# Break-even Volatility Project

Nicolas LECONTE – Ruiying LI

Our work is based on the slides of Dupire, which can be found at the following address: <http://www.stagirit.org/sites/default/files/articles/a_0369_cboe-bruno-2015.pdf>

## Presentation of our code

### Working with time series

In order to manipulate time series, we created a time\_series class. It primarily consists in two vector data members: one for dates (std::tm format) and one for values (double). Using vectors would allow us to be efficient when computing the volatilities, as we will only use the values vector.

The first thing was to build a few functions to manipulate .csv datafile, as well as a method of time\_series to load the data from a .csv file. We first compute the size of the file, then resize the vectors and fill them up. It seemed more efficient than using push\_back.

To facilitate the navigation through the time\_series, we built several methods, such as operator[] overloadings or approx\_index which finds the closest date. Functions for manipulating std::tm structures can be found in the functions.cpp file. Important: the indexing of the time\_series is in base 1, to be consistent with the Excel file.

### Computing Break-even volatility

Those computations will rely on a hedged\_ptf class. It would itself have a time\_series data member, containing the underlying’s data. The interest rate is constant, but we could imagine building an inheriting class using another time\_series for finding the relevant interest rate.

We decided not to include dividends at this stage. Doing so would add too many complexifications: treating the dividends data for the index, discounting all dividends to compute a dividend yield, and reinvest dividends in the auto-financing portfolio.

We set up the hedged\_ptf: time window using m\_start and m\_end, strike, rate. Spot and Maturities are based on the time window. Then, we can compute either the P&L for a given volatility or the implied volatility (get\_implied\_vol method), ie. the volatility that cancels the P&L.

We developed three P&L computation methods:

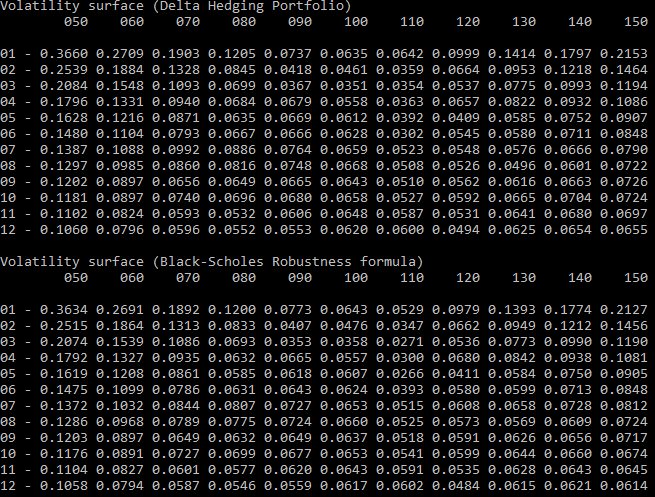
1. **get\_pnl** computes the P&L for an auto-financing portfolio which delta hedges daily a short option position. The cash difference between the portfolio’s value and the delta position is invested overnight at the constant rate.
2. **get\_delta\_pnl** computes the P&L without consideration for the cash difference, as suggested in the reference slides p.31. Under certain circumstances when the rate is non-null, the resulting P&L is constant in sign for any volatility. Therefore, we will not use this method for estimating the breakeven volatility.
3. **get\_robust\_pnl** computes the P&L using the so-called "Black-Scholes Robustness formula” (reference slides p.32). The user can choose between this method or the delta-hedging one when computing the implied volatility with a boolean parameter.

Comments in the code explain further the different steps of each algorithm.

### Producing a complete volatility surface

The manipulation of the volatility surface is done through a third class called vol\_surface. It has a pointer on a hedged\_ptf instance, from which it will compute the implied volatilities. The volatilities are stored in a vectorized matrix, as we have seen in class.

Volatilities are computed with the load\_vol\_surface method. Then we can easily print either the whole surface (print\_vol\_surface) or parts of it (a term structure with print\_strike, or a skew with print\_maturity). Finally, we can export the results to a .csv file using the export\_to\_csv method.

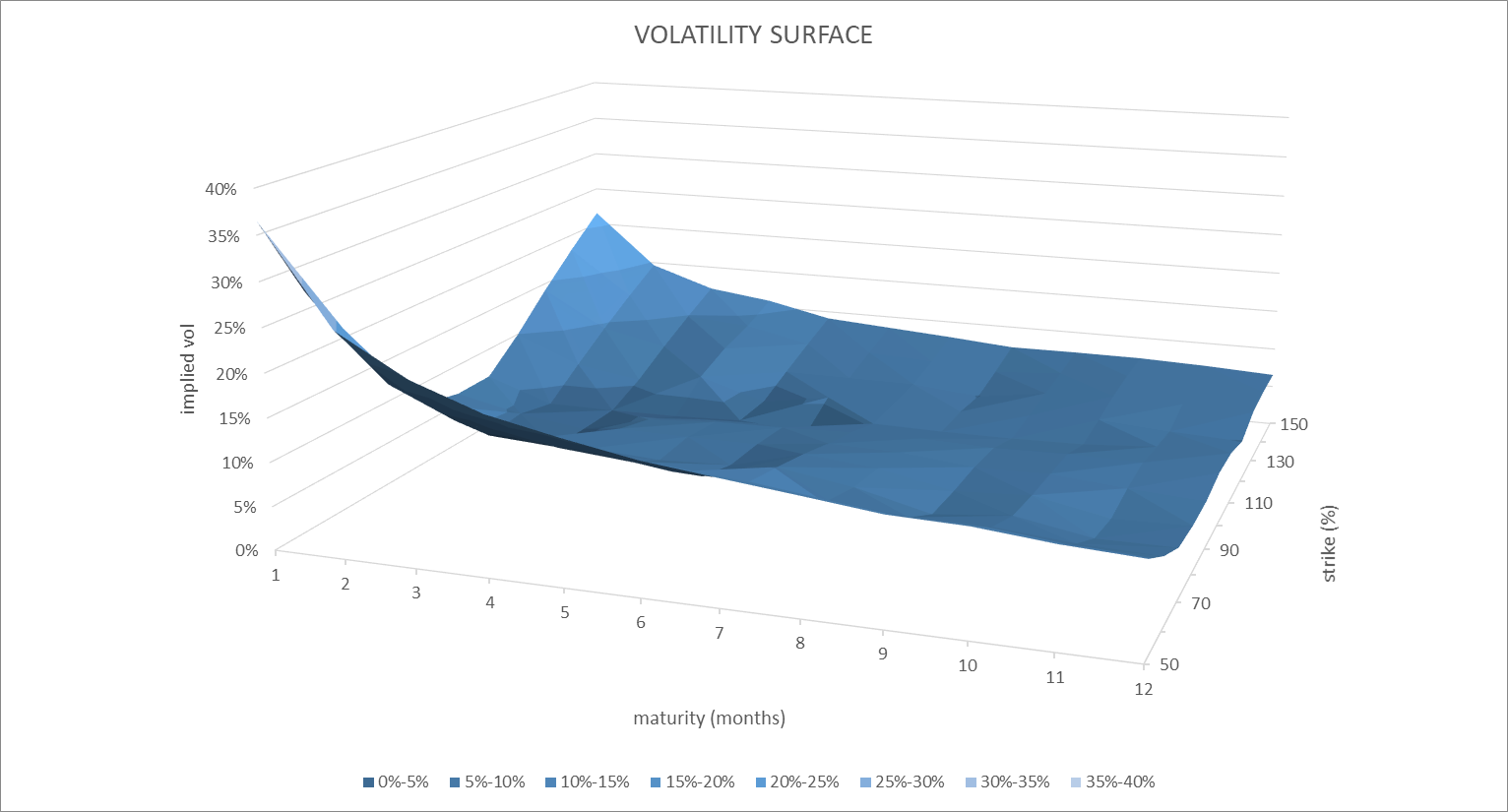


## Commenting our results

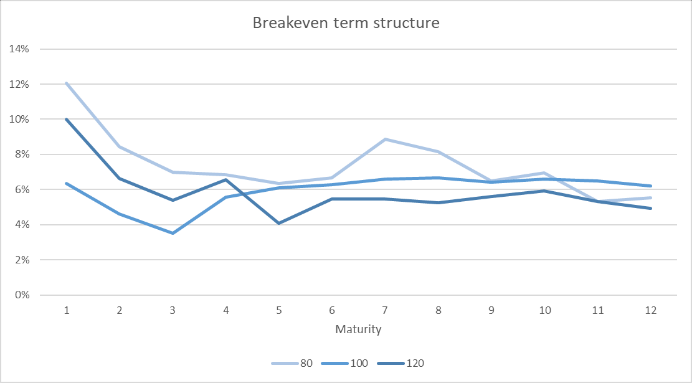
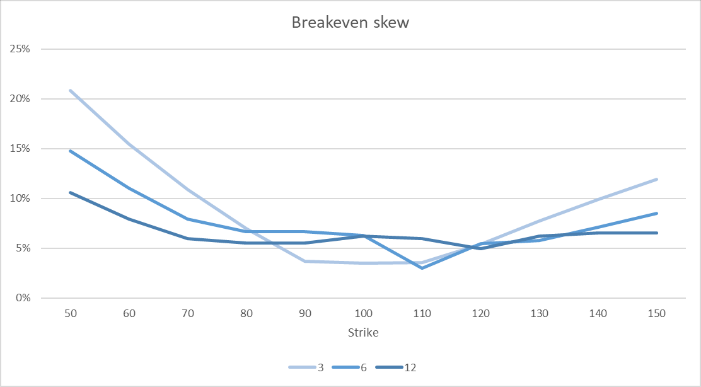
### Standard Breakeven volatility

We work on the historical closing prices of the S&P 500 index, with data ranging from 02/01/2015 to 12/12/2017. As it is USD-denominated, we chose a default risk-free rate of 1%.

We obtain the following shape of volatility surface:

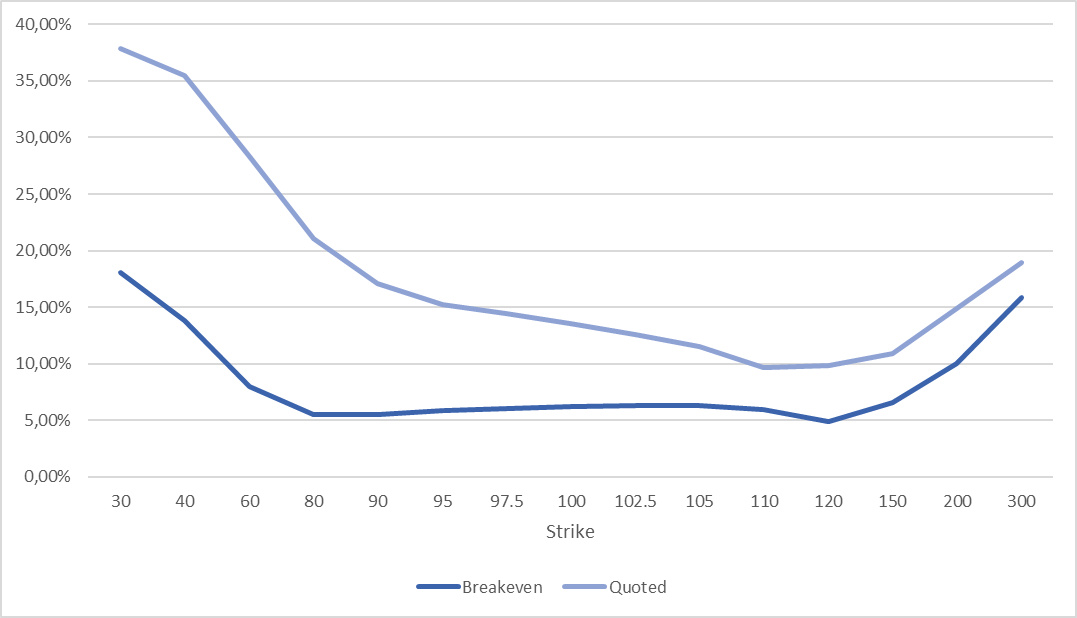


This is coherent with what we expect from a volatility surface obtained from quoted prices. There is an apparent skew, and a decreasing term structure.



When we compare the quoted 1-year skew with our results, we notice several things.

* First, we obtain a similar shape, but way below. This is probably due to dividends, and market participants requiring a higher price because they sell an insurance.
* Second, we obtain a very low volatility for low strikes, compared to the market. This is because market participants price



However, if we believe that volatility is overpriced, we can take advantage of the difference by selling volatility (ie. selling options) and performing a daily delta hedging. This strategy should generally yield a positive P&L.

### Comparing with robust P&L method

The “Black-Scholes Robustness formula” basically weights realized volatility events by the gamma, which is path dependent. Hence, when the option is around the money close to maturity (ie. high gamma), the implied volatility cancelling the P&L is higher, and so should be our breakeven volatility. Otherwise, it should be lower as it assumes continuous delta hedging and null interest rates.

When we observe the evolution of the S&P over the past year, we see that it has a strong upward trend. Below is a heatmap of the difference between implied volatilities obtained with the standard method, and with the robust (gamma) method.



We visualize that OTM calls obtain higher implied volatilities with the gamma method, compared to ITM calls, which is consistent with our guess. A thumb rule to check this is to see that gross (non-annualized) returns of the S&P over the past 6-12 months are indeed around 10%-20%.

### Limits to our approach and further improvements

A first limit to our results is that the S&P index excludes dividends. Using a total return index or incorporating dividends in the computations may yield better results.

The results are very dependent on the time window, ie. the past returns. A way to avoid this would be to proceed by averaging volatilities or P&Ls over different time windows, as suggested in the reference slides p.37. This could be implemented with the shift\_days method of time\_series class.