Computational Finance
FIN-472
Homework 7 - Solutions

November 3, 2017

Exercise 1: Consider a GARCH model $(X_t)_{0\leq t\leq T}$ of the form

$$dX_t = \kappa(\theta - X_t)\,dt + \sigma X_t\,dW_t,$$

where κ, θ and σ are model parameters.

a) Prove that X_t is a polynomial diffusion and write its infinitesimal generator $\mathcal{G}$.

Solution: We have that the drift and volatility function are given by

$$b(x) = \kappa(\theta - x) \in \mathrm{Pol}_1(\mathbb{R}), \quad a(x) = \sigma^2 x^2 \in \mathrm{Pol}_2(\mathbb{R}).$$

By the characterization theorem (Lemma 2, page 9, slides Lecture 7) X_t is a polynomial diffusion. The infinitesimal generator is

$$\mathcal{G}f(x) = \kappa(\theta - x)f'(x) + \frac{1}{2}\sigma^2 x^2 f''(x).$$

b) For $u \in \mathbb{R}$, define v as

$$v(t,x) = \mathbb{E}[\exp(iuX_T)|X_t = x].$$

If the conditions of the Feyman-Kac theorem are satisfied, v solves the equation

$$v_t + \mathcal{G}v = 0, \quad v(T,x) = \exp(iux).$$

Can we write the solution of the PDE as

$$v(t,x) = \exp(\phi(T-t) + \psi(T-t)x)$$

for some function ϕ and ψ with $\phi(0) = 0$ and $\psi(0) = iu$?

Solution: If one assumes that $v(t,x)$ has the exponential affine form then

$$v_t = -v(\phi'(T-t) - \psi'(T-t)x), \;\; v_x = v\psi(T-t), \;\; v_{xx} = v\psi^2(T-t).$$

Then we would need to have

$$-\phi' - \psi' x + \psi\kappa(\theta - x) + \frac{1}{2}\sigma^2 x^2 \psi^2 = 0,$$

for all x. This would imply that $\psi^2 \equiv 0$ and hence it would be impossible to have $\psi(0) = iu$. This explains why X_t is not an affine diffusion.

c) Solve explicitly the differential equation for X_t.

Hint: Consider the Ansatz $L_t := e^{\left(\kappa+\frac{\sigma^2}{2}\right)t-\sigma W_t}X_t$.

Solution: Considering

$$L_t = \mathrm{e}^{\left(\kappa+\frac{\sigma^2}{2}\right)t-\sigma W_t}X_t$$

and using Itô, we can conclude that L_t solves the equation

$$dL_t = \kappa\theta \mathrm{e}^{\left(\kappa+\frac{\sigma^2}{2}\right)t-\sigma W_t}\,dt, \quad L_0 = X_0.$$

Hence

$$X_t = X_0\mathrm{e}^{-\left(\kappa+\frac{\sigma^2}{2}\right)t+\sigma W_t} + \kappa\theta\int_0^t \mathrm{e}^{\left(\kappa+\frac{\sigma^2}{2}\right)(s-t)+\sigma(W_t-W_s)}\,ds.$$

d) For $N \in \mathbb{N}$, write the matrix representation G_N of the infinitesimal generator restricted to $\mathrm{Pol}_N(\mathbb{R})$, with respect to the monomial basis given by

$$H_N(x) = (1, x, x^2, \cdots, x^N).$$

Solution: Plug in $b = \kappa\theta$, $\beta = -\kappa$, $a = \alpha = 0$ and $A = \sigma^2$ in the formula of slide 14 of Lecture 7, in order to get

$$G_n = \begin{pmatrix} 0 & \kappa\theta & 0 & 0 & \dots & 0 \\ 0 & -\kappa & 2\kappa\theta & 0 & \dots & 0 \\ 0 & 0 & 2(-\kappa+\sigma^2/2) & 3\kappa\theta & \ddots & 0 \\ 0 & 0 & 0 & 3(-\kappa+\sigma^2) & \ddots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & n\kappa\theta \\ 0 & 0 & 0 & 0 & \dots & n(-\kappa+(n-1)\sigma^2/2) \end{pmatrix} \in \mathbb{R}^{n+1\times n+1}.$$

e) Use the moment formula for polynomial diffusions to calculate the first moment $\mathbb{E}[X_T]$. Check that the obtained result is coherent with what one gets from the explicit formula derived in part c).

Solution: By using the moment formula for polynomial diffusions we get

$$\begin{aligned} \mathbb{E}[X_T] &= (1, X_0)\exp(G_1 T)\begin{pmatrix}0\\1\end{pmatrix} \\ &= (1, X_0)\exp\left(\begin{pmatrix}0 & \kappa\theta\\ 0 & -\kappa\end{pmatrix}T\right)\begin{pmatrix}0\\1\end{pmatrix} \\ &= (1, X_0)\begin{pmatrix}1 & \theta e^{-\kappa T}(e^{\kappa T}-1)\\ 0 & e^{-\kappa T}\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix} \\ &= \theta e^{-\kappa T}(e^{\kappa T}-1) + X_0 e^{-\kappa T}. \end{aligned}$$

While by computing the expectation directly by using the explicit for of X_T, one has

$$
\begin{aligned}
\mathbb{E}[X_T] &= \mathbb{E}[X_0 \mathrm{e}^{-\left(\kappa+\frac{\sigma^2}{2}\right)T+\sigma W_T} + \kappa\theta \int_0^T \mathrm{e}^{\left(\kappa+\frac{\sigma^2}{2}\right)(s-T)+\sigma(W_T-W_s)} ds] \\
&= X_0 e^{-\kappa T} + \kappa\theta \int_0^T e^{\left(\kappa+\frac{\sigma^2}{2}\right)(s-T)+\frac{\sigma^2}{2}(T-s)} ds \\
&= X_0 e^{-\kappa T} + \kappa\theta \int_0^T e^{\kappa(s-T)} ds \\
&= X_0 e^{-\kappa T} + \theta e^{-\kappa T}(e^{\kappa T} - 1),
\end{aligned}
$$

which is coherent with what we got above.

f) Consider the set of parameters

$$\kappa = 0.5, \;\; \theta = 0.4, \;\; \sigma = 0.2, \;\; X_0 = 1, \;\; T = 0.5.$$

Use the moment formula for polynomial diffusions to find the first 4 moments

$$\mathbb{E}[X_T], \;\; \mathbb{E}[X_T^2], \;\; \mathbb{E}[X_T^3], \;\; \mathbb{E}[X_T^4]$$

and calculate the 4-order “approximation” of the density of X_T with a Gaussian that matches the first two moments. Plot the density approximation for orders $1, 2, 3, 4$.

Solution: The first 4 moments are given by

$$\mathbb{E}[X_T] \approx 0.87, \;\; \mathbb{E}[X_T^2] \approx 0.77, \;\; \mathbb{E}[X_T^3] \approx 0.69, \;\; \mathbb{E}[X_T^4] \approx 0.63.$$

A possible Matlab implementation performing the density approximation can be found hereunder. Moreover, the produced plot showing the approximated densities can be seen in Figure 1.

```
function [f, M, NM, HM] = GARCH_Dens_Approx(kappa,theta,sigma,X0,N,T)
% Density approximation for a GARCH model
% dX_t = kappa*(theta-X_t)d_t + sigma*X_t dW_t
%
% Input:  -X0: initial value
%         -N: number of moments calculated (N ≤ 4)
% Output: -M: vector of moments
%         -NM: vector of normalized moments
%         -HM: vector of Hermite moments
%         -f: density approximation up to order N
if N > 4
    error('N too large. Maximal allowed N: 4')
end
% If N == 1, we switch for one second to N = 2 in order to compute
% muw and sigw
N1 = N;
if N1 == 1
    N = 2;
end

% Construction of the Matrix G_N
```

```
d1 = (0:N) .* (-kappa + [0 0:N-1] * sigma^2/2);
d2 = (1:N) * kappa * theta;
G = diag(d1, 0) + diag(d2, 1);

% Calculation of moments E[X_T^n] via moment formula
H = X0.^(0:N);
M = H * expm(T*G);

% Mean and variance of X_T
mu = M(2);
sigw = sqrt(M(3)-M(2)^2);

% Switch back if N == 1
if N1 == 1
    N = 1;
end

% Calculation of ''normalized'' moments of the form
% E[((X_T-mu)/sqrt(2)*sigfit)^n] via binomial formula

NM = zeros(N+1,1);
for n = 1:N+1
    for m = 0:n-1
     NM(n) = NM(n) + ...
         (nchoosek(n-1,m)*(-mu)^(n-1-m)/(sqrt(2)*sigw)^(n-1))*M(m+1);
    end
end

% Hermite moments
syms y
HM = zeros(N+1,1);
for n = 1:N+1
    hermite_coeff = coeffs(hermiteH(n-1, y),'All');
    HM(n) = 2^(-0.5*(n-1))/sqrt(factorial(n-1))*(hermite_coeff * ...
        NM(n:-1:1));
end
% N - th order approximation of the density

w = @(x)(normpdf(1.*x, mu, sigw));

h1 = ...
    @(x)(2^(-0.5*(1))/sqrt(factorial(1))*polyval(double(coeffs(hermiteH(1, ...
    y),'All')),(1.*x-mu)/(sqrt(2)*sigw)));

h2 = ...
    @(x)(2^(-0.5*(2))/sqrt(factorial(2))*polyval(double(coeffs(hermiteH(2, ...
    y),'All')),(1.*x-mu)/(sqrt(2)*sigw)));

h3 = ...
    @(x)(2^(-0.5*(3))/sqrt(factorial(3))*polyval(double(coeffs(hermiteH(3, ...
    y),'All')),(1.*x-mu)/(sqrt(2)*sigw)));

h4 = ...
    @(x)(2^(-0.5*(4))/sqrt(factorial(4))*polyval(double(coeffs(hermiteH(4, ...
    y),'All')),(1.*x-mu)/(sqrt(2)*sigw)));

```

```
if N == 1
 f = @(x)(1+HM(2).*h1(x)).*w(x);
end
if N == 2
 f = @(x)(1+HM(2).*h1(x)+HM(3).*h2(x)).*w(x);
end

if N == 3
 f = @(x)(1+HM(2).*h1(x)+HM(3).*h2(x)+HM(4).*h3(x)).*w(x);
end

if N == 4
 f = @(x)(1+HM(2).*h1(x)+HM(3).*h2(x)+HM(4).*h3(x)+HM(5).*h4(x)).*w(x);
end
end
```

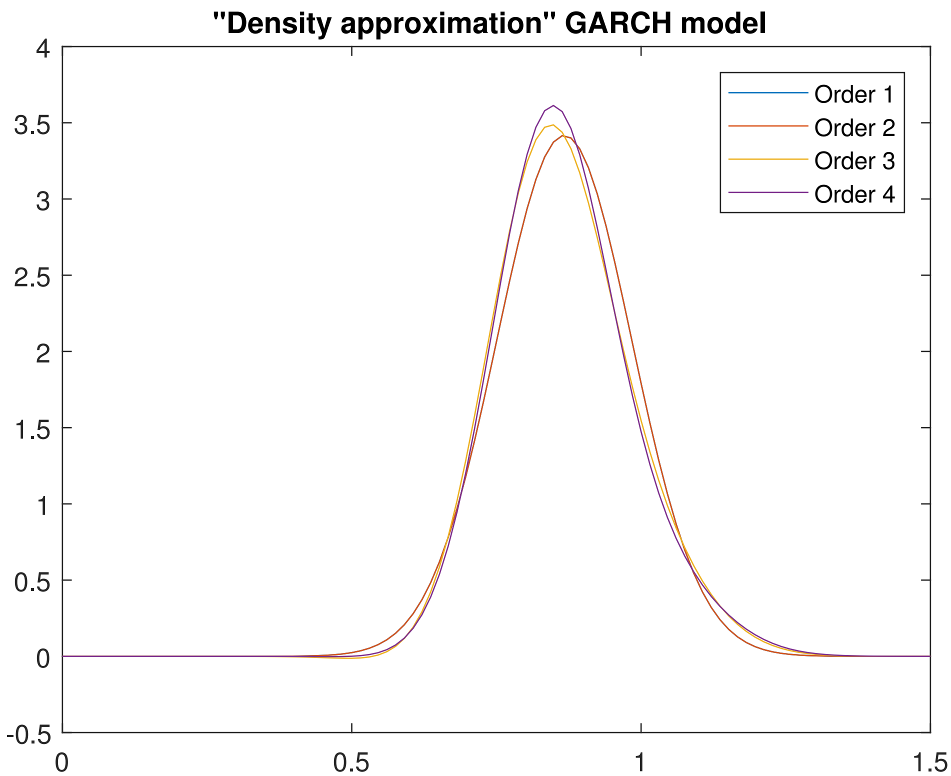

Figure 1: Density approximations for GARCH model.

Exercise 2: Consider the Heston model where the squared volatility V_t and the log-asset price X_t are given by

$$dV_t = \kappa(\theta - V_t)dt + \sigma\sqrt{V_t}dW_t^{(1)},$$
$$dX_t = (r - V_t/2)dt + \rho\sqrt{V_t}dW_t^{(1)} + \sqrt{V_t}\sqrt{1-\rho^2}dW_t^{(2)}.$$

Here, $W^{(1)}$ and $W^{(2)}$ are independent Brownian motions and $\kappa, \theta, \sigma, \rho$ are model parameters.

a) Let $\mu_w \in \mathbb{R}$ and $\sigma_w > 0$ be arbitrary parameters. Consider the basis of $\mathrm{Pol}_N(\mathbb{R}^2)$ defined as

$$\mathcal{H}_N = \{1, v, \frac{x-\mu_w}{\sigma_w}, v^2, v\Big(\frac{x-\mu_w}{\sigma_w}\Big), \Big(\frac{x-\mu_w}{\sigma_w}\Big)^2, \cdots, v^n, v^{n-1}\Big(\frac{x-\mu_w}{\sigma_w}\Big), \cdots, \Big(\frac{x-\mu_w}{\sigma_w}\Big)^n\}$$

and write it in a row vector

$$H_N = (1, v, \frac{x-\mu_w}{\sigma_w}, v^2, v\Big(\frac{x-\mu_w}{\sigma_w}\Big), \Big(\frac{x-\mu_w}{\sigma_w}\Big)^2, \cdots, v^n, v^{n-1}\Big(\frac{x-\mu_w}{\sigma_w}\Big), \cdots, \Big(\frac{x-\mu_w}{\sigma_w}\Big)^n).$$

Note that the dimension of $\mathrm{Pol}_N(\mathbb{R}^2)$ is $M := \frac{(N+1)(N+2)}{2}$.
Define a bijective function

$$\pi : \{(m,n) \in \mathbb{N}_0 \times \mathbb{N}_0 \mid m, n \geq 0;\ \ m+n \leq N\} \to \{1, 2, \cdots, M\},$$

that describes the ordering for the basis H_N. In other words, each basis element of the form $v^m\Big(\frac{x-\mu_w}{\sigma_w}\Big)^n$ is stored in the position $\pi(m,n)$ in the vector H_N. Implement this function in Matlab and call it `Ind.m`.

Solution: The function π can be defined as

$$\pi(m,n) := \binom{m+n+1}{2} + n + 1 = \frac{(m+n+1)(m+n)}{2} + n + 1.$$

The Matlab function `Ind.m` can be easily written as suggested hereunder.

```
function [ Index ] = Ind( m, n )
% Input:  m,n - powers of v and (x-mu_w)/sigma_w of basis elements.
%
% Output: Index - position of the corresponding basis element in H_n.

Index = (m + n + 1) * (m + n) / 2 + n + 1;
end
```

b) Write a Matlab function `GenHeston.m` that constructs G_N, the matrix representation of the infinitesimal generator $\mathcal{G}$ restricted to $\mathrm{Pol}_N(\mathbb{R}^2)$ with respect to the basis H_N.

Solution: A possible basic Matlab implementation can be found in the following (note that we are using the results of slide 21 Lecture 7).

```
function [G] = GenHeston(kappa, theta, r, sigma, rho, N, sigma_w)
% Input:  - Parameter of the Heston model k, theta, r, sigma, rho
%         - N maximal total degree of polynomials
% Output: - A M x M matrix G, representation of the generator of the
%           Heston model with respect to the given basis

M = 1/2*(N+1)*(N+2);
G = zeros(M,M);
for m = 0:N
    for n = 0:N-m
        ColInd = Ind(m,n);
```

```
        if m > 0
            G(Ind(m-1,n), ColInd) = (kappa*theta+sigma^2*(m-1)/2)*m;
        end
        if n > 0
            G(Ind(m,n-1), ColInd) = n * (sigma*rho*m+r)/sigma_w;
            G(Ind(m+1,n-1), ColInd) = -n / (2 * sigma_w);
        end
        if n > 1
            G(Ind(m+1,n-2), ColInd) = n * (n-1) / (2*sigma_w^2);
        end
        G(ColInd, ColInd) = -kappa * m;
    end
end
end
```

c) Consider the model parameters

$$X_0 = 5.1,\ \ V_0 = 0.04,\ \ \kappa = 1,\ \ \theta = 0.04,\ \ \sigma = 0.2,\ \ r = 0.03,\ \ \rho = -0.8,\ \ T = 1/52,$$

together with $\mu_w = \mathbb{E}[X_T]$ and $\sigma_w^2 = \mathrm{Var}[X_T]$. Using the moment formula for polynomial diffusions, compute

$$\mathbb{E}[\Big(\frac{X_T - \mu_w}{\sigma_w}\Big)],\ \ \mathbb{E}[\Big(\frac{X_T - \mu_w}{\sigma_w}\Big)^2],\ \ \mathbb{E}[\Big(\frac{X_T - \mu_w}{\sigma_w}\Big)^3],\ \ \mathbb{E}[\Big(\frac{X_T - \mu_w}{\sigma_w}\Big)^4].$$

Solution: Here is a basic code that computes the first 4 moments as asked in the statement. The numerical results are

$$\mathbb{E}[\Big(\frac{X_T - \mu_w}{\sigma_w}\Big)] = 0, \quad \mathbb{E}[\Big(\frac{X_T - \mu_w}{\sigma_w}\Big)^2] = 1,$$
$$\mathbb{E}[\Big(\frac{X_T - \mu_w}{\sigma_w}\Big)^3] = -0.1654, \quad \mathbb{E}[\Big(\frac{X_T - \mu_w}{\sigma_w}\Big)^4] = 3.0433.$$

```
% We compute the first 4 moments as described in the statement of ...
    exercise
% 2c HW 7.

% Set of parameters
X0 = 5.1;
V0 = 0.04;
kappa = 1;
theta = 0.04;
sigma = 0.2;
r = 0.003;
rho = -0.8;
T = 1/52;

% In order to determine mu_w and sigma_w, we first assume mu_w = 0 and
% sigma_w = 1 and we apply the moment formula to compute E[X_T] and ...
    Var[X_T]
sigma_w = 1;
N = 2;
G = GenHeston(kappa, theta, r, sigma, rho, N, sigma_w);
```

```
H = [1, V0, X0, V0^2, V0*X0, X0^2];
moments = H * expm(G*T);
mu_w = moments(3);
sigma_w = sqrt(moments(6) - mu_w^2);

% We now compute the 4 moments as asked in the exercise
N = 4;
G = GenHeston(kappa, theta, r, sigma, rho, N, sigma_w);
X0_tilde = (X0-mu_w) / sigma_w;
H = zeros(length(G), 1);
for m = 0:N
    for n = 0:N-m
        ColInd = Ind(m,n);
        H(ColInd, 1) = V0^m * X0_tilde^n;
    end
end
moments = H' * expm(G*T);
% We print the needed moments
fprintf('First moment:  %d\n', moments(Ind(0,1)))
fprintf('Second moment:  %d\n', moments(Ind(0,2)))
fprintf('Third moment:  %d\n', moments(Ind(0,3)))
fprintf('Fourth moment:  %d\n', moments(Ind(0,4)))
```