Armored AMMs:
Capturing Arbitrage Opportunities with Defensive Rebalancing

Sam Devorsetz ’24
Advisor: Maurice Herlihy
Reader: Amy Greenwald

April 2024
Abstract

This thesis investigates the new paradigm of Defensive Rebalancing in the context of Automatic Market Makers (AMMs) in decentralized finance (DeFi). With the prevalence of arbitrage whenever AMMs present different spot prices, there arises the question of how liquidity providers (LPs) can capture some of the arbitrageur’s would-be profit. One promising strategy is Defensive Rebalancing, where AMMs make trades with one another to eliminate an arbitrage opportunity between a given set of AMMs. We present the Optimal Rebalancing Problem, a modification of the Optimal Routing Problem several nuances. We provide a mathematical framework to analyze various self-rebalancing strategies in terms of optimality and relative welfare. We connect proportional increases in the trading function to monetary value and detail a convex optimization problem which can be efficiently solved to maximize LPs’ profits under specific fairness constraints.
Contents

1 Introduction 4

2 Model 6

2.1 AMMs ................................................................. 6

2.1.1 The Reachable Set .............................................. 10

2.2 Price Representations ............................................. 10

2.2.1 Scaling With a Numeraire .................................... 11

2.2.2 Unit Vector Normalization ................................. 12

2.3 Properties ............................................................ 12

2.3.1 Price Expressiveness ............................................. 12

2.3.2 Concavity of the trading function ............................ 13

2.3.3 Trading Functions are k-Homogeneous ..................... 13

2.4 Network of AMMs .................................................. 14

3 Arbitrage 15

3.1 The Menace of Arbitrage ......................................... 15

3.2 Classes of Arbitrage ............................................... 16
3.3 Arbitrage Between two CPMMs .................................................. 16
3.4 The Optimal Routing Problem ..................................................... 18
3.5 Access to Stable Prices ............................................................. 19
  3.5.1 Arbitrage given a Central Market .......................................... 19
  3.5.2 AMMs Given a Price Oracle .................................................. 20

4 Defensive Rebalancing ............................................................... 22
  4.1 A Motivating Example ............................................................ 22
  4.2 Problem Statement ............................................................... 23
    4.2.1 Why is this difficult? ......................................................... 23
  4.3 Optimal Rebalancings ............................................................ 24
    4.3.1 Same Trading Function ....................................................... 25
    4.3.2 Different Trading Functions ............................................... 25
  4.4 Space of Rebalancings ........................................................... 26
    4.4.1 Example: Two CPMMs ......................................................... 26
    4.4.2 Generalizing ................................................................. 26
  4.5 Existence .......................................................................... 27
    4.5.1 Without Price Expressiveness ............................................. 27
    4.5.2 With Price Expressiveness ................................................. 28
  4.6 Price Uniqueness ................................................................. 30
    4.6.1 Identical Trading Functions ............................................... 30
    4.6.2 Different Trading Functions .............................................. 31
4.7 Trading Functions as Utility ................................................. 31
  4.7.1 Normalizing the Trading Function ................................. 33
4.8 Connecting Profit to Monetary Value ................................. 33
4.9 Optimal Rebalancing via Convex Optimization ..................... 34
  4.9.1 Profit Equalizing Rebalancing .................................... 35
  4.9.2 Mapping onto the Optimal Routing Problem .................... 36
  4.9.3 Maximizing Total Welfare ......................................... 36
4.10 Revisiting Profit ............................................................ 38
  4.10.1 Does Profit Work? .................................................. 38
  4.10.2 Alternatives To Profit ............................................. 39

5 Practical Considerations .................................................. 40
  5.1 Scalability problems .................................................. 40
  5.2 Implementing Rebalancing ........................................... 40
  5.3 Transaction Ordering ................................................ 41

6 Conclusion ................................................................. 43

A Appendix ................................................................. 44
  A.1 Proof: Price-Equalizing Rebalancings are Undominated .......... 44
  A.2 Code: Equal-Profit Rebalancing .................................... 47
Chapter 1

Introduction

The evolving landscape of financial technology has given rise to the Automatic Market Maker (AMM), which has become a dominant force in Decentralized Finance (DeFi). Traditionally, financial markets operate on an order book model, where buyers and sellers list their orders, and a trade is executed when two orders match. This model, while efficient in centralized markets, is not particularly well-suited to full decentralization, for several reasons including scalability and the inherent latency in confirming on-chain transactions.

In DeFi, there are two major classes of exchanges: Centralized Exchanges (CEXs) and Decentralized Exchanges (DEXs). CEXs, such as Coinbase, facilitate the conversion of fiat currencies like USD and EUR into cryptocurrencies such as ETH, BTC, and USDC. They function similarly to traditional stock exchanges and offer users the choice to either store their funds in managed wallets or transfer them to on-chain wallets. DEXs, on the other hand, are exchanges that operate fully on-chain [7]. Automatic Market Makers (AMMs), the focus of this thesis, are a key component of the DEX ecosystem, and have gained traction due to their ability to facilitate trading without the need for manual intervention.

AMMs represent a paradigm shift by completely eliminating the need for order books, using mathematical formulas to set market prices instead. Rather than matching buyers and sellers, transactions occur directly with the AMM’s liquidity pool. This thesis focuses on Constant Function Market Makers (CFMMs), a type of AMM that maintains an invariant function of its reserves, allowing it to execute trades. CFMMs offer the flexibility that they can adapt to new market conditions without intervention.
For a sense of scale, on Uniswap, the most popular AMM platform, the USDC/ETH trading pair alone has $302 million in total value locked and $376 million in daily trade volume at the time of writing. Uniswap is far from the only AMM platform, with others including PancakeSwap and StableSwap, in addition to UniswapV2, UniswapV3, and the upcoming release of UniswapV4.

AMMs are passive systems that rely on arbitrage to correct their prices in response to price changes in external markets. In a perfectly efficient market, prices between all exchanges, centralized and decentralized, would be equal, thereby eliminating arbitrage opportunities. Arbitrage, however, typically results in a cost to liquidity providers, known as *impermanent loss*. Millionis et. al. define the metric *Loss Versus Rebalancing (LVR)* to quantify this loss [17]. There has been research into minimizing LVR for LPs [16] and this remains a significant focus of cryptocurrency research.

Price discrepancies between CEXs and DEXs are an important source of losses for LPs, but there is more to consider. The economies for major tokens, such as USDC, ETH, BTC, etc., are dominated by centralized markets, which are so liquid that we can think of CEX prices as a “source of truth”. Conversely, for longer-tail assets, a central reference price might not exist, and their markets are determined by a range of decentralized exchanges. Arbitrageurs also stabilize prices between DEXs trading overlapping tokens at inconsistent prices by executing arbitrage trades fully on-chain. We propose a system for AMMs to team up to capture would-be arbitrage profit resulting from divergent prices.

*Defensive rebalancing*, the focus of this thesis, serves as a countermeasure against arbitrageurs. We propose a system where AMMs engage in mutually beneficial trades with each other to eliminate arbitrage opportunities, and in the process redistribute these profits back to their respective liquidity pools. In this thesis, we detail strategies to effectively implement defensive rebalancing and analyze the optimality of various approaches. Given the practically infinite range of possible rebalancings, we also explore different utility functions and fairness conditions to ensure that rebalancing is cost-effective for LPs.
Chapter 2

Model

In this chapter, we define a model necessary for our analysis in later chapters. We present a formal definition for Automatic Market Makers and their mathematical properties which we enforce, along with certain assumptions.

2.1 AMMs

Because CFMMs are the dominant form of AMM today [14], we will use the two terms interchangeably in this thesis. AMMs have become so popular in large part due to their simplicity.

Since AMMs operate as smart contracts on the blockchain, they are passive. Specifically, its state will only change when another party transacts with it. There are two main types of transactions for an AMM. The most common is a trade, where a trader exchanges some assets with the AMM’s liquidity pool, according to some trading function, and also pays a small trading fee (eg. 0.3% for UniswapV1). The second type of transaction involves a liquidity provider (LP). A liquidity provider is a party that funds the AMM’s liquidity pool and gains revenue via trading fees. An LP’s stake is represented by LP Tokens, which are NFTs that can be used to redeem a portion of trading fees and remove liquidity. LPs can interact with an AMM by adding or removing liquidity, or by withdrawing accumulated trading fees.

A simple form of an AMM, the Constant Product AMM (CPMM), is the following, based on UniswapV1. This AMM exchanges two assets, Tokens ($T$), and Dollars ($D$). Its state
defined by the token pair \((T, D) \in \mathbb{R}^2\) (its reserves), a constant \(C \in \mathbb{R}\) such that \(T \times D = C\), where the AMM accepts each trade such that the AMM’s new state \((T + \Delta_T, D + \Delta_D)\) fits the following invariant:
\[
(D + \Delta_T) \times (D + \Delta_D) = C
\]

The CPMM’s spot price of dollars in terms of tokens is \(\frac{T}{D}\) tokens for one dollar. One of the biggest advantages is that CPMMs bear the quality that its reserves are always priced such that the value of each asset in its reserves is equal. This is useful because the CPMM is equally exposed to each asset under all price conditions.

We will frequently reference the CPMM throughout this thesis, due to its prevalence and simplicity. While the two-token CPMM is a good starting point to the study of AMMs and is still widely used, we can generalize this definition to fit the broader class of CFMMs, trading potentially more than two assets and using different trading functions. While we often think of the invariant in AMMs as strict equality, for the purpose of this thesis, we are using the definition of a CFMM in Angeris et al. [5], which defines the invariant as a trading function.

A Constant Function Market Maker is a state machine defined by two objects:

- A reserves vector \(X \in \mathbb{R}_+^k\), where \(k \in \mathbb{N}\) represents the number of tokens, and \(X_i\) represents the AMM’s amount of token \(i\).

- A twice-differentiable, concave trading function \(f : \mathbb{R}_+^k \to \mathbb{R}_+\) that is strictly increasing with respect to each reserve, requiring each asset to have positive value. The AMM accepts any trade vector \(\Delta \in \mathbb{R}_+^k\) that maintains its invariant\(^1\):

\[
f(X + \Delta) = f(X) \tag{2.1}
\]

and the AMM’s reserves remain non-negative:

\[
\forall i : 0 \leq i < k; \quad X_i + \Delta_i > 0
\]

In this thesis, we relax the equality in Equation 2.1 to allow trades that lie above the level curve \(f(X)\), in essence modifying Equation 2.1 to be \(f(X + \Delta) \geq f(X)\). Any reasonable trader would maximize their value by trading on the level curve, since for any trade such that \(f(X + \Delta) > f(X)\) there is another \(\Delta' < \Delta\) such that \(f(X + \Delta) = f(X)\). Since the trader’s

\(^1\)The invariant holds for all trades, but LPs shift the invariant when adding and removing liquidity.
side is $-\Delta$, $-\Delta' > -\Delta$ would be more valuable and therefore a strictly better trade than $-\Delta$. However, in this thesis, we assume AMMs accept any trade satisfying the inequality below, because from the LP’s perspective, why turn such a trade down?

\[
\Delta \geq \Delta'
\]

This modification leaves room for our analysis on self-rebalancing trades, where trades between AMMs increase the trading function.

By generalizing our trading function to any function $f : \mathbb{R}^k_+ \to \mathbb{R}_+$, we can define AMMs with different types of invariants. We can reformulate the CPMM as $f(X) = \prod_{i=1}^k X_i$. Other invariants include the Constant Sum Market Maker, defined by the invariant $\sum_{i=1}^k p_i X_i = C$, where $p_i$ is the price of asset $i$ in terms of asset 1. Stableswap (Figure 2.2) mixes the constant-sum and constant-product formula with the invariant

\[
\chi D^{k-1} + \sum_{i=1}^k X_i + \prod_{i=1}^k X_i = \chi D^k + \left(\frac{D}{k}\right)^k
\]

The parameter $D$ represents the sum of token values when all tokens are at the same price, and $\chi$ is a “leverage” parameter. Higher values of $\chi$ correspond to a curve resembling the constant-sum curve, while lower values favor the constant-product curve [18, 11].

An AMM’s invariant is crucial, as it affects the movement of an AMM’s prices and balance of liquidity in different market conditions. For example, Stableswap concentrates liquidity in a narrow price range, making it suitable for stablecoins. CPMM prices are more sensitive
to trades, decreasing their exposure to impermanent loss, which we will discuss in the next section.

In AMMs, the difference between the spot price and realized price is called \textit{price slippage} \cite{23}. This discrepancy exists because because of the convexity of the reachable set. If we view a trade as a line connecting two points $X$ and $X' \in S$, this line will lie above or on the tangent line, which represents the spot price (or on the tangent line, representing no slippage when we have strict equality such as in a CSMM, where the boundary of the reachable set is a plane and there is no slippage).

To illustrate this with an example, imagine a CPMM with reserves $(T, D) = (40, 40)$, and a trader hopes to purchase 8 tokens ($\Delta_T = 8$). While the spot price for tokens is $1$ per token, the realized price ends up at $1.20$:

$$(T + 8) \times (D + \Delta_D) = T \times D$$

$$\Delta_D = -10$$
2.1.1 The Reachable Set

We can define the \textit{reachable set} of an AMM given its trading function and current state. The reachable set $S \subseteq \mathbb{R}_+^k$ is the set of reserves such that $X' \in S \implies f(X') \geq f(X)$ [5]. In other words, the reachable set is the set of reserves that, when viewing the trading function as a utility function, yield a higher utility. For example, the reachable set for a 2-asset CPMM with reserves $(T, D)$ is $S = \{(T', D') \in \mathbb{R}_+^2 | T'D' \geq TD \}$.

A reachable set $S \in \mathbb{R}_+^k$ of an AMM follows the following properties, given an AMM's current state. These properties hold when we ignore fees such as gas costs and trading fees (Angeris et. al. call this property “path independence”) [5]:

1. $X_i \geq 0$ for each $X \in S$ and token $i$.
2. The set $S$ is nonempty, closed, and convex.
3. The set $S$ is \textit{upward closed}, meaning given $X \in S$, $X' \geq X \implies X' \in S$.

2.2 Price Representations

Earlier, we defined our trading function $f$ to be differentiable. This is important because we can derive spot prices from partial derivates of $f$. We can use the trading function to derive the spot price of asset $i$ in terms of asset $j$:

$$P(i,j) = \frac{\partial f(X)}{\partial X_i} \frac{\partial f(X)}{\partial X_j}$$

For example, with the two-token CPMM, $f(T, D) = TD$, so $\frac{\partial}{\partial X} f(X) = (D, T)$, yielding $P(T, D) = \frac{D}{T}$.

Similarly, we can take the gradient $\nabla f(X)$ to derive a price vector. However, the values of $\nabla f(X)$ need to be scaled to be interpreted as prices.
2.2.1 Scaling With a Numeraire

One option is to take prices with respect to a numeraire asset, by convention the first asset \((i = 1)\). Normalizing with respect to asset 1, we have a price vector \(P\), where \(P_i = P(i, 1)\) \[3\]:

\[
P = \nabla f(X) \frac{\partial f}{\partial X_1}
\]

Numeraire scaling is especially useful where the numeraire asset has a stable value and acts as a good reference price. In that scenario, we can think of every other asset in terms of its value with respect to the numeraire. However, in the context of this thesis, this type of scaling has its flaws, particularly because we aren’t guaranteed to have a stable asset with agreed-upon value.

Imagine Alice and Bob value tokens 2, \ldots, \(k\) equally, such that they would be indifferent exchanging any of these assets at a 1-1 rate. Say we are using token 1 as our numeraire, and Alice values the numeraire twice as much as all the other tokens, and Bob values the numeraire the same as he does the other tokens. Then,

\[
P_{Alice} = (1, 0.5, \ldots, 0.5), \quad P_{Bob} = (1, 1, \ldots, 1)
\]

Now, imagine we wanted to quantify how valuable a basket of tokens \(B = (1, \ldots, 1)\) is for Alice and Bob. Our goal is for each \(p_j\)’s magnitude to represent somebody’s relative value for each good, which we can compare to others. Then, the value of \(B\) to Alice is the dot product \(P_{Alice} \cdot B = 1 + \frac{k-1}{2} = \frac{k+1}{2}\). For Bob, the value of \(B\) is \(P_{Alice} \cdot B = k\). This does not work. Just because Alice values the numeraire more than Bob does, her value for all non-numeraire assets, under the dot-product definition of value, is half of Bob’s. This does not capture reality though, since they both are willing to trade those assets on a 1-1 basis. Alice’s greater valuation of the numeraire should not imply that she values the other assets less in comparison to Bob.

For this reason, numeraire scaling is less than ideal in scenarios where we do not have a numeraire with agreed-upon value. For this reason, we use an approach that weighs each asset’s value equally.
2.2.2 Unit Vector Normalization

Another option is to divide the price vector by its sum, the L1 norm defined by $\| P \|_1$, so that $\sum_{j=1}^{k} P_j = 1$. In the example above,

$$P_{\text{Alice}} = \left( \frac{2}{k+1}, \frac{1}{k+1}, \cdots, \frac{1}{k+1} \right), \quad P_{\text{Bob}} = \left( \frac{1}{k}, \frac{1}{k}, \cdots, \frac{1}{k} \right)$$

In this case, Bob’s non-numeraire assets are not twice as valuable for him than for Alice. Now, Alice and Bob’s values for basket $B$ (one of each asset) are both 1, trivially. We now have a more appropriate relationship, where each asset has an equal impact on the other assets’ values when compared to someone else with a different price belief. For this reason, in this thesis, we use unit vector normalization.

2.3 Properties

In this section, we define some notable properties of AMMs for our analysis.

2.3.1 Price Expressiveness

We define Price Expressiveness as the property that an AMM can provide liquidity at any price. A price-expressive AMM has a trading function $f$ with unit-normalized price $P$ that assumes all values in the unit $k$-simplex on any level curve $C$. That is, if we denote the unit simplex on $k$ dimensions as $\Omega^k = \{ P \in \mathbb{R}^k_+ \mid \sum_{j=1}^{k} P_j = 1 \}$, we have,

$$\forall P \in \Omega^k, C \in \mathbb{R}_+, \exists X' \in \mathbb{R}^k_+: \frac{\nabla f(X')}{\| \nabla f(X') \|_1} = P \wedge f(X') = C$$

Because prices are decreasing with respect to each asset and the trading function has a smooth gradient, price expressiveness implies that prices approach infinity as reserves of an asset approach zero:

$$\lim_{X_i \to 0} P_i = \infty$$

Not all AMMs exhibit price-expressiveness. Namely, the CSMM has fixed price so clearly doesn’t exhibit price expressiveness, while the CPMM does. In this thesis, we reason about both AMMs exhibiting this property and AMMs that do not.
2.3.2 Concavity of the trading function

We define a convex function \( f : \mathbb{R}^k \to \mathbb{R} \) as a function that fits the following property:

\[
f(tX + (1-t)Y) \leq tf(X) + (1-t)f(Y), \quad t \in [0,1]
\]

A convex set is a set where the above condition holds on its boundary. A strictly convex set satisfies the strict inequality. Another interpretation of convexity is that the tangent line always lies below the curve. Concave functions and sets reverse this equality, so that tangent lines lay above the curve.

Concavity is an important property of the trading function which ensures non-decreasing marginal prices. Given \( f \) is differentiable, we can take its gradient \( \nabla f(X) \), and \( f \)'s concavity implies [5]:

\[
f(X + \Delta) \leq f(X) + \nabla f(X)^T \Delta
\]

It follows that the reachable set, the set above \( f \), is convex. These concavity and convexity conditions are fundamental properties of AMMs that allow AMMs to adapt to different prices and prevent phenomena such as runaway trading. AMMs with trading functions with strictly decreasing gradients, such as the CPMM, have strictly convex reachable sets, meaning the above inequality is strict. CSMMs on the other hand are convex but not strictly convex, as they satisfy the equality.

2.3.3 Trading Functions are \( k \)-Homogeneous

We define a function \( f \) as \( k \)-homogeneous if it fits the following condition:

\[
f(\alpha X) = \alpha^k f(X)
\]

We can convert any \( k \)-homogeneous trading function to a 1-homogeneous trading function by taking its \( k \)th root. Thus, we may accept any \( k \)-homogeneous trading function. For the purpose of our analysis, we assume that our trading function is 1-homogeneous. In terms of AMMs, a 1-homogeneous trading function gives us the simple property that scaling our reserves scales our trading function by the same amount. Angeris et. al. argue that this is a required property for any AMM to exhibit “reasonable” behavior [5]. Many trading functions, if not at first 1-homogeneous, can be transformed into 1-homogeneous trading that
are functionally equivalent (prices are the same). For example, the constant product AMM, defined by

\[ f(X) = \prod_{j=1}^{k} X_j \]

can be transformed into a 1-homogeneous function by taking the \( k \)-th root, turning it into a constant geometric mean:

\[ f(X) = \left( \prod_{j=1}^{k} X_j \right)^{1/k} \]

so that

\[ f(\alpha X) = \left( \alpha^k \prod_{j=1}^{k} X_j \right)^{1/k} = \alpha \left( \prod_{j=1}^{k} X_j \right)^{1/k} \]

### 2.4 Network of AMMs

This thesis concerns the scenario when we have a “network” of \( n \) AMMs, similar to the models in [13, 4]. Our network of AMMs can be thought of either in a vacuum, where these AMMs determine market prices, or as a subset of a larger market, where stable prices are determined externally from this network. Each scenario requires careful consideration. We define our network of AMMs \( A_i \) where \( i = 1 \ldots n \), each trading a subset of \( k \) tokens. We denote AMM \( A_i \)'s reserves as \( X_i \), where \( (X_i)_j \) are the reserves for token \( j \). And we denote each AMM’s trading function as \( f_i : \mathbb{R}_+^k \rightarrow \mathbb{R} \).

While not the main focus of this thesis, an interesting framework to view such a network of AMMs is as a population protocol, as researched by Maurice Herlihy in an upcoming paper [13]. A population protocol is a framework, popularized in distributed computing and mobile networks. In a population protocol, agents meet up in in pairs randomly, each starting with initial values and eventually converging to some stable value [6].
Chapter 3

Arbitrage

Before we discuss defensive rebalancing, it is important to discuss the role of arbitrage in DeFi and its impact on AMMs.

3.1 The Menace of Arbitrage

While it is easy to think of arbitrage as a “bad” in the eyes of LPs, arbitrageurs serve a vital role and have several benefits from the perspective of LPs. First of all, arbitrageurs align prices with broader market price to arrive at a “trustworthy” price where uninformed traders will not trade at a loss. It is because of arbitrageurs that AMMs do not need access to a central market price feed to provide liquidity at fair prices. In addition, arbitrage in effect increases the liquidity of AMMs at the stable price. Consider an AMM that processes a large price that incurs price slippage, disrupting balances of reserves. An arbitrageur will notice this price discrepancy and return the AMM to a stable price, restoring the AMM’s liquidity at the market stable price. In effect, the arbitrageur moves assets between platforms, even across chains [15], to ensure an efficient market.

An arbitrageur will act on an arbitrage opportunity whenever it is profitable. Profitability for arbitrageurs depends on on several conditions. Milionis et. al. [17] study the effect of block time on arbitrage profits in the presence of fees and find that faster block times correspond to smaller arbitrageur profits. The same paper defines the variance of prices in different market conditions, viewing the decentralized market as a stochastic process. Rao
and Shah [19] propose a new fee structure, the “triangle fee structure” that lowers fees for large trades, encouraging arbitrage trades and also keeping fees high for small “noise” trades. The motivation is that when there are high trading fees, prices need to be misaligned by a sometimes significant amount to make the arbitrage profit outweigh the trading fees (called the no-trade region [17]). By lowering fees for large trades where there is non-negligible price slippage, arbitrage trades can align prices much closer to market prices. While fees have a relevant impact on arbitrageurs, we ignore fees in our analysis, as mentioned before, to preserve the “path-independence” property of the reachable set for each AMM.

3.2 Classes of Arbitrage

In general, we can categorize arbitrage trades involving DEXs into two categories: (1) CEX-DEX arbitrage. The arbitrageur exploits an outdated DEX price with a trade that aligns the DEX price with the central market price. (2) DEX-DEX arbitrage. The arbitrageur exploits the price gaps between two DEXs. Arbitrageurs solve the optimal routing problem (3.4) to perform cyclic arbitrage in a sequence of trades between DEXs [22]. DEX-DEX arbitrage has the benefit that profits are held completely on-chain, while CEX-DEX arbitrage fragments the trade into two trades: one on the CEX platform, and one on the DEX platform. This fragmentation means the arbitrageur must have enough funds in their wallet to fund the transaction. DEX-DEX arbitrageurs on the other hand doesn’t require a deep pocket. They can use flash loans (loans that must be paid back by the end of the on-chain transaction) to fund their transaction [21]. Depending on market conditions, these two types of arbitrage may be profitable for arbitrageurs to varying degrees.

3.3 Arbitrage Between two CPMMs

We outline an example of DEX-DEX arbitrage between two AMMs $A_1$ and $A_2$, in the simple case of a two-token CPMM. The AMMs have balances $(T_1, D_1)$ and $(T_2, D_2)$, and constant products $C_1$ and $C_2$. If $A_1$ and $A_2$ have different spot prices ($T_1/D_1 \neq T_2/D_2$), there is an arbitrage opportunity. Choose $A_1$ to have greater value for tokens, so $T_1/D_1 < T_2/D_2$. Say the arbitrageur wishes to take their profits in dollars. The arbitrageur makes one trade that depends on the variables $\Delta T_1, \Delta D_1, \Delta T_2, \Delta D_2 \in \mathbb{R}_+$, where the arbitrageur inputs $\Delta D_1$ dollars and receives $\Delta T_1$ tokens from $A_1$, and inputs $\Delta T_2$ tokens and receives $\Delta D_2$ dollars
from $A_2$. Our constant product post-arbitrage is the following:

$$(T_1 - \Delta T_1) \times (D_1 + \Delta D_1) = C_1$$

$$(T_2 + \Delta T_2) \times (D_2 - \Delta D_2) = C_2$$

Since the arbitrageur is taking their profits in dollars, the number of tokens stays the same, meaning $\Delta T_1 = \Delta T_2$. The arbitrageur seeks the value $\Delta D_1$ that maximizes its profit, $\Delta D_2 - \Delta D_1$. The arbitrageur maximizes profit by equalizing prices such that:

$$\frac{T_1 + \Delta T_1}{D_1 - \Delta D_1} = \frac{T_2 - \Delta T_2}{D_2 + \Delta D_2}$$

Given the price-equality constraint, the unique solution to this optimization problem can be solved algebraically [13]:

$$\Delta D_1 = \frac{\sqrt{C_1}}{T_1} \frac{\sqrt{C_2} T_1 - \sqrt{C_1} T_2}{T_1 + T_2}$$

Where the arbitrageur makes a profit of

$$\Delta D_2 - \Delta D_1 = \frac{(\sqrt{C_2} T_1 - \sqrt{C_1} T_2)^2}{T_1 T_2 (T_1 + T_2)}$$

After arbitrage, the AMMs trade at the spot price

$$\frac{T_1'}{D_1'} = \frac{T_2'}{D_2'} = \left(\frac{T_1 + T_2}{\sqrt{C_1} + \sqrt{C_2}}\right)^2$$

In this example, the arbitrageur chooses to take its profits entirely in dollars. Assuming the population protocol model, where an arbitrageur performs these arbitrage trades until price convergence and continues to keep its profits in dollars, we can derive the equilibrium price. Since the arbitrageur is removing dollars, $C_i$ and $\sum_{i=1}^n T_i$ remain the same for all $i = 1 \ldots n$, so we can derive our final price for a given $A_i$.

$$p_i = \frac{T_i}{D_i} = \frac{T_i^2}{C_i}$$

While $T_i$ change throughout arbitrage, since $\sum_{i=1}^n T_i$ remains constant we can derive the final stable price

$$p_{\text{stable}} = \frac{\sum_{i=1}^n T_i^2}{\sum_{i=1}^n C_i}$$

In this example by imagining a sequence of pairwise arbitrage trades, we follow the population
protocol model, where the arbitrageur executes each trade in a vacuum, with no knowledge of a central price. Depending on the arbitrageur’s preference, the stable price will be different. [13] goes into more depth on the arbitrageur’s potential for price manipulation and shows that the price will converge in $O(\max(n^2 \log n, \log \frac{1}{\epsilon}))$ arbitrage steps. It is important to remember that this follows the population protocol model and assumes the lack of a central market. However, an arbitrageur with access to a stable market would bring each $A_i$ to the stable price immediately (after $n$ interactions).

### 3.4 The Optimal Routing Problem

Now, we move on from arbitrage between just two AMMs. Instead, they have visibility into our entire network of AMMs and their goal is to execute a sequence of trades that maximizes their profit. The general class of problems involving maximizing profit for an arbitrageur is the *optimal routing problem*, which has been studied by Angeris et. al. [4]. To define this function, we introduce a *utility function* that the arbitrageur seeks to maximize. We present the problem formulation as follows, inspired by [4]:

The **Optimal Routing Problem** is the problem where an arbitrageur seeks to maximize their utility based on a self-defined utility function $f(Y) : \mathbb{R}^k_+ \rightarrow \mathbb{R}_+$. The variable we seek to optimize is $\Delta$, which we can think of as a $n \times k$ matrix, where $(\Delta_i)_j$ is the change in reserves of token $j$ for AMM $A_i$. Positive values of $\Delta$ mean the AMM receives this asset as an input, and negative values of $\Delta$ mean the AMM outputs this asset in the trade.

\[
\text{maximize} \quad f(Y) \\
\text{subject to} \quad Y_j = -\sum_{i=1}^{n} (\Delta_i)_j, \quad i = 1 \ldots n \\
f_i(X_i + \Delta_i) = f_i(X_i), \quad i = 1, \ldots, n \\
(X_i)_j + (\Delta_i)_j \geq 0, \quad i = 1, \ldots, n, \quad j = 1, \ldots, k
\]

While this problem as stated above is not convex, it is possible to restate the problem as a convex problem by replacing the trading function equality to an inequality: $f_i(X_i + \Delta_i) \geq f_i(X_i)$. However, it is easy to show that the solution must lie on the boundary of the reachable set, where $f_i(X_i + \Delta_i) \geq f_i(X_i)$. Due to the modified problem’s convexity, it can be solved using solver tools for convex optimization [4]. In our section about defensive rebalancing, we...
will compare the optimal arbitrage problem to the goals of defensive rebalancing and discuss in more depth the impact of the utility function \( u(X) \) in both scenarios.

### 3.5 Access to Stable Prices

In the previous section, we discussed market behavior when arbitrageurs lack access to central market prices, as in the population protocol model \([13]\). Here, we first discuss the behavior of an arbitrageur with access to central market prices, and second, we discuss the behavior of an AMM with access to a price oracle, providing a live feed of market prices.

#### 3.5.1 Arbitrage given a Central Market

An arbitrageur, given access to a centralized market would execute a trade that neutralizes any AMM’s price to \( p \), where \( p \) is the central market price. Let’s consider the scenario where \( A_1 \) is a two-token CPMM, trading at price \( p_1 \neq p \). The arbitrageur hopes to maximize \( p Y_T + Y_D \), where \( (Y_T, Y_D) \in \mathbb{R}^2 \) the net reserves for the trader. The AMM takes the other side of the trade, so its net reserves are \( \Delta_T = -Y_T \) and \( \Delta_D = -Y_D \). To maximize their profits, the arbitrageur sets the prices equal to \( p \):

\[
p = \frac{T + \Delta_T}{D + \Delta_D}
\]

Using this constraint and the constant product constraint \( (T + \Delta_T)(T + \Delta_D) = TD \), we can solve this algebraically and yield

\[
\Delta_T = \frac{T(pD - T)}{T + pD}, \quad \Delta_D = \frac{D(pD - T)}{T + pD}
\]

The arbitrageur’s profit and AMM’s impermanent loss is

\[
p Y_T + Y_D = -(p \Delta_T + \Delta_D) = \frac{(pD - T)^2}{T + pD}
\]

This loss comes due to the fact that the \( A_1 \) is unaware of the central market price. By design, an AMM trades at its best guess of the price at any given moment, determined by its initial reserves and past trades.
3.5.2 AMMs Given a Price Oracle

If, in some idealized world, \( A_1 \) could access the central market price via an oracle service, it could modify its trading function to account for this price shift, before arbitrageurs have the time to exploit its old prices. It can do this by hiding \( d \) of its dollars to drive up the token price as follows, assuming \( p_1 > p \):

\[
\frac{T}{D-d} = p
\]

\[
d = \frac{pD - T}{p}
\]

to create a new trading function to eliminate an arbitrage opportunity

\[
f^*(T, D) = T \times (D - d)
\]

We can also consider the scenario where \( A_1 \) has the ability to trade with the centralized exchange. Then, \( A_1 \) has the option to trade any number of its assets at the exchange rate \( p \). This way, the AMM can execute a trade such that it aligns its price with the centralized price while maintaining its trading function. Still considering the case of the two-token CPMM, our trade space is \((\Delta_T, \Delta_D) = \delta(p, -1)\), where \( \delta \in \mathbb{R} \). Thus, we choose the value of \( \delta \) that maximizes its trading function \( f(T, D) = (T_i + p\delta)(D_i - \delta) \):

\[
\frac{du}{d\delta} = -2\delta p + Dp - T_i
\]

\[
2\delta p = Dp - T
\]

\[
\delta = \frac{Dp - T}{2p}
\]

Using this value for \( \delta \), our new reserves are

\[
(T + p\delta)(D - \delta) = \left( T_i + \frac{Dp - T}{2}, D_i - \frac{Dp - T}{2p} \right)
\]

There are several problems with this in practice. First of all, while services such as Chainlink attempt to provide accurate price feeds, the service is not perfect. Second of all, if this oracle existed, changing the price function changes the balance of reserves in the AMM, while this stability is important to the appeal of a AMMs such as the CPMM. However, this is applicable in stablecoin markets such as Stableswap, where prices are adjusted with a dynamic peg according to an “internal price oracle”[10]. In general, we can assume that
AMMs do not have access to a trustworthy external price feed. And, even if they did, there is no clear manner how by which they would have access to liquidity at market prices. In the next chapter, we discuss ways AMMs can make trades with other AMMs, who also hold imperfect information about stable prices.
Chapter 4

Defensive Rebalancing

In a new scenario, imagine that an AMM $A_1$ has the opportunity to make a trade with $A_2$. The goal for the two AMMs is to execute a trade that is mutually-beneficial. But how can we quantify a mutually beneficial trade?

4.1 A Motivating Example

To give an example of self-rebalancing, we can revisit the two-token CPMM scenario. We consider AMMs $A_1$ and $A_2$ such that $\frac{T_1}{D_1} < \frac{T_2}{D_2}$ ($A_1$ values tokens more than $A_2$ does). The goal for both AMMs is to make a perceived profit. The AMMs execute a trade $(\tau, \delta) \in \mathbb{R}_+^2$ such that the trade leaves $A_1$ and $A_2$ with reserves $(T_1 + \tau, D_1 - \delta)$ and $A_2$ with reserves $(T_2 - \tau, D_2 + \delta)$. Since we have two free variables, we must choose price equality and one more constraint to isolate a unique solution. In this case we choose the price $p$ at which the AMMs will exchange assets, letting us parameterize $\tau = p\delta$. An infinitesimal price $p$ where $\frac{T_1}{D_1} < p < \frac{T_2}{D_2}$ would be profitable for each AMM, since $A_1$ would be receiving tokens for a price less than $p$, and $A_1$ would be selling tokens for a price greater than $p$. However, at some value $\delta$, the AMMs’ prices would meet and this exchange would cease to be profitable. Let us choose $p = \frac{T_1 + T_2}{D_1 + D_2}$, so that $p$ is between the two AMMs’ prices. We now solve (as in [13]):

$$\frac{T_1 + \delta \frac{T_1 + T_2}{D_1 + D_2}}{D_1 - \delta} = \frac{T_2 - \delta \frac{T_1 + T_2}{D_1 + D_2}}{D_2 + \delta}$$
yielding
\[ \delta = \frac{D_1 T_2}{2(D_1 + D_2)} \]

However, not every rebalancing trade is made equal. In our example, depending on the chosen exchange price \( p \), each AMM would end up in different states, though they would end at the same stable price (see 4.6.1). Our choice of \( p \) impacts the relative utility for each AMM in the rebalancing. Values of \( p \) closer to \( \frac{T_2}{D_2} \) favor \( A_1 \) more, whereas values of \( p \) closer to \( \frac{T_1}{D_1} \) give a better deal to \( A_2 \). In this chapter, we will discuss different notions of fairness in self-rebalancing trades.

4.2 Problem Statement

Here, we present the Optimal Rebalancing Problem. We informally define a self-rebalancing trade to be any series of exchanges between two or more AMMs where each AMM ends up in a better state. Now we present our formal definition.

Given \( n \) AMMs each trading \( k \) tokens, a self-rebalancing trade can be represented by \( \Delta \in \mathbb{R}^{n \times k} \), where each row, a vector denoted by \( \Delta_i \), represents the change in assets for \( A_i \).

A self-rebalancing trade \( \Delta \) follows two conditions:

- No asset exits the network of AMMs. This means that the each column in \( \Delta \) sums to zero:
  \[ \sum_{i=0}^{n} (\Delta_i)_j = 0, \quad j = 1, \ldots, k \]

- Each AMM ends up within their reachable set:
  \[ f_i(X_i + \Delta_i) \geq f_i(X_i), \quad i = 1, \ldots, n \]

4.2.1 Why is this difficult?

One might ask: We already know how to solve the optimal routing problem (3.4), which captures all arbitrage profit, so why don’t we just perform arbitrage on our network of AMMs and distribute the profits?

At first glance, this would make sense. Our rebalancing service would pick some utility
function $f$ which they use to solve the optimal routing problem (3.4), ending up with some output vector $\Delta \in \mathbb{R}^k$ which is the opposite net network flow of the AMMs. But the question arises of how we might distribute these assets to the LPs.

### 4.3 Optimal Rebalancings

Since assets can be divided into tiny denominations ($1$ ETH = $10^9$ gwei), we can treat the number of ways we could redistribute assets (rebalancings) as infinite. But not rebalancings are created equal, and we are curious about rebalancings that maximize welfare for participants.

We define an **Optimal Rebalancing** as an AMM rebalancing that is *undominated*. That is, given a rebalancing $\Delta \in \mathbb{R}^{n\times k}$, there does not exist another rebalancing $\Delta' \in \mathbb{R}^{n\times k}$ such that $f_i(X_i + \Delta'_i) > f_i(X_i + \Delta)$ for all $i = 1, \ldots, n$. In other words, if we have an optimal rebalancing, we cannot improve it such that each AMM is better off. It follows that if we begin with an optimal rebalancing, any subsequent self-rebalancing trade that increases $A_i$’s utility must decrease some $A_j$’s utility. We can use the properties of our trading function to prove that any price-equalizing rebalancing is undominated. You may find the proof in A.1. Also, any rebalancing leaving AMMs with different spot prices can be dominated with a rebalancing including a subsequent trade somewhere between the two different spot prices.

Thus, rebalancings are optimal (undominated) *if and only if* AMMs end up with equal spot prices.

Since spot prices are defined by $\frac{\partial f}{\partial X_i} / \frac{\partial f}{\partial X_j}$, Price equality gives the following condition for any two AMMs $A_i, A_j$ post-rebalancing:

$$\nabla f_i(X_i) = \lambda \nabla f_j(X_j), \quad \lambda \in \mathbb{R}_+ \quad (4.1)$$

In other words, equal prices across AMMs means the gradients of their trading functions are scalar multiples of each other.
4.3.1 Same Trading Function

If all AMMs have the same trading function $f$, we could use $f$ as our utility function in the Optimal Routing Problem, and the resulting $\Delta$ would be proportional to each $X_i$ after arbitrage, such that

$$\Delta = \lambda_i X_i, \quad i = 1, \ldots, n, \quad \lambda_i \in \mathbb{R}_+$$

Then, we could divide $\Delta$ into proportional pieces $\Delta_i$ according to some agreed distribution $\eta \in \mathbb{R}^k$:

$$\Delta = \sum_{i=1}^n \Delta_i, \quad \Delta_i = \eta_i \Delta, \quad \sum_{i=1}^n \eta_i = 1$$

So $A_i$’s new balance would be $X_i + \Delta_i$. This works because each $f$ is 1-homogeneous and since $\Delta$ is proportional to $X_i$, the change in reserves is equivalent to multiplying $X_i$ by the scalar $1 + \| \frac{\Delta_i}{X_i} \|_1$, and scaling $f$’s input yields a constant gradient, meaning prices do not move. Thus, the assets can be easily distributed, so this is a reasonable solution to the Optimal Arbitrage Problem. However, we are curious about the more complex scenario, where our network of AMMs exhibits non-equivalent trading functions.

4.3.2 Different Trading Functions

Unfortunately, this strategy is not a one-size-fits-all solution. To start off, how do we choose the utility function to maximize in the Optimal Rebalancing Problem? The utility function is crucial, since it affects the post-arbitrage price of the AMMs. We could choose to use a set of market prices, denoting a constant value, similar to a CPMM, or we could use a constant-product utility function. Any way we do this, the resulting $\Delta$ which maximizes the chosen utility function would not be necessarily proportional to our $X_i$’s after arbitrage. Thus, we could not distribute assets to each LP’s reserves without modifying their price vector from its stable state.

We could, instead of adding assets to the AMM’s reserves, choose to distribute these rebalancing “rewards” into some vault, separate from each AMM’s tradable reserves, as a potential solution. However, in this thesis, we are interested in the rebalancing rewards going directly to the liquidity pool, creating a system where each participating AMM’s liquidity would increase with each rebalancing, and rewards would be distributed naturally, with a proportional increase in the value of LP tokens.


4.4 Space of Rebalancings

In the previous section, we discussed why finding the optimal rebalancing given a set of AMMs is nontrivial. Here, we define the space of all optimal rebalancings.

4.4.1 Example: Two CPMMs

In 4.1, we show merely one example of a self-rebalancing between two CPMMs. In our example, we were optimizing the variables $(\tau, \delta) \in \mathbb{R}^2_+$ and arbitrarily chose our exchange rate $p = \frac{T_1 + T_2}{D_1 + D_2}$, but we really could have made any choice for $p$ and solved for the AMMs’ reserves with prices matching $p$. Given that $\frac{T_1}{D_1} < \frac{T_2}{D_2}$, any $(\tau, \delta)$ where $\frac{T_1}{D_1} < \frac{T_1 + \tau}{D_1 - \delta}$ will be an undesirable trade for $\mathcal{A}_1$ because they will be accumulating asset $T$ for less than their spot price $\frac{T_1}{D_1}$. Similarly, if $\frac{T_1}{D_1} > \frac{T_2}{D_2}$, $\mathcal{A}_2$ will be accumulating asset $D$ for less than $\frac{D_2}{T_2}$.

We can draw a curve in the space of possible rebalancings. We let $\tau \in (0, T_2)$ and $\delta \in (0, D_1)$. Thus, given any $\delta$, there is a unique $\tau$ such that prices stabilize. This is because $P_1(T, D)$ is decreasing with respect to $\tau$, and $P_2(T, D)$ is decreasing, and both functions range over $\mathbb{R}_+$, so there must be some $\tau$ where $P_1(T_1 + \tau, D_1 - \delta) = P_2(T_2 - \tau, D_2 + \delta)$ by the Intermediate Value Theorem.

To our point in the first paragraph, not all rebalancings in this graph are contained in the reachable sets of both AMMs; only rebalancings at the exchange rate between the original spot prices, a segment of the curve, would represent acceptable trades for both AMMs.

4.4.2 Generalizing

We now discuss properties of optimal rebalancings in the generalized, $n$-AMM, $k$-token model. Given any stable price vector $P$, due to the homogeneity of the trading function,

$$\nabla f_i(X) = \lambda_i P, \quad \lambda \in \mathbb{R}_+, \quad i = 1, \ldots, n, \quad \lambda_i \in \mathbb{R}_+$$

We can scale $X$ and its prices will not change:

$$\nabla f_i(\eta X) = \eta \lambda_i P$$
In addition, if we have the property that the trading function $f$ is strictly concave, $\nabla f_i(X)$ is invertible, meaning the direction of $X$ is a function of $P$. Let us call this function $\mathcal{X}(P)$:

$$\mathcal{X}(P) = \{ X \in \mathbb{R}^k_+ \mid \nabla f_i(X) = \lambda P, \ \lambda \in \mathbb{R}_+ \}$$  \hspace{1cm} (4.2)

$\mathcal{X}(P)$, when $f$ is strictly concave, is a line in $k$-dimensional space. We can normalize this to the unit vector, which gives us a unique vector, not a set.

$$\mathcal{X}^*(P) = \frac{\mathcal{X}(P)}{\|\mathcal{X}(P)\|_1}$$  \hspace{1cm} (4.3)

And, given any price $P$, any price-equalizing rebalancing must satisfy

$$X_i + \Delta_i = \upsilon_i \mathcal{X}^*_i(P)$$

$$\sum_{i=1}^{n} \Delta_i = 0$$

Thus, we can think of solving an optimal rebalancing as finding a linear combination of $\mathcal{X}^*_i(P)$ that satisfies the sum-to-zero constraint for $\Delta$. Depending on our value of $P$ and our trading functions $f_i$, this linear combination may or may not have a solution. In Section 4.6.1, we showed that there is only one value of $P$ that will yield a solution when all trading functions are identical. In Figure 4.2, we give an example of a scenario where different values of $P$ yield an optimal rebalancing. A further question to be considered is the range of prices $P$ at which an optimal rebalancing is possible, given an initial state.

### 4.5 Existence

Here, we prove the existence of optimal rebalancings in two scenarios. Our first scenario is where all AMMs might not exhibit price expressiveness. Our second scenario enforces the price expressiveness condition, and we show the existence of an undominated rebalancing with equal prices.

#### 4.5.1 Without Price Expressiveness

We consider the case where one or more AMMs may not exhibit price expressiveness. An AMM without price expressiveness can have one of its assets fully depleted. Naturally, this
means there is not necessarily a price-equalizing rebalancing for all starting assets. However, we can argue that regardless, there exists an undominated rebalancing, even though it may not yield equal prices. One such rebalancing is the solution to the Equal Profit Rebalancing Problem (4.7). Due to the convexity of the problem, it has a solution, which must be undominated by definition. Therefore, there exists at least one optimal rebalancing for any combination of AMMs with or without the property of price-expressivity.

4.5.2 With Price Expressiveness

Following from above, given price expressiveness for all AMMs, there still must exist an undominated rebalancing, but we can add the property that any optimal rebalancing must stabilize prices. Our proof in A.1 states that any price-stabilizing rebalancing is undominated, but it does not prove the converse, that any undominated rebalancing stabilizes prices. In 4.1, we describe how to construct an optimal rebalancing when we have two CPMMs which trade two tokens, but with higher-dimensional AMMs, existence of a price-equalizing rebalancing is nontrivial, for reasons outlined in 4.6.2. Here, we prove the existence of a price-equalizing rebalancing given a set of $n$ price-expressive AMMs.

Given reserves $X_i$, and trading sets $S_i$ we can define a function $\phi : \mathbb{R}^{n \times k} \rightarrow \mathbb{R}^{n \times k}$ that moves prices closer to each other, by the following rule:

$$\phi(X)_{ij} = X + \alpha \cdot \left( \frac{1}{n} \sum_{\gamma=1}^{n} \frac{\nabla f_\gamma(X_\gamma)_j}{\|\nabla f_\gamma(X_\gamma)\|_1} \right)^{-1} \left( \frac{\nabla f_i(X_i)_j}{\|\nabla f_i(X_i)\|_1} - \frac{1}{n-1} \sum_{\gamma=1,\gamma \neq i}^{n} \frac{\nabla f_\gamma(X_\gamma)_j}{\|\nabla f_\gamma(X_\gamma)\|_1} \right)$$

where $\alpha > 0$ is sufficiently small. We need the full price expressiveness condition because this prevents our function from reaching negative reserves. In fact, we can use this function as an algorithm to help us find an optimal rebalancing by repeatedly applying it.

We prove that this function narrows prices and is a legal rebalancing. First, we show that the network flow is zero:

$$\sum_{i=1}^{n} \phi(X)_{ij} - X_{ij} = \alpha \cdot \left( \frac{1}{n} \sum_{\gamma=1}^{n} \frac{\nabla f_\gamma(X_\gamma)_j}{\|\nabla f_\gamma(X_\gamma)\|_1} \right)^{-1} \sum_{i=1}^{n} \left( \frac{\nabla f_i(X_i)_j}{\|\nabla f_i(X_i)\|_1} - \frac{1}{n-1} \sum_{\gamma=1,\gamma \neq i}^{n} \frac{\nabla f_\gamma(X_\gamma)_j}{\|\nabla f_\gamma(X_\gamma)\|_1} \right)$$

In the summation on the right, for each $i$, we have one value of $\frac{\nabla f_i(X_i)_j}{\|\nabla f_i(X_i)\|_1}$ from the first term, and $n-1$ values of $\frac{\nabla f_\gamma(X_\gamma)_j}{\|\nabla f_\gamma(X_\gamma)\|_1}$, which amounts to one $\frac{\nabla f_i(X_i)_j}{\|\nabla f_i(X_i)\|_1}$ for each
Thus, across the summation, our terms cancel out, meaning \( \sum_{i=1}^{n} \phi(X)_{ij} - X = 0 \). Below is a Python (with NumPy) implementation of \( \phi \):

```python
def phi(X, alpha, amms):
    n = len(X)
    grads = np.array([amm.gradient(X) for amm in amms])
    amm_grad_sums = np.sum(grads, axis=1)
    grads_normalized = grads / amm_grad_sums[:, np.newaxis]
    sum_each_token = np.sum(grads_normalized, axis=0)
    avg_excluding_self = (sum_each_token - grads_normalized) / (n - 1)
    Delta = grads_normalized - avg_excluding_self
    Delta /= (sum_each_token / n)
    return X + alpha * Delta
```

Next, we prove that \( \phi \) maps to each AMM’s reachable set. We do this by considering the directional derivative of \( f_i \) in the direction \( \phi(X)_{ij} - X_{ij} \):

\[
D_{\phi(X)_{ij} - X_{ij}} f_i(X) = \nabla f_i(X) \cdot (\phi(X)_{ij} - X_{ij})
\]

A non-negative directional derivative means the output is within \( \mathcal{A}_i \)'s reachable set, given a sufficiently small \( \alpha \). We need a sufficiently small alpha because the first-order taylor approximation holds when \( \alpha \) is small. If \( \mathcal{A}_i \) favors some asset \( j \) more than average (its normalized gradient for token \( j \) is above average for the network), \( \phi(X)_{ij} - X_{ij} \) is positive, and vice versa. The difference from the average excluding itself, \( \frac{\sum_{\gamma=1, \gamma \neq i}^{n} \nabla f_\gamma(X_\gamma)_j}{\|\nabla f_\gamma(X_\gamma)\|} - \frac{1}{n-1} \sum_{\gamma=1, \gamma \neq i}^{n} \nabla f_\gamma(X_\gamma)_j \), is scaled down by the average normalized gradient, which is less than \( \frac{\nabla f_i(X)_j}{\|\nabla f_i(X)_j\|} \) if negative, essentially scaling down negative values and boosting positive values in the directional derivative’s dot product. Thus, \( D_{\phi(X)_{ij} - X_{ij}} f_i(X) \geq 0 \). More specifically, \( D_{\phi(X)_{ij} - X_{ij}} f_i(X) > 0 \) if prices are not equal, and we have equality if \( \mathcal{A}_i \)'s price vector is the average. Thus, our function \( f \) maps AMMs from their reachable set to their reachable set. Since this removes assets with higher-than-average price and vice versa, this causes prices to move towards the average.

Now, we apply the Brouwer Fixed-Point Theorem to show that there must be a fixed point. To apply this theorem, the set of rebalancings must be convex and compact. First, the set of possible rebalancings \( (n \times k \text{ dimensions}) \) is convex because the reachable sets are convex. Second, the rebalancing set is compact because there is a finite number of tokens for each asset, meaning any AMM can have at most \( \sum_{i=1}^{n} X_{ij} \) of token \( j \). Therefore, the set of possible rebalancings is convex and compact, so we can apply the Brouwer Fixed-Point Theorem, meaning \( \phi(X) \) has a fixed point, which is a rebalancing where prices are equal.
Thus, there exists a price-equalizing rebalancing for any set of \( n \) AMMs trading \( k \) tokens.

### 4.6 Price Uniqueness

Depending on the trading functions of the AMMs in the network, the stable price may or may not be unique.

#### 4.6.1 Identical Trading Functions

We claim that any optimal rebalancing must end up at the same stable price if all AMMs use the same trading function \( f \) with the required properties of *strict* concavity.

\( f \)'s strict concavity implies the existence of a function \( \lambda^*(P) \) as defined in Equation 4.3, which gives us the direction of the reserves as a function of the price vector \( P \). Any optimal rebalancing must end up at *some* stable price \( P^* \). We can derive the optimal reserve unit vector as \( X^* = \lambda^*(P^*) \), which means that all AMMs end up in the state \( X^*_i = \lambda_i X^*, \lambda_i \in \mathbb{R}_+ \), where \( i \) denotes the specific AMM in the rebalancing network.

If we imagine there were some other stable price, \( P^{**} \neq P^* \), such that \( X^{**} = \lambda^*(P^{**}) \) and \( X^{**}_i = \alpha_i X^{**}, \alpha_i \in \mathbb{R}_+ \), it follows that \( X^{**}_i \neq \eta X^*_i \) (the reserves must be non-proportional since they yield different prices given the same trading function \( f \)). Then, there must be

![Figure 4.1: Rebalancings with identical AMMs lead to unique stable price](image-url)
some assets $a, b$ such that ($X_a$ denotes the amount of asset $a$):
\[
\frac{X_a^*}{X_b^*} \neq \frac{X_a^{**}}{X_b^{**}}
\]

However, the sum of all of asset $a$ in the network of AMMs is constant:
\[
\text{Total}_a = X_a^* \sum_{i=1}^n \lambda_i = X_a^{**} \sum_{i=1}^n \alpha_i
\]
\[
\text{Total}_b = X_b^* \sum_{i=1}^n \lambda_i = X_b^{**} \sum_{i=1}^n \alpha_i
\]
\[
\frac{\text{Total}_a}{\text{Total}_b} = \frac{X_a^* \sum_{i=1}^n \lambda_i}{X_b^* \sum_{i=1}^n \lambda_i} = \frac{X_a^{**} \sum_{i=1}^n \alpha_i}{X_b^{**} \sum_{i=1}^n \alpha_i}
\]

So, we arrive at a contradiction, implying the stable price $P^*$ is unique:
\[
\frac{X_a^*}{X_b^*} = \frac{X_a^{**}}{X_b^{**}}
\]

### 4.6.2 Different Trading Functions

If our network of AMMs consists of different trading functions, this property of price uniqueness does not necessarily hold. For example, consider a network of two StableSwap AMMs with price coefficients $p_0 = (3, 1)$ and $p_1 = (1, 3)$, and leverage 1, and initial assets $X_0 = (30, 60)$ and $X_1 = (60, 30)$. We see two different price-equalizing rebalancings in Figure 4.2.

\[
f_1(X) = \sum_{j=1}^k X_j, \quad f_2(X) = \left( \prod_{j=1}^k X_j \right)^{1/k}
\]

Clearly, given varying trading functions, stable prices after rebalancing are not guaranteed to be equal. In further research we hope to formally define this range of stable prices, which is likely some simply connected subset of the unit $k$-simplex.

### 4.7 Trading Functions as Utility

An arbitrageur will always maximize its profit by minimizing an AMM’s portfolio value, defined by $p^T X$, where $p \in \mathbb{R}_+^k$ is the arbitrageur’s value for each asset, generally defined
by available market prices. That is, assuming some market price $p$, the value of an AMM's portfolio is the value of its reserves post-arbitrage. We can quantify our post-arbitrage portfolio value with the "portfolio value function", introduced by Angeris et. al. [2]:

$$V(k,p) = \inf \{ p^T X' | f(X') \geq k, X' \in \mathbb{R}^k_+ \} \quad (4.4)$$

So the stable reserves $X$ are:

$$X = R(k,p) = \arg \min_{X'} \{ p^T X' | f(X') \geq k \} \quad (4.5)$$

Since $f$ is strictly increasing with respect to its reserves and $p$ is a positive vector, the stable reserves $X$ will be at the boundary of $S$, defined by the level set $f(X) = k$. If our reserves change such that our level set to some $k' > k$, we arrive at a new reachable set $S'$, a proper subset of $S$. Since $S'$ does not contain the boundary of $S$, the stable reserves $X$ for $S$ are not in $S'$. Thus, $V(k',p) > V(k,p)$ whenever $k' > k$.

So we can define the LP’s non-normalized profit as a function of our new and old level curves and $p$:

$$\text{Profit}(k',k,p) = V(k',p) - V(k,p) = p^T (R(k',p) - R(k,p))$$

Here, profit is the change in portfolio value if the liquidity provider were hypothetically able to cash out at price vector $p$, the AMM’s best guess of market prices. Note that profit is positive regardless of the value of $p$. This means that any change in reserves $\Delta$ with the property that $f(X + \Delta) > f(X)$ yields a profit for the liquidity provider, so we can use our
function $f$ as a utility function. In 4.8, we connect changes in $f$ to profit.

### 4.7.1 Normalizing the Trading Function

Using $f$ as a utility function requires special care, because we want proportional changes in utility to be equivalent to changes in profit. Consider the example where we have the two trading functions, $f_1(T,D) = TD$, and $f_2 = (TD)^2$. We can easily show that these two AMMs are equivalent:

$$P_2(T,D) = \frac{\partial f_2(T,D)}{\partial D} \frac{\partial f_2(T,D)}{\partial T} = \frac{2DT^2}{2TD} = \frac{T}{D} = P_1(T,D)$$

While their spot prices are equal, $f_1$ and $f_2$ scale differently with respect to changes in reserves. In this case,

$$\nabla f_1(T,D) = (T,D) \quad \text{and} \quad \nabla f_2(T,D) = (2DT^2,2TD^2)$$

If we double the reserves of $A_1$, we get $f_1(2T,2D) = 4TD$ and $f_2(2T,2D) = 16(TD)^2$, yielding a 4X utility change for $A_1$ and a 16X utility change for $A_2$ (in this case, scaling reserves by a multiple gives $f_1$ the square root of the utility change of $f_1$). We must figure out a way to normalize our utility function so that we can compare utility changes fairly.

For this reason, we enforce the property of **1-homogeneity** in 2.3.3, which ensures that $f$ scales proportionally to changes in reserves. $f_1$ and $f_2$ are 2-homogeneous and 4-homogeneous, respectively, and we can scale them by taking their square and 4th roots to yield our transformed trading function, $f'_1(T,D) = f'_2(T,D) = \sqrt{TD}$, which is 1-homogeneous.

### 4.8 Connecting Profit to Monetary Value

If we hope to use $f$ as a utility function, it would be very helpful for changes in utility to correspond linearly with monetary value. That is, if $f(R') = k'$ and $f(R) = k$, we want to find out the impact of the proportion $\frac{k'}{k}$ on profit, as defined in 4.7.

We define our portfolio value function $V(k,p)$ and stable reserves $X^*$ as the reserve vector
that realizes the minimum dot product $p^T X^*$ where $X^*$ lies in the AMM’s reachable set:

$$X^* = \arg\min_{X \in \mathbb{R}^k_+} \{ p^T X | f(X) \geq k \}$$

$$V(k, p) = p^T X^*$$

Suppose that $X^*$ is an AMM’s arbitrage-free reserves allocation given $f(X) = k$. Due to the 1-homogeneity of $f$:

$$f(\frac{k'}{k}X^*) = \frac{k'}{k}k = k'$$

We argue that $\frac{k'}{k}X^*$ is the optimal reserves given the new reachable set defined by $k'$. The cost of $\frac{k'}{k}X^*$ is $\frac{k'}{k}(p^T R^*)$. Assume that there exists some $X'$ such that $f(R') \geq k'$ and $p^T X' < \frac{k'}{k}(p^T R^*)$. We can assume $f(X') = k$, since $f$ being monotone increasing means $f(X') > k$ implies we can scale it down to $X'' < X'$ element-wise such that $p^T X'' < p^T X' < \frac{k'}{k}(p^T X^*)$.

Then, we can scale $X'$ down by the factor $\frac{k}{k'}$ such that $f(\frac{k}{k'}X') = \frac{k}{k}k = k$, meaning $R^*$ is not the arbitrage-optimal reserves for $f(X) = k$. This is a contradiction. Thus, $V(k, p)$ scales linearly with respect to $k$. Thus, we can break $V(k, p)$ into

$$V(k, p) = k \cdot \phi(p)$$

So since $V(k, p)$ represents monetary value given price vector $p$, we can write our change in $V$ (our monetary gain from moving from $k$ to $k'$) as

$$\Delta V(p) = \phi(p)(k' - k)$$

To find $\phi(p)$, we can double $k$ by doubling $X$ and see the factor by which $V(c) = p^T X$ increases:

$$\phi(p) = \frac{p^T X}{k}$$

so

$$\text{Profit}(k', k, p) = \frac{p^T X}{k}(k' - k) = p^T X\left(\frac{k'}{k} - 1\right)$$

### 4.9 Optimal Rebalancing via Convex Optimization

Here, we describe how we can use convex optimization to efficiently solve the optimal rebalancing problem, given various definitions of optimality.
4.9.1 Profit Equalizing Rebalancing

To maximize profit per AMM under an equal-profit condition, we can formulate a convex optimization problem as follows. We are optimizing the variable $\Delta$, where $\Delta_i$ affects $k' = f_i(X_i + \Delta_i)$. We arbitrarily maximize the profit of $A_1$ because we have the constraint that the profit for each AMM is the same. We could just as easily optimize the sum of all profits, or moreover some subset of them and yield the same results. In Figure 4.3, we see the movement of assets and level sets under this rebalancing strategy. A Python script that solves this problem is included in Section A.2.

\[
\begin{align*}
\text{maximize} & \quad \text{Prof}_1(k'_1, k_1, p_1) \\
\text{subject to} & \quad X_{ij} + \Delta_{ij} \geq 0, \quad \text{for } i = 1, \ldots, n, \quad j = 1, \ldots, k \\
& \quad \sum_{i=1}^{k} \Delta_{ij} = 0 \quad \text{for } j = 1, \ldots, k \\
& \quad \text{Prof}_i(k'_i, k_i, p_i) - \text{Prof}_1(k'_1, k_1, p_1) = 0 \quad \text{for } i = 2, \ldots, n
\end{align*}
\]
4.9.2 Mapping onto the Optimal Routing Problem

In the Optimal Routing Problem (3.1), the arbitrageur optimizes the utility of the net network flow. We can map this onto 4.7 by pretending the arbitrageur is an AMM ($\mathcal{A}_1$ for simplicity) and removing the condition that profits are equal. We can also think about this same problem in another light, where an AMM performs rebalancing selfishly by maximizing their own value. We get the same results as in the optimal routing problem with the utility function $u$ replaced with $f_1$, since $\text{Prof}_1(k'_1,k_1,p_1)$ is a monotone increasing function in terms of $f_1(X_1 + \Delta_i)$. However we must add the constraint that the other AMMs’ trading functions assume a net positive.

4.9.3 Maximizing Total Welfare

If we are rolling out a self-rebalancing service, one goal might be to maximize the total profit of the rebalancing, represented as the sum of the profits for each AMM in the rebalancing. This strategy makes sense in practice: the more valuable this rebalancing is for the AMMs in total, the more the AMMs would be able to pay in total for fixed costs like gas.
Figure 4.5: Two examples of total-profit maximizing rebalancing

When thinking of this rebalancing vs arbitrage “war” as a zero-sum game (a dollar made for an arbitrageur is a dollar lost for the LP), it is easy to think of the monetary value of the arbitrage opportunity as equal to the sum of welfare for each AMM, which we quantify using our Profit function. Under that perspective, any optimal rebalancing has the same total welfare since it eliminates an arbitrage opportunity. However, this notion of total welfare depends on our individual price vector.

Because we define our Profit based on the value function (Equation 4.4), which depends on a price vector $p$, welfare in this context is in the eyes of the beholder (the individual AMM or arbitrageur). Thus, profit maximization is not a zero sum game, and depending on our AMMs’ trading functions and initial states, a profit-maximizing rebalancing may favor one AMM’s trading function over another. In Figure 4.5a, we see the range of rebalancings over two StableSwap curves where the x-axis represents the ratio of profits for each AMM, and the y-axis represents the profit. We can see that the sum of profit reaches its maximum at $v = 0.2$. In Figure 4.5b, the total profit reaches its maximum at $v = 1$.

In our definition of Profit (Equation 4.6), we scale profit by $\frac{p^T X}{k}$. This number is maximized when $p$ is more closely aligned with the reserves $X$, in terms of cosine similarity. Thus, because the cosine similarity of $X_1$ and $p_1$ is higher than the value for $A_0$ and $A_2$ and the trading functions are equal, Profit-Sum Maximization favors $A_1$, since it is “easier” to increase the profit for $A_1$. We are currently looking for different ways to scale Profit with respect to changes in $k$, and their impact on rebalancing.
Figure 4.6: Maximum Profit-Sum Rebalancing favors the AMM with closest-aligned initial price

4.10 Revisiting Profit

In this section, we discuss Profit (4.6) as a viable metric for utility gain in rebalancing.

4.10.1 Does Profit Work?

Profit as a metric defines the monetary value of a rebalancing trade with reference to an AMM’s original spot price $p$. That is, it ignores prices offered by other exchanges or, in other words, the fact that there might be a stable price that is not $p$. Several factors might cause its spot price to be different from the rest of a group of AMMs. One reason might be a large trade that incurred high slippage, meaning the AMM should correct to the consensus price. Another reason might be due to a shift in CEX prices. Movement in CEX prices would lead to a sequence of arbitrage trades that align the AMMs’ prices with CEX prices. If this happens all at once, the AMMs prices will still be the same. However, if only some of the AMMs have undergone arbitrage, there would be two cohorts of AMMs: AMMs with new (corrected) prices and AMMs with stale prices. These are just two reasons why AMMs might post different spot prices. From the perspective of an AMM $A_i$, which price should
they trust more... their original price $p$ or a price coming from other AMMs, such as a stable rebalancing price?

If an AMM were a person, they would be stubborn to use Profit as their utility function. In a network with many AMMs, the post-rebalancing price, assuming we lack some manipulative adversary, should be a better market belief than the AMM’s original price vector $p$. This is especially the case when the other AMMs in the rebalancing network comprise a large amount of liquidity because its stable price is less-easily manipulated by noise trades and by a market manipulator. The stable price is somewhere in the interior of the range of all AMMs’ initial prices, similar to a weighted average. So, in a sense, the stable price after rebalancing can be thought of as a price oracle stemming from the “wisdom of the crowd”, which in most contexts will be more accurate than the AMM’s individual belief [9].

4.10.2 Alternatives To Profit

Further research will involve studying different ways to measure utility in self-rebalancing. One possible alternative to profit uses some stable price $p'$ as the trusted price, and profit is determined by their value with respect to this new price $p'$. Then, we could define a new Profit function as

$$Profit_2(k', k, p') = V(k', p') - V(k, p')$$  (4.8)

Calculating Profit2 would require solving for the optimal reserves for $A_i$ under price $p_i$, before running the convex optimization problem. If there is a deterministic stable price $p'$, we can calculate each $A_i$’s impermanent loss with:

$$IL = p'^T X - V(k, p')$$

Then, we could hypothetically factor each AMM’s impermanent loss into profit:

$$Profit_3(k', k, p') = V(k', p') - V(k, p') - IL = V(k', p') - p'^T X$$  (4.9)

The main added complexity for incorporating example alternative profit functions Profit2 and Profit3 are the fact that they rely on knowledge of the stable price $p'$ before rebalancing. In the case where all trading functions are the same, the stable price is deterministic, so these Profit functions would be easy to apply. In other scenarios, however, this will prove more difficult, but perhaps not impossible.
Chapter 5

Practical Considerations

In a vacuum, self-rebalancing is an attractive mechanism to stabilize prices and protect against arbitrage. However, there are several challenges facing self-rebalancing in practice.

5.1 Scalability problems

A key problem facing blockchain technology is the cost of computation. Smart contract code must be run by all validators, meaning complex computation would bottleneck the entire blockchain. As a result, most blockchains use a fee structure for computation, like gas in Ethereum [12]. Gas is very expensive, and smart contract code in Solidity must be carefully optimized to save on gas fees, which can go as high as over $10 per Uniswap transaction.

Any mechanism that executes self-rebalancing would be difficult to implement fully on-chain, especially on blockchains like Ethereum where gas prices are so high. Other blockchains offer lower gas fees, such as Solana, but still, if rebalancing requires computation such as solving a convex optimization problem, computation costs would be a very serious challenge.

5.2 Implementing Rebalancing

In Section 4.2, we defined self-rebalancing as an exchange of assets between two AMMs. One natural question that arises is: how? Generally, we view AMMs as passive, where an AMM’s
state only changes by means of a trade, or via an action by a liquidity provider, such as adding or removing liquidity. Because AMMs exist as smart contracts, their code is only executed when a third party calls them in some form: either directly, or indirectly down the call stack of some smart contract initially called by a third party. Thus, it is not natural to consider a trade initiated by an AMM (or a group of AMMs).

The expected Q3 2024 release of UniswapV4 will introduce hooks, a new feature giving people the freedom to create unique pools with customized logic [1]. This new AMM platform will allow people to specify code to be run at different stages of a trade, including before the trade (beforeSwap) and after the trade (afterSwap). Since the announcement of UniswapV4, the Uniswap community has developed many hooks, equipping the AMMs with new functionality [8]. One potential implementation of self-rebalancing could be a beforeSwap UniswapV4 hook, where the rebalancing logic is executed before each trade. Alternatively, such rebalancing could occur at the beginning of the block, as in the TWAMM hook [20].

Alternatively, a form of self-rebalancing that keeps in mind computation costs would be in the form of an off-chain third-party, which looks for self-rebalancing opportunities and executes a self-rebalancing whenever there is an opportunity. In this format, the self-rebalancer would be on a level playing field with arbitrageurs, who also search for opportunities off chain and execute the arbitrage on chain. We could imagine that a group of AMM owners unite to fund this rebalancing service. It would make sense for AMM owners as long as their profits outweigh the price of the service. Dynamic pricing structures could be hypothesized to ensure LPs only pay for the service when it is beneficial for them.

In our market model of a network of \( n \) AMMs, each AMM posts potentially different spot prices and is vulnerable to arbitrage. At any given moment, there is an arbitrage opportunity equal to the solution of the Optimal Arbitrage Problem, given some utility function \( u \). The purpose of self-rebalancing is for AMMs to capture some of this profit.

### 5.3 Transaction Ordering

In an efficient market with active arbitrageurs, we assume that arbitrageurs are actively looking for opportunities. When a new block is finalized, different AMMs may post different prices, for various reasons. First of all, noise traders may incur price slippage, which may move one AMM’s price out of line with others. Also, at the beginning of each block, there
might be a change in central market prices, giving arbitrageurs the opportunity to perform arbitrage with CEX prices. In either scenario, the arbitrage opportunity is only available until someone fully neutralizes it. Transactions in a block are ordered sequentially, so if two transactions are submitted that fully execute an arbitrage the same set of AMMs, the first one in the block will get all of the arbitrage profits, and the second one will trade at the newly neutralized prices. As a result, transactions early in any given block are very valuable, because they possess the right to capture any arbitrage opportunity.

Block creators typically choose the transactions with the highest fees to include in the next block, with transactions containing the highest gas fees appearing first. This creates, in effect, a first-price auction, where arbitrageurs bid to be included first in the block. In a market with highly active arbitrageurs, the equilibrium bid price for an arbitrage opportunity will approach the actual arbitrage profit. We can imagine that if AMMs deployed self-rebalancing bots, it they would need to compete in these same gas auctions with arbitrageurs, which would be an obstacle for self-rebalancing to be effective.
Chapter 6

Conclusion

In this thesis, we present the Optimal Rebalancing Problem and present some methods for solving the problem under different optimality conditions. In addition, we present some proofs that may be useful for continued study on self-rebalancing. Defensive Rebalancing strategies outlined in this paper would be directly useful in market models where arbitrageurs have a lesser presence. In such a market, self-rebalancing would assume the arbitrageur’s vital role of stabilizing prices, with the added benefit of liquidity providers capturing this arbitrage profit for themselves. In addition, self-rebalancing poses an advantage over simply redistributing the rewards of arbitrage, since self-rebalancing profits are added directly to the liquidity pools of participating AMMs, thereby increasing their liquidity and providing a straightforward way for LPs to collect such rewards. Avenues of further study include simulating self-rebalancing in varying market conditions, implementing self-rebalancing using UniswapV4 hooks, incorporating transaction fees (fixed and dynamic) into the convex optimization problem in Section 3.1, and studying alternative measures of utility under different assumptions (4.10.2).
Appendix A

Appendix

A.1 Proof: Price-Equalizing Rebalancings are Undominated

Consider $\mathcal{A}_1$ and $\mathcal{A}_2$, trading the same $k$ tokens, with the same spot price vector. Let $\mathcal{A}_1$’s and $\mathcal{A}_2$’s reserves be $X_1$ and $X_2$, where $X_1, X_2 \in \mathbb{R}^k$. Let $\mathcal{A}_1$’s utility be defined as $f_1(X_1)$ and $\mathcal{A}_2$’s be $f_2(X_2)$. We show that any exchange increasing $\mathcal{A}_1$’s utility must decrease $\mathcal{A}_2$’s utility. We derive the gradients of $f_1$ and $f_2$ with respect to $X$:

$$\nabla f_1(X) = \left( \frac{\partial f_1}{\partial X_1}, \frac{\partial f_1}{\partial X_2}, \ldots, \frac{\partial f_1}{\partial X_k} \right)$$

and

$$\nabla f_2(X) = \left( \frac{\partial f_2}{\partial X_1}, \frac{\partial f_2}{\partial X_2}, \ldots, \frac{\partial f_2}{\partial X_k} \right)$$

Let the vector $\Delta_1 \in \mathbb{R}^k$ represent the direction of the change in assets for $\mathcal{A}_1$. $\mathcal{A}_1$ and $\mathcal{A}_2$ are exchanging assets, so $\Delta_2 = -\Delta_1$. Then, the directional derivative $D_{\Delta_1} f_1(X_1)$ in the direction $\Delta_1$ is given by the dot product:

$$D_{\Delta_1} f_1(X_1) = \nabla f_1(X_1) \cdot \Delta_1$$

Since $\mathcal{A}_1$ and $\mathcal{A}_2$ have equal prices, $\nabla f_1(X_1) = m \nabla f_2(X_2)$, where $m \in \mathbb{R}$. Let’s assume $\Delta_1$
increases utility for $A_1$, meaning:

$$\nabla f_1(X_1) \cdot \Delta_1 > 0$$

Now let's derive the change in utility for $A_2$:

$$\nabla f_1(X_1) \cdot \Delta_1 > 0$$

$$m \nabla f_2(X_2) \cdot \Delta_1 > 0$$

$$m \nabla f_2(X_2) \cdot (-\Delta_2) > 0$$

$$\nabla f_2(X_2) \cdot \Delta_2 < 0$$

Thus, when instantaneous prices are equal:

$$\nabla f_1(X_1) \cdot \Delta_1 > 0 \implies \nabla f_2(X_2) \cdot \Delta_2 < 0$$

In other words, if a trading direction $\Delta_1$ instantaneously increases utility for $A_1$, $\Delta_2 = -\Delta_1$ decreases utility for $A_2$.

Next, we must next prove that $\Delta_2, D_{\Delta_2}f_2(X_2 + t\Delta_2) < 0$ holds for all $t > 0$, as trading along $\Delta_2$ changes marginal prices (the gradient). Since an AMM's utility function must have a non-negative gradient for each token, $\nabla f_2(X_2) \cdot \Delta_2 < 0 \implies (\Delta_2)_i < 0$ for some $0 \leq i < k$. Let $N$ be the set of $i$ such that $v_i < 0$, and $P$ be the set of $i$ such that $v_i \geq 0$.

$$D_{\Delta_2}f_2(X_2 + t\Delta_2) = \nabla f_2(X_2 + t\Delta_2) \cdot \Delta_2$$

Rewrite the dot product as a sum:

$$D_{\Delta_2}f_2(X_2 + t\Delta_2) = \nabla f_2(X_2 + t\Delta_2) \cdot \Delta_2$$

$$D_{\Delta_2}f_2(X_2 + t\Delta_2) = \sum_{i=0}^{k} (\nabla f_2(X_2 + t\Delta_2))_i(\Delta_2)_i$$

$$= \sum_{i \in N} (\nabla f_2(X_2 + t\Delta_2))_i(\Delta_2)_i + \sum_{i \in P} (\nabla f_2(X_2 + t\Delta_2))_i(\Delta_2)_i$$

At $t = 0$, $\sum_{i \in N} (\nabla f_2(X_2 + t\Delta_2))_i(\Delta_2)_i > \sum_{i \in P} (\nabla f_2(X_2 + t\Delta_2))_i(\Delta_2)_i$ since $D_{\Delta_2}f_2(X_2) <$
0. Due to the convexity property of AMMs\textsuperscript{1}. In other words, token value goes down as it becomes more abundant, and up as it becomes more scarce, to prevent runaway trading:

\[ \forall i \in N, \frac{d}{dt} (\nabla f_2(X_2 + t\Delta_2))_i \geq 0 \]

\[ \forall i \in P, \frac{d}{dt} (\nabla f_2(X_2 + t\Delta_2))_i \leq 0 \]

Thus, since \( v_i \) are negative when \( i \in N \), we reverse the inequality:

\[ \forall t \geq 0, \frac{d}{dt} \left( \sum_{i \in N} (\nabla f_2(X_2 + t\Delta_2))_i (\Delta_2)_i \right) < 0 \]

And conversely when \( i \in P \), \( v_i \) are nonnegative, so we do not reverse the inequality:

\[ \forall t \geq 0, \frac{d}{dt} \left( \sum_{i \in P} (\nabla f_2(X_2 + t\Delta_2))_i (\Delta_2)_i \right) < 0 \]

Putting things together:

\[ \frac{d}{dt} (D_\Delta f_2(X_2 + t\Delta_2)) = \frac{d}{dt} \left( \sum_{i \in N} (\nabla f_2(X_2 + t\Delta_2))_i (\Delta_2)_i \right) + \frac{d}{dt} \left( \sum_{i \in P} (\nabla f_2(X_2 + t\Delta_2))_i (\Delta_2)_i \right) < 0 \]

And using our initial condition \( D_\Delta f_2(X_2 + t\Delta_2) < 0 \) when \( t = 0 \), we can let \( D_\Delta f_2(X_2 + t\Delta_2) = C, C < 0 \). For all \( t > 0 \):

\[ D_\Delta f_2(X_2 + t\Delta_2) = \sum_{i \in N} (\nabla f_2(X_2 + t\Delta_2))_i (\Delta_2)_i + \sum_{i \in P} (\nabla f_2(X_2 + t\Delta_2))_i (\Delta_2)_i < C \]

Since \( C < 0 \), for all \( t > 0 \):

\[ D_\Delta f_2(X_2 + t\Delta_2) = \sum_{i \in N} (\nabla f_2(X_2 + t\Delta_2))_i (\Delta_2)_i + \sum_{i \in P} (\nabla f_2(X_2 + t\Delta_2))_i (\Delta_2)_i < 0 \]

\[ D_\Delta f_2(X_2 + t\Delta_2) < 0 \]

So for all \( t > 0 \),

\[ f_2(X_2 + t\Delta_2) < f_2(X_2) \]

Thus, as \( \mathcal{A}_2 \) trades along vector \( \Delta_2 \), its utility \( f_2(X_2 + t\Delta_2) \) decreases. This completes our proof, since it is impossible to have any exchange vector \( \Delta_1 \) that increases an AMM’s utility such that \( \Delta_2 = -\Delta_2 \) increases \( \mathcal{A}_2 \)’s utility when \( \mathcal{A}_1 \) and \( \mathcal{A}_2 \) start at the same price. \( \square \)

\textsuperscript{1}See informal axiom 4 in “Composing Networks of Automated Market Makers”
A.2 Code: Equal-Profit Rebalancing

```python
import numpy as np
import scipy.optimize as opt

def constant_sum(p: np.array):
    p = np.array(p)
    def f(x: np.array):
        x = np.array(x)
        return p @ x
    def f_grad(x: np.array):
        x = np.array(x)
        return np.array(p)
    return f, f_grad

def constant_geo_mean():
    def f(x: np.array):
        x = np.array(x)
        return np.prod(x) ** (1 / len(x))
    def f_grad(x: np.array):
        x = np.array(x)
        return f(x) / (x * len(x))
    return f, f_grad

def stableswap(p: np.array, X: float):
    p = np.array(p)
    def f(x: np.array):
        x = np.array(x)
        return np.prod(x) ** (1 / len(x)) + X * np.sum(p * x)
    def f_grad(x: np.array):
        x = np.array(x)
        return (np.prod(x) ** (1 / len(x))) / (x * len(x)) + X * p
    return f, f_grad

AMMs = [
    constant_sum([1, 1.5, 2]),
    stableswap([1, 2, 2], 0.4),
    constant_geo_mean(),
    constant_geo_mean(),
    constant_geo_mean()
]
```
N = len(AMMs)
K = 3

AMM_funs, AMM_grads = zip(*AMMs)

constraints = []

Delta = np.zeros((N, K))
initial_quantities = np.array([
    [10, 8, 20],
    [10, 30, 20],
    [10, 15, 5],
    [10, 25, 100],
    [10, 20, 50]
])

# initial prices, normalized to sum to 1
initial_prices = np.array([
    grad(q) / np.sum(grad(q))
    for q, grad
    in zip(initial_quantities, AMM_grads)])

# initial values of trading functions
orig_ks = np.array([f(q)
    for f, q
    in zip(AMM_funs, initial_quantities)])

# amount to scale profit, in terms of increase in trading function
multipliers = np.sum(initial_prices * initial_quantities, axis=1)

def reachable_set_constraints():
    # trading function must increase
    def objective_satisfied(d, f, i):
        d = np.reshape(d, (N, K))
        return f(initial_quantities[i] + d[i]) - f(initial_quantities[i])
    cons = []
    for i, f in enumerate(AMM_funs):
        cons.append({
            'type': 'ineq',
            'fun': lambda d, f=f, i=i: objective_satisfied(d, f, i)
            })
    return cons
def nonnegative_constraint():
    # reserves must be nonnegative
    def nonnegative(d):
        d = np.reshape(d, (N, K))
        val = initial_quantities + d
        return np.min(val)
    return {
        'type': 'ineq',
        'fun': lambda d: nonnegative(d)
    }
def sums_to_zero_constraint():
    def sums_to_zero(d):
        d = np.reshape(d, (N, K))
        return -np.sum(d, axis=0)
    return {
        'type': 'eq',
        'fun': lambda d: sums_to_zero(d)
    }

# profit = proportional change in k times multiplier
def profits(d):
    new_quantities = initial_quantities + d
    new_ks = np.array([f(q) for f, q in zip(AMM_funs, new_quantities)])
    profit_per_amm = multipliers * (new_ks / orig_ks - 1)
    return profit_per_amm

# profits must be equal
def equal_profits_constraint():
    cons = []
    for i in range(1, N):
        cons.append({
            'type': 'eq',
            'fun': lambda d, i=i: profits(d.reshape((N, K)))[i] -
            profits(d.reshape((N, K)))[0]
        })
    return cons

def objective(d):
    d = np.reshape(d, (N, K))
    return -profits(d).sum()

constraints.extend(reachable_set_constraints())
constraints.append(nonnegative_constraint())
constraints.append(sums_to_zero_constraint())
constraints.extend(equal_profits_constraint())

res = opt.minimize(
    objective,
    Delta.flatten(),
    constraints=constraints
)
Bibliography


[14] Roger Lee. All AMMs are CFMMs. All DeFi markets have invariants. A DeFi market is arbitrage-free if and only if it has an increasing invariant. 2023. arXiv: 2310.09782 [q-fin.TR].


