Computational Finance_Project 1

January 16, 2019

In [ ]: Xiangui Mei

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import math as math
import time
```

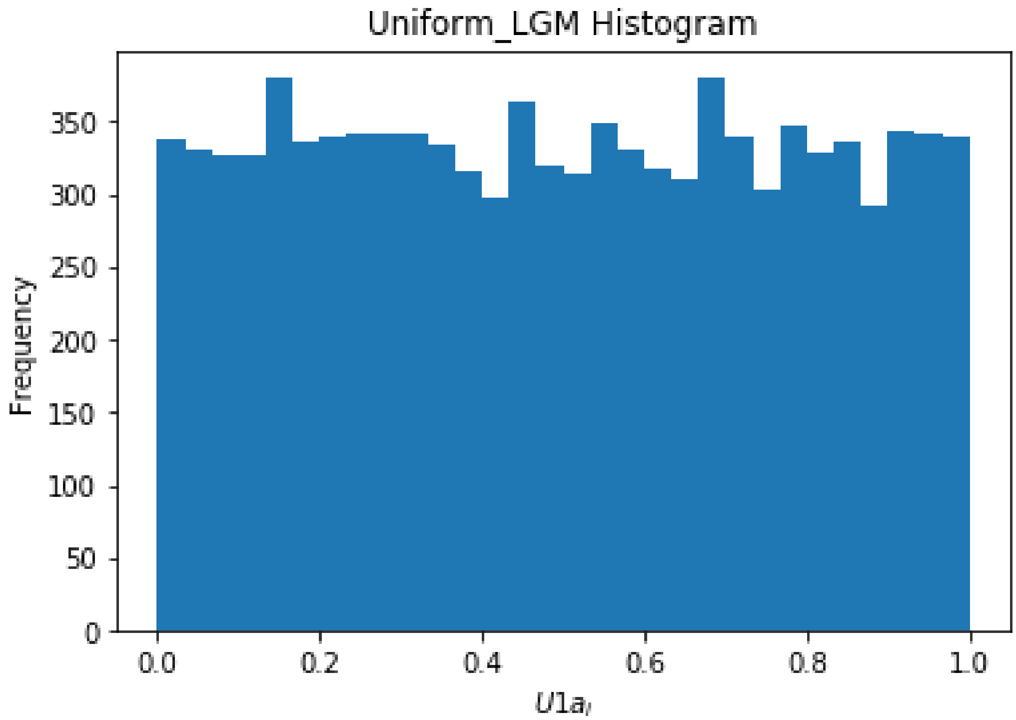
In [3]:
```
Q1 (a)  Using LGM method generate 10000 Uniformly distributed random numbers,
compute the empirical mean and the standard deviation.
```

```
  File "<ipython-input-3-93bd838cb5ea>", line 1
Q1 (a)  Using LGM method generate 10000 Uniformly distributed random numbers,
            ^
SyntaxError: invalid syntax
```

In [4]:
```
def LGM_uniform(seed, n):
        m = pow(2,31)-1
        b = 0
        a = pow(7,5)
        X = np.zeros(n)
        U = np.zeros(n)
        X[0] = seed
        for i in range(1,n):
            X[i] = np.mod(X[i-1]*a+b,m)
        for i in range(0,n):
            U[i] = X[i]/m
        return U

U1a = LGM_uniform(101,10000)
plt.hist(U1a, bins=30)
plt.title("Uniform_LGM Histogram")
plt.xlabel("$U1a_i$")
plt.ylabel("Frequency")
print("The empirical mean and std for LGM_Uniform is:")
print(round(np.mean(U1a),4),round(np.std(U1a),4))
```

```
        plt.show()
```

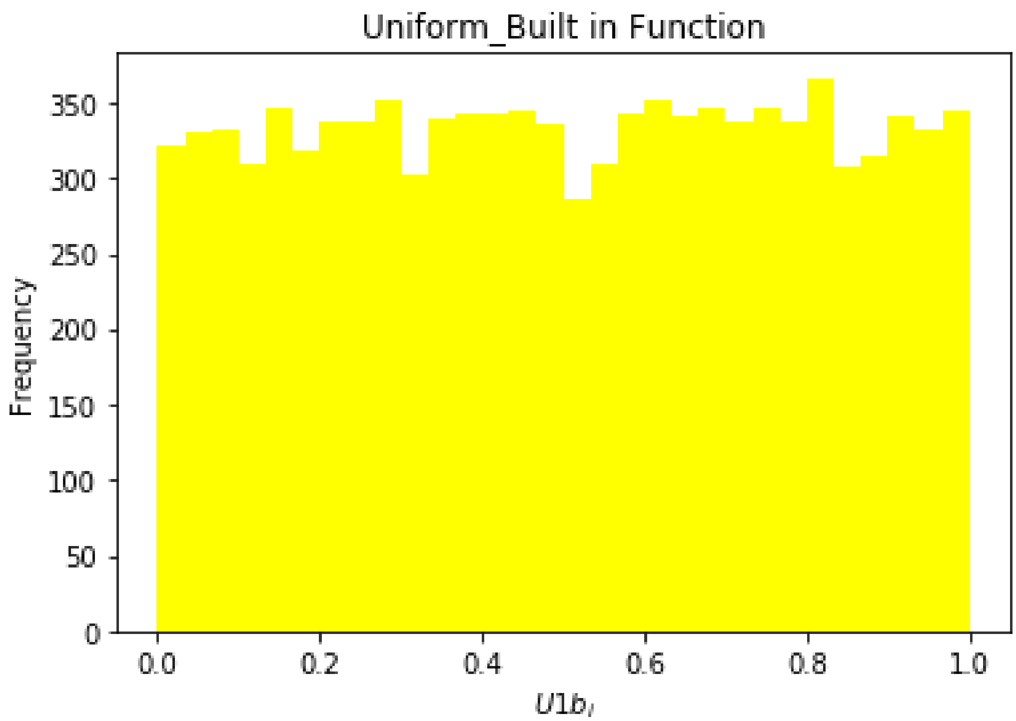
```
The empirical mean and std for LGM_Uniform is:
(0.498, 0.2894)
```




```
In [ ]: Q1 (b) Now use built-in functions of whatever software you are using
        to do the same thing as in (a).
```

```
In [3]: np.random.seed(5)
        U1b = np.random.uniform(size=10000)
        plt.hist(U1b, bins=30, color='yellow')
        plt.title("Uniform_Built in Function")
        plt.xlabel("$U1b_i$")
        plt.ylabel("Frequency")
        print("The mean and std for Builtin_Uniform is:")
        print(round(np.mean(U1b),4),round(np.std(U1b),4))

        plt.show()
```

```
Q1 (b)
('The mean and std for Builtin_Uniform is:', 0.5024, 0.2881)
```



In [ ]: Q1 (c) Compare your findings in (a) and (b) and comment (be short but precise)

In [40]:
```
print("the histogram of two look very similar.")
print("means and std are also very similar.means of two distributions are separately:")
print(round(np.mean(U1a),4),round(np.mean(U1b),4))
print("and std of two distributions are separately:")
print(round(np.std(U1a),4),round(np.std(U1b),4))
```

```
the histogram of two look very similar.
means and std are also very similar.means of two distributions are separately:
(0.498, 0.5024)
and std of two distributions are separately:
(0.2894, 0.2881)
```

In [ ]: Q2 (a) Generate 10000 random numbers with the following distribution:

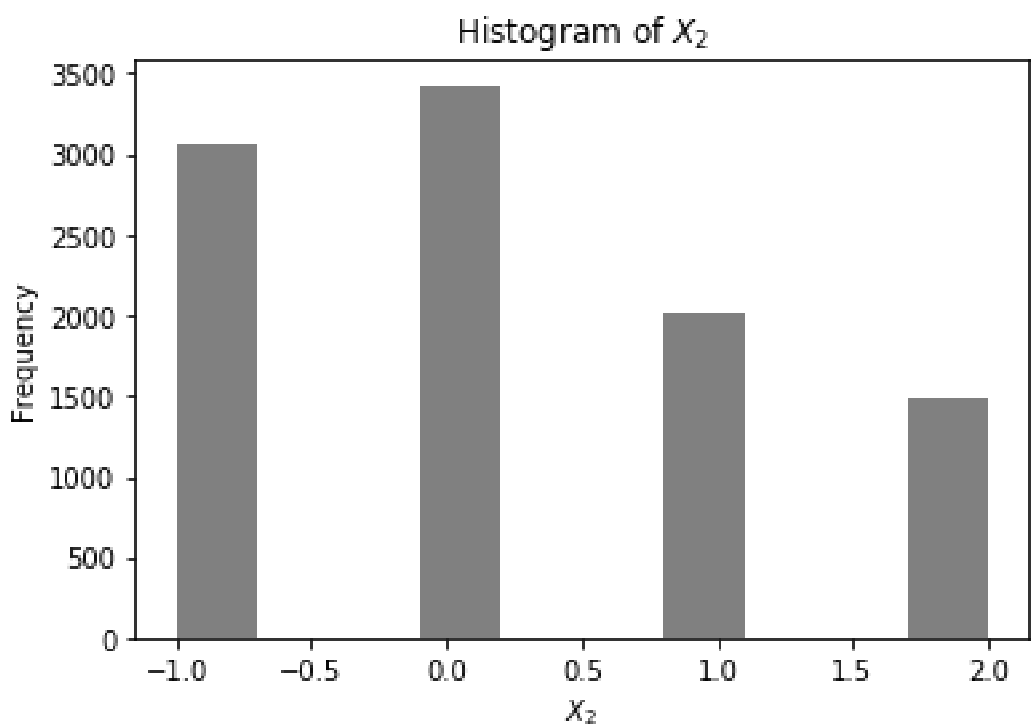
In [42]:
```
U2 = LGM_uniform(101,10000)
X2 = np.zeros(10000)
for i in range(0,10000):
    if U2[i] < 0.3:
        X2[i] = -1
    if U2[i] >= 0.3 and U2[i] < 0.65:
        X2[i] = 0
```

```
            if U2[i] >= 0.65 and U2[i] < 0.85:
                X2[i] = 1
            if U2[i] >= 0.85:
                X2[i] = 2
        print("X2 is")
        print(X2)
```

```
X2 is
[-1. -1. -1. ...  2.  0.  1.]
```

In [ ]: Q2 (b) Draw the histogram and compute the empirical mean and the standard deviation of the sequence of 10000 numbers generated above in part (a)

In [43]:
```
plt.hist(X2, color='grey')
plt.title("Histogram of $X_2$")
plt.xlabel("$X_2$")
plt.ylabel("Frequency")
plt.show()

print("The mean and standard deviation are separately:")
print(round(np.mean(X2),4),round(np.std(X2),4))
```



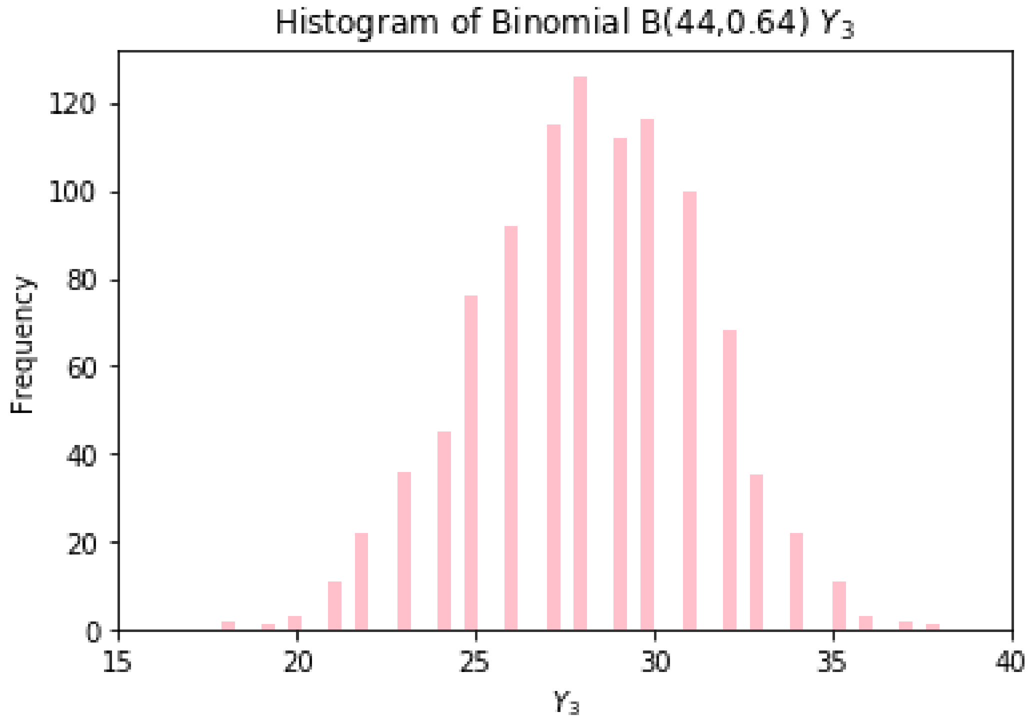
```
The mean and standard deviation are separately:
(0.195, 1.0329)
```

In [ ]: Q3 (a) Generate 1000 random numbers with Binomial distribution with n= 44 and p = 0.64.

```
In [44]: # generate 44,000 uniform (0,1)
         U3 = LGM_uniform(2,44000)

         # generate 44,000 Bernoulli random numbers
         X3 = np.zeros(44000)
         for i in range(0,44000):
             if U3[i] < 0.64:
                 X3[i] = 1
             if U3[i] >= 0.64:
                 X3[i] = 0

         # Define Binomial variable Y3[i]=sum of sublist
         split= list(range(0, 44000, 44))
         sub=[X3[i: i + 44] for i in split]
         Y3=np.zeros(1000)
         for i in range(1,1000):
             Y3[i]=sum(sub[i])
```

In [ ]: Q3 (b) Draw the histogram. Compute the probability that the random variable X that has Binomial (44, 0.64) distribution, is at least 40: P(X>=40).

```
In [46]: # plot the histogram
         plt.hist(Y3, bins=100, color='pink')
         plt.title("Histogram of Binomial B(44,0.64) $Y_3$")
         plt.xlim(15,40)
         plt.xlabel("$Y_3$")
         plt.ylabel("Frequency")
         plt.show()

         # compute the probability P(Y3>=40)
         m=1000
         m1=0
         for i in range(1,1000):
             if Y3[i]>40 or Y3[i]==40:
                 m1=m1+1
         print("The number of Y3 is at least 40 is:")
         print(m1)
         prob=m1/1000.0
         print ("The empirical probability that the random variable Y3 is at least 40 is:")
         print(prob)
```




```
The number of Y3 is at least 40 is:
0
The empirical probability that the random variable Y3 is at least 40 is:
0.0
```

In [ ]: Q4 (a) Generate 10000 Exponentially distributed random numbers with parameter lamb=1.5

In [5]:
```
#np.random.exponential(1.5,10000)
lamb=1.5
Y4=np.zeros(10000)
Y4=-(1/lamb)*np.log(U1a)
print(" Y4 is")
print(Y4)
```

```
 Y4 is
[11.24829472  4.76192756  0.83610389 ...  0.03461895  0.44600453
  0.12248846]
```

In [ ]: Q4 (b) Compute P(X>= 1) and P(X>=1).

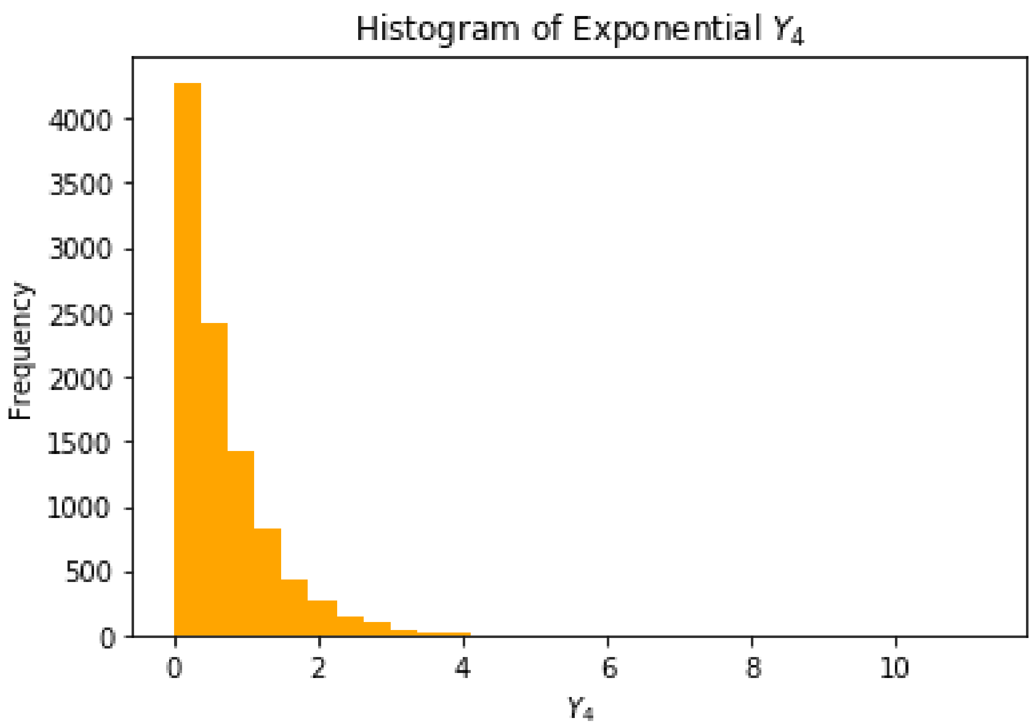
In [6]:
```
prob1=np.size(np.where(Y4 >= 1))/10000.0
prob2=np.size(np.where(Y4 >= 4))/10000.0
```

```
print ("The probability that the random variable Y4 is at least 1 is")
print(prob1)
print ("The probability that the random variable Y4 is at least 4 is")
print(prob2)
```

```
The probability that the random variable Y4 is at least 1 is
0.2276
The probability that the random variable Y4 is at least 4 is
0.0028
```

In [ ]: Q4 (c) Compute the empirical mean and the standard deviation of the sequence of 10000 numbers generated above in part (a).

In [7]:
```
# plot the histogram
plt.hist(Y4, bins=30, color='orange')
plt.title("Histogram of Exponential $Y_4$")
plt.xlabel("$Y_4$")
plt.ylabel("Frequency")
plt.show()
# caculate the mean and standard deviation
print("The mean and standard deviation of Exponential Distribution are separately")
print(round(np.mean(Y4),4),round(np.std(Y4),4))
```



```
The mean and standard deviation of Exponential Distribution are separately
(0.6726, 0.6806)
```

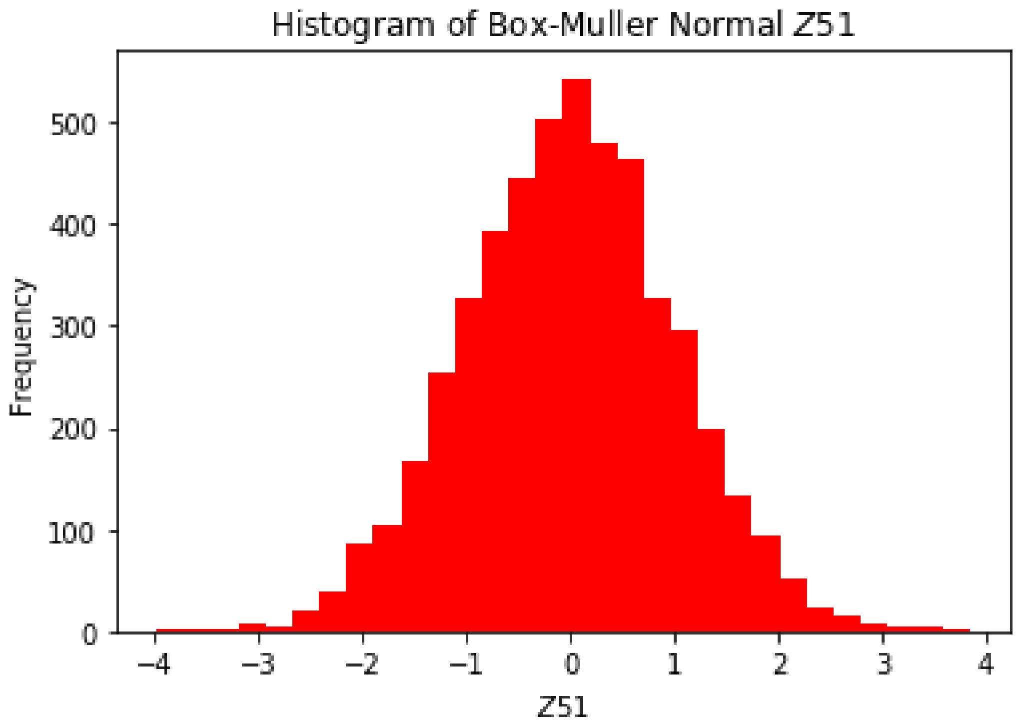
In [ ]: Q5 (a) Generate 5000 Uniformly distributed random numbers on [0,1]

In [13]:
```
# generate 5000 uniformly distributed random numbers
U51=LGM_uniform(101,5000)
print(U51)
```

```
Q5 (a)
[4.70317900e-08 7.90463295e-04 2.85316602e-01 ... 2.23372674e-01
 2.24527209e-01 6.28804527e-01]
```

In [ ]: Q5 (b) Generate 5000 Normally distributed random numbers with mean 0 and variance 1, by Box Muller Method.

In [37]:
```
# generate normal dist random numbers
U52=LGM_uniform(123,5000)
Z51=np.zeros(5000)
start_time = time.time()
for i in range(1,5000):
    Z51[i]=np.sqrt(-2*np.log(U51[i]))*math.cos(2*math.pi*U52[i])
t1=time.time() - start_time

# plot the histogram
plt.hist(Z51,bins=30,color="red")
plt.title("Histogram of Box-Muller Normal $Z51$")
plt.xlabel("$Z51$")
plt.ylabel("Frequency")
plt.show()
```



In [ ]: Q5 (c) Compute the empirical mean and the standard deviation

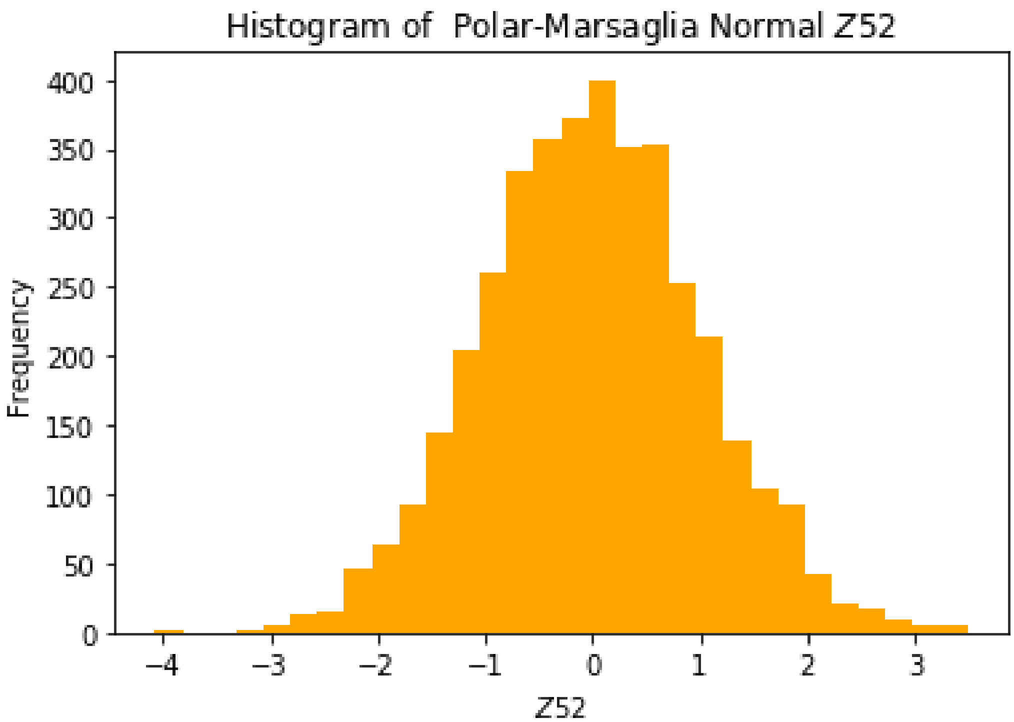
```
In [50]: # caculate the mean and variance
         print("The mean and standard deviation are separately")
         print(round(np.mean(Z51),4),round(np.std(Z51),4))
```

```
The mean and standard deviation are separately
(-0.0074, 1.0084)
```

In [ ]: Q5 (d) Now use the Polar-Marsaglia method to do the same as in (b)

```
In [38]: W=np.zeros(5000)
         V1=np.zeros(5000)
         V2=np.zeros(5000)
         Z52=np.zeros(5000)
         V1[:]=2*U51[:]-1
         V2[:]=2*U52[:]-1
         W[:]=pow(V1[:],2)+pow(V2[:],2)
         j=0
         start_time = time.time()
         for i in range(0,5000):
             if W[i]<=1:
                 Z52[j]=V1[i]*np.sqrt(-2*np.log(W[i])/(W[i]))
```

```
        j=j+1
Z52=Z52[0:j]
t2=time.time() - start_time

# plot the histogram
plt.hist(Z52, bins=30,color="orange")
plt.title("Histogram of  Polar-Marsaglia Normal $Z52$")
plt.xlabel("$Z52$")
plt.ylabel("Frequency")
plt.show()
```



In [ ]: Q5 (e) Compute the empirical mean and the standard deviation of the sequence

In [17]:
```
# caculate the mean and variance
print("The mean and standard deviation are separately")
print(round(np.mean(Z52),4),round(np.std(Z52),4))
```

```
Q5 (e)
('The mean and standard deviation are separately', -0.0027, 1.0127)
```

In [ ]: Q5 (f) Now compare the efficiencies of the two above-algorithms

In [39]:
```
print("By using python's build-in function:")
print("The effiency for Box-Muller is:", round(t1,5), "s.")
```

```
        print("The effiency for Polar-Marsaglia is:", round(t2,5), "s.")
        print("Polar-Marsaglia method is more efficient based on the result presented above.")
        print("because trigonometric function is time consuming than polynomial function.")
```

```
By using python's build-in function:
('The effiency for Box-Muller is:', 0.05451, 's.')
('The effiency for Polar-Marsaglia is:', 0.03767, 's.')
Polar-Marsaglia method is more efficient based on the result presented above.
because trigonometric function is time consuming than polynomial function.
```