Lyceum

Aristotle's Lyceum is the institution considered to be the forerunner of the modern university. Opened in 335 BC, the Lyceum was a center of study and research in both science and philosophy.

Solving Black-Scholes by Finite Differences

Discretizing BS

Last issue we saw how to set up a grid and how to discretize option prices to get the greeks. In this tutorial we'll see how to use all of this to create a very simple finite-difference algorithm for pricing vanilla options. The first step is to understand the equation we are solving. To start with, this is going to be the basic Black-Scholes equation, but we can still easily solve more complicated problems.

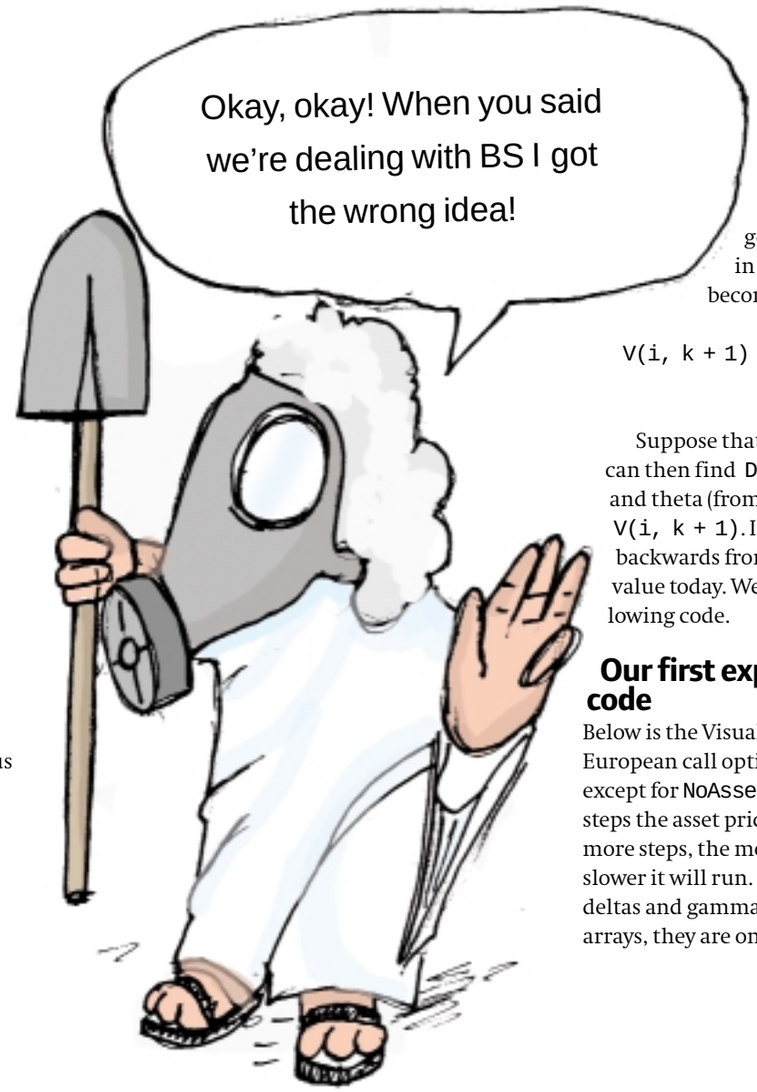
The Black-Scholes equation, written in terms of the greeks, is

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

The notation here is the standard. This equation is more usually written as a partial differential equation, but that's not strictly necessary, especially since we are trying to keep the jargon to a minimum.

To recap. from last week, we can write the greeks in terms of the option values at the various nodes as

```
Delta(i, k) = (V(i + 1, k) - _
    V(i - 1, k)) / dS / 2

Gamma(i, k) = (V(i + 1, k) - _
    2 * V(i, k) + V(i - 1, k)) _
    / dS / dS
```




```
Theta(i, k) = (V(i, k)
- V(i, k + 1)) / dt
```

The delta and gamma are going to appear in the algorithm in these forms but the theta will become

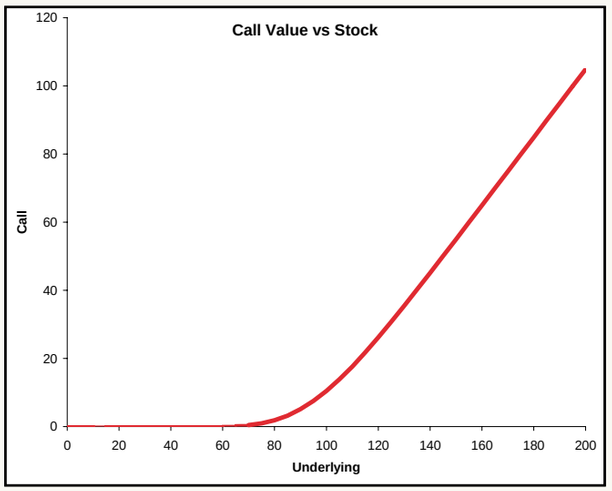
```
V(i, k + 1) = V(i, k) - _
        dt * Theta(i, k)
```

Suppose that we know `V(i, k)` for all `i` we can then find `Delta(i, k)` and `Gamma(i, k)` and theta (from Black-Scholes) and finally find `V(i, k + 1)`. In other words, we can work backwards from expiry to find the option value today. We'll see this happening in the following code.

Our first explicit finite-difference code

Below is the Visual Basic code for pricing a European call option. The notation is obvious except for `NoAssetSteps`. This is the number of steps the asset price grid is divided up into. The more steps, the more accurate the code, but the slower it will run. Since we won't be storing the deltas and gammas we don't need to make them arrays, they are only used on a temporary basis.

```
Function OptionValue(Asset As Double, Strike As Double, _
     Expiry As Double, Volatility As Double, Intrate As Double, _
     NoAssetSteps As Integer)

Dim VOld(0 To 100) As Double '(a)
Dim VNew(0 To 100) As Double '(a)
Dim S(0 To 100) As Double

dS = 2 * Strike / NoAssetSteps '(b)
NearestGridPt = Int(Asset / dS) '(c)
dummy = (Asset - NearestGridPt * dS) / dS '(c)
Timestep = dS * dS / Volatility / Volatility / (4 * Strike * Strike)
'(d)
NoTimesteps = Int(Expiry / Timestep) + 1 '(d)
Timestep = Expiry / NoTimesteps '(d)

For i = 0 To NoAssetSteps
S(i) = i * dS
VOld(i) = Application.Max(S(i) - Strike, 0) '(e)
Next i

For j = 1 To NoTimesteps '(f)

For i = 1 To NoAssetSteps - 1
Delta = (VOld(i + 1) - VOld(i - 1)) / (2 * dS) '(g)
Gamma = (VOld(i + 1) - 2 * VOld(i) + VOld(i - 1)) / (dS * dS) '(g)
VNew(i) = VOld(i) + Timestep * (0.5 * Volatility * Volatility * _
     S(i) * S(i) * Gamma + Intrate * S(i) * Delta - Intrate * VOld(i)) '(h)
Next i

VNew(0) = 0 '(i)
VNew(NoAssetSteps) = 2 * VNew(NoAssetSteps - 1) - VNew(NoAssetSteps - 2) '(i)

For i = 0 To NoAssetSteps
VOld(i) = VNew(i) '(j)
Next i

Next j '(f)

OptionValue = (1 - dummy) * VOld(NearestGridPt) + _
     dummy * VOld(NearestGridPt + 1)'(k)

End Function
```



Call option values produced by a simple finite-difference code

NOTES

(a) We only need two arrays for the option value. We update the old values at step (j)
(b) Setting up the asset step size
(c) The current asset value might not lie on a grid point so later we'll have to do some interpolating
(d) The timestep is chosen to make sure that the method is stable and that both today and expiry lie on grid points
(e) This is where the payoff is set up
(f) The timestepping loop starts at expiry and works back to the present
(g) Here's where we calculate delta and gamma
(h) This line combines the definition of theta with the Black-Scholes formula
(i) We know the value of a call when the asset is zero, it is also zero. We know that the gamma is zero for very large asset price
(j) Here's the updating
(k) This is a simple linear interpolation to find the option value at a point between two asset nodes

Exercise: This code is written just to price a call option. How would you modify it to price a put option?

Next issue we will extend the finite-difference method and examine how to price American options.