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.

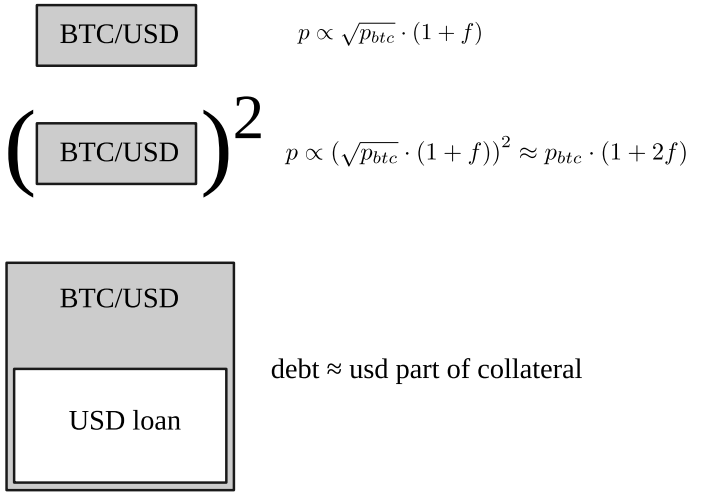
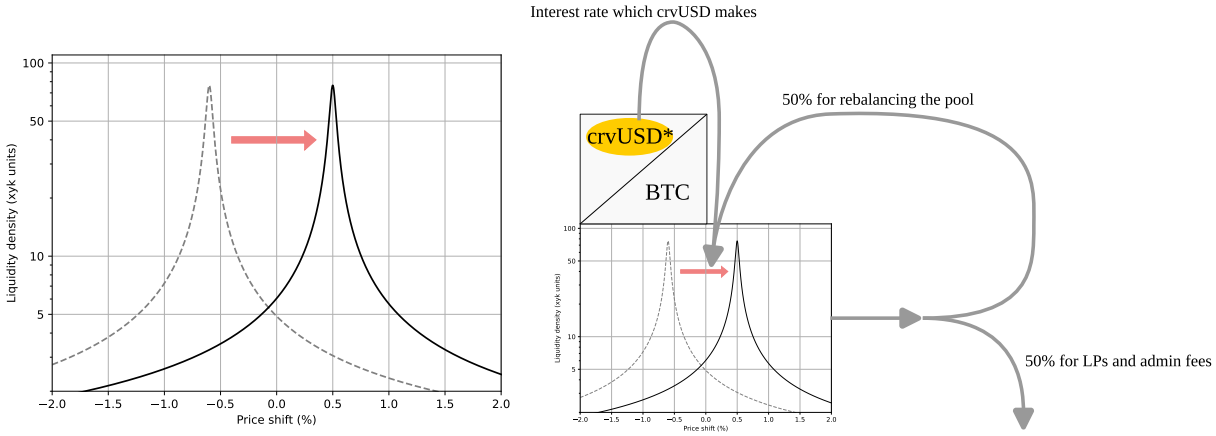


Figure 1: Schematic of leveraging liquidity to have no impermanent loss



(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. 7).

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:

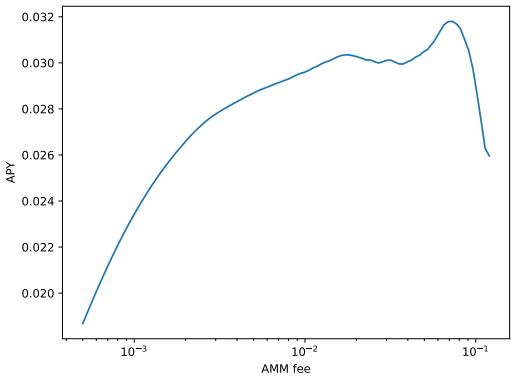
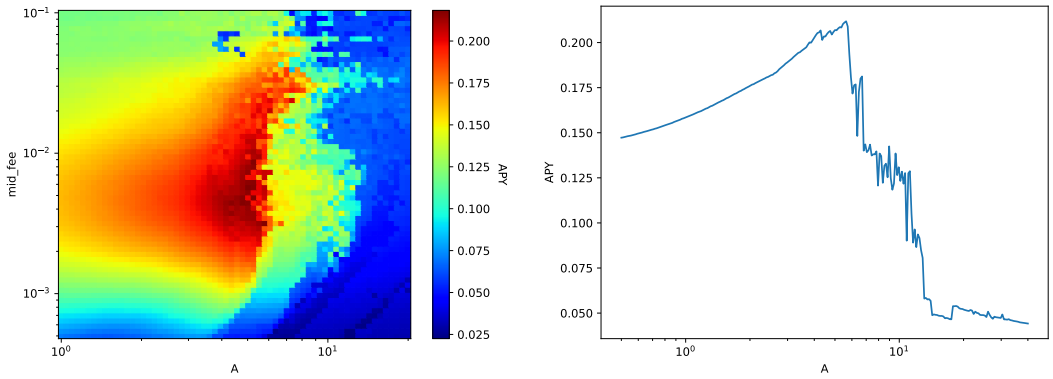


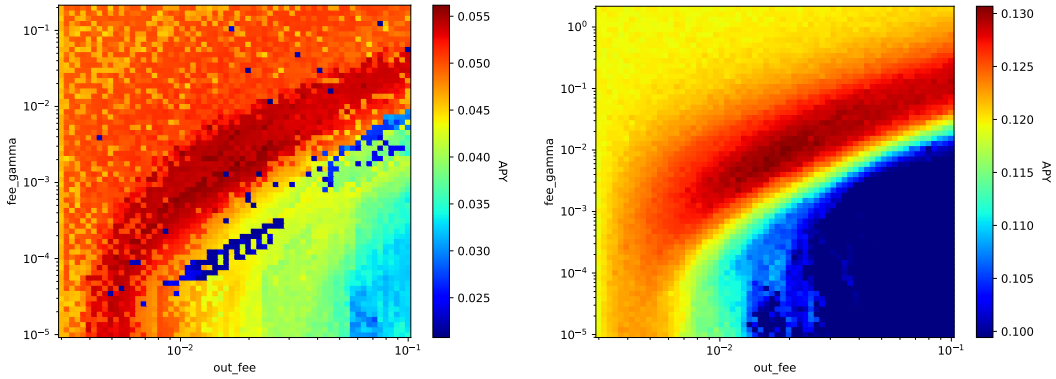
Figure 3: Reference study: optimizing $xy = k$ pool fee. Data over 2 years (Jan 2023 - Oct 2024) used



(a) Optimizing concentration parameter A and fee f_{mid} simultaneously

(b) Optimizing concentration parameter A at $f_{mid} = 0.003$ - slice of the 2D scan

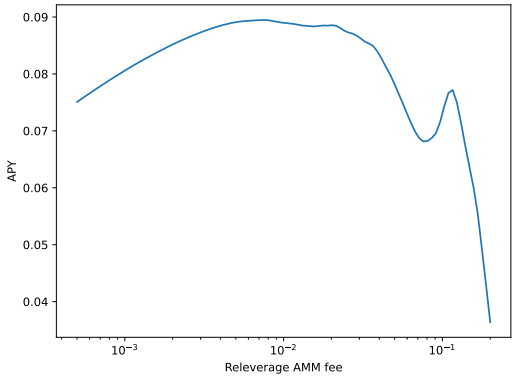
Figure 4: Initial optimization of cryptopool: no dynamic fee applied. Data over 6 years (Jan 2019 - Oct 2024) used



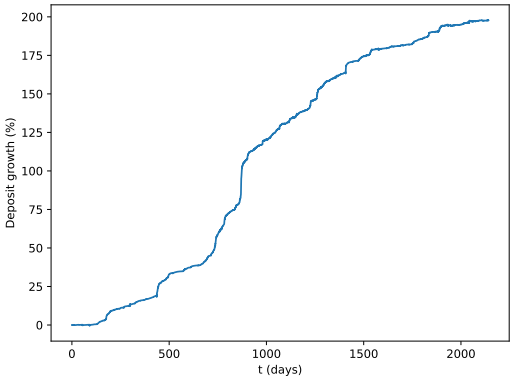
(a) Optimization of unboosted cryptoswap pool: $\left(f_{out}, \gamma_{fee}\right)$optimization

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

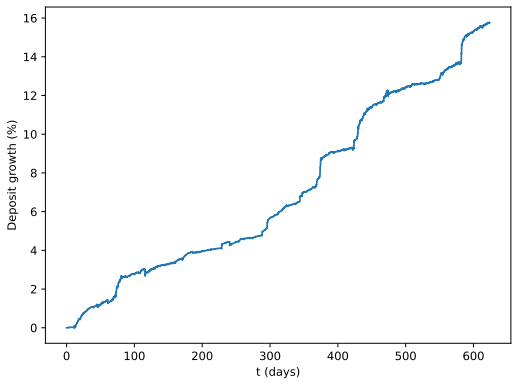
Figure 5: Final optimization step for standard "unboosted" and boosted cryptoswap pools. Data over 2 years (Jan 2023 - Oct 2024) used



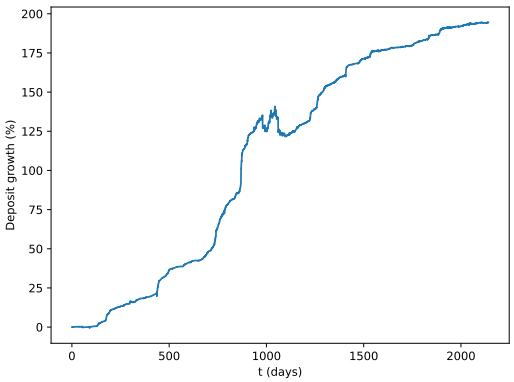
(a) Optimizing releverage AMM fee (optimal value: $f = 0.007$)



(b) Simulation from Jan 2019 till Oct 2024: average APR = 20%

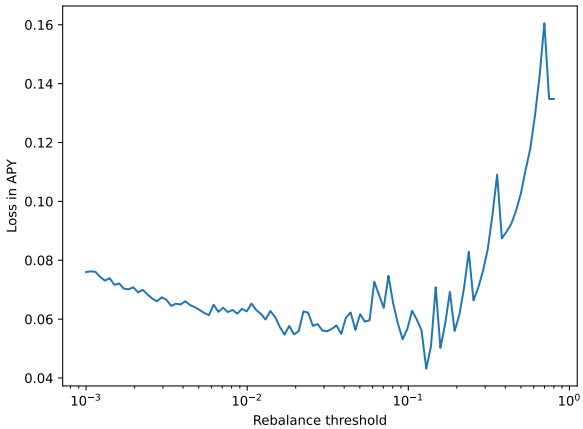


(c) Simulation from Jan 2023 till Oct 2024 (tail of Fig. b): average APR = 9%

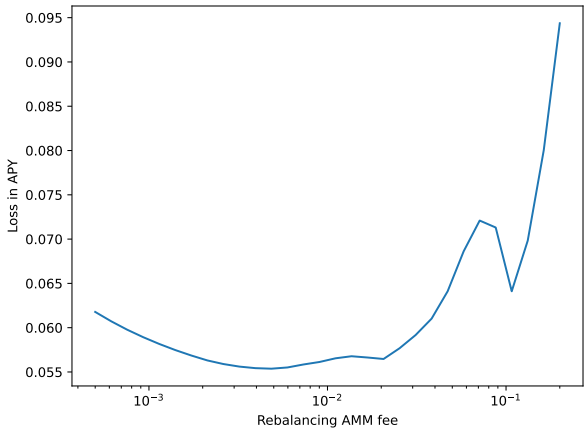


(d) Example of too aggressive parameters ($A = 4.8$ instead of $A = 4.13$). Partial loss is visible (in year 2021), however average APR is approximately the same

Figure 6: Simulation with releverage AMM removing impermanent loss. Data is obtained for BTC/USD price feed



(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 7: 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:

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

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

We are removing impermanent loss in curve Cryptoswap pool, and keeping $L = 2$ leverage of its liquidity (LP tokens) for that.

Deposits

When deposit is happening, we deposit first in Cryptoswap pool, and take a loan with d stablecoins against LP token created by this deposit. User brings just cryptocurrency (like BTC) with the amount c_{in}. So deposit works in the following sequence:

- User brings c_{in} of cryptocurrency and specifies the debt Δd (which has value equal to the value of cryptocurrency ideally, or can be calculated proportionally to balances in Cryptoswap);
- System takes a loan of Δd stablecoins and deposits $(\Delta d, c_{in})$ in Cryptoswap, yielding l LP tokens;
- Calculate value in Yield Basis AMM using Eq. 13 and x_0 calculated using Eq. 12 for the state before deposit (d, y) and after the deposit $(d + \Delta d, y + l)$;
- Calculate amount of YB tokens to mint for the depositor proportionally to the value increase from our deposit.

Witdrawals

When a user withdraws a fraction of liqudity, in order to not have any unfair disadvantage for liquidity providers, one cannot specify an arbitrary value of the debt. Instead, both collateral and debt are reduced by the same fraction equal to fraction of overall LP tokens being withdrawn. So withdrawal sequence inside the smart contract looks as the following:

- User brings t LP tokens to withdraw. If supply of LP tokens is s, liquidity gets reduced by s/t;
- Collateral gets reduced by $\delta c = cs/t$, and debt gets reduced by $\delta d = ds/t$;
- Since collateral is LP tokens of cryptopool, we withdraw δc LP tokens but in such a way that the amount of stablecoin withdrawn is exactly δd, and amount of cryptocurrency is kept varied as a function of δc and δd, given by `withdraw_fixed_out` method of the smart contract;
- Part of debt δd gets repaid from the withdrawn stablecoins, and cryptocurrency withdrawn from cryptoswap gets passed to the user.

SPLITTING REVENUES WITH STAKED AND UNSTAKED LIQUIDITY

In Yield Basis, real yields are only going to those users who did not opt in to earn YB tokens (e.g. stake). If they did not stake - they earn real yield from fees. If they did - they don't earn any yield from fees, but they earn governance tokens from emissions. The system also earns admin fee which is taken from the fees earned and given to YB token stakers (e.g. veYB).

The fee portion going to veYB dynamically depends on how much is staked. Let's denote:

- s - amount of LP tokens staked to earn YB tokens;
- T - total amount of LP tokens;
- $f_{\min}$- minimal value of admin fee;
- r - natural return rate (e.g. how much LPs would have been earning if no token system / staking / unstaking existed).

With those, admin fee will be:

$$f_a = 1 - (1 - f_{\min})\sqrt{1 - \frac{s}{T}}. \tag{16}$$

This formula has the following properties:

- if nothing is staked ($s/T = 0$), system admin fee is equal to $f_{\min}$ (for example, 10%), and the rest (90%) goes to LPs;
- if everything is staked ($s/T = 100\%$), system admin fee is equal to 100% because LPs are all earning YB tokens and nothing else.

Let's consider returns for the unstaked r_{us} part in different limits of how much is staked. If $s \to 0$:

$$f_a \to f_{\min,} \tag{17}$$
$$r_{us} \to (1 - f_{\min})\mathrm{r}. \tag{18}$$

And if $s \to T$:

$$f_a \to 100\%, \tag{19}$$
$$r_{us} \to \infty. \tag{20}$$

The infinite r_{us}while admin fee is going to 100% is deliberate and the reason why the admin fee formula is constructed this way.

From Eq. 13, total value of liquidity expressed in crypto is:

$$v = \frac{x_0}{(2L - 1)\, p_o}. \tag{21}$$

We define:

- v_{-1} - previous total value of liquidity;
- v_{+1} - new total value of liquidity before admin fee is taken;
- v_s - value assigned to staked liquidity (which earns no fees);
- v_{us} - value assigned to unstaked liquidity (the one which earns fees);
- v_{s*} - "ideal" value of staked liquidity (the it'd have if we have no losses);
- a - running value of collected admin fees.

Let's look at the situation when the value v is just changed due to trades (not deposits or withdrawals). Then running value of admin fees increases by:

$$\Delta a = (v_1 - v_{-1})\, f_a, \tag{22}$$

and value to be split between v_s and v_{us} is the rest:

$$\Delta v_{use} = (v_1 - v_{-1})\,(1 - f_a)\,. \tag{23}$$

After admin fee is taken, new total value of liquidity is:

$$v_1' = v_{-1} + \Delta v_{use}. \tag{24}$$

If we did not have any losses, staked liquidity value us equal to its upper limit: $v_s = v_{s*}$. If we, however, happened to experience a loss, e.g. $\Delta v_{use} < 0$, we share this loss between staked and unstaked liquidity:

$$\Delta v_s = \Delta v_{use} \frac{s}{T}, \qquad \Delta v_{use} \leq 0. \tag{25}$$

If we had profit but previously have loss, then $v_s < v_{s*}$, and we need to add value to v_s until it reaches v_{s*}:

$$\Delta v_s = \min\left(\Delta v_{use} \frac{s}{T}, \max\left(v_{s*} - v_s, 0\right)\right), \qquad \Delta v_{use} > 0. \tag{26}$$

The amount of unstaked LP tokens does not change if no deposits/withdrawals are happening, so one LP token represents a value-accruing cryptocurrency. Without deposits or withdrawals then $T - s = \text{const}$. The value of "staked" tokens, however, experiences negative rebases. Let's call reduction of number of staked LP tokens as δs. Then total and staked tokens after such a rebase accordingly change:

$$s' = s - \delta s, \tag{27}$$
$$T' = T - \delta s. \tag{28}$$

The ratio between these values should be the same as ratio between new staked and total values:

$$\frac{s - \delta s}{T - \delta s} = \frac{v_s'}{v_1'}. \tag{29}$$

From this:

$$\delta s = \frac{s v_1' - T v_s'}{v_1' - v_s'}. \tag{30}$$