Eliminating impermanent loss by leveraged liquidity

Michael Egorov
(Dated: 12 June 2025)

Ever since Automatic Market Makers (AMMs) were introduced in Decentralized Finance (DeFi), they have suffered from "impermanent loss" (IL, or sometimes called LVR - loss vs rebalance). As a result, AMMs often perform worse as store of value than simply holding the components of liquidity idle. In this work, I propose a method to eliminate IL and price the position similarly to an individual component of liquidity while still earning exchange fees. Simulations show, for example, that BTC/USD liquidity can earn around 20% APR (averaged over 6 years), with price exposure similar to BTC. The same method is also applicable to other cryptocurrencies.

GENERAL IDEA

In Curve Cryptoswap AMM, the price of liquidity (excluding earned trading fees) is approximately calculated similarly to that in classic $xy = k$ invariant, where $p_{LP} = \sqrt{p}$. Here, p is price of 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 $p \neq 1$. This means that, with trading fee set to zero, holding holding an asset with an initial price of 1 and an equal amount of USD would outperform always-rebalanced liquidity.

Now, consider *compounding* leverage, denoted by L. We define compounding leverage as the number determining the fraction of debt relative to collateral size. If one borrows against any token with a price p' to buy even more of that token, and keep the loan value *dalways* equal to $d = V_c\,(1 - 1/L)$, where V_c is value of collateral, the position will remain leveraged with compounding leverage L at all times. The price of the whole position p_* will be proportional to $(p')^L$.

Let's prove this formula. Suppose the value of collateral V_c changes by a small amount δV_c. We split compounding leverage in three steps: growth of collateral price, changing the debt to maintain the d/V_c ratio, and adjusting the collateral amount. We take the collateral amount during the first step and the collateral price during the second step as constant. At equilibrium, if the collateral amount is y and its price is p':

$$V_* = V_c - d = \frac{V_c}{L} = \frac{p'y}{L}. \tag{1}$$

Consider now the first step; in it, the value of leverage position V_* changes as follows:

$$V_* + \delta V_* = V_c - d + \delta V_c - \delta d \overset{d=\mathbf{const}}{=} (V_c + \delta V_c) - d = (V_c - d) + \delta V_c = V_* + \delta V_c = V_* + y\delta p', \tag{2}$$

$$\frac{\delta V_*}{V_*} = \frac{y\delta p'}{p'y/L} = L\frac{\delta p'}{p'} = L\frac{\delta V_c}{V_c}. \tag{3}$$

Where the latter equality holds because the amount of collateral is constant in the first step, and $V_c = yp'$.

Consider now the other way whereby the value of the position can be changed, namely, the change in the amount of collateral (step 3). When the collateral is changed with the price and the debt kept fixed, the value of leverage position V_* would change as:

$$V_* + \delta V_* = V_c - d + \delta V_c - \delta d \overset{d=const}{=} (V_c + \delta V_c) - d = (V_c - d) + \delta V_c = V_* + \delta V_c = V_* + p'\delta y \tag{4}$$

And, likewise,

$$\frac{\delta V_*}{V_*} = \frac{p'\delta y}{p'y/L} = L\frac{\delta y}{y} = L\frac{\delta V_c}{V_c} \tag{5}$$

We have, therefore, obtained that for any way of changing the value of the collateral, the change in the value of the leveraged position obeys the relationship:

$$\frac{\delta V_*}{V_*} = L\frac{\delta V_c}{V_c}. \tag{6}$$

We have so far ignored the second step; we now show that it will not lead to a degradation of value. As the debt is changed in such a way as to be constrained by a linear relationship with V_c, consider a combination of second and third steps taken together. By definition:

$$d = V_c\,(1 - 1/L) \tag{7}$$

Then the change in the value of the leveraged position will be given as:

$$V_* + \delta V_* = V_c - d + \delta V_c - \delta d = (V_c + \delta V_c) - d - (1 - 1/L)\,\delta V_c = (V_c - d) + \delta V_c - (1 - 1/L)\,\delta V_c = V_* + \frac{1}{L}\delta V_c = V_* + \frac{1}{L}p'\delta y \tag{8}$$

$$\frac{\delta V_*}{V_*} = \frac{\frac{1}{L}p'\delta y}{p'y/L} = \frac{\delta y}{y} = \frac{\delta V_c}{V_c} \tag{9}$$

It follows that in such a combined step, the value of the leveraged position behaves exactly like the value of the collateral, and, therefore, the total rate of the change in the value will be given by the preceding step. Intuitively, this result corresponds to the observation that this smooth change in the amount of collateral is purely begotten by and commensurate to the change in the debt. Should we have bundled the first and the second steps instead of the first and the third, the results will have been the same, as demonstrated by the above study in the relative rate of change of the leveraged position value.

It follows, then, that only one of the steps produces a nontrivial result for the rates of change of the collateral and position values, and for either step taken as such, the relationship is given by Eq. 6.

Integrating that gives:

$$\log V_* = L \log V_c + \textbf{const}. \tag{10}$$

Exponentiation of both sides gives:

$$V_* \propto (V_c)^L\,. \tag{11}$$

Therefore, if $L = 2$ and $p_{LP} = \sqrt{p}$, leveraging liquidity would give $V_* \propto \left(\sqrt{p}\right)^2 = p$, which is simply the price of token y, while the position earns exchange fees in addition (Fig. 1). While this idea appears simple, it does not work with the $xy = k$ invariant. The losses from rebalancing to maintain constant leverage (i.e. *releverage losses*) are not lower than the pool's earnings. The situation is significantly improved with Curve Cryptoswap, as demonstrated in the simulations.

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

APR of the resulting position can be expressed as:

$$APR = 2r_{pool} - (r_{borrow} + r_{loss})\,. \tag{12}$$

This method only works when the pool rate (multiplied by leverage) is significantly higher than the sum of borrow rate and losses introduced by releverage of the position. It's important to point out that this expression is only approximate, and a more precise APR can be obtained by simulating both the cryptopool and releverage.

USING CDP INTEREST RATE FOR REBALANCING

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

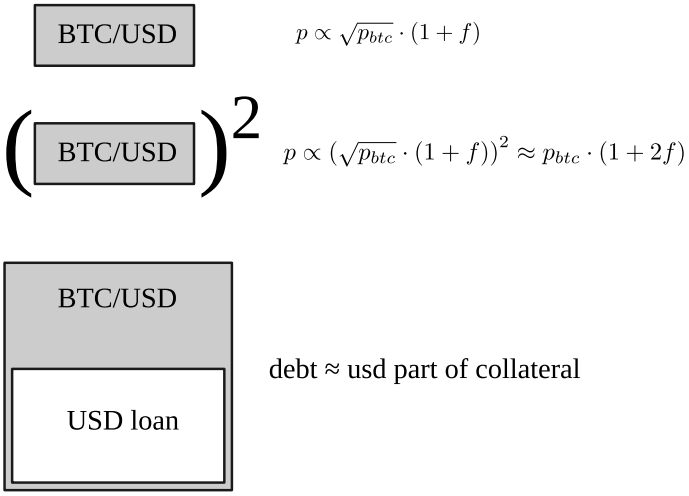
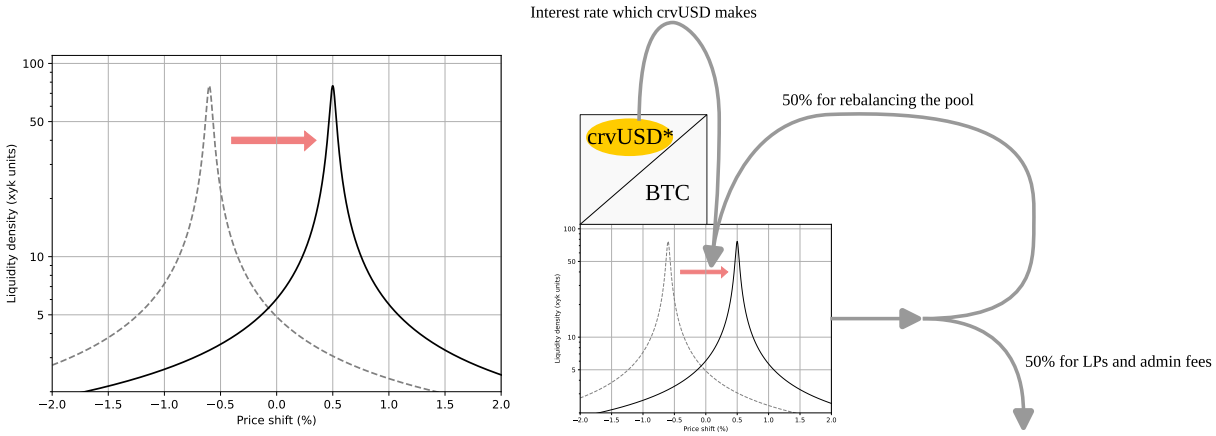


Figure 1: Schematic of leveraging liquidity to have no impermanent loss



(a) Movement of concentrated liquidity towards current prices

(b) Subsidy for moving concentrated liquidity

Figure 2: Automatic management of concentrated liquidity

SIMULATIONS AND POOL OPTIMIZATION

All simulations in this paper are performed by simulating arbitrage between the AMM and an external price feed (usually taken from Binance). In this setup, the AMM pool is arbitraged against an infinitely deep "heatsink" whose price changes between highs and lows of the candles of the feed's price candles. All AMM volume is assumed to come only from arbitrage between the AMM's current state and external prices. If the arbitrage profit is smaller than a certain threshold value (typically 0.03%), no trade occurs. Trading volume also stops once the arbitrage volume reaches half of the total exchange volume in the pair, although this has only minor impact in the simulations.

First, we simulate $xy = k$ bonding curve (e.g. like Uniswap2) as a sanity check of the method (Fig. 3). The only adjustable parameter in this AMM is its fee. There is a clear maximum in the dependency of APR on the AMM fee, but it is fair to say that over the simulated range (1 Jan 2023 - 1 Nov 2024) the APR of $xy = k$ AMM appears to be around 3%. This APR is in addition to what the value $xy = k$ bonding curve would yield if the AMM fee were 0 (e.g. on top of unremoved impermanent loss).

Simulation for cryptoswap is more complicated. I have found that replacing cryptoswap invariant with stableswap invariant, while keeping the algorithm of the cryptoswap unchanged, gives better results when a subsidy with crvUSD interest rate is used while having less adjustable parameters. In this simulation we set borrow rate to 10% meaning that -5% APR is spent on subsidizing liquidity rebalancing and continuously taken from LPs (while they earn fees). If we use stableswap invariant, parameter optimization seeks the maximum APR within a parameter space that

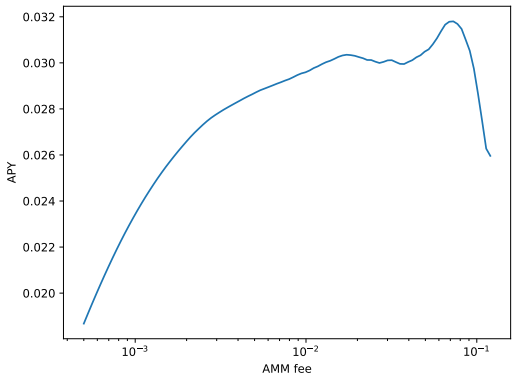
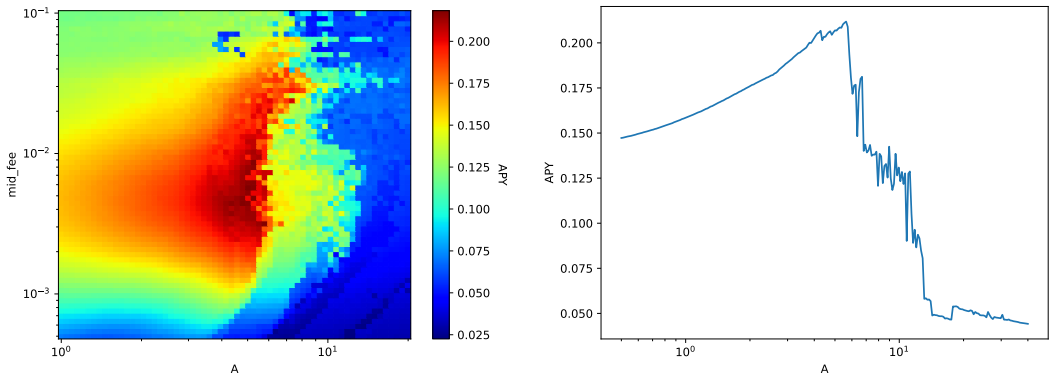


Figure 3: Reference study: optimizing $xy = k$ pool fee. Data over 2 years (Jan 2023 - Oct 2024) used



(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

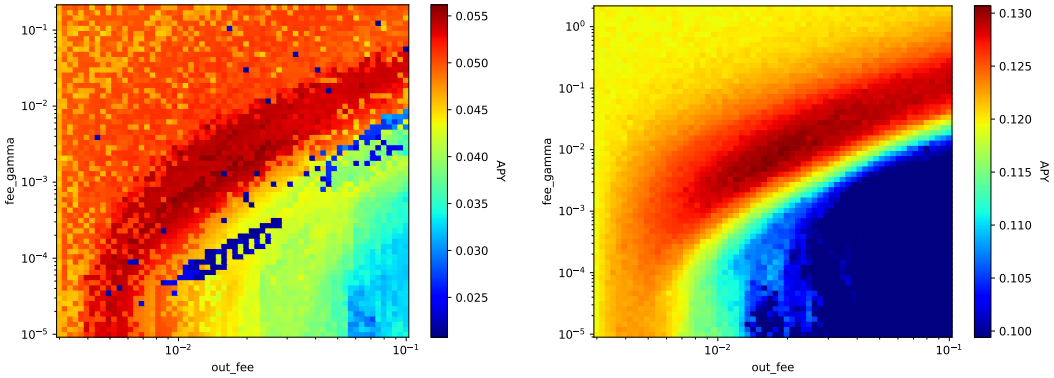
includes A (concentration parameter), f_{mid} (minimum AMM fee), f_{out} (maximum AMM fee) and γ_{fee} (fee steepness parameter). Ideally the borrow rate should also be optimized since we are in full control of it, but this was not done in these simulations (this will be addressed in future work).

Initially, we assume the fee to be always constant to approximately find the good range for A and f_{mid}. In this simulation, we set f_{out} to be equal to f_{mid}, which disables the dynamic fee. The results of such a scan are shown on Fig. 4. As concentrated liquidity becomes denser (e.g. A increases), the APR grows. However, around a certain threshold (around $A = 4$), the APR drops in a step-wise manner. At that point, volatility becomes too high for the rebalance algorithm to follow. When there's another place of high volatility on the price graph - we observe the second step forming. We should be choosing the highest value of A before the first discontinuity. It is also notable that the optimal f_{mid} appears to be 0.3% - similar to the typical Uniswap 2 fee.

Next, we optimize for dynamic fee (Fig. 5). It appears that the f_{mid} found previously with non-dynamic fees ($f_{mid} = 0.3\%$) also applies to dynamic fees, and the optimal A could be slightly higher than previously found. We first set A and f_{mid} at the parameters found in the non-dynamic scan, and then scan (f_{out}, γ_{fee}). Once we find optimal values - we can slightly vary A and f_{mid} to get the final adjustments and then rescan. Figure 5 shows the result. For comparison, I also present optimization results without the borrow rate donation feature.

While the releverage algorithm will be described om detail later, I reference the full simulation of releverage here (Fig. 6). The releverage is made via a special AMM which is also simulated as arbitrage.

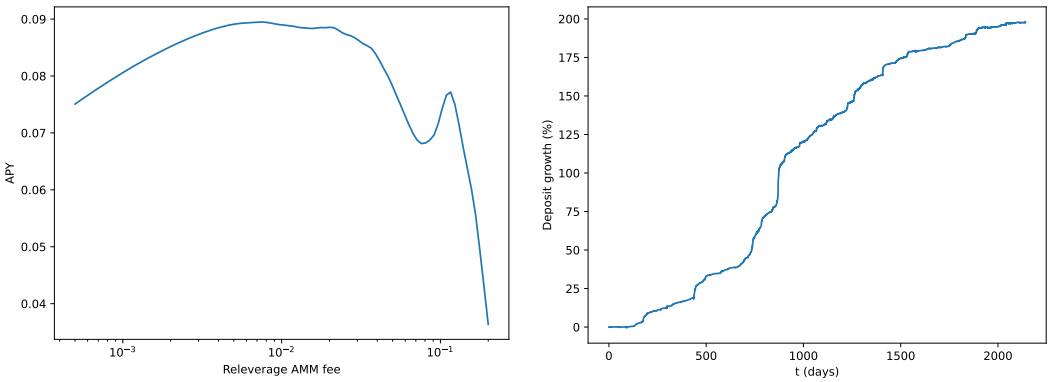
Noteworthy, simulation of releverage is separable from simulation of the base AMM (Fig. 7). It turns out that losses on releverage r_{loss} from Eq. 12 appears to be extremely close to twice the optimized returns of $xy = k$ pools (Fig. 3) results in the APR to be not higher than 0 for $xy = k$ bonding curve. This is worth further study. However, the APR in Curve cryptopools is consistently higher than $xy = k$ returns when the parameters are optimal, indicating that releverage should produce a positive APR. We confirm this with direct simulations of the releverage AMM.



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

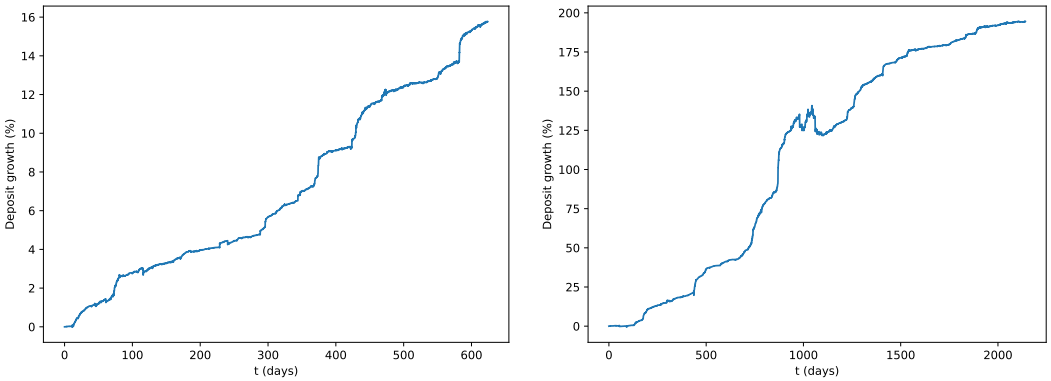
(b) Optimization of boosted cryptoswap pool: $\left(f_{out}, \gamma_{fee}\right)$optimization

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: $f = 0.007$)

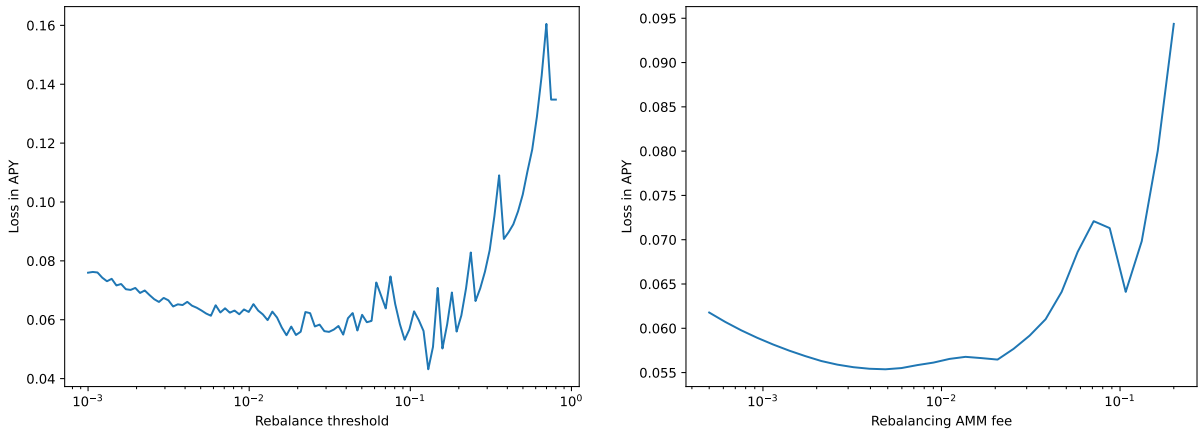
(b) Simulation from Jan 2019 till Oct 2024: average APR = 20%



(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

Simulation results from Fig. 6 show promising BTC/USD returns. The graphs display the value of the resultant LP position in BTC over time. Notable features of the simulation include:

- Dependency on the releverage AMM fee is very flat (Fig. 6a);
- Average APR over 6 years is around 20% (peaking over year 2021 to 60%) (Fig. 6b), however being just under 10% in more calm (and bearish) market conditions (Fig. 6c);
- If concentration A is set slightly higher than necessary - releverage algorithm experiences a loss at the point of the rebalance breaking down, however growth rate in calmer market conditions is higher, so the average APR over 6 years remains unchanded (Fig. 6d). This example shows how to estimate volatility risks in the product.

RELEVERAGE ALGORITHM

Keeping leverage constant manually is inefficient, though it can be done. The problem with the manual approach is that the threshold price change required to trigger adjustment is relatively high (10%) which leads to large fluctuations in returns (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 and holds reserves of collateral y. Rather than holding reserves of stablecoins, it borrows them, resulting in a debt d. To maintain leverage L, when the market price p is equal to p_o, the ideal debt should be:

$$\widetilde{d} = \frac{L-1}{L} p_o \widetilde{y}. \tag{13}$$

Here, values with a tilde ($\widetilde{}$) indicate 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 maintaining constant leverage.

We maintain constant leverage using a variant of $xy = k$ AMM with x defined as a function of oracle price and debt:

$$\left(x_0(p_o) - d\right) y = I\left(p_o\right), \tag{14}$$

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 $p = p_o$:

$$\frac{x_0\left(p_o\right)-\widetilde{d}}{\widetilde{y}}=p_o, \tag{15}$$

and therefore, substituting Eq. 13:

$$x_0\left(p_o\right)=\frac{2L-1}{L}p_o\widetilde{y}. \tag{16}$$

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" $\widetilde{y}$:

$$\widetilde{y}=\frac{L}{2L-1}\frac{x_0}{p_o}. \tag{17}$$

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

$$y\left(x_0-d\right)=\widetilde{y}\left(x_0-\frac{L-1}{L}p_o\widetilde{y}\right). \tag{18}$$

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. 17 into Eq. 18, we obtain a quadratic equation for x_0:

$$x_0^2\left(\frac{L}{2L-1}\right)^2-p_oyx_0+p_oyd=0 \tag{19}$$

Given the "simple" obvious solution mentioned previously in Eq. 16, we choose the larger root of the quadratic equation as the solution for x_0:

$$x_0\left(p_o\right)=\frac{p_oy+\sqrt{p_o^2y^2-4p_oyd\left(\frac{L}{2L-1}\right)^2}}{2\left(\frac{L}{2L-1}\right)^2}. \tag{20}$$

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):

$$V=\widetilde{y}p_o-d=\frac{1}{L}\widetilde{y}p_o=\frac{x_0}{2L-1}, \tag{21}$$

value of invariant in such conditions is:

$$I=\left(x_0-\widetilde{d}\right)\widetilde{y}=\frac{x_0^2}{p_o}\left(\frac{L}{2L-1}\right)^2, \tag{22}$$

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

$$V=2\sqrt{Ip_o}-x_0. \tag{23}$$

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

From Eq. 22, 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 depositing, funds are first provided to the Cryptoswap pool, creating an LP token. A loan of d stablecoins is then taken against this LP token as collateral. User brings in a single cryptocurrency (like BTC) in amount c_{in}. Deposit proceeds in the following sequence:

- User brings c_{in} of cryptocurrency and specifies the debt Δd (ideally equal to the deposited value, or set proportionally to Cryptoswap pool balances);
- 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. 21 and x_0 calculated using Eq. 20 for the state before deposit (d, y) and after the deposit $(d + \Delta d, y + l)$;
- Calculate amount of ybBTC tokens to mint for the depositor proportionally to the value increase from their deposit.

Withdrawals

When a user withdraws a fraction of liquidity, 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, the withdrawal sequence in the smart contract goes as follows:

- The user brings t LP tokens for withdraw. If total supply of LP tokens is s, liquidity is reduced by s/t;
- Collateral is reduced by $\delta c = cs/t$, and debt gets reduced by $\delta d = ds/t$;
- Since the collateral is cryptopool LP tokens, we withdraw δc LP tokens so that the withdrawn stablecoin amount is exactly δd, and the cryptocurrency amount 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 using the withdrawn stablecoins, and any cryptocurrency withdrawn from cryptoswap is sent to the user.

SPLITTING REVENUES BETWEEN STAKED AND UNSTAKED LIQUIDITY

In Yield Basis, real yields are paid only to users who do not opt in to earn YB tokens (e.g. stake). Users who do not stake earn real yield from trading fees. Users who stake do not earn yield from fees, but instead they receive governance tokens from emissions. The system also charges an admin fee which is taken from the fees earned and given to YB token stakers (e.g. veYB).

The portion of fees given to veYB depends dynamically on the amount 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 these definitions, the admin fee is:

$$f_a = 1 - (1 - f_{\min}) \sqrt{1 - \frac{s}{T}}. \tag{24}$$

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 $s \to 0$:

$$f_a \to f_{\min,} \tag{25}$$
$$r_{us} \to (1 - f_{\min})\mathrm{r}. \tag{26}$$

And if $s \to T$:

$$f_a \to 100\%, \tag{27}$$
$$r_{us} \to \infty. \tag{28}$$

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. 21, total value of liquidity expressed in crypto is:

$$v = \frac{x_0}{(2L - 1)\, p_o}. \tag{29}$$

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:

$$\Delta a = (v_1 - v_{-1})\, f_a, \tag{30}$$

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

$$\Delta v_{use} = (v_1 - v_{-1})\,(1 - f_a)\,. \tag{31}$$

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

$$v_1' = v_{-1} + \Delta v_{use}. \tag{32}$$

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

$$\Delta v_s = \Delta v_{use} \frac{s}{T}, \qquad \Delta v_{use} \leq 0. \tag{33}$$

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*}:

$$\Delta v_s = \min\left(\Delta v_{use} \frac{s}{T}, \max\left(v_{s*} - v_s, 0\right)\right), \qquad \Delta v_{use} > 0. \tag{34}$$

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:

$$s' = s - \delta s, \tag{35}$$
$$T' = T - \delta s. \tag{36}$$

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

$$\frac{s - \delta s}{T - \delta s} = \frac{v'_s}{v'_1}. \tag{37}$$

From this:

$$\delta s = \frac{s v'_1 - T v'_s}{v'_1 - v'_s}. \tag{38}$$

ARBITRAGING LEVERAGE AMM AND VIRTUAL POOL

When we leverage liquidity, the AMM exchanges stablecoins for the LP token of cryptopool (or in the opposite direction). However, an arbitrage trader wants to exchange between real cryptocurrency and stablecoin, not LP tokens. Therefore, we introduce a "virtual pool" smart contract which combines leverage AMM, cryptopool and stablecoin (crvUSD) flash loans to facilitate this process.

Exchange from cryptoasset to stablecoin in virtual pool

The exchange consists of the following steps:

- Use a flash loan to borrow exact amount of crvUSD needed to add both cryptoasset and crvUSD as balanced liquidity in the cryptopool. This action does not change the spot price in cryptopool;
- Exchange the LP token for crvUSD in the leverage AMM;
- Repay the flash loan from the crvUSD obtained in leverage AMM;
- Return any remaining crvUSD to the user.

Exchange from stablecoin to cryptoasset in virtual pool

Exchange in this direction is much more elaborate:

- Flash-borrow an additional amount of crvUSD (the exact amount will be calculated below);

- Exchange all the crvUSD acquired (provided by user + flash loan) to cryptopool LP in the leverage AMM;
- Remove liquidity from the cryptopool symmetrically (e.g. without changing crypto-to-stablecoin ratio and price);
- The result should be the exact amount of stablecoins needed to repay the flash loan. Therefore, it is essential to calculate this amount precisely before executing.

If leverage AMM fee is f, debt is d, collateral amount in AMM is y, and x_0 is known, whole the cryptopool has x_c stablecoins, y_c amount of crypto, and total LP supply of S_c, then the required flash amount φ can be determined from the input amount of stablecoins x by solving a quadratic equation:

$$\varphi^2 + \varphi\left(x_0 - d + x - \frac{x_c}{S_c}(1-f)\,y\right) - yx\frac{x_c}{S_c}(1-f) = 0. \tag{39}$$

If we denote $r = \frac{x_c}{S_c}(1-f)$, we obtain:

$$D = b^2 + 4yxr, \qquad b = x_0 - d + x - ry, \tag{40}$$

$$\varphi = \frac{1}{2}\left(\sqrt{D} - b\right). \tag{41}$$

CONCLUSION

This work demonstrates a method to eliminate impermanent loss and provide automatically managed concentrated liquidity in a special automatic market-maker which is a combination of Curve cryptoswap AMM and a special releverage AMM. Simulations over a 6 year period show an average APR of 20 percent. The described method is currently being implemented in Yield Basis.

Ongoing work focuses on optimizing borrow rate, making AMM parameters dynamic in response to asset volatility, and ensuring simulated results remain independent on price data quality.