ISYE 6767: Sys-Computation Finance

Mid-Term Project Report

On

Implementing a Dynamic Delta Hedging Strategy

By- Suvrasoumya Mohanty (GT id -903582405)

Under Guidance of Prof. Shije Deng and Moyi Guo

Introduction

Dynamic delta hedging strategy is when a counterparty sells a financial instrument and replicates the payoff of the instrument using the independent payoff of different instruments. The counterparty can then earn a profit on commissions in an illiquid market. This strategy can also be used to identify instrument mispricing and one can net the arbitrage on the position.

This project on dynamic delta hedging is implemented with C++ and results are extracted out in CSV files. These files are then visualized and interpreted using a Python Jupyter notebook.

Code Explanation

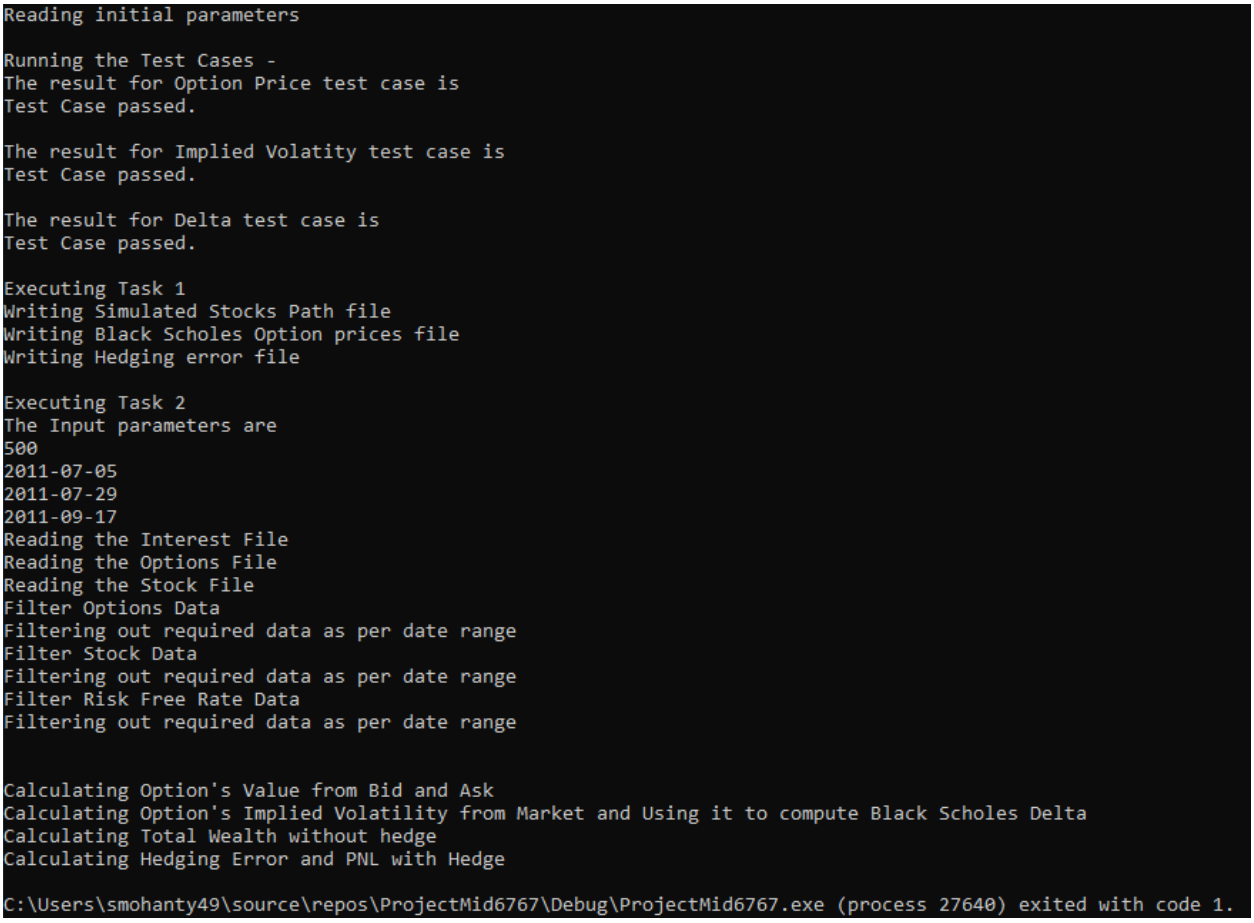
Here I provide a brief explanation of the structure of the project framework. The main() function executes both tasks to satisfy the project requirements.

Task 1 uses the `vector<double> normal_dist(int n, int p)` function to generate the sample paths and then `Option class` to create option objects and then price these options and output their respective Black Scholes delta's. This is all used as an input for the `vector <vector<double>> HedgingError` function which takes multiple paths of stocks and options prices and returns their respective hedging errors. This result is then output to csv files and then a Jupyter notebook (attached in the submission) is used to create the visualizations (attached below). This concludes the flow for Task 1

Task 2 is run by function which requires the dates (start date, end date, expiry date) and strikes to be considered as the f parameters. This then reads the input data files provided using `read_csv(path, mask)` function. The data of interest is then filtered out by `filter_data(stock_file, t0, tN, T, "US", 0)` as per input specifications. As a result, we handle less data and work more efficiently. The Implied Volatility and Delta for the option on the dates of interest is computed by `getIV(S[i], V[i], riskfree[i], strike_GOOG, ttm / 252)` which in turn uses option class for Black Scholes pricing function and cumulative normal distribution. Once we have delta and options price, we move to delta hedging strategy which is executed by `vector <vector<double>> HedgingError` (from task 1). Finally, respective PNL's are calculated, and the results are compiled in a csv file by `create_result_file(result_column, option_filtered, S, V, IV, del_Goog, hedge_err, PNL, PNL_hedge) & csv_out1("result.csv", final_result)` functions.

Refer below to Implementation Section for various functions and class definitions and their workings.

Code Run console screenshot



Implementation

The code is implemented in a modular programming and object-oriented programming fashion to reduce dependencies between independent tasks. The various modules of the project are detailed below.

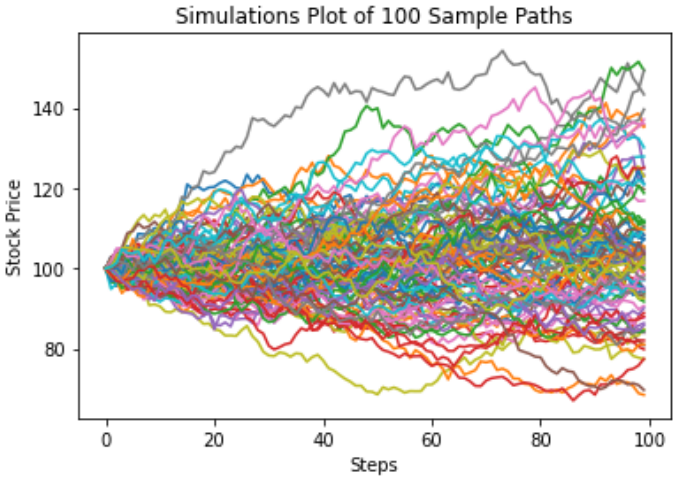
Name	Type	Definition
`Option`	Class	Defines an Option object (which stores various Option characteristics) and contains the Black Scholes pricing function and 'delta' value.
`double Option::BlackScholes()`	Function	Used to compute BlackScholes price of an option given the various option parameters by creating an object of Option Class. It is defined in the Option class is a functionality.
`double N(double x)`	Internal Function	This is another function inside Option class for the purpose of giving an approximation of cumulative normal distribution.

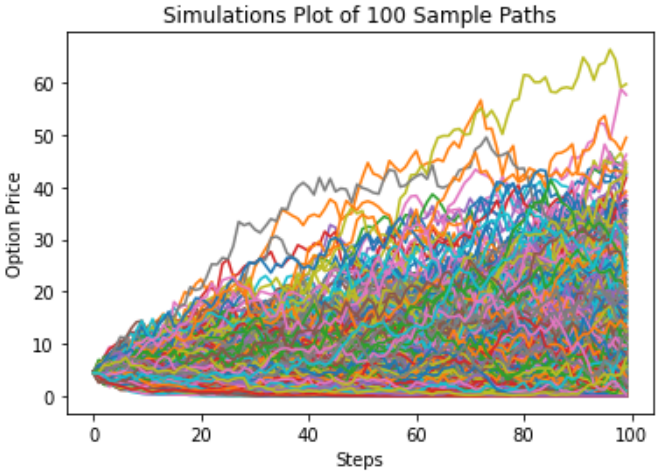
`vector<double> normal_dist(int n, int p)`	Function	This is used to generate various paths that are taken by the stock price over the course of its steps. This randomly generates 'n' sample from normal distribution given the seed 'p'.
`vector <vector<double>> HedgingError(vector<vector<double>> S_t, vector <vector<double>> C_t, vector <vector<double>> delta, double delt, vector<double> r, double T)`	Function	This is the function that computes the hedging error for a range of stock paths, option values and risk-free rates. This follows that we can hedge a call option by buying 'delta' shares and loan the money required after using the proceeds from selling the call option.
`vector<double> getIV(double S, double V, double riskfree, double strike_GOOG, double ttm)`	Function	This function returns the Implied Volatility of an option given its Market Price, risk-free rates, underlying price, maturity and strike. It uses a binary search method and has an error threshold of $1*10^{-6}$ which is flexible.
`vector<vector<string>> filter_data(vector<vector<string>> op, string t0, string tN, string T, string convention, int col2, bool option =false, int col1 = 0)`	Function	This is the function that filters out only the required data to be analyzed after user-defined input ranges. This is critical since the options data is very high-dimensional and requires significant compute power to go through it entirely.
`void print(vector <vector <string>>file)`	Function	This is used to print out the data files that have been read or created throughout the process for better visibility
`void csv_out(string name, vector <vector <double>> data)`	Function	Creates csv output file for `vector <vector <double>> dataset` and saves it with filename passed to the function. Used to create stocks path, options and hedging error file.
`void csv_out1(string name, vector <vector <string>> data)`	Function	Creates csv output file for `vector <vector <string>> dataset` and saves it with filename passed to the function. Used to create the final 'results' file
`void run_test_cases()`	Function	This runs the test cases that are defined in the 'unit_test' class
`unit_test`	Class	Class consisting of 3-unit test case for the code to check the Black Scholes Price, Implied Volatility and Delta respectively.
`vector <string> get_input()`	Function	This function is used to get input from the user
`void Task2(double strike_GOOG, string t0, string tN, string T)`	Function	Executes the entire task2 given the user defined inputs 'strike price', 'start date', 'end date' and 'expiry'.

`int main()`	Main Function	It is where both tasks of this project get implemented. Various function calls happen.

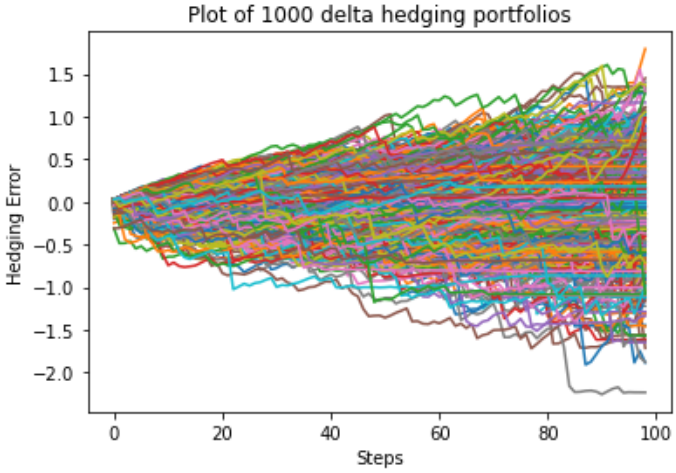
Results and Analysis

The visualization of 100 sample paths informs us that the simulated paths are truly random and are not biased. The corresponding option price plot tells a similar story except that it bottoms out at 0, since option price (naïve case) cannot be negative. However, the paths tell us that the simulated normal distribution are working fine as per the assumption of our model and can give a fair estimate of future option price.

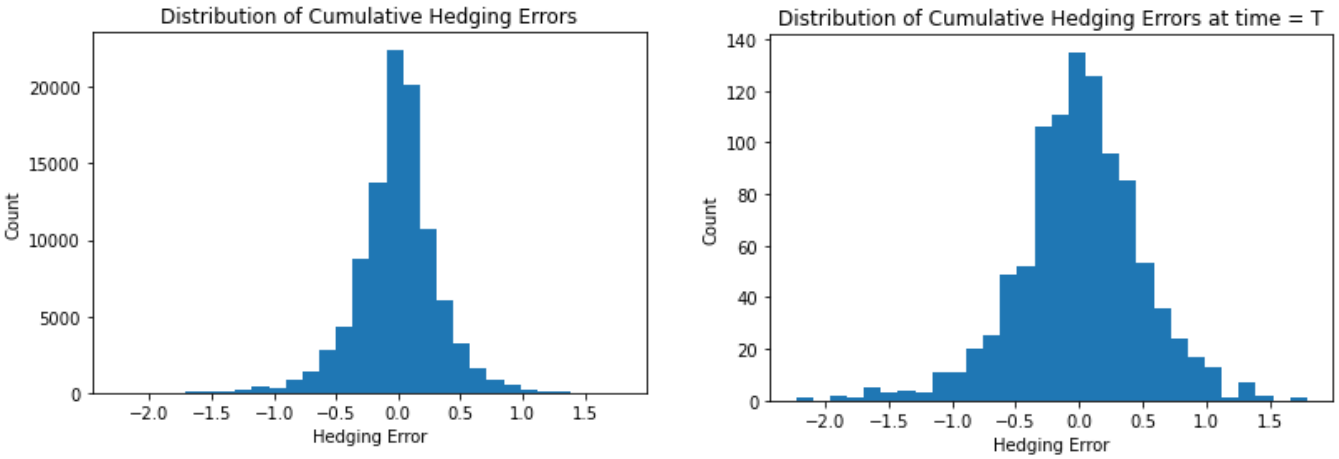




Below are the plots of hedging error paths and their distribution – throughout the simulated period and at maturity. From the first plot, we can see that in most cases there is minimal error, and it is range bound with absolute value of 2. Most of the lines are flat, implying that there is no/little additional cost to hedging the sold call option.



The above becomes even more apparent histogram (or distribution plot). In the total time-period case, it is heavily centered around 0 with minimal distribution elsewhere. At maturity, the distribution is slightly more spread out (expected) as it is cumulative distribution. The error that we obtain is due to theta or time interval of discrete hedging. This drives home the point that despite being dynamic, it is still far from the perfect hedge. However, despite these weaknesses, the cumulative loss (cost) is minimal which can be recovered in the commissions. The errors are particularly small because the step size considered for the process is very small (0.4/100 = 0.004).



The final screenshot displays the results file created at the end of Task 2. This is a particularly insightful summary of the process completed in Task 2. The sample start and end dates are taken as 7/5/2011 and 7/29/2011 respectively. The stock price of GOOG moves from 532.44 to 603.69 during this period. This implies that we are at a significant disadvantage as we have sold a call option. Naturally, the price moves against us as the stock is rising. Next, we have the column for market observed call option price with strike 500 and expiry at 2011-09-17. As the stock price rises, the implied volatility calculated from Black Scholes rises as well. The delta of the call follows this trend as the option becomes more ITM. It approaches 1 from the start of around 0.723. The total wealth without putting any is simply PNL (without hedge) + Proceeds from Selling the call option (at T0).

The final couple columns are PNL (without hedge) and PNL (with hedge). These two columns summarize the entire argument of dynamic delta hedging. PNL (without hedge) ranges from 3.2 to -79.05 (worst day) whereas PNL (with hedge) ranges from ~ -12.8 (final day) to 0 (start day). It is observed that hedging protects us from significant downsides that can result from asset prices moving against us. The negative PNL obtained with the hedge is just the cost associated with hedging. The cost is impacted by how frequently we are hedging out position (here 1 day), the smaller the time interval, the lesser the negative PNL. However, it is important to note that these inferences are based on the simplifying assumption of zero transaction costs. In the real world, where these frictions exist the actual cost might be much higher with very small-time intervals and the process must be optimized to minimize the cost.

date	Stock Price	V	IV	Delta	Hedging Error	Total Wealth (without Hedge)	PNL (without hedge)	PNL (with hedge)
7/5/2011	532.44	44.2	0.21921	0.723497	0	0	0	0
7/6/2011	535.36	46.9	0.226738	0.734158	-0.589959	-2.7	-2.7	-0.58996
7/7/2011	546.6	55.3	0.226381	0.788611	-0.150673	-11.1	-11.1	-0.74063
7/8/2011	531.99	43.95	0.224866	0.720139	-0.764558	0.25	0.25	-0.91523
7/11/2011	527.28	41	0.233814	0.69212	-0.594823	3.2	3.2	-1.35938
7/12/2011	534.01	46.4	0.242239	0.723448	-1.508784	-2.2	-2.2	-2.10361
7/13/2011	538.26	49.3	0.241825	0.745705	-0.422612	-5.1	-5.1	-1.9314
7/14/2011	528.94	41.15	0.228559	0.707495	-0.311005	3.05	3.05	-0.73362
7/15/2011	597.62	99.65	0.23455	0.941925	-10.33384	-55.45	-55.45	-10.6448
7/18/2011	594.94	97.65	0.254009	0.927373	-0.838185	-53.45	-53.45	-11.172
7/19/2011	602.55	103.8	0.22318	0.961745	-9.4293	-59.6	-59.6	-10.2675
7/20/2011	595.35	97.8	0.251308	0.93317	-1.766025	-53.6	-53.6	-11.1953
7/21/2011	606.99	108.15	0.228898	0.965988	-8.920741	-63.95	-63.95	-10.6868
7/22/2011	618.23	118.7	0.201264	0.988396	-1.462201	-74.5	-74.5	-10.3829
7/25/2011	618.98	119.95	0.246907	0.973106	-9.433434	-75.75	-75.75	-10.8956
7/26/2011	622.52	123.25	0.238687	0.980347	-1.321321	-79.05	-79.05	-10.7548
7/27/2011	607.22	108.65	0.254697	0.959141	-9.836891	-64.45	-64.45	-11.1582
7/28/2011	610.94	112.1	0.251412	0.966529	-1.207357	-67.9	-67.9	-11.0442
7/29/2011	603.69	106.8	0.308185	0.925674	-11.548304	-62.6	-62.6	-12.7557

Conclusions

The Hedging errors are minimal and lie between -2.5 to +2 for task 1 for the extreme case. However, that was a theoretic implementation with stricter assumption and conditions. When we move this model to the real-world case, we get more reasonable and practical results. The hedging error for the sample period is -12, which is to be expected. However, this is in line with prior objective that is to hedge the downside risks associated with portfolio. This is clear from the difference between hedged and unhedged portfolio difference. Dynamic delta hedging can work to replicate cash flows of a financial instrument and has multiple applications in the financial industry.