On the practical implementation of 3pools

immediate

September 20, 2024

Abstract

No abstract

1 Introduction

A major problem in all of decentralized finance (DeFi) is the scarcity of liquidity. One of the most promising strategies to overcome the limitations is the concept of concnetrated liquidity as introduced by Uniswap in 2021 with Uniswap v3. I will review the key characteristics of that concept and discuss what limits the implementation.

2 A concentrated liquidity pool

As described before, a concentrated liquidty position is described by a position that is very similar to a Uniswap v2 position, albeit with increased liquidity. The formula describing a position is given by

$$(x_A + Lc_A)\,(x_B + Lc_B) = L^2 \tag{1}$$

There are two (related) main differences with regards to Uniswap v2: (i) the possible price that a positions covers does not extend from $0 \to \infty$ and (ii) the virtual liquidity for positions in range is higher.

2.1 Key features of a single CL position

I will quickly summarize the key features of a CL positions and what type of requirement this has on a possible implementation. First of all, let us have a look at the equation again

$$(x_A + Lc_A)\,(x_B + Lc_B) = L^2\ . \tag{2}$$

This is one equation and it contains five variables. For this equation to have a unique solution, one needs to fix 4 variables. The most standard implementation is to start with specifying the two variables, c_A and c_B. It is straightforward to relate those to a maximum and minimum price according to

$$\left(x_A + L\frac{1}{\sqrt{p_{\max}}}\right)(x_B + L\sqrt{p_{\min}}) = L^2 \ . \tag{3}$$

The position is active if the current price $p = (x_B + L\sqrt{p_{\min}})/(x_A + L\frac{1}{\sqrt{p_{\max}}})$ is $p_{\min} \leq p \leq p_{\max}$. Most often, this price is used as a third condition fixing the above discussed ratio. This leaves one more thing to fix. A standard implementation is that the user now specifies x_A or x_B or maybe the total amount of funds invested $x_{\text{tot}} = px_A + x_B$. Now this allows to calculate the liquidity L and the equation is fully specified.

There are many more useful statements one can make and I just summarize some of them. One of them is that we can express the actual number of assets at a given price as

$$\begin{aligned} x_A &= L\left(\frac{1}{\sqrt{p}} - \frac{1}{\sqrt{p_{\max}}}\right) \\ x_B &= L\left(\sqrt{p} - \sqrt{p_{\min}}\right) \ . \end{aligned} \tag{4}$$

This also shows how a code can fix L in very simple terms. If you have a price p and decided on the range parameters $p_{\max}$ and $p_{\min}$, one can choose the amount x_A to deposite and L can easily be determined. From that one can easily fix x_B.

2.1.1 Adding Uniswap v2 positions

Let's first ask how a Uniswap v2 pool ($c_A = c_B = 0$) works with multiple positions involved. For concreteness, we consider two positions.

$$\begin{aligned} x_{A1}x_{B1} &= L_1^2 \\ x_{A2}x_{B2} &= L_2^2 \end{aligned} \tag{5}$$

The token is at every moment in time trading at a price

$$p = \frac{x_{B1}}{x_{A1}} = \frac{x_{B2}}{x_{A2}} \tag{6}$$

Let's assume we can just add the quantities

$$\begin{aligned} X_A &= x_{A1} + x_{A2} \\ X_B &= x_{B1} + x_{B2} = p\,(x_{A1} + x_{A2}) \\ L &= L_1 + L_2 = \sqrt{p}\,(x_{A1} + x_{A2}) \end{aligned} \tag{7}$$

Taking all together, it is easy to see that

$$X_A X_B = L^2 \tag{8}$$

which means it is very easy to add and also take out positions. It is easy to check that

$$\frac{X_B}{X_A} = \frac{x_{B1} + x_{B2}}{x_{A1} + x_{A2}} = \frac{p\,(x_{A1} + x_{A2})}{x_{A1} + x_{A2}} = p \tag{9}$$

It is easy to extend this to the case of having N different positions. In that case we have $X_A = \sum_{i=1}^{N} x_{Ai}$, $X_B = \sum_{i=1}^{N} x_{Bi}$, and $L = \sum_{i=1}^{N} L_i$. Overall, this combined position obeys all the same math as a single position which makes managing this very simple.

2.1.2 Adding Uniswap v3 positions

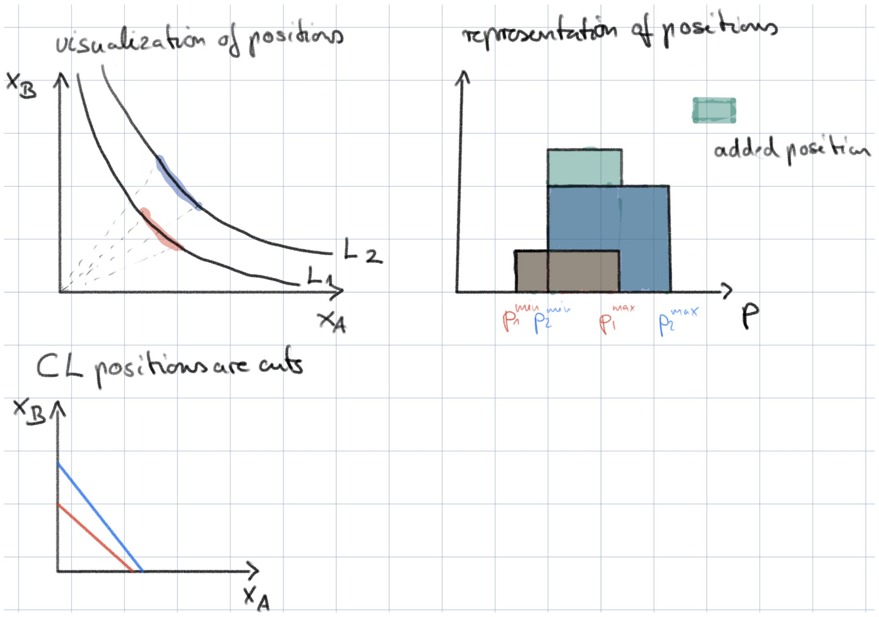


Figure 1: LHS: connection between a v2 and a v3 position; RHS: one possible representation of positions, this is what uniswap uses.

There is actually nothing new in adding Uniswap v3 positions once we identify x_A with $x_A + Lc_A$. There is, however, one major problem, which is that every position has an individual range. Let's assume that we have N total positions. We consider a range which at the lower end is p_b and at the upper end it is p_t and we are currently at an active price $p_b < p < p_t$. Out of the N positions, there are m positions in range (I assume the ones in range are in order $1 \to m$, the out of range ones are $m + 1 \to N$). This implies that within this range we can also have

$$\left(X_A + L\frac{1}{\sqrt{\tilde{p}_{\max}}}\right)\left(X_B + L\sqrt{\tilde{p}_{\min}}\right) = L^2 \tag{10}$$

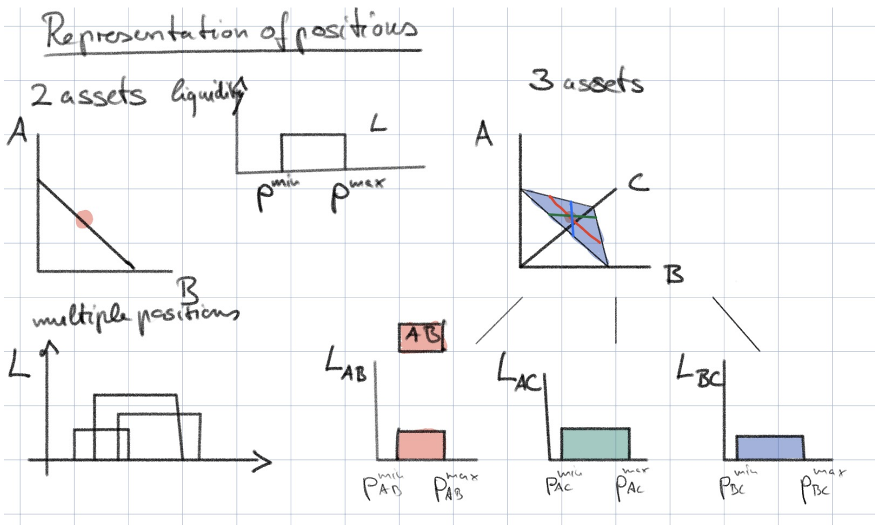


Figure 2: Difference in the representation of a position between two assets and three assets.

with $X_A = \sum_{i=1}^{m} x_{Ai}$, $X_B = \sum_{i=1}^{m} x_{Bi}$, $L\frac{1}{\sqrt{\tilde{p}_{\max}}} = \sum_{i=1}^{m} L_i \frac{1}{\sqrt{p^i_{\max}}}$, $L\sqrt{\tilde{p}_{\min}} = \sum_{i=1}^{m} L_i \sqrt{p^i_{\min}}$, and $L = \sum_{i=1}^{m} L_i$. Now within that range, this is the same swapping logic as a single position.

There is a way to implement this in a continuous fashion without ticks but it is computationally more demanding. One has to optimize. On the other hand, it is a basic ordering exercise that can in principle be done.

3 Pools made of three assets

I am now moving to a balancer style implementation of concentrated liquidity using three assets. The formula describing this reads

$$(x_A + Lc_A)\,(x_B + Lc_B)\,(x_C + Lc_C) = L^3 \tag{11}$$

The case of a conventional v2 implementation ($c_A = c_B = c_C = 0$) is very simple and follows the same rules as a pool of two assets. One can add the assets in the exact same way for multiple positions meaning. Before I go into those details, let us first determine some basics. The above equation is an equation of 7 variables. This means we need to fix six of them to have a unique position. A good choice would be as follows: We have in principle three prices, $p_{AB} = (x_B + Lc_B) / (x_A + Lc_A)$, $p_{AC} = (x_C + Lc_C) / (x_A + Lc_A)$, and $p_{BC} = (x_C + Lc_C) / (x_B + Lc_B)$. It is easy to see that only two of them are independent meaning getting two prices, for instance p_{AB}, and p_{AC} is sufficient and p_{BC} follows. Having specified two prices, we have four more variables to fix. In the spirit of a standard CL pool, it seems natural to choose, c_A, c_B, and c_C which are also related to price ranges as explained elsewhere, meaning we have fixed five parameters in total. Now one could fix the liquidity parameter L or, more naturally, the amount of tokens x_A (or the sum of all assets or whatever). In the spirit of how Uniswap v3 characterizes positions it would be most natural to represent it as p_{AB}, p_{AC}, L, c_A, c_B, and c_C. Mostly for notational purposes I will introduce the abbreviation $L_A = Lc_A$, $L_B = Lc_B$, and $L_C = Lc_C$.

It turns out, that it is relatively straightforward to access all the quantities characteristic of a positions just from

The question now is how we can characterize a position. For a single position of two assets the required information was L, the lower price $p_{\min}$, the upper price $p_{\max}$. Now the corresponding information is contained in a box, see Fig. 2. Within such a box, one has the liquidity L which then has to be converted into a position for respective swaps. Next step is how to make that most efficiently.

A first insightful exercise is to express all the single underlying assets as a function of the liquidity and the prices. Elementary manipulations lead to

$$\begin{aligned} x_A &= \left(\frac{1}{p_{AB}p_{AC}}\right)^{1/3} L - L_A \\ x_B &= \left(\frac{p_{AB}^2}{p_{AC}}\right)^{1/3} L - L_B \\ x_C &= \left(\frac{p_{AC}^2}{p_{AB}}\right)^{1/3} L - L_C \end{aligned} \tag{12}$$

as the equations. This implies that we can formulate three effective positions. For a swap $A \leftrightarrow B$ we get

$$\left(x_A + \frac{L_{AB}}{\sqrt{p_{AB}^{\max}}}\right)\left(x_B + L_{AB}\sqrt{p_{AB}^{\min}}\right) = L_{AB}^2 \tag{13}$$

where L_{AB}, $p_{AB}^{\max}$, and $p_{AB}^{\min}$ are all functions of the current p_{AB} and p_{AC} according to

$$\begin{aligned} L_{AB}\left(p_{AB}, p_{AC}\right) &= L\left(\frac{p_{AB}}{p_{AC}^2}\right)^{1/6} \\ p_{AB}^{\max}\left(p_{AB}, p_{AC}\right) &= \left(\frac{p_{AB}}{c_A^6 p_{AC}^2}\right)^{1/3} \\ p_{AB}^{\min}\left(p_{AB}, p_{AC}\right) &= \left(\frac{c_B^6 p_{AC}^2}{p_{AB}}\right)^{1/3} \quad . \end{aligned} \tag{14}$$

We can do the same for swaps between $A \leftrightarrow C$ and find

$$\left(x_A + \frac{L_{AC}}{\sqrt{p_{AC}^{\max}}}\right)\left(x_C + L_{AC}\sqrt{p_{AC}^{\min}}\right) = L_{AC}^2 \tag{15}$$

where L_{AC}, $p_{AC}^{\max}$, and $p_{AC}^{\min}$ are all functions of the current p_{AB} and p_{AC} according to

$$\begin{aligned} L_{AC}\left(p_{AB},p_{AC}\right) &= L\left(\frac{p_{AC}}{p_{AB}^2}\right)^{1/6} \\ p_{AC}^{\max}\left(p_{AB},p_{AC}\right) &= \left(\frac{p_{AC}}{c_A^6 p_{AB}^2}\right)^{1/3} \\ p_{AC}^{\min}\left(p_{AB},p_{AC}\right) &= \left(\frac{c_C^6 p_{AB}^2}{p_{AC}}\right)^{1/3} . \end{aligned} \tag{16}$$

We can do the equivalent thing for $B \leftrightarrow C$ and find

$$\left(x_B + \frac{L_{BC}}{\sqrt{p_{BC}^{\max}}}\right)\left(x_C + L_{BC}\sqrt{p_{BC}^{\min}}\right) = L_{BC}^2 \tag{17}$$

where L_{BC}, $p_{BC}^{\max}$, and $p_{BC}^{\min}$ are all functions of the current p_{AB} and p_{AC} according to

$$\begin{aligned} L_{BC}\left(p_{AB},p_{AC}\right) &= L\left(p_{AB}p_{AC}\right)^{1/6} \\ p_{BC}^{\max}\left(p_{AB},p_{AC}\right) &= \left(\frac{p_{AB}p_{AC}}{c_B^6}\right)^{1/3} \\ p_{BC}^{\min}\left(p_{AB},p_{AC}\right) &= \left(\frac{c_C^6}{p_{AB}p_{AC}}\right)^{1/3} . \end{aligned} \tag{18}$$

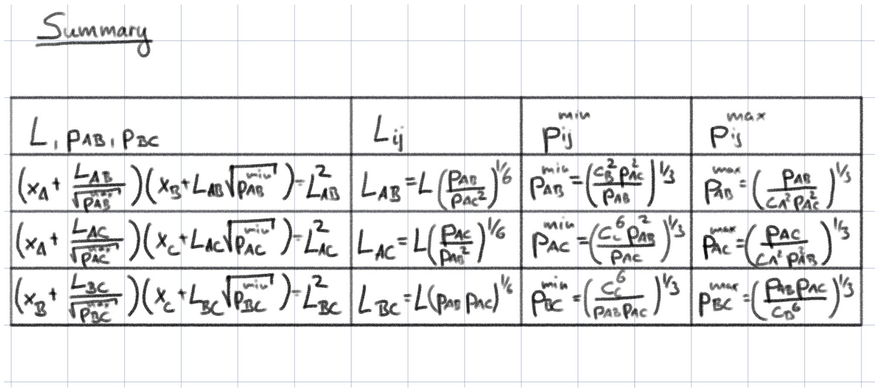
. In principle, in order to track the change in the individual swap positions, it suffices to track p_{AB} and p_{AC} and everything else is fixed. This is analoguous to a conventional pool where tracking the price is sufficient. So instead of tracking one quantity, we need to track two. There is one catch, though, which is that one has to track the positions as the prices move along since they change.

There are a number of conditions that are met in addition. For instance, the liquidity parameter is given by

$$\begin{aligned} \frac{L_{AB}}{\sqrt{p_{AB}^{\max}}} &= \frac{L_{AC}}{\sqrt{p_{AC}^{\max}}} \\ L_{AC}\sqrt{p_{AC}^{\min}} &= L_{BC}\sqrt{p_{BC}^{\min}} \\ L_{AB}\sqrt{p_{AB}^{\min}} &= \frac{L_{BC}}{\sqrt{p_{BC}^{\min}}} \end{aligned} \tag{19}$$

This also leads to conditions on the boundaries according to

$$\sqrt{\frac{p_{AC}^{\min}p_{AC}^{\max}}{p_{AB}^{\min}p_{AB}^{\max}p_{BC}^{\min}p_{BC}^{\max}}} = 1 . \tag{20}$$



Summary

L, p_{AB}, p_{BC}	L_{ij}	p_{ij}^{min}	p_{ij}^{max}
$\left(x_A + \frac{L_{AB}}{\sqrt{p_{AB}^{max}}}\right)\left(x_B + L_{AB}\sqrt{p_{AB}^{min}}\right) = L_{AB}^2$	$L_{AB} = L\left(\frac{p_{AB}}{p_{AC}^2}\right)^{1/6}$	$p_{AB}^{min} = \left(\frac{c_B^2 p_{AC}^2}{p_{AB}}\right)^{1/3}$	$p_{AB}^{max} = \left(\frac{p_{AB}}{c_A^2 p_{AC}^2}\right)^{1/3}$
$\left(x_A + \frac{L_{AC}}{\sqrt{p_{AC}^{max}}}\right)\left(x_C + L_{AC}\sqrt{p_{AC}^{min}}\right) = L_{AC}^2$	$L_{AC} = L\left(\frac{p_{AC}}{p_{AB}^2}\right)^{1/6}$	$p_{AC}^{min} = \left(\frac{c_C^6 p_{AB}^2}{p_{AC}}\right)^{1/3}$	$p_{AC}^{max} = \left(\frac{p_{AC}}{c_A^2 p_{AB}^2}\right)^{1/3}$
$\left(x_B + \frac{L_{BC}}{\sqrt{p_{BC}^{max}}}\right)\left(x_C + L_{BC}\sqrt{p_{BC}^{min}}\right) = L_{BC}^2$	$L_{BC} = L(p_{AB}p_{AC})^{1/6}$	$p_{BC}^{min} = \left(\frac{c_C^6}{p_{AB}p_{AC}}\right)^{1/3}$	$p_{BC}^{max} = \left(\frac{p_{AB}p_{AC}}{c_B^6}\right)^{1/3}$

Figure 3: Summary of how all the individual positions are related to the parameters of the position.

4 How to swap?

I will now investigate how a swap is taking place between two assets. I consider assets A and B but could use it for any pair. We use the equation

$$\left(x_A + \frac{L_{AB}}{\sqrt{p_{AB}^{\text{max}}}}\right)\left(x_B + L_{AB}\sqrt{p_{AB}^{\text{min}}}\right) = L_{AB}^2 \tag{21}$$

and deposit an amount δx_A and in return we get δx_B. The price before the swap, p_{AB}^i, the actual price for the swap, p_{AB}^{act}, and the price after the swap, p_{AB}^f read

$$\begin{aligned}
p_{AB}^i &= \frac{x_B + L_{AB}\sqrt{p_{AB}^{\text{min}}}}{x_A + \frac{L_{AB}}{\sqrt{p_{AB}^{\text{max}}}}} \\
p_{AB}^{\text{act}} &= \frac{x_B + L_{AB}\sqrt{p_{AB}^{\text{min}}}}{x_A + \frac{L_{AB}}{\sqrt{p_{AB}^{\text{max}}}} + \delta x_A} \\
p_{AB}^f &= \frac{x_B + L_{AB}\sqrt{p_{AB}^{\text{min}}} - \delta x_B}{x_A + \frac{L_{AB}}{\sqrt{p_{AB}^{\text{max}}}} + \delta x_A}
\end{aligned} \tag{22}$$

where

$$\delta x_B = \delta x_A \frac{x_B + L_{AB}\sqrt{p_{AB}^{\text{min}}}}{x_A + \frac{L_{AB}}{\sqrt{p_{AB}^{\text{max}}}} + \delta x_A} \tag{23}$$

It is interesting to check how the positions will be updated. First of all, the amounts x_A and x_B are updated in the above described way. Furthermore, the price goes from $p_{AB}^i \to p_{AB}^f$. Furthermore, the price p_{AC} goes from $p_{AC}^i =$

$(x_C + Lc_C)/(x_A + \frac{L_{AB}}{\sqrt{p_{AB}^{\max}}})$ to $p_{AC}^f = (x_C + Lc_C)/(x_A + \frac{L_{AB}}{\sqrt{p_{AB}^{\max}}} + \delta x_A)$ which also implies that p_{BC}^i goes to $p_{BC}^f = p_{AC}^f / p_{AB}^f$. Furthermore, the parameters in Fig. 3 will shift accordingly.

Again, if there was only a single position, that would be it and we could simply implement the swaps without any problems. But there are many and that will make life a lot harder. A sketch of a swap between two assets, A and B is shown in Fig. 4 and how it leads to the requirement to update to positions. In the end it comes down to managing three positions simultaneously as explained in Fig. 4. There are different strategies that probably and I first want to sketch how Uniswap v3 handles this in the simpler case of just one pool.

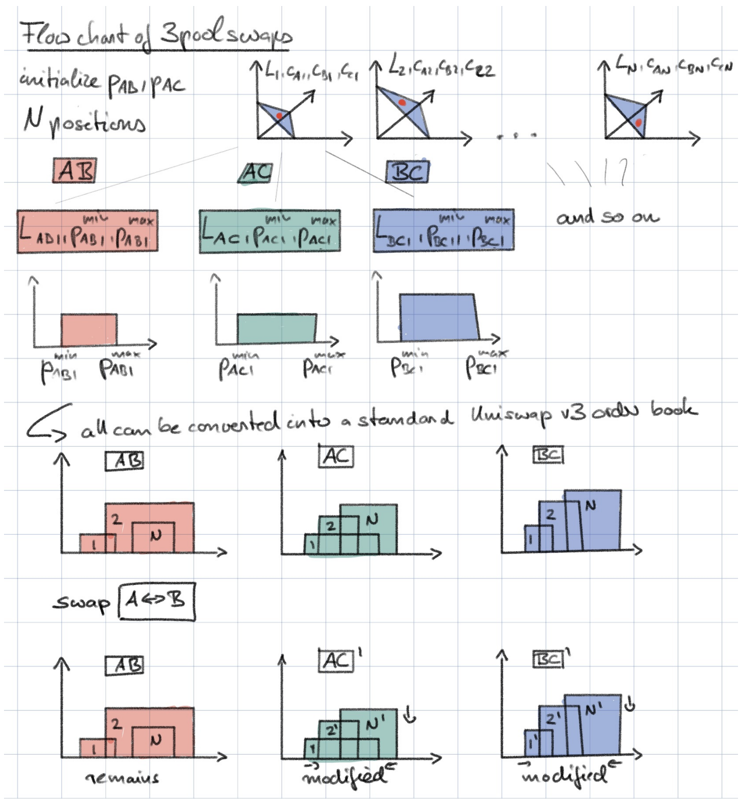

Flow chart of 3 pool swaps
initialize p_{AB}, p_{AC}
N positions
$L_1, c_{A1}, c_{B1}, c_{C1}$
$L_2, c_{A2}, c_{B2}, c_{C2}$
$L_N, c_{AN}, c_{BN}, c_{CN}$
AB
AC
BC
$L_{AB1}, p_{AB1}^{min}, p_{AB1}^{max}$
$L_{AC1}, p_{AC1}^{min}, p_{AC1}^{max}$
$L_{BC1}, p_{BC1}^{min}, p_{BC1}^{max}$
and so on
p_{AB1}^{min} p_{AB1}^{max}
p_{AC1}^{min} p_{AC1}^{max}
p_{BC1}^{min} p_{BC1}^{max}
all can be converted into a standard Uniswap v3 order book
1 2 N
swap A<>B
AC'
BC'
1' 2' N'
remains
modified
modified

Figure 4: Flow chart of a swap.

5 Uniswap handling of multiple positions

5.1 Managing multiple positions

The main question here is how to efficiently manage a number of positions. To attempt to answer that I want to review how Uniswap v3 does that in the case of two assets. I will consider the case of two assets, explain the difficulties and how to solve them. There is a position 1 described by

$$\begin{aligned}\left(x_{A1} + L_1 \frac{1}{\sqrt{p_1^{\max}}}\right)\left(x_{B1} + L_1 \sqrt{p_1^{\min}}\right) &= L_1^2 \\ \left(x_{A2} + L_1 \frac{1}{\sqrt{p_2^{\max}}}\right)\left(x_{B2} + L_2 \sqrt{p_2^{\min}}\right) &= L_2^2 \, . \end{aligned} \tag{24}$$

I discussed above how to add two positions. However, there is a condition for this addition to be meaningful which is that both positions are in range. Uniswap v3 positions are projected positions of Uniswap v2 where specific parts are cut out. For the case of having two positions, this is illustrated in Fig. **??** on the left hand side. Knowing, that we can fully represent a position by the lower price, $\sqrt{p^{\min}}$, the price p (if in range), the upper price $p^{\max}$, and the liquidity L, there is another way to represent it shown on the right hand side. For simplicity let's assume the price is in range for both positions. How can we perform the best swap. This is a maximization problem in the sense that we are looking for the maximal δx_B for a given δx_A.

5.2 How does Uniswap v3 record positions?

In Uniswap v3, the mathematics is done in terms of the so-called tick math. What this means is that a price is in fact always characterized by an integer according to

$$p = 1.0001^i \tag{25}$$

where i is the aforementioned integer. This implies that not every rational number can be represented but the grid is fine enough to represent prices. So what does this mean? It means that in order to characterize a position when we initialize it, we need four things: $p_{\min} = 1.0001^{i_l}$, $p_{\min} = 1.0001^{i_u}$, $p = 1.0001^{i_p}$, and L. If all of these are specified, the position is completely specified. The position is in-range if $i_l \leq i \leq l_u$.

For a single position, a swap proceed the following way:

The code has a current tick i that records the price. He chooses the two ticks adjacent to i as the interval it tries to perform the swap in. The bottom tick is called i_b (the corresponding price is p_b) and the top tick is called i_t (corresponds to p_t). It knows the position is in range from $i_l \leq i_b < i < i_t \leq i_u$. If we assume it wants to deposite an amount of δx_A into the pool and get out an amount

δx_B, it first checks whether the ticks have enough 'room' for the swap. The amount $\delta x_A^{\max}$ that can be added in the interval is given by

$$\delta x_A^{\max} = L\left(\frac{1}{\sqrt{p_b}} - \frac{1}{\sqrt{p_{\max}}}\right) - L\left(\frac{1}{\sqrt{p}} - \frac{1}{\sqrt{p_{\max}}}\right) = L\left(\frac{1}{\sqrt{p_b}} - \frac{1}{\sqrt{p}}\right) \tag{26}$$

If the amount $\delta x_A^{\max} > \delta x_A$ is can perform the swap inside the interval. It can easily compute the amount δx_B of tokens, release them, and adjust the price accordingly. No further action is required. If $\delta x_A^{\max} < \delta x_A$ one proceeds to lower the tick by one, *i.e.*, $i_b \to i_b - 1$ and repeats until the swap can be carried out. If the final i_b in this procedure ends up below p_l it simply means the swap cannot be completed.

5.3 How to record a collection of positions?

If there was only one position in a Uniswap v3 pool the logic of ticks would not help much and in principle one could manage the whole pool like a Uniswap v2 pool with very minor modifications. The problem in Uniswap v3 is that all positions are individual. This means that in principle finding a possible swap can be very costly. In an ideal situation, one would try to do the following things:

- Check what positions are in range and only consider them.
- Find the optimal way of using the liquidity of the individual positions. This is a non-trivial task since some positions will run out of liquidity upon moving the price.
- Perform the swap and update the positions.

Especially the latter two items require considerable computational power. To overcome this let us first analyze why this is no issue in Uniswap v2.

5.4 Conclusion

One of the problems of the Uniswap logic applied to a three pool is that we would always have to recalculate two order books after each swap, see Fig. 4. The one of the swapped pair would stay intact and the two other ones would have to be recalculated. It is unclear to me how costly this is. The updates were discussed in Sec. 4. **If, however, the updating is not an expensive operation, one could simply use Uniswap v3 and combine them in an appropriate way. A general minimally complicated algorithm looks like the one shown in Fig. ??. Possibly, we have to check Uniswap v4 whether this could be one of the so-called hooks they dicuss.**

6 Idea for a code

Fig. 5 shows a possible structure for a code that could integrate with Uniswap v3.

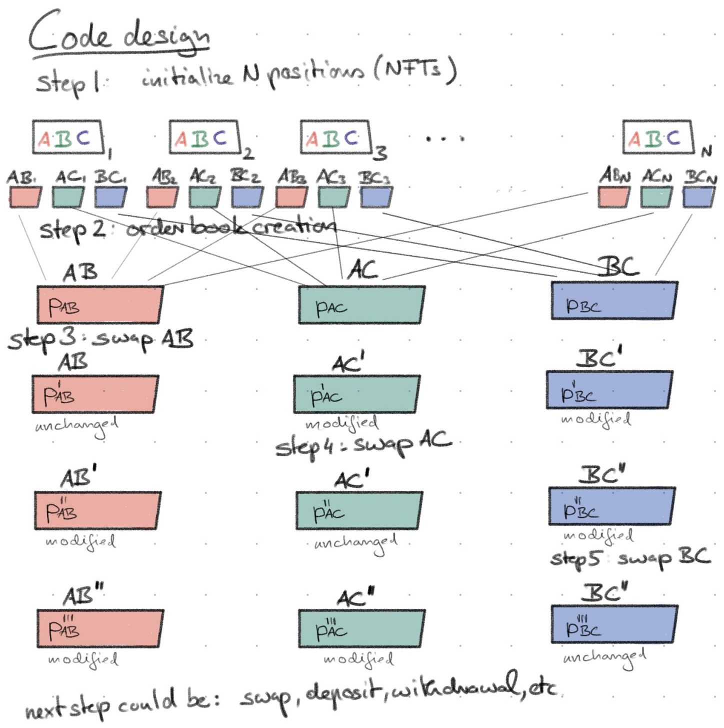


Figure 5: Visualization of a swap and the potential integration.