A Convenient Way of Generating Normal Random Variables Using Generalized Exponential Distribution

Monte Carlo Simulation

Under Prof. Arabin Kumar Dey

Prithvi Sriram 150123052
Sangeet Chandaliya 150123037
Paridhi Kothari 150123030
Akash Mahalik 150123004
Achintya Singh 150123002

April 9, 2017

1 MOTIVATION

Generating Normal Random numbers is an old and very important in the statistical literature. The book of Johnson, Kotz and Balakrishnan provides an extensive list of reference of different algorithms available today, the most popular ones being Box-Muller transformation method and it's improvement as suggested by Marsagilia and Bary. Most of the statistical packages like SAS, IMSL, SPSS, S-PLUS, or Numerical Recipes use this method.In this project we explore paper 'A Convenient Way of Generating Normal Random Variables Using Generalized Exponential Distribution' by Debasis Kundu, Rameshwar D Gupta, and Anubhav Manglick.

2 INTRODUCTION

The two-parameter GE distribution has the following distribution function;

$$F_{GE}\left(x;\alpha,\lambda\right) = \left(1 - e^{-\lambda x}\right)^{\alpha}; \alpha, \lambda > 0$$

for x>0 and 0 otherwise. The corresponding density function is;

$$f_{GE}\left(x;\alpha,\lambda\right) = \alpha\lambda\left(1 - e^{-\lambda x}\right)^{\alpha-1} e^{-\lambda x}; \alpha, \lambda > 0$$

for x>0 and 0 otherwise. 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 unimodal density function.

In a recent study by Kundu, Gupta and Manglick, it was observed that in certain cases log-normal distribution can be approximated quite well by GE distribution and vice versa. In fact for certain ranges of the shape parameters of the GE distributions the distance between the GE and log-normal distributions can be very small. The main idea of the paper studied was to use this particular property of a GE distribution to generate log-normal random variables and in turn generate normal random variables. It may be mentioned that the GE distribution function is an analytically invertible function, therefore, the generation of GE random variables is immediate using uniform random variables.

3 PROPOSED ALGORITHMS

In this paper we denote the density function of a log-normal random variable with scale parameter θ and shape parameter σ as

$$f_{LN}\left(x;\theta,\sigma\right)=\frac{1}{\sqrt{2\pi}x\sigma}e^{-\frac{(\ln x-\ln\theta)^2}{2\sigma^2}};\theta,\sigma>0$$

for x $>$ 0 and 0 otherwise. If X is a log-normal random variable with scale parameter θ and shape parameter σ, then

$$E\left(X\right)=\theta e^{\frac{\sigma^2}{2}}\qquad \mathit{and}\qquad V\left(X\right)=\theta^2e^{\sigma^2}\left(e^{\sigma^2}-1\right)$$

Note that $\ln X$ is a normal random variable with mean $\ln\theta=\mu$ and variance σ^2. We equate the first two moments of the two distribution functions to compute σ and θ from a given α and λ. Without loss of generality we take $\lambda=1$.For a given $\alpha=\alpha_0$, we obtain:

$$\theta e^{\frac{\sigma^2}{2}}=A_0$$

$$\theta^2e^{\sigma^2}\left(e^{\sigma^2}-1\right)=B_0$$

Therefore, solving the above, we obtain:

$$\ln\theta_0=\mu_0=\ln A_0-\frac{1}{2}\ln\left(1+\frac{B_0}{A_0^2}\right)$$

$$\sigma_0=\sqrt{\ln\left(1+\frac{B_0}{A_0^2}\right)}$$

Using the above equations, we can easily generate standard normal random variables as Follows:

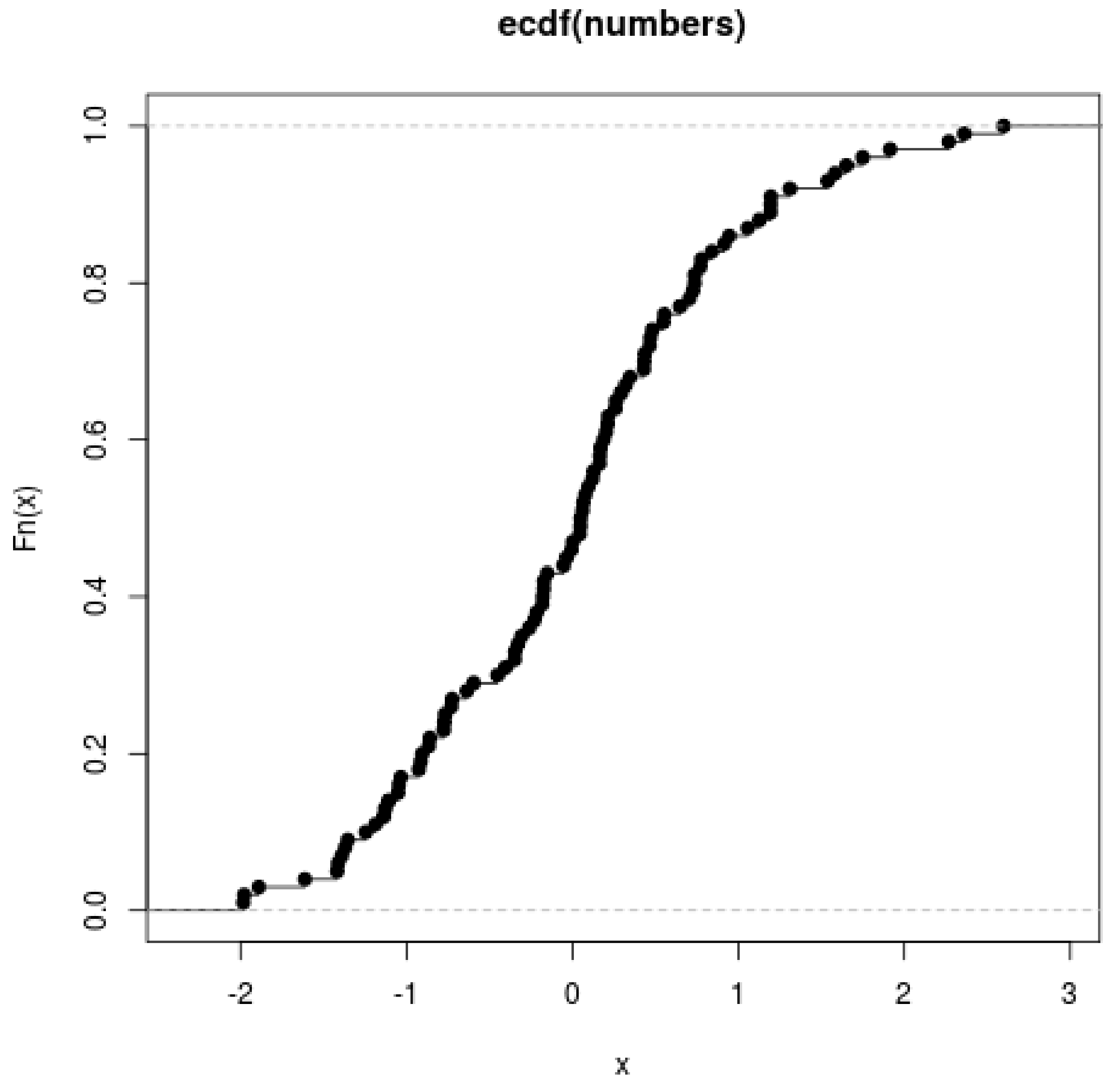
Algorithm:

- Step 1: Generate U an uniform (0,1) random variable.
- Step 2: For a fixed α_0, generate $X=-\ln\left(1-U^{\frac{1}{\alpha_0}}\right)$. Note that X is a generalized exponential random variable with shape parameter α_0 and scale parameter 1.
- Step 3: Compute Z $=\frac{\ln X-\mu_0}{\sigma_0}$. Here Z is the desired standard normal random variable.

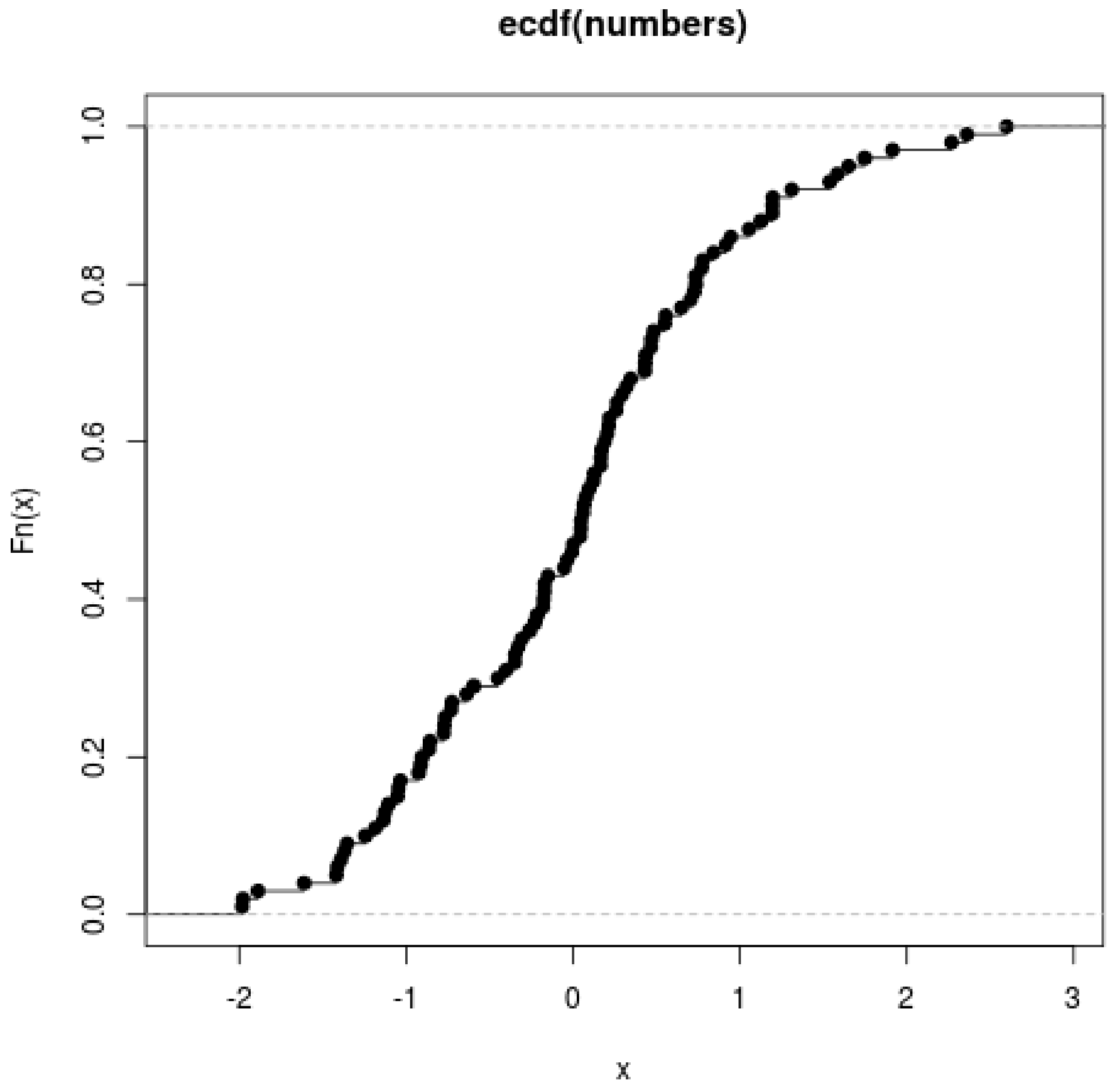
The code is as follows:

```
GeneralisedDistribution <- function(n,alpha) {
  Generalised_Distribution<-c()
  u<-runif(n,0,1)
  x<-(-log(1-u**(1/alpha)))


  m=log(mean(x))-0.5*log(1+(var(x)/(mean(x)**2)))
  s=sqrt(log(1+var(x)/(mean(x)**2)))
  #print(m)
  #print(s)
  Generalised_Distribution<-(log(x)-m)/s
  #z<-ks.test(Generalised_Distribution,ecdf(Generalised_Distribution))
  return(Generalised_Distribution)
}

n<-c(10,20,30,40,50,100)
alpha<-c(1,4,7,10,13)


for (i in 1:6){
  for (j in 1:5) {
    #print(n,alpha)
    numbers<-GeneralisedDistribution(n[i],alpha[j])
    #plot(numbers)
    print(numbers)
    print(mean(numbers))
    print(sd(numbers))
    png(paste("graph_", n[i],".png"))
    plot(ecdf(numbers))
    dev.off()
    }
}
```

Listing 1: Code in R



Standard Normal Random numbers from Genralized Exponential (using moments eqns)



Standard Normal Random numbers from Genralized Exponential (using L-moments eqns)

An alternative approximation is also possible. Instead of equating the moments of the two distributions, we can equate the corresponding L-moments also. The L-moments of any distribution are analogous to the conventional moments but they are based on the quantiles and they can be estimated by the linear combination of order statistics, i:e: by L-statistics.It is observed that in a similar study of approximating gamma distribution by generalized exponential distribution that the L-moments perform better than the ordinary moments.

The two L-moments of a log-normal distribution are :

$$\lambda_1 = \theta e^{\frac{\sigma^2}{2}} \qquad and \qquad \lambda_2 = \theta e^{\frac{\sigma^2}{2}} erf\left(\frac{\sigma}{2}\right)$$

where erf $(x) = 2\Phi\left(\sqrt{2}x\right) - 1$ and $\Phi(x)$ is the distribution function of the standard normal distribution.
Therefore, as before equating the first two L-moments for a given $\alpha = \alpha_0$ and for $\lambda = 1$, we obtain

$$\theta e^{\frac{\sigma^2}{2}} = A_0$$
$$\theta e^{\frac{\sigma^2}{2}} erf\left(\frac{\sigma}{2}\right) = B_1$$

Solving the above equations, we obtain the solutions of $\theta and \sigma$ as

$$\ln\theta_1 = \mu_1 = \ln A_0 - \frac{\sigma_1^2}{2}$$

$$\sigma_1 = \sqrt{2}\Phi^{-1}\left(\frac{1}{2}\left(1 + \frac{B_1}{A_0}\right)\right)$$

Therefore in the proposed algorithm, instead of using $(\mu_0, \sigma_0), (\mu_1, \sigma_1)$ also can be used.

4 NUMERICAL COMPARISONS

In this section first we try to determine the value of α_0 , so that the distance between the generalized exponential distribution and the corresponding log-normal distribution is minimum. All the computations are performed using Pentium IV processor . We consider the distance function between the two distribution functions as the Kolmogorv-Smirnov (K-S) distance only. To be more precise we compute the K-S distance between the GE, with the shape and scale parameter as $\alpha 0$ and 1 respectively, and log-normal distribution with the corresponding shape and scale parameter as $\sigma_0\,(\sigma_1)$ and $\theta_0\,(\theta_1)$ respectively . We believe that the distance function should not make much difference, any other distance function may be considered also. It is observed that as α_0 increases from 0 the K-S distance first decreases and then increases. When we have used the moments (L-moments) equations, the minimum K-S distance occurred at $\alpha_0 = 12.9$ (12.8). When $\alpha_0 = 12.9$ (12.8), then from the above equations, we obtained the corresponding $mu_0 = 1.0820991$ ($\mu_1 = 1.0792510$) and $\sigma_0 = 0.3807482$ ($\sigma_1 = 0.3820198$).
Now to compare our proposed method with the other existing methods we use mainly the K-S statistics and the corresponding p-values. The method can be described as follows. We generate standard normal random variables for different sample sizes namely n = 10, 20, 30, 40, 50 and 100 by using Box-Muller (BM) method, Marsaglia-Bray (MB) method, Acceptance-Rejection (AR) method, using moments equations (MM) and using L-moments equations (LM). In each case we compute the K-S distance and the corresponding p-value between the empirical distribution function and the standard normal distribution function. We replicate the process 10,000 times and compute the average K-S distances, the average p-values and the corresponding standard deviations. The results are reported in Table 1. In each case the standard deviations are reported within bracket

below the average values. From the table values it is quite clear that, based on the K-S distances and p values the proposed methods work quite well.

We also try to compute P (Z ≤ z) using the proposed approximation, where Z denotes the standard normal random variable. Note that,

$$P(Z \leq z) \approx \left(1 - e^{-e^{z\sigma 0 + \mu_0}}\right)^{12.9} \qquad or \qquad P(Z \leq z) \approx \left(1 - e^{-e^{z\sigma 1 + \mu_1}}\right)^{12.8}$$

We report the results in Table 2. It is clear from Table 2 that using μ_0 and σ_0 the maximum error can be 0.0005, where as using μ_1 and σ_1 , the maximum error can be 0.0003. From Table 2, it is clear that L-moments approximations work better than the moments approximations.

The code is as follows:

```
GeneralisedDistribution <- function(n) {
  Generalised_Distribution<-c()
  alpha<-12.9
  u<-runif(n,0,1)
  x<-(-log(1-u**(1/alpha)))


  m=log(mean(x))-0.5*log(1+(var(x)/(mean(x)**2)))
  s=sqrt(log(1+var(x)/(mean(x)**2)))
  #print(m)
  #print(s)
  Generalised_Distribution<-(log(x)-m)/s
  #z<-ks.test(Generalised_Distribution,ecdf(Generalised_Distribution))
  return(Generalised_Distribution)
}

BoxMuller<-function(n) {
  Box_Muller<-c()

  #x<-ks.test(Box_Muller,ecdf(Box_Muller))
  while (j<=n) {
    u<-runif(2,0,1)
    r<--2*log(u[1])
    v<-2*pi*u[2]
    Box_Muller[j]<-sqrt(r)*cos(v)
    Box_Muller[j+1]<-sqrt(r)*sin(v)
    j<-j+2
  }
  return(Box_Muller)
}

MarsagliaBay<-function(n) {
  j<-1
  Marsaglia_Bay<-c()
  j<-1
  while (j<=n) {
    u<-runif(2,0,1)
    v<-2*u-1
    if (v[1]**2+v[2]**2<1) {
      Marsaglia_Bay[j]<-(sqrt(-log(v[1]**2+v[2]**2))*v[1])/sqrt(v[1]**2+v
      [2]**2)
```

```
        Marsaglia_Bay[j+1]<-(sqrt(-log(v[1]**2+v[2]**2))*v[2])/sqrt(v[1]**2+v
    [2]**2)
        j<-j+2
      }
  }
  #x<-(ks.test(Marsaglia_Bay,ecdf(Marsaglia_Bay)))
  return(Marsaglia_Bay)
}

AcceptanceRejection<-function(n) {
  j<-1
  Acceptance_Rejection<-c()
  while (j<=n) {
    u<-runif(2,0,1)
    y<--log(u)
    if (y[2]>=((y[1]-1)**2)/2){
      v<-runif(1,0,1)
      if (v<=0.5) {
        Acceptance_Rejection[j]<-y[1]
      }
      else {
        Acceptance_Rejection[i]<--y[i]
      }
      j<-j+1
    }
  }
  #t<-ks.test(Acceptance_Rejection,ecdf(Acceptance_Rejection))
  return(Acceptance_Rejection)
}


LMoment<-function(n) {
  alpha<-12.9
  u <-runif(n,0,1)
  gd <--log(1-u**(1/alpha))
  A0 <-mean(gd)
  sig <-sqrt(log( (var(gd)/(mean(gd)^2)) + 1 ))
  B1 <-mean(gd)*(2*pnorm(sqrt(2)*sig/2)-1)
  sig1 <-sqrt(2)*qnorm(0.5*(1+B1/A0))
  mu1 <-log(A0)-sig1*sig1/2
  Z <-(log(gd)-mu1)/sig1
  #cat("n = ",n,", alpha = ",alpha,"\n")
  #print(mu1)
  #print(sig1)
  return(Z)
}


n<-c(10,20,30,40,50,100)
for (i in 1:6) {
  bm<-BoxMuller(n[i])
  mb<-MarsagliaBay(n[i])
  ar<-AcceptanceRejection(n[i])
  ge<-GeneralisedDistribution(n[i])
  lm<-LMoment(n[i])
  print(n[i])
  a<-rnorm(n[i])
  v<-ks.test(ge,a)
  w<-ks.test(lm,a)
  x<-ks.test(bm,a)
  y<-ks.test(mb,a)
  z<-ks.test(ar,a)


  print(v)
  print(w)
  print(x)
  print(y)
  print(z)
  #ad<-AhrenDiester(n[i])
  #mm<-MomentMethod(n[i])
  #lm<LMoment(n[i])

}



u <-runif(1000,0,1)
gd <--log(1-u**(1/alpha))
A0 <-mean(gd)
sig <-sqrt(log( (var(gd)/(mean(gd)^2)) + 1 ))
B1 <-mean(gd)*(2*pnorm(sqrt(2)*sig/2)-1)

sig1 <-sqrt(2)*qnorm(0.5*(1+B1/A0))
mu1 <-log(A0)-sig1*sig1/2

mu0=log(mean(gd))-0.5*log(1+(var(gd)/(mean(gd)**2)))
sig0=sqrt(log(1+var(gd)/(mean(gd)**2)))

comp<-c()
for(i in 1:31)
{0
  comp[i]<-(i-1)/10.0
}
tabl<-matrix(nrow = 30,ncol = 4)
for(i in 1:31)
{
  tabl[i,1]=comp[i]
  tabl[i,2]=(1-exp(-exp(comp[i]*sig0 + mu0)))^12.9
  tabl[i,3]=pnorm(comp[i])
  tabl[i,4]=(1-exp(-exp(comp[i]*sig1 + mu1)))^12.8
}
print(tabl)
#print(ad)
#print(mm)
#print(lm)

#t<-sort(t)
#png("M.png");
#plot.ecdf(t)
#par(new=TRUE)
#plot(pnorm(t),col="red")
#dev.off()
```

5 CONCLUSION

In this simulation we have provided a very simple and convenient method of generating normal random variables and even compared it with different other methods.

Even simple scientific calculator can be used to generate normal random number from the uniform generator very quickly. It is also observed that the standard normal distribution function can be approximated at least up to three decimal places using the simple approximations.

References

[1] Gupta, R D and Kundu, D (2001), "Exponentiated exponential distribution: An alter- native to gamma and Weibull distributions", Biometrical Journal, vol. 43, no. 1, 117-130.

[2] Kundu, D., Gupta, R.D. and Manglick, A. (2005), "Discriminating between log-normal and generalized exponential distribution", Journal of Statistical Planning and Inference, vol. 127, 213-227.

[3] Gupta, R.D. and Kundu, D. (2003), "Closeness of gamma and generalized exponential distribution", Communications in Statistics - Theory and Methods, vol. 32, no. 4, 705- 721.

Table 1: Comparision of K-S and p values

n		BM	MB	AR	MM	LM
10	K-S	0.2587	0.2587	0.2597	0.2586	0.2587
		(0.0796)	(0.0796)	(0.0809)	(0.0794)	(0.0795)
	p	0.5127	0.5128	0.5109	0.5135	0.5132
		(0.2938)	(0.2938)	(0.2970)	(0.2930)	(0.2931)
20	K-S	0.1851	0.1851	0.1871	0.1866	0.1867
		(0.0571)	(0.0571)	(0.0575)	(0.0571)	(0.0572)
	p	0.5178	0.5178	0.5068	0.5089	0.5085
		(0.2934)	(0.2934)	(0.2934)	(0.2927)	(0.2928)
30	K-S	0.1532	0.1532	0.1533	0.1524	0.1525
		(0.0467)	(0.0467)	(0.0466)	(0.0465)	(0.0465)
	p	0.5094	0.5094	0.5086	0.5150	0.5145
		(0.2937)	(0.2937)	(0.2923)	(0.2930)	(0.2930)
40	K-S	0.1331	0.1331	0.1331	0.1334	0.1334
		(0.0409)	(0.0488)	(0.0410)	(0.0410)	(0.0410)
	p	0.5111	0.5111	0.5121	0.5097	0.5092
		(0.2923)	(0.2923)	(0.2926)	(0.2927)	(0.2928)
50	K-S	0.1191	0.1191	0.1197	0.1199	0.1200
		(0.0370)	(0.0370)	(0.0364)	(0.0366)	(0.0366)
	p	0.5140	0.5140	0.5071	0.5058	0.5053
		0.5140	0.5140	0.5071	0.5058	0.5053
100	K-S	0.5140	0.5140	0.5071	0.5058	0.5053
		(0.0257)	(0.0257)	(0.0262)	(0.0259)	(0.0259)
	p	0.5059	0.5059	0.5096	0.5082	0.5077
		(0.2914)	(0.2914)	(0.2932)	(0.2912)	(0.2912)

Table 2: Φ(z) vs Moment vs L-Moment values

z	L-Moment	Exact	Moment
0.0	0.49984	0.50000	0.50014
0.1	0.53981	0.53983	0.54006
0.2	0.57935	0.57926	0.57955
0.3	0.61808	0.61791	0.61824
0.4	0.65564	0.65541	0.65574
0.5	0.69168	0.69145	0.69174
0.6	0.72594	0.72572	0.72595
0.7	0.75818	0.75800	0.75815
0.8	0.78822	0.78810	0.78814
0.9	0.81593	0.81588	0.81582
1.0	0.84125	0.84127	0.84112
1.1	0.86416	0.86424	0.86400
1.2	0.88469	0.88482	0.88452
1.3	0.90292	0.90308	0.90273
1.4	0.91893	0.91911	0.91875
1.5	0.93288	0.93305	0.93269
1.6	0.94490	0.94505	0.94472
1.7	0.95517	0.95528	0.95500
1.8	0.96385	0.96392	0.96369
1.9	0.97112	0.97114	0.97097
2.0	0.97714	0.97711	0.97701
2.1	0.98209	0.98200	0.98197
2.2	0.98610	0.98597	0.98600
2.3	0.98933	0.98916	0.98924
2.4	0.99189	0.99170	0.99181
2.5	0.99390	0.99370	0.99384
2.6	0.99547	0.99526	0.99542
2.7	0.99667	0.99647	0.99663
2.8	0.99759	0.99739	0.99755
2.9	0.99827	0.99809	0.99825
3.0	0.99878	0.99861	0.99876