C++ in Quantitative Finance II

Labs #1

Implementing the pay-off class

Paweł Sakowski

Spring 2026

University of Warsaw
Faculty of Economic Sciences

Grading policy in the Spring Term

- final exam 30 pts.
- home taken project with oral defence 70 pts.
- 100 pts. to collect in total
- minimum requirement to pass:
  1. at least 60 pts. in total
  2. and at least 15 pts. from the final exam,
  3. and at least 35 pts. from the project
- Attendance is mandatory, max three unexcused absences are allowed.
- The minimum threshold will be lowered to 50 pts. for active students.
- Additional homeworks, exercices, etc. are not mandatory, but can help later to gain additional points for credit.

Literature

- Joshi (2008) C++ Design Patterns and Derivatives Pricing

Technical elements of code

Today we will use:

- pointers and address references
- variables of `enum` type
- `switch` case structure
- classes, with their members (variables) and methods (functions)
- constructors for classes
- functors, ie. classes that behave like functions

Key concepts for today

- We'll look at one way of using a class to encapsulate the notion of an option pay-off.
- Using a pay-off class allows us to add extra forms of pay-offs without modifying our Monte Carlo routine.
- By overloading `operator()` we can make an object look and behave like a function.
- `const` enforces extra discipline by forcing the coder to be aware of which code is allowed to change things and which code cannot.
- `const` code can run faster.
- The **open-closed** principle says that code should be open for extension but closed for modification.
- Private data helps us to separate interface from implementation.

Stock price motion in Monte Carlo methods I

Price of the underlying asset is described by:

$$dS_t = \mu S_t dt + \sigma S_t dW_t \tag{1}$$

and a continuously compounding risk-free rate r.
From the BS we know that the price of a vanilla option, with expiry T and pay-off f, is equal to

$$e^{-rT}\mathbb{E}(f(S_T)) \tag{2}$$

where the expectation is calculated with respect to the risk-neutral process

$$dS_t = rS_t dt + \sigma S_t dW_t \tag{3}$$

Stock price motion in Monte Carlo methods II

By passing to the log and using Ito's lemma we can solve Eq. (3)

$$d \log S_t = \left(r - \frac{1}{2}\sigma^2 \right) dt + \sigma dW_t \tag{4}$$

which has the solution

$$\log S_t = \log S_0 + \left(r - \frac{1}{2}\sigma^2 \right) t + \sigma W_t \tag{5}$$

Stock price motion in Monte Carlo methods III

Since W_t is a Brownian motion, W_T is distributed as $N(0,T)$ and we can write

$$W_T = \sqrt{T}N(0,1) \tag{6}$$

which results in

$$\log S_T = \log S_0 + \left(r - \frac{1}{2}\sigma^2\right)T + \sigma\sqrt{T}N(0,1) \tag{7}$$

or equivalently

$$S_T = S_0 e^{(r-\frac{1}{2}\sigma^2)T+\sigma\sqrt{T}N(0,1)} \tag{8}$$

The price of a vanilla option is therefore given by

$$e^{-rT}\mathbb{E}\left(f(S_0 e^{(r-\frac{1}{2}\sigma^2)T+\sigma\sqrt{T}N(0,1)})\right) \tag{9}$$

Stock price motion in Monte Carlo methods IV

This expectation is approximated by Monte Carlo simulation. From the law of large numbers we know that if Y_j are a sequence of identically distributed independent random variables, then with probability 1 the sequence

$$\frac{1}{N}\sum_{j=1}^{N} Y_j \tag{10}$$

converges to $\mathbb{E}(Y)$.

Stock price motion in Monte Carlo methods V

The algorithm of Monte Carlo method

1. Draw a random variable $x \sim N(0,1)$ and compute

$$f(S_0 e^{(r-\frac{1}{2}\sigma^2)T+\sigma\sqrt{T}x}) \tag{11}$$

   where for European call $f(S) = (S-K)_+$.
2. Repeat this possibly many times and calculate the average.
3. Multiply this average by e^{-rT}.

Exercises I

1. Modify `project01` to obtain price of:
   - European put price,
   - digital option,
   - double digital option.
2. Set a random-like seed by adding the following code at the beginning of `main()`:

   ```
   srand(time(NULL));
   ```

   Do not forget to `#include` two external header files:

   ```
   #include <cstdlib>
   #include <ctime>
   ```

   After this, every time you run a simulation you should get different approximation of the theoretical option price. Is the precision of our option prices acceptable?

Exercises II

3. In `project02`, modify the pay-off class so that it can handle squared power options, with the payout $\max^2[S_T - K, 0]$ for the call and $\max^2[K - S_T, 0]$ for the put.
4. In `project02`, modify the pay-off class so that it can handle squared power options, with the payout $\max[S_T^2 - K^2, 0]$ for the call and $\max[K^2 - S_T^2, 0]$ for the put.
5. In `project02`, modify the pay-off class so that it can handle digital options.
6. In `project02`, modify the pay-off class so that it can handle double digital options.

Thank you!

Paweł Sakowski
sakowski@wne.uw.edu.pl