Numerical Solutions to PDEs and Option Pricing

Ying Zhang
B068867
MATH11013 Dissertation (Fin. Math.)

A thesis submitted in partial fulfillment of the MSc in
Financial Mathematics
August 2015

Supervisor:
Prof. István Gyöngy
Abstract

This project is to use the finite different method to solve option pricing problems. The relationship between SDEs and PDEs will be investigated and the log transformation is applied to PDE problems. For spatial and fully discretized schemes, the infinite systems of ODEs and equations are truncated so that the numerical method can be implemented, and the truncation error will be discussed. Numerical solutions for European put and call options will be obtained using spatial and fully finite difference methods. Moreover, the accuracy of the solutions will be tested for various mesh size and time steps. The properties of the fully discretized scheme will also be studied. Finally, there will be other European type options considered as the applications of the finite difference methods.

Acknowledgements

Thanks for the help of my supervisor Professor István Gyöngy who give me very useful suggestions for the project. I would also like to thank Dr. David Siska who helped me in programming problems.
Contents

1 Introduction 3

2 Relationship between SDEs and PDEs 5

3 Log Transformation 10

4 Truncation Error 11

5 Spatial finite difference scheme 15
   5.1 Spatial finite difference scheme for the European put option 15
   5.2 Spatial finite difference scheme under log transformation 17

6 Fully discretized scheme 20
   6.1 The implicit finite difference scheme 20
   6.2 Implicit finite difference scheme under log transformation 22
   6.3 The explicit finite difference scheme 24
   6.4 The Crank-Nicolson method 25

7 Consistency, Stability and Convergence 26
   7.1 Consistency 27
   7.2 Stability 28
      7.2.1 Von Neumann analysis for the explicit scheme 28
      7.2.2 Von Neumann analysis for the implicit scheme 29
      7.2.3 Von Neumann analysis for the Crank-Nicolson method 29
   7.3 Convergence 30
      7.3.1 The order of accuracy for the implicit scheme 31
      7.3.2 The order of accuracy for the Crank-Nicolson method 31

8 Apply the numerical scheme to other options 33
   8.1 European options with more general payoff functions 33
   8.2 European style barrier options 33

9 Conclusion 34

References 35

Appendix 37
1 Introduction

In the Black-Scholes market, there are riskless assets and risky assets. The price of a riskless asset at time $t$, which is denoted by $B_t$, follows the stochastic differential equation (SDE)

$$dB_t = rB_t dt, \quad B_0 = 1, \quad B_t = e^{rt},$$

where $r$ is the risk-free rate and $r > 0$. As for the price of a risky asset at time $t$, the process $(S_t)_{t\in[0,T]}$ can be defined as follows

$$dS_t = \mu S_t dt + \sigma S_t dW_t, \quad S_0 > 0,$$

where $\mu \in (-\infty, \infty), \sigma > 0, \mu$ and $\sigma$ are constants, $S_0$ is the spot price and $(W_t)_{t\geq0}$ is a Wiener process. Then

$$S_t = S_0 e^{\sigma W_t + (\mu - \frac{1}{2} \sigma^2)t}, \quad t \geq 0.$$

An option is a financial contract that offers the owner the right, but not the obligation, to buy or sell the underlying instrument at a certain strike price on a specific date or during a certain period of time. For an European call option, the buyer has the right to buy an underlying asset $S$ at the strike price $K$ on the exercise date $T$. From an option buyer's point of view, the payoff $h$ is

$$h := \max(S_T - K, 0),$$

where $S_T$ is the underlying share price at maturity. Since there may be potential gains from holding the options, premiums need to be charged so that there is no arbitrage opportunity in the market. In order to determine the premium (fair price) of the option, the concept of the trading strategy needs to be introduced. $(\psi_t, \phi_t)_{t\in[0,T]}$ is called a trading strategy, where $\psi_t$ is the amount of the riskless asset (bonds) and $\phi_t$ is the amount of the risky asset in a portfolio at time $t$. $(\psi_t)_{t\in[0,T]}$ and $(\phi_t)_{t\in[0,T]}$ are stochastic processes such that $V_t$, the value of the portfolio at time $t$ can be defined in the following way:

$$\psi_t B_t + \phi_t S_t =: V_t(\psi, \phi).$$

A self-financing strategy $(\psi_t, \phi_t)_{t\in[0,T]}$ is a trading strategy, such that

$$dV_t(\psi, \phi) = \psi_t dB_t + \phi_t dS_t, \quad t \in [0, T].$$

This means that for a self-financing portfolio, the losses or gains of the portfolio solely come from the changes in the value of the assets in the portfolio. Furthermore, a self-financing strategy $(\psi_t, \phi_t)_{t\in[0,T]}$ with $V_t(\psi, \phi) \geq 0, t \in [0, T]$ is an admissible strategy. In addition, for a European type option with payoff $h$, a hedging strategy is the strategy $(\psi, \phi)$ such that $V_T(\psi, \phi) \geq h$. Consequently, the fair price for the European type option at time zero is

$$\inf\{V_0(\psi, \phi) : V_T(\psi, \phi) \geq h, (\psi, \phi) \text{ hedging strategies}\}.$$

Besides, the price of a European type option can also be calculated using the following theorem [5].

**Theorem 1.** (Main Theorem on Pricing European Type Options) For a European type option with payoff $h$ at time $T$. Assume $h$ is a non-negative $\mathcal{F}_T$ measurable random variable such
that $E_Q h^2 < \infty$, where $Q$ is the “risk-neutral” measure. Then there is a replicating strategy $(\psi, \phi)$, and the price of the option at time $t$ is $V_t(\psi, \phi)$, which can be calculated using the formula

$$V_t(\psi, \phi) = e^{-r(T-t)} E_Q(h | \mathcal{F}_T).$$

At time zero, it equals to

$$V_0 = e^{-rT} E(g(X_T)),$$

where $g(x) = \max(x - K, 0)$ is the payoff function for the European call option and $X_T$ satisfies the stochastic differential equation SDE

$$dX_t = rX_t dt + \sigma X_t dW_t, \quad X_0 > 0,$$

and

$$X_T = X_0 e^{\sigma W_T + (r - \frac{1}{2} \sigma^2)T}.$$

With some knowledge of the distribution of $\log S_T$, the Black-Scholes formula for the European call option can be derived

$$V_0 = S_0 \Phi(d_1) - Ke^{-rT} \Phi(d_2),$$

where

$$d_1 = \frac{\ln(S_0/K) + (r + \frac{1}{2} \sigma^2)T}{\sigma \sqrt{T}}, \quad d_2 = d_1 - \sigma \sqrt{T},$$

and $\Phi(\cdot)$ is the distribution function of the standard normal distribution. Once the variables in the formula are known, one can calculate the fair price of a European call. As for a corresponding European put, the option price can be derived by the put-call parity. However, for options with complicated payoff functions, there are no explicit formulas to calculate their prices. Numerical methods are therefore employed in this situation to find approximations to the option prices.

In this project, the relationship between SDEs and PDEs is investigated. Then, the log transformation of PDE problems will be discussed. In order to implement the finite difference method, the infinite systems of ODEs and equations need to be truncated to the finite systems and the truncation error will be analyzed. Next, spatial finite difference methods will be applied to the PDE problem to find numerical solutions for option pricing problems. As for the fully discretized methods, explicit scheme, implicit scheme and Crank-Nicolson method will be studied. The numerical results will be compared with the analytical solution obtained using Black-Scholes formula. In addition, the accuracy of numerical approximations is tested, and the consistency, stability and convergence of these schemes will be discussed. Finally, there will be some other pricing examples for European type options.
2 Relationship between SDEs and PDEs

There are relationships between SDEs and PDEs. Consider a process $X_s$ governed by the SDE

$$dX_s = a(X_s)ds + b(X_s)dW_s, \quad s \in [t, T], \quad t \in [0, T],$$

$$X_t = x, \quad x \in \mathbb{R},$$

where $a$ and $b$ are Lipschitz functions, $a = a(x), b = b(x), x \in \mathbb{R}$. $(X^i_s)_{s \in [t, T]}$ is the solution of the SDE. The following is the Kolmogorov backward equation

$$Lu(t, x) = 0$$

for $t \in [0, T], x \in \mathbb{R}$, with the terminal condition

$$u(T, x) = g(x),$$

where $g$ is a Lipschitz function and

$$L := \frac{\partial}{\partial t} + a(x) \frac{\partial}{\partial x} + \frac{1}{2} b^2(x) \frac{\partial^2}{\partial x^2}.$$ 

Then, a theorem [5] can be introduced.

**Theorem 2.** Let $u(t, x) \in C^{1,2}([0, T] \times \mathbb{R})$ be the solution of the above PDE problem, such that $u$ is continuous on $[0, T] \times \mathbb{R}$ and satisfies the polynomial growth

$$|u(t, x)| \leq K(1 + |x|^m)$$

for all $t \in [0, T], x \in \mathbb{R}$, where $m$ and $K$ are positive constants. Then

$$u(t, x) = Eg(X^i_{t, X})$$

for every $t \in [0, T], x \in \mathbb{R}$.

**Proof.** By Itô’s formula,

$$du(s, X_s) = \frac{\partial}{\partial s} u(s, X_s)ds + \frac{\partial}{\partial x} u(s, X_s)dX_s + \frac{1}{2} \frac{\partial^2}{\partial x^2} u(s, X_s)(dX_s)^2,$$

which can be rewritten as

$$du(s, X_s) = Lu(s, X_s)ds + b(X_s) \frac{\partial}{\partial x} u(s, X_s)dW_s.$$ 

Then, integrate from $t$ to $T$

$$u(T, X_s) = u(t, x) + \int_t^T Lu(s, X_s)ds + \int_t^T b(X_s) \frac{\partial}{\partial x} u(s, X_s)dW_s.$$

Since $Lu(t, x) = 0$,

$$u(T, X_s) = u(t, x) + \int_t^T b(X_s) \frac{\partial}{\partial x} u(s, X_s)dW_s.$$
Define a stopping time
\[ \tau_R = \inf\{w \geq t : \int_t^w |b(X_s)|^2 \left( \frac{\partial}{\partial x} u(s, X_s) \right)^2 ds \geq R \}. \]

Then
\[ u(T \wedge \tau_R, X_{T \wedge \tau_R}) = u(t, x) + \int_t^{T \wedge \tau_R} b(X_s) \frac{\partial}{\partial x} u(s, X_s) dW_s. \]

Knowing that \( E \int_t^{T \wedge \tau_R} b^2(X_s) \left( \frac{\partial}{\partial x} u(s, X_s) \right)^2 ds \leq R, \)
\[ Eu(T \wedge \tau_R, X_{T \wedge \tau_R}^{l,x}) = u(t, x). \]

Furthermore, since \( u(t, x) \) satisfies the polynomial growth condition, one can know that
\[ |u(s \wedge \tau_R, X_{s \wedge \tau_R})| \leq K(1 + |X_{s \wedge \tau_R}|^m) \leq K(1 + \sup_{s \leq T} |X_s|^m), \quad \forall s \in [t, T]. \]

By the dominated convergence theorem, as \( \tau_R \to \infty, \)
\[ E \lim_{\tau_R \to \infty} u(T \wedge \tau_R, X_{T \wedge \tau_R}^{l,x}) = \lim_{\tau_R \to \infty} Eu(T \wedge \tau_R, X_{T \wedge \tau_R}^{l,x}). \]

Then
\[ E(u(T, X_{T}^{l,x}))) = u(t, x). \]

Together with \( u(T, x) = g(x), \)
\[ Eu(T, X_{T}^{l,x}) = Eg(X_{T}^{l,x}) = u(t, x). \]

The above discussion illustrates that given an SDE and a PDE problem, the relationship between the two can be found. Similarly, consider the SDE
\[ dX_s = \mu(X_s)ds + \sigma(X_s)dW_s, \quad s \in [t, T], \quad t \in [0, T] \]

with the initial condition
\[ X_t = x, \quad x \in \mathbb{R}, \]
where \( \mu \) and \( \sigma \) are Lipschitz functions, \( \mu = \mu(x), \sigma = \sigma(x), x \in \mathbb{R}. \) Let \( (X_{s}^{l,x})_{s \in [t,T]} \) denote the unique solution of the SDE, which exists by virtue of Itô’s theorem. Meanwhile, consider a PDE problem
\[ L_1 q(t, x) - r(x)q(t, x) = 0, \quad t \in [0, T], \quad x \in \mathbb{R}, \quad (2.1) \]

with the terminal condition
\[ q(T, x) = G(x), \quad x \in \mathbb{R}, \quad (2.2) \]
where \( G \) is a Lipschitz function and
\[ L_1 := \frac{\partial}{\partial t} + \mu(x) \frac{\partial}{\partial x} + \frac{1}{2} \sigma^2(x) \frac{\partial^2}{\partial x^2}. \]

Notice that \( q(t, x) \) is defined on \([0, T) \times \mathbb{R}\) and is continuous in \((t, x) \in [0, T) \times \mathbb{R}. \) \( q(t, x) \)
also satisfies
\[ |q(t, x)| \leq K(1 + |x|^m) \]
for all \( t \in [0, T], x \in \mathbb{R}, \) and \( m, K \) are positive constants. In addition, if the derivatives of \( \mu, \sigma, G \) and \( r \) in \( x \) up to second order are continuous in \((t, x) \in [0, T] \times \mathbb{R}\) and satisfy the polynomial growth condition, then the derivatives \( \frac{\partial}{\partial t} q, \frac{\partial}{\partial x} q \) and \( \frac{\partial^2}{\partial x^2} q \) exist and are continuous in \((t, x) \in [0, T] \times \mathbb{R}(\text{see the book [8]}). \) There is a theorem [4] which can be used to describe the relationship between the above SDE and PDE.
Theorem 3. Assume (2.1)-(2.2) has a solution \( q \in C^{1,2}([0, T] \times \mathbb{R}) \) that is continuous in 
\( (t, x) \in [0, T] \times \mathbb{R} \) and satisfies the polynomial growth condition. Then
\[
q(t, x) = E(\lambda_{T}^{t,x}G(X_{T}^{t,x}))
\]
for every \( t \in [0, T], x \in \mathbb{R} \), where the discount process \( \lambda_{t,x} = (\lambda_{s}^{t,x})_{s \in [t, T]} \) is defined as
\[
\lambda_{s}^{t,x} = e^{-\int_{t}^{s}r(X_{u}^{t,x})du}, \quad s \in [t, T]
\]
for a non-negative Lipschitz function \( r = r(x) \).

Proof. The proof is similar to the previous one. First apply Itô’s formula to \( q(s, x) \) and \( \lambda_{s}q(s, x) \), one can get
\[
dq(s, X_{s}) = \frac{\partial}{\partial s}q(s, X_{s})ds + \frac{\partial}{\partial x}q(s, X_{s})dX_{s} + \frac{1}{2} \frac{\partial^{2}}{\partial x^{2}}q(s, X_{s})(dX_{s})^{2},
\]
and
\[
d(\lambda_{s}q(s, X_{s})) = \lambda_{s}dq(s, X_{s}) - r(X_{s})\lambda_{s}q(s, X_{s})ds.
\]
After some arrangement, the second equation can be simplified as
\[
d(\lambda_{s}q(s, X_{s})) = \lambda_{s}(L_{1}q(s, X_{s}) - r(X_{s})q(s, X_{s}))ds + \lambda_{s}\sigma(X_{s})\frac{\partial}{\partial x}q(s, X_{s})dW_{s}.
\]

Notice that
\[
L_{1}q(s, X_{s}) - r(X_{s})q(s, X_{s}) = 0.
\]
Then, integrate the simplified equation from \( t \) to \( s \), the equation becomes
\[
\lambda_{s}q(s, X_{s}) = q(t, x) + \int_{t}^{s} \lambda_{u}\sigma(X_{u})\frac{\partial}{\partial x}q(u, X_{u})dW_{u}, \quad s \in [0, T).
\]

Next, define
\[
\tau_{M} = \inf\{w \geq t : \int_{t}^{w} |\sigma(X_{u})|^{2}(\lambda_{u})^{2}(\frac{\partial}{\partial x}q(u, X_{u}))^{2}du \geq M\}.
\]

Then,
\[
\lambda_{s \wedge \tau_{M}}q(s \wedge \tau_{M}, X_{s \wedge \tau_{M}}) = q(t, x) + \int_{t}^{s \wedge \tau_{M}} \lambda_{u}\sigma(X_{u})\frac{\partial}{\partial x}q(u, X_{u})dW_{u}.
\]

By taking the expectations of both sides
\[
E(\lambda_{s \wedge \tau_{M}}^{t,x}q(s \wedge \tau_{M}, X_{s \wedge \tau_{M}}^{t,x})) = q(t, x),
\]
since \( E\int_{t}^{s \wedge \tau_{M}} |\sigma(X_{u})|^{2}(\lambda_{u})^{2}(\frac{\partial}{\partial x}q(u, X_{u}))^{2}du \leq M \) and \( E\int_{t}^{s \wedge \tau_{M}} \lambda_{u}\sigma(X_{u})\frac{\partial}{\partial x}q(u, X_{u})dW_{u} = 0 \).

Meanwhile, by the polynomial growth and the fact that \( \lambda_{s \wedge \tau_{M}} \leq 1 \),
\[
|\lambda_{s \wedge \tau_{M}}q(s \wedge \tau_{M}, X_{s \wedge \tau_{M}})| \leq |q(s \wedge \tau_{M}, X_{s \wedge \tau_{M}})| \leq K(1 + \sup_{s \leq T}|X_{s}|^{m})
\]
for all \( s \in [0, T) \). Finally, using the dominated convergence theorem, as \( M \to \infty \),
\[
E \lim_{\tau_{M} \to \infty} (\lambda_{s \wedge \tau_{M}}^{t,x}q(s \wedge \tau_{M}, X_{s \wedge \tau_{M}}^{t,x})) = \lim_{\tau_{M} \to \infty} E(\lambda_{s \wedge \tau_{M}}^{t,x}q(s \wedge \tau_{M}, X_{s \wedge \tau_{M}}^{t,x})).
\]
Then,
\[ E(\lambda_s^t q(s, X_s^t)) = q(t, x), \]
and as \( s \to T \),
\[ E(\lambda_T^t q(T, X_T^t)) = q(t, x). \]
Take into consideration the terminal condition of the PDE problem,
\[ E(\lambda_T^t G(X_T^t)) = q(t, x). \]

Furthermore, consider an SDE
\[ dX_u = \mu(X_u)du + \sigma(X_u)dW_u \]
with the initial condition
\[ X_0 = x. \]
Denote by \((X_s^t)_{s \in [0,T]}\) the solution of the SDE. The probability distribution of \((X_u^s)_{u \in [0,T-t]}\) should be the same as the distribution of \((X_{t+u}^s)_{u \in [0,T-t]}\). Then,
\[ E(\lambda_T^s G(X_T^s)) = E(\lambda_{T-t}^t G(X_{T-t}^s)) = q(t, x), \]
where
\[ \lambda_T^s = e^{-\int_0^s r(X_u^s)du}, \quad s \geq 0. \]

Next, let \((X_s^t)_{s \in [t,T]}\) be the solution of the SDE
\[ dX_s = rX_s ds + \sigma X_s dW_s, \]
where \( \sigma \) is a constant and \( r > 0 \), with the initial condition
\[ X_t = x, \quad x \in \mathbb{R}. \]
Consider the PDE problem
\[ \frac{\partial}{\partial t} v(t, x) + r x \frac{\partial}{\partial x} v(t, x) + \frac{1}{2} \sigma^2 x^2 \frac{\partial^2}{\partial x^2} v(t, x) - rv(t, x) = 0 \]
for \( t \in [0, T), x \in \mathbb{R} \), with the terminal condition
\[ v(T, x) = g(x), \quad x \in \mathbb{R}. \]
\( v \in C^{1,2}([0, T] \times \mathbb{R}) \) is the solution, which is continuous on \([0, T] \times \mathbb{R}\) and satisfies the polynomial growth. Then by the theorem 3,
\[ v(t, x) = E(\lambda_T^s g(X_T^s)), \]
where the discount process \((\lambda_s^t)_{s \in [t,T]}\) is defined as
\[ \lambda_s^t = e^{-r(s-t)}, \]
and
\[ X_T^s = x e^{(r - \frac{1}{2} \sigma^2)(T-t) + \sigma (W_T - W_t)}. \]
In addition, if \((X^x_r)_{r \in [0,T-t]}\) satisfies
\[
dX_s = rX_s ds + \sigma X_s dW_s, \quad X_0 = x,
\]
it is mentioned that the probability distribution of \(X^x_{T-t}\) should be the same as \(X^{t,x}_T\). Thus,
\[
v(t, x) = E(\lambda^{t,x}_T g(X^{t,x}_T)) = E(\lambda^{t,x}_{T-t} g(X^{t,x}_{T-t}))
\]
where \(\lambda^{t,x}_{T-t} = e^{-r(T-t)}\).

In this section, the relationship between SDEs and PDEs is studied. For options without an explicit formula to calculate its price, one can instead consider solving a PDE problem by applying numerical methods to obtain approximations of option prices.
3 Log Transformation

Denote by \( v \) the solution of the PDE problem

\[
\frac{\partial}{\partial t} v(t, y) + r y \frac{\partial}{\partial y} v(t, y) + \frac{1}{2} \sigma^2 y^2 \frac{\partial^2}{\partial y^2} v(t, y) - rv(t, y) = 0
\]

for \( t \in [0, T), y \in \mathbb{R} \), with the terminal condition

\[
v(T, y) = g(y), \quad y \in \mathbb{R}.
\]

Define \( v(t, e^x) = p(t, x) \), and let \( x = \ln y \) for \( y > 0 \). Then,

\[
\frac{\partial}{\partial t} v(t, y) = \frac{\partial}{\partial t} p(t, \ln y),
\]

\[
\frac{\partial}{\partial y} v(t, y) = y^{-1} \frac{\partial}{\partial x} p(t, \ln y),
\]

\[
\frac{\partial^2}{\partial y^2} v(t, y) = -y^{-2} \frac{\partial}{\partial x} p(t, \ln y) + y^{-2} \frac{\partial^2}{\partial x^2} p(t, \ln y).
\]

Substitute (3.3)-(3.5) into the PDE (3.1) and let \( x = \ln y \), the PDE becomes

\[
\frac{\partial}{\partial t} p(t, x) + r \frac{\partial}{\partial x} p(t, x) + \frac{1}{2} \sigma^2 (-\frac{\partial}{\partial x} p(t, x) + \frac{\partial^2}{\partial x^2} p(t, x)) - rp(t, x) = 0.
\]

After the arrangements,

\[
\frac{\partial}{\partial t} p(t, x) + b \frac{\partial}{\partial x} p(t, x) + a \frac{\partial^2}{\partial x^2} p(t, x) - rp(t, x) = 0
\]

for \( t \in [0, T), x \in \mathbb{R} \), with the terminal condition

\[
p(T, x) = g(e^x),
\]

where

\[
a := \frac{1}{2} \sigma^2, \quad b := (r - \frac{1}{2} \sigma^2) .
\]

According to theorem 3, with the log transformation, the function \( p \) can be given by

\[
p(t, x) = E(e^{-r(T-t)}g(e^{Y_{T-t}^x})),
\]

where \((Y_{t}^{s,x})_{s \in [t,T]}\) is the solution of the SDE

\[
dY_s = b ds + \sigma dW_s
\]

with the initial condition \( Y_t = x \). Thus,

\[
Y_{T-t}^x = x + b(T-t) + \sigma (W_T - W_t).
\]

For \((Y_{r}^{x})_{r \in [0,T-t]}\) satisfying the SDE

\[
dY_s = b ds + \sigma dW_s, \quad Y_0 = x,
\]

the probability distribution of \( Y_{T-t}^x \) and \( Y_{T-t}^x \) are the same. Therefore,

\[
Y_{T-t}^x = x + b(T-t) + \sigma W_{T-t},
\]

and

\[
p(t, x) = E(e^{-r(T-t)}g(e^{Y_{T-t}^x})).
\]
4 Truncation Error

The spatial discretized scheme for PDE (3.6)-(3.7) is as follows:

\[ \frac{\partial}{\partial t} p_h(t, x_i) + b \delta_h p_h(t, x_i) + a \delta_{-h} \delta_h p_h(t, x_i) - r p_h(t, x_i) = 0 \]

with the terminal condition

\[ p_h(T, x_i) = g(e^{x_i}), \]

where \( x_i = ih \) for \( i = 0, \pm 1, \pm 2, \ldots \) and for \( h \neq 0, \)

\[ \delta_h p_h(t, x_i) = \frac{1}{h} (p_h(t, x_{i+1}) - p_h(t, x_i)), \]

\[ \delta_{-h} \delta_h p_h(t, x_i) = \frac{1}{h^2} (p_h(t, x_{i+1}) - 2p_h(t, x_i) + p_h(t, x_{i-1})). \]

To solve the system of ODEs numerically, the infinite system should be truncated to a finite one. This means that for sufficiently large \( N, \) \( p_h(t, x_{i+1}) = 0 \) for \( |i| \geq N, \) which results in the truncation error.

The truncation function \( \chi_R(x) \) is a sufficiently smooth nonnegative even function on \( \mathbb{R} \) such that

\[ \chi_R(x) = \begin{cases} 1, & |x| \leq R, \\ 0, & |x| \geq R + 1, \end{cases} \]

and for \( R \leq x \leq R + 1, \) define \( \chi_R(x) = -x + R + 1. \) Then consider the PDE problem:

\[ L_R p_R(t, x) = r_R(x) p_R(t, x) = 0, \quad (t, x) \in [0, T) \times \mathbb{R}, \]

\[ p_R(T, x) = g_R(e^x), \quad x \in \mathbb{R}, \]

where

\[ L_R = \frac{\partial}{\partial t} + b_R(x) \frac{\partial}{\partial x} + a_R(x) \frac{\partial^2}{\partial x^2} \]

with

\[ r_R(x) = r \chi_R(x), \quad g_R(e^x) = g(e^x) \chi_R(x), \]

and

\[ a_R(x) = a \chi_R(x) = \frac{1}{2} \sigma^2 \chi_R(x), \quad b_R(x) = b \chi_R(x) = (r - \frac{1}{2} \sigma^2) \chi_R(x). \]

Let \( p_R(t, x) \in C^{1,2}([0, T) \times \mathbb{R}) \) be the solution of the PDE problem above, and satisfies the polynomial growth condition. Then by the theorem 3,

\[ p_R(t, x) = E(\phi^x_{T-t} g_R(e^{Z^x_{T-t}})), \]

where

\[ \phi^x_{T-t} = e^{r_{T-t} \int_{0}^{T-t} r_R(Z^x_s) ds}, \]

and \( (Z^x_s)_{s \in [0, T-t]} \) is the solution of the SDE

\[ dY_s = b_R(Y_s) ds + \sigma_R(Y_s) dW_s, \quad s \in [0, T-t] \]

with the initial condition

\[ Y_0 = x, \]
where $\sigma_R(x) = \sigma x R(x)$. The truncation error can be estimated by considering

$$
\max_{t \in [0, T]} \max_{x \in [-R_v, R_v]} |p(t, x) - p_R(t, x)|,
$$

where $R_v = v R$, for $v \in (0, 1)$. Then,

$$
|p(t, x) - p_R(t, x)| = |E(\lambda_T^x g(e^{Y^x_T}) - E(\phi^x_T g_R(e^{Z^x_T})))|.
$$

Define a stopping time

$$
\tau^x := \inf \{ s \in [0, T] : |Y^x_s| \geq R \},
$$

and for $\tau^x > T$,

$$
|p(t, x) - p_R(t, x)| = 0.
$$

Then,

$$
|p(t, x) - p_R(t, x)| = |E(1_{\tau^x \leq T}(\lambda_T^x g(e^{Y^x_T}) - \phi^x_T g_R(e^{Z^x_T})))|.
$$

Since $\lambda^x S \leq 1$ and $\phi^x S \leq 1$, 

$$
|p(t, x) - p_R(t, x)| \leq |E(1_{\tau^x \leq T} g(e^{Y^x_T}))| + |E(1_{\tau^x \leq T} g_R(e^{Z^x_T})))|,
$$

and by Cauchy-Schwarz inequality,

$$
|p(t, x) - p_R(t, x)| \leq P(\tau^x \leq T)^{\frac{1}{2}} \{ (E|g(e^{Y^x_T})|^2) \}^{\frac{1}{2}} + (E|g_R(e^{Z^x_T})|^2) \}. \}
$$

For $|x| \leq R_v$,

$$
E|g(e^{Y^x_T})|^2 \leq K^2(1 + e^{mY^x_T}) \leq 2K^2(1 + e^{2mY^x_T}).
$$

Notice that

$$
Y^x_{T-t} = x + b(T-t) + \sigma W_{T-t},
$$

then

$$
E|g(e^{Y^x_T})|^2 \leq 2K^2(1 + e^{2m(x+b(T-t)+\sigma W_{T-t})}) \\
\leq 2K^2(1 + e^{2m(x+b(T-t))}) e^{2m\sigma^2(T-t)} \\
\leq 2K^2(1 + e^{2m(vR+b|T|)}).\]

Meanwhile, for the term $E|g_R(e^{Z^x_T})|^2$, since $|x| \leq R_v$,

$$
E|g_R(e^x)|^2 \leq 2K^2(1 + e^{2mx}) \leq 2K^2(1 + e^{2mvR}).
$$

As for $P(\tau^x \leq T)$,

$$
P(\tau^x \leq T) = P( \sup_{0 \leq s \leq T} |Y^x_s| \geq R) \\
= P( \sup_{0 \leq s \leq T} |x + bs + \sigma W_s| \geq R) \\
\leq P( \sup_{0 \leq s \leq T} |bs + \sigma W_s| \geq R - vR) \\
= P( \sup_{0 \leq s \leq T} |\beta s + W_s| \geq \alpha),
$$

where

$$
\beta := \frac{b}{\sigma}, \quad \alpha := \frac{(1-v)R}{\sigma}.
$$
Since the Wiener process \((W_s)_{s \in [0, T]}\) follows the normal distribution,
\[
P(\tau^x \leq T) = P\left( \sup_{0 \leq s \leq T} |\beta s + W_s| \geq \alpha \right)
= P\left( \sup_{0 \leq s \leq T} (W_s + \beta s) \geq \alpha \right) + P\left( \sup_{0 \leq s \leq T} (W_s - \beta s) \geq \alpha \right).
\]
To calculate \(P(\sup_{0 \leq s \leq T}(W_s + \beta s) \geq \alpha)\), denote by \(\bar{W}_s = W_s + \beta s\). Then define
\[
\gamma := \frac{d\bar{Q}}{d\bar{P}},
\]
where \(\gamma = \exp\left(-\frac{1}{2} \beta^2 T - \beta W_T\right)\). By Girsanov theorem,
\[
P\left( \sup_{0 \leq s \leq T} (W_s + \beta s) \geq \alpha \right)
= E_{\bar{Q}}\left[ \sup_{0 \leq s \leq T} \bar{W}_s \geq \alpha \exp\left(\beta \bar{W}_T\right) \right]
= E_{\bar{Q}}\left[ \sup_{0 \leq s \leq T} \bar{W}_s \geq \alpha \exp\left(-\frac{1}{2} \beta^2 T + \beta W_T\right) \right].
\]
Denote by \(M_T := \sup_{0 \leq s \leq T} W_s\), \(B_x := [M_T \geq \alpha, W_T \leq x]\), then by reflection principle:
\[
P(B_x) = \begin{cases} 
P(W_T \geq 2\alpha - x), & x < \alpha, \\
P(W_T \geq \alpha) - P(W_T \geq x), & x \geq \alpha, 
\end{cases}
\]
and
\[
P(B_x) = \int_{-\infty}^{x} h(y)dy, \quad \forall x \in (-\infty, \infty),
\]
where
\[
h(x) = \frac{\partial}{\partial x} P(B_x) = \begin{cases} 
\frac{1}{\sqrt{2\pi T}} e^{-\frac{2\alpha - x}{2T}}, & x < \alpha, \\
\frac{1}{\sqrt{2\pi T}} e^{-\frac{\alpha^2}{2T}}, & x \geq \alpha.
\end{cases}
\]
Therefore, for every \(\beta > 0\) and \(\alpha > 0\),
\[
E_{\bar{Q}}(1_{M_T \geq \alpha} \exp\left(-\frac{1}{2} \beta^2 T + \beta W_T\right)) = \int_{-\infty}^{\infty} h(x) e^{-\frac{1}{2} \beta^2 T + \beta x} dx
= \frac{1}{\sqrt{2\pi T}} e^{-\frac{1}{2} \beta^2 T} \left( \int_{-\infty}^{0} e^{\beta x} e^{-\frac{2\alpha - x}{2T}} dx + \int_{0}^{\infty} e^{\beta x} e^{-\frac{\alpha^2}{2T}} dx \right)
= \frac{1}{\sqrt{2\pi T}} e^{-\frac{1}{2} \beta^2 T} \int_{0}^{\infty} (e^{\beta(2\alpha - x)} + e^{\beta x}) e^{-\frac{\alpha^2}{2T}} dx.
\]
Then,
\[
P\left( \sup_{0 \leq s \leq T} (W_s + \beta s) \geq \alpha \right)
\leq \frac{1}{\sqrt{2\pi T}} e^{-\frac{1}{2} \beta^2 T} \int_{0}^{\infty} (e^{\beta(2\alpha - x)} + e^{\beta x}) e^{-\frac{\alpha^2}{2T}} dx
\leq \frac{1}{\sqrt{2\pi T}} e^{-\frac{1}{2} \beta^2 T} \int_{0}^{\infty} 2e^{2|\beta|\alpha} e^{\beta x} e^{-\frac{\alpha^2}{2T}} dx
\leq Ce^{2|\beta|\alpha} \int_{0}^{\infty} e^{-\frac{\alpha^2}{2T}} dx
\leq Ce^{2|\beta|\alpha} \int_{0}^{\infty} \frac{x}{\alpha} e^{-\frac{x^2}{2T}} dx
= Ce^{2|\beta|\alpha} \frac{3T}{\alpha} e^{-\frac{\alpha^2}{2T}}
\leq C_1 e^{-\frac{\alpha^2}{2T}},
\]
where $C$ and $C_1$ are constants depending on $b, \sigma$ and $T$. As for $P(\sup_{0 \leq s \leq T}(W_s - \beta s) \geq \alpha)$, the result is the same. This can be shown by denoting

$$W^*_s = W_s - \beta s.$$  

According to Girsanov theorem, define

$$\gamma_1 := \frac{dQ^*}{dP},$$

where $\gamma_1 = \exp(-\frac{1}{2} \beta^2 T - \beta W_T)$, then

$$P(\sup_{0 \leq s \leq T} (W_s - \beta s) \geq \alpha) = E_P(1_{\sup_{0 \leq s \leq T} (W_s - \beta s) \geq \alpha})$$

$$= E_Q^*(1_{\sup_{0 \leq s \leq T} W^*_s \geq \alpha} \exp(-\frac{1}{2} \beta^2 T + \beta W_T))$$

$$= E_P(1_{\sup_{0 \leq s \leq T} W_s \geq \alpha} \exp(-\frac{1}{2} \beta^2 T + \beta W_T))$$

$$= P(\sup_{0 \leq s \leq T} (W_s + \beta s) \geq \alpha).$$

Therefore,

$$P(\tau^x \leq T) \leq 2C_1 \exp(-\frac{\alpha^2}{\pi}).$$

Finally,

$$|p(t, x) - p_R(t, x)| \leq P(\tau^x \leq T)^{\frac{1}{2}} \{(E|g(e^{Y^x_T - t})|^2)^{\frac{1}{2}} + (E|g_R(e^{Z^x_T - t})|^2)^{\frac{1}{2}}\}$$

$$\leq (2C_1 \exp(-\frac{\alpha^2}{\pi}))^{\frac{1}{2}} \{(2K^2(1 + e^{2m(vR + |b|T) + 2m^2\sigma^2T}))^{\frac{1}{2}} + (2K^2(1 + e^{2mvR}))^{\frac{1}{2}}\}$$

$$\leq C_2 \exp(-C_3 R^2),$$

and

$$\sup_{t \in [0, T]} \sup_{x \in [-R, R]} |p(t, x) - p_R(t, x)| \leq C_2 \exp(-C_3 R^2),$$

where $C_2$ is a positive constant depending on $b, \sigma, T, m$ and $K$, and $C_3$ is also a constant depending on $v, \sigma$ and $T$. The localisation error for a more complicated option pricing problem, for example, the American put option, is discussed in [20].
\section{Spatial finite difference scheme}

In order to illustrate the numerical procedures of the finite difference methods, consider an European put option as an example. \(S_0\) is the initial underlying share price of the option, denote by \(K\) and \(T\) its strike price and maturity. Then consider the Black-Scholes PDE (2.3) with the terminal condition

\[ v(T, x) = g(x) = \max(K - x, 0), \quad x \geq 0, \]

where \(r\) is the risk-free rate and \(v(t, x)\) is the solution of the PDE problem.

\subsection{Spatial finite difference scheme for the European put option}

The spatial finite difference scheme for the Black-Scholes PDE and its terminal condition can be written as

\[ \frac{\partial}{\partial t} v^h_R(t, x_i) + rx_i \partial_h v^h_R(t, x_i) + \frac{1}{2} \sigma_i^2 \partial_{kk} v^h_R(t, x_i) - r v^h_R(t, x_i) = 0, \quad (5.1) \]

and

\[ v^h_R(T, x_i) = g(x_i) = \max(K - x_i, 0), \quad (5.2) \]

where \(x_i = ih\), for \(i = 0, 1, \ldots, M\) and \(x = 0, x_M = R\), then for \(h \neq 0\), by central difference,

\[ \delta_h f(t, x_i) = \frac{1}{2h} (f(t, x_{i+1}) - f(t, x_{i-1})), \quad (5.3) \]

and

\[ \delta_{kk} f(t, x_i) = \frac{1}{h^2} (f(t, x_{i+1}) - 2f(t, x_i) + f(t, x_{i-1})), \quad (5.4) \]

This is a system of ODEs with terminal conditions, but MATLAB ode solvers accept ODEs with initial conditions. Thus one needs to make some adjustments to the ODE system. In order to replace terminal conditions by initial conditions, \(T - t\) will be used instead of \(t\). Denote by \(\bar{t} = T - t\), the ODE system (5.1)-(5.2) becomes

\[ - \frac{\partial}{\partial \bar{t}} v^h_R(\bar{t}, x_i) + rx_i \delta_h v^h_R(\bar{t}, x_i) + \frac{1}{2} \sigma_i^2 \delta_{kk} v^h_R(\bar{t}, x_i) - r v^h_R(\bar{t}, x_i) = 0, \quad (5.5) \]

\[ v^h_R(0, x_i) = \max(K - x_i, 0). \quad (5.6) \]

Let \(v_{\bar{t}, i} = v^h_R(\bar{t}, x_i)\), after the substitution and simplification, for \(i = 1, 2, \ldots, M - 1\), (5.5) can be rewritten as

\[ \frac{\partial}{\partial \bar{t}} v_{\bar{t}, i} = a_i v_{\bar{t}, i-1} + b_i v_{\bar{t}, i} + c_i v_{\bar{t}, i+1}, \]

where

\[ a_i = \frac{1}{2} \sigma_i^2 \bar{t}^2 - \frac{1}{2} ri, \quad b_i = -(r + \sigma_i^2 \bar{t}), \quad c_i = \frac{1}{2} \sigma_i^2 \bar{t}^2 + \frac{1}{2} ri. \]

Then in the matrix form:

\[ \begin{bmatrix} b_1 & c_1 \\ a_2 & b_2 & c_2 \\ & \ddots & \ddots \\ & & a_{M-2} & b_{M-2} & c_{M-2} \\ a_{M-1} & b_{M-1} \end{bmatrix} \begin{bmatrix} v_{\bar{t}, 1} \\ v_{\bar{t}, 2} \\ \vdots \\ v_{\bar{t}, M-2} \\ v_{\bar{t}, M-1} \end{bmatrix} = \begin{bmatrix} a_1 v_{\bar{t}, 0} \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \]
where $v_{t,0}$ and $v_{t,M}$ are boundary conditions. $v_{t,0}$ represents the option price at time $\bar{t}$ when the underlying share price is zero. It can be expressed as

$$v_{t,0} = v^h_R(\bar{t}, x_0) = Ke^{-r\bar{t}}, \quad \bar{t} \in [0, T].$$

For $v_{t,M}$, it is the option price at time $\bar{t}$ when the share price is very large. In this case, the option is worthless, hence

$$v_{t,M} = v^h_R(\bar{t}, x_M) = 0, \quad \bar{t} \in [0, T].$$

Together with the boundary conditions, the matrix can be improved as

$$\begin{align*}
\frac{\partial}{\partial t} v_{t,i} = & 
\begin{bmatrix}
-r \\
a_1 & b_1 & c_1 \\
& a_2 & b_2 & c_2 \\
& & a_3 & b_3 & c_3 \\
& & & \ddots & \ddots & \ddots \\
& & & & a_{M-2} & b_{M-2} & c_{M-2} \\
& & & & & a_{M-1} & b_{M-1}
\end{bmatrix}
\begin{bmatrix}
v_{t,0} \\
v_{t,1} \\
v_{t,2} \\
v_{t,3} \\
\vdots \\
v_{t,M-2} \\
v_{t,M-1}
\end{bmatrix},
\end{align*}$$

since

$$\frac{\partial}{\partial t} v_{t,0} = -rv_{t,0}.$$

**Figure 1:** Graph for European put prices at time 0 and time T

Then, set $K = 50, T = 1, r = 0.1, \sigma = 0.4, R = 200$ and $M = 200$. Figure 1 is a graph for the option price at time zero and time $T$. If the current share price $S_0 = 50$, the numerical result for the option price is 5.3987 which is close to the price 5.4011 calculate using Black-Scholes formula. Different values for $S_0$ can be set to get corresponding approximated option prices. Furthermore, to test the accuracy of the spatial finite difference (SFD) scheme, $v_{t,i}$ can be computed for different $R$. To obtain approximations for different mesh sizes $h$, it
is equivalent to test for various number of grid points $M$. Then one can compare these approximations with results from Black-Scholes (BS) formula.

Table 1 is the accuracy test for different $R$ given different $S_0$, it illustrates that the accuracy of the numerical results is related to the value of $R$. For example, when the spot price is relatively small ($S_0 = 20$), more accurate results are obtained for small values of $R$ ($R = 100$), on the contrary, one should choose a relatively large $R$ ($R = 200$), for large values of $S_0$ ($S_0 = 60$). Hence, it’s important to set suitably large values of $R$ for different spot prices to improve the accuracy of numerical solutions. In table 2, numerical results are obtained for different number of grid points $M$, which imply that small mesh size $h(h = \frac{R}{M})$ can result in more accurate approximations.

<table>
<thead>
<tr>
<th>SFD</th>
<th>$R = 100$</th>
<th>$R = 200$</th>
<th>$R = 400$</th>
<th>$R = 500$</th>
<th>BS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0 = 20$</td>
<td>25.3321</td>
<td>25.3324</td>
<td>25.3337</td>
<td>25.3347</td>
<td>25.3320</td>
</tr>
<tr>
<td>$S_0 = 50$</td>
<td>5.3994</td>
<td>5.3987</td>
<td>5.3918</td>
<td>5.3865</td>
<td>5.4011</td>
</tr>
<tr>
<td>$S_0 = 60$</td>
<td>2.9091</td>
<td>2.9135</td>
<td>2.9083</td>
<td>2.9044</td>
<td>2.9153</td>
</tr>
</tbody>
</table>

For fixed $K = 50, T = 1, r = 0.1, \sigma = 0.4$ and $M = 200$

Table 1: Accuracy test for different $R$ and $S_0$

<table>
<thead>
<tr>
<th>SFD</th>
<th>$M = 150$</th>
<th>$M = 200$</th>
<th>$M = 300$</th>
<th>$M = 400$</th>
<th>$M = 500$</th>
<th>BS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0 = 20$</td>
<td>25.3330</td>
<td>25.3324</td>
<td>25.3321</td>
<td>25.3321</td>
<td>25.3320</td>
<td>25.3320</td>
</tr>
<tr>
<td>$S_0 = 60$</td>
<td>2.9157</td>
<td>2.9135</td>
<td>2.9145</td>
<td>2.9148</td>
<td>2.9150</td>
<td>2.9153</td>
</tr>
<tr>
<td>$S_0 = 80$</td>
<td>0.8241</td>
<td>0.8226</td>
<td>0.8229</td>
<td>0.8230</td>
<td>0.8231</td>
<td>0.8231</td>
</tr>
</tbody>
</table>

For fixed $K = 50, T = 1, r = 0.1, \sigma = 0.4$ and $R = 200$

Table 2: Accuracy test for different $M$ and $S_0$

5.2 Spatial finite difference scheme under log transformation

If log transformation is applied to PDEs, those PDEs will have constant coefficients and infinite domain, which may improve numerical schemes [12]. Consider the European put option described above, let $p(t, x) = v(t, e^x)$, then the transformed Black-Scholes PDE is (3.6), with the terminal condition

$$p(T, x) = g(e^x) = \max(K - e^x, 0), \quad x \in \mathbb{R},$$

and $p(t, x)$ is the solution of the PDE problem. Then the spatial discretization for the transformed PDE problem is as follows:

$$\frac{\partial}{\partial t} p_R^K(t, x_i) + b \delta_{x'} p_R^K(t, x_i) + a \delta_{-t'} \delta_R p_R^K(t, x_i) - r p_R^K(t, x_i) = 0, \quad (5.7)$$
and
\[ p_R^H(T, x_i) = \max(K - e^{x_i}, 0), \tag{5.8} \]
where \( x_i = ih', \) for \( i = 0, 1, \ldots, M, \) then \( x_0 = 0, x_M = R' = \ln R \) and \( h' = \frac{R'}{M}. \) For \( h' \neq 0, \)
\( \delta_K p_R^H(t, x_i) \) and \( \delta_{-h'} p_R^H(t, x_i) \) can be expressed using (5.3)-(5.4). Terminal conditions
should again be transformed to initial conditions. Denoted by \( \bar{t} = T - t \) and \( p_{\bar{t},i} = p_R^H(\bar{t}, x_i), \)
and after some calculations, the system of ODEs (5.7) can be expressed as
\[
\frac{\partial}{\partial \bar{t}} p_{\bar{t},i} = ap_{\bar{t},i-1} + bp_{\bar{t},i} + cp_{\bar{t},i+1}
\]
for \( i = 1, \ldots, M - 1, \)
where
\[
a = \frac{\sigma^2}{2h'^2} - \frac{1}{2h'}(r - \frac{\sigma^2}{2}), \quad b = -(r + \frac{\sigma^2}{h'^2}), \quad c = \frac{\sigma^2}{2h'^2} + \frac{1}{2h'}(r - \frac{\sigma^2}{2}).
\]
The boundary conditions are
\[
p_{\bar{t},0} = p_R^H(\bar{t}, x_0) = Ke^{-rt},
p_{\bar{t},M} = p_R^H(\bar{t}, x_0) = 0
\]
for \( \bar{t} \in [0, T]. \) Then in matrix form:
\[
\frac{\partial}{\partial \bar{t}} p_{\bar{t},i} = \begin{bmatrix}
-a & b & c \\
 & a & b & c \\
 & & a & b & c \\
 & & & \ddots & \ddots & \ddots \\
 & & & & a & b & c \\
 & & & & & a & b \\
\end{bmatrix}
\begin{bmatrix}
p_{\bar{t},0} \\
p_{\bar{t},1} \\
p_{\bar{t},2} \\
\vdots \\
p_{\bar{t},M-2} \\
p_{\bar{t},M-1} \\
\end{bmatrix}
\]

One can use ode15 solver in MATLAB to find numerical solutions. These solutions are
the approximations for the European put prices, and can be compared with results from
Black-Scholes formula to check the accuracy of numerical solutions. Let \( K = 50, T = 1, r = 0.1, \sigma = 0.4, R' = 5 \) and \( M = 100. \) For \( \ln S_0 = 3, \) the numerical result is 25.2510, while the
analytical solution is 25.2493. There is a graph for \( p_0,i \) and \( p_T,i \) and it should be noticed that
\( p_T,i \) represents the European put prices at \( t = 0 \) since \( \bar{t} = T - t. \) Then, as shown in the table
3 and table 4 below, the accuracy of numerical results is tested for different \( R' \) and mesh
size \( h'. \) In both tables, the approximations at grid points are of great accuracy.
Figure 2: Graph for European put prices at time 0 and time T

<table>
<thead>
<tr>
<th>SFD</th>
<th>$R' = 4.8$</th>
<th>$R' = 5$</th>
<th>$R' = 6$</th>
<th>BS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ln S_0 = 2.4$</td>
<td>34.219103</td>
<td>34.219103</td>
<td>34.219080</td>
<td>34.219149</td>
</tr>
<tr>
<td>$\ln S_0 = 3.6$</td>
<td>11.681478</td>
<td>11.684037</td>
<td>11.684284</td>
<td>11.683040</td>
</tr>
<tr>
<td>$\ln S_0 = 4.2$</td>
<td>1.911000</td>
<td>1.913807</td>
<td>1.913800</td>
<td>1.913538</td>
</tr>
</tbody>
</table>

For fixed $K = 50, T = 1, r = 0.1, \sigma = 0.4$ and $M = 200$

Table 3: Accuracy test for different $R'$ and $S_0$

<table>
<thead>
<tr>
<th>SFD</th>
<th>$M = 200$</th>
<th>$M = 250$</th>
<th>$M = 300$</th>
<th>$M = 500$</th>
<th>BS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ln S_0 = 2.4$</td>
<td>34.219103</td>
<td>34.219121</td>
<td>34.219130</td>
<td>34.219144</td>
<td>34.219149</td>
</tr>
<tr>
<td>$\ln S_0 = 3.6$</td>
<td>11.684037</td>
<td>11.683633</td>
<td>11.683275</td>
<td>11.683079</td>
<td>11.683040</td>
</tr>
<tr>
<td>$\ln S_0 = 4.2$</td>
<td>1.913807</td>
<td>1.913637</td>
<td>1.913421</td>
<td>1.913429</td>
<td>1.9135</td>
</tr>
</tbody>
</table>

For fixed $K = 50, T = 1, r = 0.1, \sigma = 0.4$ and $R' = 5$

Table 4: Accuracy test for different $M$ and $S_0$
6 Fully discretized scheme

For fully discretized scheme, one needs to set up grids with respect to time and space. Implicit scheme, explicit scheme and Crank-Nicolson method are three ways to deal with option pricing problems. These schemes will be discussed respectively and the notations used in this section are the same as described in section 5.

Now, consider a European call option with strike price $K$ and maturity $T$. The numerical methods will be applied to the PDE problem (2.3) with the terminal condition

$$v(T, x) = \max(x - K, 0), \quad x \geq 0. \quad (6.1)$$

6.1 The implicit finite difference scheme

The fully discretized scheme for the PDE problem (2.3) and (6.1) is as follows:

$$\delta_t v_R^{\tau h}(t_j, x_i) + rx_i \delta_x v_R^{\tau h}(t_j, x_i) + \frac{1}{2} \sigma^2 x_i^2 \delta_{xx} v_R^{\tau h}(t_j, x_i) - rv_R^{\tau h}(t_j, x_i) = 0, \quad (6.2)$$

$$v_R^{\tau h}(T, x_i) = \max(x_i - K, 0), \quad (6.3)$$

where $t_j = j \tau$, for $j = 0, 1, \ldots, N$, then, $t_0 = 0, t_N = T$ and $\tau = \frac{T}{N}$, and $x_i = ih$ for $i = 0, 1, \ldots, M$. For $\tau \neq 0$,

$$\delta_t f(t_j, x_i) = \frac{1}{\tau} (f(t_{j+1}, x_i) - f(t_j, x_i)), \quad (6.4)$$

and one can use (5.3)-(5.4) as the expressions for $\delta_h v_R^{\tau h}(t_j, x_i)$ and $\delta_{xx} v_R^{\tau h}(t_j, x_i)$ at time $t_j$. Then denote by $v_{j,i} = v_R^{\tau h}(t_j, x_i)$, the simplified equation for (6.2) can be rewritten as

$$v_{j+1,i} = A_i v_{j,i-1} + B_i v_{j,i} + C_i v_{j,i+1} \quad (6.5)$$

for $i = 1, 2, \ldots, M - 1$ and $j = 0, 1, \ldots, N - 1$, where

$$A_i = (-\frac{1}{2} \sigma^2 i^2 + \frac{1}{2} ri) \tau, \quad B_i = 1 + r \tau + \sigma^2 \tau^2, \quad C_i = -(\frac{1}{2} \sigma^2 i^2 + \frac{1}{2} ri) \tau.$$

One can express the system of equations in matrix form as follows to facilitate further numerical calculations.

$$\begin{bmatrix} v_{j+1,1} \\ v_{j+1,2} \\ \vdots \\ v_{j+1,M-1} \end{bmatrix} = \begin{bmatrix} B_1 & C_1 \\ A_2 & B_2 & C_2 \\ \vdots & \vdots & \vdots \\ A_{M-1} & B_{M-1} & C_{M-1} \end{bmatrix} \begin{bmatrix} v_{j,1} \\ v_{j,2} \\ \vdots \\ v_{j,M-1} \end{bmatrix} + \begin{bmatrix} A_1 v_{j,0} \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

where $v_{j,0}$ and $v_{j,M}$ are the boundary conditions. $v_{j,0}$ stands for the option price at time $j$ when the underlying share price is zero. For a European call option, it's obvious that

$$v_{j,0} = 0$$
for \( j = 0, 1, \ldots, N \). As for \( v_{j,M} \), it is the option price at time \( j \) when the underlying share price is very large, then in the numerical scheme it should be equal to:

\[
v_{j,M} = R - e^{-r(T-j\tau)}K.
\]

After specifying these conditions, set \( S_0 = 50, K = 60, T = 1, r = 0.05, \sigma = 0.2, R = 100, N = 100 \) and \( M = 100 \) to check whether the numerical scheme works properly. The numerical solution obtained is 1.6224, while the result calculated using Black-Scholes formula gives 1.6237. There are two graphs in figure 3. Figure 3(a) shows the relationship between the option prices and the underlying share prices. The blue line is the option payoff at maturity, while the red line is the numerical approximations of option prices at time zero. As for the figure 3(b), it illustrates the European call prices at different time points and underlying share prices. Then, the accuracy of fully finite difference (FFD) approximations can be tested for various \( R, h \) and \( \tau \), which is similar to test the accuracy for different values of \( R, M \) and \( N \). The followings are tables of the results.

<table>
<thead>
<tr>
<th>FFD</th>
<th>( R = 100 )</th>
<th>( R = 200 )</th>
<th>( R = 250 )</th>
<th>( R = 400 )</th>
<th>BS</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_0 = 30 )</td>
<td>0.0016</td>
<td>0.0017</td>
<td>0.0018</td>
<td>0.0021</td>
<td>0.0014</td>
</tr>
<tr>
<td>( S_0 = 40 )</td>
<td>0.1460</td>
<td>0.1464</td>
<td>0.1468</td>
<td>0.1483</td>
<td>0.1439</td>
</tr>
<tr>
<td>( S_0 = 60 )</td>
<td>6.2662</td>
<td>6.2631</td>
<td>6.2607</td>
<td>6.2506</td>
<td>6.2704</td>
</tr>
</tbody>
</table>

For fixed \( K = 60, T = 1, r = 0.05, \sigma = 0.2, N = 200 \) and \( M = 200 \)

Table 5: Accuracy test for different \( R \) and \( S_0 \)

Table 5 is the accuracy test for various \( R \). In the numerical scheme, \( R \) represents the infinity of the underlying share price. It can be noticed that when \( R \) is a large value, for example \( R = 400 \), the numerical approximations are less accurate. Thus, one needs to choose more relevant values of \( R \) for different spot prices to make the numerical method work properly. In the table 6, numerical results are obtained for different mesh size \( M \) and step size \( N \). The table illustrates that as \( M \) and \( N \) become larger, the approximations of option prices converge to the analytical results.
### 6.2 Implicit finite difference scheme under log transformation

Now apply the fully discretized scheme to the PDE (3.6) and its terminal condition

\[ p(T, x) = \max(e^x - K, 0) \quad x \in \mathbb{R}. \]

Then, the PDE problem becomes

\[
\begin{align*}
\delta_x p_{R'}^{r,h'}(t_j, x_i) + b \delta_{h'} p_{R'}^{r,h'}(t_j, x_i) + a \delta_{-h'} \delta_{h'} p_{R'}^{r,h'}(t_j, x_i) - rp_{R'}^{r,h'}(t_j, x_i) &= 0, \\
\end{align*}
\]

and

\[
p_{R'}^{r,h'}(T, x_i) = \max(e^{x_i} - K, 0),
\]

where \( t_j = j\tau \) for \( j = 0, 1, \ldots, N \), \( t_N = T \) and \( x_i = ih' \) for \( i = 0, 1, \ldots, M \), \( x_M = R' \). For \( h' \neq 0 \) and \( \tau \neq 0 \), (6.6) can be simplified by substituting (5.3), (5.4) and (6.4) into it. Denote by \( p_{j,i} = p_{R'}^{r,h'}(t_j, x_i) \), the simplified scheme can be expressed as:

\[
 p_{j+1,i} = Ap_{j,i} + Bp_{j,i} + Cp_{j,i+1}
\]

for \( i = 1, 2, \ldots, M - 1 \) and \( j = 0, 1, \ldots, N - 1 \), where

\[
 A = \frac{\tau}{2h'^2}(h'(r - \frac{1}{2}\sigma^2) - \sigma^2), \quad B = 1 + (r + \frac{\sigma^2}{h'^2})\tau, \quad C = -\frac{\tau}{2h'^2}(h'(r - \frac{1}{2}\sigma^2) + \sigma^2).
\]

Then, rewrite the system of equations in matrix form:

\[
\begin{bmatrix}
p_{j+1,1} \\
p_{j+1,2} \\
p_{j+1,3} \\
\vdots \\
p_{j+1,M-2} \\
p_{j+1,M-1}
\end{bmatrix}
= \begin{bmatrix}
B & C \\
A & B & C \\
& A & B & C \\
& & \ddots & \ddots & \ddots \\
& & & A & B
\end{bmatrix}
\begin{bmatrix}
p_{j,1} \\
p_{j,2} \\
p_{j,3} \\
\vdots \\
p_{j,M-2} \\
p_{j,M-1}
\end{bmatrix}
+ \begin{bmatrix}
Ap_{j,0} \\
0 \\
0 \\
\vdots \\
0 \\
Cp_{j,M}
\end{bmatrix},
\]
Figure 4: Graphs for European call prices with log transformation

where \( p_{j,0} \) and \( p_{j,M} \) are boundary conditions with the expressions

\[
p_{j,0} = 0, \quad p_{j,M} = e^{R'} - e^{-r(T-j\tau)} K
\]

for \( j = 0, 1, \ldots, N \).

Let \( K = 35, T = 1, r = 0.05, \sigma = 0.2, R' = 5, N = 200 \) and \( M = 200 \). Consider \( \ln S_0 = 3 \) as an example. The numerical result give 0.0104, while the result from Black-Scholes formula is 0.0095. The are graphs, figure 4(a) and 4(b), for the European call prices, and the following tables are the accuracy test of numerical approximations for various \( R' \), \( M \) and \( N \).

In table 7, numerical results for call option prices are obtained for various \( R' \). Table 8 gives numerical approximations for different \( M \) and \( N \), which indicates that in order to get more accurate approximations, one needs to set relatively large values for both \( M \) and \( N \).

<table>
<thead>
<tr>
<th>FFD</th>
<th>( R' = 4.8 )</th>
<th>( R' = 5 )</th>
<th>( R' = 6 )</th>
<th>BS</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \ln S_0 = 3 )</td>
<td>0.0103</td>
<td>0.0104</td>
<td>0.0106</td>
<td>0.0095</td>
</tr>
<tr>
<td>( \ln S_0 = 3.6 )</td>
<td>4.7369</td>
<td>4.7377</td>
<td>4.7393</td>
<td>4.7410</td>
</tr>
<tr>
<td>( \ln S_0 = 4.2 )</td>
<td>33.3940</td>
<td>33.3941</td>
<td>33.3942</td>
<td>33.3939</td>
</tr>
</tbody>
</table>

For fixed \( K = 35, T = 1, r = 0.05, \sigma = 0.2, N = 200 \) and \( M = 200 \)

Table 7: Accuracy test for different \( R' \) and \( S_0 \)
<table>
<thead>
<tr>
<th>$FFD$</th>
<th>$M = 200$</th>
<th>$M = 250$</th>
<th>$M = 350$</th>
<th>$M = 450$</th>
<th>$M = 600$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N = 200$</td>
<td><strong>4.7377</strong></td>
<td>4.7383</td>
<td>4.7382</td>
<td>4.7382</td>
<td>4.7391</td>
</tr>
<tr>
<td>$N = 250$</td>
<td>4.7380</td>
<td>4.7386</td>
<td>4.7386</td>
<td>4.7386</td>
<td>4.7394</td>
</tr>
<tr>
<td>$N = 350$</td>
<td>4.7385</td>
<td>4.7391</td>
<td>4.7390</td>
<td>4.7390</td>
<td>4.7399</td>
</tr>
<tr>
<td>$N = 450$</td>
<td>4.7387</td>
<td>4.7393</td>
<td>4.7393</td>
<td>4.7393</td>
<td>4.7401</td>
</tr>
<tr>
<td>$N = 600$</td>
<td>4.7389</td>
<td>4.7395</td>
<td>4.7395</td>
<td>4.7395</td>
<td><strong>4.7403</strong></td>
</tr>
</tbody>
</table>

For fixed $\ln S_0 = 3.6, K = 35, T = 1, r = 0.05, \sigma = 0.2$ and $R' = 5$.

The result obtained by the Black-Scholes formula is 4.7410.

**Table 8: Accuracy test for different $M$ and $N$**

### 6.3 The explicit finite difference scheme

Different from the implicit scheme, derivative with respect to time in the explicit scheme is approximated by a forward difference, which means that the expression (6.4) for $\delta_r f(t_j, x_i)$ in the equation (6.2) should be replaced by $\bar{\delta}_r f(t_j, x_i)$, where

$$\bar{\delta}_r f(t_j, x_i) = \frac{1}{\tau}(f(t_{j+1}, x_i) - f(t_j, x_i)). \quad (6.7)$$

Then the explicit scheme for the Black-Scholes PDE can be written as

$$v_{j-1,i} = \bar{A}_i v_{j,i-1} + \bar{B}_i v_{j,i} + \bar{C}_i v_{j,i+1} \quad (6.8)$$

for $i = 1, 2, \ldots, M - 1$ and $j = 1, \ldots, N - 1, N$, where

$$\bar{A}_i = \left(\frac{1}{2}\sigma^2 i^2 - \frac{1}{2}ri\right)\tau, \quad \bar{B}_i = 1 - r\tau - \sigma^2 i^2 \tau, \quad \bar{C}_i = \left(\frac{1}{2}\sigma^2 i^2 + \frac{1}{2}ri\right)\tau.$$  

Together with the terminal and boundary conditions:

$$v_R^h(t_j, x_0) = 0, \quad v_R^h(t_j, x_M) = R - e^{-r(T-t_j)}K, \quad v_R^h(T, x_i) = \max(x_i - K, 0).$$

Then the scheme can be implemented in MATLAB to approximate option prices. However, the explicit scheme is unstable, and incorrect choices of grid will lead to numerical instability. For example, let $S_0 = 50, K = 40, T = 1, r = 0.05, \sigma = 0.2, R = 100, M = 50$ and $N = 100$, the numerical result is 12.2878, while the Black-Scholes formula gives 12.2944. The numerical scheme works properly in this case and the approximation is fairly accurate. But if the value of $M$ is set to be 100, the result given by the explicit scheme turns out to be $7.8568e+45$, which implies the instability of explicit method.
6.4 The Crank-Nicolson method

Crank-Nicolson method combines the implicit and explicit scheme which can be expressed as

\[
\frac{v_{j,i} - v_{j-1,i}}{\tau} = \frac{r}{2}(v_{j,i} + v_{j-1,i}) - \frac{rx_i}{2} \left( \frac{v_{j,i+1} - v_{j,i-1}}{2h} + \frac{v_{j-1,i+1} - v_{j-1,i-1}}{2h} \right) - \frac{\sigma^2 x_i^2}{4} \left( \frac{v_{j,i+1} - 2v_{j,i} + v_{j,i-1}}{h^2} + \frac{v_{j-1,i+1} - 2v_{j-1,i} + v_{j-1,i-1}}{h^2} \right).
\]

After some calculations, one can get the following expression:

\[
a_i^* v_{j+1,i-1} + (1 + b_i^*) v_{j+1,i} + c_i^* v_{j+1,i+1} = -a_i^* v_{j,i-1} + (1 - b_i^*) v_{j,i} - c_i^* v_{j,i+1},
\]

where

\[
a_i^* = \frac{r \sigma^2}{4} (\sigma^2 i^2 - ri), \quad b_i^* = -\frac{r}{2} (r + \sigma^2 i^2), \quad c_i^* = \frac{r}{4} (\sigma^2 i^2 + ri).
\]

In the matrix form:

\[
M_i^* v_j = M_2^* v_{j+1} + \Gamma,
\]

where

\[
v_j = [v_{j,1}, v_{j,2}, v_{j,3}, \ldots, v_{j,M-2}, v_{j,M-1}]^T,
\]

\[
\Gamma = [a_i^*(v_{j,0} + v_{j+1,0}), 0, 0, \ldots, 0, c_{M-1}^*(v_{j,M} + v_{j+1,M})]^T,
\]

\[
M_1^* = \begin{bmatrix}
1 - b_1^* & -c_1^* \\
-a_2^* & 1 - b_2^* & -c_2^*
\end{bmatrix}
\]

\[
M_2^* = \begin{bmatrix}
1 + b_1^* & c_1^* \\
a_2^* & 1 + b_2^* & c_2^*
\end{bmatrix}
\]

After specifying the terminal and boundary conditions for the European call option, one can get numerical approximations. The table 9 provides the comparison for the prices of the European call option between Black-Scholes results and approximations obtained using finite difference schemes (implicit scheme and Crank-Nicolson method). It illustrates that the Crank-Nicolson method which combines explicit and implicit schemes can improve the accuracy of the numerical approximations.
7 Consistency, Stability and Convergence

The approximated solutions calculated using numerical methods are expected to converge to the solutions of the differential equations. However, in the section 6.3, there is an example indicating that the numerical solutions of the explicit scheme do not approximate the exact solutions when the grids are not properly chosen. Therefore, the properties of the numerical methods need to be studied and the discussions are mainly about the fully discretized schemes. In order to facilitate the analysis, first transform the PDE problem (3.6)-(3.7) to the heat equation \[17\].

Denote by \(\tilde{t} = \frac{\sigma^2}{2} (T - t)\) and \(p(t, x) = q(\tilde{t}, x)\), then,

\[
\frac{\partial}{\partial \tilde{t}} p = -\frac{\sigma^2}{2} \frac{\partial}{\partial \tilde{t}} q.
\]

The PDE (3.6) becomes

\[
\frac{\partial}{\partial \tilde{t}} q(\tilde{t}, x) - \left(\frac{2r}{\sigma^2} - 1\right) \frac{\partial}{\partial x} q(\tilde{t}, x) - \frac{\partial^2}{\partial x^2} q(\tilde{t}, x) + \frac{2r}{\sigma^2} q(\tilde{t}, x) = 0.
\] (7.1)

Let \(t = \tilde{t}\), rewrite (7.1) as

\[
\frac{\partial}{\partial t} q(t, x) = \frac{\partial^2}{\partial x^2} q(t, x) + \left(\frac{2r}{\sigma^2} - 1\right) \frac{\partial}{\partial x} q(t, x) - \frac{2r}{\sigma^2} q(t, x)
\] (7.2)

for \(x \in \mathbb{R}\) and \(t \in [0, \frac{\sigma^2}{2} T]\), with the initial condition

\[
q(0, x) = g(e^x), \quad x \in \mathbb{R}.
\]

Furthermore, set \(q(t, x) = e^{\alpha x + \beta t} u(t, x) = \gamma u(t, x)\)

\[
\frac{\partial}{\partial \tilde{t}} q = \beta \gamma u + \gamma \frac{\partial}{\partial \tilde{t}} u,
\]

\[
\frac{\partial}{\partial x} q = \alpha \gamma u + \gamma \frac{\partial}{\partial x} u,
\]

\[
\frac{\partial^2}{\partial x^2} q = \alpha^2 \gamma u + 2\alpha \gamma \frac{\partial}{\partial x} u + \gamma \frac{\partial^2}{\partial x^2} u.
\]

<table>
<thead>
<tr>
<th>(S_0)</th>
<th>Black-Scholes</th>
<th>Implicit Scheme</th>
<th>Crank-Nicolson</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>0.2207</td>
<td>0.2227</td>
<td>0.2208</td>
</tr>
<tr>
<td>37</td>
<td>0.4181</td>
<td>0.4199</td>
<td>0.4181</td>
</tr>
<tr>
<td>44</td>
<td>2.1376</td>
<td>2.1359</td>
<td>2.1366</td>
</tr>
<tr>
<td>50</td>
<td>5.2253</td>
<td>5.2214</td>
<td>5.2241</td>
</tr>
<tr>
<td>57</td>
<td>10.4714</td>
<td>10.4687</td>
<td>10.4706</td>
</tr>
<tr>
<td>60</td>
<td>13.0845</td>
<td>13.0828</td>
<td>13.0840</td>
</tr>
</tbody>
</table>

For \(K = 50, T = 1, r = 0.05, \sigma = 0.2, N = 200, M = 200\) and \(R = 100\).

Table 9: Comparison between different numerical schemes

<table>
<thead>
<tr>
<th>(S_0)</th>
<th>Black-Scholes</th>
<th>Implicit Scheme</th>
<th>Crank-Nicolson</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>0.2207</td>
<td>0.2227</td>
<td>0.2208</td>
</tr>
<tr>
<td>37</td>
<td>0.4181</td>
<td>0.4199</td>
<td>0.4181</td>
</tr>
<tr>
<td>44</td>
<td>2.1376</td>
<td>2.1359</td>
<td>2.1366</td>
</tr>
<tr>
<td>50</td>
<td>5.2253</td>
<td>5.2214</td>
<td>5.2241</td>
</tr>
<tr>
<td>57</td>
<td>10.4714</td>
<td>10.4687</td>
<td>10.4706</td>
</tr>
<tr>
<td>60</td>
<td>13.0845</td>
<td>13.0828</td>
<td>13.0840</td>
</tr>
</tbody>
</table>
Then, substitute these expressions into (7.2), one can get

$$\frac{\partial}{\partial t}u(t, x) = \frac{\partial^2}{\partial x^2}u(t, x) + (2\alpha + \frac{2r}{\sigma^2} - 1)\frac{\partial}{\partial x}u(t, x) + (\alpha^2 - \beta + \frac{2r}{\sigma^2} - 1)\alpha - \frac{2r}{\sigma^2}u(t, x).$$

In order to eliminate the last two terms and get the heat equation, set

$$\alpha = \frac{\sigma^2 - 2r}{2\sigma^2},$$

$$\beta = -\left(\frac{\sigma^2 + 2r}{2\sigma^2}\right)^2.$$  

Finally,

$$\frac{\partial}{\partial t}u(t, x) = \frac{\partial^2}{\partial x^2}u(t, x)$$  

(7.3)

for \(x \in \mathbb{R}\) and \(t \in [0, \frac{\sigma^2}{2}T]\), with initial condition

$$u(0, x) = e^{-\alpha x}g(\sigma^2), x \in \mathbb{R}.  \quad (7.4)$$

### 7.1 Consistency

According to [18], the definition of consistency is as follows.

**Definition 1.** Given a partial differential equation \(Pu = f\), and a finite difference equation \(P_{\tau,h}v = f\), we say that the finite difference scheme is consistent with the partial differential equation if for any smooth function \(\phi(t, x)\)

$$P\phi - P_{\tau,h}\phi \to 0 \quad \text{as} \quad \tau, h \to 0.$$  

Using a differential operator \(P = \frac{\partial}{\partial t} - \frac{\partial^2}{\partial x^2}\), (7.3) can be written as

$$P\phi = 0.  \quad (7.5)$$

At the point \((t_j, x_i)\), denote by \(\phi_{j,i} = \phi(t_j, x_i)\). The forward-time central-space scheme for the PDE given the discretized operator \(P_{\tau,h}\) can be expressed as

$$P_{\tau,h}\phi = \frac{\phi_{j+i,i} - \phi_{j,i}}{\tau} - \frac{\phi_{j,i+1} - 2\phi_{j,i} + \phi_{j,i-1}}{h^2}.$$  

Then consider the Taylor expansion of \(\phi\) in \(t\) and \(x\) at \((t_j, x_i)\),

$$\frac{\phi_{j+i,i} - \phi_{j,i}}{\tau} = \frac{\partial}{\partial t}\phi + O(t),$$

$$\frac{\phi_{j,i+1} - \phi_{j,i-1}}{2h} = \frac{\partial}{\partial x}\phi + O(h^2),$$

$$\frac{\phi_{j,i+1} - 2\phi_{j,i} + \phi_{j,i-1}}{h^2} = \frac{\partial^2}{\partial x^2}\phi + O(h^2).$$

Therefore, as \((\tau, h) \to 0,\)

$$Pv - P_{\tau,h}v = O(t) + O(h^2) \to 0.$$  

The explicit scheme is consistent. As for the implicit scheme, the only difference is that the first derivative with respect to \(t\) is approximated by backward difference, hence it is also
consistent. In addition, the consistency of Crank-Nicolson method can be proved by noticing that central difference is used for the approximation of the first derivative with respect to time, that is, at the point \((t_{j-\frac{1}{2}}, x_i)\),

\[
\frac{1}{\tau} (\phi_{j,i} - \phi_{j-1,i}) = \frac{\partial}{\partial t} \phi + O(t^2).
\]

and the term \(\frac{\partial^2}{\partial x^2} \phi\) at \((t_{j-\frac{1}{2}}, x_i)\) can be expressed as

\[
\frac{1}{2} \left( \frac{\phi_{j,i+1} - 2\phi_{j,i} + \phi_{j,i-1}}{h^2} + \frac{\phi_{j-1,i+1} - 2\phi_{j-1,i} + \phi_{j-1,i-1}}{h^2} \right) = \frac{\partial^2}{\partial x^2} \phi + O(h^2).
\]

Therefore, by the similar discussion above, the Crank-Nicolson method is consistent.

### 7.2 Stability

A numerical scheme is said to be stable with respect to a given norm, if the approximations obtained by the scheme are bounded in the norm whenever the step size goes to zero. Von Neumann analysis will be used here to check the stability of the explicit scheme, implicit scheme and the Crank-Nicolson method.

#### 7.2.1 Von Neumann analysis for the explicit scheme

The explicit finite difference scheme for (7.3) can be expressed as

\[
\frac{u(t_{j+1}, x_i) - u(t_j, x_i)}{\tau} = \frac{1}{h^2} \left( u(t_j, x_{i+1}) - 2u(t_j, x_i) + u(t_j, x_{i-1}) \right), \tag{7.6}
\]

where \(x_i = ih\) for \(i = 0, 1, \ldots, M\) and \(t_j = j\tau^*\) for \(j = 0, 1, \ldots, N, t_N = \frac{a^2}{\tau} T\). Denote by \(u_{j,i} = u(t_j, x_i)\), and assume that at the point \((t_j, x_i)\), the solution is of the form [11]:

\[
u_{j,i} = \xi_j e^{\sqrt{-k(ih)}}, \tag{7.7}
\]

where \(\xi\) shows the the time dependence of the solution and \(k\) is the wave number. Then, the expressions for \(u_{j+1,i}, u_{j,i+1}\) and \(u_{j,i-1}\) can be written as

\[
u_{j+1,i} = \xi_{j+1} e^{\sqrt{-k(ih)}}, \quad u_{j,i+1} = \xi_j e^{\sqrt{-k((i+1)h)}}, \quad u_{j,i-1} = \xi_j e^{\sqrt{-k((i-1)h)}}.
\]

Substitute these expression into (7.6), one can get

\[
\frac{\xi_{j+1}}{\xi_j} = 1 + \frac{\tau}{h^2} \left( e^{\sqrt{-k}h} - 2 + e^{-\sqrt{-k}h} \right).
\]

By the definition in [18], a finite difference scheme is stable if and only if the amplification factor satisfies the condition

\[
|G| \leq 1,
\]

where \(G\) can be expressed as

\[
G = \frac{\xi_{j+1}}{\xi_j}.
\]
In this case,

\[ |G| = \left| \frac{\xi_{j+1}}{\xi_j} \right| = \left| 1 + \frac{\tau}{h^2} (e^{\sqrt{-1}kh} - 2 + e^{-\sqrt{-1}kh}) \right| = \left| 1 - \frac{4\tau}{h^2} \sin^2 \left( \frac{kh}{2} \right) \right| \leq 1. \]

Then, in order to make the condition \(|G| \leq 1\) satisfied all the time, the step size \(\tau\) and the mesh size \(h\) should satisfy the following inequality:

\[ \frac{\tau}{h^2} \leq \frac{1}{2}. \]

Therefore, the explicit scheme is conditionally stable.

### 7.2.2 Von Neumann analysis for the implicit scheme

Apply the implicit scheme to (7.3), the partial differential equation can be discretized as

\[ u_{j,i} = u_{j+1,i} - \frac{\tau}{h^2} (u_{j+1,i+1} - 2u_{j+1,i} + u_{j+1,i-1}). \]

The solution at the point \((t_j, x_i)\) is of the form (7.7), then one may obtain the following equation:

\[ \xi_j e^{\sqrt{-1}kh(ih)} = \xi_{j+1} e^{\sqrt{-1}kh((i+1)h)} - \frac{\tau}{h^2} (\xi_{j+1} e^{\sqrt{-1}kh((i+1)h)} - 2\xi_{j+1} e^{\sqrt{-1}kh((i+1)h)} + \xi_{j+1} e^{\sqrt{-1}kh((i-1)h)}). \]

After simplification, it can be rewritten as

\[ \frac{\xi_{j+1}}{\xi_j} = \frac{1}{1 - \frac{\tau}{h^2} (e^{\sqrt{-1}kh} - 2 + e^{-\sqrt{-1}kh})} = \frac{1}{1 + \frac{4\tau}{h^2} \sin^2 \left( \frac{kh}{2} \right)}. \]

Then, by the stability condition on the amplification factor \(|G| \leq 1\), the implicit scheme is unconditionally stable since in all cases

\[ |G| = \left| \frac{1}{1 + \frac{4\tau}{h^2} \sin^2 \left( \frac{kh}{2} \right)} \right| \leq 1. \]

### 7.2.3 Von Neumann analysis for the Crank-Nicolson method

The numerical scheme of the Crank-Nicolson method for (7.3) is as follows:

\[ \frac{u_{j+1,i} - u_{j,i}}{\tau} = \frac{1}{2} \left( \frac{u_{j+1,i+1} - 2u_{j,i} + u_{j+1,i-1}}{h^2} + \frac{u_{j+1,i+1} - 2u_{j,i+1} + u_{j+1,i-1}}{h^2} \right). \]

With the solution expressed in (7.7), the above equation can be simplified to be

\[ \frac{\xi_{j+1}}{\xi_j} = \frac{1 - \frac{2\tau}{h^2} \sin^2 \left( \frac{kh}{2} \right)}{1 + \frac{2\tau}{h^2} \sin^2 \left( \frac{kh}{2} \right)}. \]

Since

\[ |G| = \left| \frac{1 - \frac{2\tau}{h^2} \sin^2 \left( \frac{kh}{2} \right)}{1 + \frac{2\tau}{h^2} \sin^2 \left( \frac{kh}{2} \right)} \right| \leq 1, \]

the stability condition on the amplification factor is satisfied, thus the Crank-Nicolson method is unconditionally stable.
7.3 Convergence

The fundamental theorem [18] in the analysis of finite difference schemes can be introduced.

**Theorem 4.** (The Lax-Richtmyer Equivalence Theorem) A consistent finite difference scheme for a partial differential equation for which the initial value problem is well-posed is convergent if and only if it is stable.

As discussed above, the implicit finite difference scheme and the Crank-Nicolson method are consistent and stable. The PDE problem (2.3)-(2.4) can also be regarded as a well-posed linear initial value problem. Then according to the theorem and by the following definition [18], one can determine the order of accuracy of these schemes.

**Definition 2.** A scheme \( P_{\tau,h}v = R_{\tau,h}f \) that is consistent with the differential equation \( Pu = f \) is accurate of order \( q \) in time and order \( p \) in space if for any smooth function \( \phi(t,x) \),

\[
P_{\tau,h}\phi - R_{\tau,h}P\phi = O(\tau^q) + O(h^p).
\]

We say that such a scheme is accurate of order \((p,q)\).

Note that \( R_{k,h} \) is an approximation of the identity operator. When the order of accuracy of a scheme is determined theoretically, one can verify it numerically. Denote by \( v = v(t,x) \) the exact solution of (2.3) and \( v_h \) the numerical approximation that depends on the spatial mesh size \( h \). By [14], the numerical scheme is of order \( p \) in space if

\[
|v_h - v| \leq Ch^p,
\]

where \( C \) is a number that is not depend on \( h \). Then, one can write the following equation:

\[
v_h - v = Ch^p + O(h^{p+1}). \quad (7.8)
\]

To verify the order of accuracy numerically in space, other approximated values for different mesh size \( h \) need to be calculated. For example, consider \( v_{h/2} \) and \( v_{h/4} \) which satisfy the equation (7.8), one can get the expressions:

\[
v_{h/2} - v = C(\frac{h}{2})^p + O((\frac{h}{2})^{p+1}),
\]

\[
v_{h/4} - v = C(\frac{h}{4})^p + O((\frac{h}{4})^{p+1}).
\]

Then,

\[
\frac{v_h - v_{h/2}}{v_{h/2} - v_{h/4}} = \frac{2^p + O(h)}{1 + O(h)},
\]

and by taking the logarithm of both sides,

\[
\log_2 \left| \frac{v_h - v_{h/2}}{v_{h/2} - v_{h/4}} \right| = p + O(h).
\]

Hence the scheme can be verified to be \( p \)th order accurate in space. Meanwhile, if \( v_\tau \) is denoted as the numerical solution depending on \( \tau \), the scheme is \( q \)th order accurate in time if

\[
|v_\tau - v| \leq C\tau^q.
\]

The order of accuracy for implicit scheme and Crank-Nicolson method will be determined theoretically, and then verified by numerical calculations.
### 7.3.1 The order of accuracy for the implicit scheme

By the definition 2, it can be determined that the implicit scheme is first order accurate in time and second order accurate in space. In order to verify the order of accuracy, first find the numerical approximations for different mesh sizes. Let $S_0 = 60, K = 50, T = 1, r = 0.05, \sigma = 0.2, R = 100$, the following table is the approximations calculated using implicit method and the order of accuracy in space.

| $h$  | $v_h$  | $v_h - v_{h/2}$ | $\frac{v_h - v_{h/2}}{v_{h/2} - v_{h/4}}$ | $\log_2 |\frac{v_h - v_{h/2}}{v_{h/2} - v_{h/4}}|$ |
|------|--------|-----------------|---------------------------------|----------------------------------|
| 2    | 13.0749| -0.0063         | 3.9853                          | 1.9947                           |
| 1    | 13.0812| -0.0016         | 3.9965                          | 1.9987                           |
| 0.5  | 13.0828| -0.0004         | 3.9991                          | 1.9997                           |
| 0.25 | 13.0832| -0.0001         |                                 |                                  |
| 0.125| 13.0833|                |                                 |                                  |

**Table 10:** The order of accuracy $p$ setting time step $\tau = 0.005$

From the last column of table 10, it can be verified numerically that the implicit scheme is of second order accuracy in space. Similarly, the order of accuracy in time can be verified using the equation,

$$\log_2 |\frac{v_r - v_{r/2}}{v_{r/2} - v_{r/4}}| = q + O(\tau).$$

It can be shown from the table 11 that the implicit finite difference scheme is first order accurate in time. Therefore, the implicit method is first-order accurate in time and second-order accurate in space.

| $\tau$ | $v_r$  | $v_r - v_{r/2}$ | $\frac{v_r - v_{r/2}}{v_{r/2} - v_{r/4}}$ | $\log_2 |\frac{v_r - v_{r/2}}{v_{r/2} - v_{r/4}}|$ |
|--------|--------|-----------------|---------------------------------|----------------------------------|
| 0.02   | 13.0794| -0.0023         | 1.9815                          | 0.9866                           |
| 0.01   | 13.0817| -0.0011         | 1.9907                          | 0.9933                           |
| 0.005  | 13.0828| -0.0006         | 1.9954                          | 0.9966                           |
| 0.0025 | 13.0834| -0.0003         |                                 |                                  |
| 0.00125| 13.0837|                |                                 |                                  |

**Table 11:** The order of accuracy $q$ setting mesh size $h = 0.5$

### 7.3.2 The order of accuracy for the Crank-Nicolson method

By considering the approximations of the derivatives at $(t_{j-\frac{1}{2}}, x_i)$, the order of accuracy of the Crank-Nicolson method should be of second order in time and space. One can again verify the order of accuracy by analyzing numerical approximations. Let $S_0 = 50, K = 50, T = 1, r = 0.05, \sigma = 0.2, R = 100$, the tables below shows approximated values for different $h$ and $\tau$, and the last column of the tables are the order of accuracy in space and time.
### Table 12: The order of accuracy \( p \) setting time step \( \tau = 0.005 \)

| \( \tau \) | \( v_{\tau} \) | \( v_{\tau} - v_{\tau/2} \) | \( \frac{v_{\tau} - v_{\tau/2}}{v_{\tau/2} - v_{\tau/4}} \) | \( \log_2 | \frac{v_{\tau} - v_{\tau/2}}{v_{\tau/2} - v_{\tau/4}} | \) |
|---|---|---|---|---|
| 0.02 | 5.2241 | 1.0e-04 *0.3722 | 3.9920 | 1.9971 |
| 0.01 | 5.2241 | 1.0e-04 *0.0932 | 3.9999 | 2.0000 |
| 0.005 | 5.2241 | 1.0e-04 *0.0233 | 4.0000 | 2.0000 |
| 0.0025 | 5.2241 | 1.0e-04 *0.0058 |
| 0.00125 | 5.2241 |

### Table 13: The order of accuracy \( q \) setting mesh size \( h = 0.5 \)

From the table 12 and 13, it can be tested numerically that the Crank-Nicolson method is a second order accurate scheme in space and time.
8 Apply the numerical scheme to other options

In previous sections, European call and put options are used to illustrate the implementation of the finite difference methods. In this section, the numerical method will be applied to the European type options with other payoff functions and the European-style barrier options.

8.1 European options with more general payoff functions

Assume that the stochastic process of the share price \((S_t)_{t \in [0,T]}\) is governed by the SDE
\[
dS_t = \mu S_t dt + \sigma S_t dW_t,
\]
where \(\mu \in \mathbb{R}, \sigma > 0\) and \(S_0\) is the initial share price. For a European type call with maturity \(T\), given the option payoff \(g\):
\[
g = \max(S_T^2 + 4S_T - 140, 0),
\]
then by the theorem 1 and after some calculations, the price of the option at time 0 can be obtained using the formula
\[
C_0 = e^{-rT}E_Q(S_T^2 1_{S_T > 10}) + 4e^{-rT}E_Q(S_T 1_{S_T > 10}) - 140e^{-rT}E_Q(1_{S_T > 10}).
\]
The term \(E_Q(S_T^2 1_{S_T > 10})\) can be further expressed as
\[
E_Q(S_T^2 1_{S_T > 10}) = S_0^2 e^{i\sigma^2 T} \Phi(d_i)
\]
for \(i = 0, 1, 2\), where \(\Phi(\cdot)\) is the standard normal distribution function and
\[
d_i = \frac{1}{\sigma \sqrt{T}} (\ln\left(\frac{S_0}{10}\right) + (r - \frac{1}{2}\sigma^2)T + i\sigma^2 T).
\]
Let \(S_0 = 10, K = 50, \sigma = 0.2, r = 0.05, M = 800, N = 800, R = 80\) and \(T = 1\), the result calculated using the analytical formula is 28.5072 while the approximation obtained by the Crank-Nicolson method is 28.5014. The graphs for the option prices can be found in the appendix 5. Therefore, to calculate the prices of European type option with complicated payoff functions, the finite difference methods can be employed to get numerical approximations.

8.2 European style barrier options

Consider a European style up-and-out put with strike price \(K\) and maturity \(T\), the barrier price is denoted as \(S_b\). If the Crank-Nicolson method is applied to approximate the prices of the barrier option, using the same notations as in section 6.4, one only needs to think carefully the settings of the boundary conditions. These boundary conditions can be written as
\[
v_{j,0} = v_R^{\tau,h}(t_j, x_0) = Ke^{-r(T-j\tau)}, \quad v_{j,M} = v_R^{\tau,h}(t_j, x_M) = 0.
\]
In this case, notice that \(x_M = S_b\). Then, let \(S_0 = 30, K = 40, S_b = 50, \sigma = 0.2, r = 0.05, M = 100, N = 100, R = 100\) and \(T = 1\), the approximation given by the Crank-Nicolson method is 8.43568 and the solution obtained using the formula is 8.43582. Meanwhile, the price for corresponding up-and-in put can be found using the formula,
\[
P = P_{UI} + P_{UO},
\]
where \(P\) is the option price for the European put option, \(P_{UI}\) is the option price for up-and-in put and \(P_{UO}\) is the price for up-and-out put.
9 Conclusion

The goal of the project is to approximate option prices numerically. For an option with underlying asset $S$, strike price $K$ and maturity $T$, first, the relationship between SDEs and PDEs is investigated so that the numerical methods can be employed to solve the PDE problems related to the option pricing. Then, the log transformation is applied to obtain the PDEs with constant coefficients. Next, the original PDEs are converted into discretized schemes. However, the discretization gives infinite systems of ODEs and equations. Thus, in order to be implemented numerically, the infinite systems need to be truncated and. The truncation error is discussed in section 4, and it vanishes as $R$ tends to infinity.

The spatial discretization for the PDEs is considered in section 5. Numerical approximations can be obtained by solving the finite system of ODEs using ode15 solver in MATLAB. Furthermore, the accuracy of the numerical results are tested for different $R$ and mesh size $h$. Despite the fact that these approximations are quite close to analytical solutions calculated using Black-Scholes formula, one should set suitably large $R$ for the scheme to work properly.

In section 6, fully discretized schemes are investigated. In order to fully discretize the PDEs, one needs to set up grids with respect to time and space. Explicit scheme, implicit scheme and Crank-Nicolson method are studied respectively as three examples of the fully discretized scheme to solve option pricing problems. For an explicit scheme, the time derivative is approximated by forward difference, and spatial derivative is approximated by central difference, while in an implicit scheme, the only difference is that backward difference is applied for the time derivative. As for the Crank-Nicolson method, it combines the explicit and implicit schemes and can be regarded as a central-time central-space scheme. Accuracy tests are then conducted for different $R$, mesh size $h$ and time step $\tau$. It could be noticed that the numerical approximations calculated using the Crank-Nicolson methods are of greater accuracy compared to the explicit and implicit methods.

Moreover, in section 7, consistency, stability and convergence are considered for explicit, implicit scheme and Crank-Nicolson method. It can be proved that these fully finite difference methods are all consistent. To test the stability property, Von Neumann analysis is used. By the stability condition on the amplification factor $G$, it can be shown that the explicit scheme is conditionally stable, whereas the implicit scheme and the Crank-Nicolson method are unconditionally stable. Then according to the theorem 4, the implicit scheme and the Crank-Nicolson method are also convergent. For a convergent scheme, one can determine its order of accuracy. The order of accuracy for the implicit scheme is first order accurate in time and second order accurate in space. This can be verified numerically. Similarly, the Crank-Nicolson method can be verified to be a second order accurate scheme in both space and time.

Finally, the finite difference method is applied to other European type options. The first example considered is a European call option with a different payoff function. The other one is a European style barrier option. The accurate numerical results indicate that the finite difference method is a practical alternative to find approximations for option prices.
References


   URL: http://www.math.umn.edu/~olver/num_/lnp.pdf


   URL: http://math.mit.edu/classes/18.311/Notes/NumSchemeSlab.pdf


differential equations.  
URL: http://people.maths.ox.ac.uk/trefethen/pdetext.html

Appendix

Figures

Figure 5: The graphs for the option in section 8.1

Figure 6: The graphs for the option in section 8.2
Codes

The Matlab code 3-8 are developed from the code in [2].

**Matlab code 1: Spatial finite difference scheme for European put option**

```matlab
S0 = 50;
r = 0.1;
K = 50;
sigma = 0.4;
Smax = 200; T=1;
M = 200;
dS = Smax/M; %mesh size h
vetS = linspace(0,Smax,M+1)';
veti = 0:M;
% initial condition
y_0_f = max(K-vetS,0);
y_0 = y_0_f(1:M);
%coefficient (for BS PDE without log transformation)
a = -(0.5*veti*r - 0.5*sigma^2*veti.^2);
b = -(r + sigma^2*veti.^2);
c = -(0.5*veti*r - 0.5*sigma^2*veti.^2);
%coefficient matrix
partA = diag(a(3:M),-1) + diag(b(2:M)) + diag(c(2:M-1),1);
% for boundary cond. at S = 0
first_row_A = zeros(1,M);
first_row_A(1) = -r;
first_col_A = zeros(M-1,1);
A_1 = [first_col_A, partA];
A = [first_row_A; A_1];
ode_rhs = @(t,x) A*x;
options = odeset('RelTol',1e-4,'AbsTol',1e-4);
[t_grid,Y] = ode15s(node_rhs,[0 T],y_0,options);
N=length(t_grid);
plot(vetS(1:M),Y(N,:),',r');
ylabel('','Interpreter','latex','String','v(\bar t,x)')
line(vetS(1:M),y_0);
interp1(vetS(1:M),Y(N,:),S0);
```

**Matlab code 2: Spatial finite difference scheme for European put under log transformation**

```matlab
logS0 = 2.4;
T=1;
r = 0.1;
sigma = 0.4;
K = 50;
logSmax = 5;
M = 200;
dS = logSmax/M; %mesh size h
vetS = linspace(0,logSmax,M+1)';
veti = 0:M;
% initial condition
y_f = max(K-exp(vetS),0);
y_0 = y_f(1:M);
%coefficient (for BS PDE under log transformation)
a = -((r-sigma^2/2)/dS - 0.5*sigma^2/dS^2);
b = -(r + sigma^2/dS^2);
```
c = -( -0.5*(r-sigma^2)/dS - 0.5*sigma^2/dS^2);
% coefficient matrix
partA = diag(a*ones(M-2,1),-1) + diag(b*ones(M-1,1)) + diag(c*ones(M-2,1),1);
% for boundary cond. at S = 0
first_row_A = zeros(1,M);
first_row_A(1) = -r;
first_col_A = zeros(M-1,1);
first_col_A(1) = a;
A_1 = [first_col_A, partA];
A = [first_row_A; A_1];

ode_rhs = @(t,x) A*x;
options = odeset('RelTol',1e-4,'AbsTol',1e-4);
[t_grid,Y] = ode15s(node_rhs,[0 T],y_0,options);
N=length(t_grid);
interpl(vetS(1:M),Y(N,:),logS0);
ylabel('','Interpreter','latex','String','p($\bar{t}$,x')
plot(vetS(1:M),Y(N,:),'r'); line(vetS(1:M),y_0);

Matlab code 3: Implicit scheme for European call option

S0 =30; % Spot price
K = 60; % Strike price
r = 0.05; % Risk free interest rate
sigma = 0.2; % Volatility
T = 1; % Maturity
M = 200; % Number of spatial grid points
N = 100; % Number of time grid points
Smax = 100;
dS = Smax/M;
dt = T/N;
solmatrix = zeros(N+1,M+1);
vetS = linspace(0,Smax,M+1);
vetT = linspace(0,T,N+1)';
veti = 0:M;
vetj = 0:N;
solmatrix(N+1,:) = max(vetS-K,0);
solmatrix(:,1) = 0;
solmatrix(:,M+1) = Smax-exp(-r*dt*(N-vetj))*(K);

a = 0.5*r*dt*veti - 0.5*sigma^2*dt*(veti.^2);
b = 1+sigma^2*dt*(veti.^2)+r*dt;
c = -0.5*(r*dt*veti+sigma^2*dt*(veti.^2));
coeff = diag(a(3:M),-1) + diag(b(2:M)) + diag(c(2:M-1),1);
[L,U] = lu(coeff);
aux = zeros(1,M-1);
for j=N:-1:1
    aux(M-1) = - c(M) * solmatrix(j,M+1);
solmatrix(j,2:M) = U \ (L \ (solmatrix(j+1,2:M) + aux'));
end
price = interpl(vetS, solmatrix(1,:), S0);
mesh(vetS,vetT,solmatrix); xlabel('x');ylabel('t');zlabel('v(t,x)');
plot(vetS,solmatrix(1,:), 'r'); line(vetS,solmatrix(N+1,:));
xlabel('x'); ylabel('v(t,x)');
Matlab code 4: Fully implicit scheme with log transformation

```matlab
1 logS0=3.6;
2 K = 35;
3 r = 0.05;
4 sigma = 0.2;
5 T = 1;
6 M = 200;
7 N = 800;
8 logSmax = 4.8;
9 dS = logSmax/M;
10 dt = T/N;
11 solmatrix = zeros(N+1,M+1);
12 vetS = linspace(0,logSmax,M+1);
13 vetT = linspace(0,T,N+1)';
14 veti = 0:M;
15 vetj = 0:N;
16 solmatrix(N+1,:) = max(exp(vetS)-K,0);
17 solmatrix(:,1) = 0;
18 solmatrix(:,M+1) = exp(logSmax)-exp(-r*dt*(N-veti))*K;
19 a = 0.5*dt*((r - 0.5*sigma^2)*dS - sigma^2) ./ (dS^2);
20 b = 1 + dt*(r + sigma^2 ./ (dS^2));
21 c = -0.5*dt*((r-0.5*sigma^2)*dS + sigma^2) ./ (dS^2);
22 coeff = diag(a*ones(M-2,1),-1) + diag(b*ones(M-1,1)) + diag(c*ones(M
23 -2,1),1));
24 [L,U] = lu(coeff);
25 aux = zeros(1,M-1);
26 for j=N:-1:1
27 aux(M-1) = - c * solmatrix(j,M+1);
28 solmatrix(j,2:M) = U \ (L \ (solmatrix(j+1,2:M) + aux))';
29 end
30
31 interp1(vetS, solmatrix(1,:), logS0);
32 mesh(vetS,vetT,solmatrix); xlabel('x');ylabel('t');zlabel('v(t,x)');
33 plot(vetS,solmatrix(1,:), 'r'); line(vetS,solmatrix(N+1,:));
34 xlabel('x'); ylabel('v(t,x)');
```

Matlab code 5: Explicit scheme for European call

```matlab
1 K = 40;
2 S0 = 50;
3 sigma = 0.2;
4 r = 0.05;
5 T = 1;
6 M = 50;
7 N = 100;
8 Smax = 100;
9 dS = Smax/M;
10 dt = T/N;
11 solmatrix = zeros(N+1,M+1);
12 vetS = linspace(0,Smax,M+1);
13 vetT = linspace(0,T,N+1)';
14 veti = 0:M;
15 vetj = 0:N;
16 solmatrix(N+1,:) = max(vetS-K,0);
17 solmatrix(:,1) = 0;
18 solmatrix(:,M+1) = (Smax-K)*exp(-r*dt*(N-veti));
```
\begin{verbatim}
a = (0.5*sigma^2*veti - 0.5*r).*veti*dt;
b = 1 - dt*sigma^2*veti.^2 - dt*r;
c = (0.5*sigma^2*veti + 0.5*r).*veti*dt;
for j=N:-1:1
    for i=2:M
        solmatrix(j,i) = a(i)*solmatrix(j+1,i-1) + b(i)*
                          solmatrix(j+1,i) + c(i)*solmatrix(j+1,i+1);
    end
end
price = interp1(vetS, solmatrix(1,:), S0);
\end{verbatim}

Matlab code 6: The Crank-Nicolson method for European call

\begin{verbatim}
K = 50;
S0 = 60;
sigma = 0.2;
r = 0.05;
T = 1;
M = 200;
N = 200;
Smax = 100;
dt = T/N;

dS = Smax/M;
solmatrix = zeros(N+1,M+1);
vetS = linspace(0,Smax,M+1);
vetT = linspace(0,T,N+1)';
veti = 0:M;
vetj = 0:N;
solmatrix(N+1,:) = max(vetS-K,0);
solmatrix(:,1) = 0;
solmatrix(:,M+1) = Smax - exp(-r*dt*(N-vetj))*K;
alpha = ( 0.25*sigma^2*(veti.^2) - 0.25*r*veti )*dt;
beta = -0.5*( r*dt + sigma^2*(veti.^2)*dt );
gamma = ( 0.25*sigma^2*(veti.^2) + 0.25*r*veti )*dt;
M1 = -diag(alpha(3:M),-1) + diag(1-beta(2:M)) - diag(gamma(2:M-1),1);
M2 = diag(alpha(3:M),-1) + diag(1+beta(2:M)) + diag(gamma(2:M-1),1);
[L,U] = lu(M1);
aux = zeros(1,M-1);
for j=N:-1:1
    aux(1) = alpha(2)*(solmatrix(j,1)+solmatrix(j+1,1));
aux(M-1) = gamma(M) * (solmatrix(j,M+1)+solmatrix(j+1,M+1));
solmatrix(j,2:M) = U \ (L \ ((M2*(solmatrix(j+1,2:M))' + aux')
                      ));
end
price = interp1(vetS, solmatrix(1,:), S0);
\end{verbatim}

Matlab code 7: The Crank-Nicolson method for European call with a more general payoff

\begin{verbatim}
S0 = 10;
K = 50;
r = 0.05;
sigma = 0.2;
T = 1;
M = 800;
N = 800;
Smax = 80;
dS = Smax/M;
dt = T/N;
\end{verbatim}
solution = zeros(N+1,M+1);
vetS = linspace(0,Smax,M+1);
vetT = linspace(0,T,N+1)';
veti = 0:M;
vetj = 0:N;
solution(N+1,:) = max((vetS.^2)+4*vetS-140,0);
solution(:,1) = 0;
solution(:,M+1) = (Smax^2)+4*Smax-140;
alpha = 0.25*( sigma^2*(veti.^2) - r*veti )*dt;
beta = -0.5*( r + sigma^2*(veti.^2)) * dt;
gamma = 0.25*( sigma^2*(veti.^2) + r*veti )*dt;
M1 = -diag(alpha(3:M),-1) + diag(1-beta(2:M)) - diag(gamma(2:M-1),1);
[L,U] = lu(M1);
M2 = diag(alpha(3:M),-1) + diag(1+beta(2:M)) + diag(gamma(2:M-1),1);
aux = zeros(1,M-1);
for j=N:-1:1
  aux(1) = alpha(2)*(solution(j,1)+solution(j+1,1));
  aux(M-1) = gamma(M) * (solution(j,M+1)+solution(j+1,M+1));
solution(j,2:M) = U \ (L \ (M2*(solution(j+1,2:M))' + aux'));
end
plot(vetS(1:200),solution(1,1:200),'r'); line(vetS(1:200),solution(N+1,1:200)); mesh(vetS(1:200),vetT,solution(:,1:200)); xlabel('S'); ylabel('t'); zlabel('V(t,S)');
interpl(vetS, solution(1,:),S0);

Matlab code 8: The Crank-Nicolson method for European style up-and-out put

S0 = 30;
K = 40;
r = 0.05;
sigma = 0.2;
T = 1;
M = 100;
N = 100;
Smax = 100;
Sb = 50;
dS = Sb/M;
dt = T/N;
solmatrix = zeros(N+1,M+1);
vetS = linspace(0,Sb,M+1);
vetT = linspace(0,T,N+1)';
veti = vetS/dS;
vetj = 0:N;
solmatrix(N+1,:) = max(K-vetS,0);
solmatrix(:,1) = K*exp(-r*dt*(N-vetj));
solmatrix(:,M+1) = 0;
alpha = 0.25*( sigma^2*(veti.^2) - r*veti )*dt;
beta = -( 0.5*r + 0.5*sigma^2*(veti.^2)) * dt;
gamma = 0.25*( sigma^2*(veti.^2) + r*veti )*dt;
M1 = -diag(alpha(3:M),-1) + diag(1-beta(2:M)) - diag(gamma(2:M-1),1);
[L,U] = lu(M1);
M2 = diag(alpha(3:M),-1) + diag(1+beta(2:M)) + diag(gamma(2:M-1),1);
aux = zeros(1,M-1);
for j=N:-1:1
  aux(1) = alpha(2)*(solmatrix(j,1)+solmatrix(j+1,1));
\[
\begin{align*}
\text{aux}(M-1) &= \gamma(M) \cdot (\text{solmatrix}(j,M+1) + \text{solmatrix}(j+1,M+1)) \\
\text{solmatrix}(j,2:M) &= U \backslash (L \backslash (M2*(\text{solmatrix}(j+1,2:M))' + \text{aux}') ) \\
\text{end}
\end{align*}
\]

\[
\begin{align*}
\text{mesh}(\text{vetS},\text{vetT},\text{solmatrix}); & \text{xlabel('S'); ylabel('t'); zlabel('V(t,S)');}
\plot(\text{vetS},\text{solmatrix}(1,:),'r');
\line(\text{vetS},\text{solmatrix}(N+1,:));
\text{interpl(\text{vetS}, \text{solmatrix}(1,:),S0);}
\end{align*}
\]

**Matlab code 9: Analytical formula for the option in section 8.1**

%analytical formula

\[
\begin{align*}
\text{function } &\text{ price } = \text{ parabolac}(S0,r,T,sigma) \\
&d0 = (\log(S0/10) + (r-0.5*sigma^2)*T)/(sigma*sqrt(T));
&d1 = (\log(S0/10) + (r-0.5*sigma^2)*T+(sigma^2)*T)/(sigma*sqrt(T));
&d2 = (\log(S0/10) + (r-0.5*sigma^2)*T+2*(sigma^2)*T)/(sigma*sqrt(T));
&A = (S0^2)*exp(2*r*T+(sigma^2)*T)*normcdf(d2);
&B = S0*exp(r*T)*normcdf(d1);
&C = normcdf(d0);
&price = exp(-r*T)*A + 4*exp(-r*T)*B - 140*exp(-r*T)*C;
\end{align*}
\]