MATH 60082 Computational Finance

Dr Paul Johnson

LECTURE: Monday 11:00am - 12:00pm Turing Ground Floor G-108
LAB CLASS: Thursday 12:00pm - 14:00pm Turing Ground Floor G-105
OFFICE HOURS: Tuesday 10:30-11:30am (check website)

CONTACT DETAILS: Office - 2.107 Turing Building
Direct tel - (0161 30)63660
E-mail - paul.johnson-2@manchester.ac.uk
Website - http://www.maths.manchester.ac.uk/~pjohnson

ASSESSMENT:

This course is entirely assessed by project work. There will be five assignments in total, with two short mini tasks accounting for 5%, and three main assignments each of which will account for 30% of your final mark for this module. I will attempt to spread these as evenly throughout the semester; I will allow at least three weeks (for main assignments) between giving out project details and the date to be handed in. DEADLINES MUST BE STRICTLY OBSERVED!!!

IMPORTANT DATES:

Support Computing Class 13:00pm - 17:00pm Wednesday 28th January 2015 - AT G.105
Mini Task 1 8th February 2015
Mini Task 2 15th February 2015
Main Assignment 1 8th March 2015
Main Assignment 2 12th April 2015
Last Lecture 27th April 2015
Main Assignment 3 3rd May 2015

RECOMMENDED TEXTS:

There are a number of texts on Computational Finance; most are fairly awful. Other texts on more general aspects of the subject are good on some areas of Computational Finance, but not on others. Here are some suggestions:

Text books:
• A good basic text for Mathematical Finance (also useful for MATH 39032/60008) is:

• Alternatively, as an introductory text to the area:

• For a very detailed (and expensive) look at mathematical finance:

• For a more financial look at options and derivatives the following is excellent and is the course text for finance students (usually MBA or PhD) studying derivatives (with a decent treatment of binomial trees):

• For a readable book on Stochastic Finance (including a good treatment of Monte Carlo methods):

• For a book which describes numerous computational routines

• For a great book on the numerical solution of PDEs (written long before CF was invented!)

• For an excellent book for programing using C++
  M. Joshi, 2004 C++ Design Patterns and Derivatives Pricing. Cambridge University Press. ISBN 0 521 83235 7
1 Introduction

Important - always remember:

\textit{Garbage in, garbage out}

- this is the golden rule of computing!

1.1 Introduction

In this course we will be studying computational finance (or CF). Whilst basic financial mathematics problems \textit{may} have analytical solutions (which even then may ultimately require an element of computation, such as the evaluation of special functions - see below), most \textit{real} finance problems rely heavily on numerical (i.e. computational) techniques. Indeed, the calculation of the values of early exercise (e.g. American) options is inherently nonlinear, and as a consequence these must be tackled numerically.

The aim of the course is to give a broad outline of the main numerical techniques employed in the finance world. The list, in chronological order is:

- ‘Exact’ solutions - evaluation of the Black-Scholes formula
- Lattice (tree) methods
- Quadrature methods
- Monte Carlo methods
- Methods for partial differential equations (PDEs) - finite difference methods

Throughout the emphasis will be on a critical appraisal of methods, especially accuracy (i.e. errors).

The different types of errors can be categorised into the following main types:

- Roundoff errors
- Truncation and discretization errors
- Errors in modelling
- AND THE MOST IMPORTANT FOR BEGINNERS: Programming errors

Let us consider these in turn:

1.2 Roundoff errors

These arise when a computer is used for doing numerical calculations. Some typical examples include the inexact representation of (e.g. irrational) numbers such as \( \pi \), \( \sqrt{2} \). Roundoff and chopping errors arise from the way numbers are stored on the computer, and in the way arithmetic operations are carried out. Whereas most of the time the numbers are stored on a computer is not under our control, the way certain expressions are computed definitely is. A simple example will illustrate the point. Consider the computation of the roots of a quadratic equation \( ax^2 + bx + c = 0 \) with the expressions

\[ x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}, \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \]
Let us take $a = c = 1$, $b = -28$. Then $x_1 = 14 + \sqrt{195}$, $x_2 = 14 - \sqrt{195}$. Now to 5 significant figures we have $\sqrt{195} = 13.964$. Hence $x_1 = 27.964$, $x_2 = 0.036$. So what can we say about the error? We need some way to quantify errors. There are two useful measures for this.

Absolute error

Suppose that $\phi^*$ is an approximation to a quantity $\phi$. Then the absolute error is defined by $|\phi^* - \phi|$.

Relative error

Another measure is the relative error and this is defined by $|\phi^* - \phi|/|\phi|$ if $\phi \neq 0$.

For our example above we see that $|x_1^* - x_1| = 2.4 \times 10^{-4}$ and $|x_2^* - x_2| = 2.4 \times 10^{-4}$ which look small. On the other hand the relative errors are $|x_1^* - x_1|/|x_1| = 8.6 \times 10^{-6}$ and $|x_2^* - x_2|/|x_2| = 6.7 \times 10^{-3}$. Thus the accuracy in computing $x_2$ is far less than in computing $x_1$. On the other hand if we compute $x_2$ via

$$x_2 = 14 - \sqrt{195}$$

we obtain $x_2 = 0.03576$ with a much smaller absolute error of $3.4 \times 10^{-7}$ and a relative error of $9.6 \times 10^{-6}$.

Note that roundoff error can be reduced by performing arithmetic operations at higher precision (i.e. more significant figures).

1.3 Truncation and discretization errors

These errors arise when we take the continuum model and replace it with a discrete approximation. For example suppose we wish to solve

$$\frac{d^2U}{dx^2} = f(x),$$

using Taylor series (more details later) we can approximate the second derivative term by

$$\frac{U(x_{i+1}) - 2U(x_i) + U(x_{i-1})}{h^2}$$

where we consider a set of $x$ points $x_i = x_0 + ih$, $i = 1, 2, \ldots$, where we have taken a uniform grid with spacing $h$ say and node points $x_i$. As far the approximation of the equation is concerned we will have a truncation error given by

$$\tau(x_i) = \frac{d^2U(x_i)}{dx^2} - f(x_i) = \frac{h^2 d^4U}{12 dx^4} + \ldots$$

Even though the discrete equations may be solved to high accuracy, there will be still an error of $O(h^2)$ arising from the discretization of the equations. Of course with more points, we would expect this error to diminish.
1.4 Errors in modelling

These arise for example when the equations being solved are not the proper equations for the problem. For example the Black Scholes equation has a number (many) underlying assumptions (some of which can be relaxed, others of which are still open to question). No matter how accurately the solution has been computed, it may not be close to the real solution because other factors have been neglected in the computation. This class of error is not dealt with in this course, but you should always be aware of the limitations of the model you are using.

1.5 Programming errors (bugs)

These are errors entirely under the control of the programmer. To eliminate these requires careful testing of the code and logic, as well as comparison with previous work. It is always useful to have benchmarks - for example an exact solution may exist (sometimes), or previous work with which it is possible to compare your numerical results. However for your problem for which there may not be previous work to compare with, one has to do numerous self-consistency checks with further analysis as necessary.

1.6 Benchmarking

As mentioned above, if you are lucky enough to have an exact solution with which to compare your numerical results - use it!

Consider the Black Scholes PDE (this has been/will be derived in other modules)

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

Here $V$ denotes the value of the option, $S$ that of the underlying, $t$ time, $\sigma$ the volatility and $r$ risk-free interest rate.

For the case of a European call option (final condition $C(t = T) = \max(S - X, 0)$), the price is

$$C(S, t) = SN(d_1) - Xe^{-r(T-t)}N(d_2)$$

whilst for a European put option (final condition $P(t = T) = \max(X - S, 0)$), the price is

$$P(S, t) = Xe^{-r(T-t)}N(-d_2) - SN(-d_1)$$

where

$$d_1 = \frac{\log(S/X) + (r + \frac{1}{2}\sigma^2)(T-t)}{\sigma \sqrt{T-t}},$$

$$d_2 = \frac{\log(S/X) + (r - \frac{1}{2}\sigma^2)(T-t)}{\sigma \sqrt{T-t}},$$

and

$$N(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}s^2} ds$$

which we recognise as the cumulative distribution function for a Normal distribution. This is an example of the need to undertake a calculation, even when an analytic solution exists.
2 Euler’s method - a simple procedure for ODEs

This method is a major component of Monte Carlo simulations!

Consider the initial value problem

\[ \frac{dy}{dx} = f(x, y), \quad a \leq x \leq b, \]

subject to the initial condition \( y(a) = \alpha. \)

Note that this formulation also encompasses nonlinear ODEs.

Using a Taylor series expansion of \( y(x + h) \) about \( y(x) \) we obtain

\[ y(x + h) = y(x) + h \frac{dy}{dx}(x) + \frac{1}{2}h^2 \frac{d^2y}{dx^2}(x) + \ldots \]

Rearranging this, we obtain

\[ \frac{dy}{dx}(x) = \frac{y(x + h) - y(x)}{h} - \frac{1}{2}h \frac{d^2y}{dx^2}(x) + \ldots \]

Substituting this into our ODE, neglecting the \( O(h) \) terms leads to

\[ \frac{y(x + h) - y(x)}{h} \approx f(x, y(x)) \]

or

\[ y(x + h) \approx y(x) + hf(x, y(x)). \]

Given \( y(x = a) = \alpha \), this gives us a recipe for progressing the solution forwards in \( x \). This is a marching/stepping process, with the solution obtained at \( x = a + h \) (from \( x = a \)), \( x = a + 2h \) (from \( x = a + h \)), ... \( x = a + ih \) (from \( x = a + (i - 1)h \)), etc.

Advantages of the method

- It is simple (and also simple to program)
- It is robust
- \( h \) can be changed from step to step

Disadvantages of the method

- It is not very accurate

In general we say that a method is of order \( h^p \) if the error is of order \( h^p \). In principle if \( h \) decreases, we should be able to achieve greater accuracy, although in practice roundoff error limits the smallest size of \( h \) that we can take. Euler’s method is first-order accurate. Higher order schemes certainly exist, but Euler’s method is certainly the most popular with financial practitioners using Monte Carlo simulations.
3 Monte Carlo methods

3.1 Introduction

- We now look at a numerical scheme that uses the probabilistic solution - Monte Carlo techniques.
- The main idea behind the Monte Carlo technique is that you simulate paths that could be taken by the underlying asset (under the risk-neutral probability) and then use these to estimate an expected option price at expiry, which can be discounted back to today.
- Monte Carlo techniques are very useful for options on more than one underlying asset.
- Sadly, the convergence of Monte Carlo methods is slow and it is hard to determine the error terms.
- The convergence to the correct option value will be at a rate of $N^{-\frac{1}{2}}$ where $N$ is the number of sample paths.
- As it is a forward induction technique, which makes it particularly suitable for valuing path dependent options such as lookback and Asian options.
- It is very unsuitable for valuing American style options for exactly the same reasons, although we shall see methods for overcoming this.
- The computational effort increases linearly as you add underlying assets, thus to price an option with $d$ underlying assets (or sources of uncertainty such as stochastic volatility or stochastic interest rates) then an $N$ sample paths Monte Carlo method requires approximately $Nd$ calculations.
- Practitioners love these methods!!

3.2 Large numbers

- If we have a sequence of independent, identically distributed random variables $Y_n$ then we have that

$$\lim_{N \to \infty} \frac{1}{N} \sum_{n=1}^{N} Y_n = E[Y_1]$$

which is the law of large numbers.
- In other words the expectation is exactly like taking a long run average (exactly as we’d expect), so to evaluate the expectation to any desired accuracy we can simply take more and more draws of $Y_n$.
- With the Monte Carlo technique what we are trying to do is to evaluate the value of $E[f(Y_T)]$ which is the expectation of a function of a random variable $Y_T$. 
3.3 Application to options

- If we consider $S_t$ as the value of a share price at time $t$, then the option value at expiry, $t = T$ we can think of as $V(S_T, T)$ and from the fundamental theorem of finance we know that

$$V(S_t, t) = E_t^Q[e^{-\int_t^T r(s)ds}V(S_T, T)]$$

or

$$e^{-r(T-t)}E_t^Q[V(S_T, T)]$$

if $r$ is constant, where $Q$ is the risk-neutral measure and $E_t$ denotes taking the expectation at time $t$.

- Thus if we can estimate the expectation on the right hand side then we can simply discount this value at the risk-free rate to obtain the option price today. In fact, with Monte-Carlo methods it is also fairly straightforward to factor in stochastic interest rates as well.

3.4 Simple example with GBM

- If we assume that we are in the risk-neutral world and the underlying asset follows geometric Brownian motion, thus, under the real-world measure

$$dS_t = \mu S_t dt + \sigma S_t dX$$

and under the risk-neutral measure

$$dS_t = rS_t dt + \sigma S_t dW$$

where $W$ and $X$ are both Brownian motions under the respective measures.

- The above stochastic differential equation can be solved exactly to yield:

$$S_T = S_t e^{(r - \frac{1}{2} \sigma^2)(T - t) + \sigma dW}$$

or

$$S_T = S_t e^{(r - \frac{1}{2} \sigma^2)(T - t) + \sigma \phi \sqrt{T - t}}$$

where $\phi$ here is a variable drawn at random from a Normal distribution with a mean of 0 and a variance of 1, $N(0,1)$.

- To estimate the expected option value at time $T$, $V(S_T)$ then we take random draws from the $N(0,1)$ distribution which enables us to calculate $S_T$ and then calculate $V(S_T)$. To get an approximation of the expectation we then average $V(S_T)$.

- Thus if the $n$th draw from the normal distribution gives $V^n(S_T)$ then by the law of large numbers:

$$\frac{1}{N} \sum_{n=1}^N V(S^n_T) \to E_t^Q[V(S_T)] \quad \text{as} \quad N \to \infty$$

- Now if we define the error from the $n$th sample path as $\epsilon_n$ so

$$\epsilon_n = V(S^n_T) - E_t^Q[V(S_T)]$$
3.5 Central limit theorem and error

- The Central Limit Theorem tells us that for large \( N \) if the individual errors \( \epsilon_n \) have variance \( \nu = \text{var}(\epsilon_n) \) (which is the same for all \( n \)) then the error when approximating the expectation:

\[
\frac{1}{N} \sum_{n=1}^{N} V(S_{t}^{n}) - E_{t}^{Q}[V(S_{T})]
\]

is approximately normally distributed with mean zero and variance \( \nu/N \) and standard deviation \( (\nu/N)^{1/2} \).

- Sadly this error bound is difficult to estimate as it is probabilistic, in that we only know the distribution of the errors rather than their actual values.

- Also the standard deviation of the error only declines with the square root of the number of paths \( N \).

- For each individual path the error will be random, depending upon the draw of \( \phi \).

- This implies we are unable to use extrapolation techniques since the errors will not be monotonic.

3.6 The Monte-Carlo method for European options

- That gives the basics of the Monte Carlo method, it is very simple to implement for many different types of options.

- For a European call option the payoff at maturity \( V(S_{T}) \) is given by

\[
V(S_{T}) = \max(S_{T} - X, 0)
\]

and so, to value the option one simulates \( N \) possible values or paths for \( S_{T} \) by making \( N \) independent draws from \( N(0, 1) \) then to use these possible values, call them \( \phi_n \) we have for \( 1 \leq n \leq N \):

\[
S_{t}^{n} = S_{t} \exp\left( (r - \frac{1}{2} \sigma^2)(T - t) + \sigma \phi_n \sqrt{T - t} \right)
\]

\[
V(S_{t}^{n}) = \max(S_{t}^{n} - X, 0)
\]

\[
V(S_{t}, t) = e^{-r(T-t)} \frac{1}{N} \sum_{n=1}^{N} V(S_{t}^{n})
\]

3.7 Valuing Asian options

- An Asian option is an option whose payoff is a function of the average price of the underlying asset over the lifetime of the option. The share price is observed at \( M \) points in time (every day, every trading day, every week etc.) and the average is calculated by using either geometric or arithmetic averaging.

- Consider an Asian option whose payoff is

\[
V(T) = \max(A - S, 0)
\]
where

\[ A = \frac{1}{M} \sum_{i=1}^{M} S(t_i) \]

and \( S(t_i) \) are the share prices at the \( M \) sampling times \( t_1, \ldots, t_M \).

- We need to modify our Monte-Carlo method slightly to deal with this.
- From the solution to the SDE we know that

\[ S_{t_i} = S_t \exp[(r - \frac{1}{2}\sigma^2)(t_i - t) + \sigma(W(t) - W(t))] \]

\[ S_{t_{i-1}} = S_{t_i} \exp[(r - \frac{1}{2}\sigma^2)(t_{i+1} - t_i) + \sigma(W(t_{i-1}) - W(t_i))] \]

and so the procedure is as follows:

- Simulate each of the \( M \) increments \( dW_i \) by drawing \( \phi_i \) from the Normal distribution and use

\[ S^n_{t_i} = S^n_{t_{i-1}} \exp[(r - \frac{1}{2}\sigma^2)(t_i - t_{i-1}) + \sigma\sqrt{t_i - t_{i-1}}\phi_i] \]

to estimate the underlying asset values at each time.

- Calculate the value of \( A^n \) and the payoff \( \max(A^n - X, 0) \)

- Then repeat this procedure \( N \) times to get the option value:

\[ V(t) = e^{-r(T-t)} \frac{1}{N} \sum_{n=1}^{N} \max(A^n - X, 0) \]

- Note that here, we need to break up the path of \( S_t \) into \( M \) time steps and then use the \( N \) created paths to take the average.

### 3.8 Generating (Pseudo-)Random Numbers

- Many statistical packages have (normal) random number generators

- Otherwise, generate Normal random numbers by:
  - generating random numbers that are uniformly distributed on \([0, 1]\).
  - Transforming them to obtain normally distributed random numbers

- Most straightforward way: - let \( F^{-1} \) denote inverse of Normal distribution function

- If \( x \) is uniformly distributed on \([0, 1]\), then \( y = F^{-1}(x) \) is a standard variable from the normal distribution.

- If you have function \( F^{-1} \) then this is easy. For example, in Excel:-
  - Has function RAND() which generates a random number uniformly distributed on \([0, 1]\)
  - Has function \( F^{-1} \), called NORSINV()
Thus, function call NORSINV(RAND()) returns a realisation of a standard normal random variable.

- Disadvantage: \( F^{-1} \) is not known in closed form, so this approach is not always fast.

- The other issue is that these are only "pseudo" random numbers in that the computer program typically has an algorithm for calculating the ‘random numbers’, as John von Neumann said in 1951 "Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin".

- There are many alternatives (The "Recipe" books of Press, Teuklovsky, Vetterling and Flannery are a good source, and have a good discussion of the problem), such as the Mid-Square method, Congruential generation or the popular Mersenne twister...

### 3.9 The Box-Muller method

- The simplest technique for generating ‘decent’ Normally distributed random numbers (according to Wilmott) is the Box-Muller method which takes any uniformly distributed variables (from any source you prefer) and turns them into Normally distributed ones.

- Given two uniformly distributed random numbers \( x_1, x_2 \), two Normally distributed random numbers, \( y_1 \) and \( y_2 \) are given by:

\[
y_1 = \cos(2\pi x_2) \sqrt{-2\log(x_1)}, \quad y_2 = \sin(2\pi x_1) \sqrt{-2\log(x_2)}
\]

### 3.10 More General Problems

- The two examples we talked about above - BSM setup (geometric Brownian motion) are particularly easy because increments to \( \log S \) are normally distributed or alternatively we can write the solution to the SDE out explicitly in terms of the Normal distribution.

- What do we do for non-Normal stochastic processes?

- Previously we introduced the SDE,

\[
dS_t = \mu S_t dt + \sigma S_t dX;
\]

we first introduced a process

\[
S_{t+\delta t} = S_t + \mu S_t \delta t + \sigma S_t \phi \sqrt{\delta t}
\]

where \( \delta t \) a small increment of time and \( \phi \) a Normal r.v.

- The second process is more than a way to explain the first process.

  - Use Euler scheme to approximate the SDE

  - Letting \( S_{t}^{dt} \) denote the value of the Euler process at time \( t \) using an increment of \( \delta t \), \( S_{t}^{dt} \rightarrow S_t \) in the sense of mean square.
3.11 Stochastic volatility

- The Euler approximation is useful for processes that are not normal, i.e. something other than GBM
- Example: A stochastic volatility model. Under $Q$, we have

$$
\begin{align*}
    dS_t &= rS_t dt + \sigma_t S_t dW_1 \\
    d\sigma_t &= (a - b\sigma_t) dt + v\sigma_t \sqrt{1 - \rho^2} dW_3
\end{align*}
$$

- Here
  - $a, b, v, r, \rho$ are constants,
  - $\sigma$ and $S$ are stochastic processes
  - $\sigma(t)$ is interpreted as volatility at time $t$
- Consider $M + 1$ equally spaced times $t_0, t_1, \ldots, t_M$, where $\delta t = t_i - t_{i-1}$.
- Euler approximation is

$$
\begin{align*}
    S_{t_i} &= S_{t_{i-1}} + rS_{t_{i-1}} \delta t + \sigma_{t_{i-1}} S_{t_{i-1}} \phi_{1i} \sqrt{\delta t} \\
    \sigma_{t_i} &= \sigma_{t_{i-1}} + (a - b\sigma_{t_{i-1}}) \delta t + v\sigma_{t_{i-1}} \phi_{1i} \sqrt{\delta t} + v\sigma_{t_{i-1}} \sqrt{1 - \rho^2} \phi_{2i} \sqrt{\delta t}
\end{align*}
$$

where the $\phi_{ki}$ ($k = 1, 2$) are independent standard normal r.v.’s.
- Only change from before is that we now do simulation using Euler approximation - we must compute values at intermediate timesteps (GBM is special - SDE can be integrated ‘exactly’).

3.12 Two underlying assets

- If we have two correlated underlying assets such that

$$
\begin{align*}
    dS_1 &= \mu_1 S_1 dt + \sigma_1 S_1 dX_1 \\
    dS_2 &= \mu_2 S_2 dt + \sigma_2 S_2 dX_2^* \\
\end{align*}
$$

where

$$
    X_2^* = \rho X_1 + \sqrt{1 - \rho^2} X_2
$$

or changing the notation slightly

$$
\begin{align*}
    dS_1 &= \mu_1 S_1 dt + \sigma_1 S_1 dX_1 \\
    dS_2 &= \mu_2 S_2 dt + \sigma_2 \rho S_2 dX_1 + \sigma_2 \sqrt{1 - \rho^2} S_2 dX_2
\end{align*}
$$

then it is very straightforward to value an option whose payoff depends upon both of the asset prices at expiry.
- Consider an option where $V(T) = \max(S_1 - X_1, S_2 - X_2, 0)$. So

$$
    V(t) = e^{-r(T-t)} E_t^Q [V(T)]
$$
• Under risk-neutrality we have

\[ dS_1 = rS_1 dt + \sigma_1 S_1 dX_1 \]
\[ dS_2 = rS_2 dt + \sigma_2 \rho S_2 dX_1 + \sigma_2 \sqrt{1 - \rho^2} S_2 dX_2 \]

thus from solving the SDEs

\[ S_{1T}^n = S_{1t}^n \exp\left[ (r - \frac{1}{2} \sigma_1^2)(T - t) + \sigma_1 \sqrt{T - t} \phi_{1n} \right] \]
\[ S_{2T}^n = S_{2t}^n \exp\left[ (r - \frac{1}{2} \sigma_2^2)(T - t) + \sigma_2 \rho \sqrt{T - t} \phi_{1n} + \sigma_2 \sqrt{1 - \rho^2} \sqrt{T - t} \phi_{2n} \right] \]

• So to value the equations simulate \( 2N \) draws of normally distributed random numbers and then determine the possible values of \( V(T) \) and then discount the average

• Thus the option value can be estimated as follows:

\[ V(t) = e^{-r(T-t)} E_t^Q [V(T)] = \frac{1}{N} \sum_{n=1}^{N} \max(S_{1t}^n - X_1, S_{2t}^n - X_2, 0) \]

• One question, however, is how to get the general form of the set of SDEs when we have more than one underlying asset.

• The way in which we typically see correlation matrices (in two underlying assets) are in terms of the covariance matrix of continuously compounded returns, i.e. \( \log(S_i(T)/S_i(t)) \) and this matrix is typically of the form

\[
\begin{pmatrix}
\sigma_1^2 & \rho \sigma_1 \sigma_2 \\
\rho \sigma_1 \sigma_2 & \sigma_2^2
\end{pmatrix}
\]

• We can write our two SDEs as follows:

\[
\begin{pmatrix}
\frac{dS_{1t}}{dS_{2t}}
\end{pmatrix} = \begin{pmatrix}
(rS_{1t}) & 0 \\
(rS_{2t}) & (r \sigma_1 S_{1t} \sigma_2 \rho S_{2t}) & (r \sigma_1 S_{1t} \sigma_2 \sqrt{1 - \rho^2} S_{2t})
\end{pmatrix}
\begin{pmatrix}
dW_{1t} \\
dW_{2t}
\end{pmatrix}
\]

• It so happens that the matrix, \( M \), here

\[
\begin{pmatrix}
\sigma_1 \\
\rho \sigma_2 \\
\rho \sigma_1 \sigma_2 \\
\sigma_2 \sqrt{1 - \rho^2}
\end{pmatrix}
\]

is the square root of the covariance matrix, \( \Sigma \)

\[
\begin{pmatrix}
\sigma_1^2 & \rho \sigma_1 \sigma_2 \\
\rho \sigma_1 \sigma_2 & \sigma_2^2
\end{pmatrix}
\]

in that \( \Sigma = MM^T \)

• To check note that

\[
\begin{pmatrix}
\sigma_1 \\
\rho \sigma_2 \\
\sigma_2 \sqrt{1 - \rho^2}
\end{pmatrix} \begin{pmatrix}
\sigma_1 & 0 \\
\rho \sigma_2 & \sigma_2 \sqrt{1 - \rho^2}
\end{pmatrix} = \begin{pmatrix}
\sigma_1^2 & \rho \sigma_1 \sigma_2 \\
\rho \sigma_1 \sigma_2 & \sigma_2^2
\end{pmatrix}
\]

13
3.13 In general

- In general, given any size of covariance matrix for any number of underlying assets it will be possible to generate correlated normally distributed random variables.

- For a covariance matrix for two underlyings the correlated random variables, \(y_1, y_2\), say were obtained from

\[
\begin{pmatrix}
y_1 \\
y_2
\end{pmatrix} = M \begin{pmatrix}
\phi_1 \\
\phi_2
\end{pmatrix}
\]

where \(M\) is the square root of the covariance matrix and \(\phi\) are the independent Normally distributed random variables.

- In general to create \(d\)-correlated Normally distributed variables \(y_1, \ldots, y_d\) from \(d\) uncorrelated variables \(\phi_1, \ldots, \phi_d\) and \(MM^T = \Sigma = \text{covariance matrix}\), use

\[
\begin{pmatrix}
y_1 \\
\vdots \\
y_d
\end{pmatrix} = M \begin{pmatrix}
\phi_1 \\
\vdots \\
\phi_d
\end{pmatrix}
\]

3.14 Cholesky factorisation

- In general calculating the matrix \(M\) is fairly straightforward, one way of doing so is by using a process called Cholesky factorisation.

- Cholesky factorisation is a special case of LU decomposition, specialised to symmetric, positive definite matrices

- See Paul Wilmott on Quantitative finance pp934-5 for the computer algorithm or alternatively Press et al., numerical recipes in Fortran/C etc.

3.15 Basic Monte-Carlo methods - summary

- Simple to program and to understand, they are ideal for a first approximation to a derivative value.

- Convergence is slow, and determined probabilistically which makes extrapolation impossible.

- Naturally designed for forward looking problems including path dependent derivatives such as lookback options and Asian options.

- Good for derivatives where there are multiple sources of uncertainty, as the computational effort only increases linearly.

- We have introduced Monte Carlo methods which are very closely related to the probabilistic solution, in that you use simulation to determine an expected value for the option in the future that can be discounted at the risk-free rate (according to the fundamental theorem) to obtain the value today.

- To simulate the paths we typically use the solution to the SDE or the Euler approximation, along with a decent generator of Normally distributed random variables.
- It is easy to apply to path dependent options and to options on more than one underlying, for these you need to know the covariance matrix and be able to ‘square root’ it by means of Cholesky factorisation to be able to perform the simulations.
4 Improvements on and extensions of Monte Carlo Methods

4.1 Overview
• Here we will extend our basic theory and concentrate on some simple techniques to improve the basic method.
• The techniques we will look at are using antithetic variables which is a very simple adjustment, as is the control variate technique. We will mention something about moment matching, importance sampling and low discrepancy sequences.
• Most of these are designed to reduce the variance of the error, except for low discrepancy sequences which are used to improve the convergence rate.

4.2 Improving Monte Carlo
• Monte Carlo is typically the simplest numerical scheme to implement but as you will see its accuracy and uncertain convergence is not ideal for accurate valuation.
• However, for complex problems with multiple Brownian motions it is often the only method that can be used which makes it important for us to improve the accuracy of the standard model.
• In addition to this, when there are multiple Brownian motions AND early exercise features it is crucial that there is an adaptation to the Monte Carlo method that allows us to compute an option value as it may well be the case that other numerical methods will not provide a reasonable estimate.

4.3 Antithetic variables
• Antithetic variables or antithetic sampling is a simple adjustment to generating the \( \phi_n \) (\( 1 \neq n \leq N \)).
• Instead of making \( N \) independent draws, you draw the sample in pairs; if the \( i \)th Normally distributed variable is \( \phi_i \), then choose \( \phi_{i+1} \) to be \(-\phi_i\), then draw again for \( \phi_{i+2} \)
• \(-\phi_i\) is also Normally distributed and most importantly the mean of the two draws is zero, and so this ensures that the mean of the sample paths will be correct and the distributions of draws will be symmetric.
• Thus if \( N = 500,000 \), you only need to make 250,000 random draws for \( \phi \) and use the negative of the draw to complete the required 500,000 values.
• This should improve the convergence as the distribution of paths is better matched to the model, i.e. the mean of \( \phi \) is zero

4.4 Control variate technique
• Control variate technique: This is explained through an example:
  - We want to compute \( E^Q[V(T)] \)
\[ V(T) = V(T) - V_1(T) + V_1(T), \]

where

\[ * \ E^Q[V_1(T)] \text{ is known analytically} \]
\[ * \ \text{and error in estimating } E^Q[V(T) - V_1(T)] \text{ by simulation is less than error in estimating } E^Q[V(T)] \]

- Then, a better estimate of \( E^Q[V(T)] \) is the sum of

\[ * \ \text{The known value of } E^Q[V_1(T)] \]
\[ * \ \text{Plus the estimate of } E^Q[V(T) - V_1(T)] \]

- Consider a basket option with payoff of \( V(T) = \max\left[ \frac{1}{2} S_1(T) + \frac{1}{2} S_2(T) - X, 0 \right] \) (assuming the BSM framework)

- A natural choice of control variate is

\[ V_1(T) = \max\left[ (S_1(T)S_2(T))^{\frac{1}{2}} - K, 0 \right] \]

- Why is this a good choice?

\[ * \ E^Q[V_1(T)] \text{ is known analytically as products and powers of lognormal r.v.'s are log-normal, so this is a similar calculation to Black Scholes} \]
\[ * \ \text{The difference } V(T) - V_1(T) \text{ is relatively small, and thus a Monte Carlo estimate of } E^Q[V(T) - V_1(T)] \text{ will have a relatively small error} \]

4.5 Moment matching

- One fairly simple strategy that works in a similar way to using antithetic variables is called moment matching.

- The typical way it is done is to ensure that the variance of the sample paths match the variance of the required distribution (antithetic variables will automatically ensure that the mean and skewness match).

- Our Brownian motion modelling requires the variance of \( \phi \) to be 1, so we would like the variance of our random \( \phi \)’s to share this property.

- To do this we first sample our \( N \phi \) values (requiring \( N/2 \) random numbers). Then calculate their variance, \( v \) say, now replace all of the \( \phi \) values with \( \phi \times v^{-\frac{1}{2}} \) and the variance of the new random draws is 1, as required.

4.6 Importance sampling

- The idea behind importance sampling is that if you know that the payoff function is zero outside of an interval \([a, b]\) then any draw which makes \( S_T \) lie outside of \([a, b]\) is wasted.

- Ideally, you would only to sample from distributions that cause \( S_T \) to lie in \([a, b]\) and then multiply the result by the actual probability of \( S_T \) being in this region.
• Recall that we have a function that turns a uniform random variable \([0, 1]\) into a realisation of \(S_T\) and so we can invert this map to find an interval \([x_1, x_2]\) that is mapped onto \([a, b]\), thus the probability of \(S_T\) being in \([a, b]\) is \(x_2 - x_1\).

• Thus to compute the expectation at time \(T\), draw variables from \([0, 1]\), multiply by \(x_2 - x_1\) and then add to \(x_1\) so that they are all in \([x_1, x_2]\) and then convert the \(x\) value into \(\phi\) as before and get a value of \(S_T\).

• Determine the option value \(V_T\) from \(S_T\) and then average the option values to obtain an expectation. Then multiply this expectation by \((x_2 - x_1)\).

• For example, \(S_0 = 100, X = 100, r = 0.05, \sigma = 0.2, T = 1\). Consider a call option and so for \([0, 100]\) the payoff contribution is zero. Thus we only need to sample in \([100, \infty)\). \(S_T = 100\) corresponds to a \(\phi\) value of:

\[
\phi = \frac{\log(100/100) - (r - \frac{1}{2}\sigma^2)}{\sigma} = -0.1
\]

but \(P(\phi = -0.1) = 0.461\). So \([0.461, 1]\) is the range of \(x\) values required.

• So every draw of \(x\) is multiplied by 0.539 and then added to 0.461 to obtain a \(\phi\) value and this only gives \(S_T\) values greater than 100.

4.7 Low discrepancy sequences

• One of the most useful practical techniques for improving the accuracy of Monte Carlo methods is by using Low discrepancy sequences (also known as Quasi Monte Carlo methods).

• The theory behind Monte Carlo techniques is that as you take more and more sample paths then they will eventually cover the entire distribution of \(S_T\) in the correct manner.

• Another way to think of this is that our random numbers drawn from \([0, 1]\) will eventually cover this interval in a uniform manner.

• Unfortunately, we cannot actually draw an infinite number of paths and for any size of \(N\) it may well be the case that our \(S_T\) values all cluster around particular values while missing out other regions of \(S\) space entirely.

• This problem becomes more pronounced as we increase the number of dimensions, \(d\)

• To overcome this problem we throw away the idea of using ‘random’ numbers at all and instead choose a deterministic sequence of numbers that does a very good job of covering the \([0, 1]\) interval.

• Note that as we have already discussed, most random number generators are deterministic to some extent and so this approach isn’t as odd as you would imagine.

• The most interesting thing about a low discrepancy series is that it can improve the convergence of the Monte Carlo method from \(1/N^{\frac{1}{2}}\) to \(1/N\), making the Monte Carlo method fully competitive with binomial lattices.

• We will deal with a simple sequence here but there is an extensive literature on low discrepancy series, see Jäckel, *Monte Carlo Methods in Finance* for an excellent summary
4.8 The Halton Sequence

- Sobol sequences are the most common of the low discrepancy sequences, but for ease of explanation we will consider the Halton sequence.

- The Halton sequence is a sequence of numbers \( h(i;b) \) for \( i = 1, 2, \ldots \) where \( b \) is the base and all of the numbers in the sequence are in \([0, 1]\). You can choose the base, let us select base 2.

- The Halton sequence is the reflection of the positive integers in the decimal point and is best observed through an example:

<table>
<thead>
<tr>
<th>Integers base 10</th>
<th>Integers base 2</th>
<th>Halton sequence base 2</th>
<th>Halton number base 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>( 1 \times \frac{1}{2} )</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>( 0 \times \frac{1}{2} + 1 \times \frac{1}{4} )</td>
<td>0.25</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>( 1 \times \frac{1}{2} + 1 \times \frac{1}{4} )</td>
<td>0.75</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>( 0 \times \frac{1}{2} + 0 \times \frac{1}{4} + 1 \times \frac{1}{8} )</td>
<td>0.125</td>
</tr>
</tbody>
</table>

- So \( 10 \rightarrow 0.01 \) in base 2 terms.

- In general the \( i \)th integer, \( i \), can be expressed as:

\[
i = \sum_{j=1}^{m} a_j b^{j-1}
\]

where \( b \) is the base and \( 0 \leq a_j < b \). The Halton numbers are given by:

\[
h(i;b) = \sum_{j=1}^{m} a_j b^{-j}
\]

- What is nice about the Halton sequence is that it fills the range \([0, 1]\) gradually

- When extending to \( d \) multiple dimensions you should choose \( d \) prime bases

4.9 American options

- One of the key unanswered questions in finance is how to value options with early exercise features by using a Monte Carlo method.

- Recall that the American option value \( V_0 \) is

\[
V_0 = \max_{\tau} \left[ E^Q_{\tau} \left[ e^{-r\tau} \max(S_{\tau} - X, 0) \right] \right]
\]

- The problem comes from the fact that Monte Carlo is a forward looking method. To use the sample paths we would have to test early exercise at each point in time on each sample path in order to determine what the optimal exercise strategy would be.

- This is incredibly time consuming and not practical, and simplistic approaches, such as the perfect foresight method where you simply choose the highest early exercise value during the lifetime of the option do not give acceptable approximations to the option value.
• If you follow the optimal strategy (shaded) then you should only exercise at expiry, but you can get a better payoff from exercising at $t_3$. This is perfect hindsight and does not value the option correctly.

4.10 Early attempts

• Tilley (1993) made the first effort to adapt Monte Carlo methods to cope with early exercise features. The method involves a technique known as “bundling” where the paths are constructed as usual and, at each timestep, they are bracketed into regions of asset price.

• At expiry the option price is the average of the payoff from all the paths in the bundle. Working backwards through time, as the paths are known at each point it can be checked whether or not it was, on average, worth exercising in each bundle by comparing this to the discounted option price from the future time step.

• There are evident shortcomings with this approach:
  – first, all the paths have to be stored, which can cause computer memory problems
  – more importantly the process overvalues the options.
  – Crucially, it is also very difficult to extend to options on multiple underlyings (usually the Monte Carlo method’s advantage over other numerical techniques).

• Realising the main drawback in Tilley’s method, Barraquand and Martineau (1995) adapted his approach so that the bundling was in terms of payoff value rather than underlying asset value. Payoff value only has one dimension and so extension to many underlyings does not create any undue problems.

• However, although not requiring as much memory as Tilley’s approach, the approach still does not converge to the correct value and always underestimates the option value (see Boyle et al., 1997) and this estimation error can be serious. This approach is extended by Raymar and Zwecher (1998) but is still less effective than the following two methods.
• Broadie and Glasserman (1997) approach the Monte Carlo method as by creating upper and lower bounds. To create these bounds they use a ‘bushy tree effect’ to pursue sub and superoptimal strategies. The superoptimal strategy is obtained by creating a tree whose possible next states are determined, by simulation, all the way to expiry.

• Then, in a similar vein to Tilley (1993) the option value at the previous time, \( t \), is the maximum of the average of the values at \( t + \Delta t \) discounted and the value from early exercise at \( t \). This strategy does assume the investor has some foresight and so overvalues the option.

• The suboptimal procedure entails using the \( b \) possible paths at each time and, for each path, using the remaining \( b - 1 \) paths to determine whether the option is continued or exercised. This exercise choice is then applied the initial path one was focusing on. All the possible combinations are then averaged at each timestep.

• This method is shown to be suboptimal, for more details see Broadie and Glasserman (1997). These can then be combined to provide bounds for the put option value.

### 4.11 Broadie and Glasserman (1997)

• As usual, the computational effort increases only linearly as more underlying assets are added. However, as the number of observation times is increased the calculations increase exponentially, i.e with \( n \) paths, \( d \) observation dates and \( b \) branches in the tree the effort is \( nbd \) (\( b \) here is typically quite large, e.g. 50)

• Thus, to estimate the value of a continuously, or even daily, observed option involves the use of extrapolation which is somewhat ad hoc if, as is the case here, the initial results are not converging at a known rate.

• There has been more recent work by Fu et al. (2001) who parametrise the early exercise curve by using Monte Carlo simulations and Rogers (2002) who uses a Lagrangian Martingale to achieve a close upper bound on the option value.

### 4.12 Recent advances

• The most popular method for incorporating early exercise features in the Monte Carlo methods is by Longstaff and Schwartz (2001) which we will see in detail in later. Its appeal is that it is simple to implement although there remain questions about its accuracy and efficiency.

• Another, more academically rigorous approach is the dual approach Haugh and Kogan (2004) which expresses the option pricing problem as a minimisation problem, from which a tight upper bound on the price is calculated. Unfortunately, its calculation is often problematic, although there are practical approaches to circumvent this see Andersen and Broadie (2004) for more details.

• The Carriere (1996) valuation of early exercise price for options using simulations and non-parametric regression is also highly regarded.

### 4.13 Overview

• We have looked through a variety of extensions to the standard Monte Carlo in an effort to reduce the variance of the error or to improve the convergence.
• Most of the improvements are simple to apply such as antithetic variables and moment matching, others are more complex such as low discrepancy sequences.

• Finally, we looked at some of the early attempts to use Monte Carlo methods to value American style options. This is a precursor to the Longstaff and Schwartz approach.
5 The binomial model

- A fundamental theorem of finance (in discrete time), also commonly known as The fundamental theorem of asset pricing, or The fundamental theorem of arbitrage pricing or The fundamental theorem of arbitrage-free pricing: if there are no arbitrage opportunities and markets are complete (i.e. all assets are replicable) then there exists a unique, risk-neutral, pricing measure.

- As such we can write the value of any asset, in particular an option at time $t$, $V_t$, as

$$V_0 = \frac{1}{1 + R^Q} E^Q[V_T]$$

where $R$ is the risk-free interest rate over time $T$. We can also write this in continuously compounded terms as

$$V_0 = e^{-rT} E^Q[V_T]$$

- We can apply the same argument from time period to time period and so it is possible to have binomial trees with multiple time steps to simulate the movement of the underlying asset more accurately.

- As we have more steps in our tree we essentially have a binomial distribution with more and more possible outcomes which should eventually approximate to the continuous, lognormal distribution we saw in our continuous time pricing models such as Black-Scholes.

5.1 Basic binomial set up

- If we have a three asset world with a Bond, $B_t$, a Stock, $S_t$ and a call option $C_t$, where interest rates are continuously compounded and the risk neutral probability of the up and down states occurring are $q$ and $(1 - q)$ then we have

\[
\begin{align*}
B_0 &= 1 \\
B_t &= e^{rT} \\
S_0 &= s \\
S_t &= us \\
B_T &= e^{rT} \\
S_T &= ds \\
C_T &= \max(us - X, 0) \\
C_0 &= ? \\
C_T &= \max(ds - X, 0)
\end{align*}
\]
Determining \( q, u \) and \( d \)

- As the probabilities are risk neutral we require that the expected return on the stock is the same as that of the risk-free bond, thus
  
  \[
  qsu + (1 - q)sd = se^{rT} \\
  qu + (1 - q)d = e^{rT}
  \]

- We would also like to match the variance of our returns to that from the data. From our continuous model we know that under the risk-neutral model
  
  \[
  dS = rSdt + \sigma SdX
  \]
  
  and we can solve this SDE to obtain
  
  \[
  S_T = s \exp[(r - \frac{1}{2}\sigma^2)T + \sigma \sqrt{T} \phi]
  \]

- Taking expectations we have for the continuous case:
  
  \[
  E[S_T] = s \exp(rT) \\
  E[(S_T)^2] = s^2 \exp[(2r + \sigma^2)T]
  \]
  
  and in the binomial case
  
  \[
  E[S_T] = s(qu + (1 - q)d) \\
  E[S_T^2] = s^2(qu^2 + (1 - q)d^2)
  \]
  
  thus
  
  \[
  e^{rT} = qu + (1 - q)d \\
  e^{(2r+\sigma^2)T} = qu^2 + (1 - q)d^2
  \]

- However, this still leaves us with one degree of freedom to determine all of \( q, u \) and \( d \) since there are 3 unknowns and only 2 equations.

5.2 Some possible models

- The two most popular models for binomial pricing are Cox, Ross and Rubinstein (1979) (CRR for short) whose extra degree of freedom is to set
  
  \[ ud = 1 \]

  thus

  \[
  u = e^{\sigma \sqrt{T}}, \quad d = e^{-\sigma \sqrt{T}}, \quad q = \frac{e^{rT} - d}{u - d}
  \]

- The other is Rendleman and Bartter (1979) who choose:
  
  \[
  q = \frac{1}{2}
  \]

  and so

  \[
  u = e^{(r - \frac{1}{2}\sigma^2)T + \sigma \sqrt{T}}, \quad d = e^{(r - \frac{1}{2}\sigma^2)T - \sigma \sqrt{T}}.
  \]
5.3 Constructing the tree

- So now we have expressions for \( u \) and \( d \) which will ensure that the binomial tree will approximate the continuous lognormal distribution which arises from the geometric Brownian motion assumptions.
- There are other ways to construct the tree if the underlying asset follows a different stochastic process but we do not consider those here.
- Now we turn our attention to valuing a European call option before considering natural extensions.
- The current value of the underlying asset is \( S_0 \), the time to expiry is \( T \), we have \( N \) time steps, the continuously compounded risk-free rate is \( r \), the volatility of the underlying asset is \( \sigma \) and the exercise price of the option is \( X \). The size of the time-step \( \Delta t = T/N \).
- If we denote the value of the underlying asset after timestep \( i \) and upstate \( j \) by \( S_{ij} \) and the option price by \( V_{ij} \) then we have that:

\[
S_{ij} = S_0 u^j d^{i-j}
\]

\[
V_{Nj} = \max(S_0 u^j d^{N-j} - X, 0)
\]

\[
V_{ij} = e^{-r\Delta t}(qV_{i+1,j+1} + (1-q)V_{i+1,j}) \quad \text{for} \quad i < N
\]

where \( q, u \) and \( d \) are selected according to your preferred model (CRR or alternative).

5.4 Example

- Consider a European call option where \( S_0 = 100, X = 100, T = 1, r = 0.06, \sigma = 0.2. \)
- Choosing the CRR tree we have

\[
u = e^{\sigma\sqrt{\Delta t}} = 1.1224
\]

\[
d = e^{-\sigma\sqrt{\Delta t}} = 0.8909
\]

\[
q = \frac{e^{r\Delta t} - d}{u - d} = 0.5584
\]

- Next we show the calculation of the European call option price using 3 time steps, where we end up with an option value of \$11.55.

5.5 American option valuation

- American options are call (put) options where it is possible to exercise early at time \( t \) to receive \( S_t - X \) (\( (X - S_t) \) for a put).
- We will consider the American option pricing problem from different perspectives for the three types of numerical methods. As noted already, it can be problematic to solve using forward induction methods such as Monte Carlo techniques.
- The problem is a free boundary or optimal stopping problem where the current option value \( V_t \) is given by

\[
V_t = \max_{\tau} E_t^Q[e^{-r(\tau-t)}\text{Payoff}(S_\tau)]
\]

where \( \tau \) denotes the continuum of possible stopping times.
- This representation is not particularly useful when attempting to value the option.
5.6 American options and lattices

- Obviously any rational investor would only exercise if the value from exercising is greater than the value from not exercising, i.e. holding the option for one more period.

- The nice thing about binomial lattices is that as we calculate backward we already know the value of holding the option until the next period (the continuation value) and we know the early exercise value (the payoff from the option) and so it is straightforward to adapt our European option pricing model to deal with American options.

- Thus at each node in the tree we need to compare two values, the continuation (or hold) value, \( V_{hij} \) and the early exercise value \( V_{xij} \) to determine \( V_{ij} \) where

\[
V_{ij} = \text{Payoff}(S_{Nj}) \quad \text{for} \quad t = T
\]

\[
V_{hij} = e^{-r\Delta t}(qV_{i+1,j+1} + (1-q)V_{i+1,j}) \quad \text{for} \quad t < T
\]

\[
V_{xij} = \text{Payoff}(S_{ij}) \quad \text{for} \quad t < T
\]

\[
V_{ij} = \max(V_{hij}, V_{xij}) \quad \text{for} \quad t < T
\]

and Payoff is the appropriate payoff function for each option.

5.7 Example

- Consider an American put option where \( S_0 = 100 \), \( X = 100 \), \( T = 1 \), \( r = 0.06 \), \( \sigma = 0.2 \).

- Choosing the CRR tree we have

\[
u = e^{\sigma \sqrt{\Delta t}} = 1.1224
\]

\[
d = e^{-\sigma \sqrt{\Delta t}} = 0.8909
\]

\[
q = \frac{e^{\sigma \Delta t} - d}{u - d} = 0.5584
\]
• Next we show the calculation of the American put option price where we end up with an option value of $6.099.

• The nodes in red denote that the holder of the option exercised early.

\[
\begin{array}{c|c|c|c|c}
\text{time} & \text{node} & \text{stock price} & \text{put option value} & \text{call option value} \\
\hline
0 & S_{1,0} = 112.24 & V_{1,0} = 11.509 & X_{1,0} = 10.91 & V_{1,1} = 2.043 \\
1/3 & S_{2,1} = 125.98 & V_{2,1} = 4.7199 & X_{2,1} = 10.91 & V_{2,2} = 0.000 \\
2/3 & S_{3,2} = 125.98 & V_{3,2} = 0.000 & X_{3,2} = 10.91 & V_{3,3} = 0.000 \\
1 & S_{4,3} = 141.40 & V_{4,3} = 0.000 & X_{4,3} = 10.91 & V_{4,4} = 0.000 \\
\end{array}
\]

5.8 Additional notes

• For American call options with no dividends it is never optimal to exercise early.

• From the lattice we can determine the early exercise region, the values of \( S \) and \( t \) for which you would exercise and the early exercise boundary which separates the exercise and non-exercise regions.

• Technically what we have evaluated here is a Bermudan option, which is an American option that can only be exercised on certain specified dates. However, as we have more and more observation dates then this value will approach the American option price.

5.9 Continuous dividends

• The fundamental theorem of finance does not directly apply to dividend paying assets but it can be easily adjusted to do so.

• If \( S_t \) is the value of an asset at time \( t \) which pays out a continuously compounded dividend yield, \( \delta \), then consider a new asset \( X \) which is defined as

\[
X_0 = e^{-\delta t} S_0
\]

at time \( t \) the \( S_0 \) will have grown to \( S_t e^{\delta t} \) and so \( X_t = S_t \). Thus it will be possible to replicate the value of an option expiring at time \( t \) by holding \( e^{-\delta t} \) of the underlying asset.
• Thus by the fundamental theorem of finance it will be \( X \) which is priced under the risk-neutral measure given a known future asset price \( S_t \) thus:

\[
X_0 = e^{-rt} E_Q^0 [S_t]
\]

and so

\[
S_0 = e^{-(r-\delta)t} E_Q^0 [S_t]
\]

5.10 Lattices with continuous dividends

• Now given this slightly different calculation we have new values of \( u \) and \( d \) where

\[
E[S_T] = s \exp[(r - \delta)T]
\]

\[
E[(S_T)^2] = s^2 \exp[(2(r - \delta) + \sigma^2)T]
\]

and again you have another choice of a degree of freedom and the CRR approach gives:

\[
u = e^{\sigma \sqrt{\Delta t}}, \quad d = e^{-\sigma \sqrt{\Delta t}}, \quad q = \frac{e^{(r-\delta)\Delta t} - d}{u - d}
\]

in a tree with step size \( \Delta t \)

• Thus if we consider the American put option only now when \( \delta = 0.03 \) we see that the theoretical price is now $7.32

\[
\begin{array}{c}
\text{\( t = 0 \)} \quad \text{\( t = 1/3 \)} \quad \text{\( t = 2/3 \)} \quad \text{\( t = 1 \)} \\
S_{0,0} = 100.00 & S_{1,0} = 89.094 & S_{1,1} = 92.79 & S_{1,2} = 141.40 \\
V_{0,0} = 0.00 & V_{1,0} = 12.43 & V_{2,0} = 5.189 & V_{3,0} = 20.62 \\
V_{0,1} = 7.159 & V_{1,1} = 19.43 & V_{2,1} = 26.5 & V_{3,1} = 29.28 \\
V_{0,2} = 0.00 & V_{1,2} = 10.91 & V_{2,2} = 7.00 & V_{3,2} = 29.28 \\
\end{array}
\]

• Perhaps a more realistic case is when there is a known discrete dividend payment at a certain point in time. In our example, imagine there is a known dividend, payable after 2/3 of a year which is 3% of the share price.
• Here the fundamental theorem will hold from period to period and our values of \( u \), \( d \) and \( q \) will remain the same as for the no dividend case but at \( t = 2/3 \), \( S_{2j} \rightarrow 0.97 \times S_{2j} \).

• This is depicted in the worked example next:

<table>
<thead>
<tr>
<th>( t = 0 )</th>
<th>( t = 1/3 )</th>
<th>( t = 2/3 ) Pre Div</th>
<th>( t = 2/3 ) Post Div</th>
<th>( t = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_{10} = 112.24 )</td>
<td>( V_{10} = 0.000 )</td>
<td>( V_{11} = 2.544 )</td>
<td>( V_{12} = 2.544 )</td>
<td>( V_{13} = 2.544 )</td>
</tr>
<tr>
<td>( V_{10} = 10.91 )</td>
<td>( V_{11} = 13.172 )</td>
<td>( V_{12} = 20.62 )</td>
<td>( V_{13} = 23.00 )</td>
<td>( V_{14} = 23.00 )</td>
</tr>
<tr>
<td>( S_{20} = 79.38 )</td>
<td>( S_{21} = 5.877 )</td>
<td>( V_{20} = 108.87 )</td>
<td>( V_{22} = 0.000 )</td>
<td>( V_{23} = 0.000 )</td>
</tr>
<tr>
<td>( V_{20} = 13.172 )</td>
<td>( V_{21} = 3 )</td>
<td>( V_{22} = 0.000 )</td>
<td>( V_{23} = 0.000 )</td>
<td>( V_{24} = 0.000 )</td>
</tr>
<tr>
<td>( S_{30} = 68.60 )</td>
<td>( S_{31} = 31.40 )</td>
<td>( V_{30} = 3.140 )</td>
<td>( V_{32} = 31.40 )</td>
<td>( V_{33} = 31.40 )</td>
</tr>
</tbody>
</table>

**Bold** figures denote value if held until after the dividend date.

5.11 Cash dividends

• Non-proportional cash dividends can be problematic as this leads to a non-recombining tree as if the dividend is $2 regardless of the share price then an up move followed by a down move will not be the same as a down move followed by an up move.

• This leads to a large increase in the computational effort, which will increase exponentially after the cash dividend payment date.

• There is an adjustment for European options but this is not of great practical use as Black-Scholes can be used quite simply in the European case.

• In American option cases the simplest approach is to use interpolation, although this is more naturally done in a finite difference setting as we will see later.

5.12 Overview

• We have developed a multistep binomial lattice which will approximate the value of a European or American call option when the underlying asset pays out dividends.

• The construction comes from an extension to the fundamental theorem of finance and you have a choice of parameters which are typically chosen to fit the binomial distribution to the Black-Scholes lognormal distribution.

• The most useful outcome is the ability to price American options easily.
6 Analysis of binomial option pricing

6.1 Overview

- Having introduced how to value European and American options on dividend paying underlying assets we now look at the accuracy of the binomial method.
- In particular we consider the difference between ‘distribution error’ and ‘non-linearity’ error and the difficulty in ensuring monotonic convergence.
- We extend the analysis to look at the computational effort when valuing an option on two underlying assets and a simple method for reducing the computational effort here.

6.2 Convergence

- When analysing convergence we need to consider the error from a numerical scheme, if $V_{\text{exact}}$ is the correct option value and $V_n$ is the value from a binomial tree with $n$ steps then:

$$\text{Error}_n = V_{\text{exact}} - V_n$$

- To formally define convergence, there exists a constant, $\kappa$, such that for all time steps, $n$, where $c$ is the order of convergence. This can also be written as

$$\text{Error}_n = O\left(\frac{1}{n^c}\right)$$

- As long as $c > 0$ then $V_n$ will converge to $V_{\text{exact}}$.

6.3 Convergence for trees

- Unfortunately there is not a simple proof to show the convergence of the binomial lattice to the correct option price.
- When considering European options it is simple to look at this empirically because we have an analytic expression for $V_{\text{exact}}$ (the Black-Scholes price). By the Central Limit Theorem we also know that the prices will eventually converge as the binomial distribution converges to the lognormal distribution.
- All empirical evidence indicates that for all basic binomial models (CRR, RB etc) $c = 1$, or that $V_n$ converges to the Black Scholes at a rate of $1/n$. So, in general to halve the error you must double the number of time steps.
- For a rigorous proof see Leisen and Reimer (Applied Mathematical Finance, 1996).

6.4 Empirical evidence

- The diagram (from Leisen and Reimer, 1996) shows the error from the CRR model relative to $1/n$, where the upper line denotes how the error would reduce with $1/n$ convergence and the sawtooth pattern is the actual error
- We would like this convergence to be monotonic for two reasons.

30
First, we would like to know that as we construct a lattice with more steps we will get closer to the exact answer. This is especially important when we have no analytic value for the exact answer.

Second, if the problem is computationally intensive, we can save effort by using extrapolation procedures.

- As we see from the graph on the previous figure the convergence to the exact option value looks far from monotonic for the binomial lattice. We would like to investigate why this is the case...

6.5 Extrapolation

- If convergence to the exact option value is monotonic and at a known rate then there is a simple extrapolation technique. Consider the following equations for lattices with different numbers of time steps

\[ V_{\text{exact}} = V_n + \kappa n + o\left(\frac{1}{n}\right) \]

\[ V_{\text{exact}} = V_m + \kappa m + o\left(\frac{1}{m}\right) \]

- We can use these simultaneous equations to determine \( \kappa \) and thus know the first error terms and thus improve the accuracy of the method. The equation becomes

\[ V_{\text{exact}} = \frac{nV_n - mV_m}{n - m} + o\left(\frac{1}{m - n}\right) \]

6.6 Sawtooth Effect

- For a European option, when we increase \( n \) and plot \( \text{Error}_n \) against \( n \) we see the following shape:

- We see two distinct features, the first is a sawtooothing and the second is periodic humps.

- The sawtooothing is known as the ‘odd-even effect’ (Omberg, Advances in Futures and Options research, 1987) where as you move from say 25 steps to 26 steps the change in \( V_n \) is very large. This happens as the final nodes in the lattice move relative to the exercise price of the option, where there is a discontinuity in the option price.
- The periodic humps are also a result of this as (unless \( S = X \)) the position of the nodes relative to the exercise price change as you increase the number of time steps \( n \).

- The following explains the odd-even effect: The binomial approximation to the normal is depicted for lattices with 5 and 6 steps. The shading denotes which nodes contribute value to the option price if \( X = 100 \).

6.7 Explanation of periodic ringing

- Due to the discontinuity in the option payoff, the location of the final nodes are very important in determining \( V_n \)

- In the first diagram, the node at 110 contributes an option value of 10 with a large probability and so this lattice overvalues the European option. However, in the second diagram the node at 100, contributes nothing to the option value and so this lattice undervalues the option.
• The periodic humps can also be demonstrated to be connected to the position of the binomial nodes. If we introduce a measure \( \Lambda \) denoted by

\[
\Lambda = \frac{S_k - X}{S_k - S_{k-1}}
\]

where \( S_k \) is the closest node above the exercise price and \( S_{k-1} \) is the node below the exercise price. The next diagram plots \( \Lambda \) against the error from the binomial lattice (with only even steps to remove the odd-even effect) The dashed lines denote the error and the solid lines the corresponding value of \( \Lambda \). Only even numbers of steps were considered.

6.8 Types of Error

• Figlewski (Journal of Financial Economics, 1999) introduced a definition to distinguish between the two types of error that one observes when pricing derivatives using binomial lattices.

• First there is ‘distribution error’ which arises from the binomial distribution only providing an approximation to the lognormal distribution. This is the error that reduces at \( 1/n \), this can typically be reduced by extrapolation techniques.

• Second, and more importantly there is ‘non-linearity error’. This arises from not having the nodes in the tree or grid aligned correctly with the features for the option. For example, the strike price in a vanilla European. This can cause serious errors for more exotic options, especially barriers and lookbacks

6.9 Removing non-linearity error

• In the case of European options the most elegant way of overcoming non-linearity error is Leisen and Reimer (1996), they use the degree of freedom in selecting \( q \), \( u \) and \( d \) so that the lattice is centred around the exercise price \( X \), to ensure that the non-linearity error is removed (or remains constant).

• The choices which do this are as follows where \( N \) is the total number of time steps.

\[
q = h(d_2)
\]
\begin{align*}
u &= e^{(r-\delta)\Delta t}q^* / q \\
d &= \frac{e^{(r-\delta)\Delta t} - qu}{1 - q}
\end{align*}

where
\[d_{1,2} = \frac{\log(S/X) + (r - \delta \pm \frac{1}{2}\sigma^2)(T - t)}{\sigma\sqrt{T - t}}\]

\[h(x) = \frac{1}{2} + \sqrt{\frac{1}{4} - \frac{1}{4} \exp[-\left(\frac{x}{N + \frac{1}{6}}\right)^2(N + \frac{1}{6})]}\]

\[q^* = h(d_1)\]

- This method, although seemingly complex is very simple to program (compared to Wid- dicks et al., Journal of Futures markets, 2002) and provides very accurate option values.
- The convergence is smooth and so is amenable to standard extrapolation techniques.
- Unfortunately, it was designed specifically to price European options, for which we already have analytic solutions, the real test will come when evaluating American options.

### 6.10 What about Americans

- The issue of nonlinearity error is more complex in American options as we have to worry about more than simply the discontinuous payoff. At every time step there is also the early exercise boundary (which separates exercise from non-exercise).
- We do not know where this boundary will be a priori and so naturally the binomial lattice will not be constructed to remove the nonlinearity error from the early exercise boundary.
- There are many approaches to improving the standard CRR method for valuing American options, two of which are detailed here.
- The Leisen and Reimer approach for European options also works well for American options as the largest nonlinearity error (from the discontinuous payoff) has been entirely removed. Thus, this method still provides accurate American option values and is simple to program.
- An alternative method for pricing American options is provided by Broadie and Detemple (Review of Financial Studies, 1996) who avoid the problem of the discontinuous payoff by using a combination of the Black-Scholes formula for a European option and the CRR binomial lattice.
- The idea is that between the penultimate timestep and expiry the continuation value of the American option is a European option with time to expiry $\Delta t$. So you can calculate the American option values at $T - \Delta t$ precisely without having any nonlinearity error from the discontinuous payoff.
- If we have an $n$ step tree with $u$, $d$ and $q$ as in CRR the Broadie and Detemple method suggests the following algorithm for pricing an American put option with $N$ time steps and time to expiry $T$ (ERROR):

\[S_{i,j} = S_0 u^i d^{N-i} \]
\[ V_{N,j} = \max(X - S_{N,j}, 0) \]
\[ V_{N-1,j} = \max(V_{h,N-1,j}, V_{x,N-1,j}) \]
\[ V_{h,N-1,j} = BS(S_{N-1,j}, (N - 1)\Delta t) \]
\[ V_{x,N-1,j} = X - S_{N-1,j} \]
\[ V_{i,j} = \max(V_{h,i,j}, V_{x,i,j}) \quad \text{for} \quad i < N - 1 \]
\[ V_{h,i,j} = e^{-r\Delta t}(qV_{i+1,j+1} + (1-q)V_{i+1,j}) \quad \text{for} \quad i < N - 1 \]
\[ V_{x,i,j} = X - S_{i,j} \quad \text{for} \quad i < N - 1 \]
\[ BS(S, t) = X e^{-(T-t)}N(-d_2) - S e^{-\delta(T-t)}N(d_2) \]
\[ d_{1,2} = \frac{\log(S/X) + (r - \delta \pm \frac{1}{2}\sigma^2)(T-t)}{\sigma \sqrt{T-t}} \]

6.11 Exotic options

- The issue of nonlinearity error can become more pronounced for options with exotic features.

- A particular example is barrier options. Typically as more and more sources of nonlinearity error are introduced it becomes increasingly difficult to adapt the binomial lattice to provide monotonic convergence.

- Often this is not problematic as due to increasing computing power one can just use a lattice with enough time steps to overcome the problem (such as with American options).

- However, if the problem has multiple stochastic variables (such as stochastic volatility) or an interest rate derivative with a sophisticated term structure model then nonlinearity error can be a real problem.

6.12 More than one underlying asset

- There are many derivative pricing problems that require modelling more than one stochastic variable. These could be problems where we consider stochastic volatility or when the payoff from the derivative is a function of two or more underlying assets.

- We have seen exchange options, but there are also basket options, best of options, chooser options and a whole host of exotic derivatives which require such modelling.

- Here we focus on one lattice approach to valuing such options when there are two underlying assets. This approach can be generalised to any number of assets (see Kamrad and Ritchken, Management Science, 1991 amongst others).
6.13 Boyle, Evnine and Gibbs

- The model presented here is from Boyle, Evnine and Gibbs (Review of Financial Studies, 1989). They assume that we have two assets both of which follow geometric Brownian motion as before:
  \[
  dS_1 = \mu_1 S_1 dt + \sigma_1 S_1 dX_1 \\
  dS_2 = \mu_2 S_2 dt + \sigma_2 S_2 dX_2
  \]
  where \(X_2 = \rho_2 X_1 + \sqrt{1-\rho^2}X_3\)

- To discretise this problem they consider a tree where at each time step the underlying asset prices \((S_1, S_2)\) can both move up or down, giving four possible states as in the table:

<table>
<thead>
<tr>
<th>Nature of jumps</th>
<th>Probability</th>
<th>Asset prices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up, up</td>
<td>(p_{uu})</td>
<td>(S_{1u}, S_{1u})</td>
</tr>
<tr>
<td>Up, down</td>
<td>(p_{ud})</td>
<td>(S_{1u}, S_{1d})</td>
</tr>
<tr>
<td>Down, up</td>
<td>(p_{du})</td>
<td>(S_{1d}, S_{1u})</td>
</tr>
<tr>
<td>Down, down</td>
<td>(p_{dd})</td>
<td>(S_{1d}, S_{1d})</td>
</tr>
</tbody>
</table>

- In a similar way as for one underlying asset, they ensure that the first two moments of the distributions match and use a CRR analogy which states:
  \[
  u_i d_i = 1 \\
  u_i = e^{\sigma_i \sqrt{\Delta t}}
  \]
  This gives rise to the following probabilities:
  \[
  p_1 = \frac{1}{4}[1 + \rho + \sqrt{\Delta t}(\frac{r - \frac{1}{2}\sigma_1^2}{\sigma_1} + \frac{r - \frac{1}{2}\sigma_2^2}{\sigma_2})] \\
  p_2 = \frac{1}{4}[1 - \rho + \sqrt{\Delta t}(\frac{r - \frac{1}{2}\sigma_1^2}{\sigma_1} - \frac{r - \frac{1}{2}\sigma_2^2}{\sigma_2})] \\
  p_3 = \frac{1}{4}[1 - \rho + \sqrt{\Delta t}(-\frac{r - \frac{1}{2}\sigma_1^2}{\sigma_1} + \frac{r - \frac{1}{2}\sigma_2^2}{\sigma_2})] \\
  p_4 = \frac{1}{4}[1 + \rho + \sqrt{\Delta t}(-\frac{r - \frac{1}{2}\sigma_1^2}{\sigma_1} - \frac{r - \frac{1}{2}\sigma_2^2}{\sigma_2})]
  \]

**Computational effort**

- In the binomial lattice for one underlying asset at each time step \(i\) there are \(i + 1\) nodes giving \((N + 1)(N + 2)/2\) total calculations in an \(N\)-step lattice.
- As we move to the two underlying model then each step has \((i + 1)^2\) nodes giving \((N + 1)^2(N + 2)^2/4\) total calculations, which is the square of the effort in the one underlying case.
- As you introduce \(k\) underlying assets the the total number of calculations grows exponentially to \((N + 1)^k(N + 2)^k/2^k\) which can become a very large number.
- Typically due to memory constraints it is difficult to get reasonable accuracy with more than 5 underlying assets or sources of uncertainty...
6.14 ‘Work’

- So as to compare the strengths of different numerical methods Broadie and Detemple (Management Science, 2004) introduce the idea of representing the convergence as a function of work which is the computational effort required.

- Thus for a lattice with \( N \) time steps and \( d \) underlying assets the work \( w \) is approximately \( N^{d+1} \) and the convergence is at the rate of \( 1/N \) and so the convergence can be seen as \( O(w^{-1/d+1}) \).

- With Monte-Carlo methods this is \( (O(w^{-1/2})) \) and finite-difference methods \( (O(w^{-2/d+1})) \).

6.15 Curtailed range

- For most options (especially American options) in more than one underlying asset a simple way of reducing the computational effort is simply to ignore the vast majority of lattice calculations.

- In their curtailed range method Andricopoulos et al., (Journal of Derivatives, 2004) showed that for options on just one underlying with 1000 steps, the time saving was 87%, for options on three underlying assets with 100 steps the time saving was 91%.

- The idea is that in large lattices many of the calculations are superfluous as they represent scenarios where the underlying asset price has moved in excess of ten standard deviations and so contribute nothing to the value of the option.

6.16 Overview

- We have analysed the binomial pricing model in detail, in general it converges at the rate of \( 1/N \) where \( N \) is the number of time-steps in the tree.

- However, this convergence is often non-monotonic due to nonlinearity error caused by discontinuities in the option price. This can be illustrated by considering the discontinuous payoff from a European call or put option.

- There are methods of overcoming this, and it is particularly important for American options where there is no analytic solution.

- Finally, we analysed how to construct a lattice for more than one underlying asset and how this effects the computational effort or work.
7 Finite-difference methods

7.1 Overview

- We now introduce the final numerical scheme which is related to the PDE solution.
- Finite difference methods are numerical solutions to (in CF, generally) parabolic PDEs. They work by approximating the derivatives at each point in time and then rearranging the equations to solve backward in time.
- There are three types of methods - the explicit method, which is analogous to the trinomial tree, the implicit method (which you would never use!) and the Crank-Nicolson method which has the best convergence characteristics.

7.2 Finite-difference approximations

- Consider a function of two variables $V(S, t)$, if we consider small changes in $S$ and $t$ we can use a Taylor’s series to express $V(S + \Delta S, t)$, $V(S - \Delta S, t)$, $V(S, t + \Delta t)$ as follows (all the derivatives are evaluated at $(S, t)$)

\[
V(S + \Delta S, t) = V(S, t) + \Delta S \frac{\partial V}{\partial S} + \frac{1}{2}(\Delta S)^2 \frac{\partial^2 V}{\partial S^2} + O((\Delta S)^3) \quad (1)
\]

\[
V(S - \Delta S, t) = V(S, t) - \Delta S \frac{\partial V}{\partial S} + \frac{1}{2}(\Delta S)^2 \frac{\partial^2 V}{\partial S^2} + O((\Delta S)^3) \quad (2)
\]

\[
V(S, t + \Delta t) = V(S, t) + \Delta t \frac{\partial V}{\partial t} + \frac{1}{2}(\Delta t)^2 \frac{\partial^2 V}{\partial t^2} + O((\Delta t)^2) \quad (3)
\]

- In order to use a finite difference scheme we need to use these expansions to approximate the first and second derivatives with respect to $S$ and $t$.

- For $S$, we have two options for the first derivative:
  - From the first (or third) equation:

\[
\frac{\partial V}{\partial S}(S, t) = \frac{V(S + \Delta S, t) - V(S, t)}{\Delta S} + \frac{1}{2} \Delta S \frac{\partial^2 V}{\partial S^2} + O((\Delta S)^2)
\]

\[
= \frac{V(S + \Delta S, t) - V(S, t)}{\Delta S} + O(\Delta S)
\]

  - From equations (1) and (2):

\[
\frac{\partial V}{\partial S}(S, t) = \frac{V(S + \Delta S, t) - V(S - \Delta S, t)}{2\Delta S} + O((\Delta S)^2)
\]

- For the second derivative we use equations (1) and (2) to get:

\[
\frac{\partial^2 V}{\partial S^2}(S, t) = \frac{V(S + \Delta S, t) - 2V(S, t) + V(S - \Delta S, t)}{(\Delta S)^2} + O((\Delta S)^2)
\]
For $t$ we have

\[
\frac{\partial V}{\partial t} (S, t) = \frac{V(S, t + \Delta t) - V(S, t)}{\Delta t} + \frac{1}{2} \Delta t \frac{\partial^2 V}{\partial t^2} + O((\Delta t)^2)
\]

\[
= \frac{V(S, t + \Delta t) - V(S, t)}{\Delta t} + O(\Delta t)
\]

This particular approximation is called forward differencing whilst the preferred method for $S$ is called central differencing. In general central differencing (when appropriate) is the most accurate.

7.3 How does this help us?

- Reconsider the Black-Scholes equation and in particular the Black-Scholes equation for a European options where there are continuous dividends:

\[
\frac{\partial V}{\partial t} + \frac{1}{2} \sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r - \delta) S \frac{\partial V}{\partial S} - rV = 0
\]

- The boundary conditions are:
  - For a call:
    \[
    V(S, T) = \max(S - X, 0) \\
    V(0, t) = 0 \\
    V(S, t) \to Se^{-\delta(T-t)} - Xe^{-r(T-t)} \quad \text{as} \quad S \to \infty
    \]
  - For a put:
    \[
    V(S, T) = \max(X - S, 0) \\
    V(0, t) = Xe^{-r(T-t)} \\
    V(S, t) \to 0 \quad \text{as} \quad S \to \infty
    \]

- We will now form a finite difference grid that describes the $S - t$ space in which we need to solve the Black-Scholes equation.

- For a numerical method we need to truncate the range of $S$ to $[S^L, S^U]$ where $S^L$ is typically chosen to be zero and $S^U$ needs to be sufficiently large.

7.4 Constructing the grid

- We now need to ensure that we have a fine enough grid to allow for most possible movements in $S$ and enough time steps $t$.

- As for the binomial and Monte-Carlo method we will discuss later what is a suitable size/number for these steps.

- We partition the interval $[0, S^U]$ into $j_{max}$ subintervals each of length $\Delta S = S^U/j_{max}$, thus the endpoints of the intervals (or grid nodes) are 0, $\Delta S$, $2\Delta S$, $\ldots$, $(j_{max} - 1)\Delta S$, $j_{max}\Delta S = S^U$.
We also partition the interval $[0, T]$ into $imax$ subintervals each of length $\Delta t = T/imax$, thus the nodes on the grid are $0, \Delta t, \ldots, (imax - 1)\Delta t, imax\Delta t = T$.

We will denote the option price at each node $V(j\Delta S, i\Delta t)$ as $V_{j}^{i}$

Focus attention on $i, j$th value $V_{j}^{i}$, and a little piece of the grid around that point
7.5 Using equations

- We clearly know the information at \( t = T \) as this is the payoff from the option, by limiting our focus on

\[
\begin{align*}
V_{j+1}^i & \\
V_j^i & \\
V_{j-1}^i &
\end{align*}
\]

we can approximate the derivatives in the Black-Scholes equation by using our difference equations and from this we can write \( V_j^i \) in terms of the other three terms.

- Recall the BSM equation

\[
\frac{\partial V}{\partial t} + \frac{1}{2} \sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r - \delta)S \frac{\partial V}{\partial S} - rV = 0
\]

7.6 Explicit finite difference method

- The BSM equation approximates to

\[
\frac{V_{j+1}^i - V_j^i}{\Delta t} + \frac{1}{2} \sigma^2 j^2 (\Delta S)^2 \frac{V_{j+1}^{i+1} - 2V_j^{i+1} - V_{j-1}^{i+1}}{(\Delta S)^2} + (r - \delta)j \Delta S \frac{V_{j+1}^{i+1} - V_{j-1}^{i+1}}{2\Delta S} - rV_j^i = 0
\]

the unknown here is \( V_j^i \) as we have been working backward in time. So we can rearrange in terms of this unknown:

\[
V_j^i = \frac{1}{1 + r\Delta t} (AV_{j+1}^{i+1} + BV_j^{i+1} + CV_{j-1}^{i+1}) \quad (*)
\]

where:

\[
A = \left( \frac{1}{2} \sigma^2 j^2 + \frac{1}{2} (r - \delta)j \right) \Delta t \\
B = 1 - \sigma^2 j^2 \Delta t \\
C = \left( \frac{1}{2} \sigma^2 j^2 - \frac{1}{2} (r - \delta)j \right) \Delta t
\]

- Thus just as with a binomial tree we have a way of calculating the option value at expiry - the known payoff, and we have scheme for calculating the option value for all values of \( S \) at the previous time step.

- Thus we can use the backward scheme and equation (*) for \( V_j^i \) to calculate the option value all the way back to \( t = 0 \).

- The differences between the binomial and explicit finite difference method are
  - the binomial uses two nodes to the explicit finite difference’s three.
  - You get to choose the specifications of the grid in the finite difference method
  - You also need to specify the behaviour on the upper and lower \( S \) boundaries.
The grid again:

7.7 Upper and Lower boundaries

- If we attempt to use equation (*) to calculate \( V^i_0 \) then we need to have values of \( V^{i-1} \) which we don’t have (e.g. for calls):

- So for \( V^i_0 \) and \( V^i_{j_{max}} \) we need to use our boundary conditions.

\[
\begin{align*}
V^i_0 &= 0 \\
V^i_{j_{max}} &= S^u e^{-\delta(T-i\Delta t)} - X e^{-r(T-i\Delta t)}
\end{align*}
\]

- These conditions will naturally be different for different options, such as barrier options, put options etc.

- It is possible to have derivative conditions on these boundaries, as you can approximate them using \( V^i_1 \) etc. For example if at \( S = S^u \) \( \frac{\partial V}{\partial S} = e^{-\delta(T-i\Delta t)} \) then this ‘translates’ to

\[
\frac{V^i_{j_{max}} - V^i_{j_{max}-1}}{\Delta S} = e^{-\delta(T-i\Delta t)}
\]

7.8 Probabilistic interpretation

- You will see that it is possible to think of the explicit finite difference scheme as a trinomial tree and \( A, B \) and \( C \) as probabilities.

- First note that \( A + B + C = 1 \), second consider what the expected value of \( S \) is at time \( i\Delta t \):
\[ E[S^i_j] = \frac{1}{1 + r\Delta t}(A(S^i_j + \Delta S) + B(S^i_j) + C(S^i_j - \Delta S)) \]
\[ = \frac{1}{1 + r\Delta t}(S^i_j (1 + (r - \delta)\Delta t)) \]
\[ = \frac{1}{1 + r\Delta t}E[S^i_{j+1}] \]

the expected future value of \( S \), following GBM, under the risk-neutral probability discounted at the risk-free rate. So \( A, B \) and \( C \) can also be interpreted as risk-neutral probabilities (you can also check that the variance works).

7.9 Stability

• Unfortunately, the explicit finite difference scheme is occasionally unstable, in that for particular choices of \( \Delta t \) and \( \Delta S \), the scheme will not give an option value even close to the correct answer as small errors magnify during the iterative procedure.

• There is a mathematical method that can be used to determine what the constraint is, however, we can also appeal to our probabilistic explanation to see what the constraint is for the explicit finite difference method.

• If we consider \( A, B \) and \( C \) as probabilities, we require that \( A, B, C \geq 0 \).

• For \( A \) and \( C \) this requires:
  \[ j > \left| \frac{r - \delta}{\sigma^2} \right| \]
  which is not too large a constraint unless \( \sigma \) is very small and \( |r - \delta| \) is very large, which are rare.

• A far bigger problem is for \( B \) where this says that
  \[ \Delta t < \frac{1}{\sigma^2 j^2} \]
  which means that you need to ensure that the time interval is small enough. Specifically the required size of the time interval will shrink as you increase the number of \( S \) steps or the volatility increases.

• The stability often severely restricts choice of \( \Delta t, \Delta S \)
  – \( \Delta t \) cannot be too small, or else computation will take too long
  – then this puts lower bound on size of \( \Delta S \)

• Nonetheless, we have some flexibility. Two common choices:
  – choose \( \Delta t, \Delta S \) so that \( B = 2/3 \) (means \( A, C \) approx. 1/6)
  – choose \( \Delta t, \Delta S \) so that \( B = 1/3 \) (means \( A,C \) approx. 1/3)
7.10 Convergence

- Assuming that the scheme is stable then we would like to analyse the accuracy of the method.
- The errors will arise from only approximating the derivatives, in particular, in the explicit finite difference method:

\[
\frac{\partial^2 V}{\partial S^2}(S,t) = \frac{V(S + \Delta S, t) - 2V(S, t) + V(S - \Delta S, t)}{(\Delta S)^2} + O((\Delta S)^2)
\]

\[
\frac{\partial V}{\partial S}(S,t) = \frac{V(S + \Delta S, t) - V(S - \Delta S, t)}{2\Delta S} + O((\Delta S)^2)
\]

\[
\frac{\partial V}{\partial t}(S,t) = \frac{V(S, t + \Delta t) - V(S, t)}{\Delta t} + O(\Delta t)
\]

- And so we would expect the error to decrease linearly with the number of time steps (as with the binomial model) and quadratically with the number of steps in S.

7.11 Nonlinearity error

- One should be careful when assuming that this is how the scheme converges, if we write out the Taylor expansion in full we see that this convergence depends upon all of the derivatives being well behaved (e.g. not infinite).
- However, we know that in the case of European options, the payoff at expiry is discontinuous leading to an infinite first derivative - and so it seems likely that our approximation may not work as well here.
- Additionally, in the case of an American option, across the early exercise boundary, the second derivative is infinite, this again will lead to difficulty in the approximations - in particular, the assumption as to the error from finite-difference methods.
- There are ways of overcoming this as we shall see.

7.12 Discrete dividends

- It is straightforward to factor in the payment of discrete dividends into the explicit finite difference method.
- Assume that the dividend is paid at \(t_d\) and that there is a node on \(t_d\) in our finite difference grid. The dividend for the underlying asset is a function of the price at \(t_d\), \(D(S)\).
- Denote the time just before the dividend as \(t_d^-\) and the time just after as \(t_d^+\)
- Then we use the fact that \(V(S, t_d^-) = V(S - D(S), t_d^+)\) So calculate \(V(S, t_d^-)\) as usual and then calculate \(V(S + D(S), t_d^-)\) by using interpolation.
- If the grid values of \(S\) above and below \(S + D(S)\) are \(V(S_{up})\) and \(V(S_{down})\) then

\[
V(S, t_d^-) = (1 - h)V(S_{down}) + hV(S_{up})
\]

where \(h = (S + D(S)) - S_{down}\)
7.13 Intro to a better method

- The problems with the explicit finite difference method are twofold
- It has stability issues
- The convergence is only linear in $\Delta t$
- There is a related finite difference method that overcomes both of these problems.

Consider approximating the derivatives of a function at $t + \Delta t/2$. Then we can write the following Taylor series expansions:

$$V(S, t + \Delta t) = V(S, t + \Delta t/2) + \frac{1}{2}\Delta t \frac{\partial V}{\partial t} + \frac{1}{4}(\Delta t)^2 \frac{\partial^2 V}{\partial t^2} + O((\Delta t)^3)$$

$$V(S, t) = V(S, t + \Delta t/2) - \frac{1}{2}\Delta t \frac{\partial V}{\partial t} + \frac{1}{4}(\Delta t)^2 \frac{\partial^2 V}{\partial t^2} + O((\Delta t)^3)$$

- Thus we can estimate the first derivative at $S, t + \Delta t/2$ as
  $$\frac{\partial V}{\partial t} = \frac{V(S, t + \Delta t) - V(S, t)}{\Delta t} + O((\Delta t)^2)$$

- The $S$ derivatives must also be evaluated at $S, t + \Delta t/2$ to give
  $$\frac{\partial V}{\partial S} = \frac{1}{2}\left(V(S + \Delta S, t) - V(S - \Delta S, t)\right) + \frac{1}{2}\left(V(S + \Delta S, t + \Delta t) - V(S - \Delta S, t + \Delta t)\right) + O((\Delta S)^2, (\Delta t)^2)$$

  $$\frac{\partial^2 V}{\partial S^2} = \frac{V(S + \Delta S, t) - 2V(S, t) + V(S + \Delta S, t)}{2(\Delta S)^2} + \frac{V(S + \Delta S, t + \Delta t) - 2V(S, t + \Delta t) + V(S + \Delta S, t + \Delta t)}{2(\Delta S)^2} + O((\Delta S)^2, (\Delta t)^2)$$

7.14 Overview

- We have introduced the finite-difference method which is a way of solving partial differential equations by estimating the first and second derivatives and then substituting the estimates in the PDE.
- This scheme was explained for the Black Scholes PDE and in particular we derived the explicit finite difference scheme to solve the European call and put option problems.
- The convergence of the method is similar to the binomial tree and, in fact, the method can be considered a trinomial tree. Unfortunately, however, the method can be unstable which puts constraints on our grid size.
- We finished by introducing a method with improved convergence and stability, we will see more on this later....
8 Extensions to finite-difference methods

8.1 Overview

- We have seen the basic idea behind finite-difference methods and introduced the explicit finite difference method. Unfortunately it is an unstable scheme that can put limitations on the size of the $S$- and $t$-steps.

- Here we will introduce the Crank-Nicolson method which is a stable method and also has improved convergence.

- In addition we will discuss how to price American options using both of these methods as well as looking how to remove nonlinearity error in a variety of cases.

8.2 The Crank-Nicolson Method

- As introduced at the end of the last section the Crank-Nicolson scheme works by evaluating the derivatives at $V(S, t + \Delta t/2)$.

- The main advantages of this is that the error in the time derivative is now $(\Delta t)^2$ rather than $\Delta t$ and that there are no stability constraints

- The only problem with using the Crank-Nicolson method rather than the explicit method is that we will need to use three option values in the future $(t + \Delta t)$ to calculate three option values not $(t)$. This will make the scheme slightly harder.

- The diagram shows the six points of interest.

Crank-Nicolson grid

Focus attention on $i$, $j$-th value $V^i_j$, and a little piece of the grid around that point.
8.3 Approximations of the derivatives

- From our new approximations, in terms of $V^i_j$ we have

$$\frac{\partial V}{\partial t} \approx \frac{V^{i+1}_j - V^i_j}{\Delta t}$$

$$\frac{\partial V}{\partial S} \approx \frac{1}{4\Delta S} (V_{j+1}^i - V^i_{j-1} + V^{i+1}_{j+1} - V^{i+1}_{j-1})$$

$$\frac{\partial^2 V}{\partial S^2} \approx \frac{1}{2\Delta S^2} (V_{j+1}^i - 2V^i_j + V^i_{j-1} + V^{i+1}_{j+1} - 2V^{i+1}_j + V^{i+1}_{j-1})$$

- Here the $V^i_j$ values are all unknown, so we rearrange our equations to have the known values on one side and the unknown values on the other.

$$\frac{1}{4}(\sigma^2 j^2 - rj)V_{j-1}^i + (-\frac{\sigma^2 j^2}{2} - \frac{r}{2} - \frac{1}{\Delta t})V^i_j + \frac{1}{4}(\sigma^2 j^2 + rj)V_{j+1}^i =$$

$$-\frac{1}{4}(\sigma^2 j^2 - rj)V_{j-1}^{i+1} - (-\frac{\sigma^2 j^2}{2} - \frac{r}{2} + \frac{1}{\Delta t})V^{i+1}_j - \frac{1}{4}(\sigma^2 j^2 + rj)V_{j+1}^{i+1}$$

8.4 The problem

- We can rewrite the valuation problem in terms of a matrix as follows:

$$\begin{pmatrix}
  b_0 & c_0 & 0 & 0 & \ldots & 0 & 0
  a_1 & b_1 & c_1 & 0 & \ldots & 0 & 0
  0 & a_2 & b_2 & c_2 & 0 & \ldots & 0
  \ldots & a_3 & b_3 & c_3 & \ldots & \ldots & 0
  \ldots & \ldots & a_j & b_j & c_j & \ldots & 0
  0 & \ldots & \ldots & a_{j_{max}} & b_{j_{max}}
\end{pmatrix}
\begin{pmatrix}
  V^i_0 \\
  V^i_1 \\
  V^i_2 \\
  \vdots \\
  V^i_{j_{max}} -1 \\
  V^i_{j_{max}}
\end{pmatrix}
= \begin{pmatrix}
  d_0^i \\
  d_1^i \\
  d_2^i \\
  \vdots \\
  d_{j_{max} - 1}^i \\
  d_{j_{max}}^i
\end{pmatrix}$$

where:

$$a_j = \frac{1}{4}(\sigma^2 j^2 - rj)$$

$$b_j = -\frac{\sigma^2 j^2}{2} - \frac{r}{2} - \frac{1}{\Delta t}$$

$$c_j = \frac{1}{4}(\sigma^2 j^2 + rj)$$

$$d_j = -\frac{1}{4}(\sigma^2 j^2 - rj)V_{j-1}^{i+1} - (-\frac{\sigma^2 j^2}{2} - \frac{r}{2} + \frac{1}{\Delta t})V^{i+1}_j - \frac{1}{4}(\sigma^2 j^2 + rj)V_{j+1}^{i+1}$$

8.5 Behaviour on the boundaries

- Again you will note that what happens to the matrix on the boundary is a little different from what happens inside.

- For most PDEs we know the boundary conditions for large and small $S$ and so it is trivial to put in the values of $a$, $b$ and $c$ on the boundaries.

- For call options

$$a_{j_{max}} = 0, b_{j_{max}} = 1, d_{j_{max}} = S^u e^{-\delta (T-i\Delta t)} - X e^{-r(T-i\Delta t)}, b_0 = 1, c_0 = 0, d_0 = 0$$
• For put options
\[ b_0 = 1, \ c_0 = 0, \ d_0 = X e^{-r(T-i\Delta t)}, \ a_{j_{\text{max}}} = 0, \ b_{j_{\text{max}}} = 1, \ d_{j_{\text{max}}} = 0 \]

• In general we can always determine the values of \( b_0, c_0, d_0, a_{j_{\text{max}}}, b_{j_{\text{max}}} \) and \( d_{j_{\text{max}}} \) from our boundary conditions.

8.6 The Crank-Nicolson method

• At each point in time we need to solve the matrix equation in order to calculate the \( V^i_j \) values. There are two approaches to doing this, the first is to solve the matrix equation directly (LU decomposition) the second is to solve the matrix equation via an iterative method (SOR).

• Typically the LU approach is the preferred approach as it gives you an exact value for \( V^i_j \). However, as we will see it does not work particularly well for American options. The SOR (Successive Over Relaxation) approach is easier to program and can be easily adapted to value American options or more exotic options in general (this is actually PSOR - Projected Successive Over Relaxation).

• We will detail both approaches.

8.7 LU decomposition

• The matrix equation can be written as \( AV = d \).

• One method would be to invert \( A \) to get \( V = A^{-1}d \) but this is computationally intensive, especially as most of the entries in \( A \) are zero.

• The matrix \( A \) is a special type of matrix in that we only have non-zero entries around the diagonal. This type of matrix is called tridiagonal.

• LU decomposition allows us to decompose a tridiagonal matrix (in fact it works for more than this) into a lower triangular matrix and an upper triangular matrix. This makes solving the equation far simpler.

• To demonstrate this we will look at the individual equations rather than the matrix so that it is easier to program.

• In order to solve this the method described by G.D. Smith (1978) is used. This method gives a systematic way of solving the tri-diagonal equations. Using the notation from before assume that the following stage of eliminations has been reached:

\[
\beta_{j-1} V^j_{j-1} + c_{j-1} V^i_j = D_{j-1}
\]

\[
a_j V^j_{j-1} + b_j V^i_j + c_j V^i_{j+1} = d^i_j
\]

where \( \beta_0 = b_0 \) and \( D_0 = d^i_0 \).

Eliminating \( V^i_{j-1} \) and rearranging gives

\[
(b_j - \frac{a_j c_{j-1}}{\beta_j}) V^i_j + c_j V^i_{j+1} = d^i_j - \frac{a_j D_{j-1}}{\beta_j}
\]
which can be written in the form
\[ \beta_j V^i_j + c_j V^i_{j+1} = D_j \]
where \( \beta_j = b_j - \frac{a_j c_{j-1}}{\beta_{j-1}} \) and \( D_j = d^i_j - \frac{a_j D_{j-1}}{\beta_{j-1}} \) (1)

The last set of simultaneous equations are given by
\[ \beta_{j_{\text{max}}-1} V^i_{j_{\text{max}}-1} + c_{j_{\text{max}}-1} V^i_{j_{\text{max}}} = D_{j_{\text{max}}-1} \]
\[ a_{j_{\text{max}}} V^i_{j_{\text{max}}-1} + b_{j_{\text{max}}} V^i_{j_{\text{max}}} = d^i_{j_{\text{max}}} \]
and by the same process as before elimination of the \( V^i_{j_{\text{max}}-1} \) term yields
\[ (b_{j_{\text{max}}} - \frac{a_{j_{\text{max}}} c_{j_{\text{max}}-1}}{\beta_{j_{\text{max}}-1}}) V^i_{j_{\text{max}}} = d^i_{j_{\text{max}}} - \frac{a_{j_{\text{max}}} c_{j_{\text{max}}-1}}{\beta_{j_{\text{max}}-1}} \]
i.e.
\[ \alpha_{j_{\text{max}}} V^i_{j_{\text{max}}} = D_{j_{\text{max}}} \] (2)

- The equations are now in the form where one can start obtaining results for the unknown \( V \) values. Using equations 1 and 2, the values for \( V \) are as follows:
\[ V^i_{j_{\text{max}}} = \frac{D_{j_{\text{max}}}}{\beta_{j_{\text{max}}}} \]
\[ V^i_j = \frac{1}{\alpha_j} (D_j - c_j V^i_{j+1}) \] (3)
where the \( \beta \)s and \( D \)s are given by
\[ \alpha_0 = b_0; \quad \beta_j = b_j - \frac{a_j c_{j-1}}{\beta_{j-1}} \] (4)
\[ D_0 = d^i_0; \quad D_j = d^i_j - \frac{a_j}{\beta_{j-1}} D_{j-1}, \] (5)
where \( j = 1, 2, 3, \ldots, j_{\text{max}} \).

8.8 Comments
- So this method allows us to move from a known value of \( V^i_{j_{\text{max}}} \) up to \( V^i_0 \) using the formulae here. Thus we have an explicit algorithm for calculating \( V^i_j \) for all values of \( i \) and \( j \) that does not involve the inversion of a large, sparse matrix.

- The method will be to start from expiry, then move back a time period, calculate the values of \( a_j, b_j, c_j \) and \( d^i_j \) (note that \( d^i_j \) uses \( V^i_{j_{\text{max}}} \) values) and use these together with the equations in the previous subsection to calculate all of the values of \( V^i_{j_{\text{max}}-1} \) and then step back one more time step and repeat the process until we reach \( i = 0 \) when we have the current option value.

- To calculate the \( V^i_j \) values, work up from \( j = 0 \) to obtain \( \beta_j \) and \( D_j \) using equations (4) and (5) and then work down from \( j_{\text{max}} \) to determine \( V^i_j \) using equation (3).

- This is not too difficult to program.
8.9 SOR method

- The SOR method is a simpler approach but can be slightly less accurate and take a little longer as it relies upon iteration.

- If we consider each of the individual equations from $AV = d$ we have that

$$a_1V^i_0 + b_1V^i_1 + c_1V^i_2 = d^i_1$$
$$a_2V^i_1 + b_2V^i_2 + c_2V^i_3 = d^i_1$$

$$\ldots \quad = \ldots$$

$$a_jV^i_{j-1} + b_jV^i_j + c_jV^i_{j+1} = d^i_j$$

$$\ldots \quad = \ldots$$

$$a_{j_{max}-1}V^i_{j_{max}-2} + b_{j_{max}-1}V^i_{j_{max}-1} + c_{j_{max}-1}V^i_{j_{max}} = d^i_{j_{max}-1}$$

- The Jacobi method says that we can trivially rearrange these equations to get:

$$V^i_j = \frac{1}{b_j}(d^i_j - a_jV^i_{j-1} - c_jV^i_{j+1})$$

- The Jacobi method is an iterative one that relies upon the previous equation. Taking an initial guess for $V^i_j$, denoted as $V^{i,0}_j$ (typically $V^{i+1}_j$) then we iterate using the formula below for the $(k+1)$th iteration:

$$V^{i,k+1}_j = \frac{1}{b_j}(d^i_j - a_jV^{i,k}_j - c_jV^{i,k+1}_j)$$

this is performed until the difference between $V^{i,k}_j$ and $V^{i,k+1}_j$ is sufficiently small for all $j$.

- A more effective process is the Gauss-Seidel method that uses the fact that when we calculate $V^{i,k+1}_j$ we already know $V^{i,k}_j$ and so we use this information to write a new iterative formula

$$V^{i,k+1}_j = \frac{1}{b_j}(d^i_j - a_jV^{i,k+1}_j - c_jV^{i,k}_j)$$

- The SOR method is another slight adjustment. It starts from the trivial observation that

$$V^{i,k+1}_j = V^{i,k}_j + (V^{i,k+1}_j - V^{i,k}_j)$$

and so $(V^{i,k+1}_j - V^{i,k}_j)$ is a correction term. It may well be the case that the iterations converge faster to the correct value if we over correct. This is true if $V^{i,k}_j \rightarrow V^i_j$ monotonically in $k$. So the SOR algorithm says that

$$y^{i,k+1}_j = \frac{1}{b_j}(d^i_j - a_jV^{i,k+1}_j - c_jV^{i,k}_j)$$

$$V^{i,k+1}_j = V^{i,k}_j + \omega(y^{i,k+1}_j - V^{i,k}_j)$$

where $1 < \omega < 2$ is called the over-relaxation parameter.

- This can be shown to converge for our values of $a$ and $c$. 

50
8.10 American options: Explicit

- As we well know the American option pricing problem requires us to calculate the optimal early exercise strategy. To do this you typically compare the continuation value with the early exercise value, if the latter is larger then you exercise.

- To value an American option using the explicit finite difference method is pretty straightforward, you calculate the continuation value \( CoV^j_i \) by the valuation formula below

\[
CoV^j_i = \frac{1}{1 + r \Delta t} (AV_{j+1}^i + BV_{j+1}^i + CV_{j-1}^i)
\]

and then compare this to the early exercise payoff. Thus for a put:

\[
V^j_i = \max[X - j \Delta S, \frac{1}{1 + r \Delta t} (AV_{j+1}^i + BV_{j+1}^i + CV_{j-1}^i)]
\]

- This is similar to how we value American options using the binomial tree

\[\text{American put option: Explicit}\]

\[\text{Impose upper boundary at } S^U\]

\[\text{Move through "interior" of mesh/grid using this rule}\]

\[V^j_i = \max[j \Delta S - X, \frac{1}{1 + r \Delta t} (AV_{j+1}^i + BV_{j+1}^i + CV_{j-1}^i)]\]

\[\text{Impose lower boundary at 0}\]

\[0 \quad \Delta t \quad 2\Delta t \quad \ldots \quad i\Delta t \quad \ldots \quad T\]

8.11 American put option: C-N

- The American option pricing problem is slightly more complex for the Crank-Nicolson method. To see this consider the process of calculating \( V^j_i \)

- The value of the option \( V^j_i \), for all values of \( j \), depends also upon the value of \( V_{j-1}^i \) and \( V_{j+1}^i \). Thus, when we are determining where to early exercise we also need to simultaneously know these values. However as we change one of these values then the other values will also change.

- This means that it is not possible to simply compare the early exercise value to the continuation value as, if you decide to early exercise this will change all of the values of \( V^j_i \) and then this decision may well have been suboptimal.
8.12 PSOR

- One, simple solution to this problem is to adapt our SOR method to PSOR (Projected SOR) which enables us to deal with early exercise constraints.

- The only difference between SOR and PSOR is that for each calculation in the iteration you also check whether or not it would be optimal to exercise. Thus our previous iteration technique

\[
y_j^{i,k+1} = \frac{1}{b_j}(d_j^i - a_j y_j^{i,k+1} - c_j V_j^{i,k+1})
\]

\[
V_j^{i,k+1} = V_j^{i,k} + \omega(y_j^{i,k+1} - V_j^{i,k})
\]

to (in case of the American put option)

\[
y_j^{i,k+1} = \frac{1}{b_j}(d_j^i - a_j y_j^{i,k+1} - c_j V_j^{i,k+1})
\]

\[
V_j^{i,k+1} = \max(V_j^{i,k} + \omega(y_j^{i,k+1} - V_j^{i,k}), X - j\Delta S)
\]

8.13 Convergence and accuracy

- If the option price and the derivatives are well behaved then we know that the error of the Explicit method should be \(O(\Delta t, (\Delta S)^2)\) and the Crank-Nicolson method should be \(O((\Delta t)^2, (\Delta S)^2)\). These can be considered similar to the distribution error for the binomial tree as this will be the general convergence to the correct option value.

- If convergence is smooth like this then the results are also amenable to extrapolation.

- Unfortunately just as for binomial trees, finite-difference methods will also suffer from non-linearity error if the grid is not correctly aligned with respect to any discontinuities in the option value, or the derivatives of the option value.

8.14 Non-linearity error

- The nice thing about finite difference methods in general are that you have the freedom to construct the grid as desired and so it is quite simple to construct the grid so that you have a grid point upon any discontinuities.

- For example, if we consider an European call or put option then the only source of non-linearity error is at \(S = X\) at expiry. Thus you should always choose \(\Delta S\) so that \(X = j\Delta S\) for some integer value of \(j\).

- So if in this case \(S_0 = 100\) and \(X = 95\), you need a suitably large \(S^U\) and a \(\Delta S\) which is a divisor of 95. Thus if you want 5000 \(S\) steps then a reasonable choice for \(\Delta S\) may be 0.076, which give \(S^U = 4 \times 95 = 380\) and \(j = 1250\) is the position of the exercise price.

8.15 Barrier options

- As we have seen when pricing barrier options with the binomial method, there is a large amount of non-linearity error that comes from not having the nodes in the tree aligned with the position of the barrier.

- Thus with barrier options we have two sources of non-linearity error, the error from the barrier and the error from the discontinuous payoff.
• However, with the finite difference grid it is relatively easy to align the grid so that you have nodes on the barrier and on the exercise price at expiry.

• For a down and out barrier option choose $S^L$ (the lower value of $S$) to be on the barrier and then, as in the previous example, choose $\Delta S$ so that the exercise price is also on a node.

• For example if $B = 90$ and $X = 95$ if you require 5000 steps, choose $\Delta S = 0.05$, so that $S^U = 340$, and the barrier is at $j = 0$ and the exercise price is at $j = 100$.

8.16 Overview

• We have introduced the Crank-Nicolson finite difference method. This is slightly harder to program than the explicit method in that you have to solve a matrix equation at each time point but it has faster convergence and is stable.

• Applying the method to American options requires the use of PSOR which again is more complex than the method for valuing American options using the explicit method.

• Finally, we saw that with finite-difference methods as you can choose the dimensions of the grid so as to remove the nonlinearity error.
9 Finite differences: further considerations

- Here we will complete our look at finite-difference methods by looking at body-fitted coordinates, these allow us to fit the grid to avoid any non-linearity error even when there is early exercise.

9.1 Body-fitted coordinates

- As discussed, the main problem when applying numerical schemes to problems involving early exercise is that the position of the early exercise boundary is not known a priori. This often means that when constructing a grid this boundary does not often fall on a grid point, or node, causing a degree of non-linearity error.

- The usual numerical schemes: lattice methods and basic finite difference schemes are unable to fix the grid correctly.

- The method described here involves keeping the time axis fixed but transforming the $S$ to an $\hat{S}$ axis that moves with the free boundary.

9.2 The American put option

- The American put option price, $P(S,t)$ is described by the Black-Scholes equation for the following boundary conditions:

$$P(S,T) = \max(X - S, 0)$$

$$P(S \leq S_f(t), t) = X - S$$

$$P(S, t) \to 0 \text{ as } S \to \infty$$

- where, as usual, $S_f(t)$ is the position of the free boundary at time $t$. To incorporate the body-fitted co-ordinate system simply introduce the transformation

$$\hat{S} = S - S_f(t)$$

- and so the range of $\hat{S}$ is from 0 to $\infty$. This will also ensure that, at every timestep, the $S$-dimension of the grid will range from the exact position of the free-boundary to a sufficiently large value of $S$.

- This removes the nonlinearity error present in the previous schemes. This desired effect does come at a cost though. As the position of the early exercise boundary is not known a priori, then the position of all the $\hat{S}$ values are also not known at each point in time.

- It is, thus, necessary to perform a technique analogous to a multivalued Newton-Raphson iteration. This is undertaken in combination with the usual Crank Nicolson scheme. The first point to note is that the above transformation leads to a slight change in the governing equation, since

$$\frac{\partial P}{\partial t} \to \frac{\partial P}{\partial \hat{t}} + \frac{\partial P}{\partial \hat{S}} \frac{\partial \hat{S}}{\partial t} = \frac{\partial P}{\partial \hat{t}} - \frac{\partial P}{\partial \hat{S}} \frac{dS_f}{dt}$$

and so the Black-Scholes equation becomes:

$$\frac{1}{2} \sigma^2 (\hat{S} + S_f)^2 \frac{\partial^2 P}{\partial \hat{S}^2} + r(\hat{S} + S_f) \frac{\partial P}{\partial \hat{S}} + \frac{\partial P}{\partial t} - \frac{\partial P}{\partial \hat{S}} \frac{dS_f}{dt} - rP = 0$$
Fortunately the option value, \( P \) and the position of the free boundary \( S_f(t = T) \) are both known at expiry (i.e. \( S_f = X \) and \( P = \max(-\dot{S}, 0) \)) and so to calculate the values at \( T - \Delta t \) we simply perform an iterative scheme using these values as a starting point.

\[
P^{(m+1)}(i\Delta \dot{S}, j\Delta \tau) = P^{(m)}(i\Delta \dot{S}, j\Delta \tau) + \delta P_i \quad \text{for} \quad 0 \leq i \leq n
\]

At a given timestep \( j\Delta \tau \), after \( m \) iterations write the values at the \( n \) distinct \( P \) values in the finite difference scheme and the value of \( S_f \) as

\[
S^{(m+1)}_f(j\Delta \tau) = S^{(m)}_f(j\Delta \tau) + \delta S_f
\]

The value of \( P^{(0)}(i\Delta \dot{S}, j\Delta \tau) \) and \( S^{(0)}_f(j\Delta \tau) \) are taken to be the converged values of \( P_i(i\Delta \dot{S}, (j+1)\Delta \tau) \) and \( S_f((j+1)\Delta \tau) \) respectively. All that remains is to calculate the successive values of the \( \delta P_i \)'s and \( \delta S_f \). In order to do this a Crank-Nicolson scheme is used.

### 9.3 Crank-Nicolson

The principal differences from the basic scheme are, first, that the unknown values are the \( \delta P_i \)'s and \( \delta S_f \). Secondly, there is also be an extra column in the matrix to cope with the new extra unknown, \( \delta S_f \). There are, now, \( N+1 \) equations in \( N+2 \) unknowns, this is easily overcome by using both the boundary conditions at \( \dot{S} = 0 \) (\( S = S_f \)) including the condition from before and the smooth pasting condition

\[
\frac{\partial P}{\partial S} = -1
\]

The next step requires arranging the difference equations into a system with the unknown values on one side and the known ones on the other. The approximation for the derivatives is as usual. In general \( P(i\Delta \dot{S}, j\Delta t) \) is denoted \( P_{i,j} \) and the resulting set of equations, for \( 0 \leq i \leq N \), is

\[
a_i \delta P_{i-1} + b_i \delta P_i + c_i \delta P_{i+1} + d_i \delta S_f = e_i
\]

where, as before, the values of \( a_i, b_i, c_i, d_i \) and \( e_i \) need to be calculated.

### 9.4 Boundary conditions

There are two boundary conditions at \( \dot{S} = 0 \) and one at \( \dot{S} = \dot{S}_{\text{max}} \). For these the values of \( a, b, c, d \) and \( e \) are known and are as follows:

- At \( \dot{S} = 0 \):

\[
P_0^m + \delta P_0 = X - (S_f^{(m)} + \delta S_f)
\]

and using a one-sided difference scheme we can generate an equation for the derivative condition

\[
-3P_{0,j}^m + 4P_{1,j}^m - P_{2,j}^m - 3\delta P_0 + 4\delta P_1 - \delta P_2 = -\frac{2\Delta \dot{S}}{2\Delta \dot{S}}
\]
9.5 Method

- There are now a series of $N + 2$ equations in $N + 2$ unknowns which can be solved using linear algebra techniques. In matrix form the problem can be displayed as

\[
\begin{pmatrix}
1 & 0 & 0 & \ldots & 0 & 1 \\
-3 & 4 & -1 & 0 & \ldots & 0 \\
a_1 & b_1 & c_1 & 0 & \ldots & d_1 \\
0 & a_2 & b_2 & c_2 & \ldots & d_2 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & \ldots & a_i & b_i & c_i & d_i \\
0 & \ldots & 0 & b_N & 0
\end{pmatrix}
\begin{pmatrix}
\delta P_0 \\
\delta P_1 \\
\delta P_2 \\
\delta P_{N-1} \\
\delta P_N \\
\delta S_f \\
\delta S_f
\end{pmatrix}
= \begin{pmatrix}
X - P^{(m)}_{0,j} - S_{f,j}^{(m)} \\
0 \\
e_1 \\
\vdots \\
\vdots \\
- P^{(m)}_{N,j}
\end{pmatrix}
\]

A reduction technique is then applied to solve the system. First, remove all the $c_i$ terms, starting from $c_{N-1}$ by using

\[c'_i = c_i - b_{i+1} \frac{c_i}{b_{i+1}}\]

- and as a result the other terms are affected, namely

\[b'_i = b_i - a_{i+1} \frac{c_i}{b_{i+1}}\]
\[d'_i = d_i - d_{i+1} \frac{c_i}{b_{i+1}}\]
\[e'_i = e_i - e_{i+1} \frac{c_i}{b_{i+1}}\]

- There is a slight anomaly in the reduction technique in that the second row in the matrix has an extra column. This is easily overcome as this last term is removed using $b_2$ and the other terms adjusted accordingly and then the $c_0$ term is removed by using $c_1$ as before.

- The matrix is now in the form:

\[
\begin{pmatrix}
0 & 0 & 0 & \ldots & d'_{N-1} \\
b'_0 & 0 & 0 & \ldots & d'_0 \\
a_1 & b_1 & 0 & \ldots & d'_1 \\
0 & a_2' & b_2' & 0 & \ldots & d'_2 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & \ldots & a'_i & b'_i & 0 & d'_i \\
0 & \ldots & 0 & b_N & 0
\end{pmatrix}
\begin{pmatrix}
\delta P_0 \\
\delta P_1 \\
\delta P_2 \\
\delta P_{N-1} \\
\delta P_N \\
\delta S_f \\
\delta S_f
\end{pmatrix}
= \begin{pmatrix}
X - P^{(m)}_{0,j} - S_{f,j}^{(m)} \\
0 \\
e_1 \\
\vdots \\
\vdots \\
- P^{(m)}_{N,j}
\end{pmatrix}
\]
• but from the top row it is possible to calculate the value of $\delta S_f$ as

$$\delta S_f = \frac{e'_{-1}}{d'_{-1}}$$

and from the second row the value of $\delta P_0$ can be deduced

$$\delta P_0 = \frac{e'_0 - d'_0 \delta S_f}{b'_0}$$

• then continue this regime to calculate the values of $\delta P_i$ for $1 \leq i \leq N$ using

$$\delta P_i = \frac{e'_i - d'_i \delta S_f - a'_i \delta P_{i-1}}{b'_i}$$

• This scheme is equivalent to a Newton-Raphson iteration and, as such, must have converged before moving forward to the next timestep.

• Depending on the accuracy required, we select a value for which all the $\delta P_i$'s or $\delta S_f$ need to be less than to have assumed convergence - a typical value is $1 \times 10^{-8}$

### 9.6 Time adjustment

• In order to try and predict the movement of the early exercise boundary, a somewhat unrigorous scaling of the timesteps is carried out. It is known that the early exercise boundary does not move exactly as $\sqrt{T}$ (This is because there are also log-terms present in the exact representation of the early exercise boundary as $T \to 0$.) However, it is a good enough approximation to ensure that there are no stepsize constraints in the body-fitted Crank Nicolson scheme.

• This scaling is very easy to implement. Simply divide time into N steps of size $\Delta t = T/N$ as usual, however when using $\Delta t$ in the numerical calculation instead use $\Delta t = T/N^2$

• Often, when using body-fitted co-ordinates, it can be difficult to correctly determine the position of the boundary (in this case, near to $\tau = 0$) but this ensures that the early exercise boundary follows, approximately, its proper form.

• Now that any nonlinearity error has been removed it would be expected that the convergence to the correct option price is at the rate $O((\Delta t)^2, (\Delta S)^2)$. If so, it should be possible to extrapolate results, assuming that there is also no non-linearity error occurring at the strike price at maturity.

• At expiry the early exercise boundary is at the strike price and, thus, extrapolation can be performed. As the results converge at the same rate in both the $t$ and the $S$ directions it is possible to perform just one extrapolation.
10 Monte Carlo methods for American options the Longstaff and Schwartz method

10.1 Overview

- We have looked at using Monte Carlo methods for most types of options, but they struggle with early exercise.
- This is the subject of lots of research at the moment and we have seen the basic idea of some of the methods.
- What we present here is one of the easiest methods to understand and implement.
- There are still issues with how it performs in practise which we will also deal with here.
- The key calculations when valuing an American option are determining the early exercise value and the expected continuation value.
- The option value at any point in time will be the larger of these two values.
- The early exercise value is simple to calculate at any point in time. The continuation value is typically determined by a backward iteration approach, where the value now is the discounted expected option value at the next instance in time.
- The key information we need to know is given the share price at time $t$, what is the expected option value at time $t + \Delta t$.
- This is simple to determine in binomial models but less so in Monte Carlo approaches.

10.2 The least squares method

- This is a technique for fitting a set of functions to (given) data.
- Here we describe the procedure for an $m$th degree polynomial - it is straightforward to extend the idea to a general class of polynomial.
- When using an $m$th degree polynomial
  
  \[ y = a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^m \]

  to approximate the given set of data, $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where $n \geq 3$, the best fitting curve $f(x)$ has the least square error, i.e.,

  \[ \Pi = \sum_{i=1}^{n} [y_i - f(x_i)]^2 = \sum_{i=1}^{n} [y_i - (a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_m x_i^m)]^2 \]

  is minimized.

- Note that $a_0, a_1, a_2, \ldots,$ and $a_m$ are unknown coefficients while all $x_i$ and $y_i$ are given. To obtain the least square error, the unknown coefficients $a_0, a_1, a_2, \ldots,$ and $a_m$ must yield
zero first derivatives.

\[ \frac{\partial \Pi}{\partial a_0} = 2 \sum_{i=1}^{n} [y_i - (a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_m x_i^m)] = 0 \]

\[ \frac{\partial \Pi}{\partial a_1} = 2 \sum_{i=1}^{n} x_i [y_i - (a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_m x_i^m)] = 0 \]

\[ \frac{\partial \Pi}{\partial a_2} = 2 \sum_{i=1}^{n} x_i^2 [y_i - (a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_m x_i^m)] = 0 \]

\[ \frac{\partial \Pi}{\partial a_m} = 2 \sum_{i=1}^{n} x_i^m [y_i - (a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_m x_i^m)] = 0 \]

- Expanding the above equations, we have

\[ \sum_{i=1}^{n} y_i = a_0 \sum_{i=1}^{n} 1 + a_1 \sum_{i=1}^{n} x_i + a_2 \sum_{i=1}^{n} x_i^2 + \ldots + a_m \sum_{i=1}^{n} x_i^m \]

\[ \sum_{i=1}^{n} x_i y_i = a_0 \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^2 + a_2 \sum_{i=1}^{n} x_i^3 + \ldots + a_m \sum_{i=1}^{n} x_i^{m+1} \]

\[ \sum_{i=1}^{n} x_i^2 y_i = a_0 \sum_{i=1}^{n} x_i^2 + a_1 \sum_{i=1}^{n} x_i^3 + a_2 \sum_{i=1}^{n} x_i^4 + \ldots + a_m \sum_{i=1}^{n} x_i^{m+2} \]

\[ \sum_{i=1}^{n} x_i^m y_i = a_0 \sum_{i=1}^{n} x_i^m + a_1 \sum_{i=1}^{n} x_i^{m+1} + a_2 \sum_{i=1}^{n} x_i^{m+2} + \ldots + a_m \sum_{i=1}^{n} x_i^{2m} \]

- The unknown coefficients \( a_0, a_1, a_2, \ldots, \) and \( a_m \) can hence be obtained by solving the above linear equations.

- There are also many library routines available to do the job.

10.3 Longstaff and Schwartz (2001)

- The Longstaff and Schwartz method, essentially estimates the conditional expected option value at the next time step by simulating lots of paths and then carrying out a regression of the future realised option value as a function of the current value of the underlying asset.

- This gives an approximation for the continuation value that can then be compared to the early exercise value and then we know the option value at each point in time on each path.

- In terms of Monte Carlo pricing, all we actually need to know is the rule for early exercising, so we know when we receive the cash flows and the value of the option is the average of the discounted payoffs for each path.

- We will explain the method via an example and then describe the general method.
10.4 Example

- We will attempt to value a Bermudan put option where exercise is possible now and at three future dates. \( S_0 = 1, \ X = 1.1, \ r = 0.06 \).

- The first step is to simulate some paths, the table below denotes the results:

<table>
<thead>
<tr>
<th>Path</th>
<th>( t = 0 )</th>
<th>( t = 1 )</th>
<th>( t = 2 )</th>
<th>( t = 3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.00</td>
<td>1.09</td>
<td>1.08</td>
<td>1.34</td>
</tr>
<tr>
<td>2</td>
<td>1.00</td>
<td>1.16</td>
<td>1.26</td>
<td>1.54</td>
</tr>
<tr>
<td>3</td>
<td>1.00</td>
<td>1.22</td>
<td>1.07</td>
<td>1.03</td>
</tr>
<tr>
<td>4</td>
<td>1.00</td>
<td>.93</td>
<td>.97</td>
<td>.92</td>
</tr>
<tr>
<td>5</td>
<td>1.00</td>
<td>1.11</td>
<td>1.56</td>
<td>1.52</td>
</tr>
<tr>
<td>6</td>
<td>1.00</td>
<td>.76</td>
<td>.77</td>
<td>.90</td>
</tr>
<tr>
<td>7</td>
<td>1.00</td>
<td>.92</td>
<td>.84</td>
<td>1.01</td>
</tr>
<tr>
<td>8</td>
<td>1.00</td>
<td>.88</td>
<td>1.22</td>
<td>1.34</td>
</tr>
</tbody>
</table>

- We need to use this information to determine the continuation value at each point in time for each path. To do this we will construct a "Cash Flow Matrix" at each point in time.

  Continuation value at \( t = 2 \)

- The table below denotes the cash flows at \( t = 3 \) assuming that we held the option that far:

<table>
<thead>
<tr>
<th>Path</th>
<th>( t = 1 )</th>
<th>( t = 2 )</th>
<th>( t = 3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>–</td>
<td>–</td>
<td>.00</td>
</tr>
<tr>
<td>2</td>
<td>–</td>
<td>–</td>
<td>.00</td>
</tr>
<tr>
<td>3</td>
<td>–</td>
<td>–</td>
<td>.07</td>
</tr>
<tr>
<td>4</td>
<td>–</td>
<td>–</td>
<td>.18</td>
</tr>
<tr>
<td>5</td>
<td>–</td>
<td>–</td>
<td>.00</td>
</tr>
<tr>
<td>6</td>
<td>–</td>
<td>–</td>
<td>.20</td>
</tr>
<tr>
<td>7</td>
<td>–</td>
<td>–</td>
<td>.09</td>
</tr>
<tr>
<td>8</td>
<td>–</td>
<td>–</td>
<td>.00</td>
</tr>
</tbody>
</table>

- The next step is to attempt to find a function that describes the continuation value at time 2 as a function of the value of \( S \) at time 2.

- To do this we use a regression technique, that takes the values at time 2 as the "\( x \)" values and the discounted payoff at time 3 as the "\( y \)" values.

  Continuation value at \( t = 2 \)
• Note that the regression is only carried out on paths that are in the money at time 2.

• The regression here is simple where \( y \) is regressed upon \( x \) and \( x^2 \) (the actual scheme is slightly more sophisticated). In this particular example (using least squares): \( y = -1.070 + 2.938x - 1.813x^2 \), and so we can use this to estimate the continuation value for each of the current share prices (\( x \) in the regression).

• For example for path 1, where \( S = 1.08 \), the regression formula gives \( y \) (the continuation value) to be 0.0369.

<table>
<thead>
<tr>
<th>Path</th>
<th>( y )</th>
<th>( x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.00 × .94176</td>
<td>1.08</td>
</tr>
<tr>
<td>2</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>.07 × .94176</td>
<td>1.07</td>
</tr>
<tr>
<td>4</td>
<td>.18 × .94176</td>
<td>.97</td>
</tr>
<tr>
<td>5</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>.20 × .94176</td>
<td>.77</td>
</tr>
<tr>
<td>7</td>
<td>.09 × .94176</td>
<td>.84</td>
</tr>
<tr>
<td>8</td>
<td>–</td>
<td></td>
</tr>
</tbody>
</table>

Option value at \( t = 2 \)

Optimal early exercise decision at time 2

<table>
<thead>
<tr>
<th>Path</th>
<th>Exercise</th>
<th>Continuation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.02</td>
<td>.0369</td>
</tr>
<tr>
<td>2</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>3</td>
<td>.03</td>
<td>.0461</td>
</tr>
<tr>
<td>4</td>
<td>.13</td>
<td>.1176</td>
</tr>
<tr>
<td>5</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>6</td>
<td>.33</td>
<td>.1520</td>
</tr>
<tr>
<td>7</td>
<td>.26</td>
<td>.1565</td>
</tr>
<tr>
<td>8</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

• This then allows you to decide at which points in time you would exercise and thus determine the cash flows at \( t = 2 \) (below). Notice that for each path, if you exercise at \( t = 2 \) then you do not also exercise at \( t = 3 \)
Cash-flow matrix at time 2

<table>
<thead>
<tr>
<th>Path</th>
<th>t = 1</th>
<th>t = 2</th>
<th>t = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>–</td>
<td>.00</td>
<td>.00</td>
</tr>
<tr>
<td>2</td>
<td>–</td>
<td>.00</td>
<td>.00</td>
</tr>
<tr>
<td>3</td>
<td>–</td>
<td>.00</td>
<td>.07</td>
</tr>
<tr>
<td>4</td>
<td>–</td>
<td>.13</td>
<td>.00</td>
</tr>
<tr>
<td>5</td>
<td>–</td>
<td>.00</td>
<td>.00</td>
</tr>
<tr>
<td>6</td>
<td>–</td>
<td>.33</td>
<td>.00</td>
</tr>
<tr>
<td>7</td>
<td>–</td>
<td>.26</td>
<td>.00</td>
</tr>
<tr>
<td>8</td>
<td>–</td>
<td>.00</td>
<td>.00</td>
</tr>
</tbody>
</table>

Continuation value at t = 1

• We can apply the same process to t = 1, for each of the paths that are in the money we regress the discounted future cash flows (y) on the current value of the underlying asset (x), where x and y are as given below:

<table>
<thead>
<tr>
<th>Regression at time 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

• The regression equation here is $y = 2.038 - 3.335x + 1.356x^2$ and again we use this to estimate the continuation value and decide on an early exercise strategy.

• The next table compares the two values and the final table denotes the early exercise or stopping rule:

<table>
<thead>
<tr>
<th>Optimal early exercise decision at time 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

• The early exercise strategy is as follows:

Stopping rule
<table>
<thead>
<tr>
<th>Path</th>
<th>$t = 1$</th>
<th>$t = 2$</th>
<th>$t = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- From this we can then value the option, by forming the final cash flow matrix from this rule.

<table>
<thead>
<tr>
<th>Option cash flow matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

- So the option value is the average of the discounted cash flows, so in this case:

$$V_0 = \frac{1}{8}(0 + 0 + 0.07e^{-3r} + 0.127e^{-r} + 0.34e^{-r} + 0.18e^{-r} + 0.22e^{-r}) = 0.1144$$

10.5 More sophisticated regression

- In general the regression here $y = a_1 + a_2x + a_3x^2$ is not going to be satisfactory, especially as we will have far more than 8 paths when attempting to find the functional form of the continuation values.

- In fact the general form of $y$ is:

$$y = \sum_{j=0}^{M} a_j F_j(x)$$

- The user decides upon $M$ the number and the functional form $F_j(x)$ (in our example $F_j(x) = x^j$) where $F_j(x)$ is called a basis function.

- Although Longstaff and Schwartz suggest Laguerre polynomials provide the best fit, my favourite is Chebyshev polynomials. However, you are free to choose whichever basis functions you want (e.g. Polynomial as in the example, Chebyshev, Hermite, Laguerre, etc.)

- The Laguerre polynomials are given by:

$$F_0(x) = e^{x/2}$$
$$F_1(x) = (1 - x)e^{x/2}$$
$$F_n(x) = e^{x/2} \frac{d^n}{dx^n}(x^n e^{-x})$$
Chebyshev polynomials are given by:

\[ F_n(x) = \cos(n(\cos^{-1} x)) \]

In particular \( T_0(x) = 1, T_1(x) = 1, T_2(x) = 2x^2 - 1, \ldots \)

Then the least squares approach approximates the constants \( a_i \), and when we have these values we can use them to predict the continuation value for each value of \( S \) at every point in time.

10.6 Longstaff and Schwartz - general procedure

- Decide on the number of sample paths \( N \), the number of basis functions for the regression, \( M \), and the type of basis functions \( F_j(x) \) and the number of observation dates \( d \).

- Draw \( Nd \) Normally distributed random numbers and simulate the sample paths for the underlying asset at each point in time \( S_{t_i}^n \) \( 1 \leq i \leq d, 1 \leq n \leq N \)

- At expiry \( t = t_d \), record the cash flow values \( CF^n(t_d) \) which for a put are \( \max(X - S_{t_d}^n, 0) \)

- Move back to \( t = t_{d-1} \) for each path where \( S_{t_{d-1}}^n < X \) calculate the continuation value as \( CV^n(t_{d-1}) = e^{-r(t_d - t_{d-1})}CF^n(t_d) \). From these values perform the regression to determine the functional form of the continuation value, \( y(S) \) where \( S \) is the value of the underlying asset

- Recalculate the continuation value as \( CV^n(t_{d-1}) = y(S_{t_{d-1}}^n) \)

- For every path calculate the cash flow value where if the continuation value \( CV^n(t_{d-1}) < X - S_{t_{d-1}}^n \) then \( CF^n(t_{d-1}) = X - S_{t_{d-1}}^n \) and \( CF^n(t_i) = 0 \) for \( i > d - 1 \) otherwise \( CF^n(t_{d-1}) = 0 \)

- Repeat this process for the previous time step until you have \( CF^n(t_i) \) for all \( i \) and \( n \). Note that in general to calculate \( CV^n(t_i) \) before the regression

\[ CV^n(t_i) = \sum_{n=i+1}^{d} e^{-r(t_n - t_i)}CF^n(t_n) \]

- The option value \( V_0 \) is then

\[ V_0 = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} e^{-r_t i}CF^i(t_j) \]

- When you have more than one underlying asset in order to perform the regression you need to have basis functions in all of the underlying assets as well as in the cross terms between them (i.e. in \( S_1, S_2 \) and \( S_1S_2 \)).

- This means that the number of basis functions will increase exponentially as you increase the number of underlying assets, although it is not necessary in practise to have too large a number of basis functions.
10.7 How well does it perform?

- Longstaff and Schwartz provide proofs that as \( M \to \infty \) and \( N \to \infty \) the option value obtained from their scheme converges to the theoretical value.

- This isn’t much use for practical considerations as you will be limited by how many basis functions you can calculate and how many simulations you can perform.

- There have been a few investigations into the method and the view of the method’s performance are mixed: Broadie and Detemple (2004) say regression methods ‘often incur unknown approximation errors and are limited by a lack of error bounds’.

- A detailed appraisal of this technique by Moreno and Navas (2003) investigates the use of various polynomial fits and numbers of basis functions.

- It is not clear that increasing the number of basis functions actually increases the accuracy of the method and there is no real difference from using different types of basis functions (e.g. Chebyshev rather than Laguerre)

- For more complicated derivative pricing problems the trend is even less clear, sometimes errors can increase as you add more basis functions (too many essentially fits the stochastic values of \( S \) exactly)

- In general, the method will provide good estimates but will be difficult to assess exactly how accurate it is.

- See Duck et al (2005) for some improvements on the basic Longstaff and Schwartz scheme (the latter provides superoptimal estimates, but these can be exploited using extrapolation).

10.8 Overview

- We have introduced a method for valuing options with early exercise features using simulation.

- The main idea is to estimate the continuation value (as a function of the current underlying asset price) by performing a least squares regression.

- The method converges to the correct option price but research shows that it is unclear how well the method performs and how to estimate the error when using a finite number of paths/ basis functions.
11 The QUAD method

In this chapter we consider a simple, widely applicable numerical approach for option pricing based on quadrature methods. Though in some ways similar to lattice or finite-difference schemes, it possesses exceptional accuracy and speed. Discretely monitored options are valued with only one time step between observations, and nodes can be perfectly placed in relation to discontinuities. Convergence is improved greatly; in the extrapolated scheme, a doubling of points can reduce error by a factor of two-hundred and fifty-six. Complex problems (e.g. fixed-strike lookback discrete barrier options) can be evaluated accurately and orders of magnitude faster than by existing methods.

11.1 Introduction

With QUAD, a full range of final positions can be evaluated using just one time-step and the improvement in efficiency is quite remarkable. For example, using a lattice method a doubling of accuracy requires approximately four times the computational effort, whereas using QUAD with Simpson’s rule implementation (see later), a four-fold increase in computational time improves accuracy by a factor of sixteen rising to a factor of two-hundred and fifty-six with extrapolation.

With QUAD, nodes can be positioned exactly as required, thus avoiding all non-linearity error, even when pricing such exotic products as moving discrete barrier options and lookback options with barriers. A welcome consequence of this eradication of non-linearity error is that extrapolation can be used to reduce distribution error even further.

QUAD can be seen as analogous to a multinomial lattice in that the option prices are calculated from values at nodes later in time, working backwards from maturity. However, it also has the added flexibility that nodes can be placed wherever desired and only one time step is required between observations of exotic features. In many ways, it could be considered ‘the perfect tree’ method.

Figure 1 shows schematically how QUAD evaluates a down-and-out moving discrete barrier call option; a much more detailed summary of the method follows later. This option has a pay-off at maturity identical to a vanilla-call option, but this pay-off is contingent on the underlying asset remaining above the level $B_m$ at times $T_m$. If at these observation times the underlying is below the barrier level then the option is ‘knocked out’ and becomes worthless.

Between barriers, the option value behaves precisely as a vanilla European option, and as a consequence the well-known exact solution for such options can be employed to relate values, backwards in time, between successive barriers. It is only necessary to calculate option values when the special feature, the knock out, applies. In contrast with lattice and finite-difference methods there is no need to consider intermediate timesteps and it is possible to move to any value of the underlying at each time-step. The nodes at each significant time-step may be simply positioned so as to cope with non-linearity, in this case the barriers and the strike price at maturity.

It is well known that the majority of options can be written as either a finite or an infinite set of nested (multiple) integrals.

11.2 Building blocks of the QUAD method

11.2.1 Adapting the Black-Scholes equation

As a starting point, consider the well-known Black and Scholes (1973) partial differential equation for an option with an underlying asset following geometric Brownian motion:

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

(1)
Figure 1: A schematic diagram demonstrating the evaluation of a discrete moving barrier option with barriers at $B_1$, $B_2$ and $B_3$, strike price $X$, time to maturity $T_4$ and current underlying price of $S$. QUAD integrates across a set of points at each observation time (reduced in number and shown in bold for only one point in each set before maturity).
where $V(S, t)$ is the price of the derivative product, $S$ the current value of the underlying asset, $t$ is time, $T$ is the time to maturity $r$ the risk-free interest rate, $\sigma$ the volatility of the underlying asset and $X$ is the exercise price of the option. $D_c$ is a continuous dividend yield which could, for example, be the foreign interest rate in a foreign exchange option.

Next make the following standard transformations in Eq. 1

\[ x = \log(S_t/X), \quad (2) \]
\[ y = \log(S_{t+\Delta t}/X). \quad (3) \]

It is important to note in what follows that $\Delta t$ is not restricted to small time periods. The final conditions for a European option expiring at time $T = t + \Delta t$ with $V(y, t + \Delta t)$ are transformed in straightforward fashion; e.g. the payoff for a call option $\max(S_{t+\Delta t} - X, 0)$ becomes $\max(Xe^y - X, 0)$. The value of the option at time $t$ on an underlying asset $S_t$ has an exact solution given by

\[ V(x, t) = A(x) \int_{-\infty}^{\infty} B(x, y)V(y, t + \Delta t) dy, \quad (4) \]

where,

\[ A(x) = \frac{1}{\sqrt{2\sigma^2 \pi \Delta t}} e^{-\frac{1}{2}kx - \frac{1}{8} \sigma^2 k^2 \Delta t - r \Delta t}, \quad (5) \]

and

\[ B(x, y) = e^{-\frac{(x-y)^2}{2\sigma^2 \Delta t} + \frac{1}{2}ky} \quad (6) \]

and

\[ k = \frac{2(r - D_c)}{\sigma^2} - 1. \quad (7) \]

Also, let $x_0$ be the $x$ corresponding to the value of $S$ at the outset, $s_t$ i.e.

\[ x_0 = \log(S_t/X). \quad (8) \]

Henceforth the integrand here will be denoted as $f(x, y)$, so that Eq. (4) becomes

\[ V(x, t) = A(x) \int_{-\infty}^{\infty} f(x, y) dy \quad (9) \]

where the integrand is given by

\[ f(x, y) = B(x, y)V(y, t + \Delta t). \quad (10) \]

The solution (4) contains an integral which, in general, cannot be evaluated analytically in terms of simple functions, although for European options it is easily converted to the probability density function for the Normal distribution. For more complicated options, numerical techniques are required to evaluate the integral in Eq. (4). The intuition behind QUAD is that although Eq. (4) is not valid across all points in time for anything other than a plain European option, the valuation problem can be sliced into consecutive time intervals, during which Eq. (4) is locally applicable. Imposition of the appropriate option features at their corresponding ‘observation’ times provides a link between these consecutive intervals, and solution of complex problems becomes possible.
11.3 Quadrature methods

There are many different methods of numerical integral evaluation. Simpson’s rule is used as a starting point because it is robust, simple, nimble, well-known and has fast convergence: of order $(\delta y)^4$, where $\delta y$ is the size of the gap between points (or of order $N^{-4}$ where $N$ is the number of steps). By comparison a trinomial tree converges merely with $N^{-1}$ or in some cases only with $N^{-\frac{1}{2}}$. Even the more sophisticated finite difference methods at best converge at the rate of $(\Delta S^2, \Delta t^2)$ where $\Delta S$ and $\Delta t$ are the step sizes in the $S$ and $t$ directions respectively. Use of the trapezoidal rule will also be discussed here, although virtually any quadrature method is applicable, including higher-order schemes and Gaussian quadrature.

11.3.1 Simpson’s rule

Although Simpson’s rule was devised as long ago as 1743, it remains one of the most accurate and popular methods for approximating integrals. The idea is conceptually simple: for a function of $y$, $f(y)$, plotted against $y$ divide the desired range, $[a_1, a_2]$ into intervals of a fixed length and then approximate the area under the curve by summing the area of the individual regions. This yields the following expression, for a step size of $\delta y$,

$$\int_{a_1}^{a_2} f(y) \, dy \approx \frac{\delta y}{6} \{f(a_1) + 4f(a_1 + \frac{1}{2}\delta y) + 2f(a_1 + \delta y) + 4f(a_1 + \frac{3}{2}\delta y)$$

$$+ 2f(a_1 + 2\delta y) + \cdots + 2f(a_2 - \delta y) + 4f(a_2 - \frac{1}{2}\delta y) + f(a_2)\}. \quad (11)$$

It is straightforward to show that for smooth functions the error term associated with this method decreases at a rate of order $(\delta y)^4$, i.e. a doubling of the number of steps reduces the error by a factor of sixteen.

11.3.2 Trapezoidal rule

The trapezoidal rule is an even simpler quadrature method but is slower to converge, at a rate of $(\delta y)^2$. It is included here because the procedure can save computational time when pricing options for which the overall valuation technique puts other, more stringent, limitations on convergence, thus making the use of more accurate quadrature schemes superfluous.

$$\int_{a_1}^{a_2} f(y) \, dy \approx \frac{\delta y}{2} \{f(a_1) + 2f(a_1 + \delta y) + 2f(a_1 + 2\delta y) \cdots + 2f(a_2 - \delta y) + f(a_2)\}. \quad (12)$$

The trapezoidal rule will find a reasonably accurate value more quickly than will Simpson’s rule and so when speed is required rather than accuracy, the trapezoidal rule is sometimes more appropriate.

11.4 Method and general application

In order to demonstrate how the quadrature component is incorporated into the QUAD method, this section first considers the simple case of a vanilla call option. Here there is only one observation time, at maturity, and calculations are complete in a single time step. The closed form solution for this option is then used to verify the accuracy of the quadrature scheme. More interesting, challenging and complex problems will follow later.
11.4.1 Illustration using vanilla calls

In the case of a vanilla call valued now, time \( t \), the payoff at maturity time, \( t + \Delta t \), is \( \max(S_{t+\Delta t} - X, 0) \). The integrand, Eq. (10) becomes
\[
f(x, y) = B(x, y) \times X \max(e^y - 1, 0).
\]
A numerical scheme is then implemented to evaluate the integral in Eq. (9) using Simpson’s Rule, Eq. (11).

Simpson’s rule converges with \((\delta y)^4\) but only when evaluating functions with continuous derivatives. The first derivative of the payoff to a call option is clearly discontinuous at the strike price (since for \( S < X \), \( \frac{\partial V}{\partial S} = 0 \), for \( S > X \), \( \frac{\partial V}{\partial S} = 1 \)). In order remove this ‘non-linearity error’ and to conserve the overall accuracy of the quadrature scheme it is necessary to split the integral into two parts, with the boundary precisely at \( S = X \) (or \( y = 0 \)). The payoff function in the range \(-\infty < S < X\) is continuous, as is the function in the range \( X < S < \infty\). In fact, for a call option, since the value at maturity in the range \(-\infty < S \leq X\) is equal to zero, there is no contribution to the integral from this range (correspondingly, for a put there is no contribution from \( X \leq S < \infty\)). The range of integration in \( y \) in Eq. (9) is doubly infinite and so this must be truncated; however for \(|y| \gg |x|\), the \( B(x, y) \) term in Eq. (11) will tend to zero very quickly, and consequently leads to insignificant contribution to the overall integral outside a computationally quite modest range of \( y \). Practically, a range corresponding to asset price movements, up or down, within ten standard deviations during a time step proved more than adequate.

To value the call option at time \( t \) with asset price \( S_t \), the method is as follows (see also figure 2). The value of \( y \) at the discontinuity (here, the strike price) is defined to be \( b \), which in the case of a vanilla option is clearly zero. The truncated range of integration is taken to be \([y_{\text{min}}, y_{\text{max}}]\), and the integers \( N^+ \) and \( N^- \) are the number of steps taken in the half ranges \([b, y_{\text{max}}]\) and \([y_{\text{min}}, b]\) respectively. Since the value of the function below \( b \) for a vanilla call option is zero, this region does not contribute to the option valuation. For a call option, then,
\[
N^+ = \left\lfloor \frac{y_{\text{max}} - b}{\delta y} \right\rfloor \quad \text{and} \quad N^- = 0,
\]
where \( [a] \) denotes the integer closest to \( a \) and \( \delta y \) is the step size in \( y \). An implementation of Simpson’s rule would have the steps, \( i \), say, with \( 0 \leq i \leq N^+ \) with \( y = b + i\delta y \) and use values at these points and half way in between.

Using these new limits along with Eq. (11) the expression for the option price \( V(x, t) \) is
\[
V(x, t) \approx A(x) \int_0^{N^+\delta y} f(x, y) \, dy
\]
which, on implementing Simpson’s rule, Eq. (11), is
\[
\approx \frac{A(x)\delta y}{6} \left( f(x, 0) + 4f(x, \frac{1}{2}\delta y) + \left( \sum_{i=1}^{N^+-1} 2f(x, i\delta y) + 4f(x, (i + \frac{1}{2})\delta y) \right) + f(x, N^+\delta y) \right)
\]
and, by the definition of \( f(x, y) \) in Eqs. (9) and (13), this is
\[
\approx \frac{A(x)\delta y}{6} \left( B(x, 0)V(0, t + \Delta t) + 4B(x, \frac{1}{2}\delta y)V(\frac{1}{2}\delta y, t + \Delta t) + \ldots + B(x, N^+\delta y)V(N^+\delta y, t + \Delta t) \right).
\]
Figure 2: A demonstration of QUAD used to evaluate a European call option \( V(S_t, t) \). \( y \) is the transformed value of the underlying, \( S \), \( \delta y \) is the step size, \( X \) is the strike price, \( \Delta t \) is the time to maturity and functions \( A(x) \) and \( B(x, y) \) are as described in Eqs. 5 and 6 respectively.

This calculation is no more complicated than calculating the option value using a lattice method, however in QUAD the contribution at the previous timestep comes from many nodes representing different levels of the underlying asset not just two or three.

Figure 2 depicts the valuation process. A range of \( y \) values at maturity is chosen, increasing in steps of \( \delta y \) from \( y = 0 \) (the discontinuity) where the asset price, \( S \), is equal to the exercise price, \( X \), see Eq. (3). The option value at time \( t \) is then found by integrating Eq. (4) using Simpson’s rule via Eq. (17). Since a plain European call is being used as a simple example, valuation is achieved in a single time step and, in this case, is directly comparable with valuation using the known analytical solution commonly available in finance texts, such as Hull (2000).

The valuation can be quickly repeated for other possible current asset prices (transformed as \( x \) in Eq. (2)). In more complex option valuations these values one step back in time from maturity to another discontinuity become, in their turn, the basis for further integration backwards in time. This is reminiscent of backward valuation using a tree or finite difference lattice but with the important difference that large time steps are used, set by the period between discontinuities.

Table I shows the results of applying this method to the pricing of a European call option using Simpson’s rule. To reduce the distribution error still further, a simple Richardson-type extrapolation procedure was adopted. This was undertaken by performing calculations with two stepizes \( \delta y_1 \) and \( \delta y_2 \), producing option values \( V_1 \) and \( V_2 \), respectively. An improved value \( \hat{V} \) may be obtained through extrapolation, viz.

\[
\hat{V} = \frac{(\delta y_1)^d V_2 - (\delta y_2)^d V_1}{(\delta y_1)^d - (\delta y_2)^d},
\]  

(18)
where \((\delta y)^d\) is the rate at which the error decreases for the unextrapolated data. Assuming that the position of the discontinuity has been accurately gauged, then for Simpson’s rule \(d = 4\).

Absolute RMS errors have been calculated using sixteen different options, both with and without the extrapolation described by Eq. (18) (denoted by QUAD and QUAD\(_{\text{ext}}\) respectively). Results are benchmarked against those from a trinomial method (denoted by TRIN). Timings were obtained using a Pentium III 550 MHz computer and the NAG Fortran 95 v4.0 optimized compiler, and indicate excellent performance of the present scheme when compared with the standard trinomial tree method. However, a more demanding test of the scheme is its application to more complex options for which no analytic, closed form solutions are known; these are considered next.

<table>
<thead>
<tr>
<th>(N)</th>
<th>(\text{TRIN}(\text{time/seconds}))</th>
<th>(K)</th>
<th>(\text{QUAD}_{\text{ext}}(\text{time/seconds}))</th>
<th>(\text{QUAD}(\text{time/seconds}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.013879152 (0.00)</td>
<td>10</td>
<td>0.0000008539 (0.00)</td>
<td>0.000246595 (0.00)</td>
</tr>
<tr>
<td>200</td>
<td>0.010018265 (0.01)</td>
<td>20</td>
<td>0.000000087 (0.00)</td>
<td>0.000015034 (0.00)</td>
</tr>
<tr>
<td>500</td>
<td>0.001833449 (0.03)</td>
<td>30</td>
<td>0.000000007 (0.00)</td>
<td>0.000002956 (0.00)</td>
</tr>
<tr>
<td>1000</td>
<td>0.000600399 (0.14)</td>
<td>40</td>
<td>0.000000001 (0.00)</td>
<td>0.000000934 (0.00)</td>
</tr>
<tr>
<td>2000</td>
<td>0.000405737 (0.56)</td>
<td>50</td>
<td>0.000000000 (0.00)</td>
<td>0.000000382 (0.00)</td>
</tr>
</tbody>
</table>

Table 1: Parameters are as follows: \(S = 100\), \(X = 95/105\), \(\sigma = 0.2/0.4\), \(r = 0.06/0.2\), \(\Delta t = 0.5/1\), \(D_e = 0\). TRIN is the basic trinomial model described in Hull (2000) with \(N\) steps in the lattice. QUAD, QUAD\(_{\text{ext}}\) are results using Simpson’s rule, the latter extrapolated. \(K\) is such that \(\delta y = \sqrt{\Delta t/K}\).

### 11.5 Application to multiply-observed options

A discrete path-dependent option is one where the price of the underlying asset before maturity is important only at discrete points in time, referred to here as observation times. The valuation of the option in between these times is described by the Black-Scholes partial differential equation Eq. (1).

Consider a path-dependent option observed \(M\) times during its lifetime. Conceptually this is divided for valuation purposes into \(M\) separate options. These options, denoted \(V_m\), have maturities \(T_m\) where \(m = 1, 2, \ldots, M\). Let \(\Delta t_m\) be equal to \(T_m - T_{m-1}\). Given the current time, \(t\), the maturity of the complete option is at time \(T_M\) given by

\[
T_M = t + \sum_{m=1}^{M} \Delta t_m. \tag{19}
\]

Moving backwards from this maturity, whenever an observation time is encountered, Eq. (1), is evaluated for the period between that time and the next observation later in time. For example, with a Bermudan option the process could begin with integration of Eq. (1) covering the time period from the final possibility of early exercise until the maturity of the option.
At each observation time $T_m$ the option is priced for a full range of $x$. These $x$ values then become the $y$ values used in the integration to find the $x$ values at the previous time step, $T_{m-1}$ and so on until the complete option is valued. The values of $y_{max}$ and $y_{min}$ for a truncated range of integration will change at each $T_m$ and as a consequence will be denoted by $y_{max_m}$ and $y_{min_m}$ respectively. The corresponding integer values of $N^+$ and $N^-$ for the steps in the integration will be termed $N^+_m$ and $N^-_m$ respectively.

The probability of an asset following geometric Brownian motion moving more than $10$ standard deviations within a time period is so small as to be insignificant. For this reason, curtailing the range is possible such that, if $T = T_m - t$ is the time before maturity, then it is suggested that $y_{max_m} = x_0 + q$ and $y_{min_m} = x_0 - q$ are sufficient where

$$q = 10\sigma \sqrt{T}. \quad (20)$$

This assumes that $(r - D_c)$ is not unrealistically large, although adjustment for such values would be routine. For the same reason, it is not actually necessary to integrate over the entire range of $y$ for each value of $x$ at every time step, but instead, letting

$$q^* = 10\sigma \sqrt{\Delta T_m}, \quad (21)$$

the integral for each individual $x$ only needs to extend over the range $[x - q^*, x + q^*]$. A point of discontinuity in $V_m$ will be denoted by $b_m$. If the integer values of $i$ at time $T_m$ corresponding to this range are $i^+$ and $i^-$, where

$$i^+ = \left\lfloor \frac{x - b_m + q^*}{\delta y} \right\rfloor, \quad (22)$$

and

$$i^- = \left\lfloor \frac{x - b_m - q^*}{\delta y} \right\rfloor, \quad (23)$$

then Simpson’s rule should be employed for $i$ between $\max(i^-, N^-)$ and $\min(i^+, N^+)$. If the $\delta y$ chosen is too large, then results will clearly be inaccurate; fortuitously, it is generally very obvious when this is the case. As a rough guideline, $\delta y$ should always be smaller than $\sqrt{\Delta T_m}$.

The location of discontinuities in the payoff function will be known a priori for some classes of options. For example, for a vanilla call or put option with exercise price, $X$, a discontinuity occurs at $y = 0$; for a discrete barrier option, with a barrier position $B_m$ at time $T_m$, a discontinuity occurs where $y = \log(\frac{B_m}{x})$. For other classes of option the position of the discontinuity must be calculated at every observation time. For a Bermudan put option, for example, the location of the discontinuity, $b_m$, at time $T_m$ is where $X - S = P_m + 1(b_m, T_m)$. Taking this class of option as a prototype, the location is determined as follows. It is convenient to define a function, $g(x)$ say, which is zero at the desired value of $x$, $b_m$. For a Bermudan put is

$$g(x) = X(1 - e^x) - P_m(x, T_m). \quad (24)$$

The equation $g(b_m) = 0$ may then be solved using Newton-Raphson iteration, which is generally very quick to converge (rarely are more than ten iterations necessary to obtain machine precision). In brief, initial values for $b_m$ and $\Delta b_m$ are guessed (e.g 0 and $-0.01$, respectively) and a new value for $b_m$ is then:

$$b^*_m = b_m + \Delta b_m. \quad (25)$$

g($b_m$) and $g(b^*_m)$ must then be calculated, using Simpson’s rule for example, and then a new $\Delta b_m$ is given by

$$\Delta b^*_m = -\frac{\Delta b_m g(b^*_m)}{g(b^*_m) - g(b_m)}. \quad (26)$$
Valuation of a Discrete Down-and-Out Call Barrier Option (M=2)

<table>
<thead>
<tr>
<th>x = \log(S/E)</th>
<th>y = b_2 + N\gamma \delta y</th>
<th>y = b_2 + 3\delta y</th>
<th>y = b_2 + 3/2 \delta y</th>
<th>y = b_2 + 2\delta y</th>
<th>y = b_2 + \delta y</th>
<th>y = b_2 + 1/2 \delta y</th>
<th>y = b_2</th>
<th>S = B_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>y = x = b_1 + N\gamma \delta y</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
</tr>
<tr>
<td>y = x = b_1 + 3\delta y</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
</tr>
<tr>
<td>y = x = b_1 + 3/2 \delta y</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
</tr>
<tr>
<td>y = x = b_1 + 2\delta y</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
</tr>
<tr>
<td>y = x = b_1 + \delta y</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
</tr>
<tr>
<td>y = x = b_1 + 1/2 \delta y</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
</tr>
<tr>
<td>y = x = b_1</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
</tr>
</tbody>
</table>

\( V_0(y < b_1, T_1) = 0 \quad V_2(y < b_1, T_2) = 0 \)

Figure 3: A diagram depicting how QUAD is used to evaluate a discrete down and out call option with barrier at \( S = B_1 \) at time \( T_1 \). \( x \) and \( y \) are the transformed values of the underlying \( S \), \( b_1 \) is the transformed value of \( B_1 \), \( \delta y \) is the step size, the time to maturity is \( T_2 \). The option value at \( T_0 \) is calculated using an implementation of Simpson’s rule on the \( V_1(y \geq b_1, T_1) \) values.

The iterative process is repeated until \( g(b_m) \) is sufficiently small, with the programming caveat that iterations cease if the discontinuity is out of the range of integration, \([y_{\text{min}_m}, y_{\text{max}_m}]\) and hence, cannot be found.

Working back through time, from known final conditions, the first individual option \( V_M(x, T_{M-1}) \) can be priced for the entire range of underlying prices, in the manner described previously. The condition at the observation time (e.g. an early exercise condition) is imposed on these values and then these adjusted values provide the new final conditions for the next option, \( V_{M-1}(y, T_{M-1}) \). The next option is priced using these final conditions and the process is repeated until the value is found for \( V_1(x, T_0) \). Values for \( V_m(x_i, T_{m-1}) \) must be calculated for all values of \( x_i \), where

\[ x_i = b_m + i\delta y, \quad (27) \]

and also, in the case of Simpson’s rule, at the points half way between. The \( i \) will run from \( N_m^- \) to \( N_m^+ \), with one of these sometimes equal to zero (as in figure 2 for a European vanilla call option). For the final option, \( V_1 \), if the option value is just required at a single asset value, it is only necessary to calculate the value of \( V \) corresponding to \( S_t \), i.e. \( x = \log(S_t/X) \). Figure 3 shows how QUAD is used to price a discrete down-and-out call option with one barrier, \( B_1 \) at \( T_1 \).
11.5.1 High frequency observations and continuous American features

Extrapolation is not only useful in removing ‘distribution error’ caused by the finiteness of the quadrature stepsize $\delta y$ (using Eq. (18)) but can also be used to value options with more observations, or even with continuous, American style exercise. For continuous observation, the formula Eq. (18) may be utilized, but with $\delta y$ replaced by $T/M$, where $M$ is the number of observations. Although QUAD is tailored to be exceptionally powerful in valuing options with discrete observations, for those with continuous observation, its performance is at least comparable to all standard techniques, and, if the option is a particularly complex one with American features, then QUAD may prove very useful. To value options with many observations, where computing time is an issue (for example, three-dimensional lookbacks), extrapolation to many multiple observations (taken at moments equally distributed in time) can be performed using:

$$V_{M_3} = \frac{M_1^d (M_2^d - M_3^d) V_{M_1} + M_2^d (M_3^d - M_1^d) V_{M_2}}{M_3^d (M_2^d - M_1^d)},$$

(28)

where $M_1, M_2 < M_3$ are the number of observations and $V_M$ is the respective option value. Note that there are certain cases (e.g. random moving barriers) where such extrapolation is not applicable. A value of $d$ of either $\frac{1}{2}$ or 1 should be used depending on the particular problem. Discretely monitored American put options (Bermudans), for example, converge to the continuous solution at the rate of $\frac{1}{\sqrt{M}}$, where $M$ is the number of observations, hence $d = 1$; discrete barrier options converge with $\frac{1}{\sqrt{M}}$ and so $d = \frac{1}{2}$. For a given type option it is straightforward to gauge the appropriate value for $d$ simply by running a few trial calculations.

11.6 Bermudan put options

A Bermudan put option is one which has an early exercise feature, similar to an American put, but only at certain prescribed dates. For a Bermudan put option the value of $P_m$ (replacing $V$ by $P$ for a put option) at time $T_m$ is

$$\max(P_{m+1}(y = x, T_m), X - S),$$

(29)

where $P_{M+1} = \max(X - S, 0)$.

There will be a discontinuity in the second derivative of the option at the point where $P_{m+1}(T_m)$ is equal to $(X - S)$. Above this point the value of the option is given by $P_{m+1}(T_m)$, below it is equal to $(X - S)$. Therefore the integration is split into two components to avoid any non-linearity error.

For the first option period evaluated, $b_M$ is the point where $X = S$, and so $b_M = 0$. $P_{M+1}$ above this point is equal to zero. Newton-Raphson may be used to find subsequent values of $b_m$, as described in Section 3.2.

Above $b_m$, the value of $P_m(y \geq b_m, T_m)$ is $P_{m+1}(y = x, T_m)$, below the boundary the value is simply the payoff, namely $X - S$, and so $P_m(y \leq b_m, T_m) = X(1 - e^y)$. It is possible to continue using quadrature as before using Eq. (4)

$$P_m(x, T_{m-1}) \approx A(x) \int_{b_m}^{y_{\text{max}}} B(x, y) P_m(y \geq b_m, T_m) \, dy$$

$$+ A(x) \int_{y_{\text{min}}}^{b_m} B(x, y) \times X(1 - e^y) \, dy = I_1 + I_2.$$

However $I_2$ has a simple analytic form, related to the Black-Scholes solution for a put option within the bound $(-\infty, b_m]$. In this region, in order to save computational time, the procedure
is then to use this analytic (Black-Scholes) form for $I_2(x)$, namely

$$I_2(x) = X e^{-r \Delta t_m} N(-d_2) - X e^{-D_c \Delta t_m} N(-d_1),$$

(31)

where $N(.)$ is the cumulative probability distribution for a function that is normally distributed with a mean of zero and standard deviation of 1, and $d_1$ and $d_2$ are given by:

$$d_1 = \frac{x - b_m + (r - D_c + \sigma^2/2)(\Delta t_m)}{\sigma \sqrt{\Delta t_m}},$$

$$d_2 = d_1 - \sigma \sqrt{\Delta t_m}.$$  

(32)

11.7 Results

The accuracy and efficiency of QUAD are now compared with more familiar numerical techniques for valuing the different types of options, described in the previous section. Table 2 shows absolute root mean squared errors for a set of sixteen options. Results (QUAD) were generally extrapolated once (labelled QUAD$\_ext$) using Eq. (18) to further enhance accuracy. In the case of calculations performed using trapezoidal quadrature (in particular lookback options in three dimensions), extrapolation using Eq. (18) was carried out twice (QUAD$\_ext2$), first with $d = 2$, and then the resulting values (which may be regarded as having an error of $O((\delta y)^4)$) were the subject of a further application of Eq. (18) with $d = 4$. Choosing $\delta y = \sqrt{\Delta t_m}/K$, extrapolation was performed with values obtained from $K$ and $K - 2$, except in the case of lookback puts where $K$ and $K - 1$ were used. For the lookback call options it was performed twice using $K$, $K - 1$, and $K - 2$. Also, $D_c$ is considered to be 0 for all options valued here.

For multiple compounds, Bermudan put, moving barrier, and American call options, the results of the present method are compared with the basic trinomial model (Hull, 2000) denoted by TRIN. When valuing stationary discrete barrier options Ritchken’s (1995) approach (TRIN) is used but adapted, as suggested by Cheuk and Vorst (1996) in a straightforward manner to treat discrete rather than continuous barriers. To value lookback options with a similarity reduction, Cheuk and Vorst’s (1997) two-dimensional binomial method (BINC), incorporating the placing of extra time steps between observations, (and adapted for $\bar{A}_M$ different from $S_t$) is used for comparison. Finally, for the barrier lookback option, the benchmark is a basic three-dimensional trinomial model adapted for multiple steps between observations (TRIN3). For each case the exact solutions were obtained by using the current scheme when the $\delta y$ was small enough for the price to have converged to the given degree of accuracy ($\sim 1 \times 10^{-10}$). As an indication of the efficiency of the QUAD, average computational times are also displayed. As before, timings were taken on a Pentium III 550 MHz system, using the NAG Fortran 95 v4.0 compiler with optimization.

Although the basic trinomial tree approach was used as a benchmark for moving barrier options, improved results can be obtained using the AMM method of Ahn, Figlewski, and Gao (1999), though at a cost of considerably more complex programming. For comparison, AMM8 with eighty time steps between observations gives absolute RMS of 0.00034 and takes an average time of 8.65 seconds whereas QUAD$\_ext$ achieves better accuracy in a mere 0.28 of a second.

The results are shown in Table 2 and are excellent; for the two-dimensional models results in a given time are consistently orders of magnitude better than the benchmarks for all options considered. In the case of fixed-strike lookback discrete barrier options penny accuracy is achieved in 1.16 seconds whereas the trinomial method is, on average, 40 cents out after 23 minutes. The superiority of QUAD over present methods is clear.
### Absolute Root Mean Squared Errors

<table>
<thead>
<tr>
<th>N between obs.</th>
<th>TRIN (time/seconds)</th>
<th>K</th>
<th>QUAD$_{ext}$ (time/seconds)</th>
<th>QUAD (time/seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0.009969399 (1.68)</td>
<td>6</td>
<td>0.000121107 (0.41)</td>
<td>0.000460985 (0.28)</td>
</tr>
<tr>
<td>50</td>
<td>0.009670104 (2.59)</td>
<td>8</td>
<td>0.000008423 (0.75)</td>
<td>0.000140402 (0.47)</td>
</tr>
<tr>
<td>60</td>
<td>0.008317414 (3.86)</td>
<td>10</td>
<td>0.000001673 (1.19)</td>
<td>0.000056567 (0.72)</td>
</tr>
<tr>
<td>70</td>
<td>0.008159162 (5.93)</td>
<td>12</td>
<td>0.000000481 (1.74)</td>
<td>0.000027042 (1.02)</td>
</tr>
<tr>
<td>80</td>
<td>0.008055604 (9.23)</td>
<td>20</td>
<td>0.000000035 (5.02)</td>
<td>0.00003461 (2.77)</td>
</tr>
</tbody>
</table>

**Discrete down and out barrier call options:**

<table>
<thead>
<tr>
<th>N between obs.</th>
<th>TRIN (time/seconds)</th>
<th>K</th>
<th>QUAD$_{ext}$ (time/seconds)</th>
<th>QUAD (time/seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0.147934594 (1.84)</td>
<td>6</td>
<td>0.000143546 (0.41)</td>
<td>0.000283021 (0.28)</td>
</tr>
<tr>
<td>50</td>
<td>0.091196513 (3.11)</td>
<td>8</td>
<td>0.00012456 (0.75)</td>
<td>0.000082115 (0.47)</td>
</tr>
<tr>
<td>60</td>
<td>0.074112820 (4.06)</td>
<td>10</td>
<td>0.00002472 (1.19)</td>
<td>0.000032372 (0.72)</td>
</tr>
<tr>
<td>70</td>
<td>0.040418332 (6.89)</td>
<td>12</td>
<td>0.00000706 (1.74)</td>
<td>0.000015296 (1.03)</td>
</tr>
<tr>
<td>80</td>
<td>0.062995092 (8.58)</td>
<td>20</td>
<td>0.00000025 (5.04)</td>
<td>0.000001925 (2.78)</td>
</tr>
</tbody>
</table>

**Moving discrete down and out barrier call options:**

<table>
<thead>
<tr>
<th>N between obs.</th>
<th>TRIN (time/seconds)</th>
<th>K</th>
<th>QUAD$_{ext}$ (time/seconds)</th>
<th>QUAD (time/seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>0.007175522 (0.01)</td>
<td>6</td>
<td>0.000183337 (0.01)</td>
<td>0.000485888 (0.01)</td>
</tr>
<tr>
<td>100</td>
<td>0.002157462 (0.04)</td>
<td>8</td>
<td>0.00014998 (0.01)</td>
<td>0.000144786 (0.01)</td>
</tr>
<tr>
<td>200</td>
<td>0.001524073 (0.14)</td>
<td>10</td>
<td>0.00002991 (0.02)</td>
<td>0.000057748 (0.01)</td>
</tr>
<tr>
<td>300</td>
<td>0.000782219 (0.31)</td>
<td>12</td>
<td>0.00000855 (0.03)</td>
<td>0.000027458 (0.02)</td>
</tr>
<tr>
<td>500</td>
<td>0.000356063 (0.87)</td>
<td>20</td>
<td>0.00000030 (0.08)</td>
<td>0.00003487 (0.05)</td>
</tr>
</tbody>
</table>

**Multiply-compounded call options:**

<table>
<thead>
<tr>
<th>N between obs.</th>
<th>TRIN (time/seconds)</th>
<th>K</th>
<th>QUAD$_{ext}$ (time/seconds)</th>
<th>QUAD (time/seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>0.007175522 (0.01)</td>
<td>6</td>
<td>0.000183337 (0.01)</td>
<td>0.000485888 (0.01)</td>
</tr>
<tr>
<td>100</td>
<td>0.002157462 (0.04)</td>
<td>8</td>
<td>0.00014998 (0.01)</td>
<td>0.000144786 (0.01)</td>
</tr>
<tr>
<td>200</td>
<td>0.001524073 (0.14)</td>
<td>10</td>
<td>0.00002991 (0.02)</td>
<td>0.000057748 (0.01)</td>
</tr>
<tr>
<td>300</td>
<td>0.000782219 (0.31)</td>
<td>12</td>
<td>0.00000855 (0.03)</td>
<td>0.000027458 (0.02)</td>
</tr>
<tr>
<td>500</td>
<td>0.000356063 (0.87)</td>
<td>20</td>
<td>0.00000030 (0.08)</td>
<td>0.00003487 (0.05)</td>
</tr>
</tbody>
</table>

Table 2a: Parameters in each case are as follows. For discrete barriers: $M = 6/125, B = 90/99, S = 100, X = 95/105, \sigma = 0.2/0.4, r = 0.06, T_M - t = 0.5$. For moving barriers: $M = 5/125; B_4, B_9, B_{14}, \ldots, B_{124} = 94; B_3, B_8, \ldots, B_{123} = 93; B_2, \ldots, B_{122} = 92; B_1, \ldots, B_{121} = 91; B_0, \ldots, B_{125} = 90; S = 100, X = 95/105, \sigma = 0.2/0.4, r = 0.06, T_M - t = 0.5/1$. For multiply compounded call options: $M = 4/6, S = 100, X_M = 95/105, X_{1,2,\ldots,M-1} = 2/3, \sigma = 0.2/0.4, r = 0.06, T_M - t = 1$. 

77
### Table 2b: Parameters in each case are as follows. For Bermuda puts:

\[
N = 6, S = 10, X = 95/105, \sigma = 0.2/0.4, r = 0.06, T_M - t = 0.5/1.
\]

For American calls with changing strike price:

\[
N = 6, S = 10, X_M = 95/105, X_m = X_{m+1} - \Delta X \text{ (where } \Delta X = 1/0.5 \text{ for } N = 6, 12 \text{ respectively),} \]

\[
\sigma = 0.2/0.4, r = 0.06, T_M - t = 0.5/1.\]

For American calls with dividend payments:

\[
N = 3, S = 10, X = 95, D = 2/3, \sigma = 0.2/0.4, r = 0.06, T_M - t = 1.5/3.
\]
<table>
<thead>
<tr>
<th>$N$ between obs.</th>
<th>BINV (time/seconds)</th>
<th>$K$</th>
<th>QUAD$_{ext}$ (time/seconds)</th>
<th>QUAD (time/seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lookback put options with payoff $A - S$:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>0.067937970 (0.97)</td>
<td>2</td>
<td>0.000038073 (1.78)</td>
<td>0.000205447 (1.08)</td>
</tr>
<tr>
<td>60</td>
<td>0.088310529 (2.18)</td>
<td>3</td>
<td>0.000001070 (2.81)</td>
<td>0.000039762 (1.73)</td>
</tr>
<tr>
<td>80</td>
<td>0.061800587 (3.86)</td>
<td>4</td>
<td>0.000000134 (4.38)</td>
<td>0.000027261 (2.65)</td>
</tr>
<tr>
<td>100</td>
<td>0.041746916 (6.21)</td>
<td>5</td>
<td>0.000000029 (6.42)</td>
<td>0.000005101 (3.77)</td>
</tr>
<tr>
<td>120</td>
<td>0.056115193 (8.70)</td>
<td>6</td>
<td>0.000000009 (8.98)</td>
<td>0.000002456 (5.21)</td>
</tr>
<tr>
<td>Fixed strike lookback call options with barrier:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.071426815 (1.92)</td>
<td>10</td>
<td>0.007466455 (1.16)</td>
<td>0.119313542 (0.68)</td>
</tr>
<tr>
<td>20</td>
<td>0.798811819 (15.39)</td>
<td>14</td>
<td>0.000458519 (2.78)</td>
<td>0.06134855 (1.52)</td>
</tr>
<tr>
<td>30</td>
<td>0.712570741 (63.75)</td>
<td>16</td>
<td>0.000182039 (5.41)</td>
<td>0.039947736 (2.82)</td>
</tr>
<tr>
<td>60</td>
<td>0.482999391 (445.80)</td>
<td>20</td>
<td>0.000078422 (9.55)</td>
<td>0.026971794 (4.85)</td>
</tr>
<tr>
<td>90</td>
<td>0.412661235 (1427.48)</td>
<td>28</td>
<td>0.000016221 (31.00)</td>
<td>0.013740692 (15.33)</td>
</tr>
</tbody>
</table>

Table 2c: Parameters in each case are as follows. For lookback puts: $M = 6/125$, $S = 100$, $\overline{A_0} = 100/105$, $X = 95$, $\sigma = 0.2/0.4$, $r = 0.06$, $T_M - t = 0.5/1$. For lookback calls with barrier: $M = 6/26$, $B = 90/100$, $S = 100$, $\overline{A_0} = 100/105$, $X = 95$, $\sigma = 0.2/0.4$, $r = 0.06$, $T_M - t = 0.5$. Table 2: The $M$ barriers or observations are equally spaced throughout the time period. TRIN indicates the relevant trinomial model for the option; BINC is Cheuk and Vorst’s (1997) binomial model; TRIN3 is a three-dimensional trinomial model for lookbacks. QUAD are the results obtained using the given method with Simpson’s rule except in the case of the lookback with barrier which uses the trapezium rule. QUAD$_{ext}$ and QUAD$_{ext2}$ are the results extrapolated once and twice respectively. In the case of lookback barrier options $q = 7.5\sigma\sqrt{T}$ and $q^* = 7.5\sigma\sqrt{\Delta t_m}$. $N$ is the number of steps in between observations in the trinomial tree (i.e. total number of time steps is $N \times M$), and $K$ is such that $\delta y = \sqrt{T_{\Delta t_m}}/K$. ‘Exact’ solutions were obtained using QUAD$_{ext}$ with $K > 150$. 

79
11.8 Overview

Numerical methods based on quadrature are a fast and extremely accurate means for option pricing. In effect, between observations it is analogous to a multinomial tree method, but uses only a single time step. The assumption that in between observations the option behaves as a European option enables the problem to be broken down into a series of integrals which, coupled with numerical quadrature, are used to value the option over the whole required period. The method benefits from the simple and exact placement of ‘nodes’ on boundaries/discontinuities (whether stationary or moving), thus removing any non-linearity error, a problem that plagues other numerical techniques; indeed, much of the success of the method can be attributed to this ability; the results are extremely accurate. As an example of its effectiveness, an option that would take hours, if not days, to price with penny accuracy using a trinomial method (such as the lookback barrier in Table 2) can now be priced correct to four decimal places in just seconds. The simplicity and universality for all discretely monitored options shows this method to be considerably superior to the traditional lattice or finite-difference methods, making it a powerful addition to the mathematical financier’s toolkit.
12 Numerical methods in general

- We have seen the four fundamental numerical methods used in C/F: Monte Carlo, Binomial lattices, Finite difference and Quadrature methods.
- Each one will be useful for different types of derivative pricing problems and it is important that you choose the right method for the right problem.
- The main issues are:
  - Accuracy/Speed/Ease of programming
  - Hedging estimation
  - Different processes
  - Exotic features
  - More than one Brownian motion

12.1 Accuracy

- For a basic option pricing problem in one underlying asset the most accurate methods are the Crank-Nicolson finite difference method and QUAD.
- However, for these basic options either analytic solutions are available or any of the other methods will provide perfectly acceptable levels of accuracy given the available computing power.
- The real test comes with different types of problems.

12.2 Hedging estimation

- We have not really discussed calculation of the ‘greeks’ or implied volatility but as one would expect these are easy to estimate using binomial lattices and especially finite difference methods as the greeks involve estimating derivatives.
- It is less easy with Monte Carlo methods, as you typically have to perturb the parameter in question and estimate the derivative from the answer. To avoid error from random numbers you also have to use the same random numbers.
- This can still cause problems for options with discontinuities - like barrier options.

12.3 Different processes

- If we assume a process other than GBM then we things become a little more difficult.
- Such problems often arrive when pricing interest rate derivatives where the interest rate follows a mean reverting process (e.g. Vasicek; CIR etc.)
- Here it is simple to adapt the Monte Carlo method to deal with this as we typically have an SDE formulation and we can use the Euler approximation.
- In certain cases it is also possible to adapt the binomial tree but here we require the tree to match the new distribution.
- More problematic are finite difference methods where you need to derive an entire new PDE (not actually possible in the more sophisticated interest rate model, HJM).
12.4 Exotic features

- Exotic features are often path dependent features, such as payoffs as a function of the average or maximum option value. They may also be knock-out features such as barriers.

- Typically path dependent options require another dimension for lattice and finite difference methods and some level of programming difficulty.

- However, the Monte Carlo method is easy to adapt for path dependent options, although you now need to include more time steps. Here there is thus a trade off between the accurate, slow lattice methods and the inaccurate fast Monte Carlo method.

- By comparison if we consider the convergence as a function of the computational effort (or work) then we typically see Monte Carlo being $O(w^{-1/4})$ as the computation is time steps x sample paths. For one underlying and one path dependent feature a lattice is $O(w^{-1/3})$ and Crank-Nicolson is $O(w^{-2/3})$.

- The problem becomes more pronounced when there are early exercise features too as in this case it is most inefficient to use Monte Carlo as the convergence now is uncertain and it is far more difficult to program.

- However, if the number of underlying assets are also increased then this becomes a very difficult problem indeed as the lattice/f-d methods will struggle with the dimensionality but the Monte-Carlo method will struggle with the early exercise.

12.5 More than one Brownian motion

- This could be either more than one underlying asset, stochastic volatility or interest rates or a multifactor interest rate model.

- In each of these if you have $d$ Brownian motions the effort convergence as a function of the computational effort $w$ is as follows: Monte-Carlo: $O(w^{-1/2})$, Lattice and Explicit finite-difference: $O(w^{-1/(d+1)})$ and Crank-Nicolson $O(w^{-2/(d+1)})$.

- Thus for $d > 3$, Monte Carlo methods should be preferred. Again there is a large problem when there are also American features as it is very difficult to estimate the convergence from these Monte-Carlo schemes and it may well be that for $d < 6$, you could attempt to use finite-difference methods or a binomial lattice.

- Quadature methods:
  - These are very accurate, often with $1/N^8$ convergence, this means that even as you add underlying assets the convergence as a function of work is only $O(w^{-8/(d+1)})$.
  - These methods will still have memory requirements for large $d$, but require less nodes due to their excellent accuracy. They also cope well with early exercise features.
  - The only problem occurs when there are non-GBM underlying processes as the method relies on having analytic transition density functions. However, there are functional approximations that may overcome these difficulties...
  - See Andricopoulos et al., 2003
12.6 Overview

- We have seen the final adaptation to the finite difference method: body-fitted co-ordinates that enable you to even remove non-linearity error for American options.

- There was then a brief discussion about the advantages of each of the numerical methods