5 April 2020
Cuttaz Armand
OUASFI Amine
HAJJAM EL HASSANI Mehdi
LABRIRHLI Badr

QuantLib project 3

Add greeks to binomial tree engines

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Contents

1 Introduction

This project aims at improving the computation of Greeks on binomial trees. Currently, QuantLib 's documentation cannot calculate the Greeks at the first step of a binomial tree. That's what we intend to fix in our work. To that end, we will first provide the main ideas that enable the computation. Then, we will show how the code has been changed accordingly. Finally, the results of the modifications will be highlighted.

2 Theory

We need to consider to kind of trees : equal jumps trees and equal probability trees.

2.1 Equal jumps trees

In the case of equal jump trees, the velue of the underlying has a probability p_{up} to go *up* and p_{down} to go *down*. Its value can be calculated with the following formula:

$$S_{n_{step},n_{up}} = S_0.\exp^{(2n_{up}-n_{step})\Delta.x}$$

Therefore :

$$S_{2,1} = S_0$$

Thus, the main idea is to add 2 steps to the tree, add 2Δ to the maturity and consider that the tree starts at step 2. Now that we have got 3 nodes at the start of the tree, we can compute the greeks as it was done before at step 2.

2.2 Equal probabilities trees

In this case,

$$S_{n_{step},n_{up}} = S_0.\exp^{(2n_{up}-n_{step})\cdot n_{up}}\exp^{n_{step}\cdot dps}$$

The idea here is to build a tree S' so that $S'_{2,1} = S_0$, namely :

$$S_{2,1} = S_0.\exp^{(2\times 1-2)}\exp^{2\cdot dps} = S_0.\exp^{2\cdot dps} \tag{1}$$

Then by letting,

$$S'_0 = S_0.\exp^{-2\cdot dps}$$

we can proceed as in the first case.

3 Code changes

The main modifications take place in the class `binomialtree`. First, the function `Size` now returns $i+3$ since there are two more nodes at each step.

```
Size size(Size i) const {
            return i+3;
```

Then, we have to change the calculation of the underlying for both `EqualJumpsTree` and `EqualProbabilitiesTree` methods. Since there are two more steps in the tree, we need to increase the number of steps in the calculation of j by 2 .

```
Real underlying(Size i, Size index) const {
            BigInteger j = 2*BigInteger(index) - BigInteger(i)-BigInteger(2);
```

We have also changed the `binomialengine` class to compute the greeks at step 0.

```
option.rollback(grid[0]);
        Array va0(option.values());
        QL_ENSURE(va0.size() == 3, "Expect nodes in grid at second step");
        Real p0u = va0[2]; // up
        Real p0m = va0[1]; // mid
        Real p0d = va0[0]; // down (low)
        Real s0u = lattice->underlying(0, 2); // up price
        Real s0m = lattice->underlying(0, 1); // middle price
        Real s0d = lattice->underlying(0, 0); // down (low) price

         // calculate gamma by taking the first derivate of the two deltas
        Real delta0u = (p0u - p0m)/(s0u-s0m);
        Real delta0d = (p0m-p0d)/(s0m-s0d);
        Real delta0 = (delta0u+delta0d)/2;
        Real gamma0 = (delta0u - delta0d) / ((s0u-s0d)/2);


         // Store results
        results_.value = p0m;
        results_.delta = delta0;
        results_.gamma = gamma0;
        results_.theta = blackScholesTheta(process_,
                                          results_.value,
                                          results_.delta,
                                          results_.gamma);
    }
```

Finally, the main class is coded so that it can be timed. We will analyse the results in the next section.

In order to further improve the derivatives, an idea would be to use formulas that take into account the asymmetry of the values. An interesting way to do this is to resort to a finite difference formulae. However, this not straightforward in our case since the computations need to be made on unequal subintervals. Here we followed a method proposed in [1] that addresses this problem by deriving a Three Point Finite difference formulae from the Lagrangian interpolation of the considered function f . This means to interpolate f using three points (x_0 x_1, x_2) and then differentiate this interpolation with respect to each point. . Since we want to compute the derivatives with regard to the middle point , the more appropriate equation to use for the calculation of the first derivative is the equation 8(b) in [1].

$$f'(x_0) = -\frac{h_2}{h_1(h_1+h_2)}f_0 - \frac{h_1-h_2}{h_1h_2}f_1 - \frac{h_1}{h_1(h_1+h_2)}f_2$$

Here f_i is the value of f at the ith interpolating point. The interval $[x_0, x_n]$ is divided into n subintervals of unequal widths h_1 ,h_2 , ...,h_n such that $x_n = x_0 + \sum_{i=1}^{n} h_i$.

A second Differentiation produces difference formula for second derivative as

$$f''(x_0) = -\frac{2[h_2f_0 - (h_1+h_2)f_1 + h_1f_2]}{h_1h_2(h_1+h_2)}$$

This leads to the following delta and gamma calculations.

```
// option values (p0) at this point
        option.rollback(grid[0]);
        Array va0(option.values());
        QL_ENSURE(va0.size() == 3, "Expect 3 nodes in grid at second step");
        Real p0u = va0[2]; // up
        Real p0m = va0[1]; // mid
        Real p0d = va0[0]; // down (low)
        Real s0u = lattice->underlying(0, 2); // up price
        Real s0m = lattice->underlying(0, 1); // middle price
        Real s0d = lattice->underlying(0, 0); // down (low) price

        // calculate gamma by taking the first derivative of the two deltas
        /* Real delta0u = (p0u - p0m)/(s0u-s0m);
        Real delta0d = (p0m-p0d)/(s0m-s0d);
        Real gamma0 = (delta0u - delta0d) / ((s0u-s0d)/2);
        Real delta0 = (delta0u + delta0d)/2; */

        Real h2 = s0u - s0m;
        Real h1 = s0m - s0d;
        Real f0 = p0d;
        Real f1 = p0m;
        Real f2 = p0u;

        Real gamma0 = 2*(h2*f0 - (h1+h2)*f1 + h1*f2)/((h1*h2)*(h1+h2));
        Real delta0 = (-h2/(h1*(h1+h2)))*f0 - ((h1-h2)/(h1*h2))*f1 +
         ↪  (h1/(h2*(h1+h2)))*f2;
```

4 Results

Concerning the performance of the engine, we expect our modification to have no noticeable changes. Given the number of time steps (801) , adding 2 extra nodes per step only generates 1602 nodes more whereas the tree contains $801 \times 802/2 = 321201$of them. An increase of approximately $0,5\%$ of the number of nodes shouldn't cause a significant drop of the engine's performance.

The appendix display the performance for each code. As expected, the changes slightly extended the computation time. However, nothing significant. Finally, it should be noted that the improvements the additional improvements of the delta calculations increased slightly the precision. The following table presents the results for deltas computation.

Table 1: **Results for deltas computation**.

Type	**Former code**	**New code**	**Difference**
Cox Ross Rubinstein	0.008005	0.008007	$2e^{-06}$
Trigeorgis	0.007844	0.008267	0.000423
Jarrow Rudd	0.007844	0.007653	0.000191
Additive EQP	0.00761	0.008207	0.000597
Tian	0.008608	0.009503	0.000895
Leisen Reimer	0.008411	0.008971	0.00056
Joshi	0.007957	0.008706	0.000749