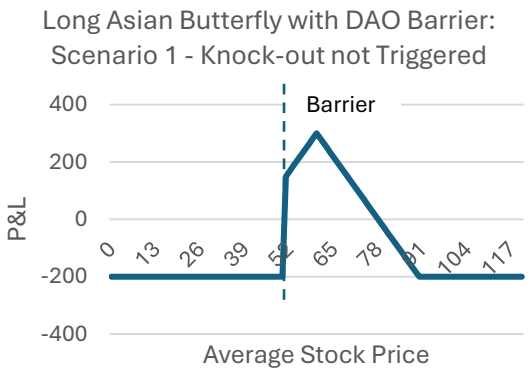
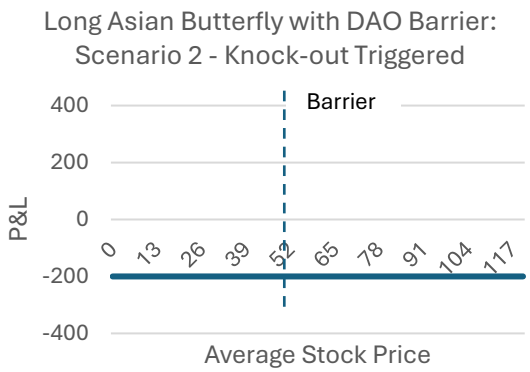
C++ in Quantitative Finance: Individual Home Project

Adam Foster

The project entailed the creation and pricing of a long call butterfly option strategy for arithmetic Asian options with a down-and-out barrier in C++ and subsequent analysis in R.

The strategy consisted of one long ITM call, two short ATM calls, and one long OTM call – all with the same time to maturity. The payoff of such a structure looks as follows.





The two long calls are offset by the two short calls in such a way that there exists a window of earning positive P&L around the short ATM call strike price, thus this is for investors who anticipate no major upward or downward movements in underlying price. Arithmetic Asian options consider the average underlying price over time against the strike price. The volatility of the average price is lower than the volatility of the spot price, making this type of option cheaper. Down-and-out barriers knock the contract out when the underlying price decreases below a specified threshold. This feature makes it less likely to produce a positive payoff, also making the option cheaper.

Several parameters and assumptions were embedded in the pricing of the portfolio.

- Price of the underlying at the moment of option pricing $S_0 = 115$
- Annualized volatility rate $\sigma = 25\%$
- Annualized risk-free rate $r = 5\%$
- Annualized dividend rate $d = 0\%$
- Time to maturity $t = 5$ months
- Barrier is 20% below S_0 for down-and-out options

Monte Carlo valuation was adopted. Spot price followed a geometric Brownian motion and was evolved up to maturity. Provided the minimum price along the path did not decrease below the barrier, the arithmetic average of the spot price of the underlying was calculated and applied to the payoff formula (see *PathDependentAsianBarrier* class). This was repeated 1,000,000 times and the average payoff was discounted to arrive at the price of a single option. The large number of paths ensured accurate convergence to the expected price.

```
unsigned long PathDependentAsianBarrier::cashFlows(const MJArray& spotValues,
                                    std::vector<CashFlow>& generatedFlows)
const
```

```
{
    double sum = spotValues.sum();
    double mean = sum/numberOfTimes;

    generatedFlows[0].TimeIndex = 0UL;

    double payoff = 0.0;
    if (spotValues.min() > barrier ) payoff = thePayOff(mean);

    generatedFlows[0].Amount = payoff;

    return 1UL;
}
```

The process was repeated for each of the four options constituting the portfolio. The long ITM call assumed a strike of 110, the short ATM calls assumed 115 and the long OTM call had 120, keeping the long calls equidistant to the short calls and spot price. Short call prices were multiplied by -1 to reflect the appropriate direction. The four option prices were added together to arrive at the price of the portfolio, all embedded in the *MCAsianBarrierPortfolioPricer* function.

```
// [[Rcpp::export]]
double MCAsianBarrierPortfolioPricer(
    double expiry = 5.0 / 12.0,
    double spot = 115,
    double vol = 0.25,
    double r = 0.05,
    double d = 0.0,
    unsigned long int numberOfPaths = 1000000,
    unsigned long int numberOfDates = 126,
    double barrier = 0.8 * 115
)
{
    double strike;
    double position;

    // Define long call butterfly portfolio of strike-position pairs
    vector<vector<double>> options;
    options.push_back({110, 1}); // long ITM call
    options.push_back({115, -1}); // short ATM call
    options.push_back({115, -1}); // short ATM call
    options.push_back({120, 1}); // long OTM call
    double portfolioPrice = 0.0;

    for (int option = 0; option < options.size(); option++)
    {
        strike = options[option][0];
        position = options[option][1]; // 1 for long, -1 for short
// ----------------------------
```

```
// More operations, see raw code
// ----------------------------
        // Add Asian barrier call price to portfolio price
        portfolioPrice += results[results.size() - 1][0] * position;
    }

    return portfolioPrice;
}
```

An R source package (OptionPricer_0.1.0.tar.gz) was created for further use. It returned portfolio price and relied on supporting cpp and h files. The analysis.R file in home_project_analysis.Rproj then called the *MCAsianBarrierPortfolioPricer* function from the package and contained further analysis.

The price of the portfolio assuming the starting parameters amounted to 0.8807532.

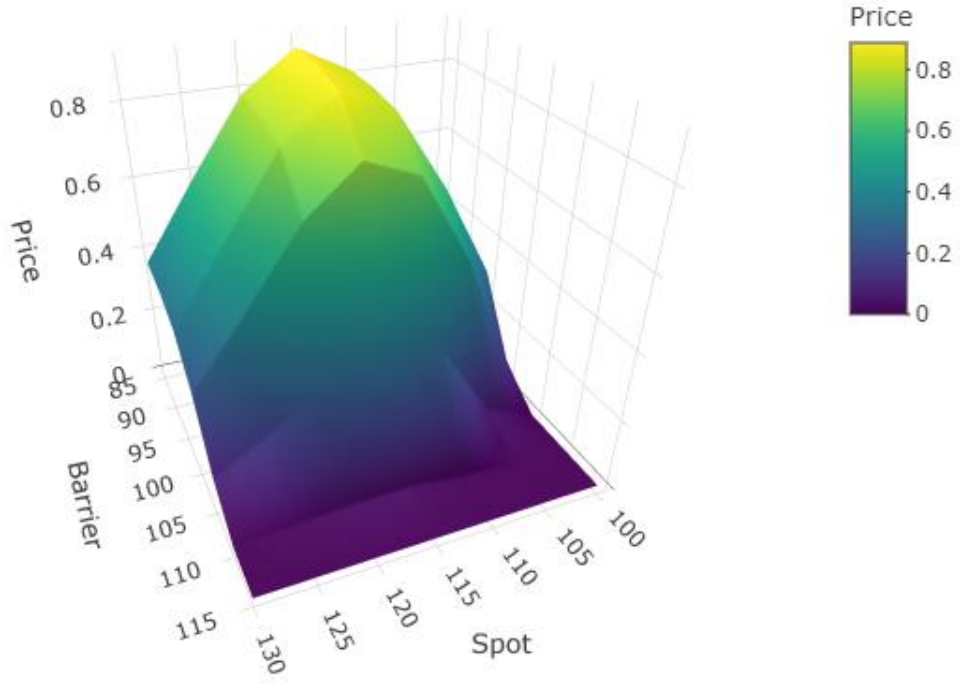
In order to investigate the relationship between the theoretical price of the portfolio and the price of the underlying and the level of the barrier, pricing was performed for each element of the cartesian product of underlying price and barrier from the sets below.

Spot prices: {100, 105, 110, 115, 120, 125, 130}

Barriers: {85, 90, 95, 100, 105, 110, 115}

Spot prices either side of S_0 were chosen as price could have evolved in either direction. Barriers were limited to levels below S_0 as values near and exceeding spot price were expected to trigger the knock-out frequently and render the price close to zero.

MCAsianBarrierPortfolioPricer was incorporated into a wrapper function. The {spot price, barrier} pair of {115, 90} produced a price close to the starting price, suggesting the wrapper was working correctly.



The combinations were visualised using the plotly package in R which portrayed each of the factors and portfolio price on separate axes.

As expected, portfolio prices were highest around S_0 due to high probabilities of the resulting average spot price exceeding the ITM call strike but not by too much, thus triggering the ITM call payoff and not being offset too much by the ATM calls. The further away spot price moved, the lower the expected payoff from the butterfly strategy and the lower the portfolio price. The relatively smooth change in gradient was attributable to the Asian option type which considers average spot prices, as opposed spot prices at maturity.

In addition, portfolio prices consistently decreased as the down-and-out barrier increased. When the barrier was low, it was triggered relatively infrequently, thus allowing the option to reach maturity and its payoff to resemble an ordinary Asian option. When the barrier increased, it triggered the knock-out more often during simulation which resulted in zero payoff and a lower price. Once it reached S_0 there were hardly any scenarios in which the price path never dipped below it and remained in positive butterfly P&L territory, explaining why portfolio prices were effectively free.

The extent to which the investor believes the underlying price would fluctuate throughout the life of the option should drive any investment decisions, particularly with considerations of downside moves which can trigger knock-out. With expectations of modest moves and few drops, this portfolio might be of benefit given its lower price.

In accordance with the Honor Code, I certify that my answers here are my own work, and I did not make my solutions available to anyone else.