Option Valuation using Finite Differences

October 2015

TFG

FINANCIAL SYSTEMS

Option Valuation using Finite Differences

Martin Toyer, CTO, TFG Financial Systems.

Peter Russell, Team Lead, TFG Financial Systems

07 Oct 2015

Abstract

TFG has built a method to value options with a non-standard pay-out function and Knock-Out barriers. As this calculation is integrated into a real-time system, performance and stability are critical. This has been achieved by using implicit finite differences where more detailed grids are used when the estimate of the error exceeds the required tolerance.

All algorithms have been tuned to ensure that the performance of the calculation is acceptable: for example, a double Knock-Out window barrier digital option can be valued in less than one second and is accurate to the third decimal place.

The valuation is based upon the Black-Scholes PDE; however, to include the effect of volatility smiles, the Vanna-Volga 'rule of thumb' adjustment has been implemented.

Objective

We wish to value an option that has an upper and a lower barrier; neither of these barriers last as long as the option (window barriers). If the underlying rates don't touch either of these barriers, the option has a digital pay-off at expiry - the notional amount or nothing depending on the level of the underlying rate at expiry.

As we are valuing in real-time we need the algorithm to be as fast as possible. We also need a stable result as we are using the valuation model in various risk measurement contexts: Value at Risk, stress tests, as well as position level sensitivities.

Finite Differences

The Black-Scholes Partial Differential Equation (PDE) models how an asset value changes through time. As we know what the asset value is at expiry (the pay-out function) we want to use the PDE to 'rewind' time back to the valuation date. This is performed by converting the differential equation into a difference equation in two variables (the time and the exchange rate).

$$\frac{dv}{dt} + \frac{\sigma^2}{2}\frac{d^2v}{dx^2} + \left(r - r_f - \frac{\sigma^2}{2}\right)\frac{dv}{dx} = rv$$

Equation 1 - Black Scholes PDE

$$\frac{\Delta v}{\Delta t} = rv - \frac{\sigma^2}{2}\frac{\Delta^2 v}{\Delta x^2} - \left(r - r_f - \frac{\sigma^2}{2}\right)\frac{\Delta v}{\Delta x}$$

$$v(t - \Delta t, x) = v(t, x) + \Delta t\left(\frac{\sigma^2}{2}\frac{\Delta^2 v}{\Delta x^2} + \left(r - r_f - \frac{\sigma^2}{2}\right)\frac{\Delta v}{\Delta x} - rv\right)$$

Equation 2 - Black Scholes Difference Equation

If we have the value of the asset at time t we can determine the value of the asset at time $t - \Delta t$ by using the following finite difference relationships:

$$\frac{\Delta v}{\Delta x} = \frac{v(t, x + \Delta x) - v(t, x - \Delta x)}{2\Delta x}$$

$$\frac{\Delta^2 v}{\Delta x^2} = \frac{v(t, x + \Delta x) - 2v(t, x) + v(t, x - \Delta x)}{\Delta x^2}$$

Equation 3 - Mid-point finite differences

The simplest way of using these difference equations is by using the known asset value as of time t. This explicit method allows you to simply move back in time. Unfortunately, this method is known to be unstable as $\frac{\Delta x^2}{\Delta t}$ becomes small; so the time steps have to be very small; making for a very inefficient algorithm.

It is possible, however to construct a set of simultaneous equations in the unknown asset values as of time $t - \Delta t$. This implicit method does not suffer from the instability of the explicit method but is more complex: requiring the inversion of a matrix to solve the simultaneous equations. By examining the difference equations above, it is apparent that each point is only dependent upon its neighbours so the matrix will be tri-diagonal (i.e. it will have values on the diagonal and the two diagonal lines next to the main diagonal). The Thomas algorithm is the fastest method to solve these simultaneous equations as the main diagonal will dominate the off-diagonal element as the off-diagonal elements are multiplied by Δt.

The third method is to use a combination of the two approaches: with equal weight this is the Crank-Nicolson method. The error between the differential equation and the difference equation is smallest if you use this method. However, there is a problem with using any explicit values when you have a barrier: the explicit values will have the discontinuity whilst the implicit values will be the asset value before the effect of any barrier is included. Any derivative estimated from the known values will be badly estimated if it is a point adjacent to a barrier.

Therefore we have settled on using implicit finite differences starting with the asset value determined by the pay-off function and then stepping back in time to the valuation date.

Grid set up

To use the above relationships, we set up a grid of points separated by Δx in the asset price dimension and Δt in the time dimension.

As seen in the previous section, we need to be careful with barriers; we are only concerned with barriers at fixed levels, so we ensure that the barriers are points on the grid. That way the finite difference estimates for points adjacent to the barrier will not be affected by spurious interpolation across barriers.

Richardson Extrapolation

At the centre of our finite difference algorithm we use Richardson's Extrapolation technique. If you calculate a value using a grid sizes of Δx, Δt and then using $\frac{\Delta x}{k}, \frac{\Delta t}{k}$ you will get two estimates. The second estimate will obviously be better than the first. However, Richardson showed that you gain much more information. You can combine the two numbers to a third estimate which is even better than the second estimate and, more importantly still, gain an understanding of how accurate this value is.

To be more specific to our algorithm, the mid-point finite differences (Equation 3) that we are using allow us to approximate our value function at a time point as a quadratic function (i.e.: $v(x+\Delta x) = v(x) + \Delta x. v'(x) + \Delta x^2 \hat{} 2. v''(x) + o(\Delta x^3)$). Using Richardson extrapolation it can be shown that a better estimate would be the first estimate subtracted from k^3 times the second estimate all divided by (k^3-1).

The error on the second value can be estimated by the difference between the two estimates divided by (k^3-1); we use this value to determine whether we are 'close enough'.

Quadratic Interpolation

Once we have determined the value of the asset at each grid point we need to be able to interpolate between them. The finite difference estimation of the value of the asset is based on a quadratic function. However, this is only valid if the derivatives exist. When we are attempting to value near the discontinuities at barriers, we have to either use different points or use linear interpolation.

Adaptive Finite Differences

Putting all of these techniques together we can produce an adaptive implicit finite difference scheme where we can control the estimation error.

At each time step we calculate the asset value at each grid point. We then sub-divide the grid and calculate the asset value again using two time steps and halving the gap between each grid point on the asset axis. For each point on the original grid we have two estimates. We improve our estimates using Richardson's extrapolation technique and note which are outside our estimation tolerance. It is important to note that, when sub-dividing the original grid, we have to interpolate along the underlying rate axis to calculate the value of the asset at the previous time step (or the pay-off function if this is the first step): to do this we use quadratic interpolation ensuring that we don't cross over a discontinuity caused by a barrier.

For those points that are outside our estimation tolerance we repeat the sub-division of the grid with ever smaller step sizes until the error is small enough.

When performing the implicit finite difference step, you have to set boundary values (otherwise there are too many equations in the unknown grid points). This can be done by either setting the value at the end points or the first derivative. For the entire grid this is not very important: one can set the value at the end points to be the discounted value from the previous points; or to have the same gradient as the pay-off function.

However, when we are taking subsets of the grid and making finer grids to improve our estimate, we have to be much more careful. We include in our finer grid end-points for which a value has been successfully fitted and also match the gradient at those end-points.

After each time step the barrier conditions must be over-laid on the asset values calculated by the finite difference calculation. It is tempting to include these barrier conditions in the matrix calculation; however that would be a mistake as the derivatives of the asset value do not exist over the barrier and so the finite difference estimate is not valid.

Performance

By carefully implementing all of these sub-algorithms we have an algorithm that can be used to value an option with any pay-off function and Knock-Out barriers covering any sub-period of the lifetime of the option. Note that Knock-In options can be supported by valuing a combination of vanilla and Knock-Out options. More varied barrier functions were out of the scope of the project.

Paradoxically, the more carefully we fitted each time step the faster the total valuation could be performed. We were able (on a standard desktop computer) to produce the following results:

Option type	Closed Form	Finite Differences	Time (seconds)	Monte Carlo
Vanilla Call	0.5289070	0.5289380	0.0002670	0.5287900
Knock Out Call	0.1024500	0.1018110	0.0071310	0.1025450
Digital Call	0.7131990	0.7159720	0.0272600	0.7138180
Digital Knock-Out Call	0.6006760	0.6055180	0.0459000	0.6025470
Double Knock-Out Call	0.0705420	0.0702471	0.0183230	0.0709906
Digital Double (window) Knock-Out		0.0788448	0.0608190	0.0788635

Other considerations:

There are a number of smaller items that we had to consider whilst working through the implementation of the algorithm.

Payment currency

For digital options the payment can be in the 'natural' currency: the domestic currency. If the payment currency is foreign currency we need to alter the payoff function by converting it (using the expiry rate) into the natural currency.

Monte Carlo valuation

For many of our tests we used standard closed-form valuations to compare against. However, we do not know of any closed form solutions for window barrier options so we developed a simple Monte Carlo algorithm.

One of the key advantages of using a Monte Carlo algorithm is that you can estimate your error as you progress: and also estimate how many more simulations are needed to get you close to the accuracy required.

We tested our Monte Carlo algorithm against the same closed form models as the finite difference algorithm in order that when we are test using a trade that does not have a closed form valuation formula and find that the two simulation approaches agree we confidence in the result.

Vanna-Volga 'rule of thumb'

To handle the effect of implied volatility smiles when we are using models based on Black-Scholes assumptions, we use the Vanna-Volga method. Although this is not a rigorous method it is often used in the market place.

The Vanna Volga method adds to the Black-Scholes valuation an adjustment which considers the cost of hedging the position's Vanna and Volga exposure. These adjustments are calculated by determining the Vanna and Volga of the position (with Black-Scholes assumptions) and then determining a portfolio built of three vanilla positions (one at the money, one 25-delta call, one 75-delta call) which have the required Vanna and Volga, but which does not have any exposure to Vega.

This value of this portfolio is then weighted by the survival probability of the option as, if the option were to become worthless due the underlying rate hitting a barrier, the hedge portfolio would not be necessary. We estimate this probability by using the barriers from the trade and combining them with a pay-off which provides an amount that when discounted is worth 1 on the valuation date whatever the underlying rate is.

Greeks

To determine the sensitivity of the valuation to changes in the underlying (i.e. delta and gamma), we can interpolate along the final set of estimates. This is preferable to re-running the whole calculation as it should be wasted calculation time. If the error tolerance is set low enough (we currently use 1e-6), re-running the algorithm with a different grid will not alter the result.

When calculating vega, the grid does not have to change but all of the valuations associated with it will.