Mid-Term Project Report: ISYE/MATH 6767 - Implementation of Financial systems using C++

Name: Rachit Maheshwari GT ID: 903291455 Date: Oct 19,2017

Note: I want to exercise my option of one-time extension for the midterm project.

Problem Addressed by the Project and the Models

The Goal of this project is to implement the Delta Hedging Strategy using Black-Scholes-Merton (BSM) Option pricing formula. We do this by first calculating the implied volatility of the option based on real Market European Option price. Using this implied volatility, we can calculate the delta and also the hedging error. We also validate the applicability of BSM option pricing Model on actual prices seen in the market as compared to Stock Prices following a Brownian motion.

Structure of Model Implementation

Class Structure:

The solution implements five classes: PayOff Class, Option Class, BSCalculator Class, StockPriceSimulator Class and StdNormalCDF Class are used in the model.

1. PayOff Class:

This is a generic class which can be used to calculate the payoff for any European option (Call & Put). It also supports calculating the payoff of power option . It has four elements option type, power, Spot Price and Strike Price) which are protected elements and can be retrieved using corresponding get and set methods.

2. StdNormalCDF Class:

This generic calculator class helps in getting the CDF value of any parameter on a standard Normal distribution.

3. BSCalculator Class:

This class implements the BSM model and should be used as a calculator. It takes in Spot price, Strike price, Time to maturity, interest rate, volatility, dividend yield rate and option type as inputs. It can calculate the BSM implied option Price, Option Delta and Option Vega. It calls the StdNormalCDF calculator for computing some of the values.

4. Option Class:

This is a generic European Option Class. It inherits all the functionalities of the PayOff Class. Using the Black Scholes calculator, it has method to find the implied volatility of an option based on the Market Prices. The implied volatility is calculated using bisection method. It also has various calculation methods for calculating option price, option delta and option vega.

Currently these methods calculate the option Greeks using the Black Scholes calculator. In future we can extend and add methods using other pricing methods as well.

5. StockPriceSimulator Class

This Class outputs a simulated Price vector for n periods following Brownian motion based on volatility, Mu and interest rate.

Code Structure:

The main method solves two problems:

Part 1: Stock Simulation following Brownian motion, finding Delta & Cumulative Hedging error.

The code uses the StockPriceSimulator to generate a vector of simulated Stock prices. Using the volatility, interest rate, Strike Price and time to maturity the delta is calculated using the BSCalculator. The setMaturity(updated_maturity) & setSpot(Spot Price) methods help in updating the BSCalculator parameters. Using the updated Delta, the Hedging error can be computed at each period. The output is written into 'hedgingError_simulation_part1.csv' file.

Part2: Dynamic Delta Hedging on observed Option market prices

The QuantLib Date variable is used and the String date stored in files is converted to QuantLib Date object. This is done for ease in handing dates and for calculating the business days between two days for US Calendar. Corresponding helper String to Date and Date to String functions are created.

Reading the files: Interest and Security price data is read into one dimensional maps and the option data is read into multidimensional map (four) for easy and fast data access and retrieval. Corresponding Reading helper fuctions are created.

While taking the user inputs relevant checks are made:

- Correct date format for start, end and expiration date
- Correct Call or Put Option Flag
- Correct format for Strike Price which is a double. While storing Strike price it is converted to integer for matching easily as a key in a hashmap. (multiply by 10000 and divide while using by 10000 for storing the decimals till 4 decimal points)
- Check for end date less than expiration Date
- Using the next Business day if the Start date is non-business day
- Multiple runs (while loop till the user wants to exit)

Using the user inputs relevant data is retrieved and passed to an Option object. Using the getImpliedVolatility functionality the implied volatility is calculated, which then helps in calculating the Delta and corresponding Hedging error

Running the Code:

We require the QuantLib and Boost library directories to be included while compiling. Additionally, we need to include their corresponding libraries. Inside the zipfile we can see a midterm folder and midterm.sln file. This file when opened in Visual studio (and adding all required directories and libraries) will work well.
The midterm folder has all the header and .cpp files. It also has the midterm.exe file which can be directly used for running the code. The "midterm.exe" file should be in a folder which contains the folder "data_midterm" which has the data folder containing input data files. (op_GOOG.csv, sec_GOOG.csv,interest.csv)

3. Unittest cases for the code

Unit test Cases are written for all major fnctions in the Classes in the "test_all.h" header file. These are called at the beginning using test_all() function. Some checks Performed are:

- 1.StdNormalCDF Norm : Norm(-3) +Norm(3)==1 ; Norm(2) == 97725;
- 2.BSCalculator Class: option_price(), option_delta() and option_vega() for call & put options
- 3.Option Class: getImpliedVolatility is tested for call & put europeanoptions
- 4.PayOff Class: OptionPayOff() is tested for call & put options

4. Whether the Outcome solves the problem:

We observed that the cumulative hedging error is high for actual Market Option prices (around 12%) in 25 days then what was observed for simulated prices which follow Brownian motion. is around 1% for 100 days. Thus, the BSM pricing formula is just an approximation and not very accurate as the actual observed Stock prices do not perfectly follow Brownian motion. However, it provides suitable direction for hedging.