Delta Hedging Project

The objective of this project was to build a dynamic delta hedger when given stock data, interest rate data, and option bid-ask spreads. The delta hedging techniques involves buying or selling a specific number of stocks to offset the risk in an investor's options positions or vice versa. The delta of an option was calculated based on the Black-Scholes-Merton equations and assumptions. If an investor was exposed to a certain amount of delta from the option, an equivalent amount (in delta terms) was bought or sold (depending on the delta sign exposure) in the underlying stock. In our case, it was the GOOG stock. Typically, the investor is exposed to the stock position, and an equivalent delta exposure in the opposite direction is bought/sold in the derivative (the options) in order to offset the risk and be delta neutral.

The project consists of several files: main.cpp, Option.cpp, Option.h, StdNormalCDF.cpp, StdNormalCDF.h, unit_test.cpp, unit_test.h, DeltaHedger.cpp, DeltaHedger.h, DeltaHedgerWithInputData.cpp, DeltaHedgerWithInputData.h.

The main.cpp file is the main code that gets compiled (along with all the other *.cpp files). It calls the DeltaHedger class as well as the DeltaHedgerWithInputData class. Once those two files are run and the outputs are saved to csv files, the code will ask the user if they wish to run unit tests. If the user inputs yes, then the code will run unit tests.

The Option class defines all the attributes and functions of an option. When the constructor is called after passing the strike, spot, option price, risk-free-rate, type of option, start date, and end date, the Option class will generate an instance of it with the parameters defined. Further, the Option class has a function called BSM_Pricer which uses the BSM to price option. Note, that the pricer requires the volatility as an input.

The StdNormalCDF class is a helper class to define the standard normal CDF and be able to generate standard normal random numbers as well as to price options using the BSM formulas. It is also used to calculate delta.

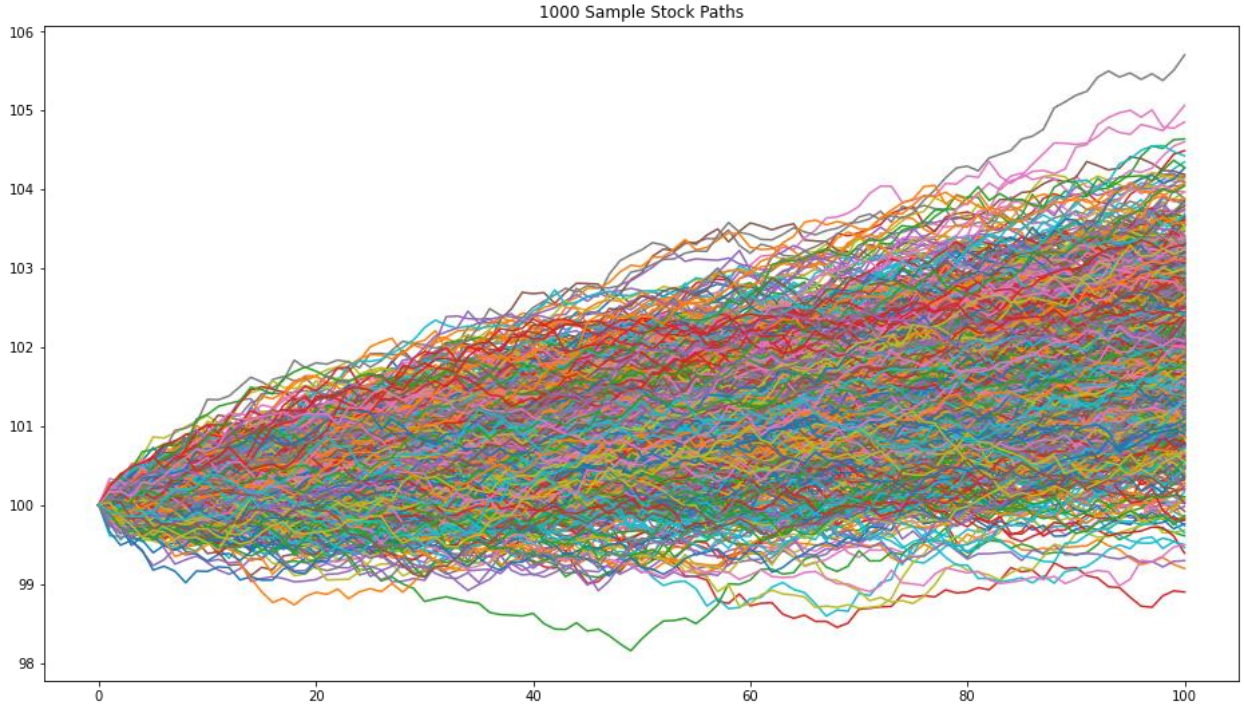
The DeltaHedger class initializes an option with the same parameters as those defined in the project pdf (1.c.). It takes in strike, spot, time to maturity, mean, volatility, risk free rate, number of simulations, and the type of option as an input. It also has a BSM_pricer formula similar to that of the Option class. Further, there are four functions utilized for simulations the outcome series: SimulateStockSeries, SimulateOptionSeries, HegdingError, and SimulateSeries. The SimulateStockSeries function simulates 1000 sample paths that a stock price can take (for 100 steps. It does so by generate standard normal random numbers using the Mersenne Twister engine. This enginer is a random number engine based on the Mersenne Twister algorithm and it produces high quality unsigned integer numbers on a desired interval. Once the numbers are generated, it is passed to the formula used in the project pdf (1.a.). The SimulateOptionSeries function calculates the BSM option price given the simulated stock prices and stores the option prices in a vector of type double. The HedgingError function takes in the simulated stock prices and the simulated option prices as parameters. It first calculates the B

value based on the assignment pdf and then calculates the hedging error (bullet point 5). Lastly, the SimulateSeries function takes in the number of simulations as an input. It then calls the other simulation functions and stores each of the outputs into a csv file for plotting and other uses.

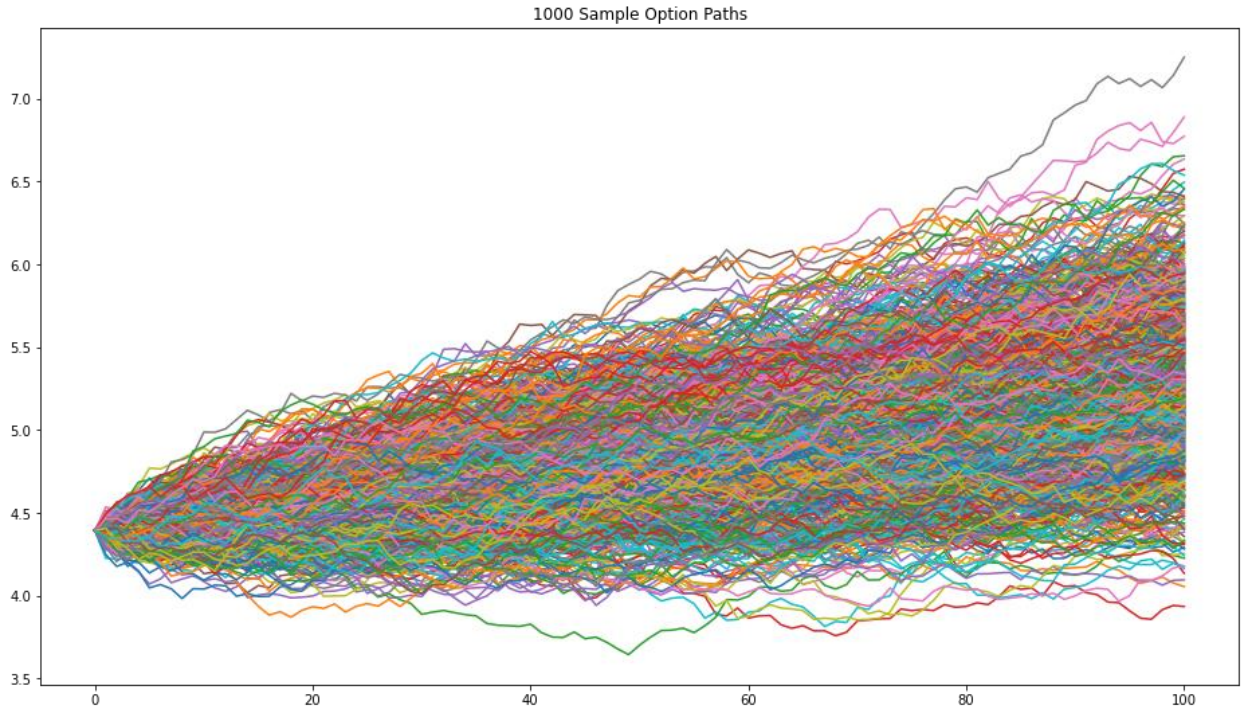
The DeltaHedgerWithInput class takes in the start date, end date and expiration date as well as the strike as input variables. It then calculates the implied volatility, delta, hedging error, profit and loss, and hedged profit and loss. There are also helper functions that are outside of the class but the class calls them for efficient and readable code. Helper functions include reading the interest rate csv file and the GOOG stock price csv file. There is another helper function that matches values between arrays based on given dates. The read_option function in the DeltaHedgerWithInput class combines the csv files and a vector with values being from the Option class. The get_IV function calculates the implied volatility by creating an array of volatility values from 0.01 to 0.99 incremented by 0.001. Then through dynamically updating the minimum, we obtain the necessary implied volatility based on the BSM_pricer function. The get_Delta function calculates delta based on the BSM formulas. This is done for each option in as read from the csv files. In order to get the time to maturity, we used the get_T function that calculates the time to maturity between two string dates. The get_HE function calculates the hedging error as conveyed in the assignment PDF. The results were annualized by dividing by 252 trading days. The get_PNL function calculates the profit or loss of the strategy by calculating the difference between the original option price and the newly hedged option price. Latly, the to_csv function outputs the results to a csv file (output.csv). Note that the *HE* column was ignored since it has the same values as that of the *PNL Hedged* column (this was also answered in piazza).

The unit_test class calculates unit tests on the implied volatility and delta functions.

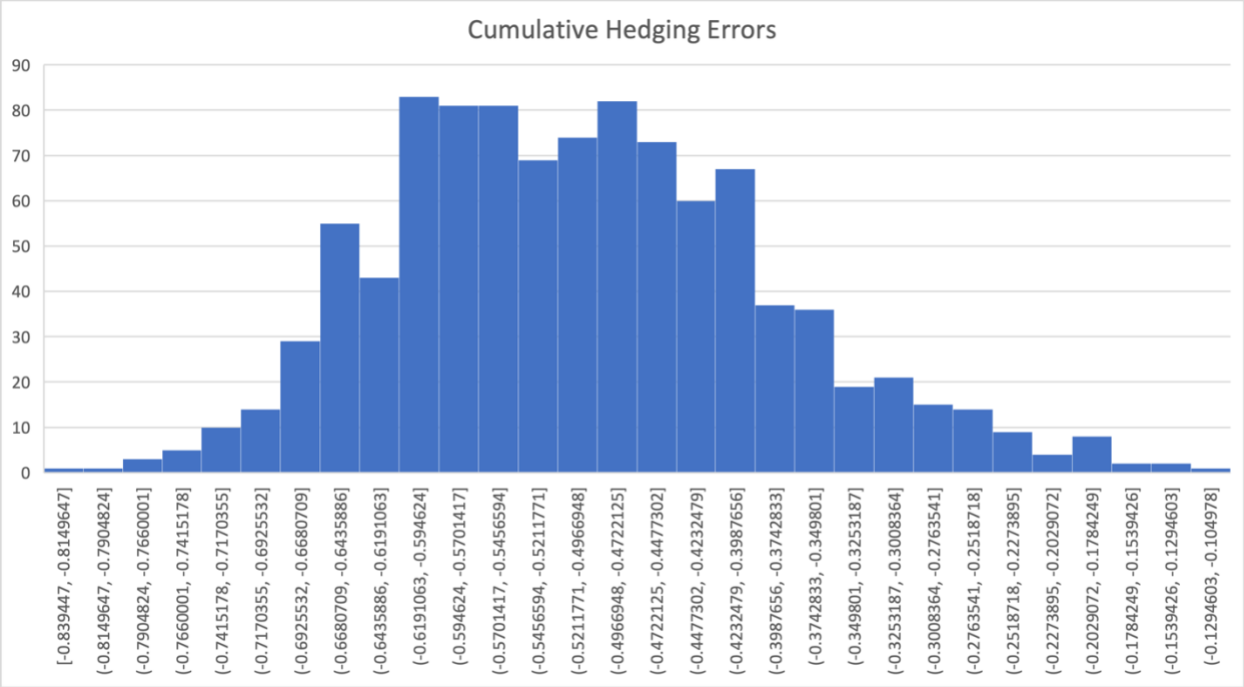
The results of the project were interesting. First, simulating 1000 observations for a stock with given parameters yielded the following graph. Based on the Monte Carlo simulation, on average the stock price increases.



As for the option series simulations, the results show that the average price of an option increases (see figure below).



Lastly, the distribution of the cumulative hedging errors exhibited skewed normal distributive behavior as seen from the figure below.



All in all, as seen from the output.csv file, the hedged PNL had a much lesser impact on the portfolio of the investor from July 5, 2011, to July 29, 2011 on the GOOG stock. The PNL if it were not hedged would have been around 62.6 loss versus the hedged position would have only yielded a 12.12 loss. This is however not taking into account transaction costs, dividends, and such. This strategy also requires constant rebalancing in order to be delta neutral, where in reality, especially at regulated banks, that might not be so possible.

Date	Stock Price	Option Price	Implied Vola	Delta	PNL	PNL Hedged
7/5/11	532.44	44.2	0.207	0.732854	0	0
7/6/11	535.36	46.9	0.214	0.743968	-2.7	-0.562676
7/7/11	546.6	55.3	0.214	0.799882	-11.1	-0.603265
7/8/11	531.99	43.95	0.212	0.729611	0.25	-0.942118
7/11/11	527.28	41	0.221	0.69991	3.2	-1.43091
7/12/11	534.01	46.4	0.229	0.732503	-2.2	-2.12288
7/13/11	538.26	49.3	0.228	0.756111	-5.1	-1.91194
7/14/11	528.94	41.15	0.216	0.716098	3.05	-0.811038
7/15/11	597.62	99.65	0.221	0.951588	-55.45	-10.1314
7/18/11	594.94	97.65	0.24	0.93754	-53.45	-10.6845
7/19/11	602.55	103.8	0.21	0.969662	-59.6	-9.70303
7/20/11	595.35	97.8	0.237	0.943318	-53.6	-10.6883
7/21/11	606.99	108.15	0.215	0.973562	-63.95	-10.0618
7/22/11	618.23	118.7	0.187	0.992577	-74.5	-9.67291
7/25/11	618.98	119.95	0.232	0.97959	-75.75	-10.1825
7/26/11	622.52	123.25	0.224	0.985707	-79.05	-10.0189
7/27/11	607.22	108.65	0.24	0.967156	-64.45	-10.5044
7/28/11	610.94	112.1	0.236	0.974086	-67.9	-10.3606
7/29/11	603.69	106.8	0.291	0.935963	-62.6	-12.1267

Appendix

Here is a screenshot of the code running successfully!