COSC5910

XXX

YYY

Winter 2004

Table of Content

1. The Problem

The purpose of this program is to implement two well known option pricing models (Black-Scholes model and Cox-Ross-Rubinstein model) and compare the results generated by them to the prices actually being observed in the market. Automating the inputs to allow for large samples of data is important as the models are computationally intensive. As option prices for each specific company may be affected by factors specific to that company, it is necessary to test the models against a range of different companies. The models do not incorporate the effect of dividends so they work most accurately on stocks which do not pay dividends. The program is set up to price only call options.

2. User's Guide

The main inputs for the OptionPrice program are data files obtained from the Montreal Stock Exchange (ME) website, http://www.me.org. These files contain the close of day prices on a single date for all options issued for a particular stock. The steps for downloading these files are as follows.

1. Click the <OK> box next to the input for quotes near the top center of the page.
2. Click the "Historical data" tag on the right hand side about halfway down the page.
3. Use the "Downloading end-of-day data in Excel CSV format" section at the bottom of the page. Choose an option from the drop down menu. Enter start and end dates that are the same (The OptionPrice program accepts only a single day's data at a time). Click on the <DOWNLOAD> button. Name and save the file in .txt format. We found that naming the file its stock symbol works well (note: add date identifier to name if doing multiple runs for the same option). The file should be saved in the same folder as the programs are stored in order to avoid having to specify long path names when operating the program. Samples of these files (BLD.txt, CLS.txt, JDU.txt, WJA.txt) have been provided for testing purposes.

Running OptionPrice:

1. The first prompt is for the risk free interest rate in decimal format (i.e. 4% should be entered as .04). The input for this prompt is based upon the user's expectation of what annual rate an investor could expect to earn on a no risk or relatively low risk investment. We found that .035 worked well. A default rate of .04 will be used if the user does not wish to specify a rate.
2. The user must supply the stock symbol of the stock for which the program is to be run (bld , cls, wja, jdu for the sample files). This input is not case-sensitive.
3. The name of the input file to be used must be entered.
4. The name of the output file to be used must be entered. For convenience an output file called price.txt is provided if the user does not wish to specify the filename. This filename is also provided as a default input option in the program Statistics.
5. Enter a Y (not case sensitive) if the results are to be appended to a previous run of the program. Any other input results in any previous contents of the output file being over-written. The Y must be used in order to process multiple stocks in a single run of the Statistics program.
6. The date that the data is for must be entered by the user in the format specified. (The sample data provided is all for 2004-04-27)
7. The number of calendar days is entered by the user. All stock market opening days within this number of days prior to the data date are used in the volatility calculation.
8. The user is provided with the price of the underlying stock and the calculated volatility for the specified period. Depending upon the size of the input file there may be a slight pause at this time while each option is priced and the output is written to the file.

Running Statistics:

Program Statistics must be run only using input files produced by the program OptionPrice.

1. The name of the input file must be entered. A default option matching the default of the OptionPrice program is provided.
2. The output of the program is provided on the screen. Under spread indicates that the model predicted a price that is less than the bid price observed in the market. Within spread indicates that the model has priced the option within the range of prices currently seen in the market, while over spread indicates an overpricing relative to the ask spread. The average percentage error is an indication of the expected error for each model on the data set provided. A negative number is the percentage of the bid price by which the model under-estimates while a positive number is the percentage of the ask price by which the model over-estimates.

3. Programmer's Guide

See documentation within the source code.

4. Design & Implementation

The intention was to develop a testing methodology for the two models that would allow the user to obtain statistically significant results for each method's accuracy with a minimum of inputs. The required information could not be extracted from the ME using a Java screen scraper. As the information was available in an Excel CSV format for download we modified our inputs to accept data in this format. The goal was to minimize the number of additional inputs that would be required from the user at runtime. To this end, an algorithm was developed to allow for the price of the underlying stock on the data date and the volatility to be obtained from historical data obtained from the Yahoo web-site. Appendix A outlines the procedure used to obtain a suitable date range with minimal input from the user.

The algorithm for calculating the volatility appears to be a little drawn out. Two loops are required. The first calculates the average of the period's daily price changes. The second calculates the square of the difference between each daily price change and the average daily price change. After the second loop, the sum of the squared differences is divided by the number of entries used in the calculation and the square root of the result is calculated. The result of this calculation is then multiplied by the square root of 252 in order to convert the result (which is a daily volatility) into an annualized volatility (It is assumed 252 business days in a year) as required by the pricing models.

The user is prompted to supply input and output files and to specify whether to append the output to an existing file in order to increase flexibility in testing various data sets.

Within the OptionPrice program more data fields than are currently utilized by the Statistics program are recorded in the output file. Fields such as the stock ticker symbol, the number of days to maturity of the option and the option strike price are recorded for possible future additions to the statistics program. Examples of possible additions are checking whether the models perform better on options with a longer time to expiry or a shorter time to expiry and/or checking whether they price options with a strike price near to the current price better than those with a strike price far from the current price.

The programs were developed in a modular format allowing parallel development and testing to occur. Only when the output of a developed section could be obtained in the required format was that section added to main program. The modules were developed in the following sequence:

a. B-S model

b. CRR model
c. Data extraction from ME file
d. Statistics program
e. Data extraction from Yahoo
f. Calculation of historical volatility
g. Calculation of data range from user supplied # of days.

5. Error Checking

Due to time constraints explicit error checking has only been implemented in the section where a connection to Yahoo for data extraction is established. Error messages are displayed for the user if a connection cannot be obtained or the data extraction fails.

The general structure of the programs was developed to minimize the possible sources of error. The number of inputs from the user were reduced to as few as possible and prompts with easy to follow instructions were provided. Default entries were provided where appropriate to reduce the necessary inputs even further. Input data and files were obtained from reliable sources where the data is easily obtained in a standard format. The inputs to the Statistics program are protected from many errors by utilizing only files produced by the OptionPrice program.

6. Testing

As our programs use files for inputs with a minimum of user inputs a test harness was not required. The default options on user inputs made the test runs quick to initiate.

The B-S model is a set of standard algebraic formulas that requires 5 inputs. In developing the code for this model we simply converted the formulas into Java code and prompted for each of these inputs with a hard-coded default option (later removed). Results were checked against the results from an interactive on-line calculator for the B-S model, http://www.margrabe.com/OptionPricing.html .

As the CRR model is a class obtained from the U of T finance department website, limited testing was required. We tested the CRR results for reasonableness against the results from our B-S model. The prices obtained in our first runs were wildly off indicating that there was a problem with the format of an input. As these problems were corrected the prices became similar in scale. One input required by the CRR model that is not required by the B-S model is DT (Delta T). This represents the number of time steps to be simulated in the model. As the DT is increased the accuracy of the model is supposed to increase but a significant deterioration in processing speed occurs. Trial and error was used to settle on 64 as sufficiently accurate without a significant delay.

For the file extraction modules we simply compared the extracted data visually to the file or screen source to ensure that that the correct fields were being accessed.

The volatility calculation was tested by entering the number of calendar days that would result in the 30 business days previous to the current date being downloaded from Yahoo. The current 30-day historical volatility for each stock is available on the ME web-site. We simply compared our calculated volatility to their calculated volatility. As we were accurate to greater than 2 decimal places in all cases we are confident that we have implemented the algorithms correctly.

The calculation of date ranges was tested by entering different start dates and number of days and obtaining a date range. Excel was used to subtract the starting and ending days from each other and the result compared to the number of days entered. Testing included dates that overlapped year and month ends to ensure accuracy.

7. Conclusions

The tested models seem to be about equally effective at estimating the prices of the options. They both tend to over or under estimate at the same time. Whether they over or under estimate is stock specific. The models tend to overprice call options on stocks which the market expects to go down and under price those that the market expects to increase. As larger samples of options based on different stocks are entered the bias towards over or underestimating disappears. This indicates that as general models they work well.

Both models are for an ideal world and as such are subject to the estimates used as inputs. The interest rate is an estimate while the historical volatility is how much the price of the underlying stock varied in the past. However, the market price is based upon how much the market expects the price of the underlying stock to vary in the future.

Our program works well and provides data entirely consistent with the theories taught in our finance program. We achieved essentially all of the goals as outlined in our project proposal. We decided not to calculate the implied volatility from the current option prices as this information is available directly from the ME website.

8. Code Listing

The classes OptionPrice and Statistics are forwarded with this package.

Appendix A

Calculating the Date Range for Volatility

The OptionPrice program gets a text file with a user specified number of dates in order to calculate the annualized volatility of the underlying asset. This parameter is necessary for the calculation of both the Cox Ross Rubinstein (CRR) model and the Black-Scholes (BS) model.

The Date Range Procedure

This module prompts the user to give the option data date and the number of past calendar days to be considered for the volatility calculation. The dateInput variable is the date string. The calculation of the dates uses a parsing function to recognize the year, month and day string fragments and change them into integers. They are inputYear, inputMonth and inputDate, respectively. It is important to note at this point that normal date format begins counting at January from the number 1. Java begins counting from January at 0. Therefore we use the variable inputMontinJava (inputMonth – 1) to translate the user's input into Java format. The variables inputYear, inputMonthinJava and inputDate are used to find the first date in the TargetUrl variable string that queries for the stock text file.

GregorianCalendar is used to create a calendar instance lastDate which will be used to create the final date in the date range. The GregorianCalander is set at the current date using the above three variables and the lastDate.add() method is used to find the final date in the date range. The range is defined by the variable period. The new date is placed in the format yyyy.MM.dd given by the variable string pattern using the getInstance() method in the DateFormat class. This formatted date is assigned the string variable pastDate. The variable pastDate is parsed and the variables pastYear, pastMonth and pastDate are created. Again, date format is given in the regular form, so pastMonthinJava is used to translate the variable into Java format.

These variables in addition to the ticker symbol are used in the string TargetUrl to get the text file with the specified information.