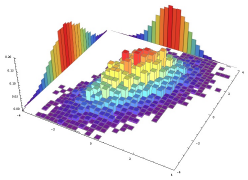
Marginal Economic Simulations

SmolQuants

March 2024



Contents

1 Introduction

Marginal is a decentralized, permissionless, peer-to-pool AMM for physically settled perpetuals on any pair of fungible tokens.

SmolQuants was engaged by Marginal for their V1 to:

- Backtest the performance of liquidity providers (LPs) in Marginal V1 pools using historical data from Uniswap V3.
- Stress test the claim of no bad debt to LPs even if leverage positions are liquidated below their bankruptcy price, given the insurance mechanism of the Marginal pool.

2 Summary

- Backtests were performed against Marginal V1 smart contracts for a range of simulation parameters using historical data from Ethereum mainnet Uniswap V3 USDC/ETH 5bps and UNI/ETH 30bps pools. Simulations were backtested from block 17998181 (Aug-26-2023) to 19311400 (Feb-26-2024) in steps of 2400 blocks.
- Simulation parameters included Marginal V1 pool utilization for leveraged positions, skew in the fraction of total liquidity used for shorts versus for longs, number of blocks a position was held before settling, and the position leverage taken.
- At each step of the simulation, actors arbitraged the Marginal and Uniswap pools to keep the Marginal pool price in line with its reference, liquidated existing unsafe leveraged positions, settled any outstanding leveraged positions held beyond their turnover age, opened new leveraged positions to maintain the specified pool utilization, and performed swaps to simulate fee volume.
- The Marginal V1 pool insurance mechanism worked as intended, particularly in the most extreme scenarios. Liquidity providers gained liquidity from liquidations in all cases simulated, even in the case when a position was liquidated below the bankruptcy price.
- Marginal LP performance exceeded that of a Uniswap V2 pool, given Marginal LPs are rewarded with trader margin upon position liquidation in addition to the usual swap fees. Albeit, with the caveat that our swap fee simulation is rather crude and likely unreliable in practice. Though the general trend for Marginal LPs from our simulations: expect providing liquidity to significantly outperform Uniswap V2 when traders are more leveraged with their positions on the pool.

3 Background

Marginal V1 enables leverage trading on any pair of fungible tokens by allowing traders to borrow liquidity directly from the pool to collateralize their leveraged positions. Liquidity providers (LPs) pool both X and Y tokens as passive liquidity, similar to Uniswap V2. If a trader wishes to open a leveraged long X relative to Y position, the pool internally

- Removes δL of liquidity from available liquid reserves
- Sets aside a fraction of that removed liquidity (i_x, i_y) as insurance
- Swaps the remaining Y removed reserves through the new pool liquidity curve at $L - \delta L$ for more X
- Sets aside the total X tokens plus insurance to back the position

The trader sends in c_x of margin as collateral for their newly opened position.

At settlement, the trader receives the set aside X tokens (i.e. physical settlement) and their original margin in exchange for delivering the Y debt owed that returns at least the original liquidity δL removed back to the pool. Similar to lending protocols like Aave, a fraction of LP passive liquidity becomes illiquid as the pool becomes more utilized for leveraged positions.

To keep the position open, the trader must manage minimum margin requirements

$$(c_x + s_x) \cdot \bar{P}_t \geq (1 + M) \cdot d_y$$

where s_x is the net position size in X token to be received by the trader at settlement, d_y is the Y debt to be delivered, M is the minimum maintenance requirement, and $\bar{P}_t$ is the time-weighted average price fetched from the Uniswap V3 oracle referenced by the Marginal pool. If not satisfied, liquidators can call a function that liquidates the position, sending the X tokens set aside, the trader's margin, and the insurance reserves back to the pool. Thereby, increasing available pool liquidity and decreasing price.

Insurance reserves backing the position (i_x, i_y) are set such that at least the originally removed liquidity δL is returned to the pool after position liquidation regardless of the Marginal pool price prior. This should deterministically ensure safety for LPs, irrespective of the speed at which liquidations happen. However, does this insurance mechanism work in practice?

4 Backtesting

Backtests were performed using the SmolQuants `backtest-ape` package, which is built on the ApeWorX `ape` framework. To backtest, `backtest-ape` begins by forking the chain at a user-specified historical start block. It then deploys duplicate mock contracts, which are updated at each block of the simulation to what the state of a set of reference contracts (e.g. Uniswap V3 pool) had been at that historical block. A user-implemented runner instance submits transactions to a separate `Backtest.sol` contract deployed on the forked chain to implement their strategy on these mock contracts. Over each block of the simulation, relevant user-defined values (e.g. total liquidity associated with an LP position) are queried from the `Backtest.sol::values` function to generate a timeseries of historical data. At the end of each block iteration, the runner can update its strategy based off of the current state of the fork by sending additional transactions through `Backtest.sol`.

For Marginal V1 economic simulations, we used historical data on Uniswap V3 pools and implemented one type of runner. The `MarginalV1LPRunner` runner class creates a hypothetical LP position on a deployment of the exact `MarginalV1Pool.sol` pool instance. The Marginal V1 pool references a mock of the Uniswap V3 pool as its oracle.

The user specifies:

- `refs["univ3_pool"]`: Uniswap V3 pool to reference.
- `maintenance`: The minimum maintenance requirement of the Marginal V1 pool.
- `liquidity`: The initial amount of liquidity L to provide to the Marginal V1 pool.
- `utilization`: The fraction of total liquidity in the Marginal V1 pool used for leverage positions.
- `skew`: The fraction of total liquidity used for shorts versus for longs $\in [-1, 1]$. -1 is all utilization long, +1 is all short.
- `blocks_held`: The number of blocks to hold a Marginal V1 leverage position for before settling.
- `sqrt_price_tol`: The relative difference in `sqrtPriceX96` values between the Marginal V1 and Uniswap V3 pools above which the runner should arbitrage the pools.

They can also choose to specify either of the following two parameters to determine the margin backing each Marginal V1 leverage position:

- `leverage`: The leverage to use for each leverage position taken out on the Marginal V1 pool.
- `rel_margin_above_safe_min`: The relative buffer in margin above the safe margin minimum for each leverage position taken out on the Marginal V1 pool. The position is liquidated by the runner when its margin is below the safe margin minimum, making it unsafe.

For each block iteration of the backtest, the runner sets the state of the mock Uniswap V3 pool contract to the state of the V3 reference pool's values for `slot0`, `liquidity`, `feeGrowthGlobal{0,1}X128` at the number of the block iteration.

Additionally, the runner sets the `tickCumulatives` accumulator snapshots in the mock Uniswap pool's observations array to allow the Marginal V1 pool to accurately use the oracle in the backtest, as it would have historically. The last two observations of the Uniswap mock are set to the accumulator values returned from the reference V3 pool's oracle observations array with input `[43200, 0]`, for tick cumulatives 12 hours

before and 0 seconds before the iteration timestamp, given the `secondsAgo` constant on the Marginal V1 pool.

The runner then updates the state of the Marginal V1 pool by

- Arbitraging the Marginal V1 pool and the mock Uniswap V3 mock pool via flash swapping for profit, if the relative difference in `sqrtPriceX96` between the two pools is larger than `sqrt_price_tol`.
- Liquidating any leveraged positions on the pool that have become unsafe since the last iteration due to changes in the Uniswap V3 TWAP oracle averaged over 12 hours for the Marginal V1 pool.
- Settling any leveraged positions on the pool that were opened more than `blocks_held` blocks ago.
- Opening new long and/or short leveraged positions on the pool to maintain liquidity locked of `utilization * totalLiquidity`. The proportion of shorts to longs is determined by the `skew` parameter and follows: $S = (a - b)/(a + b)$, where a is the liquidity locked for shorts and b for longs.
- Simulating (roughly) accumulated swap fees on the Marginal V1 pool by swapping back and forth the size needed to generate the global swap fees accumulated on the reference Uniswap V3 pool, scaled down by the relative difference in liquidity between the two pools: `feeAmountMarginal = feeAmountUniswap * liquidityMarginal / liquidityUniswap` .
- Arbitraging the pools again if the price has deviated enough since the beginning of the update.

Backtests were simulated from block 17998181 (Aug-26-2023) to 19311400 (Feb-26-2024) in steps of 2400 blocks ($\sim$ 8 hours) on the Ethereum mainnet Uniswap V3 USDC/ETH 5bps pool and the UNI/ETH 30bps pool with ETH as the quote currency. Shorts on the UNI/ETH 30bps pool provided a good stress test of the no bad debt on liquidations claim for Marginal V1 pools, given the nearly 67% increase in price in a single hour on Feb-23-2024:

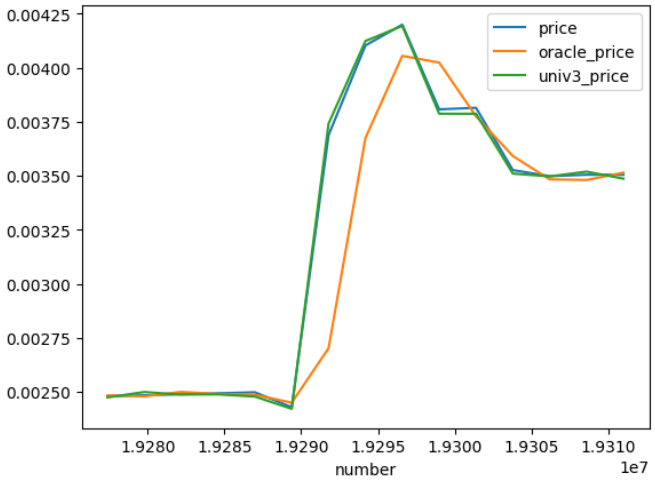


Figure 1: Price of UNI/ETH around Feb-23-2024 on the Uniswap V3 reference pool (green), the associated 12 hr oracle TWAP (orange) calculated from the Uniswap V3 reference pool, and the corresponding Marginal V1 pool price (blue) *after* arbitrage occurs in the runner updates. Square root prices arbitraged to within 50 bps for UNI/ETH simulations.

On the UNI/ETH 30bps pool, we simulated over multiple variations of runner parameters

- `maintenance = 250000` (in units of `1e6`)
- `liquidity = 10226805916623746957691` ($\sim$ 1110.56 ETH total value)
- `utilization = [0.25, 0.5]`
- `skew = [0.25, 0.5]`
- `blocks_held = [3600, 50400]`

- `sqrt_price_tol = 0.005`
- `leverage = 3.5`

Similarly, on the USDC/ETH 5bps pool

- `maintenance = [250000, 1000000]` (in units of `1e6`)
- `liquidity = 91287092917527680` ($\sim$ 4497.59 ETH total value)
- `utilization = 0.25`
- `skew = [-0.5, 0.5, 0.9]`
- `blocks_held = 50400`
- `sqrt_price_tol = 0.0025`

Notebook to generate the plots provided in `backtest.ipynb`.

For the report, we focus on the UNI/ETH run with 25% pool utilization, 25% skew short, 50400 blocks held (7 days between settlement) and leverage of 3.5x as a stress test of the no bad debt on liquidations claim, given the large hourly candle toward the end of the simulation.

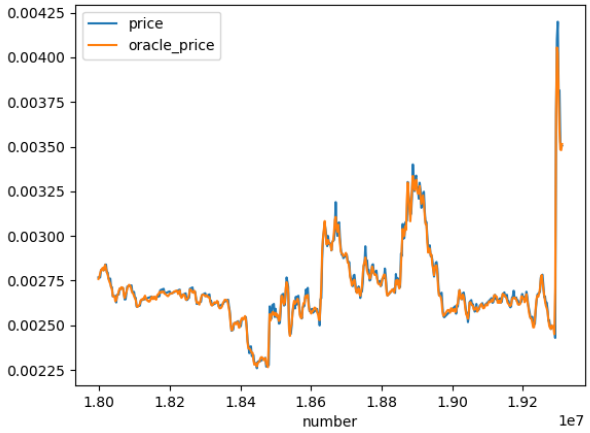


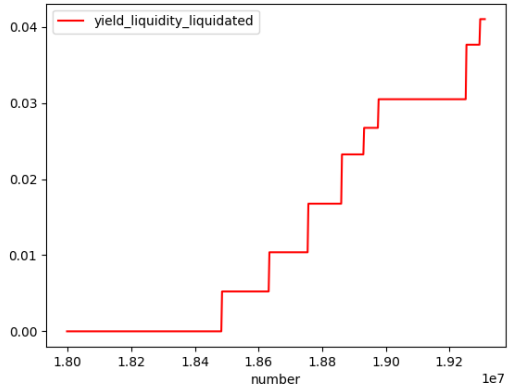
Figure 2: Price of UNI/ETH over the full simulation on the 12 hr oracle TWAP (orange) calculated from the Uniswap V3 reference pool, and the corresponding Marginal V1 pool price (blue).

If the Marginal V1 pool insurance mechanism is working as intended, the amount of liquidity returned to the pool upon settlement or liquidation must be at least as large as the original liquidity used to collateralize the position at open – i.e. liquidity returned should exceed the liquidity debt owed. In the simulations, the runner tracks the net gains in liquidity upon position settlement or liquidation in separate accumulators. The net gains for each position are given by

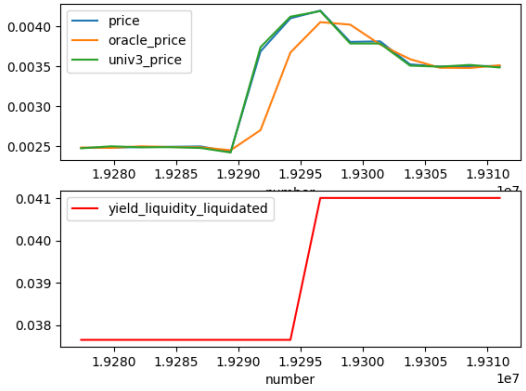
$$\delta L_{returned} - \delta L_{owed}$$

A strictly increasing cumulatives chart should indicate the mechanism is working properly as net gains would be positive for each position settled or liquidated. Figure 3 demonstrates the insurance mechanism working as intended for the UNI/ETH simulation in question. Liquidity providers continuously *gained* pool liquidity from liquidations. Further, for the most extreme stress test on the largest UNI/ETH price move ($>$ 67% in one hour), the pool still gained liquidity even though leveraged shorts were liquidated below the bankruptcy price. This was consistently the case across all of our simulations for the Marginal V1 pools.

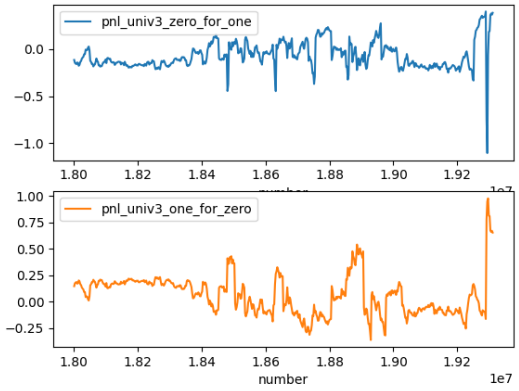
Cumulative yields broken down by category over the entire simulation for two separate runs shown in Figure 4. For the first run shown in Figure 4a, swap fees constitute the majority of the yield from the simulations given the longer position hold time of 7 days with a somewhat less risky position leverage of 3.5x. Total yield exceeded expectations from only swaps by about 10% due to gains from liquidations on leveraged positions.



(a) UNI/ETH cumulative yields due to liquidations on the pool over entire simulation.

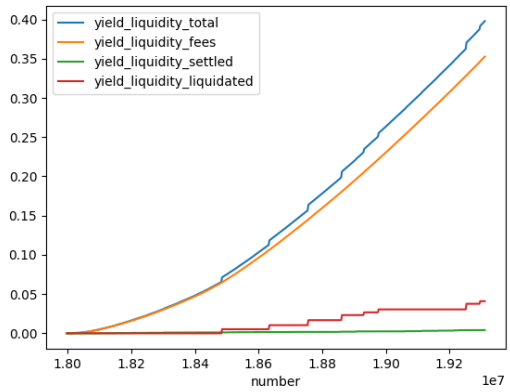


(b) UNI/ETH price, oracle TWAP and cumulative yields due to liquidations over last 5 simulated days.

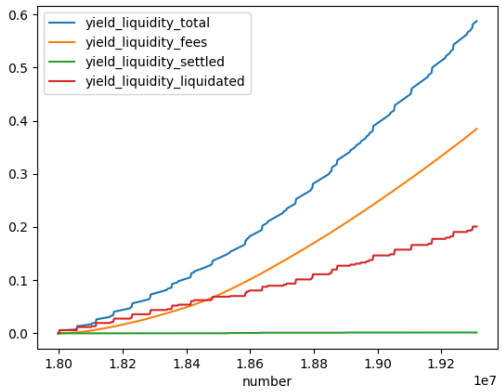


(c) Unrealized PnL of shorts `zeroForOne=True` and longs `zeroForOne=False`.

Figure 3: UNI/ETH Marginal V1 pool cumulative yields relative to initial liquidity provided, isolated for net gains/losses due to position liquidations, for the run with 25% pool utilization, 25% skew short, 50400 blocks held and position leverage of 3.5x. The pool insurance mechanism appears to work as intended, given liquidity providers *gained* pool liquidity from liquidating shorts on the largest UNI/ETH move (> 67% in one hour), even though the leveraged short position was liquidated below the bankruptcy price (`PnL < -1.0`).



(a) UNI/ETH cumulative yields on the pool over entire simulation for the run with 25% pool utilization, 25% skew short, 50400 blocks held and position leverage of 3.5x.



(b) UNI/ETH cumulative yields on the pool over entire simulation for the run with 25% pool utilization, 50% skew short, 50400 blocks held and position leverage of 10bps above the safe margin minimum.

Figure 4: UNI/ETH cumulative yields relative to initial liquidity provided broken down by category, for two runs: (a) 25% pool utilization, 25% skew short, 50400 blocks held and position leverage of 3.5x and (b) 25% pool utilization, 50% skew short, 50400 blocks held and position leverage of 10bps above the safe margin minimum ($\sim$ 5x).

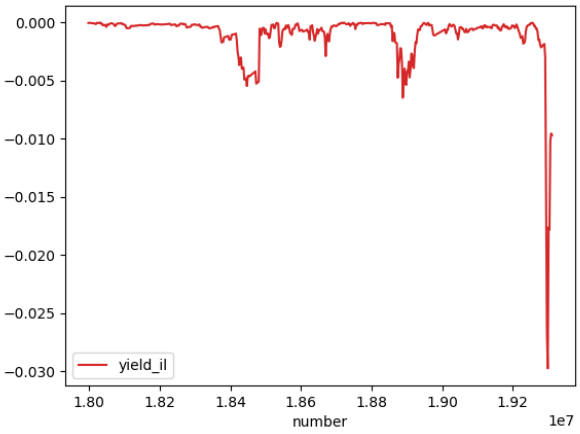


Figure 5: UNI/ETH Marginal V1 pool yields relative to initial liquidity provided, isolated for net losses due to impermanent loss (IL).

In the second run, traders took the maximum leverage possible at each position open. This resulted in even higher yields for Marginal LPs due to more frequent liquidations, shown in Figure 4b. Yield due to liquidations was 5x higher in the latter when compared to the former, and comprised about 1/3 of the total yield in the latter run.

Marginal LPs should expect providing liquidity to significantly outperform Uniswap V2 yields the more leveraged traders are with their positions on the pool. Associated unrealized loss on LP yield due to impermanent loss (IL) over the entire simulation shown in Figure 5. IL did not reduce LP profitability too significantly, but LPs should be careful for even longer tail pools like PEPE/ETH. Fees and liquidations would need to overcome significant IL in these types of assets to maintain overall LP profitability.

References

Adams et. al (2021). Uniswap v3 Core. URL https://uniswap.org/whitepaper-v3.pdf

- (2023). Marginal V1. URL https://github.com/MarginalProtocol/v1-core/blob/main/wp/v1.pdf