Project 3 - Mastery Question

Tom McGrath

April 10, 2014

1 Mastery Question

1.1 MQ.i.

I used the Euler-Marayama method to generate a trajectory for the process:

$$dX_t = (\mu - \nu_t/2)dt + \sigma_t dW_t^1 \tag{1}$$

$$d\nu_t = \kappa(\alpha - \nu_t)dt + \gamma\nu^{\frac{1}{2}}dW_t^2 \tag{2}$$

where $\mathbb{E}(dW_t^1 dW_t^2) = \rho dt$ and $\sigma_t^2 = \nu_t$ with constants $\mu = 0.05$, $\kappa = 5$, $\alpha = 0.04$, $\gamma = 0.5$ and $\rho = -0.5$ on the domain $[0, 10]$. This choice of constants satisfies the Feller condition $2\kappa\alpha \geq \gamma^2$ (from Ait-Sahalia's paper) so there is in theory no danger of crossing the zero boundary and generating complex-valued results. In practice with a discretisation of $\Delta t = 10^{-3}$ this does occasionally occur. When this happened I re-generated the results.

I then added noise:

$$Y_{t_i} = X_{t_i} + \epsilon_{t_i}, \quad i = 0, ..., N \tag{3}$$

with independently distributed noise $\epsilon_{t_i} \sim \mathcal{N}(0, 0.1)$. A sample trajectory is shown below.

1.2 MQ.ii.

The quadratic variation $\langle Y, Y \rangle$ is given by:

$$\langle Y, Y \rangle = \lim_{\Delta t_k \to 0} |X_{t_{k+1}} - X_{t_k}|^2 \tag{4}$$

which gives the drift coefficient as as:

$$\sigma_J^2 = \frac{1}{J\delta} \sum_{j=0}^{J-1} (X_{j+1} - X_j)^2 \tag{5}$$

and this converges as $J \to \infty$ to σ^2. Sampling using this method gives the drift coefficient as 0.1 in the case of using all the data with this estimator. We can obtain the integrated

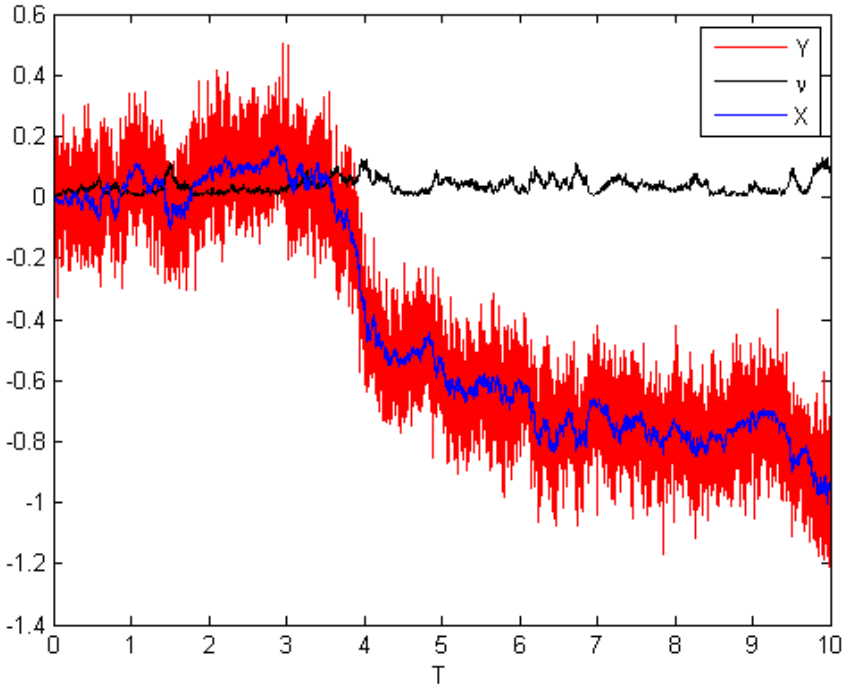

Figure 1: Sample trajectory from the Heston process

stochastic volatility (ISV) using the equatioin

$$\lim \sum (X_{t_{i+1}} - X_{t_i})^2 = \int_0^T \sigma_t^2 dt \tag{6}$$

Using different subsampling frequencies alters the ISV as follows: I suspect I have made a mistake here - the ISV appears to go to nearly zero as the sampling rate increases and there does not appear to be any indication of optimality. However, the ISV shows the correct behaviour (sampling the noise!) when all of the data is used.

1.3 MQ.iii.

We can improve on the subsampling method by using a multi-grid subsampling method, followed by averaging the subsamples. The optimal grid size is given by:

$$n_{opt} = \left(\frac{T}{6(\sigma_\epsilon^2)^2} \int_0^T \sigma_t^4 dt \right)^{\frac{1}{3}} \tag{7}$$

I chose a grid size of 300. Assigning each point to a grid equidistantly (Ait-Sahalia shows this is optimal in section 4.2 of the paper) the variance of the averaged estimator is calculated to be 2.3477×10^{-6}. This is higher than expected from Ait-Sahalia's table of

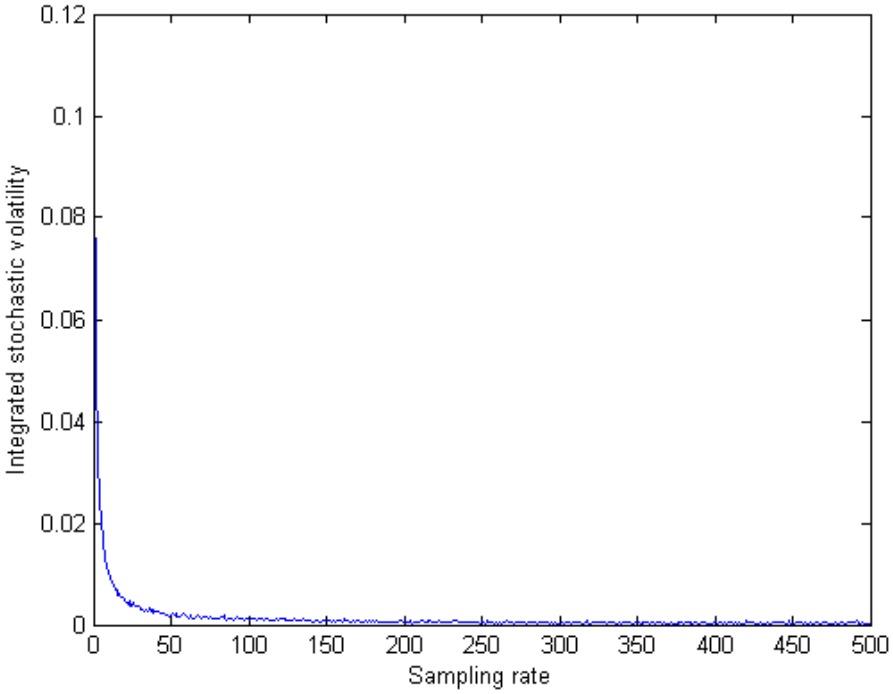

Figure 2: Variation of ISV with subsampling frequency

results (Table 2) which indicates I may have made an error in my code as the parameters chosen are the same.

1.4 MQ.iv.

I have not written code for this section as I think I have made a coding error in the previous section. However, all that would be necessary is to subtract the estimate from section ii from the multi-grid variance estimate given by section iii. This gives the first-best estimator and is unbiased as it removes the $2n\sigma_\epsilon^2$ bias term from the second-best estimator. This could be shown by observing the convergence of mean of the estimator as the number of samples increases.

2 Code

```
rate = 100;
Tmax = 10;
dt = 0.001;
T = linspace(0,Tmax,Tmax/dt);
```

```
mu = 0.05;
kappa = 5;
alpha = 0.04;
gamma = 0.5;
rho = -0.5;

cov = [1,rho;rho,1];

dW = sqrt(dt)*(1/sqrt(2))*cov*randn(2,length(T));
X = zeros(1,length(T));
v = zeros(1,length(T));
Y = zeros(1,length(T));
eta = 0.1*randn(1,length(T));

% generate v
v(1) = 0;
for i = 2:length(T)
   v(i) =  v(i-1) + kappa*(alpha - v(i-1))*dt + sqrt(v(i-1))*gamma*dW(1,i);
end

% generate X
X(1) = 0;
for i = 2:length(T)
    X(i) = X(i-1) + (mu - 0.5*v(i))*dt + sqrt(v(i))*dW(2,i);
end

% generate Y
for i = 1:length(T)
   Y(i) = X(i) + eta(i);
end

qv = 0;
for i = 1:rate:length(T)-rate
    qv = qv+(Y(i+rate)-Y(i))^2;
end

qv = qv/(2*Tmax/dt);
var(X)
var(Y)
qv
sigX = (0.5*dt/Tmax)*sum(diff(X).^2)
sigY = (0.5*dt/Tmax)*sum(diff(Y).^2)
% mean(v)
```

```
%
plot(T,Y, 'Color', 'red')
hold on
plot(T,v, 'Color', 'black')
plot(T,X, 'Color', 'blue')
hold off


function [ qv ] = MQfn( rate )
Tmax = 10;
dt = 0.001;
T = linspace(0,Tmax,Tmax/dt);

mu = 0.05;
kappa = 5;
alpha = 0.04;
gamma = 0.5;
rho = -0.5;

cov = [1,rho;rho,1];

dW = sqrt(dt)*(1/sqrt(2))*cov*randn(2,length(T));
X = zeros(1,length(T));
v = zeros(1,length(T));
Y = zeros(1,length(T));
eta = 0.1*randn(1,length(T));

% generate v
v(1) = 0;
for i = 2:length(T)
   v(i) =  v(i-1) + kappa*(alpha - v(i-1))*dt + sqrt(v(i-1))*gamma*dW(1,i);
end

% generate X
X(1) = 0;
for i = 2:length(T)
    X(i) = X(i-1) + (mu - 0.5*v(i))*dt + sqrt(v(i))*dW(2,i);
end

% generate Y
for i = 1:length(T)
   Y(i) = X(i) + eta(i);
end
```

```
Z = zeros(1,floor(length(T)/rate));
for i = 1:length(Z)
   Z(i) = Y(rate*i);
end

qv = 0.5*dt*sum(diff(Z).^2);

end

function [ qv ] = MQfn2( rate )
Tmax = 10;
dt = 0.001;
T = linspace(0,Tmax,Tmax/dt);

mu = 0.05;
kappa = 5;
alpha = 0.04;
gamma = 0.5;
rho = -0.5;

cov = [1,rho;rho,1];

dW = sqrt(dt)*(1/sqrt(2))*cov*randn(2,length(T));
X = zeros(1,length(T));
v = zeros(1,length(T));
Y = zeros(1,length(T));
eta = 0.1*randn(1,length(T));

% generate v
v(1) = 0;
for i = 2:length(T)
   v(i) =  v(i-1) + kappa*(alpha - v(i-1))*dt + sqrt(v(i-1))*gamma*dW(1,i);
end

% generate X
X(1) = 0;
for i = 2:length(T)
    X(i) = X(i-1) + (mu - 0.5*v(i))*dt + sqrt(v(i))*dW(2,i);
end

% generate Y
for i = 1:length(T)
```

```
   Y(i) = X(i) + eta(i);
end

grids = zeros(1,rate);

for i = 1:rate
   for j = i:rate:(length(T)-rate)
      grids(i) = grids(i) + (Y(j+rate)-Y(j))^2;
   end
end

qv = mean(grids);
qv = rate*qv/(2*Tmax/dt);

end
```