Efficient Calculation of Pool Fees

El Ranchero

November 4, 2020

1 Introduction

AlpacaSwap is an evolution of automated market maker protocols that provides optimal liquidity by consolidating all assets into a single pool. This ensures that liquidity of popular assets is not fragmented across a number of pairs, creating the first viable, plug-and-play liquidity pool for DApps.

AlpacaSwap builds off of a number of existing protocols, with significant influence from Balancer and Uniswap, while introducing a number of important innovations. Among them is an improved governance model, where the protocol is truly owned and managed by the community from the start, without a pre-mine. A critical part of this model is the distribution of a portion of transaction fees to the PACA token holders to align long term incentives. While Uniswap has an implementation of fee distribution hard coded into their protocol, this trivial case does not generalize to a pool with many assets. Therefore, we had to implement a new fee distribution algorithm in AlpacaSwap, which is described in this paper.

Note: for continuity for those familiar with the precedent protocols, we try to stick with Balancer and Uniswap notation, even though it is somewhat clumsy.

2 Basics of Fee Distribution

As in all liquidity venues, each trade on AlpacaSwap incurs a fee charged to the liquidity taker, ϕ, in return for the protocol providing guaranteed liquidity at all times. This fee is initially set to 0.2%, but is configurable by governance.

There are two key constituents that enable AlpacaSwap to function: the current liquidity providers, who contribute the asset bases, and the PACA token holders, who govern the system. The current LPs are rewarded with PACA tokens over time, diluting the current PACA token holders to reflect the contribution of the LPs to the system.

To best align long-term incentives, AlpacaSwap splits the fee between these two constituent groups. Initially, the split is 75% to the PACA token holders (15 bps) and 25% to the LPs (5 bps) since the majority of economics at the start of the protocol will be in the PACA token rewards. However, this too is modifiable via governance.

Paying fees to the LP holders is quite easy: it can be done implicitly by adding the collected fees to the pool. However, paying out fees to the PACA token holders is a bit more complex. We don't want to have to pay out every transaction, which would be gas prohibitive, but it is challenging to track the total value of fees accumulated (and thus the amount owed to the PACA token holders) as the value of assets and the pool fluctuates.

The solution is to have some means of calculating and tracking the fraction of the pool that is attributable to fees at any point in time, and then periodically paying it out by issuing new LP shares to a collector contract when triggered by certain LP and governance actions. The fees then can be efficiently distributed to PACA holders from the collector contract on demand. An optional intermediate step, which is initially enabled in AlpacaSwap as an incentive alignment mechanism, is to convert the LP shares to PACA tokens prior to paying them to the collector contract.

Since we are tracking the fraction ownership of the pool, fees payout must be triggered anytime one of the following actions occurs:

1. Asset weights change (since formulae depend on weights)

2. Someone adds or removes liquidity from the pool (since this changes share supply)

3. Any fee variables are modified by governance

3 The Constant Mean Curve

AlpacaSwap uses a constant mean curve[1], which takes the form of the following invariant (using Balancer notation):

$$K = \prod_k B_k^{w_k}$$

where k is the asset in the pool, B_k the balance of that asset in the pool, and w_k is the weight of the asset in the pool. This invariant can be interpreted as a weighted geometric mean, hence the name of the curve. The core principle of the AMM is that any trade must ensure that this invariant holds, modulo any fees collected. In reality, K increases over time (henceforth K_t) as fees accumulate, and K_t changes as LPs add or remove liquidity from the pool.

4 Naive Method: Iterative Update

Let's assume that at time $t = 0$, the pool has $n \geq 2$ assets with balances $B_{k,0} = B_k$ and a invariant of K_0. At $t = 0$, a user trades in Δ_i of asset i and receives out Δ_o of asset o, advancing time to $t = 1$. Per the protocol standard, the fee is applied on the asset in ahead of the trade, meaning that $(1 - \phi)\Delta_i$ is the amount in for the purposes of the invariant calculation. Then, by the invariant, we can solve for the amount out:

[1]For more information, see https://arxiv.org/abs/2003.10001

$$
\begin{aligned}
K_0 = \prod_k B_k^{w_k} &= K_1 \\
\prod_{k \neq i,o} (B_k)^{w_k} \cdot B_i^{w_i} \cdot B_o^{w_o} &= \prod_{k \neq i,o} (B_k)^{w_k} \cdot (B_i + (1-\phi)\Delta_i)^{w_i} (B_o + \Delta_o)^{w_o} \\
B_i^{w_i} B_o^{w_o} &= (B_i + (1-\phi)\Delta_i)^{w_i} (B_o + \Delta_o)^{w_o} \\
\Delta_o &= B_o \cdot \frac{B_i^{w_i/w_o}}{(B_i + (1-\phi)\Delta_i)^{w_i/w_o}} - B_o \\
&= B_o \left(\frac{B_i^{w_i/w_o}}{(B_i + (1-\phi)\Delta_i)^{w_i/w_o}} - 1 \right) \\
B_o + \Delta_o &= B_o \left(\frac{B_i}{B_i + (1-\phi)\Delta_i} \right)^{w_i/w_o}
\end{aligned}
$$

Intuitively, this means that the amount of asset o after the trade in the pool is the initial amount, B_o, scaled by the weighted change in asset i. To solve for K_1, we note that the full Δ_i is added to the pool:

$$
\begin{aligned}
K_1 &= \prod_{k \neq i,o} (B_k)^{w_k} \cdot (B_i + \Delta_i)^{w_i} (B_o + \Delta_o)^{w_o} \\
&= \prod_{k \neq i,o} (B_k)^{w_k} \cdot (B_i + \Delta_i)^{w_i} \left[B_o^{w_o} \left(\frac{B_i}{B_i + (1-\phi)\Delta_i} \right)^{w_i} \right] \\
&= \left[\prod_k B_k^{w_k} \right] \left(\frac{(B_i + \Delta_i)}{B_i + (1-\phi)\Delta_i} \right)^{w_i} \\
&= K_0 \left(\frac{(B_i + \Delta_i)}{B_i + (1-\phi)\Delta_i} \right)^{w_i} \\
&= K_0 \left(\frac{(B_i + \Delta_i)}{B_i + (1-\phi)\Delta_i} \right)^{w_i}
\end{aligned}
$$

Solving for the value of the fee between $t = 0$ and $t = 1$:

$$\begin{aligned}
\phi\Delta_i &= (B_i + \Delta_i)\left(1 - \left(\frac{K_0}{K_1}\right)^{1/w_i}\right) \\
&= B_{i,1}\left(1 - \left(\frac{K_0}{K_1}\right)^{1/w_i}\right)
\end{aligned}$$

Let $V_t(\cdot)$ indicate the value function at time t in terms of the chosen numeraire asset ν and P^i_ν be the price of asset i in terms of the numeraire asset. Then the value of that fee is:

$$\begin{aligned}
V_1(\phi\Delta_i) &= \phi\Delta_i \cdot P^i_\nu \\
&= B_{i,1}\left(1 - \left(\frac{K_0}{K_1}\right)^{1/w_i}\right) \cdot P^i_\nu \\
&= B_{i,1}\left(1 - \left(\frac{K_0}{K_1}\right)^{1/w_i}\right) \cdot \frac{B_\nu / w_\nu}{B_i / w_i} \\
&= B_{\nu,1}\frac{w_i}{w_\nu}\left(1 - \left(\frac{K_0}{K_1}\right)^{1/w_i}\right)
\end{aligned}$$

While the value of the pool, $V_t(P)$, at that time in terms of the numeraire is the amount of numeraire divided by its weight:

$$V_1(P) = \frac{B_{\nu,1}}{w_\nu}$$

Let the fraction of the portfolio attributable to the fees at time T from the transaction that advanced time to t be $F_{T,t}$, which equals at $T = t = 1$:

$$
\begin{aligned}
F_{1,1} &= \frac{V_1(\phi\Delta_i)}{V_1(P)} \\
&= \frac{B_{\nu,1}\frac{w_i}{w_\nu}\left(1-\left(\frac{K_0}{K_1}\right)^{1/w_i}\right)}{\frac{B_{\nu,1}}{w_\nu}} \\
&= w_i\left(1-\left(\frac{K_0}{K_1}\right)^{1/w_i}\right)
\end{aligned}
$$

which is independent of the choice of numeraire.

Each time there is a new trade, the value of each previous F gets diluted by the fraction of the pool owned by the latest fee. That is:

$$F_{t,t+1} = F_{t,t}(1 - F_{t+1,t+1})$$

The total value attributed to fees at any time, G_t is then:

$$G_t = \sum_{j=0}^{t} F_{j,t} = \sum_{j=0}^{t}\left(F_{j,j}\prod_{k=j+1}^{t}(1 - F_k, k)\right)$$

There is no simple closed form solution to this equation, so the naive method is to keep a running value of G_t, updated each transaction as follows:

$$G_{t+1} = G_t(1 - F_{t+1,t+1}) + F_{t+1,t+1}$$

Instead of using K to calculate the fee value, we instead can solve for it in terms of the other variables to make calculation simpler:

$$\begin{aligned}
F_{1,1} &= w_i\left(1-\left(\frac{K_0}{K_1}\right)^{1/w_i}\right)\\
&= w_i\left(1-\left(\frac{K_0}{K_0\left(\frac{(B_{i,0}+\Delta_i)}{B_{i,0}+(1-\phi)\Delta_i}\right)^{w_i}}\right)^{1/w_i}\right)\\
&= w_i\left(1-\frac{B_{i,0}+(1-\phi)\Delta_i}{B_{i,0}+\Delta_i}\right)\\
&= w_i\left(1-\frac{[B_{i,0}+\Delta_i]-\phi\Delta_i}{B_{i,0}+\Delta_i}\right)\\
&= w_i\left(1-1+\frac{\phi\Delta_i}{B_{i,1}}\right)\\
&= w_i\frac{\phi\Delta_i}{B_{i,1}}
\end{aligned}$$

This makes intuitive sense as the fee value in terms of asset i divided by the post-trade pool value in terms of asset i.

For the initial version of Alpaca, we wanted to keep the code simple to make it easy for people to understand, so we employ this naive method.

5 Optimized Method

The downside of the naive method is that it requires several extra computations per trade, which increases gas cost. Luckily, we find a closed-form approximation of the accumulated fee value (as a fraction of total pool value) using some mathematical tricks.

For the optimized method, we return to the expression of $F_{t,t}$ in terms of K:

$$F_{t,t} = w_i\left(1-\left(\frac{K_{t-1}}{K_t}\right)^{1/w_i}\right)$$

We note that fees should be relatively small as a fraction of total pool size, meaning that K_1 will be pretty close to K_0. We therefore let $x = \frac{K_{t-1}}{K_t} \approx 1 = a$, and restate the previous function as

$$f(x) = 1 - x^{1/w_i}$$

The derivative of this function is

$$f'(x) = -\frac{1}{w_i} x^{(1/w_i - 1)} = -\frac{1}{w_i} x^{\frac{1-w_i}{w_i}}$$

We can use a Taylor approximation of the function for neighbourhoods of a

$$\begin{aligned} f(x) &\approx f(a) + f'(a)(x-a) + o\left((x-a)^2\right) \\ &= 0 - \frac{1}{w_i}(x-1) &&= \frac{1}{w_i}(1-x) \end{aligned}$$

Plugging back in for $F_{t,t}$:

$$F_{t,t} = w_i \cdot f(x) \approx w_i \cdot \frac{1}{w_i}(1-x) = 1 - x = 1 - \frac{K_{t-1}}{K_t}$$

Now we need to apply this approximation to the total accumulated fees, G_t. To do so, we will use a proof by induction. Assume that the total fee value is given by

$$G_t = 1 - \frac{K_0}{K_t} \quad \forall t > 0$$

We show that this expression also holds for G_{t+1}:

$$\begin{aligned} G_{t+1} &= G_t(1 - F_{t+1,t+1}) + F_{t+1,t+1} \\ &= \left(1 - \frac{K_0}{K_t}\right)\left(1 - \left[1 - \frac{K_t}{K_{t+1}}\right]\right) + \left(1 - \frac{K_t}{K_{t+1}}\right) \\ &= \left(\frac{K_t}{K_{t+1}} - \frac{K_0}{K_{t+1}}\right) + \left(1 - \frac{K_t}{K_{t+1}}\right) \\ &= 1 - \frac{K_0}{K_{t+1}} \end{aligned}$$

Our base case for $t = 1$ also holds:

$$G_1 = F_{1,1} = 1 - \frac{K_0}{K_1}$$

Therefore, the expression holds in general:

$$\boxed{G_t = 1 - \frac{K_0}{K_t}}$$

This has the nice property that it does not need to be actively tracked. Instead, it can be calculated on the fly whenever a particular trigger event occurs. The calculation of K at any point would require additional safe math libraries (for exponentiation / logarithms), so we will hold off on including it until a future version in favor of simplicity for the current version.

5.1 A Trivial Example: Uniswap

Uniswap is the trivial case where $n = 2$ assets. The standard Uniswap curve expression is slightly different in that it is actually the square of the general constant product invariant that Alpaca uses:

$$k = B_x \cdot B_y$$

where $k = K^2$.

Substituting in $\sqrt{k}$ for K, we get:

$$G_t = 1 - \frac{\sqrt{k_0}}{\sqrt{k_t}}$$

which matches equation (4) of the Uniswap V2 whitepaper[2].

6 Share Dilution Calculation

Now that we know how to calculate the fee amount itself, we now turn to how the fee is paid out via LP share issuance. In particular, we need to calculate how many shares should be printed and distributed to the PACA token holders (remember, the fees given to LPs is implicit in the growth of the pool value). We introduce some new notation to do so:

[2] https://uniswap.org/whitepaper.pdf

$$\begin{aligned}
s_t &= \text{number of LP shares at time } t \\
\sigma_t &= \text{number of LP shares issued to fee recipient } c \text{ at distribution time } t \\
\lambda_c &= \text{fraction of fees owed to the fee recipient } c \\
s_{t_2} &= s_{t_1} + \sigma_{t_2}
\end{aligned}$$

Then, the share of the pool owed to c is

$$\lambda_c G_{t_1,t_2} = \frac{\sigma_{t_2}}{s_{t_2}} = \frac{\sigma_{t_2}}{s_{t_1} + \sigma_{t_2}}$$

Solving for σ_{t_2}:

$$\begin{aligned}
\sigma_{t_2} &= \boxed{\frac{\lambda_c G_{t_1,t_2} s_{t_1}}{1 - \lambda_c G_{t_1,t_2}}} \\
&= \frac{\lambda_c \left(1 - \frac{K_{t_1}}{K_{t_2}}\right) s_{t_1}}{1 - \lambda_c \left(1 - \frac{K_{t_1}}{K_{t_2}}\right)} && \text{multiply by } \frac{K_{t_2}}{K_{t_2}} \\
&= s_{t_1} \cdot \frac{\lambda_c (K_{t_2} - K_{t_1})}{K_{t_2} - \lambda_c (K_{t_2} - K_{t_1})} && \text{multiply by } \frac{1/\lambda_c}{1/\lambda_c} \\
&= \boxed{s_{t_1} \cdot \frac{K_{t_2} - K_{t_1}}{\left(\frac{1}{\lambda_c} - 1\right) K_{t_2} + K_{t_1}}}
\end{aligned}$$

where the first boxed formula would be used with the naive method where G_{t_1,t_2} is tracked, while the second boxed formula would be used in with the optimized approximation method. Note that this matches the trivial case solved for in formula (6) of the Uniswap whitepaper using the same substitution as above.

Whenever a trigger event occurs, σ_t is calculated and distributed to the recipients, and the tracking system is reset. There could be multiple external recipients of these fees, but for now Alpaca only supports one - the PACA holders.