AsCSMM: A Peer-to-Peer Async Swap Constant Sum Market Maker For Uniswap V4 Pools (WIP)

Msaki, Meek
meek10x@gmail.com

Li, Jiasun
Jli29@gmu.edu

May 4, 2025

Abstract

We implement an async swap constant-sum market maker that bypasses both the swap logic in Uniswap V4's constant-product market maker (CPMM) and the concentrated liquidity tick pricing for asset pools, in favor of custom accounting. Our async market maker pauses transactions in the block they are submitted, allowing any user to manually fill the order at a later, non-deterministic time. This implementation removes the need for liquidity providers (LPs), as each async swap order is filled on demand in a peer-to-peer manner.

1 Introduction

The Uniswap V4 [2] protocol is an automated market maker (AMM) in DeFi. It functions similarly to its predecessor, Uniswap V3 [1], where it features a constant-product market maker (CPMM) whose asset prices are calculated using a tick pricing.

A key feature of Uniswap V4 is the ability to attach a Hooks contract during pool initialization. This contract enables custom logic to be applied to swaps, liquidity changes, and donations. We leverage this mechanism to override default behaviors and implement an async swap constant-sum market maker (AsCSMM).

2 Async Swap Hook

Our Hooks contract implements custom logic in the following hook callbacks:

- `beforeSwap`
- `beforeSwapReturnsDelta`

In our `beforeSwap` function, we first take the user-specified amount from the Pool Manager and emit an async swap order event. We then record a 1:1 claimable amount of the user's asset within our hook contract. Execution is returned to Uniswap V4's Pool Manager, after which the user, typically interacting through a router, sends the asset they want to swap to the Pool Manager. This transfer settles the delta balance created by our hook. At this point, the async swap is recorded, an async order event has been emitted, and the order is ready to be filled in a future block.

Our implementation is able to create async swaps because, whenever we return execution to the pool manager, we always specify a zero swap value, which will cause the Uniswap V4 swap logic to be bypassed. At the end of the transaction, the user's specified swapped asset has decreased by an amount they specified, and our hook's asset balance has increased by an amount the user specified.

With this async swap hook, we can implement custom accounting during the filling of swap intents. One such feature is the constant-sum market maker (CSMM).

3 Filling Async Orders

A user filling async orders, referred to as the filler, must provide a valid swap order from previous swap events. By specifying this previous swap order to our hook contract, the order's asset will be sent to the filler. The filler must then supply an equivalent amount of the desired asset requested by the user who submitted the order.

Since any user can fill previous orders, we have introduced a peer-to-peer order-filling process that eliminates the need for liquidity providers (LPs).

4 Fees

As previously noted, our implementation bypasses Uniswap V4's swap logic and removes the need for liquidity providers. This means users pay no protocol fees, experience zero slippage, and are protected from block-level MEV, as active liquidity is not supported.

Currently, our ASMM charges zero fees for both submitting and filling orders.

5 Future work

Our implementation enables permissionless order filling, potentially sparking competition among fillers through arbitrage opportunities created when interacting with intents across other DeFi protocols. While the impact remains uncertain, we welcome technical mentorship to investigate further. We are seeking capital to reach audit readiness, and our future work remains a work in progress (WIP).

References

[1] Hayden Adams et al. *Uniswap v3 Core*. Mar. 2021. URL: https://uniswap.org/whitepaper-v3.pdf.

[2] Hayden Adams et al. *Uniswap v4 Core*. Aug. 2024. URL: https://github.com/Uniswap/v4-core/blob/main/docs/whitepaper/whitepaper-v4.pdf.