The assignment should be done alone; however, discussions with fellow students are encouraged. Matlab programs should be sufficiently documented. **Efficiency matters.**

1. (20 points) Assume that the continuously compounding interest is a constant $r > 0$ and the stock price follows a geometric Brownian motion

$$\frac{dS_t}{S_t} = \mu dt + \sigma dX_t,$$

where $\mu, \sigma$ are positive constants and $X_t$ is a standard Brownian motion.

A forward start contract is specified as follows: at time $T_1$, the holder is given a European call option with exercise price $S(T_1)$ and expiry at time $T_1 + T_2$. What is the value of the option for $0 \leq t \leq T_1$?

2. (80 points) In practice, option prices are usually quoted in a market and volatility is not directly observable. The arbitrage-free option pricing method can potentially be used to glean the volatility information of the underlying asset from the market option prices. In this problem, you will explore this technique.

Assume that the underlying price satisfies the following stochastic differential equation

$$\frac{dS_t}{S_t} = (\mu - q) dt + \sigma(S_t) dX_t,$$  \hspace{1cm} (1)

where the expected rate of return $\mu$ and the continuous dividend yield $q$ are positive constants, $X_t$ is a standard Brownian motion, and $\sigma(S)$ is a deterministic function satisfying some suitable conditions. Assume that the interest rate $r > 0$ is a constant. The European option value $V(S,t)$ satisfies the following backward parabolic partial differential equation,

$$\frac{\partial V}{\partial t} + \frac{1}{2} \sigma^2(S) S \frac{\partial^2 V}{\partial S^2} + (r - q) S \frac{\partial V}{\partial S} - rV = 0, \quad S \in [0, +\infty), t \in [0, T).$$  \hspace{1cm} (2)

(a) What are the appropriate boundary conditions (at $S = 0$ and $S \to +\infty$) for European call and put respectively?

(b) Assume that $S_{\max} > 0$ is sufficiently large and $T$ is the expiry of the options. Let $\delta t = \frac{T}{M}$, $\delta S = \frac{S_{\max}}{N}$, $t_j = (j - 1) \cdot \delta t$, and $S_i = (i - 1) \cdot \delta S$. Let $\{(S_i, t_j)\}_{i=1,\ldots,N+1}^{j=1,\ldots,M+1}$ denote a uniform discretization of the region $[0, S_{\max}] \times [0, T]$. Let $V^j_i$ denote the
option value at \( t_j \) and \( S_i \), and \( \sigma_i \) denote the volatility \( \sigma(S_i) \). Using the Crank-Nicolson finite difference, show that \( x^{j-1} = [V_2^{j-1}; \cdots; V_N^{j-1}] \) satisfies the finite difference equation

\[
A x^{j-1} = b^j, \quad j = M + 1, M, \cdots, 2
\]

where \( A \) is the tridiagonal matrix below

\[
A = \text{diag}(\hat{e}(2:N - 1), -1) + \text{diag}(\hat{d}, 0) + \text{diag}(\hat{f}(1:N - 2), 1)
\]

with

\[
\hat{e}_i = \frac{1}{4}(r - q)S_i^2 \frac{\delta t}{\delta S^2} - \frac{1}{4} \frac{\delta^2 S_i^2}{\delta S^2} \delta S^2
\]

\[
\hat{d}_i = 1 + \frac{1}{2} \sigma_i^2 \delta t + \frac{1}{2} \frac{\delta^2 S_i^2}{\delta S^2} \delta S^2
\]

\[
\hat{f}_i = -\frac{1}{4}(r - q)S_i^2 \frac{\delta t}{\delta S^2} - \frac{1}{4} \frac{\delta^2 S_i^2}{\delta S^2} \delta S^2
\]

In addition, for \( i = 3, 4, \cdots, N - 1, \)

\[
b_{i-1}^j = V_i^j + \frac{1}{4} \sigma_i^2 \frac{\delta t}{\delta S^2} \left( \frac{V_i^{j-1} - 2V_i^j + V_i^{j+1}}{\delta S^2} \right) + \frac{1}{4}(r - q)S_i \delta t \frac{V_i^{j+1} - V_i^{j-1}}{\delta S} - \frac{1}{2} \sigma_i \delta t V_i^j.
\]

and

\[
b_1^j = V_2^j + \frac{1}{4} \sigma_2^2 \frac{\delta S^2}{\delta S} \delta t \frac{V_2^{j-1} - V_2^j + V_2^{j+1}}{\delta S^2} + \frac{1}{4}(r - q)S_2 \delta t \frac{V_2^{j+1} - V_2^{j-1}}{\delta S} - \frac{1}{2} \sigma_2 \delta t V_2^j - \hat{e}_2 V_2^{j-1},
\]

\[
b_{N-1}^j = V_N^j + \frac{1}{4} \sigma_N^2 S_N^2 \delta t \frac{V_N^{j-1} - 2V_N^j + V_N^{j+1}}{\delta S^2} + \frac{1}{4}(r - q)S_N \delta t \frac{V_N^{j+1} - V_N^{j-1}}{\delta S} - \frac{1}{2} \sigma_N \delta t V_N^j - \hat{f}_N V_N^{j-1}.
\]

(c) Using your Crank-Nicolson finite difference method and the final and boundary conditions for the backward PDE, write a Matlab function

\[
V = \text{BSPDEeuro}(S_0, r, q, S_{\text{max}}, T, M, N, E, \text{flag}, \text{sigfcn}, \text{param})
\]

which returns a column vector of the initial European option prices; \( V_i \) is the initial price of the option with the strike \( E_i \). The function returns call prices when \( \text{flag} \) equals 1 and put prices when \( \text{flag} \) equals zero. The function \( \text{sigfcn}(S, \text{param}) \) returns the volatility function value \( \sigma(S) \) at \( S \) (which may depend on some parameters specified by the vector \( \text{param} \)). The argument \( S_0 \) is the initial underlying asset price, \( r \) and \( q \) are constant interest rate and continuous dividend yield, \( S_{\text{max}} > 0 \) is sufficiently large, and \( T > 0 \) is the expiry of the options. You are given Matlab functions TriDiLU, LBidiSol, and UBidiSol to solve the linear equations.
Assume that the underlying asset price follows an absolute diffusion, i.e.,
\[
\frac{dS_t}{S_t} = (\mu - q) dt + \frac{\alpha}{S} dX_t.
\]

Let \( \alpha = 30 \) and \( \sigma(0) = 100 \) (set arbitrarily) in your implementation. In addition, set
\[
S_0 = 100, \quad S_{\max} = 2S_0, \quad T = 1, \quad r = 0.03, \quad q = 0.02, \quad M = 100, \quad N = 200.
\]

Write a Matlab function to compute the specified volatility function and use \texttt{BSPDEuro} to tabulate the initial call option prices with the expiry \( T = 1 \) and strikes \( E = 90 : 2 : 110 \); the first column lists the strikes and the second column lists the corresponding option values.

(d) Suppose your computed call option prices in (c) are the given market European call prices \( \{V^{mkt}(E_i)\}_{i=1}^{11} \) today. Let us pretend that we do not know the volatility function of the underlying asset and want to estimate the volatility from the market prices. In order to do this, let us assume that the unknown volatility function is parameterized by a quadratic function
\[
\sigma(S) = x_1 S^2 + x_2 S + x_3
\]
with unknown parameter \( x = [x_1, x_2, x_3] \). This unknown parameter \( x \) can be estimated by solving a nonlinear least squares problem
\[
\min_{x \in \mathbb{R}^3} f(x) = \frac{1}{2} \sum_{i=1}^{11} \left( V(x; E_i) - V^{mkt}(E_i) \right)^2.
\]

Using \( x_0 = [0, 0, 0.3] \) as the starting point for the Matlab function
\[
x = \texttt{lsqnonlin}(\text{fun}, x_0, \text{options}, P_1, P_2, \cdots),
\]
estimate the unknown coefficients \([x_1; x_2; x_3]\) from the call option prices \( \{V^{mkt}(E_i)\}_{i=1}^{11} \) by writing a Matlab function \( y = \texttt{myfun}(x, P_1, P_2, \cdots) \) which returns the vector
\[
y = \begin{bmatrix}
V(x; E_1) - V^{mkt}(E_1) \\
\vdots \\
V(x; E_{11}) - V^{mkt}(E_{11})
\end{bmatrix}.
\]

Here \( P_1, P_2, \cdots \) are parameters necessary for computing option values, e.g., \( P_1 = S_0, \quad P_2 = r \), etc. Tabulate your computed model coefficients \( x_1, x_2, x_3 \). In one graph, plot the exact volatility function \( \frac{\sigma}{S} \) and your estimated volatility function \( x_1 S^2 + x_2 S + x_3 \) in the interval \([S_0, 2S_0]\). Comment on the accuracy of your approximation.
(e) To check how accurate the estimated volatility function is able to model the option price, using the local volatility function $\frac{2}{3}$ and estimated local volatility function $x_1S^2 + x_2S + x_3$ respectively, tabulate computed initial option values for the same options as in (c) but with the expiry $T = 0.5$. Compare the accuracy.

3. (Bonus Version 100 points) Alternative to the backward PDE, we can regard the European option value $V(E, T; S_0, 0)$ as a function of the strike $E$ and expiry $T$. It can be shown that the following **forward** parabolic partial differential equation is satisfied

$$\frac{\partial V}{\partial T} - \frac{1}{2}(\sigma(E, T))^2E^2\frac{\partial^2 V}{\partial E^2} + (r - q)E\frac{\partial V}{\partial E} + qV = 0$$

and the derivative contract specification gives an initial condition $V(E, 0; S_0, 0) = \max(S_0 - E, 0)$ for a call for example. (This forward equation is discussed in Q19 in Chapter 5 of the textbook).

Do the same (a), (b), (c), (d), and (e) parts of the question 2 using the forward equation (3) instead of the backward (2). How does this approach compare to the approach in question 2 in computational efficiency?

**Note.** If you choose to do the bonus version, then please do not hand in the second question.