Strategy Backtesting Framework for Smart Order Routing

Part 2: Code Implementation

To implement this backtester, I first started by generating the simulated market that my SOR would run in. I chose to have 3 venues within my market for the SOR to choose from. The prices for each of these venues was calculated by first creating a baseline price movement using a volatility of 1 and then adding noise to simulate different pricing at different venues. I then generated orders that the SOR would need to execute by creating orders of random sizes at random times in the market. These synthetic prices and orders served as the simulated environment that would test the TWAP SOR.

The next step was to create the TWAP SOR that would actually execute the requested orders. The logic for this algorithm is relatively simple. Once an order is received, the SOR evenly divides the total order quantity into 10 smaller orders, a number chosen arbitrarily, to be executed over the course of the next 10 time periods. At each time period, the SOR analyzes the venues to find which one has the lowest price and executes the trade at that price. To simulate slippage caused by latency, I implemented a delay in order execution by making the actual price of execution slightly after the current price. The execution cost of each of these orders was calculated using the difference between this execution price and a 5 period VWAP. Finally, the slippage and execution cost of each trade was displayed along with the cumulative performance of the SOR at the end. I also added a graph to show the price movement of the market and where each order is placed to provide an easy-to-understand visual representation of what occurred during the market simulation.

Finally, we can evaluate the results of our TWAP SOR in our simulated market. I ran the simulation 100 times, each time the simulated market experienced different price movement and different order placements. Then using the total execution cost and total slippage for each run, I found the average execution cost and average slippage for the SOR getting an average execution cost of -5.57 and average slippage of -4.37. Having a negative execution cost means that on average our SOR is generally able to obtain better prices in the market beating the VWAP benchmark, however, negative slippage means that out TWAP strategy does suffer from latency and is losing some potential profits. This slippage is expected as it was integrated into the design. Overall, the metrics produced by the TWAP SOR and the backtester performed well, however, there are still many improvements that could be made. We didn’t include any complex order types in our simulated environment, nor did we implement unique characteristics to the different venues. Adding such features would make the simulated market more realistic and improve the ability to create a successful SOR.