Computational Finance
FIN-472
Take-Home Exam 3
Polynomial expansion methods

October 23, 2020

- **Please upload your solutions on moodle by Friday 06.11.2020 at 8h15.**
- **Please do not forget to upload in Moodle your Matlab codes. You should submit exactly one Matlab file for each part, for a total of 6 Matlab files.**
- **You should also upload one PDF file containing the Matlab codes and the plot for Exercise e).**

Consider the Jacobi stochastic volatility model where the stock price dynamics $S_t = e^{X_t}$ and the squared volatility V_t are given by

$$\begin{aligned} dV_t &= \kappa(\theta - V_t)\,dt + \sigma\sqrt{Q(V_t)}\,dW_{1t}, \\ dX_t &= (r - V_t/2)\,dt + \rho\sqrt{Q(V_t)}\,dW_{1t} + \sqrt{V_t - \rho^2\,Q(V_t)}\,dW_{2t}, \end{aligned}$$

for $\kappa, \sigma > 0$, $\theta \in (v_{min}, v_{max}]$, interest rate r, $\rho \in [-1, 1]$, and a 2-dimensional Brownian motion $W = (W_1, W_2)$. Here, the function Q is defined as

$$Q(v) = \frac{(v - v_{min})(v_{max} - v)}{(\sqrt{v_{max}} - \sqrt{v_{min}})^2},$$

for some parameters v_{min} and v_{max} satisfying $0 \le v_{min} < v_{max}$.
The first goal of this exercise is to implement the polynomial expansion method described in Lecture 6. This method allows us to write the price $\mathbb{E}[f(X_T)]$ of a European option maturing at time T with payoff function f as

$$\pi_f := \mathbb{E}[f(X_T)] = \sum_{n\ge 0} f_n l_n, \tag{1}$$

where $\{l_n, n = 0, \cdots\}$ are the Hermite moments and $\{f_n, n = 0, \cdots\}$ are the Fourier coefficients. The second goal is to implement a Monte Carlo algorithm that computes, again, the price $\mathbb{E}[f(X_T)]$ of the European option of interest in the Jacobi model. Finally, we will compare these two methods.

a) **(6/20)** Let $\mu_w \in \mathbb{R}$ and $\sigma_w > 0$ be arbitrary parameters. Consider the basis vector of $\mathrm{Pol}_N(\mathbb{R}^2)$ defined as

$$B_N(v, x) = (1, v, H_1(x), v^2, vH_1(x), H_2(x), \cdots, v^n, v^{n-1}H_1(x), \cdots, H_N(x)),$$

where $H_n(x)$ denotes the generalized Hermite polynomials

$$H_n(x) = \frac{1}{\sqrt{n!}}\mathcal{H}_n\left(\frac{x-\mu_w}{\sigma_w}\right), \quad n \geq 1.$$

Here, $\mathcal{H}_n(x)$ are the standard Hermite polynomials.

Write a Matlab function `HermiteMoments.m` that computes the first N Hermite moments using the moment formula given by

$$l_n = B_N(V_0, X_0)\,\mathrm{e}^{G_N T}\,\mathbf{e}_{\pi(0,n)}, \quad 0 \leq n \leq N,$$

where G_N is the matrix representation of the generator $\mathcal{G}$ of (V_t, X_t) restricted to $\mathrm{Pol}_N(\mathbb{R}^2)$, with respect to the basis B_N, and $\pi : \mathcal{E} \to \{1, \ldots, M = (N+2)(N+1)/2\}$ is an enumeration of the set of exponents

$$\mathcal{E} = \{(m,n) : m, n \geq 0;\ m+n \leq N\}.$$

Remark: You can use the same enumerating function π as defined in Exercise 2a of Homework 6.
In order to deal with the Hermite polynomials $\mathcal{H}_n$ you can use the built-in Matlab function `hermiteH.m`. Please also see the solutions of Homework 6 for a reference.

b) **(4/20)** Taking inspiration from Exercise 1 of Homework 5, *write* a Matlab function `SimSDEJacobi.m` that simulates N_{sim} paths of the process X_t, from $t = 0$ to $t = T$, via Euler discretization. In particular, the function should take in input the model parameters, the number of simulations N_{sim} and the number of time intervals N_{time}, and should return a vector of size $N_{sim} \times 1$ containing the simulated points at final time T. It is not necessary to store the whole paths.
Remark: Following the same reasoning as in Exercise 1 of Homework 5, make sure that the argument under the square roots is always positive.

From now on, we consider f to be the payoff function of a European call with log strike k,

$$f(x) := e^{-rT}(e^x - e^k)^+.$$

c) **(4/20)** In the case of the European call option, the Fourier coefficients in (1) can be recursively computed by

$$\begin{aligned} f_0 =& \mathrm{e}^{-rT+\mu_w} I_0\left(\frac{k-\mu_w}{\sigma_w};\sigma_w\right) - \mathrm{e}^{-rT+k}\Phi\left(\frac{\mu_w - k}{\sigma_w}\right), \\ f_n =& \mathrm{e}^{-rT+\mu_w}\frac{1}{\sqrt{n!}}\sigma_w I_{n-1}\left(\frac{k-\mu_w}{\sigma_w};\sigma_w\right), \quad n \geq 1. \end{aligned}$$

The functions $I_n(\mu;\nu)$ are defined recursively by

$$\begin{aligned} I_0(\mu;\nu) &= \mathrm{e}^{\frac{\nu^2}{2}}\Phi(\nu - \mu); \\ I_n(\mu;\nu) &= \mathcal{H}_{n-1}(\mu)\mathrm{e}^{\nu\mu}\phi(\mu) + \nu I_{n-1}(\mu;\nu), \quad n \geq 1 \end{aligned}$$

where $\mathcal{H}_n(x)$ are again the standard Hermite polynomials, $\Phi(x)$ denotes the standard Gaussian distribution function, and $\phi(x)$ its density.

Write a Matlab function `FourierCoefficients.m` that computes the first N coefficients following above recursions.

d) **(2/20)** *Write* a Matlab function `PriceApprox.m` that computes the approximation of the European call option price in the Jacobi model arising from cutting the sum in (1) after $N+1$ terms, i.e.

$$\pi_f^{(N)} := \sum_{n=0}^{N} f_n l_n.$$

e) **(2/20)** Consider the parameters

$$X_0 = 0,\ \ V_0 = 0.04,\ \ \kappa = 0.5,\ \ \theta = 0.04,\ \ \sigma = 1,\ \ r = 0,\ \ \rho = -0.5,$$
$$T = 1/12,\ \ v_{min} = 10^{-4},\ \ v_{max} = 0.08,\ \ N = 10,$$

together with $\mu_w = \mathbb{E}[X_T]$ and $\sigma_w^2 = \mathrm{Var}[X_T]$.

Using the function `PriceApprox.m`, *plot* the implied volatility smile as a function of the log strike k in the Jacobi model. In particular, for each value of k in `linspace(-0.1, 0.1, 50)` compute the corresponding implied volatility using the built-in Matlab function `blsimpv.m`. Moreover, plot on the same figure the implied volatility smile for the same call option computed in the Heston model, using the same model parameters.

Remarks:

- In order to compute the Heston price, please use the Fourier approach you have implemented in Exercise 3, Homework 3, with parameters $L = 100$ and $\alpha = 1$.
- For this part, please submit on the moodle the Matlab script/function that generates the needed plot.

f) **(2/20)** For the following model and payoff parameters

$$X_0 = 0,\ \ V_0 = 0.04,\ \ \kappa = 0.5,\ \ \theta = 0.04,\ \ \sigma = 1,\ \ r = 0,\ \ \rho = -0.5,$$
$$T = 1/12,\ \ v_{min} = 10^{-4},\ \ v_{max} = 0.08,\ \ k = -0.1$$

together with $\mu_w = \mathbb{E}[X_T]$ and $\sigma_w^2 = \mathrm{Var}[X_T]$, *compute* the corresponding European call option price using

- the polynomial expansion approach (formula (1)) with truncation level $N = 50$, and
- the standard Monte Carlo approach where you use the previously implemented function `SimSDEJacobi.m` with $N_{sim} = 10^6$ and $N_{time} = 100$ to obtain the needed simulated points.

Compute the absolute error between the two obtained prices. What can you conclude?