Julius Stiemer

Homework 1 – FE621

Full Code attached to the Assignment

Part 1 – Data Gathering

I used Yahoo Finance for this, with the Python package yfinance, I pulled daily closing prices for SPY, TSLA, and ^VIX, as well as Option chains. To fulfill the task, I used February 12 and 13 for both SPY and TSLA.

```
Symbol: TSLA
  Day 1: 2026-02-12, Close: $417.07
  Day 2: 2026-02-13, Close: $417.44
  Option expirations: ['2026-02-18', '2026-02-20', '2026-02-23']

Symbol: SPY
  Day 1: 2026-02-12, Close: $681.27
  Day 2: 2026-02-13, Close: $681.75
  Option expirations: ['2026-02-17', '2026-02-18', '2026-02-19']

Symbol: ^VIX
  Day 1: 2026-02-13, Close: $20.60
  Day 2: 2026-02-16, Close: $21.20

✓ Data download complete!
Risk-free rate: 4.33%
```

- TSLA is a common stock of Tesla, an EV car producer traded on the NASDAQ.
- The SPDR S&P500 ETF Trust has the ticker symbol SPY. ETF refers to a security that gets traded but with multiple underlying assets.
- ^VIX refers to the CBOE Volatility Index, which is a measurement of market forward volatility (30 days) in the S&P.
- **Notes**: Multiple different maturities exist, and different horizons collide, which improves the overall liquidity in the market.

Risk-free rate: **4.33%**

Julius Stiemer

Part 2 - Black-Scholes, Implied Vol., Greeks

Problem 5:

```
Sample parameters:
  S=$100.0, K=$105.0, T=0.5 years, r=5.00%, sigma=25.00%

--- Problem 5 ---
Call: $5.9885
Put:  $8.3960

Put-Call parity check:
  C - P          = -2.407541
  S - K e^(-rT)  = -2.407541
  Difference     = 0.00000000
```

```python
class BS:
    @staticmethod
    def d1(S, K, T, r, sigma):
        return (np.log(S / K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))

    @staticmethod
    def d2(S, K, T, r, sigma):
        return BS.d1(S, K, T, r, sigma) - sigma * np.sqrt(T)

    @staticmethod
    def call(S, K, T, r, sigma):
        d1 = BS.d1(S, K, T, r, sigma)
        d2 = BS.d2(S, K, T, r, sigma)
        return float(S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2))

    @staticmethod
    def put(S, K, T, r, sigma):
        d1 = BS.d1(S, K, T, r, sigma)
        d2 = BS.d2(S, K, T, r, sigma)
        return float(K * np.exp(-r * T) * norm.cdf(-d2) - S * norm.cdf(-d1))

    @staticmethod
    def vega(S, K, T, r, sigma):
        d1 = BS.d1(S, K, T, r, sigma)
        return float(S * np.sqrt(T) * norm.pdf(d1))
```

Julius Stiemer

Problem 6-7:

```
def bisection(self, market_price, S, K, T, r, kind="call"):
    lo, hi = 0.01, 5.0
    it = 0
    for _ in range(self.max_iter):
        it += 1
        mid = 0.5 * (lo + hi)
        price = BS.call(S, K, T, r, mid) if kind == "call" else BS.put(S, K, T, r, mid)
        if abs(price - market_price) < self.tol:
            return mid, it
        if price > market_price:
            hi = mid
        else:
            lo = mid
    return 0.5 * (lo + hi), it

def newton(self, market_price, S, K, T, r, kind="call"):
    sigma = 0.5
    it = 0
    for _ in range(self.max_iter):
        it += 1
        price = BS.call(S, K, T, r, sigma) if kind == "call" else BS.put(S, K, T, r, sigma)
        diff = price - market_price
        if abs(diff) < self.tol:
            return sigma, it

        v = BS.vega(S, K, T, r, sigma)
        if abs(v) < 1e-10:
            # if vega is basically 0 then Newton is unstable
            return sigma, it

        sigma_new = sigma - diff / v
        sigma_new = max(0.001, min(5.0, float(sigma_new)))

        if abs(sigma_new - sigma) < self.tol:
            return sigma_new, it
```

Julius Stiemer

Problem 11:

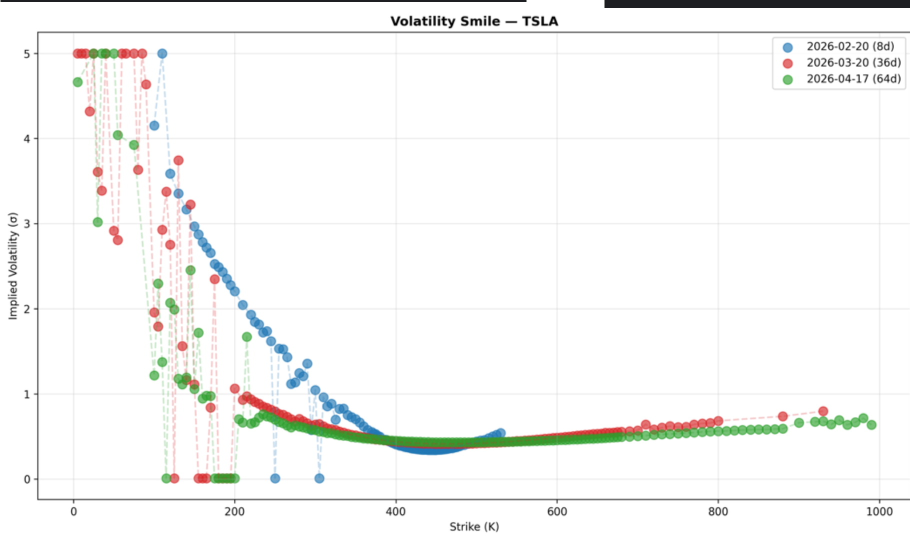
```
class Greeks:
    @staticmethod
    def delta(S, K, T, r, sigma, kind="call"):
        d1 = BS.d1(S, K, T, r, sigma)
        return float(norm.cdf(d1) if kind == "call" else norm.cdf(d1) - 1.0)

    @staticmethod
    def gamma(S, K, T, r, sigma):
        d1 = BS.d1(S, K, T, r, sigma)
        return float(norm.pdf(d1) / (S * sigma * np.sqrt(T)))

    @staticmethod
    def vega(S, K, T, r, sigma):
        return BS.vega(S, K, T, r, sigma)

    @staticmethod
    def delta_cd(S, K, T, r, sigma, kind="call", h=0.01):
        up = BS.call(S + h, K, T, r, sigma) if kind == "call" else BS.put(S + h, K, T, r, sigma)
        dn = BS.call(S - h, K, T, r, sigma) if kind == "call" else BS.put(S - h, K, T, r, sigma)
        return float((up - dn) / (2.0 * h))

    @staticmethod
    def gamma_cd(S, K, T, r, sigma, kind="call", h=0.01):
        up = BS.call(S + h, K, T, r, sigma) if kind == "call" else BS.put(S + h, K, T, r, sigma)
        mid = BS.call(S, K, T, r, sigma) if kind == "call" else BS.put(S, K, T, r, sigma)
        dn = BS.call(S - h, K, T, r, sigma) if kind == "call" else BS.put(S - h, K, T, r, sigma)
        return float((up - 2.0 * mid + dn) / (h**2))

    @staticmethod
    def vega_cd(S, K, T, r, sigma, kind="call", h=0.01):
        up = BS.call(S, K, T, r, sigma + h) if kind == "call" else BS.put(S, K, T, r, sigma + h)
        dn = BS.call(S, K, T, r, sigma - h) if kind == "call" else BS.put(S, K, T, r, sigma - h)
        return float((up - dn) / (2.0 * h))
```

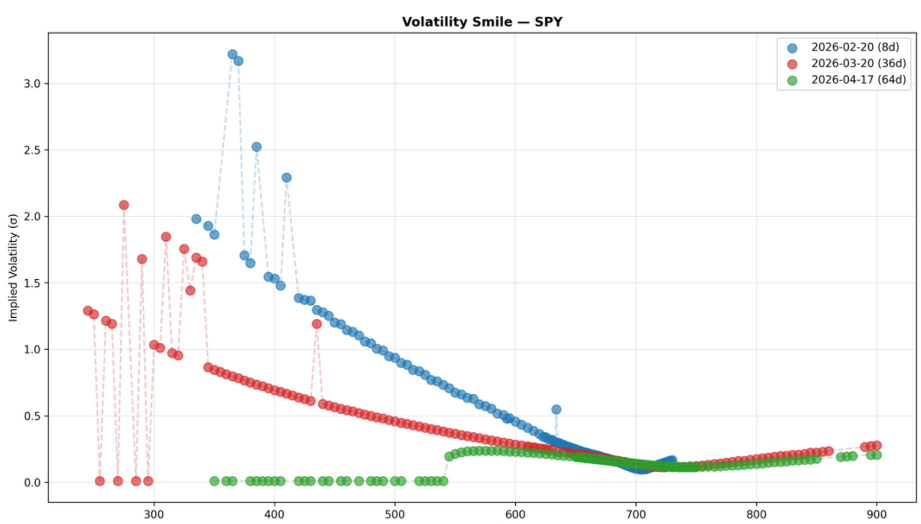
Julius Stiemer

Real Data TSLA:

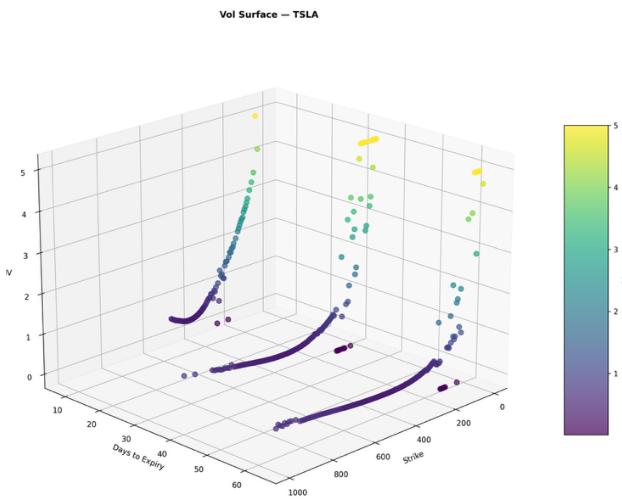
```
Stock Price (S): $417.07
Risk-free rate (r): 4.33%
As-of date: 2026-02-12

--- Maturity 1: 2026-02-18 ---
Time to maturity: 0.0164 years (6 days)
Valid options: 60

--- Maturity 2: 2026-02-20 ---
Time to maturity: 0.0219 years (8 days)
Valid options: 101

--- Maturity 3: 2026-02-23 ---
Time to maturity: 0.0301 years (11 days)
Valid options: 42
```

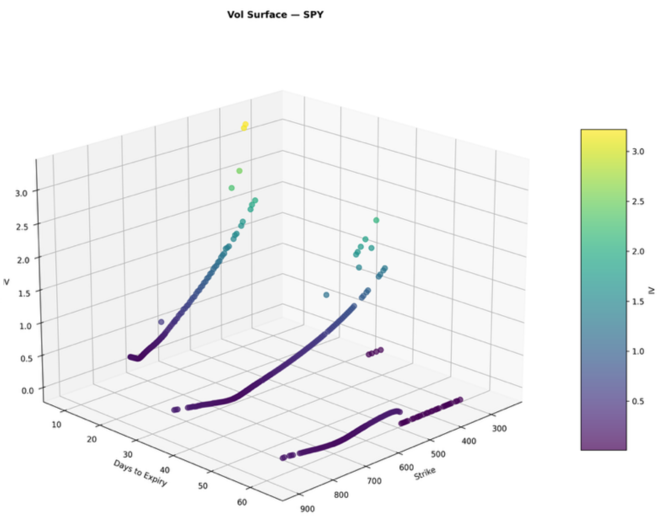
```
Average IV by maturity:
    Expiry    Avg_IV  Time_to_Maturity
2026-02-18 0.432423          0.016438
2026-02-20 1.023903          0.021918
2026-02-23 0.361410          0.030137

----------------------------------------
Method comparison (averages):
  Bisection iterations: 24.1
  Newton iterations:     4.5
  |IV_bis - IV_new|:     0.23484168
```

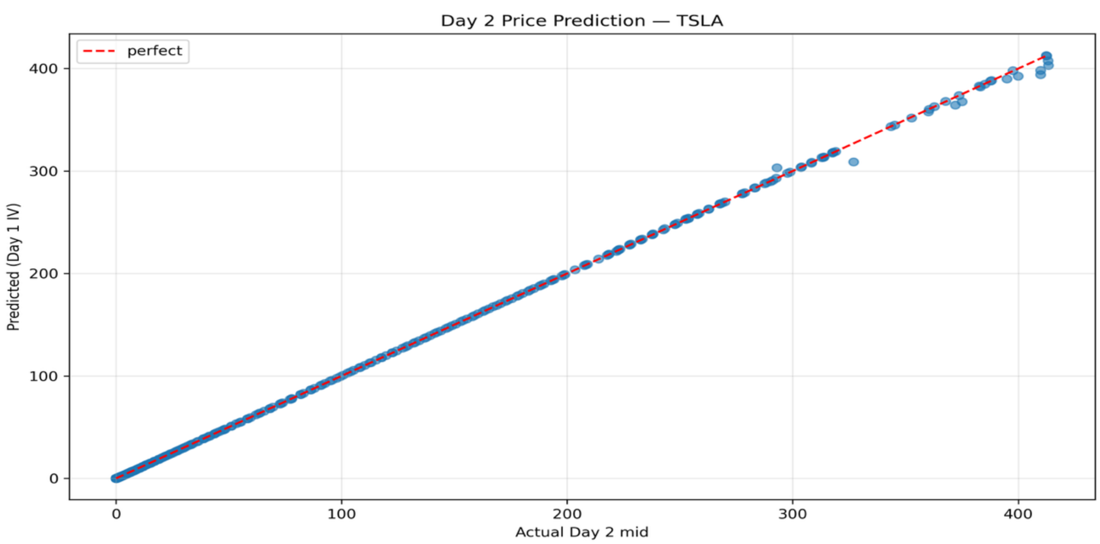


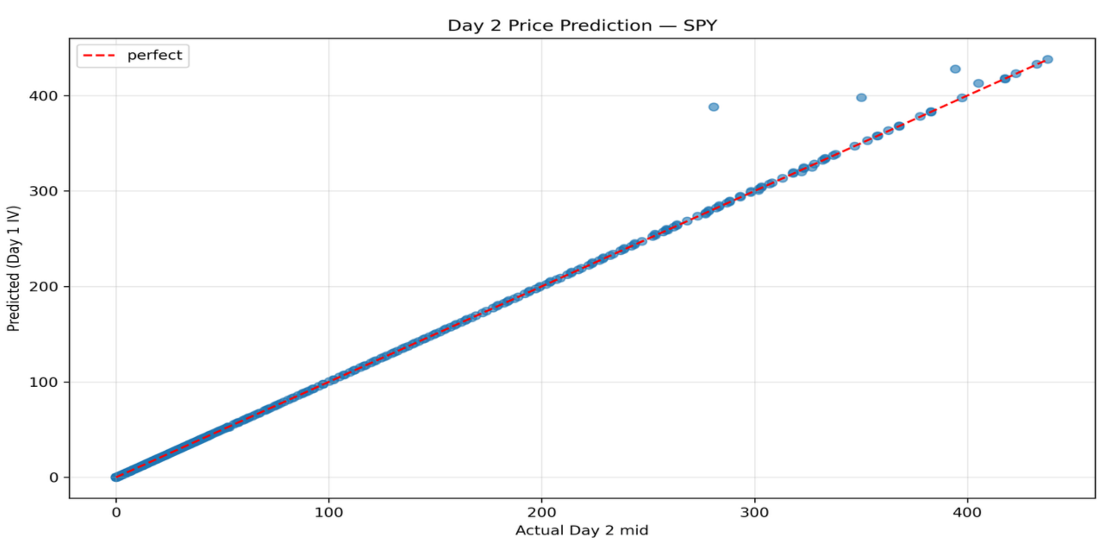


Julius Stiemer









Julius Stiemer

$\boxed{\text{Part 3}}$

3a) Variables: X_t = BTC reserves

Y_t = USDC –"–

$S_t = \frac{Y_t}{X_t}$ = USDC per BTC

y = Fee rate

P_t

Δx ? Δy ? S_t ?

$$X_{t+1} \cdot Y_{t+1} = x_t \cdot y_t = k$$

Case 1 Boundries: $S_{t+1} > \frac{P_t}{1-y}$

$$\frac{P_{t+1}}{1-y} = S_{t+1} = \frac{Y_{t+1}}{X_{t+1}(1-y)}$$

Case 2 Boundries: $S_t < P_t(1-y)$

$$P_{t+1}(1-y) = S_{t+1} = \frac{Y_{t+1}}{X_{t+1}(1-y)}$$

Case 1)

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

$$X_{t+1} \cdot Y_{t+1} = k$$

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

$$\longrightarrow X_{t+1}(S_{t+1}(X_{t+1} \cdot (1-y))) = k$$

$$\cancel{X_{t+1}^2 S_{t+1}(1-y)}$$

$$X_{t+1}^2 S_{t+1}(1-y) = k \quad | : S_{t+1}(1-y)$$

$$X_{t+1}^2 = \frac{k}{S_{t+1}(1-y)} \quad | \sqrt{\ }$$

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

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

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

$$\Delta y = Y_t - Y_{t+1} = Y_t - S_{t+1}\left(\sqrt{\frac{k}{S_{t+1}(1-y)}}\right)(1-y)$$

Case 2)

$$X_{t+1} \cdot Y_{t+1} = k$$

$$\longrightarrow X_{t+1}^2 \frac{S_{t+1}}{1-y} = k \quad | : \frac{S_{t+1}}{1-y}$$

$$X_{t+1}^2 = \frac{k \cdot (1-y)}{S_{t+1}} \quad | \sqrt{\ }$$

$$X_{t+1} = \sqrt{\frac{k \cdot (1-y)}{S_{t+1}}}$$

$$\boxed{\Delta x = X_t - \sqrt{\frac{k(1-y)}{S_{t+1}}} \qquad \Delta Y = y_t - \frac{S_{t+1}}{1-y} \cdot \sqrt{\frac{k(1-y)}{S_{t+1}}}}$$

Julius Stiemer

3b) $x_t = y_t = 1000$, $P_t = \frac{y_t}{x_t} = 1$, $S_t = 1$, $\Delta_t = \frac{1}{365}$

$U_i = \text{Steps}$

$X_i = \frac{P_t}{1-4} + i\,U_1$

$z_i = P_t \cdot (1-4) + j\,U_2$

$g_1(s) = y \cdot \Delta y \cdot f_{S_{t+1}}(s)$

$g_2(s) = g\,\Delta x \cdot s \cdot f_{S_{t+1}}(s)$

$E[R(S_{t+1})] = \ldots$ look task

$$= \sum_{i=1}^{n} U_1 \frac{g_1(y_i - 1) + g_1(x_1)}{2} + \sum_{j=1}^{n_2} U_2 \frac{g_2(z_j - 1) + g_2(z_j)}{2}$$

3c) ?

Part 4

1

$f_1(x,y) = xy$

$= \int_0^1 \int_0^3 xy \, dy \, dx$

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

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

$= \frac{9}{2} \left(|_0^1 \frac{x^2}{2} \right)$

$= \frac{9}{2} \cdot \left(\frac{1}{2}\right) = 9/4$

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

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

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

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

$= (e^3 - 1) \; |_0^1 \; e^x$

$= (e^3 - 1) \cdot (e^1 - 1)$

$= e^4 - e^3 - e + 1$

?