Eliminating impermanent loss by leveraged liquidity

Michael Egorov
(Dated: 2 March 2025)

Ever since the Automatic Market Makers (AMMs) were introduced in Decentralized Finance (DeFi), there was an issue of so-called "impermanent loss" (IL, or sometimes called LVR - loss vs rebalance), that is that AMMs often perform as a worse store of value than simply holding the components of liquidity idle. In this work, I propose a method to eliminate IL and make the position priced similarly to an individual component of liquidity while earning exchange fees. The simulations show, for example, that BTC/USD liquidity can make around 20% APR (average over 6 years) fundamentally while being priced similarly to BTC. Of course, the same method is applicable to other cryptocurrencies.

GENERAL IDEA

In Curve Cryptoswap AMM, price of liquidity (excluding earned trading fees) is approximately calculated similarly to that in classic $xy = k$ invariant $p_{LP} = \sqrt{p}$, where p is price of the token y (for example, BTC) in terms of token x (for example, USD). This is where impermanent loss comes from: for example $\sqrt{p} < 1/2 + p/2$ for all $p \neq 1$ means that holding an asset with initial price of 1 and equal amount of USD would outperform always-rebalanced liquidity if trading fee is set to zero.

Now, let's consider leverage L. If one borrows against any token with a price p' to buy even more of that token so that the value of the loan $d$is *always* kept to be equal to $d = V_c\,(1 - 1/L)$, where V_c is value of collateral, the position will be leveraged with the leverage L at all times, and price of the whole position p_* will be proportional to $(p')^L$.

Let's prove this formula. Small change in the price p_* of a token with price p' leveraged with the leverage L satisfied the relationship:

$$\frac{dp_*}{p_*} = L\frac{dp'}{p'}. \tag{1}$$

Integrating that gives:

$$\log p_* = L \log p' + \mathbf{const}. \tag{2}$$

Exponentiation of both sides gives:

$$p_* \propto (p')^L\,. \tag{3}$$

Therefore, if $L = 2$ and $p_{LP} = \sqrt{p}$, leveraging liquidity would give $p_* \propto \left(\sqrt{p}\right)^2 = p$, simply price of the token y, while the position makes exchange fees in addition (Fig. 1). While it sounds simple, it will not work with $xy = k$ invariant for the liquidity: losses on rebalancing to keep the leverage constant (or *releverage losses*) will simply be not lower than the earnings of the pool. Situation with Curve Cryptoswap, however, is much better, as will be shown in simulations.

Another curious property is that for $L = 2$, size of the loan is equal to half of the size of liquidity leveraged, on average. But USD part of that liquidity is also equal to half of its size in value. Therefore, if leverage is kept constant and equal to 2, liquidity will on aberage have enough USD to close the position, which is a very convenient property.

APR of the resulting position can be expressed as:

$$APR = 2r_{pool} - (r_{borrow} + r_{loss})\,. \tag{4}$$

So this method only works when the rate pool (multiplied by leverage) is significantly higher than the total of borrow rate and losses introduced by releverage of the position. Important to point out that this expression is only approximate, and a more precise APR is given by a combined simulation of cryptopool and releverage.

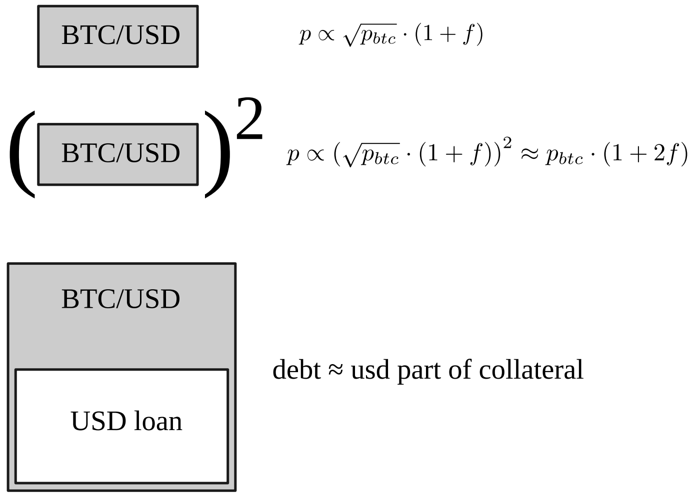
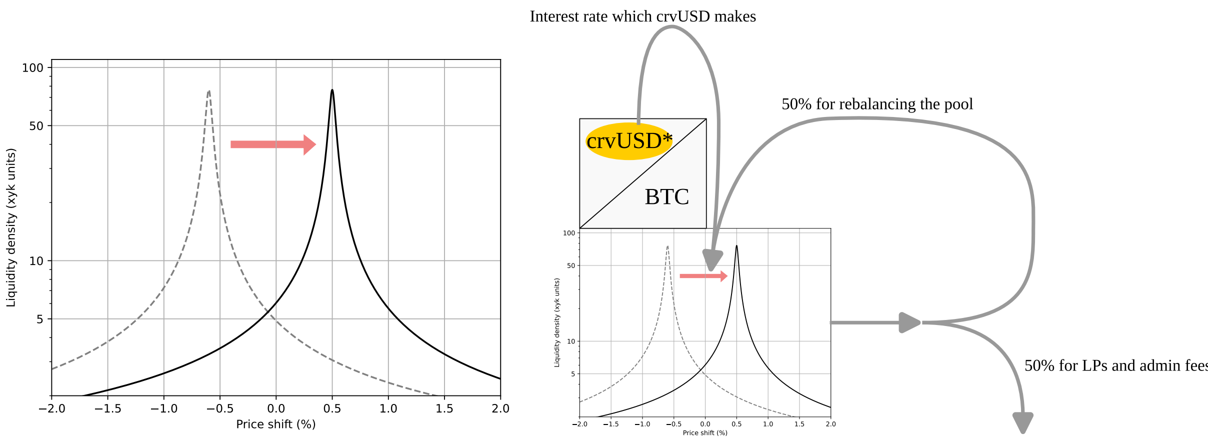


Figure 1: Schematic of leveraging liquidity to have no impermanent loss



(a) Movement of concentrated liquidity towards current prices

(b) Subsidy for moving concentrated liquidity

Figure 2: Automatic management of concentrated liquidity

USING CDP INTEREST RATE FOR REBALANCING

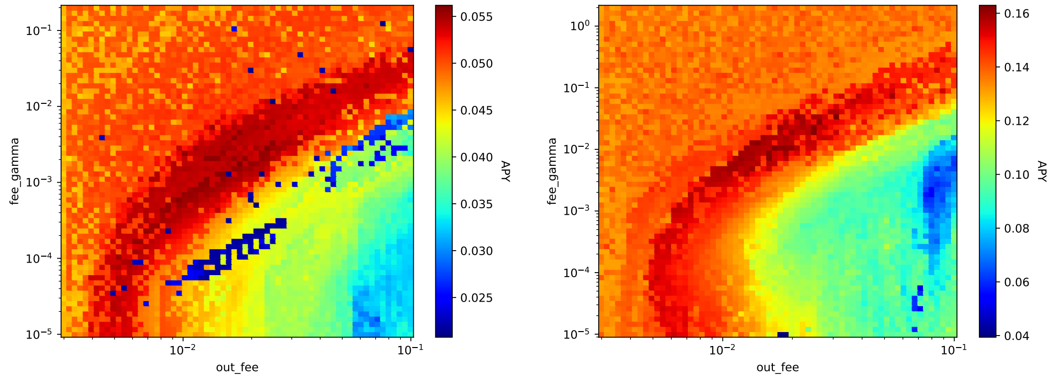
In Curve, all liquidity pools have concentrated liquidity. However, when price of the asset is not constant, concentrated liquidity is moved towards current prices automatically (Fig. 2a).

SIMULATIONS AND POOL OPTIMIZATION

RELEVERAGE ALGORITHM

Keeping leverage constant manually is not efficient, although can be done. Problem with manual approach is that the threshold price change at which it should happen is relatively high (10%) which makes variations in the returns very high (e.g. returns can go negative very often) (Fig. 4).

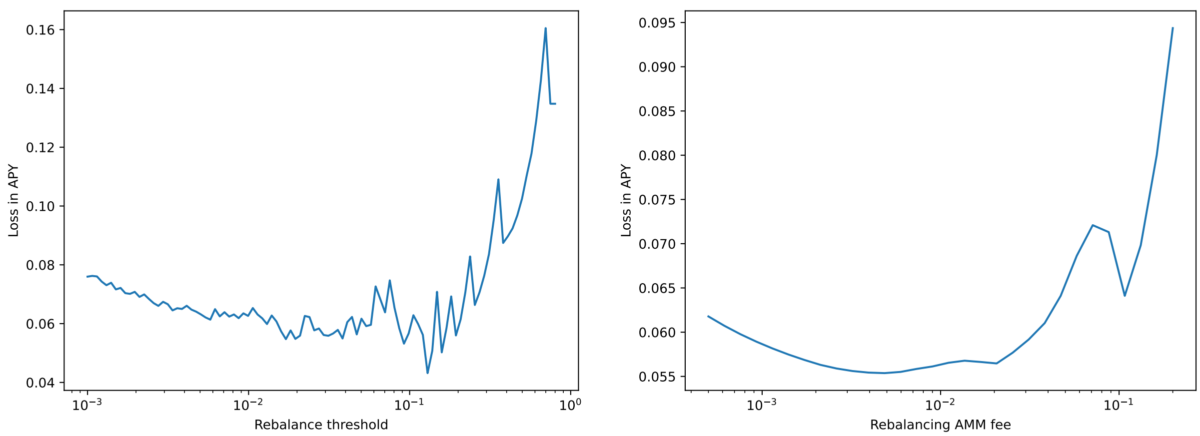
Instead, a special AMM is used for releveraging. The AMM uses an external oracle with price p_o for the asset which is being re-leveraged. The AMM keeps reserves of collateral y. Instead of keeping reserves of stablecoins, it borrows those having a debt d. When market price p is equal to p_o, in order to keep the leverage L, ideal debt should be equal to:



(a) Optimization of unboosted cryptoswap pool: $\left(f_{out}, \gamma_{fee}\right)$optimization

(b) Optimization of boosted cryptoswap pool: $\left(f_{out}, \gamma_{fee}\right)$optimization

Figure 3: Final optimization step for standard "unboosted" and boosted cryptoswap pools



(a) Losses from manual releverage depending on a price change threshold for triggering it

(b) Losses from releverage via an AMM depending on the fee of the AMM

Figure 4: Comparison of manual releverage and releverage via a special AMM

$$\widetilde{d} = \frac{L-1}{L} p_o \widetilde{y}, \tag{5}$$

where values with ~ mean that they are taken at the time when market price is equal to the oracle price. For example, one can see that for $L = 2$ (our case) $\widetilde{d} = p_o y/2$, which matches the intuition of keeping constant leverage.

We keep leverage constant via a variant of $xy = k$ AMM with x being represented as a function of oracle price and debt:

$$\left(x_0(p_o) - d\right) y = I\left(p_o\right), \tag{6}$$

where invariant I is constant at the same p_o, $x \equiv x_0\left(p_o\right) - d$.

In order to find $x_o\left(p_o\right)$ function, let's use the ideal values for $p = p_o$and property of $xy = k$ invariant: $p = x/y$. When we apply this to $p = p_o$:

$$\frac{x_0\left(p_o\right) - \widetilde{d}}{\widetilde{y}} = p_o, \tag{7}$$

and therefore, substituting Eq. 5:

$$x_0\left(p_o\right) = \frac{2L-1}{L} p_o \widetilde{y}. \tag{8}$$

As an example (which we will use later to choose the right solution), at $\widetilde{y} = 2$, $p_o = 1$, $L = 2$, $\widetilde{d} = 1$, we find $x_0 = 3$, and indeed, that satisfies $x_0 - \widetilde{d} = p_o \widetilde{y}$.

Now, let's find the function $x_0\left(p_o\right)$ for *any* current values of y and d (since y,d and p_o should fully define the state of the AMM). First, if we did know x_0 - we would be able to express "ideal" $\widetilde{y}$:

$$\widetilde{y} = \frac{L}{2L-1} \frac{x_0}{p_o}. \tag{9}$$

We also know that at constant p_o the value of invariant I is conserved and the same as at "ideal" parameters (e.g. $\widetilde{y}$, $\widetilde{d}$):

$$y\left(x_0 - d\right) = \widetilde{y}\left(x_0 - \frac{L-1}{L} p_o \widetilde{y}\right). \tag{10}$$

Here we take $x_0 \equiv x_0\left(p_o\right)$ for simplicity.

Now, when we substitute $\widetilde{y}$ expressed from x_0 in Eq. 9 into Eq. 10, we obtain a quadratic equation for x_0:

$$x_0^2 \left(\frac{L}{2L-1}\right)^2 - p_o y x_0 + p_o y d = 0 \tag{11}$$

Given the "simple" obvious solution mentioned previously in Eq. 8, we choose the larger root of the quadratic equation as the solution for x_0:

$$x_0\left(p_o\right) = \frac{p_o y + \sqrt{p_o^2 y^2 - 4 p_o y d \left(\frac{L}{2L-1}\right)^2}}{2\left(\frac{L}{2L-1}\right)^2}. \tag{12}$$

This expression defines everything necessary for the state of the AMM. Before any exchange, one should calculate x_0 for the current state, and it stays the same while we are on the same bonding curve and p_o is unchanged.

Now let's calculate value in the AMM. In order to reduce noise, it makes sense to base it on $p = p_o$ setting (in this case, value obtained on chain would not be susceptible to sandwich attacks, for example):

$$V = \widetilde{y} p_o - d = \frac{1}{L} \widetilde{y} p_o = \frac{x_0}{2L-1}, \tag{13}$$

value of invariant in such conditions is:

$$I = \left(x_0 - \widetilde{d}\right) \widetilde{y} = \frac{x_0^2}{p_o} \left(\frac{L}{2L-1}\right)^2, \tag{14}$$

so another way to express value in the pool V is:

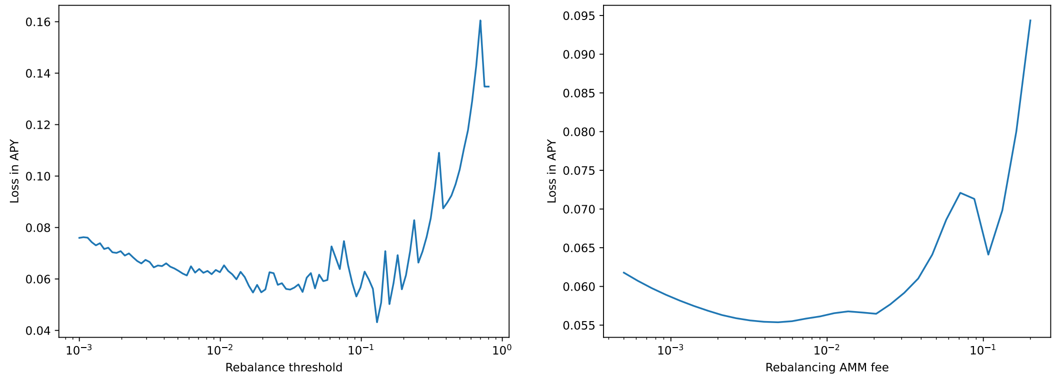
$$V = 2\sqrt{I p_o} - x_0. \tag{15}$$

We can use V when calculating shares when doing deposits and withdrawals.

From Eq. 14, we can clearly see that x_0 is proportional to $\sqrt{I}$ at a given p_o which appears useful when we work around deposits and withdrawals further.

DEPOSITS AND WITHDRAWALS

SPLITTING REVENUES WITH STAKED AND UNSTAKED LIQUIDITY



(a) Losses at "manual" releveraging vs price shift when it is triggered

(b) Automatic releveraging via AMM: dependence on the AMM fee

Figure 5: Losses introduced by releveraging