Trinomial and Finite Difference Option Pricing

A MATH345 University of Wollongong Project

Andrew Morris

Spring 2011

Contents

Chapter 1

Introduction

It is somewhat widely to known to those in financial mathematics that the trinomial and finite difference methods of option pricing are equivalent. However, it is not so widely known exactly what that equivalence means, or how it can be demonstrated. The bulk of this project has been about understanding this equivalence and its implications.

The other part of this project is about the implementation of these methods. This was done in C++, and enabled the exploration of variant methods, and how to optimise the calculation.

Many details included in this report are already known to those familiar with mathematical finance. They are included to make this report as widely accessible as possible, with the goal of being readily understood at the undergraduate level.

Chapter 2

Background

2.1 | Options

An option is an entitlement to engage in a particular transaction, but not an obligation to engage in that transaction. Thus the option's owner has the 'option' of engaging in the transaction. More specifically, taking the simplest relevant case, a European style call option entitles its owner to buy a particular asset (the **underlying**) at a particular price (the **strike**) at a particular time (the **expiry date**).

For example, you might pay $400 for an option to buy 100 shares of BHP in 3 months' time at $85 each. After 3 months, if BHP is trading at $95, you can use the option to pay just $8500 to buy 100 BHP shares. If you wish, you can immediately sell those shares for $9500, realising a gain of $1000; a profit of $600. However, after 3 months, if BHP is trading at $80 (or even $84.90) you will not wish use the option to buy it for $85. This is because you can just buy it from the market for $80, so your option expires worthless, and you just lose the $400 premium you paid.

However, looking at options this way conceals their use in risk management. A business for example may need to purchase some commodity at a future date, and purchase an option on that commodity. Because the business plans to buy the commodity anyway, purchasing from the market if not via the option, the option is best understood as providing the business with a price *ceiling*. In this case, the business *prefers* the option to expire worthless.

In addition to call options, there are put options. Put options are exactly like call options except that the optional transaction is to sell an asset instead of to buy an asset. In Chapter 12 of *Basic Economics*[1], Thomas Sowell describes a hypothetical wheat farmer entering a futures contract to sell wheat he has not yet planted. A futures contract is like having an option, except that you have the obligation to engage in the transaction, not just the option to do so if you wish. The wheat farmer enters the futures contract to avoid the risk of the wheat price being low after harvest, as the contract locks in a price today. Alternatively, the farmer could invest in put options, and this would provide him with a price *floor*.

Mathematically, options are understood in terms of **payoff functions**, often denoted $Q(S)$. If you own a European call option at expiry ($t = T$) with strike price K and the underlying price S is greater than K, then you will exercise the option, buying at K and (if you like) selling at S. This amounts to a **payoff** of $S - K$. On the other hand, if the underlying price S is less than K, you will not exercise the option, which is a payoff of 0. Combining these, the general payoff is $\max(S - K, 0)$, or:

$$V(S,T) = Q(S) = \max(S - K, 0). \qquad \text{- Eq 2.1.1}$$

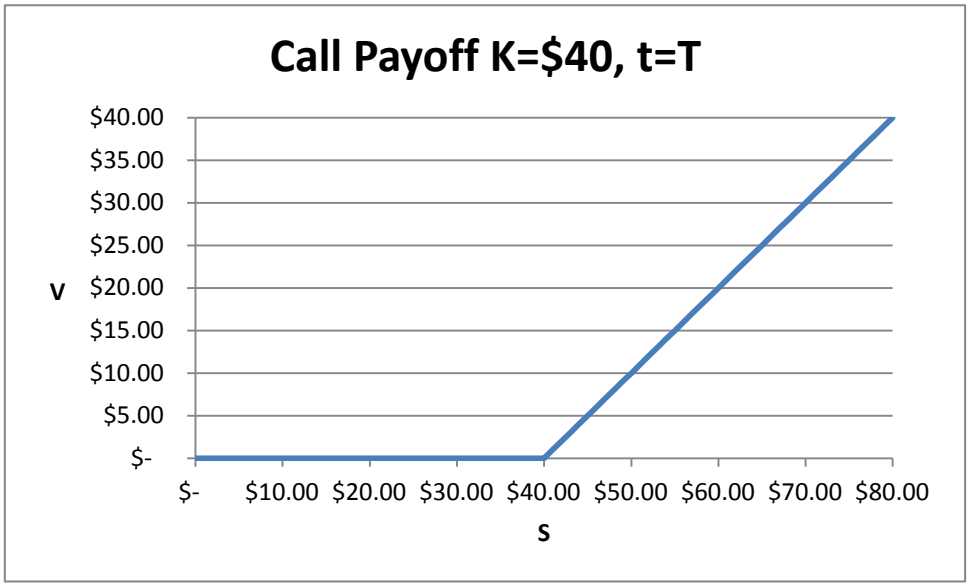

Fig 2.1.1 – Payoff function for a call option with a $40 strike price.

Similarly, for put options the gain is realised if the underlying price S is below K, and the payoff is instead:

$$V(S,T) = Q(S) = \max(K - S, 0). \quad \text{- Eq 2.1.2}$$

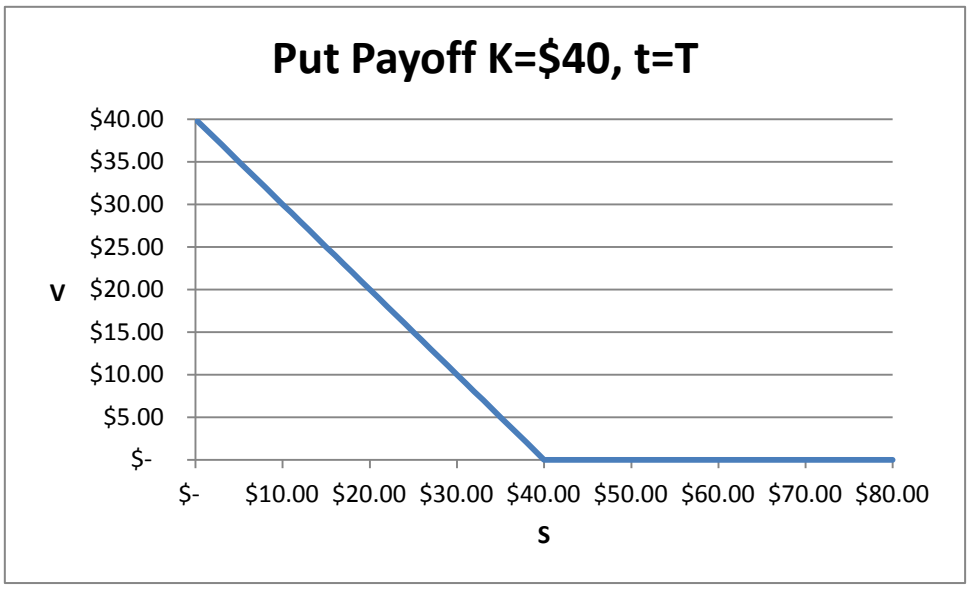

Fig 2.1.2 – Payoff function for a put option with a $40 strike price.

This provides the essential boundary condition (final condition) to use the trinomial and finite difference methods. In addition to European style options discussed so far, there are American style options, which allow the transaction to take place at any time until the expiry date, rather than on the expiry date. Mathematically, this imposes a lower limit on the value of an American option at all times, as the owner will exercise the option early if its value were to ever fall below the payoff function. For call options, it so happens that this is inconsequential, but for put options, it adds an additional boundary condition[3]:

$$V\big(S_f(t), T\big) = K - S_f(t). \qquad \text{- Eq 2.1.3}$$

$S_f(t)$ is called the "optimal exercise boundary"; the boundary below which you really should exercise early, and is defined by[3]:

$$\frac{\partial V}{\partial S}\big(S_f(t), t\big) = -1. \qquad \text{- Eq 2.1.4}$$

This makes the location of the optimal exercise boundary part of the problem itself, creating a moving boundary problem. For the trinomial and finite difference methods, it is fairly trivial to incorporate this condition, but it makes the analytical solution far more difficult. However, it was solved by Zhu in 2005[2].

2.2 | The Black-Scholes Framework

To price options, a model of how asset prices change over time is required. Of course, no one knows what the price of Google shares will be tomorrow, so the model is probabilistic. Asset prices are modelled by a stochastic equation:

$$\frac{dS}{S} = \mu dt + \sigma dX. \qquad \text{- Eq 2.2.1}$$

Note: dX is a sample from a normal distribution with mean zero and standard deviation $\sqrt{dt}$.

This can be used to produce simulations like the one in figure 2.2.1.

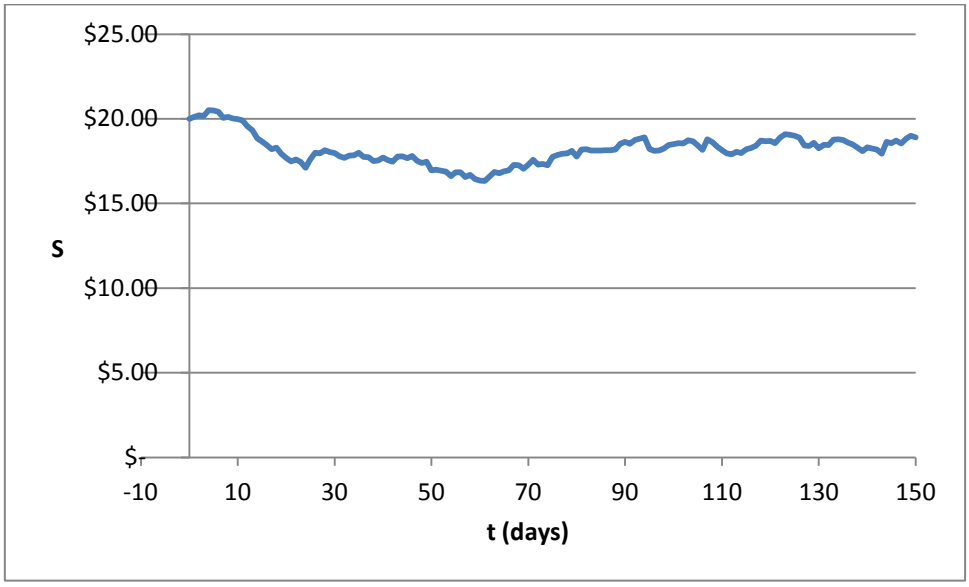

Fig 2.2.1 – A simulation of an asset price under the stochastic process in eq 2.2.1.

The Black-Scholes equation is derived from this model. It is the partial differential equation we will use to derive finite difference methods:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0. \qquad \text{- Eq 2.2.2}$$

The analytical solution to option pricing is found by transforming the Black-Scholes equation to the heat equation:

$$\frac{\partial U}{\partial t} = \kappa \frac{\partial^2 U}{\partial x^2}. \qquad \text{- Eq 2.2.3}$$

Using the final condition for call options (2.1.1) as well as a few other boundary conditions not required for the finite difference and trinomial methods, the solution is the Black-Scholes formula:

$$V(S,t) = N(d_1)S - N(d_2)Ke^{-r(T-t)} \qquad \text{- Eq 2.2.4}$$

where d_1, d_2, $N(x)$ are defined by:

$$d_1 = \frac{\log\left(\frac{S}{K}\right) + \left(r + \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}} \quad \text{- Eq 2.2.5}$$

$$d_2 = \frac{\log\left(\frac{S}{K}\right) + \left(r - \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}} \quad \text{- Eq 2.2.6}$$

$$N(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-z^2} dz. \quad \text{- Eq 2.2.7}$$

2.3 | Interpreting Substitution as Scaling

Whenever substitution is used in algebraic manipulation, it can be interpreted instead as a scaling process. For example, suppose we have this equation approximating some population of bacteria:

$$N = 100e^{5t} \qquad \text{- Eq 2.3.1}$$

Here N is the population of the culture and t is time in days. Figure 2.2.1 plots this trend.

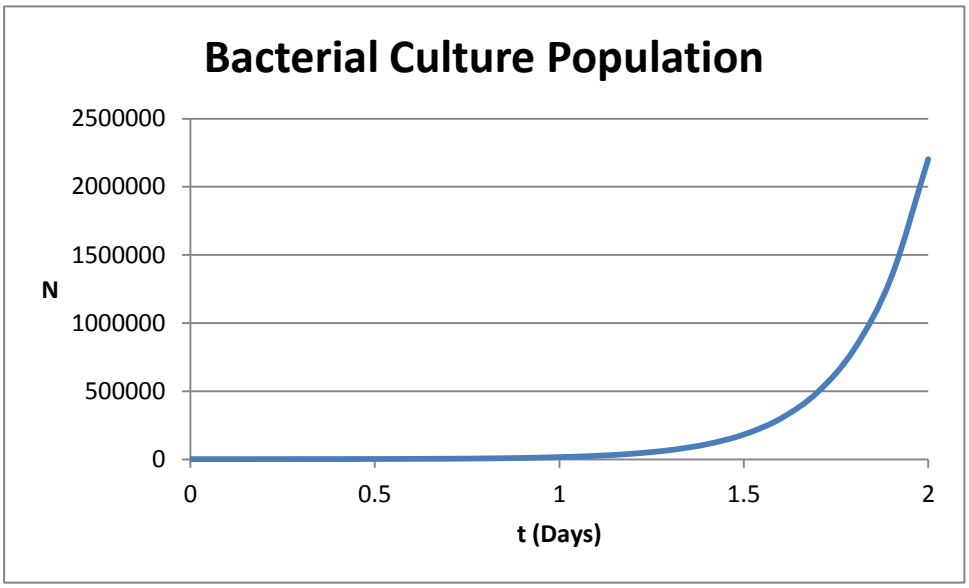

Fig 2.2.1 – Plot of the exponential growth phase of a hypothetical bacterial culture.

Now, the population explosion makes it difficult to see the growth behaviour in the early stages. To make it more reasonable we perform the substitution:

$$u = \log_{10} N, \qquad \text{- Eq 2.3.2}$$

Obtaining:

$$\begin{gathered} 10^u = 100e^{5t} \\ u = \log_{10}(100 * e^{5t}) \\ u = 2 + \log_{10} e^{5t} \\ u = 2 + (5\log_{10} e)t. \end{gathered} \qquad \text{- Eq 2.3.1}$$

Figure 2.2.2 plots this relationship.

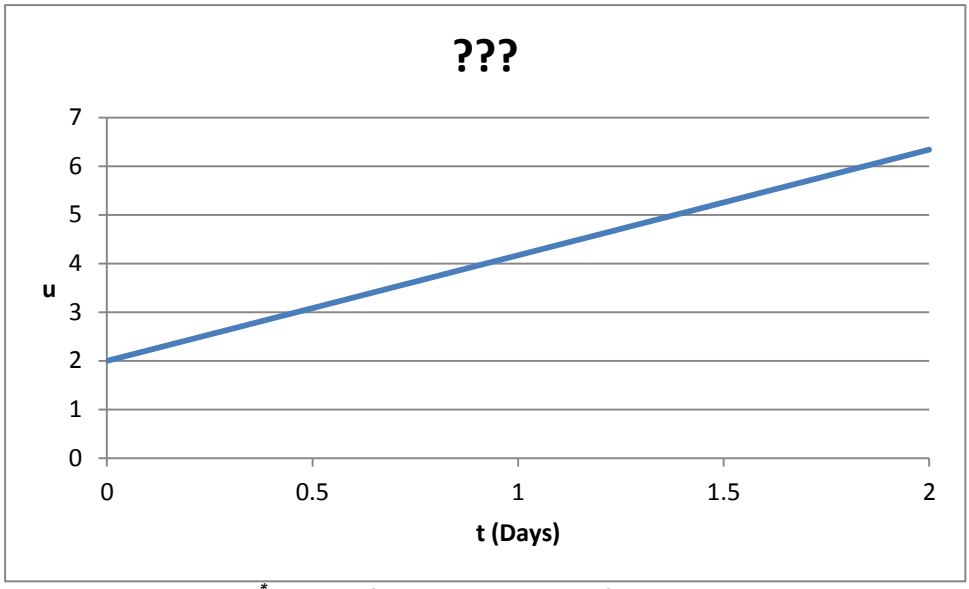

Fig 2.2.2[*] – Plot of bacterial growth after substitution.

However, this is not what we want. Instead, we'd like to re-interpret the u-axis as the N-axis again, only scaled. We do this by recognising equivalence between u values and N values. For example, when $u = 2$, $N = 10^2 = 100$. All we have to do is replace 1 with 10, 2 with 100, 3 with 1000 etc. making the vertical axis the N axis again, producing scaling instead of substitution. This is illustrated in figure 2.2.3.

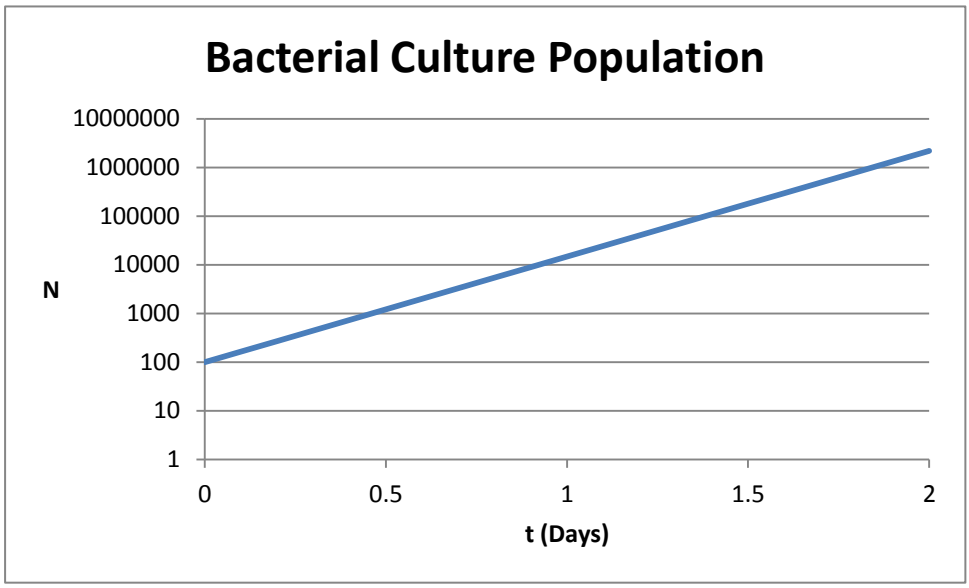

Fig 2.2.3 – The exponential growth phase of a bacterial culture under a logarithmic scale.

[*]The title '???' is intended to convey the lost understanding of what the plot means.

2.4 | Finite Differences

Understanding finite differences is paramount to understanding how finite difference methods can be constructed as equal to trinomial methods. For this reason, and because this report should be accessible at the undergraduate level, this section has a fairly complete account of everything you need to know about finite differences to understand the equivalence.

Finite differences are like doing calculus in reverse. Remember the definition of the derivative:

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}. \qquad \text{- Eq 2.4.1}$$

In calculus, we take this gradient in the limit of $\Delta x \to 0$ as the derivative. With finite differences, we start with a derivative, pick a small but non-zero value for Δx, (or Δt, etc.) and use that gradient as an approximation to the derivative. These constructs are illustrated in figure 2.4.1.

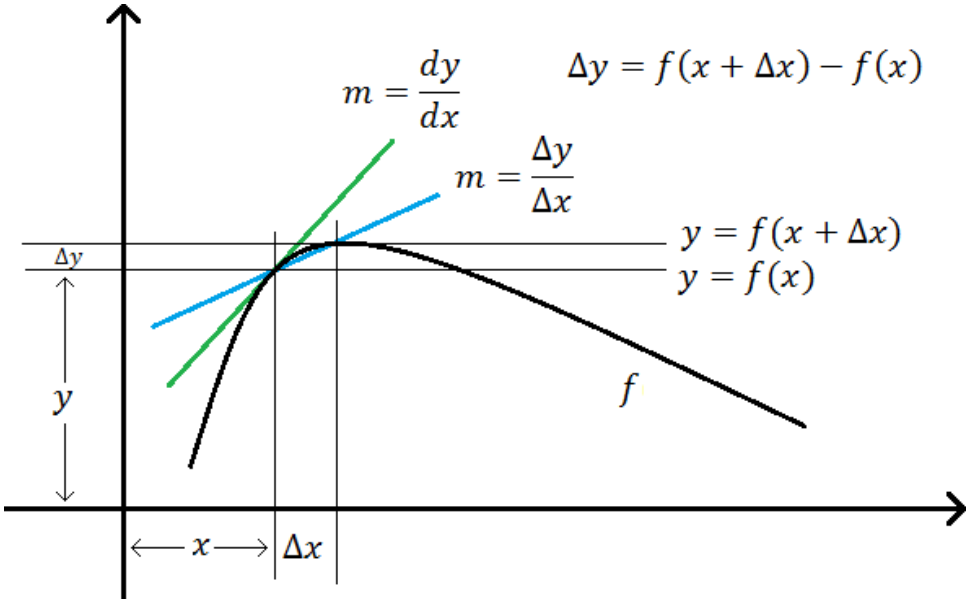

Fig 2.4.1 – Illustration of derivatives and finite differences.

When applying finite differences, in addition to discretising derivatives, we discretise the functions involved, so that they are no longer continuous, instead consisting of a series of **nodes**. Nodes are separated by the step size(s) of the independent variable(s), and are indexed using integers. This is illustrated in figure 2.4.2.

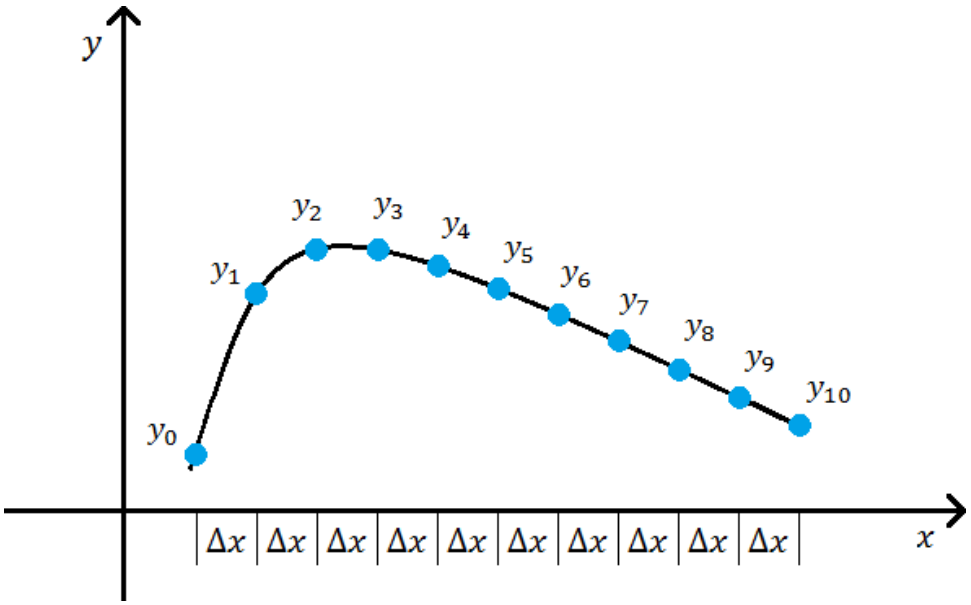

Figure 2.4.2 – Finite difference methods discretise functions into series of nodes.

With nodes like this setup, we expand differences of dependent variables to differences between adjacent nodes, e.g.:

$$\frac{\Delta y}{\Delta x} = \frac{y_{n+1} - y_n}{\Delta x}. \qquad \text{- Eq 2.4.2}$$

If we have a differential equation, such as:

$$\frac{dy}{dx} = g(x), \qquad \text{- Eq 2.4.3}$$

then we can tie this all together to use a finite difference method to approximate the function without analytically solving the differential equation:

$$\frac{\Delta y}{\Delta x} = g(x)$$

$$\frac{y_{n+1} - y_n}{\Delta x} = g(n\Delta x)$$

$$y_{n+1} = y_n + g(n\Delta x)\Delta x. \qquad \text{- Eq 2.4.4}$$

Note $x = n\Delta x$, this comes from $y_0 = y(0)$.

Just like in analytic solutions, we need y_0 or some other boundary condition to determine the particular solution. So if we know y_0, and $g(x)$ for any x, we have our approximations for y_1, y_2, etc. as:

$$y_1 = y_0 + g(0)\Delta x$$

$$y_2 = y_0 + \big(g(0) + g(\Delta x)\big)\Delta x$$

$$y_n = y_0 + \left(\sum_{i=0}^{n-1} g(i\Delta x)\right)\Delta x. \qquad \text{- Eq 2.4.5}$$

So far we have talked about only *forward* differences. There are also *backward* differences:

$$\frac{\Delta y}{\Delta x} = \frac{y_n - y_{n-1}}{\Delta x}. \qquad \text{- Eq 2.4.6}$$

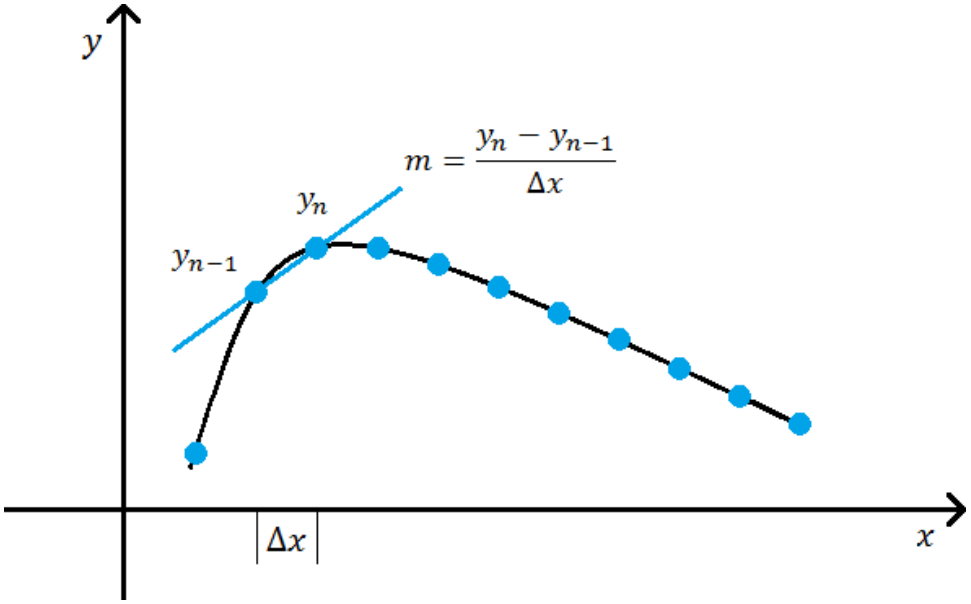

Fig 2.4.3 – A backward difference.

In addition there are central differences:

$$\frac{\Delta y}{\Delta x} = \frac{y_{n+1} - y_{n-1}}{2\Delta x}. \qquad \text{- Eq 2.4.7}$$

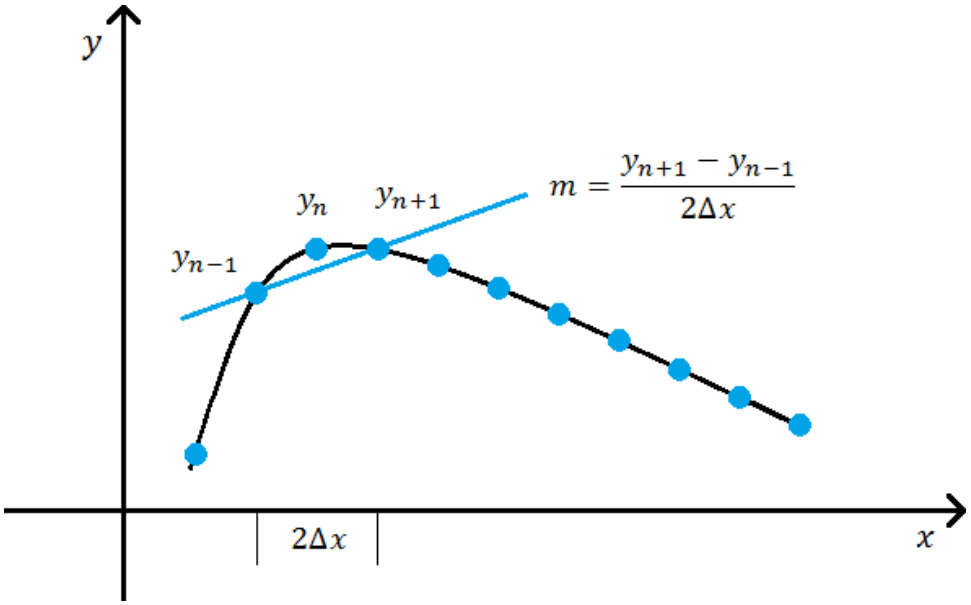

Fig 2.4.4 – A central difference.

When constructing a finite difference method, we are entitled to choose which kind of difference to use. They are all approximations which (under normal/most conditions) converge to the derivative.

Formally, when discretising a differential equation to apply the finite difference method, we simply replace derivatives with difference operators. This also works for second and higher order derivatives, for example:

$$\frac{d^2y}{dx^2} \approx \frac{\Delta^2 y}{\Delta x^2} = \frac{\Delta}{\Delta x}\left(\frac{\Delta y}{\Delta x}\right) = \frac{\Delta(y_{n+1} - y_n)}{\Delta x^2} = \frac{\Delta y_{n+1} - \Delta y_n}{\Delta x^2}. \quad \text{- Eq 2.4.8}$$

A forward difference was used to expand the difference operator the first time. Usually this is balanced out by taking a backward difference the second time:

$$\frac{\Delta y_{n+1} - \Delta y_n}{\Delta x^2} = \frac{(y_{n+1} - y_n) - (y_n - y_{n-1})}{\Delta x^2}$$

$$\frac{\Delta y_{n+1} - \Delta y_n}{\Delta x^2} = \frac{y_{n+1} - 2y_n + y_{n-1}}{\Delta x^2}. \quad \text{- Eq 2.4.9}$$

Finally, finite difference methods can be applied to partial differential equations, involving more than one independent variable. For example, a finite difference scheme for the heat equation can be derived like so:

$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2}$$

$$\frac{\Delta U}{\Delta t} = \kappa \frac{\Delta^2 T}{\Delta x^2}$$

$$\frac{T_m^{(n+1)} - T_m^{(n)}}{\Delta t} = \kappa \frac{T_{m+1}^{(n)} - 2T_m^{(n)} + T_{m-1}^{(n)}}{\Delta x^2}$$

$$T_m^{(n+1)} = T_m^{(n)} + \frac{\kappa \Delta t}{\Delta x^2}\left(T_{m+1}^{(n)} - 2T_m^{(n)} + T_{m-1}^{(n)}\right). \quad \text{- Eq 2.4.10}$$

The two indexes, m and n are a reflection of the two independent variables x and t. Parentheses around the superscripts are there to indicate that they indicate locations in sequences rather than powers as in $3^2 = 9$. In this situation we have a grid of nodes rather than a sequence. This is illustrated in figure 2.4.5.

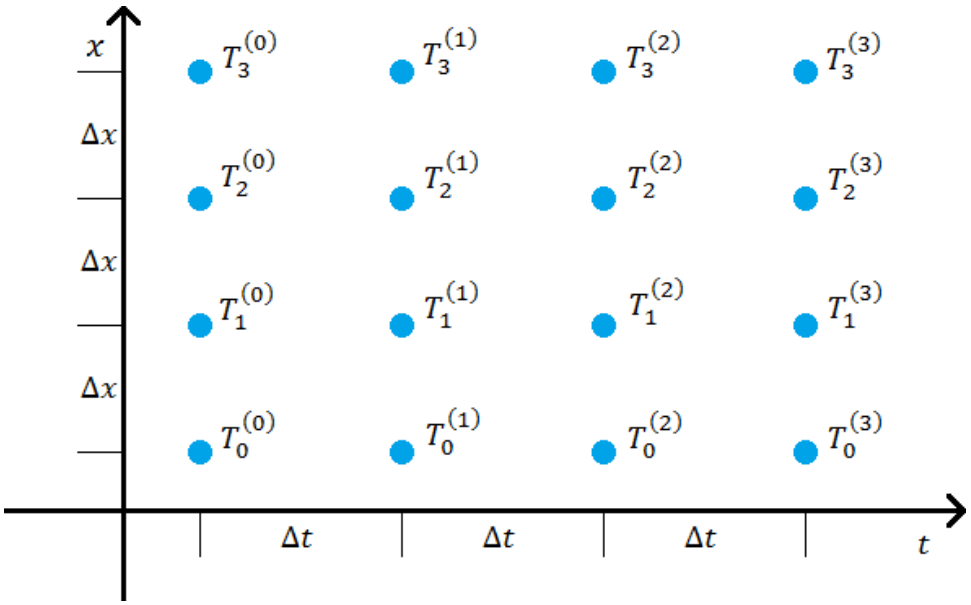

Fig 2.4.5 – A grid of nodes. Note that the vertical location of the nodes does not indicate their value as in fig 2.4.2. Another perpendicular axis would be needed for that, which would go into or out from the page.

2.5 | The Trinomial Method

The idea behind the trinomial method is a heuristic approach to option pricing. Recall equation 2.2.1:

$$\frac{dS}{S} = \mu dt + \sigma dX. \qquad \text{- Eq 2.2.1}$$

Firstly, μ, the 'drift term' is replaced by the risk-free interest rate r. The rationale is that if the asset were expected to rise more than the interest rate, traders would buy that asset right now, raising the price now until the expectation of the future rise was in line with the interest rate. Likewise, if it were expected to rise less than the interest rate, it would be sold until it was similarly balanced. I'm keen to get rid of this term straight away because unfortunately, μ is mixed up with a different meaning in convention, and will be used later on with that other meaning.

$$\frac{dS}{S} = rdt + \sigma dX \qquad \text{- Eq 2.5.1}$$

We want a discrete version of this equation, so we change the derivatives to difference operators, and simplify:

$$\frac{\Delta S}{S} = r\Delta t + \sigma \Delta X$$

$$\frac{S_m^{(n+1)} - S_m^{(n)}}{S_m^{(n)}} = r\Delta t + \sigma \Delta X$$

$$S_m^{(n+1)} = S_m^{(n)}(1 + r\Delta t + \sigma \Delta X). \qquad \text{- Eq 2.5.2}$$

In the equation above, ΔX is normally distributed (mean 0, variance Δt), which means the next asset price is potentially anywhere in $(-\infty, \infty)$*. In the trinomial method, we approximate $(1 + r\Delta t + \sigma \Delta X)$ with a discrete distribution over 3 values with the same mean and variance within errors $\mathcal{O}(\Delta t^{1.5})$. These parameters are the multipliers U, M, and D, with corresponding probabilities P_U, P_M, P_D. This is illustrated in figure 2.5.1.

*It may seem troubling that a non-zero probability is placed on $S_m^{(n+1)} < 0$. However, in the limit $\Delta t \to 0$, this probability converges to zero.

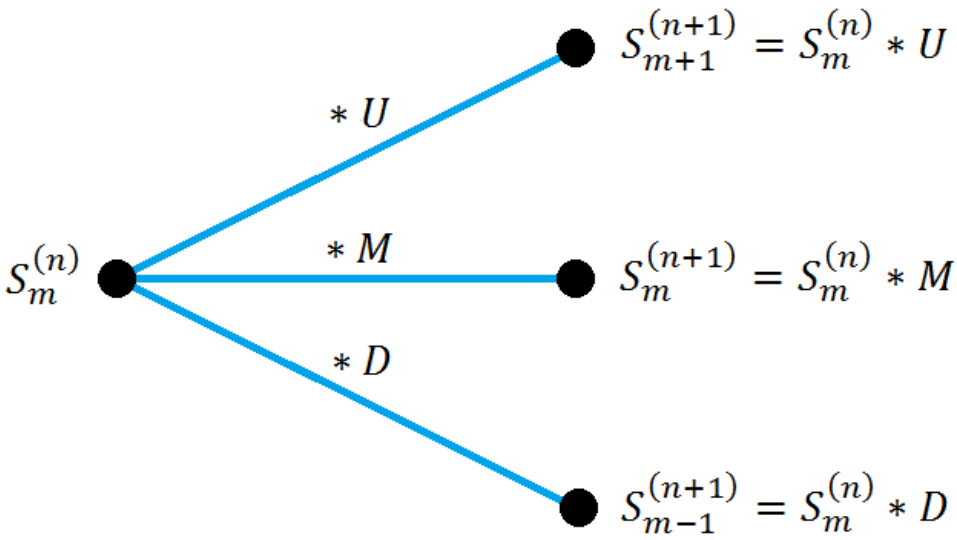

Fig 2.5.1 – $S_m^{(n)}$ and its child nodes in a trinomial tree.

These relationships can be described by a general formula; equation 2.5.3:

$$S_m^{(n)} = U^m M^{n-m} S_0^{(0)}. \qquad \text{- Eq 2.5.3}$$

With this established, each node is associated with the option value $V_m^{(n)}$ for that time and asset (underlying) value. We reason that each option price $V_m^{(n)}$ should be its expected value at the next time step discounted to the current time step. Hence:

$$V_m^{(n)} = e^{-r\Delta t}\left(P_U V_{m+1}^{(n+1)} + P_M V_m^{(n+1)} + P_D V_{m-1}^{(n+1)}\right)$$

$$e^{r\Delta t} V_m^{(n)} = P_U V_{m+1}^{(n+1)} + P_M V_m^{(n+1)} + P_D V_{m-1}^{(n+1)}. \qquad \text{- Eq 2.5.4}$$

We put many of these time steps together to form a trinomial tree, as illustrated by figure 2.5.2.

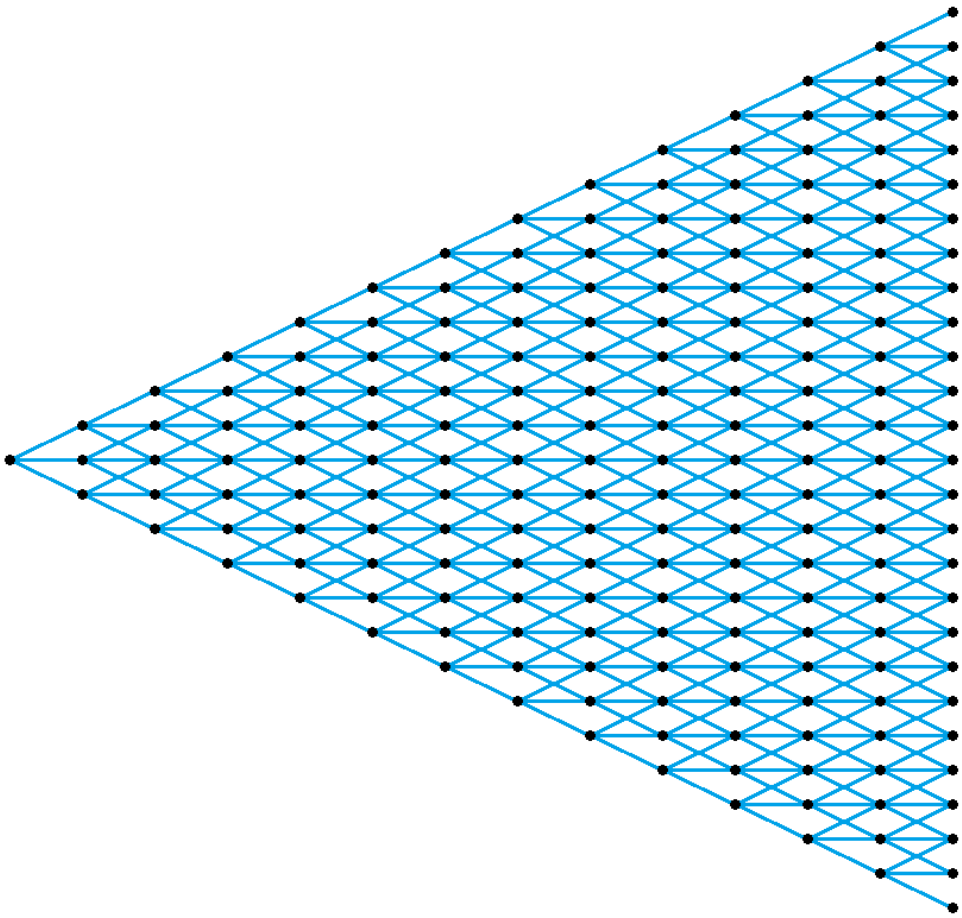

Fig 2.5.2 – A trinomial tree.

Note the recombination of the nodes; going up and down arrives at the same places as going along the middle path twice. Without this, each time step would have triple the nodes of the previous time step, and the method would be vastly less efficient. Enforcing re-combination means equation 2.5.5 must hold. This should clear up any confusion about equation 2.5.3 not involving D.

$$M = \sqrt{UD}. \qquad \text{- Eq 2.5.5}$$

To apply this in pricing an option, the time to expiry, T, is divided into N time steps each of size Δt. At expiry, the option price is simply the payoff function, denoted by $Q(S)$, so we have:

$$V_m^{(N)} = Q\left(S_m^{(n)}\right). \qquad \text{- Eq 2.5.6}$$

Using 2.5.4 and 2.5.6, we calculate the $V_m^{(n)}$ values back through the tree until $V_0^{(0)}$ is calculated.

There is more than one way to parameterise the trinomial method. One of them is the Jarrow-Rudd* parameterisation[4]:

$$U = e^{\mu\Delta t + \sigma\sqrt{2\Delta t}} \qquad P_U = 1/4$$

*I credit Jarrow and Rudd with this parameterisation, but really it is derived from their *binomial* method parameterisation by skipping every second time step.

$$M = e^{\mu\Delta t} \qquad P_M = 1/2$$

$$D = e^{\mu\Delta t - \sigma\sqrt{2\Delta t}} \qquad P_D = 1/4,$$

- Eq 2.5.7-12

where

$$\mu = r - \frac{1}{2}\sigma^2. \qquad \text{- Eq 2.5.13}$$

The fact that μ is $r - (1/2)\sigma^2$ and not r is rather puzzling. Because the middle multiplier M is $e^{\mu\Delta t}$, the nodes drift upward at a rate of μ. This is a bit counter-intuitive, as one might expect that if the nodes are to drift, they should drift upward by the interest rate, not the interest rate perturbed by $-(1/2)\sigma^2$. This is explained in appendix A.1.

Chapter 3

Equivalence

3.1 | First Attempt

Saying "the trinomial method is equivalent to the finite difference method" is incomplete. To see this, let's go from the Black Scholes equation (2.2.2) straight into a finite difference method, and see where it goes wrong.

This is the Black-Scholes equation:

$$\frac{\partial V}{\partial t}+\frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2}+rS\frac{\partial V}{\partial S}-rV=0. \qquad \text{- Eq 2.2.2}$$

The equation is discretised:

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

To get as much likeness as possible, we take $\Delta^2V/\Delta S^2$ and $\Delta V/\Delta S$ from the $(n+1)$th time step, use a central difference for $\Delta V/\Delta S$, and take V in rV from the current time step:

$$\frac{V_m^{(n+1)}-V_m^{(n)}}{\Delta t}+\frac{1}{2}\sigma^2\,(S_m)^2\frac{V_{m+1}^{(n+1)}-2V_m^{(n+1)}+V_{m-1}^{(n+1)}}{\Delta S^2}+rS_m\frac{V_{m+1}^{(n+1)}-V_{m-1}^{(n+1)}}{2\Delta S}-rV_m^{(n)}=0.$$

- Eq 3.1.1

The reason why $\Delta^2V/\Delta S^2$ and $\Delta V/\Delta S$ are from the $(n+1)$th time step is because we want to derive $V_m^{(n)}$ from information in the $(n+1)$th time step. We could achieve the same thing by reversing time with $\tau=-t$ and derive the scheme more conventionally, but by doing that we could easily lose sight of what's actually going on – deriving $V_m^{(n)}$ from $V_{m+1}^{(n)}$, $V_m^{(n+1)}$, $V_{m-1}^{(n+1)}$, which is crucial to finding equivalence with the trinomial method.

Equation 3.1.1 can be rearranged into equation 3.1.2.

$$\begin{aligned}(1+r\Delta t)V_m^{(n)}=&\left(\frac{\sigma^2(S_m)^2\Delta t}{2\Delta S^2}+\frac{rS_m\Delta t}{2\Delta S}\right)V_{m+1}^{(n+1)}+\left(1-\frac{\sigma^2\,(S_m)^2\Delta t}{\Delta S^2}\right)V_m^{(n+1)}\\&+\left(\frac{\sigma^2(S_m)^2\Delta t}{2\Delta S^2}-\frac{rS_m\Delta t}{2\Delta S}\right)V_{m-1}^{(n+1)}\end{aligned}$$

- Eq 3.1.2

Equation 3.1.2 shows a lot of promise towards equivalence with the trinomial method. Note the similarity with equation 2.5.4:

$$e^{r\Delta t}V_m^{(n)} = P_U V_{m+1}^{(n+1)} + P_M V_m^{(n+1)} + P_D V_{m-1}^{(n+1)}. \quad \text{- Eq 2.5.4}$$

In both cases, $V_m^{(n)}$ is a linear combination of $V_{m+1}^{(n)}$, $V_m^{(n+1)}$, and $V_{m-1}^{(n+1)}$. This means calculating $V_0^{(0)}$ involves a triangular region of influence as illustrated in figure 3.1.1.

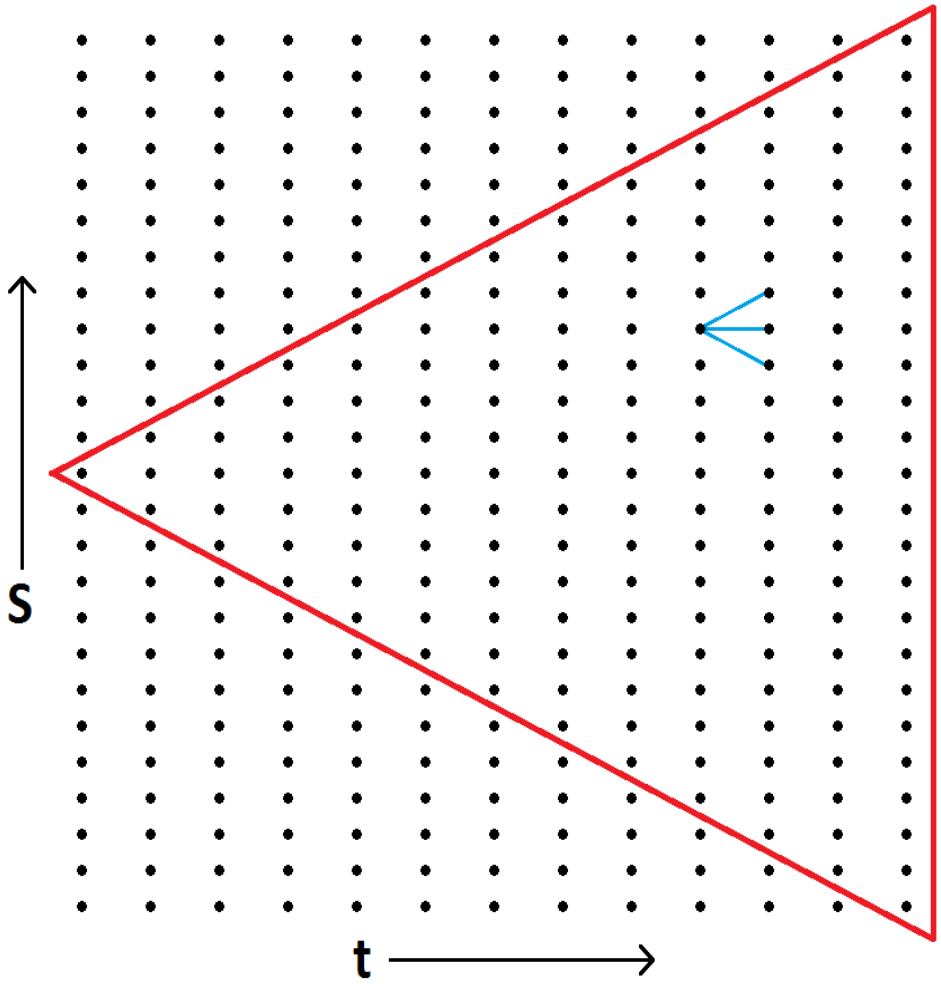

Fig 3.1.1 – The triangular region of influence for the finite difference method.

The difference between this triangular region of influence and the tree structure in the trinomial method is only cosmetic. In both cases each node value is derived from the values of the right, upper right, and lower right nodes. However, there are a couple of numerical differences.

The first difference, $1 + r\Delta t$ vs. $e^{r\Delta t}$ is actually acceptable, since:

$$e^{r\Delta t} = 1 + r\Delta t + \frac{1}{2}r^2\Delta t^2 + \frac{1}{6}r^3\Delta t^3 + \cdots = 1 + r\Delta t + \mathcal{O}(\Delta t^2)$$

This is why V in rV was chosen to be $V_m^{(n)}$ instead of $V_m^{(n+1)}$ which would be more conventional. This choice creates the $(1 + r\Delta t)$ term, which would otherwise change the coefficient of $V_m^{(n+1)}$.

The critical difference is that the coefficients on the right hand side of 3.1.2 are not constant, as they depend on S_m. This could be avoided by allowing ΔS to vary and instead be ΔS_m, such that $S_m/\Delta S_m$ were constant. In fact, doing this in the right way could make for a scheme extremely similar to the

trinomial method. However, variable steps are not desirable to work with and the coefficients couldn't be made exactly the same.

Indeed (using a constant ΔS), the coefficients have to be different because the nodes are not structured in the same way as the trinomial method. This is illustrated in figure 3.1.2.

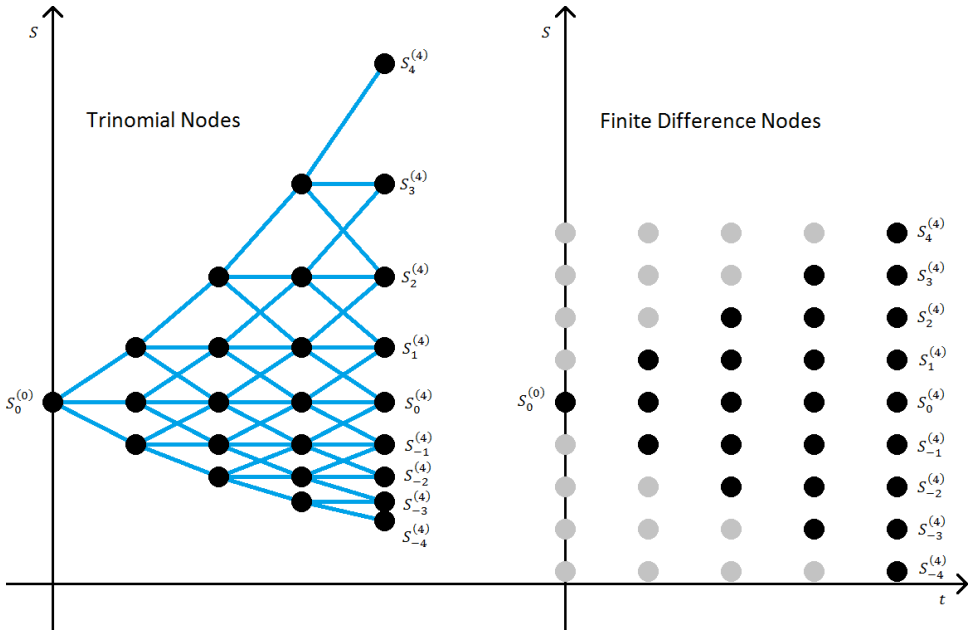

Fig 3.1.2 – The structures of Trinomial and Finite Difference Nodes.

If the finite difference method is to be equivalent to the trinomial method, we'll need the nodes to match trinomial nodes. Clearly, the trinomial method nodes are not spaced evenly along the S axis, but as mentioned we can't get equivalence by simply using variable ΔS steps. The way we get around this is to scale the S axis by performing a substitution, so that an even spacing along the new axis gives us the correct variable spacing along the S axis (see section 2.3).

3.2 | Equivalence under Kamrad-Ritchken Parameters

If we first perform a substitution

$$x = \log S, \qquad \text{- Eq 3.2.1}$$

and otherwise derive the finite difference method in the same way as in section 3.1, the method is equivalent to the trinomial method under Kamrad-Ritchken[5] parameters[*]:

$$U = e^{\lambda\sigma\sqrt{\Delta t}} \qquad P_U = \frac{1}{2\lambda^2} + \frac{\mu\sqrt{\Delta t}}{2\lambda\sigma}$$

$$M = 1 \qquad P_M = 1 - \frac{1}{\lambda^2}$$

$$D = e^{-\lambda\sigma\sqrt{\Delta t}} \qquad P_D = \frac{1}{2\lambda^2} + \frac{\mu\sqrt{\Delta t}}{2\lambda\sigma}$$

- Eq 3.2.2-7

where

$$\mu = r - \frac{1}{2}\sigma^2. \qquad \text{- Eq 2.5.13}$$

We start with the Black-Scholes equation (2.2.2):

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0. \qquad \text{- Eq 2.2.2}$$

In appendix A.2 this equation is transformed by $x = \log S$, and a finite difference scheme derived in the same way as in section 3.1 to produce equation 3.2.8:

$$(1 + r\Delta t)V_m^{(n)} = \left(\frac{\sigma^2\Delta t}{2\Delta x^2} + \frac{\mu\Delta t}{2\Delta x}\right)V_{m+1}^{(n+1)} + \left(1 - \frac{\sigma^2\Delta t}{\Delta x^2}\right)V_m^{(n+1)} + \left(\frac{\sigma^2\Delta t}{2\Delta x^2} - \frac{\mu\Delta t}{2\Delta x}\right)V_{m-1}^{(n+1)}.$$

- Eq 3.2.8

If we place these nodes on the S, t plane, we want them to exactly match the locations of the trinomial method nodes. For this to happen, we choose Δx to recreate the spacing in the trinomial method. We derive Δx loosely here and show later that the nodes actually match.

In the Kamrad-Ritchken parameterisation, the S values at the nodes are constant across time, so we have that:

$$S_{m+1}^{(n)} = US_m^{(n)}.$$

Substituting equation 3.2.2, we have:

[*] This is actually a family or parameterisations as λ is not specified and may be chosen arbitrarily. However, $\lambda = \sqrt{2}$ is thought to give good results[1].

$$S_{m+1}^{(n)} = e^{\lambda\sigma\sqrt{\Delta t}} S_m^{(n)}$$

$$\log S_{m+1}^{(n)} = \log\left(e^{\lambda\sigma\sqrt{\Delta t}} S_m^{(n)}\right)$$

$$\log S_{m+1}^{(n)} = \log\left(S_m^{(n)}\right) + \lambda\sigma\sqrt{\Delta t}$$

$$x_{m+1}^{(n)} = x_m^{(n)} + \lambda\sigma\sqrt{\Delta t}$$

$$\Delta x = \lambda\sigma\sqrt{\Delta t}. \qquad \text{- Eq 3.2.9}$$

As mentioned, we will see later how this actually does correctly distribute the nodes in the S, t plane. First, we see that:

$$\frac{\sigma^2\Delta t}{\Delta x^2} = \frac{\sigma^2\Delta t}{\lambda^2\sigma^2\Delta t} = \frac{1}{\lambda^2}$$

$$\frac{\mu\Delta t}{2\Delta x} = \frac{\mu\Delta t}{2\lambda\sigma\sqrt{\Delta t}} = \frac{\mu\sqrt{\Delta t}}{2\lambda\sigma},$$

and substitute them into 3.2.8:

$$(1 + r\Delta t)V_m^{(n)} = \left(\frac{1}{2\lambda^2} + \frac{\mu\sqrt{\Delta t}}{2\lambda\sigma}\right)V_{m+1}^{(n+1)} + \left(1 - \frac{1}{\lambda^2}\right)V_m^{(n+1)} + \left(\frac{1}{2\lambda^2} - \frac{\mu\sqrt{\Delta t}}{2\lambda\sigma}\right)V_{m-1}^{(n+1)}.$$

- Eq 3.2.10

If we substitute the Kamrad-Ritchken parameters (equations 3.2.2-7) into the trinomial method equation relating node values together (equation 2.5.4), we get:

$$e^{r\Delta t}V_m^{(n)} = \left(\frac{1}{2\lambda^2} + \frac{\mu\sqrt{\Delta t}}{2\lambda\sigma}\right)V_{m+1}^{(n+1)} + \left(1 - \frac{1}{\lambda^2}\right)V_m^{(n+1)} + \left(\frac{1}{2\lambda^2} - \frac{\mu\sqrt{\Delta t}}{2\lambda\sigma}\right)V_{m-1}^{(n+1)}.$$

- Eq 3.2.11

The only difference here is $(1 + r\Delta t)$ vs. $e^{r\Delta t}$, which, as mentioned in section 3.1, only differ by $\mathcal{O}(\Delta t^2)$. This is a minor caveat in the equivalence between the two methods.

From here all that's left is to show that the nodes really are structured in the same way. In the finite difference method:

$$x_{m+1}^{(n)} = x_m^{(n)} + \Delta x$$

$$x_m^{(n)} = x_0^{(n)} + m\Delta x.$$

Equation 3.2.1 gives $x = \log S$, so we can put these nodes on the S axis using $e^{x_m^{(n)}} = S_m^{(n)}$:

$$e^{x_m^{(n)}} = e^{x_0^{(n)} + m\Delta x}$$

$$S_m^{(n)} = S_0^{(n)}\left(e^{\Delta x}\right)^m.$$

Now substitute 3.2.9:

$$S_m^{(n)} = S_0^{(n)} \left(e^{\lambda\sigma\sqrt{\Delta t}} \right)^m,$$

use 3.2.2 to substitute U for $e^{\lambda\sigma\sqrt{\Delta t}}$:

$$S_m^{(n)} = U^m S_0^{(n)}.$$

In the Kamrad-Ritchken parameterisation, $M = 1$, so we can multiply the RHS by M without losing equality:

$$S_m^{(n)} = U^m M^{n-m} S_0^{(n)}. \quad \text{- Eq 3.2.12}$$

Equation 3.2.12 is the same as 2.5.3. Therefore the nodes are distributed the same way along the S axis. As for distribution along the t axis, if the nodes start at $t = 0$, and the same time step Δt is used, then the nodes will be placed the same along this axis as well.

Therefore, as the nodes are located in the same places on the S, t plane, the same payoff (final condition) can be applied to the $V_m^{(N)}$ values, and the same $V_0^{(0)}$ will be derived. (Except for $(1 + r\Delta t)$ vs. $e^{r\Delta t}$.)

More conventionally, what "should" be done, is for the final condition to be transformed by $x = \log S$, that is:

$$V(S, T) = Q(S)$$

$$V(x, T) = Q(e^x), \quad \text{- Eq 3.2.13}$$

and discretised:

$$V_m^{(N)} = Q\left(e^{x_m^{(N)}} \right)$$

$$V_m^{(N)} = Q\left(S_m^{(N)} \right). \quad \text{- Eq 3.2.14}$$

This is the same final condition as the trinomial method's final condition (equation 2.5.6).

3.3 | A Problem with Kamrad-Ritchken Parameters

There's something unsettling about the Kamrad-Ritchken parameterisation. The probability of going up is greater than the probability of going down, and intuitively it seems this is to compensate for the upward drift of asset prices. This is illustrated in figure 3.3.1.

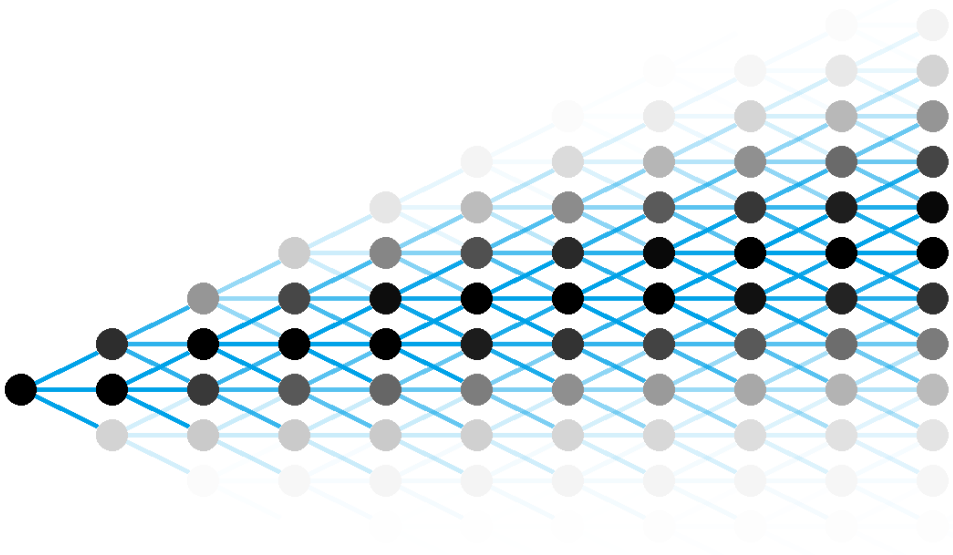

Fig 3.3.1 – A trinomial tree with node darkness in each time slice proportional to that node's weight using Kamrad-Ritchken parameters.

The greater probability of going up results in an upward drift of the distribution of node weights. By node weights, I mean the coefficients used when expressing $V_0^{(0)}$ as a linear combination of the node values in that time slice. This is the probability of reaching that node.

By allowing the distribution to drift upward, there is much less weight in the nodes of the bottom half of the tree, yet, they consume the same computational resources.

For these reasons, it is more conceptually satisfying, and perhaps more efficient, to instead allow the *nodes* to drift upward. This reflects the upward drift of asset prices, and results in a symmetrical distribution of node weights. This is provided by the Jarrow-Rudd parameterisation.

3.4 | Equivalence under Jarrow-Rudd Parameters

Recall the Jarrow-Rudd parameterisation from section 2.5:

$$U = e^{\mu\Delta t + \sigma\sqrt{2\Delta t}} \qquad P_U = 1/4$$

$$M = e^{\mu\Delta t} \qquad P_M = 1/2$$

$$D = e^{\mu\Delta t - \sigma\sqrt{2\Delta t}} \qquad P_D = 1/4,$$

- Eq 2.5.7-12

where

$$\mu = r - \frac{1}{2}\sigma^2. \qquad \text{- Eq 2.5.13}$$

As discussed in section 3.3, using these parameters means the nodes drift upwards. The middle multiplier is $e^{\mu\Delta t}$ and not 1. To construct a finite difference method equivalent to the trinomial method under this parameterisation, we need a transform that will structure the nodes with this upward drift. The transform is:

$$x = \log(Se^{-\mu t}). \qquad \text{- Eq 3.4.1}$$

Recall the Black-Scholes equation (2.2.2):

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0. \qquad \text{- Eq 2.2.2}$$

Appendix A.3 applies the transform to the Black-Scholes equation and a finite difference scheme is derived in the same way as in section 3.1 to produce equation 3.4.2:

$$(1 + r\Delta t)V_m^{(n)} = \frac{\sigma^2\Delta t}{2\Delta x^2}V_{m+1}^{(n+1)} + \left(1 - \frac{\sigma^2\Delta t}{\Delta x^2}\right)V_m^{(n+1)} + \frac{\sigma^2\Delta t}{2\Delta x^2}V_{m-1}^{(n+1)}.$$

- Eq 3.4.2

Just like in section 3.1, we derive Δx loosely with the intention of recreating the spacing between the nodes in the trinomial method. From any node $S_m^{(n)}$, we get to the node above it by multiplying by U and dividing by M, as illustrated by figure 3.4.1.

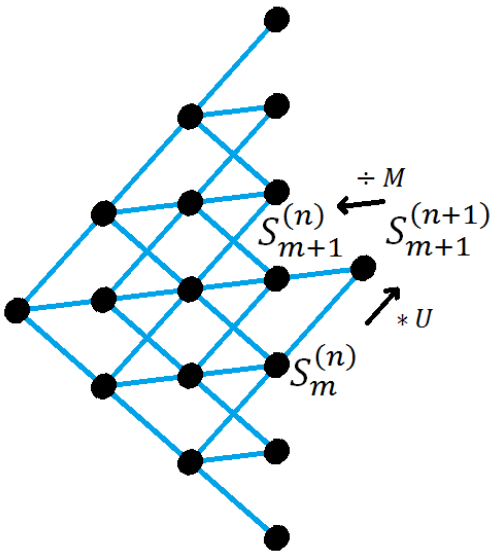

Fig 3.4.1 – The connection between $S_m^{(n)}$ and $S_{m+1}^{(n)}$.

So, we have:

$$S_{m+1}^{(n)} = \frac{U}{M} * S_m^{(n)}, \qquad \text{- Eq 3.4.3}$$

substitute 2.5.7, 2.5.9:

$$S_{m+1}^{(n)} = \frac{e^{\mu\Delta t+\sigma\sqrt{2\Delta t}}}{e^{\mu\Delta t}} * S_m^{(n)}$$

$$S_{m+1}^{(n)} = S_m^{(n)} e^{\sigma\sqrt{2\Delta t}}$$

$$S_{m+1}^{(n)} e^{-\mu n\Delta t} = S_m^{(n)} e^{-\mu n\Delta t} e^{\sigma\sqrt{2\Delta t}}$$

$$\log\left(S_{m+1}^{(n)} e^{-\mu n\Delta t}\right) = \log\left(S_m^{(n)} e^{-\mu n\Delta t} e^{\sigma\sqrt{2\Delta t}}\right)$$

$$\log\left(S_{m+1}^{(n)} e^{-\mu n\Delta t}\right) = \log\left(S_m^{(n)} e^{-\mu n\Delta t}\right) + \sigma\sqrt{2\Delta t}$$

$$x_{m+1}^{(n)} = x_m^{(n)} + \sigma\sqrt{2\Delta t}$$

$$\Delta x = \sigma\sqrt{2\Delta t}.^{*} \qquad \text{- Eq 3.4.4}$$

We will see later how this actually does correctly distribute the nodes in the S, t plane. First:

$$\frac{\sigma^2\Delta t}{\Delta x^2} = \frac{\sigma^2\Delta t}{\sigma^2 * 2\Delta t} = \frac{1}{2},$$

substitute this into 3.4.2:

---

[*] Note that this is compatible with the Kamrad-Ritchken value of Δx with $\lambda = \sqrt{2}$. By using the Kamrad-Ritchken Δx instead, leaving λ as an arbitrary parameter, the Jarrow-Rudd trinomial method can be extended to allow independent variation of the time and S node spacing.

$$(1 + r\Delta t)V_m^{(n)} = \frac{1}{4}V_{m+1}^{(n+1)} + \frac{1}{2}V_m^{(n+1)} + \frac{1}{4}V_{m-1}^{(n+1)}. \quad \text{- Eq 3.4.5}$$

If we substitute the Jarrow-Rudd parameters (equations 2.5.7-12) into the trinomial method equation relating node values together (equation 2.5.4), we get:

$$e^{r\Delta t}V_m^{(n)} = \frac{1}{4}V_{m+1}^{(n+1)} + \frac{1}{2}V_m^{(n+1)} + \frac{1}{4}V_{m-1}^{(n+1)}. \quad \text{- Eq 3.4.6}$$

Once again, the only difference is $(1 + r\Delta t)$ vs. $e^{r\Delta t}$, which differ negligibly by $\mathcal{O}(\Delta t^2)$.

The only thing left is to confirm that our choice of Δx really does result in the same node placement as the trinomial method:

$$\frac{S_m^{(n+1)}}{S_m^{(n)}} = \frac{e^{x+\mu(n+1)\Delta t}}{e^{x+\mu n\Delta t}}$$

$$S_m^{(n+1)} = e^{\mu\Delta t}S_m^{(n)}$$

$$S_m^{(n)} = e^{\mu n\Delta t}S_m^{(0)} \quad \text{- Eq 3.4.7}$$

$$x_{m+1}^{(n)} = x_m^{(n)} + \Delta x,$$

substitute 3.4.4:

$$x_{m+1}^{(n)} = x_m^{(n)} + \sigma\sqrt{2\Delta t}$$

$$x_{m+1}^{(n)} - \mu n\Delta t = x_m^{(n)} - \mu n\Delta t + \sigma\sqrt{2\Delta t}$$

$$e^{x_{m+1}^{(n)} - \mu n\Delta t} = e^{x_m^{(n)} - \mu n\Delta t + \sigma\sqrt{2\Delta t}}$$

$$e^{x_{m+1}^{(n)} - \mu n\Delta t} = e^{\sigma\sqrt{2\Delta t}}e^{x_m^{(n)} - \mu n\Delta t}$$

$$S_{m+1}^{(n)} = e^{\sigma\sqrt{2\Delta t}}S_m^{(n)}$$

$$S_m^{(n)} = e^{m\sigma\sqrt{2\Delta t}}S_0^{(n)}$$

$$S_m^{(0)} = e^{m\sigma\sqrt{2\Delta t}}S_0^{(0)},$$

substitute this into 3.4.7:

$$S_m^{(n)} = e^{\mu n\Delta t}e^{m\sigma\sqrt{2\Delta t}}S_0^{(0)}$$

$$S_m^{(n)} = e^{n\mu\Delta t + m\sigma\sqrt{2\Delta t}}S_0^{(0)}$$

$$S_m^{(n)} = e^{(\mu\Delta t + \sigma\sqrt{2\Delta t})m + \mu\Delta t(n-m)}S_0^{(0)}$$

$$S_m^{(n)} = U^m M^{n-m}S_0^{(0)}. \quad \text{- Eq 3.4.8}$$

Equation 3.4.8 is identical to 2.5.3. Therefore, the node placement is the same in the two methods. This means the same final values will be applied to $V_m^{(N)}$ and the same value for $V_0^{(0)}$ will be calculated, except for the difference due to $(1 + r\Delta t)$ vs $e^{r\Delta t}$.

Just like in section 3.2, convention would have us instead transform the final condition (payoff function) and discretise it in order to show the $V_m^{(n)}$ values are the same:

$$V(S,T) = Q(S)$$

$$V(x,T) = Q(e^{x+\mu t})$$

$$V_m^{(N)} = Q\left(e^{x_m^{(N)}+\mu t}\right)$$

$$V_m^{(N)} = Q\left(S_m^{(N)}\right). \qquad \text{- Eq 3.4.9}$$

This is indeed the same as the final condition for the trinomial method (2.5.6).

3.5 | Perfect Equivalence

In the equivalence of finite difference methods with both the Kamrad-Ritchken and Jarrow-Rudd trinomial methods, there has been the caveat of $(1 + r\Delta t)$ vs. $e^{r\Delta t}$. Rubinstein (2000)[1] notes this caveat and describes its negligibility with a numerical example; that with $\Delta t = 0.001$ and $r = \log(1.1)$, we have $1 + r\Delta t \cong 1.000095310$ and $e^{r\Delta t} \cong 1.000095315$.

However, as it turns out, if we transform A.3.8 (derived from the Black-Scholes equation) one step further, it is not necessary to admit this imperfection[*].

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 \frac{\partial^2 V}{\partial x^2} - rV = 0 \qquad \text{- Eq A.3.8}$$

The transform is:

$$U = Ve^{-rt} \qquad \text{- Eq 3.5.1}$$

$$V = Ue^{rt},$$

from which we derive:

$$\frac{\partial V}{\partial t} = \frac{\partial U}{\partial t}e^{rt} + rUe^{rt} = \frac{\partial U}{\partial t}e^{rt} + rV$$

$$\frac{\partial^2 V}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial V}{\partial x}\right) = \frac{\partial}{\partial x}\left(\frac{\partial U}{\partial x}e^{rt}\right) = \frac{\partial^2 U}{\partial x^2}e^{rt}.$$

Substitute these into A.3.8:

$$\frac{\partial U}{\partial t}e^{rt} + rV + \frac{1}{2}\sigma^2 \frac{\partial^2 U}{\partial x^2}e^{rt} - rV = 0$$

$$\frac{\partial U}{\partial t}e^{rt} + \frac{1}{2}\sigma^2 \frac{\partial^2 U}{\partial x^2}e^{rt} = 0$$

$$\frac{\partial U}{\partial t} + \frac{1}{2}\sigma^2 \frac{\partial^2 U}{\partial x^2} = 0. \qquad \text{- Eq 3.5.2}$$

It is rather satisfying to see that this journey to equivalence has culminated in transforming the Black-Scholes equation into the heat equation ($\kappa = -(1/2)\sigma^2$). Also, by removing rV, this eliminates the inconsistency of choosing $rV_m^{(n)}$ instead of $rV_m^{(n+1)}$, and we can discretise in the conventional backward-Euler fashion[†]:

$$\frac{\Delta U}{\Delta t} + \frac{1}{2}\sigma^2 \frac{\Delta^2 U}{\Delta x^2} = 0$$

[*] If you apply the same transform to A.2.6 and follow the same steps, the same perfect equivalence can be found with the Kamrad-Ritchken trinomial method.

[†] Usually, backward-Euler discretisation results in an *implicit* scheme as it generates multiple terms from the $(n+1)$th time step. However, the resulting scheme is *explicit* because we are deriving nth step terms from $(n+1)$th step terms and not the other way around.

$$\frac{U_m^{(n+1)} - U_m^{(n)}}{\Delta t} + \frac{\sigma^2}{2\Delta x^2}\left(U_{m+1}^{(n+1)} - 2U_m^{(n+1)} + U_{m-1}^{(n+1)}\right) = 0$$

$$U_m^{(n)} = \frac{\sigma^2 \Delta t}{2\Delta x^2} U_{m+1}^{(n+1)} + \left(1 - \frac{\sigma^2 \Delta t}{\Delta x^2}\right) U_m^{(n+1)} + \frac{\sigma^2 \Delta t}{2\Delta x^2} U_{m-1}^{(n+1)}. \quad \text{- Eq 3.5.3}$$

Since the nodes are still distributed as they are in section 3.4 we again take $\Delta x = \sigma\sqrt{2\Delta t}$ (3.4.4):

$$\frac{\sigma^2 \Delta t}{\Delta x^2} = \frac{\sigma^2 \Delta t}{\sigma^2 2\Delta t} = \frac{1}{2},$$

substitute this into 3.5.3:

$$U_m^{(n)} = \frac{1}{4} U_{m+1}^{(n+1)} + \frac{1}{2} U_m^{(n+1)} + \frac{1}{4} U_{m-1}^{(n+1)}. \quad \text{- Eq 3.5.3}$$

At this point we do something a bit different; undo the transformation now that discretisation has been applied by substituting $U_m^{(n)} = V_m^{(n)} e^{-rn\Delta t}$ into 3.5.3:

$$V_m^{(n)} e^{-rn\Delta t} = \frac{1}{4} V_{m+1}^{(n)} e^{-r(n+1)\Delta t} + \frac{1}{2} V_m^{(n)} e^{-r(n+1)\Delta t} + \frac{1}{4} V_{m-1}^{(n)} e^{-r(n+1)\Delta t}$$

multiply both sides by $e^{r(n+1)\Delta t}$:

$$e^{r\Delta t} V_m^{(n)} = \frac{1}{4} V_{m+1}^{(n)} + \frac{1}{2} V_m^{(n)} + \frac{1}{4} V_{m-1}^{(n)}. \quad \text{- Eq 3.5.4}$$

This is not just nearly the same but *identical* to 3.4.6, the relation between the nodes values in the trinomial method under Jarrow-Rudd parameters. Further, we can once again argue that the final condition is the same because the locations of $V_m^{(N)}$ in the S, t plane are the same in both methods. However, for convention's sake, we transform $V(x,T) = Q(e^{x+\mu t})$ from section 3.4 to:

$$U(x,T) = V(x,T)e^{-rT} = Q(e^{x+\mu T})e^{-rT},$$

discretise:

$$U_m^{(N)} = Q\left(e^{x_m^{(N)} + \mu T}\right) e^{-rN\Delta t},$$

and undo the transformation as before using $U_m^{(n)} = V_m^{(n)} e^{-rn\Delta t}$:

$$V_m^{(N)} e^{-rN\Delta t} = Q\left(e^{x_m^{(N)} + \mu N\Delta t}\right) e^{-rN\Delta t}$$

$$V_m^{(N)} = Q\left(e^{x_m^{(N)} + \mu N\Delta t}\right)$$

$$V_m^{(N)} = Q\left(S_m^{(N)}\right). \quad \text{- Eq 3.5.4}$$

This too is identical to the trinomial method, so perfect equivalence has been achieved.

3.6 | Binomial Equivalence

Binomial methods are equivalent to special cases of trinomial methods with an extra step between each trinomial time step. This is illustrated in figure 3.6.1.

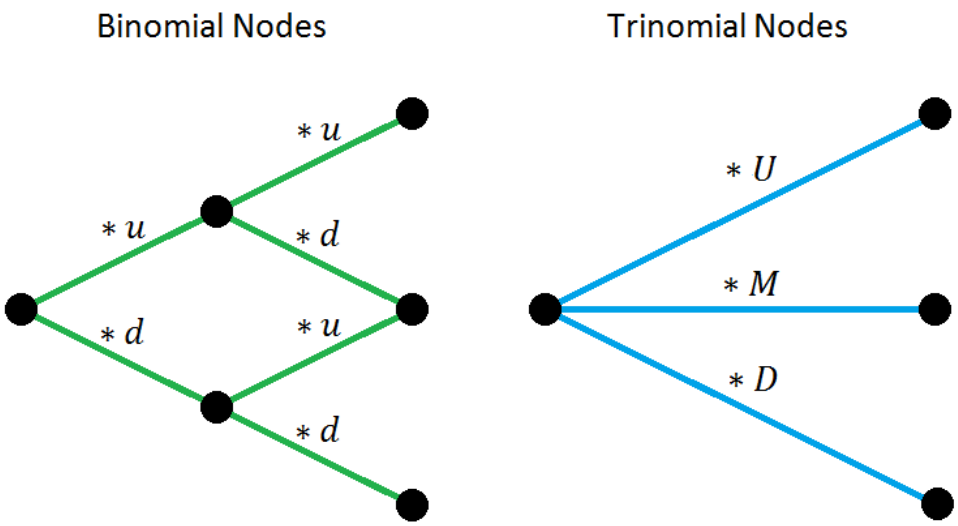

Fig 3.6.1 – The compatible structure of binomial and trinomial nodes.

Intuitively, we see that a binomial method with an even number of time steps $2N$ should be equivalent to a trinomial method of N time steps with parameters:

$$U = u^2 \qquad P_U = p_u^2$$

$$M = ud \qquad P_M = 2p_u p_d$$

$$D = d^2 \qquad P_D = p_d^2.$$

- Eq 3.6.1-6

We will demonstrate this by using $s_m^{(n)}$ and $v_m^{(n)}$ for the asset and option prices in the binomial method. Multiplying by u to go up and by d to go down produces the following relations:

$$s_{m+1}^{(n+1)} = us_m^{(n)} \quad \text{- Eq 3.6.7}$$

$$s_{m-1}^{(n+1)} = ds_m^{(n)}. \quad \text{- Eq 3.6.8}$$

Note that this arrangement excludes some s coordinates from corresponding to actual nodes, as for example the original $s_0^{(0)}$ leads to $s_1^{(1)}$ and $s_{-1}^{(1)}$ and there is no $s_0^{(1)}$. These relations are satisfied by this general formula for $s_m^{(n)}$:

$$s_m^{(n)} = u^{\frac{1}{2}(n+m)} d^{\frac{1}{2}(n-m)}. \quad \text{- Eq 3.6.9}$$

Using this we can show a correspondence between the binomial and trinomial nodes if they have the same starting node:

$$s_0^{(0)} = S_0^{(0)}. \quad \text{- Eq 3.6.10}$$

Recall equation 2.5.3:

$$S_m^{(n)} = U^m M^{n-m} S_0^{(0)}, \quad \text{- Eq 2.5.3}$$

substitute 3.6.1, 3.6.3:

$$S_m^{(n)} = u^{2m}(ud)^{n-m} s_0^{(0)} = u^{n+m} d^{n-m} s_0^{(0)},$$

substitute 3.6.9:

$$S_m^{(n)} = s_{2m}^{(2n)}. \quad \text{- Eq 3.6.11}$$

Naturally, the option values for the final nodes in the binomial method are set to the payoff function, just like in the trinomial method, hence:

$$v_{2m}^{(2N)} = Q\left(s_{2m}^{(2N)}\right),$$

substitute 3.6.11:

$$v_{2m}^{(2N)} = Q\left(S_m^{(n)}\right),$$

substitute 2.5.6:

$$v_{2m}^{(2N)} = V_m^{(N)}. \quad \text{- Eq 3.6.13}$$

The binomial method has the same heuristic feature that the option value at each node is the sum of the option values in the derived nodes each multiplied by the probability of travelling to that node, discounted by the time step, so we have:

$$e^{\frac{1}{2}r\Delta t} v_m^{(n)} = p_u v_{m+1}^{(n+1)} + p_d v_{m-1}^{(n+1)}. \quad \text{- Eq 3.6.14}$$

Manipulating the m, n values of 3.6.14 produces the following variants:

$$e^{\frac{1}{2}r\Delta t} v_{2m}^{(2n)} = p_u v_{2m+1}^{(2n+1)} + p_d v_{2m-1}^{(2n+1)} \quad \text{- Eq 3.6.15}$$

$$e^{\frac{1}{2}r\Delta t} v_{2m+1}^{(2n+1)} = p_u v_{2m+2}^{(2n+2)} + p_d v_{2m}^{(2n+2)} \quad \text{- Eq 3.6.16}$$

$$e^{\frac{1}{2}r\Delta t} v_{2m-1}^{(2n+1)} = p_u v_{2m}^{(2n+2)} + p_d v_{2m-2}^{(2n+2)}. \quad \text{- Eq 3.6.17}$$

Multiply both sides of 3.6.15 by $e^{\frac{1}{2}r\Delta t}$:

$$e^{r\Delta t} v_{2m}^{(2n)} = p_u \left(e^{\frac{1}{2}r\Delta t} v_{2m+1}^{(2n+1)}\right) + p_d \left(e^{\frac{1}{2}r\Delta t} v_{2m-1}^{(2n+1)}\right),$$

substitute 3.6.16, 3.6.17:

$$e^{r\Delta t}v_{2m}^{(2n)} = p_u\left(p_u v_{2m+2}^{(2n+2)} + p_d v_{2m}^{(2n+2)}\right) + p_d\left(p_u v_{2m}^{(2n+2)} + p_d v_{2m-2}^{(2n+2)}\right)$$

$$e^{r\Delta t}v_{2m}^{(2n)} = p_u^2 v_{2m+2}^{(2n+2)} + 2p_u p_d v_{2m}^{(2n+2)} + p_d^2 v_{2m-2}^{(2n+2)}. \quad \text{- Eq 3.6.18}$$

From here we let R_n be the proposition that $V_m^{(n)} = v_{2m}^{(2n)}$. We already have R_N from 3.6.13. If we have R_{n+1} for some n, this gives:

$$V_{m+1}^{(n+1)} = v_{2m+2}^{(2n+2)} \quad \text{- Eq 3.6.19}$$

$$V_m^{(n+1)} = v_{2m}^{(2n+2)} \quad \text{- Eq 3.6.20}$$

$$V_{m-1}^{(n+1)} = v_{2m-2}^{(2n+2)}, \quad \text{- Eq 3.6.21}$$

substitute 3.6.19-21 into 3.6.18:

$$e^{r\Delta t}v_{2m}^{(2n)} = p_u^2 V_{m+1}^{(n+1)} + 2p_u p_d V_m^{(n+1)} + p_d^2 V_{m-1}^{(n+1)}$$

substitute 3.6.2, 3.6.4, 3.6.6:

$$e^{r\Delta t}v_{2m}^{(2n)} = P_U V_{m+1}^{(n+1)} + P_M V_m^{(n+1)} + P_D V_{m-1}^{(n+1)}, \quad \text{- Eq 3.6.22}$$

Recall 2.5.4:

$$e^{r\Delta t}V_m^{(n)} = P_U V_{m+1}^{(n+1)} + P_M V_m^{(n+1)} + P_D V_{m-1}^{(n+1)}. \quad \text{- Eq 2.5.4}$$

This implies:

$$V_m^{(n)} = v_{2m}^{(2n)}. \quad \text{- Eq 3.6.23}$$

Hence $R_{n+1} \Rightarrow R_n$. Since we have R_N, by mathematical induction, we have R_n for all $0 \leq n \leq N$. In particular, R_0 gives:

$$V_0^{(0)} = v_0^{(0)}. \quad \text{- Eq 3.6.24}$$

Therefore every binomial method has an equivalent trinomial method.

It's quite nice to see that just as this research has revolved around using finite differences in reverse, it has also led to a reverse induction proof.

The Jarrow-Rudd parameterisation is actually a *binomial* parameterisation. Equations 2.5.7-12 presented elsewhere as the Jarrow-Rudd parameterisation is actually derived from the following[*]:

$$u = e^{\mu\Delta t + \sigma\sqrt{\Delta t}} \qquad p_u = 1/2$$

[*]When doing this derivation, be careful not to confuse Δt in the binomial method and Δt in the trinomial method, which are different. I suggest replacing Δt in the binomial parameters with h and then using the fact that the trinomial's Δt is $2h$. You could use Δt_b and Δt_t but notation must always have at least some context-dependence.

$$d = e^{\mu\Delta t - \sigma\sqrt{\Delta t}} \qquad p_d = 1/2.$$

- Eq 3.6.25-28

Also, Rubinstein (2000)[1] shows that there is also an equivalent binomial method for the Kamrad-Ritchken parameterisation with $\lambda = \sqrt{2(1-(1/2)(\mu/\sigma)^2\Delta t}$.

The consequence of all this is, by extension, the finite difference method is also equivalent to the binomial method.

Chapter 4

Variants and Implementation

4.1 | Basic European Options

In a purist approach to implementing the trinomial method, the trinomial tree would actually be constructed. The underlying prices would be calculated by moving forward through the tree, the payoff function applied, and then the option prices would be calculated by moving backwards through the tree. This is horribly inefficient.

Instead, the underlying prices for the final nodes can be calculated directly by substituting $n = N$ into 2.5.3:

$$S_m^{(N)} = U^m M^{N-m} S_0^{(0)}. \qquad \text{- Eq 4.1.1}$$

The optimisation, which is much more dramatic, is not keeping the entire tree in memory. Instead, two arrays (`std::vectors`) are used to store time slices of the tree. At any one time, one array stores the nth time slice, and the $(n-1)$th time slice is calculated into the other array. The nth time slice data is no longer needed, so the first array can then be re-used for the $(n-2)$th time slice, and so on. This is illustrated in figure 4.1.1.

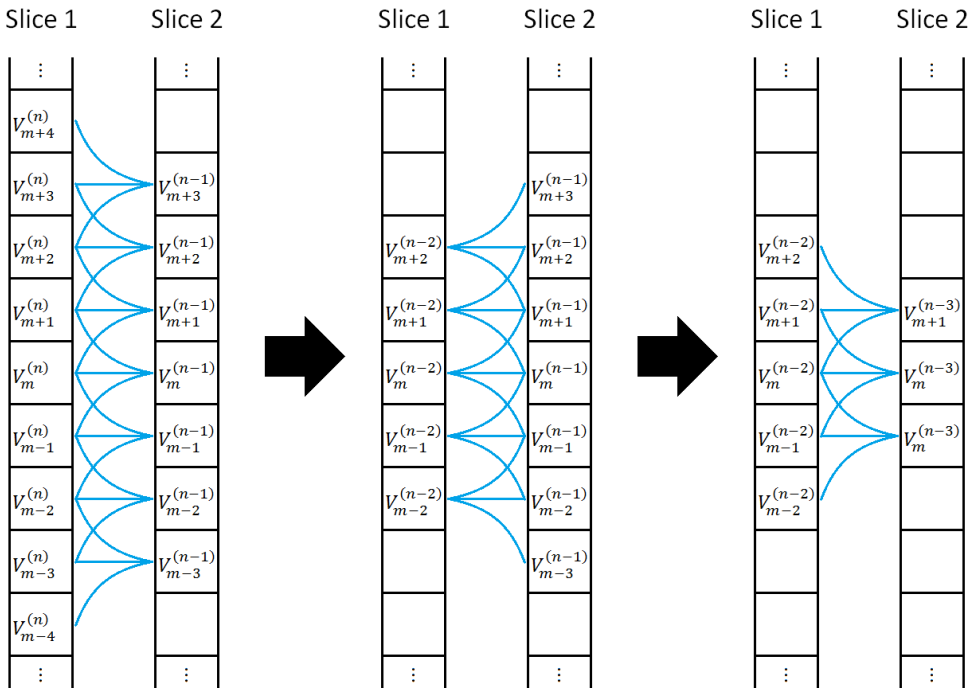

Fig 4.1.1 – Calculation of option values without storing the entire trinomial tree.

Table 4.1.1 shows some sample call calculations for a few strike prices with 256 and 1024 time steps.

$S_0^{(0)}$	K	T	r	σ	Steps	Kam/Rit	Jar/Rud	Analytic	Exec time
\$30	\$40	$1y$	5%	20%	256	\$0.3866	\$0.3871	\$0.3869	$0.346ms$
\$30	\$40	$1y$	5%	20%	1024	\$0.3869	\$0.3868	\$0.3869	$5.379ms$
\$40	\$40	$1y$	5%	20%	256	\$4.1785	\$4.1812	\$4.1802	$0.346ms$
\$40	\$40	$1y$	5%	20%	1024	\$4.1798	\$4.1806	\$4.1802	$5.385ms$
\$50	\$40	$1y$	5%	20%	256	\$12.2947	\$12.2946	\$12.2944	$0.345ms$
\$50	\$40	$1y$	5%	20%	1024	\$12.2944	\$12.2945	\$12.2944	$5.398ms$

Table 4.1.1 – Sample call calculations using the trinomial method.

Table 4.1.2 shows some sample at-the-money put calculations for increasing time steps.

$S_0^{(0)}$	K	T	r	σ	Steps	Kam/Rit	Jar/Rud	Analytic	Exec time
\$20	\$20	$3mo$	8%	25%	256	\$0.800920	\$0.801286	\$0.801413	$0.346ms$
\$20	\$20	$3mo$	8%	25%	512	\$0.801167	\$0.801643	\$0.801413	$1.355ms$
\$20	\$20	$3mo$	8%	25%	1024	\$0.801290	\$0.801449	\$0.801413	$5.381ms$
\$20	\$20	$3mo$	8%	25%	2048	\$0.801351	\$0.801404	\$0.801413	$25.00ms$
\$20	\$20	$3mo$	8%	25%	4096	\$0.801382	\$0.801441	\$0.801413	$104.07ms$
\$20	\$20	$3mo$	8%	25%	8192	\$0.801398	\$0.801420	\$0.801413	$415.38ms$

Table 4.1.2 – Sample put calculations using the trinomial method.

Figure 4.1.2 shows the convergence of the Kamrad-Ritchken and Jarrow-Rudd trinomial methods by plotting the relative error against the number of time steps.

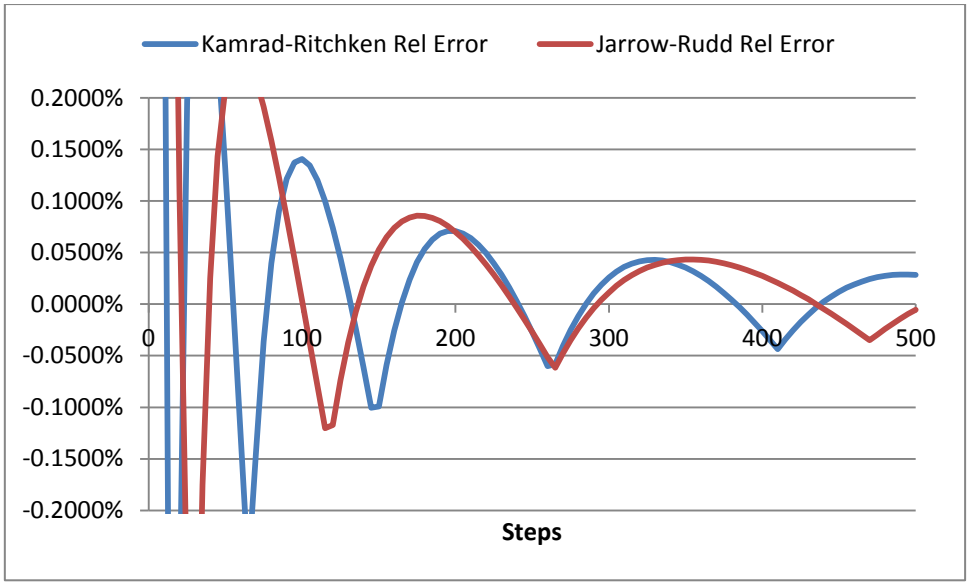

Fig 4.1.2 – Convergence of the Kamrad-Ritchken and Jarrow-Rudd trinomial methods. $S_0^{(0)} = \$53$, $K = \$55$, $T = 6\ months$, $r = 3\%$, $\sigma = 15\%$.

Chen, Chen and Chung (2001)[6] note that binomial methods have a convergence of roughly $\mathcal{O}(1/N)$. Due to the equivalence established in section 3.6 we should expect the same "rough" rate of convergence in these trinomial methods. Since we suspect the error is $\mathcal{O}(1/N)$, we should expect $error * N$ to be $\mathcal{O}(1)$. Figure 4.1.3 shows that this appears to be the case.

*Fig 4.1.3 – Relative error * N for the Kamrad-Ritchken and Jarrow-Rudd trinomial methods.* $S_0^{(0)} = \$53$, $K = \$55$, $T = 6\ months$, $r = 3\%$, $\sigma = 15\%$.

4.2 | Combining Multiple Underlying Prices

There's another really big optimisation to be made when calculating the option price for multiple underlying prices, which is often the case. When using the trinomial method, option values are calculated for every node in the tree, not just the root node. For future time steps, the option price is calculated for more and more underlying prices for *each* time step. Inspired by this idea, we place multiple nodes at $t = 0$, and their trees overlap. Indeed, the *vast majority* of the nodes can simply be shared by both trees. This is illustrated with just two nodes at $t = 0$ in figure 4.2.1.

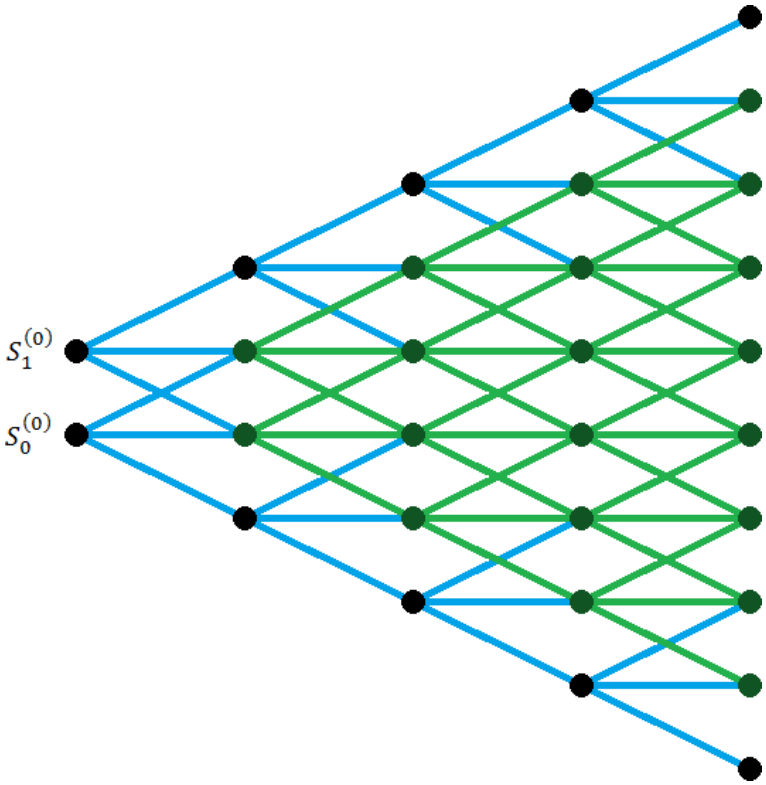

Fig 4.2.1 – Two overlapping trinomial trees.

If we use N_S for the number of starting nodes, then without this optimisation you would have to do N_S separate $\mathcal{O}(N^2)$ calculations, which is $\mathcal{O}(N_S N^2)$. With this optimisation, the calculation is $\mathcal{O}(N^2 + N_S N)$. In practical terms, this generally means that the option prices for the entirety of the desired domain of underlying prices can be calculated in almost the same time as the calculation of just one option price.

Figure 4.2.1 shows a plot of V against S with data calculated in this way.

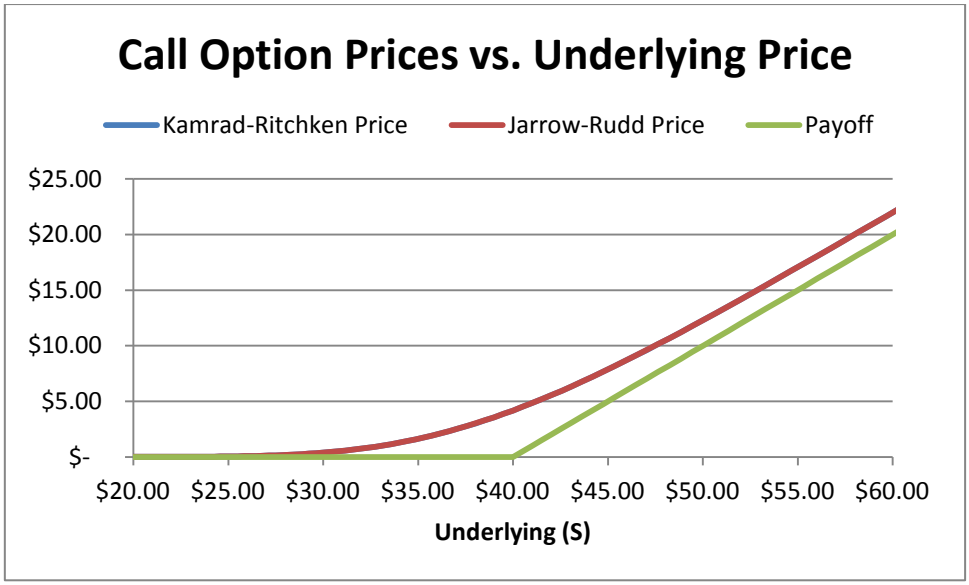

Fig 4.2.1 – Plot of call option price against underlying price. The Kamrad-Ritchken price curve is not visible as it is covered by the Jarrow-Rudd price curve. $K = \$40$, $T = 1\ year$, $r = 5\%$, $\sigma = 20\%$, $N = 512$.

Figure 4.2.2 shows a similar plot for put option prices.

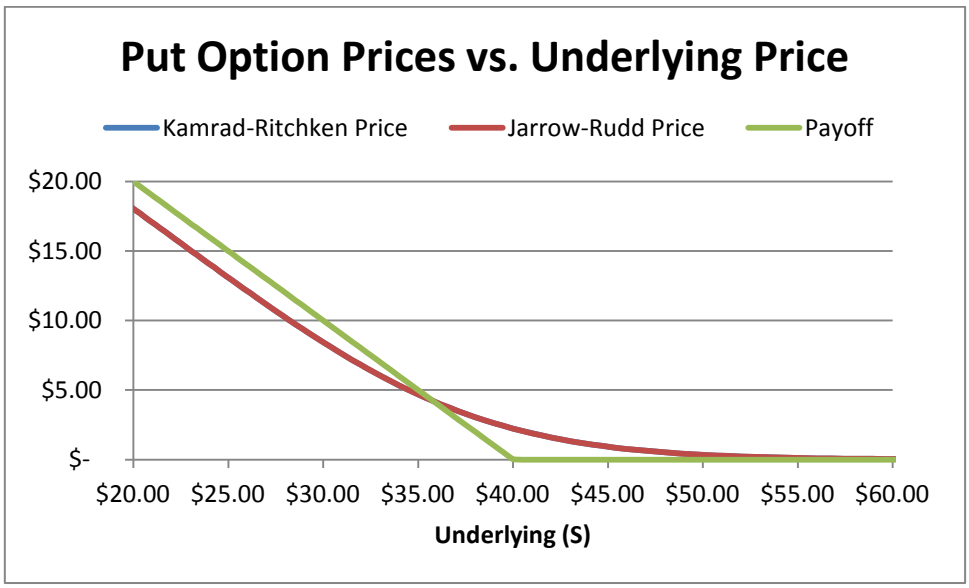

Fig 4.2.2 – Plot of put option price against underlying price. The Kamrad-Ritchken price curve is not visible as it is covered by the Jarrow-Rudd price curve. $K = \$40$, $T = 1\ year$, $r = 5\%$, $\sigma = 20\%$, $N = 512$.

4.3 | American Put Options

As mentioned in section 2.1, American put options must always be at least as valuable as the payoff function, since the option can be exercised early. For the trinomial method, implementing this is quite easy. Instead of using equation 2.5.4 to derive option values:

$$e^{r\Delta t}V_m^{(n)} = P_U V_{m+1}^{(n+1)} + P_M V_m^{(n+1)} + P_D V_{m-1}^{(n+1)}, \quad \text{- Eq 2.5.4}$$

we modify this to:

$$V_m^{(n)} = \max\left(e^{-r\Delta t}\left(P_U V_{m+1}^{(n+1)} + P_M V_m^{(n+1)} + P_D V_{m-1}^{(n+1)}\right), Q\left(S_m^{(n)}\right)\right).$$

- Eq 4.3.1

This can be seen visually as the option values formerly near or below the payoff function are lifted so that they no longer do so. This is illustrated in figure 4.3.1.

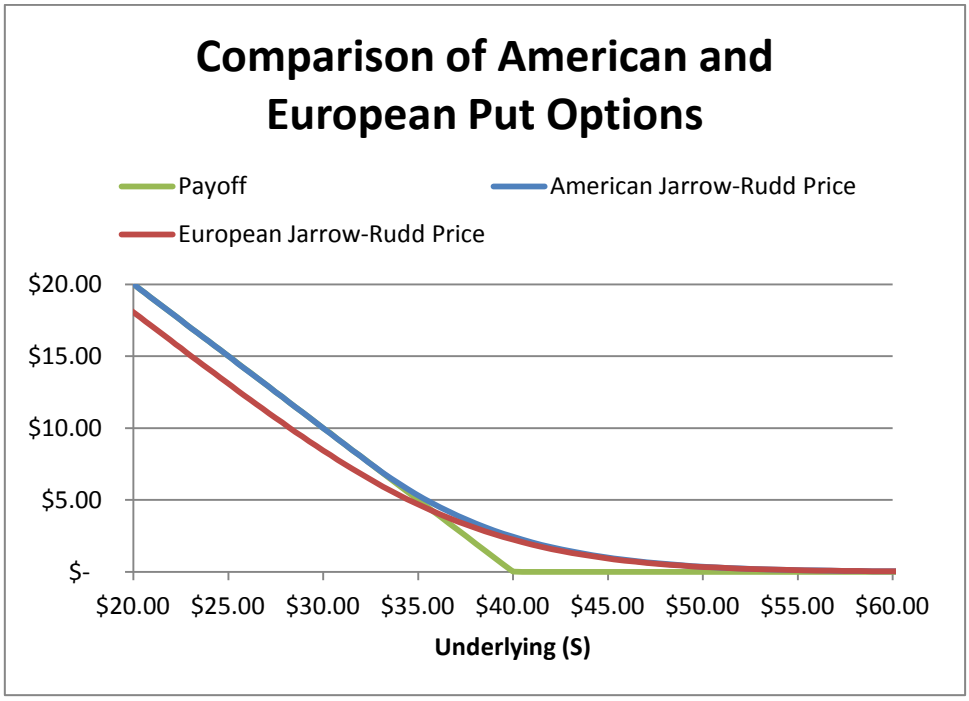

Fig 4.3.1 – American and European put options. Corresponding Kamrad-Ritchken prices are excluded as they would not be visible. $K = \$40$, $T = 1\ year$, $r = 5\%$, $\sigma = 20\%$, $N = 512$.

One ghastly way to implement this is to literally use equation 4.3.1 and calculate $S_m^{(n)}$ using equation 2.5.3:

$$S_m^{(n)} = U^m M^{n-m} S_0^{(0)}. \quad \text{- Eq 2.5.3}$$

When I tried this, calculating the American prices for figure 4.3.1 took 34.4 milliseconds. By comparison, calculating the European prices for figure 4.3.1 took just 0.890 milliseconds!

The reason for the miserable performance hit is that the European calculation for each node is actually extremely simple. By combining the discount with the $V_{m+1}^{(n+1)}$, $V_m^{(n+1)}$, $V_{m-1}^{(n+1)}$ coefficients, equation 2.5.4 can be implemented with just three multiplications and two additions, as well as another four integer additions for array look-up. Using `std::pow` for 2.5.3 probably means doing logarithms and exponentiations, which are far more expensive, not to mention the two extra multiplications, and taking the maximum of two values.

One solution is to keep track of $S_m^{(n)}$ while progressing vertically through the time slice. If starting at the top of the time slice, you can get each next S value using $S_{m-1}^{(n)} = (M/U)S_m^{(n)}$, a corollary of equation 3.4.3. Using this method, the time to calculate the figure 4.3.1 data dropped to a much more respectable 1.679 milliseconds, roughly double the European prices calculation.

Despite the significant extra costs of these straightforward approaches, the early exercise of American options actually provides optimisation *opportunities*. These should actually make it possible to achieve better performance than European option calculation. I leave this to others to implement, but basically, by keeping track of the optimal exercise boundary, it can be known ahead of time which nodes will be set to the payoff function and which will be calculated conventionally. The payoff function is significantly cheaper than equation 2.5.4, so those nodes should have a quicker calculation. In addition to this, with more sophistication, those nodes beyond the optimal exercise boundary could instead be pruned so that no calculation is needed for them at all.

4.4 | Dividends

Two kinds of dividends are quite simple to incorporate into the trinomial method. The first is the continuous dividend, which pays the holder of the asset at a continuous rate of $d_c S$. Continuous dividends cause asset prices to drift differently, since otherwise investors would always prefer assets which paid continuous dividends, which would provide an expected return of not only the interest rate, but also the dividend. Under the Efficient Market Hypothesis, which is part of the Black-Scholes framework, assets which pay continuous dividends must therefore drift at $r - d_c$ instead of r.

What this means for the trinomial method, is that asset prices should additionally decline by a factor of $e^{d_c \Delta t}$ for each time step. This can be incorporated into existing parameters by multiplying each of U, M, and D by $e^{-d_c \Delta t}$. For example, the Jarrow-Rudd parameterisation can be extended to options on assets with continuous dividends as follows:

$$U = e^{(\mu - d_c)\Delta t + \sigma\sqrt{2\Delta t}} \qquad P_U = 1/4$$

$$M = e^{(\mu - d_c)\Delta t} \qquad P_M = 1/2$$

$$D = e^{(\mu - d_c)\Delta t - \sigma\sqrt{2\Delta t}} \qquad P_D = 1/4,$$

- Eq 4.4.1-6

where:

$$\mu = r - \frac{1}{2}\sigma^2. \qquad \text{- Eq 2.5.13}$$

Table 4.4.1 contains some option prices calculated in this way.

$S_0^{(0)}$	K	T	r	σ	d_c	*Steps*	*Trinomial*	*Analytic*
\$5.10044	\$7	$1y$	7%	20%	1%	1024	\$0.05401	\$0.05402
\$5.98003	\$7	$1y$	7%	20%	1%	1024	\$0.25016	\$0.25014
\$6.70820	\$7	$1y$	7%	20%	1%	1024	\$0.58383	\$0.58379
\$7.86506	\$7	$1y$	7%	20%	1%	1024	\$1.40743	\$1.40741
\$8.82276	\$7	$1y$	7%	20%	1%	1024	\$2.25681	\$2.25681

Table 4.4.1 – Sample call calculations with continuous dividends based on the Jarrow-Rudd trinomial method.

The other kind of dividend simple to incorporate is a discrete dividend that is paid as a proportion of the underlying price at that time. In a similar way to continuous dividends, this has the effect at the time of the dividend of the asset price losing d_p of its value (multiplying by $1 - d_p$). There is a slightly simpler argument this time that if this were not the case, arbitrageurs would buy the asset immediately before the dividend and sell immediately afterwards. The drop in price must match the dividend exactly or there would be a (theoretically) risk-free profit to be made.

In the trinomial method, this could be implemented by multiplying the asset prices by $1 - d_p$ at the time step closest to the dividend time. However, the impact of this is that every successive asset price $1 - d_p$ multiplied by what it would otherwise have been, including the final nodes. Therefore the same effect can be achieved by simply multiplying $S_0^{(0)}$ by $1 - d_p$. This suggests that the price of options with a proportional dividend d_p at some time before expiry can be calculated simply as $V\big(S(1 - d_p), t\big)$, which is indeed the case. Although it is trivial, for completeness, table 4.4.2 provides some numerical examples.

$S_0^{(0)}$	K	T	r	σ	d_p	*Steps*	*Trinomial*	*Analytic*
\$25	\$22	$6mo$	8%	30%	1.5%	512	\$4.1540	\$4.1540
\$60	\$78	$1y$	7%	20%	1%	512	\$1.0667	\$1.0669
\$5	\$7	$1y$	4%	20%	1.1%	512	\$0.0309	\$0.0309
\$10	\$8	$2y$	7%	20%	1%	512	\$3.0635	\$3.0635
\$40	\$45	$3mo$	5%	17%	3%	512	\$0.0825	\$0.0826

Table 4.4.2 – Sample call calculations with discrete proportional dividends based on the Jarrow-Rudd trinomial method.

Chapter 5

Final Thoughts

So this project was chiefly about the equivalence between the trinomial and finite difference methods of option pricing. The most remarkable and concise way to put this is that under the transforms $x = \log\left(Se^{-\left(r-(1/2)\sigma^2\right)t}\right)$, $U = Ve^{-rt}$, the explicit finite difference scheme of the Black-Scholes equation is identical to the Jarrow-Rudd binomial method, and thus identical to its derived trinomial method. Parallel to this, under the transforms $x = \log S$, $U = Ve^{-rt}$, the scheme is identical to the Kamrad-Ritchken trinomial method, for which there exists an equivalent binomial method. Probably the most important thing about equivalence is that it grants legitimacy to the trinomial methods, which are heuristic in their derivation.

I've been a bit surprised that this report has been so dominated by the theoretical side. As a result, the C++ implementation of the trinomial/finite difference method is less mature than I would have liked. In particular, the code for American put options is not optimal and the dividend calculations were simply hacked together and are not accessible in a user friendly manner.

Especially after studying numerical heat diffusion as a project last semester, perhaps the most important thing I've gained from this study is an easy familiarity with various numerical methods. I've always thought it incredibly important to gain intuitive understanding of mathematical concepts, and this project has delivered that for me. When explaining the background of finite differences, I was surprised at how natural it felt and how easily that content unfolded.

I've identified a few peripheral ideas that look worth exploring.

First, it appears that in the trinomial method, the weights of the nodes quickly become extremely small as they deviate from the centre. Having a minimum weight and pruning all nodes lower than that weight could result in a trinomial method which is computable in less than $\mathcal{O}(N^2)$ time. If done right, by lowering the minimum weight as N increases, it might be provable that the method converges and still does so at the rate of $\mathcal{O}(1/N)$.

Second, speaking of the rate of convergence, figure 4.1.3 suggests it follows a very predictable pattern. By identifying the properties of this pattern, perhaps calculations at multiple time steps could be used to produce option prices far more accurately than either of those calculations, or perhaps even the calculation that could be done with the combined time it took to perform them.

Also, I did some preliminary work on a hybrid scheme that derived the trinomial prices at the upper and lower edges of the tree in the conventional way, but used a Crank-Nicholson method and the tridiagonal matrix algorithm to calculate the prices in between. It appeared that the error from that scheme converged to double the error from the normal scheme, and that might conceivably be exploited in a similar way.

Third, I have a solution in mind for the problem of fixed discrete dividends. These discrete dividends pay a fixed amount instead of a proportion of the underlying. This causes a jump in the asset price at

the time of the dividend, but it can't be solved so easily because attempting the same approach used for proportional dividends would destroy the re-connectivity of the tree's nodes. However, this might be solved by perturbing the option prices at the relevant time step instead. We would have liked the asset prices to be adjusted by the dividend, but alternatively we could approximate the option prices that would have resulted from that adjustment. We can approximate $\partial V/\partial S$ by $\left(V_{m+1}^{(n)} - V_m^{(n)}\right)/\Delta S_m$, thus if we want to perturb the option values to what they would have been we could simply perform $V_m^{(n)} \to V_m^{(n)} - d_f * (\partial V/\partial S)$, where d_f is the fixed discrete dividend.

Fourth, the potential optimisations described at the end of section 4.3 for American put option pricing look useful. After all, since the Black-Scholes formula provides plain European option prices anyway, and what really motivates the binomial and trinomial methods is to price these and other complex options.

Finally, there is the prospect that a couple of results in this report are original. After some searching through academic papers, I have not found anyone describe the "perfect equivalence" I describe in section 3.5. Also, the fact that $x = \log\left(Se^{-\left(r-(1/2)\sigma^2\right)t}\right)$, $U = Ve^{-rt}$ transforms the Black-Scholes equation to the heat equation might be important because it appears non-trivially different from the transform usually used to do this in the process of deriving the Black-Scholes formula. As I initially only found Rubinstein's[1] result of equivalence with Kamrad-Ritchken parameters, I had hoped that my discovery of equivalence with Jarrow-Rudd parameters was previously undiscovered. However, there is a draft paper by Vonatsos[7] available online which notes this. It might still be an unpublished discovery though.

Bibliography

[1] Rubinstein, MR 2000, 'On the Relation Between Binomial and Trinomial Option Pricing Models', *Derivatives*, vol.8, no.2, pp47-50

Black-Scholes, accessed 1/11/2011
http://en.wikipedia.org/wiki/Black-Scholes

Tridiagonal matrix algorithm, accessed 1/11/2011,
http://en.wikipedia.org/wiki/Tridiagonal_matrix_algorithm

[2] Sowell, T 2011, *Basic Economics: A Common Sense Guide to the Economy*, 4th edn, Perseus Books Group, Cambridge.

[3] Zhu, S 2006, 'An exact and explicit solution for the valuation of American put options', *Quantitative Finance*, vol.6, no.3

[4] Rubinstein, MR 2000, 'On the Relation Between Binomial and Trinomial Option Pricing Models', *Derivatives*, vol.8, no.2, pp47-50 cited Jarrow, R & Rudd, A 1983, 'Option Pricing', *Banking & Finance*, vol.10, no.1, pp157-161.

[5] Rubinstein, MR 2000, 'On the Relation Between Binomial and Trinomial Option Pricing Models', *Derivatives*, vol.8, no.2, pp47-50 cited Kamrad, B & Ritchken, P 1991, 'Multinomial Approximating Models for Options with k-State Variables', *Management Science*, vol.37, no.12, pp1640-1652.

[6] Chen, H Chen, D Chung, S 2001, 'The accuracy and efficiency of alternative option pricing approaches relative to a log-transformed trinomial model', *Futures Markets*, vol.22, no.6, pp557-577.

[7] Vonatsos, KN 2006[*], 'Multinomial trees and explicit finite-difference schemes: a unifying approach', draft paper, accessed 7/11/2011,
http://research.mbs.ac.uk/accounting-finance/Portals/0/docs/2006/MultinomialandFDschemes_Vonatsos.pdf

[*] I found a presentation with the same name by Vonatsos mentioned at the web adress http://dereeacg.com/deree/publications.shtml with a date of 2006. I have no concrete date to associate with the draft paper though.

Appendix

A.1 | $\mu = r - (1/2)\sigma^2$

The term $r - (1/2)\sigma^2$ turns up in quite a number of places in this report. We also see $r + (1/2)\sigma^2$ in equation 2.2.5, part of the Black-Scholes formula, which is strangely inconsistent.

The reason for its appearance in the middle multiplier of the Jarrow-Rudd parameterisation $M = e^{\mu\Delta t}$ is to compensate for the volatility bringing up the mean. We can see this by calculating the weighted mean of U, M, D in the Jarrow-Rudd parameterisation while neglecting terms of $\mathcal{O}(\Delta t^{1.5})$. Remember this weighted mean should be approximately $1 + r\Delta t$, to match the discretised stochastic process given by equation 2.5.2.

$$Weighted\ Mean = \frac{1}{4}e^{\mu\Delta t+\sigma\sqrt{2\Delta t}} + \frac{1}{2}e^{\mu\Delta t} + \frac{1}{4}e^{\mu\Delta t-\sigma\sqrt{2\Delta t}}$$

$$= \frac{1}{4}\left(1 + \mu\Delta t + \sigma\sqrt{2\Delta t} + \frac{1}{2}\left(\mu\Delta t + \sigma\sqrt{2\Delta t}\right)^2\right) + \frac{1}{2}(1 + \mu\Delta t) + \frac{1}{4}\left(1 + \mu\Delta t - \sigma\sqrt{2\Delta t} + \frac{1}{2}\left(\mu\Delta t - \sigma\sqrt{2\Delta t}\right)^2\right)$$

$$= \frac{1}{4}\left(1 + \mu\Delta t + \sigma\sqrt{2\Delta t} + \frac{1}{2}\sigma^2 * 2\Delta t\right) + \frac{1}{2}(1 + \mu\Delta t) + \frac{1}{4}\left(1 + \mu\Delta t - \sigma\sqrt{2\Delta t} + \frac{1}{2}\sigma^2 * 2\Delta t\right)$$

$$= \frac{1}{4}(1 + \mu\Delta t + \sigma^2\Delta t) + \frac{1}{2}(1 + \mu\Delta t) + \frac{1}{4}(1 + \mu\Delta t + \sigma^2\Delta t)$$

$$= 1 + \left(\mu + \frac{1}{2}\sigma^2\right)\Delta t$$

So the volatility component brings up the mean by $(1/2)\sigma^2\Delta t$. Clearly, if we want the weighted mean to be $\approx 1 + r\Delta t$, we require that $\mu + (1/2)\sigma^2 = r$, which gives $\mu = r - (1/2)\sigma^2$.

A.2 | Finite Difference Scheme under $x = \log S$

We perform a logarithmic transformation of S so that the nodes will be distributed on a logarithmic scale:

$$x = \log S. \qquad \text{- Eq A.2.1}$$

This gives:

$$S = e^x$$

$$\frac{\partial S}{\partial x} = e^x = S$$

$$\frac{\partial x}{\partial S} = \frac{1}{S}. \qquad \text{- Eq A.2.2}$$

We'll need to replace $\partial V/\partial S$:

$$\frac{\partial V}{\partial S} = \frac{\partial V}{\partial x} * \frac{\partial x}{\partial S},$$

substitute A.2.2 for $\partial x/\partial S$:

$$\frac{\partial V}{\partial S} = \frac{1}{S} * \frac{\partial V}{\partial x} \qquad \text{- Eq A.2.3}$$

We'll also need something for $\partial^2 V/\partial S^2$:

$$\frac{\partial^2 V}{\partial S^2} = \frac{\partial}{\partial S}\left(\frac{\partial V}{\partial S}\right),$$

substitute A.2.2 for $\partial V/\partial S$:

$$\frac{\partial^2 V}{\partial S^2} = \frac{\partial}{\partial S}\left(\frac{1}{S} * \frac{\partial V}{\partial x}\right)$$

$$\frac{\partial^2 V}{\partial S^2} = -\frac{1}{S^2} * \frac{\partial V}{\partial x} + \frac{1}{S} * \frac{\partial}{\partial S}\left(\frac{\partial V}{\partial x}\right)$$

$$\frac{\partial^2 V}{\partial S^2} = -\frac{1}{S^2} * \frac{\partial V}{\partial x} + \frac{1}{S} * \frac{\partial x}{\partial S} * \frac{\partial}{\partial x}\left(\frac{\partial V}{\partial x}\right),$$

Substitute A.2.2 for $\partial x/\partial S$:

$$\frac{\partial^2 V}{\partial S^2} = -\frac{1}{S^2} * \frac{\partial V}{\partial x} + \frac{1}{S^2} * \frac{\partial^2 V}{\partial x^2}$$

$$\frac{\partial^2 V}{\partial S^2} = \frac{1}{S^2}\left(\frac{\partial^2 V}{\partial x^2} - \frac{\partial V}{\partial x}\right). \qquad \text{- Eq A.2.4}$$

Recall 2.2.2 the Black-Scholes equation:

$$\frac{\partial V}{\partial t}+\frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2}+rS\frac{\partial V}{\partial S}-rV=0. \qquad \text{- Eq 2.2.2}$$

Now substitute A.2.3, A.2.4 for $\partial V/\partial S$, $\partial^2 V/\partial S^2$:

$$\frac{\partial V}{\partial t}+\frac{1}{2}\sigma^2 S^2 * \frac{1}{S^2}\left(\frac{\partial^2 V}{\partial x^2}-\frac{\partial V}{\partial x}\right)+rS*\frac{1}{S}*\frac{\partial V}{\partial x}-rV=0$$

$$\frac{\partial V}{\partial t}+\frac{1}{2}\sigma^2\left(\frac{\partial^2 V}{\partial x^2}-\frac{\partial V}{\partial x}\right)+r\frac{\partial V}{\partial x}-rV=0$$

$$\frac{\partial V}{\partial t}+\frac{1}{2}\sigma^2\frac{\partial^2 V}{\partial x^2}+\left(r-\frac{1}{2}\sigma^2\right)\frac{\partial V}{\partial x}-rV=0. \qquad \text{- Eq A.2.5}$$

Recall 2.5.13:

$$\mu=r-\frac{1}{2}\sigma^2. \qquad \text{- Eq 2.5.13}$$

Now substitute 2.5.13 (RHS for LHS) in A.2.5:

$$\frac{\partial V}{\partial t}+\frac{1}{2}\sigma^2\frac{\partial^2 V}{\partial x^2}+\mu\frac{\partial V}{\partial x}-rV=0, \qquad \text{- Eq A.2.6}$$

discretise:

$$\frac{\Delta V}{\Delta t}+\frac{1}{2}\sigma^2\frac{\Delta^2 V}{\Delta x^2}+\mu\frac{\Delta V}{\Delta x}-rV=0$$

$$\frac{\left(V_m^{(n+1)}-V_m^{(n)}\right)}{\Delta t}+\frac{\sigma^2}{2\Delta x^2}\left(V_{m+1}^{(n+1)}-2V_m^{(n+1)}+V_{m-1}^{(n+1)}\right)+\frac{\mu}{2\Delta x}\left(V_{m+1}^{(n+1)}-V_{m-1}^{(n+1)}\right)-rV_m^{(n)}=0,$$

(Notice the central difference for $\Delta V/\Delta x$, See section 2.4.)

$$\frac{\left(V_m^{(n+1)}-V_m^{(n)}\right)}{\Delta t}-rV_m^{(n)}-\frac{\sigma^2}{\Delta x^2}V_m^{(n+1)}+\frac{\sigma^2+\mu\Delta x}{2\Delta x^2}V_{m+1}^{(n+1)}+\frac{\sigma^2-\mu\Delta x}{2\Delta x^2}V_{m-1}^{(n+1)}=0$$

$$(1+r\Delta t)V_m^{(n)}=\left(\frac{\sigma^2\Delta t}{2\Delta x^2}+\frac{\mu\Delta t}{2\Delta x}\right)V_{m+1}^{(n+1)}+\left(1-\frac{\sigma^2\Delta t}{\Delta x^2}\right)V_m^{(n+1)}+\left(\frac{\sigma^2\Delta t}{2\Delta x^2}-\frac{\mu\Delta t}{2\Delta x}\right)V_{m-1}^{(n+1)}.$$

- Eq 3.2.7

A.3 | Finite Difference Scheme under $x = \log(Se^{-\mu t})$

$$x = \log(Se^{-\mu t}) \quad \text{- Eq A.3.1}$$

$$\mu = r - \frac{1}{2}\sigma^2 \quad \text{- Eq 2.5.13}$$

Rewriting in terms of S we have:

$$e^x = Se^{-\mu t}$$

$$S = e^{x+\mu t}. \quad \text{- Eq A.3.2}$$

Differentiating by x, we get the following:

$$\frac{\partial S}{\partial x} = e^{x+\mu t} = S$$

$$\frac{\partial x}{\partial S} = \frac{1}{S}. \quad \text{- Eq A.3.3}$$

A.3.1 may be rewritten as $x = \log(S) - \mu t$. From this we have:

$$\frac{\partial x}{\partial t} = -\mu. \quad \text{- Eq A.3.4}$$

We will want something to replace $\partial V/\partial S$:

$$\frac{\partial V}{\partial S} = \frac{\partial V}{\partial x} * \frac{\partial x}{\partial S},$$

substitute A.3.3:

$$\frac{\partial V}{\partial S} = \frac{1}{S} * \frac{\partial V}{\partial x}. \quad \text{- Eq A.3.5}$$

We also need to replace $\partial^2 V/\partial S^2$:

$$\frac{\partial^2 V}{\partial S^2} = \frac{\partial}{\partial S}\left(\frac{\partial V}{\partial S}\right),$$

substitute A.3.5:

$$\frac{\partial^2 V}{\partial S^2} = \frac{\partial}{\partial S}\left(\frac{1}{S} * \frac{\partial V}{\partial x}\right) = \frac{\partial x}{\partial S} * \frac{\partial}{\partial x}\left(\frac{1}{S} * \frac{\partial V}{\partial x}\right),$$

substitute A.3.3:

$$\frac{\partial^2 V}{\partial S^2} = \frac{1}{S} * \frac{\partial}{\partial x}\left(\frac{1}{S} * \frac{\partial V}{\partial x}\right) = \frac{1}{S} * \left(\frac{1}{S} * \frac{\partial^2 V}{\partial x^2} + -\frac{1}{S^2} * \frac{\partial S}{\partial x} * \frac{\partial V}{\partial x}\right),$$

from A.3.3 $\partial x/\partial S = 1/S$ so $\partial S/\partial x = S$:

$$\frac{\partial^2 V}{\partial S^2} = \frac{1}{S} * \left(\frac{1}{S} * \frac{\partial^2 V}{\partial x^2} + -\frac{1}{S^2} * S * \frac{\partial V}{\partial x}\right) = \frac{1}{S^2}\left(\frac{\partial^2 V}{\partial x^2} - \frac{\partial V}{\partial x}\right). \quad \text{- Eq A.3.6}$$

For those not particularly familiar with multivariate calculus, what comes next is a tricky and subtle point: $\partial V/\partial t$ also needs replacing. The problem is that $\partial V/\partial t$ originally meant the derivative of V across t as S remains constant. However, after V changes from a function of S and t to a function of x and t, $\partial V/\partial t$ will mean the derivative of V across t as x remains constant, which is different. If t goes to t_+ while S is constant, x changes from $\log(Se^{-\mu t})$ to $\log(Se^{-\mu t_+})$. Similarly, if t goes to t_+ while x remains constant, S will go from $e^{x+\mu t}$ to $e^{x+\mu t_+}$.

The equation below comes from the theory of multivariate calculus, and relates the *original meaning* of $\partial V/\partial t$ on the left to the *new meaning* of $\partial V/\partial t$ on the right:

$$\frac{\partial V}{\partial t} = \frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} * \frac{\partial x}{\partial t},$$

substitute A.3.4:

$$\frac{\partial V}{\partial t} = \frac{\partial V}{\partial t} - \mu\frac{\partial V}{\partial x}. \quad \text{- Eq A.3.7}$$

Recall the Black-Scholes equation (2.2.2):

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0, \quad \text{- Eq 2.2.2}$$

now substitute A.3.5, A.3.6, A.3.7:

$$\frac{\partial V}{\partial t} - \mu\frac{\partial V}{\partial x} + \frac{1}{2}\sigma^2 S^2 * \frac{1}{S^2}\left(\frac{\partial^2 V}{\partial x^2} - \frac{\partial V}{\partial x}\right) + rS * \frac{1}{S} * \frac{\partial V}{\partial x} - rV = 0$$

$$\frac{\partial V}{\partial t} - \mu\frac{\partial V}{\partial x} + \frac{1}{2}\sigma^2\left(\frac{\partial^2 V}{\partial x^2} - \frac{\partial V}{\partial x}\right) + r\frac{\partial V}{\partial x} - rV = 0,$$

substitute 2.5.13:

$$\frac{\partial V}{\partial t} - \left(r - \frac{1}{2}\sigma^2\right)\frac{\partial V}{\partial x} + \frac{1}{2}\sigma^2\left(\frac{\partial^2 V}{\partial x^2} - \frac{\partial V}{\partial x}\right) + r\frac{\partial V}{\partial x} - rV = 0$$

$$\frac{\partial V}{\partial t} - \left(r - \frac{1}{2}\sigma^2\right)\frac{\partial V}{\partial x} + \frac{1}{2}\sigma^2\frac{\partial^2 V}{\partial x^2} + \left(r - \frac{1}{2}\sigma^2\right)\frac{\partial V}{\partial x} - rV = 0$$

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2\frac{\partial^2 V}{\partial x^2} - rV = 0. \quad \text{- Eq A.3.8}$$

Now discretise:

$$\frac{\Delta V}{\Delta t} + \frac{1}{2}\sigma^2\frac{\Delta^2 V}{\Delta x^2} - rV = 0,$$

just like in section 3.1, we take $\Delta^2 V/\Delta x^2$ from the $(n+1)$th time step and rV from the nth time step:

$$\frac{V_m^{(n+1)} - V_m^{(n)}}{\Delta t} + \frac{1}{2}\sigma^2 \frac{V_{m+1}^{(n+1)} - 2V_m^{(n+1)} + V_{m-1}^{(n+1)}}{\Delta x^2} - rV_m^{(n)} = 0$$

$$V_m^{(n+1)} - V_m^{(n)} + \frac{\sigma^2 \Delta t}{2\Delta x^2}\left(V_{m+1}^{(n+1)} - 2V_m^{(n+1)} + V_{m-1}^{(n+1)}\right) - r\Delta t V_m^{(n)} = 0$$

$$(1 + r\Delta t)V_m^{(n)} = V_m^{(n+1)} + \frac{\sigma^2 \Delta t}{2\Delta x^2}\left(V_{m+1}^{(n+1)} - 2V_m^{(n+1)} + V_{m-1}^{(n+1)}\right)$$

$$(1 + r\Delta t)V_m^{(n)} = \frac{\sigma^2 \Delta t}{2\Delta x^2}V_{m+1}^{(n+1)} + \left(1 - \frac{\sigma^2 \Delta t}{\Delta x^2}\right)V_m^{(n+1)} + \frac{\sigma^2 \Delta t}{2\Delta x^2}V_{m-1}^{(n+1)}.$$

- Eq 3.4.2