**Christian-Albrechts-Universität zu Kiel** **SS 2017**
Mathematisches Seminar
Prof. Dr. Mathias Vetter
Ole Martin, Adrian Theopold Sheet 10

# Computational Finance

Exercises for participants of the programme **'Quantitative Finance'**

**C-Exercise 30 (Up and Out Put in the Black-Scholes model)**
In the Black-Scholes model with initial stock price $S(0)$, interest rate $r$ and volatility $\sigma$, let $V(0)$ be the initial price of a European up-and-out call option on the stock $S$ with strike price $K$ and barrier $B > K$. I.e., the option pays off at maturity $T$ the amount

$$V(T) = 1_{\{S(t)<B \text{ for all } t\in[0,T]\}}(K - S(T))^+.$$

Write a scilab function

```
V0 = UpOutPut_BS_MC_Richardson (S0, r, sigma, T, K, B, M, m)
```

that computes the price of an up-and-out call option in the Black-Scholes model via Monte-Carlo simulation using $M$ samples via the Richardson extrapolation. Use $m$ resp. $2m$ points on the coarse resp. on the fine grid for the Euler method.

Test your function for

$$S(0) = 100, \quad r = 0.05, \quad \sigma = 0.2, \quad T = 1,$$
$$K = 100, \quad B = 110, \quad M = 10000, \quad m = 250.$$

and compare the result with C-Exercise 12.

**C-Exercise 31 (Valuation of a European Put using the explicit finite difference scheme)**
Write a Scilab function

```
V0 = BS_EuPut_FiDi_Explicit (r, sigma, a, b, m, nu_max, T, K)
```

that approximates the option values $v(0,x_1),\dots,v(0,x_{m-1})$ of a European put option with strike $K > 0$ and maturity $T > 0$ in the Black-Scholes model using the explicit finite difference scheme. Here, $x_i = K\exp(a + i\frac{b-a}{m})$ denote the initial stock prices and $a, b, m, \nu_{max}$ are the parameters of the algorithm presented in the course. Test your function for

$$r = 0.05, \quad \sigma = 0.2, \quad a = -0.7, \quad b = 0.4, \quad m = 100, \quad \nu_{max} = 2000,$$
$$T = 1, \quad K = 100.$$

Compare your result with the exact solution using the closed formula from the paragraph following (3.23) in the lecture notes.

*This is an exercise taken from last years exam. The exercise should be worked on in the following way:*

- When specifying **modifications of Scilab codes**, use the provided line numbers. E.g., write "insert after line 10: ..." or "replace line 20 by: ...".
- If your modified programs are executed on a (mental) computer and produce the correct number, it's fine. Cosmetic changes of variable or function names, etc. are not necessary.

**T-Exercise 32 (Explicit finite difference scheme for the linear transport equation)**
Consider the heat equation

$$\frac{\partial}{\partial t}y(t,x) = \frac{\partial^2}{\partial x^2}y(t,x) \tag{1}$$

for $t \in [0,10]$ and $x \in [-5,5]$ with the following boundary conditions

$$y(0,x) = e^{-x} \ , \ y(t,-5) = e^{5} \ , \ y(t,5) = e^{-5} \tag{2}$$

In the lecture we discussed the explicit finite difference scheme, which allows us to compute an approximate solution of the latter PDE. The central idea was to discretize equation (1) by

$$w_{\nu+1,i} = w_{\nu,i} + \frac{\Delta t}{\Delta x^2}\left(w_{\nu,i+1} - 2w_{\nu,i} + w_{\nu,i-1}\right) \tag{3}$$

where $w_{\nu,i} := w(t_\nu, x_i)$ denotes the approximation of the function $y$ on the grid

$$\begin{aligned} t_\nu &:= \nu\Delta t \ , & \nu &= 0,...,\nu_{\max} \\ x_i &:= -5 + i\Delta x \ , & i &= 0,...,m \end{aligned}$$

with $\Delta t := 10/\nu_{\max}$ and $\Delta x := 10/m$. The scilab function listed below is an implementation of this finite difference scheme. Tasks:

(a) Show that the derivative of a smooth function $f : \mathbb{R} \to \mathbb{R}$ can be approximated as

$$\frac{f(x+h) - f(x-h)}{2h} = f'(x) + O(h^2).$$

(b) Instead of the heat equation (1) with the boundary conditions (2) we now want to solve the PDE

$$\frac{\partial}{\partial t}y(t,x) = \frac{\partial}{\partial x}y(t,x) \tag{4}$$

for $t \in [0,10]$ and $x \in [-5,5]$ with the boundary conditions

$$y(0,x) = e^{-2x} \ , \ y(t,-5) = e^{10} \ , \ y(t,5) = e^{-10}. \tag{5}$$

How does the explicit scheme (3) change in this case?

(c) Which parts of the source code below have to be changed if we want to solve the PDE (4) with the boundary conditions (5)? Please list all the required modifications and explain why they are necessary.

```
function w = HeatEQ_FiDi_Explicit (m, nu_max)

    w = zeros(nu_max + 1, m + 1);

    // discretization
    delta_t = 10 / nu_max;
    delta_x = 10 / m;
    lambda = delta_t / delta_x^2;
    x = (0:1:m) * delta_x  - 5;

    // boundary conditions
    w(1, :) = exp(-x);
    w(:, 1) = exp(5);
    w(:, m+1) = exp(-5);

    // iterate through time layers
    for nu=1:nu_max
        w(nu+1,2:$-1) =  lambda*w(nu,1:$-2) + (1-2*lambda)*w(nu,2:$-1) + lambda*w(nu,3:$);
    end

endfunction
```

Please save your solution of each C-Exercise in a file named `Exercise_##.sce`, where ## denotes the number of the exercise. Please include your name(s) as comment in the beginning of the file.

**Submit until:** Fri, 07.07.2017, 10:00
**Discussion:** Tue, 10./12.07.2017, 14:15