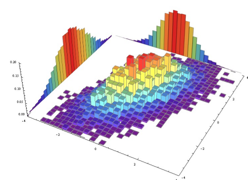
Addendum for Kodiak Economic Simulations

SmolQuants

February 2024



Contents

1 Introduction

Kodiak is a concentrated liquidity AMM with vaults to be launched on Berachain.

This addendum adds backtests of vault strategies against historical price data from a risk asset pool: PRIME/ETH 100 bps on Uniswap v3.

2 Summary

- The proposed strategy consists of providing liquidity over a fixed time period τ at a fixed half tick width Δ, and rebalancing around the current pool tick once this time period elapses.
- As in the original report, we also test an opportunistic vault strategy that only concentrates liquidity when fee volumes from the prior period are high enough to make it +EV to concentrate, relative to expected losses due to IL and rebalancing.
- The opportunistic strategy LPs over the full tick range when fee volume θ is less than the bound $(\sigma^2/8)(l+1)$ to minimize impermanent loss (IL). However, when θ exceeds the bound, concentrate liquidity around the critical half tick width Δ_c calculated by `optimize.py` that solves

$$0 = \partial_\Delta \mathbb{E}_0[V_\tau]|_{\Delta=\Delta_c}$$

- The strategy was backtested assuming a 1 day rebalance period from block 18648490 (Nov-25-2023) to 19080490 (Jan-25-2024) on the mainnet Uniswap v3 PRIME/ETH 100bps pool with ETH as the quote currency. Yields underperformed by 14% relative to simply LPing over the full tick range due to significant price shocks while concentrating the LP position's tick width, which caused more extreme realized IL at the end of the rebalance period.
- As with USDC/ETH simulations, significant negative changes to strategy yield relative to full range LPing happened after extreme price shocks in periods of high volatility when the opportunistic strategy aims to take advantage of high volumes and thus high fees.
- Coupled with the small probability of such shocks occurring under GBM gives further credibility to the argument that the vault strategy is underperforming due to mismodeled price behavior (i.e. no tails modeled) when calculating the optimal tick width to LP over.
- To avoid these issues, Kodiak should consider implementing vaults at larger tick widths and supplement earned fee revenues in the vaults with token incentives. Aim for a reward yield R greater than the Uniswap v2 loss-versus-rebalancing (LVR) metric to ensure +EV LPing for most vaults at a chosen target vault TVL:

$$R \gg \frac{\sigma^2}{8}$$

3 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. principal, fee values of 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 Kodiak economic simulations, we used historical data on Uniswap v3 pools and implemented two types of runners.

The first is a `UniswapV3LPSimpleRunner` runner class that creates a hypothetical LP position on the mock of the reference pool at a pre-determined tick width. Liquidity is provided symmetrically around the current tick.

The tick width used at each rebalance is fixed over the entire simulation (i.e. no optimization calculations are performed). The user specifies:

- `refs["pool"]`: Uniswap v3 pool to reference.
- `tick_width`: Tick width 2Δ to LP over.
- `blocks_between_rebalance`: The rebalance period τ in blocks.
- `compound_fees_at_rebalance`: Whether to fold in accumulated fees into the rebalance swap.
- `amount1`: The physical amount of quote tokens δy_0 to provide.

The second is a `UniswapV3LPOptimizedRunner` runner class that inherits from `UniswapV3LPSimpleRunner`, but optimizes the tick width after each rebalance based on average fee revenues per unit of external liquidity θ over the prior period.

The strategy is as follows:

- If $\theta \leq \frac{\sigma^2}{8}(l+1)$, the runner provides liquidity over the full tick range for the next period to minimize IL.
- If $\theta > \frac{\sigma^2}{8}(l+1)$, the runner opportunistically concentrates liquidity around the current tick using the optimal tick width $2\Delta_c$, which is computed via the optimization procedure in `optimize.py`.

In addition to the parameters for the simple runner, the user also specifies:

- `mu`: Drift μ calculated from fits to historical log-price data on the pool.
- `sigma`: Volatility σ calculated from fits to historical log-price data on the pool.
- `max_tick_width`: Max optimal tick width, above which default to full tick range.
- `rewards`: Rewards per unit of external virtual liquidity R for active LP incentives

Backtests were simulated from block 18648490 (Nov-25-2023) to 19080490 (Jan-25-2024) on the Ethereum mainnet Uniswap v3 PRIME/ETH 100bps pool with ETH as the quote currency. In each backtest simulation, the runner rebalances every $\tau = 7200$ blocks or about once per day, and compounds fees at each rebalance swap. Runner deploys 100 ETH of liquidity and the corresponding amount of PRIME at the start tick.

We run simulations for the following fixed tick widths with the simple runner: `[2800, 5600, 8400, 1774400]`. The last element in the tick width list is full tick range for the pool.

We also run a simulation with the optimized runner strategy. Price was taken to be driftless $\mu = 0$ over the 1 day rebalance period, and per-day volatility taken from fits to be $\sigma = 0.06084$. Max tick width set to $\Delta_c \leq 14000$ above which the runner provides liquidity over the full tick range. No token incentives $R = 0$ were assumed.

Backtest results of strategy yields relative to passively holding initial capital $p_t \delta x_0 + \delta y_0$ are produced in Figure 1. We also plot accumulated fee returns per strategy in Figure 2 and relative price changes in Figure 3 to compare with anticipated principal losses due to realized IL. Notebook to generate the plots is provided in the repo as `addendum.ipynb`.

For the full range LP, which does not need to rebalance given its chosen infinite tick width, anticipated losses relative to passively holding using the end of simulation price deviation figure of `-0.103` would be $2\sqrt{p_t/p_0}/(p_t/p_0+1) - 1 =$ `-0.001479`.

Backtest results in Figure 1 confirm those of the original report. The more concentrated the tick width, the more significant the losses as IL over the simulation becomes realized at each rebalance. The best performing strategy for the PRIME/ETH pair was the simple full range LP, which minimizes IL losses and does not require a rebalance swap of LP principal.

Figure 4 produces similar strategy yield comparison plots as in the original report: strategy yields relative to passively holding only for the opportunistic strategy and the full range LP. In the same panel, we also plot log-price changes (8 hour candles), the optimized tick widths used by the opportunistic strategy, and fee revenues per unit of external virtual liquidity θ calculated over the prior rebalance period by the optimized runner.

This time there are several instances where the optimized runner decides to concentrate liquidity over the simulation, given the high volumes in the two recent months chosen for the backtest timeframe. We obtain

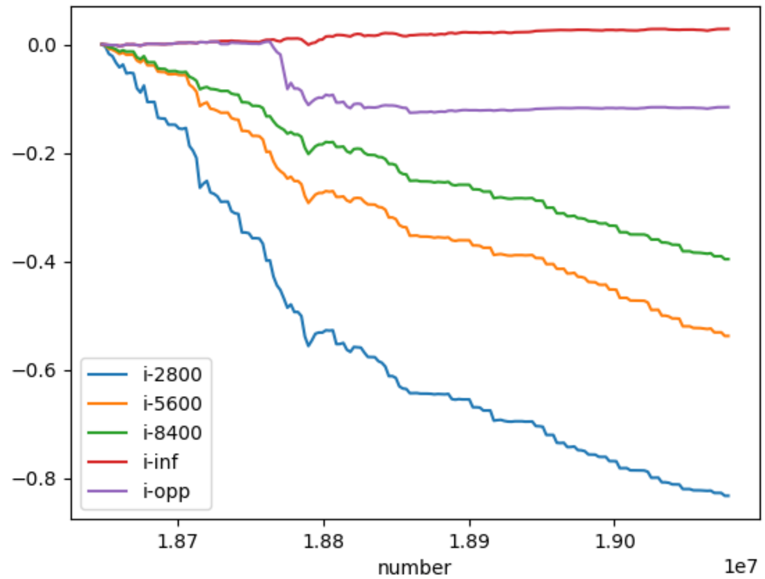


Figure 1: Backtested strategy yields benchmarked relative to passively holding initial capital, with ETH as quote. Simple runner simulations labeled in the legend as `i-[tick_width]`, where `tick_width` is the fixed tick width used over the entire simulation. `i-inf` simulates full range LPing by the simple runner and `i-opp` the optimized runner strategy.

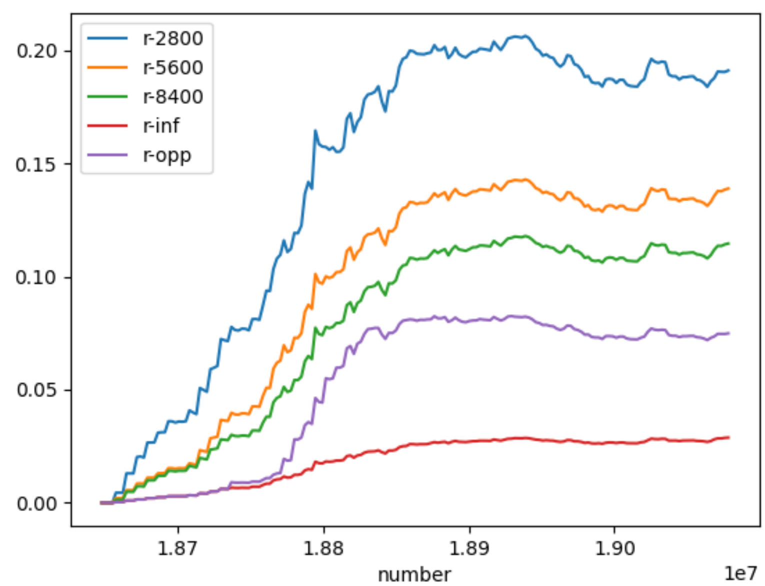


Figure 2: Backtested strategy fee returns benchmarked relative to initial value of LP principal. Simple runner simulations labeled in the legend as `r-[tick_width]`, where `tick_width` is the fixed tick width used over the entire simulation. `r-inf` simulates full range LPing by the simple runner and `r-opp` the optimized runner strategy.

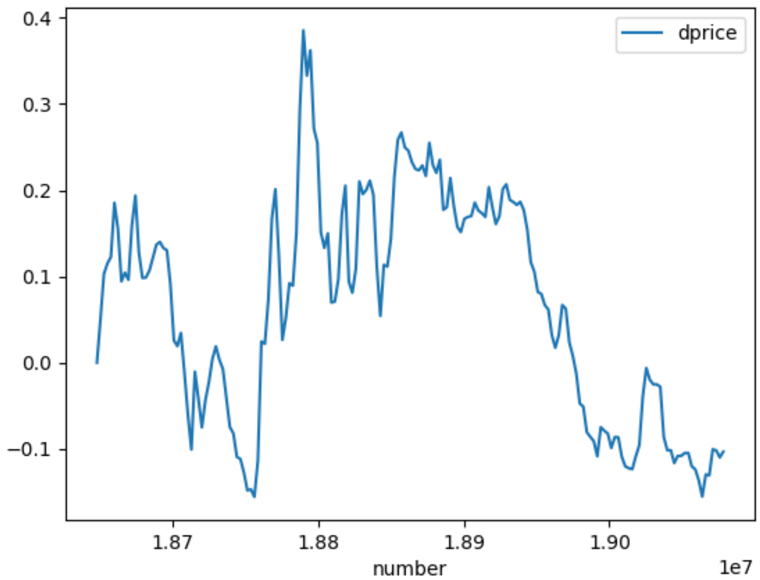


Figure 3: Relative change in price p_t/p_0 v.s. block number on the Uniswap v3 PRIME/ETH 100 bps pool. ETH as the quote token.

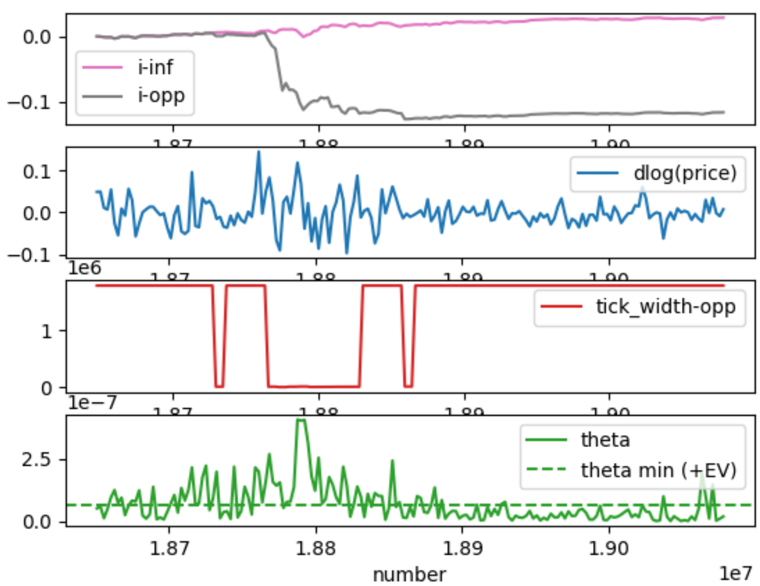


Figure 4: Backtested strategy yields benchmarked relative to passively holding initial capital for only the full range LP (pink) and optimized runner strategy (gray). Plots in the panel also display log-price differences per 8 hour candle (blue), the tick width 2Δ used by the optimized runner (red), and the fee revenues per unit of external liquidity θ (green).

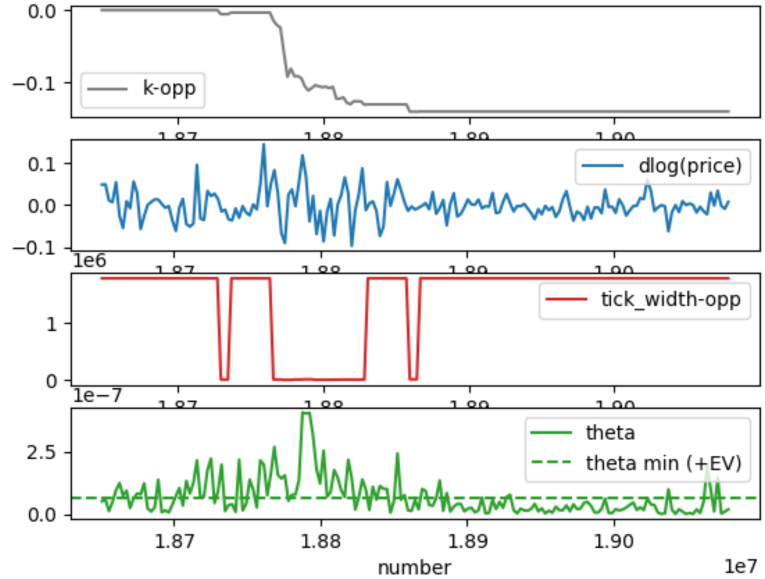


Figure 5: Backtested strategy yields for the optimized runner benchmarked relative to the full range LP, labeled `k-opp`.

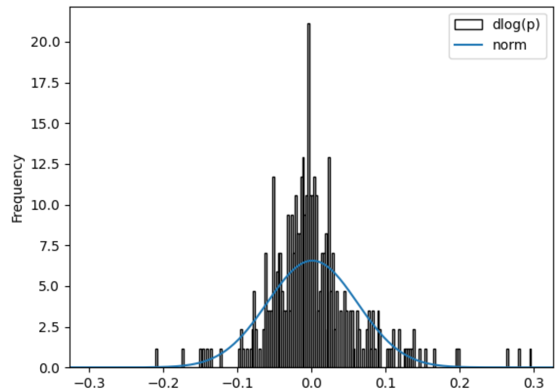
similar performance results in these concentrated periods when compared with the optimized runner in the original report but now with a 2x higher per-day volatility pair. The periods when the optimized runner decides to concentrate liquidity are those when fee volumes are highest. But periods in which fee volume ramps up are usually those in which volatility also spikes, and price jumps tend to be most extreme.

Table 1 provides a timeseries of the optimized runner's performance relative to simply full range LPing (`k-opp`) and change in performance per 8 hour candle (`d[k-opp]`) over the period in which the runner was most active. Performance appears to degrade the most when liquidity is most concentrated and a significant price shock occurs, with several 2.5-3σ+ events over the time frame analyzed.

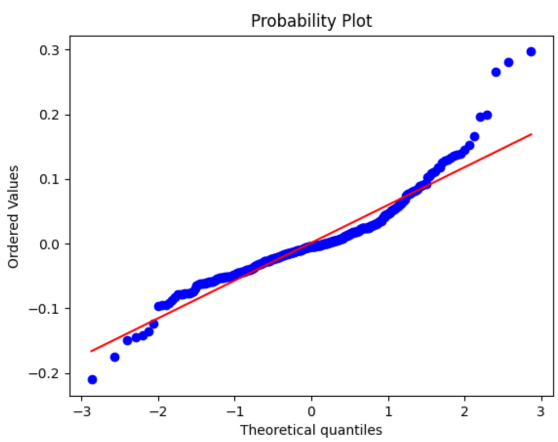
This supports the original report's conjecture that we need to model tail behavior (or at least some level of stochastic volatility) when actively LPing. Otherwise what the optimized runner expects to be +EV, will in reality not actually be given we're mismodelling the price behavior. Histogram and probability plots of PRIME/ETH log-price changes in Figure 6 also support this claim.

References

Milionis et. al (2022). Automated Market Making and Loss-Versus-Rebalancing. URL https://arxiv.org/abs/2208.06046



(a) Histogram



(b) Probability plot

Figure 6: Histogram and probability plots of log-price 24 hour candles taken from the Uniswap v3 PRIME/ETH 100 bps pool. Data queried from block 16741748 to 19160948.

Block number	$\log(P_{t_{i+1}}/P_{t_i})$	$\mathbb{P}[\lvert\log(P_{t_{i+1}}/P_{t_i})\rvert > z]$	$2\Delta_{opp}$	`k-opp`	`d[k-opp]`
18763691	-0.002600	0.470499	1774400	-0.003270	0.000000e+00
18766091	0.049698	0.078555	10000	-0.014354	-1.108465e-02
18768491	0.083096	0.008998	10000	-0.020505	-6.150473e-03
18770891	0.028699	0.206955	10000	-0.024034	-3.529080e-03
18773291	-0.066697	0.028794	2800	-0.058663	-3.462894e-02
18775691	-0.090295	0.005075	2800	-0.092256	-3.359298e-02
18778091	0.024699	0.240978	2800	-0.080980	1.127583e-02
18780491	0.037298	0.144151	9600	-0.091420	-1.044049e-02
18782891	-0.002700	0.469366	9600	-0.091334	8.605354e-05
18785291	0.054197	0.061420	9600	-0.094282	-2.947282e-03
18787691	0.118394	0.000375	13200	-0.105593	-1.131154e-02
18790091	0.067797	0.026795	13200	-0.111220	-5.627021e-03
18792491	-0.038598	0.135914	13200	-0.107691	3.529161e-03
18794891	0.021799	0.267431	4800	-0.103871	3.820429e-03
18797291	-0.068797	0.025080	4800	-0.105443	-1.572086e-03
18799691	-0.013399	0.351427	4800	-0.106712	-1.268757e-03
18802091	-0.086196	0.007065	6000	-0.105919	7.921903e-04
18804491	-0.015699	0.327457	6000	-0.107984	-2.064350e-03
18806891	0.014699	0.337798	6000	-0.106042	1.941164e-03
18809291	-0.072296	0.019784	7600	-0.122594	-1.655198e-02
18811691	0.001200	0.486374	7600	-0.122492	1.020537e-04
18814091	0.023899	0.248130	7600	-0.120802	1.690353e-03
18816491	0.066097	0.029936	8800	-0.128271	-7.468827e-03
18818891	0.028099	0.211870	8800	-0.130393	-2.122143e-03
18821291	-0.096895	0.002903	8800	-0.126038	4.355101e-03
18823691	-0.011799	0.368465	8800	-0.126276	-2.377987e-04
18826091	0.025299	0.235689	8800	-0.126343	-6.707839e-05
18828491	0.087396	0.006421	8800	-0.130951	-4.608211e-03
18830891	-0.011999	0.366320	1774400	-0.130452	4.990786e-04

Table 1: Performance `k-opp` of the opportunistic strategy relative to the full range LP over the time period during which the runner was most active, from block 18763691 to block 18830891. Candle probabilities are calculated assuming GBM with zero drift and per day volatility of $\sigma = 0.06084$.