

# EE511 Assignment 6

Zechen Ha

March 22, 2020

## Question 1

### Experiment

Use Box-Muller and Polar Marsaglia method respectively to get the pairs of independent normal random variables using the uniform random variables. Then multiply the result with standard deviation and shift by mean steps to get the required normal random variables. Calculate the sample mean and sample variance.

### Code

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import time
4
5
6 start_box = time.time()
7 N = 1000 #no of samples
8 M1 = 1 # Mean of X
9 M2 = 2 # Mean of Y
10 V1 = 4 # Variance of X
11 V2 = 9 # Variance of Y
12
13 u1 = np.random.rand(N,1)
14 u2 = np.random.rand(N,1)
15
16 # Generate X and Y that are N(0,1) random variables and independent
17 X = np.sqrt(-2*np.log(u1))*np.cos(2*np.pi*u2)
18 Y = np.sqrt(-2*np.log(u1))*np.sin(2*np.pi*u2)
19
20 # Scale them to a particular mean and variance
21 x = np.sqrt(V1)*X + M1; # x~ N(M1,V1)
22 y = np.sqrt(V2)*Y + M2; # y~ N(M2,V2)
23
24 end_box = time.time()
25 print(end_box - start_box)
26
27 covariance_xy = np.dot((x-M1).T, y-M2)/N
28 print(covariance_xy)
29
30 A = x+y
31 t = np.arange(-5,11,0.01)
32 ft = np.e**(-np.square(t-3)/26)/np.sqrt(26*np.pi)
33 plt.hist(A, alpha = 0.7, edgecolor = "black", normed=True)
34 plt.plot(t,ft,color="red")
35 plt.ylabel("frequency",fontsize=12)
36 plt.title("Histogram of A",fontsize=15)
37 plt.show()
38
```

```

39 A_meanBM = np.sum(A)/N
40 A_varianceBM = np.sum(np.square(A-A_meanBM))/(N-1)
41 print(A_meanBM, A_varianceBM)
42
43 start_mar = time.time()
44 X = []
45 Y = []
46 i = 0
47 while i<1000000:
48     u1 = 2*np.random.rand()-1
49     u2 = 2*np.random.rand()-1
50     s = u1*u1 + u2*u2
51     if s < 1:
52         X.append(np.sqrt(-2*np.log(s)/s)*u1)
53         Y.append(np.sqrt(-2*np.log(s)/s)*u2)
54         i = i+1
55 X = np.array(X)
56 Y = np.array(Y)
57 size = X.size
58 # Scale them to a particular mean and variance
59 x = np.sqrt(V1)*X + M1; # x~ N(M1,V1)
60 y = np.sqrt(V2)*Y + M2; # y~ N(M2,V2)
61
62 end_mar = time.time()
63 print((end_mar - start_mar)/X.size*1000)
64
65 A = x+y
66 A_meanPM = np.sum(A)/size
67 A_variancePM = np.sum(np.square(A-A_meanPM))/(size-1)
68 print(A_meanBM, A_variancePM)

```

Listing 1: Question1 code

## Result

The formula of co-variance between X and Y is:

$$\sigma_{XY} = E[(X - \mu_X)(Y - \mu_Y)]$$

In consequence, we can get the result:

**The co-variance  $\sigma_{XY}$  is equal to 0.032**

It is obvious the co-variance is very close to zero, which means that X and Y are not correlated.

In probability, we know that two independent random variables are uncorrelated.

The histogram of  $A = X + Y$  is as follows:

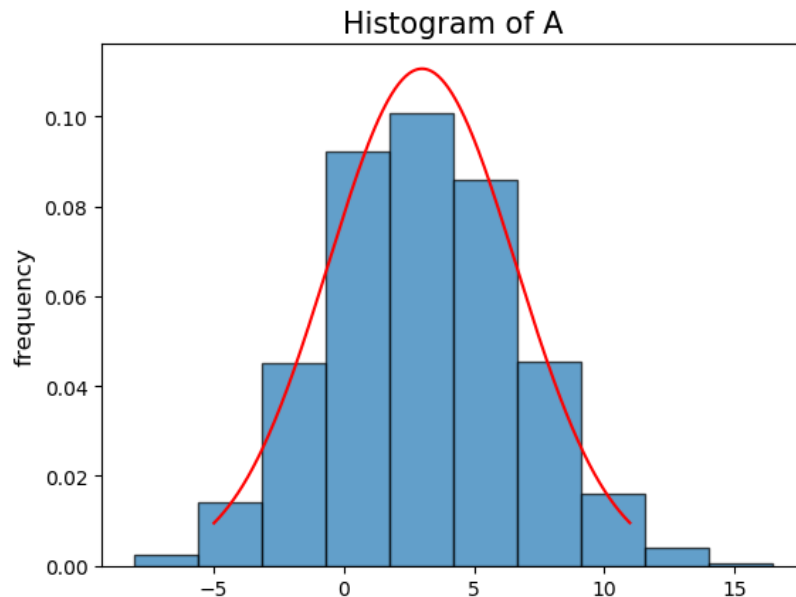


Figure 1: Histogram of A

The formula of sample mean and sample variance is:

$$\bar{X}_n = \frac{1}{n} \sum_{k=1}^n k$$

$$S_x^2 = \frac{1}{n-1} \sum_{k=1}^n (X_k - \bar{X}_n)^2$$

So we can get the result:

**The sample mean of A using Box Muller is 2.996**  
**The sample variance of A using Box Muller is 13.003**

For independent random variables  $X_1, X_2 \dots X_k \sim N(\mu_k, \sigma_k^2)$ ,

$$\sum X_k \sim N\left(\sum \mu_k, \sum \sigma_k^2\right)$$

So A follows the normal distribution of  $N(3,13)$  theoretically. The sample is close to the theoretical value.

Using the same formula to calculate the sample mean, sample variance and co-variance, the result are as follows:

**The co-variance  $\sigma_{XY}$  is equal to -0.001**  
**The sample mean of A using Polar Marsaglia is 3.071**  
**The sample variance of A using Polar Marsaglia is 12.966**

Compare the computational time required to generate 1000000 pairs of independent samples using the Polar Marsaglia method and the Box-Muller method. The results are:

**The needed time of Box Muller method is 0.097s**  
**The needed time of Polar Marsaglia method is 6.808s**

We can see that the Box Muller is significantly faster than Polar Marsaglia method.

## Question 2

### Experiment

Set the function  $g(x) = \frac{1}{6}e^{(-x/6)}$  and constant  $c = \frac{6}{5.5}$ , we can see that  $c*g(x)$  is always larger than the function  $f(x)$ . Then we can start the accept and reject process.

### Code

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 N = 100000
5 theta = 5.5
6 theta_g = 6
7 c = 6/5.5
8 result = []
9
10 for i in range(N):
11     u = np.random.rand()
12     x = np.random.exponential(theta_g)
13     f = 1/theta*np.exp(-x/theta)
14     g = 1/theta_g*np.exp(-x/theta_g)
15     if (f/(c*g))>=u:
16         result.append(x)
17
18 result = np.array(result)
19 t = np.arange(0,30,0.01)
20 pdf = 1/theta*np.exp(-t/theta)
21 print(result.size)
22
23 plt.hist(result,alpha = 0.7, edgecolor = "black",bins = 20, normed=True,
24         range=(0,30))
25 plt.plot(t,pdf,color="red")
26 plt.title("Histogram of the exp(5.5) random variable", fontsize=15)
27 plt.ylabel("Frequency", fontsize=12)
28 plt.show()
29 accept_rate = result.size/N
```

```
30 print(accept_rate, 1/c)
```

Listing 2: Question2 Code

## Result

The histogram of the generated Gamma(5.5,1) random variable is as follows:

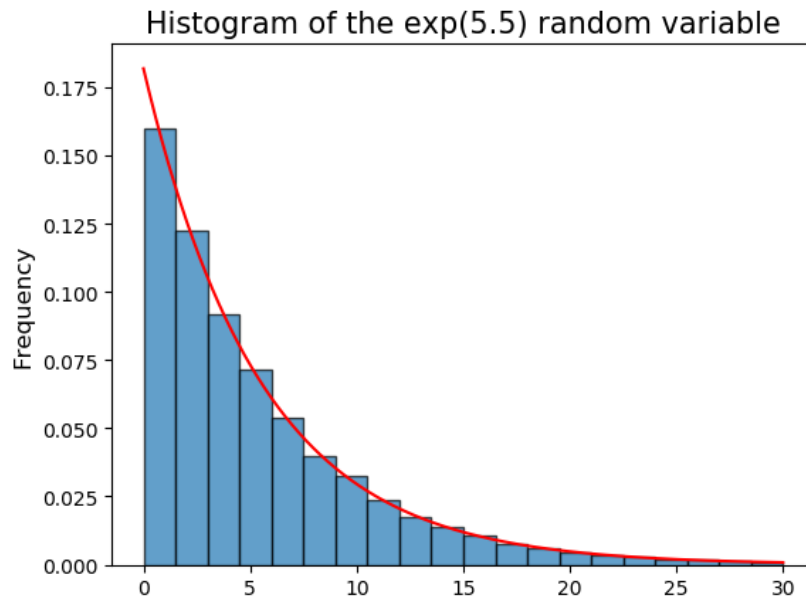


Figure 2: Histogram of Exp(5.5) random variable

The acceptance rate in the simulation is  $\frac{\#accepted}{\#total}$ , which is equal to 0.916. The theoretical acceptance rate is  $\frac{1}{c}$ , which is equal to 0.916. They are the same.

## Question3

### Experiment

Use the Chambers-Mallows-Stuck method to simulate the whole process with different parameters.

### Code

When  $\beta = 0$ , simulate with  $\alpha = 0.5, 1, 1.8, 2$

```
1 import numpy as np
2 import matplotlib.pyplot as plt
```

```
3 from scipy.stats import levy_stable
4
5 # alpha = 0.5
6 alpha = 0.5
7 Kalpha = alpha
8 beta = 0.75
9 beta2 = np.arctan(beta*np.tan(np.pi*alpha/2))*2/np.pi/Kalpha
10 gamma0 = -0.5*np.pi*beta2*Kalpha/alpha
11 N = 1000
12 result = []
13
14 for i in range(N):
15     W = np.random.exponential(1)
16     gamma = np.random.uniform(-np.pi/2,np.pi/2)
17     X = np.sin(alpha*(gamma-gamma0))/np.cos(gamma)**(1/alpha)*(np.cos(
18         gamma-alpha*(gamma-gamma0))/W)**((1-alpha)/alpha)
19     result.append(X)
20 result = np.array(result)
21
22 # x = np.arange(levy_stable.ppf(0.01,alpha,beta),levy_stable.ppf(0.99,
23     alpha,beta),0.01)
24 x = np.arange(-10,10,0.1)
25
26 fig,ax = plt.subplots(1,2)
27 ax[0].hist(result,alpha=0.7,edgecolor="black",bins=20,range=(-10,10),
28     normed=True)
29 ax[0].plot(x,levy_stable.pdf(x,alpha,beta),color="red")
30 ax[0].set_ylabel("Frequency", fontsize=12)
31 ax[0].set_title("Histogram of alpha=0.5 standard stable r.v.")
32
33 ax[1].plot(range(N),result)
34 ax[1].set_title("Time series")
35 plt.show()
36
37 # alpha = 1 beta = 0
38 alpha = 1
39 beta = 0.75
40 beta2 = beta
41 result = []
42
43 for i in range(N):
44     W = np.random.exponential(1)
45     gamma = np.random.uniform(-np.pi/2,np.pi/2)
46     X = (np.pi/2+beta2*gamma)*np.tan(gamma)-beta2*np.log(W*np.cos(gamma)/(
47         np.pi/2+beta2*gamma))
48     result.append(X)
49 result = np.array(result)
50
51 fig,ax = plt.subplots(1,2)
```

```

48 ax[0].hist(result,alpha=0.7,edgecolor="black",bins=20,range=(-10,10),
    normed=True)
49 ax[0].plot(x,levy_stable.pdf(x,alpha,beta),color="red")
50 ax[0].set_ylabel("Frequency", fontsize=12)
51 ax[0].set_title("Histogram of alpha=1 standard stable r.v.")
52
53 ax[1].plot(range(N),result)
54 ax[1].set_title("Time series")
55 plt.show()
56
57 # alpha = 1.8
58 alpha = 1.8
59 Kalpha = alpha-2
60 beta = 0.75
61 beta2 = np.arctan(beta*np.tan(np.pi*alpha/2))*2/np.pi/Kalpha
62 gamma0 = -0.5*np.pi*beta2*Kalpha/alpha
63 result = []
64
65 for i in range(N):
66     W = np.random.exponential(1)
67     gamma = np.random.uniform(-np.pi/2,np.pi/2)
68     X = np.sin(alpha*(gamma-gamma0))/np.cos(gamma)**(1/alpha)*(np.cos(
        gamma-alpha*(gamma-gamma0))/W)**((1-alpha)/alpha)
69     result.append(X)
70 result = np.array(result)
71
72 fig,ax = plt.subplots(1,2)
73 ax[0].hist(result,alpha=0.7,edgecolor="black",bins=20,range=(-10,10),
    normed=True)
74 ax[0].plot(x,levy_stable.pdf(x,alpha,beta),color="red")
75 ax[0].set_ylabel("Frequency", fontsize=12)
76 ax[0].set_title("Histogram of alpha=1.8 standard stable r.v.")
77
78 ax[1].plot(range(N),result)
79 ax[1].set_title("Time series")
80 plt.show()
81
82 # alpha = 2; beta = 0-----standard Gaussian distribution
83 alpha = 2
84 beta = 0.75
85 Kalpha = alpha-2
86 gamma0 = 0
87 result = []
88
89 for i in range(N):
90     W = np.random.exponential(1)
91     gamma = np.random.uniform(-np.pi/2,np.pi/2)
92     X = np.sin(alpha*(gamma-gamma0))/np.cos(gamma)**(1/alpha)*(np.cos(
        gamma-alpha*(gamma-gamma0))/W)**((1-alpha)/alpha)
93     result.append(X)

```



```

94 result = np.array(result)
95
96 fig, ax = plt.subplots(1, 2)
97 ax[0].hist(result, alpha=0.7, edgecolor="black", bins=20, range=(-10, 10),
98            density=True)
99 ax[0].plot(x, levy_stable.pdf(x, alpha, beta), color="red")
100 ax[0].set_ylabel("Frequency", fontsize=12)
101 ax[0].set_title("Histogram of alpha=2 standard stable r.v.")
102
103 ax[1].plot(range(N), result)
104 ax[1].set_title("Time series")
105 plt.show()

```

When  $\beta = 0.75$ , simulate with  $\alpha = 0.5, 1, 1.8, 2$

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from scipy.stats import levy_stable
4
5 # alpha = 0.5
6 alpha = 0.5
7 Kalpha = alpha
8 beta = 0.75
9 beta2 = np.arctan(beta*np.tan(np.pi*alpha/2))*2/np.pi/Kalpha
10 gamma0 = -0.5*np.pi*beta2*Kalpha/alpha
11 N = 1000
12 result = []
13
14 for i in range(N):
15     W = np.random.exponential(1)
16     gamma = np.random.uniform(-np.pi/2, np.pi/2)
17     X = np.sin(alpha*(gamma-gamma0))/np.cos(gamma)**(1/alpha)*(np.cos(
18         gamma-alpha*(gamma-gamma0))/W)**((1-alpha)/alpha)
19     result.append(X)
20 result = np.array(result)
21
22 # x = np.arange(levy_stable.ppf(0.01, alpha, beta), levy_stable.ppf(0.99,
23     alpha, beta), 0.01)
24 x = np.arange(-10, 10, 0.1)
25
26 fig, ax = plt.subplots(1, 2)
27 ax[0].hist(result, alpha=0.7, edgecolor="black", bins=20, range=(-10, 10),
28            normed=True)
29 ax[0].plot(x, levy_stable.pdf(x, alpha, beta), color="red")
30 ax[0].set_ylabel("Frequency", fontsize=12)
31 ax[0].set_title("Histogram of alpha=0.5 standard stable r.v.")
32
33 ax[1].plot(range(N), result)
34 ax[1].set_title("Time series")
35 plt.show()
36

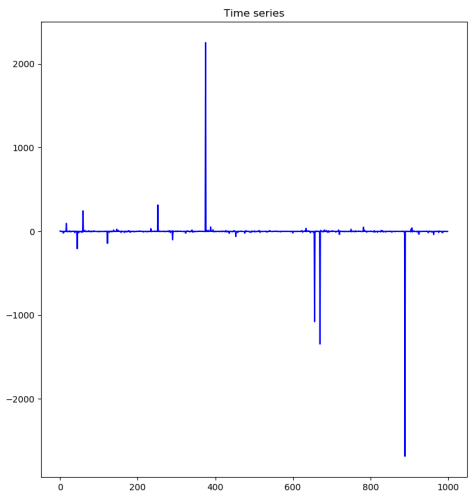
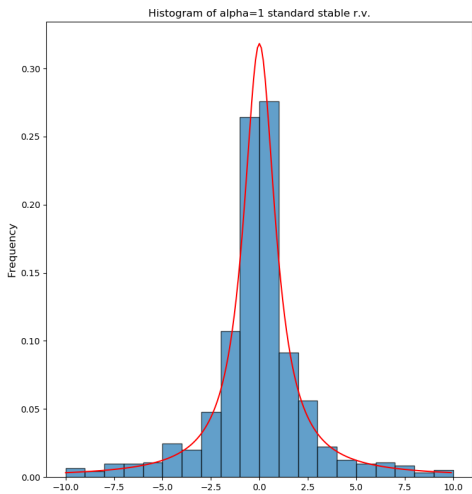
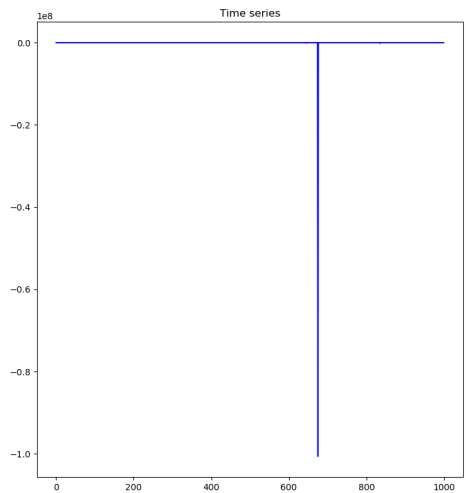
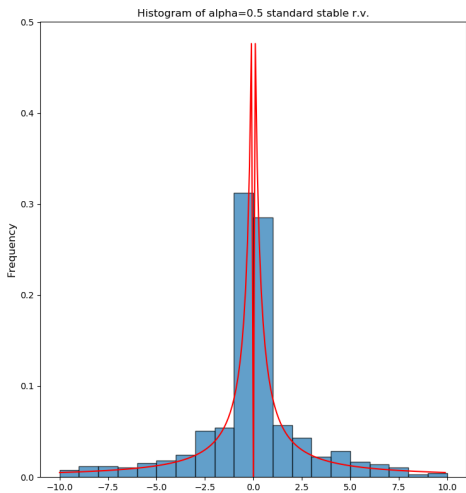
```

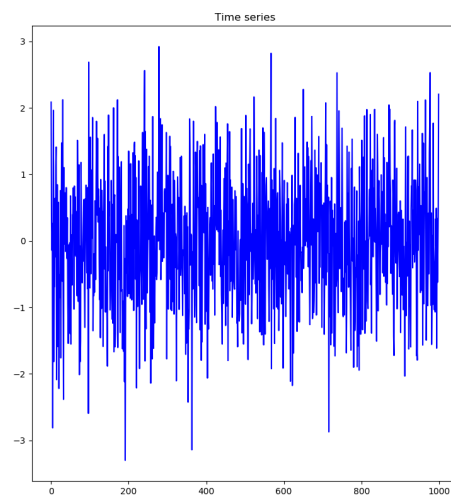
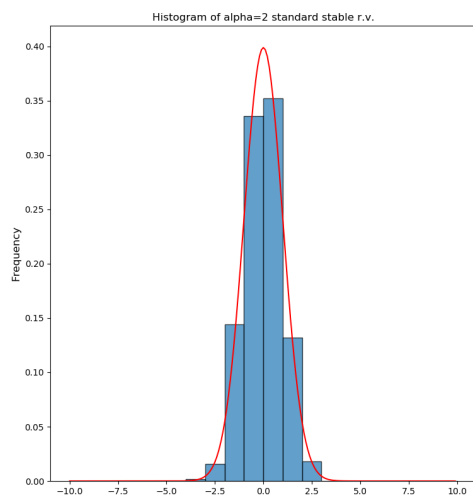
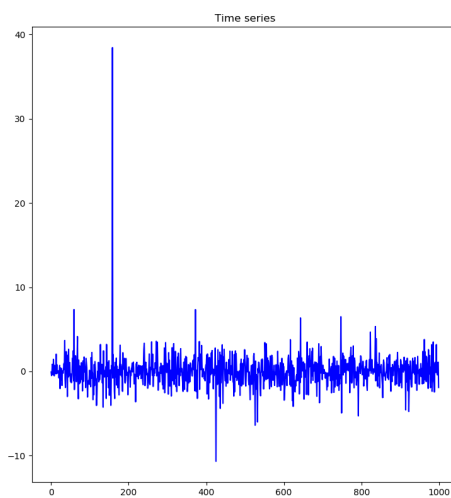
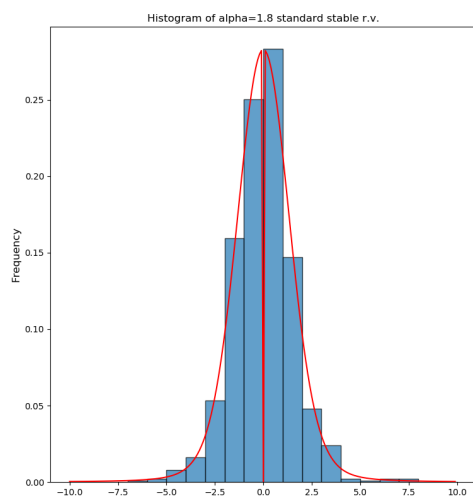
```
34 # alpha = 1 beta = 0
35 alpha = 1
36 beta = 0.75
37 beta2 = beta
38 result = []
39
40 for i in range(N):
41     W = np.random.exponential(1)
42     gamma = np.random.uniform(-np.pi/2, np.pi/2)
43     X = (np.pi/2 + beta2*gamma)*np.tan(gamma) - beta2*np.log(W*np.cos(gamma)/(
44         np.pi/2 + beta2*gamma))
45     result.append(X)
46 result = np.array(result)
47
48 fig, ax = plt.subplots(1, 2)
49 ax[0].hist(result, alpha=0.7, edgecolor="black", bins=20, range=(-10, 10),
50     normed=True)
51 ax[0].plot(x, levy_stable.pdf(x, alpha, beta), color="red")
52 ax[0].set_ylabel("Frequency", fontsize=12)
53 ax[0].set_title("Histogram of alpha=1 standard stable r.v.")
54
55 ax[1].plot(range(N), result)
56 ax[1].set_title("Time series")
57 plt.show()
58
59 # alpha = 1.8
60 alpha = 1.8
61 Kalpha = alpha - 2
62 beta = 0.75
63 beta2 = np.arctan(beta*np.tan(np.pi*alpha/2))*2/np.pi/Kalpha
64 gamma0 = -0.5*np.pi*beta2*Kalpha/alpha
65 result = []
66
67 for i in range(N):
68     W = np.random.exponential(1)
69     gamma = np.random.uniform(-np.pi/2, np.pi/2)
70     X = np.sin(alpha*(gamma - gamma0))/np.cos(gamma)**(1/alpha)*(np.cos(
71         gamma - alpha*(gamma - gamma0))/W)**((1 - alpha)/alpha)
72     result.append(X)
73 result = np.array(result)
74
75 fig, ax = plt.subplots(1, 2)
76 ax[0].hist(result, alpha=0.7, edgecolor="black", bins=20, range=(-10, 10),
77     normed=True)
78 ax[0].plot(x, levy_stable.pdf(x, alpha, beta), color="red")
79 ax[0].set_ylabel("Frequency", fontsize=12)
80 ax[0].set_title("Histogram of alpha=1.8 standard stable r.v.")
81
82 ax[1].plot(range(N), result)
83 ax[1].set_title("Time series")
```

```
80 plt.show()
81
82 # alpha = 2; beta = 0-----standard Gaussian distribution
83 alpha = 2
84 beta = 0.75
85 Kalpha = alpha-2
86 gamma0 = 0
87 result = []
88
89 for i in range(N):
90     W = np.random.exponential(1)
91     gamma = np.random.uniform(-np.pi/2,np.pi/2)
92     X = np.sin(alpha*(gamma-gamma0))/np.cos(gamma)**(1/alpha)*(np.cos(
        gamma-alpha*(gamma-gamma0))/W)**((1-alpha)/alpha)
93     result.append(X)
94 result = np.array(result)
95
96 fig,ax = plt.subplots(1,2)
97 ax[0].hist(result,alpha=0.7,edgecolor="black",bins=20,range=(-10,10),
        density=True)
98 ax[0].plot(x,levy_stable.pdf(x,alpha,beta),color="red")
99 ax[0].set_ylabel("Frequency", fontsize=12)
100 ax[0].set_title("Histogram of alpha=2 standard stable r.v.")
101
102 ax[1].plot(range(N),result)
103 ax[1].set_title("Time series")
104 plt.show()
```

## Result

When  $\beta = 0$ , the graph is symmetric. Simulate with  $\alpha = 0.5, 1, 1.8, 2$





When  $\beta = 0.75$ , simulate with  $\alpha = 0.5, 1, 1.8, 2$

