Conic Finance: An Introduction to Conic Omnipools

Abstract

Conic Finance introduces Conic Omnipools, which allocate liquidity in a single asset across multiple Curve pools, giving Conic liquidity providers (LPs) exposure to multiple Curve pools through a single LP token. All Curve LP tokens are automatically staked on Convex to earn CVX and CRV rewards. Additionally, Conic LPs receive CNC, the Conic DAO token. Liquidity in an Omnipool is allocated to Curve pools based on target allocation weights, which get updated regularly through a liquidity allocation vote held by the vote-locked CNC holders.

1 Introduction

Over the past two years, Curve Finance has evolved into one of the core pillars of DeFi. Not only has Curve become a protocol that generates billions of USD in daily trading volume with very low slippage, Curve has also introduced a novel and now commonly adopted mechanism of using a protocol's DAO token to vote on inflation weights that determine how DAO token emissions are allocated to different liquidity pools. The rise of Convex Finance has additionally enabled Curve LPs to benefit from a group boost on their received CRV emissions, while also introducing CVX rewards that are minted pro rata with CRV earned and allow CVX lockers to vote on CRV emissions. However, deciding on which Curve pool to allocate liquidity to is not trivial, as liquidity gauge inflation weights on Curve change weekly and the number of Curve pools constantly increases, especially since the introduction of factory pools. Furthermore, even when an asset is deposited into multiple Curve pools, it can become very costly to rebalance liquidity allocations in line with inflation weight changes.

Conic relies on Omnipools, which are liquidity pools for a single asset that allocate liquidity into multiple Curve pools, while Curve LP tokens get deposited on Convex and staked in the respective Convex pool rewards contracts. This allows an LP to gain exposure to multiple Curve pools through a single LP token and earn CRV and CVX. Furthermore, Conic Omnipools assign a *liquidity allocation weight* to each Curve pool, which determines the share of an Omnipool's liquidity that a Curve pool receives. In addition to CRV and CVX, Conic LPs receive CNC, the Conic Finance DAO token, which can be locked for vlCNC to participate in protocol governance. Every two weeks, the liquidity allocation weights of every Omnipool get updated via a DAO vote in which vlCNC holders may participate. While Curve and Convex provide mechanisms in which CRV and CVX can be used to vote on gauge inflation weights to modify CRV emissions across Curve pools, Conic provides a mechanism for updating weights that directly shift liquidity of one or more assets across Curve pools. While single sided liquidity provision is challenging when it comes to managing slippage, gas costs and flash loan attack risks, Conic employs a combination of oracles, LP token pricing mechanisms and rebalancing incentives to ensure that liquidity shifts are performed in a secure manner.

2 Goals

Conic aims to allow users to diversify their exposure across multiple Curve pools with a single asset. Specifically, enabling users to deploy their asset (e.g. ETH) to multiple Curve pools in a single transaction.

The main goals of Conic can be summarized by the following:

- **Omnipools**: Allow users to deposit a single asset into a Conic Omnipool which deploys liquidity across multiple Curve pools.
- **Liquidity distribution**: Offer a solution to the current liquidity provision in the Curve ecosystem which is dominated by incentivized liquidity. This is done by establishing Conic governance which allows vlCNC holders to directly control which Curve pools receive liquidity.
- **Governance**: Create a protocol that is both led and controlled by the Conic community via a self-sustaining DAO.
- **Ecosystem**: Conic aims to build a protocol that supports the growth and sustainability of the Curve ecosystem.

3 Omnipools

Omnipools are liquidity pools that Conic utilizes to allocate a single underlying asset across various Curve pools. The core mechanisms of Omnipools are: deposits and withdrawals, Curve LP token pricing and rebalancing liquidity across Curve pools.

3.1 Deposits and withdrawals

Each Omnipool has a single underlying asset which can be exchanged for the Omnipool LP Token (depositing), and the LP Tokens can be redeemed for the underlying (withdrawing) at any time.

One of the key challenges faced when depositing and withdrawing single sided liquidity is ensuring that the pool is protected from slippage incurred during these transactions. If not handled carefully, several attack vectors could exist that allow a malicious agent to drain funds from the pool. Slippage is unavoidable, and so Conic mitigates this risk by passing all slippage on to the depositing and withdrawing user during their transaction. This is done by using a new method to calculate Conic LP token minting and burning. When a user deposits into the pool, the current TVL of the pool is calculated first before any other steps. The pricing of the LP Tokens for this calculation uses the process outlined in 3.2 which is manipulation resistant. Then the deposit or withdrawal is processed. After this, the TVL is calculated again. During a deposit the amount of LP Tokens minted is based on the difference in the TVL from before and after the deposit (opposed to the amount of underlying deposited). Similarly, the amount of underlying redeemed depends on the change in TVL during a withdrawal. The impact of this is that any slippage incurred during deposits or withdrawals is passed on to the user. This ensures the assets in the pool are safe from any attack or ongoing cost of slippage. To protect users from excessive slippage, they can specify a slippage tolerance during the deposit or withdrawal process.

When depositing into an Omnipool, a calculation is done first to assess which Curve pool

is the most under-allocated relative to its target allocation. The deposit will then be made into that Curve pool. Similarly, when withdrawing from an Omnipool, the withdrawal will always be made from the most over-allocated Curve pool. The exception to this is when the deposit or withdrawal is so large, that it would push the balance of that Curve pool the other way, resulting in it being more imbalanced than before and over a specified imbalance threshold. In this case, a *full deposit* is made, where the deposit is split over several Curve pools such that the Omnipool is perfectly balanced after the deposit/withdrawal. This *full deposit* case should be quite rare and only affect extremely large deposits and withdrawals. This approach results in deposits and withdrawals being a lot more gas efficient, as most of the time, they only deposit into one Curve pool. Furthermore, regular deposits and withdrawals assist in maintaining the target balance of the Omnipool set by governance.

3.2 Curve LP token pricing

As any AMM, Curve employs pools which rely on the balance of different assets in the pool to determine a price. While the *stableswap* logic ensures comparatively stable prices, users can still imbalance a Curve pool (for instance using a flashloan) to influence the price in that pool. This gives rise to a number of attacks that must be prevented against when engaging in single-sided liquidity provision. Specifically, any transaction with a withdrawal or deposit of only one asset into a multi-asset pool on Curve is susceptible to such attacks. For instance, an attacker might imbalance a pool such that the price of the asset we want to deposit is lower than it would otherwise be. That way, we would receive comparatively few Curve LP tokens for our deposit, leading to an appreciation in the LP token value for the existing LPs of the Curve pool (including the attacker). To preclude this sort of attack from happening during the rebalancing of a Conic Omnipool, a manipulation resistant LP Token pricing methodology is needed. The LP token pricing mechanism proposed by Conic relies on the invariant of the Curve pool and price oracles to arrive at a manipulation resistant LP token value.

The LP token pricing mechanism proposed by Conic relies on the invariant of the Curve pool and price oracles to arrive at a manipulation resistant LP token value.

In the following we outline the proposed LP token pricing mechanism for a two-asset Curve pool. We start with a reshaped form of the Curve invariant equation,

$$y^2 * x + y * (x^2 + a * x) - b = 0\,, \tag{1}$$

where $a = \frac{D}{A*n^n} - D$, $b = \frac{D^{n+1}}{A*n^{2n}}$, and x and y are the two assets in the pool. Here, D is the curve invariant for the pool, which can be obtained on-chain, A is the amplification coefficient for the pool, and n is the number of assets in the curve pool. Note that we are initially limiting ourselves to two-asset Curve pools (see `https://curve.fi/files/stableswap-paper.pdf`). The broad idea is then as follows: To find an unbiased estimate of the amount of each asset in a pool, we can take the derivative of the invariant curve with respect to one of the assets, equate it to the price s (obtained from an oracle) and then solve it for the remaining asset. To solve this first derivative of the invariant curve, we use Newton's method. We start by solving Eq. 1 for y and only use the positive root

$$f(x) = \frac{-ax - x^2 + \sqrt{4bx + (ax + x^2)^2}}{2x}\,. \tag{2}$$

The first derivative of $f(x)$ is then

$$f'(x) = \frac{-2b + x(ax + x^2 - \sqrt{x(4b + x(a + x)^2)})}{2x\sqrt{x(4b + x(a + x)^2)}}\,. \tag{3}$$

Since we are interested in the the value of x at the current oracle price s, we subtract this from the negative of Eq. 3 and arrive at

$$f'_s(x) = -s - \frac{-2b + x(ax + x^2 - \sqrt{x(4b + x(a+x)^2)})}{2x\sqrt{x(4b + x(a+x)^2)}} = 0\,. \tag{4}$$

To apply Newton's method for solving Eq. 4, we also require the second derivative of $f(x)$:

$$f''(x) = -\frac{2b(3b + x(a^2 + 3ax + 3x^2))}{x(x(4b + x(a+x)^2))^{3/2}}\,. \tag{5}$$

We can then iteratively find a root of $f'_s(x)$ at value s, i.e., at the oracle price, by applying

$$x_{n+1} = x_n - \frac{f'_s(x)}{f''(x)}\,. \tag{6}$$

From this process, we obtain an unbiased estimate of the amount of the first asset present in the curve pool under consideration. Using this, it is straightforward to compute the amount of the other asset using the invariant D and the same method implemented in Curve pools, which is also resistant to manipulation.

3.3 Liquidity Rebalancing

Shifting liquidity between Curve pools is fundamental to the workings of Conic. However, single sided liquidity deposits and withdrawals are not feasible in the context of redistributing liquidity across multiple Curve pools. In order to restore and retain balance in terms of liquidity allocations being in line with their target, Conic relies on an incentivized and passive rebalancing system based on deposits to and withdrawals from the Omnipools. When a liquidity provider deposits into an Omnipool the deposit will always go to the least allocated Curve Pool. Similarly, when a liquidity provider withdraws funds from an Omnipool, the withdrawal will come from the most over-allocated Curve Pool. This means that with regular deposits and withdrawals the pool will maintain a balanced state. To incentivize regular deposits and withdrawals, CNC emissions will be distributed to liquidity providers who deposit into Omnipools while the pools are imbalanced. The CNC received will be based on the amount deposited, and will also increase over time while a pool is imbalanced, and will stop when the pool is balanced again.

There is a limiter in the CNC emissions, which ensures that the CNC rewards a user receives is based on how far they bring the pools in line with being rebalanced. For example, if the pools are currently 30% imbalanced, and the user brings them to being 20% imbalanced, then they would receive 33% of the remaining CNC rewards allocated for the rebalancing period. Once the pool is balanced, the rewards are set to zero and will remain at zero until the next rebalance period.

3.4 Platform fees

A Conic Omnipool may charge platform fees on CRV and CVX earnings. However, this will need to be enabled via protocol governance. All charged fees will be paid out to vlCNC holders (see Section 5).

4 Tokenomics

The CNC token is the governance token used by the Conic DAO and may be locked for vlCNC to participate in governance (see below). CNC is distributed as follows:

- **Liquidity providers**: Conic LPs receive a share of CNC that is minted proportional to the share of liquidity that they supply.
- **Incentivized deposits and withdrawals**: When the liquidity allocation weights in an Omnipool are adjusted, deposits and withdrawals that "rebalance" the Curve pools towards their target weights are incentivized with CNC.
- **Stakers of the CNC/ETH AMM LP token**: A total of 10% of the total CNC supply is distributed to stakers of the Curve factory pool CNC/ETH LP token to incentivize sufficient levels of liquidity for swaps.

5 Conic DAO

Holders of CNC may lock their tokens for vlCNC. Similar to other vote-locked tokens (e.g. vlCVX or veCRV), vlCNC is not transferable. CNC can be locked from a minimum duration of 4 months up to a maximum lock duration of 8 months. A user may relock all or one of their locks at any point in time.

5.1 vlCNC

Conic employs a set of standard boosts for CNC lockers, which will determine their final vlCNC balance. Additionally, there will be two temporary boosts that get airdropped to v1 lockers and veCRV holders at launch. The vlCNC balance determines how much voting power and fees a vlCNC holder receives. Note that not all boost factors apply to the vlCNC balance that is used for fee distribution (see below). All boost factors are multiplicative when calculating the total boost for a single lock.

An overview of the boosts is shown in Figure 1.

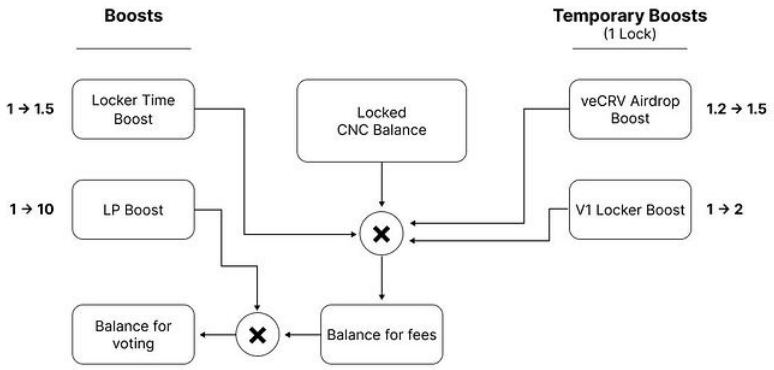

Figure 1: Conic boosts applied to the amount of CNC that gets locked.

5.1.1 Standard Boosts

The standard boosts are the boosts that are not airdropped to users.

These boosts are:

- **Lock time boost**: This is a boost that is proportionally linear to the duration of the lock. This boost remains constant throughout the duration of a single lock.
- **LP boost**: Conic LP tokens can be staked to receive a boost. Note that this boost only applies to the vlCNC balance in the context of voting (not fee earnings). The default, i.e., minimum, factor for LP boosts is 1. LPs can stake/unstake their LP tokens at any point in time. The LP boost factor will increase linearly with time (i.e. over 1 month), taking into account the LP token's USD value (as non-USD stablecoin Omnipools may exist). The LP boost for a Conic LP i is calculated as follows:

$$1 + 50 \times S_i \times W \tag{7}$$

  where S is the share of the total TVL that user i holds and W is a time weight that increases from $0 \longrightarrow 1$ over the period of 1 month to incentivize LPs to stake their LP tokens for longer periods and to protect against flash loan attacks. LP boosts are not constant and applied in real time to the total vlCNC balance, i.e., not to an individual lock. Meaning, as an LP's share of TVL and LP staking time changes, the updated boost factor will continuously be applied to their vlCNC balance.

Note that the vlCNC balance that gets used to calculate potential fee earnings is only dependent on the locker time boost.

5.1.2 Temporary Boosts

Apart from the standard boosts, there will be two temporary boosts that get airdropped once Conic gets launched. These temporary boosts can be used within the first 6 months after protocol launch. All temporary boosts can be applied once to a single lock (on creation of the lock) and will last for the entire duration of the lock.

The recipients of the temporary airdrop boosts are:

- **v1 Locker vlCNC holders**: Users that have locked CNC in the v1 locker will receive a one-time boost that can be applied to a single lock. The total amount of CNC that a user has locked prior to launch A is calculated as follows:

$$A = \sum_{t=v1LockerLaunch}^{t=now} (A_t \times D_t) + 4 \times \left(\sum_{t=now}^{t=v2LockerLaunch} A_t \times D_t \right) \tag{8}$$

  where A_t is the amount of CNC locked at time t and D_t is the duration of that lock at time t. The v1 locker boost B for a user is then calculated as:

$$B = \frac{\sqrt{A}}{\sqrt{A_{max}}} + 1 \tag{9}$$

  where A_{max} is the maximum amount of A that was locked out of all the addresses that have locked CNC prior to launch.

- **veCRV holders**: The snapshot of veCRV holders will be taken prior to launch and will be similar to the CNC airdrop that was allocated to vlCVX holders. Again, this boost will be applicable to a single lock.

Note that the boost factor may vary between multiple locks created by the same user.

In summary, the boost factors that exist and their respective minimum and maximum values are:

- v1 Locker Boost: 1.0 $\longrightarrow$ 2.0 **(temporary boost)**
- veCRV Airdrop Boost: 1.2 $\longrightarrow$ 1.5 **(temporary boost)**
- Lock Time Boost: 1.0 $\longrightarrow$ 1.5
- Conic LP Boost: 1.0 $\longrightarrow$ 10.0

5.2 Penalties

If a user does not interact with the locker after their lock has expired, the user's boost factor may not be accurate. Hence, there is a "kick" function, which may be called on users that have expired locks once the grace period of 4 weeks has ended. The caller would receive a share of the earnings that were generated since the lock has expired.

5.3 DAO Votes

Holders of vlCNC can participate in the Conic DAO and vote on proposals via Snapshot. This includes the bi-weekly liquidity allocation vote (LAV), whereby the liquidity allocation weights of each Omnipool (the respective Curve pools) are updated. Additionally, vlCNC holders may receive a share of the platform fees generated by Conic. Note that Conic platform fees are not enabled by default, but will need to be enabled via Conic governance. In addition to the bi-weekly LAV, vlCNC holders may vote on other proposals, such as protocol parameter updates. A critical vote will be the whitelisting of Curve pools. In order for a Curve pool to be eligible to be part of an Omnipool and receive a non-zero liquidity allocation weight, the Curve pool must be whitelisted via a DAO vote first.