Problem set # 7

Problem 1: Simulation in the Heston Model: Suppose that the underlying security SPY evolves according to the Heston model. That is, we know its dynamics are defined by the following system of SDEs:

$$dS_t = rS_t \mathrm{dt} + \sqrt{\nu_t} S_t dW_t^1$$

$$d\nu_t = \kappa(\theta - \nu_t)\mathrm{dt} + \sigma\sqrt{\nu_t} dW_t^2$$

$$\mathrm{Cov}\left(dW_t^1, dW_t^2\right) = \rho \mathrm{dt}$$

You know that the last closing price for SPY was 282. You also know that the dividend yield for SPY is 1.77% and the corresponding risk-free rate is 1.5%. Using this information, you want to build a simulation algorithm to price a knock-out option on SPY, where the payoff is a European call option contingent on the option not being knocked out, and the knock-out is an upside barrier that is continuously monitored. We will refer to this as an up-and-out call. This payoff can be written as:

$$c_0 = E\left[(S_T - K_1)^+ 1_{\{M_T < K_2\}}\right]$$

where MT is the maximum value of S over the observation period, and K1 < K2 are the strikes of the European call and the knock-out trigger respectively.

1. Find a set of Heston parameters that you believe govern the dynamics of SPY. You may use results from a previous Homework, do this via a new calibration, or some other empirical process. Explain how you got these and why you think they are reasonable.

According to the result from last homework, I choose the set of parameters as:

```
clear;
s0 = 282; q = 1.77/100; r = 1.5/100;
kappa=3; theta=0.1; sigma=0.2; rho=-0.4; v0=0.08;
```

kappa is the intensity of mean-reverting, which should be greater than 0;

theta represents the "long-term" level of volatility, and 10% fits for SPY;

sigma is the "volatility of volatility", and 20% should be fine;

rho is the correlation between random shock of underlying asset and volatility, according to the common phenomenon from market that volatility increases as underlying asset plummet, rho should be negative and -0.4 is okay;

v0 is the initial level of volatility, and 8% is good to go.

2. Choose a discretization for the Heston SDE. In particular, choose the time spacing, ∆T as well as the number of simulated paths, N. Explain why you think these choices will lead to an accurate result.

I choose the Euler discretization for the Heston model and ∆T = 1/252, one day every step, which is intuitive and easy to implement. And N = 10000, since simulation converges as the speed of 1/sqrt(N), N=10000 is likely to give a precision of 0.01, and this would be suffient for SPY.

```
dt = 1/252; N = 10000;
```

3. Write a simulation algorithm to price a European call with strike K = 285 and time to expiry T = 1. Calculate the price of this European call using FFT and comment on the difference in price.

```
K = 285; T = 1;
s_path = zeros(T/dt,N); v_path = zeros(T/dt,N);
s_path(1,:) = ones(1,N)*s0; v_path(1,:) = ones(1,N)*v0;

for n = 1:N
    wt_path = mvnrnd([0 0],dt*[1 rho;rho, 1],T/dt);
    wt1 = wt_path(:,1);
    wt2 = wt_path(:,2);
    for i = 2:(T/dt)
        v_path(i,n) = v_path(i-1,n) + kappa*(theta-v_path(i-1,n))*dt + sigma*sqrt(v_pa
        s_path(i,n) = s_path(i-1,n) + (r-q)*s_path(i-1,n)*dt + sqrt(v_path(i-1,n))*s_pa
    end
end

c_sim = exp(-(r-q)*T)*mean(max(0, s_path(end,:) - K))
```

```
c_sim = 31.4614
```

```
par = struct(...
'rf', r, 'q', q, 'x0', log(s0), 'v0', v0, 'kappa', kappa, ...
'theta', theta, 'sigma', sigma, 'rho', rho);
aux.K = K;
aux.x0 = par.x0;
cf = @(u) cflib(u, T, par, 'Heston');
c_fft = cf2call(cf, aux)
```

```
c_fft = 31.7961
```

results from simulation are basically the same with results from FFT. difference in place is resulted from the speed of convergence of simulation. In other words, N is not large enough.

4. Update your simulation algorithm to price an up-and-out call with T = 1, K1 = 285 and K2 = 315. Try this for several values of N. How many do you need to get an accurate price?

```
K1 = 285; K2 = 315; T = 1;
N_list = [500, 5000 ,10000, 20000];
uoc_sim = zeros(1, size(N_list,2));
for j = 1:size(N_list,2)
    N = N_list(j);
    s_path = zeros(T/dt,N); v_path = zeros(T/dt,N);
    s_path(1,:) = ones(1,N)*s0; v_path(1,:) = ones(1,N)*v0;
    for n = 1:N
        wt_path = mvnrnd([0 0],dt*[1 rho;rho, 1],T/dt);
        wt1 = wt_path(:,1);
        wt2 = wt_path(:,2);
        for i = 2:(T/dt)
            v_path(i,n) = v_path(i-1,n) + kappa*(theta-v_path(i-1,n))*dt + sigma*sqrt(v
            s_path(i,n) = s_path(i-1,n) + (r-q)*s_path(i-1,n)*dt + sqrt(v_path(i-1,n))
        end
    end
    uoc_sim(j) = exp(-(r-q)*T)*mean(max(0, s_path(end,:) - K1).* (s_path(end,:) < K2))
end
uoc_sim
```

```
uoc_sim = 1×4
    1.3938    1.8748    1.8525    1.7523
```

N should be larger than 5000, in order to get an relative accurate price.

5. Re-price the up-and-out call using the European call as a control variate. Try this for several values of N. Does this converge faster than before?

```
K1 = 285; K2 = 315; T = 1;
Ez = c_fft;
N_list = [500, 5000 ,10000, 20000];
uoc_sim = zeros(1, size(N_list,2));
for j = 1:size(N_list,2)
    N = N_list(j);
    s_path = zeros(T/dt,N); v_path = zeros(T/dt,N);
    s_path(1,:) = ones(1,N)*s0; v_path(1,:) = ones(1,N)*v0;
    for n = 1:N
        wt_path = mvnrnd([0 0],dt*[1 rho;rho, 1],T/dt);
        wt1 = wt_path(:,1);
        wt2 = wt_path(:,2);
        for i = 2:(T/dt)
            v_path(i,n) = v_path(i-1,n) + kappa*(theta-v_path(i-1,n))*dt + sigma*sqrt(
            s_path(i,n) = s_path(i-1,n) + (r-q)*s_path(i-1,n)*dt + sqrt(v_path(i-1,n))
        end
    end
    hx = max(0, s_path(end,:) - K1).* (s_path(end,:) < K2);
    z =  max(0, s_path(end,:) - K1);
    cov_theta_z = cov(hx, z);
    c = - cov_theta_z(1,2) / var(z);
    theta_bar = hx + c*(z-Ez);
    uoc_sim(j) = exp(-(r-q)*T)*mean(theta_bar);
end
uoc_sim
```

```
uoc_sim = 1×4
    1.6997    1.8276    1.8221    1.8483
```

Yes, this does converge faster than before. It's obvious the first result (when N = 500) is closer to accurate price.