

Optimal Rebalancing in Dynamic AMMs

Matthew Willetts and Christian Harrington

QuantAMM.fi

March 2024

Abstract

Dynamic AMM pools, as found in Temporal Function Market Making, rebalance their holdings to a new desired ratio (e.g. moving from being 50-50 between two assets to being 90-10 in favour of one of them) by introducing an arbitrage opportunity that disappears when their holdings are in line with their target. Structuring this arbitrage opportunity reduces to the problem of choosing the sequence of portfolio weights the pool exposes to the market via its trading function. Linear interpolation from start weights to end weights has been used to reduce the cost paid by pools to arbitrageurs to rebalance. Here we obtain the *optimal* interpolation in the limit of small weight changes (which has the downside of requiring a call to a transcendental function) and then obtain a cheap-to-compute approximation to that optimal approach that gives almost the same performance improvement. We then demonstrate this method on a range of market backtests, including simulating pool performance when trading fees are present, finding that the new approximately-optimal method of changing weights gives robust increases in pool performance. For a BTC-ETH-DAI pool from July 2022 to June 2023, the increases of pool P&L from approximately-optimal weight changes is $\sim 25\%$ for a range of different strategies and trading fees.

1 Introduction

For AMM with fixed, unchanging trading functions, any change in the ratios between market prices of tokens in a pool lead to that pool holding less value than when the pool was capitalised. So the market prices changing in ratio means the liquidity providers (in the absence of fees) are guaranteed to be outperformed by those who simply held their initial capital.

This problem of impermanent loss (IL), which can also be linked to Loss-versus-Rebalancing (LVR) [1, 2], is a serious challenge to AMM mechanisms where the trading function is static (which are the overwhelming majority of AMMs today).

In *Temporal Function Market Making* [3], AMM mechanisms are not used for core liquidity providing, but instead as an ultra-efficient rebalancing engine for running a time-varying portfolio. This is done by having geometric mean market maker (G3M) pools where the weights are always changing from block to block following a chosen quantitative asset management strategy.

Arbitrage opportunities arising from changes in a pool's weights are how the pool pays for rebalancing to be done. This can be interpreted as an auction [3, A.4], and if the pool's strategy for changing weights over time can provide greater P&L than has to be paid to arbitrageurs to rebalance, then the pool's LPs can experience an uplift in the value of their holdings.

TFMM pools have a G3M trading function, for N tokens with reserves $\mathbf{R} = \{R_1, R_2, \dots, R_N\}$ we have:

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

with $\mathbf{w}(t) = \{w_1(t), w_2(t), \dots, w_N(t)\}$ varying from block-to-block. Trades still have to preserve (or increase) $k(t)$ at the time of that trade. (For simplicity we study the zero-fees case.)

In this paper, we explore the setting where we assume *constant* market prices during the rebalancing process and *changing* weights. The question emerges: if we have initial weights $\mathbf{w}(t_0)$ and the pool's strategy is such that we need to get to $\mathbf{w}(t_f)$ (where t_f is multiple blocks in the future) what is the optimal sequence of weights to incur the least amount of slippage for the pool, maximally reducing the pool's rebalancing cost?

We will first give a quick overview of how pools' reserves change when weights change, then describe how we can optimise this process, and then give simulated results demonstrating improved performance.

2 Weight Interpolation: Execution Management on TFMMS

Weight changes when the pool is in equilibrium—quoting the market price—creates an arbitrage opportunity. This is desirable, we need to incentivise the pool’s reserves to be re-balanced to be in line with the new weights.

We begin with some weights $\mathbf{w}(t_0)$ and the pool in equilibrium. Then a block later, at time $t' = t_0 + \delta_t$, we have new weights $\mathbf{w}(t')$ (but market prices p do not change, so $\mathbf{p}(t_0) = \mathbf{p}(t')$). This change in weights leads to a change in quoted prices. For tokens with increasing weights, the pool is offering to buy the tokens from arbitrageurs for slightly more than their market price; for tokens with decreasing weights, the pool is offering to sell the tokens to arbitrageurs for slightly less than their market price. In this way, the pool pays arbitrageurs to rebalance to the pool’s desired holdings.

After this rebalancing arbitrage trade has happened, the pool holds reserves

$$\mathbf{R}(t') = \mathbf{R}(t_0) \frac{\mathbf{w}(t')}{\mathbf{w}(t_0)} \prod_{i=1}^N \left(\frac{w_i(t_0)}{w_i(t')} \right)^{w_i(t')}. \quad (2)$$

See Appendix B for the derivation.

Of course, it is desirable for LPs for the arbitrage opportunity to be smaller rather than larger. 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 over a period in time.

In other words, when weight updates are divided up into a series of smaller steps the total resulting arbitrage from all of those multiple steps is less than if the update were done in one step, under the assumption that market prices are constant over the period of time in which weights are changing.

2.1 A family of beneficial interpolations

We can compare $\mathbf{R}^{1\text{-step}}$, the reserves in the pool when we have directly updated the weights from $\mathbf{w}(t_0) \rightarrow \mathbf{w}(t_0) + \Delta\mathbf{w}$, a one-step process, to $\mathbf{R}^{2\text{-step}}$, the reserves when we have done this weight update via a two-step process: $\mathbf{w}(t_0) \rightarrow \tilde{\mathbf{w}} \rightarrow \mathbf{w}(t_0) + \Delta\mathbf{w}$, 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 C.1 that any member of this family of intermediate values for $\tilde{\mathbf{w}}$ gives a lower arbitrage cost.

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

$$\mathbf{R}_{\text{linear}}^{2\text{-step}} = \mathbf{R}_{\text{linear}}^{1\text{-step}} \prod_{j=1}^N \left(1 + \frac{\Delta w_j}{2w_j(t_0)} \right)^{\frac{\Delta w_j}{2}}. \quad (3)$$

We can get this result for linear interpolation via Eq (2). See Appendix B for the derivation. The elementwise ratio between $\mathbf{R}_{\text{linear}}^{2\text{-step}}$ and $\mathbf{R}_{\text{linear}}^{1\text{-step}}$ 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 to be done in one leap, we can produce a cheaper weight update procedure by going through and equilibrating at the midway value. After we apply this ‘bisection’ once, we can apply the same argument again on each half of the new trajectory, dividing changes in weights in half again.

Informally, 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 a simple and effective (and gas-efficient) way to give us good weight changes.

3 Non-linear weight interpolation schemes

While it is always better to perform a weight change via linear interpolation than to do it in one go, the benefit ‘caps out’ at the block resolution. Once one has hit that limit, is there anything else one can do? We will show that there are even more capital-efficient ways for pools to offer arbitrage opportunities to the market.

For a sequence of $f + 1$ weights, indexed from t_0 to t_f , $\{\mathbf{w}(t_k)\}_{k=0,\dots,f}$, and constant prices the final reserves are

$$\mathbf{R}(t_f) = \mathbf{R}(t_0) \frac{\mathbf{w}(t_f)}{\mathbf{w}(t_0)} \prod_{k=1}^f \prod_{j=1}^N \left(\frac{w_j(t_{k-1})}{w_j(t_k)} \right)^{w_j(t_k)},$$

which we obtain from repeated application of Eq (2).

Within this framework, one can attack this as an optimisation problem, as we are interested in finding the trajectory of weights that minimises the amount that is paid to arbs, i.e. maximises the pool value subject to constraints on the weights

$$\{\mathbf{w}^*(t_k)\}_{k=1,\dots,f-1} = \arg \max_{\{\mathbf{w}(t_k)\}_{k=1,\dots,f-1}} \left[\mathbf{p} \cdot \mathbf{R}(t_f) + \sum_{\ell=1}^{f-1} \left(\sum_{m=1}^N \mu_{\ell,m} w_m(t_\ell) \right) - \sum_{k=1}^{f-1} \Lambda_k \left(\sum_{i=1}^N w_i(t_k) - 1 \right) \right], \quad (4)$$

which fulfil the KKT conditions with $\{\mu_{\ell,m}\}$ and $\{\Lambda_k\}$ as multipliers. The optimum can be numerically calculated, as can more sophisticated approaches including changing prices (or even uncertain predictions of future prices) and the presence of fees. These calculations could be run on-chain or otherwise made available to a TFMM pool. During a TFMM weight update call is it possible to put in an external oracle call that returns a ‘weight trajectory array’. Of course, there is a trade-off here in greater running costs vs the improvement from a more-complex weight trajectory. The greater a pool’s TVL and the cheaper it is to obtain an optimised weight trajectory (by on-chain calculation or from an oracle call), the easier it would be to justify this expenditure.

3.1 Approximating the optimal weight interpolation

Optimal intermediate weights for small weight changes Assuming constant market prices over the course of interpolation, here we will describe a cheap-to-compute approximation to the zero-fees solution of Eq (4) that provides a superior method of pool rebalancing over linear interpolation.

During the derivations for §2.1 in Appendix C.1 we defined r , the ratio of reserves between when doing a 2-step weight change ($\mathbf{w}(t_0) \rightarrow \tilde{\mathbf{w}} \rightarrow \mathbf{w}(t_f)$) and a direct 1-step weight change ($\mathbf{w}(t_0) \rightarrow \mathbf{w}(t_f)$).

$$r = \prod_{j=1}^N \frac{w_j(t_0)^{\tilde{w}_j}}{w_j(t_0)^{w_j(t_f)}} \frac{\tilde{w}_j^{w_j(t_f)}}{\tilde{w}_j^{\tilde{w}_j}}. \quad (5)$$

If we assume that the weight change from $\mathbf{w}(t_0) \rightarrow \mathbf{w}(t_f)$ is small, we can directly optimise Eq (5) without needing to impose the constraints of Eq (4). Doing this (Appendix A), we find that the optimal intermediate value is

$$\tilde{w}_i^* = \frac{w_i(t_f)}{W_0 \left(\frac{e w_i(t_f)}{w_i(t_0)} \right)} \quad (6)$$

where $W_0(\cdot)$ is the principal branch of the Lambert W function and e is Euler’s constant. The Lambert W function is a special function, the solution to a transcendental equation.¹ While there are methods to approximate $W_0(\cdot)$ (eg [4, 5]) these a) can require (nested) calls to $\log(\cdot)$ that are not ideal for constrained-compute environments and b) are best over particular ranges of inputs while for us the ratios of start and end weights can take a broad range of values. We chose a different approach that gives good performance.

¹So not trivial to compute on-chain

Approximating the optimal value We can bound the value of \tilde{w}_i^* from above and below by the arithmetic mean and geometric mean (respectively) of $\{w_i(t_0), w_i(t_f)\}$:

$$\sqrt{w_i(t_0)w_i(t_f)} < \tilde{w}_i^* < \frac{1}{2}(w_i(t_0) + w_i(t_f)). \quad (7)$$

See Appendix A.2 for derivation. We find that using the arithmetic mean *of* the geometric² and arithmetic means of $\{w_i(t_0), w_i(t_f)\}$ provides a simple and effective method for approximating the optimal intermediate weight.³

From a two-step to multi-step approximation We can directly ‘bootstrap’ the approximations derived above into giving close-to-optimal many-step weight trajectories, even though the base analytic result, Eq (6), is only for a single intermediate point. Along a trajectory of many intermediate weights, the optimal \tilde{w}_i at any stage in that trajectory will lie above the geometric mean of the prior and post weights and below their arithmetic mean.

The approximately-optimal $f - 1$ -step weight trajectory $\{\check{\mathbf{w}}(t_k)\}_{k=1,\dots,f-1}$ for going from $\mathbf{w}(t_0)$ to $\mathbf{w}(t_f)$ is made by taking the average of the linear interpolation between these values and the geometric interpolation between these values (see Appendix A.3):

$$w_i^{\text{AM}}(t_k) = (1 - \frac{k}{f})w_i(t_0) + \frac{k}{f}w_i(t_f), \quad (8)$$

$$w_i^{\text{GM}}(t_k) = (w_i(t_0))^{(1-\frac{k}{f})}(w_i(t_f))^{\frac{k}{f}}, \quad (9)$$

$$\check{w}_i(t_k) = \frac{w_i^{\text{AM}}(t_k) + w_i^{\text{GM}}(t_k)}{\sum_{j=1}^N (w_j^{\text{AM}}(t_k) + w_j^{\text{GM}})}, \quad (10)$$

where we explicitly are normalising our weight trajectories to enforce that they sum to 1 at each time step.

4 Experiments

4.1 Numerical Examples

Here we find the optimal trajectory $\{\mathbf{w}^*(t_k)\}_{k=1,\dots,f-1}$ by numerically solving Eq (4) for a given choice of N , f , $\mathbf{w}(t_0)$ and $\mathbf{w}(t_f)$.

In Figures 1 & 2 we compare linear interpolation and $\{\check{\mathbf{w}}(t_k)\}_{k=1,\dots,f-1}$ and numerically-found weights $\{\mathbf{w}^*(t_k)\}_{k=1,\dots,f-1}$ for a 3-token pool. In Figure 1 we plot the weights themselves, in Figure 2 we plot the block-to-block changes in weights.

Even though the overall weight change is large, $\mathbf{w}(t_0) = \{0.05, 0.55, 0.4\}$ and $\mathbf{w}(t_f) = \{0.4, 0.5, 0.1\}$, and we do this over $f = 1000$ steps, our approximate method and the optimal are hard to visually distinguish.

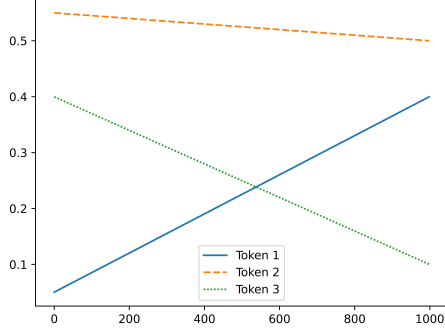
In Figure 3 we plot the differences between numerically-found $\{\mathbf{w}^*(t_k)\}_{k=1,\dots,f-1}$ and each of linear and approximately-optimal interpolations. Clearly the approximately-optimal interpolation is much closer to optimal, with a largest absolute weight discrepancy of ≈ 0.003 compared to ≈ 0.04 for linear interpolation.

In this example the approximately-optimal trajectory $\{\check{\mathbf{w}}(t_k)\}_{k=1,\dots,f-1}$ captures $\approx 95\%$ of the increase in pool value that one would get from swapping out linear interpolation with the numerical $\{\mathbf{w}^*(t_k)\}_{k=1,\dots,f-1}$.

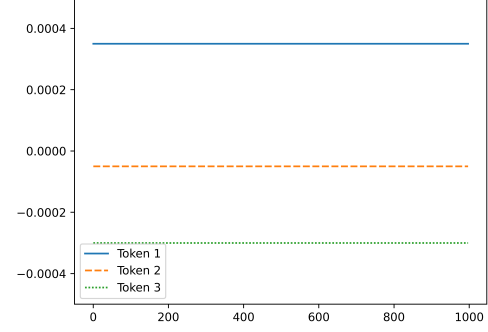
²Note that $\lim_{|\mathbf{w}(t_f) - \mathbf{w}(t_0)|_1 \rightarrow 0} (\sum_{i=1}^n \sqrt{w_i(t_0)w_i(t_f)}) = 1$.

³We could use another mean here (and in the final construction of $\check{\mathbf{w}}(t_k)$) geometric or harmonic say, but the arithmetic mean empirically works well and is cheaper to compute.

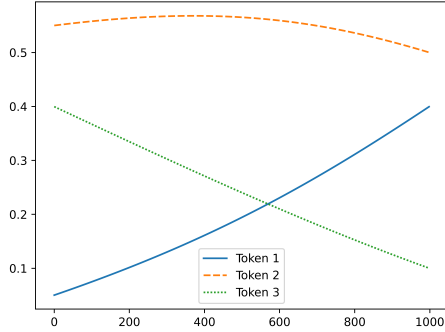
Figure 1: Example weight interpolations done by a) linear interpolation, b) approximately-optimal (Eq (10)) and c) numerically-obtained optimal (Eq (4)), for a $N = 3$ token pool with $\mathbf{w}(t_0) = \{0.05, 0.55, 0.4\}$, $\mathbf{w}(t_f) = \{0.4, 0.5, 0.1\}$ and $f = 1000$. At this scale the differences between b) and c) are not easily apparent.



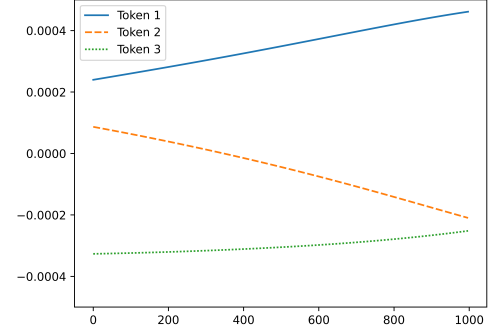
(a) Linear



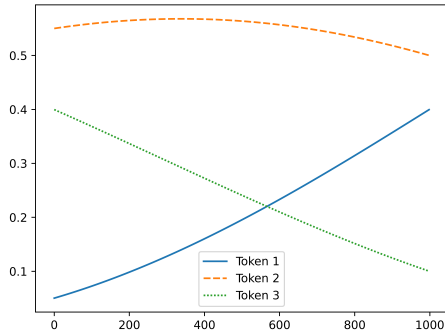
(a) Linear



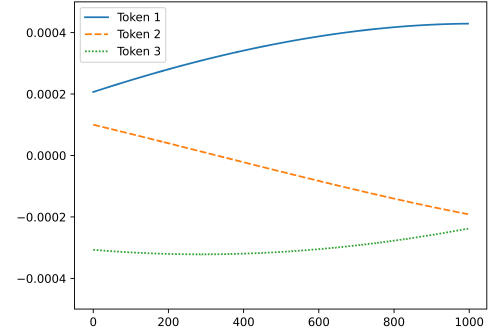
(b) Approximately-optimal



(b) Approximately-optimal



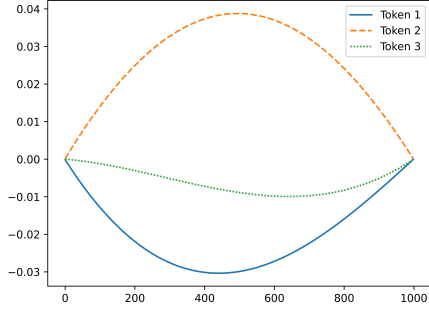
(c) Optimal



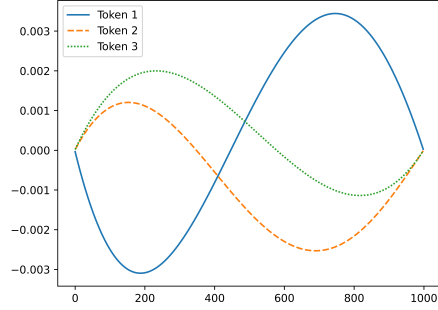
(c) Optimal

Figure 2: Plots showing the weight changes block to block (i.e. plotting each component of $\mathbf{w}(t+1) - \mathbf{w}(t)$) for a) linear interpolation, b) approximately-optimal (Eq (10)) and c) numerically-obtained optimal weights, same setup as Fig 1 above.

Figure 3: Plots showing how much linear interpolation (a) and approximately-optimal trajectories (b) deviate from the numerically-obtained trajectory $\{\mathbf{w}^*(t_k)\}_{k=1,\dots,f-1}$, in the same setup as Fig 1. We plot the differences. The approximately-optimal method very closely matches the numerically-obtained optimal.

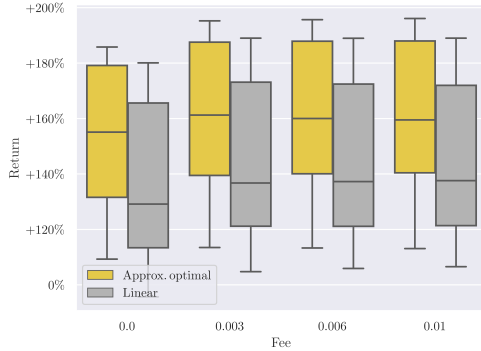


(a) Optimal minus linear

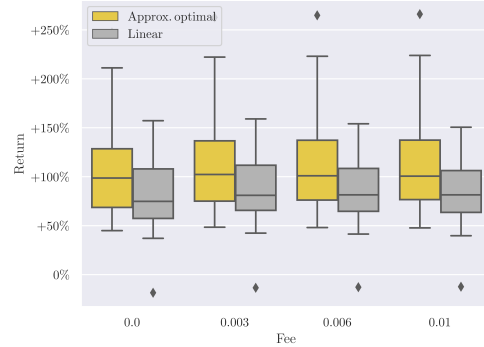


(b) Optimal minus approximately-optimal

Figure 4: Plots showing the trained returns for (a) approximately-optimal weight changes (b) linear weight changes. We plot the raw returns over the period. The performance increase from using approximately-optimal weight changes is maintained even as fees are introduced, even with fees of 1%.



(a) Momentum strategy



(b) Channel following strategy

4.2 Historical data

We can train strategies for TFMMS, following the approach outlined the QuantAMM litepaper [6]. New target weights are calculated using as input estimates of the gradients of the (log) prices of assets in the pool, for example, a simple momentum strategy. These strategies have numerical parameters that control how the price gradients are calculated and how aggressively the pool acts to change weights based on those price gradients. Different parameters lead to different performance.

Very briefly, we use stochastic gradient descent to tune a strategy. First, we simulate the reserves held by a pool held over time from the interplay between the strategy's changing weights and changing market prices over a sampled period of price history. From this we can calculate measures of the performance of the pool (e.g. Sharpe ratio) over that time period. We perform this calculation in auto-differentiable computer code so we can take the gradient of the chosen performance measure with respect to the numerical parameters of the strategy. We then update the parameters to increase measured performance.

The weight update procedure used (linear vs approximately-optimal) changes pool performance. To most fairly test the effect of weight update procedure one has to train multiple different instances of strategies with each of these weight update procedures. Instances of strategies learn to make use of the

weight change procedure used. A strategy trained under approximately-optimal weight changes might dramatically underperform if re-applied to rebalance via linear changes, but by separately training strategies that rebalance via linear changes we can then fairly compare performance.

We train 160 variants for each combination of strategy and weight change procedure, on a basket of BTC, ETH, DAI for the period July 2022 to June 2023. The different variants calculate new target weights at different cadences (every hour vs. every day), start with different random initial parameters, and have slightly different methods of windowing data during training.

We train both a simple momentum strategy and a channel-following strategy. We train with zero fees, but we run trained strategies over price data with fees present, following recent approaches for closed-form optimal arbitrage trades in multi-token pools [7].

We show the results, plotting raw returns in Figure 4. We find that strategies can effectively learn to make use of approximately-optimal weight changes, giving increases in returns of $\sim 25\%$ on average for the basket and period we have tested.

Further, this improvement is maintained (or even grows) as trading fees are introduced, showing the robustness of this approach. Incidentally, one can think of comparing approximately-optimal weight changes with high trading fees to linear weight changes with low fees as a rough test of having a larger trading gas cost for approximately-optimal weight changes—as the trader might have to pay more for the weights to be loaded or for the calculation to be done on-chain during the trade. (Of course on cheap L2s this effect would be minimal.) See Figure D.6 for the same data plotted to make this even easier to see.

5 Concluding remarks

Improved methods to dynamically change pool holdings are key to unlocking on-chain asset management for DeFi. Doing this to reduce arbitrage opportunities during rebalancing leads to greater market efficiency. We have introduced a new method for achieving an overall change in holdings that reduces rebalancing costs for a dynamic pool that rebalances its holdings by the construction of AMM-style arbitrage opportunities. This approach can lead to increased returns in simulation and is robust to large trading fees being charged.

References

- [1] Alexander Nezhlobin. Ethereum block times, mev, and lp returns, 2022. URL <https://medium.com/@alexnezlobin/ethereum-block-times-mev-and-lp-returns-5c13dc99e80>.
- [2] Jason Milionis, Ciamac C. Moallemi, Tim Roughgarden, and Anthony Lee Zhang. Automated market making and loss-versus-rebalancing, 2022.
- [3] QuantAMM team. Temporal-function market making litepaper, 2023. URL <https://www.quantamm.fi/litepapers>.
- [4] Stefano D’Angelo, Leonardo Gabrielli, and Luca Turchet. Fast approximation of the lambert w function for virtual analog modelling. In *International Conference on Digital Audio Effects (DAFx-19)*, volume 22, 2019.
- [5] Robert Corless, David Jeffrey, and Donald Knuth. A sequence of series for the lambert w function. *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC*, pages 197–204, 01 1997. doi: 10.1145/258726.258783.
- [6] QuantAMM team. Quantamm litepaper, 2023. URL <https://www.quantamm.fi/litepapers>.
- [7] Matthew Willetts and Christian Harrington. Closed-form solutions for generic n-token amm arbitrage, 2024.

Appendices

A Derivation of Optimal and Approximately-optimal weight change procedures

A.1 Optimal weight changes in the limit of small weight changes

While we show below that a broad family of possible interpolations (§C.1) and linear interpolations themselves (§C.2), can we do better?

Here we derive Eq (6), the optimal intermediate weight in a 2-step process in the limit of small weight changes.

Taking partial derivatives of Eq (C.37), we get

$$\frac{\partial r}{\partial \tilde{w}_i} = r \left(\frac{w_i(t_f)}{\tilde{w}_i} + \log \left(\frac{w_i(t_0)}{\tilde{w}_i} \right) - 1 \right). \quad (\text{A.11})$$

Solving for $\frac{\partial r}{\partial \tilde{w}_i} = 0$ gives us the transcendental equation

$$1 - \log \left(\frac{w_i(t_0)}{\tilde{w}_i} \right) = \frac{w_i(t_f)}{\tilde{w}_i}, \quad (\text{A.12})$$

which has solution

$$\tilde{w}_i = \frac{w_i(t_f)}{W_0 \left(\frac{ew_i(t_f)}{w_i(t_0)} \right)}, \quad (\text{A.13})$$

where $W_0(\cdot)$ is the principal branch of the function that solves the equation $w \exp w = x$, the Lambert W function.

As we have not required that $\sum_{i=1}^N \tilde{w}_i = 1$, Eq (A.13) is only valid in the limit $\lim_{|\mathbf{w}(t_f) - \mathbf{w}(t_0)|_1 \rightarrow 0}$.

A.2 Bounding \tilde{w}_i above and below

To avoid having to calculate $W_0(\cdot)$ on chain we instead bound \tilde{w}_i from above and below and then use the average of these bounds as an approximate value.

We will bound \tilde{w}_i from below by the geometric mean of the start and end weights, and from above by their arithmetic mean.

Bounding from below We will show

$$\tilde{w}_i \geq \sqrt{w_i(t_0)w_i(t_f)}.$$

We have that $\tilde{w}_i = \frac{w_i(t_f)}{W_0 \left(\frac{ew_i(t_f)}{w_i(t_0)} \right)}$, which can be rearranged using the property $\frac{x}{W_0(x)} = \exp(W(x))$ to

$$\tilde{w}_i = w_i(t_0) \exp \left(W_0 \left(\frac{ew_i(t_f)}{w_i(t_0)} \right) - 1 \right). \quad (\text{A.14})$$

Requiring $\tilde{w}_i \geq \sqrt{w_i(t_0)w_i(t_f)}$ is equivalent to

$$w_i(t_0) \exp \left(W_0 \left(\frac{ew_i(t_f)}{w_i(t_0)} \right) - 1 \right) \geq \sqrt{w_i(t_0)w_i(t_f)} \quad (\text{A.15})$$

$$\Rightarrow \exp \left(W_0 \left(\frac{ew_i(t_f)}{w_i(t_0)} \right) - 1 \right) \geq \sqrt{\frac{w_i(t_f)}{w_i(t_0)}} \quad (\text{A.16})$$

$$\Rightarrow W_0 \left(\frac{ew_i(t_f)}{w_i(t_0)} \right) - 1 \geq \frac{1}{2} \log \left(\frac{w_i(t_f)}{w_i(t_0)} \right). \quad (\text{A.17})$$

Defining $u := \frac{w_i(t_f)}{w_i(t_0)}$ and $g(u) := W_0(eu) - 1 - \frac{1}{2} \log(u)$, we find that $g(u)$ has a single positive real root at $u = 1$, which means that Eq (A.17) reaches equality only for $\frac{w_i(t_f)}{w_i(t_0)} = 1$.

It is easy to show that

$$\frac{\partial g}{\partial u} = \frac{1}{2} \frac{W_0(eu) - 1}{u(1 + W_0(eu))}$$

We find that at the root we have

$$\Rightarrow \left. \frac{\partial g}{\partial u} \right|_{u=1} = 0,$$

(where we have used the result $W_0(e) = 1$). Further,

$$\begin{aligned} \frac{\partial^2 g}{\partial u^2} &= \frac{1}{2} \frac{1 + 3W_0(eu) - (W_0(eu))^2 - (W_0(eu))^3}{u^2(1 + W_0(eu))^3} \\ &\Rightarrow \left. \frac{\partial^2 g}{\partial u^2} \right|_{u=1} = \frac{1}{8}. \end{aligned}$$

As $g(u)$ has a single positive real root, which is a turning point with positive second derivative, $g(u) \geq 0$ for all real values of $u > 0$. Thus $W_0\left(\frac{ew_i(t_f)}{w_i(t_0)}\right) - 1 \geq \frac{1}{2} \log\left(\frac{w_i(t_f)}{w_i(t_0)}\right)$ and so $\tilde{w}_i \geq \sqrt{w_i(t_0)w_i(t_f)}$ as required.

Bounding from above We will show that

$$\tilde{w}_i \leq \frac{1}{2} (w_i(t_0) + w_i(t_f))$$

is satisfied under the requirement that weights have entries between 0 and 1. Requiring $\tilde{w}_i \leq \frac{1}{2} (w_i(t_0) + w_i(t_f))$ is equivalent to

$$w_i(t_0) \exp\left(W_0\left(\frac{ew_i(t_f)}{w_i(t_0)}\right) - 1\right) \leq \frac{1}{2} (w_i(t_0) + w_i(t_f)) \quad (\text{A.18})$$

$$\Rightarrow \exp\left(W_0\left(\frac{ew_i(t_f)}{w_i(t_0)}\right) - 1\right) \leq \frac{1}{2} \left(1 + \frac{w_i(t_f)}{w_i(t_0)}\right) \quad (\text{A.19})$$

$$\Rightarrow W_0\left(\frac{ew_i(t_f)}{w_i(t_0)}\right) - 1 \leq \log\left(\frac{1}{2} \left(1 + \frac{w_i(t_f)}{w_i(t_0)}\right)\right). \quad (\text{A.20})$$

Defining $u := \frac{w_i(t_f)}{w_i(t_0)}$ and $h(u) := W_0(eu) - 1 - \log\left(\frac{1}{2}(1 + u)\right)$, we find that $h(u)$ has a single positive real root at $u = 1$, which means that Eq (A.20) reaches equality only for $\frac{w_i(t_f)}{w_i(t_0)} = 1$.

It is easy to show that

$$\frac{\partial h}{\partial u} = \frac{W_0(eu)}{u(1 + W_0(eu))} - \frac{1}{1 + u}$$

We find that at the root we have

$$\Rightarrow \left. \frac{\partial h}{\partial u} \right|_{u=1} = 0,$$

(where we have used the result $W_0(e) = 1$). Further,

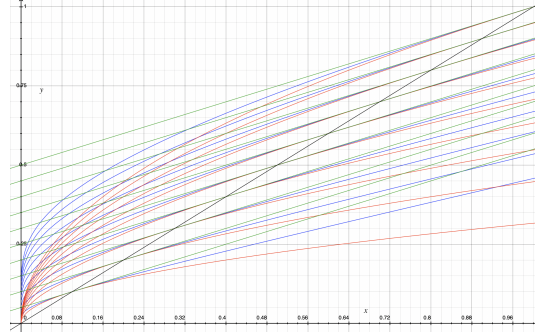
$$\begin{aligned} \frac{\partial^2 h}{\partial u^2} &= \frac{1}{u^2(1 + W_0(eu))} + \frac{1}{u^2(1 + W_0(eu))^2} - \frac{1}{u^2(1 + W_0(eu))^3} - \frac{2u + 1}{u^2(1 + u)^2} \\ &\Rightarrow \left. \frac{\partial^2 h}{\partial u^2} \right|_{u=1} = -\frac{1}{8}. \end{aligned}$$

As $h(u)$ has a single positive real root, which is a turning point with negative second derivative, $h(u) \leq 0$ for all real values of $u > 0$. Thus $W_0\left(\frac{ew_i(t_f)}{w_i(t_0)}\right) - 1 \leq \log\left(\frac{1}{2} \left(1 + \frac{w_i(t_0)}{w_i(t_f)}\right)\right)$ and so $\tilde{w}_i \leq \frac{1}{2} (w_i(t_0) + w_i(t_f))$ as required.

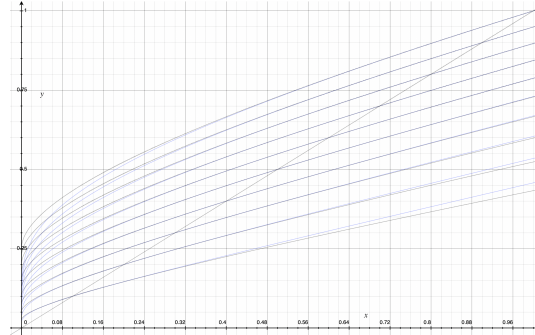
A.2.1 Plots

Here we plot the arithmetic & geometric means of $w_i(t_0), w_i(t_f)$ and the optimal \tilde{w}_i , showing the bounding we have derived.

Figure A.5: Plots showing graphically how the bounds work and how they form an approximation to the optimal \tilde{w}_i .



(a) The arithmetic means (green) and geometric means (red) of $w_i(t_0), w_i(t_f)$, plotted as a function of $w_i(t_f)$ for 10 uniform values of $w_i(t_0)$ from 0.1 to 1.0. In blue are the optimal \tilde{w}_i . Note that for each set of curves the optima (blue) is bounded from below by the geometric mean (red) and from above by the arithmetic mean (green). The black line is $y=x$; when $w_i(t_0) = w_i(t_f)$ the means and the optima all converge.



(b) \tilde{w}_i (blue) plotted as a function of $w_i(t_f)$ for 10 uniform values of $w_i(t_0)$ from 0.1 to 1.0. The 10 black curves are the arithmetic means of the geometric and arithmetic means of $w_i(t_0), w_i(t_f)$ (the red and green curves above). This bound-average gives a close approximation to \tilde{w}_i and is most accurate near $y=x$ (i.e. $w_i(t_0) \approx w_i(t_f)$).

A.3 Approximately-optimal trajectories

We bootstrap our ‘infinitesimal’ results for optimal and approximately-optimal intermediate weights into a trajectory of many intermediates, Eqs (8-10).

$$\check{w}_i(t_k) = \frac{w_i^{\text{AM}}(t_k) + w_i^{\text{GM}}(t_k)}{\sum_{j=1}^N (w_j^{\text{AM}}(t_k) + w_j^{\text{GM}})} \quad (\text{A.21})$$

$$w_i^{\text{AM}}(t_k) = \left(1 - \frac{k}{f}\right) w_i(t_0) + \frac{k}{f} w_i(t_f) \quad (\text{A.22})$$

$$w_i^{\text{GM}}(t_k) = (w_i(t_0))^{(1-\frac{k}{f})} (w_i(t_f))^{\frac{k}{f}}, \quad (\text{A.23})$$

This works as locally-intermediate values of $w_i^{\text{AM}}(t_k)$ and $w_i^{\text{GM}}(t_k)$ are themselves the arithmetic and geometric means of their immediate neighbours:

$$w_i^{\text{GM}}(t_k) = \sqrt{w_i^{\text{GM}}(t_{k-1}) w_i^{\text{GM}}(t_{k+1})}, \quad w_i^{\text{AM}}(t_k) = \frac{1}{2} (w_i^{\text{GM}}(t_{k-1+}) w_i^{\text{GM}}(t_{k+1})) .$$

So locally we have a sequence of relatively small weight changes, each $\check{w}_i(t_k)$ of which closely approximates Eq (6).

B Reserve-Update Derivation: Weights Change, Prices Constant

The result in this section previously appeared in the TFMM litepaper [3, Appendix A.2], we reproduce them here for clarity and ease of access.

Here we derive Eq (2), 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

$$\prod_{i=1}^N R_i(t_0)^{w_i(t')} = k(t'), \quad (\text{B.24})$$

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

$$\prod_{i=1}^N R_i(t')^{w_i(t')} = k(t') = \prod_{i=1}^N R_i(t_0)^{w_i(t')}. \quad (\text{B.25})$$

$$\Rightarrow \prod_{i=1}^N \left(\frac{R_i(t')}{R_i(t_0)} \right)^{w_i(t')} = 1. \quad (\text{B.26})$$

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 standard results for quoted prices in G3Ms (for example [6, Eq A.14]), and without loss of generality using the first token as the numéraire, we have that initially, before the change in weights,

$$\frac{\frac{w_i(t_0)}{R_i(t_0)}}{\frac{w_1(t_0)}{R_1(t_0)}} = p_i(t_0) \quad (\text{B.27})$$

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

$$\frac{\frac{w_i(t')}{R_i(t')}}{\frac{w_1(t')}{R_1(t')}} = p_i(t_0). \quad (\text{B.28})$$

Thus we have

$$\frac{w_i(t')w_1(t_0)}{R_i(t')R_1(t_0)} = \frac{w_1(t')w_i(t_0)}{R_1(t')R_i(t_0)}, \quad (\text{B.29})$$

$$\frac{R_i(t')}{R_i(t_0)} = \frac{w_i(t')w_1(t_0)R_1(t')}{w_1(t')w_i(t_0)R_1(t_0)}. \quad (\text{B.30})$$

Subbing Eq (B.30) into Eq (B.26), we get

$$\prod_{i=1}^N \left(\frac{R_i(t')}{R_i(t_0)} \right)^{w_i(t')} = \prod_{i=1}^N \left(\frac{w_i(t')w_1(t_0)R_1(t')}{w_1(t')w_i(t_0)R_1(t_0)} \right)^{w_i(t')} = 1 \quad (\text{B.31})$$

$$= \frac{w_1(t_0)R_1(t')}{w_1(t')R_1(t_0)} \prod_{i=1}^N \left(\frac{w_i(t')}{w_i(t_0)} \right)^{w_i(t')} = 1 \quad (\text{B.32})$$

$$\Rightarrow \frac{w_1(t_0)R_1(t')}{w_1(t')R_1(t_0)} = \prod_{i=1}^N \left(\frac{w_i(t_0)}{w_i(t')} \right)^{w_i(t')}. \quad (\text{B.33})$$

Now subbing Eq (B.33) into Eq (B.30) and rearranging we get

$$R_i(t') = R_i(t_0) \frac{w_i(t')}{w_i(t_0)} \prod_{j=1}^N \left(\frac{w_j(t_0)}{w_j(t')} \right)^{w_j(t')}, \quad (\text{B.34})$$

as we have in Eq (2) in the main paper, completing this derivation.

C Two-step updates are better than one-step updates

Some of the result in this section previously appeared in the TFMM litepaper [3, Appendix A.3], we reproduce them here for clarity and ease of access.

C.1 A broad family of possible interpolations

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 (B.34), 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 $\mathbf{w}(t_0) \rightarrow \mathbf{w}(t_f)$.

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: $\mathbf{w}(t_0) \rightarrow \tilde{\mathbf{w}} \rightarrow \mathbf{w}(t_f)$. 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 (B.34) directly gives us the ratio of the change of reserves,

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

Two-step Applying Eq (B.34) to the first leg, we get

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

And to the second leg

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

This gives an overall change of reserves of

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

Comparison The ratio $r_i^{2\text{-step}}/r_i^{1\text{-step}}$ is

$$r_i^{2\text{-step}}/r_i^{1\text{-step}} = r = \prod_{j=1}^N \left(\frac{w_j(t_0)}{\tilde{w}_j} \right)^{\tilde{w}_j} \left(\frac{\tilde{w}_j}{w_j(t_f)} \right)^{w_j(t_f)} \left(\frac{w_j(t_f)}{w_j(t_0)} \right)^{w_j(t_f)} \quad (\text{C.35})$$

$$= \prod_{j=1}^N \left(\frac{w_j(t_0)}{\tilde{w}_j} \right)^{\tilde{w}_j} \left(\frac{\tilde{w}_j}{w_j(t_0)} \right)^{w_j(t_f)} \quad (\text{C.36})$$

$$\Rightarrow r = \prod_{j=1}^N \frac{w_j(t_0)^{\tilde{w}_j}}{w_j(t_0)^{w_j(t_f)}} \frac{\tilde{w}_j^{w_j(t_f)}}{\tilde{w}_j^{\tilde{w}_j}}. \quad (\text{C.37})$$

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

$$f_j = \frac{w_j(t_0)^{\tilde{w}_j}}{w_j(t_0)^{w_j(t_f)}} \frac{\tilde{w}_j^{w_j(t_f)}}{\tilde{w}_j^{\tilde{w}_j}}. \quad (\text{C.38})$$

$w_j(t_f) > w_j(t_0)$: 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 (C.38), 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 (C.38), 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.

C.2 Linear Interpolation

Here we will derive Eq (3), finding the relationship between $\mathbf{R}^{1\text{-step}}$, the reserves in the pool when we have directly updated the weights from $\mathbf{w}(t_0) \rightarrow \mathbf{w}(t_0) + \Delta\mathbf{w}$, a one-step process, and $\mathbf{R}^{2\text{-step}}$, the reserves when we have done this weight update via a two step process: $\mathbf{w}(t_0) \rightarrow \mathbf{w}(t_0) + \frac{1}{2}\Delta\mathbf{w} \rightarrow \mathbf{w}(t_0) + \Delta\mathbf{w}$.

First Eq (B.34) derived above directly gives us the one-step value:

$$R_i^{1\text{-step}} = R_i(t') \tag{C.39}$$

$$= R_i(t_0) \frac{w_i(t_0) + \Delta w_i}{w_i(t_0)} \prod_{j=1}^N \left(\frac{w_j(t_0)}{w_j(t_0) + \Delta w_j} \right)^{w_j(t_0) + \Delta w_j}. \tag{C.40}$$

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

$$R_i^{\Delta w/2} = R_i(t_0) \frac{w_i(t_0) + \frac{\Delta w_i}{2}}{w_i(t_0)} \prod_{j=1}^N \left(\frac{w_j(t_0)}{w_j(t_0) + \frac{\Delta w_j}{2}} \right)^{w_j(t_0) + \frac{\Delta w_j}{2}}. \tag{C.41}$$

And then completing this two-step process

$$R_i^{2\text{-step}} = R_i^{\Delta w/2} \frac{w_i(t_0) + \Delta w_i}{w_i(t_0) + \frac{\Delta w_i}{2}} \prod_{j=1}^N \left(\frac{w_j(t_0) + \frac{\Delta w_j}{2}}{w_j(t_0) + \Delta w_j} \right)^{w_j(t_0) + \Delta w_j} \tag{C.42}$$

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

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

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

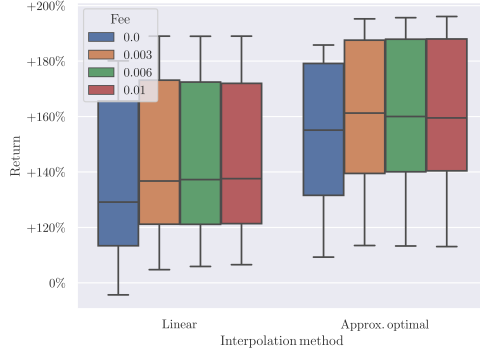
Thus we have that

$$\mathbf{R}^{2\text{-step}} = \mathbf{R}^{1\text{-step}} \prod_{j=1}^N \left(1 + \frac{\Delta w_j}{2w_j(t_0)} \right)^{\frac{\Delta w_j}{2}}, \tag{C.45}$$

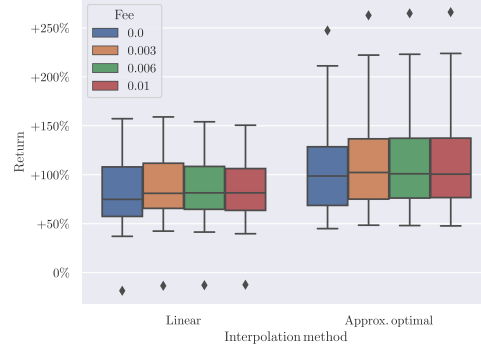
as required.

D Supplementary plots

Figure D.6: Differing from Figure 4, we group results here by weight change method to make it easier to see the effect of trading fees. We show the trained returns for (a) approximately-optimal weight changes (b) linear weight changes. We plot the raw returns over the period. The performance increase from using approximately-optimal weight changes is maintained even as fees are introduced, even with fees of 1%.



(a) Momentum strategy



(b) Channel following strategy

DISCLAIMER This paper is for general information purposes only. It does not constitute investment advice or a recommendation or solicitation to buy or sell any investment or asset, or participate in systems that use TFMM. This paper should not be used in the evaluation of the merits of making any investment decision. It should not be relied upon for accounting, legal or tax advice or investment recommendations. This paper reflects current opinions of the authors regarding the development and functionality of TFMM and is subject to change without notice or update.

While some aspects, such as altering target weights in geometric mean market makers is prior art, aspects of TFMM that are novel such as, but not exclusively, composability mechanisms, efficient methods for gradients and covariances, generic form multi-token amplification and advanced execution management mechanisms for use in dynamic weight AMMs, for purposes of core liquidity providing or forms of asset management including, but not exclusively, fund construction, structured products, treasury management are covered by patent filing date of 21st February 2023.