Introduction

The CoW AMM design was launched a few months ago by the CoW DAO as an alternative AMM design that mitigates many of the problems known in the existing and widely used AMMs today.

To initiate the launch, a simple framework was set up that is generating programmatically CoW AMM orders every few minutes for the existing CoW AMMs based on some oracle prices; this is what we call Phase 1. However, to take full advantage of the power and potential of CoW AMMs, a more flexible and dynamic way to accessing their liquidity is needed. Specifically, solvers should be allowed to trade with the CoW AMM in any batch auction they want to and with amounts determined by them; this is what we call Phase 2.

The core backend team has addressed this by setting up the infrastructure so that solvers can indeed fully control the ways the trade with a CoW AMM. The weekly release on May 21, 2024, thus, will initiate Phase 2 of CoW AMM, turning it into a full-fledged function maximizing AMM.

To incentivize solvers to integrate CoW AMMs as soon as possible in this Phase 2, we propose to set up a bounty of total value 10k DAI, split into three prizes (5k, 3k and 2k DAI) and financed by our grant program.

We now provide some details around CoW AMMs and then proceed to define the terms and rules associated with the proposed bounty.

Function maximizing AMM

In Phase 1 of CoW AMMs, an oracle was used to add orders for those AMMs to the orderbook. While this is relatively close to the description of CoW AMM in this paper, it does restrict the solvers in executing the intent of Cow AMMs. Phase 2 of the project is meant to unleash the full potential of CoW AMMs and turn them into a fully functioning function maximizing AMM.

In phase 2, Instead of adding individual orders to the order book on behalf of a CoW AMM, solvers can freely propose trades for those AMMs. Those trades are of very similar form as normal user trades as they require to specify traded amounts and a fee. Unlike regular orders, solvers can choose which of the two tokens is the buy and which is the sell token.

There is a hard restriction enforced on the smart contract of not violating the constant product invariant of CoW AMMs. Providing better executions to a CoW AMM is rewarded similarly to how it is rewarded for user orders, via surplus which enters ranking and rewards.

The surplus for a CoW AMM is computed conceptually differently compared to user orders. A CoW AMM for tokens A and B with reserves X and Y, respectively, and buying x of token A for y of token B will have a surplus of

$$x - y X / Y - x y / Y,$$

naturally expressed in token A. This is just the change in the constant product invariant,

$$(X + x) (Y - y) - X Y,$$

rescaled by Y. When batching a trade for a CoW AMM together with users orders, the surplus of the CoW AMM is added together with the surplus of user orders, as usual.

CoW AMMs do pay network fees as normal user orders do and settle at uniform clearing prices and uniform directional prices. CoW AMMs can be used in CoWs with other orders or traded with external liquidity, but there can ever be at most one order for the same CoW AMM in a batch.

Implementation via JIT orders

Some details can be found in thedocumentation.

- CoW AMM orders are included in the solutions as JitOrders.
- If AMM should buy x for y, the JitOrder needs to be a fill-or-kill sell order with sell amount y and limit buy amount $y * X / Y + x * y / Y$. Then, the usual surplus for that order corresponds to the nonlinear surplus defined above. The reference driver does not check for correctly set limit amounts. Deviating from the limit amounts defined above amounts to lying about the surplus of your solution.
- Generate a suitable signature and the order uid as described in the documentation.
- Appropriate pre- and post-interactions need to be included in the new preInteractions

and postInteractions

field of the solution. This is described in the linked documentation as well.

For every distinct CoW AMM there can be at most a single order for it in a batch. This also applies to the CoW AMM orders that are already included in the orderbook: if a JIT order is used in a batch, then the order in the orderbook must be ignored.

Bounty specifications

- Settle 20 orders for CoW AMMs satisfying the following properties:
- The order is created as a JIT order.
- The order must receive more surplus than executing the automatically generated order.
- The order is created as a JIT order.
- The order must receive more surplus than executing the automatically generated order.
- The first three solvers satisfying the above are rewarded a bounty.
- 5k DAI for the first solver.
- 3k DAI for the second solver.
- 2k DAI for the third solver.
- 5k DAI for the first solver.
- 3k DAI for the second solver.
- 2k DAI for the third solver.

Authors: @harisang , @AndreaC , @felixhenneke