FE 621: HW1

Due date: Sunday Feb 11th at 11:59 pm

- This assignment can be completed individually or in pairs. If working in pairs, clearly state who you are working with.
- Make sure that your solutions are presented in an organized and readable manner. Points are subtracted for solutions that are unclear and difficult to follow or understand.

Problem 1 (analyzing options data)

(a) Collect data:

- Download market prices and implied volatilities for S&P 500 index options (SPX options). You also need the value of the S&P 500 index. The data can be obtained from, e.g., Yahoo Finance or Bloomberg.
- Download risk-free interest rate data from `http://www.federalreserve.gov/releases/H15/Current/`.[1]

(b) Write a function that computes implied volatilities:

- Implement a function that computes the Black-Scholes prices of call and put options with parameters S_0 (stock price), σ (vol), $\tau = T - t$ (time to maturity), K (strike), r (interest rate), and δ (dividend yield).
- Implement a function that uses Newton's method to compute the implied volatility of call and put options. Provide pseudocode for your approach (i.e., provide step-by-step algorithmic instructions).

  Note: Newton's method requires computing the derivative of the Black-Scholes price with respect to the volatility σ. This derivative is known as vega *and it has a closed-form formula in the Black-Scholes model.*

(c) Generate implied volatility smiles:

- Use your function in part (b) to compute the implied volatilities of options with the following maturities: 1 month, 3 months, 6 months, 1 year. You may also consider other maturities. For the market price of an option, use the average of the bid and ask prices.
- For each maturity, plot your computed implied volatilities and the downloaded market implied volatilities. Do this for both call and put options. How do the computed and market volatilities compare?

  Note: Rather than plotting implied volatilities as a function of strike price, you may explore plotting them as a function of the so-called option moneyness. The moneyness is commonly defined as S_0/K, or $\frac{\ln(K/F)}{\sigma_{ATM}\sqrt{\tau}}$, where $F = S_0 e^{r\tau}$ is the forward price, and σ_{ATM} is the implied volatility of the ATM option (i.e., the option with strike $K \approx F$).

(d) Generate an implied volatility surface:

- In a single plot, display the implied volatility smiles for all the maturities considered in part (c). You may use either your computed volatilities or the downloaded market volatilities.

  Note: Do not use a three-dimensional surface. Create a two-dimensional plot with strike (or moneyness) on the x-axis, implied volatility on the y-axis, and with each smile labeled by its time to maturity.
- Comment on how the implied volatility depends on time-to-maturity. For a fixed maturity, comment on the dependence of implied volatility on strike (or moneyness).

[1] Note that multiple types of interest rates are listed. There is not a universal rule when it comes to which one to use, but the *effective federal funds rate* is commonly used in option pricing applications.

(e) Test the put-call parity:

- The put-call parity is a no-arbitrage relation between the prices of call and put options. Show theoretically that the put-call parity implies that call and put options with the same strike and same time to maturity must have the same implied volatility.
- Do the call and put implied volatility smiles in part (c) coincide? If not, are such violations of the put-call parity arbitrage opportunities?

(f) Compute Greeks:

- For the same maturities as in part (c), compute the Black-Scholes delta and gamma of both call and put options. Note that closed-form formulas can be used to compute the delta and gamma of call and put options in the Black-Scholes model.
- Comment on the shapes of the delta and gamma curves for both call and put options, and how they depend on strike (or moneyness) and time to maturity.

Problem 2 (root-finding algorithms)

(a) Write pseudocode for using the bisection method and Newton's method to iteratively compute the square root of a positive number a. Remember to state how you choose the initial search interval for the bisection method, and the initial guess for Newton's method.

Hint: Finding the square root of a is equivalent to finding the positive root of the function $f(x) = x^2 - a$.

(b) Implement your procedures in part (a) with tolerance $\epsilon = 10^{-6}$.

(i) How many iterations do the algorithms need for $a = 0.5$ and $a = 2$?

(ii) To visualize the convergence, plot $|err_n|$ as a function of n ($|err_n|$ should converge to zero), and $\log(|err_n|)$ as a function of $\log(n)$, where $err_n = \sqrt{a} - x_n$ is the error on the n-th iteration of the algorithm.

(iii) Which algorithm seems to converge faster? Is what you observe in line with theoretical results about the rate of convergence of the two methods?

(c) *Extra credit:* Show theoretically that Newton's method has *quadratic order of convergence*. Specifically, denote by x^* a solution of $f(x) = 0$ for some "nice" function f, and let $\epsilon_n = |x_n - x^*|$ be the error on the n-th iteration of Newton's algorithm. Show that

$$\epsilon_{n+1} \leq K\epsilon_n^2,$$

for some constant $K > 0$, which is the definition of quadratic order of convergence.

Remark: For comparison, the error of the bisection method satisfies

$$\epsilon_{n+1} \leq \frac{1}{2}\epsilon_n,$$

where $\epsilon_n = |x_n - x^*|$ is the error on the n-th iteration, and $x_n = (b_n - a_n)/2$ is the halfwidth of the n-th interval (a_n, b_n). This is called linear order of convergence.