David Šiška School of Mathematics Sciences, University of Edinburgh

2018/19 Semester 1

Object Oriented Programming with Applications

Problem Sheet 5 - Wednesday 24th October 2018[1]

Black–Scholes and Monte Carlo methods in C#

Please read everything including *all* the exercises *before* you start writing any code. Make sure you look at the section "Suggested project structure" before writing any code.

Very brief overview of option pricing in Black–Scholes framework

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be some probability space and $W = W(t)_{t\in[0,T]}$ a Wiener process. Let $(\mathcal{F}_t)_{t\in[0,T]}$ be the filtration generated by W.
Assume that there is a "risk-free" asset with its evolution governed by the differential equation

$$dB(t) = rB(t)dt\,, \quad B(0) = 1,$$

for some constant $r \in \mathbb{R}$. Assume that there is a risky asset with its evolution governed by the stochastic differential equation

$$dS(t) = \mu S(t)dt + \sigma S(t)dW(t)\,, \quad S(0) = S_0,$$

for some constant $\mu \in \mathbb{R}$ and $\sigma > 0$. Assume that one can buy and borrow arbitrary amounts of both the risky and risk-free asset. Assume that there are no transaction costs or bid-ask spreads. Assume that one's trading does not influence the market prices. Consider some T-maturity "contingent claim" ξ (which we assume to be an $\mathcal{F}_T$-measurable and square integrable random variable). It can be shown that there is a unique probability measure $\mathbb{Q}$, equivalent to $\mathbb{P}$, and that the price $v(t)$ of this contingent claim is

$$v(t) = \mathbb{E}^{\mathbb{Q}}\left[e^{-r(T-t)}\xi|\mathcal{F}_t\right]. \tag{1}$$

Moreover there is a stochastic process $W^{\mathbb{Q}}$ which is a Wiener process under $\mathbb{Q}$ and the risky asset dynamics under $\mathbb{Q}$ are given by

$$dS(t) = rS(t)dt + \sigma S(t)dW^{\mathbb{Q}}(t)\,, \quad S(0) = S_0. \tag{2}$$

Finally, if $S(t) = S$ and $\xi = \max(S(T) - K, 0)$ (i.e. the claim is simply a call option) then

$$c(t) = \mathbb{E}^{\mathbb{Q}}\left[e^{-r(T-t)}\max(S(T) - K, 0)|S(t) = S\right] \tag{3}$$

and v is given by the Black–Scholes formula

$$c(t) = SN(d_1) - Ke^{-r(T-t)}N(d_2),$$

where $x \mapsto N(x)$ is the distribution function of standard normal random variable and where

$$d_1 = \frac{\ln\frac{S}{K} + \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}}, \quad d_2 = d_1 - \sigma\sqrt{T-t}.$$

[1] Last updated 1st October 2018

Exercise 5.1. Implement a C# class for pricing European call and put options. For call options use the formula. For put options use put-call parity. Let $c(t), p(t)$ denote price of a call / put options respectively. Recall that, due to put-call parity:

$$p(t) - c(t) = e^{-r(T-t)}K - S(t).$$

Hint 1. To evaluate $x \mapsto N(x)$ you can do one of the following:

1. Use your composite integrator from previous workshop.
2. Use the approximation given on p. 34 of *C# for Financial Markets*.
3. The simplest is to use the MathNet.Numerics library:

```
double N_of_x = MathNet.Numerics.Distributions.Normal.CDF(0, 1, x);
```

Hint 2. You will notice that the Black–Scholes formula only depends on the "time-to-maturity" $T - t$. So without loss of generality you may remove t by assuming it is 0 and that T now represents this time-to-maturity.
Test parameters and output values.

Ass. Pr. S	Mod. r.-f. rate r	Mod. vol. σ	Tm. to Mat. T	Strk. K	Call Pr. $c(0)$	Put Pr. $p(0)$
100	0.05	0.1	1	100	6.805	1.928
100	0.00	0.1	1	100	3.989	3.989
100	0.05	0.2	1	100	10.449	5.572
100	0.05	0.1	10	100	39.939	0.592
100	0.05	0.1	1	120	0.462	14.610

Exercise 5.2. Write a function that will calculate the "implied volatility" given a market price of a call (or put option). This is the σ that needs to be inserted into Black–Scholes formula so that the price given by the formula matches the given market price.
Hint. You will need to use the Newton–Solver from previous workshop.
Test parameters and output values. Here Fix the following: $S = 100$, $r = 0.05$, $T = 1$ and $K = 100$ and check whether you are getting the values below:

Call price $c(0)$	Put price $p(0)$	Start. Guess Solver	Implied vol. σ
10		0.5	0.19
	3	0.5	0.13

Monte Carlo methods in C#

Exercise 5.3. Pricing options where there is no closed-form formula in the Black–Scholes world can be done using Monte–Carlo methods. But it might be useful to try a Monte–Carlo algorithm with a simple call option.
The solution to (2), which can be obtained using Itô's formula, is:

$$S(T) = S(t) \exp\left(\left(r - \frac{1}{2}\sigma^2\right)(T-t) + \sigma[W^{\mathbb{Q}}(T) - W^{\mathbb{Q}}(t)]\right).$$

Then (3) becomes

$$\begin{aligned} c(t) &= \mathbb{E}^{\mathbb{Q}}\left[e^{-r(T-t)} \max\left(S(t)\exp\left(\left(r - \frac{1}{2}\sigma^2\right)(T-t) + \sigma(W^{\mathbb{Q}}(T) - W^{\mathbb{Q}}(t)\right) - K, 0\right)\right] \\ &= \mathbb{E}^{\mathbb{Q}}\left[e^{-r(T-t)} \max\left(S(t)\exp\left(\left(r - \frac{1}{2}\sigma^2\right)(T-t) + \sigma\sqrt{T-t}\xi\right) - K, 0\right)\right], \end{aligned}$$

where $\xi \sim N(0,1)$.

A Monte–Carlo approximation to $c(t)$ using N independent samples from $N(0,1)$ denoted $(\xi^{(i)})_{i=1}^{N}$ is

$$c_N(t) := \frac{1}{N}\sum_{i=1}^{N}\left[e^{-r(T-t)}\max\left(S(t)\exp\left(\left(r-\frac{1}{2}\sigma^2\right)(T-t)+\sigma\sqrt{T-t}\xi^{(i)}\right)-K,0\right)\right].$$

Your task is to implement C# class that will calculate this approximation.
Hints. You can get a sample from Normal distribution using

```
using MathNet.Numerics.Distributions;
// ...
     double xi = Normal.Sample(0, 1);
```

Or, if you want $N \in \mathbb{N}$ samples then you can get them all in one go:

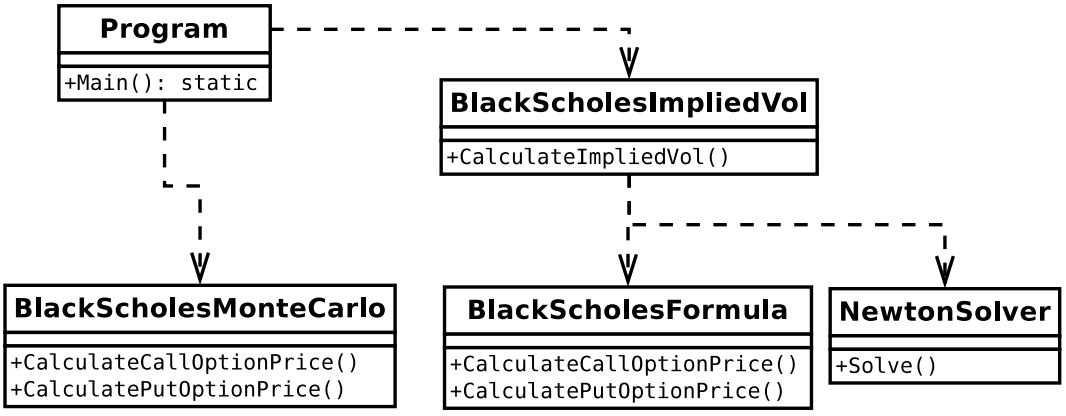
```
int N = 10000;
double[] Z = new double[N];
Normal.Samples(Z, 0, 1); // will fill Z with samples from standard normal
```

You will need to take N of about 10^4 to get a decent approximation.
Test parameters and output values: Your Monte-Carlo code should give approximately the same values as your Black–Scholes formula for call option prices. In particular you should see the error halve every time you quadruple N.
You will also see the exact error change every time you run your code because you are using random samples. To make comparisons easier run your code 1000 times with each N samples you are interested in and compare each run to the Black–Scholes formula $c(0)$.
Fix the following: $S = 100$, $r = 0.05$, $\sigma = 0.1$, $T = 1$ and $K = 100$ and check whether you are getting roughly the values below:

Num. Samples N	Average error with 1000 runs of N
100	0.61
400	0.31
1600	0.15
6400	0.08
25600	0.04

Suggested project structure

You will see that the first three exercises are closely related. You should keep them inside one *project* in your *solution*. A possible arrangement of the functionality of the above exercises into classes is the following:



- Each box indicates a class and should be in a separate `.cs` file.
- You can see the suggested public methods each class should have; you may have more private methods and private variables inside the class.

- The diagram doesn't indicate whether the methods should be static or not and what input arguments they should have. This is for you to decide.
- The dashed arrows indicate dependency. To complete Exercise 5.2. your `Main` method should use the `BlackScholesImpliedVol` class and the `BlackScholesMonteCarlo` class. You see that the `Program` class does not need to use `NewtonSolver` directly.

Exercise 5.4. Now we will use Monte–Carlo method to price the Asian arithmetic option. Such options have the following payoff: given a set of dates

$$T_1 \leq T_2 \leq \cdots \leq T_M \leq T$$

and a strike K the option payoff at time T is given by

$$\xi = \max\left(\frac{1}{M}\sum_{m=1}^{M} S(T_m) - K, 0\right).$$

One can then use the solution to (2) - which can be obtained using Itô's formula - and which is:

$$S(T_m) = S(T_{m-1}) \exp\left(\left(r - \frac{1}{2}\sigma^2\right)(T_m - T_{m-1}) + \sigma(W^{\mathbb{Q}}(T_m) - W^{\mathbb{Q}}(T_{m-1}))\right)$$

for $m = 1, \ldots, M$ with the convention that $T_0 := t$ which is the current time at which $S(t) = S$. Thus from (1) we get

$$\begin{aligned} v(t) &= e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}\left[\max\left(\frac{1}{M}\sum_{m=1}^{M} S(T_m) - K, 0\right)\middle| S(t) = S\right] \\ &= e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}\left[\max\left(\frac{1}{M}\sum_{m=1}^{M} S(T_{m-1}) e^{\left(r-\frac{1}{2}\sigma^2\right)(T_m - T_{m-1}) + \sigma\sqrt{T_m - T_{m-1}}\xi_m} - K, 0\right)\right], \end{aligned}$$

where ξ_m are M independent $N(0,1)$ random variables.
For Monte Carlo approximation with N samples we need $(\xi_m^i)_{i=1}^N$ for each $m = 1, \ldots, M$ where each x_m^i is an independent sample from $N(0,1)$. Let $(s_m^i)_{i=1}^N$ denote samples from the law of $S(T_m)$. We can get these iteratively (recall $S(t) = S$):

$$s_1^i = S \exp\left(\left(r - \frac{1}{2}\sigma^2\right)(T_1 - T_0) + \sigma\sqrt{T_1 - T_0}\xi_1^i\right)$$

for $i = 1, \ldots, N$ while

$$s_m^i = s_{m-1}^i \exp\left(\left(r - \frac{1}{2}\sigma^2\right)(T_m - T_{m-1}) + \sigma\sqrt{T_m - T_{m-1}}\xi_m^i\right).$$

with $m = 2, \ldots, M$ and $i = 1, \ldots, N$.
Finally the approximation to $v(t)$ obtained using N samples is

$$v_N(t) = e^{-r(T-t)}\frac{1}{N}\sum_{i=1}^{N}\left[\max\left(\frac{1}{M}\sum_{m=1}^{M} s_m^i - K, 0\right)\right].$$

Your task is to implement C# class that will calculate this approximation.
Test parameters and output values: This one is the hardest to test due to the sheer number of inputs.
Take $T_1 = 1/4$, $T_2 = 1/2$ and $T_3 = 1$ with $T = 1$. Take $r = 0.05, \sigma = 0.1$, $S = 100$, $K = 100$. Take $N = 10^5$. Then the price of the Asian Arithmetic Call should come near 2.87.