

Deviations from Tradition: Stylized Facts in the Era of DeFi

Daniele Maria Di Nosse^{*,†1}, Federico Gatta^{*,†1}, Fabrizio Lillo^{1,2}, and Sebastian Jaimungal^{3,4}

¹Scuola Normale Superiore, Pisa, Italy

²Dipartimento di Matematica, Università di Bologna, Italy

³Department of Statistical Sciences, University of Toronto, Canada

⁴Oxford-Man Institute for Quantitative Finance, University of Oxford, United Kingdom

November 19, 2025

* These authors contributed equally to this work.

†Corresponding authors. Email: daniele.dinosse@sns.it, federico.gatta@sns.it

Abstract

Decentralized Exchanges (DEXs) are now a significant component of the financial world where billions of dollars are traded daily. Differently from traditional markets, which are typically based on Limit Order Books, DEXs typically work as Automated Market Makers, and, since the implementation of Uniswap v3, feature concentrated liquidity. By investigating the twenty-four most active pools in Uniswap v3 during 2023 and 2024, we empirically study how this structural change in the organization of the markets modifies the well-studied stylized facts of prices, liquidity, and order flow observed in traditional markets. We find a series of new statistical regularities in the distributions and cross-autocorrelation functions of these variables that we are able to associate either with the market structure (e.g., the execution of orders in blocks) or with the intense activity of Maximal Extractable Value searchers, such as Just-in-Time liquidity providers and sandwich attackers.

Keywords— Blockchain; Decentralized Exchanges;; Maximal Extractable Value; Market Microstructure; Uniswap v3; Cryptocurrency

1 Introduction

Financial markets are physical or virtual venues where assets such as stocks, commodities, or currencies are traded. Traditionally, they have been dominated by Centralized Exchanges (CEXs), where a central party safeguards assets and oversees transactions. In recent years, however, Decentralized Exchanges (DEXs) have gained attraction by leveraging advancements in blockchain technology to eliminate the need for intermediaries. In this paper, we study how the trading organization of Decentralized Finance (DeFi) modifies the typical stylized facts and statistical properties of prices, orders, and volume observed in Traditional Finance (TradFi). Indeed, nowadays, billions of dollars are traded daily on these platforms¹. DEXs operate on the blockchain using specialized programs called smart contracts [34]. In the majority of DEXs, these contracts are implemented as Automated Market Makers (AMMs), which are algorithms that manage liquidity and regulate asset pricing on the exchange [4].

AMMs can be thought of as exchange desks and rely on two main components: liquidity pools and price formation rules. Liquidity pools can be conceptualized as “digital vaults” containing two types of coins, also referred to as tokens. In the following, we call them token X and token Y. The volumes of tokens stored in these pools are called reserves. As there is no centralized entity supplying liquidity, AMMs have two types of participants: Liquidity Providers (LPs) and Liquidity Takers (LTs). LPs contribute to liquidity by depositing both tokens into the pool, the amounts of which must satisfy specific constraints. In return, they receive a certain share of the pool, for example, in terms of Liquidity Pool tokens (LP-tokens) or via Non-Fungible Tokens (NFTs). Conversely, LTs interact with the pool by swapping one token for another and paying a fee for this service. These fees are distributed among LPs, typically in proportion to their pool share. To summarize, there are three admissible basic events in a DEX: swap, mint, and burn. A swap occurs when an LT exchanges one token for the other. A mint represents the action of an LP that provides liquidity to the pool. A burn corresponds to an LP withdrawing liquidity from the pool. Finally, it is worth underlining that every operation in the blockchain, independently of its nature, is subject to network-level execution costs paid to the blockchain,

¹See <https://defillama.com> for updates in the trading activity across DEXs.

such as the gas fee in the Ethereum chain or the transaction fee in Solana. The value of such fees depends on the network congestion, and it is typically made up of two components: i) the base fee – which is the minimum amount to be paid, and ii) the priority fee – which is an incentive paid to the network, chosen by the user, for being executed faster than others.

The price formation in AMMs is governed by fixed mathematical rules. The most widespread AMM mechanism is the Constant Function Market Maker, in which the value of a function of the reserves in the pool must be constant before and after every swap. Among all the possible functions, the most widely adopted is the Constant Product Rule, and such AMMs are termed Constant Product Market Makers (CPMMs). Under this rule, the product of the reserves of the two tokens must remain constant before and after a swap. This ensures that the price of one token relative to the other adjusts dynamically based on the reserves' balances. Finally, a mint (burn) shifts the constant product curve upwards (downwards) while keeping the price fixed. A prominent example of a DEX is Uniswap. Among the various blockchains on which Uniswap is deployed, the Ethereum implementation consistently accounts for the highest trading volume. When considering all DEXs and blockchain ecosystems, PancakeSwap [17] – a fork of Uniswap deployed on the Binance Smart Chain (BSC) – currently ranks first by traded volume. Despite this, our analysis focuses on the Ethereum blockchain, motivated by the vast number of projects developed on it, the presence of a dynamic and innovative community, and its comparatively higher degree of decentralization and censorship resistance relative to BSC.

At the time of writing, there are three operating versions of Uniswap: v2 [2], v3 [3], and v4 [1]. Historically, v1 (no longer working) was the first proposal, while v2 introduced significant improvements. Later, v3 brought important innovations that allowed for Concentrated Liquidity, and it is by far (at the time of writing) the most traded Uniswap version. Finally, v4 introduced even more possibilities for the users with the so-called 'hooks' that allow for a greater refinement of the dynamics of the liquidity pools. Although Uniswap v4 is the most complex version, both practically and theoretically, our analysis focuses on Uniswap v3 because Uniswap v4 is only recently released; hence, data is limited and not sufficient for studying the statistical properties of the pools' dynamics. Nonetheless, as the skeleton of Uniswap v3 and v4 is roughly the same, we expect our results to hold across both.

Since DEXs appeared, the scientific community's interest has consistently grown. Several papers, mostly addressing the problem of optimizing liquidity provision strategies, have appeared. There are, however, a limited contributions that study the statistical properties of the price, orders, and volumes time series in these markets, such as [33], [37], and [40]. Existing research primarily focuses on high-level metrics at coarse granularity, such as daily or hourly prices and returns, offering restricted insights into the fine structure of DEX market dynamics. These earlier works, therefore, miss many of the innovations in the DEX behavior, which are driven by the peculiar trading mechanism and thus only observable at very high-frequency.

Further, as existing studies primarily focus on price dynamics (which corresponds to the swap events) and none work at very high frequencies, there is, therefore, a significant gap in the literature. Our study fills this gap by focusing on event time analysis, enabling a detailed examination of DEX microstructure dynamics that is characterized by several departures from TradFi. Our work provides novel insights that bridge the gap between the theoretical understanding of traditional markets and the emerging dynamics of DEXs. Specifically, we take the publicly available (on the blockchain) data about the Uniswap v3 market and study its statistical properties at the event level. Our analysis involves both the LTs and LPs activity. We compare the stylized facts found on DEXs with what is observed in TradFi. Next, we show that the main differences are driven by the trading flow organization and the presence of special agents, named Maximal Extractable Value (MEV) searchers, which are involved with the creation of the block of transactions added to the blockchain. Even if this role is formally public, we find that most of the MEV searchers' activity is led by a few agents, thus highlighting the presence of significant technical barriers that prevent small investors from assuming this role. Finally, our work is also aimed at providing help for building realistic simulators of the DEX dynamics at the order level, similarly to what has been done in the last twenty years for CEX.

Overall, our analysis highlights the presence of significant patterns in the data that, while partly paralleling traditional financial markets, also diverge due to the specific market structure of DEXs and their technological infrastructure. These deviations are mostly (though not exclusively) associated with MEV strategies. The main differences are:

- ▷ The individual swap returns are leptokurtic, while, at least for large tick stocks, they are not in TradFi. In swap-time, the kurtosis linearly or super-linearly decays for the first few aggregated swaps, before behaving as a sub-linear power-law as the aggregation scale increases. See Section 4.1.
- ▷ Echo swaps are found in DEXs. Their impact is small yet not negligible, and may be related to MEV attacks occurring on other pools. This is not observed in TradFi. See Section 3.3.
- ▷ The AutoCorrelation Function (ACF) of returns shows some peaks for small lags that can be associated with echo swaps and the activity of reverse arbitrageurs (at lag 1) and of sandwich attacks (at lags 2 and 3). On the contrary, in TradFi markets, there is only one peak at lag one, which is associated with the

bid-ask spread. The arbitrageurs effect is persistent when measuring the ACF in clock-time. We also see a similar pattern in the signed volume ACF. See Section 4.2.

- ▷ Order flow is correlated, both in swap-time and tick-time, but herding is more important than splitting, contrary to what is observed in TradFi markets. The herding component remains the most relevant even when aggregating transactions from different pools. See Section 4.5.
- ▷ Liquidity is characterized by significant drops as a consequence of the Concentrated Liquidity mechanism. These drops generate jumps in the marginal price leading to deviations from the CEX benchmark. Although these divergences are not profitable and may persist for minutes, they generate artificial volatility in the pools. See Section 5.
- ▷ There is a very weak correlation between volatility and Liquidity provision range, while in TradFi markets, volatility and spread are strongly correlated. See Section 5.1.
- ▷ The transition probabilities between events display several peaks, which can be associated with MEV strategies. See Section 6.

The remainder of this work is organized as follows. Section 2 introduces the data we use for this study. Section 3 analyzes the MEV, which is a major driver of several statistical properties of Uniswap v3. Section 4 focuses on the swaps and returns time series. Here, we draw parallels with traditional assets. Section 5 discusses liquidity, corresponding to mint and burn events. Section 6 examines the transition probabilities between different types of events. Section 7 concludes. Some technical Appendices present descriptive elements that complete the narrative in the main text. Specifically, Appendix A provides the mathematical formalism of the Uniswap v3 framework. It is intended for readers who are approaching DEXs for the first time. Appendix B discusses general patterns about the market activity. Finally, Appendix C shows the numerical results concerning the swap events analysis.

2 Data

In this paper, we focus on analyzing the statistical properties and stylized facts of Uniswap v3. As discussed, this choice is driven by the significant volume exchanged on Uniswap v3 and the plethora of data available for carrying out statistical analysis. The study period covers the entire 2023 and 2024. Specifically, we focus on the 24 most active pools², and involves 15 token pairs. The data are retrieved using the Python package `eth_defi` and the official Ethereum subgraph of Uniswap v3 hosted by TheGraph³. Both methods provide standardized and reliable access to on-chain DeFi information.

To capture a broad spectrum of market dynamics, we categorized the pairs into four distinct groups based on the nature of the tokens involved. For every pair, we specify the numeraire against which the price and returns are computed. The pools are clustered in the same way.

- ▷ **Normal Pairs:** Consisting of a volatile asset and a stablecoin, which serves as the numeraire. The pairs included in this group are USDC-WETH, WETH-USDT, WBTC-USDC, and WBTC-USDT.
- ▷ **Volatile Pairs:** Comprising of pairs of two volatile assets. The numeraire is the less valuable token, i.e., the one with larger reserves in the pool. The volatile category consists of WBTC-WETH, LINK-WETH, MNT-WETH, and UNI-WETH.
- ▷ **Stable Pairs:** These pairs consist exclusively of stablecoins and include USDC-USDT, DAI-USDC, DAI-USDT, and USDe-USDT. As all these stablecoins are anchored to the USD, the price of these pairs oscillates around 1 and is characterized by low volatility. The numeraire used is the second token of each pair.
- ▷ **Synthetic Pairs:** Defined by pairs that feature WETH or WBTC coupled with ERC-20 wrapped version of liquid staking tokens. The synthetic pairs included are WETH-weETH, wstETH-WETH, and WBTC-LBTC. By construction, the price oscillates around 1. For each pair, the numeraire is either WETH or WBTC.

Henceforth, while the study includes all the pairs above, for illustrative examples we use the USDC-WETH 0.05% pool, as it is the most traded. When differences between pool clusters are worth highlighting, we display the most traded pool in each cluster: USDC-WETH 0.05% (Normal cluster); WBTC-WETH 0.05% (Volatile cluster); USDC-USDT

²By “most active” we mean those with the largest total number of events (swaps, mints, and burns). Specifically, we consider pools with at least 40,000 events recorded in the study period.

³Further details on data extraction may be found at <https://github.com/tradingstrategy-ai/web3-ethereum-defi> and <https://thegraph.com/explorer/subgraphs/5zvR82QoaXYFyDEKLZ9t6v9adgnptxYpKpSbxtgVENFV?view=Query&chain=arbitrum-one>, respectively.

0.01% (Stable cluster); **wstETH-WETH** 0.01% (Synthetic cluster). Whenever necessary, general quantitative results for all the pools are provided in the appendices.

Figure 1 summarizes the amount of data available in the considered pools. There is a significant heterogeneity across pools, especially pertaining to the number of liquidity provision (mint and burn) events. Most of the swaps occur in the Normal pools; the number of LP events is higher in the first two clusters (whose price is not anchored to any fair value) and lower in the last two (whose price oscillates around 1, by design). A richer summary of the data is provided in Appendix B, Table 2. Finally, we refer to Appendix B.1 for an analysis of the pool activity over the course of the day.

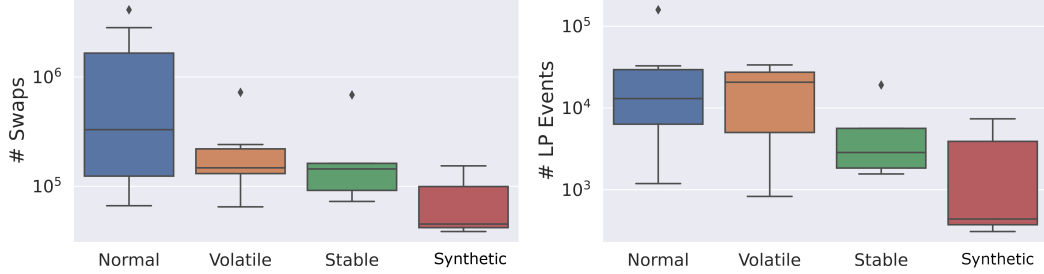


Figure 1: Amount of available data in the considered period (the entire 2023 and 2024), on log-scale, namely swaps (left) and LP Events (right), which include mints and burns. Each box plot describes all the pools in a cluster.

We carry out the analysis on different timescales, according to the microstructure property we study. Specifically, we consider:

- ▷ **Event-time** increases by one when a new event is recorded in the blockchain. Hereafter, ‘event’ means either a swap, a mint, or a burn occurring in the considered pool. When a new block with multiple events is added to the blockchain, the event time increases by one at each event, and the block index is used to order them.
- ▷ **Swap, mint, burn-time** refer to filtered versions of event-time, where only one specific type of event is considered. For example, the swap-time counter increases exclusively when a swap event occurs in the pool.
- ▷ **Tick-time** increases by one when the tick changes. Uniswap v3 price space range is partitioned by discrete ticks. A tick is an integer, defined as $\lfloor \log_{1.0001} S \rfloor$, where S is the marginal price.
- ▷ **Clock-time** is the usual physical time and we consider different time scales ranging from 30 seconds to 15 minutes.

Regarding the tick-time, a brief clarification is necessary. It is well-known that swap events always move the price; however, due to the discrete nature of ticks, they may or may not change in swap-time. Working in tick-time is particularly important when dealing with Concentrated Liquidity. Indeed, a tick movement could change the virtual reserves used to provide liquidity to the swaps, changing the liquidity profile, thus the price formation, and how the fees are distributed among the LPs. Moreover, moving the price tick could also result in higher gas fees (see the white paper [3] for more details). Finally, interestingly, the magnitude of the scale from swap-time to tick-time is strongly affected by the pair type: For Normal and Volatile pools, the average number of swaps for a tick change ranges between 1 and 6. For Synthetic pools, the range is between 11 and 49. For Stable pools, the highest recorded average is 131.53, corresponding to the **DAI-USDC** 0.01% pool. We suggest that this outlier is due to the DAI value being anchored to the USDC via both collateralization and a pegging mechanism.

3 Maximal Extractable Value

Before discussing the statistical analysis, we provide some background on the technical organization of DEXs and, specifically, on how transactions are submitted and validated on the Ethereum blockchain. This peculiar framework allows agents charged with handling orders to extract profit from transaction manipulations such as rearrangements, additions, or removals. This profit is known as Maximal Extractable Value (MEV)⁴, and it is an important driver of the trading dynamics in DEXs and thus in Uniswap. In this paper, we show that the pervasiveness of MEV practices is a key driver of the stylized facts of price and orders in DEX.

⁴Historically, it has been introduced as Miner Extractable Value, see [19].

In TradFi, orders are instantaneously processed, and the difference between submission from the user and insertion in the limit order book is usually of the order of microsecond, and it is rare to find trades sharing the same timestamp. Contrastingly, the order flow in DEXs is lumpy, as the transactions are grouped into blocks that are validated sparsely in time – currently, on the Ethereum blockchain, every 12 seconds; see [11] for a detailed discussion of the impact of block time length. Hence, submitted orders are not immediately executed, but rather, they need to wait before being added to the blockchain.

This is the main source of differences in the stylized facts between DeFi and TradFi, especially at event time. The orders submitted between two blocks are received by an Ethereum node and stored in the ‘memory pool’ (mempool), which can be seen as a ‘waiting room’ where transactions stay before being validated and appended to the next block. Typically, the mempool is public and accessible to everyone⁵. Before private bundle auctions [23] and later MEV-Boost implementation of Proposer-Builder Separation (PBS) [22] by Flashbots, *MEV searchers* often competed through public Priority Gas Auctions (PGAs) [20]. As of 2021, Flashbots MEV-Geth moved MEV bidding to sealed-bid private bundles that a large share of miners adopted, reducing PGA spam in the public mempool that contributed to gas fee price spikes. MEV-Boost then standardized PBS on proof-of-stake Ethereum by letting validators source blocks from specialized actors, the *block builders*⁶, thus, introducing a new marketplace where MEV opportunities could be captured off-chain. The purpose was, instead of competing in the public mempool, to allow MEV searchers, trying to extract MEV by continuously scanning the mempool, to bundle transactions and send them directly to the block builders in a bid-auction mechanism. Then, builders construct block proposals that are submitted to a validator, again in a bid-auction mechanism, which is managed by a trusted third-party known as *relay*. Figure 2 summarizes the pipeline. For a detailed description of this chain of actors, see [29].

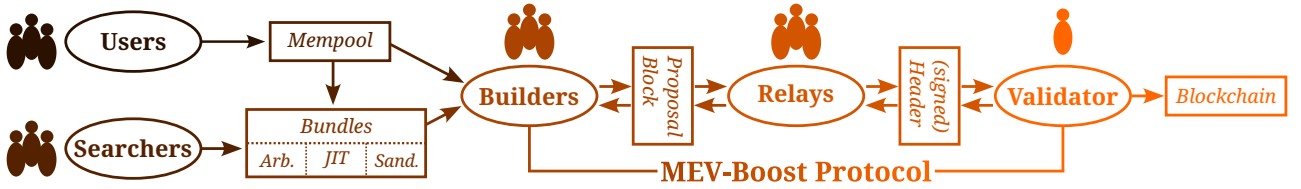


Figure 2: DEX orders submission pipeline. Users submit their orders to the mempool, which is accessed by searchers that create bundles of transactions to extract MEV. Each builder creates its own block proposal by taking transactions from the mempools and bundles from the searchers. Then, builders submit their proposals to the validator, which selects the most profitable one to be added to the blockchain. Relays act as intermediaries, and the whole auction is handled by the MEV-Boost protocol.

For our purposes, MEV searchers play a central role. Indeed, they have significant freedom pertaining to which transactions to include in the bundles and what ordering they have – which is generally different from the arrival order in the mempool. Thus, if they find exploitable situations (e.g., a big swap to front-run), they can rearrange the transactions to gain a profit. The most widely used MEV strategies include⁷:

- ▷ **Arbitrage opportunities:** MEV searchers can exploit price mismatches between two DEXs or between a DEX and a CEX. Consequently, they contribute to market efficiency.
- ▷ **Liquidations:** DeFi protocols allow users to borrow money against collateral. If the collateral’s value drops below a threshold, the user faces liquidation, where their crypto assets are sold to repay the debt. Liquidators buy these assets at a discount, making a risk-free profit. Consequently, MEV searchers compete for these opportunities.
- ▷ **Sandwich attacks:** strategies designed for gaining from the price impact of a large swap (victim) pending in the mempool. The MEV searcher first places a swap in the same direction as the one of the victim is placed (also known as front run); then, the victim swap is executed; lastly, the attacker closes the position by placing a swap with an equal amount and opposite sign to the front run (also known as back run).
- ▷ **Just-in-Time (JIT) liquidity** [36]: a strategy for gaining the fee that a large swap in the mempool will pay once executed. The MEV searcher deposits liquidity in a very narrow price range (containing the current price) just before the transaction is processed, earning most of the trading fee, and removes the

⁵For example, via Ethernw <https://ethernw.xyz/mempool/all>.

⁶While PGAs contributed to fee spikes, high gas periods also reflected general demand; moreover, the Ethereum Improvement Proposal (EIP) 1559 reformed the fee market and dampened fee-volatility dynamics [10]

⁷For a comprehensive survey about MEV in DeFi see [28].

liquidity immediately after the transaction. This strategy is also referred to as *LP sandwich* or *active liquidity provision*⁸ [13], as the liquidity provision is elicited by a pending order in the mempool.

- ▷ **Mixed (Sandwich+JIT) attacks:** When a large swap is detected in the mempool, a mixing attack made up of a JIT encapsulated into a sandwich could sometimes bring larger profit. Specifically, the exact scheme is: front run, mint liquidity, victim swap, burn liquidity, back run.

To summarize, there are mainly three strategies that an MEV searcher uses to react to a large pending swap. The choice strongly depends on the liquidity in the pool and the chance for hedging when providing active liquidity. Nonetheless, the key parameter is the pool fee. In low-fee pools, performing sandwich attacks is cheaper, and providing active liquidity provision is less profitable; in high-fee pools, the opposite is true. In the following subsections, we discuss some MEV strategies in more detail and study their prevalence in the investigated dataset.

3.1 Just-in-Time Liquidity

Liquidity provision is classically thought of as a passive operation, as the LP puts tokens in the pool and waits for LTs to enter the market. In contrast, Just-in-Time liquidity [36] is a kind of active liquidity provision [13] aimed at zeroing the periods when the LP’s tokens are unused and reducing the impact of adverse selection. When a big swap appears in the mempool, a MEV searcher can choose to instantaneously provide liquidity to that order. The attacker’s tokens are used and freed in the same block, and he/she earn a large fraction of the fee paid by the LT⁹. Specifically, the structure that we observe in the block is made up of a mint order with a very narrow tick range¹⁰ followed by the victim swap (or more than one) and, finally, the burn order that removes the previously minted liquidity, i.e., the mint and burn are placed by the same wallet and have the same liquidity amount and range. Algorithm 1 shows the pseudo-code for detecting JITs occurring in a pool.

Algorithm 1 Just-In-Time liquidity detection

```

1: Input: pool Dataframe.
2: Output: JITs list of triples describing JIT attacks.
3: Initialize JITs = list()
4: for block in pool.groupby("block_number") do
5:   for mint in block[ event=="Mint" ] do
6:     if exists burn in block[ event=="Burn" ] s.t. (burn.log_index > mint.log_index) and
       (burn.values== mint.values) then
7:       Define swaps = block[ event=="SwapX2Y" or event=="SwapY2X" ]
8:       if swaps not empty then
9:         JITs.append( (mint, burn, swaps) )
10:      end if
11:    end if
12:  end for
13: end for

```

Specifically, our JITs detection algorithm works as follows:

1. The input is a dataframe, `pool`, whose rows are the events occurring in the pool. The columns of the dataframe are: `block_number` and `log_index` indicate the block and the position of the event; `wallet` is for the wallet address sending the transaction; `event` is for the event-type – either `SwapX2Y`, `SwapY2X`, `Mint`, or `Burn`; `tick_l`, `tick_u`, and `amount` are for the lower tick, upper tick, and virtual liquidity of the liquidity position – they are `nan` if `event` is `SwapX2Y` or `SwapY2X`. See **line 1**.
2. The output is a list, `JITs`, containing the JIT attacks. These are recorded as triples: the first element is the attacker mint, the second element is the attacker burn, and the third element is the victim swaps. See **lines 2-3**.
3. Iterate over the blocks. For every block, scan all the mint and burn events. Looks for the couples (`mint`, `burn`) such that:

⁸Despite this definition is very appealing, it should not be confused with the so-called *active liquidity*, that is the sum of all the liquidity whose range contains the price, as discussed in Appendix A.2.

⁹In our dataset, the median share of the swap fee captured by the JIT attacker (computed over all the JIT attacks within a pool) exceeds 90% for most of the pools.

¹⁰Usually, the range is the narrowest possible. That is, the difference between the upper and lower tick is equal to the so-called `tickSpacing`, see [3] for further details.

- ▷ `mint` is executed in the same block and before `burn`;
- ▷ `mint[tick_l, tick_u, amount]` is equal to `burn[tick_l, tick_u, amount]`.

See lines 4-6.

4. If `mint` and `burn` are spaced by some `swaps`, then save the triple (`mint`, `burn`, `swaps`) as a JIT attack. See lines 7-9.

It is worth noting, unlike in [36], we do not impose the constraint that the three events are consecutive in the block and instead only require the log index order to be `mint`, `swaps`, and `burn`. That is, we also identify as JITs the patterns that are spaced out by other transactions within the same block. This is because the occurrence of other events, in different pools and/or blockchain protocols, inserted between does not harm the economic reason and the profitability of JIT attacks.

JIT attacks usually have a beneficial effect on the LT¹¹, as the swap price impact is reduced due to the presence of new liquidity in the pool. Finally, even if the LT is formally trading against the pool, as most of the liquidity is provided by the MEV searcher, the LT is mainly trading against the attacker. Thus, the searcher is in a similar position as the dealer in CEXs, as the searcher is positively (negatively) exposed on the token sold (bought) by the LT. Thus, the searcher faces the classical internalization-externalization problem (e.g., see [7]), where the problem is whether to hedge this exposure on other markets (either DEX or CEX) or hold the exposure and wait for other MEV opportunities to offset it.

As the JIT attacker aims to take almost all the fee paid by the victim LT, the virtual liquidity¹² minted by the active LP is expected to be far larger than the median liquidity minted by passive LPs. This is indeed the case, as shown in Figure 3, which compares the Kernel Density Estimation (KDE) of the liquidity minted by passive and active LPs for the USDC-WETH 0.05% pool. The latter is significantly larger as the attacker, being focused on a single swap, chooses a very narrow tick range.

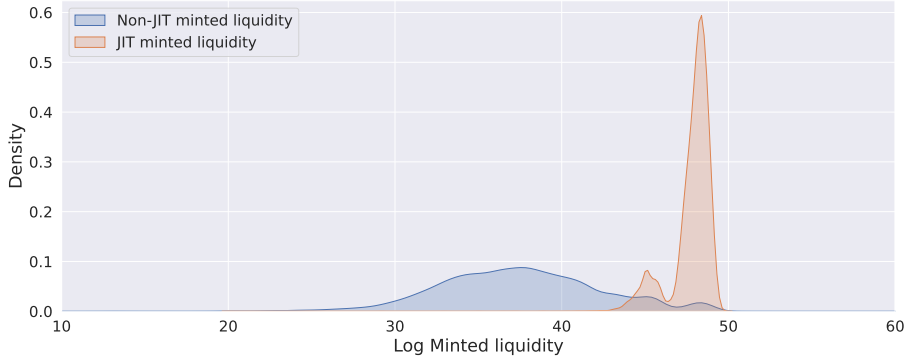


Figure 3: Decomposition of log liquidity minted in JIT liquidity and non-JIT liquidity for the USDC-WETH 0.05% pool. JIT liquidity is concentrated in the top region of the abscissa. This is mainly due to the narrow range characterized by these events.

Figure 4 shows the prevalence of JIT strategies in the pool as the percentage of mint events and minted liquidity involved in JIT attacks, as well as the percentage volume of victim swaps. We find that more than 20% of the total mint operations and more than 50% of the minted liquidity in our data is due to JIT¹³. This seems to be consistent with the active liquidity provision as proposed in [13], as these events appear more akin to a fundamental component of the market rather than a rare profit opportunity for MEV searchers. Furthermore, different fee tiers behave significantly differently. Indeed, the empirical observations confirm what has been previously observed: pure JIT attacks are profitable almost only in high-fee pools.

We next investigate the population of active liquidity providers. A previous analysis carried out in [39] highlights that more than 70% of all JITs have been made by a single user. However, the dataset used spans from June 2021 to January 2023; thus, only January 2023 overlaps with our dataset. Thus, following along the lines of [39], we study what occurred in the next two years. We collect the wallet addresses of the active LPs and compute the percentage of transactions executed from each wallet. Interestingly, the situation seems to be less heterogeneous. Nonetheless, 26% of all the JITs were carried out by an MEV bot operated by a user publicly known as `jaredfromsubway`¹⁴, one of the most famous MEV searchers on the Ethereum blockchain.

¹¹There is a slight misrepresentation in calling the user that places the central swap in a JIT attack “victim”. Nonetheless, we use the definition “victim” to unify JIT and sandwich nomenclature and to be consistent with the notation used in the MEV literature.

¹²See Appendix A.2 for further details on the virtual liquidity in Uniswap v3

¹³The percentage jumps to 90% when considering fee tiers 0.05% and 0.3%. However, it is slightly biased by the fact that the JIT ranges are usually very narrow.

¹⁴Wallet address `0xae2Fc483527B8EF99EB5D9B44875F005ba1FaE13`

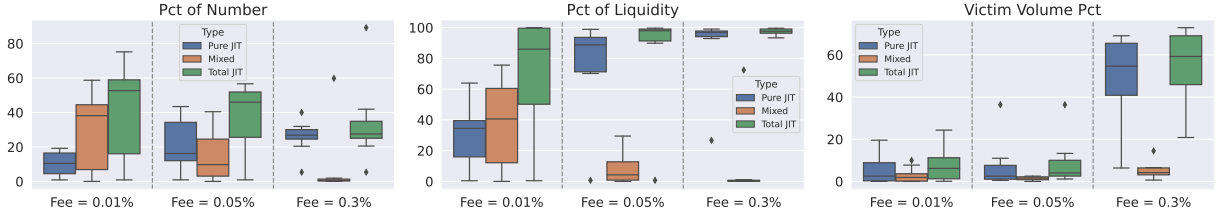


Figure 4: JIT impact – The left panel shows the percentage of mint events that are JIT; the central panel shows the percentage of minted liquidity belonging to JIT attacks; the right panel shows the percentage of the total volume traded in the pool that has been victim of a sandwich attack. In each panel, the information belonging to pool with different fee tiers is shown in a different box. Moreover, blue boxes show pure JIT events; orange boxes are for JIT involved in mixed JIT-sandwich attacks; green boxes show the total (pure+mixed) JIT attacks.

Overall, there are 13 accounts with at least 2% of the transactions. This number rises to 25 and 42 if we set the threshold to 1% and 0.5%, respectively. Ultimately, the trend seems to be an opening of the JIT provider role to even more agents.

Contrastingly, mixed attacks appear far more centralized. `jaredfromsubway` is still the most active wallet, responsible for about 88% of them. Only 3 attackers contribute more than 1% and 2%; and, 5 searchers contribute more than 0.5%.

3.2 Sandwich Attacks

Sandwich attacks are performed by MEV searchers willing to gain from the price impact of a large swap pending in the mempool. To achieve this task, the searcher adds their own swaps just before and after the victim's, with the intention of front-running and then back-running it. It is worth noting that this strategy has a negative effect on the LT, as the front run moves the price unfavorably, leading the victim to buy (sell) at a higher (lower) price. Algorithm 2 shows the algorithm we use for detecting these attacks.

Algorithm 2 Detection of Sandwich Attacks

```

1: Input: pool Dataframe.
2: Output: Sands list of triples describing sandwich attacks.
3: Initialize Sands = list()
4: for block in pool.groupby("block_number") do
5:   Define swaps = block[ event=="SwapX2Y" or event=="SwapY2X" ]; N = len(swaps)
6:   for front in swaps.pop() do
7:     if front.event == "SwapY2X" then
8:       back = swaps[ event=="SwapX2Y" ][0]
9:     else
10:      back = swaps[ event=="SwapY2X" ][0]
11:    end if
12:    if (back not nan) and (back.wallet == front.wallet) then
13:      Define victims = swaps[ front.log_index < log_index < back.log_index ]
14:      if victims not empty then
15:        Sands.append( (front, back, victims) )
16:      end if
17:    end if
18:  end for
19: end for

```

In detail, our algorithm works as follows:

1. The input is the same dataframe `pool` as in Subsection 3.1. See **line 1**.
2. The output `Sands` is a list of triples, each one describing a sandwich attack: (front-run, back-run, one or more victims). See **lines 2-3**.
3. Iterate over the blocks. Scan all the swaps in a block as potential front-run. See **lines 4-6**.

4. Search for the first swap in the block with the opposite sign. If it exists and has the same wallet as the candidate front-run¹⁵, then it is a potential back-run. See **lines 7-12**.
5. If the potential attacker’s swaps are spaced by at least one victim swap of the same sign as the front-run, then classify the pattern as a sandwich attack. See **lines 13-15**.

After applying our procedure, sandwich attacks targeting a single intermediate swap account for for $\sim 95\%$ of all cases, followed by attacks with two swaps ($\sim 4.5\%$), with the remaining patterns in total representing less than 1% of the cases. The largest number of swaps affected by a single attack is 11. Figure 5 shows the KDE of the logarithm of the absolute value of the USDC amount swapped in the sandwich attacks identified on the USDC-WETH 0.05% pool. There is a significant deviation in the amounts involved with sandwich attacks and the

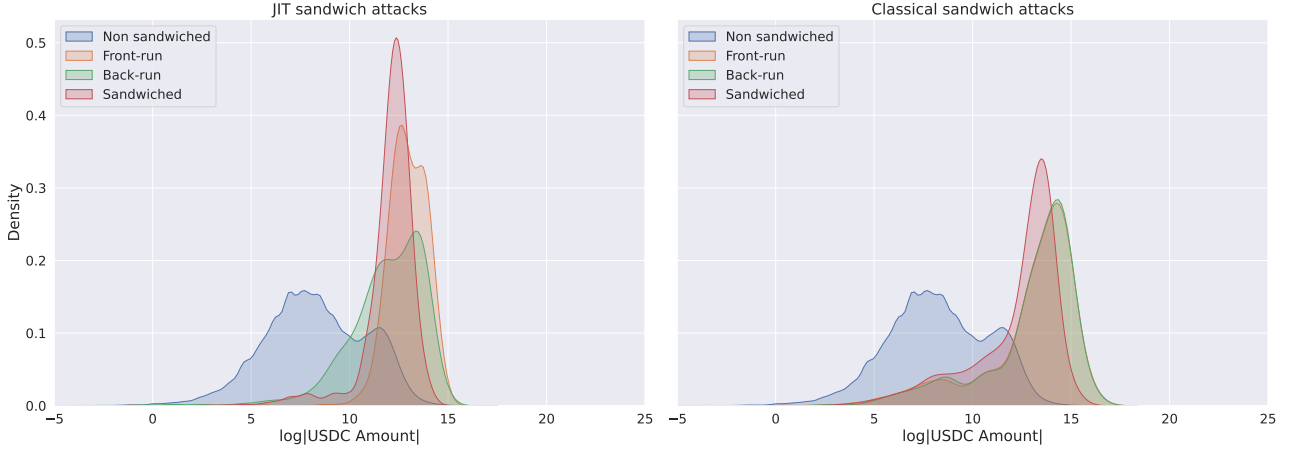


Figure 5: Comparison of KDE of the absolute value of the log USDC amount swapped during all the sandwich attacks discovered (36,353) on USDC-WETH 0.05% . The left plot is for mixed sandwich+JIT attacks; the right plot is for pure sandwich. The blue area is for swaps not involved in sandwich attacks (non-sandwiched swaps); the red area is for the victim swaps; the orange and green areas are for the front-run and back-run swaps.

other swap amounts. This is expected, as the larger the victim’s and attacker’s swaps, the higher the attacker’s profit. Furthermore, the front-run and back-run amounts in the pure sandwich attacks are almost equal (as it happens in TradFi), as they are designed to eliminate all exposure. Instead, in mixed attacks, the back-run distribution is shifted to the left of the front-run one. This is because part of the exposure opened with the front-run is liquidated trading against the victim in the JIT part of the attack.

Sandwich attacks account for a sizable fraction of the volume traded in Uniswap v3. Figure 6 shows the percentage volume related to sandwich attacks, both pure and mixed. Surprisingly, in low-fee tier pools, about 40% of the total traded volume is related to malignant sandwich swaps. When the fee tier increases, however, pure sandwich attacks become less frequent, as they become more expensive and JIT strategies become more rewarding. From this perspective, the pool fee can be intended as the price of the protection against sandwich attacks.

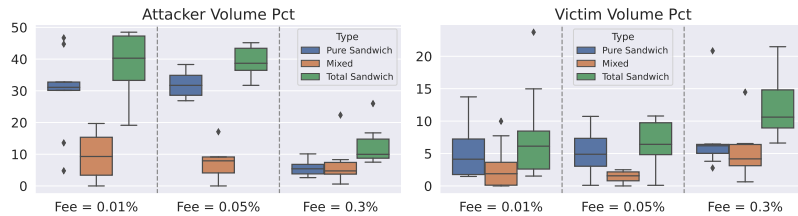


Figure 6: Sandwich impact – The left plot shows the percentage of the total volume traded that is involved with sandwich attacks (front-run and back-run). The right plot shows the percentage linked to victims. Blue and orange boxes are for pure and mixed attacks, green boxes are for their sum. Pools have been grouped according to the fee tier.

¹⁵It is possible that the attacker uses two different wallets. For example, the searcher could move the output of the front-run to another wallet before the back-run. As an additional gas fee should be paid for this, it is not very clear if this strategy makes sense. Nonetheless, the presence of two-wallet attacks could explain the inconsistency between the number of sandwiches found by different sources, which is highlighted in [26]. Thus, we left this kind of analysis for further research.

Regarding the attackers' identity, we find that $\sim 64\%$ of the pure sandwich attacks have been carried out by `jaredfromsubway`. Beyond this, 5, 8, and 24 agents contribute at least 2%, 1%, and 0.5% of the total attacks, respectively.

3.3 Echo Swaps

To conclude the discussion about MEV strategies, it is worth mentioning a peculiar pattern that we name *echo swaps*. It consists of two swaps with opposite signs and placed by the same agent in the same block, and without any other swaps in between. In other words, they can be thought of as sandwich attacks without any victim swaps in between. Echo swaps and sandwich attacks typically appear as sharp peaks in the price time series observed in swap-time. Figure 7 shows an example from the USDC-WETH 0.05% pool.

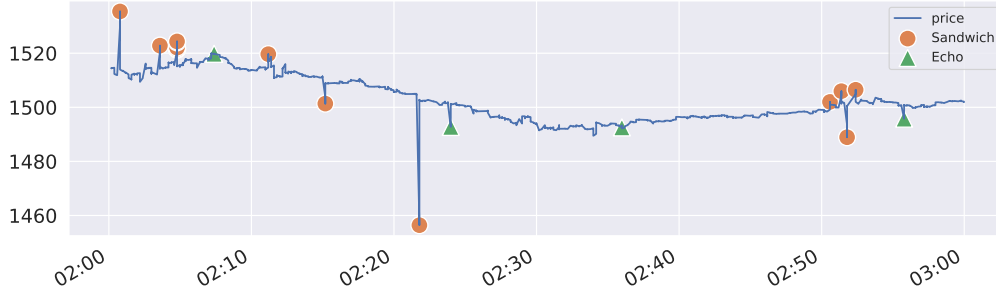


Figure 7: USDC-WETH 0.05% marginal price (swap-time) on March 11, 2023, between 2 and 3 am (UTC). Orange circles mark sandwich attacks; green triangles are for echo swaps.

Regarding the transaction hashes of the swaps, sometimes they are equal, meaning echo swaps are part of an atomic transaction, and other times they are different. Figure 8 shows the statistics related to echo swaps. We find no specific pattern in the percentage of atomic and non-atomic echo swaps. Regarding volume, their impact is usually smaller than 1% of the total swapped volume (17 out of 24 pools), and only in the USDC-USDT 0.01% and DAI-USDC 0.01% pools is it larger than 5%.

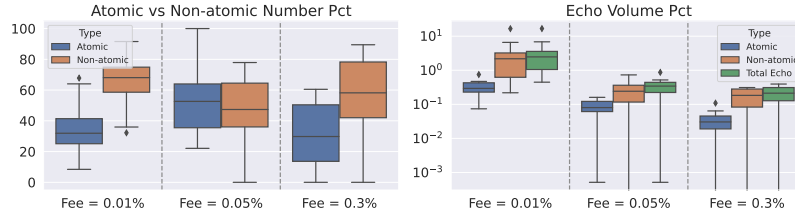


Figure 8: Echo Swaps impact – The left plot shows the percentage of echo swaps that are atomic (i.e., the swaps have the same transaction hash) or non-atomic. Blue (orange) boxes are for (non) atomic echo swaps. The right plot displays the percentage of the total volume swapped in the pools that is associated with echo swaps. Green boxes are for the sum of atomic and non-atomic echo swaps.

The economic motivations underlying echo swaps remain uncertain, and we leave a detailed exploration of these drivers to future research. However, our current conjecture is that they are used to supply the LT with tokens that are needed for MEV positions in other pools.

By analyzing their transaction hashes, we find that about 65% of them involve more than one pool.

Regarding the identity of the wallets behind echo swaps, we found that the most active one is still `jaredfromsubway` (8%). Generally, echo swaps are quite spread, with 8, 15, and 34 agents responsible for at least 2%, 1%, and 0.5% of their total, respectively.

Finally, we investigate whether attackers 'specialize', i.e., do agents perform one type of MEV strategy and neglect the other MEV opportunities. Our findings are shown in Figure 9. The most striking is that JIT attackers appear to be separated from the others. Further, we note a dependence between searchers of different strategies. Specifically, sandwich and mix wallets are usually also responsible for echo patterns. Similarly, mix and echo wallets also perform sandwich attacks. We interpret these results as evidence that echo swaps are related to MEV strategies, as they are often executed by wallets known to be MEV searchers.

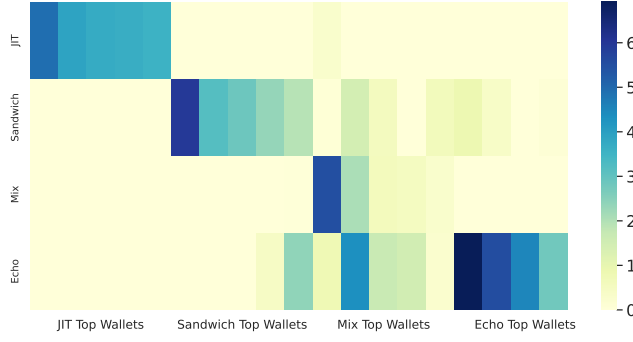


Figure 9: Percentage MEV activity from the most active attackers of each strategy (`jaredfromsubway0xae2Fc483527B8EF99EB5D9B44875F005ba1FaE13` excluded from the computation). On the x-axis, there are the top 5 attackers of each strategy. On the y-axis, there are the percentages of MEV attacks carried out.

4 Statistics of the Swaps

In this section, we study some stylized facts about the swap events and the return time series. Clearly, these quantities are strictly related, as swaps are the only events that can directly modify the price. As shown in Figure 1, swaps represent the majority of the events, accounting for 90% to 99% of the entire dataset, depending on the pool. Moreover, as MEV strategies are responsible for a huge share of the swapped volume (see Figure 6), they drive some peculiar patterns in the return time series observed in swap-time. In the following, we consider log-returns. Table 3 in Appendix B summarizes the main information about the returns over different pools.

4.1 Fat-Tail Return Distribution

Fat tails are ubiquitous across returns in financial markets. It is well known that the main driver of prices in DEXs is activity in CEXs – for example, see [14] or [24], which provides a theoretical analysis of the interaction between trades on CEXs and swaps on DEXs. Nonetheless, over very short time scales (e.g., one or a few swaps), the processes are not synchronized, making the resulting return distribution uncertain. Therefore, we investigate the Gaussianity of the return distribution through the D’Agostino-Pearson’s test [18]. The test, applied both in swap-time and tick-time, leads to the rejection of the null hypothesis of Gaussian returns. Contrastingly, if we aggregate more swaps, then the DEX price dynamics should mimic the CEX dynamics. It is known that by aggregating more and more returns they become more and more Gaussian.

Convergence to Gaussian occurs more quickly when working with transactions compared with working with clock time [16]. This observation may be partially replicated in Uniswap by generating a Q-Q plot of empirical data and the Gaussian fit as the number of aggregated blocks increases. Specifically, we aggregate n swaps with n equal to the average number of swaps in 15 minutes, 1 hour, 6 hours, and 1 day. The result of this exercise, for the USDC-WETH 0.05% pool, is reported in Figure 10. The figure indicates that as the number of aggregated swaps increases, the QQ plot progressively aligns with the diagonal, suggesting convergence toward a Gaussian distribution.

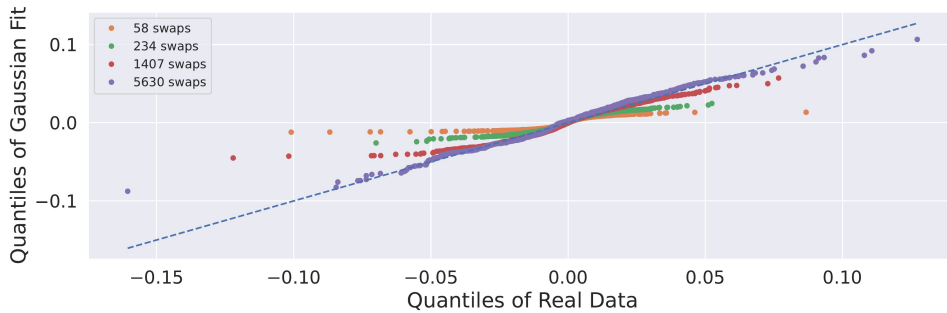


Figure 10: Q-Q Plot of the USDC-WETH 0.05% returns - empirical distribution vs Gaussian fit at different levels of swaps aggregation. 58, 234, 1,407, and 5,630 are the average swaps in 15 minutes, 1 hour, 6 hours, and 1 day, respectively.

The non-Gaussianity of returns can be examined through their kurtosis. At long aggregation scales, we expect DEX and CEX prices to be synchronized. When aggregating only a few swaps or blocks, however, the

DEX dynamics can diverge from those of CEX. Focusing on the returns' excess kurtosis in swap-time, we find extremely large values (typically decreasing as the fee tier ϕ increases), almost always above 1,000. The source for this pattern is not restricted to the size of traded volume. Indeed, the excess kurtosis of the swapped amount of token Y, while still positive, is up to two orders of magnitude smaller than that of the returns.

Next, we analyze the decay of the excess kurtosis viewed as a function $K(n)$ of the number of aggregated swaps (or blocks) n , with n smaller than or equal to 5. Table 1 shows the result of the regression $K(n) \sim A \cdot n^{-p}$ (estimated by regressing the $\log(K(n))$ against $\log(n)$).

Pool	N=2	N=3	N=4	N=5	Pool	N=2	N=3	N=4	N=5
USDC-WETH 0.01%	0.928	0.752	0.443	0.671	WBTC-WETH 0.3%	1.200	1.246	1.350	1.628
	0.773	0.437	<u>0.478</u>	0.856		0.916	<u>1.919</u>	<u>1.461</u>	<u>1.961</u>
USDC-WETH 0.05%	1.039	0.945	0.983	1.104	LINK-WETH 0.3%	0.975	1.078	1.085	1.264
	2.370	0.785	0.935	0.961		1.331	1.504	1.610*	1.683*
USDC-WETH 0.3%	1.648	1.798	1.710*	1.781*	MNT-WETH 0.3%	1.035	1.023	1.390	1.522
	2.477	2.269	2.127*	1.790		1.336	1.450	1.423*	1.325*
WETH-USDT 0.01%	0.915	0.853	0.782	0.555	UNI-WETH 0.3%	1.911	2.021	1.780	1.482
	2.609	0.576	0.993	0.655		1.543	1.755	1.250	1.166
WETH-USDT 0.05%	0.580	0.916	0.433	1.283	USDC-USDT 0.01%	0.563	2.185	1.106	0.658
	1.521	1.851	1.648	1.571*		0.393	0.182	0.151	0.061
WETH-USDT 0.3%	0.967	1.189	1.210	1.345	USDC-USDT 0.05%	0.664	<u>0.392</u>	<u>0.492</u>	<u>0.613</u>
	1.131	<u>2.097</u>	<u>1.807</u>	<u>1.497</u>		0.501	0.397	0.835	0.734
WBTC-USDC 0.05%	0.802	0.714	0.424	0.259	DAI-USDC 0.01%	5.097	1.437	2.007	1.803
	1.833	1.341	1.420	1.016		-0.107	0.087	<u>0.006</u>	<u>0.263</u>
WBTC-USDC 0.3%	1.179	1.660	1.737	1.818*	DAI-USDT 0.01%	0.960	0.673	0.734	0.468
	1.406	1.435*	1.469*	1.766		-0.567	0.502	0.393	0.739
WBTC-USDT 0.05%	0.533	0.494	0.124	0.152	USDe-USDT 0.01%	0.999	0.567	<u>0.838</u>	<u>0.936</u>
	0.148	0.776	0.603	<u>0.786</u>		0.446	0.740	1.078	0.996
WBTC-USDT 0.3%	0.758	<u>2.150</u>	1.385	1.272	WETH-weETH 0.01%	0.533	<u>1.130</u>	0.758	0.681
	1.948	2.000*	1.929*	<u>1.529</u>		1.748	1.512	<u>1.084</u>	<u>0.939</u>
WBTC-WETH 0.01%	0.545	0.777	0.701	0.427	wstETH-WETH 0.01%	0.772	0.233	0.249	0.267
	3.194	2.546	1.712	1.460		0.996	0.934	0.953	<u>1.253</u>
WBTC-WETH 0.05%	2.666	0.650	2.845	1.474	WBTC-LBTC 0.05%	0.804	0.678	0.832	0.762
	1.092	1.228	1.094	1.167		1.149	1.096	1.089	1.116*

Table 1: Excess kurtosis as a function of the number of aggregated swaps (top row) and blocks (bottom row). The p coefficient from the regression $K(n) \sim A \cdot n^{-p}$ is shown. The regression is performed using up to N values, and every column is for N . Coefficients obtained from a regression with the R^2 score greater than 0.9 (0.75) are in bold (underlined). * highlights those coefficients whose 95% confidence interval (according to the t-test) is on the right of 1.

We find that some pools (especially those with a high fee tier) have a decay coefficient p greater than or equal to 1. This pattern, which is theoretically associated with short memory in the stochastic variance process, represents a deviation from what is typically observed in TradFi. Contrastingly, when working with larger aggregation values n , we expect to recover a slow-decaying kurtosis function. Table 5 in Appendix C shows the results of the power-law fit and the mesokurtosis test proposed in [6] with the null hypothesis of a mesokurtotic distribution. The number of aggregated swaps n ranges from 15 to the number of swaps in the pool divided by 20, as 20 is the minimum sample size for performing the mesokurtosis test. The power-law provides an acceptable fit in almost all the pools. Generally, the estimated coefficient p decreases as the fee tier ϕ increases, and they lie in the range $[0.2, 1]$.

4.2 Returns Autocorrelation

Another set of quantities to investigate is the returns AutoCorrelation Function (ACF). Figure 11 shows the ACF, in swap-time, of the most traded pool in each cluster. The significance region is estimated using a bootstrap procedure: we generate 500 i.i.d. bootstrap time series from the original data and compute the ACF for each. For every lag, the bounds of the significance region are given by the 0.05 and 0.95 quantiles of the bootstrapped ACF distribution.

Interestingly, we find strong negative and significant autocorrelation for the first 3 lags. The magnitude of the peaks depends on the type and fee rate of the pool. For example, for Normal and Volatile clusters, this pattern is mainly present in low fee pools and is more pronounced at the second lag, while for Synthetic pools, we also observe large negative autocorrelation at lag 1. Finally, we find that the second lag peak is persistent when computing the ACF in tick-time, however, it vanishes when working in clock-time.

To understand the factors driving this pattern, we investigate each lag separately. First, let's focus on the peaks at lag 2 and 3. In this case, we conjecture that they are due to sandwich attacks discussed in Subsection 3.2. This is consistent with the observation that the pattern typically appears in pools with low fees, where

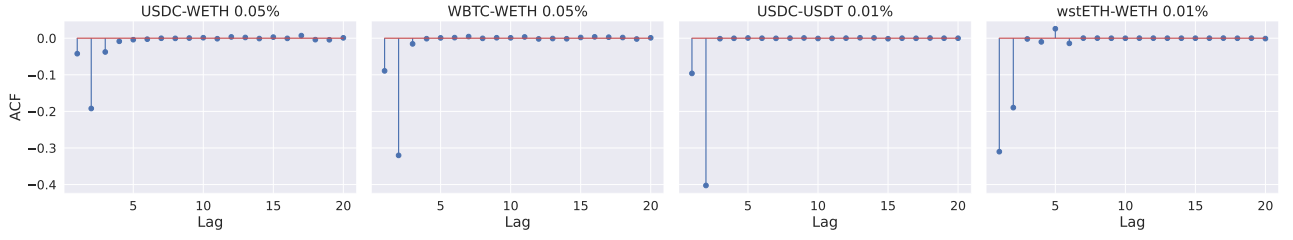


Figure 11: Returns ACF in swap-time, up to lag 20. The red area corresponds to the significance level, which is obtained with a bootstrap approach. Each subplot is for a pools cluster: USDC-WETH 0.05% (Normal cluster); WBTC-WETH 0.05% (Volatile cluster); USDC-USDT 0.01% (Stable cluster); wstETH-WETH 0.01% (Synthetic cluster).

front-running is more profitable. Moreover, it is coherent with the persistence of the ACF in tick-time and its vanishing in clock-time. Indeed, front-run and back-run trades are of significant volume and, consequently, they typically move the price tick. Furthermore, all the swaps involved within a single sandwich are in the same block, thus they share the same timestamp. To investigate our claim, we compute the ACF of log returns, conditioning on the absence of sandwich attacks between time t and $t + l$ ($l > 0$, swap-time). Figure 12 shows these results for the same pools as in Figure 11. When doing so, we find that the strong negative peaks at lags 2 and 3 we observe in the unconditional ACF disappear, while the autocorrelation at the first lag, due to a different strategy, survives.

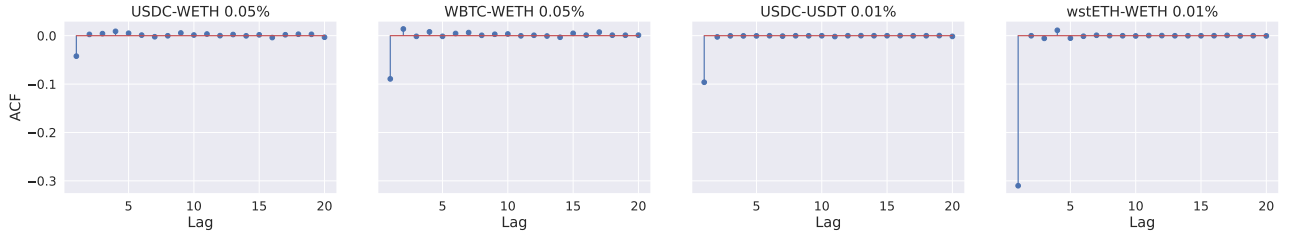


Figure 12: Returns ACF in swap-time, up to lag 20, conditional on the absence of sandwich attacks between two lags. The red area corresponds to the significance level, which is obtained with a bootstrap approach. Each subplot is for a pools cluster.

Next, we focus on the negative autocorrelation at lag 1. We recall that in TradFi, negative autocorrelation at lag 1 is due mainly to the bid-ask bounce effect at high frequencies. In DEXs, however, there is no spread, and therefore, one should not expect a bid-ask bounce. We, instead, conjecture that the negative autocorrelation at lag 1 is driven by the presence of echo swaps and the action of *reverse arbitrageurs*¹⁶, that is, a class of arbitrageurs who trade against the price impact generated by noise traders [12]. As for the echo swaps, we find that the ACF in the absence of their contribution is larger than that when including all the swaps in 21 out of 23 pools with negative first-lag ACF. This result is insensitive to whether we include or exclude sandwich attacks from the ACF computation. Regarding the reverse arbitrageurs,

findings in [12] highlight that this strategy is profitable if the impact is larger than the pool fee rate ϕ ¹⁷. To test the relevance of reverse arbitrageurs, let ϵ_t be the sign of the swap occurring in swap-time t , r_t be the log return between t and $t - 1$, and ϕ be the fee tier of the pool, we then compute the conditional probabilities $p_g = \mathbb{P}[\epsilon_{t+1} \neq \epsilon_t \mid |r_t| > \phi]$ and $p_l = \mathbb{P}[\epsilon_{t+1} \neq \epsilon_t \mid |r_t| \leq \phi]$. That is, we compute the probability of a swap sign reversal, after a swap at time t , conditional on the absolute return exceeding/falling short of ϕ . We then perform a one-sided bootstrap test with 1,000 replicas under the null hypothesis $H_0 : p_g \leq p_l$. The test produces a p-value smaller than 0.01 across all pools; hence, we reject H_0 . Further evidence for our conjecture is provided by computing $\text{Corr}(r_{t+1}, r_t)$ conditional on $|r_t| > \phi$ and $|r_t| \leq \phi$. Conditional on $|r_t| > \phi$, we recover the negative correlation observed. Conditional on $|r_t| \leq \phi$, the ACF shows a slightly positive trend that is explained by the presence of herd effect between LT, as we discuss in Section 4.5. Further quantitative results are reported in Appendix C, Table 6.

¹⁶Conversely, there exist stale-price arbitrageurs who profit by transmitting the impact of new information – i.e., changes in the fundamental value of the tokens – from centralized exchanges (CEXs) to decentralized exchanges (DEXs).

¹⁷To be completely accurate, we should also consider the gas fee. Our aim here, however, is to provide the intuition for the negative autocorrelation observed at the first lag. Thus, we keep the framework as simple as possible.

4.3 Returns Magnitude Autocorrelation

In this section, we investigate whether these markets exhibit volatility clustering. We do so by studying the autocorrelation of the absolute value of returns. Figure 13 shows the ACF in swap-time for the most traded pool in each cluster.

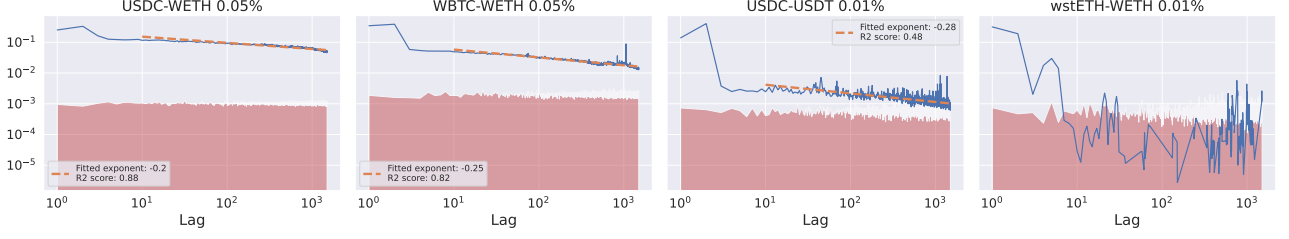


Figure 13: Autocorrelation in the absolute returns, swap-time. The panels show the most traded pool in each cluster. For sufficiently large lags, the data is fit well by a power-law $ACF(L) \sim A \cdot L^p$. The exponent p and the R^2 score are reported in the legend. The significance level described by the red area is obtained via the bootstrap approach described in Subsection 4.2.

In almost all the Normal and Volatile pools, we find a decay consistent with long-memory. Contrastingly, we find that Stable and Synthetic pools have ACFs that are consistent with short-memory, except for USDC-USDT 0.01%. The fitted coefficients of the power-law $ACF(L) \sim A \cdot L^p$, pool-by-pool, are provided in Appendix C, Table 7. The exponent p typically increases with the fee tier ϕ and is in the interval $[0.2, 0.5]$ for almost all the pools, hence, the exponent p is roughly in line with the literature on TradFi [16]. We note that these results are robust to switching to both tick-time and clock-time. As expected, the decay becomes faster as we reduce the observation frequency.

4.4 Volume & Signed Volume Autocorrelation

We now turn to the analysis of volume and signed volume. The volume is computed as the unsigned numeraire volume exchanged. In swap-time, this is simply the volume of the swap, whereas at aggregated timescales we use the sum of volumes in a given time interval. Similarly, the signed volume time series is obtained by multiplying the numeraire volume by the sign ϵ , which equals $+1$ if the swap occurs from the first to the second token of the pool, otherwise equals -1 . Figure 14 shows the volume and signed volume ACFs for the most traded pool in each cluster. The observed decays are consistent with long-memory in the volume ACF for almost all the pools, with only a few exceptions among Stable and Synthetic pools. See Appendix C, Table 7 for a comprehensive summary of the power-law fitted coefficients. Contrastingly, the signed volume is short-memory and shows the same negative peaks at the first lags as the ACF of returns. These findings are consistent with the price formation rule in Uniswap v3. As for the returns, the explanation behind such a pattern is related to the activity of MEV searchers performing sandwich attacks. When we remove events related to the activity of MEV searchers, the resulting ACF is similar to those shown in Figure 12.

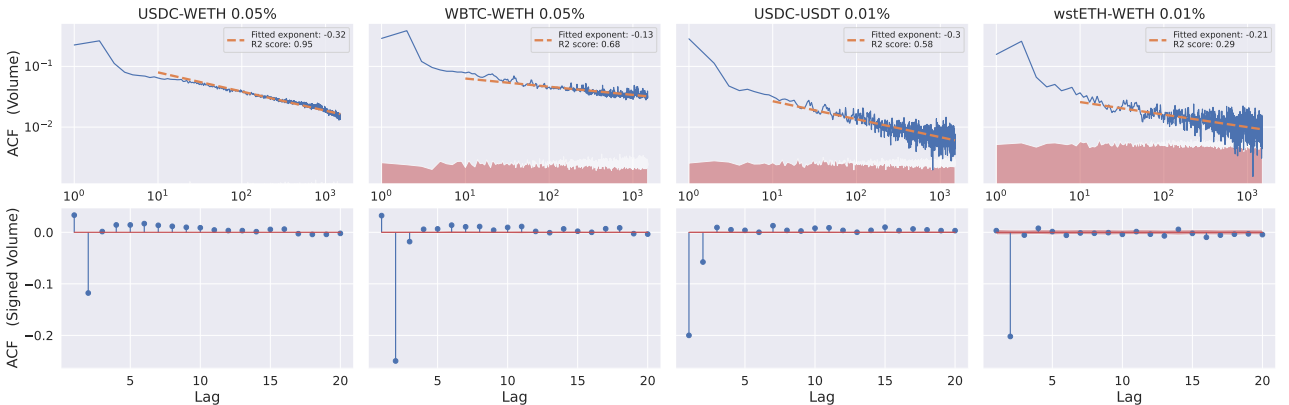


Figure 14: Volume ACF (upper) and signed volume ACF (lower) in swap-time. The panels show the most traded pool in each cluster. The legend reports the coefficient and the R^2 score of the power-law fit. The significance level described by the red area is obtained via the bootstrap approach described in Subsection 4.2.

Finally, we study the relationship between volume and variance by comparing the logarithms of Realized Variance (RV) and Volume. To this end, we partition the dataset according to clock-time at different aggregations. Inside each time interval, we compute the RV and the aggregated volume, and we investigate the contemporaneous relation between these quantities. The result concerning the **USDC-WETH 0.05%** pool is shown in Figure 15.

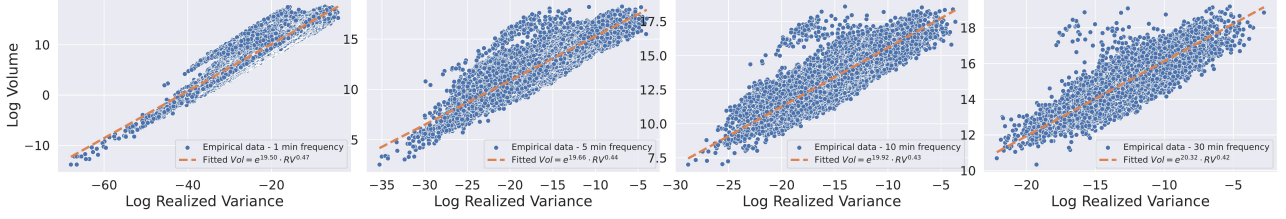


Figure 15: **USDC-WETH 0.05%** pool - Swaps Volume (referred to USDC) vs Realized Variance (RV), both on a log-scale. They are synchronously computed on time windows of fixed length. The orange dashed line shows the linear fit between the logarithms.

As shown, we find almost a linear relationship between the logarithms; moreover, the fitted coefficients are roughly the same at each frequency. Hence, the RV is approximately a power of the volume. This result holds for almost all the pools. Table 8 in Appendix C gives further insights into the linear relationship between $\log(RV)$ and $\log(Vol)$.

4.5 Trade Direction Autocorrelation

An important feature often observed in TradFi is the slow decay of the trade signs time series ACF (see, for example, [27] and [9]), both in event-time and tick-time [16]. Here, we investigate whether this pattern survives in DEXs. As discussed in the previous subsections, the sign of a swap is defined as $+1$ or -1 , depending on whether the swap is from the first to the second token or the reverse. Thus, the sign series in swap-time is a binary sequence of ± 1 . In tick time, however, we define the sign as the mean of the signs of all the trades between two consecutive tick changes. Figure 16 shows the ACF of the sign time series for the most traded pool in each cluster in swap-time and tick-time.

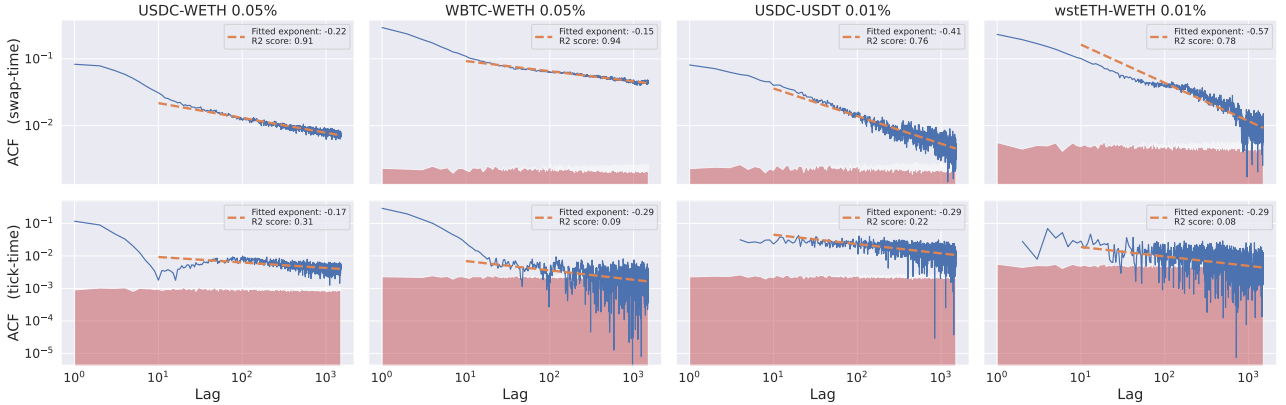


Figure 16: Sign ACF in swap-time (upper) and tick-time (lower). The panels show the most traded pool in each cluster. The output of the power-law fit is in the legend. The significance level is obtained via the bootstrap approach described in Subsection 4.2.

Interestingly, we observe a slow decay of the trade signs ACF also in our data. This phenomenon is particularly evident for Stable and Synthetic pools, while it is less pronounced in Normal and Volatile ones. The full picture is in Appendix C, Table 7. To investigate the causes that originate the slow decaying observed, we draw from the TradFi literature, where mainly two explanations have been proposed. The most respected one attributes the long-memory property to the execution of meta-orders¹⁸. It is not clear, however, whether this strategy is equally relevant for DEXs. On the one hand, it can be helpful to mitigate sandwich attacks and exploit mean-reversion to reduce execution costs; on the other hand, every swap generates an additional gas

¹⁸A meta-order is a large order split into multiple smaller ones executed incrementally to avoid incurring huge transaction costs.

fee, which raises doubts about the meta-order profitability. The second explanation proposed in TradFi relates the slow decline to the herding effect – that is, the tendency of investors to replicate the trades of others [25].

This hypothesis has been found to be less compelling in traditional markets. This hypothesis may, however, be more relevant in DEXs given their underlying dynamics. Indeed, the cryptocurrency price is characterized by many strong price trends that can be attributed to herding effects. Such patterns introduce strong autocorrelation in the direction of the swaps.

We empirically compare the reliability of the two explanations by following the procedure described in [35]. Specifically, we decompose the autocorrelation function in order to disentangle, at each lag, the contribution of the swaps performed by the same wallet (C_{split}) from the swaps performed by different wallets (C_{herd}). Specifically, let $C(\tau)$ be the ACF evaluated at lag τ and ϵ_t^i the action of wallet i at time t , defined as +1 for a swap from the first to the second token, -1 for the opposite direction, and 0 if the swap is not performed by wallet i . Specifically, we define

$$C(\tau) := \frac{1}{N} \sum_t \sum_{i,j} \epsilon_t^i \epsilon_{t+\tau}^j - \left(\frac{1}{N} \sum_t \sum_i \epsilon_t^i \right)^2. \quad (1)$$

This expression may be rearranged as

$$C(\tau) = C_{split}(\tau) + C_{herd}(\tau) \quad (2)$$

where

$$C_{split}(\tau) := \frac{1}{N} \sum_i \left[\sum_t \epsilon_t^i \epsilon_{t+\tau}^i - \frac{1}{N} \left(\sum_t \epsilon_t^i \right)^2 \right] \quad (3)$$

$$C_{herd}(\tau) := \frac{1}{N} \sum_{i \neq j} \left[\sum_t \epsilon_t^i \epsilon_{t+\tau}^j - \frac{1}{N} \left(\sum_t \epsilon_t^i \right) \left(\sum_t \epsilon_t^j \right) \right] \quad (4)$$

The results of this decomposition are reported in Figure 17, where we observe that the herding component dominates the autocorrelation values. This is a notable departure from trends seen in traditional markets.

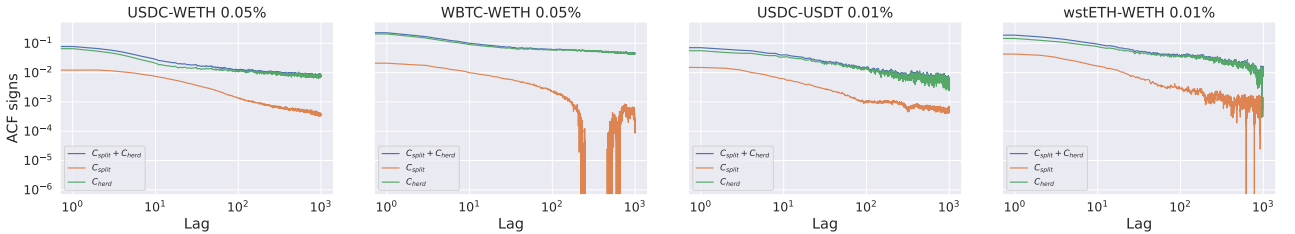


Figure 17: Decomposition of the ACF of swap signs (swap-time) in two components: one for the swaps performed by the same wallet (C_{split}) and one for the swaps performed by different wallets (C_{herd}). Their sum is equal to what is shown in the top panels in Figure 16.

It is worth noting that this result is also observed when accounting for some specific features of Uniswap. First, the result is qualitatively the same when (pure and mixed) sandwich swaps are removed. Second, the result is robust to *routing*: when users want to submit an order on Uniswap, they can either interact directly with specific pools or rely on Uniswaps routing system (or third-party protocols like 1Inch or CoW Swap). The former method is preferable for performing the MEV strategies discussed in the previous sections, whereas the latter leverages specialized router smart contracts to identify the most efficient way to execute the trade, in order to achieve the best possible execution price. Specifically, router smart contracts often trade the same pair on pools with different fee tiers¹⁹ or split the swap into smaller parts and route them across multiple pools (with different pair) and/or multiple DEXs. Therefore, what we find in a specific pool can be part of a path originating from the intention to swap between two other tokens²⁰.

One may ask whether the sign persistence previously observed at the “pools” level survives at the “routing” level – i.e., when accounting for these dynamics. To assess the cause, we track token flows associated with

¹⁹For large swaps, the impact generated on the most favorable pool could make it profitable to execute a portion of the swap on the other pools.

²⁰For example, if the market condition is favorable, a trade from USDC to WBTC can be split into: trade from USDC to USDT; trade from USDT to WETH; trade from WETH to WBTC.

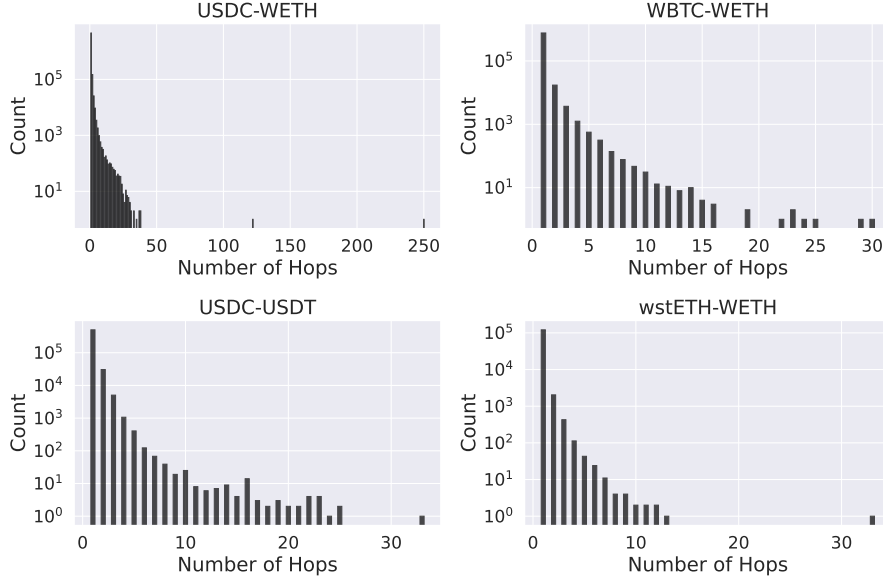


Figure 18: Observed distribution of routing path lengths (in hops) between selected token pairs during routing.

each transaction hash²¹, clearly identifying both the input and output tokens along with the actual direction of the aggregated swap event. In this way, we track the actual ‘intention’ of the user’s swap. Figure 18 shows the number of steps to go from one token to the other for the most active pair in each group. We then apply exactly the same analytical framework employed at the single-pool level. Ultimately, the results are consistent.

5 Statistics of the Liquidity

Liquidity provision is the most important source of revenue for investors in Uniswap – see [32] for a detailed analysis of liquidity at medium-frequency. An LP can add tokens to the pool and earn fees whenever a swap occurs. Therefore, it is important to study the statistical properties of liquidity in pools to acquire insights about how investors choose to act. In this section, we focus our attention on the liquidity minted, as the liquidity burned shows similar patterns. When providing liquidity, the two key quantities of interest are the amount minted and the width of the provision range. We compute their ACF to study the temporal persistence. For the liquidity minted, we separately study active and passive liquidity. Regarding the width range, we only consider non-JIT events, as JITs range is almost always set to the pool’s `tickSpacing` parameter. Generally, we do not find ACF consistent with long-memory. The only exception is the liquidity minted by JIT attackers. Indeed, in the pools where JIT occurrences are sufficient (more than 2,000), we usually find a slow decay.

An interesting pattern observed in DEXs is related to the liquidity droughts. That is, sometimes the active liquidity suddenly drops close to 0. These drops are triggered by swaps rather than burn events, so they are due to the Concentrated Liquidity mechanism. As a reference, one can consider Table 3 in Appendix B, where the minimum active liquidity in the considered time range (pool-by-pool) is compared to the median one. The Table shows that four pools reach a minimum equal to 0, and another four display a minimum not exceeding 100. Generally, large liquidity drops may lead to a significant amplification of the price impact, resulting in abnormal absolute returns and deviations from the CEX benchmark, which in turn lead to an “artificial” increase in the volatility measures (such as the RV), as shown in Figure 19.

However, these deviations from the CEX price are not profitable. Indeed, due to liquidity droughts, any trade would cause a significant impact, which immediately realigns the price with the benchmark and draws the liquidity into the normal regime. Finally, Figure 20 illustrates an example of the effect of liquidity droughts, which result in spikes in the marginal price time series.

5.1 Provision Range vs Volatility

We next investigate the relationship between the liquidity provision range and the volatility. The intuition based on [21], [15], and [12] suggests that, with other market conditions fixed, the optimal (i.e., that which maximizes the expected utility) strategy for the LP is to widen the tick range when the volatility increases. Such reasoning applies only to passive liquidity, i.e., non-JIT mints. Therefore, we empirically study the relation

²¹Usually, a smart routing swap is atomic. Therefore, all the transactions share the same hash.

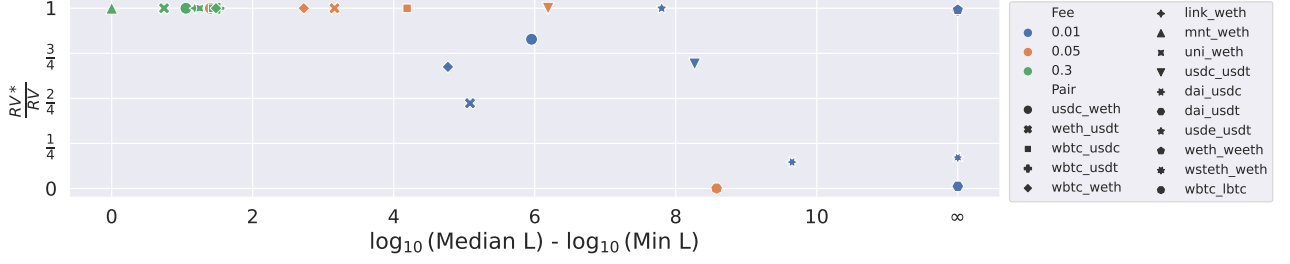


Figure 19: Differences in the Realized Variance (RV) as a function of the strength of liquidity droughts, measured as the difference between the median and minimum log liquidity. RV is for the annualized Realized Variance, computed from marginal prices sampled every 10 minutes over the entire period. RV^* is defined analogously, but only includes swaps whose corresponding active liquidity is above the 5% quantile of the pool’s active liquidity. The infinite value on the x-axis is for those pools whose minimum active liquidity is 0.

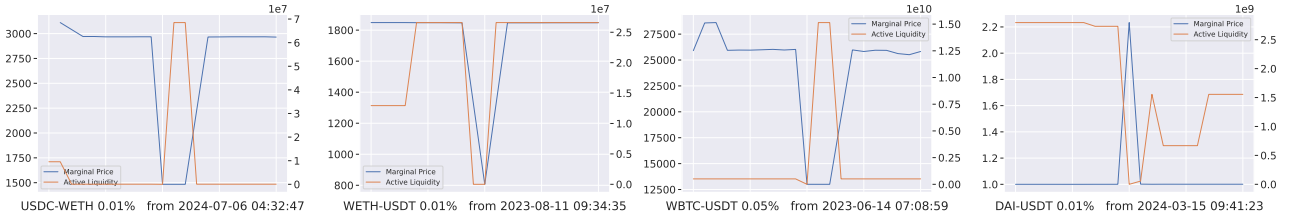


Figure 20: Liquidity droughts and corresponding price peaks. Each subplot reports one example covering 10 events before and after the liquidity drop. The blue line is for the marginal price time series; the orange line is for the active liquidity. The x-label specifies the pool and the date of the first event of the example. The x-axis is in event-time. Two y-axes for each subplot: the left one is for the price scale; the right one is for the liquidity scale.

between volatility and liquidity provision width. Figure 21 shows an example related to the most traded pool in each cluster by plotting the logarithm of the range width against the logarithm of the Realized Variance, which we use as a proxy of the volatility. The RV is computed in a time window of ten minutes before the mint. Similar results are obtained for the other pools.

From the figure, the liquidity provision range is essentially independent of the volatility, resulting in a discrepancy between what we expect from theory and what we see in the markets. This can be due to several factors, such as irrational investors, different assumptions about the market behavior, and changes in the other market conditions (expected return, risk-aversion coefficient). Other experiments with a longer time window (30 minutes) and with different proxies for the volatility (modulated RV, number of swaps, swapped volume) lead to the same conclusion. This evidence aligns with the findings of Capponi et al. [12], who demonstrated that liquidity providers (LPs) select wider price ranges not primarily due to increased exchange rate volatility itself, but rather due to increased risks of adverse selection from arbitrageurs. They leveraged the Silicon Valley Bank (SVB) crisis on March 10, 2023, as a natural experiment to test their model. This event caused a significant depeg of USDC. By comparing LP price ranges before and after the SVB event across Layer 1 (Ethereum) and Layer 2 platforms²² (Arbitrum and Optimism, which feature significantly lower gas fees for range adjustments), they effectively excluded the simpler hypothesis of an unconditional positive relationship between volatility and price range adjustments. Indeed, focusing specifically on the SVB crisis event, we observe an increase in the relationship between volatility and liquidity provision range, as illustrated in Figure 22. This dynamics is not present in random periods of the same length taken from our data.

We carry out further analysis on the relationship between the provision’s range width and the volatility through a cross-sectional study. For each pool, we compute the annualized RV from 10-minute frequency, both with and without removing the peaks associated with liquidity droughts. Furthermore, we calculate the average difference between upper and lower ticks of the mint events, weighted by the amount of liquidity minted. Figure 23 provides plots of these quantities, using marker shapes to distinguish token pairs and colors to distinguish fee tiers.

²²A Layer 2 solution on Ethereum is a secondary framework built atop the main blockchain (Layer 1) that processes transactions off-chain to reduce congestion and lower gas fees. Layer 2 solutions improve scalability and transaction throughput while still benefiting from Ethereum’s robust security and decentralization

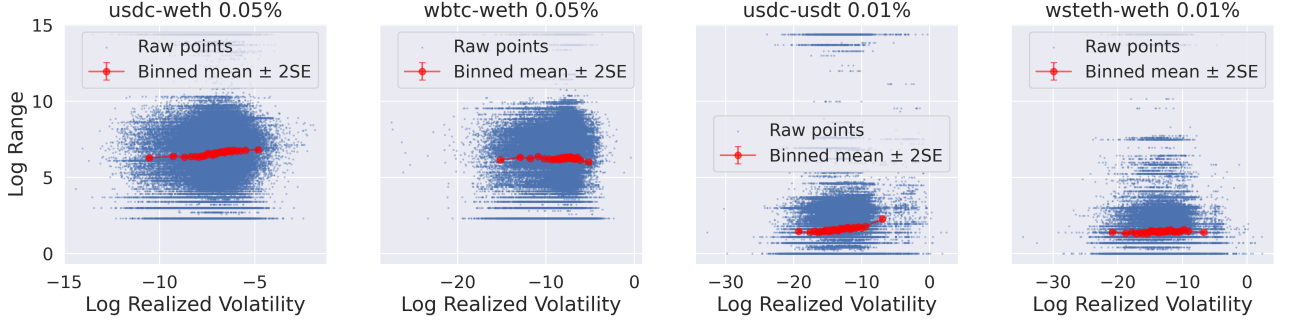


Figure 21: Liquidity provision width (Range) vs RV for the most active pools for each group. The RV is computed over a time window of 10 minutes before the mint. The dotted red line represents the average log provision width together with the standard errors calculated within each of the 30 equal sized bins into which the log RV values have been divided.

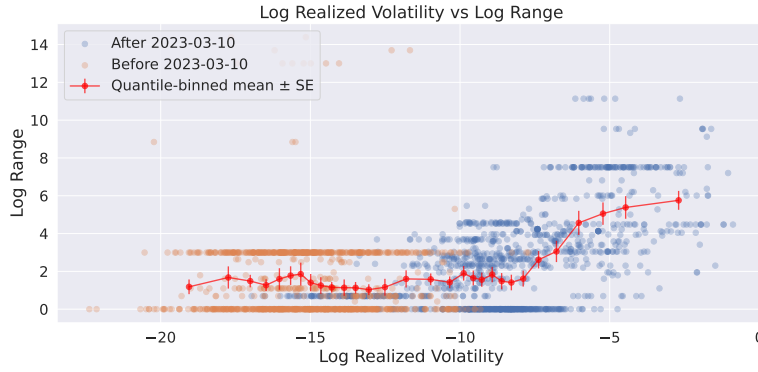


Figure 22: Liquidity Provision Width vs Realized Volatility (RV) for USDT-USDC 0.01% from March 3, 2023 to March 17, 2023, i.e. 7 days before and after the SVB collapse. The RV is computed using a 10-minute time window preceding each mint event. The red points represent the average log provision widths calculated within each of the 30 equally sized bins into which the log RV values have been divided.

The figure suggests a positive correlation between the provision’s range width and the (cleaned) RV. This is confirmed by the Spearman correlation test with permutation [41], whose null hypothesis is the absence of correlation between the variables. Furthermore, the result is stable if we normalize the tick range by the `tickSpacing`, suggesting it is not merely due to the mechanics of liquidity provision. The relationship is, however, weaker than what is typically observed in TradFi²³ – see, for example, [38].

6 Transition probabilities

This section focuses on the interdependence between events by studying the transition probabilities from one event type to another. Indeed, as our goal is to study the collective dynamics occurring on Uniswap, we wish to understand how (and if) a particular event plays a role in triggering another one. To this end, we study the transition probabilities. Specifically, given the events’ time series $\{e_t\}_{t=1}^T$ measured in event-time, we estimate

$$\mathbb{P}(e_{t+l} = E | e_t = E') \quad l = 1, 2, 3, \dots \quad E, E' \in \{\text{Mint}, \text{Burn}, \text{Swap zero}, \text{Swap ch}\} \quad (5)$$

Swaps are partitioned into *Swap zero* and *Swap ch*, according to whether the corresponding tick movement is zero or not. Figure 24 shows the transition probabilities as a function of the lag in the USDC–WETH 0.05% pool.

The gray lines are the conditional probabilities obtained from the whole event’s time series; the dashed gray lines represent the unconditional probabilities. As expected, as the lag l increases, all transition probabilities converge to their unconditional probabilities, although with differing rates. There appears to be a sharp peak at lag 2 in the probabilities conditional to a mint event; similar peaks are in the probabilities of a burn conditional to

²³This statement assumes that the TradFi relationship volatility-spread is a proxy of the DeFi relationship volatility-provision’s range width.

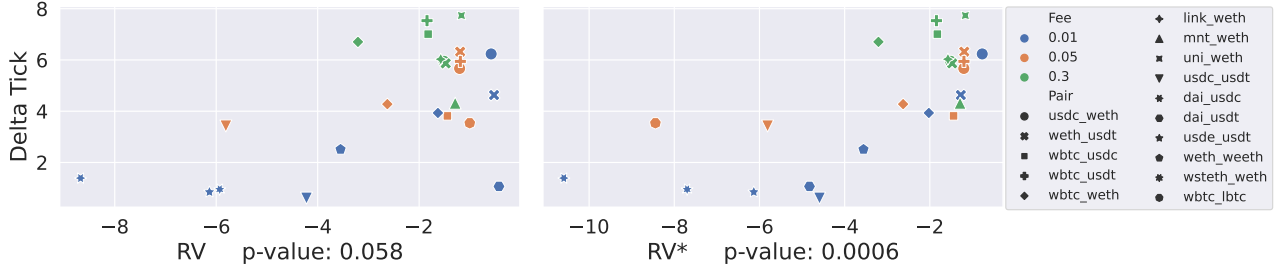


Figure 23: Provision’s range versus Realized Variance, log-log scale. For every pool, the provision’s range is expressed as the liquidity-weighted mean difference between the upper and lower ticks over all mint events. RV is for the Realized Variance (annualized) computed over the whole time series sampled at 10-minute frequency. RV* is computed in the same way, but removing observations corresponding to liquidity droughts (defined as active liquidity below the median $\times 10^{-4}$). Both axes are on a log scale. The p-value on the x-axis label refers to the Spearman correlation test with permutation.

a swap event. We conjecture that this is the result of the large volume of JIT events occurring in the pool. As already stated, a JIT event consists of a specific operational pattern occurring in the same block: mint, (often only one) swap, and burn. Furthermore, they account for about half of the liquidity provision events in almost all the pools. Therefore, it is reasonable that $\arg \max_E (\mathbb{P}(e_{t+1} = E | e_t = \text{Mint}) - \mathbb{P}(e_{t+1} = E))$ is Swap zero (the large amount of the active liquidity minted prevents from tick changes), $\arg \max_E (\mathbb{P}(e_{t+2} = E | e_t = \text{Mint}) - \mathbb{P}(e_{t+2} = E))$ is Burn, and so on. To have a more complete picture, we filter out the effect of MEV strategies by removing them from the data when computing the transition probabilities – the output is plotted with the colored lines, contrasting with the gray lines that correspond to the conditional probabilities obtained without filtering MEV attacks. On a closer examination, all the events appear to elicit themselves with a probability greater than the unconditional one if they have just occurred – that is, on the main diagonal of the plot, the continuous lines lie above the dashed ones. This pattern has already been observed in traditional finance and is known as the *diagonal effect*²⁴. Moreover, mint and burn orders excite each other. The strongest effect is from burn to mint, probably due to position adjustments. Conversely, swaps are not affected by the occurrences of liquidity provision events. Furthermore, swaps with and without tick changes seem to inhibit each other. One possible explanation for this is related to volatility clustering: during high-variance periods, swaps with tick changes are far more probable, whereas during low-volatility periods, swaps without tick movements frequently appear. Finally, the above patterns are also observed in the other pools. Table 6 in Appendix B presents the results for each pool. Deviations from these stylized patterns appear just in five Stable and Synthetic pools, the same ones where volume long-memory is not observed. These are those with fewer events, making it likely that this ambiguity originates from the scarcity of data.

7 Conclusion and Research Perspectives

This study offers a first analysis, at the microstructural level, of Uniswap v3. This market has been chosen due to its academic and industrial relevance and the plethora of information and publicly available data. These features make it a perfect benchmark to analyze the impact of the DEXs’ market structure on the statistical properties observed at high-frequency. Specifically, we focused on twenty-four highly traded pools spanning different types of underlying pairs and fee tiers. We analyzed the economic drivers behind the most widely used strategies, their impact on the pool activity, and the composition of the attackers. Specifically, we highlight that MEV strategies have a dramatic impact on the trading activity, and that the relevance of each specific strategy is strongly affected by the pool fee tier.

The core of this work focuses on price-related time series, where we detect significant patterns that partly follow those observed in TradFi. Nonetheless, the technical implementation of DEXs and the existence of MEV strategies lead to some differences with TradFi. Specifically, we find that swap returns are leptokurtic, with kurtosis (super)linearly decaying when a few swaps are aggregated. In DEXs, we observe a pattern known as echo swaps, characterized by two consecutive swaps placed by the same wallet in opposite directions – a phenomenon not reported in TradFi. The autocorrelation of returns displays multiple peaks at lag 1, reflecting the presence of echo swaps and reverse arbitrage, and at lags 2 and 3, corresponding to sandwich attacks. Contrastingly, in TradFi, only a single lag-one peak is typically observed. Order flow is correlated as well, but herding effects dominate over order splitting, and this pattern is persistent even when aggregating across pools.

²⁴The definition of diagonal effect is “the tendency to see the same order types appear in succession”; see [8]

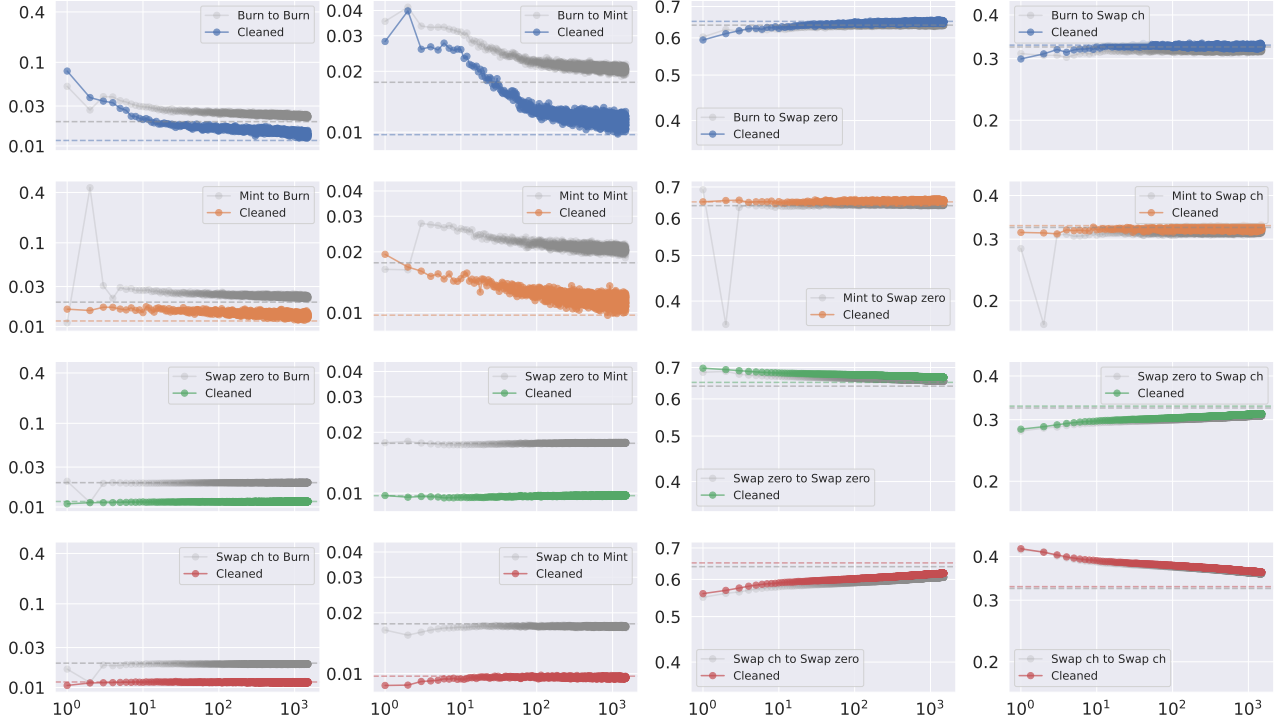


Figure 24: USDC-WETH 0.05% pool - Empirical transition probabilities from one event type to another as a function of the lag l , see Eq. (5). The gray lines are obtained from the whole dataset; the colored lines are obtained by removing MEV strategies from the dataset. By row, there is the conditioning event e_t ; by column, there is the target e_{t+l} . The horizontal dashed lines indicate the unconditional probabilities $\mathbb{P}(e_{t+l} = E)$.

Liquidity often experiences sharp drops due to the Concentrated Liquidity mechanism; although not directly profitable, these drops trigger jumps in the marginal price, producing deviations from the CEX benchmark and artificially inflating volatility. Moreover, while in traditional markets volatility and spread are strongly related, in DEXs the correlation between volatility and the liquidity provision range is very weak. Finally, transition probabilities from one event to another also exhibit distinctive peaks that can be attributed to MEV activity. The appendix corroborates our findings with figures and tables describing the quantitative results.

From our study, there are many opportunities for future work. We cluster some of them into three main directions.

- ▷ General questions about pools' agents, such as the presence of inter-pool patterns and correlation or causality effects between different pools. Furthermore, it would be extremely valuable to consider Ethereum Layers 2, like Arbitrium or Base. Due to their lower gas fees and the increased number of transactions validated per unit of time, they could reveal other intriguing statistical properties in the DeFi markets.
- ▷ Deeper analysis of the LTs' behavior and their activity. For example, a detailed analysis of router smart contracts, their strategies, and how they impact MEV searchers' activity, as well as a study of the impact of liquidity droughts.
- ▷ MEV searchers' advanced strategies and their impact – e.g., an exhaustive explanation of the echo swaps economic drivers, as discussed in Subsection 3.3. Moreover, what is the cause of long-memory ACF in the liquidity minted by JIT attacks? Finally, it may be interesting to carry out an analysis of the victims of MEV attacks. Are there agents who let them be jitted on purpose? Are MEV searchers, in turn, frequently victims of attacks by other searchers?

In conclusion, this study represents a step toward a comprehensive understanding of DEX microstructure, bridging the gap between traditional market analysis and DeFi. Nonetheless, additional effort is essential to fully understand the mechanisms driving DEX behavior. As this new ecosystem evolves, such foundational research will be critical for theoretical advancements and practical applications.

Acknowledgements

FL acknowledges support from the grant PRIN2022 DD N. 104 of February 2, 2022 “Liquidity and systemic risks in centralized and decentralized markets”, codice proposta 20227TCX5W - CUP J53D23004130006 funded by the European Union NextGenerationEU through the Piano Nazionale di Ripresa e Resilienza (PNRR). SJ acknowledges support from the Natural Sciences and Engineering Research Council of Canada (RGPIN-2024-04317). We also thank Davide Barone for the insightful discussion about MEV.

References

- [1] H. Adams, M. Salem, N. Zinsmeister, S. Reynolds, A. Adams, W. Pote, M. Toda, A. Henshaw, E. Williams, and D. Robinson. Uniswap v4 core. *Technical report, Uniswap*, 2024.
- [2] H. Adams, N. Zinsmeister, and D. Robinson. Uniswap v2 core, 2020. *Technical report, Uniswap*, 2020.
- [3] H. Adams, N. Zinsmeister, M. Salem, R. Keefer, and D. Robinson. Uniswap v3 core. *Technical report, Uniswap*, 2021.
- [4] L. S. Alotaibi and S. S. Alshamrani. Smart contract: Security and privacy. *Computer Systems Science & Engineering*, 38(1), 2021.
- [5] G. Angeris, H.-T. Kao, R. Chiang, C. Noyes, and T. Chitra. An analysis of uniswap markets. *arXiv preprint arXiv:1911.03380*, 2021.
- [6] F. J. Anscombe and W. J. Glynn. Distribution of the kurtosis statistic b_2 for normal samples. *Biometrika*, 70(1):227–234, 1983.
- [7] A. Barzykin, P. Bergault, and O. Guéant. Market-making by a foreign exchange dealer. *Risk (Cutting Edge)*, 2022.
- [8] B. Biais, P. Hillion, and C. Spatt. An empirical analysis of the limit order book and the order flow in the paris bourse. *The Journal of Finance*, 50(5):1655–1689, 1995.
- [9] J.-P. Bouchaud, J. D. Farmer, and F. Lillo. How markets slowly digest changes in supply and demand. In *Handbook of financial markets: dynamics and evolution*, pages 57–160. Elsevier, 2009.
- [10] V. Buterin, E. Conner, R. Dudley, M. Slipper, I. Norden, and A. Bakhta. EIP-1559: Fee market change for ETH 1.0 chain. <https://eips.ethereum.org/EIPS/eip-1559>, 2019. Ethereum Improvement Proposal 1559.
- [11] A. Capponi, A. Cartea, and F. Drissi. Do longer block times impair market efficiency in decentralized markets? *Available at SSRN 5290232*, 2025.
- [12] A. Capponi and R. Jia. Liquidity provision on blockchain-based decentralized exchanges. *Available at SSRN 3805095*, 2025.
- [13] A. Capponi, R. Jia, and B. Zhu. The paradox of just-in-time liquidity in decentralized exchanges: More providers can sometimes mean less liquidity. *arXiv preprint arXiv:2311.18164*, 2023.
- [14] Á. Cartea, F. Drissi, and M. Monga. Decentralised finance and automated market making: Execution and speculation. *arXiv preprint arXiv:2307.03499*, 2023.
- [15] Á. Cartea, F. Drissi, and M. Monga. Decentralized finance and automated market making: Predictable loss and optimal liquidity provision. *SIAM Journal on Financial Mathematics*, 15(3):931–959, 2024.
- [16] A. Chakraborti, I. M. Toke, M. Patriarca, and F. Abergel. Econophysics review: I. empirical facts. *Quantitative Finance*, 11(7):991–1012, 2011.
- [17] CoinMarketCap. Top cryptocurrency decentralized exchanges ranked. <https://coinmarketcap.com/rankings/exchanges/dex/>, 2025. Accessed: 2025-06-30.
- [18] R. D’agostino and E. S. Pearson. Tests for departure from normality. empirical results for the distributions of b^2 and $\sqrt{b}$. *Biometrika*, 60(3):613–622, 1973.

- [19] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels. Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges. *arXiv preprint arXiv:1904.05234*, 2019.
- [20] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels. Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges. *CoRR*, abs/1904.05234, 2019.
- [21] Z. Fan, F. J. Marmolejo-Cossío, B. Altschuler, H. Sun, X. Wang, and D. Parkes. Differential liquidity provision in uniswap v3 and implications for contract design. In *Proceedings of the Third ACM International Conference on AI in Finance*, pages 9–17, 2022.
- [22] Flashbots. MEV-Boost in a Nutshell. Flashbots MEV-Boost Overview.
- [23] Flashbots. Overview of flashbots auction. Flashbots Documentation.
- [24] S. Jaimungal, Y. F. Saporito, M. O. Souza, and Y. Thamsten. Optimal trading in automatic market makers with deep learning. *arXiv preprint arXiv:2304.02180*, 2023.
- [25] B. LeBaron and R. Yamamoto. Long-memory in an order-driven market. *Physica A: Statistical Mechanics and its Applications*, 383(1):85–89, 2007. Econophysics Colloquium 2006 and Third Bonzenfreies Colloquium.
- [26] D. Li, K. Zhang, L. Wang, and G. Du. A geth-based real-time detection system for sandwich attacks in ethereum. *Discover Computing*, 27(1):11, 2024.
- [27] F. Lillo and J. D. Farmer. The long memory of the efficient market. *Studies in nonlinear dynamics & econometrics*, 8(3), 2004.
- [28] H. Materwala, S. M. Naik, A. Taha, T. A. Abed, and D. Svetinovic. Maximal extractable value in decentralized finance: Taxonomy, detection, and mitigation. *arXiv preprint arXiv:2411.03327*, 2024.
- [29] B. Öz, D. Sui, T. Thiery, and F. Matthes. Who wins ethereum block building auctions and why? *arXiv preprint arXiv:2407.13931*, 2024.
- [30] M. Podolskij and M. Vetter. Estimation of volatility functionals in the simultaneous presence of microstructure noise and jumps. *Bernoulli*, 15(3):634 – 658, 2009.
- [31] J. O. Ramsay and B. W. Silverman. *Functional Data Analysis*. Springer Series in Statistics. Springer New York, NY, 2013.
- [32] J. Risk, S.-N. Tung, and T.-H. Wang. Dynamics of liquidity surfaces in uniswap v3. *arXiv preprint arXiv:2509.05013*, 2025.
- [33] A. C. Silva, S.-N. Tung, and W.-R. Chen. Stylized facts in web3. *Frontiers of Mathematical Finance*, 3(4), 2024.
- [34] N. Szabo. Formalizing and securing relationships on public networks. *First monday*, 1997.
- [35] B. Tóth, I. Palit, F. Lillo, and J. D. Farmer. Why is equity order flow so persistent? *Journal of Economic Dynamics and Control*, 51:218–239, 2015.
- [36] X. Wan and A. Adams. Just-in-time liquidity on the uniswap protocol. *Available at SSRN 4382303*, 2022.
- [37] M. Wątarek, M. Królczyk, J. Kwapien, T. Stanisz, and S. Drożdż. Approaching multifractal complexity in decentralized cryptocurrency trading. *Fractal and Fractional*, 8(11):652, 2024.
- [38] M. Wyart, J.-P. Bouchaud, J. Kockelkoren, M. Potters, and M. Vettorazzo. Relation between bid–ask spread, impact and volatility in order-driven markets. *Quantitative finance*, 8(1):41–57, 2008.
- [39] X. Xiong, Z. Wang, W. Knottenbelt, and M. Huth. Demystifying just-in-time (jit) liquidity attacks on uniswap v3. In *2023 5th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, pages 1–8. IEEE, 2023.
- [40] Y. Zhang, S. Chan, J. Chu, X. Liao, and M. Helu. Stylized facts of decentralized finance (defi). In *Artificial Intelligence and Beyond for Finance*, pages 289–314. World Scientific, 2024.
- [41] D. Zwillinger and S. Kokoska. *CRC standard probability and statistics tables and formulae*. Crc Press, 1999.

A Uniswap - Mathematical Details

This appendix is intended for the inexperienced reader approaching Uniswap markets for the first time. Here, we provide the mathematical rules behind Uniswap markets. First, we describe Uniswap v2. Then, we move to Uniswap v3, introducing Concentrated Liquidity and how it changes the game.

A.1 Uniswap v2

Uniswap v2 [2] is based on a specific type of Automated Market Makers named Constant Product Market Makers (CPMMs). In such systems, liquidity is stored in pools that operate independently, each containing a pair of tokens -let's denote them as X and Y. The latter is used as the numeraire. Denote the reserves in event-time t as R_t^X and R_t^Y . CPMMs allow three core events: swap, mint, and burn.

A swap refers to the exchange of the two tokens: the Liquidity Taker (LT) deposits x_{swap} units of token X (or Y) into the pool and receives y_{swap} units of token Y (or X) in return. The output y_{swap} is computed to keep constant the product between reserves, net of the transaction fee paid by the LT. This is a fixed fraction $\phi \in (0, 1)$ of the volume x_{swap} . It is intrinsic in the pool, so it cannot change (unless a decision of the pool governance is made in very special situations; however, we can neglect this chance and consider ϕ as a constant). Thus, the ultimate equation is:

$$(R_t^X + (1 - \phi)x_{swap})(R_t^Y - y_{swap}) = R_t^X R_t^Y = L_{tot,t}^2 \implies y_{swap} = x_{swap} \frac{(1 - \phi)R_t^Y}{R_t^X + (1 - \phi)x_{swap}} \quad (6)$$

$L_{tot,t}$ is referred to as the total liquidity in the pool at time t . The transaction price is defined as the ratio $\frac{y_{swap}}{x_{swap}} = \frac{(1 - \phi)R_t^Y}{R_t^X + (1 - \phi)x_{swap}}$. If we neglect the fee and take the limit as $x \rightarrow 0$, the transaction price approaches $\frac{R_t^Y}{R_t^X}$, which is referred to as the marginal price S_t , and it is used as the reference price. Moreover, it is worth noting that, in Uniswap v2, the fee paid by the LT is reinjected into the pool as additional liquidity. In this case, we talk about compounding fees. That is, even if in the price computation in Eq. (6) the pair $(R_t^X + (1 - \phi)x_{swap}, R_t^Y - y_{swap})$ is used, the actual pool reserves of token X are updated as $R_t^X + x_{swap}$ and $R_t^Y - y_{swap}$. Thus, the new marginal price is $\frac{R_t^X R_t^Y}{(R_t^X + (1 - \phi)x_{swap})(R_t^X + x_{swap})}$.

A mint operation involves supplying liquidity to the pool-i.e., adding a specified positive amount of both tokens X and Y, denoted as x_{mint} and y_{mint} . Minting new liquidity requires the pool price to remain unchanged. Thus, for any liquidity addition, the following condition must hold:

$$\frac{R_t^Y}{R_t^X} = \frac{R_t^Y + y_{mint}}{R_t^X + x_{mint}} \implies \frac{y_{mint}}{x_{mint}} = \frac{R_t^Y}{R_t^X} \quad (7)$$

An LP's stake in the pool is proportional to the liquidity they provide L . Elementary algebra then shows that the additional liquidity L is given by $L = \sqrt{x_{mint}y_{mint}}$. Indeed, the constancy of price enforces $R_t^X y_{mint} = R_t^Y x_{mint}$, so we have that $R_t^X y_{mint} = R_t^Y x_{mint} = \sqrt{R_t^X y_{mint}} \sqrt{R_t^Y x_{mint}} = \sqrt{R_t^X R_t^Y} \sqrt{x_{mint}y_{mint}}$. Therefore:

$$(R_t^X + x_{mint})(R_t^Y + y_{mint}) = (L_{tot,t} + L)^2 = \left(\sqrt{R_t^X R_t^Y} + L \right)^2 \implies L = \sqrt{x_{mint}y_{mint}} \quad (8)$$

For analyzing the position of an individual LP, it is often sufficient to relate their personal liquidity L to the specific quantities of tokens deposited, disregarding the aggregate pool reserves.

Furthermore, it is more convenient to describe the pool (respectively, the LP position) in terms of total liquidity and price $(L_{tot,t}, S_t)$ (respectively, the LP's stake and the price (L, S_t)) rather than token balances (R_t^X, R_t^Y) (respectively, the tokens minted (x_{mint}, y_{mint})). Indeed, the former representation is more practical for analytical purposes and the smart contract reasons in terms of $L_{tot,t}$ and S_t . Temporal indices may be omitted for simplicity when no ambiguity arises. Finally, as for the burn operations, one can exploit the same computation as above, with x_{burn} and y_{burn} negative quantities.

A.2 Uniswap v3

The main limitation of Uniswap v2 is the lazy liquidity -i.e., the LPs are forced to provide liquidity and spread their tokens over the whole price interval $(0, +\infty)$. Uniswap v3 [3] is built upon and generalizes its previous version. The main innovation is the Concentrated Liquidity feature that allows LPs to select a specific position's range $R = [S_l, S_u) \subset \mathbb{R}_+$ within which to provide liquidity. Uniswap v2 can be viewed as the limiting

case where $S_l = 0$ and $S_u = \infty$. Thus, we first analyze the LP point of view. An LP is said to be active²⁵-contributing liquidity to the pool- whenever the current price falls within the chosen interval R . LP activity can be interpreted as follows: when the price lies within R , the LP holds both X and Y, facilitating trades for LTs. If the price drops below S_l or rises above S_u , the LPs entire liquidity is converted fully into X or Y, respectively, thus rendering them inactive. The total pool liquidity L_{tot} and the total reserves become the sum of those of all active positions, and may fluctuate as swaps alter the price and thus the set of active LPs. Thus, the pool's liquidity is no longer constant. It can rather be viewed as a function of the price. This function is sometimes referred to as the liquidity profile. In the limiting Uniswap v2 case, all the provision positions have a range of $(0, \infty)$, thus the liquidity profile is a constant function.

To understand how the Concentrated Liquidity effectively increases the utility of the reserves, let's consider an LP minting $\tilde{x}$ units of X and $\tilde{y}$ units of Y into the position's range $R = [S_l, S_u]$. $\tilde{x}$ and $\tilde{y}$ are the real reserves -i.e., the tokens actually injected into the pool. However, the pool contract works in terms of artificial quantities known as the virtual reserves for X and Y, defined as $x_{mint} = \tilde{x} + \frac{L}{\sqrt{S_u}}$ and $y_{mint} = \tilde{y} + L\sqrt{S_l}$, where L is the virtual liquidity minted, which is defined as the positive root of the following:

$$x_{mint}y_{mint} = (\tilde{x} + \frac{L}{\sqrt{S_u}})(\tilde{y} + L\sqrt{S_l}) = L^2 \quad \implies \quad L = \frac{\tilde{x}\sqrt{S_l} + \frac{\tilde{y}}{\sqrt{S_u}} + \sqrt{(\tilde{x}\sqrt{S_l} - \frac{\tilde{y}}{\sqrt{S_u}})^2 + 4\tilde{x}\tilde{y}}}{2(1 - \sqrt{\frac{S_l}{S_u}})} \quad (9)$$

Specifically, $x_{mint} \geq \tilde{x}$, $y_{mint} \geq \tilde{y}$, and $L \geq \sqrt{\tilde{x}\tilde{y}}$ -that is, the smart contract assigns us more tokens than those actually provided. The equality is recovered in the limit cases $S_u = \infty$ and $S_l = 0$ -i.e., when the concentrated liquidity feature is not exploited. When the liquidity is active, the virtual reserves ratio equals the marginal price -that is, $S = \frac{y_{mint}}{x_{mint}}$. Thus, we can rearrange $x_{mint} = \frac{L}{\sqrt{S}}$ and $y_{mint} = L\sqrt{S}$, which lead to $\tilde{x} = L(\frac{1}{\sqrt{S}} - \frac{1}{\sqrt{S_u}})$ and $\tilde{y} = L(\sqrt{S} - \sqrt{S_l})$. If $S < S_l$, then the real reserve of token Y is $\tilde{y} = 0$. As for the price constraint, it will be active only if the liquidity touches the position's range -i.e., the price reaches S_l . Thus, $S_l = \frac{L\sqrt{S_l}}{x_{mint}}$, which leads to $\tilde{x} = L(\frac{1}{\sqrt{S_l}} - \frac{1}{\sqrt{S_u}})$. A similar reasoning holds if $S > S_u$. Summarizing, the mapping from (L, S) to the true reserves $(\tilde{x}, \tilde{y})$ is described as:

$$(\tilde{x}, \tilde{y}) = \begin{cases} \left(L\left(\frac{1}{\sqrt{S_l}} - \frac{1}{\sqrt{S_u}}\right), 0 \right) & \text{if } S < S_l \\ \left(L\left(\frac{1}{\sqrt{S}} - \frac{1}{\sqrt{S_u}}\right), L(\sqrt{S} - \sqrt{S_l}) \right) & \text{if } S_l \leq S \leq S_u \\ (0, L(\sqrt{S_u} - \sqrt{S_l})) & \text{if } S > S_u \end{cases} \quad (10)$$

The introduction of Concentrated Liquidity forced a technical adjustment of the pools' contracts that now implement a logic based on ticks. That is, the price space is partitioned into a discrete set of points defined as 1.0001^j , with $j \in \mathbb{N}$. The integer exponents j are known as ticks. So, at each price S lying in the continuous set²⁶ $\mathbb{R}_+$, it is associated the tick $\lfloor \log_{1.0001}(S) \rfloor$ in the discrete set $\mathbb{N}$. When an LP provides liquidity, the range bounds S_l and S_u have to belong to the set $\{1.0001^{j \cdot ts} | j \in \mathbb{N}\}$, where $ts \in \mathbb{N}$ is the `tickSpacing` parameter (fixed by the pool's governance). Thus, the range bounds are identified with the corresponding ticks. Values in $\{j \cdot ts | j \in \mathbb{N}\}$ are called initializable ticks. Initializable ticks are said to be initialized if there is a liquidity provision position having that tick as the lower or upper bound. Finally, it is worth underlining that the Concentrated Liquidity feature makes it infeasible to use the Uniswap v2 logic based on compounding fees. Thus, the fees collected by LPs in Uniswap v3 are stored separately and not reinjected into the pool. This way of collecting fees is referred to as non-compounding fees. This technicality could make the difference when looking at some mathematical properties of the markets. Thus, it is not certain that the results holding for Uniswap v2 pools are still valid in Uniswap v3.

Now, we take the point of view of an LT willing to place a swap order. Without loss of generality, assume that the LT wants to swap x_{swap} units of token X with token Y, thus moving the price downward. Let j be the price tick, l the greatest initialized tick smaller than j , u the smallest initialized tick greater than j , and R^X and R^Y the virtual reserves of the pool in $[l, u]$. By using the Uniswap v2 logic, we can obtain the output amount of a swap that does not pass the initialized ticks. Then, taking into account the non-compounding fees mechanism, it is easy to compute the price after such a swap. Ultimately, we recover the amount x^* that, if

²⁵In the literature, there is a duality in the definition of "active" LPs. Here, "active" means that the price is inside their price range R . In other contexts, such as the Just-in-Time liquidity analyzed in Subsection 3.1, "active" means that the LP acts as a reaction to a swap pending in the memory-pool. See Section 3.1 of the main text for the full details. Nonetheless, we use the first meaning only in the current Appendix A.2. Elsewhere, we always refer to the second meaning.

²⁶Here, "continuous" has to be intended net of the numerical approximation.

traded, moves the marginal price to l :

$$\text{New price v3: } \frac{R^X R^Y}{(R^X + (1 - \phi)x_{\text{swap}})^2} \quad x^* = \sqrt{R^X} \frac{\sqrt{R^X + 1.0001^l R^Y} - \sqrt{R^X}}{1 - \phi} \quad (11)$$

As long as $x_{\text{swap}} \leq x^*$, the price tick after the swap is still inside $[l, u)$. Thus, there is no liquidity adjustment and the output is computed as done in Eq. (6). Otherwise, the swap is split into two components. First, x^* is traded. Then, the active liquidity is adjusted accordingly, causing an additional gas fee to be paid by the LT; lastly, the remaining part $x_{\text{swap}} - x^*$ is traded as if it were a new swap.

Finally, it is interesting to see how the non-compounding fee mechanism of Uniswap v3 causes some difference with respect to Uniswap v2. Indeed, let's consider a swap in a v3 pool. Assume that the swap does not pass any initialized tick -i.e., it does not cause a change in the liquidity profile. Then, the price impact, measured as the variation in the marginal price, is smaller, *ceteris paribus*, than that on the v2 pool. This is because more units of token X are put into the pool due to the compounding fees mechanism. For example, let us consider the order splitting problem, whose relevance is empirically analyzed in Subsection 4.5 of the main text. Regarding the Uniswap v2 market, [5] shows that splitting an order is more expensive than executing it one-shot. That is, if an LT wants to swap $x_1 + x_2$ units of token X for token Y, the output generated by trading them at once is, *ceteris paribus*, bigger than the sum of those obtained by first swapping x_1 and then x_2 . Instead, when switching to Uniswap v3, this result does not hold. With straightforward computations, it is possible to show that the two quantities are precisely the same; thus, splitting the order is not more expensive than the one-shot execution²⁷.

B General Events Patterns

This appendix discusses some general patterns found in the event time series. First, we provide a detailed summary of the dataset used; then, we analyze the transition probabilities from one event type to another; finally, we have a look at the daily seasonality in the market activity. Figure 2 summarizes the main properties of the pools considered in our analysis.

Section 6 of the main text points out that the transition probabilities are strongly affected by the presence of MEV transactions, especially JIT strategies. Once cleaned for these, we notice mainly four patterns: diagonal effect, mint is more likely to occur after a burn, swap occurrences are independent of liquidity provision events, and swaps with or without a tick change inhibit each other. Table 4 summarizes occurrences or deviations from these patterns.

B.1 Intraday Patterns

Now, we focus on the intraday patterns. The analysis concerns the market activity measured as the number of events or price volatility. DEXs are continuously open 24 hours²⁸ and trading is possible also during holidays. Thus, it makes sense to search for intraday patterns driven by changes in worldwide trading activity during the day. By examining these patterns, we can uncover the underlying mechanisms of liquidity, volatility, and investor behavior unique to DeFi platforms.

To study the intraday pattern, we first divide a day into time windows of 10 minutes and compute the percentage quantities of the number of events for each window. These can be viewed as discrete observations of an underlying market intensity process. Then, we project onto the Fourier space and perform an Functional Principal Components Analysis (FPCA) [31] to obtain the principal modes of variation in the data.

Figure 25 shows, as an example, the result for the USDC-WETH 0.05% pool, while the results of the other pools are in Figure 26 of the Appendix B. The left panel shows the mean of the market activity process and the right panel shows the first three Functional Principal Components (FPCs) that explain 86% of the total variance.

As for the FPCs, the first component explains 58% of the variance and is roughly flat. Thus, it corresponds to the variation of the market activity process between different days. Instead, the other FPCs can indicate the impact on the market intensity process of the European and American sessions (second component) and the Tokyo and European sessions (third component).

As for the mean process, we find that the USDC-WETH 0.05% pool follows a common pattern shared by several pools, mostly in Normal and Volatile clusters. They are characterized by a decrease in market activity between 8 and 10 am UTC time (corresponding to the close of Sydney and Tokyo sessions) and an increase between 3 and 5 pm. As different pools share this pattern, we can conclude it does not appear to be a matter of chance.

²⁷Nonetheless, when the gas fees are considered, we find that the split order is more expensive than the one-shot one.

²⁸According to this perspective, Foreign Exchanges (FXs) are the TradFi market most similar to DEXs. As intra-day patterns corresponding to different market sessions are noticed in FXs, we are motivated to investigate similar behavior in our dataset.

Pool	Start	# Swaps	S/B	S/T	# Mint	# Burn	Wait B	# Wallets
USDC-WETH 0.01%	2023-01-01	861,917	1.44	1.28	3,009	3,243	8.69	189,594
USDC-WETH 0.05%	2023-01-01	4,115,698	1.56	2.97	75,449	83,461	1.95	824,769
USDC-WETH 0.3%	2023-01-01	238,996	1.10	1.61	9,339	17,629	21.93	75,447
WETH-USDT 0.01%	2023-02-22	1,917,101	1.61	1.43	6,970	7,441	4.05	404,850
WETH-USDT 0.05%	2023-01-01	2,826,625	1.37	2.20	15,641	17,134	2.53	640,902
WETH-USDT 0.3%	2023-01-01	420,329	1.10	2.46	13,757	16,524	13.00	149,109
WBTC-USDC 0.05%	2023-01-01	205,205	1.14	1.09	2,812	3,058	28.61	36,743
WBTC-USDC 0.3%	2023-01-01	91,080	1.07	1.09	5,236	6,404	56.02	14,035
WBTC-USDT 0.05%	2023-03-31	66,454	1.12	1.08	586	603	76.41	17,784
WBTC-USDT 0.3%	2023-01-01	96,804	1.06	1.13	3,068	3,529	54.75	17,791
WBTC-WETH 0.01%	2023-01-03	138,521	1.38	1.66	935	953	51.69	50,036
WBTC-WETH 0.05%	2023-01-01	723,345	1.21	1.77	12,478	14,460	8.53	171,079
WBTC-WETH 0.3%	2023-01-01	128,129	1.09	1.60	11,562	15,964	37.01	26,575
LINK-WETH 0.3%	2023-01-01	241,087	1.07	1.25	15,211	18,416	21.18	35,812
MNT-WETH 0.3%	2023-07-17	64,877	1.07	1.56	418	410	62.62	23,647
UNI-WETH 0.3%	2023-01-01	156,546	1.05	1.17	6,401	7,990	32.71	22,188
USDC-USDT 0.01%	2023-01-01	685,117	1.12	19.94	9,095	10,032	8.42	232,696
USDC-USDT 0.05%	2023-01-01	72,706	1.41	2.76	1,306	1,548	98.28	32,874
DAI-USDC 0.01%	2023-01-01	162,182	1.06	131.43	917	924	33.79	55,482
DAI-USDT 0.01%	2023-01-01	91,725	1.06	5.08	718	841	59.87	33,924
USDe-USDT 0.01%	2023-11-23	144,103	1.16	4.58	2,778	2,854	22.94	53,749
WETH-weETH 0.01%	2024-01-08	45,166	1.16	18.20	207	231	65.90	20,457
wstETH-WETH 0.01%	2023-01-24	153,790	1.10	11.74	3,545	3,823	35.01	36,203
WBTC-LBTC 0.05%	2024-06-18	38,577	1.14	49.14	112	196	41.45	28,749

Table 2: Summary of the dataset used. **Start** is for the first date in the pool (different from 2023-01-01 only if the pool has been initialized after this date). **S/B** and **S/T** are for the average number of swaps per block and tick change. **Wait B** is for the average number of blocks elapsed between two blocks containing an event related to the pool.

Moreover, a similar result holds when changing the window length or considering the returns’ volatility as a proxy for the market activity. To focus this point, we extend the previous analysis by showing the results obtained for all the examined pools and introducing the Modulated Realized Variance (MRV) as a proxy of the market activity. Specifically, Figure 26 shows the mean value (across different days) of the projection of the percentage market activity in the Fourier space. It has been normalized by dividing by the daily mean. In this way, the activity curves of all the pools have the same mean and are comparable. As shown, most of the Normal pools share a common pattern made up of a drop in the pool activity around 08 am (UTC time) and an increase between 03 and 05 pm. The same behavior is noticeable in some of the Volatile and Stable pools, even if it is much lighter.

Another analysis about intraday patterns has been carried out by using the MRV as a proxy of the market activity. Specifically, the MRV is used to filter the integrated variance when the data are affected by noise -see [30] for further details. The remainder of the pipeline is the same as before. The mean process is shown in Figure 27 and is roughly coherent with what is observed in Figure 26.

Pool	Q0.1 S	Q0.5 S	Q0.9 S	std(r)	Kurt	RV	RV*	L min	Q0.5 S	L max
USDC-WETH 0.01%	1595.7	2545.6	3489.4	4.46	53.9	0.5594	0.4632	-0.2	5.7	10.3
USDC-WETH 0.05%	1599.2	2326	3544.6	0.06	2.2	0.3006	0.3006	5.8	7.3	9.6
USDC-WETH 0.3%	1617.8	2432	3555.3	0.12	1.1	0.2208	0.2208	5.8	6.9	8.8
WETH-USDT 0.01%	2302.1	2747.3	3623.4	2.05	69.0	0.5926	0.2802	0.9	6.0	10.1
WETH-USDT 0.05%	1623.1	2247.8	3554.3	0.17	156.6	0.3037	0.3037	3.5	6.7	9.2
WETH-USDT 0.3%	1654.6	2518.2	3614.5	0.09	0.9	0.2284	0.2284	6.0	6.7	8.6
WBTC-USDC 0.05%	26266	54441	80225	5.19	19.5	0.2363	0.2363	5.7	10.0	13.6
WBTC-USDC 0.3%	25685	57795	90956	0.18	0.2	0.1616	0.1616	9.5	10.9	13.0
WBTC-USDT 0.05%	29299	72238	98477	3.43	0.8	0.3049	0.3016	-inf	10.3	13.3
WBTC-USDT 0.3%	26784	61377	94456	0.21	1.6	0.1581	0.1581	9.2	10.8	12.6
WBTC-WETH 0.01%	20.852	23.396	25.065	3.22	18.8	0.1959	0.1322	22.0	27.2	30.7
WBTC-WETH 0.05%	14.508	18.818	26.402	0.11	84.7	0.0723	0.0723	24.9	27.7	30.1
WBTC-WETH 0.3%	14.331	17.982	25.349	0.08	2.2	0.0405	0.0405	26.6	28.0	29.3
LINK-WETH 0.3%	141.2	213.53	264.29	0.15	2.4	0.2071	0.2071	22.6	23.9	26.2
MNT-WETH 0.3%	2770.2	3780.1	4374.2	0.26	0.7	0.2754	0.2754	23.4	23.8	25.4
UNI-WETH 0.3%	239.62	341.17	409.04	0.19	0.8	0.3116	0.3116	22.3	23.6	26.1
USDC-USDT 0.01%	0.99908	0.99988	1.0007	1.89	135.0	0.0147	0.0102	8.5	16.6	18.1
USDC-USDT 0.05%	0.91193	0.99859	1.0004	5.23	9.5	0.0030	0.0030	9.2	15.0	16.9
DAI-USDC 0.01%	0.99996	1	1.0001	0.28	80.0	0.0002	0.0000	2.0	11.7	12.1
DAI-USDT 0.01%	0.99924	0.99991	1.0007	18.05	11.0	0.6527	0.0080	-inf	9.4	11.5
USDe-USDT 0.01%	0.99885	1.0012	1.0019	1.12	41.9	0.0022	0.0022	2.0	9.2	11.3
WETH-weETH 0.01%	1.043	1.0469	1.0544	3.34	6.0	0.0287	0.0285	-inf	24.7	25.6
wstETH-WETH 0.01%	1.1389	1.1669	1.1859	4.10	24.1	0.0027	0.0005	-inf	25.2	26.2
WBTC-LBTC 0.05%	0.99666	0.99834	0.99966	0.91	5.4	0.3671	0.0002	4.2	12.7	13.3

Table 3: Summary of the price and liquidity time series. **Q0.1**, **Q0.5**, and **Q0.9** are for the 0.1, 0.5, and 0.9 quantiles. **S** and **L** are for the (marginal) price and $\log_{10}$ (Liquidity) time series. **std(r)** and **Kurt** are for the standard deviation ($\cdot 10^2$) and the excess kurtosis ($\cdot 10^{-3}$) of the returns unconditional distribution. **RV** is for the Realized Variance (annualized) computed using the marginal prices sampled from the entire period every 10 minutes. **RV*** is the same, but computed by only considering prices such that the corresponding active liquidity is at least equal to the 5% quantile of the historical active liquidity for that pool. Extremely small values of the minimum $\log_{10}$ (Liquidity) in the **L min** column highlight the presence of the liquidity droughts discussed in the main text; their effect is highlighted by the difference between the columns **RV** and **RV***.

Pool	Diag Eff.	Burn Mint	Indep. Swaps	Swaps Block	Pool	Diag Eff.	Burn Mint	Indep. Swaps	Swaps Block
USDC-WETH 0.01%	Yes	Yes	Yes	Yes	WBTC-WETH 0.3%	Yes	Yes	Yes	Yes
USDC-WETH 0.05%	Yes	Yes	Yes	Yes	LINK-WETH 0.3%	Yes	Yes	Yes	Yes
USDC-WETH 0.3%	Yes	Yes	Yes	Yes	MNT-WETH 0.3%	No	No	Yes	Yes
WETH-USDT 0.01%	Yes	Yes	Yes	Yes	UNI-WETH 0.3%	Yes	Yes	Yes	Yes
WETH-USDT 0.05%	Yes	Yes	Yes	Yes	USDC-USDT 0.01%	Yes	Yes	Yes	Yes
WETH-USDT 0.3%	Yes	Yes	Yes	Yes	USDC-USDT 0.05%	Yes	Yes	Yes	Yes
WBTC-USDC 0.05%	Yes	Yes	Yes	Yes	DAI-USDC 0.01%	No	No	Yes	No
WBTC-USDC 0.3%	Yes	Yes	Yes	Yes	DAI-USDT 0.01%	No	Yes	Yes	Yes
WBTC-USDT 0.05%	Yes	Yes	Yes	Yes	USDe-USDT 0.01%	Yes	Yes	Yes	Yes
WBTC-USDT 0.3%	Yes	Yes	Yes	Yes	WETH-weETH 0.01%	No	No	Yes	No
WBTC-WETH 0.01%	Yes	Yes	Yes	Yes	wstETH-WETH 0.01%	Yes	Yes	Yes	Yes
WBTC-WETH 0.05%	Yes	Yes	Yes	Yes	WBTC-LBTC 0.05%	No	Yes	No	No

Table 4: Transition Probability Patterns - The table shows which pools clearly exhibit a given pattern. **Diag Eff.** is for the diagonal effect; **Burn Mint** is for the mint events elicitation by burn; **Indep. Swaps** represents the swaps' independence of liquidity provision events; **Swaps Block** represents swaps with or without a tick change that inhibit each other.

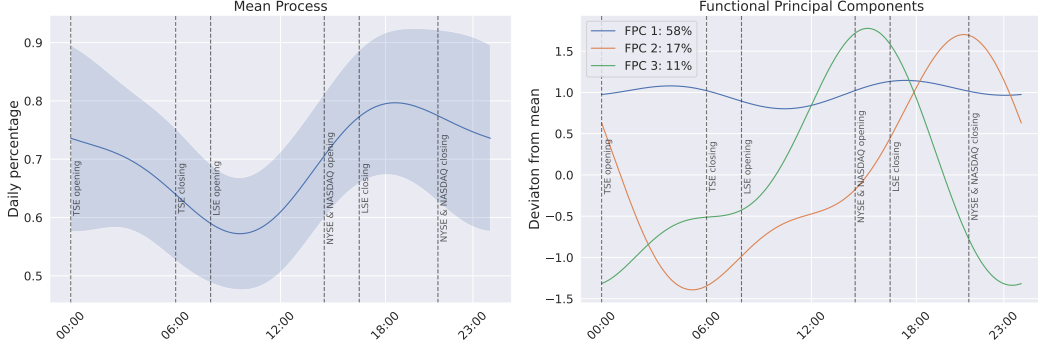


Figure 25: The left plot shows the mean process from the projection of the daily percentage market activity onto the first five Fourier bases. The shaded area corresponds to $\pm$ one standard deviation. On the right plot, there are the first 3 functional principal components with the corresponding explained variance ratio in the legend. They explain in total about 86% of the variation present in the data. The horizontal axis represents normalized daily time (time zone UTC). The vertical dashed lines correspond to the opening and closing of the main market sessions. TSE is for the Tokyo Stock Exchange; LSE is for the London Stock Exchange; NYSE is for the New York Stock Exchange.

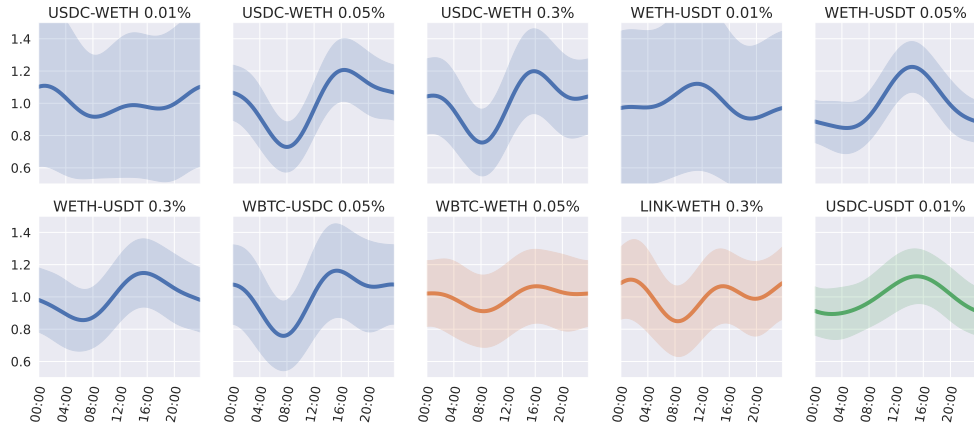


Figure 26: Market activity - comparison between pools (with 200,000 events, at least). The market activity is measured as the mean rescaled percentages of the number of events projected onto the Fourier space. Normal, Volatile, and Stable pools are highlighted in blue, orange, and green colours. The shadow area corresponds to $\pm \frac{1}{2}$ standard deviation.

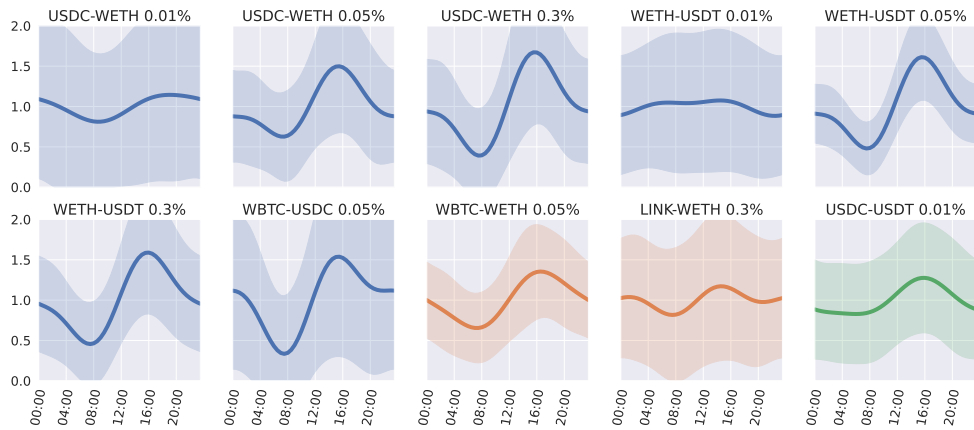


Figure 27: Daily market activity measured in terms of the MRV. Normal, Volatile, and Stable pools are highlighted in blue, orange, and green. The shadow area is for the mean $\pm \frac{1}{2}$ standard deviation.

C Swap Events - Quantitative Results

This section provides the quantitative results for the swap-related time series. Specifically, we first analyze the kurtosis of the log returns distribution. Next, we discuss the first lag in the returns ACF. Then, we move to the long-memory property exhibited by some time series based on swap events: absolute return, volume, and sign. Finally, we discuss the relationship between volume and variance.

Section 4.1 of the main text discusses the kurtosis and its scaling properties as the swaps aggregation size increases. When aggregating more swaps, we recover the power-law behavior typically observed in TradFi. Table 5 shows the quantitative results, pool-by-pool, of the power-law fit and the mesokurtosis test.

Pool	$\log(A)$	p	R2	$p\text{-val} > 0.01$	$p\text{-val} > 0.05$	Pool	$\log(A)$	p	R2	$p\text{-val} > 0.01$	$p\text{-val} > 0.05$
USDC-WETH 0.01%	<u>9.65</u>	<u>0.77</u>	<u>0.76</u>	<u>10.1</u>	<u>16.2</u>	WBTC-WETH 0.3%	3.26	0.26	0.66	1.80	2.83
USDC-WETH 0.05%	<u>5.31</u>	<u>0.38</u>	<u>0.84</u>	<u>2.14</u>	<u>2.70</u>	LINK-WETH 0.3%	2.80	0.20	0.68	1.56	1.68
USDC-WETH 0.3%	2.83	0.20	0.67	1.13	1.13	MNT-WETH 0.3%	3.74	0.32	0.71	2.81	3.55
WETH-USDT 0.01%	<u>10.80</u>	<u>0.81</u>	<u>0.89</u>	<u>21.47</u>	<u>25.55</u>	UNI-WETH 0.3%	3.93	0.32	0.64	3.03	3.17
WETH-USDT 0.05%	7.00	0.57	0.70	3.08	4.21	USDC-USDT 0.01%	11.12	0.85	0.93	16.12	20.80
WETH-USDT 0.3%	<u>3.62</u>	<u>0.27</u>	<u>0.81</u>	<u>1.86</u>	<u>3.80</u>	USDC-USDT 0.05%	7.80	0.83	0.61	10.98	17.08
WBTC-USDC 0.05%	8.76	0.95	0.70	2.10	2.90	DAI-USDC 0.01%	12.16	1.21	0.90	7.10	8.05
WBTC-USDC 0.3%	2.06	0.11	0.40	1.19	1.40	DAI-USDT 0.01%	<u>9.58</u>	<u>0.95</u>	<u>0.80</u>	<u>8.18</u>	<u>15.32</u>
WBTC-USDT 0.05%	7.43	0.67	0.68	26.02	28.21	USDe-USDT 0.01%	<u>9.77</u>	<u>1.01</u>	<u>0.79</u>	<u>4.93</u>	<u>4.93</u>
WBTC-USDT 0.3%	2.44	0.18	0.74	1.76	2.07	WETH-weETH 0.01%	9.39	0.83	0.92	0.01	0.01
WBTC-WETH 0.01%	7.29	0.54	0.64	0.01	0.01	wstETH-WETH 0.01%	<u>9.28</u>	<u>0.72</u>	<u>0.86</u>	<u>0.00</u>	<u>0.00</u>
WBTC-WETH 0.05%	4.28	0.32	0.60	2.19	3.52	WBTC-LBTC 0.05%	9.84	0.90	0.96	0.00	0.00

Table 5: Kurtosis as a function of the number of aggregated swaps. $\log(A)$ and p are the coefficients fitted from the regression $K(n) \sim A \cdot n^{-p}$. **R2** is for the R2 score of the linear regression $\log(K(n)) \sim \log(A) - p \log(n)$. **p-val** > 0.01 and **p-val** > 0.05 equal the maximum aggregation size n such that the test in [6] has p-value greater than 0.01 or 0.05, normalized by the mean number of swaps in a day. Pools where the R2 score of the power-law regression is greater than 0.9 (0.75) are in bold (underlined).

As outlined in Section 4.2 of the main text, the returns ACF of several pools shows a huge negative peak at the first lag. This is robust to sandwich attacks. Our guess is that it is related to the action of reverse arbitrageurs. To test for this, we disentangle the first lag ACF by conditioning on $abs(r_{t-1}) > \phi$ and $abs(r_{t-1}) \leq \phi$, where r_{t-1} is the return induced by the transaction in swap-time $t - 1$, $abs(\cdot)$ is the absolute value, and ϕ is the fee tier. The results of the decomposition are in Table 6.

Pool	ACF (1)	ACF (1 g)	ACF (1 l)	Pool	ACF (1)	ACF (1 g)	ACF (1 l)
USDC-WETH 0.01%	-0.014	-0.015	0.000	WBTC-WETH 0.3%	0.056	-0.142	0.256
USDC-WETH 0.05%	-0.042	-0.071	0.038	LINK-WETH 0.3%	-0.158	-0.414	0.211
USDC-WETH 0.3%	0.029	-0.210	0.296	MNT-WETH 0.3%	-0.000	-0.122	0.166
WETH-USDT 0.01%	-0.026	-0.032	-0.002	UNI-WETH 0.3%	-0.077	-0.311	0.317
WETH-USDT 0.05%	-0.035	-0.046	0.010	USDC-USDT 0.01%	-0.096	-0.123	-0.001
WETH-USDT 0.3%	0.016	-0.194	0.259	USDC-USDT 0.05%	-0.494	-0.536	0.007
WBTC-USDC 0.05%	-0.133	-0.139	0.003	DAI-USDC 0.01%	-0.497	-0.497	-0.021
WBTC-USDC 0.3%	0.003	-0.231	0.385	DAI-USDT 0.01%	-0.003	-0.004	-0.005
WBTC-USDT 0.05%	-0.218	<u>-0.226</u>	0.022	USDe-USDT 0.01%	-0.139	-0.174	-0.008
WBTC-USDT 0.3%	0.020	-0.128	0.301	WETH-weETH 0.01%	-0.496	-0.682	0.027
WBTC-WETH 0.01%	-0.270	-0.271	0.004	wstETH-WETH 0.01%	-0.310	-0.359	0.008
WBTC-WETH 0.05%	-0.089	-0.124	0.074	WBTC-LBTC 0.05%	-0.077	-0.085	0.011

Table 6: ACF lag 1 decomposition - The ACF at lag 1 **ACF(1)** is decomposed by conditioning on the previous return’s magnitude bigger or smaller than the pool fee tier, respectively, **ACF(1 | g)** and **ACF(1 | l)**. The conditional ACFs outside the 99% (95%) confidence interval of the unconditional ACF are in bold (underlined).

Next, we analyze the quantitative results for long-memory. Specifically, Table 7 examines long-memory in time series based on swap events by showing the power-law fit to the ACF of absolute return, volume, and sign.

First, we observe that the pool fee tier could change the statistical properties of the pool. That is, fixed one pair, if there is ACF persistence in the 0.05% pool, it is not certain to recover the same pattern in the 0.01% or 0.3% pools. As for Normal pairs, we notice that: pools involving USDC or USDT similarly behave (that is, there is long-memory for the same ACFs and fee tiers); and the exponent p typically increases with the pool fee tier. Regarding short-memory in the volume time series ACF, we notice just a few occurrences, which are

Pool	Abs			Vol			Sign		
	log(A)	p	R2	log(A)	p	R2	log(A)	p	R2
USDC-WETH 0.01%	-6.38	0.09	0.03	-3.18	0.05	0.22	-2.86	0.59	0.27
USDC-WETH 0.05%	-1.42	0.20	0.88	-1.78	0.32	0.95	-3.35	0.22	0.91
USDC-WETH 0.3%	0.16	0.51	0.94	0.36	0.51	0.82	-5.32	0.06	0.00
WETH-USDT 0.01%	-5.12	0.12	0.15	-3.64	0.11	0.52	-2.31	0.68	0.43
WETH-USDT 0.05%	-2.85	0.21	0.90	-2.15	0.17	0.94	-3.62	0.26	0.78
WETH-USDT 0.3%	-0.07	0.41	0.92	-1.50	0.21	0.95	-3.93	0.09	0.06
WBTC-USDC 0.05%	-5.12	0.35	0.11	-1.88	0.17	0.79	-5.25	0.19	0.03
WBTC-USDC 0.3%	-0.19	0.33	0.91	-1.34	0.19	0.87	-5.77	-0.03	0.00
WBTC-USDT 0.05%	-1.92	0.34	0.31	-1.17	0.16	0.75	-3.66	0.30	0.09
WBTC-USDT 0.3%	-0.14	0.46	0.90	-1.39	0.22	0.86	-4.37	0.18	0.02
WBTC-WETH 0.01%	-1.90	0.47	0.39	-1.41	0.39	0.74	-1.30	0.25	0.87
WBTC-WETH 0.05%	-2.27	0.25	0.82	-2.48	0.13	0.66	-2.04	0.15	0.94
WBTC-WETH 0.3%	-0.72	0.36	0.91	-2.26	0.31	0.52	0.34	0.69	0.42
LINK-WETH 0.3%	1.37	0.81	0.78	-2.39	0.12	0.64	-2.73	0.50	0.22
MNT-WETH 0.3%	-0.13	0.63	0.45	0.01	0.66	0.40	-2.53	0.13	0.39
UNI-WETH 0.3%	0.02	0.47	0.77	-1.94	0.16	0.83	-5.28	0.08	0.01
USDC-USDT 0.01%	-4.82	0.28	0.48	-2.96	0.29	0.56	-2.39	0.41	0.75
USDC-USDT 0.05%	-7.40	0.10	0.00	-2.06	0.10	0.56	-1.62	0.13	0.70
DAI-USDC 0.01%	0.65	1.37	0.55	-3.94	0.21	0.11	-1.37	0.47	0.73
DAI-USDT 0.01%	-3.09	0.85	0.24	-1.43	0.49	0.37	-2.90	0.46	0.20
USDe-USDT 0.01%	-3.57	0.87	0.12	-1.90	0.10	0.81	-1.66	0.04	0.81
WETH-weETH 0.01%	0.05	0.98	0.19	-0.36	0.78	0.37	0.37	0.63	0.66
wstETH-WETH 0.01%	-9.84	-0.20	0.02	-3.20	0.20	0.27	-0.44	0.58	0.77
WBTC-LBTC 0.05%	9.60	3.69	0.63	-3.43	0.33	0.09	-0.00	0.34	0.75

Table 7: Power-law fit results from the regression $ACF(L) \sim A \cdot L^{-p}$. **R2** is for the R2 score of the linear regression $\log(ACF(L)) \sim \log(A) - p \log(L)$. The left, central, and right columns are for the returns' absolute value, traded volume, and swap signs, respectively. Bold cells indicate long-memory.

overlapping with deviations in the transition probabilities patterns and with the least active pools -to have a look, compare Table 4 and central columns in Table 7. As for long-memory in the swap signs series, we find this pattern to be more frequently shown in Synthetic and Stable pools. These also show significantly bigger exponent p and coefficient $\log(A)$.

Finally, we focus on the relationship between volume and Realized Variance (RV), which is used as a proxy for the integrated variance. In Section 4.4 of the main text, we show that several pools exhibit a significant linear relationship between the logarithms of volume and RV. We fit the linear regression $\log(Vol) = A + q \log(RV)$ to verify this. Specifically, the Volume is intended to be the aggregated volume, and both Vol and RV are computed over a time window of n minutes, with $n = 1, 5, 10, 30$. Table 8 shows the fitted coefficient q and the $R2$ score. The results show $q \approx 0.5$. There is not a monotonic relationship of q from n , as it seems to be strongly pool-dependent.

Pool	1 min		5 min		10 min		30 min	
	q	R2	q	R2	q	R2	q	R2
USDC-WETH 0.01%	0.46	0.38	0.54	0.33	0.67	0.37	0.95	0.51
USDC-WETH 0.05%	0.47	0.91	<u>0.44</u>	<u>0.80</u>	0.43	0.74	0.42	0.68
USDC-WETH 0.3%	0.49	0.92	0.50	0.93	0.51	0.93	0.53	0.93
WETH-USDT 0.01%	<u>0.49</u>	<u>0.76</u>	0.45	0.51	0.42	0.37	0.49	0.25
WETH-USDT 0.05%	<u>0.46</u>	<u>0.84</u>	0.44	0.64	0.41	0.53	0.37	0.43
WETH-USDT 0.3%	0.49	0.95	0.50	0.95	0.51	0.95	0.52	0.93
WBTC-USDC 0.05%	0.40	0.46	0.44	0.49	0.46	0.50	0.49	0.48
WBTC-USDC 0.3%	0.38	0.54	0.41	0.57	0.43	0.59	0.46	0.61
WBTC-USDT 0.05%	0.27	0.14	0.28	0.12	0.30	0.11	0.36	0.11
WBTC-USDT 0.3%	0.44	0.70	0.46	0.72	0.47	0.73	0.5	0.73
WBTC-WETH 0.01%	0.35	0.40	0.31	0.16	0.32	0.12	0.37	0.11
WBTC-WETH 0.05%	<u>0.46</u>	<u>0.88</u>	<u>0.46</u>	<u>0.85</u>	<u>0.46</u>	<u>0.82</u>	0.44	0.68
WBTC-WETH 0.3%	<u>0.45</u>	<u>0.88</u>	0.46	0.90	0.47	0.90	<u>0.48</u>	<u>0.89</u>
LINK-WETH 0.3%	<u>0.46</u>	<u>0.85</u>	<u>0.48</u>	<u>0.86</u>	<u>0.49</u>	<u>0.86</u>	<u>0.50</u>	<u>0.84</u>
MNT-WETH 0.3%	0.50	1.00	0.51	1.00	0.51	0.99	0.52	0.99
UNI-WETH 0.3%	0.48	0.93	0.50	0.94	0.50	0.93	0.52	0.92
USDC-USDT 0.01%	0.36	0.74	0.35	0.71	0.33	0.66	0.25	0.52
USDC-USDT 0.05%	<u>0.42</u>	<u>0.86</u>	<u>0.43</u>	<u>0.85</u>	<u>0.44</u>	<u>0.84</u>	<u>0.45</u>	<u>0.84</u>
DAI-USDC 0.01%	0.47	0.93	0.48	0.95	0.49	0.95	0.50	0.94
DAI-USDT 0.01%	<u>0.43</u>	<u>0.85</u>	<u>0.43</u>	<u>0.84</u>	<u>0.43</u>	<u>0.83</u>	<u>0.42</u>	<u>0.79</u>
USDe-USDT 0.01%	0.40	0.73	<u>0.40</u>	<u>0.76</u>	<u>0.40</u>	<u>0.75</u>	0.37	0.66
WETH-weETH 0.01%	<u>0.41</u>	<u>0.79</u>	<u>0.41</u>	<u>0.76</u>	0.40	0.72	0.36	0.57
wstETH-WETH 0.01%	<u>0.40</u>	<u>0.80</u>	<u>0.41</u>	<u>0.81</u>	<u>0.40</u>	<u>0.80</u>	<u>0.37</u>	<u>0.75</u>
WBTC-LBTC 0.05%	<u>0.44</u>	<u>0.84</u>	<u>0.42</u>	<u>0.78</u>	0.40	0.73	0.37	0.61

Table 8: Volume vs Realized Variance - fitted coefficient q and $R2$ score of the linear regression $\log(Vol) = \log(A) + q \log(RV)$. The fits with $R2 \geq 0.90$ are highlighted in bold; fits with $R2 \geq 0.75$ are underlined.