FE621 HW1, Spring 2026, Feb 14, 2026 Name: Tzuling Hsieh

1 Part 1

1.1 Queation 1 and 2

- Write a function (program) to connect to sources and download data from yahoo finance
- download data on options and equity for the following symbol: TSLA, SPY, ^VIX
- **Bonus**: Create a program that is capable of downloading multiple assets, combine them with the associated time column, and save the data into a csv or excel file

I developed a Python script using the yfinance library to automate the retrieval of both equity and options data for TSLA, SPY, and VIX.

1. Equity Fetching: For each ticker, the script pulls the most recent price. I implemented a fallback logic: it tries to get the live market price first, but if the market is closed or the API is delayed, it automatically grabs the most recent daily closing price.
2. Smart Option Filtering: Instead of downloading every available contract, I wrote a filter to target "Monthly Options" (the 3rd Friday of the month) for the next three months. This ensures the data focuses on the most liquid contracts.
3. Bonus (Consolidation and Export): The program merges the data for all three assets into a single, structured DataFrame. I've included a function to export this combined data into a CSV or Excel file, ensuring all timestamps are aligned for easy analysis.

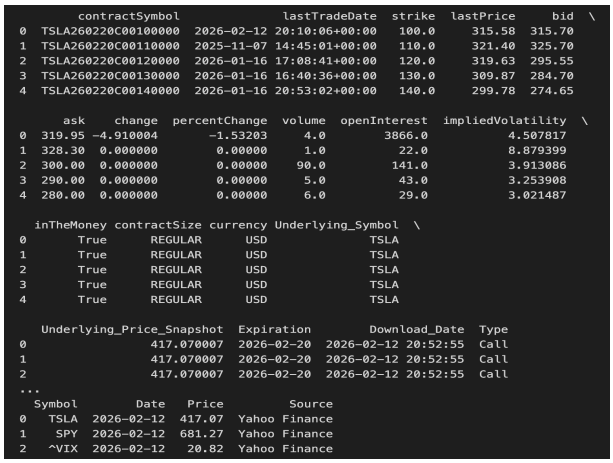
Pseudocode: Financial Data Extractor

```
FUNCTION get_data_from_yahoo(tickers):
1. Initialize two empty tables: all_equity_data and all_options_data.
2. Get the current system date.
3. FOR EACH symbol in the list of tickers:
    A. Get Equity Price:
        a. Try to fetch the real-time market price from the source.
        b. IF real-time price fails, THEN use the most recent historical closing price.
        c. Add a row (Symbol, Date, Price) to all_equity_data.
    B. Filter Option Expirations:
        a. Retrieve all available expiration dates for the symbol.
        b. FOR EACH date in expirations:
            - Calculate the time difference between today and the date.
            - IF the date is within the next 3 months AND it is a "Third Friday" (Standard Mont
                Add this date to the valid_dates list.
    C. Fetch Option Details:
        a. FOR EACH exp_date in valid_dates:
            - Download the Call and Put option chains for that date.
            - Label each contract as 'Call' or 'Put'.
            - Attach the symbol, download_date, and the current_equity_price to every row.
```

```
            - Combine these into all_options_data.
4. Data Cleaning:
    - Remove any duplicate rows from both tables.
    - Reset the row numbering (indexing).
5. RETURN all_equity_data and all_options_data.

MAIN PROGRAM:
1. DEFINE target assets: ['TSLA', 'SPY', '^VIX'].
2. CALL get_data_from_yahoo with the target assets.
3. DISPLAY a preview of the resulting data.
4. SAVE the data to CSV/Excel files.
```

Please see the appendix for the complete code.



```
         contractSymbol             lastTradeDate  strike  lastPrice     bid  \
0  TSLA260220C00100000  2026-02-12 20:10:06+00:00   100.0     315.58  315.70
1  TSLA260220C00110000  2025-11-07 14:45:01+00:00   110.0     321.40  325.70
2  TSLA260220C00120000  2026-01-16 17:08:41+00:00   120.0     319.63  295.55
3  TSLA260220C00130000  2026-01-16 16:40:36+00:00   130.0     309.87  284.70
4  TSLA260220C00140000  2026-01-16 20:53:02+00:00   140.0     299.78  274.65

      ask    change  percentChange  volume  openInterest  impliedVolatility  \
0  319.95 -4.910004       -1.53203     4.0        3866.0           4.507817
1  328.30  0.000000        0.00000     1.0          22.0           8.879399
2  300.00  0.000000        0.00000    90.0         141.0           3.913086
3  290.00  0.000000        0.00000     5.0          43.0           3.253908
4  280.00  0.000000        0.00000     6.0          29.0           3.021487

  inTheMoney contractSize currency Underlying_Symbol  \
0       True      REGULAR      USD              TSLA
1       True      REGULAR      USD              TSLA
2       True      REGULAR      USD              TSLA
3       True      REGULAR      USD              TSLA
4       True      REGULAR      USD              TSLA

   Underlying_Price_Snapshot  Expiration        Download_Date  Type
0                 417.070007  2026-02-20  2026-02-12 20:52:55  Call
1                 417.070007  2026-02-20  2026-02-12 20:52:55  Call
2                 417.070007  2026-02-20  2026-02-12 20:52:55  Call
...
  Symbol        Date   Price          Source
0   TSLA  2026-02-12  417.07  Yahoo Finance
1    SPY  2026-02-12  681.27  Yahoo Finance
2   ^VIX  2026-02-12   20.82  Yahoo Finance
```

Figure 1: Snapshot of data

1.2 Queation 3

1. Explain what is SPY and its purpose
2. Explain what is VIX and its purpose
3. Understand the options' symbols
4. Understand when each option expires

Answers:

1. SPY (SPDR $S\&P$ 500 ETF Trust)
   (a) SPY is the ticker for the SPDR S&P 500 ETF (Exchange Traded Fund). Basically, it is a fund that is designed to track the performance of the S&P 500 index. Since it is an ETF, it trades on the stock exchange just like a regular stock (e.g., Apple or Tesla), making it very easy to buy and sell.

(b) Purpose: The main purpose of SPY is to give investors exposure to the top 500 U.S. companies in a single trade. It holds the stocks in the index according to their market-cap weights (so bigger companies have a larger share). For us as quants, SPY is important because it is highly liquid, making it a standard tool for benchmarking, diversification, or hedging market risk.

2. VIX (CBOE Volatility Index)

   (a) VIX stands for the CBOE Volatility Index, often called the market's "fear gauge."

   (b) Purpose: It measures how much volatility the market expects over the next 30 days based on S&P 500 options prices. Unlike TSLA or SPY, the VIX is not an asset you can buy directly. Instead, traders use VIX options to hedge against market drops or to speculate that the market will become more unstable.

3. The option symbols(or contractSymbol) downloaded follow a standardized format containing four key pieces of information: For example, if we look at a symbol like-SPY260220C00677000, it can be broken down into:

   - Root Symbol: SPY (The underlying stock is SPY)
   - Expiration Date: 260220 (Year 2026, Month 02, Day 20)
   - Option Type: C (C for Call, P for Put)
   - Strike Price: 00677000 (This represents the strike price of 677.00)

4. For this assignment, I focused on the standard monthly options, which always expire on the third Friday of the month. Since I downloaded the data in February 2026, I specifically chose the next three expiration cycles:

   (a) Feb 20, 2026 (This month)

   (b) Mar 20, 2026 (Next month)

   (c) Apr 17, 2026 (Two months out)

1.3 Question 4

Required Parameters

1. Spot Price: The price of the underlying asset at the exact moment the option data is fetched.

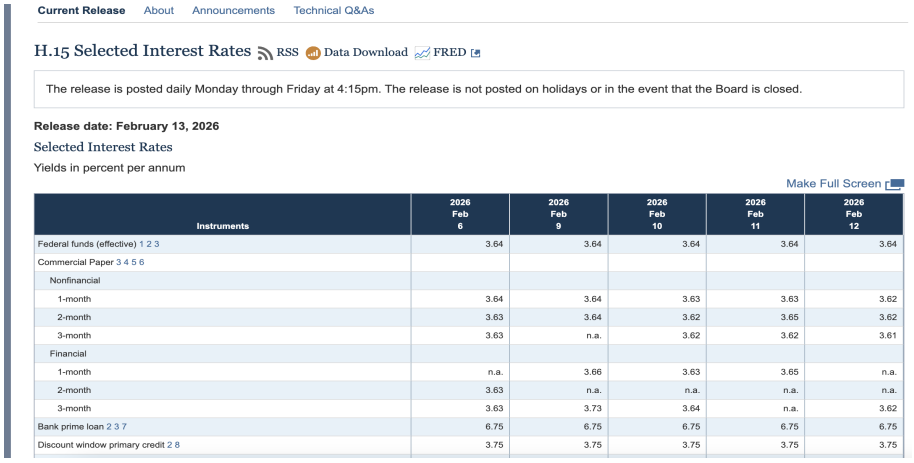
2. Risk-Free Interest Rate: You need to source a short-term rate (ideally the Federal Funds Effective Rate) from the Federal Reserve website.

3. Time to Maturity: You need to calculate the time remaining until the option expires.

Answers

1. Spot Price Tracking: Download from yahoo finance

2. Risk-Free Interest Rate: I sourced the current Federal Funds Effective Rate (3.64) from the Federal Reserve website.



Current Release About Announcements Technical Q&As

H.15 Selected Interest Rates RSS Data Download FRED

The release is posted daily Monday through Friday at 4:15pm. The release is not posted on holidays or in the event that the Board is closed.

Release date: February 13, 2026

Selected Interest Rates

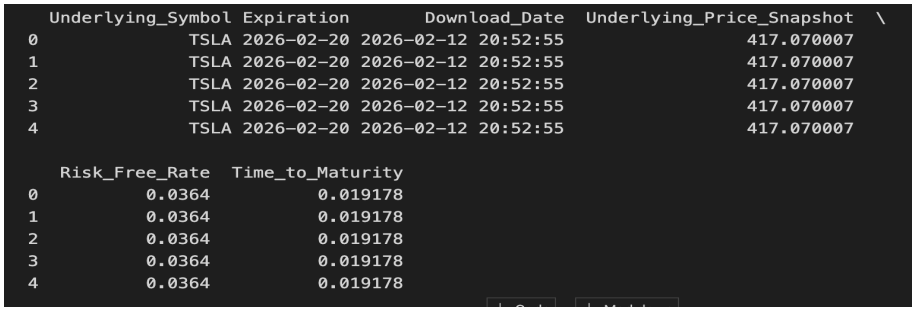
Yields in percent per annum

Make Full Screen

Instruments	2026 Feb 6	2026 Feb 9	2026 Feb 10	2026 Feb 11	2026 Feb 12
Federal funds (effective) 1 2 3	3.64	3.64	3.64	3.64	3.64
Commercial Paper 3 4 5 6					
Nonfinancial					
1-month	3.64	3.64	3.63	3.63	3.62
2-month	3.63	3.64	3.62	3.65	3.62
3-month	3.63	n.a.	3.62	3.62	3.61
Financial					
1-month	n.a.	3.66	3.63	3.65	n.a.
2-month	3.63	n.a.	n.a.	n.a.	n.a.
3-month	3.63	3.73	3.64	n.a.	3.62
Bank prime loan 2 3 7	6.75	6.75	6.75	6.75	6.75
Discount window primary credit 2 8	3.75	3.75	3.75	3.75	3.75

Figure 2: Federal Funds Effective Rate

3. Time to Maturity (T): Calculated the difference in days between the Expiration Date and the Download Date. This "Days to Maturity" is then converted into a yearly basis (Days / 365) to standardize the time variable for implied volatility analysis.



	Underlying_Symbol	Expiration	Download_Date	Underlying_Price_Snapshot	Risk_Free_Rate	Time_to_Maturity
0	TSLA	2026-02-20	2026-02-12 20:52:55	417.070007	0.0364	0.019178
1	TSLA	2026-02-20	2026-02-12 20:52:55	417.070007	0.0364	0.019178
2	TSLA	2026-02-20	2026-02-12 20:52:55	417.070007	0.0364	0.019178
3	TSLA	2026-02-20	2026-02-12 20:52:55	417.070007	0.0364	0.019178
4	TSLA	2026-02-20	2026-02-12 20:52:55	417.070007	0.0364	0.019178

Figure 3: Q4 Parameter Recording

2 Part 2

2.1 Queation 5

Black-Scholes formula

```
import numpy as np
from scipy.stats import norm
def black_scholes_price(S, K, T, r, sigma, option_type='Call'):
    if T <= 0 or sigma <= 0:
        return 0.0

    d1 = (np.log(S / K) + (r + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)

    if option_type == 'Call':
```

```
        price = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
    else:
        price = K * np.exp(-r * T) * norm.cdf(-d2) - S * norm.cdf(-d1)
    return price
```

2.2 Queation 6

Pseudocode: Bisection Method for Implied Volatility

```
FUNCTION bisection_method(target_price, S, K, T, r, type):
1. Initialize bounds for Volatility: low = 0.0001, high = 5.0.
2. Set tolerance = 1e-6.
3. WHILE (high - low) > tolerance:
    a. Calculate the midpoint: mid_sigma = (low + high) / 2.
    b. Calculate the theoretical option price using the Black-Scholes formula with mid_sigma.
    c. IF (Theoretical Price - target_price) > 0:
        The volatility is too high; set high = mid_sigma.
    d. ELSE:The volatility is too low; set low = mid_sigma.
4. RETURN mid_sigma as the Implied Volatility.
```

Implied Volatility (IV) Calculation Logic:

- Data Quality and Selection: To ensure the accuracy of the IV calculation, I applied the following filters based on the assignment requirements:
  - Liquidity Filter: Only options with non-zero volume are included to ensure prices reflect actual trading activity.
  - Price Cleaning: I required both Bid and Ask prices to be greater than zero, then calculated the Market Price as the average of the two (Mid-price).
- ATM and Moneyness Analysis
  - At-the-Money (ATM) IV: The script identifies the contract with the strike price closest to the current spot price (S_0) and solves for its specific IV.
  - Average IV (Moneyness Range):
    - Defined Moneyness as the ratio between the spot price and the strike price ($\frac{S_0}{K}$)
    - Filtered for options within the 0.9 to 1.1 range
    - Calculates the IV for every contract in this group and reports the average IV for the entire range.

Pseudocode: IV Calculation and Reporting

```
1. Filter the dataset for TSLA and SPY where volume > 0 and Bid/Ask prices are valid.
2. Calculate the Target Market Price as: (Bid + Ask)/2
3. FOR EACH target symbol:
    - Find the At-the-Money (ATM) option by minimizing |Strike - SpotPrice|.
    - Execute the Bisection Solver to find the IV for this specific ATM contract
    - Identify all contracts within the Moneyness range (0.9 <= S0/K <= 1.1)
    - Solve for the IV of each contract in that range and Average the results.
4. Display a formatted table showing the calculated IVs for both the specific ATM strike
and the broader range.
```

Symbol	Target	Strike	Type	Calculated IV
TSLA	Closest ATM	417.5	Call	0.451094
TSLA	Average ATM Range	0.9-1.1	All	0.461716
SPY	Closest ATM	681.0	Put	0.228158
SPY	Average ATM Range	0.9-1.1	All	0.391247

Figure 4: IV Calculation using Bisection method

2.3 Queation 7

Core Formulas for Newton's Method

1. Objective Function ($f(\sigma)$): Our goal is to find the volatility (σ) where the Black-Scholes model price matches the observed market price:

$$f(\sigma) = \text{BS\_Price}(\sigma) - \text{Market\_Price} = 0$$

2. The Derivative: Vega (ν)
The "derivative of the option price with respect to volatility" mentioned in the prompt is known as Vega. It measures how much the option price changes for a 1% change in volatility:

$$\text{Vega} = \frac{\partial \text{Price}}{\partial \sigma} = S_0\sqrt{T}\phi(d_1)$$

where $\phi(d_1)$: The probability density function (PDF) of the standard normal distribution

3. The Newton-Raphson Update Rule Starting with an initial guess (e.g., $\sigma_0 = 0.5$), we refine the estimate in each step using this formula:

$$\sigma_{n+1} = \sigma_n - \frac{f(\sigma_n)}{f'(\sigma_n)} = \sigma_n - \frac{\text{BS\_Price}(\sigma_n) - \text{Market\_Price}}{\text{Vega}(\sigma_n)}$$

Pseudocode: Newton's Method for IV

```
FUNCTION newton_iv_solver(target_price, S, K, T, r, option_type):
1. Initialize Parameters:
    - Set initial guess for volatility: sigma_0 = 0.5
    - Set precision limit: tolerance = 1e-6
    - Set safety limit: max_iterations = 100
2. FOR i from 1 to max_iterations:
    a. Calculate Theoretical Price: Use the Black-Scholes formula with the
    current sigma_i to find the bs_price.
    b. Calculate Vega: Compute the derivative of the price with respect to volatility
    c. Find the Error: price_error = bs_price - target_price.
    d. Convergence Check: absolute value of price_error < tolerance
    e. Update Volatility: Apply the Newton-Raphson formula:
    sigma_{i+1} = sigma_i - (price_error)/Vega
3. RETURN the final sigma after the loop completes.
```

```
Method     | Calculated IV   | Time per Option (µs)
-------------------------------------------------------
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
Vega is zero, Newton method fails.
...
Bisection  | (Many IVs)      | 3310.27 µs
Newton     | (Many IVs)      | 178.79 µs
-------------------------------------------------------
Newton method faster than Bisection method  18.5 times
```

Figure 5: Compare the time it takes to get the root with the same level of accuracy

Since Newton's Method is quadratic convergence, making it much more efficient than the Bisection method.

2.4 Question 8

Analysis of Implied Volatility and Market Observations

- TSLA vs. SPY: Idiosyncratic vs. Systematic Risk
  I observed a significant gap between the IV of TSLA and SPY. TSLA's volatility remains high at around $45\% - 47\%$, while SPY is much lower, ranging from $15\% - 24\%$. This makes sense because TSLA is an individual growth stock subject to idiosyncratic risks
  On the other hand, SPY is a diversified index of 500 companies. Most individual company risks are diversified away, leaving only systematic market risk, which naturally results in a lower IV.

- Alignment with the VIX Index
  At the time of my data collection, the VIX index was at 20.82. My calculated IV for the short-term SPY Put (Feb 20) is 24.66%, which is very close to the VIX level. Since the VIX is essentially the "fear gauge" derived from S&P 500 options, this consistency confirms that my model is accurately capturing the market's expectation of near-term volatility.

- Term Structure: The Impact of Maturity
  I noticed that as maturity increases, SPY's IV actually decreases (from 24.6% in Feb

```
Implied Volatility Summary Table
   Underlying_Symbol Expiration  Type  Average_ATM_IV
0                SPY 2026-02-20  Call        0.231898
1                SPY 2026-02-20   Put        0.246628
2                SPY 2026-03-20  Call        0.175213
3                SPY 2026-03-20   Put        0.199805
4                SPY 2026-04-17  Call        0.153139
5                SPY 2026-04-17   Put        0.180859
6               TSLA 2026-02-20  Call        0.463717
7               TSLA 2026-02-20   Put        0.470925
8               TSLA 2026-03-20  Call        0.452722
9               TSLA 2026-03-20   Put        0.452108
10              TSLA 2026-04-17  Call        0.461318
11              TSLA 2026-04-17   Put        0.458411
============================
vix value:  2093    20.82
Name: Underlying_Price_Snapshot, dtype: float64
```

Figure 6: Implied Volatility Summary Table

to 18.0% in April). This suggesting the market is worried about short-term events but expects things to calm down in the long run.
TSLA's IV stays consistently high (around 45-47%) regardless of the expiration date. This implies that the market expects high volatility for TSLA to be a permanent feature, both now and in the future.

- Volatility Skew (Put vs. Call) For SPY, the Put IV is higher than the Call IV. This represents the "Volatility Skew." It shows that investors are willing to pay a premium for Puts as "crash insurance" to protect their portfolios. For TSLA, the IVs for Puts and Calls are nearly the same.

2.5 Question 9

Put-Call Parity
For a European-style option on a non-dividend-paying stock, the Put-Call Parity relationship is defined as:

$$C + Ke^{-rT} = P + S_0$$

- Synthetic Call Price (C_{theory}): $C = P_{market} + S_0 - Ke^{-rT}$
- Synthetic Put Price (P_{theory}): $P = C_{market} - S_0 + Ke^{-rT}$

Pseudocode for Put-Call Parity Implementation

```
1. Synthetic Price Derivation
    - Apply the Put-Call Parity formula
        - Calculate Theoretical Call
        - Calculate Theoretical Put
2. Spread Validation and Accuracy Reporting
    - IF Bid_market <= Price_theory <= Ask_market, THEN mark as "In Spread".
```

From the results in the Put-Call Parity Check Table, we observe that the "Correct Rate" for Call options is 48.64%, while for Put options it is significantly lower at 22.44%. The American style problem was probably the biggest factor. The formula we used is strictly

```
=== Put-Call Parity Check Table ===
   Underlying_Symbol Expiration  strike  Market_Price_C  Call_Parity_Price  \
46              TSLA 2026-02-20   397.5          23.825          23.772398
47              TSLA 2026-02-20   400.0          21.850          21.799143
48              TSLA 2026-02-20   402.5          19.950          19.900887
49              TSLA 2026-02-20   405.0          18.050          18.077632
50              TSLA 2026-02-20   407.5          16.375          16.329377

    bid_C  ask_C  Call_In_Spread  Market_Price_P  Put_Parity_Price  bid_P  \
46  23.75  23.90           True           3.925          3.977602   3.90
47  21.75  21.95           True           4.450          4.500857   4.40
48  19.85  20.05           True           5.050          5.099113   5.00
49  17.90  18.20           True           5.725          5.697368   5.70
50  16.30  16.45           True           6.475          6.520623   6.45

    ask_P  Put_In_Spread
46   3.95          False
47   4.50          False
48   5.10           True
49   5.75          False
50   6.50          False

Call Parity correct rate: 48.64%
Put Parity correct rate: 22.44%
```

Figure 7: Put-Call Parity Check Table

for European options, but TSLA and SPY use American options. For American Puts, the right to exercise early is very valuable, especially when interest rates are high or the stock is volatile. This "early exercise premium" pushes market Put prices above the theoretical parity price, which explains why my calculated prices often missed the spread.

2.6 Question 10

We use specifically for TSLA options for example.

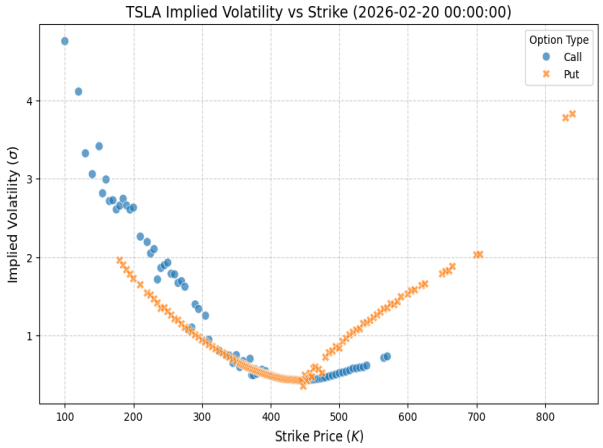


Figure 8: Implied Volatility vs Strike

- Implied Volatility vs Strike
  The Implied Volatility (IV) exhibits a classic "U-shaped" curve. he IV reaches its minimum near the At-The-Money (ATM) strike (around 450) and increases sharply

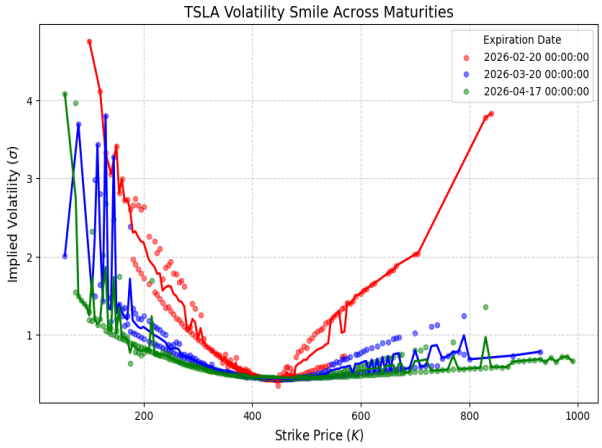


Figure 9: Volatility Smile Across Maturities

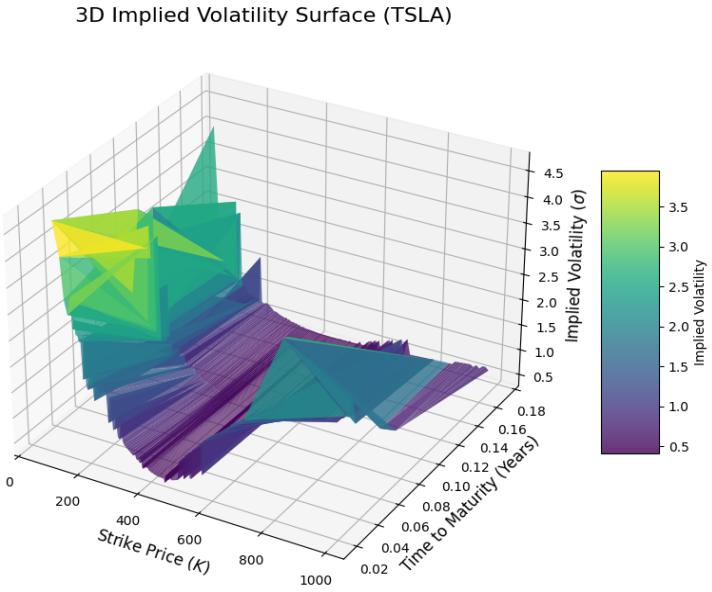


Figure 10: 3D Implied Volatility Surface

as the strike moves toward Deep-OTM or Deep-ITM territory, peaking above 4.0. This suggests that the market prices in a much higher probability of "extreme events" than what is assumed by the normal distribution in the Black-Scholes model.

- Multi-Maturity Comparison (Term Structure)
  The red line (shortest maturity) is significantly steeper and more volatile; the green line (longest maturity) is noticeably flatter. This illustrates the Term Structure of Volatility. Short-term IV is highly sensitive to immediate market shocks, whereas long-term expectations tend to stabilize as they mean-revert over time.

- 3D Volatility Surface (Bonus)
  The surface forms a deep valley where the floor represents ATM options. The steepness of the walls increases as maturity decreases, proving that IV is dynamic and varies across both price and time dimensions.

2.7 Question 11

- Black-Scholes Exact Formulas
  - Delta (Δ): Measures the rate of change of the option price with respect to changes in the underlying asset's price (S).
    $$\Delta = N(d_1)$$
  - Vega (ν): Measures the sensitivity to the asset's volatility (σ).
    $$\nu = S\sqrt{T}N'(d_1)$$
  - Gamma (Γ): Measures the rate of change in Delta with respect to changes in the underlying price.
    $$\Gamma = \frac{N'(d_1)}{S\sigma\sqrt{T}}$$

- Numerical Approximation
  I used the Central Difference method to approximate these derivatives, which generally provides higher accuracy than forward or backward differences. Using a small step size (ϵ):
  - Delta Approx: $\frac{C(S+\epsilon)-C(S-\epsilon)}{2\epsilon}$
  - Vega Approx: $\frac{C(\sigma+\epsilon)-C(\sigma-\epsilon)}{2\epsilon}$
  - Gamma Approx: $\frac{C(S+\epsilon)-2C(S)+C(S-\epsilon)}{\epsilon^2}$

Greeks Comparison Table

	Strike	Delta_Ex	Delta_App	Diff_D	Vega_Ex	Vega_App	Diff_V	Gamma_Ex	Gamma_App	Diff_G
0	100.0	0.993773	0.993773	0.0	1.014870	1.014870	0.0	0.000064	0.000064	0.0
1	120.0	0.993316	0.993316	0.0	1.080548	1.080548	0.0	0.000079	0.000079	0.0
2	130.0	0.997143	0.997143	0.0	0.505755	0.505755	0.0	0.000046	0.000046	0.0
3	140.0	0.997368	0.997368	0.0	0.469520	0.469520	0.0	0.000046	0.000046	0.0
4	150.0	0.991815	0.991815	0.0	1.291708	1.291708	0.0	0.000113	0.000113	0.0

Figure 11: Greeks Comparison Table

2.8 Question 12

Predicting DATA2 Prices using DATA1 IV

	Underlying_Symbol	Expiration	Type	strike	Calculated_IV	Market_Price_Actual	BS_Price_Theory	Price_Error
0	TSLA	2026-02-20	Call	100.0	4.754973	319.950	318.192693	-1.757307
1	TSLA	2026-02-20	Call	110.0	NaN	327.000	NaN	NaN
2	TSLA	2026-02-20	Call	120.0	4.111580	300.225	298.142526	-2.082474
3	TSLA	2026-02-20	Call	130.0	3.324528	290.175	287.718939	-2.456061
4	TSLA	2026-02-20	Call	140.0	3.058768	280.275	277.694022	-2.580978
5	TSLA	2026-02-20	Call	150.0	3.414462	269.775	268.166971	-1.608029
6	TSLA	2026-02-20	Call	155.0	2.814554	265.175	262.718847	-2.456153
7	TSLA	2026-02-20	Call	160.0	2.990751	259.775	257.917840	-1.857160
8	TSLA	2026-02-20	Call	165.0	2.715063	254.900	252.768535	-2.131465
9	TSLA	2026-02-20	Call	170.0	2.725477	249.650	247.843109	-1.806891

MAE: $1.1967

Figure 12: Predicting DATA2 Prices using DATA1 IV

The MAE of 1.1967 is relatively small compared to the absolute price of the options (which are mostly ¿250). This suggests that implied volatility is quite stable over a 24-hour period and acts as a strong persistence signal for short-term pricing. By updating the spot price and time to maturity, the model effectively accounts for the Delta and Theta effects, confirming that the Black-Scholes framework is robust for estimating next-day prices provided that volatility does not undergo a regime shift.

3 Part 3

3.1 Queation (a)

General RelationshipsFor a Constant Product Market Maker (CPMM), the pool satisfies the constant product rule at all times:

$$x_{t+1}y_{t+1} = x_t y_t = k$$

The spot price in the pool is defined as:

$$P_{t+1} = \frac{y_{t+1}}{x_{t+1}}$$

From these two equations, we can express the reserve quantities x_{t+1} and y_{t+1} in terms of the new price P_{t+1} and the constant k:

$$y_{t+1} = P_{t+1}x_{t+1} \Rightarrow x_{t+1}(P_{t+1}x_{t+1}) = k \Rightarrow x_{t+1} = \sqrt{\frac{k}{P_{t+1}}}$$

$$y_{t+1} = \sqrt{k \cdot P_{t+1}}$$

Case 1: $S_{t+1} > \frac{P_t}{1-\gamma}$ (BTC is more expensive outside)
In this case, arbitrageurs will buy BTC from the pool (reducing x) and sell it on the external market. They inject USDC (y) into the pool.The no-arbitrage boundary condition requires that after the trade, the pool price adjusted for fees matches the external price:

$$\frac{P_{t+1}}{1-\gamma} = S_{t+1} \Rightarrow P_{t+1} = S_{t+1}(1-\gamma)$$

1. Derive Δx (Amount of BTC bought from pool):The relationship for x update is $x_{t+1} = x_t - \Delta x$. Thus:

   $$\Delta x = x_t - x_{t+1}$$

   Substituting $x_{t+1} = \sqrt{\frac{k}{P_{t+1}}}$ and using the boundary condition for P_{t+1}:

   $$\Delta x = x_t - \sqrt{\frac{k}{S_{t+1}(1-\gamma)}}$$

2. Derive Δy (Amount of USDC sold to pool):The relationship for y update involves the fee: $y_{t+1} = y_t + (1-\gamma)\Delta y$. Thus:

   $$\Delta y = \frac{y_{t+1} - y_t}{1-\gamma}$$

   Substituting $y_{t+1} = \sqrt{kP_{t+1}}$ and the boundary condition:

   $$\Delta y = \frac{\sqrt{kS_{t+1}(1-\gamma)} - y_t}{1-\gamma}$$

Case 2: $S_{t+1} < P_t(1-\gamma)$ (BTC is cheaper outside)
In this case, arbitrageurs will buy BTC from the external market and sell it to the pool (increasing x). They take USDC (y) out of the pool.The no-arbitrage boundary condition is:

$$P_{t+1}(1-\gamma) = S_{t+1} \Rightarrow P_{t+1} = \frac{S_{t+1}}{1-\gamma}$$

1. Derive Δy (Amount of USDC bought from pool):The relationship for y update is $y_{t+1} = y_t - \Delta y$. Thus:

   $$\Delta y = y_t - y_{t+1}$$

   Substituting $y_{t+1} = \sqrt{kP_{t+1}}$ and the boundary condition:

   $$\Delta y = y_t - \sqrt{\frac{kS_{t+1}}{1-\gamma}}$$

2. Derive Δx (Amount of BTC sold to pool):The relationship for x update involves the fee: $x_{t+1} = x_t + (1-\gamma)\Delta x$. Thus:

   $$\Delta x = \frac{x_{t+1} - x_t}{1-\gamma}$$

   Substituting $x_{t+1} = \sqrt{\frac{k}{P_{t+1}}}$ and the boundary condition $P_{t+1} = \frac{S_{t+1}}{1-\gamma}$:

   $$x_{t+1} = \sqrt{\frac{k}{\frac{S_{t+1}}{1-\gamma}}} = \sqrt{\frac{k(1-\gamma)}{S_{t+1}}}$$

   Therefore:

   $$\Delta x = \frac{\sqrt{\frac{k(1-\gamma)}{S_{t+1}}} - x_t}{1-\gamma}$$

Summary of Results for Part (a)
The swap sizes Δx and Δy derived from the boundary conditions are:

- If $S_{t+1} > \frac{P_t}{1-\gamma}$:

  $$\Delta x = x_t - \sqrt{\frac{k}{S_{t+1}(1-\gamma)}}, \quad \Delta y = \frac{\sqrt{kS_{t+1}(1-\gamma)} - y_t}{1-\gamma}$$

- If $S_{t+1} < P_t(1-\gamma)$:

  $$\Delta x = \frac{\sqrt{\frac{k(1-\gamma)}{S_{t+1}}} - x_t}{1-\gamma}, \quad \Delta y = y_t - \sqrt{\frac{kS_{t+1}}{1-\gamma}}$$

3.2 Queation (b)

To approximate the expected fee revenue $\mathbf{E}[R(S_{t+1})]$, we use the Trapezoidal Rule for numerical integration.

1. Distribution of External Price S_{t+1}
   Since S_{t+1} follows a Geometric Brownian Motion (GBM), the logarithm of the price is normally distributed:

$$\ln S_{t+1} \sim \mathcal{N}\left(\ln S_t - \frac{1}{2}\sigma^2\Delta t, \sigma^2\Delta t\right)$$

   Let $x = \ln S_{t+1}$. The probability density function of S_{t+1} is log-normal.

2. Numerical Setup
   We implement the integration over a grid of potential future prices S.

   - Grid Range: To capture the probability mass effectively, we define the range for $\ln S_{t+1}$ from $\mu - 8\hat{\sigma}$ to $\mu + 8\hat{\sigma}$, where $\hat{\sigma} = \sigma\sqrt{\Delta t}$.
   - Discretization: We divide this range into $N = 20,000$ sub-intervals using linearly spaced points $s_0, s_1, \ldots, s_N$
   - Revenue Calculation: For each point s_i on the grid, we calculate the revenue $R(s_i)$ using the swap amounts derived in Part (a):
     - If $s_i > \frac{P_t}{1-\gamma}$: Revenue comes from $\gamma\Delta y$.
     - If $s_i < P_t(1-\gamma)$: Revenue comes from $\gamma\Delta x \cdot s_i$ (converted to USDC).
     - Otherwise: Revenue is 0 (inside the no-arbitrage band).

3. Trapezoidal Rule Implementation
   The expected revenue is approximated as:

$$\mathbf{E}[R] \approx \sum_{i=0}^{N-1} \frac{R(s_i)f(s_i) + R(s_{i+1})f(s_{i+1})}{2} \cdot (s_{i+1} - s_i)$$

   where $f(s)$ is the log-normal PDF of the external price.

Pseudocode: Expected Fee Revenue Calculation (Trapezoidal Rule)

```
Inputs:
  S_t (current price), sigma (volatility), gamma (fee rate),
  dt (time step), N (number of grid points)

1. Setup Grid for S_{t+1}:
   mu_log = ln(S_t) - 0.5 * sigma^2 * dt
   sigma_log = sigma * sqrt(dt)

   Define range [S_min, S_max] covering +/- 8 standard deviations:
   S_grid = linspace(exp(mu_log - 8*sigma_log), exp(mu_log + 8*sigma_log), N)
   ds = (S_max - S_min) / (N - 1)
```

```
2. Define Revenue Function R(s):
   Function GetRevenue(s):
      If s > P_t / (1 - gamma):
         Return gamma * Delta_y(s)   // Case 1: Arb buys BTC
      Else If s < P_t * (1 - gamma):
         Return gamma * Delta_x(s) * s // Case 2: Arb sells BTC (converted to USDC)
      Else:
         Return 0                     // Inside no-arbitrage band

3. Compute PDF values:
   For each s in S_grid:
      f(s) = LognormalPDF(s, mu_log, sigma_log)

4. Numerical Integration (Trapezoidal Rule):
   Integral = 0
   For i from 0 to N-2:
      R_left  = GetRevenue(S_grid[i])
      R_right = GetRevenue(S_grid[i+1])

      Area = 0.5 * (R_left * f(S_grid[i]) + R_right * f(S_grid[i+1])) * ds
      Integral = Integral + Area

Output: Integral (Approximated Expected Revenue)
```

3.3 Queation (c)

```
Table Results:
Sigma      | Gamma=0.1%   | Gamma=0.3%   | Gamma=1.0%   | Best Gamma
-------------------------------------------------------------------
0.2        | 0.003685     | 0.008522     | 0.009430     | 0.01
0.6        | 0.011923     | 0.032983     | 0.081082     | 0.01
1.0        | 0.020061     | 0.057384     | 0.160690     | 0.01
```

Figure 13: Optimal Fee Rate under different volatilities

Based on the scatter plot, we observe a strong positive linear relationship between volatility (σ) and the optimal fee rate (γ^*).
As volatility increases, the optimal fee rate increases. When volatility is high, the external price S_{t+1} is more likely to make large moves away from the pool price P_t. Large price deviations mean that arbitrage opportunities are more frequent and more profitable.
Consequently, the pool can afford to charge a higher fee (γ). A higher fee widens the "no-arbitrage band" $[P_t(1-\gamma), \frac{P_t}{1-\gamma}]$, but because the volatility is high enough to cross this wider band, the pool captures higher fees per trade without completely discouraging arbitrage volume.
Conversely, when volatility is low, the pool must lower fees to narrow the band; otherwise, the external price would rarely move enough to trigger an arbitrage trade, resulting in zero revenue.

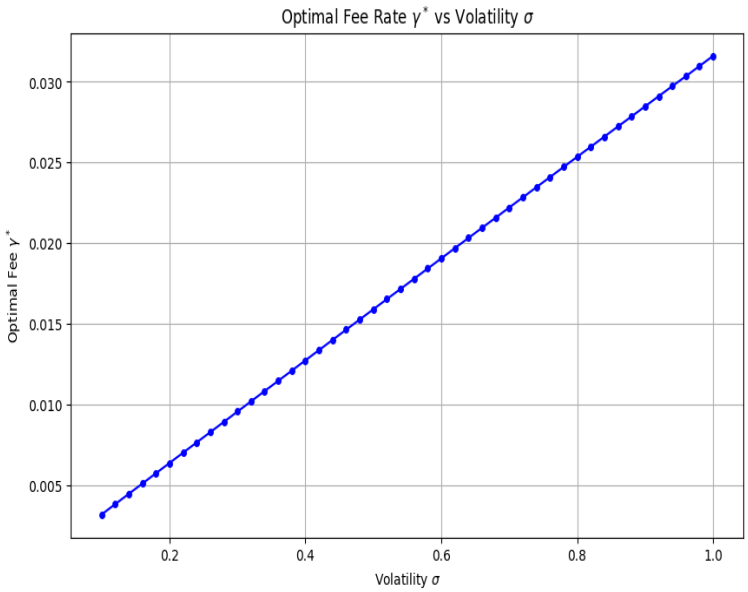


Figure 14: Optimal Fee Rate vs. Volatility Plot

4 Part 4

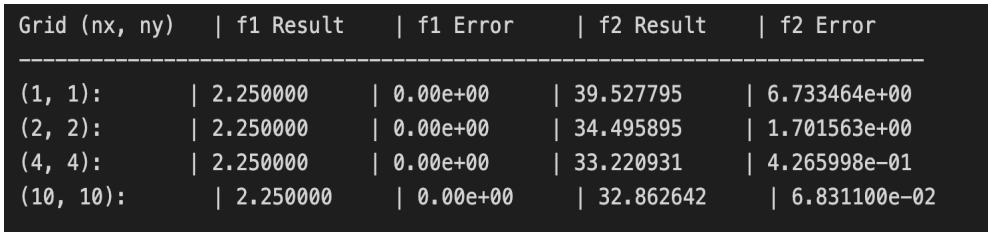
4.1 Question 1

For $f_1(x, y) = xy$:

$$\begin{aligned} I_1 &= \int_0^1 \int_0^3 xy \, dy \, dx \\ &= \left(\int_0^1 x \, dx \right) \left(\int_0^3 y \, dy \right) \\ &= \left[\frac{x^2}{2} \right]_0^1 \cdot \left[\frac{y^2}{2} \right]_0^3 \\ &= \frac{1}{2} \frac{9}{2} = \mathbf{2.25} \end{aligned}$$

For $f_2(x, y) = e^{x+y}$:

$$\begin{aligned} I_2 &= \int_0^1 \int_0^3 e^{x+y} \, dy \, dx = \int_0^1 \int_0^3 e^x \cdot e^y \, dy \, dx \\ &= \left(\int_0^1 e^x \, dx \right) \left(\int_0^3 e^y \, dy \right) \\ &= [e^x]_0^1 \cdot [e^y]_0^3 \\ &= (e^1 - 1)(e^3 - 1) \\ &\approx 1.71828 \cdot 19.0855 \approx \mathbf{32.794331} \end{aligned}$$



Grid (nx, ny)	f1 Result	f1 Error	f2 Result	f2 Error
(1, 1):	2.250000	0.00e+00	39.527795	6.733464e+00
(2, 2):	2.250000	0.00e+00	34.495895	1.701563e+00
(4, 4):	2.250000	0.00e+00	33.220931	4.265998e-01
(10, 10):	2.250000	0.00e+00	32.862642	6.831100e-02

Figure 15: summarizes the numerical integration results and the absolute error

4.2 Question 2

Pseudocode of Numerical Integration

```
Function Numerical_Integral(f, x_range, y_range, dx, dy):
    Initialize total_sum = 0
    Determine number of steps nx, ny based on dx, dy

    For each cell i from 0 to nx-1, j from 0 to ny-1:
        Define coordinates:
            x_left, x_right, y_bottom, y_top
            x_mid = (x_left + x_right) / 2
            y_mid = (y_bottom + y_top) / 2

        Calculate terms based on Hint 1:
            Corners = f(x_left, y_bottom) + f(x_left, y_top) + ...
            Edges   = 2 * (f(x_mid, y_bottom) + f(x_mid, y_top) + ...)
            Center  = 4 * f(x_mid, y_mid)

        total_sum += (Corners + Edges + Center)

    Result = total_sum * (dx * dy / 16)
    Return Result
```

We chose four sets of $(\Delta x, \Delta y)$ representing different grid densities to observe the convergence behavior:

- Grid 1 (1x1): $\Delta x = 1.0, \Delta y = 3.0$ (Coarsest)
- Grid 2 (2x2): $\Delta x = 0.5, \Delta y = 1.5$
- Grid 3 (4x4): $\Delta x = 0.25, \Delta y = 0.75$
- Grid 4 (10x10): $\Delta x = 0.1, \Delta y = 0.3$ (Finest)

Comments on the results:

1. Perfect Accuracy for f_1:
   For the function $f_1(x, y) = xy$, the numerical error is exactly 0.0 for all grid sizes, even the coarsest one.
   The integration formula provided in Hint 1 involves midpoints and a specific weighting

(1, 2, 4) that resembles higher-order methods like Simpson's rule. Since $f_1(x, y) = xy$ is a bilinear polynomial, this numerical scheme is capable of integrating it exactly, unlike the standard basic Trapezoidal rule which might introduce errors.

2. Convergence for f_2:
   For the exponential function $f_2(x, y) = e^{x+y}$, the error is non-zero but decreases rapidly as we refine the mesh (decrease $\Delta x, \Delta y$). As shown in the table, increasing the grid from 1×1 to 10×10 reduces the error from approximately 6.73 to 0.068, demonstrating that the approximation converges to the true value as the step size approaches zero.