Leveraging the Uniswap Invariant

Dmitri Senchenko

January 5, 2020

Abstract

Automated market makers often serve a dual purpose in decentralised finance – to generate trading fees for liquidity providers and to rebalance their asset portfolios towards a predetermined target composition. In this paper, we argue that these two goals are often contradictory, and that it is possible to design an AMM that serves the needs of fee revenue-focused market makers better than Uniswap's original constant product design. Using leverage coefficients for individual assets, we construct an AMM that decouples terms of trade from market markers' inventory.

1 MOTIVATION

Automatic market makers (AMMs) are playing an increasingly important role in decentralised finance. Such protocols as Uniswap and Balancer allow liquidity providers (LPs) to generate passive income from trading fees and, at the same time, to algorithmically rebalance their investment portfolios. However, even at these early stages of their development, it is becoming clear that the two key functions of these constructs – market making and portfolio optimisation – may at times be working against each other.

Market making aims to maximise trading volumes and the associated fee revenue, while making sure that the terms of trade do not expose LPs to excessive market risk. Portfolio rebalancing, on the other hand, focuses on creating incentives to trade against the LPs in such a way as to move their portfolio closer to some target composition. In the former case, high asset turnover within the LP portfolio is part of the goal. In the latter, it may be part of the problem – the shortest path to the target state is often preferable.

These conflicting goals have led to inefficiencies, particularly for LPs of the market making persuasion. Even in the most popular AMM pools, only a small fraction of assets is ever traded, while the rest lie idle and do not generate trading fees. For the market maker, that is inefficient use of capital. It begs the question – is it possible to design an AMM such that removing excess reserves from its pools would not significantly affect its trading dynamics?

Below, we will argue that such a result can be achieved by introducing leverage directly into the Uniswap's constant product invariant. Our approach differs significantly from that used to construct the leveraged StableSwap invariant employed by Curve – it supports leverage levels below 1 and gives the freedom to choose asset prices independently of their quantities in the pool.

2 HOW IT WORKS

For any given level of asset reserves, we would like to create an AMM pool where we could specify any desired starting price and level of slippage. For instance, let's say we start off with two assets, one of which is cheap but scarce. Under the original Uniswap design, if a pool contains a relatively small amount of an asset, it automatically becomes expensive. This would force us to either buy more of the scarce asset than we want, or to take some of the other asset out of the pool. Furthermore, suppose the amounts of both assets in our possession are small relative to their average trade size in the overall market. When our pool comes into contact with such average trade, it will be interpreted as "large" and deserving of high slippage – the terms of trade offered by our pool will not be competitive.

What if, when creating the pool, we could "trick" the AMM into thinking that we have more, or less, of any asset than we do in reality? We could then control the "perceived" asset ratio in the pool, and hence set any initial price that we want. We could also make any incoming trade look relatively small, and thus have the AMM offer more competitive terms of trade. Below, we set out a method to do just that. Using leverage coefficients, we effectively allow pool creators to choose the scale of the invariant (i.e. perceived value of assets in the pool), and to pick any starting point on it (i.e. set the starting price) independently of the actual asset reserves in their possession. In other words, our AMM design decouples terms of trade from market markers' inventory.

2.1 Adding Leverage to the Constant Product Invariant

First, we restate the original version of the constant product invariant. To simplify presentation, we will assume zero fees throughout. Thus, given pool reserves (R_i) of assets $i \leq n$, the equation for a permitted trade (Δ_i) and pairwise prices of i in units of j are as follows ($\Delta_j > 0$ is paid by the trader):

$$\prod_{i=1}^{n}(R_i + \Delta_i) = \prod_{i=1}^{n} R_i$$

$$SpotPrice(i,j) = \frac{R_j}{R_i}$$

$$AveragePrice(i,j) = \frac{R_j + \Delta_j}{R_i}$$

We now introduce leverage coefficients (l_i), with $l > 0$. Taking the original reserves (R_i), we multiply them by these coefficients and insert the resulting amounts into the original invariant. Additionally, we now need to explicitly enforce the asset availability condition – one that was previously dealt with by the fact that prices would tend to infinity if one tried to acquire the entire reserve of any asset.

$$\prod_{i=1}^{n}(l_i \cdot R_i + \Delta_i) = \prod_{i=1}^{n}(l_i \cdot R_i)$$

$$R_i + \Delta_i > R_i$$

The trade amounts (Δ_i) are not subject to leverage, which is what gives the desired ability to scale the hypothetical reserves arbitrarily relative to the trades. The corresponding pairwise prices are:

$$SpotPrice(i,j) = \frac{l_j \cdot R_j}{l_i \cdot R_i}$$

$$AveragePrice(i,j) = \frac{l_j \cdot R_j + \Delta_j}{l_i \cdot R_i}$$

From the spot price equation, we can see that $\frac{R_j}{R_i}$ no longer dictates the price level – the choice of l_i and l_j does. Just as importantly, their choice also determines slippage. For any given trade size, one can close the gap between the spot price and the average price arbitrarily by scaling both l_i and l_j by the same, large enough constant. Thus, when creating a new pool, one can choose (l_i) which give the desired prices no matter the starting reserves, and then scale the leverage coefficients to achieve the desired slippage levels. Loosely speaking, $\prod l_i = 2$ is similar to 2x leverage.

To aid intuition, we visualise two transformations of the original Uniswap invariant into a leveraged one, each highlighting a distinct effect. Figure 1 depicts the case where $l_i = l > 1$. The resulting spot prices remain unchanged, and the effect of leverage is akin to LPs adding more of each asset to the pool while maintaining proportions.

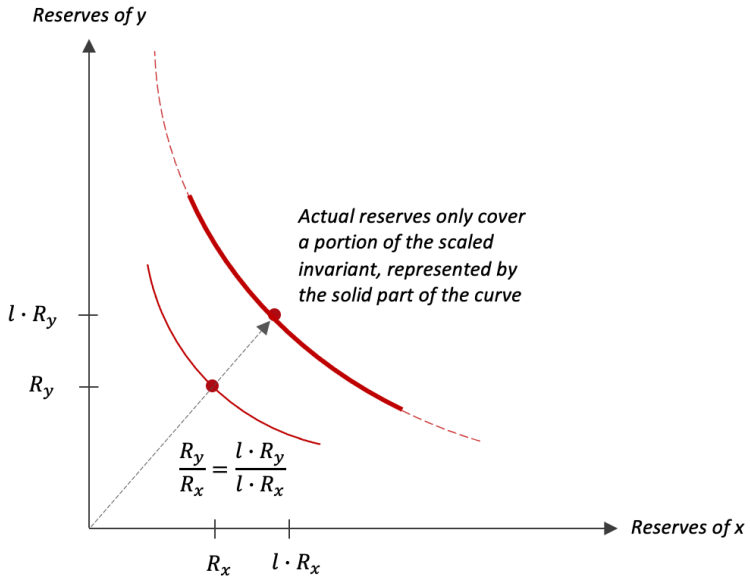

Fig. 1. Leverage preserves original price

Now consider the case where $l_x \neq l_y$, but $l_x \cdot l_y = 1$. In Figure 2, the product of leveraged reserves does not change vs the original, but their ratio does – we move to a different point on the same invariant, and thus alter the spot price.

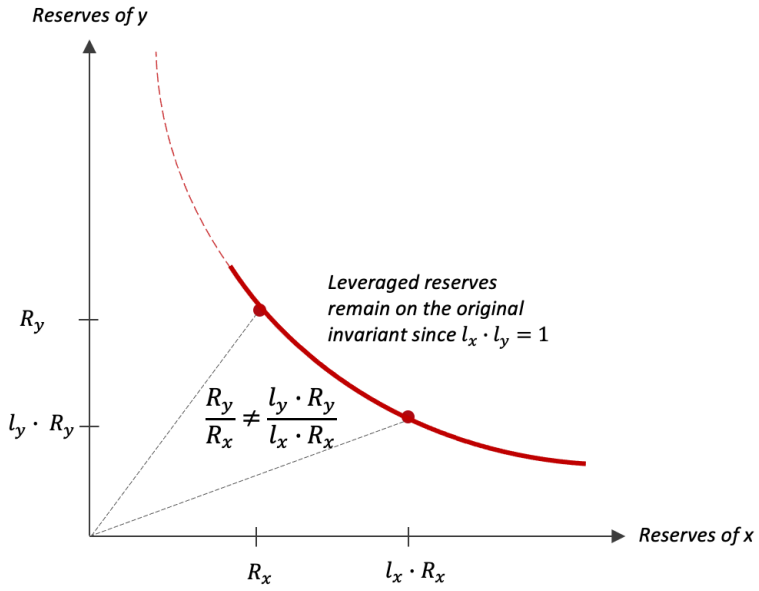

Fig. 2. Changing the price while staying on the original invariant

2.2 State Transition

In the previous section, we looked at the static situation where an AMM with reserves (R_i) and leverage coefficients (l_i) offers terms of trade for a single transaction. We now consider what happens after a transaction is performed.

For the original Uniswap invariant, this matter is relatively simple – reserves need to be updated to take into account the swapped amounts, but otherwise the invariant remains the same. With leverage however, we face an additional problem – once the traded amounts become part of the pool, leverage coefficients start to apply to them and the following inequality comes into play:

$$\prod_{i=1}^{n}(l_i \cdot R_i + \Delta_i) \neq \prod_{i=1}^{n} l_i \cdot (R_i + \Delta_i)$$

It is worth noting that there may be a number of interesting ways to evolve the terms of trade after each transaction. Below, we look at an approach that we believe preserves certain key properties of Uniswap and allows to extend many of the useful results already derived for the original invariant to our leveraged version.

More specifically, we would like our state transition rules to be such that, given a sequence of trades vs the leveraged AMM, one could achieve the same economic result with the same sequence of trades vs the original Uniswap AMM with some appropriate level of reserves.

Informally, such terms of trade would have two properties under zero fees: a) the new terms would remain on the same invariant curve as the previous ones; b) the spot price in the new terms would equal the marginal price of the final increment of the previous trade.

More formally, we want to find new leverage coefficients (l'_i) such that the following equations hold:

$$\prod_{i=1}^{n} l'_i \cdot (R_i + \Delta_i) = \prod_{i=1}^{n} (l_i \cdot R_i + \Delta_i)$$

$$\frac{l'_j \cdot (R_j + \Delta_j)}{l'_i \cdot (R_i + \Delta_i)} = \frac{l_j \cdot R_j + \Delta_j}{l_i \cdot R_i + \Delta_i}$$

This system of equations has the following solution:

$$l'_i = \frac{l_i \cdot R_i + \Delta_i}{R_i + \Delta_i}$$

Thus, after every trade, the leveraged AMM should re-calculate its leverage coefficients as per the formula above. Aside from this step and the already mentioned boundary condition with respect to asset availability, the leveraged AMM operates in this same way as the original Uniswap design and mimics it completely when $l_i = 1$.