hAMM

Daniel[1] and Keyshoe[2]

Abstract— **A novel design of half-real and half-virtual automated market maker (hAMM) algorithm suitable for non-fungible token (NFT) swap trading has been proposed in this paper. Considering the limited NFT decentralized trading market size and sequential large slippage compared with fungible tokens, we have researched the constant product automated market making (AMM) model implemented by Uniswap [1] and vAMM model invented by Perpetual [2] in a deep level, then we find a solution to adjustedly solve it. Another advantage coming along is there will be an efficient and automated buy-out mechanism of a swapping pool or NFT vault rather than the inefficient Dutch auction [3]. This paper will introduce technical analysis of hAMM, its critical points and simulations to better show its advantages in the field of NFT's fractional protocols among projects like unicly [4], nftx [5], nft20 [6], fractional [7], and etc.**

I. INTRODUCTION

Until the end of 2018, there is barely no convenient decentralized exchange (DEX) infrastructure in most public chains, and centralized exchanges (CEX), like Coinbase, Binance, OKEx and etc, are the most popular places for traders to buy and sell cryptocurrencis. Although the speed of trading is Satisfactory, traders have to trust the custody and security of the centralized platforms and bear the potential price manipulation behind them.

The DEXs back then attempted to use the classical order book mechanism, like EtherDelta, 0x or Idex, which suffered from liquidity issues as making (and cancelling) orders required spending gas and waiting for block confirmation time. And Ethereum's low TPS (transactions per second) limited the number of effective transactions. This was especially a huge problem for market makers who provides liquidity in order book exchanges. Making orders in a market typically requires constantly adjusting buy and sell orders to the latest price to get them filled by takers with high probability. However, the submitted orders cost money and time, meaning the market makers may lose more than what they profit from the bid-ask spread, the difference between the highest offered buy price and lowest sell price. The problem exists until AMM protocol firstly adopted by Uniswap comes out to the blockchain.

With AMM, traders do not need to worry about the security and manipulation since the price moving curve and asset custody are guaranteed by the smart contract code logic running on virtual machines, which cannot be sabotaged by anyone except the underlying blockchain consensus is compromised. Unlike order book mechanism, the swap needs in AMM can be satisfied immediately and completed within several block times as long as the invocation parameters are configured well and the market makers can merely supply the funds to the AMM protocol rather than specify the latest price when providing liquidity.

Uniswap is a protocol for decentralized exchange of ERC20 [8] tokens utilizing the constant product market maker model [10]. This AMM has a particularly desirable feature where it can always provide liquidity, no matter how large the order size nor how tiny the liquidity pool. The trick is to asymptotically increase the price of the token as the desired quantity increases. While larger orders tend to suffer more from price slippage, the system never has to worry about running out of liquidity. With AMM drawing more attention and becoming popular, the team behind Curve [11] proposed another AMM algorithm called StableSwap [12] designed for different stable tokens like USDT, USDC, DAI, TUSD and later adopted it in different sets of tokens with similar value like wBTC, renBTC, sBTC set.

Along as the decentralized finance (DeFi) flourishes, the decentralized derivative protocols spring out like perpetual protocol [13], Mai protocol [14] created by MCDEX [15] and Flamingo perp protocol. Perpetual utilizes the idea behind constant product AMM and invented virtual AMM named as vAMM in which the constant product value K can be adjusted by the vAMM operator manually and reasonably based on the market performance in the deep dive introduction here [16], which is also the core behind the Flamingo perp derivative product. As for MCDEX, the market participants in perpetual derivative consist of both liquidity providers and traders while perpetual protocol only has traders, meaning the liquidity behind MCDEX's Mai procotol is backed by providers while liquidity in perpetual protocol is supported by the constant K. The ongoing game behind perpetual protocol and flamingo perp is between traders while that behind mcdex is between liquidity providers and traders. At this point, it should comes to our mind that K of constant product AMM model is especially important. By intuition, if we initiate two trades of the same size in similar swap pools with different liquidity or different constant product K, large K will result in small slippage or insensitive for price change, which is good for traders yet may be bad for liquidity providers (or staking miner) considering the pool is the pair of some project token and UDST stable coin. Technically speaking, there is no standard for what the slippage or price sensitivity should be when a trade with a certain volume hits the CEX market, or more specifically, a swapping pool. In CEX, the market behavior absorbs the dump or pump and it's almost only interest related to the order takers. However, in

*This work was not supported by any organization

[1]Development leader of hoglet.io project .

[2]Operation leader of hoglet.io project .

DEX, different AMM models will have different impact on both liquidity providers and traders in one trade with same volume.

Obviously there is no best answer to the above mentioned ambiguous problem. But when it comes to real scenario, we should do our best to rebalance the interest of both traders and liquidity providers, especially considering recent emerging fractional protocols for NFTs rather than using the existing infrastructure just for the convenience of project development team. Let us take a peek at nftx CryptoPunk pool in SushiSwap [17], the most popular fractional protocol, at timestap of 1636605144 (aprox. November 11, 2021 4:32:24 AM), the cost to buy one Punk ERC20 token is 83.2271 Ether while the gains to sell one Punk ERC20 token is 75.6482 Ether, from which we see the price slippage is approximately 10%. The price impact is almost unacceptable if we consider CryptoPunk pool is the largest pool in nftx platform, meaning other pools may probably have larger slippage when trade happens. It's may be delightful for those who hold a bunch of Punk NFTs or Punk ERC20 tokens if the buy swaps happen yet painful for those who just bought Punk ERC20 or users wishing to swap one NFT for another NFT in nftx platform. And the same thing happens in other fractional protocol platform like unily, nft20, fractional.art and etc.

To better mediate the above problem for those who have realized it and seek for a probably better protocol to perform the NFT swap trade, we inspired by perpetual protocol introduce hAMM algorithm, combining the traditional constant product AMM model with its virtual AMM model to make the slippage smaller yet not too small such that it's even too hard or needs too much cost to move the price up or down.

The main contributions are as follows.

(a) A novel AMM algorighm, hAMM is proposed, which gives us another option to make markets especially when the market depth (or the whole market volume) is not enough or large.
(b) hAMM consists of the real parts and virtual parts, the ratio between which can be adjusted at the creation of the pool or pair based on the expectation of the pool's scale or size.
(c) It is compatible with the constant product AMM model if the virtual part is set to empty or zero, which means if the pool creator doesnot want to utilize the design of virtual parts in hAMM, he/she can just set it zero so that hAMM model will converge to AMM in Uniswap or SushiSwap.

II. hAMM MODEL

A. Model Explanation

In decentralized exchange adopting constant product AMM model that trades two token X and Y, let us assume x and y be the number of tokens X and Y currently reserved, respectively, and we know they should follow the equation, where c denotes the abbreviation of constant.

$$xy = c \tag{1}$$

Let us define Δx and Δy are fluctuations of x and y after a normal trade happens, where $\Delta x > 0$ and $\Delta y < 0$ if someone sell Δx of token X to buy token Y and $\Delta x < 0$ and $\Delta y > 0$ if someone sell Δy of token Y to buy token X. Therefore, we have

$$xy = (x + \Delta x)(y + \Delta y) = c \tag{2}$$

The swap trading price of token X in terms of token Y is

$$p_{swap} = |\frac{\Delta y}{\Delta x}| \tag{3}$$

And the price slippage can be deducted as

$$\begin{aligned} p_{slippage} &= p_{after} - p_{before} \\ &= \frac{y + \Delta y}{x + \Delta x} - \frac{y}{x} \\ &= \frac{c}{(x + \Delta x)^2} - \frac{c}{x^2} \\ &= -c\frac{\Delta x^2 + 2x\Delta x}{x^2(x + \Delta x)^2} \end{aligned} \tag{4}$$

It should be noted that we have not considered the swap fee or trading fee in above constant product AMM model utilized in UniSwap, SushiSwap, Perpetual protocol, MCDEX and Flamingo Perp product. Considering swap fee rate f, where $0 \leq f < 1$, we can demonstrate the gradually increased constant c in the following equations from state to another state. Let's take a snapshot and define the initial state as

$$x_0 y_0 = c_0 \tag{5}$$

If Alice sells Δx_0, she should get $-\Delta y_0$, which can be calculated from

$$x_0 y_0 = [x_0 + \Delta x_0(1 - f)](y_0 + \Delta y_0) = c_0 \tag{6}$$

If Bob sells Δy_1, he would get $-\Delta x_1$ of token X which can be calculated from

$$x_1 y_1 = (x_1 + \Delta x_1)[y_1 + \Delta y_1(1 - f)] = c_1, \tag{7}$$

where

$$\begin{aligned} x_1 &= x_0 + \Delta x_0 \\ y_1 &= y_0 + \Delta y_0 \\ c_1 &= x_1 y_1 \\ &= (x_0 + \Delta x_0)(y_0 + \Delta y_0) \end{aligned} \tag{8}$$

And apparently, we know $c_1 = c_0$ if $f = 0$ and $c_1 > c_0$ if $f > 0$ where the growing constant c contributes to the liquidity providers or the fee acts as incentives to encourage users to provide liquidity so that the swap slippage is better trader-friendly, which will appeal more traders and thus liquidity providers can profit more from trading or swapping fee.

B. How hAMM works

In hAMM model, the price (or token volumes) still follows Equation 1 with variables x and y consisting of two parts, respectively.

$$x_h y_h = c_h, \tag{9}$$

where

$$\begin{aligned} x_h &= x_r + x_v \\ y_h &= y_r + y_v \end{aligned} \tag{10}$$

Within swapping, the real parts, x_r and y_r, are modified while virtual parts, x_v, y_v, will remain the same. Within liquidity changing process, virtual parts will and should be modified while the trading price should remain the same before and after liquidity change.

Based on the above principles, the details of potential interaction are provided below.

1) create swap pools: Trading pair is created with underlying two ERC20 smart contract addresses as parameters in constant product AMM while additional parameter r, ratio for virtual to real part, is required to create hAMM pair. It is used to determine the initial virtual x_{v0} and y_{v0} compared with initial real x_{r0} and y_{r0}. For instance, the first liquidity provider (LP_1) wants to add x_{r0} amount of token X and y_{r0} amount of token Y to the created hAMM pair and obtain l_0 liquidity token. We have

$$x_{h0} \times y_{h0} = c_{h0}, \tag{11}$$

where

$$\begin{aligned} x_{h0} &= x_{r0} + x_{v0} \\ &= x_{r0} + r x_{r0} \\ &= x_{r0}(1+r) \\ y_{h0} &= y_{r0} + y_{v0} \\ &= y_{r0} + r y_{r0} \\ &= y_{r0}(1+r) \\ l_0 &= \sqrt{x_{h0} \times y_{h0}} \\ &= (1+r)\sqrt{x_{r0} \times y_{r0}} \end{aligned} \tag{12}$$

2) swapping: Suppose user A wants to sell Δx amount of token X to get -Δy amount of token Y, we know

$$\begin{aligned} x_{h0} \times y_{h0} = x_{h1} &\times y_{h1} = c_{h0} \\ x_{h1} &= x_{h0} + \Delta x \\ y_{h1} &= y_{h0} + \Delta y \end{aligned} \tag{13}$$

where the following condition should be met.

$$\begin{aligned} x_{h1} &= x_{h0} + \Delta x \geq x_{v0} \\ y_{h1} &= y_{h0} + \Delta y \geq y_{v0} \end{aligned} \tag{14}$$

which means after swapping, the remaining amounts of token X and Y should always be greater than or equal to their corresponding virtual parts.

That is to say, in hAMM, there is a limit condition for a given state to be swapped into another state, which is the following for Equation 13

$$\begin{aligned} \Delta x &\in [-x_{r0}, \frac{c_{h0}}{y_{v0}} - x_{h0}] \\ \Delta y &\in [-y_{r0}, \frac{c_{h0}}{x_{v0}} - y_{h0}] \end{aligned} \tag{15}$$

3) add liquidity: Suppose now second liquidity provider (LP_2) wants to obtain l_1 liquidity token, he/she needs to provide Δx_r and Δy_r amount of token X and Y. We have

$$x_{h2} \times y_{h2} = c_{h2} \tag{16}$$

$$\begin{aligned} x_{h2} &= x_{h1} + \Delta x_r + \Delta x_v \\ y_{h2} &= y_{h1} + \Delta y_r + \Delta y_v \\ \frac{x_{h1}}{y_{h1}} &= \frac{x_{h2}}{y_{h2}} \\ \frac{l_1}{l_0} &= \frac{\Delta x_r}{x_{r1}} = \frac{\Delta y_r}{y_{r1}} = \frac{\Delta x_v}{x_{v0}} = \frac{\Delta y_v}{y_{v0}} \end{aligned} \tag{17}$$

4) remove liquidity: The process of remove liquidity is similar to liquidity addition. It's needed to remove the virtual parts of X and Y proportionally and remove the real parts proportionally back to liquidity provider while keeping the swapping price consistent.

5) impermanent loss: Introduced by Binance academy [18], we know the more volatile the assets are in the pool, the more likely it is that you can be exposed to impermanent loss. However, if there's a lot of trading volume happening in a given pool, it can be profitable to provide liquidity even if the pool is heavily exposed to impermanent loss.

Although the above loss obviously may exist, one may have noticed the provided real (actual) amounts of token X and Y are not proportional to the swapping price within liquidity addition or remove action and may ask if others can flash attack the trading pool. To break it into parts, we need to check if the attacker can profit in one transaction containing adding/removing liquidity, swapping and then removing/adding liquidity. Let us just focus on the price change Equation 13, the constant product c_{h0} will remain the same with swapping fee $f = 0$ and even increase with $0 < f < 1$ while the underlying virtual parts remain the same within swapping action. Conditions of Equation 17 makes sure the virtual parts of hAMM remain the same no matter the attacker add liquidity first then remove liquidity or remove first then add it.

6) fee: The fee calculation is the same as constant product AMM model, meaning a small portion of the actual trading volume will be deducted from trader's input and acts as contribution to liquidity providers (LPs) to help them profit, or at least negate the loss from price change.

C. Model analysis

Suppose the initial liquidity in (x, y) AMM coordinate is the same as the real parts state in (x_h, y_h) hAMM coordinate. For any point or price, let's assume the potential token volume change is legal and the initial price at AMM equals that in hAMM.

1) Slippage: For any given legal Δx sold to pools, the price slippage in AMM is shown in Equation 4, and the slippage in hAMM $p^{hAMM}_{slippage}$ can be concluded as

$$p^{hAMM}_{slippage} = -c_h \frac{\Delta x^2 + 2x_h \Delta x}{x_h^2 (x_h + \Delta x)^2} \tag{18}$$

with premise shown as below where the price before swapping is defined as p_b.

$$\begin{aligned} p^{AMM}_{before} = \frac{y}{x} \equiv p^{hAMM}_{before} = \frac{y_h}{x_h} = p_b \\ c_h = x_h y_h = (1+r)^2 \times c = (1+r)^2 \times xy \end{aligned} \tag{19}$$

Now let's dive deep into the price slippage ratio between hAMM and AMM $R_{ps} = \frac{p^{hAMM}_{slippage}}{p^{AMM}_{slippage}}$.

$$\begin{aligned} R_{ps} &= (1+r)^2 \frac{\Delta x^2 + 2x_h \Delta x}{x_h^2 (x_h + \Delta x)^2} \frac{x^2 (x+\Delta x)^2}{\Delta x^2 + 2x\Delta x} \\ &= \frac{(1+r)^2 [\Delta x^2 + 2(x_r + x_v)\Delta x]}{(x_r + x_v)^2 (x_r + x_v + \Delta x)^2} \frac{x^2 (x+\Delta x)^2}{\Delta x^2 + 2x\Delta x} \end{aligned} \tag{20}$$

When $r = 0$, $x_v = 0$ and $y_v = 0$, then $x_h = x$, $y_h = y$, it can derived that $R_{ps} = 1$, meaning the hAMM converges to AMM.

When $r > 0$,

$$\begin{aligned} x &= \sqrt{\frac{c}{p_b}} \\ x_h &= \sqrt{\frac{c_h}{p_b}} = (1+r)\sqrt{\frac{c}{p_b}} \\ R_{ps} &= \frac{c_h}{c} \frac{\Delta x^2 + 2x_h \Delta x}{x_h^2 (x_h + \Delta x)^2} \frac{x^2 (x+\Delta x)^2}{\Delta x^2 + 2x\Delta x} \\ &= \frac{x_h y_h}{xy} \frac{\Delta x^2 + 2x_h \Delta x}{x_h^2 (x_h + \Delta x)^2} \frac{x^2 (x+\Delta x)^2}{\Delta x^2 + 2x\Delta x} \\ &= \frac{\Delta x^2 + 2x_h \Delta x}{(x_h + \Delta x)^2} \frac{(x+\Delta x)^2}{\Delta x^2 + 2x\Delta x} \\ &= \frac{(\frac{\Delta x}{x_h})^2 + 2\frac{\Delta x}{x_h}}{(1+\frac{\Delta x}{x_h})^2} \div \frac{(\frac{\Delta x}{x})^2 + 2\frac{\Delta x}{x}}{(1+\frac{\Delta x}{x})^2} \end{aligned} \tag{21}$$

For $f(z) = \frac{z^2+2z}{(1+z)^2} = 1 - \frac{1}{(1+z)^2}$, known as a monotonically increasing function, it is for sure $f(\frac{\Delta x}{x_h}) < f(\frac{\Delta x}{x})$ since $\frac{\Delta x}{x_h} < \frac{\Delta x}{x}$. Therefore $R_{ps} < 1$ always holds true when $r > 0$.

Hence, it has been mathematically proved that hAMM does have smaller slippage than AMM.

2) Price moving cost: Also can be called as critical points as long as it helps your understanding.

We are seeking one appropriate trade-off between small slippage and heavy cost behind price move. In constant product AMM, the cost to move price is unreasonable when swapping happens along large x or large y. In hAMM, the similar scenario occurs around the maximum of x_h or y_h. Next, we will analyze the worst case step by step.

For any given legal Δx sold to pools with conditions shown in Equation 19, the relative effort or cost of token Y is defined as Δy in AMM and as Δy_h in hAMM.

$$\begin{aligned} \Delta y &= -(y - \frac{c}{x+\Delta x}) \\ \Delta y_h &= -(y_h - \frac{c_h}{x_h + \Delta x}) \end{aligned} \tag{22}$$

$$\begin{aligned} \frac{\Delta y_h}{\Delta y} &= \frac{p_b x_h - \frac{c_h}{x_h + \Delta x}}{p_b x - \frac{c}{x+\Delta x}} \\ &= \frac{p_b (1+r)x - \frac{(1+r)^2 c}{(1+r)x + \Delta x}}{p_b x - \frac{c}{x+\Delta x}} \\ &= \frac{p_b (1+r)^2 x^2 + p_b (1+r)x\Delta x - (1+r)^2 c}{(1+r)x + \Delta x} \div \\ &\quad \frac{p_b x^2 + p_b x \Delta x - c}{x + \Delta x} \\ &= \frac{(1+r)(x+\Delta x)}{(1+r)x + \Delta x} \\ &= 1 + \frac{r\Delta x}{(1+r)x + \Delta x} \end{aligned} \tag{23}$$

When $r = 0$, hAMM turns into AMM, $\frac{\Delta y_h}{\Delta y} = 1$.

When $r > 0$, it can also be written as

$$\frac{\Delta y_h}{\Delta y} = 1 + \frac{\Delta x}{(1+\frac{1}{r})x + \frac{\Delta x}{r}} \tag{24}$$

When $r \to +\infty$, $\frac{\Delta y_h}{\Delta y} = 1 + \frac{\Delta x}{x}$, meaning the cost in hAMM to push token X to move Δx depends on the value of Δx and the state x before swap.

When r is a specific non-zero positive limited value, from the following Equation 25, we know $\frac{\Delta y_h}{\Delta y}$ increases along with r increasing, which indicates that the pool creator should be cautious about the value of r to make sure it won't be too large, otherwise it will be too hard for the market to move the price up or down.

$$\begin{aligned} \frac{\Delta y_h}{\Delta y} &= 1 + \frac{r}{(1+r)\frac{x}{\Delta x} + 1} \\ &= 1 + \frac{1}{\frac{x}{\Delta x} + (1 + \frac{x}{\Delta x})\frac{1}{r}} \end{aligned} \tag{25}$$

III. SIMULATION RESULTS

Aside from what the trading curve looks like, the fundamental question one may ask how much effort or cost do we need to cast to the hAMM market to make the price change be its $\frac{1}{100}$, $\frac{1}{50}$, $\frac{1}{20}$, $\frac{1}{10}$, $\frac{1}{50}$, $\frac{1}{20}$, $\frac{1}{10}$, $\frac{1}{5}$, $\frac{1}{2}$, 1, 2, 5, 10, 20, 50, 100? Especially, what is the difference if we compare the results with those from existing AMM models or order book utilized by centralized exchange.

A. Trading Curve

Fig.1 shows that hAMM has more smooth curve than AMM. The larger virtual to real part parameter r is, the more the trader would enjoy it. Considering the head and tail of one curve, hAMM can provide much better user performance. And along with the liquidity come and left, the head and tail of hAMM curve is always acceptable compared with constant product AMM algorithm.

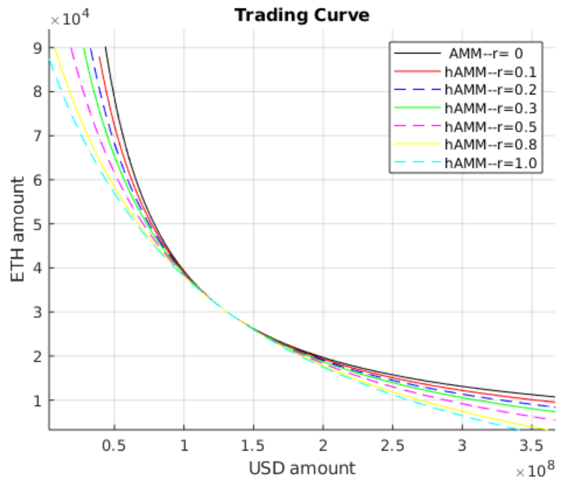

Fig. 1: Trading curve of hAMM algorithm with no swapping fee.

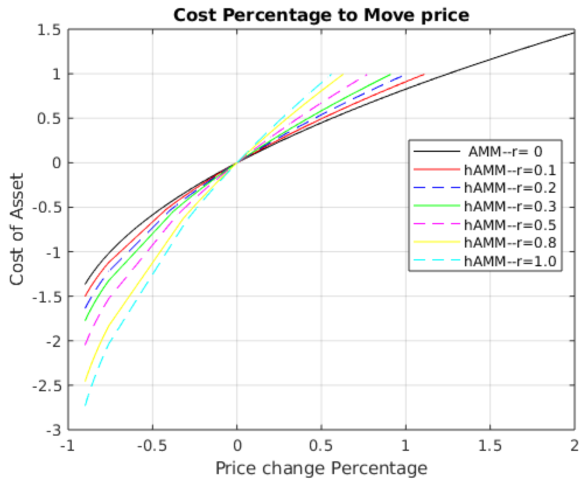

Fig. 3: Cost percentage of asset when price changes.

B. *Slippage Curve*

Fig.2 directly shows that hAMM has smaller slippage than AMM under the same price quoted from different pools. The larger virtual to real part parameter r is, the smaller the slippage would be. However, it's known that smaller slippage does not always means better since AMM with smaller slippage requires more cost to change the price, shown in Fig.3.

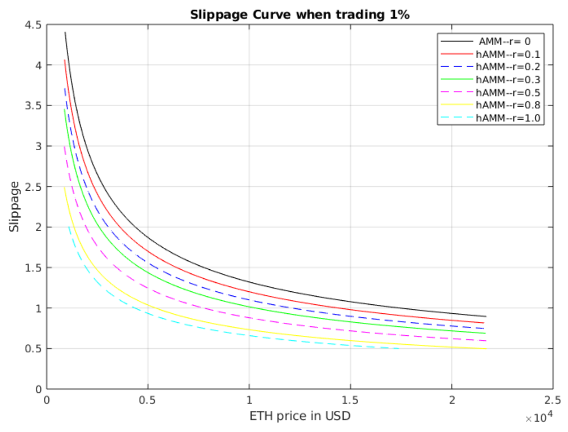

Fig. 2: Slippage curve along with the ETH price in USD with normal 0.3% fee.

C. *Cost to Move price*

In Fig.3, X-axis price denotes how much the price is changed compared with the initial price coming from one snapshot of Uniswap ETH-USDT swapping pair smart contract, which can be check by anyone from the simulation source code [19]. Although larger r provides smaller slippage, it also requires more cost to change price, which may not be expected by some project development teams.

IV. CONCLUSIONS

To sum up, constant product AMM algorithm adopted by several mainstream decentralized exchanges is introduced. The model has been utilized by almost all NFT fractional platforms, which results in the slippage issue. Therefore, hAMM algorithm, which is slightly complicated than AMM, has been proposed in this article. The technical analysis and simulation results are provided above. The source code is open and anyone is welcome to make a pull request to the github repository[19] or contact us through email [20] directly to discuss hAMM and improve it.

REFERENCES

[1] https://uniswap.org/
[2] https://www.perp.fi/
[3] https://en.wikipedia.org/wiki/Dutch_auction
[4] https://www.unic.ly/
[5] https://nftx.org/
[6] https://nft20.io/
[7] https://fractional.art/
[8] https://ethereum.org/en/developers/docs/standards/tokens/erc-20/
[9] https://hackmd.io/@HaydenAdams/HJ9jLsfTz?type=view
[10] https://github.com/runtimeverification/verified-smart-contracts/blob/uniswap/uniswap/x-y-k.pdf
[11] https://curve.fi/
[12] https://curve.fi/files/stableswap-paper.pdf
[13] https://docs.perp.fi/
[14] https://docs.mcdex.io/protocol/mai-protocol-v3
[15] https://mcdex.io/homepage/
[16] https://gov.perp.fi/t/deep-dive-into-our-virtual-amm-vamm/38
[17] https://app.sushi.com/swap?outputCurrency=ETH&inputCurrency=0x269616d549d7e8eaa82dfb17028d0b212d11232a
[18] https://academy.binance.com/en/articles/impermanent-loss-explained
[19] https://github.com/hogletdev/docs/blob/main/resource/hamm/sim.m
[20] support@hoglet.io