Quadratic Concentrated Liquidity Pool (2-CLP): Technical Overview

Ariah Klages Mundt Steffen Schuldenzucker

Aug 2022

Gyroscope's 2-CLPs are AMMs that concentrate liquidity uniformly within a pricing range $[\alpha, \beta]$. 2-CLPs are implemented using a constant product curve with virtual reserves. In this implementation, virtual reserves include both the real reserves and 'reserve offsets' that allow the pool to achieve its pricing bounds. For instance, when the pool price is at one of the pricing bounds, one of the real reserves will be zero, but the virtual reserves will be nonzero due to the offsets. To specify a 2-CLP, we use the parameters in Table 1.

Parameter	Description
$\sqrt{\alpha} > 0$	Square root of the lower bound of the price range
$\sqrt{\beta} > \sqrt{\alpha}$	Square root of the upper bound of the price range

Table 1: Parameters for a 2-CLP

Calculating the Invariant L. Let x and y denote the reserves in the pool. The invariant is the constant product

$$L^2 = (x + a)(y + b)$$

where the virtual reserve offsets are a and b. The price bounds are achieved when

$$a = L/\sqrt{\beta}$$

$$b = L\sqrt{\alpha}$$

This makes the invariant equation into the following quadratic equation

$$\left(1 - \sqrt{\frac{\alpha}{\beta}}\right) L^2 - \left(\frac{y}{\sqrt{\beta}} + x\sqrt{\alpha}\right) L - xy = 0$$

which is solved by the quadratic formula.

Swap Execution. Given the invariant, and so also a and b, and the current reserves x and y, we can compute a swap of Δx to Δy by preserving the constant product where the final reserves are $x + \Delta x$ and $y - \Delta y \geq 0$. The swap calculations simplify to

$$\Delta y = \frac{(y + b)\Delta x}{x + a + \Delta x}$$

$$\Delta x = \frac{(x + a)\Delta y}{y + b - \Delta y}.$$

Note that we need $\Delta y \leq y$. When the swap is calculated based on Δy, this is obvious; when the swap is calculated based on Δx, the resulting Δy needs to satisfy $\Delta y \leq y$.

The case where Δy is swapped to Δx, is of course analogous.

Implementation. Table 2 lists the most important functions that implement the above calculations. They are all defined in the file `GyroTwoMath.sol`.

`Function`	Purpose
`calculateInvariant`	Computes the invariant L from current reserves (x, y).
`calcOutGivenIn`	Computes the amount that leaves the pool when a certain amount enters it, after fees.
`calcInGivenOut`	Computes the amount that needs to enter the pool when a certain amount should leave it, after fees.
`liquidityInvariantUpdate`	New invariant when liquidity is added/removed in a "balanced" fashion (without affecting the price). This avoids fully re-calculating the invariant.

Table 2: Most important functions