Dradex: Unified Order Book & AMM Liquidity Model

Hai Nguyen[1] and Ha Nguyen[1]

[1]Dradex

Abstract

This whitepaper introduces our unified order book & AMM liquidity pool design. It explains the downside of single AMM liquidity pool as well as single order book. It also covers the optimal proof of our matching algorithm.

1. Introduction

Dradex is a protocol for decentralized token exchange on Solana, where an Order Book works together in tandem with an AMM Liquidity Pool to provide the best prices, lowest slippage.

There are problems with current AMM models. Due to the *Constant Product* mechanism, the AMM Price is based on reserve amounts of tokens in the pool, it is changing as a match happens. A larger match size or volume (compared to the reserve amounts in the pool) would result in a larger difference (or slippage) between the initial LP price and the final matching price. This results in highly volatile markets where arbitragers make a lot of profit and Liquidity Providers suffer substantial amounts of Impermanent Loss.

Standalone order books, on the other hand, suffer for lack of liquidity. In the DEX environment where there are fewer market makers than CEX, launching a new order book is much more difficult compared to AMM liquidity pool , it lacks liquidity and becomes a highly inefficient market. Consequently, there are fewer takers coming in to trade, then fewer makers to place maker orders. It is a death spiral. As a result, a market order would be impractical in these markets because of this lack of liquidity and would result in huge slippage.

Therefore, we come up with Dradex's *Unified Order Book & AMM Liquidity Model*, which integrates the Order Book model with the AMM Constant Product Liquidity formula, this increases the liquidity, reduces slippage, also provide best price and more options in order book for user to trade.

Thanks to Solana's speed and low transaction costs, it is feasible to implement our unique architecture on Solana.

2. Unified Formula

2.2. Order Book definition

The traditional Order Book model works by having a list of buys and sells that traders are willing to make, at a certain price, that price is called the *limit price*.

When a trader A submits a BUY order at limit price p_{order}, they expect it to be matched with a SELL order at price $p \leq p_{order}$ (1.1)

The highest price for a BUY order on the Order Book is called the *bid price* p_{bid} , by order O_{bid}.

When a trader B submits a SELL order at limit price p_{order}, they expect it be matched with a BUY order at price $p \geq p_{order}$ (1.2)

The lowest price for a SELL order on the Order Book is called the *ask price* p_{ask} , by order O_{ask}.

Whenever there is a new BUY order at price $p_{order} \geq p_{ask}$, or a new SELL order at price $p_{order} \leq p_{bid}$, (1.1) and (1.2) are satisfied at the same time, a match occurs, and a transaction happens.

2.2. Liquidity Pool definition

In order to provide liquidity, we match the orders not only among themselves, but also against an AMM Liquidity Pool (the *LP* for short), which uses the *Constant Product* AMM mechanism

$$x \cdot y = k$$

Where x and y are reserve amounts for the 2 tokens in the pool, and k is kept constant.

The *Constant Product* AMM mechanism has been proven effective as a means to facilitate trades on decentralized exchanges without needing an order book. Turns out it is effective as a means to provide liquidity for an order book as well.

The price of the asset in the pool (the *LP Price*) is determined as

$$p_{lp} = \frac{y}{x}$$

It is treated like a permanent, omnipresent BUY & SELL order at a dynamic price p_{lp} on the exchange, therefore, whichever BUY order that has $p_{order} \geq p_{lp}$ (satisfies (1.1)), or SELL order that has $p_{order} \leq p_{lp}$ (satisfies (1.2)) would be matched with it and be filled by the LP as if it is the other trading side.

2.3. Unified AMM liquidity pool & order book model.

The state of a hybrid AMM liquidity pool and a order book for a pair A/B can be defined by following.

Liquidity pool:

x_0 : the total number of X tokens in the liquidity pool

y_0: the total number of Y tokens in the liquidity pool

k: the constant product $k = x_0 \cdot y_0$

Order book:

Ask list: $(p_{ask_i},\ s_{ask_i})$, $i\ =\ \overline{1,\ n_1}$. p means price, s means size, n_1 is the number of ask orders in the order book.

$$p_{ask_i} < p_{ask_{i+1}}$$

Bid list: $(p_{bid_i},\ s_{bid_i})$, $\ i\ =\ \overline{1,\ n_2}$. P means price, S means size, n_2 is the number of bid orders in the order book.

$$p_{bid_i}\ >\ p_{bid_{i+1}}$$

Characteristic of liquidity pool and order book

$$p_{bid_1}\ \leq\ p_{lp}\ \leq\ p_{ask_1}$$

2.3.1. Matching against a BUY order

When a new BUY order O with volume $v_{order} > 0$, at limit price $p_{order}\ \geq p_{lp}$ is submitted to the exchange, a match can occur between the LP or maker order from order book and this order. The objective is to maximize the matching volume v_m then the minimize matching price p_m to buy $v_m$tokens such that we don't violate the below constraints.

Constraints:

1. The final matching price p_m cannot exceed the order's limit price: (C_1)

$$p_m\ \leq\ p_{order}$$

2. The price of liquidity pool after matching cannot be worse than the *target price*: (C_2)

$$p'_{lp}\ \leq\ p_t\ = \min\{p_{order},\ p_{ask_s}\}$$

Where p_{ask_s} is the lowest ask after matching with order O done.

3. BUY order can only be matched against SELL orders with lower prices $p_{ask} \leq p_{order}$: (C_3)

First step: we need to find the maximum volume $v_m$we could fill for order O without violate the constraint C_1.

Apparently we will try to fill as much as possible from both liquidity pool and order bool.

We take all the any ask order such that

$$p_{ask}\ \leq\ p_{order}$$

$v_{o_{\max}}\ = \sum_{i\ =\ 1}^{t} S_{ask_i}$, where $p_{ask_i}\ <\ p_{order}$

$v_{l_{\max}}$ is the maximum match size we could fill from only the liquidity pool without violating (C_1), (C_2) and (C_3).

$v_{m_{\max}} =\ v_{o_{\max}} +\ v_{l_{\max}}$.

$v_{o_{\max}}\ =\ \sum_{i\ =\ 1}^{t} s_{ask_i}$, where t is the maximum number such that $p_{ask_t}\ <\ p_{order}$

$v_{l_{\max}} =\ x_0\ -\ \sqrt{\frac{k}{p_{order}}}$

v_m is the volume we match for buy order O, without violating 3 constraints $(C_1,\ C_2, C_3)$

$v_m\ =\ \min\ \{v_{m_{\max}},\ v_{order}\}$

If $v_{m_{\max}}\ \leq\ v_{order}$, then we can simply fill $v_{o_{\max}},\ v_{l_{\max}}$against the order book and the liquidity pool, respectively.

if $v_{m_{\max}}\ >\ v_{order}$, then we can fully fill the order. Our combined model is able to fill v_{order} with better price than p_{order}. We need to determine the optimal matching volume from liquidity pool v_l and $v_{order}\ -\ v_l$ from order book in order to minimize the matching price p_m.

If there is no ask with better price than p_{order}, then $v_m\ =\ v_l\ =\ \min\ \left(x_0\ -\ \sqrt{\frac{k}{p_{order}}},\ \ v_{order}\right)$

Second step: we need to determine the optimal matching volume to take from liquidity pool and order book each to fill v_m with best price.

Let t_0 be the maximum i such that $\sum_{j\ =\ 1}^{i} s_{ask_i}\ \leq\ v_m, p_{ask_i}\ \leq p_{order}$.

If $p_{ask_{t_0+1}}\ >\ p_{order}$, we could add an extremely small order with price $p_{order}\ -\epsilon_1$ and size ϵ_2. where $\epsilon_1,\ \epsilon_2\ \rightarrow$ 0. Then we could assume $p_{ask_{t_0}}\ \rightarrow p_{order}$

Let $k_i\ =\ \sum_{j\ =\ 1}^{t} s_{ask_j}$ for $t\ =\ \overline{1,\ t_0+1}$ and $k_0\ =\ 0$

Let v_l be the optimal matched size from liquidity pool and $v_o\ =\ v_m\ \ -\ v_l$ is the optimal matching size from order book.

Let $f\left(v_l\right)$ be the cost function, representing the amount of money the user has to pay for v_m tokens of X.

Then we could split the range for v_l to below intervals:

$[v_m\ -\ \ k_i\ ,\ v_m\ -\ k_{i-1}]$ for $i = \overline{1,\ t_0+1}$

Then

$f_i\left(v_l\right)\ =\ \left(\frac{k}{x_0\ -\ v_l}\ -\ y_0\ \right)+\ \left[\left(\sum_{j=1}^{i-1} p_{ask_j}\ .\ s_{ask_j}\right)\ +\ \left(s_{order}\ -\ v_l\ -\ \sum_{j\ =\ 1}^{i-1} s_{ask_{i=j-1}}\right)\cdot\ p_{ask_i}\right]$

Where $\frac{k}{x_0\ -v_l}\ -\ y_0$ is the matching cost from liquidity pool

and $\left[\left(\sum_{j=1}^{i} p_{ask_j}\ .\ s_{ask_j}\right)\ +\ \left(s_{order}\ -\ v_l\ -\ \sum_{j\ =\ 1}^{i-1} s_{ask_{j-1}}\right)\cdot\ p_{ask_i}\right]$ is the matching cost from order book.

And $v_l\ \in (v_m\ -\ \ k_i\ ,\ v_m\ -\ k_{i-1}]$ for $i\ =\ \overline{1,\ t_0+1}$

$f_{t_0+1}\left(v_l\right)\ =\ \left(\frac{k}{x_0\ -\ v_l}\ -\ y_0\ \right)+\ \left(\sum_{j=1}^{t_0} p_{ask_j}\ .\ s_{ask_j}\right)\ \ +\ \left(s_{order}\ -v_l-\sum_{j\ =\ 1}^{t_0} s_{ask_j}\right)\cdot\ p_{ask_{t_0+1}}$ for $i\ =\ t_0\ +1$

Then we need to determine the optimal v_{l_i} for each interval to minimize the cost in $f_{i\left(v_{l_i}\right)}$

From $t_0\ +1$ optimal points, we could pick the most optimal v_{l_i}. This is the optimal matching volume against liquidity pool we need to find i such that $f_{i\left(v_{l_i}\right)}$ is minimum.

For $i\ =\ \overline{1,\ t_0+1}\ $ ***:***

$f_i\left(v_l\right)\ =\ \left(\frac{k}{x_0\ -\ v_l}\ -\ y_0\ \right)+\ \left[\left(\sum_{j=1}^{i-1} p_{ask_j}\ .\ s_{ask_j}\right)\ +\ \left(s_{order}\ -\ v_l\ -\ \sum_{j\ =\ 1}^{i-1} s_{ask_{j-1}}\right)\cdot\ p_{ask_i}\right]$

$v_l\ \in (v_m\ -\ \ k_i\ ,\ v_m\ -\ k_{i-1}]$

Taking the first derivative, we study the behavior of the cost function:

$f_i'\left(v_l\right)\ =\ \frac{k}{\left(x_0\ -\ v_l\right)^2}\ -\ p_{ask_i}$

$f_i'(v_l) = 0 \leftrightarrow v_l = x_0 - \sqrt{\frac{k}{p_{ask_i}}} = v_{l_{i_{\min}}}$

$f_i'(v_l) > 0$ when $v_l > v_{l_{i_{\min}}}$

$f_i'(v_l) < 0$ when $v_l < v_{l_{i_{\min}}}$

Then consider the interval $v_l \in [v_m - k_i,\ v_m - k_{i-1}]$:

If $v_{l_{i_{\min}}} \leq v_m - k_i$ then $f_i(v_l)$ has minimum at $v_l = v_m - k_i$

If $v_{l_{i_{\min}}} \geq v_m - k_{i-1}$ then $f_i(v_l)$ has minimum at $v_l = v_m - k_{i-1}$

We have:

- $p_{ask_1} < p_{ask_2} < p_{ask_3} < \ ... \ < p_{ask_{t_0+1}} \leq p_{order}$

$\rightarrow 0 = x_0 - \sqrt{\frac{k}{p_{lp}}} \leq v_{l_{1_{\min}}} = x_0 - \sqrt{\frac{k}{p_{ask_1}}} < v_{l_{2_{\min}}} = x_0 - \sqrt{\frac{k}{p_{ask_2}}} < \ ... \ < v_{l_{t_0+1_{\min}}} = x_0 - \sqrt{\frac{k}{p_{ask_{t_0+1}}}}$ (2.3.1.1)

- $k_0 < k_1 < k_2 < ... < k_{t_0+1}$

$\rightarrow v_m = v_m - k_0 > v_m - k_1 > v_m - k_2 > \ ... \ > v_m - k_{t_0} > v_m - k_{t_0+1}$ (2.3.1.2)

Case 1: $v_{l_{1_{\min}}} = x_0 - \sqrt{\frac{k}{p_{ask_1}}} > v_m - k_0$

$f_i(v_l)$ where $v_l \in (v_m - k_i\ ,\ v_m - k_{i-1}]$ is minimum at $v_l = v_m - k_{i-1}$ since $v_{l_{i_{\min}}} > v_m - k_{i-1}$

$f_{i-1}(v_l)$ where $v_l \in [v_m - k_{i-1}\ ,\ v_m - k_{i-2}]$ is continuous at $v_l = v_m - k_{k-1}$ and

$f_{i-1}(v_l)$ is minimum at $v_l = v_m - k_{i-2}$ since $v_{l_{i-1_{\min}}} > v_m - k_{i-2}$

then $f_{i-1}(v_m - k_{i-2}) < f_i(v_m - k_{i-1})$

Then $f_1(v_m - k_0) = f_1(v_m)$ is the optimal cost we need to find, the volume matching against liquidity pool we need to find is v_m.

Case 2: $v_{l_{1_{\min}}} \leq v_m - k_0$ and $v_{l_{t_0+1_{\min}}} \geq v_m - k_{t_0+1}$

From (2.3.1.1) and (2.3.1.2) there exists θ such that $v_m - k_\theta < v_{l_{\theta_{\min}}} \leq v_m - k_{\theta-1}$

For $i < \theta$, $f_i(v_l)$ where $v_l \in (v_m - k_i\ ,\ v_m - k_{i-1}]$

- $v_{l_{i_{\min}}} < v_{l_{\theta_{\min}}} \leq v_m - k_{\theta-1} \leq v_m - k_i$.

Then $f_i(v_l)$ has minimum at $v_m - k_i$

Also if $i+1 < \theta$, $f_{i+1}(v_l)$ has minimum at $v_m - k_{i+1}$ and is continuous at $v_m - k_i$

Then $f_i(v_m - k_i) \leq f_{i+1}(v_m - k_{i+1}) \leq \ ... \ \leq f_\theta\ (v_m - k_\theta) \leq f_\theta\left(v_{l_{\theta_{\min}}}\right)$

Similarly, for $i > \theta$, $f_i(v_l)$ *where* $v_l \in (v_m - k_i,\ v_m - k_{i-1}]$

- $v_{l_{i_{\min}}} > v_{l_{\theta_{\min}}} > v_m - k_\theta \geq v_m - k_{i-1}$

Then $f_i(v_l)$ has minimum at $v_m - k_{i-1}$

Also if $i-1 > \theta$, $f_{i-1}(v_l)$ has minimum at $v_m - k_i$ and is continuous at $v_m - k_{i-1}$

Then $f_i(v_m - k_{i-1}) \leq f_{i-1}(v_m - k_{i-2}) \leq ... \leq f_\theta(v_m - k_{\theta-1}) \leq f_\theta\left(v_{l_{\theta_{min}}}\right)$

Then $f_\theta\left(v_{l_{\theta_{min}}}\right)$ is the optimal cost we need to find, and the volume matching against liquidity pool we need to find is $v_{l_{\theta_{\min}}} = x_0 - \sqrt{\frac{k}{p_{ask_\theta}}}$.

We price that θ also unique. Assume otherwise, there is 2 optimal $\theta_1 < \theta_2$

Then $v_m - k_{\theta_1} < v_{l_{\theta_{1_{\min}}}} \leq v_m - k_{\theta_1 - 1} \leq v_m - k_{\theta_2} < v_{l_{\theta_{2_{\min}}}} \leq v_m - k_{\theta_2 - 1}$

$\rightarrow v_m - k_{\theta_1} < v_m - k_{\theta_2} \leftrightarrow k_{\theta_1} > k_{\theta_2} \leftrightarrow \theta_1 > \theta_2$ (contradiction!)

Therefore θ is unique and optimal.

Then $x_0 - \sqrt{\frac{k}{p_{ask_\theta}}}$ is the optimal matching volume against liquidity pool we need to find. It is also easy to see that $x_0 - \sqrt{\frac{k}{p_{ask_\theta}}}$ satisfies constraints (C_1), (C_2)

Case 3: $v_{l_{t_0+1_{\min}}} < v_m - k_{t_0+1}$

$f_i(v_l)$ where $v_l \in (v_m - k_i, v_m - k_{i-1}]$ is minimum at $v_l = v_m - k_i$ since $v_{l_{i_{\min}}} < v_m - k_i$

$f_{i-1}(v_l)$ where $v_l \in [v_m - k_{i-1}, v_m - k_{i-2}]$ is continuous at $v_l = v_m - k_{k-2}$ and

$f_{i-1}(v_l)$ is minimum at $v_l = v_m - k_{i-2}$ since $v_{l_{i-1_{\min}}} < v_m - k_{i-2}$

then $f_{i-1}(v_m - k_{i-1}) > f_i(v_m - k_i)$

Then $f_{t_0+1}(v_m - k_{t_0+1})$ is the optimal cost we need to find, the volume matching against liquidity pool we need to find is $v_m - k_{t_o+1}$.

To summarize the optimal result for matching against a BUY order:

v_m is the volume we match for buy order O, without violating constraints (C_1, C_2, C_3)

$v_{m_{\max}}$ is the maximum volume we could match from liquidity pool and order book without violating constraints (C_1, C_2, C_3)

$v_m = \min\{v_{m_{\max}}, v_{order}\}$,

v_l is the optimal volume we match from liquidity pool, $v_m - v_l$ matched from order book in order to offer the best price.

- If there is no ask order at all, $v_m = v_l = \min\left(x_0 - \sqrt{\frac{k}{p_{order}}}, v_{order}\right)$
- If there are asks,

  Case 1: $x_0 - \sqrt{\frac{k}{p_{ask_1}}} > v_m$ then $v_l = v_m$

  Case 2: $v_{l_{t_0+1_{\min}}} = x_0 - \sqrt{\frac{k}{\min\left\{p_{order}, p_{ask_{t_0+1}}\right\}}} < v_m - k_{t_0+1}$ then $v_l = v_m - k_{t_o+1}$

  Case 3: else

  $v_l = v_{l_{\theta_{min}}} = x_0 - \sqrt{\frac{k}{p_{ask_\theta}}}$

  Where θ satisfies: $v_m - k_\theta < v_{l_{\theta_{\min}}} = x_0 - \sqrt{\frac{k}{p_{ask_\theta}}} \leq v_m - k_{\theta-1}$

  and $k_j = \sum_{i=1}^{j} s_{ask_i}$.

2.3.2. Matching against a SELL order

We could define a similar steps matching against a buy order above, to get the optimal result as follow.

v_m is the volume we match for sell order O, without violating constraints $(C_1,\ C_2,\ C_3)$

$v_{m_{\max}}$ is the maximum volume we could match from liquidity pool and order book without violating constraints $(C_1,\ C_2,\ C_3)$

$v_m\ =\ \min\ \{v_{m_{\max}},\ v_{order}\}$

v_l is the optimal volume we match from liquidity pool, $v_m - v_l$ matched from order book in order to offer the best price.

- If there is no bid order at all, $v_m\ =\ v_l\ =\ \min\ \left(\sqrt{\frac{k}{p_{order}}} - x_0,\ \ v_{order}\right)$
- If there are bids,

Case 1: $\sqrt{\frac{k}{p_{bid_1}}} - x_0 >\ v_m$ then $v_l\ =\ v_m$

Case 2: $v_{l_{t_0+1_{\min}}} = \sqrt{\frac{k}{\min\left\{p_{order},\ p_{ask_{t_0+1}}\right\}}}\ - x_0 < v_m - k_{t_0+1}$ then $v_l =\ v_m\ -\ k_{t_o+1}$

Case 3: else

$v_l\ =\ v_{l_{\theta_{min}}}\ =\ \sqrt{\frac{k}{p_{bid_\theta}}} -\ x_0$

Where θ satisfies: $v_m\ - k_\theta\ <\ v_{l_{\theta_{\min}}} = \sqrt{\frac{k}{p_{bid_\theta}}} - x_0\ \leq\ v_m\ - k_{\theta-1}$

and $k_j\ =\ \sum_{i\ =\ 1}^{j} s_{bid_i}$.

3. The Unified Matching Algorithm

We propose a matching algorithm and prove that it offers a best price for any of combination of matching size from liquidity pool and order book.

3.1.1. The algorithm

We propose a matching algorithm based on the Unified Formula in Section 2.

For a new order with size v_{order} and price p_{order} submitted onto the exchange, we do the following:

1. $remaining\ \leftarrow\ v_{order}$
2. If $remaining\ \leq\ 0$, stop matching.
3. Calculate the match size: $v_l\ =\ x_0\ - \sqrt{\frac{k}{p_t}}$, where $p_t\ =\ \min\{p_{order},\ p_{ask}\}$ for BUY; $v_l\ =\ \sqrt{\frac{k}{p_t}} - x_0$, where $p_t\ =\ \max\{p_{order},\ p_{bid}\}$ for SELL
4. If $v_l\ > 0$, match the order against LP with match size $v_l = \min\{v_l,\ remaining\}$; then update LP reserve amounts, and update $remaining\ \leftarrow\ remaining\ -\ v_l$
5. If possible, match the order against the best available offer O_{offer} in the order book (O_{ask} for BUY, O_{bid} for SELL) with size $v_o = \min\{v_{offer},\ remaining\}$, then update $remaining\ \leftarrow\ remaining\ - v_o$. If not possible, break the loop, stop matching.
6. Go back to 2

3.2.1 Optimal proof

It is easy to see that our algorithm is greedily taking the maximum volume against liquidity pool such that it satisfies constraint (C_1), (C_2), (C_3) at each step.

We will prove that if the result from our algorithm matches the optimal result we found in section (2.3)

Again, we only prove for matching a BUY order. For SELL order, we handle similarly.

Note: If we split $\delta \rightarrow \delta_1 + \delta_2$ and match against Liquidty pool. There is no impact, the LP state $(x,\ y,\ k)$ after matching against δ or against $\delta_1 \rightarrow \delta_2$ is still the same

If there is no ask at all, our algorithm got the optimal result in step 4 and stop.

If there are asks:

If $x_0 - \sqrt{\frac{k}{p_{ask_1}}} > v_m$, the algorithm stop at step 5 and got the optimal result.

else:

Let β be the number of iterations our algorithm takes. Then the we have two cases.

Case 1: stop at step 2

Let v_{l_i} be the volume of our algorithm matching against liquidity pool in iteration i.

v_{o_i} be the volume of our algorithm matching against order book in iteration i.

We prove by induction that $v_{l_i} = \sqrt{\frac{k}{p_{ask_{i-1}}}} - \sqrt{\frac{k}{p_{ask_i}}}$ and $v_l = x_0 - \sqrt{\frac{p}{p_{ask_i}}},\ x_i = \sqrt{\frac{k}{p_{ask_i}}}$ for $i = \overline{1,\beta}$. (3.2.1.1)

If $\beta = 1$, then

$v_{l_1} = x_0 - \sqrt{\frac{k}{p_{ask_1}}}$

$x_1 = \sqrt{\frac{k}{p_{ask_1}}},\ y_1 = \frac{k}{\sqrt{\frac{k}{p_{ask}}}} = \sqrt{k.p_{ask_1}},\ p_{lp_1} = \frac{\sqrt{k \cdot p_{ask_1}}}{\sqrt{\frac{k}{p_{ask_1}}}} = p_{ask_1}$

$v_{o_1} = v_m - v_{l_1}$

(3.2.1.1) is true for $\beta = 1$

If $\beta > 1$, then.

For $i = 1$:

$v_l = v_{l_1} = x_0 - \sqrt{\frac{k}{p_{ask_1}}}$

(3.2.1.1) is true for $i = 1$

For $i = 2$:

$v_{o_1} = s_{ask_1}$

$v_{l_2} = x_1 - \sqrt{\frac{k}{p_{ask_2}}} = \sqrt{\frac{k}{p_{ask_1}}} - \sqrt{\frac{k}{p_{ask_2}}}$

$v_l = v_{l_1} + v_{l_2} = \left(x_0 - \sqrt{\frac{k}{p_{ask_1}}}\right) + \left(\sqrt{\frac{k}{p_{ask_1}}} - \sqrt{\frac{k}{p_{ask_2}}}\right) = x_0 - \sqrt{\frac{k}{p_{ask_2}}}$

(3.2.1.1) is true for $i = 2$

Assume (3.2.1.1) is true for $i = j < \beta$.

For $i = j + 1$:

We have:

$v_{l_{j+1}} = x_k - \sqrt{\frac{k}{p_{ask_{j+1}}}} = \sqrt{\frac{k}{p_{ask_j}}} - \sqrt{\frac{k}{p_{ask_{j+1}}}}$

$$v_l = \left(x_0 \ - \sqrt{\frac{k}{p_{ask_j}}}\right) \ + \ \left(\sqrt{\frac{k}{p_{ask_j}}} - \sqrt{\frac{k}{p_{ask_{j+1}}}}\right) \ = \ x_0 - \sqrt{\frac{k}{p_{ask_{j+1}}}}$$

$$x_{j+1} \ = x_0 \ - \ v_l \ = \ \sqrt{\frac{k}{p_{ask_{j+1}}}}$$

Then (3.2.1.1) is true for $i \ = \ j+1$

By induction then (3.2.1.1) is true.

The algorithm stops at step 2 means:

$v_m \ - k_\beta \ < \ v_l = x_0 - \sqrt{\frac{k}{p_{ask_\beta}}} \ \leq \ v_m \ - k_{\beta-1}$.

Therefore, β is one of θ

Since θ is unique, proved in section (2.3), then the algorithm is optimal in this case.

Case 2: stop at step 5

This means either steps 4 has filled all the orders or there is no more ask orders to fill.

This falls within case 1 and case 2 in the above summary.

In both cases, our algorithm returns the optimal result.

3.2. Handling Market Orders

Market Order is a special order that doesn't have a limit price:

$p_{order} = \infty$ for BUY and $p_{order} = 0$ for SELL

Thus the *target price* would be the best available offer price in the order book

$$p_t = p_{offer} = \begin{cases} p_{ask} \text{ for BUY} \\ p_{bid} \text{ for SELL} \end{cases}$$

We then apply the same remaining calculations and procedures as outlined in Section 2.3 and 3.1.

4. Conclusion

We have proposed a way to integrate the traditional Order Book with an AMM liquidity pool, using the Unified Formula and the Unified Matching Algorithm.

This integration allows the exchange to match orders with the best available price at any moment, whether that price is from the AMM Liquidity Pool or the Order Book. The two system work together in tandem and result in the minimum slippage for traders.

As a result, Dradex exchange is able to provide low slippage Market Order, because the exchange doesn't have to rely entirely on market makers, it can tap into the AMM Liquidity Pool for liquidity. Even when there are no matching orders in the order book, the final slippage would be equivalent to that of a typical AMM exchange based on the Constant Product formula (e.g Uniswap, Pancakeswap).