Descartes Network

SenSwap Whitepaper

Version 1

Authors
Tu Phan
Nguyen Anh Tuan
Thanh-Phuong Nguyen

December 26, 2020

Contents

Chapter 1

Abstract

Chapter 2

Introduction

Chapter 3

Related Works

Chapter 4

Problems of Uniswap

Uniswap is a fast and simple AMM. The product operator makes it suitable to run on Ethereum. However, there are some problems with this AMM. One of them is quadratic slippage rate. The slippage rate of Uniswap is shown in the following

$$\frac{R_B^{(t+1)}}{R_A^{(t+1)}} = \frac{1}{\alpha^2} \frac{R_B^{(t)}}{R_A^{(t)}},$$

where $R_A^{(t)}$ is reserve of token A at the t^{th} transaction, $\alpha = \frac{R_A^{(t+1)}}{R_A^{(t)}}$ is the changing rate of reserve of token A. When a transaction trading an amount nearly full reserve of a token, the price will change very fast. Such transactions are easy to occur when the pool is still small. Since impermanent loss is the decreasing value of liquidity provider's stake when the price changes compared to itself at the time depositing. High slippage rate leads to potentially high impermanent loss for liquidity providers. The relationship between slippage rate and impermanent loss is written as follow (see article)

$$IL = \frac{2\sqrt{SR}}{1 + SR} - 1,$$

where IL is impermanent loss, and SR is slippage rate. Without trading fee, IL is always less than or equal to zero. The equality occurs when the price is the same as it at the time of depositing. Despite of trading fee balancing LPs' returns, they still suffer the risk of loss if the market could not attract enough trades.

Another major problem of Uniswap is that liquidity providers have to deposit both side of a pool. This is a problem because if a liquidity provider, who possesses only one token, wishes to add liquidity, he/she have to trade in other markets to exchange the other token. This situation may prevent them to participate in Uniswap.

Uniswap is decentralized for liquidity provision on Ethereum. Therefore, there exists a gas fee for Uniswap to deploy on Ethereum. Recently, many complains that Uniswap is so expensive (see article). The reason for this is that Ethereum fee is getting higher. According to article, the price of an ETH to DAI transaction on Uniswap is \$55 compared to \$33 on Curve, \$44 on Aave, and over \$80 on Mooniswap. The gas fee problem is making Uniswap less attractive to both traders and liquidity providers.

Another drawback of Uniswap is that it is open for any new tokens. Listing new tokens without monitoring makes Uniswap easy to be scammed by fake tokens. It was reported that a fake Teller token and Uniswap pool had been created on August 19, 2020 (see article). Attackers could possibly create new tokens with similar name to real tokens to deceive users to trade worthless tokens.

Chapter 5

SenSwap - An Extension of Constant Ellipse Function

In this chapter, we will introduce a new AMM, called Constant Ellipse Function (CEF) [1], which is proposed by Yongge Wang. However, the author only gave the cost function and the marginal price formula. Therefore, in subsequent parts, we extend the CEF such that this AMM can possess properties that we desire.

5.1 Constant Ellipse Function

Constant Ellipse Function describes an ellipse. The general form of CEF - cost function is defined as the following

$$C(q) = \sum_{i=1}^{n} (q_i - a)^2 + b \sum_{i \neq j} q_i q_j = k, \tag{5.1}$$

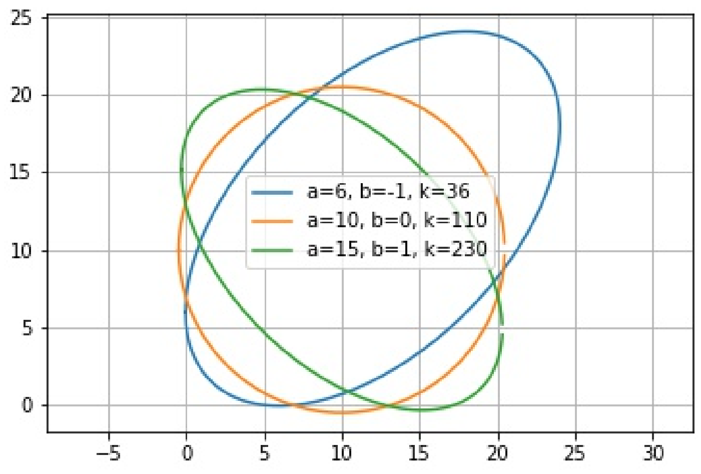

Figure 5.1: Geometrical representation of Constant Ellipse Function

Figure 5.1 shows geometric form of CEF. The blue ellipse has a form quite close to Uniswap. The green one has a form close to constant sum function. As we can see, CEF has a flexible bonding curve which can behave like some other AMMs.

However, we do not need the whole ellipse. In order to obtain the convexity property of an AMM, we only take the lower-left part of the ellipse as a bonding curve as Figure 5.2.

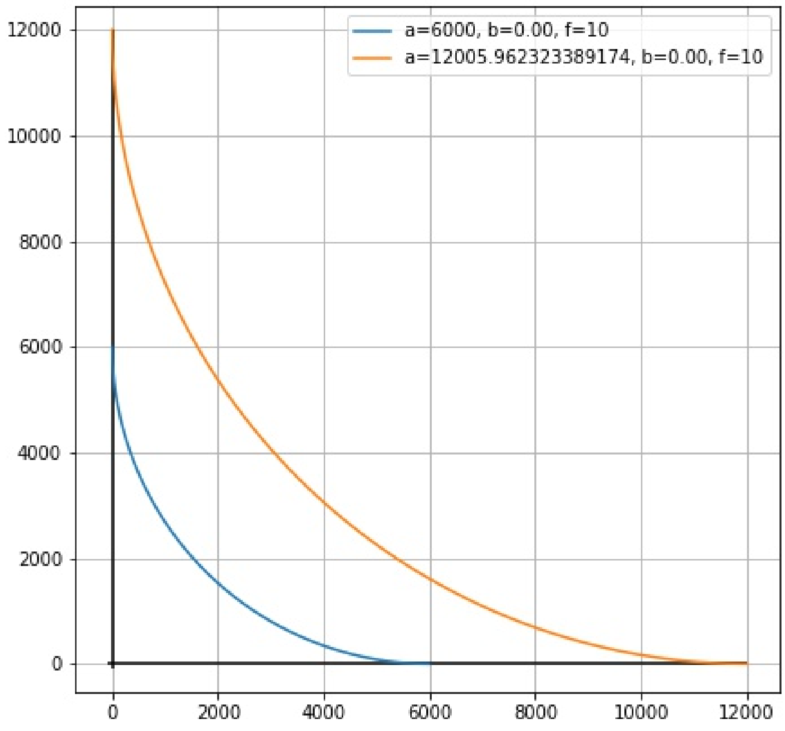

Figure 5.2: The bonding curve of Constant Ellipse Function.

Another important thing is marginal price. It is also called relative price between two assets which is determined by taking ratio of the derivatives of the cost function with respect to each asset. It is written as the following

$$p_{i,j} = \frac{P_i(q)}{P_j(q)} = \frac{2(q_i - a) + b\sum_{k\neq i} q_k}{2(q_j - a) + b\sum_{k\neq j} q_k}, \tag{5.2}$$

where $P_i(q)$ is the derivative of the cost function, $P_i(q) = \frac{\partial C(q)}{\partial q_i}$.

As the author mentioned in [1], the CEF has an advantage compared to the constant product function that it can prevent slippage rate problem which caused by massive trades compared to liquidity of the pool (usually occurs when the pool is still small). In Constant Product function, the price amplitude is from zero to infinity. The slippage rate problem causes pool's relative price go to infinity which is infeasible for trading. Meanwhile, CEF can have an amplitude price. This price amplitude can adapt to liquidity of the pool automatically, in SenSwap, the price amplitude gets larger when liquidity of the pool gets larger.

5.2 SenSwap

Like Uniswap, SenSwap uses the pricing mechanism of pool of two assets. Therefore, a liquidity pool in SenSwap is a pool of pair of tokens. Thus, the cost function is written as the following

$$C_{a,b,f}(x,y) = (x-a)^2 + (y-a)^2 + bxy = a^2 + f, \tag{5.3}$$

where x and y are reserves of token X and Y respectively; parameter a is the position of ellipse; and b controls shape of ellipse (i.e. $b = 0$ the ellipse is a circle). It can be noticed that we modify the right hand side of Eq. 5.3. We introduce a new parameter $0 \leq f \leq a^2$ for controlling price amplitude of the pool. The parameter f must be relatively small compared to the liquidity in order to keep its price amplitude large enough for trading. If $f \longrightarrow 0^+$, the price amplitude approximates $(0, \infty)$. Before we go to the formula determining price amplitude, we can take a look at marginal price formula as the following

$$p_{X/Y} = \frac{P_X(x,y)}{P_Y(x,y)} = \frac{2(x-a)+by}{2(y-a)+bx}. \tag{5.4}$$

Price amplitude is defined when $x = 0$ for upper bound and $y = 0$ for lower bound, its formula is written as below

$$(lower, upper) = \left(\frac{2\sqrt{f}}{b(a-\sqrt{f})-2a}, \frac{b(a-\sqrt{f})-2a}{2\sqrt{f}} \right).$$

All the transactions after which marginal prices are out of price amplitude will be reversed.

5.3 Offered Features

5.3.1 Swap Procedure

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. Since the CEF remains constant during the trading process, then

$$C(q_1) = C(q_2)$$

$$(x_1-a)^2 + (y_1-a)^2 + bx_1y_1 = (x_2-a)^2 + (y_2-a)^2 + bx_2y_2. \tag{5.5}$$

Associate 5.3 and 5.5, we obtain

$$y_2 = \frac{(2a-bx_2) - \sqrt{(2a-bx_2)^2 - 4((x_2-a)^2-f)}}{2}. \tag{5.6}$$

And $\Delta y = y_2 - y_1$. This formula allows us to determine exchanging prices that traders must pay when executing their transactions.

Let's take an example (see Figure 5.3). 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.

5.3.2 Liquidity Pool

In liquidity provision, there are two criteria that we desire. First, when liquidity providers deposit an arbitrary amount of each token, the marginal 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 respect to the amount of reserves, the larger price amplitude. In this case, we keep f constant during the liquidity provision.

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

$$a = (x + y) + \frac{xy(1-p)}{px - y} + \sqrt{\frac{x^2y^2(1-p)^2}{(px-y)^2} + f} \tag{5.7}$$

$$b = \frac{2}{px - y}(x - py - (1-p)a) \tag{5.8}$$

where (x, y) is the pool after depositing, p is the marginal price. With this formula, a pool can be deposited merely one side of the pool, this is called single exposure. We will discuss this in the next section.

After depositing, liquidity providers (LP) will receive a proportion that represents for their contribution to the pool. If a 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, the LP can withdraw part or all of his/her liquidity in the pool. Moreover, liquidity provision also makes price amplitude larger. With a liquidity pool of $(90, 1027.2147)$ after deposit $(30, 0)$, its price amplitude goes from $(0.00204, 489.27543)$ to $(0.00105, 949.02035)$.

5.3.3 Single Exposure

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 minor 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 $(90, 1027.2147)$ (see Figure 5.3). 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.

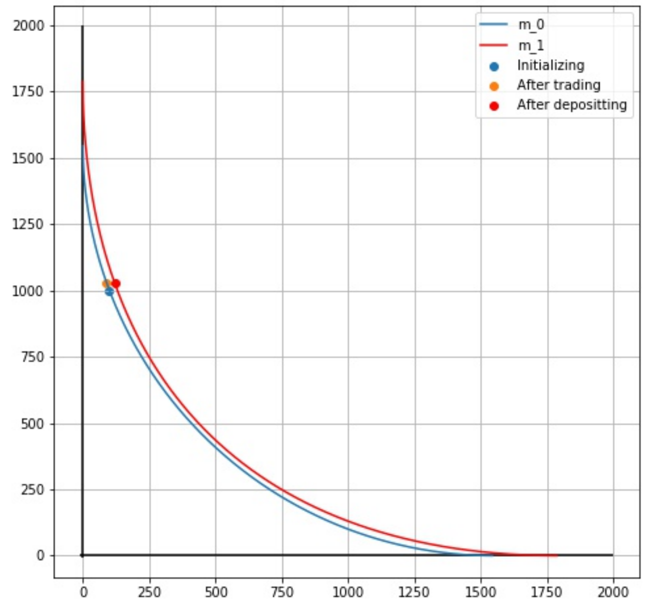

Figure 5.3: This is the proof of concept of SenSwap. The blue line m_0 is the bonding curve after the first depositing. At this time, the pool X/Y is $(100, 1000)$ which is the blue dot. The orange dot is the state of the pool after a trading in which a trader gets 10 coins of token X and pays 27.2147 coins of token Y. The red line and red dot in the new state of the pool after single exposure $(30, 0)$ to 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.

Chapter 6

Implementation

6.1 System Design

Senswap system consists of three main components including

- **Liquidity Pool**: store all information of a smart contract including reserves, bonding curve, and liquidity providers' contributions.
- **Transaction Manager**: manage all transactions between Users and Liquidity Pool, make the system more secured.
- **Users**: include traders and liquidity providers information.

6.1.1 Liquidity Pool

Liquidity Pool is featured by three parameters a, b, f. The parameters is called bonding curve. They were used to determine the marginal price when a trading transaction is executing. Parameters a and b characterize the shape and size of bonding curve which means they are used for calculating trading price and marginal price. Meanwhile, f is to control the price amplitude. We let f fixed at one during the existent of the pool, by this, the price amplitude can widen when liquidity added.

The pair (x, y) is the reserves corresponding to pair of tokens in the pool. We will introduce the procedures of how the quantities were initialized, traded and provided by uses.

When Liquidity providers contribute to the pool, their contributions are stored in form of pair (s_x, s_y) which is the proportion of the additional liquidity. This pair of numbers will be used for withdrawing liquidity when users leaving. This information was held by both the pool and liquidity providers.

6.1.2 Transaction Manager

Transaction Manager is responsible for verifying and executing transactions between users and liquidity pools. We divide transactions into two types which are transactions between pools and traders; ones between pools and liquidity providers.

Transactions between pools and traders are trading transactions. Traders enter symbols of traded tokens and the amount of token they would like to get. Transaction Manager determines the pool and query the information of the pool. It calculates the price and check if the price exceed the price amplitude of the pool. If the transaction satisfies requirements, then it will execute the transaction and return to trader amount of required token.

The second type of transaction is deposition and withdrawing ones. In deposition, transaction manager receives requests from liquidity providers including deposited amount of each token. The transaction manager determines the pool and uses the information of the pool to calculate a and b. If there exists the pair of (a, b) satisfying the constraints, the transaction will be executed and return to liquidity providers the token share representing their contributions to the pool. In withdrawing transactions, transaction manager employs the proportion of contribution of liquidity providers and their token share to compute the amount they wish to withdraw and update contributions of all liquidity providers of the pool.

6.1.3 Users

This module is to store information of users in Senswap including their wallets and token shares of each pool. The token shares should be the same as token shares stored in the liquidity pool.

6.2 Features

6.2.1 Bootstrap of Liquidity Pool

To initialize a liquidity pool, we need to provide a certain amount of liquidity to the pool. At this point, we can leverage the advantage of flexibility of Senswap that we can deposit an arbitrary amount of each token. Then we use the equations 5.7 and 5.8 to calculate the pool's parameters. The parameter f we will leave with value one. The good thing of our formula is that these equations, we have the parameter p which the marginal price, in this case, p is the current price of the reference market. This make Senswap can approximate market price at the beginning of the pool with arbitrary amount depositing liquidity.

The bootstrap API is

```
bootstrap(token_X, token_Y, amount_X, amount_Y, ref_price)
```

6.2.2 Liquidity Provision

Liquidity provision also use the equations 5.7 and 5.8 to calculate the pool's parameters. However, the parameter p is the current marginal price of the pool instead of reference market.

```
deposit(user_id, token_X, token_Y, amount_X, amount_Y) -> token_share
```

`user_id` is to determine user in the list of contributors in liquidity pool

6.2.3 Token Trading

```
swap(user_id, token_out, amount_out, token_in) -> amount_in
```

- `token_out` is the token that trader want to buy,
- `amount_out` is the amount of `token_out`,
- `token_in` is the token that trader want to pay,
- `amount_in` is the amount of `token_in` that trader must pay.

Chapter 7

Evaluation

Chapter 8

Economic Model

Chapter 9

Conclusion

Bibliography

[1] Yongge Wang. *Automated Market Makers for Decentralized Finance (DeFi)*. URL: https://arxiv.org/abs/2009.01676 (pages 6, 7).