Interim Project Report

Karim Layoun

GTID: 903210227, ISyE 6767, MSQCF, Gatech

0.1. What is Delta Hedging

Dynamic Delta Hedging is an option trading strategy aiming to mitigate risk associated with the price movements of the underlying assets. The ultimate goal of the strategy is to achieve a delta neutral position, shorting a certain amount n of calls and buying $\delta \times n$ of the underlying asset. In this project, we implement a Dynamic Delta Hedging Strategy in C++.

The δ of an option is calculated as per the Black-Scholes model, where V is the value of the option (a call option), and S is the value of the underlying:

$$\delta = \frac{\partial V}{\partial S}$$

And so, the resulting value of the position/portfolio $\Pi = \delta \times S - V$

0.2. Part I

In part I we test the delta hedging implementation using the Black-Scholes model, and expect great results as both the call price and delta are calculated from the same model and parameters.

0.3. Part II

In part II we back test the delta hedging implementation on real historical Google asset prices, and expect lesser results.

1. Structure of Implementation

1.1. Project File Description

data

A folder including all required data: interest.csv, op_GOOG.csv, sec_GOOG.csv.

out

A folder including all code outputs, including graphs and the final delta hedging backtest given the unittest input as described in the Project Prompt.

main.cpp

Main source code file, processing data, prompting the user for input, and generating the required outputs as graphs (png) and csv files.

theOptionClass.h

Header file defining all option parameters and functions, i.e. S, T, K, r, σ, pricing and implied volatility functions, etc.

theOptionClass.cpp

Source code file implementing all functions defined in the associated header file.

Normal.h

Header file defining an implementation of an approximation of the Normal CDF.

Normal.cpp

Source code file implementing all functions defined in the associated header file

matplotlibcpp.h

Header file defining all functions necessary to the implementation of matplotlib in C++.

Test.h

Header file defining unit test cases for implied volatility and delta calculations.

Test.cpp

Source code file implementing unit test cases in the associated header file

theProject.out

The project executable file.

1.2. Command to Run Project

Compiling the project requires the installation of Boost, Quantlib, Python. To run the project, I used the following command, on a M1 mac:

```
g++ -o theProject.out main.cpp theOptionClass.cpp Normal.cpp Test.cpp
-I/opt/homebrew/Cellar/boost/1.76.0/include
-I/opt/homebrew/Cellar/quantlib/1.23/include -std=c++11
-L/opt/homebrew/Cellar/quantlib/1.23/lib -lQuantLib -lpython2.7
-DWITHOUT_NUMPY
```

The command is also available in the README.md file for the user's convenience.

2. Results & Unit Testing

2.1. Stock Prices & Plot

Asset prices are modeled as stochastic processes following a Geometric Brownian Motion, satisfying the below SDE:

$$S_{t+\Delta t} = S_t + \mu S_t \Delta t + \sigma S_t \sqrt{\Delta t} Z_t$$

$$\begin{aligned} \text{where} \quad & Z_t = \text{a Wiener Process} \\ \text{and} \quad & \mu = \text{drift} \\ \text{and} \quad & \sigma = \text{volatility} \end{aligned}$$

Given the parameters in the project prompt $S_0 = 100, T = 0.4, \mu = 0.05, \sigma = 0.24$, we obtain the below paths modeling the price of an asset

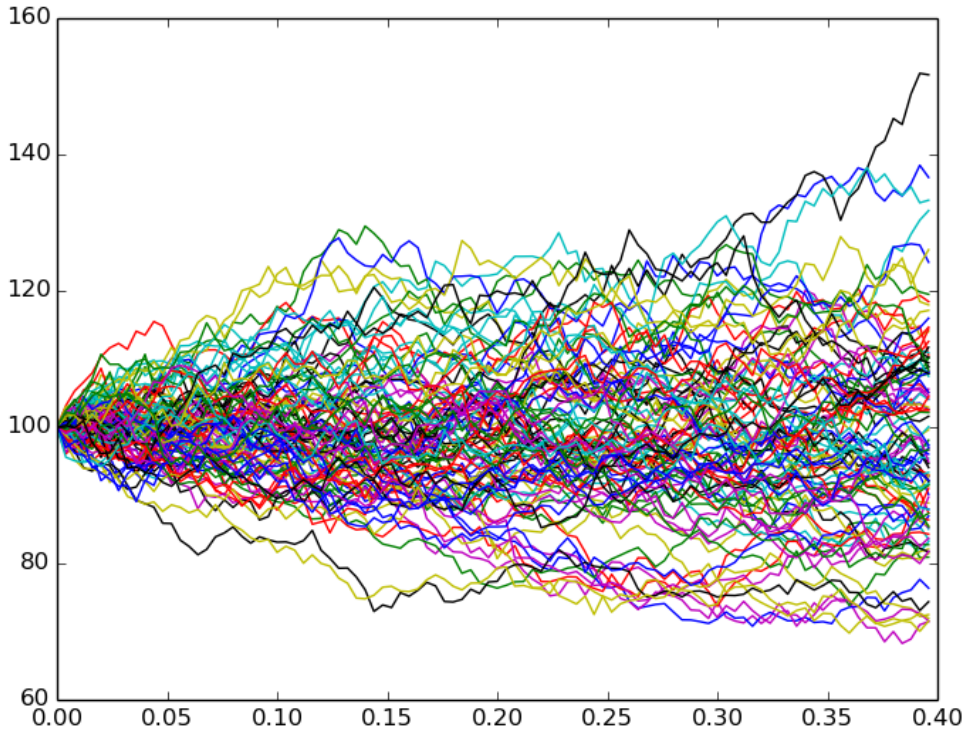

Fig. 1. Stock Price Series

2.2. European Call Prices & Plot

When we solve the Black-Scholes equation for a European Call: $\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial^2 S} + rS\frac{\partial V}{\partial S} - rV = 0$, we obtain the following solution

$$C = N(d_1)S_t - N(d_2)Ke^{-rt}$$

$$\Delta = N(d_1)$$

$$\text{where} \quad d_1 = \frac{ln\frac{S_0}{K} + (r + \frac{\sigma^2}{2})t}{\sigma\sqrt{t}}$$

$$\text{and} \quad d_2 = d_1 = \sigma t$$

For each asset price path, we calculate the associated Call prices, and obtain the below paths.

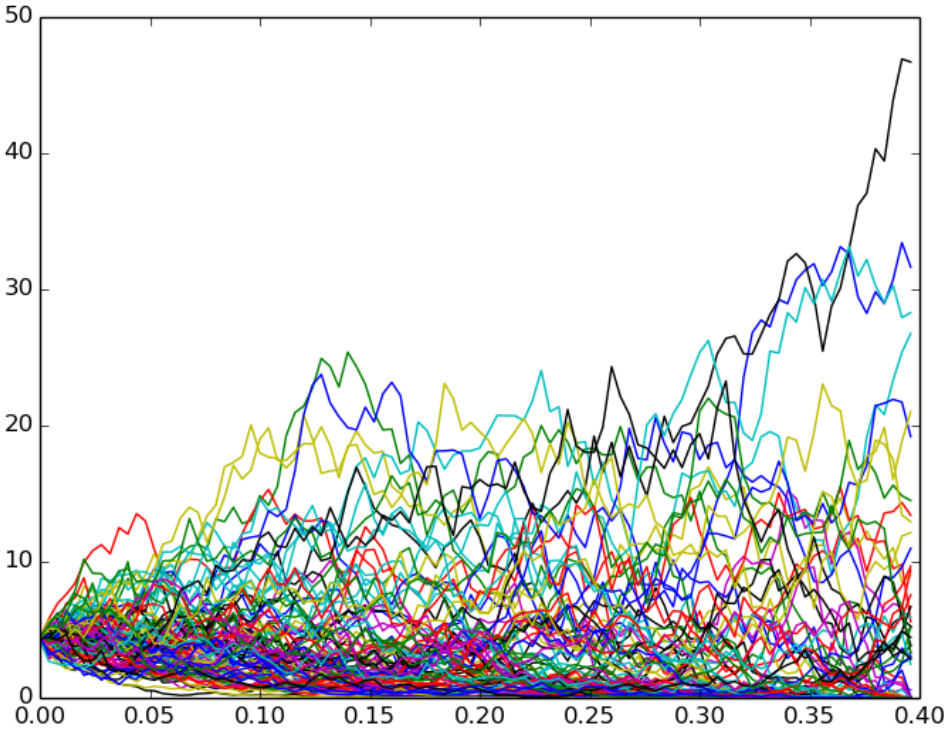

Fig. 2. Call Price Series

2.3. Hedging Error & Distribution

For each date, we calculate the hedging error, as per the equations below.

$$
\begin{aligned}
HE_i &= \delta_{i-1} S_i + B_{i-1} e^{r_{i-1}\Delta t} - V_i \\
B_i &= \delta_{i-1} S_i + B_{i-1} e^{r_{i-1}\delta t} - \delta_i S_i \quad \forall i \geq 1 \\
B_0 &= V_0 - \delta_0 S_0 \\
\delta &= N(d_1)
\end{aligned}
$$

We obtain the following distribution of hedging errors. They seem to follow a normal distribution, validating the assumptions of the Black-Scholes model. Indeed, the asset and call prices are calculated using the same model, which justifies our results.

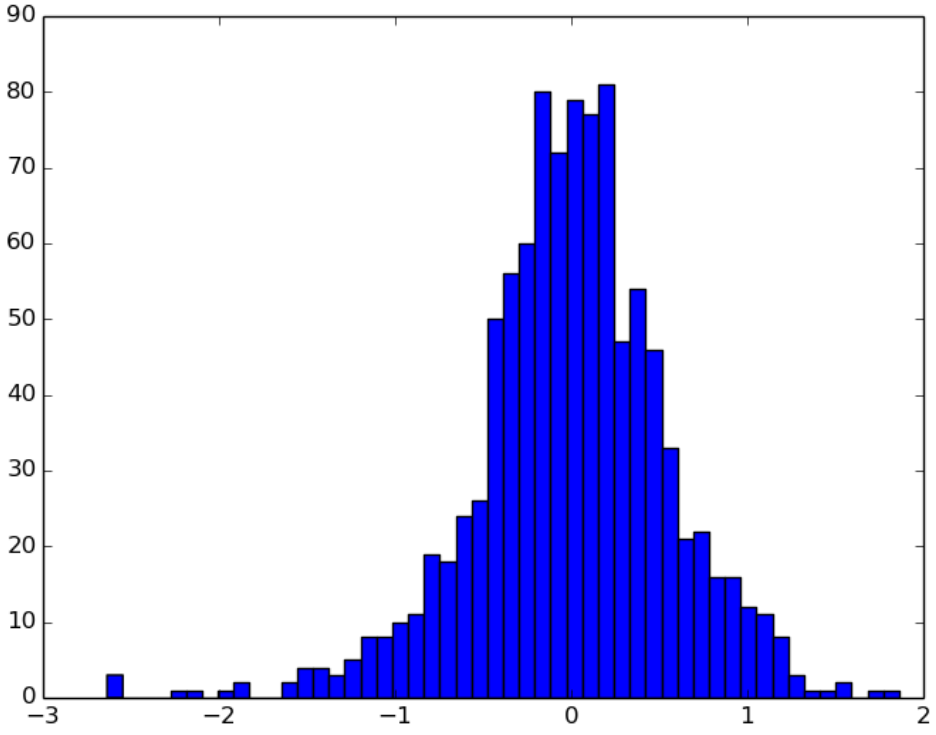

Fig. 3. Cumulative Hedging Distribution

2.4. Delta Hedging Results

In part II, we import Google stock data and interest data, given the inputs of the user prompted to select an option start date, option end date, option maturity date, and option strike price. The graph and table below, found in the output folder, are the result of the inputs stated at the top of Figure 5.

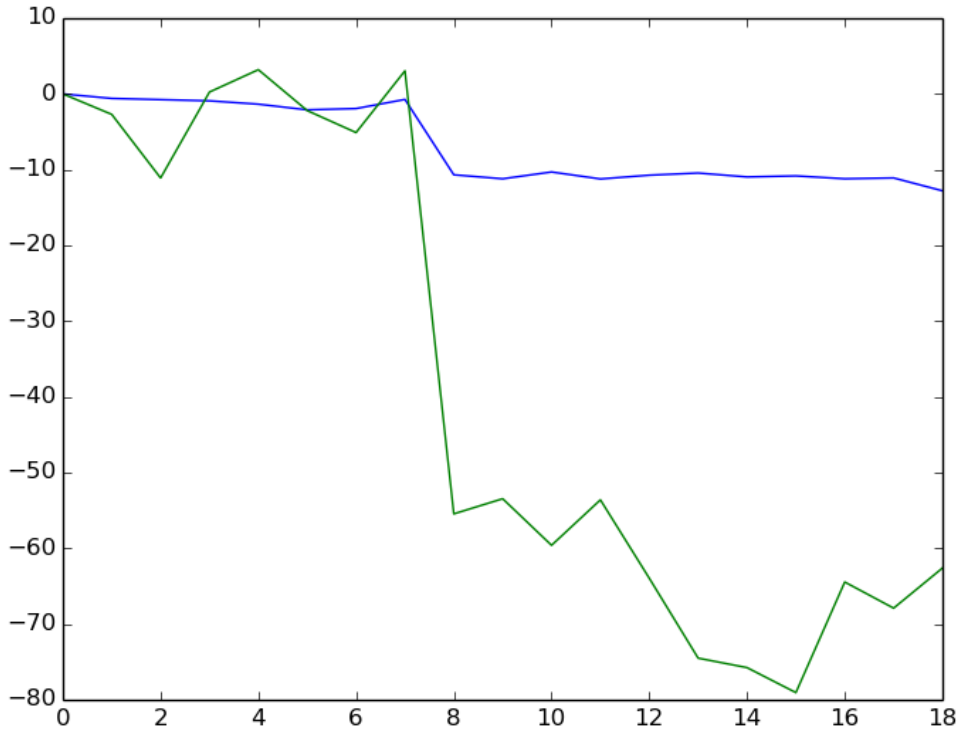

Fig. 4. PNL Hedge(blue) v.s. PNL no Hedge(green)

Figures 4 and 5 clearly indicate that hedging is a more profitable solution than simply selling the call without hedging. Furthermore, a non-hedged portfolio is exposed to infinite downside, as opposed to a hedged portfolio.

results

t0: 2011-07-05							
tN: 2011-07-29							
T: 2011-09-17							
K: 500							
date	S	V	implied volatility	delta	hedging error	PNL	PNL with Hedge
2011-07-05	532.44	44.2	0.259686	0.722654	0	0	0
2011-07-06	535.36	46.9	0.269337	0.733318	-0.592419	-2.7	-0.592419
2011-07-07	546.6	55.3	0.269861	0.787576	-0.160118	-11.1	-0.752537
2011-07-08	531.99	43.95	0.268563	0.719396	-0.159465	0.25	-0.912002
2011-07-11	527.28	41	0.27595	0.69151	-0.440646	3.2	-1.35265
2011-07-12	534.01	46.4	0.286787	0.722767	-0.748333	-2.2	-2.10098
2011-07-13	538.26	49.3	0.287137	0.745056	0.169321	-5.1	-1.93166
2011-07-14	528.94	41.15	0.272178	0.706903	1.20383	3.05	-0.72783
2011-07-15	597.62	99.65	0.28149	0.940745	-9.95187	-55.45	-10.6797
2011-07-18	594.94	97.65	0.300413	0.926444	-0.524013	-53.45	-11.2037
2011-07-19	602.55	103.8	0.266227	0.96029	0.897471	-59.6	-10.3062
2011-07-20	595.35	97.8	0.299707	0.931925	-0.917363	-53.6	-11.2236
2011-07-21	606.99	108.15	0.275665	0.964258	0.494077	-63.95	-10.7295
2011-07-22	618.23	118.7	0.24842	0.985999	0.284388	-74.5	-10.4451
2011-07-25	618.98	119.95	0.294242	0.971589	-0.51448	-75.75	-10.9596
2011-07-26	622.52	123.25	0.286969	0.978582	0.135517	-79.05	-10.8241
2011-07-27	607.22	108.65	0.304804	0.957691	-0.376449	-64.45	-11.2006
2011-07-28	610.94	112.1	0.302651	0.964978	0.108578	-67.9	-11.092
2011-07-29	603.69	106.8	0.370045	0.924709	-1.70016	-62.6	-12.7921

Fig. 5. Results

2.5. Unit Test

Figure 6 is a screen shot of a unittest, with status updates indicating the proper execution of theProject.out, and tests for the implemented delta and implied volatility functions.

```
ISyE6767 - Interim Project - Karim Layoun

Part I
price series generated
price series plot exported to out/thePricePaths
call price and call delta series generated
call price series plot exported to out/theCallPaths
B series generated
cumulative hedging error series generated
cumulative hedging error distribution plot exported to out/CumHedgingDistribution
exporting the price series, the call price series, the call delta series, the B series, and the cumulative hedging error series to the folder out
file export successful
End of Part I

Part II
relevant interest data imported
relevant GOOG closing price data imported
relevant GOOG option data imported
delta hedging backtest complete
results exported to out/results.csv
End of Part II

find below the unit test cases for delta and implied volatility calculations
C(T =0.400000, K =105.000000, S =100.000000, r =0.025000, vol =0.240000) has a delta of 0.428711
C(T =0.400000, K =105.000000, S =100.000000, r =0.025000, vol =0.240000) with a BLS price of 4.39167 has an implied volatility of 0.240000

End of Interim Project
```

Fig. 6. UnitTest

2.6. Conclusion

We have not fully solved the practical problem of hedging call/put options. Delta hedging does not result in a complete risk-free portfolio; delta-gamma hedging comes closer to that objective. However, the Black-Scholes model indicates that perfect hedging involves continuous hedging, hence continuous rebalancing of the portfolio. Consequently, perfectly hedging call/put options is practically impossible, especially given that we have completely ignored transaction fees.