



The next generation DEX on SUI

WABKA

I	AN OVERVIEW	4
	1	WHY A DECENTRALIZED EXCHANGE ?
	2	WHY WABKA ?
II	UNIV4: THE CORE MODEL	7
	3	CONCENTRATED LIQUIDITY
	3.1	Pool basics
	3.2	Virtual Reserve Equation in Wabka DEX
	4	LIQUIDITY PROVIDER INTERACTION
	4.1	Providing Liquidity in Wabka
	5	TRADER INTERACTION
III	CUSTOMIZATION: HOOKS	17
	6	THE TWAMM
	6.1	An overview and key points
	6.2	Large orders on current AMM
	6.3	Analogy to traditionnal finance
	6.4	The algorithm
	6.5	Low slippage and the impact of arbitrage
	6.6	TWAMM LP, Arbitrageurs and validators balance.
	6.7	Sandwich attacks: A solution attempt
IV	OTHER TYPES OF HOOK	28
	7	LIMIT ORDERS
	8	DYNAMIC FEES WITH VOLATILITY ACCUMULATOR
30	9	POWER PERPETUALS TO HEDGE IL
	9.1	Power Perpetuals Overview
	9.2	Hedging LP with Power Perpetuals in Wabka on SUI
	10	INNOVATIVE ORACLE
	10.1	Oracles and their importance
	10.2	Oracles manipulations

	10.3	Pyth oracle on SUI	42
	10.4	Oracle as an optionnal hook: the geometric mean oracle	43
V	CUSTOMIZABLE AMM'S		45
	11	THE STABLE COIN SWAP FUNCTION	46
VI	THE TEAM		49
VII	OUR VISION OF WABKA		52
	11.1	Short term	53
	11.2	Medium term	53
	11.3	Long Term Vision	53
VIII	BIBLIOGRAPHY		54
IX	SUMMARY		56



AN OVERVIEW



1 . WHY A DECENTRALIZED EXCHANGE ?

At the heart of our decision to create Wabka, our innovative decentralized exchange (DEX), lies our firm belief in the potential and forthcoming popularity of the SUI blockchain. **We foresee SUI rapidly becoming a hub for DeFi activity**, attracting a vast and diverse user base, driven by its unique and compelling features conducive to DeFi applications. **Also, we believe that DEXES will continue to rise in comparison to CEXES due to Privacy and Anonymity as well as the recent events around the crisis of Cexes like Binance**

The goal is to leverage SUI's Unique Features:

High Transaction Speed and Low Cost: The remarkable transaction throughput and cost-effectiveness of SUI are fundamental for a DEX where efficiency and affordability are key. Wabka utilizes these aspects to offer an unparalleled trading experience, ensuring optimal operation for both liquidity providers and swappers.

Instant Transaction Finality and Reduced Latency: SUI's emphasis on instant transaction finality and reduced latency is vital in the fast-paced world of decentralized exchanges. Wabka leverages this to provide real-time trading experiences, catering to modern traders who demand swift and decisive market interactions.

Scalability Without Compromising Security: SUI's approach to scalability, achieved without sacrificing security, mirrors Wabka's vision for a reliable and expansive DeFi platform. Building on SUI, Wabka assures a secure trading environment that can scale to meet the demands of a growing user base.

As SUI grows in popularity, Wabka will be perfectly positioned to serve this expanding ecosystem, offering a user-friendly, secure, and efficient platform for the next wave of DeFi enthusiasts.

2. WHY WABKA ?

Our project presents an innovative decentralized exchange tailored for the rapidly expanding SUI blockchain ecosystem. We envision a future where SUI's popularity continues to soar, attracting a diverse array of DeFi users, each with unique needs and interests. Our DEX is designed to be the foundation of this burgeoning landscape, offering a user-centric platform that caters to the varied preferences of the DeFi community.

Here is how Wabka DEX excels at being user-friendly design, adapting to the diverse needs and creativity of its users and aiming at reducing their losses and costs:

For Liquidity Providers (LPs): Our DEX introduces a groundbreaking approach for liquidity providers. Inspired by the **concentrated liquidity model of Uniswap v3** and enriched with the **advanced features of Uniswap v4 hooks**, our platform offers LPs an unparalleled level of control and flexibility. LPs can create pools in the Uniswap v3 style or **stablecoin pools reminiscent of Curve**, adapting their strategies to market conditions. Additionally, our unique fee structure, which adjusts in proportion to market volatility, provides LPs with extra earnings during turbulent market phases, thereby offering enhanced financial protection. **To further bolster LP security, we present options for hedging against impermanent loss, including innovative financial instruments like perpetuals and options.** Our DEX empowers LPs to select their liquidity range, accompanied by various strategic options for distribution, ranging from Gaussian to inverse Gaussian, ensuring optimized liquidity provision.

For Swappers: The swapper experience on our DEX is equally revolutionary. **We provide an array of pool hooks, including the Time Weighted Average Market Maker (TWAMM), available on all pools to mitigate price impact and ensure equitable trading conditions.** This feature is particularly beneficial for swappers, safeguarding them against significant market shifts and reducing **slippage**. By integrating these advanced mechanisms, our DEX ensures that swappers have access to a variety of trading options, allowing them to execute transactions that align best with their strategies and market outlook.

On top of this, WABKA will adopt the innovative **zk login** to enhance the SUI ecosystem signature and to assert it's beginner-friendly aspect.

Our DEX is not merely a trading platform; **it is a nexus of innovation and security**, fostering a DeFi environment that is both creative and robust. By aligning with the SUI blockchain's commitment to scalability, security, and decentralization, our project is poised to revolutionize the DeFi space, making it more accessible, efficient, and secure for a wide range of users. We are not just building a DEX; we are preparing the groundwork for a future where DeFi is integral to every user's digital life.



UNIV4: THE CORE MODEL



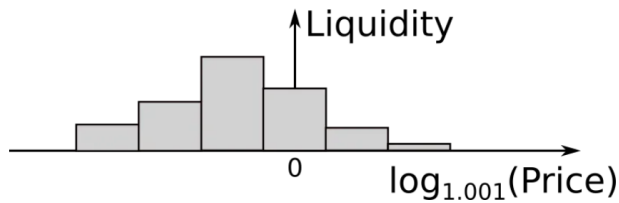
3. CONCENTRATED LIQUIDITY

WABKA embraces the advanced model of a concentrated liquidity Automated Market Maker (AMM), a concept that, while established in the DeFi space, warrants a brief recapitulation to understand its integration and innovation in our platform.

3.1. POOL BASICS

At the heart of creating a liquidity pool in WABKA is the selection of two distinct assets: the Base asset (Asset X) and the Quote asset (Asset Y). For instance, in a SUI/USDC pool, SUI is the Base asset (X), and USDC is the Quote asset (Y). The price in our model is conventionally expressed as the value of the Base asset in terms of the Quote asset.

To comprehend the current price of X in terms of Y, it is essential to examine the structure of our pool. We visualize the pool using a coordinate system where the horizontal axis denotes the price, and the vertical axis represents liquidity. At this moment, we will treat 'liquidity' as an arbitrary quantity, with its precise definition and implications to be elaborated on later in the report.



An illustration of a coordinate system that represents the quantity of liquidity locked in a certain price range represented by ticks.

The graphical representation shown below illustrates the theoretical concept of segmented liquidity across various intervals in WABKA's concentrated liquidity model. In this model, liquidity providers (LPs) have the discretion to select specific price ranges where they wish their tokens to be actively engaged in trades. The process of defining these price ranges begins with the establishment of a continuous price spectrum from 0 to ∞ . This spectrum is then discretized into distinct segments using price ticks. The segments each have their own reserve of tokens. The methodology for setting up these price ticks follows a specific scheme: the i^{th} tick represents a price $p(i)$, where $p(i)$ is determined by

$$p(i) = (1 + s)^i$$

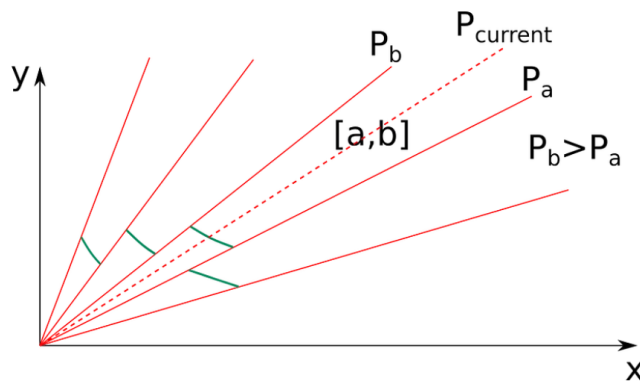
The decision to space discretization points, or price ticks, exponentially in WABKA's model is grounded in a common practice within DeFi. This approach balances the need for precision in price representation with practical considerations of liquidity management. The variable s serves as a crucial hyperparameter that defines the tick step. An example of tick spacing is $s = 0001$ which explains the $\log_{1.001}$ horizontal axis in the graph. In WABKA, customization is offered to pool creators, allowing them to adjust the density of these ticks by choosing the spacing rate. By modifying the tick spacing, pool creators can effectively control the granularity of price ranges within their pool. Opting for fewer ticks results in broader price ranges covered by each reserve, and conversely, a higher number of

ticks leads to more narrowly defined price intervals. This flexibility empowers pool creators in WABKA to tailor the liquidity distribution in a way that aligns with their specific strategies and the unique characteristics of their pools. It is important to note the direct correlation between tick spacing and the fees received by liquidity providers in WABKA. As the spacing between ticks increases, resulting in wider reserves, the percentage of fees accrued by liquidity providers correspondingly decreases. This relationship underscores the strategic considerations that must be taken into account when setting up a pool. Pool creators must balance the desire for broader market coverage with the potential for fee generation, ensuring that the chosen tick spacing aligns with the overall objectives of the liquidity providers in their pool.

This preliminary overview sets the foundation for the discussions we will undertake, focusing on the specific interactions of liquidity providers or traders. However, to make it a bit more complete and rigorous, we must at least present the main math equations behind concentrated liquidity, the ones which will govern LPs and traders classical interactions. Indeed, the concept of concentrated liquidity is not novel in the DeFi space, and its intricate workings needs a closer examination. The following section aims to unpack the core mathematical principle that governs this system. We will not delve into exhaustive details of each mathematical interaction but rather focus on presenting the fundamental equations that underpin concentrated liquidity, providing insights into its operational mechanics.

3.2. VIRTUAL RESERVE EQUATION IN WABKA DEX

Situated between each pair of consecutive price ticks lies a specific reserve. While the price/liquidity coordinate system offered an intuitive approach, it lacked rigor regarding liquidity and reserves. Adopting a more technically-oriented perspective, we consider a coordinate system that offers enhanced comprehension of the underlying mechanics. In this system, the horizontal axis represents the virtual (or offsetted) amount of token X (the actual amount in the reserve plus an offset value), while the vertical axis corresponds to the virtual amount of token Y. This approach necessitates an explanation of what we mean by 'offset'.



Each of the lines represent a certain price, between two lines (the price range) a Constant product function takes place in the form of a portion of a hyperbola.

Through this perspective, the dynamics of the system become clearer, including the shifting of reserves, the progression of prices in relation to the virtual quantities, the transitions between different reserves, and the variation of prices within these reserves. This viewpoint offers a comprehensive understanding of the nuanced movements and adjustments within our concentrated liquidity model.

Let's denote x as the virtual quantity of asset X, and y for asset Y. At any given point (x, y) on our graph, the price of asset X in terms of asset Y is defined by the formula: $P = \frac{y}{x}$. Since the price is essentially a ratio, in this coordinate system, it is represented by an entire line $y = P * x$. This implies that if the virtual quantities change disproportionately, the price shifts; conversely, if they change proportionally, the price remains constant. As one can observe, there are specific prices that establish the boundaries of the green curves depicted in the graph. These green lines each symbolize liquidity of its reserve, and the boundaries they reach correspond to the price ticks defined earlier. For instance, in the graph, the prices P_b and P_a are boundaries, representing their respective tick. As a result, in the concentrated liquidity model, there are two types of prices to consider: the prices that form the boundaries of a reserve, and the prices possible within a reserve.

Do these two types of prices seamlessly integrate? Does applying the price formula to points on a boundary price line yield a consistent price? The answer is yes, thanks to the offsetting of quantities. These offsets are calculated to align smoothly with the boundary prices, let's delve deeper into the concept.

In our concentrated liquidity model, the x and y axes represent virtual quantities of the tokens. These are defined as $x = x_{\text{real}} + x_{\text{offset}}$ and similarly, $y = y_{\text{real}} + y_{\text{offset}}$. The real quantities at a certain point in the graph, are the respective total amount of tokens deposited in the active reserve at that point. The offset, on the other hand, is a virtual quantity, a constant whose primary purpose is to constrain the range of possible prices within its reserve. By implementing these offsets, we effectively lock each interval into a specific price range. This means that even if the real quantity of a token in a reserve drops to zero, the reserve still maintains a defined maximum and minimum price using the formula $P = y/x$.

This highlights the importance of differentiating between 'real' quantities in a reserve and 'virtual' quantities – giving rise to terms like 'real reserve' and 'virtual reserve'. It's important to note that all subsequent calculations related to a reserve will be based on its 'virtual' perspective. While the actual reserve is presented for easier understanding, it is not utilized in computational processes. This is because real quantities are already included in the virtual framework. The virtual setup offers a more precise and rigorous approach, as it incorporates offsets that specifically encapsulate price ranges within a reserve. This distinction ensures accuracy and relevance in our calculations.

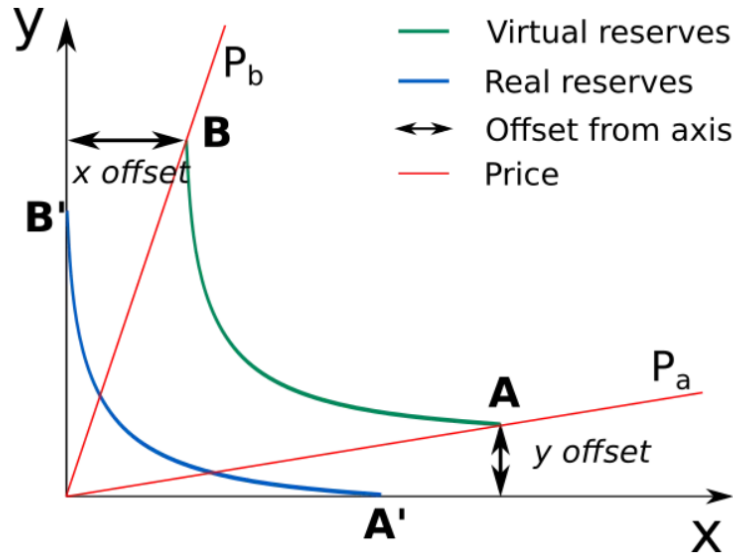
The dynamics of these virtual reserves are governed by a fundamental principle known as the constant product function, which incorporates the offsets in token quantities. Indeed, the equation for any given virtual reserve is defined as:

$$(x_{\text{real}} + x_{\text{offset}}) * (y_{\text{real}} + y_{\text{offset}}) = L^2$$

The symbol L is called the liquidity of the reserve, a distinct and quantifiable attribute of each reserve.

To illustrate the equation with a practical example, let's select a specific virtual reserve. This requires choosing two price ticks that define the boundaries of a reserve. Let P_a denote the price corresponding to "tick a" and P_b for "tick b". The interval between these two ticks, denoted as $[P_a, P_b]$, constitutes our chosen virtual reserve.

Let's derive all the quantities characterizing this virtual reserve: x_{real} will be the total amount of token X deposited by liquidity providers in the $[P_a, P_b]$ virtual reserve only, excluding the amount in other reserves. The equivalent for y_{real} follows the same principle.



A clear graphical illustration of the transition from the virtual quantities to the real quantities by subtracting the offset quantities.

Finding the offset is slightly more complex. Beginning with x_{offset} , we note that x_{offset} corresponds to the x-axis value at point B. Indeed, x_{offset} of a specific reserve is the virtual amount of token X in the reserve when the real amount is down to zero. Notice also that point B lies on the line representing the price ratio $y/x = P_b$. By using this price equation in conjunction with the virtual reserve equation at point B, where $x = x_{\text{offset}}$ and $y = y_{\text{real}} + y_{\text{offset}}$, and substituting the value of y, we can solve the system for x. The solution yields:

$$x = x_{\text{offset}} = \frac{L}{\sqrt{P_b}}$$

Using a similar approach for point A to determine y_{offset} , we arrive at the formula:

$$y_{\text{offset}} = L \cdot \sqrt{P_a}$$

Consequently, the virtual reserve equation for the reserve $[P_a, P_b]$ is expressed as:

$$(x_{\text{real}} + \frac{L}{\sqrt{P_b}}) \cdot (y_{\text{real}} + L \cdot \sqrt{P_a}) = L^2$$

Notice that the offset is proper to each reserve by depending on its boundaries, but also its liquidity L . How is L determined? The liquidity L of a virtual reserve is dynamic; it changes every time a liquidity provider contributes to the reserve (adding or removing). This change reflects the actual increase (or decrease) in the real amount of tokens within the reserve. L is rigorously computed by ensuring the validity of the virtual reserve equation after each provision of liquidity (or removal). It's recalculated after each such provision (or removal). And so are the offsets, depending on L ! Importantly, L remains constant during swaps, which is central to the functionality of the system. Indeed, we will later see that L remaining constant dictates the swap amounts. It governs the movement of token amounts within the reserve, and consequently, the price dynamics during swaps. On the graphs we previously discussed, the liquidity of a reserve can be interpreted as the distance of the green lines from the origin (this is an intuition, liquidity is not rigorously the distance to zero).

4. LIQUIDITY PROVIDER INTERACTION

4.1. PROVIDING LIQUIDITY IN WABKA

For someone aspiring to become a liquidity provider in Wabka, the initial step involves either selecting an existing pool to deposit in or creating a new pool. Let's consider the most complete scenario: creating a pool and depositing tokens. Imagine wanting to establish a SUI/USDC pool on Wabka. The process begins with choosing the tokens and setting the base fees for liquidity providers. This fee decision influences which ticks are available or disabled within the pool.

A unique aspect of Wabka is the ability to choose or create hooks for a pool at the time of its creation, though we will delve into this feature later. What about the pool's starting price? If a pool with the same token pair already exists, the new pool initializes at the current price of the other pool. If it's the first pool for these two selected tokens, the creator determines the starting price.

The next critical decision is the selection of a price interval for depositing liquidity. The chosen interval's boundaries automatically align with the nearest price ticks. If this interval spans multiple virtual reserves, Wabka offers a distinctive feature where users can select from various liquidity distribution models, including uniform, Gaussian, and inverse Gaussian distributions, all centered around the current price of the pool (in our particular example: the starting price).

A crucial point to note is the initial price setting. If the creator does not deposit liquidity within the active reserve, the reserve containing the starting price, then the pool remains inactive until another user contributes active liquidity. Conversely, setting a price that does not reflect the market rate and depositing in this active reserve can lead to immediate arbitrage, resulting in losses for the initial liquidity provider. Therefore, aligning the pool's starting price with the market price is advisable to avoid unfavorable arbitrage situations. Before finalizing their contribution, every liquidity provider (creator or not) is presented with a simulated visualization of the his liquidity distribution, allowing them to confirm that the proposed distribution aligns with their expectations and requirements.

For subsequent liquidity providers joining an already established pool, Wabka also offers the advantage of visualizing the price/liquidity graph. This feature provides an immediate grasp of the existing dynamics within the pool, aiding them in tailoring their liquidity provisioning strategy to current market conditions. Complementing this graphical representation in a more technical way, Wabka also displays the estimated Annual Percentage Rate (APR) and Annual Percentage Yield (APY) of their liquidity provision. This additional insight allows providers to make more informed decisions by understanding the potential returns on their investment in the pool.

Now that the global picture is painted for LP interaction, let's dive into the main technical element: Let's see how much must the LP provide of the second asset in the active reserve to keep the price unchanged, because actually it isn't exactly straightforward. Regarding other reserves not containing the current price, we will later see that providing outside this active reserve is very different.

First of all, recall the virtual reserve equation:

$$(x_{\text{real}} + \frac{L}{\sqrt{P_b}}) \cdot (y_{\text{real}} + L \cdot \sqrt{P_a}) = L^2$$

Let's start our providing calculations by noticing that we can simplify the process by considering that outside the price range, the liquidity (or the real amount of token in the reserve) is entirely provided by a single asset, either X or Y asset, depending on the current price's position relative to the price range. This leads us to three scenarios:

1. Assuming $P \leq P_a$, the reserve is fully in X asset, so $y_{\text{real}} = 0$. We obtain the following equation:

$$\begin{aligned} (x_{\text{real}} + \frac{L}{\sqrt{P_b}}) \sqrt{P_a} &= L^2 \\ x_{\text{real}} \sqrt{P_a} + L \sqrt{P_a} \sqrt{P_b} &= L^2 \\ x_{\text{real}} &= \frac{L \sqrt{P_a} - L \sqrt{P_b}}{\sqrt{P_a} \sqrt{P_b}} \\ x_{\text{real}} &= \frac{L(\sqrt{P_b} - \sqrt{P_a})}{\sqrt{P_a} \sqrt{P_b}} \\ L &= \frac{x_{\text{real}} \sqrt{P_a} \sqrt{P_b}}{\sqrt{P_b} - \sqrt{P_a}} \end{aligned}$$

2. Assuming $P \geq P_b$, the position is fully in Y asset, so $x_{\text{real}} = 0$. The liquidity L is given by:

$$\begin{aligned} \frac{L}{\sqrt{P_b}} (y_{\text{real}} + L \sqrt{P_a}) &= L^2 \\ y_{\text{real}} \sqrt{P_b} + L \sqrt{P_a} \sqrt{P_b} &= L^2 \\ y_{\text{real}} &= L(\sqrt{P_b} - \sqrt{P_a}) \\ L &= \frac{y_{\text{real}}}{\sqrt{P_b} - \sqrt{P_a}} \end{aligned}$$

3. Now, the point that interests us: if the current price P is within the range ($P_a < P < P_b$). We just saw how to calculate the liquidity of a single-asset range. The trick is, when P is in the range (p_a, p_b), we can think of (P, p_b) as the sub-range where X provides liquidity and (p_a, P) as the sub-range where Y provides liquidity. Therefore:

$$\begin{aligned} x_{\text{real}} &= \frac{L(\sqrt{P_b} - \sqrt{P})}{\sqrt{P} \sqrt{P_b}} \\ y_{\text{real}} &= L(\sqrt{P} - \sqrt{P_a}) \end{aligned}$$

We have derived formulas to calculate the real quantities present in the reserve. However, these calculations reflect the aggregated real quantities of all LPs, as they are based on the virtual equation of the entire reserve. This equation correlates the total amounts with the overall liquidity. The key insight here is that each LP effectively operates with their own virtual equation within their specific price ranges, characterized by their unique real quantities x_{real} , y_{real} , and L , rather than the aggregate totals. Consequently, we can apply the previously derived formulas for y_{real} and x_{real} individually to each LP. This application assumes the utilization of each LP's specific liquidity level L , rather than the aggregate total liquidity L of the entire reserve. This distinction is crucial for accurately calculating the real quantities for each LP based on their unique contribution to the reserve.

Actually, each LP's contribution is independent of others, being influenced solely by the active reserve boundaries P_a and P_b , and the current price P . Indeed, we can prove it

mathematically by noticing that the derived formulas for the total x_{real} and y_{real} reveal that:

$$\frac{x_{\text{real}}\sqrt{P}}{\sqrt{P_b} - \sqrt{P}} = \frac{y_{\text{real}}}{\sqrt{P} - \sqrt{P_a}}$$

So $y_{\text{real}} = D \times x_{\text{real}}$, where D is a function of P , P_a , and P_b . This formula establish the necessary proportions to maintain the current price within the reserve's boundaries. Should a LP contribute with a ratio that deviates from this prescribed proportion, the validity of the linear equation becomes compromised. The only way to restore its validity under such circumstances is by altering the sole other variable that can be modified, which is P . As a result, when an LP contributes, they must adhere to this proportion D to keep the current price stable. It's important to note that the correct proportion is not simply $P = \frac{y}{x}$ as in UniV2, since this price formula includes offsets. This means that P does not directly represent the 'real' token ratio in the reserve.

Having discussed the theory, we now proceed with an example. Consider a scenario where an LP has $x_{\text{real}} = 2$ units of SUI and intends to establish a liquidity position in a SUI/USDC pool. Suppose the current price of SUI is $P = 2000$ USDC, and the LP aims for a price range from $P_a = 1500$ USDC to $P_b = 2500$ USDC. The question is: what amount of USDC (y_{real}) is needed for this position? We have x_{real} , we can use it to compute the liquidity L provide by this LP and then insert L into the formula we derived for y_{real} , or immediately use the combination:

$$\frac{x_{\text{real}}\sqrt{P}}{\sqrt{P_b} - \sqrt{P}} = \frac{y_{\text{real}}}{\sqrt{P} - \sqrt{P_a}}$$

Which yields $y_{\text{real}} = 5076.10$, indicating the USDC amount the LP should provide.

That concludes the process of providing liquidity into the active reserve. For positions entirely outside the active reserve, an LP has the flexibility to contribute as much as desired of one token only (the appropriate one). When dealing with ranges that include the active reserve, the allocation of quantities to each segment of the reserve will be determined by straightforward formulas, taking into account the selected shape of the position. Lastly, the process of removing liquidity is simplified due to the fungible token setup. A user, who may have established multiple liquidity positions, has the option to remove these positions individually, one at a time, without affecting the others.

5. TRADER INTERACTION

The trading process begins with the trader selecting two tokens. They are then provided with the current price and a price chart for the chosen pair. This chart offers various time windows, enabling the trader to gain insights into the historical prices and fluctuations of the tokens. As the trader specifies the amount they wish to swap, the interface displays the expected price impact and a slippage tolerance setting. This slippage tolerance is preset to a default value but can be adjusted by the user. Additionally, a price impact warning is integrated to alert the trader if the potential trade exceeds a certain threshold, which could result in a disadvantageous transaction.

Upon selecting the token pair, the trading route will be visually represented, indicating which pool or pools will facilitate the swap. This route may involve multiple pools, especially if it leads to reduced slippage owing to superior liquidity, thereby offering more favorable conditions for the trader. The user also has the option to manually select a specific pool, should they prefer its unique features or hooks that might be tailored for traders. This flexibility allows traders to optimize their trades based on their preferences or the distinctive attributes of different pools.

Fees that will go to the LPs are also displayed to the trader, indeed those fees mentioned before are a necessary feature, they impact both swappers and liquidity providers. As we already mentioned, when creating the pool the creator sets up the fees percentage per reserve (implying the initialized tick spacing). Whenever a swap is performed by a trader on a certain pool, the fees times the quantity swapped are split among the liquidity providers of the active reserve, proportionally to the liquidity providers of this reserve.

Let's delve into the specifics of an 'in reserve' swap. This refers to a swap where, post-execution, the price remains within the same reserve as it was prior to the swap, it does not cross a price tick. Define γ as the transaction fee, which, for example, could be 0.003, and y_{in} as the quantity of token1 that is being exchanged. The increase in y_{real} in the reserve, accounting for the fee, is denoted by $\Delta y = y_{in} \cdot (1 - \gamma)$.

We begin by considering the total virtual reserve equation:

$$x \cdot y = L^2$$

Here, recall that $x = x_{real} + x_{offset}$ and $y = y_{real} + y_{offset}$, where the real quantities represent the total aggregate contributions of all LPs in the reserve, and L the total liquidity of it. Since a trade does not alter the liquidity of a reserve L , it also does not change the offsets. Therefore, only the real quantities are affected. These changes in real quantities must still satisfy the virtual reserve equation:

$$(x + \Delta x)(y + \Delta y) = L^2$$

From this, we derive that

$$\Delta x = \frac{L^2}{y + \Delta y} - x$$

is the amount of token x exiting the reserve, and consequently, being transferred to the trader.

However, in concentrated liquidity models, contracts typically monitor liquidity (L) and the square root of the price (\sqrt{P}), rather than x and y . As a result, in practical scenarios, we need to deduce the amount being transferred to the trader using these tracked quantities.

Remarkably, this can be accomplished through the application of formulas we derived earlier. Recall for a reserve:

$$x_{real} = \frac{L(\sqrt{P_b} - \sqrt{P})}{\sqrt{P}\sqrt{P_b}}$$

$$y_{real} = L(\sqrt{P} - \sqrt{P_a})$$

Now, let's suppose P is the price before the trade and P_{new} is the price after the trade. By considering the interval $[P, P_{new}]$ as a new reserve, we can derive the following relationship (Note: if the trader opts to provide some amount of token x instead of token y , then the roles of P and P_{new} would be reversed in the calculation.):

$$x_{real} = \frac{L(\sqrt{P_{new}} - \sqrt{P})}{\sqrt{P}\sqrt{P_{new}}}$$

$$y_{real} = L(\sqrt{P_{new}} - \sqrt{P})$$

And here, the y_{real} of this abstract reserve represents the Δy introduced by the trader. Consequently, the x_{real} will be the amount returned to the trader, denoted as Δx , for his trade involving $\Delta\sqrt{P}$. But how do we determine P_{new} ? By acknowledging that L remains constant, we understand the liquidity L of this "new subpool" and can utilize the Δy provided by the trader. Let's denote $\Delta P = \sqrt{P_{new}} - \sqrt{P}$. We can compute it using:

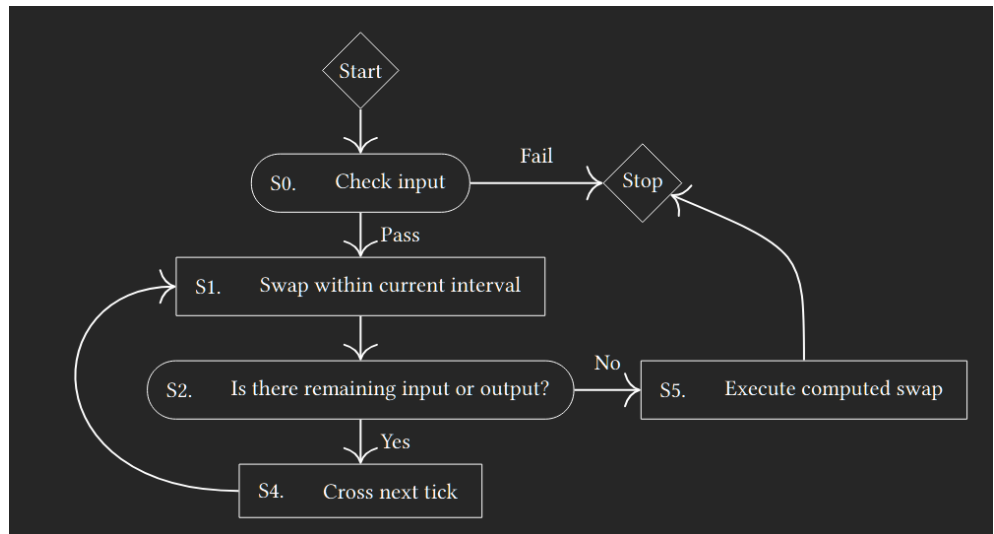
$$\Delta\sqrt{P} = \frac{\Delta y}{L}$$

And obtain Δx with the second equation, denoting by $\Delta\frac{1}{\sqrt{P}} = \frac{(\sqrt{P_{new}} - \sqrt{P})}{\sqrt{P}\sqrt{P_{new}}}$:

$$\Delta\frac{1}{\sqrt{P}} = \frac{\Delta x}{L}$$

$$\Delta x = \Delta\frac{1}{\sqrt{P}} \cdot L$$

These formulas remain effective for any swap that does not propel P beyond the price of the nearest initialized tick. If the calculated $\Delta\sqrt{P}$ risks moving \sqrt{P} past this tick, the contract should only proceed up to that tick, utilizing only a portion of the swap. It then crosses the tick before proceeding with the remainder of the swap.



An illustration of the swap algorithm



CUSTOMIZATION: HOOKS



One of the most important features that our DEX intends to bring to the SUI ecosystem is the concept of hooks, let us delve through it:

Liquidity pools undergo a defined lifecycle, involving creation with an initial fee tier, adjustments in liquidity, and token swaps by users. In classical DEXs, these events follow a rigid sequence.

With WABKA, **the goal is to allow more flexibility in liquidity management.** This is achieved by enabling pool deployers to insert specific code at crucial moments in the pool's lifecycle, such as before or after swaps, or when liquidity provider (LP) positions are modified. In general the goal is to offer customization to any person interacting with the contract.

This is exactly what uniswap means with 'hooks'. They act as customizable plugins, altering the way pools, swaps, fees, and LP positions interact. Through these hooks, developers can build upon Uniswap Protocol's foundational liquidity and security features, crafting bespoke Automated Market Maker pools that are integrated with v4's smart contracts.

hooks can range from defining a new constant function for updating of the price initiated by a LP to a customizable fees distribution ,the concept of hooks is quite crucial for the futur of DEX's as it allows freedom to the user but also most importantly it allows the user to keep up with the new methods and algorithms and implement them on WABKA protocol. The idea is to set a number of predefined hooks, our most important ones are : **The TWAMM, the dynamic fees hook** and the **stableswap** hook and last but not least the **geometric mean** oracle hook. In the next section, we present each of these innovative hooks that we would like to implement.

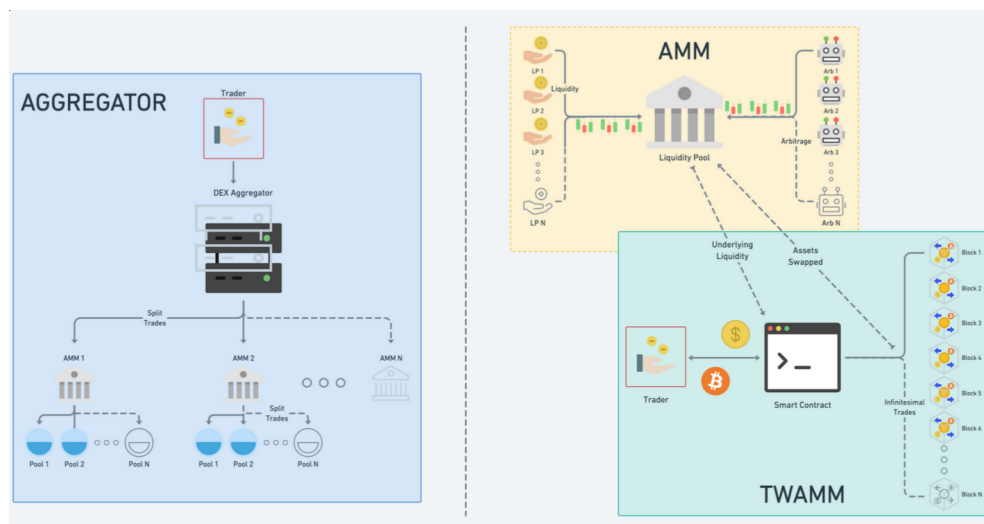
6. THE TWAMM

6.1. AN OVERVIEW AND KEY POINTS

Many believe that the efficiency of orders on DEXs depends entirely on Automated Market Makers (AMMs), which expedite order matching and execution without the need for intermediaries, as explained before, using pricing algorithms and liquidity pools to set exchange rates and execute trades through smart contracts. This method may reduce gas fees and slippage while increasing liquidity, yet its effectiveness wanes as the order size grows, with large trades unbalancing the liquidity pools and leading to price slippage. Consequently, DEXs confront a trade-off, having to choose between minimizing slippage, offering competitive pricing, or maximizing liquidity when handling hefty orders, and some DEXs manage to maintain low slippage at the expense of capping the liquidity available to traders.

We thus adapt an innovative solution **unprecedented on SUI** to better optimize large-order execution so traders can avoid unnecessary costs resulting from slippage or wasting time and effort manually separating and batching large orders into smaller ones: **The Time Weighted Average Market Making algorithm.**

The idea is simple and clear: **divide big trades over many small trades over time**, it can be seen thus as a competitor to data aggregators that splits trades over several AMM's.



Data aggregator vs TWAMM embedded AMM, the TWAMM competes for a better trade experience for traders that do not care much about executing their trades instantly.

the TWAMM we propose permits :

- **low slippage and price impact for the trader** especially when the pool does not contain a big amount of liquidity
- **Higher gains for passive liquidity provider**

In the next sections we give a comprehensive explanation of how the TWAMM we want to build works and how it can hugely benefit the SUI ecosystem.

6.2. LARGE ORDERS ON CURRENT AMM

When dealing with Automated Market Makers (AMMs), it's inherently costly to fill a substantial order in a singular transaction due to the **price impact** or the **slippage** that the trader encounters. It is advisable for traders intending to place large orders through an AMM to avoid executing them all at once. Instead, breaking down the order into smaller segments is more prudent. The greater the volume of the order, the more beneficial it is to divide it across **time**.

Let us illustrate this through an excellent example provided in the original paper of paradigm that introduced the concept :

Consider an ETH/USDC CPAMM with 2,000 USDC and 1 ETH in the reserves, so that $x = 2,000$, $y = 1$, and $x * y = k = 2,000$. The instantaneous price of this AMM is $2,000 / 1 = 2,000$ USDC per ETH. If a trader comes and buys 2,000 USDC worth of ETH, that means they are depositing 2,000 USDC into the X reserves, so that we will have $x = 2,000 + 2,000 = 4,000$. Then, since $k = 2000$, we must have $y = k/x = 2000/4000 = 0.5$ after the trade. Since y was originally 1, $1 - 0.5 = 0.5$ ETH must have gone to the trader. Since the trader bought 0.5 ETH with their 2000 USDC, they paid an average price of 4,000 USDC per ETH.

This high price of transaction translates how UNISWAP protocols (or any AMM in general) attempt to counter **adverse selection**. However, this limitates the liberty of the traders when placing an order.

A first attempt to combat high price impact would be to adopt an on-chain equivalent of algorithmic trading strategies, the most common and intuitive one would be the **TWAP : Time weighted average price** that ensures executing orders at the average price of the asset over a period of time, in the next section we wil explain intuitively how the TWAP works.

6.3. ANALOGY TO TRADITIONNAL FINANCE

An important approach to the TWAMM would be to delve into it's origin, a widely used algorithmic trading strategy in traditionnal finance: **The Time-Weighted Average Price (TWAP)**.

The Time-Weighted Average Price (TWAP) is designed to minimize the market impact of large orders by distributing the trade evenly across a specified time period. The TWAP strategy is particularly useful when trading large quantities of stocks, as it helps to avoid causing significant price fluctuations that can happen if a large order is executed all at once. This approach is beneficial for market participants who want for example to trade large quantities of a security without drawing attention to their activity, which could result in unfavorable price movements due to the size of their order.

Let's go through an illustrative example to understand the TWAP practically:

Imagine an institutional investor needs to sell 100,000 shares of a stock. To avoid impacting the market price significantly, the investor decides to use the TWAP strategy over a 5-hour trading window. The trading day is 8 hours long, so they might choose to execute their order in 10 equal parts of 10,000 shares each, spread evenly over the 5 hours.

Here's a hypothetical schedule for selling the shares:

- 9:00 AM: Sell 10,000 shares at \$50.00
- 10:00 AM: Sell 10,000 shares at \$50.25
- 11:00 AM: Sell 10,000 shares at \$49.75
- 12:00 PM: Sell 10,000 shares at \$50.50
- 1:00 PM: Sell 10,000 shares at \$49.50
- 2:00 PM: Sell 10,000 shares at \$50.75
- 3:00 PM: Sell 10,000 shares at \$50.25
- 4:00 PM: Sell 10,000 shares at \$49.95

To calculate the TWAP for these trades, the investor would take the sum of the products of the number of shares sold at each price point and divide it by the total number of shares sold:

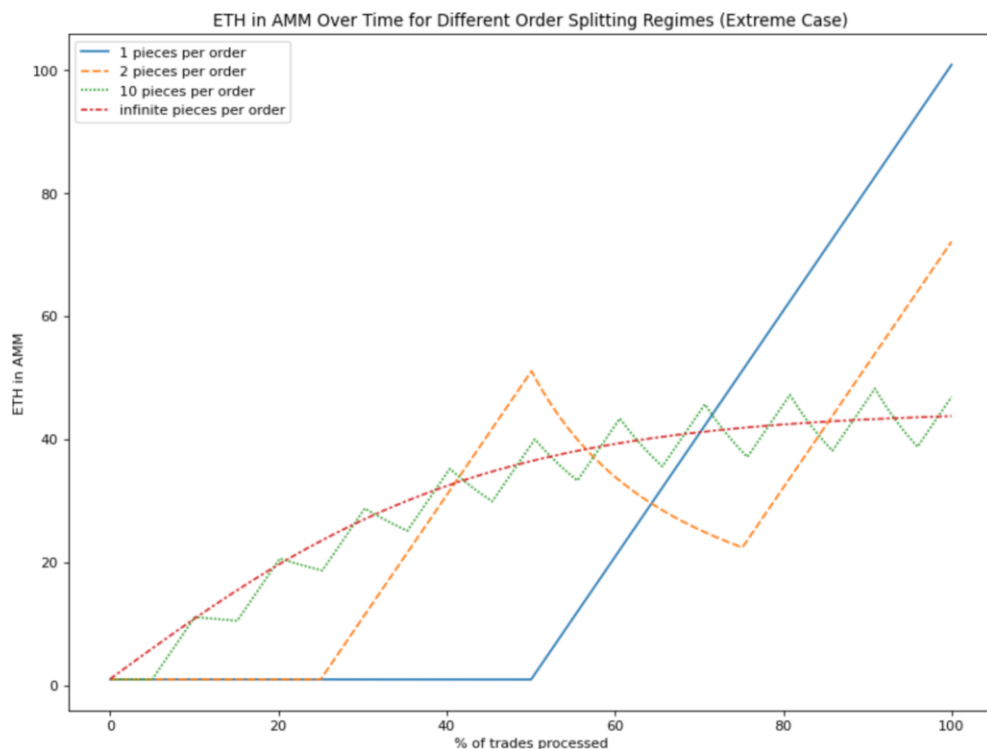
$$\text{TWAP} = (10,000 * 50.00 + 10,000 * 50.25 + 10,000 * 49.75 + 10,000 * 50.50 + 10,000 * 49.50 + 10,000 * 50.75 + 10,000 * 50.25 + 10,000 * 49.95) / 100,000$$

By executing the order in smaller, evenly spaced increments, the investor reduces the risk of causing a significant change in the stock's price, and approximates trades at a price equal to the TWAP.

6.4. THE ALGORITHM

The main algorithm relies on letting the user (trader) chooses **the amount to trade as well as the number of blocks for the transaction**. The TWAMM pool then initiates the TWAMM process, however, for gas optimization (although for now it is not a big matter, since sui price gas are low and the implementation of sponsor transaction) our goal for our v1 is to implement a similar model to UNISWAP: **virtual trades**, that records the trades off chain and rely on arbitrageurs to initiate an action in the AMM so that the trades are executed, this permits the user to not pay a big amount of gas fees.

The original paper of paradigm explains the need to adopt the infinite virtual trades strategy as the following: As the size of the virtual trades, price movement on the AMM becomes less and less jagged and in the case of "infinitely" many virtual trades the price movement is perfectly smooth and that in the limit, with infinitely many infinitely small trades, price movement is perfectly smooth as virtual trades are executed.

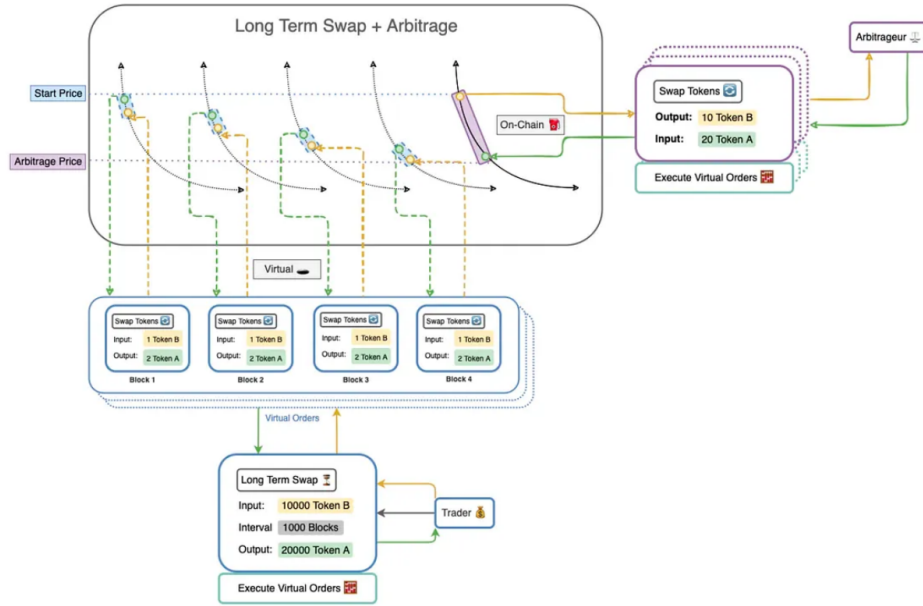


As we decide to split more and more, we diminish the abrupt change of price

The illustration below explains pretty well what happens: a trader gives the order as well as the number of blocks, the virtual trades are thus recorded as well as the changes of the price, when the price attains an arbitrage tolerance, an arbitrageur interacts with the contracts and at this moment we move on chain, hence **initiating the execution first of the virtual trades and then the trade of the arbitrageur**

6.5. LOW SLIPPAGE AND THE IMPACT OF ARBITRAGE

Traditional decentralized exchanges (DEXs) necessitate substantial liquidity, or a high concentration of it, to minimize significant price changes (slippage) during large trades.



A clear illustration of what happens from the moment the transaction in the TWAMM pool is recorded to the moment when the first external person interacts with the AMM.

Determining the amount of liquidity at the end of the virtual order can be computed through mathematical computations that result in the following:

The original paradigm TWAMM computations give the following result:

$$Y_{\text{endParadigm}} = \sqrt{k \cdot \frac{Y_{\text{input}}}{X_{\text{input}}}} \cdot e^{\left(2\sqrt{\frac{Y_{\text{input}} \cdot X_{\text{input}}}{k}} + c\right)} - e^{\left(2\sqrt{\frac{Y_{\text{input}} \cdot X_{\text{input}}}{k}} - c\right)}$$

$$X_{\text{endParadigm}} = \frac{k}{Y_{\text{endParadigm}}}$$

$$c = \frac{(\sqrt{Y_{\text{start}} \cdot X_{\text{input}}} - \sqrt{X_{\text{start}} \cdot Y_{\text{input}}})}{(\sqrt{Y_{\text{start}} \cdot X_{\text{input}}} + \sqrt{X_{\text{start}} \cdot Y_{\text{input}}})}$$

FRAX finance came up with the first implementation of the TWAMM on a DEX, it also

implements the TWAMM on UNIV3 model, hence we took inspiration from their protocol and will use it along with some optimizations to implement it in the SUI environment.

FRAX uses the the following closed and optimised approximation for efficiency

:

$$Y_{\text{endFraxSwap}} = X_{\text{start}} \cdot \frac{Y_{\text{start}} + Y_{\text{input}}}{X_{\text{start}} + X_{\text{input}}}$$

$$X_{\text{endFraxSwap}} = \frac{k}{Y_{\text{endFraxSwap}}}$$

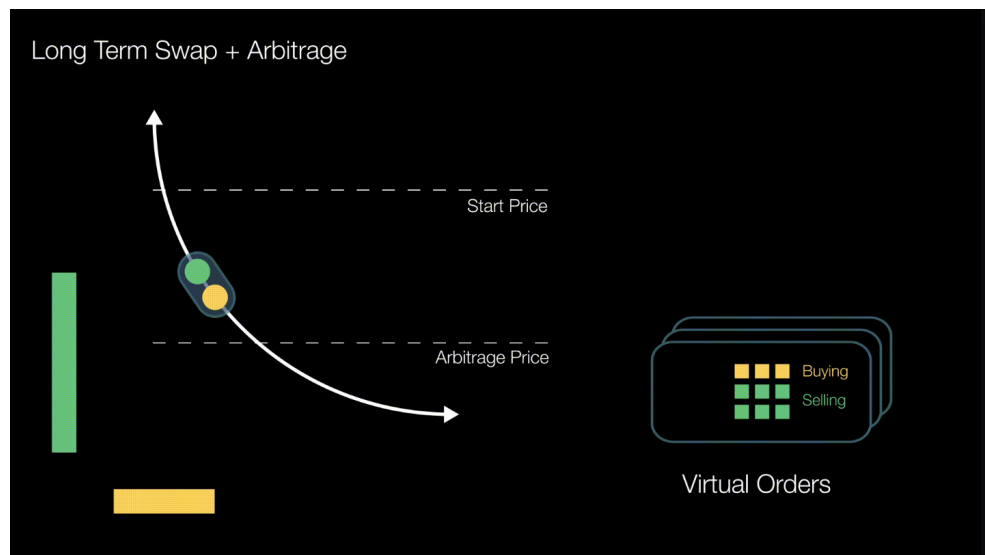
However, these platforms generally struggle with very large transactions, such as those in the hundreds of millions of dollars range. To mitigate this, even large orders might be broken down into smaller segments, but this still requires liquidity providers (LPs) to inject more liquidity to maintain desired slippage levels.

Hence, long term traders often rely on CEX or OTC in order to fulfill their order, which is understandable also because of other several reasons like gas prices, avoiding frontrunning or privacy, however, recent events around binance or FTX can discourage traders to go for CEX, hence the TWAMM is a perfect alternative for them especially because it solves most of the problems of DEXs.

Time-Weighted Average Market Makers (TWAMMs) segment trades into smaller pieces, significantly lowering the liquidity required to execute swaps while maintaining minimal price disruption. For example, dispersing a \$100 million trade over 10,000 blocks equates to individual trades of approximately \$10,000 each. **A TWAMM with an asset pool around \$2 million is capable of processing these trades with an estimated slippage of about 1%, with arbitrageurs stepping in to adjust the prices accordingly.**

Furthermore, TWAMMs serve as a **convergence point for liquidity** by incentivizing participants from across the market to introduce additional capital from a variety of sources, such as decentralized exchanges (DEXs), lending entities, over-the-counter (OTC) trading desks, and centralized exchanges (CEXs). This dynamic fosters arbitrage opportunities for market actors aiming to capitalize on the realignment of the pool's asset prices to current market rates. Nonetheless, it's important to note that even the most liquid OTC desks or DEXs have their own set of constraints regarding the volume of assets they can hold in their order books. They are not perfect.

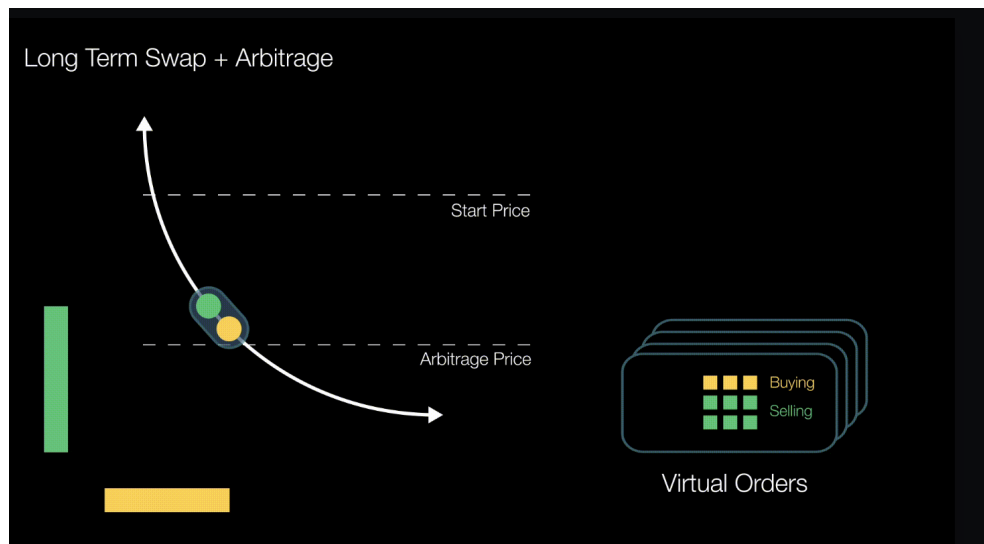
Arbitrage is thus the main component of the algorithm. The following images give an even more precise illustrative description of the arbitrage mechanism using the quantities of the assets traded.



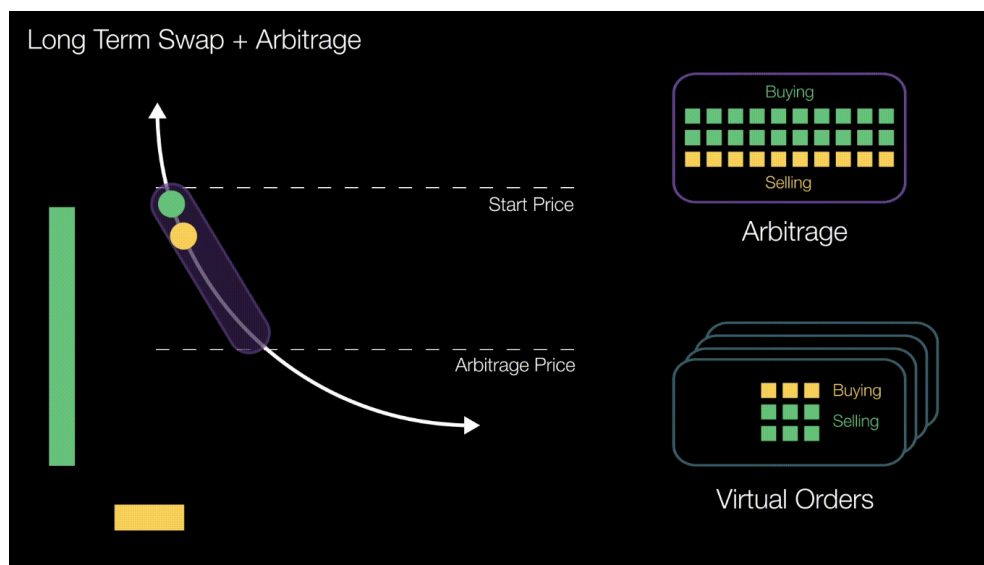
a midstate of a TWAMM order pushing the price towards the arbitrage price

6.6. TWAMM LP, ARBITRAGEURS AND VALIDATORS BALANCE.

To ensure the success of a TWAMM pool in general or a TWAMM transaction specifically, we need the contribution from three different independent actors: **The TWAMM LP's** that provide liquidity, **the validators** that encode and validates the transaction and of course **the arbitrageurs**. The problem that occurs is that LP can sometimes face



Reaching the arbitrage price after executing a certain number of blocks



Arbitrage regulates the price and hence setting up the price and quantities lower.

passivity in terms of fees, although the dynamic fees customization helps to tackle this problem but what can be quite interesting is the fact that we should allow the LP to gain also from the arbitrage opportunities that occurs in his pool, this would encourage more and more liquidity providers to deposit into TWAMM pools and hence to benefit more and in a dynamical way from their deposits. Finding a balanced distribution between the gain of those 3 actors is then a priority. Let us go more in details through this :

Our goal is that the LP also benefits from the gain captured by the arbitrageur, this would be like a win win strategy , here is why:

- **For the LP:** the fees would be superior than the ones earned in a fixed swapped fee.
- **For the arbitrageurs:** It may appear counter intuitive that it is beneficial for an arbitrageur to share the profits within a TWAMM, but this arbitrageur would waste time and effort looking at arbitrage opportunities elsewhere while in a TWAMM transaction,

the profits are lower but the insurance of gain is 100% as surely after a number of blocks the arbitrage price would be reached.

A practical example would be useful in our case:

let us consider that Anis really appreciated the TWAMM feature and decided to order another transaction within a TWAMM pool. Amro, is an arbitrageur that interacts with the order of Anis, so whenever the price reaches the arbitrage price after an execution of a number of blocks, Amro takes the arbitrage opportunity, suppose the pool swaps asset A and asset B and that the order of Anis is that he wants to swap 1000 A for B and that the price is recorded as 1A gives 1B (before the execution of any blocks) as soon as the price of 1A becomes 2B for example, Amro buys 1A for 1B from an external liquidity provider and sells in the TWAMM pool, he then recorded a gain of 1B, however, 1/3 of this gain would go to the TWAMM LP and 1/3 would go to the validators, Amro then recorded a gain of 1/3 B but he is sure to have even more arbitrage opportunities because of the nature of the transaction that Anis ordered.

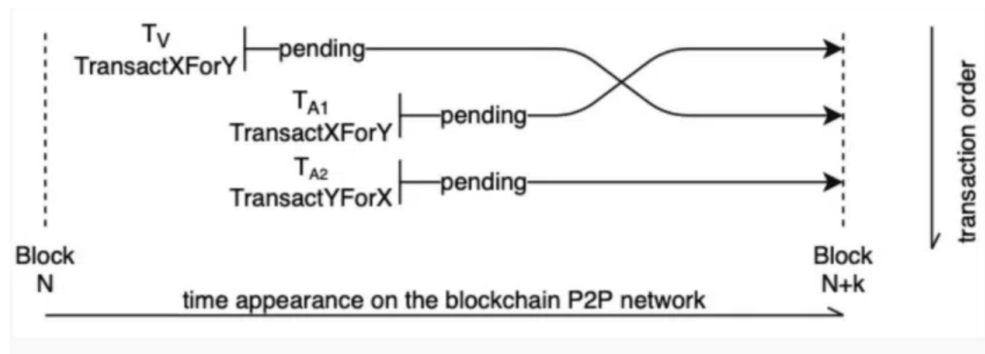
6.7. SANDWICH ATTACKS: A SOLUTION ATTEMPT

An important thing to consider while aiming to implement the TWAMM is vector attacks that can occur while trading.

The most common and dangerous attack is the Sandwich attacks, a bad actor can perform a sandwich attack in two ways:

1. **Liquidity Taker vs. Taker:** Attackers can target other traders by placing their own transactions (front-running and back-running) around a victim's pending transaction. If the attacker's transaction fees are higher, they increase the odds of their transaction being processed first, potentially manipulating the trade outcome.
2. **Liquidity Provider vs. Taker:** In this scenario, an attacker who is also a liquidity provider, he proceeds in 3 steps first withdraws liquidity to worsen the victim's slippage, then adds it back after the victim's transaction, and finally swaps assets to rebalance the liquidity pool. This sequence can disrupt the victim's expected transaction cost and outcome.

The first case is quite interesting for us, let us suppose Anis, a trader wants to trade 1000 ETH for dollars at once, before mining the transactions, it is visible in the mempool, hence any front runner has access to it. The latter will then understand that the price before Anis's swap and after it will change in his favor if he sold ETH before and bought it right after. Ordinarily, transactions within a block are ordered such that those offering a higher gas price are executed sooner, while those with lower bids are processed later. Front-runners can exploit this by observing transactions waiting in the mempool and then deliberately setting a higher gas price for their own transaction, ensuring it is executed before the transaction they are targeting. Which results in a benefit for the front runner but also a **price slippage for Anis**.



An illustration of a sandwich attack in the case of a taker vs a taker.

The question is **are sandwich attacks really worth it?**

Well, everything depends on the gas costs, if the transaction is big then the front runner is likely to earn a gain even though he pays a big amount of gas fees. However, by splitting a big transaction into many small ones, the price change can be small enough so that the gas fees paid to front run and back run are discouraging.

The TWAMM is a good solution for preventing sandwich attacks in case of high and not neglectable gas fees. Also, an important remark is that **short term traders wanting to benefit from a TWAMM transaction will likely be more tempted to take advantage of arbitrage opportunities when the price deviates sufficiently than to perform front and back running**, and now Anis will likely benefit from the intervention of short term traders than loose. By doing so, **the TWAMM is indeed a kind of protection against sandwich attacks.**

However, the information that Anis initiates a big trade is still visible although it is divided into blocks thus it is still accessible to front runners and thus front running is still "theoretically" possible, one way Anis could counter this would be to cancel his trade in the TWAMM pool after a number of transactions, but this is not optimal.

The way we thought about to tackling this problem is inspired from linch protocol, the latter introduced the concept of **flashbots transactions**, the idea is to connect directly to trustworthy validators and to not pass by the mempool, hence the trade pending is not publicly accessed.

Another way would be the Uniswap v4 method, which states that **any trade coming from a TWAMM pool will automatically be executed first in a block**, preventing any kind of front-running, however, a TWAMM trade itself can always be "sandwiched" by 2 other TWAMM trades.

We are still performing research to come up to a clear conclusion on how to solve this problem optimally on SUI ecosystem.



OTHER TYPES OF HOOK



7. LIMIT ORDERS

Having outlined the primary hook, TWAMM, we now turn our attention to a non-exhaustive list of additional hooks that Wabka will develop, independent of user-generated ones. It's important to note that these hooks are still under research, and what follows are only their foundational concepts. Let's start with limit orders.

Liquidity positions established on very narrow ranges function in a manner akin to limit orders. When the range is traversed, the composition of the position shifts from being entirely one asset to being completely the other asset, in addition to any accrued fees. However, there are notable distinctions between this type of range order and a conventional limit order:

- The range of a position is subject to a minimum width. As long as the price remains within this range, the limit order may only be partially executed.
- Once the position has been crossed, it necessitates withdrawal. Failure to do so means that if the price moves back through that range, the position will revert, essentially undoing the initial trade.

Consequently, Wabka DEX introduces a hook for limit orders, a feature historically less common in decentralized exchanges due to technical and blockchain-specific challenges. Recall that a trader can choose the pool (or even pools) he will operate in (called the "route") or it will work the other way around, if a trader wants to use a specific feature like limit order, then his route will be redirected to a the most optimal pool in term of fees and slippage allowing this feature. Limit orders allow traders to specify the exact price at which they wish to buy or sell an asset, unlike market orders which execute at the current market price. This addition to Wabka DEX not only enhances trading precision but also caters to a strategic approach to asset management, enabling users to set predefined entry and exit points in the market. The integration of limit orders is particularly significant considering the dynamic nature of DeFi markets, where price fluctuations can be rapid and pronounced. By offering limit orders, Wabka DEX addresses a crucial need for advanced trading functionalities, thereby empowering its users with greater control over their trading strategies and solidifying its role as a comprehensive and user-oriented DeFi platform.

We are also contemplating the inclusion of long and short positions in Wabka's offerings. It is envisaged that Wabka might feature two distinct user interfaces: one catering to straightforward swaps and another designed for more seasoned traders, where advanced functionalities like shorting will be accessible. However, details regarding this aspect are still under discussion and have not been finalized.

8. DYNAMIC FEES WITH VOLATILITY ACCUMULATOR

Wabka DEX introduces another hook set by default. This hook implements an approach to dynamic fee adjustment in a concentrated liquidity model to mitigate impermanent loss, credits to TraderJoe for the idea. In a pool enabling this hook, the dynamic fees are adjusted based on the volatility across price reserves

- **Dynamic Fee Structure:** The total swap fee (f_s) comprises the classical base fee (f_b) and a variable fee (f_v), influenced by the instantaneous price volatility within each reserve. The total fees are distributed proportionally among liquidity providers within each reserve range.
- **Base Fee:** The base fee is determined by a base factor (B) and influenced by the pool stepsize (s):

$$f_b = B \cdot s$$

- **Variable Fee and Volatility Accumulator:** The variable fee is contingent on market volatility, captured by a volatility accumulator (v_a). This accumulator is updated with each swap to reflect current market volatility based on the pool's movement:

$$v_a(k) = v_r + |i_r - (\text{activeId} + k)|$$

where v_r is the volatility reference, i_r is the index reference, and activeId denotes the ID of the active reserve before the swap. Here, k represents the difference in reserve IDs between the initial reserve, from which the swap originated, and the current reserve where the fee calculation is taking place.

The variable fee is calculated using:

$$f_v(k) = A \cdot (v_a(k) \cdot s)^2$$

Fees for swaps are determined in an iterative manner as the swap crosses each reserve, especially in cases of large swaps. Specifically, if a swap traverses n reserves, the total swap fee is computed for each reserve k , where $0 \leq k \leq n$.

- **Volatility Reference Adjustment:** The volatility reference (v_r) adjusts based on the time elapsed (t) since the last transaction, within bounds defined by a filter period (t_f) and a decay period (t_d). This ensures that high-frequency trades amplify volatility, while low-frequency trades diminish it.
- **Fee Calculation:** The total fee γ during a swap is the sum of the base and variable fees applied to the swap amount within that reserve:

$$\gamma = \cdot f_b + f_v$$

By implementing dynamic fees based on market volatility and reserve movements, Wabka DEX aims to provide a fair and effective fee structure. This approach not only incentivizes liquidity provision during high volatility periods but also offers protection against impermanent loss, aligning with the platform's goal of enhancing the DeFi experience for its users.

9. POWER PERPETUALS TO HEDGE IL

Power Perpetuals within the Wabka ecosystem emerge as a sophisticated financial instrument tailored to the unique requirements of Liquidity Providers (LPs). These perpetual contracts provide a means for LPs to hedge against adverse market movements, thereby offering a safeguard for their staked assets.

Indeed, addressing the issue of impermanent loss to protect our liquidity providers is a multifaceted challenge. Having already introduced dynamic fees, we now turn our focus to hedging the LP position.

One might naturally consider options for hedging purposes. However, in the crypto domain, options are not widely popular due to fragmented liquidity. This fragmentation is caused by multiple factors such as varying expiry dates, diverse strike prices, and the inability of pool liquidity to adequately adapt to these variables.

On the other hand, power perpetuals provide global options-like exposure without the need for either strikes or expiries, giving them the potential to consolidate much of options market liquidity into a single instrument.

9.1. POWER PERPETUALS OVERVIEW

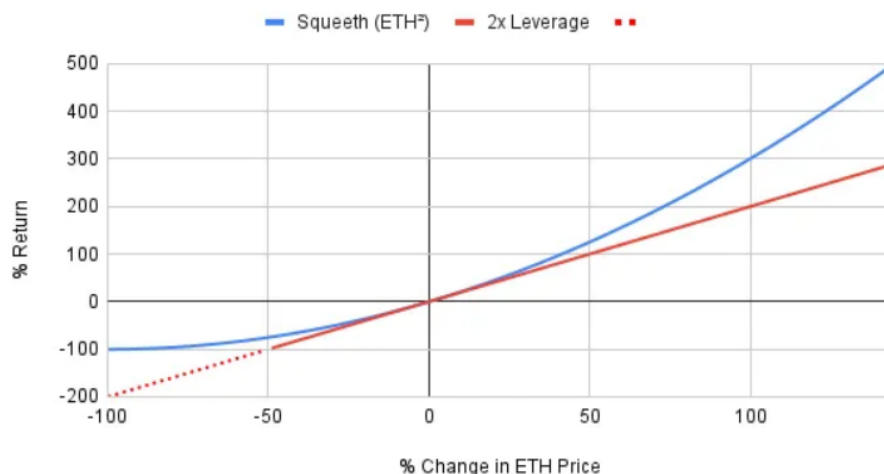
Power Perpetuals represent an innovative advancement beyond the concept of perpetual futures. To encapsulate their essence:

- Traditional perpetual contracts mirror the price of an asset. Although leverage in these contracts amplifies asset exposure, the nature of value exposure remains linear.
- Power Perpetuals, contrastingly, track a higher-order function of an asset's price, such as its square, cube, or another power function.

Let's dive into the technical details of Power Perpetuals using the first one ever created squared Eth: Squeeth, by Oryn. This particular case, focusing on ETH squared (ETH^2), generally extends to other tokens and power functions with minor variances.

Squeeth, developed by Oryn's Research Team, offers continuous exposure to ETH^2 , revolutionizing the derivatives market. A crucial attribute of Power Perpetuals like Squeeth is their convexity, meaning enhanced returns compared to a 2x leveraged position when ETH's price increases, and mitigated losses when ETH's price decreases. Of course, this asymmetric upside doesn't come for free. Those who are long a power perpetual must regularly pay a premium yield to those who are short it.

Wabka will specifically focus on defining Power Perpetuals with the power $p = 2$, primarily for technical reasons. The consideration of other powers has been deemed not beneficial enough compared to the protocol risk they add. Indeed, Power perpetuals rely on external oracles for accurate price tracking of the underlying, which introduces additional complexity and potential vulnerabilities.



Return comparison between ETH^2 and 2x leverage on ETH





Important Power Perpetuals characteristics to understand

- *Index Price:* The index price for Squeeth is defined as ETH^2 .
- *Mark Price:* The Mark Price represents the current trading price of Squeeth in the market.
- *Funding Rate:* This is a periodic payment exchanged between long and short traders of Squeeth. It is determined based on the difference between the Index Price (ETH^2) and the Mark Price. The calculation is typically Mark – Index. The funding rate is influenced by the contract's demand and supply, which in turn affects the gap between the index price (ETH^2) and the mark price (the current trading price of Squeeth). Notably, the funding rate for Squeeth is generally expected to be positive, reflecting the convexity premium of holding a long Squeeth position.
- *In-kind Funding:* In Squeeth, funding is not paid out directly but is instead continuous, paid at a regular interval, say, daily.
- *oSQTH:* oSQTH represents the ERC20 token for Squeeth. The price of your position in oSQTH represents your exposure to ETH^2 , adjusted for the impact of the funding rate over the duration of your position. This adjustment includes both the changes in the underlying ETH^2 value and the net effect of daily funding payments or receipts.
- *Normalization Factor:* This is the variable that adjusts the value of your position in oSQTH to account for the effects of daily in-kind funding. For buyers of Squeeth, the normalization factor leads to a gradual decrease in the oSQTH value relative to the Index Price (ETH^2), due to the in-kind funding mechanism. Conversely, for sellers of Squeeth (those who have minted Squeeth), the normalization factor decreases the amount of debt on a short Squeeth position, reflecting the in-kind funding received.

Long Squeeth

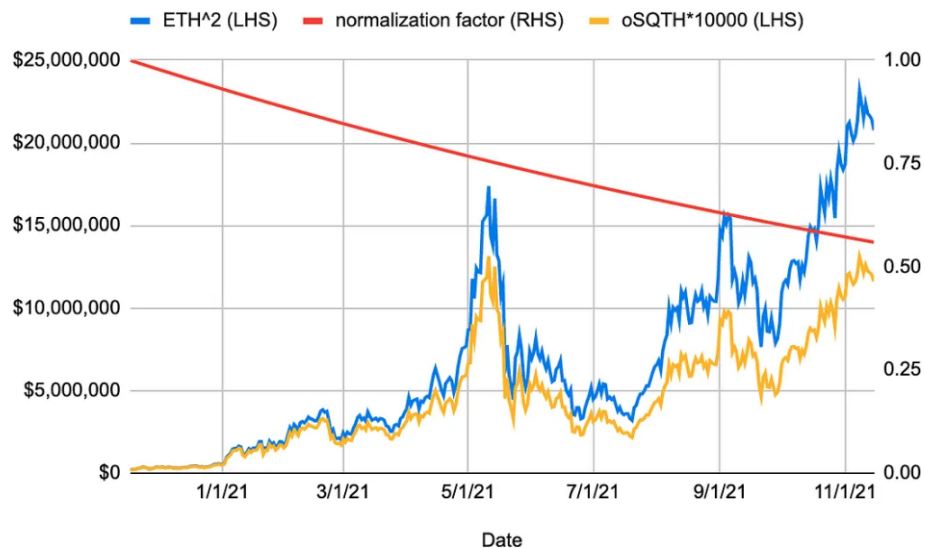
Long Squeeth positions offer traders leveraged exposure with unlimited ETH^2 upside, protected downside, and are free from the risk of liquidations. This type of position provides pure convexity with a payoff pattern similar to ETH^2 , along with applicable funding rates for holding the position long-term.

When you go long on Squeeth, you essentially maintain a position akin to an at-the-money call option but with constant gamma exposure. This setup operates similarly to a perpetual swap, with the critical difference being the focus on ETH^2 as opposed to ETH itself.

	Payoff	Liquidations	Payment	Perpetual
Long Squeeth		No	Funding	Yes
2x Leverage		Yes	Interest	Yes
Perpetual Swap		Yes	Funding	Yes
Call Option		No	Premium	No

Derivatives comparison table.

The most favorable market condition for holding Long Squeeth is when there is a strong belief in ETH's upward price trajectory in the short to medium term. However, it's important to be mindful of the potential impact of prolonged holding periods. Specifically, if you hold a Long Squeeth position for an extended duration (e.g., more than one year) and ETH's price either remains stagnant or decreases, the ETH^2 exposure of the Long Squeeth position may diminish. This reduction is primarily due to the in-kind funding payments made to Short Squeeth sellers over time.



The impact of funding over time (ETH^2 in USDC)

The graph above shows the impact of a hypothetical constant funding rate for a long Squeeth position from November 2020 to November 2021. Leverage (ETH^2 payoff) comes from being long convexity, and the reason long positions can't be liquidated is due to in-kind funding.

Funding rates details

As we saw, Squeeth is described as tracking, rather than equating to, ETH^2 due to the funding rate dynamics involved. This mechanism is analogous to the funding payments in perpetual futures, but with a key difference: there is no direct cash transaction. Instead, the value exchange between long and short positions occurs through the relative price movements of a user's Squeeth position (denoted as oSQTH) and ETH^2 . Hence the term: In-kind funding. It does not involve direct payments. Indeed, as we can see on the illustrations above, the exchange of value is represented by changes in the relative price of a user's Squeeth position (oSQTH) compared to ETH^2 . Thus:

- For buyers of Squeeth, the value of oSQTH gradually declines in relation to the Index Price (ETH^2).
- For sellers (short positions), the amount of debt in oSQTH adjusts downward over time.

In essence, if you purchase Squeeth, in-kind funding effectively reduces your position value. Conversely, if you sell Squeeth, the debt you owe diminishes due to accumulated funding payments. This results in an impact akin to cash funding, but without the need for additional mechanisms within the token to handle funding transactions.

More technically, in-kind funding, is governed by a key variable called the normalization factor, presented in the graph above. This global variable is the one systematically decreasing the relative value of a long Squeeth position and lowering the debt owed by a short position. The adjustment of debt value for a short Squeeth position is illustrated by the following equation:

$$\text{value of debt in ETH} = \text{original debt amount} \times \text{normalization factor} \times \text{ETH price}$$

Given that funding is typically expected to flow from long to short positions, this normalization factor is anticipated to decrease over time. This reduction lowers the relative price of oSQTH and makes the debt cheaper to repay for short sellers.

Short Squeeth

Short Squeeth is a short ETH^2 position, collateralized with ETH on Opyn, due to ETH well established nature in Defi. Recall the short traders also earn the funding rate for taking on this position, paid by long Squeeth holders. This is explained by the fact that, if you short Squeeth, you have constant negative gamma exposure, instead of constant positive gamma exposure that we saw for longs. Another difference with longs, is that short Squeeth positions have the possibility of being partially or completely liquidated.

At the recommended collateralization ratio of 200% (by Opyn), the ideal market condition to short Squeeth is when there is conviction the market is overestimating volatility and the price of ETH will trade sideways.

Because short Squeeth positions are collateralized with ETH, decreases in the price of ETH do not result in an ETH^2 payoff. Recall that Squeeth is solely an example, and that Wabka will reserve the right to chose which collateral is needed for any of the implemented Power Perpetuals. So in case where the collateral is not the underlying, the payoff for short traders might be higher (also recall the total payoff includes the fees the short trader earns). Finally, the maximum amount you can lose with a short Squeeth position is the amount of collateral used to mint and sell Squeeth, minus the premium received from selling.

How does a trader open a short Squeeth position ? He simply puts down ETH collateral to mint and sell Squeeth.

Wabka's directives

We have provided a detailed overview of power perpetual trading, specifically focusing on Squeeth as implemented by Opy. The question now arises: How will Wabka approach the implementation of such financial instruments? Currently, Wabka is likely to adopt a method very similar to Opy's strategy for handling power perpetuals, hence the presentation of their product. Indeed, the reason for using Squeeth as our primary example is its relevance to our intended approach. Our approach may also draw inspiration from Deri protocol, which has explored an alternative method for implementing these complex financial instruments. While Opy's strategy will probably be our primary model, we anticipate introducing some slight modifications to tailor the concept more closely to Wabka's specific needs and objectives.

9.2. HEDGING LP WITH POWER PERPETUALS IN WABKA ON SUI

Wabka Dex particularly aims at LP engagement. LPs play a pivotal role by providing liquidity to various pools, earning fees and incentives in return. However, the inherent volatility of cryptocurrency markets exposes LPs to potential risks, necessitating effective risk management strategies.

Power perpetuals are a sophisticated tool for liquidity providers to hedge against impermanent loss. Power perpetuals, with their unique payoff structure, are ideal for managing nonlinear risks such as impermanent loss, a prevalent issue for LPs in automated market makers. Moreover, power perpetuals (of power 2) provide constant gamma exposure and have no liquidations on the long side, offering advantages over perpetual swaps. Hence why Wabka decided to implement a hook introducing this strategic approach to managing risk and optimizing returns. Pools enabling this hook will provide LPs with the possibility to hedge their position and thus enhance their overall portfolio resilience.

Technical dive

Let us derive the hedging strategy with power perpetuals from a Greeks perspective, focusing on Delta (Δ) and Gamma (Γ). These are defined as follows, under the Black-Scholes assumptions:

1. Delta (Δ): This measures the rate of change in the power's price (M for Mark price) with respect to changes in the underlying asset's price (S). The Delta can actually be shown to be:

$$\Delta = \frac{\partial M}{\partial S} = \frac{p \cdot S^{p-1}}{1 - h \cdot T}$$

Here, M denotes the price of the power (the Mark price presented above), S is the price of the underlying asset, T represents the funding period (e.g., 1 week or 1 day) and $h = r + \frac{\sigma^2}{2}$.

2. Gamma (Γ): This is the rate of change in Delta with respect to changes in S . Gamma

for power perpetuals is expressed as:

$$\Gamma = \frac{\partial^2 M}{\partial S^2} = \frac{p \cdot (p-1) \cdot S^{p-2}}{1 - h \cdot T}$$

Notably, when $p = 2$, Gamma exhibits an important characteristic: it becomes independent of the underlying asset's price. This feature is crucial in hedging against impermanent loss in Constant-Product Market Making. The independence of Gamma from the underlying price is mathematically represented as:

$$\Gamma = \frac{2}{1 - h \cdot T} = \frac{2}{1 - \left(r + \frac{\sigma^2}{2}\right) \cdot T}$$

Here, r is the risk-free interest rate, and σ is the volatility.

This independence of Gamma in the case of square powers (i.e., $p = 2$) is the cornerstone for their use in mitigating impermanent loss, highlighting their significance in financial strategies involving power perpetuals.

In the DeFi world, it is usually safe to assume that the risk-free interest rate (r) is zero, $r = 0$. Given that volatility (σ) is relatively stable for most of the time, Gamma (Γ) remains constant for prolonged periods. This near-constant state of Gamma introduces several practical use-cases.

Now let's consider the value of a Liquidity Provider position when the current price P is in the range he provided in:

$$V = x_{real} \cdot P + y_{real}$$

Here, x_{real} and y_{real} are his real provided quantities, and P is the current price of in the pool.

Now recall that each LP has his own virtual reserve equation governing his quantities, and so utilizing our previously derived formulas, the value of the LP position can be further refined as:

$$V = L \left(2\sqrt{P} - \frac{P}{\sqrt{P_b}} - \sqrt{P_a} \right)$$

In this equation, L denotes the liquidity provided by the LP (his liquidity, not the total one), P is the current price of the asset, and P_b and P_a are still the price ticks (boundaries of the reserve).

Consequently, the Delta (Δ) and the Gamma (Γ) of the LP position value are expressed as:

$$\Delta = \frac{\partial V}{\partial P} = L \left(\frac{1}{\sqrt{P}} - \frac{1}{\sqrt{P_b}} \right)$$

$$\Gamma = \frac{\partial^2 V}{\partial P^2} = -\frac{L}{2P^{\frac{3}{2}}}$$

Now using the delta and gamma of powers previously computed too, hedging the delta and gamma of our portfolio value can be done. Assume you need w units of ETH² to make the portfolio Gamma-neutral:

$$w \cdot \frac{2}{1 - hT} - \frac{L}{2p^{\frac{3}{2}}} = 0$$

$$w = \frac{(1 - hT) \cdot L}{4p^{\frac{3}{2}}}$$

And then you would need to take z units of first-order perpetual to make the portfolio Delta-neutral (the expectation of the first-order perpetual funding fee is 0):

$$z + L \left(\frac{1}{\sqrt{p}} - \frac{1}{\sqrt{p_b}} \right) + w \cdot \frac{2S}{1 - hT} = 0$$

In other words, you need to short z units of first-order perpetual. In practice, as a static hedge entered at a certain price, we just need to fix w and z with the value at this point. To summarize, the following portfolio has 0 Delta and 0 Gamma at the entry point (x_{real}, y_{real}) , which are the real quantities provided by the LP:

- Providing x_{real} ETH + y_{real} USDC to the ETH-USDT pair.
- Long w units of ETH² power perps.
- Short z units of ETH perpetual.

Please note that L represents the measure of liquidity provided exclusively by this liquidity provider. It is determined by the following equation:

$$(x_{real} + L\sqrt{P_b})(y_{real} + L\sqrt{P_a}) = L^2$$

Here, x_{real} and y_{real} are the quantities provided by the LP.

Attempting to calculate L directly using the virtual equation of the LP results in a complex expression. Therefore, it is more practical to use our derived formula from the "Liquidity Provider Interaction" section.

As a result, To hedge this impermanent loss using power perpetuals, an LP would need to balance the portfolio to be Gamma-neutral and Delta-neutral. This involves taking positions in both ETH second-order and first-order perpetuals.

The cost of constructing such a hedged portfolio primarily involves the funding fees associated with power perpetuals and futures positions. While futures funding fees are typically negligible, the power perpetuals' funding fees, which are determined by their pricing formula, represent the primary cost of the hedge. Despite this cost, the income generated from liquidity provision often outweighs the hedging cost, making it a profitable endeavor, especially in high-volume trading pairs.

No substantial costs of dynamic rebalancing come along with this strategy. Indeed usually when considering the hedging of portfolio value using linear derivatives like futures, it becomes necessary to frequently rebalance the hedging position. This is due to the rapidly changing Delta of the portfolio value. This process is known as dynamic Delta hedging, which is causing a substantial cost. A more efficient alternative involves using a tool specifically designed to hedge the Gamma component. By hedging Gamma, we are essentially stabilizing Delta over a wider range of the underlying asset's price movements, which in theory reduces the need for frequent adjustments to the hedge. In this context, we saw that Power Perpetuals emerge as an ideal instrument for Gamma hedging.

While power perpetuals can reduce the frequency of rebalancing, they do not completely eliminate the need for it. Market conditions can change, and the effectiveness of a hedge can vary over time. Factors such as significant price movements, volatility changes, and the passage of time can all influence the hedge's effectiveness. It is crucial for LPs to continuously monitor the hedged position and make adjustments when necessary. The

stability offered by power perpetuals in Gamma hedging does ease the management process, but it doesn't render it entirely hands-off. Vigilance is still required to ensure that the hedge remains effective against market dynamics.

In conclusion, the integration of Power Perpetuals as a hedging instrument within the Wabka protocol on SUI empowers liquidity providers with advanced risk management capabilities, leading to enhanced stability, increased confidence, and sustainable yield generation. For a more in-depth understanding, further research will be undertaken by Wabka's team on power perpetuals. Such extended study will offer a complete comprehensive insight into the domain.

10. INNOVATIVE ORACLE

10.1. ORACLES AND THEIR IMPORTANCE

A price oracle is a crucial component that provides smart contracts with external information about the current market prices of various assets. Blockchains are typically isolated environments, and smart contracts within these blockchains have limited access to information outside the blockchain itself. In our context, a decentralized exchange do not need necessarily a price oracle, however, providing an oracle for external sources is important for keeping track of

A price oracle acts as a bridge between the blockchain and the real world by fetching and supplying accurate and up-to-date information on asset prices. This is especially important for decentralized applications (DApps) and smart contracts that involve financial transactions, such as decentralized exchanges, lending protocols, and other DeFi platforms.

Here are two key types of price oracles:

1. On-Chain Oracles:

- **Decentralized Data Sources:** On-chain oracles rely on decentralized data sources, which are usually aggregated from multiple independent providers. These providers may include API services, other blockchain networks, or decentralized data networks.
- **Smart Contract Execution:** The oracle smart contract is responsible for fetching and verifying the external data and then providing it to other smart contracts on the blockchain.

2. Off-Chain Oracles:

- **Centralized Data Sources:** Off-chain oracles use centralized data sources to fetch information from the real world. This can involve APIs provided by external entities.
- **Manual Input or Trusted Entities:** Some off-chain oracles might rely on manual input or data provided by trusted entities. However, this introduces a level of centralization and potential vulnerability to manipulation.

10.2. ORACLES MANIPULATIONS

Theoretically, one can always attack and manipulate an oracle, hence the need to create robust ones that send realistic and reliable data to external sources. To comprehend oracle manipulations, we delve into this issue by **providing an example** then a practical and famous oracle attack: **The mango market attack**, we also finish by presenting the **TWAP oracle** as a way to strengthen oracles at Uniswap.

10.2.1. How can one attack an oracle ?

Let's suppose an attacker aims to manipulate the price of tokens on Uniswap and subsequently exploits this manipulated price on a third-party decentralized finance (DeFi)

service that depends on Uniswap price oracles. Here's a step-by-step explanation of how the attacker might proceed:

1. **Token Acquisition:** The attacker begins by acquiring a significant amount of a specific token available on a Uniswap liquidity pool.
2. **Inflating Token Price on Uniswap:** With a large volume of the token, the attacker executes a series of trades on Uniswap to buy the token, strategically designed to increase the token's price within the Uniswap pool.
3. **Exploiting Higher Price on DeFi Service:** Once the token's price is artificially inflated on Uniswap, the attacker proceeds to the third-party DeFi service that relies on Uniswap price oracles. The DeFi service may use Uniswap's reported token prices to make various financial calculations.
4. **Benefiting from Manipulated Price:** By exploiting the higher price reported by Uniswap, the attacker engages in transactions or activities on the third-party DeFi service, potentially making a profit.
5. **Token Dumping and Profit:** After benefiting from the manipulated price on the third-party DeFi service, the attacker may sell or trade the tokens back into the Uniswap pool at the original or lower market price, realizing a profit.
6. **Mitigations:** To mitigate such attacks, Uniswap employs a strategy where it tracks prices at the end of a block, after the last trade of a block. This ensures that the prices used for calculations, including those provided to third-party DeFi services, are determined after all trades within a block are completed, reducing the risk of in-block price manipulations.

10.2.2. The mango markets example

Mango Markets, operating on the Solana blockchain, positioned itself as a decentralized exchange (DEX) offering a comprehensive suite of services for traders. With a vision to serve as a centralized hub, Mango catered to traders seeking spot markets, perpetual futures, and lending capabilities. Notably, Mango facilitated cross-margin trading with leverage of up to 5x, utilizing a margin trading protocol that permitted cross-margin trades involving various crypto assets as collateral. To ascertain the listed value of crypto assets on the exchange, Mango relied on an oracle that calculates the price of an asset by using a moving average of centralized exchange price feeds. However, in 2022 the protocol was attacked by Avraham Eisenberg (who publicly confirmed committing the attack, however it was a legal market operation for him) causing a traumatic loss to Mango Markets. We explain the strategy:

First, the attacker initiated the exploit by funding two wallets with \$5,000,000 USDC each. Wallet 1 and Wallet 2 were successively funded with 5 million USDC. Subsequently, Wallet 1 placed a significant sell offer for 483 million \$MNGO perpetual futures at 3.8 cents per unit.

In the attack strategy, Wallet 2 purchased all 483 million \$MNGO perpetual futures from Wallet 1 at the offered price. The attacker then engaged in spot \$MNGO purchases across multiple exchanges, causing the price to spike to \$.91. Profits and losses between Wallet 1 and Wallet 2 were settled, providing Wallet 2 with a substantial unrealized profit from the 483 million long position.

Leveraging this unrealized profit, the attacker secured a \$116 million loan on the platform, using crypto tokens as collateral. In a swift sequence of actions, the attacker withdrew the loaned crypto assets, relying on the unrealized profits as collateral. The entire process

unfolded rapidly within a matter of minutes.

As the spot \$MNGO prices corrected down to 2 cents, falling below the initial purchase prices on Wallet 1, the attacker effectively extracted over \$116 million from Mango Markets. Importantly, Mango Markets was not hacked; rather, the attacker strategically exploited the platform's trading features within the defined smart contract parameters, capitalizing on token liquidity to undermine and deplete Mango Markets.

What actually happened is that the oracle accurately reported the current price of \$MNGO, by using the moving averages from a few exchanges to allow borrowing against it on Mango Markets. It's just that the price of \$MNGO at the time of the exploit was manipulatively inflated for just enough time for the attacker to borrow a tremendous sum against this temporarily inflated price.

10.2.3. Univ3 TWAP oracle

Uniswap employs a unique approach in managing price information. Unlike traditional methods, Uniswap maintains a record of accumulated prices over time. Instead of tracking individual prices, it focuses on the sum of prices at each second throughout the history of a pool contract.

$$a_i = \sum_{i=1}^t p_i$$

This methodology enables the calculation of the time-weighted average price between two points in time (t_1 and t_2) by subtracting the accumulated prices at these points (a_{t1} and a_{t2}) and dividing by the time difference:

$$p_{t1,t2} = \frac{a_{t2} - a_{t1}}{t2 - t1}$$

While this was the approach in Uniswap V2, in V3, there is a slight modification. The accumulated value is the current tick (which is $\log_{1.0001}$ of the price):

$$a_i = \sum_{i=1}^t \log_{1.0001} P(i)$$

Instead of averaging prices, the geometric mean is now employed:

$$P_{t1,t2} = \left(\prod_{i=t1}^{t2} P_i \right)^{\frac{1}{t2-t1}}$$

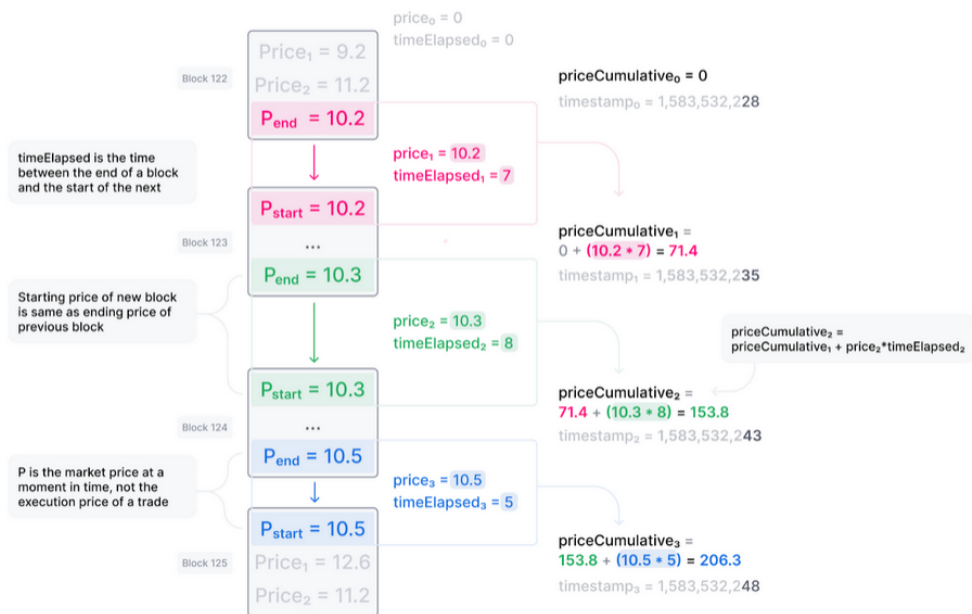
To calculate the time-weighted geometric mean price between two points in time, the accumulated values at these points are subtracted, divided by the time difference, and then multiplied by 1.0001:

$$\log_{1.0001}(P_{t1,t2}) = \frac{\sum_{i=t1}^{t2} \log_{1.0001}(P_i)}{t2 - t1} = \frac{a_{t2} - a_{t1}}{t2 - t1} = \frac{t2 - t1}{a_{t2} - a_{t1}}$$

$$P_{t1,t2} = 1.0001^{\frac{a_{t2} - a_{t1}}{t2 - t1}}$$

Uniswap V2 faced challenges in referencing historical prices, relying on third-party blockchain data indexing services. Uniswap V3, however, addresses this by allowing storage of up to 65,535 historical accumulated prices, simplifying the calculation of historical time-weighted geometric prices.

Storing Cumulative Price Data On-Chain



An Illustration of the algorithm behind the TWAP

10.3. PYTH ORACLE ON SUI

For our advanced features that we intend to implement our main used oracle would be the **pyth oracle** already implemented in the SUI ecosystem. With pyth oracle we would then pull the live exchange rates and provide them to the smart contract. (Need to discuss why we would need that and if oracles are not just another problem that can cause attacks for us) however we also intend to provide a personalized oracle that we present in the next part:

10.4. ORACLE AS AN OPTIONNAL HOOK: THE GEOMETRIC MEAN ORACLE

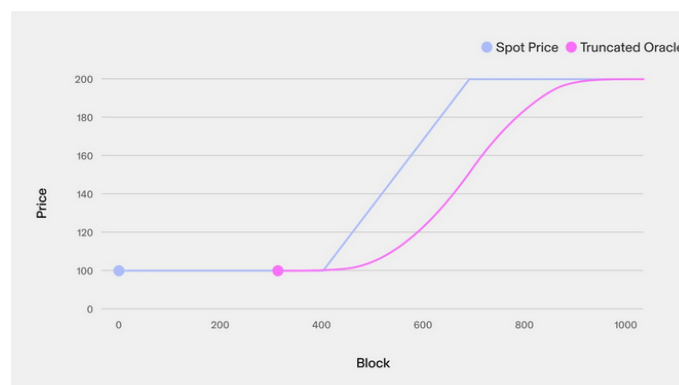
The last part of this chapter showed how useful oracles are but most importantly **how dangerous it is when oracles are not solid and robust** hence the need for trustworthy oracles when one decides to build one.

At WABKA we indeed tends to use a robust customized oracle that can be used by the liquidity provider when setting a pool, this would be completely optionnal, hence letting the choice for the user to link this oracle to his pool or not.

The oracle we intend to build is similar and closely related to the one announced by Uniswap v4, **the truncated oracle hook**.

The truncated oracle in Uniswap v4 is an onchain price oracle designed to record asset prices in a liquidity pool using a geometric mean formula, with a key feature being that the price is truncated, or limited, to only move up or down a maximum amount within a single block. This truncation smooths out the price impact of large trades over time, making it harder for manipulators to influence prices, as they would need to sustain their efforts over many blocks, significantly increasing the cost of manipulation.

In the mechanics of the truncated oracle, the smart contract diligently archives the pool's pricing points, which are quantified as ticks. During a transaction or liquidity provision modification, the contract scrutinizes the present pool tick against its last saved tick. Uniswap has established a benchmark of 9,116 ticks for this comparison based on their empirical studies. If the observed tick variance is under this predefined benchmark, the contract aligns with the new tick. If the discrepancy surpasses the 9,116 tick threshold, the contract constrains the tick's adjustment span to $\pm 9,116$ for that block, averting drastic fluctuations within that timeframe. This measured approach curtails the amplitude of price swings, thus preventing swift manipulation. For our adaptation within the SUI ecosystem, we will embark on a meticulous examination to identify a suitable equivalent to Uniswap's 9,116 tick parameter, ensuring our oracle's resilience and reliability.



A graphical illustration of the smoothness of price change in terms of the number of blocks for the Truncated price oracle compared to the TWAP

Thanks to this new oracle model, malicious actors such as Avraham presented below won't stand a chance manipulating the market with external landing defi protocols because after performing a big trade, the price can only go futher up to the threshold decided and hence

an enormous change won't happen. Besides preventing attacks, the oracle also permits **a very low price impact** by updating smoothly the price.

On top of the TWAMM which itself permits to combat the price impact and to give more freedom to the user, a geometric mean oracle will ensure more robustness and even better shield for price manipulations.



CUSTOMIZABLE AMM'S



11. THE STABLE COIN SWAP FUNCTION

In WABKA, we allow the liquidity provider not only **to choose the assets to provide, the fees to implement or to implement a TWAMM or not**, but we also let him favor stability and composability over volatility and speculation by **choosing the AMM he would like to use**. Indeed, one of our pre implemented hook would be the stablecoin automated market maker inspired from the famous stablecoin protocol curve finance. The latter is known for it's efficiency in terms of pairing assets that behave similarly together or stablecoins.

Let us explain the basics of the concept :

A Constant Sum Market Maker (CSMM) is an Automated Market Maker (AMM) mechanism where the total value of assets in a liquidity pool is kept constant, the mathematical formulation is $x+y=k$ unlike the AMM already encountered before which uses an invariant of $x*y=k$. A CSMM is typically suited for assets that are expected to have a stable value relative to each other. The CSMM model guarantees no price slippage, which stands in contrast to the Constant Product where slippage can occur. However, CSMMs are vulnerable to corner equilibriums during volatile market conditions, an issue that CPMM/CMMM models are designed to avoid. To harness the advantages of both models, **a Hybrid Function Market Maker (HFMM) can be proposed**, which blends the mechanisms of CPMM and CSMM. This hybrid is aimed at enhancing the exchange of stablecoins and is constitutes the basis of curve protocol, here we provide a first simpler form than the one detailed in the Curve Finance whitepaper, notably excluding the consideration of trading fees for the purpose of this explanation. Since we first provide the general setting for many assets in the pool and not only two, we will then use the general function of the CPMM which is the Constant mean market maker function.

We analyze a Constant Mean Market Maker (CMMM), ensuring the product of the tokens' quantities and their respective weights equals a constant value k , thus:

$$\prod_{i=1}^n x_i^{\frac{1}{a_i}} = k$$

In this scenario, each weight a_i is uniform across tokens, leading to $a_1 = a_2 = \dots = a_n = \frac{1}{n}$. This uniformity results in the average sum of token quantities equating k :

$$\frac{\sum_{i=1}^n x_i}{n} = k$$

We also examine a Constant Sum Market Maker (CSMM) setup, with analogous coefficients for tokens, delivering a combined sum that also equals k . The rationale behind choosing these uniform distributions for weights and coefficients will be clarified in the subsequent discussion. The bilateral price ratio for CMMM is given by $P_{x_j}/x_i = x_j/x_i$, while for CSMM, it remains constant at unity.

Our objective is to amalgamate these two market mechanisms into a singular, hybrid form. The most straightforward approach is to average their respective market-making equations with a variable weight λ :

$$\lambda \frac{\sum_{i=1}^n x_i}{n} + (1 - \lambda) \left(\prod_{i=1}^n x_i^{\frac{1}{a_i}} \right) = k, \quad \lambda \in [0, 1]$$

The fixed weighting in the initial hybrid model does not reflect the fluid nature of the token reserves (x_1, x_2, \dots, x_n) . Ideally, our model should apply a greater weight to the CMMM as reserves dwindle to amplify the price response, and favor the CSMM when reserves are balanced to stabilize pricing and curtail slippage.

Constructing a more responsive λ requires examining the arithmetic mean (AM) of the reserves $A = \frac{1}{n} \sum_{i=1}^n x_i$ and the geometric mean (GM) $G = \left(\prod_{i=1}^n x_i\right)^{\frac{1}{n}}$. Based on mathematical considerations of these means, we recognize that $0 < \frac{G}{A} \leq 1$ provides a nuanced approach to weighting λ , where $\frac{G}{A}$ aligns with the relative equality of the token reserves.

Incorporating our results the invariance function would then be :

$$\left(\frac{G}{A}\right) \frac{\sum_{i=1}^n x_i}{n} + \left(1 - \frac{G}{A}\right) \left(\prod_{i=1}^n x_i^{\frac{1}{n}}\right) = k, \quad \text{where } G = \left(\prod_{i=1}^n x_i\right)^{\frac{1}{n}} \text{ and } A = \frac{1}{n} \sum_{i=1}^n x_i$$

Curve Finance employs an Automated Market Maker (AMM) model specifically tailored for the exchange of stablecoins inspired from the model above, it's mathematical foundations are quite complex compared to the ones already described, however the official whitepaper provided by curve documents explain the transition from the base case to their final formulation very well, we will now describe the general formulation of the curve model:

For a pool with n assets, the invariant can be represented as:

$$D = \sum_{i=1}^n x_i$$

Where D is the sum of all assets x_i in the pool, adjusted for their respective prices.

Curve introduces the concept of A , the amplification coefficient, to the invariant. The amplification is used to make the curve more flat when the asset quantities are close to being equal, which means that the assets are being traded at or near the targeted price, typically 1.0 for stablecoins. We can then clearly see that A represents somehow the new .

The stableswap invariant is then defined as:

$$A \cdot D \cdot n^n + D = A \cdot n^n \cdot \sum_{i=1}^n x_i + \prod_{i=1}^n x_i$$

When a trade occurs, the invariant must remain constant. To ensure this, when an asset x_i is increased by an amount Δx_i , there must be a decrease in one or more of the other assets to maintain the value of the invariant.

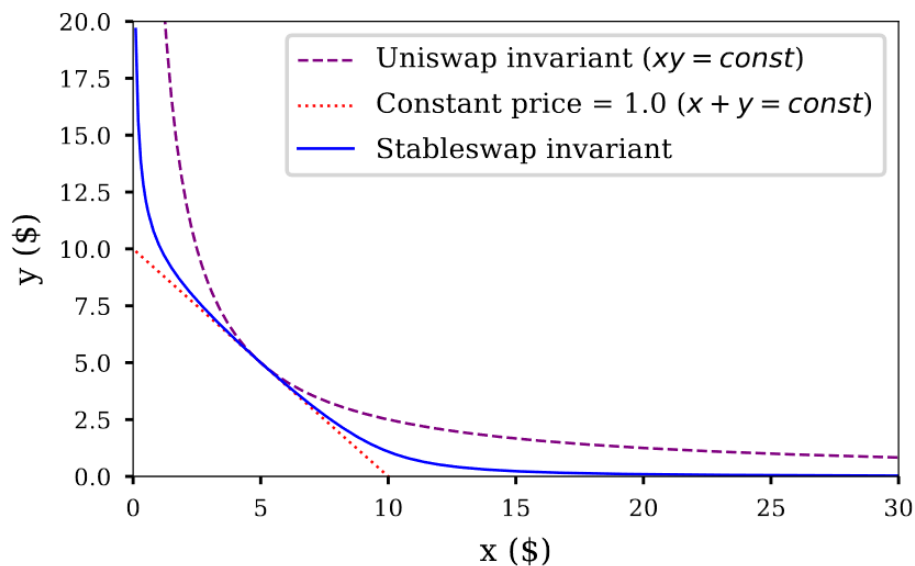
The leverage provided by A allows the liquidity to be more effective when prices are close to the targeted price, leading to the reduced slippage. The specific formula for a pool of two assets would simplify and is used to calculate the new balances after a trade.

the actual implementation involves solving this equation iteratively to find the new balances after a trade, as closed-form solutions are not feasible due to the non-linearity of the equation. The liquidity is thus 'concentrated' near the 1.0 price due to the flat part of the curve, which is a result of the high AAA value.

To calculate the exact new balances and prices, one would typically use a numerical method such as Newton-Raphson for finding roots of equations, the algorithm runs off chain and we record the result on chain which encodes the amount of a certain token to achieve the new balance.

All these detailed explanation is to demonstrate that WABKA intends to implement a hook that can take the LP to a protocol very similar to curve v2 for stablecoins inside SUI ecosystem by providing this AMM for the pool and not the standard implemented one that revolves around concentrated liquidity.

WABKA intends then to be stablecoin friendly with its unique features on the SUI ecosystem.



A graphical illustration of the stablecoin invariant, we can see that the more we approach the price of 1.0, the more the curve flattens to give more weight to the constant sum function.



THE TEAM



PRESENTATION OF THE TEAM

Our team, which includes students from EPFL and ETH Zurich—both highly regarded for their tech and science programs—comprises skilled developers and mathematicians. Each of us has stepped up to significant roles, whether through research, internships or blockchain projects. Our practical experience has earned us a reputation in the SUI community, **highlighted by our second-place finish at the first European SUI hackathon at EPFL.**

Arda Özenç: I completed the preparatory year at EPFL and am currently pursuing a degree in mechanical engineering. Alongside my studies, I've cultivated a passion for web development, blockchains and smart contracts.

Walid Sofiane: After completing a bachelor degree in mathematics at EPFL, I am currently enrolled in a masters degree in financial engineering at the same university, I mainly focus on quantitative finance as well as the application of machine learning in the sector of finance.

Adam Bouabda: Pursuing a Master's in Applied Mathematics at ETH Zurich, focusing on machine learning and financial mathematics. DeFi enthusiast in my personal time.

Zayed Kriem: I am currently a third-year Mathematics student at EPFL, where I have developed a strong foundation in mathematical concepts and a keen interest in their practical applications.

Alper Ozyurt: I am a first-year computer science student at EPFL, and I have dedicated the past three years to developing full-stack web3 applications. My focus lies in decentralized blockchain technologies and their mathematical applications, in addition to traditional development methods.

TASK ALLOCATION FOR TEAM MEMBERS

Development Team:

- **Development:** Responsible for coding and programming tasks related to the project's software, smart contracts, and technical infrastructure.

Research Team:

- **Research:** Engaged in comprehensive research to gather relevant information, insights, and data pertinent to the project.

Whitepaper Team:

- **Updating Live Whitepaper:** Responsible for continuously updating the official whitepaper, ensuring it reflects the latest information about the project. Note: This will only be officially available at the launch.

Website Team:

- **Website Design (Beginning):** Initial design and development of the project website, focusing on layout and structure.
- **Website Design (Mid State):** Ongoing updates and enhancements to the website, ensuring it aligns with the project's progress and goals.
- **Website Design (End State at Official Launch):** Finalizing and polishing the website design to be ready for the official launch.

Social Media and partnership Team:

- **Social Media Updates:** Regularly updating and managing social media platforms, sharing current news, project updates, and engaging with the community with the help of Social Media User Interface (SUI).
- **Partnerships :** This is more of a long term goal, which can be useful for oracles as an example (chainlink)

Blog Team:

- **Creating a Blog:** Establishing and maintaining a blog platform for the project to share different research findings, updates, and other relevant content.
- **Updating Blog:** Regularly adding new blog posts, ensuring that the content is informative, engaging, and aligns with the project's narrative.



OUR VISION OF WABKA



WABKA aims to be the most complete DEX ever created on the SUI blockchain, however, we have short term, mid term and long term goals that we explain as the following:

11.1. SHORT TERM

Our short term goal is to first implement a complete DEX that allows trading as well as the concept of hooks, namely the hooks we will define ourselves and that we offer to the user, hence the short term priorities are : **the concentrated liquidity implementation as well as the already implemented TWAMM (already implemented in the hackathon) and the dynamic fees hook.** A part of the team will also conduct extensive research in order to delve through power perpetuals.

11.2. MEDIUM TERM

Our next step would be to continue with the hooks, namely to **implement a robust personalized oracle** inspired from the uniswap geometric oracle, in order to prepare perfectly for our power perpetuals implementation and to assert security as already explained before.

11.3. LONG TERM VISION

In the pursuit of long-term advancements, our primary objective is to establish a complete framework for power perpetuals within the Sui ecosystem. We also aim to develop novel functionalities and hooks, with a special emphasis on leveraging machine learning techniques. An example of such an application could be the use of regression analysis for dynamically adjusting fees.



BIBLIOGRAPHY



- Atis, E. (2021). Uniswap v3 Liquidity Math. from <https://atiselsts.github.io/pdfs/uniswap-v3-liquidity-math.pdf>
- Uniswap. (2023). Uniswap v4 Truncated Oracle Hook. 2023, from <https://blog.uniswap.org>
- Uniswap. (2023). Uniswap v4 TWAMM Hook. 2023, from <https://blog.uniswap.org/v4-twamm-hook>
- Curve.fi. (2020). StableSwap - efficient mechanism for Stablecoin liquidity. 2023, from <https://classic.curve.fi/files/crypto-pools-paper.pdf>
- Cronfi. (2023). TWAMM - Time Weighted Average Market Maker. 2023, from <https://docs.cronfi.com/twamm/>
- Uniswap. (2021). Uniswap v3 Core. from <https://uniswap.org/whitepaper-v3.pdf>
- AuthorLastName, A. F. (2022). Title of the Bachelor Thesis. 2023, from <https://pub.tik.ee.ethz.ch/students/2022-FS/BA-2022-19.pdf>
- Chainalysis. (2022). Oracle Manipulation Attacks on the Rise. 2023, from <https://www.chainalysis.com/blog/oracle-manipulation-attacks-rising/>
- Chainlink. (2023). What is an Automated Market Maker (AMM)? 2023, from <https://chain.link/education-hub>
- AuthorLastName, A. F. (2021). Title of the Article. Journal of Financial Innovation, 7(1). <https://jfin-swufe.springeropen.com/articles/10.1186/s40854-021-00314-5>



SUMMARY



In summary WABKA intends to create the most complete DEX and trading platforms on SUI blockchain offering unique features and customizations that allow to drastically reduce price impact and slippage, it also leverages the zk login and intends to build power perpetuals.

LIST OF FIGURES

An illustration of a coordinate system that represents the quantity of liquidity locked in a certain price range represented by ticks.	8
Each of the lines represent a certain price, between two lines (the price range) a Constant product function takes place in the form of a portion of a hyperbola.	9
A clear graphical illustration of the transition from the virtual quantites to the real quantities by substracting the offset quantities.	11
An illustration of the swap algorithm	16
Data aggregator vs TWAMM embedded AMM, the TWAMM competes for a better trade experience for traders that do not care much about executing their trades instantly.	19
As we decide to split more and more, we diminish the abrupt change of price	22
A clear illustration of what happens from the moment the transaction in the TWAMM pool is recorded to the moment when the first external person interacts with the AMM.	23
a midstate of a TWAMM order pushing the price towards the arbitrage price	24
Reaching the arbitrage price after executing a certain number of blocks	25
Arbitrage regulates the price and hence setting up the price and quantities lower.	25
An illustration of a sandwich attack in the case of a taker vs a taker.	27
Return comparison between ETH^2 and 2x leverage on ETH	32
Derivatives comparison table.	33
The impact of funding over time (ETH^2 in USDC)	33
An Illustration of the algorithm behind the TWAP	42
A graphical illustration of the smoothness of price change in terms of the number of blocks for the Truncated price oracle compared to the TWAP	43
A graphical illustration of the stablecoin invariant, we can see that the more we approach the price of 1.0, the more the curve flattens to give more weight to the constant sum function.	48

LIST OF TABLES