Senswap Protocol

Thanh-Phuong Nguyen - Tu Phan - Nguyen Anh Tuan
{phuongnguyen, tuphan, tuannguyen}@descartes.network

Senswap

1 Introduction

The core of Decentralized Finance (DeFi) is Automated Market Maker (AMM) which allows financial transactions executed quickly based on pricing assets instantly. Constant product function currently is the most popular AMM that is applied to smart contracts like Uniswap [1], Bancor [2], or Balancer [3]. In this article, we would like to introduce a new AMM which was proposed by Yongge Wang [4], called constant ellipse/circle function (CEF). The general form of CEF - cost function is defined as the following

$$C(q) = \sum_{i=1}^{n} (q_i - a)^2 + b \sum_{i \neq j} q_i q_j = k,$$

where $q \in \mathbb{R}_+^n$; a and b are coefficients of the ellipse, and $-2 < b < 2$. The marginal price - the relative price between two assets is determined by taking ratio of the derivatives of the cost function with respect to each asset. It is written as the following

$$p_{i,j} = \frac{P_i(q)}{P_j(q)} = \frac{2(q_i - a) + b \sum_{k \neq i} q_k}{2(q_j - a) + b \sum_{k \neq j} q_k},$$

where $P_i(q)$ is the derivative of $C(q)$ with respect to q_i.

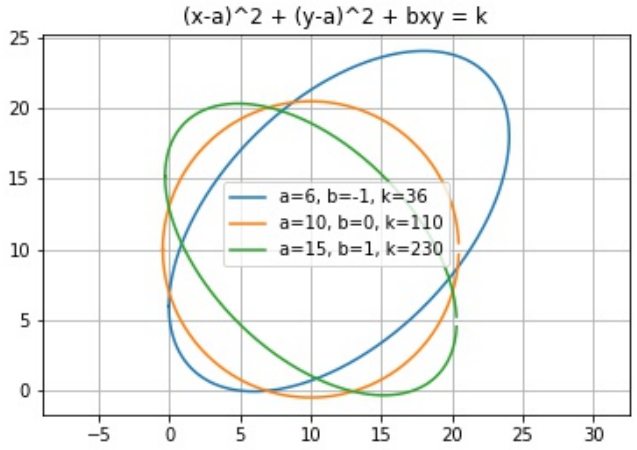

Fig. 1. Constant Ellipse function.

As the author mentioned in [4], the CEF has an advantage compared to the constant product function that it can prevent slippage rate problem which caused by the large trades (usually occurs

when the pool is still small). The CEF has bounds of relative price. In constant product function, the price amplitude is from 0 to ∞. The slippage rate problem cause the relative price go to ∞ which is infeasible for trading. Meanwhile, the CEF has the bound and the bound can adapt with liquidity of the pool automatically (we will discuss in details later).

In the subsequent sections, we will extend CEF into a protocol of our new smart contract, called SenSwap. The cost function is as the following

$$C(x, y) = (x - a)^2 + (y - a)^2 + bxy = a^2 + f,$$

where x and y are reserves of two assets in a pool; a, b and f are the coefficients of the AMM. Figure 1 shows how the coefficients change the bonding curve.

2 Proposed Protocol

In this section, we will introduce our protocol that describes how assets are traded and liquidity can be added to or withdrawn from liquidity pool. Moreover, our protocol also solves some disadvantages existing in the available DeFis like Uniswap.

2.1 Trading Process

Trading is a process that a trader wish to buy some amount of a token X, and he/she must pay a corresponding amount of token Y in return. In SenSwap, the price or the must-paid amount of token Y is automatically calculated by CEF. In a formal way, given a pool $q_1 = (X_1, Y_1)$ and a trader would like to buy an amount ΔX and pay ΔY.

$$q_1 = (X_1, Y_1) \longrightarrow q_2 = (X_1 - \Delta X, Y_1 + \Delta Y) = (X_2, Y_2).$$

Since the CEF remains constant during the trading process, then

$$C(q_1) = C(q_2)$$
$$(X_1 - a)^2 + (Y_1 - a)^2 + bX_1Y_1 = (X_2 - a)^2 + (Y_2 - a)^2 + bX_2Y_2$$

Solve the above equation, we obtain

$$Y_2 = \frac{(2a - bX_2) - \sqrt{(2a - bX_2)^2 - 4((X_2 - a)^2 - f)}}{2}.$$

And $\Delta Y = Y_2 - Y_1$.

Let's take an example. Given a pool $q = (X, Y) = (100, 1000)$ with a CEF $(a, b, f) = (1547.2245, 0, 10)$. If a trader buy 10 coins of token X then the pool will be $(90, 1027.215)$ after the trading. The marginal price of the trading is approximately 2.8023.

In the next section, we will learn how we define a CEF (a, b, f) when a liquidity provider add liquidity to the X/Y pool.

2.2 Liquidity Pool

In liquidity provision, there are two criterions that we want. First, when liquidity providers deposit an arbitrary amount of each token, the relative price stays the same. Second, the amplitude price expands as the liquidity of the pool. For the second criterion, the CEF control this by leveraging $f \geq 0$. The smaller f is, the larger price amplitude is. In this case, we keep f constant during the liquidity provision.

In order to calculate a and b such that the relative price remains its value, a and b are calculated as the following

$$a = (X + Y) + \frac{XY(1-p)}{pX - Y} + \sqrt{\frac{X^2Y^2(1-p)^2}{(pX-Y)^2} + f}$$
$$b = \frac{2}{pX - Y}(X - pY - (1-p)a)$$

where (X, Y) is the pool after depositing, p is the relative price. With this formula, a pool can be deposited only one side of the pool, this is called single exposure. We will discuss this in the next section.

After depositing, liquidity provider (LP) will receive a proportion that represents for his/her contribution to the pool. If the LP contributes ΔX to the X side of the pool, then the proportion is $S_X = \frac{\Delta X}{\Delta X + X}$, and $1 - S_X$ will be used to update proportions of contributions of the rest of LPs of the pool. With this proportion, a LP can withdraw part or all of his/her liquidity in the pool.

2.3 Single Exposure and Constraint of Liquidity Provision

As we mentioned, the flexibility of liquidity provision makes a corollary of single exposure. This is an unsolved problem of Uniswap. LPs have to deposit both sides of a liquidity pool with a ratio of amount of two tokens corresponding to the marginal price. This is to prevent any arbitrage caused by arbitraging price between pool price and external price. The problem is that LPs, who only have one token in their wallet, cannot make provision unless they trade for obtaining the other token. In that case, they have to give up the long position on the token.

Bancor V2 proposed a solution for this problem in article. Bancor V2 uses dynamic weights to incentivize LPs to equalize the current pool. Our solution is more internal. By changing parameters of CEF with respect to added liquidity, the pool can adapt with arbitrary amount liquidity provision but still remaining the marginal price as before depositing.

However, there is a constraint for deposition. The pool has a upper bound for depositing to the lesser side of the pool. Since if the amount of added liquidity to the lesser side is too much compared to the other side, the marginal price will change in the trend that creates arbitrage which is a potential risk for attackers to exploit.

For example, given a pool $(90, 1027.2147)$. If attackers would like to deposit only to the lesser side in order to create any arbitrage, then execute a trading transaction later to gain profit. The maximized amount they can deposit is just 40.806. Because the pool cannot adapt further to remain marginal price. On the other side, LP can add liquidity at any amount to the greater side of the pool.

The upper bound of deposition is calculate as the following

$$\Delta X < \frac{(p + \frac{1}{p})Y_2 - \sqrt{(1-p)^2 f + ((p + \frac{1}{p})^2 - 4)Y_2^2}}{2p} - X_1,$$

where X_1 is the reserve of the lesser side of the pool before deposition, and Y_2 is the reserve of the greater side of the pool after deposition. As we see, the upper bound depends on the amount of depositing to the greater side.

To sum up, SenSwap does not only inherits the advantages of convexity of constant-based function, but also leverages its flexibility to solve some drawbacks of Uniswap like slippage rate and single exposure problems. Our work on SenSwap is still in process.

References

1. Adams, et al., *Uniswap V2 Core*, white paper at homepage
2. Hertzog, et al., *Bancor Protocol*, white paper at homepage
3. Martinelli, et al., *A non-custodial portfolio manager, liquidity provider, and price sensor.*, white paper at homepage
4. Yongge Wang, *Automated Market Makers for Decentralized Finance (DeFi)*, arXiv, q-fin.TR