



MTHM005 - MATHEMATICAL SCIENCES PROJECT

# Pricing Asian Options

Candidate Number: 145670

Supervised by Prof. BYOTT

June 24, 2022

## Abstract

Insert some cool abstract that makes our examiner go, "hmm that's a pretty good abstract".

All files available here:

<https://github.com/leele2/Mathematics-in-Business-Project>

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	A brief history of options . . . . .	1
1.2	Standard options . . . . .	1
1.3	Asian options . . . . .	1
<b>Appendix A Matlab Files</b>		<b>4</b>
<b>Appendix B Python Files</b>		<b>6</b>

## List of Figures and Tables

1.1	Time series plot of the average daily option and future contracts trading volume per annum. Data provided by the Options Clearing Corporation (OCC) [Optne]. Source code Listing B.1 . . . . .	1
-----	--	---

# Chapter 1: Introduction

This report studies a vital financial derivative in today's markets, namely options. The option market is widely considered a venue for informed trading [Bla75, Li21, Hu14, CGM04], that is investors trading with superior knowledge of the probability distribution of share prices, through either access to private information or skillful processing of public information [Gro75].

Numerous empirical studies have shown the importance of the option market, reference and examples. The popularity of options in today's market can easily be seen by the exponential growth in their trading volume from when standardized, exchange-traded stock options were first listed in The Chicago Board Options Exchange in 1973 [Mar02], shown in Figure 1.1. In 2020 option trading volume became higher than the underlying stock volume for the first time ever [Wayne].

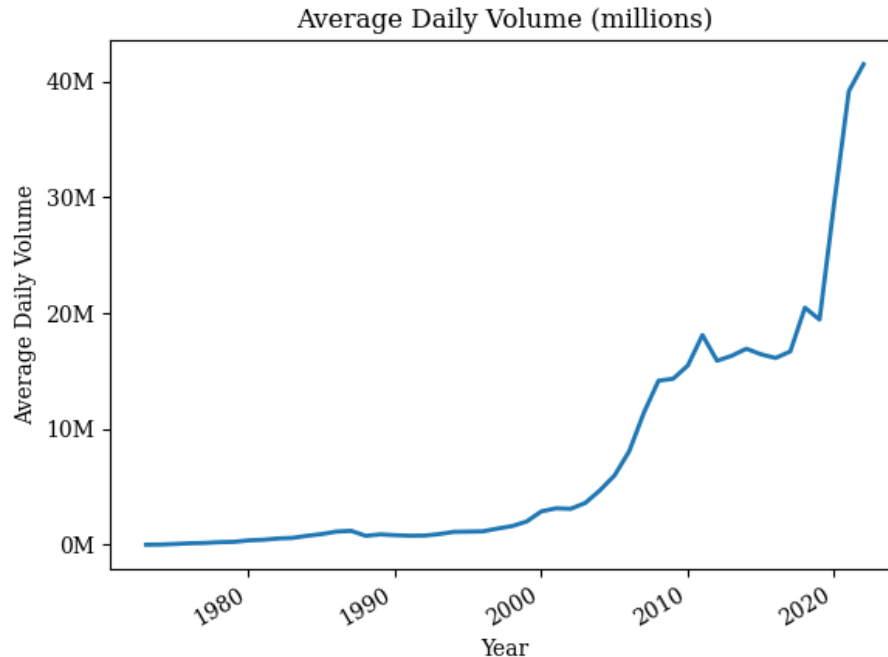


Figure 1.1: Time series plot of the average daily option and future contracts trading volume per annum. Data provided by the Options Clearing Corporation (OCC) [Optne]. Source code Listing B.1

## 1.1 A brief history of options

## 1.2 Standard options

A standard option is a contract between two parties which gives the holder the right to buy (or sell) an asset for an agreed upon (exercise) price prior to, or on a determined (expiry) time in the future; regardless of the current (spot) price. Since the holder of the contract is not obliged to exercise the contract at the expiry time, they would not hold any liability in the absence of a price to purchase the option. The problem then becomes what is the correct price to charge the holder of the option to balance this inequality of liability.

## 1.3 Asian options

Whilst standard options, namely European and American style involve using the spot price as the underlying value of the asset; this is not always the case with so-called exotic options. Exotic options differ in their payment structures, expiration dates, and/or strike prices. In the case of exotic fixed-strike price Asian options, the averaging price of the asset is used in place of the underlying asset value. This differs from fixed-price Asian options which instead use the averaging price of the asset to take place of the strike price. These are the two main variations of Asian style options but both of these can be varied further in how the averaging is calculated, for example: geometrically, arithmetically, average taken every

day or average taken at the start of each month and so on. They can be varied further by having an expiry structure matching a European or American style option.

# Bibliography

- [Bla75] Fischer Black. Fact and fantasy in the use of options. *Financial Analysts Journal*, 31(4):36–41, 1975.
- [CGM04] Sugato Chakravarty, Huseyin Gulen, and Stewart Mayhew. Informed trading in stock and option markets. *The Journal of Finance*, 59(3):1235–1257, 2004.
- [Gro75] Steven D Grossman. An application of communication theory to financial statement analysis. *Financial Review*, 10(1):12–20, 1975.
- [Hu14] Jianfeng Hu. Does option trading convey stock price information? *Journal of Financial Economics*, 111(3):625–645, 2014.
- [Li21] Keming Li. The effect of option trading. *Financial Innovation*, 7(1):1–32, 2021.
- [Mar02] J.W. Markham. *A Financial History of the United States Volume III (1970-2001) From the Age of Derivatives to the New Millennium*. A Financial History of the United States. M.E. Sharpe, 2002.
- [Optne] Options Clearing Corporation. Annual Volume and Open Interest Statistics. <https://www.theocc.com/Market-Data/Market-Data-Reports/Volume-and-Open-Interest/Historical-Volume-Statistics>, 24th June 2022 [Online]. Accessed: 08-06-2022.
- [Wayne] Wayne Duggan. Option Trading Volume Higher Than Underlying Stock Volume For First Time Ever. [https://finance.yahoo.com/news/option-trading-volume-higher-underlying-211006236.html?guccounter=1&guce\\_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLnNvbS8&guce\\_referrer\\_sig=AQAAAGAPwh\\_iWDvx0DcVe3TMvrMR9gvv7XIYDYOXVc15LPN5ZLNnVmaq1490IXGIFPOPbL2J31GH2JpvFN\\_eb7bzK4n042h7DWpHfmW5Y-EgRt9l\\_3vWsKIIB-jMSN7bCqAaI14Lkx3Jfv\\_vXhajuXKmbUnP3fZnEA5h54dj1GmRNw1G](https://finance.yahoo.com/news/option-trading-volume-higher-underlying-211006236.html?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLnNvbS8&guce_referrer_sig=AQAAAGAPwh_iWDvx0DcVe3TMvrMR9gvv7XIYDYOXVc15LPN5ZLNnVmaq1490IXGIFPOPbL2J31GH2JpvFN_eb7bzK4n042h7DWpHfmW5Y-EgRt9l_3vWsKIIB-jMSN7bCqAaI14Lkx3Jfv_vXhajuXKmbUnP3fZnEA5h54dj1GmRNw1G), 8th July 2020 [Online]. Accessed: 24-06-2022.

# Appendix A: Matlab Files

All files can be found: <https://github.com/leele2/Mathematics-in-Business-Project/tree/master/MATLAB%20Files>

Listing A.1: .././MATLAB Files/BinoAsian.m

```
1 %function BinoAsian(S0,E,T,r,sigma,N,F)
2 %% Test
3 clear; tic; S0=50; E=40; T=1.0; r=0.1; sigma=0.3; N=500; F=@(S,A)max(A-E,0);
4 %% Function to evaluate European Call option by Binomial Method
5 % Parameters:
6 % S0 = initial share price
7 % E = exercise price`
8 % T = time to expiry
9 % r = riskfree interest rate
10 % sigma = volatility
11 % N = Number of steps
12 % F = Option Payoff (European Call in given)
13 %% Calculated parameters
14 dt = T/N; %Timestep
15 u = exp(sigma*sqrt(dt)); %Up price movement
16 d = 1/u; %Down price movement
17 dis = exp(-r*dt); %Discount factor over each timestep
18 p = (1/dis - d)/(u-d); %Risk-neutral probability
19 %% Initializing Arrays and Functions
20 S = zeros(N+1,N+1); %Underlying Asset Price
21 S.k = cell(N+1,N+1); %S_max %% Cells are used so different sized vectors can be stored at each element in
    cell array
22 A = zeros(N+1,N+1); %Average of Underlying Asset Price
23 A.k = cell(N+1,N+1); %Representative averages
24 C = zeros(N+1,N+1); %Price of Option
25 C.k = cell(N+1,N+1); %Option price for given representative avg
26 %F = @(S,A)max(A-E,0); %Option Payoff (European Call)
27 %% Functions to Calculate Maximum and Minimum Representative Averages
28 A_min = @(i,j) (1/(i+1)) * (sum(S0*d.^(0:i-j)) + sum(S0*d.^(0:(j-1)+i-(2*j)))));
29 A_max = @(i,j) (1/(i+1)) * (sum(S0*u.^(0:j)) + sum(S0*u.^(0:(i-j-1)+(2*j)-i)));
30 %% Calculate Underlying asset price
31 for i = 0:N
32     for j = 0:i
33         S(i+1,j+1) = S0*u^(j)*d^(i-j);
34     end
35 end
36 %% Calculating All S_max and Representative Averages
37 for i = 0:N
38     for j = 0:i %j indexes at j+1 due to matlab not allowing C.k{:,0)
39         A.k{i+1,j+1} = zeros(j*(i-j)+1,1); %Create Vector to hold rep avgs
40         S.k{i+1,j+1} = NaN(j*(i-j),1); %Create Vector to hold S_max
41         A.k{i+1,j+1}(1) = A_max(i,j); %Assign A_max to first element in vector
42         % Paths with only up (i = j) or down movements (j = 0) or i = 1 will only have one representative
            average
43         if i < 1 || i == j || j == 0 || j*(i-j) == 1
44             S.k{i+1,j+1}(1) = S0*(u^j)*d^(i-j);
45             continue
46         end
47         %% Filling S_max Vector
48         % Setting Paramaters
49         n = j*(i-j); %Size of vector
50         rep = min([j,i-j]); %Maximum element repetition in vector
51         % Calculating unique elements in vector
52         unique = 2*sum(1:rep-1) + (n - (2*sum(1:rep-1)))/rep;
53         % First and last values
54         S.k{i+1,j+1}(1) = S0 * (u ^ j);
55         S.k{i+1,j+1}(end) = S0 * (u ^ (j - unique + 1));
56         %%The following if statement changes the final output%%
57         if n ~= 2
58             % Ascending/Descending repeated values
59             k = 2; %iterator
60             count = 2; %Number of elements filled
61             while k < rep
62                 S.k{i+1,j+1}(sum(1:k-1)+1:sum(1:k-1)+k) = repmat(S0*(u^(j-k+1)), k, 1);
63                 S.k{i+1,j+1}(n-sum(1:k-1)+1-k:n-sum(1:k-1)) = repmat(S0*(u^(j-unique+k)), k, 1);
64                 count = count + 2*k;
65                 k = k + 1;
66             end
67             % "Middle" repeated values
68             for l = 0 : (n-count)/rep - 1
69                 S.k{i+1,j+1}(count/2+1+l*rep:end) = [repmat(S0*u^(j-k+1), rep, 1);
70                     S.k{i+1,j+1}(count/2+1+l*rep+rep:end)];
71                 k = k + 1;
72             end
73         end
74         %% Calculating Representative Averages Using S_max
75         A.k{i+1,j+1}(j*(i-j) + 1) = A_min(i,j); %Assign A_min to last element in vector
76         for k = 2:j*(i-j)
```

```

77         A_k{i+1,j+1}(k) = A_k{i+1,j+1}(k-1) - (1/(i+1))* ...
78             (S_k{i+1,j+1}(k-1) - S_k{i+1,j+1}(k-1)*(d^2));
79     end
80 end
81 end
82 %% Pricing Option Value at Final Time (N)
83 for j = 0:N
84     C_k{N+1,j+1} = F(S(N+1,j+1),A_k{N+1,j+1});
85 end
86 %% Pricing Option
87 err = 1e-3;
88 for i = N-1:-1:0
89     for j = 0:i
90         C_k{i+1,j+1} = zeros(j*(i-j)+1,1);
91         for k = 1:j*(i-j)+1
92             %% Find K_u
93             Ku = ( (i+1)*A_k{i+1,j+1}(k) + u*S(i+1,j+1) )/(i+2);
94             [loc, ubound, lbound] = findInSorted(Ku,A_k{i+2,j+2},err);
95             % If found set accordingly
96             if loc > 0
97                 Cu = C_k{i+2,j+2}(loc);
98             else
99                 % If not found, interpolate between closest values
100                 Cu = C_k{i+2,j+2}(lbound)+(Ku-A_k{i+2,j+2}(lbound))*(...
101                     (C_k{i+2,j+2}(ubound)-C_k{i+2,j+2}(lbound))/...
102                     (A_k{i+2,j+2}(ubound)-A_k{i+2,j+2}(lbound)));
103             end
104             %% Find K_d
105             Kd = ( (i+1)*A_k{i+1,j+1}(k) + d*S(i+1,j+1) )/(i+2);
106             [loc, ubound, lbound] = findInSorted(Kd,A_k{i+2,j+1},err);
107             %If found set accordingly
108             if loc > 0
109                 Cd = C_k{i+2,j+1}(loc);
110             else
111                 %If not, interpolate between closest values
112                 Cd = C_k{i+2,j+1}(lbound)+(Kd-A_k{i+2,j+1}(lbound))*(...
113                     (C_k{i+2,j+1}(ubound)-C_k{i+2,j+1}(lbound))/...
114                     (A_k{i+2,j+1}(ubound)-A_k{i+2,j+1}(lbound)));
115             end
116             %% Calculate option value at previous node (i,j decreasing)
117             C_k{i+1,j+1}(k) = disf*(p*C_u + (1-p)*C_d);
118         end
119     end
120 end
121 disp(C_k{1,1})
122 toc;

```

Listing A.2: ../MATLAB Files/Functions/findInSorted.m

```

1 function [loc, A, B] = findInSorted(x,range,err)
2 %%%%%%%%%%%%%%% inputs %%%%%%%%%%%%%%%
3 % x = value to find %
4 % range = (sorted) vector to search %
5 % err = tolerance on search %
6 %%%%%%%%%%%%%%%
7 %% Outputs
8 A = 1; %left boundary
9 B = numel(range); %right boundary
10 loc = 0; %initially set to 0 implying not found
11 %% Binary Search Algorithm
12 while B-A > 1
13     mid = floor((A+B)/2);
14     if x > range(mid)
15         B = mid;
16     else
17         A = mid;
18     end
19 end
20 %% Returning returned value if found (within tolerance)
21 if abs(range(A) - x) < err
22     loc = A;
23     return
24 elseif abs(range(B) - x) < err
25     loc = B;
26     return
27 end

```

# Appendix B: Python Files

All files can be found: <https://github.com/leele2/Mathematics-in-Business-Project/tree/master/Python%20Files>

Listing B.1: ../../Python Files/OptionVolume.py

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Wed Jun  8 18:05:55 2022
4  Scraping and plotting data from theocc.com
5  @author: leeel2
6  """
7  import pandas as pd
8  import matplotlib as mpl
9  import matplotlib.pyplot as plt
10 import matplotlib.dates as mdates
11 from datetime import datetime
12 from bs4 import BeautifulSoup
13 from os import listdir
14 from pathlib import Path
15
16 def string_to_int(string):
17     if not string:
18         return 0
19     if string[0] == "$":
20         string = string[1:]
21     return int(string.replace(',', ''))
22
23 # Ensuring html file is on path
24 """
25 html file taken from:
26 https://www.theocc.com/Market-Data/Market-Data-Reports/Volume-and-Open-Interest/Historical-Volume-Statistics
27 """
28 html_file = [f for f in listdir('.') if f.endswith('.html')]
29 if len(html_file) != 1:
30     raise ValueError('should be exactly one html file in the current directory')
31 # Importing html file
32 with open(html_file[0], 'r', encoding='utf-8') as file:
33     soup = BeautifulSoup(file, 'xml')
34 # Finding Table
35 table = soup.find_all("table")
36 # Selecting Data
37 table_data = table[0].find_all("tr")
38 # Finding Table Names
39 table_names = []
40 for names in table_data[0]:
41     table_names.append(str(names.find("span").string))
42 del table_names[0]
43 del table_names[-1]
44 # Finding Headers
45 headers = []
46 for header in table_data[1]:
47     headers.append(str(header.find("span").string))
48 # Populating Data
49 Data = {}
50 for rows in table_data[2:]:
51     temp = {}
52     for i, data in enumerate(rows):
53         if i > 6 and i < 10 or i == 0:
54             temp[headers[i]] = data.get_text()
55             Date = datetime.strptime(temp["Date"], "%Y")
56             del temp["Date"]
57             Data[Date] = {k: string_to_int(v) for k, v in temp.items()}
58 # Converting data into pandas dataframes
59 Data = pd.DataFrame.from_dict(Data, orient="index")
60 # Plotting time-series
61 plt.rc('font', family='serif')
62 fig, ax1 = plt.subplots()
63 plt.gcf().set_size_inches(6, 4.5)
64 ax1.plot(Data['Options']/1e6, linewidth=2, zorder=1, label="Options")
65 # ax1.plot(Data[Data['Futures'] != 0]['Futures']/1e6, linewidth=2, zorder=1, label="Futures")
66 # ax1.legend()
67 ax1.set_title('Average Daily Volume (millions)')
68 ax1.set_xlabel('Year')
69 ax1.xaxis.set_minor_formatter(mdates.DateFormatter("%Y"))
70 ax1.set_ylabel('Average Daily Volume')
71 ax1.yaxis.set_major_formatter(mpl.ticker.StrMethodFormatter('{x:,.0f}M'))
72 fig.autofmt_xdate()
73 plt.tight_layout()
74 plt.show()
75 # Saving plot
76 cwd = str(Path(__file__).parent.parent.absolute())
77 fig.savefig(cwd + "\Latex-Files\Main\Chapters\C1\plots\OptionVolume.png", bbox_inches='tight')
```