Christian-Albrechts-Universität zu Kiel **SS 2023**
Mathematisches Seminar
Prof. Dr. Sören Christensen
Henrik Valett, Fan Yu, Oskar Hallmann, Nele Rothert Sheet 08

Computational Finance

Exercises for all participants

C-Exercise 29 (Calculating the Delta/Hedge of a European option via MC methods, infinitesimal perturbation) (4 points)
Consider a Black-Scholes model with parameters r, $\sigma > 0$. The goal is to approximate the hedging position $\varphi_1(t)$ (check equation (3.29)) of a European option with payoff $g(S(T))$ at maturity T by the infinitesimal perturbation method for some time point $t \in [0,T]$ and current stock price $S(t)$. To this end write a Python function

```
phi1_t=EuOptionHedge_BS_MC_IP (St, r, sigma, g, T, t, N)
```

that calculates the hedging position $\varphi_1(t)$.

Test your function for $t = 0, S_0 = 100, r = 0.05, \sigma = 0.2, T = 1, N = 10000, g(x) = (x-90)^+$ and compare you result to the lecture notes by implementing the formula below (3.30).

Hint: To calculate the derivative needed for the infinitesimal perturbation you may use finite differences or the function `scipy.misc.derivative`.

C-Exercise 30 (Valuation of European options in the Heston model using the Euler method)
Write a Python function

```
[V0, c1, c2] = Heston_EuCall_MC_Euler (S0, r, gamma0, kappa,
                lambda, sigma_tilde, T, g, M, m)
```

that computes the price $V(0)$ of a European option with payoff $g(S_T)$ and maturity T in the Heston model via the Monte-Carlo method using M samples together with the 95%-confidence interval. To this end, approximate the paths by the Euler method with a grid of m equidistant points in time. Therefore first think how the Euler method from Section 4.2 in the lecture notes can be extended for the Heston model, where the model contains a stochastic price and a stochastic volatility process.
Test your function for the European call option using the parameters

$$S0 = 100, \quad r = 0.05, \quad \gamma(0) = 0.2^2, \quad \kappa = 0.5, \quad \lambda = 2.5,$$
$$\tilde{\sigma} = 1, \quad T = 1, \quad g(x) = (x-100)^+, \quad M = 10000, \quad m = 250.$$

C-Exercise 31 (Simulating paths of geometric Brownian motion) (4 points + 1 bonuspoint)
The Milshtein part of this exercise rewards one bonuspoint, if done correctly and can be ignored to get the regular 4 points.
Consider a geometric Brownian motion defined via

$$dX(t) = \mu X(t)dt + \sigma X(t)dW(t), \ \ X(0) = X_0,$$

with $\mu, X_0 \in \mathbb{R}$, $\sigma > 0$ and a Brownian motion $W(t)$. Write a function

```
(X_exact,X_Euler,X_Milshtein)=Sim_Paths_GeoBM(X0, mu, sigma, T, N)
```

that simulates the discrete path approximation $X(0), X(T/N), X(2T/N), \ldots, X(T)$ using the exact solution for the geometric Brownian motion SDE (`X_exact`), using the Euler method (`X_Euler`) and using the Milshtein method (`X_Milshtein`). Use the same simulations of $W(kT/N) - W((k-1)T/N)$, $k = 1, \ldots, N$, for all three methods.

Test the function for $X_0 = 100, \mu = 0.1, \sigma = 0.3, T = 1$ and $N = 10, 100, 1000, 10000$. For each value of N plot all three simulated paths into a single plot.

Hint: In order to implement the Milshtein method it is enough to implement the pseudo code on p.58 in the lecture notes. It is not mandatory to read through the complete chapter about 'Stochastic Taylor expansion' although it will help to understand why the Milshtein method yields a better approximation.

Please include your name(s) as comment in the beginning of the file.
Do not forget to include comments in your Python-programs.
Submit until: Fri, 16.06.2023, 10:00