Finite Differences

We use an integrated carry-cost version of the Black-Scholes PDE

$$\frac{\partial F}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 F}{\partial S^2} - rF + cS\frac{\partial F}{\partial S} = 0 \tag{1}$$

along with the usual assumptions, and solve it by means of reducing it to the well-understood *diffusion equation*

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2}. \tag{2}$$

1. Changing Variables

A change of variable helps us create analytic solutions by transforming our PDE into one that is routine and familiar, and which has well-known methods of solution. Changing variables is useful for similar reasons when we apply finite difference methods, and also can improve the stability and computational cost of a scheme.

In the case of the Black-Scholes PDE (1), consider the change of variable

$$x = \log\left(\frac{S}{K}\right) \quad \tau = \frac{1}{2}\sigma^2 (T-t) \tag{3}$$
$$v = \frac{F}{K} \quad k_1 = \frac{r}{\frac{1}{2}\sigma^2} \quad k_2 = \frac{r-q}{\frac{1}{2}\sigma^2} = \frac{c}{\frac{1}{2}\sigma^2}.$$

It is worth noting that the new time parameter τ has the opposite sign from T. This will change the equation to a "forward time" rather than a "backward time" PDE[1]. We will timestep forward from $\tau = 0$ rather than (as with the tree) backward from τ, stopping at "time" $\tau_{\max}$, defined by

$$\tau_{\max} = \frac{1}{2}\sigma^2 T$$

(assuming that today is $t = 0$).

Our transformation gives

$$S = Ke^x \quad t = T - \frac{\tau}{\frac{1}{2}\sigma^2} \quad F = Kv(x,\tau)$$

[1]Changing the sign of time is an inevitable feature of solutions to the Black-Scholes PDE, and indeed in any diffusion-related financial options pricing scheme. In an actual diffusion, a known solution at time 0 is propagated forward (diffused) through time. Recall that our definition of a financial derivative postulates a *future* time T at which the derivative value is known exactly. Thus information about the value must propagate *backwards* to the present from that future time T.

so that

$$-rF = -rKv$$

$$\frac{\partial F}{\partial t} = \frac{\partial Kv}{\partial t} = \frac{\partial F}{\partial \tau}\frac{\partial \tau}{\partial t} = K \cdot \left(-\frac{1}{2}\sigma^2\right)\frac{\partial v}{\partial \tau}$$

$$\begin{aligned} cS\frac{\partial F}{\partial S} = cS\frac{\partial Kv}{\partial S} &= cKe^x\frac{\partial Kv}{\partial x}\frac{\partial x}{\partial S} = cKe^x \cdot K\frac{\partial v}{\partial x}\cdot\frac{1}{S} \\ &= cKe^x \cdot K\frac{\partial v}{\partial x}\cdot\frac{1}{Ke^x} = cK\frac{\partial v}{\partial x} \end{aligned}$$

$$\begin{aligned} \frac{1}{2}\sigma^2 S^2\frac{\partial^2 F}{\partial S^2} &= \frac{1}{2}\sigma^2\left(Ke^x\right)^2\frac{\partial}{\partial S}\left(e^{-x}\frac{\partial v}{\partial x}\right) \\ &= \frac{1}{2}\sigma^2\left(Ke^x\right)^2\frac{\partial}{\partial x}\left(e^{-x}\frac{\partial v}{\partial x}\right)\frac{\partial x}{\partial S} \\ &= \frac{1}{2}\sigma^2\left(Ke^x\right)^2\frac{1}{Ke^x}\left[-e^{-x}\frac{\partial v}{\partial x} + e^{-x}\frac{\partial^2 v}{\partial x^2}\right] \end{aligned}$$

yielding

$$K\left(-\frac{1}{2}\sigma^2\right)\frac{\partial v}{\partial \tau} - rKv + cK\frac{\partial v}{\partial x} + \frac{1}{2}\sigma^2 K\left[\frac{\partial^2 v}{\partial x^2} - \frac{\partial v}{\partial x}\right] = 0.$$

Divide both sides by $\frac{1}{2}\sigma^2 K$ to obtain

$$-\frac{\partial v}{\partial \tau} - k_1 v + (k_2 - 1)\frac{\partial v}{\partial x} + \frac{\partial^2 v}{\partial x^2} = 0$$

with the boundary condition

$$Kv = \max\left(Ke^x - K, 0\right) \qquad \text{for } t = T, \text{ i.e. } \tau = 0$$

implies

$$v\left(x, 0\right) = \max\left(e^x - 1, 0\right).$$

We have succeeded in making our PDE dimensionless–no parameter contains units of money, time, or anything else. Since the diffusion equation itself is dimensionless, this is progress. However, we need to go further to reduce our equation completely–after all there is still a $\frac{\partial v}{\partial x}$ term to eliminate. We will attempt to do so by multiplying both sides by exponentials, and trying to get them to cancel each other out. Define

$$u\left(x, \tau\right) = e^{-\beta_1 x - \beta_2 \tau} v\left(x, \tau\right)$$

or

$$v\left(x, \tau\right) = e^{\beta_1 x + \beta_2 \tau} u\left(x, \tau\right).$$

Our PDE is

$$\frac{\partial}{\partial \tau}\left(e^{\beta_1 x + \beta_2 \tau} u\right) = -k_1\left(e^{\beta_1 x + \beta_2 \tau} u\right) + (k_2 - 1)\frac{\partial}{\partial x}\left(e^{\beta_1 x + \beta_2 \tau} u\right) + \frac{\partial^2}{\partial x^2}\left(e^{\beta_1 x + \beta_2 \tau} u\right)$$

which gives

$$\beta_2 e^{\beta_1 x+\beta_2 \tau} u + e^{\beta_1 x+\beta_2 \tau}\frac{\partial u}{\partial \tau} = \frac{\partial}{\partial x}\left(\beta_1 e^{\beta_1 x+\beta_2 \tau} u + e^{\beta_1 x+\beta_2 \tau}\frac{\partial u}{\partial x}\right)$$
$$+ (k_2 - 1)\left(\beta_1 e^{\beta_1 x+\beta_2 \tau} u + e^{\beta_1 x+\beta_2 \tau}\frac{\partial u}{\partial x}\right) - k_1 e^{\beta_1 x+\beta_2 \tau} u$$

We compute the first right side term

$$\frac{\partial}{\partial x}\left(\beta_1 e^{\beta_1 x+\beta_2 \tau} u + e^{\beta_1 x+\beta_2 \tau}\frac{\partial u}{\partial x}\right) =$$
$$\beta_1{}^2 e^{\beta_1 x+\beta_2 \tau} u + \beta_1 e^{\beta_1 x+\beta_2 \tau}\frac{\partial u}{\partial x} + \beta_1 e^{\beta_1 x+\beta_2 \tau}\frac{\partial u}{\partial x} + e^{\beta_1 x+\beta_2 \tau}\frac{\partial^2 u}{\partial x^2}$$

to obtain

$$\beta_2 u + \frac{\partial u}{\partial \tau} = \beta_1{}^2 u + 2\beta_1\frac{\partial u}{\partial x} + \frac{\partial^2 u}{\partial x^2} + (k_2 - 1)\left(\beta_1 u + \frac{\partial u}{\partial x}\right) - k_1 u.$$

This suggests that we define

$$\beta_1 = -\frac{1}{2}(k_2 - 1)$$
$$\beta_2 = -\frac{1}{4}\left((k_2 - 1)^2 + 4k_1\right)$$

to be the constants solving the quadratic system

$$2\beta_1 + (k_2 - 1) = 0$$
$$\beta_1{}^2 + (k_2 - 1)\beta_1 - k_1 = \beta_2$$

which can be seen by expanding

$$\beta_1{}^2 + (k_2 - 1)\beta_1 - k_1 = \beta_1{}^2 - (2\beta_1)\beta_1 - k_1 = -\beta_1{}^2 - k_1 = \beta_2.$$

Having used our clever choice of β_1, β_2 we obtain the diffusion equation (2) with the initial boundary condition

$$u(x, 0) = \max\left(e^{\frac{1}{2}(k_2+1)x} - e^{\frac{1}{2}(k_2-1)x}, 0\right). \tag{4}$$

Often, of course, we are interested in the case of American options. Let

$$a(x) = e^{\frac{1}{2}(k_2+1)x} - e^{\frac{1}{2}(k_2-1)x}.$$

The boundary constraints transform from

$$V(S, t) \geq \max(S - K, 0)$$

to

$$u(x, \tau) \geq g(x, \tau)$$

where

$$g_C(x, \tau) = e^{\frac{1}{4}\left((k_2-1)^2+4k_1\right)\tau}\max(a(x), 0) \tag{5}$$

for calls, and

$$g_P(x, \tau) = e^{\frac{1}{4}\left((k_2-1)^2+4k_1\right)\tau}\max(-a(x), 0) \tag{6}$$

for puts. The change of coordinates remains essentially the same, with the value given by

$$V = Ke^{-\frac{1}{2}(k_2-1)x-\frac{1}{4}\left((k_2-1)^2+4k_1\right)\tau}u\left(x,\tau\right). \tag{7}$$

Note that although the transformation of the Black-Scholes PDE itself is always possible, it is *not* always possible to transform the boundary conditions g. Thus boundary conditions (such as those for Asian options) can sometimes prevent us from achieving a form as simple as this one.

2. Numerical Integration

Given a function $f(x)$ and an interval (a, b), the Fundamental Theorem of Calculus says that the area under $f(x)$ between a and b is given by the *antiderivative* F, and is equal to $F(b)-F(a)$. Since many of our options problems involve integrating a differential equation involving derivatives through time, we should think about how this helps us do so.

Consider the simplest such problem: position under constant acceleration. Here we have

$$\frac{\partial^2 x}{\partial t^2} = a$$

which we can solve directly to

$$v = \frac{\partial x}{\partial t} = v_0 + at$$

and

$$x = x_0 + v_0 t + \frac{1}{2}at^2.$$

This is possible because it was easy for us to compute the antiderivative of a constant a. But suppose it was not. How might we approximate the answer?

Let's begin by ignoring the "boundary conditions" that handle any initial position and velocity, by pretending these quantities are zero. We know that the definition of a derivative is

$$F'(t_0) = \lim_{h\to 0}\frac{F(t_0+h)-F(t_0)}{h}.$$

If we assume h is small, but finite, we can approximate

$$F'(t_0) \approx \frac{F(t_0+h)-F(t_0)}{h}$$

and rearrange terms to attain *Euler's Formula*

$$F(t_0+h) \approx F(t_0) + hF'(t_0). \tag{8}$$

Now when we work with options, or the acceleration problem above, we typically know the value of F at a particular time. And we know the derivative $f = F'$. What we don't know is values of F for times other than expiration (for an option) or initial time zero (for the acceleration problem).

So say we know $F(0)$. We can use the formula in Equation 8 to "bootstrap" ourselves into finding the value of F at any time T, by dividing $[0, T]$ into M subintervals of length h or smaller.

In the double-derivative case of acceleration, we might solve the problem with the following pseudocode:

```
position(t, M, a, x0,v0)
x=x0;
v=v0;
h=t/M;

for i=1 to M

    x=x+h * v;
    v=v + h * a;

return x;
```

So even *without* knowledge of what the actual position function is, we can get a pretty good approximation of it just by integrating values of its derivatives. This process of numerical integration in time is precisely what we want to imitate in the more complicated case of the diffusion PDE.

Consider briefly the problem of a time-varying acceleration $a(t)$. In this case, we really have two obvious, equally valid choices for updating the velocity v. First, we could take

$$v(t+h) = v(t) + a(t)$$

or, we could take

$$v(t+h) = v(t) + a(t+h).$$

In addition, we could take

$$v(t+h) = v(t) + a\left(\frac{1}{2}t + \frac{1}{2}(t+h)\right). \tag{9}$$

or even

$$v(t+h) = v(t) + \frac{1}{2}\left(a(t) + a(t+h)\right). \tag{10}$$

Since the acceleration a is supposed to represent the acceleration experience over the interval $[t, t+h]$, it is not surprising that the latter two formulas 9 and 10 give more accurate results. We will see later that, although the midpoint formula 9 is more appealing in the case of acceleration, it is 10 that has a direct analogue in finite difference techniques for pricing derivatives. As a matter of fact, we generalize by considering the situation where we take θ of the left-hand acceleration

value and $1-\theta$ of the right-hand value to obtain

$$v(t+h) = v(t) + \theta a(t) + (1-\theta)a(t+h). \tag{11}$$

Recall that the diffusion PDE is

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2}.$$

As with the acceleration example, we have a value for the time derivative $\frac{\partial u}{\partial \tau}$. Of course, that value is a little abstract, being in terms of a different derivative, but the idea remains the same: find $u(\tau)$ by adding

$$u(t+h) \approx u(t) + h\frac{\partial u}{\partial \tau} \tag{12}$$

lots of times. The only new concept we will see is that we need to use an approximation to $\frac{\partial^2 u}{\partial x^2}$, because we know initial values of u, but not initial values of its derivatives.

3. Difference Formulas

Recall that Euler's Formula

$$F(t_0+h) \approx F(t_0) + hF'(t_0)$$

arises from the fact that the derivative $F'(t_0)$ is close to the finite difference

$$\frac{F(t_0+h) - F(t_0)}{h}.$$

Often, we want to go in the other direction. That is, given some values of F we want to generate values for the first and second derivative F' and F''. To obtain the first derivative estimate, we could in theory use the approximation we have already seen

$$F'(x) \approx \frac{F(x+h) - F(x)}{h},$$

but we obtain a much more accurate[2] estimate by using the *central difference*

$$F'(x) \approx \frac{F(x+h) - F(x-h)}{2h}. \tag{13}$$

If we want to find a second derivative, we combine two neighboring first derivative estimates.

$$\begin{aligned} F''(x) &\approx \frac{F'(x) - F'(x-h)}{h} \\ &= \frac{F(x+h) - 2F(x) + F(x-h)}{h^2}. \end{aligned} \tag{14}$$

These difference formulas are surprisingly accurate, as can be seen in a table of them for the cosine function

[2]Second order, or $O(h^3)$ error

Finite differences and true derivatives of $\cos(x)$					
x	y	y'	y' (est)	y''	y'' (est)
0.07853	0.99691				
0.15707	0.98768	-0.11753	-0.11750	-0.98768	-0.98718
0.23561	0.97236	-0.19509	-0.19504	-0.97236	-0.97187
0.31415	0.95105	-0.27144	-0.27137	-0.95105	-0.95056
0.39269	0.92387	-0.34611	-0.34602	-0.92387	-0.92340
0.47123	0.89100	-0.41865	-0.41855	-0.89100	-0.89054
0.54977	0.85264	-0.48862	-0.48849	-0.85264	-0.85220
0.62831	0.80901	-0.55557	-0.55542	-0.80901	-0.80860
0.70685	0.76040	-0.61909	-0.61893	-0.76040	-0.76001
0.78539	0.70710	-0.67880	-0.67862	-0.70710	-0.70674
0.86393	0.64944	-0.73432	-0.73413	-0.64944	-0.64911
0.94247	0.58778	-0.78531	-0.78511	-0.58778	-0.58748
1.02101	0.52249	-0.83146	-0.83125	-0.52249	-0.52223
1.09955	0.45399	-0.87249	-0.87227	-0.45399	-0.45375
1.17809	0.38268	-0.90814	-0.90790	-0.38268	-0.38248
1.25663	0.30901	-0.93819	-0.93795	-0.30901	-0.30885
1.33517	0.23344	-0.96245	-0.96220	-0.23344	-0.23332
1.41371	0.15643	-0.98078	-0.98053	-0.15643	-0.15635
1.49225	0.07845	-0.99306	-0.99281	-0.07845	-0.07841
1.57079	0	-0.99922	-0.99897		

Note that, even though the x grid is relatively coarse, with jumps of about 0.08, the error in the first and second derivative estimates is far less than 0.01. This is typical for smooth functions.

3.1. **One-sided Estimates.** It is occasionally useful to obtain derivative approximation formulas that depend only on x values less than (or greater) than the point under consideration. We have already seen that the Euler Formula is a one-sided estimate for the first derivative, accurate to within error $O(h)$. We can use Taylor series to find estimates to higher orders, such as

$$F'(x) \approx \frac{3F(x) - 4F(x-h) + F(x-2h)}{2h} \tag{15}$$

and

$$F''(x) \approx \frac{2F(x) - 5F(x-h) + 4F(x-2h) - F(x-3h)}{h^2}. \tag{16}$$

both good to error $O(h^2)$.

4. Stability for Finite Difference Schemes

As we have seen, the essential idea of finite difference schemes is that a (partial) derivative can be successfully approximated by a difference formula, and that this can successfully be done many times in a row to approximate a solution to a differential equation. However, finite difference schemes for solving PDE's occasionally suffer from *instability*.

The small errors arising in various difference approximations can interact and magnify each other. When this happens, the solution "blows up", with wildly different values at neighboring grid points.

Instability can actually be useful, as a scheme can be selected to be stable only if it converges to the correct solution. Since it would be generally impossible to detect a scheme converging smoothly to the wrong solution, we need something obvious like instability to warn us of the situation.

In trees, instability manifests itself by negative probabilities.

5. Numerical Solution of the Diffusion PDE

Think of the problem of finding the value of an American option as one of solving a PDE with "free" boundary conditions. That is, the PDE is solved according to constraints that cannot be represented merely by taking a particular set of initial conditions and then timestepping merrily along. Instead, we have to check against early exercise conditions at each timestep. By using the conversion from a backward to a forward diffusion equation presented above, we can avoid the problem of backwardation in a tree (or instability in the case of finite difference methods).

First, we discretize space into a grid of $2N + 1$ points, spaced a distance dx apart,and convert the diffusion PDE into $2N + 1$ linked ODE's. This system is then solved by discretizing time into units of size $\delta\tau$ and numerically integrating according to Euler's Rule, just like our previous example of acceleration. We will denote our "true" solution by u and the approximation to it by v.

Since we are working on a grid, where the only allowed values of x and τ are integer multiple of dx and $\delta\tau$, we will find it convenient to define

$$v^m = v(x, m\delta\tau)$$
$$v_n = v(ndx, \tau)$$

and

$$v_n^m = v(ndx, m\delta\tau).$$

Once we use the change of variables in section 1, the main partial differential equation is the diffusion equation:

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2}. \tag{17}$$

Euler's Rule

$$v(x, \tau + \delta\tau) \approx v(x, \tau) + \delta\tau \cdot \frac{\partial u}{\partial \tau}(x, \tau) \tag{18}$$

leads to the time discretization for the left hand side of Equation 17

$$\begin{aligned}\frac{\partial u}{\partial \tau}(x,\tau) &\approx \frac{v(x,\tau+\delta\tau)-v(x,\tau)}{\delta\tau} \\ &= \frac{v(x,(m+1)\delta\tau)-v(x,m\delta\tau)}{\delta\tau} \\ &= \frac{v^{m+1}-v^m}{\delta\tau}\end{aligned} \tag{19}$$

We have seen in section 3 that the space discretization for the second derivative on the right hand side of Equation 17 must be

$$\begin{aligned}\frac{\partial^2 u}{\partial x^2}(x,\tau) &\approx \frac{\frac{\partial v}{\partial x}(x+dx,\tau)-\frac{\partial v}{\partial x}(x,\tau)}{dx} \\ &\approx \frac{\frac{v(x+dx,\tau)-v(x,\tau)}{dx}-\frac{v(x,\tau)-v(x-dx,\tau)}{dx}}{dx} \\ &= \frac{v(x+dx,\tau)-2v(x,\tau)+v(x-dx,\tau)}{dx^2} \\ &= \frac{v((n+1)dx,\tau)-2v(ndx,\tau)+v((n-1)dx,\tau)}{dx^2} \\ &= \frac{v_{n+1}-2v_n+v_{n-1}}{dx^2} \\ &= \frac{v_{n+1}-2v_n+v_{n-1}}{dx^2}.\end{aligned} \tag{20}$$

We can combine the two at a particular time to write a new version of Euler's Formula

$$v_n^{m+1} = v_n^m + \delta\tau \cdot \frac{v_{n+1}-2v_n+v_{n-1}}{dx^2}. \tag{21}$$

Note that we have not put superscripts on all the v's on the right hand side, because we have done nothing to decide what they must be. The obvious choice is to make them the most recently known values at timestep m

$$v_n^{m+1} = v_n^m + \delta\tau \cdot \frac{v_{n+1}^m-2v_n^m+v_{n-1}^m}{dx^2}, \tag{22}$$

which would lead to a discretization of Equation 17

$$\frac{v_n^{m+1}-v_n^m}{\delta\tau} = \frac{v_{n+1}^m-2v_n^m+v_{n-1}^m}{dx^2}. \tag{23}$$

There is no reason, though, that we can't assume the v's are taken from the *upcoming* values of v, which would be written

$$v_n^{m+1} = v_n^m + \delta\tau \cdot \frac{v_{n+1}^{m+1}-2v_n^{m+1}+v_{n-1}^{m+1}}{dx^2} \tag{24}$$

and lead to the discretization

$$\frac{v_n^{m+1}-v_n^m}{\delta\tau} = \frac{v_{n+1}^{m+1}-2v_n^{m+1}+v_{n-1}^{m+1}}{dx^2}. \tag{25}$$

To see how this works, consider the case where $N = 1$. We will have the equations

$$v_1^{m+1} = v_1^m + \delta\tau \cdot \frac{v_2^{m+1} - 2v_1^{m+1} + v_0^{m+1}}{dx^2}$$
$$v_0^{m+1} = v_0^m + \delta\tau \cdot \frac{v_1^{m+1} - 2v_0^{m+1} + v_{-1}^{m+1}}{dx^2}$$

and

$$v_{-1}^{m+1} = v_{-1}^m + \delta\tau \cdot \frac{v_0^{m+1} - 2v_{-1}^{m+1} + v_{-2}^{m+1}}{dx^2},$$

equivalent to

$$v_1^{m+1} - \delta\tau \cdot \frac{v_2^{m+1} - 2v_1^{m+1} + v_0^{m+1}}{dx^2} = v_1^m$$
$$v_0^{m+1} - \delta\tau \cdot \frac{v_1^{m+1} - 2v_0^{m+1} + v_{-1}^{m+1}}{dx^2} = v_0^m$$

and

$$v_{-1}^{m+1} + \delta\tau \cdot \frac{v_0^{m+1} - 2v_{-1}^{m+1} + v_{-2}^{m+1}}{dx^2} = v_{-1}^m$$

which are three linear equations in the three unknown quantities v_1^{m+1}, v_0^{m+1} and v_{-1}^{m+1}, and therefore solvable by matrix techniques.

Thus, upcoming, or forward, values of v can be used for estimating the second derivative. Indeed, any combination of the past values and upcoming values will work. Say we choose a parameter $\theta \in [0, 1]$ to indicate how much of each we are using. Effectively, we are saying

(26)
$$v_n^{m+1} = v_n^m + \delta\tau \cdot \left(\theta \frac{v_{n+1}^{m+1} - 2v_n^{m+1} + v_{n-1}^{m+1}}{dx^2} + (1-\theta) \frac{v_{n+1}^m - 2v_n^m + v_{n-1}^m}{dx^2} \right),$$

which can be written

(27)
$$\frac{\partial^2 u}{\partial x^2} = \theta \left(\frac{v_{n+1}^{m+1} - 2v_n^{m+1} + v_{n-1}^{m+1}}{dx^2} \right) + (1-\theta) \left(\frac{v_{n+1}^m - 2v_n^m + v_{n-1}^m}{dx^2} \right)$$

Defining the *aspect ratio* to be

(28)
$$\alpha = \frac{\delta\tau}{dx^2},$$

we obtain

(29)
$$v_n^{m+1} - \theta\alpha \left(v_{n+1}^{m+1} - 2v_n^{m+1} + v_{n-1}^{m+1} \right) = v_n^m + (1-\theta)\,\alpha \left(v_{n+1}^m - 2v_n^m + v_{n-1}^m \right)$$

This last form gathers all terms involving time $(m+1)\delta\tau$ on the left hand side, and all "known" terms involving earlier times on the right. Our goal is to solve it repeatedly until $m + 1 = M$.

We start with some initial conditions[3] that provide us with $\mathbf{v}^0$ and then to use the above equation (along with, say, early exercise conditions) to find $\mathbf{v}^m$ for $m > 0$. Note that unless $\theta = 0$ we have an equation with implicit terms which must be solved by indirect linear algebraic techniques for matrix algebra. Since equations with implicit terms always end up giving the best stability properties, we almost always end up doing a little matrix algebra.

We will take the solution at timestep $m\delta\tau$ to be $\mathbf{v}^m$, or the vector

$$\mathbf{v}^m = \mathbf{v}^{\mathbf{m}}_{\mathbf{-N+1}} v^m_{N-1}. \tag{30}$$

The exercise payoff constraint $\mathbf{g}^m$ will be given by

$$\mathbf{g}^m = \mathbf{g}^{\mathbf{m}}_{\mathbf{-N+1}} g^m_{N-1} \tag{31}$$

with

$$g^m_n = g(ndx, m\delta\tau) \tag{32}$$

where g is the continuous payoff constraint. For the American option, early exercise is a possibility at any time, so we know that for all m and n

$$v^m_n \geq g^m_n \tag{33}$$

which we can write more concisely in vector form as

$$\mathbf{v}^m - \mathbf{g}^m \geq 0. \tag{34}$$

Note that we don't include elements at the extrema $-N, N$ because they are determined by the boundary conditions.

Let

$$\alpha = \frac{\delta\tau}{dx^2}. \tag{35}$$

Represent the space derivative taken implicitly and numerically by the $2N-1$ square matrix given by

$$\mathbf{C} = \begin{pmatrix} 1+2\alpha\theta & -\alpha\theta & 0 & \cdots & 0 \\ -\alpha\theta & 1+2\alpha\theta & -\alpha\theta & & \vdots \\ 0 & -\alpha\theta & \ddots & \ddots & 0 \\ \vdots & & \ddots & 1+2\alpha\theta & -\alpha\theta \\ 0 & \cdots & 0 & -\alpha\theta & 1+2\alpha\theta \end{pmatrix} \tag{36}$$

In order to orient yourself to this, note that when $\theta = 0$ we have the explicit finite difference case and $\mathbf{C} = \mathbf{I}$. For the Crank-Nicolson case,

[3]Remember that our change of variables switches the "final" conditions at option expiry to initial conditions for the change of variables.

this is

$$\mathbf{C} = \begin{pmatrix} 1+\alpha & -\frac{1}{2}\alpha & 0 & \cdots & 0 \\ -\frac{1}{2}\alpha & 1+\alpha & -\frac{1}{2}\alpha & & \vdots \\ 0 & -\frac{1}{2}\alpha & \ddots & \ddots & 0 \\ \vdots & & \ddots & 1+\alpha & -\frac{1}{2}\alpha \\ 0 & \cdots & 0 & -\frac{1}{2}\alpha & 1+\alpha \end{pmatrix} \tag{37}$$

and for the implicit case, $\theta = 1$ and

$$\mathbf{C} = \begin{pmatrix} 1+2\alpha & -\alpha & 0 & \cdots & 0 \\ -\alpha & 1+2\alpha & -\alpha & & \vdots \\ 0 & -\alpha & \ddots & \ddots & 0 \\ \vdots & & \ddots & 1+2\alpha & -\alpha \\ 0 & \cdots & 0 & -\alpha & 1+2\alpha \end{pmatrix} \tag{38}$$

When α grows large, the off-diagonal elements will become just as large as elements on the diagonal. Numerical errors begin to appear in our solutions. For this reason aspect ratio values of α greater than 5 or 10 are impractical.

Let

$$\mathbf{b}^m = \mathbf{b}^{\mathbf{m}}_{\mathbf{-N+1}} b^m_{N-1} \tag{39}$$

$$\begin{aligned} b^m_n &= v^m_n + \alpha(1-\theta)\left(v^m_{n+1} - 2v^m_n + v^m_{n-1}\right) \\ b^m_{-N+1} &= v^m_{-N+1} + \alpha(1-\theta)\left(g^m_{-N} - 2v^m_{-N+1} + v^m_{-N+2}\right) + \alpha\theta g^{m+1}_{-N} \\ b^m_{N-1} &= v^m_{N-1} + \alpha(1-\theta)\left(g^m_N - 2v^m_{N-1} + v^m_{N-2}\right) + \alpha\theta g^{m+1}_N \end{aligned} \tag{40}$$

represent the value (estimated at time mdt) of holding the riskless option+hedge portfolio. Since the rate of return of such a portfolio cannot, by arbitrage arguments, exceed the risk-free rate of return, we can write that

$$\left(\frac{\partial u}{\partial \tau} - \frac{\partial^2 u}{\partial x^2}\right) \geq 0. \tag{41}$$

or

$$\begin{aligned} \frac{\partial u}{\partial \tau} - \frac{\partial^2 u}{\partial x^2} \approx \frac{v^{m+1} - v^m}{\delta\tau} \qquad & -\theta\left(\frac{v^{m+1}_{n+1} - 2v^{m+1}_n + v^{m+1}_{n-1}}{dx^2}\right) \\ & -(1-\theta)\left(\frac{v^m_{n+1} - 2v^m_n + v^m_{n-1}}{dx^2}\right) \\ \geq 0 & \end{aligned} \tag{42}$$

which yields

$$\mathbf{C}\mathbf{v}^{m+1} - \mathbf{b}^m \geq 0. \tag{43}$$

Now we know that at any given time $m\delta\tau$, either the option ought to be exercised, in which case $\mathbf{v}^{m+1} = \mathbf{g}^{m+1}$, or it ought not to be

exercised, in which case its value is determined by the diffusion equation, so that $\mathbf{C}\mathbf{v}^{m+1} = \mathbf{b}^{\mathbf{m}}$. Thus our problem can be stated in linear complementarity form

$$\left(\mathbf{v}^{m+1} - \mathbf{g}^{m+1}\right) \cdot \left(\mathbf{C}\mathbf{v}^{m+1} - \mathbf{b}^{\mathbf{m}}\right) = 0. \tag{44}$$

Using the dot product in this way represents our condition that every node either is or is not an early exercise. Note that (as in the case of vanilla options) if no early exercise is possible, the equations reduce simply to

$$\mathbf{C}\mathbf{v}^{m+1} - \mathbf{b}^{m} = 0. \tag{45}$$

which can be solved by any matrix equation solver, or (though it would be foolish in practice) by inverting the matrix $\mathbf{C}$.

If $\theta > 0$ then solving the linear complementarity formula can *not* be solved by first solving Equation 43 for $\mathbf{v}$ and subsequently checking for early exercise by requiring each vector element to satisfy $v_n^m \geq g_n^m$. This is a common oversight in finance texts and implementations, and it violates continuity of the first derivative of the solution. It tends to go unnoticed since the introduced error is fairly small.

Equations 34, 43, and 44 represent the matrix algebra problem we need to solve. Our boundary conditions are implicit in Equation 40. In general, we solve Equation 44 with a *projected successive over-relaxation* (PSOR) technique. The algorithm can be found in Wilmott, DeWynne, and Howison. Here is their example in pseudo-code:

```
function [v,cv,loops] = psor( v, cv, b, cvb, g, N,a,omega,eps,isAmer)
 loops=0;
 a2=0.5 * a;
 v[-N]=g[-N];
 v[N]=g[N];
 cv[-N]=g[-N];
 cv[N]=g[N];
 error = eps+eps;
 while (error>eps)
   error = 0;
   for n=1-N:1:N-1
     y=(b[n]+a2 * (v[n-1]+v[n+1]))/(1+a);
     y=v[n]+omega * (y-v[n]);
     cvy=(cvb[n]+a2 * (cv[n-1]+cv[n+1]))/(1+a);
     cvy=cv[n]+omega * (cvy-cv[n]);
     if (isAmer)
        y=max( g[n], y);
     error=error + (v[n]-y)  * (v[n]-y);
     v[n]=y;
     cv[n]=cvy;
  loops=loops+1;
 return;
```

PSOR is based on the Gauss-Siedel method, which itself is based on the Jacobi method. Essentially, the Jacobi method is an iterative

method that subtracts the first-order errors arising from substitution into Equation 44 at each step. The method stops when the first-order errors are deemed sufficiently small. Higher-order errors are unlikely to sum to anything larger. Gauss-Siedel makes the algorithm more efficient by using updated v values as soon as they are available, and PSOR attempts to extrapolate several Gauss-Siedel iterations from a single one using the *relaxation parameter* ω.

Other valid techniques include pivoting techniques and simplex methods.

As a practical matter, one can also efficiently solve Equation 45 by conjugate gradient techniques or by special types of LU decomposition, which tends to be superior for n-banded matrices C. Note that LU decomposition is not compatible with the early exercise projection in the true form Equation 44. The errors introduced by using LU decomposition here are nevertheless frequently tolerable.

LU decomposition works by taking advantage of the fact that matrix equations involving triangular matrices are easily solved. To be precise, assume we wish to solve

$$\mathbf{Cv} = \mathbf{b} \tag{46}$$

and that we can find matrices $\mathbf{L}$ and $\mathbf{U}$ which are respectively lower- and upper-triangular such that

$$\mathbf{C} = \mathbf{LU}. \tag{47}$$

Then we know that

$$\mathbf{L} \cdot \mathbf{Uv} = \mathbf{b}, \tag{48}$$

so first we solve the equation

$$\mathbf{Ly} = \mathbf{b} \tag{49}$$

and then we solve

$$\mathbf{Uv} = \mathbf{y}, \tag{50}$$

both of which are quite easy.

In PSOR, `omega` is a constant representing the relaxation parameter. In theory, the best value for it is obtained by *Chebyschev acceleration*, but practitioners usually just choose a value for it, often 1.0, expressing the oringial Gauss-Seidel algorithm.

6. Practical Considerations

6.1. **Convergence.** The θ-method schemes generally have total error $O(\delta\tau) + O(dx^2)$, except for $\theta = 1/2$ (Crank-Nicolson) which has error $O(\delta\tau^2) + O(dx^2)$. In principle this means that Crank-Nicolson should be run with very large values of the aspect ratio α. In practice, it becomes hard to solve the resulting matrix equations if α is too large, since the matrix C is no longer diagonally dominant. Thus, even for Crank-Nicolson, α is limited.

6.2. **Width.** It is necessary to choose the extent of the interval in x over which we will solve the PDE. Above, we chose to make that interval symmetric about the origin $x = 0$. This makes the notation simple, but might not be appropriate for options that are far from being at-the-money. The fix is obvious.

Less clear is just how wide that interval should be. Consider that we are modeling an initial value problem for the diffusion equation. Thus, there is a characteristic length

$$\ell = \sqrt{\tau_{\max}} = \frac{\sigma}{\sqrt{2}}\sqrt{T} \tag{51}$$

representing the propagation of information in a diffusion. Mathematically speaking, this is because the diffusion equation describes the evolution of the probability distribution for the position of a particle undergoing brownian motion. Since such a particle travels a net distance of about $\sqrt{t}$ in time t, we expect to find this constant as above.

Thus, we infer that a reasonable width is some multiple of ℓ. For example, to capture three standard deviations worth of information about the option at expiry, we would choose our interval to be $[-w\ell, w\ell]$ where the width $w = 3$.

6.3. **Parameters.** A tree scheme depends on only one (or a few) parameters to determine its shape–generally, this is expressed in the number of timesteps. Our finite difference scheme, on the other hand, has a long list of interdependent parameters: a width w, α, `eps` (ϵ^2), `omega` (ω), `N`, `M`, `dtau` ($\delta\tau$) and `dx` (dx). We need to choose reasonable values for them.

Note that we have the following relations:

$$\begin{aligned} \delta\tau &= \tau_{\max}/M \\ dx &= \sqrt{\delta\tau/\alpha} \\ N &= w/dx \end{aligned}$$

Also, ω will be chosen for us dynamically.

The parameter `eps` must necessarily be small enough that negligible extra errors are introduced by the `psor` scheme. Each timestep could introduce an error of size ϵ. There will be $M = O(1/\delta\tau)$ of them, and Crank-Nicolson has timestep errors of size about $\delta\tau^2$. Thus we want

$$\begin{aligned} M\epsilon &< \delta\tau^2 \\ \frac{\epsilon}{\delta\tau} &< \delta\tau^2 \\ \epsilon &< \delta\tau^3 \\ \epsilon^2 &< \delta\tau^6 \\ \texttt{eps} &< \delta\tau^6 \end{aligned}$$

Similar logic shows it should also be less[4] than δx^6.

We have discussed the aspect ratio α before. Practical limitations on computation time keep it from being larger than about 10, or smaller than 1/4. Within this range, experiment usually determines the optimal value.

Thus, by specifying only M, w, and α, we can completely determine the remaining parameters in our problem.

6.4. **Efficiency.** Now that we specify our solution in terms of M, w, and α, how efficient is it? Clearly, the computation time is linear in w. It is nonlinearly dependent on α, so we ignore that parameter. However, noting that PSOR or Gauss-Siedel converges roughly like a secant technique, we can assume it takes about $\log(N)$ steps to converge. Thus, each PSOR iteration takes time $N\log(N)$. Total calculation time is $O\left(MN\log(N)\right)$. Using the relations above, we then compute times

$$O\left(MN\log(N)\right)$$
$$O\left(M\frac{w}{dx}\log\left(\frac{w}{dx}\right)\right)$$
$$O\left(M\frac{1}{\sqrt{\delta\tau}}\log\left(\frac{1}{\sqrt{\delta\tau}}\right)\right)$$
$$O\left(M\frac{1}{\sqrt{1/M}}\log\left(\frac{1}{\sqrt{1/M}}\right)\right)$$
$$O\left(M\sqrt{M}\log\left(M\right)\right)$$

and error

$$O(\delta\tau)+O(dx^2)$$
$$O(\delta\tau)$$
$$O(1/M)$$

for an overall efficiency of approximately

$$O\left(\frac{1}{\sqrt{M}\log(M)}\right). \tag{52}$$

6.5. **Stability.** If $\theta<\frac{1}{2}$ then the numerical scheme contains too much explicitness. The result is that we must take $0<\alpha<\frac{1}{2}\left(1-\theta\right)$. This fact can be proven rigorously, by taking Taylor series approximations of the error at the spatial gridpoints, representing its evolution by a matrix and then showing the matrix has an eigenvalue λ with $|\lambda|>1$.

As a practical matter, values of α larger than about 10 start to make PSOR schemes work too hard to solve the matrix equation. The basic reason this is true can be seen from the C matrix. With a scheme containing significant implicitness, the value v_n^{m+1} depends on the values

[4]As a practical matter, it is usually legitimate to allow `eps` to be comparable in size to the *larger* of $\delta\tau^6$ and dx^6.

at its neighbors v_{n+1}^{m+1} and v_{n-1}^{m+1}. If α is large, this dependence is just as great as on the diagonal element v_n^m. Those neighbors in turn depend strongly on their own neighbors, and so on all the way to the boundary. So the set of equations (represented by the C matrix) we are trying to solve are highly interdependent. In contrast, as $\alpha \to 0$, the equations begin to depend very little on each other, and an iterative scheme may succeed nicely.

We will work almost exclusively with the Crank-Nicolson scheme, where $\theta = 1/2$. However, it is good to notice that an explicit scheme with $\theta = 0$ exactly corresponds to a trinomial tree. It is easy to see that the correspondence occurs with

$$\delta x = \log(u) \tag{53}$$

and

$$N = M \tag{54}$$

6.6. **Improving Accuracy.** Finite Difference Techniques are amenable to control variates, and *extrapolation*. In extrapolation, we make the assumption that the accuracy is an analytic function of the step size h. Then we can take the Taylor series

$$\begin{aligned} f(h) &= f(0) + f'(0)\ h + \frac{f''(0)}{2}\quad h^2 + O(h^3) \\ f(2h) &= f(0) + f'(0)2h + \frac{f''(0)}{2}(2h)^2 + O(h^3) \\ f(4h) &= f(0) + f'(0)4h + \frac{f''(0)}{2}(4h)^2 + O(h^3) \end{aligned}$$

which can be solved to obtain

$$f(0) \approx \frac{8f(h) - 6f(2h) + f(4h)}{3} \tag{55}$$

This technique can be dangerous for methods that are oscillating, or not yet in an analytically amenable region.

6.7. **Putting it together.** The steps involved in creating a finite difference solver for american options are many. Here they are in simple form:

- Verify the inputs, checking for negative prices and volatilities, etc.
- Based on the inputs, choose step sizes (or equivalently, counts) and other tolerances in a reasonable fashion.
- Change variables to the dimensionless set, and find parameters like k_1.
- Set up a grid of u values to correspond to a conceptual grid of x values
- Call the iteration routine to solve the matrix problem. This routine must

- Interpolate to find the solution at the particular x you need
- Change variables back to the original dollars.