Midterm Project

Dynamic Delta Hedging Report and Discussions

ISYE/MATH 6767- Fall 2017

-Yashovardhan Jallan

1. Problem addressed by the project and the model(s) used in the project

This project is aimed to implement Dynamic Delta Hedging strategy while on selling a call option. Delta hedging is an options strategy that aims to reduce, or hedge, the risk associated with price movements in the underlying asset, by offsetting long and short positions. For example, a short call position may be delta hedged by longing the underlying stock. This strategy is based on the change in premium, or price of option, caused by a change in the price of the underlying security.

In our given case, in which we are shorting a call option, we calculate Delta for each day and take the appropriate long position in stock. Delta, by definition, is the rate of change of the option premium by rate of change of stock.

Delta is calculated as CDF (d_1) where d_1 is the Black Scholes parameter given by,

$$d_1 = \frac{\ln(S_0/K) + (r - q + \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}$$

where the symbols represent their usual meanings.

2. The structure of the model implementation: functionality of each code block or each code piece.

Files Submitted and their brief description:

1) main.cpp (Main File)
2) all_funcs.h (Header file for functions defined)
3) all_funcs.cpp (Implementation file for functions defined)
4) StdNormalCDF.h (Header file for StdNormalCDF class)
5) StdNormalCDF.cpp (Implementation file for StdNormalCDF class)
6) options.h (Header file for my Black Scholes option price calculator class)
7) options.cpp (Implementation file for my Black Scholes option price calculator class)

Key discussion points from the code submitted:

- DrawNormal() function in the file ‘all_funcs.cpp’

  I have used the Box-Muller method to generate the standard normal random variable. Professor had discussed the implementation of this method and I have used it to simulate the stock prices for the first part of the project.

- interval_bisection() function in the file ‘all_funcs.cpp’

  To calculate implied volatility, I have used the bisection method. It is a simple method which requires a high value and a low value and if set up properly, the numerical approximation converges. I searched on the internet for this method and used this link as inspiration: https://www.quantstart.com/articles/Implied-Volatility-in-C-using-Template-Functions-and-Interval-Bisection

- ‘main.cpp’ file discussions

  For the first part, it is pretty straight forward. I have declared several variables of the type *double* to store the given values of stock price, time to expiration, volatility, mew, number of intervals etc. Then I have created several vectors of the type *double* to store the values obtained for each iteration. This exercise is not necessary as we can directly print the values at each step to the csv file without storing, but I have nonetheless kept it in the vectors. If speed is an issue in future, I can change this part.

  For the second part, first I have taken user input to get the start date (t0), the date until which hedging is done (tn) and the time of expiration of the option (texp). The strike price of the option is also taken as user input.

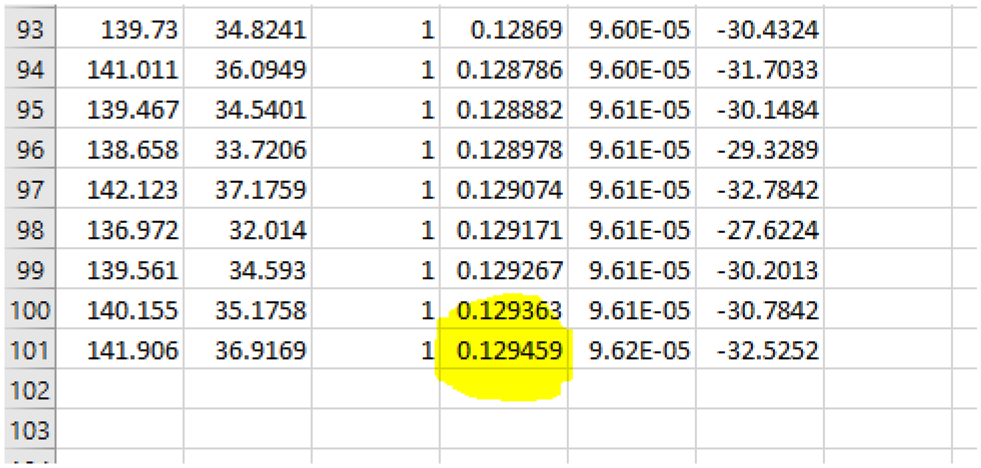
  Next, I am calculating the number of business days between t0 and texp, using QuantLib library. We need this because we want to know the time of expiration of the option correctly.

  After this step, I have read the three given csv files in my code and stored them in vectors for convenience. The first file ‘interest.csv’ provides the risk-free interest rates for the given dates. The second file ‘sec_GOOG.csv’ is the Google adjusted close price dataset. And the third file ‘op_GOOG.csv’ is the option price for google call and put options for various strike prices.

  Once, I have finished reading the files, the rest of the code is similar to part I except there is one key difference. In part I, the volatility was provided to us. In part II, it is not given. We are numerically approximating the implied volatility using the bisection method mentioned above.

3. Outcome of the implementation and discussion on whether the outcome solves the problem.

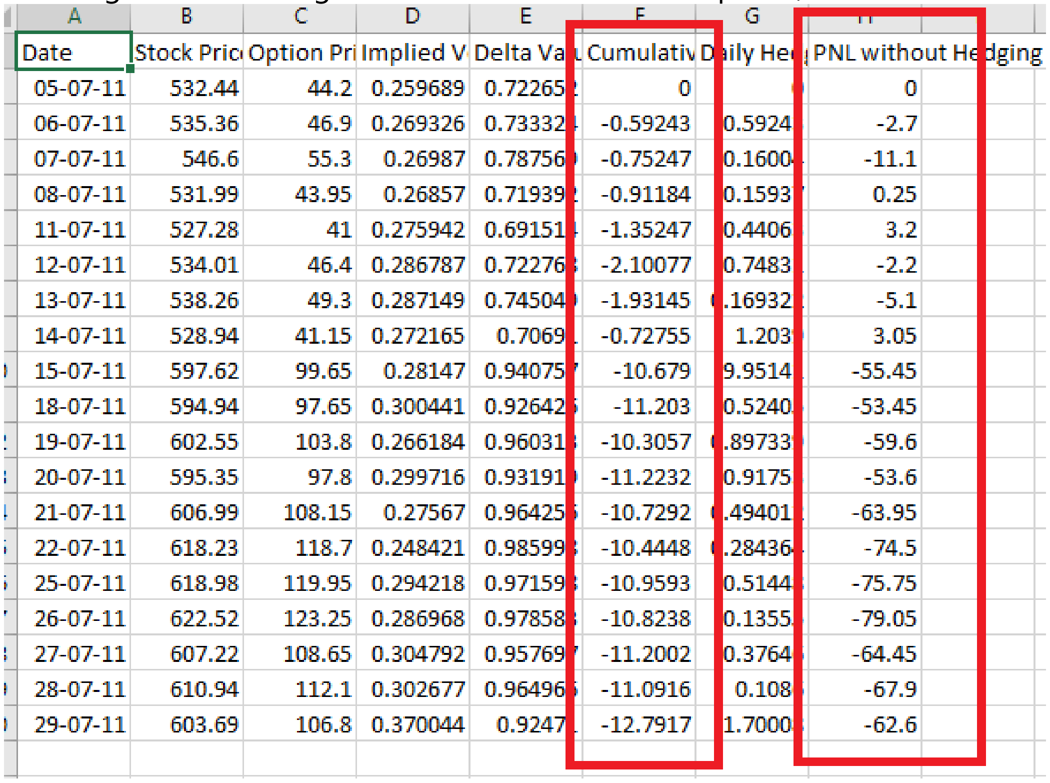
Looking at the output generated for the first part of the project in the given figure,



93	139.73	34.8241	1	0.12869	9.60E-05	-30.4324
94	141.011	36.0949	1	0.128786	9.60E-05	-31.7033
95	139.467	34.5401	1	0.128882	9.61E-05	-30.1484
96	138.658	33.7206	1	0.128978	9.61E-05	-29.3289
97	142.123	37.1759	1	0.129074	9.61E-05	-32.7842
98	136.972	32.014	1	0.129171	9.61E-05	-27.6224
99	139.561	34.593	1	0.129267	9.61E-05	-30.2013
100	140.155	35.1758	1	0.129363	9.61E-05	-30.7842
101	141.906	36.9169	1	0.129459	9.62E-05	-32.5252
102						
103						

We can see that the cumulative hedging error after 100 days is 0.129459. Since we are hedging based on Delta obtained by our own Black Scholes calculator and we are also obtaining the option price using the same parameters, it is expected to be close to zero. Which, in this case, is being satisfied as the error of 0.129459 is small compared to the price of the call option which is close to $37 on the last day.

Now looking at the results generated for the test case in part II,



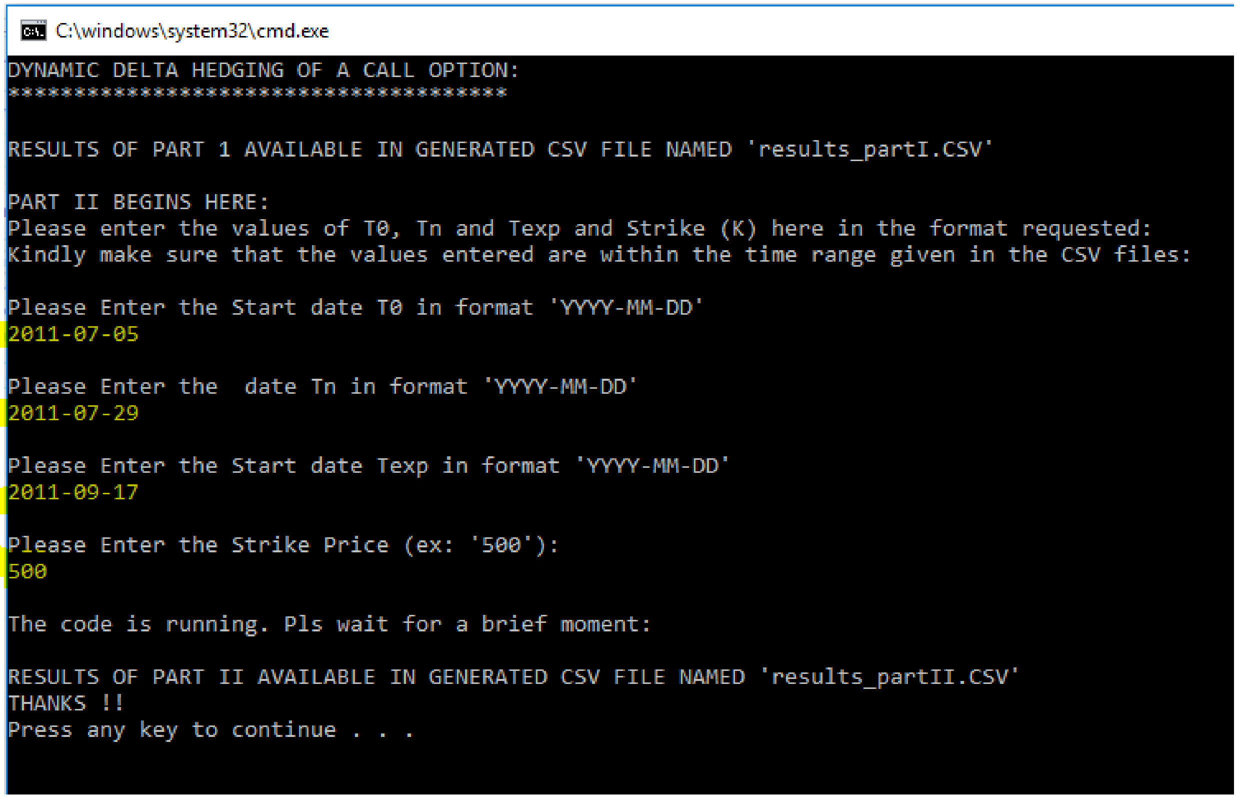
A	B	C	D	E	F	G	H
Date	Stock Pric	Option Pri	Implied V	Delta Va	Cumulativ	Daily He	PNL without Hedging
05-07-11	532.44	44.2	0.259689	0.72265	0		0
06-07-11	535.36	46.9	0.269326	0.73332	-0.59243	0.5924	-2.7
07-07-11	546.6	55.3	0.26987	0.78756	-0.75247	0.1600	-11.1
08-07-11	531.99	43.95	0.26857	0.71939	-0.91184	0.1593	0.25
11-07-11	527.28	41	0.275942	0.69151	-1.35247	0.4406	3.2
12-07-11	534.01	46.4	0.286787	0.72276	-2.10077	0.7483	-2.2
13-07-11	538.26	49.3	0.287149	0.74504	-1.93145	.16932	-5.1
14-07-11	528.94	41.15	0.272165	0.7069	-0.72755	1.203	3.05
15-07-11	597.62	99.65	0.28147	0.94075	-10.679	9.9514	-55.45
18-07-11	594.94	97.65	0.300441	0.92642	-11.203	0.5240	-53.45
19-07-11	602.55	103.8	0.266184	0.96031	-10.3057	.89733	-59.6
20-07-11	595.35	97.8	0.299716	0.93191	-11.2232	0.9175	-53.6
21-07-11	606.99	108.15	0.27567	0.96425	-10.7292	.49401	-63.95
22-07-11	618.23	118.7	0.248421	0.98599	-10.4448	.28436	-74.5
25-07-11	618.98	119.95	0.294218	0.97159	-10.9593	0.5144	-75.75
26-07-11	622.52	123.25	0.286968	0.97858	-10.8238	0.1355	-79.05
27-07-11	607.22	108.65	0.304792	0.95769	-11.2002	0.3764	-64.45
28-07-11	610.94	112.1	0.302677	0.96496	-11.0916	0.108	-67.9
29-07-11	603.69	106.8	0.370044	0.9247	-12.7917	1.7000	-62.6

We see that the cumulative hedging error at the end of the time period is -$12.791. Although this is not exactly a small and insignificant number, the loss is still much lower than the loss without hedging (-$62.6). We can conclude that in this case, i.e. modeling delta hedging strategy for a Google call option by longing stocks in the given period, dynamic delta hedging strategy is definitely useful but it does expose us to some small losses.

4. Unit-Test cases for the code

To run the code, I have provided a unit test case here. Before running the code, make sure that the Boost and QuantLib libraries are installed and linked to the compiler.

I have run the code for the sample input given to us to generate the output files.



```
C:\windows\system32\cmd.exe
DYNAMIC DELTA HEDGING OF A CALL OPTION:
***************************************

RESULTS OF PART 1 AVAILABLE IN GENERATED CSV FILE NAMED 'results_partI.CSV'

PART II BEGINS HERE:
Please enter the values of T0, Tn and Texp and Strike (K) here in the format requested:
Kindly make sure that the values entered are within the time range given in the CSV files:

Please Enter the Start date T0 in format 'YYYY-MM-DD'
2011-07-05

Please Enter the  date Tn in format 'YYYY-MM-DD'
2011-07-29

Please Enter the Start date Texp in format 'YYYY-MM-DD'
2011-09-17

Please Enter the Strike Price (ex: '500'):
500

The code is running. Pls wait for a brief moment:

RESULTS OF PART II AVAILABLE IN GENERATED CSV FILE NAMED 'results_partII.CSV'
THANKS !!
Press any key to continue . . .
```

The output I received are stored in CSV files ‘results_partI.csv’ and ‘results_partII.csv’. The pics above are taken from the same files.