Christian-Albrechts-Universität zu Kiel **SS 2024**
Mathematisches Seminar
Prof. Dr. Mathias Vetter
Henrik Valett, Fan Yu, Ivo Richert, Anton Schellin Sheet 07

Computational Finance

Exercises for all participants

T-Exercise 25QF (for QF only) (4 points)

A *Poisson process* N with intensity $\lambda \in \mathbb{R}_+$ is a process with right-continuous, increasing paths such that for all $s,t \in \mathbb{R}_+$ the increments $N(t+s)-N(t)$ are independent of $N(t)$ and such that $N(t)$ follows a Poisson distribution with parameter λt. For $\rho, \mu \in \mathbb{R}$ and a Poisson process N with intensity $\lambda \in \mathbb{R}_+$, compute the characteristic function of $X(t) := \rho N(t) - \mu t$.

T-Exercise 25Math (Merton's jump-diffusion model) (4 points)

In Merton's jump-diffusion model the logarithm $X = \log(S)$ of the stock price is of the form

$$X(t) = X(0) + \mu t + \sigma W(t) + \sum_{j=1}^{N(t)} Y_j,$$

where W is a standard Brownian motion, $\mu \in \mathbb{R}$, $\sigma > 0$, $N(t)$ is a Poisson random variable with parameter λt for $\lambda > 0$ and $Y_1, Y_2, \ldots$ are normally distributed with mean $m \in \mathbb{R}$ and variance $s^2 > 0$. Moreover $W, N(t), Y_1, Y_2, \ldots$ are all independent. Calculate the characteristic function

$$\chi_t(u) := E[e^{iuX(t)}].$$

Hints:

(a) *The characteristic function of a normally distributed random variable Y with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 \in \mathbb{R}_+$ is given by* $E[\exp(iuY)] = \exp(iu\mu - u^2\sigma^2/2)$ *for $u \in \mathbb{R}$.*

(b) *It holds that* $\exp(\sum_{j=1}^{N_t} Y_j) = \sum_{m=0}^{\infty} 1_{\{N(t)=m\}} \exp(\sum_{j=1}^{m} Y_j)$ *for any $t \in \mathbb{R}_+$.*

C-Exercise 26 (Power call) (4 points)

The power call with strike K and $p \geq 1$ is an option with payoff $(S(T)^p - K)^+$. In the Black–Scholes model, these were treated in In-Tutorial Exercise 10(ii). Write a Python function

```
Heston_PCall_Lapl(S0, r, gam0, kappa, lamb, sig_tild, T, K, R, p)
```

that computes the fair price of the power call option in the Heston model using the Laplace transform approach. Choose an appropriate R and test your function with the parameters

$$S(0) = [50:150],\ r = 0.05,\ \gamma(0) = 0.3^2,\ \kappa = 0.3^2,\ \lambda = 2.5, \widetilde{\sigma} = 0.2,\ T = 1,\ K = 100,\ p = 1$$

Hint: In the material folder in OLAT you can find a pre-implementation of the characteristic function the log-stock price $X = \log(S)$ in the Heston model in the file `heston_char.py`.
Useful Python commands: `np.exp`, `complex`, `scipy.integrate.quad`, `real`

C-Exercise 27 (Implied volatility in the Heston-model) (4 points)

(a) The Black–Scholes formula establishes a one-to-one correspondence between the volatility σ and the initial call option price $V(0)$ if all other input variables are fixed. Instead of computing the call option price $V(0)$ for a given volatility σ, we can therefore look for an algorithm that computes the volatility σ from a given call option price $V(0)$ by inverting the Black–Scholes formula. To this end, write a Python function

```
sigma = ImpVol(V0, S0, r, T, K)
```

that computes and returns this so-called *implied volatility*. Test your function with

$$S(0) = 100,\ \ r = 0.05,\ \ T = 1,\ \ K = 100,\ \ V(0) = 6.09.$$

Useful Python commands: `scipy.optimize.minimize`

Suppose now you have a function that computes European Call prices in the Heston model

```
Heston_EuCall(S0, r, gam0, kappa, lamb, sig_tild, T, K, R).
```

This is given in In-Tutorial Exercise 13, or you can modify your function from C-Exercise 26.

(b) We now calculate the implied volatility in the Heston model. Write a Python function

```
ImpVol_Heston(V0, S0, r, gam0, kappa, lamb, sig_tild, T, K, R)
```

that first computes call option prices in the Heston model (using the above function `Heston_EuPut`) and then calculates the associated *implied volatility* (using the function from part a). Test your function with the same parameters as in C-Exercise 26 and plot the Heston-implied volatility against the corresponding strike price K.

C-Exercise 28 (Calibration in the Heston-model) (4 points)

Suppose again you have a function that computes European Call prices in the Heston model

```
Heston_EuCall(S0, r, gam0, kappa, lamb, sig_tild, T, K, R).
```
$(*)$

This is given in In-Tutorial Exercise 13, or you can modify your function from C-Exercise 26.

In the material folder in OLAT, you can find a dataset `option_prices_sp500.csv` containing the prices of European call options on the S&P 500 index with one year maturity for 128 different strike prices K on 06 Sept 2022. Our goal is now to calibrate the parameters κ, λ and $\widetilde{\sigma}$ as well as the initial squared volatility $\gamma(0)$ in the Heston model, that is, we want to determine the quadruple $\theta := (\kappa, \lambda, \widetilde{\sigma}, \gamma(0))$ that minimises the sum of squared errors

$$\sum_{j=1}^{128} \left[V_\theta(0, S(0), K_j, T, r) - V_{\text{data}}(0, S(0), K_j, T, r) \right]^2 \quad \longrightarrow \min_\theta,$$

where K_j, $j \in \{1, \ldots, 128\}$ denote the given 128 strike prices, where $V_\theta(t, S(t), K, T, r)$ denotes the European call option price in the Heston model with parameters θ (for example computed using the function $(*)$) and where $V_{\text{data}}(0, S(0), K_j, T, r)$ denotes the given Call option price from the dataset corresponding to the strike price K_j. Write a python function

```
theta = calibrate_Heston(V0_data, S0, r, K, T)
```

that calibrates the parameters $\theta := (\kappa, \lambda, \widetilde{\sigma}, \gamma(0))$ using the procedure described above. Afterwards, plot the option prices from the dataset as well as the Heston call option prices using the calibrated parameters against the strike prices K_j, $j \in \{1, \ldots, 128\}$, in a common figure.

Hint: You may for example use the `scipy.optimize.minimize` routine together with the 'L-BFGS-B' method. This optimisation method lets you specify additional upper and lower bounds for the parameter values that are to calibrate.

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, 20.06.2023, 12:00