Optimizing Financial Option Contracts - Implementation

March 30, 2024

1 Introduction

This is a continuation of a previous notebook titled *Optimizing Financial Option Contracts - Theory*. This notebook will cover the implementation details of optimizing financial options contracts using the `scipy` library.

1.1 Review

The Black-Scholes PDE is given by:

$$C(S,t) = S_t N(d_1) - Ke^{-r(T-t)}N(d_2)$$

Where:

- $C(S,t)$ is the price of the call option at time t given the price of the underlying asset is S
- $N(x)$ is the cumulative distribution function of the standard normal distribution
- $d_1 = \frac{1}{\sigma\sqrt{T-t}}\left[\ln\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right]$
- $d_2 = d_1 - \sigma\sqrt{T-t}$

Optimizing black-scholes involves finding the price to buy an option at time t which maximizes the payoff. This can be written as:

$$\begin{aligned}
&\text{Objective:} \quad \text{Minimize}_{r,\sigma} \quad (C(S,t) - C_{\text{market}})^2 \\
&\text{Subject to:} \\
&\qquad C(S,t) \geq 0 \\
&\qquad C_{\text{market}} \geq 0 \\
&\qquad r \sim \text{Distribution of Market Interest Rates} \\
&\qquad \sigma \sim \text{Distribution of Market Volatility}
\end{aligned}$$

The goal is to check if a given options contract is underpriced or overpriced in our current economic environment. This can be done by finding the optimal values of r and σ that minimize the difference between the theoretical price and the market price of the option. If there is a difference between the theoretical r and σ and the current r and σ, then this tells us that the option is either underpriced or overpriced.

2 Implementation

2.1 Assumptions

There are many ways we can optimize the Black-Scholes model. For this implementation, we will make the following assumptions:

1. The volatility is uniformly distributed between 0.1 and 1
2. The interest rate parameter is uniformly distributed between 0.005 and 0.1

2.2 Objective Function

Options pricing optimizing is a non-concave optimization problem. CVXPY is a convex optimization library, so it is not suitable for this problem. We will use SciPy optimization library to solve this problem. Therefore, we used `SciPy` optimization library, which is more powerful and flexible than `CVXPY` for this problem.

Due to the non-convex nature of this problem, `SciPy` uses Sequential Least SQuares Programming (SLSQP) to solve this problem. SLSQP iteratively approximates the objective function and constraints using quadratic models, aiming to minimize the model while satisfying constraints. It updates the solution until convergence, making it effective for nonlinear optimization problems where convex techniques may not apply.

2.3 Code

[365]:
```
import numpy as np
from scipy.stats import norm
from scipy.optimize import minimize
```

Below, we will implement the optimization of the Black-Scholes model, which takes the current price of the underlying asset, the strike price, the time to maturity, and the market price of the option as inputs. The black-scholes model will then return the theoretical price of the option. Our objective function takes the theoretical price of the option and the market price of the option as inputs and returns the squared difference between the two. We will then use the `scipy.optimize.minimize` function to find the optimal parameters that minimize the objective function.

[359]:
```
def black_scholes(S, K, T, r, sigma):
    d1 = (np.log(S / K) + (r + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    option_price = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
    return option_price

def objective_function(params, S, K, T, market_price):
    sigma, r = params

    option_price = black_scholes(S, K, T, r, sigma)
    return (option_price - market_price) ** 2
```

3 Usage

Lets try this on actual options data. We will use the website https://www.nasdaq.com/market-activity/stocks/tsla/option-chain to get current options data for Tesla stock. Below is a screenshot of the data we will use from the website. The circled row is the option we will use as an example.

TSLA Option Chain

Expiration Dates: April 2024 | Option: Composite | Strategy: Calls & Puts | Moneyness: Near the Money | Type: All (Types)

	Calls							Puts					
Exp. Date	Last	Change	Bid	Ask	Volume	Open Int.	Strike	Last	Change	Bid	Ask	Volume	Open Int.
April 5, 2024													
Apr 5	16.90	-3.72	15.80	18.05	1264	1313	160.00	0.88	+0.20	0.88	0.92	14112	30255
Apr 5	14.75	-3.56	13.70	15.80	178	188	162.50	1.20	+0.29	1.18	1.25	8711	5237
Apr 5	12.72	-3.46	12.30	13.60	846	1819	165.00	1.64	+0.43	1.60	1.67	11365	23419
Apr 5	10.85	-3.22	9.80	11.30	328	918	167.50	2.17	+0.56	2.14	2.20	6409	8239
Apr 5	8.95	-3.30	8.75	9.00	2917	3710	170.00	2.87	+0.77	2.83	2.91	31875	23035
Apr 5	7.25	-3.10	7.00	7.80	2226	3004	172.50	3.75	+1.01	3.70	3.80	18903	44371
Apr 5	5.77	-2.83	5.70	5.90	11699	6596	175.00	4.85	+1.33	4.75	4.90	39539	19766
Apr 5	4.50	-2.56	4.50	4.60	25067	4062	177.50	6.10	+1.61	6.00	6.15	13902	4852
Apr 5	3.45	-2.20	3.40	3.55	32161	17303	180.00	7.45	+1.79	7.25	7.65	9221	10478
Apr 5	2.59	-1.91	2.56	2.64	16824	7586	182.50	9.15	+2.15	8.95	9.25	583	2400
Apr 5	1.91	-1.65	1.90	1.95	40606	14632	185.00	10.84	+2.30	10.70	11.45	3144	2964
Apr 5	1.36	-1.39	1.35	1.40	39798	4903	187.50	13.15	+3.15	12.25	13.10	113	544
Apr 5	0.99	-1.12	0.97	1.00	20297	16786	190.00	14.89	+2.88	14.10	15.25	533	3519
Apr 5	0.69	-0.91	0.60	0.71	5651	4130	192.50	17.25	+3.15	16.05	18.35	70	206
April 12, 2024													
Apr 12	18.02	-3.29	16.95	18.10	71	422	160.00	1.71	+0.42	1.60	1.76	1869	7964
Apr 12	16.10	-3.20	15.15	16.60	26	12	162.50	2.15	+0.52	2.16	2.23	513	737
Apr 12	13.50	-3.75	13.15	14.80	98	1090	165.00	2.72	+0.64	2.72	2.81	4737	5916
Apr 12	12.30	-3.70	11.50	12.85	58	237	167.50	3.51	+0.89	3.40	3.55	302	672
Apr 12	10.65	-3.10	10.40	10.65	486	2102	170.00	4.27	+1.02	4.25	4.50	2330	3345
Apr 12	9.07	-2.53	8.80	9.05	662	424	172.50	5.30	+1.20	5.20	5.30	667	507
Apr 12	7.50	-2.67	7.50	7.65	2279	3048	175.00	6.35	+1.40	6.30	6.50	2919	7339
Apr 12	6.30	-2.35	6.30	6.40	4000	620	177.50	7.65	+1.65	7.60	7.75	1523	865
Apr 12	5.30	-2.15	5.20	5.30	5324	3897	180.00	9.22	+2.05	9.00	9.20	1220	1443
Apr 12	4.40	-1.85	4.25	4.40	1948	1034	182.50	10.70	+2.18	10.55	10.75	217	646
Apr 12	3.50	-1.75	3.50	3.55	2488	6576	185.00	12.45	+2.60	11.75	12.90	404	774
Apr 12	2.85	-1.50	2.81	2.90	448	612	187.50	14.04	+2.14	13.50	14.85	63	170
Apr 12	2.31	-1.29	2.26	2.33	2681	5177	190.00	16.20	+2.70	15.75	16.70	140	678
Apr 12 2	1.84	-1.10	1.80	1.88	453	547	192.50	18.32	+2.87	17.35	19.15	15	242
April 19, 2024													
Apr 19	19.71	-3.31	18.50	20.85	122	2813	160.00	3.40	+0.73	3.35	3.45	2571	15410
Apr 19	17.82	-2.93	17.30	18.15	41	257	162.50	4.05	+0.90	4.00	4.10	264	5372
Apr 19	16.20	-3.20	14.95	17.30	99	7677	165.00	4.80	+0.99	4.75	4.90	1741	16729
Apr 19	14.40	-3.00	13.45	15.65	27	579	167.50	5.75	+1.20	5.60	5.75	669	25836
Apr 19	13.05	-2.69	12.90	13.10	1239	16398	170.00	6.64	+1.34	6.55	6.70	3821	40596
Apr 19	11.65	-2.60	11.50	11.70	634	2094	172.50	7.65	+1.35	7.65	7.80	615	2203
Apr 19	10.30	-2.45	10.20	10.35	15272	34717	175.00	8.95	+1.69	8.80	8.95	16076	39585
Apr 19	9.15	-2.25	9.00	9.15	3020	1178	177.50	10.13	+1.63	10.10	10.30	1777	855
Apr 19	7.90	-2.22	7.90	8.05	4972	30390	180.00	11.54	+1.89	11.50	11.70	1476	34257
Apr 19	7.00	-2.00	6.90	7.05	681	6618	182.50	13.10	+2.10	13.05	13.20	426	1393
Apr 19	6.05	-1.80	6.05	6.20	1891	12948	185.00	14.62	+2.15	14.65	14.80	552	12858
Apr 19	5.38	-1.62	5.25	5.40	1478	1603	187.50	16.50	+2.48	16.35	16.50	608	478
Apr 19	4.64	-1.51	4.60	4.70	1946	19407	190.00	18.33	+2.58	17.60	18.45	148	14356
Apr 19	4.08	-1.24	3.95	5.10	329	679	192.50	20.00	+3.00	19.90	20.85	11	130
April 26, 2024													
Apr 26	21.23	-3.07	20.25	21.95	47	368	160.00	4.57	+0.92	4.50	4.65	8774	11513
Apr 26	17.28	-3.72	17.00	18.35	43	475	165.00	6.10	+1.10	6.05	6.25	693	6517
Apr 26	14.70	-2.40	14.50	14.75	120	1328	170.00	8.01	+1.33	7.95	8.10	740	7859
Apr 26	11.95	-2.45	11.80	12.05	554	2000	175.00	10.30	+1.60	10.20	10.45	694	2057
Apr 26	9.70	-1.99	9.50	9.70	1451	2581	180.00	12.95	+1.86	12.90	13.15	412	1885
Apr 26	7.80	-1.69	7.55	7.75	328	2166	185.00	15.96	+2.06	15.90	16.20	41	399
Apr 26 1	6.05	-1.40	5.95	6.15	632	1966	190.00	19.65	+2.50	18.80	20.05	36	313

LAST TRADE: $175.79 (AS OF MAR 28, 2024)

Below is the code which sets up the optimization problem and solves it. We first initialize a random guess for the r and σ parameters. We then set the constraints that r and σ to be within the range defined in our assumtions. We pass the inital guesses and our options contract data into the

objective function and minimize.

[367]:
```
def get_results(S, K, T, market_price):
    np.random.seed(0)

    constraints = [
        {'type': 'ineq', 'fun': lambda params: params[0] - 0.1},
        {'type': 'ineq', 'fun': lambda params: 1 - params[0]},
        {'type': 'ineq', 'fun': lambda params: params[1] - 0.005},
        {'type': 'ineq', 'fun': lambda params: 0.1 - params[1]}
    ]

    initial_guess_lambda_sigma = 0.5
    initial_guess_lambda_r = 0.04
    initial_guess = [initial_guess_lambda_sigma, initial_guess_lambda_r]

    result = minimize(objective_function, initial_guess, args=(S, K, T,␣
 ↪market_price),
                      constraints=constraints)

    if result.success:
        optimal_lambda_sigma, optimal_lambda_r = result.x

        print("Optimized rate parameter for sigma (lambda_sigma):",␣
 ↪optimal_lambda_sigma)
        print("Optimized rate parameter for interest rate (lambda_r):",␣
 ↪optimal_lambda_r)
    else:
        print("Optimization failed:", result.message)
```

4 Case Study: Options Contract 1

The first options contract, highlighted in the image above, has the following parameters:

- Strike Price: $190
- Time to Maturity: April 26, 2024 (~ 1/12 years)
- Market Price of Call Option: $5.95 (this is the bid-price, which is the price at which the buyer is willing to buy the option at)
- As of Fri. Mar 28, 2024, Tesla is trading at $175, so this is the price of the underlying asset

This contract states that the buyer has the right to buy Tesla stock at $190 on April 26, 2024. The buyer pays $5.95 for this right. If Tesla is trading above $190 on April 26, 2024, the buyer can exercise the option and make a profit. If Tesla is trading below $190, the buyer can let the option expire and lose the $5.95 paid for the option.

We plug that information into the optimization problem and solve it below.

[368]:
```
S = 175
K = 190
T = 1/12
market_price = 5.95

get_results(S, K, T, market_price)
```

```
Optimized rate parameter for sigma (lambda_sigma): 0.5520447592581064
Optimized rate parameter for interest rate (lambda_r): 0.07925645706441156
```

From the above, we see that the optimal r and σ parameters are 0.08 and 0.55 respectively. This means that, based on our model, we should buy this option if the interest rate is 8.4% and the volatility is 57%. As of March. 2024, the US baseline interest rate is around 5%, so the model didn't perform terribly, but to make this more realistics, we'd need better assumptions for the interest rate and volatility distributions.

5 Case Study: Options Contract 2

The second options contract, highlighted in the image above, has the following parameters:

- Strike Price: $192.50
- Time to Maturity: April 19, 2024 ($\sim 1/24$ years)
- Market Price of Call Option: $1.84
- As of Fri. Mar 28, 2024, Tesla is trading at $175, so this is the price of the underlying asset

We plug that information into the optimization problem and solve it below.

[369]:
```
S = 175
K = 192.50
T = 1/24
market_price = 1.84

get_results(S, K, T, market_price)
```

```
Optimized rate parameter for sigma (lambda_sigma): 0.5021255518530079
Optimized rate parameter for interest rate (lambda_r): 0.04028925142128283
```

From the above, we see that the optimal r and σ parameters are 0.04 and 0.50 respectively. This means that, based on our model, we should buy this option if the interest rate is 4.0% and the volatility is 50%.

6 Conclusion

In this report, we implemented a method to optimize financial options contracts using the Black-Scholes model and the `scipy` library. By comparing theoretical option prices with market prices, we assessed whether options were underpriced or overpriced in the current market. Our implementation utilized the `scipy.optimize.minimize` function with constraints on interest rates and volatilities. Through case studies on Tesla options contracts, we demonstrated the effectiveness of our approach. While our method provides a practical framework for option pricing, further refinements are needed

to enhance its accuracy and realism, particularly in incorporating more nuanced assumptions for market parameters.