SynFutures v3: The Oyster AMM Model for Next-Gen DeFi Derivatives [DRAFT]*

SynFutures Team

info@synfutures.com

April 9, 2024

Abstract

SynFutures v3 is the latest iteration of the decentralized derivatives platform. SynFutures v1 introduced the first permissionless listing and trading of any crypto asset. SynFutures v2 offered a more streamlined, easier-to-navigate user experience for traders and liquidity providers. SynFutures v3 is designed to cater to a larger pool of market participants. The revolutionary Oyster Automated Market Maker, deployed fully on-chain, combines the best of concentrated liquidity and order book in a single model for improved liquidity, enabling everyday traders, professional market makers, and liquidity providers to benefit.

1 Harmonizing Simplicity and Efficiency

Fifteen years after the inception of Bitcoin, decentralized finance (DeFi) has delivered its promises of transparency, permissionlessness, and accessibility. Remarkably, this financial inclusivity has been extended to those seeking and providing services. For instance, Automated Market Makers (AMMs) have democratized activities once exclusive to professionals; market making is now as straightforward as making a deposit. Based on the pioneering work of spot market platforms such as Uniswap and Curve, SynFutures v1 and SynFutures v2 further developed the AMM to enable single token liquidity provision for derivatives. Now, anyone can launch a new derivatives market with a single token for an expansive array of assets, including major cryptocurrencies, altcoins, NFTs, hash rate indices, and, theoretically, any global asset with an available price feed. This innovation has diversified the range of assets and enabled non-professionals to reap yields and fees in a permissionless environment.

However, this simplicity has also sparked criticism. On the one hand, it undermines capital efficiency and, consequently, liquidity depth. On the other hand, it hampers operational efficiency. Less experienced users may struggle to grasp all the rules, risking losses, and even seasoned users may find it challenging to stay updated on every protocol and market movement.

In response to these challenges and after facilitating billions of dollars in trades, SynFutures has increasingly realized that improving the existing financial system does not eliminate the need for professionals. Instead, an optimal system, akin to other natural systems, should be sufficiently decentralized and permissionless to welcome a wider variety of participants, transparent enough to select genuine experts, and accessible enough for everyone to benefit from the services provided by professionals, all without prohibitive intermediary costs or entry barriers.

With this conviction, we present our groundbreaking Oyster AMM model, a comprehensive derivatives market-making model that combines the best of the order book and AMM models on-chain and is

*This document is a draft and is intended for review and comment purposes only. Its content may be subject to change, and it does not represent a final version. Please do not distribute or rely on the information contained herein without consulting the official, final release. Your feedback and suggestions for improvement are welcome and encouraged.

engineered to harmoniously blend simplicity with efficiency and tailored for both novices and experts alike.

2 Oyster AMM: Key Features

The Oyster AMM model builds upon the foundational architecture of SynFutures' sAMM model while taking a significant leap forward to introduce the following features.

2.1 Single-Token Concentrated Liquidity for Derivatives

The Oyster AMM model facilitates liquidity concentration within specific price ranges and incorporates leverage to increase capital efficiency.

Unlike prevalent spot market-focused liquidity models such as Uniswap v3, the Oyster AMM introduces a margin management and liquidation framework tailored specifically for derivatives. Moreover, this model embraces the concept of two-sided liquidity while utilizing just a single token, eliminating the necessity to provide liquidity for both ends of a token pair. This streamlined approach enhances the efficiency of the trading ecosystem, making it even easier for traders to take advantage of SynFutures' permissionless listings' feature established in previous iterations of the protocol.

2.2 A Fully On-chain Order Book

The AMM model democratizes market access, offering automated "market maker" functionality even for niche assets, enhancing diversity. However, this comes at the expense of capital efficiency as AMMs demand significant liquidity for equivalent price impact compared to order book models.

Order book models don't often offer volatile digital assets due to complex infrastructure and risk management concerns. However, they are ideal for capital efficiency, concentrating liquidity around mid-price. With that in mind, we also introduced an order book model in SynFutures v3.

After evaluating off-chain and hybrid alternatives, we chose a fully on-chain approach for the order book, guaranteeing transparency, trustlessness, and anti-censorship. The model promises security and robustness by eliminating dependence on centralized administrators to process orders with the potential for "backdoors" and mitigating vulnerabilities across on-off chain systems, including order management matching and executions in alternative models.

2.3 A Single Model for Unified Liquidity

The Oyster AMM introduces an innovative liquidity paradigm by seamlessly integrating concentrated liquidity and order book in a single model, offering a unified liquidity system tailored to active traders and passive liquidity providers. This cohesive approach ensures traders can enjoy efficient atomic transactions with predictability.

Conversely, in systems that combine on-chain AMM with off-chain limit order systems or separate on-chain limit order systems, trade requests are split between these systems, leading to inefficiencies, non-atomic execution, and unpredictability. This dual-process execution risks non-synchronization, potentially leaving the AMM to process the transaction while the order book system falters, introducing potential confusion if the order book operates off-chain.

2.4 Stabilization Mechanism for User Protection

The Oyster AMM introduces advanced financial risk management mechanisms from past protocol iterations to enhance user protection and price stability. These mechanisms include a dynamic penalty fee

system that discourages price manipulation by imposing penalties for significant deviations between trade prices and mark prices. The dynamic fee system also balances the LP's risk-reward profile.

The other is the stabilized mark prices mechanism, which uses an exponential moving average process to mitigate the risk of sudden price fluctuations and mass liquidations.

3 Single-Token Concentrated Liquidity for Derivatives

3.1 Recap of Previous sAMM Model

The Synthetic Automated Market Maker (sAMM model), the first iteration of SynFutures' AMM model, was introduced in SynFutures v1, enabling LPs to list and add liquidity to a trading pair with a single digital token. Improvements were made in SynFutures v2, including a more streamlined trader and LPs experience, and new methods of liquidity provision to enhance capital efficiency further.

Oyster AMM represents a significant step forward in the evolution of market maker models. Before diving deeper into Oyster AMM, let's evaluate the vanilla constant product model. We use x, y, and k to denote the number of token 0, token 1, and the constant product, respectively. The price of the trading pair of token 0 in token 1 is P and the total value of the tokens added is M. The relations between these variables are as follows:

$$xy = k$$
$$y/x = P$$

For the spot model, there is an additional constraint:

$$xP + y = M$$

When adding liquidity to a spot AMM, the user chooses one of x and y. The other one is implied by the smart contract with current price P.

For derivatives, the introduction of margin increases the dimension of the entire mathematical model. However, the sAMM in previous versions was kept almost identical to the vanilla constant product AMM model to flatten the learning curve and reuse existing tools. You may recall in previous platform iterations the process of adding liquidity from the sAMM to an ETH-USDC and USDC margined pool when the price is at P:

1. The LP adds the total margin of M USDC.
2. $M/2$ The USDC equivalent of ETH-USDC long position is created and added into AMM's long position inventory. (An equal amount of ETH-USDC short position is created and added into LP' s account to balance the market.)

In common AMM terms, $x = M/2P$, $y = M/2$ and they follow $xy = k$. Compared to the vanilla AMM model for spot, the value of the total tokens added are the same (M USDC). However, the token type is different, as $M/2P$ ETH and $M/2$ USDC would be added to a spot AMM. With the same number for x and y in all the AMM-related calculations, all theorems for the spot model are valid for the sAMM.

3.2 Concentrated Liquidity Model and Margin Requirements

Unlike liquidity pools in the sAMM where there is only one AMM for all LPs in a pool, each concentrated liquidity in Oyster AMM is a homogeneous yet independent AMM following its constant product models. Trading volume is allocated based on the liquidity of each concentrated liquidity covering the price range. The discussions below focus on a single concentrated liquidity, which applies to a collection of concentrated liquidity.

To boost capital efficiency, Oyster AMM employs the concentrated liquidity approach. Like concentrated liquidity for the spot market, each liquidity in Oyster AMM only supports trading in a specific price range. For this discussion, we need to introduce extra symbols: use x_{virtual}, y_{virtual} for the implied x and y value for a full range constant product model, respectively. Suppose the current price of the trading pair of token 0 in token 1 is P_c, and the lower price and upper price of the chosen range are denoted as P_a and P_b. Also, the initial margin requirement ratio is denoted as r_i. To reduce the degrees of complexity for LPs and avoid one-sided liquidity resulting in unintended liquidations, Oyster AMM mandates that:

$$P_a = P_c/\alpha,\ P_b = \alpha \cdot P_c,\ \alpha > 1$$

LPs are only required to specify the width of the price range instead of the lower and upper price of the range. Note that the lower and upper prices are symmetric in a geometric sense.

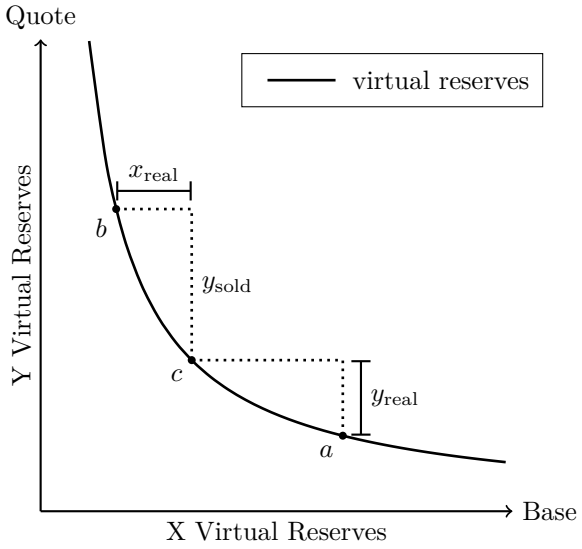

Figure 1: Virtual price curve in Oyster AMM

The main process of adding liquidity to Oyster AMM is the same as the sAMM, where a long position is created (x_{real}) for the liquidity and an offset short position is created for the LP. However, Oyster AMM has a different margin requirement compared to the previous model.

To determine the margin required for a concentrated liquidity of a specific price range, Oyster AMM mandates that the liquidity maintains sufficient margin to meet the initial margin requirement for the resulted net position at both price range boundaries. Note that the margin requirement would be much lower for the case at P_a as the net position value is much smaller due to the lower price. We only need to check the condition at P_b. Based on the constant product formula and the fact x_{real} is the trade size for the price to go from P_c to P_b, it is trivial to derive the relationships below:

$$x_{\text{virtual}} = x_{\text{real}} \cdot \sqrt{\alpha} / \left(\sqrt{\alpha} - 1\right) \tag{1}$$

$$y_{\text{virtual}} = x_{\text{virtual}} \cdot P_c,\ x_{\text{virtual}} \cdot y_{\text{virtual}} = k \tag{2}$$

$$\mathsf{NetPosition}\left(P\right) = x_{\text{virtual}} \cdot \left(\sqrt{P_c/P} - 1\right),\ \text{if } P \in [P_a, P_b] \tag{3}$$

$$\mathsf{PnL}\left(P\right) = -x_{\text{virtual}} \cdot \left(\sqrt{P_c} - \sqrt{P}\right)^2,\ \text{if } P \in [P_a, P_b] \tag{4}$$

Considering this extreme case and ignoring any potential fee income, the margin requirement translates to:

$$M + \mathsf{PnL}\left(P_b\right) = |\mathsf{NetPosition}\left(P_b\right)| \cdot P_b \cdot r_i$$

Using Equation (1), (3) and (4), we have:

$$M = x_{\text{virtual}} \cdot \left(\sqrt{P_c} - \sqrt{P_b}\right)^2 + x_{\text{virtual}} \cdot \left(1 - \sqrt{P_c/P_b}\right) \cdot P_b \cdot r_i = x_{\text{real}} \cdot P_c \cdot \left(\alpha \cdot (1 + r_i) - \sqrt{\alpha}\right)$$

Or,

$$x_{\text{real}} = \frac{M}{(P_c \cdot (\alpha \cdot (1 + r_i) - \sqrt{\alpha}))} \tag{5}$$

Now, the main process of adding liquidity to Oyster AMM can be described. Let's use an ETH-USDC and USDC-margined pool as an example:

1. At the current price P, the LP chooses α and adds a total margin of M USDC.
2. $\frac{M}{\left(P \cdot \left(\alpha \cdot (1+r_i) - \sqrt{\alpha}\right)\right)}$ of ETH-USDC long position is created and added into AMM's long position inventory. (An equal amount of ETH-USDC short position is created and added into the LP's account to balance the market.)
3. This concentrated liquidity will provide pricing from P/α to αP.

As the AMM price deviates from the price when concentrated liquidity is created, that liquidity would inherently have an implied net position, which can be calculated from Equation (3). Each concentrated liquidity will always meet the margin requirement for the implied net position within its chosen price range. LPs are allowed to remove the liquidity from their concentrated liquidity anytime, and that is converted into a trading position the size of the implied net position and the remaining margin, along with any fees earned.

If the AMM price goes out of the price range of concentrated liquidity, anyone could initiate the removal of that liquidity and convert that into a trading position, which goes to the account of the original owner of the concentrated liquidity. LP can then manage the trading position accordingly after concentrated liquidity is removed and converted.

3.3 Capital Efficiency Boost

The most common definition of capital efficiency is to compare the value of assets required for the same trade size and the slippage. That is equivalent to comparing the M required to result in the same x_{virtual} in Oyster AMM and x in the sAMM, as these two are used in the same constant product formula. By using Equation (1) and (5), we can express them as:

$$\mathsf{CapitalEfficiencyBoost} = \frac{x \cdot P_c \cdot 2}{x_{\text{virtual}} \cdot P_c \cdot (\alpha \cdot (1 + r_i) - \sqrt{\alpha}) \cdot \frac{\sqrt{\alpha}-1}{\sqrt{\alpha}}} = \frac{2}{((\sqrt{\alpha} \cdot (1 + r_i) - 1) \cdot (\sqrt{\alpha} - 1))} \tag{6}$$

When we plug in some α numbers, the following occurs in Table 1. Please note that the smart contract restricts the choice of α to be at least $1 + r_i$.

Based on discussions in Section 3.1 and 3.2, it is trivial to see single sided spot concentrated liquidity falls into the case when $r_i = 100\%$. Indeed, when comparing the publicized boosts from Uniswap v3, we can see similar results.

When comparing with two-sided spot concentrated liquidity, it is easy to see Oyster AMM would be twice as efficient as spot models, as Oyster AMM only requires one set of margins to provide liquidity for both sides. In summary, Oyster AMM has a superior capital efficiency boost compared to spot concentrated liquidity models.

However, a capital efficiency boost comes with higher risks for LPs, especially in derivatives market settings where leverage further amplifies the risks. These risks can be difficult to grasp with the overlay of leverage and capital efficiency boost. So, beyond the numbers, the mechanism must be clear concerning these risks so that LPs know what they are doing.

Table 1: Capital Efficiency Boost

α	$1/\alpha,\ \alpha$	Capital Efficiency Boost			
		IMR = 10%	IMR = 5%	IMR = 3%	IMR = 1%
1.01	99.0%, 101.0%	–	–	–	26,666.6x
1.03	98.0%, 103.0%	–	–	2,962.9x	5,364.9x
1.05	95.2%, 105.0%	–	1,066.6x	1,460.9x	2,317.8x
1.10	90.9%, 110.0%	266.6x	404.7x	510.5x	691.0x
1.20	83.3%, 120.0%	102.2x	139.5x	163.3x	196.9x
1.30	76.9%, 130.0%	56.1x	72.4x	81.8x	94.1x
1.50	66.7%, 150.0%	25.6x	31.1x	34.0x	37.5x
2.00	50.0%, 200.0%	8.7x	10.0x	10.6x	11.3x
3.00	33.3%, 300.0%	3.0x	3.3x	3.5x	3.6x
5.00	20.0%, 500.0%	1.1x	1.2x	1.2x	1.3x

Table 2: Capital Efficiency Boost Comparison Across Models

Model	Price Range	Capital Efficiency Boost
Oyster Amm	99.99%, 100.01%	39,997.0x
Uniswap v3	99.99%, 100.00%	40,002.5x
Uniswap v3	100.00%, 100.01%	39,998.5x
Uniswap v3	99.99%, 100.01%	20,000.5x

3.4 Implementation Highlights

The smart contract implementation describes concentrated liquidity in a struct called Range. It is stored in the smart contract with an index composed of pair expiry, liquidity owner's address, price range start, and price range end. The struct definition and index information define concentrated liquidity in the smart contract entirely and uniquely.

```
struct Range {
    uint128 liquidity;        // uniswap v3 style liquidity
    uint128 entryFeeIndex;    // snapshot of the global fee index at the time of entry
     uint96 balance;          // margin supplied
    uint160 sqrtEntryPX96;    // AMM price at the time of entry
}
```

To add liquidity, a user needs to send a transaction on-chain specifying the pair to add liquidity, the price range to provide liquidity, and the margin amount for this concentrated liquidity. The smart contract then creates a Range struct for the user with $liquidity = \sqrt{k} = \sqrt{x_{\text{virtual}} \cdot y_{\text{virtual}}}$, as calculated with Equation (1), (2) and (5).

The smart contract then stores this Range struct in this user's account for this pair with an index composed of price range start and price range end. Based on Equation (3) and (4), the state of a Range struct can be implied using the data stored in the struct, its index, and the current AMM price at any point in time.

4 Fully On-chain Order Book

4.1 Native Irreversible Limit Order

In Oyster AMM, the buy and sell limit orders on single price points are combined with concentrated liquidity to provide pricing for traders. Unlike in some spot AMM models, where highly concentrated liquidity is used as a proxy of limit orders, Oyster AMM enables native limit orders like those in central limited order book systems. These are defined on single price points and irreversible once filled. The design provides makers with certainty of the status of their limit orders.

According to the definition of capital efficiency in Equation (6), one could easily conclude that limit orders provide an infinite capital efficiency boost, if the taker's trade size does not exceed the limit orders at that price point. This is impossible in a constant product AMM model, where an infinitesimal trade would still result in non-zero slippage. In other words, infinite capital is required for a constant product AMM to produce no slippage for any trade size.

4.2 Matching and Overall Pricing Mechanism

To circumvent current smart contract limitations, order matching in Oyster AMM does not follow the traditional centralized limit order book model, which is the first-in-first-out principle. Rather it follows a matching process more suitable for an AMM, which will be formally described in Section 5. The characteristics of the matching and pricing mechanism can be summarized below:

1. At a price point where limit orders exist, they are filled before any concentrated liquidity is consumed.
2. Trade volume is allocated to multiple limit orders proportionately at the same price. A just-in-time limit order is welcomed.
3. Overall slippage can be greatly reduced when limit orders exist on the AMM price curve.

4.3 Implementation Highlights

In the smart contract implementation, the limit order is described in a struct called Order and is stored in the smart contract with an index composed of pair expiry, limit order owner's address, and limit price. Below, struct definition and index information completely and uniquely define a limit order in the smart contract.

```
struct Order {
    uint128 balance;    // margin supplied
    int128 size;        // positive number as buy order, negative number as sell order
}
```

To create a limit order, a user must send a transaction on-chain specifying the pair for this order, the limit price, the order size, and the margin amount. The smart contract then creates an Order struct for the user and stores it in the user's account for this pair with an index composed of the limit price and the nonce of this Order. The nonce is a system-generated number indicating the version of limit orders at this price. The native limit order in Oyster AMM is designed and implemented asynchronously.

5 A Single Model for Unified liquidity

Oyster AMM unifies both concentrated liquidity and limit orders in a single model. In smart contract terms, Range and Order can provide liquidity at each price point. For Range covering the same price point, their liquidity, or square root of k, is added for AMM curve-related calculation. For Order at the

same price point, their order sizes are added together to cater to the taker's trade size. In addition, Order is always consumed before liquidity from Range is consumed.

```
struct Pearl {
    uint128 liquidityGross;         // total range liquidity that references this pearl
    int128 liquidityNet;            // amount of net liquidity when the tick is crossed

    uint24 nonce;                   // nonce for record version control
    int96 left;                     // orders to be taken
    int96 taken;                    // orders to be filled

    uint128 fee;                    // should only be distributed w.r.t. filled order
    uint128 entrySocialLossIndex;   // social loss index borne by taken but unfilled order

    int128 entryFundingIndex;       // funding index owned by taken but unfilled order
}
```

The collection of concentrated liquidity covering a price point and all open limit orders on the same price point is described in a struct called Pearl and is stored in the smart contract indexed by price. Oyster AMM can be viewed as the collection of Pearl along with the AMM price curve. Among all the Pearl fields, liquidityGross and liquidityNet are related to Range and the rest are related to Order.

Introducing native irreversible limit orders in Oyster AMM eliminated the need to create fully flexible concentrated liquidity of an extremely narrow range to mimic a limit order. Consequently, the fee distribution mechanism in Oyster AMM is significantly simplified compared to other spot-concentrated liquidity DEXes. This update drastically reduces engineering complexity and makes implementing Oyster AMM in smart contracts possible.

In fact, Pearl serves as the pool for limit orders from makers, which is also the key to the feasibility of the asynchronous design of limit orders in Oyster AMM. In this way, logic on the takers' side is simplified greatly, where takers take as much as they want and nothing more. With the fungible approach of Pearl, the gas cost of a trade is directly proportional to the price impact, i.e., the number of ticks crossed.

For a trade of size S_0, the process of trading, or consuming unified liquidity, follows the steps below and is shown in Figure 2.

1. In the Pearl of current price P_0, check if there are unfilled limit orders.
   (a) If not, go to step 2 with $S_1 = S_0$.
   (b) If so, fill the limit orders as much as possible.
      i. If S_0 is fully filled, terminate. (Note that the current price does not change in this case.)
      ii. If not, continue to step 2 with the remaining size S_1.
2. Find the Pearl at the next price P_1.
   (a) Trade for size S_1 on the AMM curve between Pearl at P_0 and P_1.
      i. If S_1 is fully filled, terminate. (Note that the current price does change in this case.)
      ii. If not, update the current price as P_1 and go to step 1 with the remaining size S_2.

6 Stabilization Mechanism for User Protection

6.1 Stabilized Mark Prices

As in previous versions of SynFutures models, Oyster AMM maintains a few prices for different purposes.

- P_{fair}: Price quoted by the Oyster AMM by combining all the concentrated liquidity and limit orders, like the mid-price of a CLOB system. This price is used to determine if concentrated liquidity is out of its price range. This price is also used when a concentrated liquidity is converted into a trading position. Simply put, this price is used for all liquidity-related calculations.

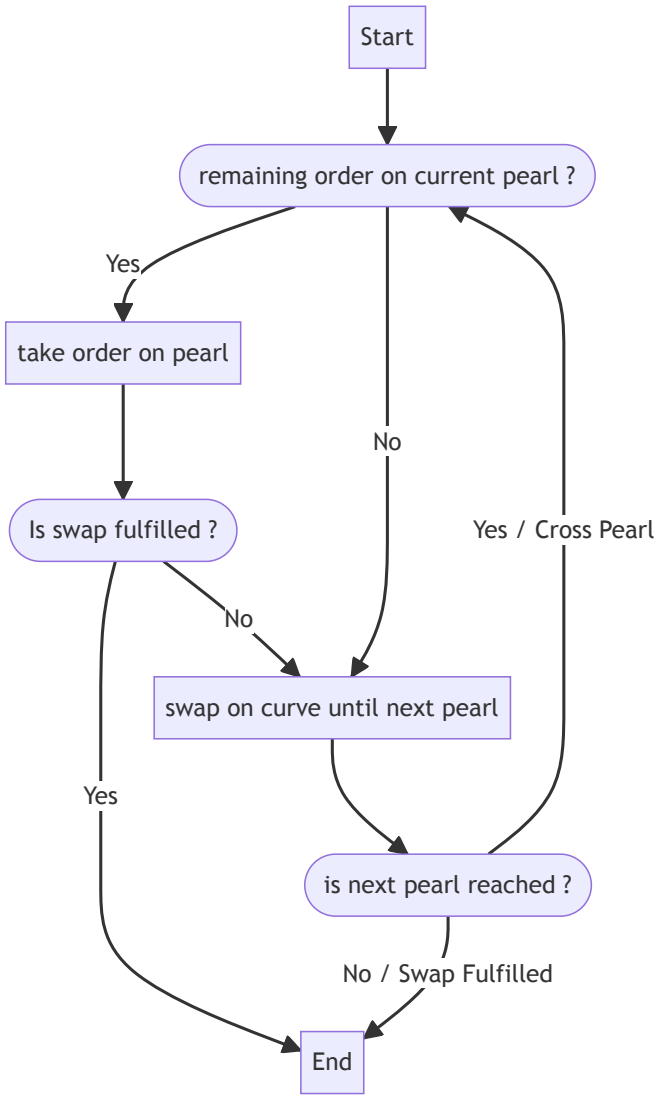

Figure 2: Overview of Swap Procedure

- P_{mark}: Price used to determine the unrealized profit and loss and margin requirements of all positions in a pair. This price is also used to determine the initial margin requirement to open a position. This is the price for position risk management. Note that unrealized profit is not available for withdrawal until it is realized, i.e., when the position is closed.
- P_{spot}: Price originated from oracles such as Chainlink or various spot DEX markets but undergoes an exponential moving average process to smooth the fluctuation. This price is also subject to a change rate limit for safety reasons.
- $P_{\text{settlement}}$: Time weight average price from oracles such as Chainlink or various spot DEX markets, used for final settlement of all positions in a dated futures market.

Just as the derivative markets are related to the spot markets but function independently, these prices are closely related and can even be the same number, though they are often different. P_{fair} solely depends on the trading and liquidity of the pool, while P_{spot} and $P_{\text{settlement}}$ solely depend on spot market data and time. P_{mark} is based on P_{spot} but also incorporates a daily interest component of the underlying trading pair. With the segregation of duty of trading and marking, the pricing mechanism in Oyster AMM shields all users from attacks, such as flash loan and price manipulation, refer to Figure 3.

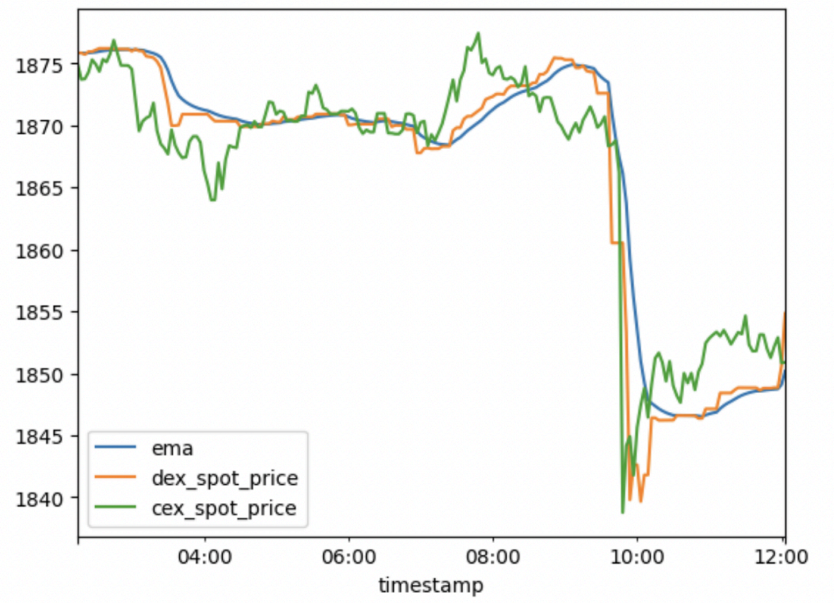

Figure 3: An illustration of the smoothing effect of the EMA on the DEX spot prices

6.2 Continuous Funding Fee

As perpetual futures do not have a final settlement at the maturity to guarantee the convergence to the spot market index price, Oyster AMM employs continuous funding for its perpetual futures markets. The principle is to expect the deviation to converge in a daily cycle, and thus, the funding fee ratio for a period is defined below:

$$\mathsf{FundingFeeRate} = \left(P_{\text{fair}} - P_{\text{spot}}\right) / P_{\text{spot}} \cdot \Delta t / 86400$$

Δt is the time difference in seconds between the current timestamp and the last timestamp when the funding fee rate is calculated. Like other centralized and decentralized perpetual futures markets, the funding fee encourages traders to return the perpetual futures price to the index price by continuously rewarding positions against such deviation and punishing positions in favor of such deviation.

6.3 Dynamic Penalty Fees

In a functional derivatives market, it is common for P_{fair} to deviate from P_{spot} depending on the liquidity and market situation. However, excessive deviation and unreasonable trading behavior should be discouraged through economic means. The definition of P_{mark} reflects this principle.

To protect our LPs and limit order makers while not impacting normal trading behavior, Oyster AMM employs a stability penalty mechanism. When a trade results in a higher deviation of P_{fair} to P_{mark} than before the trade, a stability penalty will be charged based on the following, in addition to the normal trading fees paid to LPs or limit order makers.

$$\mathcal{D}\left(P_{\text{fair}}, P_{\text{mark}}\right) = \frac{\max\left(P_{\text{fair}}, P_{\text{mark}}\right)}{\min\left(P_{\text{fair}}, P_{\text{mark}}\right)} - 1$$

$$\mathsf{StabilityPenaltyRatio}\left(P_{\text{fair}}\right) = a \cdot \mathcal{D}^3 + b \cdot \mathcal{D}^2 + c \cdot \mathcal{D} + d$$

Trades that result in deviation less than $0.5 \cdot r_i$ would have no stability penalty. Trades that result in deviation above $0.5 \cdot r_i$ would have an increasing stability penalty ratio.

6.4 Liquidations

When the margin supports a trading position that falls below its maintenance margin requirement, that position is subject to liquidation. Like in most decentralized margin trading systems, liquidation in Oyster AMM is performed primarily via the taking-over approach, where the liquidator takes over the target trading position along with the remaining margins and tops up the margin to meet the initial margin requirement.

Effectively, the remaining margin of the trading position to be taken over is a potential profit for the liquidator—that is, if the liquidator can adequately manage the risk. In addition, Oyster AMM also allows a liquidation mechanism where trading positions failing the maintenance margin requirement are forced to trade against the Oyster AMM directly to close the position. Both approaches support partial liquidation, where the initiator specifies the amount of position to be taken over or forcibly closed.

DISCLAIMER

This document serves solely for informational purposes. It is not intended as investment guidance, a suggestion, or an invitation to engage in the buying or selling of any products or investments. It should not be employed for assessing the appropriateness of any investment choices, nor should it be considered a source of accounting, legal, or tax counsel, or investment suggestions. The views expressed in this document are those of the authors at the time of publication and are not made on behalf of SynFutures. They may not necessarily represent the perspectives of SynFutures or individuals associated with the project. The expressed opinions are subject to change without notice or updates.