David Šiška School of Mathematics Sciences, University of Edinburgh

2018/19 Semester 1

Object Oriented Programming with Applications

Problem Sheet 6 - Wednesday 7th November 2018[1]

Exercises marked with an asterisk (*) need to be handed in by noon on 6th November

Black–Scholes PDE

Consider an European type option with a payoff given by a function $g : [0,\infty) \to \mathbb{R}$ (e.g. $g(S) = \max(S-K,0)$ for a call option) and expiry at time $T > 0$. Using the assumptions of Black and Scholes the price $v(t,S)$ of the option at time $t \in [0,T)$ for the price of risky asset S is the solution to

$$\frac{\partial v}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 v}{\partial S^2} + rS\frac{\partial v}{\partial S} - rv = 0 \ \text{ on } \ [0,T) \times (0,\infty) \tag{1}$$

subject to $v(T,S) = g(S)$ for all $S \in (0,\infty)$.

Let $\theta := T-t$ denote the time-to-expiry of the option. Then $v(t,S) = u(T-t,S) = u(\theta,S)$ where u is the unique solution to

$$\frac{\partial u}{\partial \theta} - \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 u}{\partial S^2} - rS\frac{\partial u}{\partial S} + ru = 0 \ \text{ on } \ [0,T) \times (0,\infty) \tag{2}$$

subject to $u(0,S) = g(S)$ for all $S \in (0,\infty)$.

Numerical approximation using finite differences

The first thing one notices when considering a numerical approximation is that $(0,\infty)$ is infinitely long. So even if we discretise derivatives we get a system of infinitely many linear equations to be solved - not something we can ask a computer to do. Thus we must replace $(0,\infty)$ with $(0,R)$ with R large enough. The next question is what would be reasonable boundary conditions on $S=0$ and $S=R$?

From the stochastic representation of the risky asset as geometric brownian motion it can be shown that $\lim_{S\to 0} v(t,S) = e^{-r(T-t)}g(0)$. For large S it makes most sense to say that the option price changes with S like the payoff itself. That is

$$\lim_{S\to\infty} \frac{\partial v}{\partial S}(t,S) = \lim_{S\to\infty} \frac{dg}{dS}(S)$$

for all $t \in [0,T)$ and hence

$$\lim_{S\to 0} u(\theta,S) = e^{-r\theta}g(0), \ \ \lim_{S\to\infty} \frac{\partial u}{\partial S}(t,S) = \lim_{S\to\infty} \frac{dg}{dS}(S) \ \text{ for all } \ \theta \in (0,T].$$

[1]Last updated 7th November 2018

From this we could see that a reasonable approximation to u (denoted u_R) would be the solution to

$$\frac{\partial u_R}{\partial \theta} - \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 u_R}{\partial S^2} - rS\frac{\partial u_R}{\partial S} + ru_R = 0 \text{ on } [0,T)\times(0,R)$$

subject to $u_R(0,S) = g(S)$ for all $S \in (0,\infty)$, $u_R(\theta,0) = e^{-r\theta}g(0)$ and $\frac{\partial u_R}{\partial S}(\theta,R) = \frac{dg}{dS}(R)$ for all $\theta \in (0,T]$.

This equation can be approximated using finite differences. Let $N \in \mathbb{N}$ be given. This will denote the total number of "time steps" used in the approximation. Based on this define $\tau := T/N$. Moreover let $M \in \mathbb{N}$ be given. This will determine the number of partitions of the interval $(0,R)$. Based on this define $h := R/(M-1)$.

Let us introduce, for a function $f = f(\tau, S)$ the following differencing operators:

$$\begin{aligned}
\delta_{-\tau} f(\theta,S) &= \frac{f(\theta,S) - f(\theta-\tau,S)}{\tau},\\
\delta_h f(\theta,S) &= \frac{f(\theta,S+h) - f(\theta,S)}{h},\\
\delta_{-h} f(\theta,S) &= \frac{f(\theta,S) - f(\theta,S-h)}{h},\\
\Delta_h f(\theta,S) &= \frac{f(\theta,S+h) - 2f(\theta,S) + f(\theta,S-h)}{h^2}.
\end{aligned}$$

Let us also define sgn$(r) = 1$ if $r \geq 0$ and sgn$(r) = -1$ if $r < 0$. Then a good approximation of the PDE is

$$\delta_{-\tau} u_{R,\tau,h} - \frac{1}{2}\sigma^2 S^2 \Delta_h u_{R,\tau,h} - rS\delta_{-\text{sgn}(r)h} u_{R,\tau,h} + ru_{R,\tau,h} = 0 \quad \text{on} \quad M_T \tag{3}$$

subject to $u_{R,\tau,h}(0,S) = g(S)$, $u_{R,\tau,h}(\theta,0) = e^{-r\theta}K$ and $\delta_{-h}u_{R,\tau,h}(\theta,R) = \delta_{-h}g(R)$, where

$$M_T := \{(\theta,S) \in (0,T]\times(0,R) : \theta = n\tau,\ n = 1,2,\ldots,N\,,\ S = mh,\ m = 1,2,\ldots,(M-2)\}.$$

Let $U^{n,m} := u_{R,\tau,h}(n\tau, mh)$. Let $G^m := g(mh)$. For $x \in \mathbb{R}$ define $x^+ := \max(x,0) \geq 0$ and $x^- := -\min(x,0) \geq 0$. Then (3) with its boundary constraints and initial condition is equivalent to

$$\begin{aligned}
0 =& \frac{U^{n,m} - U^{n-1,m}}{\tau} - \frac{1}{2}\sigma^2 (mh)^2 \frac{U^{n,m+1} - 2U^{n,m} + U^{n,m-1}}{h^2}\\
&- r^+ mh \frac{U^{n,m+1} - U^{n,m}}{h} + r^- mh \frac{U^{n,m} - U^{n,m-1}}{h}\\
&+ rU^{n,m},\ \ n = 1,\ldots,N\,,\ \ m = 1,\ldots,(M-2)
\end{aligned} \tag{4}$$

subject to $U^{0,m} = g(mh)$ for $m = 1,\ldots,(M-2)$, $U^{n,0} = e^{-n\tau r}K$ and

$$U^{n,M-1} - U^{n,M-2} = G^{M-1} - G^{M-2}$$

for $n = 1,\ldots,N$.

The finite difference based algorithm

To get an implementable algorithm we rewrite (4) in a matrix form. Define a column vector $U^n := (U^{n,m})_{m=0}^{M-1}$ We get that

$$(I + \tau A)U^n = B^{n-1},\ \ n = 1,\ldots,N\,, \tag{5}$$

where

$$A := \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ a(1) & b(1) & c(1) & 0 & \cdots & 0 \\ 0 & a(2) & b(2) & c(2) & \cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & a(M-2) & b(M-2) & c(M-2) \\ 0 & 0 & 0 & \cdots & 1 & 1 \end{pmatrix}, \quad B^{n-1} := \begin{pmatrix} e^{-r\tau n} g(0) \\ U^{n-1,1} \\ U^{n-1,2} \\ \vdots \\ U^{n-1,M-2} \\ \gamma \end{pmatrix} \tag{6}$$

and

$$\begin{aligned} a(m) &:= -\frac{1}{2}\sigma^2 m^2 - r^- m, \\ b(m) &:= \sigma^2 m^2 + r^+ m + r^- m + r, \\ c(m) &:= -\frac{1}{2}\sigma^2 m^2 - r^+ m, \\ \gamma &:= G^{M-1} - G^{M-2}. \end{aligned} \tag{7}$$

Note that A is an $M \times M$ matrix, while B^{n-1} is a vector in $\mathbb{R}^M$ and I is the $M \times M$ identity matrix.

Exercise 6.1. * Implement the scheme given by (5), (6) and (7). Check the correctness of your code by comparing the approximation you get to the put and call prices given by Black–Scholes formula.

Hints:

i) Look at the Lecture 9 slides where there is example for solving the heat equation using finite differences. This task is more or less the same apart from the different matrix A.

ii) You should complete this week's lab by adding this class to the solution you started working on last week. This will allow you to use the Black–Scholes formula you already have to check whether you have correct answers.

Take the following steps:

1. Create a class

```
public class BlackScholesFiniteDifferenceSolver
{
    // ... private member variables of your choice
    // useful constructor
    public BlackScholesFiniteDifferenceSolver(double maturity,
        Func<double, double> payoffFunction,
        double riskFreeRate,
        double sigma,
        double R,
        uint numTimeSteps,
        uint numSpacePartitions)
    {
        // ... constructor code
    }

    // ... some private methods may be useful

    public double Price(double S)
    {
```

```
            // ... solve the finite difference approximation
            // and provide the option price if the underlying is S
            // use linear interpolation if S is not one of the grid points
        }
    }
```

2. Test it by checking convergence to the price given by Black–Scholes formula. Use the following code:

```
public static void Main (string[] args)
{
    // Model params
    double r = 0.05;
    double sigma = 0.1;
    double K = 100;
    double T = 1;
    double S0 = 100;
    double bsPrice = 0; // fill in your code to calculate the put price from the BS formula
    Func<double, double> putPayoff = (S) => Math.Max(K - S, 0);

    uint N, M;

    int numberRefinments = 5;

    // test convergence w.r.t. number of partitions of space interval
    N = 200; M = 100;
    for (int refinementIndex = 0; refinementIndex < numberRefinments; ++refinementIndex,M *= 2)
    {
        BlackScholesFiniteDifferenceSolver solverForThisLevelOfRefinement =
            new BlackScholesFiniteDifferenceSolver (T, putPayoff, r, sigma, 5 * K, N, M);
        double error = Math.Abs (bsPrice - solverForThisLevelOfRefinement.Price (S0));
        Console.WriteLine ("Space partitions: {0}, time steps: {1}, error: {2}", M, N, error);
    }

    // test convergence w.r.t. number of time steps
    N = 10; M = 8001;
    for (int refinementIndex = 0; refinementIndex < numberRefinments; ++refinementIndex, N*=2)
    {
        BlackScholesFiniteDifferenceSolver solverForThisLevelOfRefinement =
            new BlackScholesFiniteDifferenceSolver (T, putPayoff, r, sigma, 5 * K, N, M);
        double error = Math.Abs (bsPrice - solverForThisLevelOfRefinement.Price (S0));
        Console.WriteLine ("Space partitions: {0}, time steps: {1}, error: {2}", M, N, error);
    }
    Console.WriteLine ("Finished. Press any key.");
    Console.ReadKey ();
}
```

Test parameters and output values.
The test parameters are in the block of code above.

Using $R = 3K$, $K = 100$, $\sigma = 0.1$, $r = 0.05$ and $M = 10$ the matrix A should be:

```
SparseMatrix 10x10-Double 27.00% Filled
     1       0        0       0        0       0        0       0        0       0
-0.005    0.11   -0.055       0        0       0        0       0        0       0
     0   -0.02     0.19   -0.12        0       0        0       0        0       0
     0       0   -0.045    0.29   -0.195       0        0       0        0       0
     0       0        0   -0.08     0.41   -0.28        0       0        0       0
     0       0        0       0   -0.125    0.55   -0.375       0        0       0
     0       0        0       0        0   -0.18     0.71   -0.48        0       0
     0       0        0       0        0       0   -0.245    0.89   -0.595       0
     0       0        0       0        0       0        0   -0.32     1.09   -0.72
     0       0        0       0        0       0        0       0        1       1
```

You should be seeing that the error roughly halves every time we double the number of space partitions (provided the number of time steps is large enough).

You should see the error halve each time you double the number of time steps (provided the number of space partitions is large enough).

```
Space partitions: 100, time steps: 200, error: 0.4226
Space partitions: 200, time steps: 200, error: 0.2117
Space partitions: 400, time steps: 200, error: 0.1047
Space partitions: 800, time steps: 200, error: 0.0406
Space partitions: 1600, time steps: 200, error: 0.0213
```

```
Space partitions: 8001, time steps: 10, error: 0.0259
Space partitions: 8001, time steps: 20, error: 0.0076
Space partitions: 8001, time steps: 40, error: 0.0025
Space partitions: 8001, time steps: 80, error: 0.0009
Space partitions: 8001, time steps: 160, error: 0.0051
```