Wombat Volatile Pool AMM

Wombat Team

Wombat Exchange

Abstract. The volatile pool is built on top of the DynamicPool invariant curve with additional features, including efficient re-pegging, dynamic haircut, and arbitrage block protection. In the case of volatile assets, the external price oracle is deemed to be vulnerable to manipulation. Therefore, we have introduced a computationally efficient method for measuring market prices and an internal oracle for our re-pegging system.

1 Volatile Pool Invariant Curve

We inherited the AMM through DyanmicPool which:

$$\sum_{k \in \mathcal{T}} L_k p_k^{scale}(r_k - \frac{\alpha}{r_k}) = D \tag{1}$$

, where p_k^{scale} is *price_scale* to define the price of the token k. $\mathcal{T}$ denotes as the total type of tokens in the system. L_k and r_k are the liability and coverage ratio of token k. With the configurable amplification factor, α, it formulates the AMM invariant curve with the invariant term, D.

1.1 Determination of the Market Price of an Asset

There are two other defined prices for our implementation to update our AMM $price_k^{scale}$, which are internal price oracle, $price_k^{oracle}$, and the defined market price, $price_k^{last}$. Since our internal oracle is calculated from the defined market price of a token, we need to accurately calculate the swapping amount between the volatile asset and the numéraire, usually a stablecoin in a volatile pool. For a volatile pool with three tokens, our AMM can be expressed as follows:

$$L_x p_x(\frac{A_x}{L_x} - \frac{\alpha}{\frac{A_x}{L_x}}) + L_y p_y(\frac{A_y}{L_y} - \frac{\alpha}{\frac{A_y}{L_y}}) + L_z p_z(\frac{A_z}{L_z} - \frac{\alpha}{\frac{A_z}{L_z}}) = D$$

, where we temporarily replace the notation of p_k^{scale} to p_k.

When we swap asset x, A_x, to asset y, A_y, with an infinitely small amount, we take a derivative with respect to A_x:

$$\begin{aligned} L_x p_x \frac{1}{L_x}(\frac{dA_x}{dA_x}) - \alpha L_x^2 p_x \frac{d(\frac{1}{A_x})}{dA_x} \\ + L_y p_y \frac{1}{L_y}(\frac{dA_y}{dA_x}) - \alpha L_y^2 p_y \frac{d(\frac{1}{A_y})}{dA_x} \\ + L_z p_z \frac{1}{L_z}(\frac{dA_z}{dA_z}) - \alpha L_z^2 p_z \frac{d(\frac{1}{A_z})}{dA_z} = 0 \end{aligned}$$

$$\Rightarrow p_x + \frac{L_x^2 p_x \alpha}{A_x^2} + [\frac{dA_y}{dA_x} \cdot p_y + \frac{L_y^2 p_y \alpha}{A_y^2} \cdot \frac{dA_y}{dA_x}] + [\frac{dA_z}{dA_x} \cdot p_z + \frac{L_z^2 p_z \alpha}{A_z^2} \cdot \frac{dA_z}{dA_x}] = 0$$

, where $\frac{dA_z}{dA_x} = 0$ when we swap *x to y*. Thus, we have:

$$-\frac{dA_y}{dA_x} = \frac{p_x + \frac{L_x^2 p_x \alpha}{A_x^2}}{p_y + \frac{L_y^2 p_y \alpha}{A_y^2}} = \frac{p_x + \frac{p_x \alpha}{r_x^2}}{p_y + \frac{p_y \alpha}{r_y^2}} \tag{2}$$

This is the formula to calculate the market price of token x given the $price^{scale}$ of tokens x and y and their corresponding coverage ratio.

On the smart contract level, Curve Finance implements a small amount (dx) to determine the last recorded price, $price^{last}$. In contrast, our AMM algorithm enables precise calculation of the spot price, eliminating counting errors and computational inefficiencies.

1.2 Uniformity of Spot Price

Another difference between Curve and our AMM algorithm lies in the computation of balances during the swapping process. Curve considers the token balance of z when swapping x to y. Meanwhile, with Wombat, we isolate the balance of the remaining tokens, enabling true scalability.

However, some may be concerned about the uniformity of spot price across tokens, particularly when traders gain advantages by routing through another token within the same pool. To address this concern, we show the uniformity of spot price that the market price of swapping from x to y is equivalent to the price of swapping x to z to y, $p_{x \to y}^{last} = p_{x \to z \to y}^{last}$. Algebraically, we can show:

$$(-\frac{dA_y}{dA_x})(-\frac{dA_z}{dA_y}) \equiv (-\frac{dA_z}{dA_x}) \to \frac{\Delta_{USDT}}{\Delta_{BTC}} \frac{\Delta_{ETH}}{\Delta_{USDT}} \equiv \frac{\Delta_{ETH}}{\Delta_{BTC}}$$

Following the previous derivation and swap *y to x and x to z*, we have

$$-\frac{dA_z}{dA_y} = \frac{p_y + \frac{p_y \alpha}{r_y^2}}{p_z + \frac{p_z \alpha}{r_z^2}}$$

$$\textit{and R.H.S:} -\frac{dA_z}{dA_x} = \frac{p_x + \frac{p_x \alpha}{r_x^2}}{p_z + \frac{p_z \alpha}{r_z^2}}$$

Thus,

$$\frac{dA_y}{dA_x}\frac{dA_z}{dA_y} = \left(\frac{p_x + \frac{p_x \alpha}{r_x^2}}{p_y + \frac{p_y \alpha}{r_y^2}}\right) \cdot \left(\frac{p_y + \frac{p_y \alpha}{r_y^2}}{p_z + \frac{p_z \alpha}{r_z^2}}\right) = \frac{p_x + \frac{p_x \alpha}{r_x^2}}{p_z + \frac{p_z \alpha}{r_z^2}} = -\frac{dA_z}{dA_x}$$

We can take the spot price between the volatile asset and the numéraire. i.e. $\frac{\Delta_{USDT}}{\Delta_{BTC}}$ for the spot price of BTC.

2 Repegging Mechanism

2.1 Updating Internal Prices

We will have the following variables created for the re-pegging process.

1. $price^{last}$: spot price to resemble the market spot price.

$$-\frac{dA_{num\acute{e}raire}}{dA_{volatile\_asset}}$$

2. $price^{oracle}$: the exponential moving average (EMA) price oracle, where α_{EMA} determines the impact of the spot price, t_i refers to the discrete block-time when a swap executed and $t_1 - t_0$ is the time duration between swaps. T_{EMA} is a configurable parameter that determines the sensitivity of the EMA oracle. The initial price oracle at time 0 when launching the system is equal to the price scale.

$$\begin{aligned} p_{t_1}^{oracle} &= price_{t_1}^{last}(1 - \alpha_{EMA}) + \alpha_{EMA}(p_{t_0}^{oracle}) \\ \alpha_{EMA} &= 2^{-\frac{t_1 - t_0}{T_{EMA}}} \end{aligned}$$

3. $price^{scale}$: the price of token k identified in the AMM, where χ is a configurable parameter that determines the re-pegging distance between the new price scale and the price oracle.

$$\frac{p_{i,t+1}^{scale}}{p_{i,t}^{scale}} - 1 = \frac{\chi}{\sqrt{\sum(\frac{p_{j,t}^{oracle}}{p_{j,t}^{scale}} - 1)^2}}\left(\frac{p_{i,t}^{oracle}}{p_{i,t}^{scale}} - 1\right) \quad (3)$$

We can understand the process as re-pegging the LHS relative ratio, $(\frac{p_{i,t+1}^{scale}}{p_{i,t}^{scale}} - 1)$, based on the $(\frac{p_{i,t}^{oracle}}{p_{i,t}^{scale}} - 1)$ by the relative distance of $(\frac{\chi}{\sqrt{\sum(\frac{p_{j,t}^{oracle}}{p_{j,t}^{scale}} - 1)^2}})$.

With re-arranagement, we can get a new price scale by:

$$p^{scale}_{i,t+1} = \frac{p^{scale}_{i,t}(\sqrt{\sum(\frac{p^{oracle}_{j,t}}{p^{scale}_{j,t}} - 1)^2} - \chi) + p^{oracle}_{i,t} \cdot \chi}{\sqrt{\sum(\frac{p^{oracle}_{j,t}}{p^{scale}_{j,t}} - 1)^2}}$$

2.2 Repegging Conditions

For every swap, we check two re-pegging conditions which are 1) the sum of relative deviation of price oracle and price scale of all tokens, and 2) the system health with extra re-pegging fee, r^*, introduced at 2.2.1 and 2.2.3.

2.2.1 Relative Deviation of Price Oracle and Price Scale

We impose another condition on top of equation (3), and we only update the price scale if

$$\sqrt{\sum(\frac{p^{oracle}_{j,t}}{p^{scale}_{j,t}} - 1)^2} > \chi$$

This condition ensures no re-pegging occurs if the sum of relative distance of all token between $p^{oracle}_{j,t}$ and $p^{scale}_{j,t}$ is too low. We should not re-peg when the $p^{oracle}_{j} = p^{scale}_{j}$ $\forall$ token j $\in \mathcal{T}$.

2.2.2 Adjustment step Under Volatile Circumstance

To automate the re-pegging step under a volatile circumstance, we further implement another automation that $adjustment\ step = max\{\chi, \frac{norm}{\psi}\}$, where $norm = \sqrt{\sum(\frac{p^{oracle}_{j,t}}{p^{scale}_{j,t}} - 1)^2}$, χ is a configurable parameter as mentioned above and ψ controls the extend of further facilitated step toward the price oracle. When ψ is smaller, it allows the adjustment step to increase more significant. If $\frac{norm}{\psi} > \chi$, we will replace χ in equation (3) by this new value of *adjustment step*. It means that when the sum of the price ratio for each token deviates significantly, we will increase the size of adjustment step for more significant re-pegging toward the oracle.

2.2.3 Maintaining Global Equilibrium and System Health

The price re-pegging process can effectively enhance liquidity efficiency and balance across tokens. Our single LP system allows us to simplify the re-pegging process without convoluted computation. We allocate a configurable portion of the swap fee (from volatile assets) earned by the system back to the token assets. To ensure the system does not generate any spurious value during the re-pegging process, we will re-peg only if the total value of assets of the pool with fee exceeds the total value of liabilities of the pool with their new corresponding prices at the global equilibrium, i.e., $r^* \geq 1$. In other words, the system will check the two re-pegging conditions in every swap. If they are fulfilled, the system will allocate the fee into token assets and update the price scale.

In our dynamic pool whitepaper, *Theorem 2* shows that the change in r^* is positively related to change in price when the coverage ratio of the token is greater than r^*. However, when prices of more than 2 tokens change simultaneously, the impact on r^* is unknown and it may not be viable to re-peg automatically solely based on market prices.

Therefore, we reallocate a portion of the swap fee back into token assets to facilitate the re-pegging process. We show that the addition of swap fee will strictly increase the global equilibrium, r^*.

Recall the calculation of r^* from the dynamic whitepaper r^*:

$$r^* = \frac{D + \sqrt{D^2 + 4\alpha \sum_{k\in\tau} L_k p_k}}{2\sum_{k\in\tau} L_k p_k} \tag{4}$$

We derive the impact of adding the swap fee to the asset for token i by taking partial derivative with respect to A_i from equation (1):

$$\frac{\partial D}{\partial A_i} = p_i + \alpha(\frac{L_i^2 p_i}{A_i^2}) > 0$$

The invariant, D, strictly increases with A_i and r* increases with D from equation (4). This allows us to use the swap fee to facilitate the re-pegging process while ensuring that the global equilibrium remains greater than 1.

3 Dynamic Haircut

$$\begin{aligned} Fee(t) =& Fee_{base} + Fee_{\sigma}\overbrace{(\frac{1}{1+e^{k_v(\beta_v - \bar{\sigma}_{T,x,y})}} + \frac{1}{1+e^{k_{v2}(\beta_{v2} - \bar{\sigma}_{T,x,y})}})}^{volatility} \\ &+ Fee_{imbalance} * \underbrace{(\frac{\Theta e^{-\theta r_{x,t}} + \Theta e^{-\theta r_{y,t}}}{2})}_{imbalance\ level)} \end{aligned}$$

, where $k_i, \beta_i, \Theta, \theta$ are configurable parameters. σ_T represents the average price volatility of the token pair at the last T time. β determines the required magnitude of $\bar{\sigma}_{T,x,y}$ to have an impact on Fee. k_v determines the pace of impact of $\bar{\sigma}_{T,x,y}$ on Fee. Fee_{σ} represents the dynamic fee based on volatility. $r_{x,t}$ and $r_{y,t}$ are the coverage ratio of swapping pair x and y respectively.

We also implement a dynamic fee based on the pool balance calculated from the post-swap coverage ratio. For an asset-liability model, we are less concerned about situations when the asset exceeds liability, but we should increase the fee when the token is in high demand. Therefore, we introduce a single-tail imbalance dynamic fee with an adjustable exponential function to address such situations. This minimizes the fee's impact when the coverage ratio exceeds 1 and encourages users to swap away the low-demand token.

The decrease in asset balances signifies the market's demand and reflects price movement. Introducing the $Fee_{imbalance}$ adaptively increases the reward for the supply side and mitigates the impermanent loss if the coverage ratio does not return to equilibrium.

The rationale behind the dynamic fee can be traced back to the paper on Loss-Versus-Rebalancing (Milionis et al., 2023). LP profit depends on the fee relative to the price volatility. LP losses are more significant when there is higher instantaneous volatility. Consequently, arbitrageurs trade more aggressively in response to price movements. Wombat dynamic fee is designed to address such circumstances automatically.

3.1 Measure of Volatility

We first evaluate the continuous returns of a token as follows:

$$\bar{r}_i = ln(\frac{price_{t_1}^{last}}{price_{t_0}^{last}})\frac{1}{\tau_i} \tag{5}$$

, where i refers to each discrete block-time and $\tau_i = t_1 - t_0$ which represents the lasting time between swap i at t_1 and swap i-1 at t_0. $ln(\frac{price_{t_1}^{last}}{price_{t_0}^{last}})$ is the continuous returns between block-time t_1 and t_0 by second, and we take the average with τ_i to obtain the average continuous returns $\bar{r}_{i,t}$.

With the additive nature of single-period returns, we can use the arithmetic average to compute the standard deviation over a period of time. It significantly saves computation costs on the smart contract implementation level. The average volatility of the swapping pair x and y is:

$$\bar{\sigma}_{T,x,y} = \frac{\sqrt{\frac{1}{T}\sum_i(\bar{r}_{i,x}\cdot\tau_{i,x} - \mu_x)^2} + \sqrt{\frac{1}{T}\sum_i(\bar{r}_{i,y}\cdot\tau_{i,y} - \mu_y)^2}}{2} \tag{6}$$

, T = 86400, indicating the past 24 hours and

$$\mu = \frac{1}{T}\sum_i(\bar{r}_i \cdot \tau_i) \tag{7}$$

4 Arbitrage Block

4.1 Oracle Manipulation Prevention

On the smart contract level, we prevent a situation where we excessively update the $price^{oracle}$. We enforce a condition that updates the $price^{oracle}$ only if the last recorded block time is earlier than the current block time, as expressed by $price\_last\_timestamp < block.timestamp$. As such, we update the $price^{oracle}$ only once per block, minimizing the risk of oracle manipulation.

In addition to the oracle update restriction by block, we also limit the impact of spot price on the oracle. Regardless of the motive, any potential attack can only manipulate the internal oracle price through a large trading volume affecting the spot price. To prevent this and for the smooth operation of the internal oracle, we cap the impact of *price_last* to be 100% of the price scale. Therefore, our EMA formula becomes:

$$price_t^{oracle} = \begin{cases} (min\{p^{last}, 2^{*}p^{scale}\}) \cdot (1 - \alpha_{EMA}) + p_{t-1}^{oracle} \cdot \alpha_{EMA} & \text{if } p^{last} > p^{scale} \\ (max\{p^{last}, \frac{1}{2} * p^{scale}\}) \cdot (1 - \alpha_{EMA}) + p_{t-1}^{oracle} \cdot \alpha_{EMA}, & \text{otherwise} \end{cases}$$

4.2 Deposit and Withdrawal Block

We are aware of one of the re-pegging conditions is based on the global equilibrium, and swapping x to y can cause re-pegging for z. This may increase the coverage ratio of token z due to the increase of the asset by fee, creating an arbitrage opportunity through deposit gains and withdrawal penalties. We encourage arbitrageurs to proactively trigger the re-pegging process by swapping volume, contributing fees, and re-balancing liquidity. However, It's worth noting that arbitrageur may attempt to predict the re-pegging process and sandwich a swap with deposit and withdrawal.

To prevent such situations, on implementation level, we limit

1. The re-pegging process occurs only once per block
2. Actions involving both deposit and withdrawal from the same address cannot be executed within the same block.

These measures further avoid the possibility of the sandwich attack and eliminate the risk-free arbitrage opportunity.

4.3 Withdrawal Stability

Our current withdrawal penalty and deposit gain aim to preserve the global equilibrium coverage ratio, r^*, ensuring that withdrawal and deposit actions do not adversely affect the system's health. However, we recognize the potential concern of 'panic withdrawal,' where the first person to withdraw may pay less withdrawal penalty while causing the coverage ratio to decrease. The current mechanism works well with arbitrageurs, as it automates the price within stable and liquid staking pairs and rebalances the coverage ratio to its state before the withdrawal.

However, given the nature of volatile assets, the pool's composition can be skewed more than a dynamic and stable pool. As a precautionary measure, we introduce a withdrawal mechanism, where after surpassing a certain threshold of coverage ratio, users will partially withdraw in another token, selecting the token pair that offers the best slippage in the pool. Since the withdrawal of another token is irreversible with an imbalance swap fee and slippage, a user cannot reverse a swap by immediately swapping back the second token to their original withdrawal token without incurring a slight slippage loss. Therefore, we can maintain the coverage ratio of the withdrawing token while ensuring the late withdrawers do not incur higher impermanent losses. We find the required amount of this withdrawal mechanism by:

We find the required amount of this withdrawal mechanism by:

$$\begin{aligned}
r_{y,t-1} &= r_{y,t} \\
\frac{A_y}{L_y} &= \frac{A_y - W_y\delta_y + \Delta y}{L_y - W_y} \\
\frac{A_y}{L_y} - \frac{A_y - W_y\delta_y}{L_y - W} &= \frac{\Delta y}{L_y - W_y} \\
r_{y,t-1} - r'_{y,t-1} &= \frac{\Delta y}{L_y - W_y} \\
(r_{y,t-1} - r'_{y,t-1}) \cdot (L_y - W_y) &= \Delta y \\
\Delta r_{withdrawal} \cdot \Delta L_{withdrawal} &= \Delta y
\end{aligned}$$

, where Δy is the amount we require to withdraw in another asset to maintain the coverage ratio after withdrawal. $r'_{y,t-1}$ is the coverage ratio after withdrawal without withdrawing in another assets. W is defined as total liability deducted from withdrawal, and $W\delta$ is the total withdrawal amount of token in asset where $(1-\delta)$ can be understood as withdrawal penalty.

Based on Δy and market condition, we aim to smooth the system by partially withdrawing Δy in another asset and the final withdraw amount $\Delta y'$ is calculated as:

$$\Delta y' = min\{\omega, 1\}\Delta y, where\ \omega = Ke^{-\kappa \cdot r_y}$$

, where ω is capped at 1. K and κ are the parameters that determine the amount we partially withdraw in another asset from Δy. When $\omega = 1$, we withdraw the exact amount in another asset to maintain the coverage ratio of the withdrawing token. When $\omega < 1$, We partially withdraw a smaller fraction, mitigating the impact of withdrawal on the coverage ratio. This mechanism resembles a conventional AMM, where you eventually hold different tokens, most of which have a lower value. However, under this mechanism, as a precautionary measure, we help realize the impermanent loss at a market price upon withdrawal, which is typically better than the early stale price.

5 Empirical Study

We have extracted hourly historical data covering the period from 1st May 2021 to 2022. The lowest and highest prices of BTC and ETH are {29154; 68640} and {1718; 4849} respectively (see Appendix B). The primary objective of this empirical simulation is to subject our volatile pool design to rigorous testing under highly volatile conditions, effectively serving as a stress test.

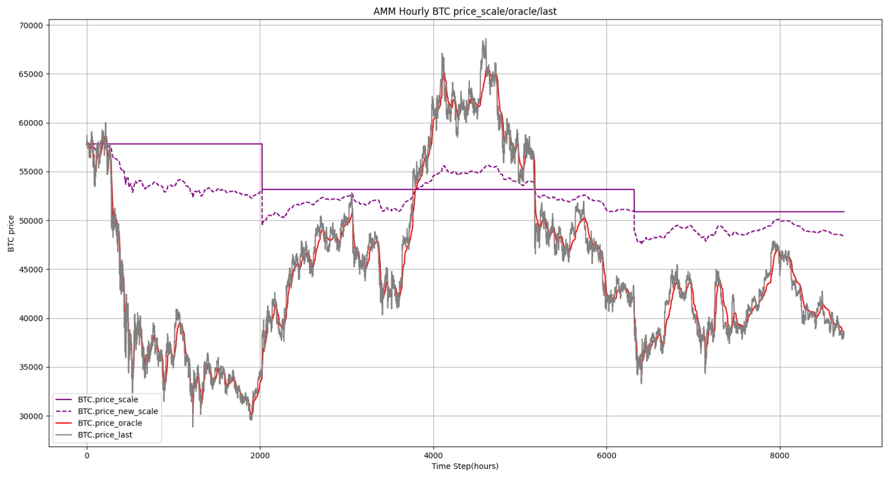
Using historical data can guide an AMM to emulate the price behavior of a centralized exchange without reliance on an external price oracle. To accomplish this, we have developed an arbitrage bot designed to adjust the internal swap price by executing trades that offset the fractional difference between the CEX price and the AMM's internal price (see Appendix B).

The simulation aims to address the following key objectives:

1. To assess the behavior of the re-pegging mechanism and its corresponding impact on impermanent loss.
2. To evaluate the Annual Percentage Yield (APY) generated by our dynamic fee design.
3. To serve as a simulator for parameter setting within our governance community.

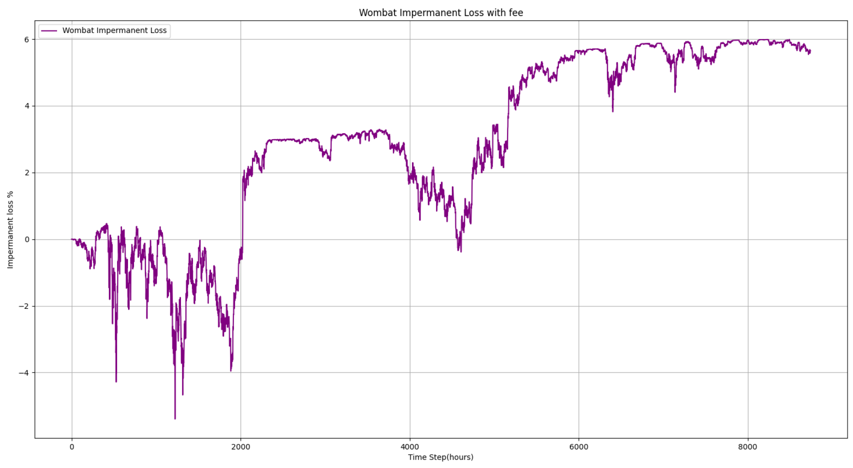
5.1 Repegging Behavior and Impermanent Loss

The following simulated results resemble the settings where we start with 1,500,000 USDT and the equivalent value of BTC and ETH at the beginning of the volatile pool. We can observe the change in impermanent loss based on targeted tokens in our single-asset staking mechanism.



The figure above shows the re-pegging mechanism behavior and our internal oracle under hourly price fluctuation settings[1]. It is important to note that in Wombat's single-staking design, re-pegging is primarily governed by the condition $r^* > 1$. The re-pegging process aims to reallocate liquidity, concentrating it more around the bonding curve. The price scale deviates from the price oracle. Still, the swapping price remains closely aligned with the market price, influenced by the difference in coverage ratio and the corresponding deviated price scale. We intentionally prioritize the system's health over any liquidity reallocation that might lead to a situation where the total system liability exceeds the total assets.

[1]Since our simulation is conducted with hourly price fluctuations, in a real-world setting with highly frequent trading, we would incur more fees and the re-pegging process would improve significantly. Intuitively, evidence can be shown when we simulated daily price fluctuations, which resulted in approximately four times fewer fees compared to hourly price fluctuations, but took twice as long for the re-pegging process.



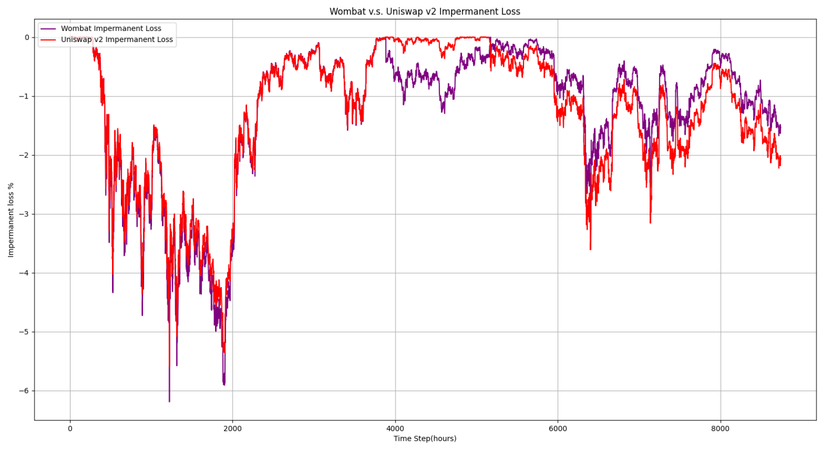
To interpret the benefit for our liquidity provider, we simulate the impermanent loss for the stablecoin liquidity provider (stimulation code is also available for BTC and ETH on Git Hub). For our single staking mechanism, it is intuitive to have more incentive for volatile asset holders to stake in Wombat. In contrast, it must have an attractive reward for stablecoin LP to be exposed to the price fluctuation of volatile assets. We simulate the scenario where liquidity provides a deposit of 20% of the total stablecoin liquidity, and we observe the impermanent loss with fee across the entire period.

As observed from our simulation, even in the absence of close re-pegging between price scale and price oracle, our volatile pool with the dynamic fee can effectively mitigate the impermanent loss. In the long run, with a fee, the dynamic fee will sufficiently cover the impermanent loss.

5.2 Impermanent Loss Comparison

To provide an essential insight into our competitiveness of the single-asset staking and re-pegging mechanism, we arbitrarily make our volatile pool design similar to the Uniswap v2 settings in that we only impose two tokens in our volatile pool without any swap fee.

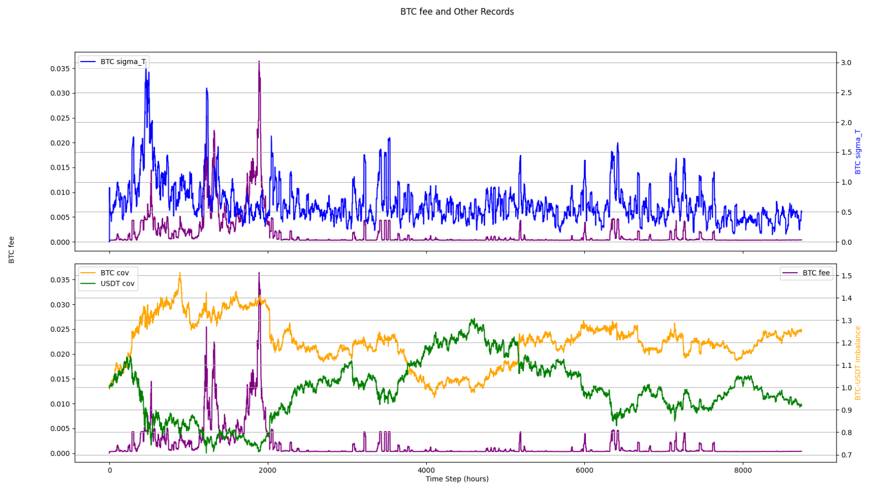
We compare the impermanent loss between Uniswap v2 and our volatile pool design. The rationale for comparing Uniswap v2 instead of Curve Tri-Crypto is that the bonding curve of Curve Tri-Crypto is fundamentally close to the bonding curve of v2 when their amplification factor is significantly large. Therefore, we can compare through the tractable formula of Uniswap v2 of impermanent loss and provide a value-adding insight for our users.



We observe that the AMM algorithm shares a similar pattern of impermanent loss with Uniswap, and the deviation between them is attributed to the pegging mechanism. Although it is less effective without a dynamic fee, some price compositions may allow r*¿1 and trigger re-pegging even without a fee.

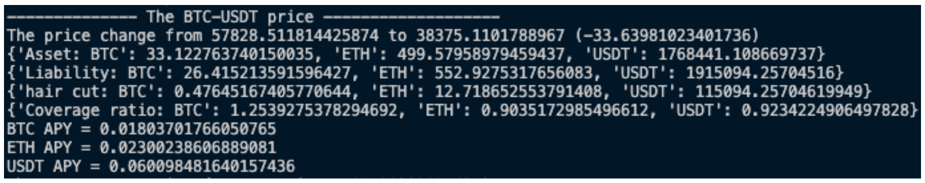
As we observe from the historical data, the re-pegging mechanism could effectively reduce impermanent loss, which provides a relatively lower impermanent loss for some periods. Our dynamic fee mechanism, which pragmatically facilitates re-pegging, will effectively reduce impermanent loss better compared to the standard benchmark of Uniswap v2 under asset equals to liability system.

5.3 Dynamic Fee and Real APY



Our dynamic fee design can be configured with different market circumstances to maximize our LP's benefit under highly volatile situations while remaining competitive under normal circumstances. For our simulation, the volatility, measured as in standard deviation of change in price, ranges between 0; 3.0. By setting $Fee_{base} = 0.0002$, $Fee_{\{\sigma,U\}} = 0.002$, $Fee_{imbalance} = 0.000125$. We show that our dynamic fee can incorporate circumstances with significant price fluctuation.

In our sample period, when bitcoin drops from \$57727 to \$29154 (-49.49%), and the coverage ratio of USDT drops to around 0.7067, the impermanent loss is highest, and we are charging around 3.64% swap fee to protect our LP. In contrast, when the percentage change in BTC price is within 20% and with different volatility, the dynamic fee varies between 0.035% and 0.115% [See Appendix D].



```
-------------- The BTC-USDT price -------------------
The price change from 57828.511814425874 to 38375.1101788967 (-33.63981023401736)
{'Asset: BTC': 33.122763740150035, 'ETH': 499.57958979459437, 'USDT': 1768441.108669737}
{'Liability: BTC': 26.415213591596427, 'ETH': 552.9275317656083, 'USDT': 1915094.25704516}
{'hair cut: BTC': 0.47645167405770644, 'ETH': 12.718652553791408, 'USDT': 115094.25704619949}
{'Coverage ratio: BTC': 1.2539275378294692, 'ETH': 0.9035172985496612, 'USDT': 0.9234224906497828}
BTC APY = 0.01803701766050765
ETH APY = 0.02300238606889081
USDT APY = 0.060098481640157436
```

If our wombat volatile pool were introduced during the sample period, we would pragmatically generate at least 1.8%, 2.3%, and 6% APY for BTC, ETH, and USDT, respectively. This is believed to be an underestimated APY because our simulation is conducted under hourly price fluctuation, which significantly lowers the price volatility and arbitrage opportunity compared to minutes- or seconds-based trading.

Reference

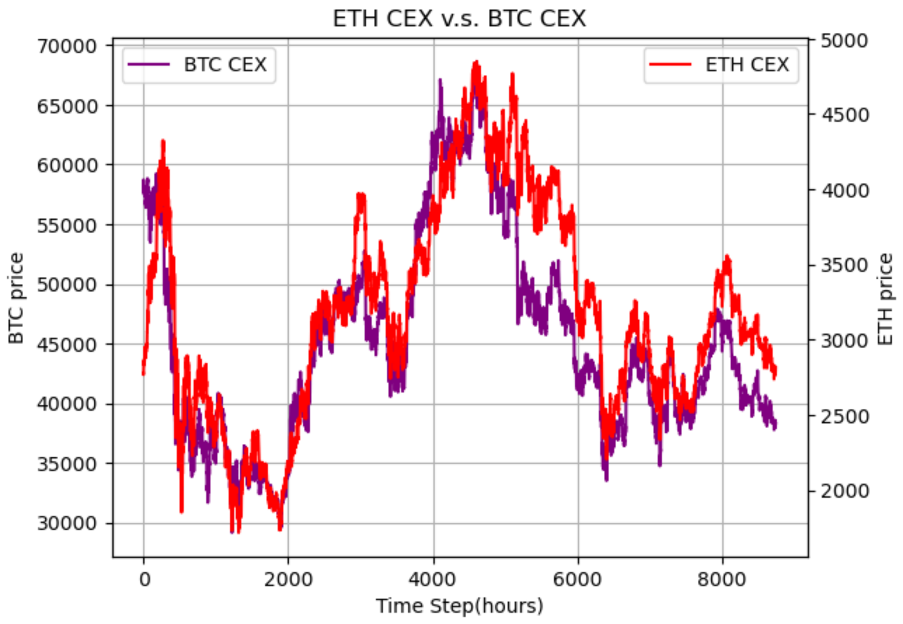
Milionis J., Moallemi C. C., Roughgarden T. and Zhang A. L. (2023). Automated Market Making and Loss-Versus-Rebalancing. arXiv:2208.06046.

Wombat Team. (2022). Wombat - An Efficient Stableswap Algorithm.

Wombat Team. (2022). Wombat Side Pool and Dynamic Pool.

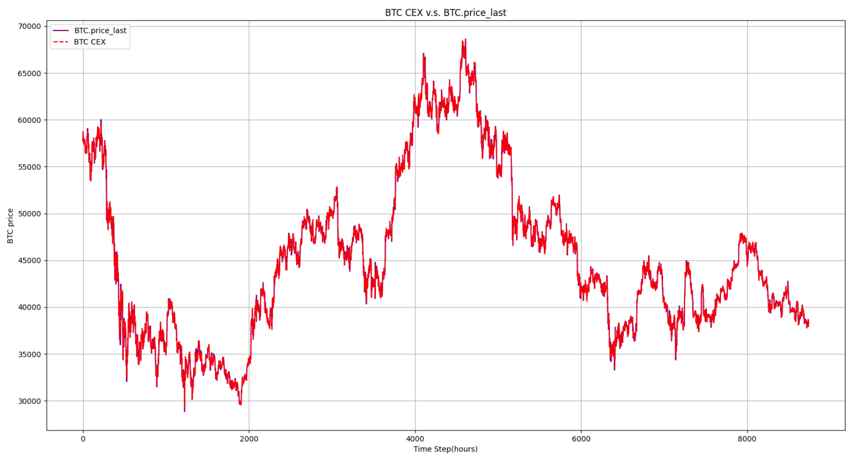
Appendix

A Cex Price of the Sample Period



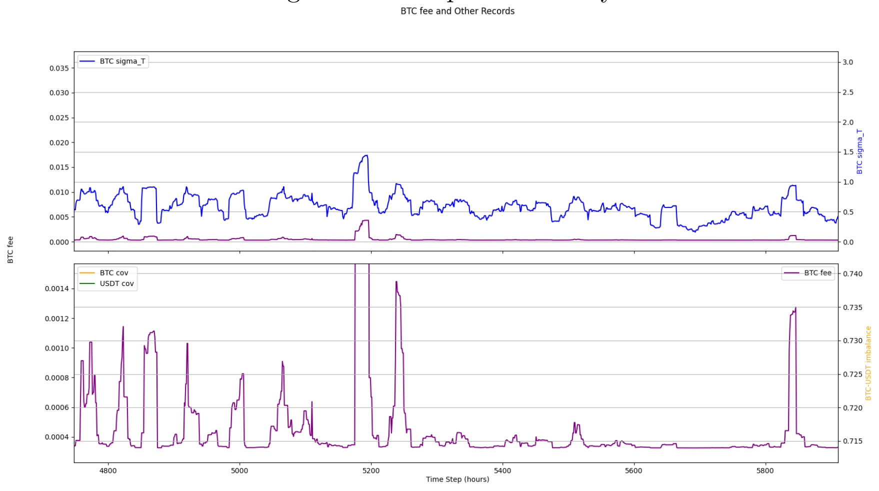
B Arbitrage Bot Behaviour

Appendix B shows that the arbitrage bot we designed has closely adjusted the price of the BTC within our volatile pool.



C Fee Impact through Volatility

The purple line represents the dynamic fee charged, which corresponds to different factors such as coverage ratios and price volatility.


BTC fee and Other Records
BTC sigma_T
BTC cov
USDT cov
BTC fee
BTC fee
BTC-USDT imbalance
Time Step (hours)