In this post, I would like to raise attention to an important decision the protocol needs to make regarding the design and implementation of partially-fillable limit orders; these are limit orders that can be partially executed, meaning that their execution can be split in multiple chunks, that will be executed in distinct auctions over multiple blocks.

One decision that needs to be made regarding such orders is whether solvers should be incentivized to fill "as much as possible" of such an order, or whether solvers should be incentivized to maximize the surplus such an order gets. I will come this dilemma the "surplus-capturing vs volume-maximizing issue".

## An example

To get a better understanding of the issue, let's consider the following example. Suppose a SELL order is placed, that is selling 20 ETH and buying some token X at a limit price of 1000 X per ETH. For simplicity, let's assume there is only one liquidity source, and this is a constant-product pool with no fees, with the following reserves:

ETH: 100

X: 120,000

### The fill-or-kill case (i.e., current status quo)

Let's assume that the order is fill-or-kill. We now check whether we can trade with the pool. We need to satisfy the constant-product formula of the pool, which means the following must hold, where b denotes the amount the AMM returns (i.e., the buy amount of the user order).

$(100 + 20)\ (120{,}000 - b) = 100 * 120{,}000 \leftrightarrow$

$b = 120{,}000 - 12{,}000{,}000 / 120 = 120{,}000 - 100{,}000 = 20{,}000$

Thus, the pool, although quite shallow, can accommodate the execution of the order, matching it exactly at its limit price.

### The partially-fillable case

Suppose now that the order was marked as partially fillable. Would (or should) that change the way it is executed? I will now discuss two different approaches to this.

#### 1. Surplus-maximization

In this approach, the order would be executed such that surplus is maximized. It is not hard to see if the order sells an amount s

of ETH, where $0 < s <= 20$

and receives a buy amount b

, then its surplus, measured in the token X, is equal to $b - s * 1000$

.

Note that for every value of s

, we have $b = 120{,}000 - 12{,}000{,}000 / (100 + s)$

.

One can analytically show that the maximum is attained for s = 20 sqrt{30} - 100 = 9.54

, and results in a surplus of roughly 911

tokens X. In such a case, the user perceives an exchange rate equal to

clearing_exchange_rate = b / s = 1095 X per ETH.

It is worth mentioning that by trading this amount s

with the pool, the new pool reserves are:

ETH: 20sqrt{30}

X: 12,000,000 / 20sqrt{30}

Looking at the so-called new spot-price of the pool, i.e., at the ratio X/ETH, we see that it is equal to

new_spot_price = 12,000,000 / ETH^2 = 12,000,000 / (400 * 30) = 12,000 / 12 = 1000

which is exactly equal to the limit price of the order. Intuitively, this means that the order would trade with the AMM as long as it provides a price that is at least as good as the limit price of the order, for every epsilon-chunk of the volume traded. As soon as this stops being the case, we don't trade anymore with the AMM.

### 2. Volume-maximization

In this approach, the order would be executed as long as the average price it would observe would respect its limit price. In particular, this would imply that in a strict partial fill, the order would get matched at its limit price.

For the example above, the order would trade an amount s' with the AMM, such that the order's buy amount b' satisfies

b'/s' = 1000.

This is equivalent to

120,000 - 12,000,000 / (100 + s') = 1000s'

which gives

s' = 20.

This means the order would send s' = 20 ETH

to the AMM and receive back b' = 20000

of token X.

Regarding the new state of the pool, the updated reserves now are:

ETH: 120

X: 100,000

Note that this means that the AMM's new spot price is

new_spot_price = 100,000 / 120 = 833 X per ETH.

This would suggest a potential backrunning opportunity, as we have moved the AMM beyond the order's limit price, and the only reason we were able to do so was that the avg price the user got respected their limit price.

## Comments

I would like to highlight here that trading with an AMM after moving its spot price beyond an order's limit price is already the case for fill-or-kill orders, as implied in the beginning. Thus, the main question here is whether the protocol would like that partially fillable orders behave like fill-or-kill orders, whenever possible, which are, in a way, volume-maximizing as well, or whether the semantics of partial fills are fundamentally different and thus, activating the partial-fill flag would change the execution of the order completely.

One additional comment is that this choice for partially fillable orders also affects the execution of regular fill-or-kill market orders. To see this, let's modify the example above and suppose that we have two orders:

- one that is fill-or-kill and selling 10 ETH for at least 10,000 X (i.e., limit price of 1000 X per ETH), and
- one that is partially fillable, and is selling 10 ETH for X at the same limit price of 1000 X per ETH.

It is not hard to see that if we decide on the surplus-maximization approach, then the optimal execution would be to execute only the partially fillable order and trade 9.54 ETH with the AMM, as calculated above. On the other hand, if we go with the volume-maximizing approach, either of the two orders could be executed, or both, assumming that detecting the CoW and interacting with the pool only once allows for a reduction of costs that compensates for the zero surplus.

Finally, a point to consider is the potential use cases of partially fillable orders. Can we think of many cases where we would need volume-maximizing partially-fillable orders? If so, this would mean that fill-or-kill limit orders cannot work in such cases. If not, then maybe the fundamentally new semantics a surplus-maximizing partially-fillable order introduces might be a more meaningful addition to the current set of orders the protocol supports.