An analysis of Uniswap markets

Guillermo Angeris[1]

Hsien-Tang Kao[2]

[1]angeris@stanford.edu

[2]{hsien-tang, rei, tarun}@gauntlet.network

Rei Chiang[2]

Charlie Noyes[3]

[3]charlie@paradigm.xyz

Tarun Chitra[2]

(November 2019)

Abstract

Uniswap—and other constant product markets—appear to work well in practice despite their simplicity. In this paper, we give a simple formal analysis of constant product markets and their generalizations, showing that, under some common conditions, these markets must closely track the reference market price. We also show that Uniswap satisfies many other desirable properties and numerically demonstrate, via a large-scale agent-based simulation, that Uniswap is stable under a wide range of market conditions.

1 Introduction

Smart contract systems such as Ethereum [1] and Tezos [2] have allowed for the design and implementation of decentralized versions of traditional financial primitives. The use of these primitives has grown dramatically in 2019: in January, less than $20 million of Ethereum-based assets were utilized in these systems, a number which increased to $750 million by December [3]. Decentralized financial primitives allow for censorship-resistant participation in a number of digital markets, expanding the reach of lending [4], stable assets [5, 6], and exchanges [7, 8, 9] past the conventional financial world. In particular, a secure decentralized exchange for cryptocurrencies has been desired almost since the advent of Bitcoin as centralized exchanges, such as Mt. Gox [10], Quadriga [11], and Bitfinex [12], have suffered catastrophic losses aggregating to billions of dollars' worth of depositors' funds.

Historically, many different decentralized exchanges (DEXs) have been proposed using different market maker mechanisms, ranging from classic order book mechanisms [8] to other, more complicated approaches with particular bonding curves [9]. Yet, a simple but surprisingly effective market maker appears to be the constant product market maker used by Uniswap [7], likely the first and possibly the most popular implementation. These markets provide a simple approach for trading between pairs of coins in a decentralized fashion, and have, in recent years, become a popular (and practically useful) alternative to other types of DEXs.

Additionally, other protocols (such as Celo [5]) have used the idea of constant-product markets as a decentralized price oracle — a contract that can be queried to find the relative price between two coins of interest. These protocols make use the fact that, if the price indicated by the oracle differs from the true market price of the traded coins, an arbitrageur can always make a profit by trading with this oracle, implying that the price indicated by this oracle is likely close to the reference market price.

Comparison with other DEXs.

While order book mechanisms are the dominant medium of exchange of electronic assets in traditional finance [13], they are challenging to use within a smart contract environment. The size of the state needed by an order book to represent the set of outstanding orders (*e.g.*, passive liquidity) is large and extremely costly in the smart contract environment, where users must pay for space and compute power utilized [1]. Moreover, the matching logic for order books is often complicated as it must often support several different order types (such as icebergs, good-till-cancel, and stop-limit orders [13]).

In order to avoid the costly on-chain execution costs (paid to miners/validators of the smart contract by agents executing trades) a variety of designs for decentralized exchanges use the blockchain underlying a smart contract for settlement, while executing trades off-chain [8, 14]. These exchanges, however, have a number of drawbacks. First, the complicated interaction between off-chain and on-chain mechanisms, coupled with the transaction ordering ambiguity inherent in blockchain-based systems, allows for front-running, which has been observed in practice [15]. Second, keeping the order book state in the hands of multiple participants (such as 'relayers,' in 0x parlance [8]) often leads to stale orders and latency arbitrage that is many orders of magnitude worse than that of high-frequency trading. Finally, these exchanges have a more complicated threat model than that of a smart contract which does not have to interact with an exogenous, non-blockchain system to determine state transitions. Due to this, users must often take extra security precautions when trading or potentially use complicated exit games to release their funds.

Automated market makers.

On the other hand, automated market makers (AMMs) have been studied extensively in algorithmic game theory, beginning with Hanson's logarithmic market scoring rule (LMSR) [16], of-

ten used in practice as an AMM for prediction markets. Such AMMs are constructed by first having liquidity providers deposit assets in some fixed ratio to specify an initial distribution of beliefs over possible outcomes. An AMM then provides a *scoring rule* which specifies the cost of changing the distribution of beliefs from its current state to a new, desired state. This scoring rule incentivizes traders to report their true belief such that the expected value of adjusting the distribution is positive. Since the state of the exchange depends only on the total amount of quantities deposited, the corresponding storage requirements are much smaller than those of traditional exchanges. Additionally, pricing a trade requires only a single function evaluation, as opposed to more complicated matching algorithms, as in the case of order books.

Bonding curves.

In general, LMSRs (and similar AMMs) are designed to predict the outcome of some set of (disjoint) events rather than predict a specific price. In many cases, the proposed mechanism cannot be directly used to price arbitrary assets without requiring a large state space and often suffers from other practical issues such as attracting liquidity [17].

To solve this problem, early Ethereum-based AMMs such as Bancor [9] moved to a second model for pricing assets: in this model, the function specifies the cost of an asset based on the total available supply (as, for example, its tokens are minted or burned), rather than specifying the cost of changing a given distribution. The function itself is called a bonding curve, and the resulting equilibrium price of the asset is then equal to the market price under certain conditions.

Uniswap.

Another possible model for pricing assets, first introduced by Uniswap [7], does not require the ability to change the supply of an asset in order to measure its price. Rather, an AMM holds some quantity of assets whose relative price we wish to measure in its reserves. The AMM then specifies a pricing function, which maps the quantities of the assets in reserves to their marginal price (with respect to any numéraire). Agents are then allowed to trade with this contract at the price specified by the pricing function, and this price is continually updated as its reserves change after each trade. For example, constant product markets such as Uniswap are specific cases in which the pricing function is exactly equal to the ratio of the reserves available to the contract, when no trading fees are taken (§2.1).

While Uniswap, and its associated class of AMMs, is similar in spirit to bonding curve-based AMMs, we will distinguish them as a separate class of AMMs with a fairly distinct range of applicability. We will focus only on Uniswap-like AMMs—the constant product and constant mean AMMs—and leave the discussion of bonding curves and their theoretical properties for future research.

Summary.

In this paper, we present optimal arbitrage actions and bounds under some simple reference market models for Uniswap's AMM model, which we call the *constant product markets*, showing that Uniswap must closely track the reference market price under common market conditions. We also show that a recent generalization of constant product markets, the *constant mean markets*, first proposed in [18], have nearly identical theoretical properties and may be of future interest. Finally, we run a large-scale simulation of agents interacting with the Uniswap contract over a wide range of market parameters, suggesting that the system may be stable under a variety of market conditions. These results help us conclude that Uniswap serves as a censorship resistant price oracle for smart contracts, provided that there exists an external reference market with sufficient liquidity.

2 Constant product markets

A *constant product market* [7] is a market for trading coins of type α for coins of type β (and vice versa). This market has reserves $R_\alpha > 0$ and $R_\beta > 0$, constant product $k = R_\alpha R_\beta$, and percentage fee $(1 - \gamma)$. A transaction in this market, trading $\Delta_\beta > 0$ coins β for $\Delta_\alpha > 0$ coins α, must satisfy

$$(R_\alpha - \Delta_\alpha)(R_\beta + \gamma\Delta_\beta) = k. \tag{1}$$

After each transaction, the reserves are updated in the following way: $R_\alpha \mapsto R_\alpha - \Delta_\alpha$, $R_\beta \mapsto R_\beta + \Delta_\beta$, and $k \mapsto (R_\alpha - \Delta_\alpha)(R_\beta + \Delta_\beta)$. We will always require that $R_\alpha, R_\beta > 0$, such that any trade that results in a nonpositive reserve is never fulfilled (equivalently, we say that such a trade has infinite cost).

The name 'constant product market' comes from the fact that, when the fee is zero (*i.e.*, $\gamma = 1$), any trade Δ_β to Δ_α must change the reserves in such a way that the product $R_\alpha R_\beta$ remains equal to the constant k.

In this section, we derive bounds on the marginal price of the market relative to a reference market and show that this market maker has other desirable properties.

2.1 Optimal arbitrage in Uniswap

In the optimal arbitrage problem, we have two coins, α and β, which we can trade either with a reference market or a Uniswap contract. In this problem, we seek to maximize the profit made from trading, say, some amount of loaned coin, Δ_β of coin β to some amount Δ_α of coin α via the Uniswap market. We then trade back the received Δ_α for Δ_β' and pay back the loan Δ_β to receive profit $\Delta_\beta' - \Delta_\beta$.

If our profit is positive (that is, if $\Delta_\beta' - \Delta_\beta > 0$), then we say that there is an *arbitrage opportunity*, since we have made money 'for free' (*i.e.*, by only trading coins within different markets) with-

out taking on any risk. The optimal arbitrage problem then asks: what is the maximum profit that can be made by this scheme?

In the infinitely liquid market case, *i.e.*, in the case that $\Delta_\beta^{'} = m_p \Delta_\alpha$ (where m_p is the reference market price of coin α), we can phrase the optimal arbitrage problem as the following optimization problem:

$$\begin{array}{ll} \textit{maximize} & m_p \Delta_\alpha - \Delta_\beta \\ \textit{subject to} & \Delta_\alpha, \Delta_\beta \geq 0 \\ (R_\alpha - \Delta_\alpha)(R_\beta + \gamma \Delta_\beta) = k, & \end{array} \tag{2}$$

with optimization variables $\Delta_\alpha \in \mathbf{R}$ and $\Delta_\beta \in \mathbf{R}$, constrained to be nonnegative. Here, $(1-\gamma)$ is the Uniswap exchange fee, and the constraint is the definition of a constant product market (1).

When written in this form, problem (2) is not obviously convex, though we show in appendix A that it can easily be written in a convex form and then derive an analytical form for the optimal trade amounts $\Delta_\alpha^\star$ and $\Delta_\beta^\star$.

No-arbitrage conditions.

Assuming that the no-arbitrage condition is satisfied (which often approximately holds in practice; see [19, §1.2]), we can show that the Uniswap market price deviates from the market price by at most a factor of γ.

The marginal price of coin α in Uniswap is defined as the price of an infinitesimally small trade. This price can be found by differentiating the constant-product market formula:

$$\frac{d}{d\Delta_\alpha}\Big((R_\alpha - \Delta_\alpha)(R_\beta + \gamma\Delta_\beta)\Big) = 0 \Longrightarrow \left.\frac{d\Delta_\beta}{d\Delta_\alpha}\right|_{\Delta_\alpha = 0} = \frac{1}{\gamma}\frac{R_\beta}{R_\alpha} = \gamma^{-1} m_u,$$

where we have written $m_u = R_\beta / R_\alpha$ for the Uniswap price of coin α without the fee.

We can always make a nonzero profit if the Uniswap marginal price of α, $\gamma^{-1} m_u$ is smaller than the market marginal price m_p, by performing a small enough trade. Assuming there is no arbitrage, this means we must have

$$m_u \geq \gamma m_p.$$

Similarly, by swapping α for β and combining the resulting statements we get the following bounds on the Uniswap market price, relative to the true market price, m_p:

$$\gamma m_p \leq m_u \leq \gamma^{-1} m_p, \tag{3}$$

assuming no-arbitrage conditions.

This suggests that, in practice, the larger the trade fees are, the larger the gap between the true market price and the Uniswap market price may be. For example, in the case that the trade fee $\tau = 1 - \gamma$ is small, bound (3) is approximately equivalent to

$$(1 - \tau) m_p \leq m_u \leq (1 + \tau) m_p.$$

While we derived these conditions using a simple argument, we can also easily derive them by analyzing problem (2) and noting that its optimal value is nonzero if, and only if, there is an arbitrage opportunity. For more details, see appendix A.

2.2 Extensions of the optimal arbitrage problem

There are several natural extensions to the optimal problem (2), which retain most of its useful properties.

Risk models.

There are many factors which could potentially cause an agent to fail to close an arbitrage opportunity, including noisy information, front-running [15], and delay in trades—the latter of which is quite common in distributed platforms. In particular, it may not be desirable for an agent to perform a trade with the exact solution of problem (2).

In this case, we can add an additional term penalizing large trades: in other words, we can assign some 'cost of risk' which can be any convex function $f : \mathbf{R}_+^2 \to \mathbf{R}$ which is nondecreasing in its second argument. This gives a problem of the form:

$$\begin{array}{ll} \underset{\Delta_\alpha, \Delta_\beta}{\text{maximize}} & m_p \Delta_\alpha - \Delta_\beta - f(\Delta_\alpha, \Delta_\beta) \\ \textit{subject to} & (R_\alpha - \Delta_\alpha)(R_\beta + \gamma \Delta_\beta) = k \\ & \Delta_\alpha, \Delta_\beta \geq 0. \end{array} \tag{4}$$

It should be noted that problem (4) can be written in a convex way and is the problem we use to simulate arbitrageurs in our agent simulation (see §4.2 for more details).

One example of a risk function f could be a model of the resulting changes in market price due to the arbitrage trade. A common (and simple) model for the marginal market price $m : \mathbf{R}_+ \to \mathbf{R}_+$ after a trade of size $0 \leq \Delta_\alpha \leq \eta^{-1/\xi}$ is that the resulting price is

$$m(\Delta_\alpha) = m_p \left(1 - \eta \Delta_\alpha^\xi\right),$$

with any $\eta \geq 0$, $\xi > 0$. The resulting risk function, $f(\Delta_\alpha) = \int_0^{\Delta_\alpha} (m_p - m(t))\, dt$, is convex. See appendix A.1 for more details and extensions.

Additionally, in this model, unlike in problem (4), an arbitrageur is not guaranteed to reach the no-arbitrage condition in a single round if the penalty function does not reflect the true underlying market movement.

2.3 Other conditions

In this section, we show a few basic and useful properties which Uniswap (and other constant-product markets) satisfy.

Increasing product constant.

For every trade, the product constant k is nondecreasing (and strictly increasing if $\gamma < 1$). Let $R_\alpha^t > 0$, $R_\beta^t > 0$, and $k^t > 0$ be the reserve of α, β, and the constant product at trade t. If trade $t+1$ sells Δ_β coins β for Δ_α coins α, we have (by definition, see (1)):

$$k^t = (R_\alpha^t - \Delta_\alpha)(R_\beta^t + \gamma\Delta_\beta) \leq (R_\alpha^t - \Delta_\alpha)(R_\beta^t + \Delta_\beta) = k^{t+1}, \tag{5}$$

where inequality always holds strictly except when $\gamma = 1$. So, if $\gamma < 1$, we have that $k^t < k^{t+1}$, always.

Splitting trades is more expensive.

Whenever $\gamma < 1$, any agent agent trading some amount $\Delta_\alpha > 0$ for some output Δ_β and immediately trading another amount $\Delta_\alpha' > 0$ for Δ_β' will always have smaller output than if the agent had traded $\Delta_\alpha + \Delta_\alpha'$ for some output $\Delta_\beta^{\text{tot}}$ all at once (this property is also sometimes called *path-dependence* [17]). This can easily be proven, though it is mostly an exercise in algebra. We outline the steps in appendix D.

This fact implies that, in the case where the reference market is infinitely liquid (as in §2.1), an arbitrage agent who considers a strategy several steps into the future cannot do better than simply solving (2) and executing the corresponding trade.

No-depletion property.

It turns out to be impossible to fully deplete Uniswap of all coins only by trading α and β, even if the attacker has an unbounded amount of coins. We can easily show that the total reserve is always bounded from below. This follows immediately from applying the arithmetic-geometric mean inequality [20, §3.1.9]:

$$2\sqrt{k} = 2\sqrt{R_\alpha R_\beta} \leq R_\alpha + R_\beta.$$

Since this is true and k is nondecreasing (and usually strictly increasing) after each trade, then the total number of coins in Uniswap can never decrease below the twice the square root of the initial product by only trading between coins α and β.

Increasing liquidity with increasing reserves.

Intuitively, the larger the amount of reserves, the less any one trade will cost. The marginal cost change of Uniswap (the negative of the infinitesimal price change of the Uniswap market after an infinitesimal trade) can be computed by differentiating (1) twice and is given by

$$\left.\frac{d^2\Delta_\beta}{d\Delta_\alpha^2}\right|_{\Delta_\alpha = 0} = \frac{2m_u}{\gamma R_\alpha}, \tag{6}$$

where $m_u = R_\beta / R_\alpha$ is the Uniswap price without fees. Note that (6) is strictly decreasing as R_β increases (assuming m_u, the Uniswap price, stays constant).

We can similarly show this property directly by assuming that we have two markets, one with strictly larger reserves $R_\alpha^{'} > R_\alpha$, and both with price

$$\frac{R_\beta}{R_\alpha} = m_u = \frac{R_\beta^{'}}{R_\alpha^{'}}.$$

In this case, assuming we trade Δ_α coins with both markets, we can show that the markets will have a price gap of

$$\Delta_\beta^{'} - \Delta_\beta = m_u \gamma^{-1} \Delta_\alpha^2 \left(\frac{1}{R_\alpha} - \frac{1}{R_\alpha^{'}}\right) + O\left(\frac{\Delta_\alpha^2}{R_\alpha^2}\right). \tag{7}$$

This price gap is always positive in the case that $R_\alpha^{'} > R_\alpha$ and this construction gives essentially a more precise version of (6). The derivation can be found in appendix B.

Cost of manipulation.

In the no-fee case with an infinitely liquid reference market, the cost of manipulating the price of a constant product market can be bounded from below.

If the attacker wishes to manipulate the constant product market to be $(1 + \varepsilon)m_p$, where m_p is the true market price and $\varepsilon > 0$ is some desired constant, then the attacker requires an amount of, at least

$$C(\varepsilon) \geq K R_\beta \min\{\varepsilon^2, \sqrt{\varepsilon}\ \}, \tag{8}$$

coin β for each period (here, a period could be, *e.g.*, the time taken for a block to be mined) with $K > 0$ a universal constant.

This result implies that the cost of manipulation scales linearly with the reserve amounts, marking the importance of having large reserve pools. Additionally, the result extends immediately (as a lower bound) to the case with fees. For a derivation of (8) and an explicit bound on K, see appendix E.

Liquidity provider returns.

In the no fee case (*i.e.*, when $\gamma = 1$), we can easily compute the portfolio value of the Uniswap contract (and, correspondingly, any liquidity provider). Let R_α^t, R_β^t, and $m_p^t \in \mathbf{R}_+$ be the reserves for coin α, coin β, and the market price of coin α, respectively, at each time $t = 1, \ldots, T$.

In the no fee case, the no-arbitrage bounds (3) imply that, $m_p^t = R_\beta^t / R_\alpha^t$, while the definition of a constant product market (1) implies that $R_\alpha^t R_\beta^t = k$ for all t. Combining both statements:

$$R_\beta^t = \sqrt{k m_p^t}.$$

We can use this expression to write a simple form for the relative return for Uniswap between time $t-1$ to t, given by

$$\delta^t = \frac{m_p^t R_\alpha^t + R_\beta^t}{m_p^{t-1} R_\alpha^{t-1} + R_\beta^{t-1}} = \frac{R_\beta^t}{R_\beta^{t-1}} = \sqrt{\frac{m_p^t}{m_p^{t-1}}}.$$

This implies that the total relative gain for a Uniswap contract is

$$\delta = \prod_{t=2}^{T} \delta_t = \sqrt{\frac{m_p^T}{m_p^1}}, \tag{9}$$

and the total portfolio value is

$$P_V = (m_p^1 R_\alpha^1 + R_\beta^1)\delta = 2\sqrt{k m_p^T}. \tag{10}$$

Since δ^t depends only on the relative ratios of R_α and R_β, the result holds even when liquidity providers add tokens or remove tokens from the reserves—although in practice, as suggested by (6), the price is likely to be less stable in the case of smaller reserves. See appendix C for further discussion.

2.4 Discussion

Because the arbitrage problem (2), and its risky variant, problem (4), are convex, we suspect that the no-arbitrage assumption is very likely to be met in practice. The convexity of this prob-

lem additionally implies that arbitrage can be efficiently performed even across several Uniswap markets (see appendix A for more details).

As shown in (3), we analytically observe the effect that the Uniswap market fee has on the price of the market: the bounds guaranteed by the no-arbitrage assumption become looser. In other words, as the fee increases (or, equivalently, γ decreases), we expect the Uniswap market to deviate further from the price of a given reference market.

Result (8) implies that, when the reserves are large, the cost of manipulation is generally expensive for all but the smallest of changes—though, due to the quadratic scaling when ε is small, we can expect small changes to the price to be relatively inexpensive. This implies that protocols which depend on Uniswap (or other constant product markets) as a price oracle should not be extremely sensitive to small price fluctuations; otherwise, it may be possible for an attacker with moderate resources to exploit this for their own gain.

These properties, combined with results (5) and (7), suggest that constant product markets are likely to be robust in the the practical setting where the number of tokens in reserve is large and the number of trades is also large.

3 Constant mean markets

A *constant mean market* is a market which generalizes constant product markets. First introduced in [18], constant mean markets satisfy the following equation in the absence of fees:

$$\prod_{i=1}^{n} R_i^{w_i} = k, \tag{11}$$

where R_i are the reserves of coin $i = 1, \ldots, n$, $w \in \mathbf{R}_+^n$ are the weights associated with each coin, and $k \in \mathbf{R}_+$ is the constant product. In this case, the weights all satisfy $w \geq 0$ with $\mathbf{1}^T w = 1$. In other words, in the absence of fees, constant product markets ensure that the product of its reserves stays constant, while constant mean markets ensure that the weighted geometric mean of the reserves, R_i for $i = 1, \ldots, n$, stays constant.

Similar to constant product markets, trading Δ_j amount of coin j for some amount $\Lambda_{j\ell}$ of a distinct coin $\ell \neq j$ should always satisfy the equation,

$$\left(\prod_{\substack{i=1 \\ i \neq j,\ell}}^{n} R_i^{w_i} \right) \left(R_j + \gamma_j \Delta_j \right)^{w_j} \left(R_\ell - \Lambda_{j\ell} \right)^{w_\ell} = k,$$

where $(1-\gamma_j)$ is the percentage fee associated with trading coin j. The corresponding reserves, R_j and R_ℓ are updated as in §2, as is the mean constant, k.

Note that constant product markets are a special case of a constant mean market where $n = 2$ and we have $w_1 = w_2 = 1/2$ and $\gamma_1 = \gamma_2$, with the constant product k of (1) replaced with its square root.

3.1 Optimal arbitrage problem

We can write the optimal arbitrage problem for constant mean markets in the following way:

$$\begin{aligned} &\textit{maximize} && \sum_{i=1}^{n}\left(\sum_{j=1}^{n}\Lambda_{ij}m_j^p - \Delta_i m_i^p\right) \\ &\textit{subject to} && \prod_{i=1}^{n}\left(R_i + \gamma_i\Delta_i - \sum_{j=1}^{n}\Lambda_{ij}\right)^{w_i} = k \\ &\Delta_i \geq 0, \quad i = 1,\ldots,n \\ &\Lambda_{ij} \geq 0, \quad i,j = 1,\ldots,n, \end{aligned} \tag{12}$$

where we will assume that Λ_{ii} is constrained to be zero for notational convenience.[4]

[4] If $\gamma_i < 1$, it is not hard to prove that any locally optimal point will have $\Lambda_{ii} = 0$.

The variables in this optimization problem are the amount $\Delta \in \mathbf{R}^n$ whose entries state how much of each coin to purchase from the external market, while i,jth entry of $\Lambda \in \mathbf{R}^{n\times n}$ states how much of coin i to trade for coin j.

Since the weighted geometric mean is a concave function that is increasing in all of its arguments [20, §3.1.5], it turns out that the equality constraint in problem (12) can be relaxed to an inequality constraint to get an equivalent, but convex, optimal arbitrage problem [20, §3.2.4]. See appendix F for more details.

3.2 Extensions and properties

Surprisingly, almost all conditions and properties (except the no-arbitrage condition of (3)) that hold for constant product markets also hold for constant mean markets in a similar form. We describe a few cases below.

Extensions.

All of the same extensions given in section §2.2 hold for problem (12). More specifically, for any convex function $f : \mathbf{R}_+^{n^2} \times \mathbf{R}_+^n \to \mathbf{R}$, which is increasing in its first n^2 arguments and decreasing in

the remaining n arguments, the following convex relaxation of (12) is an equivalent (but convex) problem with the additional penalty term, f:

$$\begin{aligned} &\textit{maximize} && \sum_{i=1}^{n}\left(\sum_{j=1}^{n}\Lambda_{ij}m_j^p - \Delta_i m_i^p\right) - f(\Lambda,\Delta) \\ &\textit{subject to} && \prod_{i=1}^{n}\left(R_i + \gamma_i\Delta_i - \sum_{j=1}^{n}\Lambda_{ij}\right)^{w_i} \geq k \\ &\Delta_i \geq 0, \quad i = 1,\ldots,n \\ &\Lambda_{ij} \geq 0, \quad i,j = 1,\ldots,n, \end{aligned} \tag{13}$$

with the same variables, $\Delta \in \mathbf{R}^n$ and $\Lambda \in \mathbf{R}^{n\times n}$, as problem (12).

Properties.

Additionally, some of the same properties given in §2.3, hold essentially in their exact form for constant mean markets: all instances will have an nondecreasing product constant and satisfy the corresponding no-depletion property. The notion of increasing liquidity with increasing reserves also holds and can be easily derived from [18, Out-Given-In].

No-arbitrage conditions.

It is, in general, not clear that there exists a closed-form solution for the no-arbitrage conditions specified in §2.1. It is possible to give simple necessary (but not sufficient) conditions on each pair of coins found in a given constant-product market via a similar argument to the one in §2.1, but we expect the general conditions are more complicated.

3.3 Discussion

We present these results for constant mean markets as they carry over nearly immediately from those given in §2. Though we suspect that markets which satisfy the constant mean property, such as Balancer [18]—where the idea originated—may become more important in the near future, we focus on the specific case of Uniswap (and, more generally, constant product markets) as other protocols such as Celo [5] heavily depend on the robustness of this particular market maker mechanism.

4 Agent-based simulation of Uniswap markets

While the properties presented in §2 lead us to believe that the Uniswap market is likely well-behaved under most scenarios, it is hard to make stronger claims about the robustness of the Uniswap market mechanism without making assumptions that are unlikely to be realistic.

To verify this experimentally, we created an agent-based simulation by using the Gauntlet DSL to specify how several types of agents interact with the current Uniswap contract[5]

[5]As of this time, the tested contract was based on commit `c10c08d` in the Uniswap Github repository, `https://github.com/Uniswap/contracts-vyper`.

on a simulated Ethereum blockchain. The simulation environment interacts with the Uniswap contract deployed on a simulated blockchain via Python bindings. The environment allows for configuration of the network's initial conditions, including distributions of agent behaviors and agent-specific parameters.

The simulation is run for a pre-defined number of time steps. For each simulated time step, environment state variables are updated based on the state of the on-chain contracts. We additionally evaluate policies for adding new agents to the environment. We evaluate the utility of an action for a given agent, and execute agent actions that have positive utility by submitting the corresponding transaction to the blockchain.

4.1 Simulation markets

In the current simulation, we have two possible markets for agents to trade with: one is given by the Uniswap contract and the other is given by a simple stochastic market model.

Uniswap market.

When interacting with the Uniswap contract, an agent has a few possible actions they can perform: the agent can either (a) trade coins α for β (and vice versa) subject to the constant product market equation (1), or (b) add or remove liquidity.

In the latter case, an agent is able to add, say, Δ_β amount of coin β to reserve R_β and is required to also add $R_\alpha \Delta_\beta$ / R_β of coin α to reserve R_α. The agent is then awarded

$$\Delta_{\text{UNI}} = \frac{\Delta_\beta}{R_\beta} R_{\text{UNI}},$$

where R_{UNI} is the total amount of outstanding UNI coins given by the contract. The agent can also similarly remove liquidity by burning Δ_{UNI} coins. The contract then gives the agent

$$\Delta_\alpha = \frac{\Delta_{\text{UNI}}}{R_{\text{UNI}}} R_\alpha, \quad \Delta_\beta = \frac{\Delta_{\text{UNI}}}{R_{\text{UNI}}} R_\beta,$$

and burns the given Δ_{UNI} coins.

This market mechanism allows the agents who purchase these UNI coins, often called liquidity providers, to earn a profit given by the exchange fee, assuming the market price stays constant.

Additionally, it provides a mechanism for adding and removing to reserves, thus making the Uniswap market more liquid, as shown in section §2.3.

For the remainder of this section, we will assume the price of a UNI token is exactly given by the market prices of the equivalent amount of coins α and β that would be received by burning the token. Additionally, the Uniswap contract defines $\gamma = .997$, which is the value we use from here on out.

Reference market.

The reference market follows a simple power law model where the price m_p of some coin α, is updated in the following way:

$$m_p \mapsto m_p + \kappa \Delta_\alpha^{1+\xi},$$

where $\kappa \geq 0$ and $\xi \geq 0$ are given in the problem data. While it is possible for an arbitrageur to solve the arbitrage problem exactly (as it is a special case of (4)), we choose a simpler risk model for the arbitrageur as the true market price model is not known in practice.

We additionally update the market price every time step (after all agents have completed their actions) in the following way:

$$m_p \mapsto m_p \cdot e^{\sigma X + \mu}.$$

Here $X \sim \mathcal{N}(0,1)$ is drawn from a normal distribution and $\mu, \sigma \in \mathbf{R}$ represent the mean returns and volatility of the market when no trades are performed.

4.2 Simulation agents

There are three broad classes of agents used in this simulation: arbitrageurs (who attempt to profit from deviations between Uniswap and the market), liquidity providers (who hold portfolios of UNI coins and currencies α and β), and traders (who trade coins with the Uniswap market, subject to simple rules). We describe each type of agent in detail below.

Arbitrageurs.

Arbitrageurs seek to maximize their profit by trading between the Uniswap market and a reference market. An arbitrageur agent solves an instance of problem (4) with a simple quadratic cost of risk model:

$$f_\alpha(\Delta_\alpha) = \frac{\rho_\alpha}{2}\Delta_\alpha^2, \quad f_\beta(\Delta_\beta) = \frac{\rho_\beta}{2}\Delta_\beta^2,$$

and performs the necessary trades. Here, $\rho_\alpha, \rho_\beta \geq 0$ are parameters which control the penalty incurred for a trade.

We use this model instead of the riskless case of problem (2), as live systems have noise during trades (*e.g.*, network delays) such that opportunities found by exact minimization of the riskless problem (2) may not be easily executed or may close during the time it takes to perform the trade.

Additionally, we are able to model a wide variety of risk-taking behavior by arbitrageurs by, for example, increasing the parameters ρ_α and ρ_β to have an arbitrageur who is less prone to perform risky (large) trades, even with large arbitrage opportunities available.

Liquidity providers.

There are two types of liquidity providers we model in this simulation.

The first are the *initial* liquidity providers, who begin the simulation by providing some amount of coins α, and β to the reserve and seek to gain profits by taking fees from Uniswap trades. We assume these liquidity providers will not withdraw their position until the end of the simulation.

The second are *rational* liquidity providers. These agents perform Markowitz portfolio optimization (see [20, §4.4.1]) on the three possible coins in the market: α, β, and the corresponding UNI coin minted from the Uniswap market. Each agent then solves the (convex) portfolio optimization problem,

$$\begin{array}{ll} \underset{x}{\text{maximize}} & \hat{\mu}^T x - \dfrac{\lambda}{2} x^T \hat{\Sigma} x \\ \textit{subject to} & 1^T x = 1 \\ & x \geq 0, \end{array} \tag{14}$$

and makes the appropriate trades to rebalance their portfolio. In this problem, $x \in \mathbf{R}^3$ is the vector containing the portfolio positions of α, β, and UNI respectively. The problem data $\hat{\mu} \in \mathbf{R}^3$ is the vector of exponentially-weighted average returns for each of the three coins (with respect to the reference market), $\hat{\Sigma} \in \mathbf{S}^3_+$ is the exponentially-weighted covariance of returns, and $\lambda > 0$ is the penalty incurred by the risk. Since (14) is not known to have a closed form solution, we set up and solve problem (14) using the CVXPY [21] modeling language and the ECOS solver [22] in our simulation.

The rational liquidity providers simulate agents who seek to maximize profits by trading their positions in each coin and holding the respective coins to maximize their expected return on investment, based on observed historical averages.

Traders.

The final agent we model is a trader. In this case, the trader seeks to trade some amount Δ_α of coin for some second amount Δ_β (or vice versa), so long as the price of trading Δ_α coins on Uniswap differs no more than a constant percentage off from the same trade in the reference market.

In a way, traders embody the 'demand for liquidity,' or the fact that trades on Uniswap might happen due to exogenous influences. In our case, a trader will draw Δ_α (or Δ_β) from some probability distribution and check if performing this trade in the Uniswap market is at most some percentage more expensive than performing it in the reference market. If not—*i.e.*, if the agent is able to trade with Uniswap for a reasonable price—then the agent makes the trade using the Uniswap contract. Otherwise, no trade is performed.

4.3 Results

The results of our simulations suggest that the theoretical results derived above hold in practice under a wide variety of market conditions.

Arbitrage bounds.

Figures 1 and 2 show that, even under the presence of outside noise—such as random noise, or changes in reserves due to rational liquidity providers—the marginal Uniswap price stays within the predicted no-arbitrage bounds. In particular, figure 1 shows when the market has a small amount of noise and little drift, while figure 2 shows this when there is large negative drift with moderate noise. We find that these bounds hold in our simulations, even under large amounts of market noise and large drift rates, so long as the trades performed are not large.

Due to the discrete nature of the simulation, if a trader is the last agent to run at each time step, the no-arbitrage bounds could be broken as an arbitrageur may not have yet been able to perform arbitrage. While this is true (and is due to the nature of the simulation), the final result still clearly tracks the market price, as shown in figure 3.

Initial liquidity provider returns.

In the simulation, we also record the utility of the initial liquidity providers, defined as the difference between the value of the respective UNI coin the agents hold, as defined in §4.1, and a portfolio composed of the coins α and β traded in for UNI at the start of the simulation.

As shown in figure 4, we find that in almost all of our simulations, initial liquidity providers end up having negative utility (using this definition), over most varying conditions. More specifically, we note that initial liquidity providers only have positive expected value when the mean market return is close to zero, as they earn revenue from all transaction fees while not falling behind an equivalent portfolio of the given pairs of coins. This approximately follows the result given in (9), except that liquidity providers earn fees in this case.

5 Conclusion

Though simple, constant product markets and their generalizations have very nice theoretical properties (such as fairly strict no-arbitrage bounds on the reference price) which appear to hold in practice. Our simulations confirm that this is the case under a wide range of different market parameters and conditions, implying that the use of constant product markets as price oracles is, at least at first glance, sound.

Additionally, we suspect that there is an even larger class of automated market maker mechanisms which satisfy the above properties, and it would be interesting to further explore its mathematical properties. We leave this possible generalization for future work.

Acknowledgments

The authors would like to thank Tony Salvatore for the initial implementation of the market model used in the simulation and Jonathan Tuck, Hayden Adams, Alex Evans, and Dan Robinson for helpful comments and edits on the paper. The authors would also like to thank Regina Cai, Georgios Konstantopoulos, and Merrick Wong for pointing out some inconsistencies in the text, reviewers 1 and 2 for their comments and suggestions, and Thoma Zoto for pointing out that the original proof in appendix D was incorrect.

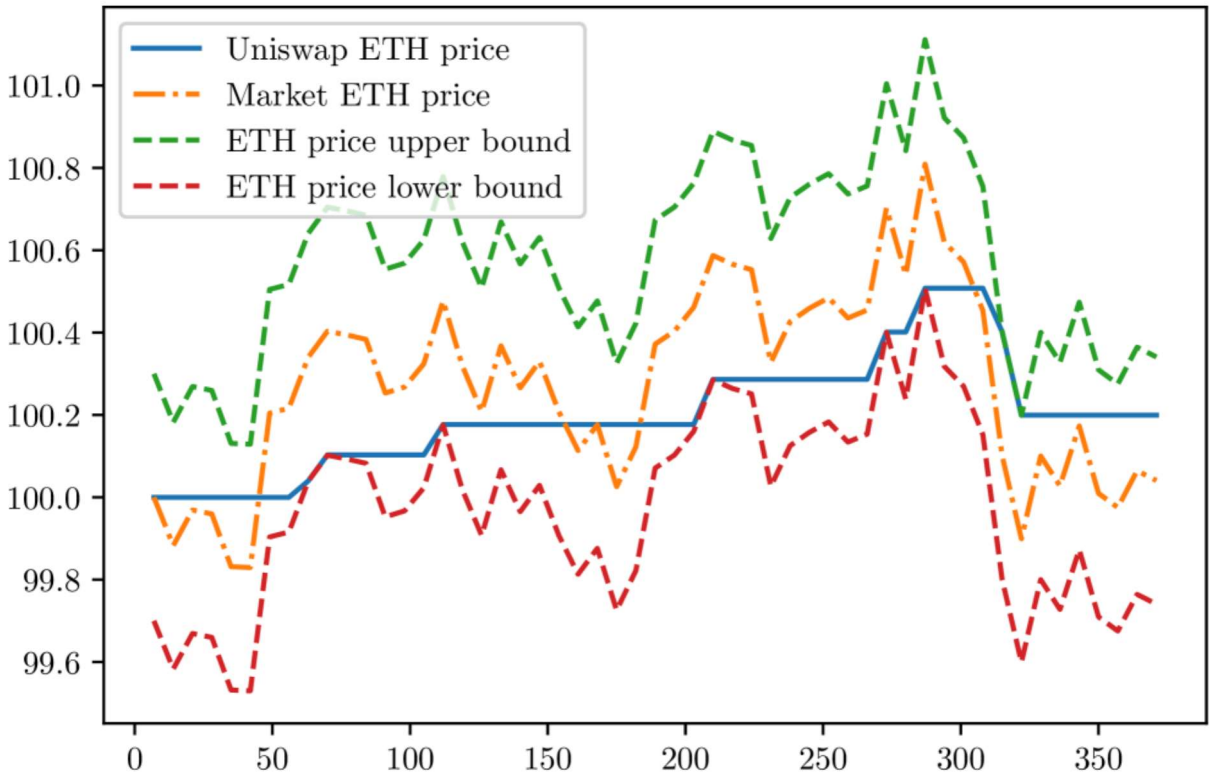

Figure 1: Market price (m_p) vs. Uniswap price (m_u) in no-drift condition with small noise and no traders. The plotted bounds are $\gamma m_p \leq m_u \leq \gamma^{-1} m_p$.

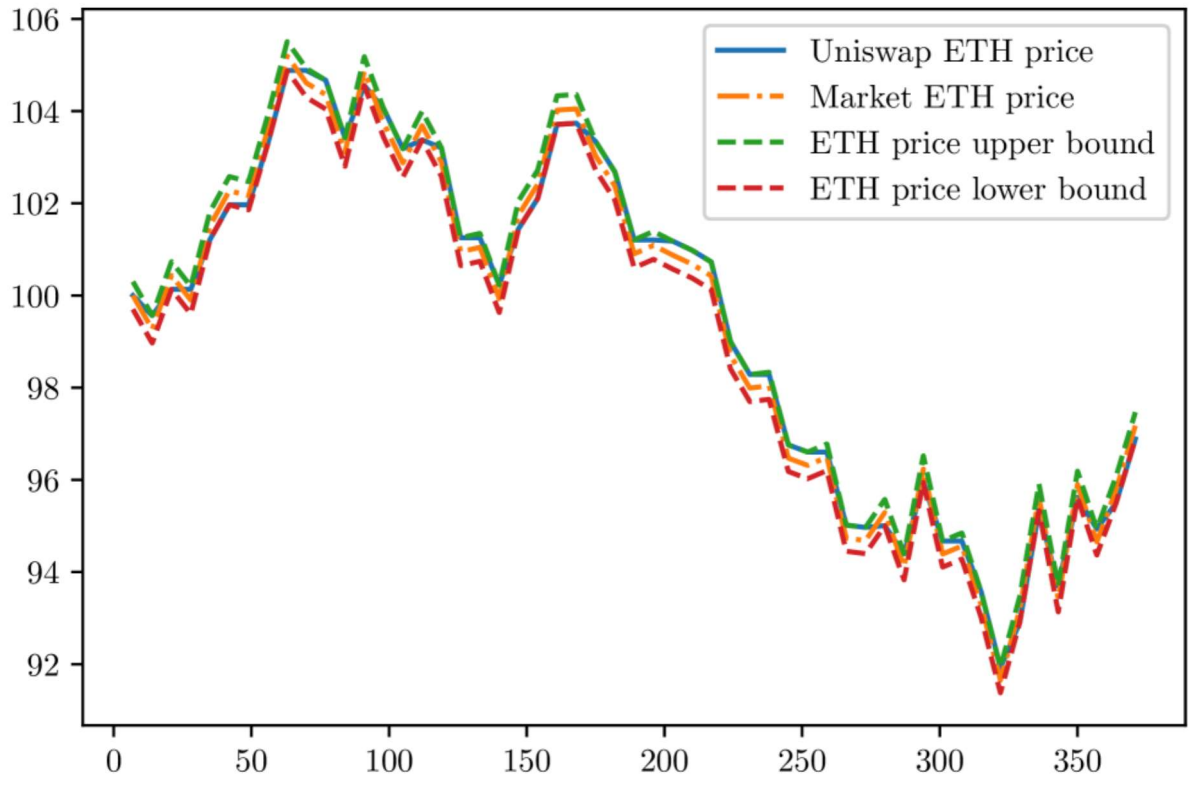

Figure 2: Market price (m_p) vs. Uniswap price (m_u) in negative-drift condition with moderate noise and no traders. The plotted bounds are $\gamma m_p \leq m_u \leq \gamma^{-1} m_p$

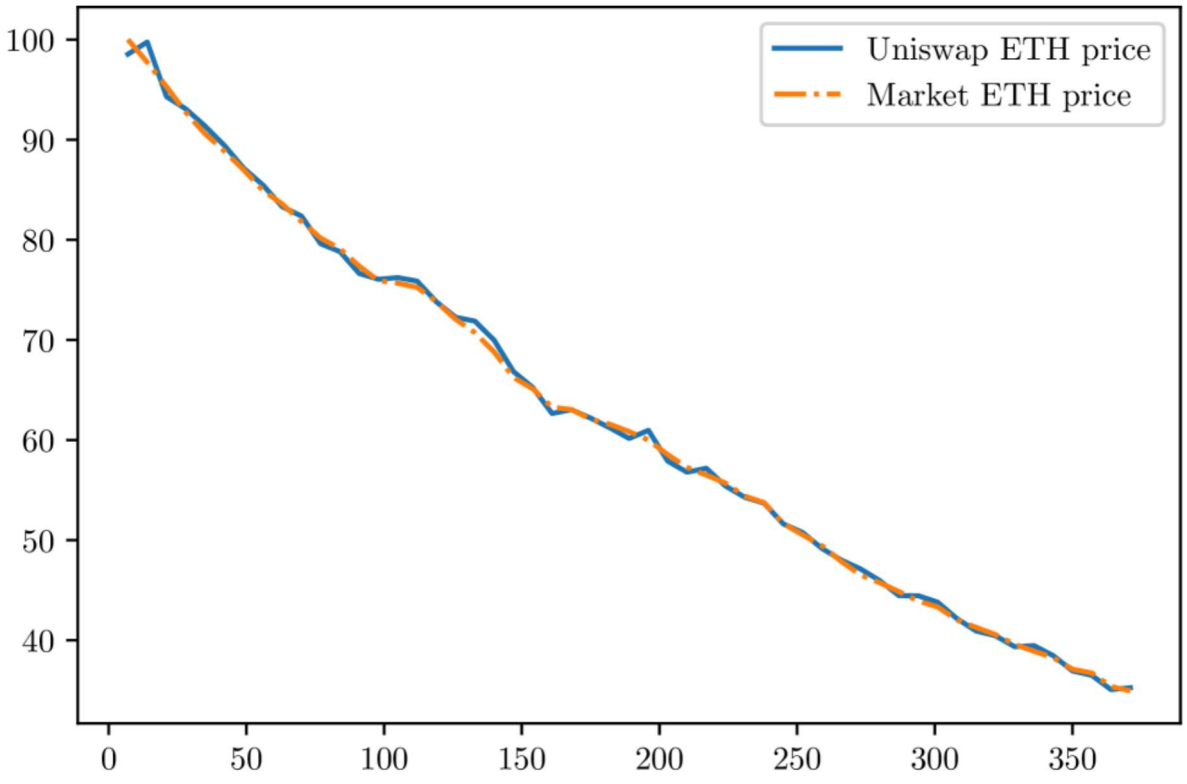

Figure 3: Market price (m_p) vs. Uniswap price (m_u) in large negative-drift condition with small noise, in the presence of traders. No bounds are plotted as they are nearly indistinguishable from the market ETH price, at this scale.

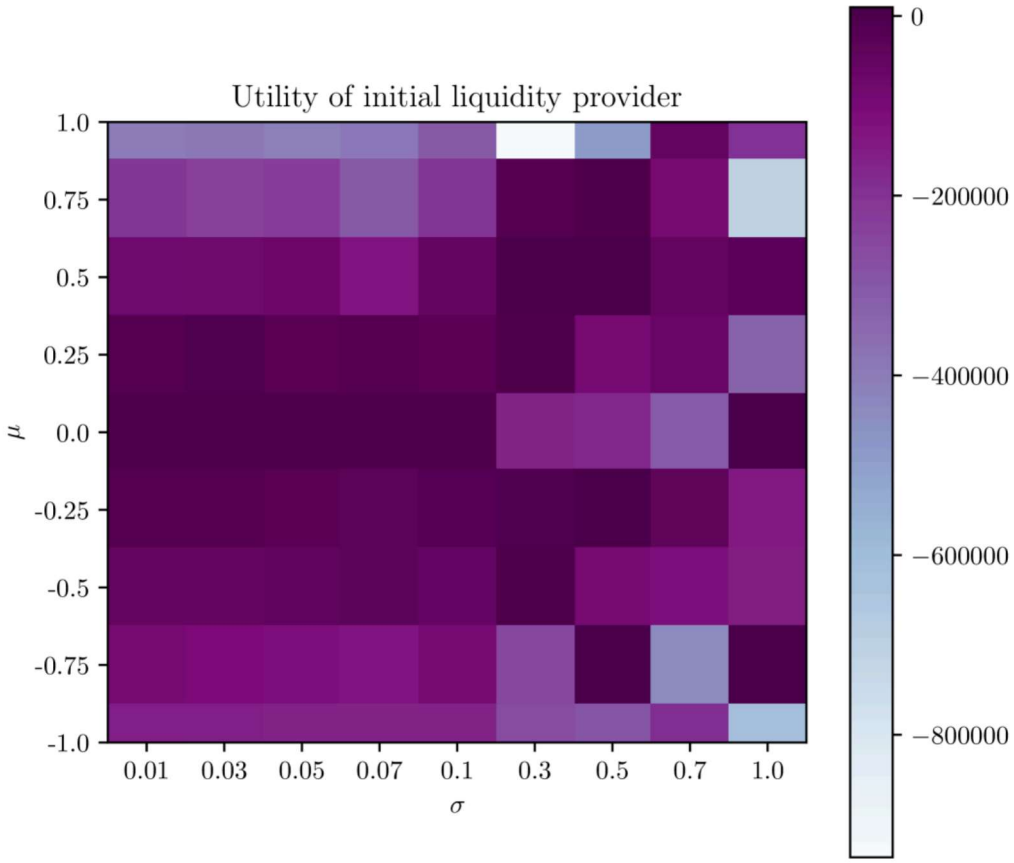

Figure 4: Varying utilities for initial liquidity providers over varying market conditions. As before, μ is the mean market return and σ is the volatility.

Appendix A The Uniswap arbitrage problem is convex

We can easily show that this problem is convex by using (1) to solve for Δ_β,

$$\Delta_\beta = \frac{1}{\gamma}\left(\frac{k}{R_\alpha - \Delta_\alpha} - R_\beta\right).$$

Note that Δ_β is then a convex function of Δ_α as $x \mapsto 1 / x$ is convex for x positive, and a convex function composed with an affine function is convex [20, §3.2.2]. The resulting (equivalent) problem,

$$\begin{array}{ll} \textit{maximize} & m_p\Delta_\alpha - \frac{1}{\gamma}\left(\frac{k}{R_\alpha - \Delta_\alpha} - R_\beta\right) \\ \textit{subject to} & \Delta_\alpha \geq 0, \end{array} \tag{15}$$

with variable Δ_α is then convex.

Since the problem is one-dimensional, the fact that problem (2) is convex is not immediately useful, but it can lead to several simple generalizations. For example, the optimal m-stage trading strategy with several interacting constant product markets can be efficiently evaluated in this case.

Optimality conditions.

Note that a maximum of a concave function over an interval happens either at (a) the interior of an interval or (b) at its boundary. In the latter case, it is not hard to show that a maximum is attained at the point on the boundary closest to the unconstrained maximum.[6]

[6]The proof follows from the fact that a concave function over $\mathbf{R}$ is monotonically nondecreasing (nonincreasing) to the left (right) of its maximum.

Because of this, we only have to consider the unconstrained version of problem (15), over the interval $[0, +\infty)$.

In this case, the unconstrained optimal points are those for which the objective of (15) has zero derivative. This happens when

$$\Delta_\alpha = R_\alpha - \sqrt{\frac{k}{\gamma m_p}}.$$

By the statement above, then the optimal solution to (15) is

$$\Delta_\alpha^\star = \left(R_\alpha - \sqrt{\frac{k}{\gamma m_p}}\right)_+,$$

where $(x)_+ = \max\{x, 0\}$ for $x \in \mathbf{R}$. The optimal $\Delta_\beta^\star$ can be easily derived using the $\Delta_\alpha^\star$ given above and the constant product formula (1).

Now, note that $\Delta_\alpha^\star$ is zero if, and only if

$$R_\alpha - \sqrt{\frac{k}{\gamma m_p}} \leq 0 \Leftrightarrow \gamma m_p \leq m_u,$$

where $m_u = R_\beta / R_\alpha$ is the marginal Uniswap market price of α without fees.

Since the objective of (15) is zero whenever there is no arbitrage opportunity (and the objective is only zero at $\Delta_\alpha^\star = 0$, by strict convexity), then the above implies there is no $\alpha \to \beta$ arbitrage in the presence of an infinitely liquid market. Swapping α for β yields the same result derived via no-arbitrage in the general case:

$$\gamma m_p \leq m_u \leq \gamma^{-1} m_p.$$

A.1 Extensions to Uniswap arbitrage problem

In a similar vein to (15), we can add any penalty given by a convex function $f : \mathbf{R}_+^2 \to \mathbf{R}$ which is nondecreasing in its second argument to the objective. This yields yet another convex optimization problem:

$$\begin{array}{ll} \textit{maximize} & m_p \Delta_\alpha - \Delta_\beta - f(\Delta_\alpha, \Delta_\beta) \\ \textit{subject to} & \dfrac{1}{\gamma}\left(\dfrac{k}{R_\alpha - \Delta_\alpha} - R_\beta\right) \leq \Delta_\beta \\ & \Delta_\alpha \geq 0, \end{array} \tag{16}$$

with variables $\Delta_\alpha \in \mathbf{R}$ and $\Delta_\beta \in \mathbf{R}$. This problem is equivalent to problem (4) as the objective function is decreasing with respect to Δ_β, implying that the first inequality constraint is always tight at an optimal point. Additionally, problem (16) is also convex as it is maximizing a concave function, subject to convex constraints. [20, §4.2.1]

Market models.

One particularly useful example of such a function f is for modeling the market response of trading coin α for coin β. In particular, if the marginal market price $m : \mathbf{R}_+ \to \mathbf{R}_+$ is some decreasing function of the total amount of coin Δ_α traded, satisfying $m(0) = m_p$, then the slippage cost of trading Δ_α coin for Δ_β is given by,

$$f(\Delta_\alpha) = \int_0^{\Delta_\alpha} (m_p - m(t))\ dt,$$

with f convex (as its derivative, $-m$, is increasing by definition). While several simple market models exist, the one we use in this paper is a special case of

$$m(t) = m_p(1 - \eta t^\xi),$$

with $\xi > 0$ and $\eta \geq 0$. This results in

$$f(\Delta_\alpha) = m_p\left(\frac{\eta}{\xi + 1}\Delta_\alpha^{\xi+1}\right).$$

Appendix B Derivation of price gap

By definition (1), the output of Δ_α can be written as

$$\Delta_\beta = \frac{R_\beta \Delta_\alpha}{\gamma(R_\beta - \Delta_\alpha)}.$$

Dividing the numerator and denominator by $R_\alpha > 0$ and using the fact that $(1-x)^{-1} = 1 + x + O(x^2)$ we get the (nearly final) result:

$$\Delta_\beta = \frac{\Delta_\alpha m_u}{\gamma(1 - \frac{\Delta_\alpha}{R_\alpha})} = m_u \gamma^{-1} \Delta_\alpha \left(1 - \frac{\Delta_\alpha}{R_\alpha} + O\left(\frac{\Delta_\alpha^2}{R_\alpha^2}\right)\right).$$

Subtracting Δ_β from Δ_β' and using the fact that $R_\alpha < R_\alpha'$ then yields the statement given in (7).

Appendix C Uniswap portfolio value under Brownian dynamics

Assume the trajectory of the market price m_p^t follows a geometric Brownian motion process [23, §5.1]

$$\frac{dm_p^t}{m_p^t} = \mu dt + \sigma dW^t,$$

where $\mu \in \mathbf{R}$ is the drift, $\sigma \in \mathbf{R}_+$ is the volatility, and W^t is a standard Brownian motion [23, §3]. Without loss of generality, let $m_p^1 = 1$. The portfolio value then has expectation

$$\mathbf{E}[P_V] = 2\sqrt{k} \quad \mathbf{E}\left[\sqrt{m_p^t}\right] = 2e^{\frac{T}{8}(4\mu - \sigma^2)}\sqrt{k} \quad = 2e^{-\frac{T}{8}\sigma^2}\sqrt{k\ \mathbf{E}[m_p^T]}, \tag{17}$$

where we have used the fact that the nth moment of a log-normal random variable X is given by [24]

$$\mathbf{E}\left[X^n\right] = e^{n\mu + n^2\sigma^2/2}.$$

This argument shows one could replicate the time T payoff of a no-fee liquidity provider with less initial capital than that required in the constant product market itself, which, in turn, implies that the portfolio value at time T must grow at a rate of $e^{\frac{T}{8}\sigma^2}$ under the no-arbitrage hypothesis. The choice of γ that gives this growth rate is the no-arbitrage fee. A simple replication of a volatility harvesting strategy can be used to show that some such γ exists under mild conditions on μ and σ [25, §11].

Appendix D Splitting trades is always more expensive

Given two sequentially-feasible trades $(\Delta_\alpha, \Delta_\beta)$ and $(\Delta_\alpha^{'}, \Delta_\beta^{'})$, we will show that the 'aggregate trade' $(\Delta_\alpha + \Delta_\alpha^{'}, \Delta_\beta^{\text{tot}})$ always satisfies $\Delta_\beta^{\text{tot}} > \Delta_\beta + \Delta_\beta^{'}$ for any fee $0 \leq \gamma < 1$ with $\Delta_\alpha, \Delta_\alpha^{'} > 0$. In other words, the total output for trading Δ_α and $\Delta_\alpha^{'}$ sequentially is always smaller than the output of trading $\Delta_\alpha + \Delta_\alpha^{'}$ all at once.

To show this, note that after the trade $(\Delta_\alpha, \Delta_\beta)$ is performed, the new reserves are given by $R_\alpha + \Delta_\alpha$ and $R_\beta - \Delta_\beta$. So, since $(\Delta_\alpha^{'}, \Delta_\beta^{'})$ is a feasible trade when performed after $(\Delta_\alpha, \Delta_\beta)$ (by definition) we have that

$$(R_\alpha + \Delta_\alpha + \gamma\Delta_\alpha^{'})(R_\beta - \Delta_\beta - \Delta_\beta^{'}) = (R_\alpha + \Delta_\alpha)(R_\beta - \Delta_\beta),$$

while, by definition, $(\Delta_\alpha + \Delta_\alpha^{'}, \Delta_\beta^{\text{tot}})$ is feasible for reserves R_α, R_β, and we have

$$(R_\alpha + \gamma(\Delta_\alpha + \Delta_\alpha^{'}))(R_\beta - \Delta_\beta^{\text{tot}}) = R_\alpha R_\beta.$$

Since $(\Delta_\alpha, \Delta_\beta)$ is also feasible for these reserves, we have that $(R_\alpha + \gamma\Delta_\alpha)(R_\beta - \Delta_\beta) = R_\alpha R_\beta$ so,

$$(R_\alpha + \gamma(\Delta_\alpha + \Delta_\alpha^{'}))(R_\beta - \Delta_\beta^{\text{tot}}) = (R_\alpha + \gamma\Delta_\alpha)(R_\beta - \Delta_\beta).$$

Solving for $\Delta_\beta + \Delta_\beta^{'}$, we find

$$\Delta_\beta + \Delta_\beta^{'} = R_\beta - \frac{(R_\alpha + \Delta_\alpha)(R_\beta - \Delta_\beta)}{R_\alpha + \Delta_\alpha + \gamma\Delta_\alpha^{'}},$$

while solving for $\Delta_\beta^{\text{tot}}$ gives

$$\Delta_\beta^{\text{tot}} = R_\beta - \frac{(R_\alpha + \gamma\Delta_\alpha)(R_\beta - \Delta_\beta)}{R_\alpha + \gamma(\Delta_\alpha + \Delta_\alpha^{'})}.$$

This gives

$$\Delta_\beta^{\text{tot}} - (\Delta_\beta + \Delta_\beta^{'}) = (R_\beta - \Delta_\beta)\left(\frac{R_\alpha + \Delta_\alpha}{R_\alpha + \Delta_\alpha + \gamma\Delta_\alpha^{'}} - \frac{R_\alpha + \gamma\Delta_\alpha}{R_\alpha + \gamma(\Delta_\alpha + \Delta_\alpha^{'})}\right) > 0,$$

where the inequality follows from the fact that $R_\beta - \Delta_\beta > 0$ combined with the inequality:

$$\frac{x + z}{y + z} > \frac{x}{y},$$

whenever $y > x \geq 0$ for any $z > 0$.

Appendix E Cost of manipulation

We can derive the cost of manipulation in the case where the reference market is infinitely liquid (*i.e.*, when $\Delta_\beta = m_p\Delta_\alpha$). We will derive this cost in the no-fee case as this lower bound on the

cost is still a lower bound in the case with fees.

First, assume that an attacker wishes to increase the Uniswap price of a pair of coins α and β, by adding Δ_β coins to the system and removing Δ_α coins such that

$$\frac{R_\beta + \Delta_\beta}{R_\alpha - \Delta_\alpha} = (1+\varepsilon)m_p,$$

where $\varepsilon > 0$ is some desired constant. Using the fact that $(R_\alpha - \Delta_\alpha)(R_\beta + \Delta_\beta) = k = m_p R_\alpha^2$ (where the second equality follows from (3) with $\gamma = 1$), we get that

$$(1+\varepsilon)m_p^2 R_\alpha^2 = (R_\beta + \Delta_\beta)^2,$$

or, after some simplification,

$$\Delta_\beta = R_\beta(\sqrt{1+\varepsilon} - 1),$$

required to increase the price. Since the attacker receives $\Delta_\alpha = R_\beta(1 - (\sqrt{1+\varepsilon})^{-1}) / m_p$ as a result of this trade, the total cost of the attack is then

$$C(\varepsilon) = \Delta_\beta - m_p\Delta_\alpha = R_\beta(\sqrt{1+\varepsilon} + (\sqrt{1+\varepsilon})^{-1} - 2),$$

where $C(\varepsilon)$ is the cost of carrying out the attack for a single time period.

Bounding from below.

Since $C(\varepsilon)$ is twice differentiable in ε over $0 \le \varepsilon \le 1$ with $C(0) = 0$ and is increasing, then,

$$C(\varepsilon) \ge R_\beta \frac{\varepsilon^2}{2} \inf_{0 \le \varepsilon' \le 1} C''(\varepsilon) = \left(\frac{1}{32\sqrt{2}}\right) R_\beta \varepsilon^2,$$

where the second equality follows since $C''(\varepsilon)$ is decreasing over $0 \le \varepsilon \le 1$.

The $\varepsilon \ge 1$ case is simple to bound by noting that

$$x + x^{-1} - 2 \ge \kappa x,$$

for all $x \ge \sqrt{2}$ whenever $\kappa = 3/2 - \sqrt{2}$ (this can be verified by multiplying the above inequality by x to receive a convex quadratic whose largest root is $\sqrt{2}$). Therefore,

$$\sqrt{1+\varepsilon} + (\sqrt{1+\varepsilon})^{-1} - 2 \ge \kappa\sqrt{1+\varepsilon} \ge \kappa\sqrt{\varepsilon},$$

so the claim follows,

$$C(\varepsilon) \ge \kappa R_\beta \sqrt{\varepsilon}.$$

For an explicit bound on K, note that

$$K \geq \min\left\{ \frac{1}{32\sqrt{2}}, \frac{3}{2} - \sqrt{2} \right\} = \frac{1}{32\sqrt{2}},$$

follows from the above.

The bounds, as stated, are tight up to a constant multiplicative factor (which follows from the asymptotic behavior of $C(\varepsilon)$ as $\varepsilon \downarrow 0$ and $\varepsilon \uparrow +\infty$, respectively) although they are—in their current state—only useful as a theoretical tool, as the provided constants are very loose.

Appendix F The arbitrage problem in constant mean markets

The problem

$$\begin{aligned}
&\textit{maximize} && \sum_{i=1}^{n} \left(\sum_{j=1}^{n} \Lambda_{ij} m_j^p - \Delta_i m_i^p \right) \\
&\textit{subject to} && \prod_{i=1}^{n} \left(R_i + \gamma_i \Delta_i - \sum_{j=1}^{n} \Lambda_{ij} \right)^{w_i} \geq k \qquad (18) \\
&\Delta_i \geq 0, \quad i = 1, \ldots, n \\
&\Lambda_{ij} \geq 0, \quad i, j = 1, \ldots, n,
\end{aligned}$$

with variables $\Delta \in \mathbf{R}^n$ and $\Lambda \in \mathbf{R}^{n \times n}$ and with weights $w \geq 0$ and $\mathbf{1}^T w = 1$, is equivalent to problem (12) in that any optimal solution for problem (18) is optimal for problem (12).

Clearly, if any optimal point of (18) has the first inequality constraint holding at equality, then this point is optimal for (12). Because the objective is decreasing in Δ_i for each $i = 1, \ldots, n$ and increasing in Λ_{ij} for each $i, j = 1, \ldots, n$, while the geometric mean function in the inequality constraint is decreasing in Λ_{ij} and increasing in Δ_i, then the inequality constraint is always tight at equality at any optimal point.

Since the weighted geometric mean function is concave [20, §3.1.5], then problem (18) is a convex problem.

We also note that problem (12) is log-log convex [26] with the obvious change of variables. While we do not explore this connection here, we suspect that this approach might provide a simple way of stating what class of functions can be used as an automated market maker mechanism such that the resulting no-arbitrage bounds are useful.

References

[1] G. Wood *et al.*, “Ethereum: A secure decentralised generalised transaction ledger,” *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[2] L. Goodman, “Tezos—a self-amending crypto-ledger white paper,” *URL: https://www. tezos. com/static/papers/white_paper. pdf*, 2014.

[3] H. Murphy, “‘DeFi’ movement promises high interest but high risk,” Dec 2019.

[4] R. Leshner and G. Hayes, “Compound: The money market protocol,” tech. rep., February 2019.

[5] S. Kamvar, M. Olszewski, and R. Reinsberg, “Celo: A multi-asset cryptographic protocol for decentralized social payments,” 2017.

[6] M. Team, “The dai stablecoin system,” *URl: https://makerdao. com/whitepaper/DaiDec17WP. pdf*, 2017.

[7] Y. Zhang, X. Chen, and D. Park, “Formal specification of constant product (xy=k) market maker model and implementation,” 2018.

[8] W. Warren and A. Bandeali, “0x: An open protocol for decentralized exchange on the Ethereum blockchain,” 2017.

[9] E. Hertzog, G. Benartzi, and G. Benartzi, “Bancor protocol,” 2017.

[10] C. Decker and R. Wattenhofer, “Bitcoin transaction malleability and mtgox,” in *European Symposium on Research in Computer Security*, pp. 313–326, Springer, 2014.

[11] D. Shane, “A crypto exchange may have lost $145 million after its CEO suddenly died,” *CNN Business, https://edition.cnn.com/2019/02/05/tech/quadrig gerald-cotten-cryptocurrency/index. html*, 2019.

[12] I. Kaminska, “Bitcoin bitfinex exchange hacked: The unanswered questions,” *Financial Times*, vol. 4, 2016.

[13] M. Wyart, J.-P. Bouchaud, J. Kockelkoren, M. Potters, and M. Vettorazzo, “Relation between bid–ask spread, impact and volatility in order-driven markets,” *Quantitative Finance*, vol. 8, no. 1, pp. 41–57, 2008.

[14] A. Juliano, “dydx: A standard for decentralized derivatives,” 2017.

[15] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels, “Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges,” *arXiv:1904.05234 [cs]*, Apr. 2019.

[16] R. Hanson, “Combinatorial information market design,” *Information Systems Frontiers*, vol. 5, no. 1, pp. 107–119, 2003.

[17] A. Othman, D. M. Pennock, D. M. Reeves, and T. Sandholm, “A practical liquidity-sensitive automated market maker,” *ACM Transactions on Economics and Computation (TEAC)*, vol. 1, no. 3, pp. 1–25, 2013.

[18] F. Martinelli and N. Mushegian, “Balancer: A non-custodial portfolio manager, liquidity provider, and price sensor,” 2019.

[19] J. R. Birge and V. Linetsky, *Handbooks in Operations Research and Management Science: Financial Engineering*, vol. 15. Elsevier, 2007.

[20] S. P. Boyd and L. Vandenberghe, *Convex Optimization*.
Cambridge, UK ; New York: Cambridge University Press, 2004.

[21] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, "A rewriting system for convex optimization problems," *Journal of Control and Decision*, vol. 5, no. 1, pp. 42–60, 2018.

[22] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An SOCP solver for embedded systems," in *2013 European Control Conference (ECC)*, (Zurich), pp. 3071–3076, IEEE, July 2013.

[23] B. K. Øksendal, *Stochastic Differential Equations: An Introduction with Applications*.
Universitext, Berlin Heidelberg New York Dordrecht London: Springer, sixth edition, sixth corrected printing ed., 2013.
OCLC: 935584333.

[24] C. Walck, "Hand-book on statistical distributions for experimentalists," tech. rep., 1996.

[25] L. C. MacLean, E. O. Thorp, and W. T. Ziemba, eds., *The Kelly Capital Growth Investment Criterion: Theory and Practice*.
No. 3 in World Scientific Handbook in Financial Economic Series, 2010-1732, Singapore ; Hackensack, NJ: World Scientific, 2011.

[26] A. Agrawal, S. Diamond, and S. Boyd, "Disciplined Geometric Programming," *arXiv:1812.04074 [cs, math]*, Mar. 2019.