ISyE 6767 Homework 3

Yuliang Li GTID# 903012703

Project Obeject

In this project, we use binomial tree method to price some American option on an ETF. Some parameters are like,

initial spot: 106.14
dividend yield: 1.6%
implied volatility: 22.72%
historical volatility: 12.66%
risk free rate: 1%
time to expiration: 355 days
strike price: 100, 106 ,107, 110

We price options under both implied and historical volatility with the four strikes. Before price the option, we use Black Sholes Model to calibrate the binomial tree.

Project Structure

Main.cpp: the main file, complete calibration and pricing. Input the files and output the option value and the delta of the option to csv files.
BSE.h and **BSE.cpp:** Black Sholes Model to price European Option, written in the previous homework.
NormDistIntegral.h and **NormDistIntegral.cpp**: generate a norm distribution, also writtedn in the previous homework.
BinomialTree.h and **BinomialTree.cpp**: generate binomial tree. There are two inherited classes, one is Stock_BiTree described stock tree, and the other one is Option_BiTree described option tree.
OptionPay.h and **OptionPay.cpp**: generate a class calculate the payoff of the option at a certain spot price.
OptionPrice.h: contains two template functions, and both can price European option and American option.

double pricer(Stock_BiTree& st, Option_BiTree& ot, T pay): the regular binomial tree to price options.

double pricer_smooth(Stock_BiTree& st, Option_BiTree& ot, T pay): the smooth method to construct the binomial tree to price, which means the option price in second last row determined by Black Scholes Model instead of backward

from the last row. This method can speed the convergence of the tree.
BSCalibra.h and **BSCalibra.cpp**: calibrate the binomial tree with Black Sholes Model. Use both methods to price European option, whose underlying price is between 100 and 110, and the volatility is between 0.10 and 0.30. In our files, the strike price is set as 100.0.
Greek.h and **Greek.cpp**: contains some functions to calculate the greeks of option. In this home work we only need the function to calculate delta.
IO.h and IO.cpp: input file contains parameters for pricing and output the results to csv files.

Project implementation

Binomial tree

To construct the option tree easily, we set the Stock_BiTree as the friend class of Option_BiTree, so the latter class can visit the former class's private member variables.
In the class and its inherited classes, there are some functions useful, including to get one certain data, to get a row of data, to get the whole data, to set the value of a certain position, a row and the whole data.

Price the option

In the project, we use the smooth binomial tree method to price options. The second last row's option value is determined by Black Scholes model.
The code is like,

```
//smooth method, option prices at second last date determined by Black-Scholes Model, increase the speed of convergence
template <class T>
double pricer_smooth(Stock_BiTree& st, Option_BiTree& ot, T& pay)
{
	int N = st.deep();
	std::vector<double> lastpay = st.Rowshow(N);
	int K = lastpay.size();
	for(int i = 0; i < K; i++)
	{
		lastpay[i] = pay(lastpay[i]);
	}
	ot.setdata(lastpay, N);
	// BS Model get the second last continuous value of Option
	double ttoexpir = ot.expiry - ot.delta_t;
	BS_EuroOption EuroBS(ot.strike, ot.rate, ot.dividend, ot.vol, ot.expiry, ot.call, ttoexpir);
```

```
    std::vector<double> seclastpay(K-1); // get the second last row of option value
    for(int i = 0; i < K-1; i++)
    {
        seclastpay[i] = EuroBS(st.Datashow(N-1, i));
    }

    if(ot.euro == true)
    {
        ot.backward_fill(seclastpay);
    }
    else
    {
        // construct a tree which store the intrinsic value of the American option
        BinomialTree in_value(N-2, 0.0);
        in_value.zeros();
        int in_N = in_value.deep();
        for(int i = 0; i <= in_N; i++)
        {
            for(int j = 0; j <= i; j++)
            {
                in_value.setdata(pay(st.Datashow(i, j)), i, j);
            }
        }
        std::vector<std::vector<double> > intrin = in_value.Datashow();
        ot.backward_fill(seclastpay, intrin);
    }

    return ot.get_root();
}
```

Calibrate the tree

Price some European options with Black Sholes model and binomial tree model. To speed the process to get the optimal deep of tree, each loop we double the test deep. When the difference of the price calculated by Black Sholes Model and binomial tree model is less than 0.05, we consider it completes the calibration.

Delta

To get the delta of option at time t, we use the formula below,

$$Delta_i^t = \frac{O_{i+1}^t - O_i^t}{S_{i+1}^t - S_i^t}$$

where O_i^t means the option price at time t on the i-th position of the row, and S_i^t is the stock price at the same position.

Results

Optimal deep of tree

The optimal deep of the binomial tree is 32 under the condition we set.

Option value

The American option values with two volatilities and four strike prices are shown in Table.1.

Table 1 American option value

volatility	strike price	option value
0.2272	100	6.62231
0.2272	106	9.60529
0.2272	107	10.1608
0.2272	110	11.9221
0.1266	100	2.80111
0.1266	106	5.46676
0.1266	107	6.01545
0.1266	110	7.83235

Delta

Fig.1 shows the delta of option with strike price as 106 and volatility as 22.72%

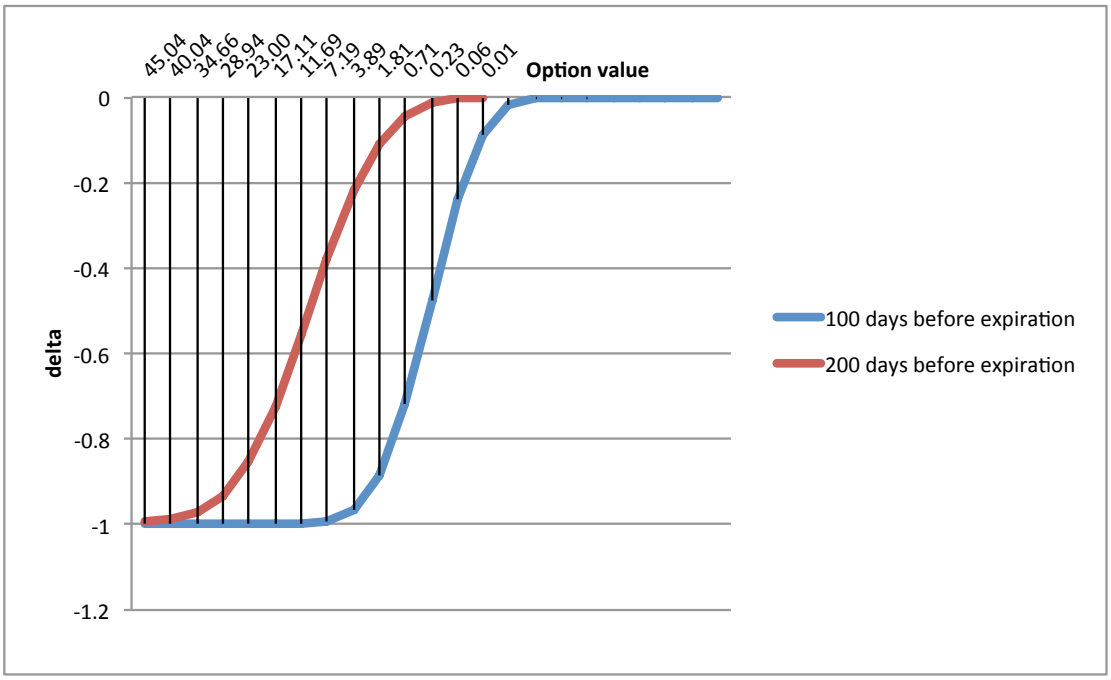

Fig. 1 Delta of option when strike price is 106 and volatility is 22.72%

Further work

1. The data in class BinomialTree are set as public for convenience, so it is lack of protection. Maybe need to find a more appropriate way to protect it. One of the ways I think is to set the Option_BiTree's data individually, then we can use it without distubing on the mother class.