Liquifi Protocol

Igor Struchkov
igors@liquifi.org

Alexey Zubritskiy
azubr@liquifi.org

Igor Mikhalev
igorm@liquifi.org

Bihao Song
bihao@liquifi.org

October 2020

Abstract

This whitepaper contains a technical description of Liquifi protocol principles, architecture, smart contracts, governance and evaluation. Liquifi is a protocol for a decentralized digital asset exchange (DEX). It has all functions of an ordinary liquidity pool DEX with some traits of order book DEX that allow traders to get better exchange rates and give arbitrageurs new profitable opportunities. The whitepaper contains a proof of the main protocol feature - time-locked swap operations that allow counter-trades to be performed in parallel, compensating price movements and thus decreasing price slippage. Liquifi protocol also introduces a possibility of zero-fee trades under certain conditions. Liquifi follows decentralized governance and fair launch principles from the very beginning. We present Liquifi governance smart contracts that allow community of liquidity providers to take control over the protocol parameters in a decentralized way, without any party having exclusive rights. Besides the theoretical proof, we present simulation results that show protocol performance in different market conditions.

1 Introduction

DeFi, short for "decentralized finance", is an umbrella term for a variety of financial applications in cryptocurrency or blockchain geared towards removing existing intermediaries, banks and others, from the value chain. It gives

users an alternative by removing the need to trust any central party at all, which results in major cost savings and potentially service performance improvement for FI (financial institution) users. The approach of using smart contract to provide financial services, e.g. exchange, lending platoform and later the services based on new concepts, such as yield farming, liquidity mining, composability and money legos, give the new and the narrowed definition of DeFi that we are refering to these days.
Decentralized digital asset exchanges (DEXes) are important players on DeFi market, allowing to trade digital assets (usually, ERC20 tokens) automatically using smart contracts without any third party interaction. There are two main types of DEXes being used, the liquidity-pool based, with the examples like Uniswap, Balancer, and the order-book based, like Loopring and DiversiFi. We listed the different characteristics of these DEX below.
Liquidity-pool based DEX:

- Liquidity pool providers invest digital assets
- Trades are performed against the pool, a counterparty is not required
- Current exchange price is determined automatically depending on digital assets ratio in the pool (automatic market maker, AMM)
- Every trade changes digital assets ratio and therefore the price
- When the pool price differs from the market price, arbitrageurs make profitable trades against the pool and push the pool price back to the market level

Orderbook based DEX:

- Trades are performed between two counterparties
- Parties put orders which are recorded in the order book at the corresponding price levels
- The first party (Maker) sets an initial order, others (Takers) close deals
- Current exchange price is identified by the top of the order book (highest bid and lowest ask)

	Liquidity pool DEX	Order book DEX
Advantages	No need for market makers – trades are completely automatic	Order flow familiar for traders
	Possible to implement completely on-chain	Less price slippage
	Easy to invest	Look more professional
Disadvantages	Price slippage. For big deals the impact on price is stronger, making the exchange less attractive.	Need at least 2 participants (one is a market maker)
	On-chain implementation limits throughput	Order fulfillment needs off-chain processing

Table 1: comparison of liquidity pool DEX and order book DEX

By comparing the two types of DEX as shown in table 1, we can see that liquidity pool DEX is easier to be set up and used by investors at different scale. But it has price slippage problem, bigger the deal is, more slippage of the price it will cause.
In order to resolve the problem faced when using common liquidity pool DEX protocols, we introduced Liquifi, another DeFi protocol for a decentralized digital asset exchange. It is based on liquidity pool principle which allows Liquifi to maintain pools for various digital asset pairs that are used to perform swap operations automatically using Liquifi automatic market maker (AMM) model. Anybody can invest liquidity (digital assets) into the pools and earn pool fees from swap operations.
Comparing with other common DEX protocols, Liquifi brings the following changes:

- First, Liquifi has an advanced automatic market maker (AMM) model that is built on the classic constant product model with a new smooth liquidity flow feature. This feature allows to get better exchange rates than with the classic model, especially for large deals. This is achieved by extending swap operations over a given time period. Thus, arbitrage trades can occur in parallel with the large deal, providing liquidity to make the final price better.

- Second, Liquifi is not vulnerable to frontrunning and transaction reordering problems because arbitrageurs do not compete to place their transactions in front of the others. Instead, arbitrageurs can implement more complex strategies when choosing a proper time instant to perform their trades.

Generally speaking, Liquifi has all functions of an ordinary liquidity pool DEX with some traits of order book DEX that allow traders to get better exchange rates and give arbitrageurs new profitable opportunities.
Liquifi is set up and maintained by an open community of developers, investors, liquidity providers interested in proper operation and evolution of the protocol. Together they form a decentralized autonomous organization (DAO) backed by Liquifi governance smart contracts. DAO is the only entity that can influence the protocol parameters. Based on this well established DAO, Liquifi follows decentralized governance and fair launch principles from the very beginning.

2 Key Concepts

Liquifi automatic market maker (AMM) model is based on constant product model, i.e. $x \cdot y = k$ invariant is held for a liquidity pool token pair (x, y). On top of the standard model Liquifi introduces a new exchange algorithm, which is especially benefitial for big deals and/or small liquidity pools. The main idea behind Liquifi model is that traders are allowed to set a timeout for their swap operation and wait to get better price for their deal - so called time-locked operation. This is implemented by utilizing some elements of order book exchange: namely, a time-locked operation is placed as an order and can be (at least partially) compensated by symmentric deals from other traders.

2.1 Proof

Assume Trader1 wants to sell x_1 of tokens A for tokens B. Let (x_0, y_0) be the current quantity of tokens A and tokens B respectively in the liquidity pool. Let us also assume for simplicity that pool fee is 0. Using the standard constant product formula, Trader1 then will get y_1 of tokens B:

$$y_1 = y_0 - \frac{x_0 y_0}{x_1 + x_0}$$

So the constant product price for tokens A is

$$P_A = \frac{y_1}{x_1}$$

Now, instead of performing immediate transaction at price P_A, Trader1 specifies deal timeout T and enters pool wait list. One can imagine that Trader1's deal is not done at once, but is spread over time with uniform liquidity flow over this period. The number of tokens A swapped at time t, $0 \leq t \leq T$ is:

$$x_A(t) = x_1 \cdot \frac{t}{T}$$

Without any interfering transactions, the process will result to the same swap price P_A.
Assume that at time moment $0 \leq t_2 \leq T$ Trader2 wants to sell y_2 of tokens B for tokens A. The pool balance at this moment will be:

$$x(t_2) = x_0 + x_A(t_2), y(t_2) = \frac{x_0 y_0}{x(t_2)}$$

So, Trader2 will get x_2 of tokens A:

$$x_2 = x(t_2) - \frac{x_0 y_0}{y_2 + y(t_2)}$$

The exchange price for him will be:

$$P_B = \frac{x_2}{y_2}$$

If nothing else happens until time T, the final pool balance at time T will be:

$$x(T) = x_0 + x_1 - x_2, y(T) = \frac{x_0 y_0}{x(T)} = \frac{x_0 y_0}{x_0 + x_1 - x_2}$$

So, now Trader1 will get y_1' of tokens B:

$$y_1' = y_0 + y_2 - \frac{x_0 y_0}{x_0 + x_1 - x_2}$$

Taking into account that tokens A price would be the lowest after full completion of Trader1's deal, we get:

$$x_2 < x(T) - \frac{x_0 y_0}{y_2 + y(T)} = x_0 + x_1 - \frac{x_0 y_0}{y_2 + y_0 - y_1}$$

when $t_2 < T$.
Then we get:

$$x_0 + x_1 - x_2 > \frac{x_0 y_0}{y_2 + y_0 - y_1}$$

$$\frac{x_0 y_0}{x_0 + x_1 - x_2} < y_2 + y_0 - y_1$$

$$y_1' > y_1$$

Thus, Trader1 will get more for his tokens A than in constant product AMM and

$$P_A' = \frac{y_1'}{x_1} > P_A$$

3 Functional description

3.0.1 Liquifi operation types

Liquifi protocol is an extension to the standard constant product exchange and allows all types of basic operations. Liquifi also adds new featured operation types. Overall list of operations is following:

- Standard immediate swap operation;
- Immediate swap operation with zero fee;
- Flash swap operation;
- Time-locked operation.

3.1 Standard immediate swap operations

Standard swap operations are performed in one transaction. Let a pool contain x_0 amount of tokens A and y_0 amount of tokens B. Assume operation to swap N_A tokens A for tokens B. Then the operation output N_B is calculated using constant product formula:

$$N_B = y_0 - \frac{x_0 y_0}{x_0 + \gamma N_A}$$

where $\gamma = 1 - \alpha$, α is pool fee.
The fee amount αN_A is added to the pool at the operation completion.

3.2 Time-locked operations

3.2.1 Swap order wait lists

Let a swap order to sell TokenA for TokenB be denoted by a tuple $O_{Ai} = \{x_i, T_i, P_{si}\}$ where x_i is the number of TokenA to sell, T_i is timeout, P_{si} is stop loss swap price.
Let a swap order to sell TokenB for TokenA be denoted by a tuple $O_{Bj} = \{y_j, T_j, P_{sj}\}$ where y_j is the number of TokenB to sell, T_j is timeout, P_{sj} is stop loss swap price.
Then each pool maintains a swap order wait list for O_{Ai} and O_{Bj} orders.

3.2.2 Liquidity flow

Every order in a wait list has a corresponding token count:

- For TokenA/TokenB orders S_{Ai} is the currently spent TokenA amount;
- For TokenB/TokenA orders S_{Bj} is the currently spent TokenB amount.

Let t be time elapsed since an order entered the wait list. Then

$$S_{Ai}(t) = x_i \cdot \frac{t}{T_i}, S_{Bj}(t) = y_j \cdot \frac{t}{T_j}$$

We can use the first derivative to calculate the speed of liquidity flow from each order:

$$S'_{Ai} = \frac{dS_{Ai}}{dt} = \frac{x_i}{T_i}, S'_{Bj} = \frac{dS_{Bj}}{dt} = \frac{y_j}{T_j}$$

And the total liquidity flow from all currently active orders (let n be the number of active TokenA/TokenB orders and m be the number of active TokenB/TokenA orders) is:

$$S'_A = \frac{dS_A}{dt} = \sum_{1 \le i \le n} \frac{x_i}{T_i}, S'_B = \frac{dS_B}{dt} = \sum_{1 \le j \le m} \frac{y_j}{T_j}$$

Liquidity flow is a piecewise linear function with break points at time moments when the wait list changes. This can be:

- Entering a new order to the wait list
- Liquidity deposit/withdraw operation
- All kinds of immediate swaps
- Closing an order that has timeout expired
- Closing an order for which stop loss price limit is reached (P_{si})
- Closing an order by user request
- Closing an order by governance

3.2.3 Swap price calculation

Let α be the pool fee and $\gamma = (1 - \alpha)$.
Using the constant product market maker model we get:

$$(\gamma S'_A t + x_0 - \gamma S'_B t P)(\gamma S'_B t + y_0 - \gamma S'_A \frac{t}{P}) = x_0 y_0$$

Here x_0 and y_0 are TokenA and TokenB amounts respectively at the beginning of the current linear flow section, t is time from the beginning of the current linear flow section, P is a calculated fair price for partial orders in both directions.
If $S'_A = 0$ or $S'_B = 0$ then the price P (or $\frac{1}{P}$) becomes a usual constant product swap price.
In general case we have to solve the following quadratic equation to find the price P at time t:

$$(\gamma S'^2_B t^2 + S'_B y_0 t)P^2 - (2\gamma S'_A S'_B t^2 + S'_B x_0 t + S'_A y_0 t)P + \gamma S'^2_A t^2 + S'_A x_0 t = 0$$

This equation appears to have two positive roots when $\frac{S'_A}{S'_B}$ ratio is different from the token ratio in the pool. One root equals to $\frac{S'_A}{S'_B}$ and the second root is the target price P. For a special case when $\frac{S'_A}{S'_B}$ ratio equals to the token ratio in the pool, the equation has just one root.
Using Vieta's formula we can get the final expression for the target price P:

$$P = \frac{\gamma S'_A t + x_0}{\gamma S'_B t + y_0}$$

We can also derive a formula to get time t when a given price P is reached:

$$t(P) = \frac{x_0 - P y_0}{\gamma (P S'_B - S'_A)}$$

From this a moment when stop loss price limit $P_{si(j)}$ is reached for some order i(j) is

$$t_s = t(\frac{1}{\gamma} \cdot P_{si(j)})$$

taking into account the fee factor.

3.2.4 Pool pressure

Having calculated P(t), we can predict swap price at some future moment t provided that no other orders come or complete. If we take a standard time interval T (e.g. 1 minute) for t, and the current pool swap price P_0, we can calculate pool pressure R as

$$R = \frac{P(T) - P_0}{P_0}$$

3.2.5 Smart contract operation

For on-chain swap operations the following implementation is proposed.

1. Liquifi pool smart contract keeps track of the closest linear liquidity flow function break point t^*. It could be either the closest order timeout expiration or calculted time for the closest stop loss limit event.

2. When a wait list recalculation operation is triggered, e.g. when a new order or transaction comes, the smart contract checks whether t^* has been reached. If true, a corresponding order is completed and new values for S'_A, S'_B, x_0, y_0 are calculated as well as a new break point t^*. If it is also in the past, then the recalculation is repeated until t^* is in the future.

3. Then, if there is an outstanding immediate swap operation, it is performed, and again S'_A, S'_B, x_0, y_0, t^* are recalculated.

To make the break points recalculation more efficient, wait lists may be indexed in two ways: sorted by order timeout expiration and sorted by stop loss events time (separately for O_{Ai} and O_{Bj} orders). When a new order is added, it must be placed in these indices accordingly (search for a proper insertion place may be performed off-chain and then verified by the smart contract).

3.2.6 Handling fees

It is known that when fees are applied making several small swap operations gives less output than one operation with equal amount. Therefore, to make time-locked operations at least as profitable as immediate swaps, Liquifi accumulates fees for a time-locked operation and adds them to the pool only after the operation completes.

3.3 Immediate swap operations with zero fee

To facilitate arbitrage counter-deals when time-locked operations are in progress, Liquifi introduces zero-fee incentive for arbitrageurs that perform counter-deals in parallel with time-locked operations. Maximum amount of such zero-fee arbitrage deals is limited by the following algorithm.
When a new order arrives, the contract starts to add input token amount of the order with a corresponding liquidity flow speed to a corresponding variable: I_A for A/B orders and I_B for B/A orders.

$$I_A(t) = I_A(t_0) + \sum_{1 \le i \le n} S'_{Ai}(t - t_0), I_B(t) = I_B(t_0) + \sum_{1 \le j \le m} S'_{Bj}(t - t_0)$$

where t_0 is the last moment of variable recalculation.

When any order is closed, a corresponding input token amount is subtracted or, if a corresponding I variable is less than this amount, it is assigned 0, enforcing that always $I_A(t) \geq 0$ and $I_B(t) \geq 0$:

$$I_A(t) = max(I_A(t) - S'_{Ai}(t - t_{0i}), 0)$$

or

$$I_B(t) = max(I_B(t) - S'_{Bj}(t - t_{0j}), 0)$$

where t_{0i} and t_{0j} are starting times of corresponding time-locked operations. When an arbitrage deal request, pretending on zero fee, arrives, it's amount (D) is checked against the I variables difference, i.e. if the arbitrage deal is B/A, then it is checked as follows:

$$D_B \leq I_A(t) \cdot \frac{B}{A} - I_B(t)$$

where A and B are total amounts of tokens in the pool.
If the condition above holds, the deal is allowed zero fee. Anyway, the amount of arbitrage deal is then subtracted from an opposite I variable or, if a corresponding I variable is less than this amount, it is assigned 0:

$$I_A(t) = max(I_A(t) - D_B \cdot \frac{A}{B}, 0)$$

3.4 Flash swap operations

Flash swaps allow to perform token exchange even without holding tokens for input, use the output and then return the input tokens to the pool. All the above operations may be done in one atomic transaction. Inside the operation Liquifi contract make a call to a user-specified callback contract between output tokens transfer and checking the constant product invariant. If after the call the invariant is not satisfied, the whole transaction is reverted.

3.5 Protocol fee

Liquifi protocol allows protocol fee to be enabled by governance (by default protocol fee is disabled). Protocol fee is deducted from pool fees. Protocol fee level can be adjusted by governance, but is limited to at most 25% of pool fee. Protocol fee destination address is also specified by governance.

Protocol fee collection and sending. When swap operations are performed, protocol fee is kept in the pool together with liquidity provider fee. For sake of gas economy, protocol fee is transferred to the destination address only when operations with liquidity (deposit or withdraw) take place. During these operations extra LPTX tokens are minted to fit protocol fee amount and then transferred to the destination address.

Protocol fee calculation. According to [1] the total collected fees can be calculated by the growth of factor $\sqrt{k} = \sqrt{x \cdot y}$. Accumulated fees between time moments t_1 and t_2 as a share of the pool are given by:

$$f_{1,2} = 1 - \frac{\sqrt{k_1}}{\sqrt{k_2}}$$

Protocol fee share is then $\phi \cdot f_{1,2}$ where ϕ is protocol fee level defined by governance.
Then, the number of LPTX tokens to mint for protocol fee s_m is defined by equation:

$$\frac{s_m}{s_m + s_1} = \phi \cdot f_{1,2}$$

where s_1 is total number of LPTX tokens at time t_1.
Solving the equation for s_m we get:

$$s_m = \frac{\sqrt{k_2} - \sqrt{k_1}}{(\frac{1}{\phi} - 1) \cdot \sqrt{k_2} + \sqrt{k_1}} \cdot s_1$$

4 Governance

Liquifi is governed by the community from the date of launch. Liquifi smart contracts do not contain any code that could give anybody exclusive rights to change or update the contracts or their parameters. Possible changes are limited to be initiated by Liquifi governance smart contract only. These changes may include adjusting pool parameters (e.g. liquidity provider fee), upgrading to a newer version of the pool smart contract or replacing the governance contract itself. But there is no way to modify LQF tokens minting schedule or rules, so LQF token distribution is completely predictable.

4.1 Principles

- Governance is done by a community of LQF token holders
- LQF tokens will be minted with respect to participation in liquidity pools and given to liquidity providers

Total supply of LQF tokens will be about 107M (see Implementation section for details).

4.2 Tokens distribution

A predefined number of governance tokens will be minted every week. This number will decrease every week with a predefined speed (decay factor). The minted tokens will be automatically distributed among liquidity Providers proportionally to their share in the total liquidity amount.

4.3 Governable protocol parameters

The following parameters may be modified using Liquifi decentralized governance:

- Pool parameters
  - Immediate and time-locked swap fee level for each pool (default - 0.3%)
  - Protocol fee level for each pool (disabled by default)
  - Maximun timeout for time-locked operations (default - 1 hour)
  - Maximum length of order wait lists (default - 100)
  - Pool locked flag (default - unlocked)
  - Zero fee on/off (default - on)
- Parameters for all pools
  - Protocol fee receiver (default - empty)
  - Governance contract address (used to upgrade to newer versions of governance)
- Governance parameters

  - Liquifi pool factory (used to upgrade to newer versions of the protocol)
  - On-chain voting parameters (voting period, quorum, approval and veto thresholds, deposit amount to create a proposal)

4.4 Communications

Event	**Schedule**	**Participants**
Core team sync call	Once in 2 weeks (online)	Core team
Liquifi status call	Monthly (online)	All community members

5 Implementation

Liquifi is a decentralized exchange based on liquidity pool. Core exchange logic in implemented in smart contracts (which are open-source and externally audited) and processing is performed completely on-chain. This approach guarantees that all liquidity invested into Liquifi pools is kept safely and is not vulnerable to theft or other attacks. This also ensures fair and predictable exchange rates for digital asset traders and arbitrageurs.
Besides the smart contracts Liquifi also includes back-end and front-end components. These components are not critical to exchange operation and may be used or bypassed at wish. The main reasons of including these components, other than user convenience, are:

- Gas economy. Part of operations needed to perform time-locked swap operations, like scanning the history of orders in wait lists, is performed off-chain and provided to the smart contract when needed. Using a chain of hashes that are checked by the smart contract prevents malicious users from providing forged data input.
- Model-based price prediction. Built-in simulation model provides calculated predictions of expected exchange rate for time-locked operations. This model takes current wait list state and market situation as inputs and includes expected arbitrageurs behavior. This allows Liquifi to assist traders in selecting the most profitable operation parameters.

5.1 Architecture

Liquifi consists of the following components:

- Main Ethereum smart contracts
  - Liquidity pool smart contract
  - Factory smart contract
  - Register smart contract
- Governance Ethereum smart contracts
  - ERC20 governance token (LQF) / Minter smart contract
  - Activity meter smart contract
  - Governance smart contract
  - Proposal (voting) smart contract
  - Governance router smart contract
- Off-chain service
  - Support for reactive front-end operations
  - Instant pool price calculation based on the wait list state
  - Time-locked deal price prediction based on built-in simulation model
- Front-end
  - Exchange UI
  - Liquidity provider UI
  - Governance UI
  - Statistics dashboard

5.2 Implemented Logic of Key Components

5.2.1 Liquifi Pool Register

Liquifi pool register smart contract is a preferred entry point for any operation on the pools. Pool register relies on the pool factory smart contract to keep track of all existing liquidity pools and to create new pools. Once a pool is found or created, pool register delegates core operations to a liquidity pool contract. Another function of pool register contract is wrapping ETH to WETH when interacting with pools with WETH as one of tokens. This feature allows users to directly use these pools with their ETH balances without manual wrapping.

5.2.2 Liquidity Pool

Token pair registration Everybody can register a new pool for any token pair if such a pool does not exist. Therefore, for each token pair only one pool can be registered. When a user starts add liquidity transaction on a non-existing pool, pool register delegates to pool factory actual creation of a new liquidity pool smart contract.

Liquifi pool tokens Each liquidity pool smart contract implements ERC20 interface. This allows to mint special Liquifi pool tokens when liquidity providers add liquidity to a pool. These tokens can be used as any normal ERC20 token, including transferring them to other users. When a liquidity provider wishes to withdraw his liquidity from a pool, he is required to provide his Liquifi pool tokens from this pool, which are burnt during this transction.
Liquidity providers can also use their Liquifi pool tokens to generate LQF governance tokens as described below.
Liquifi pool tokens are named LPTX, where X is replaced by a sequential number of a pool.

Liquifi pool exchange fee A default exchange fee value is assigned to a new liquidity pool at creation. This fee value can be later changed by Liquifi community via the governance process.

Adding liquidity to a pool Anybody can add liquidity to any pool and become a liquidity provider. Liquidity is added by purchasing one or more

associated ERC20 pool tokens that are minted during this transaction. The number of pool tokens minted is calculated as follows:

- For the first deposit in a pool (let $x_{deposited}$ and $y_{deposited}$ be the deposited numbers of respective tokens in a pair):

$$s_{minted} = \sqrt{x_{deposited} \cdot y_{deposited}}$$

  To prevent the cost of minimal pool share (in theory) to grow extremely high, Liquifi sends the first 10^{-15} pool shares to address 0 instead of the liquidity provider. This cost is negligible for the provider but makes it very hard to perform pool share growing attack.

- For next deposits

$$s_{minted} = min(\frac{x_{deposited}}{x_{starting}} \cdot s_{starting}, \frac{y_{deposited}}{y_{starting}} \cdot s_{starting})$$

The price of a Liquifi pool token consists of 2 components - for each exchangeable token in the pair - and is calculated by dividing the total pool amount (for each token in the pair) by the number of all existing pool tokens.

Withdrawing liquidity from a pool Anybody holding Liquifi pool tokens can return them to the pool at any time. Returned tokens are burnt and their price (amount of 2 tokens in the pair) is returned to the pool token holder.

Pools that have ETH in a pair. For pairs with ETH Liquifi automatically wraps/unwraps ETH operations with WETH tokens. So internally these pools always contain WETH and not ETH, but this is transparent to users. These pools also allow operations with WETH tokens directly.

Immediate swap operations Anybody can trade against token pairs in any pool (become a Trader). To perform a swap operation a Trader sends some tokens of type A to a liquidity pool smart contract and specifies the worst case price that is acceptable (slippage tolerance). Deal price is calculated automatically (see Automatic Market Maker Model). If the calculated price is acceptable for the Trader, the supplied tokens of type A are added to

the pool and a corresponding number of tokens of type B are removed from the pool and sent to the Trader in one transaction. If the calculated price is unacceptable, the supplied tokens of type A are sent back to the Trader.

Time-locked swap operations A Trader can provide an optional timeout parameter for his deal and thus start a time-locked swap operation. A liquidity pool smart contract maintains a wait-list that contains orders with timeouts, sorted using 2 indices: by timeout expiration time and by stop loss events time. Each order, besides timeout, contains stop loss limit price P_{si} and temporary token balances for source tokens and swapped tokens so far. When a new order is added, the source balance equals to the amount of tokens to swap, the swapped balance is 0.

For every smart contract call, besides claiming an order result, before the operation itself is performed, price recalculation procedure takes place: starting from the saved last recalculation timestamp, calculate time interval length until the earliest timeout expiration time or stop loss event time in the wait-list. Calculate liquidity flow speed for all active orders within the interval, multiply the speed on the interval length to get a partial swap amount for each order, then sum the amounts separately for A/B and B/A orders. Perform a partial deal maintaining $x \cdot y = k$ invariant, taking the pool fee into account; the taken tokens are distributed among the orders propotionally to swap amounts. If the swap price reaches stop loss limit for some order $P \geq P_{si}$, the corresponding order is closed immediately. The source tokens that have not been exchanged yet, are returned to the Trader.

Orders for which the exchange is completed (by timeout expiration or by reaching limit prices), are removed from the wait list and marked as ready to claim.

5.2.3 Off-chain Service

Instant pool price calculation based on the wait list state Instant value of pool exchange price depends on liquidity amounts in the pool. When some time-locked operations are in place the liquidity is not fixed, but is changing continueously as a function of liquidity flow speed and time. Because the smart contract is reactive and cannot perform instant pool price

recalculation without some party performing a transaction and paying gas, Liquifi relies on the back-end service to perform this task.

Liquifi pool smart contracts provide complete pool state information to the back-end service, including current token balances, liquidity flow speed and order wait list. Using this information, the back-end service performs similar calculations that the smart contract does, but without gas price. This allows Liquifi users to always have up-to-date price and pool pressure information.

Time-locked deal price prediction based on built-in simulation model Trader benefit from choosing to perform a time-locked swap operations instead of an immediate swap depends on several factors:

- Current state of order wait list and pool pressure. This is known at the moment when a trader starts a new operation and the effect of this factor can be exactly calculated. For example, if there is already pool pressure in the opposite direction, then a trader will get the best price if properly selects his operation timeout.
- Arbitrage deals. Arbitrageurs may perform counter-deals (including zero-fee deals) in parallel with trader's time-locked operation, making it more profitable. Though it depends on a priori unknown arbitrageurs stategy, this factor can be predicted based on some reference arbitrageur model.
- Other trades. Some traders may want to perform their deals while our time-locked operation is in progress. They can do this not regarding the pool pressure and other factors. Therefore we cannot predict this behavior (though some statistical model may be proposed later, when enough data are captured).
- External markets. Generally we cannot predict price movements on external markets which can have impact on other traders and arbitrageurs behavior. But some probabilistic model may be proposed.

In the current Liquifi implementation we take the first 2 factors into account, i.e. current wait list state and arbitrageurs behavior, and simulate them in our agent-based model (see a description of this model below). The model is built into Liquifi back-end service and provides swap price estimations as a

functional of user input parameters: deal size and operation timeout. Running the simulation on different timeout values allow Liquifi to additionally provide model-based recommendations for traders to select the best timeout value to maximize the expected operation output. Liquifi exchange UI shows these estimations and recommendations to assist traders.

5.2.4 Statistics and historical data

Liquifi collects different types of statistical and historical data by monitoring pool transactions and stores the data off-chain. The Graph protocol will be used provide this

5.2.5 Governance operations

LQF tokens minting schedule. LQF tokens will be minted weekly. After the first week (n = 0) of Liquifi operation a = 2500000 LQF tokens will be minted. From the next week and on the number of LQF tokens minted every week n will be decreased with decay factor r = (250 / 256), $week\_lqf\_minted = a \cdot r^n$ Therefore, LQF minting schedule is an infinite geometric progression and the sum of terms is given by:

$$S = \frac{a}{1-r} = \frac{2500000}{1-\frac{250}{256}} = 106666666,6(6)$$

So the total number of LQF tokens minted will be about 107M.

Determining pool token weights. Only those pools that contain ETH (WETH) in a pair participate in LQF tokens distribution. This limitation is caused by the need of weighing pool shares when calculating an effective user share in LQF distribution. We rely on token ratios in Liquifi pools with ETH to get effective Liquifi pool token (LPTX) prices. When any Liquifi pool detects changes in the ratio, it reports the new value to ActivityMeter contract. ActivityMeter keeps the history of price changes and use these values to calculate input of liquidity providers in ETH equivalent.

Governance tokens minting and distribution among liquidity providers Every week newly minted LQF tokens are automatically distributed among liquidity providers proportionally to their share in the total liquidity amount. To calculate the share of each provider the following algorithm is used:

- Step 1: For each pool find an accumulated pool token price in ETH equivalent multiplied by the time the price held according to the following procedure.
  - At the beginning of a new week, the accumulated price value is set to 0, price modify date is set to the beginning of the week.
  - When a pool reports price change or the week ends, the accumulated price is updated by (pool_token_price_in_eth * price_duration), price modify date is set to the moment of the price change.
- Step 2: For each provider find an accumulated value of LPTX tokens multiplied by the time of the tokens being staked in ActivityMeter during the week according to the following procedure.
  - At the beginning of a new week, the accumulated value is set to 0, modify date is set to the beginning of the week.
  - When a deposit/withdraw operation occurs or the week ends, the accumulated value is updated by (staked_tokens_number * value_duration), modify date is set to the moment of the deposit/withdraw operation.
- Step 3: When the week ends, calculate average pool price for every pool and average LPTX amount for each provider by dividing the accumulated values by week seconds.
- Step 4: Calculate staked liquidity value for each provider by summarizing average_LPTX_amount * average_pool_price for every pool.
- Step 5: When the second week ends, calculate the sum of staked liquidity values for all providers
- Step 6: Number of LQF tokens that a provider gets is (staked_liquidity_value / total_staked_liquidity_value) * week_lqf_minted

Note that shares of all providers must be known to the contract to calculate the final number of LQF tokens to handle. That is why Liquifi requires that liquidity providers claim there LQF tokens every week. If claimed at week n, the claim will return LQF tokens for week (n-2) and calculate the provider's share for week (n-1). If some provider skips a week without claiming, his share will not be accounted in this week distribution. The provider will still

be able to claim his tokens later, but they will be accounted in the week before the week of claim. The provider will get less LQF tokens if he skips a week because his share will not be accounted in the previous week distribution.

Governance process Liquifi community-based governance is performed by submitting proposals and voting for or againts them by LQF token holders. Liquifi supports on-chain and off-chain voting procedures.

On-chain voting is performed within Liquifi governance smart contract. This is the most reliable way of voting and allows to automatically execute some kinds of decisions by the smart contract without human interaction. The downside of on-chain voting are gas costs involved in performing smart-contract transactions. Therefore, Liquifi will use on-chain voting only for several types of decisions: changing pool fees, changing voting parameters (acceptance and quorum thresholds), replacing the governance smart contract itself with a newer version.
On-chain voting procedure is as follows:

- Each governance token holder can initiate change or feature request, providing enough tokens to pass deposit threshold
- The governance tokens used to initiate the request are locked for the duration of decision process
- Each governance token holder can vote for the request
- If decision voting is passed, or no quorum, or all abstain - the locked tokens are returned to the initiator
- If decision voting is rejected - the locked tokens are not returned to the initiator

Note on governance smart contract replacing. Obviously the strongest on-chain governance decision is changing the governance smart contract. This can in the future lead to significant changes in Liquifi governance, including removing the possibility to change the governance smart contract or even disabling on-chain governance at all. Liquifi leaves this possibility taking into account that not every future issue may be foreseen now. As no centralized party, including Liquifi founders, is allowed to perform this without

community voting, we believe that this is a fair tradeoff between governance predictability and flexibility. But nevertheless, there are components that can never be changed. These are LQF token minting algorithm and liquidity providers activity meter. Protecting these mechanisms from change gives LQF token holders guarantee that their tokens will never loose their value as the only method of Liquifi governance.

Off-chain voting will also require possession of LQF tokens, but will not involve smart contract transactions and related gas costs. Instead, Liquifi will rely on Snapshot protocol, which is widely used by DeFi projects, as an off-chain governance implementation. Snapshot protocol will guarantee that only LQF token holders can make proposals and vote for them. Decisions made by off-chain voting may include development of new Liquifi protocol features, selecting development team, etc.

6 Evaluation

6.1 Agent-based simulation

Problem definition We perform simulation modeling to obtain experimental data on how our AMM will perform in conditions close to real. We use simulated Liquifi smart contracts to perform swap and liquidity add/remove operations. We also simulate markets to investigate model behavior when token prices change. We create actor models (liquidity providers, arbitrageurs, traders) to simulate user activity based on the models published in [2].

6.1.1 Market simulation

Liquifi market When performing a swap operation of token B for token A, constant product market model is applied:

$$(R_A - \Delta_A)(R_B + \gamma\Delta_B) = k$$

where R_A, R_B are initial reserves of tokens A and B in the pool, $R_A \cdot R_B = k$, Δ_A, Δ_B are amounts of exchanged tokens, $(1 - \gamma)$ is pool fee.
When a long swap operation with timeout is performed, according to Liquifi AMM model this operation may be split in several smaller operations, allowing for other operations to be performed in parallel.

When adding liquidity to a pool, agent must add a pair of tokens in the same proportion that is currently in the pool. When adding Δ_B tokens B and $R_A\Delta_B/R_B$ tokens A, the agent will be awarded by Liquifi pool tokens

$$\Delta_{LP} = \frac{\Delta_B}{R_B} R_{LP}$$

where R_{LP} is the total amount of outstanding pool tokens given by the contract.

Reference market The reference market follows a simple power law model where the price m_p of some token A, is updated in the following way:

$$m_p \to m_p + \kappa \Delta_A^{1+\xi}$$

where $\kappa \geq 0$ and $\xi \geq 0$ are given in the problem data.
We additionally update the market price every time step (after all agents have completed their actions) in the following way:

$$m_p \to m_p \cdot e^{\sigma X + \mu}$$

where X $N(0,1)$ is drawn from a normal distribution and μ, $\sigma \in R$ represent the mean returns and volatility of the market when no trades are performed.

6.1.2 Agent simulation

Arbitrageurs Arbitrageurs perform trades between the Liquifi market and a reference market when prices differ. They solve the following optimization problem:

$$\begin{gathered} max_{\Delta_A, \Delta_B} \quad m_p\Delta_A - \Delta_B - f(\Delta_A, \Delta_B) \\ (R_A - \Delta_A)(R_B + \gamma\Delta_B) = k \\ \Delta_A, \Delta_B \geq 0 \end{gathered}$$

We use a simle quadratic cost of risk model:

$$f(\Delta_A, \Delta_B) = \frac{\rho_A}{2}\Delta_A^2 + \frac{\rho_B}{2}\Delta_B^2$$

Setting parameters ρ_A, ρ_B we can control the appropriate level of risk for arbitrageurs.
If there is pool pressure that can make the price more profitable for arbitrage, an arbitrageur will wait for a random time interval and then perform the trade.

Liquidity providers We model liquidity providers who begin the simulation by providing some amount of tokens A and B to the pool and seek to gain profits by taking fees from Liquifi trades. We assume these liquidity providers will not withdraw their position until the end of the simulation.

Traders A trader seeks to trade some amount Δ_A token A for some second amount Δ_B of token B (or vice versa), so long as the price of trading tokens on Liquifi differs no more than a constant percentage off from the same trade in the reference market. The trader will draw Δ_A (or Δ_B) from some probability distribution and check if performing this trade in the Liquifi market is at most some percentage more expensive than performing it in the reference market. If not — i.e., if the agent is able to trade with Liquifi for a reasonable price — then the agent makes the trade using the Liquifi contract. Otherwise, no trade is performed.
The trader also picks a random timeout for the swap operation (between 5 and 60 minutes) to perform a long swap and get a better price.

6.1.3 Simulation results

To investigate the behavior of Liquifi market in different conditions we performed several series of simulation experiments with varying reference market price and different states of wait lists in Liquifi smart contract. For each combination of factors we simulated the same sample time-locked swap operation and registered Liquifi price changes as well as the final output of the operation.

Reference market conditions. Using our reference market model, we considered three different cases that can be significant for our model behavior:

1. Upward trend
2. Downward trend
3. Negligible price changes

Wait list conditions. We considered initial wait list configuration with orders that have timeouts higher than an incoming new order and three different pool pressure values:

1. Positive pool pressure, i.e. A to B liquidity flow speed is greater than B to A liquidity flow speed ($S'_A > S'_B$)

2. Negative pool pressure, i.e. A to B liquidity flow speed is less than B to A liquidity flow speed ($S'_A < S'_B$)

3. Zero pool pressure, i.e. A to B liquidity flow speed is equal to B to A liquidity flow speed ($S'_A = S'_B$) or the wait list is empty

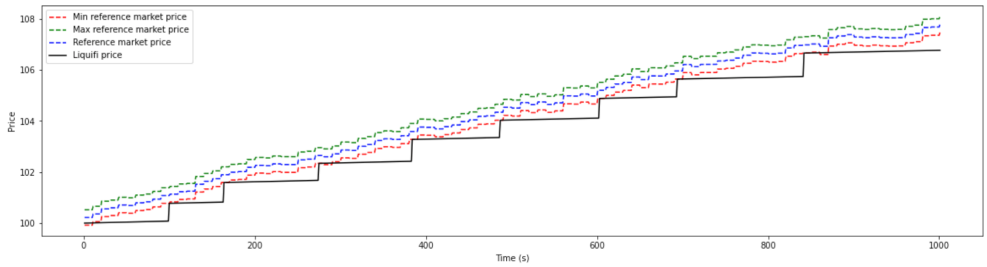

Figure 1: Upward trend with an opposite order

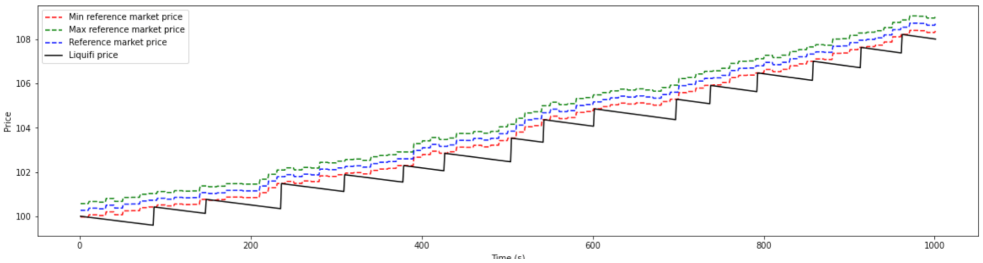

Figure 2: Upward trend with an order in the same direction

Initial data. For each of the simulation experiments we took the following initial parameters:

- Ininital amount of tokens A in the pool $x_0 = 10000$
- Initial amount of tokens B in the pool $y_0 = 1000000$
- Fee $\alpha = 0.003, \gamma = 1 - \alpha = 0.997$

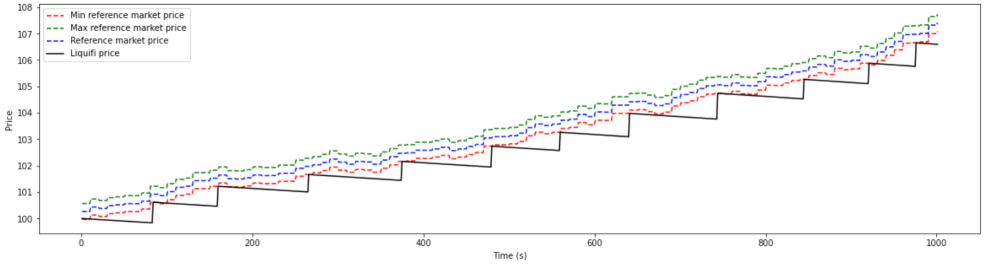

Figure 3: Upward trend with empty wait list

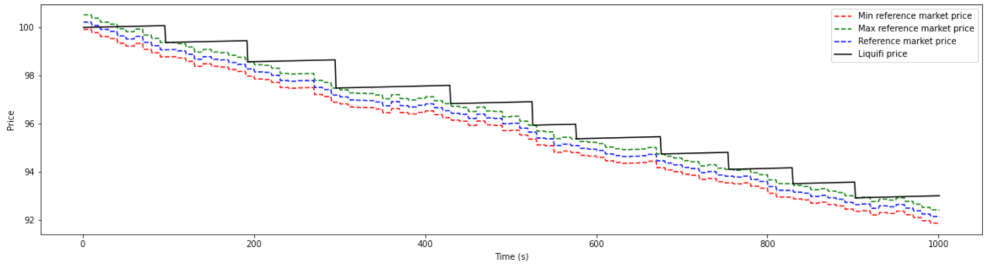

Figure 4: Downward trend with an opposite order

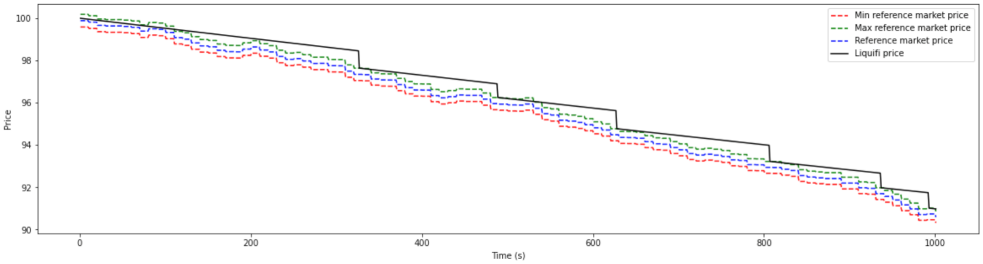

Figure 5: Downward trend with an order in the same direction

- Minimum interest of an arbitrageur to perform a deal $R_{min} = 0.1$ of tokens A
- Sample time-locked swap operation: swap $N_A = 100$ tokens A for tokens B with timeout 1000 seconds
- Number of simulations for each scenario $N = 50$

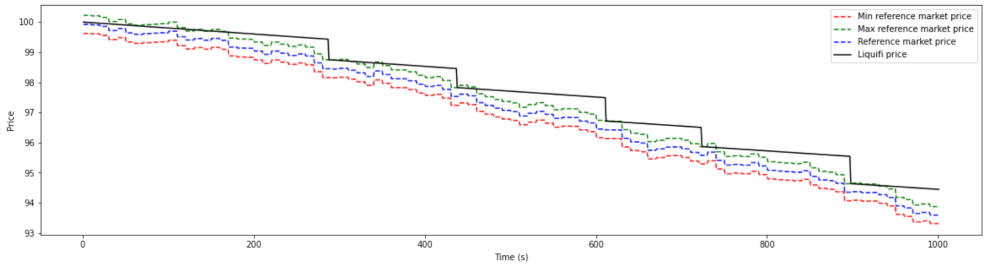

Figure 6: Downward trend with empty wait list

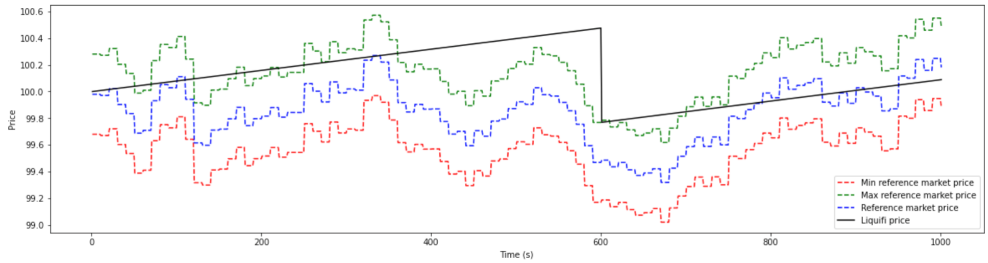

Figure 7: No trend with an opposite order

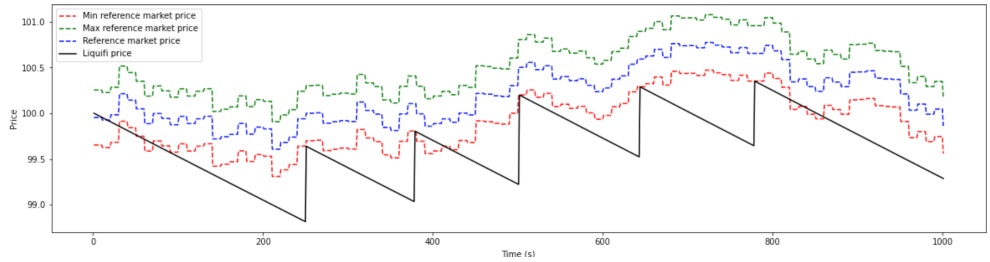

Figure 8: No trend with an order in the same direction

Reference output. As a reference value for expected deal output we take the number of tokens returned by an immediate swap using standard constant product formula:

$$N_{Br} = y_0 - \frac{x_0 y_0}{x_0 + \gamma N_A} = 9871.58$$

Model output.

1. Upward trend and there is an opposite order in the wait list to swap

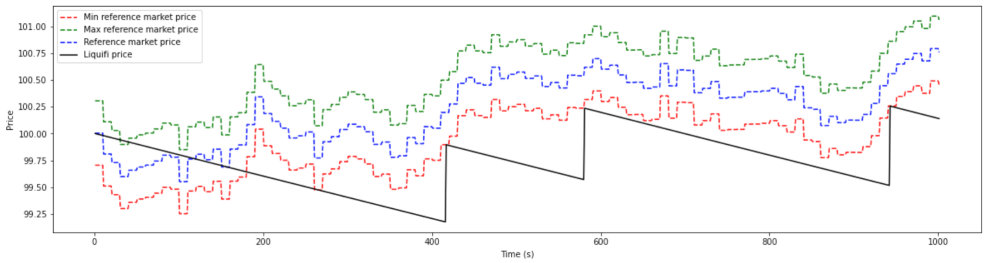

Figure 9: No trend with empty wait list

14000 tokens B for tokens A. Model output: 10320.69.

2. Upward trend and there is an order in the same direction in the wait list to swap 140 tokens A for tokens B. Model output: 10303.04.

3. Upward trend and the wait list is empty. Model output: 10310.54.

4. Downward trend and there is an opposite order in the wait list to swap 14000 tokens B for tokens A. Model output: 9635.18.

5. Downward trend and there is an order in the same direction in the wait list to swap 140 tokens A for tokens B. Model output: 9628.53.

6. Downward trend and the wait list is empty. Model output: 9624.95.

7. No trend and there is an opposite order in the wait list to swap 14000 tokens B for tokens A. Model output: 9987.71.

8. No trend and there is an order in the same direction in the wait list to swap 140 tokens A for tokens B. Model output: 9925.84.

9. No trend and the wait list is empty. Model output: 9953.27.

Dicussion. As we can see, the only situation when the simulated time-locked operation produces less output than an immediate swap is a strong downward trend on a reference market. This could happen when some fundamental factors influence the reference market and therefore time-locked operations are not recommended in this case. In other cases (actually, most of the time) time-locked operations produce better output than immediate swaps. This is true even if there is already pool pressure in the same direction

which negatively influences the price. Arbitrage deals performed in parallel with the time-locked operation compensate these negative price movements.

References

[1] Hayden Adams, Noah Zinsmeister, and Dan Robinson. Uniswap v2 core. `https://uniswap.org/whitepaper.pdf`.

[2] Guillermo Angeris, Hsien-Tang Kao, Rei Chiang, Charlie Noyes, and Tarun Chitra. An analysis of uniswap markets. *Cryptoeconomic Systems Journal*, 2019.

Disclaimer

This paper is for general information purposes only. It does not constitute investment advice or a recommendation or solicitation to buy or sell any investment and should not be used in the evaluation of the merits of making any investment decision. It should not be relied upon for accounting, legal or tax advice or investment recommendations.