Computational Finance
Computer Lab 15

The aim of the lab is to learn to compute prices of Asian options.
Let us assume that the volatility σ in the Black-Scholes market model depends only on the current stock price S.
Let $v(s, I, t)$ be the function giving the price of an Asian option (depending on arithmetic average) with exercise time T and payoff $p(s, A_T)$. For finding approximate option prices we introduce artificial boundaries I_{max}, S_{max}, choose natural numbers n_s, n_I, m and look for approximate values of v at points (s_i, I_j, t_k), where

$$s_i = i\Delta s = i\frac{S_{max}}{n_s},\ I_j = j\Delta I = j\frac{I_{max}}{n_I},\ t_k = k\Delta t = k\frac{T}{m}.$$

Denote those approximate values by V_{ijk}. A finite difference approximation gives the following equations for the unknown values:

$$a_i V_{i-1,jk} + b_i V_{ijk} + c_i V_{i+1,jk} = \frac{s_i \Delta t}{\Delta I} V_{i,j+1,k} + V_{ij,k+1},$$

where $i = 1, 2, \ldots, n_s - 1,\ j = 0, 1, \ldots, n_I - 1,\ k = 0, \ldots, m-1$ and (using the notation $\rho = \frac{\Delta t}{\Delta s^2}$)

$$\begin{aligned}
a_i &= \frac{\rho}{2}(-s_i^2\sigma^2(s_i) + (r-D)s_i\Delta s),\\
b_i &= \left(1 + \rho s_i^2\sigma^2(s_i) + \frac{s_i\Delta t}{\Delta I} + r\Delta t\right),\\
c_i &= -\frac{\rho}{2}(s_i^2\sigma^2(s_i) + (r-D)s_i\Delta s).
\end{aligned}$$

The equations together with equations for V_{0jk} and $V_{n_s,j,k}$ (specified below) can be viewed as a three-diagonal system for finding the values of $V_{ijk},\ i = 0, \ldots, n_s$, if the values corresponding to the level $t = t_{k+1}$ and the values $V_{i,j+1,k},\ i = 0, \ldots, n_s$ have been found earlier. So we solve the system in backward direction with respect to k (for $k = m-1, m-2, ..., 0$) for each k, we solve it for $j = n_I - 1, n_I - 2, ..., 0$. The values of V_{ijm} can be found from the final condition:

$$V_{ijm} = p(s_i, \frac{I_j}{T}),\ i = 0, \ldots, n_s,\ j = 0, \ldots, n_I.$$

At the boundary $S = 0$ we have the exact boundary condition $v(0, I, t) = p(0, \frac{I}{T})e^{-r\,(T-t)}$, hence

$$V_{0jk} = p(0, \frac{I_j}{T})e^{-r(T-t_k)},\ k = 0, \ldots, m-1.$$

In order to determine all values of V_{ijk} uniquely, we have to specify boundary conditions for the boundary $I = I_{max}$ and the boundary $s = S_{max}$. Denote by $\phi_1(s, I_{max}, t)$ and $\phi_2(S_{max}, I, t)$ the functions describing the boundary conditions, then

$$V_{i,n_I,k} = \phi_1(s_i, I_{max}, t_k),\ V_{n_s,j,k} = \phi_2(S_{max}, I_j, t_k).$$

Exercise 1 Let us consider an average price put option (payoff function $p(s, A_T) = \max(E - A_T, 0)$, where E is the exercise price specified in the option contract). Assume $r = 0.05$, $D = 0$, $\sigma = 0.5$, $T = 0.5$, $E = 100$, $S(0) = 95$. Define $I_{max} = ET$, $S_{max} = 190$, $\phi_1(s, I_{max}, t) = 0$,

$$\phi_2(S_{max}, I, t) = \max\{Ee^{-r(T-t)} + \frac{e^{-D(T-t)}}{(r-D)T}(e^{-(r-D)(T-t)} - 1)S_{max} - \frac{e^{-r(T-t)}}{T}I, 0\}.$$

Write a function that for given values n_s, n_I, m finds an approximate option price at $t = 0$ using the finite difference method described above.

Exercise 2 Keeping a 3-dimensional array of approximate option prices in memory requires a lot of space (RAM). If we need only prices at time $t = 0$, we can avoid storing the full matrix. One possibility is to keep available only values for two time levels - one, which we have already filled with values and the other one with which we currently work. So, inside the for cycle for k we should have $(n_s+1)\times(n_I+1)$ matrix `Vold`, which corresponds to `V[i,j,k+1]` for the current value of k, and a $(n_s+1)\times(n_I+1)$ matrix `Vnew`, corresponding to `V[i,j,k]`, which we fill with computed values step-by-step.

Please modify the solver and the pricing function of the previous exercise so that only 2-dimensional matrices are used.