# QFGB8960 Advanced C++ for Finance
# Homework 5

Spring 2025

## Problem 1 (50 points) P&L Accounting

At time $t = 0$ we sold $N = 10,000$ units of a call option with strike $K = 100$ and time to expiration $T = 2$ years. Every quarter year we delta hedge our option position, i.e. we buy/sell stock so that we hold $N\Delta_t$ shares of the underlying stock.

Create a Python notebook with a pandas dataframe that at each quarter until expiration tracks the positions and values of a portfolio containing the short option position, the stock shares (delta hedge), and a cash account for maintaining our debits/credits. To compute the value $V_t$ and the delta $\Delta_t$ use the function `qf.euroBS` from homework 4.

For valuation of the option and cash account use the parameters: $r = 4\%$, $q = 2\%$, $\sigma = 30\%$.

The stock prices at each quarter year are given in the column "Spot" in the table below. In each row we compute the option price $V_t$, delta $\Delta_t$, and the one-period growth factor $M_t = e^{r(t_i - t_{i-1})}$. We hold $-N$ option units (we sold, so we are short) and $N\Delta_t$ stock units as a hedge. To buy the initial hedge, which is worth $N\Delta_0 S_0$, we need to borrow this amount from our cash account. Since we deposited the premium $NV_0$ into the account, the initial cash balance is $-N\Delta_0 S_0 + NV_0$. At the end of the following period, $t = 0.25$, we adjust the hedge from $N\Delta_{t=0}$ shares to $N\Delta_{t=0.25}$ shares. This will cost $(N\Delta_{t=0.25} - N\Delta_{t=0})\, S_{t=0.25}$. The cash account at the end of $t = 0.25$ will contain the prior balance, grown by the factor $M_{t=0.25}$, plus the cost from adjusting the delta hedge.

Use the above procedure to fill in the rest of the table below.

For every time $t$ we have: Tot_Val$_t$ = Opt_Val$_t$ + Stk_Val$_t$ + Cash_Acc$_t$

Report the final P&L, i.e. the total value Tot_Val$_{t=2.00}$.

Submit the Python notebook with the above table filled out as a Pandas data frame.

| Time | Spot | Price | Delta | Growth | Opt_Units | Stk_Units | Opt_Val | Stk_Val | Cash_Acc | Tot_Val |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.00 | 100.00 | 17.78 | 0.60 | 1.0000 | -10,000 | 5,960.30 | -177,773.03 | 596,030.41 | -418,257.38 | 0.00 |
| 0.25 | 105.00 | 19.72 | 0.64 | 1.0101 | -10,000 | | | | | |
| 0.50 | 110.00 | | | | | | | | | |
| 0.75 | 92.00 | | | | | | | | | |
| 1.00 | 103.00 | | | | | | | | | |
| 1.25 | 90.00 | | | | | | | | | |
| 1.50 | 110.00 | | | | | | | | | |
| 1.75 | 95.00 | | | | | | | | | |
| 2.00 | 88.00 | | | | | | | | | ???? |

## Problem 2 (30 points) PPoly Curve Operations

Implement a Python callable C++ function that adds two piecewise polynomial curves. Call the C++ function `pyQfPPolySum` and the corresponding Python function `qf.ppolySum`. The singature of the function should be

```
qf.ppolySum(bkpoints1, values1, bkpoints2, values2, porder)
```

The inputs define two piecewise polynomial curves of the same order. Restrict the order to be either 0 or 1.

The function returns a $n \times 2$ numpy array. The first column contains the $n$ common breakpoints of the two curves and the second contains the corresponding values.

## Problem 3 (20 points) PPoly Curve Check

On the same Python notebook as for problem 1, validate the results of the function `qf.ppolySum` in problem 2 by calling it on two polynomial curves with common order = 0. Then use the function `qf.ppolyEval` to evaluate the sum of the two polynomials for a sequence of x-values, which should include the common breakpoints. Plot the original two polynomials and the sum. Does the plot confirm that the sum was correctly computed?

- Put your Python notebook in the folder qflib-0.5.0/examples/Python.