AstroZap

larry
Delphi Labs
larry@delphidigital.io

1 Derivation

Consider two assets A and B. An xy-k liquidity pool contains the two assets of amounts A_{pool} and B_{pool}, respectively, and a liquidity token with the total supply of L_{total}. A user wishes to enter the pool with A_{user} and B_{user}.

The amount of liquidity tokens that will be minted to the user is calculated as

$$L_{\mathrm{mint}} = L_{\mathrm{total}} \cdot \min\left(\frac{A_{\mathrm{user}}}{A_{\mathrm{pool}}}, \frac{B_{\mathrm{user}}}{B_{\mathrm{pool}}}\right) \tag{1}$$

Without lose of generality, let us assume $A_{\mathrm{user}}/A_{\mathrm{pool}} > B_{\mathrm{user}}/B_{\mathrm{pool}}$. In order to maximize L_{mint}, we need to swap some amounts of A into B. Assume we offer A_{offer} for the swap, then the pre-commission return amount is B_{return} is calculated by

$$A_{\mathrm{pool}} B_{\mathrm{pool}} = (A_{\mathrm{pool}} + A_{\mathrm{offer}})(B_{\mathrm{pool}} - B_{\mathrm{return}}) \tag{2}$$

Assume the commission rate is r $(0 < r < 1)$; after the swap, the pool has A'_{pool}, B'_{pool}, and the user has A'_{user}, B'_{user}. The relation of the these variables are described by

$$A'_{\mathrm{pool}} = A_{\mathrm{pool}} + A_{\mathrm{offer}} \tag{3}$$
$$B'_{\mathrm{pool}} = B_{\mathrm{pool}} - (1-r)B_{\mathrm{return}} \tag{4}$$
$$A'_{\mathrm{user}} = A_{\mathrm{user}} - A_{\mathrm{offer}} \tag{5}$$
$$B'_{\mathrm{user}} = B_{\mathrm{user}} + (1-r)B_{\mathrm{return}} \tag{6}$$

It is not hard to see that L_{mint} is maximized when

$$\frac{A'_{\mathrm{user}}}{A'_{\mathrm{pool}}} = \frac{B'_{\mathrm{user}}}{B'_{\mathrm{pool}}} \tag{7}$$

Combine (2) – (7), we arrive at the quadratic equation

$$aA_{\mathrm{offer}}^2 + bA_{\mathrm{offer}} + c = 0 \tag{8}$$

where

$$a = B_{\mathrm{pool}} + B_{\mathrm{user}} \tag{9}$$
$$b = 2A_{\mathrm{pool}}(B_{\mathrm{pool}} + B_{\mathrm{user}}) - rB_{\mathrm{pool}}(A_{\mathrm{pool}} + A_{\mathrm{user}}) \tag{10}$$
$$c = A_{\mathrm{pool}}(A_{\mathrm{pool}} B_{\mathrm{user}} - A_{\mathrm{user}} B_{\mathrm{pool}}) \tag{11}$$

In the Astrozap implementation, we use Newton's method to solve the above equation and find A_{offer}.

2 Example

Astroport LUNA-UST pool has 120,911,368,717,323 `uusd` and 1,410,005,459,618 `uluna`, and a commission rate of 0.3%. User wishes to enter asymmetrically with 100,000,000,000 `uusd`.

$$
\begin{aligned}
a &= 1410005459618 + 0 \\
&= 1410005459618 \\
b &= 2 \times 120911368717323 \times (1410005459618 + 0) - 0.003 \times 1410005459618 \times (120911368717323 + 100000000000) \\
&= 340459499970919150615924540 \\
c &= 120911368717323 \times (120911368717323 \times 0 - 100000000000 \times 1410005459618) \\
&= -17048569002131048373356261400000000000
\end{aligned}
$$

Using an online Newton's method calculator,[1] with 0 as the initial guess for A_{offer}, convergence was reached after only 5 iterations with the solution of A_{offer} = 50,064,794,338 `uusd`, slightly more than half of the user's deposit.

The return amount, before commission, is[2]

$$
\begin{aligned}
B_{\text{return}} &= \texttt{computeXykSwapOutput}(A_{\text{offer}}, A_{\text{pool}}, B_{\text{pool}}) \\
&= \texttt{computeXykSwapOutput}(50064794338, 120911368717323, 1410005459618) \\
&= 583587936 \ (\texttt{uluna})
\end{aligned}
$$

The user's and the pool's asset balances after the swap are

$$
\begin{aligned}
A'_{\text{pool}} &= 120911368717323 + 50064794338 \\
&= 120911952305259 \\
B'_{\text{pool}} &= 1410005459618 - (1 - 0.003) \times 583587936 \\
&= 1409423622445 \\
A'_{\text{user}} &= 100000000000 - 50064794338 \\
&= 49935205662 \\
B'_{\text{user}} &= 0 + (1 - 0.003) \times 583587936 \\
&= 581837173
\end{aligned}
$$

The user's shares in the pool:

$$
\begin{aligned}
\frac{A'_{\text{user}}}{A'_{\text{pool}}} &= 49935205662 \div 120911952305259 = 0.000412988 \\
\frac{B'_{\text{user}}}{B'_{\text{pool}}} &= 581837173 \div 1409423622445 = 0.000412819
\end{aligned}
$$

Which are indeed very close (only 0.04% difference.)

[1] `https://keisan.casio.com/exec/system/1244946907`

[2] The function `computeXykSwapOutput` is implemented in TypeScript at `https://github.com/mars-protocol/fields-of-mars/blob/master/scripts/cfmm.ts#L10`