mean(i)*mean(i);
temp= sum(ys(i,1:i))/i;
meansquarederror(i,j)= temp;
variancesquarederror(i,j)= sum(ys(i,1:i).*ys(i,1:i))/i
- temp*temp;
range(i)=i;
end samplegroupmean=sum(mean)/M;
samplegroupvariance=sum(variance.*variance)/M;
end
for i = 1:M
mse(i)=sum(meansquarederror(i,:))/N;
vse(i)=sum(variancesquarederror(i,:))/N;
end
figure(2)
subplot(211)
plot(range,(mse-1).*(mse-1)*5);
f = @(x) 1/(x);
hold on
fplot(f,[0,M],'red')
hold off
axis([0 100 0 0.5])
title('Mean Squared Error')
xlabel('Sample Size')
ylabel('Error Squared')
subplot(212)
plot(range,(vse-1).*(vse-1));
f = @(x) 1/(2*sqrt(x));
hold on
fplot(f,[0,M],'red')
hold off
axis([0 100 0 1])
title('Variance Squared Error')
xlabel('Sample Size')
ylabel('Error Squared')
```

4.0 Simulation from a difficult density

```
v=exprnd(1,1000,1);
u=rand(1000,1);
u=(u-0.5)*pi ;
alpha=0.5;
betavals=[-1,-0.5,0,0.5,1,-1,-0.5,0,0.5,1];
x=zeros(length(u),10);

   figure(1)
histrange=50;
range=700;

   for i = 1:10
beta = betavals(i);
if i>5
alpha = 1.5;
histrange=10;
```

```
range=400;
end
b=1/alpha * atan(beta*tan(pi*alpha/2)) ;
s=(1+beta^2*(tan(pi*alpha/2))^2)^(1/(2*alpha));
x(:,i)=s*      sin(alpha*(u+b))./(cos(u).^(1/alpha)).*
(cos(u-alpha*(u+b))./v).^((1-alpha)/alpha);
subplot(2,5,i),
histogram(x(:,i),[-histrange:histrange/10:histrange])
axis([-histrange histrange 0 range])
tit = strcat('\fontsize{18} \alpha=',num2str(alpha),
'\fontsize{18} , \beta=',num2str(beta));
title(tit)
end
```

4.1 Tail probabilities

```
v=exprnd(1,10000,1);
u=rand(10000,1);
u=(u-0.5)*pi ;
alphavals=[0.5,1.5];
beta=0;
tvalues=[0,1,3,6];
x=zeros(length(u),2);
tailprob=zeros(4,4);

   for i = 1:2
alpha = alphavals(i);
b=1/alpha * atan(beta*tan(pi*alpha/2)) ;
s=(1+beta^2*(tan(pi*alpha/2))^2)^(1/(2*alpha));
x(:,i)=s*      sin(alpha*(u+b))./(cos(u).^(1/alpha)).*
(cos(u-alpha*(u+b))./v).^((1-alpha)/alpha);
for j = 1:length(u) %calculate tail probabilities
X=abs(x(j,i));
if X > tvalues(4)
tailprob(:,i)=tailprob(:,i)+1;
elseif X>tvalues(3)
tailprob(1:3,i)=tailprob(1:3,i)+1;
elseif X>tvalues(2)
tailprob(1:2,i)=tailprob(1:2,i)+1;
elseif X>tvalues(1)
tailprob(1,i)=tailprob(1,i)+1;
end
end
end
tailprob=tailprob./length(u);
for i = 1:length(tvalues)
tailprob(i,3:4)=2*normcdf(-tvalues(i));
end
tailprob
```

```
   figure(1)

   subplot(2,1,1),
histogram(x(:,1),[-10:10/20:10])
axis([-10 10 0 2000])
tit = strcat('\fontsize{18} \alpha=0.5 \fontsize{18}
, \beta=0');
title(tit)
hold on
n = [-10:.1:10];
norm = 4000*normpdf(n,0,1);
plot(n,norm,'red')
hold off

   subplot(2,1,2),
histogram(x(:,2),[-10:10/20:10])
axis([-10 10 0 2000])
tit = strcat('\fontsize{18} \alpha=1.5 \fontsize{18}
, \beta=0');
title(tit)
hold on
n = [-10:.1:10];
norm = 3000*normpdf(n,0,1);
plot(n,norm,'red')
hold off
```

4.2 Calculating gamma

```
v=exprnd(1,10000000,1);
u=rand(10000000,1);
u=(u-0.5)*pi ;
alphavals=[0.5,1.5];
beta=0;
x=zeros(length(u),2);
pdfx=zeros(1000,2);
xi=zeros(1000,2);
pts = linspace(0,100,1000); % points to evaluate
the estimator
for i = 1:2
alpha = alphavals(i);
b=1/alpha * atan(beta*tan(pi*alpha/2)) ;
s=(1+beta^2*(tan(pi*alpha/2))^2)^(1/(2*alpha));
x(:,i)=s*      sin(alpha*(u+b))./(cos(u).^(1/alpha)).*
(cos(u-alpha*(u+b))./v).^((1-alpha)/alpha);
(p,xd]=ksdensity(x(:,i),pts);
pdfx(:,i)=p;
xi(:,i)=xd;
end
```

```
figure(1)
lgpdfx=log10(pdfx);
subplot(2,2,1),
plot(xi(:,1),pdfx(:,1),'red')
tit = strcat('\fontsize{18} \alpha=0.5 \fontsize{18}
, \beta=0 pdf');
title(tit)
subplot(2,2,2)
plot(xi(:,1),lgpdfx(:,1),'blue')
tit = strcat('\fontsize{18}Log plot of \alpha=0.5
\fontsize{18} , \beta=0 pdf');
title(tit)
xlim([0 100]);
f = @(x) -0.008*x-2.8;
hold on
fplot(f,[0,100],'red')
hold off


   subplot(2,2,3),
plot(xi(:,2),pdfx(:,2),'red')
tit = strcat('\fontsize{18} \alpha=1.5 \fontsize{18}
, \beta=0 pdf');
title(tit)
subplot(2,2,4)
plot(xi(:,2),lgpdfx(:,2),'blue')
tit = strcat(' \fontsize{18}Log plot of \alpha=1.5
\fontsize{18} , \beta=0 pdf');
title(tit)
xlim([0 100]);
f = @(x) -0.015*x-3.9;
hold on
fplot(f,[0,100],'red')
hold off
```

4.3 Links to the Gaussian

```
v=exprnd(1,1000000,1);
u=rand(1000000,1);
u=(u-0.5)*pi ;
alphavals=[0.5,1,1.5,1.75,2];
beta=0;
x=zeros(length(u),length(alphavals));
pdfx=zeros(1000,length(alphavals));
xi=zeros(1000,length(alphavals));
pts = linspace(-4,4,1000); % points to evaluate the
estimator
for i = 1:length(alphavals)
alpha = alphavals(i);
b=1/alpha * atan(beta*tan(pi*alpha/2)) ;
s=(1+beta^2*(tan(pi*alpha/2))^2)^(1/(2*alpha));
x(:,i)=s*      sin(alpha*(u+b))./(cos(u).^(1/alpha)).*
(cos(u-alpha*(u+b))./v).^((1-alpha)/alpha);
(p,xd]=ksdensity(x(:,i),pts);
pdfx(:,i)=p;
xi(:,i)=xd;
end
figure(1)
n = [-4:.1:4];
for i = 1:length(alphavals)
hold on
plot(xi(:,i),pdfx(:,i),'LineWidth',1)
hold off
end
hold on
norm = normpdf(n,0,sqrt(2));
plot(n,norm,'–red','LineWidth',1)
hold off
legend('\alpha = 0.5','\alpha = 1','\alpha =
1.5','\alpha = 1.75','\alpha = 2','Gaussian(0,2)')
```