Temporal-Function Market Making Litepaper

QuantAMM Team

Version 0.1.0
17 November 2023

Abstract

First generation AMMs instantiate an extremely limited and fixed strategy: a simple holding strategy very often provides superior returns compared to depositing into a CFMM. Recent analysis has expanded on the issue of loss vs rebalancing, and LPs have moved to new solutions of ranged liquidity and limit order books. The fixed-weight strategy is a defining reason for the difficulties of AMMs, and altering the strategies applied is the simplest and most effective mechanism to make AMMs profitable.

We introduce Temporal-Function Market Marking (TFMM), where the trading function of an AMM pool, and thus its desired deployment of value, changes with time. Differing from pools that allow external or stochastic updates to weight vectors, TFMMs enables dynamic AMM pools that run continuous, fully-on-chain quantitative strategies.

We give a short background on the fundamentals of AMMs and then outline, both theoretically and in simulation (expanded to real-world scenarios in the QuantAMM paper), how TFMMs can outperform HODL as well as earlier-generation AMMs.

We then expand on how a dynamic re-weighting of an AMM pool can be implemented inexpensively on-chain. Further, a novel approach to hierarchical composability of AMM pools (composite pools) allows automated risk management on top of multiple strategies, without adding additional trade routing complexity.

1 Introduction

In recent years, automated market makers (AMMs) and decentralised exchanges (DEXs) have grown to become integral parts of decentralised Finance (DeFi) infrastructure. These systems notionally enable liquidity providers (LPs), the depositors, to earn a return on their capital, their liquidity, by enabling traders to exchange one token for another against the deposited capital. For providing liquidity, LPs receive 'liquidity tokens' representing their share of the capital deposited in the pool. Trading fees go to LPs in proportion to their LP token holdings. Some of these systems allow liquidity provision over a pair of tokens, others a basket.

There is a massive and well known problem: AMM pools generally lose value for their LPs, even after fees are taken into account. This is known as Impermanent Loss. LVR [Milionis et al., 2022] is a separate (but related Nezlobin [2022]) metric for measuring AMM performance, where fixed-weight AMM pool's rebalancing is compared to rebalancing via a CEX, and again AMM pools perform poorly. Even recent advances in AMM technology suffer these problems.

TFMMs attack this problem by going back to first principles and recognising that a fixed-allocation strategy is just one choice of opinionated strategy. Once this is realised, the fundamental question can be asked if other, non-fixed-allocation, strategies are cursed to also suffer from impermanent loss –we discover both in theory and in practice that this is not the case. We extend the basic notion of constant-function market makers (CFMMs) by developing a decentralised method of dynamically re-weighting the held reserves using oracle signals and functions thereof, increasing the available returns on deposited capital. This is done by continuously changing the portfolio vector inside the constant function of the pool, where the new portfolio vectors are a (chosen) function of (chosen) oracles. In fact, TFMM pool re-balancing is often substantially cheaper than, in an LVR-like comparison, running a 'mirrored' strategy that is re-balanced by trading with a CEX.

2 Temporal-Function Market Makers

Our proposal is to enable $\mathbf{w}$, the portfolio vector, to update continuously on-chain using low frequency, periodic on-chain oracle calls, for example using the price of the tokens in the pool: the allocation of assets in the pool is thus responsive to market information. This is the key to how TFMMs can potentially circumvent impermanent loss. We will write $\mathbf{w}(t)$ to indicate this dependency on time, and the time-varying equation that defines a TFMM pool block-to-block, the *trading function*, becomes:

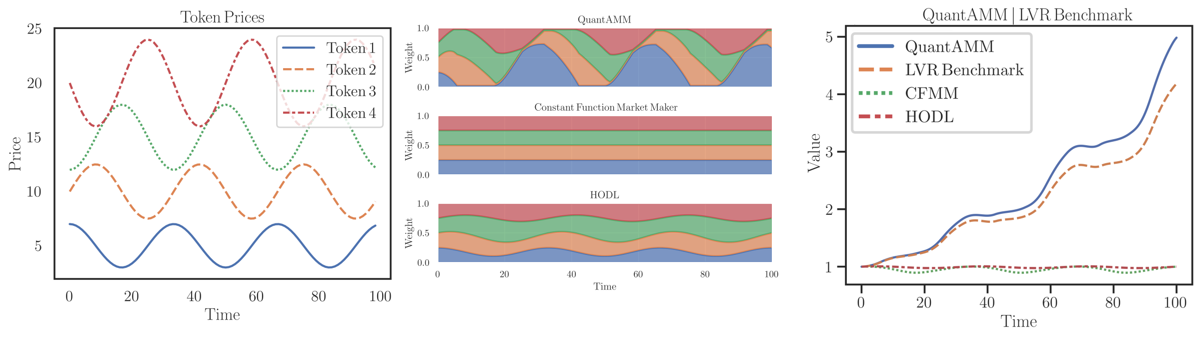
$$\prod_{i=1}^{N} R_i^{w_i(t)} = k(t), \quad \text{where} \sum_{i=1}^{N} w_i(t) = 1, \text{ and } \forall i \; 0 < w_i(t) < 1. \tag{1}$$

A trade still has to preserve (or increase) the value of $k(t)$ that exists at the time of that trade. Weight changes when the pool is in equilibrium—quoting the market price—create an arbitrage opportunity. This is desirable, we need to incentivise the pool's reserves to be re-balanced to the new weights.

We call the resulting systems *Temporal Function Market Makers*. While some CFMMs allow external and stochastic updates to a pool's $\mathbf{w}$ vector, TFMMs have continuously changing $\mathbf{w}$s, given by an open-horizon update rule, that never reach some particular final weight. This has cumulative impact on almost all aspects of AMM functionality and implementation sufficient to warrant a distinction from CFMMs.[1]

2.1 Superior performance of TFMM Pools

Figure 1: Synthetic example of TFMM performance.



(a) Synthetic price data. (b) Weight changes over time (c) Total USD pool value

Fig 1a Simulation Data: Four token sinusoidal price trajectories that are offset and used every hour to update the pool's weights.

Fig 1b Simulation analysis: a TFMM pool running a mean-reversion update rule[2] actively changes its quoted weights over time compared to a Balancer pool. While HODL does not use CFMM pricing and trading, we also show HODL proportional USD market value weights that change directly corresponding to price fluctuations.

Fig 1c Simulation results: TFMM returns over time generate superior pool value compared to the Loss-versus-Rebalancing (LVR) Milionis et al. [2022] benchmark's returns over time. This benchmark is, roughly speaking, from rebalancing the same portfolio but via a CEX. See 4.1 and the full TFMM paper for more detail. We also plot HODL and a fixed-CFMM pool for comparison.

While this is a simulation based on idealised synthetic data, the relationship between the data and the strategy is very important. See the QuantAMM protocol papers for test results of multiple update rules against historic and Monte Carlo data–results showing considerable alpha over HODL for most market conditions. All simulations include advanced components such as fee modelling and "no-arb region" modelling.

[1]More broadly, TFMMs are also different to Replicating Market Makers (RMMs) in that TFMMs naturally implement quantitative strategies for the reallocation of value over a basket of many assets over time, whereas RMMs aim to mimic the payoffs of (often finite-duration) financial instruments (eg options under a Black-Scholes pricing model).

[2]The update rules is within the family given by Eq (13), see QuantAMM litepaper for more detail.

2.2 TFMM innovations beyond current AMMs

Update Rules: Strategies for Pools The first question is: what algorithm should determine pools' weight changes over time? Traditional finance can provide an encyclopedia of possible strategies, which we call *update rules*. While some protocols attempt TradFi strategies by introducing off-chain dependencies and calculations, novel mathematical approaches had to be developed to make TradFi strategies feasible to be run continuously and completely on-chain. As a result, TFMM pools can even run successfully on L1 with relatively small TVL.

Changes in quoted weights motivate the pool's reserves to be passively re-balanced via an arbitrage opportunity. While the strategy might update the weight *trajectories* every hour or every day using oracles, the weights visible to traders and to the market changes every block solely using stored variables. New analysis on multi-block attack vectors, Appendix C, informs explicit tuning of the block-by-block weight change protections based on the level of protection the pool creator feels comfortable with. To the best of our knowledge such protections are neither explored or tuned in any other protocol.

Composite pools: Enabling diverse portfolios and re-balancing strategies Novel architecture in TFMMs enables LPs to avoid split management of multiple pools by constructing "pools of pools": the liquidity placed in a composite pool is distributed across its underlying pools, enabling an LP to benefit from exposure to these underlying pools.

While some protocols enable virtual or phantom pools, these are for the benefit of the trader and not the reallocation of capital between pools. Composite pools allow for liquidity to be shifted between base pools that are intended to perform better in different market conditions—for example between a 'bull market' subpool and a 'bear market' subpool.

AMM "execution management" techniques When requiring large weight changes, such a move could produce large slippage if attempted naively. While simple execution management techniques can perform well, some novel techniques (computationally cheap enough to be run on-chain) are explored to increase efficiencies even further. This makes the inter-block weight updates somewhat reminiscent of trades in Time-weighted average market makers (TWAMMs) [White et al., 2021]. However unlike traders on TWAMMs, TFMMs rebalance by being a 'price maker' rather than a 'price taker'.

Advanced pool management and tooling Given easy parallels to TradFi strategies and predictability, TFMMs allow automated advanced tooling to lower sophistication barriers. An example of tooling that can be provided is auto-generated key investor information documents (KIID). With regards to alpha leak, for instance if a new update rule creator does not want to expose all aspects of that rule on-chain, ZK-wrapped strategies conform with both the spirit of continuous, on-chain TFMMs (without manual or voted updates) while keeping the exact fixed strategy mechanics private.

TFMM interoperability with current AMM ecosystems While TFMM introduces automated and effective inter-block functionality to deal with current CFMM shortfalls, certain aspects are kept consistent with CFMMs. This means that no drastically new infrastructure is needed to integrate TFMMs with current DEX ecosystems of aggregation, leverage and arbitrage. Multi-token trades are allowed on TFMMs in the same manner as other geometric mean market makers, as well as standard quoting of some amount Λ_k of the k^{th} token in exchange Δ_j for a j^{th} token:

$$\Lambda_k = R_k \left(1 - \frac{1}{\left(1 + \gamma \frac{\Delta_j}{R_j}\right)}^{w_j(t)/w_k(t)}\right) \tag{2}$$

Exactly like pools on prior AMMs, TFMM pools issue liquidity tokens to LPs in proportion to the value they place in a pool, whether this is in a single or multi asset deposit. At time t, the total supply of liquidity tokens $\tau_{\mathcal{P}}(t)$ for a pool $\mathcal{P}$ is therefore given by $\tau_{\mathcal{P}} = s(\mathbf{w}(t))\, k(t)$, where s is a scaling factor that determines the total size of the supply and depends on the current weights. Within a block, s is constant.

3 How TFMMs beat Impermanent Loss when CFMMs cannot

Non-Dynamic AMM pools must suffer Impermanent Loss In previous AMMs liquidity providers suffer from impermanent (a.k.a. divergence) loss: any change in the ratios of the market prices of tokens in a basket leads to that pool holding less value than when the pool was capitalised. When the market prices change from some initial value $\mathbf{p}(t_0)$ to a later value $\mathbf{p}(t')$; arbitrageurs then trade with the pool until, once again, the quoted prices of the pool match the *new* market prices. Performing this analysis, the pool's reserves become

$$\mathbf{R}(t') = \mathbf{R}(t_0)\frac{\mathbf{p}(t_0)}{\mathbf{p}(t')}\prod_{i=1}^{N}\left(\frac{p_i(t')}{p_i(t_0)}\right)^{w_i}, \tag{3}$$

where multiplication and division between vectors is performed elementwise (this is the case for the rest of this section). The value in a pool, in the market-price numéraire, at any time t is $V(t) = \mathbf{p}(t)\cdot\mathbf{R}(t) = \sum_i^N p_i(t)R_i(t)$. From this we can derive the equation that describes Impermanent Loss. ΔV, the proportional change in the value of the pool compared to holding, is

$$\Delta V = \frac{\frac{V(t')}{V(t_0)}}{\frac{V_{\text{hold}}(t')}{V_{\text{hold}}(t_0)}} - 1 = \frac{\prod_{k=1}^{N}\left(\frac{p_k(t')}{p_k(t_0)}\right)^{w_k}}{\sum_{i=1}^{n} w_i\frac{p_i(t')}{p_i(t_0)}} - 1. \tag{4}$$

This function range is ≤ 0, due to the Weighted AM–GM inequality; in other words, for any $\mathbf{p}(t') \neq b\mathbf{p}(t_0)$ (where b is any scalar > 0), $\Delta V < 0$ so liquidity providers have lost value compared to holding. For full derivation, see the full TFMM paper.

TFMM pools do not have to suffer Impermanent Loss How do the reserves, and thus value, of a TFMM pool change when the weights change? We consider *constant* market prices with *changing* weights. Arbitrageurs will trade with the pool until the quoted prices of the pool again match the current market prices. Initial weights are $\mathbf{w}(t_0)$, and a block later, with time $t' = t_0 + \delta_t$, we have new weights $\mathbf{w}(t')$ (and $\mathbf{p}(t') = \mathbf{p}(t_0)$). The new reserves and value after the weight-change are

$$\mathbf{R}_{\Delta w}(t') = \mathbf{R}(t_0)\frac{\mathbf{w}(t')}{\mathbf{w}(t_0)}\prod_{i=1}^{N}\left(\frac{w_i(t_0)}{w_i(t')}\right)^{w_i(t')}, \tag{5}$$

$$V_{\Delta w}(t') = \sum_{i=1}^{N} p_i(t')R_i(t') = \sum_{i=1}^{N} p_i(t_0)R_i(t') = \sum_{i=1}^{N} p_i(t_0)R_i(t_0)\frac{w_i(t')}{w_i(t_0)}\prod_{k=1}^{N}\left(\frac{w_k(t_0)}{w_k(t')}\right)^{w_k(t')}. \tag{6}$$

Of course when a TFMM pool is running, market prices will change, so to model how reserves change we have to apply both Eq (3) and Eq (5). Within a single block the market prices change, and then between blocks the weights change, so we get

$$\mathbf{R}_{\text{TFMM}}(t') = \mathbf{R}(t_0)\frac{\mathbf{p}(t_0)}{\mathbf{p}(t')}\frac{\mathbf{w}(t')}{\mathbf{w}(t_0)}\prod_{i=1}^{N}\left(\frac{p_i(t')}{p_i(t_0)}\right)^{w_i(t_0)}\prod_{k=1}^{N}\left(\frac{w_k(t_0)}{w_k(t')}\right)^{w_k(t')}. \tag{7}$$

For full derivation, see the full TFMM paper. From the path-dependency of Eq (5) (when applied repeatedly, terms involving intermediate values of variables do not cancel out), this 'combined' update is also path-dependent and now depends on *both* the trajectory of prices and that of weights. The corresponding change in value for one combined update is

$$V_{\text{TFMM}}(t') = \sum_{i=1}^{N} p_i(t')R_{\text{TFMM},i}(t') = \sum_{i=1}^{N} p_i(t_0)R_i(t_0)\frac{p_i(t_0)}{p_i(t')}\frac{w_i(t')}{w_i(t_0)}\prod_{k=1}^{N}\left(\frac{p_k(t')}{p_k(t_0)}\right)^{w_k(t_0)}\prod_{\ell=1}^{N}\left(\frac{w_\ell(t_0)}{w_\ell(t')}\right)^{w_\ell(t')}. \tag{8}$$

We cannot write a simple equivalent to Eq (4) for these combined TFMM updates—the value of the TFMM pool at some later time is not guaranteed to be less than holding one's initial capital, as it is for non-dynamic AMM pools. *TFMM pools do not have to suffer from Impermanent Loss* (but of course capital is at risk, update rules can be ill-tuned or have market behaviour change such that they no longer operate as expected). The value of a TFMM pool depends intimately on the sequence of weights and prices over time, and as such its value can increase above simply holding one's capital.

4 Rebalancing efficiency and comparisons

Weight Interpolation: Execution Management on TFMMs Weight changes when the pool is in equilibrium, i.e. quoting the market price, create an arbitrage opportunity. It is desirable for LPs for the arbitrage opportunity to be smaller rather than larger to reduce slippage. This is roughly analogous to 'execution management' in TradFi, where a trader wishes for their transactions to follow best execution policies that reduce cost. A simple way to reduce the arbitrage opportunity from a weight update is to spread out the weight update out over a period in time.

General Case We can want to compare $\mathbf{R}^{\text{1-step}}$, the reserves in the pool when we have directly updated the weights from $\mathbf{w}(t_0) \to \mathbf{w}(t_0) + \Delta\mathbf{w}$, a one step process, to $\mathbf{R}^{\text{2-step}}$, the reserves when we have done this weight update via a two step process: $\mathbf{w}(t_0) \to \tilde{\mathbf{w}} \to \mathbf{w}(t_0) + \Delta\mathbf{w}$, where $\forall i$, $\tilde{w}_i \in [w_i(t_0), w_i(t_0) + \Delta w_i]$ (as well as $0 < \tilde{w}_i < 1$ and $\sum_{i=1}^N \tilde{w}_i = 1$). We show in Appendix A.3.1 that going through any of this broad family of intermediate values of the weight vector $\mathbf{w}$ gives lower arb cost.

Example: Linear Interpolation We can show this result for the particular case of linear interpolation via Eq (5). Here the two step process is: $\mathbf{w}(t_0) \to \mathbf{w}(t_0) + \frac{1}{2}\Delta\mathbf{w} \to \mathbf{w}(t_0) + \Delta\mathbf{w}$. This technique is used for liquidity bootstrap pools and stochastic changing CFMMs. We find that

$$\mathbf{R}^{\text{2-step}} = \mathbf{R}^{\text{1-step}} \prod_{j=1}^{N} \left(1 + \frac{\Delta w_j}{2 w_j(t_0)}\right)^{\frac{\Delta w_j}{2}}. \tag{9}$$

This is by no means the only mechanism to reduce arbitrage slippage for dynamic inter-block weights and the full TFMM paper describes other more efficient mechanisms that can add further efficiency.[3] The cumulative slippage cost over time means that this can be a significant result.

4.1 Reserve and value changes for small weight changes

Given the above, of particular importance is how TFMM pools' reserves and value change when the weight changes are vanishingly small. From this we can show how TFMM pools rebalance in the limit of small weight changes, including with fees present. The value of the pool post rebalance is

$$V_{\delta w}^{\gamma\text{-fees}}(t') \gtrsim V(t_0) \left(1 + (1-\gamma) \sum_{i=1}^{N} \left(\mathbb{I}_{\delta w_i > 0} \delta w_i\right)\right). \tag{10}$$

These results show how a TFMM pool rebalancing can lead, with swap fees present and under the assumptions made, to an increase in value of the pool.

Comparing to CEX rebalancing We can derive the equivalent results for a more vanilla rebalancing procedure: using a CEX. The CEX-rebalancing counterpart to Eq (10) is

$$V_{\text{trad}}^{\gamma\text{-fees}} > V(t_0) \left(1 - (1-\gamma_{\text{trad}}) \sum_{i=1}^{N} \left(\mathbb{I}_{\delta w_i > 0} \delta w_i\right)\right). \tag{11}$$

Here the pool is a taker, while in the TFMM-rebalancing results above the pool is a maker. This means here the pool is worth slightly *less* than $V(t_0)$ after the rebalance, rather than slightly *more*.

See Appendix A for further details and derivations on weight interpolation, linear interpolation and comparison of TFMM rebalancing vs CEX rebalancing.

[3] And sometimes, it can even be better to not allow a rebalance and instead to temporarily use a HODL strategy for a period of time.

5 Update Rules

Many classical strategies from TradFi can be implemented using the tools available for TFMM pools, as well as many new approaches. One of the main issues with implementing known approaches, especially on L1s, is operational gas cost. In this section we describe advances made to allow construction of TradFi strategies fully on-chain and runnable on L1s. We then provide an example family of rules that can be created using such novel building blocks.

On blockchains with cheaper computations, including L2s, more sophisticated update equations become possible, including calculations involving higher derivatives and special functions.

5.1 On-Chain Methods for Gradients, Covariances and Precisions

Knowing, in isolation, that a signal takes a particular value at the present moment is not necessarily useful for making a decision based on that signal. It is often by knowing how the current signal compares to previous values that enables us to make use of a signal. This naturally means that in building update rules we need a gradient operator, to tell us by how much a market signal has changed. Simple 'rise-over-run' approaches do not work well with financial data due to noise.[4]

Some update rules need access to the variance of an signal or the covariance between signals. For example, any update rule that wants to use Markovitz/Mean-Variance Portfolio Theory (MVPT) or related ideas will need them. We can use a set of mathematical methods, very similar to those used to calculate our method for the gradient of a signal, to calculate these quantities. Note that often one wants not the covariance (nor precision) of raw prices, but instead the covariance (or precision) of returns.

As well as giving us useful, usable, results, we want methods that are sufficiently computationally cheap that it can be done on-chain. Current modelling gives gas costs, running a four token pool with a strategy based on these methods, of **221k gas units per pool per day**. This means the methods are pivotal to enabling TFMM pools on Ethereum Mainnet. Here we will briefly discuss how our gradient method can be used. See the full TFMM paper for information about the covariance and precision methods.

5.2 Trend-Following Update Rules

A broad family of update rules are of the form

$$\mathbf{w}(t) = f\left(\mathbf{w}(t-1), \mathbf{p}(t), \frac{\partial \mathbf{p}(t)}{\partial t}\right), \tag{12}$$

where $\mathbf{p}(t)$ is a vector giving the required oracle values, for example the prices of the N tokens in the pool. How could we chose f? One choice is for the *change* in weights to be a function of $\mathbf{p}(t)$ and its time-derivative giving us a family of momentum-like (ie, trend-following) strategies,

$$\mathbf{w}(t) = \mathbf{w}(t-1) + g\left(\mathbf{p}(t), \frac{\partial \mathbf{p}(t)}{\partial t}\right). \tag{13}$$

For update rules to use them in conjunction with our gradient method, simply replace $\frac{\partial \mathbf{p}(t)}{\partial t}$ with it.

All update rules contain numerical parameters that control their behaviour, and these parameters must be set. Choosing an uninformed parameter value can lead to negative performance of the pool.

[4]Financial time series are of course often modelled as stochastic processes, which are not differentiable (you cannot find a gradient as an infinitesimal rise-over-run calculation).

6 Composite pools

A problem with existing AMMs which aspire to manage LPs' value for them is that most trades are executed against pools containing only two or three assets: such small baskets of tokens have downsides as they do not allow for broad exposure to different tokens. LPs who seek to distribute their liquidity across a number of assets—that is, LPs who seek to deposit into pools face typical overhead in maintaining and running such a portfolio.

The mathematical architecture of TFMM pools enables LPs to circumvent this dilemma, through *composite pools* constructed by "pooling pools": the capital placed in a composite pool is distributed across its underlying pools, enabling an LP to benefit from the volume in these underlying pools while covering their assets of interest. This method differs from current virtual and phantom pools.

This structuring is pivotal to allow the creation of complex portfolios with multiple layers of capital allocation logic while never increasing stack complexity.

For example: Different sub-pools can also contain the same tokens but have different update rules that are highly tuned to specific market conditions, such as bull runs or bear runs. Then a composite pool could manage risk by shifting capital from one pool to the other using an update rule that applies standard risk management algorithms, allowing for LP controlled automatic risk management as well as automatic strategy based constituent token re-balancing.

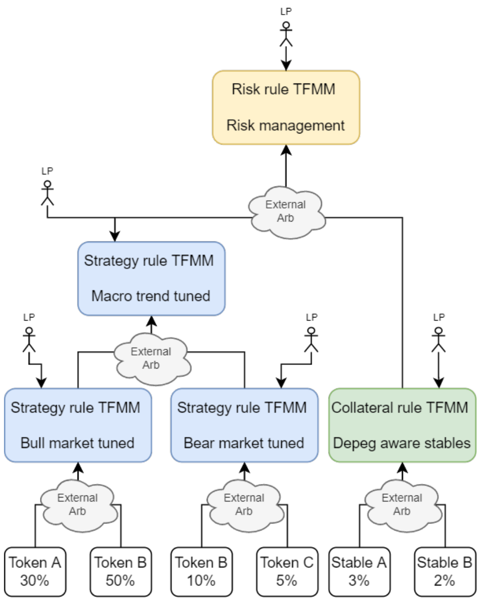


Figure 2: Example LP portfolio structure without added trade stack complexity. Liquidity depth is also maintained as the tokens' balances remain in the base pools.

This approach also allows for more tailored sleeves or managed accounts within a broad portfolio, direct participation at any level while, crucially, never increasing execution management complexity or fragmenting liquidity depth in the base layer pools.

7 Multi-block MEV protection

A key consideration for a TFMM pool is whether, through the weightings between tokens changing over time, their temporal dependence leads to an opportunity for an MEV attack against LPs. This situation is roughly analogous to a 'sandwich attack' on a trade in a vanilla AMM pool. Recall that the changes in weight happen, in effect, *between blocks* making weight updates somewhat like TWAMM trades. This would require *multi-block* MEV manipulation. The TFMM paper contains a range of detailed mathematical analysis and simulation tests of novel protections layered on top of standard AMM protections. See Appendix C for detailed theoretical analysis. This means that TFMM rebalancing can avoid MEV costs being accrued to LPs as their capital is reallocated in accordance with their pool's update rule.

Technical Appendix for TFMM Litepaper

A Reserve-Update Derivations

A.1 Weights Constant, Prices Change

Here we derive Eq (3), the change in reserves of a pool where weights are fixed but market prices change. For simplicity we are assuming the no-fees case.

First, recall that to calculate $p_{i,j}^{\mathrm{TFMM}}(t)$, the prices of the i^{th} token in terms of the j^{th}, that is, using the j^{th} token as the numéraire, simply divide the i^{th} unscaled price by the j^{th},

$$p_{i,j}^{\mathrm{TFMM}}(t) = \frac{\left(\frac{\partial}{\partial \mathbf{R}} k(t)\right)_i}{\left(\frac{\partial}{\partial \mathbf{R}} k(t)\right)_j} = \frac{w_i(t)/R_i}{w_j(t)/R_j}. \tag{A.14}$$

At t_0 we have the invariant of the pool

$$\prod_{i=1}^{N} R_i(t_0)^{w_i} = k, \tag{A.15}$$

and by the requirement that this quantity does not change under trading, we also have that at a later time t'

$$\prod_{i=1}^{N} R_i(t')^{w_i} = k = \prod_{i=1}^{N} R_i(t_0)^{w_i}. \tag{A.16}$$

Thus

$$\prod_{i=1}^{N} \left(\frac{R_i(t')}{R_i(t_0)}\right)^{w_i} = 1. \tag{A.17}$$

Meanwhile, we are assuming that arbitrageurs trade with the pool until the quoted prices match the market prices. At time t_0 we have prices $\mathbf{p}(t_0)$ and at time t' we have prices $\mathbf{p}(t')$. Using Eq (A.14), and without loss of generality using the first token as the numéraire, we have that at equilibrium

$$\frac{\frac{w_i}{R_i(t_0)}}{\frac{w_1}{R_1(t_0)}} = p_i(t_0) \tag{A.18}$$

and

$$\frac{\frac{w_i}{R_i(t')}}{\frac{w_1}{R_1(t')}} = p_i(t'). \tag{A.19}$$

Thus we have

$$\frac{p_i(t_0)}{p_i(t')} = \frac{R_i(t')R_1(t_0)}{R_i(t_0)R_1(t')}, \tag{A.20}$$

$$\frac{R_i(t')}{R_i(t_0)} = \frac{p_i(t_0)R_1(t')}{p_i(t')R_1(t_0)}. \tag{A.21}$$

Subbing Eq (A.21) into Eq (A.17), we get

$$\prod_{i=1}^{N}\left(\frac{R_i(t')}{R_i(t_0)}\right)^{w_i} = \prod_{i=1}^{N}\left(\frac{p_i(t_0)R_1(t')}{p_i(t')R_1(t_0)}\right)^{w_i} = 1 \tag{A.22}$$

$$= \frac{R_1(t')}{R_1(t_0)}\prod_{i=1}^{N}\left(\frac{p_i(t_0)}{p_i(t')}\right)^{w_i} = 1 \tag{A.23}$$

$$\Rightarrow \frac{R_1(t')}{R_1(t_0)} = \prod_{i=1}^{N}\left(\frac{p_i(t')}{p_i(t_0)}\right)^{w_i}. \tag{A.24}$$

Now subbing Eq (A.24) into Eq (A.20) and rearranging we get

$$R_i(t') = R_i(t_0)\frac{p_i(t_0)}{p_i(t')}\prod_{j=1}^{N}\left(\frac{p_j(t')}{p_j(t_0)}\right)^{w_j}, \tag{A.25}$$

as we have in Eq (3) above, completing this derivation.

A.2 Weights Change, Prices Constant

Here we derive Eq (5), the change in reserves of a pool where weights change but market prices are fixed. For simplicity we are assuming the no-fees case.

At time t_0 we have weights $\mathbf{w}(t_0)$ and a block later $t' = t_0 + \delta_t$ we have weights $\mathbf{w}(t')$.

Just after the change in weights, we have the invariant of the pool

$$\prod_{i=1}^{N} R_i(t_0)^{w_i(t')} = k(t'), \tag{A.26}$$

and by the requirement that this quantity does not change under trading, we also have that after trading

$$\prod_{i=1}^{N} R_i(t')^{w_i(t')} = k(t') = \prod_{i=1}^{N} R_i(t_0)^{w_i(t')}. \tag{A.27}$$

$$\Rightarrow \prod_{i=1}^{N}\left(\frac{R_i(t')}{R_i(t_0)}\right)^{w_i(t')} = 1. \tag{A.28}$$

Meanwhile, we are assuming that arbitrageurs trade with the pool until the quoted prices match the market prices. Prices are $\mathbf{p}(t_0)$ before and after the change in weights. Using Eq (A.14), and without loss of generality using the first token as the numéraire, we have that initially, before the change in weights,

$$\frac{\frac{w_i(t_0)}{R_i(t_0)}}{\frac{w_1(t_0)}{R_1(t_0)}} = p_i(t_0) \tag{A.29}$$

and then after both updating the weights & the pool reaching a new equilibrium

$$\frac{\frac{w_i(t')}{R_i(t')}}{\frac{w_1(t')}{R_1(t')}} = p_i(t_0). \tag{A.30}$$

Thus we have

$$\frac{w_i(t')w_1(t_0)}{R_i(t')R_1(t_0)} = \frac{w_1(t')w_i(t_0)}{R_1(t')R_i(t_0)}, \tag{A.31}$$

$$\frac{R_i(t')}{R_i(t_0)} = \frac{w_i(t')w_1(t_0)R_1(t')}{w_1(t')w_i(t_0)R_1(t_0)}. \tag{A.32}$$

Subbing Eq (A.32) into Eq (A.28), we get

$$\prod_{i=1}^{N}\left(\frac{R_i(t')}{R_i(t_0)}\right)^{w_i(t')} = \prod_{i=1}^{N}\left(\frac{w_i(t')w_1(t_0)R_1(t')}{w_1(t')w_i(t_0)R_1(t_0)}\right)^{w_i(t')} = 1 \tag{A.33}$$

$$= \frac{w_1(t_0)R_1(t')}{w_1(t')R_1(t_0)}\prod_{i=1}^{N}\left(\frac{w_i(t')}{w_i(t_0)}\right)^{w_i(t')} = 1 \tag{A.34}$$

$$\Rightarrow \frac{w_1(t_0)R_1(t')}{w_1(t')R_1(t_0)} = \prod_{i=1}^{N}\left(\frac{w_i(t_0)}{w_i(t')}\right)^{w_i(t')}. \tag{A.35}$$

Now subbing Eq (A.35) into Eq (A.32) and rearranging we get

$$R_i(t') = R_i(t_0)\frac{w_i(t')}{w_i(t_0)}\prod_{j=1}^{N}\left(\frac{w_j(t_0)}{w_j(t')}\right)^{w_j(t')}, \tag{A.36}$$

as we have in Eq (5) in the main paper, completing this derivation.

A.3 Two-step updates are better than one-step updates

A.3.1 General Interpolation

We will compare two possible weight change methods, against a background of constant prices. We have $\gamma = 1$.

The first is the simple 1-step process, Eq (A.36), derived above. Here we go directly from initial weights to final weights. For compactness here, we will denote $\mathbf{w}(t_0) + \Delta\mathbf{w}$ as $\mathbf{w}(t_f)$. We have $\mathbf{w}(t_0) \to \mathbf{w}(t_f)$.

The second process is to go to an intermediate set of weights $\tilde{\mathbf{w}}$, be arbed to those weights, and then proceed to the final weights: $\mathbf{w}(t_0) \to \tilde{\mathbf{w}} \to \mathbf{w}(t_f)$. The restriction we place here on the elements of $\tilde{\mathbf{w}}$ is that $\forall i,\ \tilde{w}_i \in [w_i(t_0), w_i(t_f)]$. We enforce the standard requirements on weights, that $0 < \tilde{w}_i < 1$ and $\sum_{i=1}^{N} \tilde{w}_i = 1$.

We will show that the two-step process is always superior as it leads to greater pool reserves.

One-step Eq (A.36) directly gives us the ratio of the change of reserves,

$$\frac{R_i(t_f)}{R_i(t_0)} = r_i^{\text{one-step}} = \frac{w_i(t_f)}{w_i(t_0)}\prod_{j=1}^{N}\left(\frac{w_j(t_0)}{w_j(t_f)}\right)^{w_j(t_f)}.$$

And to the second leg,

Two-step Applying Eq (A.36) to the first leg, we get

$$r_i^{\text{1st}-\text{step}} = \frac{\tilde{w}_i}{w_i(t_0)} \prod_{j=1}^{N} \left(\frac{w_j(t_0)}{\tilde{w}_j} \right)^{\tilde{w}_j}.$$

And to the second leg

$$r_i^{\text{2nd}-\text{step}} = \frac{w_i(t_f)}{\tilde{w}_i} \prod_{j=1}^{N} \left(\frac{\tilde{w}_j}{w_j(t_f)} \right)^{w_j(t_f)}.$$

This gives and overall change of reserves of

$$r_i^{\text{two}-\text{step}} = r_i^{\text{1st}-\text{step}} r_i^{\text{2nd}-\text{step}} = \frac{w_i(t_f)}{w_i(t_0)} \prod_{j=1}^{N} \left(\frac{w_j(t_0)}{\tilde{w}_j} \right)^{\tilde{w}_j} \left(\frac{\tilde{w}_j}{w_j(t_f)} \right)^{w_j(t_f)}.$$

Comparison The ratio $r_i^{\text{two}-\text{step}}/r_i^{\text{one}-\text{step}}$ is

$$r_i^{\text{two}-\text{step}}/r_i^{\text{one}-\text{step}} = r = \prod_{j=1}^{N} \left(\frac{w_j(t_0)}{\tilde{w}_j} \right)^{\tilde{w}_j} \left(\frac{\tilde{w}_j}{w_j(t_f)} \right)^{w_j(t_f)} \left(\frac{w_j(t_f)}{w_j(t_0)} \right)^{w_j(t_f)} \tag{A.37}$$

$$= \prod_{j=1}^{N} \left(\frac{w_j(t_0)}{\tilde{w}_j} \right)^{\tilde{w}_j} \left(\frac{\tilde{w}_j}{w_j(t_0)} \right)^{w_j(t_f)} \tag{A.38}$$

$$= \prod_{j=1}^{N} \frac{w_j(t_0)^{\tilde{w}_j}}{w_j(t_0)^{w_j(t_f)}} \frac{\tilde{w}_j^{w_j(t_f)}}{\tilde{w}_j^{\tilde{w}_j}}. \tag{A.39}$$

If each term in this product is > 1, then we will have shown that the two step process leads to greater final reserves than a one-step process. Recall that $\forall i$, $w_i(t_0) < \tilde{w}_i < w_i(t_f)$. Let us now consider the j^{th} term of the product. There are two cases to consider, either $w_i(t_f) > w_i(t_0)$ or $w_i(t_f) < w_i(t_0)$.

The j^{th} term in the product is

$$f_j = \frac{w_j(t_0)^{\tilde{w}_j}}{w_j(t_0)^{w_j(t_f)}} \frac{\tilde{w}_j^{w_j(t_f)}}{\tilde{w}_j^{\tilde{w}_j}}. \tag{A.40}$$

$w_j(t_f) > w_j(t_0)$: If $w_j(t_f) > w_j(t_0)$, then $w_j(t_f) > \tilde{w}_j$ and $w_j(t_0) < \tilde{w}_j$. Thus $\tilde{w}_j = a_j w_j(t_0)$ for some $a_j > 1$. Subbing into Eq (A.40), we get

$$f_j = \frac{w_j(t_0)^{\tilde{w}_j}}{w_j(t_0)^{w_j(t_f)}} \frac{w_j(t_0)^{w_j(t_f)}}{w_j(t_0)^{\tilde{w}_j}} \frac{a_j^{w_j(t_f)}}{a_j^{\tilde{w}_j}} = a_j^{w_j(t_f)-\tilde{w}_j} > 1.$$

$w_j(t_f) < w_j(t_0)$: If $w_j(t_f) < w_j(t_0)$, then $w_j(t_f) < \tilde{w}_j$ and $w_j(t_0) > \tilde{w}_j$. Thus $w_j(t_0) = b_j \tilde{w}_j$ for some $b_j > 1$. Subbing into Eq (A.40), we get

$$f_j = \frac{\tilde{w}_j^{\tilde{w}_j}}{\tilde{w}_j^{w_j(t_f)}} \frac{\tilde{w}_j^{w_j(t_f)}}{\tilde{w}_j^{\tilde{w}_j}} \frac{b_j^{\tilde{w}_j}}{b_j^{w_j(t_f)}} = b_j^{\tilde{w}_j - w_j(t_f)} > 1.$$

Summary Thus for either increasing elements, $w_j(t_f) > w_j(t_0)$, or decreasing elements, $w_j(t_f) < w_j(t_0)$, the terms in the product r are always > 1, therefore taking the two-step process is always superior, as required.

A.3.2 Linear Interpolation

Here we will derive Eq (9), finding the relationship between $\mathbf{R}^{\text{1-step}}$, the reserves in the pool when we have directly updated the weights from $\mathbf{w}(t_0) \to \mathbf{w}(t_0) + \Delta\mathbf{w}$, a one step process, and $\mathbf{R}^{\text{2-step}}$, the reserves when we have done this weight update via a two step process: $\mathbf{w}(t_0) \to \mathbf{w}(t_0) + \frac{1}{2}\Delta\mathbf{w} \to \mathbf{w}(t_0) + \Delta\mathbf{w}$.

First Eq (A.36) derived above directly gives us the one-step value:

$$R_i^{\text{1-step}} = R_i(t') \tag{A.41}$$

$$= R_i(t_0)\frac{w_i(t_0) + \Delta w_i}{w_i(t_0)} \prod_{j=1}^{N} \left(\frac{w_j(t_0)}{w_j(t_0) + \Delta w_j}\right)^{w_j(t_0)+\Delta w_j}. \tag{A.42}$$

Now let's consider the two-step process. First we do half of the update,

$$R_i^{\Delta w/2} = R_i(t_0)\frac{w_i(t_0) + \frac{\Delta w_i}{2}}{w_i(t_0)} \prod_{j=1}^{N} \left(\frac{w_j(t_0)}{w_j(t_0) + \frac{\Delta w_j}{2}}\right)^{w_j(t_0)+\frac{\Delta w_j}{2}}. \tag{A.43}$$

And then completing this two-step process

$$R_i^{\text{2-step}} = R_i^{\Delta w/2}\frac{w_i(t_0) + \Delta w_i}{w_i(t_0) + \frac{\Delta w_i}{2}} \prod_{j=1}^{N} \left(\frac{w_j(t_0) + \frac{\Delta w_j}{2}}{w_j(t_0) + \Delta w_j}\right)^{w_j(t_0)+\Delta w_j} \tag{A.44}$$

$$= R_i(t_0)\frac{w_i(t_0) + \Delta w_i}{w_i(t_0)} \prod_{j=1}^{N} \frac{w_j(t_0)^{w_j(t_0)+\frac{\Delta w_j}{2}} \left(w_j(t_0) + \frac{\Delta w_j}{2}\right)^{\frac{\Delta w_j}{2}}}{\left(w_j(t_0) + \Delta w_j\right)^{w_j(t_0)+\Delta w_j}}. \tag{A.45}$$

Dividing $R_i^{\text{2-step}}$ by $R_i^{\text{1-step}}$ we get

$$\frac{R_i^{\text{2-step}}}{R_i^{\text{1-step}}} = \prod_{j=1}^{N} \left(1 + \frac{\Delta w_j}{2 w_j(t_0)}\right)^{\frac{\Delta w_j}{2}}. \tag{A.46}$$

Thus we have that

$$\mathbf{R}^{\text{2-step}} = \mathbf{R}^{\text{1-step}} \prod_{j=1}^{N} \left(1 + \frac{\Delta w_j}{2 w_j(t_0)}\right)^{\frac{\Delta w_j}{2}}, \tag{A.47}$$

as required.

Linear interpolation conclusions The ratio between $R_{\text{2-step}}$ and $R_{\text{1-step}}$, is always greater than or equal to one. This can be seen trivially. When Δw_j is > 0, its term in the product is > 1, as we have a number > 1 raised by a positive power. When Δw_j is < 0 its term in the product is again > 1, as we have a number < 1 raised by a negative power. (And when Δw_j is $= 0$ the term in the product is 1, as we have $1^0 = 1$.)

This means that for any weight change currently being done in one leap, we can produce a cheaper weight update procedure by going through and equilibrating at the linear midway value. And after we apply this 'bisection' once, we can apply the exact same argument again on each half the new procedure, dividing changes in weights in half again. This tells us that we want weight changes to be maximally smooth. For given start and end weights, choosing to linearly interpolate the value of the weights in the pool at block-level resolution is thus a simple and effective (and gas-efficient) way to give us smooth weight changes.

A.4 Interpretation of re-weighting as an auction

Imagine the weights changing every block, but no arbitrageur acts. The arbitrage opportunity increases every block (assuming constant market prices). This means that in effect there is a form of Dutch auction taking place, where the amount a TFMM pool will pay to rebalance increases with blocktime. Arbitrageurs have their own costs (gas, infrastructure costs, etc) that have to be paid for by their trades. In this way, the arbitrage opportunities of TFMM pools result in healthy competition between arbs for who can first fulfil the arbitrage trade, ahead of competitors, by having the leanest, lowest-cost method of arbing.

A.5 Small weight changes

Here we have a very small change in weights,

$$\mathbf{w}(t') = \mathbf{w}(t_0) + \delta\mathbf{w}, \tag{A.48}$$

where the components $\delta w_i \ll 1$.

A pool is 'in equilibrium' if the actual division of value between assets is in accordance with its weight vector $\mathbf{w}$. If that is the case, $\mathbf{w}$ *is* the division of value over assets in a pool. At time t_0, for market prices $p_i(t_0)$ and pool reserves $R_i(t_0)$, for a pool in equilibrium it is the case that

$$w_i(t_0) = \frac{p_i(t_0)R_i(t_0)}{V(t_0)}, \tag{A.49}$$

$$\Rightarrow R_i(t_0) = \frac{w_i(t_0)V(t_0)}{p_i(t_0)}, \tag{A.50}$$

$$\Rightarrow p_i(t_0) = \frac{w_i(t_0)V(t_0)}{R_i(t_0)}, \tag{A.51}$$

$$\Rightarrow V(t_0) = \frac{p_i(t_0)R_i(t_0)}{w_i(t_0)}, \tag{A.52}$$

where $V(t_0) := \sum_{i=1}^{N} p_i(t_0)R_i(t_0)$ is the value of the pool, in the numéraire of the market prices $p_i(t_0)$[5].

A.5.1 TFMM pools, No Fees

Here we derive

$$\mathbf{R}(t_0)\left(1 + \frac{\delta\mathbf{w}}{\mathbf{w}(t_0)}\right) + \mathcal{O}\left(\{(\delta w_i)^2\}\right). \tag{A.53}$$

[5]Of course changes in numéraire do not change w_i or $R_i(t_0)$ as such changes equally scale $p_i(t_0)$ *and* $V(t_0)$ and so cancel out.

Start with Eq (5), rearranging, sub in Eq (A.48) and expand out:

$$\mathbf{R}(t') = \mathbf{R}(t_0)\frac{\mathbf{w}(t_0)+\delta\mathbf{w}}{\mathbf{w}(t_0)}\prod_{i=1}^{N}\left(\frac{w_i(t_0)}{w_i(t')}\right)^{w_i(t')} \tag{A.54}$$

$$= \mathbf{R}(t_0)\frac{\mathbf{w}(t_0)+\delta\mathbf{w}}{\mathbf{w}(t_0)}\prod_{i=1}^{N}\left(\frac{w_i(t_0)}{w_i(t_0)+\delta w_i}\right)^{w_i(t_0)+\delta w_i} \tag{A.55}$$

$$= \mathbf{R}(t_0)\frac{\mathbf{w}(t_0)+\delta\mathbf{w}}{\mathbf{w}(t_0)}\prod_{i=1}^{N}\left(1+\frac{\delta w_i}{w_i(t_0)}\right)^{-w_i(t_0)-\delta w_i} \tag{A.56}$$

$$= \mathbf{R}(t_0)\frac{\mathbf{w}(t_0)+\delta\mathbf{w}}{\mathbf{w}(t_0)}\prod_{i=1}^{N}\left(1+(-w_i(t_0)-\delta w_i)\frac{\delta w_i}{w_i(t_0)}+\text{H.O.T.}\right) \tag{A.57}$$

$$= \mathbf{R}(t_0)\frac{\mathbf{w}(t_0)+\delta\mathbf{w}}{\mathbf{w}(t_0)}\prod_{i=1}^{N}(1-\delta w_i)+\text{H.O.T.} \tag{A.58}$$

$$= \mathbf{R}(t_0)\frac{\mathbf{w}(t_0)+\delta\mathbf{w}}{\mathbf{w}(t_0)}\left(1-\sum_{i=1}^{N}\delta w_i\right)+\text{H.O.T.} \tag{A.59}$$

$$= \mathbf{R}(t_0)\frac{\mathbf{w}(t_0)+\delta\mathbf{w}}{\mathbf{w}(t_0)}+\text{H.O.T.} \tag{A.60}$$

$$= \mathbf{R}(t_0)\left(1+\frac{\delta\mathbf{w}}{\mathbf{w}(t_0)}\right)+\text{H.O.T.} \tag{A.61}$$

Here 'H.O.T.' means higher order terms in the elements of $\delta\mathbf{w}$.

In Eq (A.57) we have used the binomial expansion, which in Eq (A.58) we have approximated to just first-order terms in δw_i (as the components of $\delta\mathbf{w}$ are small).

In going from Eq (A.58) to Eq (A.59) we have kept only first-order terms in δw_i when performing the product in Eq (A.58).

To go from Eq (A.59) to Eq (A.60) we have used that $\sum_{i=1}^{N}\delta w_i = 0$ (weight vectors sums to 1 at all times, so the sum of changes is necessarily zero).

Using Eq (A.61) with Eq (6), we get, for small weight changes,

$$V_{\delta w}(t') = \sum_{i=1}^{N} p_i(t')R_i(t') \tag{A.62}$$

$$= \sum_{i=1}^{N} p_i(t_0)R_i(t') \tag{A.63}$$

$$= \sum_{i=1}^{N} p_i(t_0)R_i(t_0)\left(1+\frac{\delta w_i}{w_i(t_0)}\right)+\text{H.O.T.} \tag{A.64}$$

Using $\sum_{i=1}^{N}\delta w_i = 0$, we can simply this further:

$$
\begin{aligned}
V_{\delta w}(t') &= \sum_{i=1}^{N} p_i(t_0)R_i(t_0)\left(1+\frac{\delta w_i}{w_i(t_0)}\right) + \text{H.O.T.} && \text{(A.65)}\\
&= \sum_{i=1}^{N} p_i(t_0)R_i(t_0) + \sum_{i=1}^{N}\frac{p_i(t_0)R_i(t_0)}{w_i(t_0)}\delta w_i + \text{H.O.T.} && \text{(A.66)}\\
&= \sum_{i=1}^{N} p_i(t_0)R_i(t_0) + \sum_{i=1}^{N} V(t_0)\delta w_i + \text{H.O.T.} && \text{(A.67)}\\
&= \sum_{i=1}^{N} p_i(t_0)R_i(t_0) + V(t_0)\cancelto{0}{\sum_{i=1}^{N}\delta w_i} + \text{H.O.T.} && \text{(A.68)}\\
&= \sum_{i=1}^{N} p_i(t_0)R_i(t_0) + \text{H.O.T.} && \text{(A.69)}\\
&= V(t_0) + \text{H.O.T.}. && \text{(A.70)}
\end{aligned}
$$

where we have used Eq (A.52) to go from Eq (A.66) to Eq (A.67). Again, 'H.O.T.' means higher order terms in the elements of $\delta\mathbf{w}$.

A.5.2 TFMM pools, With Fees

Here we derive Eq (10).

Using the results in Appendix B of Angeris et al. [2020]:

$$
V^{\gamma\text{-fees}} > V^{\text{no-fees}} + (1-\gamma)\sum_{i=1}^{N}\Delta_i p_i, \tag{A.71}
$$

where $\boldsymbol{\Delta}$ is the vector giving the assets being traded into the pool.

Δ_i in Eq (A.71) is the positive-value-only components in the vector giving the change in reserves from the arb-trade that brought the pool back into equilibrium (i.e. quoting market prices under its new weights). Denoting the raw change in reserves $\boldsymbol{\Phi} := \mathbf{R}(t') - \mathbf{R}(t_0)$ then

$$
\Delta_i = \mathbb{I}_{\Phi_i>0}\Phi_i, = \mathbb{I}_{R_i(t')-R_i(t_0)>0}\left(R_i(t') - R_i(t_0)\right),
$$

where $\mathbb{I}_f$ is the indicator function. Eq (A.61) gives us that $R_i(t') - R_i(t_0) \simeq \frac{R_i(t_0)\delta w_i}{w_i(t_0)}$, enabling an approximate[6] lower bound on the value of a TFMM pool after a small change in weight that has led to an arbitrageur rebalancing it to have reserves in line with its new weights:

$$
\begin{aligned}
V_{\delta w}^{\gamma\text{-fees}}(t') &\gtrsim V(t_0) + (1-\gamma)\sum_{i=1}^{N}\left(p_i(t_0)\mathbb{I}_{\frac{R_i(t_0)\delta w_i}{w_i(t_0)}>0}\left(\frac{R_i(t_0)\delta w_i}{w_i(t_0)}\right)\right) && \text{(A.72)}\\
&= V(t_0) + (1-\gamma)\sum_{i=1}^{N}\left(\mathbb{I}_{\delta w_i>0}\left(\frac{p_i(t_0)R_i(t_0)\delta w_i}{w_i(t_0)}\right)\right) && \text{(A.73)}\\
&= V(t_0) + (1-\gamma)\sum_{i=1}^{N}\left(\mathbb{I}_{\delta w_i>0}\left(V(t_0)\delta w_i\right)\right) && \text{(A.74)}\\
\Rightarrow V_{\delta w}^{\gamma\text{-fees}}(t') &\gtrsim V(t_0)\left(1 + (1-\gamma)\sum_{i=1}^{N}\left(\mathbb{I}_{\delta w_i>0}\delta w_i\right)\right), && \text{(A.75)}
\end{aligned}
$$

as required.

[6]Due to our first-order-in-$\delta\mathbf{w}$ truncations.

A.5.3 CEX rebalancing, No Fees

Here we consider active rebalancing, where the pool's assets are rebalanced by going out to market to trade, by being a taker. As above we will assume prices are constant over the course of the rebalancing so $p_i(t_0) = p_i(t')$. Again, we have the weights change a small amount, as in Eq (A.48),

$$\mathbf{w}(t') = \mathbf{w}(t_0) + \delta\mathbf{w}.$$

An ideal weight change would mean that no cost at all is paid to rebalance, so $V(t_0) = V(t')$. That would then mean that after the weights have changed and this perfect rebalance has been done, we have that

$$w_i(t_0) + \delta w_i = \frac{p_i(t')R_i(t')}{V(t_0)}, \tag{A.76}$$

$$\Rightarrow R_i(t') = \frac{(w_i(t_0) + \delta w_i)\, V(t_0)}{p_i(t_0)}, \tag{A.77}$$

$$R_i(t') = \frac{(w_i(t_0) + \delta w_i)\, R(t_0)}{w_i(t_0)}, \tag{A.78}$$

$$R_i(t') = \left(1 + \frac{\delta w_i}{w_i(t_0)}\right) R(t_0), \tag{A.79}$$

where we have made use Eq (A.51) to write the post-update reserves in terms only of initial reserves, initial weights, and weight changes. This gives us exactly Eq (A.53) written in component form, which shouldn't be surprising as we have essentially just reversed the derivations in the above section.

A.5.4 CEX rebalancing, With Fees

Here we derive Eq (11).

A simple model for fees for this rebalance is to charge a fee amount on the outgoing leg of the trade. To keep similarity with how we parameterise fees in DEX trades, we will parameterise the fees as $1 - \gamma_{\text{trad}}$, so $\gamma_{\text{trad}} = 0.999$ would mean fees of 10bps.

As in the section above, we denote the outgoing leg of the trade $\boldsymbol{\Delta}$, with

$$\Delta_i = \mathbb{I}_{R_i(t') - R_i(t_0) > 0}\left(R_i(t') - R_i(t_0)\right) = \mathbb{I}_{\frac{\delta w_i}{w_i(t_0)} R(t_0) > 0}\left(\frac{\delta w_i}{w_i(t_0)} R(t_0)\right),$$

Where we have used Eq (A.79) that tells us $R_i(t') - R_i(t_0) = \frac{\delta w_i}{w_i(t_0)} R(t_0)$ (for TFMM the equals sign was a $\simeq$ due to the small-δw_i first-order approximation, here it is exact). This gives us Eq (11), our counterpart for CEX rebalancing to Eq (10),

$$V_{\text{trad}}^{\gamma\text{-fees}} > V(t_0)\left(1 - (1 - \gamma_{\text{trad}}) \sum_{i=1}^{N} \left(\mathbb{I}_{\delta w_i > 0}\, \delta w_i\right)\right), \tag{A.80}$$

as required.

B Concentrated Liquidity for Multi-token TFMM Pools

Previous projects have introduced the use of 'concentrated liquidity' Adams et al. [2021], Nguyen et al. [2021], where trades are performed 'as if' a pool contains more reserves than it in fact does. If liquidity is *concentrated* by quoting trading using these *virtual reserves*, that are greater than the true reserves, then it can be possible for trades to remove all of the reserves of any token in the pool by a single trade or a sequence of trades. This has the effect of making the reserves of a pool only available within a certain range of market prices. Why is this method desirable? It leads to reduced slippage for traders, as higher reserves in a pool naturally decrease the curvature of commonly-used trading functions. This means that pools with concentrated liquidity are more appealing for traders than those that do not concentrate their liquidity, all else being equal. Previous projects have implemented concentrated liquidity for pools composed of pairs of tokens. *Here we extend the virtual reserves approach to pools with non-uniform weights over tokens and simultaneously to multi-token pools.*

A TFMM pool has, in a given block, the invariant

$$\prod_{i=1}^{N} R_i^{w_i(t)} = k(t), \quad \text{where} \sum_{i=1}^{N} w_i(t) = 1, \text{ and } \forall i \ 0 < w_i(t) < 1$$

We *scale* the reserves uniformly by a factor ν (≥ 1) to get our virtual reserves, $\breve{R}_i$,

$$\breve{R}_i = \nu R_i,$$

and further we require that these new virtual reserves follow a matched virtual invariant function

$$\prod_{i=1}^{N} \breve{R}_i^{w_i(t)} = \breve{k}(t), \quad \text{where} \sum_{i=1}^{N} w_i(t) = 1, \text{ and } \forall i \ 0 < w_i(t) < 1 \tag{B.81}$$

Subbing in $\breve{R}_i = \nu R_i$, we obtain

$$\breve{k}(t) = \nu k(t), \tag{B.82}$$

First let us obtain Eq (B.82):

$$\begin{aligned}
\breve{k}(t) &= \prod_{i=1}^{N} \breve{R}_i^{w_i(t)} && \text{(B.83)}\\
&= \prod_{i=1}^{N} (\nu R_i)^{w_i(t)} && \text{(B.84)}\\
&= \prod_{i=1}^{N} (\nu)^{w_i(t)} \prod_{i=1}^{N} \left(R_i^{w_i(t)}\right) && \text{(B.85)}\\
&= \nu^{\sum_{i=1}^{N} w_i(t)} \prod_{i=1}^{N} R_i^{w_i(t)} && \text{(B.86)}\\
&= \nu \prod_{i=1}^{N} R_i^{w_i(t)} && \text{(B.87)}\\
&= \nu k(t) && \text{(B.88)}
\end{aligned}$$

What about after a trade has been done? Using primes to denote post-trade quantities,

$$\begin{aligned}
R_i' &= R_i + \Delta_i - \Lambda_i && \text{(B.89)}\\
\breve{R}_i' &= \nu R_i + \Delta_i - \Lambda_i && \text{(B.90)}\\
\Rightarrow \breve{R}_i' &= R_i' + (\nu - 1) R_i. && \text{(B.91)}
\end{aligned}$$

This enables us to write the post-trade invariant of the virtual reserves, Eq (B.81), in terms of the true reserves

$$\prod_{i=1}^{N} \breve{R}_i'^{w_i(t)} = \prod_{i=1}^{N} \left(R_i' + (\nu - 1)R_i\right)^{w_i(t)} = \nu k'(t) \tag{B.92}$$

which shows us clearly that the virtual invariant is a scaled and shifted version of the original invariant.

NB: For the rest of this section of the appendix we are going to drop explicit time dependence for clarity.

B.1 Minimum and Maximum Reserves

Say that we wish bound out post-trade underlying reserves, R_i'. What are the corresponding bounds on the virtual, scaled reserves, $\breve{R}_i$?

Minimum $\breve{R}_i'$ We bound the post-trade underlying reserves, R_i', to be $(1-\delta)R_i$ for $0 < \delta < 1$.

$$\breve{R}_{i,\min}'(\delta) = (1-\delta)R_i + (\nu - 1)R_i = (\nu - \delta)R_i.$$

So the absolute minimum is when the underlying reserves go to zero, so $\breve{R}_{i,\min}'(\delta = 1) = (\nu - 1)R_i$.

Maximum $\breve{R}_i'$ If all tokens but one have their virtual post-trade reserves reduced to $\breve{R}_{i,\min}' = (\nu - 1)R_i$, what are the virtual reserves of the one, maximised token? For the no-fees case, whatever trades took us to this situation must have, at minimum, preserved the invariant of the virtual pool, so

$$\nu k = \prod_{i=1}^{N} \breve{R}_i'^{w_i} = \prod_{\substack{i=1 \\ i \neq j}}^{N} \breve{R}_{i,\min}'^{w_i} \breve{R}_{j,\max}'^{w_j} = \prod_{\substack{i=1 \\ i \neq j}}^{N} (\nu - \delta)^{w_i} \prod_{\substack{i=1 \\ i \neq j}}^{N} R_i'^{w_i} \breve{R}_{j,\max}'^{w_j} \tag{B.93}$$

$$\Rightarrow \nu k = (\nu - \delta)'^{1-w_j} \frac{k}{R_j^{w_j}} \breve{R}_{j,\max}'^{w_j} \tag{B.94}$$

$$\Rightarrow \breve{R}_{j,\max}' = R_j \nu^{1/w_j} (\nu - \delta)^{1-1/w_j}, \tag{B.95}$$

as required. (With fees present, this simply becomes $\breve{R}_{j,\max}'(\gamma) = R_j \nu^{1/w_j} (\nu - \gamma\delta)^{1-1/w_j}$, assuming that this change was done in one batch trade.)

B.2 Interpreting the Transform

At this stage, it is worth reminding ourselves of the geometric construction behind this form of concentration. Recall that, speaking informally, the slippage in a trade is a function of the curvature of the underlying trading function. So to improve the experience of traders, we can aim to find a 'virtual' pool which has less curvature than the original, and expose that new virtual pool's trading function to traders to interact with. We want to find an invariant function of the same geometric form as the original, but scaled and translated in such a way that curvature is minimised.

In theory this gives us $N + 1$ degrees of freedom, one for each translation in a token direction and an overall scaling. But these movements of the trading function have the potential to change the quoted prices of the pool. If trading using the virtual trading function is done using prices that are not consistent, the system will create enormous and costly arbitrage opportunities. That is, the gradient of the virtual trading function with respect to virtual token reserves, evaluated at the virtual reserves that the true reserves are mapped to, has to match the gradients of the original trading function evaluated at the true reserves. These requirements takes up N degrees of freedom, meaning that we can parameterise the set of possible virtual reserves by a single parameter, ν. Let us now discuss this in more mathematical depth.

Why Linear, Uniform, Scaling of Reserves? Consider a set of N monotonic, positive definite functions $\{f_i(\cdot)\}, t \in 1, ..., N$, which we use to make to our virtual reserves:

$$\breve{R}_i = f_i(R_i).$$

The virtual pool has a constant $\breve{k} = \prod_{i=1}^N f_i(R_i)^{w_i}$. A trade now happens, Δ_i being deposited and Λ_i being withdrawn, that maintains (or increases) the virtual pool's constant. It is still the case that 1) the actual trade amounts, Δ_i, Λ_i must be added and subtracted the true reserves, as that is what it means to trade with a pool and 2) the actual trade amounts, Δ_i, Λ_i must be added and subtracted from the virtual reserves in the virtual invariant, because again that is what it is to run a pool. (Note that we handle the zero-fees case here, for simplicity, but the argument made here is not fundamentally changed by the presence of fees).

This means we can write down the generalised versions of Eqs (B.89-B.91):

$$R_i' = R_i + \Delta_i - \Lambda_i \tag{B.96}$$

$$\breve{R}_i' = f_i(R_i) + \Delta_i - \Lambda_i \tag{B.97}$$

$$\Rightarrow \breve{R}_i' = R_i' + (f_i(R_i) - 1)R_i. \tag{B.98}$$

This enables us to write the generalised version of Eq (B.92):

$$\prod_{i=1}^N \breve{R}_i'^{w_i} = \prod_{i=1}^N (R_i' + (f_i(R_i) - 1)R_i)^{w_i} = \breve{k}' \tag{B.99}$$

Because of this linearity we can apply the underlying logic of Eq (A.14) to our virtual pool invariant, that the marginal prices quoted by a pool for one token in terms of another is the gradient of the derivatives of the trading function w.r.t. each token's post-trade reserves, evaluated at the current reserves. Applying that here results in

$$p_{i,j}^{\text{virtual}} = \left.\frac{\left(\frac{\partial}{\partial \mathbf{R}'}\breve{k}'\right)_i}{\left(\frac{\partial}{\partial \mathbf{R}'}\breve{k}'\right)_j}\right|_{\mathbf{R}'=\mathbf{R}} \tag{B.100}$$

$$= \left.\frac{w_i}{w_j}\frac{(R_j' + (f_j(R_j) - 1)R_j)}{(R_i' + (f_i(R_i) - 1)R_i)}\right|_{\mathbf{R}'=\mathbf{R}} \tag{B.101}$$

$$= \frac{w_i}{w_j}\frac{f_j(R_j)}{f_i(R_i)}. \tag{B.102}$$

We need the prices quoted by our virtual invariant to match the prices we obtain from the original invariant, Eq (A.14), otherwise allowing trades under the virtual invariant will not lead to the pool have the desired division of value between assets at equilibrium—i.e. these two sets of prices have to match for $\mathbf{w}$ to describe the division of pool value between token holdings. So we require that

$$\frac{f_j(R_j)}{f_i(R_i)} = \frac{R_j}{R_i},$$

which is satisfied for $f_k(x) = f(x) = \nu x$ for some constant ν, as required.

B.3 What should ν be set to?

Pool creators can choose a particular value based on their analysis, analogous to how update rules' parameters are set; but also, if they prefer, they can choose an external smart contract to be called at the same time as the weight update call to calculate a dynamic $\nu(t)$ that depends on market conditions, which could include external oracle calls.

C Study of Potential Multiblock Attack

An important question about TFMMs is whether their temporal dependence means there is an opportunity for MEV. Here we study such attacks, where an attacker aims to manipulate a TFMM pool shortly before a weight update. This situation is the TFMM equivalent of a 'sandwich attack', but with the attack being done against pool LPs rather than a trader.

First the attacker trades against the pool to take it out of equilibrium, then the weights change for the next block, and the attacker does a trade in the reverse direction – if effect this is a multi-block sandwich attack, where the change of weights between blocks is roughly analogous to a trade. Like trades in vanilla sandwich attacks, certain criteria have to be met for this attack to be profitable/possible.

We perform mathematical analysis of the circumstances where attacks of this kind are possible and where they are impossible, *even when granting the attacker (for free and with certainty) the final transaction in one block and the first transaction in the next.*

We will derive the cost of manipulating a TFMM pool, with particular weights, such that it quotes a chosen price, rather than the true, reference, *market price*. We do this in the presence of fees. After having manipulated the price in this way, the weights of the pool change, and a 'boosted' arbitrage trade is then available in the following block. We derive the returns to an arbitrageur from bringing the manipulated-pool back to quoting market prices after the weights have been updated.

From these quantities we can then calculate the overall return on this three-step process, the cost of manipulating the quoted prices of a pool, the weights changing, and the return of performing an arbitrage trade against the pool's new weights

C.1 Trading & Fees

Consider an N-token pool. For simplicity of analysis we trade a subset of 2 tokens for this attack—in our numerical work later we allow all tokens to be traded and empirically we find that optimal attacks are done by trading between a pair of tokens in this way. Those two tokens have weights $\mathbf{w} = \{w_1, w_2\}$ and reserves $\mathbf{R} = \{R_1, R_2\}$. We are here going to consider the quoted prices of the tokens in the pool.

We assume the existence of true, external market prices for the tokens. $m_{p,2} = m_p$ is the market price of the second token in terms of the first. $m_{p,1}$, the market price of the first token in terms of the second is simply the reciprocal of this: $m_{p,1} = 1/m_{p,2} = 1/m_p$. (Without loss of generality, we will be using the first token as the numéraire throughout unless otherwise noted.[7])

Here we consider an attacker who wishes the pools quoted price for the second token, in terms of the first, to deviate from the market prices by a factor of $(1+\epsilon)$. *Without loss of generality we will consider attacks where the initial price manipulation is done to increase the quoted price of the second token.* The attacker trades the first token into the pool and withdraws the second tokens.

This is without loss of generality as *a)* we can freely permute the indices of assets in a pool's basket and as *b)* a manipulation to increase the price of one token in the pool, by reducing its reserves, necessarily means we are decreasing the price of the other token. So an attack based around *decreasing* the price of token, rather than increasing it, is obtained simply by exchanging the indices of the tokens.

[7]Either you can imagine that the first token in the pool is a stablecoin in USD, say. Or it can be an arbitrary token, and we then are applying a linear scaling (that token's price) to our vanilla market prices to get them in the denomination of our particular chosen numéraire.

C.2 Mathematical Analysis of Potential Attack

With fees of $\tau = 1 - \gamma$ (commonly $\gamma = 0.997$), when trading in $\Delta_1 > 0$ of token 1 to receive $\Delta_2 > 0$ of token 2 the trade must fulfill

$$(R_1 + \gamma\Delta_1)^{w_1}(R_2 - \Delta_2)^{w_2} \geq k = R_1^{w_1} R_2^{w_2}, \tag{C.103}$$

with the best deal for the trader found when

$$(R_1 + \gamma\Delta_1)^{w_1}(R_2 - \Delta_2)^{w_2} = k = R_1^{w_1} R_2^{w_2}. \tag{C.104}$$

Eq (C.104) can be rearranged into a neater form,

$$1 - \frac{\Delta_2}{R_2} = \left(1 + \gamma\frac{\Delta_1}{R_1}\right)^{-\frac{w_1}{w_2}}. \tag{C.105}$$

In this attack, the attacker first manipulates the pool's quoted price for second token to be

$$m_{\mathrm{AMM},2}^{\mathrm{manip.}} = (1+\epsilon) m_p, \tag{C.106}$$

where $\epsilon \geq \epsilon_0$. ϵ_0 is the 'do nothing' or 'NULL' value—equivalent to their being no attack carried out. When fees are present and we are in a worst-case scenario for the pool, $\epsilon_0 > 0$.

$C(\epsilon)$ denotes the cost to the attacker of performing the first, price-manipulating, trade, making clear the dependence of this cost on the scale of the price deviation the attacker does. $X(\epsilon)$ denotes the return to the attacker from the post-weight-update arbitrage trade. The overall benefit to the attacker over not carrying out the attack and just being an arbitrageur, $Z(\epsilon)$, is thus

$$Z(\epsilon) = X(\epsilon) - C(\epsilon) - X(\epsilon_0), \tag{C.107}$$

When $Z(\epsilon) > 0$ the return from the attack, $X(\epsilon) - C(\epsilon)$, is greater than the vanilla, just-doing-arbitrage return $X(\epsilon_0)$. We can obtain bounds on $X(\epsilon)$ and $C(\epsilon)$ when fees are present without having them in closed form.

C.2.1 Cost of Manipulating Quoted Prices

Post-trade, the quoted prices are

$$m_{\mathrm{attacker}} = m_p(1+\epsilon) = \frac{1}{\gamma}\frac{\frac{w_2}{R_2-\Delta_2}}{\frac{w_1}{R_1+\Delta_1}}. \tag{C.108}$$

Subbing in that $m_p = \gamma m_u$, we find that after the attack trade

$$\frac{R_1 + \Delta_1}{R_2 - \Delta_2} = \gamma^2(1+\epsilon)\frac{R_1}{R_2}. \tag{C.109}$$

Combining Eq (C.109) with Eq (C.104) and rearranging we have that

$$\left(1 + \frac{\Delta_1}{R_1}\right)\left(1 + \gamma\frac{\Delta_1}{R_1}\right)^{\frac{w_1}{w_2}} = \gamma^2\left(1+\epsilon\right) \tag{C.110}$$

Similar manipulations give us

$$\left(1 - \frac{\Delta_2}{R_2}\right)^{-1}\left(1 + \frac{1}{\gamma}\left(\left(1 - \frac{\Delta_2}{R_2}\right)^{-\frac{w_2}{w_1}} - 1\right)\right) = \gamma^2\left(1+\epsilon\right). \tag{C.111}$$

$C(\epsilon)$, the *cost* of the attack to the manipulator, again using token 1 as the numéraire, is

$$C(\epsilon) = \Delta_1 - m_p\Delta_2. \tag{C.112}$$

Eq (C.110) links together Δ_1, R_1, γ and ϵ, and Eq (C.111) separately links Δ_2, R_2, γ and ϵ. These are *implicit* equations for Δ_1 or Δ_2 in terms of the other variables.

This means we cannot trivially write down $C(\epsilon)$ in closed form. We can make progress as we need only either the ratio of Δ_1 to Δ_2 or the *partial derivatives* of the cost, of Δ_1, and of Δ_2, with respect to ϵ for us to find a bound on $Z(\epsilon)$.

C.2.2 Change of Reserves of an Attacked Pool After a Weight Update

We assume that all this takes place fast enough that the market prices are constant–this could all take place in one block. After the price manipulation above we have new ($'$ed) reserves

$$R_1{}' = R_1 + \Delta_1 \tag{C.113}$$
$$R_2{}' = R_2 - \Delta_2 \tag{C.114}$$

The weights of the pool change, so now we have new (again, $'$ed) weights $w_1' = w_1 + \Delta w_1$ and $w_2' = w_2 + \Delta w_2$.

After the arbitrage trade, the reserves of the pool will change again (to $''$ed values) such that the new weights, w_1', w_2', and the new post-arbitrage-trade reserves, $R_1{}'', R_2{}''$, minimise the value in the pool (thus maximising the returns to the arb). We thus have

$$R_1{}'' = R_1{}' - \Delta_1' \tag{C.115}$$
$$R_2{}'' = R_2 + \Delta_2'. \tag{C.116}$$

The return to the arbitrageur is

$$X(\epsilon) = \Delta_1' - m_p \Delta_2' \tag{C.117}$$

What is the best-for-the-arb trade? Instead of directly obtaining the value of $X(\epsilon)$ we will upper bound its value by $X_{\gamma=1}(\epsilon)$ (the return to the arbitrageur when the arbitrageur's trade takes place in a no-fees way—$\gamma = 1$ for this trade). Intuitively $X_{\gamma=1}(\epsilon) > X(\epsilon)$ (after all it would be surprising if fees made the trade cheaper), and in Appendix C.3 we prove that this is indeed the case.

And thus that

$$m'_{\text{AMM}} = m'_u = \frac{\frac{w_2'}{R_2{}''}}{\frac{w_1'}{R_1{}''}} = m_p, \tag{C.118}$$

Combining Eqs (C.109) and (C.118) we get

$$\frac{\frac{w_2'}{R_2{}''}}{\frac{w_1'}{R_1{}''}} = \frac{1}{\gamma}\frac{1}{1+\epsilon}\frac{\frac{w_2}{R_2{}'}}{\frac{w_1}{R_1{}'}} \tag{C.119}$$
$$\Rightarrow \frac{R_2{}''}{R_2{}'} = \gamma(1+\epsilon)\frac{w_2'}{w_2}\frac{w_1}{w_1'}\frac{R_1{}''}{R_1{}'}. \tag{C.120}$$

Thus we can consider the no-fees invariant before and after this arb-trade:

$$\tilde{k}' = R_1{}'^{w_1'} R_2{}'^{w_2'} = R_1{}''^{w_1'} R_2{}''^{w_2'}$$
$$\Rightarrow 1 = \left(\frac{R_1{}''}{R_1{}'}\right)^{w_1'} \left(\frac{R_2{}''}{R_2{}'}\right)^{w_2'}. \tag{C.121}$$

Using Eq (C.120) we get

$$1 = \left(\frac{R_1{}''}{R_1{}'}\right)^{w_1'} \left(\gamma^2(1+\epsilon)\frac{w_2'}{w_2}\frac{w_1}{w_1'}\frac{R_1{}''}{R_1{}'}\right)^{w_2'},$$
$$\Rightarrow \frac{R_1{}''}{R_1{}'} = \left(\frac{w_2}{w_2'}\frac{w_1'}{w_1}\frac{1}{\gamma}\frac{1}{1+\epsilon}\right)^{\frac{w_2'}{w_1'+w_2'}}, \tag{C.122}$$

and thus, similarly, that

$$\frac{R_2{}''}{R_2{}'} = \left(\frac{w_2}{w_2'}\frac{w_1'}{w_1}\frac{1}{\gamma}\frac{1}{1+\epsilon}\right)^{\frac{-w_1'}{w_1'+w_2'}}. \tag{C.123}$$

From algebraic manipulation we obtain that

$$\Delta_1' = R_1{}' \left[1 - \left(\frac{w_2}{w_2'} \frac{w_1'}{w_1} \frac{1}{\gamma} \frac{1}{1+\epsilon} \right)^{\frac{w_2'}{w_1'+w_2'}} \right], \tag{C.124}$$

$$\Delta_2' = R_2{}' \left[\left(\frac{w_2}{w_2'} \frac{w_1'}{w_1} \frac{1}{\gamma} \frac{1}{1+\epsilon} \right)^{\frac{-w_1'}{w_1'+w_2'}} - 1 \right], \tag{C.125}$$

Our upper bound on the return to the arbitrageur is thus

$$\begin{aligned} X_{\gamma=1}(\epsilon) =& \Delta_1' - m_p \Delta_2' \\ \Rightarrow X_{\gamma=1}(\epsilon) =& R_1{}' \left[1 - \left(\frac{w_2}{w_2'} \frac{w_1'}{w_1} \frac{1}{\gamma} \frac{1}{1+\epsilon} \right)^{\frac{w_2'}{w_1'+w_2'}} \right] - m_p R_2{}' \left[\left(\frac{w_2}{w_2'} \frac{w_1'}{w_1} \frac{1}{\gamma} \frac{1}{1+\epsilon} \right)^{\frac{-w_1'}{w_1'+w_2'}} - 1 \right]. \end{aligned}$$

C.2.3 Putting it all together: When is there no extractable value?

Our upper bound on $Z(\epsilon)$ is thus:

$$Z(\epsilon) \leq \tilde{Z}(\epsilon) = X_{\gamma=1}(\epsilon) - C(\epsilon) - X_{\gamma=1}(\epsilon_0). \tag{C.126}$$

C.2.4 Bounding via gradients of $Z(\epsilon)$

Recall that there is a 'NULL' value of ϵ, ϵ_0, which corresponds to no-price-manipulation. As $\Delta_1(\epsilon_0) = \Delta_2(\epsilon_0) = 0$, $Z(\epsilon_0) = \tilde{Z}(\epsilon_0) = 0$.

We want to find settings of pool parameters such that $Z(\epsilon) < 0$ for all $\epsilon > \epsilon_0$. If $\frac{\partial \tilde{Z}(\epsilon)}{\partial \epsilon} < 0$ for all $\epsilon > \epsilon_0$ (ie if $Z(\epsilon)$ is a monotonically non-increasing function for $\epsilon > \epsilon_0$) then $Z(\epsilon) < 0$ for all $\epsilon > \epsilon_0$. In the zero fees case the 'NULL' value $\epsilon_0 = 0$.

Taking partial derivatives of $\tilde{Z}(\epsilon)$ w.r.t. ϵ we get

$$\frac{\partial \tilde{Z}(\epsilon)}{\partial \epsilon} = \frac{\partial}{\partial \epsilon} \left(\Delta_1' - \Delta_1 \right) + m_p \frac{\partial}{\partial \epsilon} \left(\Delta_2 - \Delta_2' \right), \tag{C.127}$$

so if $\frac{\partial}{\partial \epsilon} \left(\Delta_1' - \Delta_1 \right) \leq 0$ and $\frac{\partial}{\partial \epsilon} \left(\Delta_2 - \Delta_2' \right) \leq 0$, then we can guarantee that the attack will not work.

Gradient of $\Delta_1' - \Delta_1$ w.r.t. ϵ Using Eq (C.124), recalling that $R_1{}' = R_1 + \Delta_1$, we have that

$$\Delta_1' - \Delta_1 = R_1 - \left(\frac{w_2}{w_2'} \frac{w_1'}{w_1} \frac{1}{\gamma} \frac{1}{1+\epsilon} \right)^{\frac{w_2'}{w_1'+w_2'}} (\Delta_1 + R_1) \tag{C.128}$$

$$\Rightarrow \frac{\partial}{\partial \epsilon} \left(\Delta_1' - \Delta_1 \right) = \left(\frac{w_2}{w_2'} \frac{w_1'}{w_1} \frac{1}{\gamma} \frac{1}{1+\epsilon} \right)^{\frac{w_2'}{w_1'+w_2'}} \left(\frac{1}{1+\epsilon} \frac{w_2'}{w_1'+w_2'} (\Delta_1 + R_1) - \frac{\partial \Delta_1}{\partial \epsilon} \right). \tag{C.129}$$

We find that (see Appendix C.4.1)

$$\frac{\partial \Delta_1}{\partial \epsilon} = \frac{\gamma^2 R_1}{\left(1 + \gamma \frac{w_1}{w_2} \left(1 + \frac{\Delta_1}{R_1} \right) \left(1 + \gamma \frac{\Delta_1}{R_1} \right)^{-1} \right) \left(1 + \gamma \frac{\Delta_1}{R_1} \right)^{\frac{w_1}{w_2}}}. \tag{C.130}$$

Subbing this into Eq (C.129), and using Eq (C.110), for $\frac{\partial}{\partial \epsilon} \left(\Delta_1' - \Delta_1 \right) \leq 0$ it must be that

$$\frac{w_2'}{w_1' + w_2'} \left(1 + \gamma \frac{w_1}{w_2} \left(1 + \frac{\Delta_1}{R_1} \right) \left(1 + \gamma \frac{\Delta_1}{R_1} \right)^{-1} \right) \leq 1, \tag{C.131}$$

where we have used that $\left(\frac{w_2}{w_2'} \frac{w_1'}{w_1} \frac{1}{\gamma} \frac{1}{1+\epsilon} \right)^{w_2'} > 0$ when $w_1 \in (0,1)$, $w_2 \in (0,1)$, $w_1' \in (0,1)$, $w_2' \in (0,1)$, $0 < \gamma < 1$, and $\epsilon > \epsilon_0$.

Gradient of $\Delta_2 - \Delta_2'$ w.r.t. ϵ Using Eq (C.125), recalling that ${R_2}' = R_2 - \Delta_2$, we have that

$$\Delta_2 - \Delta_2' = R_2 + \left(\frac{w_2}{w_2'}\frac{w_1'}{w_1}\frac{1}{\gamma}\frac{1}{1+\epsilon}\right)^{-w_1'}(\Delta_2 - R_2) \tag{C.132}$$

$$\Rightarrow \frac{\partial}{\partial\epsilon}(\Delta_2 - \Delta_2') = \left(\frac{w_2}{w_2'}\frac{w_1'}{w_1}\frac{1}{\gamma}\frac{1}{1+\epsilon}\right)^{\frac{-w_1'}{w_1'+w_2'}}\left(\frac{1}{1+\epsilon}\frac{w_1'}{w_1'+w_2'}(\Delta_2 - R_2) + \frac{\partial\Delta_2}{\partial\epsilon}\right). \tag{C.133}$$

We find that (see Appendix C.4.2)

$$\frac{\partial\Delta_2}{\partial\epsilon} = \frac{\gamma^3 R_2\left(1-\frac{\Delta_2}{R_2}\right)^2}{\left(1+\frac{w_2}{w_1}\right)\left(1-\frac{\Delta_2}{R_2}\right)^{-\frac{w_2}{w_1}} - (1-\gamma)} \tag{C.134}$$

Subbing this into Eq (C.133), and using Eq (C.111), for $\frac{\partial}{\partial\epsilon}(\Delta_2 - \Delta_2') \leq 0$ it must be that

$$\frac{w_1'}{w_1'+w_2'} \geq \frac{1-(1-\gamma)\left(1-\frac{\Delta_2}{R_2}\right)^{\frac{w_2}{w_1}}}{1+\frac{w_2}{w_1}-(1-\gamma)\left(1-\frac{\Delta_2}{R_2}\right)^{\frac{w_2}{w_1}}}, \tag{C.135}$$

where we have used that $\left(\frac{w_2}{w_2'}\frac{w_1'}{w_1}\frac{1}{\gamma}\frac{1}{1+\epsilon}\right)^{-w_1'} > 0$ when $w_1 \in (0,1)$, $w_2 \in (0,1)$, $w_1' \in (0,1)$, $w_2' \in (0,1)$, $0 < \gamma < 1$, and $\epsilon > \epsilon_0$.

These results tell us that if the changes in weights are within these bounds, then no attack is possible.

2-token case For the case of a two-token pool, so the two tokens being traded are the old tokens present, we can simplify the above equations and plot them. We are interested in knowing what weight changes we can 'get away with' without a pool being open to attack. That means we are most interested in the above inequalities reformulated explicitly to give us bounds on the weight changes. As we are now in the two-token case, $w_1' + w_2' = w_1 + w_2 = 1$ and $\Delta w_1 = -\Delta w_2$. Thus we can re-write Eq (C.131) and Eq (C.135) in terms of just $w = w_1 = 1 - w_2$ and $\Delta w = \Delta w_1 = -\Delta w_2$:

$$\Delta w \geq 1 - \frac{1}{\left(1+\gamma\frac{w}{(1-w)}\left(1+\frac{\Delta_1}{R_1}\right)\left(1+\gamma\frac{\Delta_1}{R_1}\right)^{-1}\right)} - w, \tag{C.136}$$

$$\Delta w \geq \frac{1-(1-\gamma)\left(1-\frac{\Delta_2}{R_2}\right)^{\frac{(1-w)}{w}}}{1+\frac{(1-w)}{w}-(1-\gamma)\left(1-\frac{\Delta_2}{R_2}\right)^{\frac{(1-w)}{w}}} - w. \tag{C.137}$$

In §C.2 we described how we could study the case where the attacker initially increases the quoted price of the second token, denominated in the first, *without loss of generality* as we can simply swap indices in our final equations to get the results for the mirror-attack where the price of the first token is initially pumped up by the attacker. We can now do that swapping, giving the additional constraints:

$$\Delta w \leq \frac{1}{\left(1+\gamma\frac{(1-w)}{w}\left(1+\frac{\Delta_2}{R_2}\right)\left(1+\gamma\frac{\Delta_2}{R_2}\right)^{-1}\right)} - w, \tag{C.138}$$

$$\Delta w \leq 1 - \frac{1-(1-\gamma)\left(1-\frac{\Delta_1}{R_1}\right)^{\frac{w}{(1-w)}}}{1+\frac{w}{(1-w)}-(1-\gamma)\left(1-\frac{\Delta_1}{R_1}\right)^{\frac{w}{(1-w)}}} - w. \tag{C.139}$$

If the changes in weight fulfil Eqs (C.136-C.139) then no attack is possible for a given initial price-deviating trade. These bounds are expressed as functions of that initial trade, Δ_1 and Δ_2. These bounds are also monotonic in Δ_1 and Δ_2 such that if a pool is safe under an initial trade $(\bar{\Delta}_1, \bar{\Delta}_2)$ then it is safe also for all trades $\Delta_1 < \bar{\Delta}_1$ and $\Delta_2 < \bar{\Delta}_2$. See Appendix C.5 for this.

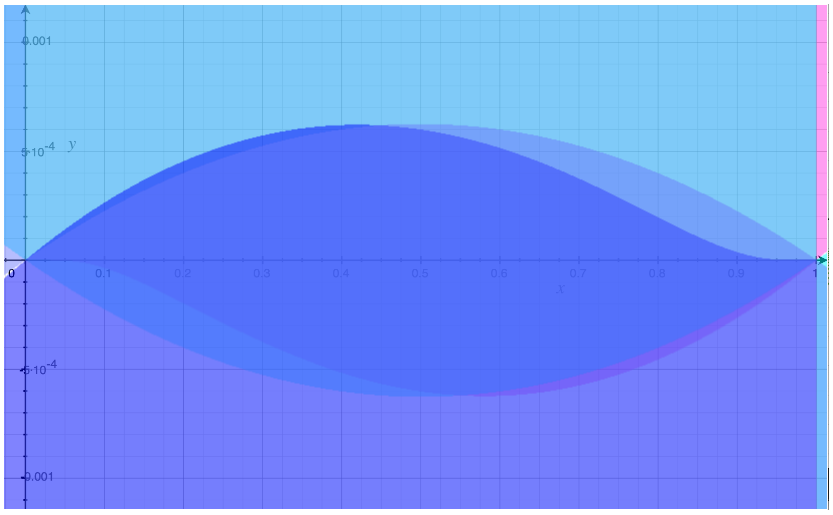


Figure C.3: Plots of the two-token inequalities Eqs (C.136-C.139), where shaded regions are the safe region of each inequality. This is done for $\Delta_1 = 0.2R_1$ and $\gamma = 0.997$. The x-axis is w and the y-axis is Δw.

C.3 Proof that $X_{\gamma=1}(\epsilon)$ is an upper bound on $X(\epsilon)$

Here we will demonstrate that $X_{\gamma=1}(\epsilon) > X(\epsilon)$ for $\epsilon > \epsilon_0$. First recall that

$$X(\epsilon) = \Delta_1'(\epsilon,\gamma) - m_p\Delta_2'(\epsilon,\gamma)$$

and that

$$X_{\gamma=1}(\epsilon) = \Delta_1'(\epsilon,\gamma=1) - m_p\Delta_2'(\epsilon,\gamma=1),$$

where we have made explicit the dependence of Δ_1' and Δ_2' on ϵ and γ.

$$X_{\gamma=1}(\epsilon) - X(\epsilon) = \Delta_1'(\epsilon,\gamma=1) - \Delta_1'(\epsilon,\gamma) - \left(m_p\Delta_2'(\epsilon,\gamma=1) - \Delta_2'(\epsilon,\gamma)\right),$$

This means for $X_{\gamma=1}(\epsilon) > X(\epsilon)$ it is sufficient to show that $\Delta_2'(\epsilon,\gamma) > \Delta_2'(\epsilon,\gamma=1)$ and that $\Delta_1'(\epsilon,\gamma) < \Delta_1'(\epsilon,\gamma=1)$. We will handle these in turn

Showing $\Delta_2'(\epsilon,\gamma) > \Delta_2'(\epsilon,\gamma=1)$: We begin by writing down the trade-invariant for Δ_1' and Δ_2' in the presence of fees. It is, naturally, that

$$({R_1}' - \Delta_1'(\epsilon,\gamma))^{w_1'}({R_2}' + \gamma\Delta_2'(\epsilon,\gamma))^{w_2'} = k' = {R_1}'^{w_1'}{R_2}'^{w_2}, \tag{C.140}$$

as the trader is putting Δ_2' of token 2 into the pool and withdrawing Δ_1' of token 1.

Next, we need the trade invariant if $\gamma = 1$—if there are no fees. Then

$$({R_1}' - \Delta_1'(\epsilon,\gamma=1))^{w_1'}({R_2}' + \Delta_2'(\epsilon,\gamma=1))^{w_2'} = k' = {R_1}'^{w_1'}{R_2}'^{w_2}. \tag{C.141}$$

The final part we need is the following. We know that the purpose of this trade is to get the pool to quote a certain price after the trade has been performed, based on its post-trade pool reserves. This means that the quoted prices after the trade-with-fees will, indeed must, be the same as the quoted price after the trade-with-no-fees[8]. Thus, using Eq (C.118) we get that

$$\frac{({R_1}' - \Delta_1'(\epsilon,\gamma=1))}{({R_2}' + \Delta_2'(\epsilon,\gamma=1))} = \frac{({R_1}' - \Delta_1'(\epsilon,\gamma))}{({R_2}' + \Delta_2'(\epsilon,\gamma))}, \tag{C.142}$$

[8]Note that we are only setting $\gamma = 1$ *for the trade* in the $\gamma = 1$ part of this construction. We are still having the arbitrage trade, with fees or not, bring the quoted price to the upper end of the (γ defined) no-arb region.

where the weights have cancelled out. Rearranging we get that

$$({R_1}' - \Delta_1'(\epsilon,\gamma)) = ({R_2}' + \Delta_2'(\epsilon,\gamma))\frac{({R_1}' - \Delta_1'(\epsilon,\gamma=1))}{({R_2}' + \Delta_2'(\epsilon,\gamma=1))},$$

which we can sub in to Eq (C.140) to get

$$\left(({R_2}' + \Delta_2'(\epsilon,\gamma))\frac{({R_1}' - \Delta_1'(\epsilon,\gamma=1))}{({R_2}' + \Delta_2'(\epsilon,\gamma=1))}\right)^{w_1'}({R_2}' + \gamma\Delta_2'(\epsilon,\gamma))^{w_2'} = k'.$$

Rearranging Eq (C.141) we get

$$({R_1}' - \Delta_1'(\epsilon,\gamma=1))^{w_1'} = \frac{k'}{({R_2}' + \Delta_2'(\epsilon,\gamma=1))^{w_2'}},$$

which we can then sub in to the previous equation to obtain

$$({R_2}' + \Delta_2'(\epsilon,\gamma))^{w_1'}({R_2}' + \gamma\Delta_2'(\epsilon,\gamma))^{w_2'} = ({R_2}' + \Delta_2'(\epsilon,\gamma=1)).$$

As $w_1' + w_2' = 1$, $0 < w_1' < 1$, $0 < w_2' < 1$ and $0 < \gamma < 1$, it is thus clear that $\Delta_2'(\epsilon,\gamma) > \Delta_2'(\epsilon,\gamma=1)$.

Showing $\Delta_1'(\epsilon,\gamma) < \Delta_1'(\epsilon,\gamma=1)$: Rearranging Eq (C.142) we get

$$\begin{aligned}\Delta_1'(\epsilon,\gamma=1) &= {R_1}' - ({R_1}' - \Delta_1'(\epsilon,\gamma))\frac{({R_2}' + \Delta_2'(\epsilon,\gamma=1))}{({R_2}' + \Delta_2'(\epsilon,\gamma))},\\ \Rightarrow \Delta_2'(\epsilon,\gamma=1) &= {R_1}'(1-k) + k\Delta_1'(\epsilon,\gamma),\\ \Rightarrow \Delta_2'(\epsilon,\gamma) &= \frac{1}{k}\Delta_1'(\epsilon,\gamma=1) + {R_1}'\frac{(k-1)}{k},\end{aligned}$$

where $k = \frac{({R_2}'+\Delta_2'(\epsilon,\gamma=1))}{({R_2}'+\Delta_2'(\epsilon,\gamma))}$. $k < 1$ as $\Delta_2'(\epsilon,\gamma) > \Delta_2'(\epsilon,\gamma=1)$ and $k > 0$ as both its numerator and denominator are > 0. This last equation, for $\Delta_1'(\epsilon,\gamma)$ as a function of $\Delta_1'(\epsilon,\gamma=1)$, is a straight line with gradient $1/k > 1$ and intercept ${R_1}'\frac{(k-1)}{k} < 0$. This line crosses '$y = x$' when ${R_1}' = \Delta_1'(\epsilon,\gamma=1)$, so for all $\Delta_1'(\epsilon,\gamma=1) < {R_1}'$ (which are the only possible values as the pool cannot be drained) we have that $\Delta_1(\epsilon,\gamma=1) < \Delta_1'(\epsilon,\gamma=1)$ as required.

C.4 Finding partial derivatives

C.4.1 Finding $\partial\Delta_1/\partial\epsilon$

Recall the implicit equation that defines Δ_1, Eq (C.110):

$$\left(1 + \frac{\Delta_1}{R_1}\right)\left(1 + \gamma\frac{\Delta_1}{R_1}\right)^{\frac{w_1}{w_2}} = \gamma^2(1+\epsilon).$$

Taking partial derivatives of both sides with respect to ϵ:

$$\begin{aligned}&\frac{\partial}{\partial\epsilon}\left(\left(1 + \frac{\Delta_1}{R_1}\right)\left(1 + \gamma\frac{\Delta_1}{R_1}\right)^{\frac{w_1}{w_2}}\right) = \frac{\partial}{\partial\epsilon}\left(\gamma^2(1+\epsilon)\right)\\ \Rightarrow &\frac{1}{R_1}\frac{\partial\Delta_1}{\partial\epsilon}\left(1 + \gamma\frac{\Delta_1}{R_1}\right)^{\frac{w_1}{w_2}} + \left(1 + \frac{\Delta_1}{R_1}\right)\frac{w_1}{w_2}\left(1 + \gamma\frac{\Delta_1}{R_1}\right)^{\frac{w_1}{w_2}-1}\frac{\gamma}{R_1}\frac{\partial\Delta_1}{\partial\epsilon} = \gamma^2\\ \Rightarrow &\frac{\partial\Delta_1}{\partial\epsilon}\frac{1}{R_1}\left(1 + \gamma\frac{\Delta_1}{R_1}\right)^{\frac{w_1}{w_2}}\left(1 + \gamma\left(1 + \frac{\Delta_1}{R_1}\right)\frac{w_1}{w_2}\left(1 + \gamma\frac{\Delta_1}{R_1}\right)^{-1}\right) = \gamma^2\\ \Rightarrow &\frac{\partial\Delta_1}{\partial\epsilon} = \frac{\gamma^2 R_1}{\left(1 + \gamma\frac{w_1}{w_2}\left(1 + \frac{\Delta_1}{R_1}\right)\left(1 + \gamma\frac{\Delta_1}{R_1}\right)^{-1}\right)\left(1 + \gamma\frac{\Delta_1}{R_1}\right)^{\frac{w_1}{w_2}}},\end{aligned}$$

as required.

C.4.2 Finding $\partial\Delta_2/\partial\epsilon$

Recall the implicit equation that defines Δ_2, Eq (C.111):

$$\left(1-\frac{\Delta_2}{R_2}\right)^{-1}\left(1+\frac{1}{\gamma}\left(\left(1-\frac{\Delta_2}{R_2}\right)^{-\frac{w_2}{w_1}}-1\right)\right)=\gamma^2\left(1+\epsilon\right).$$

Taking partial derivatives of both sides with respect to ϵ:

$$\begin{aligned}
&\frac{\partial}{\partial\epsilon}\left(\left(1-\frac{\Delta_2}{R_2}\right)^{-1}\left(1+\frac{1}{\gamma}\left(\left(1-\frac{\Delta_2}{R_2}\right)^{-\frac{w_2}{w_1}}-1\right)\right)\right)=\frac{\partial}{\partial\epsilon}\left(\gamma^2\left(1+\epsilon\right)\right)\\
\Rightarrow&\frac{\partial\Delta_2}{\partial\epsilon}\frac{1}{R_2}\left(1-\frac{\Delta_2}{R_2}\right)^{-2}\left(\left(1+\frac{1}{\gamma}\left(1-\frac{\Delta_2}{R_2}\right)^{-\frac{w_2}{w_1}}-\frac{1}{\gamma}\right)\right.\\
&\qquad\left.+\left(\frac{1}{\gamma}\frac{w_2}{w_1}\left(1-\frac{\Delta_2}{R_2}\right)^{-\frac{w_2}{w_1}}\right)\right)=\gamma^2\\
\Rightarrow&\frac{\partial\Delta_2}{\partial\epsilon}=\frac{\gamma^3R_2\left(1-\frac{\Delta_2}{R_2}\right)^2}{\left(1+\frac{w_2}{w_1}\right)\left(1-\frac{\Delta_2}{R_2}\right)^{-\frac{w_2}{w_1}}-(1-\gamma)},
\end{aligned}$$

as required.

C.5 Monotonicity of inequalities

We have two inequalities that we derive above, Eq (C.131),

$$\frac{w_2'}{w_1'+w_2'}\left(1+\gamma\frac{w_1}{w_2}\left(1+\frac{\Delta_1}{R_1}\right)\left(1+\gamma\frac{\Delta_1}{R_1}\right)^{-1}\right)\leq 1,$$

and Eq (C.135),

$$\frac{w_1'}{w_1'+w_2'}\geq\frac{1-(1-\gamma)\left(1-\frac{\Delta_2}{R_2}\right)^{\frac{w_2}{w_1}}}{1+\frac{w_2}{w_1}-(1-\gamma)\left(1-\frac{\Delta_2}{R_2}\right)^{\frac{w_2}{w_1}}}.$$

Here we are interested in the monotonicity of the level-sets of these w.r.t. Δ_1 and Δ_2. That is, if our values of w_1', w_2' satisfy their inequalities for some particular values of $w_1, w_2, R_1, R_2, \Delta_1, \Delta_2$, can we guarantee that the same values of w_1', w_2' also satisfy the inequalities for $\hat{\Delta}_1 < \Delta_1$ and $\hat{\Delta}_2 < \Delta_2$ (with w_1, w_2, R_1, R_2 fixed)?

For this to be the case, we need the level sets of the inequalities to always have the correct gradient so that smaller values of Δ_1, Δ_2 always move the inequality's boundary 'further away' from the value of w_1', w_2'. Let us handle the above equations in turn.

Δ_1 Consider the case where our w_1', w_2' values just satisfies the first inequality above, so that

$$\frac{\bar{w}_2'}{\bar{w}_1'+\bar{w}_2'}=\frac{1}{1+\gamma\frac{w_1}{w_2}\left(1+\frac{\Delta_1}{R_1}\right)\left(1+\gamma\frac{\Delta_1}{R_1}\right)^{-1}},,$$

where $\bar{w}'_1, \bar{w}'_2$ denotes these critical value of w'_1, w'_2. For $\bar{w}'_1, \bar{w}'_2$ to also satisfy the inequality for $\hat{\Delta}_1 < \Delta_1$, we need that

$$\frac{\partial}{\partial \Delta_1}\left(\frac{1}{1+\gamma\frac{w_1}{w_2}\left(1+\frac{\Delta_1}{R_1}\right)\left(1+\gamma\frac{\Delta_1}{R_1}\right)^{-1}}\right) < 0,$$

as then the required critical value will be larger for smaller Δ_1 values. Evaluating this partial derivative, we find that

$$\frac{\partial}{\partial \Delta_1}\left(\frac{1}{1+\gamma\frac{w_1}{w_2}\left(1+\frac{\Delta_1}{R_1}\right)\left(1+\gamma\frac{\Delta_1}{R_1}\right)^{-1}}\right) = -\frac{\frac{\gamma w_1}{w_2}\left(1+\gamma\frac{\Delta_1}{R_1}\right)^{-1}\left(1+\gamma\left(1+\frac{\Delta_1}{R_1}\right)\left(1+\gamma\frac{\Delta_1}{R_1}\right)^{-1}\right)}{\left(1+\gamma\frac{w_1}{w_2}\left(1+\frac{\Delta_1}{R_1}\right)\left(1+\gamma\frac{\Delta_1}{R_1}\right)^{-1}\right)^2}.$$

For $w_1 \in (0,1)$, $w_2 \in (0,1)$, $\gamma > 0$, $R_1 > 0$ and $\Delta_1 > 0$, clearly this gradient is always negative, as required.

Δ_2 Again let us take the critical value of the new weight such that the inequality is just satisfied, so

$$\frac{\bar{w}'_1}{\bar{w}'_1+\bar{w}'_2} = \frac{1-(1-\gamma)\left(1-\frac{\Delta_2}{R_2}\right)^{\frac{w_2}{w_1}}}{1+\frac{w_2}{w_1}-(1-\gamma)\left(1-\frac{\Delta_2}{R_2}\right)^{\frac{w_2}{w_1}}}. \tag{C.143}$$

Here we want the partial derivative of this critical value to always be positive, so that the required critical value is always smaller for smaller Δ_2 values, so we want that

$$\frac{\partial}{\partial \Delta_2}\left(\frac{1-(1-\gamma)\left(1-\frac{\Delta_2}{R_2}\right)^{\frac{w_2}{w_1}}}{1+\frac{w_2}{w_1}-(1-\gamma)\left(1-\frac{\Delta_2}{R_2}\right)^{\frac{w_2}{w_1}}}\right) > 0.$$

We can re-write Eq (C.143) more compactly as

$$\frac{\bar{w}'_1}{\bar{w}'_1+\bar{w}'_2} = \frac{1-f(\Delta_2)}{1+a-f(\Delta_2)}. \tag{C.144}$$

where $f(\Delta_2) = (1-\gamma)\left(1-\frac{\Delta_2}{R_2}\right)^{\frac{w_2}{w_1}}$ and $a = \frac{w_2}{w_1}$.

$$\frac{\partial}{\partial \Delta_2}\left(\frac{1-f(\Delta_2)}{1+a-f(\Delta_2)}\right) = -\frac{a\frac{\partial f(\Delta_2)}{\partial \Delta_2}}{(1+a-f(\Delta_2))^2}.$$

As $a > 0$, as $w_1 \in (0,1)$ and $w_2 \in (0,1)$, the gradient $\frac{\partial \bar{w}'_1}{\partial \Delta_2}$ is positive if $\frac{\partial f(\Delta_2)}{\partial \Delta_2} < 0$. Let us evaluate $\frac{\partial f(\Delta_2)}{\partial \Delta_2}$:

$$\begin{aligned}\frac{\partial f(\Delta_2)}{\partial \Delta_2} &= \frac{\partial}{\partial \Delta_2}\left((1-\gamma)\left(1-\frac{\Delta_2}{R_2}\right)^{\frac{w_2}{w_1}}\right)\\ &= \left(-\frac{1}{R_2}\frac{w_2}{w_1}(1-\gamma)\left(1-\frac{\Delta_2}{R_2}\right)^{\frac{w_2}{w_1}-1}\right)\end{aligned}$$

As $R_2 < \Delta_2$, $w_1 \in (0,1)$, $w_2 \in (0,1)$, $0 < \gamma < 1$, this gradient is always negative, so $\frac{\partial}{\partial \Delta_2}\left(\frac{1-(1-\gamma)\left(1-\frac{\Delta_2}{R_2}\right)^{\frac{w_2}{w_1}}}{1+\frac{w_2}{w_1}-(1-\gamma)\left(1-\frac{\Delta_2}{R_2}\right)^{\frac{w_2}{w_1}}}\right) >$ 0, as required.

References

Jason Milionis, Ciamac C. Moallemi, Tim Roughgarden, and Anthony Lee Zhang. Automated market making and loss-versus-rebalancing, 2022.

Alexander Nezlobin. Ethereum block times, mev, and lp returns, 2022. URL https://medium.com/@alexnezlobin/ethereum-block-times-mev-and-lp-returns-5c13dc99e80.

Dave White, Dan Robinson, and Hayden Adams. Automated market making and loss-versus-rebalancing, 2021. URL https://www.paradigm.xyz/2021/07/twamm.

Guillermo Angeris, Alex Evans, and Tarun Chitra. When does the tail wag the dog? curvature and market making, 2020.

Hayden Adams, Noah Zinsmeister, Moody Salem River Keefer, and Dan Robinson. Uniswap v3 core, 2021.

Andrew Nguyen, Loi Luu, and Ming Ng. Kyberswap: Dynamic automated market making, 2021.