Numerical Methods for Option Pricing: Binomial and Finite-difference Approximations

Jouni Kerman
Courant Institute of Mathematical Sciences
New York University

January 15, 2002

Abstract

Assuming the well-known model of lognormal asset price dynamics, we can value options using the Black-Scholes partial differential equation or the risk-neutral expectation formula. Solutions to these formulas can be approximated using finite-difference methods or the binomial method.

This thesis summarizes the main points of the arbitrage-free option pricing theory and the lognormal option pricing model. The lognormal model is derived from the points of view of both the continuous stochastic process model and the binomial model. The equivalence of these models is established analytically. Necessary formulas are derived.

Binomial and finite-difference numerical methods are introduced. Stability, accuracy, and convergence of the various numerical methods are discussed and analyzed.

A customized computer program is used to verify the analytical results, and to approximate solutions for pricing European and American options. This is illustrated by numerous graphs. Sample program code is included.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

Financial option valuation problems arise from the random nature of the underlying asset's price. Assuming the well-known model of lognormal asset price dynamics, we can value options using the Black-Scholes partial differential equation (PDE) or the risk-neutral expectation formula.

It is not always possible to find a closed form for the option value function, so numerical methods are necessary to find a solution for the value of the option. Solutions to these formulas can be approximated with numerical methods such as finite-difference methods or the binomial method.

This thesis introduces the finite-difference and binomial numerical methods for option pricing and compare their efficiency, accuracy, and stability. Testing is done using a customized computer program written in C++ .

In the first two chapters, the underlying concepts, theories, and assumptions are reviewed and necessary formulas are derived.

The following two chapters introduces the binomial and finite-difference methods and their application to European options. Convergence, accuracy, and stability issues are discussed and demonstrated.

The fifth chapter introduces the applications of the numerical methods to American options. American options do not have explicit formulas, so numerical methods are indispensable for pricing them.

The sixth chapter concludes and summarizes the main points of the thesis. Finally, the appendix contains selected parts of the C++ program.

1.2 Options

An *option* is a tradable financial security whose value depends on the value of an underlying asset.

As its name implies, an option is a contract which gives its holder a *right* to *exercise* it, i.e. to realize the claim the option exists for. Usually this involves converting the option to cash at the expense of the counterparty who issued the option. This is the *writer* of an option, who charges a fee for the risk of incurring possible loss.

Since an option is not an obligation, its holder can also choose not to exercise it. In the world of rational traders and investors, options are exercised only when their value is greater than zero—then it is said that the option is *in-the-money.*

Options are characterized typically by the following properties.

(1) An *underlying asset.* The price of an option depends on the price of an underlying asset, which is (typically) random. Hence an option is a *derivative security*, i.e. a security the value of which is derived from the price of the underlying security. Derivatives are also called *contingent claim*s.

(2) The time to *expiry.* Practically all options have a limited lifespan. They become worthless after they have expired. Thus, one does not usually speak of “investing” in options, but rather of “trading” options.

(3) *payoff function* which describes the value of the option as a function of the underlying asset at the time of expiry and possibly also at specified times before expiry. This can be any function which maps all possible underlying asset prices to some monetary value. A payoff function need not be differentiable nor even continuous.

(4) Some options have limitations on when they can be exercised. The most typical option is a *European option*, which can only be exercised at expiry. An *American option*, once written, can be exercised at any time before expiry. There is a variety of other kinds of options, for example *Bermudan option* can be exercised on only at dates specified in advance. The usage of these names are mere conventions and do not have any bearing on where they are traded in or where they are originated from.

The most basic options are the *call option* and the *put option.* These are often referred to as (plain) ‘vanilla’ options. A call option conveys the right to *buy* the underlying asset at a pre-specified price called the *strike price*; a put option is a right to *sell* the underlying asset at the strike price.

The payoff function for a put option depends on the price of the under-

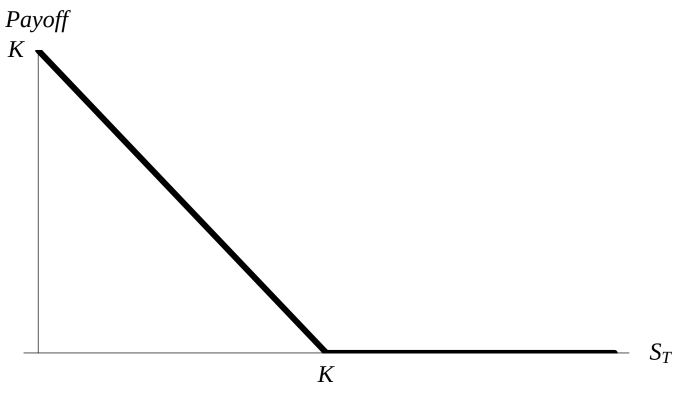


Figure 1.1: Payoff function of the put option.

lying asset (e.g. a stock) at expiry T and the strike price K. Call this future price S_T. Then the put payoff $P(S_T)$ can be expressed as [1]

$$P_E(S_T, T) = (K - S_T)_+ \tag{1.1}$$

and the European call option payoff is correspondingly

$$C_E(S_T, T) = (S_T - K)_+. \tag{1.2}$$

$C_E(S_t, t)$ and $P_E(S_t, t)$ are the prices of European call and European put options at time t, respectively.

Another class of very simple options is the *binary option.* A binary option pays off a fixed amount only if the asset price is above or below certain strike price K. The *binary put option* has a payoff

$$P_B(S_T) = \begin{cases} 1 & \text{if} \quad S_T < K \\ 0 & \text{if} \quad S_T \geq K \end{cases} \tag{1.3}$$

and the *binary call option* has a payoff

$$C_B(S_T) = \begin{cases} 0 & \text{if} \quad S_T \leq K \\ 1 & \text{if} \quad S_T > K \end{cases} \tag{1.4}$$

The binary options are examples of options with a discontinuous payoff.

[1]The notation $f(x)_+$ is equivalent to $\max\{f(x), 0\}$.

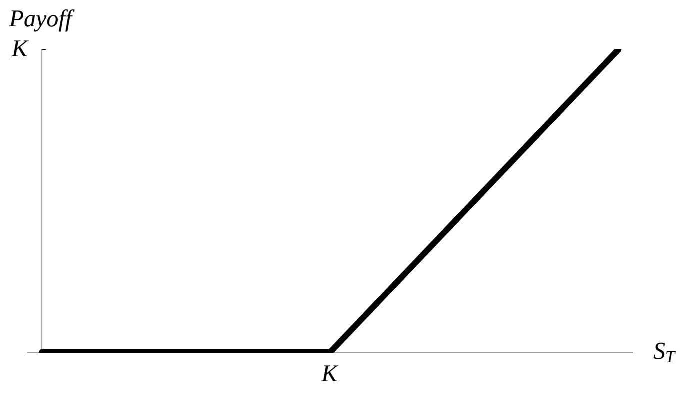


Figure 1.2: Payoff function of the call option.

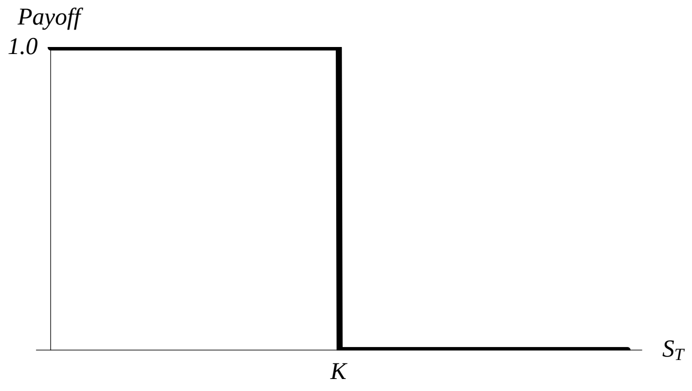


Figure 1.3: Payoff function of the binary put option.

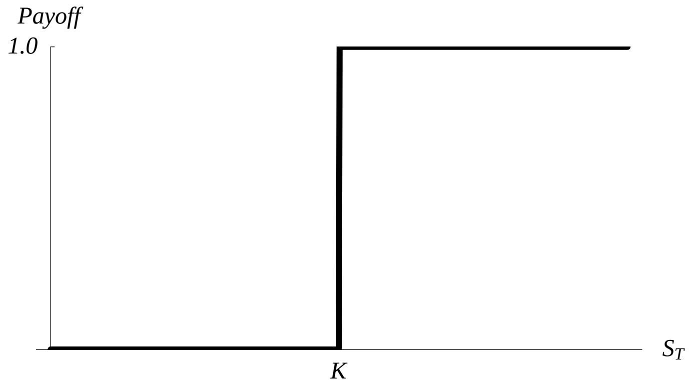


Figure 1.4: Payoff function of the binary call option.

1.2.1 Arbitrage-free pricing

A key concept in the theory of option pricing is the assumption of arbitrage-free market. The absence of *arbitrage* means that no player in the market is able to make a *riskless profit* by selling and buying securities. This assumption leads to finding the *fair price* for an option; any price lower or higher this price would lead to arbitrage.

The same principle should of course apply to any collection of securities, which we call a *portfolio*. A portfolio may contain long and short positions on assets. A *position* is an item in the portfolio, such as a stock or cash in a money market account. A *long position*, when liquidated, creates a positive cash flow, and is thus an asset to us. A stock or an option that we have purchased, or a Treasury bond are examples of long positions. A *short position* creates a potential negative cash flow when liquidated, so it is a liability. Examples of short positions are a borrowed stock or a written option.

To apply the principle of no arbitrage, one must assume the existence of a *risk-free security*, a default-free asset. In this thesis we assume that there is a risk-free security that has a constant continuously compounded yield r. The traders and investors are assumed to be able to borrow *and* lend at this rate.

Suppose we have two riskless portfolios A and B with respective yields a and b over some period of time T based on their current prices A_0, B_0. Suppose further that $A_0 = B_0$ while $a > b$. Now consider a portfolio C with a long position on portfolio A and a short position on portfolio B. Its present value C_0 is

$$C_0 = A_0 - B_0 = 0,$$

while at time T the value of portfolio C is

$$C_T = A_T - B_T = A_0 e^{aT} - B_0 e^{bT} = A_0 e^{bT}[e^{(a-b)T} - 1] > 0,$$

since $A_0 = B_0$ and $e^{(a-b)T} > 1$. Forming portfolio C does not cost anything and hence involves no risk. Nevertheless, it has a positive payoff. We have thus an *arbitrage opportunity*: a possibility to make riskless profit by assuming a long position on this portfolio. Indeed, we have an *unlimited* interest to buy these portfolios at this price.

Arbitrage is thus an extremely attractive proposition. However, the assumption of the *absence* of arbitrage is not as naïve as it sounds, although there are certainly *arbitrageurs* in the market looking for 'mispriced' securities.

Should an asset be mispriced, a deviation from its arbitrage-free price would quickly be exploited in the market. High demand or supply for the mispriced portfolio would cause its price to rise or to fall, eventually settling at the arbitrage-free price and bringing the market to equilibrium. If portfolio C would be available in the market indefinitely at price C_0, buying this portfolio would inflate the profits of the arbitrageur without bound. Since this is essentially a zero-sum game, eventually all the other traders and investors selling portfolio A and buying portfolio B would drain out of cash.

In the above example, excess demand of the high-yielding portfolio A and excess supply of the low-yielding portfolio B would drive the price of portfolio A up and drag the price of portfolio B down.

If instead we assume $a < b$ while holding the other parameters the same, a portfolio C' with a long position on portfolio B and a short position on portfolio A will also yield a riskless profit. In this case, the price of portfolio B would go up and the price of portfolio A would go down.

As the prices of the portfolios change, so do their yields. The equilibrium price is obtained when $a = b$. This argument shows that the yields of any two riskless portfolios must be equivalent. This means that in this market $r = a = b$.

Binary put-call parity. The binary puts and calls have a simple relationship. Since a portfolio consisting of one binary put and one binary call yields 1 regardless of the price of the underlying S_T at expiry, the portfolio is riskless. By the no-arbitrage argument, the portfolio must grow at the risk-free rate r. Thus the present value of the combination portfolio is e^{-rT}:

$$P_B + C_B = e^{-rT}, \tag{1.5}$$

where T is the time remaining until expiry.

Forward contract. Consider a contingent claim called a *forward contract*, which is an obligation to enter into a trade of an underlying asset at a pre-specified point of time T in the future at a pre-specified price K, called the *delivery price*. It is said that T is the time of *maturity* of the forward contract. This is not an option since both sides of the contract are required to trade. If the price of the asset (say, a stock) at time T is S_T, the value of the contract to the party holding the long position is $S_T - K$ at time T. The other party is obliged to sell the asset at K; the value for the selling party is then $K - S_T$.

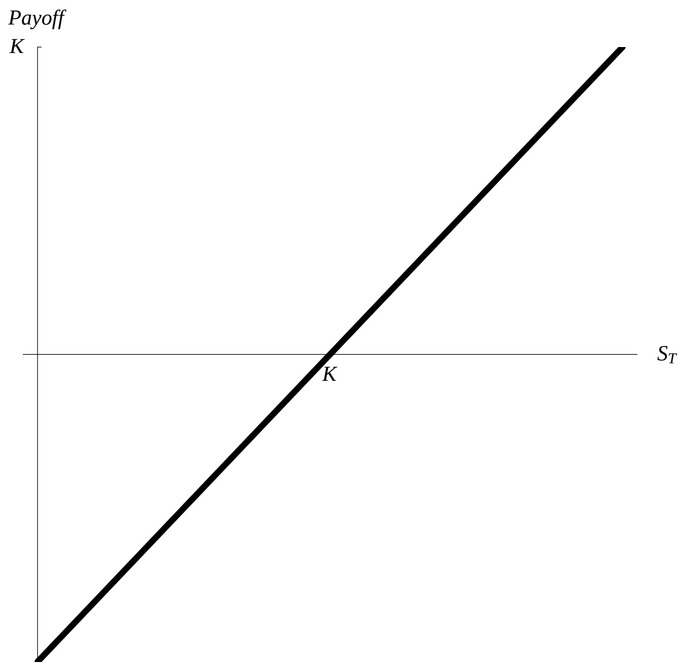


Figure 1.5: Payoff function of a forward contract. By the put-call parity, a forward contract is equal to a short put position and a long call position with the same strike and expiry.

What is the fair, arbitrage-free price of this asset at time $t = 0$? Consider two portfolios. Portfolio A contains a long position on the forward contract. Portfolio B contains the asset and an amount of cash Ke^{-rT} invested at the current risk-free rate r. At time T, portfolio A has the value

$$S_T - K,$$

while portfolio B *also* has the value $S_T - K$, since the cash has grown to $(Ke^{-rT})e^{rT} = K$. Since the present value of portfolio B is

$$S_0 - Ke^{-rT},$$

this must also be the present value for portfolio A, namely the present value of the forward contract. To see why this must be so, suppose that the present value of the forward contract is F_0 such that $F_0 > S_0 - Ke^{-rT}$ so that going long on the forward contract and going short on portfolio B would yield an immediate positive cash flow $F_0 - (S_0 - Ke^{rT})$ while at time T their payoffs cancel. If $F_0 < S_0 - Ke^{-rT}$, we should take a reversed market position. Again, $F_0 = S_0 - Ke^{-rT}$ is the only fair arbitrage-free value for the contract at time t.

A forward contract is defined to have an initial value zero. This requirement will force the delivery price K to be exactly $K = S_0 e^{rT}$.

A portfolio C=A−B must therefore be worth zero in an arbitrage-free market. Since the position on portfolio B effectively eliminates all randomness from the long forward contract A, it is called a *hedge portfolio*. It is also said to *replicate* the forward contract. Hence, to price a contingent claim, we must find a replicating portfolio that imitates perfectly the random movements of the contingent claim.

In the case of a forward contract, finding the fair price was easy. The fact that many options have payoffs that are nonnegative (calls and puts) and even discontinuous (binary options) complicates matters so much that it is not any more possible to find a replicating portfolio once and for all. As we will see later, this may require continuous adjusting the replicating portfolio.

Put-call parity. Consider again two portfolios: portfolio A is a long forward contract with delivery price K at time T, and portfolio B with a long position on a European call option with strike K and expiry T, a short position on a European put option with the same parameters. The value of portfolio B at expiry T is

$$(S_T - K)_+ - (K - S_T)_+ = S_T - K, \tag{1.6}$$

which matches exactly the payoff of portfolio A. Thus a long position on a forward contract, or alternatively a short position on a (riskless) cash position Ke^{-rT} and a long position on a stock replicates the portfolio of a long call and a short put. In the absence of arbitrage, the yields of these two portfolios must be the same; since they are both riskless, the yield must equal to that of the riskless security, namely r. Since T was arbitrary, this must hold for any T, and so for at any time t before maturity:

$$C_E(S_t, t) - P_E(S_t, t) = S_t - Ke^{-r(T-t)}. \tag{1.7}$$

1.3 Numerical methods

This thesis presents the two most popular nonanalytic, numerical methods for pricing options.

The *binomial method* attempts to create a binomial lattice of all possible price paths of the underlying asset, and then in essence to compute an expectation value that is shown to be the fair price of the option.

The *finite-difference method* tackles the *Black-Scholes partial differential equation* directly by breaking the space-time plane into small discrete differences and then deriving the solution backward, one timestep at a time.

Some options *can* be priced using explicit formulas, but others require numerical methods. Numerical methods are essential in trying to value American options, for most of which analytical formulas are unavailable.

The computational methods have always a trade-off between speed and accuracy, i.e. between maximizing speed and minimizing errors. To gain a better accuracy, one must incur a cost of lengthier computing time. Fortunately, some methods converge to the correct solution faster than other methods, so a desired accuracy can be obtained with fewer computer operations.

Chapter 2

The Option Pricing Models

2.1 The Black-Scholes Option Pricing Model

Fischer Black and Myron Scholes showed in their 1973 paper [1] that it was possible to set an unambiguous price for an option which price depended on the (random) price of a traded security or asset. Even if the traders had different opinions about the future value of the asset, they had to agree on the price of the option written on this asset. The key point was that, given a set of assumptions, there was only one price that would guarantee no arbitrage opportunities.

In this section we will review the theory and derive the Black-Scholes partial differential equation governing the price of the option.

2.1.1 Lognormal Dynamics

Consider a European option with expiry at time T. We are only concerned of the time remaining to the expiry of the option, so without loss of generality, we can assume that the present time is zero.

The value of the option depends both on the current value of the underlying asset's price and the time to expiry T. It is clear that the asset price is a random variable. Let us denote the price of the asset at some point of time t by S_t. To find the value of the option, we must first study how the asset price S_t is dependent on time.

The *spot price* S_0 is the current (fully known) price of the asset. In the Black-Scholes analysis, the actual absolute level S_t of the stock price in itself is irrelevant. What matters is its *relative change* during some time interval.

Since absolute time is irrelevant, again we may consider without loss of generality the change of the asset value only between the current time $t = 0$ and at some time t; thus, we want to know how the ratio S_t/S_0 is distributed.

The key assumption in the Black-Scholes option pricing model is that the relative change of the asset over a period of time is normally distributed. Given the spot price S_0, the *rate of return* (or, more simply, *return*) over some time interval t is

$$\frac{S_t - S_0}{S_0}$$

is normally distributed. According to the Black-Scholes model, we moreover assume that (1) the returns are identically distributed; (2) the returns are independently distributed; (3) the mean and the variance are $(\mu t, \sigma^2 t)$, respectively.

The first assumption implies that the distributions will remain the same. The second assumption tells us that the probability distributions are not affected by previous events: "the market has no memory." The third assumption means that as time passes, the return drifts steadily at a rate of μ away from their current mean. In time, the variance also grows, making the probability distribution more spread out.

It follows that disjoint time intervals are independent of each other; thus the rate of return calculated from time point t_1 to t_2 is independent from that of t_2 to t_3.

The rate of return can be then expressed as

$$\frac{S_t - S_0}{S_0} = \mu t + \sigma \sqrt{t}\, Z \tag{2.1}$$

where Z is a standard normal random variable with mean 0 and variance 1. The classical Black-Scholes theory assumes μ and σ are constants; in practice this is unlikely to be the case for long time periods.

The equation (2.1) tells us that as time passes by an amount of t, the asset price changes by μt, and also jumps up or down by a random amount $\sigma\sqrt{t}Z$.

Since there is a random change every interval of arbitrary length t, then there are actually several random variables involved over any time period. Thus it is preferable to call the sequence of random variables over time a *random process*. In this case the process is often called a *random walk*. We prefer to denote the random component of equation (2.1) by a single variable; let us write $W_t = \sqrt{t}Z$.

Consider now what happens if we make the time intervals smaller and smaller. In the limit $t \to 0$, the random process becomes a continuous random process; we call this a *stochastic process* and may use differentials to describe infinitesimal changes. We may write the equation as

$$\frac{\mathrm{d}S_t}{S_0} = \mu \,\mathrm{d}t + \sigma \,\mathrm{d}W_t \tag{2.2}$$

where W_t (and, consequently S_t) is a *stochastic variable.* Passing to the limit $t \to 0$ involves the assumption that the asset is traded continuously.

Denote the right-hand side of (2.2) by

$$\mathrm{d}X_t = \mu \,\mathrm{d}t + \sigma \,\mathrm{d}W_t \tag{2.3}$$

The variable μ is called the *drift rate.*

Using the fact that $\mathrm{d}S_t/S_0 = \mathrm{d}(\log S_t)$, we can write S_t as

$$S_t = S_0 e^{X_t}. \tag{2.4}$$

This means that the logarithm of S_t is normally distributed; hence we say that the distribution of S_t is *lognormal.*

Brownian motion. The variable $\mathrm{d}W_t$ appearing in the *stochastic differential equation* (2.2) describes a special kind of a random walk, called *Brownian motion*, or *Wiener process.* It is characterized by W_t being a normally distributed random variable having mean 0 and variance t.

S_t is continuous, but nowhere differentiable. This can be seen formally by dividing [1] δS_t by δt and letting $\delta t \to 0$:

$$\lim_{\delta t \to 0} \delta S_t/\delta t = \lim_{\delta t \to 0} O(\sqrt{\delta t})/\delta t = \lim_{\delta t \to 0} O(1/\sqrt{\delta t}) = \infty,$$

so the differential does not exist.

Equation 2.2 tells us that the logarithm of S_t/S_0 is a "Brownian motion with drift" $\mu \,\mathrm{d}t$, and gives us a way to describe the changes $\mathrm{d}S_t$ in the asset price S_t as time passes.

We now consider a function V depending of the asset price S_t and time t and see how it changes during the infinitesimal timestep $\mathrm{d}t$.

[1]The symbol δ is used here as a linear operator. It denotes a small but not infinitesimally small change.

Itô's formula. If the asset price S were a deterministic variable we would simply expand $V(S_0+\delta S, \delta t)$ at $V(S,0)$ in Taylor series:

$$\delta V = \left(\frac{\partial V}{\partial t}\delta t + \tfrac{1}{2}\frac{\partial^2 V}{\partial t^2}\delta t^2 + \cdots\right) + \left(\frac{\partial V}{\partial S}\delta S + \tfrac{1}{2}\frac{\partial^2 V}{\partial S^2}\delta S^2 + \cdots\right) + \cdots$$

The *Itô calculus* is the stochastic process equivalent of Newtonian differential calculus. In the limit $\delta t \to 0$, terms of δt of higher order than 1, as in the ordinary differential calculus are negligibly small and can be omitted.

The difference in the case of the random process, is that also δS_t depends on δt. In the case of lognormal random walk, equation (2.1) gives an expression for $\delta S = S_0\mu\delta t + \sigma\sqrt{\delta t}S_0 Z$. Consider then

$$\delta S^2 = (S_0\mu\delta t + \sigma\sqrt{\delta t}S_0 Z)^2.$$

The only term that contributes to a term of at most order $O(\delta t)$ is $(\sigma\sqrt{\delta t}S_0 Z)^2$. Since Z is standard normal, Z^2 is distributed with a gamma distribution with mean 1. Therefore,

$$\mathbf{E}[(\sigma\sqrt{\delta t}Z - \sigma^2 S_0\delta t)^2] = \sigma^2 S_0\delta t\mathbf{E}[Z^2] + O(t^{3/2}) = \sigma^2 S_0\delta t + O(t^{3/2}).$$

In the limit $\delta t \to 0$,

$$\mathrm{d}S_t^2 = \sigma^2 S_0^2\,\mathrm{d}t.$$

Therefore, if V is a function dependent on S_t, it is also a stochastic process (V_t) such that

$$\mathrm{d}V_t = \frac{\partial V}{\partial t}\,\mathrm{d}t + \frac{\partial V}{\partial S}\,\mathrm{d}S_t + \tfrac{1}{2}\sigma^2 S_0^2\frac{\partial^2 V}{\partial S^2}\,\mathrm{d}t \tag{2.5}$$

or, writing out $\mathrm{d}S_t$, we obtain *Itô's formula* for the option, given the underlying asset is a stochastic process with $\mathrm{d}S_t = \mu S_0\,\mathrm{d}t + \sigma S_0\,\mathrm{d}W$:

$$\mathrm{d}V_t = \frac{\partial V}{\partial S}\sigma\,\mathrm{d}W_t + \left(\frac{\partial V}{\partial t} + \mu S_0\frac{\partial V}{\partial S} + \tfrac{1}{2}\sigma^2 S_0^2\frac{\partial^2 V}{\partial S^2}\right)\mathrm{d}t \tag{2.6}$$

The stochastic variable $\mathrm{d}W_t$ is present in the formula; this means the option price $V(S_t,t)$ also moves randomly. This is what we expected. In the previous chapter, we learned that if we can eliminate the randomness by a replicating (hedging) portfolio, we can find the price for the option.

By definition, a replicating portfolio eliminates the randomness of the option, and in essence, making the option equivalent to a riskless portfolio. Once we have such a portfolio, its yield must match that of a riskless portfolio or else an arbitrage opportunity will occur. We must thus assume the absence of arbitrage to determine the fair price of the option.

It is now appropriate to define the concept of 'risk' of a portfolio. A popular definition of the risk of a portfolio is the *variance of the return* on the investment [2]. In our lognormal model, this is σ^2. σ is also called the *volatility* of the portfolio. A riskless portfolio thus has $\sigma = 0$ and $\mu = r$, i.e. a drift of rt and no random component.

Finding a way to eliminate the random component in equation (2.5) or (2.6) governing the price of an option is thus the key to finding the 'fair price' for the option. The result will be the Black-Scholes equation, which must hold in the absence of arbitrage.

2.1.2 The Black-Scholes Equation

Assume now that equation (2.6) holds so we have a stochastic process V_t depending on another process S_t. Construct a portfolio consisting of one option and a short position of Δ_0 units on the stock. Δ_0 is a real number, not necessarily an integer. The value of this portfolio is initially

$$\Pi_0 = V_0 - \Delta_0 S_0.$$

With a suitable Δ_0, the value of this portfolio will match that of the replicated option. After one timestep $\mathrm{d}t$, the portfolio will have changed by

$$\mathrm{d}\Pi_t = \mathrm{d}V_t - \Delta_0\,\mathrm{d}S_t. \tag{2.7}$$

Note here Δ_0 is assumed to be constant during the timestep $\mathrm{d}t$. Applying equation 2.5,

$$\mathrm{d}\Pi_t = \frac{\partial V}{\partial t}\,\mathrm{d}t + (\frac{\partial V}{\partial S} - \Delta_0)\,\mathrm{d}S_t + \tfrac{1}{2}\sigma^2 S_0^2 \frac{\partial^2 V}{\partial S^2}\,\mathrm{d}t$$

It is now obvious that we can eliminate the random component $\mathrm{d}S_t$ if we choose $\Delta_0 = \partial V/\partial S$, the rate of change of the option with relation to its underlying calculated at $t = 0$. $\mathrm{d}\Pi_t$ becomes thus fully deterministic:

$$\mathrm{d}\Pi_t = \left(\frac{\partial V}{\partial t} + \tfrac{1}{2}\sigma^2 S_0^2 \frac{\partial^2 V}{\partial S^2}\right)\mathrm{d}t.$$

This portfolio does not have any random component in it, so it is deterministic and thus riskless. By the no-arbitrage argument the yield on this portfolio must be the same as that of a riskless security.

If we assume now that the yield of a riskless security, is r, a portfolio valued Π_0 at $t = 0$ invested in this security yields $r\Pi_0 \, \mathrm{d}t$ during timestep $\mathrm{d}t$.

The deterministic portfolio (replicated option), on the other hand, yields $\mathrm{d}\Pi_t$. Their yields must be equal, so $r\Pi_0 \, \mathrm{d}t = \mathrm{d}\Pi_t$, and

$$r\Pi_0 \, \mathrm{d}t = \left(\frac{\partial V}{\partial t} + \tfrac{1}{2}\sigma^2 S_0^2 \frac{\partial^2 V}{\partial S^2} \right) \mathrm{d}t.$$

This means that since we know the derivative $\partial V / \partial S$ at $t = 0$ so we also know how fast the portfolio Π is growing. Since $\Pi_0 = V_0 - \Delta_0 S_0 = V_0 - S_0 \frac{\partial V}{\partial S}$,

$$r\left(V_0 - S_0 \frac{\partial V}{\partial S} \right) = \left(\frac{\partial V}{\partial t} + \tfrac{1}{2}\sigma^2 S_0^2 \frac{\partial^2 V}{\partial S^2} \right).$$

This is the Black-Scholes partial differential equation, which must hold for all European options (which are to be held until expiry).

Since the point of time $t = 0$ was arbitrary, and all S in the above equation refer to S_0, we can drop the subscript 0. The equation is usually written

$$\frac{\partial V}{\partial t} + \tfrac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS \frac{\partial V}{\partial S} - rV = 0. \tag{2.8}$$

The drift rate μ is not present in the equation. This means that the only parameter in the option value equation that is not directly observable is the *volatility* term σ. The no-arbitrage requirement dictates that the drift rate of the underlying asset be exactly r, i.e. that of the riskless asset.

Eliminating the random movements of a portfolio is called *delta hedging.* Such a portfolio is thus risk-free and in an arbitrage-free market one can expect that its yield equals to that of the other risk-free securities.

However, setting $\Delta_0 = \partial V / \partial S$ implies that Δ_0, i.e. the number (or, fraction) of shares and riskless security position kept in the portfolio must be readjusted (hedged) continuously, since $\partial V / \partial S$ varies in time. This is, of course, unrealistic in practice and even more so if we consider transaction costs.

In conclusion, the price of a derivative security is governed by equation (2.8) given that the underlying asset follows lognormal dynamics in the absence of arbitrage.

2.1.3 Finding a solution

The Black-Scholes partial differential equation (2.8) must be true for *any* option assuming the underlying asset S_t is a stochastic process with drift μ and volatility σ, given riskless rate r.

A PDE such as (2.8) has a unique solution only if we can impose a *final condition* and *boundary conditions* [2]. These conditions are very important in implementing the finite-difference numerical methods.

The Black-Scholes equation can only be solved backward in time. It is obvious from the no-arbitrage argument that the option cannot have any other value at $t = T$ than its payoff. Given the payoff $V(S_T, T)$ of the option, we can possibly only recover the values for $V(S_t, t)$ for $t < T$. But this is what we need; it does not even make sense to try to find values for $t > T$, since the option expires at $t = T$ and will be worthless.

Equation of type (2.8), with a first partial derivative with relation to the first variable and a second partial derivative with relation to the second variable is called a *backward parabolic equation.* When written on the same side of the equation, a backward parabolic equation has same algebraic signs for both of the first and second partial derivatives.

A *forward parabolic equation* appears later in this thesis when we (mainly for convenience) reverse the flow of time so that the final condition becomes an *initial condition.* This can be done by a simple variable change ($t' = -t$). Both problems are well-posed, if a backward equation is solved backward in time starting from a final condition and if a forward equation is solved forward in time starting from an initial condition.

The PDE (2.8) requires that we apply two conditions to the variable [2] S, which has the second partial derivative $\partial^2 V/\partial S^2$, and one condition to the variable t, which has only the first partial derivative $\partial V/\partial t$. The condition with relation to t is the payoff condition. The (boundary) conditions with relation to S must specify the values for V in the boundaries of the domain of V. Since S is defined for $S \in [0, \infty)$, the boundary conditions must be specified for $S = 0$ and for the limit $S \to \infty$ for each $t < T$. This condition depends on the type of option.

For example, for a European call option $C(S, t)$ with strike K the payoff

[2] Whenever we want to emphasize the fact that the asset price is a stochastic random variable, we write S_t. When it is clear that S is continuous variable, we will drop the subscript and write S. In this text, V_t denotes the stochastic process of the option or its value at time t.

is $C(S,T) = (S-K)_+$, so the boundary conditions are

$$\begin{cases} \quad C(0,t) = 0 & \forall\, t < T \\ \lim\limits_{S\to\infty} C(S,t) = S & \forall\, t < T \end{cases}$$

For a European put, the payoff is $P(S,T) = (K-S)_+$ and the boundary conditions become

$$\begin{cases} \quad P(0,t) = K & \forall\, t < T \\ \lim\limits_{S\to\infty} C(S,t) = 0 & \forall\, t < T \end{cases}$$

2.1.4 Constant dividend yield

Consider the case when the underlying asset pays a continuous dividend at some fixed rate D. Without loss of generality, let $t = 0$ and the spot price of the asset be S_0. After an infinitesimal timestep $\mathrm{d}t$, the holder of the asset gains $DS_0\,\mathrm{d}t$ in dividends. However, the asset price must fall by the same amount or else there is an arbitrage opportunity: buying the asset at $t = 0$ at S_0 and selling it immediately at $S_0 + \mathrm{d}S_t\,\mathrm{d}t$ after receiving the dividend would yield a risk-free profit of $DS\,\mathrm{d}t$. Thus we must have

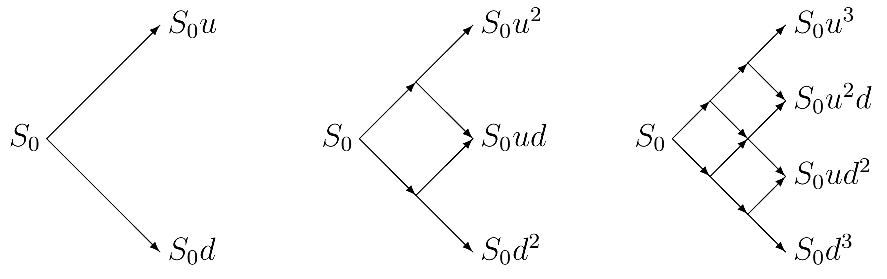
$$\mathrm{d}S_t = \sigma S_0\,\mathrm{d}W_t + \mu S_0\,\mathrm{d}t - DS_0\,\mathrm{d}t. \tag{2.9}$$

Since the holder of the option does not receive the dividend, the payment must not change the price of the option, so we must have a correction term for (2.7)

$$\mathrm{d}\Pi_t = \mathrm{d}V_t - \Delta_0\,\mathrm{d}S_t - D\Delta_0 S_0\,\mathrm{d}t$$

to eliminate the effect of the dividend in the price of the option. This changes the equation 2.8 to

$$\frac{\partial V}{\partial t} + \tfrac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r-D)S\frac{\partial V}{\partial S} - rV = 0. \tag{2.10}$$



(a) (b) (c)

Figure 2.1: One, two, and three-period binomial models applied to the same total period of time $T = N\delta t$. As the number of periods increases and the length of the timestep $\delta t \to 0$, the distribution of $\log S_T/S_0$ tends to the normal distribution.

2.2 Binomial Model

The binomial model presents another way to describe the random asset price dynamics. Starting out with a simple case of two possible asset prices per timestep, we will show that by increasing the number of timesteps, in the limit we will eventually arrive at the correct price of the option and find an alternative way to represent the value of the option, namely the risk-neutral expectation formula.

The original binomial model concept is due to Cox, Ross, and Rubinstein [3].

2.2.1 The binomial asset price process

We have essentially the same set of assumptions as in the above case where we described the random asset price process with a stochastic differential equation.

Instead of such an equation, the binomial model starts out with an extremely simple two-state market model shown on the left of Figure 2.1.

If S_0 is the spot price of a risky asset at time $t = 0$, one supposes that after some time period T, it can only assume two distinct values: S_0u or S_0d, where u, d are real numbers such that $u > d$. Moreover, we assume the existence of a riskless asset with a constant yield r. Thus an investment of S_0 dollars at time $t = 0$ yields S_0e^{rT} dollars at time $t = T$.

The no-arbitrage argument is valid also here. We must require that $S_0 d < S_0 e^{rT} < S_0 u$, or

$$d < e^{rT} < u. \tag{2.11}$$

This states that a riskless investment must not yield better, worse, or even as well as a risky investment. If this is not true, then the risky asset is not risky at all. If $e^{rT} < d < u$, one would never prefer the risk-free asset to the risky asset; borrowing ϕ units of money at the riskless rate r and buying the risky asset would yield a profit of at least $\phi(d - e^{rT}) > 0$ at time $t = T$. Even if e^{rT} were equal to d, we would at least be able to *expect* that such market position (long asset, short bond) would yield a positive value. The same argument with a reversed market position (long bond, short asset) would apply to the situation when $d < u \leq e^{rT}$.

Our goal is to replicate an option written on the risky asset in this setting. Suppose now that the option yields f_u and f_d if the underlying asset goes up or down, respectively. Consider a portfolio consisting of Δ units of the risky asset (say, a stock) and ψ units of the riskless asset (e.g. a money market account) forms a replicating portfolio when

$$\begin{cases} \Delta S_0 u + \psi e^{rT} &= f_u \\ \Delta S_0 d + \psi e^{rT} &= f_d \end{cases} \tag{2.12}$$

This is a system of two equations with two unknowns (Δ, ψ). There is a unique solution if and only if $u \neq d$, which is indeed clear from the no-arbitrage requirement 2.11. The solution is

$$\Delta = \frac{f_u - f_d}{S_0 u - S_0 d}, \quad \psi = e^{-rT}\frac{u f_d - d f_u}{u - d}.$$

Since the option payoff here is arbitrary, *any* option is replicatable in this market model. Since the option payoff at $t = T$ is equal to that of this portfolio, the value of the portfolio must be equal to that of the option. Denoting the present value of the option by V_0:

$$V_0 = \Delta S_0 + \psi = \frac{f_u - f_d + e^{-rT}(u f_d - d f_u)}{u - d};$$

introducing a new variable

$$q = \frac{e^{rT} - d}{u - d},$$

the value of the option at $t = 0$ can be expressed as

$$V_0 = e^{-rT}[qf_u + (1-q)f_d]. \tag{2.13}$$

The no-arbitrage argument (2.11) guarantees that $0 < q < 1$; thus the above formula can be considered to be a certain kind of an expectation formula:

$$V_0 = e^{-rT}\mathbf{E}_q[f], \tag{2.14}$$

where the expectation is taken under the probability measure q. This measure has the special property that if V_T is the value of the option at $t = T$, then using the values for f_u, f_d in the system (2.12) we obtain

$$\mathbf{E}_q[V_T] = (\Delta S_0 + \psi)e^{rT} = V_0 e^{rT}.$$

The expectation value of the option at some future time point is thus the same as that of the risk-free asset: it is irrelevant whether we buy the option or invest an equivalent amount of money in the risk-free security: we can expect the investment grows at the risk-free rate in either case. Thus this probability measure is called the *risk-neutral probability measure.*

Since the option is always replicatable, we can synthesize the option using long or short positions in the underlying asset and cash, and sell the option. The total value of this portfolio is zero if there is no arbitrage; a non-zero value would indicate an opportunity to make riskless profit.

2.2.2 Multiperiod model

Consider now a two-period model where the expiry of the option T is divided into two equal timesteps $T = 2\delta t$. As before, the risky asset moves upward by a factor of u and downward by a factor of d. This is illustrated in Figure 2.1 (b).

This *recombining binomial tree* has the end asset values (S_0u^2, S_0ud, S_0d^2) at time $t = T = 2\delta t$. Suppose now the option payoff function is $f(S)$. Let the three possible values for the option at $t = T$ be then

$$f_u = f(S_0u^2), \quad f_m = f(S_0ud), \quad f_d = f(S_0d^2)$$

Since the tree consists only of binomial branches, we can expect to find a value for the option in each node, including the node at $t = 0$.

Assuming no arbitrage and risk-neutrality we can apply the formula (2.13) to each of the individual branches in this tree to obtain a value for the option step by step. At time $t = \delta t$, the value of the option can be either of the two values

$$V_1^u = e^{-r\delta t}(qf_u + (1-q)f_m), \quad V_1^d = e^{-r\delta t}(qf_m + (1-q)f_d),$$

depending on whether the value of the asset jumped initially up or down. Applying the formula once again,

$$V_0 = e^{-r\delta t}\left(qV_1^u + (1-q)V_1^d\right). \tag{2.15}$$

If we substitute for V_1^u, V_1^d, we can write this as

$$V_0 = e^{-rT}\left(q^2 f(S_0u^2) + q(1-q)f(S_0ud) + (1-q)^2 f(S_0d^2)\right),$$

or

$$V_0 = e^{-rT}\sum_{j=0}^{2} q^j(1-q)^{2-j} f(S_0u^jd^{2-j}).$$

Now assume this formula works for a N-period model, where $T = N\delta t$:

$$V_0 = e^{-rT}\sum_{j=0}^{N}\binom{N}{j} q^j(1-q)^{N-j}\, f(S_0u^jd^{N-j}). \tag{2.16}$$

The payoffs at each end node in the N-period model can be expressed as functions of the payoffs in an $N+1$-period model:

$$f(S_0u^jd^{N-j}) = e^{-r\delta t}\left[qf(S_0u^{j+1}d^{N-j}) + (1-q)f(S_0u^jd^{N+1-j})\right], \tag{2.17}$$

Replacing (2.17) into (2.16) will yield

$$\begin{aligned} V_0 &= e^{-r(T+\delta t)}q^{N+1}f(S_0u^N) \\ &+ e^{-r(T+\delta t)}\sum_{j=1}^{N}\left[\binom{N}{j} + \binom{N}{j-1}\right] q^j(1-q)^{N-j}f(S_0u^jd^{N-j}) \\ &+ e^{-r(T+\delta t)}(1-q)^{N+1}f(S_0d^N); \end{aligned}$$

since $\binom{N}{j} + \binom{N}{j-1} = \binom{N+1}{j}$, this equals

$$e^{-r(N+1)\delta t)}\sum_{j=0}^{N+1}\binom{N+1}{j} q^j(1-q)^{N+1-j}\, f(S_0u^jd^{N+1-j}),$$

which confirms that the formula is also valid for a $N+1$-period model. Given that (2.15) is true, by induction, the formula (2.16) is true for all N.

Since the weights $\binom{N}{j}q^j(1-q)^{N-j}$ add up to 1, we can again express this as an expectation with respect to the measure q: $V_0 = e^{-rT}\mathbf{E}_q[f]$.

2.2.3 Continuum limit

The N-period model derived above is still a discrete model, while the Black-Scholes model derived by a stochastic differential equation was continuous. What would happen if $N \to \infty$, or $\delta t \to 0$? We would like to claim that the lognormal process can be approximated by this process so that in the limit the two models are equivalent. If this is true, we will have two equivalent ways of deriving the value of the option.

Consider value of the underlying asset (the stock) after n periods have passed. There has been a random number of, say X_n, 'up-jumps,' and consequently $n - X_n$ 'down-jumps.' The value of the asset is then

$$S_n = S_0 u^{X_n} d^{(n-X_n)}.$$

The lognormal process introduced in the previous section involves a stock price $S_t = S_0 e^{X_t}$ where X_t is given by its differential equation (2.3): it is a stochastic process with drift μt and variance $\sigma^2 t$. We have not specified the variables u, d yet; let us try

$$u = e^{\mu\delta t + \sigma\sqrt{\delta t}}, \quad d = e^{\mu\delta t - \sigma\sqrt{\delta t}}.$$

It is clear that $d < u$. For now, assume that this satisfies the arbitrage requirement (2.11). Then the asset price after n periods is

$$\begin{aligned} S_n &= S_0 e^{(\mu\delta t + \sigma\sqrt{\delta t})X_n} e^{(\mu\delta t - \sigma\sqrt{\delta t})(N-X_n)} \\ &= S_0 e^{\mu t + (2X_n - n)\sigma\sqrt{\delta t}} \end{aligned}$$

The question is, in the limit $n \to \infty$, will $\mu t + (2X_n - n)\sigma\sqrt{\delta t}$ tend to X_t ? If it does, the binomial model can be used for approximating the lognormal process.

Using the values for u, d we can write q as

$$q = \frac{e^{r\delta t} - d}{u - d} = \frac{e^{r\delta t} - e^{\mu t - \sigma\sqrt{\delta t}}}{e^{\mu t + \sigma\sqrt{\delta t}} - e^{\mu t + \sigma\sqrt{\delta t}}}.$$

Since we are attempting to get down to the limit $\delta t \to 0$, we would like to know how q behaves when δt is near 0. Expanding the denominator and the numerator in Taylor series,

$$\begin{aligned} q &= \frac{r\delta t - (\mu\delta t - \sigma\sqrt{\delta t} + \frac{1}{2}\sigma^2\delta t) + O(\delta t^{3/2})}{(\mu\delta t + \sigma\sqrt{\delta t} + \frac{1}{2}\sigma^2\delta t) - (\mu\delta t - \sigma\sqrt{\delta t} + \frac{1}{2}\sigma^2\delta t) + O(\delta t^{3/2})} \\ &= \frac{\sigma\sqrt{\delta t} + (r - \mu - \frac{1}{2}\sigma^2)\delta t + O(\delta t^{3/2})}{2\sigma\sqrt{\delta t} + O(\delta t^{3/2})} \\ &= \tfrac{1}{2} + \frac{(r - \mu - \frac{1}{2}\sigma^2)}{\sigma}\sqrt{\delta t} + O(\delta t) \end{aligned}$$

The random component of S_n is

$$(2X_n - n)\sigma\sqrt{\delta t} = (2X_n - n)\sigma\sqrt{\frac{t}{n}} = (\frac{2X_n - n}{\sqrt{n}})\sigma\sqrt{t}$$

Let us denote $(2X_n - n)/\sqrt{n}$ by Y_n and find the mean and variance of Y_n. X_n can be characterized as a sum of n independent Bernoulli random variables, i.e. a random variable denoting the number of heads in n flips of a coin; here the coin is not fair but a 'risk-neutral coin' with the probability of 'heads' being equal to q. X_n has thus mean nq and variance $nq(1-q)$.

The mean of Y_n is then

$$\mathbf{E}[Y_n] = \frac{(2\mathbf{E}[X_n] - n)}{\sqrt{n}} = (2q-1)\sqrt{n} = \sqrt{t}\frac{(r - \mu - \frac{1}{2}\sigma^2)}{\sigma} + O(\delta t),$$

and the variance

$$\mathrm{Var}[Y_n] = \mathrm{Var}[2X_n/\sqrt{n}] = 4\mathrm{Var}[X_n]/n = 4q(1-q) = 1 + O(\delta t).$$

In the limit $N \to \infty$, $\delta t \to 0$, and $Y_n\sigma\sqrt{t}$ tends to a stochastic process Y_t distributed normally with mean $(r - \mu - \frac{1}{2}\sigma^2)t$ and variance $\sigma^2 t$. We can express this as

$$Y_t = (r - \mu - \tfrac{1}{2}\sigma^2)t + \sigma W_t$$

where W_t is a Brownian motion. We have shown that the discrete random process

$$S_n = S_0 e^{\mu t + (2X_n - n)\sigma\sqrt{\delta t}}$$

tends to the (continuous) stochastic process

$$S_t = S_0 e^{\mu t + Y_t}.$$

If we denote the exponent $\mu t + Y_t$ by X_t, we obtain

$$X_t = (r - \tfrac{1}{2}\sigma^2)t + \sigma W_t. \tag{2.18}$$

We can thus approximate the lognormal asset price process with the binomial model by setting

$$u = e^{(r-\frac{1}{2}\sigma^2)\delta t + \sigma\sqrt{\delta t}}, \quad d = e^{(r-\frac{1}{2}\sigma^2)\delta t - \sigma\sqrt{\delta t}}, \quad q = \frac{e^{r\delta t} - d}{u - d}. \tag{2.19}$$

The no-arbitrage requirement $d < e^{r\delta t} < u$ translates to

$$r\delta t - \sigma\sqrt{\delta t} + O(\delta t^{3/2}) < r\delta t + O(\delta t^2) < r\delta t + \sigma\sqrt{\delta t} + O(\delta t^{3/2})$$

which is true as $\delta t \to 0$.

Let us now find $\mathrm{d}X_t$. Apply Itô calculus to equation (2.18) :

$$\mathrm{d}X_t = (r - \tfrac{1}{2}\sigma^2)\,\mathrm{d}t + \tfrac{1}{2}\sigma^2\,\mathrm{d}t + \sigma\,\mathrm{d}W_t = r\,\mathrm{d}t + \sigma\,\mathrm{d}W_t.$$

This implies that in the risk-neutral world, the stochastic differential equation (2.3) has drift $\mu = r$. In other words, $\mathbf{E}[\,\mathrm{d}X_t] = \mathbf{E}[\,\mathrm{d}S_t/S_0] = r\,\mathrm{d}t$, or

$$\mathbf{E}[S_t] = S_0\mathbf{E}[e^{X_t}] = S_0 e^{rt}, \tag{2.20}$$

i.e. the price of the stock is expected to grow at the risk-neutral rate r. This result also immediately confirms the risk-neutral expectation formula in the case of an option which is the underlying itself, i.e. $V_t = S_t$:

$$V_0 = e^{-rT}\mathbf{E}[V_T] = e^{-rT}\mathbf{E}[S_T] = e^{-rT}S_0 e^{rT} = S_0.$$

Conclusion. In the absence of arbitrage we may assume the underlying asset S_t is a stochastic process $S_0 e^{X_t}$, where X_t has mean $(r - \frac{1}{2}\sigma^2)t$ and variance $\sigma^2 t$. Using the distribution function of X to evaluate the expected value of V_T, we obtain the *risk-neutral expectation formula*

$$V(S_0, 0) = e^{-rT}\mathbf{E}[V(S_0 e^{X_T}, T)] \tag{2.21}$$

where the expectation is taken under the risk-neutral probability measure.

The price of an option thus not depend on the (subjective) rate of drift of the underlying μ at all. This is obvious if we consider the fact that we are actually assuming that the option is perfectly replicated at each timestep $\mathrm{d}t$ *before* the price of the underlying changes, so we actually are able to predict its fair price in an arbitrage-free market. Any guess of the magnitude of μ is therefore irrelevant to the pricing of options.

Thus all traders and investors in the market can agree on the price of the option if only they agree on the volatility term. Since σ is usually derived from actual observed market data (this is called *implied volatility*), it can be assumed to be universally known.

The expected growth rate of the option being equal to that of a risk-free security reflects the essence of risk-neutrality: options are priced considering the yield of the alternative, riskless investment opportunity, the deterministic *opportunity cost*, which acts as a benchmark when comparing other investments. A replicated option is made up deterministic and since risk-free. As we have shown both in the stochastic model and the binomial model, if the option is replicatable then it can be considered to be comparable to a risk-free security.

2.2.4 Determining the parameters u, d, q

(2.19) is not the only possible way to construct a risk-neutral binomial tree. It also is possible to choose q so that it does not depend on r nor on σ.

The lognormal model is fully specified by the mean and variance of the random variable $S_T = S_0 e^{X_T}$, or those of $S_T/S_0 = e^{X_T}$, where X_t is defined in equation (2.18) . The variance of e^{X_T} is

$$\mathrm{Var}[e^{X_T}] = \mathbf{E}[e^{2X_T}] - \mathbf{E}[e^{X_T}]^2.$$

Also, e^{X_T} has mean e^{rT}, as shown in equation (2.20) . To find the mean of e^{2X_T}, we can apply Itô calculus to $2X_t = 2(r - \frac{1}{2}\sigma^2)t + 2\sigma W_t$ to find $\mathrm{d}X_t$:

$$\mathrm{d}X_t = 2(r - \tfrac{1}{2}\sigma^2)\,\mathrm{d}t + \tfrac{1}{2}(2\sigma)^2\,\mathrm{d}t + 2\sigma\,\mathrm{d}W_t = (2r + \sigma^2)\,\mathrm{d}t + 2\sigma\,\mathrm{d}W_t$$

This means that e^{2X_T} has mean $e^{(2r+\sigma^2)T}$. Thus e^{X_T} has the variance

$$\mathrm{Var}[e^{X_T}] = e^{2rT+\sigma^2 T} - \mathbf{E}[e^{X_T}]^2$$

Since $e^{X_T} = S_T/S_0$, we can write the equations as

$$\begin{cases} \mathbf{E}[S_T/S_0] & = e^{rT} \\ \mathrm{Var}[S_T/S_0] & = e^{(2r+\sigma^2)T} - \mathbf{E}[S_T/S_0]^2 \end{cases} \tag{2.22}$$

We will now apply this to the one-period binomial model with $T = \delta t$ to emphasize the fact that we want to find the limit $\delta t \to 0$. In this model, the mean and variance are given by

$$\begin{cases} \mathbf{E}[S_T/S_0] & = qu + (1-q)d \\ \mathrm{Var}[S_T/S_0] & = qu^2 + (1-q)d^2 - \mathbf{E}[S_T/S_0]^2. \end{cases} \tag{2.23}$$

Equating (2.23) and (2.22) we obtain the conditions determining u, d, and q:

$$\begin{cases} qu + (1-q)d & = e^{r\delta t} \\ qu^2 + (1-q)d^2 & = e^{2r\delta t + \sigma^2 \delta t}. \end{cases} \tag{2.24}$$

This is two equations with three unknowns, and hence one variable can be chosen as we please.

Popular choices are $q = \frac{1}{2}$ and $d = 1/u$. For example, if we set $q = \frac{1}{2}$ to get

$$\begin{cases} q + d & = 2e^{r\delta t} \\ u^2 + d^2 & = 2e^{2r\delta t + \sigma^2 \delta t}, \end{cases}$$

from where we obtain

$$\begin{cases} u & = e^{r\delta t}\left(1 + \sqrt{e^{\sigma^2 \delta t} - 1}\right) \\ d & = e^{r\delta t}\left(1 - \sqrt{e^{\sigma^2 \delta t} - 1}\right) \\ q & = \frac{1}{2} \end{cases} \tag{2.25}$$

2.2.5 Deriving the Black-Scholes equation

The above argument proves that the binomial model approximates the log-normal price process. To conclude the discussion about the equivalence of the two models, we will now show that given the risk-neutral binomial process, we can derive the Black-Scholes equation from the risk-neutral expectation formula (2.14).

Consider the single-period binomial model. Let the current spot price be S. The risk-neutral expectation is given by

$$\mathbf{E}_q[V(S,0)] = qV(Su, \delta t) + (1-q)V(Sd, \delta t). \tag{2.26}$$

Now expand $V(Su, \delta t)$ in Taylor series about $V(S, \delta t)$:

$$\begin{aligned} V(Su, \delta t) &= V(S + S(u-1), \delta t) \\ &= V(S, \delta t) + V'(S, \delta t)S(u-1) + \tfrac{1}{2}V''(S, \delta t)S^2(u-1)^2 + O(u^3) \end{aligned}$$

Here $' \equiv \partial/\partial S$. Similarly, we can find a formula for $V(Sd, \delta t)$. Then, equation 2.26 yields

$$\begin{aligned}
\mathbf{E}_q[V(S,0)] &= V(S,\delta t) + V'(S,\delta t)S[q(u-1)-(1-q)(d-1)] \\
&\quad + \tfrac{1}{2}V''(S,\delta t)S^2(u-1)^2 + O(u^3) \\
&= V(S,\delta t) + V'(S,\delta t)S[e^{r\delta t}-1] \\
&\quad + \tfrac{1}{2}V''(S,\delta t)S^2(u-1)^2 + O(u^3) \\
&= V(S,\delta t) + V'(S,\delta t)Sr\delta t + \tfrac{1}{2}V''(S,\delta t)S^2\sigma^2\delta t + O(\delta t^{3/2})
\end{aligned}$$

Above we used the equality $qu + (1-q)d = e^{r\delta t}$. By the risk-neutrality argument, this must be equal to

$$V(S,0)e^{r\delta t} = V(S,0)(1 + r\delta t) + O(\delta t^2).$$

Rearranging,

$$V(S,\delta t) - V(S,0) + \tfrac{1}{2}S^2\sigma^2 V''(S,\delta t)\delta t + rSV'(S,\delta t)\delta t - rV(S,0)\delta t + O(\delta t^{3/2}) = 0$$

and dividing by δt,

$$\frac{V(S,\delta t) - V(S,0)}{\delta t} + \tfrac{1}{2}S^2\sigma^2 V''(S,\delta t) + rSV'(S,\delta t) - rV(S,0) + O(\delta t) = 0$$

Finally, let $\delta t \to 0$ to obtain the Black-Scholes partial differential equation:

$$\frac{\partial V}{\partial t} + \tfrac{1}{2}S^2\sigma^2\frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - Vr = 0.$$

2.2.6 Constant dividend yield

Consider now an asset with constant dividend yield D so that buying ϕ units of the asset at the current spot price S_0 is worth $\phi e^{D\delta t}S_0 u$ if the price goes up and $\phi e^{D\delta t}S_0 d$ if the price goes down. This is illustrated in Figure 2.2 (a).

By multiplying the tree by $e^{-D\delta t}$ we get the binomial tree with the familiar end node values $(S_0 u, S_0 d)$ but with initial value $S_0 e^{-D\delta t}$; see Figures 2.2 (b) and (c). Substituting S_0 for $S_0 e^{-D\delta t}$ in (2.12) we get

$$q = \frac{S_0 e^{-D\delta t}e^{r\delta t} - S_0 d}{S_0 u - S_0 d} = \frac{e^{(r-D)\delta t} - d}{u - d}.$$

This means that in the derivation of u and d above we must use $r - D$ instead of r. In the end we obtain

$$u = e^{(r-D-\frac{1}{2}\sigma^2)\delta t + \sigma\sqrt{\delta t}}, \quad d = e^{(r-D-\frac{1}{2}\sigma^2)\delta t - \sigma\sqrt{\delta t}}, \quad q = \frac{e^{(r-D)\delta t} - d}{u - d}. \tag{2.27}$$

$$\begin{tikzcd}
& \phi e^{D\delta t}S_0u & & \phi S_0u & & S_0u \\
\phi S_0 \arrow[ur] \arrow[dr] & & \phi S_0e^{-D\delta t} \arrow[ur] \arrow[dr] & & S_0e^{-D\delta t} \arrow[ur] \arrow[dr] & \\
& \phi e^{D\delta t}S_0d & & \phi S_0d & & S_0d \\
\text{(a)} & & \text{(b)} & & \text{(c)} &
\end{tikzcd}$$

Figure 2.2: Binomial model of an asset with constant dividend yield. (a) Buying ϕ units of the asset at S_0 yields either $\phi e^{D\delta t}S_0u$ or $\phi e^{D\delta t}S_0u$ after timestep δt. (b) Buying $\phi e^{D\delta t}$ units instead makes the end node values the same as in the no-dividend model. (c) Thus if we discount the present value of the asset by its dividend yield D, we can apply the no-dividend model with the risk-free rate $(r - D)$ instead of r.

2.3 Explicit formulas

The Black-Scholes equation can be solved for the plain vanilla European options such as the European put and call options and the European binary option. These formulas will be used as reference when testing the numerical methods.

There are many ways to derive the basic formulas, but perhaps the most straightforward is to use the risk-neutral expectation formula (2.21).

Take the European put option $P(S_t, t)$. Its payoff at expiry $t = T$ is $(K - S_T)_+$, so the value of the option at $t = 0$ can be expressed as

$$P(S_0, 0) = e^{-rT}\mathbf{E}[(K - S_T)_+] = e^{-rT}\mathbf{E}[(K - S_0e^{X_T})_+].$$

The argument under the operator $\mathbf{E}$ is positive when $X_T < \log K/S_0$. Denote the expectation value restricted to this domain by $\mathbf{E}_+$ so by the linearity of the expectation operator we can write

$$P(S_0, 0) = e^{-rT}\left(K\mathbf{E}_+[1] - S_0\mathbf{E}_+[e^{X_T}]\right). \tag{2.28}$$

For a standard normal random variable Z, the expectation value $\mathbf{E}_+[f(Z)]$

for $Z < z_0$ is given by

$$\mathbf{E}_+[f(Z)] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z_0} f(z) \exp\left(-\tfrac{1}{2}z^2\right) \mathrm{d}z.$$

If X_T is the stochastic process defined in equation (2.18) , then

$$Z = \frac{X_T - (r - \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}} \tag{2.29}$$

is standard normal. Then X_T is restricted to $-\infty < X_T < \log(K/S_0)$:

$$\mathbf{E}_+[1] = \frac{1}{\sigma\sqrt{2\pi T}} \int_{-\infty}^{\log K/S_0} \exp\left(-\tfrac{1}{2}\left[\frac{x - (r - \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}\right]^2\right) \mathrm{d}x.$$

By the variable change (2.29), we obtain

$$\mathbf{E}_+[1] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z_1} \exp\left(-\tfrac{1}{2}z^2\right) \mathrm{d}z \equiv \Phi(z_1), \tag{2.30}$$

where $\Phi(\cdot)$ is the cumulative normal distribution function defined by the improper integral on the left-hand side, and z_1 is obtained by the variable change (2.29):

$$z_1 = \frac{\log K/S_0 - (r - \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}. \tag{2.31}$$

Similarly, we compute

$$\mathbf{E}_+[e^{X_T}] = \frac{1}{\sigma\sqrt{2\pi T}} \int_{-\infty}^{\log K/S_0} \exp\left(x - \tfrac{1}{2}\left[\frac{x - (r - \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}\right]^2\right) \mathrm{d}x.$$

Completing the square in the exponent and changing the variables, we obtain

$$\mathbf{E}_+[e^{X_T}] = \frac{e^{rT}}{\sqrt{2\pi}} \int_{-\infty}^{z_2} \exp\left(-\tfrac{1}{2}z^2\right) \mathrm{d}z = e^{rT}\Phi(z_2),$$

where

$$z_2 = z_1 - \sigma\sqrt{T} = \frac{\log K/S_0 - (r + \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}. \tag{2.32}$$

Applying the expectation formulas to (2.28) yields the *Black-Scholes put option formula*:

$$P(S_0, 0) = Ke^{-rT}\Phi(z_1) - S_0\Phi(z_2). \tag{2.33}$$

The *Black-Scholes call option formula* is derived easily using the put-call parity (1.7):

$$C(S_0, 0) - \Big(Ke^{-rT}\Phi(z_1) - S_0\Phi(z_2)\Big) = S_0 - Ke^{-rT}$$

gives

$$C(S_0, 0) = S_0[1 - \Phi(z_2)] - Ke^{-rT}[1 - \Phi(z_1)]$$

which is equivalent to

$$C(S_0, 0) = S_0\Phi(-z_2) - Ke^{-rT}\Phi(-z_1). \tag{2.34}$$

Binary puts and calls

The binary put option value is simply

$$P_B(S_0, 0) = e^{-rT}\mathbf{E}_+[1],$$

which by (2.30) can be written immediately as

$$P_B(S_0, 0) = e^{-rT}\Phi(z_1), \tag{2.35}$$

By the binary put-call parity (1.5), we obtain the binary call option formula

$$C_B(S_0, 0) = e^{-rT}\Phi(-z_1). \tag{2.36}$$

Constant dividend yield

If the underlying asset yields a constant dividend yield D, the formulas can be modified simply by replacing the risk-free rate r with $r-D$. Alternatively, we can replace the spot price S_0 with S_0e^{-DT} everywhere in the formulas. This affects also the variables z_1, z_2.

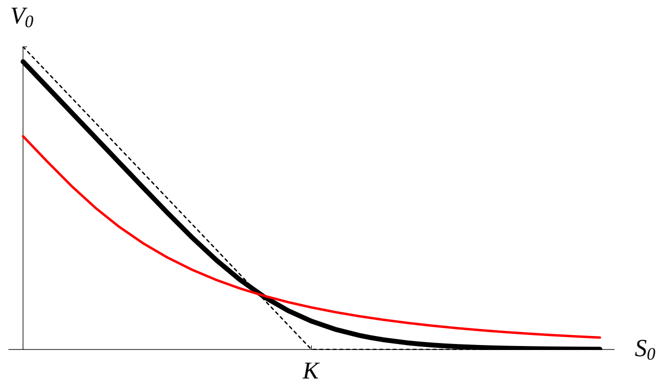


Figure 2.3: Black-Scholes solution (2.33) for a European put option. $\sigma = 0.3$, $r = 0.05$. The thick continuous curve has expiry = 1.0, and the thin (red) continuous curve has expiry = 7.0. By the put-call parity, $P = Ke^{-rT} - S_0 + C$, so at $S_0 = 0$, the curve touches the V_0-axis at $V_0 = Ke^{-rT}$. In time, the curve flattens and approaches zero. The payoff function is shown as a dotted line.

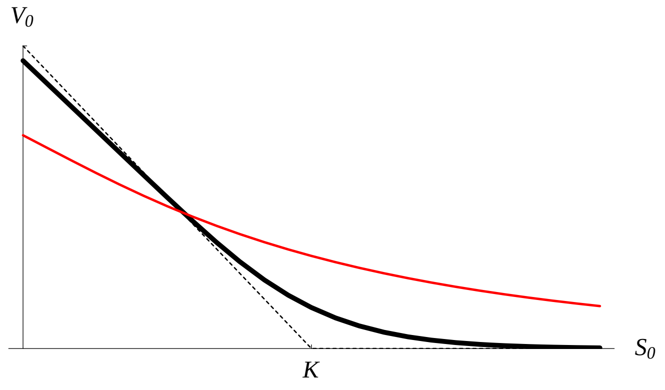


Figure 2.4: Black-Scholes solution (2.33) for a European put option on an underlying with constant dividend yield $D = 0.10$. $\sigma = 0.3$, $r = 0.05$. The thick continuous curve has expiry = 1.0, and the thin (red) continuous curve has expiry = 7.0. By the put-call parity, $P = Ke^{-rT} - S_0e^{-DT} + C$, so at $S_t = 0$, the curve touches the V_0-axis at $V_0 = Ke^{-rT}$. In time, the curve flattens and approaches zero. The payoff function is shown as a dotted line.

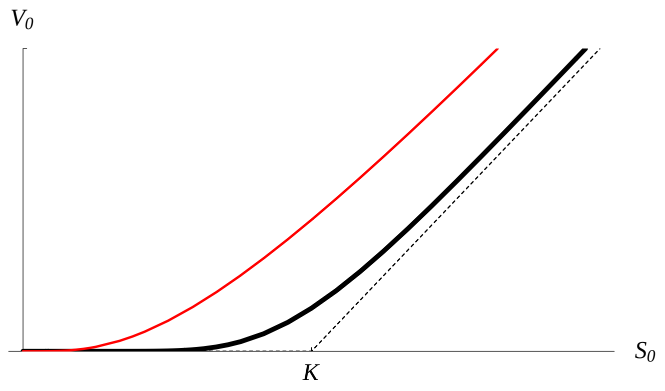


Figure 2.5: Black-Scholes solution (2.34) for a European call option. $\sigma = 0.3$, $r = 0.05$. The thick continuous curve has expiry $= 1.0$, and the thin (red) continuous curve has expiry $= 7.0$. As $S_t \to \infty$, the value of the call approaches the forward contract $C \to S_0 - Ke^{-rT}$. In time, the term $-Ke^{-rT}$ approaches zero so the value function of the call approaches that of the asset itself, S_0. The payoff function is shown as a dotted line.

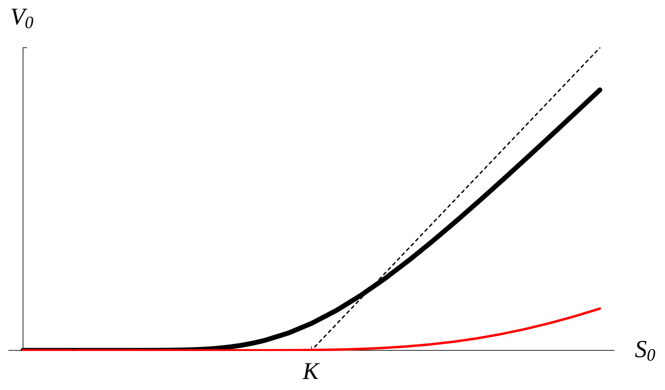


Figure 2.6: Black-Scholes solution (2.34) for a European call option on an underlying with constant dividend yield $D = 0.10$. $\sigma = 0.3$, $r = 0.05$. The thick continuous curve has expiry $= 1.0$, and the thin (red) continuous curve has expiry $= 7.0$. As $S_t \to \infty$, the value of the call approaches the forward contract $C \to S_0 e^{-DT} - Ke^{-rT}$. In time, both of the terms approach zero. The function tends to zero at both ends of the positive S_0 axis, and it gradually loses its value. The payoff function is shown as a dotted line.

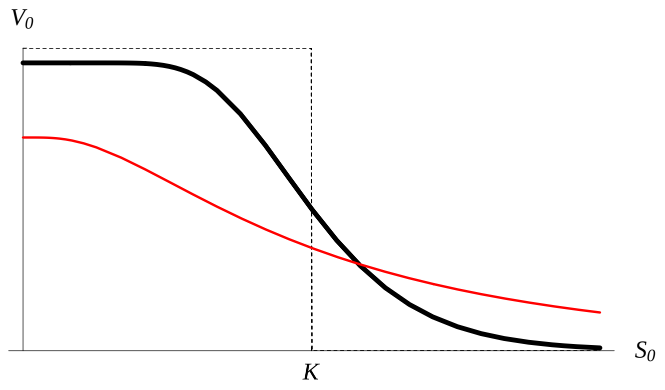


Figure 2.7: Black-Scholes solution (2.35) for a European binary put option. $\sigma = 0.3$, $r = 0.05$. Expiry = 1.0. The thick continuous curve has expiry = 1.0, and the thin (red) continuous curve has expiry = 7.0. By the binary put-call parity, it can be shown that at $S_t = 0$, the curve touches the V-axis at e^{-rT}. In time, the curve flattens and approaches zero. The payoff function is shown as a dotted line.

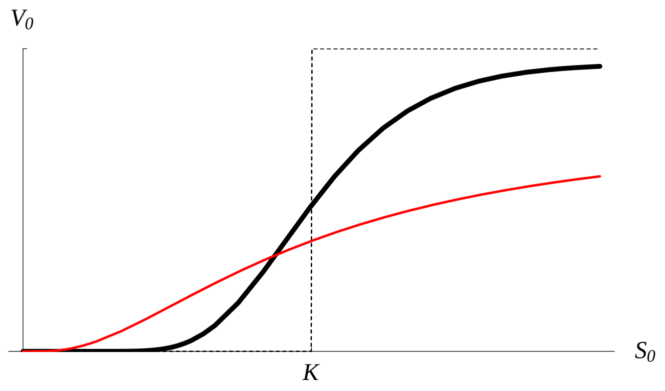


Figure 2.8: Black-Scholes solution (2.36) for a European binary call option. $\sigma = 0.3$, $r = 0.05$. The thick continuous curve has expiry = 1.0, and the thin (red) continuous curve has expiry = 7.0. By the binary put-call parity, it can be shown that for large values of $S_t > K$, the curve is near e^{-rT}. In time, the curve flattens and approaches zero. The payoff function is shown as a dotted line.

Chapter 3

The Binomial Method

The binomial method is a very straightforward and easily programmable method to value options. Using the binomial model described in the previous section, we can approximate the value of an option by using the recursive formula (2.13), or alternatively the risk-neutral expectation formula (2.16).

As shown in the previous chapter, the binomial process approximates the lognormal asset price process with increasing accuracy as the number of periods used N increases.

We will use the recombining tree model as shown in Figure 2.1. It is clear that if T is divided into N periods, the end nodes in a recombining binomial tree specifies $N+1$ possible stock prices.

Using the values for u, d, q as shown in (2.19), we can calculate the jth stock price at time $t = n\delta t$:

$$S_j^n = S_0 u^j d^{n-j}, \quad 0 \le j \le n. \tag{3.1}$$

At expiry $t = T = N\delta t$, the prices for the option are all known for certain, so if $f(\cdot)$ is the payoff function for the option, we can assign the option the price

$$V_j^N = f(S_j^N) = f(S_0 u^j d^{N-j}). \tag{3.2}$$

For example, if we have a plain vanilla European put or call option with strike K,

$$\begin{array}{ll} V_j^N = (K - S_j^N)_+ & \text{for a European put} \\ V_j^N = (S_j^N - K)_+ & \text{for a European call} \end{array}$$

3.1 Pricing European Options

Most of the theory is already discussed in section 2.2. Once u, d, q are fixed, we can use the formula (2.13) to calculate the option values recursively working our way backward in the tree from the payoff values applying the formula at each node, and eventually recover the option value at $t = 0$.

Another method is to use the risk-neutral expectation formula to sum up the $N+1$ weighted payoffs using formula (2.16) and take the discounted value. This method is not usually presented in the standard literature; however with a smart choice of the risk-neutral probability q it is easily implemented even on a spreadsheet application.

3.1.1 Recursive method

The formula 2.13 describes a method to find all of the possible values for the option at $t = (n-1)\delta t$ given the option values (i.e. the payoff) at expiry $t = n\delta t = T$.

Given the option payoffs at the final nodes V_j^N for all $0 \le j \le n$. and the appropriate parameters u, d, q.

The previous N values can be then calculated by the recursive formula

$$V_j^{n-1} = e^{-r\delta t}\left(qV_{j+1}^n + (1-q)V_j^n\right) \quad \text{for} \quad 0 \le j \le n-1. \tag{3.3}$$

If we consider the above calculation one operation, it takes N operations to find the option values for the period $N-1$, and $N-1$ operations to find the option values for the period $N-2$, and so on. Thus, to find the present value of the option (at time $t = 0$), we need

$$N + (N-1) + (N-2) + \cdots 1 = \frac{N(N-1)}{2}$$

operations. The method takes $O(\frac{1}{2}N^2)$ operations to compute the value of the option.

3.1.2 Summation method

We use the discrete risk-neutral expectation formula (2.16). First, we calculate the binomial coefficients

$$B_j = \binom{N}{j} q^j (1-q)^{N-j}, \quad 0 \le j \le N.$$

and then sum them up, each multiplied by the corresponding payoff value, and discount the sum to get the present value:

$$V_0 = e^{-rT} \sum_{j=0}^{N} B_j f(S_j^N)$$

This method has the advantage that when implemented as a computer program, we can store the binomial coefficients B_j and S_j^N in arrays and reuse them for different payoffs.

We can choose any valid combination for u, d, and q. In any case, if r or σ changes, either the binomial coefficients B_j or the asset values S_j^N must be recomputed.

An easily computable choice is (2.25), where $q = \frac{1}{2}$. For an N-period model we need $N + 1$ values

$$B_j = \binom{N}{j} q^j (1-q)^{N-j} = \binom{N}{j} \frac{1}{2^N}, \quad 0 \leq j \leq N$$

so the value of the option is given by

$$V_0 = \frac{e^{-rT}}{2^N} \sum_{j=0}^{N} \binom{N}{j} f(S_j^N) \tag{3.4}$$

where S_j^N is given by (3.1) with u, d specified by (2.25).

Once N is fixed, it is a trivial task to calculate the binomial coefficients. The coefficients $\binom{N}{j} 2^{-N}$ and the asset values S_j^N can be stored in an array and recalled each time the payoff function or the parameters $r, \sigma, \delta t, N$ change. Given that the coefficients and the asset values S_j^N are immediately available, the method requires $O(N)$ operations. A typical algorithm takes $O(\frac{1}{2}N^2)$ operations to calculate the coefficients. If $r, \sigma, \delta t, N$ do change often, the recursive method will have a slight time advantage due to fewer calculations needed—but both methods still converge at the same rate, as seen in figure (3.5).

3.1.3 Testing and Calibration

The figure (3.1) shows a European put option valued by the binomial method with 64 periods. The binomial method only allows us to calculate one single option value (for a given spot price); here we have calculated several sample points. The figure (3.2) shows a call option with the same parameters.

The choice of the strike value $K = 1.0$ may sound somewhat unusual, but for testing purposes it is as good as any price; at least it is the most neutral one. The expiry T is set to 1.0 years, while the volatility is 0.30 .

Errors. We define the error simply as the difference between the binomial approximation and the value computed by the Black-Scholes formula.

The figure (3.3) shows the corresponding errors relative to the spot price. The maximum error is the largest around the strike K, where $\partial V/\partial S$ is discontinuous. The discontinuity implies large values for the second derivative $\partial^2 V/\partial S^2$ near expiry, approaching infinity as $t \to T$. This causes inaccuracies in the pricing process.

The figure (3.5) shows how the maximum error over several (64) sample points in the interval $[0..2K]$ changes as we increase the number of periods. Both the results from the recursive method and summation method are shown; the results are almost identical.

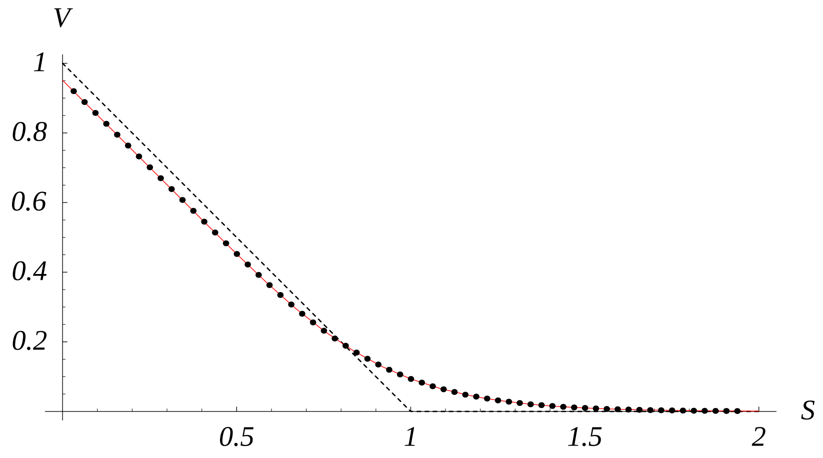


Figure 3.1: European put option calculated by the binomial method. Strike $K = 1.0$, $T = 1.0$, $\sigma = 0.30$, $r = 0.05$. Each of the sample points (shown as dots) were calculated using a 64-period binomial tree. The binomial method was applied repeatedly for several spot prices S_0. The continuous thin (red) curve is the Black-Scholes solution for the put option.

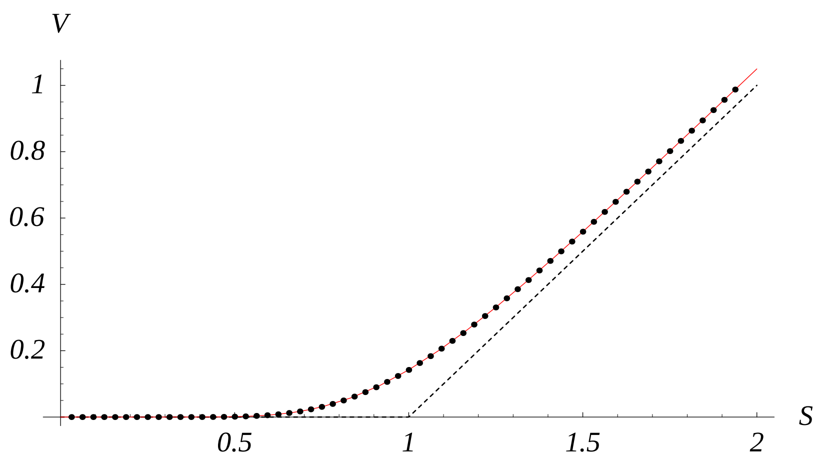


Figure 3.2: European call option calculated by the binomial method. Strike $K = 1.0, T = 1.0$, $\sigma = 0.30$, $r = 0.05$. Each of the sample points (shown as dots) were calculated using a 64-period binomial tree. The binomial method was applied repeatedly for several spot prices S_0. The continuous thin (red) curve is the Black-Scholes solution for the put option.

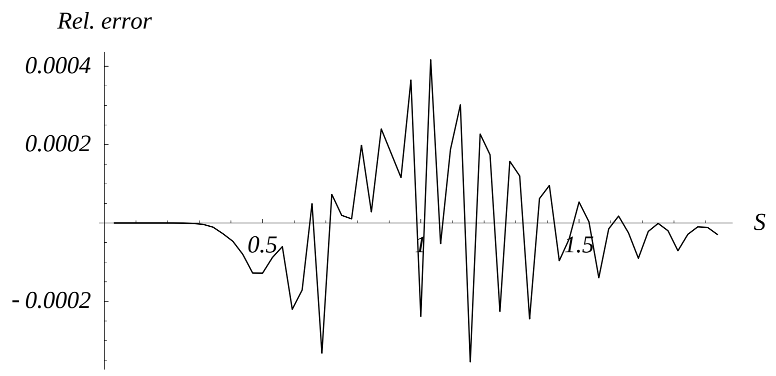


Figure 3.3: Relative errors (ε/S) in the European put valuation due to the binomial recursive method using a 64-period tree. The errors in call valuation are very similar.

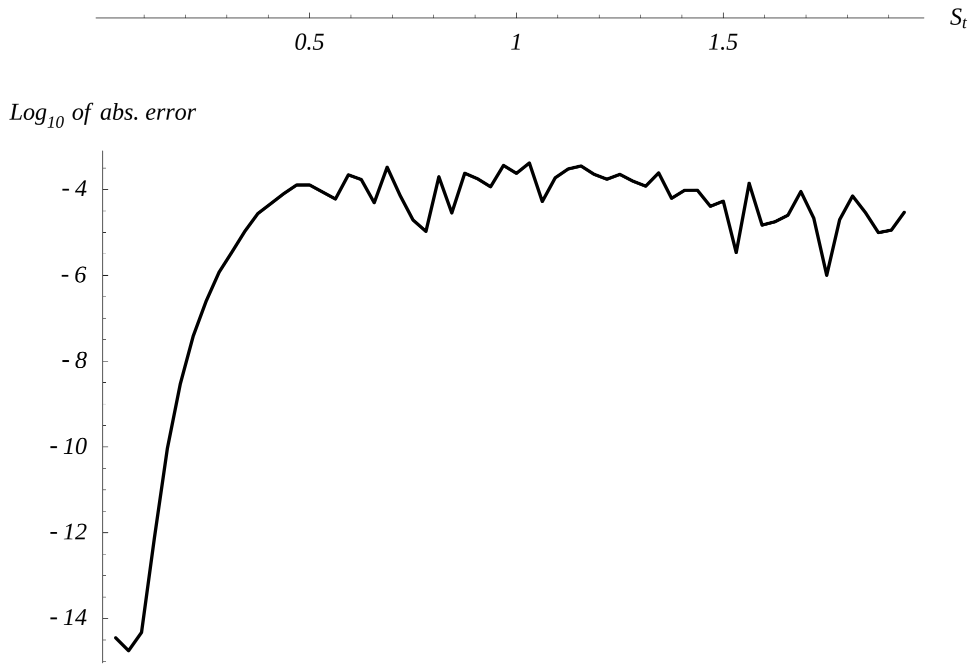


Figure 3.4: Logarithm of relative errors $\log(\varepsilon/S)$ in the European put valuation due to the binomial recursive method using a 64-period tree. The errors measured in valuing calls, and those due to the summation method are all very similar and of the same order of magnitude.

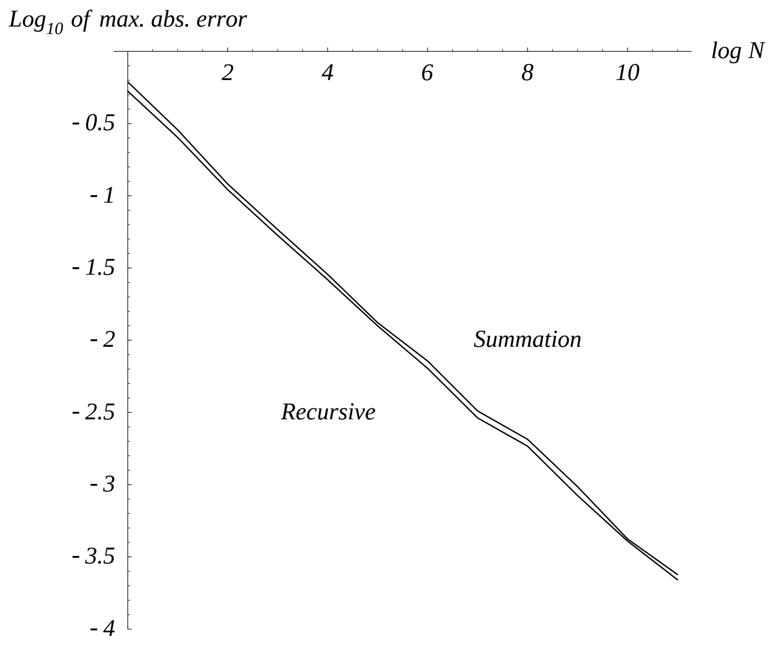


Figure 3.5: Convergence of the binomial methods. Strike $K = 1.0$, $T = 1.0$, $\sigma = 0.30$, $r = 0.05$. Errors in an approximation of an European call option were sampled on 64 points in the interval $[0..2K]$ and the largest one in absolute value was chosen for the plot. The graph shows the logarithm of the absolute error with relation to the logarithm of the number of periods. The slope is approximately -0.7.

Chapter 4

Finite-Difference Methods

Finite-difference methods attempt to solve the Black-Scholes partial differential equation by approximating the partial derivatives by either an explicit or implicit solution equation.

The methods are the simplest and most effective when applied to the diffusion equation $u_\tau = u_{xx}$. We will first show that the Black-Scholes equation can indeed be transformed into this simple form by a routine variable change. Then, we will introduce the numerical methods for solving it.

The most commonest finite-difference methods for solving the diffusion equation are the so-called explicit method the fully implicit method, and the Crank-Nicolson method. We will see that these are closely related but differ in stability, accuracy, and execution speed.

4.1 The Diffusion Equation

Consider the Black-Scholes equation

$$\frac{\partial V}{\partial t} + \tfrac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r-D)S\frac{\partial V}{\partial S} - rV = 0$$

and apply the following change of variables:

$$S = Ke^x,\ t = T - \tau/(\tfrac{1}{2}\sigma^2),\ V(S(x), t(\tau)) = Kv(x,\tau). \tag{4.1}$$

Substituting, we get

$$v_\tau = v_{xx} + (\frac{r-D}{\frac{1}{2}\sigma^2} - 1)v_x - \frac{r}{\frac{1}{2}\sigma^2}v = 0.$$

Denote the term $(r-D)/\frac{1}{2}\sigma^2$ by k, and $D/\frac{1}{2}\sigma^2$ by d to obtain

$$v_\tau = v_{xx} + (k-1)v_x - (k+d)v = 0.$$

We can eliminate the terms v_x and v by introducing another variable change:

$$v = e^{-\gamma x - (\gamma^2+k+d)\tau} u(x,\tau).$$

Finally, we have

$$\frac{\mathrm{d}u}{\mathrm{d}\tau} = \frac{\mathrm{d}^2 u}{\mathrm{d}x^2} \tag{4.2}$$

with

$$u(x,\tau) = e^{\gamma x + (\gamma^2+k+d)\tau} V/K,$$

where $\gamma = \frac{1}{2}(k-1)$, $k = 2(r-D)/\sigma^2$, $d = 2D/\sigma^2$. Alternatively, we will write $\beta = \frac{1}{2}(k+1)$ so that $\gamma^2 + k = \beta^2$:

$$u(x,\tau) = e^{\gamma x + (\beta^2+d)\tau} V/K. \tag{4.3}$$

By this transformation, we are able to 'hide' the coefficients and most of the terms in the Black-Scholes equation and reveal the core dynamics of the equation.

As the mapping is one-to-one, we can concern ourselves with solving the equation 4.2 for $u(x,\tau)$ instead of finding $V(S,t)$ from the full Black-Scholes PDE. $u(x,\tau)$ is not measured in any units of currency: u is a scalar.

Once $u(x,\tau)$ is solved, we can recover $V(S,t)$ at each point (x,τ) by the inverse mapping

$$V(Ke^x, T - \tfrac{1}{2}\sigma^2\tau) = Ku(x,\tau)e^{-\gamma x - (\beta^2+d)\tau} \tag{4.4}$$

The domain of the values S of the asset, bounded by zero and theoretically unbounded above, when transformed by $x = \log(S/K)$ corresponds to the interval $-\infty < x < \infty$.

The variable τ is defined by the equation

$$\tau = (T-t)\tfrac{1}{2}\sigma^2,$$

so we have effectively reversed the course of time. The expiry T corresponds now to $\tau = 0$ and the present time $t = 0$ corresponds to $\tau = T\frac{1}{2}\sigma^2$. The scaling by $\frac{1}{2}\sigma^2$ makes τ a dimensionless variable: σ^2 is measured in units of *time*$^{-1}$, so the variable $\tau = (T - t)\frac{1}{2}\sigma^2$ is just a scalar denoting the 'dimensionless time' remaining until the expiry of the option.

By the choice of τ we have made the resulting diffusion equation a *forward* equation; the expiry of the option, which was denoted by T, corresponds now to $\tau = 0$, and the present time corresponds to $\tau = \frac{1}{2}\sigma^2$. Thus the payoff function in the (x, τ) coordinates corresponds to $u(x, 0)$.

Initial value problem. The solution $u(x, \tau)$ we are seeking is restricted by the payoff function $u(x, 0)$ of the option. We want to find a continuous function $u(x, \tau)$ which solves the diffusion equation (4.2) and at expiry equals the (transformed) payoff function $u(x, 0)$ of the option.

The (known) payoff function $u(x, 0)$ is the *initial condition* for solving the diffusion equation: initially, at $\tau = 0$, the function $u(x, \tau)$ is known, and we are interested in finding the values forward of the point $\tau = 0$ (up to $\tau = T\frac{1}{2}\sigma^2$, which equals the present time). The variable change $\tau = (T - t)\frac{1}{2}\sigma^2$ reversed the direction of time to make the equation a *forward equation*. This was discussed in section 2.1.3.

Going to the wrong direction can cause stability problems [4]: consider the diffusion equation with a coefficient s:

$$\frac{\mathrm{d}u}{\mathrm{d}\tau} = s\frac{\mathrm{d}^2u}{\mathrm{d}x^2}.$$

The independent solutions (*eigenmodes*) of the solutions satisfying the diffusion equation are of the form

$$u(x, \tau) = e^{ikx+\omega\tau},$$

where k is a real number and ω is a complex number that depends on k. Substituting $u(x, \tau) = e^{ikx+\omega\tau}$ into equation 4.2 gives us the relation $\omega = sk^2$.

It is now clear that if $s < 0$, $|u(x, \tau)| = e^{-sk^2\tau}$ grows without bound as $\tau \to \infty$. This may lead to unstable solutions: small errors in approximations may grow arbitrarily large. This happens if we move forward in time (from an initial condition) in a backward equation. We should thus solve the equation by moving backward in time.

In contrast, if $s > 0$, $|u(x, \tau)| = e^{-sk^2\tau}$ diminishes as $\tau \to \infty$; approximation errors in the initial data tend to get smaller.

In the Black-Scholes/diffusion equation transformation, the direction of time is reversed mainly for convenience. τ now conveniently denotes the dimensionless time till expiry.

4.1.1 The fundamental solution

Consider now the general diffusion equation [2]

$$\frac{\mathrm{d}u}{\mathrm{d}\tau} = \frac{\mathrm{d}^2u}{\mathrm{d}x^2}, \qquad -\infty < x < \infty, \quad \tau > 0.$$

with the initial condition

$$u_0(x) \equiv u(x, 0)$$

and the boundary conditions

$$u \to 0 \quad \text{as} \quad x \to \pm\infty.$$

The *fundamental solution* for this problem is

$$\phi(x,\tau) = \tfrac{1}{2\sqrt{\pi\tau}} e^{-x^2/4\tau} \tag{4.5}$$

which is a Gaussian curve. Its integral over all x for all $\tau > 0$

$$\int\limits_{-\infty}^{\infty} \phi(x,\tau)\ \mathrm{d}x = 1,$$

so it is also a normal probability density function. Its mean and variance are readily found to be 0 and 2τ, respectively. The solution can be thus described by a probability density with ever-increasing variance in time. The point of special interest should be $\tau = 0$. We find that

$$\lim_{\tau\to 0} \int\limits_{-\infty}^{\infty} \phi(x,\tau)\ \mathrm{d}x = 1.$$

$\tau = 0$ implies that the variance of the probability density function is zero. Zero variance and zero mean implies full certainty that the value of x is zero at $\tau = 0$.

Delta function. We are seeking a solution that would give us the complete initial solution at the limit $\tau = 0$. The limit of the fundamental solution at $\tau = 0$ is not a regular kind of function; however, this type of function is called a *Dirac delta function*, or *delta function* for short. This is commonly denoted by $\delta(x)$. Its most useful property is that, for any smooth function $f(\cdot)$

$$\int\limits_{-\infty}^{\infty} f(y)\delta(y)\,\mathrm{d}y = f(0)$$

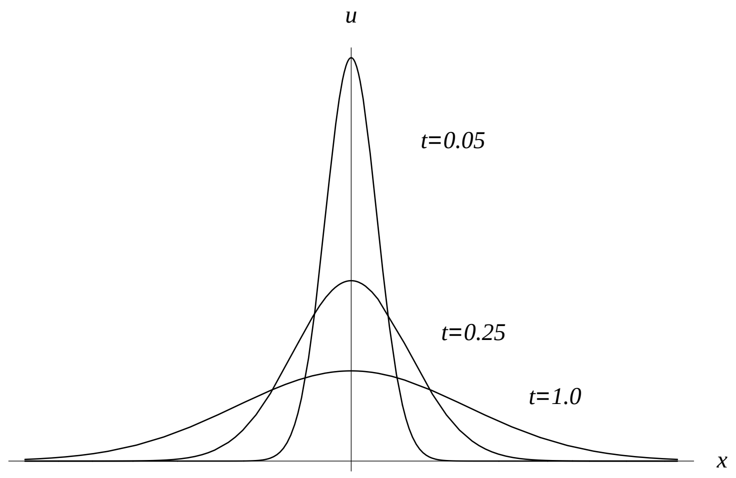


Figure 4.1: The Fundamental Solution for different τ. As $\tau \to \infty$, the curve 'flattens off'; initially, near $\tau = 0$ the height of the curve is arbitrarily close to infinity but always has an area below it equal to 1.

and

$$\int_{-\infty}^{\infty} f(y)\delta(y-x)\,\mathrm{d}y = f(x).$$

We will use this fact and the fact that the limit of the fundamental solution is a delta function to derive the solution.

The linearity of the diffusion equation guarantees that any linear combination of solutions are also solutions. Thus, for any fixed y_0

$$u(x,\tau) = u_0(y_0)\phi(y_0 - x, \tau)$$

is a solution. Since the integration operator is linear, also expressions like

$$u(x,\tau) = \int_{-\infty}^{\infty} u_0(y) f(y-x,\tau)\,\mathrm{d}y \tag{4.6}$$

are solutions. At $\tau = 0$, this solution satisfies the initial condition

$$u(x,0) = \int_{-\infty}^{\infty} u_0(y)\delta(y-x)\,\mathrm{d}y = u_0(x),$$

and it is continuous for all $\tau > 0$.

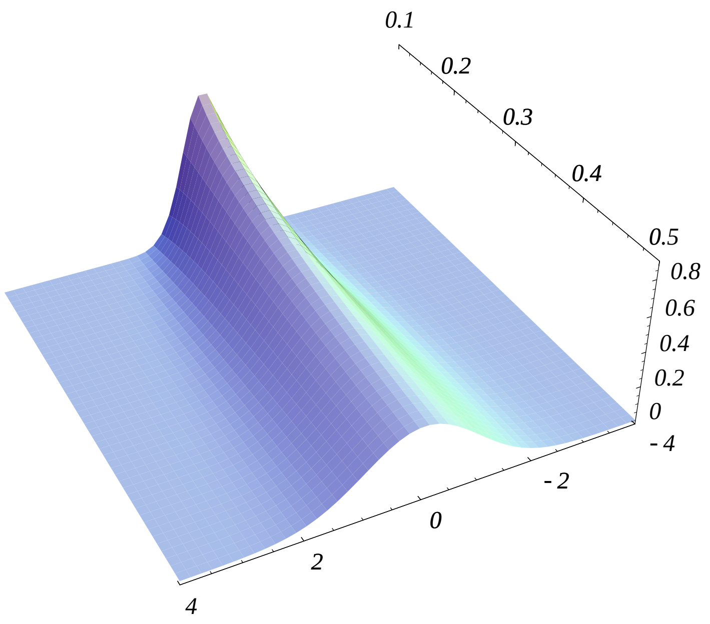


Figure 4.2: The Fundamental Solution for $\tau = 0.10 \ldots 0.50$. The graph shows how the solution evolves in time. The graph is truncated at points $x_{\min} = -4.0$ and $x_{\min} = +4.0$ but in reality the function is positive everywhere in $-\infty \leq x \leq +\infty$.

We have derived a solution that satisfies the initial condition. It can also be shown that this solution is unique The Black-Scholes formulas for puts 2.33 and calls(2.34) can be derived from this formula [2].

Figure (4.2) shows how the probability density of one single point of the initial condition $u_0(y_0)$ is spread over the x-axis as time passes. The formula (4.6) merges the densities of *all* these points together; the value of the option at any point is affected by the magnitude of each and every point of the payoff function.

We will use this generic solution to test the implementations of the algorithms.

4.2 Discretization of the plane

The idea of the finite difference methods is to divide the (x, τ)-plane into a sufficiently dense grid (or, *mesh*), and approximate the infinitesimal steps $\mathrm{d}x$ and $\mathrm{d}\tau$ by some small fixed finite steps δx and $\delta\tau$.

In the European option valuation problem we are given the values $V(S, t)$ of an option at some future timepoint $t = T$, and asked to find the option values at earlier timepoints down to the present time $t = 0$.

In the transformed (x, τ)-plane, this problem is equal to having the dimensionless option payoff $u_0(x)$ specified at time $\tau = 0$, and recovering the option values for the finite interval $(0, \frac{1}{2}\sigma^2] = (0, \tau_{\max}]$. The present time $t = 0$ corresponds to $\tau_{\max} = \frac{1}{2}\sigma^2$

The dimensionless asset price x is unbounded, running from $-\infty$ to $+\infty$. To make the problem feasible, the x-axis must be truncated at some points $x_{\min}$ and $x_{\max}$ in the hope that no significant errors are introduced. Naturally, the points for which we want to recover the option values must be included in this interval.

It is clear that the true solution is found only in the limit $x_{\min} \to \infty$ and $x_{\max} \to \infty$, so we will only be able to find an approximation by dropping off an infinite number of values at each end of the x-axis.

The interval $[x_{\min}, x_{\max}]$ should be large enough to include any important behavior of the payoff function. In the case of a call option, for example, one does not have to make $x_{\min}$ very small, since the payoff for all negative x-values (less than the strike) is zero, but should make $x_{\max}$ large enough to include the effect of the payoff function. The option values far away from nonzero payoff points tend to be negligible, if $\tau_{\max}$ is not very large, but the option value is zero *nowhere* unless the payoff function is zero. If $\tau_{\max}$ is large, we should take a sufficiently large spatial (x) interval, since the effects of the nonzero payoff points will propagate to the interval where the payoff originally was zero. With too narrow an interval, these values may be poorly approximated.

Boundary conditions. The points of interest in the interval $[x_{\min}, x_{\max}]$ are the boundary points $x_{\min}$,$x_{\max}$. Initially, at $\tau = 0$, the value of $u(x, \tau)$ at these points is known, since at $\tau = 0$ we know for certain $u(x, \tau)$ equals the payoff function $u_0(x)$. However, at timepoints $\tau > 0$, we must choose boundary conditions to represent the asymptotic behavior of the function $u(x, \tau)$ at $x \to \pm\infty$.

Alternatively [2], one can fix the boundary conditions based on the derivative $\partial u/\partial x$, but in this text we will set the boundary conditions based on the asymptotic behavior of the function u. Boundary conditions for the basic European options are discussed in section (4.8.2).

The boundary values $u_{\inf}$, $u_{\sup}$ may depend on τ. We thus need to $u(x_{\min}, \tau)$ and $u(x_{\max}, \tau)$ values corresponding to the limits

$$u_{\sup}(\tau) := \lim_{x \to +\infty} u(x, \tau) \qquad u_{\inf}(x_{\max}, \tau) := \lim_{x \to -\infty} u(x, \tau).$$

We thus have a rectangular region on the (x, τ) plane extending from $(x_{\min}, 0)$ to $(x_{\max}, \tau_{\max})$ with known values for $u(x, \tau)$ in the boundaries of x for all timepoints, and also all points $u(x, 0)$. These conditions give us the values of u on the 'three sides' of the (x, τ)-rectangle.

The goal is to recover values $u(x, \tau)$ for all $\tau > 0$ up to $\tau_{\max}$, which corresponds the expiry of the option. These values will all be calculated using the knowledge of the payoff $u_0(x)$ and of the behavior of u in the extreme x-boundaries.

Having confined the domain of the problem into a finite rectangle, we now divide it into intervals of length $\delta\tau$ in the τ-direction [1] and into intervals of length δx in the x-direction. Let us adopt a shorthand notation

$$u_j^n \equiv u(j\delta x, n\delta\tau).$$

which will denote the value of u at grid coordinates (j, n).

The problem boils down to finding good approximations for the first- and second-degree partial derivatives, and then rearranging the diffusion equation such that the values $u(x, \tau + \delta\tau) = u_j^{n+1}$ can be solved either explicitly or implicitly, using matrix decomposition or iterative methods.

Once an appropriate finite-difference method is applied to an interval, we will have the solution computed at discrete points u_j^n only. Ideally, the point S_0 for which we want an accurate answer $V(S_0, 0)$ should be at some grid point $j = (\log(S_0/K))/\delta x$ to get the most accurate answer. Otherwise, we will just have to do a linear interpolation between two grid points.

It is of course possible to use a different change of variables, for example $S = S_0 e^x$ so that S_0 will always be at a grid point $x = 0$.

In the following discussion, we will write $x = j\delta x$, $\tau = n\delta\tau$, unless men-

[1]Throughout this text, $\delta x^2 \equiv (\delta x)^2$.

tioned otherwise. Also, we will use the following identities:

$$\begin{cases} j_{\min} &= x_{\min}/\delta x \\ j_{\max} &= x_{\max}/\delta x \\ J &= j_{\max} - j_{\min} + 1 \\ N &= \tau_{\max}/\delta\tau \end{cases}$$

4.2.1 Approximating the partial derivatives

The approximations for the partial derivatives $\partial u/\partial\tau$ and $\partial^2 u/\partial x^2$ are obtained from Taylor expansions.

Approximating the first derivative

If we expand $u(x, \tau + \delta\tau)$ in Taylor series,

$$u(x, \tau + \delta\tau) = u(x, \tau) + \frac{\partial u}{\partial\tau}\delta\tau + \tfrac{1}{2}\frac{\partial^2 u}{\partial\tau^2}\delta\tau^2 + O(\delta\tau^3),$$

we get an approximation for $\partial u/\partial\tau$ called the *forward difference*:

$$\frac{\partial u(x,\tau)}{\partial\tau} = \frac{u(x, \tau + \delta\tau) - u(x,\tau)}{\delta\tau} + O(\delta\tau) \approx \frac{u_j^{n+1} - u_j^n}{\delta\tau}. \tag{4.7}$$

If we correspondingly expand $u(x, \tau - \delta\tau)$,

$$u(x, \tau - \delta\tau) = u(x, \tau) - \frac{\partial u}{\partial\tau}\delta\tau + \tfrac{1}{2}\frac{\partial^2 u}{\partial\tau^2}\delta\tau^2 - O(\delta\tau^3).$$

we get the *backward difference*

$$\frac{\partial u(x,\tau)}{\partial\tau} = \frac{u(x,\tau) - u(x, \tau - \delta\tau)}{\delta\tau} + O(\delta\tau) \approx \frac{u_j^n - u_j^{n+1}}{\delta\tau}, \tag{4.8}$$

again accurate up to $O(\delta\tau)$. Subtracting these two expansions up will yield

$$u(x, \tau + \delta\tau) - u(x, \tau - \delta\tau) = 2\frac{\partial u}{\partial\tau}\delta\tau + O(\delta\tau^3).$$

Dividing by $2\delta\tau$, we get the *central difference*

$$\frac{\partial u(x,\tau)}{\partial\tau} = \frac{u(x, \tau + \delta\tau) - u(x, \tau - \delta\tau)}{2\delta\tau} + O(\delta\tau^2) \approx \frac{u_j^{n+1} - u_j^{n-1}}{2\delta\tau}, \tag{4.9}$$

where the first-order terms cancel each other, leading to a more accurate approximation (since $\delta\tau \ll 1$). Alternatively, we get the same result by taking the average of the forward and backward differences.

Approximating the second derivative

Taking the forward difference of u with relation to x gives

$$\frac{\partial u(x,\tau)}{\partial \tau} = \frac{u(x+\delta x,\tau) - u(x,\tau)}{\delta x} + O(\delta x)$$

Applying the backward-difference to this expression will yield

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x+\delta x,\tau) - 2u + u(x-\delta x,\tau)}{\delta x^2} + O(\delta x^2)$$

which is approximately

$$\frac{\partial^2 u}{\partial x^2} = \frac{u^n_{j+1} - 2u^n_j + u^n_{j-1}}{\delta x^2}$$

and accurate up to the order of $O(\delta x^2)$. This is called the *symmetric central-difference* approximation.

Replacing the first and second derivatives in the diffusion equation will result in a difference equation which gives an relation that we can use to approximate the solution $u(x,\tau)$. Different choices for the left and right sides yield schemes that may be more stable or more accurate than others.

When using a finite-difference grid, we encounter two kinds of problems: stability of the method and the accuracy of the method. Moreover, we would like to obtain a sufficiently accurate answer with as few computations as possible.

4.3 The Explicit Finite-Difference Method

Given the initial condition u_j^0, we want to find $u_j^1, \ldots, u_j^N$ for all $j_{\min} \leq j \leq j_{\max}$. It is possible to write an expression that gives the 'next value' $u(x, \tau + \delta\tau)$ in terms of the known values explicitly; hence the name *explicit method.*

The diffusion equation can be discretized by taking the forward-difference for the left side and the symmetric central difference for the right side at (x, τ), yielding

$$\frac{u_j^{n+1} - u_j^n}{\delta\tau} = \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\delta x^2}.$$

The forward difference (4.7) is accurate to $O(\delta\tau)$, so so the equality sign holds up to order $O(\delta\tau, \delta x^2)$. Rearranging terms, we write the above equation as

$$u_j^{n+1} = u_j^n + \frac{\delta\tau}{\delta x^2}(u_{j+1}^n - 2u_j^n + u_{j-1}^n).$$

The ratio $\delta\tau/\delta x^2$ appears frequently enough to justify labeling it: let us denote the coefficient $\delta\tau/\delta x^2$ by α:

$$u_j^{n+1} = (1 - 2\alpha)u_j^n + \alpha(u_{j+1}^n + u_{j-1}^n). \tag{4.10}$$

4.3.1 Implementation

The explicit method is easy to implement with any programming language capable of storing arrays of data; even a spreadsheet program can be used.

Using formula (4.10), we can calculate all of the values for the next time-point one by one. The discretized rectangular region can be stored as a matrix; however, since we have a method to calculate u_j^n one timestep at a time, we do not need to keep any 'old' values in memory. We only need one array to store the initial condition values u_j^0 and another to store the computed values u_j^1.

Initially, the array u_j^0 should contain the values of the transformed payoff function for each $x = j\delta t$, $j_{\min} \leq j \leq j_{\max}$. Examples of transformed payoff functions for European options are shown in section 4.8.1.

An example of the code in C++ is given in the appendix, figure (A.3).

4.3.2 Stability issues

The finite-precision arithmetic used in computers may cause problems [4]. As we derive the values for u_j^n, at each timestep $\delta\tau$ the rounding errors may be amplified. The independent solutions of the difference equations are of the form

$$u_j^n = e^{ikj\delta x + \omega n \delta\tau} = (e^{\omega\delta\tau})^n e^{ikj\delta x}$$

where k is a real number and ω is a number that depends on k. Let us write $\xi \equiv \xi(k) = e^{\omega\delta t}$, so the solution can be written $u_j^n = \xi^n e^{ikj\delta x}$. By the von Neumann stability analysis [4], the difference equation is unstable if $|\xi| > 1$. This can be seen easily if we write $u_j^{n+1} = \xi u_j^n$: if the magnitude of ξ is greater than 1, $u(x,\tau)$ tends to 'blow up' as time τ increases.

The explicit method is simple and obviously easily implementable, but its drawback is instability for certain values of α. Applying the von Neumann stability analysis, and substituting u_{j+m}^n for $\xi^n e^{ik(j+m)\delta x} = u_j^n e^{ikm\delta x}$, we get

$$\begin{aligned} \xi = 1 - 2\alpha\left(e^{-ik\delta x} + e^{ik\delta x} - 2\right) &= 1 - 4\alpha(\cos(k\delta x) - 1) \\ &= 1 - 4\alpha\sin^2(\tfrac{1}{2}k\delta x). \end{aligned} \tag{4.11}$$

From this it follows that $|\xi| > 1$ if

$$\alpha > \frac{1}{2\sin^2(\frac{1}{2}k\delta x)} \Longrightarrow \alpha > \tfrac{1}{2}.$$

This means that for a given x-step size δx, $\delta\tau$ cannot be larger than $\frac{1}{2}\delta x^2$, which restricts our "speed" when moving from $\tau = 0$ to $\tau = \tau_{\max}$, thus adding up to our computing time.

Especially, changing the size of δx by a factor of ζ results in having to multiply the number of timesteps by ζ^2. Doubling the number of x-steps (the shape of the rectangle unchanged) requires us to quadruple the number of τ-steps, multiplying the number of operations by a factor of 8.

4.4 The Fully Implicit Finite-Difference Method

The explicit method is easy to implement, but severely limited by its poor stability for some choices of $(\delta x, \delta\tau)$. Instead of representing the values u_j^{n+1} explicitly from u_j^n, we can represent them implicitly by using a backward difference for $\partial u/\partial\tau$. We will arrive at the following relation:

$$-\alpha u_{j-1}^{n+1} + (1+2\alpha)u_j^{n+1} - \alpha u_{j+1}^{n+1} = u_j^n,$$

which gives the solution for u_j^{n+1} implicitly in terms of u_j^n, hence called the *fully implicit method*. This is solvable because we have the boundary solutions fixed. If we write up the relation starting with $j = j_{\min}+1$ and ending with $j = j_{\max}-1$ on the right-hand side, we obtain a system of equations involving all the unknown variables and the known boundary variables $u_{j_{\min}}^{n+1}$, $u_{j_{\max}}^{n+1}$:

$$\begin{cases} -\alpha u_{j_{\min}}^{n+1} & + & (1+2\alpha)u_{j_{\min}+1}^{n+1} & - & \alpha u_{j_{\min}+2}^{n+1} & = & u_{j_{\min}+1}^{n} \\ -\alpha u_{j_{\min}+1}^{n+1} & + & (1+2\alpha)u_{j_{\min}+2}^{n+2} & - & \alpha u_{j_{\min}+3}^{n+1} & = & u_{j_{\min}+2}^{n} \\ \cdots \\ -\alpha u_{j_{\max}-3}^{n+1} & + & (1+2\alpha)u_{j_{\max}-2}^{n+1} & - & \alpha u_{j_{\max}-1}^{n+1} & = & u_{j_{\max}-2}^{n} \\ -\alpha u_{j_{\max}-2}^{n+1} & + & (1+2\alpha)u_{j_{\max}-1}^{n+1} & - & \alpha u_{j_{\max}}^{n+1} & = & u_{j_{\max}-1}^{n}. \end{cases}$$

In the initial state $\tau = 0$, all values u_j^n on the right-hand side are known. Writing

$$A = \begin{pmatrix} 1+2\alpha & -\alpha & \cdots & 0 & 0 \\ -\alpha & 1+2\alpha & -\alpha & \cdots & 0 \\ 0 & -\alpha & 1+2\alpha & -\alpha & \vdots \\ \vdots & \vdots & \ddots & \ddots & -\alpha \\ 0 & \cdots & 0 & -\alpha & 1+2\alpha \end{pmatrix} \tag{4.12}$$

we can move all constants to the right-hand side and write the system of equations as

$$A \begin{pmatrix} u_{j_{\min}+1}^{n+1} \\ \vdots \\ u_{j_{\max}-1}^{n+1} \end{pmatrix} = \begin{pmatrix} u_{j_{\min}+1}^{n} \\ \vdots \\ u_{j_{\max}-1}^{n} \end{pmatrix} + \alpha \begin{pmatrix} u_{j_{\min}}^{n+1} \\ 0 \\ \vdots \\ 0 \\ u_{j_{\max}}^{n+1} \end{pmatrix},$$

or, with a more compact notation,

$$A\vec{u}^{n+1} = \vec{b}^n. \tag{4.13}$$

This system has $J-2$ equations and as many unknowns, so there is a unique solution for u_j^{n+1} as long as the corresponding coefficient matrix A is nonsingular.

The matrix A has $1+2\alpha$ in the diagonal, and indeed, since in our case α is always positive, the product of the diagonal elements is nonzero and therefore the matrix can never be nonsingular.

An obvious way to solve this is to find the inverse matrix A^{-1}, but there are faster methods. Matrix inversion would take $O(J^2)$ operations, but the matrix A being tridiagonal makes it an easy task for the *LU decomposition algorithm*, which solves it in time of order $O(J)$.

The tridiagonal matrix A can be represented as a matrix product of a lower-diagonal matrix L and an upper-diagonal matrix U, so the problem (4.13) can be written as

$$L(U\vec{u}^{n+1}) = \vec{b}^n.$$

We will break this problem into two separate subproblems, first solving an intermediate vector $\vec{v}$:

$$\vec{v} = L^{-1}\vec{b}^n,$$

and then the actual target vector

$$\vec{u}^{n+1} = U^{-1}\vec{v}.$$

Since A is tridiagonal, L can be written as a matrix that has only ones on the diagonal and other nonzero values only on the subdiagonal. U has correspondingly nonzero values only on the diagonal and superdiagonal. Therefore, the two problems can be solved by simple back-substitution, avoiding the need for a general matrix inversion algorithm.

The actual decomposition needs to be done only once and the result can be stored in a simple one-dimensional array. To calculate $\vec{u}^{n+1} = U^{-1}L^{-1}\vec{b}^n$, we only need $O(2J)$ operations. If the size of the matrix (depending on J) or α changes, the decomposition needs to be redone.

The decomposition can be written[2]

$$A = LU = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ \lambda_1 & 1 & 0 & \cdots & 0 \\ 0 & \lambda_1 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_{J-3} & 1 \end{pmatrix} \begin{pmatrix} \delta_1 & \upsilon_1 & \cdots & 0 & 0 \\ 0 & \delta_2 & \upsilon_2 & \cdots & 0 \\ 0 & 0 & \delta_3 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \upsilon_{J-3} \\ 0 & \cdots & 0 & 0 & \delta_{J-2} \end{pmatrix} \tag{4.14}$$

Considering only the entries $\delta_1, \lambda_1, \upsilon_1$, we get the following identities:

$$\begin{aligned} \delta_1 &= A_{11} = 1 + 2\alpha \\ \upsilon_1 &= A_{12} = -\alpha \\ \lambda_1\delta_1 &= A_{21} = -\alpha \\ \lambda_1\upsilon_1 + \delta_2 &= A_{22} = 1 + 2\alpha \end{aligned}$$

This gives us a solution for the variables $\lambda_1, \upsilon_1, \delta_1$ and a formula to calculate δ_2. We can now reduce the dimension of the matrix by one and apply the same procedure to calculate δ_3, given δ_2. Therefore, given $\delta_1 = 1 + 2\alpha$, for $1 \le j \le J - 3$ we have

$$\begin{aligned} \upsilon_j &= -\alpha \\ \lambda_j &= -\alpha/\delta_j \\ \delta_{j+1} &= (1 + 2\alpha) - \lambda_j\upsilon_j = (1 + 2\alpha) - \alpha^2/\delta_j. \end{aligned}$$

Assume that we have computed δ_j. Since we also know $\lambda_j = -\alpha/\delta_j$, we can solve $L\vec{v} = \vec{b}^n$ for $\vec{v}$ by simple forward substitution. Starting from the identity $v_{j_{\min}+1} = b_{j_{\min}+1}$, we can write $\lambda_1 v_{j_{\min}+1} + v_{j_{\min}+2} = b_{j_{\min}+2}$, or

$$v_{j_{\min}+2} = b_{j_{\min}+2} + \alpha v_{j_{\min}+1}/\delta_1,$$

from which we can deduce that given $v_{j_{\min}+1} = b_{j_{\min}+1}$, then for $1 \le j \le J-2$:

$$v_{j_{\min}+j+1} = b_{j_{\min}+j+1} + \alpha\frac{v_{j_{\min}+j}}{\delta_j}.$$

Given that we have computed $v_{j_{\min}+j}$, we can solve $U\vec{u}^{n+1} = \vec{v}$ by backward substitution. Initially, we know that $\delta_{J-2}u^{n+1}_{j_{\max}-1} = v_{j_{\max}-1}$, or

$$u^{n+1}_{j_{\max}-1} = \frac{v_{j_{\max}-1}}{\delta_{J-2}}.$$

[2]The entry υ_j on the superdiagonal of the matrix U is 'upsilon' (υ), not 'v' (v). The elements υ_j and λ_j are indexed $1..J-3$ whereas the intermediate vector v below is indexed $v_{j_{\min}+1}..v_{j_{\max}-1}$.

Then for $j = 2 \dots J-3$ we have:

$$u^{n+1}_{j_{\max}-j} = \frac{v_{j_{\max}-j} + \alpha u^{n+1}_{j_{\max}-j+1}}{\delta_{J-j-1}}.$$

The array δ_j depends on α and the number of spatial points J. If these do not change, we only need to compute δ_j once; solving for $\vec{u}^{\,n+1}$ requires then only $O(2J)$ operations.

The LU decomposition is discussed in detail in Wilmott et al [2] and in Numerical Recipes [4].

LU decomposition is a direct and fast method to solve for the vector $\vec{u}_{n+1}$. However, there are other methods, namely iterative methods. In the case of European option pricing, iterative methods are inferior in the sense that they require more arithmetic operations and do not produce accurate results directly, but rather by convergence to the solution within certain error margin. An iterative method can be successfully used in the case of American option pricing. Such an algorithm is presented in chapter 5.

Stability. Applying the von Neumann stability analysis to the fully implicit finite-difference method yields

$$\begin{aligned} -\alpha(\xi e^{-ik\delta x} + \xi e^{ik\delta x}) + (1+2\alpha)\xi &= 1 \\ \xi &= \frac{1}{1 + 4\alpha \sin^2(\frac{1}{2}k\delta x)} \end{aligned} \tag{4.15}$$

so for all α, $|\xi| \leq 1$. The fully implicit finite-difference method is thus unconditionally stable.

4.5 The Crank-Nicolson method

The fully implicit method was stable for any $\alpha = \delta\tau/\delta x^2$, but it was only accurate up to $O(\delta\tau, \delta x^2)$. The *Crank-Nicolson method* has an accuracy of $O(\delta\tau^2, \delta x^2)$ and it is also stable for all α.

The Crank-Nicolson scheme is formed by taking the average of the explicit and implicit methods:

$$(1+\alpha)u_j^{n+1} - \tfrac{1}{2}\alpha(u_{j+1}^{n+1} + u_{j-1}^{n+1}) = (1-\alpha)u_j^n + \tfrac{1}{2}\alpha(u_{j+1}^n + u_{j-1}^n). \qquad (4.16)$$

We will show that the Crank-Nicolson scheme is accurate up to $O(\delta\tau^2, \delta x^2)$ by equating the central difference and the symmetric central difference at $u_j^{n+1/2} \equiv u(x, \tau + \delta\tau/2)$. Expanding u_j^{n+1} in Taylor series at $u_j^{n+1/2}$ yields

$$u_j^{n+1} = u_j^{n+1/2} + \tfrac{1}{2}\frac{\partial u}{\partial \tau}\delta\tau + O(\delta\tau^2); \qquad (4.17)$$

expanding u_j^n at $u_j^{n+1/2}$ gives

$$u_j^n = u_j^{n+1/2} - \tfrac{1}{2}\frac{\partial u}{\partial \tau}\delta\tau + O(\delta\tau^2). \qquad (4.18)$$

Taking the average of these equations yields

$$\tfrac{1}{2}(u_j^n + u_j^{n+1}) = u_j^{n+1/2} + O(\delta\tau^2).$$

Since the subscript j was arbitrary, we can write this for subscripts $j-1$, j, and $j+1$ as follows:

$$u_{j-1}^{n+1/2} - 2u_j^{n+1/2} + u_{j+1}^{n+1/2}$$
$$= \tfrac{1}{2}\left(u_{j-1}^n - 2u_j^n + u_{j+1}^n\right) + \tfrac{1}{2}\left(u_{j-1}^{n+1} - 2u_j^{n+1} + u_{j+1}^{n+1}\right) + O(\delta\tau^2).$$

The right-hand side is an average of two symmetric central differences centered at grid points n and $n+1$. Dividing by δx^2, we obtain the equality

$$\frac{\partial^2 u(x, \tau+\delta\tau/2)}{\partial x^2} = \tfrac{1}{2}\left(\frac{\partial^2 u(x, \tau+\delta\tau/2)}{\partial x^2} + \frac{\partial^2 u(x, \tau+\delta\tau/2)}{\partial x^2}\right) + O(\delta x^2, \delta\tau^2).$$

Now subtract the equations (4.17) and (4.18) to obtain the central difference approximation of $\partial u/\partial\tau$ centered at $(x, \tau + \delta\tau/2)$:

$$\frac{\partial u(x, \tau+\delta\tau/2)}{\partial \tau} = \frac{u_j^{n+1} - u_j^n}{\delta\tau} + O(\delta\tau^2) \qquad (4.19)$$

Hence the diffusion equation centered at $(x, \tau + \delta\tau/2)$

$$\frac{\partial u(x,\tau+\delta\tau/2)}{\partial\tau} = \frac{\partial^2 u(x,\tau+\delta\tau/2)}{\partial\delta x^2}$$

has a finite-difference approximation

$$\frac{u_j^{n+1} - u_j^n}{\delta\tau} = \frac{1}{2\delta x^2}\left(u_{j-1}^n - 2u_j^n + u_{j+1}^n\right) + \frac{1}{2\delta x^2}\left(u_{j-1}^{n+1} - 2u_j^{n+1} + u_{j+1}^{n+1}\right)$$

which is exactly the Crank-Nicolson scheme (4.16), and thus accurate up to $O(\delta x^2, \delta\tau^2)$.

Stability. Replacing α with $\frac{1}{2}\alpha$, from (4.11) and (4.15) we obtain immediately

$$\xi = \frac{1 - 2\alpha\sin^2(\frac{1}{2}k\delta x)}{1 + 2\alpha\sin^2(\frac{1}{2}k\delta x)}. \tag{4.20}$$

Thus $|\xi| \leq 1$ for all values of α. The Crank-Nicholson scheme is stable for all combinations of δt and δx^2, and accurate up to $O(\delta\tau^2 + \delta x^2)$.

4.5.1 Implementation

To understand how the Crank-Nicolson scheme works, let us denote the right-hand side of (4.16) by

$$u_j^{n+1/2} = (1-\alpha)u_j^n + \tfrac{1}{2}\alpha(u_{j+1}^n + u_{j-1}^n); \tag{4.21}$$

This is an explicit scheme for computing $u_j^{n+1/2}$ from u_j^n. This is equal to the left-hand side of the equation:

$$(1+\alpha)u_j^{n+1} - \tfrac{1}{2}\alpha(u_{j+1}^{n+1} + u_{j-1}^{n+1}) = u_j^{n+1/2}.$$

This is an implicit scheme for computing u_j^{n+1} from $u_j^{n+1/2}$. Thus, the Crank-Nicolson method is essentially a combination of the explicit and fully implicit methods: we take one half-step using the explicit method, and then take another half-step using the fully implicit method. Since we take only half-steps, we must use $\frac{1}{2}\alpha$ instead of α.

We already know how to compute $u_j^{n+1/2}$ from u_j^n by using the explicit method. Now let us see how to compute u_j^{n+1}. Using the same notation as in (4.13), we can write (4.16) for all points $j_{\min}+1 \le j \le j_{\max}-1$ as

$$A' \begin{pmatrix} u^{n+1}_{j_{\min}+1} \\ \vdots \\ u^{n+1}_{j_{\max}-1} \end{pmatrix} = \begin{pmatrix} u^{n+1/2}_{j_{\min}+1} \\ \vdots \\ u^{n+1/2}_{j_{\max}-1} \end{pmatrix} + \frac{1}{2}\alpha \begin{pmatrix} u^{n+1}_{j_{\min}} \\ 0 \\ \vdots \\ 0 \\ u^{n+1}_{j_{\max}} \end{pmatrix},$$

where the matrix A' is

$$A' = \begin{pmatrix} 1+\alpha & -\frac{1}{2}\alpha & \cdots & 0 & 0 \\ -\frac{1}{2}\alpha & 1+\frac{1}{2}\alpha & -\frac{1}{2}\alpha & \cdots & 0 \\ 0 & -\frac{1}{2}\alpha & 1+\frac{1}{2}\alpha & -\frac{1}{2}\alpha & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -\frac{1}{2}\alpha & 1+\frac{1}{2}\alpha \end{pmatrix}. \tag{4.22}$$

Alternatively, we can write this as

$$A'\vec{u}^{n+1} = \vec{b}^{n+1/2}. \tag{4.23}$$

As discussed in the previous section, the implicit vector $\vec{u}^{n+1}$ can be solved by using LU decomposition.

The computing algorithm for the fully implicit method can be modified to make it a Crank-Nicolson method by (1) using $\frac{1}{2}\alpha$ instead of α; (2) using $\vec{u}^{n+1/2}$ instead of $\vec{u}^{n}$ as the input vector for the LU decomposition algorithm. $\vec{u}^{n+1/2}$ must be first computed by the explicit method, and then input to the LU decomposition algorithm.

Figure (A.5) shows an implementation of the Crank-Nicolson method.

4.6 Testing and Calibration

Once algorithms have been implemented as program code, they need to be tested. We will present a method to verify the accuracy and validity of the implemented finite-difference algorithms using the fundamental solution for the diffusion equation.

To test the algorithms, we use a function $\nu(x,\tau)$, with initial values

$$\nu_0(x) = \phi(x,\lambda) = \tfrac{1}{2\sqrt{\pi\lambda}} e^{-x^2/4\lambda}$$

where $\phi(x,\lambda)$ is the fundamental solution to the diffusion equation calculated at $\tau = \lambda$. We also assume the boundary conditions $\nu \to 0$ for $x \to \pm\infty$. After a period of τ, the initial values evolve to

$$\nu(x,\tau) = \int\limits_{-\infty}^{\infty} \nu_0(y) f(y-x,\tau)\,\mathrm{d}y = \int\limits_{-\infty}^{\infty} \phi(y,\lambda) f(y-x,\tau)\,\mathrm{d}y$$

which, after integration, simplifies to an explicit solution

$$\nu(x,\tau) = \phi(x,\lambda+\tau) = \tfrac{1}{2\sqrt{\pi(\lambda+\tau)}} e^{-x^2/4(\lambda+\tau)}$$

which we could have guessed, since the solution evolved from a given set of initial values and boundary conditions is unique; the fundamental solution at $f(x,\lambda)$ can only evolve to $f(x,\lambda+\tau)$ during a period of τ.

Thus we have an explicit formula to calculate the exact values of the solution, which we will use to compare the approximations computed by the finite-difference methods.

4.6.1 Definition of error

Our test function $\nu(x,\tau)$ solves the diffusion equation $\nu_\tau = \nu_{xx}$. Given a set of initial values $\nu(x,0)$, we will let the system evolve for some given time τ_0. The correct values can be calculated directly by $\nu(x,\tau_0) = \phi(x,\lambda+\tau_0)$. Using some numerical method, we should be able to approximate the correct values so that if we let $\delta\tau \to 0$ and $\delta x \to 0$, we get smaller errors.

Let $u(j\delta x, n\delta\tau) \equiv u_j^n$ be the approximation obtained by a numerical method and define the error ε_n as the average absolute error at $\tau = n\delta\tau$:

$$\varepsilon_n = \frac{1}{(J-2)} \sum_{j=j_{\min}+1}^{j_{\max}-1} \left| u_j^n - \nu(j\delta x, n\delta\tau) \right|. \tag{4.24}$$

Here $J-2$ is the number of spatial points, excluding the two boundary points which are not computed.

Local and global errors. The accuracy of the method was expressed as the order of the error term. However, this was the order of the error for *one* timestep. Hence this error is called the *local error*.

Since the total time needed to reach the solution is fixed, $\tau_{\max} = \delta\tau N$, the order of the total error is N times the local error. For example, if a method is locally accurate up to the term $\delta\tau^2$, we have

$$N\,O(\delta\tau^2) = \frac{\tau_{\max}}{\delta\tau} O(\delta\tau^2) = O(\delta\tau)$$

if we keep δx fixed. This is called the *global error*.

4.6.2 Accuracy and convergence

The order of the error term describes the accuracy and the speed of convergence of the solution. For example, the Crank-Nicolson method is locally accurate to $O(\delta\tau^2)$ and it converges one order faster than the fully implicit and explicit methods as $\delta\tau \to 0$.

Our final goal is to find an acceptable balance between computing time and accuracy of the result. Given a certain accuracy for the option value $V(S,t)$, we may find corresponding bounds for the diffusion function $u(x,\tau)$.

Let $V^*(S,t)$ denote the correct (dollar) value of the option at (S,t), and $V(S,t)$ its approximation that we have computed. If ε is some small positive real number, say $\varepsilon = 10^{-6}$, let us define the desired accuracy as

$$\delta V = |V^*(S,t) - V(S,t)| < \varepsilon \quad \text{for all} \quad (S,t). \tag{4.25}$$

In practice, this would tell us that the price of a transaction of one million options would be off by about one dollar. $V(S,t)$ is transformed by

$$V(S,t) = Ke^{\gamma x + (\gamma^2+k+d)\tau} u(x,\tau),$$

where $u(x,\tau)$ is our approximation. If the real value of $u(x,\tau)$ is obtained by adding a correction term ξ, we can write

$$V^*(S,t) = Ke^{\gamma x + (\gamma^2+k+d)\tau}(u(x,\tau) + \xi).$$

δV is then equal to $Ke^{\gamma x + (\gamma^2+k+d)\tau}|\xi|$, which implies that

$$|\xi| < e^{-\log K - \gamma x - (\gamma^2+k+d)\tau}\varepsilon.$$

We are interested in finding the maximum allowed absolute value for ξ which satisfies the accuracy requirement (4.25). This amounts up to finding the minimum for the right-hand side of the inequality. If we are away from the spot price $x \neq 0$, a negative γ may cause the error to be amplified. This happens when $r < \frac{1}{2}\sigma^2$.

In practice, we are often not interested in finding the optimal set of parameters that would guarantee a given accuracy. Instead, knowing that as $\delta\tau \to 0$ and $\delta x^2 \to 0$, the errors will tend *uniformly* to zero, we can simply either add more timesteps (e.g. by halving $\delta\tau$) or more spatial points (e.g. by halving δx) to get smaller errors. However, since the errors depend on *both* $\delta\tau$ and δx, we cannot make one smaller without considering the other.

For example, making $\delta\tau$ smaller and smaller (by adding more and more timesteps) while keeping δx fixed will be eventually useless, as we will learn in the next section.

4.7 Comparison

4.7.1 Accuracy and Convergence

As shown analytically, the Crank-Nicolson method is accurate up to $O(\delta\tau^2, \delta x^2)$ while the explicit and fully implicit methods are accurate to $O(\delta\tau, \delta x^2)$. In practice, this means that increasing the number of timesteps —or, equivalently, making $\delta\tau$ smaller— as we keep $\tau_{\max}$ fixed will result in smaller errors when using the Crank-Nicolson (CN) scheme than using the others.

However, since the error terms contain *both* terms including δx^2 *and* $\delta\tau^2$, one must keep in mind that just making the timestep smaller and smaller while keeping δx fixed will eventually make $\delta\tau$ smaller than δx^2. If then $\delta\tau \ll \delta x^2$, making $\delta\tau$ smaller will not make the overall errors smaller since δx^2 will dominate the error terms of order $\delta\tau$ and higher.

Especially, if we make the mistake of keeping α fixed as we make the timestep smaller, the CN method does *not* converge faster than the other schemes, since for each fixed α, $\delta\tau = \alpha\delta x^2$ so even the CN method will converge like

$$O(\delta\tau^2, \delta x^2) = O(\delta x^4, \delta x^2) = O(\delta x^2)$$

when $\delta x \ll 1$.

In any case, the *accuracy* of the CN method will be higher than those of the two other methods for given $\delta\tau, \delta x$. We should thus, for example, set δx to a sufficiently small number and then try different values of discretizations in the τ-direction.

The convergence of these three methods is illustrated in the figure 4.3, where the logarithm of the error is plotted against the logarithm of $\delta\tau$. δx is set to a small value compared to $\delta\tau$ so it is clear from the picture what happens to the error if we make $\delta\tau$ smaller. We have used $\lambda = 0.25$ and truncated the x-axis at $x_{\min} = -10.0$ and $x_{\max} = +10.0$.

4.7.2 Execution speed

Since the actual execution speed varies between different computing platforms, we choose to compare rather the order (using the big-O notation) of the computer instructions required to reach the expiry of the option ($\tau_{\max}$). This naturally depends on the number of timesteps N and the number of spatial points J.

An implementation of the explicit finite-difference method typically requires $O(J)$ instructions per timestep, since there is only one set of operations per each vector component u_j^n.

The fully implicit finite-difference method requires a routine such as LU decomposition algorithm. The time required for the LU algorithm is $O(2J)$, so the fully implicit method takes $O(3J)$ operations per timestep.

This is about 3 times compared to that of the explicit method, but the explicit method has the handicap of being seriously constrained by the ratio $\alpha = \delta\tau/\delta x^2 \le \frac{1}{2}$. Thus if we fix δx to a small value, we may be required to take several times more timesteps than using the implicit method. The implicit method is clearly preferable to the explicit method.

The Crank-Nicolson method using the *LU* decomposition algorithm requires extra $J-2$ loops for each timestep, thus the time required is of order $O(4J)$ per timestep.

This does not mean that the CN method is necessarily the slowest method. Since the goal is to attain a desired accuracy, the CN may require only a fraction of the timesteps required by the other methods. To increase accuracy, we only need to find sufficiently small $\delta\tau$ and δx. Once δx is fixed to a sufficiently small value, the explicit method is constrained by $\alpha \le \frac{1}{2}$, i.e.

$$N \ge 2\frac{\tau_{\max}}{\delta x^2}. \tag{4.26}$$

4.7.3 Stability

To demonstrate the stability of the algorithms already discussed earlier and proven analytically, we have calculated the errors for each of the three methods using different combinations of N (number of timesteps) and J (number of spatial points) on given $\tau_{\max}$ and truncation points $x_{\min}$ and $x_{\max}$.

Fixing $\tau_{\max} = 1.0$ and $x_{\min} = -10.0$, $x_{\max} = +10.0$, we will calculate the error for combinations of (N, J) for $N = 10..250$ and $J = 10..500$. This of course fixes $\delta\tau, \delta x$ and consequently α. Since $\delta\tau = \tau_{\max}/N$, $\delta x = (x_{\max} - x_{\min})/J$, for a given $\alpha = \alpha_0$ we have

$$\alpha_0 = \frac{\delta\tau}{\delta x^2} = \frac{\tau_{\max}}{(x_{\max} - x_{\min})^2} \cdot \frac{J^2}{N} \iff N = CJ^2,$$

where C depends on $\tau_{\max}, x_{\max}, x_{\min}$, and α_0. Thus α is constant on the parabolic lines $N = CJ^2$.

The Explicit method

We can see in figure (4.4) that there is a parabolic boundary beyond which the explicit finite-difference method is unstable. This boundary corresponds to $\alpha = \frac{1}{2}$. This is especially clear in Figure 4.5 where the all white region to the right corresponds to values where $\alpha > \frac{1}{2}$. Elsewhere, black regions indicate high errors and white regions indicate low errors.

The convex shape of the region where errors are the lowest as seen in Figure 4.4 tells that there is possibly some value of α where the error might attain its lowest value. Indeed, if we consider the explicit finite-difference method once more but write out the next error terms, we get

$$\frac{u_j^{n+1} - u_j^n}{\delta\tau} = \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\delta x)^2} + \tfrac{1}{2}\frac{\partial^2 u}{\partial \tau^2}\delta t - \frac{1}{12}\frac{\partial^4 u}{\partial x^4}\delta x^2 + O(\delta\tau^2, \delta x^4).$$

Or, ignoring the partial derivatives but only considering the order of the error:

$$\frac{u_j^{n+1} - u_j^n}{\delta\tau} = \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\delta x)^2} + O(\tfrac{1}{2}\delta t - \frac{1}{12}\delta x^2, \delta\tau^2, \delta x^4),$$

we can make the average error smaller by taking

$$\tfrac{1}{2}\delta t - \frac{1}{12}\delta x^2 \ = 0 \iff \alpha = \frac{1}{6}.$$

The parabolic curve where $\alpha = 1/6$ can be clearly seen in the contour plot (4.5).

The Fully implicit method

The stability of the fully implicit method can be seen in Figures 4.6 and 4.7. There are no regions where the error goes out of the scale.

It is very clear especially from Figure 4.7 that the denser the discretization, the smaller errors we get.

The Crank-Nicolson method

As the fully implicit method, the Crank-Nicolson (CN) method is stable everywhere. There are no regions where the error goes out of the scale.

Comparing the figures 4.6 and 4.8, we can see that the CN method converges much faster, i.e. produces smaller errors with fewer timesteps. In fact, it can be seen that increasing timesteps may be not necessary; since the method is convergent at the rate $O(\delta\tau^2, \delta x^2)$, making timesteps smaller than δx will make the error terms with δx^2 dominate the errors, and the gain is lost.

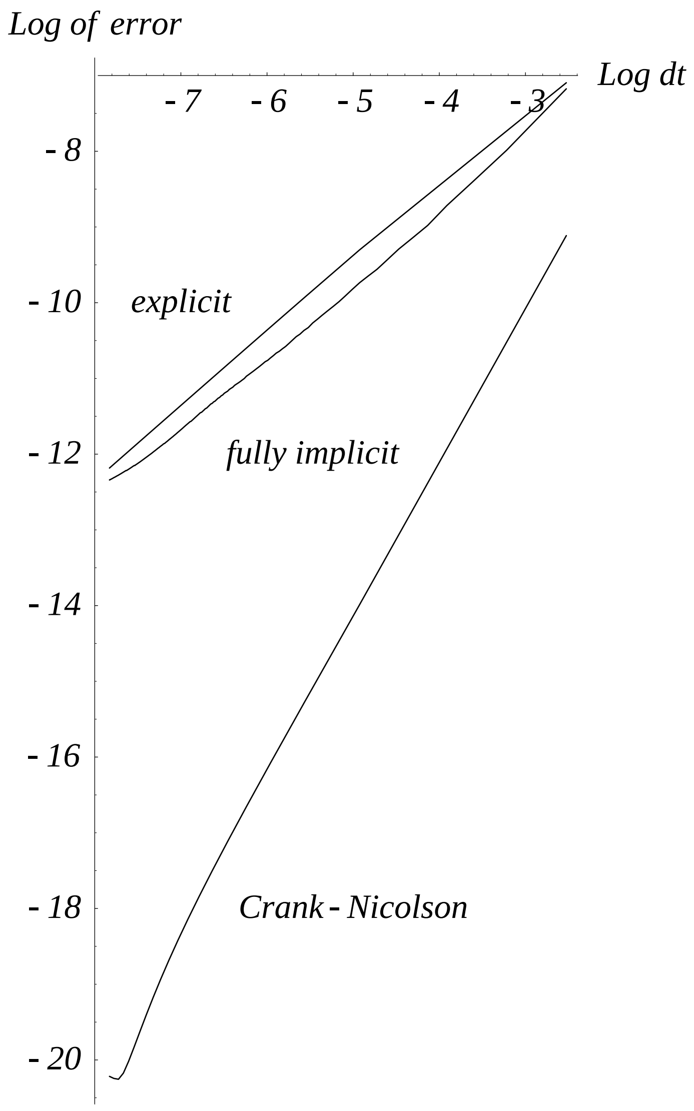


Figure 4.3: Convergence of the three finite-difference methods with $\delta x \approx 0.001$, $\tau_{\max} = 1.0$. The Crank-Nicolson method converges one order faster than the other two methods.

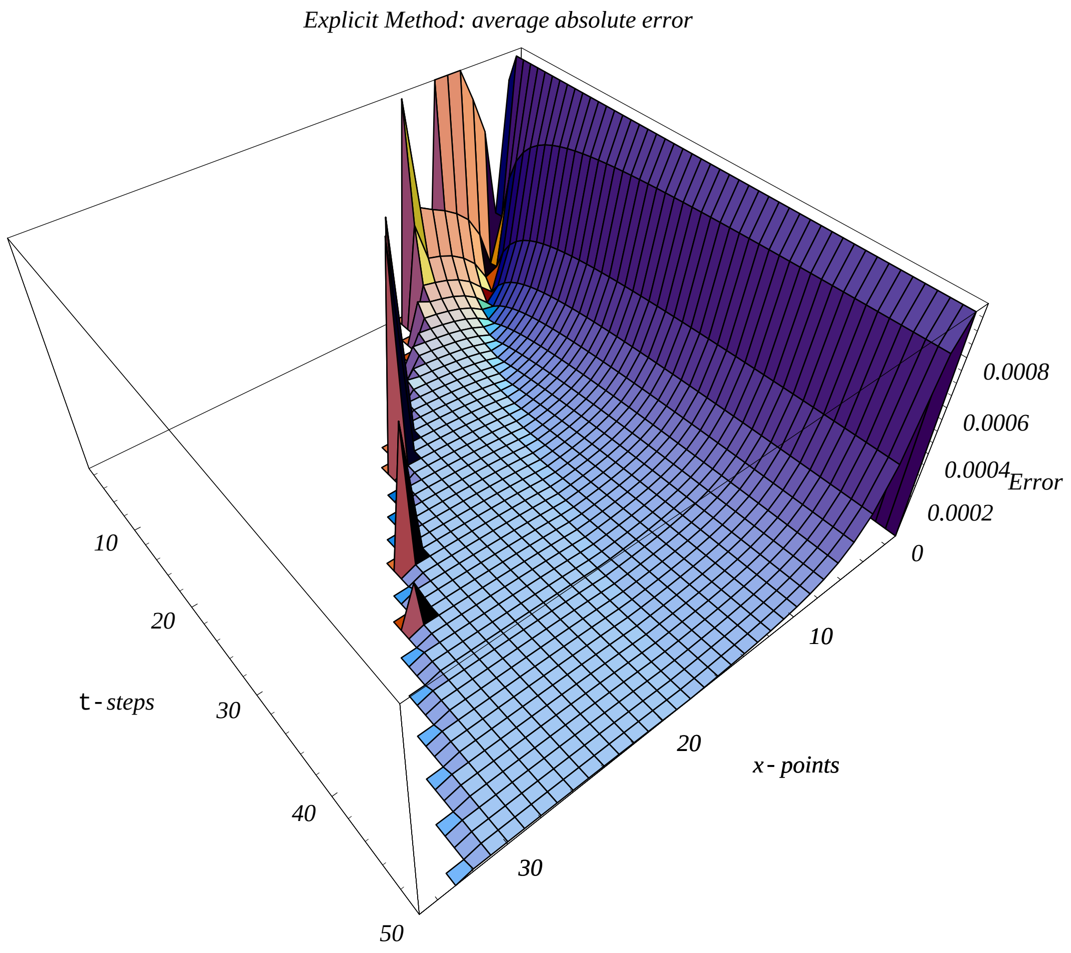


Figure 4.4: The explicit finite-difference method: errors when keeping $\tau_{\max} = 1.0$ fixed and varying the number of timesteps and spatial points. The errors on the left of the graph are out of the scale and thus not shown. The number of timesteps changes from 10 to 500 while the number of spatial points changes from 10 to 350.

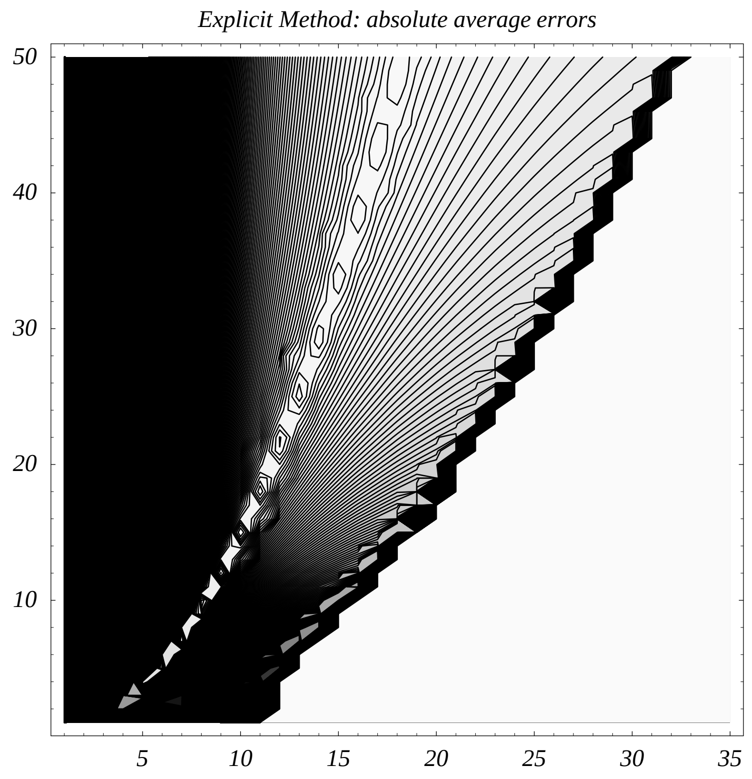


Figure 4.5: The explicit finite-difference method: contour plot of the errors when keeping $\tau_{\max} = 1.0$ fixed and varying the number of timesteps and spatial points. The N-axis (number of timesteps/10) runs vertically and J (number of spatial points/10) horizontally. The empty white space on the right represents the region where the errors are out of the scale; elsewhere whiter areas represent smaller errors. The boundary separating the two areas corresponds to $\alpha = \frac{1}{2}$. For $\alpha > \frac{1}{2}$, the explicit method is unstable and makes the rounding errors grow without bound. The smallest errors are concentrated on the parabolic curve which corresponds to the value $\alpha = \frac{1}{6}$, i.e. $\delta\tau = \frac{1}{6}\delta x^2$.

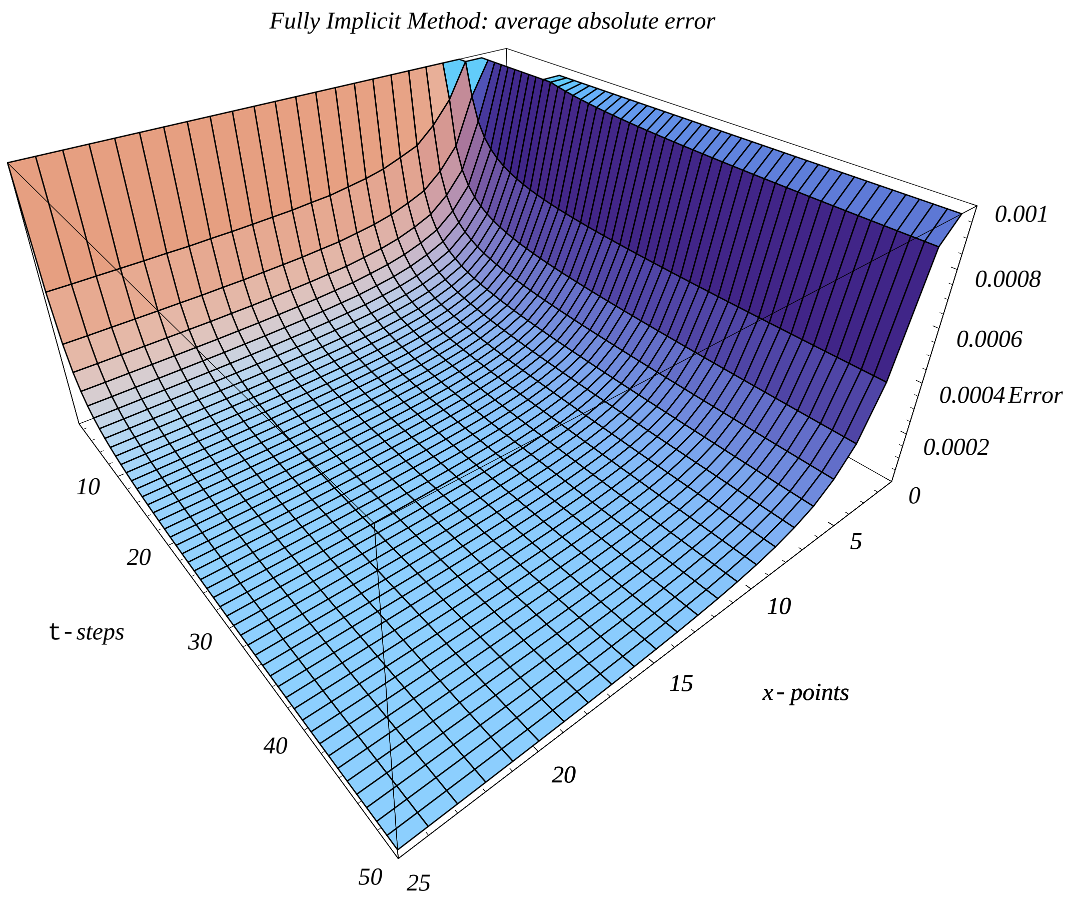


Figure 4.6: The fully implicit finite-difference method: errors when keeping $\tau_{\max} = 1.0$ fixed and varying the number of timesteps and spatial points. The number of timesteps changes from 10 to 500 while the number of spatial points changes from 10 to 250. The method is stable for any combination of (N, J).

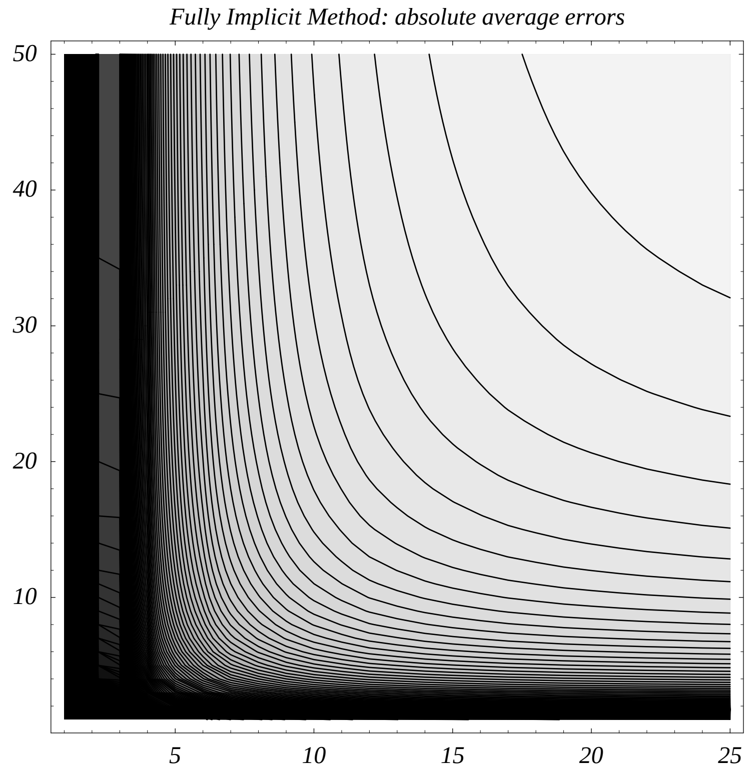


Figure 4.7: The fully implicit finite-difference method: contour plot of the errors (4.6) when keeping $\tau_{\max} = 1.0$ fixed and varying the number of timesteps and spatial points. Whiter areas represent smaller errors. The N-axis (number of timesteps/10) runs vertically and J (number of spatial points/10) horizontally. Making both N and J larger will increase the accuracy. Keeping one of the variables fixed while increasing the other results in marginal gains; sometimes the error actually rises. This is due to the fact that the error depends on both $\delta\tau$ and δx^2.

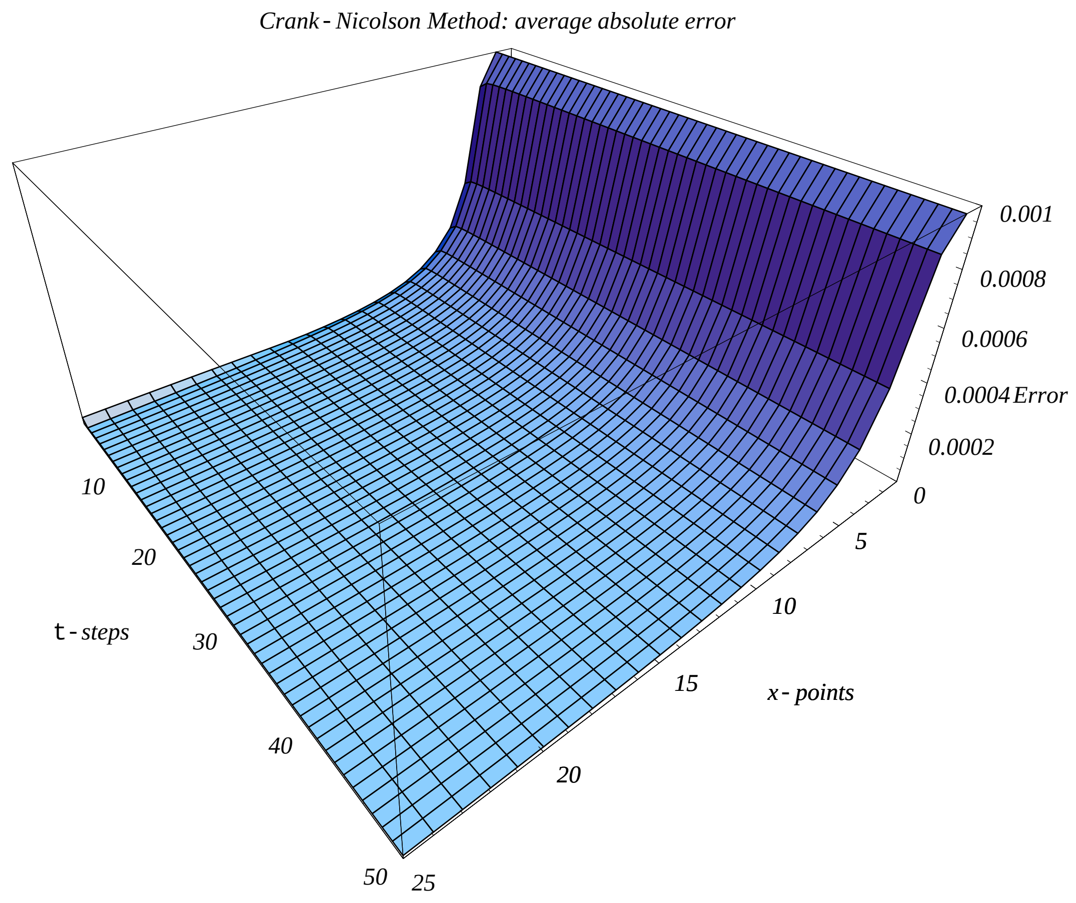


Figure 4.8: The Crank-Nicolson finite-difference method: errors when keeping $\tau_{\max} = 1.0$ fixed and varying the number of timesteps and spatial points. The number of timesteps changes from 10 to 500 while the number of spatial points changes from 10 to 250. The method is stable for any combination of (N, J).

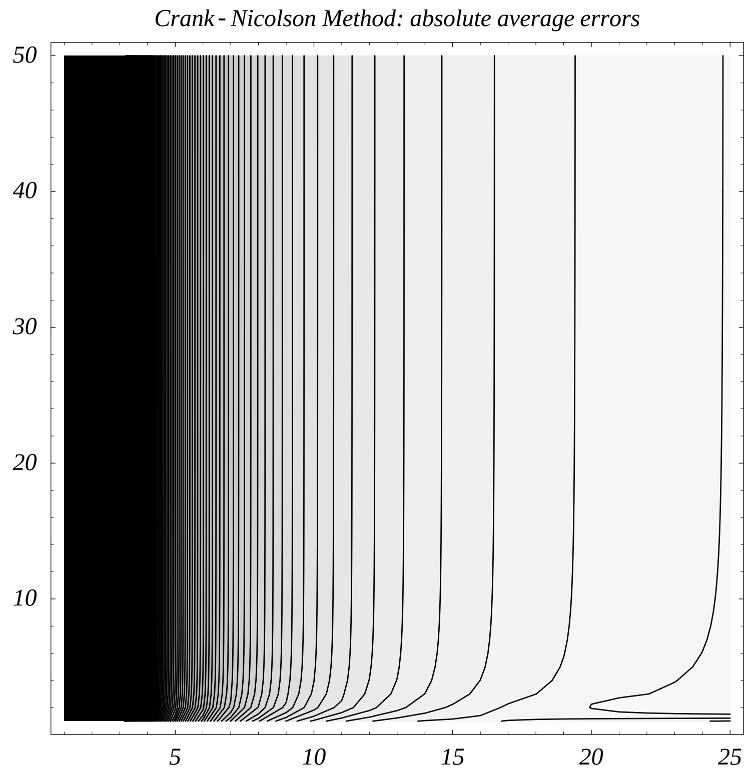


Figure 4.9: The Crank-Nicolson finite-difference method: contour plot of the errors (4.8) when keeping $\tau_{\max} = 1.0$ fixed and varying the number of timesteps and spatial points. Whiter areas represent smaller errors. The N-axis (number of timesteps/10) runs vertically and J (number of spatial points/10) horizontally. It is obvious that if we only make the number of timesteps N larger (decrease $\delta\tau$) while keeping δx fixed, the errors will not decrease dramatically. This is due to the order of accuracy $O(\delta\tau^2, \delta x^2)$: if $\delta\tau \ll \delta x$, the errors represented by the terms including δx^2 dominate the overall errors.

4.8 Applications to European Options

We have successfully tested and calibrated our implementation of finite-difference algorithms to solve the diffusion equation. In this section, we will use a finite-difference method to price actual European options.

Everything we needed to know about convergence and accuracy we already have found out in the previous sections. We only need to transform the payoff functions and the boundary conditions in terms of the variables (x, τ).

In this section we will only use the Crank-Nicolson method (using the LU decomposition algorithm) to demonstrate the valuation of the basic European options.

4.8.1 Payoff function transformation

Using the transformation (4.1), the variable τ vanishes at expiry. The transformation to the (x, τ) plane from (S, t) plane is therefore simply

$$u = e^{\gamma x} V_T / K,$$

where $V_T = Kv$ is the payoff function. For example, the European put payoff function is $V_T = (K - S_T)_+ = K(1 - e^x)_+$. Therefore, for a European put, we have

$$u_0(x) = e^{\gamma x}(1 - e^x)_+ = (e^{\gamma x} - e^{\beta x})_+.$$

For a European call, $V_T = (S_T - K) = K(e^x - 1)$, so the transformed payoff function is

$$u_0(x) = e^{\gamma x}(e^x - 1)_+ = (e^{\beta x} - e^{\gamma x})_+.$$

The binary options yield a payoff that does not depend on S nor K. If we use the transformation (4.3), the function u will be scaled by the strike price K, so it will appear in the formulas for $u(x, \tau)$. For example, the Binary put payoff function scaled by $1/K$ is $V_T/K = 1/K = e^{-\log K}$ for $S_T < K$, or $x < 0$. Using the transformation for a European binary put, we get

$$u_0(x) = e^{-\log K + \gamma x} \quad \text{when} \quad S_T < K \iff x < 0.$$

For a European binary call,

$$u_0(x) = e^{-\log K + \gamma x} \quad \text{when} \quad S_T > K \iff x > 0.$$

Elsewhere, the binaries yield $u_0 \equiv 0$.

4.8.2 Boundary conditions

Setting the boundary conditions for the test curve ν was easy since for $x \to \pm\infty$, $\nu(x,\tau) \to 0$: both boundaries were zero everywhere. This is not the case for the calls, puts, and binaries. The boundary values depend on the variables x and τ.

European put and call options. From put-call parity (1.6) we can deduce that the put option is equivalent to a short position on a forward contract and a long position on a call option with the same parameters K, T. Then at time t before expiry we must have

$$P = Ke^{-r(T-t)} - S_t + C.$$

Consider the case when $S_t \to 0$. The price of a call option goes to zero:

$$C_{\min} = 0, \tag{4.27}$$

along with the price of the asset, giving us a boundary condition for the put as $S_t \to 0$:

$$P_{\min} = Ke^{-r(T-t)}. \tag{4.28}$$

As $S_t \to \infty$, the value of a put option goes to zero,

$$P_{\max} = 0, \tag{4.29}$$

while the value of a call option, by the put-call parity again, equals that of the corresponding forward contract:

$$C_{\max} = S_t - Ke^{-r(T-t)} \tag{4.30}$$

We must transform these boundary conditions into terms of (x,τ). The transformation is

$$u(x,\tau) = e^{\gamma x + (\gamma^2+k+d)\tau} V/K = e^{\gamma x + (\beta^2+d)\tau} V/K,$$

where V is the actual option value in monetary units. Since $(T-t) = \frac{1}{2}\sigma^2\tau$, $P_{\min} = Ke^{-k\tau}$, so the condition (4.28) is equivalent to

$$u_{\inf}(\tau) = e^{\gamma x_{\min} + (\gamma^2+k+d)\tau - k\tau} = e^{\gamma x_{\min} + (\gamma^2+d)\tau}, \tag{4.31}$$

and the call boundary condition (4.30) is equivalent to

$$u_{\sup}(\tau) = e^{\gamma x_{\max} + (\beta^2+d)\tau} \left(e^{x_{\max}} - e^{-k\tau}\right) \tag{4.32}$$

To summarize, for the put we must have

$$\begin{aligned} u_{\text{inf}}(\tau) &= e^{\gamma x_{\min}+(\gamma^2+d)\tau} \\ u_{\text{sup}}(\tau) &= 0. \end{aligned} \tag{4.33}$$

For the call, the boundary conditions are

$$\begin{aligned} u_{\text{inf}}(\tau) &= 0 \\ u_{\text{sup}}(\tau) &= e^{\gamma x_{\max}+(\beta^2+d)\tau}\left(e^{x_{\max}} - e^{-k\tau}\right) \end{aligned} \tag{4.34}$$

Binary puts and calls. From the binary put-call parity (1.5) we find that at any time t before expiry we must have

$$P_B + C_B = e^{-r(T-t)}$$

When $S_t \to 0$, the price of a call option tends to zero, so we have

$$C_{\min} = 0. \tag{4.35}$$

When this happens, the put option tends to $e^{-r(T-t)}$, from where we get the boundary value for the put option:

$$P_{\min} = \lim_{S_t \to 0} P_B(S_t, t) = e^{-r(T-t)}, \tag{4.36}$$

which equals to $e^{-k\tau}$ when transformed. By the symmetry of the binary put-call parity, as $S_t \to \infty$,

$$C_{\max} = \lim_{S_t \to \infty} C_B(S_t, t) = e^{-r(T-t)} = e^{-k\tau}, \tag{4.37}$$

while the put option tends to zero: $P_{\max} = 0$.

Hence we obtain the following boundary conditions on the (x, τ) plane: for the binary put:

$$\begin{aligned} u_{\text{inf}} &= e^{\gamma x_{\min}+(\gamma^2+d)\tau} \\ u_{\text{sup}} &= 0. \end{aligned} \tag{4.38}$$

For the binary call, we obtain the symmetric result

$$\begin{aligned} u_{\text{inf}} &= 0 \\ u_{\text{sup}} &= e^{\gamma x_{\max}+(\gamma^2+d)\tau} \end{aligned} \tag{4.39}$$

If the boundary condition is set to the wrong value, the error propagates to other x-points. Figure (4.20) has the wrong value for the lower boundary: $u_{\text{inf}} \equiv 0$. Since both of the boundary conditions are set to zero, the function vanishes in time.

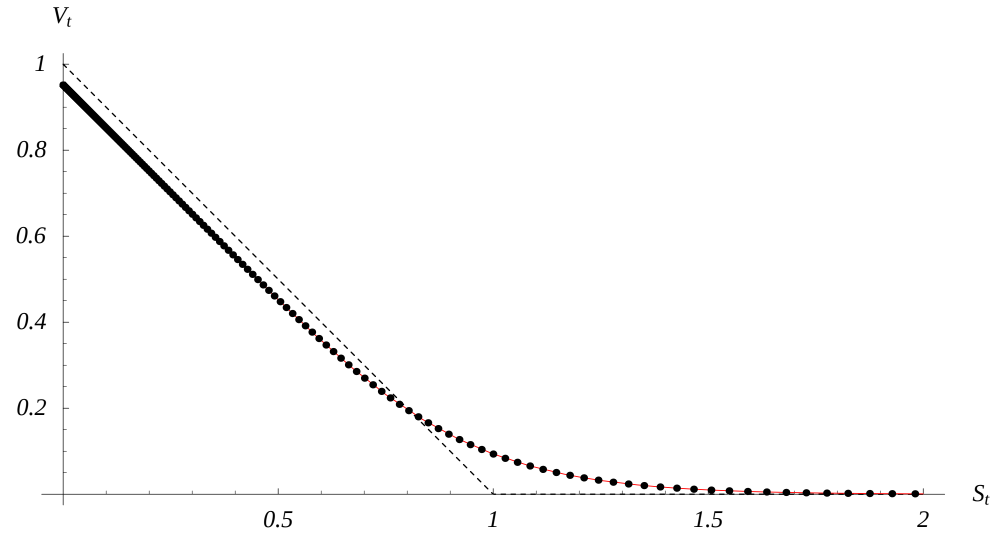


Figure 4.10: European put approximated by the Crank-Nicolson method (black dots) with the exact values from the Black-Scholes European put formula (continuous thin red curve). $K = 1.0$, $T = 1.0$, $\sigma = 0.30$, $r = 0.05$ The boundaries were set to $x_{\min} = -10.0$, $x_{\max} = +4.0$. Timesteps: $N = 32$, x-steps: $J = 512$.

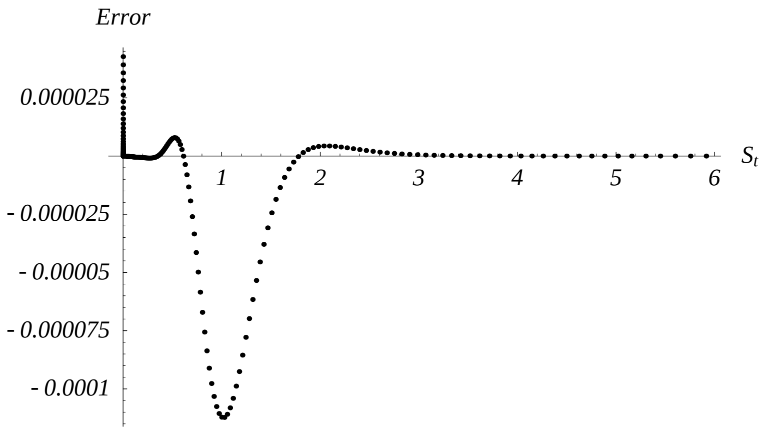


Figure 4.11: European Put: errors. The errors are the largest near the strike price (1.0), where $\partial V/\partial S$ is discontinuous. The difference of the lower boundary condition $u_{\inf}$ and the values $u(x_{\min}, \tau)$ causes the rising error near zero. The nearer $-\infty$ we draw the lower boundary, the smaller the error.

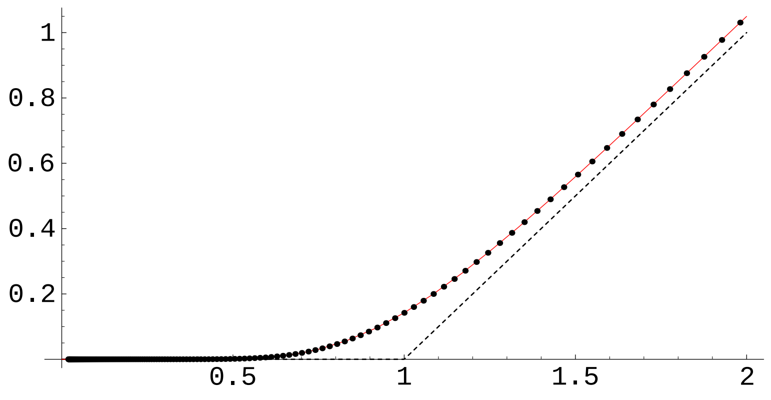


Figure 4.12: European call approximated by the Crank-Nicolson method (black dots) with the exact values from the Black-Scholes European put formula (continuous thin red curve). The underlying asset yields no dividends, $D = 0$. $K = 1.0$, $T = 1.0$, $\sigma = 0.30$, $r = 0.05$

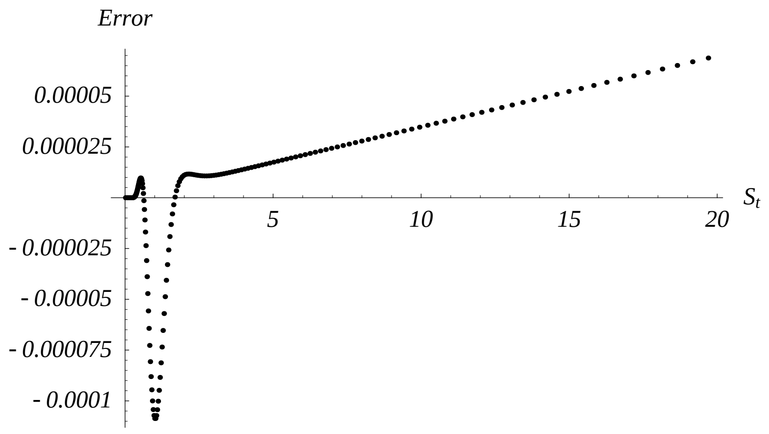


Figure 4.13: European Call: errors. The difference of the upper boundary condition u_{sup} and the values $u(x_{\min}, \tau)$ causes the rising error when $u \to \infty$. The nearer $+\infty$ we draw the upper boundary, the smaller the error. This is the same phenomenon as seen in figure (4.11), for $x \to \infty$ instead of $x \to -\infty$.

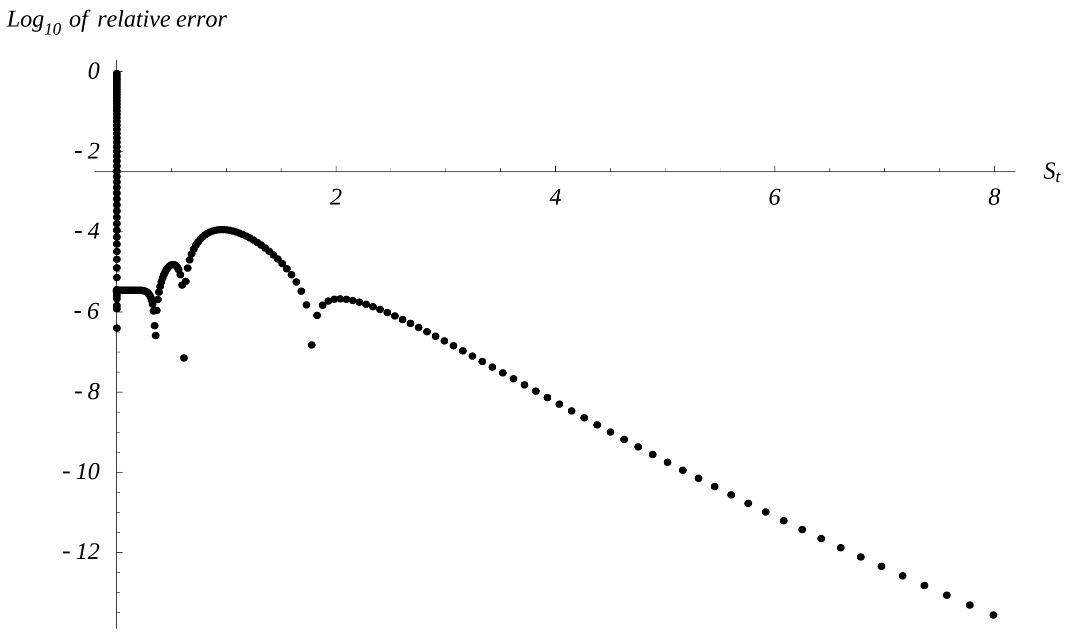


Figure 4.14: European Put: logarithm (base 10) of relative errors.

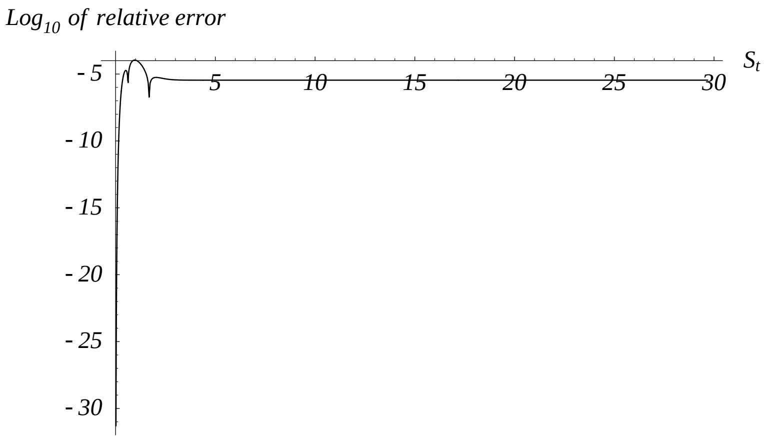


Figure 4.15: European Call: logarithm (base 10) of relative absolute errors.

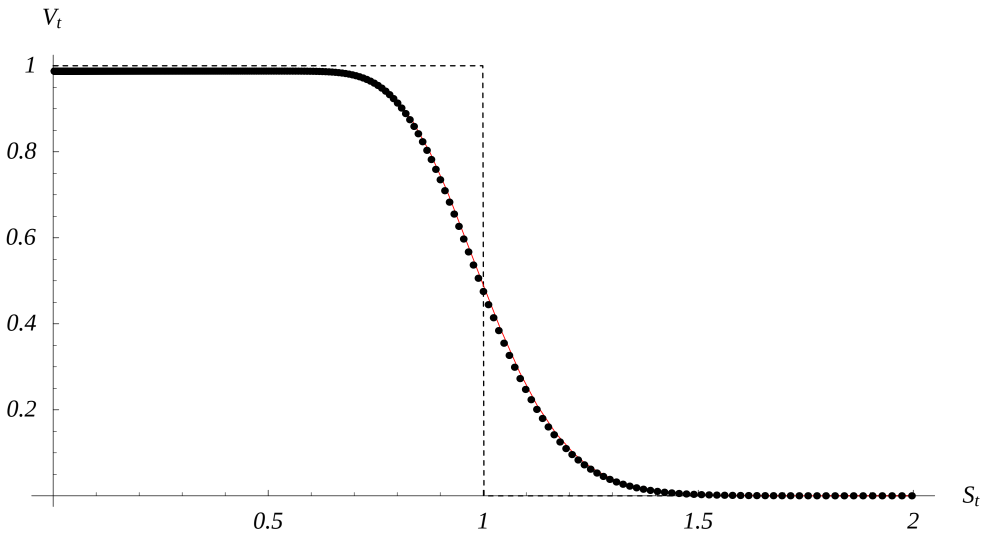


Figure 4.16: European Binary Put approximated by the Crank-Nicolson method (black dots) with the exact values from the Black-Scholes European binary put formula (continuous red curve). $K = 1.0$, $T = 0.25$, $\sigma = 0.30$, $r = 0.05$, $D = 0$.

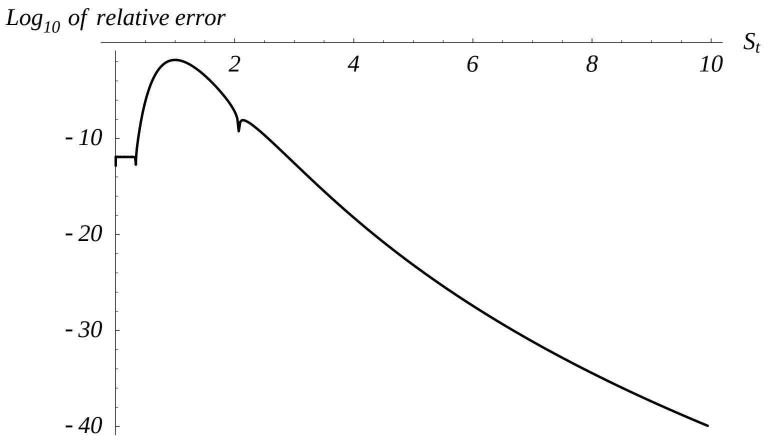


Figure 4.17: European Binary Put: logarithm (base 10) of relative errors.

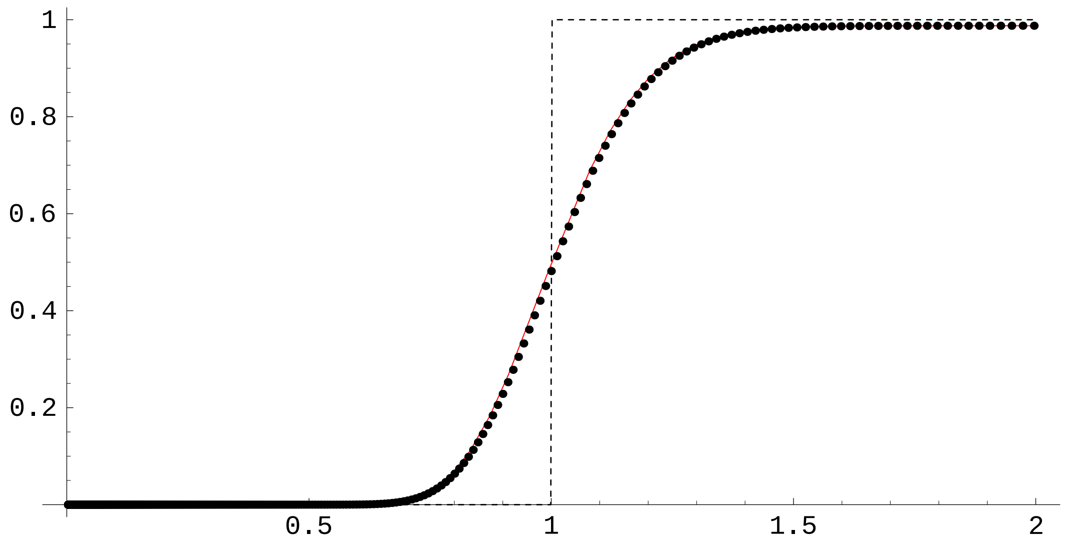


Figure 4.18: European Binary Call approximated by the Crank-Nicolson method (black dots) with the exact values from the Black-Scholes European binary put formula (continuous red curve). $K = 1.0$, $T = 0.25$, $\sigma = 0.30$, $r = 0.05$, $D = 0$.

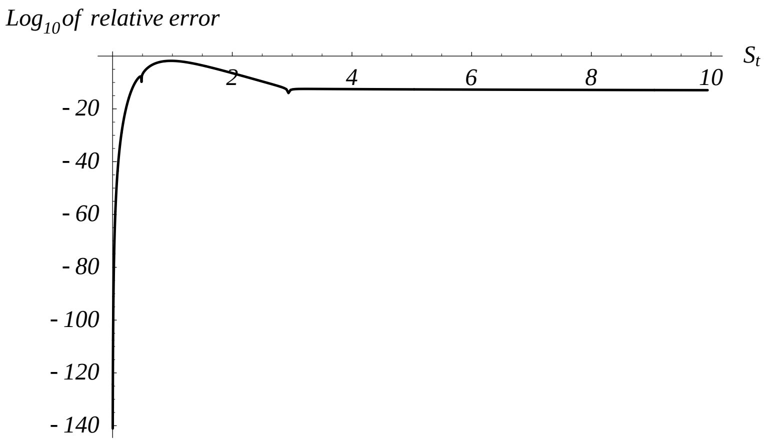


Figure 4.19: European Binary Call: logarithm (base 10) of relative errors.

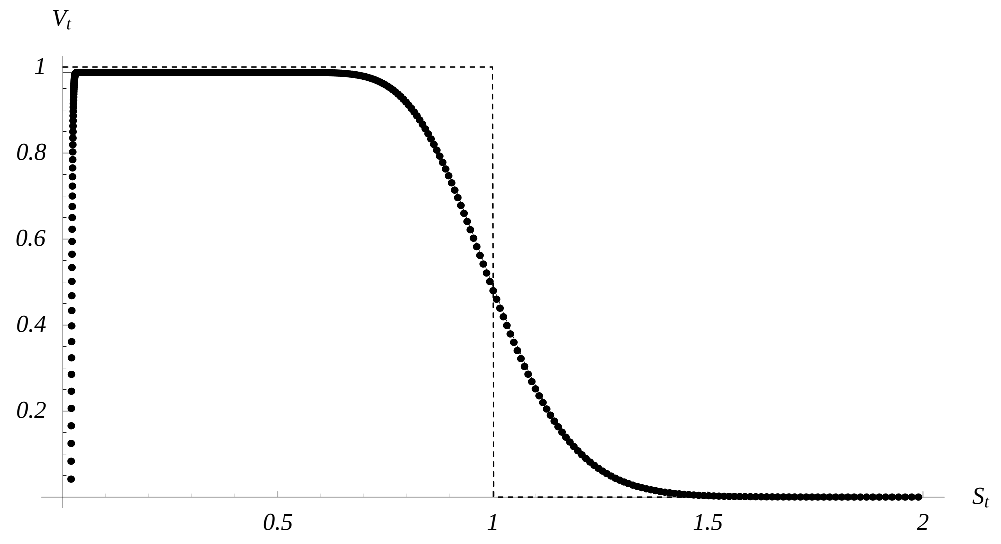


Figure 4.20: Example of an incorrect boundary condition for European Binary Put finite-difference approximation. The lower boundary was set to $u_{\text{inf}} \equiv 0$ instead of the correct boundary condition $u_{\text{inf}} = e^{\gamma x_{\min}+(\gamma^2+d)\tau}$. $K = 1.0$, $T = 0.25$, $\sigma = 0.30$, $r = 0.05$, $D = 0$.

Chapter 5

Pricing American Options

.

5.1 Introduction

An *American option* differs from the European option so that it can be exercised before expiry. The holder of the option has the right to decide whether to exercise or to hold the option, while the writer of the option cannot deny the holder's right to *early exercise.*

The basic American option can be exercised any time before expiry. However, there are variations to this. For example, a *Bermudan option* can be exercised only on certain pre-determined dates before expiry.

In general, analytical solutions for American options cannot be found, except in some rare cases. For example, if there is only one dividend payment during the lifetime of the option, an analytic solution called the Roll, Geske, and Whaley formula exists [5]. An American call option on a non-dividend paying asset can be valued simply by the European option formula. Also, there is an explicit solution to perpetual options, i.e. to options that have no expiry.

We will now consider the case of American options on underlying assets with dividend yield D on the underlying asset, possibly allowing $D = 0$.

Let us compare an American option V_A and a European option V_E written on the same underlying asset with the same strike price and the same expiry. Since the holder of such option can choose not to exercise the option early, it is obvious that the American option should be at least as valuable as the

European one:

$$V_A \geq V_E. \tag{5.1}$$

At each point of time before expiry, given the value of the asset S_t, the holder of an American option must decide whether it is more advantageous to hold the asset or exercise it.

If one exercises an American option, its value equals to the option payoff. For example, if we have a plain American put option, exercising at time t is worth

$$f_P(S_t, t) = (K - S_t)_+.$$

The payoff function may be time-dependent.

Let the payoff of an American option V_A be $f(S_t, t)$ at time t. A rational investor or trader will definitely choose to exercise an American option if the current value of the payoff exceeds that of the option: $f(S_t, t) > V_A(S_t, t)$. It is obvious that in a market with no arbitrage this cannot hold: buying the option and immediately exercising it would allow us to make riskless profit. If however $V_A(S_t, t) = f(S_t, t)$, one may want to exercise. Furthermore, if $V_A(S_t, t) > f(S_t, t)$, one prefers to hold the option. We can thus impose the first constraint: in the absence of arbitrage, we must have

$$V_A(S_t, t) \geq f(S_t, t). \tag{5.2}$$

At expiry T, the price of the option cannot exceed the payoff or else there would be an arbitrage opportunity for writing the option. Thus at expiry, (5.2) must become

$$V_A(S_T, T) = f(S_T, T). \tag{5.3}$$

American call. Consider an American and a European call options $V_{AC}(S_t, t)$ and $V_{EC}(S_t, t)$ with identical strike prices and expiry dates, written on an underlying with *no dividends*. By the put-call parity (1.6), such a European call V_{EC} is equal to a long forward contract plus a long European put option V_{EP} with the delivery date and price matching the expiry and the strike price. Since a long put is always nonnegative,

$$V_{EC} = (S_t - Ke^{-rT}) + V_{EP} \geq (S_t - Ke^{-rT})$$

Assuming that r is strictly positive, the value of the forward contract is strictly higher than the American call option payoff $f_C(S_t, t) = S_t - K$. Combining this with the condition (5.1), we obtain

$$V_{AC} \geq V_{EC} \geq S_t - Ke^{-rt} > S_t - K = f_C(S_t, t). \tag{5.4}$$

Thus $V_{AC} > f_C$ for all (S_t, t) and the holder of a such an American call option will choose *not* to exercise early. In the absence of arbitrage, the possibility that $V_{AC} > V_{EC}$ is ruled out, so the two options are of equal value: $V_{AC} = V_{EC}$.

American put. Consider an American and a European put options $V_{AP}(S_t, t)$, $V_{EP}(S_t, t)$ with identical strike prices and expiry dates; the underlying may or may not pay dividends. We can see from figure (2.3) that for some (S_t, t), the option payoff $f_P(S_t, t)$ is strictly higher than the corresponding European put option value $V_{EP}(S_t, t)$. By (5.2), we know that $f_P \leq V$ must hold, and by (5.1), we know that $V_E \leq V$. Therefore,

$$V_{EP} < f_P \leq V_{AP}$$

so the value of the American put must be different from the corresponding European put *for some* values (S_t, t). This shows that there are cases when early exercise is optimal.

Black-Scholes inequality. Assume now that we have an American option V and its European counterpart V_E, and suppose that $V \neq V_E$ for some (S_t, t). This is possible, as we saw in the case of an American put option. Since the value of V_E is governed by the Black-Scholes equation (2.8), for V this does not hold. Assume now the same setup as in section 2.1.2, so that the portfolio Π_0 contains a long American put option and Δ_0 units of the underlying. Also, allow the possibility that the asset yields continuously compounded dividends. The replicated option portfolio changes at a rate of

$$\frac{\partial V}{\partial t} + \tfrac{1}{2}\sigma^2 S_0^2 \frac{\partial^2 V}{\partial S^2}.$$

If this were a European put option, in the absence of arbitrage we would require that this equals $r\Pi_0$, the risk-free rate of change of a portfolio Π_0. Suppose now that the value of the option P_A grows at a rate higher than the risk-free rate. Then we can go short on the risk-free security and long on the option, exercise immediately and profit from the arbitrage opportunity. However, if the value of the option grows at a rate lower than the risk-free rate, we cannot assume that going short on the American will provide any arbitrage opportunities: the holder of the option will then exercise immediately and switch to the risk-free security instead. Thus to preclude arbitrage, we can only require that P_A do not grow faster than the risk-free security:

$$\frac{\partial V}{\partial t} + \tfrac{1}{2}\sigma^2 S_0^2 \frac{\partial^2 V}{\partial S^2} \leq r\Pi_0.$$

Therefore, in general, for American options in the absence of arbitrage we must have

$$\frac{\partial V}{\partial t} + \tfrac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r-D)S\frac{\partial V}{\partial S} - rV \leq 0. \tag{5.5}$$

Free-boundary problem. If the equality sign (5.5) holds, it signals that the value of V is equal to that of an European option, and it is not optimal to exercise. If the inequality sign holds, early exercise is optimal.

The conditions (5.2) and (5.5) must always hold. Moreover, since at each timepoint it is optimal *either* to exercise *or* to hold the option, we must have either

$$V(S_t,t) = f(S_t,t) \quad \text{or} \quad \frac{\partial V}{\partial t} + \tfrac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r-D)S\frac{\partial V}{\partial S} - rV = 0. \tag{5.6}$$

This is called a *free-boundary problem* in the framework of partial differential equations. The Black-Scholes equality holds in *some* region of the (S_t, t) plane, but not everywhere; other parts of the plane are governed by the equality $V = f$. We do not know in advance where the boundary between these regions is.

Continuity. So far it has not been discussed whether V or $\partial V/\partial S$ were continuous. We will only mention that the American option value is maximized by an exercise strategy that makes the option value V and $\partial V/\partial S$ continuous [6]. If the option value V were discontinuous, we would have an obvious arbitrage opportunity. A continuous first derivative means V must be smooth.

5.2 Binomial method

The binomial method makes American option valuation relatively easy. In a binomial tree, we have all possible values of (S_t, t) available; at each node we can make the decision of whether to exercise or not. The option value at that node is then the maximum of the two possible values: the risk-neutral expectation of the option price (if we hold) or the payoff (if we exercise).

The present value of the option is thus, much like in the case of European options, a sort of a risk-neutral expectation of the yield of the option. Now, it is not any more possible to use the expectation of final payoff alone; some binomial tree paths lead to cases when it is optimal to exercise early, while others lead to the end nodes.

The setup is the same as in the case of valuing European options. We will use the same recombining tree model with $n+1$ nodes at time $t = n\delta t$. Let S_j^n be the value of the option as defined in (3.1); V_j^n is the value of the option (3.2).

Let us denote the payoff function by $f(S_t, t)$, and introduce a shorthand notation for the payoff at node (j, n):

$$f_j^n \equiv f(S_0 u^j d^{n-j}, n\delta t), \quad 0 \le n \le N, \quad 0 \le j \le n. \tag{5.7}$$

For a plain American put or call option with strike K,

$$\begin{aligned} f_j^n &= (K - S_j^n)_+ &\quad& \text{for an American put} \\ f_j^n &= (S_j^n - K)_+ && \text{for an American call} \end{aligned}$$

At the end nodes (at expiry T), by the condition (5.3), the value of the option equals to the payoffs at expiry:

$$V_j^N = f_j^N. \tag{5.8}$$

Assuming appropriate parameters u, d, q, the previous values can be then calculated by the recursive formula

$$V_j^{n-1} = \max\left\{e^{-r\delta t}[qV_{j+1}^n + (1-q)V_j^n],\ f_j^{n-1}\right\}, \quad 0 \le j \le n-1. \tag{5.9}$$

There is no need to consider the free-boundary problem introduced earlier. This procedure leads automatically to the correct price of the option.

5.2.1 Implementation

The implementation of the binomial method for American options does not differ much from that for the European options. Obviously (5.9) takes more operations than (3.3). The American method has somewhat more complicated storage requirements, such as two-dimensional arrays (matrices).

The procedure again requires two nested loops, the outer for the $n = N-1..0$ and the inner for $j = 0..n$.

For each operation (5.9), we need to access V_j^n twice for $1 \le j \le n-1$ and once for $j = 0$ or $j = n$. Thus we need at least two one-dimensional arrays: one for storing the new values V_j^{n-1} and another containing the old values V_j^n. Since storage is cheap but time is expensive, it is not very practical to try to minimize storage requirements. An $N+1 \times N+1$ matrix `V[n][j]` is very convenient and makes the code easy to read—although the actual storage requirement of V_n^j is a triangular array.

S_j^n are needed for calculating f_j^n. During the procedure, each f_j^n is accessed exactly once, so it suffices to store S_j^n and call the payoff function $f_j^n \equiv f(S_j^n)$ whenever f_j^n is needed.

At the end of the procedure, the present value of the option is given by V_0^0.

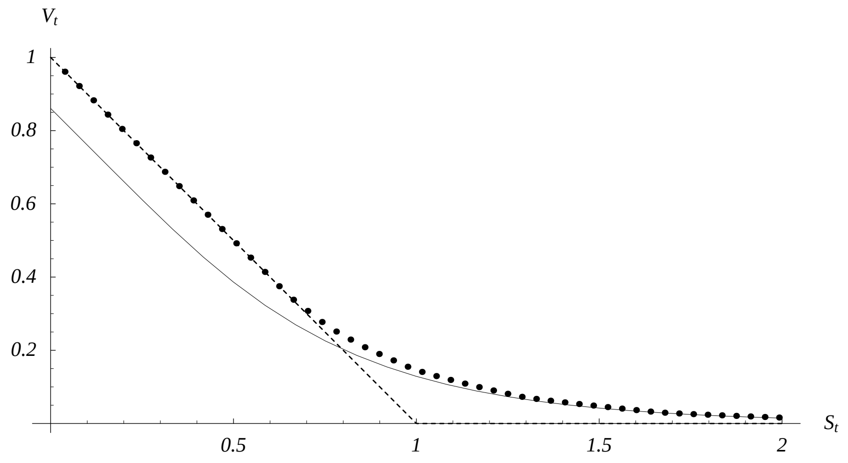


Figure 5.1: American Put calculated using the Binomial method (black dots). $\sigma = 0.30$, $r = 0.05$, $T = 3.0$. The value of the corresponding European put option is shown as a smooth curve (partly overlapped by the dots). The payoff function is shown as a dotted curve. The function joins smoothly into the payoff function.

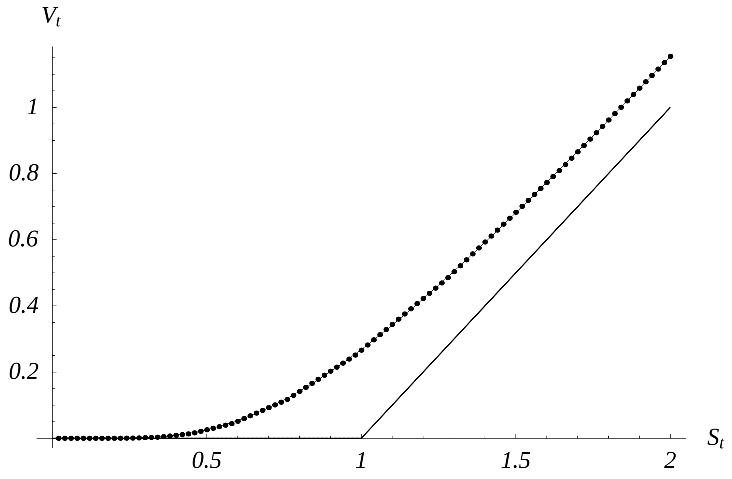


Figure 5.2: American Call calculated using the Binomial method (black dots). $\sigma = 0.30$, $r = 0.05$, $T = 3.0$. The value of the option is exactly the same as that of the corresponding European call option. The value of the corresponding European call option is shown as a smooth curve (overlapped by the dots).

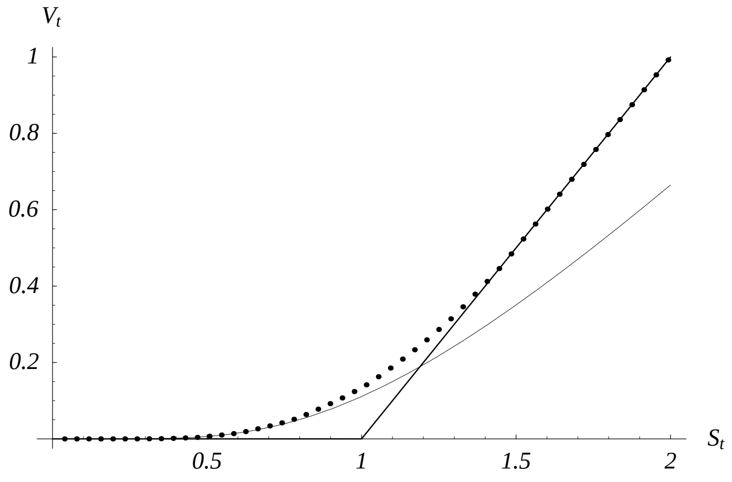


Figure 5.3: American Call with a dividend-paying underlying calculated using the Binomial method (black dots). $\sigma = 0.30$, $r = 0.05$, $T = 3.0$, $D = 0.10$. The payoff function is shown as a dotted curve. The value of the corresponding European call option, is shown as a smooth curve (overlapped by the dots). The payoff function is shown as a dotted curve.

5.3 Finite-difference methods

The finite-difference methods for European options attempt to solve the Black-Scholes partial differential equation. Now, however, we have two conditions (5.6) that make things slightly more difficult.

Explicit method. Wilmott [6] points out that the explicit finite-difference method is nothing but a "fancy version of the binomial method," which can be easily accommodated for American option valuation. Since the explicit method is constrained by the stability condition $\alpha \leq \frac{1}{2}$ and consequently by (4.26), it is not nearly as efficient as the implicit methods, so we choose not to consider it further.

Implicit methods. Both of the presented implicit methods provide better stability than the explicit method. As discussed in section (4.7), the Crank-Nicolson method is superior to the fully implicit method when it comes to accuracy and convergence, without much of a penalty in execution speed. We will consider only the Crank-Nicolson method.

Crank-Nicolson method. First, we need to put the conditions (5.6) in matrix and vector form to match our finite-difference grid. For convenience, let us denote all vector components by $\{1, \ldots, J-2\}$ instead of $\{j_{\min}+1, \ldots, j_{\max}-1\}$. Denote the payoffs at the grid point (n, j) by

$$g_j^n \equiv g(j\delta x, n\delta\tau). \tag{5.10}$$

where $g(x, \tau)$ is the payoff function $f(S_t, t)$ transformation in the (x, τ) plane. Let the vector $\vec{g}^n$ be the payoffs at time $t = n\delta\tau$.

The condition (5.2) corresponds to

$$\vec{u}^n \geq \vec{g}^n, \tag{5.11}$$

which is shorthand for $u_j^n \geq g_j^n$ for all j. The Black-Scholes inequality condition (5.5) corresponds to

$$A\vec{u}^{n+1} \geq \vec{b}^{n+1/2} \tag{5.12}$$

and the *linear complementarity condition* (5.6) corresponds to

$$(\vec{u}^n - \vec{g}^n) \cdot (A\vec{u}^{n+1} - \vec{b}^{n+1/2}) = 0. \tag{5.13}$$

As shown earlier, in the Crank-Nicolson (CN) method we take one half-step with the explicit method to find the values at timepoint $t = \delta\tau(n+1/2)$:

$$u_j^{n+1/2} = (1-\alpha)u_j^n + \tfrac{1}{2}\alpha(u_{j+1}^n + u_{j-1}^n).,\quad j_{\min}+1 \leq j \leq j_{\max}-1.$$

Then, we take another half-step to recover u_j^{n+1} using the fully implicit method:

$$(1+\alpha)u_j^{n+1} + \tfrac{1}{2}\alpha\left(u_{j-1}^{n+1} + u_{j+1}^{n+1}\right) = u_j^{n+1/2} \tag{5.14}$$

The values u_j^{n+1} to be solved for depend implicitly on *all* of the values u_j^n, for all j. The solution must thus hold for all u_j^{n+1} simultaneously.

In the case of European options, a fast and straightforward way to calculate the relation was to solve the simultaneous system of equations by LU decomposition, which in effect was just an efficient algorithm to invert a tridiagonal matrix.

Now, we have the problem of the possibility of early exercise to deal with. After taking the half-step and finding $u_j^{n+1/2}$, we must take into account that the payoffs g_j^{n+1} may yield higher values than the Black-Scholes values u_j^{n+1} that we are about to recover by solving the implicit system of equations.

This is not as easy as first finding the Black-Scholes solution u_j^{n+1} and then assigning u_j^{n+1} the value $\max\{u_j^{n+1},\ g_j^{n+1}\}$ for each j. Since all of the option values are interlinked by the implicit equation, this assignment must be made to work *simultaneously for all j*. The 'solution' must satisfy the constraints everywhere or the 'solution' is invalid [6].

Since u_j^{n+1} are not explicitly available, we do not have any values to compare the payoffs g_j^{n+1} with. Trying to find the indices j where the Black-Scholes price u_j^n and the payoff g_j^n hold ('tracking the free boundary') is not a particularly attractive method [2]. We will use a simple iterative method to compute the values.

5.3.1 Projected SOR Algorithm

The Projected Successive Over-Relaxation Algorithm is an *iterative method* to compute the values u_j^{n+1} from (5.14). By iteration, the method finds better and better estimates of the true solution; once a desired accuracy is obtained, the algorithm quits. It does *not* offer an *exact* solution such as matrix inversion or LU decomposition do.

Iterative methods. Recall that in the case of solving the value of an European option we had the problem

$$A\vec{u}^{n+1} = \vec{b}^{n+1/2}. \tag{5.15}$$

where A is tridiagonal with $1+\alpha$ in the diagonals and $-\frac{1}{2}\alpha$ in the sub- and superdiagonals. Now decompose A into a lower triangular matrix L, diagonal matrix D, and an upper triangular matrix U such that

$$A = L + D + U, \tag{5.16}$$

Substitute this into (5.15) to obtain an implicit equation for the vector $\vec{u}^{\,n+1}$:

$$\vec{u}^{\,n+1} = D^{-1}\left(\vec{b}^{n+1/2} - (L+U)\vec{u}^{\,n+1}\right).$$

Now let us use the notation

$$\vec{u}^{\,n+1,k}$$

to denote the kth *iterate* of $\vec{u}^{\,n+1}$ and set up an iteration algorithm

$$\vec{u}^{\,n+1,k+1} = D^{-1}\left(\vec{b}^{n+1/2} - (L+U)\vec{u}^{\,n+1,k}\right) \tag{5.17}$$

with

$$\vec{u}^{n+1,0} = \vec{u}^{\,n+1/2}. \tag{5.18}$$

This is called the *Jacobi method*. We start with an initial 'guess' $\vec{u}^{\,n+1/2}$ and use the formula (5.17) to compute the next guess. This involves computing the J components of the vector $\vec{u}^{\,n+1,k+1}$. We can now write each component $u_j^{n+1,k+1}$ explicitly as

$$u_j^{n+1,k+1} = D_{jj}^{-1}\left(b_j^{n+1/2} - \sum_{i=1}^{j-1} A_{ji}u_i^{n+1,k} - \sum_{i=j+1}^{J} A_{ji}u_i^{n+1,k}\right) \tag{5.19}$$

This method is known to converge to the correct solution for any value $\alpha > 0$ as $k \to \infty$ [2]:

$$\vec{u}^{\,n+1,k} \to \vec{u}^{\,n+1} \quad \text{as} \quad k \to \infty.$$

By convergence of the vector we mean that for some norm $\|\cdot\|$ we have

$$\|\vec{u}^{\,n+1,k+1} - \vec{u}^{\,n+1,k}\| \to 0 \quad \text{as} \quad k \to \infty.$$

There is a more efficient version of this algorithm. In the Jacobi method, we have to first calculate *all* values $\vec{u}^{\,n+1,k+1}$ before we can use them as new guesses for the next round. Since the newly computed vector components $u_j^{n+1,k+1}$ are better guesses than the old ones, $u_j^{n+1,k}$ the method would converge faster if we could use them as soon as possible.

The *Gauss-Seidel method* does just that: replace $u_i^{n+1,k}$ with $u_i^{n+1,k+1}$ in (5.19) for all $i < j$, i.e. whenever we already have a new iterate available.

This involves replacing the coefficients $u_i^{n+1,k}$ of the first sum (associated with matrix L) with $u_i^{n+1,k+1}$, since the matrix L is 'lagging behind' in the iteration process. The Gauss-Seidel method is expressed by

$$u_j^{n+1,k+1} = D_{jj}^{-1}\left(b_j^{n+1/2} - \sum_{i=1}^{j-1} A_{ji}u_i^{n+1,k+1} - \sum_{i=j+1}^{J-2} A_{ji}u_i^{n+1,k}\right) \tag{5.20}$$

When implementing the Gauss-Seidel method as a program, we notice that the old iterates can be overwritten as we compute the new ones; since we only need one array to store the iterates $u_j^{n+1,k+1}$, so the Gauss-Seidel method is even more memory-efficient than the Jacobi method.

Since the matrix A has few entries and even fewer distinct entries, we can write (5.20) in a very compact form:

$$u_j^{n+1,k+1} = \frac{1}{1+\alpha}\left(b_j^{n+1/2} + \tfrac{1}{2}\alpha\left(u_{j-1}^{n+1,k+1} + u_{j+1}^{n+1,k}\right)\right) \tag{5.21}$$

Successive Over-Relaxation. This is a refinement of the Gauss-Seidel method. The setup (5.20) is exactly the same, but we try to make the sequence $u_j^{n+1,k+1}$ converge faster.

The above methods usually converge to the correct solution from "one side" [6]; that is, if we obtain the next iterate $u_j^{n+1,k+1}$ by adding a correction term ϵ_j^k to it,

$$u_j^{n+1,k+1} = u_j^{n+1,k} + \epsilon_j^k,$$

then ϵ has usually the *same sign* throughout the iteration process. This means that the partial sums

$$\left|\sum_{k=1}^{m} \epsilon_j^k\right|$$

increase monotonically as $m \to \infty$. We should then be able to speed up the convergence by correction terms that are larger in absolute value than ϵ_j^k. Let us introduce an 'over-correction,' or *over-relaxation parameter* $\omega > 1$.

$$u_j^{n+1,k+1} = u_j^{n+1,k} + \omega\epsilon_j^k.$$

We can define ϵ_j^k simply as

$$\epsilon_j^k = y_j^{n+1,k+1} - u_j^{n+1,k}, \tag{5.22}$$

where $y_j^{n+1,k+1}$ is calculated by (5.20). Note that if $\omega \equiv 1$ then we have the Gauss-Seidel method again. The SOR method converges to the correct solution for $\alpha > 0$ and $0 < \omega < 2$ [2].

The Projected SOR. We have described an efficient iterative method which solves the matrix equation (5.15); however, this is done more efficiently by LU decomposition. The utility of the SOR method is proven when applying it to valuing American options depending on the conditions (5.11), (5.12), and (5.13). With some minor modifications to the SOR algorithm, we can accommodate these conditions easily.

The *Projected SOR Method* works as follows. First, set up a vector $\vec{y}^{n+1,0} = \vec{u}^n$. Then calculate the next iterate $y_j^{n+1,k+1}$ using (5.21):

$$y_j^{n+1,k+1} = \frac{1}{1+\alpha}\left(b_j^{n+1/2} + \tfrac{1}{2}\alpha\left(u_{j-1}^{n+1,k+1} + u_{j+1}^{n+1,k}\right)\right) \tag{5.23}$$

which is the 'guess' for $u_j^{n+1,k+1}$ before over-relaxation. The over-relaxed iterate is

$$w_j^{n+1,k+1} = u_j^{n+1,k} + \omega\left(y_j^{n+1,k+1} - u_j^{n+1,k}\right) \tag{5.24}$$

Here if $w = u$ then we have the usual SOR algorithm and the iteration solves $A\vec{u}^{n+1} = \vec{b}^{n+1/2}$; however since we must have $\vec{u}^n \geq \vec{g}^n$, we enforce this inequality by

$$u_j^{n+1,k+1} = \max\left\{w_j^{n+1,k+1},\ g_j^{n+1}\right\}. \tag{5.25}$$

Given the initial guess (5.18), the steps (5.23), (5.24), (5.25) are repeated for $k = 1, 2, \ldots$ until

$$\|\vec{u}^{n+1,k+1} - \vec{u}^{n+1,k}\| = \sum_{j=1}^{J-2}(u_j^{n+1,k+1} - u_j^{n+1,k})^2 < \varepsilon^2,$$

where ε is the pre-chosen tolerance of error. Then we can put

$$\vec{u}^{n+1} = \vec{u}^{n+1,k+1},$$

and repeat the procedure for each timestep until $\tau_{\max} = N\delta\tau$ (expiry) is reached.

The Projected SOR method can be easily applied to Bermudan options. We should modify the payoff array g_j^n such that $g_j^n \equiv 0$ whenever the Bermudan option cannot be exercised at $t = n\delta\tau$. Then (5.25) will automatically exclude the possibility of early exercise.

5.4 Pricing American puts and calls

In this section we will demonstrate using the Crank-Nicolson method with Projected SOR algorithm applied to vanilla American put and call options.

5.4.1 Payoff function transformation

When computing the value for American options, we need to access the payoff function at different points of time. Therefore, we must consider the payoff function as a function of τ for $\tau \geq 0$. The transformation (4.3) is

$$u = e^{\gamma x + (\beta^2 + d)\tau} V(S_t, t)/K,$$

so the payoff function for the American option has an extra τ-dependent coefficient $e^{(\beta^2+d)\tau}$ when compared to European options. At expiry $\tau = 0$, the payoffs of European and American options coincide.

The transformed payoff for the American put is

$$g(x, \tau) = e^{(\beta^2+d)\tau} \left(e^{\gamma x} - e^{\beta x} \right)_+ .$$

For the American call,

$$u_0(x) = e^{(\beta^2+d)\tau} \left(e^{\beta x} - e^{\gamma x} \right)_+ .$$

Similarly, for the American binary put,

$$u_0(x) = e^{-\log K + \gamma x + (\beta^2+d)\tau} \quad \text{when} \quad S_T < K \iff x < 0.$$

For the American binary call,

$$u_0(x) = e^{-\log K + \gamma x + (\beta^2+d)\tau} \quad \text{when} \quad S_T > K \iff x > 0.$$

5.4.2 Boundary conditions

For American options, boundary conditions can be derived from the payoff function [2]: if $g(x, \tau)$ is the transformed payoff function,

$$\begin{cases} u_{\inf}(\tau) & = \lim\limits_{x \to -\infty} g(x, \tau) \\ u_{\sup}(\tau) & = \lim\limits_{x \to +\infty} g(x, \tau) \end{cases}$$

In the finite-discretization framework, we have simply:

$$\begin{cases} u_{\inf}(\tau) & = g(x_{\min}, \tau) \\ u_{\sup}(\tau) & = g(x_{\max}, \tau) \end{cases}$$

In the case of American puts and calls, the function $g(x, \tau)$ is just the European payoff function at $\tau = 0$ multiplied by the time-dependent factor $e^{(\beta^2+d)\tau}$. To summarize,

$$\begin{aligned} g(x,\tau) &= e^{(\gamma^2+d)\tau}\left(e^{\gamma x} - e^{\beta x}\right) \quad \text{for the Put} \\ g(x,\tau) &= e^{(\beta^2+d)\tau}\left(e^{\beta x} - e^{\gamma x}\right) \quad \text{for the Call} \\ \\ g(x,\tau) &= e^{(\beta^2+d)\tau - \log K + \gamma x}, \quad x < 0 \quad \text{for the Binary Put} \\ g(x,\tau) &= e^{(\beta^2+d)\tau - \log K + \gamma x}, \quad x > 0 \quad \text{for the Binary Call.} \end{aligned} \tag{5.26}$$

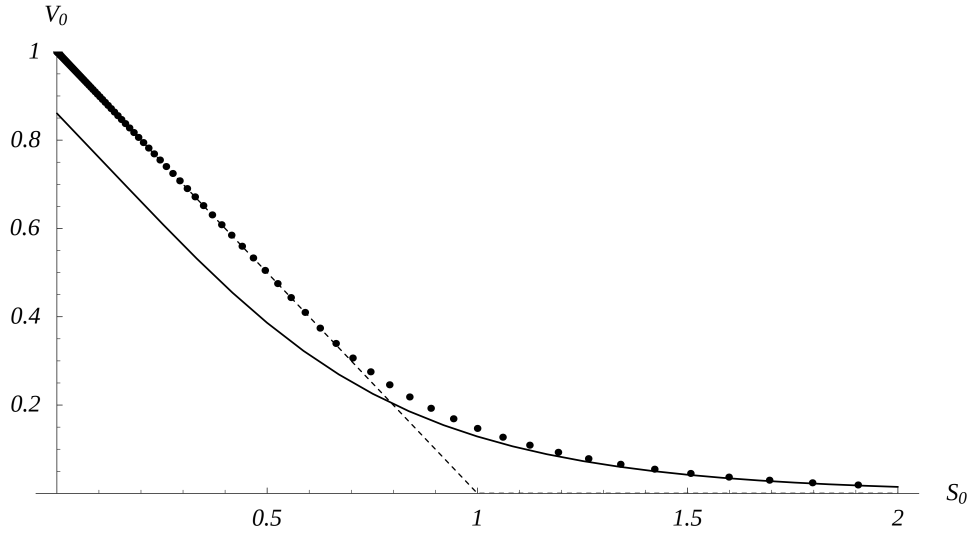


Figure 5.4: American Put calculated using the Crank-Nicolson finite-difference method and the Projected SOR algorithm. $\sigma = 0.30$, $r = 0.05$, $T = 3.0$. Note how the function joins smoothly into the payoff function, which is shown as a dotted curve. The continuous smooth curve is the value of the corresponding European option.

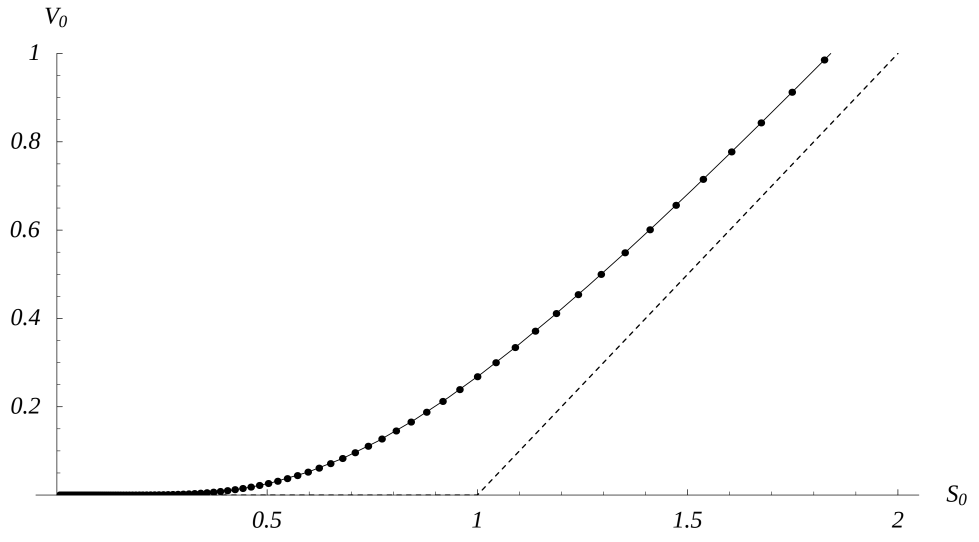


Figure 5.5: American Call calculated using the Crank-Nicolson finite-difference method and the Projected SOR algorithm. $\sigma = 0.30$, $r = 0.05$, $T = 3.0$. The value of the option is exactly the same as that of the corresponding European call option, which is shown as a smooth curve. The payoff function is shown as a dotted curve.

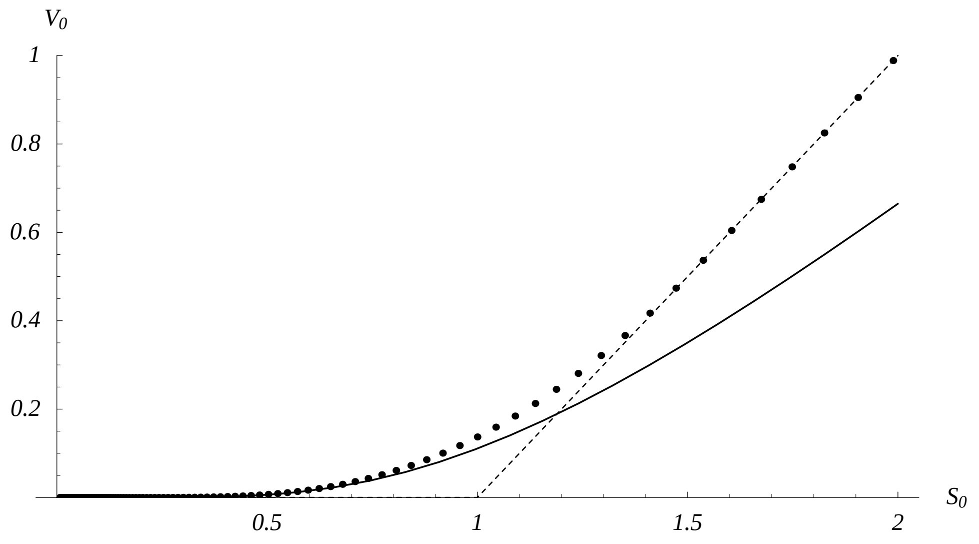


Figure 5.6: American Call calculated dividend-paying underlying, calculated using the Crank-Nicolson method. $\sigma = 0.30$, $r = 0.05$, $T = 3.0$, $D = 0.10$. The value of the corresponding European call option, is shown as a smooth curve. The payoff function is shown as a dotted curve.

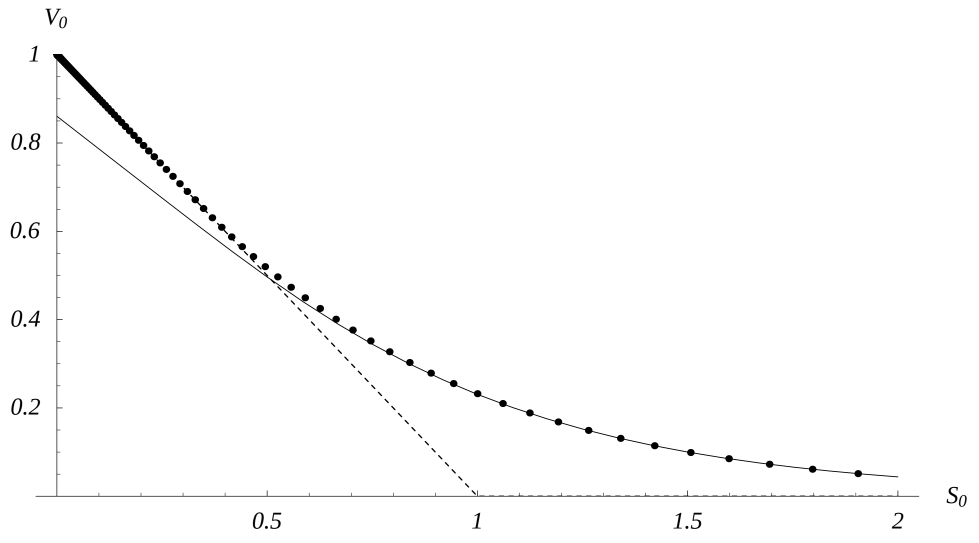


Figure 5.7: American Put with dividend-paying underlying, calculated using the Crank-Nicolson finite-difference method and the Projected SOR algorithm. $\sigma = 0.30$, $r = 0.05$, $T = 3.0$, $D = 0.10$. The value of the corresponding European call option, is shown as a smooth curve. The payoff function is shown as a dotted curve.

5.5 Perpetual Options

A *perpetual option* is defined to be an option that does not expire, but conveys the right to exercise it at any time; in essence it is a kind of an American option.

No expiry means that the expiration date T is, in theory, infinity. It follows that the time to expiry $(T - t)$ is always infinity and the option value cannot thus depend on t at all; the value of such an option is thus independent of time and therefore a function of the underlying only: $V(S)$.

The perpetual option case is interesting because the value of a perpetual option V_∞ can be seen as the limit of the value of a 'regular' American option V_{am} with the expiry tending to infinity:

$$V_\infty(S) = \lim_{T\to\infty} V_{am}(S_T, T - t_0),$$

where t_0 denotes the present time.

The perpetual puts and calls do have an exact solution. We can try to match these solutions with our finite-difference approximations to calibrate the computer program by setting the expiry of an American option to a large number, for example 250 years. If the finite-difference approximation does not approach the exact solution of the perpetual option, there is obviously something wrong with the program. As the finite-difference approximation evolves in time, it should tend to the exact solution, i.e. seem to 'freeze' eventually.

We will only take the perpetual put option as the final example.

Perpetual Put. Holding a perpetual put option V_{PP} allows us to gain the payoff $(K - S)_+$ any time. It is then clear that $V_{PP} \geq (K - S)_+$. Since $\max(K - S)_+ = K$, we then must have

$$(K - S)_+ \leq V_{PP} \leq K.$$

The maximum value is gained when $S = 0$; then it is clearly optimal to exercise. On the other hand, for $S > K$, early exercise is not optimal since it yields zero. Thus it is preferable to hold it (or to sell it). Now it seems that there are some S when it is optimal to hold and some S when it is optimal to exercise.

Whenever $V_{PP} > (K - S)_+$, it is optimal to hold and therefore it must satisfy the Black-Scholes partial differential equation (2.8). Since perpetual options are independent of time, $\partial V/\partial t \equiv 0$ and the Black-Scholes equation becomes

$$\tfrac{1}{2}\sigma^2 S^2 \frac{\partial^2 V_{PP}}{\partial S^2} + (r-D)S\frac{\partial V_{PP}}{\partial S} - rV_{PP} = 0. \tag{5.27}$$

There is an explicit solution to this PDE:

$$V_{PP}(S) = AS + BS^{-k},$$

where $k = 2r/\sigma^2$. For the perpetual put, $A \neq 0$ is impossible, since as $S \to \infty$, $V(S) \to 0$.

It therefore seems that there is a point S_* such that the value of the option is

$$V_{PP}(S) = \begin{cases} (K-S)_+ & \text{if} \quad S \le S_* \\ BS^{-k} & \text{if} \quad S > S_* \end{cases}$$

We require that V_{PP} be continuous, or else there will be arbitrage opportunities. Therefore, The two curves must join at the same point. Moreover, $S_* < K$ since otherwise the payoff is zero at S_* and then it is certainly not optimal to exercise in the neighborhood of S_*. Thus

$$K - S_* = BS_*^{-k}. \tag{5.28}$$

As once stated earlier, the value of an American option is maximized when both the function V and the derivative $\partial V/\partial S$ are continuous. Hence, if we require that $\partial V_{PP}/\partial S$ is continous at S_*, differentiating (5.28) we get another condition for S_*:

$$-1 = -kBS_*^{-k-1}.$$

These two equations yield

$$S_* = \frac{k}{k+1}K. \tag{5.29}$$

Finally, the value of the perpetual put is

$$V_{PP}(S) = \begin{cases} (K-S)_+ & \text{if} \quad S \le S_* \\ \left(\frac{S}{S_*}\right)^{-k}(K-S_*) & \text{if} \quad S > S_* \end{cases} \tag{5.30}$$

This solution is illustrated in figure (5.9).

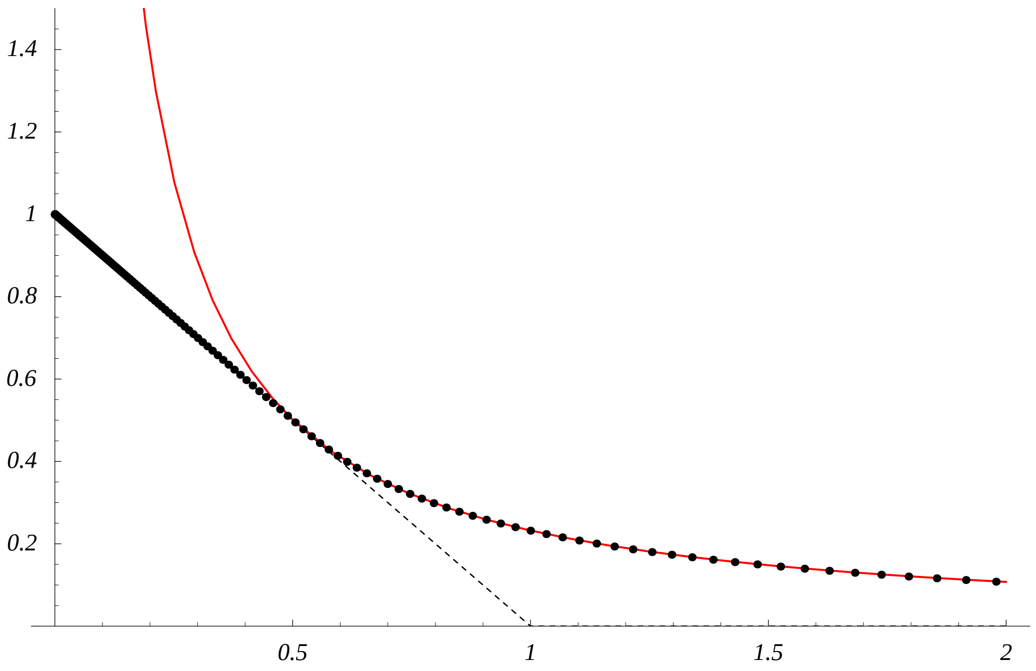


Figure 5.8: American put option with an expiry of 250 years (black dots). Crank-Nicolson finite-difference approximation with the x-axis truncated at $x_{\min} = -10.0$, $x_{\max} = +10.0$. $N = 1000$ timesteps, $J = 400$ spatial points. $K = 1.0$, $r = 0.05$, $\sigma = 0.30$, $D = 0$. The continuous curve is the solution $V(S) = BS^{-2r/\sigma^2}$ to the corresponding time-independent Black-Scholes partial differential equation (thin, red line). The option value function with $T = 250.0$ is a good approximation to the corresponding perpetual option.

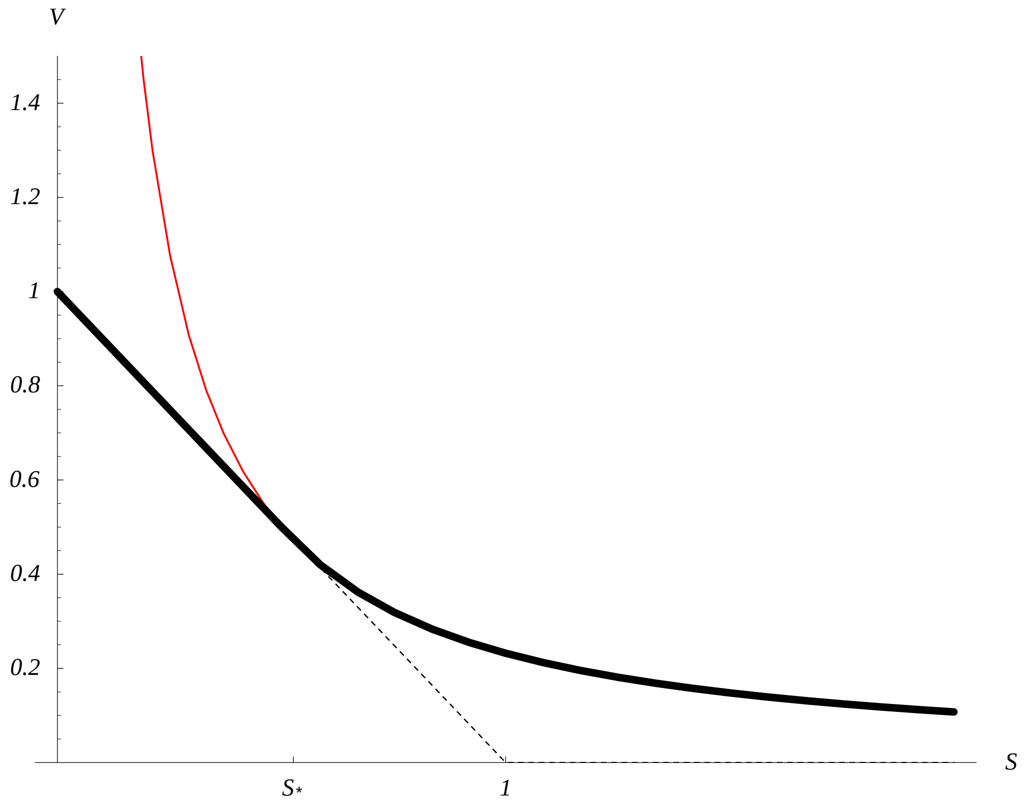


Figure 5.9: Exact Perpetual Put solution shown as a thick black line joining the payoff function $(K - S)_+$ and the solution $V_\infty(S) = BS^{-k}$ to the time-independent Black-Scholes PDE, with $k = 2r/\sigma^2$ (thin, red line). The payoff curve and the solution curve V_∞ join at the point $S_* = \frac{k}{k+1}K$. Here we have the usual parameters $K = 1.0$, $r = 0.05$, $\sigma = 0.30$, $D = 0$. The function is shown in figure (A.11).

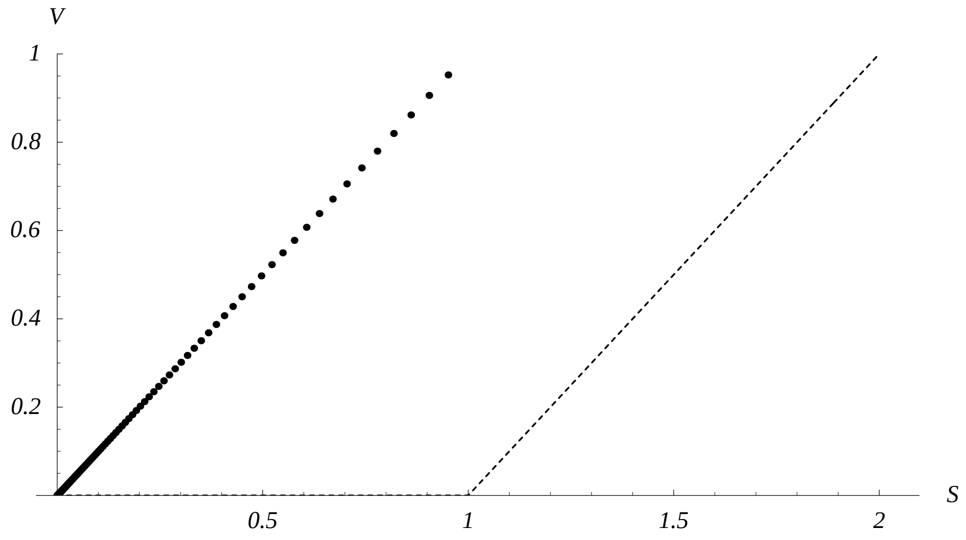


Figure 5.10: American call option on a non-dividend paying underlying, with an expiry of 250 years (black dots). Crank-Nicolson finite-difference approximation with the x-axis truncated at $x_{\min} = -10.0$, $x_{\max} = +10.0$. $N = 1000$ timesteps, $J = 400$ spatial points. The value of the perpetual call on a non-dividend paying underlying is equivalent to that of the underlying itself. $K = 1.0$, $r = 0.05$, $\sigma = 0.30$, $D = 0$. The value of the American call on a non-dividend paying asset is exactly the same as that of corresponding the European option.

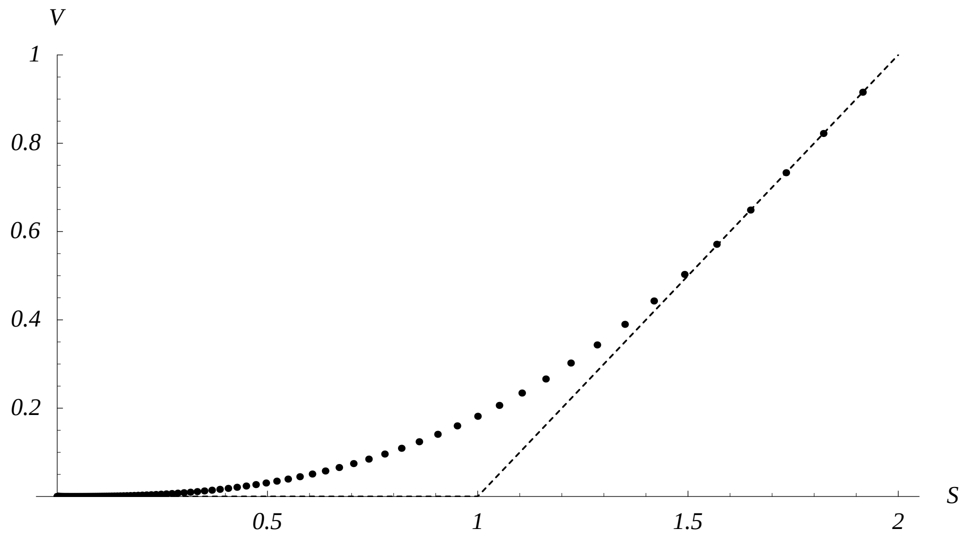


Figure 5.11: American call option on a dividend paying underlying, with an expiry of 250 years (black dots). Crank-Nicolson finite-difference approximation with the x-axis truncated at $x_{\min} = -10.0$, $x_{\max} = +10.0$. $N = 1000$ timesteps, $J = 400$ spatial points. $K = 1.0$, $r = 0.05$, $\sigma = 0.30$, $D = 0.10$. As shown in figure (2.6), the value of the European option tends to vanish as $T \to \infty$; the value of the American option is always at least equal to the payoff function. The function is maximized when the curve and its first derivative $\partial V/\partial S$ are continuous.

Chapter 6

Conclusion

Introducing the lognormal model of asset price dynamics and the arbitrage-free pricing concept, the value of an option can be uniquely determined, given the risk-free yield r, and the volatility σ and the spot price of the asset S_0.

There are two equivalent ways of determining the arbitrage-free value of the option. These are the risk-neutral expectation formula, and the Black-Scholes partial differential equation, given the necessary final conditions and boundary conditions.

The binomial model used to derive the risk-neutral expectation formula lends itself naturally to a very simple and accurate option valuation algorithm on a computer. The model can be implemented as a recursive method or a summation method. The recursive method is more flexible and easily modifiable to valuing American options. The summation method is suitable only for European options. Both algorithms are easily adaptable to computer languages, even spreadsheets.

The finite-difference methods attempt to solve for the option value function V from the Black-Scholes equations, given the payoff function and boundary conditions in a finite-difference grid. Rather than apply them directly to the Black-Scholes equation, we prefer to first transform the equation into the standard diffusion equation $u_\tau = u_{xx}$. Starting from various Taylor expansions of this equation, we can derive several difference equations that approximate the evolution of the actual continuous function $u(x, \tau)$ to different degrees of accuracy.

The finite-differencing approach applied to the diffusion equation introduces two sources of inaccuracies: one is the fact that we cannot use infinitesimal values $\mathrm{d}\tau$ and $\mathrm{d}x$ but rather their finite but small approximations $\delta\tau$

and δx; the other is that we must truncate the x-axis at both ends since the spatial domain is infinite. We can choose the cutoff points at each end, upon considering the shape of the payoff function: any interesting behavior of the function should be included in our domain. Also, since the errors tend to propagate everywhere along the x-axis, the longer the expiry, the larger the domain should be.

Given the transformed payoff function and boundary conditions at each ends of the truncated spatial domain, we can compute the evolution of the diffusion equation one timestep at a time using an appropriate method. there are several finite-difference methods, of which a method called the Crank-Nicolson method was chosen as the most useful because of its universal stability and superior convergence rate compared to the other two methods presented. Its better accuracy also makes it the most efficient, when the number of required computer operations needed for the solution is considered.

The finite-difference methods are much more complicated than the binomial methods, but their most notable difference is that they allow to recover many solution points at once while one the binomial method solves for only one option value. The finite-difference methods are thus usually more efficient than binomial methods.

The Crank-Nicolson method is an implicit method, which requires the solution of a system of equations for each timestep; in the case of European options, we are able to find the solutions exactly using an efficient tridiagonal matrix inversion scheme called the LU decomposition.

The numerical methods are not necessary for pricing plain vanilla European options. Since American options do not usually have explicit valuation formulas, numerical schemes become more important.

The binomial method for American options is almost as straightforward to implement as it is for European options. Only slightly more memory overhead is needed; otherwise, the method is very simple. Again, the binomial solution solves for only one value at a time, which is inefficient compared with the finite-difference methods.

The Crank-Nicolson method and other finite-difference methods are applicable to American options, but we must use an iterative method to accommodate the various constraints and inequalities due to the possibility of the early exercise of an option. An efficient iterative algorithm called the Projected Successive Over-Relaxation was introduced for solving the American option valuation problem with finite differences.

Some special American options have explicit solution formulas. The exact solution for the perpetual can be easily derived and implemented. It is also notable that perpetual options are American options with the expiry at infinity; by letting the initial condition evolve for a sufficiently long time, the perpetual option solution can be found. This fact can also be utilized to detect possible mistakes in the implementation.

Appendix A

Sample Program Code

This appendix contains all the relevant functions from the C++ program used to implement the binomial and finite-difference methods.

The program was used to output the necessary data that was subsequently formatted in *Mathematica* to output all of the graphs in this thesis.

The pseudo-code examples from the book [2] by Wilmott, Howison, and Dewynne were used for reference when writing the actual C++ routines.

None of the 'administrative' routines and easily programmable routines such as payoff functions are included here. The complete program is available for download on my world wide web site [1].

Program Structure

The actual C++ program consists of several classes. Each option type has its own class, for example, `European_Option`, `American_Option`, which have the derived classes such as `European_Call` and `European_Put`. All of the basic option types are derived from the base class `Option`, which contains an instance of the class `Discretization`. The `Option` class also contains instances of such classes as `Asset`.

The `Discretization` class object, which is available for each of the option instance contains all the necessary parameters for a specific finite-difference discretization or a binomial method, including the parameters $\delta\tau$,δx, α u, d, etc.

[1]`http://www.financialmathematics.org/`

The member variable names have been translated directly to `dx`, `alpha`, etc., with some exceptions such as the number of timesteps N is called `nmax`, `dt` denotes δt or $\delta\tau$ depending on the context.

All of the variables related to a certain numerical method or discretization are initialized prior to the actual numerical method routine is called. These parameters are contained in a `Discretization` class instance.

At the top of each routine that is presented here, local variables are assigned values from the `Discretization` object. This is done to separate the actual main part of the routine from dependencies of the 'infrastructure' of the program.

However, this 'uninteresting' variable-assignment part of each routine was removed for the purposes of presentation in this thesis. Apart from the missing variable assignments, the code presented here is the actual code. The complete list of the undeclared variables is shown below. The variable names are mostly self-explanatory for the readers of this text.

The code shown here is purposedly written so that it as clear as possible to show only the essence of the algorithms.

All arrays and matrices were re-indexed so that negative indices could be used so that for example, $j_{\min}$ is typically a negative integer, and correspondingly `jmin` is actually negative. The indices `n`,`j` in the program correspond to the indices n, j in the text. These arrangements make the programs look very close to the pseudo-code examples in [2].

The LU decomposition and iterative methods were programmed as separate classes `LU_solver` and `Itsolver`. The options have direct access to an instance of the `Discretization` class, which contains an instance of both of these solver classes. Prior to running the fully implicit finite-difference method or the Crank-Nicolson method, the necessary setup is done. Relevant methods from these classes are included in this appendix.

Program Listings

List of variables and subroutines

The subroutines assume the following variables have been set up:

Global variable:

`double r`	—r, riskless rate of return, assumed constant

For each of the option object instance we have,

`double expiry`	—T, expiry of the option
`double spot`	—S_0, spot price of the underlying

For the binomial methods:

`int nmax`	—N, number of periods
`double u`	—u, up-jump coefficient
`double d`	—d, down-jump coefficient
`double q`	—q, risk-neutral probability
`double dt`	—$\delta t = T/N$

For the European binomial valuation procedure:

`double *V`	—pointer to a vector V_j `V[0..nmax]`
`double *S`	—pointer to a vector S_j `S[0..nmax]`
`double *B`	—pointer to a vector B_j `B[0..nmax]`

For the American binomial valuation procedure:

`double **V`	—pointer to a matrix V_j^n `V[0..nmax][0..nmax]`
`double **S`	—pointer to a matrix S_j^n `S[0..nmax][0..nmax]`

For the finite-difference methods,

`int nmax`	—N, number of timesteps
`int taumax`	—$\tau_{\max}$, dimensionless time remaining until expiry

`double xmin`	—$x_{\min}$, x-value at the lower boundary
`double xmax`	—$x_{\max}$, x-value at the upper boundary
`int jmin`	—$j_{\min} = x_{\min}/\delta x$.
`int jmax`	—$j_{\max} = x_{\min}/\delta x$,
`double dt`	—$\delta\tau = \tau_{\max}/N$
`double dx`	—$\delta x = (x_{\max} - x_{\min})/N$
`double alpha`	—$\alpha = \delta\tau/\delta x^2$
`double *U`	—pointer to a vector array `U[jmin..jmax]`
`double *B`	—pointer to a vector array `B[jmin..jmax]`
`double *G`	—pointer to a vector array `G[jmin..jmax]`
`double *nextU`	—pointer to a vector array `nextU[jmin..jmax]`
`Itsolver  itsolver`	—Iterative method solver object
`LU_solver lusolver`	—LU solver object

For the LU decomposition object,

`double alpha`	—α, as above
`int jmin`	—$j_{\min}$, as above
`int jmax`	—$j_{\max}$, as above
`double *y`	—pointer to a vector array δ_j `y[jmin..jmax]`
`double *q`	—pointer to a vector array v_j `q[jmin..jmax]`

For the SOR Iterative solver object,

`double alpha`	—α, as above
`int jmin`	—$j_{\min}$, as above
`int jmax`	—$j_{\max}$, as above
`double *y`	—pointer to a vector array `y[jmin..jmax]`
`double *q`	—pointer to a vector array `q[jmin..jmax]`
`int early_exercise`	—0 or 1, true or false
`double epsilon`	—ϵ, error tolerance, e.g. 10^{-6}

Subroutines:

`payoff(double s)`	—payoff in monetary terms
`u_payoff(double x)`	—payoff in terms of dimensionless u

Binomial recursive method for European options

```
void European_Option::binomial(void) {
  int n, j;
  double p  = 1.0 - q;

  S[0] = spot;
  // Calculate the final asset values in a recombining tree
  for ( n=1; n<=nmax; ++n ) {
      for ( j=n; j>0; --j )
        S[j] = u*S[j-1];
      S[0] = d*S[0];
  }
  // Calculate the payoffs
  for ( n=0; n<=nmax; ++n )
    V[n] = payoff( S[n] );
  // Work backward in the tree
  for ( n=nmax; n>0; --n ) {
    for ( j=0; j<n; ++j ) {
      V[j] = ( q*V[j+1] + p*V[j] );
    }
  }
  V[0] = exp(-r*expiry) * V[0]; // discount
}
```

Figure A.1: The Binomial Method using the recursive algorithm for European options (C++). If desired, the computation of asset values $Su^j d^{N-j}$ can be done prior to calling to this routine. The risk-free yield (discount rate) is supposed to be constant during the period T. If the discount rate changes during the valuation, the line involving the computation of `S[j]` inside the double loop must be changed to `S[j] = exp(-f*dt)*( p*S[j+1] + q*S[j] )` where `f` is the appropriate discount rate for that period. The vector `S[j]` can be set up in advance. If the parameter setup (2.25) is used, it does not need to be recomputed unless the number of periods `nmax` changes.

Binomial summation method for European options

```
void European_Option::binomial_sum(void)
 int n,j;
 double v = 0.0;
 B[0]=1.0;
 S[0]=1.0;
 for (n=1; n<=nmax; n++) {
  B[n]=0.5*B[n-1];
  S[n]=u*S[n-1];
  for (j=n-1;j>=1;--j) {
    B[j]=0.5*(B[j]+B[j-1]);
    S[j]=u*S[j-1];
  }
  S[0]=S[0]*d;
  B[0]=0.5*B[0];
 }
 // The following loop can be run independently
 // of the above code provided that none of the variables
 // r,sigma,D,dt,nmax has changed.
 for (j=0; j<=nmax; j++) {
   v += B[j]*payoff(spot*S[j]);
 }
 return exp(-r*expiry)*v;
}
```

Figure A.2: The Binomial summation method for European option valuation (C++). First, the binomial coefficients $B_j = \binom{N}{j}2^{-N}$ are computed using a simple application of Pascal's triangle. B_j can be calculated by the recursive formula $B_j = \frac{1}{2}\left[\binom{N-1}{j} + \binom{N-1}{j-1}\right]$, $1 \le j \le N-1$ with $B_0 = B_N = 2^{-N}$. The array `S[j]` contains the asset value coefficients. The actual option value calculation loop can be run independently for various spot prices and payoff functions after the arrays `S[j]`, `B[j]` have been set up. The algorithm takes $O(\frac{1}{2}N^2)$ operations, but the actual option valuation takes only $O(N)$ operations.

Explicit finite-difference method for European options

```
void European_Option::explicit_fd(void) {
  int n, j;
  double alpha2   = 1.0-2.0*alpha;
  double tau  = 0.0;
  double *ptr;

  for ( j=jmin; j<=jmax; ++j )
    U[j] = u_payoff( j*dx, 0.0 );

  for ( n=1; n<=nmax; ++n ) {
    tau = dt*n;
    nextU[jmin] = u_inf(xmin,tau);
    nextU[jmax] = u_sup(xmax,tau);
    for ( j=jmin+1; j<=jmax-1; ++j )
      nextU[j] = alpha2*U[j] + alpha*(U[j+1]+U[j-1]);
    ptr = U; U = nextU; nextU = ptr;
  }
  if ( nmax % 2 == 1 )
    for ( j=imin;j<=imax;++j) U[j]=nextU[j];
}
```

Figure A.3: The Explicit finite-difference method in C++ . The initial condition is stored in array `U[j]` and the solution for the next timestep is stored in `nextU[j]`. After each timestep, the pointers of the arrays are swapped so the old solution becomes the initial condition for the next timestep. Finally, we will make sure that the solution is found in the same physical array (because of swapping of pointers, this is different if the number of timesteps is odd).

Fully implicit finite-difference method for European options

```
void European_Option::implicit_fd_lu(void) {
  int j, n;
  double tau;

  for ( j=jmin; j<=jmax; ++j )
    U[i] = u_payoff( j*dx, 0.0 );

  for ( n=1; n<=nmax; ++n ) {
    tau = n*dt;

    // Boundary value adjustments
    U[jmin+1] += alpha*u_inf(xmin,tau);
    U[jmax-1] += alpha*u_sup(xmax,tau);

    // Solve for U[] by LU decomposition, store in U[]
    lusolver.tridiagonal(U,U);
  }
}
```

Figure A.4: The Fully implicit finite-difference method (C++). The code looks trivial, since most of the action is hidden in the LU decomposition algorithm, which is implemented as method `LU_solver::tridiagonal`. This routine is invoked on the class instance `lusolver`, which was initialized prior to calling of this routine.

Crank-Nicolson finite-difference method for European options

```
void European_Option::crank_nicolson_lu(void)
  int n, j;
  double alpha2   = alpha/2.0;
  double alpha1   = 1.0-alpha;
  double tau;

  for ( j=jmin; j<=jmax; ++j )
    U[j] = u_payoff( j*dx, 0.0 );

  for ( n=1; n<=nmax; ++n ) {
    tau = n*dt;
    for ( j=jmin+1; j<=jmax-1; ++j )
      B[j] = alpha1*U[j]+alpha2*(U[j+1]+U[j-1]);
    U[jmin] = u_inf( xmin, tau );
    U[jmax] = u_sup( xmax, tau );
    B[jmin+1] += alpha2*U[jmin];
    B[jmax-1] += alpha2*U[jmax];
    lusolver.tridiagonal(B,U);
  }
}
```

Figure A.5: The Crank-Nicolson finite-difference method (C++). The method is a combination of the Explicit and Fully Implicit methods: first, the vector `B[j]`$= u_j^{n+1/2}$ is computed using the explicit method. This vector is given as input to the LU decomposition routine, which computes `U[j]`$= u_j^{n+1}$. `U[j]` becomes the input array for the next round; in the end, the solution will be in array `U[j]`. The routines `u_inf` and `u_sup` return boundary values.

Binomial method for American options

```
void American_Option::binomial(void) {
  int n, j;
  double p = 1.0 - q;
  double discount = exp(-r*dt);

  S[0][0] = spot;

  for ( n=1; n<=nmax; ++n ) {
    for ( j=n; j>=1; --j )
      S[n][j] = u*S[n-1][j-1];
    S[n][0] = d*S[n-1][0];
  }

  // Calculate the payoffs
  for ( j=0; j<=nmax; ++j )
    V[nmax][j] = payoff( S[nmax][j] );

  // Work backward in the tree to find the current value of the option
  for ( n=nmax-1; n>=0; --n ) {
    for ( j=0; j<=n; ++j ) {
      double hold, exercise;
      hold     = discount*( q*V[n+1][j+1] + p*V[n+1][j] );
      exercise = payoff( S[n][j] );
      V[n][j] = hold > exercise ? hold : exercise;  // MAX(hold,exercise)
    }
  }
}
```

Figure A.6: The Binomial method for American option valuation (C++).

Crank-Nicolson/Projected SOR method for American options

```
void American_Option::crank_nicolson_psor(void) {
  int n, j;
  int loops       = 0;
  int oldloops    = 10000;
  double alpha1   = 1.0-alpha;
  double alpha2   = alpha/2.0;
  double omega    = 1.0;
  double domega   = 0.05;
  double tau;
  itsolver.early_exercise = TRUE;

  for ( j=jmin; j<=jmax; ++j )
    U[j] = u_payoff( j*dx, 0.0 );

  for ( n=1; n<=nmax; ++n ) {
    tau = n*dt;
    for ( j=jmin+1; j<=jmax-1; ++j ) {
      G[j] = u_payoff( j*dx, tau );
      B[j] = alpha1*U[j]+alpha2*(U[j+1]+U[j-1]);
    }
    G[jmin] = u_payoff( xmin, tau );
    G[jmax] = u_payoff( xmax, tau );
    U[jmin] = G[jmin];
    U[jmax] = G[jmax];
    loops = itsolver.SOR(B, U, G, omega);
    if ( loops > oldloops ) domega *= -1.0;
    omega += domega; oldloops = loops;
  }
}
```

Figure A.7: Implementation of the Crank-Nicolson method for American options (C++).

Tridiagonal LU decomposition setup procedure

```
int LU_solver::tridiagonal_setup(double a, int jmn, int jmx) {
  int i;
  int lu_result;
  double diag = 1.0+2.0*a;
  double asq  = a*a;

  alpha = a;    // save into object for later retrieval
  jmin  = jmn;  // save into object for later retrieval
  jmax  = jmx;  // save into object for later retrieval

  lu_result = allocate_arrays(); // allocate d[], q[]

  if ( lu_result != SUCCESS )
    return lu_result;

  d[jmin+1] = diag;
  for ( i=jmin+2; i<=jmax-1; ++i ) {
    d[i] = diag - asq/d[i-1];
    if ( d[i] == 0 ) {
      clear_arrays();      // delete d[], q[]
      return(LU_SINGULAR); // Singular matrix---return error
    }
  }
  return(SUCCESS);
}
```

Figure A.8: The LU decomposition setup routine (C++). This routine must be called before running the LU tridiagonal solver (A.9). Only the diagonal elements of the array U δ_j are computed here. The result is stored in the one-dimensional array `d[]`.

Tridiagonal LU solver

```
void LU_solver::tridiagonal(double *b, double *u)
{
  int j;

  // Compute the intermediate vector v.

  v[jmin+1] = b[jmin+1];
  for (j=jmin+2; j<jmax; ++j) {
    v[j] = b[j] + alpha*v[j-1]/d[j-1];
  }

  // Compute the target vector u.

  u[jmax-1] = v[jmax-1]/d[jmax-1];

  for (j=jmax-2; j>jmin; --j)
    u[j] = (v[j]+alpha*u[j+1])/d[j];


}
```

Figure A.9: The LU tridiagonal solver (C++). Given the vector `b[j]` containing the data $\vec{b}^n$, and some vector `u[j]` of the same size, the matrix equation $A\vec{u}^{n+1} = \vec{b}^n$ is solved for `u[]`$=\vec{u}^{n+1}$. The vector containing the diagonal elements of the matrix U `d[]` must be computed prior to calling this routine using `LU_solver::tridiagonal_setup`. Note that the vector `b[]` is not needed after the first loop, so it is possible to call this routine so that `b`=`u`: `tridiagonal(U,U)`.

Projected SOR algorithm

```
int It_solver::SOR(double *B, double *U, double *G, double omega) {
  int j;
  double coeff = 1.0/(1.0+2.0*alpha);
  double error;
  int loops = 0;
  int maxloops = 10000;
  do {
    ++loops;
    if ( loops > maxloops ) {
      cerr "SOR does not converge after "<<maxloops<<" iterations";
      break;
    }
    error = 0.0;
    for ( j=jmin+1; j<jmax; ++j ) {
      double err, y, alive;
      y = coeff*(B[j]+alpha*(U[j-1]+U[j+1]));
      alive = U[j]+omega*(y-U[j]);
      if ( early_exercise ) {
        double dead  = G[j];
        y = alive > dead ? alive : dead; // MAX(alive,dead)
      } else {
        y = alive;
      }
      err = y-U[j];
      error += err*err;
      U[j] = y;
    }
  } while ( error > epsilon );
  return(loops);
}
```

Figure A.10: Projected SOR algorithm (C++). The variable `y` is the Gauss-Seidel iteration. `U[j-1]` is the latest iteration computed.

Perpetual Put valuation

```
perpetualPut[spot_, strike_, r_, sigma_] :=
  Module[{k, ss, b, joiningPoint},
   k = 2*r/sigma^2;
   ss = k/(k + 1);
   b = ss^k*(strike - ss);
   joiningPoint = ss*strike;
   Return[
      If[spot > joiningPoint, b*spot^(-k), K-spot]
   ]
]
```

Figure A.11: Perpetual Put formula implemented in Mathematica. This trivial routine returns the exact solution for the Perpetual Put. First, we calculate the point where the payoff function joins the function that is the solution for the time-independent Black-Scholes PDE. Then, depending on the value of `spot` (price), we return the value of either function. The functions are equal at `joiningPoint`; The spot values less than this point yield the payoff; the spot values greater than the point yield the solution $V(S) = BS^{-2r/\sigma^2}$.

Index

Bibliography

[1] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81:637–659, 1973.

[2] Paul Wilmott, Sam Howison, and Jeff Dewynne. *The Mathematics of Financial Derivatives: A Student Introduction.* Cambridge University Press, Cambridge, UK, first edition, 1995.

[3] J.C. Cox, S. Ross, and Rubinstein M. Option pricing: a simplified approach. *Journal of Financial Economics*, 7:229–263, 1979.

[4] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, Cambridge, UK, second edition, 1992.

[5] John C. Hull. *Options, Futures, and Other Derivatives.* Prentice-Hall, New Jersey, international edition, 1997.

[6] Paul Wilmott. *Derivatives: The Theory and Practice of Financial Engineering.* John Wiley & Sons Ltd., West Sussex, UK, first edition, 1998.