QFGB8960 Advanced C++ for Finance
Homework 10

Spring 2025

Problem 1 (40 points) PDE Pricing with Volatility Term Structure

a) Change the `PdeBase` class so that it stores a vector of smart pointers to objects of type `VolatilityTermStructure` as

```
std::vector<SPtrVolatilityTermStructure> vols_;
```

`SPtrVolatilityTermStructure` is a convenient alias for

`std::shared_ptr<orf::VolatilityTermStructure>`.

The vector `vols_` should contain as many smart pointers as the number of assets, i.e. one volatility term structure for each asset.

Update the methods `PdeBase::initGrid` and `PdeBase::solve` so that they use the term structure object in `vols_` for computing spot and forward volatilities.

Update the `Pde1DSolver` constructor to accept `SPtrVolatilityTermStructure`.

b) Modify the Python function `qf.euroBSPDE` to accept either a scalar value or an object handle for the volatility term structure. When the input is scalar, a constant volatility term structure should be created on the fly and passed to the `Pde1DSolver` constructor.

c) Create a Python notebook. Construct a flat 4% yield curve and a non-flat volatility term structure as named objects. Make the spot volatility at one year be equal to 40% and at two years to 60%.

Then perform PDE pricing of an ATM one year put that uses the yield curve and volatility handles from the market. Use spot = strike = 100 and zero dividend yield. Compare the PDE price from using the volatility handle with the PDE price using the equivalent numerical value of the volatility to expiration. Use a 800x800 grid and two schemes, implicit ($\theta = 1$) and Crank-Nicholson (θ =0.5).

Problem 2 (20 points) Calendar Spread in PDE

A long calendar spread is a combination of a short position in a call with strike K and expiration T_1, and long position in a call with the same strike K and expiration T_2 with. $T_2 > T_1$. Since option prices are increasing functions of time to expiration, the calendar spread must have a non-negative price.

On the notebook of problem 1, price a calendar spread with strike $K = 100$, $T_1 = 1$ year and $T_2 = 2$ years, using the yield curve and volatility term structure of problem 1. Compute the price using PDE (`qf.euroBSPDE`) on a 800x800 grid, Crank-Nicholson (θ =0.5), for two values of the dividend yield $q = 2\%$ and $q = 0\%$. Use a range of spots 50, 60, 70, ...,180, 190, 200. Plot the calendar spread price as a function of spot for the two dividend yield values. Note how the increasing dividend yield asymmetrically suppresses the calendar spread price.

Problem 3 (40 points) Digital PDE Convergence Graph

a) Create a new product header file `qflib/products/digitalcallput.hpp`, with all the necessary methods for a digital call or put payoff.

b) Implement the function `pyQfDigiBSPDE` in the file `pyqflib/pyfunctions4.cpp` and register it as `qf.digiBSPDE` in the file `pyqrflib/qflib/__init__.py`. This is the PDE pricer for a digital call or put option.

c) On the notebook of the previous problem, price a one year ATM digital put with spot = strike = 100, using the same yield curve and volatility as in problem 1, and zero dividend yield.

Use the Crank-Nicholson scheme ($\theta = 1/2$) and compute the price for a sequence of grid refinements, i.e. for

Refinement	NTimeSteps	NSpotNodes
1	25	25
2	50	50
3	100	100
4	200	200
5	400	400
6	800	800

Compute the pricing error as: Error = PDE_Price - BS_Price

Plot the error as a function of grid refinement, i.e. put the refinement index in the x-axis and the error in the y-axis.

Since at each refinement we increase the number of time and spot points by two, the error follows linear (quadratic) scaling if the error at refinement $i + 1$ is one half (one quarter) the error at refinement i. Does the error on the plot follow linear or quadratic scaling?