Imperial College London

COURSEWORK 3

IMPERIAL COLLEGE LONDON

DEPARTMENT OF MATHEMATICS

Stochastic Simulation

Author: 01844579

Date: December 13, 2022

1 Question 1

Figure 1 shows the pretty plot of our Markov chain.

2 State Space Models

2.1 Gaussian Time Series With Noise

Figure 2 shows a simulation of our model. The following model is a AR(1) (Markov) process which can be used to model numerous situations. Some examples of what it can model could be wind velocity, asset price or disease processes.

2.2 Stochastic Volatility Model

We develop a volatility model by defining our Markov transition kernel and likelihood by following a modified version of a model described by Kim, Shephard, and Chib (1998)[1]. Define our model as

$$x_t|x_{t-1} \sim N(\mu + \phi(x_{t-1} - \mu), \sigma^2) \quad (1)$$

$$y_t|x_t \sim N(0, exp(x_t/2)) \quad (2)$$

$$x_0 \sim N\left(\mu, \frac{\sigma^2}{1-\phi^2}\right) \quad (3)$$

Figure 1: Scatter Plot of Markov Chain

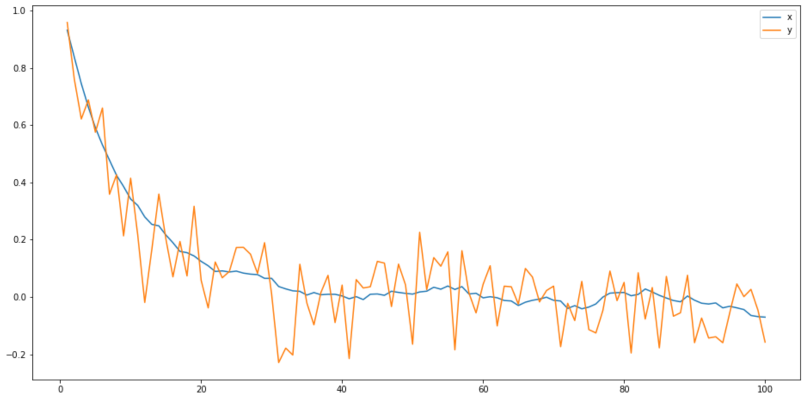
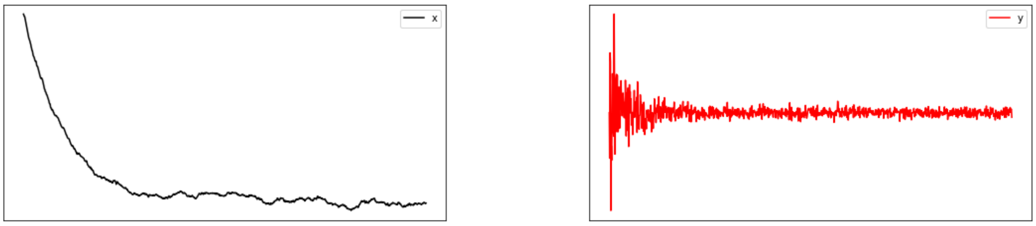


Figure 2: Gaussian Time Series Corrupted By Noise



(a) Decaying System

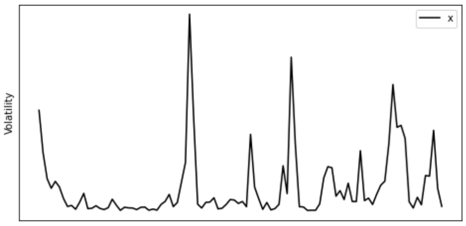
(b) Variance of a Decaying System

Figure 3: Decaying System as our 'Sanity' Check

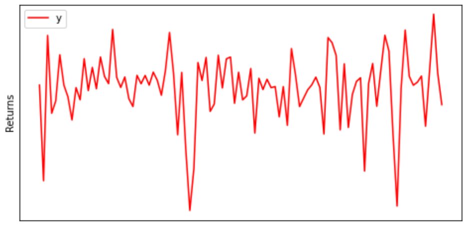
where (1) is our Markov transition kernel, (2) is our likelihood and (3) is our initial starting point. This Gaussian kernel would give negative values, however we note this kernel depicts log-volatility and so we can use it. Next we define our likelihood as (2) and will be used to model returns and can take negative values, additionally the use of the exponential is due to the 'normally distributed' nature of returns. As we can see in the figures below our model and simulations model the behaviour as expected.

A Code for 1

```
def discrete(s, w):
    cw = np.cumsum(w)
    sample = []
    u = np.random.uniform(0,1)
    for k in range(len(cw)):
        if cw[k] > u:
            sample = s[k]
            break
```

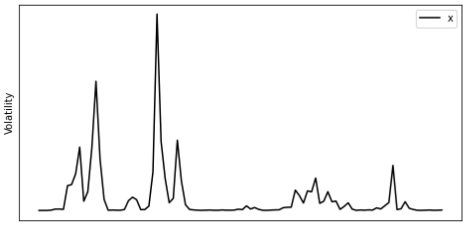


(a) Volatility

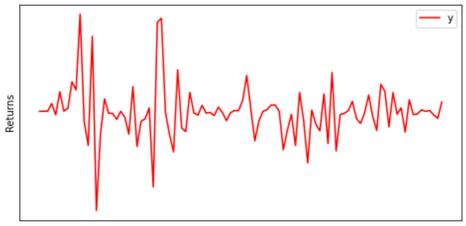


(b) Returns

Figure 4: Simulation of our Model

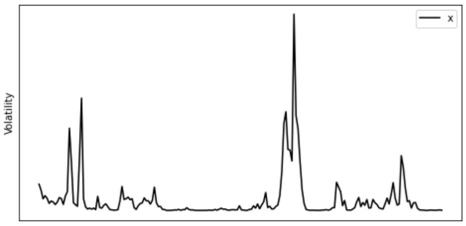


(a) Volatility

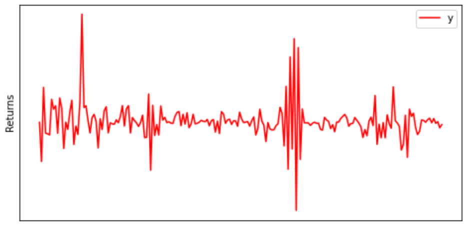


(b) Returns

Figure 5: Simulation of our Model



(a) Volatility



(b) Returns

Figure 6: Simulation of our Model

```
    return sample

A1 = np.array([[0.4, -0.3733], [0.06, 0.6]])
A2 = np.array([[-0.8, -0.1867], [0.1371, 0.8]])
b1 = np.array([[0.3533], [0.0]])
b2 = np.array([[1.1], [0.1]])

s = [1, 2]
w = [0.2993, 0.7007]
N = 10000
x0 = np.array([[0],[0]])

```

```
def Q1(x0, s, w, N):
    x = x0
    sample = np.zeros((2,N))
    for j in range(N+1):
        i = discrete(s, w)
        if i == 1:
            sample[0][j-1] = (A1@x + b1)[0]
            sample[1][j-1] = (A1@x + b1)[1]
            x = A1@x + b1
        if i == 2:
            sample[0][j-1] = (A2@x + b2)[0]
            sample[1][j-1] = (A2@x + b2)[1]
            x = A2@x + b2
    return sample

x = Q1(x0, s, w, N)
fig1 = plt.figure(figsize=(12,8))
plt.scatter(x[0, 20:N], x[1, 20:N], s=0.1, color = [0.8, 0, 0])
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['bottom'].set_visible(False)
plt.gca().spines['left'].set_visible(False)
plt.gca().set_xticks([])
plt.gca().set_yticks([])
plt.gca().set_xlim(0, 1.05)
plt.gca().set_ylim(0, 1)
plt.show()
fig1.savefig('1.png')
```

B Code for 2.1

```
x0 = 1
a = 0.9
xsigma = 0.01
ysigma = 0.1
T = 100
x = np.zeros(T+1)
y = np.zeros(T+1)

for t in range(T+1):
    if t == 0:
        x[t] = x0
    else:
        x[t] = np.random.normal(a*x[t-1], xsigma)
        y[t] = np.random.normal(x[t], ysigma)

fig2 = plt.figure(figsize=(16,8))
plt.plot(np.arange(1,101,1), x[1:], label= 'x')
plt.plot(np.arange(1,101,1),y[1:], label = 'y')
plt.legend()
fig2.savefig('2.png')
```

C Code for 2.2

```
x = np.zeros(T)
y = np.zeros(T)

for i in range(T):
    if i == 0:
        x[i] = np.random.normal(mu, np.sqrt((sigma**2)/(1 - phi**2)))
    else:
        x[i] = np.random.normal(mu + phi * (x[i-1] - mu), sigma)
        y[i] = np.random.normal(0, np.exp(x[i]/2))

plt.figure(figsize=(8,4))
plt.plot(np.exp(x), color = 'black', label = 'x')
plt.xticks([])
plt.yticks([])
plt.legend()
plt.ylabel('Volatility')

plt.figure(figsize=(8,4))
plt.plot(y, color = 'red', label = 'y')
plt.xticks([])
plt.yticks([])
plt.legend()
plt.ylabel('Returns')
```

References

[1] Sangjoom Kim, Neil Shephard, Siddartha Chib (1998) Stochastic Volatility: Likelihood Inference and Comparison with ARCH Models.