Homework 3: Dual Listing Arbitrage

Francisco García Flórez, Joris van Lammeren, Wouter Varenkamp

Abstract. In this homework we study pure arbitrage as a way of making risk-free profits by trading some volume between two markets. Even though profits per trade are low compared to the amount of cash involved, it is still possible to make a significant profit over time.

1 Trading robot

In this homework we focus on pure arbitrage, meaning buying low in one market and selling high in another, however we can only use immediate transactions (fill or kill), and remain with a null position.

The trading algorithm is implemented in three separate steps, executed every time there is a book update. These three steps are *offers and bids checking*, *trading* and *book updating*, each of them explained in the following sections.

We consider each book to be a set of price and volume pairs, that we denote as $(P_i^{a,b}, V_i^{a,b})$, where the superscript a, b refer to *ask* and *bid* respectively, and $i \in [1, N_{a,b}]$ is the index of the entry. In a more general setting we would also want to specify the market, however in this case there are only two options: buying `CHI_AKZA` and selling `EUR_AKZA` or the opposite, so references to bids or asks are implicitly connected to the right market.

Offers and bids checking In the first place, right after the robot receives a book update, it computes a matrix of logical values (`True` or `False`) representing which trades are profitable. This matrix T is computed as follows:

$$T_{ij} = P_i^a < P_j^b \ , \quad \forall i, j \in [1, N_a] \times [1, N_b] \quad .$$

Since not every time we receive a book update there are going to be any profitable trades, we need to check that at least one of the elements of T is `True`, which we could check by computing the product of its components:

$$\texttt{check} = \bigoplus_{i,j}^{N_a, N_b} T_{ij} \quad ,$$

thus if `check` is `True`, we proceed with the next step, in which we compute the data we need for the *trading* part. This data is contained in two matrices, each element composed of a pair of prices and volumes:

$$\begin{aligned} V_{ij} &= (V_i^a, V_j^b) \ , \quad \forall i, j \in [1, N_a] \times [1, N_b] \\ P_{ij} &= (P_i^a, P_j^b) \ , \quad \forall i, j \in [1, N_a] \times [1, N_b] \end{aligned} \quad .$$

However we still need to do one extra computation with V, since the volume that can be traded is limited by either V_i^a or V_i^b the final volume we can trade is given by $\tilde{V}_{ij} = \min(V_{ij,1}, V_{ij,2})$, where the subscript 1 and 2 reference each element of the pair.

Finally, selecting components $\tilde{V}_{ij}$ and P_{ij} for which $T_{ij} =$ `True`, we can proceed to the next step.

Trading Now, taking the volumes and prices selected in the last step and iterating over them, buying at price P_i^a and selling at P_j^b, we perform the profitable trades. Algorithmically this can be implemented as

```
for i, j such that T[i, j] is True {
  buy(ask_price[i], limit_volume[i, j])
  sell(bid_price[j], limit_volume[i, j])
}
```

This step can be slightly improved by optimizing the traded volume when more there are more than one trades possible, maximizing the volume we buy selling to more than one bidder and vice versa. However there is one detail that we need to take care of, discussed in the next step.

Book updating Finally we can proceed to perform the last step of the implementation: update the books. After every successful transaction the market changes, but in the next book update only one of the books will be saved. This means that we need to manually modify the stored book after selling and buying, so that the algorithm doesn't get tricked into sending the same transaction into the market several times.

We achieve this by looking up the entry by price and either removing it completely if the volume moved is maximum, or adjusting it if it isn't.

2 Results

As we can see in Table 3 (see Appendix) as we expected the cash moved is several orders of magnitude higher than profits, but these are completely risk-free earned in about 21 minutes of trading. Since this is only pure arbitrage there

are many opportunities to improve on it, basically introducing all the possible strategies that characterize algorithmic trading.

3 Appendix

#	CHI (Buy)	CHI (Sell)	EUR (Buy)	EUR (Sell)	Profit
1	-37407.95	74623.63	-74613.81	37412.86	14.73
2	-57272.87	24620.31	-24617.09	57280.36	10.71
3	-37686.14	29292.16	-29288.31	37691.10	8.81
4	-19650.57	85786.14	-85774.92	19653.14	13.79
5	-66249.23	57756.53	-57748.97	66259.82	18.15

Table 1: Cash moved for each feed.

#	CHI (Buy)	CHI (Sell)	EUR (Buy)	EUR (Sell)	Position
1	491	-982	982	-491	0
2	749	-322	322	-749	0
3	496	-385	385	-496	0
4	257	-1122	1122	-257	0
5	866	-756	756	-866	0

Table 2: Assets moved for each feed.

References

[1] P. Wilmott et al, *The Mathematics of Financial Derivatives*, 1995.