Derivative Securities, Courant Institute, Fall 2008
http://www.math.nyu.edu/faculty/goodman/teaching/DerivSec08/index.html
Jonathan Goodman and Keith Lewis

Supplementary notes and comments, Section 8

1 Dynamics, initial values, final values

Suppose $f(t)$ represents the state of a system at time t. By *dymanics*, we mean relations between values of $f(t$ for different values of t. In the best case, knowing $f(0)$ would determine $f(t)$ for all other t. A special case is to determine the values of $f(t)$ for $t > 0$ from the given $f(0)$. This is the *initial value problem*. A slightly more general form of the initial value problem is to give $f(T)$ for some specific T and ask for values $f(t)$ for $t > T$. Below, well call this *marching forward in time*. The *final value problem* is specifying $f(T)$ and seeking the values $f(t)$ for $t < T$. This corresponds to marching backward in time.

We sill see that ordinary differential equations can "march" in either direction. But partial differential equations often have the ability to march only in one direction, not both. This little supplement is supposed to provide intuition, if not a mathematical foundation, for this situation.

Differential equations are a common way to express dynamics. For example, suppose that it takes a single number to specify the state of the system, which is to say that $f(t) \in R$, or that $f(t)$ at any given t is just a number. Suppose we know that this number satisfies the differential equation

$$\partial_t f \;=\; -rf \;. \tag{1}$$

Then $f(0)$ determines $f(t)$ through the formula

$$f(t) \;=\; f(0)\, e^{-rt} \;. \tag{2}$$

This formula holds for $t > 0$ or $t < 0$ – the ODE (1) can march either forward or backward. If we give the extra condition

$$f(0) \;=\; a \;, \tag{3}$$

then we get $f(t) = ae^{-rt}$.

There are several ways to view the process of deriving (2) from (1). One may be familiar from a class in ordinary differential equations. We multiply both sides by dt and divide by f to get

$$\frac{df}{f} \;=\; -r dt \;,$$

then we integrate both sides:

$$\ln(f(t)) \;=\; -rt \,+\, C \;,$$

then we get $f(t) = Ce^{-rt}$. Finally, the initial condition (3) determines $C = a$, and then

$$f(t) \;=\; ae^{-rt} \;. \tag{4}$$

A less satisfying (at first) approach is simply to check that the formula (2) satisfies the ODE (Ordinary Differential Equation) (1) and has the correct value (3). That is, simply check that the given function (4) satisfies the requirements we put on it.

The second approach (and also the first, if you think about it) relies on *uniqueness* of solutions of the initial value problem or the final value problem. Uniqueness means that there is only one function $f(t)$ that satisfies both the ODE (1) and the condition (3) is specified. Because of uniqueness, if you find a function that satisfies the ODE and the initial or final condition, then you have found what you were looking for. The uniqueness would be false (for a set of specifications) if those specifications did not determine f completely. For example, if you just give the ODE (1) but not the initial condition (3), then both $f(t) = e^{-rt}$ and $f(t) = 2e^{-rt}$ are candidates.

The other main theorem in the general theory of ordinary differential equations is *existence*, that there is a solution. With just the right number of specifications we have both existence and uniqueness. If we give too few conditions, we may have existence but not uniqueness, as in the example above. If we give too many conditions, we may have uniqueness but not existence. For example, if we seek a function that satisfies (1) together with the two conditions $f(0) = 1$ and $f(1) = 0$, then there is no solution at all. The problem is *overdetermined* and existence fails. If by some accident there were a solution, (say, we asked for $f(0) = 0$ and $f(1) = 0$), that solution would be unique.

In applications (finance and elsewhere), we use differential equations to find functions. For that, we want to know what combinations of differential equations and initial or final conditions determine the desired function completely. In practice, the actual f often is found by the computer. But before going to the computer, the practitioner should know that the problem makes sense.

For partial differential equations there is the more subtle point that the initial value problem or final value problem needs to be well posed. The precise definition of this is out of place here, but it boils down to the requirement that the solution should exist for *generic* initial conditions (or final conditions for the final value problem). The initial value problem is *ill posed* if there are solutions for some initial conditions, such as those given by specific formulas, but the same initial value problem with generic initial values does not have a solution[1].

[1]This is equivalent to the usual definition involving continuous dependence on the initial conditions by the closed graph theorem of functional analysis, as was pointed out by the Polish mathematician Banach.

2 Time stepping and existence

Marching, also called *time stepping*, provides important intuition as well as a mechanism for proving existence (and, if you think harder than we have time to do here, uniqueness). This means generating a sequence of approximate solutions that depend on a parameter δt and giving the solution as the $\delta t \to 0$ limit. We use the usual notation $t_k = k\delta t$.

For the initial value problem, we march forward in time. If we have $F_k(\delta t) \approx f(t_k)$, we get $F_{k+1} = f(t_{k+1})$ using the differential equation:

$$\frac{df}{dt} \;=\; -rf \quad \Longrightarrow \quad \frac{f(t_{k+1}) - f(t_k)}{\delta t} \;\approx\; rf(t_k) \quad \Longrightarrow \quad \frac{F_{k+1} - F_k}{\delta t} \;=\; -rF_k$$

which ultimately becomes

$$F_{k+1} \;=\; F_k \;-\; rF_k\delta t \;. \tag{5}$$

To construct an approximate solution to the initial value problem, we start with $F_0 = a$, and then us (5) to compute the numbers F_k for $k > 0$. If δt is very small, these numbers should be very close to the exact numbers $f(t_k)$.

This marching process is part of a mathematical proof of the *existence theorem*. Even if we do not know that there is a function $f(t)$ that satisfies the ODE and the initial condition, we can attempt to use the definition

$$f(t) \;=\; \lim_{\delta t \to 0,\; t_k = t} F_k \;. \tag{6}$$

The mathematics comes in showing first that the limit exists. Once we know that the limit exists, we must show that this f satisfies the differential equation.

The hardest part is showing that the limit exists, as we will see below. The marching method (5) is a finite difference approximation to the ODE (1). If nature created a solution to the ODE, then the marching scheme has a good chance to find it. If nature did not create a solution to the ODE, then something should go wrong with the marching method. What goes wrong (in most cases) is that teh limit (6) does not exist because the numbers F_k go to infinity (*blow up*) as $\delta t \to 0$ with $t_k = t$ fixed.

The ODE (1) also allows us to march backwards in time if we want to find approximate solutions to the final value problem. A simple way to do this would be

$$F_k \;=\; F_{k+1} + rF_{k+1}\delta t \;.$$

3 Expected values

We often use expressions for solutions in terms of expectations of things. This is a different way to define a function of time, so it gives a different possible way to construct a solution of the initial value problem or the final value problem. For example, suppose we have the OU process

$$dX(t) \;=\; -rX dt \;+\; \sigma dW \;. \tag{7}$$

Suppose we take $X(0) = a$ and define

$$f(t) \;=\; E\left[X(t)\right] \;. \tag{8}$$

We can show that this $f(t)$ satisfies the ODE (1) by calculating

$$df \;=\; dE\left[X(t)\right] \;=\; E\left[dX(t)\right] \;=\; E\left[-rX(t)dt + \sigma dW\right] \;=\; -rE\left[X(t)\right]dt \;=\; -rf(t)dt \;.$$

It also is clear that $f(0) = a$ because $X(0) = a$ is not random.

But the solution of the initial value problem is unique and we know what it is (4). This shows that

$$E\left[X(t)\right] \;=\; ae^{-mrt} \;.$$

We can do this without the explicit solution to (7) given in Kohn's notes.

4 The backward equation

If $X(t)$ satisfies the SDE

$$dX \;=\; a(X,t)dt \;+\; b(X,t)dW \;, \tag{9}$$

then the expected values

$$f(x,t) \;=\; E_{x,t}\left[V(X(T))\right] \;, \tag{10}$$

(recall that $E_{\cdot x,t}$ means that the expected value is taken under the condition that $X(t) = x$) satisfy

$$\partial_t f \;+\; \frac{1}{2}b(x,t)^2\partial_x^2 f \;+\; a(x,t)\partial_x f \;=\; 0 \;. \tag{11}$$

The function $f(x,t)$ defined by (10) obviously satisfies the equation

$$f(x,T) \;=\; V(x) \;. \tag{12}$$

A slight generalization is that the function,

$$f(x,t) \;=\; r^{-r(T-t)}\,E_{x,t}\left[V(X(T))\right] \;, \tag{13}$$

satisfies the backward equation

$$\partial_t f \;+\; \frac{1}{2}b(x,t)^2\partial_x^2 f \;+\; a(x,t)\partial_x f \;-\; rf \;=\; 0 \;. \tag{14}$$

The *existence theorem* states[2] that there is a solution to the PDE (11) defined for all $t \leq T$ that also satisfies the final conditions (12). In our case, this theorem may not be strictly necessary, given that we already have a solution given by the formula (10). The *uniqueness theorem* states that there is the function $f(x,t)$

[2]Disclaimer: there are technical hypotheses to this theorem. A good book on PDE (e.g by Fritz John, or L. C. Evans) has the full story.

is completely determined by the condtions (11) and (12). This means that if $f(x,t)$ satisfies these conditions, it must be the same as the function defined by (10).

These theorems are used in both directions. If we know that $f(x,t)$ satisfies (11) or (14), than the foumulas (10) or (13) provide the solution. The SDE (9) is found from the PDE by matching coefficients. This is how the Black Scholes derivation goes. First we use the hedging argument to show that the option price satisfies

$$\partial_t f \ + \ \frac{s^2\sigma^2}{2}\partial_s^2 f \ + \ rs\partial_s f \ - \ rf \ = \ 0 \ .$$

Then we argue that this is a backward equation of the form (14), so the solution is given by

$$f(s,t) \ = \ e^{-r(T-t)}E_{s,t}\left[V(S(T))\right] \ .$$

where

$$dS \ = \ rSdt \ + \ \sigma SdW \ . \tag{15}$$

I repeat (maybe not for the last time) that this does not mean that the actual stock price satisfies the SDE (15). In fact, we assumed that $S(t)$ satisfies a different process. The *risk neutral* process (15) is defined to make (14) work, not to match the actual dynamics of $S(t)$.

The other direction is to find a solution to the final value problem somehow and use this to compute the expected value (10) or (13). This is illustrated by the last part of Problem 3 of Homework 6. It is complicated to compute the fourth moment of $X(t)$ directly from the solution formula, but it is easy to find the solution of the PDE directly as a polynomial with time dependent coefficients. Once you find the solution, it is *the* solution (uniqueness), so it is the expected value (10).

These theorems also have implications for computing, that again go both ways. The expected values (10) or (13) may be computed by numerical solution of the PDE, as explained below. This is often preferable to a direct Monte Carlo approach because the finite difference method has no statistical error. Monte Carlo usually has large statistical errors, so that it takes a huge number of paths to achieve even medium accuracy. The other way is that we may use Monte Carlo to find the solution of the final value problems (11) or (14). This is done, despite the inaccuracy of Monte Carlo methods, for higher dimensional models where direct finite difference methods are impractical (the so called *curse of dimensionality*). We do not discuss this further here.

5 The log variable

The Black Scholes PDE can be reformulated using the log variable transformation

$$x \ = \ \log(s/K) \ = \ \log(s) \ - \ \log(K) \ . \tag{16}$$

The $-\log(K)$ on the right has the effect that $s = K$ corresponds to $x = 0$, which is convenient but not essential. This transformation makes the numerical

solution of the Black Scholes equation easier and has some theoretical uses (which will not appear in this class).

We rewrite the Black Scholes equation in the log variable using the chain rule. First

$$\frac{\partial f}{\partial s} \;=\; \frac{\partial f}{\partial x} \cdot \frac{\partial x}{\partial s} \;=\; \frac{1}{s} \cdot \partial_x f \; .$$

Next

$$\partial_s^2 f \;=\; \partial_s \left(\frac{1}{s} \cdot \partial_x f \right) \;=\; \frac{-1}{s^2} \cdot \partial_x f \;+\; \frac{1}{s} \partial_s \left(\partial_x f \right) \; .$$

For the second term on the right, we apply the first derivative formula above to get

$$\partial_s \left(\partial_x f \right) \;=\; \frac{1}{s} \partial_x^2 f \; .$$

The whole result is

$$\partial_s^2 f \;=\; \frac{1}{s^2} \left(\partial_x^2 f - \partial_x f \right) \; .$$

The Black Scholes *operator* is

$$\mathcal{L}_{BS} f \;=\; \frac{\sigma^2 s^2}{2} \partial_s^2 f \;+\; r \partial_s f \;-\; r f \;=\; \frac{\sigma^2}{2} \left(\partial_x^2 f - \partial_x f \right) \;+\; r \partial_x f \;-\; r f \; ,$$

so the Black Scholes equation is

$$\partial_t f \;+\; \frac{\sigma^2}{2} \partial_x^2 f \;+\; \left(r - \frac{\sigma^2}{2} \right) \partial_x f \;-\; r f \;=\; 0 \; . \tag{17}$$

In its original form, the coefficients of $\partial_s f$ and $\partial_s^2 f$ were functions of s. In the new form (17), the coefficients are independent of s or x. This makes the numerical finite difference solution easier, as we will see below. It also makes it possible to apply standard techniques from the theory of partial differential equations, such as the Fourier transform and Green's functions. We will not pursue those applications in this class.

There is a related way to derive the new formulation (17) of the Black Scholes equation. Suppose $S(t)$ is the geometric Brownian motion process (15) and $X(t) = \log(S(t)) - \log(K)$. From the Ito calculus, we find

$$\begin{aligned} dX &= \left(\partial_s X \right) dS \;+\; \frac{1}{2} \left(\partial_s^2 X \right) (dS)^2 \\ &= \frac{1}{S} \left(rS dt + \sigma S dW \right) \;-\; \frac{1}{2} \frac{1}{S^2} \sigma^2 S^2 dt \\ dX &= \left(r - \frac{\sigma^2}{2} \right) dt \;+\; \sigma dW \; . \end{aligned} \tag{18}$$

The backward equation for (18) is exactly (17). The extra term $\frac{\sigma^2}{2} \partial_x f$ in (17) corresponds to the Ito term involving $(dS)^2$. This kind of thing is discussed much more in the stochastic calculus class.

6 Finite difference solution of the backward equation

The finite difference method is a way to construct an approximation to $f(x,t)$ by marching backwards in time from the final time T towards the present $t = 0$. If δt is a small time step, we go from $f(x,T)$ to $f(x,T-\delta t)$ to $f(x,T-2\delta t)$ and so on back to $t = 0$. The difference between a PDE such as (11) and an ODE such as (1) is that at each t, the solution at time t is whole function of x. The computer cannot store the values $f(x,t)$ for all t, so we must make more approximations.

The *finite difference* approach is to store approximations to the values $f(x_j, t_k)$, where the *grid points* x_j are uniformly spaced with *grid spacing* δx. That is, $t_k = k\delta t$ and $x_j = j\delta x$. The approximate values are $F_{jk} \approx f(x_j, t_k)$. The finite difference marching method starts by using the final conditions to get $F_{jn} = V(x_j)$ for all j, where $t_n = T = n\delta t$. It then computes all the numbers $F_{j,n-1}$ using the finite difference formula below. This is one *time step*. The general time step starts with the numbers F_{jk} for all j and computes all numbers $F_{j,k-1}$ for all j. This continues until the desired numbers $F_{j,0}$ have been computed. That is, we start with final values and take time steps backwards in time until time $t = 0$.

The finite difference formula that allows marching in time comes from finite difference approximations to derivatives. We need three of these. For a function $g(y)$, we have

$$g'(y) \;\approx\; \frac{g(y+\delta y) \,-\, g(y)}{\delta y} \;\;, \tag{19}$$

$$g'(y) \;\approx\; \frac{g(y+\delta y) \,-\, g(y-\delta y)}{2\delta y} \;\;, \tag{20}$$

and

$$g''(y) \;\approx\; \frac{g(y+\delta y) \,-\, 2g(y) \,+\, g(y-\delta y)}{\delta y^2} \;\;. \tag{21}$$

In particular, we use

$$\partial_t f(x,t) \;\approx\; \frac{f(x,t) - f(x,t-\delta t)}{\delta t} \;, \tag{22}$$

$$\partial_x f(x,t) \;\approx\; \frac{f(x+\delta x,t) - f(x-\delta x,t)}{2\delta x} \;, \tag{23}$$

and

$$\partial_x f(x,t) \;\approx\; \frac{f(x+\delta x,t) - 2f(x,t) + f(x-\delta t,t)}{\delta x^2} \;. \tag{24}$$

These formulas allow us to approximate derivatives of f by finite differences of the numbers F_{jk} using facts such as $x_j + \delta x \; x_{j+1}$. For example,

$$\partial_t f(x_j,t_k) \;\approx\; \frac{f(x_j,t_k) - f(x_j,t_{k-1})}{\delta t} \;\approx\; \frac{F_{jk} - F_{j,k-1}}{\delta t} \;.$$

The other two derivative approximations we use are

$$\partial_x f(x_j, t_k) \;\approx\; \frac{F_{j+1,k} - F_{j-1,k}}{2\delta x} \;,$$

and

$$\partial_x^2 f(x_j, t_k \;\approx\; \frac{F_{j+1,k} - 2F_{jk} + F_{j-1,k}}{\delta t^2} \;.$$

There are two steps in deriving a finite difference approximation that can be used for time marching. The first is to use the differential equation (17) to give an algebraic formula relating different numbers F_{jk}. We do this by replacing each of the derivatives in (17) by it's finite difference approximation using the F_{jk}:

$$\frac{F_{jk} - F_{j,k-1}}{\delta t} + \frac{\sigma^2}{2}\,\frac{F_{j+1,k} - 2F_{jk} + F_{j-1,k}}{\delta t^2} + \left(r \;-\; \frac{\sigma^2}{2}\right)\frac{F_{j+1,k} - F_{j-1,k}}{2\delta x} - rF_{jk} \;=\; 0 \;. \tag{25}$$

The second step is to use this equation to find a formula for $F_{j,k-1}$ in terms of values of F at time t_k:

$$F_{j,k-1} \;=\; F_{jk} + \frac{\delta t}{\delta x^2}\,\frac{\sigma^2}{2}\,(F_{j+1,k} - 2F_{jk} + F_{j-1,k}) + \frac{\delta t}{2\delta x}\left(r \;-\; \frac{\sigma^2}{2}\right)(F_{j+1,k} - F_{j-1,k}) - \delta t r F_{jk} \;.$$

This may be expresses as

$$F_{j,k-1} \;=\; aF_{j+1,k} \;+\; bF_{jk} \;+\; cF_{j,k-1} \;, \tag{26}$$

where

$$a \;=\; \frac{\sigma^2}{2}\frac{\delta t}{\delta x^2} \;+\; \frac{\delta t}{2\delta x}\left(r \;-\; \frac{\sigma^2}{2}\right) \;, \tag{27}$$

$$b \;=\; 1 \;-\; \sigma^2\,\frac{\delta t}{\delta x^2} \;-\; r\delta t \;, \tag{28}$$

$$c \;=\; \frac{\sigma^2}{2}\frac{\delta t}{\delta x^2} \;-\; \frac{\delta t}{2\delta x}\left(r \;-\; \frac{\sigma^2}{2}\right) \;, \tag{29}$$

The computational algorithm is to set $F_{jn} = V(x_j)$ then take a series of time steps that compute the numbers $F_{j,k-1}$ from the numbers F_{jk} using (26). The various choices made in (22), (23), and (24) make that possible. In particular, if we had used

$$\partial_t f \;\approx\; \frac{f(x, t+\delta t) - f(x,t)}{\delta t} \;,$$

it would have got instead of (25) a relation between $F_{j,k+1}$ and the numbers $F_{j-1,k}$, F_{jk}, and $F_{j+1,k}$. This would not help us determine the time $k-1$ numbers from the time k numbers with an explicit formula like (26). The choices we made give approximations that are both true and useful for marching backwards in time.

There is an important *stability constraint* relating the time step δt and the space step δx. The constraint essentially is that the coefficient b should be non-negative. The condition simplifies if we neglect the last term on the right of (28). We do this because when δx is small, the fraction $\delta t/\delta x^2$ is much larger than δt. In this case, the condition $b \geq 0$ is

$$\delta t \;\leq\; \frac{\delta x^2}{\sigma^2} \;. \tag{30}$$

Conditions such as (30) were first introduced in a paper of 1928 by Richard Courant, Kurt Friedrichs, and Hans Lewy. At the time they were at the mathematics institute in Göttingen, which was the most distinguished center of mathematics in Germany and possibly the world. Ten years later, all three had been chased out of German and were in the United States. Courant founded an Institute of Mathematical Sciences at New York University and hired Kurt Friedrichs as one of its first faculty members. Hans Lewy joined the mathematics department at Berkeley. In 1973, the Institute of Mathematical Sciences at NYU was renamed, after its founded and long time Director, the Courant Institute of Mathematical Sciences. The stability condition (30) is called the CFL condition after the three authors. The ratio $\lambda = \sigma^2\delta t/\delta x^2$ is called the CFL ratio or the *Courant number*. The stability condition is that the Courant number should be less than (or equal to) one.

In practice, a good way to satisfy the CFL stability condition is to specify δx and the CFL ratio λ and have the program choose

$$\delta t \;=\; \lambda\delta x^2/\sigma^2 \;. \tag{31}$$

This is what the posted program for assignment 7 does. Of course, the computational work depends on the number of time steps, which depends on the size of the time step. Smaller λ means smaller δt, more time steps, and a longer run time. The largest possible time step, $\lambda = 1$, has certain computational drawbacks that we do not go into here. The posted code has $\lambda = 1/2$, which is reasonable for the computational experiments we will do in assignment 7.

The formula (26) gives $F_{j,k-1}$ in terms of three neighboring values at time t_k. For this reason is sometimes is called the *trinomial tree* method. If you neglect the $r\delta t$ term in (28) and take the largest possible time step consistent with stability in (31), which is $\lambda = 1$, you get $b = 0$. Thus, the binomial tree method is what the trinomial tree method becomes if you take the largest possible time step allowed by stability.

The trinomial tree interpretation is made stronger if the stability condition is satisfied and we neglect $r\delta t$ in (28). In that case, for δt small enough, $a > 0$, $b > 0$, and $c > 0$ and $a + b + c = 1$. In other words, $F_{j,k-1}$ is an average of the numbers $F_{j+1,k}$, F_{jk}, and $F_{j-1,k}$. In the case $b = 0$, these are the binomial tree coefficients q_u and q_d from before. The difference is that there is no arbitrage argument to derive the formulas (27), (28), and (29). Instead, the derivation is purely mathematical. It starts with the Black Scholes PDE, derived the way Black and Scholes did, then makes finite difference approximations. The

trinomial tree method is better in practice than the binomial tree method. In the end, the binomial tree method is just a way of explaining derivatives pricing to people who can't or won't or don't have the time to figure out the Ito calculus and stochastic differential equations.

There's another similarity between this finite difference method and the binomial tree method, contraction. Up to now we have ignored possible limits on the spatial variable, j. It is not possible for j to go from $-\infty$ to ∞ in the computer. There must be a lower and upper bound. Suppose these bounds are $j_{\min}(k) \le j \le j_{\max}(k)$, and that we have computed values F_{jk} for all j values in this range. The finite difference (trinomial tree) formula (26) allows us to calculate values $F_{j,k-1}$ for $j_{\min}(k)+1 \le j \le j_{\max}(k)-1$, but not for $F_{j_{\min}(k),k-1}$ of $F_{j_{\max}(k),k-1}$ That is, $j_{\min}(k-1) = j_{\min}(k)+1$ and $j_{\max}(k-1) = j_{\max}(k)-1$. The number of active j values decreases by two each time step. This is also true of the binomial tree method. If we want to take n time steps, we start with $2n+1$ values. each time step the number of values decreases by two, so that at the end there is just one value left.

The code that is posted works as follows. The user specifies the following parameters using `#define` statements at the top of the main program:

- The numbers x_{min}, and x_{max}. The code is supposed to compute the solution $f(x,0)$ for $x \in [x_{min}, x_{max}]$.
- The space step δx desired. The code may not use the value specified. It will use the largest possible δx smaller than the one you specified so that nx, the number of gris points remaining at time $t = 0$ is an integer. The formula is $x_{max} - x_{min} = (nx-1)\delta x$, because there are $nx-1$ intervals of length δx between left endpoint is x_{min}, the right endpoint, x_{max}.
- The parameters of the problem, σ, and r. The code as posted is for the *heat equation*, which is $\partial_t u + \frac{D}{2}\partial_x^2 u = 0$. You will have to modify it so that you can specify σ and r, the strike price, K, and whether it's a put or a call, instead of D. You might also want to rename the solution variable f instead of u to be consistent with the notation of these notes. That is not necessary, but it might be helpful.
- The final time, T.
- The CFL parameter, λ. The code posted has $\lambda = 1/2$. Which will work for most of the assignment.

The code has two procedures, `FinalValues`, and `TimeStep`. See the header file for the precise calling arguments and return values. `FinalValues` sets the final values, basically $F_{jn} = V(x_j)$ for j in the appropriate range (larger than x_{min}, x_{max}, see above). For options valuation, you will set these to the payouts for a put or call. Remember to take into account the log change of variables, so that, for a put, $V(x) = (K - s(x))_+ = K\left(1 - e^x\right)_+$. `TimeStep` takes as arguments two arrays, called `uOld` and `uNew` (which you can make `fOld` and `fNew`). It assumes `uOld` has the values F_{jk}. It computes the values $F_{j,k-1}$

and puts these into `uNew`. The calling program calles `TimeStep` n times but alternating between `TimeStep( u2, u1, ...)` and `TimeStep( u1, u2, ...)`. The call `TimeStep( u2, u1, ...)` treats `u2` as `uNew` and `u1` as `uOld`. The call `TimeStep( u1, u2, ...)` does the reverse. In this way, we use only two one dimensional arrays rather than one two dimensional array to hold all the numbers F_{jk}. This cuts the storage requirement by a factor of $n/2$, which may not be so important in the present application but matter a lot in larger scale computation.

The code computes a time step in a way similar to the way it computes a space step. Once it has determined δx, it uses the formula (31) to get a time step. It then reduces δt a little more to insure that the number of time steps of size δt is an integer. One of the advantages of the trinomial tree method over the binomial tree method is that with the trinomial tree, it is possible to adjust the time step and the space step separately (within limits determined by the stability condition). This is not possible with the binomial tree method.

7 American options

An *American style* option is an option that may be exercised at any time up to the *expiration time*, T. For example, an American style put (an American put) with strike price K may be exercised at any time $t \leq T$ to trade one share of stock with price S_t for cash amount K. This terminology is not related to geography. There are European style options for sale in America and American style options in Europe. But many exchange traded stock options are American style. It is clear that an American option value is at least as great as the European style option with the same expiration and strike price, because the American option gives the holder more rights than the European option does. The price of an American option is an increasing function of the expiration time, T, for the same reason.

It is curious to note that there is a kind of option part way between American and European. These options have a set number of exercise dates, T_k. The option may be exercised not at an arbitrary time, but at one of these determined dates. Interest rate options, swaptions in particular, often have this feature. Such options are called *Bermudan*, because Bermuda is between Europe and America. Actually, Bermuda is closer to America than Europe (at least geographically). Pricing Bermudan options is more like pricing American options than European. In particular, standard Monte Carlo methods that apply to European options do not apply to Bermudans or Americans.

In the context of American options, we often call $V(s)$ the *intrinsic value* rather than the payout. Because the option may be exercised at any moment, the price of the option may never fall below the intrinsic value (which would be an arbitrage of the simplest kind).

The holder of an American style option is faced with an *optimal execution* problem. If she holds an option at time t, she must decide at time t whether to exercise the option or hold it (not exercise). This decision is made on the

basis of the known price $S(t)$, and possibly on known prices $S(t')$, for $t' < t$. It may not depend on future prices $S(t')$, for $t' > t$. For simple puts and calls, it seems clear that the optimal strategy involves a *decision boundary*, $b(t)$. For $S(t)$ on one side of $b(t)$, you hold the option. On the other side you exercise. For example, for a put, you exercise if $S(t) \leq b(t)$ and hold if $S(t) > b(t)$. This makes intuitive sense. A put becomes more valuable as the stock price decreases. If you exercise early for that value when $S(t) = b(t)$, you probably exercise early also for the greater value when $S(t) < b(t)$. For a put, the *hold region* is $S(t) > b(t)$ and the *early exercise region* is $S(t) \leq b(t)$. It is the reverse for a call.

The difficulty is valuing an American style option is that the early exercise strategy must be computed at the same time as the price. The price depends on the early exercise strategy, and conversely. In the early exercise region, clearly the price is equal to the intrinsic value. For a put, that is $f(s,t) = V(s)$ for $s \leq b(t)$. In the hold region, the hedging argument shows (think this though) that the price satisfies the Black Scholes PDE. We have seen (both computationally and theoretically in Assignment 5) that the European put price goes below the intrinsic value when the option is deep in the money. In fact, deep in the money, the put is similar to a forward contract to sell the stock for price K at time T, which is worth $Ke^{-rt} - S_0$, which is less than the intrinsic value $K - S_0$. A computational method for computing American option prices must have a mechanism that prevents the $f(s,t)$ from falling below $V(s)$.

The simplest mechanism for doing this is as follows. Suppose you have computed the (approximation to the) American price at time t_k in the form of numbers[3] $F_{jk} \approx f(x_j, t_k)$. We first use the finite difference formula (26) to compute tentative values at time t_{k-1}:

$$\widetilde{F}_{j,k-1} \;=\; aF_{j+1,k} + bF_{jk} + cF_{j+1,k} \;. \tag{32}$$

We then restore the necessary inequality by taking the max with the intrinsic value:

$$F_{j,k-1} \;=\; \max\left(\widetilde{F}_{j,k-1}, V(x_j)\right) \;. \tag{33}$$

The computed early exercise boundary is

$$b(t_k) \;\approx\; B_k \;=\; \max\left\{j \mid F_{jk} = V(x_j)\right\} \;.$$

This computational strategy may be understood from the point of view of the person making the decision of when to exercise. If she holds the option (and has not exercised already), at time t_{k-1}, she has to choose between exercising and not exercising. If she exercised, she gets value equal to the intrinsic value $V(x_j)$. If she holds, she gets (in the risk neutral world) expected value $\widetilde{F}_{j,k-1}$ given by (32). She choses the greater of these, which is the formula (33). This is explained more completely in the context of the binomial tree model in Section 8 of Kohn's notes.

[3] We go back and forth between x and s variables as is convenient.

It goes without saying that we would not focus entirely on computational methods for pricing American style options if there were an explicit formula like the Black Scholes formula. You also can imagine that we would not waste time talking about finite difference methods for European style options if that were out only target. We value vanilla European options using the Black Scholes formula, but almost any departure from that – early exercise features, volatility surfaces, other exotio features – puts us in a position where we must solve a PDE (or do Monte Carlo in extreme cases).