Blockhouse Work Trial Task

1 Introduction

When you choose to buy or sell an asset, you can place two main types of orders: a **market** order and a **limit** order. The mechanics of the two order types are very different in nature; here we explain how it works:

1. A limit order allows you to **buy** or **sell** an asset at a predetermined price. Given that you have to wait in a queue; the first orders that join the limit order book is executed first in a FIFO style queue.
   Benefit/Drawback : You are able to execute an order at the price you want, but you have to wait for orders in front of you to execute first. Sometimes you wait 1 second, 1 minute or ... for eternity (will explain).

2. A market order allows you to **buy** or **sell** an asset immediately, but the price is variable depending on the available liquidity.
   Benefit/Drawback : You are able to execute an order immediately, but at a price you can't decide.

Let us look at the dynamics of the two more closely in this example scenario (this is a dummy scenario not indicative of a true market!):

Consider an empty exchange for a stock called BLOCKHOUSE. Currently there is no one in the exchange.

At time 9:00 a seller joins the market, they place an **ask**-side limit order at a price of $9.00 for a quantity of 10 shares. This means they are willing to wait in the queue (which is empty right now) to sell 10 shares of the asset at a price of $9.

At time 9:01 a buyer joins the market, they place a **bid**-side limit order at a price of $8.90 for a quantity of 20 shares. This means they are willing to wait in the queue (which is empty right now) to buy 20 shares of the asset at a price of $8.90.

At this instant we can create a snapshot of the limit order book.

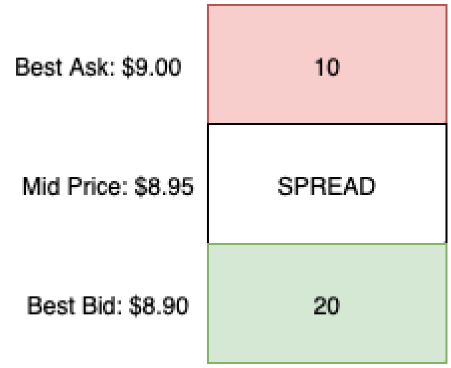

Figure 1: Limit order book at 9:00

Now at 9:02 another buyer joins at $ 8.85 with 13 shares and another seller joined at $9.05 with 8 shares. We have **depths** to the limit order book!

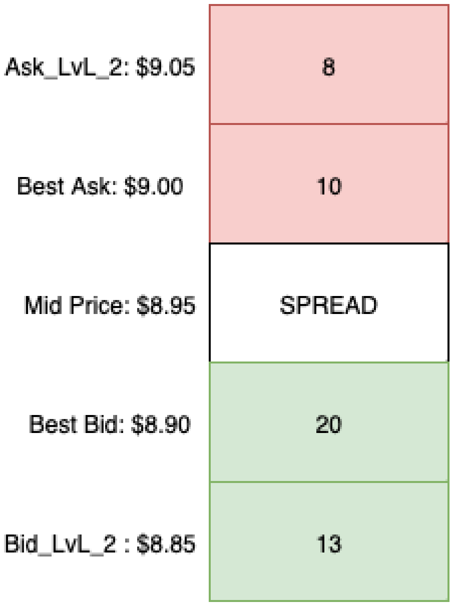

Figure 2: Snapshot at 9:02

At this time all the orders so far are still waiting! This is why we wait when we place limit orders. How do we get an execution? Suppose that at 9:03 someone joins the market, very eager to buy 8 shares. They place a buy-side market order for 8 shares, immediately their order get satisfied. How? Their orders are now matched with the cheapest ask-side limit orders! The best-ask queue drops from 10 to 2, the person who placed their orders at $9.00 now sees that 8 of their orders are filled at $9.00 (they sold 8 shares at $9.00) and the

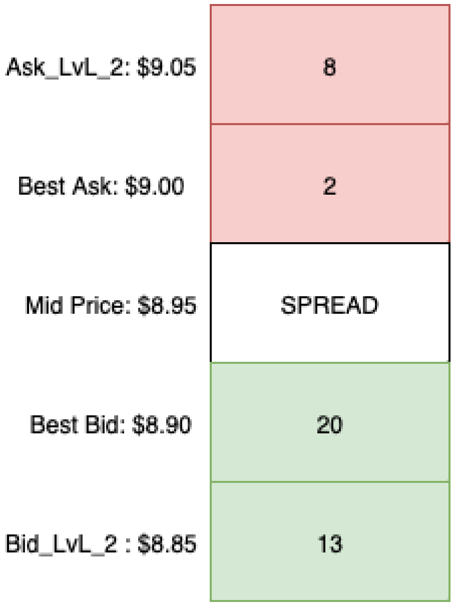

Figure 3: Snapshot at 9:03

trader who placed the market order bought 8 shares at a relatively expensive \$9.00.

Now, what we do is we log **slippage**, it describes how much ”worse” it costed you to execute your order with respect to the mid price. For the limit order trader their slippage was \$8.95 -\$9.00 = -\$0.05 , they were able to sell for 5 cents more than the mid price. While the market order trader has the slippage of \$9.00 - \$8.95 = \$0.05, they had to buy the asset at 5 cents more expensive than the mid price. Sometimes your slippage can be very high. For example, let us suppose that at 9:04 a trader comes in and places a market order to sell the asset at a quantity of 30 shares. Notice that the best-bid level only has 20 shares, so his order of 30 will be split into: 20 shares at \$8.90 and 10 shares at \$8.85. Their slippage is now $\frac{20\times(8.95-8.90)+10\times(8.95-8.85)}{30} = \$0.067$. This is because they **eat** into the second level, the order book now looks like this:

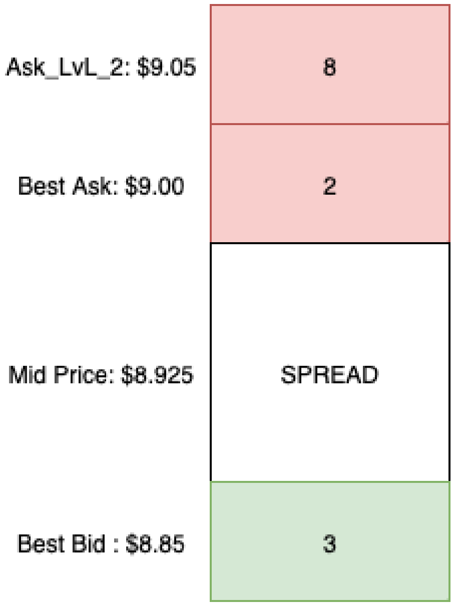

Figure 4: Snapshot at 9:04

This should be sufficient to understand the basic workings of an order book, what limit and market orders are, and how we compute slippage.

Note: Recall that we said sometimes you wait for an eternity if you place limit orders. This can happen if you place an order at a price that is too high or too low (depending on if you sell or buy) and the price drifts away from you never to return.

2 Problem Statement

Let us define the **temporary impact** function $g_t(X)$ as the amount of slippage you incur if you place X orders at the current time t. For example, consider the following limit order book data:

	Price		Ask Size		Bid Size	
	$81.00		140			
	$80.99		123			
	$80.97		110			
	$80.96			SPREAD		
	$80.95					
	$80.93				112	
	$80.92				138	
	$80.91				200	

For this order book state, the best bid is \$80.93, the best ask is \$80.97, the spread is \$0.04. The amount of orders in each limit order price is specified in red (ask) and green (bid). The temporary impact function at that instant would look like this:

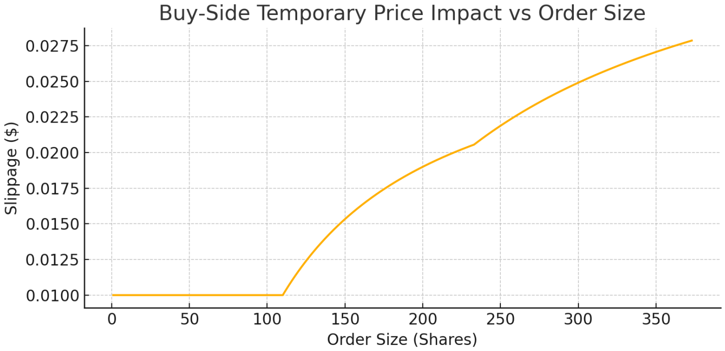

Figure 5: Buy Side (Ask) temp impact

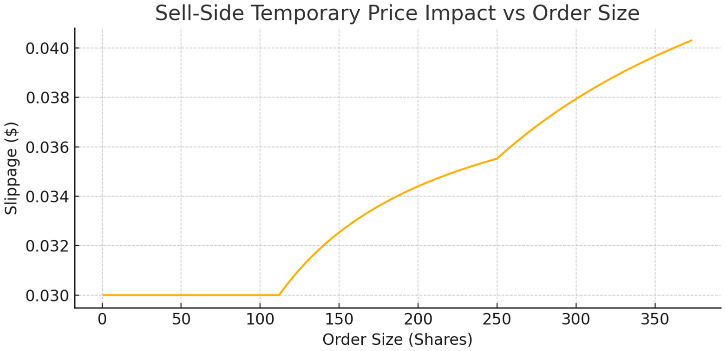

Figure 6: Sell Side (Bid) temp impact

Let us suppose that we have S total amount of orders to be bought, you are given how many shares S to buy before market opens, and must buy exactly S shares by end of day. You are tasked with creating a strategy that is supposed to minimize the total temporary impact of the executed orders. Mathematically speaking, let $g_t(X)$ be the temporary impact function at some time t. Let us split the day into N trading periods. Let $\mathbf{x} \in \mathbb{R}^N$ be your allocation vector (we generally have to take integer quantities but real numbers are okay for now) where you choose to buy x_i shares at the i^{th} period. Given data for $N = 390$ (one minute trading window) and data for 3 stocks. Please answer the following questions:

1. How do you choose to model the temporary impact $g_t(x)$? For example, sometimes people try to "linearize" the model $g_t(x) \approx \beta_t x$. If you think linear models are gross oversimplifications, how would you model it? Please write a 1-2 page explanation on your model, using data from the 3 tickers provided. We understand that 3 tickers is not enough data so any valid reasoning/conclusions derived from these 3 tickers would be accepted. Please also attach a link to a python notebook or code (pref ably uploaded on GitHub) where you conducted your analysis.

2. Formulate roughly but rigorously a mathematical framework / algorithm that gives us x_i when we are at time t_i. Make sure that $\sum_i x_i = S$. This should be relatively short, at most 2 pages. You don't have to fully solve the problem, but a clear mathematical setup and discourse into the techniques + tools used to solve the problem would be sufficient.