Monte Carlo Simulation
(MA226)

Project

A convenient way to generate Gamma Random variables using generalized Exponential Distribution

By
Mohammed Bilal Girach
Kodi Digvijay Yadav
Mamilla Rajasekhar
M.S.K.Prateek
G.Sai Vardhan Reddy

Supervised by
Arabin Kumar Dey
April 12, 2017

Contents

1 Ahrens and Dieter(AD) 1

1.1 Pseudo Code . . . 2
1.2 Code . . . 2
1.3 Sample graph . . . 3
1.4 Output . . . 4

2 Best 4

2.1 Pseudo code . . . 5
2.2 Code . . . 6
2.3 Sample graph . . . 7
2.4 Output . . . 7

3 Proposed Methology 8

3.1 Algorithm 1 . . . 8
3.1.1 Psuedo Code . . . 8
3.1.2 Code . . . 8
3.1.3 Sample graph . . . 9
3.1.4 Output . . . 10
3.2 Algorithm 2 . . . 10
3.2.1 Pseudo Code . . . 11
3.2.2 Code . . . 11
3.2.3 Sample graph . . . 13
3.2.4 Output . . . 13
3.3 Algorithm 3 . . . 14
3.3.1 Pseudo Code . . . 14
3.3.2 Code . . . 15
3.3.3 Sample graph . . . 16
3.3.4 Output . . . 16

4 Analysis on different methods 17

4.1 Code . . . 17
4.2 Tabulated Data . . . 22
4.3 Graph . . . 22

There are algorithms for shape parameter(α) < 1 and shape parameter(α) > 1.In this project we cite the methods for the case $0 < \alpha < 1$ and scale parameter $\lambda = 1$
Two most popular methods are
1.Ahrens and Dieter(AD)
2.Best

1 Ahrens and Dieter(AD)

This method is based on the acceptance-rejection method with proper choice of majorization functions
Following is the majorization function

$$t_{AD}(x;\alpha) = \begin{cases} \dfrac{x^{\alpha-1}}{\Gamma(\alpha)}, & 0 < x < 1 \\ \dfrac{e^{-x}}{\Gamma(\alpha)}, & x > 1 \end{cases}$$

$c_{AD} = \int_0^\infty t_{AD}(x;\alpha)\,\mathrm{d}x = \frac{(e+\alpha)}{e\Gamma(\alpha+1)}$
Now we define a probability density function r_{AD} which is obtained by dividing c_{AD} with $\int_0^\infty t_{AD}(x;\alpha)\,\mathrm{d}x$

$$r_{AD} = \frac{t_{AD}}{\int_0^\infty t_{AD}(x;\alpha)\,\mathrm{d}x}$$

$$r_{AD}(x;\alpha) = \begin{cases} \dfrac{e\alpha x^{\alpha-1}}{e+\alpha}, & 0 < x < 1 \\ \dfrac{e\alpha e^{-x}}{e+\alpha}, & x > 1 \end{cases}$$

The Cumulative Distribution Function of r_{AD} is

$$R_{AD}(x;\alpha) = \begin{cases} \dfrac{ex^{\alpha}}{e+\alpha}, & 0 < x < 1 \\ 1 - \dfrac{\alpha e^{1-x}}{e+\alpha}, & x > 1 \end{cases}$$

Gamma random variable for $\lambda = 1$ and also $0 < \alpha < 1$

$$f_{GA}(x;\alpha) = \frac{e^{-x}(x)^{(\alpha-1)}}{\Gamma(\alpha)}; x > 0$$

Supremum of $\frac{f_{GA}(x;\alpha)}{r_{AD}(x;\alpha)} = c_{AD}$. Proof is as follows

$$\frac{f_{GA}(x;\alpha)}{r_{AD}(x;\alpha)} = \begin{cases} \dfrac{e+\alpha}{e\Gamma(\alpha+1)e^x} \leq \dfrac{e+\alpha}{e\Gamma(\alpha+1)} = c_{AD}, & 0 < x < 1 \\ \dfrac{(e+\alpha)x^{\alpha-1}}{e\Gamma(\alpha+1)} \leq \dfrac{e+\alpha}{e\Gamma(\alpha+1)} = c_{AD}, & x > 1 \end{cases}$$

1.1 Pseudo Code

- Generate a uniform random number U
- If $U \leq \frac{e}{e+\alpha}$ then $X = \left(U\left(\frac{e+\alpha}{e\alpha}\right)^{1/\alpha}\right)$ otherwise $X = -log(\frac{e+\alpha}{e\alpha}(1-U))$
- Generate a uniform random number V;if $X \leq 1$ and if $V \leq e^{-X}$ then accept X otherwise goto 1.If $X > 1$ and if $V \leq X^{\alpha-1}$ then accept$X$ otherwise go back 1

1.2 Code

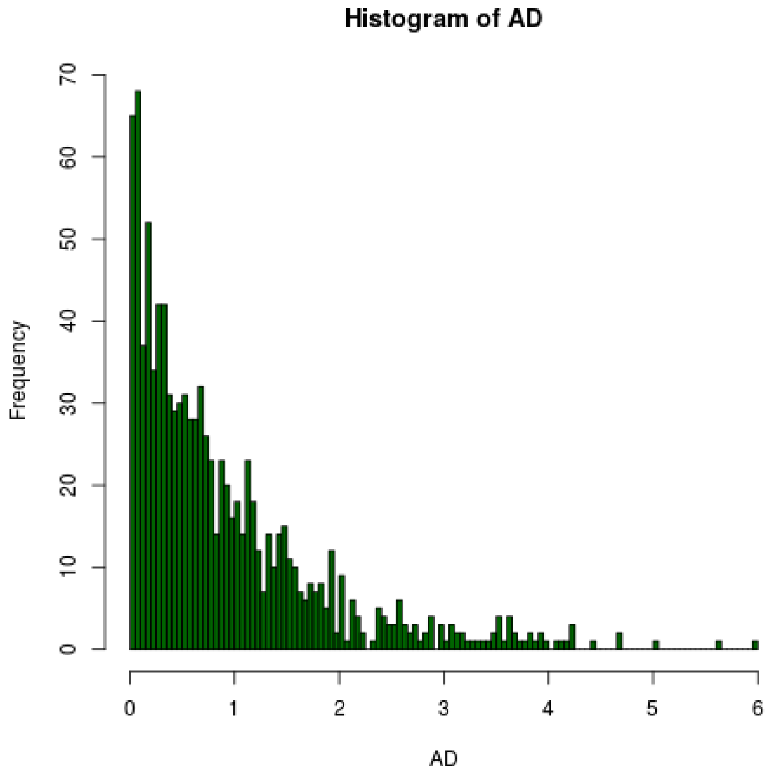
```
AD=function(n,alpha)
{
AD=vector(length=n)
i=1;count=0
e=exp(1)
#print(e/(e+alpha))
k=(e+alpha)/(e*alpha)
#print(k)
while(i<=n)
{
arr=runif(2)
count=count+1
if  (arr[1]<=e/(e+alpha))
{
X=((arr[1]*(e+alpha))/e)^(1/alpha)
}
else
{
X=-log(k*(1-arr[1]))
}
```

```
#cat(arr[1],arr[2],X,"\n")
if (X<=1 && X!=0 && arr[2]<=exp(-X))
{
AD[i]=X
i=i+1
}
if (X>1 && arr[2]<=X^(alpha-1))
{
AD[i]=X
i=i+1
}
}
integrand=function(x){x^(alpha)*exp(-1*x)}
c=integrate(integrand,lower=0,upper=Inf)$value
cat("Theoritical Acceptance via AD method:",
e*c/(e+alpha),"\n")
#print(1/c)
cat("Acceptance Obtained:",n/count,"\n")
png("image1.png")
hist(AD,breaks=100,col="darkgreen")
dev.off()
#cat((n/count))

}
AD(1000,0.9)
```

1.3 Sample graph

For n=1000;$\alpha = 0.9$



1.4 Output

- Theoritical Acceptance via AD method: 0.7225393
- Acceptance Obtained: 0.7267442

2 Best

Best used the following majorization function

$$t_B(x;\alpha) = \begin{cases} \dfrac{ex^{\alpha-1}}{\Gamma(\alpha)}, & 0 < x < d \\ \dfrac{d^{\alpha-1e^{-x}}}{\Gamma(\alpha)}, & x > d \end{cases}$$

$\int_0^\infty t_B(x;\alpha)\,\mathrm{d}x = \frac{d^\alpha}{\Gamma(\alpha)}\left[\frac{1}{\alpha} + e^{-d}\right]$
Choose d such that $\frac{d^\alpha}{\Gamma(\alpha)}\left[\frac{1}{\alpha} + e^{-d}\right]$ is minimum.It gives

$$d = e^{-d}(1 + \alpha - d)$$

The approximation for d given by Best is

$$d = 0.07 + 0.75\sqrt{1-\alpha}$$

Now we define a probability density function r_{AD} which is obtained by dividing r_B with $\int_0^\infty t_B(x;\alpha)\,\mathrm{d}x$

$$r_B(x;\alpha) = \begin{cases} \dfrac{\alpha e x^{\alpha-1}}{bd^\alpha}, & 0 < x < d \\ \dfrac{\alpha e^{-x}}{bd}, & x > d \end{cases}$$

where $b = 1 + \frac{\alpha e^{-d}}{d}$
c_B=supremum of $\frac{f_{GA}(x,\alpha)}{r_B(x,\alpha)}$=$\frac{(d+\alpha e^{-d})d^{\alpha-1}}{\Gamma(\alpha+1)}$

The Cumulative Distribution Function of r_{AD} is

$$R_B(x;\alpha) = \begin{cases} \dfrac{1}{b}\left(\dfrac{x}{d}\right)^\alpha, & 0 < x < d \\ \dfrac{1}{b} + \dfrac{a(e^{-d} - e^{-x})}{bd}, & x > d \end{cases}$$

Gamma random variable for $\lambda = 1$ and also $0 < \alpha < 1$

$$f_{GA}(x;\alpha) = \frac{e^{-x}(x)^{(\alpha-1)}}{\Gamma(\alpha)}; x > 0$$

2.1 Pseudo code

- Generate a uniform random number U
- If $U \leq \frac{1}{b}$ then $X = d(bU)^{1/\alpha}$ otherwise $X = -log(e^{-d} + (\frac{1}{b} - U)\frac{bd}{\alpha})$
- Generate a uniform random number V;if $X \leq 1$ and if $V \leq e^{-X}$ then accept X otherwise goto 1.If $X > 1$ and if $V \leq \left(\frac{X}{d}\right)^{\alpha-1}$ then accept$X$ otherwise go back 1

2.2 Code

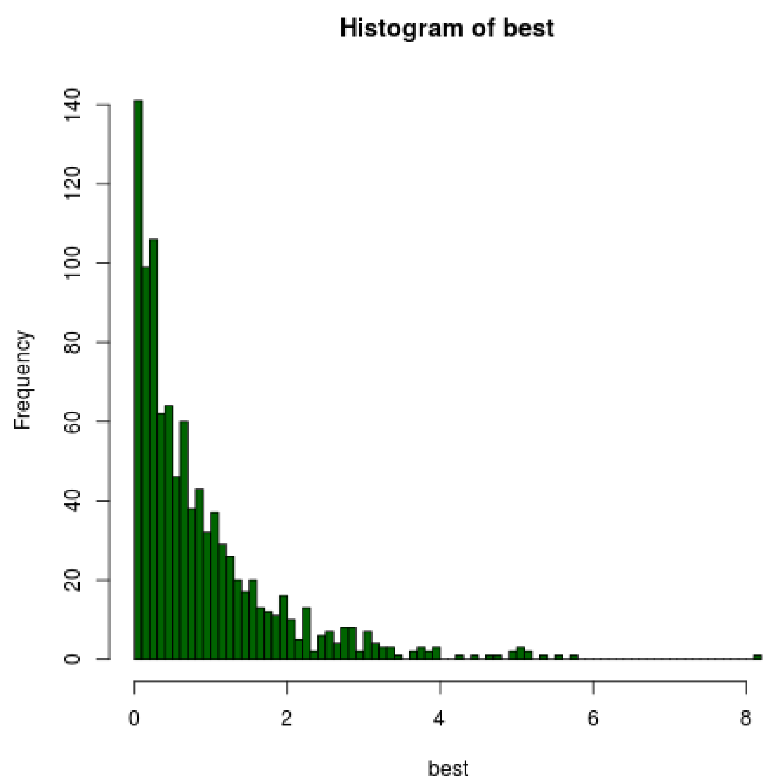
```
Best=function(n,alpha)
{
best=vector(length=n)
i=1;count=0
d=0.07+0.75*((1-alpha)^(0.5))
b=1+exp(-1*d)*alpha/d
while(i<=n)
{
arr=runif(2)
count=count+1
if (arr[1]<=1/b)
{
X=((b*arr[1])^(1/alpha))*d
}
else
{
X=-log(exp(-1*d)+(d-b*d*arr[1])/(alpha))
}
#cat(arr[1],arr[2],X,"\n")
if (X<=d && X!=0 && arr[2]<=exp(-X))
{
best[i]=X
i=i+1
}
if (X>d && arr[2]<=(X/d)^(alpha-1))
{
best[i]=X
i=i+1
}
}
integrand=function(x){x^(alpha)*exp(-1*x)}
c=integrate(integrand,lower=0,upper=Inf)$value
cat("Theoritical Acceptance via Best method:",
c/((d+exp(-1*d)*alpha)*(d^(alpha-1))),"\n")
```

```
#print(1/c)
cat("Acceptance Obtained:",n/count,"\n")
png("image2.png")
hist(best,breaks=100,col="darkgreen")
dev.off()
#cat((n/count))

}
Best(1000,0.9)
```

2.3 Sample graph

For n=1000;$\alpha = 0.9$



2.4 Output

- Theoritical Acceptance via Best method: 0.8819001
- Acceptance Obtained: 0.8710801

3 Proposed Methology

In this section, we provide the new gamma random number generator using the generalized exponential distribution.

Generalized exponential distribution

$$f_{GE}(x;\alpha) = \begin{cases} 0 & x < 0 \\ \alpha\lambda(1-e^{-\lambda x})^{\alpha-1}, & x > 0 \end{cases}$$

Gamma random variable for $\lambda = 1$ and also $0 < \alpha < 1$

$$f_{GA}(x;\alpha) = \frac{e^{-x}(x)^{(\alpha-1)}}{\Gamma(\alpha)}; x > 0$$

3.1 Algorithm 1

In this Algorithm we choose Generalized exponential density function $f_{GE}(x;\alpha;\frac{1}{2})$ to be our majorization function.

Supremum of $\frac{f_{GA}}{f_{GE}(x;\alpha;\frac{1}{2})} = \frac{2^{\alpha}}{\Gamma(\alpha+1)}$

3.1.1 Psuedo Code

- Generate U from uniform (0,1).
- Compute $X = -2\ln\left(1 - U^{\frac{1}{\alpha}}\right)$.
- Generate V from uniform (0,1) independent of U.
- if $V \leq \frac{X^{\alpha-1}e^{-X/2}}{2^{\alpha-1}\left(1-e^{-X/2}\right)^{\alpha-1}}$ accept X, otherwise goto beginning.

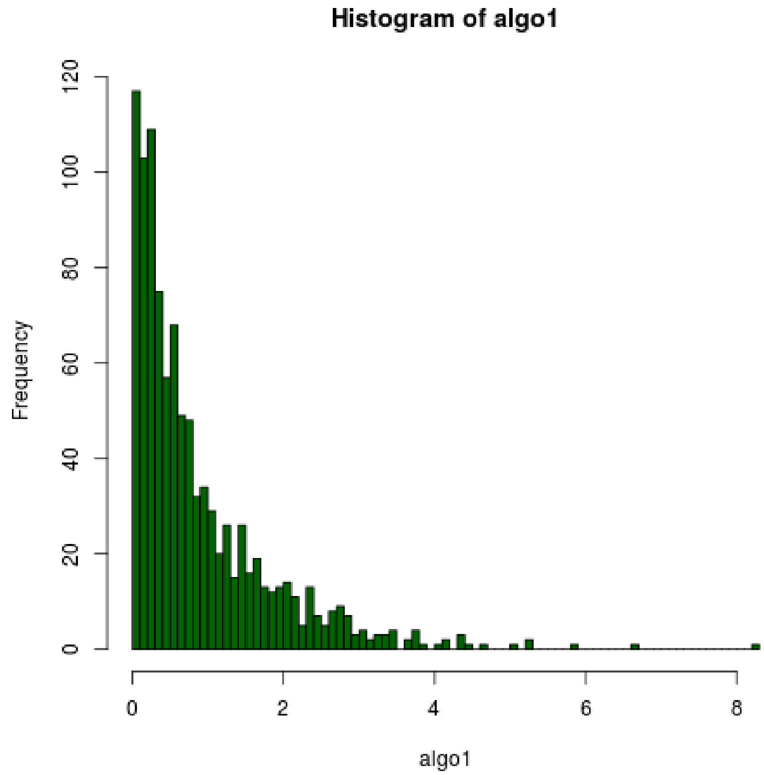
3.1.2 Code

```
alg1=function(n,alpha)
{
algo1=vector(length=n)
i=1;count=0
```

```
while(i<=n)
{
a=runif(2)
count=count+1
X=-2*log(1-(a[1]^(1/alpha)))
if (X!=0){
if (a[2]<=((X^(alpha-1))*(exp(-X/2)))/
((2*(1-exp(-1*X/2)))^(alpha-1)))
{
algo1[i]=X
i=i+1
}}
}
integrand=function(x){x^(alpha)*exp(-1*x)}
c=integrate(integrand,lower=0,upper=Inf)$value
cat("Theoritical acceptance via alg1:",c/2^(alpha),"\n")
#print(1/c)
cat("Acceptance Obtained:",n/count,"\n")
png("image3.png")
hist(algo1,breaks=100,col="darkgreen")
dev.off()
#cat((n/count))
}
alg1(1000,0.9)
```

3.1.3 Sample graph

For n=1000;$\alpha = 0.9$



3.1.4 Output

- Theoritical acceptance via alg1: 0.5153976
- Acceptance Obtained: 0.5216484

3.2 Algorithm 2

The bound proposed is sharp for $0 < x < 1$,but for $1 < x < \infty$ the bound is not very sharp.So the following majorization function $t_1(x, \alpha)$ of $f_{GA}(x, \alpha)$ is proposed i.e, $\forall\ x > 0\ f_{GA}(x, \alpha) \leq t_1(x, \alpha)$

It is observed that multiplication of GE function by a constant can be used as a majorization function in the interval $(0 < x < 1)$.The constant used is c.We propose the following **majorization** function:

We propose the following **majorization** function:

$$t_1(x;\alpha) = \begin{cases} \dfrac{2^\alpha}{\Gamma(\alpha+1)} f_{GE}(x;\alpha,\dfrac{1}{2}), & 0 < x < 1 \\ \dfrac{1}{\Gamma(\alpha)} e^{-x}, & x > 1 \end{cases}$$

$c_1 = \int_0^\infty t_1(x;\alpha)\,\mathrm{d}x = \frac{1}{\Gamma(\alpha+1)}\left[2^\alpha \left(1 - e^{\frac{-1}{2}}\right)^\alpha + \alpha e^{-1}\right]$

Now we define a probability density fumction r_1 which is obtained by dividing $t_1(x;\alpha)$ with c_1:

$$r_1(x;\alpha) = \frac{1}{c_1} t_1(x;\alpha); x > 0$$

which has the following distribution function:

$$R_1(x;\alpha) = \begin{cases} 0, & x < 0 \\ \dfrac{2^\alpha}{c_1\Gamma(\alpha+1)} \left(1 - e^{\frac{-x}{2}}\right)^\alpha, & 0 < x < 1 \\ 1 - \dfrac{1}{c_1\Gamma(\alpha)} e^{-x}, & x > 1 \end{cases}$$

3.2.1 Pseudo Code

Set $a = \frac{\left(1-e^{-1/2}\right)^\alpha}{\left(1-e^{-1/2}\right)^\alpha + \frac{\alpha e^{-1}}{2^\alpha}}$ and $b = \left(1 - e^{-1/2}\right) + \frac{\alpha e^{-1}}{2^\alpha}$

- Generate U from uniform (0,1).
- If U $\leq X = -2\ln\left[1 - (Ub)^{\frac{1}{\alpha}}\right]$, otherwise $X = -\ln\left[\frac{2^\alpha}{\alpha} b(1-U)\right]$.
- Generate V from uniform (0,1) independent of U. If X $\leq$ 1, check whether $V \leq \frac{X^{\alpha-1}e^{-X/2}}{2^{\alpha-1}\left(1-e^{-X/2}\right)^{\alpha-1}}$. If true return X, otherwise bo back to the beginning. If X $>$ 1, check whether V $\leq X^{\alpha-1}$. If true return X, otherwise go back to the beginning.

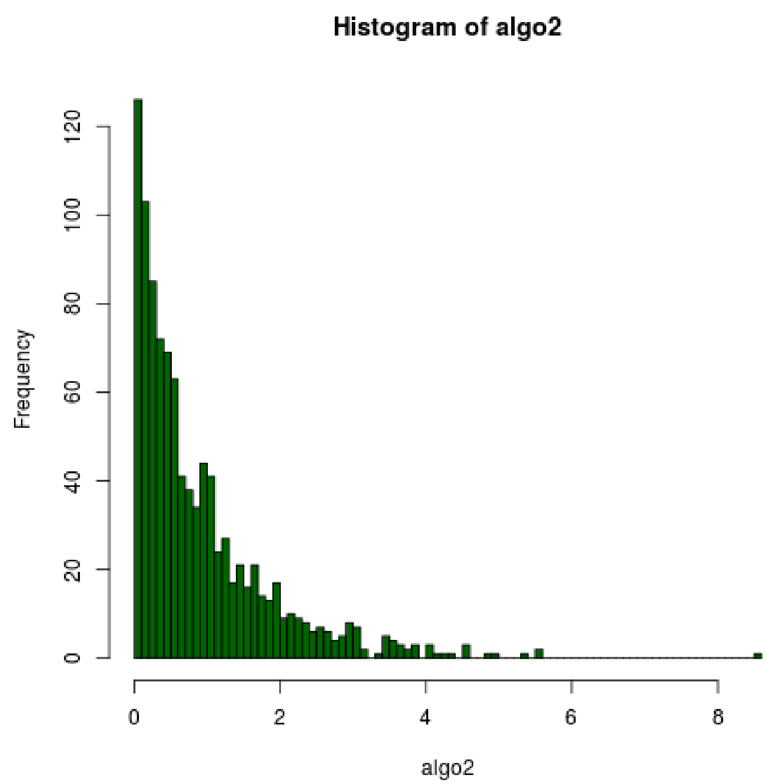
3.2.2 Code

```
alg2=function(n,alpha)
{
algo2=vector(length=n)
i=1;count=0
b=((1-exp(-0.5))^(alpha))+((alpha)*exp(-1))/2^(alpha)
#print(b)
a=((1-exp(-0.5))^(alpha))/b
#print(a)
while(i<=n)
{
arr=runif(2)
count=count+1
if (arr[1]<=a)
{
X=-2*log(1-(arr[1]*b)^(1/alpha))
}
else
{
X=-1*log(((2^(alpha))/(alpha))*b*(1-arr[1]))
}
if (X<=1 && X!=0 && (arr[2]<=((X^(alpha-1))*
(exp(-X/2)))/((2*(1-exp(-1*X/2)))^(alpha-1))))
{
algo2[i]=X
i=i+1
}
if (X>1 && arr[2]<=(X^(alpha-1)))
{
algo2[i]=X
i=i+1
}
}
integrand=function(x){x^(alpha)*exp(-1*x)}
c=integrate(integrand,lower=0,upper=Inf)$value
cat("Theoritical Acceptance via alg2:",
```

```
c/((2^(alpha))*b),"\n")
#print(1/c)
cat("Acceptance Obtained",n/count,"\n")
png("image4.png")
hist(algo2, breaks=100,col="darkgreen")
dev.off()
}
alg2(1000,0.9)
```

3.2.3 Sample graph

For n=1000;$\alpha = 0.9$



3.2.4 Output

- Theoritical Acceptance via alg2: 0.845796
- Acceptance Obtained 0.8591065

3.3 Algorithm 3

Now we propose the following modified majorization function:

$$t_2(x;\alpha) = \begin{cases} \dfrac{2^\alpha}{\Gamma(\alpha+1)} f_{GE}(x,\alpha,\dfrac{1}{2}), & 0 < x < d_\alpha \\ \dfrac{1}{\Gamma(\alpha)} d_\alpha^{\alpha-1} e^{-x}, & x > d_\alpha \end{cases}$$

$c_2 = \int_0^\infty t_2(x;\alpha)\,\mathrm{d}x = \frac{1}{\Gamma(\alpha+1)}\left[2^\alpha\left(1-e^{\frac{-d_\alpha}{2}}\right)^\alpha + \alpha d_\alpha^{\alpha-1}e^{-d_\alpha}\right]$

We now generate the following distribution function:

$$R_2(x;\alpha) = \begin{cases} 0, & x < 0 \\ \dfrac{2^\alpha}{c_2\Gamma(\alpha+1)}\left(1-e^{\frac{-x}{2}}\right)^\alpha, & 0 < x < d_\alpha \\ 1 - \dfrac{1}{c_2\Gamma(\alpha)} d_\alpha^{\alpha-1} e^{-x}, & x > d_\alpha \end{cases}$$

We denote the optimum choice of d_α as d_α^o.
We suggest the following approximation of the optimum d_α^o as d_α^*, where

$$d_\alpha^* = 1.0334 - 0.0766e^{2.2942\alpha}$$

3.3.1 Pseudo Code

Set $d = 1.0334 - 0.0766e^{2.2942\alpha}$, $a = 2^\alpha\left(1-e^{\frac{-d}{2}}\right)^\alpha$, $b = \alpha d^{\alpha-1}e^{-d}$ and $c = a+b$.

- Generate U from uniform (0,1).
- If U $\leq \frac{a}{a+b}$, then $X = -2\ln\left[1-\frac{(cU)^{1/\alpha}}{2}\right]$, otherwise $X = -\ln\left[\frac{c(1-U)}{\alpha d^{\alpha-1}}\right]$.
- Generate V from uniform (0,1). If X $\leq$ d, check whether $V \leq \frac{X^{\alpha-1}e^{-X/2}}{2^{\alpha-1}\left(1-e^{-X/2}\right)^{\alpha-1}}$. If true return X, otherwise bo back to the beginning. If X > d, check whether V $\leq \left(\frac{d}{X}\right)^{1-\alpha}$. If true return X, otherwise go back to the beginning.

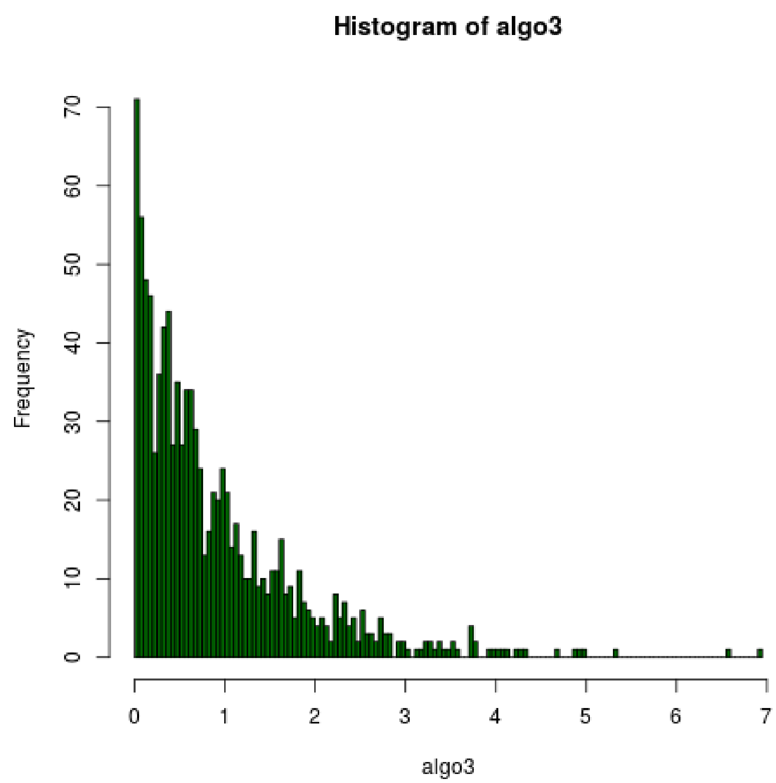
3.3.2 Code

```
alg3=function(n,alpha)
{
algo3=vector(length=n)
i=1;count=0
d=1.0334-0.0766*exp(2.2942*(alpha))
b=alpha*(d^(alpha-1))*exp(-1*d)
#print(b)
a=(2*(1-exp(-0.5*d)))^(alpha)
#print(a)
c=a+b
while(i<=n)
{
arr=runif(2)
count=count+1
if (arr[1]<=a/c)
{
X=-2*log(1-((c*arr[1])^(1/alpha))/2)
}
else
{
X=-log((c*(1-arr[1]))/(alpha*(d^(alpha-1))))
}
if (X<=d && X!=0 && (arr[2]<=((X^(alpha-1))*
(exp(-X/2)))/((2*(1-exp(-1*X/2)))^(alpha-1))))
{
algo3[i]=X
i=i+1
}
if (X>d && arr[2]<(d/X)^(1-alpha))
{
algo3[i]=X
i=i+1
}
}
```

```
integrand=function(x){x^(alpha)*exp(-1*x)}
d=integrate(integrand,lower=0,upper=Inf)$value
cat("Theoritical acceptance via alg3:",d/c,"\n")
#print(1/c)
cat("Acceptance Obtained:",n/count,"\n")
png("image5.png")
hist(algo3,breaks=100,col="darkgreen")
dev.off()
}
alg3(1000,0.9)
```

3.3.3 Sample graph

For n=1000;$\alpha = 0.9$



3.3.4 Output

- Theoritical acceptance via alg3: 0.9052209

- Acceptance Obtained: 0.8976661

4 Analysis on different methods

4.1 Code

```
alg1=function(n,alpha)
{
algo1=vector(length=n)
i=1;count=0
while(i<=n)
{
a=runif(2)
count=count+1
X=-2*log(1-(a[1]^(1/alpha)))
if (X!=0){
if (a[2]<=((X^(alpha-1))*(exp(-X/2)))/
((2*(1-exp(-1*X/2)))^(alpha-1)))
{
algo1[i]=X
i=i+1
}}
}
integrand=function(x){x^(alpha)*exp(-1*x)}
c=integrate(integrand,lower=0,upper=Inf)$value
return (n/count)
}
alg2=function(n,alpha)
{
algo2=vector(length=n)
i=1;count=0
b=((1-exp(-0.5))^(alpha))+((alpha)*exp(-1))/2^(alpha)
#print(b)
a=((1-exp(-0.5))^(alpha))/b
while(i<=n)
{
arr=runif(2)
```

```
count=count+1
if (arr[1]<=a)
{
X=-2*log(1-(arr[1]*b)^(1/alpha))
}
else
{
X=-1*log(((2^(alpha))/(alpha))*b*(1-arr[1]))
}
if (X<=1 && X!=0 && (arr[2]<=((X^(alpha-1))*
(exp(-X/2)))/((2*(1-exp(-1*X/2)))^(alpha-1))))
{
algo2[i]=X
i=i+1
}
if (X>1 && arr[2]<=(X^(alpha-1)))
{
algo2[i]=X
i=i+1
}
}
integrand=function(x){x^(alpha)*exp(-1*x)}
c=integrate(integrand,lower=0,upper=Inf)$value
return (n/count)
}
alg3=function(n,alpha)
{
algo3=vector(length=n)
i=1;count=0
d=1.0334-0.0766*exp(2.2942*(alpha))
b=alpha*(d^(alpha-1))*exp(-1*d)
a=(2*(1-exp(-0.5*d)))^(alpha)
c=a+b
while(i<=n)
{
arr=runif(2)
```

```
count=count+1
if (arr[1]<=a/c)
{
X=-2*log(1-((c*arr[1])^(1/alpha))/2)
}
else
{
X=-log((c*(1-arr[1]))/(alpha*(d^(alpha-1))))
}
if (X<=d && X!=0 && (arr[2]<=((X^(alpha-1))*
(exp(-X/2)))/((2*(1-exp(-1*X/2)))^(alpha-1))))
{
algo3[i]=X
i=i+1
}
if (X>d && arr[2]<(d/X)^(1-alpha))
{
algo3[i]=X
i=i+1
}
}
integrand=function(x){x^(alpha)*exp(-1*x)}
d=integrate(integrand,lower=0,upper=Inf)$value
return (n/count)
}
AD=function(n,alpha)
{
AD=vector(length=n)
i=1;count=0
e=exp(1)
#print(e/(e+alpha))
k=(e+alpha)/(e*alpha)
#print(k)
while(i<=n)
{
arr=runif(2)
```

```
count=count+1
if (arr[1]<=e/(e+alpha))
{
X=((arr[1]*(e+alpha))/e)^(1/alpha)
}
else
{
X=-log(k*(1-arr[1]))
}
#cat(arr[1],arr[2],X,"\n")
if (X<=1 && X!=0 && arr[2]<=exp(-X))
{
AD[i]=X
i=i+1
}
if (X>1 && arr[2]<=X^(alpha-1))
{
AD[i]=X
i=i+1
}
}
integrand=function(x){x^(alpha)*exp(-1*x)}
c=integrate(integrand,lower=0,upper=Inf)$value
return (n/count)
}
Best=function(n,alpha)
{
best=vector(length=n)
i=1;count=0
d=0.07+0.75*((1-alpha)^(0.5))
b=1+exp(-1*d)*alpha/d
while(i<=n)
{
arr=runif(2)
count=count+1
if (arr[1]<=1/b)
```

```
{
X=((b*arr[1])^(1/alpha))*d
}
else
{
X=-log(exp(-1*d)+(d-b*d*arr[1])/(alpha))
}
if (X<=d && X!=0 && arr[2]<=exp(-X))
{
best[i]=X
i=i+1
}
if (X>d && arr[2]<=(X/d)^(alpha-1))
{
best[i]=X
i=i+1
}
}
integrand=function(x){x^(alpha)*exp(-1*x)}
c=integrate(integrand,lower=0,upper=Inf)$value
return (n/count)
}
v=10
ADans=vector(length=v)
Bestans=vector(length=v)
Algo1ans=vector(length=v)
Algo2ans=vector(length=v)
Algo3ans=vector(length=v)
i=1
while(i<=v)
{

n=10000;alpha=0.1*i
ADans[i]=AD(n,alpha)
Bestans[i]=Best(n,alpha)
Algo1ans[i]=alg1(n,alpha)
```

```
Algo2ans[i]=alg2(n,alpha)
Algo3ans[i]=alg3(n,alpha)
i=i+1
}
x=seq(0,0.999,0.1)
png("Main.png")
plot(x,ADans,type="l",ylab="Acception Probability",
ylim=c(0,1),col=5)
lines(x,Bestans,col=2,type="l",lty=2)
lines(x,Algo1ans,col=3,type="l",lty=3)
lines(x,Algo2ans,col=4,type="l",lty=4)
lines(x,Algo3ans,col=1,type="l",lty=5)
legend('topleft',legend=c("ADans","Bestans","Algo1ans",
"Algo2ans","Algo3ans"),lty=1:2:3:4:5,col=5:2:3:4:1,
bty='n')
dev.off()
```

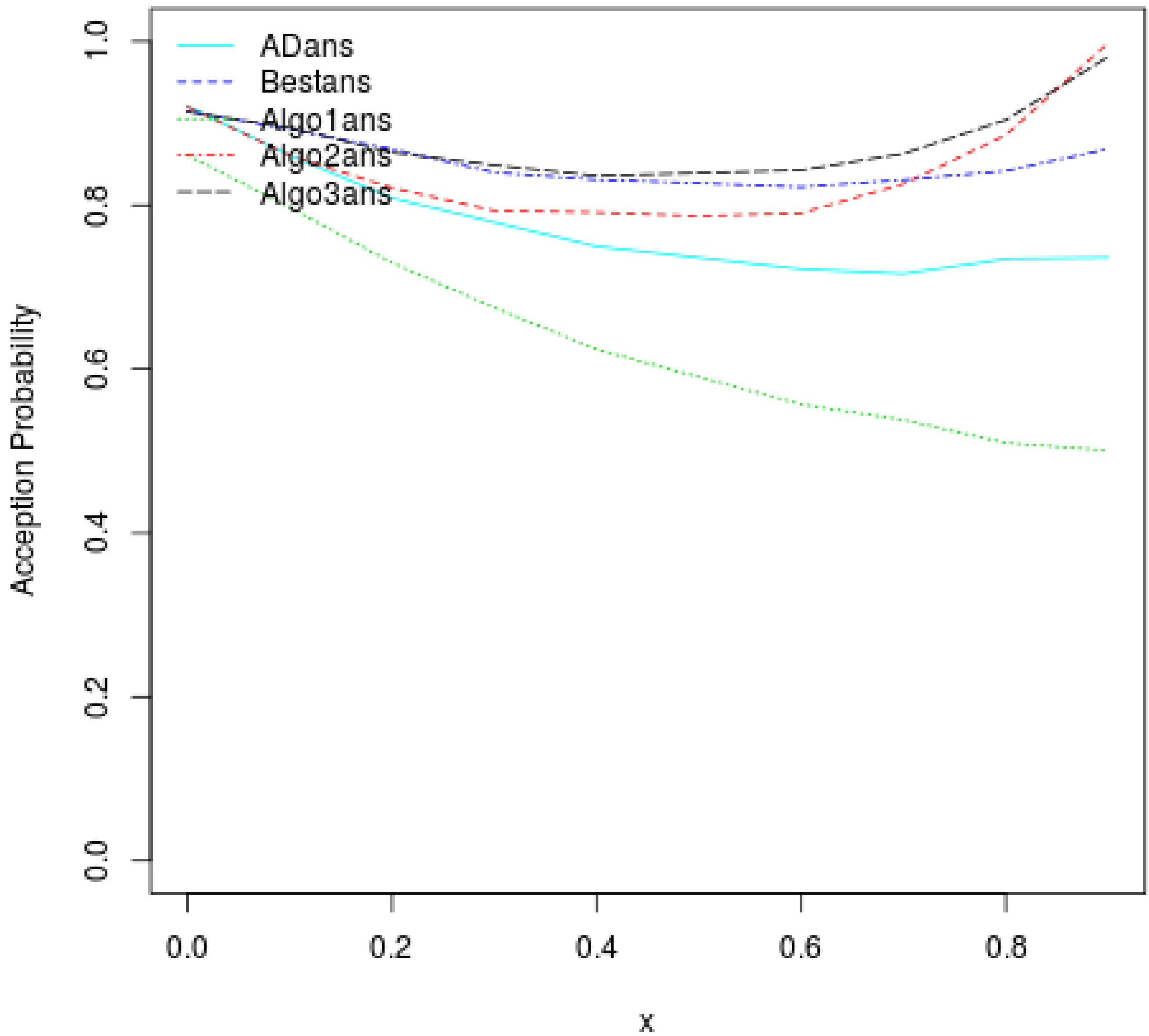
4.2 Tabulated Data

Acception Probability through various methods

α	AD Method	Best Method	Algorithm1	Algorithm2	Algorithm3
0.1	0.9141603	0.9198786	0.8608815	0.9203019	0.9126586
0.2	0.8514261	0.8586639	0.7924558	0.8945344	0.891504
0.3	0.8082114	0.8230453	0.7248478	0.8644537	0.8719156
0.4	0.7713074	0.7983395	0.6719527	0.8368901	0.8423181
0.5	0.7475518	0.7786949	0.6282196	0.8290499	0.8385041
0.6	0.7373544	0.7845599	0.5934366	0.8273352	0.8281573
0.7	0.7184424	0.7994244	0.5560807	0.8211529	0.8386448
0.8	0.7203573	0.8271983	0.5368839	0.8358409	0.8628128
0.9	0.7251632	0.877809	0.5147475	0.8489685	0.903424

4.3 Graph

For all methods



ADans
Bestans
Algo1ans
Algo2ans
Algo3ans
Acception Probability
0.0
0.2
0.4
0.6
0.8
1.0
x