Name: Somya Khandelwal

Roll Number: 200123056

Department: Mathematics and Computing

Course: MA 323 - Monte Carlo Simulation

Lab: 04

In [1]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import math
import time
```

In [2]:

```
def LCG(a,b,m,seed):
  xi=seed
  xi=(xi*a+b)%m
  return xi
```

1) Box-Muller method

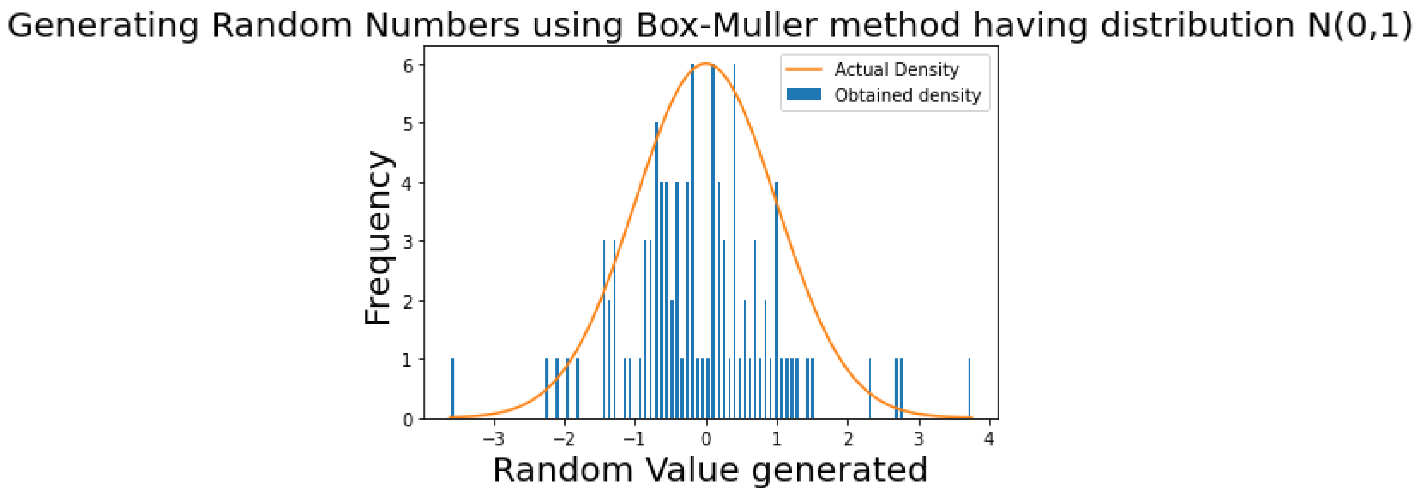
In [3]:

```
N= [100, 10000]
Z= []
Mean= []
Variance= []
Time= []
```

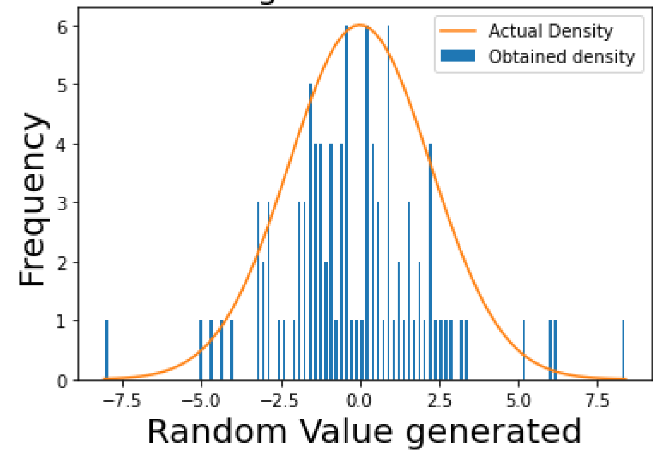
In [4]:

```
for n in N:
  start= time.time()
  seed_u1=3
  seed_u2=5
  for i in range(0,n//2):
    seed_u1=LCG(1741,2731,12960,seed_u1)
    seed_u2=LCG(1741,2731,12960,seed_u2)
    U1=seed_u1/12960
    U2=seed_u2/12960
    if(U1!=0):
      R= -2*math.log(U1)
    V= 2*math.pi*U2
    Z1= math.sqrt(R)*math.cos(V)
    Z2= math.sqrt(R)*math.sin(V)
    Z.append(Z1)
    Z.append(Z2)
  end= time.time()
  # print(len(Z))
  Time.append(end-start)
  mean= sum(Z) / len(Z)
  variance= sum((k - mean) ** 2 for k in Z) / len(Z)
  Mean.append(mean)
  Variance.append(variance)
```

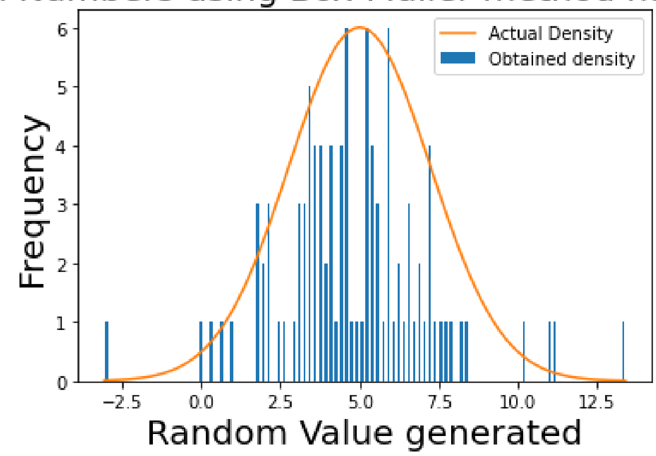
```
  x1,x2,x3= plt.hist(Z, bins = 100, rwidth = 0.5, label= "Obtained density")
  y= []
  c= (1/math.sqrt(2.00*math.pi))*pow(math.e,(-1*0*0)/2.0)
  for x in x2:
    f= (1/math.sqrt(2.00*math.pi))*pow(math.e,(-1*x*x)/2.0)
    y.append((f*max(x1))/c)
  plt.plot(x2,y, label= "Actual Density")
  plt.xlabel("Random Value generated", size=20)
  plt.ylabel("Frequency", size=20)
  plt.title("Generating Random Numbers using Box-Muller method having distribution N(0,1)
", size=20)
  plt.legend()
  plt.show()

  # Now for N(0,5)
  Z1= []
  for i in Z:
      Z1.append(0+math.sqrt(5)*i)
  x1,x2,x3= plt.hist(Z1, bins = 100, rwidth = 0.5, label= "Obtained density")
  y= []
  c= (1/math.sqrt(2.00*math.pi*5))*pow(math.e,(-1*0*0)/(2.0*5.0))
  for x in x2:
      f= (1/math.sqrt(2.00*math.pi*5))*pow(math.e,(-1*x*x)/(2.0*5.0))
      y.append((f*max(x1))/c)
  plt.plot(x2,y, label= "Actual Density")
  plt.xlabel("Random Value generated", size=20)
  plt.ylabel("Frequency", size=20)
  plt.title("Generating Random Numbers using Box-Muller method having distribution N(0,5)
", size=20)
  plt.legend()
  plt.show()

  # Now for N(5,5)
  Z2= []
  for i in Z:
      Z2.append(5+math.sqrt(5)*i)
  x1,x2,x3= plt.hist(Z2, bins = 100, rwidth = 0.5, label= "Obtained density")
  y= []
  c= (1/math.sqrt(2.00*math.pi*5))*pow(math.e,(-1*0*0)/(2.0*5.0))
  for x in x2:
      f= (1/math.sqrt(2.00*math.pi*5))*pow(math.e,(-1*(x-5)*(x-5))/(2.0*5.0))
      y.append((f*max(x1))/c)
  plt.plot(x2,y, label= "Actual Density")
  plt.xlabel("Random Value generated", size=20)
  plt.ylabel("Frequency", size=20)
  plt.title("Generating Random Numbers using Box-Muller method having distribution N(5,5)
", size=20)
  plt.legend()
  plt.show()
print("Mean for N=100 is ", Mean[0])
print("Mean for N=10000 is ", Mean[1])
print("Variance for N=100 is ", Variance[0])
print("Variance for N=10000 is ", Variance[1])
print("Time for N=100 is ", Time[0])
print("Time for N=10000 is ", Time[1])
```



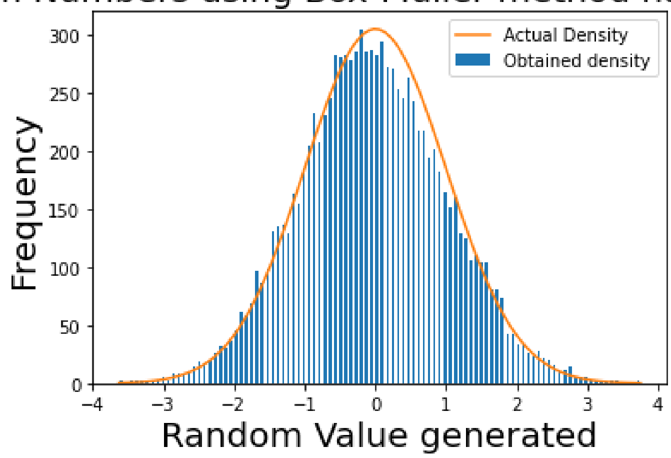
Generating Random Numbers using Box-Muller method having distribution N(0,5)



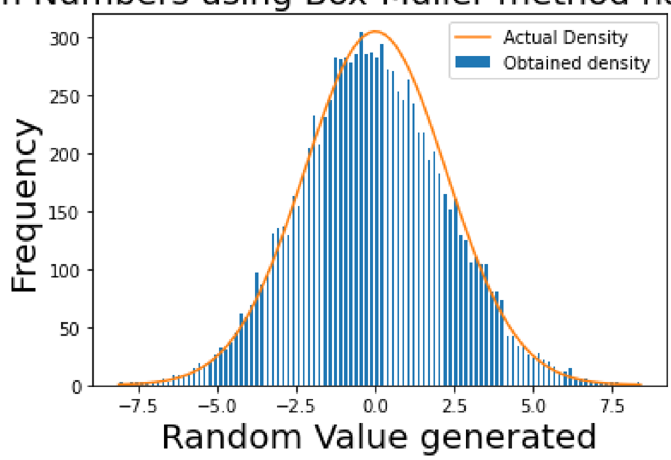
Generating Random Numbers using Box-Muller method having distribution N(5,5)



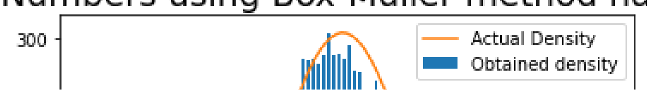
Generating Random Numbers using Box-Muller method having distribution N(0,1)

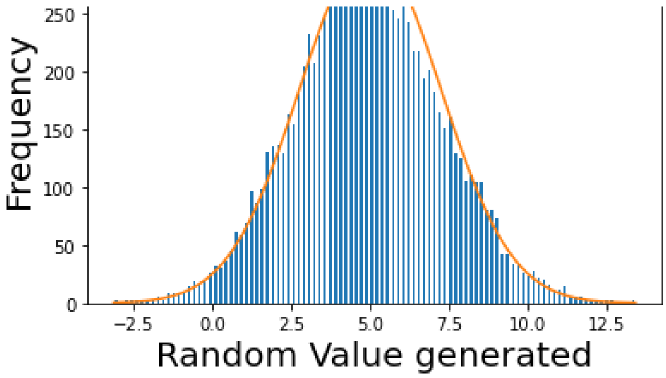


Generating Random Numbers using Box-Muller method having distribution N(0,5)



Generating Random Numbers using Box-Muller method having distribution N(5,5)






```
Mean for N=100 is  -0.08301780364112182
Mean for N=10000 is  -0.03241891940542742
Variance for N=100 is  1.1292005571213954
Variance for N=10000 is  1.0704260144375155
Time for N=100 is  0.0001995563507080078
Time for N=10000 is  0.01676344871520996
```

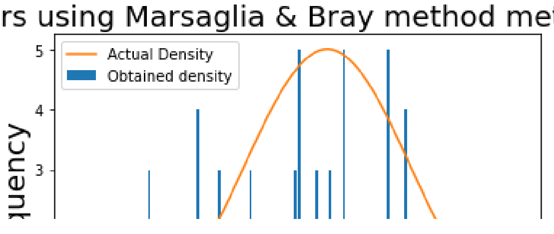
2) Marsaglia and Bray method

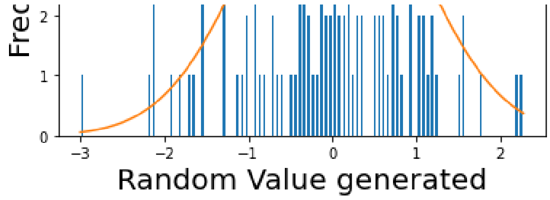
In [5]:

```
Z= []
NoOfValues= [100, 10000]
Mean= []
Variance= []
Time= []
NoOfAcceptedValues= []
NoOfRejectedValues= []

for n in NoOfValues:
  accepted= 0
  rejected= 0
  start= time.time()
  seed_u1=3
  seed_u2=5
  for i in range(0,n//2):
    while(1):
      seed_u1=LCG(1741,2731,12960,seed_u1)
      seed_u2=LCG(1741,2731,12960,seed_u2)
      U1=seed_u1/12960
      U2=seed_u2/12960
      U1= 2*U1-1
      U2= 2*U2-1
      X= pow(U1,2)+pow(U2,2)
      if X<=1:
        accepted+=1
        break
      else:
        rejected+=1
    Y= math.sqrt(-2.0*math.log(X)/X)
    Z1= U1*Y
    Z2= U2*Y
    Z.append(Z1)
    Z.append(Z2)
  # print(Z)
  NoOfAcceptedValues.append(accepted)
  NoOfRejectedValues.append(rejected)
  end= time.time()
  Time.append(end-start)

  mean= sum(Z) / len(Z)
  variance= sum((k - mean) ** 2 for k in Z) / len(Z)

  Mean.append(mean)
  Variance.append(variance)
```

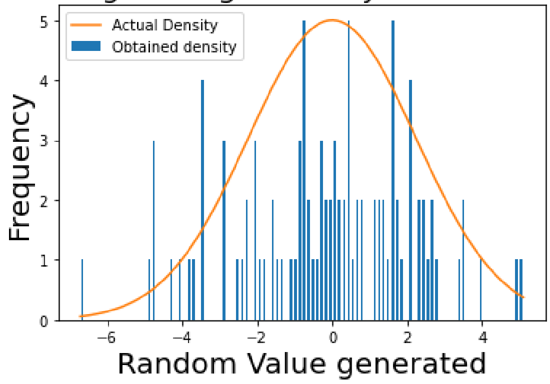
```
  x1,x2,x3= plt.hist(Z, bins = 100, rwidth = 0.5, label= "Obtained density")
  y= []
  c= (1/math.sqrt(2.00*math.pi))*pow(math.e,(-1*0*0)/2.0)
  for x in x2:
    f= (1/math.sqrt(2.00*math.pi))*pow(math.e,(-1*x*x)/2.0)
    y.append((f*max(x1))/c)
  plt.plot(x2,y, label= "Actual Density")
  plt.xlabel("Random Value generated", size=20)
  plt.ylabel("Frequency", size=20)
  plt.title("Generating Random Numbers using Marsaglia & Bray method method having distr
ibution N(0,1)", size=20)
  plt.legend()
  plt.show()


  # Now for N(0,5)
  Z1= []
  for i in Z:
    Z1.append(0+math.sqrt(5)*i)
  x1,x2,x3= plt.hist(Z1, bins = 100, rwidth = 0.5, label= "Obtained density")
  y= []
  c= (1/math.sqrt(2.00*math.pi*5))*pow(math.e,(-1*0*0)/(2.0*5.0))
  for x in x2:
    f= (1/math.sqrt(2.00*math.pi*5))*pow(math.e,(-1*x*x)/(2.0*5.0))
    y.append((f*max(x1))/c)
  plt.plot(x2,y, label= "Actual Density")
  plt.xlabel("Random Value generated", size=20)
  plt.ylabel("Frequency", size=20)
  plt.title("Generating Random Numbers using Marsaglia & Bray method method having distr
ibution N(0,5)", size=20)
  plt.legend()
  plt.show()

  # Now for N(5,5)
  Z2= []
  for i in Z:
    Z2.append(5+math.sqrt(5)*i)
  x1,x2,x3= plt.hist(Z2, bins = 100, rwidth = 0.5, label= "Obtained density")
  y= []
  c= (1/math.sqrt(2.00*math.pi*5))*pow(math.e,(-1*0*0)/(2.0*5.0))
  for x in x2:
      f= (1/math.sqrt(2.00*math.pi*5))*pow(math.e,(-1*(x-5)*(x-5))/(2.0*5.0))
      y.append((f*max(x1))/c)
  plt.plot(x2,y, label= "Actual Density")
  plt.xlabel("Random Value generated", size=20)
  plt.ylabel("Frequency", size=20)
  plt.title("Generating Random Numbers using Marsaglia & Bray method method having distr
ibution N(5,5)", size=20)
  plt.legend()
  plt.show()

print("Mean for N=100 is ", Mean[0])
print("Mean for N=10000 is ", Mean[1])
print("Variance for N=100 is ", Variance[0])
print("Variance for N=10000 is ", Variance[1])
print("Time for N=100 is ", Time[0])
print("Time for N=10000 is ", Time[1])

print("\n\n")
print("Proportion of values rejected in both cases(100 and 10,1000 values)")
for j in range(0,2):
    print(NoOfRejectedValues[j]*1.00/(NoOfRejectedValues[j]*1.00+NoOfAcceptedValues[j]*1
.00))
print("Actual value of 1-pi/4 : ",1.00-math.pi/4.00)
```

Generating Random Numbers using Marsaglia & Bray method method having distribution N(0,1)

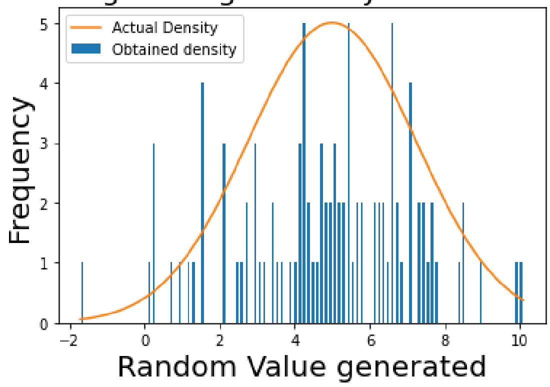




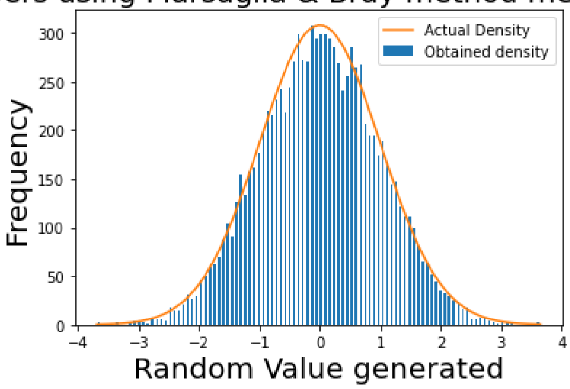
Generating Random Numbers using Marsaglia & Bray method method having distribution N(0,5)



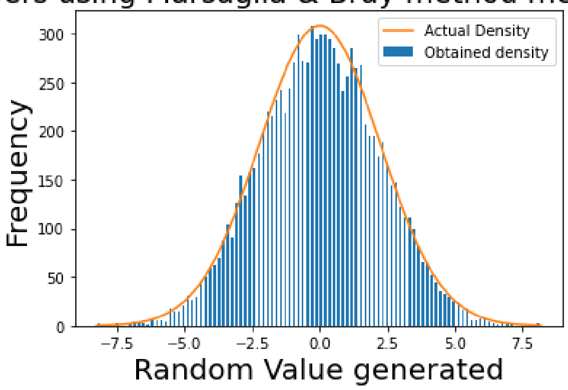
Generating Random Numbers using Marsaglia & Bray method method having distribution N(5,5)



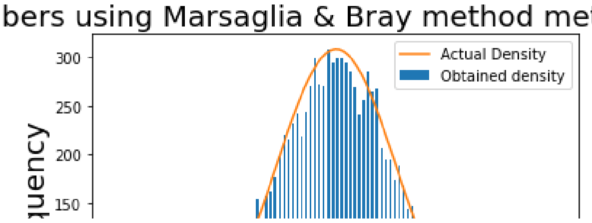
Generating Random Numbers using Marsaglia & Bray method method having distribution N(0,1)

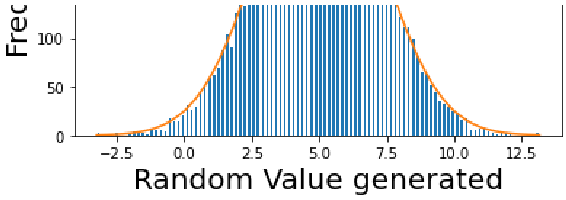


Generating Random Numbers using Marsaglia & Bray method method having distribution N(0,5)



Generating Random Numbers using Marsaglia & Bray method method having distribution N(5,5)






```
Mean for N=100 is  -0.08758640892644651
Mean for N=10000 is  0.0008888468932104934
Variance for N=100 is  1.0762504461887914
Variance for N=10000 is  0.9730110527166492
Time for N=100 is  0.00037288665771484375
Time for N=10000 is  0.019336462020874023
```

```
Proportion of values rejected in both cases(100 and 10,1000 values)
0.2537313432835821
0.22528664394174155
Actual value of 1-pi/4 :  0.21460183660255172
```

Question 1(c):

As expected, it is clear that as we generate more samples, our distribution approaces the theoritical plot. \ Changing the mean from 0 to 5 is basically just moving the bars by 5 units towards right, it would not affect the shape of the distribution.

Question2:

From the above computed time for N=100 and N=10000 for both the methods, we observe that the Box-Muller method is a bit faster than the Marsaglia & Bray method

Question 3:

For Marsaglia & Bray method, by keeping the track of proportion of values rejected, we observe : 0.2537313432835821 for n=100 \ 0.22528664394174155 for n=10000\ and the theoritical value of 1- pi/4 is 0.21460183660255172