QFGB8960 Advanced C++ for Finance
Homework 9

Spring 2025

Problem 1 (50 points) Antithetic Sampling

a) In a new file `qflib/methods/montecarlo/antitheticpathgenerator.hpp` implement the class `AntitheticPathGenerator : public PathGenerator`.

The constructor of the new class should take a smart pointer to `PathGenerator`. Internally, the class should have a member variable of type `SPtrPathGenerator`, which stores the pointer to `PathGenerator` passed in the constructor. This allows the `AntitheticPathGenerator` to add antithetic sampling to any existing path generator.

In its `next()` method it should return a newly generated price path in the first call and the antithetic pair of the already generated path in the second call. The class should keep track inside when to generate a new path. This can be done with a Boolean switch which is flipped to true/false when a new path or the antithetic of the prior path is needed. The `AntitheticPathGenerator` class should store internally the generated path, so it can return its antithetic next time it is called. The generation of the new path should be delegated to the internal pointer of the inner generator.

b) Extend the `McParams` structure to contain an enumeration of the form

```
enum class ControlVarType { NONE, ANTITHETIC };
```

and a field

```
ControlVarType controlVarType;
```

Extend the ctor of `McParams` to accept an extra argument of type `ControlVarType` and store it in the `controlVarType` field.

c) In the existing file `qflib/pricers/bsmcpricer.cpp` add the case when the `mcparams.controlVariate` is equal to ANTITHETIC in which case a smart pointer to an `AntitheticPathGenerator` should be created.

d) In the existing file `pyqflib/pyutils.cpp`, extend the function `asMcParams()` to recognize the name "ControlVarType" and the values "NONE" or "ANTITHETIC" and set the `mcparams` field accordingly.

e) Create a Python notebook, and set uyp the pricing a two-year European ITM put (spot=100 strike=120) using EULER path generator with MT19937 random number generator.Use a dividend yield of 0%, volatility of 60% and a yield curve constructed from the following spot rates

Maturity	SpotRate
1/12	1.00%
1/4	2.00%
1/2	3.00%
3/4	3.50%
1	4.00%
2	4.80%
3	5.30%
5	6.00%

Price the call with 512, 1024, 2048, ... 131072, 262144 MC paths, without and with antithetic control variate.

Create two plots, one with the prices and one with the standard errors. Each plot should display the data without and with antithetic control variate, as a function of the number of paths. Use logarithmic scale for the x-axis (number of MC paths).

Problem 2 (50 points) Worst-of Digital Call/Put on M Assets

a) In a new file `qflib/products/worstofdigitalcallput.hpp` implement the payoff of a digital call or put on the worst performing asset in a basket of m assets. The product has two observation times, the fixing (strike) time T_1 and the expiration/payment time T_2. The pricing time is assumed to be $t = 0$. The payoff for the put is

$$V_p(T_2) = \Theta\left(K - \min_{j=1\ldots m} \frac{S_j(T_2)}{S_j(T_1)}\right), \tag{1}$$

and for the call

$$V_c(T_2) = \Theta\left(\min_{j=1\ldots m} \frac{S_j(T_2)}{S_j(T_1)} - K\right). \tag{2}$$

with $\Theta(x)$ the Theta (step) function.

b) Implement the C++ function `pyQfWorstOfDigiBSMC` in the file `pyqflib/pyfunctions3.hpp`, and expose it to Python as `qf.wostofDigiBSMC`. This new function should have the same input parameters as the function `qf.asianBasketBSMC` with two exceptions:

1. Instead of the `fixingTimes` array, pass two scalar parameters, `timeToFix` and `timeToExp`,
2. Do not pass an `assetquantities` parameters, it is not needed,

c) On the notebook used for problem 1, price an ATM ($K = 100\%$) worst-of digital call and put on three assets, with the same pairwise correlation $\rho = 50\%$. Set the fixing time to one year and the expiration time to two years. Use the yield curve of the previous problem, 0% dividend yield and 30% volatility for all three assets. Price the worst-of options with Monte Carlo, using EULER path generator with MT19937 random number generator. Verify that the sum of the digital call and put price is equal to the present value of 1 payed at expiration, independent of strike.