Configurable arbitrage and slippage in automated market making systems

Scott Condie*

July 30, 2024

ABSTRACT

Automated market making systems have become increasingly popular in recent years. This paper studies the constant elastiticy pricing function, a generalization of the typical constant product pricing function. This generalization allows for oracle-based, arbitrage-free pricing, as well as configurable liquidity. Each of these possibilities come with trade-offs that are discussed, including the competitive environments in which this increased flexibility is desirable.

1 Introduction

Many smart contracts, like those available through the Ethereum cryptocurrency system, require that an exchange always have liquidity available to trade. This has led to the rise of automated market making (AMM) systems like those used by the Uniswap[1] cryptocurrency system. As of July 2024, Uniswap had daily trade volume of over $1.9 billion with total value locked (TVL) of $5.6 billion.[2] These AMM systems use pricing functions to determine the prices between two assets in a pool, as opposed to the more traditional limit order books used in many asset markets.

*Department of Economics, Brigham Young University. Email: ssc@byu.edu. Telephone: +1.801.422.5306.

[1]Uniswap (https://uniswap.org)

[2]Total Value Locked is the value of all assets (at current market prices) of assets committed to contracts in the system.

In limit order book trading environments, liquidity providers add limit orders to both the bid and ask side of the market that can be traded against by liquidity demanders. These limit orders specify the price at which the order can be executed against an incoming marketable order, as well as the quantity available for trade at that price. AMM systems differ in that liquidity providers add liquidity in one or both currencies to a pool, but do not specify[3], the price at which this liquidity will be traded. Liquidity demanders can trade one asset for the other at a price specified by a pricing function. The most commonly used pricing function is the constant product pricing function where the reserves of two assets with quantities x and y must satisfy the equation $xy = k$. The constant k is greater than zero. In this system, a trader who wants to acquire Δx units of asset x must pay Δy units of asset y where Δx and Δy satisfy $(x - \Delta x)(y + \Delta y) = k$. This expression implies a price of

$$\Delta y = \frac{k}{x - \Delta x} - y. \tag{1}$$

This AMM function has several properties of interest. First, it is simple. Second, since $\lim_{\Delta x \to x} \Delta y = \infty$ and $\lim_{\Delta y \to y} \Delta x = \infty$, prices increase without bound as the demand for the asset increases, ensuring that there will always be liquidity available for each asset.

Unlike traditional limit orderbooks, individuals add orders to the liquidity pool and in so doing, determine the terms of trade. In constant product markets, however, traders do not have the option of determining the price impact of trades in the market in a manner that is unconnected to the price. Price impact and price are determined jointly by the size of the asset pool and the pricing function.

In traditional limit order books price and price impact can be determined separately and asymmetrically. For example, at a moment in time the price impact of a marketable sell order for 100 shares of an asset can be larger or smaller than the price impact of a marketable buy

[3]At least in their canonical form. Uniswap v3 introduces some ability to specify price ranges.

order for 100 shares of the same asset. This flexibility does not exist for traditional constant product pricing rules.

This paper shows that exchanges wanting to alter the liquidity/price-impact properties of their pools can do so through generalizing the constant product pricing function. Section 2 defines analytical properties of a pricing function, while section 3 presents results on the generalization considered here. Section 4 applies this generalization to develop an arbitrage-free AMM system with flexible liquidity properties.

1.1 Literature

This paper uses the concept of pricing functions which are conceptually similar to scoring rules in prediction markets. Hanson (2007) studies the logarithmic market scoring rule (LMSR) in the framework of prediction markets and is one of the foundational works in the field of AMM pricing functions. It demonstrates that the LMSR (under some assumptions) can converge to true underlying probabilities. This paper doesn't concern prediction markets, but deals with generalized scoring rules similar to the LMSR.

Lekwijit & Sutivong (2018) studies the choice of parameters in the LMSR, showing (among other things) that the choice of the liquidity parameter can affect the speed with which the market converges to a true underlying value. This result is related to the discussion surrounding figure 3 which suggests that when market liquidity increases, more variation in the liquidity pools is required to alter prices. Agrawal et al. (2009) provides a unified framework for thinking about pricing functions in AMMs using convex optimization. They give conditions for myopic truthful bidding, among several other properties and study several risk measures in the framework they derive. Park (2021) discusses front running (sometimes referred to as "sandwich attacks") in automated markets.

2 General automated market making pricing functions

AMM pricing functions have the general form

$$f(x, y, \theta) = k \tag{2}$$

where x and y are quantities of the two assets available in the pool, θ represents a vector of parameters and k is a constant. In the constant product function commonly used, $f(x, y, \theta) = xy$.

For any pricing function, we define p_x (the price of x) to be

$$p_x = -\frac{dy}{dx} = \frac{\partial f/\partial x}{\partial f/\partial y}. \tag{3}$$

Note that this is the marginal price (in terms of y) of purchasing a small amount of asset x. Define the price impact of x as follows.

Definition 1. The *price impact* of the asset x is the function

$$d_x(x, y, \theta) = -\frac{\partial p_x}{\partial x}\frac{1}{p_x}. \tag{4}$$

The price impact of x, $d_x(x, y, \theta)$, is the semi-elasticity of the price with respect to a change in the pool quantity of asset x. In other words, it is the percent change in price that occurs for a small purchase (or sale) of asset x. An asset is more *liquid* than another if its price impact d_x is smaller.[4]

The traditional constant product pricing function takes the form

$$f(x, y) = xy \tag{5}$$

[4]The negative sign in this equation is present for interpretation. A purchase of asset x in the pool decreases the quantity of x available in the pool so the expression gives the price increase from a purchase of asset x.

and has price and price impact of

$$p_x = \frac{y}{x} \tag{6}$$

and

$$d_x = \frac{1}{x} \tag{7}$$

respectively.

3 Constant elasticity pricing functions

Consider the family of pricing functions given by

$$f(x, y, \theta) = \left(\alpha x^{1+\theta} + (1-\alpha)y^{1+\theta}\right)^{\frac{1}{1+\theta}}. \tag{8}$$

with $\theta \leq 0$.

Purchasing Δx units of asset x requires adding Δy units of asset y, where

$$\Delta y = \left(\frac{k^{1+\theta} - \alpha(x - \Delta x)^{1+\theta}}{1-\alpha}\right)^{\frac{1}{1+\theta}} - y \tag{9}$$

This pricing function has marginal price

$$p_x = \frac{\alpha}{1-\alpha}\left(\frac{x}{y}\right)^{\theta} \tag{10}$$

and price impact

$$d_x = -\theta\frac{\alpha}{1-\alpha}\left(\frac{x}{y}\right)^{\theta-1}\frac{1}{y}\frac{1}{p_x} = -\theta\frac{1}{x}. \tag{11}$$

For the constant elasticity pricing function the price and price sensitivity are parameterized through α and θ in a way that allows them to be set by market designers.[5]

[5]The constant elasticity pricing function gets its name from the fact that the elasticity of price with

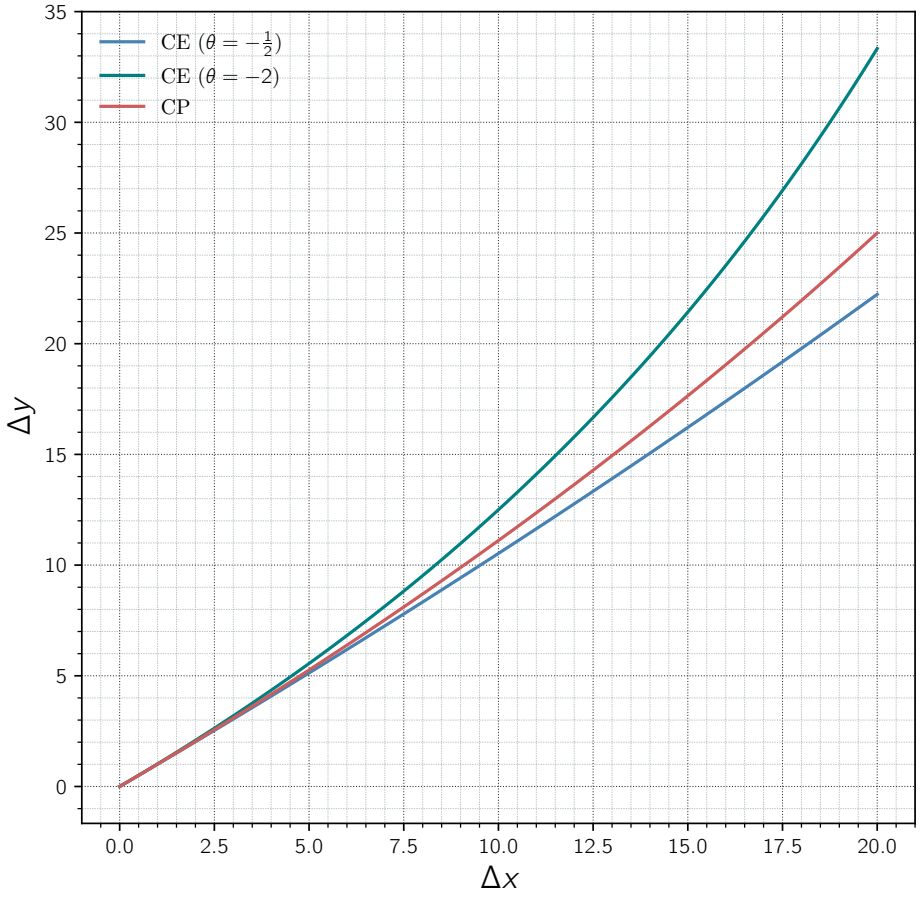

Figure 1: Constant elasticity pricing function (blue and green) as compared to a constant product pricing function (red). $\alpha = \frac{1}{2}, \theta \in \left\{-\frac{1}{2}, -2\right\}$

As can be seen, (8) approaches (5) in terms of its price and price sensitivity when $\alpha = \frac{1}{2}$ and $\theta \to -1$.

For values of θ satisfying $-1 < \theta < 0$, the constant elasticity pricing function given in (8) will be locally less price sensitive (and thus more liquid) than the constant product pricing function in (5) for equal pool sizes. This allows market designers to choose price sensitivity based on competitive concerns.

Figure 3 shows the realized price for a simulation where the same additions and withdrawals are made to markets with the constant pricing function and the CEPF with values of $\theta = -1/2$ and $\theta = -2$. The price impact of the CEPF for $\theta = -1/2$ is lower than the constant product pricing function for identical pool changes, while the price impact is larger for $\theta = -2$.

3.1 Arbitrage-free pricing

The constant elasticity pricing function allows market designers flexibility in both prices and the liquidity around prices, as given by price impact.

In automated market making situations where market designers want to provide to participants transactions at a price that replicates those in a pre-existing market, constant elasticity pricing functions can be used, together with a price oracle (e.g. the displayed price from the existing market). This will be a desirable property for market designers if providing liquidity at otherwise available market prices is advantageous, either because of legal restrictions[6], market norms, or competitive concerns.

To see how this can be done, let p_o be the reference price obtained from an oracle. If the value of the parameter α in the CEPF is set to

respect to quantity purchased $\epsilon = \frac{\partial p_x}{\partial x} \frac{x}{p_x}$ is constant at θ.

[6]Such as those similar to Regulation NMS in the United States that require that regulated market makers provide traders the nationally best available price for their trades.

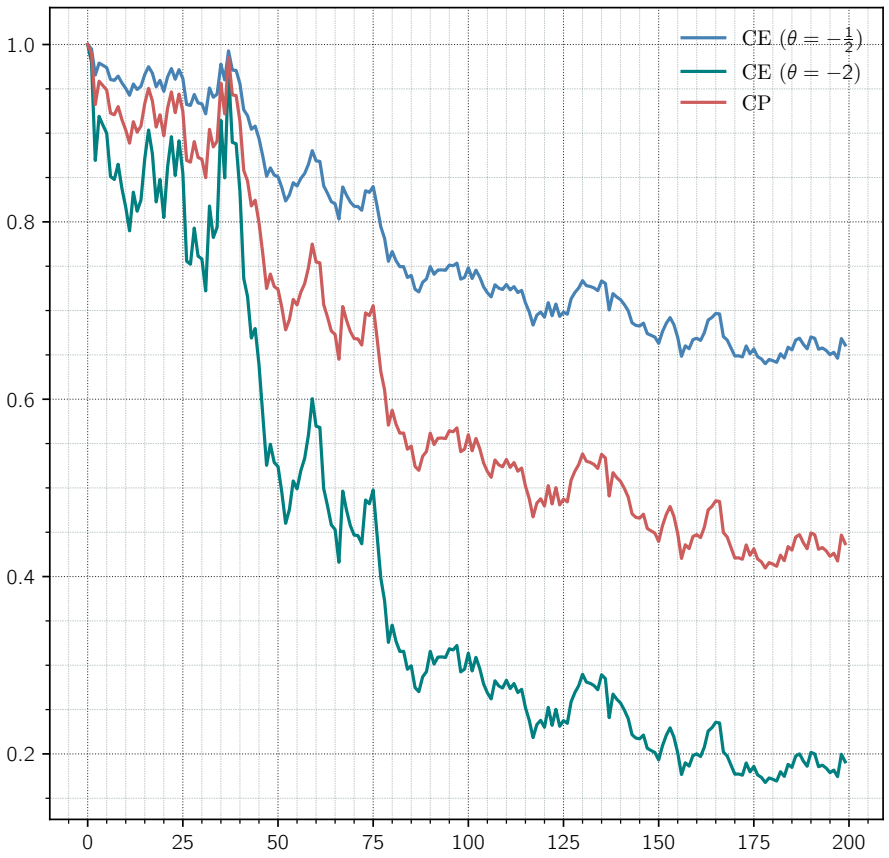

Figure 2: Constant Product (CP) and Constant Elasticity (CE) Price Impact in Identical Market Conditions. $\alpha = \frac{1}{2}, \theta \in \left\{-\frac{1}{2}, -2\right\}$.

$$\alpha = \frac{p_o}{p_o + \left(\frac{x}{y}\right)^{\theta}} \tag{6}$$

then the AMM price will always equal the oracle price. As is to be expected, providing arbitrage-free prices will have side-effects on other properties of the market. These will be discussed later.

As seen previously, the CEPF also allows for market designers to configure the liquidity or price impact of trades available at the current market price.

3.2 Configurable liquidity

The price impact for the CEPF given in equation (5) is

$$d_x = \frac{\theta}{x}. \tag{12}$$

Since $\theta < 0$, increasing θ increases liquidity by decreasing price impact and decreasing θ decreases the liquidity available in the market by increasing the price impact.

One implication of the decreased price impact, is that more market activity is required to maintain no-arbitrage for a given oracle price. Specifically, since

$$p_x^{CP} = \frac{y}{x} \tag{13}$$

and

$$p_x^{CE} = \frac{\alpha}{1-\alpha}\left(\frac{x}{y}\right)^{\theta}, \tag{14}$$

Consider a small *percent change* in the price of an external oracle around which arbitrageurs

will equate prices. The percent change[7] in price then satisfies

$$\%\Delta(p_o) = \%\Delta\left(p_x^{CE}\right) = \theta\left(\%\Delta x^{CE} - \%\Delta y^{CE}\right). \tag{15}$$

For a constant product pricing function this equation becomes

$$\%\Delta(p_o) = \%\Delta\left(p_x^{CP}\right) = \%\Delta y^{CP} - \%\Delta x^{CP}. \tag{16}$$

Together, these imply that for the same oracle-based price, maintaining price parity requires that

$$\frac{\%\Delta y^{CE} - \%\Delta x^{CE}}{\%\Delta y^{CP} - \%\Delta x^{CP}} = -\frac{1}{\theta} \tag{17}$$

which implies that for a given small change in the external price, there will need to be more change in the liquidity pool available on the CEPF market than on the CP market in order to maintain price parity.

As an example, for a CE market with $\theta = -1/2$, a 1 percent increase in the oracle price would require twice as much change in the ratio y/x in the CE market than in the CP market.

Figure 3 demonstrates this using data from the Binance Bitcoin/USD Tether exchange from March 9, 2022. The blue, green and red lines give the required BTC liquidity pool size in order to maintain price parity between a theoretical automated market and the price given in the Binance market (shown by the gray line with right axis).

[7]This uses the standard first-order approximation for percent changes.

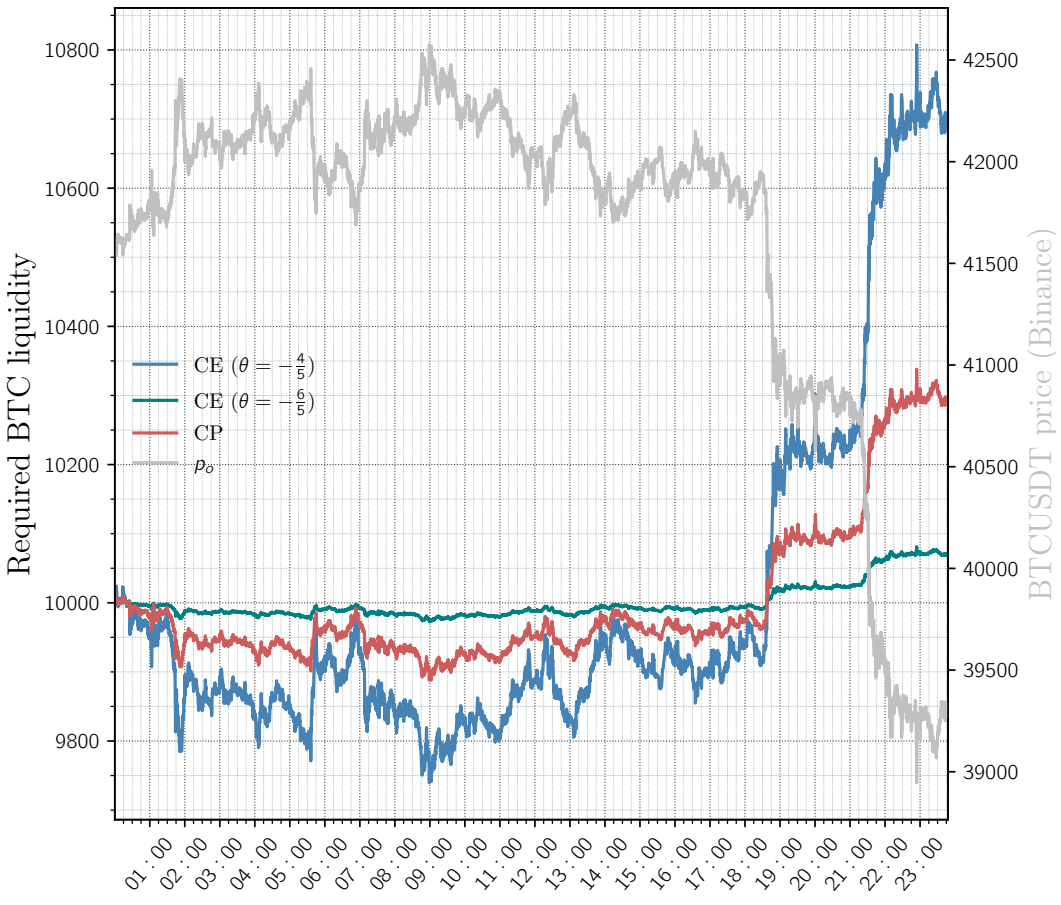

Figure 3: Implied BTC pool size under differing θ (Binance BTC/USDT–March 9, 2022)

4 Simultaneous matching of oracle prices and oracle slippage

Since constant elasticity pricing functions allow for oracle-based marginal prices with calibrated slippage, they can also be used to mimic the slippage found in a limit orderbook based oracle. To do this, the market designer chooses a benchmark trade quantity to be taken from the oracle's limit orderbook of Δx units of asset x. Let s_o be the oracle's orderbook slippage defined to be

$$s_o = \frac{p_o(\Delta x) - p_o}{p_o} \tag{18}$$

The price sensitivity given in equation (4) can be used as an approximation to the slippage found in the orderbook. For relatively small values of Δx,

$$s_{AMM} = d_x \Delta x = -\frac{\theta}{x}\Delta x \tag{7}$$

will be used to approximate the slippage in the oracle market. Setting $s_{AMM} = s_o$ so that

$$\theta = -\frac{s_o}{\frac{\Delta x}{x}} = -\frac{s_0 x}{\Delta x}. \tag{19}$$

matches slippage for small quantities in the automated market with slippage in the oracle.

Given this value for θ, α can be calculated given the price in (6), which implies a parameterization

$$\begin{aligned} \alpha &= \frac{p_o}{p_o + \left(\frac{x}{y}\right)^{\theta}} \\ \theta &= -\frac{s_0 x}{\Delta x} \end{aligned} \tag{8}$$

As an example, figure 4 shows the values of α and θ necessary to match the price and slippage given in the Binance data shown previously for a hypothetical AMM market. The

oracle's price and slippage are presented in the bottom two graphs. The simulations are drawn with no change in the liquidity pool over the day. Changes in the liquidity pool will change the implied parameter values, but not the process of calculating those parameters given in (7).

Looking at the oracle's slippage in the bottom graph, it can be seen that in order to purchase $\Delta x = 1$ units of BTC requires price slippage of between 0 and 0.00025

To investigate the precision of the parameterization of θ to match slippage, figure 5 shows the slippage present in the orderbook at a moment of time on the day in question, along with the slippage of the AMM price function where θ has been calibrated in the way described above for three candidate values of Δx.

This figure demonstrates that calibrating slippage to an orderbook can be sensitive to the Δx that is chosen to match AMM slippage to the orderbook slippage. Since slippage in traditional orderbooks is often discontinuous in Δx, the slippage parameterized by θ will change discontinuously as Δx spans discontinuous jumps in slippage in the orderbook.

5 Conclusion

Generalizing automated market making functions allows exchanges to alter the liquidity properties of their pools independent of the price. This flexibility allows pools to cater to particular user needs and market segments, as well as mitigate risk in certain circumstances.

Necessarily, this flexibility comes with added complexity. However, the specific functional forms described in this paper remain reasonable computationally and are easily verifiable by humans.

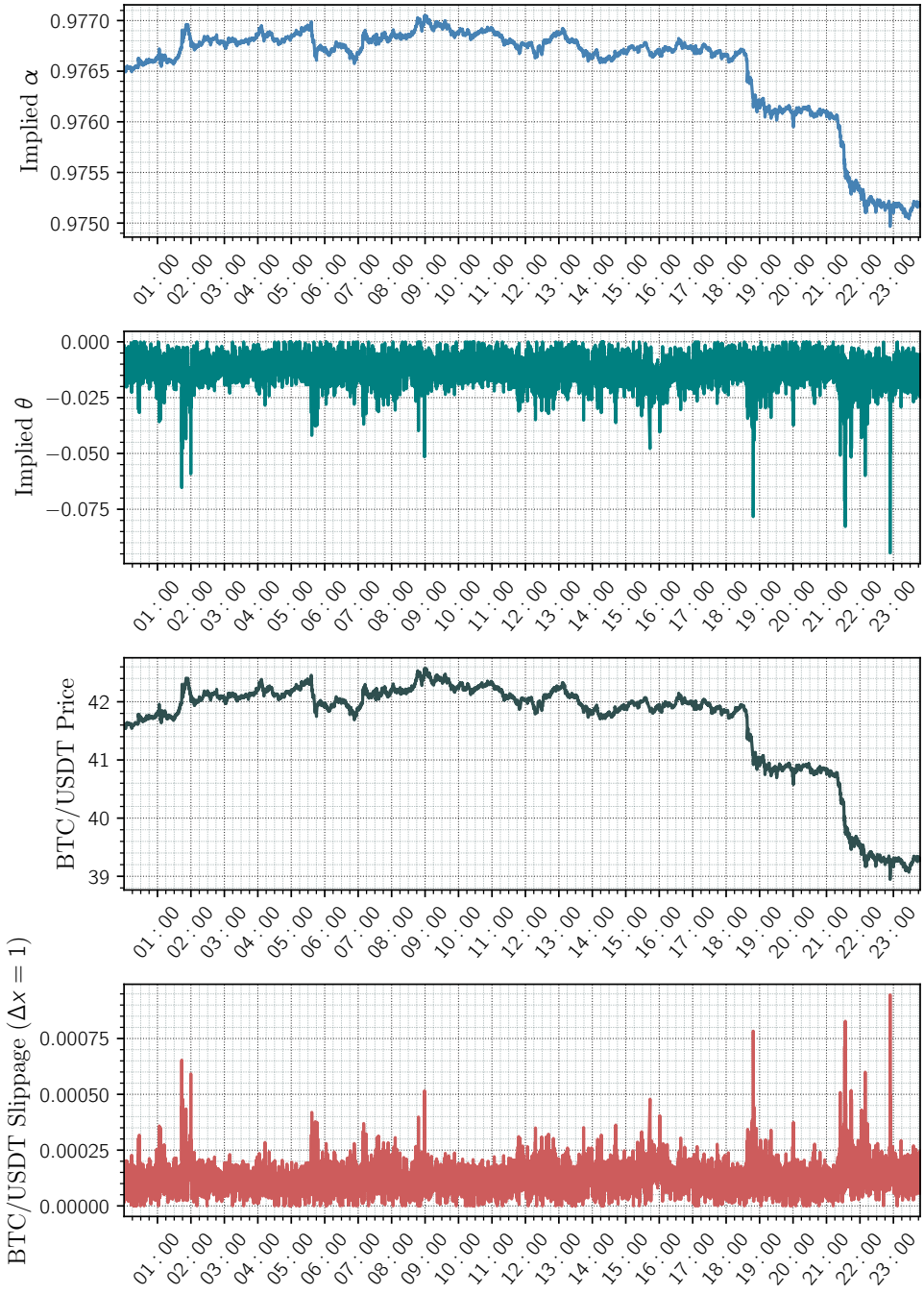

Figure 4: Values of α and θ required to match the liquidity and price slippage found in a traditional market. (Binance BTC/USDT–March 9, 2022.)

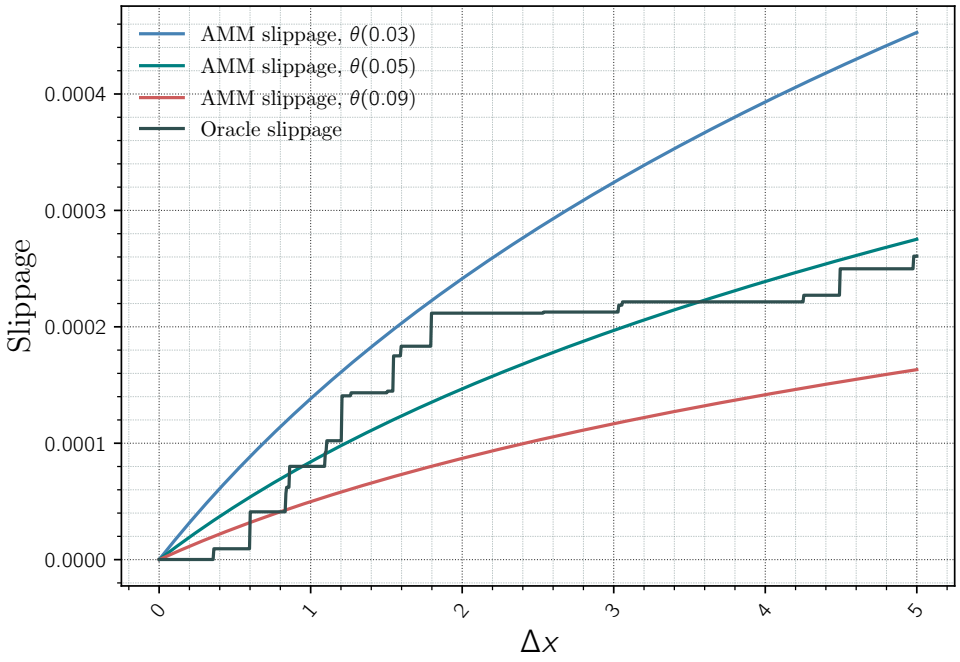

Figure 5: AMM slippage relative to oracle slippage for three parameterizations of price impact. Slippage is shown for θ corresponding to $\Delta x \in \{0.03, 0.05, 0.9\}$. Binance BTC/USDT–March 9, 2022: 10:00:00 a.m.

References

Agrawal, S., Delage, E., Peters, M., Wang, Z. & Ye, Y. (2009), A unified framework for dynamic pari-mutuel information market design, *in* 'Proceedings of the 10th ACM conference on Electronic commerce', pp. 255–264.

Hanson, R. (2007), 'Logarithmic market scoring rules for modular combinatorial information aggregation', *The Journal of Prediction Markets* **1**(1), 3–15.

Lekwijit, S. & Sutivong, D. (2018), 'Optimizing the liquidity parameter of logarithmic market scoring rules prediction markets', *Journal of Modelling in Management* .

Park, A. (2021), 'The conceptual flaws of constant product automated market making', *Available at SSRN 3805750* .