**Assignment 1 Dynamic Hedging**

# **Introduction**

Dynamic hedging is a risk management often used in the option market, where the prices of option contracts are highly sensitive to changes in the underlying assets. The hedge position is adjusted as the price of the underlying asset changes, so that the overall risk remains at an acceptable level. This requires ongoing monitoring and re-adjustment of the hedge positions, which is why it is referred to as dynamic hedging.

In this assignment, we will act as an option maker, we sell put option at day 0, the price is higher than the fair value of the option. As long as option maker can hedge this position perfectly, he is guaranteed to lock in the premium. However, a perfect hedge is hard to achieve. We will try to make delta neutral at all time, and evaluate the hedging performance.

There are 3 main objectives in this assignment:

1. Understand dynamic hedging in practice.
2. Evaluate hedging performance, by changing rebalance frequency and volatility.
3. Enhance coding skill: vectorized programming with numpy array, simple class to enhance reusability of the code.

Note, in python main method, we often need to comment out (disable) some code block, and uncomment (enable) some other blocks, in order to test different functionalities.

# **Task 1: Black Scholes class**

In *DynamicHedge.py*, in the main function, by default *test\_bs* function is enabled, and scripts after that are disabled.

Implement **class BlackScholes** (constructor is given) in *BlackScholes.py*, 6 functions in all, *d1, d2, price, delta, gamma, vega, theta* function. Black Scholes analytical formula can be found either in notes or John Hull’s book.

This class supports vectorized programming, which means spot price is a numpy array, and all functions above should return a numpy arra. Thus, you should use *np.log, np.exp* function when operating on numpy array.

Pay attention to the constructor, “if” statement below is to handle a scalar input, we wrap it into a numpy array.

if not isinstance(s, np.ndarray):  
 self.s = np.array([s])

After you implement the class, *test\_bs* function will print out output. You may compare it with an online calculator to make sure the implementation is correct.

<https://goodcalculators.com/black-scholes-calculator/>

# **Task 2: simulate share price**

Enable the next coding block until the line *test\_share\_distribution*. In risk neutral world, stock price follows lognormal distribution.

Derive and

In *Utilities.py*, Complete function *simulate\_share\_price*, output “simu\_prices” is passed by reference, a 2 dimensional numpy array, time steps \* the number of paths.

After implementation, test\_share\_distribution will print mean and variance of simulated share prices and compare it with the theoretical value that you derived earlier.

The first 2 tasks offer you **good practice of coding**: break down a big task into smaller pieces: functions or classes. Perform functional test as early as possible.

# **Task 3: hedge performance evaluation**

This is core block of this assignment. Hedging performance by simulation is defined as standard deviation of hedging cost divided by initial option price by Black Scholes model. Per each simulation path, we can compute hedging cost in this way:

Change option spot price and strike both to 48, and is\_call parameter set to false, enable the last coding block. In *Utilities.py*, complete function *compute\_hedge\_costs*  , then we can print statistics from function *process\_hedge\_costs.*

We hedge the short put option periodically (by bulling or selling stocks), and keep delta to 0 after every portfolio rebalance. Assume we hold option until expiry date, exercise if it is in the money, expires with 0 value otherwise. We compute hedge ratio by changing rebalance frequency (in weeks), 0.25, 0.5, 1, 2, 4. Report hedge ratio and standard deviation.

Next evaluate the hedging performance at week 12, report the same statistics as last step. Compare hedge ratio and standard deviation, explain the findings.

Secondly, assume we hold option until expiry date, and rebalance frequency is 1 week. Yet in input parameters region, set *simu\_vol* to different values 10%, 15%, 20%, 25%, 30%. Compute hedge ratio and standard deviation per each vol scenario. Explain your findings.

What o submit:

* A word document, report all results, simple equations and derivation in task 1 and 2. Explanation to your findings in task 3.
* BlackScholes.py and Utilities.py with your codes, thus I can run main function by calling the 2 files that you submitted and reproduce results in your report.
* Save report and 2 py files in a zip file, name it as Assignment1.1\_GroupX.zip, X is your team group number.