Christian-Albrechts-Universität zu Kiel **SS 2022**
Mathematisches Seminar
Prof. Dr. Jan Kallsen
Henrik Valett, Fan Yu, Boy Schultz Sheet 02

Computational Finance

Exercises for all participants

C-Exercise 05 (4 points)
Let $s_1, \ldots, s_N$ denote a time series of, e.g., stock prices on days $1, \ldots, N$. The *logarithmic return* (log-return) of the stock on day $n \in \{2, \ldots, N\}$ is given by

$$l_n := \log\left(\frac{s_n}{s_{n-1}}\right) = \log(s_n) - \log(s_{n-1}).$$

Assuming 250 trading days per year, the *annualized empirical mean* of log-returns is given by

$$\hat{\mu} = \frac{250}{N-1} \sum_{k=2}^{N} l_k$$

and the *annualized empirical standard deviation* of log-returns is given by

$$\hat{\sigma} = \sqrt{\frac{250}{N-2} \sum_{k=2}^{N} \left(l_k - \frac{\hat{\mu}}{250}\right)^2}.$$

(a) Write a Python function

```
log_returns(data)
```

that computes and returns the time series of log-returns for the time series given in `data`.

(b) In the Material folder you find the file `time_series_dax_2022.csv` containing a time series of daily DAX data. Import this time series and test your function with it. This includes to

- apply the function to the imported time series,
- visualize the time series of log-returns in a plot,
- compute and display the annualized empirical mean and standard deviation of the log-returns.

(c) Simulate a time series of log-returns with the assumption that these are normal distributed. Use your result of the empirical mean and standard deviation as parameters. Plot this simulated log-returns in the same figure as the log-returns from the data (in different colors).

(d) What differences do you observe between the two time series? What do you conclude? (Write a short answer as comment at the end of your program.)

Hint: Use the data from the column ‘Close‘ and pay attention to correct symbols indicating the seperator between columns and the decimal point.

Useful Python commands: `numpy.diff`, `numpy.log`, `numpy.genfromtxt`, `numpy.mean`, `numpy.var`, `numpy.random.normal`, `print`, `str`

Black-Scholes formula

The fair price of a European Call option with strike $K > 0$ in the Black-Scholes model at time $0 \leq t \leq T$ with stock price $S(t)$ is given by:

$$C(t, S(t), r, \sigma, T, K) = S(t)\Phi(d_1) - Ke^{-r(T-t)}\Phi(d_2).$$

Here, Φ denotes the cumulative distribution function of the standard normal distribution and

$$d_1 := \frac{\log\left(\frac{S(t)}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}}, \quad d_2 := d_1 - \sigma\sqrt{T-t}.$$

The fair price of a European Put option with strike $K > 0$ in the Black-Scholes model at time $0 \leq t \leq T$ with stock price $S(t)$ is given by:

$$P(t, S(t), r, \sigma, T, K) = Ke^{-r(T-t)}\Phi(-d_2) - S(t)\Phi(-d_1).$$

Here, Φ denotes the cumulative distribution function of the standard normal distribution and

$$d_1 := \frac{\log\left(\frac{S(t)}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}}, \quad d_2 := d_1 - \sigma\sqrt{T-t}.$$

C-Exercise 06 (4 points)

We want to price call options in the CRR-model.

(a) Write a Python function

```
V_0 = CRR_EuCall (S_0, r, sigma, T, M, K)
```

that computes and returns an approximation to the price of an European call option with strike $K > 0$ and maturity $T > 0$ in the Black-Scholes model with initial stock price $S(0) > 0$, interest rate $r > 0$ and volatility $\sigma > 0$ using the CRR-model as presented in the course and $M \in \mathbb{N}$ time steps.
Hint: Use the results from C-Exercise 02 and the process from T-Exercise 07.

(b) We want to compare the CRR model to the true price in the BS-model. To this end implement the BS-Formula for European call options as a Python function:

```
V_0 = BlackScholes_EuCall (t, S_t, r, sigma, T, K)
```

(c) Compare the results by plotting the error of the CRR model against the BS-price in a common graph. Use the following parameters

$$S(0) = 100,\ r = 0.03,\ \sigma = 0.3,\ T = 1,\ M = 100,\ K = 70, \ldots, 200.$$

T-Exercise 07 (4 points)
We want to price an American put option with strike price $K = 1.2$ and time to maturity being three years. For this purpose we want to utilize a CRR model with $M = 3$ equally spaced time periods, $S(0) = 1$, $\sigma^2 = 0.3$ and an annual interest rate of 5%.

a) Draw and calculate the corresponding CRR model by hand (of course you can still use a calculator) and write beneath each point the corresponding price of the option (please round on four position after the decimal point after each calculation).

b) Calculate the replicating portfolio $\varphi = (\varphi_0, \varphi_1)$ for all time periods.

T-Exercise 08 (Barrier options in the CRR model) (for math 4 points; for QF 4 bonuspoints)
In the binomial model from Section 2.1 with parameters $S(0), r, \sigma, T > 0$ and $M \in \mathbb{N}$, we denote by

- V the fair price process of a *European call option* on the stock S with strike $K > 0$, i.e. its payoff is given by $V(T) = (S(T) - K)^+$,
- $\tilde{V}$ the fair price process of a *down-and-out call option* on the stock S with strike $K > 0$ and barrier $B < K$, i.e. its payoff is given by

$$\tilde{V}(T) = 1_{\{S(t_i) > B \text{ for all } i = 0, \ldots, M\}} (S(T) - K)^+,$$

- $\hat{V}$ the fair price process of a *down-and-in call option* on the stock S with strike $K > 0$ and barrier $B < K$, i.e. its payoff is given by

$$\tilde{V}(T) = 1_{\{S(t_i) \leq B \text{ for one } i = 0, \ldots, M\}} (S(T) - K)^+.$$

Outline (e.g. in pseudo code) an algorithm that computes the initial price $\tilde{V}(0), \hat{V}(0)$ of the barrier options in $O(M^2)$ steps, i.e. there is a constant $C > 0$ independent of M such that the algorithm terminates after less than CM^2 operations. Please explain where your algorithm differs from the algorithm for European call options presented in the lecture and why these changes make sense/are needed.

Hint:

(a) Start with the *down-and-out call option*.

(b) Express the *down-and-in call option* in terms of a *down-and-out call option*.

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: Thu, 28.04.2022, 08:15