Monte Carlo Simulation Lab

Project Report

Akul Goel 140123005
Vaibhav Saxena 140123041
Yash Vanjani 140123046
Koppar Manas Ravi 140123018
Soumitra Agarwal 140123036

April 13^{th}, 2016

0.1 Motivation

Generating gamma random numbers is an old and very important problem in the statistical literature. Several methods are available in the literature to generate gamma random numbers. It is well known that the available algorithms can be divided into two distinct cases:

Case I: Shape Parameter < 1
Case II: Shape Paramter > 1

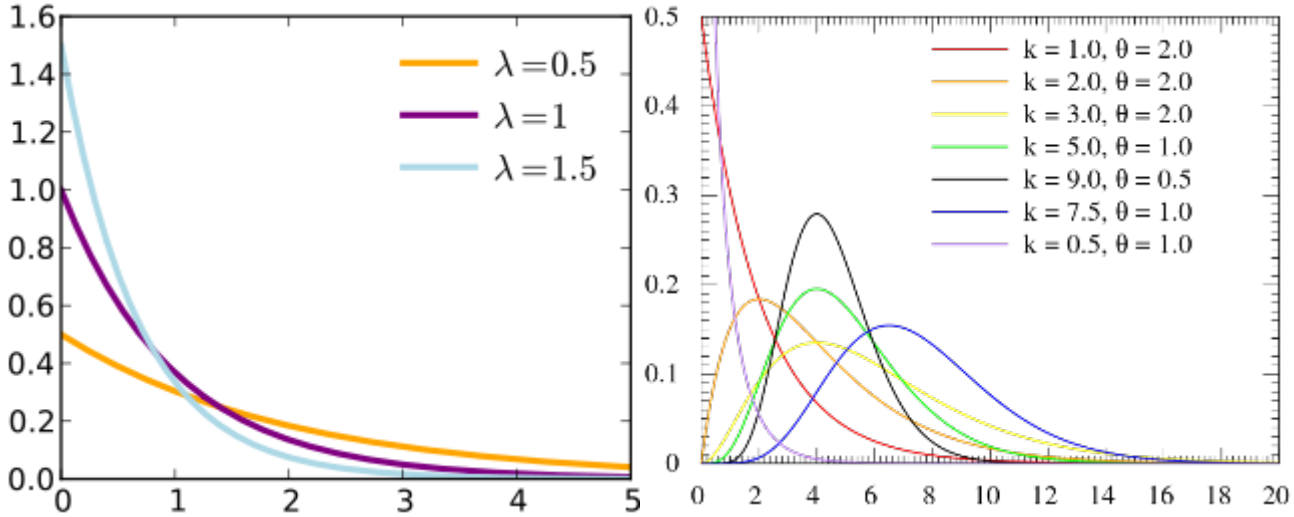
Although several methods are available for case 2, but for case 1, mainly two methods are well known; (a) the most popular and very simple method proposed by Ahrens & Dieter, (b) the modified Ahrens & Dieter's method proposed by Best. Both the methods mainly use the majorization functions and the acceptance-rejection principle. It is known that Best's method has greater acceptance proportion than Ahrens & Dieter's method.

The two-parameter Generalised Exponential Distribution for $\alpha,\ \lambda > 0$, has the following density function;

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

Here α and λ are the shape and scale parameters respectively. When $\alpha = 1$, it coincides with the exponential distribution. If $\alpha \leq 1$, the density function of a GE distribution is a strictly decreasing function and for $\alpha > 1$ it has uni-modal density function.
On the other hand, the probability density function of gamma distribution using the shape-scale parametrization is:

$$f(x;k,\theta) = \frac{x^{k-1}e^{-\frac{x}{\theta}}}{\theta^k\Gamma(k)}$$



(a) a (b) b

Figure 1: Showing similarity between General Exponential and Gamma (*shape parameter* < 1) distributions

From now on, we take $0 < \alpha < 1$, unless otherwise mentioned. It is observed that when the shape parameter is less than one, then a constant multiplication of the GE density function can be used as a majorization function. Therefore, using the acceptance-rejection principle, gamma random deviates can be easily generated using GE random numbers. It is observed that the proposed method has greater acceptance proportion than Ahrens & Dieter and Best generators for all $0 < \alpha < 1$. In section 2 we briefly describe the two most popular methods. The proposed methods are discussed in section 3. The numerical comparisons are provided in section 4 and finally we conclude the paper in section 5.

0.2 The Two Most Popular Methods - Ahren-Dieter and Best

In this section we will discuss the two most popular methods that are used to generate gamma random numbers when the shape parameter is less than 1. The first method is proposed by Ahrens & Dieter and the second one is by Best. Both the methods are based on the acceptance-rejection principle with proper choice of the majorization functions.

Ahrens & Dieter used the following majorization function:

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

Since $c_{AD} = \int_0^\infty t_{AD}(x;\alpha)\,dx = \frac{e+\alpha}{e\Gamma(\alpha+1)}$ therefore it needs to generate random deviate from the following probability density function:

$$r_{AD}(x;\alpha) = \frac{t_{AD}(x;\alpha)}{\int_0^\infty t_{AD}(x;\alpha)\,dx} = \begin{cases} \frac{\alpha e x^{\alpha-1}}{e+\alpha}, & \text{if } 0 < \text{x} < 1 \\ \frac{\alpha e}{e+\alpha} e^{-x}, & \text{if x} > 1 \end{cases}$$

and using the standard acceptance-rejection method, the gamma random deviate can be easily obtained. In this case, the rejection proportion is c_{AD} -1.

The implementation of the Ahrens & Dieter Method is as follows:

```
gammadist<-vector("numeric")
j<-0
NoOfRand<-10000
alpha<-0.4 # 0<alpha<=1
for(i in 1:NoOfRand)
{
    u<-runif(1)
    v<-runif(1)
    b<-(exp(1)+alpha)/exp(1)
    p<-b*u
    if(p>1) # case x>1
    {
        x<-(-1)*log((b-p)/alpha)
        if(v<=(x^(alpha-1)))
        {
            j<-j+1
```

```
            gammadist[j]=x
        }
    }
    else # case x<=1
    {
        x<-p^(1/alpha)
        if(v<=exp(-x))
        {
            j<-j+1
            gammadist[j]=x
        }
    }
}

acceptance_prob<-(j/NoOfRand)
cat("Acceptance probability =",acceptance_prob,"\n")

png(file="gammaAhrens.png")
hist(gammadist, xlab="Range of random number", col="red", border="black", breaks=50)
dev.off()
```

The plot of the intended Gamma Distribution looks like this

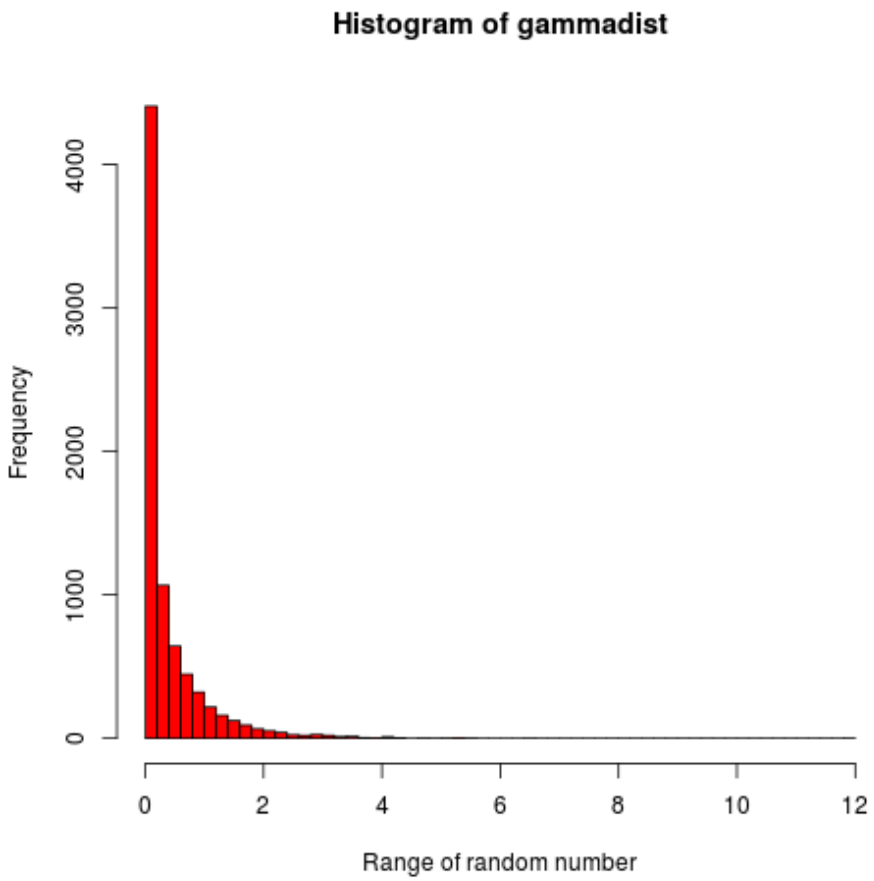


Figure 2: Using Ahrens & Dieter's Method to plot Gamma Distribution

Best modified Ahrens & Dieters method, by using the following majorization function:

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

Here d should be chosen in such a manner such that $\mathrm{c}_B = \int_0^\infty t_B(x;\alpha)\,dx$ is minimum. It can be easily seen that in that case d must satisfy the following non-linear equation

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

Best approximated the value of d as $d = 0.07 + 0.75\sqrt{1-\alpha}$. Now, the acceptance-rejection method can be easily used to generate gamma random deviate using the majorization function. Therefore, the probability density function is:

$$r_B(x;\alpha) = \frac{t_B(x;\alpha)}{\int_0^\infty t_B(x;\alpha)\,dx} = \begin{cases} \frac{\alpha x^{\alpha-1}}{bd^\alpha}, & \text{if } 0 < \mathrm{x} < \mathrm{d} \\ \frac{\alpha}{bd} e^{-x}, & \text{if } \mathrm{x} > \mathrm{d} \end{cases}$$

where $b = 1 + \frac{e^{-d}\alpha}{d}$.

The implementation of the Best's Method is as follows:

```
gammadist<-vector("numeric")
j<-0
NoOfRand<-10000
alpha<-0.4 # 0<alpha<=1
z<-0.07+(0.75*sqrt(1-alpha)) #this z is d of Kundu's paper
b<-1+(exp(-z)*alpha/z)
for(i in 1:NoOfRand)
{
   u<-runif(1)
   v<-runif(1)
   p<-b*u
   if(p>1)
   {
      x<-(-1)*log(z*(b-p)/alpha)
      y<-x/z
      if((v*(alpha+y-(alpha*y)))<=1) # approx. to increase speed of algo, if
          inequlity is not satisfied then anyways else condition is checked which
          contains actual inequlity. this approximation is given in Second
          modification of Algorithm GS in Best's paper
      {
         j<-j+1
         gammadist[j]=x
      }
      else if(v<=y^(alpha-1)) # actually inequality has to be checked at this step
          for acceptance
      {
         j<-j+1
         gammadist[j]=x
      }
   }
   else
   {
```

```
        x<-z*p^(1/alpha)
        if(v<=(2-x)/(2+x)) # approx. to increase speed of algo, if inequlity is not
            satisfied then anyways else condition is checked which contains actual
            inequlity. this approximation is given in Second modification of Algorithm
             GS in Best's paper
        {
            j<-j+1
            gammadist[j]=x
        }
        else if(v<=exp(-x)) # actually inequality has to be checked at this step for
            acceptance
        {
            j<-j+1
            gammadist[j]=x
        }
    }
}

acceptance_prob<-(j/NoOfRand)
cat("Acceptance probability =",acceptance_prob,"\n")

png(file="gammaBest_paper.png")
hist(gammadist, xlab="Range of random number", col="red", border="black", breaks=50)
dev.off()
```

The plot of the intended Gamma Distribution looks like this

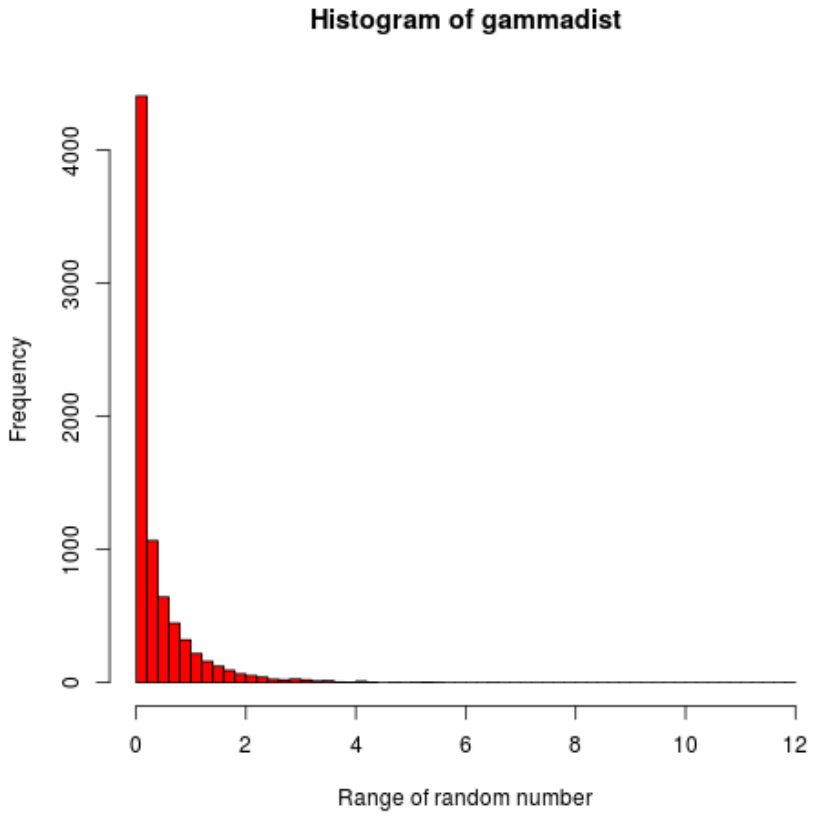


Figure 3: Using Best's Method to plot Gamma Distribution

It has been shown theoretically that Best's method has lower rejection proportion that Ahrens & Dieter's method.

0.3 Proposed Algorithms

In this section, we provide the new gamma random number generator using the generalized exponential distribution. Observe that for all x $\geq$ 0,

$$1 - e^{-x} \leq x.$$

Suppose β = 1-α, the for all x $\geq$ 0, we have

$$\frac{x^{\alpha-1}e^{-x}}{(1-e^{-x})^{\alpha-1}} = \frac{e^{-x}(1-e^{-x})^{\beta}}{x^{\beta}} \leq e^{-x}$$

Therefore,

$$\frac{\frac{1}{\Gamma(\alpha)}x^{\alpha-1}e^{-x}}{\frac{\alpha}{2}(1-e^{-x/2})^{\alpha-1}e^{-x/2}} = \frac{2e^{-x/2}x^{\alpha-1}}{\Gamma(\alpha+1)(1-e^{-x/2})^{\alpha-1}} \leq \frac{2^{\alpha}}{\Gamma(\alpha+1)}$$

Now,

$$f_{GA}(x;\alpha) \leq \frac{2^{\alpha}}{\Gamma(\alpha+1)} f_{GE}(x;\alpha,\frac{1}{2})$$

Based on this, the proposed algorithm is as follows:

ALGORITHM 1:

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

The implementation of Algorithm 1 is as follows:

```
noofrand <- 10000

u1 <- runif(noofrand, min=0, max=1)

# Generating Generalized Exponential distribution
alpha <- 0.4   #arbit
X <- -2*log(1-(u1^(1/alpha)))

u2 <- runif(noofrand, min=0, max=1)

# Generating Gamma distribution using Acceptance-Rejection Method
gammadist <- vector('numeric')
for(i in 1:noofrand) {
   val <- ( (X[i]^(alpha-1))*exp(-0.5*X[i]) )/( (2^(alpha-1)) * ((1-exp(-0.5*X[i]))
      ^(alpha-1)) )
```

```
    if(u2[i] <= val) {
        gammadist <- c(gammadist, X[i])
    }
}
# print(length(X))
# print(length(gammadist))

acceptance_prob<-(length(gammadist)/length(X))
cat("Acceptance probability =",acceptance_prob,"\n")

png(file="gamma1.png")
hist(gammadist, xlab="Range of random number", col="red", border="black", breaks=50)
dev.off()
```

The plot of the intented Gamma Distribution looks like this

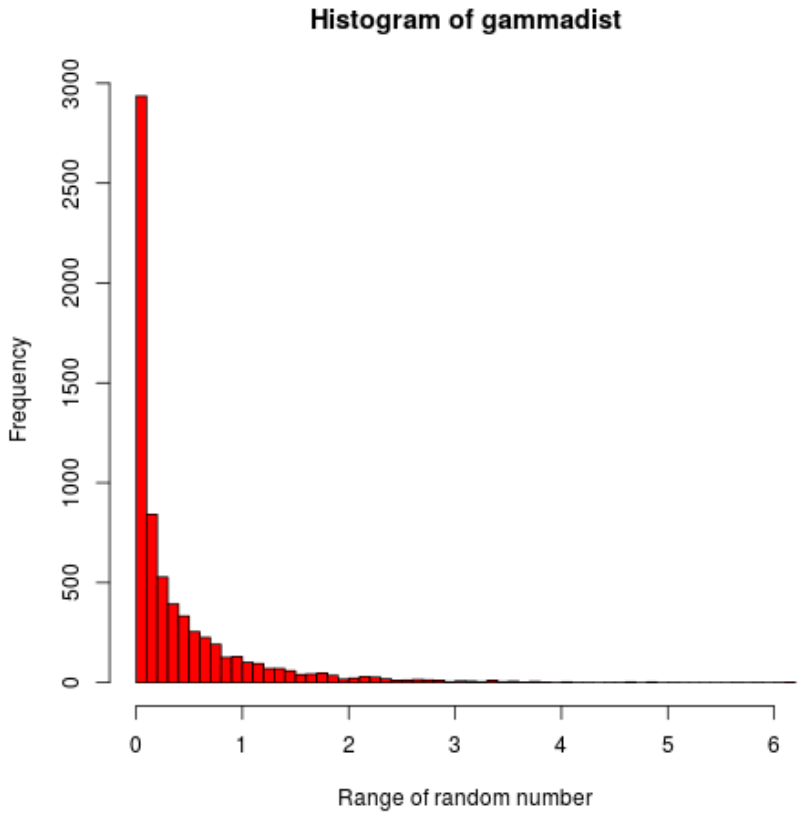


Figure 4: Using proposed algorithm 1 to plot Gamma Distribution

It is observed that the bound provided by the majorization function for $0 < x < 1$ is quite sharp, but for $1 < x < \infty$ the bound is not very sharp. Therefore, we propose the following majorization function $t_1(x;\alpha)$ of $f_{GA}(x;\alpha)$, *i.e.*, for all $x > 0$, $f_{GA}(x,\alpha) \leq t_1(x;\alpha)$, where

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

Note that,

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

We need to generate from the following density function;

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

which has the following distribution function:

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

Now we propose the following modified algorithm.

ALGORITHM 2:

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

1. Generate U from uniform (0,1).
2. If $U \leq a$, then $X = -2ln[1-(Ub)^{1/\alpha}]$, otherwise $X = -ln[\frac{2^\alpha}{\alpha}b(1-U)]$
3. Generate V from uniform (0,1). If $X \leq 1$, check whether $v \leq \frac{X^{\alpha-1}e^{-X/2}}{2^{\alpha-1}(1-e^{-X/2})^{\alpha-1}}$. If true return X, otherwise go back to 1. If $X > 1$, check whether $V \leq X^{\alpha-1}$. If true return X, otherwise go back to 1.

The implementation of Algorithm 2 is as follows:

```
func_a<-function(x)
{
    return(((1-exp(-0.5))^x)/(((1-exp(-0.5))^x)+(x*exp(-1)/(2^x))))
}
func_b<-function(x)
{
    return(((1-exp(-0.5))^x)+(x*exp(-1)/(2^x)))
}
NoOfRand<-10000
alpha<-0.4 # arbitrary
a<-func_a(alpha)
b<-func_b(alpha)
j<-0
gammadist<-vector("numeric")
# x(u,alpha,lambda=1); x is value of majorization function
for(i in 1:NoOfRand)
{
    u<-runif(1) # u is 1st random number
    if(u<=a)
        {x<-(-2)*log(1-((u*b)^(1/alpha)))}
    else
        {x<-(-1)*log((2^alpha)/alpha*b*(1-u))}
    v<-runif(1) # v is 2nd random number
    if(x<=1)
```

```
    {
       #acceptance condition of x (generally being v<=f(x)/(c*g(x)))
       if(v<=(((x^(alpha-1))*exp(-x/2))/((2^(alpha-1))*((1-exp(-x/2))^(alpha-1)))))
       {
          j<-j+1
          gammadist[j]=x
       }
    }
    else
    {
       #acceptance condition of x (generally being v<=f(x)/(c*g(x)))
       if(v<=(x^(alpha-1)))
       {
          j<-j+1
          gammadist[j]=x
       }
    }
}
acceptance_prob<-(j/NoOfRand)
cat("Acceptance probability =",acceptance_prob,"\n")

png(file="gamma2.png")
hist(gammadist, xlab="Range of random number", col="red", border="black", breaks=50)
dev.off()
```

The plot of the intented Gamma Distribution looks like this

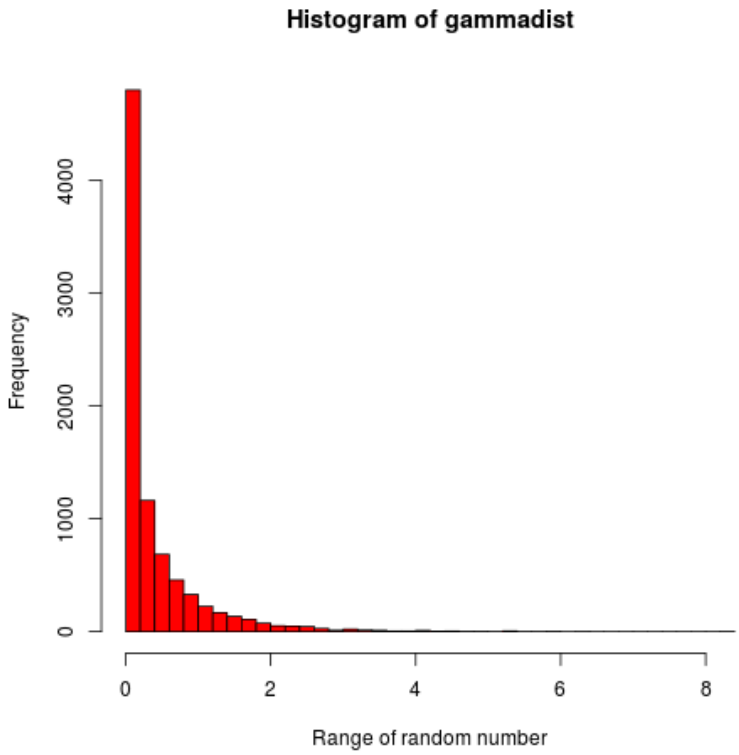


Figure 5: Using proposed algorithm 2 to plot Gamma Distribution

Note that the majorization function $t(x;\alpha)$ has two part envelop with the change point at 1, similar to the Ahrens and Dieters algorithm. But it was suggested by Atkinson and Pearce and later implemented by Best that the change point might depend on the shape parameter α and it enhanced the performance. Using that idea, we propose the following majorization function for $f_{GA}(x;\alpha) \leq t_2(x;\alpha)$, where

$$t_2(x;\alpha) = \begin{cases} \frac{2^\alpha}{\Gamma(\alpha+1)} f_{GE}(x;\alpha,\frac{1}{2}), & \text{if } 0 < \mathrm{x} < \mathrm{d}_\alpha \\ \frac{1}{\Gamma(\alpha)} e^{-x}, & \text{if } \mathrm{x} > 1 \end{cases}$$

How to choose the optimum d_α will be discussed shortly. Note that for $d_\alpha = 1$, $t_1(x;\alpha) = t_2(x;\alpha)$. Now

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

Note that $c_2 \geq 1$ is the normalizing constant and $c_2 - 1$ denotes the rejection proportion. Therefore, we should choose d_α, so that c_2 is minimum for a given α. It has to be obtained iteratively by solving a non-linear equation. We denote the optimum d_α as d_α^o. We suggest the following approximation of the optimum d_α^o as $d_\alpha^\star$, where

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

Therefore, now we have the final algorithm:

ALGORITHM 3:

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

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

The implementation of Algorithm 3 is as follows:

```
noofrand <- 10000

alpha <- 0.4

d <- 1.0344 - 0.0766*exp(2.2942*alpha)
a <- (2^alpha)*((1-exp(-0.5*d))^alpha)
b <- alpha*(d^(alpha-1))*exp(-1*d)
c <- a+b

u1 <- runif(noofrand, min=0, max=1)
X <- vector('numeric')

for(i in 1:noofrand) {
   if(u1[i] <= a/(a+b)) {
```

```
        val <- -2*log( 1 - 0.5*(c*(u1[i]^(1/alpha))) )
        X <- c(X, val)
    }
    else {
        val <- -1*log( c*(1-u1[i])/(alpha*(d^(alpha-1))) )
        X <- c(X, val)
    }
}

u2 <- runif(noofrand, min=0, max=1)

gammadist <- vector('numeric')

for(i in 1:noofrand) {
    if(X[i] <= d) {
        val <- ( (X[i]^(alpha-1))*exp(-0.5*X[i]) ) / ( (2^(alpha-1))*((1-exp(-0.5*X[i
            ]))^(alpha-1)) )
        if(u2[i] <= val) {
            gammadist <- c(gammadist, X[i])
        }
    }
    else {
        val <- (d/X[i])^(1-alpha)
        if(u2[i] <= val) {
            gammadist <- c(gammadist, X[i])
        }
    }
}

#print(length(X))
#print(length(gammadist))
acceptance_prob<-(length(gammadist)/length(X))
cat("Acceptance probability =",acceptance_prob,"\n")

png(file="gamma3.png")
hist(gammadist, xlab="Range of random number", col="red", border="black", breaks=50)
dev.off()
```

The plot of the intented Gamma Distribution looks like this

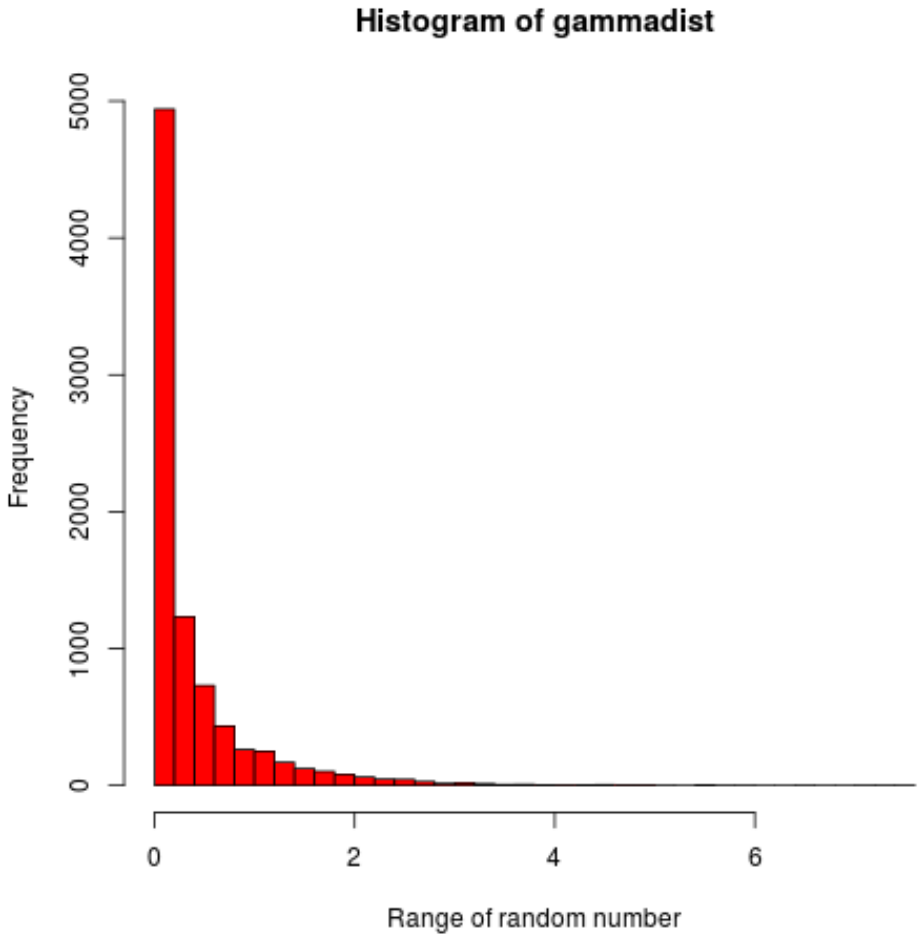


Figure 6: Using proposed algorithm 3 to plot Gamma Distribution

0.4 Numerical Comparison

In this section we compare different methods numerically. Note that all the methods discussed here, are based on the acceptance-rejection principle. We compare different methods in terms of their rejection proportion or equivalently the expected number of the uniform random deviates to be generated to produce one gamma variate.

The metric used for comparison of the algorithms is the area under the majorization function (c). Area between the pdf's of the majorization function and the gamma function is the rejection proportion. Hence, greater the area, greater is the rejection proportion, less efficient is the algorithm.

The algorithm to plot the area under majorization function is below:

```
cAD <- function(alpha) {
    val <- (1+exp(-1)*alpha)/gamma(alpha+1)
    return(val)
}
```

```
cB <- function(alpha) {
    d <- 1.0334 - 0.0766*exp(2.2942*alpha)
    val <- (d + exp(-1*d)*alpha)*(d^(alpha-1))/gamma(alpha+1)
    return(val)
}

c1 <- function(alpha) {
    val <- (2^alpha)/gamma(alpha+1)
    return(val)
}

c2 <- function(alpha) {
    val <- ((2^alpha)*((1-exp(-0.5))^alpha) + alpha*exp(-1))/gamma(alpha+1)
    return(val)
}

c3 <- function(alpha) {
    d <- 1.0334 - 0.0766*exp(2.2942*alpha)
    val <- ((2^alpha)*((1-exp(-0.5*d))^alpha) + alpha*(d^(alpha-1))*exp(-1*d))/gamma(
        alpha+1)
    return(val)
}

alphas <- (1:100)/100

cAD_vec <- vector('numeric')
for(i in alphas) {
    cAD_vec <- c(cAD_vec, cAD(i))
}

cB_vec <- vector('numeric')
for(i in alphas) {
    cB_vec <- c(cB_vec, cB(i))
}

c1_vec <- vector('numeric')
for(i in alphas) {
    c1_vec <- c(c1_vec, c1(i))
}

c2_vec <- vector('numeric')
for(i in alphas) {
    c2_vec <- c(c2_vec, c2(i))
}

c3_vec <- vector('numeric')
for(i in alphas) {
    c3_vec <- c(c3_vec, c3(i))
}

png(file="cAD.png")
plot(alphas,cAD_vec, type='l', xlab="alpha", col="red", ylim=c(1,2))
dev.off()
png(file="cB.png")
plot(alphas,cB_vec, type='l', xlab="alpha", col="red", ylim=c(1,2))
dev.off()
```

```
png(file="c1.png")
plot(alphas,c1_vec, type='l', xlab="alpha", col="red", ylim=c(1,2))
dev.off()
png(file="c2.png")
plot(alphas,c2_vec, type='l', xlab="alpha", col="red", ylim=c(1,2))
dev.off()
png(file="c3.png")
plot(alphas,c3_vec, type='l', xlab="alpha", col="red", ylim=c(1,2))
dev.off()

# all together
png(file="comparison.png")
plot(alphas,cAD_vec, type='l', lty=2, ylab="", xlab="alpha", col="black", ylim=c
    (1,2))
par(new=T)
plot(alphas,cB_vec, type='l', lty=2, ylab="", xlab="alpha", col="azure4", ylim=c
    (1,2))
par(new=T)
plot(alphas,c1_vec, type='l', lty=1, ylab="", xlab="alpha", col="coral1", ylim=c
    (1,2))
par(new=T)
plot(alphas,c2_vec, type='l', lty=1, ylab="", xlab="alpha", col="brown3", ylim=c
    (1,2))
par(new=T)
plot(alphas,c3_vec, type='l', lty=1, ylab="", xlab="alpha", col="brown4", ylim=c
    (1,2))
par(new=T)
dev.off()
```

The areas under the majorization functions for the five algorithms (expected numbers) are as follows:

1. Ahrens and Dieter Method:

$$c_{AD} = \frac{1 + e^{-1}\alpha}{\Gamma(\alpha + 1)}$$

The plot of c_{AD} looks as follows:

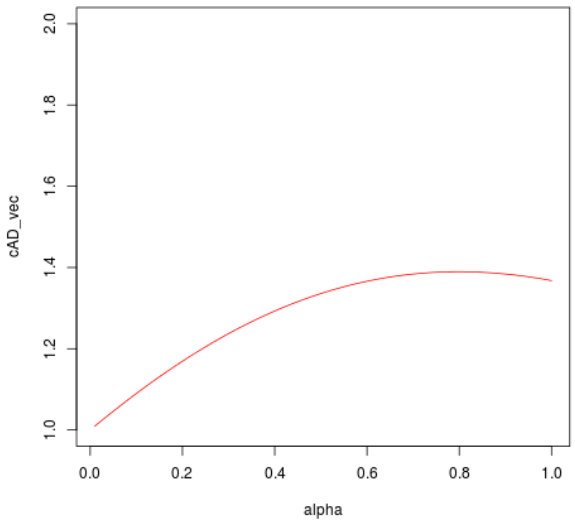


Figure 7: Area under majorization function using Ahrens & Dieter Method

2. Best Method:

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

The plot of c_B looks as follows:

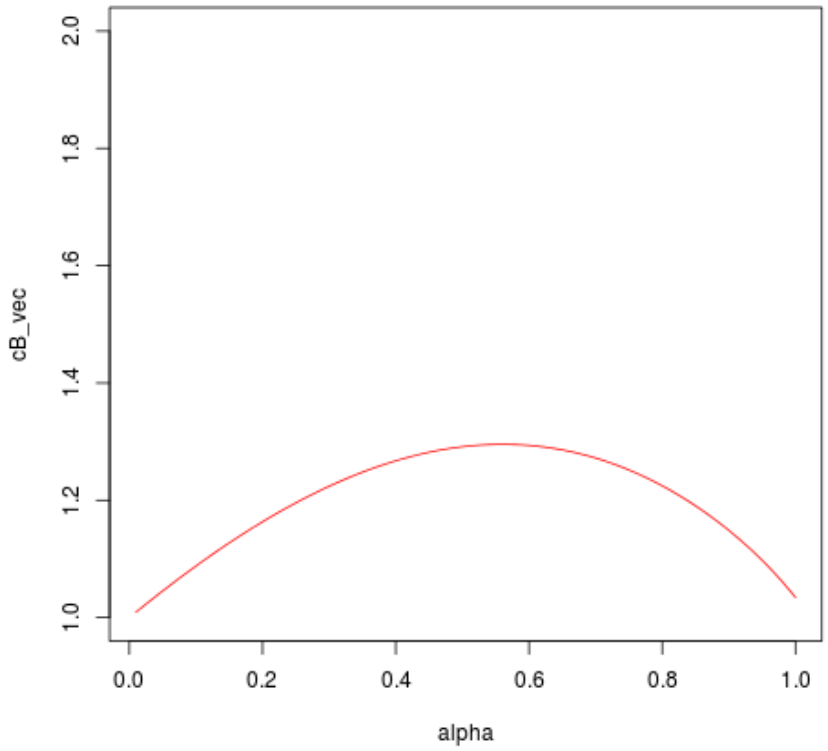


Figure 8: Area under majorization function using Best Method

3. Algorithm 1:

$$c_1 = \frac{2^{\alpha}}{\Gamma(\alpha+1)}$$

The plot of c_1 looks as follows:

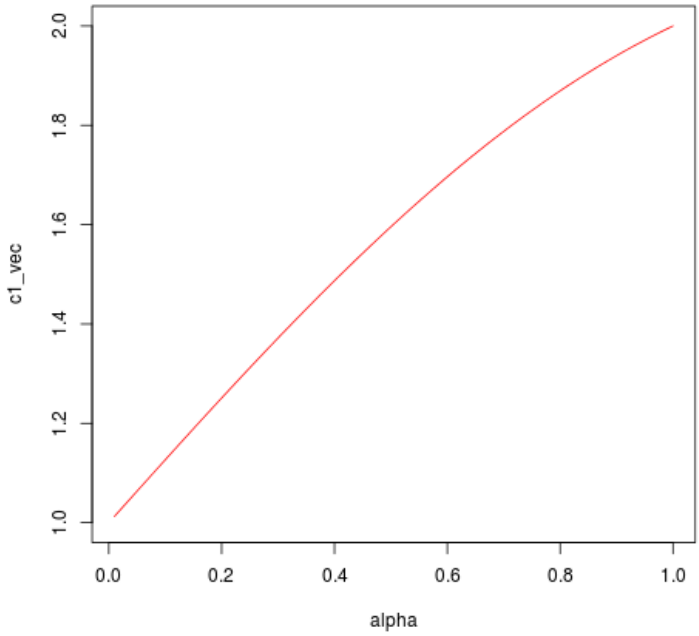


Figure 9: Area under majorization function using Algorithm 1

4. Algorithm 2:

$$c_2 = \frac{1}{\Gamma(\alpha+1)}[2^{\alpha}(1-e^{-1/2})^{\alpha} + \alpha e^{-1}]$$

The plot of c_2 looks as follows:

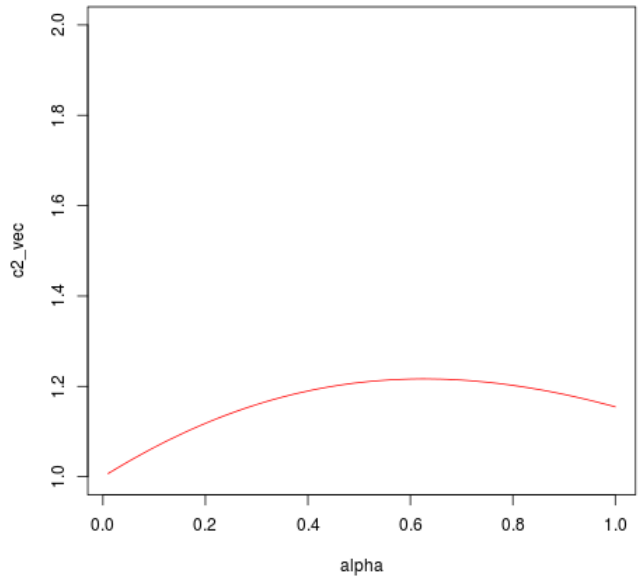


Figure 10: Area under majorization function using Algorithm 2

5. Algorithm 3:

$$c_3 = \frac{1}{\Gamma(\alpha+1)}[2^{\alpha}(1-e^{-d_{\alpha}^{\star}/2})^{\alpha} + \alpha d_{\alpha}^{\star(\alpha-1)} e^{-d_{\alpha}^{\star}}]$$

The plot of c_3 looks as follows:

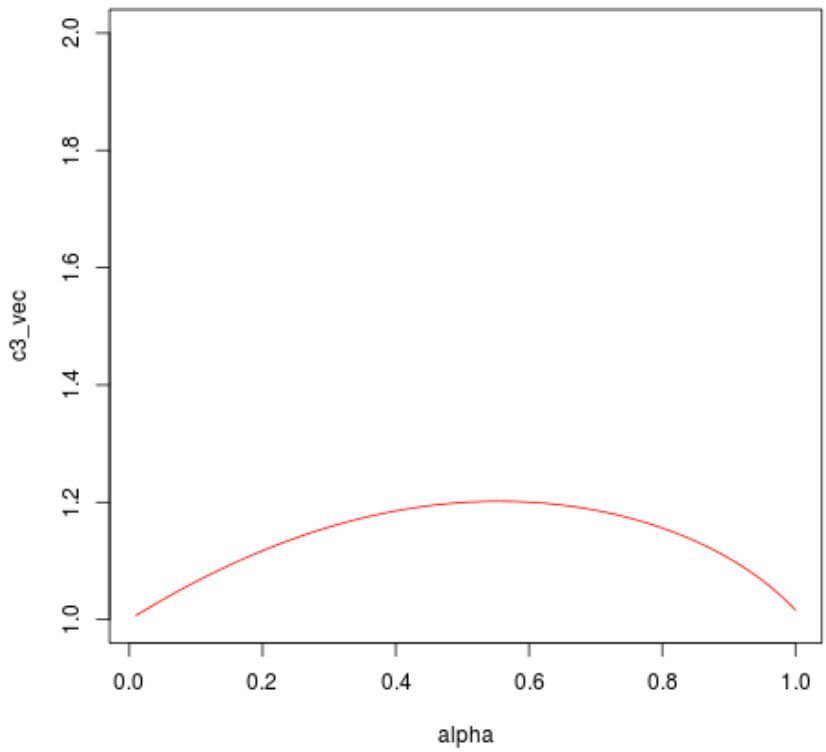


Figure 11: Area under majorization function using Algorithm 3

We report $\Gamma(\alpha+1)c$ for different values of α in Table 1. We also report $\Gamma(\alpha+1)c_4$, where c_4 is obtained from c_3 by replacing $d_{\alpha}^{\star}$ with d_{α}^{o}, where the latter is the non-approximated efficient value of d_{α}. From the table and also from the graphs it is clear that our proposed final algorithm Algorithm 3 has lower rejection proportion than Best's method.

The plot of all scaled areas under the majorization functions looks as follows:

α	$\Gamma(\alpha+1)c_{AD}$	$\Gamma(\alpha+1)c_B$	$\Gamma(\alpha+1)c_1$	$\Gamma(\alpha+1)c_2$	$\Gamma(\alpha+1)c_3$	$\Gamma(\alpha+1)c_4$
.1000	1.0368	1.0328	1.0718	1.0131	1.0129	1.0129
.2000	1.0736	1.0630	1.1487	1.0268	1.0261	1.0261
.3000	1.1104	1.0897	1.2311	1.0410	1.0392	1.0392
.4000	1.1472	1.1121	1.3195	1.0558	1.0517	1.0517
.5000	1.1839	1.1289	1.4142	1.0710	1.0632	1.0632
.6000	1.2207	1.1383	1.5157	1.0868	1.0725	1.0725
.7000	1.2575	1.1381	1.6245	1.1031	1.0780	1.0780
.8000	1.2943	1.1246	1.7411	1.1199	1.0769	1.0768
.9000	1.3311	1.0906	1.8661	1.1371	1.0625	1.0624

Table 1: The areas under majorization functions (expected numbers) multiplied by $\Gamma(\alpha+1)$ for different methods and for different values of α

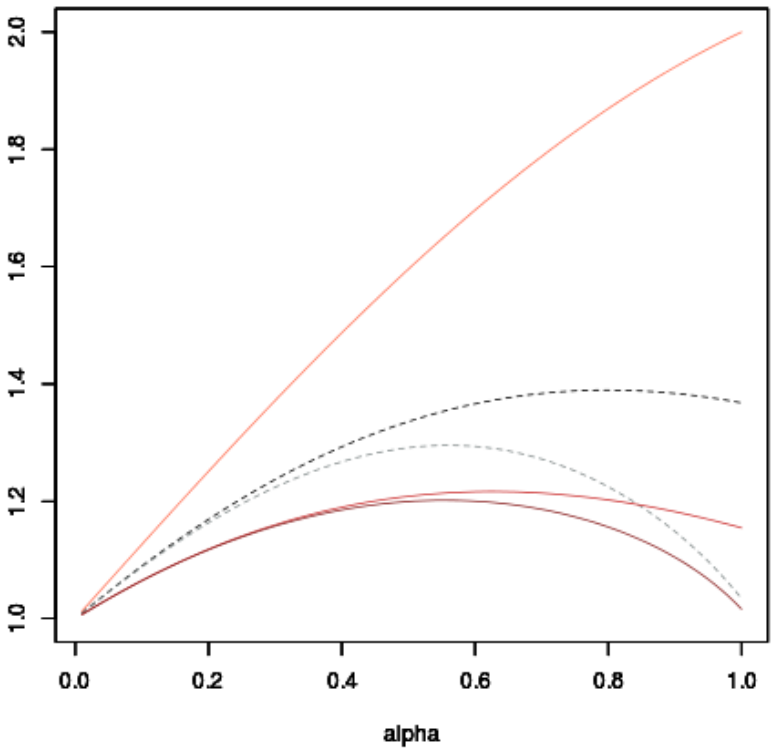


Figure 12: Area under majorization functions of the five algorithms

In the above graph, broken lines represent the existing methods, and the solid lines represent proposed algorithms. The darker broken line represents Ahrens & Dieter method, while the lighter one represents Best method. Among the solid lines, the red line represents Algorithm 1, the dark red represents Algorithm 2 and the brown one represents Algorithm 3.

From the above graph, it is evident that the rejection proportion of the Algorithm-3 is lowest among all algorithms, even lower than the algorithm proposed by Best, for all values of shape

parameter (alpha) less than 1. Hence the proposed algorithm is more efficient than Ahren-Dieter and Best Algorithms for generating Gamma distribution with shape parameter less than 1.

0.5 Conclusion

In this paper, we proposed three new algorithms of generating gamma variates from generalized exponential distribution for shape parameter less than 1. It is observed that our algorithm has a greater acceptance proportion than popular Ahrens & Dieter or Best method. The proposed algorithms were under the motivation that the Gamma distribution with shape parameter less than 1 resembles the Generalized Exponential distribution.

The metric used for judging the performance of the algorithm was the area under the majorization function, which is a measure of the rejection proportion. For the last algorithm, we observe that this area is minimum when compared to other algorithms, for all values of the shape parameter less than 1.

0.6 References

1. Ahrens, J.H. and Dieter, U. (1974) "Computer methods for sampling from gamma, beta, Poisson and Binomial distribution", Computing, vol. 12, 223 - 246.

2. Best, D.J. (1983), "A note on gamma variate generators with shape parameter less than unity", Computing, vol. 30, 185 - 188.

3. Kundu, D. and Gupta, R.D. (2007) "A Convenient Way of Generating Gamma Random Variables Using Generalized Exponential Distribution".