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

```
import numpy as np
import matplotlib.pyplot as plt
import time


start_box = time.time()
N = 1000 #no of samples
M1 = 1 # Mean of X
M2 = 2 # Mean of Y
V1 = 4 # Variance of X
V2 = 9 # Variance of Y

u1 = np.random.rand(N,1)
u2 = np.random.rand(N,1)

#  Generate X and Y that are N(0,1) random variables and independent
X = np.sqrt(-2*np.log(u1))*np.cos(2*np.pi*u2)
Y = np.sqrt(-2*np.log(u1))*np.sin(2*np.pi*u2)

# Scale them to a particular mean and variance
x = np.sqrt(V1)*X + M1; # x~ N(M1,V1)
y = np.sqrt(V2)*Y + M2; # y~ N(M2,V2)

end_box = time.time()
print(end_box - start_box)

covariance_xy = np.dot((x-M1).T, y-M2)/N
print(covariance_xy)

A = x+y
t = np.arange(-5,11,0.01)
ft = np.e**(-np.square(t-3)/26)/np.sqrt(26*np.pi)
plt.hist(A, alpha = 0.7, edgecolor = "black", normed=True)
plt.plot(t,ft,color="red")
plt.ylabel("frequency",fontsize=12)
plt.title("Histogram of A",fontsize=15)
plt.show()

```

```
A_meanBM = np.sum(A)/N
A_varianceBM = np.sum(np.square(A-A_meanBM))/(N-1)
print(A_meanBM, A_varianceBM)

start_mar = time.time()
X = []
Y = []
i = 0
while i<1000000:
    u1 = 2*np.random.rand()-1
    u2 = 2*np.random.rand()-1
    s = u1*u1 + u2*u2
    if s < 1:
        X.append(np.sqrt(-2*np.log(s)/s)*u1)
        Y.append(np.sqrt(-2*np.log(s)/s)*u2)
        i = i+1
X = np.array(X)
Y = np.array(Y)
size = X.size
# Scale them to a particular mean and variance
x = np.sqrt(V1)*X + M1; # x~ N(M1,V1)
y = np.sqrt(V2)*Y + M2; # y~ N(M2,V2)

end_mar = time.time()
print((end_mar - start_mar)/X.size*1000)

A = x+y
A_meanPM = np.sum(A)/size
A_variancePM = np.sum(np.square(A-A_meanPM))/(size-1)
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 σ_{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:

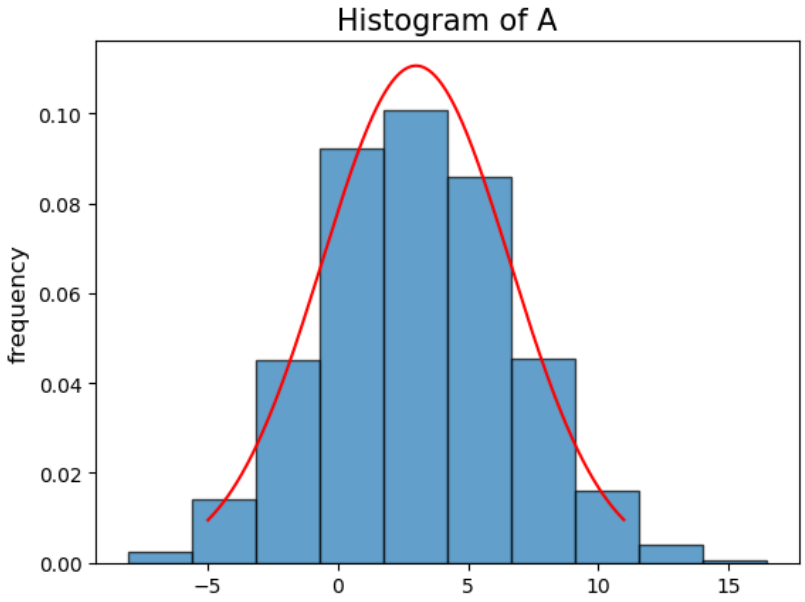


Figure 1: Histogram of A

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 $X_1, X_2..X_k$ $N(\mu_k, \sigma_k^2)$,

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

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 meansample variance and co-variance, the result are as follows:

The co-variance σ_{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

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

N = 100000
theta = 5.5
theta_g = 6
c = 6/5.5
result = []

for i in range(N):
    u = np.random.rand()
    x = np.random.exponential(theta_g)
    f = 1/theta*np.exp(-x/theta)
    g = 1/theta_g*np.exp(-x/theta_g)
    if (f/(c*g))>=u:
        result.append(x)

result = np.array(result)
t = np.arange(0,30,0.01)
pdf = 1/theta*np.exp(-t/theta)
print(result.size)

plt.hist(result,alpha = 0.7, edgecolor = "black",bins = 20, normed=True,
   range=(0,30))
plt.plot(t,pdf,color="red")
plt.title("Histogram of the exp(5.5) random variable", fontsize=15)
plt.ylabel("Frequency", fontsize=12)
plt.show()

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:

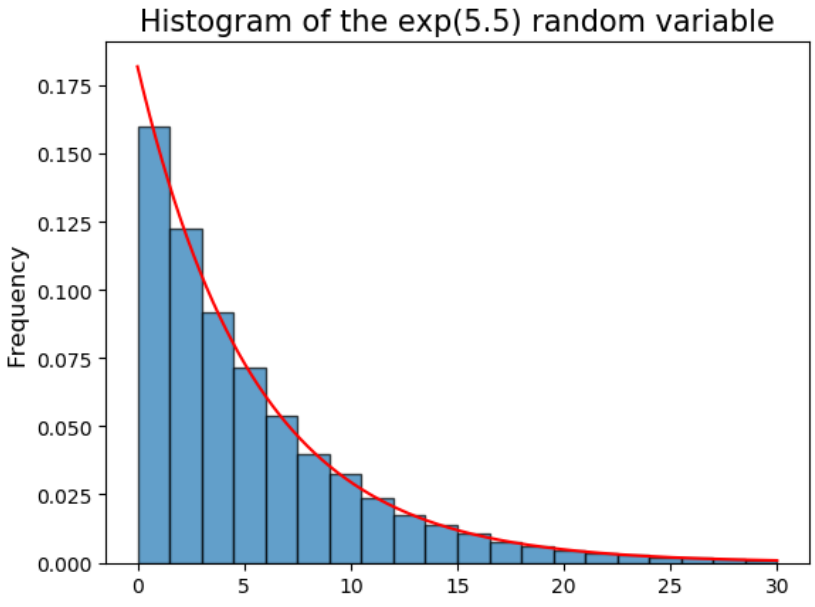


Figure 2: Histogram of Exp(5.5) random variable

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
```

```
from scipy.stats import levy_stable

# alpha = 0.5
alpha = 0.5
Kalpha = alpha
beta = 0.75
beta2 = np.arctan(beta*np.tan(np.pi*alpha/2))*2/np.pi/Kalpha
gamma0 = -0.5*np.pi*beta2*Kalpha/alpha
N = 1000
result = []

for i in range(N):
    W = np.random.exponential(1)
    gamma = np.random.uniform(-np.pi/2,np.pi/2)
    X = np.sin(alpha*(gamma-gamma0))/np.cos(gamma)**(1/alpha)*(np.cos(
    gamma-alpha*(gamma-gamma0))/W)**((1-alpha)/alpha)
    result.append(X)
result = np.array(result)

# x = np.arange(levy_stable.ppf(0.01,alpha,beta),levy_stable.ppf(0.99,
    alpha,beta),0.01)
x = np.arange(-10,10,0.1)

fig,ax = plt.subplots(1,2)
ax[0].hist(result,alpha=0.7,edgecolor="black",bins=20,range=(-10,10),
    normed=True)
ax[0].plot(x,levy_stable.pdf(x,alpha,beta),color="red")
ax[0].set_ylabel("Frequency", fontsize=12)
ax[0].set_title("Histogram of alpha=0.5 standard stable r.v.")

ax[1].plot(range(N),result)
ax[1].set_title("Time series")
plt.show()

# alpha = 1 beta = 0
alpha = 1
beta = 0.75
beta2 = beta
result = []

for i in range(N):
    W = np.random.exponential(1)
    gamma = np.random.uniform(-np.pi/2,np.pi/2)
    X = (np.pi/2+beta2*gamma)*np.tan(gamma)-beta2*np.log(W*np.cos(gamma)/(
    np.pi/2+beta2*gamma))
    result.append(X)
result = np.array(result)

fig,ax = plt.subplots(1,2)
```

```
ax[0].hist(result,alpha=0.7,edgecolor="black",bins=20,range=(-10,10),
    normed=True)
ax[0].plot(x,levy_stable.pdf(x,alpha,beta),color="red")
ax[0].set_ylabel("Frequency", fontsize=12)
ax[0].set_title("Histogram of alpha=1 standard stable r.v.")

ax[1].plot(range(N),result)
ax[1].set_title("Time series")
plt.show()

# alpha = 1.8
alpha = 1.8
Kalpha = alpha-2
beta = 0.75
beta2 = np.arctan(beta*np.tan(np.pi*alpha/2))*2/np.pi/Kalpha
gamma0 = -0.5*np.pi*beta2*Kalpha/alpha
result = []

for i in range(N):
    W = np.random.exponential(1)
    gamma = np.random.uniform(-np.pi/2,np.pi/2)
    X = np.sin(alpha*(gamma-gamma0))/np.cos(gamma)**(1/alpha)*(np.cos(
    gamma-alpha*(gamma-gamma0))/W)**((1-alpha)/alpha)
    result.append(X)
result = np.array(result)

fig,ax = plt.subplots(1,2)
ax[0].hist(result,alpha=0.7,edgecolor="black",bins=20,range=(-10,10),
    normed=True)
ax[0].plot(x,levy_stable.pdf(x,alpha,beta),color="red")
ax[0].set_ylabel("Frequency", fontsize=12)
ax[0].set_title("Histogram of alpha=1.8 standard stable r.v.")

ax[1].plot(range(N),result)
ax[1].set_title("Time series")
plt.show()

# alpha = 2; beta = 0----------standard Gaussian distribution
alpha = 2
beta = 0.75
Kalpha = alpha-2
gamma0 = 0
result = []

for i in range(N):
    W = np.random.exponential(1)
    gamma = np.random.uniform(-np.pi/2,np.pi/2)
    X = np.sin(alpha*(gamma-gamma0))/np.cos(gamma)**(1/alpha)*(np.cos(
    gamma-alpha*(gamma-gamma0))/W)**((1-alpha)/alpha)
    result.append(X)
```

```
result = np.array(result)

fig,ax = plt.subplots(1,2)
ax[0].hist(result,alpha=0.7,edgecolor="black",bins=20,range=(-10,10),
    density=True)
ax[0].plot(x,levy_stable.pdf(x,alpha,beta),color="red")
ax[0].set_ylabel("Frequency", fontsize=12)
ax[0].set_title("Histogram of alpha=2 standard stable r.v.")

ax[1].plot(range(N),result)
ax[1].set_title("Time series")
plt.show()
```

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

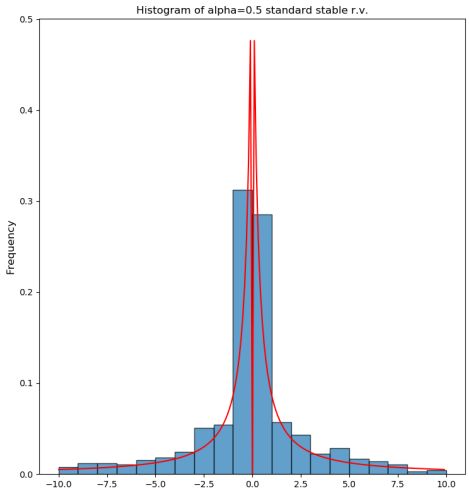
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import levy_stable

# alpha = 0.5
alpha = 0.5
Kalpha = alpha
beta = 0.75
beta2 = np.arctan(beta*np.tan(np.pi*alpha/2))*2/np.pi/Kalpha
gamma0 = -0.5*np.pi*beta2*Kalpha/alpha
N = 1000
result = []

for i in range(N):
    W = np.random.exponential(1)
    gamma = np.random.uniform(-np.pi/2,np.pi/2)
    X = np.sin(alpha*(gamma-gamma0))/np.cos(gamma)**(1/alpha)*(np.cos(
    gamma-alpha*(gamma-gamma0))/W)**((1-alpha)/alpha)
    result.append(X)
result = np.array(result)

# x = np.arange(levy_stable.ppf(0.01,alpha,beta),levy_stable.ppf(0.99,
    alpha,beta),0.01)
x = np.arange(-10,10,0.1)

fig,ax = plt.subplots(1,2)
ax[0].hist(result,alpha=0.7,edgecolor="black",bins=20,range=(-10,10),
    normed=True)
ax[0].plot(x,levy_stable.pdf(x,alpha,beta),color="red")
ax[0].set_ylabel("Frequency", fontsize=12)
ax[0].set_title("Histogram of alpha=0.5 standard stable r.v.")

ax[1].plot(range(N),result)
ax[1].set_title("Time series")
plt.show()

```

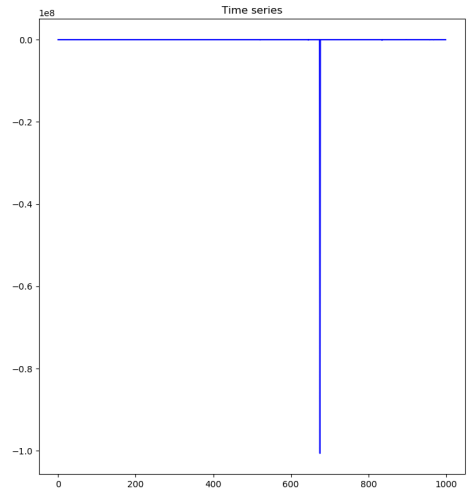
```
# alpha = 1 beta = 0
alpha = 1
beta = 0.75
beta2 = beta
result = []

for i in range(N):
    W = np.random.exponential(1)
    gamma = np.random.uniform(-np.pi/2,np.pi/2)
    X = (np.pi/2+beta2*gamma)*np.tan(gamma)-beta2*np.log(W*np.cos(gamma)/(np.pi/2+beta2*gamma))
    result.append(X)
result = np.array(result)

fig,ax = plt.subplots(1,2)
ax[0].hist(result,alpha=0.7,edgecolor="black",bins=20,range=(-10,10),normed=True)
ax[0].plot(x,levy_stable.pdf(x,alpha,beta),color="red")
ax[0].set_ylabel("Frequency", fontsize=12)
ax[0].set_title("Histogram of alpha=1 standard stable r.v.")

ax[1].plot(range(N),result)
ax[1].set_title("Time series")
plt.show()

# alpha = 1.8
alpha = 1.8
Kalpha = alpha-2
beta = 0.75
beta2 = np.arctan(beta*np.tan(np.pi*alpha/2))*2/np.pi/Kalpha
gamma0 = -0.5*np.pi*beta2*Kalpha/alpha
result = []

for i in range(N):
    W = np.random.exponential(1)
    gamma = np.random.uniform(-np.pi/2,np.pi/2)
    X = np.sin(alpha*(gamma-gamma0))/np.cos(gamma)**(1/alpha)*(np.cos(gamma-alpha*(gamma-gamma0))/W)**((1-alpha)/alpha)
    result.append(X)
result = np.array(result)

fig,ax = plt.subplots(1,2)
ax[0].hist(result,alpha=0.7,edgecolor="black",bins=20,range=(-10,10),normed=True)
ax[0].plot(x,levy_stable.pdf(x,alpha,beta),color="red")
ax[0].set_ylabel("Frequency", fontsize=12)
ax[0].set_title("Histogram of alpha=1.8 standard stable r.v.")

ax[1].plot(range(N),result)
ax[1].set_title("Time series")
```

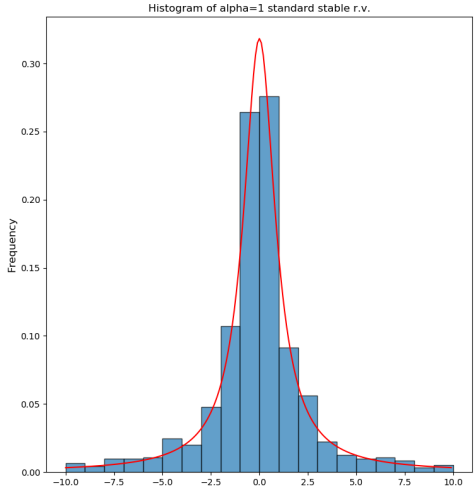
```
plt.show()

# alpha = 2; beta = 0----------standard Gaussian distribution
alpha = 2
beta = 0.75
Kalpha = alpha-2
gamma0 = 0
result = []

for i in range(N):
    W = np.random.exponential(1)
    gamma = np.random.uniform(-np.pi/2,np.pi/2)
    X = np.sin(alpha*(gamma-gamma0))/np.cos(gamma)**(1/alpha)*(np.cos(
   gamma-alpha*(gamma-gamma0))/W)**((1-alpha)/alpha)
    result.append(X)
result = np.array(result)

fig,ax = plt.subplots(1,2)
ax[0].hist(result,alpha=0.7,edgecolor="black",bins=20,range=(-10,10),
   density=True)
ax[0].plot(x,levy_stable.pdf(x,alpha,beta),color="red")
ax[0].set_ylabel("Frequency", fontsize=12)
ax[0].set_title("Histogram of alpha=2 standard stable r.v.")

ax[1].plot(range(N),result)
ax[1].set_title("Time series")
plt.show()
```

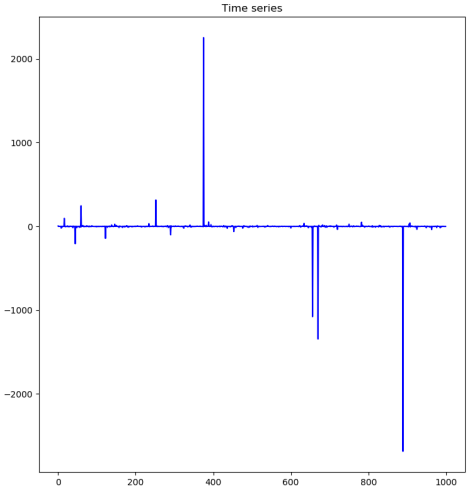
Result

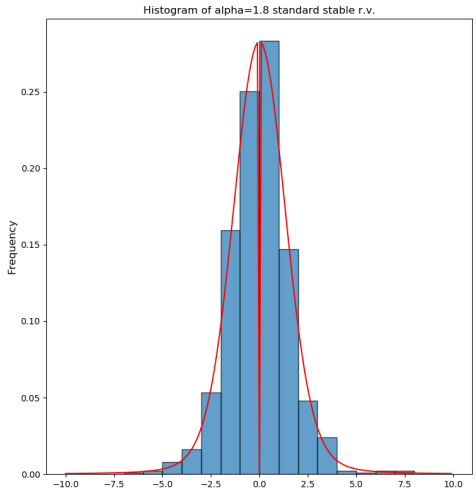
When $\beta = 0$, the graph is symmetric. Simulate with $\alpha = 0.5, 1, 1.8, 2$

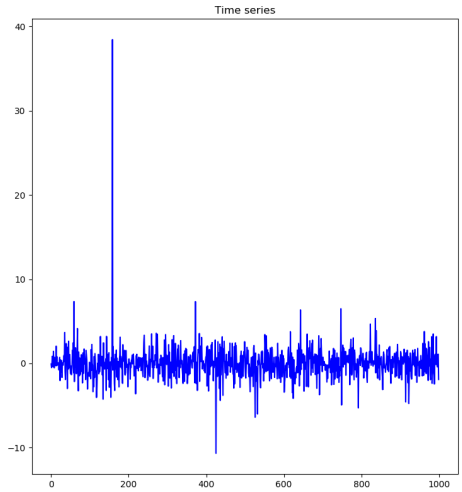

Histogram of alpha=0.5 standard stable r.v.
Frequency
0.0
0.1
0.2
0.3
0.4
0.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

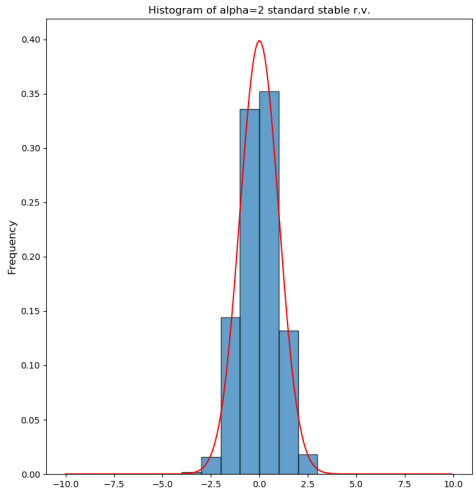

Time series
1e8
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0
200
400
600
800
1000

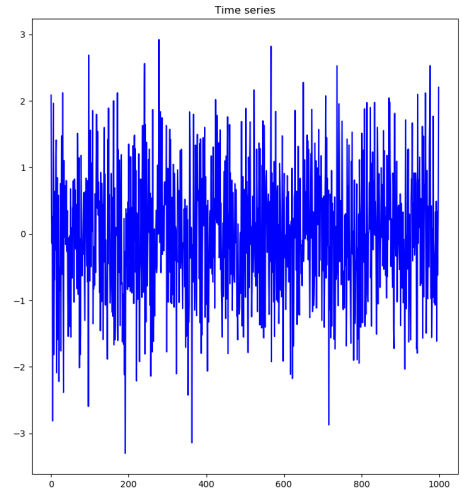

Histogram of alpha=1 standard stable r.v.
Frequency
0.00
0.05
0.10
0.15
0.20
0.25
0.30
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Time series
2000
1000
0
−1000
−2000
0
200
400
600
800
1000

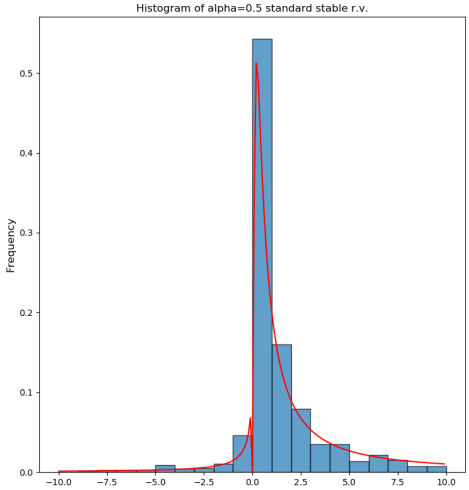


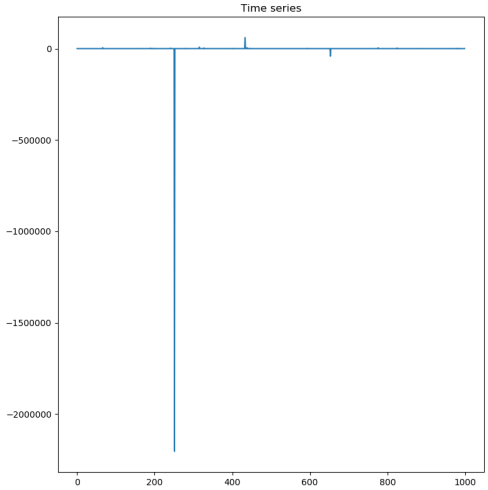


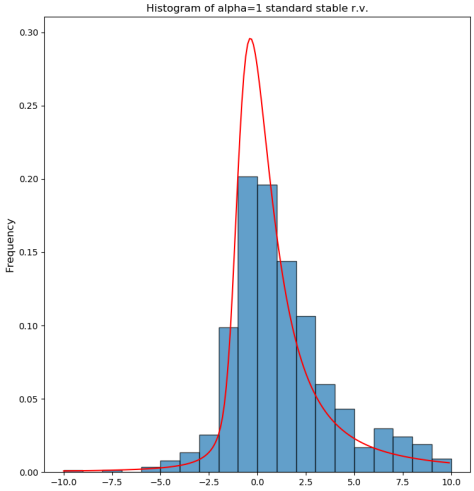


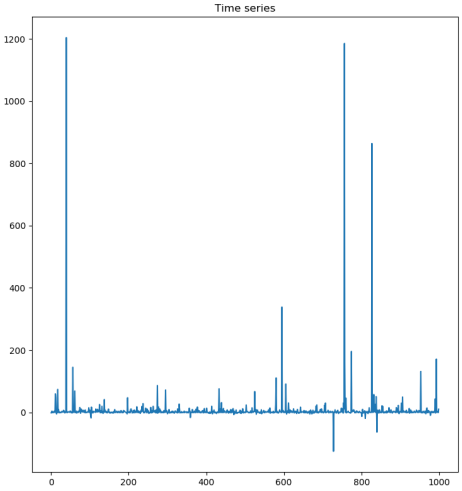


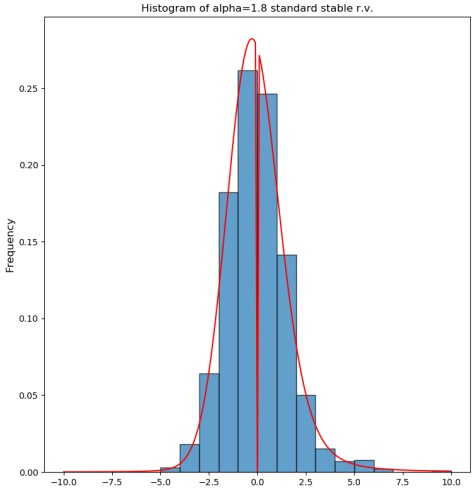
When $\beta = 0.75$, simulate with $\alpha = 0.5, 1, 1.8, 2$

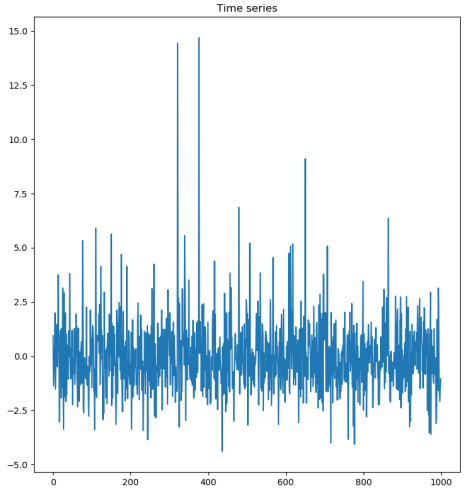

Histogram of alpha=0.5 standard stable r.v.
Frequency
0.0
0.1
0.2
0.3
0.4
0.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

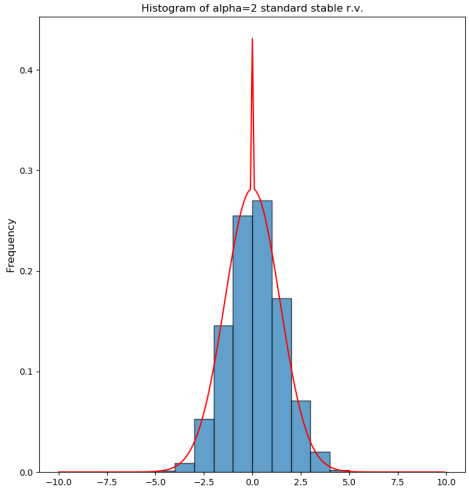

Time series
0
−500000
−1000000
−1500000
−2000000
0
200
400
600
800
1000

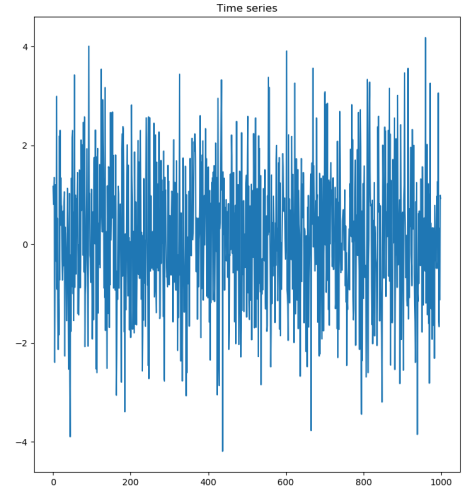

Histogram of alpha=1 standard stable r.v.
Frequency
0.00
0.05
0.10
0.15
0.20
0.25
0.30
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Time series
0
200
400
600
800
1000
1200
0
200
400
600
800
1000


Histogram of alpha=1.8 standard stable r.v.
Frequency
0.25
0.20
0.15
0.10
0.05
0.00
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Time series
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
0
200
400
600
800
1000


Histogram of alpha=2 standard stable r.v.
Frequency
0.4
0.3
0.2
0.1
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Time series
4
2
0
−2
−4
0
200
400
600
800
1000