Optimal Trading on a Dynamic Curve Automated Market Maker

Shuangge Wang and Bhaskar Krishnamachari
Ming Hsieh Dept. of Electrical and Computer Engineering

Viterbi School of Engineering
University of Southern California
Los Angeles, USA
{larrywan, bkrishna}@usc.edu

***Abstract*—In the emerging realm of decentralized finance (DeFi), most of the existing Automated Market Maker (AMM) protocols used by major platforms like Uniswap and Curve are governed by a static mathematical equation, such as the constant product curve. One major shortcoming of these curves is that they require external forces to maintain the price of the liquidity pool (LP), subjecting the LP to loss due to arbitrage. A novel solution, the dynamic curve AMM, was recently proposed to ensure that the pool price always matches the market price, making the LP invulnerable to arbitrageurs. Dynamic curves, however, have a path-dependent trading problem, meaning that the number of trades and the distribution of trades affect the trader's gain. We show how to find the optimal trading policy for a dynamic AMM curve under several settings. We first show that in a zero-transaction-fee setting the optimal trading policy is to place infinitesimally small trades, resulting in zero slippage. Then, we present an algorithm that computes the optimal policy in a fixed-number-of-trade setting. Though the problem has an exponentially large search space, our algorithm utilizes dynamic programming to achieve a polynomial run-time. Finally, we generalize the solution to more complex settings, including a per-order-fee setting and a percentage-fee setting.**

***Index Terms*—Blockchain, DeFi, AMM, Dynamic Programming**

I. Introduction

In recent years, the emergence of cryptocurrency and programmable blockchain platforms has facilitated the research and deployment of decentralized exchange (DEX) protocols. These protocols allow the trading of different coins against each other without involving any centralized financial intermediaries.

Traditional exchanges utilize a centralized limit order book running a double-sided auction to allow peer-to-peer trades. However, these can suffer from low liquidity and trading volume. These issues are addressed by Automated Market Makers, where trading takes place instead against a common liquidity pool.

Originally, AMM pricing functions such as Hanson's logarithmic market scoring rules (LMSRs) [1] were used in the context of prediction markets such as Augur [2]. Another type of AMMs, the curve-based AMMs, are most widely used for decentralized cryptocurrency exchanges such as Bancor [3], Uniswap [4], and Curve [5].

Static curve-based AMMs restrict trading to a fixed function, such as the constant-product curve used in Uniswap. One of the major challenges faced by curve-based AMMs is how to design a curve that is both arbitrageur-proof and low in slippage. The former protects the assets of the LP from arbitrage, and the latter benefits the traders, expanding the DEX's user base.

A recent work by Krishnamachari, Feng and Grippo [6] provides an novel solution that addresses the problem of arbitrage – dynamic AMM curves that are adjusted after trade to match the pool price to the current market price obtained from an Oracle. The main difficulty is that dynamic curves have a path-dependent trading problem. Specifically, the number of trades and the distribution of trades affects the slippage. The higher the slippage, the lower the gain in any exchange. Therefore, to help traders navigate this novel class of dynamic curve AMMs, we intend to find the optimal trading policy in dynamic curves that minimizes the slippage.

We first examine the optimal trading policy in a zero-fee situation, and we conclude that the optimal policy is to make infinite amount of small trades, which results in zero slippage. Then, we proceed to develop an algorithm that generates the optimal trading policy in a fixed-number-of-trades setting. While a naive exhaustive search would result in an exponential run-time to find the optimal policy, we utilize dynamic programming to devise a polynomial-time algorithm. Finally, we apply this algorithm to more complex settings including per-order and percentage-based transaction fees.

II. Related Work

Automated Market Makers were first popularized by Hanson's LMSR for prediction markets [1]. Since AMMs encourage passive market participants with low time preference to lend their digital assets to asset pools, they have been used in numerous online settings with low liquidity and trading volume.

Inspired by practical projects such as Uniswap and Curve, several researchers have studied the alternative class of AMMs known as the constant function market makers (CFMMs) or curve-based AMMs [7], [8]. Curve-based AMMs are governed by a mathematical function that describes how the LP maintains its assets. A classic example of a curve-based AMM is the constant product curve. For instance, consider an LP with two coins X and Y, whose amounts are denoted by x and y. The constant product curve's equation is $y = \frac{k}{x}$.

The pool price, which is the negative of the slope of the curve, is donated by the following equation.

$$p_X(x, y) = -\frac{dy}{dx} \tag{1}$$

One of the major indicators of a good oracle is the equality of the pool price and the market price [7]. However, traditional curve-based AMMs have a deterministic price that only dependents on the x value of the curve, which means that the pool price does not always match the market price. Capitalizing on the price discrepancy, arbitrageurs could compromise the value of the LP. To address this issue, Krishnamachari, Feng and Grippo [6] introduced dynamic curve AMMs that automatically adjust their parameters after each trade to match its price to the market price, preserving the value of the LP.

III. Dynamic AMM

Equation (2) is an example of a dynamic AMM–dynamic constant product curve. Here x_t and y_t denotes the amount of X and Y tokens in the LP at time t; a_t and w_t are state-dependent constants; k is a static constant:

$$y_t = \frac{k}{w_t \cdot (x_t - a_t)} \tag{2}$$

Applying equation (1) and substituting $p_X(x, y)$ with the market price, p_{mkt} (which we assume is constant within the trade), we get the price of the curve as:

$$p_{mkt} = \frac{k}{w_t} \cdot \frac{1}{(x_t - a_t)^2} \tag{3}$$

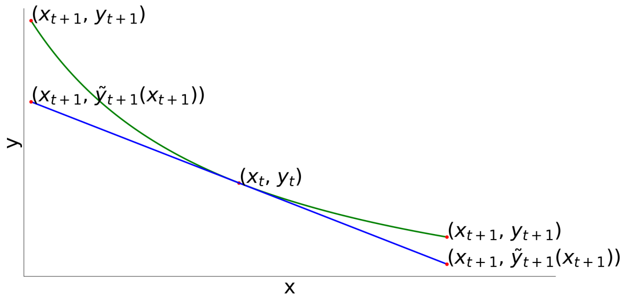
Although the prior work by Krishnamachari, Feng and Grippo ensures that the divergence loss due to arbitrage is zero, it does not immediately address the other performance indicator of an AMM, the slippage.

Consider a case where a trader comes to the AMM to sell X tokens and get Y tokens in return at time t. Since the mathematical model behind the dynamic curves is opaque to traders whose only accessible information are x, y (the corresponding amounts of the X and Y tokens in the pool), and p_{mkt}, the traders would ideally like that the amount of Y tokens they get be directly proportional to the amount of X tokens they sell. Therefore, from their perspective, ideally, the LP's y value after placing one trade at time t that changes the amount of X tokens from x_t to x is described by the tangent line, $\tilde{y}_{t+1}(x)$, of the curve at (x_t, y_t):

$$\tilde{y}_{t+1}(x) = -p_{mkt} \cdot (x - x_t - \frac{y_t}{p_{mkt}}) \tag{4}$$

Since the traders are trading on a convex curve, the y value of the LP after one trade, y_{t+1}, could never, strictly speaking, track a linear relationship with respect to x_{t+1}.

$$y_{t+1} = \frac{k}{w_t \cdot (x_{t+1} - a_t)} \tag{5}$$



(x_{t+1}, y_{t+1})
$(x_{t+1}, \tilde{y}_{t+1}(x_{t+1}))$
y
(x_t, y_t)
(x_{t+1}, y_{t+1})
$(x_{t+1}, \tilde{y}_{t+1}(x_{t+1}))$
x

Fig. 1: Lower Bound

Fig. 1 is a visual representation of the relationship between the LP's ideal (no-slippage) y value corresponding to $\tilde{y}_{t+1}(x_{t+1})$ and the actual y value, y_{t+1}, after one trade. If the trader is buying X tokens, $x_{t+1} < x_t$, and if the trader is selling X tokens, $x_{t+1} > x_t$. Observe that y_{t+1} is always above $\tilde{y}_{t+1}(x_{t+1})$ regardless of selling or buying, which means that there are $y_{t+1} - \tilde{y}_{t+1}(x_{t+1})$ tokens that are supposed to be owned by the trader (in the ideal case) that are retained by the LP. This difference is precisely the slippage. Therefore, we can formulate the problem of minimizing the slippage as that of minimizing the LP's value of y given an amount of X tokens that the trader is interested in buying/selling.

When making multiple trades, the total slippage, S, is the sum of slippages at each individual trade. Below is a more general definition of slippage when placing n trades, and x_{t+1} denotes the x value of the LP at time $t + 1$.

$$S = \sum_{t=0}^{n-1} y_{t+1} - \tilde{y}_{t+1}(x_{t+1}) \tag{6}$$

Using equations (4), we get:

$$\tilde{y}_{t+1}(x_t) = y_t \tag{7}$$

Applying equation (7), we substitute y_{t+1} with $\tilde{y}_{t+2}(x_{t+1})$.

$$S = \sum_{t=0}^{n-1} \tilde{y}_{t+2}(x_{t+1}) - \tilde{y}_{t+1}(x_{t+1}) \tag{8}$$

Since $\tilde{y}_{t+2}$ and $\tilde{y}_{t+1}$ are parallel, we have:

$$\begin{aligned} S &= \sum_{t=0}^{n-1} \tilde{y}_{t+2}(x_n) - \tilde{y}_{t+1}(x_n) \\ &= \tilde{y}_{n+1}(x_n) - \tilde{y}_1(x_n) \end{aligned} \tag{9}$$

Applying equation (7) again, we get:

$$S = y_n - \tilde{y}_1(x_n) \tag{10}$$

Therefore, the slippage when making n trades equals to the difference between the final y value of the LP, y_n, and the y value of the first tangent line at x_n, which is the LP's final x value. We will refer to this difference as the slippage for the rest of this paper.

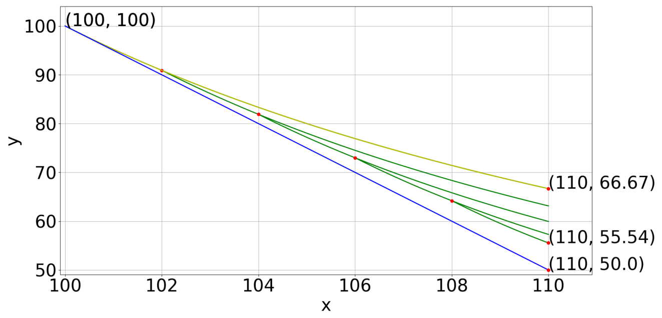


Fig. 2: 1 Trade vs. 5 Trades under the Arithmetic Sequence Policy

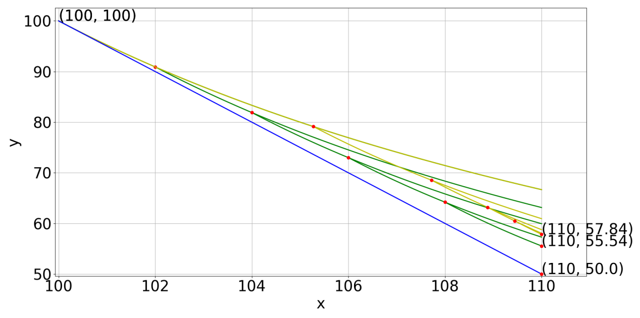


Fig. 3: Greedy Policy vs. Arithmetic Sequence Policy in 5 Trades

A. Illustration of Path Dependence

Since dynamic curve AMMs have state-dependent parameters, adopting different trading strategies to the same volume of trades produces different results. Consider a situation in which the trader sells 10 X tokens in total. The trader begins trading at $(x_0 = 100, y_0 = 100)$; the constant k is 10000; the p_{mkt} is 5. The blue line describes $\tilde{y}_1$. The green and yellow curves each indicate a different policy, with green curves describing the one that yields a lower slippage. The red dots are the intersections of two adjoining dynamic curves. An arithmetic sequence policy refers to placing the same amount of trades each time; a greedy policy refers to placing the amount such that it minimizes the slippage at that trade. Observe that different strategies result in different values of slippage.

Fig. 2 illustrates that the number of trades affect the final slippage, and Fig. 3 shows that the distribution of trades also affects the slippage. Note that the policies in these example are arbitrarily chosen, so these examples alone do not justify that an arithmetic policy is better than a greedy policy.

IV. Optimal Trading under Zero-Fee Setting

In a simple situation where there is no transaction fee, the traders are not penalized for making large number of trades. We conclude that the optimal trading policy is to make infinite number of small trades. The proof is as follows.

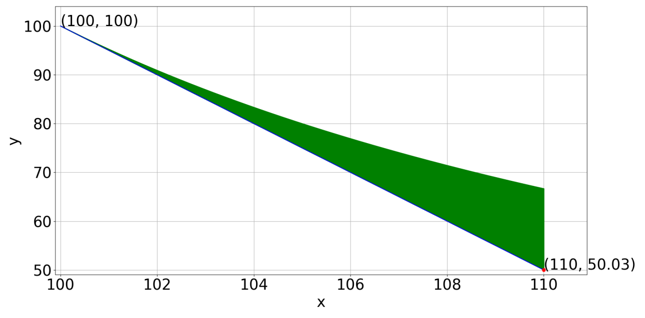


Fig. 4: 1000 Trades

Using equation (2) and (3), we solve for a_t and w_t:

$$\begin{cases} a_t = x_t - \frac{y_t}{p_{mkt}} \\ w_t = \frac{p_{mkt} \cdot k}{y_t^2} \end{cases} \tag{11}$$

Substituting a_t and w_t into equation (5), we get:

$$y_{t+1} = \frac{y_t^2}{p_{mkt} \cdot (x_{t+1} - x_t) + y_t} \tag{12}$$

As we place infinitesimally small trades, we have $x_{t+1} - x_t = dx$.

$$y_{t+1} = \frac{y_t^2}{p_{mkt} \cdot dx + y_t} \tag{13}$$

Rearranging equation (13), we have:

$$y_{t+1} \cdot p_{mkt} \cdot dx + y_{t+1} \cdot y_t = y_t^2 \tag{14}$$

Taking both sides' derivatives with respect to x, we get:

$$\begin{aligned} &\frac{dy_{t+1}}{dx} \cdot p_{mkt} \cdot dx + y_{t+1} \cdot p_{mkt} \cdot \frac{d(dx)}{dx} \\ &+ \frac{dy_{t+1}}{dx} \cdot y_t + \frac{dy_t}{dx} \cdot y_{t+1} = 2y_t \cdot \frac{dy_t}{dx} \end{aligned} \tag{15}$$

Using equation (1) and (3), we substitute $\frac{dy_{t+1}}{dx}$ and $\frac{dy_t}{dx}$ with $-p_{mkt}$. Solving for y_{t+1}, we get:

$$y_{t+1} = \frac{y_t - p_{mkt} \cdot dx}{1 - \frac{d(dx)}{dx}} \tag{16}$$

Since $\frac{\frac{d(dx)}{dx}}{dx} = \frac{d^2x}{dx^2} = 0$, $\frac{d(dx)}{dx}$ is negligible compared to dx.

$$\begin{aligned} y_{t+1} - y_t &= -p_{mkt} \cdot dx \\ dy &= -p_{mkt} \cdot dx \\ \frac{dy}{dx} &= -p_{mkt} \end{aligned} \tag{17}$$

Therefore, as the trader makes infinitesimally small trades, the LP's (x_t, y_t) value pairs track the first tangent line of the curve, $\tilde{y}_1$, meaning that $y_n = \tilde{y}_1(x_n)$, which makes the slippage 0. Since we did not specify the sign of dx, this infinite-trade policy is applicable when both buying and selling. Fig. 4 is an example in which the trader makes 1000 small trades in the same setting as Fig. 2. Observe that the slippage is close to 0.

V. Optimal Trading under Per-Order Fee

A. Optimal Trading with Fixed Number of Trades

In this section, we investigate the optimal trading policy when the trader makes n trades. Though we only examined the case of selling in section B, the convex shape of the curve made it effortless to generalize the algorithm to the case of buying.

Using equation (12), we write a dynamic programming algorithm that returns the lowest possible y value of the LP after n trades. Here x_0 and y_0 denotes the starting position of the LP, and x_n indicates the final x value.

Algorithm 1 Dynamic Programming for Optimal Trading

```
    Input: $x_0, y_0, x_n, p_{mkt}, n$
    Output: $y_f(n) =$ Minimum $y$ at $x_n$
 1: $t \leftarrow 0$
 2: while $t <= n$ do
 3:     for $x \leftarrow x_0$ to $x_n$ do
 4:         if $t == 1$ then
 5:             $OPT(x, t) \leftarrow y_0 - \frac{y_0^2}{p_{mkt} \cdot (x - x_0) + y_0}$
 6:         else
 7:             $OPT(x, t) \leftarrow \max_{x'}(y_0 - \frac{(y_0 - OPT(x', t-1))^2}{p_{mkt} \cdot (x - x') + y_0 - OPT(x', t-1)})$
 8:         end if
 9:     end for
10: end while
        return $y_0 - \max_{x'}(OPT(x', n))$
```

The dynamic programming Algorithm 1 uses a recursion on OPT, which is a mapping from every x and t to the lowest achievable value of y. The run-time complexity of this algorithm is prescribed by the for loop and its nested $\max$ operation, which translates to a nested for loop. Since the operations inside the nested for loop take O(1) time, the run time complexity of Algorithm 1 is polynomial.

To obtain the optimal trading policy (i.e. the sequence of trades at each step), we need to backtrack the OPT function. Therefore, it is vital to cache all elements of OPT in memory as they are computed. We can implement the OPT function as a linked list, which takes linear time to traverse through[1].

B. Optimal Trading Under Per-Order Fee

If the AMM charges a constant fee, c Y tokens, per order, the LP's y value is $y_f(n) + c \cdot n$. Though $y_f(n)$ decreases with n, $c \cdot n$ increases with n. Therefore, the optimization problem converts to finding the n such that $y_f(n) + c \cdot n$ is minimal. Such n can be determined by using the above algorithm along with a binary search.

Fig. 5 is an example in which the trader sells 10 X tokens in an AMM with the same initial parameters as Fig. 2, but the AMM charges the trader 1 Y token per trade. The x-axis is n, and the y-axis is the quantity $y_f(n) + c \cdot n$, which the trader

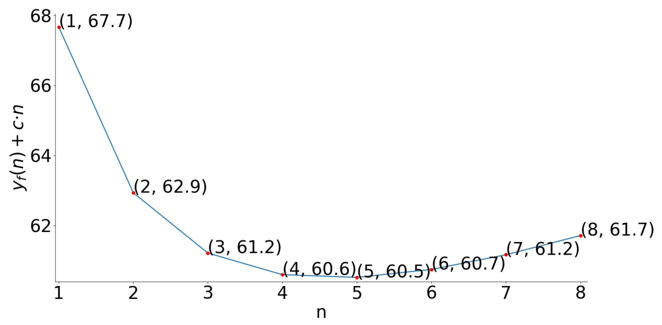


Fig. 5: Determining the optimal number of trades under a per-order fee model

would like to minimize, as $y_0 - (y_f(n) + c \cdot n)$ is the total Y tokens the trader gets in exchange for a given number of X tokens assuming the LP starts at some point (x_0, y_0). Observe that the local minimum occurs when making 5 trades. The policy of these 5 trades could be found by backtracking the OPT mapping at $y_f(5)$.

C. Optimal Trading Under Percentage Fee

In the situation where the exchange protocol charges a percentage fee, the optimal trading policy is still placing as many small trades as possible since the trader is not penalized for making large number of trades.

If the exchange charges a combination of percentage fee and per-order fee, we could again use the Algorithm 1 along with binary-search to find the optimal trading policy.

VI. Conclusions

This paper addresses the path-dependent trading issue of dynamic constant product curve AMMs by providing optimal trading policies for different common settings (zero transaction fees, per-order transaction fees, percentage transaction fees and combinations of these). We showed that dynamic constant product curves could in fact achieve arbitrarily close to zero slippage if transaction fees are negligible. For more complex settings, we established the methodology to generate the optimal trading policy in polynomial time. With these appurtenances, dynamic curve AMMs are better equipped to fulfill peer-to-peer trades, being not only arbitrage-proof, but also offering lower slippage than their static counterparts. This would make AMMs more desirable to both liquidity providers and traders. AMM engineers could adjust the transaction fees to strike a desired balance between monetization (to incentivize liquidity providers and platform stakeholders) and increasing the user base. Future directions of research could focus on optimal trading for more general dynamic AMMs including other functions and multi-dimensional AMMs that encompass more than two types of tokens.

[1]A software implementation of the dynamic programming-based algorithm for optimal trading can be found online at https://github.com/ANRGUSC/Optimal_Trading_Dynamic_AMM

References

[1] R. Hanson, "Logarithmic markets coring rules for modular combinatorial information aggregation," *The Journal of Prediction Markets*, vol. 1, no. 1, pp. 3–15, 2007.

[2] J. Peterson, J. Krug, M. Zoltu, A. K. Williams, and S. Alexander, "Augur: a decentralized oracle and prediction market platform," *arXiv preprint arXiv:1501.01042*, 2015.

[3] E. Hertzog, G. Benartzi, and G. Benartzi, "Bancor protocol," *Continuous Liquidity for Cryptographic Tokens through their Smart Contracts. Available online: https://storage. googleapis. com/website-bancor/2018/04/01ba8253-bancor_protocol_whitepaper_en. pdf (accessed on 6 June 2020)*, 2017.

[4] H. Adams, N. Zinsmeister, M. Salem, R. Keefer, and D. Robinson, "Uniswap v3 core," Tech. rep., Uniswap, Tech. Rep., 2021.

[5] M. Egorov, "Stableswap-efficient mechanism for stablecoin liquidity," *Retrieved Feb*, vol. 24, p. 2021, 2019.

[6] B. Krishnamachari, Q. Feng, and E. Grippo, "Dynamic curves for decentralized autonomous cryptocurrency exchanges," in *4th International Symposium on Foundations and Applications of Blockchain 2021 (FAB 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[7] G. Angeris and T. Chitra, "Improved price oracles: Constant function market makers," in *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, 2020, pp. 80–91.

[8] D. Engel and M. Herlihy, "Presentation and publication: Loss and slippage in networks of automated market makers," *arXiv preprint arXiv:2110.09872*, 2021.