Dynamic Liquidity Concentration with an Oracle Supplied Price and Implied Volatility

Taylor Hulsmans

July 2021

Abstract

A constant products market maker algorithm is defined that utilizes the implied volatility and oracle price to dynamically concentrate liquidity to increase capital efficiency during periods of low expected volatility, while reducing impermanent loss during periods of high expected volatility.

1 Introduction

Since the popularization of Uniswap, the innovation challenges in the AMM space have been concerned with the minimization of impermanent loss for the liquidity provider, and the maximization of capital efficiency to reduce slippage for the swapper. Ideas upon solving these issues have revealed a sort of iron triangle. Capital efficiency amplifies impermanent loss, while negating impermanent loss exposes other loss vectors such as arbitrage loss, reducing LP incentives to provide capital at all. To work around these opposing constraints, a novel approach to liquidity concentration is developed and formed into a market making algorithm.

The solution presented in this paper utilizes the implied volatility of an asset, a sort of weather prediction of the volatility based on the behaviour of the assets overlayed options market, to dynamically adjust the amount of volume around the market price. We should note that while it is possible, and perhaps wiser, to use this market price for the assets as are physically traded, and make a DAMM as opposed to an AMM, this is not the subject of this paper. Here the oracle market price and implied volatility is only utilized in the concentration of liquidity- The physical price paid by the user is still forged in the algorithm through arbitragers.

2 Concentrated Liquidity: What is it?

In traditional market makers, it is up to the users to show up and provide buy and sell orders at certain prices to generate depth, or volume available at a

certain price, to match trades. Theoretically speaking, it makes sense that a lot of buy orders exist at low prices to get deals, and a lot of sell orders exist and high prices to get a premium. Regardless of where they are on the price curve, these opportunities, or degrees of freedom to swap the assets are called *liquidity*. It is by the matching of these orders, most often at the highest buy order, and lowest sell order is the market price formed.

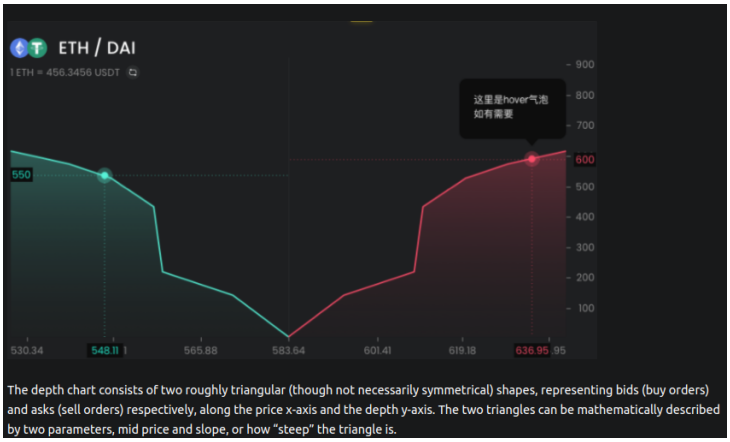

Figure 1: An orderbook based market maker depth chart

In contrast to traditional market makers, Automated ones often do not have the privilege of defining where their buy and sell orders exist on this price curve. When a liquidity provider in UniswapV2 for example, deposits the assets, in effect a bunch of buy and sell orders are scattered uniformly along the price curve. When the goal is to provide liquidity at the market price, a lot of capital that could be preventing slippage is otherwise left unused, this is referred to as capital efficiency of an automated market maker.

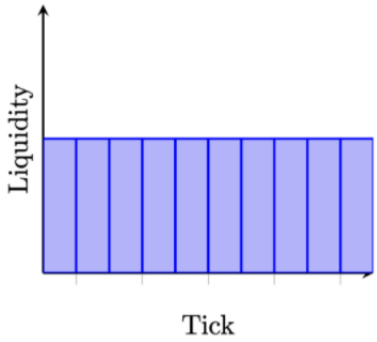

Figure 2: The UniswapV2 Liquidity at price (tick) graph

This reasoning lies at the heart of Uniswaps transition from v2 to v3, which allows users to set custom price ranges on individual liquidity positions, uniformly distributing it from their p_a to p_b. instead of 0 to ∞. Over many liquidity positions, these have a habit of forming a normal like distribution of liquidity around the market price. As we can see, liquidity becomes alot more

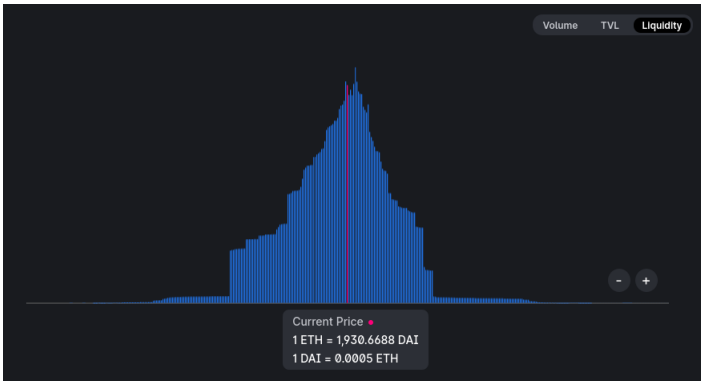

Figure 3: The UniswapV3 Liquidity at price

concentrated around the market price of an asset, providing much more depth to draw from before evoking a unit of slippage, becoming much more capital efficient. However, the downside to the LP in concentrating their liquidity is exposing themselves to a greater degree of impermanent loss. while UniswapV3 compensates users with greater proportion of the fees for doing this, past prices do not predict future prices and most pairs in crypto are inherently volatile, leaving it up to the user to really guess a liquidity concentration range and hope for the best and engage in active management when things go wrong. I'd also add that manual concentration may not be the best tool from a behavioural economics perspective, as positions fall out of range, a user may engage in the sunk cost fallacy, choosing not to make their impermanent loss permanent and simply hope that it will fall back into range, while loosing out on many days of transaction fees. These consideration are at the heart of this protocol, what would be a way to bring back the passive nature of liquidity provisioning, while increasing liquidity concentration while reducing impermanent loss. The solution here is rather intuitive- we make liquidity concentration dynamic!

3 A new Approach to Liquidity Concentration

The Black-Scholes options pricing model was a revolution in financial mathematics. Not only is it one of the best ways to determine the fair price of an option, but it can be turned on its head to infer the markets expectations on the future volatility of an asset, a sort of weather forecast for price stability.

We start by inheriting the Chainlink price and implied volatility to construct the probability density function for the asset. This is where we expect prices

to fall in the given time frame. In accordance with the financial 6 sigma event tolerance rule, we calculate the minimum and maximum price band for the asset (could be made lower with governance voting on a per asset basis).

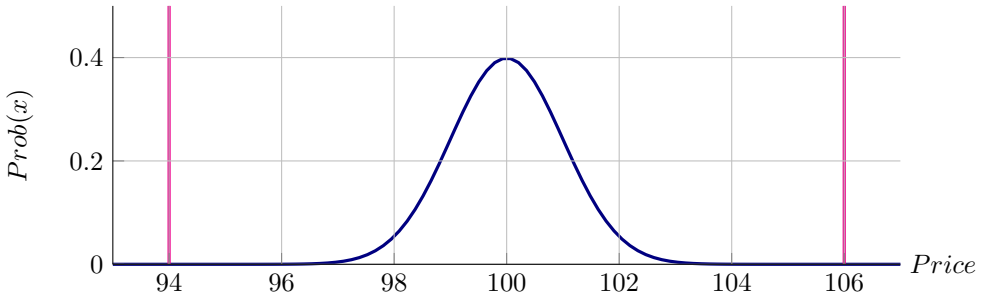

Figure 4: An asset with an average price of 100 and volatility of ($\sigma = 1$). bars denote -6 and +6 standard deviations away respectively

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) \tag{1}$$

We than find a liquidity distribution function that fills available liquidity under its curve. We solve for a in the following equation

$$\frac{a}{\sigma\sqrt{2\pi}} \int_{\mu-6\sigma}^{\mu+6\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \mathrm{d}x = \sqrt{q_a q_b} \tag{2}$$

Which, rather thankfully, is roughly equal to the liquidity

$$\frac{\sqrt{q_a q_b}}{erf(3\sqrt{2})} = a \approx \sqrt{q_a q_b} \tag{3}$$

We than take A and draw a liquidity concentration function reminiscent of Uniswapv3 'Tick space' and allocate the available liquidity along this curve. Following our example, let us use $q_a = 1000$, $q_b = 10$, $\sigma = 1$ and $\mu = 100$

$$Depth(p) = \frac{\sqrt{q_a q_b}}{\sigma\sqrt{2\pi}} e^{\frac{-(p-\mu)^2}{2\sigma^2}} \tag{4}$$

3.1 A swap

For example, let us consider a user who wishes to purchase one unit of q_b. Our depth graph than is subdivided into *ticks*, in effect buckets of liquidity where p_2 - p_1 is 0.0001% or one basis point. Each bucket has a price that is defined as its starting point along the depth-price curve. To begin our swap we calculate that

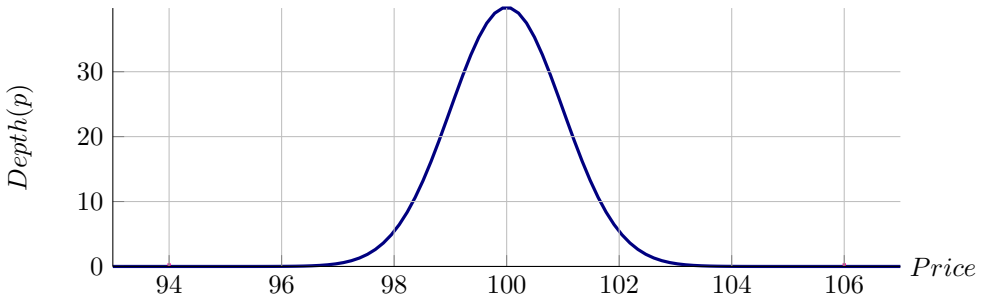

Figure 5: The Depth of the chart represents how much liquidity is available at that specific price point

amount of liquidity available at the first price by first calculating basis point price change of the asset, and integrating the curve over it

$$\int_{\mu}^{\mu+bp} Depth(p) = \int_{\mu}^{\mu+bp} \frac{\sqrt{q_a q_b}}{\sigma\sqrt{2\pi}} e^{\frac{-(p-\mu)^2}{2\sigma^2}} dp \tag{5}$$

Following our example as we traverse the first tick

$$\int_{100}^{100.01} Depth(p) = \int_{100}^{100.01} \frac{\sqrt{q_a q_b}}{1*\sqrt{2\pi}} e^{\frac{-(p-100)^2}{2*1^2}} dp = 0.398936 \tag{6}$$

For this tick, ≈ 0.4 liquidity points are exposed. We denote this L_t the quantities of each asset at this tick is given as q_{at} and q_{bt} respectively. Given the following system of equations we calculate the quantity swapped for this *tick*

$$L_t = \sqrt{q_{at} q_{bt}} = 0.398936 \tag{7}$$

$$P_t = \frac{q_{at}}{q_{bt}} = 100 \tag{8}$$

And we are trying to get q_b out, one pays

$$\sqrt{q_{at}\frac{q_{at}}{P_t}} = \sqrt{q_{at} q_{bt}} = L_t \tag{9}$$

$$\sqrt{q_{at}\frac{q_{at}}{100}} = \sqrt{1000 * 10} = L_t \tag{10}$$

$$q_a = 3.98936 \tag{11}$$

$$q_b = 0.0398936 \tag{12}$$

around 3.98936 units of a to receive around 0.0398936 units of b, we than traverse to the next tick

$$\int_{100.01}^{100.02} Depth(p) = \int_{100.01}^{100.02} \frac{\sqrt{q_a q_b}}{1*\sqrt{2\pi}} e^{\frac{-(p-100)^2}{2*1^2}} = 0.398896 \tag{13}$$

and than solve the system of equations again to determine the swap

$$L_t = \sqrt{q_{at} q_{bt}} = 0.398896 \tag{14}$$

$$P_t = \frac{q_{at}}{q_{bt}} = 100.01 \tag{15}$$

$$\sqrt{q_{at} \frac{q_{at}}{100.01}} = 0.39889 \tag{16}$$

$$q_a = 3.98916 \tag{17}$$

$$q_b = 0.0398876 \tag{18}$$

We can note that over this tick, supplies 3.98916 units of q_a for 0.0398876 units of q_b. This process continues until we allocate 1 full unit of q_b for our swapper. Generally speaking the price for an amount determined by the other is given by the set of equations

$$\sum_{i=\mu}^{x} = \sqrt{L_i^2(\mu + i(bp))} = q_{a-required} \tag{19}$$

$$\sum_{i=\mu}^{x} = \sqrt{\frac{L_i^2}{\mu + i(bp)}} = q_{b-bought} \tag{20}$$

Where

$$L_i = \int_{\mu+i(bp)}^{\mu+2i(bp)} \frac{\sqrt{q_a q_b}}{1 * \sqrt{2\pi}} e^{\frac{-(p-\mu)^2}{2*1^2}} dp \tag{21}$$

whee bp is the one basis point of μ, q_a and q_b are the total pool liquidity, and q_{ai} and q_{bi} is the liquidity of the current tick

4 Conclusion

An Automated Market maker that dynamically concentrates liquidity during low expected volatility to increase capital efficiency, while doing the converse during periods of high expected volatility to reduce impermanent loss was derived. It begins by inheriting the oracle price and implied volatility to generate a normal distribution around the average price to -6 to +6 standard deviations, as found through the implied volatility, as is standard in finance. The total liquidity of the pool is than distributed along this curve in buckets of 1 basis point called ticks. A person wishing to swap tokens in the pool starts at the point of the curve the pool is at (not nessissarily the oracle price) and takes from each successive bucket at the price of the bucket until their order is filled. The algorithm routinely updates itself, drawing parameters from the oracle as is economical to reform the curve. Parameters like the 6 σ rule could be governance set on a per pool basis, or even divided into pools with different overal risk tolerance.