Homework 1

FE621 Computational Finance

due 23:59, Sunday February 15, 2026

For all the problems in this assignment you need to design and use a computer program, output results and present the results in nicely formatted tables and figures. The computer program may be written in any programming language you want. Please write comments to all the parts of your code. They are a requirement and they will be graded.

You need to submit a PDF containing the report. Please use a word processor such as Microsoft Word, LaTeX, or whatever Apple uses to create your report. You will be judged by the quality of the writing and the interpretation of the results.

Part 1. (20 points) Data gathering component

1. Write a function (program) to connect to sources and download data from one of the following sources:

   (a) GOOGLE finance http://www.google.com/finance

   (b) Yahoo Finance http://finance.yahoo.com

   (c) Bloomberg

   Notes. For extra credit you can turn in code to download data from the other two sources. Please note that the program needs to download both option data and equity data. For this problem (and only for this problem) you may use any built in function or toolbox that will facilitate your work. The data will have to be clean (no duplicated values, only one exchange, every column labeled properly, in other words, consolidated).

Note on Bloomberg data. For the Bloomberg source, access to one of Bloomberg terminals in the lab is required. If you use Bloomberg data, you may use the API to download data in Excel automatically and organize the data. However, this should be accomplished with an automatic script. If you use R to interface with the Bloomberg data, a useful package for that is Rblpapi, but there are other packages. For the online students, who do not have access to Bloomberg terminals, you may read about the package quantmod in R to download yahoo and google data automatically.

Bonus (5 points) 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.

2. With the function created in problem 1, download data on options and equity for the following symbols:

   - TSLA
   - SPY
   - ^VIX

   for two consecutive days (does not matter when, but no later than February 14^{th}) during the trading day (9:30am to 4:00pm). Please record the asset values (both TSLA and SPY) at the time when downloading is done. Please do the same with the ^VIX. For the options please note that the traditional options are maturing third Friday of each month. However note there are a lot more options available to download. Please give your quant student explanation why there are so many maturities available. For the assignment please download option chain data for the next three months (options expiring on the third Friday of the month).

   We shall refer to the data sets gathered in the two consecutive days as DATA1 (for the first day) and DATA2 (for the second day) respectively throughout this assignment and the following ones.

3. Write a paragraph describing the symbols you are downloading data for. Explain what is SPY and its purpose. (Hint: look up the definition of an ETF). Explain what is ^VIX and its purpose. Understand the

options' symbols. Understand when each option expires. Write this information and turn it in.

4. The following items will also need to be recorded:

   - The underlying equity, ETF, or index price at the exact moment when the rest of the data is downloaded.
   - The short-term interest rate which may be obtained here: http://www.federalreserve.gov/releases/H15/Current/. There are a lot of rates posted on the site - they are all yearly, *they are in percents and need to be converted to numbers.* There is no theoretical recommendation on which to use, I used to use 3-months Treasury bills which are not available anymore. Since then I have been using the "Federal funds (effective)" rate but you can go ahead and try others. You should remember to be consistent in your choice. Also, make sure that the interest rate that you use is for the same day when the data you use for the implied volatility was gathered and note that the data is typically quoted in percents (you will need numbers). The same site has a link to past (historical) interest rates.
   - Time to Maturity.

Part 2. (50 points) Analysis of the data.

5. Using your choice of computer programming language implement the Black-Scholes formulas as a function of current stock price S_0, volatility σ, time to expiration $T - t$ (in years), strike price K and short-term interest rate r (annual). Please note that no toolbox function is allowed but you may call the normal CDF function (e.g., `pnorm` in `R` or `scipy.stats.norm.cdf` in `Python`).

6. Implement the Bisection method to find the root of arbitrary functions. Apply this method to calculate the implied volatility on the first day you downloaded (DATA1). For this purpose use as the option value the average of bid and ask price if they both exist (and if their corresponding volume is nonzero). Also use a tolerance level of 10^{-6}. Report the implied volatility at the money (for the option with strike price closest to the traded stock price). You need to do it for both the stock and the

ETF data you have (you do not need to do this for ^VIX). Then average all the implied volatilities for the options between in-the-money and out-of-the-money.
Note. There is no clearly defined boundary between options at-the-money and out-of-the-money or in-the-money options. If we define moneyness as the ratio between S_0 the stock price today and K the strike price of the option some people use values of moneyness between 0.95 and 1.05 to define the options at the money. Yet, other authors use between 0.9 and 1.1. Use these guidelines if you wish to determine which options' implied volatilities should be averaged.

7. Implement the Newton method/Secant method or Muller method to find the root of arbitrary functions. You will need to discover the formula for the option's derivative with respect to the volatility σ. Apply these methods to the same options as in the previous problem. Compare the time it takes to get the root with the same level of accuracy.

8. Present a table reporting the implied volatility values obtained for every maturity, option type and stock. Also compile the average volatilities as described in the previous point. Comment on the observed difference in values obtained for TSLA and SPY. Compare with the current value of the ^VIX. Comment on what happens when the maturity increases. Comment on what happen when the options become in the money respectively out of the money.

9. For each option in your table calculate the price of the different type (Call/Put) using the Put-Call parity (please see Section 4 from [2]). Compare the resulting values with the BID/ASK values for the corresponding option if they exist.

10. Consider the implied volatility values obtained in the previous parts. Create a 2 dimensional plot of implied volatilities versus strike K for the closest to maturity options. What do you observe? Plot all implied volatilities for the three different maturities on the same plot, where you use a different color for each maturity. In total there should be 3 sets of points plotted with different color. (**Bonus** 5 Points) Create a 3D plot of the same implied vols as a function of both maturity and strike, i.e.: $\sigma(\tau_i, K_j)$ where $i = 1, 2, 3$, and $j = 1, 2, \ldots, 20$.

11. (Greeks) Calculate the derivatives of the call option price with respect to S (Delta), and σ (Vega) and the second derivative with respect to S (Gamma). First use the Black Scholes formula then approximate these derivatives using an approximation of the partial derivatives. Compare the numbers obtained by the two methods. Output a table containing all derivatives thus calculated.

12. Next we will use the second dataset DATA2. For each strike price in the data use the Stock price for the same day, the implied volatility you calculated from DATA1 and the current short-term interest rate (corresponding to the day on which DATA2 was gathered). Use the Black-Scholes formula, to calculate the option price.

Part 3. (30 points) Numerical Integration of real-valued functions. AMM Arbitrage Fee Revenue

AMMs are decentralized exchanges that quote prices using pool reserves rather than an order book. Compared to Traditional Finance order books, a key advantage is **continuous liquidity from the pool without needing a matching counterparty**. Liquidity Providers (LPs) earn revenue mainly from **swap fees**, so selecting a good fee rate γ under different volatility levels matters [5].

For simplicity we consider the following Constant Product Market Maker (CPMM) for a **BTC/USDC** pool. These are the pool elements:

- x_t: BTC reserves at time t
- y_t: USDC reserves at time t
- pool mid price is calculated as $P_t = \dfrac{y_t}{x_t}$ (USDC per BTC)
- external market price S_t (USDC per BTC)
- fee rate $\gamma \in (0, 1)$

The pool satisfies the constant-product rule, that is at every moment in time the product of quantities must stay constant.

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

Next we describe the trade mechanics. Suppose someone wants to trade Δx with the pool. That results in a Δy obtained from the pool. This is done so that the updated reserves continue to satisfy the constant product rule. Specifically, using the γ rate that is assesed we must have:

$$(x_t + (1-\gamma)\Delta x)(y_t - \Delta y) = k$$

This will satisfy the constraint $x_{t+1}y_{t+1} = x_t y_t = k$. Note that the price ratio of the assets is changing after this trade: $P_{t+1} = \dfrac{y_t - \Delta y}{x_t + (1-\gamma)\Delta x}$.

Outside this mechanics as long as the quantity of assets in the pool stay constant the price does not change. We are considering a situation where the outside price S_t moves. If S_{t+1} leaves the no-arbitrage band, $\left[P_t(1-\gamma), \frac{P_t}{1-\gamma}\right]$ then an arbitrage opportunity arises. We assume that arbitragers act optimally and instantly. As a result the following happens.

Case 1: $S_{t+1} > P_t \dfrac{1}{1-\gamma}$ **(BTC cheaper in the pool)**

Arbitragers will swap USDC $\to$ BTC until the resulting band after the trades moves so that it contains the outside price S_{t+1}. Updates:

$$x_{t+1} = x_t - \Delta x, \qquad y_{t+1} = y_t + (1-\gamma)\Delta y, \qquad \Delta x, \Delta y > 0,$$

with boundary condition

$$P_{t+1}\frac{1}{1-\gamma} = \frac{y_{t+1}}{x_{t+1}}\frac{1}{1-\gamma} = S_{t+1},$$

The corresponding fee revenue (USDC) is coming from the input asset y. It is thus equal to $\gamma\,\Delta y$.

Case 2: $S_{t+1} < P_t(1-\gamma)$ **(BTC cheaper outside)**

Arbitragers will swap BTC $\to$ USDC and the updated reserves are:

$$x_{t+1} = x_t + (1-\gamma)\Delta x, \qquad y_{t+1} = y_t - \Delta y, \qquad \Delta x, \Delta y > 0,$$

with boundary condition

$$P_{t+1}(1-\gamma) = \frac{y_{t+1}}{x_{t+1}}(1-\gamma) = S_{t+1},$$

and the fee revenue is $\gamma\Delta x$. After converting BTC fee to USDC using S_{t+1} is calculated as $\gamma\,\Delta x\,S_{t+1}$.

Questions:

(a) (10 pts) Derive the swap amounts

Under both Case 1 and Case 2, use the corresponding boundary condition, and the reserve updates above, to derive the swap size:

$$\Delta x(S_{t+1}; x_t, y_t, \gamma, k), \quad \Delta y(S_{t+1}; x_t, y_t, \gamma, k)$$

such that the one-step fee revenue is:

$$R(S_{t+1}) = \mathbf{1}_{\left\{S_{t+1} > \frac{P_t}{1-\gamma}\right\}} \gamma \, \Delta y + \mathbf{1}_{\{S_{t+1} < P_t(1-\gamma)\}} \gamma \, \Delta x \, S_{t+1}.$$

where we used the notation $1_{\{.\}}$ for the indicator function.

(b) (10 pts) Expected fee revenue

Assume the initial BTC/USDC pool reserves are $x_t = y_t = 1000$, so $P_t = \frac{y_t}{x_t} = 1$. Let $S_t = 1$ and $\Delta t = \frac{1}{365}$. Assume the external price follows one-step GBM:

$$S_{t+1} = S_t \exp\Big(-\tfrac{1}{2}\sigma^2 \Delta t + \sigma\sqrt{\Delta t}\, Z\Big), \qquad Z \sim N(0,1),$$

so S_{t+1} is lognormal with density $f_{S_{t+1}}(s)$.

Using the setup above and the solution from (a), the expected one-step fee revenue is:

$$\mathbb{E}[R(S_{t+1})] = \int_{P_t/(1-\gamma)}^{\infty} \gamma \Delta y(S_{t+1}; x_t, y_t, \gamma, k)\, f_{S_{t+1}}(s)\, ds + \int_0^{P_t(1-\gamma)} \gamma \Delta x(S_{t+1}; x_t, y_t, \gamma, k) S_{t+1}\, f_{S_{t+1}}(s)\, ds.$$

Task: Numerically approximate $\mathbb{E}[R(S_{t+1})]$ using trapezoidal rule learned in class.

(c) (10 pts) Optimal Fee Rate under different volatilities

Use $\sigma \in \{0.2, 0.6, 1.0\}$ and $\gamma \in \{0.001, 0.003, 0.01\}$. For each σ, compute $\mathbb{E}[R]$ for the three fee rates and select:

$$\gamma^*(\sigma) = \arg\max_{\gamma} \mathbb{E}[R(S_{t+1})].$$

Construct a table reporting the $\mathbb{E}[R]$ values and the best $\gamma^*(\sigma)$ among the 3 options.

Then for each $\sigma \in [0.1, 1.0]$ on a grid (e.g. 0.01 step), compute the optimal $\gamma^*(\sigma)$. Finally, produce a scatter plot or line plot for σ vs. $\gamma^*(\sigma)$, and comment briefly on the pattern you observe.

Part 4. (**Bonus** 10 points) Consider the following functions:

$$f_1(x, y) = xy$$
$$f_2(x, y) = e^{x+y}$$

1. Analytically solve the following integral for both f_1 and f_2

$$\int_0^1 \int_0^3 f_i(x, y) \mathrm{d}y \mathrm{d}x$$

2. Calculate the numerical integral of the f_1 and f_2 by applying the trapezoidal rule for double integral as discussed in [3], Page 118-119. Please choose four different pairs of values for $(\Delta x, \Delta y)$. Use these values to approximate the double integral for both f_1 and f_2. Calculate the error of the approximation for each choice of values. Comment on the results.

   Hint 1 First, discretize the x domain into n+1 points Δx apart, where $x_0 = 0$ and $x_{n+1} = 1$, and the y domain into $m + 1$ points Δy apart, where $y_0 = 0$ and $y_{m+1} = 3$. The composite trapezoidal rule approximates the integral as

$$\int_0^1 \int_0^3 f(x, y) \mathrm{d}y \mathrm{d}x \approx \sum_{i=0}^{n} \sum_{j=0}^{m} \frac{\Delta x \Delta y}{16} [f(x_i, y_j) + f(x_i, y_{j+1}) + f(x_{i+1}, y_j)$$
$$+ f(x_{i+1}, y_{j+1}) + 2 \left(f(\frac{x_i + x_{i+1}}{2}, y_j) + f(\frac{x_i + x_{i+1}}{2}, y_{j+1}) + f(x_i, \frac{y_j + y_{j+1}}{2}) \right.$$
$$\left. + f(x_{j+1}, \frac{y_j + y_{j+1}}{2}) \right) + 4 f(\frac{x_i + x_{i+1}}{2}, \frac{y_j + y_{j+1}}{2})]$$

   Hint 2. Please note the numbers chosen for $(\Delta x, \Delta y)$ should be specific to each student in the class.

References

[1] Clewlow, Les and Strickland, Chris. *Implementing Derivative Models (Wiley Series in Financial Engineering)*, John Wiley & Sons 1996.

[2] Mariani, Maria C. and Florescu, Ionut *Quantitative Finance*, 2020, John Wiley & Sons.

[3] Rouah, F. D. *The Heston Model and Its Extensions in Matlab and C*, 2013, John Wiley & Sons.

[4] Mikhailov, Sergei and Nögel, Ulrich. "Heston's stochastic volatility model: Implementation, calibration and some extensions" *Wilmott Journal*, 2004.

[5] Angeris, Guillermo and Kao, Hsien-Tang and Chiang, Rei and Noyes, Charlie and Chitra, Tarun. "An analysis of Uniswap markets" *arXiv preprint arXiv:1911.03380*, 2019.

AMMs are decentralized exchanges that quote prices using pool reserves rather than an order book. Compared to Traditional Finance order books, a key advantage is **continuous liquidity from the pool without needing a matching counterparty**. Liquidity Providers (LPs) earn revenue mainly from **swap fees**, so selecting a good fee rate γ under different volatility levels matters [5].

For simplicity we consider the following Constant Product Market Maker (CPMM) for a **BTC/USDC** pool. These are the pool elements:

- x_t: BTC reserves at time t
- y_t: USDC reserves at time t
- pool mid price is calculated as $P_t = \frac{y_t}{x_t}$ (USDC per BTC)
- external market price S_t (USDC per BTC)
- fee rate $\gamma \in (0, 1)$

The pool satisfies the constant-product rule, that is at every moment in time the product of quantities must stay constant.

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

Next we describe the trade mechanics. Suppose someone wants to trade Δx with the pool. That results in a Δy obtained from the pool. This is done so that the updated reserves continue to satisfy the constant product rule. Specifically, using the γ rate that is assesed we must have:

$$(x_t + (1-\gamma)\Delta x)(y_t - \Delta y) = k$$

This will satisfy the constraint $x_{t+1}y_{t+1} = x_t y_t = k$. Note that the price ratio of the assets is changing after this trade: $P_{t+1} = \frac{y_t - \Delta y}{x_t + (1-\gamma)\Delta x}$.

Outside this mechanics as long as the quantity of assets in the pool stay constant the price does not change. We are considering a situation where the outside price S_t moves. If S_{t+1} leaves the no-arbitrage band, $\left[P_t(1-\gamma), \frac{P_t}{1-\gamma}\right]$ then an arbitrage opportunity arises. We assume that arbitragers act optimally and instantly. As a result the following happens.

Case 1: $S_{t+1} > P_t \frac{1}{1-\gamma}$ **(BTC cheaper in the pool)**

Arbitragers will swap USDC $\to$ BTC until the resulting band after the trades moves so that it contains the outside price S_{t+1}. Updates:

$$x_{t+1} = x_t - \Delta x, \qquad y_{t+1} = y_t + (1-\gamma)\Delta y, \qquad \Delta x, \Delta y > 0,$$

with boundary condition

$$P_{t+1}\frac{1}{1-\gamma} = \frac{y_{t+1}}{x_{t+1}}\frac{1}{1-\gamma} = S_{t+1},$$

The corresponding fee revenue (USDC) is coming from the input asset y. It is thus equal to $\gamma\,\Delta y$.

Case 2: $S_{t+1} < P_t(1-\gamma)$ **(BTC cheaper outside)**

Arbitragers will swap BTC $\to$ USDC and the updated reserves are:

$$x_{t+1} = x_t + (1-\gamma)\Delta x, \qquad y_{t+1} = y_t - \Delta y, \qquad \Delta x, \Delta y > 0,$$

with boundary condition

$$P_{t+1}(1-\gamma) = \frac{y_{t+1}}{x_{t+1}}(1-\gamma) = S_{t+1},$$

and the fee revenue is $\gamma\Delta x$. After converting BTC fee to USDC using S_{t+1} is calculated as $\gamma\,\Delta x\,S_{t+1}$.

(b) (10 pts) Expected fee revenue

Assume the initial BTC/USDC pool reserves are $x_t = y_t = 1000$, so $P_t = \frac{y_t}{x_t} = 1$. Let $S_t = 1$ and $\Delta t = \frac{1}{365}$. Assume the external price follows one-step GBM:

$$S_{t+1} = S_t \exp\left(-\tfrac{1}{2}\sigma^2\Delta t + \sigma\sqrt{\Delta t}\,Z\right), \qquad Z \sim N(0,1),$$

so S_{t+1} is lognormal with density $f_{S_{t+1}}(s)$.

Using the setup above and the solution from (a), the expected one-step fee revenue is:

$$\mathbb{E}[R(S_{t+1})] = \int_{P_t/(1-\gamma)}^{\infty} \gamma\Delta y(S_{t+1}; x_t, y_t, \gamma, k)\, f_{S_{t+1}}(s)\, ds + \int_0^{P_t(1-\gamma)} \gamma\Delta x(S_{t+1}; x_t, y_t, \gamma, k) S_{t+1}\, f_{S_{t+1}}(s)\, ds.$$

Task: Numerically approximate $\mathbb{E}[R(S_{t+1})]$ using trapezoidal rule learned in class.

Part 3. (30 points) Numerical Integration of real-valued functions. AMM Arbitrage Fee Revenue

Questions:

(a) (10 pts) Derive the swap amounts

Under both Case 1 and Case 2, use the corresponding boundary condition, and the reserve updates above, to derive the swap size:

$$\Delta x(S_{t+1}; x_t, y_t, \gamma, k), \quad \Delta y(S_{t+1}; x_t, y_t, \gamma, k)$$

such that the one-step fee revenue is:

$$R(S_{t+1}) = \mathbf{1}_{\{S_{t+1} > \frac{P_t}{1-\gamma}\}}\gamma\,\Delta y + \mathbf{1}_{\{S_{t+1} < P_t(1-\gamma)\}}\gamma\,\Delta x\,S_{t+1}.$$

where we used the notation $1_{\{\cdot\}}$ for the indicator function.

vars: $x_t \triangleq$ BTC reserves; $y_t \triangleq$ USDC reserves; $P_t = \frac{y_t}{x_t} \triangleq$ pool mid price; $S_t \triangleq$ USDC per BTC (external); $\gamma \triangleq$ fee rate $\in (0,1)$

Constraints: $x_{t+1}\cdot y_{t+1} = x_t y_t = k$; $\Delta x, \Delta y > 0$

Trade mechanics: · Trade Δx w/ pool $\Rightarrow$ Δy obtained; $(x_t + (1-\gamma)\Delta x)(y_t - \Delta y) = k$ constraint update; $P_{t+1} = \frac{y_t - \Delta y}{x_t + (1-\gamma)\Delta x}$ Price ratio changes

Case 1: $S_{t+1} > \frac{P_t}{1-\gamma}$; $\frac{P_{t+1}}{1-\gamma} = \frac{y_{t+1}}{x_{t+1}(1-\gamma)} = S_{t+1}$

No-Arb: $S_t \in \left[P_t(1-\gamma), \frac{P_t}{(1-\gamma)}\right]$

Boundary conditions

Case 2: $S_t < P_t(1-\gamma)$; $P_{t+1}(1-\gamma) = \frac{y_{t+1}}{x_{t+1}(1-\gamma)} = S_{t+1}$

3a) Find Δx, Δy s.t. $R(S_{t+1}) = \mathbf{1}_{\{S_{t+1} > \frac{P_t}{1-\gamma}\}}\gamma\cdot\Delta y + \mathbf{1}_{\{S_{t+1} < P_t(1-\gamma)\}}\gamma\cdot S_{t+1}\cdot\Delta x$

Case 1: $x_{t+1} = x_t - \Delta x$; $y_{t+1} = y_t - \Delta y = S_{t+1}(x_{t+1}(1-\gamma))$; $x_{t+1}y_{t+1} = k$

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

$\Rightarrow x_{t+1}^2 S_{t+1}(1-\gamma) = k$

$x_{t+1}^2 = \frac{k}{S_{t+1}(1-\gamma)}$

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

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

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

$\Delta y = y_t - y_{t+1} = y_t - S_{t+1}\left(\sqrt{\frac{k}{S_{t+1}(1-\gamma)}}\right)(1-\gamma)$

Case 2: $x_{t+1} = x_t - \Delta x$; $y_{t+1} = \frac{S_{t+1}}{1-\gamma}x_{t+1} = y_t - \Delta y = P_{t+1}(1-\gamma)$; $x_{t+1}y_{t+1} = k$

$x_{t+1}^2\frac{S_{t+1}}{1-\gamma} = k$

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

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

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

(c) (10 pts) Optimal Fee Rate under different volatilities

Use $\sigma \in \{0.2, 0.6, 1.0\}$ and $\gamma \in \{0.001, 0.003, 0.01\}$. For each σ, compute $\mathbb{E}[R]$ for the three fee rates and select:

$$\gamma^*(\sigma) = \arg\max_{\gamma} \mathbb{E}[R(S_{t+1})].$$

Construct a table reporting the $\mathbb{E}[R]$ values and the best $\gamma^*(\sigma)$ among the 3 options.

Then for each $\sigma \in [0.1, 1.0]$ on a grid (e.g. 0.01 step), compute the optimal $\gamma^*(\sigma)$. Finally, produce a scatter plot or line plot for σ vs. $\gamma^*(\sigma)$, and comment briefly on the pattern you observe.

3b) $x_t = y_t = 1000 \Rightarrow P_t = 1$; $S_t = 1$, $\Delta t = 1/365$

$h_1 \triangleq$ step size; $y_i \triangleq \frac{P_t}{1-\gamma} + i\cdot h_1$; $z_j \triangleq P_t(1-\gamma) + j h_2$

$g_1(s) = \gamma\cdot\Delta y\cdot f_{S_{t+1}}(s)$, $g_2 = \gamma\cdot\Delta x\cdot S\cdot f_{S_{t+1}}(s)$

$$\mathbb{E}[R(S_{t+1})] = \sum_{i=1}^{h_1} h_1 \frac{g_1(y_{i-1}) + g_1(x_i)}{2} + \sum_{j=1}^{h_2} h_2 \frac{g_2(z_{j-1}) + g_2(z_j)}{2}$$

Part 4. (**Bonus** 10 points) Consider the following functions:

a) $f_1(x, y) = xy$

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

1. Analytically solve the following integral for both f_1 and f_2

$$\int_0^1 \int_0^3 f_i(x, y)\mathrm{d}y\mathrm{d}x$$

1a) $\int_0^1 \int_0^3 xy \, dy dx$

$= \int_0^1 x\left(\left[\frac{y^2}{2}\right]_0^3\right) dx$

$= \int_0^1 x\left(\frac{9}{2} - 0\right) dx$

$= \frac{9}{2}\left(\left[\frac{x^2}{2}\right]_0^1\right) = \frac{9}{2}\left(\frac{1}{2}\right) = \boxed{9/4}$

1b) $\int_0^1 \int_0^3 e^{x+y} \, dy dx$

$= \int_0^1 \int_0^3 e^y e^x \, dy dx$

$= \int_0^1 e^x \left[e^y\right]_0^3 dx$

$= \int_0^1 (e^3 - e^0) e^x \, dx$

$= (e^3 - 1)\left[e^x\right]_0^1 = (e^3 - 1)(e^1 - 1) = \boxed{e^4 - e^3 - e + 1}$

2. Calculate the numerical integral of the f_1 and f_2 by applying the trapezoidal rule for double integral as discussed in [3], Page 118-119. Please choose four different pairs of values for $(\Delta x, \Delta y)$. Use these values to approximate the double integral for both f_1 and f_2. Calculate the error of the approximation for each choice of values. Comment on the results.

Hint 1 First, discretize the x domain into n+1 points Δx apart, where $x_0 = 0$ and $x_{n+1} = 1$, and the y domain into $m + 1$ points Δy apart, where $y_0 = 0$ and $y_{m+1} = 3$. The composite trapezoidal rule approximates the integral as

$$\int_0^1 \int_0^3 f(x, y)\mathrm{d}y\mathrm{d}x \approx \sum_{i=0}^{n} \sum_{j=0}^{m} \frac{\Delta x \Delta y}{16} [f(x_i, y_j) + f(x_i, y_{j+1}) + f(x_{i+1}, y_j) + f(x_{i+1}, y_{j+1}) + 2\left(f(\frac{x_i + x_{i+1}}{2}, y_j) + f(\frac{x_i + x_{i+1}}{2}, y_{j+1}) + f(x_i, \frac{y_j + y_{j+1}}{2}) + f(x_{j+1}, \frac{y_j + y_{j+1}}{2})\right) + 4f(\frac{x_i + x_{i+1}}{2}, \frac{y_j + y_{j+1}}{2})]$$

Hint 2. Please note the numbers chosen for $(\Delta x, \Delta y)$ should be specific to each student in the class.

Done in R → $x \in [0, 1]$, $y \in [0, 3]$ → $\Delta x = \{.1, .04, .01, .001\}$, $\Delta y = \{.2, .04, .005, .0001\}$ ← Chosen arbitrarily, I hope nobody has the same #'s

1) For each dx, dy split x, y

Need: $f(x, y) = x_i \begin{bmatrix} y_j \\ \\ \end{bmatrix}$

$\left\{ x_i = \frac{}{n+1}\begin{bmatrix} x_1 \\ \vdots \\ \end{bmatrix}, \; y_i = \frac{}{m+1}\begin{bmatrix} y_1 \\ \vdots \\ y_{m+1} \end{bmatrix} \right.$

FE 621 HW1

2026-02-13

Jackson Kelleher FE621 HW1

I pledge my Honor that I have abided by the Stevens Honor System

#Libraries

```
library(quantmod)

library(xts)
library(ggplot2)
library(openxlsx)
library(date)
```

#Part 1. (20 points)

1. Write a function (program) to connect to sources and download data from one of the following sources Bonus (5 points) 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.

```
#Works with yahoo and google, assigns to global environment and downloads as
two excel sheets
download_data = function(tickerf, fromf, tof, srcf, exp = NULL){

  for(i in 1:length(tickerf)){
    #equity data
    getSymbols(Symbols = tickerf[i], from = as.character(fromf), to   =
as.character(tof), src  = as.character(srcf), env  = .GlobalEnv)
    symbol = tickerf[i]
    if(symbol == "^VIX"){symbol = "VIX"}
    write.csv(as.data.frame(get(symbol)), paste0(symbol, "_equity.csv"))

    #option data
    if(symbol == "VIX"){symbol = "^VIX"}
    chain = getOptionChain(Symbols = symbol, src = as.character(srcf))
    assign(paste0(tickerf[i], "_chain"), getOptionChain(Symbols = symbol, src
= as.character(srcf), Exp = exp), env = .GlobalEnv)
    opt_df = rbind(cbind(Type = "Call", chain$calls), cbind(Type = "Put",
chain$puts))
    if(symbol == "^VIX"){symbol = "VIX"}
    write.csv(opt_df, paste0(symbol, "_options.csv"), row.names = FALSE)

  }

  return()
}
```

2. With the function created in problem 1, download data on options and equity for the following symbols: • TSLA • SPY • ^VIX

```
tickers = c("TSLA", "SPY", "^VIX")
download_data(tickers, "2026-02-14", "2026-02-15", "yahoo")

## NULL

names = names(SPY_chain)
exp_dates = as.Date(names, format = "%b.%d.%Y")
is_third_friday <- function(d) {
  weekdays(d) == "Friday" &
  as.numeric(format(d, "%d")) >= 15 &
  as.numeric(format(d, "%d")) <= 21
}

keep = is_third_friday(exp_dates)
index = which(keep == TRUE)
for(i in (index[3]+1):(length(keep))){
  keep[i] = FALSE
}

#VIX_chain = get("^VIX_chain")
SPY_chain2 = SPY_chain[keep]
TSLA_chain2 = TSLA_chain[keep]
#VIX_chain2 = VIX_chain[keep]
```

3. Write a paragraph describing the symbols you are downloading data for. Explain what is SPY and its purpose. (Hint: look up the definition of an ETF). Explain what is ^VIX and its purpose. Understand the options' symbols. Understand when each option expires. Write this information and turn it in.

   TSLA refers to the common stock of Tesla, a publicly traded equity.

   SPY is the ticker symbol for the SPDR S&P 500 ETF Trust. An ETF is a security that trades on an exchange like a stock but represents a basket of underlying assets.

   ^VIX is the CBOE Volatility Index, commonly referred to as the "VIX." It measures the market's expectation of 30-day forward-looking volatility implied by S&P 500 index options. It is often interpreted as a measure of market uncertainty

   Each option contract is identified by:

   The ticker

   The expiration date

   The strike price

   The option type (Call or Put)

4. The following items will also need to be recorded:

a. The underlying equity, ETF, or index price at the exact moment when the rest of the data is downloaded.
b. The short-term interest rate
c. Time to Maturity.

```
#I apologize as I downloaded these after the trading day on the 15th, I did
not see that part of the assignment until too late

#a - TSLA S0
#SPY Price: 681.75
#TSLA Price: 417.44
#^VIX Price: 20.60

#b - Fed Funds Effective
r = .0364

#c - TTM
today = as.Date("2026-02-15")
taus = as.numeric(exp_dates - today) / 365
taus = taus[index]
```

#Part 2. (50 points)

5. Using your choice of computer programming language implement the Black-Scholes formulas

```
#BS for euro call - c(s0, sigma, tau, k, r)
tau = 1
sigma = .1
k = 1

bs_call = function(s0, sigma, tau, k, r){
  d1 = 1/(sigma*sqrt(tau))*(log(s0/k)+tau*(r+((sigma^2)/2)))
  d2 = 1/(sigma*sqrt(tau))*(log(s0/k)+tau*(r-((sigma^2)/2)))

  s0*pnorm(d1)-(k*exp(-r*tau)*pnorm(d2))
}

bs_put = function(s0, sigma, tau, k, r){
  if(tau <= 0 || sigma <= 0) return(max(k - s0, 0))
  d1 = bs_d1(s0, sigma, tau, k, r)
  d2 = d1 - sigma*sqrt(tau)
  k*exp(-r*tau)*pnorm(-d2) - s0*pnorm(-d1)
}
```

6. Implement the Bisection method to find the root of arbitrary functions.

```
#Bisection Methoid from Haug
#Citation: https://quant.stackexchange.com/questions/3027/modified-bisection-
```

```
formula-for-deriving-implied-volatility-for-a-dividend-paying?rq=1

#tol = 10^-6
S0_SPY  = 681.75
S0_TSLA = 417.44

taus_spy  = as.numeric(as.Date(names(SPY_chain2),  format = "%b.%d.%Y") -
today)/365
taus_tsla = as.numeric(as.Date(names(TSLA_chain2), format = "%b.%d.%Y") -
today)/365

iv_bisect = function(price, s0, k, tau, r, type = "call"){

  tol = 1e-6
  vLow  = 0.01
  vHigh = 1

  if(is.na(price) || is.na(k) || is.na(tau) || tau <= 0) return(NA)

  #initial prices
  if(type == "call"){
    cLow  = bs_call(s0, vLow,  tau, k, r)
    cHigh = bs_call(s0, vHigh, tau, k, r)
  } else {
    cLow  = bs_put(s0, vLow,  tau, k, r)
    cHigh = bs_put(s0, vHigh, tau, k, r)
  }

  #expand upper bound if needed
  while((price - cLow)*(price - cHigh) > 0){
    vHigh = vHigh * 2

    if(type == "call"){
      cHigh = bs_call(s0, vHigh, tau, k, r)
    } else {
      cHigh = bs_put(s0, vHigh, tau, k, r)
    }

    if(vHigh > 100) return(NA)
  }

  vi = vLow + (price - cLow)*(vHigh - vLow)/(cHigh - cLow)

  if(type == "call"){
    tempval = bs_call(s0, vi, tau, k, r)
  } else {
    tempval = bs_put(s0, vi, tau, k, r)
  }
```

```r
  while(abs(price - tempval) > tol){

    if(tempval < price){
      vLow = vi
    } else {
      vHigh = vi
    }

    if(type == "call"){
      cLow  = bs_call(s0, vLow,  tau, k, r)
      cHigh = bs_call(s0, vHigh, tau, k, r)
    } else {
      cLow  = bs_put(s0, vLow,  tau, k, r)
      cHigh = bs_put(s0, vHigh, tau, k, r)
    }

    vi = vLow + (price - cLow)*(vHigh - vLow)/(cHigh - cLow)

    if(type == "call"){
      tempval = bs_call(s0, vi, tau, k, r)
    } else {
      tempval = bs_put(s0, vi, tau, k, r)
    }
  }

  return(vi)
}

#mid price helper function
mid_price = function(df){
  (df$Bid + df$Ask)/2
}

#bs d1 helper function (needed by your bs_put if you call it)
bs_d1 = function(s0, sigma, tau, k, r){
  (log(s0/k) + (r + 0.5*sigma^2)*tau) / (sigma*sqrt(tau))
}


#SPY IVs
iv_results = data.frame()

for(m in 1:length(SPY_chain2)){

  tau_m = taus_spy[m]
  exp_m = names(SPY_chain2)[m]
```

```r
  calls = SPY_chain2[[m]]$calls
  puts  = SPY_chain2[[m]]$puts

  calls = calls[calls$Vol > 0 & (calls$Bid) & (calls$Ask) & (calls$Strike), ]
  puts  = puts [puts$Vol  > 0 & (puts$Bid)  & (puts$Ask)  & (puts$Strike), ]

  calls$Mid = mid_price(calls)
  puts$Mid  = mid_price(puts)

  calls$IV = NA
  puts$IV  = NA

  for(i in 1:nrow(calls)){
    calls$IV[i] = iv_bisect(calls$Mid[i], S0_SPY, calls$Strike[i], tau_m, r,
type = "call")
  }

  for(i in 1:nrow(puts)){
    puts$IV[i] = iv_bisect(puts$Mid[i], S0_SPY, puts$Strike[i], tau_m, r,
type = "put")
  }

  calls_out = data.frame(Symbol="SPY", Expiration=exp_m, Tau=tau_m,
Type="Call",
                         Strike=calls$Strike, Bid=calls$Bid, Ask=calls$Ask,
Mid=calls$Mid,
                         Vol=calls$Vol, IV=calls$IV)

  puts_out  = data.frame(Symbol="SPY", Expiration=exp_m, Tau=tau_m,
Type="Put",
                         Strike=puts$Strike, Bid=puts$Bid, Ask=puts$Ask,
Mid=puts$Mid,
                         Vol=puts$Vol, IV=puts$IV)

  iv_results = rbind(iv_results, calls_out, puts_out)
}

print(head(iv_results, 20))
```

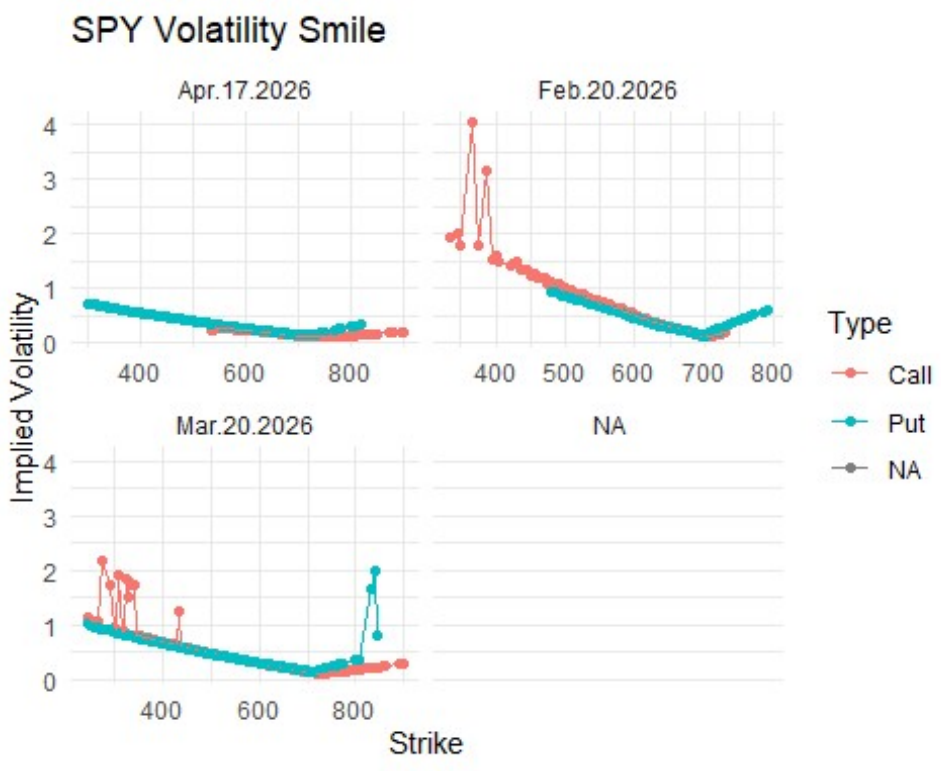
```
##    Symbol  Expiration        Tau Type Strike    Bid    Ask     Mid Vol
IV
## 1     SPY Feb.20.2026 0.01369863 Call    335 345.62 348.26 346.940   1
1.922776
## 2     SPY Feb.20.2026 0.01369863 Call    345 335.62 338.35 336.985   2
2.013056
## 3     SPY Feb.20.2026 0.01369863 Call    350 330.62 333.27 331.945   7
1.793952
## 4     SPY Feb.20.2026 0.01369863 Call    365 325.28 328.67 326.975   2
4.040389
```

```
## 5     SPY Feb.20.2026 0.01369863 Call     NA     NA     NA      NA  NA
NA
## 6     SPY Feb.20.2026 0.01369863 Call    375 305.64 308.37 307.005   1
1.793886
## 7     SPY Feb.20.2026 0.01369863 Call     NA     NA     NA      NA  NA
NA
## 8     SPY Feb.20.2026 0.01369863 Call    385 300.16 303.42 301.790   1
3.148574
## 9     SPY Feb.20.2026 0.01369863 Call     NA     NA     NA      NA  NA
NA
## 10    SPY Feb.20.2026 0.01369863 Call    395 285.65 288.31 286.980   1
1.544684
## 11    SPY Feb.20.2026 0.01369863 Call    400 280.66 283.38 282.020   5
1.618220
## 12    SPY Feb.20.2026 0.01369863 Call    405 275.66 278.31 276.985   7
1.478340
## 13    SPY Feb.20.2026 0.01369863 Call     NA     NA     NA      NA  NA
NA
## 14    SPY Feb.20.2026 0.01369863 Call    420 260.68 263.32 262.000   1
1.406561
## 15    SPY Feb.20.2026 0.01369863 Call    425 255.68 258.40 257.040   1
1.461103
## 16    SPY Feb.20.2026 0.01369863 Call    430 251.89 252.31 252.100   1
1.510833
## 17    SPY Feb.20.2026 0.01369863 Call    435 245.70 248.34 247.020   2
1.344004
## 18    SPY Feb.20.2026 0.01369863 Call    440 240.69 243.41 242.050   1
1.366076
## 19    SPY Feb.20.2026 0.01369863 Call    445 235.70 238.42 237.060  11
1.345528
## 20    SPY Feb.20.2026 0.01369863 Call    450 230.69 233.35 232.020 196
1.232342
```

```
#plot volatility smile
smile = iv_results[(iv_results$IV) & (iv_results$Strike) & iv_results$Vol >
0, ]
ggplot(smile, aes(x = Strike, y = IV, color = Type)) +
  geom_point() +
  geom_line() +
  facet_wrap(~ Expiration, scales = "free_x") +
  labs(title="SPY Volatility Smile", x="Strike", y="Implied Volatility") +
  theme_minimal()
```

```
## Warning: Removed 61 rows containing missing values or values outside the
scale range
## (`geom_point()`).
```

```
## Warning: Removed 61 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```




```r
#TSLA IV's
iv_results_tsla = data.frame()

for(m in 1:length(TSLA_chain2)){

  tau_m = taus_tsla[m]
  exp_m = names(TSLA_chain2)[m]

  calls = TSLA_chain2[[m]]$calls
  puts  = TSLA_chain2[[m]]$puts

  calls = calls[calls$Vol > 0 & (calls$Bid) & (calls$Ask) & (calls$Strike), ]
  puts  = puts [puts$Vol  > 0 & (puts$Bid)  & (puts$Ask)  & (puts$Strike), ]

  calls$Mid = mid_price(calls)
  puts$Mid  = mid_price(puts)

  calls$IV = NA
  puts$IV  = NA

  for(i in 1:nrow(calls)){
    calls$IV[i] = iv_bisect(calls$Mid[i], S0_TSLA, calls$Strike[i], tau_m, r,
type = "call")
  }
```

```
  for(i in 1:nrow(puts)){
    puts$IV[i] = iv_bisect(puts$Mid[i], S0_TSLA, puts$Strike[i], tau_m, r,
type = "put")
  }

  calls_out = data.frame(Symbol="TSLA", Expiration=exp_m, Tau=tau_m,
Type="Call",
                         Strike=calls$Strike, Bid=calls$Bid, Ask=calls$Ask,
Mid=calls$Mid,
                         Vol=calls$Vol, IV=calls$IV)

  puts_out  = data.frame(Symbol="TSLA", Expiration=exp_m, Tau=tau_m,
Type="Put",
                         Strike=puts$Strike, Bid=puts$Bid, Ask=puts$Ask,
Mid=puts$Mid,
                         Vol=puts$Vol, IV=puts$IV)

  iv_results_tsla = rbind(iv_results_tsla, calls_out, puts_out)
}

print(head(iv_results_tsla, 20))
```

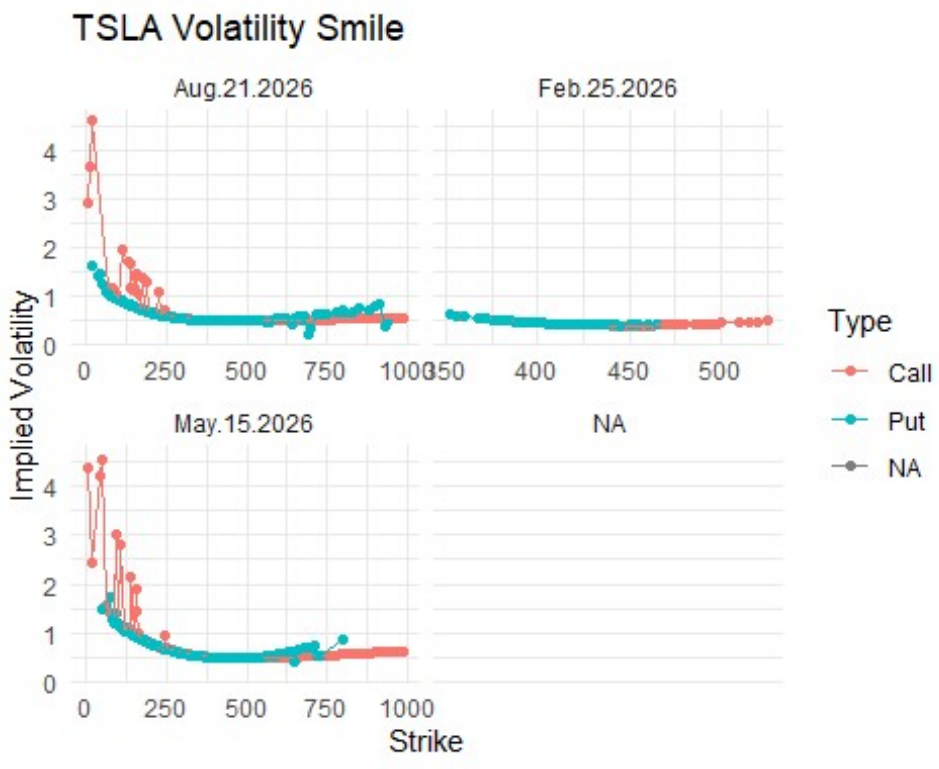
```
##    Symbol  Expiration        Tau Type Strike   Bid   Ask    Mid  Vol
IV
## 1    TSLA Feb.25.2026 0.02739726 Call  357.5 60.10 62.40 61.250    2
0.6000171
## 2    TSLA Feb.25.2026 0.02739726 Call  392.5 29.10 29.35 29.225    1
0.4773823
## 3    TSLA Feb.25.2026 0.02739726 Call  400.0 23.15 23.40 23.275    2
0.4584392
## 4    TSLA Feb.25.2026 0.02739726 Call  402.5 21.30 21.55 21.425    5
0.4531567
## 5    TSLA Feb.25.2026 0.02739726 Call  405.0 19.55 19.75 19.650   15
0.4483450
## 6    TSLA Feb.25.2026 0.02739726 Call  407.5 17.85 18.05 17.950   62
0.4438023
## 7    TSLA Feb.25.2026 0.02739726 Call  410.0 16.20 16.40 16.300   91
0.4384183
## 8    TSLA Feb.25.2026 0.02739726 Call  412.5 14.65 14.85 14.750  339
0.4339770
## 9    TSLA Feb.25.2026 0.02739726 Call  415.0 13.20 13.40 13.300 2063
0.4303094
## 10   TSLA Feb.25.2026 0.02739726 Call  417.5 11.85 12.05 11.950  358
0.4273011
## 11   TSLA Feb.25.2026 0.02739726 Call  420.0 10.60 10.75 10.675  328
0.4239799
## 12   TSLA Feb.25.2026 0.02739726 Call  422.5  9.45  9.60  9.525  122
0.4221371
## 13   TSLA Feb.25.2026 0.02739726 Call  425.0  8.35  8.50  8.425  157
```

```
0.4190399
## 14   TSLA Feb.25.2026 0.02739726 Call  427.5  7.35  7.50  7.425  114
0.4165406
## 15   TSLA Feb.25.2026 0.02739726 Call  430.0  6.45  6.60  6.525 1021
0.4147300
## 16   TSLA Feb.25.2026 0.02739726 Call  432.5  5.65  5.75  5.700   88
0.4127490
## 17   TSLA Feb.25.2026 0.02739726 Call  435.0  4.90  5.00  4.950   97
0.4106901
## 18   TSLA Feb.25.2026 0.02739726 Call  437.5  4.25  4.35  4.300   90
0.4097965
## 19   TSLA Feb.25.2026 0.02739726 Call  440.0  3.65  3.75  3.700  251
0.4080812
## 20   TSLA Feb.25.2026 0.02739726 Call  442.5  3.15  3.25  3.200   98
0.4081144
```

```
#plot volatility smile
smile_tsla = iv_results_tsla[(iv_results_tsla$IV) & (iv_results_tsla$Strike)
& iv_results_tsla$Vol > 0, ]
ggplot(smile_tsla, aes(x = Strike, y = IV, color = Type)) +
  geom_point() +
  geom_line() +
  facet_wrap(~ Expiration, scales = "free_x") +
  labs(title="TSLA Volatility Smile", x="Strike", y="Implied Volatility") +
  theme_minimal()
```

```
## Warning: Removed 44 rows containing missing values or values outside the
scale range
## (`geom_point()`).
```

```
## Warning: Removed 44 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```




```r
#Average IV in 0.95 <= K/S0 <= 1.05

iv_all = rbind(iv_results, iv_results_tsla)

iv_all$S0 = ifelse(iv_all$Symbol == "SPY", S0_SPY, S0_TSLA)

iv_all$Moneyness = iv_all$Strike / iv_all$S0

band = iv_all[
  (iv_all$IV) &
  iv_all$Moneyness >= 0.95 &
  iv_all$Moneyness <= 1.05,
]

avg_iv = aggregate(IV ~ Symbol + Expiration + Type, data = band, mean)
print(avg_iv)
```

```
##    Symbol  Expiration Type        IV
## 1     SPY Apr.17.2026 Call 0.1623674
## 2    TSLA Aug.21.2026 Call 0.5033799
## 3     SPY Feb.20.2026 Call 0.2032233
## 4    TSLA Feb.25.2026 Call 0.4289632
## 5     SPY Mar.20.2026 Call 0.1729584
## 6    TSLA May.15.2026 Call 0.4867189
## 7     SPY Apr.17.2026  Put 0.1816795
## 8    TSLA Aug.21.2026  Put 0.5007313
```

```
## 9     SPY Feb.20.2026  Put 0.2140074
## 10   TSLA Feb.25.2026  Put 0.4301861
## 11    SPY Mar.20.2026  Put 0.1858570
## 12   TSLA May.15.2026  Put 0.4827616
```

7. Implement the Newton method/Secant method or Muller method

```
# ----------------------------
# 2.7 Newton / Secant IV solver
# ----------------------------

# Vega formula: dPrice/dSigma = S0 * phi(d1) * sqrt(tau)
bs_vega = function(s0, sigma, tau, k, r){
  if(tau <= 0 || sigma <= 0) return(0)
  d1 = bs_d1(s0, sigma, tau, k, r)
  s0 * dnorm(d1) * sqrt(tau)
}

# Newton method for implied vol
iv_newton = function(price, s0, k, tau, r, type = "call",
                     sigma0 = 0.20, tol = 1e-6, max_iter = 100){

  if(is.na(price) || is.na(k) || is.na(tau) || tau <= 0) return(NA)

  sigma = max(sigma0, 1e-6)

  for(it in 1:max_iter){

    model_price = if(type == "call"){
      bs_call(s0, sigma, tau, k, r)
    } else {
      bs_put(s0, sigma, tau, k, r)
    }

    diff = model_price - price
    if(abs(diff) < tol) return(sigma)

    v = bs_vega(s0, sigma, tau, k, r)

    # if vega is too small, Newton becomes unstable
    if(!is.finite(v) || v < 1e-10) return(NA)

    sigma_new = sigma - diff / v

    # guardrails (avoid negative / exploding vols)
    if(!is.finite(sigma_new)) return(NA)
    sigma = min(max(sigma_new, 1e-6), 5)
  }
```

```
  return(NA)
}

# Secant method for implied vol (no derivative)
iv_secant = function(price, s0, k, tau, r, type = "call",
                     sigma1 = 0.10, sigma2 = 0.30, tol = 1e-6, max_iter =
100){

  if(is.na(price) || is.na(k) || is.na(tau) || tau <= 0) return(NA)

  f = function(sig){
    if(sig <= 0) return(Inf)
    if(type == "call"){
      bs_call(s0, sig, tau, k, r) - price
    } else {
      bs_put(s0, sig, tau, k, r) - price
    }
  }

  x0 = max(sigma1, 1e-6)
  x1 = max(sigma2, 1e-6)
  f0 = f(x0)
  f1 = f(x1)

  for(it in 1:max_iter){
    if(!is.finite(f0) || !is.finite(f1)) return(NA)
    if(abs(f1) < tol) return(x1)

    denom = (f1 - f0)
    if(abs(denom) < 1e-12) return(NA)

    x2 = x1 - f1 * (x1 - x0) / denom

    x0 = x1; f0 = f1
    x1 = min(max(x2, 1e-6), 5)
    f1 = f(x1)
  }

  return(NA)
}

# --- Apply to same chains as 2.6 and compare timing ---

solve_chain_iv_compare = function(chain_obj, taus_vec, s0, symbol_label){

  out = data.frame()

  # timing containers
```

```
  t_bisect = 0
  t_newton = 0
  t_secant = 0

  for(m in 1:length(chain_obj)){

    tau_m = taus_vec[m]
    exp_m = names(chain_obj)[m]

    calls = chain_obj[[m]]$calls
    puts  = chain_obj[[m]]$puts

    calls = calls[calls$Vol > 0 & (calls$Bid) & (calls$Ask) & (calls$Strike),
]
    puts  = puts [puts$Vol  > 0 & (puts$Bid)  & (puts$Ask)  & (puts$Strike),
]

    calls$Mid = mid_price(calls)
    puts$Mid  = mid_price(puts)

    calls$IV_bisect = NA; calls$IV_newton = NA; calls$IV_secant = NA
    puts$IV_bisect  = NA; puts$IV_newton  = NA; puts$IV_secant  = NA

    # --- Calls ---
    if(nrow(calls) > 0){
      tb = system.time({
        for(i in 1:nrow(calls)){
          calls$IV_bisect[i] = iv_bisect(calls$Mid[i], s0, calls$Strike[i],
tau_m, r, type="call")
        }
      })["elapsed"]

      tn = system.time({
        for(i in 1:nrow(calls)){
          # seed Newton with bisection if available (fast + stable), else
0.20
          seed = calls$IV_bisect[i]
          if(is.na(seed)) seed = 0.20
          calls$IV_newton[i] = iv_newton(calls$Mid[i], s0, calls$Strike[i],
tau_m, r, type="call", sigma0=seed)
        }
      })["elapsed"]

      ts = system.time({
        for(i in 1:nrow(calls)){
          calls$IV_secant[i] = iv_secant(calls$Mid[i], s0, calls$Strike[i],
tau_m, r, type="call")
        }
      })["elapsed"]
```

```
      t_bisect = t_bisect + tb
      t_newton = t_newton + tn
      t_secant = t_secant + ts
    }

    # --- Puts ---
    if(nrow(puts) > 0){
      tb = system.time({
        for(i in 1:nrow(puts)){
          puts$IV_bisect[i] = iv_bisect(puts$Mid[i], s0, puts$Strike[i],
tau_m, r, type="put")
        }
      })["elapsed"]

      tn = system.time({
        for(i in 1:nrow(puts)){
          seed = puts$IV_bisect[i]
          if(is.na(seed)) seed = 0.20
          puts$IV_newton[i] = iv_newton(puts$Mid[i], s0, puts$Strike[i],
tau_m, r, type="put", sigma0=seed)
        }
      })["elapsed"]

      ts = system.time({
        for(i in 1:nrow(puts)){
          puts$IV_secant[i] = iv_secant(puts$Mid[i], s0, puts$Strike[i],
tau_m, r, type="put")
        }
      })["elapsed"]

      t_bisect = t_bisect + tb
      t_newton = t_newton + tn
      t_secant = t_secant + ts
    }

    calls_out = data.frame(Symbol=symbol_label, Expiration=exp_m, Tau=tau_m,
Type="Call",
                           Strike=calls$Strike, Mid=calls$Mid, Vol=calls$Vol,
                           IV_bisect=calls$IV_bisect,
IV_newton=calls$IV_newton, IV_secant=calls$IV_secant)

    puts_out  = data.frame(Symbol=symbol_label, Expiration=exp_m, Tau=tau_m,
Type="Put",
                           Strike=puts$Strike, Mid=puts$Mid, Vol=puts$Vol,
                           IV_bisect=puts$IV_bisect,
IV_newton=puts$IV_newton, IV_secant=puts$IV_secant)

    out = rbind(out, calls_out, puts_out)
```

```r
  }

  times = data.frame(Symbol=symbol_label,
                     Method=c("Bisection","Newton","Secant"),
                     Seconds=c(t_bisect, t_newton, t_secant))

  return(list(iv_table = out, timing = times))
}

# ---- Run for SPY + TSLA using your filtered chains ----
res_spy  = solve_chain_iv_compare(SPY_chain2,  taus_spy,  S0_SPY,  "SPY")
res_tsla = solve_chain_iv_compare(TSLA_chain2, taus_tsla, S0_TSLA, "TSLA")

iv_compare_all = rbind(res_spy$iv_table, res_tsla$iv_table)
timing_all     = rbind(res_spy$timing,   res_tsla$timing)

print(head(iv_compare_all, 20))
```

```
##    Symbol  Expiration        Tau Type Strike     Mid Vol IV_bisect
IV_newton
## 1     SPY Feb.20.2026 0.01369863 Call    335 346.940   1  1.922776
1.922776
## 2     SPY Feb.20.2026 0.01369863 Call    345 336.985   2  2.013056
2.013056
## 3     SPY Feb.20.2026 0.01369863 Call    350 331.945   7  1.793952
1.793952
## 4     SPY Feb.20.2026 0.01369863 Call    365 326.975   2  4.040389
4.040389
## 5     SPY Feb.20.2026 0.01369863 Call     NA      NA  NA        NA
NA
## 6     SPY Feb.20.2026 0.01369863 Call    375 307.005   1  1.793886
1.793886
## 7     SPY Feb.20.2026 0.01369863 Call     NA      NA  NA        NA
NA
## 8     SPY Feb.20.2026 0.01369863 Call    385 301.790   1  3.148574
3.148574
## 9     SPY Feb.20.2026 0.01369863 Call     NA      NA  NA        NA
NA
## 10    SPY Feb.20.2026 0.01369863 Call    395 286.980   1  1.544684
1.544684
## 11    SPY Feb.20.2026 0.01369863 Call    400 282.020   5  1.618220
1.618220
## 12    SPY Feb.20.2026 0.01369863 Call    405 276.985   7  1.478340
1.478340
## 13    SPY Feb.20.2026 0.01369863 Call     NA      NA  NA        NA
NA
## 14    SPY Feb.20.2026 0.01369863 Call    420 262.000   1  1.406561
1.406561
## 15    SPY Feb.20.2026 0.01369863 Call    425 257.040   1  1.461103
```

```
1.461103
## 16    SPY Feb.20.2026 0.01369863 Call    430 252.100   1  1.510833
1.510833
## 17    SPY Feb.20.2026 0.01369863 Call    435 247.020   2  1.344004
1.344004
## 18    SPY Feb.20.2026 0.01369863 Call    440 242.050   1  1.366076
1.366076
## 19    SPY Feb.20.2026 0.01369863 Call    445 237.060  11  1.345528
1.345528
## 20    SPY Feb.20.2026 0.01369863 Call    450 232.020 196  1.232342
1.232342
##    IV_secant
## 1         NA
## 2         NA
## 3         NA
## 4         NA
## 5         NA
## 6         NA
## 7         NA
## 8         NA
## 9         NA
## 10        NA
## 11        NA
## 12        NA
## 13        NA
## 14        NA
## 15        NA
## 16        NA
## 17        NA
## 18        NA
## 19        NA
## 20        NA

print(timing_all)

##   Symbol    Method Seconds
## 1    SPY Bisection    0.39
## 2    SPY    Newton    0.01
## 3    SPY    Secant    0.06
## 4   TSLA Bisection    0.10
## 5   TSLA    Newton    0.02
## 6   TSLA    Secant    0.04

# ---- ATM IV comparison (strike closest to S0) per maturity/type/method ----
iv_compare_all$S0 = ifelse(iv_compare_all$Symbol=="SPY", S0_SPY, S0_TSLA)
iv_compare_all$AbsDiff = abs(iv_compare_all$Strike - iv_compare_all$S0)

atm_rows = do.call(rbind, lapply(split(iv_compare_all,
list(iv_compare_all$Symbol,
```

```
iv_compare_all$Expiration,

iv_compare_all$Type), drop=TRUE),
                                  function(df){
                                    df[which.min(df$AbsDiff), ]
                                  }))

atm_report = atm_rows[, c("Symbol","Expiration","Type","Strike","S0","Mid",
                          "IV_bisect","IV_newton","IV_secant")]
print(atm_report)
```

```
##                         Symbol  Expiration Type Strike     S0    Mid
IV_bisect
## SPY.Apr.17.2026.Call     SPY Apr.17.2026 Call  682.0 681.75 20.215
0.1642409
## TSLA.Aug.21.2026.Call   TSLA Aug.21.2026 Call  420.0 417.44 61.900
0.5030026
## SPY.Feb.20.2026.Call     SPY Feb.20.2026 Call  682.0 681.75  6.490
0.2024799
## TSLA.Feb.25.2026.Call   TSLA Feb.25.2026 Call  417.5 417.44 11.950
0.4273011
## SPY.Mar.20.2026.Call     SPY Mar.20.2026 Call  682.0 681.75 15.180
0.1734499
## TSLA.May.15.2026.Call   TSLA May.15.2026 Call  420.0 417.44 40.325
0.4851608
## SPY.Apr.17.2026.Put      SPY Apr.17.2026  Put  682.0 681.75 18.005
0.1794193
## TSLA.Aug.21.2026.Put    TSLA Aug.21.2026  Put  420.0 417.44 56.375
0.5002119
## SPY.Feb.20.2026.Put      SPY Feb.20.2026  Put  682.0 681.75  6.285
0.1988661
## TSLA.Feb.25.2026.Put    TSLA Feb.25.2026  Put  417.5 417.44 11.575
0.4266165
## SPY.Mar.20.2026.Put      SPY Mar.20.2026  Put  682.0 681.75 13.915
0.1823538
## TSLA.May.15.2026.Put    TSLA May.15.2026  Put  420.0 417.44 38.850
0.4811902
##                         IV_newton IV_secant
## SPY.Apr.17.2026.Call  0.1642409 0.1642409
## TSLA.Aug.21.2026.Call 0.5030026 0.5030026
## SPY.Feb.20.2026.Call  0.2024799 0.2024799
## TSLA.Feb.25.2026.Call 0.4273011 0.4273011
## SPY.Mar.20.2026.Call  0.1734499 0.1734499
## TSLA.May.15.2026.Call 0.4851608 0.4851608
## SPY.Apr.17.2026.Put   0.1794193 0.1794193
## TSLA.Aug.21.2026.Put  0.5002119 0.5002119
## SPY.Feb.20.2026.Put   0.1988661 0.1988661
## TSLA.Feb.25.2026.Put  0.4266165 0.4266165
## SPY.Mar.20.2026.Put   0.1823538 0.1823538
## TSLA.May.15.2026.Put  0.4811902 0.4811902
```

```r
# ---- Average IV between 0.95 and 1.05 moneyness for each method ----
iv_compare_all$Moneyness = iv_compare_all$Strike / iv_compare_all$S0

band2 = iv_compare_all[
  (iv_compare_all$Moneyness) &
  iv_compare_all$Moneyness >= 0.95 &
  iv_compare_all$Moneyness <= 1.05, ]

avg_bisect = aggregate(IV_bisect ~ Symbol + Expiration + Type, data=band2,
mean, na.rm=TRUE)
avg_newton = aggregate(IV_newton ~ Symbol + Expiration + Type, data=band2,
mean, na.rm=TRUE)
avg_secant = aggregate(IV_secant ~ Symbol + Expiration + Type, data=band2,
mean, na.rm=TRUE)

avg_report = merge(merge(avg_bisect, avg_newton,
by=c("Symbol","Expiration","Type")),
                   avg_secant, by=c("Symbol","Expiration","Type"))

print(avg_report)
```

```
##    Symbol  Expiration Type IV_bisect IV_newton IV_secant
## 1     SPY Apr.17.2026 Call 0.1623674 0.1623674 0.1623674
## 2     SPY Apr.17.2026  Put 0.1816795 0.1816795 0.1816795
## 3     SPY Feb.20.2026 Call 0.2032233 0.2032233 0.2032235
## 4     SPY Feb.20.2026  Put 0.2140074 0.2140074 0.2140074
## 5     SPY Mar.20.2026 Call 0.1729584 0.1729584 0.1729584
## 6     SPY Mar.20.2026  Put 0.1858570 0.1858570 0.1858570
## 7    TSLA Aug.21.2026 Call 0.5033799 0.5033799 0.5033799
## 8    TSLA Aug.21.2026  Put 0.5007313 0.5007313 0.5007313
## 9    TSLA Feb.25.2026 Call 0.4289632 0.4289632 0.4289633
## 10   TSLA Feb.25.2026  Put 0.4301861 0.4301861 0.4301861
## 11   TSLA May.15.2026 Call 0.4867189 0.4867189 0.4867189
## 12   TSLA May.15.2026  Put 0.4827616 0.4827616 0.4827616
```

8. Present a table reporting the implied volatility values obtained

Done above in previous R chunks, see tables above

TSLA implied vol is higher than SPY, which makes sense since TSLA is a single stock vs an entire fully diversified index. SPY's implied volatility is close to the price of the VIX index, which again makes sense since VIX measures expected S&P volatility. As maturity increases, implied volatility generally increases. When options become in or out of the money, we generally see an implied volatility increase as well.

9. For each option in your table calculate the price of the different type

```r
# Split calls and puts
calls = iv_all[iv_all$Type == "Call", ]
puts  = iv_all[iv_all$Type == "Put", ]
```

```r
# Merge by strike + expiration
pc = merge(calls, puts,
           by = c("Symbol","Expiration","Strike","Tau","S0"),
           suffixes = c("_Call","_Put"))
# Parity-implied put price
pc$Put_Implied = pc$Mid_Call - pc$S0 + pc$Strike * exp(-r * pc$Tau)

# Difference vs observed put mid
pc$Diff = pc$Put_Implied - pc$Mid_Put

print(head(pc[abs(pc$Strike - pc$S0) < 10, c("Symbol","Expiration","Strike",
"Mid_Call","Mid_Put","Put_Implied","Diff")], 10))
```

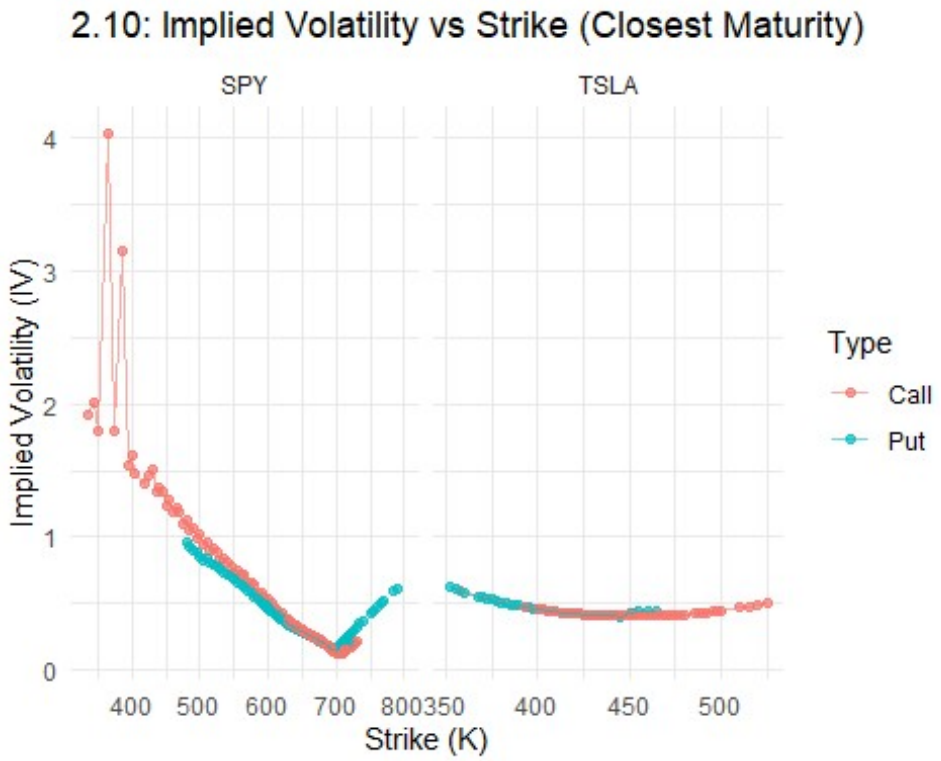
```
##    Symbol  Expiration Strike Mid_Call Mid_Put Put_Implied      Diff
## 63    SPY Apr.17.2026    672   27.075  14.765    13.24944 -1.515560
## 64    SPY Apr.17.2026    673   26.360  15.060    13.52837 -1.531625
## 65    SPY Apr.17.2026    674   25.640  15.355    13.80231 -1.552690
## 66    SPY Apr.17.2026    675   24.970  15.665    14.12625 -1.538755
## 67    SPY Apr.17.2026    676   24.270  15.975    14.42018 -1.554820
## 68    SPY Apr.17.2026    677   23.655  16.300    14.79912 -1.500884
## 69    SPY Apr.17.2026    678   23.000  16.625    15.13805 -1.486949
## 70    SPY Apr.17.2026    679   22.340  16.955    15.47199 -1.483014
## 71    SPY Apr.17.2026    680   21.540  17.295    15.66592 -1.629079
## 72    SPY Apr.17.2026    681   20.860  17.645    15.97986 -1.665144
```

```r
#Since we are using American options not european, BS formulas do not hold
up. Also, TSLA has dividends.
```

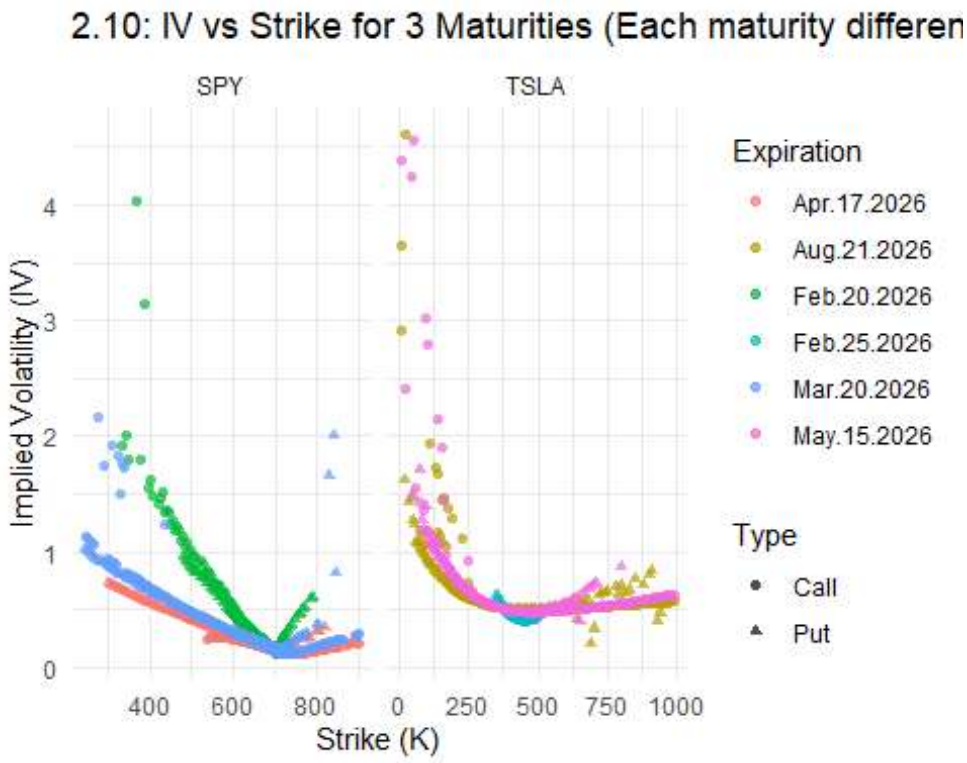
10. Create a 2 dimensional plot of implied volatilities versus strike K

```r
iv_all2 = rbind(iv_results, iv_results_tsla)

closest_exp = aggregate(Tau ~ Symbol + Expiration, data = iv_all2, FUN =
unique)
closest_exp = closest_exp[order(closest_exp$Symbol, closest_exp$Tau), ]
closest_by_symbol = do.call(rbind, lapply(split(closest_exp,
closest_exp$Symbol), function(df) df[1, ]))

iv_closest = merge(iv_all2, closest_by_symbol[, c("Symbol","Expiration")], by
= c("Symbol","Expiration"))

ggplot(iv_closest, aes(x = Strike, y = IV, color = Type)) +
  geom_point(alpha = 0.7) +
  geom_line(alpha = 0.6) +
  facet_wrap(~ Symbol, scales = "free_x") +
  labs(title="2.10: Implied Volatility vs Strike (Closest Maturity)",
       x="Strike (K)", y="Implied Volatility (IV)") +
  theme_minimal()
```

```
## Warning: Removed 11 rows containing missing values or values outside the
scale range
## (`geom_point()`).

## Warning: Removed 1 row containing missing values or values outside the
scale range
## (`geom_line()`).
```




```
exps3 = do.call(rbind, lapply(split(closest_exp, closest_exp$Symbol),
function(df) df[1:min(3,nrow(df)), ]))

iv_3m = merge(iv_all2, exps3[, c("Symbol","Expiration","Tau")], by =
c("Symbol","Expiration","Tau"))

ggplot(iv_3m, aes(x = Strike, y = IV, color = Expiration, shape = Type)) +
  geom_point(alpha = 0.7) +
  facet_wrap(~ Symbol, scales="free_x") +
  labs(title="2.10: IV vs Strike for 3 Maturities (Each maturity different
color)",
       x="Strike (K)", y="Implied Volatility (IV)") +
  theme_minimal()

## Warning: Removed 105 rows containing missing values or values outside the
scale range
## (`geom_point()`).
```




```r
#bs greeks for call
bs_delta_call = function(s0, sigma, tau, k, r){
  if(tau <= 0 || sigma <= 0) return(ifelse(s0 > k, 1, 0))
  d1 = bs_d1(s0, sigma, tau, k, r)
  pnorm(d1)
}

bs_gamma_call = function(s0, sigma, tau, k, r){
  if(tau <= 0 || sigma <= 0) return(NA)
  d1 = bs_d1(s0, sigma, tau, k, r)
  dnorm(d1) / (s0 * sigma * sqrt(tau))
}

bs_vega_call = function(s0, sigma, tau, k, r){
  if(tau <= 0 || sigma <= 0) return(NA)
  d1 = bs_d1(s0, sigma, tau, k, r)
  s0 * dnorm(d1) * sqrt(tau)
}

#finite diff for call
fd_delta_call = function(s0, sigma, tau, k, r, hS){
  (bs_call(s0 + hS, sigma, tau, k, r) - bs_call(s0 - hS, sigma, tau, k, r)) /
(2*hS)
}

fd_gamma_call = function(s0, sigma, tau, k, r, hS){
```

```
  (bs_call(s0 + hS, sigma, tau, k, r) - 2*bs_call(s0, sigma, tau, k, r) +
bs_call(s0 - hS, sigma, tau, k, r)) / (hS^2)
}

fd_vega_call = function(s0, sigma, tau, k, r, hV){
  (bs_call(s0, sigma + hV, tau, k, r) - bs_call(s0, sigma - hV, tau, k, r)) /
(2*hV)
}

calls_iv = iv_all2[iv_all2$Type == "Call" & !is.na(iv_all2$IV) & iv_all2$IV >
0 & !is.na(iv_all2$Tau), ]

calls_iv$S0 = ifelse(calls_iv$Symbol == "SPY", S0_SPY, S0_TSLA)

#steps
calls_iv$hS = 0.01 * calls_iv$S0
hV = 1e-4

#compute greeks
calls_iv$Delta_BS = mapply(bs_delta_call, calls_iv$S0, calls_iv$IV,
calls_iv$Tau, calls_iv$Strike, MoreArgs=list(r=r))
calls_iv$Gamma_BS = mapply(bs_gamma_call, calls_iv$S0, calls_iv$IV,
calls_iv$Tau, calls_iv$Strike, MoreArgs=list(r=r))
calls_iv$Vega_BS  = mapply(bs_vega_call,  calls_iv$S0, calls_iv$IV,
calls_iv$Tau, calls_iv$Strike, MoreArgs=list(r=r))

calls_iv$Delta_FD = mapply(fd_delta_call, calls_iv$S0, calls_iv$IV,
calls_iv$Tau, calls_iv$Strike, MoreArgs=list(r=r), calls_iv$hS)
calls_iv$Gamma_FD = mapply(fd_gamma_call, calls_iv$S0, calls_iv$IV,
calls_iv$Tau, calls_iv$Strike, MoreArgs=list(r=r), calls_iv$hS)
calls_iv$Vega_FD  = mapply(fd_vega_call,  calls_iv$S0, calls_iv$IV,
calls_iv$Tau, calls_iv$Strike, MoreArgs=list(r=r, hV=hV))

#compare
calls_iv$Delta_Diff = calls_iv$Delta_FD - calls_iv$Delta_BS
calls_iv$Gamma_Diff = calls_iv$Gamma_FD - calls_iv$Gamma_BS
calls_iv$Vega_Diff  = calls_iv$Vega_FD  - calls_iv$Vega_BS

greeks_table = calls_iv[, c("Symbol","Expiration","Tau","Strike","S0","IV",
                            "Delta_BS","Delta_FD","Delta_Diff",
                            "Gamma_BS","Gamma_FD","Gamma_Diff",
                            "Vega_BS","Vega_FD","Vega_Diff")]

greeks_table$AbsATM = abs(greeks_table$Strike - greeks_table$S0)
greeks_atm = do.call(rbind, lapply(split(greeks_table,
list(greeks_table$Symbol, greeks_table$Expiration), drop=TRUE),
                                   function(df) df[order(df$AbsATM),
][1:min(10, nrow(df)), ]))
```

```r
greeks_atm = greeks_atm[order(greeks_atm$Symbol, greeks_atm$Expiration,
greeks_atm$AbsATM), ]
greeks_atm$AbsATM = NULL

print(head(greeks_atm, 30))
```

```
##                       Symbol  Expiration        Tau Strike     S0        IV
## SPY.Apr.17.2026.812    SPY Apr.17.2026 0.16712329    682 681.75 0.1642409
## SPY.Apr.17.2026.811    SPY Apr.17.2026 0.16712329    681 681.75 0.1653605
## SPY.Apr.17.2026.813    SPY Apr.17.2026 0.16712329    683 681.75 0.1630941
## SPY.Apr.17.2026.810    SPY Apr.17.2026 0.16712329    680 681.75 0.1667275
## SPY.Apr.17.2026.814    SPY Apr.17.2026 0.16712329    684 681.75 0.1619187
## SPY.Apr.17.2026.809    SPY Apr.17.2026 0.16712329    679 681.75 0.1691232
## SPY.Apr.17.2026.815    SPY Apr.17.2026 0.16712329    685 681.75 0.1607584
## SPY.Apr.17.2026.808    SPY Apr.17.2026 0.16712329    678 681.75 0.1701748
## SPY.Apr.17.2026.816    SPY Apr.17.2026 0.16712329    686 681.75 0.1596115
## SPY.Apr.17.2026.807    SPY Apr.17.2026 0.16712329    677 681.75 0.1711124
## SPY.Feb.20.2026.111    SPY Feb.20.2026 0.01369863    682 681.75 0.2024799
## SPY.Feb.20.2026.110    SPY Feb.20.2026 0.01369863    681 681.75 0.2061168
## SPY.Feb.20.2026.112    SPY Feb.20.2026 0.01369863    683 681.75 0.1988612
## SPY.Feb.20.2026.109    SPY Feb.20.2026 0.01369863    680 681.75 0.2099711
## SPY.Feb.20.2026.113    SPY Feb.20.2026 0.01369863    684 681.75 0.1950626
## SPY.Feb.20.2026.108    SPY Feb.20.2026 0.01369863    679 681.75 0.2134493
## SPY.Feb.20.2026.114    SPY Feb.20.2026 0.01369863    685 681.75 0.1911987
## SPY.Feb.20.2026.107    SPY Feb.20.2026 0.01369863    678 681.75 0.2167412
## SPY.Feb.20.2026.115    SPY Feb.20.2026 0.01369863    686 681.75 0.1872245
## SPY.Feb.20.2026.106    SPY Feb.20.2026 0.01369863    677 681.75 0.2202060
## SPY.Mar.20.2026.447    SPY Mar.20.2026 0.09041096    682 681.75 0.1734499
## SPY.Mar.20.2026.446    SPY Mar.20.2026 0.09041096    681 681.75 0.1751000
## SPY.Mar.20.2026.448    SPY Mar.20.2026 0.09041096    683 681.75 0.1717318
## SPY.Mar.20.2026.445    SPY Mar.20.2026 0.09041096    680 681.75 0.1768086
## SPY.Mar.20.2026.449    SPY Mar.20.2026 0.09041096    684 681.75 0.1700033
## SPY.Mar.20.2026.444    SPY Mar.20.2026 0.09041096    679 681.75 0.1785181
## SPY.Mar.20.2026.450    SPY Mar.20.2026 0.09041096    685 681.75 0.1683218
## SPY.Mar.20.2026.443    SPY Mar.20.2026 0.09041096    678 681.75 0.1746275
## SPY.Mar.20.2026.451    SPY Mar.20.2026 0.09041096    686 681.75 0.1665609
## SPY.Mar.20.2026.442    SPY Mar.20.2026 0.09041096    677 681.75 0.1818294
##                        Delta_BS  Delta_FD    Delta_Diff    Gamma_BS
Gamma_FD
## SPY.Apr.17.2026.812 0.5472487 0.5469777 -2.709875e-04 0.008654160
0.008638944
## SPY.Apr.17.2026.811 0.5556983 0.5554004 -2.979430e-04 0.008571842
0.008557132
## SPY.Apr.17.2026.813 0.5386772 0.5384345 -2.426728e-04 0.008735361
0.008719636
## SPY.Apr.17.2026.810 0.5639689 0.5636464 -3.224940e-04 0.008474765
0.008460625
## SPY.Apr.17.2026.814 0.5299808 0.5297679 -2.129329e-04 0.008815376
0.008799136
```

```
## SPY.Apr.17.2026.809 0.5718227 0.5714818 -3.409797e-04 0.008326222
0.008312890
## SPY.Apr.17.2026.815 0.5211574 0.5209758 -1.816275e-04 0.008891634
0.008874890
## SPY.Apr.17.2026.808 0.5798509 0.5794873 -3.635673e-04 0.008242395
0.008229535
## SPY.Apr.17.2026.816 0.5122108 0.5120620 -1.487614e-04 0.008963935
0.008946700
## SPY.Apr.17.2026.807 0.5878004 0.5874149 -3.855107e-04 0.008161946
0.008149533
## SPY.Feb.20.2026.111 0.5069488 0.5064820 -4.668295e-04 0.024688723
0.024329716
## SPY.Feb.20.2026.110 0.5312286 0.5300974 -1.131171e-03 0.024182414
0.023846279
## SPY.Feb.20.2026.112 0.4817973 0.4820717  2.743950e-04 0.025115626
0.024736281
## SPY.Feb.20.2026.109 0.5545492 0.5528340 -1.715149e-03 0.023588558
0.023277788
## SPY.Feb.20.2026.113 0.4557824 0.4568776  1.095168e-03 0.025473799
0.025076515
## SPY.Feb.20.2026.108 0.5770062 0.5747733 -2.232902e-03 0.022985760
0.022699506
## SPY.Feb.20.2026.114 0.4289493 0.4309452  1.995874e-03 0.025733567
0.025322502
## SPY.Feb.20.2026.107 0.5986181 0.5959293 -2.688762e-03 0.022359286
0.022097272
## SPY.Feb.20.2026.115 0.4013352 0.4043117  2.976447e-03 0.025883540
0.025463937
## SPY.Feb.20.2026.106 0.6192508 0.6161752 -3.075579e-03 0.021682706
0.021445474
## SPY.Mar.20.2026.447 0.5327355 0.5324108 -3.247296e-04 0.011182397
0.011149109
## SPY.Mar.20.2026.446 0.5436914 0.5433069 -3.844515e-04 0.011047735
0.011015788
## SPY.Mar.20.2026.448 0.5215626 0.5213016 -2.610509e-04 0.011315890
0.011281236
## SPY.Mar.20.2026.445 0.5544174 0.5539775 -4.398598e-04 0.010904492
0.010873924
## SPY.Mar.20.2026.449 0.5101646 0.5099716 -1.930070e-04 0.011443951
0.011407944
## SPY.Mar.20.2026.444 0.5649172 0.5644259 -4.912853e-04 0.010756995
0.010727802
## SPY.Mar.20.2026.450 0.4985490 0.4984285 -1.204620e-04 0.011561953
0.011524654
## SPY.Mar.20.2026.443 0.5769026 0.5763229 -5.796301e-04 0.010936825
0.010906300
## SPY.Mar.20.2026.451 0.4867034 0.4866601 -4.331875e-05 0.011677770
0.011639167
## SPY.Mar.20.2026.442 0.5852771 0.5846935 -5.835893e-04 0.010457649
0.010431125
```

```
##                         Gamma_Diff   Vega_BS   Vega_FD     Vega_Diff
## SPY.Apr.17.2026.812 -1.521518e-05 110.40621 110.40621 -1.563086e-07
## SPY.Apr.17.2026.811 -1.470976e-05 110.10150 110.10150 -2.342330e-07
## SPY.Apr.17.2026.813 -1.572531e-05 110.66398 110.66398 -9.269111e-08
## SPY.Apr.17.2026.810 -1.413998e-05 109.75448 109.75448 -3.244678e-07
## SPY.Apr.17.2026.814 -1.623979e-05 110.87280 110.87280 -4.491625e-08
## SPY.Apr.17.2026.809 -1.333284e-05 109.38013 109.38013 -4.153384e-07
## SPY.Apr.17.2026.815 -1.674398e-05 111.03052 111.03052 -1.617903e-08
## SPY.Apr.17.2026.808 -1.286016e-05 108.95219 108.95219 -5.258043e-07
## SPY.Apr.17.2026.816 -1.723576e-05 111.13483 111.13483 -7.933636e-09
## SPY.Apr.17.2026.807 -1.241251e-05 108.48317 108.48317 -6.482557e-07
## SPY.Feb.20.2026.111 -3.590072e-04  31.82789  31.82789 -4.565237e-11
## SPY.Feb.20.2026.110 -3.361353e-04  31.73514  31.73514 -1.650700e-08
## SPY.Feb.20.2026.112 -3.793447e-04  31.79958  31.79958 -1.321819e-08
## SPY.Feb.20.2026.109 -3.107693e-04  31.53468  31.53468 -5.577310e-08
## SPY.Feb.20.2026.113 -3.972842e-04  31.63699  31.63699 -6.211970e-08
## SPY.Feb.20.2026.108 -2.862542e-04  31.23784  31.23784 -1.122118e-07
## SPY.Feb.20.2026.114 -4.110645e-04  31.32653  31.32653 -1.534397e-07
## SPY.Feb.20.2026.107 -2.620134e-04  30.85509  30.85509 -1.811291e-07
## SPY.Feb.20.2026.115 -4.196026e-04  30.85416  30.85416 -2.927929e-07
## SPY.Feb.20.2026.106 -2.372315e-04  30.39975  30.39975 -2.570010e-07
## SPY.Mar.20.2026.447 -3.328725e-05  81.50427  81.50427 -4.568655e-08
## SPY.Mar.20.2026.446 -3.194656e-05  81.28881  81.28881 -9.497133e-08
## SPY.Mar.20.2026.448 -3.465395e-05  81.66026  81.66026 -1.431793e-08
## SPY.Mar.20.2026.445 -3.056841e-05  81.01776  81.01776 -1.603702e-07
## SPY.Mar.20.2026.449 -3.600708e-05  81.75320  81.75320 -3.013270e-09
## SPY.Mar.20.2026.444 -2.919326e-05  80.69462  80.69462 -2.382238e-07
## SPY.Mar.20.2026.450 -3.729921e-05  81.77920  81.77920 -1.499885e-08
## SPY.Mar.20.2026.443 -3.052585e-05  80.25558  80.25558 -3.699622e-07
## SPY.Mar.20.2026.451 -3.860287e-05  81.73432  81.73432 -5.306639e-08
## SPY.Mar.20.2026.442 -2.652324e-05  79.90419  79.90419 -4.252184e-07
```

#Part 4. (Bonus 10 points)

2. Calculate the numerical integral of f1 and f2 by applying the trapeziodal rule

```
f1t = 9/4
f2t = exp(4)-exp(3)-exp(1)+1

f1 = function(x,y){x*y}
f2 = function(x,y){exp(x+y)}
approx1 = c()
approx2 = c()
x = 0
y = 0

dx = c(.1, .04, .01, .001)
dy = c(.2, .04, .005, .0001)

for(l in 1:4){
  #for each delta
```

```
  x = seq(from = 0, to = 1, by = dx[l])
  y = seq(from = 0, to = 3, by = dy[l])
  k = (dx[l]*dy[l])/16

  #approximation
  f1a = 0
  f2a = 0

  for(i in 1:(length(x)-1)){
    for(j in 1:(length(y)-1)){
      #this makes it easier to keep track of
      xi  = x[i]
      xi1 = x[i+1]
      yj  = y[j]
      yj1 = y[j+1]
      xm = (xi + xi1)/2
      ym = (yj + yj1)/2

      #f1 approximation
      f1a = f1a +
k*(f1(xi,yj)+f1(xi,yj1)+f1(xi1,yj)+f1(xi1,yj1)+2*(f1(xm,yj)+f1(xm,yj1)+f1(xi,
ym)+ f1(xi1,ym))+4*f1(xm,ym))

      #f2 approximation
      f2a = f2a +
k*(f2(xi,yj)+f2(xi,yj1)+f2(xi1,yj)+f2(xi1,yj1)+2*(f2(xm,yj)+f2(xm,yj1)+f2(xi,
ym)+ f2(xi1,ym))+4*f2(xm,ym))
    }
  }
  approx1[l] = f1a
  approx2[l] = f2a
}

errors1 = approx1 - f1t
errors2 = approx2 - f2t

output = data.frame(dx, dy, approx1, approx2, errors1, errors2)
print(output)

##      dx    dy approx1  approx2       errors1      errors2
## 1 0.100 2e-01    2.25 32.82849  0.000000e+00 3.416162e-02
## 2 0.040 4e-02    2.25 32.79652  1.776357e-15 2.186311e-03
## 3 0.010 5e-03    2.25 32.79442 -5.329071e-15 8.540191e-05
## 4 0.001 1e-04    2.25 32.79433 -4.884981e-15 6.899421e-07

errors1 = abs(approx1 - f1t)
errors2 = abs(approx2 - f2t)

h = pmax(dx, dy)
```

```
plot(h, errors2, type="b", log="y",xlab="Step Size", ylab="Absolute Error",
col="darkblue", pch=20)
```