Main goals of the implementation

- Should be very easy to share liquidity with as few direct dependencies between L2s
- Deterministic so AMM operations never have to be reverted because of unexpected state changes
- Aim for IL comparable to a Uniswap V2 style AMM

Context - AMM, Amplified AMM & Impermanent Loss

AMM

The standard AMM curve with 2 assets X

and Y

with respective balances x

and y

follows x*y = k

.

When ignoring trading fees, k

is invariant when trading against the AMM.

x*y = k

When adding or removing liquidity, the balances change, and thus k

increases or decreases.

Because k

is invariant, totalSupply = \sqrt{k}

can directly be used to represent the (linear) size of the AMM pool regardless of the current price in the AMM, with Liquidity Providers (LPs) gaining or losing \sqrt{\Delta k}

of shares of the total supply when adding or removing liquidity

\sqrt{\Delta k} = \sqrt{amountX*amountY}

with amountX

and amountY

being the amounts added or removed from the AMM.

Amplified AMM

Let’s consider Amplified AMM. An amplified AMM is an AMM that operates on virtual balances, as opposed to UniswapV2 AMM that operates on current balances. From the LPs perspective, the main difference lies in the Impermanent Loss (IL

) risk. More on IL

on this link.

We can define the amplificationFactor

(AF)

= \frac{\sqrt{k}}{\sqrt{balanceX*balanceY}}

as a parameter defining the factor between the virtual balances and the current balances.

- amplificationFactor == 1 \rightarrow

dAMM IL == Uniswap V2 IL

- amplificationFactor < 1 \rightarrow

dAMM IL < Uniswap V2 IL

- amplificationFactor > 1 \rightarrow

dAMM IL > Uniswap V2 IL

For the rest of the document, we will refer to Amplified AMM as an AMM with amplificationFactor \geq 1

.

dAMM, a cross-L2 AMM

In a few words, dAMM is a cross-L2 AMM that differs from existing AMMs by having multiple virtual balances, one set per L2. More details are available on this post and this thread.

One can notice that if the same liquidity is shared across multiple AMMs simultaneously, the liquidity is effectively an amplified pool.

This means that while the Amplified pool (defined by k =

AF*AF*x*y

) k

remains constant, the Uniswap V2 (k = x*y

) equivalent pool value decreases

.

To maintain parity between the virtual balances and true balances, the pool must receive additional funds. These additional funds should come from the fees generated by trading.

Note that it is not necessary to create an amplified pool to share liquidity, although it does allow the funds to be used most efficiently. The pool funds can also be divided between the different L2s (either using an on-chain mechanism by assigning different curves) or by agreeing on the splitting off-chain and having safety checks on-chain.

There is only additional IL when AF > 1

pools when the price actually moves. For StableSwap pools, IL risk is extremely limited, therefore dAMM can be used out of the box. But for other pools, we need to mitigate IL

. This is where the dynamic fees come in. Kyber's DMM (https://files.kyber.network/DMM-Feb21.pdf, 3.2 Dynamic fees) proposes modifying the fee based on volume.

However, we can do better under dAMM. By turning the amplificationFactor

into an healthFactor

, dAMM pools no longer have to naively suffer amplified IL

.

To do so, we let L2s communicate in some aspect what their latest state is. With this additional information, we can calculate the healthFactor

in real-time so fees can be adjusted immediately instead of waiting on the on-chain state. So to recap, fees can be adjusted based on:

- Volume
- Health Factor (either delayed on-chain information or real-time off-chain information)

Minimal implementation

There's a pool of funds that each L2 can use. LPs deposit/withdraw funds to the contract and L2s use the funds to facilitate trades in any way they want within the safety checks enforced on-chain, most notably a check that ensures the Health Factor remains above a certain fixed level.

We also want the balances in the pool to not just follow k

, we also want the balances to actually track the correct ratio at the real-time price pretty closely. This can either be done by:

1. Storing the tokens in the LP token of an L1 AMM
2. Allowing trading the balances if they don't track the price close enough (though, the L2 AMMs should already push the balances roughly in the correct ratio).

Note that option #1

increases the amplificationFactor

by one.

Fees will be collected by the L2s and used for both:

- Reducing the IL

by "fixing" the curve (bringing the curve defined by the actual balances to the target curve)

- Paying the LPs fees fairly between AMMs

The above minimal implementation has some drawbacks as very few direct limitations are enforced to how the funds are used and we can only enforce some basic Health Factor checks.

There is also no direct way funds can be given to a specific AMM without just giving a blank loan. This can be seen as something positive although with a bit of extra complexity we can enforce some more things on-chain.

More advanced implementation

This implementation builds on top of the minimal implementation so e.g. also contains checks on the healthFactor

.

We want to have a virtual "global" curve to rule all funds.

To achieve this, we keep track of the state of all AMMs in real-time. At any given point, each AMM on each L2 is independent. The AMM runs its own curve at some local price, and L2 joins and exits are done against this curve. The net difference of join/exit is then checkpointed on L1.

To protect the LPs even more, the L2 would be restricted by its L1 curve on top of the global healthFactor

.

Once in a while, each L2 must checkpoint on L1, where all restrictions (curve + healthFactor

) are enforced. This checkpoint would either trigger a swap using the assets directly on the contract or if funds are directly available in the L2, only the latest local curve and balances would be passed to the contract. If the healthFactor

falls below a certain level all check-ins need to improve the health factor until the healthFactor

is sufficiently high again, otherwise the check-in is rejected.

Because k

should be kept constant for trades, we know any change to k

is caused by joins/exits which are used to update the global curve. The reverse is also true, the local curve is updated with the latest known global state which contains joins/exits of the other AMMs.

Note that generally, the local AMMs will only roughly track the actual global state because joins/exits will first be done on a single AMM before the other AMMs pick these changes up when syncing with the dAMM contract. This means that for exits if we are in the case where the global AF

1, the effective AF

will actually temporarily increase because the other AMMs are still working on a curve before that exit. This should not be a problem in practice because the dynamic fees should directly take into account this decrease and if needed we can minimize the amount that can be exited within a period. If the global AF = 1

then the effective AF

actually remains the same.

We keep track of which liquidity originated from which AMM. This means that each AMM is able to control its share of funds i.e. if funds were added on a certain AMM it can also only be withdrawn by that AMM. Each L2 can mint/burn LP tokens corresponding to their increase/decrease in k

.

Conclusion

We try to express how dAMM IL can be mitigated by modulating the fees and enforcing a strong global variable called the healthFactor

, measuring directly the IL

suffered by dAMM LPs.

We have some simulations available here and a more detailed implementation should be released in the future.

This is an attempt at solving liquidity fragmentation in a multi-L2s world. Happy to hear your comments and improvement proposals.

Brecht Devos & Louis Guthmann