# The Rebalancing Automated Market Maker

Miguel Ottina

Aldrin Labs (https://aldrin.com/)

April 5, 2023

**Abstract**

We introduce the Rebalancing Automated Market Maker (RAMM) which is an oracle-based AMM that is designed to greatly reduce impermanent loss while being capital efficient at the same time. Its unprecedented rebalancing mechanism ensures that the pool stays balanced and that impermanent loss is cut down. Moreover, the RAMM allows the creation of multi-asset pools with a unique and innovative design that enables liquidity providers to perform single-asset deposits and single-asset withdrawals without incurring into trading fees or price impact due to explicit or implicit trades.

## Introduction

Automated market makers constitute key building blocks for Decentralized Finance. They are widely used and appear with several different designs, which have been developed and launched by many teams since the end of 2019 [4]. These designs were introduced either for being used in particular cases or to address different issues. For example, the first version of Curve Finance's AMM [3] was designed to trade between stablecoins in an effective way, and the Uniswap v3 AMM addressed the issue of capital efficiency [1].

A well-known issue with most of the AMMs is impermanent loss. Impermanent loss —also called divergence loss— is a loss that liquidity providers might face when they provide liquidity to an AMM pool. It is defined as the difference between the value of the liquidity provider's position in the pool and the value the original position would have if the liquidity provider had never deposited it into the pool. Impermanent loss is a consequence of price change and the fact that most AMM pools are designed to rebalance their assets to some fixed proportion between their values in USD terms (usually 50:50). This implies that liquidity providers' positions will contain more of the underperforming asset and less of the asset that performs better, causing a loss with respect to simply holding the assets, which is frequently not covered by the collected fees.

An interesting approach to deal with impermanent loss is to design AMMs which take the prices from an oracle, instead of having their spot prices set by the pool balances. In this way the pool balances are not tied to the market prices of the assets, and the root cause of impermanent loss is removed. This approach was put into practice in Swaap Finance's[1] Matrix Market Maker [2] and in the Lifinity[2] protocol [5], and both teams obtained some promising results.

In this article we introduce the Rebalancing Automated Market Maker (RAMM), which takes some important ideas from Swaap Finance's Matrix Market Maker and the Lifinity protocol, but which also has an innovative design of our own, employing formulas developed by us

[1] https://www.swaap.finance/

[2] https://lifinity.io/

with the intention of making it effective while keeping the design as simple as possible. The main features of the RAMM are the following.

- Multi-asset pools.
- Spot prices given by an external oracle.
- Impermanent loss greatly reduced.
- Concentrated liquidity via a leverage parameter, which allows to perform trades with reduced price impact.
- Dynamic fees and dynamic leverage parameter, which help rebalance the pool and reduce impermanent losses.
- Innovative design of LP tokens consisting of one type of LP token for each of the pool assets. This allows to perform single-asset deposits and single-asset withdrawals without price impact or trading fees.

It is worth mentioning that our formulas for liquidity deposits and withdrawals could be integrated to other AMMs' designs and applied beyond the RAMM.

This article is organized as follows. In the first section, we describe the design and characteristics of the RAMM in general terms, which will help the reader dive into the main features of the RAMM. Then, in section 2, we show the generalities of the mathematical formulas of the RAMM, together with some comments. In section 3, we thoroughly explain the details of the implementation of the RAMM, covering all possible scenarios. We also include some examples which will help understand the main features of our design. Finally, in section 4, we state and prove several results which support different claims made throughout the article.

# 1 Overview of the RAMM

The RAMM employs Balancer's constant geometric mean formula, which is

$$\prod_{j=1}^{n} x_j^{w_j} = C \ , \tag{1}$$

where $n$ is the number of tokens of the pool, $C$ is a real number and, for each $j \in \{1, 2, \ldots, n\}$, $x_j$ and $w_j$ are respectively the balance and the weight of token $j$ in the pool. As usual, we require that $\sum_{j=1}^{n} w_j = 1$.

Unlike the weights in the Balancer's AMM, which are constant, the weights in the RAMM are variable and are adjusted before each trade so that the internal spot prices of the RAMM coincide with the market prices given by the external oracle.

Trading against the RAMM follows the standard Balancer's formulas, as we shall see.

### Leverage parameter

In order to give better prices to traders, the RAMM implements a *leverage parameter*, which will be denoted by $k$. Therefore, the actual formula that the RAMM employs is

$$\prod_{j=1}^{n} z_j^{w_j} = kC \ ,$$

where, for each $j \in \{1, 2, \ldots, n\}$, $z_j$ denotes the *virtual balance* of token $j$ in the pool, and where the liquidity parameter $C$ is defined by equation (1) using the real balances. The relation

between virtual and real balances is $z_j = kx_j$ for all $j \in \{1, 2, \ldots, n\}$. This means that a trade will be performed as if the pool had balances $z_1, z_2, \ldots, z_n$ instead of $x_1, x_2, \ldots, x_n$.

In addition, the leverage parameter is variable and is set before each trade. The idea is that the trades that help rebalance the pool will be performed with a larger leverage parameter than those that unbalance the pool further. Thus, the traders of the first case will be offered better prices than those of the second one, since in the first case the trade will be performed with more liquidity than in the second case. In a few words, the RAMM will favor trades that help rebalance the pool.

### Dynamic fees

As usual, fees will be charged in the token that goes into the pool. The RAMM has a base fee of 0.1%. However, fees will be dynamic, in a similar way as the leverage parameter is. This means that trades that help rebalance the pool will be charged lower fees than those trades which further unbalance the pool.

### Rebalancing mechanism

The RAMM's rebalancing mechanism is performed by the dynamic fees and the dynamic leverage parameter. As we explained before, trades that help rebalance the pool will be encouraged by lower fees and a lower price impact, while trades that further unbalance the pool will be charged more fees and will face more price impact.

For example, in a RAMM pool consisting of ETH, MATIC and USDT, if the pool has less ETH than it should, the trading fees for buying ETH will be higher and the leverage parameter for buying ETH will be lower than in the completely balanced situation. Therefore, traders who want to buy ETH would have to pay higher fees and face a higher price impact than in the correctly balanced situation. On the other hand, in the previous situation, traders who want to sell ETH to the pool would pay lower fees and their trades will have a lower price impact than in the balanced situation.

### Liquidity deposits and withdrawals

Each pool will have one type of LP token for each of the assets that it contains. Hence, if a pool contains ETH, MATIC and USDT, it will mint three types of LP tokens, one for ETH, one for MATIC, and the other one for USDT, which will be called LPETH, LPMATIC and LPUSDT, respectively. In each liquidity deposit, users will deposit just one asset of the pool and, for doing so, they will receive a certain amount of LP tokens of the type that corresponds to the asset they have deposited. For example, liquidity providers that deposit ETH into the pool will receive LPETH.

When users withdraw liquidity, they will receive an amount of the asset that corresponds to the type of LP tokens they are redeeming. In this way, users that deposited ETH will be able to withdraw ETH only. Of course, they can then trade ETH for any other token.

A clear advantage of this approach is that liquidity providers do not have to pay fees or face price impact when depositing or withdrawing single-asset liquidity, which does indeed occur with the classical Balancer's single-asset deposits and single-asset withdrawals, for example. Another important feature of this design is that the pool owner can disable deposits of a certain asset when needed. For example, if the pool owner decides that the pool needs a bigger proportion of a certain asset, he/she can disable liquidity deposits of all the other assets, so that the new liquidity deposits will contribute to add liquidity to the desired asset.

### Liquidity provider's earnings

As it occurs with other AMMs liquidity providers' earnings come from the trading fees, and thus liquidity providers will benefit from high trading volumes. In addition, in the RAMM the

price impact that the traders face is also an earning for the liquidity providers.

On the other hand, in contrast to which occurs with constant product AMMs and with many others AMMs, in the RAMM (and in most oracle-based AMMs) the trading volume is independent of the volatility in the prices because the internal spot prices of the RAMM follow the oracle prices and do not depend on the pool balances. Therefore, it is important that the RAMM is integrated with aggregators to capture considerable trading volume.

In addition, the observation of the previous paragraph also implies that an increase in a RAMM pool's liquidity does not imply the same level of increase in trading volume. This situation is very different from what occurs in constant product AMMs, where an increase in the pool's liquidity implies the same level of increase in the trading volume that comes from arbitrage.

Taking into account the considerations of the previous paragraph, the pool owner might eventually decide to disallow new liquidity to enter the pool by disabling deposits of every token of the pool, in order to prevent the earnings of the existent liquitity providers from getting diluted.

In spite of the previous apparent limitations, our simulations show that liquidity providers might obtain interesting earnings on top of the fact that impermanent loss is greatly reduced and even turned into impermanent gain in some cases.

## 2 Mathematical foundations

In this section, we will exhibit and explain the mathematical formulas of the technical implementation of the RAMM in general terms. The particular cases and further details will be discussed in section 3.

### Trading formulas

The trading formulas of the RAMM are similar to Balancer's ones. However, as we mentioned above, before each trade the weights of the tokens need to be adjusted so that the internal spot prices coincide with the external market prices given by an oracle.

In order to do so, we proceed in the following way. Let $n$ be the number of tokens of the RAMM pool and, for each $j \in \{1, 2, \dots, n\}$, let $p_j$ be the market price of token $j$ in terms of a chosen currency (for example USD) or stablecoin at a certain moment. The weights are updated as

$$w_j = \frac{p_j B_j}{\sum_{i=1}^{n} p_i B_i} \tag{2}$$

for all $j \in \{1, 2, \dots, n\}$, where $B_1, B_2, \dots, B_n$ are the balances of each of the tokens in the pool. It is not difficult to verify that, with this definition of the weights, the spot prices coincide with the market prices (see Proposition 3).

Before each trade, we also need to update the trading fee $\phi$ and the leverage parameter $k$. We will describe how this is done in the following subsections.

With this data, we apply the following trading formulas. If the pool balances before a trade are $B_1, B_2, \dots, B_n$ and the updated weights are $w_1, w_2, \dots, w_n$, the amount $A_o$ of token $o$ that the trader receives when depositing an amount $A_i$ of token $i$ is

$$A_o = kB_o \left(1 - \left(\frac{kB_i}{kB_i + (1-\phi)A_i}\right)^{\frac{w_i}{w_o}}\right). \tag{3}$$

and the amount $A_i$ of token $i$ that the trader has to deposit in order to receive an amount $A_o$ of token $o$ is

$$A_i = \frac{kB_i}{1-\phi}\left(\left(\frac{kB_o}{kB_o - A_o}\right)^{\frac{w_o}{w_i}} - 1\right). \tag{4}$$

In addition, if a trader deposits an amount $A_i$ of token $i$ and receives an amount $A_o$ of token $o$, and $B'_1, B'_2, \ldots, B'_n$ are the pool balances after the trade then

$$\begin{aligned} B'_i &= B_i + A_i - \beta\phi A_i \ , \\ B'_o &= B_o - A_o \ , \qquad\qquad \text{and} \\ B'_j &= B_j \qquad\qquad\qquad\quad \text{for all } j \in \{1, 2, \ldots, n\} - \{i, o\}, \end{aligned}$$

where $\beta \in [0, 1)$ is the protocol fee. Thus, with the previous notations, $\beta\phi A_i$ is the amount that is set aside for the protocol.

Observe that we need to check that $B'_o \geq 0$ in order to perform the trade.

### Imbalance ratios

In order to mantain the pool balanced and reduce the impermanent loss for liquidity providers, we need to be able to measure how unbalanced the pool is at a given time. This is done by introducing the *imbalance ratios*, which will be defined below.

When a pool of $n$ assets is created, for each $j \in \{1, 2, \ldots, n\}$ we choose a positive real number $\lambda_j$ that will represent a scaling factor between the amount of token $j$ deposited and the amount of LP tokens of type $j$ received at the beginning. For example, if $\lambda_1 = 10$, a user that deposits 10 tokens of a certain type will receive one LP token of that type. This scaling factors might be ignored by simply setting $\lambda_j = 1$ for all $j \in \{1, 2, \ldots, n\}$, but they can help when dealing with assets that have very different market prices.[3]

Now, let $L_1, L_2, \ldots, L_n$ be the number of existent LP tokens of types $1, 2, \ldots, n$ respectively, and let $B_1, B_2, \ldots, B_n$ be the current pool balances. Let $p_1, p_2, \ldots, p_n$ be the current market prices of tokens $1, 2, \ldots, n$ in terms of the chosen currency or stablecoin. Let

$$L = \sum_{j=1}^{n} p_j \lambda_j L_j$$

and let

$$B = \sum_{j=1}^{n} p_j B_j \ .$$

For each $j \in \{1, 2, \ldots, n\}$, if $L_j \neq 0$ and $B \neq 0$, the *imbalance ratio* of token $j$ is defined as

$$\iota_j = \frac{\frac{B_j}{B}}{\frac{\lambda_j L_j}{L}} = \frac{B_j L}{\lambda_j L_j B} \ . \tag{5}$$

Observe that $\iota_j > 1$ (resp. $\iota_j < 1$) means that there is currently a bigger (resp. smaller) proportion of token $j$ than that represented by the amount of existent LP tokens of type $j$.

If either $L_j = 0$ or $B = 0$, then the imbalance ratio of token $j$ is not defined. Observe that $B = 0$ if and only if $B_k = 0$ for all $k \in \{1, 2, \ldots, n\}$, that is, the pool is empty. On the other hand, $L_j = 0$ might occur if there are no active liquidity providers of token $j$.

Observe also that $L = 0$ if and only if $L_k = 0$ for all $k \in \{1, 2, \ldots, n\}$, that is, the pool has no active liquidity providers of any token. This should only happen if the pool is empty.

Therefore, for any $j \in \{1, 2, \ldots, n\}$, assuming that the pool is not empty (and hence $B \neq 0$ and $L \neq 0$) and that $L_j \neq 0$, we obtain that $\iota_j = 0$ if and only if $B_j = 0$.

[3] In our first implementation of the RAMM we simply chose $\lambda_j = 1$ for all $j$.

### Setting a range for the imbalance ratios

We will choose a parameter $\delta$ such that $0 < \delta < 1$, which will define a range $[1-\delta, 1+\delta]$ within which the imbalance ratios should move. This will help ensure that the pool does not go too much out of balance. For example, we might choose $\delta = 0.25$, which defines the range $[0.75, 1.25]$, and implement in the smart contract's code the condition that if a trade makes an imbalance ratio decrease below 0.75 or increase above 1.25, then the trade will not be allowed.

It is important to point out that, due to price movements, an imbalance ratio can go out of range even if there are no transactions in the pool. In this case, the trades that shouldn't be performed are those that further reduce an imbalance ratio which is below $1-\delta$ and those that further increase an imbalance ratio which is above $1+\delta$. The objective of this is that, in the case that an imbalance ratio goes out of range for any reason (for example, because of huge price movements), trades that make the imbalance ratio approach the chosen range should be allowed. Although the imbalance ratios should rarely go out of the chosen range, the previous consideration serves as an additional safety measure.

In addition, we mention that simulations show that the imbalance ratios usually move between 0.9 and 1.1, thus choosing $\delta = 0.25$ (which defines the range $[0.75, 1.25]$) should not cause any problems or restrictions.

### Updating the fees and the leverage parameter

As we mentioned before, the rebalancing mechanism of our pool consists of adjusting the fees and the leverage parameter so that we favor trades that help rebalance the pool and discourage trades that will further unbalance the pool.

In order to do so, we choose two increasing functions $f_{\text{fee}}, f_{\text{lev}} \colon \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ such that $f_{\text{fee}}(1) = 1 = f_{\text{lev}}(1)$. For the sake of symmetry, these functions should satisfy $f_{\text{fee}}(\frac{1}{x}) = \frac{1}{f_{\text{fee}}(x)}$ and $f_{\text{lev}}(\frac{1}{x}) = \frac{1}{f_{\text{lev}}(x)}$ for all $x \in \mathbb{R}_{>0}$, though this is not a requirement. We suggest defining $f_{\text{fee}}(x) = f_{\text{lev}}(x) = x^3$ for all $x \in \mathbb{R}_{>0}$.

Now let $\phi_0$ be the base fee of the pool (for example, $\phi_0 = 0.001$) and let $k_0$ be the base leverage parameter of the pool (for example, $k_0 = 100$). For trades that deposit token $i$ into the pool and obtain token $o$ from the pool, the trading fee $\phi$ and the leverage parameter $k$ are computed as

$$\phi = \frac{f_{\text{fee}}(\iota_i)}{f_{\text{fee}}(\iota_o)}\phi_0 \tag{6}$$

and

$$k = \frac{f_{\text{lev}}(\iota_o)}{f_{\text{lev}}(\iota_i)}k_0 \ , \tag{7}$$

where $\iota_i$ and $\iota_o$ are the imbalance ratios of tokens $i$ and $o$ respectively. Note that we need that $\iota_i \neq 0$ and $\iota_o \neq 0$ in order to define $\phi$ and $k$.

From formulas (6) and (7) we see that if, for example, the proportion of token $i$ is bigger than it should be and the proportion of token $o$ is smaller than it should be, then $\iota_o < 1 < \iota_i$ and hence $f(\iota_o) < 1 < f(\iota_i)$ for $f \in \{f_{\text{fee}}, f_{\text{lev}}\}$. Thus, $\phi > \phi_0$ and $k < k_0$. This means that the trading fee is increased and the leverage parameter is reduced, so that the trade is more expensive than in the balanced situation. Note that this makes sense since the previous trade would add more token $i$ to the pool and remove some token $o$ from the pool, and hence it will further unbalance the pool.

### Deposits and withdrawals

As mentioned before, when depositing an amount of a certain token $j$, the liquidity provider will receive an amount of LP tokens of type $j$. Concretely, if the liquidity provider deposits an

amount $a_j$ of token $j$, he will receive an amount $T$ of LP tokens of type $j$ given by

$$T = \frac{a_j}{B_j} \iota_j L_j \ , \tag{8}$$

where $B_j$ is the current balance of token $j$ in the pool, $\iota_j$ is the imbalance ratio of token $j$ and $L_j$ is the amount of LP tokens of type $j$ in circulation. Note that, in order to apply the previous formula, we need that $B_j \neq 0$ and that $\iota_j$ is defined. We will deal with the cases in which this does not hold in the next section.

On the other hand, if a liquidity provider redeems an amount $T$ of LP tokens of type $j$ he will receive an amount $a$ of token $j$ given by

$$a = \frac{T B_j}{L_j \iota_j} \tag{9}$$

where, as above, $B_j$ is the current balance of token $j$ in the pool, $\iota_j$ is the imbalance ratio of token $j$ and $L_j$ is the amount of LP tokens of type $j$ in circulation. Note that we need that $L_j \neq 0$ and that $\iota_j$ is defined and $\iota_j \neq 0$ to apply the previous formula. As with the liquidity deposit formula, we will deal with the cases in which this does not hold in the next section.

As we can observe, the previous formulas take into account the possible imbalance of token $j$, so that temporary imbalances do not affect the deposit and withdrawal amounts.

It has to be mentioned, though, that if the pool is imbalanced and many liquidity providers want to redeem their LP tokens at the same time, it might occur that there are not enough tokens of a certain type in the pool reserves to give to the liquidity provider when he redeems his LP tokens. In that case, the liquidity provider that can not be given the whole value of their LP tokens in terms of the corresponding asset, will receive the remaining value of their LP tokens in terms of one or more different assets of the pool. Nevertheless, as we will see later (page 11), the amount of the corresponding token that the liquidity provider will receive is, at least, the amount of token $j$ in the pool that is proportional to the LP tokens of type $j$ that he is redeeming (with respect to all LP tokens of type $j$ in circulation).

### General comments about this design

We include below some comments about the design of the RAMM.

- The earnings of the pool are distributed evenly among all liquidity providers. Thus, all types of LP tokens of the same RAMM pool will have identical APR.
- Simulations show that this design is effective to keep the pool balanced. It has to be mentioned, though, that liquidity providers can face losses in some cases, but we expect them to be rare.
- We must pay special attention when dealing with the oracle and its prices, because if an attacker is able to manipulate the oracle feed in some way, he would be able to steal funds from the pool. Safety measures might include taking prices from several sources, avoiding using low cap tokens in our pools, and implementing time-weighted average prices (TWAPs) across a suitable chosen time interval.
- The RAMM design works better with tokens with high trading volume. In addition, the RAMM might not be suitable for low cap tokens.
- In order to get the most from the RAMM design, the RAMM pools should be integrated with aggregators, so that the pool can get more trading volume. This will help mantain the pool balanced and, of course, will enable the pool to collect more fees.

- From the RAMM design it is not difficult to infer that, when trading against a RAMM pool, splitting the trade into several parts is better (for the trader) than doing it in one step, because by dividing the trade into several steps the trader will be able to reduce the price impact and obtain a price that is closer to the market price. In order to avoid manipulations due to this fact, our implementation includes minimum trading amounts for each of the assets of the pool, so that trades that are too small will not be allowed.

# 3 Detailed technical implementation

In this section we will provide a thorough explanation of the technical implementation of the RAMM. We will show how to deal with the cases in which the formulas given in the previous section can not be applied and we will explain each of the steps needed to perform any operation (trade, liquidity deposit, liquidity withdrawal) with the RAMM, analyzing every possible case.

We will employ the same notation as that of the previous section. Thus, $n$ will denote the number of tokens of the corresponding RAMM pool, $B_1, B_2, \ldots, B_n$ will be the pool balances, and $L_1, L_2, \ldots, L_n$ will be the number of existent LP tokens of types $1, 2, \ldots, n$ respectively. In addition, $p_1, p_2, \ldots, p_n$ will be the current market prices of tokens $1, 2, \ldots, n$ in terms of a chosen currency or stablecoin.

As we pointed out in the previous section, if, for some $j \in \{1, 2, \ldots, n\}$, the number $L_j$ of LP tokens of type $j$ in circulation is zero, then the imbalance ratio $\iota_j$ can not be defined (see formula (5)). Similarly, formula (8) can not be applied if $B_j = 0$ and formula (9) can not be applied if $L_j = 0$.

We will show below how we should proceed so as to deal with these situations.

**One special case.** We also want to ensure that, for all $j \in \{1, 2, \ldots, n\}$, if $L_j \neq 0$ then $B_j \neq 0$. This amounts to saying that the particular case $L_j \neq 0$ and $B_j = 0$ does not occur. This case will be called **Case X**.

The mechanisms that will be discussed below will guarantee that Case X does never occur. Nevertheless, we will also show how to proceed in Case X so as to avoid issues with the smart contract.

### Trading

When trading and depositing or withdrawing liquidity, the protocol must take into account the considerations and different cases described below.

Suppose a trader wants to deposit token $i$ into the pool so as to obtain token $o$.

- If $B_o = 0$, the trade can not be performed since there is no token $o$ left in the pool.
- If $L_i = 0$, the trade is not permitted. In this way we do not allow adding token $i$ to the pool since there should not be any if there are no LP tokens of type $i$ in circulation.
- If $L_i \neq 0$ and $B_o \neq 0$ and $B_i = 0$, the trade is allowed. In this case $w_i = 0$ and formulas (3) and (4) can not be applied. But since this trade will bring some token $i$ to the pool, which is needed, the trade will be performed. The trade will be executed at the oracle price, and fees will be charged on the token that goes into the pool, as usual. Therefore, the amount $a_i$ of token $i$ that goes into the pool and the amount $a_o$ that goes out of the pool are related by the following formulas

$$a_o = (1 - \phi) a_i \frac{p_i}{p_o} \qquad \text{and} \qquad a_i = \frac{a_o}{1 - \phi} \frac{p_o}{p_i} \ .$$

  Note that this is Case X described above, and hence it should not occur, but it is better to consider it anyway as we mentioned before.

- If $L_i \neq 0$ and $B_o \neq 0$ and $B_i \neq 0$ and $L_o = 0$, the trade is allowed, but the imbalance ratio $\iota_o$ can not be defined (see formula (5)). Thus, instead of using formulas (6) and (7) to compute the fee $\phi$ and the leverage parameter $k$, we use $\phi = \phi_0$ and $k = k_0$. The trade is then performed using formula (3) or formula (4).
- If $L_i \neq 0$ and $B_o \neq 0$ and $B_i \neq 0$ and $L_o \neq 0$, the trade is performed using formulas (6), (7), (3), and (4). To this end, the protocol must follow the steps below.
  1. Retrieve the market prices from the oracle.
  2. Use the market prices to update the weights.
  3. Use the market prices to update the imbalance ratios.
  4. Use the imbalance ratios to update the fee and the leverage parameter.
  5. Use the updated parameters in the trading formula to perform the trade.

In any of the cases of the last three items above (that is, when $L_i \neq 0$ and $B_o \neq 0$), after the trade the protocol will check if the imbalance ratios of the tokens involved in the trade belong to the range $[1-\delta, 1+\delta]$ defined previously. Concretely, if for each $j \in \{1, 2, \ldots, n\}$, $\iota_j$ and $\iota'_j$ denote the values of the imbalance ratio of token $j$ just before and after the trade, respectively, then the protocol should check that

$$(1-\delta \leq \iota'_o \leq 1+\delta \quad \text{or} \quad 1+\delta < \iota'_o < \iota_o) \quad \text{and} \quad (1-\delta \leq \iota'_i \leq 1+\delta \quad \text{or} \quad \iota_i < \iota'_i < 1-\delta)$$

If the previous line does not hold, the trade will not be performed.

Observe that it might be possible that the imbalance ratio of another token different from token $i$ and token $o$ goes out of range after the trade of token $i$ for token $o$ (see Proposition 11). However, since this might happen anyway due to changes in the price of the different tokens, there is no point in checking that the imbalance ratio of every token stays in the chosen range after a trade. Nevertheless, checking that the imbalance ratios of the tokens involved in the trades belong to the interval after the trade is useful to keep the pool balanced and to prevent possible manipulations.

### Liquidity deposit

Suppose that a user wants to provide liquidity of token $j$, depositing an amount $a_j$ of that token.

- If $B = 0$, the amount of LP tokens of type $j$ that the user receives will be $\frac{a_j}{\lambda_j}$. Recall that $B = 0$ occurs when the pool is empty, for example, when it has just been created.
- If $B \neq 0$ and $L_j = 0$, the amount of LP tokens of type $j$ that the user receives is $\frac{a_j L}{\lambda_j B}$, where $L$ and $B$ are defined on page 5. In this way, if the user deposits liquidity and immediately withdraws it, he will obtain the same amount that he has deposited (see Proposition 8). In addition, if $B_j = 0$, the imbalance ratio of token $j$ after the deposit will be equal to 1, as expected (Lemma 7).
- If $B \neq 0$ and $L_j \neq 0$ and $B_j \neq 0$, formula (8) is applied to find out how many LP tokens of type $j$ the user obtains. Concretely, the amount $T$ of LP tokens of type $j$ that the liquidity provider obtains is given by

  $$T = \frac{a_j}{B_j} \iota_j L_j$$

  Note that

  $$T = \frac{a_j}{B_j} L_j \iota_j = \frac{a_j}{B_j} L_j \frac{B_j L}{\lambda_j L_j B} = \frac{a_j L}{\lambda_j B} \ ,$$

  which is the formula of the previous item.

- If $B \neq 0$ and $L_j \neq 0$ and $B_j = 0$, we are once again in Case X. Hence, this case should not happen, as explained before. However, in the case that it occurs, the amount of LP tokens of type $j$ that the user receives is $\frac{a_j L}{\lambda_j B}$, where $L$ and $B$ are defined on page 5.

As it can be seen, if $B \neq 0$, the amount of LP tokens of type $j$ that the user receives is $\frac{a_j L}{\lambda_j B}$ in any case. Therefore, this formula can be used instead of formula (8). However, formula (8) is still interesting because it uses parameters related to token $j$ only and the reasoning behind it is clear.

It is interesting to mention that if a liquidity provider deposits token $j$ then, for $k \neq j$, the imbalance ratio of token $k$ does not change, and the new imbalance ratio of token $j$ is closer to 1 than it was before the liquidity deposit (see Proposition 4).

### Liquidity withdrawal

Suppose that a liquidity provider wants to redeem an amount $T$ of LP tokens of type $j$. Note that $0 < T \leq L_j$. In particular, $L \neq 0$.

**Case 1:** $B_j \neq 0$.

Suppose that $B_j \neq 0$. Then $B \neq 0$ and $\iota_j \neq 0$. From formula (9), we know that the amount of token $j$ that the liquidity provider has to receive is

$$a = \frac{TB_j}{L_j \iota_j} \ .$$

**Case 1.1:** $T < L_j$.

If $T < L_j$, in order to prevent the imbalance ratio $\iota_j$ from going outside the range $[1-\delta, 1+\delta]$, we will compute the maximum amount of token $j$ that can be given to the liquidity provider so that $\iota_j$ stays within the range $[1-\delta, 1+\delta]$ (if it is already in that range).

In order to obtain this maximum amount, first observe that the amount $T$ of LP tokens of type $j$ is worth $a$ tokens $j$, or equivalently, $p_j a$ stablecoin (or currency) units. Thus, either in terms of token $j$ or in terms of a combination of several tokens, the liquidity provider must obtain that value of stablecoin (or currency) units. This means that after the withdrawal, the value of the pool (in terms of the chosen stablecoin) will be

$$B - p_j a = B - p_j \frac{TB_j}{L_j \iota_j} = B - p_j \frac{TB_j}{L_j} \frac{\lambda_j L_j B}{B_j L} = B - p_j \lambda_j T \frac{B}{L} \ ,$$

where $B$ and $L$ are defined on page 5 (and computed before the withdrawal). Now, since we need that the updated imbalance ratio $\iota'_j$ is greater than or equal to $1-\delta$, using the values after the possible withdrawal and applying formula (5), we obtain that

$$1 - \delta \leq \iota'_j = \frac{(B_j - a)(L - p_j \lambda_j T)}{\lambda_j (L_j - T)(B - p_j \lambda_j T \frac{B}{L})} = \frac{(B_j - a)(L - p_j \lambda_j T)}{\lambda_j (L_j - T) \frac{B}{L} (L - p_j \lambda_j T)} = \frac{B_j - a}{\lambda_j (L_j - T) \frac{B}{L}} \ .$$

Thus,

$$a \leq B_j - (1-\delta) \lambda_j (L_j - T) \frac{B}{L} \ .$$

If $\iota_j \geq 1 - \delta$, let

$$M_j = B_j - (1-\delta) \lambda_j (L_j - T) \frac{B}{L}.$$

Thus, if $\iota_j \geq 1-\delta$, $M_j$ is the maximum amount of token $j$ that the liquidity provider can receive so that the imbalance ratio of token $j$ after the withdrawal is still greater than or equal to $1-\delta$.

Otherwise, if $\iota_j < 1 - \delta$, let $M_j = \frac{T}{L_j} B_j$. Note that $\frac{T}{L_j} B_j$ is the amount of token $j$ in the pool that is proportional to the liquidity provider's LP tokens of type $j$ with respect to all LP tokens of type $j$ in circulation.

It is interesting to observe that, if $\iota_j \geq 1-\delta$,

$$\begin{aligned} M_j &= B_j - (1-\delta)\lambda_j(L_j - T)\frac{B}{L} = B_j - (1-\delta)\lambda_j L_j \frac{B}{L} + (1-\delta)\lambda_j T\frac{B}{L} = \\ &= B_j - (1-\delta)\frac{B_j}{\iota_j} + (1-\delta)T\frac{B_j}{\iota_j L_j} = B_j - \frac{(1-\delta)}{\iota_j}B_j + \frac{(1-\delta)}{\iota_j}\frac{TB_j}{L_j} - \frac{TB_j}{L_j} + \frac{TB_j}{L_j} = \\ &= B_j\left(1 - \frac{1-\delta}{\iota_j}\right) - \frac{TB_j}{L_j}\left(1 - \frac{1-\delta}{\iota_j}\right) + \frac{TB_j}{L_j} = \left(B_j - \frac{TB_j}{L_j}\right)\left(1 - \frac{1-\delta}{\iota_j}\right) + \frac{TB_j}{L_j} \geq \\ &\geq \frac{TB_j}{L_j} \end{aligned}$$

since $\frac{T}{L_j} < 1$ and $\iota_j \geq 1-\delta$. This means that the maximum amount that the liquidity provider can receive so as to satisfy our needs is at least the amount of token $j$ in the pool that is proportional to his LP tokens of type $j$ (with respect to all LP tokens of type $j$ in circulation).

**Case 1.1.1:** $a \leq M_j$.

If the amount $a$ that has to be given to the liquidity provider satisfies the previous inequality, then the liquidity provider receives the amount $a$ of token $j$, his LP tokens of type $j$ are burned, and the procedure ends.

**Case 1.1.2:** $a > M_j$.

If $a > M_j$, then the liquidity provider will receive an amount $M_j$ of token $j$ and the remaining amount $a - M_j$ will be paid in a different token. To this end, let $k \neq j$ such that $\iota_k \geq \iota_l$ for all $l \neq j$ (this is, token $k$ has the maximum imbalance ratio of all the tokens different from token $j$). If possible, the liquidity provider will be given the remaining amount $a - M_j$ of token $j$ in terms of token $k$. Thus, he has to receive an amount $\frac{(a-M_j)p_j}{p_k}$ of token $k$.

With a similar argument as before, if $\iota'_k$ is the updated imbalance ratio of token $k$ after the withdrawal, we want that $\iota'_k \geq 1-\delta$. Hence, if $a_k$ is the amount of token $k$ that is given to the liquidity provider, we obtain that

$$1-\delta \leq \iota'_k = \frac{(B_k - a_k)(L - p_j\lambda_j T)}{\lambda_k L_k(B - p_j\lambda_j T\frac{B}{L})} = \frac{(B_k - a_k)(L - p_j\lambda_j T)}{\lambda_k L_k \frac{B}{L}(L - p_j\lambda_j T)} = \frac{B_k - a_k}{\lambda_k L_k \frac{B}{L}} \ .$$

where $B$ and $L$ are the same as before, this is, these values are computed when the withdrawal procedure starts (and are not updated at this point).

Thus, we need that

$$a_k \leq B_k - (1-\delta)\lambda_k L_k \frac{B}{L} \ .$$

Hence, we define

$$M_k = \max\left\{B_k - (1-\delta)\lambda_k L_k \frac{B}{L} \ , \ 0\right\}.$$

Note that

$$B_k - (1-\delta)\lambda_k L_k \frac{B}{L} = B_k - (1-\delta)\frac{B_k}{\iota_k} = B_k\left(1 - \frac{1-\delta}{\iota_k}\right).$$

Hence, $B_k - (1-\delta)\lambda_k L_k \frac{B}{L} \geq 0$ if and only if $\iota_k \geq 1-\delta$.

It follows that if $\iota_k \geq 1-\delta$ and the liquidity provider receives an amount of token $k$ that is not greater than $M_k$, then the imbalance ratio $\iota'_k$ (after the withdrawal) will be greater than or equal to $1-\delta$.

Therefore, if $\frac{(a-M_j)p_j}{p_k} \leq M_k$, the liquidity provider is given an amount $\frac{(a-M_j)p_j}{p_k}$ of token $k$ and the procedure ends.

If instead, $\frac{(a-M_j)p_j}{p_k} > M_k$, the liquidity provider receives an amount $M_k$ of token $k$ and the remaining amount $R = \frac{(a-M_j)p_j}{p_k} - M_k$ (of token $k$) is delivered in terms of another token.

In order to do so, we proceed as above. We choose $l$ such that token $l$ has the maximum imbalance ratio of all the tokens different from tokens $j$ and $k$. We define $M_l$ in a similar way as $M_k$ and try to give an amount $\frac{Rp_k}{p_l}$ of token $l$ to the liquidity provider. If $\frac{Rp_k}{p_l} \leq M_l$ this is done and the procedure ends. If not, we repeat this argument.

**Case 1.2:** $T = L_j$.

If $T = L_j$, then $a = \frac{B_j}{\iota_j}$.

**Case 1.2.1:** $a \leq B_j$.

If $a \leq B_j$, the liquidity provider receives the amount $a$ of token $j$. The remaining amount of token $j$ in the pool will be $B_j - a$ and there will be no LP tokens of type $j$ left.

**Case 1.2.2:** $a > B_j$.

If $a > B_j$, the liquidity provider receives an amount $B_j$ of token $j$, and the remaining amount $a - B_j$ that he has to receive will be paid in a different token. In this case, we proceed as in Case 1.1.2.

**Case 2:** $B_j = 0$.

Suppose now that $B_j = 0$. As we explained before, since $L_j \neq 0$ this case should not occur (Case X). However, we will explain how to deal with it anyway.

In this case, the amount of token $j$ that the user must receive is $\frac{T\lambda_j B}{L}$, where $L$ and $B$ are defined on page 5. But since there is no token $j$ left, the user will be given another token instead. Again, we proceed as in Case 1.1.2.

## Examples

The following examples will be useful to understand how trades are performed, how the fees and the leverage parameter are dynamically adjusted, how liquidity provider's positions accrue fees and how the value of those positions change if the market prices change.

**Example 1** (Dynamic fees and leverage parameter)**.**

Consider an ETH/MATIC/USDT RAMM pool with 0.1% fee and base leverage parameter $k_0 = 100$. ETH will be token 1, MATIC will be token 2 and USDT will be token 3. We define the scale factors as $\lambda_1 = \lambda_2 = \lambda_3 = 1$. We consider a protocol fee of 50%. Suppose that the price of ETH in terms of USDT is 1800 and that the price of MATIC in terms of USDT is 1.2.

**Liquidity deposits.** Suppose that a first liquidity provider deposits 200 ETH into the pool. For doing so, he will receive 200 LPETH tokens.

Now suppose that a second liquidity provider deposits 200,000 MATIC into the pool. From the liquidity withdrawal formulas it follows that he will receive 200,000 LPMATIC tokens.

In a similar way suppose that a third liquidity provider deposits 400,000 USDT into the pool. From the liquidity withdrawal formulas it follows that he will receive 400,000 LPUSDT tokens.

After these three liqudity deposits, the balances of the pool are 200 ETH, 200,000 MATIC and 400,000 USDT. It can be easily verified that all the imbalance ratios are equal to 1.

**Trade 1.** Suppose now that a trader sells 10 ETH to the pool so as to obtain USDT. Since the imbalance ratios are all equal to one, the leverage parameter for the trade will be equal to the base leverage parameter $k_0$ (which is equal to 100) and the fee for the trade will be equal to the base fee $\phi_0$ of the pool, which is 0.001. From the corresponding trading formula, it follows that the trader obtains approximately 17973.47 USDT. Disregarding the fee, the effective price paid to the trader is approximately 1799.15 USDT/ETH. Thus, the trade had a price impact of around 0.047% (disregarding the trading fee), which is very small considering that the volume of the trade is 1.8% of the value of the whole pool (trading volume: \$18,000, TVL: \$1,000,000).

After the trade, the balances of the pool are approximately

$$B_1 = 209.995, \qquad B_2 = 200000, \qquad B_3 = 382026.53\ ,$$

and the imbalance ratios are approximately

$$\iota_1 = 1.049957, \qquad \iota_2 = 0.999982, \qquad \iota_3 = 0.955049 \ .$$

In addition, the RAMM has collected 0.005 ETH as protocol fees.

**Trade 2.** Suppose now that another trader wants to sell 5 ETH so as to obtain MATIC. For this trade the leverage parameter will be approximately 86.39 and the trading fee will be approximately 0.116%. Note that both the leverage parameter and the trading fee are less favorable for this second trader since the trade he wants to make will further unbalance the pool. This trader will obtain approximately 7488.66 MATIC.

After the second trade, the balances of the pool are approximately

$$B_1 = 214.99, \qquad B_2 = 192511.34, \qquad B_3 = 382026.53 \ ,$$

and the imbalance ratios are approximately

$$\iota_1 = 1.074933, \qquad \iota_2 = 0.962532, \qquad \iota_3 = 0.955042 \ .$$

In addition, the RAMM has already collected around 0.007894 ETH as protocol fees.

**Withdrawal.** Now, suppose that the first liquidity provider wants to redeem the 200 LPETH tokens he has received. By doing so, he will receive 200.00518458 ETH, which amounts to his initial deposit plus 0.00518458 ETH as accrued value from the two trades. Note that this accrued value comes not only from the trading fees but also from the price impact of the trades.

**Example 2** (Value of the liquidity provider's positions)**.**

Consider an ETH/USDT RAMM pool with 0.1% fee and base leverage parameter $k_0 = 100$. ETH will be token 1 and USDT will be token 2. We define the scale factors as $\lambda_1 = \lambda_2 = 1$. We consider a protocol fee of 50%. Suppose that the price of ETH in terms of USDT is 2000.

**Liquidity deposits.** Suppose that a first liquidity provider deposits 300 ETH into the pool. For doing so, he will receive 300 LPETH tokens.

Now suppose that a second liquidity provider deposits 400,000 USDT into the pool. From the liquidity withdrawal formulas it follows that he will receive 400,000 LPUSDT tokens.

**Trade 1.** Suppose now that a trader buys 20 ETH from the pool. The details of the trade are the following.

In: 40073.43271143308 USDT — Out: 20 ETH
Protocol fee: 20.03671635571654 USDT
Pool balances after the trade: 280 ETH, 440053.3959950774 USDT.
Imbalance ratios after the trade: $\iota_1 = 0.9332834997321758$, $\iota_2 = 1.100074750401736$.

**Value of each position if the price of ETH is 2000 USDT.** Observe that if the pool had been balanced, it would have had

$$\frac{280}{0.9332834997321758} = 300.01601879852322495994 \text{ ETH}$$

and

$$\frac{440053.3959950774}{1.100074750401736} = 400021.35839803106067347568 \text{ USDT} \ .$$

This means that if liquidity provider A wants to withdraw his position he should be given 300.01601879852322495994 ETH. Indeed, this is what follows from the liquidity withdrawal formulas. But since this is not possible because the pool does not have that amount of ETH, he will be given instead 280 ETH and 40032.03759704638 USDT, which amounts to the same value as 300.01601879852322495994 ETH.

On the other hand, if liquidity provider B wants to withdraw his position he should obtain 400021.35839803106067347568 USDT. Observe that the sum of the amounts of USDT that are given to the liquidity providers is equal to the balance of USDT in the pool:

$$40032.03759704638 + 400021.35839803106067347568 = 440053.39599507744067347568 \ .$$

Observe also that the ratio between the value in USDT of both positions is

$$\frac{280 \cdot 2000 + 40032.03759704638}{400021.35839803106067347568} = 1.5 \ ,$$

which is the same ratio as at the beginning, since the deposit of liquidity provider A was worth 600,000 USDT and the deposit of liquidity provider B was worth 400,000 USDT.

**Value of each position if the price of ETH is 4000 USDT.** Suppose now that the price of ETH increases to 4000 USDT. In this case the pool state is given by

Balances: 280 ETH, 440053.3959950774 USDT.
Imbalance ratios: $\iota_1 = 0.957232192928123$, $\iota_2 = 1.128303421215631$.

Applying the withdrawal formulas we obtain that the position owned by liquidity provider A is worth approximately 292.510011749077 ETH and that the position owned by liquidity provider B is worth approximately 390013.34899876936 USDT. Thus, if liquidity provider A wants to withdraw his position he should be given 292.510011749077 ETH, but since this is not possible, he will be given instead 280 ETH and 50040.04699630809 USDT, which amounts to the same value as 292.510011749077 ETH.

Note that the sum of the amounts of USDT that are given to the liquidity providers is equal to the balance of USDT in the pool:

$$50040.04699630809 + 390013.34899876936 \approx 440053.3959950774 \ .$$

We will explain now why both liquidity providers' positions have lost value compared to the previous case. After the liquidity deposits, the value of the pool was

$$300 \cdot 2000 + 400{,}000 = 1{,}000{,}000 \text{ USDT} \ ,$$

and increased a bit after the first trade, reaching

$$280 \cdot 2000 + 440053.3959950774 = 1000053.3959950774 \text{ USDT} \ .$$

Now, with a price of ETH of 4000 USDT, the value of the pool is

$$280 \cdot 4000 + 440053.3959950774 = 1560053.3959950774 \text{ USDT} \ ,$$

but this is lower than the value the liquidity providers' original positions would have, since

$$300 \cdot 4000 + 400{,}000 = 1{,}600{,}000 \text{ USDT} \ .$$

This is, the whole pool is experiencing an impermanent loss.

Observe that the value of the original position of liquidity provider A would be now 1,200,000 USDT, which is three times the value that the original position of liquidity provider B would have now. Note that the ratio between the current value of their positions is

$$\frac{280 \cdot 4000 + 50040.04699630809}{390013.34899876936} = 3 \ ,$$

which means that both liquidity providers obtain the corresponding share of their positions. This serves to validate the liquidity withdrawal formula.

**Value of each position if the price of ETH is 1000 USDT.** Suppose now that the price of ETH decreases to 1000 USDT. In this case the pool state is given by

Balances: 280 ETH, 440053.3959950774 USDT.
Imbalance ratios: $\iota_1 = 0.9071619897061661$, $\iota_2 = 1.0696285077203755$.

Applying the withdrawal formulas we obtain that the position owned by liquidity provider A is worth approximately 308.5943125693189 ETH and that the position owned by liquidity

provider B is worth approximately 411459.0834257585 USDT. Thus, if liquidity provider A wants to withdraw his position he should be given 308.5943125693189 ETH, but since this is not possible, he will be given instead 280 ETH and 28594.312569318903 USDT, which amounts to the same value as 308.5943125693189 ETH.

Note that the sum of the amounts of USDT that are given to the liquidity providers is equal to the balance of USDT in the pool:

$$28594.312569318903 + 411459.0834257585 \approx 440053.3959950774 \ .$$

We will explain now why both liquidity providers' positions have gained value compared to the case in which the price of ETH was 2000 USDT. Observe that the current value of the pool (with a price of ETH of 1000 USDT) is

$$280 \cdot 1000 + 440053.3959950774 = 720053.3959950774 \text{ USDT} \ ,$$

which is higher than the value of the liquidity providers' original positions, since

$$300 \cdot 1000 + 400{,}000 = 700{,}000 \text{ USDT} \ .$$

This is, the whole pool is experiencing an "impermanent gain".

Observe that the value of the original position owned by liquidity provider A would be now 300,000 USDT, which is just 75% of the value that the original position owned by liquidity provider B would have now. Note that the ratio between the current value of their positions is

$$\frac{280 \cdot 1000 + 28594.312569318903}{411459.0834257585} = 0.75 \ ,$$

as expected. This means that both liquidity providers are obtaining the corresponding share of their positions.

# 4 Mathematical propositions and their proofs

In this section we will state and prove some propositions that were referenced before and that are needed to demostrate that the mathematical foundation of the RAMM is sound and that everything works properly.

As we will employ the same notation as the one used in the whole article, for simplicity we will sometimes avoid stating what the different symbols stand for.

**Proposition 3.** *Consider a RAMM pool with $n$ tokens and no fees. We define the weights of each of the tokens of the pool using formula (2). Let $i, j \in \{1, 2, \ldots, n\}$. Then the spot price of token $i$ in terms of token $j$ given by the RAMM coincides with the market price of token $i$ in terms of token $j$.*

*Proof.* Let $p$ be the market price of token $i$ in terms of token $j$. For each $l \in \{1, 2, \ldots, n\}$, let $p_l$ be the market price of token $l$ in terms of a chosen stablecoin.

Let $\rho$ be the spot price of token $i$ in terms of token $j$ given by the RAMM. From the Balancer AMM's formulas we know that

$$\rho = \frac{\frac{B_j}{w_j}}{\frac{B_i}{w_i}} \ .$$

Thus, applying formula (2), we obtain that

$$\rho = \frac{\frac{B_j}{w_j}}{\frac{B_i}{w_i}} = \frac{B_j w_i}{B_i w_j} = \frac{B_j}{B_i} \cdot \frac{p_i B_i}{\sum_{l=1}^{n} p_l B_l} \cdot \frac{\sum_{l=1}^{n} p_l B_l}{p_j B_j} = \frac{p_i}{p_j} = p \ ,$$

as we wanted to prove. □

**Proposition 4.** *Let $j \in \{1, 2, \ldots, n\}$. Consider a non-empty RAMM pool with $n$ tokens. Suppose that a liquidity provider deposits an amount $a$ of token $j$. Let $\iota_1, \iota_2 \ldots, \iota_n$ be the imbalance ratios of each of the tokens of the pool before the liquidity deposit and let $\iota'_1, \iota'_2 \ldots, \iota'_n$ be the imbalance ratios of each of the tokens after the deposit. Then $\iota'_k = \iota_k$ for all $k \in \{1, 2, \ldots, n\} - \{j\}$ and*

- *if $\iota_j = 1$, then $\iota'_j = 1$;*
- *if $\iota_j < 1$, then $\iota_j < \iota'_j < 1$;*
- *if $\iota_j > 1$, then $1 < \iota'_j < \iota_j$.*

*Proof.* Let $k \in \{1, 2, \ldots, n\} - \{j\}$. Let $T$ be the amount of LP tokens of type $j$ that the liquidity provider receives. Recall that $T = \frac{La}{\lambda_j B}$. Then

$$\iota'_k = \frac{B_k(L + \lambda_j p_j T)}{\lambda_k L_k (B + p_j a)} = \frac{B_k(L + \lambda_j p_j \frac{La}{\lambda_j B})}{\lambda_k L_k (B + p_j a)} = \frac{B_k(L + p_j a \frac{L}{B})}{\lambda_k L_k (B + p_j a)} = \frac{B_k \frac{L}{B}(B + p_j a)}{\lambda_k L_k (B + p_j a)} = \frac{B_k L}{\lambda_k L_k B} = \iota_k \ .$$

Now, in order to prove the second part of the proposition, we may assume that $L_j \neq 0$, because if $L_j = 0$ then $\iota_j$ is not defined.

If $B_j = 0$, then $\iota_j = 0$ and

$$0 < \iota'_j = \frac{a(L + \lambda_j p_j T)}{\lambda_j (L_j + T)(B + p_j a)} = \frac{a(L + \lambda_j p_j \frac{La}{\lambda_j B})}{\lambda_j (L_j + T)(B + p_j a)} = \frac{a(L + p_j a \frac{L}{B})}{\lambda_j (L_j + T)(B + p_j a)} =$$
$$= \frac{a \frac{L}{B}(B + p_j a)}{\lambda_j (L_j + T)(B + p_j a)} = \frac{a \frac{L}{B}}{\lambda_j (L_j + T)} = \frac{T}{L_j + T} < 1,$$

as desired.

If $B_j \neq 0$, we have that $T = \frac{a}{B_j} \iota_j L_j = \frac{La}{\lambda_j B}$. Then,

$$\iota'_j = \frac{(B_j + a)(L + \lambda_j p_j T)}{\lambda_j (L_j + T)(B + p_j a)} = \frac{(B_j + a)(L + \lambda_j p_j \frac{La}{\lambda_j B})}{\lambda_j (L_j + \frac{a}{B_j} \iota_j L_j)(B + p_j a)} = \frac{(B_j + a)(L + p_j a \frac{L}{B})}{\lambda_j (L_j + \frac{a}{B_j} \iota_j L_j)(B + p_j a)} =$$
$$= \frac{(B_j + a) \frac{L}{B}(B + p_j a)}{\lambda_j \frac{L_j}{B_j}(B_j + a\iota_j)(B + p_j a)} = \frac{B_j L (B_j + a)}{\lambda_j L_j B (B_j + a\iota_j)} = \iota_j \cdot \frac{B_j + a}{B_j + a\iota_j} \ .$$

Thus, clearly if $\iota_j = 1$, then $\iota'_j = 1$.

If $\iota_j < 1$, then $B_j + a\iota_j < B_j + a$ and thus

$$\iota'_j = \iota_j \cdot \frac{B_j + a}{B_j + a\iota_j} > \iota_j \ .$$

Moreover,

$$\iota'_j = \iota_j \cdot \frac{B_j + a}{B_j + a\iota_j} = \frac{\iota_j B_j + a\iota_j}{B_j + a\iota_j} < 1$$

because $\iota_j B_j < B_j$.

Finally, if $\iota_j > 1$, then $B_j + a\iota_j > B_j + a$ and thus

$$\iota'_j = \iota_j \cdot \frac{B_j + a}{B_j + a\iota_j} < \iota_j \ .$$

Moreover,

$$\iota'_j = \iota_j \cdot \frac{B_j + a}{B_j + a\iota_j} = \frac{\iota_j B_j + a\iota_j}{B_j + a\iota_j} > 1$$

because $\iota_j B_j > B_j$. □

**Lemma 5.** *Consider a RAMM pool with $n$ tokens. Let $k \in \{1, 2, \ldots, n\}$. Suppose that $L_k \neq 0$ and that $L_j = 0$ for all $j \in \{1, 2, \ldots, n\} - \{k\}$. Then, if a liquidity provider redeems an amount $L_k$ of LP tokens of type $k$, he will receive all the remaining assets from the pool and the pool will have no assets left.*

*Proof.* Suppose first that $B_k \neq 0$. In this case the liquidity provider will receive assets with value equal to $a$ tokens $k$ where

$$a = \frac{L_k B_k}{L_k \iota_k} = \frac{B_k}{\iota_k} = \frac{\lambda_k L_k B}{L} = \frac{\lambda_k L_k B}{\lambda_k p_k L_k} = \frac{B}{p_k} \ .$$

Note that the value of the liquidity provider's position in terms of the chosen stablecoin or currency is $ap_k$ which is equal to $B$. Therefore, the liquidity provider will receive all the remaining assets from the pool and, after the withdrawal, the pool will be empty, as expected.

Suppose now that $B_k = 0$. In this case the liquidity provider will receive assets with value equal to $a$ tokens $k$ where

$$a = \frac{L_k \lambda_k B}{L} = \frac{\lambda_k L_k B}{\lambda_k p_k L_k} = \frac{B}{p_k} \ .$$

The proof then continues as in the previous case. □

**Proposition 6.** *Consider a RAMM pool with $n$ tokens. Then $B = 0$ if and only if $L = 0$.*

*Proof.* Suppose first that $L = 0$. If the RAMM pool did not perform any transactions, then $B = 0$, as desired. Otherwise, there has been at least one transaction in the pool. Since $L = 0$, the last transaction of the pool must have been a liquidity withdrawal. Note that before that last transaction, the pool state satisfies that there exists $k \in \{1, 2, \ldots, n\}$ such that $L_k \neq 0$ and that $L_j = 0$ for all $j \in \{1, 2, \ldots, n\} - \{k\}$. Moreover, in the last transaction a liquidity provider redeemed an amount $L_k$ of LP tokens of type $k$, since $L = 0$ after that last transaction. Thus, by the previous lemma, $B = 0$ after that liquidity withdrawal.

Suppose now that $B = 0$. If the RAMM pool did not perform any transactions (for example, if the pool has just been created), then $L = 0$, as desired. Otherwise, there has been at least one transaction in the pool. Since $B = 0$, the last transaction of the pool must have been a liquidity withdrawal. Let $k \in \{1, 2, \ldots, n\}$ be the index of the LP token that was redeemed in the last transaction and let $T > 0$ be the amount of LP tokens of type $k$ redeemed in it. Let $a$ be the value that the liquidity provider receives in terms of token $k$, and let $B_k$ be the balance of token $k$ in the RAMM pool just before the liquidity withdrawal. For each $j \in \{1, 2, \ldots, n\}$, let $L'_j$ be the amount of LP tokens of type $j$ in circulation before the liquidity withdrawal, and let $B'$ and $L'$ be the values of the parameters $B$ and $L$ before the liquidity withdrawal.

If $B_k \neq 0$ then

$$a = \frac{T B_k}{L'_k \iota_k} = \frac{T}{L'_k} \frac{\lambda_k L'_k B'}{L'} = \frac{T \lambda_k B'}{L'} \ .$$

Note that if $B_k = 0$ then $a = \frac{T \lambda_k B'}{L'}$ too (by the liquidity withdrawal formulas). Since $L'_k \geq T$, in any case, we obtain that

$$a = \frac{T \lambda_k B'}{L'} \leq \frac{T \lambda_k B'}{L'_k \lambda_k p_k} \leq \frac{T \lambda_k B'}{T \lambda_k p_k} = \frac{B'}{p_k} \ .$$

Since $B = 0$ it follows that $ap_k = B'$. Thus, the previous inequalities must be equalities and hence $L' = T \lambda_k p_k$. Then, $L'_k = T$ and $L'_j = 0$ for all $j \in \{1, 2, \ldots, n\} - \{k\}$. Therefore, $L = 0$. □

**Lemma 7.** *Let $j \in \{1, 2, \ldots, n\}$. Suppose that a liquidity provider provides liquidity to a RAMM pool depositing a certain amount of token $j$ into the pool. If just before the deposit $B_j = 0$ and $L_j = 0$, then the imbalance ratio of token $j$ after the deposit is equal to 1.*

*Proof.* Let $a$ be the amount of token $j$ that the liquidity provider deposits and let $T$ be the amount of LP tokens of type $j$ that the liquidity provider receives. Let $L$ and $B$ be the parameters defined on page 5 computed before the liquidity deposit.

Case 1: Suppose first that $B \neq 0$. Then $T = \frac{La}{\lambda_j B}$. Let $\iota'_j$ be the imbalance ratio of token $j$ after the deposit. Then

$$\iota'_j = \frac{a(L + \lambda_j p_j T)}{\lambda_j T(B + p_j a)} = \frac{a(L + \lambda_j p_j \frac{La}{\lambda_j B})}{\lambda_j \frac{La}{\lambda_j B}(B + p_j a)} = \frac{(L + p_j \frac{La}{B})}{\frac{L}{B}(B + p_j a)} = \frac{\frac{L}{B}(B + p_j a)}{\frac{L}{B}(B + p_j a)} = 1 \ ,$$

as we wanted to prove.

Case 2: Suppose now that $B = 0$. Then $T = \frac{a}{\lambda_j}$. Note that $L = 0$ by Proposition 6. Let $\iota'_j$ be the imbalance ratio of token $j$ after the deposit. Then

$$\iota'_j = \frac{a\lambda_j p_j T}{\lambda_j T p_j a} = 1 \ ,$$

as we wanted to prove. □

**Proposition 8.** *If a liquidity provider deposits an amount $a$ of token $j$ and immediately withdraws it, he will obtain again the same value as that the amount $a$ of token $j$ has.*

*Proof.* Let $B_j$ be the balance of token $j$ before the liquidity deposit and let $L_j$ be the amount of LP tokens of type $j$ in circulation before the deposit. Let $T$ be the amount of LP tokens of type $j$ that the liquidity provider receives. Let $L$ and $B$ be the parameters defined on page 5 computed before the liquidity deposit.

Case 1: Suppose first that $B \neq 0$. Then $T = \frac{La}{\lambda_j B}$. From formula (9), we know that the amount of token $j$ that the liquidity provider has to receive when redeeming his $T$ LP tokens of type $j$ is

$$a' = \frac{T(B_j + a)}{(L_j + T)\iota'_j} \ ,$$

where $\iota'_j$ is the imbalance ratio of token $j$ after the deposit. In addition, from the analysis done in the previous section we know that the liquidity provider will receive the value that the amount $a'$ of token $j$ is worth, either in token $j$ solely or in a combination of several tokens. Therefore, it suffices to prove that $a' = a$.

We will analyze two cases: $B_j = 0$ and $B_j \neq 0$.

Case 1.1: Suppose first that $B_j = 0$. If $L_j = 0$ then $\iota'_j = 1$ by Lemma 7. Hence,

$$a' = \frac{T(B_j + a)}{(L_j + T)\iota'_j} = \frac{Ta}{T\iota'_j} = a \ .$$

We assume now that $L_j \neq 0$. We know that

$$\iota'_j = \frac{a(L + \lambda_j p_j T)}{\lambda_j (L_j + T)(B + p_j a)} \ ,$$

where $L$ and $B$ are the parameters defined on page 5 computed before the liquidity deposit. We have that

$$\iota'_j = \frac{a(L + \lambda_j p_j T)}{\lambda_j (L_j + T)(B + p_j a)} = \frac{a(L + \lambda_j p_j \frac{La}{\lambda_j B})}{\lambda_j (L_j + T)(B + p_j a)} = \frac{a(L + p_j \frac{La}{B})}{\lambda_j (L_j + T)(B + p_j a)} =$$
$$= \frac{a\frac{L}{B}(B + p_j a)}{\lambda_j (L_j + T)(B + p_j a)} = \frac{a\frac{L}{B}}{\lambda_j (L_j + T)} \ .$$

Then,

$$a' = \frac{T(B_j + a)}{(L_j + T)\iota'_j} = \frac{Ta}{(L_j + T)\iota'_j} = \frac{\frac{La}{\lambda_j B} \cdot a}{L_j + T} \cdot \frac{\lambda_j(L_j + T)}{a\frac{L}{B}} = a \ .$$

Note that this case should not happen as it is Case X of the previous section. Nevertheless, the formulas also work in this case.

Case 1.2: Suppose now that $B_j \neq 0$. Let $\iota'_j$ be the imbalance ratio of token $j$ after the deposit. Note that

$$\iota'_j = \frac{(B_j + a)(L + \lambda_j p_j T)}{\lambda_j(L_j + T)(B + p_j a)} \ .$$

Thus,

$$a' = \frac{T(B_j + a)}{(L_j + T)\iota'_j} = \frac{T(B_j + a)}{(L_j + T)} \frac{\lambda_j(L_j + T)(B + p_j a)}{(B_j + a)(L + \lambda_j p_j T)} = \frac{T\lambda_j(B + p_j a)}{L + \lambda_j p_j T} = \frac{\frac{La}{\lambda_j B}\lambda_j(B + p_j a)}{L + \lambda_j p_j \frac{La}{\lambda_j B}} =$$
$$= \frac{La(B + p_j a)}{B(L + p_j \frac{La}{B})} = \frac{La(B + p_j a)}{B\frac{L}{B}(B + p_j a)} = a \ ,$$

as we wanted to prove.

Case 2: Now suppose that $B = 0$. Then, from Proposition 6 we obtain that $L = 0$. The result then follows from Lemma 5. □

**Lemma 9.** *Let $a > 0$. Then, for all $x > 0$,*

$$\left(\frac{1}{1 + a}\right)^x \geq 1 - ax \ .$$

*Proof.* Let $f \colon \mathbb{R} \to \mathbb{R}$ be given by $f(t) = \left(\frac{1}{1+a}\right)^t$, and let $x > 0$. By the Mean Value Theorem, there exists $c \in (0, x)$ such that $\frac{f(x)-f(0)}{x-0} = f'(c)$, that is,

$$\frac{\left(\frac{1}{1+a}\right)^x - 1}{x} = \left(\frac{1}{1 + a}\right)^c \ln(\tfrac{1}{1+a}) \ .$$

Thus,

$$\left(\frac{1}{1 + a}\right)^x = x\left(\frac{1}{1 + a}\right)^c \ln(\tfrac{1}{1+a}) + 1 = -x\frac{1}{(1 + a)^c}\ln(1 + a) + 1 \geq 1 - x\ln(1 + a) \geq 1 - ax \ ,$$

since $\ln(1 + a) \leq a$. □

**Proposition 10.** *Let $i, o \in \{1, 2, \ldots, n\}$. Suppose that a trader deposits an amount $A_i$ of token $i$ and receives an amount $A_o$ of token $o$. Let $\phi \in [0, 1)$ be the fee ratio of the trade.*

- *Let $p$ be the oracle price of token $i$ in terms of token $o$. Let $p_e$ be the effective price of token $i$ in terms of token $o$ paid to the trader. Then*

$$p_e = \frac{A_o}{A_i} \leq (1 - \phi)p \ .$$

- *Let $p'$ be the oracle price of token $o$ in terms of token $i$. Let $p'_e$ be the effective price of token $o$ in terms of token $i$ paid by the trader. Then*

$$p'_e = \frac{A_i}{A_o} \geq \frac{1}{1 - \phi}p' \ .$$

*Proof.* Clearly $p_e = \frac{A_o}{A_i}$. Applying formula (3) we obtain that

$$p_e = \frac{A_o}{A_i} = \frac{kB_o}{A_i}\left(1 - \left(\frac{kB_i}{kB_i + (1-\phi)A_i}\right)^{\frac{w_i}{w_o}}\right) = \frac{kB_o}{A_i}\left(1 - \left(\frac{1}{1 + \frac{(1-\phi)A_i}{kB_i}}\right)^{\frac{w_i}{w_o}}\right) \leq$$
$$\leq \frac{kB_o}{A_i}\left(1 - \left(1 - \frac{(1-\phi)A_i}{kB_i}\frac{w_i}{w_o}\right)\right) = \frac{kB_o}{A_i}\frac{(1-\phi)A_i}{kB_i}\frac{w_i}{w_o} = (1-\phi)\frac{B_o}{B_i}\frac{w_i}{w_o} \ ,$$

where the inequality holds by the previous lemma. Now, from formula (2) we obtain that

$$\frac{w_i}{w_o} = \frac{p_i B_i}{p_o B_o} = p\frac{B_i}{B_o} \ .$$

Thus,

$$p_e \leq (1-\phi)\frac{B_o}{B_i}\frac{w_i}{w_o} = (1-\phi)\frac{B_o}{B_i}p\frac{B_i}{B_o} = (1-\phi)p \ ,$$

which proves the first item.

Therefore, we have proved that

$$\frac{A_o}{A_i} = p_e \leq (1-\phi)p = (1-\phi)\frac{p_i}{p_o} \ .$$

Hence,

$$p'_e = \frac{A_i}{A_o} \geq \frac{1}{1-\phi}\cdot\frac{p_o}{p_i} = \frac{1}{1-\phi}p' \ ,$$

which completes the proof. □

**Proposition 11.** *Let $i, o \in \{1, 2, \ldots, n\}$. Let $k \in \{1, 2, \ldots, n\} - \{i, o\}$. Suppose that a trader deposits an amount $A_i$ of token $i$ and receives an amount $A_o$ of token $o$. Then, after the trade,*

- *the value of the the parameter $B$ defined on page 5 increases, and*
- *the imbalance ratio of token $k$ decreases.*

*Proof.* Let $B'$ and $B''$ be the values of the parameter $B$ defined on page 5 just before and after the trade, respectively. Let $\iota'_k$ and $\iota''_k$ be the values of the imbalance ratio of token $k$ just before and after the trade, respectively.

Note that

$$B'' = B' + p_i A_i - p_o A_o = B' + p_i A_i\left(1 - \frac{p_o}{p_i}\frac{A_o}{A_i}\right) \ .$$

By Proposition 10, $\frac{A_o}{A_i} \leq (1-\phi)p$, where $p$ is the oracle price of token $i$ in terms of token $o$ and $\phi$ is the pool fee. Note that $p = \frac{p_i}{p_o}$. Thus,

$$B'' = B' + p_i A_i\left(1 - \frac{p_o}{p_i}\frac{A_o}{A_i}\right) \geq B' + p_i A_i\left(1 - \frac{p_o}{p_i}(1-\phi)p\right) = B' + p_i A_i \phi \geq B' \ .$$

In addition, since the values of $B_k$, $\lambda_k$ and $L_k$ do not change with the trade, we obtain that

$$\iota''_k = \frac{B_k L}{\lambda_k L_k B''} \leq \frac{B_k L}{\lambda_k L_k B'} = \iota'_k \ ,$$

as we wanted to prove. □

**Acknowledgements.** The author would like to thank Hisham Khan for useful discussions during the period in which the RAMM was being designed.

## References


[1] Adams, H., Zinsmeister, N., Salem, M., Keefer, R., and Robinson, D. Uniswap v3 Core. (2021).
https://uniswap.org/whitepaper-v3.pdf.

[2] Bouba, D. Swaap.finance: Introducing the Matrix-MM. (2021).
https://www.swaap.finance/whitepaper.pdf.

[3] Egorov, M. StableSwap - efficient mechanism for Stablecoin liquidity. (2019).
https://curve.fi/files/stableswap-paper.pdf.

[4] Ottina, M., Steffensen, P. J., and Kristensen, J. *Automated Market Makers: A Practical Guide to Decentralized Exchanges and Cryptocurrency Trading*. Apress, 2023.
https://www.amazon.com/Automated-Market-Makers-Decentralized-Cryptocurrency/dp/1484286154.

[5] The Lifinity team. Lifinity: A Proactive Market Maker with Concentrated Liquidity. (2022).
https://lifinity.io/litepaper.