Computational Finance and its implementation in Python with applications to option pricing.
Exercise Handout 3

Lecture: Dr. Andrea Mazzon

Exercise 1

Write a class `thetaMethod` which numerically solves the problem

$$\begin{cases} \partial_t u(t,x) - \frac{1}{2}x^2\sigma^2(t,x)\partial_{xx}u(t,x) - rx\partial_x u(t,x) + ru(t,x) = 0 \quad \text{in } (0,T] \times (a,b) \\ u = \psi_1 \quad \text{in } (0,T] \times \{a\}, \\ u = \psi_2 \quad \text{in } (0,T] \times \{b\}, \\ u = \varphi \quad \text{in } \{0\} \times (a,b) \end{cases}$$

providing the following discretization:

$$\begin{aligned} u_j^{n+1} = u_j^n &+ \frac{\theta}{2}x_j^2(\sigma_j^{n+1})^2\frac{\Delta_t}{(\Delta_x)^2}(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}) \\ &+ \frac{1-\theta}{2}x_j^2(\sigma_j^n)^2\frac{\Delta_t}{(\Delta_x)^2}(u_{j+1}^n - 2u_j^n + u_{j-1}^n) \\ &+ \frac{\theta}{2}rx_j\frac{\Delta_t}{2\Delta_x}(u_{j+1}^{n+1} - u_{j-1}^{n+1}) + \frac{1-\theta}{2}rx_j\frac{\Delta_t}{2\Delta_x}(u_{j+1}^n - u_{j-1}^n) \\ &- \theta r\Delta_t u_j^{n+1} - (1-\theta)r\Delta_t u_j^n, \end{aligned}$$

for $\theta \in (0,1)$.
Test its performance for some values of $\theta \in (0,1)$ in the valuation of a down-and-out option. Check that for $\theta = 0$ the scheme is equivalent to Explicit Euler, for $\theta = \frac{1}{2}$ to Crank-Nicolson and for $\theta = 1$ to Implicit Euler.

Exercise 2

Consider again the market where you have zero interest rate $\rho = 0$, and where the risky asset $\bar{S} = (\bar{S}_t)_{t=0,1,\dots,T}$ is defined by

$$\bar{S}_t = \log(S_t),$$

where $S = (S_t)_{t=0,1,\dots,T}$ is the binomial model described in the script, i.e.,

$$S_t = S_0 \cdot Y_1 \cdot \dots \cdot Y_t, \quad t = 1,\dots,T,$$

where Y_t can take the two values d, u with $0 < d < 1 < u$, for any $t = 1,\dots,T$, and $(Y_t)_{t=1,\dots,T}$ are i.i.d. and such that Y_{t+1} is independent of $\mathcal{F}_t$.
In Exercise 2 of Handout 1 you had to find the risk neutral probability Q of such a market, i.e., the probability

$$q = Q(Y_t = u)$$

such that $\bar{S}$ is a martingale under Q, and in Exercise 2 of Handout 2 you had to implement such a model in Python, similarly to what has been done in the classes of

`binomialmodel.creationandcalibration`.

- Valuate now a call option for given parameters under this new model.
- Consider the model above with N time steps, for N large enough, and parameters

$$u = e^{\sigma\sqrt{T/N}}, \qquad d = 1/u, \qquad \rho = e^{r\sqrt{T/N}}. \tag{1}$$

  Find the dynamics

$$dX_t = \mu(t,X_t)dt + \sigma(t,X_t)dW_t, \quad t \geq 0,$$

of a process X which is approximated by $\bar{S}$ under this setting.

Is it possible to compute the analytic value of a call option under these dynamics for X? If yes, check that the price obtained under $\bar{S}$ with parameters (1) for large N approximate this price.